Sakarya University

Journal of Computer and Information Sciences

e-ISSN 2636-8129

VOLUME 7 ISSUE 1

APRIL 2024

VOLUME: 7 ISSUE: 1
E-ISSN 2636-8129

APRIL 2024
http://saucis.sakarya.edu.tr/tr/

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES

SAKARYA
ÜNİVERSİTESİ

The Owner on Behalf of Sakarya University

Editor in Chief

Assistant Editors

Editorial Board

Editorial Assistants - Secretary

Language Editors

Layout Editor

Indexing

Contents

Research Article

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 1-10
DOI: 10.35377/saucis...1359146
SAKARYA UNIVERSITY
RESEARCH ARTICLE

Bacterial Disease Detection of Cherry Plant Using Deep Features

Emrah Dönmez[1], Yavuz Ünal[2], Hatice Kayhan[3]

[1] Department of Software Engineering, Bandırma Onyedi Eylül University, Balıkesir, Türkiye
[2] Department of Computer Engineering, Amasya University, Amasya, Türkiye
[3] Technology and Innovation Management, Amasya University, Amasya, Türkiye

Corresponding author:
Yavuz Ünal, Department of Computer Engineering, Amasya University
yavuz.unal@amasya.edu.tr

Article History:
Received: 12.09.2023
Accepted: 10.01.2024
Published Online: 10.01.2024

ABSTRACT
Although the cherry plant is widely grown in the world and Turkey, it is a fruit tree that is difficult to grow and maintain. It can be exposed to various pesticide diseases, especially during fruiting. Today, approaches based on expert reviews and analyses are used for the identification of these diseases. In addition, cherry producers are trying to detect diseases with their knowledge based on experience. Computer-aided agricultural analysis systems are also being developed depending on the rapid developments in technology. These systems help to monitor all processes from planting, cultivation, and harvesting of agricultural products and to make decisions to grow the products healthily. One of the most important issues to be detected and monitored with these systems is plant diseases. The features of the cherry plant disease will be determined by using a pre-trained convolutional neural network (CNN) model which is DarkNet-19, within the scope of this study. These machine learning-based features have been used for the detection of bacteria-based diseases commonly seen on the leaves of cherry plants. The acquired features are classified with Linear Discriminant Analysis, K-Nearest Neighbor, and Support Vector Machine classifiers to solve the multi-class problem including diseased (less and very) and healthy plants. The experimental results show that a success rate of 88.1% was obtained in the detection of the disease.

Keywords: Cherry plant, Classification, Convolutional neural networks, Machine learning, Plant diseases

1. Introduction

Early identification of plant diseases is among the important factors affecting plant yield. Plant diseases can affect an entire plant as well as regionally. Plants can be infected with different diseases according to their species, and these diseases can be carried by bacteria, fungi, pesticides, harmful insects, and other plants. Agricultural experts usually carry out disease detection through observation. When detecting the disease of the plant, experts examine the parts of the plant that contain leaves, roots, stems, and seeds and then they diagnose the disease. Diseases can affect plant yield as well as cause partial or permanent damage to plants. For this reason, accurate detection is a critical issue for correct intervention. On the other hand, computer-aided applications are also being developed to conduct examinations as well as experts[1].

In recent years, it is seen that there has been a remarkable development in computer-aided agricultural practices. With the use of these applications, it is aimed at carrying out agricultural operations and processes more efficiently. Computer-aided applications [2], [3] are being developed in many agricultural applications from seed planting to land irrigation, from plant spraying (herbicide) to yield detection, and from disease detection to harvesting. Among these agricultural processes, the development of these applications for disease detection, which plays an important role in plant productivity, is a very important pillar. Computer-aided disease detection systems are developed based on data labeled by experts in the detection of diseases. In these systems, mainly machine learning-based approaches are used [4].

Machine learning is a method for modeling learning, storing in memory, and updating models in the nervous systems of biological creatures in a computer environment. It is possible to solve many classification and regression problems based on machine learning. From service to the industry, from transportation to education, from health to agriculture, etc. It is seen that machine learning systems have been developed in many sectors. These systems can be used both as a decision support system and as a direct decision-makers with the high decision-making skills they provide. Today, machine learning applications are being developed to increase productivity in agricultural products and to intervene in diseases and pests early. The main resource in the development of these applications is based on the decisions of agricultural experts. In recent years,

traditional machine learning approaches have been replaced by convolutional neural network models, also known as deep learning methods [5].

Cherry is one of the most intensively grown plants in Turkey. It is also among the fruits with high export value [6]. This study aims to detect the bacterial disease of the cherry plant by using deep features obtained from leaf images with machine learning approaches. With early detection of the disease, early intervention can be achieved and production efficiency can be improved.

The scope of this study is based on the detection of plant diseases with computer-aided agricultural analysis systems. As a prominent area in agricultural analysis, disease detection in plants is usually done through the observation of an agronomist. This disease detection task can be performed faster and more efficiently with computer-aided systems using machine learning methods.

In the second part of the study, related literature studies are given. In the third section, the dataset, deep features, and classifier methods are explained. In the fifth section experiments and results, the experimental configuration is given and the experimental results are examined comparatively. In the last section, the results are evaluated and the aimed future studies are mentioned.

1.1. Related works

There are significant studies in the literature on plant disease detection. While some of these studies are based on the analysis of plant diseases such as root and stem parts of the plant, some of them are based on the analysis of the symptoms in the flower part. On the other hand, the detection of diseased parts based on the analysis of plant leaves is among the studies in the literature. The most critical issue in computer-aided detection systems is to extract the features that express disease symptoms on the components of the plant such as root, stem, flower, and leaf [7]. These attributes are known as features in analysis systems. In the literature, it is mentioned that the features are obtained by image processing methods, and they form an input to the classifier methods. This approach is known as the traditional classification approach. Recently, classification has been made using deep features that can be obtained directly from convolutional neural networks. This method provides a more modern classification approach, which is still commonly used today.

Zhang et al. [8] classified the powdery mildew disease on cherry leaves with GoogleNet, SVM, KNN, and BP neural network, which are among the CNN models. They reached 99.6% classification accuracy of CNN with their study on a data set consisting of 1200 images.

Ilic et al. [9] used different math-based methods for processing data and disease infection prediction. Six important weather parameters and a variable implying the month of the year were chosen as predictive variables. The forecast situation corresponds to the two significant infections of cherry fruit, “Monilinia laxa” and “Coccomyces hiemalis”. The data sets used in the research include data for eight years. In the study, it was stated that the prediction accuracy was 95.8%.

Atilla et al. [10] proposed the EfficientNet CNN model for the identification of plant leaf diseases and compared it with other deep-learning models. These models have been trained through transfer learning with original and augmented datasets from PilantVillage with 55,448 and 61,486 images, respectively. According to the results of the experiment, with the B4 and B5 models, which are variations of the proposed CNN model, the accuracy was 99.97% and 99.91% (for the augmented and original datasets), respectively.

Joshi et al. [11] propose an automatic viral infection identification method by using deep learning for Vigna mungo, a legume plant grown largely in the Indian subcontinent. They state that the pattern is very random throughout the leaf structure due to viral infection, and therefore it is difficult to perform an automated disease identification method in real-time. They state that with the proposed method, the classification accuracy was 97.4%.

Luna et al. [12] claim to have developed an innovative approach to the disease detection of tomato plants. Utilizing a dataset of 4,923 tomato plant leaf images, they trained a deep convolutional neural network to recognize three diseases: Target Spot, Leaf Miner, and Phoma Rot. The features obtained from the network trained with transfer learning were used to determine which tomato diseases are and it was stated that the model provided 95.75% accuracy.

Özcan and Dönmez [13] proposed a bacteria-based disease detection method by performing deep feature extraction on pepper plant leaf images. In their studies, 1478 healthy and 997 bacterial diseased leaf images were used as the PilantVillage data set, a total of 2475. DarkNet-19 CNN model is used for deep feature extraction. They tested the features in four different classifiers to use both by default and size reduction with PCA. Classification success was expressed as 98.8%.

Jiang et al. [14] proposed a new model for the detection of apple leaf disease using deep CNNs, by presenting the GoogLeNet Inception and Rainbow coupling models. They trained the model to identify five common apple leaf diseases by using a dataset of 26,377 diseased apple leaves. Experimental results show that it achieves 78.80% disease detection performance.

Islam et al. [15] proposed a method that integrates image analysis and machine learning to allow the identification of diseases. In their method, they automatically classify diseases in potatoes from a public plant image database called the "PlantVillage".

They said that they performed disease classification of more than 300 images with 95% accuracy with the support vector machine classifier.

Ferentinos et al. [16] developed CNNs to implement the detection of plant disease and diagnosis through deep learning using simple leaf images of healthy and diseased plants. Models were trained to utilize an open database of 87,848 images. This database includes 25 different plants in 58 different [plant, disease] combination classes, including healthy plants. Several models have been trained and the best performance achieved a 99.53% success rate in identifying the relevant combination.

When studies in the literature are examined, the CNN model itself is generally used directly as a classifier. In this study, the classification task was performed with KNN (K-Nearest Neighbor), LDA (Linear Discriminant Analysis), and SVM (Support Vector Machine) methods using the DarkNet-19 CNN model as a feature extractor.

2. Methodology

2.1. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of artificial neural network (ANN) utilized in visual signal recognition and is specially designed to process pixel data. CNN enables the development of powerful image analysis, and artificial intelligence applications that use deep learning to implement both descriptive and generative tasks, often including image and video recognition in conjunction with recommendation systems and natural language processing (NLP). Convolutional neural network models have made the process of obtaining features and classifying these features with the feature extraction methods on the image applied in traditional methods more efficient. Convolutional neural networks determine features using arithmetic operations such as convolution etc. within a multi-layered architecture without the need for a third-party method. A basic CNN consists of input, output, and hidden layers. The hidden layer includes multiple convolutions, pool, fully connected, and normalization layers. The obtained features can be used for training the SoftMax classifier method which is placed as the last layer of the network or directly for training another classifier method. The features obtained with the use of these networks also increase the level of representation in the extraction of distinctive features of the data set.

