

Sakarya University

Journal of Computer and Information Sciences

e-ISSN 2636-8129

VOLUME 7 ISSUE 2

AUGUST 2024



VOLUME: 7 ISSUE: 2
E-ISSN 2636-8129

AUGUST 2024
<http://saucis.sakarya.edu.tr/tr/>

**SAKARYA UNIVERSITY
JOURNAL OF COMPUTER
AND
INFORMATION SCIENCES**



SAKARYA
ÜNİVERSİTESİ

The Owner on Behalf of Sakarya University

Prof. Dr. Hamza Al
Sakarya University, Sakarya-Türkiye

Editor in Chief

Ahmet Zengin
Network and Communication, Computer System Software
Sakarya University
Sakarya - Türkiye
azengin@sakarya.edu.tr

Associated Editors

Hessam Sarjoughian
Artificial Intelligence, Computer System Software
Computer Software
Arizona State University
United States

Muhammed Fatih Adak
Informatics and Computer Science, Machine Vision
Big Data, Computer Software
Sakarya University
Sakarya - Türkiye

Muhammed Kotan
Image Processing, Artificial Intelligence, Natural
Language Processing
Sakarya University
Sakarya - Türkiye

Ünal Çavuşoğlu
Information and Computer Sciences, Algorithms and
Theory of Computation
Sakarya University
Sakarya - Türkiye

A F M Suaib Akhter
Information Security Management, Network and
Communication,
Concurrent/Parallel Systems and Technologies
Database Systems
Sakarya Applied Sciences University
Sakarya - Türkiye

Selman Hızal
Information and Computer Sciences, Image Processing,
Networking and Communications
Sakarya Applied Sciences University
Sakarya - Türkiye

Mustafa Akpınar
Information and Computing Sciences, Information Systems
Higher Collages Of Technology
United Arab Emirates

Editorial Board

Aref Yelghi
Information and Computer Sciences
Istanbul Ayvansaray University
Istanbul - Türkiye

Kamal Z Zamli
Information and Computer Sciences
University of Malaysia Pahang
Malaysia (Ump)

Priyadip Ray
Decision Support and Group Support Systems
Machine Learning
Lawrence Livermore National Laboratory
United States

Nejat Yumuşak
Information and Computer Sciences, Pattern
Recognition
Sakarya University
Sakarya - Türkiye

Ayhan İstanbullu
Digital Processor Architectures, Digital Design
Balıkesir University
Balıkesir-Türkiye

Bahadır Karasulu
Information and Computer Sciences, Image Processing
Çanakkale Onsekiz Mart University
Çanakkale - Türkiye

Cihan Karakuzu
Information and Computing Sciences, Neural
Networks, Machine Learning (Other)
Bilecik Şeyh Edebali University
Bilecik - Türkiye

İbrahim Türkoğlu
Pattern Recognition, Bioinformatics, Artificial
Intelligence, Embedded Systems
Fırat University
Elazığ - Türkiye

Nuri Yılmaz
Information and Computer Sciences
Texas University, Kinsville
Tx-United States

Resul Daş
Software Engineering Department
Fırat University
Elazığ-Türkiye

Okan Erkaymaz
Department of Computer Engineering
Ministry of National Defense
Ankara - Türkiye

Ömer Hulusi Dede
Ecology, Industrial Biotechnology, Ecological
Applications
Sakarya Applied Sciences University
Sakarya - Türkiye

Editorial Assistants - Secretary

Deniz Balta
Information and Computer Sciences, Knowledge
Representation and Reasoning
Sakarya University
Sakarya - Türkiye

Gözde Yolcu Öztel
Information and Computer Sciences, Image Processing
Artificial Intelligence
Sakarya University
Sakarya - Türkiye

İbrahim Delibaşoğlu
Image Processing, Machine Learning, Artificial
Intelligence, Computer Software
Sakarya University
Sakarya - Türkiye

Sümeyye Kaynak
Information and Computer Science, Deep Learning
Sakarya University
Sakarya - Türkiye

Fatma Akalın
Image Processing, Data Mining and Knowledge
Discovery
Sakarya University
Sakarya - Türkiye

Nur Yasin Peker
Information and Computer Sciences, Image Processing
Sakarya Applied Sciences University
Sakarya - Türkiye

Language Editors

A F M Suaib Akhter
Information Security Management, Network and Communication
Sakarya Applied Sciences University
Sakarya - Türkiye

Layout Editor

Mehmet Emin Çolak
Scientific Journals Coordinatorship
Sakarya University
Sakarya-Türkiye
mehmetcolak@sakarya.edu.tr

Yakup Beriş
Scientific Journals Coordinatorship
Sakarya University
Sakarya-Türkiye
yakupberis@sakarya.edu.tr

Indexing








Applied Science &
Technology Source

Contents

Research Article

1	Comparative Analysis of Machine Learning Techniques for Prediction of the Compressive Strength of Field Concrete <i>Ajibola Oyedepi, Adekunle David, Ositola Osifeko, Abisola Olayiwola, Omobolaji Opafola</i>	127-137
2	The Importance of Rhythm Activity in Epilepsy EEG Signal Classification (An Educational Article) <i>Negin Melek</i>	138-155
3	Prediction of Multivariate Chaotic Time Series using GRU, LSTM and RNN <i>Gülyeter Öztürk, Osman Eldoğan</i>	156-172
4	Optimization Planning Techniques with Meta-Heuristic Algorithms in IoT: Performance and QoS Evaluation <i>Murat Koca, İsa Avcı</i>	173-186
5	Distributed Task Allocation for UAV Swarms with Limited Communication <i>Mutullah Eşer, Asım Egemen Yılmaz</i>	187-202
6	Network Forensics Analysis of Cyber Attacks Carried Out Over Wireless Networks Using Machine Learning Methods <i>İmran Kaçan, Batuhan Gül, Fatih Ertam</i>	203-216
7	Classification of Solar Cells EL Images with Different Busbars Via Deep Learning Models <i>Miklat Aktaş, Ferdi Doğan, İbrahim Türkoğlu</i>	217-226
8	Performance Assessment of Natural Survivor Method-Based Metaheuristic Optimizers in Global Optimization and Engineering Design Problems <i>Hüseyin Bakır</i>	227-243
9	Performance Analysis of Open Source Cloud Computing Architectures <i>Mehmet Zahid Kuzan, Abdullah Sevin</i>	244-252
10	Cigarette Detection in Images Based on YOLOv8 <i>Yerniyaz Bakhytov, Cemil Öz</i>	253-263
11	Detection and Analysis of Malicious Software Using Machine Learning Models <i>Ahmet Öztürk, Selman Hızal</i>	264-276
12	Design and Implementation of Remote Lab PID Controller Experiment Based on IoT <i>Mohammed Eishorafa, Emin Güney, İhsan Pehlivan, Cüneyt Bayılmış</i>	277-288
13	Evaluating Feature Selection Algorithms for Machine Learning-Based Musical Instrument Identification in Monophonic Recordings <i>İsmet Emre Yücel, Ulaş Yurtsever</i>	289-301
14	Predictive Model for Incipient Faults in Oil-Filled Transformers <i>Michael Osajeh, Efosa Igodan, Linda Usiosefe</i>	302-313
15	Investigation of Digital Calibration Certificate - Digital Test Report Sharing in Metrology Network <i>Erkan Danacı, Bülent Aydemir</i>	314-324

Comparative Analysis of Machine Learning Techniques for Prediction of the Compressive Strength of Field Concrete

Ajibola Oyedeji¹, Adekunle David², Ositola Osifeko¹, Abisola Olayiwola¹, Omobolaji Opafola²

¹Department of Computer Engineering, Olabisi Onabanjo University, Ago-Iwoye, Nigeria

²Department of Civil Engineering, Olabisi Onabanjo University, Ago-Iwoye, Nigeria

Corresponding author:

Ajibola Oyedeji, Department of Computer Engineering, Olabisi Onabanjo University, Ago-Iwoye, Nigeria
oyedeji.ajibola@oouagoiwoye.edu.ng

Article History:

Received: 18.01.2024

Accepted: 10.07.2024

Published Online: 23.08.2024

ABSTRACT

The determination of the concrete compressive strength remains a challenging task in the concrete industry. Machine learning (ML) algorithms offer an alternative and this study presents a comparative analysis of five ML regression models; Gradient Boosting (GB), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), and Linear Regression (LR) on a dataset of 1030 concrete samples. The findings indicate that the GB model achieved the best performance. The developed GB model achieved R-squared values of 91.60%, 91.43%, and 90.18% for the 10-fold, 5-fold, and 3-fold cross-validations, respectively, with mean absolute error, root mean squared error, and mean absolute percentage error values of 2.6776, 4.3523, and 9.19%, respectively. The GB model trained and evaluated was deployed to a web application using Streamlit for real-time prediction of the concrete compressive strength. The results of this research offer a precise and practical method for judging the quality of concrete constructions.

Keywords: Machine learning, Concrete compressive strength, Prediction, Regression models, Web application

1. Introduction

The compressive strength of concrete (CCS) is an important factor for construction experts' consideration. Concrete is typically made up of cement, fine aggregates, coarse aggregates, and water, with or without the addition of chemical admixtures. CCS at 28 days is essential for designing reinforced concrete constructions [1]. Traditionally, concrete mixture portions that satisfy the minimum 28-day curing period requisite have been determined using empirical prescriptive and performance-based mixture design techniques. However, recent literature has started to explore numerical methods for predicting the CCS after 28 days. Accurate prediction of the 28-day CCS is crucial because it (a) ensures concrete quality, (b) reduces the amount of testing that needs to be conducted on different concrete batches to achieve desired and required strength goals, and (c) enhances safety.

Recent computer studies have established the potential of modern statistical modeling methods or approaches to numerically forecast the CCS for laboratory-mixed concrete, referred to as laboratory concrete [2]–[5]. Therefore, these approaches can provide a means of accurately estimating the 28-day CCS, leading to better concrete quality and improved safety. However, despite these efforts, various uncertainties encountered during the mixing, transport, placement, curing, and finishing of concrete can make it difficult to achieve accurate predictions. These uncertainties are particularly challenging in the case of field concrete, which refers to concrete produced and placed at the construction site. In particular, the environmental conditions that can fluctuate during the production and placement of field concrete, as well as the numerous factors that can affect the quality and performance of the final product, make it difficult to achieve consistent and reliable predictions of the 28-day CCS.

Concrete's 28-day compressive strength estimation is a complex issue. CCS is influenced by the intricate physical and chemical interactions that take place between the components of concrete. Research has shown a nonlinear relationship between the CCS and its air content, which is often increased to improve workability and freeze-thaw resistance [6], [7]. As the water-to-cement ratio is increased, the CCS tends to decrease. However, other factors can affect the CCS that are not as easily identified. For example, the ratio of coarse to fine aggregate, the interfacial bindings between the coarse aggregate and mortar influence compressive strength [8]–[10]. Likewise, field concrete exhibits far greater variability in compressive

strength due to the highly varied job site conditions where it is mixed and applied. Here, the final CCS can be impacted by temperature, humidity, and bad weather [11].

While traditional methods of CCS estimation rely on time-consuming and expensive laboratory tests, machine learning algorithms offer an option for predicting concrete strength from non-destructive testing data. There is an increasing research focus on the application of machine learning (ML) models for concrete mixtures [12], [13]. These models utilize the different mix proportions to predict the CCS, the target variable [14]. The CCS was predicted by making use of ML methods such as decision tree (DT), gradient boosting (GB), and bagging regressor (BR) [15]. The BR technique was the best performing methods achieving the highest R-squared value of 0.92. The mean absolute error (MAE) and root mean square error (RMSE) values for the BR model were 4.26 and 5.69 respectively, which was the lowest compared to the other models trained; MAE = 4.96 and RMSE = 7.05 for GB and MAE = 6.39 and RMSE = 8.95 for DT. A prediction of CCS with electrical resistivity included as an input variable saw the performance of the ML models increase as against the traditional proportion mix as input variables [16]. The models performances were increased as the R-squared values of the decision trees and Gaussian models increased from 0.77 to 0.79 and 0.81 to 0.82 respectively.

Using 1030 data samples from concrete compressive strength tests obtained from the University of California, Irvine, a novel Hybrid Ensemble Model (HENSM) was the best predictor of the CCS in the study [17]. Furthermore, the adapted AdaBoost achieved an accuracy of 98% on the same dataset, outperforming the artificial neural network (ANN) and support vector machine (SVM) [18]. Likewise, ANN has succeeded in surpassing the performance of the LR, SVM and test ensembles of the models by achieving an R-value of 0.909 [19]. A proposed deep convolutional neural network (DCNN) model is trained using a data set consisting of 380 groups of concrete mixes. The results show that the DCNN achieves high coefficients of determination of 0.967 which is better than the performance of the SVM, ANN, and AdaBoost [20].

This paper aims to predict the concrete compressive strength using 5 ML methods and evaluate the performance of the models. The input variables used for predicting the Concrete Compressive Strength (CCS) are cement (C), Blast Furnace Slag (BFS), Fly Ash (FA), Water (W), Superplasticizer (SP), Coarse Aggregate (C_Ag), Fine Aggregate (F_Ag), all measured in kg/m^3 , and Age (in days). The machine learning models for the regression problem used in our study are Linear Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), and Gradient Boosting Regression (GB). The models have been selected among the many available models to study and compare different methods or approaches including the linear statistical approach, tree method, ensemble methods and distance-based models. The result of this study is of great importance to the cost-effective method of determining the strength of concrete structures capable of producing quality concrete structures, improving safety, and reducing maintenance costs over the lifetime of these structures. Furthermore, a web application has been deployed for easy interaction and prediction of the CCS based on the various inputs for real-time use.

The paper is presented as follows; the materials and methodology are presented in Section 2, section 3 contains the results and discussion and the conclusion is in Section 4.

2. Materials and Method

For this research, the following steps were followed; data collection, data preprocessing and analysis, model development, testing and evaluation, and model deployment. Figure 1 shows the process flow diagram for the listed steps.



Figure 1. Concrete Compressive Strength Prediction Process Flow Diagram

2.1. Data Collection and Analysis

The Concrete Compressive Strength dataset used in this research project has been retrieved from the UCI Machine Learning Repository available at <https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength> [21], [22]. The data used was curated by experiments from 17 different sources. Data was assembled for concrete containing Portland cement plus fly ash, blast furnace, and superplasticizer cured under normal conditions. The details of the experimental setup are presented in [21]. The dataset is acceptable as collecting and testing the data met accepted guidelines in the concrete industry.

The data set is of the form $\mathcal{D} = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where $x_i \in X$ is the i^{th} input and $y_i \in Y$ the corresponding target. $X = \mathbb{R}^d$, in which case $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ is a d -dimensional vector called an instance. The data set includes 1030 instances and 9 quantitative attributes including 8-input variables and 1-output or target variable. After analyzing the data collected, 25 instances were duplicated and the duplicate records were dropped remaining total of 1005 instances. The dataset also does not contain any missing values. The characteristics and summary statistics of the data collected are shown in Table 1.

Table 1. Summary Statistics of Attributes

Attribute	Abbreviation	Mean	Standard Deviation	Min	25th Percentile	50th Percentile	75th Percentile	Max
Cement (kg/m^3)	C	278.63	104.34	102	190.7	265	349	540
Blast Furnace Slag (kg/m^3)	BFS	72.04	86.17	0	0	20	142.5	359.4
Fly Ash (kg/m^3)	FA	55.54	64.21	0	0	0	118.3	200.1
Water (kg/m^3)	W	182.08	21.34	121.8	166.6	185.7	192.9	247
Superplasticizer (kg/m^3)	SP	6.03	5.92	0	0	6.1	10	32.2
Coarse Aggregate (kg/m^3)	C_Ag	974.38	77.58	801	932	968	1031	1145
Fine Aggregate (kg/m^3)	F_Ag	772.69	80.34	594	724.3	780	822.2	992.6
Age	Age	45.86	63.73	1	7	28	56	365
Concrete Compressive Strength	CCS	35.25	16.28	2.33	23.52	33.8	44.87	82.6

There is no strong correlation between the attributes. The highest correlation exists between the cement component and the concrete compressive strength attributes with a value of 0.49 followed by the Age and Superplasticizer components with a Pearson’s correlation value of 0.34 respectively as shown in Figure 2. This tells us that no attribute is a strong estimator of the CCS which is the output variable.



Figure 2. Pearson’s Correlation Plot of the Input and Target Variables

2.2. Model Development

The data set was randomly split into the train and test data in a ratio of 80:20 using a random state of 42 in Scikit-learn. The training dataset had 804 instances while the test dataset had 201 instances. The GridSearch cross-validation (CV) function and the 3-Fold, 5-Fold, and 10-Fold cross-validation methods were deployed to get the optimum model with the best performance and least error. There are several machine learning models used for supervised learning regression problems. However, the five machine learning regression models used in our analysis are Linear Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), and Gradient Boosting Regression (GB).

2.2.1. Linear Regression

The multivariate LR model is a statistical approach for the prediction of the value of a dependent or target variable using several independent or input variables [23]. This is achieved by fitting the line of best fit between the independent or input variable(s) x and the dependent or output variable y . For this work, the independent variables are C, BFS, FA, W, SP, C_Ag, F_Ag, and Age while the dependent variable is the CCS. This can be represented as

$$f: \mathbb{R}^D \rightarrow \mathbb{R} \tag{1}$$

where the input vector x is D-dimensional, and the function f then applied to it returns the CCS value. In essence, we aim for a function $f(x)$, such that,

$$f(x) = CCS = \theta^T x + \phi_0 \tag{2}$$

for unknown θ, ϕ . Upon training the concrete compressive dataset, the linear regression function fitted is defined in Equation 3.

$$CCS = -23.3893 + (0.1167 \times C) + (0.0981 \times BFS) + (0.0846 \times FA) - (0.1314 \times W) + (0.3315 \times SP) + (0.0155 \times C_Ag) + (0.0206 \times F_Ag) + (0.1106 \times Age) \tag{3}$$

2.2.2. Decision Tree Regression

A DT is a tree data structure made up of several nodes and branches. The decision tree algorithm utilizes a divide-and-conquer method and repeatedly partitions the input variables to classify or predict the output parameter [24].

Mathematically, a DT can be represented as a function that maps an input feature vector x to a class label y described in Equations 4 and 5 as:

$$y = f(x) \tag{4}$$

The decision tree function $f(x)$ can be defined recursively as follows:

$$f(x) = \begin{cases} C_1 & \text{if } x_i < t \\ C_2 & \text{if } x_i > t \end{cases} \tag{5}$$

where x_i is the value of the i th feature in x , t is the threshold value, and C_1 and C_2 are the class labels for the resulting subsets. The threshold value t and the class labels C_1 and C_2 are determined during the training process by optimizing a criterion such as information gain or Gini impurity. The decision tree regressor was fitted to the target variable and at each partition instance, the error was calculated between the predicted value and the known target value [25].

The decision tree's hyperparameter is tuned using cross-validation. One such hyperparameter is the depth of the tree (the number of splits a tree can make before coming to a prediction). This was set to 16 for the decision tree regression model trained on the data as presented in Table 2. Figure 3 shows the first 2 splits of the decision tree while Figure 4 shows the importance of the variables used in making prediction. In Figure 4, Cement and Age had the highest significance values at 35.7% and 32.2% respectively, and were strong estimators for predicting the value of the CCS for the DT.

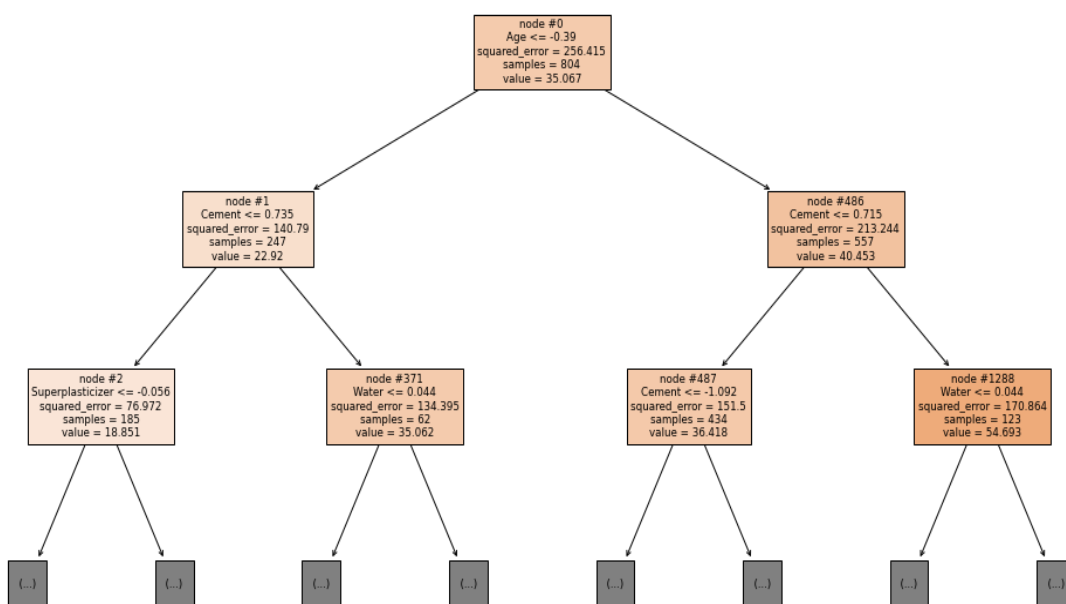


Figure 3. Decision Tree Regression Model Decision Making showing the First Two Splits

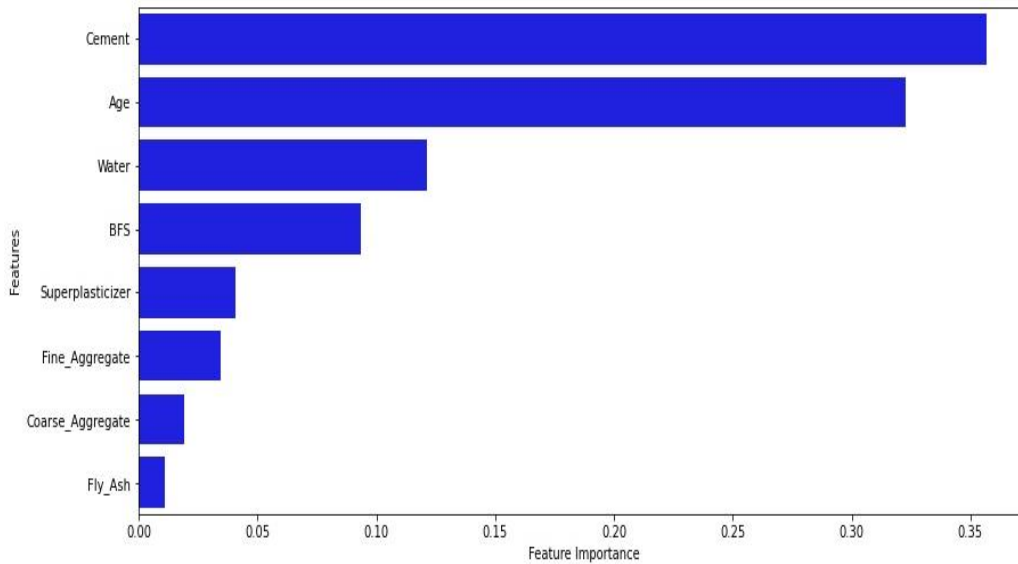


Figure 4. Variable Importance of the DT Model

2.2.3. Random Forest Regression

An RF regression is an ensemble regression technique containing numerous decision trees to predict or forecast a target variable [26]. The RF uses the Bagging (that is, bootstrapping and aggregating) method where homogenous weak learners' models (in this case, decision trees) independently learn from one another in parallel. The final prediction is achieved by a model averaging approach [25], [27]. The number of estimators (decision trees) fitted for the RF regression model was 400 while the maximum depth of each DT was 20 as shown in Table 2.

In Figure 5, Age and Cement had the highest feature importance of 35.3% and 31.1% respectively, and were the strongest estimators for predicting the value of the CCS in the RF regression model.

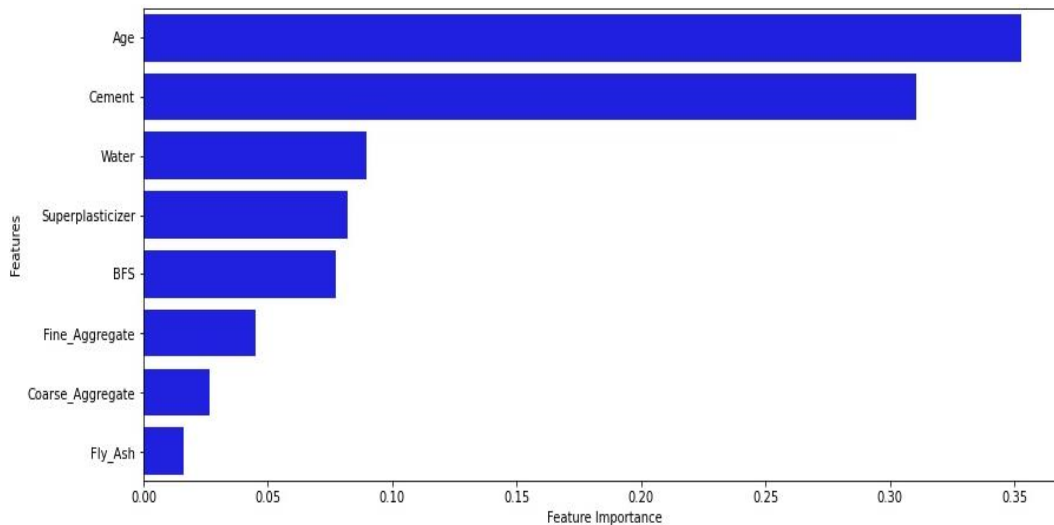


Figure 5. Variable Importance of the RF Model

2.2.4. Gradient Boosting Regression

The gradient boosting (GB) regression tree model is a stacked learning approach where a robust predictive model is formed by combining several individual weak learning regression trees (DT) [28].

For the training set described earlier with n data points (x_i, y_i) where x_i is the input features and y_i is the corresponding output or target variable. The goal of gradient boost regression is to find an ensemble of weak regression models $h_1(x), h_2(x), \dots, h_m(x)$ that can approximate the true underlying function $f(x)$ that maps the inputs to the target variable.

Each weak model $h_m(x)$ takes the input features x and returns a scalar prediction, which is then combined with the predictions of the other weak models to obtain the final prediction. The gradient boost regression algorithm iteratively adds new weak models to the ensemble, with each new model attempting to correct the errors of the previous models.

At iteration m, the current prediction of the ensemble is given in Equation 6:

$$f_m(x) = f_{m-1}(x) + \gamma_m h_m(x) \tag{6}$$

where $f_{m-1}(x)$ is the previous prediction of the ensemble, γ_m is the learning rate or step size that controls the contribution of the new weak model $h_m(x)$, and $h_m(x)$ is the new weak model that is trained to fit the negative gradient of the loss function with respect to the current prediction $f_{m-1}(x)$. The negative gradient is defined in Equation 7 as

$$\gamma_{mi} = - \left[\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)} \right] \tag{7}$$

where $L(y, f(x))$ is the loss function that measures the difference between the true target value y and the predicted value $f(x)$.

The weak model $h_m(x)$ is typically chosen to be a simple regression model such as a decision tree and is trained to minimize the loss function concerning the negative gradients γ_{mi} .

Equation 8 below is the final prediction of the gradient boost regression ensemble after m iterations:

$$f(x) = f_0(x) + \gamma_1 h_1(x) + \gamma_2 h_2(x) + \dots + \gamma_m h_m(x) \tag{8}$$

where $f_0(x)$ is the initial prediction of the ensemble and $\gamma_1, \gamma_2, \dots, \gamma_m$ are the learning rates or step sizes that control the contribution of each weak model in the ensemble [29].

The maximum depth for the Gradient Boosting Regression model was set to 4 and the number of estimators was 200 with a learning rate of 0.2 as shown in Table 2. Fig. 6 shows the feature importance of the gradient-boosting regression model where Age and Cement had the highest importance of 37.8% and 28.8% respectively. It is noteworthy that while Cement and Age features were the strongest estimators, F_Ag, C_Ag, and FA had the least feature importance for the DT, RF and GB models.

Table 2. Models' Hyperparameter Tuning

Model	Hyperparameters	Value
LR	None	-
DT	max_depth	16
RF	max_depth	20
	n_estimators	400
GB	max_depth	4
	n_estimators	200
	learning_rate	0.2
KNN	n_neighbors	7

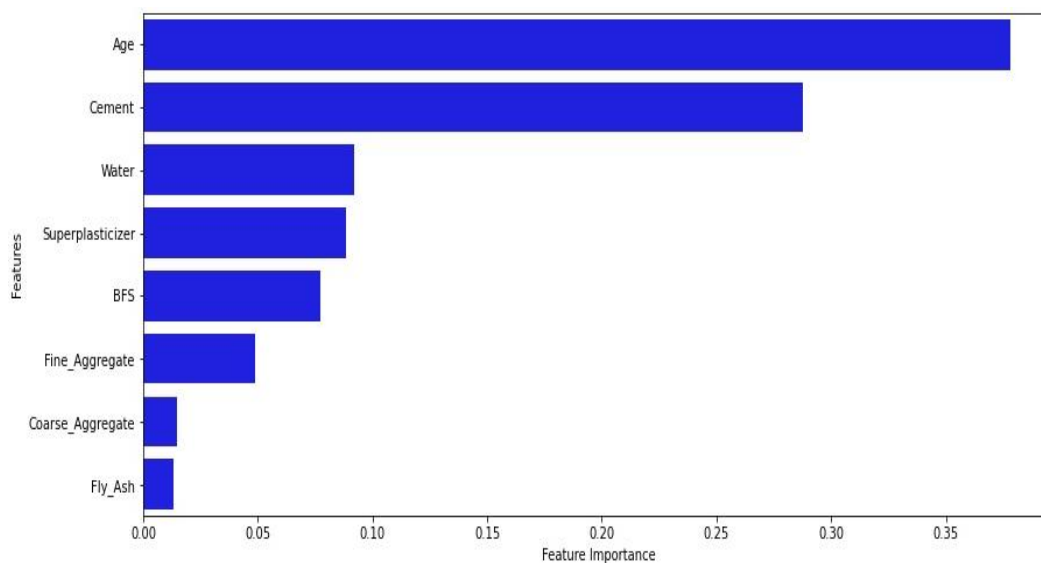


Figure 6. Variable Importance of the GB Regression Model

2.2.5. K-Nearest Neighbor Regression

K-Nearest Neighbors (KNN) is a distance-based regression algorithm [30], [31]. For the training set described earlier, the K nearest neighbor regression algorithm works as follows: Given a new input feature vector x^* , the K nearest neighbors to x^* are identified from the dataset based on a distance metric, such as Euclidean distance:

$$d(x_i, x^*) = \|x_i - x^*\| \quad (9)$$

The predicted output y^* for the new input x^* is computed as the mean of the target values of the K closest neighbors:

$$y^* = 1/K \sum_{i=1}^K y_i \quad (10)$$

where $K=7$ is the number of nearest neighbors used.

2.3. Model Testing and Evaluation

Testing and evaluating the efficiency and performance of a designed ML model is a key step of the ML lifecycle to ensure accurate and reliable predictions. To ensure accuracy and reliability, there are key performance metrics used. Such performance metrics are the mean absolute error (MAE), mean absolute percentage error (MAPE), root mean squared error (RMSE), and the R-squared (R^2) [28]. These are defined in Equations 11 – 14.

- i. MAE is the deviations between paired observations [32].

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (11)$$

where \hat{y}_i is the predicted value; y_i is the actual value; n is the number of fitted observations.

- ii. MAPE is the average of the absolute percentage errors of predictions [28].

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \quad (12)$$

- iii. RMSE is the standard deviation of the predicted errors defined in Equation 13.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}} \quad (13)$$

Generally, the lower the MAE, MAPE, and RMSE values, the better the prediction ability and capacity of the models. The model with the lowest values is said to have done the best fitting.

- iv. R-squared (R^2) is the coefficient of determination and ranges between 0 and 1. A value of 1 means the model made predictions without any error.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (14)$$

where \bar{y}_i is the mean of the actual value. A higher value of R^2 denotes a better fit.

3. Results and Discussion

This section analyzes the outcomes and performances of the various regression models employed for predicting CCS, based on the studies done in the above sections.

Cross-validation is a model validation method for assessing how the models generalize to an independent dataset. In Figure 7, the 10-fold CV had the best R^2 performances across all the regression models except for the LR model where the 3-fold and 5-fold cross-validation outperformed it. The gradient boosting regression technique had the best performance for the 3-fold, 5-fold, and 10-fold splits with values of 0.9018, 0.9143, and 0.9160 respectively compared with the other models. The random forest had the second-best performance for all cross-validation splits used with values 0.8799, 0.8878, and 0.8914 while the linear regression had the worst performance having achieved R^2 values of 0.6017, 0.5935, and 0.5892 respectively for the 3, 5 and 10-fold CVs.

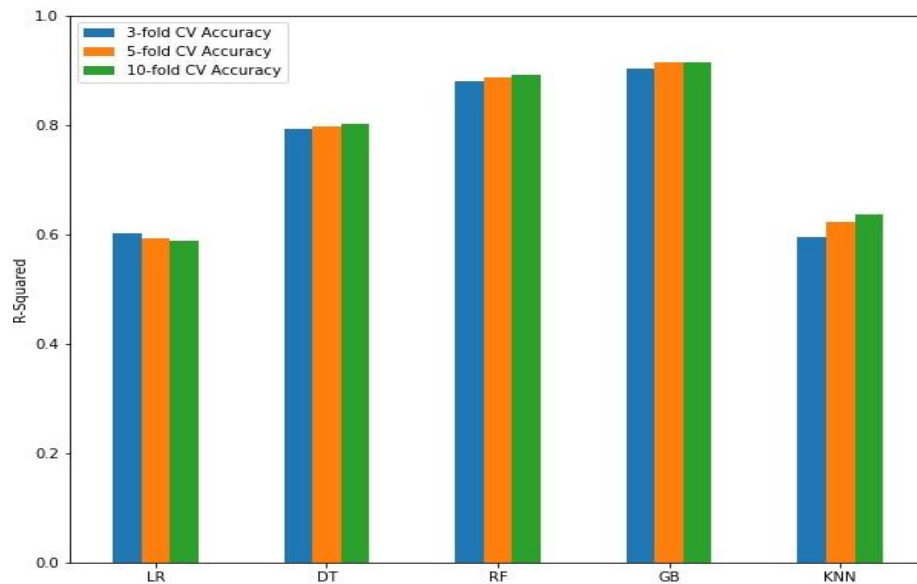


Figure 7. R² Score for Cross Validation on the Regression Models

The statistical results for the performance evaluation and testing performed on the models are presented in Table 3. It was observed from the results that the GB regression model had the lowest MAE, RMSE, and MAPE with values of 2.6776, 4.3523, and 0.0919 respectively, followed by the random forest technique with values of 3.4877, 5.1773, and 0.1224 respectively. The linear regression had the worst performance, achieving values of 8.8953, 11.1913, and 0.3469 for the MAE, RMSE, and MAPE, respectively. Likewise, GB, RF, and DT models achieved the highest value of the R-squared with 0.9365, 0.9101, and 0.8784 respectively. The KNN was the fourth-ranked performing regression model for the prediction of CCS while LR performed worst.

Table 3. Statistical Analysis of Predictive Models

ML Approaches	Training				Testing			
	MAE	R ²	RMSE	MAPE	MAE	R ²	RMSE	MAPE
LR	7.9599	0.6099	10.0018	0.3080	8.8953	0.5802	11.1913	0.3469
DT	0.0960	0.9964	0.9620	0.0026	3.8500	0.8784	6.0237	0.1390
RF	1.3584	0.9839	2.0323	0.0471	3.4877	0.9101	5.1773	0.1224
GB	0.8823	0.9912	1.5009	0.0316	2.6776	0.9365	4.3523	0.0919
KNN	0.0944	0.9964	0.9615	0.0026	5.8132	0.7592	8.4758	0.2255

For CCS prediction, innumerable works and methods have been done and applied. The comparison of our results with previous studies using individual traditional machine learning regressors is shown in Table 4. From the results, our approach had the best performance among the traditional ML approaches employed having the least errors. However, a study in [20] using a deep convolutional neural network had the overall best R-squared value of 0.97. This shows that deep learning has greater potential for more accurate prediction, albeit at a greater computational cost.

Table 4. Result Comparison in Previous Studies

Study	Algorithm	Performance Metrics			
		MAE	R ²	RMSE	MAPE
Our work	Best Model: GB	2.68	0.94	4.35	9.19%
Khan et al. [15]	BR	4.26	0.92	5.69	-
Muliauwan et al. [19]	ANN	-	0.91	-	-
Zeng et al. [20]	DCNN	-	0.97	-	-

For a real-time deployment of the model, streamlit, a framework for machine learning, is used to develop interactive web applications. The best-performing model which is the GB regression model was loaded. The developed and deployed web

application is available at the website <https://jblideal-concrete-compressive-strength-cement-strength-app-1i4gpn.streamlit.app/>. The web application is shown in Figure 8 with the input variables set and prediction of CCS.

The deployment is of great benefit for concrete strength determination in practice under normal conditions as it allows determining the concrete compressive strength with different mixture contents and ratios before mixing the different components. Real-time prediction allows varying all inputs or mixtures to achieve the desired target. However, caution must be taken in the use of the tools as it is important that the actual components match the characteristics of the input variables trained and deployed. Likewise, all conditions and guidelines should be adhered to avoid miscalculations or the concrete strength.

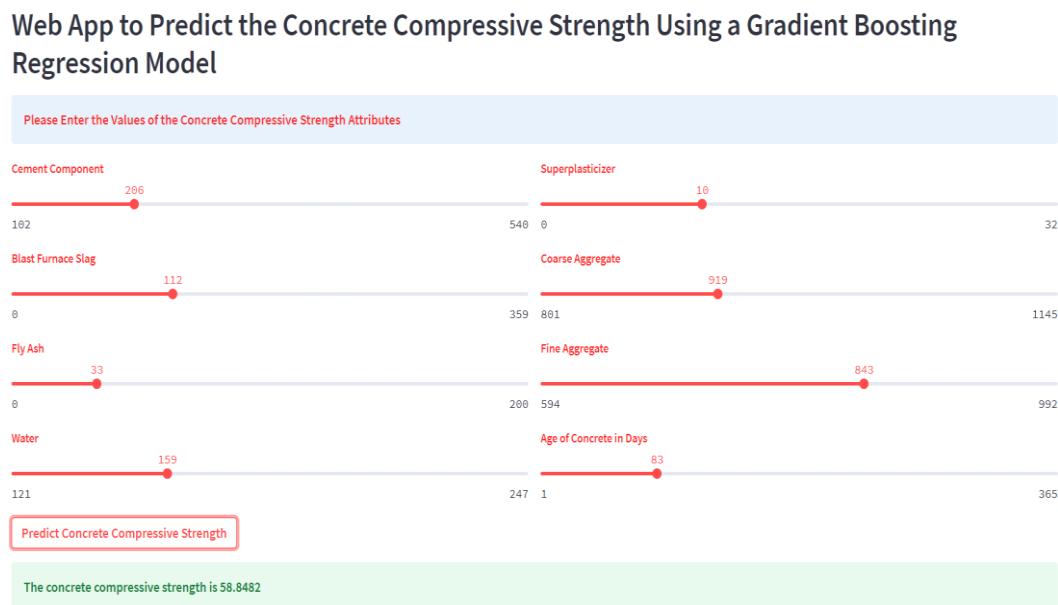


Figure 8. Deployed Web App with Prediction

4. Conclusion

A possible alternative for forecasting concrete strength from non-destructive testing data is machine learning. However, a lack of consensus on which machine learning algorithms are most appropriate for this task, and a need for more comparative studies to evaluate the performance of different methods and identify best practices for applying them in practice. This study addressed this research gap by conducting a comparative analysis of ML methods for determining concrete strength. The study showed that the gradient-boosting regression model had the best level of accuracy for predicting the CCS, followed by the random forest technique. In contrast, the linear regression had the worst performance. These findings have important implications for improving the accuracy and cost-effectiveness of methods for assessing the quality of concrete structures, improving safety, and reducing maintenance costs over the lifetime of these structures. The developed and deployed web application for the best-performing model provides an example of how these models can be used in real-time deployment for practical applications. Further research could explore the effects of external factors such as temperature, humidity, and curing time on the accuracy of the ML models and investigate the generalizability of these findings to other types of concrete.

References

- [1] ACI, *AMERICAN CONCRETE INSTITUTE. ACI 318: Building Code Requirements for Structural Concrete*. American Concrete Institute, 2008.
- [2] J. Zhang, Y. Zhao, and H. Li, "Experimental Investigation and Prediction of Compressive Strength of Ultra-High Performance Concrete Containing Supplementary Cementitious Materials," *Adv. Mater. Sci. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/4563164.
- [3] B. G. Aiyer, D. Kim, N. Karingattikkal, P. Samui, and P. R. Rao, "Prediction of compressive strength of self-compacting concrete using least square support vector machine and relevance vector machine," *KSCE J. Civ. Eng.*, vol. 18, no. 6, pp. 1753–1758, 2014, doi: 10.1007/s12205-014-0524-0.
- [4] C. Bilim, C. D. Atiş, H. Tanyildizi, and O. Karahan, "Predicting the compressive strength of ground granulated blast furnace slag concrete using artificial neural network," *Adv. Eng. Softw.*, vol. 40, no. 5, pp. 334–340, 2009, doi: 10.1016/j.advengsoft.2008.05.005.
- [5] R. Mustapha and M. EL Aroussi, "High-Performance Concrete Compressive Strength Prediction Based Weighted Support Vector Machines," *Int. J. Eng. Res. Appl.*, vol. 07, no. 01, pp. 68–75, 2017, doi: 10.9790/9622-0701016875.
- [6] S. Popovics, "Analysis of the concrete strength versus water-cement ratio relationship," *ACI Mater. J.*, vol. 87, no. 5, pp. 517–529, 1990, doi: 10.14359/1944.

- [7] J. P. Zaniewski, *MATERIALS FOR CIVIL AND 3 rd Edition Michael S. Mamlouk Arizona State University*. 2011.
- [8] R. Kozul and D. Darwin, "Effects of Aggregate Type, Size and Content on Concrete Strength and Fracture Energy," 1997.
- [9] A. Fernández-Jiménez and A. Palomo, "Characterisation of fly ashes. Potential reactivity as alkaline cements," *Fuel*, vol. 82, no. 18, pp. 2259–2265, 2003, doi: 10.1016/S0016-2361(03)00194-7.
- [10] J. M. Fox, "Fly Ash Classification-Old and New Ideas," in *Fly Ash Classification – Old and New Ideas*, 2017, pp. 1–15.
- [11] A. M. Zeyad, "Effect of curing methods in hot weather on the properties of high-strength concretes," *J. King Saud Univ. - Eng. Sci.*, vol. 31, no. 3, pp. 218–223, 2019, doi: 10.1016/j.jksues.2017.04.004.
- [12] B. A. Young, A. Hall, L. Pilon, P. Gupta, and G. Sant, "Can the compressive strength of concrete be estimated from knowledge of the mixture proportions?: New insights from statistical analysis and machine learning methods," *Cem. Concr. Res.*, vol. 115, no. July, pp. 379–388, 2019, doi: 10.1016/j.cemconres.2018.09.006.
- [13] M. A. DeRousseau, J. R. Kasprzyk, and W. V. Srubar, "Computational design optimization of concrete mixtures: A review," *Cem. Concr. Res.*, vol. 109, pp. 42–53, 2018, doi: 10.1016/j.cemconres.2018.04.007.
- [14] M. Z. Naser, "An engineer's guide to eXplainable Artificial Intelligence and Interpretable Machine Learning: Navigating causality, forced goodness, and the false perception of inference," *Autom. Constr.*, vol. 129, no. September, 2021, doi: 10.1016/j.autcon.2021.103821.
- [15] K. Khan, W. Ahmad, M. N. Amin, F. Aslam, A. Ahmad, and M. A. Al-Faiad, "Comparison of Prediction Models Based on Machine Learning for the Compressive Strength Estimation of Recycled Aggregate Concrete," *Materials (Basel)*, vol. 15, no. 10, pp. 1–36, 2022, doi: 10.3390/ma15103430.
- [16] L. Chi *et al.*, "Machine learning prediction of compressive strength of concrete with resistivity modification," *Mater. Today Commun.*, vol. 36, p. 106470, Aug. 2023, doi: 10.1016/j.mtcomm.2023.106470.
- [17] P. G. Asteris, A. D. Skentou, A. Bardhan, P. Samui, and K. Pilakoutas, "Predicting concrete compressive strength using hybrid ensembling of surrogate machine learning models," *Cem. Concr. Res.*, vol. 145, no. October 2020, p. 106449, 2021, doi: 10.1016/j.cemconres.2021.106449.
- [18] D.-C. Feng *et al.*, "Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach," *Constr. Build. Mater.*, vol. 230, p. 117000, Jan. 2020, doi: 10.1016/j.conbuildmat.2019.117000.
- [19] H. N. Muliauwan, D. Prayogo, G. Gaby, and K. Harsono, "Prediction of Concrete Compressive Strength Using Artificial Intelligence Methods," *J. Phys. Conf. Ser.*, vol. 1625, no. 1, p. 012018, Sep. 2020, doi: 10.1088/1742-6596/1625/1/012018.
- [20] Z. Zeng *et al.*, "Accurate prediction of concrete compressive strength based on explainable features using deep learning," *Constr. Build. Mater.*, vol. 329, 2022, doi: 10.1016/j.conbuildmat.2022.127082.
- [21] I.-C. Yeh, "Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks," *Cem. Concr. Res.*, vol. 28, no. 12, pp. 1797–1808, 1998.
- [22] I.-C. Yeh, "Concrete Compressive Strength." UCI Machine Learning Repository, 2007, doi: 10.24432/C5PK67.
- [23] D. Maulud and A. M. Abdulazeez, "A Review on Linear Regression Comprehensive in Machine Learning," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 4, pp. 140–147, 2020, doi: 10.38094/jastt1457.
- [24] E. Pekel, "Estimation of soil moisture using decision tree regression," *Theor. Appl. Climatol.*, vol. 139, no. 3–4, pp. 1111–1119, 2020, doi: 10.1007/s00704-019-03048-8.
- [25] M. Rakhra *et al.*, "Crop Price Prediction Using Random Forest and Decision Tree Regression:-A Review," *Mater. Today Proc.*, no. xxxx, 2021, doi: 10.1016/j.matpr.2021.03.261.
- [26] V. Rodriguez-Galiano, M. Sanchez-Castillo, M. Chica-Olmo, and M. Chica-Rivas, "Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines," *Ore Geol. Rev.*, vol. 71, pp. 804–818, 2015, doi: 10.1016/j.oregeorev.2015.01.001.
- [27] B. Singh, P. Sihag, and K. Singh, "Modelling of impact of water quality on infiltration rate of soil by random forest regression," *Model. Earth Syst. Environ.*, vol. 3, no. 3, pp. 999–1004, 2017, doi: 10.1007/s40808-017-0347-3.
- [28] J. Cai, K. Xu, Y. Zhu, F. Hu, and L. Li, "Prediction and analysis of net ecosystem carbon exchange based on gradient boosting regression and random forest," *Appl. Energy*, vol. 262, no. 114566, pp. 1–14, 2020, doi: 10.1016/j.apenergy.2020.114566.
- [29] U. Singh, M. Rizwan, M. Alaraj, and I. Alsaïdan, "A machine learning-based gradient boosting regression approach for wind power production forecasting: A step towards smart grid environments," *Energies*, vol. 14, no. 16, pp. 1–21, 2021, doi: 10.3390/en14165196.
- [30] L. E. de Oliveira Aparecido, G. de Souza Rolim, J. R. da Silva Cabral De Moraes, C. T. S. Costa, and P. S. de Souza, "Machine learning algorithms for forecasting the incidence of Coffea arabica pests and diseases," *Int. J. Biometeorol.*, vol. 64, no. 4, pp. 671–688, 2020, doi: 10.1007/s00484-019-01856-1.
- [31] C. Araújo, C. Soares, I. Pereira, D. Coelho, M. Â. Rebelo, and A. Madureira, "A Novel Approach for Send Time Prediction on Email Marketing," *Appl. Sci.*, vol. 12, no. 8310, pp. 1–13, 2022, doi: 10.3390/app12168310.
- [32] A. O. Oyedeji, A. M. Salami, O. Folorunsho, and O. R. Abolade, "Analysis and Prediction of Student Academic Performance Using Machine Learning," *JITCE (Journal Inf. Technol. Comput. Eng.)*, vol. 4, no. 01, pp. 10–15, 2020, doi: 10.25077/jitce.4.01.10-15.2020.

Author(s) Contributions

Ajibola Oyedeji: Conceptualization, Methodology, Writing – Original Draft, Software

Adekunle David: Conceptualization, Methodology, Writing – Original Draft

Ositola Osifeko: Methodology, Writing – Original Draft, Software

Abisola Olayiwola: Methodology, Writing – review & editing

Omobolaji Opafola: Methodology, Writing – review & editing

Acknowledgments

Not Applicable

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefitted from are stated in the bibliography.

Availability of data and materials

Not Applicable

Plagiarism Statement

This article has been scanned by iThenticate™.

The Importance of Rhythm Activity in Epilepsy EEG Signal Classification (An Educational Article)

Negin Melek¹ 

¹Department of Computer Engineering, Faculty of Engineering, Giresun University, Giresun, Türkiye

Corresponding author:

Negin Melek, Department of Computer Engineering, Giresun University, Giresun, Türkiye
negin.melek@giresun.edu.tr



Article History:

Received: 14.02.2024

Accepted: 30.05.2024

Published Online: 23.08.2024

ABSTRACT

Electroencephalography (EEG), used to record the random electrical activity in brain, is a known medical test. In this test, a graphical waveform is obtained by measuring the electrical activity of the cells. In the medical world, the relationship between epilepsy and EEG can be understood by examining changes in brain activity during or between epileptic seizures. EEG is a useful tool in the early treatment and diagnosis of epilepsy. Whether seizures, generally known as abnormal electrical discharges in brain cells, are of epileptic origin, comes to light through EEG. The main goal of our study was to demonstrate the EEG rhythm effectiveness for the diagnosis of epilepsy in EEG data obtained from the epilepsy center of Bonn Freiburg University Hospital. Time domain feature extraction of EEG band classification results was examined in detail against the classification results of frequency domain feature extraction of EEG rhythms in healthy subjects and subjects with epilepsy. By extracting effective features from EEG data in both time and frequency domains, the k nearest neighbor (KNN) algorithm was used for the time and frequency domain. It cannot be overlooked that among the four methods used for performance evaluation in the designed model, the classification success of frequency domain features was more successful than that of time domain features. Using the KNN algorithm, healthy individuals and epilepsy patients with seizures were classified with 100% success.

Keywords: EEG, Epilepsy, Rhythm, Feature extraction, Classification

1. Introduction

Electroencephalography (EEG) resembles a melody capturing a graceful dance of brain waves, a window that reveals the mysterious rhythm of the mental world. EEG contains useful information about the random nature and dynamics of brain waves and plays a major role in disease diagnosis. From a general perspective, the scientific application areas of EEG can be divided into three categories: entertainment, engineering, and medical [1]. Figure 1 shows the percentage distribution of EEG application areas according to the research conducted in 2020. The entertainment application field includes examples such as brain-eye combinatorial controls, brain-based control and robotic games, car racing, social interaction, and virtual marketing [2]. Most EEG-based studies in engineering can be divided into two subgroups: biometrics and brain-computer interface (BCI) [3]. The process of classifying individuals based on their behavioral and physiological characteristics is defined as biometrics. One of the extraordinary features offered by EEG signals is their typical biometric identification for each individual [4]. Thus, thanks to this uniqueness, we can say that each person has a unique neural signature [5]. BCI, another important application area of EEG in engineering, is a technology that enables direct communication between the brain and an external device such as a computer. Focusing on BCI technology, great efforts continue to be made to increase EEG signal quality and communication speed, as well as to improve model accuracy. Research in this field represents a broad spectrum, creating major revolutions from healthcare to entertainment [6].

From a medical and healthcare perspective, detailed analysis of brain signal activities provides great convenience in predicting diseases and abnormalities in neurology. In this context, EEG is an important tool for understanding brain functions, learning, and memory processes.

Quantitative EEG (QEEG) was used to diagnose and predict Parkinson's disease dementia in 2011 [7]. This study proved that θ band relative power analyses are a determining factor in the incidence of dementia.

Another study showed that the independent component analysis (ICA) preprocessing method increases the performance in developing an Alzheimer's disease detection system with automatic brain waves [8].

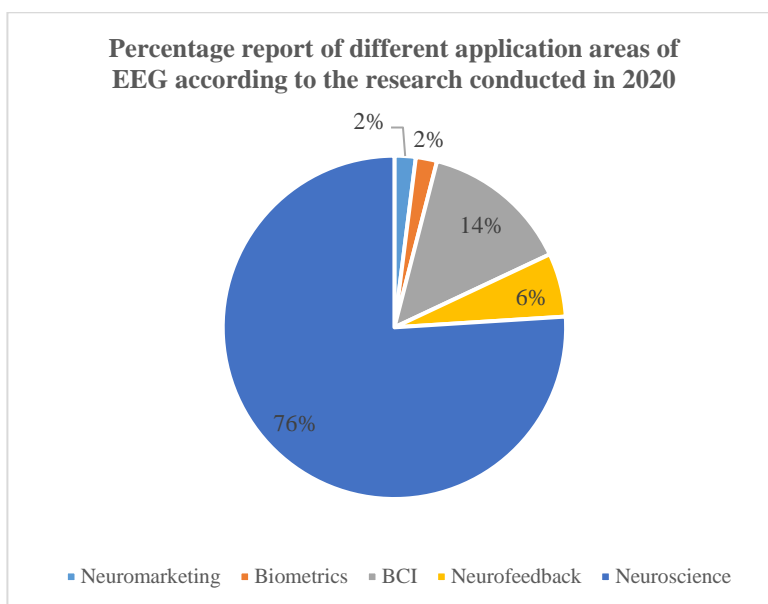


Figure 1. Percentage Distribution of EEG Application Areas According to the Research Conducted in 2020 [2]

EEG is not a direct method for treating language disorders and dyslexia, but it is useful for improving reading and writing skills by discovering abnormalities in the individual's neurological processes [9]. In addition, EEG is used as an auxiliary tool in treating language disorders, along with neurofeedback and auditory therapy methods.

Attention-deficit/hyperactivity disorder (ADHD) [10], a common psychiatric disorder of our age, is frequently observed in children. As a result of the valuable information contained in EEG bands, theta/beta ratio (TBR) is a frequently used measure in ADHD studies [11].

EEG plays an important role in detecting seizures due to abnormal electrical activities of neurons. These unwanted activities can be discovered and characterized using EEG waves [12]. There are many types of seizures with different electrical and wave patterns. EEG categorizes seizure types according to the characteristics of these patterns [13]. This application area of EEG is of great importance in medicine because it has serious benefits in the early and rapid management of individuals with seizures. It also enables healthcare experts to make informed and more precise decisions about early treatment and diagnosis of patients with seizures [14].

Early diagnosis and conscious support can increase the standards and quality of life of individuals with symptoms of autism spectrum disorder. In this context, the debate on whether the findings of research on brain waves and autism spectrum disorder are related to each other is still ongoing [15]. It has been proven that EEG signals taken from preschool children show signs of high-grade autism spectrum disorder. On the other hand, it was concluded that these symptoms are more pronounced while asleep than when awake [16].

In addition, many scientific studies have touched upon other different common and important application areas of EEG, such as insomnia detection [17] and treatment method determination, brain damage analysis and detection [18], consciousness assessment [19], and stress and anxiety disorder treatment [20], [21].

From a neuroscience perspective, real-time access to the random activity and behavior of brain waves provides instantaneous valuable information about the nature of the EEG. Thanks to this feature, the treatment of many cognitive disorders and the diagnosis of abnormal brain behavior disorders have become understandable. The ability of this type of EEG to send instant feedback in response to brain functions is considered a great advantage in early medical diagnosis and BCI studies [22]. Continuous observation of brain activities in epilepsy and monitoring of the abnormal patterns of the brain that occur during seizures have attracted many researchers in real-time EEG studies [23]. Thanks to this reality, early detection of seizures and detailed analysis of their characteristics have become possible.

Epilepsy disorder, caused by excessive electrical movements of neurons, is the fourth known neurological disorder worldwide [24]. Regardless of age, the result of these involuntary excessive electrical oscillations can manifest itself as recurrent seizures. The topics of interest in the evaluation of a patient with recurrent seizures can be expressed as follows: Is the source of the seizures neurological? What type of seizure is it? Are these seizures a sign of epilepsy? With the correct answers to these questions, the patient's treatment process is determined. In addition to determining treatment, it is important to provide regular follow-up and support for individuals with suspected epilepsy. The presentation that summarizes the care process of epilepsy patients to improve their quality of life is shown in Figure 2.

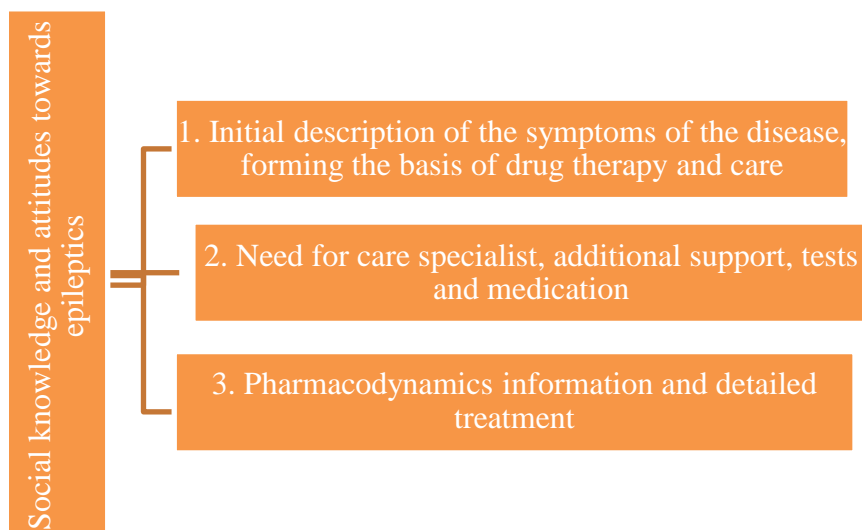


Figure 2. The Care Process of Epilepsy Patients in Order to Improve the Quality of Life [25]

2. Related Studies

Seizures in epilepsy directly affect the patient's quality of life. These seizures, which vary in duration and severity from person to person, are manifested by variable behavioral patterns [24]. These behaviors are generally classified as loss of consciousness, staring, immobility or involuntary motor symptoms, muscle stiffness, etc.

Based on the World Health Organization (WHO) report in 2019, the stark reality is that the majority of epilepsy patients live in poor countries and do not have access to low-cost anti-seizure drugs. This report can report the good news that a quarter of epilepsy patients will be saved from annoying and untimely seizures by using low-cost drugs.

Computer-based models, which provide great convenience in the medical world, can diagnose many disorders by systematically analyzing physiological signals [26]. EEG, a physiological signal, is a magical tool revealing different brain disorders. EEG still maintains its important position as a diagnostic tool in diagnosing epilepsy disorders caused by excessive electrical activity of the brain [27].

In a review study conducted in 2015 [28], EEG was proven to be useful in diagnosing epilepsy syndrome, determining whether the attack belongs to the origin of epilepsy, and predicting attack recurrence. In this situation, any abnormal activity in brain cells leaves a bright trace of information worth investigating in EEG signals.

Important factors in diagnosing epileptic patients include determining the type of epileptic seizures and investigating the region in the brain causing epilepsy syndrome, thanks to the collaboration of EEG, neuroimaging, cheap drugs, and genetic tests [29].

Epilepsy diagnosis studies based on machine learning are increasing rapidly every day. Numerous studies have been conducted to diagnose seizures using different machine-learning techniques by analyzing EEG data [30]. An effective method was presented for detecting epileptic seizures in different EEG signal combinations using four common classifiers. In that study, the weighted complex networks model appeared useful in diagnosing epilepsy syndrome despite its high noise tolerance.

With the advancement of technology, automatic epilepsy seizure detection models based on machine learning have replaced traditional and time-consuming diagnostic methods. In a detailed study conducted in 2018 [31], the support vector machine algorithm was selected for the classification step using two feature extraction techniques. The procedure of this successful study was recommended for use in epileptic and healthy signal classification.

A different approach based on machine learning using EEG signals for epileptic seizure diagnosis was presented by Amin et al. [32]. In this study, automatic seizure diagnosis was achieved using wavelet analysis and arithmetic coding. Since the visual evaluation of EEG signal analysis, widely used in epilepsy diagnosis, is error-prone, erratic, and sensitive to subjective variability, automatic machine learning-based methods have been the focus of the cited study. This computer-aided diagnosis (CAD) technique consists of three steps and can be used easily in clinical studies by providing extraordinary performance in classifying epileptic and healthy individuals.

In CAD techniques utilized in epilepsy studies, effective features in the frequency, time, or time-frequency domain are extracted from the frequently preferred EEG signals [33]. This effective step directly affects the performance of classification algorithms. Classification algorithms are then employed to diagnose seizures or to distinguish epileptic signals from healthy

signals. With the aim of shedding light on medical and clinical studies, the subject of rapid and successful seizure diagnosis and epileptic/healthy signal classification continues to be current and attracts attention in the scientific literature.

Diagnosis of epileptic seizures was possible in EEG signals with four classification methods using a genetic algorithm [33]. In this study, the success of classification algorithms was evaluated in terms of accuracy, specificity, and sensitivity. Among the algorithms used, artificial neural networks showed a more successful result than other algorithms, with 97.82%.

In another machine learning-based study for seizure diagnosis, a different approach was presented, concentrating on discrete wavelet transform and using a feature selection technique. In this new approach, using linear discriminant analysis (LDA), principal component analysis (PCA), and statistical features, epileptic and healthy EEG signals were classified with k nearest neighbor (KNN) and naive Bayes methods [31]. This model, applied in the epilepsy database of the University of Bonn, has achieved great success for the KNN classification algorithm.

Time-frequency statistical feature selection was used in another machine learning-based EEG epileptic classification model. In the model, independent component analysis (ICA) [34] was performed to effectively extract various processes from EEG. For the subjective-independent analysis of dynamic and highly non-stationary EEGs, a detailed analysis was carried out using ANOVA-based feature selection and fuzzy classification methods. The accuracy of the study was determined to be 96.48% in terms of seizure diagnosis [35].

A dataset for epilepsy seizure classification was prepared at Izmir Katip Çelebi University in 2021. In this study, the Neurofax device recorded the EEG signals of 16 subjects using surface electrodes. To expand the research, this study utilized a dataset financed by the European Union and another dataset known as EPILEPSIAE. Thanks to four technical features, such as energy, correlation, power spectral distance, and statistical significance measures, and using the empirical mode decomposition (EMD) technique, a success rate of 96.8% was achieved for the naive Bayes classifier.

Based on the literature review, epilepsy crisis detection, seizure diagnosis, and signal classification continue to be the focus of many studies. Epileptic EEG signal classification based on machine learning is the main goal of numerous studies. In the current study, the widely used dataset supplied by the University of Bonn [36] was used for our machine learning model design. In healthy and epileptic EEG signal classification, the importance of band rhythm efficiency was emphasized by analyzing all EEG bands and the rhythms of the EEG band. In this detailed Epilepsy classification analysis, attention is paid to EEG rhythm analysis, which appears to be a way to increase model accuracy.

3. Materials and Method

3.1. Dataset

This study, which offers a different perspective for those not highly experienced in machine learning-based physiological signal analysis studies, uses the EEG dataset provided by Andrzejak et al. [37] from the University of Bonn. In this case, the author of the proposed study did not apply for ethical approval. In this dataset consisting of five subsets (Z (A set), O (B set), N (C set), F (D set), and S (E set)), EEG signals were sampled with a sampling frequency of 173.61 Hz. Each set consists of a single channel and contains 100 trials. There are 4097 samples in each of these trials, which last 23.6 seconds. These EEG signals were recorded with 12-bit resolution. As a result of visual inspection, artifacts resulting from eye and muscle movements were largely eliminated from EEG recordings. In this recording system, which has a bandwidth between 0.5 Hz and 85 Hz, a 40 Hz low-pass filter was applied to the raw EEG signal as a preprocessing step.

Table 1 summarizes this data to understand the dataset efficiently and comprehensively [36]. The electrode positions used for EEG recordings taken with surface electrodes in sets A and B are shown in Figure 3. When the results obtained from the scalp with extracranial electrodes in the EEG recording process are unsuccessful, intracranial electrodes are used by the surgeon to record brain activity by placing them on the patient's skull.

Sets A and B contain EEG data recorded from 5 healthy subjects in an awake and relaxed state with eyes open in set A and closed eyes in set B. Data from 5 patients who achieved complete control of their epilepsy after removal of the epileptic seizure focus are presented in sets C and D. Finally, E contains the only ICTAL (a seizure episode) activity observed in epileptiform regions. Figure 4 displays an example of an EEG signal related to each set.

Table 1. Dataset Details

Classes	Subject	State	Electrode Type	Number of Trials	Number of Samples
Set A	5 healthy subjects	Awake with open eyes	Extracranial	100	4097
Set B	The same 5 healthy subjects	Awake with closed eyes	Extracranial	100	4097
Set C	5 epileptic subjects	seizure free	Intracranial	100	4097
Set D	The same 5 epileptic subjects	seizure free	Intracranial	100	4097
Set E	The same 5 epileptic subjects	with seizure	Intracranial	100	4097

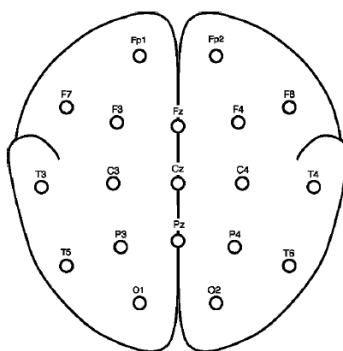


Figure 3. International 10-20 System Surface Electrode Positions

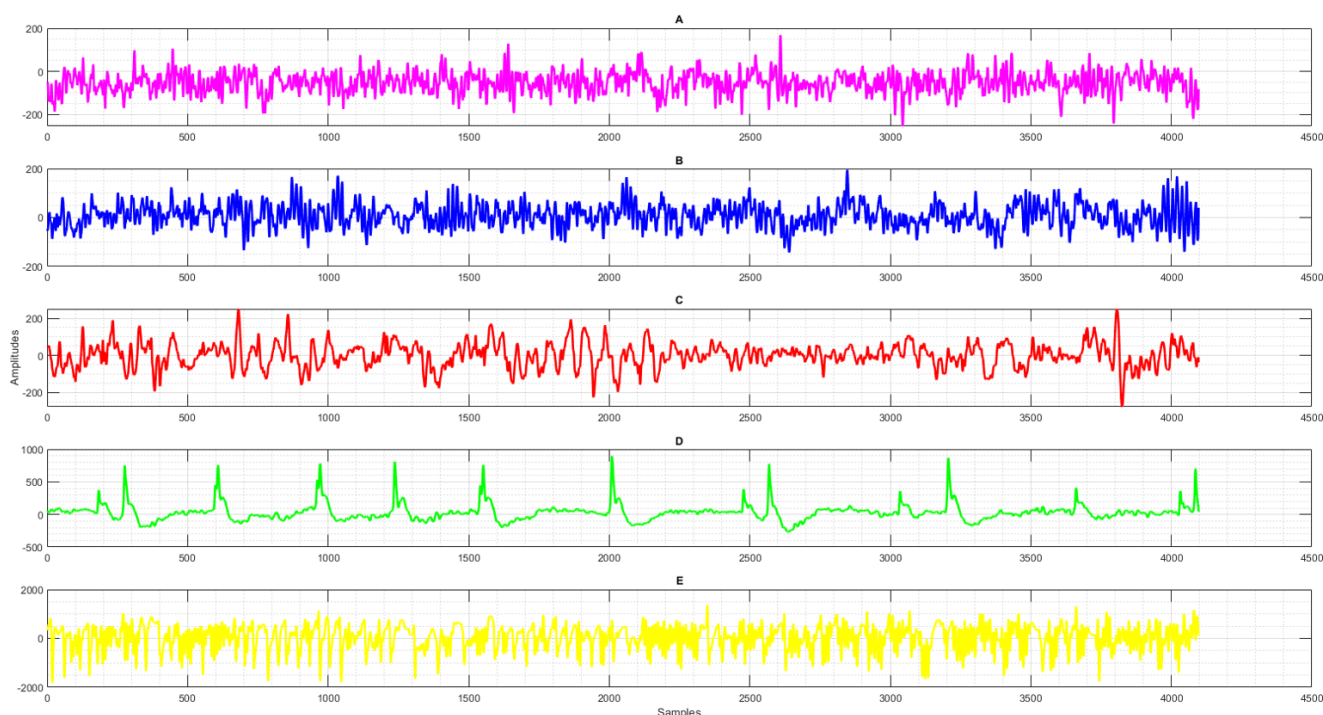


Figure 4. An Example of an EEG Signal Related to Each Set

3.2. Applying machine learning steps

This study aims to present comprehensive research to researchers who are pioneers in EEG signal processing by comparing the classification performance of time domain feature extraction [38] against the classification performance of frequency-domain feature extraction [39] in the diagnosis of epilepsy.

In the first stage, EEG records with a .txt extension are entered into MATLAB and prepared for analysis. Before analysis, these data are converted to (.mat) format. MATLAB R2023a was used for all analyses. As the initial stage of machine learning, pre-processing is performed on EEG data. In this study, noise reduction and separation of EEG data into rhythms were performed as preprocessing. In this step, noise reduction and common data deletion, which are important and useful features of preprocessing, will be realized. So that, by using the pre-processing method called filtering, unwanted components were removed and it became easier to extract meaningful information for successful classification results.

The process of measuring the parameters of data is called feature extraction. The purpose of this step is to facilitate the classifier process. A measurable parameter of the fact we observe is the feature. Valuable features properly fulfill their informative role by containing accurate and relevant information about the facts. Another point that represents a good feature is distinctiveness. That means having a different value in two or more classes of data. The acceptable feature is showing the maximum variance between classes while having similar values within the class. Finally, the condition that the features are independent of each other and provide new information about the signal directly affects the classifier's performance.

In the time domain feature extraction method, the rhythms of the EEG signal were obtained by applying the Fast Fourier transform (FFT). At this stage, we produced signals that carry each rhythm information of the EEG in the time domain. Rhythms were studied instead of focusing on the entire EEG band to prevent performance degradation because some common features are too many and the differences are few in the entire EEG. In this case, the extracted feature will be a good

ambassador for the EEG signal. So, it would be more useful to separate the rhythms and extract the characteristics of each rhythm separately. Some rhythms may be good and some may be bad, but this does not matter because rhythms that carry useful information can be selected for classification using feature selection. This step is summarized for a trial from set A in Figure 5. Time domain feature extraction steps are presented in detail in Figure 6.

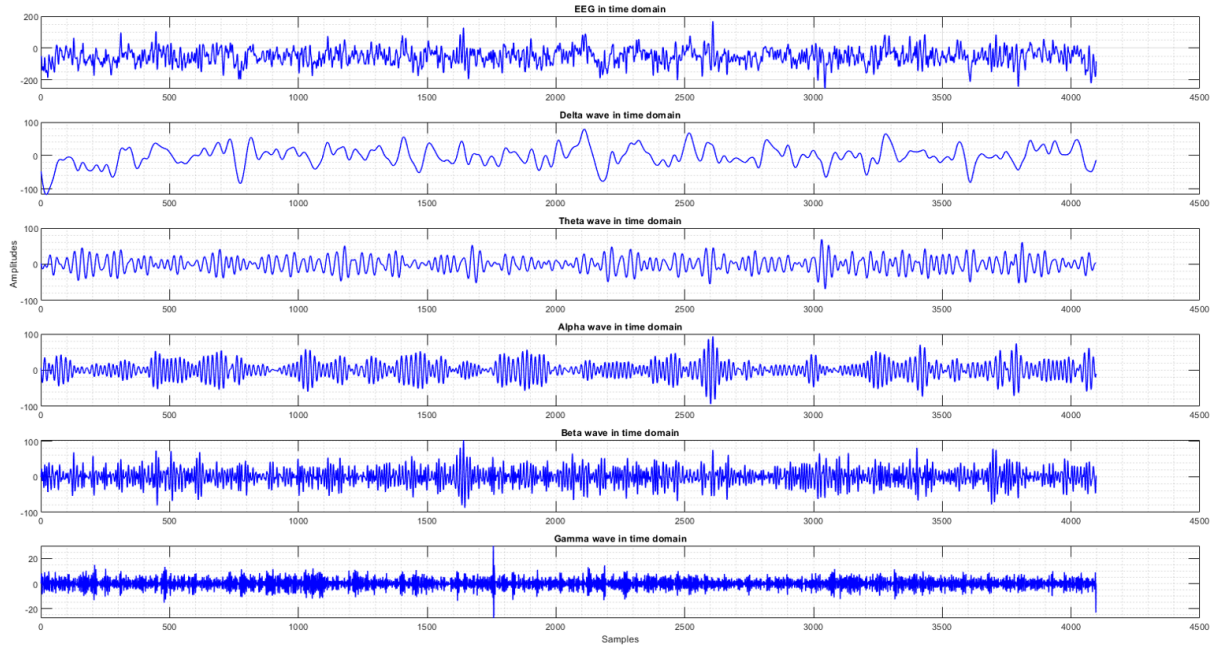


Figure 5. Preparation for Time Domain Feature Extraction from a Trial of a Set

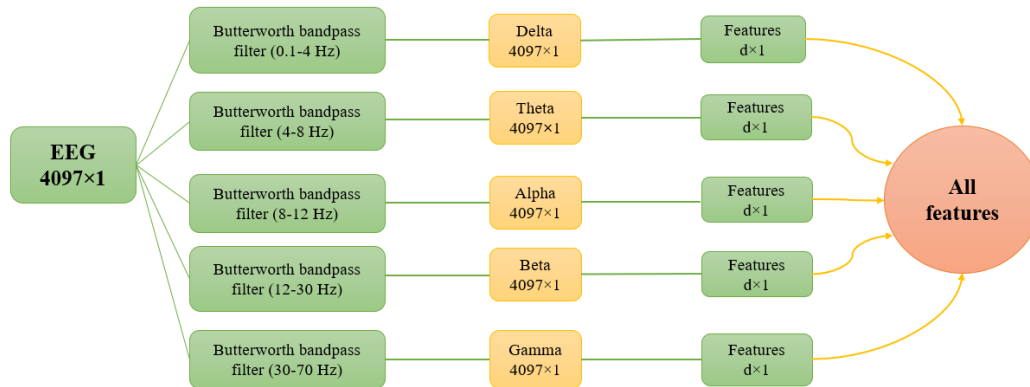


Figure 6 Time Domain Feature Extraction Steps

In calculating the statistical properties [40] of the signal in the time domain, the d value was determined as 6 by using mean, variance, skewness, kurtosis, entropy, and signal power. Mean contains information about the median of the fact data. It is represented by the parameter μ and shows the average of all signal samples. In the x signal with size $N \times 1$, the mean is calculated by Equation 1. The parameter N refers to the number of samples.

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \tag{1}$$

The variance shows the distribution of the fact data around the mean and is denoted by σ^2 . Standard deviation (σ) is the square root of the variance. Since it contains the same information as variance, it can be used for analysis. In the x signal with size $N \times 1$, the variance is calculated by Equation 2.

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2 \tag{2}$$

Skewness [41] determines information about the asymmetry of the normal distribution. This criterion, which defines the degree of asymmetry, has made decision-making in research areas more successful and provided the opportunity for accurate modeling. This type of distribution is expressed by Equation 3.

$$Skewness = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^3}{(N-1)\sigma^3} \tag{3}$$

Kurtosis determines the peak degree of the normal distribution of the event. Thanks to this statistical criterion, the degree of tailing of the distribution is clearly shown compared to the normal distribution. This type of distribution is represented by Equation 4.

$$\text{Kurtosis} = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^4}{(N-1)\sigma^4} \quad (4)$$

The calculation of entropy and signal power can be used for features. Shannon Entropy [42] determines the randomness of a phenomenon. Whether the behavior is random or factual in a case is determined by Shannon Entropy. The mathematical representation of these parameters is shown in Equations 5 and 6, respectively. By including these two features in the study, the total number of features was increased to 6 for each set.

$$H(x) = - \sum_{i=1}^N p(x_i) \log(p(x_i)) \quad (5)$$

$$P = \frac{1}{N} \sum_{i=1}^N (x_i)^2 \quad (6)$$

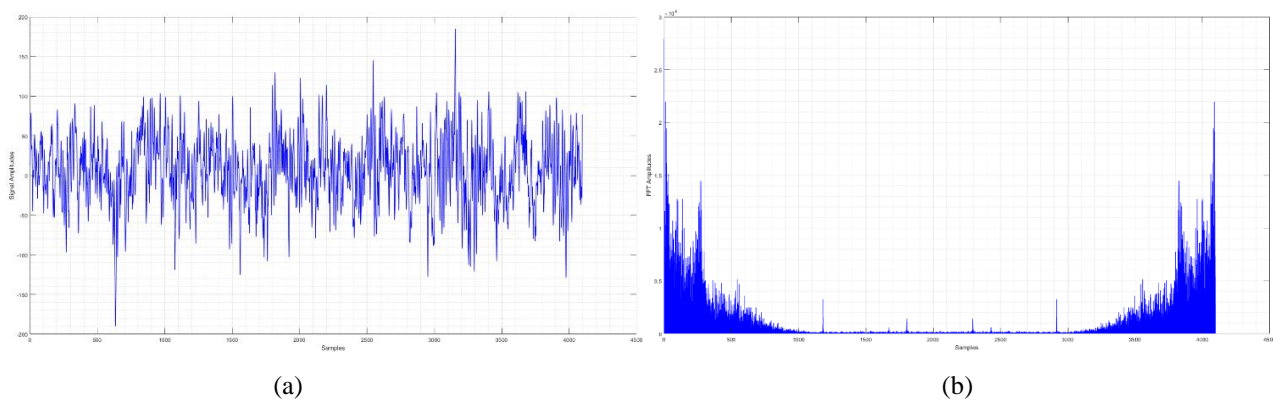
The information provided by the time domain indicates how the amplitude of the signal changes over time. In the rhythm separation process in the time domain, rhythms were obtained as a result of FFT calculation from the signal. Finally, by taking the inverse FFT, the signal in the time domain was the same size as the main signal but was directed to the feature step by obtaining a signal that would only include the coefficients of that rhythm.

In the present study, feature extraction in the frequency domain [43] plays a major role in diagnosing epilepsy. The information provided by the frequency domain shows in which frequency ranges the signal contains information. It shows the frequency band content of the EEG signal. As it is known, when FFT is calculated, the output is complex numbers. These complex numbers have an amplitude and a phase. Each signal is made of an infinite number of sinusoidal signals. Now, the frequency domain shows which sine signals the signal is made of and how this sine contributes to the formation of the main signal. Each sinusoidal wave has an amplitude and a phase.

After FFT, half of the FFT coefficients are considered for feature extraction. The first frequency is 0 Hz, and the frequency of the last coefficient is $F_{\text{sampling}}/2$ Hz. Frequency resolution (FR) was taken into account to calculate other frequencies. The time domain feature extraction steps are presented in detail in Figure 7.

Following the collaboration between medicine and engineering, the analysis of biosignals has gained momentum for early disease diagnosis [44]. EEG classification is the process of identifying a specific medical condition or classifying a specific activity by taking EEG signals. These signals are used to measure brain activity and diagnose conditions such as epilepsy, sleep disorders, ADHD [45], or in brain-computer interface (BCI) systems [46]. Classification algorithms predict whether features belong to a particular class. Common classification algorithms include support vector machines (SVM) [47], artificial neural networks (ANN) [40], decision trees [48], and k nearest neighbors (k-NN).

In the proposed EEG-based machine learning model design, the decision phase was carried out after pre-processing and time/frequency domain feature extraction in diagnosing epilepsy disease and healthy subjects. Examining how time and frequency domain features affect the classification step and which one is superior in terms of successful classification results constitute the basic framework of the study. The classification steps followed in the article are illustrated in the flowchart in Figure 8.



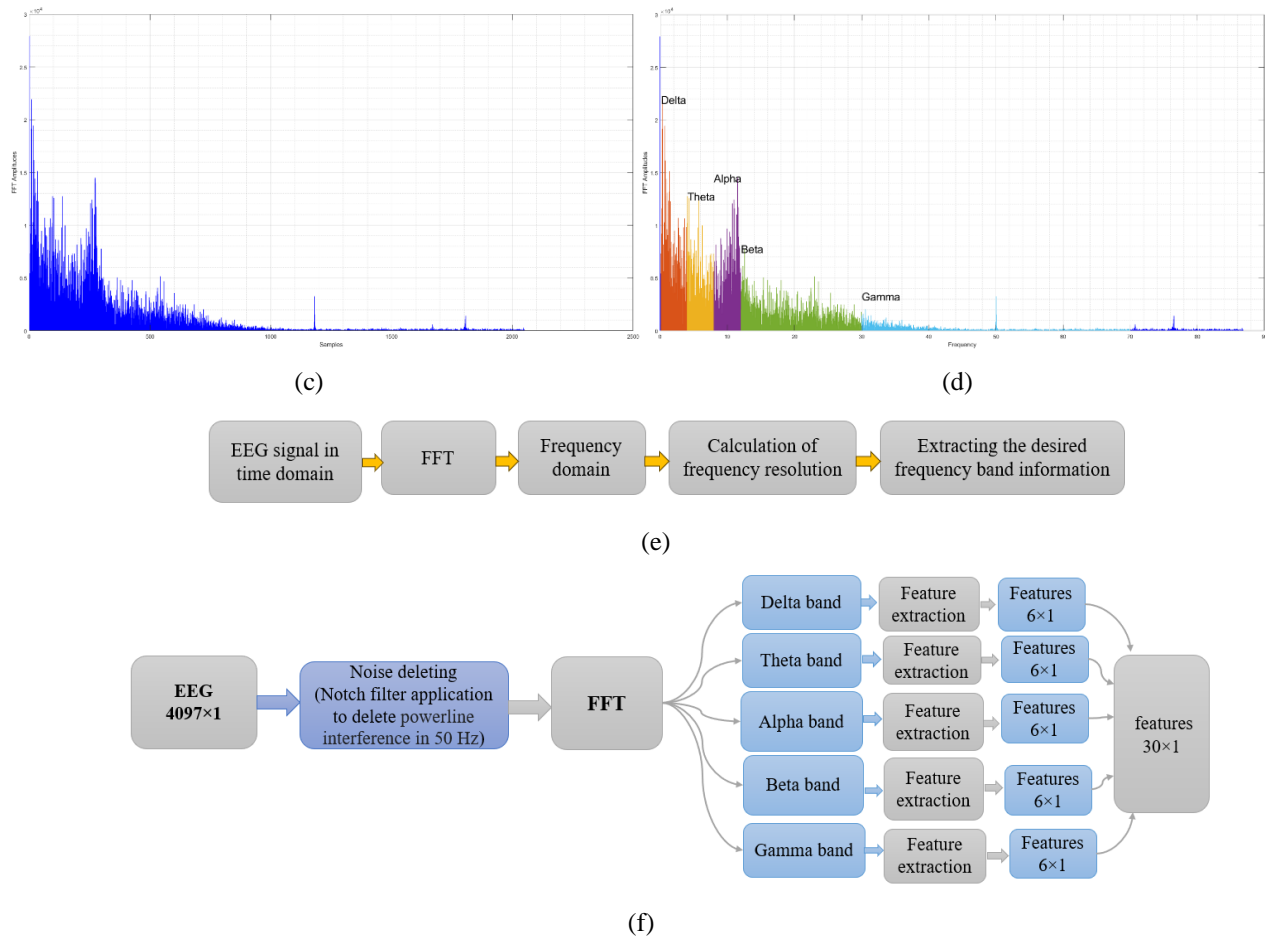


Figure 7. (a) EEG signal in the time domain, (b) Entire spectrum, (c) Half of the spectrum (select half of the coefficients), (d) Determination of rhythm coefficients, (e) and (f) General and detailed representation of feature extraction steps in the frequency domain

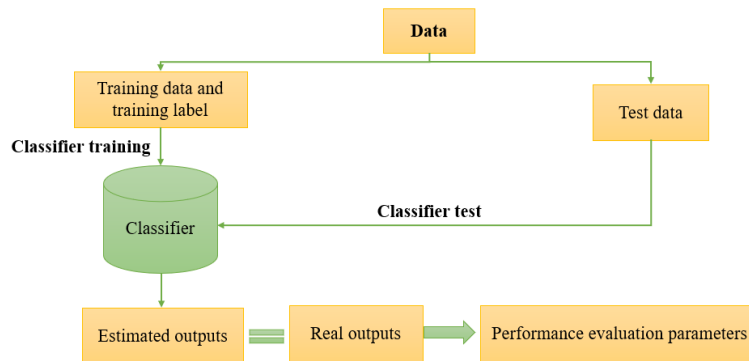


Figure 8. Classification Steps

After the necessary preprocessing on the EEG data, the classification step is started by extracting separate statistical features in the time and frequency domain. As a classification, the KNN algorithm [49] has become the chosen algorithm in many machine learning studies due to its easy use and successful performance [50]. Despite its simplicity, it is very effective and accomplished for determining an observation label. The KNN algorithm, which has the advantages of simplicity, easy understandability, and applicability, can be computationally costly and prone to overfitting when working on large datasets. Therefore, when using KNN, it is important to set the K parameter well and consider the characteristics of the dataset [51].

Although the non-parametric KNN classifier has an easy working principle, it provides a very successful and strong performance in noisy data. The KNN algorithm, which is also the focus of attention in the scientific world, is developing with the emergence of new methods. The KNN algorithm, which is frequently used in both regression and classification problems, is distance-oriented and works based on the assumption that the samples in the dataset are close to each other. In this context, the Euclidean distance definition, widely used in the proposed model, was chosen for the algorithm [52]. If the

Euclidean distance between two points with (x_1, y_1) and (x_2, y_2) coordinates is denoted by D , this distance is presented in Equation 7.

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (7)$$

Some of the solvable challenges of KNN include dealing with excessive examples, unbalanced classes, overlapping class regions, noise from irrelevant features, and outliers. Prototype development, correct adjustment of the K parameter, creation of artificial data, filter application, and outlier analysis are among the solutions that mitigate KNN difficulties.

The dataset must be divided into two sets, test and training, to carry out a classification project (as presented in Figure 8). Then, the classifier training will begin. The model will be trained with the training data and training label at this stage. Next, the trained classifier is tested with the test dataset. The most important step is to evaluate the model using performance parameters. Classifier accuracy, sensitivity, specificity, and confusion matrix were calculated as performance parameters. A detailed mathematical description of accuracy, sensitivity, and specificity is shown in Equations 8, 9, and 10, respectively. Additionally, Table 2 represents the binary problem confusion matrix detail. In the proposed epilepsy machine learning binary classification problem, TP belongs to the first class and represents the correctly classifying examples. FP shows examples that belong to the first class but are misclassified. TN belongs to the second class and shows correctly classifying examples, and finally FN belongs to the second class and shows incorrectly classifying examples.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (9)$$

$$Specificity = \frac{TN}{TN+FP} \quad (10)$$

Table 2. Binary Problem Confusion Matrix [53]

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

In our study, four validation methods were employed for model evaluation, namely, the holdout method, k-fold cross-validation method, Leave one out validation method, and Random subsampling, respectively.

The hold out method [53] is commonly used, where data are allocated only once for training and testing. The model is trained with the training data and tested with the test data; then, the performance is evaluated. Generally, 70% or 80% of all data is allocated to training. This method is suitable for large and crowded datasets. However, in small datasets, due to the limited size of training and test data, the model may not be adequately trained, and statistical problems may arise in the test data, leading to unreliable performance parameters. In the hold out method, training and test data are separated only once. When dealing with small datasets, model training may not be complete due to limited data availability. Performance evaluation aims to utilize the maximum amount of data for the training process, as the more training data available, the better the model learns. Conversely, it is more useful to use the maximum amount of data for testing to ensure statistically reliable results. However, the maximum data capacity is limited to the entire dataset. Thus, it is not possible to use the maximum data level for both testing and training since the same data cannot be utilized for both purposes. In the hold out method of the current study, training and testing data were separated using a 70% to 30% ratio, respectively.

In the leave-one-out validation method [54], which carries the same k-fold logic, there is 1 sample in each fold instead of several samples. The advantage of this method is that it allows the dataset to be used in the most beneficial way in small datasets. Calculating the performance for each stage does not make sense because there is only 1 sample in the test. Thanks to the test labels, after the prediction process is completed, the real labels will be compared with the prediction labels, and the performance will be calculated.

In the random subsampling model, the train and test process will repeat k times. Randomly, some are allocated as testing, and some as training, and the performance of each stage is calculated separately. This method has many repetitions. k is generally chosen as 100 or 200 [55]. Since the EEG data division process is random in this model, it will obtain even more reliable predictions. The use of a single training-testing in validation methods can sometimes be undesirable. Single train-test applications can diminish classifier performance by increasing the likelihood of bias and variability. The random subsampling verification method has acceptable performance in eliminating possible situations. Training and testing data were separated by 70% and 30% ratios, respectively.

Extensive use of the dataset is possible by using some techniques. k fold validation [56] is a special and common method that uses the maximum level of the data to test. Approximately 90% of the data is reserved for training. It allocates a small portion of the data for testing, but in fact, it uses the maximum level of the data for testing. It divides the data into k equal parts. k has a gradual repetition process. In article research, k is generally chosen as 5, 10, 15, 20 [57]. The data is divided into k parts: 1 is used for testing, and (k-1) is used for training. The method will continue until all folds are used as tests once. After taking the performance of each step separately, the average performance of all steps is calculated. For example, if we have 100 data, only 30 are used for testing in the hold out method. Nevertheless, k-fold will pass all 100 tests step by step. In the k-fold verification method, k = 5 was chosen for all stages.

4. Results and Discussion

With time and frequency domain feature extraction methods, KNN classification result percentages were calculated for two values of K, representing the number of neighbors.

In the tables,

- A/C and A/D represent healthy subjects (recording with eyes open) vs. subjects with epilepsy without seizures.
- A/E shows healthy subjects (recording with eyes open) vs. subjects with epilepsy with seizures.
- B/C and B/D show healthy subjects (recording with eyes closed) vs. subjects with epilepsy without seizures.
- B/E represents healthy subjects (recording with eyes closed) vs. subjects with epilepsy and seizures.

Considering six binary classes for classification, the accuracy, sensitivity, and specificity for the four evaluation models are presented in Tables 3, 4, 5, and 6 for the time and frequency features for the number of neighbors K = 3 and K = 5, respectively. To avoid the crowding of figure presentations, the confusion matrix for a single strategy method in both time and frequency domain feature extraction has been added for six binary classes (leave one out for time domain, and the hold out for frequency domain). The confusion matrix for the six binary classes under the leave one out and the hold out strategies for the KNN algorithm (K=5) based on the features extracted in the time and frequency domain is given in Figures 9 and 10 respectively.

Table 3. KNN Classification Results (K=3) for Time Domain Feature Extraction Methods

Model evaluation	Performance parameters	A/C	A/D	A/E	B/C	B/D	B/E
The holdout	Accuracy	98.34	95	98.34	100	98.34	98.34
	Sensitivity	100	100	100	100	100	100
	Specificity	96.67	90	96.67	100	96.67	96.67
Leave one out	Accuracy	71.5	62.5	100	57	51	96.5
	Sensitivity	77	70	100	57	51	97
	Specificity	66	55	100	57	51	96
Random subsampling	Accuracy	66.31	60.23	100	58.01	54.45	97.13
	Sensitivity	70.93	66.95	100	56.84	55.35	96.97
	Specificity	62.21	54.38	100	59.82	54.18	97.42
k-fold	Accuracy	92.5	89.5	99.5	97	96	99.5
	Sensitivity	88	87	100	95	93	100
	Specificity	97	92	99	99	99	99

Table 4 KNN Classification Results (K=5) for Time Domain Feature Extraction Methods

Model evaluation	Performance parameters	A/C	A/D	A/E	B/C	B/D	B/E
The holdout	Accuracy	96.67	93.34	98.34	100	98.34	98.34
	Sensitivity	100	100	100	100	100	100
	Specificity	93.34	86.67	96.67	100	96.67	96.67
Leave one out	Accuracy	68	58.5	100	59.5	56	96.5
	Sensitivity	76	68	100	59	58	97
	Specificity	60	49	100	60	54	96
Random subsampling	Accuracy	62.65	57.3	99.98	58.65	55.73	97.05
	Sensitivity	69.06	63.1	100	59.85	63.35	97.37
	Specificity	56.86	51.98	99.97	58.34	48.77	96.77
k-fold	Accuracy	92.5	88	99.5	97	96	99.5
	Sensitivity	89	87	100	95	93	100
	Specificity	96	89	99	99	99	99

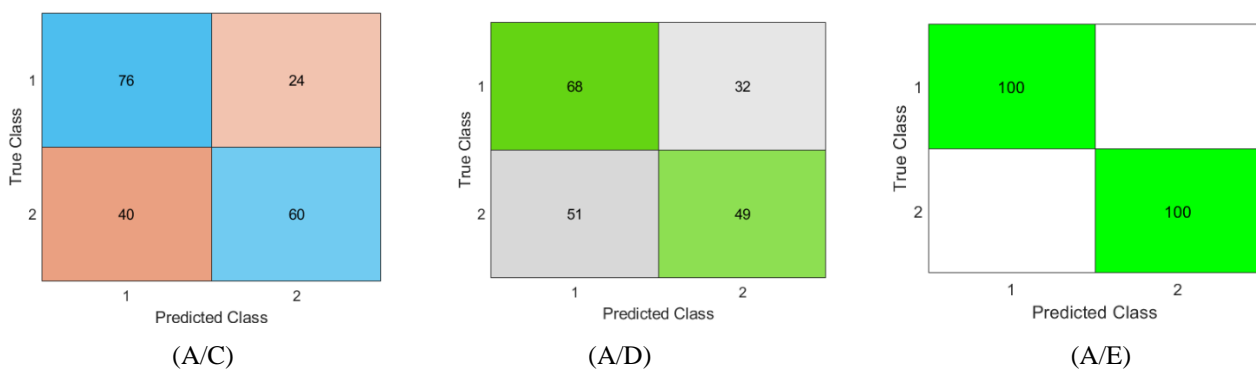
Table 5. KNN Classification Results (K=3) for Frequency Domain Feature Extraction Methods

Model evaluation	Performance parameters	A/C	A/D	A/E	B/C	B/D	B/E
The holdout	Accuracy	98.34	91.67	98.34	100	98.34	98.34
	Sensitivity	100	100	100	100	100	100
	Specificity	96.67	83.34	96.67	100	96.67	96.67
Leave one out	Accuracy	95.5	94	99.5	98.5	98	99.5
	Sensitivity	94	94	100	98	97	100
	Specificity	97	94	99	99	99	99
Random subsampling	Accuracy	94.46	92.88	99.5	98.53	97.83	99.35
	Sensitivity	92.78	93.34	100	98.07	96.2	100
	Specificity	96.16	92.68	99	99	99.5	98.68
k-fold	Accuracy	94.5	93.5	99.5	97.5	96.5	99.5
	Sensitivity	92	93	100	96	94	100
	Specificity	97	94	99	99	99	99

Table 6. KNN Classification Results (K=5) for Frequency Domain Feature Extraction Methods

Model evaluation	Performance parameters	A/C	A/D	A/E	B/C	B/D	B/E
The holdout	Accuracy	98.34	90	98.34	100	98.34	98.34
	Sensitivity	100	100	100	100	100	100
	Specificity	96.67	80	96.67	100	96.67	96.67
Leave one out	Accuracy	94.5	93	99.5	98.5	98	99.5
	Sensitivity	93	94	100	98	96	100
	Specificity	96	92	99	99	100	99
Random subsampling	Accuracy	93.76	91.13	99.43	98.51	97.61	99.51
	Sensitivity	92.96	94.44	100	97.82	95.86	99.97
	Specificity	94.79	88.11	98.87	99.17	99.4	99.02
k-fold	Accuracy	93	91	99.5	98	96	99.5
	Sensitivity	91	93	100	97	94	100
	Specificity	95	89	99	99	98	99

One of the biggest problems with machine learning model design is the issue of overfitting and underfitting. When training the model, the problem of overfitting occurs when maximum fit is achieved on the training data but generalization is not made to newly seen data. This problem was minimized by choosing the correct parameters in our proposed model. Thus, when we look at the results presented, except for the leave-one-out and random subsampling strategies in the time domain, the fit shown on the training data in other analyses shows the same consistent performance on the test data. Underfitting includes low accuracy on both training and validation/testing data. In our proposed model, it is aimed to address the issues of overfitting and underfitting by providing an adequate balance between information generalization and model complexity.