2.2. Dataset

In the dataset containing a total of 1906 different cherry leaf images, the data were labeled into three groups by the agronomist. This dataset consisting of cherry leaves is included in the Plant Village dataset[17]. These labels are “healthy”, “less diseased” and “very diseased”. The number of healthy data is 854, those with less diseased is 537, and those with very diseased is 515. Assistance has been sought from an agronomist in labeling images of less and more diseased leaves relevant to the data. An example section from the dataset is given in Figure 1 below.

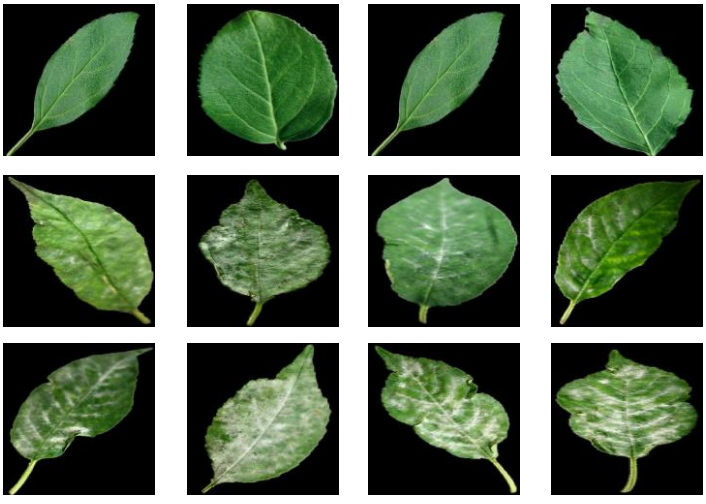


Figure 1 Sample Leaf Images From The Data-set
(top: diseased, middle: slightly diseased, bottom: very diseased)

There are two classes in the original PlantVillage dataset. These classes are cherry powdery mildew and cherry healthy. Among these classes, the cherry powdery mildew class was divided into two classes by the expert: less diseased and very diseased. It consists of three classes in total, including the healthy class. The study was carried out on these three classes.

2.3. Deep Features

In traditional methods, vector quantities (a group of numerical values that can represent vertices, color, texture, etc. in an image) are calculated on the data at hand, which best describes and represents the data. In this calculation process, third-party SURF, SIFT, FAST, etc. methods are used for image data. Deep features are acquired from fully connected layers of CNN models. These features correspond to the vector quantities that best represent the analyzed data. The features show different convolutions etc. in each hierarchical layer of the input data of the deep learning network. They are processed to provide consistent values to the output layers. Within the scope of the study, the DarkNet-19 convolutional neural network model [18] was used.

The DarkNet-19 network is a highly efficient model in resource utilization and memory management. On the other hand, it is suitable for real-time applications. The model also achieves remarkable levels of accuracy despite using a single forward pass structure [18].

Figure 2 below shows the DarkNet-19 architecture.

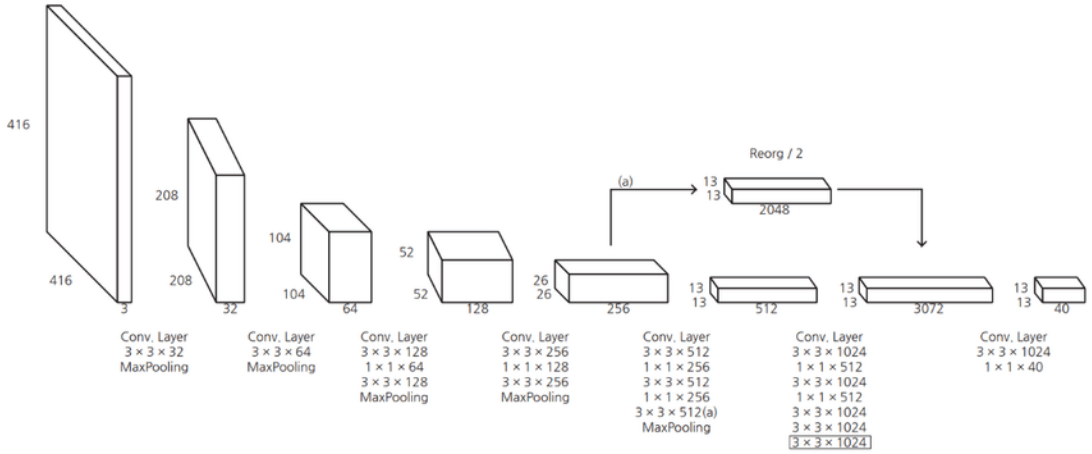


Figure 2 Darknet-19 (YoloV2) Architecture [19]

2.4. Classification

The classification algorithms used in detection processes in the mentioned applications are trained with labeled data. During the training phase, vector (feature) parameters with certain weights are calculated from the data in each data processing iteration in these classifiers and the previous values are updated. Then, the data features of the new incoming data are extracted from the system. The trained classifier algorithm determines the relevant class information of this new data by comparing its current parameters with the feature parameters of the new data. The LDA, KNN, and SVM classifier methods were used within the scope of the study.

The K-nearest neighbor (KNN) method [20] is one of the most basic supervised and non-parametric methods used for classification and regression. It is suitable for situations where little or no prior knowledge of data distribution is available. The nearest neighbor method is used to determine which of the existing classes enters new data into the environment through training vectors.

The Linear Discriminant Analysis (LDA) model estimates probabilities in the classifier [21]. They perform prediction to determine the class of a new input based on the probability. The class that has the highest probability is specified as the output class, and then LDA makes an estimation. The estimation is performed using Bayes' Theorem, which estimates the output class probability. They also utilize the probability of each class and the probability of the data for each class.

Support Vector Machine (SVM) is a two-class classifier [22]. The most common technique for multiclass classification with SVMs is to construct one-versus-all classifiers (often called 'one-against-all' or OVA classification) and choose the class that classifies the test data with the largest margin Another strategy is to create a one-to-one set of classifiers and choose the class chosen by the most classifiers.

Performance metrics used to measure model performances are given below (1), (2) and (3). TP, TN, FP and FN correspond to ‘True Positives’, ‘True Negatives’, ‘False Positives’ and ‘False Negatives’, respectively.

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

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Choosing the most important metric in classification operations depends on the specific goals and requirements of the application. Generally, accuracy is good if there is a balance between classes in the dataset. Precision becomes important in situations where the cost of false positives is high. Recall is important in cases where the cost of false negatives is high.

3. Results and Discussion

3.1. Experiment Configuration

The experiments were carried out on a computer with a 6th generation i7 2.6 GHz processor, 8GB RAM, 2GB GTX950 GPU, SSD HDD, and Windows 10 Pro configuration. The DarkNet-19 pre-trained CNN model was used to extract deep features. MATLAB 2020b version was used for feature acquiring and programming the classification process. No additional settings and optimizations have been made for the application. Similarly, there is direct use of the used data set without any pre-processing and improvement.

Features were obtained from the Fully Connected Layer (FCL) of the DarkNet-19 CNN model. A total of 1000 features were acquired from the FCL. Obtained features were analyzed according to 10-fold cross-validation as input to the training and testing processes of the classifier. In the pre-trained DarkNet-19 CNN model, the SoftMax classifier in the classifier layer was removed and replaced with LDA, KNN, and SVM classifiers, Figure 1.

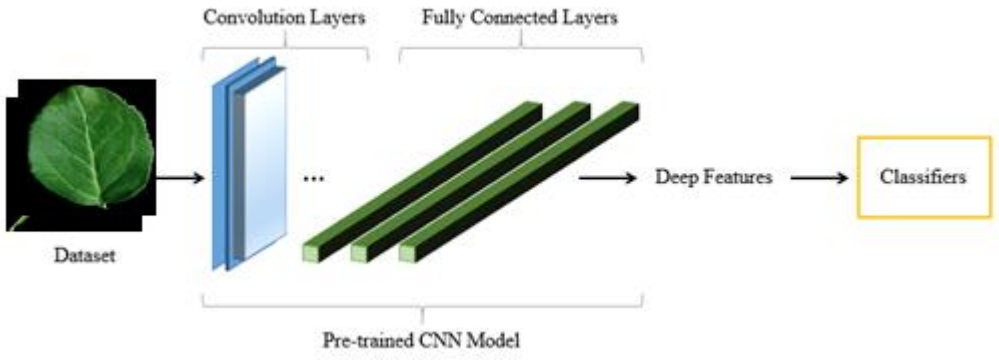


Figure 1 Representative CNN architecture model

LDA, KNN, and SVM classifiers were trained with features and tested with cross-validation. In the LDA classifier, the covariance structure is used as fully dependent. In the KNN classifier, the neighborhood value of K is chosen as 5, while the distance metric 'Euclidean' and the distance weight are chosen equally. This configuration is named FKNN. While the number of neighbors was chosen as 10 as the second configuration (WKNN) for the KNN classifier, the distance metric was again chosen as 'Euclidean' and the square inversion technique was used as the distance weight. In the SVM classifier, linear (LSVM), cubic (CSVM), and quadratic (QSVM) functions are used as kernel functions.

3.2 Experiment Results

In the first stage of the experiments, the test data and network performance were directly analyzed to observe the direct performance of the DarkNet-19 CNN model. The hyperparameters of the model are given in Table 2 below.

Table 2 DarkNet-19 Hyperparameter Settings

Common Options	**Value**
Batch size	64
Max epochs	30
Initial learn rate	1e-4
Momentum	0.8
L2 Regularization	1e-6
Shuffle	Never
Execution environment	GPU

Table 3 below presents the performance values directly resulting from the use of DarkNet-19.

Table 3 DarkNet-19 Performance Metrics

Model	**TP**	**FN**	**FP**	**TN**	**Accuracy (%)**	**Precision**	**Recall**
DarkNet-19	724	130	141	881	85.9%	0.85	0.85

According to the experimental results, complexity matrices for the “healthy”, “little diseased” and “highly diseased” class labels were obtained in Table 4, Table 5, and Table 6 below. With separate complex matrices for each class in the dataset, it is aimed to give results by the one-versus-all working style in multi-class problems. Thus, the overall performance results obtained in each class were revealed.

Table 4 Confusion Matrix (Healthy)

	TP	**FN**	**FP**	**TN**
LDA	829	25	58	994
WKNN	841	13	43	1009
FKNN	843	11	24	1028
QSVM	839	15	6	1046
CSVM	843	11	3	1049
LSVM	835	19	2	1050

Very high performance has been achieved in the classification of healthy leaves. The main factor in this high performance is the high discrimination rate of the features obtained from the CNN model for the distinction between healthy and diseased leaves. Performance metrics that characterize the overall performance status for the healthy class are given in Table below.

Table 5 Classification Performance Metrics (Healthy)

	Accuracy (%)	**Precision**	**Recall**
LDA	95.6%	0.93	0.97
WKNN	97.1%	0,95	0.98
FKNN	98,2%	0,97	**0,99**
QSVM	98,9%	0,99	0,98
CSVM	**99,3%**	**1,00**	**0,99**
LSVM	98,9%	**1,00**	0,98

The complexity matrix values for less diseased cherry leaves are given in Table below. Similarly, the complexity matrix is constructed as one class vs. all (other remainders). The discrimination between the level of disease is a hard task compared to discrimination of healthy vs. diseased leaves.

Table 6 Confusion Matrix (Less Diseased)

	TP	**FN**	**FP**	**TN**
LDA	345	192	151	1218
WKNN	389	148	140	1229
FKNN	389	148	144	1225
QSVM	413	124	115	1254
CSVM	419	118	123	1246
LSVM	424	113	109	1260

The performance metrics obtained depending on the less diseased are given in **Hata! Başvuru kaynağı bulunamadı.** below. The overall results demonstrate promising performance values.

Table 7 Classification Performance Metrics (Less Diseased)

	Accuracy (%)	**Precision**	**Recall**
LDA	82,0%	0,70	0,64
WKNN	84,9%	0,74	0,72
FKNN	84,7%	0,73	0,72
QSVM	87,5%	0,78	0,77
CSVM	87,4%	0,77	0,78
LSVM	**88,4%**	**0,80**	**0,79**

The complexity matrix values for very diseased cherry leaves are given in **Hata! Başvuru kaynağı bulunamadı.** below. Similarly, the complexity matrix is constructed as one class vs. other remainders. The main hardship for discrimination of 'less' and 'very' is weakening in distinctive features.

Table 8 Confusion Matrix (Very Diseased)

	TP	**FN**	**FP**	**TN**
LDA	354	161	169	1222
WKNN	367	148	126	1265
FKNN	366	149	140	1251
QSVM	411	104	122	1046
CSVM	402	113	116	1275
LSVM	421	94	115	1050

The performance metrics obtained depending on the very diseased class are given in **Hata! Başvuru kaynağı bulunamadı.** below. The average accuracy performance has been determined above 85%.

Table 9 Classification Performance Metrics (Healthy)

	Accuracy (%)	**Precision**	**Recall**
LDA	82,0%	0,70	0,64
WKNN	84,9%	0,74	0,72
FKNN	84,7%	0,73	0,72
QSVM	87,5%	0,78	0,77
CSVM	87,4%	0,77	0,78
LSVM	**88,4%**	**0,80**	**0,79**

Satisfactory classification performance was achieved with the features extracted from healthy and diseased cherry leaf images with the DarkNet-19 CNN model. Particularly, the performance in recognizing the healthy plant leaf was provided by the SVM classifier, which uses the cubic function as the kernel, with an accuracy rate of 99.3%. While the overall success rate in the detection of less diseased leaves was 88.4% with linear SVM, the performance in the detection of very diseased leaves was 88.0% with the SVM using a cubic kernel. No image preprocessing, parameter or function optimizations were made in the experiments. Final multi-class performances (Acc), classification costs (Cost), prediction speed - OBS/s (PS), and training time - s (TT) parameters are given in **Hata! Başvuru kaynağı bulunamadı.**.

Table 10 Overall Multi-class Performance Parameters (for 3 classes)

	Acc.	**Cost**	**PS**	**TT**
LDA	80,2%	378	2000	17,186
WKNN	83,8%	309	350	25,643
FKNN	83,8%	308	350	25,335
QSVM	87,3%	243	1700	32,169
CSVM	87,3%	242	1700	26,055
LSVM	**88,1%**	226	3200	14,508

In the analysis, 85.9% accuracy was achieved by classification with darknet-19. Using the Darknet-19 method as a feature extractor and then classifying it with machine learning algorithms increased the system's performance. In the analysis, 88.1% was reached.
ROC and AUC graphs of the models are given in Figure 4.
A summary of the literature review is given in Table 11.

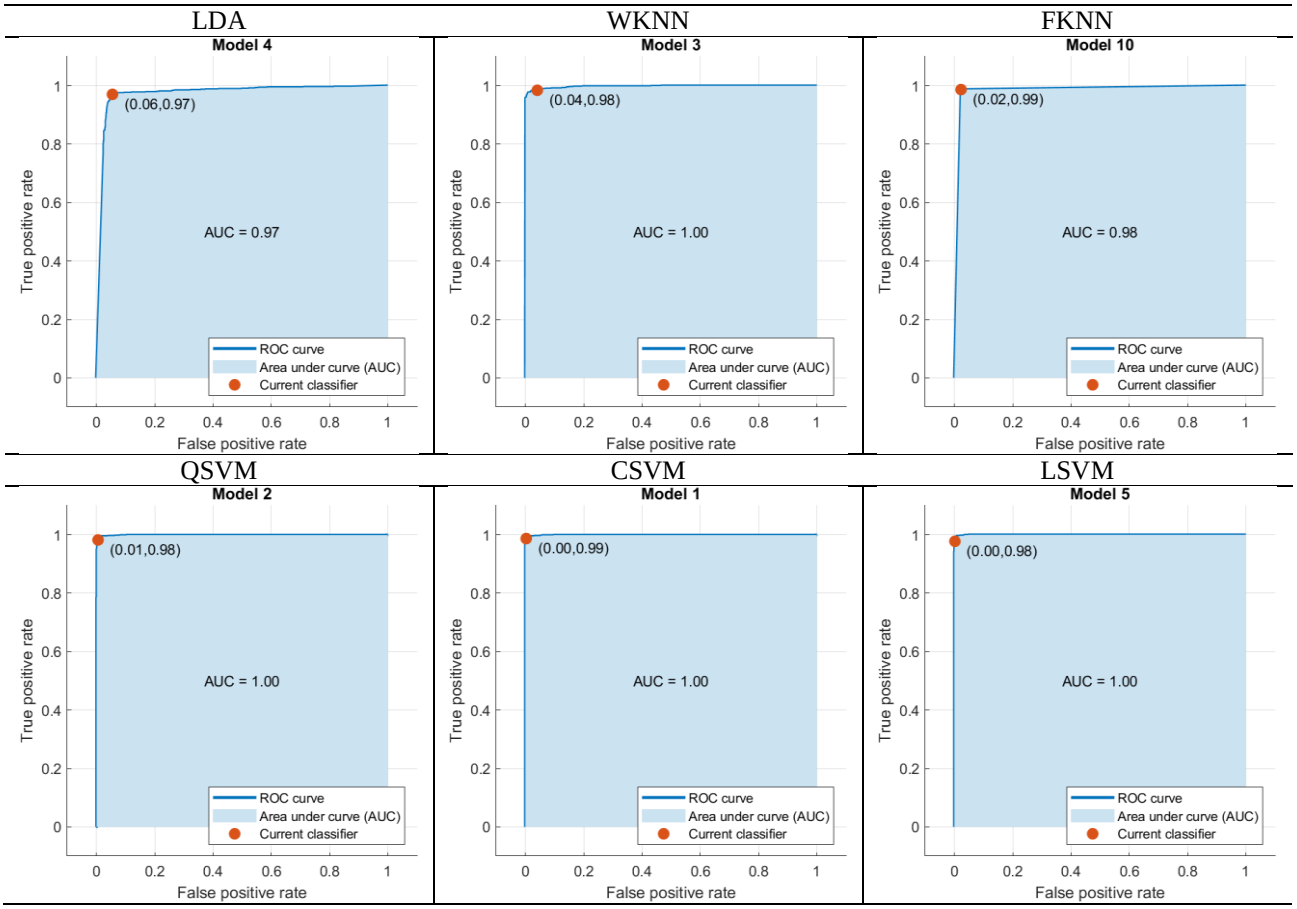


Figure 4 ROC and AUC Graphs of The Models

Table 11 Summary of The Literature Review

Description of The Problem	Data Used	Class	Accuracy	Methods	References
Cherry leaf disease infected by pedosphere pannosa	1200	2	99.6%	GoogleNet, SVM, KNN, BP	[8]
Early cherry fruit pathogen disease detection	1224	2	95.8%	LDA, QDA, Compact classification tree	[9]
Cherry leaf	55.448	2	98.42%	EfficientNet, AlexNet, VGG16, Resnet50, InceptionV3	[10]
Vigna mungo plant	433	3	97.4%	VirLeafNet	[11]
Tomato plant leaf disease	4923	2	95.75%	F-RCNN	[12]
Bacterial disease detection for pepper plant	2475	2	98.8%	Darknet-19, Naïve Bayes, K-NN, SVM	[13]
Real-Time Detection of apple leaf diseases	26.377	2	78.80%	GoogleNet, Inception Rainbow coupling	[14]
Potato diseases	300	3	95%	SVM	[15]
Plant disease detection	87.848	2	99.53%	AlexNet, GoogleNet, VGG	[16]
Cherry leaf bacterial diseases	1906	3	88.1%	DarkNet-19, SVM, KNN, LDA	Our Study

When referring to Table 11, features have been extracted in our study using Darknet-19, and classification has been performed with SVM, KNN, and LDA. The highest classification accuracy of %88.1 has been achieved.