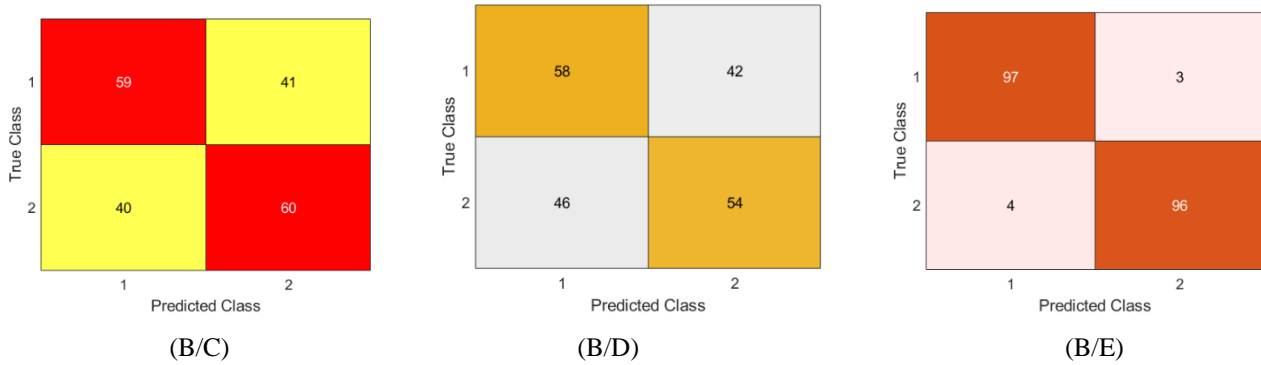


Figure 9. The Confusion Matrix for the Six Binary Classes Under the Leave One Out Strategy for the KNN Algorithm (K=5) (Time Domain Features)

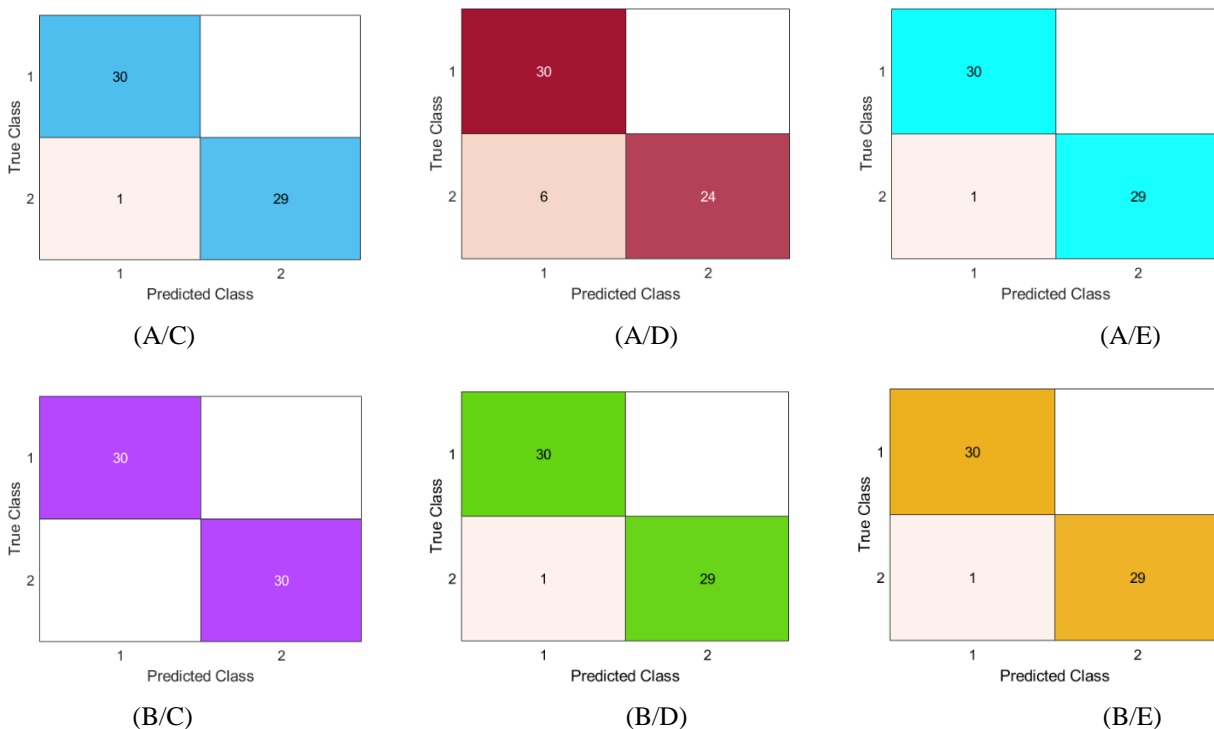


Figure 10. The Confusion Matrix for the Six Binary Classes Under the Holdout Strategy for the KNN Algorithm (K=5) (Frequency Domain Features)

In general, there was a high degree of agreement between accuracy, sensitivity, and specificity in all analyses. Tables can be evaluated from two perspectives; the first is the high accuracy evaluation between classes, and the second is the successful model evaluation of each class in model performance methods.

Much detailed research has been conducted on the dataset we are examining. We aim to simplify the research process by guiding graduate students who are new to the field of machine learning and pattern recognition. In this study, we evaluated machine learning models using time and frequency domain features, employing four performance evaluation strategies during the classification phase to distinguish between epilepsy and healthy subjects. Based on the findings from our literature review, our classification model closely aligns with previously reported results. We demonstrate the success of our proposed model through metrics such as classification accuracy, sensitivity, specificity, and the confusion matrix.

From the first perspective, the highest achievement for each accuracy row is underlined in the tables. In this evaluation, the percentage of the most successful class in 6 binary class problems shows a very good agreement with time and frequency domain feature extraction and different K neighbor numbers. From a general perspective, leave one out and random subsampling, which had poor performance in extracting time domain features, experienced a significant increase in success with frequency domain features. As expected, successful classification was achieved in all model performances due to classification with seizure epilepsy, i.e., E, in healthy EEG sets A and B due to the pattern clearly displayed in the waveform.

To compare the classification result in a proper and understandable manner, the accuracy success percentages for each evaluation model are shown in detail in Figures 11, 12, 13, and 14 for the hold out, the leave one out, random subsampling, and k-fold methods, respectively.

In Figure 11, the B/C healthy and seizure-free epilepsy classification has achieved 100% success in the hold out validation model in the KNN classification made with both time and frequency domain feature methods. On the other hand, in the healthy and seizure epilepsy classification analysis (A/E, B/E), the proposed feature extraction techniques achieved classification with a rate of 98.34%. The class that had the lowest success in the binary class groups was obtained as 90% in the classification of healthy (recording with eyes open) and non-seizure epilepsy, with the frequency domain feature extraction technique in KNN with the number of neighbors $K = 5$. In this way, frequency domain analysis provides a more successful outcome than time analysis in A/C classification. However, the time domain yields a more successful result in A/D classification.

In Figures 12 and 13, the highest success belongs to the healthy and seizure epilepsy classification (A/E, B/E). For this high success, time and frequency domain feature extraction techniques demonstrated approximately similar performance. Here, it cannot be overlooked that the number of K neighbors for KNN sometimes directly influences the classification outcome. The K parameter can be selected optimally using a few trial-and-error methods. In both time and frequency domains, the number of neighbors of $K = 5$ provides a better accuracy rate in many classes than $K = 3$. For the same number of K neighbors ($K=5$), the feature extraction method in two domains yields similar results in the leave-one-out model evaluation. In general terms, the lowest success is also evident in the healthy and seizure-free epilepsy classification (A/D, B/D).

In the k-fold model evaluation in Figure 14, the time and frequency domain classification results offer similar results in all binary classes, such as the retention model. Healthy and seizure-free epilepsy classification (A/D) has a lower success rate compared to other classes. The highest success belongs to the healthy and seizure epilepsy classification (A/E, B/E), with a success rate of 99.5%. The effect of changing the K parameter in the KNN algorithm manifests itself in a small amount in this model for the frequency domain and $K = 5$. If a general result is summarized, the hold out and k-fold cross-validation models distinguish healthy and epileptic conditions with a higher success rate than the other two models. The success of k-fold has already been proven in many research articles [58]. In the hold out model evaluation strategy, KNN demonstrates 100% success in both the time and frequency domains in the B/C subgroup classification across all classification tables. As known, in this strategy, the dataset is divided into training and testing sets. The model is trained on the training set and evaluated on the test set. In this method, whose performance varies depending on how the data is divided [59], the division ratio that yields the best results is selected using a trial-and-error method. This selection results in the highest success rates in both the time and frequency domains.

This educational and small-scale EEG study revealed that, based on the figures, choosing the right cross-validation method depends on many factors, such as the calculation steps, the nature and size of the dataset, and the ideal level of sensitivity in the designed model.

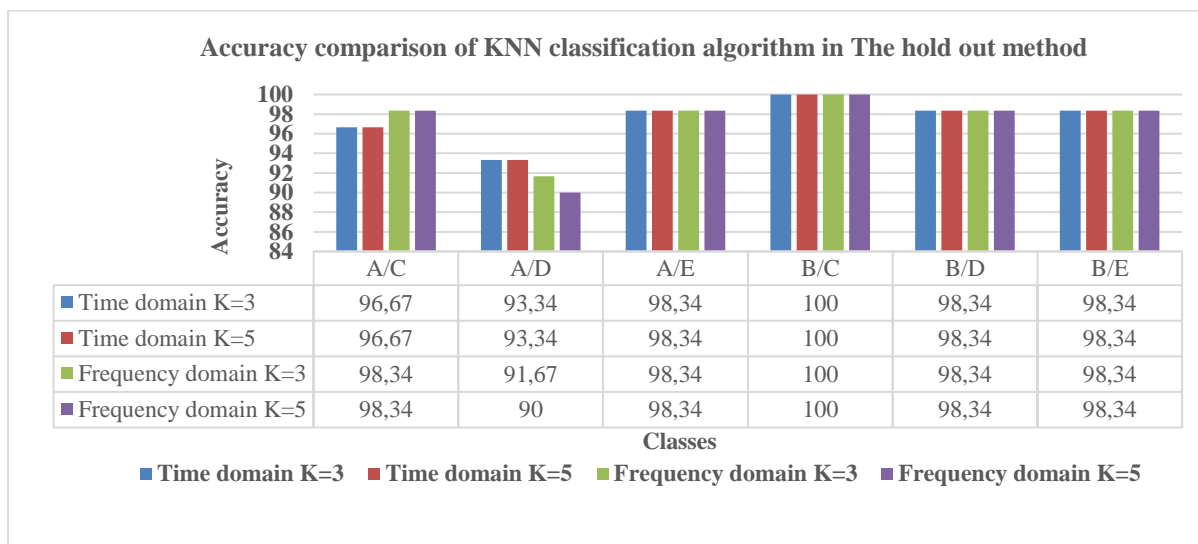


Figure 11. Accuracy Comparison of the KNN Classification Algorithm in the Hold Out Method

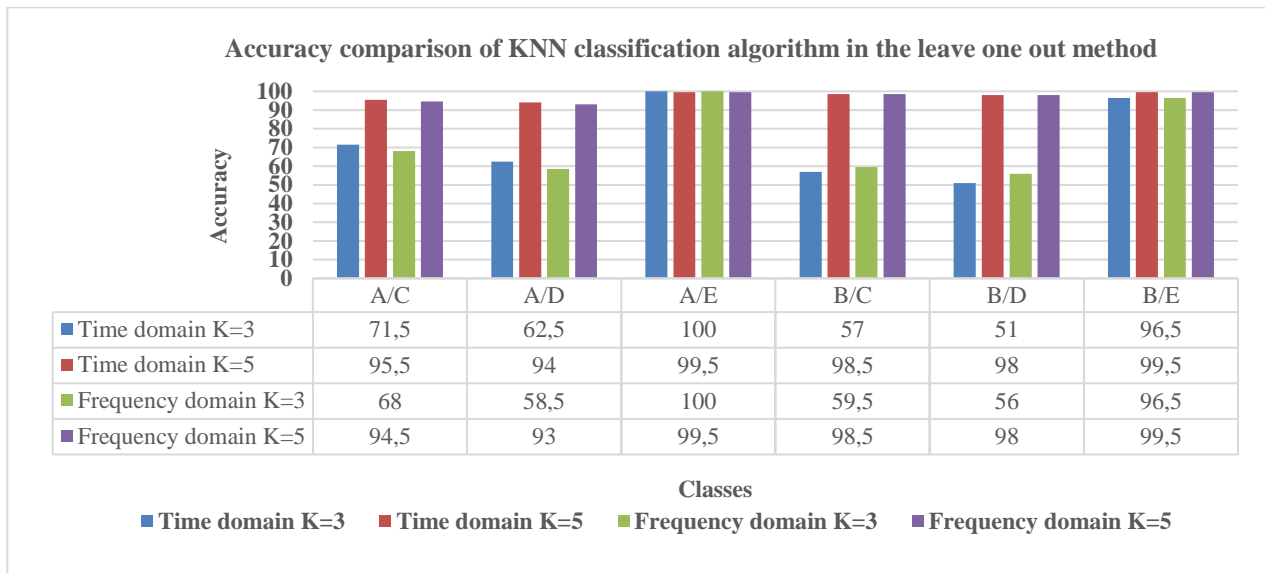


Figure 12. Accuracy Comparison of the KNN Classification Algorithm in the Leave One Out Method

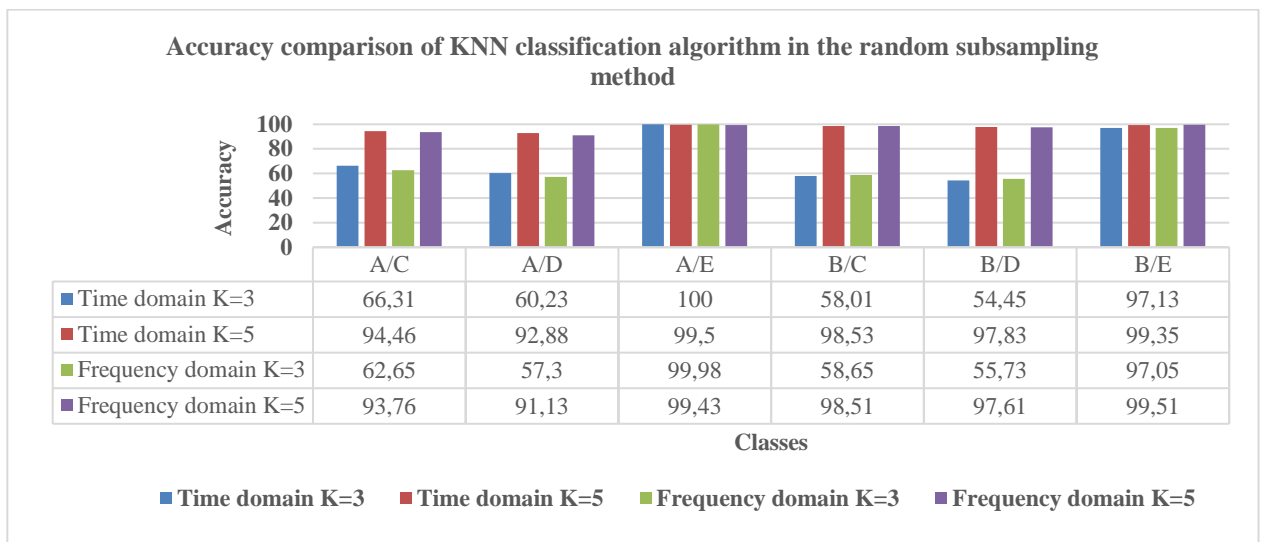


Figure 13. Accuracy Comparison of the KNN Classification Algorithm in the Random Subsampling Method

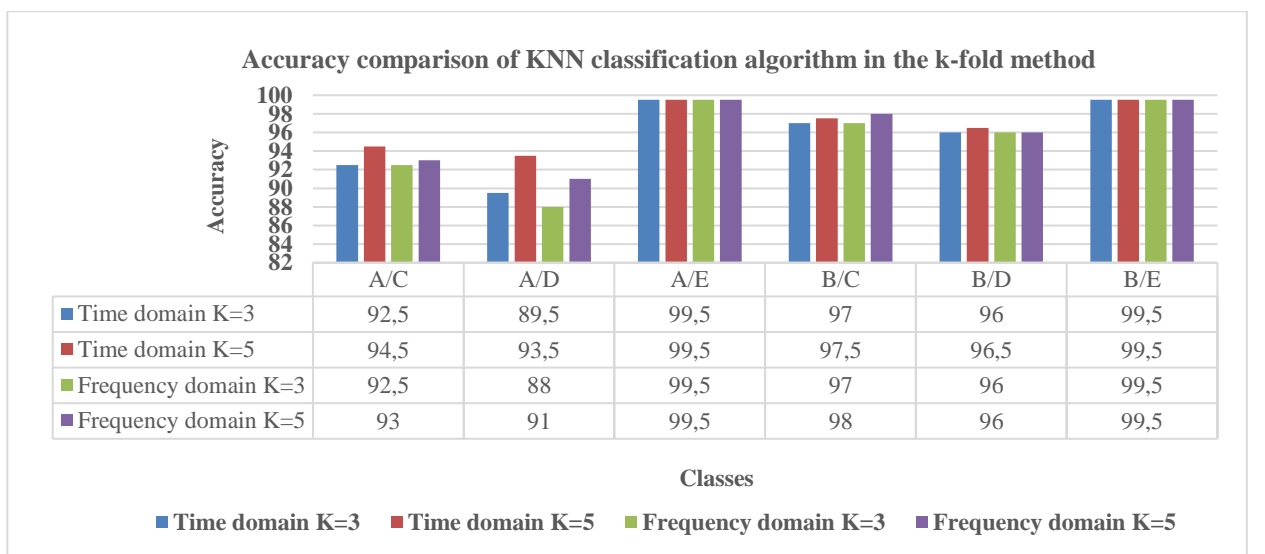


Figure 14. Accuracy Comparison of the KNN Classification Algorithm in the K-Fold Method

The dataset of the University of Bonn, which has balanced classes [60], continues to establish strong ties between the medical world and machine learning through its application for diagnosing many EEG-based epileptic diseases. Based on a literature review, the final results of this educational study are largely compatible with the findings obtained so far. In the machine learning-based epilepsy study conducted in 2021, a high success rate was achieved with the KNN algorithm using the PCA feature reduction technique [61]. After the dominant features are obtained, the KNN algorithm, which is useful in detecting abnormalities in epileptic seizures in EEG signals, achieved 97.5% success thanks to the features obtained by the wavelet transform feature extraction technique [62]. Considering the statistical feature extraction method, which has been used in many studies and has provided successful results, in the proposed high-performance machine learning model, energy, Shannon entropy, and variance significantly contribute to highlighting the dominant feature in seizure capture [63]. In a comprehensive 2021 study, various linear and non-linear features were extracted using the Bonn and Freiburg [64] datasets, and acceptable results were obtained with 10-fold cross-validation thanks to the deep learning approach. The proposed model demonstrated 99.71% and 99.13% accuracy for the Bonn and Freiburg datasets [65]. In a study conducted in 2023 with the Bonn dataset [66], an automatic epilepsy seizure detection model was designed. In this model, classification was carried out by applying a discrete wavelet transform, extracting four mixed features, and using a convolutional neural network algorithm. The classification success of this designed model was reported as 100%. Introducing a new feature extraction method [67], complex network logic was used for automatic epilepsy seizure detection. In this study, Henon and Logistic maps in the Bonn dataset were considered to demonstrate the validity of the methodology. In the detailed study, epilepsy seizure diagnosis was presented with 100% accuracy. In this epilepsy seizure diagnosis model, which is useful to expert neurologists, successful performance was demonstrated using different classifiers such as support vector machine and linear discriminant analysis. In a multiple classification study conducted in 2023 using deep learning techniques, epileptic seizures were diagnosed in the Bonn dataset [68]. Seizure diagnosis was achieved with a 99.5% success rate using scalogram and spectrogram images and multiple classification techniques.

References

- [1] N. Manshouri, M. Maleki, and T. Kayikcioglu, "An EEG-based stereoscopic research of the PSD differences in pre and post 2D&3D movies watching," *Biomed Signal Process Control*, vol. 55, Jan. 2020, doi: 10.1016/j.bspc.2019.101642.
- [2] M. Soufineyestani, D. Dowling, and A. Khan, "Electroencephalography (EEG) Technology Applications and Available Devices," *Applied Sciences 2020, Vol. 10, Page 7453*, vol. 10, no. 21, p. 7453, Oct. 2020, doi: 10.3390/AP10217453.
- [3] M. Melek, N. Manshouri, and T. Kayikcioglu, "Low-Cost Brain-Computer Interface Using the Emotiv Epoc Headset Based on Rotating Vanes," *Traitement du Signal*, vol. 37, no. 5, pp. 831–837, Nov. 2020, doi: 10.18280/ts.370516.
- [4] E. Maiorana, "Deep learning for EEG-based biometric recognition," *Neurocomputing*, vol. 410, pp. 374–386, Oct. 2020, doi: 10.1016/J.NEUCOM.2020.06.009.
- [5] P. Arnau-Gonzalez, S. Katsigiannis, M. Arevalillo-Herraez, and N. Ramzan, "BED: A New Data Set for EEG-Based Biometrics," *IEEE Internet Things J*, vol. 8, no. 15, pp. 12219–12230, Aug. 2021, doi: 10.1109/JIOT.2021.3061727.
- [6] M. Maleki and T. Kayikcioglu, "A new brain-computer interface system using the gaze on rotating vane.," *Biomedical Research-tokyo*, 2016.
- [7] B. T. Klassen *et al.*, "Quantitative EEG as a predictive biomarker for Parkinson disease dementia," *Neurology*, vol. 77, no. 2, pp. 118–124, Jul. 2011, doi: 10.1212/WNL.0B013E318224AF8D/SUPPL_FILE/KLASSEN.PDF.
- [8] C. Melissant, A. Ypma, E. E. E. Frijman, and C. J. Stam, "A method for detection of Alzheimer's disease using ICA-enhanced EEG measurements," *Artif Intell Med*, vol. 33, no. 3, pp. 209–222, Mar. 2005, doi: 10.1016/J.ARTMED.2004.07.003.
- [9] E. Cainelli, L. Vedovelli, B. Carretti, and P. Bisiacchi, "EEG correlates of developmental dyslexia: a systematic review," *Annals of Dyslexia 2022 73:2*, vol. 73, no. 2, pp. 184–213, Nov. 2022, doi: 10.1007/S11881-022-00273-1.
- [10] A. R. Clarke, R. J. Barry, S. J. Johnstone, R. McCarthy, and M. Selikowitz, "EEG development in Attention Deficit Hyperactivity Disorder: From child to adult," *Clinical Neurophysiology*, vol. 130, no. 8, pp. 1256–1262, Aug. 2019, doi: 10.1016/J.CLINPH.2019.05.001.
- [11] R. J. Barry, A. R. Clarke, S. J. Johnstone, R. McCarthy, and M. Selikowitz, "Electroencephalogram theta/beta ratio and arousal in attention-deficit/hyperactivity disorder: evidence of independent processes," *Biol Psychiatry*, vol. 66, no. 4, pp. 398–401, Aug. 2009, doi: 10.1016/J.BIOPSYCH.2009.04.027.
- [12] C. Baumgartner and J. P. Koren, "Seizure detection using scalp-EEG," *Epilepsia*, vol. 59, pp. 14–22, Jun. 2018, doi: 10.1111/EPI.14052.
- [13] S. Nasehi and H. Pourghassem, "Seizure detection algorithms based on analysis of EEG and ECG signals: A survey," *Neurophysiology*, vol. 44, no. 2, pp. 174–186, Jun. 2012, doi: 10.1007/S11062-012-9285-X/METRICS.
- [14] T. Rowberry *et al.*, "Implementation and Early Evaluation of a Quantitative Electroencephalography Program for Seizure Detection in the PICU*," *Pediatric Critical Care Medicine*, vol. 21, no. 6, pp. 543–549, Jun. 2020, doi: 10.1097/PCC.0000000000002278.
- [15] E. Milne, R. Gomez, A. Giannadou, and M. Jones, "Atypical EEG in autism spectrum disorder: Comparing a dimensional and a categorical approach," *J Abnorm Psychol*, vol. 128, no. 5, pp. 442–452, Jul. 2019, doi:

- 10.1037/ABN0000436.
- [16] M. E. Santarone *et al.*, “EEG Features in Autism Spectrum Disorder: A Retrospective Analysis in a Cohort of Preschool Children,” *Brain Sciences* 2023, Vol. 13, Page 345, vol. 13, no. 2, p. 345, Feb. 2023, doi: 10.3390/BRAINSKI13020345.
- [17] M. M. Siddiqui, G. Srivastava, and S. H. Saeed, “Diagnosis of insomnia sleep disorder using short time frequency analysis of PSD approach applied on EEG signal using channel ROC-LOC,” *Sleep Science*, vol. 9, no. 3, pp. 186–191, Jul. 2016, doi: 10.1016/J.SLSCI.2016.07.002.
- [18] C. Spironelli, M. Manfredi, and A. Angrilli, “Beta EEG band: A measure of functional brain damage and language reorganization in aphasic patients after recovery,” *Cortex*, vol. 49, no. 10, pp. 2650–2660, Nov. 2013, doi: 10.1016/J.CORTEX.2013.05.003.
- [19] S. Ballanti *et al.*, “EEG-based methods for recovery prognosis of patients with disorders of consciousness: A systematic review,” *Clinical Neurophysiology*, vol. 144, pp. 98–114, Dec. 2022, doi: 10.1016/J.CLINPH.2022.09.017.
- [20] J. A. Micoulaud-Franchi, C. Jeunet, A. Pelissolo, and T. Ros, “EEG Neurofeedback for Anxiety Disorders and Post-Traumatic Stress Disorders: A Blueprint for a Promising Brain-Based Therapy,” *Curr Psychiatry Rep*, vol. 23, no. 12, pp. 1–14, Dec. 2021, doi: 10.1007/S11920-021-01299-9/FIGURES/5.
- [21] D. A. Moscovitch, D. L. Santesso, V. Miskovic, R. E. McCabe, M. M. Antony, and L. A. Schmidt, “Frontal EEG asymmetry and symptom response to cognitive behavioral therapy in patients with social anxiety disorder,” *Biol Psychol*, vol. 87, no. 3, pp. 379–385, Jul. 2011, doi: 10.1016/J.BIOPSYCHO.2011.04.009.
- [22] E. Netzer, A. Frid, and D. Feldman, “Real-time EEG classification via coresets for BCI applications,” *Eng Appl Artif Intell*, vol. 89, p. 103455, Mar. 2020, doi: 10.1016/J.ENGAPPAL.2019.103455.
- [23] M. Shen, P. Wen, B. Song, and Y. Li, “Real-time epilepsy seizure detection based on EEG using tunable-Q wavelet transform and convolutional neural network,” *Biomed Signal Process Control*, vol. 82, p. 104566, Apr. 2023, doi: 10.1016/J.BSPC.2022.104566.
- [24] T. A. Milligan, “Epilepsy: A Clinical Overview,” *Am J Med*, vol. 134, no. 7, pp. 840–847, Jul. 2021, doi: 10.1016/J.AMJMED.2021.01.038.
- [25] T. Adewumi, E. Oladipo, and A. O. Adewuya, “Public perception and attitude towards people living with epilepsy in Nigeria,” *Epilepsy & Behavior*, vol. 106, p. 107033, May 2020, doi: 10.1016/J.YEBEH.2020.107033.
- [26] S. Ibrahim, R. Djemal, and A. Alsuwailem, “Electroencephalography (EEG) signal processing for epilepsy and autism spectrum disorder diagnosis,” *Biocybern Biomed Eng*, vol. 38, no. 1, pp. 16–26, Jan. 2018, doi: 10.1016/J.BBE.2017.08.006.
- [27] W. O. Tatum *et al.*, “Clinical utility of EEG in diagnosing and monitoring epilepsy in adults,” *Clinical Neurophysiology*, vol. 129, no. 5, pp. 1056–1082, May 2018, doi: 10.1016/J.CLINPH.2018.01.019.
- [28] F. Rosenow, K. M. Klein, and H. M. Hamer, “Non-invasive EEG evaluation in epilepsy diagnosis,” *Expert Rev Neurother*, vol. 15, no. 4, pp. 425–444, Apr. 2015, doi: 10.1586/14737175.2015.1025382.
- [29] C. A. M. Guerreiro, “Epilepsy: Is there hope?,” *Indian J Med Res*, vol. 144, no. 5, p. 657, Nov. 2016, doi: 10.4103/IJMR.IJMR_1051_16.
- [30] M. Diykh, Y. Li, and P. Wen, “Classify epileptic EEG signals using weighted complex networks based community structure detection,” *Expert Syst Appl*, vol. 90, pp. 87–100, Dec. 2017, doi: 10.1016/J.ESWA.2017.08.012.
- [31] A. K. Jaiswal and H. Banka, “Epileptic seizure detection in EEG signal using machine learning techniques,” *Australas Phys Eng Sci Med*, vol. 41, no. 1, pp. 81–94, Mar. 2018, doi: 10.1007/S13246-017-0610-Y/TABLES/12.
- [32] H. U. Amin, M. Z. Yusoff, and R. F. Ahmad, “A novel approach based on wavelet analysis and arithmetic coding for automated detection and diagnosis of epileptic seizure in EEG signals using machine learning techniques,” *Biomed Signal Process Control*, vol. 56, p. 101707, Feb. 2020, doi: 10.1016/J.BSPC.2019.101707.
- [33] M. S. Farooq, A. Zulfiqar, and S. Riaz, “Epileptic Seizure Detection Using Machine Learning: Taxonomy, Opportunities, and Challenges,” *Diagnostics* 2023, Vol. 13, Page 1058, vol. 13, no. 6, p. 1058, Mar. 2023, doi: 10.3390/DIAGNOSTICS13061058.
- [34] M. T. Akhtar, W. Mitsuhashi, and C. J. James, “Employing spatially constrained ICA and wavelet denoising, for automatic removal of artifacts from multichannel EEG data,” *Signal Processing*, vol. 92, no. 2, pp. 401–416, Feb. 2012, doi: 10.1016/J.SIGPRO.2011.08.005.
- [35] V. Harpale and V. Bairagi, “An adaptive method for feature selection and extraction for classification of epileptic EEG signal in significant states,” *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 6, pp. 668–676, Jul. 2021, doi: 10.1016/J.JKSUCI.2018.04.014.
- [36] T. Zhang, W. Chen, and M. Li, “Generalized Stockwell transform and SVD-based epileptic seizure detection in EEG using random forest,” *Biocybern Biomed Eng*, vol. 38, no. 3, pp. 519–534, Jan. 2018, doi: 10.1016/J.BBE.2018.03.007.
- [37] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, “Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state,” *Phys Rev E*, vol. 64, no. 6, p. 061907, Nov. 2001, doi: 10.1103/PhysRevE.64.061907.
- [38] A. K. Singh and S. Krishnan, “Trends in EEG signal feature extraction applications,” *Front Artif Intell*, vol. 5, p. 1072801, Jan. 2023, doi: 10.3389/FRAI.2022.1072801/BIBTEX.
- [39] I. Stancin, M. Cifrek, and A. Jovic, “A Review of EEG Signal Features and Their Application in Driver Drowsiness

- Detection Systems,” *Sensors (Basel)*, vol. 21, no. 11, Jun. 2021, doi: 10.3390/S21113786.
- [40] H. Choubey and A. Pandey, “A combination of statistical parameters for the detection of epilepsy and EEG classification using ANN and KNN classifier,” *Signal Image Video Process*, vol. 15, no. 3, pp. 475–483, Apr. 2021, doi: 10.1007/S11760-020-01767-4/TABLES/7.
- [41] J. Xiang *et al.*, “Kurtosis and skewness of high-frequency brain signals are altered in paediatric epilepsy,” *Brain Commun*, vol. 2, no. 1, Jan. 2020, doi: 10.1093/BRAINCOMMS/FCAA036.
- [42] W. Garcia-González, W. Flores-Fuentes, O. Sergiyenko, J. C. Rodríguez-Quiñonez, J. E. Miranda-Vega, and D. Hernández-Balbuena, “Shannon Entropy Used for Feature Extractions of Optical Patterns in the Context of Structural Health Monitoring,” *Entropy 2023, Vol. 25, Page 1207*, vol. 25, no. 8, p. 1207, Aug. 2023, doi: 10.3390/E25081207.
- [43] A. Mert and A. Akan, “Seizure onset detection based on frequency domain metric of empirical mode decomposition,” *Signal Image Video Process*, vol. 12, no. 8, pp. 1489–1496, Nov. 2018, doi: 10.1007/S11760-018-1304-Y/TABLES/2.
- [44] M. L. Vicchietti, F. M. Ramos, L. E. Betting, and A. S. L. O. Campanharo, “Computational methods of EEG signals analysis for Alzheimer’s disease classification,” *Scientific Reports 2023 13:1*, vol. 13, no. 1, pp. 1–14, May 2023, doi: 10.1038/s41598-023-32664-8.
- [45] M. M. Lansbergen, M. Van Dongen-Boomsma, J. K. Buitelaar, and D. Slaats-Willemse, “ADHD and EEG-neurofeedback: A double-blind randomized placebo-controlled feasibility study,” *J Neural Transm*, vol. 118, no. 2, pp. 275–284, Feb. 2011, doi: 10.1007/S00702-010-0524-2/FIGURES/1.
- [46] S. Abenna, M. Nahid, H. Bouyghf, and B. Ouacha, “EEG-based BCI: A novel improvement for EEG signals classification based on real-time preprocessing,” *Comput Biol Med*, vol. 148, Sep. 2022, doi: 10.1016/J.COMPBIOMED.2022.105931.
- [47] M. Melek, “Diagnosis of COVID-19 and non-COVID-19 patients by classifying only a single cough sound,” *Neural Computing and Applications 2021*, pp. 1–12, Jul. 2021, doi: 10.1007/S00521-021-06346-3.
- [48] M. Melek *et al.*, “An automatic EEG-based sleep staging system with introducing NAO SP and NAO GP as new metrics for sleep staging systems,” *Cogn Neurodyn*, vol. 15, no. 3, pp. 405–423, Jun. 2021, doi: 10.1007/S11571-020-09641-2.
- [49] T. Kayikcioglu, M. Maleki, and K. Eroglu, “Fast and accurate PLS-based classification of EEG sleep using single channel data,” *Expert Syst Appl*, vol. 42, no. 21, pp. 7825–7830, Jun. 2015, doi: 10.1016/j.eswa.2015.06.010.
- [50] S. Uddin, I. Haque, H. Lu, M. A. Moni, and E. Gide, “Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction,” *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/S41598-022-10358-X.
- [51] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, “A novel kNN algorithm with data-driven k parameter computation,” *Pattern Recognit Lett*, vol. 109, pp. 44–54, Jul. 2018, doi: 10.1016/J.PATREC.2017.09.036.
- [52] P. K. Syriopoulos, N. G. Kalampalikis, S. B. Kotsiantis, and M. N. Vrahatis, “kNN Classification: a review,” *Ann Math Artif Intell*, pp. 1–33, Sep. 2023, doi: 10.1007/S10472-023-09882-X/METRICS.
- [53] G. Mezzadri, T. Laloë, F. Mathy, and P. Reynaud-Bouret, “Hold-out strategy for selecting learning models: Application to categorization subjected to presentation orders,” *J Math Psychol*, vol. 109, p. 102691, Aug. 2022, doi: 10.1016/J.JMP.2022.102691.
- [54] H. Cheng, D. J. Garrick, and R. L. Fernando, “Efficient strategies for leave-one-out cross validation for genomic best linear unbiased prediction,” *J Anim Sci Biotechnol*, vol. 8, no. 1, pp. 1–5, May 2017, doi: 10.1186/S40104-017-0164-6/TABLES/5.
- [55] J. P. Cruz-Tirado, J. M. Amigo, D. F. Barbin, and S. Kucheryavskiy, “Data reduction by randomization subsampling for the study of large hyperspectral datasets,” *Anal Chim Acta*, vol. 1209, p. 339793, May 2022, doi: 10.1016/J.ACA.2022.339793.
- [56] H. Ling, C. Qian, W. Kang, C. Liang, and H. Chen, “Combination of Support Vector Machine and K-Fold cross validation to predict compressive strength of concrete in marine environment,” *Constr Build Mater*, vol. 206, pp. 355–363, May 2019, doi: 10.1016/j.conbuildmat.2019.02.071.
- [57] N. Manshouri, M. Melek, and T. Kayikcioglu, “Detection of 2D and 3D Video Transitions Based on EEG Power,” *Comput J*, Sep. 2020, doi: 10.1093/COMJNL/BXAA116.
- [58] D. François, F. Rossi, V. Wertz, and M. Verleysen, “Resampling methods for parameter-free and robust feature selection with mutual information,” *Neurocomputing*, vol. 70, no. 7–9, pp. 1276–1288, Mar. 2007, doi: 10.1016/J.NEUCOM.2006.11.019.
- [59] C. Beleites *et al.*, “Variance reduction in estimating classification error using sparse datasets,” *Chemometrics and Intelligent Laboratory Systems*, vol. 79, no. 1–2, pp. 91–100, Oct. 2005, doi: 10.1016/J.CHEMOLAB.2005.04.008.
- [60] S. Wong *et al.*, “EEG datasets for seizure detection and prediction— A review,” *Epilepsia Open*, vol. 8, no. 2, pp. 252–267, Jun. 2023, doi: 10.1002/EPI4.12704.
- [61] S. NAHZAT and M. YAĞANOĞLU, “Classification of Epileptic Seizure Dataset Using Different Machine Learning Algorithms and PCA Feature Reduction Technique,” *Journal of Investigations on Engineering and Technology*, vol. 4, no. 2, pp. 47–60, Dec. 2021, Accessed: Feb. 14, 2024. [Online]. Available: <https://dergipark.org.tr/en/pub/jiet/issue/67435/1002958>
- [62] W. Mardini, M. M. Bani Yassein, R. Al-Rawashdeh, S. Aljawarneh, Y. Khamayseh, and O. Meqdadi, “Enhanced detection of epileptic seizure using EEG signals in combination with machine learning classifiers,” *IEEE Access*, vol.

- 8, pp. 24046–24055, 2020, doi: 10.1109/ACCESS.2020.2970012.
- [63] P. Boonyakitanont, A. Lek-uthai, K. Chomtho, and J. Songsiri, “A review of feature extraction and performance evaluation in epileptic seizure detection using EEG,” *Biomed Signal Process Control*, vol. 57, p. 101702, Mar. 2020, doi: 10.1016/J.BSPC.2019.101702.
- [64] “EEG Database — Seizure Prediction Project Freiburg.” Accessed: Feb. 14, 2024. [Online]. Available: <https://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database/>
- [65] A. Malekzadeh, A. Zare, M. Yaghoobi, H. R. Kobravi, and R. Alizadehsani, “Epileptic Seizures Detection in EEG Signals Using Fusion Handcrafted and Deep Learning Features,” *Sensors 2021, Vol. 21, Page 7710*, vol. 21, no. 22, p. 7710, Nov. 2021, doi: 10.3390/S21227710.
- [66] W. Chen *et al.*, “An automated detection of epileptic seizures EEG using CNN classifier based on feature fusion with high accuracy,” *BMC Med Inform Decis Mak*, vol. 23, no. 1, pp. 1–17, Dec. 2023, doi: 10.1186/S12911-023-02180-W/TABLES/5.
- [67] S. Supriya, S. Siuly, H. Wang, and Y. Zhang, “New feature extraction for automated detection of epileptic seizure using complex network framework,” *Applied Acoustics*, vol. 180, p. 108098, Sep. 2021, doi: 10.1016/J.APACOUST.2021.108098.
- [68] M. Varlı and H. Yılmaz, “Multiple classification of EEG signals and epileptic seizure diagnosis with combined deep learning,” *J Comput Sci*, vol. 67, p. 101943, Mar. 2023, doi: 10.1016/J.JOCS.2023.101943.

Author(s) Contributions

The article is written by a single author. I hereby declare that I have prepared the article alone for the authorship declaration.

Conflict of Interest Notice

The author declares that there are no potential conflicts of interest.

Support/Supporting Organizations

This research did not receive any specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the references are provided in the bibliography. This study used the EEG dataset purchased by Andrzejak *et al.* [37]. and belonging to the University of Bonn. In this case, the author of the cited study did not apply for ethical approval.

Availability of data and material

The datasets analyzed during the current study are available from <https://www.ukbonn.de/epileptologie/arbeitsgruppen/ag-lehnertz-neurophysik/downloads/>.

Plagiarism Statement

This article has been scanned by iThenticate™.

Prediction of Multivariate Chaotic Time Series using GRU, LSTM and RNN

Gülyeter Öztürk¹, Osman Eldoğan¹

¹Department of Mechatronics Engineering, Faculty of Technology, Sakarya University of Applied Sciences, Sakarya, Türkiye

Corresponding author:

Gülyeter Öztürk, Department of Mechatronics Engineering, Faculty of Technology, Sakarya University of Applied Sciences, Sakarya, Türkiye
gulyeterozturk@subu.edu.tr



Article History:
Received: 12.12.2023
Accepted: 22.07.2024
Published Online: 23.08.2024

ABSTRACT

Chaotic systems are identified as nonlinear, deterministic dynamic systems that exhibit sensitive dependence on initial values. Some chaotic equations modeled from daily events involve time information and generate chaotic time series that are sequential data. Through successful prediction studies conducted on the generated chaotic time series, forecasts can be made about events displaying unpredictable behavior in nature, which have not yet been modeled. This enables preparation for both favorable and unfavorable situations that may arise. In this study, chaotic time series were generated using Lorenz, Chen, and Rikitake multivariate chaotic systems. To enhance prediction accuracy on the generated data, GRU, LSTM and RNN models were trained with different hyperparameters. Subsequently, comprehensive test studies were conducted to evaluate their performance. Predictions were calculated using evaluation metrics, including MSE, RMSE, MAE, MAPE, and R^2 . In the experimental study, each chaotic system was trained with different hyperparameter combinations on six network models. The experimental results indicate that the utilized models exhibited greater success in predicting chaotic time series compared to some other models in the literature.

Keywords: Chaotic time series, Multivariate, Time series prediction, GRU, LSTM, RNN

1. Introduction

Dynamical systems that seem complex in our daily lives, yet possess internal order and are sensitive to initial conditions, are defined as chaotic systems, and these systems produce chaotic data. Sequential chaotic data contains time information and is commonly termed chaotic time series. One of the important topics that scientists focus on is conducting prediction studies based on examining past and present values of a system in a time series. Time series prediction studies are applied in real-world domains with chaotic structures, such as traffic flow [1], building energy consumption [2], finance [3], electrical load [4], meteorology [5], earthquake [6], and wind energy [7]. The success achieved in predicting challenging events like this one, along with other seemingly complex phenomena, can render many occurrences in nature foreseeable and controllable.

In studies aimed at predicting time series of linear systems, traditional statistical methods such as Autoregressive Moving Average (ARMA), Autoregressive Moving Average with Exogenous Inputs (ARMAX), and Autoregressive Integrated Moving Averages (ARIMA) were previously employed. However, these methods are found to be insufficient when dealing with nonlinear complex systems [8], [9]. As a response, researchers have explored machine learning, deep learning, and hybrid methods to predict data from nonlinear and complex systems. In their study on chaotic time series prediction using both the noisy and noiseless Lorenz system, Karunasinghe and Liong found that artificial neural network models outperformed local prediction models [10]. Yuxia and Hongtao attempted to predict the time series data of the Lorenz system using a support vector machine with a chaos optimization algorithm. In this study focused on predicting the x state variable of the Lorenz chaotic system, the researchers obtained a root mean square error (RMSE) value of 0.0030335 [11]. The authors, utilizing a NARX neural network in MATLAB, employed 2100 time series data generated from the Lorenz chaotic system for prediction, with RMSE as the evaluation metric [12]. In prediction studies of multivariate chaotic time series, using all variables that constitute the system, rather than employing a single variable, enables obtaining more information about the dynamic system. In their studies, Xiu and Zhang found, in both single and multiple variable analyses, that more accurate predictions were achieved using multiple variables. Addressing the significance of predicting time series data in economics, business, and finance, Siami and Namin demonstrated in their study that the Long Short-Term Memory (LSTM) deep learning model outperforms the ARIMA model [13].

Deep learning models, known for their superior ability to capture nonlinear relationships in large datasets compared to traditional machine learning methods, have gained widespread popularity in time series prediction studies in recent years.

Due to the inherent time relationship in time series data, where the current data point is connected to both past and future data points, the Recurrent Neural Network (RNN) deep learning model, designed with memory capabilities, is commonly utilized to establish and maintain these relationships. RNN and its variations have been predominantly utilized in time series prediction studies across various fields, including finance, energy, solar radiation, and air quality, in different years [2], [3], [14-16]. The frequency of usage of RNN variations in these studies follows the order of LSTM, ELMAN (simple RNN cell) [17], and Gated Recurrent Units (GRU) [18], respectively. Sezer and his colleagues reported that over half of the publications in the field of finance from 2005 to 2019 utilized RNN and its variations for time series prediction. The utilization rates of models in prediction studies in the field of finance are 9.89% for GRU, 29.7% for ELMAN (vanilla RNN), and 60.4% for LSTM [3]. Dudukcu and his team stated in their literature review that Elman RNN, LSTM, Convolutional Neural Networks (CNN), and Temporal Convolutional Networks (TCN) are commonly used in time series prediction studies. They created a dataset for each of the x, y, and z state variables of Lorenz, Rossler, and Lorenz-like chaotic systems. They also possess real-world data in the form of an electrocardiogram dataset from 21 patients. They carried out time series predictions using hyperparameters determined through the grid search method for the entire dataset [9]. Researchers, evaluating the performance of different optimized RNN cell structures on various time series datasets, have introduced a new RNN variation called SLIM, demonstrating cost-effectiveness in terms of both time and computational resources for prediction studies [14]. Chandra and Zhang employed two distinct methods for the cooperative evolution of Elman RNN in predicting the chaotic time series of Mackey-Glass, Lorenz and Sunspot [19]. In the conducted study, the obtained values are as follows: in the Lorenz system, 0.00636 RMSE and 0.000772 normalized mean square error (NMSE); in the Mackey-Glass system, 0.00633 RMSE and 0.000279 NMSE; and in the Sunspot time series, 0.0166 RMSE and 0.00147 NMSE. The authors also demonstrated that their two implemented methods yielded superior results compared to other methods, with the exception of the Evolutionary RNN and Hybrid-NARX-Elman models. In 2020, researchers utilized a hybrid model named Complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN)-LSTM to predict the Lorenz-63 chaotic time series [20]. Utilizing a total of 5,000 data points for training and testing, along with the Adam optimizer and ReLU activation function parameters, they achieved an RMSE of 1.327 and mean absolute error (MAE) of 1.124 with their proposed model. The authors performed predictions using Support Vector Regression, ARIMA, Multilayer Perceptron, and single LSTM models with the same dataset, revealing that their proposed model exhibited superior performance.

Dudukcu and her colleagues employed four different variations of the LSTM model to predict time series data derived from three chaotic systems, which are namely Lorenz, Rossler, and Lorenz-like systems. The study utilized the grid search algorithm for hyperparameter optimization, revealing that the highest success was achieved with an RMSE value of 3.7397×10^{-5} during the validation phase of the Stacked LSTM, and an RMSE value of 0.1558 during the testing phase of the Stacked LSTM [21]. Fu and colleagues developed a hybrid model called DTIGNet, incorporating an improved temporal inception module and GRU for automatic multi-scale feature extraction. They applied DTIGNet to the Mackey-Glass, Rossler, and Lorenz chaotic systems and sunspots time series to assess its efficacy in chaotic time series forecasting. Using metrics such as MAE, RMSE, correlation coefficient (ρ), coefficient of determination (R^2), mean absolute percentage error (MAPE), and SMAPE, the authors stated that their model demonstrated higher accuracy and better performance compared to other estimation methods [22]. In a related study, Cheng and team outperformed LSTM, CNN-LSTM, and TCN models in terms of RMSE, MAE, R^2 , and ρ metrics with their designed TCN-CBAM model. They applied this model to the Lorenz system, Chen system, and sunspots dataset, achieving better performance in chaotic time series prediction [23].

When identical values are provided as inputs to mathematical models derived from natural events, the expectation is that the same results will always be produced. Given as an input to the mathematical model of chaotic systems, when the value is changed by one thousandth, a significant change occurs in the result obtained. Therefore, it is crucial to understand the method and precision with which time series are generated from chaotic systems. In the proposed study, the Lorenz system, considered the starting point in terms of chaos theory and utilized in almost all chaotic time series forecasting studies, was employed. Additionally, the Lorenz-like Chen chaotic system and the non-Lorenz-like Rikitake chaotic system were used to enhance the validity of the study. These chaotic systems were solved using the fourth-order Runge-Kutta method with a time step of 0.01. Examining numerous studies on chaotic time series prediction in the literature, it has been observed that only one of the state variables constituting the system is provided as input to the prediction models, and this variable is predicted as output. However, to make accurate predictions about a complex and unpredictable system, it is necessary to consider all variables that influence the system. This approach allows for more accurate prediction or classification results. In this study, all state variables constituting the chaotic systems are given as input to the prediction models, in an attempt to predict a single state variable with better performance. Chaotic systems trained and tested on GRU, RNN, and LSTM models with different hyperparameters were compared using the following evaluation metrics: mean square error (MSE), RMSE, MAE, MAPE, and R^2 .

The second section of this study offers a detailed explanation of the deep learning models employed, and information about the datasets obtained from chaotic systems. The third section presents the experimental results and analysis of prediction studies conducted with various hyperparameters. In the final section, the inferences derived from the experiments are stated, and recommendations for future studies are provided.

2. Methodology and Methods

2.1. Prediction Models Used

In this section, we introduce RNN, LSTM, and GRU deep learning models designed to maintain the connection between chaotic time series data, ultimately yielding successful outcomes in predicting future data.

2.1.1. RNN

RNN maintains a connection with future data by retaining information from past steps in its memory. The RNN, which has short-term memory, is successful in remembering the past and predicting the future when given short sequential input data. However, it fails to remember the past and encounters difficulty in predicting future data when given long sequential input data. The reason for this is the emergence of the vanishing gradient problem, where the gradient value diminishes significantly during backpropagation and leads to its disappearance, or the occurrence of the exploding gradient problem, where the gradient value increases excessively during backpropagation, preventing convergence to local minimum errors. To address these fundamental challenges hindering learning in RNN, LSTM and GRU models have been developed for natural language processing, speech recognition, language translation, text generation, and time series classification/prediction tasks involving sequential data. The LSTM and GRU models originate from the recurrent neural network architecture and have demonstrated success in predicting both short-term and long-term time series. The structure of the RNN is shown in Figure 1.

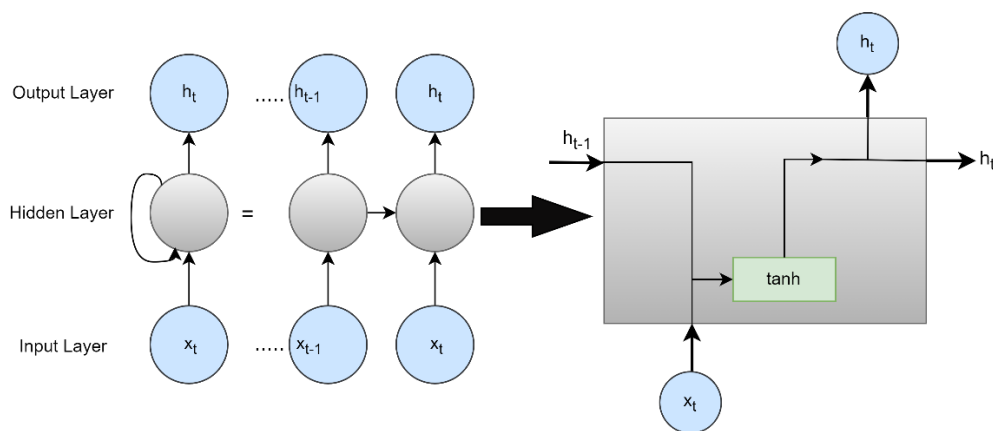


Figure 1. Structure of the RNN

2.1.2. LSTM

Each LSTM cell comprises a cell state, an input gate, a forget gate, and an output gate [24]. Temporal dependencies between dynamic nonlinear data are preserved thanks to the functions performed by the gates that determine which of the information given to the cell should be stored or forgotten. The cell state carries meaningful information received from the gates across the cells. In the LSTM architecture shown in Figure 2, each cell takes the current input (x_t), for time step t , as well as the previous cell state (c_{t-1}), and the previous hidden state (h_{t-1}). It generates a new cell state (c_t) and hidden state (h_t) at the cell output. The forget gate processes the hidden state information (h_{t-1}) from the previous cell and the current information (x_t) through the sigmoid activation function. Information close to 0 is forgotten, while information close to 1 is retained in the cell state. The input gate updates the cell state at the output by processing the previous hidden state information (h_{t-1}) and the current information (x_t) through sigmoid and hyperbolic tangent activation functions. The output gate generates the hidden state for the next LSTM cell. At the output gate, the previous hidden state information (h_{t-1}) and current information (x_t) pass through the sigmoid function, and the information from the cell state passes through the hyperbolic tangent function. By multiplying the results of both functions, the hidden state information of the next cell (h_t) is obtained. Both the hidden state and cell state contain information about previous inputs and are utilized in prediction studies. The operations performed in the LSTM cell are provided in Equation 1.

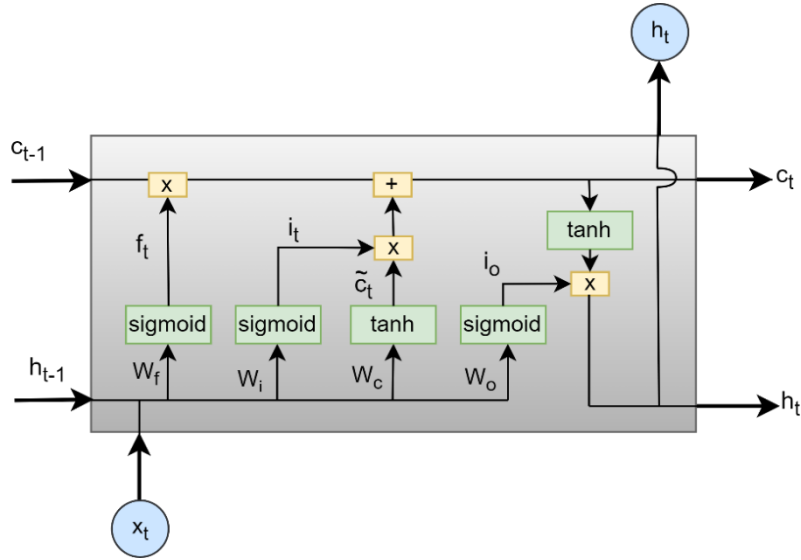


Figure 2. The Internal Structure of an LSTM Cell

$$\begin{aligned}
 f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
 \tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
 c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \\
 o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
 h_t &= o_t * \tanh(c_t)
 \end{aligned}
 \tag{1}$$

Here, x_t denotes the data at time step t ; f_t, i_t, o_t denote the forget, input and output gates, respectively, and h_{t-1} refers to the previous cell output. W_f, W_i, W_c, W_o denote the weights, while b_f, b_i, b_c, b_o denote the bias terms and σ denotes the sigmoid function.

2.1.3. GRU

GRU [18] is similar to LSTM and incorporates reset and update gates. With no cell state in its structure, GRU uses only the hidden state information of the previous cell to transfer information, thus reducing the computational cost. In the GRU cell depicted in Figure 3; the update gate determines which information to discard and which new information to retain, while the reset gate determines how much of the past information will be forgotten. When a data is input to the model, the mathematical operations taking place within the cell are provided in Equation 2.

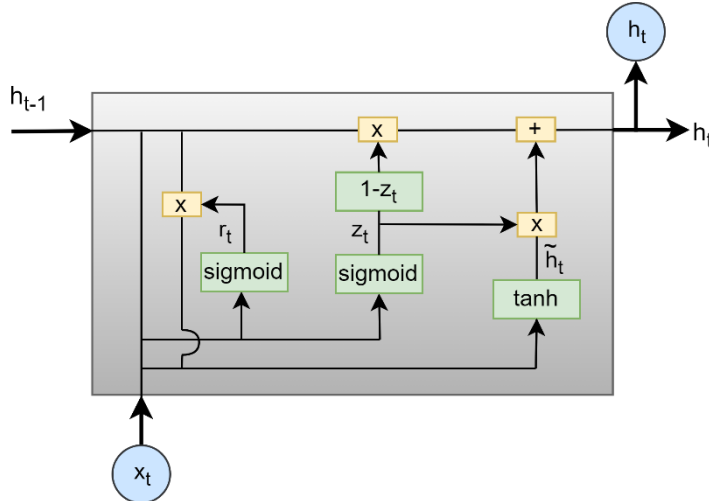


Figure 3. The Internal Structure of a GRU Cell

$$\begin{aligned}
 z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
 r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
 \tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]) \\
 h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
 \end{aligned}
 \tag{2}$$

Here, x_t represents the data at time step t ; while r_t, z_t denote the reset and update gates, respectively. Additionally, h_{t-1} refers the output of the previous cell, and W_r, W_z, W denote the weights.

2.2. Chaotic Systems and Datasets

2.2.1. Multivariate Lorenz Chaotic Time Series

The Lorenz chaotic system, widely accepted as the beginning of chaotic systems, was introduced by Edward Lorenz in 1963 and has been employed in modeling atmospheric conditions [25]. Given its widespread use in classification and prediction studies in the literature, we employed the Lorenz chaotic system, described by Equation 3, to validate our proposed methods. The parameters of the system consisting of 3 ordinary differential equations are set as $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, and the initial conditions for the state variables are chosen as $(x_o, y_o, z_o) = (0, -0.1, 9)$. 5,000 data points were derived from the solution of the chaotic system using the fourth order Runge-Kutta method with a time step value set at 0.001. The models performing predictions were trained with 4,000 data points and tested with 1,000 data points. The obtained data is depicted in Figure 4.

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z\end{aligned}\quad (3)$$

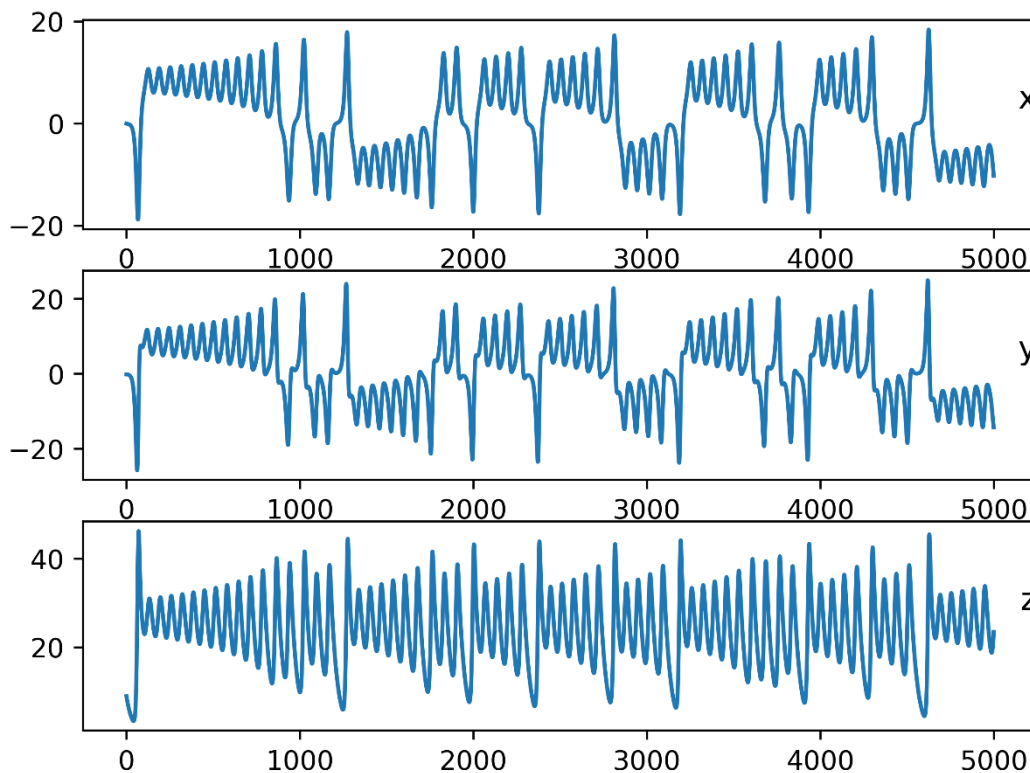


Figure 4. Multivariate Lorenz Chaotic Time Series

2.2.2. Multivariate Chen Chaotic Time Series

The Chen chaotic system [26], consisting of 3 ordinary differential equations and presented to the scientific world by Guanrong Chen and Ueta in 1999, is given in Equation 4.

$$\begin{aligned}\dot{x} &= a(y - x) \\ \dot{y} &= (c - a)x - xz + cy \\ \dot{z} &= xy - bz\end{aligned}\quad (4)$$

The system parameters are defined as $a = 35$, $b = 3$, $c = 28$, with initial values for the state variables set as $x_o = -10$, $y_o = 0$, $z_o = 37$. Employing a time step value of 0.01 and utilizing the fourth order Runge-Kutta method, 5,000 data points have been derived from solving the chaotic system. During the training phase, 4,000 data points were utilized, and during the testing phase, 1,000 data points were used. The data obtained is shown in Figure 5. In this study, the Chen system, which is particularly similar to the Lorenz system, was selected. Thus, differences in prediction results of similar systems in the conducted study will be observed.

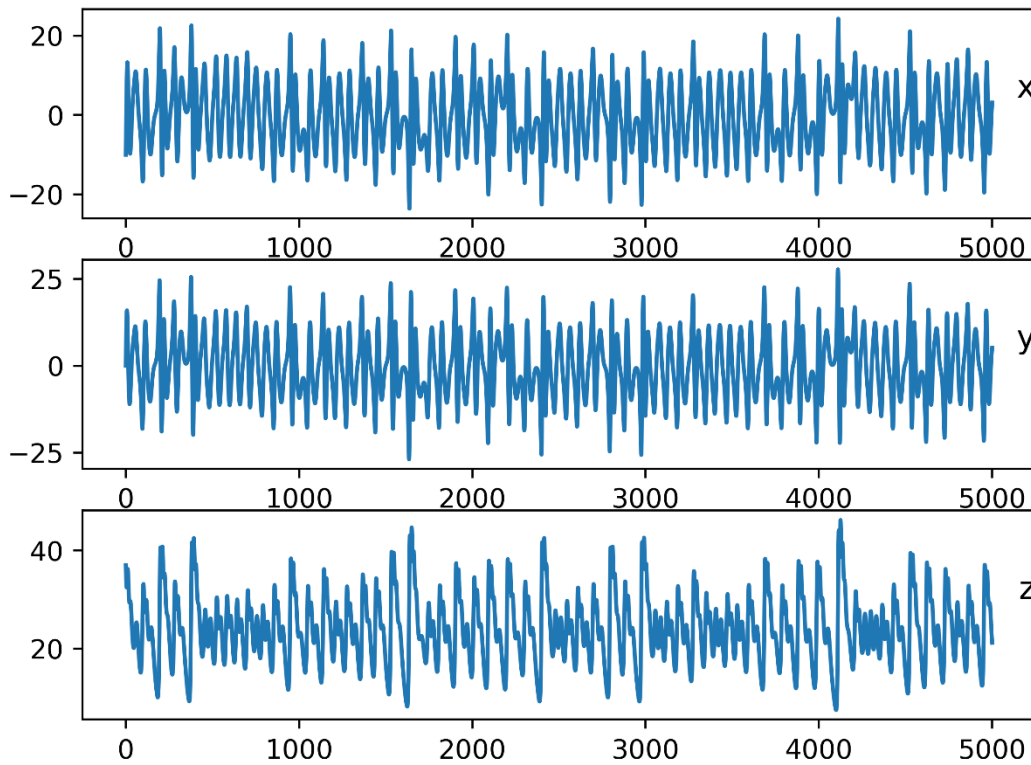


Figure 5. Multivariate Chen Chaotic Time Series

2.2.3. Multivariate Rikitake Chaotic Time Series

The Rikitake system [27], [28], employed in the fields of hydrodynamics and electromagnetic phenomena, elucidates the flow of liquid metal within a magnetic field. The complex behaviors arising from the influence of the magnetic field and the temporal variations imbue this system with chaotic characteristics. The Rikitake system has been utilized to investigate how a system, distinct from the Lorenz and Chen systems, would manifest changes under the same hyperparameter values. The system consists of three ordinary differential equations, which are provided in Equation 5.

$$\begin{aligned}\dot{x} &= -\mu x + yz \\ \dot{y} &= -\mu y + (z - a)x \\ \dot{z} &= 1 - xy\end{aligned}\quad (5)$$

The system parameters are set as $\mu = 2$ and $a = 5$, with initial values for the state variables defined as $x_0 = 3$, $y_0 = 1$, $z_0 = 6$. Employing a time step value of 0.01 and utilizing the fourth order Runge-Kutta method, a total of 20,000 data points have been obtained from solving the chaotic system. The initial 4,000 data points were discarded, and prediction studies were conducted on the remaining 16,000 data points. The resulting 20,000 data points are depicted in Figure 6.

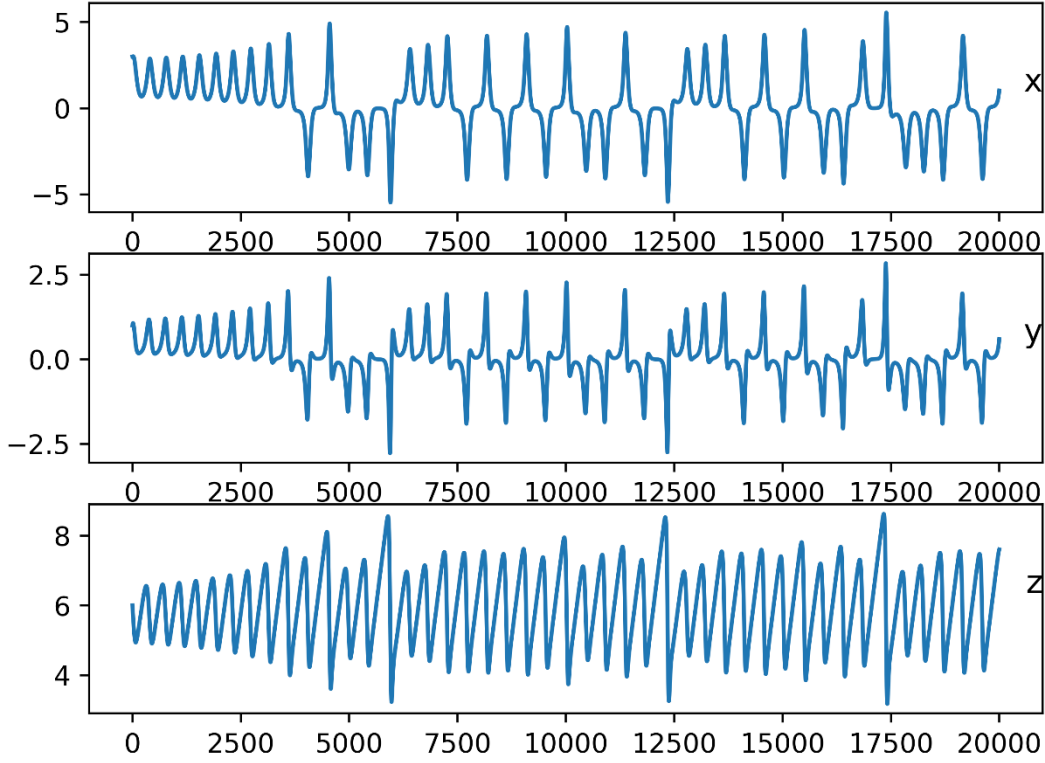


Figure 6. Multivariate Rikitake Chaotic Time Series

2.3 Evaluation Metrics

In this study, metrics such as MSE, RMSE, MAE, MAPE, and R^2 have been employed to measure the error between the predicted and actual values of the chaotic time series. The RMSE provides insights into the generalization capability of prediction models, while MSE, MAE, and MAPE represent the stability of the model. A smaller RMSE signifies enhanced predictive prowess, and lower MAE, MSE, and MAPE values denote greater model stability. R^2 reflects the correlation between the model and the data. Optimal prediction models strive for values close to 0 for RMSE, MSE, MAE, and MAPE and close to 1 for R^2 [29]. The evaluation metrics are as seen in Equation 6.

$$\begin{aligned}
 MSE &= \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\
 RMSE &= \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \\
 MAE &= \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \\
 MAPE &= \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \\
 R^2 &= 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad \bar{y} = \frac{\sum_{i=1}^n y_i}{n}
 \end{aligned} \tag{6}$$

Here, n is the total number of data points yielded from the chaotic system, y_i is the actual data values, and \hat{y}_i is the predicted values.

3. Experiment and Result Analysis

The study utilized the Python programming language, along with the Tensorflow, Keras, Scikit-Learn, Pandas, and NumPy libraries, executed within a Jupyter notebook environment. The datasets, derived from multivariate chaotic systems, were partitioned into 80% for training and 20% for testing. The training and testing processes employed the Adaptive Moment Estimation (Adam) optimizer throughout.

The data obtained from multivariate chaotic systems is initially normalized between 0 and 1, and then provided as input to the models in the form of (S, T, F), where S (samples) indicates the number of data rows given to the model, T (time steps)

represents how many steps ahead the future data will be predicted, and F (features) denotes the number of state variables included in the data. These two processes are part of the data preprocessing in the chaotic time series prediction flowchart shown in Figure 7. The model layer involves one or two layers of RNN, LSTM or GRU models. The dense layer predicts the y state variable for the Lorenz system and the x state variables for the Chen and Rikitake systems. The estimated variables are later evaluated using assessment metrics.

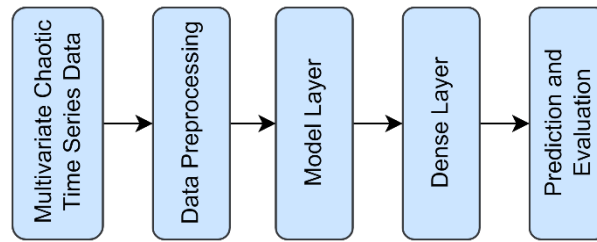


Figure 7. Flowchart of Chaotic Time Series Prediction

Data obtained from chaotic systems were trained on single-layer GRU, LSTM, and RNN models, as well as on two-layer GRU-GRU, LSTM-LSTM, and RNN-RNN models. For the time series forecasting study, it was decided to use the hyperparameters specified in the following tables as a result of the literature research. The models are trained with all hyperparameter combinations given in the tables. In the study aiming to predict the x_t state variable of the Chen chaotic system, each model was trained a total of 960 times ($2*4*3*2*4*5$) with 2 learning rate parameters, 4 units parameters, 3 loss function parameters, 2 dropout parameters, 4 batch size parameters, and 5 epoch parameters.

The hyperparameters of the models that achieved the best results during the testing phase are presented in Table 1.

Table 1. The Best Performing Hyperparameters in Chen Chaotic Time Series Prediction

	Model	Learning rate	Unit	Loss function	Dropout	Batch size	Epoch
learning_rate = [0.001, 0.01] units= [32, 50, 64, 128] loss_function=['mse','mae','mape'] dropout=[0, 0.1] batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	GRU-GRU	0.001	128	MSE	0	128	120
	LSTM-LSTM	0.001	128	MSE	0	16	32
	RNN-RNN	0.001	32	MSE	0	128	120
	GRU	0.001	128	MSE	0	32	100
	LSTM	0.001	128	MSE	0	16	100
	RNN	0.001	32	MAE	0	32	120

The evaluation metric results for the models in Table 1 are presented in Table 2, revealing that the optimal performance is obtained with the single-layer LSTM model.

Table 2. Metric Values Obtained from the Prediction of the Chen Chaotic Time Series

Model	MSE	RMSE	MAE	MAPE	R ²
GRU-GRU	0.00018915398933 3943	0.01375332648 24894	0.0097478297875 705	0.0041291130350 988	0.9999973397 2308
LSTM-LSTM	0.00035761746112 0523	0.01891077632 25237	0.0130130124454 234	0.0043935901387 45	0.9999949704 3925
RNN-RNN	0.00213821949234 0051	0.04624088550 55788	0.0298703904947 043	0.0097587028593 110	0.9999699279 0902
GRU	0.00014579000713 1551	0.01207435328 00540	0.0092421103157 197	0.0035755420327 708	0.9999979495 9761
LSTM	0.00013495373711 6325	0.01161695903 05004	0.0078688794263 481	0.0031135044182 560	0.9999981019 9978
RNN	0.00356330276033 4845	0.05969340633 88482	0.0511561370701 601	0.0242245884906 309	0.9999498854 2328

In the prediction study of the y_t state variable of the Lorenz chaotic system, each model was trained with 960 or 480 combinations of selected hyperparameters. The hyperparameters of the models that provided the best performance during the testing phase are indicated in Table 3.

Table 3. The Best Performing Hyperparameters in Lorenz Chaotic Time Series Prediction

	Model	Learning rate	Unit	Loss function	Dropout	Batch size	Epoch
learning_rate = [0.001, 0.01] units= [32, 50, 64, 128] loss_function= ['mse','mae','mape'] dropout=[0, 0.1] batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	GRU-GRU	0.001	64	MSE	0	64	120
	LSTM-LSTM	0.001	64	MSE	0	128	120
	RNN-RNN	0.001	64	MSE	0.1	128	120
	GRU	0.001	64	MSE	0	16	120
	LSTM	0.001	50	MSE	0	32	60
learning_rate = [0.001, 0.01] units= [32, 50, 64, 128] loss_function= ['mse','mae','mape'] dropout=0 batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	RNN	0.001	50	MAE	0	32	120

The evaluation metric results for the models in Table 3 are presented in Table 4, revealing that the best performance is obtained with the two-layer LSTM-LSTM model.

Table 4. Metric Values Obtained from the Prediction of the Lorenz Chaotic Time Series

Model	MSE	RMSE	MAE	MAPE	R ²
GRU-GRU	0.0015682 27371358 88	0.0396008506393 345	0.0292398252888 286	0.01200802179359 9652	0.9999680544908 985
LSTM-LSTM	0.0010024 98897151 98	0.0316622629821 682	0.0207719784188 649	0.01694382531717 0078	0.9999795786387 689
RNN-RNN	0.0049229 00971254 68	0.0701633876837 107	0.0535579514135 681	0.03678723314378 73	0.9998997182547 3
GRU	0.0013282 40409769 35	0.0364450327173 5877	0.0241061105455 2382	0.01726728006570 41	0.9999729431351 13
LSTM	0.0015806 25871862 54	0.0397570858069 6701	0.0281864592738 778	0.01907841571258 94	0.9999678019277 70
RNN	0.0059485 26427343 98	0.0771266907584 137	0.0465172340879 340	0.03898589775213 59	0.9998788257949 1

In the prediction of Lorenz and Chen's chaotic time series, good results have generally been obtained during both training and testing phases with learning rate = 0.001 and dropout = 0 values. Considering this, models for predicting the Rikitake chaotic time series were trained and tested with 960, 480, or 240 hyperparameter combinations. In two-layer models, 12,800 data points were used for training, and 3,200 data points were used for testing out of a total of 16,000 data points. For single-layer models, 8,000 data points were used for training and 2,000 data points for testing out of a total of 10,000 data points. The hyperparameters of the models that yielded the best results in predicting the x_t variable of the Rikitake system are provided in Table 5.

Table 5. The Best Performing Hyperparameters in Rikitake Chaotic Time Series Prediction

	Model	Learning rate	Unit	Loss function	Dropout	Batch size	Epoch
learning_rate = [0.001, 0.01] units= [32, 50, 64, 128] loss_function=['mse','mae','mape'] dropout=[0, 0.1] batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	GRU-GRU	0.001	64	MSE	0	128	32
learning_rate =0.001 units= [32, 50, 64, 128] loss_function=['mse','mae','mape'] dropout=0 batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	LSTM-LSTM	0.001	32	MSE	0	128	100
	RNN-RNN	0.001	32	MSE	0	128	32
learning_rate = 0.001 units= [32, 50, 64, 128] loss_function=['mse','mae','mape'] dropout=[0, 0.1] batch_size=[16, 32, 64, 128] epochs=[32, 50, 60, 100, 120]	GRU	0.001	32	MSE	0.1	128	120
	LSTM	0.001	50	MAPE	0	128	120
	RNN	0.001	64	MSE	0.1	128	100

The evaluation metric results for the models in Table 5 are presented in Table 6, revealing that the best performance is obtained with the two-layer GRU-GRU model.

Table 6. Metric Values Obtained from the Prediction of the Rikitake Chaotic Time Series

Model	MSE	RMSE	MAE	MAPE	R ²
GRU-GRU	0.000103056930158843	0.0101516959252552	0.0068777473052769	0.0696440854960179	0.999964715670350
LSTM-LSTM	0.000160392630502165	0.0126646212143185	0.0086265765293640	0.0435886229582805	0.999944952715469
RNN-RNN	0.000707852209306246	0.0266054920891579	0.0239891412013945	0.2997457660216611	0.999757647635554
GRU	0.00011047195962303	0.0105105641914709	0.0067783695547486	0.0399761636985075	0.999955300069389
LSTM	0.000117093448747761	0.0108209726340917	0.0065403506694696	0.0111550839614390	0.999952620836528
RNN	0.000150258160515648	0.01225798354198795	0.00868696034951605	0.0397396786910791	0.999939201500799

The actual values of the x_t state variable of the multivariate Chen chaotic system and the predicted values from the LSTM model are shown in Figure 8. The actual values of the y_t state variable of the multivariate Lorenz chaotic system and the predicted values from the LSTM-LSTM model are shown in Figure 9. The actual values of the x_t state variable of the multivariate Rikitake chaotic system and the predicted values from the GRU-GRU model are shown in Figure 10.

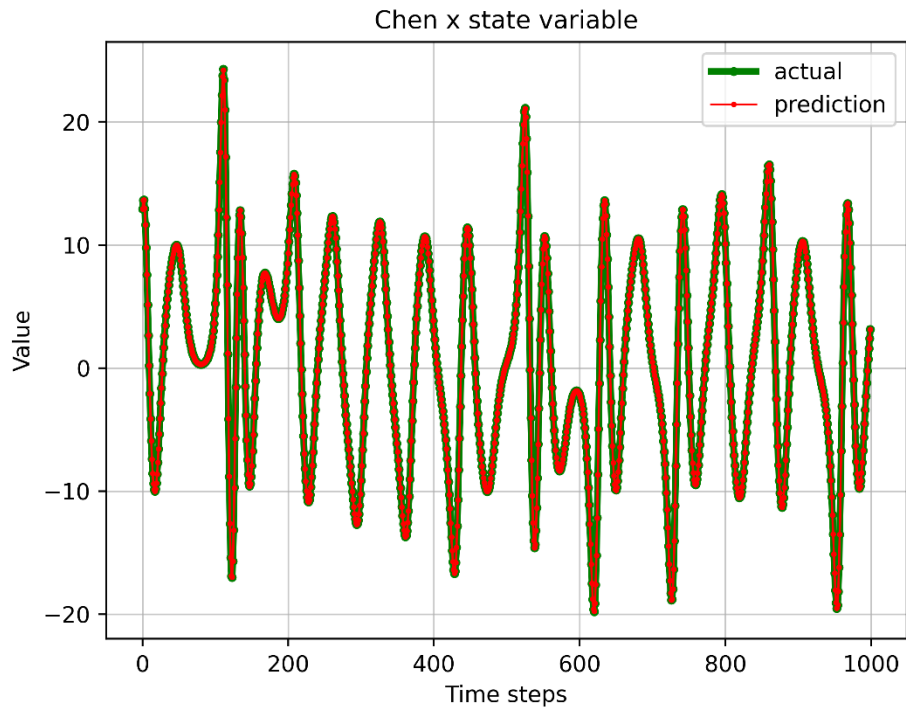


Figure 8. Prediction Result of the x_t State Variable in the Multivariate Chen Chaotic System

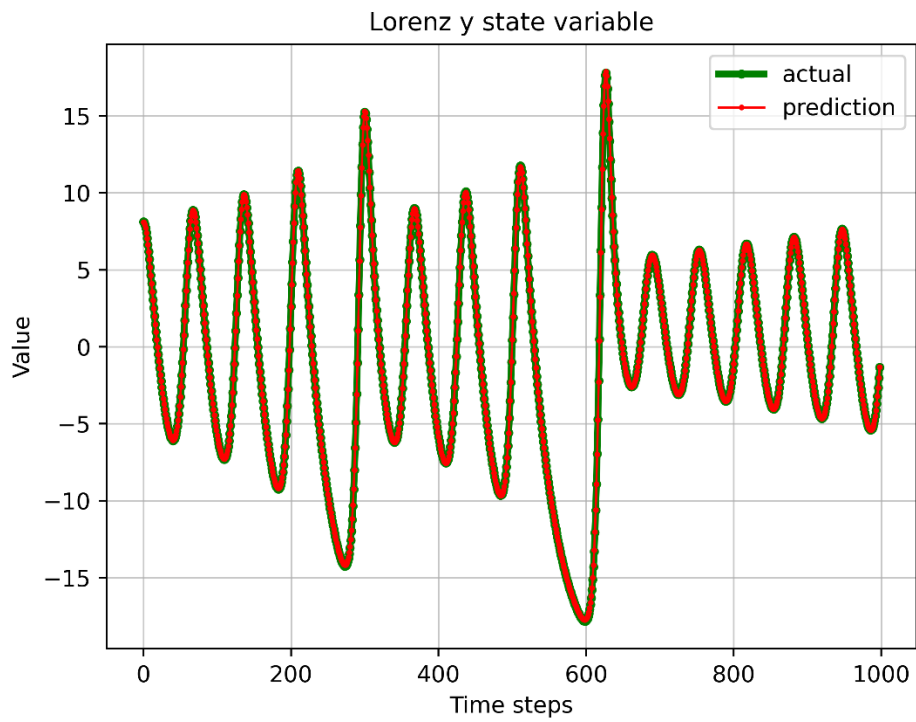


Figure 9. Prediction Result of the y_t State Variable in the Multivariate Lorenz Chaotic System

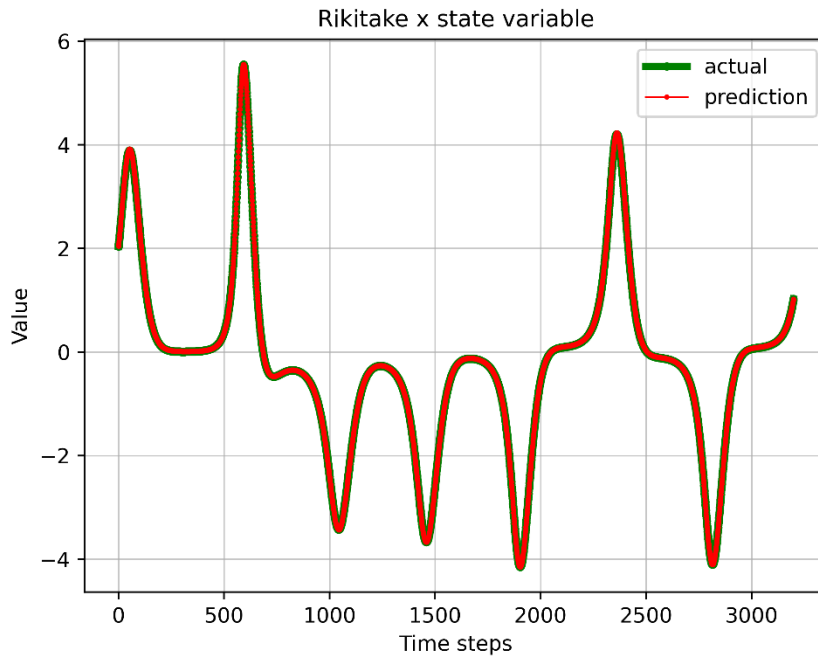


Figure 10. Prediction Result of the x_t State Variable in the Multivariate Rikitake Chaotic System

For each chaotic system, training and testing were conducted on six models using the hyperparameters specified in the above tables. The metric values for the top three combinations yielding the best results in the hyperparameter combination of the model that achieved the best performance for each chaotic system are presented in Figures 11, 12, 13, 14, and 15. For instance, the Lorenz chaotic system was trained and tested with 960 hyperparameter combinations in the GRU model. Among the results of these 960 studies, the three most successful studies are denoted as Lorenz_GRU_1, Lorenz_GRU_2, and Lorenz_GRU_3, respectively.

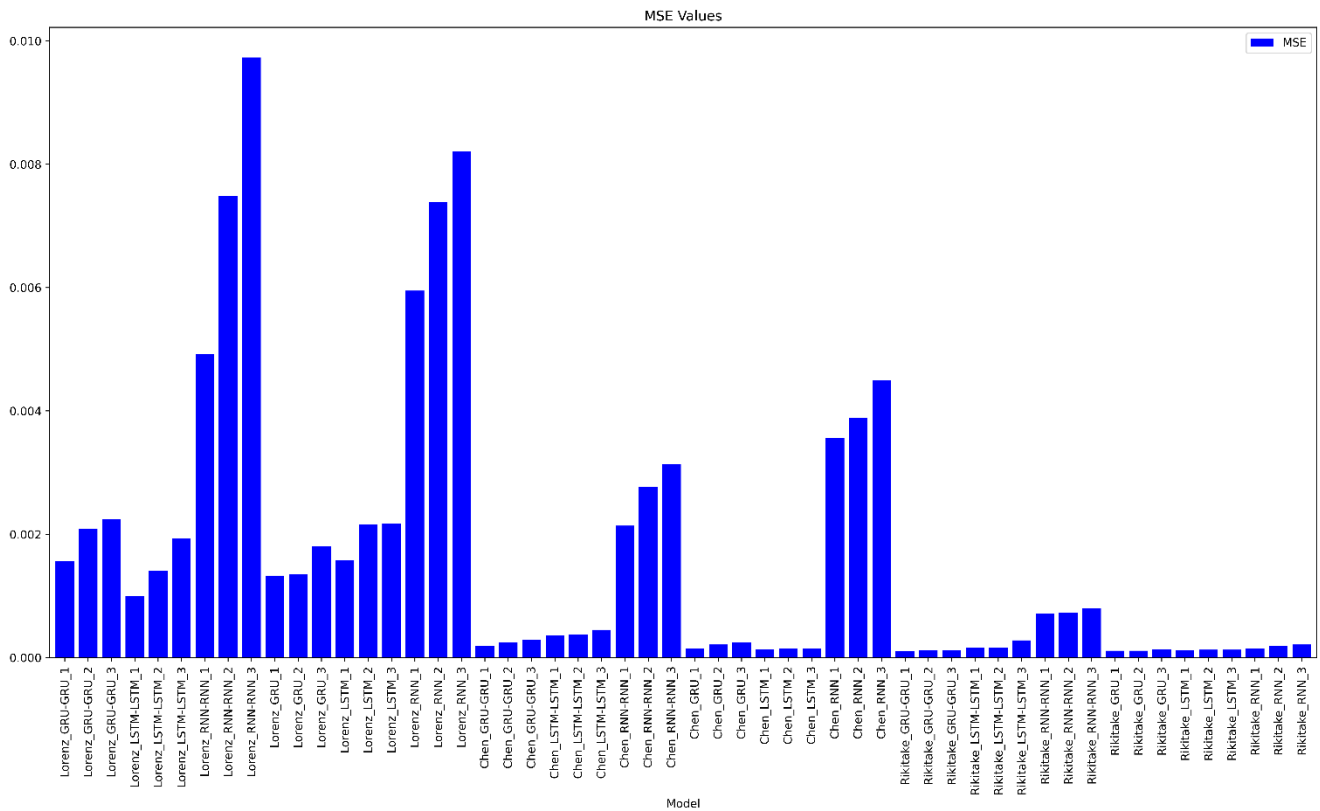


Figure 11. MSE Values of the Models That Achieved the Best Performance in Chaotic Systems

In Figure 11, it appears that models achieved lower MSE values in predicting Chen and Rikitake's chaotic time series. Additionally, RNN models obtained higher MSE values in the prediction of chaotic time series.

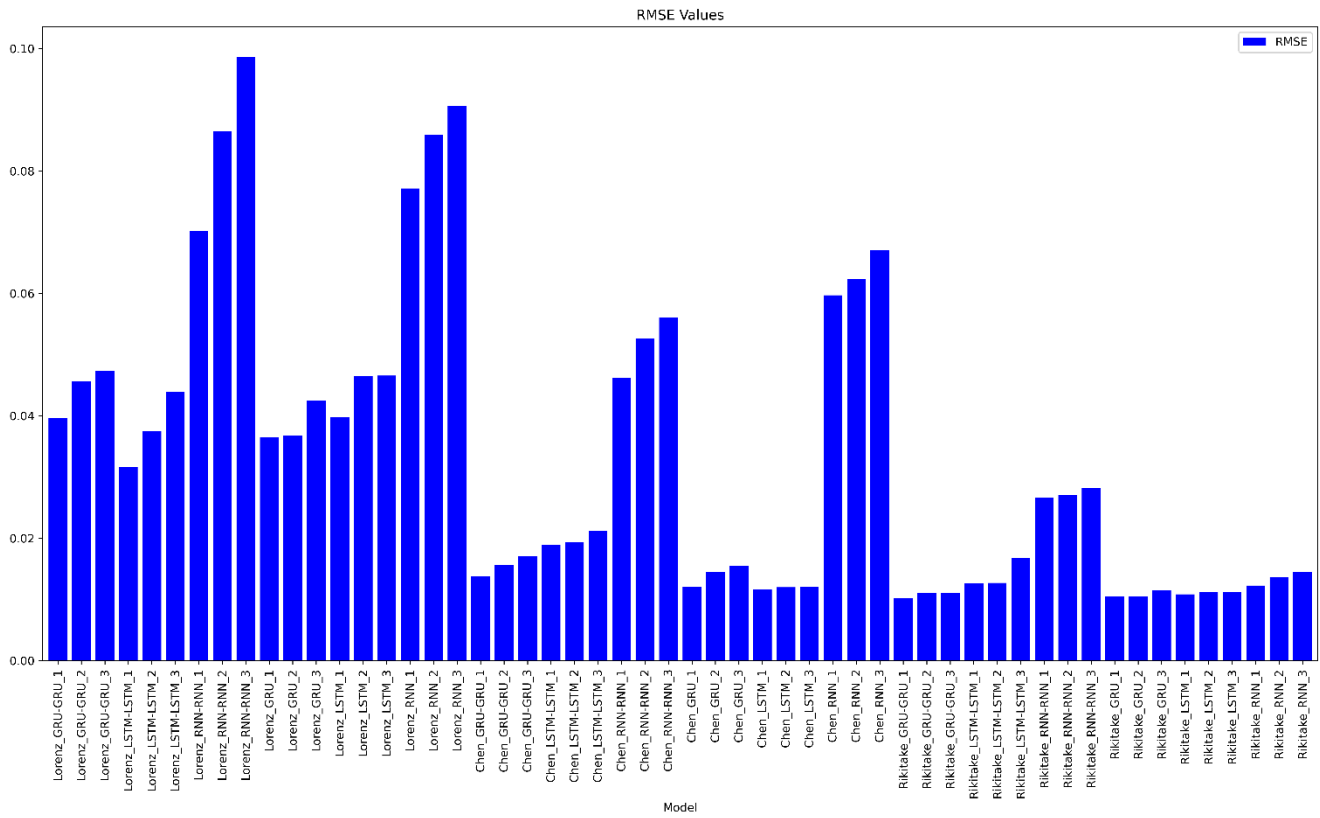


Figure 12. RMSE Values of the Models That Achieved the Best Performance in Chaotic Systems

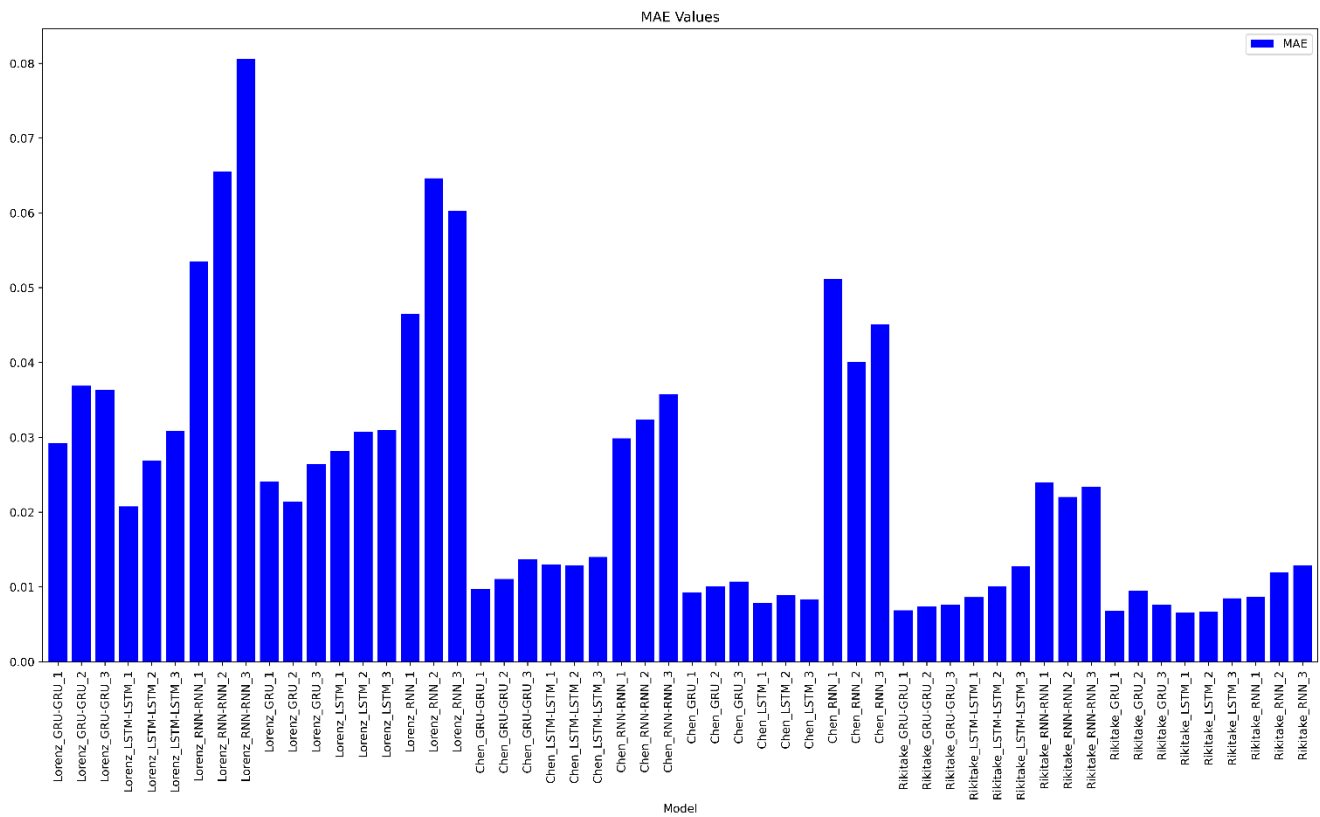


Figure 13. MAE Values of the Models That Achieved the Best Performance in Chaotic Systems

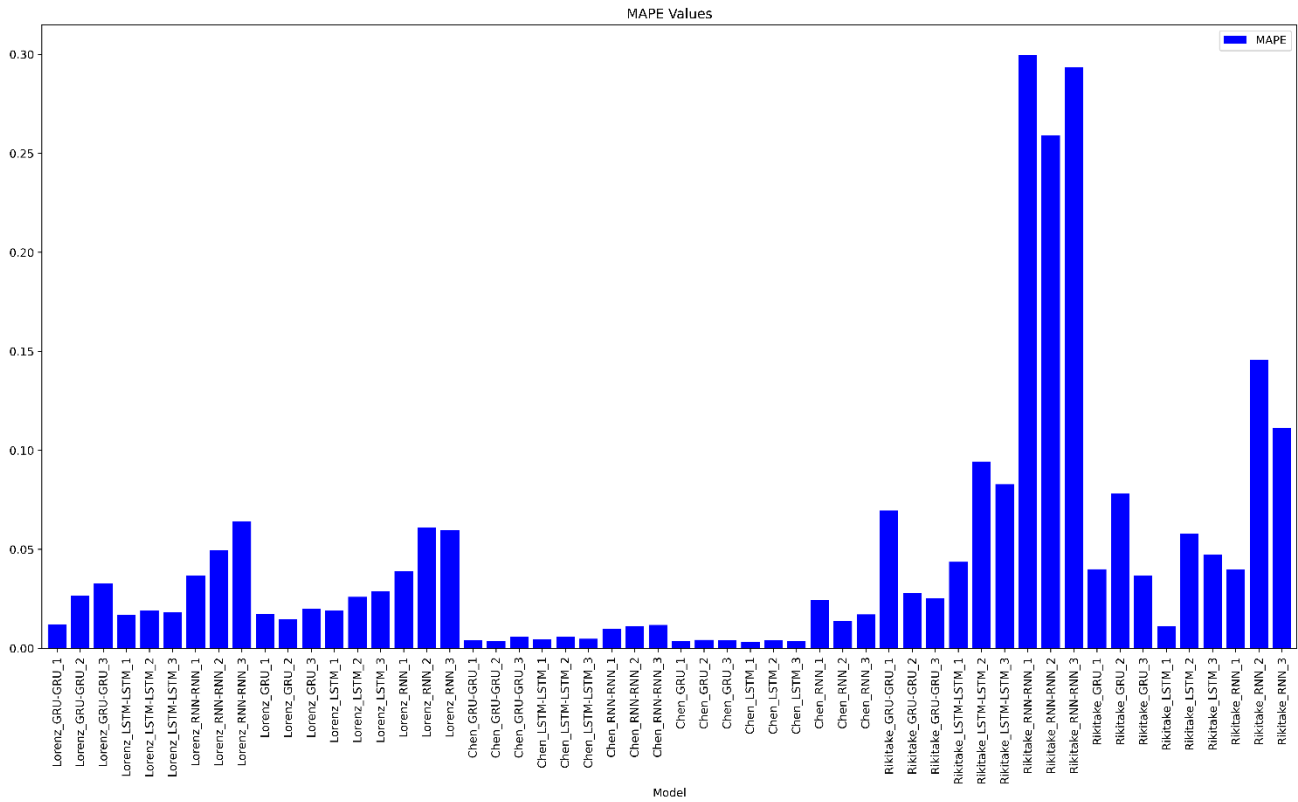


Figure 14. MAPE Values of the Models That Achieved the Best Performance in Chaotic Systems

Upon examining Figures 13 and 14, it is evident that in the prediction results of Lorenz and Chen's chaotic time series, MAPE values are lower than MAE values. This suggests that the model incurs substantial errors concerning the true values; however, when these errors are expressed relative to the entire dataset, they appear relatively smaller. The low MAE values in the prediction of the Rikitake chaotic time series indicate that the predictions are close to the real values, and small errors are made. The fact that the MAPE values are higher than the MAE values indicates that the small errors that occur have a higher effect when compared to the entire data. This indicates the presence of proportional errors in predicting the Rikitake chaotic system. This situation does not imply poor performance in the model's prediction study; rather, it indicates that the data generated from the Rikitake system has a narrower value range than the other two models.