4. Conclusion

In this study, cherry plant leaf images were utilized as input to the pre-trained DarkNet-19 CNN model to identify diseased (less or very) and healthy plant leaves and the features of the images were obtained from the FCL of the model. These obtained features are given to LDA, KNN, and SVM classifiers instead of the SoftMax layer, which is the last layer of the model. According to the experiments, a success rate of 88,1% was obtained in the detection of the disease. With this developed system, it is suggested that the detection of bacterial diseases in the cherry plant, which is grown as a fruit with high added value both in our country and in the world, should be determined by machine learning approaches. It is foreseen that this determination will be made quickly and early, and this will contribute to the increase of productivity in cherry cultivation. In this period when digitalization is becoming more and more widespread, computer-aided applications in agriculture will be used more widely with this and similar studies. In future studies, it is planned to recognize more than one type of plant disease and to determine the amount of disease more clearly. A system that recommends remedial actions against diseases is aimed to develop in the next stage. In future studies on this data set, the diseased area can be detected using YOLO or R-CNN, object detection algorithms.

References

[1] A. Jain, S. Sarsaiya, Q. Wu, Y. Lu, and J. Shi, ‘A review of plant leaf fungal diseases and its environment speciation’, *Bioengineered*, vol. 10, no. 1, pp. 409–424, Jan. 2019, doi: 10.1080/21655979.2019.1649520.

[2] E. Dönmez, ‘Classification of Haploid and Diploid Maize Seeds based on Pre-Trained Convolutional Neural Networks’, *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, vol. 16, no. 3, pp. 323–331, Sep. 2020, doi: 10.18466/cbayarfbe.742889.

[3] E. Donmez, ‘Discrimination of Haploid and Diploid Maize Seeds Based on Deep Features’, in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, Gaziantep, Turkey: IEEE, Oct. 2020, pp. 1–4. doi: 10.1109/SIU49456.2020.9302142.

[4] C. Jackulin and S. Murugavalli, ‘A comprehensive review on detection of plant disease using machine learning and deep learning approaches’, *Measurement: Sensors*, vol. 24, p. 100441, Dec. 2022, doi: 10.1016/j.measen.2022.100441.

[5] M. M. Taye, ‘Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions’, *Computers*, vol. 12, no. 5, p. 91, Apr. 2023, doi: 10.3390/computers12050091.

[6] A. Güneyli, C. E. Onursal, T. Seçmen, S. Sevinç Üzümcü, M. A. Koyuncu, and D. Erbaş, ‘The Use of Controlled Atmosphere Box in Sweet Cherry Storage’, *Horticultural Studies*, vol. 39, no. 2, pp. 33–40, Jun. 2022, doi: 10.16882/hortis.1119743.

[7] S. M. Hassan *et al.*, ‘A survey on different plant diseases detection using machine learning techniques’, *Electronics*, vol. 11, no. 17, p. 2641, Aug. 2022, doi: 10.3390/electronics11172641.

[8] K. Zhang, L. Zhang, and Q. Wu, ‘Identification of Cherry Leaf disease infected by Podosphaera Pannosa via convolutional neural network’:, *International Journal of Agricultural and Environmental Information Systems*, vol. 10, no. 2, pp. 98–110, Apr. 2019, doi: 10.4018/IJAEIS.2019040105.

[9] M. Ilic, S. Ilic, S. Jovic, and S. Panic, ‘Early cherry fruit pathogen disease detection based on data mining prediction’, *Computers and Electronics in Agriculture*, vol. 150, pp. 418–425, Jul. 2018, doi: 10.1016/j.compag.2018.05.008.

[10] Ü. Atila, M. Uçar, K. Akyol, and E. Uçar, ‘Plant leaf disease classification using EfficientNet deep learning model’, *Ecological Informatics*, vol. 61, p. 101182, Mar. 2021, doi: 10.1016/j.ecoinf.2020.101182.

[11] R. C. Joshi, M. Kaushik, M. K. Dutta, A. Srivastava, and N. Choudhary, ‘VirLeafNet: Automatic analysis and viral disease diagnosis using deep-learning in Vigna mungo plant’, *Ecological Informatics*, vol. 61, p. 101197, Mar. 2021, doi: 10.1016/j.ecoinf.2020.101197.

[12] R. G. De Luna, E. P. Dadios, and A. A. Bandala, ‘Automated image capturing system for deep learning-based Tomato Plant Leaf disease detection and recognition’, in *TENCON 2018 - 2018 IEEE Region 10 Conference*, Jeju, Korea (South): IEEE, Oct. 2018, pp. 1414–1419. doi: 10.1109/TENCON.2018.8650088.

[13] A. Özcan and E. Dönmez, ‘Bacterial disease detection for Pepper plant by utilizing deep features acquired from DarkNet-19 CNN Model’, *DÜMF Mühendislik Dergisi*, pp. 573–579, Sep. 2021, doi: 10.24012/dumf.1001901.

[14] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, ‘Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks’, *IEEE Access*, vol. 7, pp. 59069–59080, 2019, doi: 10.1109/ACCESS.2019.2914929.

[15] M. Islam, Anh Dinh, K. Wahid, and P. Bhowmik, ‘Detection of potato diseases using image segmentation and multiclass support vector machine’, in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Windsor, ON: IEEE, Apr. 2017, pp. 1–4. doi: 10.1109/CCECE.2017.7946594.

[16] K. P. Ferentinos, ‘Deep learning models for plant disease detection and diagnosis’, *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, Feb. 2018, doi: 10.1016/j.compag.2018.01.009.

[17] A. P. J, ‘Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network’. Mendeley,

Apr. 18, 2019. doi: 10.17632/TYWBTSJRJV.1.
[18] J. Redmon and A. Farhadi, 'YOLO9000: Better, faster, stronger', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 6517–6525. doi: 10.1109/CVPR.2017.690.
[19] Seong, Song, Yoon, Kim, and Choi, 'Determination of vehicle trajectory through optimization of vehicle bounding boxes using a convolutional neural network', *Sensors*, vol. 19, no. 19, p. 4263, Sep. 2019, doi: 10.3390/s19194263.
[20] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, 'KNN Model-Based approach in classification', in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, vol. 2888, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., in Lecture Notes in Computer Science, vol. 2888. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996. doi: 10.1007/978-3-540-39964-3_62.
[21] W. Jia *et al.*, 'A classification algorithm with linear discriminant analysis and axiomatic fuzzy sets', *Mathematical Foundations of Computing*, vol. 2, no. 1, pp. 73–81, 2019, doi: 10.3934/mfc.2019006.
[22] E. A. Zanaty, 'Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification', *Egyptian Informatics Journal*, vol. 13, no. 3, pp. 177–183, Nov. 2012, doi: 10.1016/j.eij.2012.08.002.

Acknowledgments
We thank Amasya University and Bandırma Onyedi Eylül University for providing the opportunity to use computer laboratories in the realization of this study.

Authors Contributions
Yavuz ÜNAL: Performed experimental analysis and wrote the paper.
Emrah Dönmez: Developed and designed the analysis.
Hatice KAYHAN: Collected and prepared data.

Conflicts of Interest
The authors declare no conflict of interest.

Ethical Approval and Informed Consent
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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 11-21
DOI: 10.35377/saucis...1402561
SAUCIS
SAKARYA UNIVERSITY

RESEARCH ARTICLE

Zynq FPGA-Based Acceleration of Kernelized Correlation Filters via High-Level Synthesis of a Custom DFT Block

Mustafa Yetiş [1], Enver Çavuş [2]

[1] Electric and Electronics Engineering Department, Faculty of Engineering and Natural Sciences, Yildirim Beyazit University, Ankara, Türkiye

[2] Electric and Electronics Engineering Department, Faculty of Engineering and Natural Sciences, Yildirim Beyazit University, Ankara, Türkiye

Corresponding author:
Mustafa Yetiş, Electric and Electronics Engineering Department, Faculty of Engineering and Natural Sciences, Yildirim Beyazit University, Ankara, Türkiye
yetis1990mustafa@gmail.com

Article History:
Received: 09.12.2023
Accepted: 15.01.2024
Published Online: 15.01.2024

ABSTRACT

This study presents a hardware-software co-design implementation of an accelerator for the Kernelized Correlation Filter (KCF) tracking algorithm. Leveraging High-Level Synthesis (HLS) and the Zynq heterogeneous platform, the KCF algorithm's performance is enhanced by using a custom hardware implementation for the computationally intensive Discrete Fourier Transform (DFT) operation. Within this framework, a custom combined DFT and inverse DFT IP, named CDFT, is developed and optimized on the Programmable Logic (PL) side of the Xilinx ZCU102 FPGA, whereas the rest of the KCF algorithm is run with customized Petalinux build on the (Processing System) side. To assess real-world performance, a driver for the CDFT IP and a user application were created to measure metrics like Center Location Error (CLE), Intersection over Union (IoU), and Frame per Second (FPS). The designed DFT accelerator achieves a remarkable speedup of 21x compared to a software DFT implementation. At the algorithm level, the KCF accelerator obtains a 6x speed up with negligible precision loss. In comparison to prior studies employing exclusively hardware implementations, the proposed approach demonstrates a high accuracy at a moderate speed, while there exists potential for further optimizations to enhance its performance even further.

Keywords: Tracking, FPGA, KCF, HLS, DFT

1. Introduction

In the modern era of computer vision and artificial intelligence, the demand for real-time and high-performance processing has grown exponentially. One attractive solution to meet this demand is the hardware accelerators, where specific functions are offloaded from the main processor to specialized hardware components to achieve faster execution and reduced power consumption. One such complex algorithm that requires a hardware accelerator for real-time applications is the Kernelized Correlation Filter (KCF) algorithm. The KCF algorithm enables efficient object tracking via kernelized correlation calculations using the Discrete Fourier Transform (DFT) [1]. Among the numerous tracking algorithms in the literature, the KCF algorithm stands out for its exceptional computational efficiency [2]. It achieves this through the use of cyclic shift and the representation of data matrices as circulant matrices.