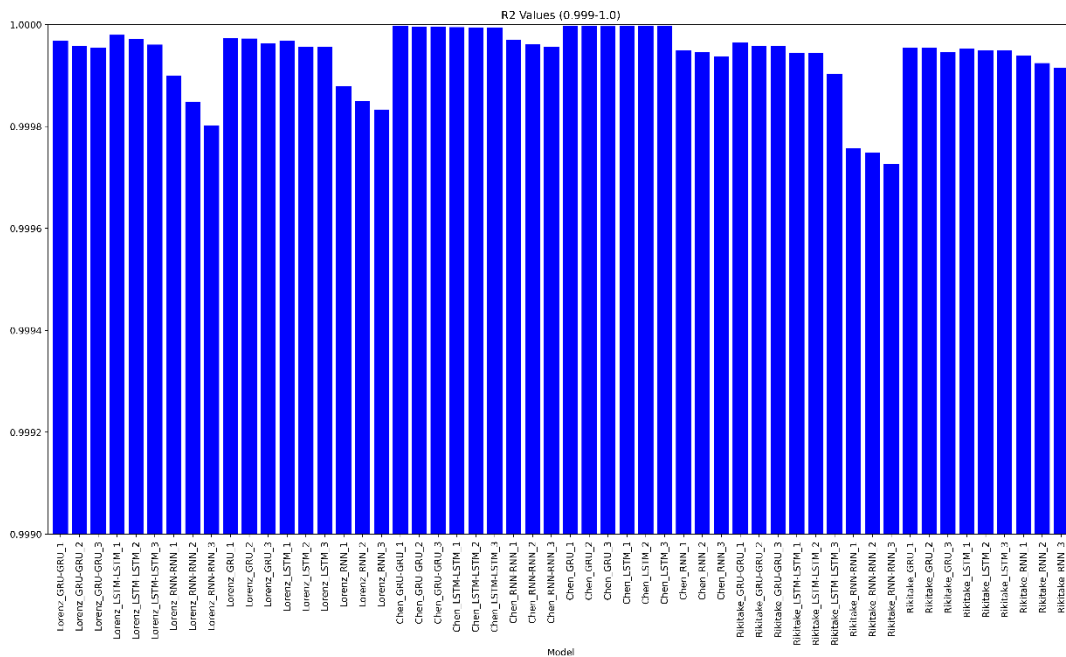


Figure 15. R² Values of the Models That Achieved the Best Performance in Chaotic Systems

Table 7 outlines the studies in the literature that utilize the Lorenz, Chen, and Rikitake chaotic systems. It specifies the number of data points generated from these systems, the split ratios of data points used in the models, employed prediction models, and the evaluation metrics used to assess the prediction results. All studies mentioned in Table 7 have attempted to predict only one state variable of the chaotic system by giving one state variable to the prediction models. The studies numbered 20, 22, and 23 created their own hybrid models to achieve good performance in prediction tasks. They demonstrated that their hybrid models yielded more accurate results than other models such as LSTM, GRU, TCN, and CNN. However, in our study, better prediction results were obtained without using any hybrid models. This was achieved by providing the state variables x , y , and z of the system to models like RNN, LSTM, and GRU for predicting a single state variable. The number of data points and the split ratios of the data points also influence the performance of the prediction models. Despite using the same data point split ratio and having a larger number of data points, study number 23 obtained less favorable results compared to our study, as indicated by their evaluation metrics. In Table 7, only study number 9 achieved better prediction results than our proposed study. The researchers utilized 100,000 data points instead of 5,000, split the data into training, testing, and validation sets at different ratios, and used Grid Search for hyperparameter selection to achieve better performance.

Table 7. Models Used in Chaotic Time Series Predictions and the Obtained Metric Values

Reference	Chaotic System	Datasets	Model	Evaluation Metric
Dudukcu et al. [9]	Lorenz $x_0 = 0.9$ $y_0 = 0.9$ $z_0 = 0$ Input: y_t Output: y_t	100,000 data point 40% train 10% validation 50% test	LSTM	RMSE: 0.0042, MAE: 0.0025, R ² : 0.9994
			GRU	RMSE: 0.0045, MAE: 0.0028, R ² : 0.9993
			LSTM-LSTM	RMSE: 0.0041, MAE:0.0025, R ² : 0.9994
			GRU-GRU	RMSE: 0.0038, MAE: 0.0023, R ² : 0.9995
			LSTM-GRU	RMSE: 0.0044, MAE: 0.0027, R ² : 0.9993
			TCN-LSTM	RMSE: 0.0024, MAE: 0.0014, R ² : 0.9998
			TCN-GRU	RMSE: 0.0029, MAE: 0.0017, R ² : 0.9997
Yanan et al. [20]	Lorenz-63 $x_0 = -0.2028$ $y_0 = 3.5418$ $z_0 = 25.0873$	5,000 data point	CEEMDAN-LSTM	RMSE: 1.327, MAE: 1.124, MAPE: 0.119
			LSTM	RMSE: 2.042, MAE: 1.527, MAPE: 0.214
Dudukcu et al. [21]	Lorenz $x_0 = 0.9$ $y_0 = 0.9$ $z_0 = 0$ Input: y_t Output: y_t	150,000 data point 40% train 10% validation 50% test	Vanilla LSTM	RMSE:0.3376
			Stacked LSTM	RMSE:0.3213
			Bidirectional LSTM	RMSE:0.3005
			CNN-LSTM	RMSE:0.3311
Fu et al. [22]	Lorenz $x_0 = 1$ $y_0 = 0$ $z_0 = 1$ Input: x_t Output: x_t	8,000 data point 70% train 5% validation 25% test	DTIGNet GRU LSTM CNN-GRU CNN-LSTM	MAE: 0.022654, RMSE:0.030894 MAPE: 0.02886535, R ² : 0.999983
				MAE: 0.049698, RMSE:0.070488 MAPE: 0.01689394, R ² : 0.999911
				MAE: 0.141781, RMSE:0.212693 MAPE: 0.18966734, R ² : 0.999822
				MAE: 0.046484, RMSE: 0.065715 MAPE: 0.22064176, R ² : 0.999922
				MAE: 0.076838, RMSE: 0.108909 MAPE: 0.23432454, R ² : 0.999787
Cheng et al. [23]	Chen $x_0 = -1$ $y_0 = -1.1$ $z_0 = 0$ Input: x_t Output: x_t	8,000 data point 80% train 20% test	TCN-CBAM TCN CNN-LSTM LSTM	MAE: 0.15410, RMSE: 0.21076, R ² : 0.99938
				MAE:0.26315, RMSE: 0.33381, R ² : 0.99843
				MAE: 0.2706, RMSE: 0.48527, R ² : 0.99671
				MAE: 0.42837, RMSE: 0.85431, R ² : 0.98984
	Lorenz $x_0 = 1$ $y_0 = 0$ $z_0 = 1$ Input: x_t Output: x_t	10,000 80% train 20% test	TCN-CBAM TCN CNN-LSTM LSTM	MAE: 0.09998, RMSE: 0.14039, R ² : 0.99969
				MAE: 0.14883, RMSE: 0.18882, R ² : 0.99943
				MAE: 0.12303, RMSE: 0.18913, R ² : 0.99943
				MAE: 0.13647, RMSE: 0.22891, R ² : 0.99917
Our study	Lorenz $x_0 = 0$ $y_0 = -0.1$	5,000 data point	LSTM	MSE: 0.000135 RMSE: 0.01162 MAE: 0.00787

	$z_0 = 9$ Input: x_t, y_t, z_t Output: y_t	80% train 20% test		MAPE: 0.00311 R^2 : 0.999998
	Chen $x_0 = -10$ $y_0 = 0$ $z_0 = 37$ Input: x_t, y_t, z_t Output: x_t	5,000 data point 80% train 20% test	LSTM-LSTM	MSE: 0.001002 RMSE: 0.03166 MAE: 0.020772 MAPE: 0.01694 R^2 : 0.99998
	Rikitake $x_0 = 3$ $y_0 = 1$ $z_0 = 6$ Input: x_t, y_t, z_t Output: x_t	16,000 data point 80% train 20% test	GRU-GRU	MSE: 0.0001031 RMSE: 0.010152 MAE: 0.006878 MAPE: 0.06964 R^2 : 0.999965

4. Conclusion and Future Work

This study assesses the effectiveness of deep learning models in predicting multivariate chaotic time series by focusing on the multivariate Lorenz, Lorenz-like Chen, and non-Lorenz-like Rikitake chaotic systems. The chaotic time series generated from chaotic systems using the fourth order Runge-Kutta method were employed in one and two-layer GRU, LSTM, and RNN models. The prediction performances of deep learning models, trained and tested with different hyperparameter combinations, were compared using evaluation metrics such as MSE, RMSE, MAE, MAPE, and R^2 . The experimental study demonstrated that using all state variables composing the chaotic systems, rather than a single state variable, in the prediction models resulted in better performance than similar studies. This approach suggests that a model predicting multivariate chaotic time series can better understand the system dynamics comprehensively, leading to more reliable predictions. In addition, compared to studies in the literature that use hybrid model design to achieve high performance, this study shows that prediction performance is improved with basic LSTM, GRU, and RNN models and appropriate hyperparameters selected for these models, without using hybrid model design.

In future studies, time series predictions can be conducted on more complex chaotic systems and real-world problems using new hyperparameters and models. Additionally, research can be carried out to predict not only the next step but also several steps ahead in time series prediction studies. This would allow for a comprehensive comparison of the most suitable models for such studies.

References

- [1] A. L. Mrgole and D. Sever, "Incorporation of Duffing Oscillator and Wigner-Ville Distribution in Traffic Flow Prediction," *Promet - Traffic & Transportation*, vol. 29, no. 1, pp. 13–22, Feb. 2017, doi: 10.7307/ptt.v29i1.2116.
- [2] J. Runge and R. Zmeureanu, "A Review of Deep Learning Techniques for Forecasting Energy Use in Buildings," *Energies*, vol. 14, no. 3, Art. no. 3, Jan. 2021, doi: 10.3390/en14030608.
- [3] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005-2019." arXiv, Nov. 29, 2019. doi: 10.48550/arXiv.1911.13288.
- [4] I. Yazici, O. F. Beyca, and D. Delen, "Deep-learning-based short-term electricity load forecasting: A real case application," *Engineering Applications of Artificial Intelligence*, vol. 109, p. 104645, Mar. 2022, doi: 10.1016/j.engappai.2021.104645.
- [5] M. Murat, I. Malinowska, M. Gos, and J. Krzyszczak, "Forecasting daily meteorological time series using ARIMA and regression models," *International Agrophysics*, vol. 32, no. 2, pp. 253–264, Apr. 2018, doi: 10.1515/intag-2017-0007.
- [6] P. Kavianpour, M. Kavianpour, E. Jahani, and A. Ramezani, "A CNN-BiLSTM model with attention mechanism for earthquake prediction," *J Supercomput*, May 2023, doi: 10.1007/s11227-023-05369-y.
- [7] T. Ouyang, H. Huang, Y. He, and Z. Tang, "Chaotic wind power time series prediction via switching data-driven modes," *Renewable Energy*, vol. 145, pp. 270–281, Jan. 2020, doi: 10.1016/j.renene.2019.06.047.
- [8] C. Cheng et al., "Time series forecasting for nonlinear and non-stationary processes: a review and comparative study." *IIE Transactions*, vol. 47, no. 10, pp. 1053–1071, Oct. 2015, doi: 10.1080/0740817X.2014.999180.
- [9] H. V. Dudukcu, M. Taskiran, Z. G. C. Taskiran, and T. Yildirim, "Temporal Convolutional Networks with RNN approach for chaotic time series prediction," *Applied Soft Computing*, vol. 133, p. 109945, Jan. 2023, doi: 10.1016/j.asoc.2022.109945.
- [10] D. S. K. Karunasinghe and S.-Y. Liong, "Chaotic time series prediction with a global model: Artificial neural network," *Journal of Hydrology*, vol. 323, no. 1, pp. 92–105, May 2006, doi: 10.1016/j.jhydrol.2005.07.048.
- [11] H. Yuxia and Z. Hongtao, "Chaos Optimization Method of SVM Parameters Selection for Chaotic Time Series Forecasting," *Physics Procedia*, vol. 25, pp. 588–594, Jan. 2012, doi: 10.1016/j.phpro.2012.03.130.

- [12] Y. Xiu and W. Zhang, "Multivariate Chaotic Time Series Prediction Based on NARX Neural Networks," in *2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017)*, vol. 86. Paris: Atlantis Press, 2017, pp. 164–167.
- [13] S. Siami-Namini and A. S. Namin, "Forecasting Economics and Financial Time Series: ARIMA vs. LSTM." arXiv, Mar. 16, 2018. doi: 10.48550/arXiv.1803.06386.
- [14] R. Khaldi, A. E. Afia, R. Chiheb, and S. Tabik, "What is the best RNN-cell structure to forecast each time series behavior?," *Expert Systems with Applications*, vol. 215, p. 119140, Apr. 2023, doi: 10.1016/j.eswa.2022.119140.
- [15] G. Alkhayat and R. Mehmood, "A review and taxonomy of wind and solar energy forecasting methods based on deep learning," *Energy and AI*, vol. 4, p. 100060, Jun. 2021, doi: 10.1016/j.egyai.2021.100060.
- [16] H. Liu, G. Yan, Z. Duan, and C. Chen, "Intelligent modeling strategies for forecasting air quality time series: A review," *Applied Soft Computing*, vol. 102, p. 106957, Apr. 2021, doi: 10.1016/j.asoc.2020.106957.
- [17] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Apr. 1990, doi: 10.1016/0364-0213(90)90002-E.
- [18] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." arXiv, Sep. 02, 2014. doi: 10.48550/arXiv.1406.1078.
- [19] R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, Jun. 2012, doi: 10.1016/j.neucom.2012.01.014.
- [20] G. Yanan, C. Xiaoqun, L. Bainian, and P. Kecheng, "Chaotic Time Series Prediction Using LSTM with CEEMDAN." *Journal of Physics: Conferences Series*, vol. 1617, no. 1, p. 012094, Aug. 2020, doi: 10.1088/1742-6596/1617/1/012094.
- [21] H. V. Dudukcu, M. Taskiran, and Z. G. C. Taskiran, "Comprehensive Comparison of LSTM Variations for the Prediction of Chaotic Time Series," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, Aug. 2021, pp. 1–5. doi: 10.1109/INISTA52262.2021.9548647.
- [22] K. Fu, H. Li, and P. Deng, "Chaotic time series prediction using DTIGNet based on improved temporal-inception and GRU," *Chaos, Solitons & Fractals*, vol. 159, p. 112183, Jun. 2022, doi: 10.1016/j.chaos.2022.112183.
- [23] W. Cheng et al., "High-efficiency chaotic time series prediction based on time convolution neural network," *Chaos, Solitons & Fractals*, vol. 152, p. 111304, Nov. 2021, doi: 10.1016/j.chaos.2021.111304.
- [24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [25] E. N. Lorenz, "Deterministic Nonperiodic Flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, Mar. 1963, doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.
- [26] G. Chen and T. Ueta, "Yet another chaotic attractor." *International Journal of Bifurcation and Chaos*, vol. 09, no. 07, pp. 1465–1466, Jul. 1999, doi: 10.1142/S0218127499001024.
- [27] K. Ito, "Chaos in the Rikitake two-disc dynamo system," *Earth and Planetary Science Letters*, vol. 51, no. 2, pp. 451–456, Dec. 1980, doi: 10.1016/0012-821X(80)90224-1.
- [28] T. Rikitake, "Oscillations of a system of disk dynamos," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 54, no. 1, pp. 89–105, Jan. 1958, doi: 10.1017/S0305004100033223.
- [29] L. Wang and L. Dai, "Chaotic Time Series Prediction of Multi-Dimensional Nonlinear System Based on Bidirectional LSTM Model," *Advanced Theory and Simulations*, vol. 6, no. 8, p. 2300148, 2023, doi: 10.1002/adts.202300148.

Author(s) Contributions

Gülyeter Öztürk: Data generation, performing analysis, writing, review, and editing.
Osman Eldoğan: Writing, review, and editing.

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable

Plagiarism Statement

This article has been scanned by iThenticate™.

Optimization Planning Techniques with Meta-Heuristic Algorithms in IoT: Performance and QoS Evaluation

Murat Koca¹ , İsa Avcı² 

¹Department of Computer Engineering, Faculty of Engineering, Van Yüzüncü Yıl University, Van, Türkiye

²Department of Computer Engineering, Faculty of Engineering, Karabük University, Karabük, Türkiye

Corresponding author:

Murat Koca, Department of Computer Engineering,
Faculty of Engineering, Van Yüzüncü Yıl
University, Van, Türkiye
muratkoca@yyu.edu.tr



Article History:

Received: 13.03.2024

Accepted: 28.06.2024

Published Online: 23.08.2024

ABSTRACT

Big data analysis used by Internet of Things (IoT) objects is one of the most difficult issues to deal with today due to the data increase rate. Container technology is one of the many technologies available to address this problem. Because of its adaptability, portability, and scalability, it is particularly useful in IoT micro-services. The most promising lightweight virtualization method for providing cloud services has emerged owing to the variety of workloads and cloud resources. The scheduler component is critical in cloud container services for optimizing performance and lowering costs. Even though containers have gained enormous traction in cloud computing, very few thorough publications address container scheduling strategies. This work organizes its most innovative contribution around optimization scheduling techniques, which are based on three meta-heuristic algorithms. These algorithms include the particle swarm algorithm, the genetic algorithm, and the ant colony algorithm. We examine the main advantages, drawbacks, and significant difficulties of the existing approaches based on performance indicators. In addition, we made a fair comparison of the employed algorithms by evaluating their performance through Quality of Service (QoS) while each algorithm proposed a contribution. Finally, it reveals a plethora of potential future research areas for maximizing the use of emergent container technology.

Keywords: IoT micro-services, Container method, Optimization algorithms, Scheduling methods, Meta-heuristic algorithms

1. Introduction

We envision the Internet of Things (IoT), the Internet of Everything or the Industrial Internet, as a network of globally interconnected instruments and devices [1]. We expect an increase in IoT devices in the coming years, which will impact supply chain partners' data access and supply chain operation. Gartner 2014 forecasted that the Internet of Things (IoT) will expand to 26 billion units by 2020, up from 0.9 billion in 2009. Similarly, Gartner 2021 projects that FinFET will generate semiconductor device revenue of \$138.6 billion in 2025, up from \$87.6 billion in 2020, from production lines and warehouses to retail delivery and store shelves [2]. The demand for IoT-connected devices, machine learning (ML) applications, audio or video streaming services, and cloud storage has increased. Therefore, as microservices gain popularity, we anticipate further expansion of cloud services [3]. Cloud computing is based on virtualization technology, which enables the sharing of resources (CPUs, memory, and networks) to execute distinct programs [4].

The Docker container is a standard software entity that encapsulates code and all its dependencies so that an application can operate swiftly and reliably in any computing environment. Containers allow developers to deploy applications in isolated environments, making them ideal for microservice architecture and modern applications[5]. There are several characteristics enjoyed by containers that enable them to displace traditional Virtual Machines (VMs), and among these characteristics are the common host operating system, fast launch, and the possibility of transfer, expansion, and rapid deployment [6], [7] Containers give great flexibility to programs to create an independent runtime on the platform, and by attaching all necessary dependencies, such as instructions, the software runs and manages the system [8].

Due to the variety of duties and available cloud resources, container scheduling has emerged as a crucial aspect of the cost-effective operation of modern cloud applications [9]. Figure 1 depicts a variety of cutting-edge scheduling algorithms devised by scientists that can yield various results.

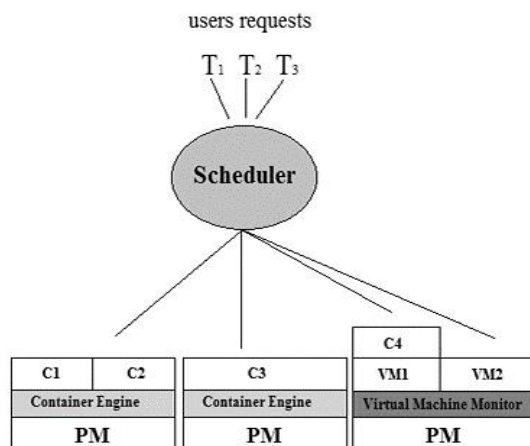


Figure 1. Sample of a Figure Caption

Other considerations include scalability, responsiveness, energy usage, cost and load balancing, resource availability, and resource efficiency, among others. Only a few studies have examined resource management in systems based on containers, although many have examined the scheduling of virtual machines [10], [11]. For large-scale container scheduling problems, no polynomial complexity algorithm exists because the problem is NP-hard. Consequently, most reported algorithms use heuristic techniques to achieve local solutions to the task [12]. Low-complexity, time-saving heuristic algorithms are typically used to generate a workable schedule. Machine learning is a hot topic regarding container scheduling that has had great success in many applications across a wide range of disciplines [13].

Machine learning algorithms rely on big data for their success, and our study's focus on meta-heuristics, a prominent class of population-based optimization algorithms, draws inspiration from the natural development of intellectual activities and actions in our environment. Two essential characteristics of these algorithms are selection and adaptation to the environment [14]. Meta-heuristics solve optimization problems in many fields.

According to Attaoui et al., the ability of mobile network function virtualization (NFV) to decouple network functions from hardware and host services on commodity hardware makes it crucial for mobile network operators. This enhances service deployment and management, improves flexibility, and reduces costs. However, the optimal placement of Virtualized Network Functions (VNFs) poses significant technical challenges, influencing network performance, reliability, and operating costs. This study explores optimization techniques, such as heuristic, meta-heuristic, and machine learning algorithms, to address VNF placement problems, focusing on both VNFs and Container Network Functions (CNFs) in edge/fog computing environments [15].

Shubha Brata Nath et al. talk about how important fog computing is for meeting the latency needs of Internet of Things (IoT) devices. They describe some problems and suggest a framework, PTC, that aims to improve response times by assigning micro-services to fog devices more efficiently. The use of Bayesian Optimization and containerization for service isolation and migration is highlighted, with experimental results demonstrating improved performance over baseline methods, providing a promising approach to enhancing fog computing architectures [16].

Hamza Mohammed Ridha Al-Khafaji proposed widespread device connectivity via the Internet of Things (IoT) and the associated challenges related to energy and cost constraints. It introduces an improved seagull optimization algorithm (ISOA) to enhance the quality of service (QoS) in IoT networks by effectively managing traffic and packet transmission. He has demonstrated that the proposed method significantly enhances QoS by outperforming previous approaches in accuracy and efficiency [17].

Satyanarayana P. et al. describe the Mobile Ad Hoc Network (MANET), a highly mobile and decentralized wireless network, and its integration with the Internet of Things (IoT) to form a novel MANET-IoT system aimed at reducing network implementation costs and enhancing user mobility. It emphasizes the need for new routing protocols and improved security measures, proposing a security protocol that utilizes an enhanced chaotic map and three advanced optimization algorithms. Performance evaluations demonstrate the superior efficiency of the proposed approach in various metrics, particularly the ABRR-CHIO algorithm, which outperforms other techniques significantly in convergence evaluations [18].

Docker is an open-source container platform for developing, shipping, and running applications. A container is a packaging method that combines our application's necessary configurations and dependencies. We run applications in isolated boxes called containers, and by isolating these boxes, we can run different applications on multiple independent containers. Docker creates containers to run and store applications and uses virtualization. Although Docker and virtual machines use isolated virtual environments for software development, they have different structures. The most important feature distinguishing Docker containers from virtual machines is that Docker containers are lighter, faster, and more resource-efficient. A virtual machine, Docker, isolates applications using container structures on a single operating system rather than creating a separate

virtual operating system for each application. This not only reduces the size of the application but also brings a significant performance increase. A piece of hardware can have multiple containers. Containers, unlike virtual machines, are virtualized at the application level. As a result, the host computer shares the kernel and virtualizes the operating system. This reduces resource usage and provides rapid and easily configurable virtual environments.

In this work includes expanded reviews and comparisons for three classical, modified, and hybrid meta-heuristic algorithms. The study included a narrative of the natural spirit and the behavior of the algorithms, additionally the mathematical model and the adaptation of container scheduling. Conversely, the comparisons concentrated on Quality of Service (QoS) metrics that optimization processes enhance. Despite its importance, this type of research remains relatively isolated. Unlike our research, which focuses on meta-heuristic algorithms, there is only a single review in the literature that provides a brief overview of all scheduling techniques. We organize the remaining sections of this essay as follows: Section 2 discusses container scheduling using optimization methods and common performance measures. Section 3 reviews and compares optimization scheduling meta-heuristics, presenting the author's perspective. Section 4 discusses the challenges and potential avenues of research and explains possible solutions. The final section presents the results and outlines the future work.

2. Management Containers by Optimization Algorithms and Performance Measurements

This work compiles research findings published in international magazines, conferences, and organizations such as IEEE, ACM, Elsevier, and Springer between January 2017 and January 2024, including the International Conference on Machine Learning. To gather papers, we searched for several container scheduling-related keywords. The section revolves around two locations: the first, a scheduling container for optimization algorithms, and the second, performance metrics.

2.1. Motivation

Since IoT microservices typically grow to meet user needs and be highly available, redundancy is frequently required. When we need additional processing power, we scale the service. When a microservice resides within a container, it scales by replicating it. Most services have substantial resource requirements and must adhere to strict performance criteria [19]. Lack of resource management might lead to rising expenses, subpar service, and energy waste [20]. Due to high service standards, finding an ideal location is necessary to ensure each service has enough resources without wasting any. To maximize consumption, cloud customers want to plan resources [20]. This project will place each microservice in its own container before scheduling it on a virtual machine. To reduce the number of Virtual Machines (VMs) and their overall cost, a container placement problem comparable to the Virtual Machine Placement (VMP) must be addressed.

2.2. Scheduling Container Problem Model Description

We formulate the Container Placement (CP) issue as follows: Given a set of containers, each with a different set of resource needs, such as CPU-intensive or memory-intensive containers, try to place all containers on VMs while using the fewest number of VMs as possible. Assume that within the specified period, between the hours of t_1 and t_2 , a set of containers arrives at the data center. When using online container allocation, the overall goal is to assign container slots to existing VMs or newly created VMs, and then assign those VMs to physical machines (PMs) to keep the total energy consumption of all PMs at its lowest possible level. The resource entities share the following characteristics (containers, virtual machines, and physical machines). CPU and memory are the two types of resources available to each entity. This study considers a data center with heterogeneous VMs of different types and homogeneous PMs, all of which share the same CPU and memory capacity. Each type of virtual machine comes pre-configured with a tuple of resources (e.g., a small VM [825 MHz, 800 MB]). Each VM has overheads in addition to its resource capacity. Virtual machine overheads represent the resources that a hypervisor uses up. A tuple of resources represents each type of VM's overhead.

Each virtual machine can run a specific operating system. A cloud provider will also specify the supported operating system categories. In the case of containers, we contemplate a one-to-one correspondence between applications and containers, as opposed to the approach of Piraghaj [21], which employs three categories of containers for all applications. The domain of required resources for containers is defined as a number between one and the capacity of PMs. Consequently, the container is defined in a much more realistic and general manner [22]. This task necessitates the use of containers that contain virtual machines running the same operating system. There are four decisions to make. Our model includes the following procedures for the online container allocation problem: Figure 2 illustrates the selection and construction of virtual machines, represented by blue lines, and virtual machines, represented by red lines.

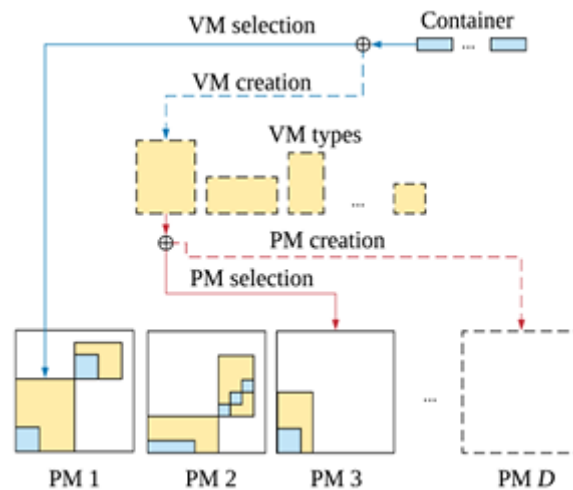


Figure 2. Container Placement Procedure

The VM selection process selects existing VMs from which to allocate containers. Another option involves creating virtual machines using cloud provider-defined virtual machine types and allocating containers to the appropriate virtual machines. The operating system requirements of the first container determine the operating system type of the virtual machine. The PM selection process chooses pre-existing PMs to assign the newly created VMs. In that it produces a kind of PM and allows the VM to use it, PM creation functions similarly to VM creation.

2.3. Scheduling Containers by Optimization Algorithms

Meta-heuristic algorithms are high-level, general optimization strategies used to solve complex problems that do not have a known solution method. We call them "meta" because they offer a higher level of abstraction than specific algorithms. Such algorithms have two important characteristics: selection of the most suitable candidates and adaptation to the environment. Meta-heuristics frequently solve optimization problems in a variety of disciplines [13].

Meta-heuristic algorithms have recently become a go-to method for tackling various industries' most challenging optimization problems. Genetic algorithms (GA) and swarm intelligence techniques like ant colony optimization, particle swarm optimization (PSO), and whale optimization are examples of meta-heuristic algorithms. Here, the meta-heuristic utilized to derive the solution has been used to categorize scheduling strategies.

2.4. Performance Metrics

We can use several performance metrics to evaluate the effectiveness of optimization algorithms:

- **Convergence:** It measures the rate at which the algorithm approaches the optimal solution.
- **Solution Quality:** It measures the algorithm's proximity to the optimal solution. The value of the objective function typically determines this.
- **Computational Time:** It measures the time the algorithm takes to find the solution.
- **Scalability:** It measures the algorithm's ability to handle larger problems.
- **Robustness:** It measures the algorithm's ability to generate effective solutions regardless of alterations to the problem or exposure to noise.
- **Repeatability:** It gauges how consistently the algorithm produces the same results for a particular problem instance each time it runs.
- **Accuracy:** It measures how closely the algorithm's solution matches the optimal solution. We recommend readers review the domain-specific optimization objectives from the most recent survey[23]. Below is a list of the most frequently used goals in the cost function specification of the container scheduling problem.

2.4.1. Energy

Regarding container deployment, the computing and storage devices hosting the containers, including servers, storage devices, switches, and other infrastructure components, consume energy. Energy consumption is an important consideration in container deployment because it can have a significant impact on the system's operating costs as well as the environment. Factors contributing to energy consumption in container deployment include the number of containers running, the utilization of computing and storage resources, the efficiency of the hardware components, and the power management policies in place. To minimize energy consumption, organizations can employ techniques such as container orchestration, resource utilization

optimization, hardware selection, and power management policies. As a result, energy efficiency serves as a critical metric for reducing carbon emissions and improving environmental sustainability [23].

2.4.2. Cost

Compute, storage, and communication costs all contribute to the overall cost of an application's execution. Compute time refers to the time it takes to run a program on the cluster's available cores. The cost increases when an application runs on a computer for a long time. Communication costs are also about how much telecommunications service providers pay to run the application. The need for more communication between nodes and clusters increases communication costs.

2.4.3. Availability

We refer to the duration of a user's access to an application as its lifetime. Users of cloud computing should have access to applications and services whenever they need them. Accessing applications and services whenever needed is an important consideration for cloud computing users.

2.4.4. Use of resources

This statistic measures the efficiency with which a worker node uses its network bandwidth, memory, and CPUs on a per-worker-node basis. According to this metric, workers are most efficient and cost-effective when used to their full capacity.

2.4.5. Load Balancing

Distributed computing ensures that no computer overburdens other idle computers. This goal affects response time, cost, and throughput. This metric is critical because of the dynamic nature of the workload in container-based applications.

2.4.6. Scalability

Additionally, container scalability is an essential measure of how well a system can handle increased demand, whether it comes from one application or a variety of different ones. These metrics measure the system's ability to adapt to changing workloads by increasing the available resources or deploying more containers on the cluster.

2.4.7. Scalability

It takes time for the application to run from start to finish. Makespan is a good scheduler's primary goal. If you're operating a real-time application that demands minimal latency or severe deadlines, this statistic is essential.

2.4.8. Throughput

The productivity of an application is determined by dividing the number of tasks by the time allotted to them. At the same time, the transfer rate provides an overview of system performance (processor, memory, and network).

2.4.9. Security

The orchestra can protect both data and services against hostile attacks or software defects by leveraging encryption and access control techniques. Online transaction processing, centralized financial services, and productivity tools enhance the performance of this metric. Because of the serious security vulnerabilities they provide, containers require special care.

2.5. The Objective Function for Performance in Algorithms

In the context of optimization or machine learning, the objective function $f(x)$, as shown in equation 1, is a mathematical expression that you aim to maximize or minimize. The form of this function depends on the specific problem you are addressing. Here is a general example:

$$f(x) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (1)$$

Where:

- y_i is the observed value,
- \hat{y}_i is the predicted value,
- n is the number of observations.

This is a common objective function used in regression problems known as the mean squared error (MSE).

2.6. Statistical Analysis for Algorithms

To evaluate the performance of different algorithms, we typically perform a statistical analysis. The key metrics often include the mean, standard deviation (std), execution time, and significance values (p-values). Here's a step-by-step guide:

1. **Mean:** Equation 2 displays the average performance metric (such as accuracy, error rate) over several runs or datasets.

$$Mean = \frac{1}{n} \sum_{i=1}^n x. \quad (2)$$

2. **Standard Deviation (std):** Equation 3 calculates the variability, or distribution, of the performance measurement.

$$Std = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - Mean)^2} \quad (3)$$

3. **Execution Time:** The time taken by the algorithm to complete its task. This can be measured using time functions in programming languages like Python's time module.
4. **Significant Value (p-value):** Determines if the performance difference between algorithms is statistically significant. Typically, you perform a hypothesis test (e.g., t-test) to compute the p-value.
- **Mean and Std:** Provide insights into the central tendency and variability of the algorithms' performance.
 - **Execution Time:** Helps assess the efficiency of the algorithms.
 - **P-value:** If the p-value is below a significance threshold (e.g., 0.05), you can conclude that there is a statistically significant difference between the performance of the two algorithms.

3. Meta-heuristic Optimization Techniques for Scheduling

Here, we demonstrate the optimization algorithms used to enhance the container scheduling problem. We also discuss meta-heuristic algorithms in detail.

3.1. Ant Colony Optimization Algorithm

3.1.1. Natural Inspired

Ant Colony Optimization (ACO), "Optimization, Learning, and Natural Algorithms," was proposed by Marco Dorigo in 1992 and is the earliest and most widely used technique [24]. The ability of actual ants to locate sustenance and determine optimal routes inspired the development of the ACO. We use this population-based general search technique to address complex combinatorial optimization problems. The Ant Colony Optimization method draws inspiration from ants' ability to communicate with each other through a specific chemical known as pheromone. Self-organizing agents or acts can communicate with each other via "stigmergy," a term that refers to "indirect communication" between them.

Real ants first travel throughout their colony in search of food. On their journey from colony to food, they leave a unique chemical 'pheromone.' When they return, they also leave behind a small amount of pheromone. Pheromone concentrations on shorter routes are higher because the ants return earlier. Because this path is the shortest, it attracts more traffic after a certain amount of time. There is a special rate at which the phenols dissipate. Consequently, ants eventually erase long trails they don't use. Pheromone trails let ants find the most efficient route between colonies and food. Therefore, the ACO algorithm mimics the behavior of real ants [25], [26].

3.1.2. ACO Algorithm

To make the ant colony algorithm suitable for continuous problems, such as the traveling salesman problem, we must first program it to generate solutions using integers. Next, we initiate the pheromone update step, which entails the deposit and evaporation of pheromones, a crucial aspect of ACO[25]. At first, ants haphazardly begin their search for food. An ant uses a probabilistic equation to choose the next node to visit. Equation 4 gives the likelihood of ant k traveling to node j while ant k is on the node.

Algorithm 1. Initializing the Pheromone

1	Procedure Initializing the pheromone matrix τ for all edges in G
2	For each iteration t from 1 to T:
3	For each k ant from 1 to m:
4	Choose the next node n in ant path k based on pheromone values and inference (e.g., using probability
5	rule).
6	Update ant path k with node n
7	Update the pheromone values at the edges based on the quality of the tracks found by the ants
8	Determine the best path found by the ants as a result of the ACO algorithm
9	end procedure

$$P_j = \begin{cases} \frac{\tau_j^\alpha \mu_j^\beta}{\sum_{j \in N_i} \tau_j^\alpha \mu_j^\beta} & \text{for } j \in N_i \\ 0 & \text{for } j \in ! N_i \end{cases} \quad (4)$$

In addition, pheromone update comprises pheromone deposit and evaporation. As an ant moves from one node to the next, its pheromone values fluctuate. The process of pheromone evaporation involves a constant factor reducing the first pheromone value on each arc.

Each ant adds some pheromone to each node it traverses as part of the pheromone deposit. Equation 5 for pheromone evaporation is as follows:

$$\tau = (1 - \rho)\tau \quad (5)$$

Where ρ is the rate of evaporation. Each ant deposits some pheromone on each node, which is known as a pheromone deposit and is given by Equation 6 below:

$$\tau = \tau + \Delta \tau \quad (6)$$

Where m is the number of ants, $\Delta \tau$ is the amount of pheromone drop on the k node. It is calculated as shown in Equation 7:

$$\Delta \tau = Q/z \quad (7)$$

The present length of the tour by k ant. It is worth mentioning that at the end, create one diminution matrix (1, the number of cities) with different values according to equation 1 divided by the summation of probabilities. The production of this equation is processed by a roulette wheel to select a numeric value between one and the city number. This operation is repeated until the estimated set, which represents by its turn the probable solution.

3.1.3. ACO for Container Adapting

The basic ant colony algorithm organized the containers perfectly, depending on the Docker algorithm; the results revealed a significant 15% improvement in performance compared to the greedy herd. However, the algorithm only considered a few optimization goals, such as maximizing the use of available resources and ensuring an even distribution of work Burvall, Multi-Objective Container Placement The presentation of Ant Colony Optimization (MOCP-ACO) transformed the traditional Ant Colony implementation by incorporating new criteria like network utilization and cost into the decision-making process [25]. Compared to Docker Swarm's spread scheduling method, MOCP-ACO outperformed the latter in all tested cases, regardless of the workload or the number of containers (16–1024) used.

3.2. Genetic Algorithm

3.2.1. Natural Inspired

Natural selection and evolution processes inspire the Genetic Algorithm (GA), a type of optimization algorithm. It is a population-based search algorithm that employs the concept of survival of the fittest to identify the optimal solution [27]. We create new populations by repeatedly applying genetic operators to members of an existing population. The essential components of GA are chromosome representation, selection, crossover, mutation, and fitness function computation. The following is GA's dispute resolution procedure: With a random sample, we start a population of chromosomes (Y). We determine the fitness of Y chromosomes by computing their fitness values. Based on their fitness, we select $C1$ and $C2$ from population Y and designate them as $C1$ and $C2$, respectively. We subject $C1$ and $C2$ to a single-point crossover operator with crossover probability (Cp), resulting in O production. We then subject the produced progeny (O) to a uniform mutation operator with a mutation probability of Mp to generate O' . It is determined where the new progeny O' will be placed in the new population. To finalize the new population, it will be necessary to repeat the selection, crossover, and mutation processes in the current population.

3.2.2. GA Algorithm

GAs are algorithms that mimic the process of natural selection. Genetic algorithms enhance problem solutions by gradually evolving a population of potential solutions as the problem's complexity increases. Each prospective solution consists of a set of chromosomes that can be modified and altered in many ways; traditionally, solutions are represented in binary as strings of 0s and 1s. We could summarize the GA by following these steps [28]:

First: Create n by m array of population,

Check the fittest chromosome: Evaluate each chromosome to calculate fitness.

Generate a new population:

Selection: Select two chromosomes from the population by following the selection procedure.

Crossover: Using the two chromosomes you've chosen, perform a crossover.

Mutation: On the chromosomes obtained, perform mutations.

Replace: Substitute the new population for the current population.

Test: Check to see if the end condition is met. If this is the case, stop. If not, move to Step 2 and return to the best solution for the current population.

3.2.3. The GA Approach for Container Scheduling

Guerrero et al. proposed the first container scheduling method based on the NSGA-II standard. The National Science Foundation created the multi-objective optimization method known as NSGA-II [27]. When creating their formulation, the authors took four optimization goals into account: resource utilization, failure rate, load balancing, and network communication overhead. The suggested method outperformed the researchers by 40–60% in the bulk of the intended objectives, according to the Kubernetes scheduling algorithm. Zhang et al. suggested an enhanced evolutionary algorithm based on a non-linear energy model to minimize energy usage [29]. e have created two new mutation operators to more effectively find the best answer. Conversely, Tan et al. designed the suggested plan with a single goal in mind: energy optimization. Tan et al. have presented an energy-saving two-level hybrid algorithm. Genetic programming is a population-based evolutionary computer strategy, much like genetic algorithms [22].

In this method, VMs are initially assigned to containers before the VMs are assigned to actual computers (PMs). Compared to heuristic-only-based techniques like first-fit, best-fit, and so forth, the hybrid approach greatly lowered energy usage, according to the experimental data. Imdoukh et al. proposed the NSGA-III-based Many-Objective Genetic Algorithm. The NSGA-III, an improved version of the NSGA-I, can handle more objective functions. They evaluated the system's performance and efficiency from various angles, including load balancing, scalability, power consumption, and resource availability and use. An examination in comparison to an ACO-based scheduling methodology proved the strategy's efficiency [30]. Tan et al. built on the work they had already done to solve the two-level container allocation problem using a hyper-heuristic method based on cooperative Coevolution Genetic Programming (CCGP) [22]. This method concurrently created allocation rules for both layers (containers-virtual machines and VMs-physical machines), lowering total energy usage.

Compared to the state-of-the-art algorithms, the results of the experiments revealed a significant reduction in energy consumption. The proposal of Dhumal and Janakiram C-Balancer proposes a scheduling framework that utilizes container runtime metrics to optimize container placement in a cluster environment [31]. Specifically, the suggested technique employs a GA-based optimizer to select the most reliable and effective container for node placement by regularly collecting container runtime information. The approach's fundamental idea is to rebalance containers by distributing them over several nodes using runtime data gathered during the migration process. The Swarm Cluster experiment demonstrated that the C-usefulness Balancers improved performance in terms of resource consumption and throughput [32].

3.3. Particle Swarm Optimization Algorithm

3.3.1. Natural Inspired

We devised the PSO algorithm to address the social behavior of animals such as schooling fish, swarming invertebrates, and avian migration. Researchers use this method to systematically search for the global optimum of numerous arbitrary problems. Kennedy and Everhart presented it for the first time [33]. The initial plans for this concept included simulating the graceful and unpredictable movements of a flock of birds to discover the patterns that govern their ability to fly synchronously and abruptly change directions while regrouping into an optimal formation, all with the goal of discovering new patterns. The correct term is stochastic, population-based evolutionary computing. Individuals' ability to maintain cognitive consistency depends on social influence and social learning. Therefore, the exchange of ideas and interactions between individuals may facilitate the resolution of problems. The particle swarm simulates this social structure. Li, Huang, and Wu claim that this method seeds the multidimensional search space of an objective function with a random set of particles, each assigned a specific position and velocity [34]. The objective function measures each particle as a potential solution to the problem. We characterize a collection of particles as a "swarm". During their voyage through multidimensional space, these particles utilize two essential reasoning skills: their memory of their optimal position and their knowledge of the optimal positions around the globe or in their immediate vicinity. In a minimization problem, "best" refers to the particle's (x_i) position with the minimum objective value, $\min f_{x_i}$. Members of a swarm exchange information about advantageous positions and adjust their position and velocity accordingly. Thus, a set of design variables (x_i) and the corresponding velocities (v_i) represent an optimization solution for each particle.

3.3.2. PSO Algorithm

First, a brief overview of the standard PSO algorithm is provided. Assume there are m particles in the colony that are being searched for in the D -dimensional space. D -dimensional vectors are used to represent the i th particle in the search space, which indicates that the particle is located as shown in Equation 8:

$$Xi = (X_{i1}, X_{i2}, X_{i3}, \dots, X_{id}) \quad (i = 1, 2, \dots, m) \quad (8)$$

Positioning each particle could be a solution to this problem. We can determine particle fitness by comparing the particle's position to a designated objective function [34], which is "better" when the fitness level is higher. The i th particle's "flying" velocity is represented by a D -dimensional vector, as shown in Equations 9, 10, and 11:

$$vi = (vi1, vi2, \dots, viD) \quad (9)$$

99

$$I = 1, 2, \dots, m \quad (10)$$

$$Pi = (pi1, pi2, \dots, pgD) \quad (11)$$

The particle's optimal position () and the colony's optimal position () are indicated. The PSO algorithm can be implemented using the following Equations 12, 13:

$$Xi(k + 1) = Xi(k) + Vi(k + 1) \quad (12)$$

$$Vi(k + 1) = wVi(k) + c1r1 Pi - xi(k) + c2r2 Pg - xi(k) \quad (13)$$

The inertia coefficient w , nonnegative constant learning rates $c1$ and $c2$, random generation of $r1$, and the inertia coefficient w are all constants in the interval $[0, 1]$. $Pbest$ and fitness $gbests$ are used to determine whether iterations should be terminated when they have reached their maximum generation or a predetermined value[35].

3.3.3. PSO for Container Scheduling

Li et al. improved resource utilization and load balancing with a PSO algorithm [34]. To get around the local minimum, the authors implemented a simulated annealing algorithm with PSO. In a series of tests, the suggested implementation of Docker Swarm outscored the spread technique by 20. As a scheduling technique based on PSO, Guo and Yao suggested employing the neighborhood division idea to fine-tune the PSO algorithm's parameters for better-quality solutions [35]. To enhance system performance, the algorithm took into account load balancing as well as reaction time. In the suggested approach, non-dependent containers were put on nearby hosts to minimize their connection [8], [36]. Compared to the existing and PSO algorithms, the results showed a 20–25 percent improvement. The increased power consumption associated with the developed solution was one of its drawbacks. With the help of a two-stage multi-type particle swarm optimization (TMPSO) algorithm, we were able to solve the container consolidation problem [21]. To increase the quality of the results, the TMPSO algorithm employed a combination of heuristic and greedy strategies. When comparing conventional and binary PSO algorithms, the experimental findings revealed a significant reduction in energy consumption. Adhikari and Srirama proposed, to reduce energy consumption and computing time, a multi-objective particle swarm method for scheduling containers for Internet of Things jobs [37].

Additionally, the suggested method accounted for lowering CO2 emissions and calculating node temperatures. For big data applications, Liu et al. proposed the Kubernetes container scheduling algorithm, K-PSO, based on particle swarm optimization [8]. The authors added a dynamic inertia weight and learning factor to the standard PSO, allowing faster convergence with higher-quality solutions. The authors added a new predicate scheduling process and priority scheduling process to the Kubernetes native scheduler to leverage the improved PSO.

The experiments revealed a 20 percent increase in resource utilization compared to Kubernetes' default scheduling strategy. Conversely, we can expand the suggested approach to include multi-criteria optimization objectives. To decide on container-based micro-service deployment in an edge computing paradigm, Fan et al. presented an edge computing-based scheduling method based on PSO [38]. The algorithm knows latency, reliability, and load balancing (LRLBAS). We demonstrated the efficacy and efficiency of the treatment by comparing trial results with PSO versions. Edge computing has demonstrated the effectiveness of the LRLBAS algorithm for micro-service scheduling. In the future, the authors intend to incorporate additional objectives into their proposal.

4. Discussion

The issue is allocating container slots to existing or new VMs and assigning those VMs to PMs while simultaneously reducing the power consumption of all PMs. The data center contains a variety of heterogeneous virtual machines and homogeneous PMs. Each virtual machine includes a set of preconfigured resources and overheads. The first container's operating system requirements determine the virtual machine's operating system type. The situation necessitates four decisions, including VM selection, VM creation, PM selection, and PM generation. The section also discusses two optimization techniques for container adaptation: ACO and GA. MOCP-ACO outperformed the Docker Swarm deployment scheduling method in every scenario tested, while the NSGA-II-based method outperformed the Kubernetes scheduling algorithm by 40–60% for many

targets. In comparison to heuristic-only techniques, the enhanced evolutionary algorithm and the two-level hybrid algorithm substantially reduced energy consumption. Compared to the most recent algorithms, the multi-objective genetic algorithm based on NSGA-III and the collaborative genetic programming approach based on the over-exploration method yielded significant results. Table 1 provides a comparison of selected works from the literature.

Table 1. Comparison of Selected Works from the Literature

Compared Algorithms	Goal								Tool	Algorithm	Limitations
	Energy	Availability	Utilizing	Balance	Explanation	Cost	Make-span	Connectivity			
ACO			*	*					Docker	-First ACO approach -Effective use of resources	-No container migration -Required parameter tuning
MOCF-ACO						*		*	Docker	-Modified ACO methodology -Be aware of network latency	-Is slow and disregards energy
MOACO	*	*	*					*	-	-Reduce network cost -No -Use resources efficiently	-Container consolidation -No energy education
NSGA-II	*	*	*					*	Kubernetes	-First GA with multiple objectives based on NSGA-II	-No container migration -No energy reduction
Improve-GA	*								-	Non-linear energy model, with an emphasis on energy savings	-Requires parameters tuning
2-level-hybrid-GA	*								-	-Genetic programming	-Single objective

										-Two-level optimization	-No cooperative evolution
MOGAS	*	*	*	*	*				Docker	-Take into account five goals based on NSGA-III	-Slow, need parallelization
Improve-PSO			*	*					Docker	-First PSO model; enhanced performance	-Not fault tolerant; needs settings to be adjusted
PSO-based- Scheduling				*			*	*	CloudSim	-Immediate -No migration of the container -Uses more RAM	-Improved efficiency
TMPSO	*		*						-	-Objective PSO -Fast	-No movement of containers
PSO-for-IoT tasks	*		*				*		CloudSim	-Think about CO2 emissions -Attention to temperature	-Needs settings to be tuned
Vhatkar and Bhole		*		*				*	-	-Hybrid whale-Lion model, incorporating availability into account	-Inadequate testing in a real-world environment; parameter-twisting
OC-balance-GA			*				*		Docker	-GA-based strategy - Migration of containers	-Needs settings to be tuned
PSO-based-container Scheduling			*						Kubernetes	-PSO method modification - Fast convergence	-Only a single goal

LRLBAS		*		*			*			-Edge computing paradigm -Objective PSO	-No energy savings, no real-world testing
--------	--	---	--	---	--	--	---	--	--	--	---

5. Conclusion

With the proliferation of IoT services, the use of containers in the cloud computing community to provide cloud services has increased dramatically in recent years. Container scheduling has grown to become a significant component of cloud computing architecture to ensure proper management of cloud resources during runtime. We conduct a comprehensive analysis in this paper to examine the landscape of container scheduling methods. To start, we divided scheduling strategies into four groups depending on the optimization method that was used to create the schedule. Following that, we considered optimization goals to determine how well the created schedules performed. Then, based on performance indicators, we categorized and described current strategies for each category to understand their benefits and drawbacks. A new generation of resource management and scheduling systems will be required because of container technology, according to present trends and probable future research paths. New developments in technology, whether fuzzy or cloud computing, provide opportunities and new horizons for researchers in reducing the power and increasing the security and communications of these settings and micro-services.

References

- [1] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus. Horiz.*, vol. 58, no. 4, pp. 431–440, Jul. 2015, doi: 10.1016/J.BUSHOR.2015.03.008.
- [2] W. W. W. Gartner, "Gartner says the Internet of Things will transform the data center."
- [3] Y. Alahmad, T. Daradkeh, and A. Agarwal, "Availability-Aware Container Scheduler for Application Services in Cloud," *2018 IEEE 37th Int. Perform. Comput. Commun. Conf. IPCCC 2018*, Jul. 2018, doi: 10.1109/PCCC.2018.8711295.
- [4] M. Alouane and H. El Bakkali, "Virtualization in Cloud Computing: Existing solutions and new approach," *Proc. 2016 Int. Conf. Cloud Comput. Technol. Appl. CloudTech 2016*, pp. 116–123, Feb. 2017, doi: 10.1109/CLOUDTECH.2016.7847687.
- [5] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment", Accessed: May 10, 2023. [Online]. Available: <http://www.docker.io>
- [6] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic Virtual Machine Scheduling in Cloud Datacenters towards Minimizing Total Energy," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1317–1331, Jun. 2018, doi: 10.1109/TPDS.2017.2688445.
- [7] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019, doi: 10.1109/ACCESS.2019.2924414.
- [8] B. Liu, J. Li, W. Lin, W. Bai, P. Li, and Q. Gao, "K-PSO: An improved PSO-based container scheduling algorithm for big data applications," *Int. J. Netw. Manag.*, vol. 31, no. 2, p. e2092, Mar. 2021, doi: 10.1002/NEM.2092.
- [9] F. Chen, X. Zhou, and C. Shi, "The container scheduling method based on the min-min in edge computing," *ACM Int. Conf. Proceeding Ser.*, pp. 83–90, May 2019, doi: 10.1145/3335484.3335506.
- [10] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Futur. Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019, doi: 10.1016/J.FUTURE.2018.09.014.
- [11] P. J. Maenhaut, B. Volckaert, V. Ongenae, and F. De Turck, "Resource Management in a Containerized Cloud: Status and Challenges," *J. Netw. Syst. Manag. 2019 282*, vol. 28, no. 2, pp. 197–246, Nov. 2019, doi: 10.1007/S10922-019-09504-0.
- [12] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Comput. Sci. Rev.*, vol. 24, pp. 63–79, May 2017, doi: 10.1016/J.COSREV.2016.12.001.
- [13] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artif. Intell. Rev. 2018 524*, vol. 52, no. 4, pp. 2191–2233, Jan. 2018, doi: 10.1007/S10462-017-9605-Z.
- [14] S. Pouyanfar *et al.*, "A Survey on Deep Learning," *ACM Comput. Surv.*, vol. 51, no. 5, Sep. 2018, doi: 10.1145/3234150.
- [15] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "VNF and CNF Placement in 5G: Recent Advances and Future Trends," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 4, pp. 4698–4733, Dec. 2023, doi:

- 10.1109/TNSM.2023.3264005.
- [16] S. B. Nath, S. Chattopadhyay, R. Karmakar, S. K. Addya, S. Chakraborty, and S. K. Ghosh, "PTC: Pick-test-choose to place containerized micro-services in IoT," *Proc. - IEEE Glob. Commun. Conf. GLOBECOM*, 2019, doi: 10.1109/GLOBECOM38437.2019.9013163.
- [17] H. M. R. Al-Khafaji, "Improving Quality Indicators of the Cloud-Based IoT Networks Using an Improved Form of Seagull Optimization Algorithm," *Futur. Internet* 2022, Vol. 14, Page 281, vol. 14, no. 10, p. 281, Sep. 2022, doi: 10.3390/FI14100281.
- [18] P. Satyanarayana, G. Diwakar, B. V. Subbayamma, N. V. Phani Sai Kumar, M. Arun, and S. Gopalakrishnan, "Comparative analysis of new meta-heuristic-variants for privacy preservation in wireless mobile adhoc networks for IoT applications," *Comput. Commun.*, vol. 198, pp. 262–281, Jan. 2023, doi: 10.1016/J.COMCOM.2022.12.006.
- [19] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Multi-objective scheduling of micro-services for optimal service function chains," *IEEE Int. Conf. Commun.*, Jul. 2017, doi: 10.1109/ICC.2017.7996729.
- [20] C. Kaewkasi and K. Chuenmuneewong, "Improvement of container scheduling for Docker using Ant Colony Optimization," *2017 9th Int. Conf. Knowl. Smart Technol. Crunching Inf. Everything, KST 2017*, pp. 254–259, Mar. 2017, doi: 10.1109/KST.2017.7886112.
- [21] T. Shi, H. Ma, and G. Chen, "Energy-Aware Container Consolidation Based on PSO in Cloud Data Centers," *2018 IEEE Congr. Evol. Comput. CEC 2018 - Proc.*, Sep. 2018, doi: 10.1109/CEC.2018.8477708.
- [22] B. Tan, H. Ma, and Y. Mei, "A Hybrid Genetic Programming Hyper-Heuristic Approach for Online Two-level Resource Allocation in Container-based Clouds," *2019 IEEE Congr. Evol. Comput. CEC 2019 - Proc.*, pp. 2681–2688, Jun. 2019, doi: 10.1109/CEC.2019.8790220.
- [23] S. S. Gill and R. Buyya, "A Taxonomy and Future Directions for Sustainable Cloud Computing," *ACM Comput. Surv.*, vol. 51, no. 5, Dec. 2018, doi: 10.1145/3241038.
- [24] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.
- [25] B. Burvall, "Improvement of Container Placement Using Multi-Objective Ant Colony Optimization," *DEGREE Proj. Comput. Sci. Eng.*, 2019, Accessed: May 10, 2023. [Online]. Available: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-249709>
- [26] J. Yang, X. Shi, M. Marchese, and Y. Liang, "An ant colony optimization method for generalized TSP problem," *Prog. Nat. Sci.*, vol. 18, no. 11, pp. 1417–1422, Nov. 2008, doi: 10.1016/J.PNSC.2008.03.028.
- [27] C. Guerrero, I. Lera, and C. Juiz, "Genetic algorithm for multi-objective optimization of container allocation in cloud architecture," *J. Grid Comput.*, vol. 16, no. 1, pp. 113–135, Nov. 2017, doi: 10.1007/S10723-017-9419-X/METRICS.
- [28] C. Teixeira, J. A. Covas, T. Stützle, and A. Gaspar-Cunha, "Multi-objective ant colony optimization for the twin-screw configuration problem," <https://doi.org/10.1080/0305215X.2011.639370>, vol. 44, no. 3, pp. 351–371, Mar. 2012, doi: 10.1080/0305215X.2011.639370.
- [29] R. Zhang, Y. Chen, B. Dong, F. Tian, and Q. Zheng, "A Genetic Algorithm-Based Energy-Efficient Container Placement Strategy in CaaS," *IEEE Access*, vol. 7, pp. 121360–121373, 2019, doi: 10.1109/ACCESS.2019.2937553.
- [30] M. Imdoukh, I. Ahmad, and M. Alfaiakawi, "Optimizing scheduling decisions of container management tool using many-objective genetic algorithm," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 5, p. e5536, Mar. 2020, doi: 10.1002/CPE.5536.
- [31] A. Dhumal and D. Janakiram, "C-Balancer: A System for Container Profiling and Scheduling," Sep. 2020, Accessed: May 10, 2023. [Online]. Available: <https://arxiv.org/abs/2009.08912v1>
- [32] L. Li, J. Chen, and W. Yan, "A particle swarm optimization-based container scheduling algorithm of docker platform," *ACM Int. Conf. Proceeding Ser.*, pp. 12–17, Nov. 2018, doi: 10.1145/3290420.3290432.
- [33] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. ICNN'95 - Int. Conf. Neural Networks*, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
- [34] L. J. Li, Z. B. Huang, F. Liu, and Q. H. Wu, "A heuristic particle swarm optimizer for optimization of pin connected structures," *Comput. Struct.*, vol. 85, no. 7–8, pp. 340–349, Apr. 2007, doi: 10.1016/J.COMPSTRUC.2006.11.020.
- [35] Y. Guo and W. Yao, "A container scheduling strategy based on neighborhood division in micro service," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–6, Jul. 2018, doi:

10.1109/NOMS.2018.8406285.

- [36] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP," *Inf. Process. Lett.*, vol. 103, no. 5, pp. 169–176, Aug. 2007, doi: 10.1016/J.IPL.2007.03.010.
- [37] M. Adhikari and S. N. Srirama, "Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment," *J. Netw. Comput. Appl.*, vol. 137, pp. 35–61, Jul. 2019, doi: 10.1016/J.JNCA.2019.04.003.
- [38] G. Fan, L. Chen, H. Yu, and W. Qi, "Multi-objective optimization of container-based microservice scheduling in edge computing," *Comput. Sci. Inf.*, vol. 18, no. 1, pp. 23–42, 2021, Accessed: May 12, 2023. [Online]. Available: <http://www.doiserbia.nb.rs/Article.aspx?id=1820-02142000041F>

Author(s) Contributions

Murat Koca: Algorithm generation, Optimization of Algorithms, writing, review, and editing.

İsa Avcı: Comparison of Algorithms, Writing, review, and editing.

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable

Plagiarism Statement

This article has been scanned by iThenticate™.

Distributed Task Allocation for UAV Swarms with Limited Communication

Mutullah Eşer¹ , Asım Egemen Yılmaz¹ 

¹Department of Electrical and Electronics Engineering, Ankara University, Ankara, Türkiye

Corresponding author:

Mutullah Eşer, Department of
Electrical and Electronics Engineering,
Ankara University, Ankara, Türkiye
eserm@ankara.edu.tr



Article History:

Received: 30.03.2024

Accepted: 26.07.2024

Published Online: 23.08.2024

ABSTRACT

Unmanned aerial vehicle (UAV) swarms have become increasingly indispensable in both military and civilian operations. Task allocation, a crucial aspect of UAV swarm autonomy, involves assigning sequential tasks to each aircraft based on environmental constraints and swarm status. While many task allocation algorithms assume reliable communication among agents, real-world environments often present challenges such as limited bandwidth and message interference. This study presents a new distributed task assignment algorithm for heterogeneous UAV swarms, addressing various task constraints. The proposed auction-based method optimizes total cost, ensures fair workload distribution, and minimizes message size through a two-stage auction process. Comparative evaluations with existing algorithms like CBBA and the central Hungarian algorithm, under the Bernoulli communication model, consider factors such as total task cost, message size, unassignable tasks, and conflict assignments. Results indicate the proposed algorithm's effectiveness in smooth communication environments and its potential advantage in low-bandwidth environments. However, it also highlights potential conflicts in scenarios with communication disruptions. To address deviations due to communication quality, Signal-to-Noise Ratio (SNR) values are monitored throughout task execution.

Keywords: UAV swarms, Distributed computing, Auction algorithm, Task allocation, Constrained communication

1. Introduction

An Unmanned Aerial Vehicle (UAV) denotes an aircraft that operates without a human pilot onboard, either under remote control by a human operator or autonomously following pre-programmed instructions. In recent years, there has been substantial progress and utilization of UAVs across various sectors. Originally dominant in military use, UAVs have increasingly ventured into civilian domains over the last twenty years, thanks to advancements in technology, smaller sizes, and decreased component expenses, making them accessible even to hobbyists [1]. While primarily acknowledged for aerial photography, potential civilian applications encompass delivery services, agriculture, law enforcement, environmental monitoring, and entertainment [2].

Scholars have explored the prospect of collaboration among multiple UAVs, envisioning tasks including surveillance, surveying, search and rescue operations, monitoring/detection in hazardous environments, terrain mapping, and handling hazardous materials [3]. The diminishing size and costs of individual units have also spurred interest in deploying UAVs in swarms [4]. This area of research is appealing due to its potential robustness, flexibility, and scalability. Robustness in this context pertains to the redundancy offered by large numbers and decentralization. A swarm potentially can accomplish objectives that a single UAV cannot or would achieve with less efficiency.

UAV swarms may comprise either homogeneous or heterogeneous groups. Employing a homogeneous group simplifies the acquisition and utilization of a cost-effective swarm, while heterogeneous groups enhance the swarm's complexity and capability.

Advancements in artificial intelligence and machine learning are transforming combat operations, with a growing inclination toward employing autonomous robots capable of reacting and cooperating as a cohesive unit [5]. The swarm represents the next evolutionary phase in warfare. UAV systems are employed across various levels, from terrorist organizations to global superpowers, and cost-effective UAV systems serve as a means to implement swarm warfare. Already, UAV swarms are utilized in heterogeneous configurations and have been demonstrated during military exhibitions [6].

The collaboration of drones within a swarm poses additional challenges not encountered in single UAV operations. Swarms operating as a unified entity require systems for collision avoidance among the drones within the swarm itself and avoidance

of obstacles encountered in the environment [7]. Failures in these systems can result in material damage and escalate the costs of developing such systems.

The global progression of UAV and swarm technologies continues, with recent advancements including Russia's "Molniya" UAV, designed to compete with the U.S. Gremlin system. It boasts 700 km/h speeds and payloads ranging from 5 to 7 kgs. Additionally, Russia's ZALA Aero has introduced the KUB-UAV, specifically tailored for naval operations. It can reach up to 130 km/h, endure for 30 minutes, and carry a three-kilogram warhead, posing a significant threat due to its rapid swarm formation capability [8]. Turkish UAV technology has also seen rapid progress, exemplified by platforms like the Bayraktar TB2, which has gained international acclaim for its efficacy in diverse military operations. China has demonstrated its dedication to swarm technology, notably in a 2017 display involving 119 UAVs, and continues to advance with companies like Zhuhai Ziyang developing versatile systems like the Blowfish family, capable of various missions with interchangeable payloads, reflecting the necessity for effective and cost-efficient solutions in future battlegrounds [9]. The U.S. deploys UAV swarms such as Predator and Reaper extensively for remote surveillance and target acquisition tasks. Similarly, Israel's UAV systems like Harop and Heron TP play active roles in defensive operations. In the civilian sphere, UAV swarms for delivery purposes, including Amazon's Prime Air, are under development.

One of the primary challenges associated with UAV swarms is the inherent difficulty in effectively coordinating them. Many coordination challenges involve solving NP-hard problems, making optimal solutions often difficult to attain and exhibiting poor scalability [10]. Among the focal points of research in UAV swarm coordination is task assignment, which presents an optimization challenge involving tasks located in varying positions and of differing significance, to be executed by UAVs with diverse capabilities and positions.

In recent years, Task Allocation (TA) has emerged as a prevalent issue in swarm robotics [11]. Given that swarm robotics applications frequently require a group of robots to perform multiple tasks, it becomes crucial to assign these tasks to robots in an optimized manner. Generally, the task allocation problem involves finding an appropriate matching or allocation, given a set of tasks, a set of robots capable of performing these tasks, and an objective function that evaluates the performance efficiency of different combinations of robots in task execution [12].

The primary factor influencing the effectiveness of a UAV swarm's mission is its ability to communicate and share information among its constituent drones. The main constraint on the size of a UAV swarm is its capacity to manage the exchange of information. Effective communication is crucial for preventing collisions among swarm drones and coordinating attack strategies. Without cohesive communication among UAVs, a swarm cannot maintain consistent functionality or achieve successful missions. Essentially, communication is a fundamental requirement for the existence and operation of a UAV swarm [13].

Existing approaches for task allocation among agents in a swarm primarily focus on distributing tasks without considering the inherent structure. They overlook factors like UAV hardware, battery levels, task time constraints, complexity, and communication bandwidth. Additionally, the impact of lossy communication on algorithm performance is often neglected in proposed algorithms.

The main contributions of this paper are the following:

- Defining the task assignment problem within the communication framework of UAV swarms.
- Introducing a novel method for task assignment in UAV swarms, with an analysis of its performance in scenarios with communication impairments.
- Demonstrating the feasibility of the proposed algorithm through comprehensive evaluations using Matlab simulations.
- Conducting a comparative evaluation of the proposed algorithm against widely utilized task assignment methods such as Hungarian and CBBA algorithms, specifically addressing communication challenges.
- The algorithm proposed is demonstrated through task allocation within drone swarms, but it offers a framework for parallel computing across various types of autonomous swarms.

The rest of this paper is organized as follows. In section 2 we discuss related work. In section 3, we describe task allocation methods. In section 4, we describe the problem. Section 5 introduces our proposed algorithm. Section 6 presents the results of the simulation. Section 7 is the conclusion and future work.

2. Related Work

In this section, we highlight recent studies on the task assignment problem in UAV swarms. Unmanned aerial vehicles have garnered significant attention due to their diverse applications. Various studies have examined the advantages and use cases of UAVs, exploring their potential in areas such as surveillance, search and rescue, and environmental monitoring. By analyzing these recent contributions, we aim to identify the current trends, advancements, and challenges in the field, and position our work within this context.

Clough [14] delves into the advantages of agents collaborating within a swarm. The benefits of using heterogeneous swarms over homogeneous ones are highlighted in recent studies, such as [15]. Heterogeneity in robotics involves employing robots with varied capabilities, features, or characteristics within a swarm. Frelinger et al. [16] demonstrated that enabling cooperative behavior through communication among weapons significantly enhances effectiveness. Despite limited sensor ranges, they share target detection information among neighboring weapons compensated for this shortfall. As a result, more targets within the designated area were successfully engaged, leading to improved overall system performance. Recent developments have also showcased the first instances of manned fighters engaging with unmanned drones [17, 18]. Edwards [19] provides an extensive study of historical battles where at least one force operated as a swarm.

The task assignment problem can manifest in different ways depending on the characteristics of the agents and the tasks. It is crucial to define the problem clearly and seek solutions accordingly to find an effective resolution. In 2004, Gerkey and Mataric [20] introduced a domain-independent classification framework for the multitask assignment problem, which remains influential in contemporary research. This classification system delineates key components such as single-task (ST) and multi-task (MT) robots, single-robot (SR) and multi-robot (MR) tasks, and instant (IA) and timed (TA) assignments. Its primary aim is to provide a structured understanding of how diverse task assignment challenges fit within the problem domain and to elucidate their relationships with solutions proposed in existing literature.

The first step in solving the task assignment problem is to express the constraints of the problem and model it accordingly. Reference [21] constructed a multi-UAV task allocation model that incorporates an interval information environment to account for uncertain factors such as revenue damage cost index, target value, and range cost index. In a similar vein, [22] tackled dynamic task allocation by breaking it down into static task allocation across multiple stages, leading to the creation of a multi-UAV dynamic reconnaissance resource allocation model, thus enhancing overall efficiency in dynamic reconnaissance tasks. Shima T [23] introduced a cooperative multiple task assignment problem model tailored for UAVs. At the same time, a subsequent paper [24] refined this model into an extended cooperative multitask assignment model, albeit with some constraints left unaddressed. Addressing the challenge of integrating new tasks or accommodating platform loss, [25] proposed a method of dynamic task local adjustment to formulate a multi base and multi-unmanned combat aerial vehicle task allocation model, thereby bolstering task allocation efficiency and platform stability.

The foundational research on the task assignment problem is documented in [26], where the initial models were developed. Further elaboration on this proposed approach can be found in [27]. Task assignment methodologies are broadly classified into two categories: centralized and decentralized.

Centralized methods are effective for determining optimal task assignments in static environments with few agents. However, as the number of agents increases and the problem becomes more complex, scaling centralized approaches becomes challenging. The seminal work in centralized task assignment, known as the Hungarian algorithm, was introduced by Kuhn, a Hungarian mathematician, in [28].

Distributed task assignment methods are more commonly used than centralized methods due to their advantages such as flexibility and scalability. In [29], the analysis of task allocation within a multi-UAV system adopts a fully distributed architecture. A key characteristic of this approach is the avoidance of centralized planning, instead relying on synchronization through token circulation. However, achieving this synchronization necessitates all-to-all communication. Auction-based algorithms represent the predominant approach for distributed task assignment, wherein agents bid on tasks and the highest bidder secures the task [30]. The determination of auction winners can be either centralized or decentralized. The inception of auction-based task assignment algorithms dates back to Dimitri's proposal in 1979 [31]. Various auction mechanisms have been employed for task allocation in multi-agent systems. In a sequential auction, the auctioneer sells a sequence of items, one item at a time, in an order chosen by the auctioneer. Lagoudakis et al. [32] compare several distributed bidding mechanisms, demonstrating that a variant of sequential auctions offers solutions that are provably close to optimal. Simmons et al. [33] introduced robots bidding to visit frontier nodes following map updates. Rekleitis et al. [34] utilized a sequence of one-round auctions to divide a Boustrophedon multi-robot coverage sweep. Similarly, Vail and Veloso [35] employed one-round auctions sequentially for role assignment in robot soccer. Approximation methods for combinatorial auctions are compared to sequential single item auctions by Cavalcante et al. [36]. Schneider et al. [37] provide experimental comparisons among various auctions, market approaches, and other multi-robot coordination mechanisms. In static multirobot task assignment, the auction-based algorithm in [38] ensures that the total travel cost of the robots is at most twice that of the optimal solution, provided all the robots are communication-connected. In [39], the auction-based CBBA algorithm, which was proposed for task assignment in static environments, has been extended for dynamic environments.

Several auction-based task assignment algorithms are customized to allocate tasks among agents in natural disasters and catastrophic situations. López et al. [40] introduced an auction system named Masictus designed to coordinate ambulances and neurologists from various ambulance trusts, particularly in instances involving stroke patients. The system employs real currency in conducting auctions to allocate ambulances that incur the lowest costs and are in close proximity to the emergency locations. Another application relating auction mechanisms to disaster management is presented by Berhault et al. [41] where combinatorial auctions are utilized to determine a schedule for a group of robots visiting target areas. Combinatorial auctions involve bidders placing bids on combinations of items to secure them. Finally, [42], [43], and [44] present various methods proposed for distributed task assignment.

Our algorithm fills a significant gap in current research by presenting a unique method for task assignment in UAV swarms, specifically designed for environments with limited communication bandwidth. Traditional task allocation algorithms typically assume stable and robust communication channels, which is often not the case in practical UAV swarm operations. Our approach, however, directly addresses these real-world communication challenges. By tackling this critical issue, our study offers an innovative solution for task allocation in UAV swarms operating under restricted communication conditions, enhancing the field with a new method that ensures efficient and effective swarm coordination.

3. Task Allocation in UAV Swarms

A primary challenge in autonomous multi-agent systems is the lack of coordination among agents, often stemming from an inefficient distribution of tasks. Task assignment plays a crucial role in determining agents' objectives and ensuring successful task completion, facilitating collaboration towards shared goals.

Task assignment involves the selection, allocation, and coordination of tasks [45]. In UAV swarms, agents, while similar in design, can vary in capabilities, necessitating task assignments compatible with their strengths. Additionally, agents may need to coordinate to tackle complex tasks requiring multiple agents, where the duration of task completion may be uncertain, making it challenging to determine when tasks might become irrelevant [46].

Approaches to task allocation in UAV swarms typically fall into two categories: centralized methods and distributed methods.

Centralized task assignment algorithms involve a central coordinator agent communicating with all other agents, manages negotiations and makes decisions on task assignments. While capable of producing optimal or near-optimal assignments, these algorithms require continuous and reliable communication between each robot and the central server to gather global information. This reliance on effective communication often leads to significant central node overhead, complex computation, and a single point of failure [47]. Moreover, certain multi robot task assignment problems are proven to be NP-hard, demanding high computational capacity from the central server [48]. Consequently, centralized methods face scalability issues with increasing numbers of robots and/or targets, and they are not suitable for scenarios where robots may encounter local or outdated information due to imperfect communication.

The Hungarian algorithm [28], introduced by the mathematician Kuhn in 1955, is a foundational centralized method for task assignment. This algorithm, known for its simplicity and wide applicability, efficiently solves assignment problems in polynomial time.

In contrast, distributed methods eliminate the need for a central agent, with task allocation occurring through negotiations among agents. Distributed systems operate under the principle that no agent possesses complete information about the system or control over other agents' actions [49]. In such systems, UAVs communicate with each other to exchange information, leveraging their autonomous capabilities to enhance assignment efficiency [50]. Decentralized algorithms empower each robot to plan its route based on locally available information.

In recent years, auction-based distributed techniques, renowned for their computational efficiency, have gained traction in addressing the task assignment problem within distributed environments. These approaches involve UAVs determining their bids for individual tasks based on internal valuation or cost metrics. Depending on whether the objective is to maximize value or minimize cost, the task is awarded to the highest or lowest bidder, respectively [51].

CBBA (Consensus Based Bundle Algorithm), an approach rooted in open incrementality, has emerged as a prominent method for task assignment, as outlined in [52]. Within CBBA, each agent maintains a list of tasks potentially allocated to them, and the auction mechanism operates at the task level rather than the package level. CBBA comprises two key stages: bundling and conflict resolution. Unlike other consensus algorithms, CBBA focuses on achieving consensus on the winning bid instead of individual agent strategies. Leveraging a parallel task assignment model, CBBA exhibits faster convergence compared to serial auction algorithms. Its primary objective is to allocate tasks without conflicts, ensuring that each task is assigned to only one UAV. However, in practical scenarios, certain tasks may necessitate collaboration among multiple UAVs with diverse capabilities. To address this, CBBA-based applications adopt a strategy of duplicating these collaborative tasks with corresponding dimensions [53]. For instance, if task j requires n UAVs for cooperative execution, it is replicated into n identical tasks $(j_1, j_2, j_3, \dots, j_n)$, treating each as a separate task during the assignment process.

Various methodologies exist in the literature to tackle the task assignment problem, encompassing centralized, distributed, and hybrid approaches, each with distinct characteristics. The selection of the optimal task assignment algorithm for UAV swarms hinges on specific application requirements, swarm size, communication capabilities, and environmental considerations. Table 1 presents a comparative analysis of centralized, distributed, and hybrid task assignment algorithms, highlighting their key features.

Table 1. Task Assignment Approaches

Feature	Centralized	Distributed	Hybrid
Scalability	No	Yes	Yes
Robustness	No	Yes	Yes
Flexibility	No	Yes	Yes
Handling unknown conditions	No	Yes	Partially
Communication obligation	High	Low	High
Team size	Small	Big	Medium
Global information need	Yes	No	No

UAV swarms are often deployed in environments where communications are unreliable. Communication can be unreliable due to weather and environmental factors such as obstacles, distance between robots, and interference. Communication may also be blocked intentionally to maintain confidentiality.

The challenge of limited bandwidth or data rate in wireless communication becomes significant when dealing with large numbers of agents operating within communication range. The constraints of current communication technology are often overlooked in the development of algorithms for aerial swarm applications. Many of these algorithms assume an arbitrarily large bandwidth for communication, assuming that neighbor information is continuously available to an agent or at every time step [54]. Recent surveys [55] indicate that decentralized topologies are underutilized due to limitations in current technology. Successful experiments reported in the literature usually involve limiting the number of agents or implementing larger safety distances with relatively low speeds to accommodate delays or dropouts [56].

In the context of a task assignment problem, the most challenging scenario arises when an agent cannot establish communication with other agents and must navigate to all designated targets. While many task allocation methodologies presuppose flawless inter-agent communication among Unmanned Aerial Vehicles (UAVs), positing an ideal connection with unlimited bandwidth for ensuring consistent situational awareness (SA) before allocation, contemporary communication technologies fall short of meeting this criterion [57].

In instances of inconsistent communication, the effectiveness of the task assignment algorithm assumes critical significance. In line with this, within the purview of this research endeavor, section 6 comprehensively examines the performance of the proposed algorithm in an environment characterized by weakened communication and potential message loss.

4. Problem Statement

In this section, initially, the mathematical model and constraints are delineated for the task assignment problem in UAV swarms amidst dynamic environments. Subsequently, the communication model employed for assessing the efficacy of the proposed task assignment algorithm is illustrated.

4.1. Task Allocation Problem

The essence of the task assignment problem lies in matching a set of tasks with a corresponding set of agents, driven by an objective function. Within a UAV swarm context, N UAVs exist, each possessing distinct capabilities. Represented as A , this collection constitutes N heterogeneous UAVs, as formulated in Equation 1.

$$A = \{1, 2, \dots, N\} \quad (1)$$

The system encompasses M dynamic tasks, with the task set denoted as T . This is represented by Equation 2. These tasks exhibit heterogeneity, meaning there is no uniform type; rather, UAVs with fitting capabilities are requisite for task execution.

$$T = \{1, 2, \dots, M\} \quad (2)$$

The quantity of potential assignments achievable within a swarm is contingent upon the count of agents and tasks. If the agents within the ensemble are homogeneous, the number of feasible assignments, denoted as S , is defined by Equation 3.

$$S = N^M \quad (3)$$

When the agents in the swarm exhibit heterogeneity, the count of feasible assignments is articulated by Equation 4.

$$S = \prod_{i=1}^n N_i^{M_i} = N_1^{M_1} \times N_2^{M_2} \times \dots \times N_n^{M_n} \quad (4)$$

where n represents the number of groups within the swarm. Each N_i agent group is tasked with executing the M_i task group.

The primary objective of task assignment is to allocate tasks in a manner that maximizes the total benefit derived from the UAVs. This objective is expressed mathematically in Equation 5.

$$\max \sum_{i=1}^N \sum_{j=1}^M x_{ij} R_{ij} \quad (5)$$

If task j is assigned to agent i , then $x_{ij}=1$; otherwise, $x_{ij}=0$. R_{ij} denotes the utility value agent i will attain if assigned task j .

The task assignment problem constitutes an optimization challenge, typically geared towards maximizing resource efficiency or achieving optimal outcomes within specific constraints. The assignment problem is fundamentally linked to linear optimization, making it a fundamental difficulty. This issue commonly arises in scenarios where multiple tasks require the allocation of resources, necessitating a strategic assignment approach.

The constraints and assumptions defined for the problem within the scope of the study are as follows:

Various types of aircraft possess distinct capabilities tailored to specific missions. Aircraft must be assigned tasks that align with their respective capabilities to ensure effective execution. Each agent is associated with an ability matrix (A) that delineates its abilities. If agent i can perform task j , $a_{ij} = 1$, otherwise $a_{ij} = 0$.

$$A_i = [a_{i1} \quad a_{i2} \quad \dots \quad a_{in}] \quad (6)$$

One aircraft must be assigned to each mission, this constraint is expressed in Equation 7.

$$\sum_{i=1}^N x_{ij} = 1, j \in T \quad (7)$$

A time frame is established for each task in Equation 8, within which the task must commence. Additionally, each task demands a duration t_{td} for completion. Upon reaching the task location, the UAV is obligated to remain at the task position for this specified duration of time.

$$t_{start_j} \leq t_{d_{ij}} \leq t_{finish_j} \quad (8)$$

where $[t_{start_j}, t_{finish_j}]$ represents the time frame of task j , and $t_{d_{ij}}$ represents the commencement time for agent i to execute task j .

The allocation of tasks to UAVs should be balanced, aiming to distribute the workload evenly among the UAVs. This constraint is articulated in Equation 9.

$$|s_i| - |s_k| \leq B, i \neq k \quad i, k \in A \quad (9)$$

where $|s_i|$ is the number of tasks in the task list of drone i , $|s_k|$ is the of tasks in the task list of drone k and B is the threshold value for the difference in the number of assigned tasks.

The total distance traveled by the UAV must not exceed the range of the UAV. This constraint is expressed mathematically by Equation 10.

$$\sum_{j=1}^M S_{ij} x_{ij} \leq D_i \quad (10)$$

where D_i is the maximum range of the UAV, S_{ij} is the distance that UAV i must travel to perform task j .

A UAV is capable of executing only one task at a time, and its speed remains constant throughout the task.

4.2. Communication Model

There are two basic ways to deal with poor communication. The first method involves the negotiating agent waiting until it receives messages from all agents in each round, akin to the TCP message transfer protocol. Alternatively, in the second method, the negotiating agent operates under the assumption that each round has a fixed duration. If it fails to receive an offer from an agent within this time frame, it interprets this as a communication issue [51]. Given the uncertainty regarding whether agents in UAV swarms are within communication range, the second method was favored within the study's scope.

During the simulations conducted for the study, the Bernoulli communication model was utilized, assuming that agents communicated within a fully connected graph structure. The fully connected graph structure is illustrated in Figure 1.

The Bernoulli Communication Model represents a framework for communication wherein information is transmitted through random variables. Developed to assess information transmission efficacy, this model derives its name from the mathematician Jakob Bernoulli.

In this framework, the information to be transmitted, denoted by the random variable X , adheres to a specific probability distribution. Typically, this distribution follows a Bernoulli distribution, often expressed as $P(X = 1) = p$ and $P(X = 0) = 1-p$, where p signifies the probability of successfully transmitting a bit. The fidelity of information transmission hinges upon the characteristics and distribution of this random variable, with the communication channel's effectiveness contingent upon the state of said variable.

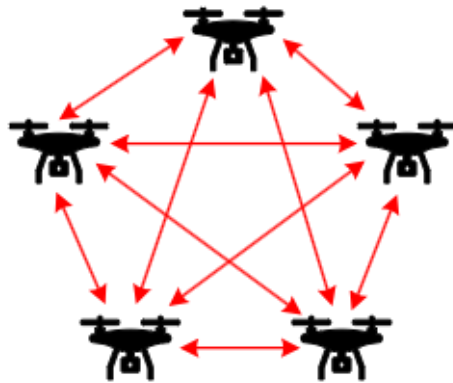


Figure 1. Fully Connected Communication Graph

The Bernoulli Communication Model finds application in scenarios involving stochastic communication channels, offering a simplified depiction that overlooks real-world factors such as range effects, directional influences, and bandwidth limitations. Nevertheless, it proves beneficial in contexts where communicating entities are proximal within their communication range. Its straightforward structure renders it a valuable analytical instrument, particularly in error analysis and performance evaluations across diverse communication systems, including wireless and radio communication networks.

To evaluate the communication channel quality parameter, first, the path loss of the channel was calculated by measuring the distance between agents. For path loss, it has been shown that the two-ray path loss model gives superior approximations of the aerial vehicle communication channel modeling in comparison to the free-space path loss model [58]. The two-ray path loss model calculates the path loss between a transmitter and a receiver considering both the direct line-of-sight (LOS) path and a reflected path. The equation for the two-ray path loss model is as Equation 11.

$$P_r = P_t - 20 \log \left(\frac{\lambda}{4\pi d} * 2 \sin \left(\frac{2\pi h_r h_t}{\lambda d} \right) \right) + G_t + G_r + AWGN \quad (11)$$

where P_r is the receiving power, P_t is the output power of the transmitter, λ is the wavelength of the transmitted signal, d is the distance parallel to the ground between the aircraft, h_t is the ground clearance of the transmitting aircraft, h_r is the receiving aircraft's ground clearance, G_t is the gain of the transmitting antenna, G_r refers to the gain of the receiving antenna. In channel modeling, it is assumed that there is no interference except thermal noise to obtain more repeatable results during simulation studies. The communication channel parameters are chosen as follows; thermal noise (W/Hz)= 4.143×10^{-21} , N_0 =-174 dBm, bandwidth=47 MHz, output power=10 W, antenna gain=1 dB.

5. Proposed Approach

One of the main contributions of the research lies in presenting an algorithm featuring a two-stage auction process for task assignment within heterogeneous unmanned aerial vehicle swarms. This section elucidates the proposed task assignment algorithm.

5.1. Utility Function Design

Within auction-based task assignment algorithms, the utility function serves as a crucial metric for prioritizing and assigning tasks based on specific criteria. Typically, a benefit value is computed for each task, and aggregating these values guides the assignment process, ensuring optimal task allocations. This section elaborates on the design of cost and utility functions incorporated within the proposed algorithm.

The cost function is utilized to compute the task expenses of the UAV, which include two elements: balance cost and distance cost. The cost of task j for agent i , denoted as C_{ij} , is computed using Equation 12, where w_1 and w_2 are the coefficients for distance and balance costs, respectively.

$$C_{ij} = w_1 C_{ij}^D + w_2 C_{ij}^B \quad (12)$$

The balance cost (C_{ij}^B) for UAV i performing task j is determined using Equation 13, which takes into account the task load of the UAV.

$$C_{ij}^B = \frac{|b_i|}{MT_i} \quad (13)$$

where $|b_i|$ represents the number of elements in the task list of UAV i and MT_i denotes the maximum number of tasks that UAV i can add to its task list.

The distance cost (C_{ij}^D) of UAV i for task j is determined by Equation 14, which considers the distance of the UAV to the task location.

$$C_{ij}^D = \frac{d_{ij}}{MFD_i} \quad (14)$$

where MFD_i represents the distance of UAV i to the farthest task, while d_{ij} denotes the distance between UAV i and task j .

The cost matrix for agent i is expressed as depicted in Equation 15.

$$C_i = [C_{i1} \quad C_{i2} \quad \dots \quad C_{iM}] \quad (15)$$

Employing the defined cost function, the local utility value is computed according to Equation 16, where R_{ij} represents the utility value for agent i of task j .

$$R_{ij} = V_j e^{-\tau*(tc_{start}-(tp_{start}+tp_{duration}))} - C_{ij} \quad (16)$$

where V_j represents the initial reward value of task j , while τ denotes the time penalty coefficient. Additionally, tc_{start} refers to the earliest time at which agent i can commence task j , tp_{start} represents the time required for the agent to initiate the last task on the agent's path, and $tp_{duration}$ signifies the duration of the last task on the agent's path.

Two different reward matrices are utilized in the algorithm: normal (R_{ij}^N) and synergy (R_{ij}^S). The normal reward matrix is computed using the cost matrix acquired by the agent in the respective iteration for tasks that have not yet been assigned.

In calculating the synergy reward matrix, the initial step involves the computation of a new synergy cost matrix, which is based on the assumption that the agent successfully completes the task with the highest reward in the normal reward matrix. Subsequently, the synergy reward matrix is determined based on this cost matrix using Equation (16). The utilization of the synergy reward matrix aims to capture the synergy between tasks, particularly their proximity to one another. This approach facilitates the agent in receiving their next assignment with maximum income by leveraging task synergies. If there are only two tasks remaining unassigned, the synergy reward matrix is not calculated.

5.2. Auction Process

The auction process comprises two distinct phases: the pre-auction phase and the synergy-auction phase. In the proposed two-stage auction process, agents strive to incorporate the second most beneficial task into their path alongside the primary task, rather than adopting a purely greedy approach of solely pursuing the most beneficial task at each iteration. This approach aims not only to minimize the overall system cost but also to reduce the message size required to address the problem effectively.

Pre-auction phase: in this phase, the UAV identifies the highest value (f_i) task (j_i) in the normal reward matrix and computes its bid (δ_i) for f_i . Subsequently, the drone broadcasts an auction message for task j_i . The calculation of δ_i is determined by Equation 17, where s_i represents the utility value of the agent's second-best task.

Once all agent's auction messages are received, the UAV proceeds to check whether other UAVs have initiated auctions for the same task. Should multiple UAVs commence an auction for a task, the UAV boasting the highest reward value (MaxReward) wins the task. In instances of equal reward values, the UAV with the lower ID wins the task. This process is termed the pre-auction phase. Subsequently, the winning drone communicates its success to other swarm agents via a Pre-Auction Result message.

$$\begin{aligned} \delta_i &= f_i - s_i \\ j_i &= \arg \max_{j=1, \dots, n} \{R_{ij}^N\} \\ f_i &= \max_j \{R_{ij}^N\} \\ s_i &= \max_{j \neq j_i} \{R_{ij}^N\} \end{aligned} \quad (17)$$

Furthermore, the winner drone in the pre-auction phase has the opportunity to bid for the synergy task in the subsequent stage. Conversely, the drone that loses the preliminary auction is ineligible to bid for the synergy task. The outlined procedure for the preliminary auction phase is outlined in Algorithm 1.

Algorithm 1. Pre-Auction Phase

1	Input: Unassigned task
2	Output: Pre-Auction Mes., Pre-Auction Result Mes.
3	if $length(P_i) < MT_i$ then
4	<i>auction</i> = 1
5	Calculate : R_i^N
6	$j_i = \text{argmax}_j(R_{ij}^N)$
7	$f_i = \max_j(R_{ij}^N)$
8	$s_i = \max_{j \neq j_i}(R_{ij}^N)$
9	$\delta_i = f_i - s_i$
10	<i>MaxTask</i> = j_i
11	<i>MaxReward</i> = f_i
12	<i>MaxTaskInc</i> = δ_i
13	SEND (<i>PreAuction Message</i>)
14	else
15	<i>auction</i> = 0
16	end if
17	RECEIVE (<i>PreAuction Messages</i>)
18	if $MaxTask_i = MaxTask_k, \forall k \in T$ then
19	Find : <i>WinnerAgent</i>
20	else
21	Agent wins the <i>PreAuction Phase</i>
22	end if
23	SEND(<i>PreAuction Result</i>)

Synergy-Auction phase: in this phase, agents place bids for synergy tasks. Should an agent secure a task during the pre-auction phase, bids for synergy tasks are extended during the synergy-auction phase. Upon receiving a Pre-Auction Result message from another UAV pertaining to the task with the highest value in the synergy revenue matrix, the drone subsequently submits its bid for the corresponding task to the winning drone. The computation of the bid for the synergy task is based on the synergy revenue matrix, as detailed in Equation 18.

$$\begin{aligned}
 \gamma_i &= g_i - h_i \\
 j_i &= \arg \max_{j=1, \dots, n} \{R_{ij}^S\} \\
 g_i &= \max_j \{R_{ij}^S\} \\
 h_i &= \max_{j \neq j_i} \{R_{ij}^S\}
 \end{aligned} \tag{18}$$

Upon receiving offer messages for synergy tasks, the agents winning the respective tasks are identified for the initial iteration. If $\delta_i > \gamma_i$ for the relevant task, the agent who broadcasts the auction message wins the task. Conversely, if a synergy offer results in $\gamma_i > \delta_i$, the agent making the offer wins the task. Should the agent who initiated an auction for the most valuable task receive no bids, they emerge as the winner.

Following the culmination of the synergy-auction phase, all agents disclose the tasks they have secured through the dissemination of the Synergy-Auction Result message. For agents unable to secure a task, the task ID is set to 0, and the message is broadcasted accordingly. Upon receiving this message, agents proceed to eliminate the corresponding task from the unassigned task list. This protocol serves to mitigate potential disagreements between agents and ensures an accurate record of unassigned tasks is maintained. The comprehensive procedure for the synergy-auction phase is delineated in Algorithm 2.

6. Simulation Experiments

We conducted three sets of simulations to assess the efficacy of our proposed task assignment algorithm, the Central Hungarian algorithm, and CBBA. Initially, we utilized our algorithm to assign 10 tasks to 5 aircraft, presuming no communication issues among the UAVs, to showcase its adeptness in heterogeneous task allocation. Following this, we employed Monte Carlo simulations across 100 scenarios without communication constraints, comparing our algorithm's performance with centralized and CBBA algorithms of regarding total task cost and message size. This comparison aimed to scrutinize assignment efficiencies in unfettered communication scenarios. Lastly, we analyzed scenarios where message transmission between aircraft was partial, comparing outcomes with Central Hungarian and CBBA algorithms, particularly focusing on unassigned and conflicting tasks, to evaluate assignment capabilities under communication limitations.

In the simulations, UAVs and assignments were distributed randomly across a 100x100 km. The UAVs move at a speed of 80 km/h and fly at an altitude of 1000 m. The UAV group consists of two types: reconnaissance and payload UAVs. Tasks are categorized into intelligence gathering (IG) and delivery (DL), each with specific time windows for execution; otherwise,

they cannot be performed. Reconnaissance UAVs are dedicated to intelligence gathering tasks, while payload drones are equipped for deliveries. In simulation and field tests, certain constants are fixed: $w_1=0.7$, $w_2=0.3$, and $\tau=0.1$. Users have the flexibility to adjust these constants based on task demands and UAV swarm characteristics. MATLAB software was utilized for simulations, executed on a personal computer featuring a 2.53 GHz Intel processor, 12GB of memory, and running the Windows 10 operating system.