In the literature of KCF algorithm implementations for real-time object tracking, a series of progressive developments have emerged over the years. Yang et al. [3] presented an accelerator built around the Histogram of Oriented Gradients (HOG) feature extraction and correlation calculations. This architecture was deployed on Xilinx's ZCU102 Ultrascale MPSoC (Multi Processor System on Chip) platform using High-Level Synthesis (HLS). Within this framework, resource-intensive processes like HLS::sqrtf, HLS::DFT, HLS::2DFilter, HLS::exp, matrix division, and element-wise multiplication were implemented on the Programmable Logic (PL). Notably, their results exhibited performance achievements at a resolution of 960x540 and 30 Frame per Second (FPS). It was highlighted that the performance was notably constrained by input image resolution. However, the absence of detailed synthesis reports impedes comprehensive comparison or evaluation of their work. Liu et al. [4] employed a scale pyramid similar to the Discriminative Scale Space Tracking (DSST) algorithm for mitigating target scale variations. Their adaptation of HOG features into a 6-dimensional format incurred a 4% accuracy loss as opposed to the original 31-dimensional format. Leveraging radix-2 Fast Fourier Transform (FFT), the work reported a potential performance of 25 fps. However, accounting for practical considerations which are elaborated in the Experimental Results section of our study, the estimated performance was around 16 fps. Cong et al. [5] proposed a distinctive strategy encompassing multi-feature fusion and the KCF algorithm, all realized through HLS techniques. Their enhancements to the

KCF algorithm were manifest in incorporating Local Binary Pattern (LBP) and HOG features, yielding an enriched target feature representation. An innovative dimensionality reduction method for LBP was introduced to amplify real-time performance while preserving feature extraction efficacy. Executed on FPGA hardware, their algorithm achieved a frame rate of 35 fps with 320x240 resolution while preserving the precision.

Unlike the previous approaches that sought to fully implement the KCF algorithm on the PL section, in this work a software-hardware co-design methodology is adopted. Instead of porting the entire KCF algorithm, we exclusively focus on the most computationally intensive component: the DFT operation. Specifically, we've developed an HLS-based accelerator tailored to optimize the 2D DFT and IDFT (Inverse DFT) functions. A custom 2D Discrete Fourier Transform (DFT) IP is developed in HLS, which achieves over a 24x reduction in latency while maintaining precision compared to the 2D FFT IP based on Xilinx FFT IP Core. The combined 2D DFT and IDFT IP block, namely CDFT, is implemented on the PL fabric of Xilinx's ZCU102 platform, whereas the remaining functionality is realized with a customized Petalinux build on the PS (Processing System) side. Also, on the PS side, a custom device driver is written for the CDFT IP, and a user application is designed to measure the performance using metrics such as FPS, Intersection over Union (IoU), and Center Location Error (CLE). The implemented accelerator system achieves a more than 6x speedup over a software implementation, obtaining an 18.5 fps performance at a resolution of 640x360 with preserving precision. When the resolution is upscaled to 1280x720, the speed reduces to 16 fps, which shows our design is not affected greatly by the resolution change. These results suggest that when designing a high-accuracy HLS-based accelerator for the KCF algorithm on SoC platforms, the dominant portion of the performance gain can be obtained by carefully designing DFT and IDFT blocks. Additional speed-up gains can be obtained by further optimizations.

The rest of this paper is organized as follows. Section 2 reviews the details of the KCF algorithm. Software optimizations and hardware implementation of the KCF algorithm are discussed in Section 3. Then, Section 4 presents the experimental results of the HLS-based accelerator design of the KCF algorithm. Finally, Section 5 concludes the paper.

2. KCF Algorithm

The Kernelized Correlation Filter (KCF) algorithm was introduced by J. F. Henriques et al. in their influential 2015 paper titled "High-Speed Tracking with Kernelized Correlation Filters" [1]. KCF's primary objective is to achieve precise and efficient object tracking across consecutive frames in a video sequence. Unlike traditional methods relying on simple pixel intensities, KCF operates in a higher-dimensional feature space facilitated by kernelization, enabling it to capture intricate relationships between the object and its surroundings.

At its core, KCF involves several key processes that collectively contribute to its robust tracking capabilities. These include feature extraction with consideration for object translations (object movements), kernelization, and learning correlation filters. The feature extraction step not only captures the object of interest and its immediate context but also takes into account potential changes in the object's position across frames. Common techniques like the HOG and Scale-Invariant Feature Transform (SIFT) are employed to extract informative feature vectors. Following feature extraction, KCF takes a crucial step in enhancing its tracking capability. By generating cyclically shifted samples from the extracted features, KCF constructs a more comprehensive training dataset. These samples, created through cyclic shifts, provide a range of variations that account for potential translations in the object's location.

The Gaussian kernel function comes into play to compute the correlation between the target sample, which represents the appearance of the object of interest, and the sample being tested, which represents a potential movement location in the current frame. The coordinates of the point with the highest response value indicate the most recent location of the target. Kernels play a pivotal role in KCF by enabling the algorithm to implicitly operate in a higher-dimensional feature space without the need for explicit transformation computation. The Gaussian kernel, often used in KCF, has shown effectiveness in capturing complex non-linear relationships between data points, enhancing its ability to handle intricate tracking scenarios.

The last key process, which is the central pillar underlying KCF's efficacy, is its approach to learning correlation filters. These filters play a critical role in establishing a robust connection between the appearance of the object and its corresponding feature representation. Notably, KCF executes this learning process within the frequency domain, adjusting filter coefficients to amplify responses for positive samples (object-related features) while attenuating responses for negative samples (background-related features). A distinct advantage of the KCF algorithm arises from its strategic use of the diagonalization property inherent to cyclic matrices in Fourier space. This property contributes to the efficiency of operations within the algorithm's frequency domain computations. By focusing on correlation filters, KCF exploits both spatial and frequency information to refine its tracking precision.

The flowchart of the KCF algorithm is shown in Figure [1]. KCF's execution comprises two primary phases: training and detection. The execution begins by setting up the algorithm parameters and extracting the next frame from the video sequence. In the first frame, a Fourier-domain Gaussian regression label for ridge regression is generated and a Hanning window-based sampling window is calculated. The sampling window is saved for later feature extraction to avoid redundant calculations. Also, HOG features are extracted to be used in the training phase. If it is not the first frame, the tracker first moves to the

detection and then the training phases. During detection, assuming the target's motion range is small, the target and surrounding sub-window images are acquired based on the target's position in the previous probe image. The previous window is used to extract features from the sub-window image. After extracting features, they are transformed using DFT. Kernelized Gaussian cross-correlation and transform dot product results are calculated using IDFT to find the new target position. After the detection phase, the training model is updated. In the training phase, autocorrelation, and update regression coefficients (alpha) are calculated to finalize the model update. For the current target, a sub-window image is taken and transformed using DFT to extract the features in the frequency domain. Rapid Ridge regression is then applied using autocorrelation results. Ultimately, features for the target's final position are extracted, and the classifier is updated. The algorithm continues by tracking the next frame.

The detection operation is repeated for the different scales until the multiscale operation is completed to obtain the best result during the detection phase. The multiscale mode enables tracking to continue even when the target undergoes scale changes and allows tracking to be performed with higher accuracy. If the multiscale mode is enabled, detection can be repeated at half and twice the scales separately in the detection phase, which increases the number of DFT and IDFT operation calls. Both kernelized auto-correlation and cross-correlation operations consist of a 2-norm squared sum, conjugate dot product, Inverse DFT, Gaussian kernel correlation, and DFT calculations. The FHOG operation consists of a Gradient Histogram, Unit Energy Distribution, and 31-dimension FHOG feature calculations.

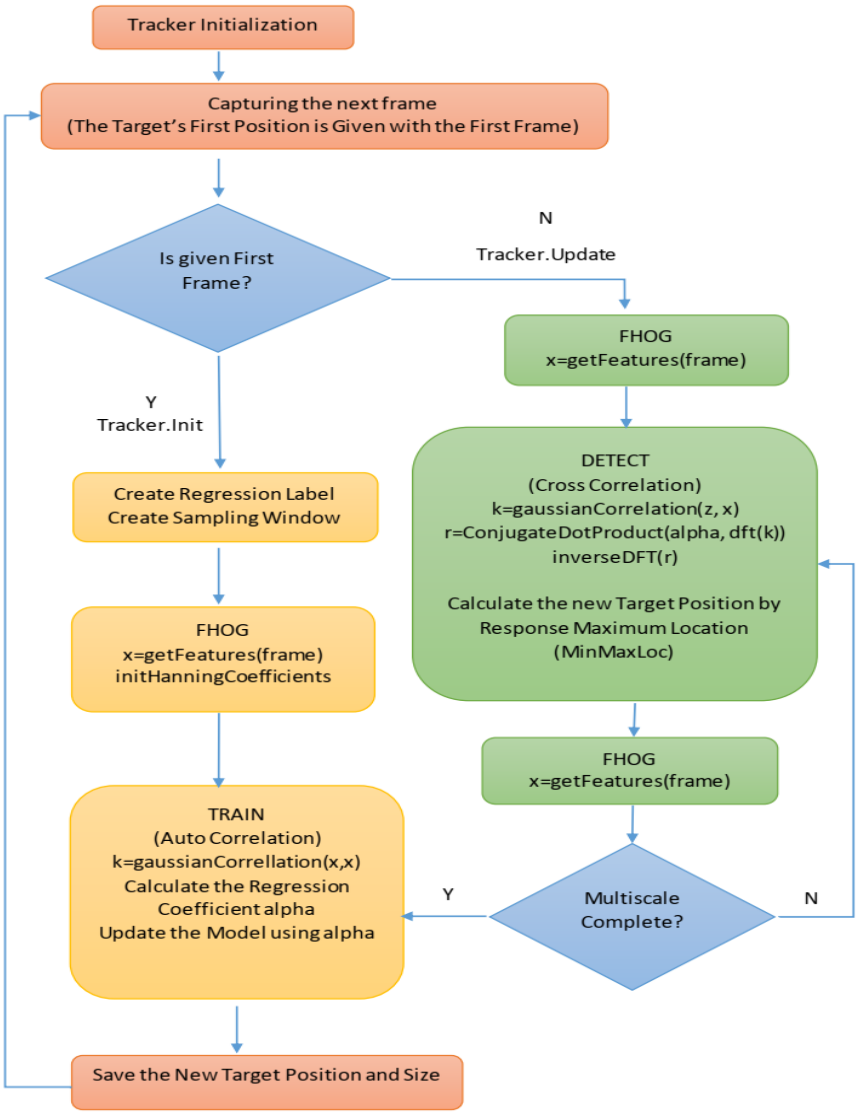


Figure 1 Flowchart of the KCF Algorithm (Multiscale Enabled)