Algorithm 2. Synergy-Auction Phase

```

1  Input: Pre-Auction Result, Unassigned task list
2  Output: Synergy-Auction Result
3  if  $length(UnassignedTask) < 2$  then
4       $auction = 1$ 
5      Calculate :  $R_i^S$ 
6       $j_i = argmax_j(R_{ij}^S)$ 
7       $g_i = max_j(R_{ij}^S)$ 
8       $h_i = max_{j \neq j_i}(R_{ij}^S)$ 
9       $\gamma_i = g_i - h_i$ 
10     SynergyTask =  $g_i$ 
11     SynergyTaskInc =  $\gamma_i$ 
12     SEND(Bidding Message)
13 else
14      $auction = 0$ 
15 end if
16 Find : WinnerAgent
    
```

6.1. Heterogeneous UAV Swarm Task Assignment

Utilizing the algorithm introduced in this section, we assigned the 10 tasks outlined in Table 2 to the 5 aircraft specified in Table 3. The resulting aircraft routes obtained from the assignment are as follows: for agent 1, the route is 5, 2, and 1; for agent 2, it is 4 and 3; for agent 3, it is 9, 7, and 10; for agent 4, it is 6; and for agent 5, the route is 8. These routes are visually depicted in Figure 2.

As depicted in Figure 2, the proposed algorithm has successfully completed the task assignment process for the specified scenario without any conflicts. In the figure, the x and y axes correspond to the longitudinal and lateral positions of the aircraft, respectively, while the z axis represents the time constraints associated with the tasks.

Table 2. Task Parameters

ID	The coordinates of the task	Type	Time window of task validity	t_{td}
1	[117.92, 382.28]	IG	[69.04, 74.04]	5
2	[257.68, 479.90]	IG	[47.79, 52.79]	5
3	[527.33, 519.80]	IG	[99.35, 104.35]	5
4	[769.09, 615.00]	IG	[83.26, 88.26]	5
5	[167.90, 430.83]	IG	[10.22, 15.22]	15
6	[267.62, 384.62]	PL	[67.12, 82.12]	15
7	[678.08, 463.60]	PL	[20.89, 35.89]	15
8	[291.03, 238.99]	PL	[49.70, 64.70]	15
9	[464.14, 44.05]	PL	[15.12, 30.12]	15
10	[549.71, 632.80]	PL	[45.50, 60.50]	15

Table 3. UAV Parameters

Parameter	Value
The number of UAV	5
Type of UAV 1	Reconnaissance
Type of UAV 2	Reconnaissance
Type of UAV 3	Payload
Type of UAV 5	Payload
Type of UAV 5	Payload
Initial position of UAV 1	[365.07, 517.17, 0]
Initial position of UAV 2	[598.60, 726.87, 0]
Initial position of UAV 3	[707.61, 58.52, 0]
Initial position of UAV 4	[191.20, 740.49, 0]
Initial position of UAV 5	[719.82, 404.64, 0]

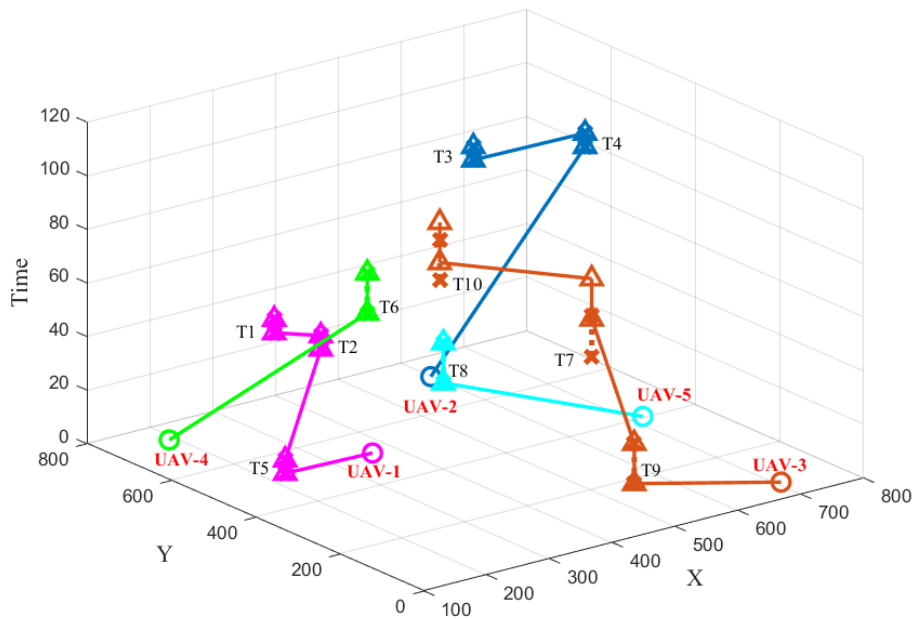


Figure 2. Agent Routes

6.2. Comparison with Similar Algorithms

In the subsequent phase of simulations, the performances of the proposed algorithm, as well as the Central and CBBA algorithms, were evaluated through Monte Carlo simulations. The objective was to allocate 20 tasks to 20 aircraft within a fully connected network structure, assuming no communication issues. Comparative analysis among the algorithms encompassed total mission cost, total message size (measured in bits), and communication channel signal-to-noise ratio (SNR) parameters. Monte Carlo simulations were executed across 100 different scenarios, with aircraft and mission locations being randomly generated for each scenario.

The total task cost cumulative probability density function (CDF) graph is provided in Figure 3. Total mission cost represents the collective distance traveled by each aircraft. The total task cost achieved by the proposed algorithm, due to its two-stage auction structure, outperforms CBBA and maintains a satisfactory proximity to the centralized solution.

The CDF graph in Figure 4 illustrates the total number of bits, signifying the overall size of messages transmitted during task assignment. This comparison enables us to assess the efficiency of various methods in terms of resource utilization. By quantifying the amount of data exchanged throughout the task assignment process, we can gain insights into the feasibility and scalability of implementing the proposed algorithm in real-world communication networks with limited bandwidth. As depicted in the figure, the proposed algorithm demonstrates the capability to address the problem with significantly smaller message sizes, attributed to the developed cost function, auction structure, and message format enhancements.

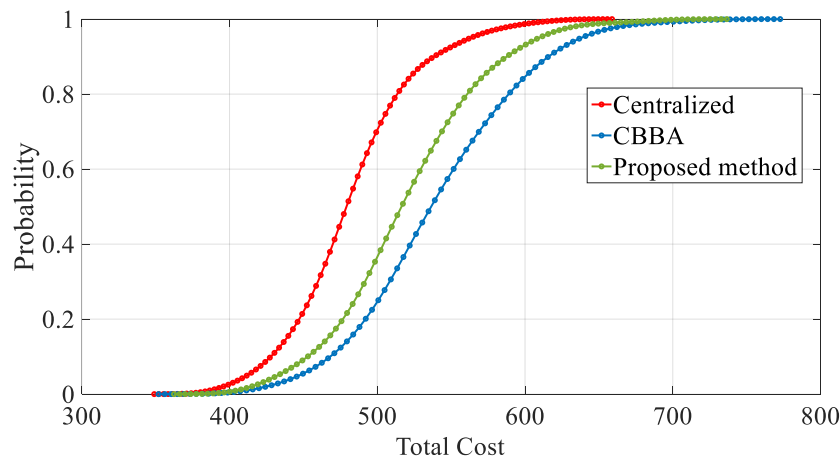


Figure 3. CDF of the Total Mission Cost

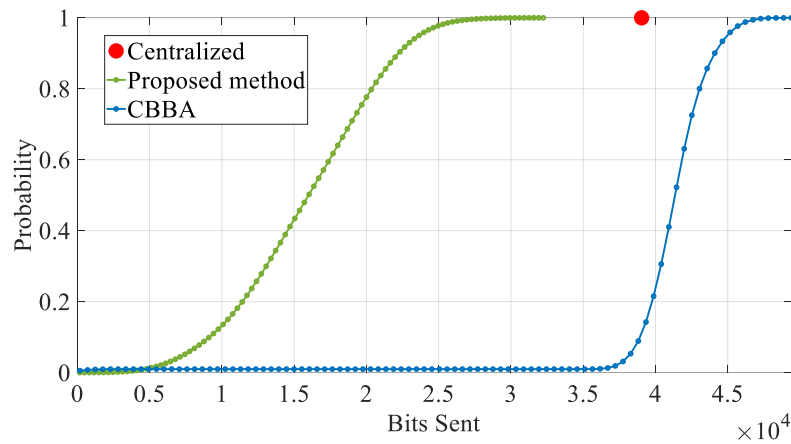


Figure 4. CDF Graph of Total Bits Sent

The comparison of average Signal-to-Noise Ratio (SNR) values of the communication channel is illustrated in Figure 5. Across all three algorithms, a consistent trend emerges where SNR values exceed 0 dB in 90% of the simulation runs. Notably, both CBBA and the centralized approach demonstrate SNR values surpassing 5 dB with nearly 0.8 probability. Conversely, Harmony achieves similar SNR values with slightly higher probabilities, around 0.9, thus creating a probability gap of approximately 0.1 to 0.2 for the same SNR values. Furthermore, in direct comparison to the other algorithms, Harmony showcases SNR values 2 dB higher for equivalent probabilities, underscoring its superior performance in maintaining reliable communication channels under various conditions.

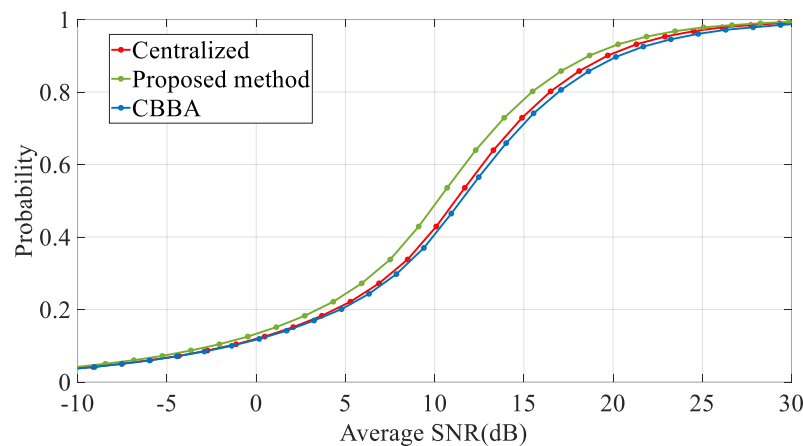


Figure 5. CDF Graph of the Average SNR

6.3. Comparison with Similar Algorithms Under Limited Communication

In this section, the proposed algorithm is evaluated against centralized and CBBA algorithms under scenarios where some messages may not be transmitted, employing the Bernoulli communication model. Simulations were conducted with varying probabilities of message non-transmission (0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%), and the algorithms were compared based on the number of unassigned tasks and conflicting tasks parameters.

Figure 6 displays the outcomes of simulating the allocation of 10 tasks among 5 agents under conditions of limited communication. In centralized assignment, where tasks are delegated by a single agent, conflict task assignment remains absent despite potential increases in communication errors. However, as the number of agents unable to communicate with the central agent rises, the count of unassigned tasks also escalates. In contrast, in CBBA, there is no notable change in the number of unassigned tasks even with heightened probabilities of communication error. Moreover, due to its consensus-based assignment approach, the fluctuation in the number of conflict tasks remains minimal even with elevated error rates. Although the proposed algorithm maintains a steady count of unassigned tasks within acceptable limits despite heightened communication error rates, its absence of a consensus stage results in an increase in conflict task assignments as the probability of communication errors rises.

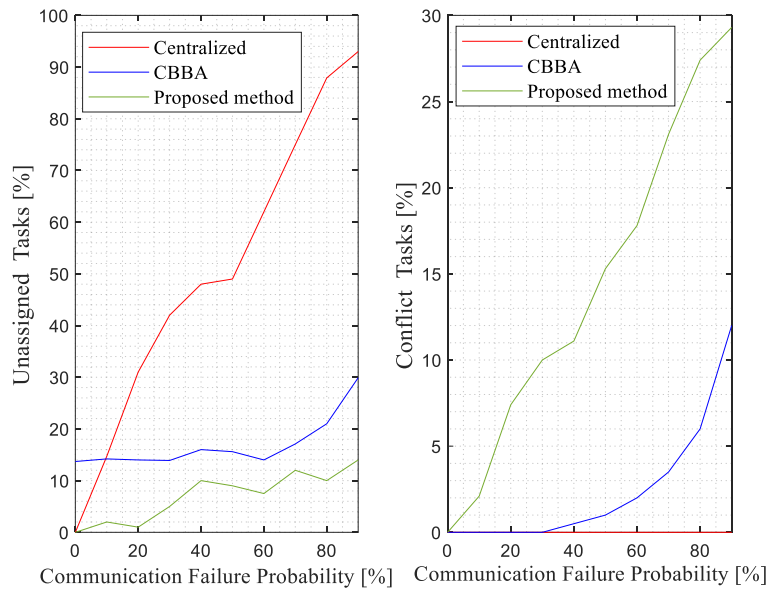


Figure 6. Assigning 10 Tasks to 5 Agents Under Limited Communication

Figure 7 presents the simulation results of distributing 100 tasks among 5 agents while varying the communication error rate in a similar manner as in Figure 6.

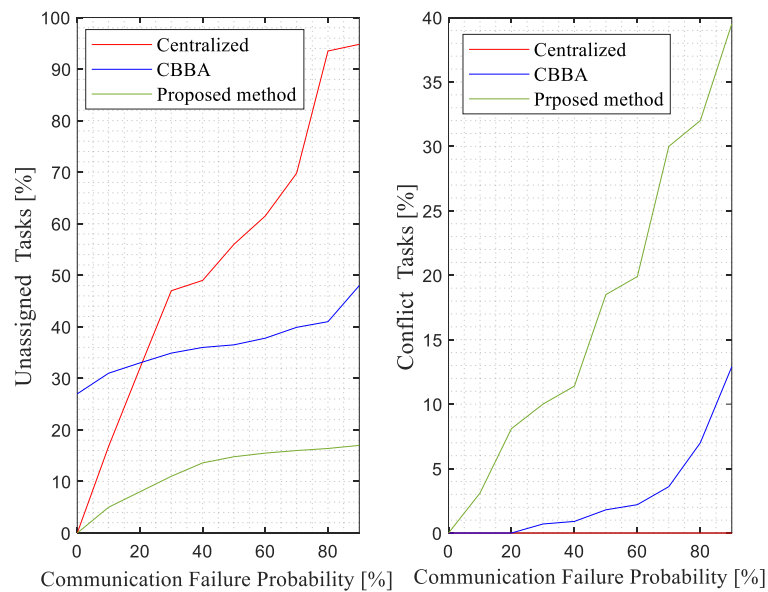


Figure 7. Assigning 100 Tasks to 5 Agents Under Limited Communication

The outcomes derived from three sets of simulations are as follows:

In situations devoid of communication issues, the centralized algorithm adeptly tackles the assignment problem with the least overall cost. However, due to the necessity for all agents to relay their cost matrices to the central agent, the collective message size expands significantly. Instances where communication obstacles arise and certain messages fail to transmit highlight a vulnerability of the centralized algorithm to single-point errors. Consequently, under such circumstances, while conflict assignments are absent due to assignments originating from a singular point, the count of unassigned tasks escalates.

CBBA, in contrast, often incurs higher task assignment costs compared to both the centralized and proposed algorithms under ideal communication conditions. Additionally, CBBA may encounter situations where it fails to assign certain tasks altogether. This is compounded by the necessity of transmitting three large datasets among agents during the consensus phase, resulting in a substantial total bit usage, which is especially problematic in bandwidth-constrained environments. Notably, CBBA's performance and system-wide efficiency are significantly impacted by bandwidth availability. However, in scenarios where communication issues arise and messages fail to transmit, CBBA demonstrates resilience by executing the assignment

process with fewer unassignable and conflicting tasks, albeit requiring extended iterations. Thus, CBBA appears more suitable for environments without bandwidth constraints but with potential message loss risks.

The proposed algorithm excels in resolving the assignment problem with lower costs compared to CBBA, particularly in scenarios without communication hindrances. Its performance is commendably close to that of the centralized approach. Moreover, owing to its two-stage auction mechanism and message structure, it achieves problem resolution with minimal bit usage, offering a distinct advantage in bandwidth-constrained environments. In environments where all messages can be reliably transmitted, the proposed algorithm demonstrates superior performance. However, when communication issues arise and messages fail to transmit, the algorithm faces challenges. Even if the proposed algorithm manages to assign all tasks under such circumstances, conflicts may arise due to the absence of a consensus stage. As the probability of communication errors increases, the incidence of conflict assignments also rises.

7. Conclusion

In this research endeavor, a novel approach is introduced for the distributed task assignment within UAV swarms. The efficacy of the proposed algorithm is scrutinized across varying conditions, including scenarios of poor communication. The algorithm's core objective revolves around curtailing message sizes exchanged between agents, thereby minimizing the overall bit consumption during assignment through a two-stage auction process. This process involves drones bidding on tasks while considering synergistic tasks, steering clear of a purely opportunistic approach of selecting tasks solely based on immediate benefits.

To gauge the performance of the proposed algorithm, Monte Carlo simulations were conducted under differing communication scenarios, ranging from seamless communication to various levels of message loss. These results were juxtaposed against those obtained from the central Hungarian algorithm and CBBA.

The findings underscore the superiority of the proposed algorithm in resolving assignment quandaries with reduced costs compared to CBBA, particularly evident in communication-unfettered environments. Notably, its performance closely rivals that of the centralized approach. Furthermore, owing to its innovative two-stage auction mechanism and streamlined message structure, the algorithm achieves problem resolution with minimal bit utilization, offering a distinct advantage in bandwidth-constrained settings. In environments conducive to reliable message transmission, the proposed algorithm exhibits superior performance.

However, challenges emerge when communication issues disrupt message transmission, potentially leading to conflicts even if all tasks are successfully assigned. This underscores the necessity for a consensus stage, particularly in scenarios with heightened probabilities of communication errors.

Future investigations will incorporate a consensus stage to bolster the algorithm's performance under limited communication conditions. Following this augmentation, the algorithm's resilience in communication-constrained environments will be evaluated across diverse communication network topologies.

References

- [1] R. Bouffanais, "Design and control of swarm dynamics," Springer, Singapore, 2016.
- [2] H. Geer, C. Bolkcom, "Unmanned aerial vehicles: Background and issues for Congress," CRS Report for Congress, CRS Report No. RL31872, 2003.
- [3] K. P. Valavanis, G.J. Vachtsevanos, "Handbook of Unmanned Aerial Vehicles-5," New York, NY, USA: Springer, 2014.
- [4] M. R. Brust, G. Danoy, D.H. Stolfi, P. Bouvry, "Swarm-based counter UAV defense system," *Discover Internet of Things*, vol.1 no.1, pp.1-19, 2016.
- [5] E. Şahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," In: *Lecture Notes in Computer Science*, vol 3342. Springer, Berlin, Heidelberg 2005.
- [6] H. David, "Indian Army Shows Off Drone Swarm of Mass Destruction," *Forbes*, January 19, 2021.
- [7] A. Bürkle, F. Segor, M. Kollmann, "Towards Autonomous Micro UAV Swarms," *Journal of Intelligent & Robotic Systems*, vol. 61, pp. 339–353, 2010.
- [8] F. Denis, "Russia Develops Molniya Swarming UAV Concept," *Defence Weekly*, March, 2021.
- [9] P. Akshara, "Expanding Focus: Rotary-Wing Swarming UAVs from China's Ziyang," *International Defence Review*, September 23, 2021.
- [10] R. M. Zlot, "An auction-based approach to complex task allocation for multirobot teams," PhD Dissertation, Carnegie Mellon University, 2006.
- [11] C. Tovey, M. Lagoudakis, S. Jain, "The generation of bidding rules for auction-based robot coordination," *Multi-Robot Systems*, vol.3, no.1, pp.3-14, 2005.
- [12] M. Dias, A. Stentz, "A free market architecture for distributed control of a multirobot system," in: *Conference on Intelligent Autonomous*, 2000.
- [13] Z. Kallenborn, "Infoswarms: drone swarms and information warfare," *The US Army War College Quarterly: Parameters*, vol.52, no.2, pp.87–102, 2022.

- [14] B. T. Clough, "UAV swarming? So what are those swarms, what are the implications, and how do we handle them?" in *AUVSI Unmanned System Conf.*, Orlando, FL, 2002.
- [15] R. C. Arkin, M. Egerstedt, "Temporal heterogeneity and the value of slowness in robotic systems," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1000–1005, 2015.
- [16] D. Frelinger, J. Kvitky, W. Stanley, "Proliferated autonomous weapons," Rand Corp., Santa Monica, CA, Tech. Rep., 1997.
- [17] N. H. Motlagh, T. Taleb, O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.
- [18] Drone incidents all over the world. [Online]. Available: <https://www.dedrone.com/resources/incidents/all>
- [19] S. Edwards, "Swarming on the Battlefield: Past, Present, and Future". Santa Monica, Rand Corp., 2000.
- [20] B. P. Gerkey, M. J. Mataric, "A formal analysis and taxonomy of task allocation in multirobot systems," *International Journal of Robotics Research*, vol.23, no.9, pp.939–954, 2004.
- [21] X. Chen, T. Tang, "Dynamic task allocation method for multiple UAVs in uncertain environment," *Firepower and Command Control*, vol. 38, no. 1, pp. 45–49, 2013.
- [22] X.L. Zhao, K.W. Zhang, Z.Z. Li, "Research on dynamic reconnaissance resource allocation of multiple UAVs," *Electronics Optics and Control*, vol. 27, no. 6, pp. 11–15, 2020.
- [23] T. Shima, S. J. Rasmussen, A. G. Sparks, K. M. Passino, "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms," *Computers & Operations Research*, vol. 33, no. 11, pp. 3252–3269, 2006.
- [24] F. Su, Y. Chen, L. C. Shen, "UAV cooperative multi task assignment based on ant colony algorithm," *Acta Aeronautica Sinica*, vol. 29, no. 1, pp. 184–191, 2008.
- [25] Z. Liu, W. Li, J.C. Ren, "Modeling and solving method of multi base and multiUCAV task allocation," *Journal of Southeast University*, vol. 49, no. 1, pp. 91–96, 2019.
- [26] D.M. Gordon, "The organization of work in social insect colonies," *Nature*, vol. 380 no.6570, pp.121–124, 1996.
- [27] I. Karsai, T. Schmickl, "Regulation of task partitioning by a "common stomach": a model of nest construction in social wasps," *Behavioral Ecology*, vol.22, no.4, pp.819–830, 2011.
- [28] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics*, vol.2, no.1, pp.83–97, 1955.
- [29] T. Lemaire, R. Alami, S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *IEEE International Conference on Robotics and Automation*, 2004.
- [30] M. Hussain, B. Kimiaghalam, A. Homaifar, "An evolutionary approach to capacitated resource distribution by a multiple-agent team," in *Genetic and Evolutionary Computation Conference*. 2003.
- [31] D.P. Bertsekas, "A distributed algorithm for the assignment problem," *Annals of Operations Research*, vol.14, no.1, pp.105–123, 1979.
- [32] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, "Auction-based multi-robot routing," in *Robotics: Science and Systems*, 2005.
- [33] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, H. Younes, "Coordination for multi-robot exploration and mapping," *American Association for Artificial Intelligence*, pp. 852–858, 2000.
- [34] I. Rekleitis, A. P. New, E. S. Rankin, H. Choset, "Efficient boustrophedon multi-robot coverage: An algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol.52, no.2, pp.109–142, 2008.
- [35] D. Vail, M. Veloso, "Dynamic multi-robot coordination," *Multi-Robot Systems: From Swarms to Intelligent Automata*, vol. 2, pp. 87–98, 2003.
- [36] R. C. Cavalcante, T. Noronha, L. Chaimowicz, "Improving combinatorial auctions for multi-robot exploration," in *16th international conference on advanced robotics*, 2013.
- [37] E. Schneider, O. Balas, A. T. Ozgelen, E. I. Sklar, S. Parsons, "Evaluating auction-based task allocation in multi-robot teams" in *AAMAS workshop on autonomous robots and multirobot systems*, 2014.
- [38] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, A. J. Kleywegt, "Simple auctions with performance guarantees for multirobot task allocation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sendai, Japan, pp. 698–705, 2004.
- [39] S. Yan, F. Pan, D. Zhang, "Research on task reassignment method of heterogeneous uav in dynamic environment," in *6th International Conference on Robotics and Automation Sciences*, 2022.
- [40] B. Lopez, B. Innocenti, D. Busquets, "A Multiagent System for Coordinating Ambulances for Emergency Medical Services," *IEEE Intelligent Systems*, vol.23, no.5, pp.50-57, 2008.
- [41] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Grin, A. Kleywegt, "Robot exploration with combinatorial auctions," in *International Conference on Intelligent Robots and Systems*, 2003.
- [42] A. Prasad, S. Sundaram, H. L. Choi, "Min-max tours for task allocation to heterogeneous agents," in *Proc. IEEE Conf. Decis. Control (CDC)*, Miami, FL, USA, 2018.
- [43] E. P. Freitas, M. Basso, A. A. S. Silva, "A distributed task allocation protocol for cooperative multi uav search and rescue systems," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021.
- [44] S. Trigui, A. Kouba, O. Cheikhrouhou, "A distributed market based algorithm for the multi robot assignment problem," *Procedia Computer Science*, vol.32, no.1, pp.1108–1114, 2014.

- [45] S. Raja, G. Habibi, J. P. How, “Communication-Aware Consensus-Based Decentralized Task Allocation in Communication Constrained Environments”, *IEEE Access*, vol.10, pp. 19753 – 19767, 2021.
- [46] S. A. S. Baron, “Dynamic Task Allocation and Coordination in Cooperative Multi-Agent Environments,” Doctoral Dissertation, University of Girona, Spain, 2010.
- [47] J. Bellingham, M. Tillerson, A. Richards, J.P. How, “Multi-task allocation and path planning for cooperative UAVs,” *Cooperative Systems*, vol 1., pp. 23–41, 2003.
- [48] G. A. Korsah, A. Stentz, M.B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.
- [49] J. Jackson, M. Faied, P. Kabamba, A. Girard, “Communication-Constrained Distributed Task Assignment,” in *IEEE Conference on Decision and Control and European Control Conference*, 2011.
- [50] D. Castanon, C. Wu, “Distributed algorithms for dynamic reassignment,” in *Proc. IEEE Conf. Decis. Control*, pp. 13–18, 2003.
- [51] M. Otte, M. J. Kuhlman, D. Sofge, “Auctions for multi-robot task allocation in communication limited environments,” *Autonomous Robots*, vol.44, pp. 547–584, 2020.
- [52] H. L. Choi, L. Brunet L, “Consensus based decentralized auctions for robust task allocation,” *IEEE Transactions on Robotics*, vol.25, no.4, pp. 912–926, 2009.
- [53] X. Fu, J. Pan, X. Gao, B. Li, J. Chen, K. Zhang, “Task Allocation Method for Multi-UAV Teams with Limited Communication Bandwidth,” in *15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018.
- [54] B. Zhu, L. Xie, D. Han, X. Meng, R. Teo, “A survey on recent progress in control of swarm systems,” *Science China Information Sciences*, vol.60, no.7, 2017.
- [55] S. J. Chung, A.vA. Paranjape, P. Dames, S. Shen, V. Kumar, “A survey on aerial swarm robotics,” *IEEE Transactions on Robotics*, vol.34, no.4, pp.837–855, 2018.
- [56] B. Balazs, G. Vasarhelyi, “Coordinated dense aerial traffic with self-driving drones,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6365–6372, 2018.
- [57] J. A. Fax, R. M. Murray, “Information flow and cooperative control of vehicle formations,” *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2004.
- [58] J. Supramongkonset, S. Duangsuwan, S. Promwong, “A Study of A2A Channel Modeling for Small UAV-Enabled Wireless Communication,” in *IEEE International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 2021.

Author(s) Contributions

All three authors contributed equally to the study.

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate™.

Network Forensics Analysis of Cyber Attacks Carried Out Over Wireless Networks Using Machine Learning Methods

İmran Kaçan¹, Batuhan Gül¹, Fatih Ertam¹

¹Firat University, Faculty of Technology, Digital Forensics Engineering, Elazığ, Türkiye

Corresponding author:

Batuhan Gül, Fırat University, Faculty of
Technology, Digital Forensics Engineering,
Elazığ, Türkiye
b.gul@firat.edu.tr

Article History:
Received: 01.04.2024
Accepted: 04.06.2024
Published Online: 23.08.2024

ABSTRACT

As technology advances, the frequency of attacks targeting technological devices has surged. This rise in cyber threats poses a constant risk to the devices we rely on. Any device connected to a network becomes vulnerable to exploitation by attackers. Given the extensive interconnectedness of devices in network environments, this research endeavors to address this pressing issue. The aim of this study is to analyze and classify network traffic generated during potential cyber attacks using various classification algorithms. By subjecting a simulated environment to different cyber attack scenarios, we extract the distinctive features of network packets generated during these attacks. Subsequently, we employ widely used classification algorithms to train and analyze the obtained data. For the comparison of models, more than 7000 attack data instances were employed. At the conclusion of the comparison, the Gradient Boosting algorithm achieved the highest accuracy value, reaching 91%, whereas the Naive Bayes algorithm obtained the lowest accuracy, reaching 74%.

Keywords: Network forensics, Cyber security, Machine learning

1. Introduction

Communication between at least two computers is called computer networks [1]. In the interconnected realm of the cyber world, computer networks can be inherently vulnerable to various types of attacks. The attack examples are such as man-in-the-middle attacks, denial-of-service attacks, distributed denial-of-service attacks, and malicious software injection [2].

The significant increase in internet usage has accompanied the advancement of technology. Numerous devices in our surroundings are now connected to networks. Furthermore, the rise in technology has led to an increase in digital crimes.

The widespread adoption of the evolving technological infrastructure implies that these systems are exposed to significant risks. While the increase in technological structures facilitates human life, it also allows threat elements access to various sources and the potential exploitation of system vulnerabilities [3]. Consequently, the emergence of individuals intending to inflict harm on these systems, coinciding with the utilization of technological devices and network technologies, has given rise to a category of crimes known as cybercrimes.

In the realm of cybercrimes, data recorded in electronic/magnetic fields is referred to as digital evidence. Various types of digital evidence exist, such as photos, videos, server log files, web history, data files and registration logs. The majority of these data are transmitted over the network. One crucial form of analysis is network analysis.

The general purpose of the programs/tools used for network analysis is to listen to network traffic, capturing incoming and outgoing packets during the listening process for network analysis. A network consists of two or more devices such as computers, servers, and network devices, sharing resources like printers, engaging in file exchange, or permitting electronic communication. Additionally, it can be asserted that the most effective network security method involves managing access to the network [4]. In implementing these security measures, a thorough understanding of the user profiles connected to the network is imperative. Having command over user profiles facilitates the work of network administrators during the authorization processes. When authorization is tailored to the user profile, managing the network becomes more straightforward. It is imperative that each user does not have unrestricted access to every network. Access to network resources should be granted only to authorized users, preventing malicious activities by restricting unauthorized access to the network. Unused ports should be closed, and structures should be left open based on user needs. Authentication methods

must be activated. Additionally, understanding various attack types is crucial for defining security policies. Considering all these aspects, the significance of forensic network analysis becomes evident. Security measures in this field should be enhanced, and the number of educated individuals in the domain should be increased.

We propose obtaining the necessary data for network forensic analysis through the application of machine learning models. Previous studies have addressed analysis topics in wireless networks using machine learning.

The utilization of machine learning models has been proposed by Dhanya et al. [5] for the detection of cyber attacks targeting wireless networks. In the suggested methodology, a comparison of nine distinct machine learning algorithms has been conducted, and this comparative analysis has been executed on the UNSW-NB15 dataset. Among the compared machine learning models, the Decision Tree algorithm exhibited the highest performance, achieving a remarkable accuracy of 99.05% and a recall value of 99%. Conversely, the SVM algorithm demonstrated the least favorable performance, attaining an accuracy of 95.17% and a recall value of 93%. The authors suggested converting network data into images to improve the detection performance of the models.

Ahmad et al. [6] discussed machine learning methods in wireless sensor networks and the potential for detecting and classifying attacks on wireless networks using these methods. The authors have asserted the effectiveness of employing machine learning models in wireless sensor networks. Furthermore, they have postulated that the utilization of Software-Defined Networking (SDN) technology will enhance the performance of the machine learning model.

Mughaid et al. [7] proposed using machine learning models to detect attacks in 5G wireless networks. The authors developed a simulation software and obtained attack data with this simulation software. The OMNeT++ software has been utilized for simulation purposes, wherein a Dropping attack scenario was created to compare the performance of various machine learning models. In the conducted experiments, the Logistic Regression model achieved the highest accuracy rate at 95.7%, while the Naive Bayes model reached the lowest accuracy rate at 76.7%.

A survey study has been conducted by Waqas et al. [8] utilizing artificial intelligence and machine learning to classify cyber threats in wireless networks. In the article, the authors initially enumerate the cyber threats that necessitate attention, subsequently comparing and categorizing these threats for analysis. The second section of the article discusses potential defense mechanisms against the identified threats, utilizing machine learning methods and artificial intelligence models.

Briefly, the main contributions of this study can be stated as follows:

- Cyber attacks are being conducted on computer systems through wireless networks, and following the execution of these attacks, network traffic is recorded using Wireshark to construct our dataset.
- We analyze the obtained network packets and, through feature extraction, employ machine learning approaches to classify these network packets.
- In classifying data within our created dataset, the most successful model is the Gradient Boosting model, achieving an accuracy rate of 91%. Conversely, the model with the lowest accuracy rate is the Naive Bayes model.

In the first part of our study, we provide general information about cyber attacks on wireless networks and the damage these attacks can cause, and we compile previous studies on the use of machine learning techniques in wireless networks. In the second part, we detail the cyber attacks that can be made on wireless networks. In the third part of our study, we provide detailed information about machine learning classifiers and introduce our method. In the fourth section, we classify and compare results obtained from machine learning techniques. In the last part of our study, we include the conclusions and suggestions.

2. Cyber Threats on Wireless Networks

With the increasing importance of portability, the utilization of wireless networks has also seen a surge. Alongside portability, wireless access points are employed to expand networks [9]. Numerous users connect to these expanded networks. Wireless networks, providing convenient usage for a large number of users, are susceptible to cyber attacks. For these structures to be considered secure, they must fulfill conditions of authenticity, confidentiality, integrity, and usability [10].

Cyber attacks occur across different layers of the OSI model. The first layer of the OSI model, known as the Physical layer, facilitates the transfer of data packets between devices. This transfer is achieved through electrical or light signals. The other layers operate in a manner dependent on the Physical layer.

The next layer, the Data Link layer, separates bits from the Physical layer into packets when transferring from one device to another. This layer controls the physical addresses of devices to ensure the accurate delivery of data packets. The data transmission process occurs bit by bit.

The Network layer, which utilizes the IP protocol for communication, performs addressing and routing operations during the transmission of data.

In the Transport layer, data is segmented and reassembled at the destination. The Transport layer header is added to ensure the correct delivery of data.

The Session layer manages the opening, maintaining, and termination of necessary sessions between applications. To facilitate seamless data exchange, this layer ensures that the session remains open for an adequate duration during data transmission. Systems can initiate bidirectional communication when establishing a connection in this layer. Once the data transfer process is completed, the session is terminated to prevent unnecessary resource consumption.

The Presentation layer determines which protocols to use during the exchange of data packets and performs data transformation. Data is transformed into suitable formats for transfer to the Application layer.

The Application layer is the topmost layer visible to end-users. Users can provide data input in this layer, offering the clearest view for users. This layer includes various protocols that allow communication with applications such as email, instant messaging, and file transfer.

Table 1 demonstrates cyber attacks against OSI layers.

Table 1. Cyber Attacks Against OSI Layers

Layer	Attack Type
Physical Layer	Eavesdropping, Jamming, Side-Channel Attacks, Random Interference, Timing Attack
Data Link Layer	MAC Spoofing, Identify Theft, man in the Middle, network Injection, Mac Flooding
Network Layer	IP Spoofing, IP Hijacking, Smurf Attack, Sinkhole Attack
Transport Layer	TCP Flooding, UDP Flooding, TCP Guessing Attack
Application Layer	Malware Attack, SQL Injection, Cross-Site Scripting, FTP Bounce

In preparation for this study, a test environment was set up to obtain the necessary data, and six wireless network attacks were conducted. Deauthentication Attack, DoS, UDP Flood, ICMP Flood, SYN Flood and Man in The Middle attacks were performed using this test environment.

2.1. Deauthentication Attack

In deauthentication attacks, which fall under the category of second-layer attacks [11], the aim is to disconnect devices connected to a wireless network by sending numerous packets. This type of attack can be considered within the class of Service Denial of Service (DoS) attacks.

The structure of a deauthentication attack, which is designed to disrupt the access of devices connected to the network, is illustrated in Figure 1.

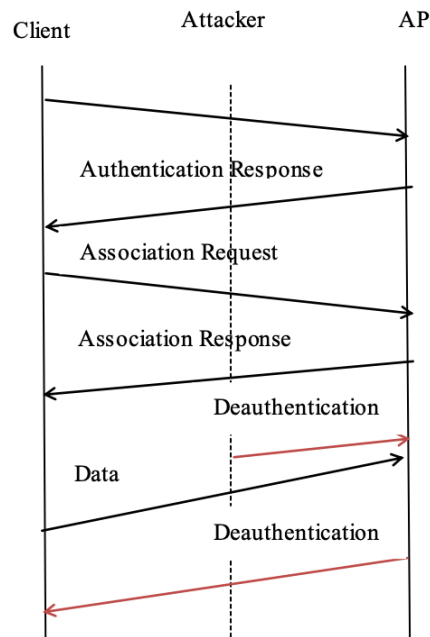


Figure 1. Deauthentication Attack [12]

Here's an overview of how a Deauthentication Attack works;

Frame Capture: The attacker puts a Wi-Fi network card into monitor mode to capture network traffic. This mode allows the attacker to see all the traffic on the network.

Sending Fake Frames: The attacker sends fake deauthentication frames to the target client or access point. These frames spoof the identity of the client or the access point. The frames cause the target client or access point to disconnect.

Disconnecting the Client: When the target client receives the fake deauthentication frame, it disconnects from the current connection. This often causes the client to try to reconnect, but if the attacker continuously sends deauthentication frames, the client keeps getting disconnected.

Service Disruption: By continuously sending deauthentication frames, the target client is repeatedly disconnected, resulting in a denial of service. This prevents the client from accessing the internet or the network.

To protect the system against such attacks, more secure network protocols such as WPA3 should be used and it is important to use IDS/IPS to monitor network traffic and block unusual packets.

2.2. DoS (Denial of Service) Attack

With the development of Internet technology, DoS attacks have become the most widely used cyber attacks [13][14]. This type of attack aims to disrupt access to information/services on target devices/systems by the threat actor. The target can be any device, as well as network connections or site access. As a result of the attack, the target device/server becomes unavailable. To elaborate on the process, the threatening machine sends many requests to the target, and the target machine/server responds to these requests. Due to the overwhelming number of requests, the resources of the target device/server are depleted after a while, rendering the system unusable during this period. A decrease in system performance, temporary unavailability of web pages after a DDoS attack, and an increase in spam emails are symptoms of a DDoS attack [15].

DoS attacks are categorized as Flooding Attacks, Application Layer Attacks, Amplification attacks, and Resource Exhaustion attacks. Firewalls, IDS and IPS should be used more to protect the system from such attacks. In addition, Rate Limiting (limiting the packets coming from a source) should be done. The consequences of a DoS attack can also be mitigated by distributing traffic across multiple servers.

2.3. UDP (User Datagram Protocol) Flood Attack

Since it has become clear that UDP attacks are much faster and more effective than TCP attacks, attackers have started to perform more UDP flood attacks. The structure of the UDP protocol does not require the processing of a packet after it is received. For this reason, the attacker can disable the system by sending a very large number of packets to the target system. If the packets reached the destination, the attack was successful. In this type of attack, a randomly generated source IP is primarily created, and UDP packets are sent to random targets between main hosts. The targeted machine undergoing the attack;

- Checks whether there is an application listening on the relevant port,
- Responds with an ICMP packet stating "Destination Unreachable" when it is observed that no application is listening on the port.

When many UDP packets are sent, the target system is forced to respond with a significant number of ICMP packets. This situation can lead to the depletion of system resources, making it difficult for other clients to access the system.

To prevent such attacks, we recommend increasing the use of Rate Limiting (blocking multiple UDP packets from the same source) and IDS/IPS.

2.4. ICMP (Internet Control Message Protocol) Flood Attack

ICMP Flood Attack utilizes the Internet Control Message Protocol (ICMP) to send an echo packet to the target device, checking whether the target user is alive or not. In this type of attack, large volumes of ping packets are sent to the target device. These packets solicit a response from the target, consequently exhausting the bandwidth of the target network. During an ICMP Flood attack, the source IP can be spoofed. When the threat actor engages in IP spoofing to conceal their true identity, tracing the attack's origin becomes more challenging [16].

Effects of an ICMP Flood Attack; Network bandwidth saturation, System resource exhaustion, Service Disruption. Nowadays, these attacks are not very easy to perform because most network routers now reject packets sent to broadcast addresses on their networks.

2.5. SYN Flood Attack

During data exchanges between servers and targets on systems, the three-way handshake is witnessed. In this situation, known as the three-way handshake, the target receives the SYN packet, and information such as IP address and source connection is verified in the corresponding source table. Once these processes are completed, SYN-ACK packets are sent back to the client with pre-established identification information. In the final step, when the target receives the ACK packet, the table is queried again to verify whether the correct identification information has been received from the client by checking the acknowledgment number. If all steps are completed, the authentication is successful [17].

In SYN Flood attacks, interference occurs during this three-way handshake, initiating the attack. The goal of this attack is to overwhelm the system by sending more data than it can handle, similar to denial-of-service attacks, and prevent the establishment of connections.

In such attacks, the system creates half-open connections for SYN packets sent by the attacker and waits for a while. For this reason, network performance drops significantly and resources such as CPU and RAM used by the system are rapidly depleted.

2.6. Man in the Middle Attack

The Man-in-the-Middle (MitM) attack type aims to eavesdrop on the data between two connections. In this type of attack, not only can data be intercepted, but modifications to the data are also possible. The logic behind executing the attack can be expressed as follows: in environments with wireless network broadcasts, the network traffic is redirected through the threatening machine to eavesdrop on the data of individuals connected to this network, leading to the capture of the data. This interception occurs between the target and the network elements. These network elements can be a modem, router, server, or switch.

MitM attacks are challenging to detect as they acquire the location of clients without severing their connection to the network [18]. Attacks such as IP spoofing, DNS spoofing, HTTPS spoofing, Wi-Fi eavesdropping, and Session hijacking are sub-branches of the Man in the Middle attack. To prevent such attacks, it is important to use relatively more reliable HTTPS protocols, use multi-factor authentication (MFA), and use IDS and IPS, which are systems that detect and block abnormal traffic on the network.

3. Material and Method

During the establishment of the working environment, it was observed that there are numerous paid and free software options. Among these, the following free software have been selected for use in this study. The devices used for the test environment were selected based on the needs of the applications.

The devices used for the test environment and their details are as follows:

- Windows 10 Pro – x64 processor– 8,00 GB RAM – Computer
- Windows 10 Home – x64 processor– 4,00 GB RAM – Computer
- Tp-link – Archer C5v – AC1200 Wireless Dual Band Gigabit VoIP Router
- USB 2.0 Wireless 802.INN

The software used for the test environment is as follows;

- Oracle VM VirtualBox 6.1.16
- Debian – x64 processor – 2,00 GB RAM – Virtual Machine
- Wireshark 4.0.5
- CicFlowMeter

Windows 10 Pro was installed on the host machine, and it was transformed into a machine where cyber attacks would be executed by setting up a virtual machine. Oracle VM VirtualBox software was chosen for virtual machine installation. A USB 2.0 Wireless adapter connected the virtual machine to the wireless network.

For the execution of applications, a Windows 10 Home device was used as the target machine. The Wireshark tool, used for listening to network packets, was installed on this machine, and the network listening processes were carried out from this target machine.

The CicFlowMeter tool was utilized for feature extraction from the packets obtained with Wireshark. Details about the implementation are given in the proposed method section.

3.1. Machine Learning Applications in Cybersecurity

Machine learning, briefly defined, involves the parsing of data through specific algorithms, leading to the learning of parsed data and resulting in making judgments or predictions about a particular subject [19]. Machine learning enables us to comprehend data, and with technological advancements, it has become essential for handling vast amounts of generated data. Considering the significant increase in cyber threats in today's landscape, it is evident that the volume of data generated in this field has also reached substantial proportions.

In this study, data obtained in cybersecurity has been analyzed using machine learning methods.

3.1.1. Machine Learning Techniques

Various machine learning techniques are employed to facilitate the learning of machines in our surroundings. Since not every machine learning approach yields optimal results for all types of data, diverse machine learning techniques exist. In the data processing phase, performance metrics come into play to determine the most suitable learning technique for the data

When categorizing machine learning techniques, they can be grouped under four main headings, with numerous classification algorithms available for training. This study, however, focuses solely on the details of classification algorithms used during the application.

Supervised Learning: In supervised learning, the goal is to make inferences from labeled training data. The training data in supervised learning includes the input data and their corresponding labels. In other words, the outputs obtained during the testing phase are generated based on the information acquired from the provided datasets during training.

Unsupervised Learning: In these models, as testing processes are performed on data, the model's decision-making ability improves. This learning model attempts to learn the relationships between data based on the input data provided. Increasing the number of data and testing processes will enhance the success rate.

Semi-Supervised Learning: In this learning model, the number of labeled data is limited. Labeled data is used to predict unlabeled data. Multiple trials are conducted in this learning model, and learning is achieved from the acquired training experiences to obtain the best performance.

Reinforcement Learning: In reinforcement learning models, specific rules are employed to achieve the best results. Multiple different methods are used together in this model, and the model is created by determining the operation that yields the best results.

3.1.2. Classification Algorithms

Classification algorithms are a supervised learning technique used to determine the category of new observations using training data. These algorithms perform the learning process from the data set and classify these learnings into several classes/groups. These classifications can be referred to as labels/categories. In classification, the fundamental aim is to specify the class into which a new data point will fall. The concept of a classifier refers to an algorithm that maps given data inputs to a specific category. At the same time, the classification model can be defined as structures that evaluate input data given for training to derive certain results. A feature can be defined as an individual measurable property of an observed phenomenon.

- **Naive Bayes**

The Naive Bayes theorem, based on the review of probabilities, was first used by Thomas Bayes, and Naive Bayes Classifiers were developed using this theorem. This theorem allows the probability of the occurrence of a second event to be determined

when a certain event has occurred. In this scenario, the first event forms the evidence data, while the second event is the hypothesis.

There are three different models for this classifier: Gaussian, multinomial and Bernoulli. It is useful in cases where there is a large number of variable data. Additionally, in this classification algorithm, unlike other classification algorithms, as the number of feature data increases, the results obtained improve.

Advantages of the Naive Bayes Classifier include its ease of understanding and creation. Even when using the Naive Bayes Classifier in examples with large datasets, it can quickly complete the data training. Structurally, it is a very simple and fast algorithm [20].

The Naive Bayes Classifier has proven to deliver excellent results in various fields such as human motion recognition projects, traffic congestion projects, and medical research projects.

- **Gradient Boosting**

The Gradient Boosting algorithm can be used for both regression and classification models.

The steps of the Gradient Boosting algorithm are as follows:

Step 1. In solving a regression problem, the initial predictions for each data point are taken as the average of their values. The logarithm of the probabilities is taken, and this value is used as the probability for the class prediction.

Step 2. The loss value in predictions is calculated.

Step 3. A new decision tree is created using the predicted values. This tree is trained on the original dataset to learn.

Step 4. This new model is added to the ensemble. When making the next prediction with this value, it is implied that the first predictive value will be used along with the new decision tree.

Step 5. Steps 2 through 4 are repeated until the defined boundary for decision trees is reached or until improvement ceases after adding a new decision tree.

- **Support Vector Machines (SVM)**

Support Vector Machines (SVM) is a classification model that performs regression analysis. The supervised learning SVM model is based on statistical learning theory.

The explanatory variables are mapped into a high-dimensional space through non-linear structures, and then, an optimal hyperplane is created that effectively separates both classes. This hyperplane aims to maximize margins or the sum of the distances from each class's nearest training examples while minimizing classification errors [21].

Kernel functions are used in this classification method to transform the input data. This transformation is a process of converting input data into a high-dimensional space between two classes. The higher the separation between these data groups, the better the performance of the support vector machines.

- **K-Nearest Neighbors**

Conceptually, K-Nearest Neighbors is one of the easiest-to-understand classification algorithms. In the K-Nearest Neighbors classification algorithm, the feature values of sample data are plotted in an n-dimensional space, where n is the number of data features. Each point in the n-dimensional space is labeled with a class value. To explore the classification of an unlabeled data point, it is plotted in the n-dimensional space, and the class labels of the k nearest data points are noted. Typically, k is an odd number. The class that occurs the most among the k nearest data points is assigned as the class of the new data point. In other words, the decision is made by the vote of the nearest neighbors. One advantage of the K-Nearest Neighbors classification algorithm is its suitability for parallel processing [22].

3.1.3. Performance Metrics

In situations where the performance of a classification model needs to be evaluated for each class, class-based performance metrics can be used, aside from accuracy. Four conditions arise in binary guessing [20]:

True Positive (TP): Defined as examples that are actually positive and are correctly predicted as positive by the classifier.

False Positive (FP): Defined as examples that are actually negative but are incorrectly predicted as positive by the classifier.

False Negative (FN): Defined as examples that are actually positive but are incorrectly predicted as negative by the classifier.

True Negative (TN): Defined as examples that are actually negative and are correctly predicted as negative by the classifier.

Based on the information above, the precision value is calculated using the first equation, and the calculation of the recall value is provided in the second equation. To elaborate further: Precision is determined by dividing the total number of elements correctly predicted as positive (TP) by the sum of true and false positives (FP). In other words, it is the fraction of

correctly predicted positive instances out of all instances predicted as positive. Equation 1 and Equation 2 demonstrate the precision and recall values.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

Precision indicates how many of the predicted positive instances are positive. Recall, on the other hand, is obtained by dividing the true positive instances by the total number of instances classified as positive. Specifically, false negatives are instances that the model has labeled as negative but are actually positive. Recall measures the predictive accuracy of the model for the positive class; intuitively, it assesses the model's ability to find all positive instances in the dataset.

Accuracy is one of the most popular metrics in multiclass classification. The calculation for accuracy is given in the third equation.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

Accuracy provides a general measure of how many correct predictions the model makes across the entire dataset.

The F1 Score evaluates the performance of a classification model starting from the confusion matrix. It combines Precision and Recall measurements under the concept of the harmonic mean, as seen in the equation number 4 below.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

The formula for the F1 Score can be interpreted as a weighted average between precision and recall. The highest possible value for the F1 Score is 1, while the lowest is 0 [23].

3.2. The Proposed Method

The planned system for our study involves initiating cyber threats on wireless networks as the first step. The subsequent step involves recording the network traffic generated during the execution of cyber threats. Analyzing these recorded network packets constitutes the third step of the created methodology. In the subsequent steps, after extracting features and optimizing the process, machine learning approaches are employed for classification.

When performing the deauthentication Attack, the Wireshark tool was initially executed on the target machine to record network traffic during the attack. The network monitor mode was activated with the "airmon-ng start wlan0" command. The next step involved scanning nearby networks using the "airodump-ng wlan0" command. Following this, the attack was initiated by executing the commands "airodump-ng --channel channel number --bssid router mac address wlan0" and "aireplay-ng --deauth desired packet length -a attacked router BSSID -c target mac address wlan0". After running these commands, the attack commenced, causing the target device to disconnect from the network due to bandwidth saturation.

For the DOS attack, the Wireshark tool was run on the target device to capture network packets during the attack. As a result of the attack, the target device's bandwidth was filled, causing the device to be unable to perform network operations and eventually disconnect from the network.

To record network traffic during UDP Flood, ICMP Flood, and SYN Flood attacks, the Wireshark tool was started before each attack. During these attacks, UDP, ICMP, and SYN packets were separately sent to the target device. The extensive packets sent led to the saturation of the target system's bandwidth. Due to the bandwidth saturation, the target system was unable to perform network operations, resulting in a service outage.

Before initiating the Man in the Middle attack, Wireshark was run to capture packets on the network. The goal of the attack was to route the target machine's data through the threatening machine before reaching the server. This way, the target machine's network operations pass through the threatening machine before reaching the server.

The CICFlowMeter tool was used to extract features from pcap files obtained with Wireshark for feature extraction. The CICFlowMeter tool was first downloaded and installed. It was then prepared for feature extraction using network traffic data recorded by Wireshark. When starting CICFlowMeter, the data source to be used was specified, and the features to be extracted were selected. The feature extraction process was started by giving the necessary commands to the tool and the feature extraction process required for the classification of pcap files obtained after attacks has been completed. Table 4.2 lists the features extracted by the CICFlowMeter, an open-source tool that generates Bitflows from pcap files and extracts features from these flows [24].

After the feature extraction process of the packets obtained after the attacks was completed, various classification algorithms were used. These algorithms include Naive Bayes, Gradient Boosting, Support Vector Machines, and K-Nearest Neighbors. These algorithms were preferred because they are very fast in terms of calculations and perform well, especially on small data sets. Other models were not preferred because they showed low accuracy and performance. The random forest algorithm was not preferred due to its high memory usage problem, being less understandable, not being successful in data sets containing many features, and being more complex and slower than its alternatives. During the classification process, the necessary packets for the algorithms were first imported.

In the next step, after reading the data set file, a filtering process was performed due to the presence of non-numeric columns in the data. The names of the non-numeric columns are added to the `numeric_columns` list. Subsequently, only the numeric columns are assigned to the `data_numeric` variable, creating a data frame containing only numerical columns.

As a next step, the process of separating independent variables (x) and target variables (y) was carried out. The x variable is assigned all columns of the `data_numeric` data frame except for the last column, while the y variable is assigned the last column of the `data_numeric` data frame.

After separating the data set (x and y) into training and test sets, the `train_test_split` function was used to randomly select 80% for training and 20% for testing the x and y data sets. The `random_state` parameter was used to ensure the randomness of the data set's division or the repeatability of a random process.

The model-building process was carried out for each classification algorithm, and the training process was performed using the training data set (`x_train` and `y_train`). The trained model was then used to predict the test data set (`x_test`).

The prediction results (`y_pred`) were compared with the actual target values (`y_test`). The model's accuracy was calculated and printed to the screen, and the necessary data for the comparison process was obtained.

For the Support Vector Machines classification algorithm, the necessary packages were added, and the process of reading the CSV file and filtering non-numeric columns was performed. In the next step, the process of separating independent variables (x) and target variables (y) was carried out. The x variable was assigned all columns of the `data_numeric` data frame except for the last column, and the y variable was assigned the last column of the `data_numeric` data frame. Additionally, categorical variables were converted to numerical values.

After separating the data set (x and y) into training and test sets, the `train_test_split` function was used to randomly select 80% for training and 20% for testing the x and y data sets. The `random_state` parameter was used for the randomness of the data set's division or the repeatability of a random process.

After the model-building and accuracy score calculation processes, the Support Vector Machines classification process was completed.

Then, the required packages for the Gradient Boosting classification algorithm are imported. In the next step, after reading the CSV file, a filtering process was performed due to the presence of non-numeric columns in the data. The names of the non-numeric columns are added to the `numeric_columns` list. Subsequently, only the numeric columns are assigned to the `data_numeric` variable, creating a data frame containing only numerical columns.

The process of separating independent variables (x) and target variables (y) was carried out. The x variable was assigned all columns of the `data_numeric` data frame except for the last column, while the y variable was assigned the last column of the `data_numeric` data frame.

After converting categorical variables to numerical values, the data set (x and y) is split into training and test sets. The `train_test_split` function was used to randomly select 80% for training and 20% for testing the x and y data sets. The `random_state` parameter was used to ensure the randomness of the data set's division or the repeatability of a random process.

After the model-building and training processes, performance metrics were calculated.

For the K-Nearest Neighbors classification algorithm, the required packages were imported, and the CSV file was read. Due to the presence of non-numeric columns in the data, a filtering process was performed. The names of the non-numeric columns are added to the `numeric_columns` list. Subsequently, only the numeric columns are assigned to the `data_numeric` variable. In the next step, the process of separating independent variables (x) and target variables (y) was carried out. The x variable was assigned all columns of the `data_numeric` data frame except for the last column, and the y variable was assigned the last column of the `data_numeric` data frame. After converting categorical variables to numerical values, the data set (x and y) was split into training and test sets. The `train_test_split` function was used to randomly select 80% for training and 20% for testing the x and y data sets. The `random_state` parameter was used for the randomness of the data set's division or the repeatability of a random process.

The model-building and training processes were completed, and with the completion of the classification processes, the classification algorithm providing the best performance was observed based on the performance values obtained.

3.3. The Other Datasets in the Literature

Various datasets have been created to train IDSs developed to detect intrusions into wireless networks. In this part of our study, we analyze and compare the most commonly used data sets.

3.3.1. KDDCup99

KDDCup99 was created by obtaining data from a military network environment and is one of the most widely used data sets [25][26]. There are 23 attack data and 1 normal class data in the data set. The attack group is categorized as DoS, Probe, R2L and U2R attacks. There are 4898431 training data and 311029 test data in the dataset. 74.41% of the attacks were DoS, 1.33% Probe, 0.07% U2R and 4.68% R2L attacks [27]. KDDCup99 requires less memory and processing power, and the dataset contains easily available features [28]. Although KDDCup99 is widely used, it also has disadvantages. Because it contains synthetic data, it does not match real network traffic. Besides, the amount of training and testing data is huge and therefore has a complex structure. And the detection accuracy rate is low.

3.3.2. NSL-KDD Dataset

To address the shortcomings of the KDD99 dataset, The NSL-KDD dataset was introduced by Tavallaee et al. [29]. This dataset was developed by removing unnecessary and redundant data from the KDDCup99 dataset and contains only the data that is truly necessary. There are 37 attacks in total, and 27 attacks were used to test the model and 23 attacks were used to train the dataset. It includes Probe attacks, Denial of Service attacks (DoS), User to Root (U2R) and Remote Local (R2L) attacks. The correct detection rate of NSL-KDD is much higher than that of KDDCup99, and the performance of the models is even higher due to the removal of unnecessary data. However, it is still an improved version of KDDCup99 and since it uses the same data, it does not reflect realistic network data.

3.3.3. UNM Dataset

The UNM dataset was introduced in 2004 and includes data such as buffer overflows, symbolic link attacks and trojan programs [30]. Although UNM is more up-to-date than the KDD dataset, the UNM dataset is extremely limited in coverage and cannot replace the KDD dataset. This data set is not used today. Since it was produced in the 1990s and contains fewer and less complex features compared to modern data sets, it cannot be used by researchers in areas that require in-depth analysis.

3.3.4. CICIDS2017

CICIDS2017 was created by the Canadian Security Institute and includes 5 days of normal and attack data. The data set includes attack types such as DDoS, Brute Force, Botnet, XSS, and SQL Injection. The dataset provides a broad set of features. In this way, researchers are allowed to make in-depth analyses. It contains 3119345 data in total and these data are located in eight different files. Since the data set consists of a lot of data and has a complex structure, it consumes a lot of time for data loading and processing. In addition, there is a large class imbalance in the CICIDS2017 dataset. This causes the intrusion detection system to raise too many false alarms.

3.3.5. UNSW-NB15 Dataset

It started to be developed after it was realized that KDDCup and NSL-KDD did not provide good enough and realistic results in an IDS evaluation [31]. This dataset was created using the IXIA Perfect Storm tool and was created at the Australian Center for Cyber Security (ACCS). Twelve algorithms were used to create the dataset with 49 features, including the class label [32]. The data set consists of attack and normal data, a total of 2.5 million data. It contains Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms attack data. It has more realistic data than its alternatives, but the high amount of data and 49 features can make the modeling process of this data set complicated.

4. Results and Discussions

As a result of the conducted studies, performance metrics have been calculated for each classification. The values related to the data quantity used for the classification process are provided in Table 2. The distribution of data quantities has influenced the interpretation of the obtained performance values. In cases of deauthentication, ICMP, and SYN Flood attacks, the data volume is minimal, leading to classifiers achieving accuracy rates very close to 0 or 1. In our future studies, we aim to increase the data volume associated with these attack types. The amount of data we have in our dataset is given in Table 2.

Table 2. Data Count

Attack	Amount of Data
Deauthentication	122
Dos Attack	5430
ICMP Flood	143
SYN Flood	140
Man in the Middle	2161

The accuracy value for the Naive Bayes classification algorithm is 0.74, and the values for other performance metrics are provided in Table 3. Due to the low amount of data associated with Deauthentication, ICMP, and SYN Flood attacks, Naive Bayes has not achieved satisfactory performance in classifying these attacks.

Table 3. Performance Metrics of Naive Bayes Algorithm

Attack	Precision	Recall	F1-Score
Deauthentication	0.00	0.00	0.00
Dos Attack	0.91	0.99	0.95
ICMP Flood	0.00	0.00	0.00
SYN Flood	0.11	0.50	0.18
Man in the Middle	1.00	0.32	0.48

The accuracy value of the Gradient Boosting classification algorithm is 0.91, and other performance metrics are provided in Table 4. The gradient Boosting algorithm has demonstrated better performance compared to the Naive Bayes algorithm in classifying deauthentication and SYN Flood attacks.

Table 4. Performance Metrics of Gradient Boosting Algorithm

Attack	Precision	Recall	F1-Score
Deauthentication	0.33	0.25	0.29
Dos Attack	0.98	0.99	0.99
ICMP Flood	0.00	0.00	0.00
SYN Flood	0.33	0.50	0.40
Man in the Middle	0.91	0.86	0.89

The accuracy value of the Support Vector Machine classification algorithm is 0.84, and other performance metrics are provided in Table 5.

Table 5. Performance Metrics of Support Vector Machine Algorithm

Attack	Precision	Recall	F1-Score
Deauthentication	0.00	0.00	0.00
Dos Attack	0.96	0.86	0.90
ICMP Flood	0.00	0.00	0.00
SYN Flood	0.00	0.00	0.00
Man in the Middle	0.69	0.96	0.80

The accuracy value of the K-Nearest Neighbors classification algorithm is 0.85, and other performance metrics are provided in Table 6.

Table 6. Performance Metrics of K-Nearest Neighbors Algorithm

Attack	Precision	Recall	F1-Score
Deauthentication	0.00	0.00	0.00
Dos Attack	0.93	0.93	0.93
ICMP Flood	0.00	0.00	0.00
SYN Flood	0.00	0.00	0.00
Man in the Middle	0.79	0.82	0.80

The performance values for all classification algorithms are provided in Table 7.

Table 7. Comparison of Performance Values of Classification Algorithms

Algorithm	Accuracy	Precision	Recall
Naive Bayes	0.74	0.40	0.36
Support Vector Machines	0.84	0.51	0.52
Gradient Boosting	0.91	0.33	0.36
K-Nearest Neighbors	0.85	0.44	0.35

As seen in the table above, the highest accuracy value is obtained in the Gradient Boosting classification algorithm, while the lowest is obtained from the Naive Bayes classification algorithm. The accuracy value of the Support Vector Machines classification algorithm is higher than that of the K-Nearest Neighbors classification algorithm.

5. Conclusions and Suggestions

In the created test environment, potential attacks that could threaten wireless networks were simulated. Alongside these cyber threats, network traffic was recorded. The objective was to animate the traffic that might occur during potential cyber threats, record these network packets, and analyze the recorded packets. Different attacks lead to different outcomes in the network, and the varied reactions of the network to these results allow obtaining different data for each attack.

In the next step, the features of the network packets obtained from the created test environment were extracted. These features were prepared for the optimization process and classification steps using machine learning approaches. The CICFlowMeter was used for feature extraction, resulting in different levels of efficiency in the obtained features for each attack. This variation was due to the CICFlowMeter's inability to separate attack packets into their features in some attack scenarios.

For the classification process, various classification algorithms were employed. The scores obtained by the dataset from the applied classification algorithms were collected. The accuracy scores for Naive Bayes, Support Vector Machine, Gradient Boosting, and K-Nearest Neighbors algorithms were 0.74, 0.84, 0.91, and 0.85, respectively. While the scores of K-Nearest Neighbors and Support Vector Machines were very close, there was a significant difference in scores between the highest and lowest-scoring classification algorithms. Although a high accuracy value generally indicates correct classification by the algorithm, the presence of imbalances in classification for cyber attacks suggests considering other performance metrics.

As a result of the performed operations, after the classification processes, the highest score was obtained from the Gradient Boosting classification algorithm, while the lowest score was obtained from the Navie Bayes classification algorithm.

6. Suggestions and Future Works

Since the data amounts of Deauthentication, SYN Flood and ICMP flood packets in our data set are very small, some machine learning classifiers failed to detect the attack data. In the future, we aim to improve the performance of these classifiers by increasing the number of data in our dataset.

The feature extractor used during the feature extraction process for some attacks, CICFlowMeter, has proven to be inadequate as it cannot separate attack packets into their features. In this regard, it is recommended to develop better feature extraction software or conduct a more comprehensive study using existing software with better performance.

Additionally, increasing the number and diversity of cyber attacks and expanding the study using different classification algorithms are suggested. This could lead to obtaining better scores with classification algorithms that perform well under various conditions. Finally, considering the increasing cyber threats with the evolving and developing technology, it is recommended that awareness be raised among the public. Alongside awareness campaigns, increasing scientific studies in this field is also suggested.

References

- [1] A. N. Ozalp, Z. Albayrak, and A. Zengin, "Expansion of Wireless Networks using IEEE 802.3af Protocol in Protected Areas," in *5th International Symposium on Innovative Technologies in Engineering and Science*, 2017.
- [2] M. Wazid, A. K. Das, V. Chamola, and Y. Park, "Uniting cyber security and machine learning: Advantages, challenges and future research," 2022. doi: 10.1016/j.ict.2022.04.007.
- [3] S. GÖNEN, H. İ. ULUS, and E. N. YILMAZ, "Bilişim Alanında İşlenen Suçlar Ve Kişisel Verilerin Korunması," *Bilişim Teknol. Derg.*, vol. 9, no. 3, Sep. 2016, doi: 10.17671/btd.90710.
- [4] E. AKBAL, Ş. DOĞAN, T. TUNCER, and N. S. ATALAY, "Adli Bilişim Alanında Ağ Analizi," *Bitlis Eren Üniversitesi Fen Bilim. Derg.*, vol. 8, no. 2, pp. 582–594, 2019, doi: 10.17798/bitlisfen.479303.
- [5] K. A. Dhanya, S. Vajipayajula, K. Srinivasan, A. Tibrewal, T. S. Kumar, and T. G. Kumar, "Detection of Network Attacks using Machine Learning and Deep Learning Models," *Procedia Comput. Sci.*, vol. 218, pp. 57–66, 2023, doi: 10.1016/j.procs.2022.12.401.
- [6] R. Ahmad, R. Wazirali, and T. Abu-Ain, "Machine Learning for Wireless Sensor Networks Security: An Overview of Challenges and Issues," 2022. doi: 10.3390/s22134730.
- [7] A. Mughaid *et al.*, "Improved dropping attacks detecting system in 5g networks using machine learning and deep learning approaches," *Multimed. Tools Appl.*, vol. 82, no. 9, pp. 13973–13995, Apr. 2023, doi: 10.1007/s11042-022-13914-9.
- [8] M. Waqas, S. Tu, Z. Halim, S. U. Rehman, G. Abbas, and Z. H. Abbas, "The role of artificial intelligence and machine learning in wireless networks security: principle, practice and challenges," *Artif. Intell. Rev.*, vol. 55, no. 7, pp. 5215–5261, Oct. 2022, doi: 10.1007/s10462-022-10143-2.
- [9] D. M. Gezgin and E. Buluş, "Kablosuz Erişim Noktalarına Yapılan DoS Saldırıları," pp. 83–89, 2008.
- [10] A. N. Kadhim and S. B. Sadkhan, "Security Threats in Wireless Network Communication-Status, Challenges, and Future Trends," in *2021 International Conference on Advanced Computer Applications (ACA)*, IEEE, Jul. 2021, pp. 176–181. doi: 10.1109/ACA52198.2021.9626810.
- [11] D. Cossa, "The Dangers of Deauthentication Attacks in an Increasingly Wireless World," *Iowa State Univ.*, vol. 537, 2014.
- [12] R. Cheema, D. Bansal, and S. Sofat, "Deauthentication/Disassociation Attack: Implementation and Security in Wireless Mesh Networks," *Int. J. Comput. Appl.*, vol. 23, no. 7, pp. 7–15, 2011, doi: 10.5120/2901-3801.
- [13] W. Liu, "Research on DoS attack and detection programming," in *3rd International Symposium on Intelligent Information Technology Application, IITA 2009*, 2009. doi: 10.1109/IITA.2009.165.
- [14] A. N. Ozalp, Z. Albayrak, M. Cakmak, and E. Ozdogan, "Layer-based examination of cyber-attacks in IoT," in *HORA 2022 - 4th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, 2022. doi: 10.1109/HORA55278.2022.9800047.
- [15] D. Mertkan Gezgin and E. Buluş, "KABLOSUZ AĞLARIN GÜVENLİK AÇIKLARININ EĞİTİM AMAÇLI İNCELENMESİ İÇİN UYGULAMA TASARIMI," *Cilt*, vol. 2, no. 1, pp. 127–135, 2012.
- [16] H. (Harshita) Harshita, "Detection and Prevention of ICMP Flood DDOS Attack," *Int. J. New Technol. Res.*, vol. 3, no. 3, p. 263333, 2017. [Online]. Available: <https://www.neliti.com/publications/263333/>
- [17] Z.-Y. Shen, M.-W. Su, Y.-Z. Cai, and M.-H. Tasi, "Mitigating SYN Flooding and UDP Flooding in P4-based SDN," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, Sep. 2021, pp. 374–377. doi: 10.23919/APNOMS52696.2021.9562660.
- [18] M. Thankappan, H. Rifà-Pous, and C. Garrigues, "Multi-Channel Man-in-the-Middle attacks against protected Wi-Fi networks: A state of the art review," *Expert Syst. Appl.*, vol. 210, p. 118401, Dec. 2022, doi: 10.1016/j.eswa.2022.118401.
- [19] B. L. Aylak, O. Oral, and K. Yazici, "Using artificial intelligence and machine learning applications in logistics," 2021. doi: 10.31202/ecjse.776314.
- [20] A. N. Özalp and Z. Albayrak, "Detecting Cyber Attacks with High-Frequency Features using Machine Learning Algorithms," *Acta Polytech. Hungarica*, 2022, doi: 10.12700/APH.19.7.2022.7.12.
- [21] A. Robles-Velasco, P. Cortés, J. Muñuzuri, and L. Onieva, "Prediction of pipe failures in water supply networks using logistic regression and support vector classification," *Reliab. Eng. Syst. Saf.*, vol. 196, p. 106754, Apr. 2020, doi: 10.1016/j.ress.2019.106754.
- [22] V. J. Pandya, "Comparing Handwritten Character Recognition by AdaBoostClassifier and KNeighborsClassifier," in *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, Dec. 2016, pp. 271–274. doi: 10.1109/CICN.2016.59.
- [23] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," pp. 1–17, 2020,

- [Online]. Available: <http://arxiv.org/abs/2008.05756>
- [24] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *ICISSP 2017 - Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, 2017, pp. 253–262. doi: 10.5220/0006105602530262.
- [25] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, L. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: A survey," *Eurasip J. Wirel. Commun. Netw.*, 2013, doi: 10.1186/1687-1499-2013-271.
- [26] C. Koliass, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," *Comput. Secur.*, 2011, doi: 10.1016/j.cose.2011.08.009.
- [27] O. Atilla and E. Hamit, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ*, 2016.
- [28] R. Bala, "A REVIEW ON KDD CUP99 AND NSL-KDD DATASET," *Int. J. Adv. Res. Comput. Sci.*, 2019, doi: 10.26483/ijarcs.v10i2.6395.
- [29] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, 2009. doi: 10.1109/CISDA.2009.5356528.
- [30] Y. Hamid, V. R. Balasaraswathi, L. Journaux, and M. Sugumaran, "Benchmark Datasets for Network Intrusion Detection: A Review," *Int. J. Netw. Secur.*, 2018.
- [31] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, 2015. doi: 10.1109/MilCIS.2015.7348942.
- [32] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J.*, 2016, doi: 10.1080/19393555.2015.1125974.

Author(s) Contributions

İmran KAÇAN: Data collection, data analysis, writing. Batuhan GÜL: Design, writing, review of content, literature review. Fatih ERTAM: Design, Data analysis, technical support, review of content.

Acknowledgments

This study is supported by the Firat University Scientific Research Projects Coordination Unit with project number TEKF.23.12.

Conflict of Interest Notice

There is no conflict of interest.

Support/Supporting Organizations

This study is supported by the Firat University Scientific Research Projects Coordination Unit

Ethical Approval

Our study does not require ethics committee approval.

Availability of data and material

Data will be provided upon request.

Plagiarism Statement

This article has been scanned by iThenticate™.

Classification of Solar Cells EL Images with Different Busbars Via Deep Learning Models

Miktat Aktaş¹, Ferdi Doğan², İbrahim Türkoğlu³

¹ GTC GUNES SAN. ve TIC. A.S./R&D Center, Adıyaman, Türkiye

² Adıyaman University, Computer Engineering, Adıyaman, Türkiye

³ Fırat University, Software Engineering, Elazığ, Türkiye

Corresponding author:

Miktat Aktaş, GTC GUNES SAN. ve TIC.
A.S./R&D Center, Adıyaman, Türkiye
mkttkts@gmail.com



Article History:

Received: 03.04.2024

Accepted: 03.06.2024

Published Online: 26.08.2024

ABSTRACT

Electricity generation from renewable energy sources such as solar energy has come to the forefront in the last decade. The solar energy cell is an indispensable part of the solar energy ecosystem of solar panels, and defective cells cause financial losses in energy production. Experienced experts are needed to detect defects on solar cells. Autonomous systems are important to accelerate the process. Classical image processing techniques are used to manually detect defects on cells. To use these techniques, many parameters are needed to be entered into EL imaging software. However, in this study, these processes were carried out automatically without the need for external intervention. False detection/classification may occur during the processes performed by EL imaging devices due to weakness of the operator experience or EL imaging software. It is aimed to use automatic image processing and then deep learning techniques to achieve faster and higher performance than the results obtained from EL imaging devices using classic image processing techniques. AI algorithm and deep learning models can be an important solution. In this study, two AI algorithm and 10 different deep learning models were used to classify solar cells. EL images of defective and normal solar cells with 4 and 5 busbars were used in the study. The dataset, includes 9360 images of solar cells, 4680 of which are defective and 4680 are normal. Performance evaluation of the models made according to the confusion matrix. According to the results, Mobilenet-v2 and VGG-19 achieved the highest validation accuracy rate of 99.68%. According to F1-score, Mobilenetv2 achieved the highest performance of 99.73%. It has been shown that the Mobilenet-v2 is slightly more successful than other models in terms of validation and F1-score. The results show that trained DL models can be used as an inspection method in the production line of solar panels and cells.

Keywords: Solar cells, Deep learning, Image processing, EL Image, Defects

1. Introduction

Electricity generation from solar panels has been very popular recently, just like artificial intelligence-supported systems. The quality of solar cells, which is the most important element of the solar panel and determines the maximum electricity production capacity, power and performance, is undoubtedly very important for manufacturers and users. For this purpose, many defects detection and classification equipment/processes are used during solar cell/panel production and after panel installation on solar plants. Determination of cell defects from Electroluminescence (EL) images is one of these methods. In EL imaging, it is possible to detect cell defects, especially in PV panels. Cell defects appear as dark lines on the solar cell in the EL image. Especially in multicrystalline solar cells, crystallographic defects typically appear as dark lines. Cell cracks are detected by a person trained to recognize cell cracks in PV cells and panels. A well-trained person can detect cracks by looking at the EL image of a solar panel. [1]

As shown in Figure 1, the classic EL imaging test is conducted in a dark environment by applying reverse current to the PV solar panel and the reflected photons are captured by the CCD (Charge Coupled Device) camera; the images similar to radiology results as in Figure 2. The captured image can be interpreted only by experts as in radiology results. EL images are controlled by an expert, and if s/he detects any defects in cells, these cell strings must be replaced with non-defective cell strings before the lamination process. Therefore, manual classification and inspection may cause a waste of time and some misclassification problems. Especially during solar cell/panel production, classification of the cell defects is quite helpful for quality & inspection processes.

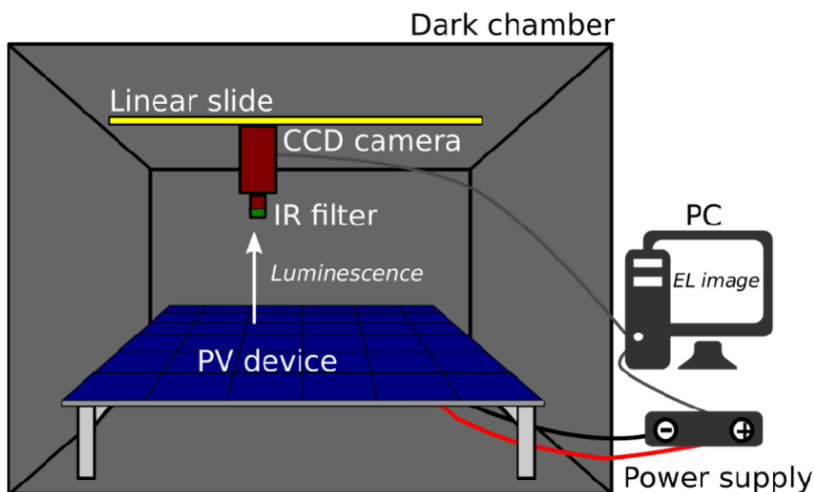


Figure 1. EL Imaging Technique Representation [2]

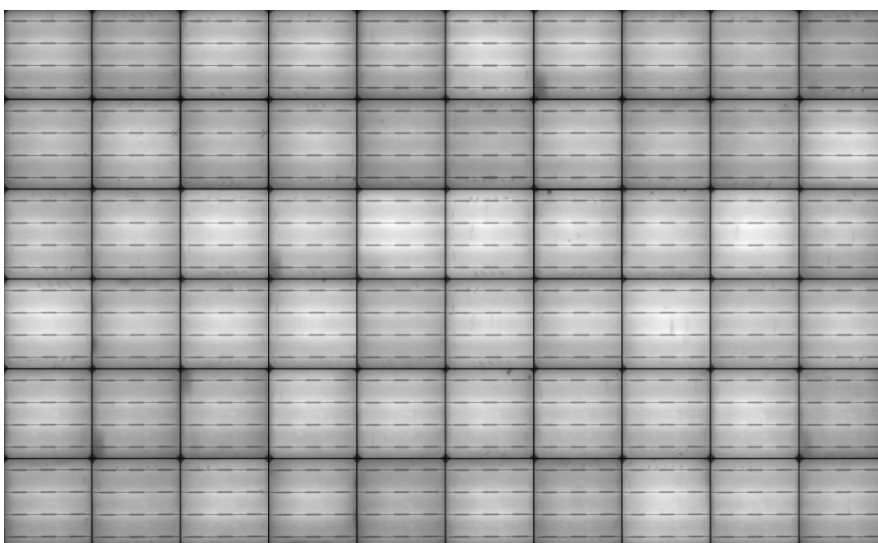


Figure 2. EL Image of 60 Cells PV Solar Panel

The Classic EL imaging software segments the image consisting of 60 or 72 cell images according to the parameters entered by the operators and writes the evaluation result of each cell to an XML file. In some cases, due to parameter or operator-related reasons, false cell segmentation may occur as shown in Figure 3.

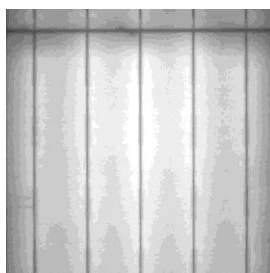


Figure 3. False Segmented Solar Cell

There are some cell defect types like micro-cracks, finger interruptions, material defects, Finger Interruptions, Cell Interconnection Problems shown in Figure 4. Each defect type impacts panel performance differently; even a combination of several may result in negligible effect. In some cases, depending on the size and number of defects may cause a panel performance loss of up to 60% [3]

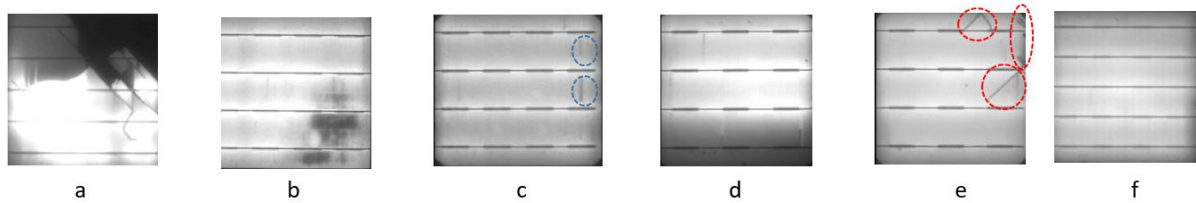


Figure 4. Types of Defects and Normal Cells in the Dataset.

- a) Electrically Insulated Cell Parts b) Material Defects c) Finger Interruptions d) Cell Inter-connection Problems
e) Microracks f) Cell without any defects (Normal)

The images obtained from the EL device are presented to the expert using classical image processing techniques. The expert examines the images from the EL device and manually detects cell damage in the image.

Using deep learning models for inspection of EL images, it will provide minimizing the faulty detections caused by the operator and the machine, reducing the time spent for damage detection to less than 1' min and to carry out the process in a healthy way without the need for an operator at the station.

Deep learning models used in the study, the arrangement and branching of the DL (Deep Learning) layers are differ. A sample architecture showing the layers used in deep learning models is shown in Figure 5.

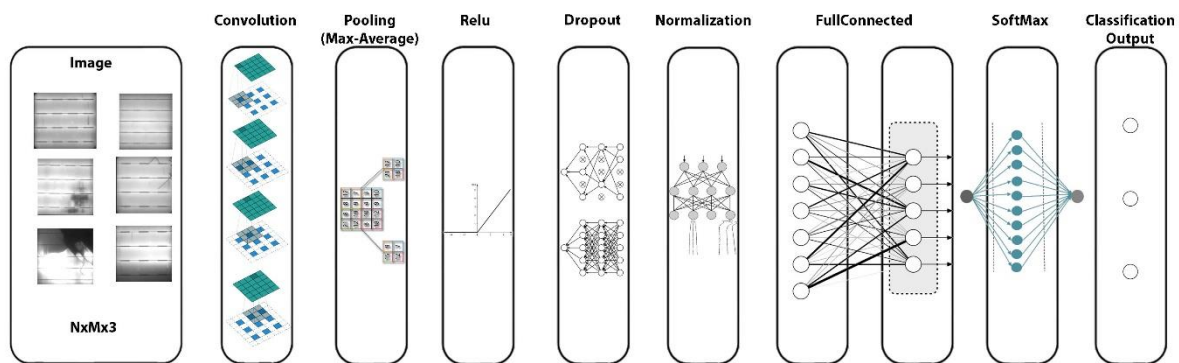


Figure 5. Sample Layer Structure Used in DL Models for Classification Problems[4]

A dataset consisting of defective and normal cells, the input image in Figure 5, is put as input data to a sample DL model. Then, feature extraction is performed via the convolutional layer. In the next layers, the image size used as input for the network is reduced by the pooling process. Relu is used for the activation function. The probability of memorization of the network is reduced, and some of the random weights in the network are discarded in the dropout layer. The image data is standardized via the normalization layer. In a fully connected layer, every neuron makes connections to all neurons in the prior layer. This allows the network to integrate the local features extracted earlier, like edges and shapes, into a bigger picture, ultimately recognizing complex patterns across the entire image. The Softmax layer is the entropy layer and it makes probabilistic estimation and estimates which class the input image belongs to via the Classification layer, [5]

This study used 10 Deep Learning CNN (Convolution Neural Network) models, which have different numbers and types of layers for cell classification. These models are Mobilenetv2, Darknet19, Darknet53, Alexnet, Googlenet, Vgg16, Vgg19, Resnet50, Resnet-101, Densenet201. In addition to these DL models we used classic AI (Artificial Intelligence) algorithms as SVM (support Vector Machine) and KNN (K-Nearest Neighbor). In this way, it will be possible to see which model achieve more successful results and this study will guide the researchers who work in this field. The dataset used in this study was selected and validated from a real production line and by experienced engineers. In addition, the Confusion matrix was used as a measure of the success of the models and precision, accuracy, sensitivity, and F1-score parameters were used.

In the introduction section, information about cell defect classification using classical image processing techniques is given and what can be done with the use of AI is briefly explained. In 2 related studies, previous studies for the detection of cell damage and the methods used in these studies are stated in detail. The 3 material and method sections mention the method we used in our study and how the study progressed. In the results section 4, the results we obtained with different deep learning models in our study are explained. The results and evaluation were made in 5 discussion sections. In the 6 discussion sections, there are discussions about the results obtained.

2. Related Work

In 2021, we studied crack detection on the solar cell by using the Alex-Net DL model in the dataset we had created ourselves. According to the results we gained from the study, we achieved the detection of cracks in solar cells with an accuracy of 79.40%. [3]

In Table 1, related to the studies listed that were classified as defect/undefect, the common feature of all of them was that the number of images used in the dataset and the number of cell defects were limited, An imbalanced dataset had been used in some researches [6][7] to get the most accurate result we had created and used a dataset that is balanced, much larger and included more defect types. At the same time, it achieved the highest result in metrics such as Accuracy, Recall, Precision, and F-1 score. Besides, 10 different DL models and 2 traditional AI were used for training and testing. Lastly, Alaa S. Al-Waisy studied on two different datasets their class numbers are unbalanced. To overcome this situation, they use some data augmentation techniques. They also trained and tested the datasets by pre-trained and hybrid model and get highest performance with the hybrid model [8].

Lee et al, developed a deep learning architecture called LIRNET (Local Integral Regression Network) for a dataset of 20000 infrared images and 12 classes and performed classification in 2 phases. The dataset is unbalanced like the dataset used in other studies, and they achieved an 89% accuracy rate with the used architecture [9].

Deutsch S. et al, studied on a limited number of dataset and try to automatic segmentation in both defect and normal cell images and they get F1 score as 97.23% [10]. Akram Waqar et al, studied on same dataset of Deutsch S. et al, they classify the dataset as defect and not defect and get 92.80% Accuracy rate [11].

Amirul Anwar S. et al, studied with a very limited dataset and they classify the dataset as defect and not defect and get 83.89% Accuracy rate [12]. Bartler A. et al, studied with a relatively large dataset and classify the dataset as defect and good and get 87.04% Recall rate [13]. Wang J. et al, studied with two different datasets and classified the dataset as defective and normal and get a 96.17% Accuracy rate [14]

In Table 1, related research is listed with the researcher's name, number of classifications, dataset size and test results in terms of Accuracy, Recall, Precision and F1-score.

Table 1. Related Works

Studies	Classification	Number of Total El Images	Accuracy	Recall	Precision	Highest F-1 Score
Balzategui J. et al [6]	Defect vs Not Defect	542	-	0.920	0.850	0.883
Balzategui J. et al [7]	Defect vs Not Defect	542	-	0.875	0.823	0.847
Waqar Akram M. et al [11]	Defect vs Not Defect	2624	0.928	0.920	0.93	0.9249
Amirul Anwar S. et al [12]	Defect vs Not Defect	600	0.8389	0.971	-	-
Bartler A. et al [13]	Defect vs Good	98280		87.04		
Wang J. et al. [14]	Defective vs Normal	2223 & 5991	0.9617	0.9516	0.9603	0.9557

The differences between classical image processing techniques and artificial intelligence studies are included in the related studies section. Here, it has been seen that studies carried out with deep learning models, one of the artificial intelligence techniques, have achieved successful results in damage detection. It is thought that this study, conducted to fill the gap in this subject, contains important findings that will contribute to the literature.

3. Material and Method

a. Requirement

Cracked or damaged solar cells cause efficiency loss in production and, consequently an increase in production costs. Micro cracks and defects not only reduce cell productivity in that area but also reduce cell reliability. [1]

A damaged cell or group of cells may cause hot-spot heating problems when the operating current in a panel exceeds the reduced short-circuit current of the fault cell. the cell is forced into reverse current and must dissipate this accumulated power. Indeed, if the dissipation force is large enough, this reverse-polar cell can overheat and melt the solder or cause the back sheet to deteriorate (Figure 6). Hot-spot cells show low parallel resistance when reverse current performance is limited by current, or high parallel resistance when reverse current performance is limited by voltage. In either case, the cell may experience hot-spot problems in different ways. [15]

Each cell defects have different impacts; for instance, dark area causes reduced power output immediately, while microcracks may leads a reduced power output in the future. Because, many operators of solar production line wish an automated detection of defect cells and a further classification of defect cells into various defect categories to decide which solar modules must to be replaced immediately or in the future. [13]

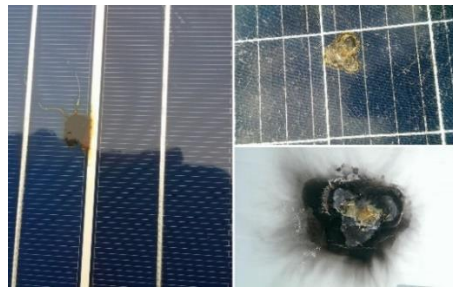


Figure 6. Hotspot's Effect on Solar Panel [15]

b. Method

As shown in Figure.7 First, firstly we extracted cell images from 72- or 60-cell panel EL images manually (manually means that we didn't use any AI methods for this process instead we have gathered the dataset among 200.000 cell images' by C# based software which we designed and coded. The software firstly import images metadata stored in xml file to sql database then filter defective cells according to this data.), all images had gathered from the EL machine used in the production line, then experts identified and labeled the defect cells from images. Sufficient defect and normal cells were collected for the dataset, 9260 EL images. 80% of the images were used for training and 20% for testing. 10% of the images used for training were also used for validation. The training and test images selected in the study are taken randomly. The training and test images in the model are automatically generated from the dataset. Then, various deep learning models were trained with this dataset and tested. At the same time, alexnet-based feature extraction method used for SVM and KNN, then classification made with these classic AI algorithms. Fully connected layer named as 'fc6' feature extraction layer used. The results were compared each other and the most successful deep learning model was determined using the confusion matrix and metrics gathered from the matrix.

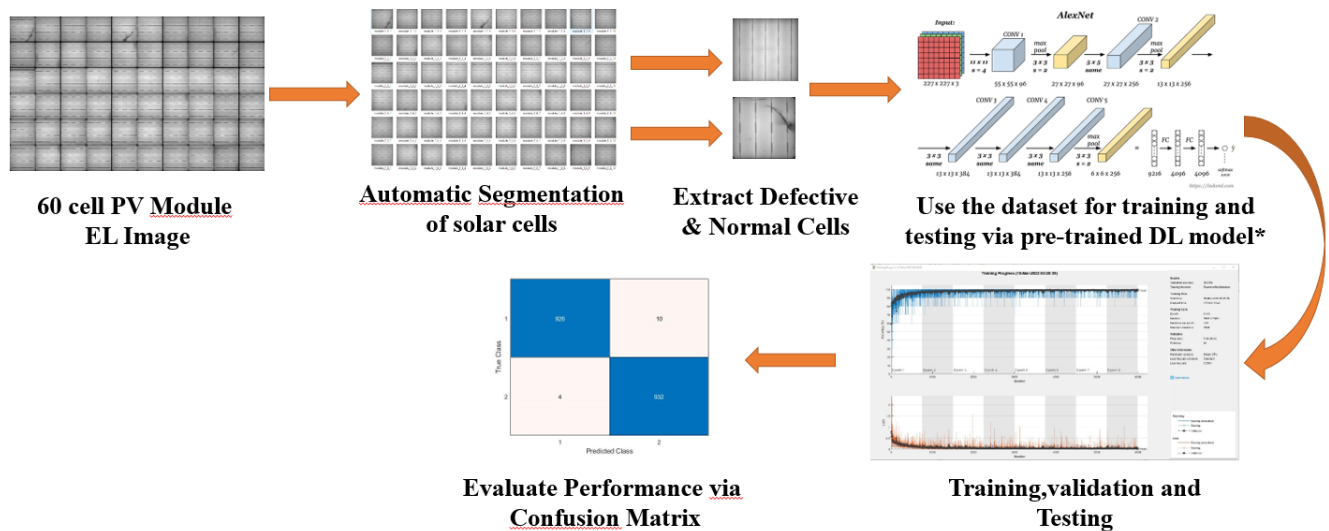


Figure 7. Flow Chart for Solar Cells Classification ,*[16]

c. Dataset

There are 9360 EL images with high resolution of solar cells, 4680 of which are defective (Electrically Insulated cell parts, microcracks, material defect, finger interruption, cell interconnection problems), as shown in Figure 4, and 4680 are non-defect. Each cell size in the panel varies between 950x960-930x850. The reason for the different cell sizes is that the panel images were taken in different sizes. Some panels contain 72 solar cells while some panels consist of 60 solar cells. As shown in Figure Figure 8, 41% of solar cell images have 5 busbars, and 59% have 4 busbars. The image resolutions were adjusted according to the required input size of the DL models before training. The dataset is separated randomly for training and testing. 80% (7488 images) of the dataset was used for training, %10 of the reserved dataset for training was used for validation and the remaining 20% (1872 images) for testing.

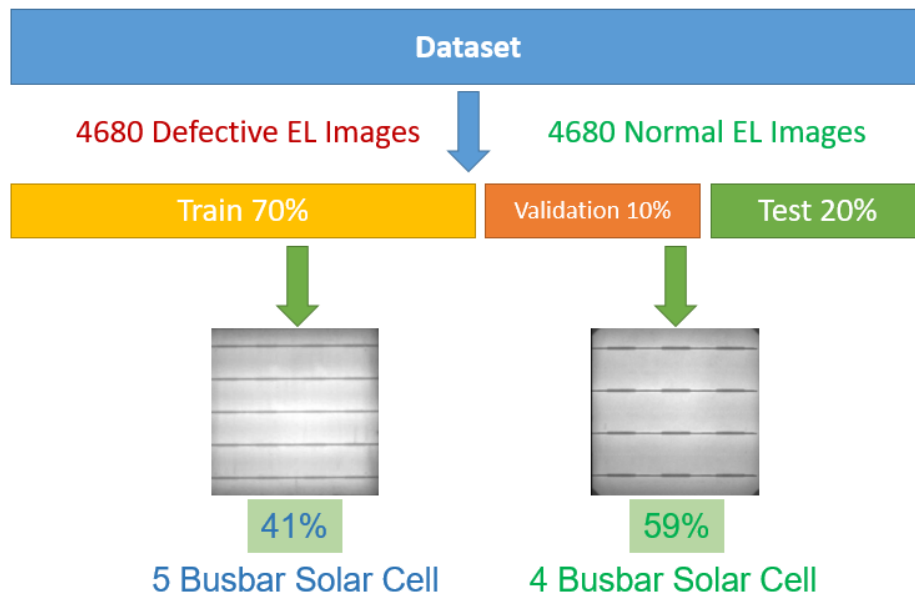


Figure 8. Dataset Features

d. Data Augmentation

Although there are enough samples in the data set, we used the data augmentation technique, as shown in Figure 9, to reduce the possible overfitting and to obtain more precise results. [17]

RandRotation: Range of rotation, in degrees, applied to the input image. RandRotation value was [90,90]

RandXReflection: Random reflection in the left-right direction, specified as a logical scalar. When RandXReflection is true (1), each image is reflected horizontally with a 50% probability. When RandXReflection is false (0), no images are reflected. RandXReflection value was true (1)

RandYReflection: Random reflection in the top-bottom direction, specified as a logical scalar. When RandYReflection is true (1), each image is reflected vertically with a 50% probability. When RandYReflection is false (0), no images are reflected. RandYReflection value was true (1)

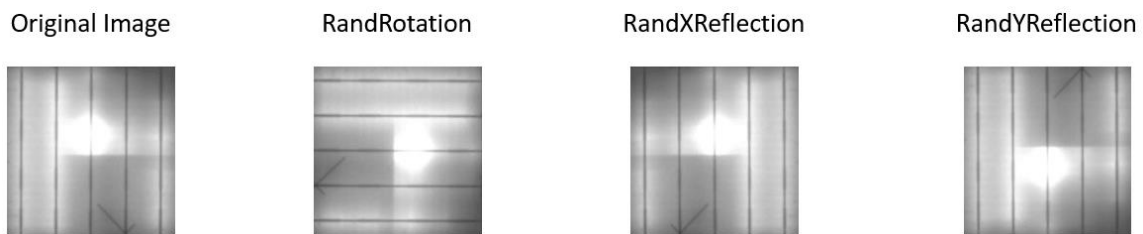


Figure 9. Data Augmentation Techniques

e. Deep Learning Models

As can be seen in Table 2, training and tests were carried out with 10 different deep learning models, apart from SVM and KNN classical artificial intelligence algorithms. Feature extraction for SVM and KNN algorithms was done by Alexnet. Darknet19 and Darknet53 models, used as backbones in YOLOv2 and YOLOv3 algorithms, which perform real-time object detection, are also used for classification. Features such as layer, depth and parameter of the models are given in the table.

Table 2. Features of Used DL Models [5]

Model	Input Size	Number of Layers	Number of Connections	Depth	Number Of Parameters	Top-1 Error rate	Top-5 Error rate
AlexNet	227x227	25	-	8	61m	36.7	15.4
VGG16	224x224	41	-	16	138m	25.6	8.1
VGG19	224x224	47	-	19	144m	25.5	8
GoogleNet	224x224	144	170	22	4m	-	6.67
Mobilenet-v2	224x224	154	164	53	3.47 m	28.2	9.0
ResNet50	224x224	177	192	50	25.6m	22.8	6.71
ResNet101	224x224	347	379	101	44.6m	21.75	6.05
Darknet-19	256x256	64	63	19	5.58 m	22.9	6.3
Darknet-53	256x256	184	206	53	40.5 m	22.8	6.2
DenseNet201	224x224	708	805	201	20m	21.46	5.54

Training and testing of models were done on the Matlab 2019b platform. Training options were selected as below;

SolverName: SGDM (Stochastic Gradient Descent with Momentum), MiniBatchSize:128, MaxEpochs:20, InitialLearnRate:1e-3, Shuffle:Every-epoch, ValidationFrequency:50, Verbose:False.

f. Evaluation Performance of Deep Learning Models

Each model was trained and tested with the same images. A confusion matrix was used for the evaluation of the performance of the DL models, and Accuracy, Sensitivity, Precision and F1-score metrics derived from this matrix were used for the final evaluation. The formulas for these metrics are below.

		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

TP: The number of defect cells which are defect in reality, that the DL model has found,

FP: The number of defect cells which are non-defect in reality, that the DL model has found,

FN: The number of non-defect cells which are defect in reality, that the DL model has found,

TN: The number of non-defect cells which are non-defect in reality, that the DL model has found,

i. Accuracy:

Accuracy gives the proportion of the total number of predictions that were correct as shown in Eq.1: [18]

$$Accuracy = \frac{(TP + TN)}{(TP + FN + FP + TN)} \quad (1)$$

ii. Sensitivity:

Sensitivity, recall, or the TP rate (TPR) is the fraction of positive values out of the total actual positive instances (i.e., the proportion of actual positive cases that are correctly identified) as shown in Eq.2.:

$$Sensitivity(Recall) = \frac{TP}{(TP + FN)} \quad (2)$$

iii. Precision:

Precision or the positive predictive value, is the fraction of positive values out of the total predicted positive instances. In other words, precision is the proportion of positive values that were correctly identified as shown in Eq.3:

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{3}$$

iv. F1 score:

The F1 score, F score, or F measure is the harmonic mean of precision and sensitivity. It gives importance to both factors as shown in Eq.4 [17]

$$F - \text{Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{4}$$

4. Results

In our study, since the training of the models was done on different computers and the features of these computers were different, a comparison was not made according to the working time, but the working time values are given in Table 3. As can be seen in the same table, the validation values obtained during the training were consistent with the test results in Table 4. The highest Validation Accuracy value was obtained with Mobilenet-v2 and Vg-19 models.

Table 3. Validation Accuracy and Runtime of DL models

Name	Validation Accuracy	Runtime (Min.)
Mobilenet-v2	0.9968	229.15
Darknet19	0.9957	239.12
Darknet53	0.9963	636.29
Alexnet	0.9893	20.14
Googlenet	0.9936	118.47
Vgg16	0.9909	52.8
Vgg19	0.9968	57.5
Resnet50	0.9936	29.24
Resnet101	0.9952	58.7
Densenet201	0.9963	636.29

The detailed test results of DL Models and Svm & Knn Classic AI Algorithms are listed in Table 4. There are the number of TP (True Positive), FP (False Positive), FN (False Negative), and TN (True Negative) predictions of the models and Precision, Sensitivity, Accuracy and F-score metrics which are derived from the predictions. The formula of these metrics is given under the “Evaluation Performance of Deep Learning Models” section.

The graphics of the training performed by the models with the dataset are shown in Figure 10. Each model's training and processing process according to the epoch is given.

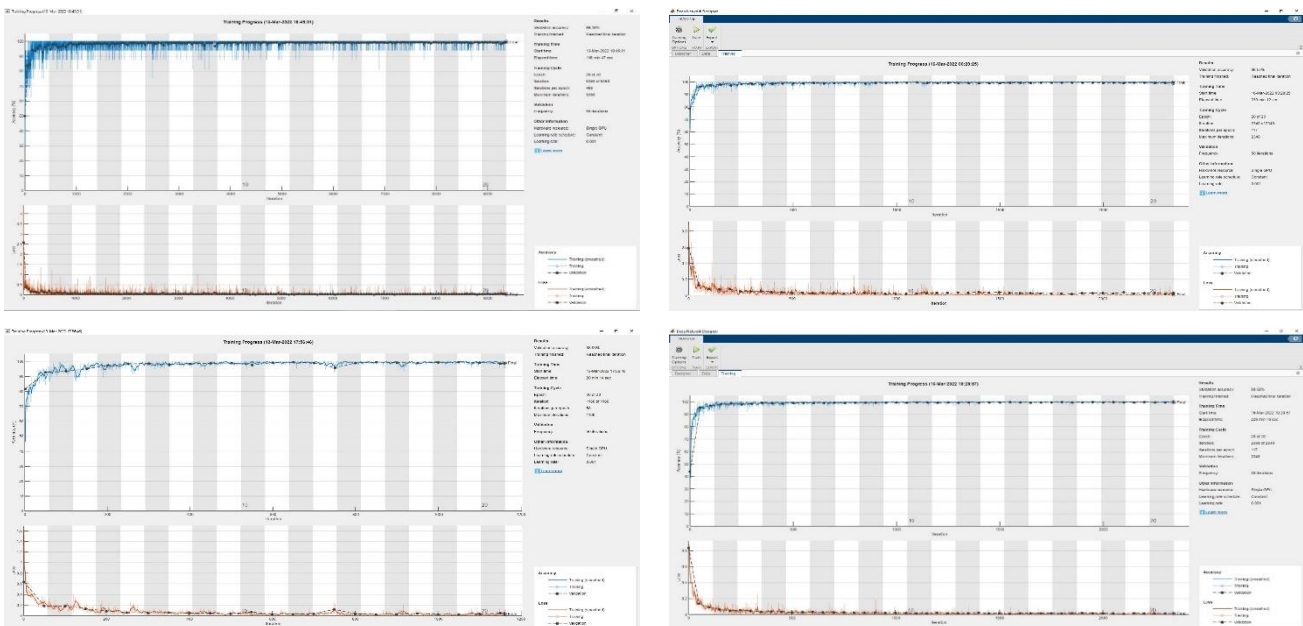


Figure 10. Training Deep Learning Models

According to the results in Table 4, Mobilenet-v2 and VGG-19 achieved the highest validation accuracy rate of 99.68%. According to F1 scores, Mobilenetv2 achieved the highest performance of 99.73%. It has been shown that all DL models show high performance for classification of defect and non-defect solar cells especially Mobilenet-v2 is more successful than other models in accuracy and F1-score by a very small margin.

Table 4. Performance of DL models & Classic AI Algorithms

Name	TP	FP	FN	TN	Precision	Sensitivity	Accuracy	F-score
Mobilenet-v2	934	3	2	933	0.9968	0.99786	0.99733	0.99733
Darknet-19	931	4	5	932	0.99572	0.99466	0.99519	0.99519
Darknet-53	930	1	6	935	0.99893	0.99359	0.99626	0.99625
Alexnet	930	15	6	921	0.98413	0.99359	0.98878	0.98884
Svm	910	9	26	927	0.99021	0.97222	0.9813	0.98113
Knn	834	13	102	923	0.98465	0.89103	0.93857	0.9355
Googlenet	926	4	10	932	0.9957	0.98932	0.99252	0.9925
Vgg16	927	3	9	933	0.99677	0.99038	0.99359	0.99357
Vgg19	935	7	1	929	0.99257	0.99893	0.99573	0.99574
Resnet50	925	3	11	933	0.99677	0.98825	0.99252	0.99249
Resnet101	931	1	5	935	0.99893	0.99466	0.99679	0.99679
Densenet201	932	2	4	934	0.99786	0.99573	0.99679	0.99679

Apart from these results, it is clearly understood that SVM and KNN algorithms perform quite well, of course, it is seen that SVM achieves higher scores than KNN.

5. Conclusion

In this study, for all performance parameters (Precision, Recall, F-score) we get the highest results among the previous studies. The reason for the higher performance than previous studies is thought to be the smaller data set sizes used in previous studies and the correct distinction between defect and non-defect cells. Another reason is that although the data set size is sufficient, high results were obtained due to data augmentation techniques. Another feature of this study compared to previous studies is the use of cells with different numbers of busbars in the data set. In this study both AI algorithms and deep learning models were tested and compared. High F-Score and Accuracy results of the DL models show defective & normal solar cells can be classified with deep learning successfully. For the next step defective cells in the Dataset will be separated according to their type then will be classified via the DL models and a new DL model which we will design and develop and compare its performance to pre-trained models. If the accuracy or F-score ratio is less than 90%, preprocessing methods will be implemented to improve the ratio.

6. Discussion

It is seen that deep learning models perform quite successfully in the classification of defects in solar cells. Replacing the classical image processing techniques used currently in EL devices with a defect classification system prepared with deep learning models will produce very economical results for solar energy panel manufacturers. The system to be integrated into EL devices will enable the process to be done automatically without requiring an expert. This once again reveals the importance and requirement of the study. The high performance rate obtained from the models may be due to the fact that the dataset consists of two classes. The results that will occur if the number of classes is large will be revealed in subsequent studies. Consisting of two classes containing defect and normal images constitute the boundaries of the study. In future studies, it is planned to work on datasets with a larger number of classes. The training process of deep learning models reveals their performance. However, it appears that it is not overfitting.

References

- [1] M. Abdelhamid, R. Singh, and M. Omar, "Review of microcrack detection techniques for silicon solar cells," *IEEE Journal of Photovoltaics*, vol. 4, no. 1, pp. 514–524, Jan. 2014. doi: 10.1109/JPHOTOV.2013.2285622.
- [2] K. G. Bedrich, "Quantitative electroluminescence measurements of PV devices," Loughborough University, 2017.
- [3] M. Aktas, F. Doğan, and İ. Türkoğlu, "Analysis Of Cracks In Photovoltaic Module Cells From Electroluminescence Images By Deep Learning.," in *1st International Conference on Computing and Machine Intelligence*, 2021, pp. 103–103.
- [4] F. Doğan and İ. Türkoğlu, "Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme," *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Derg.*, vol. 10, no. 2, pp. 409–445, 2019, doi: 10.24012/dumf.411130.
- [5] F. Doğan and I. Turkoğlu, "Comparison of deep learning models in terms of multiple object detection on satellite

- images,” *J. Eng. Res.*, Nov. 2021, doi: 10.36909/jer.12843.
- [6] J. Balzategui *et al.*, “Semi-automatic quality inspection of solar cell based on Convolutional Neural Networks,” in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2019, pp. 529–535. doi: 10.1109/ETFA.2019.8869359.
- [7] N. A.-A. Julen Balzategui, Luka Eciolaza, *2020 IEEE/SICE International Symposium on System Integration (SII)*. 2020.
- [8] A. S. Al-Waisy *et al.*, “Identifying defective solar cells in electroluminescence images using deep feature representations,” *PeerJ Comput. Sci.*, vol. 8, p. e992, 2022.
- [9] S.-H. Lee, L.-C. Yan, and C.-S. Yang, “LIRNet: A lightweight inception residual convolutional network for solar panel defect classification,” *Energies*, vol. 16, no. 5, p. 2112, 2023.
- [10] S. Deitsch *et al.*, “Automatic Classification of Defective Photovoltaic Module Cells in Electroluminescence Images,” Jul. 2018, doi: 10.1016/j.solener.2019.02.067.
- [11] M. W. Akram *et al.*, “CNN based automatic detection of photovoltaic cell defects in electroluminescence images,” *Energy*, vol. 189, Dec. 2019, doi: 10.1016/j.energy.2019.116319.
- [12] S. A. Anwar and M. Z. Abdullah, “Micro-crack detection of multicrystalline solar cells featuring an improved anisotropic diffusion filter and image segmentation technique,” *Eurasip J. Image Video Process.*, vol. 2014, 2014, doi: 10.1186/1687-5281-2014-15.
- [13] A. Bartler, L. Mauch, B. Yang, M. Reuter, and L. Stoicescu, “Automated Detection of Solar Cell Defects with Deep Learning,” in *2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, Sep. 2018, pp. 2035–2039. doi: 10.23919/EUSIPCO.2018.8553025.
- [14] J. Wang *et al.*, “Deep-Learning-Based Automatic Detection of Photovoltaic Cell Defects in Electroluminescence Images,” *Sensors*, vol. 23, no. 1, p. 297, Dec. 2022, doi: 10.3390/s23010297.
- [15] J. Wohlgemuth and W. Herrmann, “Hot spot tests for crystalline silicon modules,” in *Conference Record of the Thirty-first IEEE Photovoltaic Specialists Conference, 2005.*, 2005, pp. 1062–1063.
- [16] V. Tiwari and S. C. Jain, “An optimal feature selection method for histopathology tissue image classification using adaptive jaya algorithm,” *Evol. Intell.*, vol. 14, no. 3, pp. 1279–1292, Sep. 2021, doi: 10.1007/s12065-019-00205-w.
- [17] “ImageDataAugmenter.” Accessed: May 19, 2023. [Online]. Available: <https://www.mathworks.com/help/deeplearning/ref/imagedataaugmenter.html>
- [18] D. K. Sharma, M. Chatterjee, G. Kaur, and S. Vavilala, “Deep learning applications for disease diagnosis,” in *Deep Learning for Medical Applications with Unique Data*, Elsevier, 2022, pp. 31–51. doi: 10.1016/B978-0-12-824145-5.00005-8.

Author(s) Contributions

Miktat AKTAŞ: Conceived and designed the analysis, Collected the dataset, contributed data or analysis tools, Performed the analysis, Writing the paper

Ferdi DOĞAN: Contributed data or analysis tools, Performed the analysis, Writing the paper

İbrahim TÜRKOĞLU: Conceived and designed the analysis, contributed data or analysis tools, Performed the analysis

Acknowledgments

This study is funded by GTC Gunes Sanayi and Ticaret A.S. I am very grateful to CEO of GTC Gunes Sanayi ve Ticaret A.S, Ayşe Çiğdem BESEN for her permission to use required dataset and her support during my studies.

This study/work was presented in the 3rd International Conference on Photovoltaic Science and Technologies (PVCon2022) held on July 5-7, 2022 in Middle East Technical University (Ankara, Turkey)

This this study was supported by Scientific and Technological Research Council of Turkey (TUBITAK) under the Grant Number 123E697 The authors thank to TUBITAK for their supports

Conflict of Interest Notice

Miktat AKTAŞ, Ferdi DOĞAN and İbrahim TURKOĞLU declare that there is no conflict of interest regarding the publication of this paper.

Support/Supporting Organizations

This study is supported by GTC Gunes Sanayi and Ticaret A.S

Plagiarism Statement

This article has been scanned by iThenticate™.

Performance Assessment of Natural Survivor Method-Based Metaheuristic Optimizers in Global Optimization and Engineering Design Problems

Hüseyin Bakır¹ 

¹ Department of Electronics and Automation, Vocational School, Dogus University, Istanbul, Türkiye

Corresponding author:

Hüseyin Bakır, Department of Electronics and Automation, Vocational School, Dogus University, Istanbul, Türkiye
hbakir@dogus.edu.tr



Article History:

Received: 28.04.2024

Accepted: 29.05.2024

Published Online: 26.08.2024

ABSTRACT

This study presents the comparative performance analysis of Natural Survivor Method (NSM)-based algorithms in solving the IEEE CEC 2022 test suite benchmark problems and four real-world engineering design problems. Three different variants (Case1, Case2, Case3) of the NSM-TLABC, NSM-SFS and NSM-LSHADE-SPACMA algorithms were used in the study. The data obtained from the experimental studies were statistically analyzed using Friedman and Wilcoxon signed-rank tests. Based on the Friedman test results, NSM-LSHADE-SPACMA_Case2 showed the best performance with an average Friedman score of 3.96. The Wilcoxon signed-rank test showed that NSM-LSHADE-SPACMA_Case2 outperformed its competitors in 13 out of 16 experiments, achieving a success rate of 81.25%. NSM-LSHADE-SPACMA_Case2, which was found to be the most powerful of the NSM-based algorithms, is used to solve cantilever beam design, tension/compression spring design, pressure vessel design and gear train design problems. The optimization results are also compared with eight state-of-the-art metaheuristics, including Rime Optimization Algorithm (RIME), Nonlinear Marine Predator Algorithm (NMPA), Northern Goshawk Optimization (NGO), Kepler Optimization Algorithm (KOA), Honey Badger Algorithm (HBA), Artificial Gorilla Troops Optimizer (GTO), Exponential Distribution Optimization (EDO) and Hunger Games Search (HGS). Given that all results are together, it is seen that NSM-LSHADE-SPACMA_Case2 algorithm consistently produced the best results for the global and engineering design problems studied.

Keywords: Natural survivor method-based algorithms, Global optimization, IEEE CEC 2022 test suite, Real-world engineering design problems

1. Introduction

Metaheuristic optimization algorithms (MOAs) have a great reputation for solving global and engineering problems [1, 2]. The ability of MOAs to solve non-convex optimization problems has encouraged researchers to conduct further research on metaheuristic algorithms [3, 4]. In this direction, they have been focused on two topics. The first is the development of new metaheuristics. The second is the performance improvement of existing MOAs [5].

Metaheuristic algorithm design is an active area of research. To date, numerous MOAs have been developed inspired by events, methods, and natural processes [6]. Some examples of MOAs introduced in the last four years include Black Widow Optimization (BWO, 2020) [7], Mayfly Algorithm (MA, 2020) [8], Group Teaching Optimization Algorithm (GTOA, 2020) [9], Gradient-Based Optimizer (GBO, 2020) [10], Transient Search Optimization (TSO, 2020) [11], Chameleon Swarm Algorithm (CSA, 2021) [12], Horse herd Optimization Algorithm (HOA, 2021) [13], Capuchin Search Algorithm (CapSA, 2021) [14], Archimedes Optimization Algorithm (AOA, 2021) [15], Dwarf Mongoose Optimization (DMO, 2022) [16], Gannet Optimization Algorithm (GOA, 2022) [17], Red Fox Optimization (RFO, 2021) [18], Tasmanian Devil Optimization (TDO, 2022) [19], War Strategy Optimization (WSO, 2022) [20], Wild Horse Optimizer (WHO, 2022) [21], Coati Optimization Algorithm (COA, 2023) [22], Nutcracker Optimization Algorithm (NOA, 2023) [23], Meerkat Optimization Algorithm (MOA, 2023) [24], Energy Valley Optimizer (EVO, 2023) [25], and Growth Optimizer (GO, 2023) [26]. In the above-mentioned literature works, authors usually verified the performance of the developed algorithms using global optimization and real-world engineering problems. Particularly, optimizing constrained engineering problems from various disciplines has played an important role in testing the algorithm's performance. Rolling Element Bearing Design (REBD), Three-Bar Truss Design (TBTD), Welded Beam Design (WBD), Cantilever Beam Design (CBD), Disc Clutch Brake Design (DCBD), Speed Reducer Design (SRD), Pressure Vessel Design (PVD), Gear Train Design (GTD), and Tension/Compression Spring Design (TCSD) are among the commonly studied real-world problems. Agushaka et al. [27] developed a novel optimizer named the Gazelle Optimization Algorithm (GOA) and applied it to solve WBD, TCSD, PVD,

and SRD problems. The findings showed that GOA can produce optimal solutions to the real-world engineering problems addressed in the study. In another work, Kaveh et al. [28] showed that the Orchard Algorithm (OA) found better solutions to PVD, WBD, TCSD, and TBTD problems compared to CapSA, ChOA, BWO, PSO, and GA methods. Han et al. [29] performed the optimization of GTD, CBD, TCSD, PVD, WBD and SRD problems with the Walrus Optimizer (WO). Numerical results showed that WO successfully converges to the global optimum in real-world engineering problems under study. Zhu et al. [30] successfully applied the Human Memory Optimization (HMO) algorithm to find optimal solutions to TBTD, TCSD, and WBD problems. Optimization results illustrated that HMO is superior to competitive optimizers about solution accuracy. El-kenawy et al. [31] introduced a nature-inspired Greylag Goose Optimization (GGO) method and tested the solution ability of the algorithm by solving constrained PVD and TCSD problems. Given that all results are together, it is seen that GGO achieves consistently optimal solutions to real-world problems.

In recent years, studies on improving the performance of existing MOAs have been increasing. The reason behind this can be that it is possible to increase the robustness, solution accuracy, and convergence speed of metaheuristic optimizers using well-known strategies such as chaos maps, Levy flights, fitness-distance balance selection, opposition-based learning, etc. For example, Aydemir [32] applied the chaotic maps strategy to overcome getting stuck in local traps and poor convergence problems of the Arithmetic Optimization Algorithm (AOA). The solution ability of the Chaotic AOA (CAOA) has been evaluated on benchmark functions. The results showed that the CAOA effectively scanned the search space and converged to the best solutions. Ekinici et al. [33] developed a powerful variant of the Reptile Search Algorithm (RSA) to obtain a better optimization structure for solving challenging power system problems. The authors redesigned the exploration operator of the optimizer with the Levy flight and the exploitation operator with the Nelder–Mead simplex search approach. The results obtained from parameter extraction of the Power System Stabilizer (PSS), and the design of the Automatic Voltage Regulator (AVR) system demonstrated that the developed method could obtain better solutions compared to the original RSA. Bakır et al. [34] used the Fitness Distance Balance (FDB) strategy to enhance the convergence rate of the Levy Flight Distribution (LFD) optimizer. The proposed FDB-LFD has been tested on 39 benchmark problems as well as AVR optimization. The results show that the FDB approach provides a notable enhancement in the exploration ability of the original LFD. Zhong et al. [35] focused on enhancing the convergence rate of the Equilibrium Optimizer (EO). In this direction, the authors equipped the EO algorithm with evolutionary population dynamics, opposition-based learning, and Levy flight operators. Considering the experimental studies performed in search spaces of 100 to 5000 dimensions, it is observed that modified EO exhibits outstanding optimization performance.

The search process life cycle of metaheuristic optimization algorithms is generally divided into three phases. These are selection, search, and update [36]. The task of the selection phase is to determine the guide solution candidate/s that will direct the search process. During the search phase, exploration and exploitation tasks are fulfilled. The update phase determines which solution candidates will survive and which will be killed. As can be seen from the above-mentioned literature review, many of the studies regarding improving the performance of existing algorithms have focused on the first two stages (selection and search). As far as the authors know, the latest study related to the design of an update strategy was conducted by Kahraman et al. [6] in 2023. In the reference work, the authors developed a new strategy for updating solution candidates called the Natural Survivor Method (NSM). The introduced update strategy was applied to the base version of the Teaching-Learning-Based Artificial Bee Colony (TLABC) [37], Stochastic Fractal Search (SFS) [38], and LSHADE-SPACMA [39] algorithms, and three new algorithms named NSM-TLABC, NSM-SFS, and NSM-LSHADE-SPACMA were developed. Although the success of NSM-based algorithms in optimizing CEC 2017 and CEC 2020 benchmark problems has been investigated, their performance in the challenging CEC 2022 benchmark functions is a matter of curiosity. In this regard, this study has focused on the comparative performance analysis of NSM-based algorithms in optimizing the CEC 2022 benchmark problems. In addition, the algorithm with the best performance among the NSM-based algorithms is applied to solve real-world engineering problems.

The main contributions of the paper can be listed as follows:

- Optimization of twelve benchmark functions of the CEC 2022 test suite using NSM-based metaheuristics (NSM-TLABC, NSM-SFS, and NSM-LSHADE-SPACMA) for the first time.
- Non-parametric statistical analysis of NSM-based algorithms. Analysis results verified that the NSM-LSHADE-SPACMA_Case2 outperformed the compared ones.
- Application of NSM-LSHADE-SPACMA_Case2 algorithm to the solution of cantilever beam design, tension/compression spring design, pressure vessel design, and gear train design problems for the first time.
- Comparison of the results of NSM-LSHADE-SPACMA_Case2 on engineering optimization with eight optimization algorithms including RIME, NMPA, NGO, KOA, HBA, GTO, EDO, and HGS.

The remaining sections of the paper are designed in the following manner: Section 2 introduces the benchmark functions of the CEC 2022 test suite. Section 3 presents the optimization model of real-world engineering problems. In Section 4, the basics of NSM-based algorithms are presented. Section 5 evaluates the results of experimental studies. Finally, the findings of the present research are summarized in Section 6.

2. Benchmark Functions of the CEC 2022 Test Suite

The CEC 2022 [40] benchmark suite is an important resource for the optimization community. It provides a common platform to compare optimization algorithms on a set of challenging problems. The test suite consists of twelve benchmark functions in four categories. The summary of the CEC 2022 test suite is given in Table 1.

Table 1. Outline of CEC 2022 Test Suite [40]

Function Type	Function Number	Global Optimum	Function Type	Function Number	Global Optimum
Unimodal	F1	300	Hybrid	F6	1800
				F7	2000
				F8	2200
Multimodal	F2	400	Composition	F9	2300
	F3	600		F10	2400
	F4	800		F11	2600
	F5	900		F12	2700

As can be seen in Table 1, there is only one benchmark function with the unimodal type and that is F1. This problem measures the exploitation capability. There are multiple local optima solutions of the multimodal type F2, F3, F4, and F5 benchmark functions. Convergence to the global optimum in multimodal benchmark functions is a relatively challenging task. To overcome this, the algorithm must have strong exploration capability. In the CEC 2022 test suite, there are three hybrid (F6, F7, F8) and four composition (F9, F10, F11, F12) type benchmark functions. In both problem types, the convergence accuracy is directly related to successfully establishing the exploration-exploitation balance. Each benchmark function of the CEC 2022 is designed as a minimization problem. The search space boundaries for all benchmark functions are set to [-100, 100]^D. Detailed information on the CEC 2022 test suite problems can be found in Ref. [40].

3. Formulation of Engineering Problems

In the study, four engineering problems are optimized. The optimization model of engineering problems is introduced below.

3.1. Gear Train Design

In this problem, the objective is the minimization of gear ratio cost. Equation (1) gives the optimization model of the gear train design problem [41]. As shown in the equation there are four parameters to be optimized. These are teeth number of gearwheels: T_a, T_b, T_d, T_f . The pictorial representation of the problem is given in Figure 1 [41].

$$\begin{aligned}
 & \text{minimize } F_{obj,1}(\vec{x}) = \left(\frac{1}{6.931} - \frac{T_b \times T_d}{T_a \times T_f} \right)^2 \\
 & \text{variable vector } \vec{x} = [T_a, T_b, T_d, T_f] \\
 & \text{variable range: } 0.01 \leq T_a, T_b, T_d, T_f \leq 60
 \end{aligned} \tag{1}$$

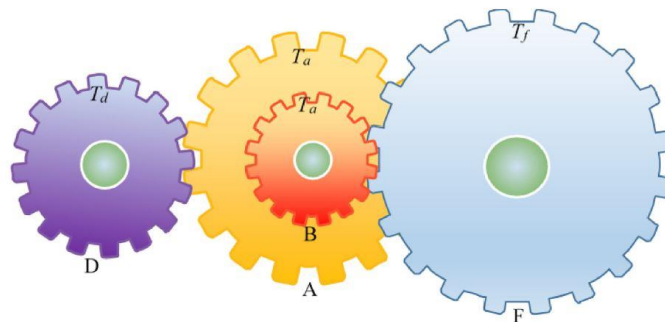


Figure 1. Gear Train Design [41]

3.2. Tension/Compression Spring Design

The primary task is to minimize the weight of the coil while satisfying various inequality constraints. As can be seen in the schematic diagram given in Figure 2, the problem includes three parameters to be optimized. These are the number of spring's active coils (N), the diameter of the winding (D), and the diameter of the wire (d). The optimization model of the problem can be written as follows [41]:

$$\text{minimize } F_{obj,2}(\vec{x}) = (x_3 + 2) x_2 x_1^2$$

$$\text{variable vector } \vec{x} = [x_1, x_2, x_3] = [d, D, N]$$

$$\text{subject to: } h_1(\vec{x}) = 1 - \frac{x_3 x_2^3}{71785 x_1^4} \leq 0$$

$$\text{subject to: } h_2(\vec{x}) = \frac{4 x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0$$

(2)

$$h_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0, \quad h_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

$$\text{variable range: } 0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15$$

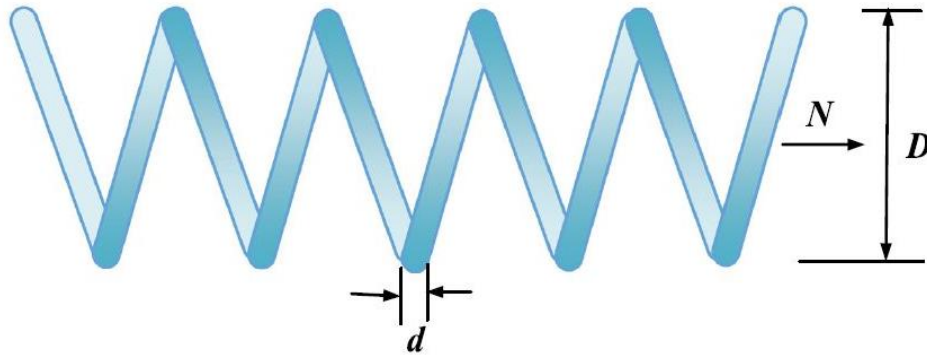


Figure 2. Tension/Compression Spring Design [41]

3.3. Pressure Vessel Design

Figure 3 illustrates the pictorial representation of the pressure vessel design. The objective of the problem is the minimization of fabrication cost through the optimal setting of the inner radius (R), the thickness of the head (T_h), length of the shell (L), and the thickness of the shell (T_s). The mathematical representation of the problem is given in Eq. (3) [41].

$$\text{minimize } F_{obj,3}(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3$$

$$\text{variable vector } \vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]$$

$$\text{subject to: } h_1(\vec{x}) = -x_1 + 0.0193 x_3 \leq 0$$

$$h_2(\vec{x}) = -x_2 + 0.00954 x_3 \leq 0$$

(3)

$$h_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3} \pi x_3^3 + 1296000 \leq 0$$

$$h_4(\vec{x}) = x_4 - 240 \leq 0$$

$$\text{variable range: } 0 \leq x_1, x_2 \leq 99, 10 \leq x_3, x_4 \leq 200$$

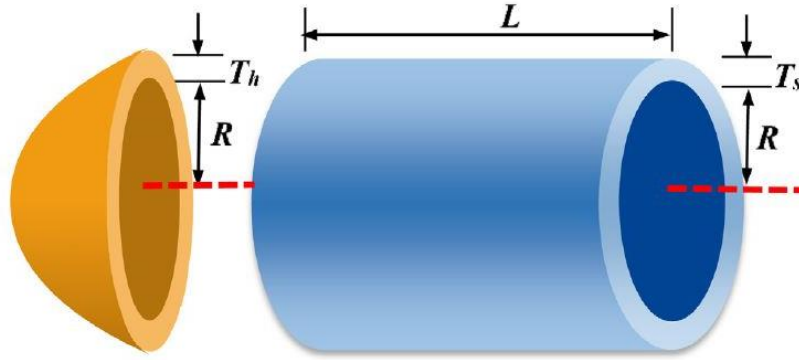


Figure 3. Pressure Vessel Design [41]

3.4. Cantilever Beam Design

This problem minimizes cantilever beam weight by optimizing five different block lengths. Figure (4) and Equation (4) give the schematic representation and mathematical model of the problem, respectively [41].

$$\text{minimize } F_{obj,4}(\vec{x}) = 0.0624 (x_1 + x_2 + x_3 + x_4 + x_5)$$

$$\text{variable vector } \vec{x} = [x_1, x_2, x_3, x_4, x_5]$$

$$\text{subject to: } h_1(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \tag{4}$$

$$\text{variable range: } 0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100$$

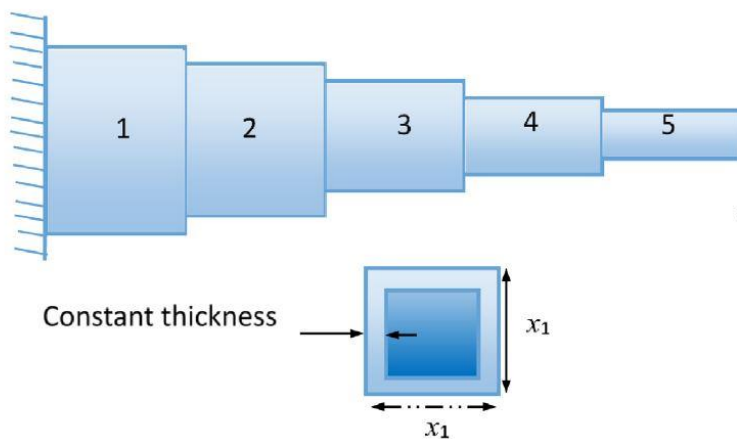


Figure 4. Cantilever Beam Design [41]

4. Natural Survivor Method-Based Metaheuristics

This section introduces the optimization framework of NSM-based algorithms. In this context, three subsections have been prepared. The following subsections provide detailed information about the NSM-TLABC, NSM-SFS and NSM-LSHADE-SPACMA algorithms, respectively.

4.1. NSM-TLABC

NSM-TLABC [6] is a powerful variant of the TLABC algorithm. The algorithm was created by redesigning the survivor selection of the original TLABC [37] using the NSM. NSM-TLABC initiates optimization by generating a random initial population. Then, it applies exploration and exploitation operators to find the global optimum solution. The pseudo-code of the NSM-TLABC algorithm is given in Algorithm 1.

Algorithm 1. NSM-TLABC Algorithm

1	Initialization
	Create initial population with defaults (population size N and number of design variables D)
2	$P \equiv \begin{bmatrix} p_{11} & \cdots & p_{1D} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{ND} \end{bmatrix}_{N \times D}$
3	for $i=1: N$ do
	Calculate objective function $f(p_i)$ and fitness value $Fit(p_i)$ for i -th individual
4	$Fit(p_i) = \begin{cases} \frac{1}{1 + f(p_i)} & \text{if } f(p_i) \geq 0 \\ 1 + f(p_i) & \text{if } f(p_i) < 0 \end{cases}$
5	Assign $trial(i)=0$ for each individual
6	end
7	repeat the following steps
8	<i>// Teaching-based employed bee //</i>
9	for $i=1: N$ do
	Create new solution using hybrid TLBO teaching strategy
10	$z_{i,d} = \begin{cases} p_{i,d}^{old} + r_2(p_{teacher,d} - T_F \times p_{mean,d}) & \text{if } r_1 < 0.5 \\ p_{R1,d} + F(p_{R2,d} - p_{R3,d}) & \text{else} \end{cases}$
11	Calculate objective function $f(z_i)$ and fitness value $Fit(z_i)$
	Apply NSM-score based survivor selection (NSM-TLABC_Case1) & (NSM-TLABC_Case3)
	if $z_{i_NSM_Score} > p_{i_NSM_Score}$
	survivor is z_i
12	else
	survivor is p_i
	end
13	if p_i does not enhance $trial(i)=trial(i)+1$, otherwise $trial(i)=0$
14	end
15	<i>// Learning-based onlooker bee //</i>
	Calculate selection probability
16	$sp_i = \frac{fit(p_i)}{\sum_{i=1}^{SN} fit(p_i)}$
17	for $i=1: N$ do
18	Apply the roulette selection approach and select p_k based on sp_i
	Generate new solution using TLBO learning strategy
19	$z_k = \begin{cases} p_k + rand(p_k - p_j) & \text{if } f(p_k) \leq f(p_j) \\ p_k + rand(p_j - p_k) & \text{if } f(p_j) > f(p_k) \end{cases}$
20	Calculate objective function $f(z_k)$ and fitness value $Fit(z_k)$
	Apply NSM-score based survivor selection-method (NSM-TLABC_Case2)
	if $z_{k_NSM_Score} > p_{k_NSM_Score}$
	survivor is z_k
21	else
	survivor is p_k
	end
22	if p_k does not enhance $trial(k)=trial(k)+1$, otherwise $trial(k)=0$ end
23	end
24	<i>// Generalized oppositional scout bee //</i>
25	if $\max(trial(i)) \geq \text{limit}$
26	Generate new solution p_i (randomly) and its generalized oppositional solution p_i^{op}
	Choose better one between p_i and p_i^{op}
27	$p_i = \begin{cases} p_i & \text{if } f(p_i) \leq f(p_i^{op}) \\ p_i^{op} & \text{otherwise} \end{cases}$
28	end
29	until the termination criterion is met
30	Display best solution (p_{best}) achieved so far

4.2. NSM-SFS

NSM-SFS [6] is a nature-inspired metaheuristic optimization technique developed by configuring of the original SFS [38] using the NSM update mechanism. The algorithm derives its power from updating the solutions in the population according to the NSM score. The NSM score is a more accurate measure of the fitness of a solution than the traditional fitness value. This allows the algorithm to focus on finding solutions that are both good and diverse, which can help to prevent premature convergence. NSM-SFS generates a random initial population and enhances the quality of existing solutions in the population with the diffusing and updating processes during the search process lifecycle. Algorithm 2 presents the pseudocode of the NSM-SFS algorithm.

Algorithm 2. NSM-SFS Algorithm

1	Initialization
	Create an initial population and calculate the fitness value of each point
2	$P \equiv \begin{bmatrix} p_{11} & \cdots & p_{1D} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{ND} \end{bmatrix}_{N \times D}, f = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}_{N \times 1}$
3	Sort the population based on the fitness value and determine the best point in P (p_{best})
4	repeat the following steps
5	<i>// Diffusion process //</i>
6	for $i=1: N$ do
7	Select two points from P randomly and select the i -th point with the ordinal-based method
8	for $j=1: m$ (maximum diffusion number)
9	Create a new point using the Gaussian Walk
10	if $\text{rand} < (\text{walk}=1)$
11	$p_{i_{new,1}} = \text{Gaussian}(\mu_{p_{best}}, \sigma) + (\varepsilon \times p_{best} - \varepsilon' p_i)$
12	else
13	$p_{i_{new,2}} = \text{Gaussian}(\mu_p, \sigma)$, where $\mu_p = p_i $
14	end
15	end
16	end
17	<i>// First updating process //</i>
	Rank all points in P based on fitness function value
18	$PBa_i = \frac{\text{rank}(p_i)}{N}, i = 1, \dots, N$
19	for $i=1: N$ do
20	for $j=1: D$ do
21	if $\text{rand}[0,1] \geq PBa_i$
22	$p_i'(j) = p_r(j) - \varepsilon(p_t(j) - p_i(j))$
23	end
24	end
25	end
	Apply NSM-score based survivor selection (NSM-SFS_Case1, NSM-SFS_Case2, NSM-SFS_Case3)
26	if $p_i'_{NSM_Score} > p_i_{NSM_Score}$ survivor is p_i'
	else survivor is p_i
	end
27	<i>// Second updating process //</i>
	Rank all points obtained from first update process based on the fitness value
28	$PBa_i' = \frac{\text{rank}(p_i)}{N}, i = 1, \dots, N$
29	for $i=1: N$ do
30	if $\text{rand}[0,1] \geq PBa_i$
31	$p_i'' = p_i' - \hat{\varepsilon} \times (p_t' - p_{best})$ if $\varepsilon' \leq 0.5$
32	$p_i'' = p_i' + \hat{\varepsilon} \times (p_t' - p_r')$ if $\varepsilon' > 0.5$
33	end
34	end
35	until the termination criterion is met
36	Display best solution (p_{best}) achieved so far

4.3. NSM-LSHADE-SPACMA

NSM-LSHADE-SPACMA [6] is an effective and powerful algorithm developed to increase the optimization performance of LSHADE-SPACMA [39]. The optimizer uses the NSM approach to perform population management. In other words, NSM-LSHADE-SPACMA determines which individuals in the population will survive and which will be killed with an NSM score value. Algorithm 3 presents the steps followed by the NSM-LSHADE-SPACMA algorithm in the optimization process.

Algorithm 3. NSM-LSHADE-SPACMA Algorithm

1	Set function evaluation counter ($FES=0$), generation counter ($g=1$), $M_{CR}=0.5$, $M_F=0.5$, and $M_{FCP}=0.5$
2	Archive $A=\emptyset$, $N_g=N_{init}$
3	Initialize population $P_g=(x_{1,g}, \dots, x_{N,g})$ and covariance matrix adaptation (CMA) parameters
4	while $FES < \max FES$
5	$S_{CR} = \emptyset$, $S_F = \emptyset$, and $S_{FCP} = \emptyset$
6	for $i=1: N$ do
7	r_i =select randomly from $[1, H]$
8	$CR_{i,g} = \text{randn}_1(M_{CR_{r_i}}, 0.1)$
9	$FCP_{i,g} = M_{FCP_{r_i}}$
10	if $FES < (\max FES/2)$
11	$F_{i,g} = 0.45 + (0.1 \times \text{rand})$
12	else
13	$F_{i,g} = \text{randc}_1 + (M_{F_{r_i}}, 0.1)$
14	end
15	end
16	$[P_{LSHADE,g}, P_{CMA,g}] = \text{Split}(P_g, FCP_g)$
17	$V_{g,LSHADE} = \text{Generate donor vectors using LSHADE}$
18	$V_{g,CMA} = \text{Generate donor vectors using CMA}$
19	$V_g = \text{Concatenate}(V_{g,LSHADE}, V_{g,CMA})$
20	$U_g = \text{Generate trial vectors}(V_g, CR_g)$
21	Evaluate U_g and update FEs
22	Update P_g using NSM approach
23	Store successful FCP_g , F_g , and CR_g
24	Update archive A
25	if (archive size) > A
26	delete randomly selected archive individuals
27	end
28	Update memory M_{CR} , M_{FCP}
29	if $FES > (\max FES/2)$
30	update memory M_F
31	end
32	$N_{g+1} = \text{round} \left[\left(\frac{N_{\min} - N_{\text{init}}}{\max FES} \right) \times FES + N_{\text{init}} \right]$
33	if $N_g < N_{g+1}$
34	sort solutions based on fitness value and delete the lowest $N_g - N_{g+1}$ members
35	resize archive size A based on new P
36	end
37	Update CMA parameters
38	$g++$
39	end
40	Display the best solution achieved so far

5. Results and Discussions

This section summarizes the results of experimental studies conducted to evaluate the performance of NSM-based optimizers. In this direction, two subsections have been prepared. The first subsection presents the optimization results of the CEC 2022 global optimization problems. The second subsection elaborated on the results of four engineering problems.

5.1. Performance Analysis of NSM-based Metaheuristics on CEC 2022 Benchmark Functions

This subsection gives the optimization results of CEC 2022 benchmark functions obtained with NSM-TLABC, NSM-SFS, and NSM-LSHADE-SPACMA algorithms. The experimental studies used the source codes of the NSM-based algorithms, which were made available by the developers on the Matlab File Exchange platform ([Click for source codes of NSM-based algorithms](#)). Three versions of each algorithm are used and named as follows: NSM-TLABC_Case1, NSM-TLABC_Case2, NSM-TLABC_Case3, NSM-SFS_Case1, NSM-SFS_Case2, NSM-SFS_Case3, NSM-LSHADE-SPACMA_Case1, NSM-LSHADE-SPACMA_Case2, and NSM-LSHADE-SPACMA_Case3. The maximum number of fitness evaluations (maxFEs) was adopted as a termination criterion. The value of maxFEs is 200,000 and 1,000,000 for the 10 and 20 dimensional optimization of IEEE CEC 2022 benchmark problems, respectively. 6,480 (9 algorithms \times 12 functions \times 30 runs \times 2 dimensions) data items are obtained from CEC 2022 experiments and used for statistical analysis.

This work uses the Friedman-rank test [42] to compare the performance of more than two algorithms. Friedman test results of NSM-based algorithms are given in Table 2. As can be seen in the table, the overall algorithm performance was determined based on the “*Mean rank*” value. In the table, a small Friedman score indicates the superior performance of the algorithm. Considering the 10-dimensional experiment, the NSM-LSHADE-SPACMA_Case2 algorithm exhibits the best result with a Friedman score of 4.30, while NSM-TLABC_Case1 gives the worst result with a Friedman score of 6.17. In 20-dimensional optimization, NSM-LSHADE-SPACMA versions performed better than all other compared algorithms. Among these versions, the NSM-LSHADE-SPACMA_Case1 comes to the fore with a Friedman score 3.58. Given that all experiments are together, it is observed that the NSM-LSHADE-SPACMA_Case2 algorithm ranked first with a mean Friedman score of 3.96.

Table 2. Friedman Test Results of NSM-Based Algorithms

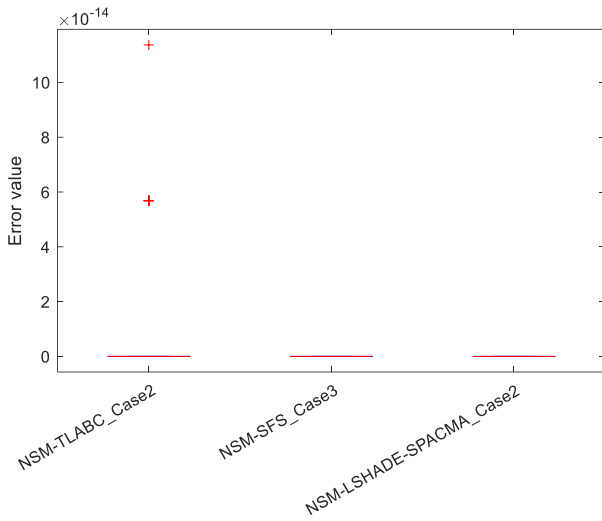
Algorithm	Dimension=10	Dimension=20	Mean rank	Overall rank
NSM-TLABC_Case1	6.17	6.73	6.45	9
NSM-TLABC_Case2	5.00	6.26	5.63	7
NSM-TLABC_Case3	5.72	6.62	6.17	8
NSM-SFS_Case1	5.34	4.57	4.95	6
NSM-SFS_Case2	4.86	4.68	4.77	5
NSM-SFS_Case3	4.44	5.08	4.76	4
NSM-LSHADE-SPACMA_Case1	4.46	3.58	4.02	2
NSM-LSHADE-SPACMA_Case2	4.30	3.62	3.96	1
NSM-LSHADE-SPACMA_Case3	4.66	3.82	4.24	3

Wilcoxon test [43] is applied for pairwise comparison between NSM-LSHADE-SPACMA_Case2 and competitive algorithms. Table 3 gives the Wilcoxon test results. The reason behind selecting the NSM-LSHADE-SPACMA_Case2 algorithm is that it is the winner of the Friedman test. As can be seen in Table 3, there are three scores in each cell. The first score shows the number of benchmark functions in which NSM-LSHADE-SPACMA_Case2 obtained better results than its rival. The second score indicates the number of problems for which the two algorithms produced similar results. The third score represents the number of benchmark functions where the competing algorithm is better than NSM-LSHADE-SPACMA_Case2. For example, in the 10-dimensional experiment between NSM-LSHADE-SPACMA_Case2 vs. NSM-TLABC_Case1, the Wilcoxon score is 8/3/1. In other words, the performance of NSM-LSHADE-SPACMA_Case2 is better in 8 test functions, while the result of the NSM-TLABC_Case1 algorithm is better in 1 function. In the 3 test functions, the algorithms could not outperform each other. In 13 out of 16 experiments, the NSM-LSHADE-SPACMA_Case2 outperformed the rival algorithm. The only exception is the 10-dimensional experiment between NSM-LSHADE-SPACMA_Case2 vs. NSM-SFS_Case3. To put it more clearly, the NSM-LSHADE-SPACMA_Case2 algorithm was only defeated by the NSM-SFS_Case3 algorithm with a score of 3/5/4.

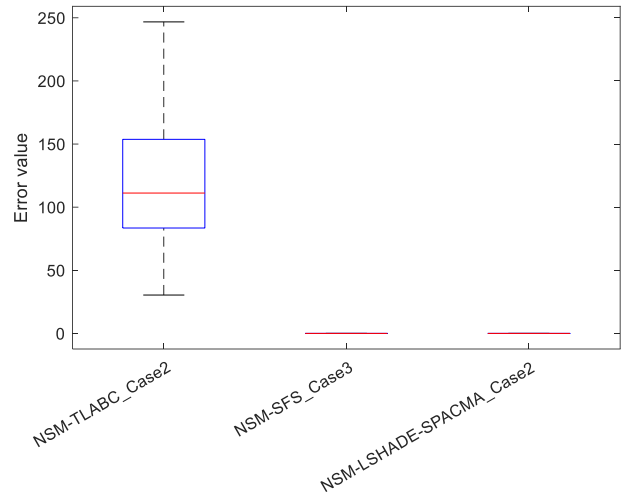
Table 3. Wilcoxon Pairwise Comparison Results

NSM-LSHADE-SPACMA_Case2 vs.	Dimension=10	Dimension=20
NSM-TLABC_Case1	8/3/1	11/1/0
NSM-TLABC_Case2	6/4/2	11/1/0
NSM-TLABC_Case3	8/3/1	11/1/0
NSM-SFS_Case1	6/3/3	5/5/2
NSM-SFS_Case2	5/4/3	6/4/2
NSM-SFS_Case3	3/5/4	6/4/2
NSM-LSHADE-SPACMA_Case1	1/10/1	3/7/2
NSM-LSHADE-SPACMA_Case3	3/8/1	1/10/1

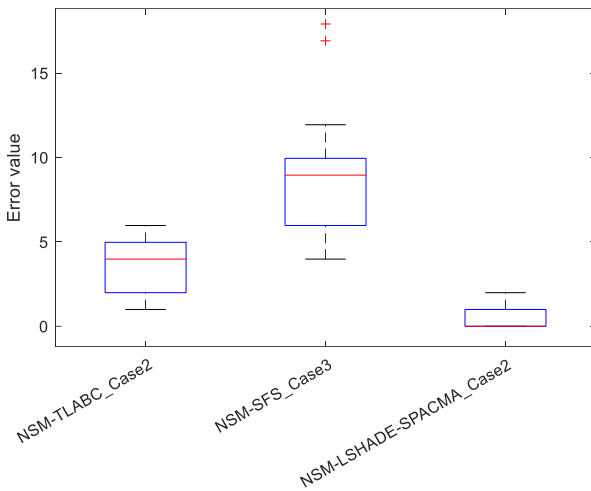
Box plots were prepared to observe the error statistics of the algorithms throughout 30 runs in 10- and 20-dimensional optimization of selected benchmark functions from CEC 2022. Figure 5 shows the box plots of F1, F4, F8, and F12 functions. In this figure, the algorithms NSM-TLABC_Case2, NSM-SFS_Case3 and NSM-LSHADE-SPACMA_Case2 represent the versions that give the best score according to the Friedman test (see Table 2).



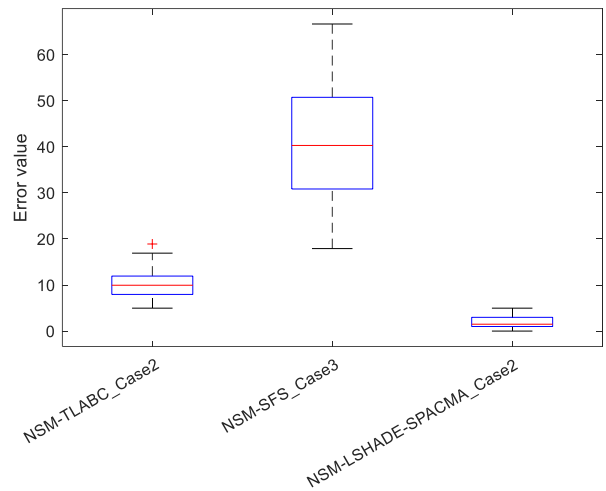
(a) F1 (Unimodal) D = 10



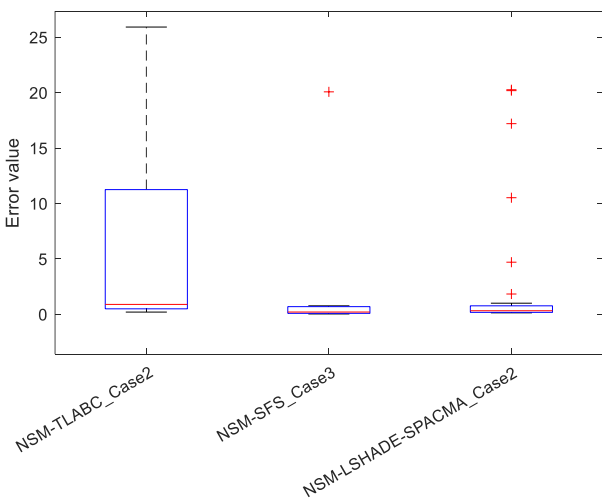
(b) F1 (Unimodal) D = 20



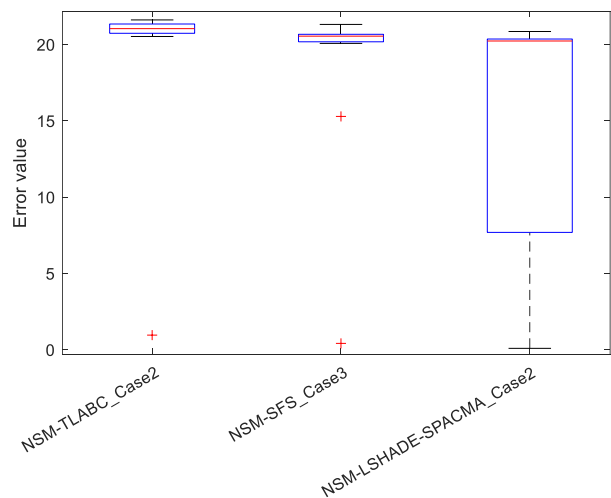
(c) F4 (Multimodal) D = 10



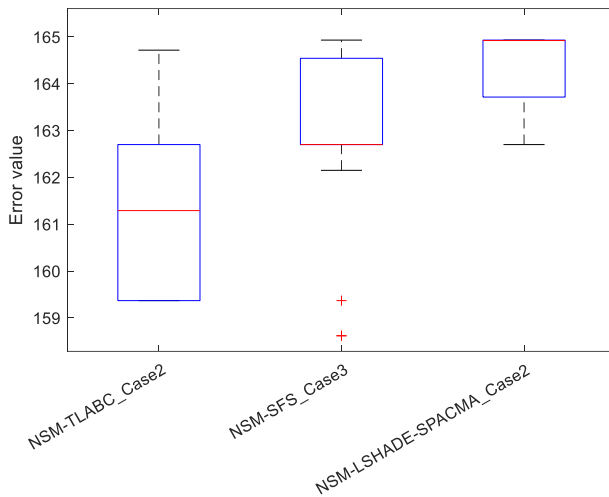
(d) F4 (Multimodal) D = 20



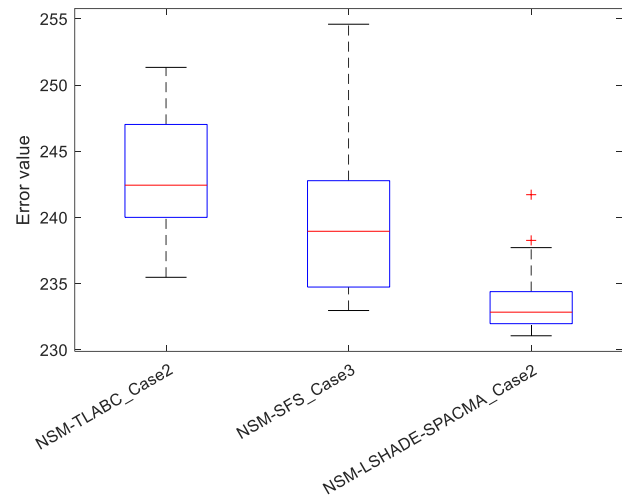
(e) F8 (Hybrid) D = 10



(f) F8 (Hybrid) D = 20



(g) F12 (Composition) D = 10



(h) F12 (Composition) D = 20

Figure 5. Box Plots of Best Versions of NSM-Based Algorithms on CEC 2022 Benchmark Functions

Upon examination of the box plots for the F1 unimodal problem, it is seen that NSM-SFS_Case3 and NSM-LSHADE-SPACMA_Case2 successfully converged to the global optimum in both problem sizes. Although NSM-TLABC_Case2 exhibits a remarkable search performance in 10 dimensions, the algorithm's performance is poor in 20 dimensions. The underlying reason behind it is that the algorithm cannot successfully fulfill the exploitation task in high-dimensional optimization. The box plots of the F4 multimodal problem show that the exploration ability of the NSM-LSHADE-SPACMA_Case2 is superior compared to the other two algorithms. On the other hand, NSM-SFS_Case3 gets stuck in local solution traps and converges prematurely. Upon examination of Figure 5e, it is seen that NSM-SFS_Case3 and NSM-LSHADE-SPACMA_Case2 algorithms have an overwhelming superiority over NSM-TLABC_Case2 in the 10-dimensional optimization of the hybrid F8 problem. In the 20-dimensional optimization of the same problem (Fig 5f), NSM-LSHADE-SPACMA_Case2 converges to a lower error value. From the box plots of the F12 composition problem (Figure 5g, h), it can be observed that NSM-TLABC_Case2 (in Dimension=10) and NSM-LSHADE-SPACMA_Case2 (in Dimension=20) balanced exploration and exploitation more successfully than competitors.

In a nutshell, this section presents the comparative performance analysis of NSM-based metaheuristic algorithms (NSM-TLABC, NSM-SFS, and NSM-LSHADE-SPACMA) in solving CEC 2022 benchmark problems. Given that the statistical analysis results and box plots are together, it is observed that the NSM-LSHADE-SPACMA_Case2 exhibits better optimization capability compared to other ones.

5.2. Application of NSM-LSHADE-SPACMA to Engineering Design Problems

In this subsection, NSM-LSHADE-SPACMA_Case2, which is determined as the most powerful of NSM-based algorithms, was applied to the solution of gear train design, tension/compression spring design, pressure vessel design, and cantilever beam design problems. Also, the performance of NSM-LSHADE-SPACMA_Case2 is compared with Rime Optimization Algorithm (RIME) [44], Nonlinear Marine Predator Algorithm (NMPA) [45], Northern Goshawk Optimization (NGO) [46], Kepler Optimization Algorithm (KOA) [47], Honey Badger Algorithm (HBA) [5], Artificial Gorilla Troops Optimizer (GTO) [48], Exponential Distribution Optimization (EDO) [49], and Hunger Games Search (HGS) [50]. The algorithms were run with the settings given in their original articles. The iteration number is 1000 for all algorithms.

▪ Gear train design

The optimized teeth numbers of gearwheels and the corresponding gear ratio cost are presented in Table 4. As per the results in the table, NSM-LSHADE-SPACMA_Case2, NMPA, NGO, HBA, GTO, and EDO algorithms obtained the best cost result with a value of 2.7009E-12. They are followed by RIME and KOA algorithms. The HGS algorithm gives the worst result (8.8876E-10). Figure 6 illustrates the convergence curves of optimizers in the gear train design problem. As is evident from the figure, NSM-LSHADE-SPACMA_Case2 reached the global optimum faster than the other algorithms.

Table 4. Optimization Results of Gear Train Design Problem

Algorithms	Optimized parameters				Optimal cost
	T_a	T_b	T_d	T_f	
NSM-LSHADE-SPACMA_Case2	49	19	16	43	2.7009E-12
RIME	34	13	20	53	2.3078E-11
NMPA	43	19	16	49	2.7009E-12
NGO	43	16	19	49	2.7009E-12
KOA	34	20	13	53	2.3078E-11
HBA	43	16	19	49	2.7009E-12
GTO	43	16	19	49	2.7009E-12
EDO	49	19	16	43	2.7009E-12
HGS	57	37	12	54	8.8876E-10

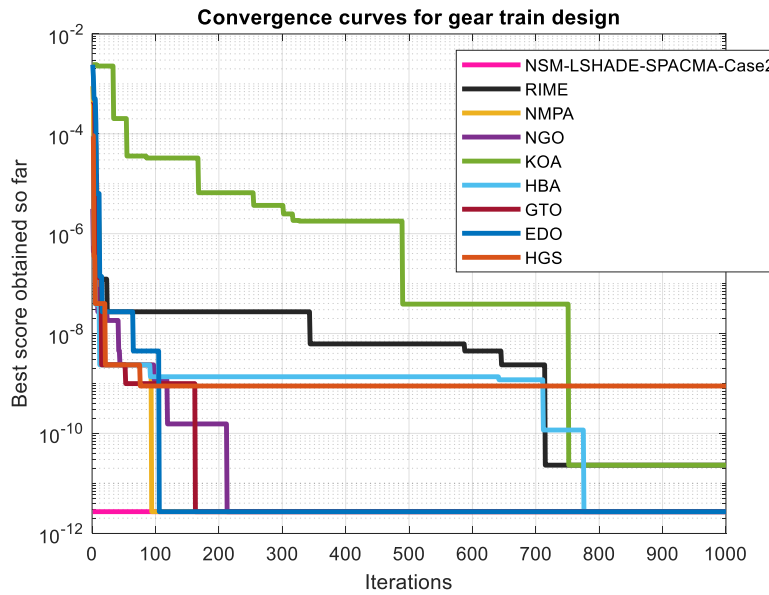


Figure 6. Convergence Curves of Metaheuristic Optimizers in Gear Train Design Problem

- Tension/compression spring design

The best settings of control variables and corresponding objective function results are depicted in Table 5. As per the results in the table, NSM-LSHADE-SPACMA_Case2, NMPA, NGO, HBA, and GTO algorithms obtained minimum coil weight with a value of 0.01266602. KOA gives the highest coil weight. The convergence behaviour of the competitive optimizer is depicted in Figure 7. From the convergence curves, it is observed that the NSM-LSHADE-SPACMA_Case2 successfully converged the best solution and reduced the iteration number. On the other hand, RIME and KOA methods get caught in local solution traps and converge prematurely.

Table 5. Optimization Results of Tension/Compression Spring Design Problem

Algorithms	Optimized parameters			Optimal weight
	d	D	N	
NSM-LSHADE-SPACMA_Case2	0.051897	0.361748	11.135056	0.01266602
RIME	0.051937	0.362229	11.313641	0.01270275
NMPA	0.051897	0.361748	10.729495	0.01266602
NGO	0.051897	0.361748	10.754378	0.01266602
KOA	0.050000	0.311346	15.000000	0.01323221
HBA	0.051897	0.361748	10.500000	0.01266602
GTO	0.051897	0.361748	11.415919	0.01266602
EDO	0.051897	0.361755	11.225818	0.01266653
HGS	0.051204	0.345163	11.685449	0.01266962

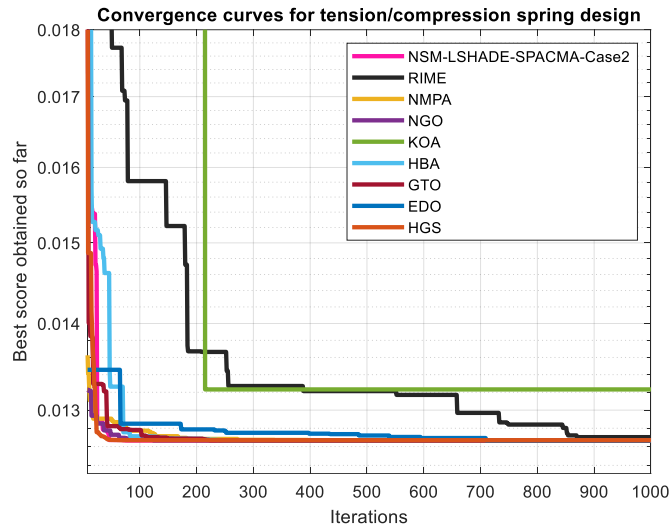


Figure 7. Convergence Curves of Metaheuristic Optimizers in Tension/Compression Spring Design Problem

▪ Pressure vessel design

The optimum parameters of the pressure vessel design problem and corresponding fabrication costs are given in Table 6. What is clear from the simulation results is that the best cost result (5885,332773) is achieved with NSM-LSHADE-SPACMA_Case2, HBA, and HGS algorithms. The respective decision variables are found to be 0.778168 for T_s , 0.384649 for T_h , 40.319618 for R , and 200 for L . The numeric results of the NMPA, NGO, and GTO algorithms are remarkable and close to each other. Figure 8 depicts the comparative convergence curves of optimizers. The illustrated charts in the figure show that the HGS algorithm exhibits a better convergence speed. Admittedly, the convergence ability of the HBA, NGO, and NSM-LSHADE-SPACMA_Case2 algorithms was also impressive.

Table 6. Optimization Results of Pressure Vessel Design Problem

Algorithms	Optimized parameters				Optimal cost
	T_s	T_h	R	L	
NSM-LSHADE-SPACMA_Case2	0.778168	0.384649	40.319618	200	5885,332773
RIME	0.888095	0.445929	45.981860	133.819439	6131,386806
NMPA	0.778168	0.384649	40.319618	199.999996	5885,332786
NGO	0.778168	0.384649	40.319618	199.999999	5885,332775
KOA	1.157765	0.573756	47.995036	145.907363	9291,834612
HBA	0.778168	0.384649	40.319618	200	5885,332773
GTO	0.778168	0.384649	40.319619	199.999985	5885,332808
EDO	0.778720	0.385479	40.333025	199.893895	5891,657726
HGS	0.778168	0.384649	40.319618	200	5885,332773

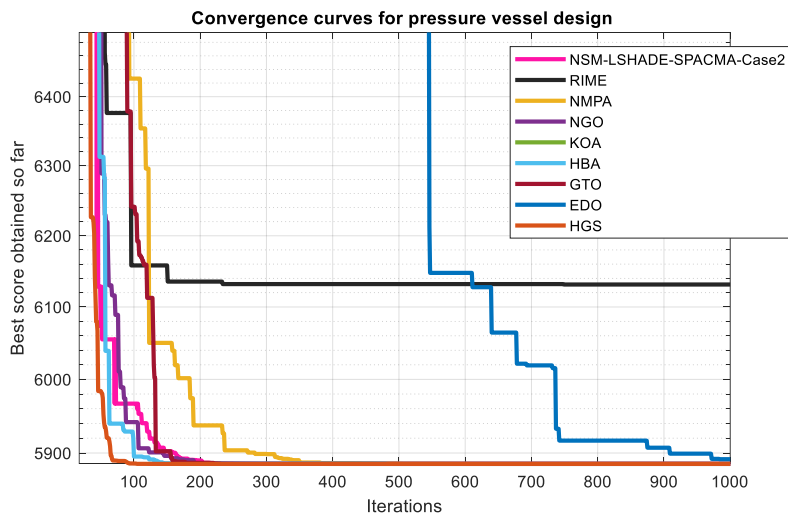


Figure 8. Convergence Curves of Metaheuristic Optimizers in Pressure Vessel Design Problem

▪ Cantilever beam design

Table 7 gives the optimized block lengths and corresponding cantilever beam weights. As per the results in the table, NSM-LSHADE-SPACMA_Case2, NMPA, and NGO algorithms obtained the optimal weight, and then followed by HBA and GTO algorithms. KOA algorithm gives the worst result. The reason behind the poor performance of the KOA can be that it gets stuck in local solution traps and convergence prematurely. Figure 9 depicts the change in the objective function over the iterations. From the figure, it can be seen that NSM-LSHADE-SPACMA_Case2 rapidly converged to the optimal weight result after 100 iterations.

Table 7. Optimization Results of Cantilever Beam Design Problem

Algorithms	Optimized parameters					Optimal weight
	x_1	x_2	x_3	x_4	x_5	
NSM-LSHADE-SPACMA_Case2	6.016015	5.309173	4.494329	3.501474	2.152665	1.339956
RIME	6.194038	5.423264	4.313638	3.502088	2.076832	1.342215
NMPA	6.016014	5.309174	4.494329	3.501474	2.152665	1.339956
NGO	6.016679	5.310302	4.493161	3.502687	2.150836	1.339956
KOA	6.990157	15.187510	5.099891	3.831937	1.850069	2.056676
HBA	6.018072	5.305518	4.497259	3.499427	2.153394	1.339957
GTO	6.015852	5.308207	4.494019	3.505487	2.150108	1.339957
EDO	5.984261	5.335545	4.493889	3.450257	2.219157	1.340546
HGS	6.028996	5.297238	4.480894	3.506450	2.160339	1.339972

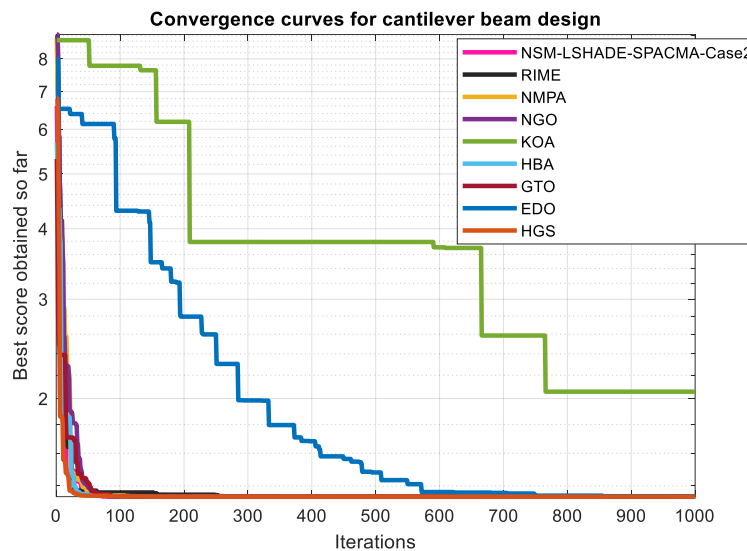


Figure 9. Convergence Curves of Metaheuristic Optimizers in Cantilever Beam Design Problem

6. Conclusions

This paper assesses the optimization ability of NSM-based metaheuristic algorithms in solving global optimization and constrained real-world engineering design problems. Firstly, NSM-TLABC, NSM-SFS, and NSM-LSHADE-SPACMA methods are applied to the optimization of CEC 2022 benchmark functions. The NSM-based algorithm that performs best in the global optimization is then used to find optimal solutions for the gear train design, tension/compression spring design, pressure vessel design and cantilever beam design problems. The results are compared with RIME, NMPA, NGO, KOA, HBA, GTO, EDO, and HGS algorithms. Based on the extensive experimental studies, the conclusions of the present research can be summarized as follows:

- Considering the Friedman test results, it is noticed that the best-performing variants of NSM-based algorithms are NSM-TLABC_Case2, NSM-SFS_Case3, and NSM-LSHADE-SPACMA_Case2 in the IEEE CEC 2022 global optimization problems.
- Among all NSM-based algorithms, NSM-LSHADE-SPACMA_Case2 stands out as the best optimiser with an average rank of 3.96 Friedman score.
- The winner of the Wilcoxon test is NSM-LSHADE-SPACMA_Case2. The algorithm gave statistically better results in 13 of 16 experiments. In other words, the NSM-LSHADE-SPACMA_Case2 showed a success rate of 81.25% against its competitors.

- Box plot analysis shows that the NSM-LSHADE-SPACMA algorithm exhibits better exploration and exploitation-exploration balance compared to other ones.
- The best objective function results are calculated with the NSM-LSHADE-SPACMA algorithm for gear train design, tension/compression spring design, pressure vessel design, and cantilever beam design problems as 2.7009E-12, 0.01266602, 5885,332773 and 1.339956, respectively.

In conclusion, this study strongly reports that NSM-LSHADE-SPACMA_Case2 performs better for global and engineering design problems. The strength of the NSM-based algorithm is that it performs population management using the NSM strategy. Based on this, it can be said that it is possible to increase the optimisation capacity of other population-based metaheuristic algorithms by hybridising them with NSM. In this way, high-quality solutions can be achieved for non-convex real-world engineering design problems.

References

- [1] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, 107250, 2021.
- [2] O. Altay, "Chaotic slime mould optimization algorithm for global optimization," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3979-4040, 2022.
- [3] P. Civicioglu and E. Besdok, "Bezier Search Differential evolution algorithm for numerical function optimization: A comparative study with CRMLSP, MVO, WA, SHADE and LSHADE," *Expert Systems with Applications*, vol. 165, 113875, 2021.
- [4] A. Özkiş and A. Babalık, "Solving constrained engineering design problems with multi-objective artificial algae algorithm," *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, vol. 29, no. 2, pp. 183-193, 2023.
- [5] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems," *Mathematics and Computers in Simulation*, vol. 192, pp. 84-110, 2022.
- [6] H. T. Kahraman, M. Katı, S. Aras, and D. A. Taşci, "Development of the Natural Survivor Method (NSM) for designing an updating mechanism in metaheuristic search algorithms," *Engineering Applications of Artificial Intelligence*, vol. 122, 106121, 2023.
- [7] V. Hayyolalam and A. A. P. Kazem, "Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 87, 103249, 2020.
- [8] K. Zervoudakis and S. Tsafarakis, "A mayfly optimization algorithm," *Computers & Industrial Engineering*, vol. 145, 106559, 2020.
- [9] Y. Zhang and Z. Jin, "Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems," *Expert Systems with Applications*, vol. 148, 113246, 2020.
- [10] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Information Sciences*, vol. 540, pp. 131-159, 2020.
- [11] M. H. Qais, H. M. Hasanien, and S. Alghuwainem, "Transient search optimization: a new meta-heuristic optimization algorithm," *Applied Intelligence*, vol. 50, 3926-3941, 2020.
- [12] M. S. Braik, "Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems," *Expert Systems with Applications*, vol. 174, 114685, 2021.
- [13] F. MiarNaeimi, G. Azizyan, and M. Rashki, "Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems," *Knowledge-Based Systems*, vol. 213, 106711, 2021.
- [14] M. Braik, A. Sheta, and H. Al-Hiary, "A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm," *Neural Computing and Applications*, vol. 33, no.7, pp. 2515-2547, 2021.
- [15] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems," *Applied Intelligence*, vol. 51, pp. 1531-1551, 2021.
- [16] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Dwarf mongoose optimization algorithm". *Computer Methods in Applied Mechanics and Engineering*, vol. 391, 114570, 2022.
- [17] J. S. Pan, L. G. Zhang, R. B. Wang, V. Snášel, and S. C. Chu, "Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems," *Mathematics and Computers in Simulation*, vol. 202, pp. 343-373, 2022.
- [18] D. Połap and M. Woźniak, "Red fox optimization algorithm," *Expert Systems with Applications*, vol. 166, 114107, 2021.
- [19] M. Dehghani, Š. Hubálovský, and P. Trojovský, "Tasmanian devil optimization: a new bio-inspired optimization algorithm for solving optimization algorithm," *IEEE Access*, vol. 10, 19599-19620, 2022.
- [20] T. S. Ayyarao, N. S. S. Ramakrishna, R. M. Elavarasan, N. Polumahanthi, M. Rambabu, G. Saini, B. Khan, and B. Alatas, "War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization," *IEEE Access*, vol. 10, pp. 25073-25105, 2022.
- [21] I. Naruei and F. Keynia, "Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems," *Engineering with Computers*, vol. 38, pp. 3025-3056, 2022.

- [22] M. Dehghani, Z. Montazeri, E. Trojovská, and P. Trojovský, "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 259, 110011, 2023.
- [23] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, "Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems," *Knowledge-Based Systems*, vol. 262, 110248, 2023.
- [24] S. Xian and X. Feng, "Meerkat optimization algorithm: A new meta-heuristic optimization algorithm for solving constrained engineering problems," *Expert Systems with Applications*, vol. 231, 120482, 2023.
- [25] M. Azizi, U. Aickelin, H. A. Khorshidi, and M. B. Shishehgharkhane, "Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization," *Scientific Reports*, vol. 13, no. 1, 2023.
- [26] Q. Zhang, H. Gao, Z. H. Zhan, J. Li, and H. Zhang, "Growth Optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems," *Knowledge-Based Systems*, vol. 261, 110206, 2023.
- [27] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer," *Neural Computing and Applications*, vol. 35, no. 5, pp. 4099-4131, 2023.
- [28] M. Kaveh, M. S. Mesgari, and B. Saeidian, "Orchard Algorithm (OA): A new meta-heuristic algorithm for solving discrete and continuous optimization problems," *Mathematics and Computers in Simulation*, vol. 208, pp. 95-135, 2023.
- [29] M. Han, Z. Du, K. Yuen, H. Zhu, Y. Li, and Q. Yuan, "Walrus Optimizer: A novel nature-inspired metaheuristic algorithm," *Expert Systems with Applications*, vol. 239, 122413, 2023.
- [30] D. Zhu, S. Wang, C. Zhou, S. Yan, and J. Xue "Human memory optimization algorithm: A memory-inspired optimizer for global optimization problems," *Expert Systems with Applications*, vol. 237, 121597, 2024.
- [31] E. S. M. El-kenawy, N. Khodadadi, S. Mirjalili, A. A. Abdelhamid, M. M. Eid, and A. Ibrahim, "Greylag Goose Optimization: Nature-inspired optimization algorithm," *Expert Systems with Applications*, vol. 238, 122147, 2023.
- [32] S. B. Aydemir, "A novel arithmetic optimization algorithm based on chaotic maps for global optimization," *Evolutionary Intelligence*, vol. 16, no. 3, pp. 981-996, 2023.
- [33] S. Ekinçi, D. Izci, R. A. Zitar, A. R. Alsoud, and L. Abualigah, "Development of Lévy flight-based reptile search algorithm with local search ability for power systems engineering design problems," *Neural Computing and Applications*, vol. 34, no. 22, pp. 20263-20283, 2022.
- [34] H. Bakır, U. Guvenc, H. T. Kahraman, and S. Duman, "Improved Lévy flight distribution algorithm with FDB-based guiding mechanism for AVR system optimal design," *Computers & Industrial Engineering*, vol. 168, 108032, 2022.
- [35] C. Zhong, G. Li, Z. Meng, and W. He, "Opposition-based learning equilibrium optimizer with Levy flight and evolutionary population dynamics for high-dimensional global optimization problems," *Expert Systems with Applications*, vol. 215, 119303, 2023.
- [36] H. Bakır, S. Duman, U. Guvenc, and H. T. Kahraman, "Improved adaptive gaining-sharing knowledge algorithm with FDB-based guiding mechanism for optimization of optimal reactive power flow problem," *Electrical Engineering*, vol. 105, no. 5, pp. 3121-3160, 2023.
- [37] X. Chen, B. Xu, C. Mei, Y. Ding, and K. Li, "Teaching-learning-based artificial bee colony for solar photovoltaic parameter estimation," *Applied Energy*, vol. 212, pp. 1578-1588, 2018.
- [38] H. Salimi, "Stochastic fractal search: a powerful metaheuristic algorithm," *Knowledge-Based Systems*, vol. 75, pp. 1-18, 2015.
- [39] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," *In 2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 145-152. IEEE, 2017.
- [40] A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization Nanyang Technological University," *Tech. Rep*, 2022.
- [41] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 114, 105082, 2022.
- [42] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044-2064, 2010.
- [43] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3-18, 2011.
- [44] H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, M. Mafarja, and H. Chen, "RIME: A physics-based optimization". *Neurocomputing*, vol. 532, pp. 183-214, 2023.
- [45] A. S. Sadiq, A. A. Dehkordi, S. Mirjalili, and Q. V. Pham, "Nonlinear marine predator algorithm: A cost-effective optimizer for fair power allocation in NOMA-VLC-B5G networks," *Expert Systems with Applications*, vol. 203, 117395, 2022.
- [46] M. Dehghani, Š. Hubálovský, and P. Trojovský, "Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems," *IEEE Access*, vol. 9, pp. 162059-162080, 2021.

- [47] M. Abdel-Basset, R. Mohamed, S. A. A. Azeem, M. Jameel, and M. Abouhawwash, "Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion," *Knowledge-Based Systems*, vol. 268, 110454, 2023.
- [48] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili "Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems," *International Journal of Intelligent Systems*, vol. 36, no. 10, pp. 5887-5958, 2021.
- [49] M. Abdel-Basset, D. El-Shahat, M. Jameel, and M. Abouhawwash, "Exponential distribution optimizer (EDO): a novel math-inspired algorithm for global optimization and engineering problems," *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9329-9400, 2023.
- [50] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications*, vol. 177, 114864, 2021.

Author Contribution

Hüseyin Bakır: Conceptualization, Methodology, Investigation, Software, Formal analysis, Data curation, Writing - original draft.

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate™.

Performance Analysis of Open Source Cloud Computing Architectures

Mehmet Zahid Kuzan¹ , Abdullah Sevin² 

¹Department of Computer and Information Engineering, Sakarya University, Sakarya, Türkiye

²Department of Computer Engineering, Sakarya University, Sakarya, Türkiye

Corresponding author:

Mehmet Zahid Kuzan, Sakarya University,
Department of Computer and Information
Engineering, Sakarya, Türkiye
mehmet.kuzan@ogr.sakarya.edu.tr



Article History:
Received: 06.02.2024
Accepted: 26.07.2024
Published Online: 26.08.2024

ABSTRACT

Organizations prioritize cloud computing as the primary solution for ensuring secure and, most importantly, uninterrupted delivery of data. In today's landscape, there are various architectural frameworks offering cloud computing infrastructures to choose from. It is crucial to understand how advantageous the performance offered by a chosen architecture is when making a selection. This study aims to comprehend how effective open-source cloud computing architectures are in specific usage scenarios, provide a comparative perspective, and assist organizations in making informed decisions when selecting their IT infrastructures. In line with the study's objectives, the network, disk, and processor performances of OpenStack, OpenNebula, and Apache CloudStack architectures on the same hardware are examined based on data-driven methods. The Iperf3 tool is used for network performance measurement, the dd (Linux) tool for disk performance measurement, and the Sysbench tool for processor performance measurement. After conducting performance measurements with these tools, it was observed that the OpenNebula architecture outperformed the other architectures in terms of overall performance.

Keywords: Open source, Performance analysis, Cloud computing

1. Introduction

Data access difficulties, data loss, or damage can disrupt business continuity in organizations and lead to reputation losses [1]. Cloud computing is considered a priority solution for organizations to ensure the secure and, most importantly, uninterrupted delivery of data [2]. In today's landscape, organizations can prefer various cloud computing architectures[3]. When selecting these architectures, criteria such as cost-effectiveness and customizability can be used in decision-making. Still, it is crucial to know how advantageous the performance offered by the chosen platform is in meeting the needs [4].

This study, titled "Performance Analysis of Open Source Cloud Computing Architectures," aims to understand how effective open-source cloud computing architectures are in specific usage scenarios, provide a comparative perspective, and assist organizations in making informed decisions when selecting their IT infrastructures. In the scope of the performance evaluation conducted in this study, the data-driven methodology proposed by Li et al. for performance evaluation in cloud services will be followed [5]. According to this methodology, to accurately conduct the evaluation, requirements must first be determined. For this, questions defining the purpose should be prepared. Once the purpose is determined, the performance metrics need to be evaluated, and the shaping of the test design process needs to be resolved. After completing the test design process, tests should be conducted based on the defined metrics, and the results obtained from the tests should be analyzed.

The study will focus on open-source cloud computing architectures, specifically OpenStack, Apache CloudStack, and OpenNebula architectures. The aim is to measure these architectures' network, disk, and processor performance. In this context, network bandwidth, disk read speed, disk write speed, and processor speed are designated as performance metrics. After preparing the relevant test environments, performance evaluation tools will be used for metric measurements. The results obtained from the measurements will be used to evaluate the performance of these architectures.

2. Related Works

This section examines previous studies conducted in the literature to acquire knowledge in the research field and interpret research findings. Eren investigated the working logic of virtualization systems, considering performance and usability

among applications [6]. The study observed disk, processor, and memory resource changes in virtualization environments. The results showed that increasing the allocated resources improved performance. A study by Doğru aimed to facilitate users' selection between hypervisor and container-based virtualization technologies in cloud computing infrastructure by measuring efficiency [7]. The study observed how disk, memory, and processors performed different tasks, concluding that container virtualization technology outperformed hypervisor virtualization. Elmas conducted a study to observe the efficiency of physical and virtual servers in cloud computing regarding service continuity [8]. The study monitored the memory, processor, and disk performance of randomly selected physical and virtual servers in specific scenarios. The results showed no significant differences between the server types except for memory usage. Ataş studied the processor, disk, and network performance evaluations of Cloud Foundry, Heroku, and Openshift cloud computing platforms [9]. The study involved tests based on four scenarios, revealing that each platform performed differently in various functions. Husain and colleagues conducted a performance analysis of OpenStack and Eucalyptus architectures in 2018 [10]. The study involved performance tests for processors, memory, disk, and network in two hardware configurations. The results indicated that OpenStack's architecture had better network, memory, and processor performance than Eucalyptus. Yadav conducted a study in 2013 examining the features of Eucalyptus, OpenStack, and OpenNebula architectures [11]. The study discussed various features and differences, such as programming languages, virtualization, storage, and image management. Bedi and colleagues conducted a study in 2018 examining the features of OpenNebula, CloudStack, Eucalyptus, and OpenStack architectures [12]. The study covered architectural components, licensing types, development communities, and their positions in the cloud ecosystem. Saisree and Shitthart conducted a 2022 study examining the features of Eucalyptus, OpenStack, and CloudStack architectures [13]. The study discussed operational features, communities contributing to the development of these architectures, and development models. Mohammed and Kiran conducted a technical evaluation study in 2015, assessing Eucalyptus, CloudStack, OpenStack, OpenNebula, Nimbus, Xen Cloud Platform (XCP), OpenIoT, and AbiCloud architectures and solutions [14]. The study covered the working principles of cloud computing architectures and the features of open-source cloud computing platforms. Işık conducted a performance evaluation on the OpenStack cloud architecture with hypervisor and container virtualization solutions [15]. The study used benchmarking tools to examine processor, memory, network, and disk performance. The results indicated that container virtualization has advantages over hypervisor virtualization in different workloads. Peng and colleagues have examined the technical specifications of the AbiCloud, Eucalyptus, Nimbus, and OpenNebula architectures [16]. Observations reveal that each architecture has its unique strengths.

3. Definitions

3.1. Cloud Computing

Although there are many definitions related to cloud computing, the definition generally referred to in the literature is the one made by the National Institute of Standards and Technology (NIST) of the United States. This definition defines cloud computing as a model with configurable resources (computer networks, databases, servers, applications, services, etc.) that can be rapidly provisioned and released with minimal management effort [17].

3.2. Open Source Cloud Computing Architectures

Cloud computing systems can be divided into two parts: the front end, the user interface for accessing the cloud, and the back end, which consists of servers that make up the system [4]. Communication between these two parts is managed by middleware, which follows specific rules and protocols through network connections [4].

Cloud computing architectures are design models to create the cloud computing environment [18]. These architectures determine the components that make up the cloud environment for the services to be provided, how these components will connect, and the infrastructure needs for the services to be offered [18]. Open-source cloud computing architectures are systems created as open-source software and designed to provide cloud computing services [19].

3.3. OpenStack

OpenStack is an open-source cloud computing platform initiated in 2010 under the Apache license by Rackspace and NASA to build, manage, and deploy infrastructure as a service [20], [21]. The platform utilizes virtualization technologies on hardware, allowing users to modularly access the computing, storage, and networking services they need through a web interface [22]. The fundamental architectural components of OpenStack are as follows [20], [21] [22];

- **Compute:** Manages resources and enables the creation and scaling of virtual machines through the Nova component.
- **Storage:** Provided through Swift (Object Storage) component offering object storage services and Cinder (Block Storage) component providing block-based storage.
- **Networking:** Network resources are managed through the component called Neutron. The Neutron component provides network services such as virtual networks, subnets, and routing.
- **Orchestration:** Automates application deployment with the component named Heat.
- **Image Service:** Manages disk images of virtual machines using the component called Glance.

- **Monitoring:** Monitors the system using the Telemetry component, allowing the creation of alarms and notifications.
- **Dashboard:** Manages cloud resources through a web-based user interface with the component named Horizon.
- **Identity Management:** Provides identity management and user authorization with the Keystone component.

3.4. Apache CloudStack

Apache CloudStack is an open-source cloud computing platform initiated by Citrix Systems in 2011[23]. It is designed with a modular structure and aims to create and automate virtual machines and other infrastructure components. The key architectural elements of CloudStack include [12]:

- **Management Server:** The primary component that manages the CloudStack cloud environment. It processes user requests, manages resources, creates virtual machines, stops them, and performs other management tasks.
- **Database:** Used to store the state of the cloud infrastructure, user information, virtual machine configurations, and other essential data.
- **Cloud Infrastructure:** CloudStack has an infrastructure layer that provides access to the resources of physical or virtual machines. This layer includes storage, networking, and virtual machines.
- **Network:** Manages virtual networks and network services in CloudStack. This layer enables users to create, configure, and manage virtual networks and network services.
- **Hypervisor:** CloudStack supports various virtualization technologies. This layer ensures integration with a specific hypervisor (such as KVM, Xen, VMware, etc.) and manages the operation of virtual machines on this hypervisor.
- **Storage:** Manages different types of storage in CloudStack. The storage layer oversees storage pools and stores disk images of virtual machines.
- **API (Application Programming Interface):** This layer is the interface through which CloudStack interacts with the outside world. It hosts APIs used to access cloud resources programmatically.
- **User Interface:** The User Interface is a web-based interface where users can visually manage cloud resources. Users can use this interface to manage virtual machines, configure networks, and monitor other cloud resources.

3.5. OpenNebula

OpenNebula was initiated by Ignacio M. Llorente and Ruben S. Montero in 2005 [11]. It is an open-source cloud computing platform and virtualization management software. The core architectural components of OpenNebula include [11]

- **Core:** This component handles user requests for resource allocation and communicates with other components to manage the cloud infrastructure. The OpenNebula Core manages the cloud infrastructure and controls fundamental cloud functions.
- **Compute:** Responsible for managing virtual machines in the cloud environment. It facilitates the creation, startup, shutdown, resizing, and disabling of virtual machines.
- **Storage:** Manages the storage resources of virtual machines. Transfer Manager (TM) and Datastore Manager (DS) are used to create, resize, copy, delete virtual disks, and manage file systems.
- **Network:** Manages tasks related to network components. This component handles the management of virtual networks, routers, and other resources.
- **SQL Database:** A SQL database stores system configuration information and user data.
- **User Interface (Front-end):** This web-based component allows users to manage and monitor cloud resources. The user interface is provided by a component called OpenNebula Sunstone.
- **Tools Layer:** This component tracks virtual machine statuses, resource usage, and user activities.
- **Marketplace:** A component where users can download ready-made virtual machine templates and applications.

4. Material and Method

Inspired by existing studies in experimental computer science and principles of experimental design, Li and colleagues developed the DoKnowMe methodology for the performance evaluation of software and computing systems [5]. The evaluation process of DoKnowMe consists of a sequential main process and recurring experimental activities. Each evaluation step uses inputs, relevant evaluation strategies, and outputs. During this process, evaluation concepts are defined, performance metrics are determined, and templates are created for repeatability. According to this methodology, requirements

must be determined to conduct the evaluation accurately. For this purpose, questions defining the objectives should be prepared. Subsequently, determining the performance metrics to be evaluated and shaping the test design process is essential. It is crucial to define why the tests for performance evaluation are conducted to outline the framework of the study. In this study, the following questions have been prepared to evaluate the network, disk, and processor performances of OpenStack, Apache CloudStack, and OpenNebula architectures:

- Question 1: Among OpenStack, Apache CloudStack, and OpenNebula architectures, which one exhibits higher network performance?
- Question 2: Among OpenStack, Apache CloudStack, and OpenNebula architectures, which one demonstrates higher disk performance?
- Question 3: Among OpenStack, Apache CloudStack, and OpenNebula architectures, which one shows higher processor performance?

Network performance will be assessed based on network speed and bandwidth, disk performance will be evaluated through reading and writing speed from/to the disk, and processor performance will be measured by the calculation speed of the processor in arithmetic operations. The identified performance metrics will be tested using performance measurement tools. iPerf3 will be used to observe network bandwidth. iPerf3 is a widely used open-source testing tool that can generate TCP and UDP data streams and measure the network's efficiency [24]. To measure disk read and write performance, the dd tool (Linux) will be used. The dd (Linux) tool, through specific commands, writes the desired-sized data to the disk in blocks, allowing the measurement of how quickly the data is written to the disk and how quickly the specified-sized data can be read from the disk [25]. The Sysbench tool will be used to measure processor performance. The Sysbench tool enables the calculation of prime numbers up to a user-defined size. It provides metrics such as the time taken during the process and the number of successful operations performed [26]. Table 1 presents the performance testing tools and measurement metrics used in the study.

Table 1. Performance Testing Tools and Measurement Metrics

Source	Test Tool	Measurement Metric	Unit
Network	Iperf3	Bandwidth	MB/s
Disk	dd (Linux)	Writing speed of 256 MB data to disk	MB/s
Disk	dd (Linux)	Writing speed of 512 MB data to disk	MB/s
Disk	dd (Linux)	Writing speed of 1 GB data to disk	MB/s
Disk	dd (Linux)	Reading speed of 256 MB data from disk	MB/s
Disk	dd (Linux)	Reading speed of 512 MB data from disk	MB/s
Disk	dd (Linux)	Reading speed of 1 GB data from disk	MB/s
Processor	Sysbench	Finding prime numbers up to 5.000	s
Processor	Sysbench	Finding prime numbers up to 10.000	s
Processor	Sysbench	Finding prime numbers up to 20.000	s

The minimum hardware requirements specified by the developers of OpenStack, Apache CloudStack, and OpenNebula architectures will be used for testing. In this context, 16 GB of RAM, 120 GB of hard disk, and 8 CPUs are sufficient for each architecture. The primary purpose of preferring hardware configurations with identical specifications is to observe the performance of each architecture on hardware with equivalent features. The hardware specifications to be used for the tests are indicated in Table 2.

Table 2. Hardware Specifications to be Used in the Scope of the Test

Requirements	Feature
Processor	Intel® Core™ i7-1255U 12.Gen. 3.50 GHz 10 Core 12 Thread
Memory	Samsung M471A2K43EB1-CWE Ram 3200MHz 32GB
Disk	Samsung MZVLQ512HBLU-00BH1 512GB SSD
Network Interface Card	Realtek RTL8111 10/100/1000M
Operating System	Ubuntu 22.04 LTS (Wayland)

The hardware used in the test has an installed Ubuntu 20.04.6 LTS (Focal Fossa) operating system. Subsequently, the KVM virtualization software was installed on the operating system to provide hardware configuration with identical features for the architectures. After preparing the host machine where the test environments will be set up, the relevant architectures were installed. Each architecture, with hardware specifications of 16 GB RAM, 120 GB disk, and 8 CPUs, was installed on the Ubuntu 20.04.6 LTS (Focal Fossa) virtual operating system as a single node according to the installation guide specified on its official website. After the architectures were installed, the Ubuntu Server 18.04 LTS (Bionic Beaver) operating system was installed as a virtual machine on each architecture to perform network, disk, and processor tests. Access to the virtual machines was established using the SSH protocol, and performance test tools were installed. Network, disk, and processor performance tests for the OpenStack, Apache CloudStack, and OpenNebula architectures will follow the flow outlined in Figure 1.

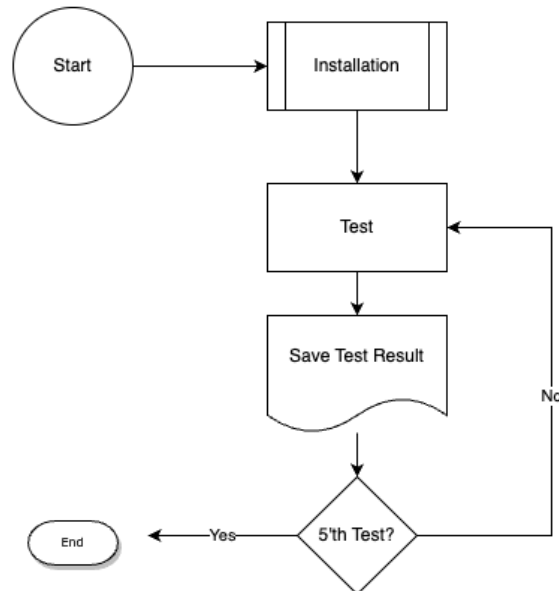


Figure 1. Test Flow Chart

Test Flow:

- Installation: Set up OpenStack, Apache CloudStack, and OpenNebula architectures separately.
- Test: Utilize performance measurement tools to test the identified performance metrics for each architecture.
- Recording: Record the results of each test for further analysis.
- Save Test Result: Repeat each test five times to ensure consistency and reliability of results.

This test plan ensures a comprehensive and reliable evaluation of the network, disk, and processor performances of OpenStack, Apache CloudStack, and OpenNebula architectures. The repetition and averaging approach aims to minimize potential outliers and accurately represent the systems' performance.