3. Implementation

This section unfolds the implementation process for the KCF tracker accelerator using High-Level Synthesis techniques, which is divided into subsections, each focusing on a specific implementation aspect. Section 3.1 performs an analysis to identify components of the system that can be optimized and parallelized and investigates the impact of the FFT implementation on the accelerator's efficiency and discusses the target platform selection. Subsequently, Section 3.2 delves into the design of DFT and IDFT functions with software optimizations. Section 3.3 covers the basics of HW optimizations using HLS Directives. The structure and design considerations of the accelerator are outlined in Section 3.4.

3.1 Hardware/Software Profiling and Target Platform Selection

Hardware-software co-design in image processing offers a holistic approach to system optimization, combining the strengths of both hardware and software to achieve superior performance, energy efficiency, scalability, reduced latency, and efficient resource utilization. In the literature, several advancements and benefits are highlighted in hardware-software co-design approaches in computer vision. In [10], a co-design framework for video stabilization on FPGA processes real-time video streams at 28 fps, which is twice as fast as software-only approaches. Compared to full-software implementations, a hardware-software co-design significantly lowers overall computational time and hardware resource usage for a feature extraction and image matching algorithm [11]. In [12], a co-design implementation preserves the operating speed of neural networks while allowing flexibility in changing parameters without impacting the FPGA part. An embedded vision services framework in [13] uses heterogeneous hardware accelerators for rapid and efficient integration in applications like image stabilization and moving target indication. A co-design approach for intelligent camera applications significantly reduces development time and boosts performance in FPGA-based services [14]. In [15], a ZYNQ platform-based co-design approach enables high real-time binocular stereo vision, enhancing driving safety.

Based on the significant benefits of the hardware-software co-design approach in FPGA implementations, this paper explores a hardware-software co-design implementation of the KCF tracking algorithm, focusing on optimizing the discrete Fourier Transform (DFT) operation through custom hardware acceleration. The bottleneck of the Kernelized Correlation Filter (KCF) algorithm often lies in the computation of the Discrete Fourier Transform (DFT) and the Inverse DFT (IDFT) operations. These operations are integral to KCF's frequency domain correlation calculations and are performed repeatedly during both the training and detection phases. These operations have a complexity of $O(N^2)$, which can become a performance bottleneck when processing large input data. The co-design approach efficiently balances performance and resource utilization, offloading the most computationally intensive parts to hardware while handling other algorithm aspects in software. The co-design not only achieves significant speed improvements but also maintains accuracy with negligible precision loss, a critical factor for tracking applications. Furthermore, this methodology provides the flexibility needed for future optimizations and enhancements, demonstrating its adaptability to evolving requirements and potential improvements in real-time tracking technology.

In this work, an open-source software implementation of the KCF algorithm is used [6]. To assess the effect of DFT operations on the KCF performance, the latency of the OpenCV DFT function is quantified, along with an assessment of the frequency of calls and the proportion of total DFT latency per frame. Notably, in the case of multiscale mode-enabled KCF, the overhead associated with DFT and IDFT operations accounted for 75% of the total latency. These observations underscore the substantial impact of the DFT function on the KCF algorithm's performance. To mitigate this, a combined accelerator Intellectual Property (IP) block for DFT and IDFT functions, called CDFT, is used within the PL fabric of the FPGA. Consequently, the residual code operates on the PS side of the FPGA. In the context of multiscale KCF, each frame necessitates a total of 379 calls to the CDFT. The initial thought might be that managing 379 input-output transitions is a significant task. However, it is important to recognize that the small array sizes and inherent computational overhead characteristic of the DFT process help to achieve significant acceleration for the KCF algorithm. It is also worth noting that the DFT process primarily encompasses multiply-accumulate operations, rendering it conducive to parallelization and optimization.

FPGAs are known for their high computational speed, abundant resources, and portability. However, implementing intensive computations on FPGAs using hardware language is challenging. In this study, algorithms are automatically converted from the algorithmic layer to the register transfer layer using high-level synthesis, thereby reducing the disadvantages of the FPGA, and leveraging its advantages. The KCF algorithm, with its simple operational structure and high calculation repeatability, is suitable for FPGA implementation. However, certain complex structures requiring flexibility are better suited to be operated on the SoC. Therefore, Xilinx's ZCU102 ZYNQ UltraScale + MPSoC multi-core heterogeneous system, which combines ARM and FPGA capabilities, is selected to meet real-time processing and image processing flexibility requirements. The computing tasks are divided between the PS, representing the software portion, and the PL, representing the hardware portion.

3.2 Custom Design of DFT and IDFT Functions Using Software Optimizations

The 2D DFT and the 2D Inverse DFT functions are seen in Equations 1 and 2, respectively.

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \quad (1)$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \quad (2)$$

Where M x N is the image size, (x, y) gives the image pixel position, and (u, v) represents the spatial frequency - the rate of change of intensity values in the image. First, DFT and IDFT functions are implemented separately based on Equations 1 and 2 respectively. Then they are combined in the CDFT IP to achieve a broader optimization. In this work, two software optimization methods are utilized, namely the usage of lookup tables and a combination of DFT and IDFT operations. These optimizations are explained in the following sub-sections.

3.2.1 Usage of Lookup Tables

After evaluating the initial implementation, it is seen that direct implementations of Equation 1 and Equation 2 have high computational complexity. To reduce the computational complexity, Equation 1 and Equation 2 can be transformed from the polar form to the cartesian form, where sine and cosine functions can be replaced with lookup operations. Equation 3 is used to transform the exponential terms of DFT and IDFT functions to Cartesian form, where in this case θ equals to $2\pi(\frac{ux}{M}+\frac{vy}{N})$.

$$e^{j\theta} = cos\theta + jsin\theta \quad (3)$$

After the transformation to the Cartesian form, sine and cosine estimations can be replaced by the lookup table operations. This is possible because the dimensions of images are fixed in the KCF algorithm and the possible values of x, y, u, and v in the 2D DFT and IDFT functions are also fixed. Therefore, pre-calculated values of sine and cosine functions can be placed in lookup tables and then these values are used in each iteration of the DFT and IDFT loops. Now the real and imaginary parts of the 2D DFT operations can be reduced to Equation 4 and Equation 5, respectively. The same optimization can also be applied to the 2D IDFT function.

$$F(u,v)_{re} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)_{re} * lutsin(u,v,x,y) + f(x,y)_{im} * lutcos(u,v,x,y) \quad (4)$$

$$F(u,v)_{im} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)_{re} * lutcos(u,v,x,y) + f(x,y)_{im} * lutsin(u,v,x,y) \quad (5)$$

In Equation 4 and Equation 5, the real part, cosθ, and the imaginary part, sinθ, are calculated for each x, y, u, and v value and they are stored in the lookup tables, lutcos[u][v][x][y] and lutsin[u][v][x][y], as const float data types. With this optimization, the calculations of sinθ and cosθ, which require 6 operations for θ and 2 operations each for Re{F(u,v)} and Im{F(u,v)}, are eliminated as these values are now retrieved from the lookup table. These operations are performed in nested loops with a total of u*v*x*y iterations during DFT calculation, so even the slightest simplification translates into significant computational savings. A minimum of 15 times speedup is obtained from the test results of DFT acceleration. Furthermore, functions like sine and cosine consume excessive resources, which causes synthesis failure in HLS. This optimization provides resource savings, enabling a successful synthesis. Thus, with this initial software optimization, both energy and logic resource consumption are reduced leading to an improved performance.

3.2.2 Combining the DFT and IDFT Functions

To save on resources and achieve further optimizations 2D DFT and IDFT functions are combined in the CDFT IP block and a parameter is added to select the operation type as DFT or IDFT. As the input image of the DFT function has only real terms, the terms containing the imaginary part of the input can be dropped from the calculations inside the loop. Dropping the redundant terms in calculations makes the algorithm much more efficient. If DFT has been directly implemented using ready-

made functions in HLS, as in other studies, this simplification would not be possible leading to inferior performance. Now, the real and imaginary portions of the 2D DFT function are simplified to Equation 6 and Equation 7, respectively.

$$F(u,v)_{re} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)_{re} * \boldsymbol{lut}\mathbf{sin}(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{y}) \tag{6}$$

$$F(u,v)_{im} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)_{re} * \boldsymbol{lut}\mathbf{cos}(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{y}) \tag{7}$$

As the input of the IDFT function is complex, the terms that contain imaginary parts of input values cannot be dropped in the calculation of the IDFT function. Although using a lookup table doubles the speedup value in DFT and IDFT operations, the effect of dropping the imaginary part in the DFT function reduces the number of calculations by 33.3% as an IDFT is called after every two consecutive DFT calculations in the KCF algorithm for the correlation calculation. It should be noted that these improvements are not at the level of the KCF algorithm, but rather obtained at the level of the DFT function.