4.1. Network Performance Measurement

The study will focus on network speed and bandwidth metrics for network performance. In this context, measuring network speed involves quantifying the data transferred between the client and server in a specific period and observing the achieved bandwidth during this process. In this study, network performance tests will be conducted using the iPerf3 tool over the TCP protocol. The test results have been visualized in Figure 2.

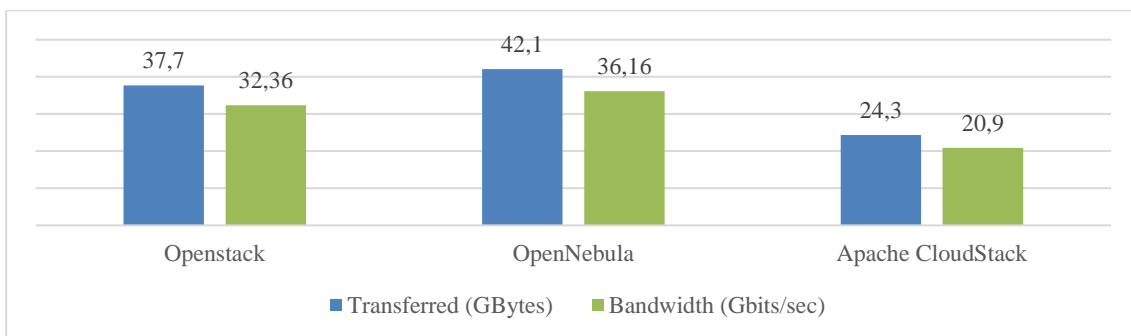


Figure 2. Network Performance Test Results

Based on the average results of the tests conducted with the iPerf3 tool, OpenNebula achieved the highest test score regarding network connection speed and bandwidth, with a data transfer rate of 42.10 GB and a bandwidth of 36.16 GB/s. Then, OpenStack ranked second with a data transfer rate of 37.7 GB and a bandwidth of 32.36 GB/s, while CloudStack came last with a data transfer rate of 24.3 GB and a bandwidth of 20.9 GB/s. It's worth noting that the tests were conducted between two operating systems running on the same Ethernet card, which may have contributed to the higher values obtained. Introducing different devices and physical connections could potentially result in reduced bandwidth. The Apache CloudStack architecture utilizes the EC-2 classic network management structure, which is no longer used by Amazon. Therefore, it can be stated that it lags behind other architectures. The OpenStack architecture manages network services through the Neutron component via APIs. The success of the OpenNebula architecture can be attributed to its superior management of message queues through APIs compared to other architectures.

4.2. Disk Performance Measurement

The study focuses on disk performance measurement using metrics such as disk read and write speeds. In this context, it is essential to observe the time it takes to write certain-sized data to the disk and read the same-sized data from it. Disk read and write performance tests will be conducted using the dd (Linux) tool. The disk reading test results are visualized in Figure 3, and the writing test results are in Figure 4.

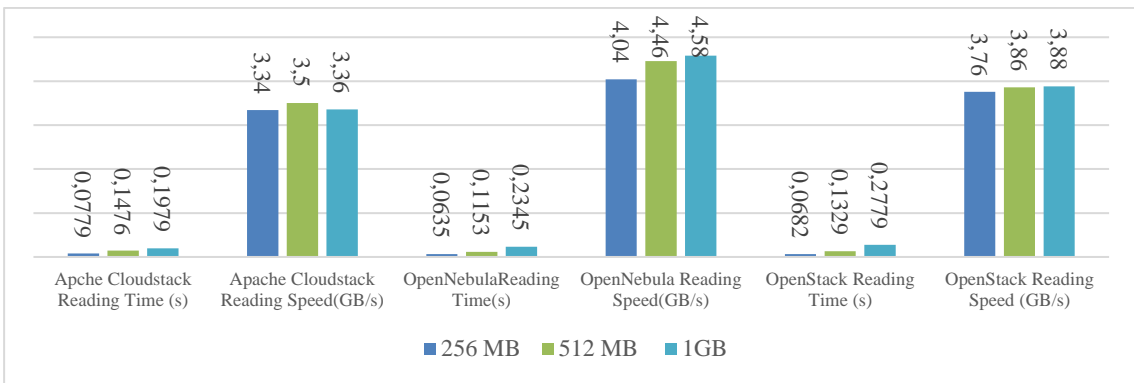


Figure 3. Disk Reading Test Results

Considering that the architecture that performs the most data reading in the least amount of time is considered more successful according to this criterion:

- In the test of reading 256 MB of data from the disk, OpenNebula architecture showed the best performance. Following in performance were the OpenStack architecture and, lastly, the CloudStack architecture.
- In the test of reading 512 MB of data from the disk, OpenNebula architecture showed the best performance. Following in performance were the OpenStack architecture and, lastly, the CloudStack architecture.
- In the test of reading 1 GB of data from the disk, OpenNebula architecture showed the best performance. Following in performance were the OpenStack architecture and, lastly, the CloudStack architecture.

According to the results obtained from the tests, it is observed that OpenNebula architecture is the most successful in disk reading tests. Following the ranking, OpenStack architecture is in the second position, and CloudStack architecture is in the last position. The performance of reading data from disk depends on the way the architecture holds and manages data through its storage components. The Apache CloudStack architecture utilizes primary and secondary storage areas for storage, managed through CloudStack APIs. The OpenStack architecture provides block storage with the Cinder component. OpenNebula architecture, on the other hand, performs storage through data stores named DataStore, and access to these is achieved through a component called Transfer Manager. The fundamental reason for the success of the OpenNebula architecture compared to other architectures lies in the stronger performance of its storage components in reading data from disk.

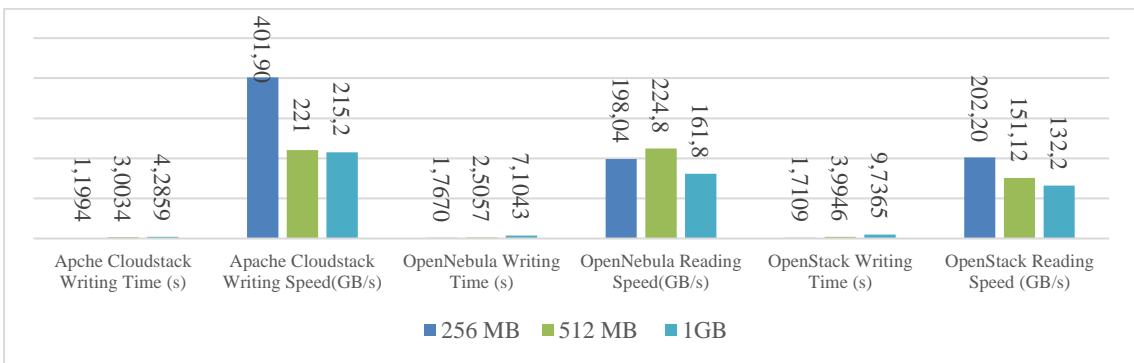


Figure 4. Disk Writing Test Results

Considering that the architecture that can write the most data to the disk in the least amount of time is considered more successful according to this criterion:

- In the test of writing 256 MB of data to the disk, Apache CloudStack architecture showed the best performance. Following in performance were the OpenStack architecture and, lastly, the OpenNebula architecture.
- In the test of writing 512 MB of data to the disk, OpenNebula architecture showed the best performance. The Apache CloudStack architecture followed in performance and, lastly, the OpenStack architecture.
- In the test of writing 1 GB of data to the disk, Apache CloudStack architecture showed the best performance. Following in performance were the OpenNebula architecture and, lastly, the OpenStack architecture.

According to the results obtained, it was observed that the most successful architecture in disk writing tests was Apache CloudStack. In the ranking, OpenNebula architecture followed, and finally, OpenStack architecture was in the last place. The performance of writing data to disk depends on the way the architecture holds and manages data through its storage components. The Apache CloudStack architecture utilizes primary and secondary storage areas for storage, managed through CloudStack APIs. The OpenStack architecture provides block storage with the Cinder component. OpenNebula architecture, on the other hand, performs storage through data stores named DataStore, and access to these is achieved through a component called Transfer Manager. The fundamental reason for the success of the Apache CloudStack architecture compared to other architectures lies in the stronger performance of its storage components in writing data to disk.

4.3. Processor Performance Measurement

The measurement of processor performance in the study will be based on the metric of the processor's computational speed in arithmetic operations. In this regard, observing how quickly the processor completes an arithmetic operation is necessary. Arithmetic operations will be performed using the Sysbench tool to conduct processor performance tests. The Processor Performance Tests' results are visualized in Figure 5.

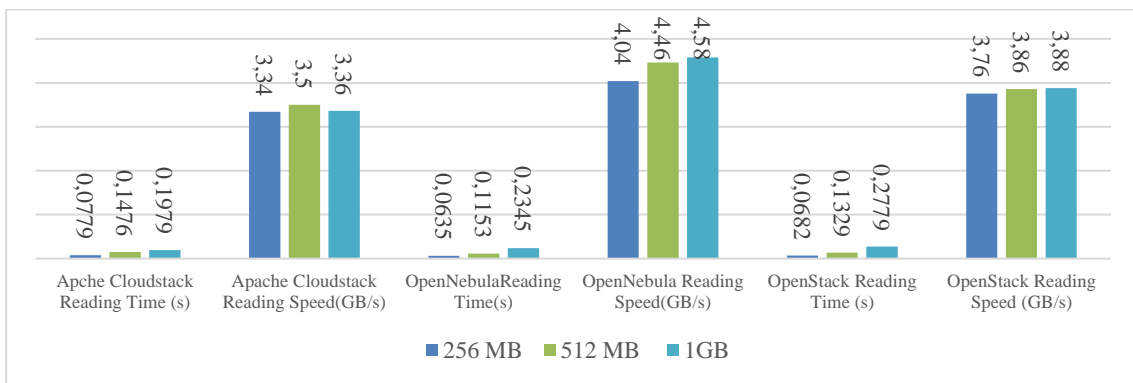


Figure 5. Processor Performance Test Results

Evaluation based on the criterion that the architecture performing the most successful operations in the least amount of time is considered more successful;

- In the prime number finding test up to 5.000, OpenNebula architecture has shown the best performance. Following this, there is the OpenStack architecture, and lastly, the CloudStack architecture.
- In the prime number finding test up to 10.000, OpenNebula architecture has shown the best performance. Following this, there is the OpenStack architecture, and lastly, the CloudStack architecture.
- In the prime number finding test up to 20.000, OpenNebula architecture has shown the best performance. Following this, there is the OpenStack architecture, and lastly, the CloudStack architecture.

According to the results obtained from the tests, it has been observed that the architectures exhibit similar values in the conducted tests. When performance ranking is considered, it can be said that Apache CloudStack architecture outperformed the others in all processor tests. Processor performance in architecture varies based on how the computing components utilize and manage the processor cores. In the OpenStack architecture, the component used for computing tasks is called Nova. This component utilizes a set of tools for creating virtual machines and resource allocation for computing tasks coming from users. In the OpenNebula architecture, the component used for computing tasks is called OpenNebula Daemon. This component utilizes a set of virtualization drivers for creating virtual machines and resource allocation for computing tasks coming from users. In the Apache CloudStack architecture, computing tasks are handled through the Management Server component and the Management Agent component. These components manage the resources of virtual machines, ensuring the fulfillment of user requests. The fundamental reason for the success of the Apache CloudStack architecture compared to other architectures lies in the stronger performance of its computing components in utilizing processor cores.

5. Conclusion and Recommendations

Critical metrics such as processor speed, network speed, and disk read/write speed were determined for performance analysis, and the test preparations were made based on these objectives. While preparing the test environments, it was ensured that each architecture operated on the same hardware configuration. The primary purpose of doing this is to observe the performances of architectures under the same hardware scale. After the test environments were prepared, a virtual server was created for each architecture, and according to the test plan, each performance metric was observed using test tools. Based on the results obtained from the tests, scores were assigned to the architectures for each performance metric. A scoring system was applied where the most successful architecture received 3 points, the second-ranked architecture received 2 points, and the last-ranked architecture received 1 point. Table 1 shows the scores each architecture received after all the tests.

Table 1. Initial Results

Architecture	Disk Writing Test			Disk Reading Test			Network Test	Processor Speed			Total
	256 MB	512 MB	1 GB	256 MB	512 MB	1 GB	Bandwidth	5.000	10.000	20.000	
OpenStack	2	1	1	2	2	2	2	1	1	1	15
OpenNebula	1	3	2	3	3	3	3	2	2	2	24
Apache CloudStack	3	2	3	1	1	1	1	3	3	3	21

According to the scoring table, it is understood that the performance values of the architectures may vary depending on the tested metrics. Based on the results of the disk writing tests, it was observed that the Apache CloudStack architecture is the fastest for writing 256 MB-sized data to the disk, OpenNebula architecture is the fastest for writing 512 MB-sized data to the disk, and Apache CloudStack architecture is the fastest for writing a 1 GB-sized data to the disk. In the results of the disk reading tests, it was observed that the OpenNebula architecture is the fastest for reading 256 MB-sized data from the disk, OpenNebula architecture is the fastest for reading 512 MB-sized data from the disk, and OpenNebula architecture is the fastest for reading a 1 GB-sized data from the disk. The results of the network bandwidth tests showed that the OpenNebula architecture provides the widest bandwidth. Then, OpenStack architecture showed the best performance and Apache CloudStack architecture showed the least performance. According to the results of the processor speed tests for finding prime numbers up to 5.000, 10.000, and 20.000, Apache CloudStack architecture ranked first, OpenNebula architecture ranked second, and OpenStack architecture ranked third. After the tests, the overall scoring resulted in OpenNebula architecture being the first with 24 points, Apache CloudStack architecture being the second with 21 points, and OpenStack architecture being the third with 15 points. It was observed that OpenNebula architecture stands out in terms of network and disk data reading. In this study, each architecture was installed on a single node due to the hardware constraint. Observations indicate that the performance of architectures varies based on how they store, manage, and process data. This variation is closely associated with the success exhibited by the components responsible for network management, computation, and storage within the architectures. The performances obtained by distributing and running the relevant architectures on multiple virtual servers have not been observed. In future studies evaluating the performances of cloud computing architectures, distributing installations to multiple virtual servers according to architectural requirements will contribute to addressing the deficiencies in the literature.

References

- [1] İ. Günebakan, "KOBİ'ler İçin Bulut Bilişimin Avantaj ve Dezavantajları," *International Journal of Academic Value Studies*, vol. 2, no. 3, pp. 116–132, May 2016.
- [2] C. Zou, H. Deng, and Q. Qiu, "Design and implementation of hybrid cloud computing architecture based on cloud bus," in *Proceedings - IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2013*, 2013, pp. 289–293. doi: 10.1109/MSN.2013.72.
- [3] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok, "A Survey on Open-source Cloud Computing Solutions," in *VIII Workshop em Clouds, Grids e Aplicações*, 2010, pp. 3–16.
- [4] J. W. Rittinghouse and J. F. Ransome, "Cloud Computing: History and Evolution," in *Encyclopedia of Information Systems and Technology - Two Volume Set*, vol. 1, CRC Press, 2011, pp. 178–192.
- [5] Z. Li, L. O'Brien, and M. Kihl, "DoKnowMe: Towards a Domain Knowledge-driven Methodology for Performance Evaluation," Aug. 2017, doi: 10.1145/2897356.2897360.
- [6] S. Eren, "VMWare ve Hyper-V Sanallaştırma Sistemlerinin Performans ve Yüksek Kullanılabilirliğinin Deneysel Olarak Karşılaştırılması," Yüksek Lisans Tezi, Maltepe Üniversitesi, 2020.
- [7] A. Doğru, "Sunucu Sanallaştırma ve Uygulama Sanallaştırma Teknolojileri Performans Karşılaştırması," Yüksek Lisans Tezi, Maltepe Üniversitesi, 2019.
- [8] H. Elmas, "Bulut Teknolojisinin Uygulama Sunucularının Yönetimi ve Performansı Üzerindeki Etkisi," Yüksek Lisans Tezi, İstanbul Üniversitesi, 2013.
- [9] G. Ataş, "Performance Evaluations of Cloud Computing Platforms," Yüksek Lisans Tezi, Bahçeşehir University, 2013.
- [10] A. Husain, M. H. Zaki, and S. Islam, "Performance Evaluation of Private Clouds: OpenStack vs Eucalyptus." [Online].

- Available: <http://www.publishingindia.com/ijdcc>
- [11] S. Yadav, "Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, OpenStack and OpenNebula." [Online]. Available: <https://www.researchgate.net/publication/262335770>
- [12] P. Bedi, B. Deep, P. Kumar, and P. Sarna, "Comparative Study of OpenNebula, CloudStack, Eucalyptus and OpenStack," *International Journal of Distributed and Cloud Computing*, vol. 6, no. 1, pp. 37–42, 2018, [Online]. Available: <http://www.publishingindia.com/ijdcc>
- [13] Saisree S. and Shitharth S., *Artificial Intelligence and Data Science*, vol. 1673. In Communications in Computer and Information Science, vol. 1673. Cham: Springer Nature Switzerland, 2022. doi: 10.1007/978-3-031-21385-4.
- [14] B. Mohammed and M. Kiran, "Analysis of Cloud Test Beds Using OpenSource Solutions," in *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, Institute of Electrical and Electronics Engineers Inc., Oct. 2015, pp. 195–203. doi: 10.1109/FiCloud.2015.106.
- [15] G. Işık, U. Gürel, and A. G. Yavuz, "Bulut Ortamlarında Hipervizör ve Konteyner Tipi Sanallaştırmanın Farklı Özellikte İş Yüklerinin Performansına Etkisinin Değerlendirilmesi," *Uludağ University Journal of The Faculty of Engineering*, pp. 981–1002, Aug. 2020, doi: 10.17482/uumfd.605560.
- [16] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," *2nd International Symposium on Information Science and Engineering, ISISE 2009*, pp. 23–27, 2009, doi: 10.1109/ISISE.2009.94.
- [17] E. Simmon, "Evaluation of Cloud Computing Services Based on NIST SP 800-145." Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2018. doi: <https://doi.org/10.6028/NIST.SP.500-322>.
- [18] J. Yang, L. Zhang, and X. A. Wang, "On Cloud Computing Middleware Architecture," in *Proceedings - 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015*, Institute of Electrical and Electronics Engineers Inc., 2015, pp. 832–835. doi: 10.1109/3PGCIC.2015.46.
- [19] T. Cordeiro *et al.*, "Open source cloud computing platforms," in *Proceedings - 9th International Conference on Grid and Cloud Computing, GCC 2010*, 2010, pp. 366–371. doi: 10.1109/GCC.2010.77.
- [20] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an Open-source Solution for Cloud Computing," *Int J Comput Appl*, vol. 55, no. 3, pp. 38–42, Oct. 2012, doi: 10.5120/8738-2991.
- [21] T. Rosado and J. Bernardino, "An overview of OpenStack architecture," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, 2014, pp. 366–367. doi: 10.1145/2628194.2628195.
- [22] Kevin. Jackson, *OpenStack Cloud Computing Cookbook*. Packt Publishing, 2013.
- [23] K. Ahokas, "Comparison of cloud management platforms," 2014.
- [24] G. Işık, "Bulut Ortamlarında Hipervizör ve Konteyner Tipi Sanallaştırmanın Farklı Özellikte İş Yüklerinin Performansına Etkisinin Değerlendirilmesi," Yüksek Lisans TEzi, Eskişehir Osmangazi Üniversitesi, 2021.
- [25] F. Gomez-Folgar, A. Garcia-Loureiro, T. F. Pena, and R. Valin, "Performance of the CloudStack KVM pod primary storage under NFS version 3," in *Proceedings of the 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2012*, 2012, pp. 845–846. doi: 10.1109/ISPA.2012.128.
- [26] A. P. Kurniawan, M. N. Ashar, and F. Hidayat, "Performance Evaluation for Deploying Dockerized Web Application on AWS, GCP, and Azure," in *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, 2023, pp. 346–350.

Author(s) Contributions

The article is multi-author articles. Author 1 conceived the presented idea and performed the calculations. Author 2 developed the theory of this study and supervised its findings. All authors discussed the results and contributed to the final paper.

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.



Availability of Data and Material

Not applicable

Plagiarism Statement

This article has been scanned by iThenticate™.

Cigarette Detection in Images Based on YOLOv8

Yerniyaz Bakhytov¹ , Cemil Öz² 

¹ Sakarya University, Graduate School of Natural and Applied Sciences, Computer and Information Engineering, Sakarya, Türkiye

² Sakarya University, Faculty of Computer and Informatics, Department of Computer Engineering, Sakarya, Türkiye

Corresponding author:

Yerniyaz Bakhytov, Sakarya University,
Graduate School of Natural and Applied Sciences,
Computer and Information Engineering,
Sakarya, Türkiye
yerniyaz.bakhytov@ogr.sakarya.edu.tr

ABSTRACT

This study investigates methods to develop and test the automatic detection of cigarettes in images using modern deep learning models such as YOLOv5 and YOLOv8. The study's primary aim is to improve the accuracy and reliability of recognizing objects associated with smoking, which could significantly enhance the monitoring of public places, media content analysis, and support for anti-smoking campaigns. Tobacco use poses a serious threat to public health, causing numerous diseases and resulting in millions of deaths annually. Advanced technologies such as computer vision and artificial intelligence offer new opportunities for more effective monitoring and analysis, which can help mitigate the negative effects of tobacco use. The training results are presented, with the YOLOv8 model achieving an accuracy of 87.4% and the YOLOv5 model slightly outperforming it with an accuracy of 89.6%. In conclusion, the article thoroughly explores the use of the YOLOv8 model in images for cigarette identification. It contributes to the existing body of knowledge by presenting a comparative analysis of the performance of the YOLOv8 and YOLOv5 models, thereby providing valuable insights for future research.

Keywords: Deep learning, Object detection, YOLOv8, Smoking detection

Article History:

Received: 29.03.2024

Accepted: 22.07.2024

Published Online: 26.08.2024

1. Introduction

Cigarette smoking, a major global health concern, is responsible for numerous deaths and illnesses worldwide, including lung cancer, cardiovascular disease, and chronic obstructive pulmonary disease. As per the World Health Organization's data, in 2023, the global smoking population was around 1.3 billion, with over eight million fatalities attributed to smoking-related diseases [1]. Therefore, it is important to develop effective methods to control and prevent smoking, and to raise awareness about the dangers of smoking.

One of the ways to control smoking is to identify cigarettes by image, which can be used to monitor people smoking in public places, detect violations of smoking laws, and also to analyze the behavior and habits of smokers. Identifying cigarettes from images can also be useful for research in psychology, sociology, and medicine related to smoking.

Cigarette detection from an image is a computer vision task that involves localizing and classifying cigarettes in an image. This task is a subtask of object detection, which consists of finding bounding boxes and class labels for all objects in an image. Object detection is one of the most rapidly growing and challenging areas of computer vision, which has many applications in various fields such as security, medicine, robotics and entertainment.

There are many object detection methods that can be divided into two main categories: region-based and single-pass based. Region-based methods first generate candidate regions that may contain objects and then classify them using convolutional neural networks (CNNs). Examples of such methods are R-CNN and Faster R-CNN. Single pass based methods perform object detection in a single pass of the network, using anchor boxes or center points to predict bounding boxes and class labels. Examples of such methods are YOLO, SSD and RetinaNet [2].

In this work, YOLOv5 and YOLOv8 were selected for the cigarette detection task due to their high performance, accuracy, flexibility, and community support. These models provide the optimal balance between processing speed and detection accuracy, making them ideal for use in monitoring and video surveillance systems where fast and accurate object detection is required. Methods such as Faster R-CNN provide high accuracy but require more processing time due to the two-step process. YOLO solves these problems in a one-step process, speeding up processing time. YOLOv5 and YOLOv8 generally outperform SSDs in accuracy due to their optimized architectures [3]. EfficientDet is resource efficient, but YOLOv8 offers better performance thanks to the latest architectural improvements.

YOLO is one of the most widely used and popular object detection methods today. YOLO differs from other object detection algorithms in its speed and accuracy, and it has become famous for these properties. This method was proposed by Joseph Redmon and Ali Farhadi in 2015. He then released YOLOv2 [4] in 2016, developing the model architecture by adding batch normalization, helper frames, and cluster dimensions. YOLOv3 [5] was released in 2018. Compared to its predecessor, this feature further improves model performance by using a more efficient backbone network, multiple reference systems, and spatial pyramid aggregation. YOLOv4 [6] was released in 2020, introducing innovations such as tiled data augmentation, a state-of-the-art anchor-free detection head, and an improved loss function. In the realm of computer vision, YOLOv5 [7] stood out by boosting model performance and incorporating cutting-edge functionalities such as hyperparameter optimization, integrated experiment tracking, and automatic export to widely used formats. Meituan introduced YOLOv6 [8] in 2022, and it now powers a significant portion of the company's autonomous delivery robots. YOLOv7 [9] introduced new functionalities, such as pose estimation using the COCO key points dataset. The latest release from Ultralytics, YOLOv8, represents a significant advancement in the YOLO series, integrating cutting-edge features and enhancements to elevate its performance, versatility, and efficiency. Supporting a wide spectrum of artificial vision tasks including detection, segmentation, pose estimation, tracking, and classification, YOLOv8 empowers users with a multifaceted solution adaptable to various use cases and industries [10].

This article presents work on cigarette detection from images using the YOLOv8 and YOLOv5 algorithms. The structure of the two algorithms was also analyzed and the similarities and differences between them were analyzed. We trained the models on separate datasets collected from different sources. The results of the two models were then analyzed and compared.

This article explores cigarette detection from images using the YOLOv8 and YOLOv5 algorithms, highlighting their structural similarities and differences. Utilizing state-of-the-art models ensures high speed and accuracy, crucial for public health monitoring and automated surveillance. By training on diverse datasets, we aim for robust performance in real-world scenarios. Comparing the results of the two models provides insights into their respective strengths and weaknesses, contributing to the optimization of object detection tasks and advancing the field of computer vision.

2. Literature review

Cigarette smoking is one of the leading causes of death and morbidity in the world, so it is important to develop effective methods to detect and prevent smoking in public places. In recent years, a lot of research has emerged on the use of deep convolutional neural networks and other computer vision techniques to solve this problem. In this review, we will look at several such studies and compare their approaches, methods, problems and results.

One such system is Eye-Smoker. Based on computer vision and neural networks, the Eye-Smoker system is innovative in detecting smoking in images. Eye-Smoker uses YOLOv3 architecture to detect people smoking in images, focusing on the nose region. The system analyzes signatures such as hand movements, smoke, and posture to detect smoking accurately. Training occurs on various data, including nose images, video, and real detection from a webcam. The system's advantages are its high accuracy and ease of use in public places. However, the system has a disadvantage: if the nose is not visible in the image, then the system cannot detect smoking [11].

Another work on this topic uses deep convolutional neural networks to detect people smoking in public places. This approach emphasizes using a small amount of training data, which is a key feature. The system can detect smoking in images by focusing on the area of the face and hands where signs of smoking often appear [12].

The following work on this topic is the YOLO-Cigarette system. The main architecture of this model is based on YOLOv5, an improved version of YOLO, which is known for its high speed of object detection. One of the critical features of YOLO-Cigarette is to solve the problem of low detection accuracy of small objects such as cigarettes. To achieve this, the model includes a new FSPP (Fine-Grained Spatial Pyramid Pooling Module) module, which improves the accuracy of detecting small targets. In addition, using the MSAM (Multi-Spatial Attention Mechanism) mechanism improves the model's ability to focus on essential parts of the image, which also helps improve detection accuracy. The model parameters are quantized to reduce computational complexity and speed up the detection process. This model demonstrates superiority over the original YOLOv5 model in the detection accuracy of people smoking and achieves high performance in actual outdoor conditions [13].

The following work describes a smoking detection model based on a convolutional neural network called SmokingNet. This model automatically detects smoking in video content through images. Unlike traditional methods based on cigarette smoke detection algorithms, SmokingNet can detect smoking images using only human smoking gesture information and cigarette image characteristics without the need to detect cigarette smoke. It shows high accuracy and superior performance for real-time monitoring [14].

The following paper presents an improved YOLOv5-based algorithm to detect smoking behavior in public places like hospitals, schools, and stations to enhance health and safety. The algorithm improves detection by replacing the C3 module with the CoT module and integrating the CBAM attention mechanism before the SPPF structure, improving feature extraction. The improved algorithm significantly enhanced detection performance through experimental comparisons, raising the accuracy from 61.3% to 62.9%, demonstrating its effectiveness and real-time detection capabilities [15].

The following article proposes an improved YOLOv5 algorithm for smoking detection in public places, addressing low detection accuracy for small and medium targets in complex scenes. The enhancements include using Mosaic-9 for better data enhancement, introducing the Convolutional Block Attention Module (CBAM) to improve focus on target locations, replacing traditional upsampling with transpose convolution for better semantic recognition, and incorporating the SPD-Conv module to enhance recognition of low-resolution and small target images. Experimental results show a 3% improvement in detection accuracy for smoking targets [16].

The subsequent work delves into the details of an advanced algorithm for detecting smoking by substation personnel. The algorithm, based on GhostNetV2-YOLOv5, enhances the original YOLOv5 by improving detection accuracy and speed. The specific improvements include a 2.58% increase in total mean Average Precision (mAP) and a 1.61-fold increase in prediction speed. These enhancements are crucial in preventing equipment damage and ensuring safety [17].

These works all use deep learning to detect smoking, but they use different approaches and techniques. Eye-Smoker and YOLO-Cigarette use variations of the YOLO model for smoking detection, while the other uses its deep learning model. All these works demonstrate high accuracy in their results, highlighting the effectiveness of using deep learning for smoking detection.

3. Methodology

The methodology of our approach is based on using YOLOv8 and YOLOv5 to detect cigarettes in images. However, detecting cigarettes is challenging because cigarettes may be small, partially hidden, or have different shapes and colors. We have developed our algorithm for solving these problems.

3.1. Dataset

The first step in our methodology is to collect a database of images. This is an essential step because the quality and variety of images in the database directly affect the training efficiency of the model. In our case, we collected an image database containing images of cigarettes in various contexts and lighting conditions. We used the Roboflow Universe [18] web application to collect the dataset and label objects in the images.

The dataset we compiled is a testament to our meticulous approach, containing 1200 images of people smoking, all collected for the specific task of detecting cigarettes in photographs. The dataset is divided into three subsets: a training set (70%) with 840 images, a validation set (15%) with 180 images, and a test set (15%) also with 180 images. Each image in the dataset is meticulously annotated, indicating the presence of a cigarette in the photo in the form of bounding boxes around the cigarette. This detailed annotation process ensures the dataset is ready for robust model training. The dataset is rich in diversity, encompassing a variety of scenes, lighting, and poses of people smoking. It also includes background variations such as streets, cafes, and houses, making it a comprehensive representation of various cigarette detection scenarios.

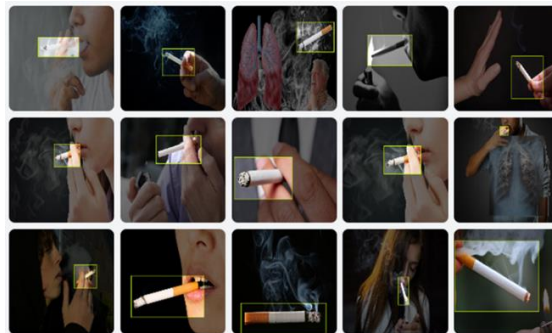


Figure 1. Smoking Cigarettes Dataset

3.2. Detection Models

In this work, using YOLOv5 and YOLOv8 models to detect cigarettes in images represents an essential aspect of computer vision research. The YOLO architecture, known for its high accuracy and speed, is the basis for these models in object recognition. Convolutional neural networks form its basis. The architectures of the YOLOv8 and YOLOv5 models consist of the backbone, neck, and head, as shown in Figure 2 and Figure 3.

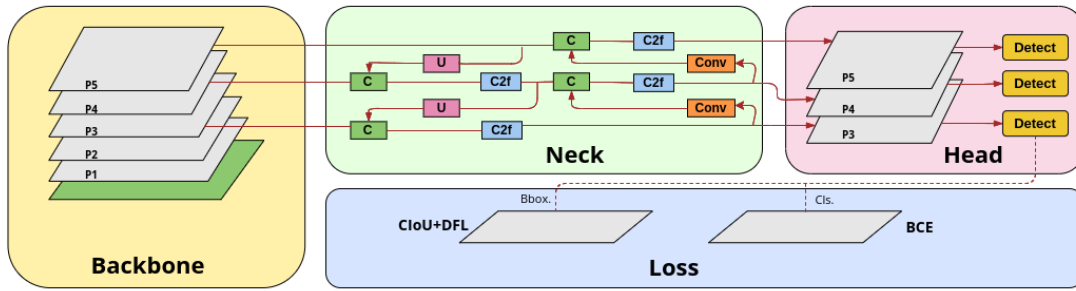


Figure 2. The Structure of the YOLOv8 Algorithm

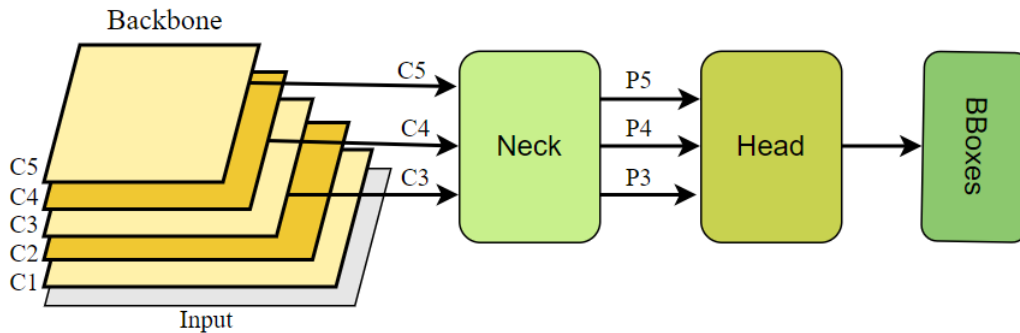


Figure 3. The Structure of the YOLOv5 Algorithm

3.3 Backbone

The two models use the Cross Stage Partial (CSP) [19] architecture, which divides the feature map into two components. Convolution operations are applied to the first part, and the second part is fused with the outcomes from the preceding stage. As a result, the CSP architecture enhances CNN training effectiveness while reducing computational overhead. Unlike YOLOv5, in YOLOv8, the first convolutional kernel was increased from (1x1) to (3x3), and the primary building block C3 was replaced with C2f [20]. In Figure 4 and Figure 5, we use k , s , and p to represent kernel, stride, and padding, respectively. n is a depth parameter that determines the number of BottleNeck stacks by adding additional layers. This varies across the model scales: nano, small, medium, large and extra-large.



Figure 4. The Layout of the YOLOv8 Algorithm’s Architecture

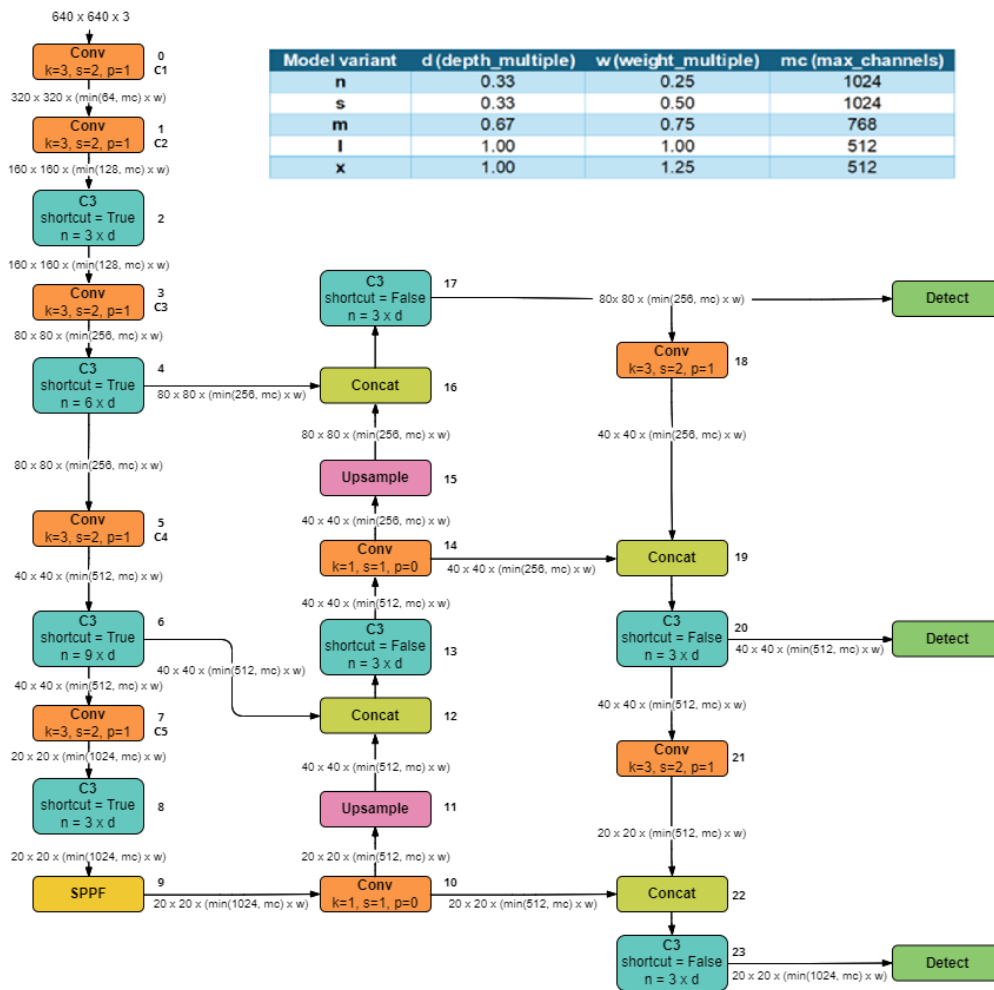


Figure 5. The Layout of the YOLOv5 Algorithm's Architecture

The difference between YOLOv5 and YOLOv8 is that one uses C3, and the second uses C2f. Block C3 significantly reduces the number of model parameters without loss of accuracy. Thus, it reduces computational complexity and information loss. Block C3 is shown in Figure 6. The C2f module, which is a combination of the C3 module and the ELAN concept from YOLOv7, is introduced in YOLOv8. This enables the model to gather more intricate details about the gradient flow [21]. The C2f module, as depicted in Figure 6, comprises 2 ConvModules and n DarknetBottleNecks, which are interconnected via Split and Concat operations [22].

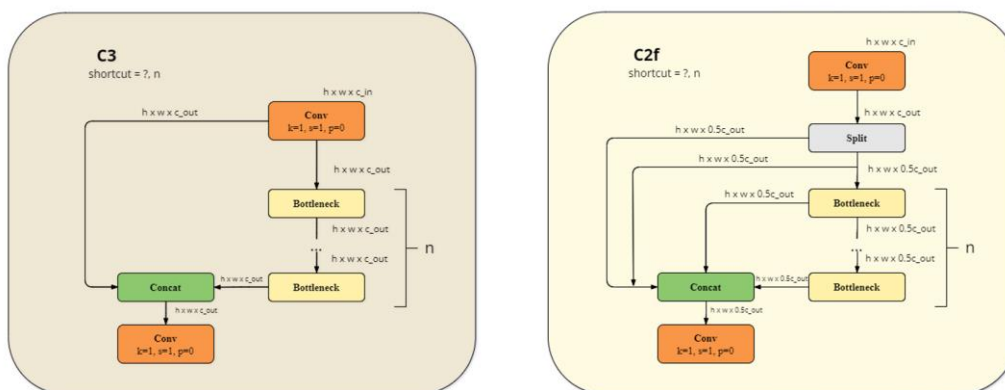


Figure 6. The Structure of the C3 and C2f Blocks

3.4. Neck

The Backbone links to the Neck at three distinct depths to combine features from various network layers. These fused features are then transmitted to the Head. The Neck incorporates path aggregation network (PAN) [23] and feature pyramid network (FPN) [24] structures to mitigate information loss caused by multiple convolutions. The Feature Pyramid Network (FPN) performs upsampling from top to bottom, enhancing the feature information in the lower-level feature map. Meanwhile, the Path Aggregation Network (PAN) downsamples from bottom to top, capturing additional information from the top-level feature map. These two feature outputs are combined to ensure accurate predictions for images of different sizes.

3.5. Head

The Head consists of three detection modules, which have been intentionally separated into distinct classification and regression tasks. This decoupling technique was initially proposed in YOLOX and YOLOv6 [25] specifically for anchor-free detection. In contrast to the YOLOv5 model's coupled Head, YOLOv8 adopts a decoupled head architecture. Specifically, we separate the classification and detection heads to enhance performance.

3.6. Evaluation Metrics

The performance of the model was assessed using metrics such as precision (P), recall (R), Intersection over Union (IOU) and mean average precision (mAP).

In the context of target detection, Intersection over Union (IOU) quantifies the agreement between the predicted and actual detection frames [26]. The IoU, represented by Equation 1, involves the comparison of the target box (Bgt) and the prediction box (B).

$$IoU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|} \quad (1)$$

Precision assesses the model's ability to predict positive instances correctly. It calculates the proportion of correct positive predictions out of all positive predictions made [27]. A higher precision score implies fewer false positives, indicating that the model is more effective at correctly identifying true positives. The formula for calculating precision is given in Equation 2.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

In machine learning, recall assesses the model's ability to capture all relevant positive examples. It is computed as the ratio of true positives to the sum of true positives and false negatives [27]. A higher recall score implies fewer false negatives, indicating that the model is more effective at identifying all positive instances. Equation 3 provides the formula for recall calculation.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Mean Average Precision (mAP) is a metric that evaluates the model's effectiveness in terms of object detection accuracy. It represents the average of precision scores at different levels of recall and is calculated using Equation 4 [28].

$$mAP = \frac{1}{N} \sum (precision \text{ at each recall level}) \quad (4)$$

The Mean Average Precision (mAP) is represented as mAP (50), which signifies the average precision value when the IoU threshold is set at 0.5. Additionally, mAP (50-95) denotes a range of IoU thresholds from 0.5 to 0.95, incrementing in steps of 0.05.

4. Results

Our training setup comprised Ultralytics YOLOv8.0.194 for object detection, Python 3.8.18 for scripting, PyTorch 2.1.2 with CUDA 11.8 for deep learning, and an NVIDIA GeForce RTX 3060 Laptop GPU for accelerated computations. The specific training parameters included 50 epochs, stochastic gradient descent (SGD) optimization, an initial learning rate of 0.01, and a momentum of 0.937.

In this work, the model was trained using different sizes of YOLOv5 and YOLOv8. The Table 1 shows the results of our trained models. Comparing the results, YOLOv5 models have higher accuracy than YOLOv8. More precisely, the highest result was 89.6%. And for YOLOv8, the highest accuracy is 87.4%. That is, the YOLOv5 model performed 3% better than YOLOv8.

Table 1. Results of our Trained Models

Model	Parameters	mAP(50)	mAP(50-95)
YOLOv8n	3011043	0.822	0.390
YOLOv8s	11135987	0.877	0.415
YOLOv8m	25856899	0.845	0.416
YOLOv8l	43630611	0.867	0.416
YOLOv8x	68153571	0.874	0.429
YOLOv5n	2508659	0.834	0.406
YOLOv5s	9122579	0.846	0.396
YOLOv5m	25065715	0.896	0.434
YOLOv5l	53164115	0.889	0.417
YOLOv5x	97200371	0.881	0.412

Figure 7 and Figure 8 show the mAP0.5 performance of the YOLOv8 and YOLOv5 models trained in the experimental environment of this work.

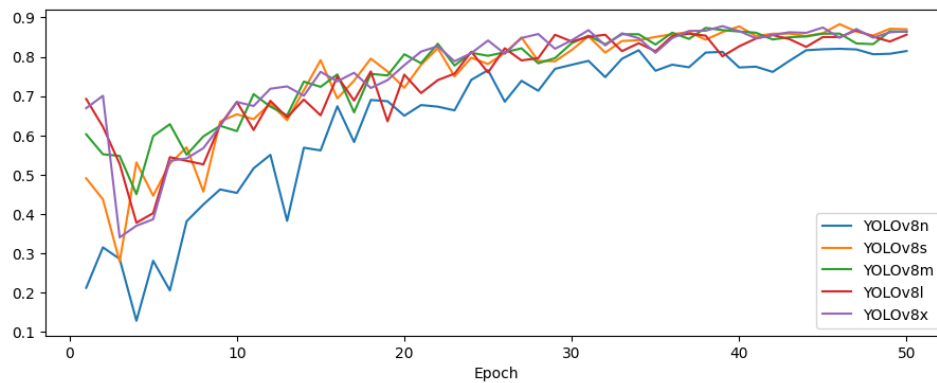


Figure 7. mAP with IoU=0.50 with the Training YOLOv8 Over 50 Epochs

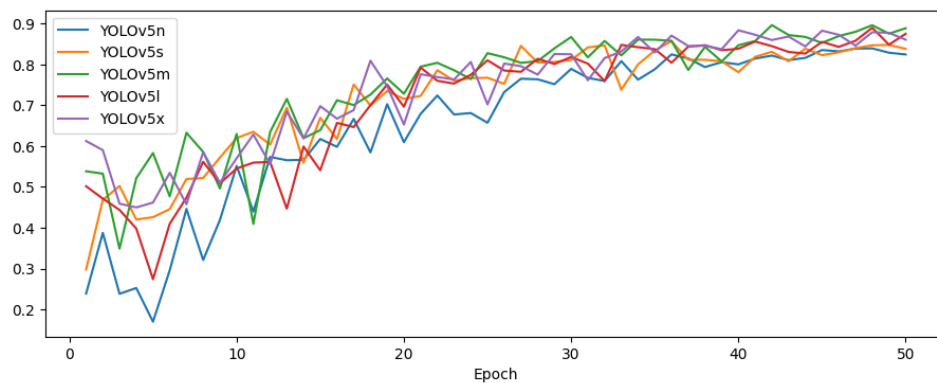


Figure 8. mAP with IoU=0.50 with the Training YOLOv5 Over 50 Epochs

Visualization of the detection effect of YOLOv8 and YOLOv5 models, several images are selected from the test dataset for detection comparison. Figure 6 in the results section shows cigarette detection using the YOLOv8 and YOLOv5 algorithms on multiple images. The apparent superiority of YOLOv5 over YOLOv8 in cigarette detection can be attributed to several factors, such as model architecture, hyperparameter tuning, object detection algorithm, and implementation details. Overall, the superior performance of YOLOv5 in Figure 9 suggests that it exhibits better accuracy and reliability in detecting cigarettes within the images than YOLOv8.



Figure 9. The Result of the Proposed Model

5. Conclusion

This article provides a significant contribution to the field of object detection, specifically in the identification of cigarettes in images. The study leverages the YOLOv8 algorithm, a powerful tool for object detection, and applies it to a critical public health issue: tobacco use. The comprehensive literature review offers a robust understanding of the current state of research, while the detailed methodology section ensures the study’s replicability.

First, our model is trained using the public dataset of similar jobs [29], achieving 78% accuracy. Then, we created our dataset to improve the accuracy of the model. First, the YOLOv8 model is trained using the public dataset [29] and achieved 78% accuracy. Second, the dataset is created to improve the accuracy of the model. Then, both models, YOLO8 and YOLO5, were trained using the dataset to perform a comparative analysis. The YOLOv8 model achieved an accuracy of 87.4%, while the YOLOv5 model reached 89.6% accuracy.

This study provides a comparative analysis of the YOLOv8 and YOLOv5 models, offering valuable insights for future research on cigarette detection. These findings could guide the development of more accurate and efficient models for cigarette detection in images, thereby aiding in the broader fight against tobacco use.

References

- [1] World Health Organization, "Tobacco" 2023. [Online]. Available: <https://www.who.int/news-room/factsheets/detail/tobacco>.
- [2] E. Arkin, N. Yadikar, X. Xu, A. Aysa and K. Ubul. "A survey: object detection methods from CNN to transformer," Proc. - Multimedia Tools and Applications, 2023.
- [3] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger" Proc. - arXiv preprint arXiv: 1612.08242, 2016.
- [4] J. Redmon, S. Divvala, R. Girshick and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection," Proc. - arXiv preprint arXiv: 1506.02640, 2016.
- [5] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement," Proc. - arXiv preprint arXiv: 1804.02767, 2018.
- [6] A. Bochkovskiy, C. Wang and M. Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection," Proc. - arXiv preprint arXiv: 2004.10934, 2020.
- [7] D. Jayakumar and S. Peddakrishna. "Performance Evaluation of YOLOv5-based Custom Object Detection Model for Campus-Specific Scenario," Proc. - International Journal of Experimental Research and Review, 2024.
- [8] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, K. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei and Wei, X. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," Proc. - arXiv preprint arXiv: 2209.02976, 2022.
- [9] C. Wang, A. Bochkovskiy and M. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Proc. - arXiv preprint arXiv: 2207.02696, 2022.
- [10] Ultralytics YOLOv8 Docs, "Introducing Ultralytics YOLOv8" 2023. [Online]. Available: <https://docs.ultralytics.com/#where-to-start>.
- [11] J. R. Macalisang, N. E. Merencilla, D. Ligayo. "Eye-Smoker: A Machine Vision-Based Nose Inference System of Cigarette Smoking Detection using Convolutional Neural Network," Proc. - 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS), 2020.
- [12] A. Khan, S. Khan, B. Hassan, and Z. Zheng. "CNN-Based Smoker Classification and Detection in Smart City Application,". Sensors 2022.
- [13] C. Santiago, M. Reyes, L. Tria. "Deep Convolutional Neural Network for Detection of Cigarette Smokers in Public Places: A Low Sample Size Training Data Approach," Proc. - 2022 International Conference on Decision Aid Sciences and Applications (DASA), 2022.
- [14] D. Zhang, C. Jiao, S. Wang, "Smoking Image Detection Based on Convolutional Neural Networks," Proc. - 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018.
- [15] C. Wang, T. Zheng, F. Sun and H. Lia "A Smoking Detection Algorithm Based on Improved YOLOV5," Proc. - 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), 2023.
- [16] Q. Ding, X. Dong, W. Guo, W. Zheng and Y. Pan, "Smoking Detection Algorithm Based On Improved YOLOv5," Proc. - 2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC), 2023.
- [17] J. Peng, C. Wang, Y. Li, H. Chen, "Substation Personnel Smoking Detection Based On GhostNetV2-YOLOv5," Proc. - 2023 6th International Symposium on Autonomous Systems (ISAS), 2023.
- [18] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. McQuade, and J. Solawetz. "Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark," Proc. - arXiv preprint arXiv: 2211.13523, 2022.
- [19] C. Wang, H. Mark Liao, Y. Wu, P. Chen, J. Hsieh, and I. Yeh. "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," Proc. - CVPR 2020 open access, 2020.
- [20] T. Huang, M. Cheng, Y. Yang, X. Lv, J. Xu. "Tiny Object Detection based on YOLOv5," Proc. - 5th International Conference on Image and Graphics Processing, 2022.
- [21] K. Jiang, T. Xie, R. Yan, X. Wen, D. Li, H. Jiang, N. Jiang, L. Feng, X. Duan, and J. Wang. "An Attention Mechanism-Improved YOLOv7 Object Detection Algorithm for Hemp Duck Count Estimation,". Proc. - Internet and Computers for Agriculture, 2022.
- [22] R. Ju, W. Cai, "Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm," Proc. - arXiv preprint arXiv:2304.05071v5, 2023.
- [23] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. "Path Aggregation Network for Instance Segmentation," Proc. - arXiv preprint arXiv:1612.03144, 2017.
- [24] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature Pyramid Networks for Object Detection," Proc. - arXiv preprint arXiv:1803.01534, 2018.
- [25] J. Terven, D. Cordova-Esparza, and J. Romero-Gonzalez. "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Proc. - Machine Learning and Knowledge Extraction, 2023.
- [26] M. Everingham, L. Gool, C. Williams, J. Winn and A. Zisserman. "The PASCAL Visual Object Classes (VOC) Challenge," Proc. - International Journal of Computer Vision, 2009.
- [27] J. Davis and M. Goadrich. "The relationship between Precision-Recall and ROC curves," Proc. - 23rd international conference on Machine learning, 2006.
- [28] R. Padilla, S. Netto and A. Eduardo. "A Survey on Performance Metrics for Object-Detection Algorithms," Proc. -

2020 International Conference on Systems, Signals and Image Processing (IWSSIP), 2020.

- [29] Dataset. “Cigarette Computer Vision Project”, 2022. [Online]. Available: <https://universe.roboflow.com/smoke-tamtu/cigarette-6ubdv>.

Author(s) Contributions

Yerniyaz Bakhytov: Data generation, Software, Methodology, Writing – Original Draft.

Cemil Öz: Methodology, Writing, review, and editing.

Conflict of Interest Notice

Authors declare no conflict of interest regarding the publication of this article.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of Data and Material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate™.

Detection and Analysis of Malicious Software Using Machine Learning Models

Ahmet Öztürk¹, Selman Hızal¹

¹Sakarya University of Applied Sciences, Department of Computer Engineering, Sakarya, Türkiye

Corresponding author:

Yermiyaz Bakhytov, Sakarya University of Applied Sciences, Department of Computer Engineering, Sakarya, Türkiye
ozturkahmet89@gmail.com



Article History:
Received: 24.05.2024
Accepted: 19.08.2024
Published Online: 26.08.2024

ABSTRACT

The continuous evolution of malware poses a significant challenge in cybersecurity, adapting to technological advancements despite implemented security measures. This paper introduces an innovative approach to enhance the detection of obfuscated malware through the integration of machine learning (ML). Utilizing a real-world dataset of prevalent malware types such as spyware, ransomware, and trojan horses, our study addresses the evolving challenges of cybersecurity. In this study, we evaluate the performance of ML algorithms for obfuscated malware detection using the CIC-MalMem-2022 dataset. Our analysis encompasses binary and multi-class classification tasks under various experimental conditions, including percentage splits and 10-fold cross-validation. The evaluated algorithms include Random Tree (RT), Random Forest (RF), J-48 (C4.5), Naive Bayes (NB), and XGBoost. Experimental results demonstrate the effectiveness of RF, J-48, and XGBoost in achieving high accuracy rates across different classification tasks. NB also shows competitive performance but faces challenges in handling imbalanced datasets and multi-class classification. Our findings highlight the importance of employing advanced ML techniques for enhancing obfuscated malware detection capabilities and provide valuable insights for cybersecurity practitioners and researchers. Future research directions include fine-tuning model hyperparameters, exploring ensemble learning approaches, and expanding evaluation to diverse datasets and real-world scenarios.

Keywords: Information security, Software analysis, Malware detection system, Machine learning

1. Introduction

The persistent evolution of malicious code, or malware, presents significant threats to internet-connected devices despite robust security measures. Malware, designed to perform unauthorized actions often without the user's knowledge, is utilized for malicious purposes such as stealing passwords, accessing confidential data, or corrupting system operations [1]. This poses profound challenges to the core aspects of information security: privacy, integrity, and availability [2]. Malware can expose sensitive organizational data (privacy), alter or corrupt records (integrity), and disrupt system functionality (availability) by deleting or overwriting files or damaging storage media. These characteristics make malware detection difficult, especially as it continually develops techniques to evade conventional detection methods [3].

Malware encompasses various forms, including worms, viruses, bots, trojan horses, ransomware, spyware, adware, spam, phishing, and rootkits, necessitating a comprehensive and nuanced detection strategy [4][5][6][7]. Traditional manual detection methods, while thorough, are impractically time-consuming and complex. This has driven the adoption of automated systems such as Machine Learning (ML) [8]. These intelligent systems can swiftly and accurately analyze data, learning from selected training datasets to optimize detection processes [9]. ML algorithms, varying in speed, accuracy, and precision, significantly impact these systems' outcomes [10]. For instance, ensemble ML techniques are well-suited to malware detection and characterization objectives [11]. By training on labeled datasets, supervised learning algorithms can achieve high accuracy rates in identifying certain types of malware [12]. In contrast, unsupervised learning algorithms can be effective in detecting unknown threats by discovering previously unidentified patterns and relationships in datasets [13]. Additionally, reinforcement learning and deep learning techniques can offer fast and flexible detection mechanisms in complex and dynamic threat environments [14]. In this context, ML-based systems go beyond traditional methods, allowing malware to be detected more effectively and efficiently [15].

Current methods for detecting malware, often involving detailed and slow analysis of computer memory, are impractical for real-world applications. There is a critical need for more efficient and effective solutions. Our proposed approach leverages features identified through memory analysis to enhance malware detection systems [16].

In this study, we use malware samples from distinct categories, such as ransomware, spyware, and trojan horses, to demonstrate our methodological approach. Specifically, we investigate the application of ML models to detect obfuscated malware using the CIC-MalMem-2022 dataset [1]. We conduct binary classification, determining the presence of malware, and multiclass classification, identifying specific malware families. We evaluate a range of traditional ML algorithms, including Random Tree (RT), Random Forest (RF), J-48, Naive Bayes (NB), and XGBoost.

The contribution of the paper can be summarized as follows:

- Evaluating the effectiveness of various machine learning algorithms against obfuscated malware using the CIC-MalMem-2022 dataset.
- Developing a machine learning-based system to address the risks posed by obfuscated malware.
- Providing a comprehensive categorization of the malware families within the dataset.

The remainder of this paper is structured as follows: Section II reviews recent works on obfuscated malware detection, Section III discusses the relevant background, Section IV presents the methodology, Section V details the experimental results, and the final section offers concluding remarks.

2. Related Works

Machine Learning (ML) has become pivotal in enhancing the detection of obfuscated malware within cybersecurity systems [1][17][18][19]. This section reviews several key contributions utilizing ML and Deep Learning (DL) techniques to address the challenges posed by sophisticated malware variants.

Yihan et al. [3] introduced a DL-powered hierarchical model to tackle data imbalance in malware datasets, achieving a binary classification accuracy of 99.67% with the CIC-MalMem-2022 dataset. Ghazi and Raghava [4] applied nature-inspired algorithms for feature selection, notably improving classification accuracy with ML techniques. Their approach demonstrated substantial efficacy in detecting malware in cloud networks, achieving a binary classification accuracy of 99.99% using the Mayfly Algorithm. Nugraha and Zeniarja [11] utilized a Decision Tree (DT) based algorithm for memory-based malware detection, achieving an accuracy of 99.982% in binary classification, with a notable reduction in false positives, highlighting the effectiveness of memory analysis in detecting malware behavior. Dener et al. [16] emphasized the importance of memory analysis in malware detection, employing various ML algorithms with Logistic Regression (LR), showing the highest accuracy of 99.97% in their studies. Tidjon and Khomh [20] proposed using topological data analysis to efficiently identify complex malware patterns, suggesting that TDA techniques combined with ML could enhance the robustness and performance of malware detection systems. Naeem et al. [21] developed a dynamic method for malware detection and classification using anti-analysis techniques, transforming memory dumps into grayscale images. Their hybrid model, combining Convolutional Neural Networks (CNNs) and a Multilayer Perceptron (MLP), achieved high accuracy on Windows (99.1%), Android (94.3%), and Windows obfuscated malware (99.8%). The study suggests further enhancements in training efficiency and resilience against adversarial attacks. Al-Qudah et al. [22] introduced a model combining One-Class Support Vector Machine (SVM) with Principal Component Analysis (PCA), achieving a significant improvement in the detection of memory dump malware, with an accuracy of 99.4% for binary classification. Smith et al. [23] the use of ML for malware detection, employing a range of algorithms to achieve 99.99% accuracy for binary classification, suggesting further research into feature selection and genetic algorithms to enhance detection capabilities.

Mezina and Burget [24] tackled the challenge of detecting obfuscated malware in memory using AI. They utilized the CIC-MalMem-2022 dataset and explored several ML techniques, including Random Forest (RF) and a newly proposed dilated CNN. The RF method achieved an accuracy of 99.99% for binary classification, while the dilated CNN excelled in classifying malware families with an accuracy of 83.53% for multiclass(4) classification. The study highlights the need for improved classification methods to keep up with cybersecurity threats. Roy et al. [25] created MalHyStack, a combined model that uses group learning methods to detect hidden malware very accurately. They stress the importance of detecting these threats early in cybersecurity. They achieved 99.98% accuracy for binary classification, 85.04% accuracy for multiclass(4) classification, and 70.29% accuracy for multiclass(16) classification.

The existing literature often focuses on binary classification, categorizing objects as benign or malware. For instance, Yihan et al. [3] achieved a 99.67% accuracy using a DL-powered hierarchical model, while Ghazi and Raghava [4] attained a 99.99% accuracy with the Mayfly Algorithm for feature selection. Nugraha and Zeniarja's [11] DT-based approach yielded a 99.982% accuracy, and Dener et al. [16] demonstrated a 99.97% accuracy using Logistic Regression. These studies underscore the efficacy of binary classification techniques in identifying malware.

However, this method is insufficient for comprehensive malware analysis. Therefore, there is a need to develop methods that can identify sub-categories of malware. Recent studies have achieved multi-class classification into four groups: benign, ransomware, spyware, and trojan horse attacks. Other studies have gone further, classifying these into 16 sub-types of attacks. Despite these advances, there is still a need for improved results in the literature.

In our research, we addressed this limitation by implementing multilevel classification, categorizing malware into more granular groups. Specifically, we performed classification into 4 main categories (benign, ransomware, spyware, and trojan horse) and further into 16 sub-categories. This approach not only enhances the granularity of malware detection but also improves the ability to identify and respond to new and emerging types of malware. Enhancing these methods could make it easier to detect new and emerging types of malware automatically.

By advancing beyond binary classification to multilevel classification, our study contributes to the field by providing a more detailed and nuanced understanding of malware, which is critical for developing robust malware detection systems. This method allows for more precise identification of malware types and can significantly aid in the development of targeted defense mechanisms against specific types of malware threats.

3. Background

In this section, we provide a contextual overview of malware types with their analysis and detection methods, focusing on the importance of memory analysis in identifying obfuscated malware. Leveraging ML algorithms is highlighted as a key strategy for effectively detecting and mitigating sophisticated malware threats. We then introduce the CIC-MalMem-2022 dataset, a valuable resource extensively employed in malware analysis. Notable for its representation of real-world obfuscated malware scenarios, the dataset's characteristics are outlined, emphasizing its relevance in evaluating ML models for malware detection. Additionally, we discuss the diverse range of ML algorithms utilized to detect obfuscated malware using the CIC-MalMem-2022 dataset and tailored feature extraction methods to enhance detection accuracy. Finally, we present performance metrics utilized for evaluating the effectiveness of ML models, providing quantitative insights into their detection capabilities.

3.1. Overview of Malware Types

In cybersecurity, understanding malware types is crucial for developing effective defense strategies against evolving cyber threats. This study discusses 3 different malware type categories: spyware, ransomware, and trojan horse. Ransomware, with variants such as Ako, Conti, and Maze, poses a significant threat by encrypting files or systems and demanding ransom payments. Spyware, including programs like 180solutions and Gator, compromises privacy and security by secretly monitoring user activity. Trojans like Emotet and Zeus camouflage themselves as legitimate software to facilitate unauthorized access and data leakage. Recognizing the characteristics and tactics of these malware strains allows cybersecurity practitioners to implement robust defenses and effectively mitigate risks.

3.2. Obfuscated Malware Analysis

Obfuscated malware presents a formidable challenge to traditional cybersecurity defenses due to its sophisticated evasion techniques to conceal malicious intent. These evasion strategies include code obfuscation, polymorphism, and encryption, which render malware payloads virtually undetectable to conventional signature-based detection systems. By disguising malicious code within legitimate-looking programs, obfuscated malware evades detection by antivirus software and intrusion detection systems, posing significant threats to individual users and organizations.

In response to the escalating threat posed by obfuscated malware, researchers have turned to advanced detection methodologies, with ML emerging as a promising approach. ML algorithms can discern subtle patterns and anomalies indicative of obfuscated malware within memory dumps. Leveraging labeled datasets such as the CIC-MalMem-2022, researchers can train ML models to recognize obfuscated malware behaviors and distinguish them from benign software. However, the dynamic nature of obfuscated malware necessitates continual adaptation and refinement of detection strategies to combat evolving threats effectively.

3.3. Dataset Description

Canadian Institute for Cybersecurity (CIC) provides data sets for studies in cyber security. The CIC-MalMem-2022 dataset is the most up-to-date dataset created in the malware category by Carrier et al. [1]. This dataset contains a modern collection of samples encompassing obfuscated malware and benign classes. The dataset included prominent malware families such as ransomware, spyware, and trojan horses to simulate real-world conditions. Within the scope of the research, 100 to 200 malware samples were collected from three main categories of trojan, ransomware, and spyware, and five subcategories under each of these categories, as shown in Figure 1.



Figure 1. The CIC-MalMem-2022 Dataset Distribution

The CIC-MalMem-2022 dataset creation process stages are shown in Figure 2. The first stage focused on memory dumping, employing VirtualBox as a sandbox to capture memory snapshots from a Microsoft Windows 10 operating system, ensuring a realistic representation of contemporary malware. Automation was introduced to execute 2,916 malware samples in a virtual machine, with concurrent benign processes to maintain realism. The resulting 29,298 malicious memory dumps were balanced using the SMOTE algorithm [26]. The second stage involved transferring dump files to a Kali Linux machine for feature extraction using the Volatility Memory Analyzer (VolMemLyzer) [27], a Python project designed to facilitate the extraction of memory features, enhancing the analysis of malicious activities within a memory snapshot using the Volatility tool. VolMemLyzer efficiently captures the essential features for in-depth memory forensics and malware detection. Each sample is represented by a vector of numerical values derived from memory dump information, with 26 new extended feature groups, including Malfind (3), Ldrmodule (3), Handles (11), Process View (6), and Apihooks (3), totaling 58 features. Finally, the third stage encompassed feature extraction from memory dump files and creating a combined comma-separated values (CSV) file [1].

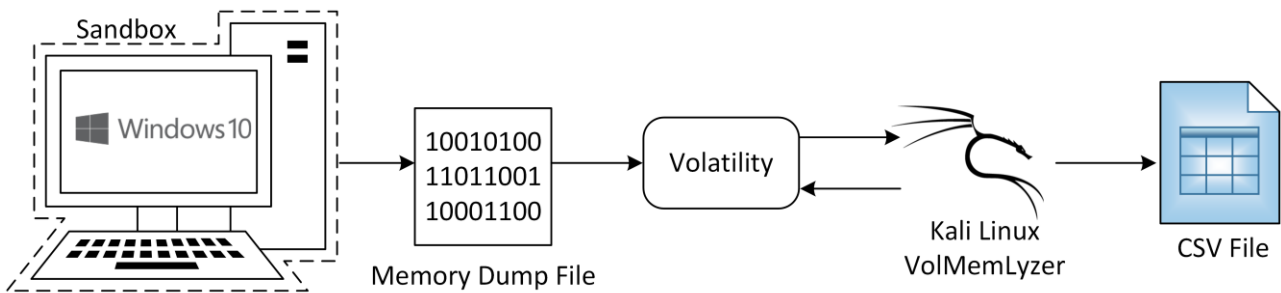


Figure 2. The CIC-MalMem-2022 Dataset Generation Processes

The CIC-MalMem-2022 dataset contains 58,596 memory dump samples. As shown in Table 1, the distribution of samples is well-balanced between benign and malware categories, with each class accounting for 50% of the total samples.

Table 1. The Number of Samples for CIC-MalMem-2022 Dataset

Class	Number of Samples	Ratio %
Benign	29,298	50
Malware	29,298	50
Total	58,596	100

The malware class is further divided into 3 categories and 15 subcategories, which are evenly distributed, as illustrated in Table 2.

Table 2. The Malware Subclasses in CIC-MalMem-2022 Dataset

Malware Details	Class	Number of Samples	Ratio %
Ransomware (9,791)	Ako	2,000	3.413
	Conti	1,988	3.413
	Maze	1,958	3.342
	Pysa	1,717	2.930
	Shade	2,128	3.632
Spyware (10,020)	180solutions	2,000	3.413
	CWS	2,000	3.413
	Gator	2,200	3.755
	TIBS	1,410	2.406
	Transponder	2,410	4.110
Trojan Horse (9,487)	Emotet	1,967	3.357
	Reconyc	1,570	2.679
	Refroso	2,000	3.413
	Scar	2,000	3.413
	Zeus	1,950	3.328

Ransomware (9,791 samples): This category includes strains such as Ako, Conti, Maze, Pysa, and Shade, which encrypt files and demand a ransom for decryption. Each strain has 1,717 and 2,128 samples, making up about 33.4% of the malware data.

Spyware (10,020 samples): This category consists of 180solutions, CWS, Gator, TIBS, and Transponder, which are programs that gather information about a person or organization without their knowledge. Each strain has between 1,410 and 2,410 samples, accounting for 34.3% of the malware data.

Trojan Horse (9,487 samples): This category includes Emotet, Reconyc, Refroso, Scar, and Zeus, programs that deceive users about their true intent, often appearing as legitimate software. Each strain has between 1,570 and 2,000 samples, making up 32.3% of the malware data.

3.4. Machine Learning Models

The ML models for classifying the CIC-MalMem-2022 dataset are generally explained in this section. We used RT, RF, J-48, NB, and XGBoost.

RT is an ML algorithm that constructs a decision tree by randomly selecting a subset of features at each node to split on. This randomness helps to reduce overfitting and improve generalization performance. RF enhances accuracy and reduces overfitting by averaging the predictions of multiple decision trees. C4.5, or J-48 in the Weka implementation, is a decision tree algorithm derived from the Iterative Dichotomiser 3 (ID3) algorithm. It selects crucial attributes from the information gain ratio and generates if-then rules from the trained trees. C4.5 is known for its robust performance across different datasets.

NB is a probabilistic classifier that applies Bayes' theorem for classification. It assumes that characteristics are conditionally independent given the class label, which simplifies the calculation of posterior probabilities. Despite its 'naive' assumption, NB often performs well in practice, especially for text classification tasks.

XGBoost is an optimized gradient-boosting algorithm widely used for classification and regression tasks. It builds a series of decision trees sequentially, where each tree attempts to correct the errors made by the previous ones. XGBoost is known for its computational efficiency and high performance on structured/tabular data.

While the dataset is not perfectly balanced, it is well-distributed and suitable for robust malware classification. Our study effectively utilizes this dataset to perform multilevel classification, yielding significant and reliable results.

4. Methodology

The system architecture and performance evaluations performed in this study are shown in Figure 3.

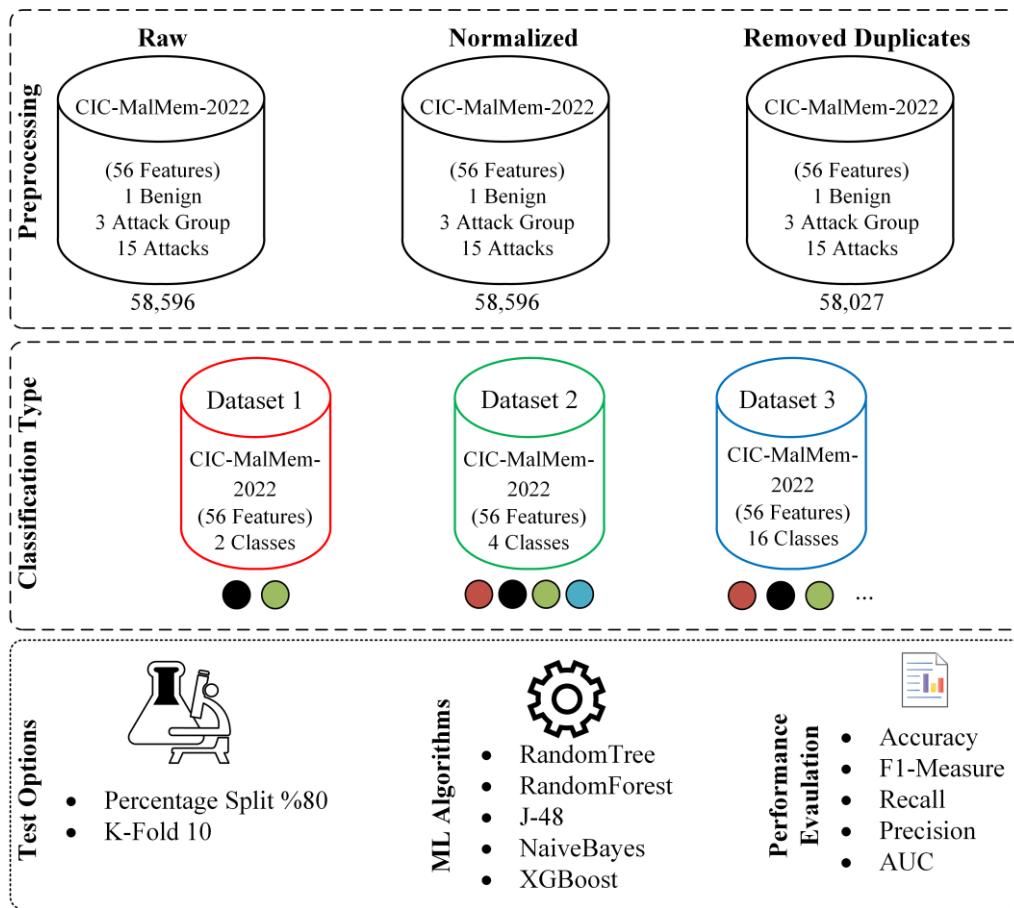


Figure 3. Proposed Malicious Software Detection System

Pre-processing steps, dataset creation, testing, and evaluation of machine learning (ML) methods were conducted using the CIC-MalMem-2022 dataset. Initially, all processes were applied to the raw dataset, and the corresponding tests were executed. It was noted that the initial low performance could be attributed to the lack of normalization and the presence of duplicate records. Consequently, all tests were repeated using a dataset from which duplicate records had been removed. The best performance was achieved with the duplicates removed dataset resulting in 58,027 records from the original 58,596 records.

The classification process began by organizing the data into binary classes (benign and malware). Subsequently, multiclass datasets with 4 and 16 classes were created. Various testing strategies were employed, including test split and k-fold cross-validation. For the percentage split test, 80% of the dataset was used for training, while the remaining 20% was used for testing. In the k-fold cross-validation method, the dataset was divided into 10 parts, with each part being used for testing in turn. The ML methods tested included Random Tree (RT), Random Forest (RF), J-48, Naive Bayes (NB), and XGBoost, all of which are commonly used in the literature. Performance metrics such as Accuracy, F1-Score, Recall, and Precision were calculated to assess the test results.

4.1. Data Preprocessing

Increasing the performance of machine learning methods depends significantly on data preprocessing steps. Therefore, analyzing the data well and carrying out the necessary pre-processing is very important. Normalization operations were performed on the dataset in our study. After this process, duplicate records in the data set were cleared. Thus, a more consistent and cleaner data set was obtained. At the end of these processes, sub-datasets such as binary, multiclass(4), and multiclass(16) were obtained from the data set containing 58,027 records, as shown in Figure 4.

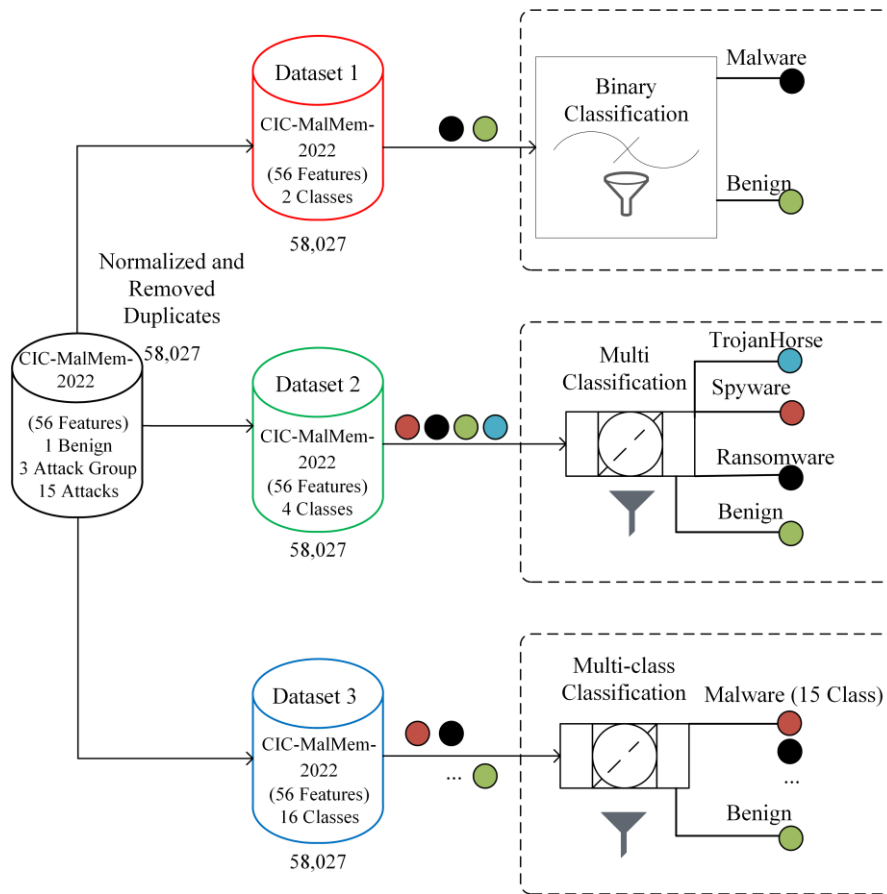


Figure 4. CIC-MalMem-2022 Dataset Preprocessing Operations

4.2. Experimental Setting

We utilized Weka version 3.9.6 [28] for training the machine learning models and conducting the evaluations. Table 3 details the hardware and software environment used in this study. The performance measurements were executed on a Windows operating system, powered by a CPU model i7-13700K @ 3.40GHz with 32 GB of RAM. The GPU employed was an NVIDIA GeForce® RTX 4070 Ti with 12 GB of memory.

Table 3. Hardware and Software Features for the Experimental Evaluations

Hardware / Software	Features
Operating System	Windows 11, 64 bit
Weka	3.9.6
CPU	13th Gen Intel(R) Core(TM) i7-13700K @ 3.40GHz
RAM	32 GB
Video Graphics Card	NVIDIA GeForce® RTX4070Ti

4.3. Evaluation Metrics

To assess the performance of the CIC-MalMem-2022 dataset with different algorithms, various performance metrics were used. The evaluations were performed using the WEKA [28]. Additionally, the confusion matrix used in computing these performance metrics is explained in Table 4.

Table 4. Confusion Matrix for Performance Calculation

Class/Attack Type		Predicted Class	
		Benign	Malware
True Class	Benign	True Positive (TP)	False Negative (FN)
	Malware	False Positive (FP)	True Negative (TN)

Accuracy: This metric indicates the percentage of correct predictions made by the model out of all predictions. It is calculated using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN} \quad (1)$$

Recall: Recall measures the percentage of actual positives that were correctly identified by the model. It is calculated as:

$$\text{Recall} = \frac{TP}{(TP + FN)} \quad (2)$$

Precision: Precision indicates the percentage of positive predictions that were actually correct. It is computed as:

$$\text{Precision} = \frac{TP}{(TP + FP)} \quad (3)$$

F-Measure: The F1 score, representing the harmonic mean of Precision and Recall, is calculated as follows:

$$F - \text{Measure} = 2 \times \left(\frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \right) \quad (4)$$

5. Experimental Results

This section presents the experimental studies conducted on the CIC-MalMem-2022 dataset. First of all, binary classification is given in the study. Then, the results obtained by considering multiclass classification with 4 and 16 classes are provided.

5.1. Binary Classification Results

For the binary classification task, algorithms like RF, J-48, and XGBoost demonstrate outstanding accuracy, with scores exceeding 99% in most cases. NB also performs reasonably well, albeit with slightly lower accuracy than other algorithms. XGBoost achieves perfect accuracy in both test modes, indicating its robustness in detecting obfuscated malware instances.

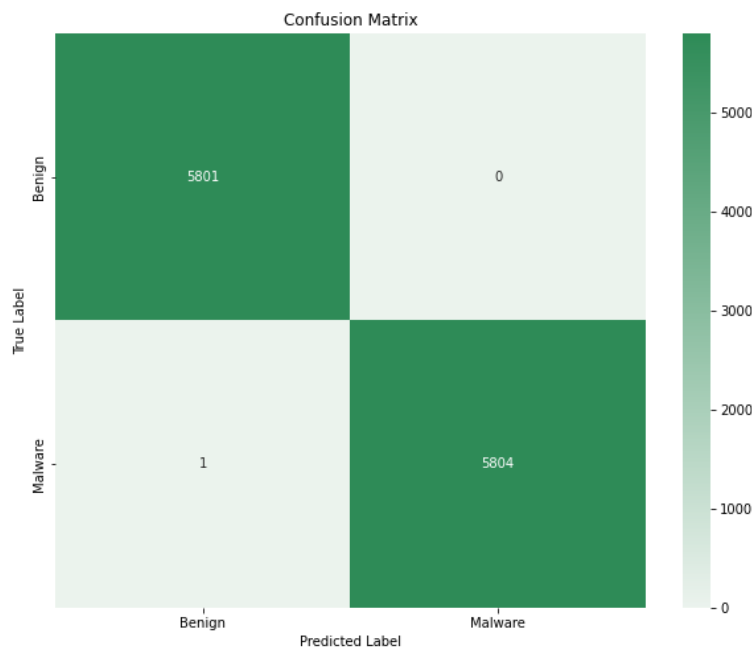


Figure 5. Confusion Matrix for XGBoost 80% Split for Binary Classification

Figure 5 shows the confusion matrix for binary classification utilizing XGBoost, demonstrating classification performance with 99.99% accuracy. The model showcases its exceptional ability to distinguish between different classes of obfuscated malware.

5.2. Multiclass Classification Results

In the multi-class classification scenarios (Multi-4 and Multi-16), RF, J-48, and XGBoost consistently deliver high accuracy rates above 70%. However, NB struggles to maintain comparable performance, especially in handling imbalanced datasets and distinguishing between multiple classes.

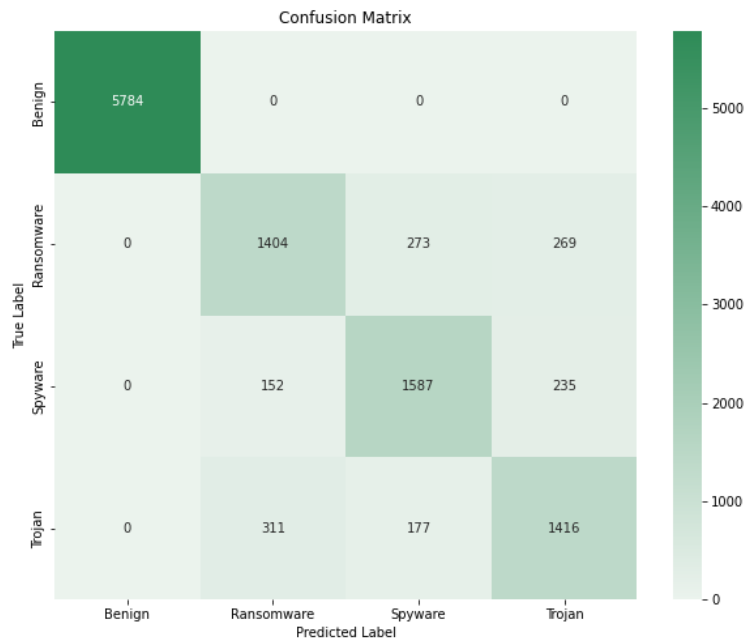


Figure 6. Confusion Matrix for XGBoost 80% Split for Multiclass(4) Classification

Figure 6 shows a confusion matrix with an accuracy of 87.79% and relatively balanced classification performance across all classes, as evidenced by the diagonal dominance and nonzero values outside the main diagonal. The classification report further reveals precision, recall, and F1-score metrics for each class, providing insights into the model’s ability to identify instances of obfuscated malware within each category correctly.

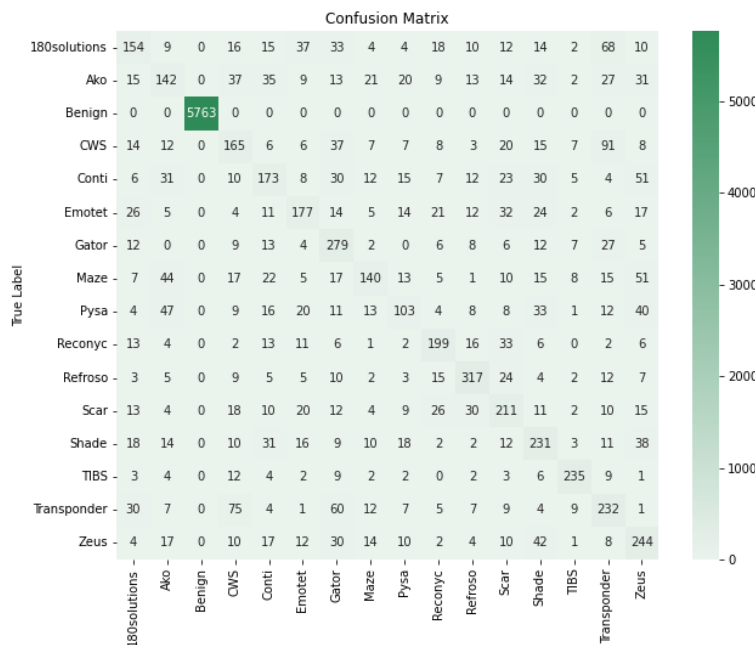


Figure 7. Confusion Matrix for XGBoost 80% Split for Multiclass(16) Classification

Figure 7 shows the confusion matrix for multi(16) classification with XGBoost. The XGBoost model trained with an 80% split achieved an accuracy of 75.49%. The confusion matrix indicates varying levels of precision and recall across different classes, with some classes showing more robust performance than others. Despite an overall macro-average F1-score of 0.54, indicating moderate classification performance, the model exhibits a relatively high mean absolute error of 1.5513, suggesting

some discrepancy between predicted and actual values. Further optimization may be necessary to enhance the model's accuracy and robustness across all classes.

5.3. Performance Comparisons

Table 5 shows the performance results of ML algorithms applied to the CIC-MalMem-2022 dataset after removing duplicates. The results are presented for both an 80% / 20% train-test split and a 10-fold cross-validation setting, covering binary, multi(4), and multi(16) classification tasks. The evaluation metrics used include Accuracy (ACC), True Positive Rate (TPR), False Positive Rate (FPR), Precision, F-Measure.

Table 5. Experimental Results on the CIC-MalMem-2022 Dataset with Removing Duplicates

Dataset	Test Mode	Algorithms	ACC	TPR	FPR	Precision	F-Measure
CIC-MalMem-2022 (Binary-2)	percentage (80%)	Random Tree	99.96	1.00	0.000	1.00	1.00
		Random Forest	99.99	1.00	0.000	1.00	1.00
		J-48	99.97	1.00	0.000	1.00	1.00
		Naive Bayes	98.87	0.98	0.011	0.98	0.98
		XG-Boost	99.99	1.00	-	1.00	1.00
CIC-MalMem-2022 (Binary-2)	k-fold 10	Random Tree	99.96	1.00	0.000	1.00	1.00
		Random Forest	99.98	1.00	0.000	1.00	1.00
		J-48	99.96	1.00	0.000	1.00	1.00
		Naive Bayes	98.89	0.98	0.011	0.98	0.98
		XG-Boost	99.87	1.00	-	1.00	1.00
CIC-MalMem-2022 (Multi-4)	percentage (80%)	Random Tree	83.14	0.83	0.034	0.83	0.83
		Random Forest	87.61	0.87	0.025	0.87	0.87
		J-48	86.84	0.86	0.026	0.86	0.86
		Naive Bayes	67.80	0.67	0.064	0.71	0.63
		XG-Boost	87.79	0.88	-	0.88	0.88
CIC-MalMem-2022 (Multi-4)	k-fold 10	Random Tree	83.53	0.83	0.033	0.83	0.83
		Random Forest	87.48	0.87	0.025	0.87	0.87
		J-48	86.76	0.86	0.026	0.86	0.86
		Naive Bayes	68.24	0.68	0.063	0.72	0.64
		XG-Boost	82.84	0.83	-	0.83	0.83
CIC-MalMem-2022 (Multi-16)	percentage (80%)	Random Tree	70.12	0.70	0.011	0.70	0.70
		Random Forest	74.66	0.74	0.009	0.74	0.74
		J-48	74.76	0.74	0.009	0.74	0.74
		Naive Bayes	57.17	0.57	0.016	0.61	0.55
		XG-Boost	75.49	0.75	-	0.76	0.75
CIC-MalMem-2022 (Multi-16)	k-fold 10	Random Tree	70.16	0.70	0.011	0.70	0.70
		Random Forest	75.42	0.75	0.009	0.75	0.75
		J-48	75.12	0.75	0.009	0.00	0.75
		Naive Bayes	56.99	0.57	0.016	0.60	0.54
		XG-Boost	71.33	0.71	-	0.71	0.71

Overall, the results underscore the efficacy of ML algorithms, mainly RF, J-48, and XGBoost, in detecting obfuscated malware. These findings provide valuable insights for cybersecurity practitioners and researchers developing more effective defense mechanisms against sophisticated malware threats.

5.4. Discussion

Despite the successes reported in recent studies, several limitations are evident, indicating substantial opportunities for further research. Most approaches require large and balanced datasets, which are not always available, particularly for new or emerging malware variants. There are also concerns about the generalizability of these models across different platforms or under real-world conditions, as many were tested in controlled environments. Furthermore, the robustness of these systems against adversarial attacks, where attackers intentionally modify malware to evade detection, remains a significant challenge. These limitations underscore that while the field has made considerable advances, there is still much work to be done to enhance the efficacy, efficiency, and adaptability of ML models in the ever-evolving landscape of malware detection.

5.5. Comparative Analysis with Existing Methods

In the literature, some studies only perform binary classification with the CIC-MalMem-2022 dataset [1], [3], [4], [16], [23], [29]. For binary classification, it is seen that many studies achieve results above 99%. There are studies on binary and multi-class (4) classification in the literature [24]. Some studies deal with sub-malware types in the entire dataset, such as binary, multi-class (4) [24], and multi-class (16) [25], [18], [19], [30].

When we compare these studies with our proposed research, the best results were obtained with 87.79% accuracy with XGBoost in multi-class (4) classification and 75.49% accuracy in multi-class (16) classification, as shown in Table 6. These results demonstrate significant improvements over previous works, especially in the context of multilevel classification, where our model achieves higher accuracy rates in both multi-class (4) and multi-class (16) settings. This is a notable advancement given the complexity and challenges associated with multilevel malware classification.

Our study contributes to the literature by addressing the limitations of existing binary classification approaches and advancing the field through the implementation of multilevel classification. This approach allows for more granular detection of malware, providing a detailed and nuanced understanding of different malware types. It enhances the ability to identify and respond to new and emerging malware threats, which is crucial for developing robust and adaptable malware detection systems.

Table 6 summarizes the recent studies related to the CIC-MalMem-2022 dataset, highlighting the comparative performance of different models. Our results not only demonstrate the effectiveness of our approach but also emphasize the potential for further improvements in the field of malware detection.

Table 6. Recent Studies Related to the CIC-MalMem-2022 Dataset

Authors	Model Tested	Best Model	Accuracy (%)		
			Binary	Multiclass (4)	Multiclass (16)
Carrier et al. [1]	NB, RF, DT and meta-learnerLR	LR	99.00	-	-
Yihan et al. [3]	Hierarchical Detection Model	DNN	99.67	-	-
Ghazi and Raghava [4]	RF, SVM, kNN	RF	99.99	-	-
Dener et al. [16]	RF, DT, GBT, LR, NaiveBayes,	LR	99.97	-	-
Smith et al. [23]	DT, RF, Ada Boost, KNN, SGD,	DT	99.99	-	-
Roshan and Zafar. [29]	NN, Adaptive RF, KNN, Voting	EnsAdp_CIDS	99.85	-	-
Mezina and Burget [24]	LR, DT, MLP, kNN, SVM, D-CNN	D-CNN	99.99	83.53	-
Cevallos-Salaset al. [18]	DT, RF, SVM, LDA, GNB	LR+DNN	99.70	75.40	68.20
Shafin et al. [19]	RobustCBL, CompactCBL	RobustCBL	99.96	84.56	72.60
Abualhaj et al. [30]	KNN	KNN	99.97	82.21	66.93
Roy et al. [25]	Extra Trees, XGBoost, SVM, kNN,	MalHyStack	99.98	85.04	70.29
Our Model 2024	RT, RF, J-48, NaiveBayes, XGBoost	XGBoost	99.99	87.79	75.49

6. Conclusion

In this study, we comprehensively evaluated ML algorithms for detecting obfuscated malware using the CIC-MalMem-2022 dataset. Our analysis encompassed binary and multi-class classification tasks under different experimental conditions, including percentage splits and 10-fold cross-validation. The results highlight the effectiveness of RF, J-48 (C4.5), and XGBoost algorithms in achieving high accuracy across various classification tasks. These algorithms consistently outperformed others, showcasing their robustness in identifying obfuscated malware patterns. Random Tree also exhibited commendable performance, albeit slightly lower than the algorithms above. Furthermore, Naive Bayes demonstrated competitive performance but faced challenges handling multi-class classification and imbalanced datasets, as evidenced by lower accuracy rates and higher mean absolute error values.

Overall, the findings show the importance of advanced ML techniques for enhancing obfuscated malware detection capabilities. The identified top-performing algorithms can be useful tools for cybersecurity researchers to develop more effective defense mechanisms against obfuscated malware. Future research efforts could focus on fine-tuning model hyperparameters, optimizing data preprocessing techniques, and further exploring ensemble learning approaches to improve detection accuracy and resilience against obfuscated malware attacks.

References

- [1] T. Carrier, P. Victor, A. Tekeoglu, and A. Habibi Lashkari, "Detecting Obfuscated Malware using Memory Feature Engineering," in *International Conference on Information Systems Security and Privacy*, 2022. doi: 10.5220/0010908200003120.
- [2] Z. A. El Houda, "Cyber Threat Actors Review: Examining the Tactics and Motivations of Adversaries in the Cyber Landscape," in *Cyber Security for Next-Generation Computing Technologies*, 2024. doi: 10.1201/9781003404361-5.
- [3] Y. Li, Z. Liu, X. Guan, Z. Wang, X. Guo, and S. Wang, "Hierarchical Obfuscation Malware Detection Method Based on Deep Learning," in *EEI 2022 - 4th International Conference on Electronic Engineering and Informatics*, 2022.
- [4] M. R. Ghazi and N. S. Raghava, "Machine Learning Based Obfuscated Malware Detection in the Cloud Environment with Nature-Inspired Feature Selection," in *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies, IMPACT 2022*, 2022. doi: 10.1109/IMPACT55510.2022.10029271.
- [5] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 1, 2024, doi: 10.1186/s42400-024-00205-z.
- [6] B. Janet, A. Nikam, and J. A. Kumar R, "Real Time Malicious URL Detection on twitch using Machine Learning," in *Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022*, 2022. doi: 10.1109/ICEARS53579.2022.9751862.
- [7] M. Hakimi, E. Ahmady, A. K. Shahidzay, A. W. Fazil, M. M. Quchi, and R. Akbari, "Securing Cyberspace: Exploring the Efficacy of SVM (Poly, Sigmoid) and ANN in Malware Analysis," *Cognizance Journal of Multidisciplinary Studies*, vol. 3, no. 12, 2023, doi: 10.47760/cognizance.2023.v03i12.017.
- [8] S. Altaha and K. Riad, "Machine Learning in Malware Analysis: Current Trends and Future Directions," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 1, 2024, doi: 10.14569/IJACSA.2024.01501124.
- [9] V. Vijayaraj, M. Balamurugan, and M. Oberai, "Machine learning approaches to identify the data types in big data environment: An overview," *The Scientific Temper*, vol. 14, no. 03, 2023, doi: 10.58414/scientifictemper.2023.14.3.60.
- [10] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, 2024, doi: 10.1016/j.heliyon.2023.e23574.
- [11] A. Nugraha and J. Zeniarja, "Malware Detection Using Decision Tree Algorithm Based on Memory Features Engineering," *Journal of Applied Intelligent System*, vol. 7, no. 3, 2022, doi: 10.33633/jais.v7i3.6735.
- [12] Akoh Atadoga, Enoch Oluwadamilade Sodiya, Uchenna Joseph Umoga, and Olukunle Oladipupo Amoo, "A comprehensive review of machine learning's role in enhancing network security and threat detection," *World Journal of Advanced Research and Reviews*, vol. 21, no. 2, 2024, doi: 10.30574/wjarr.2024.21.2.0501.
- [13] L. Zhang and Q. Yan, "Detect malicious websites by building a neural network to capture global and local features of websites," *Comput Secur*, vol. 137, 2024, doi: 10.1016/j.cose.2023.103641.
- [14] M. Kozák, M. Jureček, M. Stamp, and F. Di Troia, "Creating valid adversarial examples of malware," *Journal of Computer Virology and Hacking Techniques*, 2024, doi: 10.1007/s11416-024-00516-2.
- [15] K. Shaukat, S. Luo, and V. Varadharajan, "A novel machine learning approach for detecting first-time-appeared malware," *Eng Appl Artif Intell*, vol. 131, 2024, doi: 10.1016/j.engappai.2023.107801.
- [16] M. Dener, G. Ok, and A. Orman, "Malware Detection Using Memory Analysis Data in Big Data Environment," *Applied Sciences (Switzerland)*, vol. 12, no. 17, 2022, doi: 10.3390/app12178604.
- [17] M. A. Hossain *et al.*, "AI-enabled approach for enhancing obfuscated malware detection: a hybrid ensemble learning with combined feature selection techniques," *International Journal of System Assurance Engineering and Management*, 2024, doi: 10.1007/s13198-024-02294-y.
- [18] D. Cevallos-Salas, F. Grijalva, J. Estrada-Jimenez, D. Benitez, and R. Andrade, "Obfuscated Privacy Malware Classifiers Based on Memory Dumping Analysis," *IEEE Access*, vol. 12, 2024, doi: 10.1109/ACCESS.2024.3358840.
- [19] S. S. Shafin, G. Karmakar, and I. Mareels, "Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications," *Sensors*, vol. 23, no. 11, 2023, doi: 10.3390/s23115348.
- [20] L. N. Tidjon and F. Khomh, "Reliable malware analysis and detection using topology data analysis," *arXiv preprint arXiv:2211.01535*, 2022.
- [21] H. Naeem, S. Dong, O. J. Falana, and F. Ullah, "Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification," *Expert Syst Appl*, vol. 223, 2023, doi: 10.1016/j.eswa.2023.119952.
- [22] M. Al-Qudah, Z. Ashi, M. Alnabhan, and Q. Abu Al-Haija, "Effective One-Class Classifier Model for Memory Dump

- Malware Detection,” *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, 2023, doi: 10.3390/jsan12010005.
- [23] D. Smith, S. Khorsandroo, and K. Roy, “Supervised and Unsupervised Learning Techniques Utilizing Malware Datasets,” in *2023 IEEE 2nd International Conference on AI in Cybersecurity, ICAIC 2023*, 2023. doi: 10.1109/ICAIC57335.2023.10044169.
- [24] A. Mezina and R. Burget, “Obfuscated malware detection using dilated convolutional network,” in *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 2022. doi: 10.1109/ICUMT57764.2022.9943443.
- [25] K. S. Roy, T. Ahmed, P. B. Udas, M. E. Karim, and S. Majumdar, “MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis,” *Intelligent Systems with Applications*, vol. 20, 2023, doi: 10.1016/j.iswa.2023.200283.
- [26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, 2002, doi: 10.1613/jair.953.
- [27] A. H. Lashkari, B. Li, T. L. Carrier, and G. Kaur, “VolMemLyzer: Volatile Memory Analyzer for Malware Classification using Feature Engineering,” in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge, RDAAPS 2021*, 2021. doi: 10.1109/RDAAPS48126.2021.9452028.
- [28] E. Frank, M. A. Hall, and I. H. Witten, “The WEKA Workbench Data Mining: Practical Machine Learning Tools and Techniques,” *Morgan Kaufmann, Fourth Edition*, 2016.
- [29] K. Roshan and A. Zafar, “Ensemble adaptive online machine learning in data stream: a case study in cyber intrusion detection system,” *International Journal of Information Technology (Singapore)*, 2024, doi: 10.1007/s41870-024-01727-y.
- [30] M. M. Abualhaj, A. A. Abu-Shareha, Q. Y. Shambour, A. Alsaaidah, S. N. Al-Khatib, and M. Anbar, “Customized K-nearest neighbors’ algorithm for malware detection,” *International Journal of Data and Network Science*, vol. 8, no. 1, 2024, doi: 10.5267/j.ijdns.2023.9.012.