3.3 HW Optimizations using HLS Directives

High-level synthesis (HLS) tools play a crucial role in converting C specifications into Register Transfer Level (RTL) designs. This integration of hardware and software disciplines offers several fundamental advantages. HLS tools enable hardware engineers to operate at an elevated level of abstraction, facilitating the development of efficient hardware. Simultaneously, they offer software engineers a new avenue to enhance the performance of their algorithms by targeting Field-Programmable Gate Arrays (FPGAs) for computational acceleration. However, HLS tools alone may not guarantee optimal task scheduling and resource allocation. Consequently, it is vital for designers to incorporate their expertise through specific optimization directives. By applying these directives, developers can effectively implement various optimization strategies, tailoring the HLS process to meet specific performance and efficiency goals.

In this work, the automatic utilization of ARRAY MAP and ARRAY PARTITION are utilized to enhance the efficiency of BRAM access and generate more opportunities for PIPELINE optimization. ARRAY PARTITION directive is employed to divide extensive arrays into several smaller arrays or separate registers. This aims to enhance data access and eliminate BRAM bottlenecks. ARRAY MAP is utilized to consolidate multiple smaller arrays into a larger one, which is subsequently directed to a single extensive memory resource such as RAM or FIFO. By default, HLS stores certain small arrays in the distributed RAM, which can lead to an overuse of the distributed RAM resources. In this particular design, the pre-calculated sine and cosine values are stored in a lookup table, which is partitioned into N-port ROMs, where the number of ports in the ROMS is determined based on the need for simultaneous read operations. All the other optimized storage structures are partitioned into true dual-port RAMs. This storage approach facilitates the simultaneous reading of multiple elements during subsequent optimization, thereby reducing the II (Initiation Interval) and minimizing latency. Note that the initiation interval measures the number of clock cycles required for a specific operation to begin execution once its inputs are available. PIPELINE directive is used to improve latency and maximize kernel throughput and performance. An illustration of II and pipelining optimization is given in Figure 2.

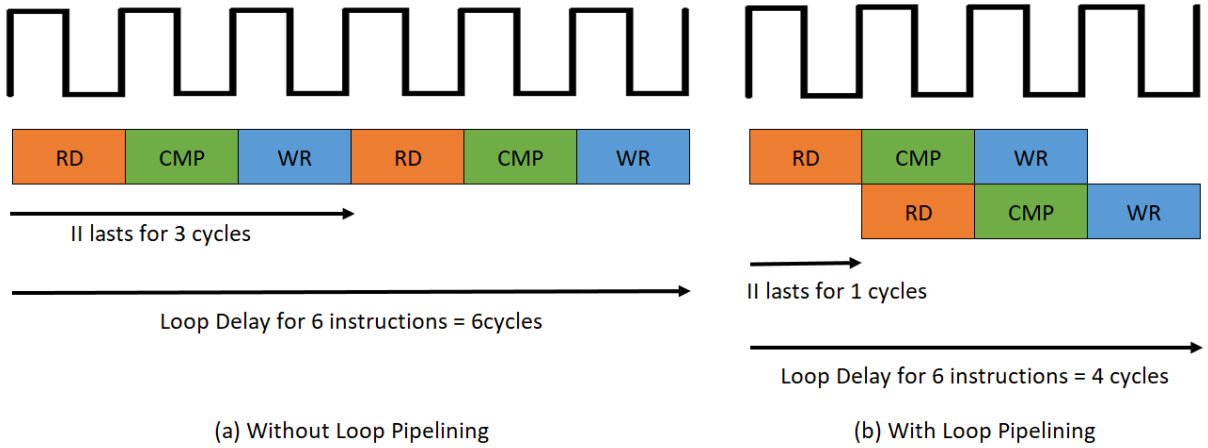


Figure 2 Pipeline Optimization

In addition to the above optimization directives, the CDFT block is realized using a fixed point to obtain a more efficient implementation. Initially, a floating point data type was used, which resulted in an initiation interval (II) of 6 with a latency of 3984 cycles. When the floating point data types are converted to the fixed-point, an II of 1 and a latency of 858 cycles is obtained. This corresponds to 4.6 times latency improvement compared to the floating point results at the IP block level. Table 1 summarizes the implementation results for floating and fixed-point implementation. One apparent conclusion is that fixed point implementation can utilize approximately 3.5 times more DSP blocks, which leads to lesser latency and reduced logic usage as given in FF and LUT columns.

Table 1 Synthesis Reports Results Before and After Optimizations

Module	II	Latency(cycles)	BRAM	DSP	FF	LUT
Floating	6	3983	485	803	187819	125724
Fixed	1	857	488	2852	72615	15460

Another optimization trial involved using dual-port BRAMs instead of single-port BRAMs. However, the synthesis process failed due to exceeding the maximum BRAM, FF, and LUT resource usage limit as a result of the increased complexity of implementing Dual Port BRAM usage. If another FPGA with more BRAM, LUT and FF resources are used, an additional 1.8 times improvement in latency could have been achieved, making the total latency improvement approximately 8.4 times compared to the floating point implementation at the block IP level.

3.4 Accelerator Structure

In this implementation, the KCF accelerator is composed of a 2D DFT and IDFT module located in the PL portion of the Zynq-based FPGA. This module includes dedicated Block RAM (BRAM) units for storing the input and output data for the required DFT and IDFT computations. An AXI interconnect is employed to facilitate seamless communication between these BRAM blocks and the PS. The system operates as follows: When the KCF algorithm calls for the execution of DFT or IDFT operations, instead of utilizing a pre-built GPU-accelerated function as found in OpenCV, the input data is routed to the custom accelerator. The processing operation is initiated by sending a start signal to the accelerator, which then begins its computations. The system subsequently enters a waiting state, awaiting the completion of the computation. Upon completion of the DFT or IDFT computations, the resulting outputs are read from the BRAM units. Writing to BRAM and reading from BRAM operations are conducted by the memcpy function on the PS side. The function of memcpy is only transferring data regions on the memory map. If it is possible, the memcpy function uses the CPU's hardware features to optimize and accelerate the data transfer rates. These outputs are subsequently returned from the function, providing the desired results. In summary, the majority of the code execution occurs within the Processing System (PS) portion of the system.

Figure 3 illustrates the overall system architecture, highlighting various key components. One important point to note is that in this specific design, the GPU remains unused. The rationale behind this decision is to accomplish the same computational tasks with comparable performance and accuracy while minimizing power consumption. Nonetheless, the GPU is retained and made available for potential future use, such as for hybrid optimization methods or other computationally intensive operations required by the application.

4. Implementation Results and Performance Comparisons

In this section, the implementation results of the HLS-based accelerator design of the KCF algorithm using Xilinx's ZCU102 Ultrascale MPSoC platform are presented. First, the resource usage and latency of CDFT are compared with Xilinx FFT IP. Then, the performance comparison of the CDFT-based accelerated KCF algorithm is provided against a pure software implementation and a GPU-accelerated KCF implementation. Finally, the CDFT-based accelerated KCF algorithm is compared with various KCF implementations in the literature. To provide a comprehensive evaluation and comparison, several metrics to assess the performance and accuracy of the implementation are utilized. These metrics, including Center Location Error (CLE), Intersection over Union (IoU), and Frames Per Second (FPS), offer insights into the precision, overlap accuracy, and computational efficiency of the tracker, respectively. Each of these metrics is crucial in understanding both the strengths and potential areas of improvement for the KCF algorithm on the specified platform.

CLE metric evaluates the accuracy of the predicted bounding box center concerning the ground truth center. It is often reported as the Euclidean distance between the predicted center and the ground truth center. The smaller the value means the more accurate the result. Intersection over Union (IoU), also known as the Jaccard Index, measures the spatial overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the ratio of the intersection area to the union area of the two bounding boxes. IoU values range from 0 to 1, where higher values indicate better tracking accuracy. It is formulated in Equation 8.