Conflicts of Interest

Authors declare no conflict of interest.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of Data and Material

The CIC-MalMem-2022 dataset is accessible at: <https://www.unb.ca/cic/datasets/malmem-2022.html>

Plagiarism Statement

This article has been scanned by iThenticate™.

Design and Implementation of Remote Lab PID Controller Experiment Based on IoT

Mohammed Elshorafa¹, Emin Güney^{1,2*}, İhsan Pehlivan¹, Cüneyt Bayılmış³

¹Department of Electrical-Electronics Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye

²Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye

³Department of Computer Engineering, Sakarya University, Sakarya, Türkiye

Corresponding author:

Emin Güney, Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye
eminguney@subu.edu.tr

ABSTRACT

The education sector has increasingly embraced distance education to ensure a safe and uninterrupted learning process, especially in wars or epidemics. This paper focuses on designing and implementing a PID (Proportional-Integral-Derivative) controller experiment in remote laboratories utilizing the Internet of Things (IoT). Also, a laboratory experiment was developed using a naturally unstable system, specifically a cart-inverted pendulum. The experiments aim to enhance students' understanding of PID controller tuning within a real-world context. This system was chosen due to its dual motion characteristics, with a linear motion for the cart and a circular motion for the pendulum. Two controllers were designed and implemented for the system to enable control and feedback. Additionally, the Blynk platform was integrated into the experiment setup, allowing for real-time visualization of the system's response, control of PID parameters, and the ability to view the video stream via platforms like Skype. For remote connectivity, NodeMCU, an IoT development board, controlled the pendulum, collected system parameters, and transmitted them to the cloud through the Internet. Moreover, the sensors and the system were mathematically modeled, and their transfer functions were extracted. That allows the students to do the PID experiment theoretically and practically and compare the results. This complete setup enables local and remote access to the experiment, ensuring students can experiment regardless of their physical location. As a result of the study, designing and implementing a remote laboratory PID controller experiment utilizing IoT technology provides students with an innovative and immersive learning experience.

Keywords: Smart laboratory system, Internet of Things (IoT), Proportional-Integral-Derivative (PID), Remote laboratory, Blynk, Inverted Pendulum, NodeMCU

Article History:

Received: 24.04.2024

Accepted: 16.08.2024

Published Online: 27.08.2024

1. Introduction

Online education has surged in importance recently due to various factors such as health concerns [1], security issues like the COVID-19 pandemic [2], and conflicts [3], which have disrupted numerous societal domains, including education. In response, some governments have imposed curfews to prevent the spread of the virus and mitigate infection rates [4]. Also, many educational institutions, particularly universities, have shifted their focus towards online learning, specifically leveraging the Internet of Things (IoT). Utilizing virtual and innovative laboratories aims to address the limitations associated with the practical aspects of education, thereby bridging the gap between theory and practice [5]. This approach ensures that students gain valuable practical experience, enhancing their scientific competence, which is pivotal for their successful transition into the workforce upon graduation [5]. The prominence of online education continues to grow, especially considering the evolving landscape influenced by factors like health and security [6], which disrupt various facets of society, including the education sector [7].

Universities increasingly turn to online education utilizing IoT technology [8], [9] to bolster their offerings, particularly in online and intelligent laboratories. This strategic shift aims to bridge the gap between theoretical and practical education, ensuring students maintain crucial hands-on experience essential for their future careers without compromising their academic rigor. Conventional classroom settings often struggle with large student-to-teacher ratios, particularly prevalent in developing nations, necessitating more classrooms with fewer students. This scenario significantly burdens educators, requiring them to manage large cohorts within limited timeframes, often proving arduous and ineffective [10]. In contrast, virtual and remote laboratories mitigate the logistical challenges of traditional setups and offer cost-effective solutions, particularly in scenarios where dangerous experiments, epidemics, or conflicts disrupt access to physical facilities.

Advancements in information technologies have catalyzed a revolution in educational methodologies, ushering in

unprecedented opportunities for innovation. Pioneering studies, such as those conducted by Fabregas et al., showcase the transformative potential of remote laboratories (RLs) in engineering education. Interactive RLs have been developed through platforms like Simulink and Easy Java Simulations (EJS), offering immersive learning experiences [11]. Responding to the evolving demands of the fourth industrial revolution, initiatives like ELLI, spearheaded by the German Federal Ministry of Education and Research (BMBF), are driving advancements in engineering education. TU Dortmund University, for instance, is at the forefront of this movement, pioneering remote and virtual labs tailored to mechanical engineering disciplines. Innovations such as tele-operative material characterization testing cells and remote labs for incremental tube forming underscore a commitment to practical, industry-aligned education [12]. Technologies like Augmented Reality and Additive Manufacturing further exemplify this dedication to preparing students for real-world challenges and opportunities.

1.1. Contributions

The contributions of this paper are as follows:

- Contribute to advancing distance education, promote practical learning experiences, and enable students worldwide to access and benefit from laboratory experiments through remote access and IoT technology.
- Enhance knowledge of PID controller tuning by providing a practical application that bridges the gap between theoretical concepts taught in the classroom and real-world experiments.
- Allow students to gain a deeper understanding of the subject matter and prepare them for the labor market, fostering innovation and creativity. The system allows for modifying PID controller values.
- Address the need for distance education in the education sector, particularly in areas where traditional - educational processes are hindered by factors such as wars, lack of resources, or epidemics like the COVID-19 pandemic.

The rest of this paper is organized as follows: Sect. 2 presents the literature review on automation and remote control based on IoT. The proposed system is introduced in Sect. 3 with the real-case scenario. Sect. 4 details the study and points to some possible future works. Sect. 5 concludes the research and points to some possible future works.

2. Literature Review

The conceptualization and execution of remote laboratory PID controller experiments are central themes in the literature, alongside pertinent research in this domain. Integrating practical experiments is paramount in enhancing the educational paradigm by bridging the gap between theoretical knowledge and real-world applications, thereby fostering a more profound understanding among students. Among the notable studies in this area, Issa et al. engineered a teleoperated 3D printer featuring a robotic arm capable of executing and regulating the printing process from any location globally, leveraging connectivity with the Repetier-Host platform [13]. However, a notable limitation arises in applying Labview HTML within the video stream of the pendulum, manifesting as a delay of several seconds between the actual pendulum response and its depiction on the Labview HTML interface. This delay adversely impacts the experiment's efficacy, particularly considering the high sensitivity of the controller to temporal factors, potentially leading to student confusion. Colak et al. introduced an innovative web-based DC motor laboratory termed NeTre-LAB aimed at augmenting the teaching of electrical machines. Nevertheless, this laboratory presents certain drawbacks, notably the requirement for a more substantial investment in equipment, including a DAQ board, Hub, and assorted services, as shown in Figure 1.

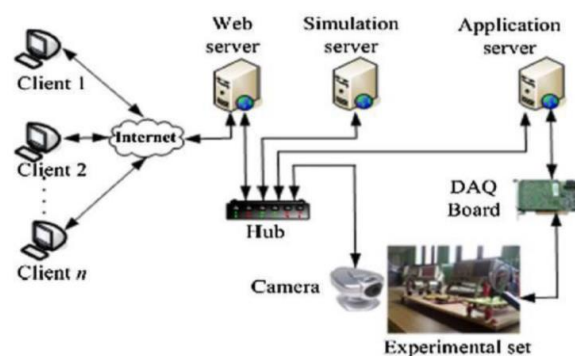


Figure 1. Hardware Structure of the NeTre-LAB [14]

The NeTre-LAB encounters sluggish response times contingent upon network velocity and system demands. In contrast, the proposed system exhibits superiority over NeTre-LAB in both aspects by employing cost-effective equipment for experiment setup and demonstrating rapid response times [14]. Kaçar et al. introduced AnalogWeb, a web-based MATLAB simulation for analog modulation techniques, as an integral component of Analog Communication [15]. Ramya et al. devised a laboratory setup for conducting experiments on actuators and sensors, aiming to enhance their application in engineering education and industrial contexts [16]. In chemical engineering, a remote distillation column experiment was pioneered at the Institute of Process and Plant Technology [17]. Güney et al. explored the utilization of thermal sensors on mobile platforms for detecting living entities, focusing on monitoring body temperatures and the spatial distribution of COVID-19

outbreaks [18]. Benitez et al. innovated a robotic arm system for online robotics instruction during pandemic contingencies, comprising the robotic arm itself, an Internet control unit, and a user interface [13]. Additionally, the Internet of Things (IoT) extends its reach beyond educational and laboratory settings, permeating into diverse domains such as smart homes, urban environments, and industrial operations as shown in Figure 2.

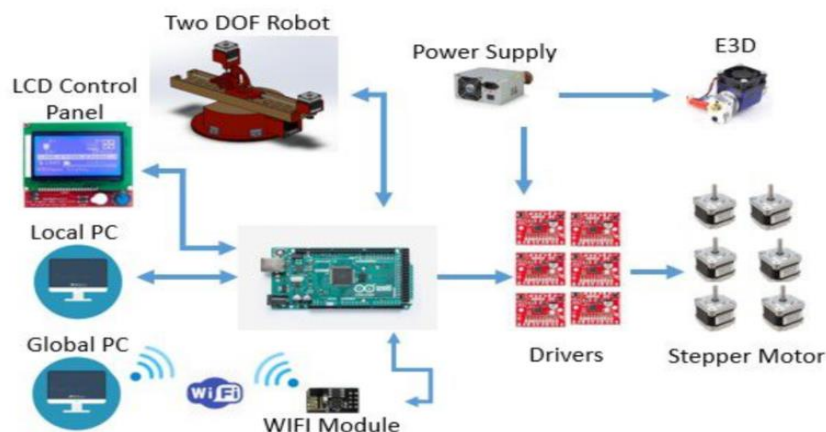


Figure 2. Schematic of the Developed Teleoperated 3D Printer System [19]

Küçük et al. introduce an Internet of Things (IoT) system tailored to motivate students in studying the development and implementation of IoT applications in real-time [20]. Additionally, various studies are available for broader application, facilitating performance comparisons within specific test scenarios [21]. Senese et al. have developed a web interface enabling students to remotely conduct experiments on rapid chemical reactions in real-life settings rather than simulations. This hands-on experience provides interactive technical support and fundamental knowledge exchange, enhancing student engagement. The system facilitates data sharing and analysis, effectively utilizing costly equipment and broadening student access across various universities [22]. Naef conducted an analytical chemical experiment utilizing headspace gas chromatography (GC). Three methods were employed, with a focus on the internet-based approach. While essential tools were utilized in the experiments, only the internet-based method was discussed, as shown in Figure 3 [23].

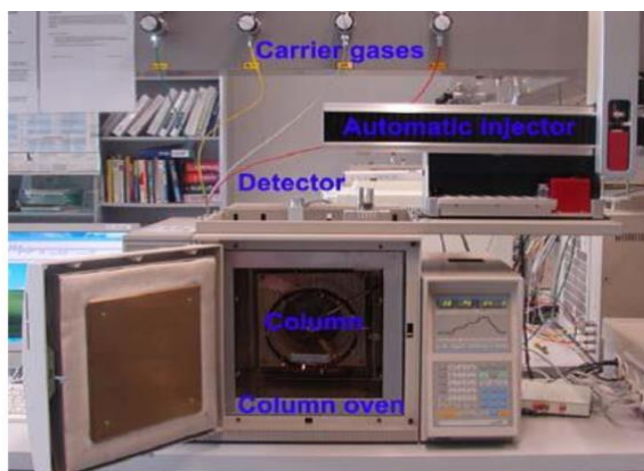


Figure 3. The Chromatography Lab Setup

Technology has emerged as a critical tool for maintaining student engagement in response to the COVID-19 pandemic's disruption of traditional education. The development of an Augmented Remote Lab (ARL) by Nicolette et al. sought to address this challenge by providing a platform for remote learning. Results from the investigation demonstrated notable increases in student motivation, particularly in attention, relevance, and satisfaction, highlighting the potential of remote labs to effectively engage students and facilitate learning experiences in unprecedented circumstances [24]. This study addresses the importance of practical work (PW) in exact sciences and introduces an IoT-based approach for remote experimentation in analog electronics. AM Taj et al. propose a low-cost system utilizing the Red Pitaya STEMLab board for remote manipulation, enabling intelligent selection of integrated practical works. Their analysis emphasizes minimizing latency and enhancing portability to streamline learning experiences while maintaining quality, demonstrated through a comparison with traditional hands-on laboratory methods as shown in Figure 4 [25].

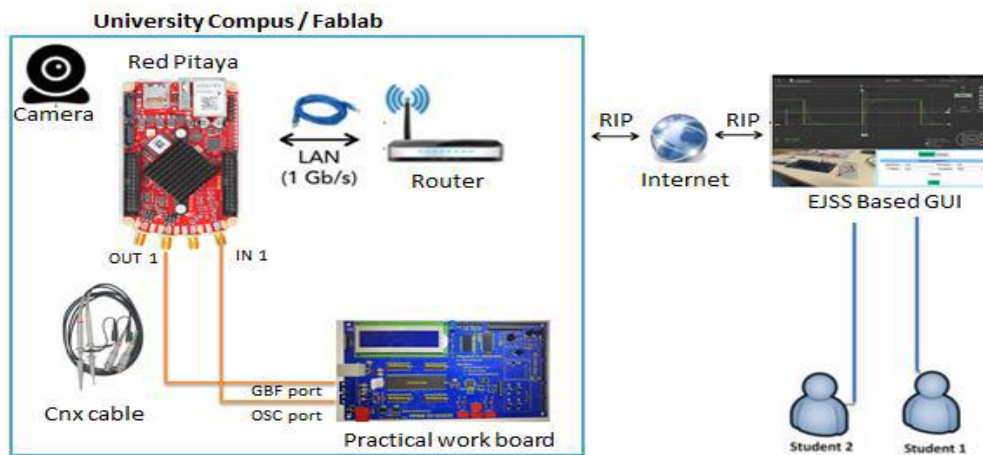


Figure 4. A General Architecture for the Remote IoT System for Practical Work

3. Design and Implementation of the Proposed System

This section presents the design stages and system components in detail. To set up a successful establishment of the educational laboratory and achieve the expected results, it must consist of several elements and components that have compatibility and synchronization, so the system consists of three structures as follows.

3.1. Mechanical System Structure

An inverted pendulum, shown in Figure 5, is a classical dynamic system that is unstable because it falls on its own, given that its stability is upwards, and this can only be achieved by using control systems. In this paper, the PID controller was used to balance the pendulum. The mechanical part of an inverted pendulum system consists of a cart, in the middle of which an inverted pendulum is installed with a rotary bearing that allows it to rotate freely with very little friction. The inverted pendulum is a one DOF system considered a good and low-cost tool for testing control strategies. Moreover, it is easy because it can be viewed as a linear system. The process of balancing the pendulum is done by moving the cart using a motor that is tied to the cart with a belt according to the values of the angle of the pendulum with the cart. The PID controller makes the necessary calculations and sends the appropriate signal to the motor to move the cart and ensure the balance of the pendulum [26], [27].

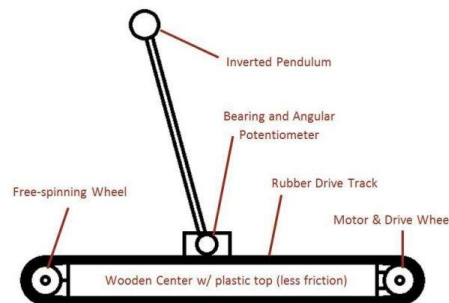


Figure 5. Schematic of Inverted Pendulum Hardware Setup

3.2. Electronic System Hardware

The electronic structure includes a NodeMCU board to perform the PID algorithm and connect the experiment with the internet to send the values of the encoders that have been read and send commands to the DC motor driver to control the motor, as shown in Figure 6. Just before the current was delivered to the motor, the current amplification was done using an H-Bridge DC motor (L298N). Absolute rotary encoders provided the cart with linear movement and angled the pendulum with the cart. A mobile device is used as an IP camera to provide a real-time video stream of the experiment via the Skype application. The developed GUI was designed using the Blynk platform to control the experiment hardware using controls to tune the PID parameters and display the system response in real time.

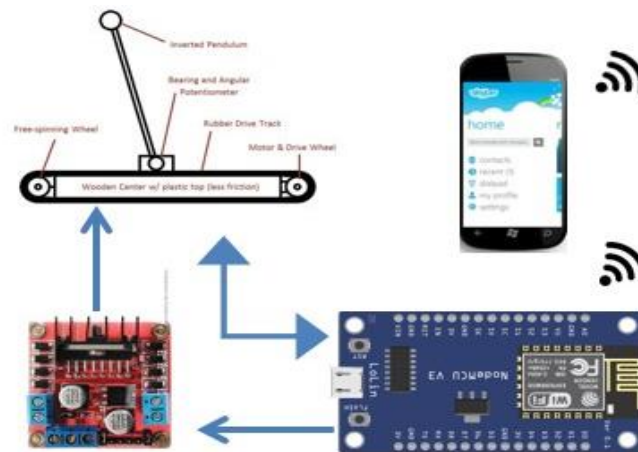


Figure 6. Electronic Control of System Hardware

3.3. System Software Design

The student can observe the real-time video effect of tuning the PID parameters on the Skype platform, as shown in Figure 7. This effect is achieved by setting video calls to auto-response, enabling students to view the experiment immediately upon making a video call on the local mobile device used as an IP camera. The developed Blynk GUI, comprises several sliders, chart displays, and buttons. Through these icons and widgets, students can adjust PID parameters using the sliders, send these values to the cloud using the buttons, and visualize the reaction or response of the pendulum on the chart, as shown in Figure 8. Additionally, the Skype platform was utilized for real-time observation of the experiment. Furthermore, the Blynk app allows students to visualize the pendulum's response as a waveform and modify the PID parameters.



Figure 7. Showing Live Video Stream of Inverted Pendulum Via Skype



Figure 8. GUI Showing the Time-Response of Inverted Pendulum and PID Controller Values on the Blynk Platform

4. System Modeling

System modeling involves converting the physical system into a set of mathematical equations. This step is crucial in system development as it allows for analyzing the system's performance and behavior before implementation. Mathematical modeling can save time and resources, particularly in complex, expensive, and dangerous systems. Various methods, such as the Transfer function and State-Space Representation, can be used for system modeling. Theoretical aspects of the PID controller have been previously discussed as a practical experiment that can be conducted locally or remotely via the Internet. As a result, both the system as a whole and each unit have undergone mathematical modeling and analysis.



Figure 9. General System Block Diagram

Transfer function Representation was used to model the system in this paper because most universities with bachelor's degrees teach it to students, not State-Space Representation. The transfer function (TF) is the system's output and input relationship, as shown in Fig. 9 and Equation 1.

$$TFL(S) = \frac{OUTPUT(S)}{INPUT(S)} \tag{1}$$

4.1. Encoder Sensor Modeling

In the system, two incremental encoders were utilized, one of which is a rotary encoder responsible for determining the pendulum's angle concerning the cart. To derive a transfer function for these sensors, it is essential to understand the properties of the sensor, as shown in Figure 10.

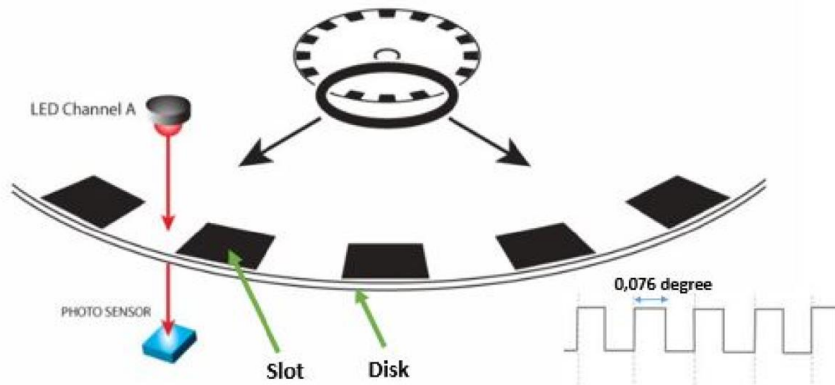


Figure 10. Incremental Rotary Encoder

The disk is divided into 5000 slots, and this means that when the pendulum is rotated at 0.072 degrees, the sensor will give one pulse as follows: 360 degrees/5000 slots = 0.072 deg/slot. The controller accounts for each falling and rising edge of the signal produced by the encoder sensor. The block diagram of the rotary encoder is shown in Fig 11. As mentioned before, TF is out/in. So, the TF of the incremental rotary encoder (TFR) is seen in Equation 2.

$$TFR(S) = \frac{1}{0.072} = 13.88 \tag{2}$$



Figure 11. Incremental Rotary Encoder Block Diagram

The other encoder is a linear encoder, which determines the linear motion or the cart's position on which the pendulum is installed. Moreover, the active length of the cart is 30 cm. It is a strip divided into 6000 slots as shown in Figure 12.

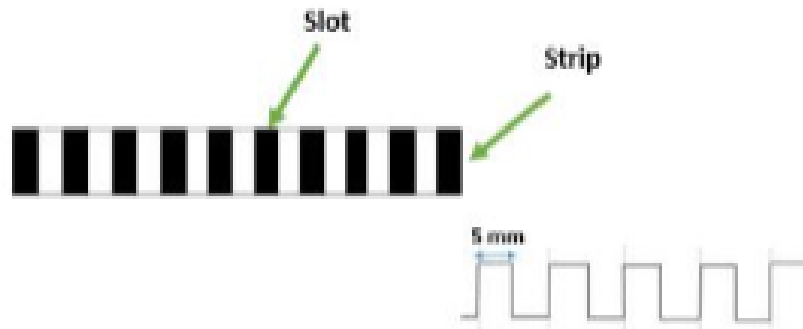


Figure 12. Incremental Linear Encoder

This means that when the cart moves 5 mm, the sensor will give one pulse: $30000 \text{ mm}/6000 \text{ slots} = 5 \text{ mm/slot}$. So, the TF of the incremental linear encoder (TFL) is seen in Equation 3. The controller accounts for each falling and rising edge of the signal produced by the encoder. The block diagram of the linear encoder is shown in Figure 13.

$$TFL(S) = \frac{1}{5\text{mm}} = 200 \tag{3}$$



Figure 13. Incremental Linear Encoder Block Diagram

4.2. Cart Modeling

The actual length of the cart, that is, the range in which the cart can move, is 30 cm, and the middle of the distance is the set point of linear movement as shown in Figure 14. That is, it is the location in which the cart is intended to achieve the optimal state of stability for the system. However, this is not the only condition for achieving the best equilibrium for the pendulum or the system. There is another condition that will be mentioned later. After the analysis of the free body diagram of the cart and converting all obtained equations by Laplace transformation, the transfer function of the cart is as seen in Equation 4.

$$TF_{cart}(S) = \frac{(I + ml^2)s^2 - gml}{s^4 + \frac{b(I + ml^2)}{q}s^3 - \frac{(M + m)mgl}{q}s^2 - \frac{bmgl}{q}s} \tag{4}$$

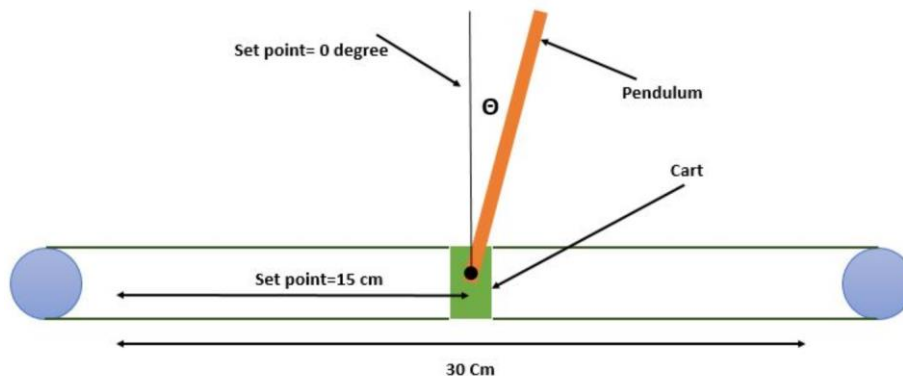


Figure 14. Cart and Pendulum Diagram

The set point of the cart is 15 cm, which means that the linear incremental encoder gives a pulse signal with 3000 falling and rising edges. So the set point equals 3000. So if it is supposed that the home of the cart is on the left side, the encoder equals 0, as shown in Fig. 14. When the encoder reads more than 3000, the cart should be moved from the left to the center. When the encoder reads less than 3000, the cart should be moved to the right to the center. All of them achieve the first condition of system stability. Fig. 15 shows the block diagram of the closed-loop system of the cart. Moreover, the transfer function of the cart closed loop system (TFc) is seen in Equation 5.

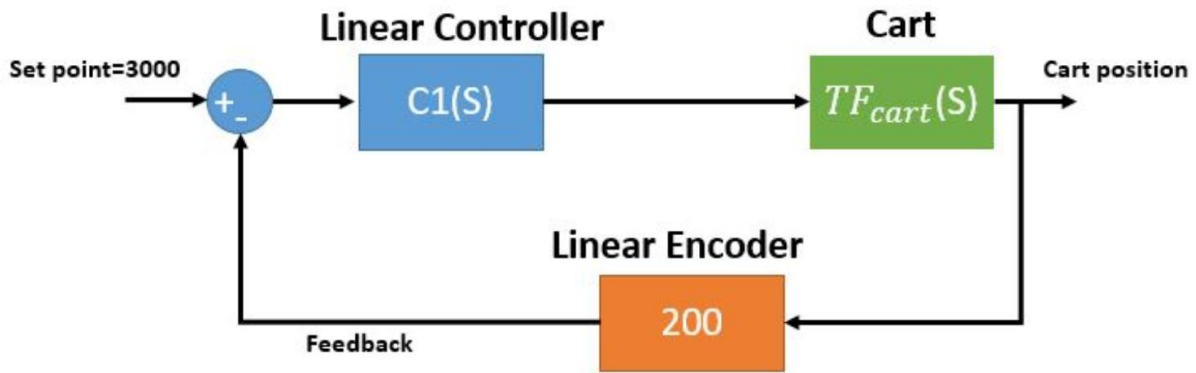


Figure 15. Closed Loop System of the Cart Block Diagram

$$TFc(S) = \frac{C1(S)TFcart(S)}{1 + 200 * C1(S)TFcart(S)} \tag{5}$$

4.3. Pendulum Modeling

After the analysis of the free body diagram of the pendulum and the conversion of all equations obtained by Laplace transformation, the transfer function of the pendulum is seen in Equation 6.

$$TFpen(S) = \frac{\frac{ml}{q}S}{S^3 + \frac{b(M+m)mgl}{q}S^2 - \frac{bmgl}{q}} \tag{6}$$

All parameters of Equations 3 and 4 are illustrated in Table 1. As shown in Fig. 13, the set point of the pendulum is when theta equals 0 degrees, or the pendulum is vertical, which means that the linear incremental encoder gives a pulse signal with zero falling and rising edges. So, the set point equals 0. Before starting the experiment, the pendulum will be downward, and the value of the encoder will be assigned to be -2500. The set point equals 0. If the pendulum wobbles right, the cart moves left to set the pendulum to the set point or to be upward vertically, and so on. After that, the cart puts itself in the center to maintain the two conditions that achieve the perfect stability for the system overall. Fig. 16 shows the block diagram of the close-looped system of the pendulum. Moreover, the pendulum closed loop system (TFp) transfer function is seen in Equation 7.

$$TFp(S) = \frac{C2(S)TFpen(S)}{1 + 13.88 * C2(S)TFpen(S)} \tag{7}$$

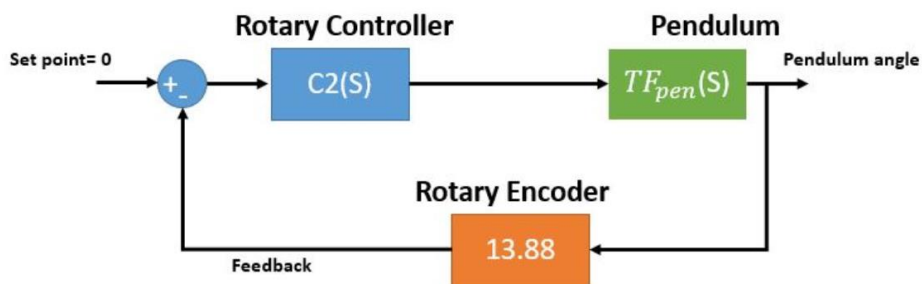


Figure 16. Closed Loop System of the Pendulum Block Diagram

Table 1. Parameters for Inverted Pendulum System

Symbol	Variables and constants	Values
M	mass of the cart	0.35kg
m	mass of the pendulum	0.15kg
b	coefficient of friction for cart	0.1
l	length to pendulum center of mass	0.2 m
L	mass moment of inertia of the pendulum	0.006 kg.m2
F	the force applied to the cart	DC motor force
X	cart position	x
theta	pendulum angle	theta degree

4.4. Overall System Modeling

After analyzing the cart and pendulum and finding the transfer function for both, the closed loop overall system block diagram is shown in Figure 17.

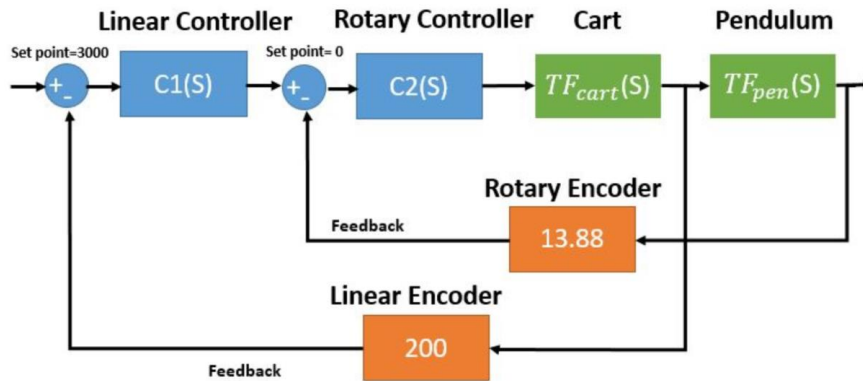


Figure 17. Overall System Block Diagram

After deducting and simplifying techniques on the system block diagram, the system transfer function is seen in Equation 8.

$$TF_{all}(S) = \frac{C1(S)C2(S)TF_{cart}(S)TF_{pen}(S)}{1 + 13.88 * C2(S)TF_{cart}(S)TF_{pen}(S) + 200 * C1(S)C2(S)TF_{cart}(S)} \tag{8}$$

5. System Development

5.1. Mechanical Development

The inverted pendulum system was developed from the cart of an old HP laser jet printer with some modifications, such as adding the pendulum rod in the middle of the cart, which is driven by a DC motor, as shown in Figure 18, The pendulum was fixed on rotary bearings (8mm X 22mm X 7 mm). Moreover, an incremental rotary encoder (5000 sectors) was fixed with the pendulum to measure its angle with the cart. The cart was tied with a pulley (20 teeth GT2) and rubber belt (GT2), and two limit switches were set on the far right and left to give the PID controller limited scope of movement of the cart.

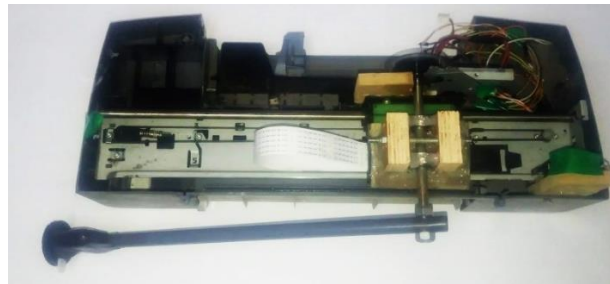


Figure 18. Cart Inverted-Pendulum Hardware

The motion of the cart moves linearly, and the pendulum rotates angularly. The system has two feedbacks, as shown in Figure 19. The Arduino code on the local NodeMcu controlled the experimental hardware.

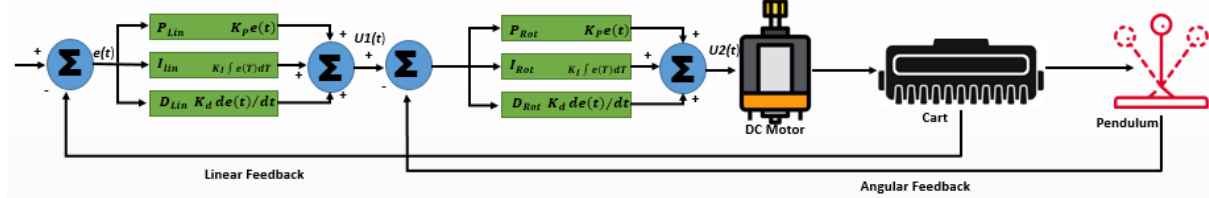


Figure 19. Feedback Control System of the Inverted Pendulum

5.2. Control System Development

The control software uses the Blynk platform, which has a built-in IoT cloud. There are two distinct areas: the lab and the student/user places. The experiment is set up in the lab using an inverted pendulum, NodeMCU, and an IP camera. The student/user place can be any location where students wish to experiment using a mobile device or computer via the Blynk and Skype platforms, as shown in Figure 20.



Figure 20. Schematic of the Developed Lab Experiment System

Generally, the system was built as described in Figure 21. The remote access devices connect with the local control NodeMcu via the Blynk cloud and receive experimental data from the cloud and video stream via Skype using video calls, which are set up to be an automatic response. Furthermore, the remote access device can send control data, such as PID tuning parameters, to the Blynk cloud via the internet and modify the stability of the inverted pendulum system to achieve the experiment’s learning objectives. The NodeMCU controller performs the PID algorithm of the pendulum system by sending control signals to the cart motor and acquiring the position values from the encoders. The balance of this system needs feedback to adjust its state and position because there are two types of movement.

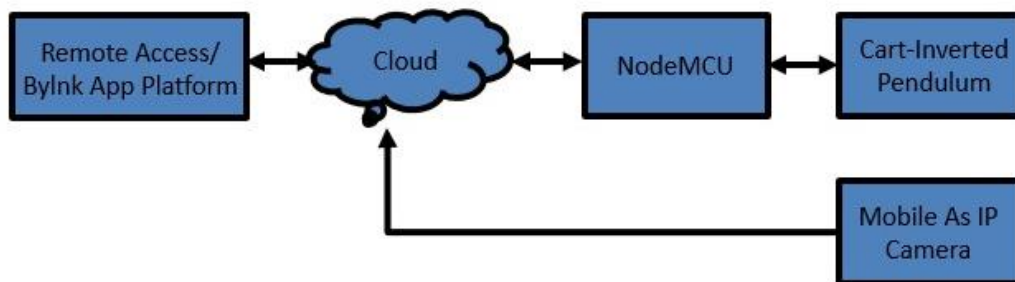


Figure 21. System General Control Layout

6. Conclusion and Future Works

This study demonstrates the increasing use of distance education, particularly during wars or epidemics, to ensure uninterrupted learning. The study focuses on designing and implementing a PID controller experiment using IoT in remote laboratories. A cart-inverted pendulum system was developed to enhance students' understanding of PID controller tuning in real-world contexts. Two controllers were implemented for system control and feedback. Integrating the Blynk platform allowed real-time visualization, PID parameter control, and video streaming. Remote connectivity was achieved through the NodeMcu IoT board, enabling data transmission to the cloud. Mathematical models were created for sensors and the system, facilitating theoretical and practical PID experiments. This setup enables local and remote access to the experiment, providing students an immersive learning experience.

Future work will focus on developing a dedicated web page seamlessly integrated with Firebase as a cloud platform and further linked to the university student portal. This progressive work aims to expand the range of experiments, encompassing endeavors like regulating DC motor speed and empowering students to generate experiment reports effortlessly. These reports will be conveniently transmitted to their instructors through an automated system, streamlining the feedback process.

References

- [1] B. B. Lockee, “Online education in the post-COVID era,” *Nature Electronics* 2021 4:1, vol. 4, no. 1, pp. 5–6, Jan. 2021, doi: 10.1038/s41928-020-00534-0.
- [2] E. Guney, G. Agirtas, and C. Bayilmis, “MongoDB Based Real-Time Monitoring Heart Rate Using Websocket For Remote Healthcare,” *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 8, no. 4, p. 500, Dec. 2022, doi: 10.26555/jiteki.v8i4.25052.
- [3] A. Nikdel Teymori and M. A. Fardin, “COVID-19 and Educational Challenges: A Review of the Benefits of Online Education,” *Annals of Military and Health Sciences Research*, vol. 18, no. 3, Sep. 2020, doi: 10.5812/amh.105778.

- [4] M. Al-Emran, S. I. Malik, and M. N. Al-Kabi, "A Survey of Internet of Things (IoT) in Education: Opportunities and Challenges," *Studies in Computational Intelligence*, vol. 846, pp. 197–209, 2020, doi: 10.1007/978-3-030-24513-9_12/COVER.
- [5] M. B. Abbasy and E. V. Quesada, "Predictable Influence of IoT (Internet of Things) in the Higher Education," *International Journal of Information and Education Technology*, vol. 7, no. 12, pp. 914–920, 2017, doi: 10.18178/ijiet.2017.7.12.995.
- [6] S. Voore, K. Srinivas, and R. S. Pavithr, "A survey on internet of things based smart, digital green and intelligent campus," *In 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 1-6, 2019.
- [7] M. Kassab, J. DeFranco, and P. Laplante, "A systematic literature review on Internet of things in education: Benefits and challenges," *J Comput Assist Learn*, vol. 36, no. 2, pp. 115–127, Apr. 2020, doi: 10.1111/jcal.12383.
- [8] D. Chen, "Application of IoT-Oriented Online Education Platform in English Teaching," *Math Probl Eng*, vol. 2022, 2022, doi: 10.1155/2022/9606706.
- [9] Y. Luo and K. K. Yee, "Research on Online Education Curriculum Resources Sharing Based on 5G and Internet of Things," *J Sens*, vol. 2022, 2022.
- [10] J. Faritha Banu, R. Revathi, M. Suganya, and N. R. Gladiss Merlin, "IoT based Cloud Integrated Smart Classroom for smart and a sustainable Campus," *Procedia Comput Sci*, vol. 172, pp. 77–81, 2020.
- [11] F. Ernesto et al. "Developing a remote laboratory for engineering education." *Computers & Education* 57.2, 1686-1697, 2011.
- [12] J. Grodotzki et al. "Remote and virtual labs for engineering education 4.0: achievements of the ELLI project at the TU Dortmund University." *Procedia manufacturing* 26, 1349-1360, 2018.
- [13] A. Issa, M. Elshorafa, M. O. A. Aqel, N. Naim, and D. Brabazon, "Design and build of a tele-operated and robot-assisted multi-material 3D printer system," *Proceedings - 2019 International Conference on Promising Electronic Technologies, ICPET 2019*, pp. 118–123, Oct. 2019.
- [14] I. Colak, S. Demirbas, S. Sagioglu, and E. Irmak, "A novel web-based laboratory for DC motor experiments," *Computer Applications in Engineering Education*, vol. 19, no. 1, pp. 125–135, Mar. 2011.
- [15] S. Kaçar and C. Bayilmis, "A web-based educational interface for an analog communication course based on MATLAB builder NE with webfigures," *IEEE Transactions on Education*, vol. 56, no. 3, pp. 346–354, 2013.
- [16] M. V. Ramya, G. K. Purushothama, and K. R. Prakash, "Design and implementation of IoT based remote laboratory for sensor experiments," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 9, pp. 227–238, 2020.
- [17] A. Klein and G. Wozny, "Web Based Remote Experiments for Chemical Engineering Education: The Online Distillation Column," *Education for Chemical Engineers*, vol. 1, no. 1, pp. 134–138, Jan. 2006.
- [18] E. Güney, A. Yaşar, G. Ağırtaş, and C. Bayılmış, "Mobil Platformda IoT Temelli ve Soket Programlamaya Dayalı Termal Sensör Uygulaması," *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, Jan. 2023.
- [19] A. Issa, M. El Shorafa, M. O. A. Aqel, D. Brabazon, and P. Young, "Remote Computer Based Learning System for Inverted Pendulum Lab Experiment," in *Proceedings - 2018 International Conference on Promising Electronic Technologies, ICPET 2018, Institute of Electrical and Electronics Engineers Inc.*, pp. 113–117, 2018.
- [20] K. Küçük and C. Bayilmis, "Nesnelerin İnternet'i: Teori ve Uygulamaları (Internet of things: theory and applications)". 2019.
- [21] E. Güney & N. Ceylan, "Response Times Comparison of MongoDB and PostgreSQL Databases in Specific Test Scenarios" *In International Congress of Electrical and Computer Engineering Cham: Springer International Publishing*, 478-188, 2022.
- [22] F. Senese and C. Bender, "The Internet Chemistry Set: Web-based Remote Laboratories for Distance..." *EdMedia + Innovate Learning*, vol. 2000, no. 1, pp. 1731–1733, 2000.
- [23] O. Naef, "Real laboratory, virtual laboratory, or remote laboratory: what is the best choice for the student and the economy?," *International Journal of Online Engineering*, 2006.
- [24] P. C. Nicolette et al. "Analysis of student motivation in using a Physics Augmented Remote Lab during the Covid-19 pandemic." 2021 IEEE global engineering education conference (EDUCON). *IEEE*, 2021.
- [25] A. M. Taj et al. "Conception and implementation of an IoT system for remote practical works in open access university's electronic laboratories." *Learntechlib*, pp. 19-36, 2021.
- [26] Y. I. Eremenko, D. A. Poleshchenko, A. I. Glushchenko, A. M. Litvinenko, A. A. Ryndin, and E. S. Podval'Nyi, "On estimating the efficiency of a neural optimizer for the parameters of a PID controller for heating objects control," *Automation and Remote Control*, vol. 75, no. 6, pp. 1137–1144, 2014.
- [27] P. L. Logunov, M. V. Shamanin, D. V. Kneller, S. P. Setin, and M. M. Shunderyuk, "Advanced Process Control: From a PID Loop up to Refinery-Wide Optimization," *Automation and Remote Control*, vol. 81, no. 10, pp. 1929–1943, 2020.

Authors Contributions

MOHAMMED ELSHORAF: Writing - Review & Editing, Conceptualization, Methodology.

Emin GÜNEY: Writing - Review & Editing, Software, Investigation, Validation.

İhsan PEHLİVAN: Supervision, Writing- Original Draft, Writing - Review & Editing, Data Curation.

Cüneyt BAYILMIŞ: Supervision, Project administration, Software, Investigation, Validation.

Conflict of Interest Notice

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate™.

Evaluating Feature Selection Algorithms for Machine Learning-Based Musical Instrument Identification in Monophonic Recordings

İsmet Emre Yücel¹, Ulaş Yurtsever²

¹Department of Music Technology, State Conservatory, Sakarya University, Sakarya, Türkiye

²Department of Computer Engineering, Sakarya University, Sakarya, Türkiye

Corresponding author:

İsmet Emre Yücel, Department of Music
Technology, State Conservatory,
Sakarya University, Sakarya, Türkiye
eyucel@sakarya.edu.tr



Article History:

Received: 15.07.2024

Accepted: 21.08.2024

Published Online: 27.08.2024

ABSTRACT

Musical instrument identification (MII) research has been studied as a subfield of the Music Information Retrieval (MIR) field. Conventional MII models are developed based on hierarchical models representing musical instrument families. However, for MII models to be used in the field of music production, they should be developed based on the arrangement-based functions of instruments in musical styles rather than these hierarchical models. This study investigates how the performance of machine learning based classification algorithms for Guitar, Bass guitar and Drum classes changes with different feature selection algorithms, considering a popular music production scenario. To determine the effect of feature statistics on model performance, Minimum Redundancy Maximum Relevance (mRMR), Chi-square (Chi2), ReliefF, Analysis of Variance (ANOVA) and Kruskal Wallis feature selection algorithms were used. In the end, the neural network algorithm with wide hyperparameters (WNN) achieved the best classification accuracy (91.4%) when using the first 20 statistics suggested by the mRMR and ReliefF feature selection algorithms.

Keywords: Musical instrument identification, Audio features, Feature selection, Machine learning

1. Introduction

MII studies emerged as a combination of Audio Content Analysis (ACA) and machine learning methodologies. ACA is an essential part of the MIR research field and uses many approaches to attain useful audio signal information. Interpretations of the acquired information allow higher-level descriptions for the audio by machine learning algorithms. Besides its various uses, from speech recognition to medical signal analysis, ACA has a higher potential in music creation and production, using a combination of machine learning techniques. In that manner, the Intelligent Music Production (IMP) research field is a special implementation of ACA, MIR, and Digital Signal Processing (DSP) techniques for developing automatic or assistive solutions for any specific tasks in music production. This study aims to investigate the most effective audio features for musical instrument identification in the context of music production by supervised machine learning algorithms.

MII studies also known as musical instrument recognition studies back in the 90s. Instrument identification from audio is described as the biggest challenge in the MIR field [1], and even today, this challenge stays current due to the potential benefits of MII in streaming services [2]. The identification scenario depends on the source types whether they consist of single (monophonic) or multiple performances (polyphonic) in the same recording. Early works started with the investigation of single-source identification problems and their solutions [3], [4], [5], [6], [7], [8]. On the other hand, MII of polyphonic audio is a more difficult task than single-sourced audio. During the last decade, many studies have focused on identifying instruments from mixed audio sources [9]. Kitahara et al. pointed out three problems (feature variations in sound mixtures, pitch dependency, and musical context) of MII when polyphonic music [10].

Having diverse audio data is a requirement in polyphonic MII tasks. Commercial expectations and technological improvements from multimedia systems projected by the MPEG group and ISO/IEC 15938 became the standard in which content descriptions for audio and other multimedia formats take place [11]. The perceptual features in this standard were also used to create descriptions of instrumental sounds [17]. Peeters [18] listed the audio features categories as temporal shape, temporal feature, energy features, spectral shape features, harmonic features, and perceptual features. Audio features

have been tested in different instrument recognition scenarios [19]. There are some data sets available. Humphrey et al. compared some available datasets for instrument identification [20]. Yücel and Özdemir [21] proposed an audio loop-based dataset regarding the music production perspective. To attain larger datasets, data augmentation may also be another alternative, which Kratimenos et al. [22] referred to for polyphonic MII tasks. Blaszkę and Kostek [23] provided a comprehensive literature summary of ML methods and their results in MII.

Audio features are computational processes to attain temporal and timbral information from raw audio signals. Feature extraction refers to acquiring information by mathematical calculations in a short time window. The time window is typically between 2-10 msec durations, and computational results in that specific time duration are described as low-level features (LLF) or descriptors. Temporal, spectral, and statistical features are the three main audio feature categories [12]. Temporal features are calculated directly from the time-domain, spectral features require Fourier transform (STFT) of the signal and statistical features are derived from LLF in longer time frames. Temporal features also known as time-domain features are zero crossing rate (ZCR), root mean square (RMS), energy, envelope related calculations (attack, decay). Spectral features are also described as frequency domain features, and some of them are centroid, bandwidth, contrast, flatness, roll-off, and MFCC (Mel-frequency cepstrum coefficients) [13]. Detailed literature about audio features has been investigated in many studies [14], [15]. Mid-level and high-level descriptors are statistical features derived from LLF and extensively used to attain musical information, such as tempo, rhythm, genre, and mood [16].

Since extensive audio features are the basis for traditional ML applications in music, the dimensionality of feature vectors gets bigger. The bigger dimension causes an over-fitting problem for model predictions. This is mentioned as the curse of dimensionality in literature [17]. Audio features are fundamental components of ML-based solutions for musical applications. Performance of combined audio features in musical similarity studies [18], most effective feature combinations for musical classification [19], efficiency dependence of different audio features in music emotion recognition tasks [20], feature combinations for emotion (considering arousal and valence) recognition in music recordings [21] and feature selection in music genre classification tasks [22] are some examples for investigation of features effectiveness in different MIR sub fields. In musical instrument classification, Liu and Wan [23] investigated 58 features of data extracted from various audio recordings, achieving %93 of classification accuracy after 19 features. Gulhane et al. stated that MFCC 1, spectral roll-off and spectral centroid can be enough for instrument identification [24]. A higher accuracy rate was obtained with MFCC for K-NN classifiers and timbral descriptors (features) when the classification algorithm is Binary Tree [25]. Although some studies focus on deep learning methods directly from raw audio [26] [27], performance estimation of temporal and spectral audio features is worth investigating.

2. Methodology

2.1. Audio Dataset

Audio loops and single-hit samples are frequently used in musical materials during music creation in various mainstream genres such as Electronic/Dance, Pop, Rock etc. The loops are short musical sound recordings categorized by metadata such as tempo, musical tone, and source type. Musicians often use audio loops when developing and refining their musical ideas from scratch. It is therefore useful to use audio loops to create the audio datasets needed to develop AI-based models to solve various problems in music production scenarios. Modern DAWs (Digital audio workstations) provide music creators with many audio loops and sample libraries. The dataset for this study was created with repetitive and single beat audio files that come with Logic Pro, a popular DAW.

In this study, machine learning models are developed considering the monaural source classification approach. The recordings were organized into three instrument family groups as Drums, Bases, and Guitars considering the basic production elements in popular music, thus presenting class names for the dataset. All audio files were checked one by one to ensure that they were appropriate to the instrument family wherein each was placed. The Drums, Bases, and Guitars classes have 731, 406, 985 audio files respectively. The duration of these files varies from 1 second to several seconds.

2.2. Audio Preprocessing

Even though audio loops and samples generally consist of well-edited files, preprocessing is required to prepare them before the execution of audio feature extraction code. The preprocessing may contain one or more tasks including shorten longer files, DC removal, filtering (low-pass, high-pass, band-pass), pre-emphasis, sampling rate conversion, mono-to-stereo conversion, normalization. In this work, some audio files having longer sustaining parts are redundant, therefore they had been shortened where the points that their signal levels drop drastically. Another preprocessing implementation was stereo-to-mono conversion.

2.3. Audio Features and Extraction

For LLF extraction, in this study the preferred time window time and step duration are 4 msec and 2 msec, respectively. The calculated audio features are ZCR, RMS, spectral centroid, spectral bandwidth, spectral contrast, spectral flatness, spectral roll-off, and MFCC (20 bins) (Table 1). Due to diverse file durations, the mean, standard deviation (std), median, and variance

(var) of the features were calculated for each audio. As a result, a 2122x108 dataset emerged. Since maximum and minimum value ranges differ for different features, data were normalized between 0-1.

Table 1. The Calculated Audio Features and Statistics.

Audio Features	Statistics
Zero crossing rate	Mean Standard deviation (std) Median Variance (var)
Root mean square	
Spectral centroid	
Spectral bandwidth	
Spectral contrast	
Spectral flatness	
Spectral roll-off	
MFCC (20 bins)	

Zero crossing rate (ZCR): This feature calculates how often a signal polarity changes from positive to negative or vice versa. In general, for a signal with simple harmonic content, such as a sine, ZCR increases as the frequency increases. On the other hand, audio signals with higher harmonic content or noise-like audio signals show higher ZCR values.

Root mean square (RMS): This feature allows the overall energy level of the audio signal to be observed by measuring the average intensity of the sound in each time window.

Spectral centroid: Spectral centroid is a measure that represents the center of mass of the frequency distribution of a sound signal. This feature helps to determine which frequencies the spectral components of a sound concentrate around. The spectral centroid provides information about the overall tonal characteristics of the sound. For example, if the spectral centroid of a music piece is close to higher frequencies, it indicates that the piece has brighter content.

Spectral bandwidth: Spectral bandwidth is a crucial feature in audio processing that describes the range of frequencies present in a sound signal. Specifically, it refers to the width of a frequency range within which most of the signal's energy is concentrated. A sound with a narrow bandwidth would have most of its energy concentrated within a small range of frequencies, like a pure tone. On the other hand, a sound with a wide bandwidth would have energy spread across a broader range of frequencies, like white noise.

Spectral Contrast: Spectral contrast computes the energy difference between adjacent frequency bands. The computation is basically a comparison of higher energy regions (peaks) to that of lower energy regions (valleys) within the spectrum. This feature is used in audio signal processing to characterize the difference in energy levels between different frequency bands within an audio signal's spectrum. It measures how pronounced the peaks and valleys are within the frequency spectrum.

Spectral Flatness: This audio feature (known as Wiener entropy) is a metric employed in audio signal analysis to quantify the relative distribution of energy across the frequency spectrum of a signal. It serves as an indicator of the signal's tonal versus noisy characteristics. Spectral flatness is calculated as the ratio between the geometric mean and the arithmetic mean of the power spectrum. A spectral flatness value closer to 1 indicates a relatively even energy distribution across the frequency spectrum, indicative of tonal or harmonic sounds. Conversely, values closer to 0 signify a more peaked distribution, suggesting a dominance of noise or non-harmonic components in the signal.

Spectral roll-off: This feature is defined as the frequency below which a specified percentage of the total spectral energy of a signal is contained. Typically, this percentage is set to 85%, but other values can be used depending on the application. The spectral roll-off point effectively separates the lower energy part of the spectrum from the higher energy part, providing insights into the signal's frequency distribution. It is particularly useful for distinguishing between harmonic and non-harmonic content in audio signals.

MFCC: Mel-Frequency Cepstral Coefficient (MFCC) is extensively utilized in speech recognition and musical classification. The frequency components derived from the STFT calculation are transformed using a non-linear Mel frequency scale and filtered into triangular frequency bands (bins), typically around 20. This process yields a concise representation of the spectral features of an audio signal, which closely resembles the human auditory system compared to raw spectral features.

2.4. ML Algorithms

This research was conducted using the Classification Learner App from Matlab version 2022a. Thus, this study was restricted to the ML algorithms provided by the software. Those algorithms are Decision Tree, Discriminant Analysis, Naïve Bayes, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Ensemble and Neural Network with predefined kernel settings.

2.4.1. Decision Tree

The Decision Tree is a tree-like model where nodes represent tests on features (attributes), edges represent outcomes of these tests, and leaves represent the final decision or prediction. The root node is the top node representing the entire dataset, which is split into subsets based on the most significant feature. Splitting is the process of dividing a node into two or more sub-

nodes to improve the homogeneity of the target variable within the sub-nodes. Decision nodes split into further sub-nodes, while leaf or terminal nodes do not split further and represent the final output. Each branch or sub-tree is a subsection of the entire tree. Decision trees offer several advantages, including interpretability, ease of visualization, and being non-parametric, making no assumptions about the data distribution. They also provide insights into feature importance. However, they have disadvantages, such as being prone to overfitting, especially with noisy data, and instability, where small changes in the data can result in different trees. Despite these drawbacks, decision trees are widely used in medical diagnosis, spam detection, customer segmentation, financial forecasting, pricing models, and risk assessment. They also serve as the foundation for more complex ensemble methods like Random Forests and Gradient Boosting Machines, which enhance predictive performance and mitigate some of the algorithm's limitations [28] [29].

2.4.2. Discriminant Analysis

Discriminant Analysis is a supervised machine learning technique primarily used for classification tasks. Its goal is to model the differences between classes by identifying the best combination of features that separates them. The algorithm prioritizes the probabilities of each class based on the training data. The algorithm then calculates these mean and covariance matrices according to two main types of approaches: Linear Discriminant Analysis (LDA), where these matrices are distributed across all classes, or Quadratic Discriminant Analysis (QDA), where these matrices are calculated separately for each class. Finally, for a given input, the algorithm calculates the value of the discriminant function for each class and identifies the class with the highest discriminant score [30].

2.4.3. Support Vector Machine

SVM is a prevalent supervised ML model. The model's reputation comes from its robustness and versatility, particularly its capability to discriminate high-dimensional spaces. The principle of the algorithm is to find the best boundary that separates different classes. This boundary tends to be a hyperplane where the dimensionality increases, especially in nonlinear conditions. The Support Vectors are the closest data points to hyperplanes influencing their position and orientation. The margin is a term that describes the distance between the hyperplane and the nearest support vectors from either class. SVM aims to maximize this margin to ensure the best separation between classes. This model uses different kernels to create higher dimensional space to separate classes. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid [31].

2.4.4. K-Nearest Neighbor

KNN is another popular supervised ML algorithm for classification and regression tasks. This algorithm discriminates data by measuring the distance between the instances. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance. The "k" symbolizes the closest training examples (neighbors) in the feature space for given data. The smaller k may indicate higher data noise and overfitting, yet the larger one causes underfitting. The algorithm consists of four main steps; first, it holds the instances from training data, then calculates distances between training and testing instances and finds the parameter "k" for each instance [32]. This algorithm has been applied in many tasks in medical applications, such as tuberculosis classification using x-ray images [33], heart disease determination [34], engineering; localization indoors [35], agricultural studies [36], music and speech classification [37], music genre classification [38].

2.4.5. Ensemble

Ensemble algorithms are conceptualized by combining multiple ML models to improve overall performance. There are several types of ensemble methods; the most common are bagging (Random Forest), boosting (AdaBoosting, Gradient Boosting), and stacking (Base Learners, Meta-Learner) [39]. The bagging technique combines statistical methods called bootstrapping, where random sampling (equal-sized subsets) from the training data is taken, and the aggregation method allows the regeneration of the training result of the whole model with the results obtained from the training data. Random forest is a featured bagging type algorithm based on decision trees [40]. The AdaBoost algorithm was proposed by Freund and Schapire to enhance previous boosting algorithms [41]. This algorithm transforms weak learners into strong learners, even without prior accurate information on the weak hypotheses. This method improved the implementation of various multiclass classification and regression problems [42]. The Gradient Boosting method proposed by Friedman function estimation/approximation is a numerical optimization problem in function space. It generalizes boosting by using gradient descent techniques to optimize any fitting criterion, not just for binary classification but also for regression and multiclass classification problems. Specific algorithms for least-squares, least absolute deviation, Huber-M loss functions for regression, and multiclass logistic likelihood for classification are presented, with a special focus on using regression trees as the base learners. This approach enhances robustness, interpretability, and performance, especially for handling less-than-clean data [43].

2.4.6. Neural Network

Neural networks are a category of machine learning architectures that draw inspiration from the configuration and operation of the human brain. Neural networks are composed of interconnected layers of neurons that analyze input data to generate an output. These networks provide a great degree of adaptability and can represent intricate connections within data, rendering

them appropriate for a diverse array of tasks including classification, regression, and pattern recognition. Neurons, the fundamental components of a neural network, receive input, do a weighted sum, and then transmit the outcome through an activation function. Neural networks consist of layers, with the input layer receiving the input data, hidden layers processing the inputs from the previous layer, and the output layer generating the final prediction. Activation functions such as sigmoid, tanh, and ReLU (Rectified Linear Unit) provide non-linear behavior in the network, allowing it to acquire knowledge of intricate patterns. Training refers to improving the weights and biases of a network using methods such as backpropagation and gradient descent [44]. Different hyperparameters may be preferred during the building of neural network-based ML models. The version of Matlab that we used in this study provides narrow, medium, wide, bi-layered, and tri-layered hyperparameters.

A narrow neural network has fewer neurons in its layers, making it simpler and faster to train with lower computational requirements. However, narrow networks may not capture complex patterns as effectively as wider networks and are suitable for simpler tasks or when computational resources are limited [45]. A medium neural network strikes a balance between complexity and computational efficiency, with a moderate number of neurons in its layers. This configuration provides a good trade-off between capturing patterns and avoiding overfitting, making it commonly used for a wide range of tasks, including classification and regression problems [46]. A bi-layered neural network features two hidden layers, allowing it to model more complex patterns than single-layer networks. Although the complexity of training increases, it remains manageable. Bi-layered networks are suitable for tasks where a single hidden layer is insufficient but additional layers do not significantly improve performance. In contrast, a tri-layered neural network has three hidden layers and can learn even more complex patterns than bi-layered networks. While this configuration incurs higher computational costs and a greater risk of overfitting, it is used in more complex tasks where deeper representations are needed, such as deep learning applications in image and speech recognition [47].

The wide neural network (thereafter referred to as WNN) is characterized by having many neurons in each layer. This type of network can capture more complex patterns in the data due to its higher capacity but requires more computational resources and is prone to overfitting if not regularized properly. WNNs are used in scenarios where the dataset is large and complex, such as image recognition and natural language processing [48]. The principle of WNN has the number of neurons in each layer, as opposed to the depth, which refers to the number of layers in the network. In the context of wide neural networks, the emphasis is on increasing the number of neurons per layer rather than stacking many layers allowing for more complex representations of the input data. WNN uses the Tangent type kernel method for the model creation. Advantages of the model are dealing with more complex data structures providing better representation of the input data, consistency in classification stable training phase, connections to kernel methods and effective interpolation of training data [49].

2.5. Feature Selection Algorithms

Generally, most datasets consist of high dimensional features that do not have the same importance to create distinction about data. Therefore, feature selection is an approach to determining the most related features to represent data. Guyon and Elisseeff [27] stated the potential benefits of feature selection. In this study, five feature selection algorithms, mRMR (Maximum Relevance and Minimum Redundancy), Chi2, ReliefF, ANOVA, and Kruskal Wallis, were examined.

2.5.1. mRMR

The mRMR is considered a robust filtering method and was initially developed for the classification of DNA microarray data. Some implementations of the method are anomaly detection, eye movement analysis, gender classification, and analysis of satellite images [28]. This method aims to determine a subset of features that are highly important for a specific job while minimizing redundancy among them. Relevance measures the importance of each feature with respect to the target variable, often using mutual information. Redundancy measures the similarity among the features, again often using mutual information. Features are selected iteratively by adding the one that maximizes the difference between relevance and redundancy. Various research has emphasized the benefits of the mRMR algorithm. Tang et al. highlighted that mRMR is recognized for its rapid computation speed and resilience, as it automatically identifies essential characteristics by considering maximum correlation and lowest redundancy criteria [50]. In addition, Alshamlan et al. examined the efficacy of integrating mRMR with genetic algorithms and particle swarm optimization algorithms for gene selection in cancer classification [51]. Their study showcased the adaptability of mRMR in improving classification tasks.

2.5.2. Chi-Square

The Chi2 (Chi-Square) is a two phased χ^2 statistical method where data discretization occurs [52]. This method examines if there is a significant connection between category variables and the target variable. It evaluates whether the observed frequency distribution of a feature differs from the expected distribution. To perform feature selection using the Chi-Square statistic, start by calculating the Chi-Square statistic for each feature, which measures the discrepancy between the observed and expected frequencies. This involves creating a contingency table for each feature with respect to the target variable and computing the Chi-Square value for each table. Next, rank the features based on their Chi-Square statistics in descending order, highlighting those with the most significant discrepancies. Finally, select the top-ranked features based on a predefined threshold or the desired number of features. This selection process ensures that the most informative features are retained for further analysis or model building. Chi2 feature selection algorithm has been widely used in various domains such as

computer security, sentiment analysis, malware detection, and health management. Studies have shown that the Chi2 algorithm effectively evaluates the dependency level of feature sets on class labels [53]. It has been used with other techniques like TFIDF for sentiment analysis [54], and Multinomial Naïve Bayes for question classification [55]. Additionally, the algorithm has been applied in feature selection for audio-related tasks, such as emotion regulation music recommendation Gong et al. [56] and malware detection systems [57].

2.5.3. ReliefF

ReliefF was originally developed by Kira and Rendell [58]. This algorithm can handle multi-class problems and is robust to noisy and incomplete data. It estimates the quality of features by considering the difference between feature values of nearest neighbors from the same and different classes. This feature selection algorithm basically searches two nearest hit and miss neighbors in an instance. The algorithm starts the selection process by setting all the feature weights to zeros. Afterward, it searches for nearest neighbors from the same class (hits) and different classes (misses) of instances that random samples from the dataset. The algorithm updates the feature weights in each process based on the difference between the sampled instance and its neighbors. Consequently, the features are selected according to their weights. The ReliefF algorithm, including its variations like I-ReliefF and SURF, has shown efficiency in estimating feature quality, especially in scenarios with strong dependencies among features [59], [60], [61]. It has been successfully applied in diverse fields such as biology, medicine, and computer science, showcasing its versatility and effectiveness [62], [63], [64]. ReliefF has also been combined with other techniques like Random Forest and Support Vector Machines to enhance classification models and improve feature subset selection [62].

2.5.4. ANOVA

The ANOVA (Analysis of Variance) is useful to check a hypothesis state whether it is null or not. The process begins by calculating the F-statistic for each feature, which measures the variance ratio between groups to the variance within groups. This helps identify features that contribute significantly to the variance in the target variable. Next, rank the features based on their F-statistic, prioritizing those with higher values. Finally, select the top-ranked features based on a predefined threshold or the desired number of features, ensuring that the most influential features are included in the model. ANOVA is utilized to reduce computations and time complexity while enhancing accuracy by overcoming the curse of dimensionality [65]. This method is widely used in many fields such as agriculture, biology [66], [67], medical research [68], [69], and engineering [70], [71].

2.5.5. Kruskal Wallis

The Kruskal Wallis is a nonparametric test that determines if all k populations are identical or if at least one of the populations tends to give observations different from those of other populations. The Kruskal Wallis test is a non-parametric alternative to ANOVA, designed to compare the medians of multiple groups without assuming data normality, making it suitable for ordinal or non-normal continuous data. The process begins by ranking all data points across groups. Next, calculate the H-statistic, which measures the discrepancy between the ranks of the groups. After calculating the H-statistic for each feature, rank the features based on their H-statistic values. Finally, select the top-ranked features based on a predefined threshold or the desired number of features. The Kruskal Wallis test offers several advantages, including not assuming data normality, suitability for ordinal or non-normal continuous data, and robustness to outliers. However, it is less powerful than ANOVA when the assumptions of normality and homogeneity of variances are met and are sensitive to ties in the data [72].

3. Results and Discussion

Model training, using a 5-fold cross-validation method, was preferred, and then the dataset was trained by 28 supervised ML algorithms with their default settings. Consequently, the WNN algorithm provided the best classification result at 93.2% before feature selection had been applied. The detailed ML model results are given in Table 2.

Table 2. The Tested Supervised ML Algorithms

Model	Hyperparameter	Accuracy
Tree	Fine	86.1%
	Medium	83.3%
	Coarse	76.0%
Discriminant	Linear	85.2%
Naïve Bayes	Gaussian	74.5%
	Kernel	65.6%
SVM	Linear	87.3%
	Quadratic	91.9%
	Cubic	92.6%
	Fine Gaussian	89.2%
	Medium Gaussian	87.4%
	Coarse Gaussian	80.9%

KNN	Fine	91.3%
	Medium	87.3%
	Coarse	79.9%
	Cosine	88.0%
	Cubic	87.1%
	Weighted	90.3%
Ensemble	Boosted	89.1%
	Bagged Trees	91.5%
	Subspace Discriminant	84.1%
	Subspace KNN	92.6%
	RUSBoosted Trees	87.8%
Neural Network	Narrow	91.7%
	Medium	92.2%
	Wide	93.2%
	Bi-layered	90.9%
	Tri-layered	90.7%

The determined top 20 audio features by mRMR, Chi2, ReliefF, ANOVA, and Kruskal Wallis algorithms are given in Table 3. The roll-off median has a higher grade in the list. It is also the primary feature according to mRMR and Chi2 algorithms in this experiment. For simplicity, the word ‘‘Spectral’’ is omitted in Tables 3 and 4, and the band index (bin) of the MFCC attribute is indicated by the number next to it.

Table 3. The Top 20 Audio Features Ranged by the Feature Selection Algorithms

	mRMR	Chi2	ReliefF	ANOVA	Kruskal Wallis
1	Roll-off median	Roll-off median	Bandwidth std	Roll-off mean	Flatness median
2	Contrast var	ZCR median	Roll-off median	Roll-off median	Flatness mean
3	MFCC 11	ZCR mean	Roll-off mean	Centroid mean	Roll-off mean
4	Bandwidth var	Centroid median	Centroid mean	Centroid median	Roll-off median
5	Flatness median	Flatness median	Centroid median	ZCR mean	Centroid median
6	MFCC 14 mean	Roll-off mean	Bandwidth var	ZCR median	Centroid mean
7	Contrast mean	Centroid mean	Bandwidth median	Flatness mean	MFCC 7 mean
8	ZCR median	Flatness mean	ZCR mean	Bandwidth mean	MFCC 6 mean
9	Flatness mean	MFCC 6 mean	Roll-off std	ZCR std	MFCC 8 mean
10	MFCC 7 mean	MFCC 7 mean	ZCR median	Flatness std	Flatness var
11	ZCR var	MFCC 8 mean	Bandwidth mean	Bandwidth median	Flatness std
12	Roll-off mean	ZCR std	Flatness std	MFCC 4 std	ZCR mean
13	MFCC 4 var	ZCR var	Contrast mean	Flatness var	MFCC 9 mean
14	Flatness var	MFCC 6 std	Contrast median	MFCC 7 mean	ZCR median
15	MFCC 9 mean	MFCC 6 var	Roll-off var	MFCC 5 std	MFCC 5 mean
16	Centroid mean	MFCC 2 std	ZCR std	MFCC 8 mean	MFCC 10 mean
17	Bandwidth median	MFCC 2 var	Flatness mean	MFCC 6 mean	ZCR var
18	MFCC 18 mean	MFCC 5 std	RMS std	MFCC 3 std	ZCR std
19	MFCC 6 mean	MFCC 5 var	Flatness var	MFCC 6 std	MFCC 7 median
20	ZCR mean	MFCC 4 std	RMS mean	MFCC 9 mean	MFCC 8 median

With the dataset and the WNN classification algorithm, it was observed that the common audio features listed by the feature selection algorithms are roll-off mean, roll-off median, ZCR mean, ZCR median, centroid mean, and flatness mean (in Table 4). MFCC 7 mean, flatness var, MFCC 6 mean, centroid median, and ZCR std are mutual features listed by four selection algorithms. Flatness median, ZCR var, MFCC 9 mean, bandwidth median, and MFCC 8 mean are common in three selection algorithm lists. bandwidth var, contrast mean, MFCC 6 std, MFCC 5 std, MFCC 4 std, and flatness std take place only in two selection lists. Eventually, 23 features are listed by only a single selection algorithm.

Table 4. The Top 20 Common Features and Uncommon Features Listed by the Selection Algorithm(s)

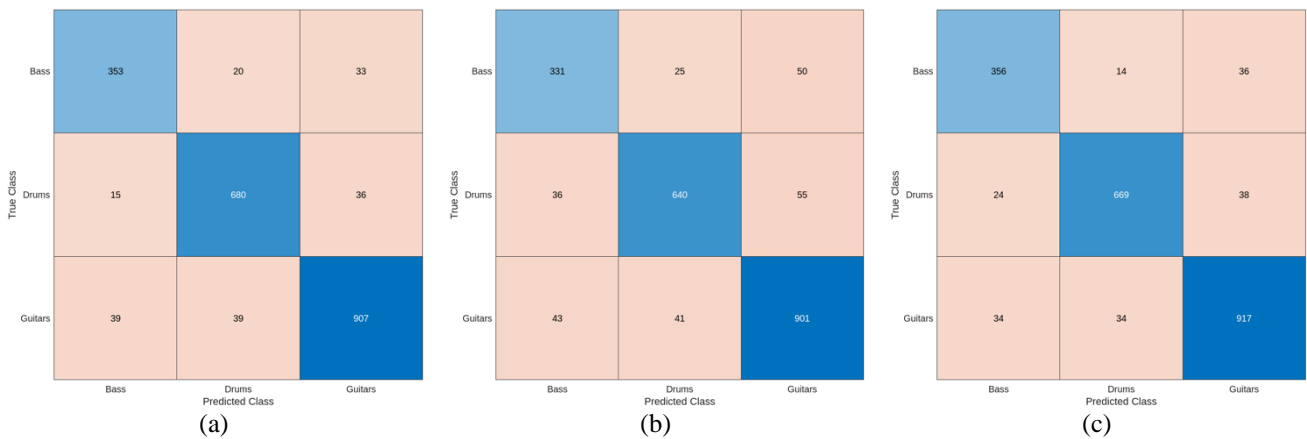
Common Selected Features by the	Audio Features Statistics	Number of features
All algorithms	Roll-off mean, roll-off median, ZCR mean, ZCR median, centroid mean, flatness mean	6
4 algorithms	MFCC 7 mean, flatness var, MFCC 6 mean, centroid median, ZCR std	5
3 algorithms	Flatness median, ZCR var, MFCC 9 mean, bandwidth median, MFCC 8 mean	5
2 algorithms	Bandwidth var, contrast mean, MFCC 6 std, MFCC 5 std, MFCC 4 std, flatness std	6
Listed only by 1 selection algorithm	Contrast var, MFCC 11, MFCC 14 mean, MFCC 4 var, MFCC 18 mean, MFCC 6 var, MFCC 2 std, MFCC 2 var, MFCC 5 var, bandwidth std, roll-off std, bandwidth mean, flatness std, contrast median, roll-off var, RMS std, RMS mean, MFCC 3 std, MFCC 10 mean, ZCR var, MFCC 5 mean, MFCC 7 median, MFCC 8 median	23

Further experiments have been performed to check ML model performances with the selected features. The best performance rate of WNN was 93.2% with 108 audio features, and this score degrades according to different feature lists by the selection algorithms. In Table 5, performance changes of the WNN model are given for each algorithm’s top 20 list. According to the table, mRMR and ReliefF share the same performance rates at 91.4%, even though their feature list contents are different. The performance results are 89.3% for ANOVA, 88.4%, for Chi2 and 88.3% for Kruskal Wallis (Table 5). Also, the confusion matrix of the WNN model is given for each algorithm's top 20 lists in Figure 1. Figure 2 shows the ROC curve of the WNN model for each algorithm's top 20 lists. Additionally, it was observed that the performance result dropped to 84.5% rate, after the ML model was trained by common features (roll-off mean, roll-off median, ZCR mean, ZCR median, centroid mean, flatness mean).

Table 5. Classification Success Rates of WNN Algorithm with Selected the First 20 Features

mRMR’s top 20	Chi2’s top 20	ReliefF’s top 20	ANOVA’s top 20	Kruskal Wallis’s top 20
91.4%	88.4%	91.4%	89.3%	88.3%

WNN algorithm models were created using the first 20 features selected with different feature selection methods. The confusion matrices created to evaluate the performance of these models using different feature selection methods are shown in Figure 1. When the confusion matrices are analyzed, it is observed that high accuracy rates are obtained in general, but the error rates on classes vary depending on the methods used.



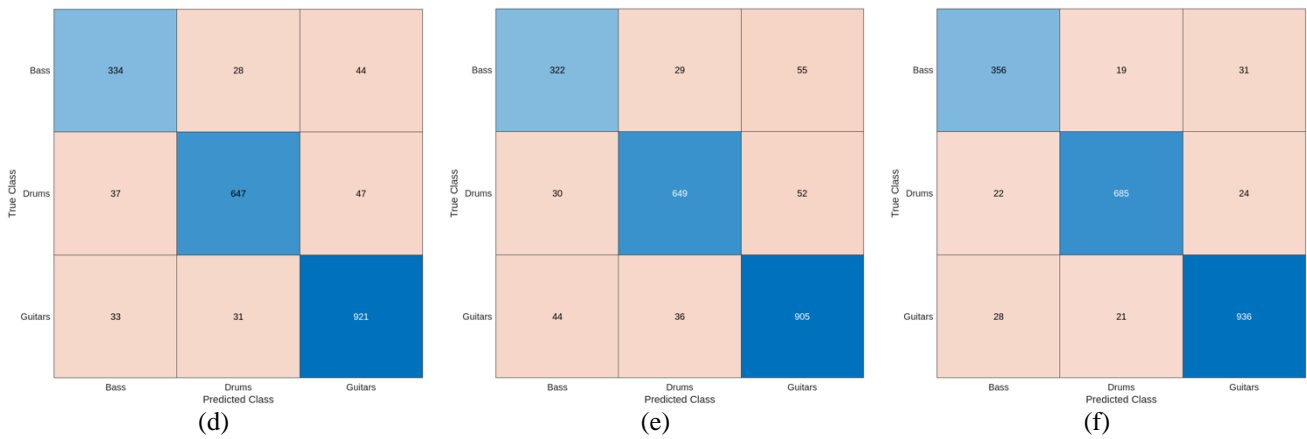


Figure 1. Confusion Matrixes of WNN Algorithm with Selected the First 20 Features a) mRMR b) Chi2 c) ReliefF d) ANOVA e) Kruskal Wallis f) No Selection

Figure 2 shows the ROC curves of the WNN algorithm models created with different feature selection methods. Each ROC curve shows the classification performance of the model in the ‘Bass’, ‘Drums’ and ‘Guitars’ classes separately. In particular, high AUC values were obtained in the ‘Guitars’ class, indicating that this class can be better classified than the others. In general, high AUC values were obtained with all feature selection methods, indicating that the classification performance of the models is high. When different feature selection methods are analyzed, similar to the confusion matrix, it is observed that the AUC values of the models created with mRMR and ReliefF methods give similar results when compared with the model performance using all features.

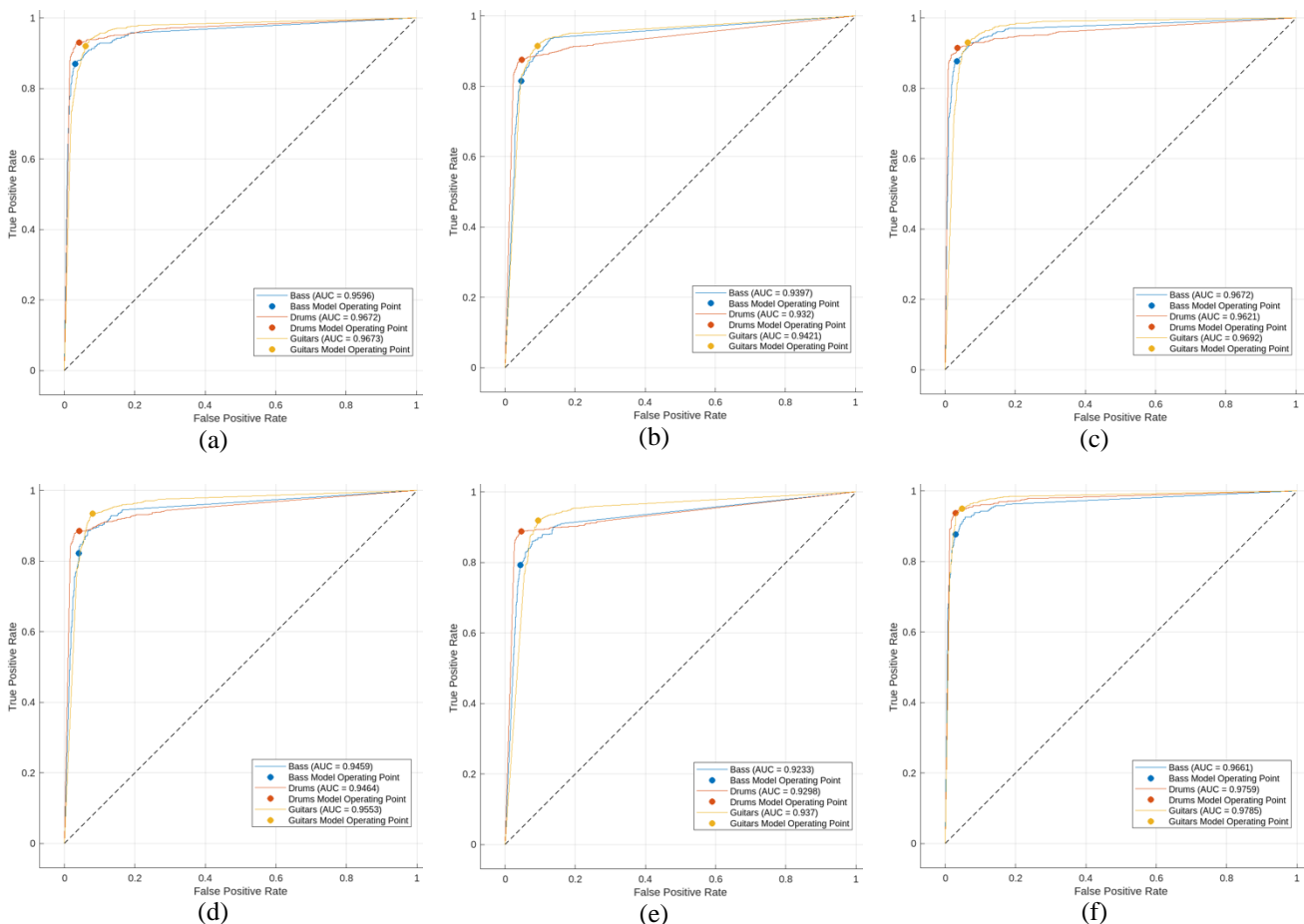


Figure 2. ROC Curves of WNN Algorithm with Selected the First 20 Features a) mRMR b) Chi2 c) ReliefF d) ANOVA e) Kruskal Wallis f) No Selection

It is worth mentioning that MFCC should be interpreted as a whole feature group even if the algorithms ranked its bins differently. Considering the results there is no direct relationship between selected features even though some of them are available in more than one selection algorithm list. In this experiment, when the model performance was analyzed after the feature selection process, there was no significant feature statistic that could be used to achieve good classification results.

At that point, further investigations are required to understand how different statistics affect the classification results. Timbre is the fundamental parameter of how human beings discriminate sound. Thus, the effort of estimating a sound source strongly depends on the timbral representation of the source, yet it is hard to model only a couple of audio features. Therefore, strict feature selection may not be suitable for audio classification especially in MII tasks. A specific MII approach exemplified in this study (single-hit and monophonic resources) may not be suitable for MII models designed with longer and poly-sourced audio files because those types of models will diversify the requirements of audio preprocessing, feature sets, ML algorithm and feature selection. Another point is that the description of a sound source from the perspective of audio production and music creation varies broadly. The functionality of an instrument in a musical arrangement cannot be explained only by its organologic root but also by the intention of the musician while creating the music.

4. Conclusion

In this study, various machine learning algorithms and feature selection methods were evaluated for their effectiveness in musical instrument identification (MII) using monophonic and single-hit sound sources. The experiment was constrained to three instrument families—Drums, Bases, and Guitars—within the context of popular music production scenarios. The dataset consisted of 27 audio features (with MFCCs having 20 bins) and their four statistical variations. Following the training of the dataset with seven machine learning algorithms and their various hyperparameters, the Wide Neural Network (WNN) demonstrated the highest classification accuracy at 93.2%. Upon further examination with five feature selection algorithms (mRMR, Chi2, ReliefF, ANOVA, and Kruskal-Wallis), it was found that common features selected by these algorithms, such as spectral roll-off mean and median, zero-crossing rate mean and median, spectral flatness mean, and spectral centroid mean, played a crucial role in classification performance. Both mRMR and ReliefF achieved a performance rate of 91.4% with the WNN, highlighting the importance of these common features.

However, this study also highlights several areas for future research and acknowledges its limitations. The current study focused on a controlled experimental setup with specific instruments and recording conditions, which may not fully represent the diversity of real-world music production scenarios. Future research should explore how these algorithms perform in different contexts, such as polyphonic recordings or real-time music production environments, to extend the applicability of the findings. Additionally, the limitations of traditional feature selection methods suggest that there is room for exploring more sophisticated techniques or integrating feature selection directly into deep learning models. Expanding the dataset to include a broader range of instruments, styles, and recording conditions, potentially through data augmentation, could enhance the generalizability of the models developed. Moreover, investigating the usability and performance of these models in actual music production software would provide valuable insights into their real-world applications. Finally, future studies could benefit from a cross-disciplinary approach, involving collaboration between experts in music production, machine learning, and digital signal processing, to develop more sophisticated models that can better understand and categorize musical instruments.

By addressing these areas, future research can build upon the findings of this study, leading to more robust and generalizable models for musical instrument identification, ultimately contributing to advancements in both music technology and artificial intelligence.

References

- [1] A. Ghosh, A. Pal, D. Sil, and S. Palit, "Music Instrument Identification Based on a 2-D Representation," in *3rd International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques, ICEECCOT 2018*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 509–513. doi: 10.1109/ICEECCOT43722.2018.9001486.
- [2] U. Shukla, U. Tiwari, V. Chawla, and S. Tiwari, "Instrument classification using image based transfer learning," in *Proceedings of the 2020 International Conference on Computing, Communication and Security, ICCCS 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/ICCCS49678.2020.9277366.
- [3] I. Kaminskyj and A. Materka, "AUTOMATIC SOURCE IDENTIFICATION OF MONOPHONIC MUSICAL INSTRUMENT SOUNDS," *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems*, 1995.
- [4] I. Kaminskyj and P. Voumard, "Enhanced automatic source identification of monophonic musical instrument sounds," *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems*, no. November, pp. 76–79, 1996.
- [5] K. D. Martin and Y. E. Kim, "2pMU9. Musical instrument identification: A pattern-recognition approach *," in *Presented at the 136th meeting of the Acoustical Society of America*, Newyork, 1998.
- [6] P. Herrera-Boyer, G. Peeters, and S. Dubnov, "Automatic classification of musical instrument sounds," in *International Journal of Phytoremediation*, Journal of New Music Research, 2003, pp. 3–21.
- [7] S. K. Banchhor and A. Khan, "Musical Instrument Recognition using Spectrogram and Autocorrelation," *Soft comput*, no. 1, pp. 1–4, 2012.
- [8] H. Mukherjee, S. M. Obaidullah, S. Phadikar, and K. Roy, "SMIL - A Musical Instrument Identification System," Springer, Singapore, 2017, pp. 129–140. doi: 10.1007/978-981-10-6427-2_11.

- [9] Y. Han, J. Kim, and K. Lee, "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music," *IEEE/ACM Trans Audio Speech Lang Process*, vol. 25, no. 1, pp. 208–221, Jan. 2017, doi: 10.1109/TASLP.2016.2632307.
- [10] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Instrument identification in polyphonic music: Feature weighting with mixed sounds, pitch-dependent timbre modeling, and use of musical context," *ISMIR 2005 - 6th International Conference on Music Information Retrieval*, no. January, pp. 558–563, 2005.
- [11] S. Chang, S. Member, T. Sikora, S. Member, and A. Puri, "Overview of the MPEG-7 Standard," vol. 11, no. 6, pp. 688–695, 2001.
- [12] M. R. Bai, A. Member, and C. Chen, "Intelligent Preprocessing and Classification of Audio Signals*."
- [13] P. Wei, F. He, L. Li, and J. Li, "Research on sound classification based on SVM," *Neural Comput Appl*, vol. 32, no. 6, pp. 1593–1607, Mar. 2020, doi: 10.1007/s00521-019-04182-0.
- [14] F. Alías, J. C. Socoró, and X. Sevillano, "A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds," *Applied Sciences*, vol. 6, no. 5. Balkan Society of Geometers, 2016. doi: 10.3390/app6050143.
- [15] J. D. Deng, C. Simmermacher, and S. Cranefield, "A study on feature analysis for musical instrument classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 2, pp. 429–438, 2008, doi: 10.1109/TSMCB.2007.913394.
- [16] A. Aljanaki and M. Soleymani, "A data-driven approach to mid-level perceptual musical feature modeling," Jun. 2018, [Online]. Available: <http://arxiv.org/abs/1806.04903>
- [17] J. L. Fernández-Martínez and Z. Fernández-Muñiz, "The curse of dimensionality in inverse problems," *J Comput Appl Math*, vol. 369, 2020, doi: 10.1016/j.cam.2019.112571.
- [18] J. Osmalskyj, M. Van Droogenbroeck, and J. J. Embrechts, "Performances of low-level audio classifiers for large-scale music similarity," in *International Conference on Systems, Signals, and Image Processing*, 2014, pp. 91–94.
- [19] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "On feature combination for music classification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, pp. 453–462. doi: 10.1007/978-3-642-14980-1_44.
- [20] M. Chmulik, R. Jarina, M. Kuba, and E. Lieskovska, "Continuous music emotion recognition using selected audio features," in *2019 42nd International Conference on Telecommunications and Signal Processing, TSP 2019*, 2019. doi: 10.1109/TSP.2019.8768806.
- [21] J. Grekow, "Audio features dedicated to the detection of arousal and valence in music recordings," in *Proceedings - 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications, INISTA 2017*, 2017, pp. 40–44. doi: 10.1109/INISTA.2017.8001129.
- [22] J. Mitra and D. Saha, "An Efficient Feature Selection in Classification of Audio Files," pp. 29–38, 2014, doi: 10.5121/csit.2014.4303.
- [23] M. Liu and C. Wan, "Feature selection for automatic classification of musical instrument sounds," *Proceedings of the ACM International Conference on Digital Libraries*, pp. 247–248, 2001, doi: 10.1145/379437.379663.
- [24] S. R. Gulhane, S. S. Badhe, and S. D. Shirbahadurkar, "Cepstral (MFCC) Feature and Spectral (Timbral) Features Analysis for Musical Instrument Sounds," *Proceedings - 2018 IEEE Global Conference on Wireless Computing and Networking, GCWCN 2018*, pp. 109–113, 2018, doi: 10.1109/GWCN.2018.8668628.
- [25] P. S. Jadhav, "Classification of Musical Instruments sounds by Using MFCC and Timbral Audio Audio Descriptors," 2015.
- [26] J. Lee, T. Kim, J. Park, and J. Nam, "Raw Waveform-based Audio Classification Using Sample-level CNN Architectures," no. Nips, 2017.
- [27] K. Avramidis, A. Kratimenos, C. Garoufis, A. Zlatintsi, and P. Maragos, "Deep convolutional and recurrent networks for polyphonic instrument classification from monophonic raw audio waveforms," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 3010–3014, 2021, doi: 10.1109/ICASSP39728.2021.9413479.
- [28] T. M. Hehn, J. F. P. Kooij, and F. A. Hamprecht, "End-to-End Learning of Decision Trees and Forests," *Int J Comput Vis*, vol. 128, no. 4, 2020, doi: 10.1007/s11263-019-01237-6.
- [29] Z. ÇETİNKAYA and F. HORASAN, "Decision Trees in Large Data Sets," *Uluslararası Muhendislik Arastirma ve Gelistirme Dergisi*, vol. 13, no. 1, 2021, doi: 10.29137/umagd.763490.
- [30] A. Araveeporn, "Comparison of Logistic Regression and Discriminant Analysis for Classification of Multicollinearity Data," *WSEAS Trans Math*, vol. 22, 2023, doi: 10.37394/23206.2023.22.15.
- [31] A. Saini, "Guide on Support Vector Machine (SVM) Algorithm," *Analytics Vidhya*, 2024.
- [32] S. Uddin, I. Haque, H. Lu, M. A. Moni, and E. Gide, "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/S41598-022-10358-X.
- [33] R. A. Rizal, N. O. Purba, L. A. Siregar, K. P. Sinaga, and N. Azizah, "Analysis of Tuberculosis (TB) on X-ray Image Using SURF Feature Extraction and the K-Nearest Neighbor (KNN) Classification Method," *Jaict*, vol. 5, no. 2, p. 9, Oct. 2020, doi: 10.32497/JAICT.V5I2.1979.
- [34] B. Akalin, Ü. Veranyurt, and O. Veranyurt, "Classification of Individuals at Risk of Heart Disease Using Machine

- Learning,” *Cumhuriyet Medical Journal*, 2020, doi: 10.7197/cmj.vi.742161.
- [35] X. Peng, R. Chen, K. Yu, F. Ye, and W. Xue, “An improved weighted k-nearest neighbor algorithm for indoor localization,” *Electronics (Switzerland)*, vol. 9, no. 12, 2020, doi: 10.3390/electronics9122117.
- [36] H. K. Karthikeya, K. Sudarshan, and D. S. Shetty, “Prediction of Agricultural Crops using KNN Algorithm,” *Int J Innov Sci Res Technol*, vol. 5, no. 5, 2020.
- [37] R. Thiruvengatanadhan, “Speech/Music Classification using MFCC and KNN,” 2017.
- [38] X. Mu, “Implementation of Music Genre Classifier Using KNN Algorithm,” *Highlights in Science, Engineering and Technology*, vol. 34, 2023, doi: 10.54097/hset.v34i.5439.
- [39] I. D. Mienye and Y. Sun, “A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3207287.
- [40] A. Verikas, A. Gelzinis, and M. Bacauskiene, “Mining data with random forests: A survey and results of new tests,” *Pattern Recognit*, vol. 44, no. 2, 2011, doi: 10.1016/j.patcog.2010.08.011.
- [41] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J Comput Syst Sci*, vol. 55, no. 1, 1997, doi: 10.1006/jcss.1997.1504.
- [42] R. E. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Mach Learn*, vol. 37, no. 3, 1999, doi: 10.1023/A:1007614523901.
- [43] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann Stat*, vol. 29, no. 5, 2001, doi: 10.1214/aos/1013203451.
- [44] S. Joshi, A. Gera, and S. Bhadra, “Neural Networks and Their Applications,” in *Evolving Networking Technologies: Developments and Future Directions*, 2023, doi: 10.1002/9781119836667.ch13.
- [45] G. Alfonso and D. R. Ramirez, “Neural networks in narrow stock markets,” *Symmetry (Basel)*, vol. 12, no. 8, 2020, doi: 10.3390/SYM12081272.
- [46] M. Saglam, C. Spataru, and O. A. Karaman, “Forecasting Electricity Demand in Turkey Using Optimization and Machine Learning Algorithms,” *Energies (Basel)*, vol. 16, no. 11, 2023, doi: 10.3390/en16114499.
- [47] J. Chen, K. Li, K. Bilal, X. Zhou, K. Li, and P. S. Yu, “A Bi-layered parallel training architecture for large-scale convolutional neural networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, 2019, doi: 10.1109/TPDS.2018.2877359.
- [48] J. Xi, O. K. Ersoy, J. Fang, T. Wu, X. Wei, and C. Zhao, “Parallel Multistage Wide Neural Network,” *IEEE Trans Neural Netw Learn Syst*, vol. 34, no. 8, 2023, doi: 10.1109/TNNLS.2021.3120331.
- [49] A. Radhakrishnan, M. Belkin, and C. Uhler, “Wide and deep neural networks achieve consistency for classification,” *Proc Natl Acad Sci U S A*, vol. 120, no. 14, 2023, doi: 10.1073/pnas.2208779120.
- [50] X. Tang, Q. He, X. Gu, C. Li, H. Zhang, and J. Lu, “A Novel Bearing Fault Diagnosis Method Based on GL-mRMR-SVM,” *Processes*, vol. 8, no. 7, Jul. 2020, doi: 10.3390/PR8070784.
- [51] H. Alshamlan, G. Badr, and Y. Alohal, “mRMR-ABC: A Hybrid Gene Selection Algorithm for Cancer Classification Using Microarray Gene Expression Profiling,” *Biomed Res Int*, vol. 2015, 2015, doi: 10.1155/2015/604910.
- [52] H. Liu and R. Setiono, “Chi2: feature selection and discretization of numeric attributes,” *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 388–391, 1995, doi: 10.1109/tai.1995.479783.
- [53] T. D. Diwan *et al.*, “Feature Entropy Estimation (FEE) for Malicious IoT Traffic and Detection Using Machine Learning,” *Mobile Information Systems*, vol. 2021, 2021, doi: 10.1155/2021/8091363.
- [54] U. I. Larasati, M. A. Muslim, R. Arifudin, and A. Alamsyah, “Improve the Accuracy of Support Vector Machine Using Chi Square Statistic and Term Frequency Inverse Document Frequency on Movie Review Sentiment Analysis,” *Scientific Journal of Informatics*, vol. 6, no. 1, pp. 138–149, May 2019, doi: 10.15294/SJI.V6I1.14244.
- [55] N. Yusliani, S. A. Q. Aruda, M. D. Marieska, D. M. Saputra, and A. Abdiansah, “The effect of Chi-Square Feature Selection on Question Classification using Multinomial Naïve Bayes,” *Sinkron*, vol. 7, no. 4, pp. 2430–2436, Oct. 2022, doi: 10.33395/SINKRON.V7I4.11788.
- [56] X. Gong, R. Yuan, H. Qian, Y. Chen, and A. G. Cohn, “Emotion Regulation Music Recommendation Based on Feature Selection,” *Frontiers in Artificial Intelligence and Applications*, vol. 337, pp. 486–495, Sep. 2021, doi: 10.3233/FAIA210047.
- [57] A. Tripathi, N. Bhoj, M. Khari, and B. Pandey, “Feature Selection and Scaling for Random Forest Powered Malware Detection System”, doi: 10.21203/RS.3.RS-778333/V1.
- [58] K. Kira and L. A. Rendell, “The Feature Selection Problem: Traditional Methods and a New Algorithm,” in *AAAI’92: Proceedings of the tenth national conference on Artificial intelligence*, 1992, pp. 129–134.
- [59] C. Zhang, M. Ye, L. Lei, and Y. Qian, “Feature Selection for Cross-Scene Hyperspectral Image Classification Using Cross-Domain I-ReliefF,” *IEEE J Sel Top Appl Earth Obs Remote Sens*, vol. 14, pp. 5932–5949, 2021, doi: 10.1109/JSTARS.2021.3086151.
- [60] Y. Zhou, R. Zhang, S. Wang, and F. Wang, “Feature Selection Method Based on High-Resolution Remote Sensing Images and the Effect of Sensitive Features on Classification Accuracy,” *Sensors*, vol. 18, no. 7, Jul. 2018, doi: 10.3390/S18072013.
- [61] L. Sun, X. Kong, J. Xu, Z. Xue, R. Zhai, and S. Zhang, “A Hybrid Gene Selection Method Based on ReliefF and Ant Colony Optimization Algorithm for Tumor Classification,” *Sci Rep*, vol. 9, no. 1, Dec. 2019, doi:

- 10.1038/S41598-019-45223-X.
- [62] H. Ding and L. Huang, "Extraction of soybean planting areas based on multi-temporal Sentinel-1/2 data," *Third International Conference on Computer Vision and Pattern Analysis (ICCPA 2023)*, p. 8, Aug. 2023, doi: 10.1117/12.2684169.
- [63] R. Togo *et al.*, "Cardiac sarcoidosis classification with deep convolutional neural network-based features using polar maps," *Comput Biol Med*, vol. 104, pp. 81–86, Jan. 2019, doi: 10.1016/J.COMPBIOMED.2018.11.008.
- [64] C. S. Greene, N. M. Penrod, J. Kiralis, and J. H. Moore, "Spatially Uniform ReliefF (SURF) for computationally-efficient filtering of gene-gene interactions," *BioData Min*, vol. 2, no. 1, 2009, doi: 10.1186/1756-0381-2-5.
- [65] H. Nasiri and S. A. Alavi, "A Novel Framework Based on Deep Learning and ANOVA Feature Selection Method for Diagnosis of COVID-19 Cases from Chest X-Ray Images," *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/4694567.
- [66] M. O. Arowol, S. O. Abdulsalam, R. M. Isiaka, and K. A. Gbolagade, "A Hybrid Dimensionality Reduction Model for Classification of Microarray Dataset," *International Journal of Information Technology and Computer Science*, vol. 9, no. 11, pp. 57–63, Nov. 2017, doi: 10.5815/IJITCS.2017.11.06.
- [67] G. F. Dong, L. Zheng, S. H. Huang, J. Gao, and Y. C. Zuo, "Amino Acid Reduction Can Help to Improve the Identification of Antimicrobial Peptides and Their Functional Activities," *Front Genet*, vol. 12, Apr. 2021, doi: 10.3389/FGENE.2021.669328.
- [68] B. Thakur, N. Kumar, and G. Gupta, "Machine learning techniques with ANOVA for the prediction of breast cancer," *International Journal of Advanced Technology and Engineering Exploration*, vol. 9, no. 87, pp. 232–245, Feb. 2022, doi: 10.19101/IJATEE.2021.874555.
- [69] F. A. Putra, S. Mandala, and M. Pramudyo, "A Study of Feature Selection Method to Detect Coronary Heart Disease (CHD) on Photoplethysmography (PPG) Signals," *Building of Informatics, Technology and Science (BITS)*, vol. 4, no. 2, Sep. 2022, doi: 10.47065/BITS.V4I2.2259.
- [70] S. Suresh and V. P. S. Naidu, "Mahalanobis-ANOVA criterion for optimum feature subset selection in multi-class planetary gear fault diagnosis," *Journal of Vibration and Control*, vol. 28, no. 21–22, pp. 3257–3268, Nov. 2022, doi: 10.1177/10775463211029153.
- [71] M. J. Siraj, T. Ahmad, and R. M. Ijtihadie, "Analyzing ANOVA F-test and Sequential Feature Selection for Intrusion Detection Systems," *International Journal of Advances in Soft Computing and Its Applications*, vol. 14, no. 2, pp. 185–194, 2022, doi: 10.15849/IJASCA.220720.13.
- [72] P. E. McKight and J. Najab, "Kruskal-Wallis Test," *The Corsini Encyclopedia of Psychology*, pp. 1–1, Jan. 2010, doi: 10.1002/9780470479216.CORPSY0491.

Conflicts of Interest

Authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of Data and Material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate™.

Predictive Model for Incipient Faults in Oil-Filled Transformers

Efosa Igodan¹, Michael Osajeh¹, Linda Usiosefe¹

¹ Unviersity of Benin, Benin City, Nigeria

Corresponding author:

Efosa Igodan, University of Benin,
Benin City, Nigeria
charles.igodan@uniben.edu



Article History:
Received: 03.01.2024
Accepted: 01.07.2024
Published Online: 29.08.2024

ABSTRACT

The power transformer is an invaluable piece of device in the power system. To prevent catastrophic failures and the ensuing power outages, the status of a transformer linked to a system must be examined for any possible faults. Despite using DGA as a global tool for detecting faults, it is limited by the inability to accurately solve the problem associated with results variability due to the intrinsic nature of the IEC TC 10 database. This study proposed a data-driven fault/defect diagnostic model using four ensemble models with three base classifiers respectively. The base classifiers are comprised of SVM, C4.5 decision tree, and naive Bayes while the ensemble methods are comprised of stacking, voting, boosting and bagging respectively. The DGA dataset used comprises seven features and 168 instances split into training (i.e. 56%) and test (i.e. 44%) datasets respectively. The results indicate that C4.5 obtained a 98.33% accuracy while stacking obtained a 99.89% accuracy as the best-performing base and ensemble models respectively. The high classification performance accuracy achieved by our proposed models indicates its capacity for real-world applications. It can be applied to advance automation in mobile-based technology.

Keywords: Dissolve gas analysis, power transformer, Naive Bayes, support vector machine, C4.5, ensemble learning

1. Introduction

A power transformer is one of the costliest, intricate, and significant pieces of equipment in electrical power systems that is often susceptible to early failures when overworked. Its malfunctioning has the potential to cause considerable damage that can lead to economic and social loss. Therefore, early detection of this incipient fault and its precise identification are of immense importance towards averting any damage that may arise. The transformer's organic insulating materials and its oil, break down and release various gases due to the electrical and thermal stress when in operation. Some of these gases released are hydrogen H₂, acetylene C₂H₂, methane CH₄, ethylene C₂H₄, and ethane C₂H₆. While carbon dioxide CO₂ and carbon monoxide CO are formed due to the decomposition of the insulating paper [1-3] as part of the fault gases, nitrogen N₂ and Oxygen O₂ are the non-fault gases [4]. The primary fault categories that are reliably identifiable are partial discharges (PD), low-energy discharges (D1), high-energy discharges (D2), brownish paper-coloured thermal faults (T1), carbonized paper-coloured thermal faults (T2), and thermal faults above 700 °C (T3) indicated by metal colouration, fusion, or oil carbonization. In recent times, fault-type prediction methods are either based on dissolved gas analysis (DGA) carried out by some conventional methods or machine learning-based (ML) methods and Artificial Intelligence (AI) or a combination of both methods [26]. The conventional methods are divided into ratio methods and graphical methods. The ratio methods include the Doernenburg ratio, Rogers' three and four-ratio [9-11,56], the gas production rate method [7,8], and the IEC 60599 code methods [26] among others. The graphical methods include the Duval triangle [5,9,12], Duval and Mansour pentagon [4,26], and Gouda heptagon [26] methods. These conventional methods are simple to implement. However, they have poor detection accuracy rates for power transformer fault types. Other limitations include their inability to deal with all data value ranges [59]. Also, because some ratio ranges lie outside the methods' parameters [14,15], it becomes difficult to classify the transformer's state correctly, causing variability in fault identification accuracy [17]. Furthermore, insufficient coding and strict coding limits constrain the three-ratio and enhanced three-ratio approaches [5]. Nevertheless, compared to traditional ratio and graphical methods, artificial intelligence-based methods have higher prediction accuracy for transformer problem diagnosis, which is required to guarantee the greatest degree of electrical grid reliability. Consequently, it was suggested that machine learning-based techniques, which are based on DGA, are better for power transformer diagnostics [26]. However, despite the progress in using these learning methods, they still have several limitations. Some of these limitations are model overfitting and underfitting, results variability issues, and the problem with bias-variance trade-off. While some are trapped

in a local optimum problem [26], others may have difficulty in parameters tuning [59] resulting in the inefficiency of the use of single models to achieving optimality. The utilization of AI and ML-based methods include: naive Bayes, decision tree-based models [4], artificial neural network (ANN) [18], expert system [19], fuzzy theory [6,14,20], hybrid grey wolf optimization technique [3], grey system [5], support vector machine (SVM) [22], K-Nearest neighbour [23], Bayesian Neural Network [24], and other intelligent systems that infuses diversity in the models as reported in [59]. Therefore, to effectively address some of these challenges, an ensemble method is essential. It does this by utilizing the diversity of the predictions, lowering the risk of overfitting and underfitting, balancing the trade-off between bias and variance, and employing various subsets and features of the data to improve performance and robustness. However, there appears to be insufficient literature that has applied ensemble models of predict faults in power transformers to the best of the researcher's knowledge. Although, more research studies have been carried out in this area, however, there is still no one-fit-all model that can solve all problems [25]. While many diagnostic models underperform because they rely too heavily on the expert's knowledge, in other situations it can be difficult to find a suitable relationship between the input and output variables to support learning [27, 58]. This study proposes developing a power transformer defect prediction system based on six machine learning classification algorithms, motivated by the abundance of problems. Multilayer perceptron (MLP), Support Vector Machines (SVM), Naïve Bayes (NB), C4.5 decision trees, Logistic Regression (LR), and ensemble methods are the six machine learning algorithms. 168 dataset samples gathered from the literature were used to test the approaches. The collected samples have a high overlap degree among different fault types. To enhance the ML predicting accuracy, data preprocessing techniques were implemented. The prediction accuracy was compared among different existing classification methods from the literature. The various methods are implemented using Python Programming Language software on the Google Collab environment.

The organization of the next sections is as follows. Section two carries out the literature review, while section three introduces materials and methods. Section four shows the performance evaluation of the proposed method, while section five discusses the findings of the results and comparisons with existing results from the literature. Finally, section six presents conclusions and recommendations.

2. Related Literature Review

Recent developments in field-deplorable computer technology have reignited curiosity in using computers for routine tasks, particularly labour-intensive ones. Various researchers have conducted numerous studies to investigate cutting-edge methods for automated faultfinding on the electrical grid. The authors in [4] observed that DGA remains one of the best methods for transformer fault identification, however, it has the problem of results variability. The authors suggested using Naïve Bayes and decision trees to develop a fault identification system using 481 instances with nine distinct input vectors. While the decision tree obtained 83.75%, the naïve Bayes achieved 86.25% respectively. However, their study was unable to perform optimally. In [30], Fuzzy logic (FL) was used to identify and safeguard power system transformer problems. Significant maintenance and repair cost savings were achieved as a result of the investigation, however, the issue of variability of results was observed in the work and the inability of the model to learn. In [16], a data-driven method utilizing SVM, back-propagation neural networks (BPNN), and extreme learning machine-radial basis function (ELM-RBF) was proposed for a fault diagnosis system based on DGA data. In comparison to ELM-RBF and BPNN, SVM demonstrated the best performance in the proposed multistage fault diagnosis system. However, it was characterized by increased computation complexity for all stages. In [31], the authors presented a genetic algorithm-based model to choose the best-dissolved gas ratios for SVM-based power transformer problem diagnostics. The model was based only on the International Electro-Technical Commission (IEC) TC 10 database. The study only used three conventional techniques: IEC criteria, DGA data and IEC three key gas ratios with SVM, and back propagation neural network (BPNN) respectively with an 87.18% accuracy achieved. The works of [32], a probabilistic neural network (PNN) and bio-inspired optimizer, were applied to artificial intelligence to build a fault diagnostic model. The PNN's hidden layer smoothing factor was optimized by the bio-inspired optimizer known as the improved salp swarm algorithm (ISSA), which gradually enhances the PNN's classification performance. The PNN served as the fundamental classifier in the fault diagnostic model and achieved 99.65% accuracy higher compared to the traditional fault diagnosis techniques indicating that the technique has a powerful learning ability for data with high complexity. The authors in [34] proposed a neural network model based on traditional methods IEC and Roger's ratio for power transformer diagnosis. Most of the constraints were eliminated in their work, and the diagnostic outcomes increased for the IEC and Roger procedures, from 20% to 70% and 40% to 70% respectively. However, they observed that in certain samples, all approaches i.e. whether the conventional or the model-based on artificial neural networks were deceptive, providing incorrect diagnoses that put the power transformer's integrity at risk. The authors in [35] presented a fault diagnostic model for power transformers using machine learning algorithms and traditional methods. The results showed that the decision tree algorithm outperformed KNN and SVM with a 93.13% accuracy. The authors observed that the classification algorithm and the input data greatly affect the diagnostic accuracy. To determine the risk of transformer fault types, a unique DGA technique based on the Parzen window (PW) estimation was created in [36] using the quantities of five combustible hydrocarbon gases: hydrogen, ethylene, acetylene, methane, and ethane.

According to the results when compared, the PW method outperforms several AIs and ML techniques, including ANN, SVM, ELM, SaE-ELM, and NNCA, and performs noticeably better than traditional ratio-based diagnostic procedures. According to the experimental data, 94.82% of these challenging circumstances are correctly classified by the suggested model, with Duval's triangle providing an unclear classification. The drawbacks of the existing transformer fault diagnosis techniques in dissolved gas-in-oil analysis were addressed in [32] by proposing a transformer fault diagnostic model based on the three

DGA Ratios and PSO-SVM. With an 85.71% accuracy attained, the results showed that the suggested PSO-SVM strategy outperformed the SVM and GA-SVM approaches. However, their results were reported as suboptimal. In [59], a performance Assessment of the IEEE/IEC Method and Duval Triangle (DT) technique for Transformer Incipient Fault Diagnosis was proposed. While the DT methodology was found to perform better than the IEEE/IEC method, the scope of their investigation was confined to the consideration of only two DGA methods and two types of electrical faults: high energy discharge and low energy discharge. Furthermore, the authors in [5] suggested that intelligent algorithms should be combined for mutual complementation to form a hybrid fault diagnosis network to prevent local optimum problems. Their work offers insightful ideas and recommendations for studying intricate power systems, along with references and directions to help researchers select the best course of action for achieving DGA-based fault detection and selecting huge oil-immersed power transformers for electrical testing intended to be preventive. As a result of this strategy, some academics recommended tackling this problem with ensemble approaches. The authors in [33] observed that all the technologies applied in power system fault diagnosis are closely related to modern information technologies such as information system theory, machine learning integration, and information entropy. The authors posited that promoting practical research on modern information technologies is conducive to the high-quality prosperity of power system fault diagnosis, which would play an imperative role in promoting the whole intelligent process of modern society. Motivated by these limitations, three well-known supervised machine learning techniques - C4.5, SVM and naive Bayes, and four ML ensemble methods - Bagging, Boosting, Stacking and Voting methods were investigated to address some of the challenges associated with power transformer fault types towards building results confidence. This includes combining several intelligent algorithms to create a hybrid classifier that mutually complements each other to handle complex issues or combining them to form an ensemble (homogeneously or heterogeneously). Furthermore, the authors in [17] were motivated by the low accuracy of fault identification characterized by the usage of traditional transformer fault diagnostic methods. They suggested combining the XGBoost with an enhanced genetic algorithm (IA) to create a hybrid diagnostic network to locate power transformer problems. The transformer failure recognition problem was broken down and reconstructed into multiple smaller issues that the model could address by combining IGA and XGBoost. An accuracy of 99.2% higher than IEC ratios, dual triangles, SVM, and CVA was obtained. However, the model lacks generalization ability. The authors in [27] observed that the DGA interpretation highly depends on the technical personnel's competence, but it's not conclusive in determining the presence of incipient defects. The authors proposed a brand-new, decision tree-based multinomial classification model called KosaNet. According to the study, KosaNet outperformed the decision tree, k-NN, random forest, naive Bayes, and gradient boost, especially when it comes to classifying multinomial data, achieving an accuracy of 99.98%. The authors also intend to investigate the application of KosaNet for time series data in the context of deploying real-time IoT-enabled smart sensors for transformer observation. In contrast, six optimized machine learning classification algorithms (DT, DA, NB, SVM, KNN, and ensemble approaches) with four data transformation strategies were used in [38] to build a novel power transformer defect type diagnostic. An accuracy of 97.14% was achieved using the ensemble learning method.

3. Materials and Methods

This part contains the basic building block of the study as well as the overall procedure of our proposed model. The entire architecture of the suggested research work technique is shown in Figure 1.

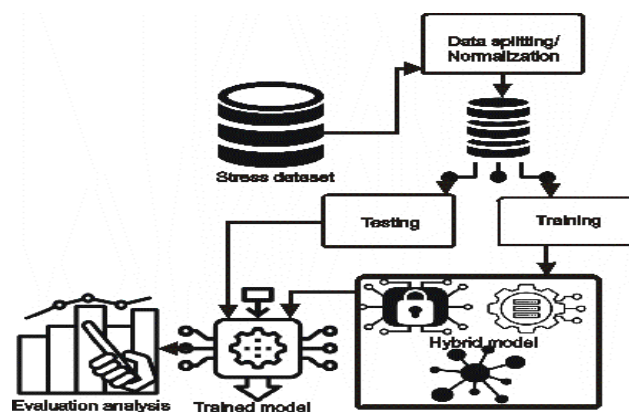


Figure 1. Methodology of the Proposed System Architecture

3.1. Transformer Fault Diagnosis System

Analysis of the dissolved gas in insulation oil sheds light on the thermal and electrical stressors that the oil-immersed power transformer experiences, which cause the oil and paper insulator to break down in the transformer. These stresses release gases as they break down the insulating materials. Whilst the paper insulator produces CO and CO₂, the oil decomposition releases H₂, CH₄, C₂H₂, C₂H₄ and C₂H₆ [16]. The fault type can be determined by type and amount [1]. The incipient fault types used in this proposed framework are based on the new IEC publication 60599 in the IEC TC 10 database, simplified into five categories as depicted in Figure 2 as our proposed methodology for this research.

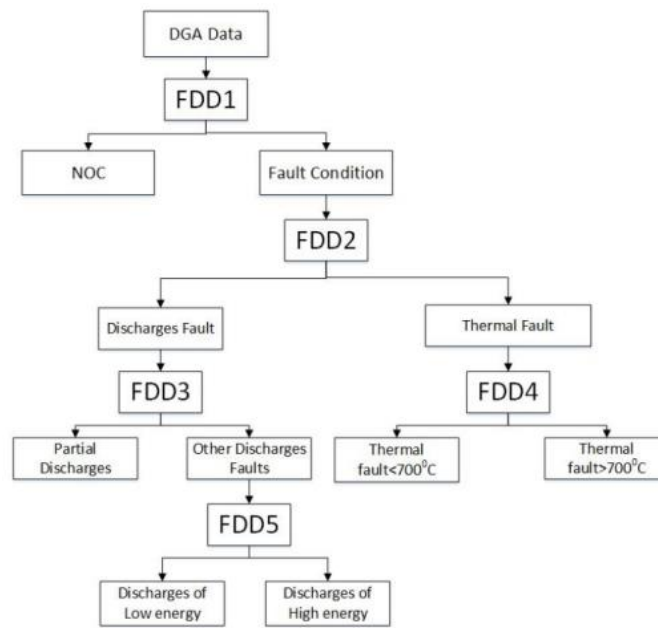


Figure 2. Data-Driven Fault Diagnosis System [16]

3.2. Data Description

The proposed fault diagnosis framework was implemented using 168 DGA data, consisting of 50 normal operating conditions (NOC) and 118 fault data with seven attributes originating from IEC TC 10 databases. The fault type distribution is shown in Table 1. A ratio of 80:20 was used to split the data into training and testing respectively.

Table 1. Distribution of DGA Instances

Fault type	PD	D1	D2	T1&T2	T3	Normal	Total
Train	6	15	28	10	10	25	94
Test	5	10	20	6	8	25	74

3.2.1. Data preprocessing: This is a crucial step in preparing data for training, ensuring it is in the right format and quality. It involves various procedures, such as data cleansing, balancing, imputing, normalizing, encoding, augmenting, and bias mitigation, to make the data suitable for further analysis and modelling.

3.2.2. Data normalization: Normalizing data prevents bias, improves algorithm convergence and speed and stabilizes variance. The normalization enhances model performance, interpretability, and the reliability of statistical analyses, by bringing all features to a common scale [33,39,40]. Min-Max normalization was used to standardize and transform the dataset using Equation 1.

$$f(x) = \frac{x_i - X_{min}}{(X_{max} - X_{min})} \tag{1}$$

where $f(X')$ is the normalized value, x_i the original values are X_{min} and X_{max} i.e. min and max values of all original values.

3.3. Classification algorithms

In the following subsection, the classifiers and their ensembles used to evaluate our proposal are briefly discussed. Five well-known classifiers of different families were used. Creating a composite global model with accurate and diverse estimates is the primary objective of the ensemble methodology over a single model. The original problem is solved by every model in the ensemble collection as it lowers the generalization error [39,40].

3.3.1. Multilayer Perceptron (MLP): One well-known "black box" neural network model is the MLP, which adjusts propagated error to reach an arbitrary level of accuracy through the use of a back-propagation algorithm [25,41]. The input vector x_i of the MLP is multiplied by a weight vector w_i , and summed with the bias b , to produce an output \hat{y} using the following Equations 2-5:

$$y_i = f(\sum_{i=1}^n w_i x_i + b) \tag{2}$$

where n stands for an input-output pairs, f stands for an activation function shown as:

$$f = \frac{1}{1+exp^{-x_i}} \tag{3}$$

$$E(\hat{y}, y) = \frac{1}{2} \sum_{i=1}^n (\hat{y} - y)^2 \tag{4}$$

where E is the error function.

$$\delta_i = \frac{dE}{dw} \tag{5}$$

Where δ_i and w are the gradient descent and weight respectively. One hidden layer is used.

3.3.2. Support Vector Machines (SVMs): In a binary classification or multi-class scenario, the Support Vector Machine (SVM) finds the hyperplane that can maximize the margin between distinct classes using a one-to-one or one-to-many technique [42]. The hyperplane is described in Equation 6. Equations 7 and 8 depict the dual issue of the objective using the Lagrange multiplier approach.

$$\hat{y}(x) = w^T \cdot x + b \tag{6}$$

$$\max_a \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{7}$$

$$s. t. \sum_{i=1}^n \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \dots, n \tag{8}$$

$$w = \sum_{i=1}^{N_{train}} \alpha_i y_i x_i \tag{9}$$

Once α_i is calculated, w can be obtained using Equation 9. The radial basis function is used as the kernel function and the regularization parameters as C and σ which are set to 100 and 10 respectively.

3.3.3. The Bayes Theorem: Is a probabilistic ML model based on the foundation of the Naive Bayes classification algorithm. The algorithm uses probability theory to predict the class of an input instance by calculating the conditional probabilities of each feature given its class [39,40].

$$P(y|X) = \frac{P(y)P(X|y=C_l)}{P(X)} = \frac{P(y) \prod_{i=1}^n P(x_i|y=C_l)}{P(X)} \tag{10}$$

Where, X is given data instance which is represented by its feature vectors (x_1, x_2, \dots, x_n) , y is a class target (normal or fault types). $P(y)$ and $P(X|y)$ are the prior and conditional probability of the outcome, while $P(X)$ represent the probability of the predictor values.

3.3.4. C4.5 Decision trees: To generate decision tree classifiers, the C4.5 methods were applied. Based on the idea of information gain, the C4.5 algorithm builds decision trees where the decisions made in each classification are connected to the target classification [4,43,44].

$$\text{InfoGain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X) \tag{11}$$

Entropy is the amount of uncertainty in the randomness of elements, and it is used to measure impurity.

$$\text{Entropy}(T, X) = - \sum_{i=1}^{c=7} p_i \log_2 (P_i) \tag{12}$$

3.3.5. Logistic Regression: Is a statistical technique for assessing the assumed relationships between the independent factors x and the dependent variable y . Our decision on logistic regression is that it is recommended for meta-level learning, and is used to combine the base learners [45].

$$y = \left(\frac{p(x)}{1-p(x)} \right) = \frac{e^{(\beta_0 + \beta_1 X)}}{1 + e^{(\beta_0 + \beta_1 X)}} \tag{13}$$

Where β_0 is the bias intercept term, β_1 is the coefficient for an input value, x is the input values, and y is the predicted output.

3.4. Ensemble Methods

Ensemble methods' [40] main goal is to create a composite global model with more accurate and reliable decision estimates than the single base models. The idea is to combine the results obtained from the multiple classifiers to increase accuracy and reduce generalization errors.

3.4.1. Adaboost: Adaboost uses the iterative ensemble approach to merge weak classifiers into a powerful strong classifier. The basic idea behind Adaboost is to use Equation 14 to illustrate how to build classifier weights and use the average majority vote [46] to train data samples to predict a class target of a given data instance with two classes.

$$\sum_{t=1}^T w_t d_{t,j}(x) = \max_{j=1}^C \sum_{t=1}^T w_t d_{t,j}(x) \tag{14}$$

where $d_{t,j}(x)$ represents support given by the t^{th} classifier to the j^{th} class for the instance x , w_t is the weight of classifier t and T is the total number of classifiers.

3.4.2. Bagging: Bagging, a technique developed from bootstrap aggregation, is the most straightforward yet efficient independent ensemble method for improving the accuracy of unstable learning algorithms. The datasets are split up among many bootstrap replicates during bagging. Every replication is made from the original dataset, which comprises, on average, 63.2% of the original data. The sluggish learner must go through multiple bootstraps repeatedly as part of the process. With every iteration, the weak learner's classifier is fused into a strong composite classifier, yielding better accuracy than any single component classifier could achieve [60]. The plurality voting method sometimes referred to as the majority voting system was used in this study and shown in Equation 15. This system is then utilized to calculate the total of all base learners.

$$\sum_{t=1}^T d_{i,j} = \max_{j=1}^C \sum_{t=1}^T d_{t,j}(x) \tag{15}$$

where the decision of the t^{th} classifier is defined as $d_{t,j} \in \{0,1\}$, $t = 1, \dots, T$ and $j = 1, \dots, C$. T represents the size of the classifiers, and C represents the size of the classes. If t^{th} chooses ω_j , then $d_{t,j} = 1$, otherwise 0.

3.4.3. Stacking: Stacking is an ensemble strategy to integrate heterogeneous models using a meta-classifier. SVM, C4.5, NB, and MLP are the five basis classifiers trained to extract the final outputs for predicting outcomes from the base classifier. Next, to avoid the overfitting issue brought on by the ensemble's base models, logistic regression is used as the meta-classifier [43].

3.4.4. Voting: Voting is decisions ensemble method in machine learning where multiple independent models are trained heterogeneously or homogeneously on the same dataset and their predictions are used to make final decisions by choosing the most frequent prediction. The combining of the predictions of the classifiers can proceed in multiple ways either using majority voting or weighted voting [48,49]. This research study adopted the majority voting as in Equation 16.

$$class(x) = \arg \max_{c_i \in dom(y)} \sum_k g(y_k(x), c_i) \tag{16}$$

where $y_k(x)$ is the classification of the k^{th} classifier, and $g(y_k(x))$ is an indicator function defined as:

$$g(y_k(x)) = \begin{cases} 1 & y = c \\ 0 & y \neq c \end{cases} \tag{17}$$

4. Performance Evaluation Methods

To evaluate the behavior of models concerning the applicability and performance, several evaluation measures need to be defined. Measures including classification accuracy, F-Measure, precision, and recall are frequently employed to highlight reducibility power of the classification models [50–52]. The classification results, which are frequently recorded in a matrix format called a Confusion Matrix summarizes the outcome of the algorithm, and is used to determine these metrics [53] with the following four outcomes as illustrated in Table 2 [54], and represented in equations 18 through 22.

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} \tag{18}$$

$$Precision = \frac{TP}{TP+FP} \tag{19}$$

$$Sensitivity = \frac{TP}{TP+FN} \tag{20}$$

$$Specificity = \frac{TN}{TN+FP} \tag{21}$$

$$F1 - score = 2 \frac{(P * Sn)}{(P + Sn)} \tag{22}$$

Table 2. Confusion Matrix

<i>Confusion matrix</i>		<i>Classifier</i>	
		<i>Negative</i>	<i>Positive</i>
<i>Actual class</i>	<i>Negative</i>	<i>TP</i>	<i>FP</i>
	<i>Positive</i>	<i>FN</i>	<i>TP</i>

5. Results and Discussions

This section discusses the results of the proposed model about the standard metrics applied for evaluation. The default parameters of the models were applied in this study.

5.1. Results

The results obtained from the analysis of each data split – training and testing datasets, are presented in Tables 3, 4, and 5 respectively. Figures 3 to 11 depict the graphical visualizations of the confusion matrices of both the single and ensemble models.

5.2. Discussion

Table 3 shows the decision tree (C4.5) classifier obtained the highest accuracy of 98.33% compared to the naïve Bayes and SVM classifiers with an accuracy of 91.67% and 85.00% respectively. C4.5 performed better than others because it mitigates issues like overfitting, and incomplete data, and can handle discrete and continuous data respectively. In Table 4, the stacking ensemble model shows significant improvement over others by obtaining an accuracy of 99.89%, while the bagging, boosting and voting ensemble obtained an accuracy of 83.33%, 96.95%, and 93.33% respectively. The stacking ensemble performed better because it is a heterogeneous ensemble model comprising the three base classifiers of SVM, C4.5, and naïve Bayes respectively.

Table 3. Performance for Single Classification Models

<i>Models</i>		<i>F-Score%</i>	<i>Precision%</i>	<i>Recall%</i>	<i>Accuracy%</i>
SVM	<i>Training</i>	83.65	84.97	85.00	85.50
	<i>Testing</i>	84.80	86.21	85.00	85.00
naive Bayes	<i>Training</i>	90.05	92.49	91.67	91.30
	<i>Testing</i>	92.10	93.29	91.67	91.67
Decision Trees	<i>Training</i>	98.90	99.98	99.92	99.95
	<i>Testing</i>	97.80	98.52	98.33	98.33

Table 4. Ensemble Models Performance

<i>Models</i>		<i>F-Score%</i>	<i>Precision%</i>	<i>Recall%</i>	<i>Accuracy%</i>
Bagging (SVM)	<i>Training</i>	82.05	83.75	83.75	83.75
	<i>Testing</i>	82.65	83.33	83.33	83.33
Boosting (NB)	<i>Training</i>	96.07	97.68	97.50	97.50
	<i>Testing</i>	95.89	96.95	96.67	96.95
Stacking	<i>Training</i>	99.05	99.92	99.83	99.97
	<i>Testing</i>	99.50	99.98	99.95	99.89
Voting	<i>Training</i>	94.64	95.50	95.00	95.00
	<i>Testing</i>	93.65	94.20	93.33	93.33

The visualization of Figures 3 to 9 depicts the various models’ misclassification. Out of the 60 samples presented, the SVM misclassified 9, naïve Bayes misclassified 5, and C4.5 misclassified 1. From the single models, C4.5 proves to be the best model with the least misclassification error. The bagged SVM misclassified 10 samples, the boosting of naïve Bayes misclassified 1 sample, the voting ensemble misclassified 4 samples and the stacking ensemble misclassified 0 samples respectively. The stacking ensemble was able to achieve this feat because it can reduce the bias and variation of the models merged and in turn increase the overall predictive performance significantly. Finally, our proposed model is compared with existing models as shown in Table 5. While our model obtained a 99.89% accuracy, others were 86.25%, 93.13%, 97.14%, 99.20%, and 87.18% respectively. Our model shows to be better than the five existing works except the work of [27], with a 0.09 difference, thereby justifying the need for future improvement of the models.

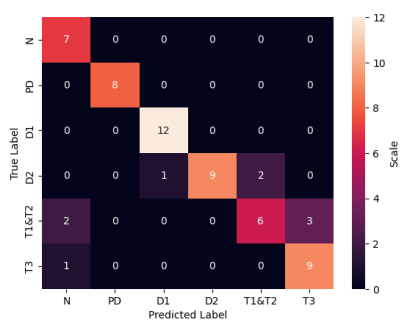


Figure 3. SVM Classifier

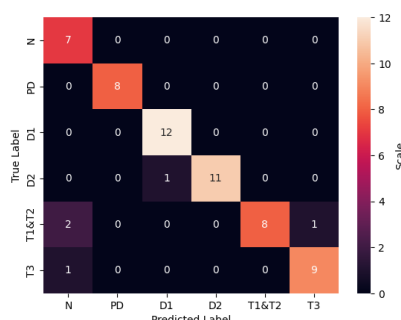


Figure 4. naive Bayes

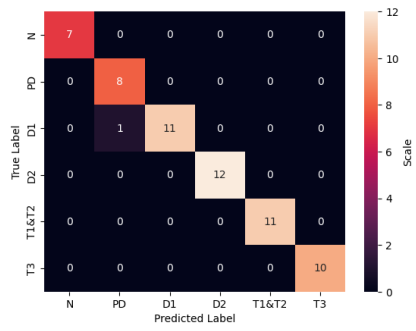


Figure 5. C4.5 DT

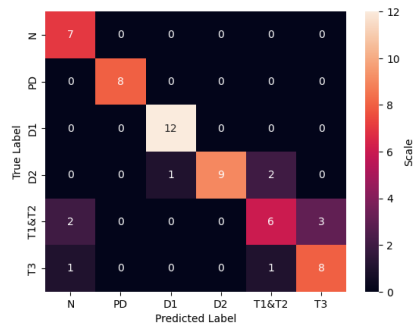


Figure 6. Bagging (SVM)

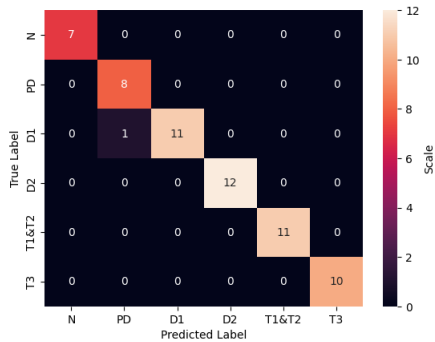


Figure 7. Boosting (naive Bayes)

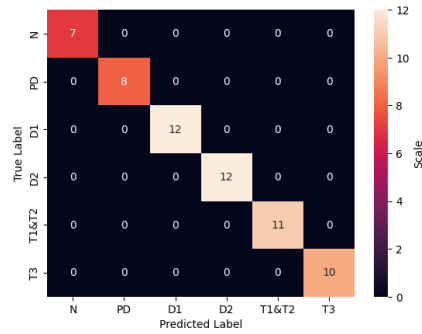


Figure 8. Stacking

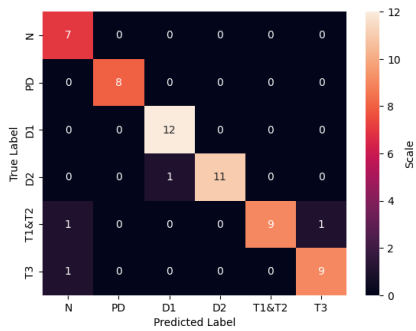


Figure 9. Voting

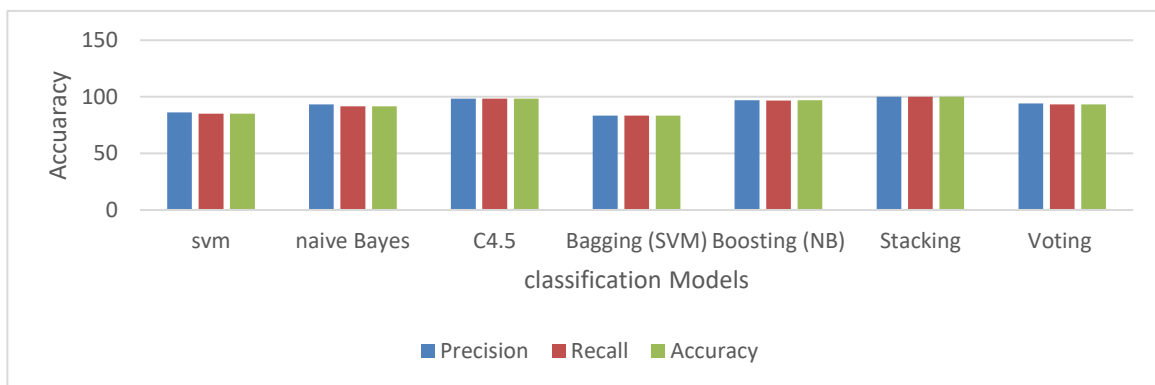


Figure 10. Performance Metrics of all Classification Models

Table 5. Comparison of Different Model Accuracies

Reference	Research methods	Accuracy (%)
Proposed model	Stacking	99.89
[4]	<i>naive Bayes</i>	86.25
[17]	<i>IGA-XGBoost</i>	99.20
[27]	<i>KosaNet</i>	99.98
[28]	<i>Ensemble (6) classifiers</i>	97.14
[29]	<i>SVM</i>	86.18
[35]	<i>Decision tree</i>	93.13

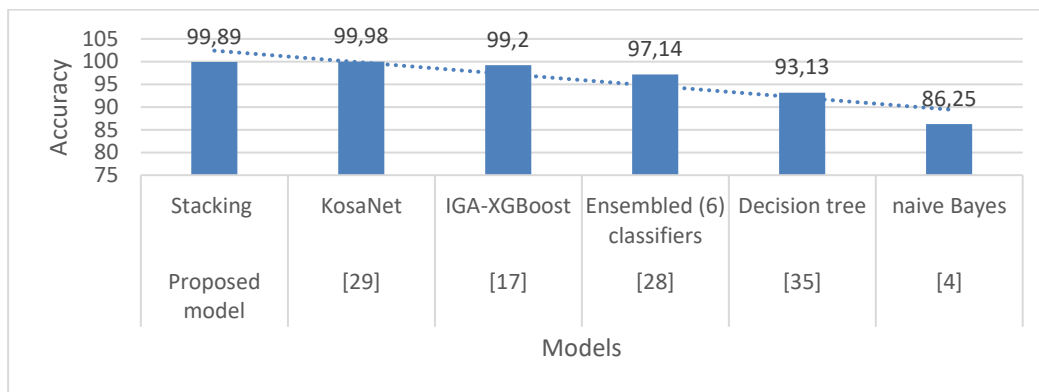


Figure 11. Comparison of Different Model Accuracy

6. Conclusion

In this study, we used seven classification methods and normalization approaches to create a power transformer fault types diagnostic model on 168 DGA data points with seven features obtained from IEC TC 10 databases. The seven classification models used were C4.5, SVM naive Bayes, and four ensemble methods: voting, stacking, bagging and boosting. The transformer fault detecting accuracies of the C4.5 decision tree classifier and the stacking ensemble models showed better performances than others with an accuracy of 98.33% and 99.89% respectively. The proposed model was compared with other existing works validating the superiority and adequacy of our proposed model with prospects. In future work, the authors hope to extend the input space, introduce other data transformation techniques like the Synthetic Minority Oversampling Technique (SMOTE) to handle data imbalance problems, introduce other DGA datasets from different domains, with improved machine learning algorithms and also introduce the concept of model interpretability. Also, we intend to extend the ensemble variants using other machine-learning techniques.

References

- [1] Duval M, dePablo A (2001) Interpretation of Gas-In-Oil Analysis Using New IEC Publication 60599 and IEC TC 10 Databases. IEEE Electrical Insulation Magazine, vol. 17, no. 2, pp. 31–41.
- [2] Izzularab MA, Aly, GEM, Mansour DA (2004) On-line diagnosis of incipient faults and cellulose degradation based on artificial intelligence methods”, IEEE Int. Conf. on Solid Dielectrics, Toulouse, France.
- [3] Hoballah, A., Mansour, J. G. and Taha IBM, (2020) Hybrid grey wolf optimizer for transformer fault diagnosis using dissolved gases considering uncertainty in measurements. IEEE Access, vol. 8, pp. 139176–139187.
- [4] Mahamdi Y, Boubakeur A, Mekhaldi A, Benmahamed Y (2022) Power Transformer Fault Prediction using Naive Bayes and Decision tree based on Dissolved Gas Analysis. ENP Engineering Science Journal, Vol. 2, No. 1. Digital Object Identifier (DOI): 10.53907/enpesj.v2i1.63.
- [5] Cheng L, Yu T (2018) Dissolved gas analysis principle-based intelligent approaches to fault diagnosis and decision making for large oil-immersed power transformers: A survey. Energies 11, 913. doi:10.3390/en11040913
- [6] Fu Wan W, Weigen Chen W, Xiaojuan Peng X, Jing Shi J (2012) Study on the Gas Pressure Characteristics of Photoacoustic Spectroscopy Detection for Dissolved Gases in Transformer Oil. In 2012 International Conference on High Voltage Engineering and Application. IEEE, 286–289. doi:10.1109/ICHVE.2012.6357108.
- [7] Chen Xi, Chen W, Gan D (2010) Properties and Gas Production Law of Surface Discharge in Transformer Oil-Paper Insulation. In 2010 Annual Report Conference on Electrical Insulation and Dielectric Phenomena. IEEE, 1–4. doi:10.1109/CEIDP.2010.5724049.
- [8] Zeng W, Yang Y, Gan C, Li H, Liu G (2011) Study on Intelligent Development of Power Transformer On-Line Monitoring Based on the Data of DGA. In 2011 Asia-Pacific Power and Energy Engineering Conference. IEEE, 1-4. doi:10.1109/appeec.2011.5749107
- [9] Jiang XQ, Gong Y, Han S, Zhou K (2014) Application of the Improved Three-Ratio Method in Chromatographic

- Analysis of Locomotive Transformer Oil. *Amr* 1030-1032, 29–33. doi:10.4028/www.scientific.net/amr.1030-1032.29
- [10] Dhote N, Helonde J (2012) Diagnosis of Power Transformer Faults Based on Five Fuzzy Ratio Method. *WSEAS Trans. Power Syst.* 7, 12.
- [11] Liu Z, Song B, Li E, Mao Y, Wang G (2015) Study of "code Absence" in the IEC Three-Ratio Method of Dissolved Gas Analysis. *IEEE Electr. Insul. Mag.* 31, 6-12. doi:10.1109/MEI.2015.7303257
- [12] Shang H, Xu J, Zheng Z, Qi B, Zhang L (2019) A Novel Fault Diagnosis Method for Power Transformer Based on Dissolved Gas Analysis Using Hypersphere Multiclass Support Vector Machine and Improved D-S Evidence Theory. *Energies* 12, 4017. doi:10.3390/en12204017
- [13] Singh S, Bandyopadhyay M (2010) Dissolved Gas Analysis Technique for Incipient Fault Diagnosis in Power Transformers: A Bibliographic Survey. *IEEE Electr. Insul. Mag.* 26, 41–46. doi:10.1109/MEI.2010.5599978
- [14] Bhalla D, Bansal RK, Gupta HO (2011) Transformer Incipient Fault Diagnosis Based on DGA using Fuzzy Logic. *IEEE*, pages 1-5.
- [15] Yadaiah N, Ravi N (2011) Internal Fault Detection Techniques for Power Transformers. *Appl. Soft Comput.* 11, 5259–5269. doi:10.1016/j.asoc.2011.05.034
- [16] Dhini A, Faqih A, Kusumoputro B, Surjandari I, Kusiak A (2020) Data-driven fault Diagnosis of Power transformers using Dissolved Gas Analysis (DGA). *International Journal of Technology.* 11(2), 388-399. <http://ijtech.eng.ui.as.id>
- [17] Wu Z, Zhou M, Lin Z, Chen X, Huang Y (2021) Improved Genetic Algorithm and XGBoost Classifier for Power Transformer Fault Diagnosis. *Front. Energy Res.* 9:745744. doi: 10.3389/fenrg.2021.745744.
- [18] Yi J-H, Wang J, Wang G-G (2016) Improved Probabilistic Neural Networks with Self-Adaptive Strategies for Transformer Fault Diagnosis Problem. *Adv. Mech. Eng.* 8, 168781401562483. doi:10.1177/ 1687814015624832
- [19] Mani G, Jerome J (2014) Intuitionistic Fuzzy Expert System Based Fault Diagnosis Using Dissolved Gas Analysis for Power Transformer. *J. Electr. Eng. Tech.* 9, 2058–2064. doi:10.5370/JEET.2014.9.6.2058
- [20] Parihar VR, Nimkar DS, Warudkar S, Deshmukh R, Thakare M (2017) Power Transformer Protection using Fuzzy Logic based Controller. *International Journal of Engineering Research.* Volume No.6, Issue No.7, pp: 366-370
- [21] Fan J, Wang F, Sun Q, Bin F, Liang F, Xiao X (2017) Hybrid RVMANFIS Algorithm for Transformer Fault Diagnosis. *IET Generation, Transm. Distribution* 11, 3637–3643. doi:10.1049/iet-gtd.2017.0547
- [22] Yin J, Zhu Y, Yu G (2011) Power Transformer Fault Diagnosis Based on Support Vector Machine with Cross Validation and Genetic Algorithm. In 2011 International Conference on Advanced Power System Automation and Protection. *IEEE*, 309–313. doi:10.1109/APAP.2011.6180419
- [23] Benmahamed Y, Tegar Y, Boubakeur A (2017) Application of SVM and KNN to Duval Pentagon 1 Transformer Oil Diagnosis. *IEEE Trans. Dielect. Electr. Inst.*, 24, 3443–3451, 2017
- [24] Aizpurua JI, Catterson VM, Stewart BG, McArthur SDJ, Lambert B, Ampofo B, Pereira G, Cross JG (2018) Power transformer dissolved gas analysis through Bayesian networks and hypothesis testing. *IEEE Trans. Dielectrics Electr. Insul.*, vol. 25, no. 2, pp. 494–506, Apr. 2018. DOI: 10.1109/TDEI.2018.006766.
- [25] Igodan CE, Ojietohamen SE, Izevbizua RI (2023) Cervical cancer prediction using ensemble models, NIPES Conference proceedings. The 2nd International conference 15th-17th Feb. 2023, pp.71-83. www.nipesjournals.org.ng
- [26] Yuan F, Guo J, Xiao Z, Zeng B, Zhu W, Huang S (2019) A Transformer Fault Diagnosis Model Based on Chemical Reaction Optimization and Twin Support Vector Machine. *Energies* 12, 960. doi:10.3390/en12050960
- [27] Li J, Zhang Q, Wang K, Jianyi W (2015) Optimal Dissolved Gas Ratios Selected by Genetic Algorithm for Power Transformer Fault Diagnosis Based on Support Vector Machine. *IEEE Transactions on Dielectrics and Electrical Insulation* Vol. 23, No. 2. DOI: 10.1109/TDEI.2015.005277.
- [28] Taha IBM, Mansour DEA (2021) Novel Power Transformer Fault Diagnosis Using Optimized Machine Learning Methods. *Intelligent Automation & Soft Computing.* IASC, 2021, vol.28, no.3. DOI:10.32604/iasc.2021.017703
- [29] Odongo G, Musabe R, Hanyurwimfura DA (2021) Multinomial DGA Classifier for Incipient Fault Detection in Oil-Impregnated Power Transformers. *Algorithms*, 14, 128. <https://doi.org/10.3390/a14040128>.
- [30] Dapshima BA, Essa YC, Chaturvedi S (2023) Fault Detection and Protection of Power Transformer Using Fuzzy Logic. *International Journal for Research in Applied Science and Engineering Technology (IJRASET).* Volume 11 Issue I. www.ijraset.com
- [31] Li J, Zhang Q, Wang K, Jianyi W (2015) Optimal Dissolved Gas Ratios Selected by Genetic Algorithm for Power Transformer Fault Diagnosis Based on Support Vector Machine. *IEEE Transactions on Dielectrics and Electrical Insulation* Vol. 23, No. 2. DOI: 10.1109/TDEI.2015.005277.
- [32] Tao L, Yang X, Zhou Y, Yang LA (2021) Novel Transformers Fault Diagnosis Method Based on Probabilistic Neural Network and Bio-Inspired Optimizer. *Sensors*, 21, 3623. <https://doi.org/10.3390/s21113623>.
- [33] Huang X, Yuan Y, Li J (2023) A Review of Transformer Fault Diagnosis Based on Information System Theory and Machine Learning. *Preprints*, pp.1-12. doi:10.20944/preprints202305.0036.v1.
- [34] Bouchaoui L, Hemsas KE, Mellah H, Benlahneche S (2021) Power Transformer faults diagnosis using undestructive methods (Rogers and ICE) and artificial neural network for dissolved gas analysis applied on the functional transformer in the Algerian north-eastern: A comparative study. *Electrical Engineering & Electromechanics* 2021(4):3-11. DOI: 10.20998/2074-272X.2021.4.01.
- [35] Demirci M, Gozde H, Taplamacioglu MC (2021) Fault Diagnosis of Power Transformers with Machine Learning Methods using Traditional Methods Data. *International Journal on Technical and Physical Problems of Engineering.*

- Issue 49, Volume 13, Number 4, pp. 225-230
- [36] Islam MM, Lee G, Hettiwatte, SN (2018) Application of Parzen Window estimation for incipient fault diagnosis in power transformers. The institute of engineering and technology. doi: 10.1049/hve.2018.5061.
- [37] Ma H, Zhang W, Wu R, Yang C (2018) A Power transformers fault diagnosis model based on three DGA ratios and PSO optimization SVM. IOP Conf. Series: Materials Science and Engineering 339 012001. doi:10.1088/1757-899X/339/1/012001.
- [38] Taha IBM, Mansour DEA (2021) Novel Power Transformer Fault Diagnosis Using Optimized Machine Learning Methods. Intelligent Automation & Soft Computing. IASC, 2021, vol.28, no.3. DOI:10.32604/iasc.2021.017703
- [39] Igodan EC, Obe O, Thompson A.F-B. and Owolafe., O. (2022a). Erythematous Squamous Disease Prediction using Ensemble Multi-Feature Selection Approach. International Journal of Computer Science and Information Security (IJCSIS), Vol. 20, No. 2, pp. 95-106.
- [40] Igodan EC, Obe, O., Thompson, AF-B, Owolafe O (2022b) Prediction of erythematous Squamous-disease using ensemble learning framework. The Institute of Engineering and Technology. In Explainable Artificial Intelligence in Medical Decision Systems, pp.197-228.
- [41] Igodan CE, Ukaoha KC (2019) Using Multilayer Perceptron and Deep Neural Networks for the Diagnosis of Breast Cancer Classification”, 2019 IEEE AfriCon, pp. 1-7, doi:10.1109/AFRICON46755.2019.9133873.
- [42] Vapnik CV, (1995) Support-vector networks. Machine Learning 20, 3, 1995, 273. doi:10.1007/BF00994018.
- [43] Liu J, Ning B, Shi D (2019) Application of Improved Decision Tree C4.5 Algorithms in the Judgment of Diabetes Diagnostic Effectiveness. IOP Conf. Series: Journal of Physics: Conf. Series 1237 (2019) 022116. doi:10.1088/1742-6596/1237/2/022116.
- [44] Xuanyuan S, Xuanyuan S, Yue Y (2022) Application of C4.5 Algorithm in Insurance and Financial Services Using Data Mining Methods. Mobile Information Systems, vol. 2022, Article ID 5670784, 8 pages, 2022. <https://doi.org/10.1155/2022/5670784>.
- [45] Witten IH, Frank E, Hall M A, Pal CJ (2017) Data Mining: Practical Machine Learning Tools and Techniques. Fourth Edition, Cambridge, U.S.A.
- [46] Seijo-Pardo B, Porto-Diaz I, Bolon-Canedo V, Alonso-Betanzos A (2017) Ensemble Feature Selection: Homogeneous and Heterogeneous Approaches. Knowledge-Based System. 2017; 118:124-139.
- [47] Liu J, Dong X, Zhao H, Tian Y (2022) Predictive Classifier for Cardiovascular Disease Based on Stacking Model Fusion. Processes. 10(4):749. <https://doi.org/10.3390/pr10040749> [Assessed 27th March 2022]
- [48] Rokach L (2019) Ensemble Learning Pattern Classification Using Ensemble Methods. Second Edition. Volumn 85. ISSN: 1793-0839. <https://dokumen.pub/ensemble-learning-pattern-classification-using-ensemble-methods-2nbsped-9811201951-9789811201950.html>
- [49] Tattar PN (2018) Hands-On Ensemble Learning with R: A beginners’ guide to combining the power of machine learning algorithms using ensemble techniques. <https://dokumen.pub/hands-on-ensemble-learning-with-r-a-beginners-guide-to-combining-the-power-of-machine-learning-algorithms-using-ensemble-techniques-1788624149-9781788624145.html>
- [50] Villacampa O (2015) Feature Selection and Classification Methods for Decision Making: A Comparative Analysis. Doctoral Dissertation Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. https://nsuworks.nova.edu/gscis_etd/63.
- [51] Bolon-Canedo V, Sanchez-Marono N, Alonso-Betanzos A (2015) Feature Selection for High-Dimensional Data. Springer.
- [52] Brownlee J (2019) Statistical methods for machine learning: Discover how to transform data into knowledge with Python. https://machinelearningmastery.com/statistics_for_machine_learning/
- [53] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16 (2002) 321-357.
- [54] Shehzad K, Zhenhua T, Shoukat S, Saeed A, Ahmad I, Sarwar Bhatti S, Chelloug, SAA (2023) Deep-Ensemble-Learning-Based Approach for Skin Cancer Diagnosis. Electronics 12, 1342. <https://doi.org/10.3390/electronics12061342>.
- [55] Witten IH, Frank E (2000) Data Mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco.
- [56] Ghoneim SSM, Taha IBM, Elkalashy NI (2016) Integrated ANN Based Proactive Fault Diagnostic Scheme for Power Transformers Using Dissolved Gas Analysis. IEEE Trans. Dielect. Electr. Insul. 23, 1838–1845. doi:10.1109/TDEI.2016.005301
- [57] Li J, Chen X, Wu C (2009) Application of Comprehensive Relational Grade Theory in Expert System of Transformer Fault Diagnosis. In 2009 International Workshop on Intelligent Systems and Applications. IEEE, 1-4. doi:10.1109/iwisa.2009.5072742
- [58] Zarkovic M, Stojkovic Z (2017) Analysis of Artificial Intelligence Expert Systems for Power Transformer Condition Monitoring and Diagnostics. Electric Power Syst. Res. 149, 125–136. doi:10.1016/j.epsr.2017.04.025
- [59] Zhu X, Xiong J, Liang Q (2018) Fault Diagnosis of Rotation Machinery Based on Support Vector Machine Optimized by Quantum Genetic Algorithm. IEEE Access 6, 33583–33588. doi:10.1109/ACCESS.2018.2789933
- [60] Zhou Z-H (2012) Ensemble methods: foundation and algorithms. CRC Press.

- [61] Manisha, Kaur, K, Sharma NK, Singh J, Bhalla D (2022) Performance Assessment of IEEE/IEC Method and Duval Triangle technique for Transformer Incipient Fault Diagnosis

Conflict of Interest

Author declares that there are no potential conflicts of interest.

Funding

This research did not receive a specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography

Plagiarism Statement

This article has been scanned by iThenticate™.

Investigation of Digital Calibration Certificate - Digital Test Report Sharing in Metrology Network

Erkan Danacı¹, Bülent Aydemir²

¹RF & Microwave Laboratory, TUBITAK UME, Kocaeli Türkiye

²Düzce University, Mechatronic Eng., Düzce, Türkiye

Corresponding author:

RF & Microwave Laboratory,
TUBITAK UME, Kocaeli, Türkiye
Erkan.danaci@tubitak.gov.tr



Article History:
Received: 06.03.2024
Accepted: 12.08.2024
Published Online: 29.08.2024

ABSTRACT

With the development of technology, digitalization studies have begun in metrology applications, and digital certificates in accordance with the ISO 17025 standard have started to be produced. There is a need for a platform where digital calibration certificates (DCC) / digital test report (DTR) produced today and to be created in the future are shared. In this study, the topics that digital calibration certificates / digital test report must include according to the ISO 17025 standard are explained, and information is given about the Blockchain structure, which is currently widely used in the financial field. The Blockchain platform suggested for use as a digital calibration certificates / digital test report sharing platform for metrological needs is given in this publication. Suggestions on how to run the process for using Blockchain in digital calibration certificates / digital test report sharing are also given in this study.

Keywords: Blockchain, Digital calibration certificate, Digital test report, Local metrology network, Wide metrology network

1. Introduction

Technological changes have become mandatory in the manufacturing equipment, measurement devices and service provider systems in the age of Industry 4.0. The fact that production and measurement devices are constantly communicating and that all communication data is digital has made digital transformation mandatory in all sectors. There are significant changes in measurement devices and output of the measurement devices such as measurement results and measurement report by the Industry 4.0 evolution. Digital transformation is one of the most essential topics that came from technological change in metrology nowadays. In particular, the digitization of metrological outputs and the sharing of these outputs in digital networks are new important research topics.

Many scientists and researchers are studying to adapt measuring devices to the digital world [1-5]. In the digital transformation of measuring devices, studies continue to not only create digital copies of the devices on the computer but also make simulations in the computer environment by providing communication between digital device models and real devices. Along with these studies, real devices are also identified in the digital environment.

Identifying measuring devices in the digital format can be under two steps. The first is the digitalization of the product identification data, and the second is the digitalization of the calibration certificates or test reports.

Product identification data of a measurement device is the data such as manufacturer, brand, model number, and serial number. Nowadays, manufacturers give these with a barcode or data matrix. Digital transformations of measuring devices such as electricity, gas, and water meters that people use daily have begun to be made with legal regulations to protect consumer rights. However, this digital transformation has not been defined as the internationally accepted format of machine-readable Digital Test Report (DTR) has yet to be completed. While measuring devices such as electricity, gas, and water meters directly affect people's quality of life, digital transformations of measurement devices used in production indirectly affect people's quality of life.

Calibration certificates and test reports are important documents that provide information about the measurement device's current status. For these documents to be reliable and acceptable to everyone, the National Metrology Institute (NMI) and Designated Institute (DI) recognized by the Bureau of Weights and Measures (BIPM) or calibration and testing laboratories

approved by accreditation authorities should be created. The measurement method, the special conditions in the measurements, the environmental conditions, the person who performed the measurement, and the laboratory information are given together with the measurement results and uncertainties in the certificates and reports currently. The abovementioned information is defined in detail in the ISO/IEC 17025 standard [6]. Project studies continue to transfer calibration certificates and test reports to digital media. These projects aimed to create a machine-readable certificate that can be output when necessary. For this purpose the working group led by Physikalisch-Technische Bundesanstalt (PTB, National Metrology Institute of Germany) created the machine-readable Digital Calibration Certificate (DCC) format. For DCC to become widespread, PTB shares its study outputs via its web page (www.ptb.de/dcc/).

When the digital transformation of the measuring devices used in production is completed, and a digital calibration and test report is created, it will be necessary to share the device definition and current device data in the digital platform. These sharing studies are the newest research topics nowadays. Digital data sharing needs to be considered comprehensively, from doing it in regional networks to doing it in global networks. We can think of regional networks where digital data of measuring devices are shared regionally, such as accreditation authorities of countries, associations of calibration and test laboratories in countries, and country quality infrastructure. When we generalize about the world, we can also call the BIPM network, of which Regional Metrology Organizations (RMO) are members, a global network.

In this study, the non-digital and digital data contents of the measurement devices are given and especially the calibration certificate as process output by measuring devices are detailed, and some predictions for Blockchain are given on how to share digital outputs in regional networks. The features of the Blockchain general structure suitable for sharing digital calibration certificates or digital test reports are stated, and the application of digital device output data into the Blockchain is detailed in this study.

This study proposes a new concept for DCC sharing in Blockchain. DCCs and DTRs must be produced to have application results. DCCs/DTRs are in the production phase, and these digital documents are not shared on platforms such as Blockchain now. Sharing examples on Blockchain will only be possible after DCC DTRs are produced. Therefore, this study only contains a concept proposal and does not include application results.

2. Digital Data Contents of Measurement Devices

The ISO/IEC 17025 standard is a reference document that sets out the rules that must be followed by all institutions and organizations that provide calibration and testing services [6]. In article 7.8 of the ISO/IEC 17025 document, the content of the calibration certificate is clearly defined. The digital data contents of measuring devices are also detailed in the digital calibration certificate portal by PTB [1, 7, 8, 9, 10]. Measurements are usually presented to the customer in a report (for example, a test report (TR), a calibration certificate (CC), or a report of sampling) to be provided accurately, clearly, concisely, and objectively. All drafted reports should contain as a technical record all the information necessary to interpret the results and all required information by the method used. Report detail can be separated into four topics as given below.

- Administrative Data
- Conditions Data
- Result of Measurements Data
- Comments

Test reports or calibration certificate should include at least the information in Table 1 according to the above separation as ISO standard requirements. In Table 1, the ADD data type represents administrative data, the CON data type represents environmental conditions data, the ROM data represents the measurement results, and the COM data type represents special situations that are taken into consideration when making measurements.

After creating the calibration certificate or test report, it is an acceptable of the current device information, and it cannot be changed at any time. If there is a typo or wrong information on the created certificate or report, a new certificate or report is created with a new number. In this new document, the reason for the new document creation can be given, and this new document is acceptable of the current device information.

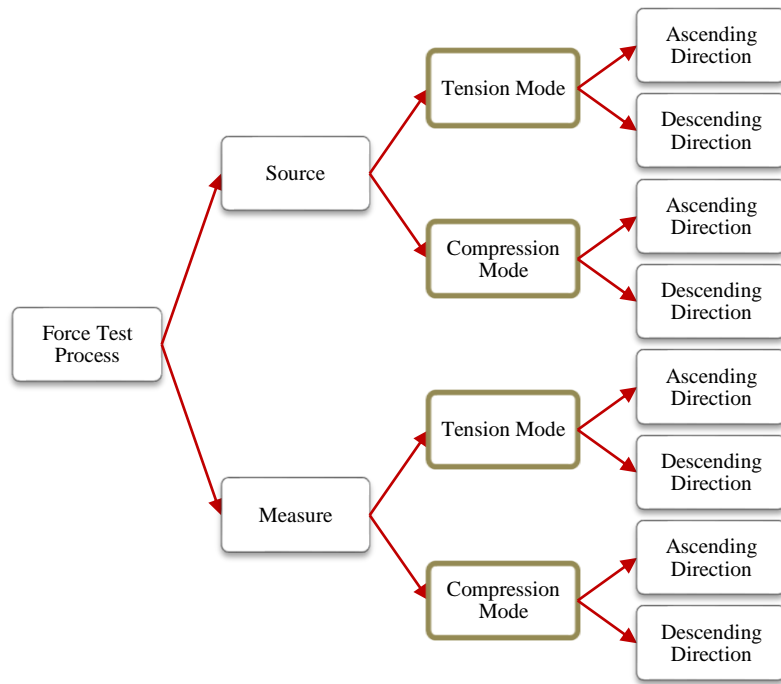
CCs/TRs are human-readable formats that can be converted into forms such as PDFs and shared in digital environments. Documents converted to PDF in human-readable form are large in size and require large memory storage. DCCs/DRTs are machine-readable formats that are smaller in size compared to CCs/TRs and are easier to distribute and share digitally.

Table 1. Digital Certificate Information and Types

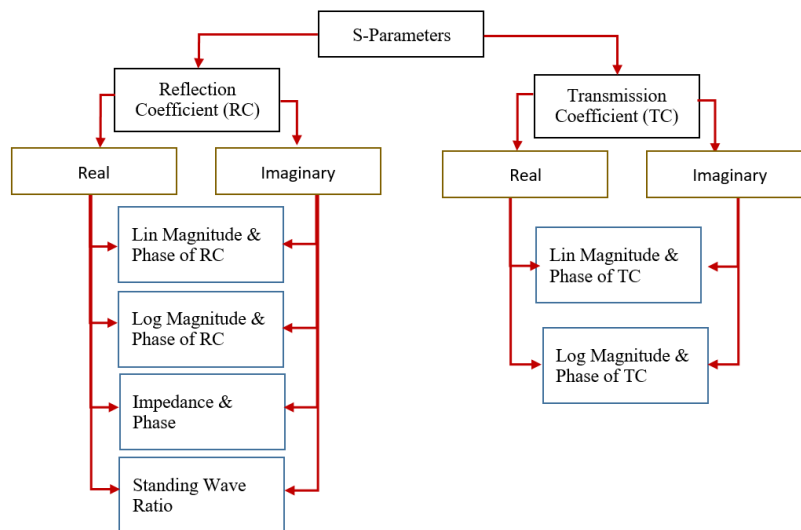
Name in ISO/IEC 17025 Standard	Data Type	Name in ISO/IEC 17025 Standard	Data Type
Document Title	ADD	Date of The Performance	CON
Laboratory's Name and Address	ADD	To Be Used Method	CON
Document Number	ADD	Special Conditions	CON
Location of The Performance	ADD	Environmental Conditions	CON
Identification Of Device Under Test	ADD	Used Software	CON
Persons	ADD	Measuring Equipment Description	CON
External Providers	ADD	The Statement to the Effect That the Results	COM
Measurement Results	ROM	Adjustment or Repairment	COM
Additions To, Deviations	ROM	Statement of Conformity	COM
Measurement Uncertainty	ROM	Opinions and Interpretations	COM

Together with the DCC, it is developing a new method for exchanging measurement data and certificate values between digital systems. The studies for DCC creation aimed at creating machine-readable report with the calibration data contained in the calibration process, instrument specifications, measurement conditions and accreditation scopes. This process is defined as taxonomy. Taxonomy is a basic concept a classification of all kinds of things. Taxonomy in DCC metrology explains the structure of DCC.

A measurement taxonomy database is intended to be a working reference database of measurement categories or type definitions. It will describe the broad parameters of each measurement without specific knowledge of the measurement technique or special equipment used. Taxonomy is necessary because a unit of measure can be an extremely ambiguous value [11, 12]. For example, a 90° angle measurement can easily be misinterpreted without knowing the measurement type. Is it a physical angle? Maybe an electric phase? Perhaps the rotation angle of a force transducer? This misunderstanding problem aims to be solved by providing certain information about the data beyond the simple unit of measure with taxonomy. Some taxonomy examples are given in Figure 1. The force measurement in tests are given in Figure 1(a). The applied force is classified in Figure 1(a), according to the force direction and application type such as compression/tension and ascending/descending. S parameter measurement in the calibration process is given in Figure 1(b). The real and imaginary components of vector reflection coefficient and vector transmission coefficients are essential measured quantities for s parameter measurement in RF metrology. All of the other measurement units are derived by real and imaginary component are given in Figure 1(b). Studies on the taxonomy of measurement quantities have been carried out by BIPM and since 2021, the digitalization studies of these taxonomies have started to be published on the BIPM website [13]. Since both the certificates on which the measurement values are printed are digitalized and the measurement quantities are defined digitally, in the following process, it is important to share the certificates and reports that will be produced digitally in digital environments such as Blockchain.



(a)



(b)

Figure 1. (a) The Force Measurement Taxonomy Diagram, (b) The RF Scattering Parameter Measurement Taxonomy Diagram

3. The Blockchain Structures and Data Sharing in the Blockchain

When we look at the history of digital transformation of IT systems, we first encounter electro-mechanical system transformations before the 1960s. Central data storage systems (Mainframe) were developed in the 1960s, and in 1975 and later, personal computers (PC, laptop, handheld computers) took their place in the markets. While Central and Distributed systems were developed in the 1980s, Decentralized systems (web3, p2p, ownership in digital assets, Blockchain-based trust) were developed in 2000 and beyond [14, 15].

The Blockchain systems are a technological innovation that has shown itself in crypto money systems today but has much more usage areas. It is a disruptive yet transformative technology, as it eliminates the need for intermediaries, trusted, and third parties in terms of trust. The Blockchain is a reliable network that stores records of transactions between peers in a secure and tamper-proof way.

The Blockchain technology is built on six fundamental principles. These are; decentralization consensus, a trustable system, immutability, tamper-resistant, transparency, highly accusable, and security [16].

In digital data storage systems, a digital data registry that is not centrally controlled and stored centrally, managed by peer nodes (Peer-to-Peer Nodes), copies of which can be updated by consensus and multiplexed in different locations is called “Distributed Ledger Technology (DLT)”. DLT refers to a decentralized technological infrastructure that is not dependent on specific centres, where encrypted and fragmented data can be accessed, verified and updated in multiple areas on the network. The DLT structure is shown in Figure 2 [17]. A distributed ledger technology (DLT) is a technology that facilitates an expanding, historically chronologically ordered list of cryptographically signed, irreversible transaction records shared by all participants in a network. Any participant with the right access rights can track a transaction event belonging to any actor in the network at any point in its history. The technology stores transactions in a decentralized manner. Value exchange transactions are conducted between directly connected peers and verified through consensus using algorithms across the network.

Blockchain is the most well-known and used DLT where transactions are recorded with an immutable cryptographic signature called a hash. A Blockchain is a type of ledger in which value exchange transactions (in the form of cryptocurrencies, tokens, or information) are sequentially grouped into blocks. Each block contains a signature based on the exact content (data string) of that block. The next block also contains this signature, linking all previous blocks back to the first block. In the Blockchain, the transaction orders are kept as blocks, the order and content of the blocks are cryptographically sealed, and consensus algorithms provide trust between the nodes. Blockchain and block structures are shown in Figure 3. Integration (Bridge, Hub, and Sidechain) can provide the connection between the two Blockchain platforms.

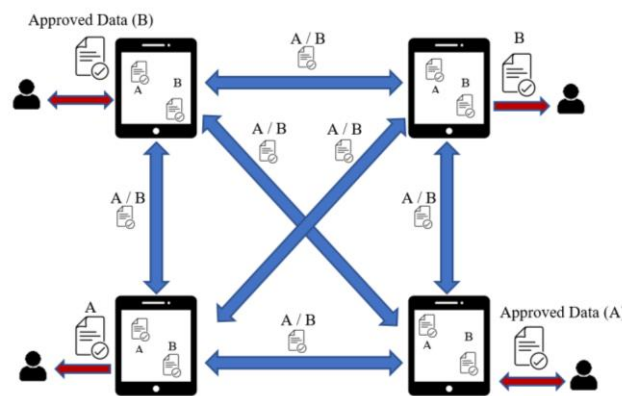


Figure 2. The Data Sharing in a DLT Structures

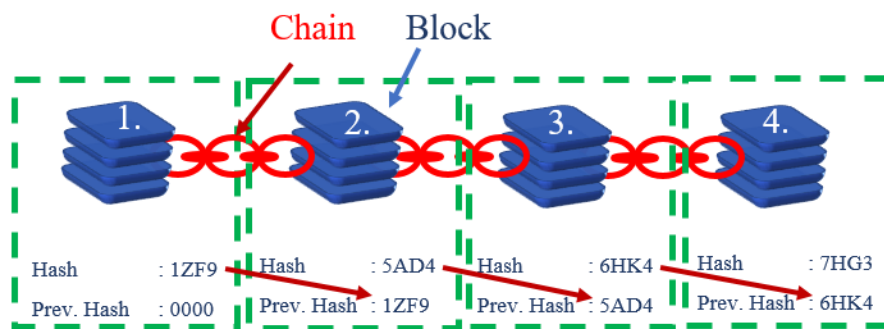


Figure 3. A DLT in Blockchain Structure and a Block Structure

Each block to be added to the Blockchain must be associated with the previous block's hash. More than 50% of Blockchain validators should permit the change of previous block data. Because previous blocks are encrypted and stored, replacing previous blocks takes a lot of time. This characteristic renders Blockchain highly resilient to adversarial attacks. Since the data in the Blockchain is not stored at a single point but is backed up at many points, it is possible to access the backup from the other in case one of the data is lost.

There are many types of Blockchain. The four most widely used Blockchain are listed below [15]:

- Private Blockchain
- Public Blockchain
- Permissioned Blockchain
- Consortium Blockchain

Private Blockchain is more centralized than Public Blockchain and Public Blockchain is more secure. Although it is not the owner of the Permissioned Blockchain, it is necessary to provide an insider invitation or predetermined special conditions to enter the chain. Consortium Blockchain is a Blockchain managed by a particular group or person [18, 19]. In order to obtain permission to upload or change data on the Blockchain, a smart contract must be signed with the Blockchain administration. Thus, unauthorized use is eliminated.

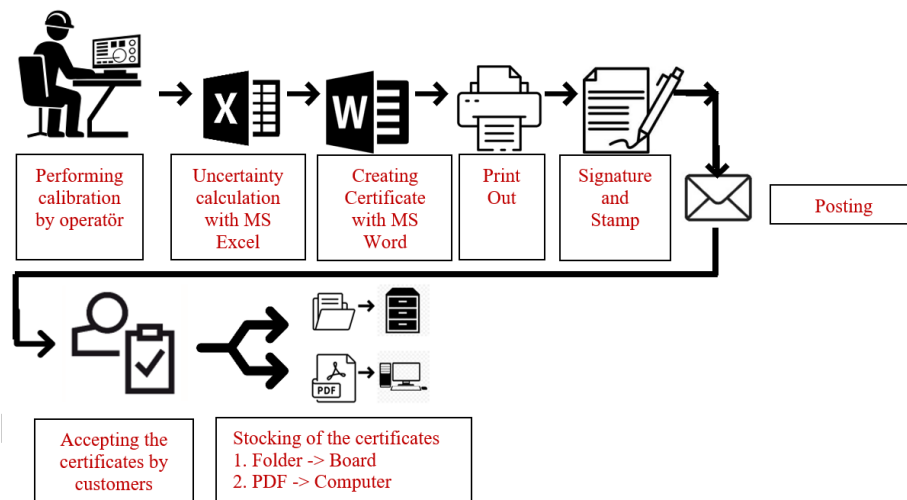
Blockchain applications can be seen to have used in the finance sector, digital identities, decentralized autonomous organization, Metaverse, defence industry, supply/logistics chain management companies, health services and additive manufacturing [12, 20, 21].

As for the general building blocks of the Blockchain, it can be thought that it will be used to store and access reliable DCC/DTR. Blockchain is one of the digital data sharing systems. It is possible that any group or network can create a new digital sharing system for their purpose. The features of the digital data sharing of the Blockchain can be applicable to the DCC/DTR sharing.

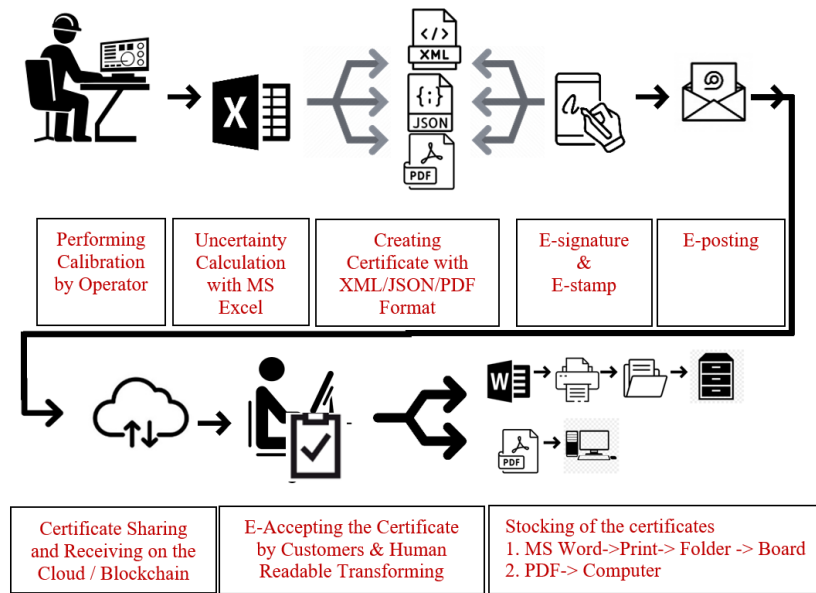
3. Sharing of DCCs/DTRs on Blockchain

The calibration certificate and testing report preparation processes are expressed in Figure 4. Figure 4 contains the currently CC/TR preparation flow and the new DCC/DTR preparation flow. When creating a CC/TR, it is usually an MS Excel application to calculate the uncertainty and MS Word or MS Excel applications are used to create a CC/TR. CC/TR is printed out and signed by an authorized person. Signed CC/TR is transferred to customers by post or hand in. Customers accepts the CC/TR and keep it in the relevant file by the customer (Figure 4.a). The preparation process for CC and TR predominantly relies on paper, leading to an inefficient depletion of global resources. The adoption of DCC and DTR will contribute to environmental sustainability. While DCC/DTR producing, operator usually uses MS Excel applications. Afterward, depending on the density of the data, it is converted into PDF, XML, or Jason formats and approved by Electronic-signature. Electronic-signatures provide non-repudiation, integrity, and authentication at the DCC/DTR application. It is transferred to the cloud data by the laboratory using Blockchain to be transmitted to the customer. The transferred information is received by the customer via Blockchain and transferred to their own systems. Afterward, they are kept in the relevant file by the customer (Figure 4.b).

Blockchain has started to be used in many metrological applications for calibration and testing processes [1, 22, 23]. Accredited calibration and testing laboratories, NMIs, and manufacturer calibration and testing laboratories need to prepare DCC/DTR.



(a)



(b)

Figure 4. (a) Currently calibration certificate creation and sharing process (b) Digital calibration certificate creation and sharing process

DCCs/DTRs must be created as both machine-readable and human-readable. The DCCs/DTRs created are desired to be easily accessible by accreditation bodies, auditors, certificate holders, and certificate producers. The new CC/TR must be created either by the laboratory that created the previous CC/TR or by another laboratory with the same competence. The new CC/TR, which an unauthorized laboratory will create, metrological means breaking the measuring device from the metrological traceability chain. CCs/TRs of measuring devices created in specific periods are considered the history of the measuring device and their accessibility by authorities when necessary, which is defined in the ISO/IEC 17025 standard.

The process from creating the CC/TR to its access must be valid within the DCC/DTR. Therefore, the Blockchain structure is suitable for DCC/DTR to be stored and accessed securely in the digital environment. Blockchain, where information that is important worldwide, such as financial data, is stored securely, is also suitable for storing DCC/DTRs, which are valuable for measurement devices.

Hardcopy CC/TR can be easily copied and modified using high-tech printers and copy systems. Although encrypted DCCs/DTRs will prevent their contents from being altered in any digital medium, they can be decrypted and changed. In the Blockchain, the original digital data is kept on several servers, and if one of its copies is altered, it will not be approved, and the change will not be allowed if it differs from the copies on other servers. Due to the decentralized nature of the Blockchain, DCC/DTR can only be modified or copied by the right people. The DCC/DTR stored on the Blockchain will also be protected against hacker attacks. With the addition of accreditation bodies, accreditation auditors, device owners, and CC/TR generators to the definition of persons or legal entities authorized to access data in the Blockchain, DCCs/DTRs can be easily and securely accessed. It will be possible to securely access DCC/DTR data stored in the Blockchain with e-signature.

Storing DCC/DTRs within the Blockchain network will eliminate the need for both the DCC/DTR creator and those who use or have access to them to use separate storage space. However, for the sustainability (eternity, et al.) of DCC/DTR storage on Blockchain, the Blockchain requires a small fee for storing each DCC/DTR [22]. Customers and DCC/DTR's creators need to be aware of this fee. This fee can be used for the sustainability of the Blockchain system and the new storage area of the DCC/DTR.

Consortium Blockchain is a suitable structure for the sharing of DCC/DTRs.

To implement these predictions, we can summarize the sharing process of DCC/DTR on Blockchain as follows.

- Creation of the certificate as DCC/DTR
- Connecting the certificate creator to the Blockchain with a smart contract
- Uploading DCC/DTR to Blockchain and spreading its copies to DLT
- Making DCC/DTR access influences
- Updating DCC/DTR when revision is required or creating new DCC/DTR when due
- For sustainability, the DCC/DTR creator pays for the Blockchain structure at specific periods.

The sharing of DCC/DTR on the Blockchain platform is shown in Figure 5. Red numbers are used to show the flow of DCC in Figure 6. In this sharing process, a DCC created by one user is sent to the Blockchain network (1). DCC is transferred to the process pool (2). DCC is controlled, and it is added as a new block (3). DCC was added as a new block (4), which is checked by the concessions mechanism and added to the Blockchain (5). Then, DCC is transferred to the Blockchain network for sharing (6). Through this process, a new DCC is approved and shared in the Blockchain network.

A visual depicting DCCs of a measuring device stored in the Blockchain is shown in Figure 6. Since the calibration certificate created in each calibration period is saved by linking to the hash of the previous calibration certificate, it will be easier to access the history of the device and the calibration data at the same time. It will also be possible to analyze data about the evolution of the device over time using DCC blocks.

Blockchain is ready and is currently used as a sharing system around the world. In this study, methods were proposed for the use of Blockchain as DCC/DTR in the field of metrology. However, the main organizations in this sector, such as BIPM, EURAMET, or the country's accreditation bodies, can create a similar data-sharing system specific to their own purposes.

The first studies to establish DCC in Türkiye are continuing at TÜBİTAK UME. Software for DCC production is being developed. The PTB-based XML file output of the software's possible DCC output types is given in Figure 7. Since the finalized XML files have not been created yet, their testing on the Blockchain network has not started.

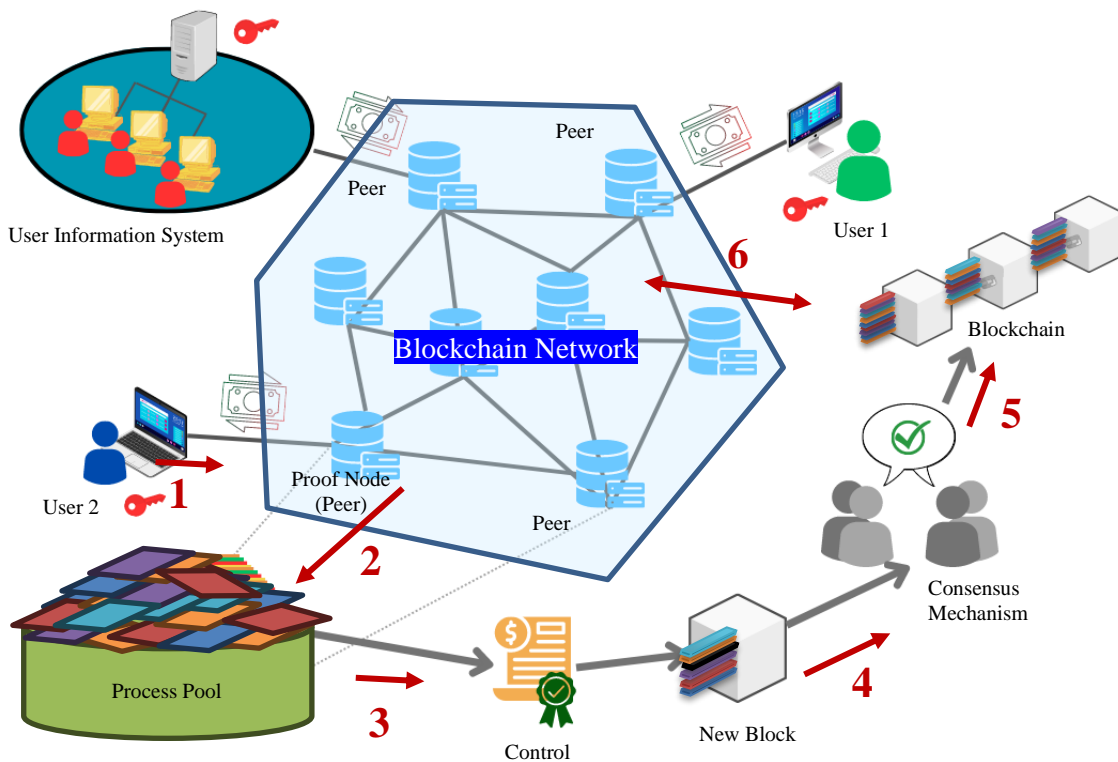


Figure 5. Sharing of DCC/DTR on the Blockchain Platform

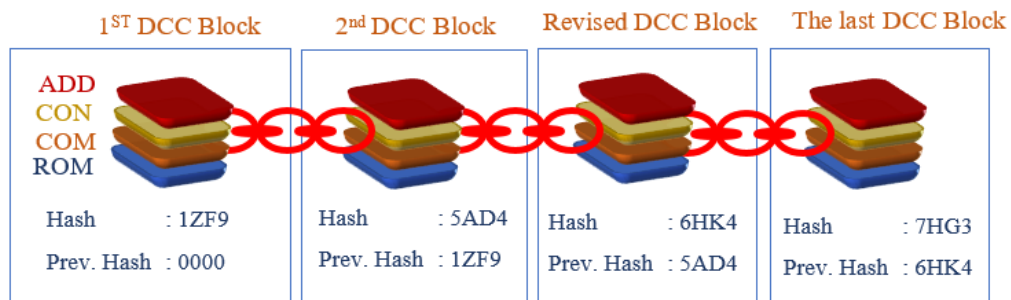


Figure 6. DCCs of a Measurement Device in the Blockchain


```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dcc="https://ptb.de/dcc" xmlns:si="https://ptb.de/si"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#" version="3.2.1"
  targetNamespace="https://ptb.de/dcc" elementFormDefault="qualified">
  <xs:import namespace="https://ptb.de/si"
    schemaLocation="https://ptb.de/si/v2.1.0/SI_Format.xsd"/>
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="https://www.ptb.de/dcc/d-sig/xmldsig-core-schema.xsd"/>
  <xs:annotation>
    ...
  </xs:annotation>
  <xs:element name="digitalCalibrationCertificate"
    type="dcc:digitalCalibrationCertificateType"/>
  <xs:complexType name="digitalCalibrationCertificateType">
    ...
  </xs:complexType>
  <xs:complexType name="administrativeDataType">
    ...
  </xs:complexType>
  <xs:complexType name="softwareListType">
    ...
  </xs:complexType>
  <xs:complexType name="softwareType">
    ...
  </xs:complexType>
  <xs:complexType name="refTypeDefinitionListType">
    ...
  </xs:complexType>
  <xs:complexType name="refTypeDefinitionType">
    ...
  </xs:complexType>
  <xs:complexType name="measuringEquipmentListType">
    ...
  </xs:complexType>
  <xs:complexType name="measuringEquipmentType">
    ...
  </xs:complexType>
  <xs:complexType name="measuringEquipmentQuantityListType">
    ...
  </xs:complexType>
  <xs:complexType name="primitiveQuantityType">
    ...
  </xs:complexType>
  ...
</xs:schema>

```

Figure 7. DCC Example in XML Format

5. Discussion and Conclusions

Blockchain, as it is known, has a structure that can be used for DCC/DTR, storage, and sharing purposes. The features of the Blockchain given below can be preferable criteria for why it is useable for sharing the DCC/DTR.

- Digital data on the Blockchain can be changed by only an authorized person
- Digital data on the Blockchain is globally accessible, ensuring transparency and availability.
- Digital data on the Blockchain can be used for the traceability of the validated and confirmed data
- In order to a part of the Blockchain digital data creator, a person / company should sign a contract and accept all the terms and conditions of the Blockchain structure.
- Blockchain's decentralized DLT structure ensures original data remains intact within the network, allowing correction if any digital data is mistakenly altered. .

DCCs/DTRs can be shared locally with storage space on the Blockchain to be created within the country or within an accreditation body. With the Blockchain structure to be created under the umbrella of BIPM or EURAMET, it will be possible to share DCCs/DTRs widely on a global scale within or outside the region. In this way, a worldwide structure will be created and DCCs/DTRs stored with Blockchain will be kept transparent, reliable, and encrypted, and they will be protected against adversarial attacks.

The necessity of charging a small fee per certificate for the sustainability of the Blockchain to be established for the purpose of storing DCC/DTR is also an important issue for customers. It is possible for metrology networks such as BIPM, EURAMET or the country's accreditation bodies to create a special digital data-sharing system for their own purposes. However Blockchain is ready and is currently used as a sharing system in the world. In this study, the necessary structure,

requirements, and process steps of DCC/DTR sharing with the Blockchain are discussed in detail. This study has been prepared to guide future studies on sharing DCC/DTRs within the Blockchain structure.

Blockchain is an option to share DCC/DTR, but the other emerging technologies can be another option for the DCC/DTR sharing. When all metrological partners such as NMIs, DIs, calibration laboratories or test laboratories prepare the DCCs/DTRs, an optimum solution will be found in their sharing in the future.

References

- [1] S. M. Wilson, "Blockchains and Legal Metrology: applications and possibilities," *OIML Bulletin Volume LXII*, Number 3, July 2021.
- [2] H. ElBouanani, C. Barakat, W. Dabbous, T. Turletti, "Passive delay measurement for fidelity monitoring of distributed network emulation," *Computer Communications*, Volume 195, pp. 40-48, 2022. ISSN 0140-3664, <https://doi.org/10.1016/j.comcom.2022.07.004>.
- [3] M. Vlaeyen, H. Haitjema, W. Dewulf, "Digital Twin of an Optical Measurement," *System Sensors* 21, 6638, 2021. <https://doi.org/10.3390/s21196638>.
- [4] O. Baer, C. Giusca, R. Kumme, A. Prato, J. Sander, D. Mirian, F. Hauschild, "Digital Twin concept of a force measuring device based on the finite element method," *ACTA IMEKO*, Volume 12, Number 1, pp 1 – 5, March 2023. ISSN: 2221-870X.
- [5] D. Schnürer, F. Hammelmüller, J.H. Holl, W. Kunze, "Offline digital twin synchronization using measurement data and machine learning methods," *Materials Today: Proceedings*, Volume 62, Part 5, pp. 2416-2420, 2022. ISSN 2214-7853, <https://doi.org/10.1016/j.matpr.2022.02.566>.
- [6] ISO/IEC 17025, 2017. General requirements for the competence of testing and calibration laboratories. [Online] <https://www.iso.org/standard/66912.html> [Accessed: 27-Februaury-2024].
- [7] Digital Calibration Certificate, [Online] <https://www.ptb.de/dcc/> [Accessed: 27-Februaury-2024].
- [8] Physikalisch-Technische Bundesanstalt (PTB). DCC Schema Version 3.1.0, DCC Version 3.1.0. [Online] <https://www.ptb.de/dcc/v3.1.0/dcc.xsd> [Accessed: 13-April-2022].
- [9] Release and Gitlab Repository of DCC Schema/DCC/xsd-dcc, GitLab. [Online] <https://gitlab.com/ptb/dcc/xsd-dcc> [Accessed: 19-April-2022].
- [10] S. Hackel, S. Schönhals, L. Doering, T. Engel, R. Baumfalk, "The Digital Calibration Certificate (DCC) for an End-to-End Digital Quality Infrastructure for Industry 4.0.," *MPDI Sci* 5, 11, 2023. <https://doi.org/10.3390/sci5010011>.
- [11] The BIPM Key Comparison Database, Classification of Services in Electricity and Magnetism. Version No 9 (dated 04 June 2020). [Online] <https://www.bipm.org/en/cipm-mra/cipm-mra-documents/service-categories> [Accessed: 27-Februaury-2024].
- [12] M. S. Nikoo, M.C. Kaya, M.L. Schwartz and H. Oguztuzun, "An MII-Aware SoA Editor for the Industrial Internet of Things," *II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*, Naples, Italy, pp. 213-218, 2019. doi: 10.1109/METRO4.2019.8792903.
- [13] The BIPM, Developing the SI Digital Framework as the anchor of trust for metrology in the digital era [Online] <https://www.bipm.org/en/digital-transformation> [Accessed 30 July 2024]
- [14] Z. Zheng, S. Xie, H.N. Dai, X. Chen, H. Wang, "An overview of Blockchain technology: architecture, consensus, and future Trends," *IEEE 6th International Congress on Big Data*, pp. 557-564, Las Angles, USA, December 2019.
- [15] M. Niranjnamurthy, B. N. Nithya, and S. Jagannatha, "Analysis of Blockchain technology: pros, cons and SWOT," *Cluster Comput* 22 (Suppl 6), 14743–14757, 2019. <https://doi.org/10.1007/s10586-018-2387-5>.
- [16] B. Lui, "Overview of the Basic Principles of Blockchain," *2021 International Conference on Intelligent Computing, Automation and Applications (ICAA)*, Nanjing, China, pp. 588-593, 2021. doi: 10.1109/ICAA53760.2021.00108.
- [17] H. Natarajan, S. Krause, H. Gradstein, "Distributed Ledger Technology and Blockchain," *FinTech Note*;No. 1.© World Bank, Washington, DC, USA, 2017. <http://hdl.handle.net/10986/29053>
- [18] B. C. Ghosh, T. Bhartia, S. K. Addya and S. Chakraborty, "Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing," *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, Vancouver, BC, Canada, pp. 1-10, 2021. doi: 10.1109/INFOCOM42981.2021.9488683.
- [19] N. Elisa, L. Yang, H. Li, F. Chao, and N. Naik, "Consortium blockchain for security and privacy-preserving in e-government systems," *In Proceedings of the 19th International Conference on Electronic Business*, pp. 99- 107, Newcastle upon Tyne, UK, 8-12 December 2019.
- [20] D. Özdemir, T. Çobanoğlu, T.F. Cihan, S. Dörterler, R. Uyar, "Eğitimde Blok Zincir Uygulamaları," *14. International Congress of Educational Research*, Çanakkale, Türkiye, pp. 222-233, 27 - 30 October 2021.
- [21] A. Ekin, and D. Ünay, "Blockchain applications in healthcare," *26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, Turkey, pp. 1-4, 2018. doi: 10.1109/SIU.2018.8404275.
- [22] N. Takegawa, N. Furuichi, "Traceability Management System Using Blockchain Technology and Cost Estimation in the Metrology Field," *MPDI Sensors*, 23, 1673i 2023. <https://doi.org/10.3390/s23031673>.
- [23] A. Softic, U. N. Zaimovic, and S. Lemes, "Blockchain-based Metrological Traceability," *Proceedings of the 32nd DAAAM International Symposium*, Vienna, Austria, pp. 0522-0526, 2021. B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-33-4, ISSN 1726-9679. doi: 10.2507/32nd.daaam.proceedings.

Author(s) Contributions

Authors contribution for this article is given in Copyright Agreement and Acknowledgement of Authorship Form.

Acknowledgments

We would like to thank the researchers working at TÜBİTAK BİLGEM Blockchain Technologies Department for the information and documents they provided.

Conflict of Interest Notice

There is no conflict of interest regarding the publication of this paper.

Ethical Approval

There is no any Ethical Approval for this study.

Availability of data and material

Not applicable

Plagiarism Statement

This article has been scanned by iThenticate™.