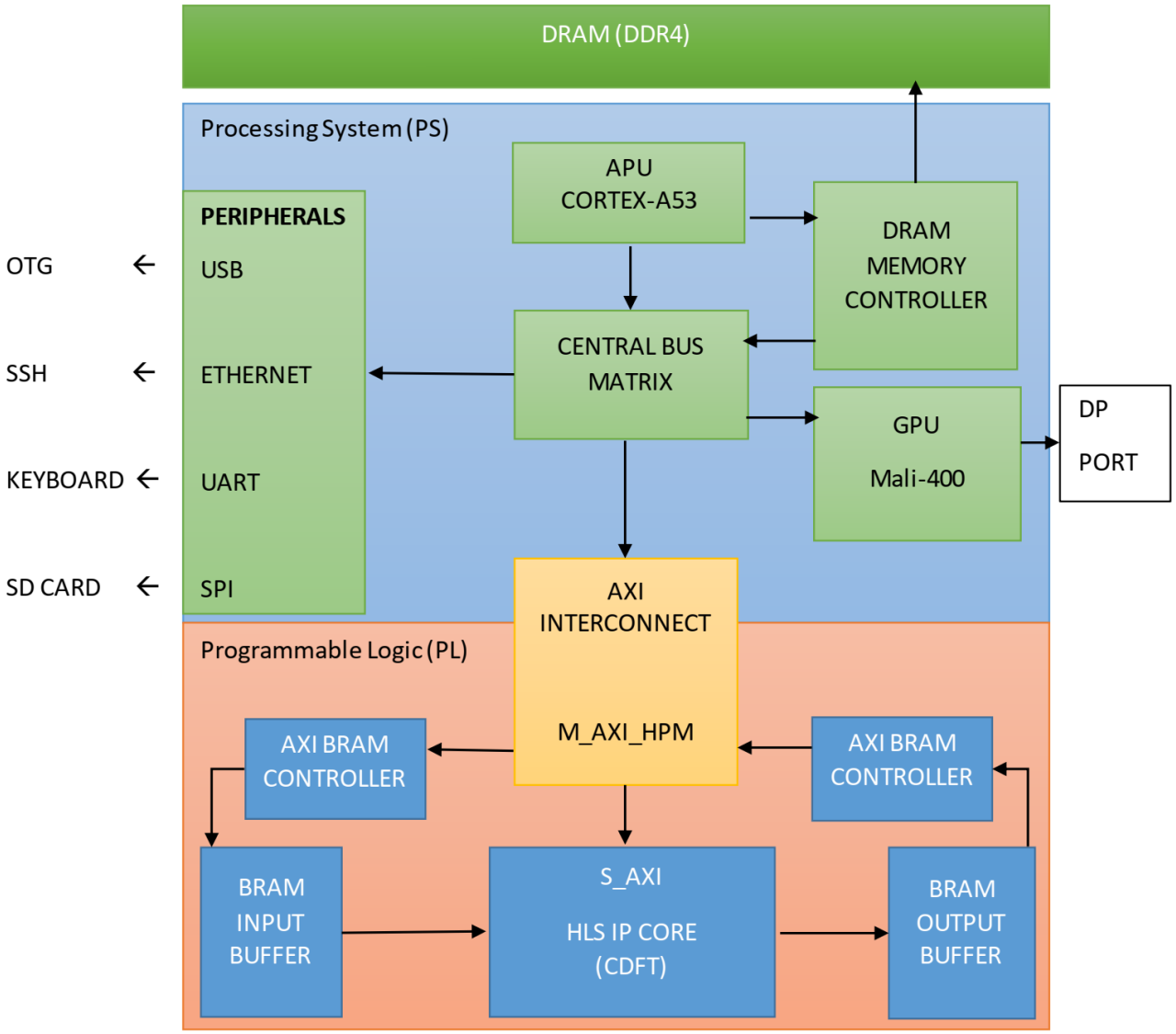


Figure 3 KCF Accelerator Design Architecture

$$IoU = \frac{Intersection\ Area}{Union\ Area} \quad (8)$$

where the Intersection Area is the area of the overlapping region between the predicted bounding box and the ground truth bounding box. Union Area is the total area encompassed by both the predicted bounding box and the ground truth bounding box. The FPS speedup factor shows how much the KCF algorithm is accelerated. While not directly a tracking-specific metric, FPS represents the processing speed of the tracking algorithm. Faster algorithms are generally preferred as they can handle real-time applications. The Latency improvement factor shows how PL implementation of the accelerator IP finishes the processing earlier than its software version. This improvement will be evaluated by the speedup factor calculated using the FPS metric.

In this paper, the speedup comparisons are calculated according to Amdahl's Law [7]. To monitor the real performance of the designed accelerator, petalinux is customized by reserving specific memory and enabling necessary packages for a user application, which measures the aforementioned metrics. This makes it possible to evaluate the performance of the accelerator design in a real-world scenario.

First, the efficiency of the designed CDFT block is compared with the Xilinx FFT IP Core provided within the HLS library. In the Xilinx FFT core, a 2D Fast Fourier Transform (FFT) is implemented using the row-column algorithm and the 1D FFT

IP core. The first step involves performing a one-dimensional FFT in each row, while the second step requires performing a one-dimensional FFT in each column. For the second step, the results of the first step are transposed, and then a one-dimensional FFT is computed for each row again. To minimize resource usage, fixed-point data types are employed in FFT IP Core, where the 'radix_2_io' architecture is used with 64-bit complex data types for input and output. In CDFT IP, the same data width is used, but real and imaginary components are implemented as separate parameters. Zero-padding is applied to adapt the input size to the FFT length (32 samples), and spectral leakage prevention is addressed using Hanning window coefficients, which are stored in a constant lookup table. The implementation based on Xilinx FFT IP resulted in a frame rate of approximately 3 fps and significant accuracy loss, making tracking unfeasible. Table 2 compares the resource consumption and latency values obtained from synthesis reports. The custom 2D DFT IP in HLS achieves over a 24x reduction in latency while maintaining precision compared to the Xilinx FFT IP Core-based 2D FFT algorithm.

Table 2 Synthesis Reports Results Comparison Between Xilinx FFT IP Core and Our custom DFT IP

Module	Optimization	Latency(cycles)	BRAM	DSP	FF	LUT
Xilinx FFT IP core	Dataflow	21198	38	12	15844	15564
CDFT IP	Pipeline II=1	857	488	2852	72615	15460

Next, the performance of the designed KCF accelerator is evaluated against software-based and GPU-based KCF algorithms using the Bolt from the VOT2014 dataset [8]. The comparison is presented in Table 3.

Table 3 Performances from our tests using the Bolt dataset from VOT2014

	KCF-SW	KCF-GPU_accelerated	KCF-CDFT_floating_point	KCF-CDFT_fixed_point
FPS	3	22	10	18.5
IoU	0.961184	0.964186	0.949865	0.965927
CLE	4.6116	4.6105	4.6558	4.8077
Speedup	-	7.33x	3.33x	6.17x
Power Consumption	12W	15W	7W	10W

The measurements show that the fixed-point CDFT-based accelerator uses less power than both the single-core software (SW) version and the GPU-accelerated version. In contrast, it achieves 18.5 fps compared to 22 fps of GPU accelerated GPU. The original KCF source code utilizes the DFT function from the OpenCV library, which leverages GPU acceleration. Implementing the CDFT block on the PL section achieves a performance close to the on-chip GPU in the PS part.

Table 4 shows the latency reductions when using the hardware-supported memcpy function for input and output transfers in the fixed-point CDFT-based accelerator. For each frame in the KCF algorithm, our CDFT is called 252 times as a DFT function and 127 times as an IDFT function. The memcpy optimization provides a significant 4 times latency improvement, increasing the frame rate from 12 fps to 18.5 fps.

Table 4 Performances and latencies before and after memcpy function usage

	Before memcpy utilized	After memcpy utilized
DFT BRAM input Transfer Latency(ns)	1630	200
IDFT BRAM input Transfer Latency(ns)	2080	2690
DFT BRAM output Transfer Latency(ns)	115231	28122
KCF Total CDFT Transfer Latency Per Frame(ms)	44.35	11.05
KCF Speedup	4x	6.17x

In Table 5, a comparison with the previous studies in the literature is also presented. Note that except for the CDFT-based accelerator, all other designs use the HLS library to implement KCF-related functions in the PL section. As it is clear from Table 5, one advantage of the hardware-software co-design is the high accuracy. Although our study achieves a similar speed as the compared study [4], the accuracy is superior for several reasons. When a significant portion of the operations in the KCF algorithm is performed in the PL using a fixed-point implementation, a high precision loss occurs. These losses include the use of FFT (possible padding losses), the use of 6-dimensional HOG instead of 31 dimensions resulted in a reported 4% accuracy loss, implementation using fixed point data type instead of float for variable types in functions, and the use of bilinear interpolation for resizing. From this perspective, our work requires less effort, provides more flexibility to the KCF algorithm for further optimizations, and offers higher accuracy at the same speed compared to the study [4].

Table 5 Performance comparison of the CDFT-based accelerator with other implementations in the literature

Study Name	Hardware	Method	Resolution	FPS	Prec. Loss
Our design	CDFT	HLS	640x360	18.5	Low
Liu et al.[4]	Scale pyramid similar to DSST, HOG dimension reduction, Radix-2 FFT, Bilinear Interpolation	HLS*	1280x720	14-22	High
Cong et al.[5]	Fusing LBP with HOG, Dimensionality reduction method for LBP	HLS*	320x240	35	Low

*Almost All functions are from the HLS library.

In the study by Cong et al. [5], The resolution is significantly lower than the 640x360 resolution in our study. The image resolution has a big impact because transferring the entire image to the PL impacts the throughput negatively. Increasing the resolution on the same platform at the same fps is not possible because it would involve excessive resource consumption. To fit it, a significant compromise would have to be made in terms of accuracy and fps values. PIPELINE, radix-8 multiple FFT instances, and similar resource-hungry optimizations were required to achieve 35fps. It is clear that if implemented at the same resolution with the same number of resources, this study would yield lower fps and accuracy compared to ours.

5. Conclusions and Future Works

In contrast to the earlier FPGA implementation studies of the KCF algorithm, which port all the functions to the PL section, a hardware-software co-design approach is utilized on Xilinx's ZCU-102 board. Only DFT and IDFT operations are performed on the PL side, resulting in a speedup of over 21 times for the IP and over 6 times at the algorithm level for KCF. By maintaining sufficient data width in the fixed-point version and implementing only the DFT function on the PL section, precision loss is kept negligible. Examining the experimental results, it is observed that a performance close to that of a GPU with lower power dissipation is achieved. While KCF is effective for tracking, for a full object tracking solution, it often works in tandem with a detection algorithm. The processing load of KCF is offloaded to the accelerator in the designed PL, freeing the GPU for the detection algorithm and enabling the meeting of real-time requirements if further optimizations are implemented. The factors influencing this speedup are discussed and compared to other works in the literature. In this approach, an advantage emerges from not transferring the entire image. It is found that, while other studies implement multiple functions to achieve the same performance as this IP, which implements only one function in the PL, this work provides higher accuracy at the same speed. In other studies, the implementation of multiple complex functions in the KCF algorithm, due to limited PL resources, prevents sufficient optimization and necessitates transferring the entire image to the PL side.

For future work, several further performance optimizations are possible, especially when using a target platform with more PL resources. The first optimization can be to use dual-port BRAMs instead of single-port BRAMs. Using dual-port BRAMs, PS-PL transfer latency can be halved, leading to better real-time performance. Additionally, if a higher clock frequency is used, real-time performance requirements can be met while preserving precision. Another improvement can be achieved using a parallel implementation of the Kernelized Gaussian Correlation. Each term in the Kernelized Gaussian Correlation estimation, can be calculated in parallel to improve processing time. The correlation calculation involves two separate DFT and one IDFT operation, which can be optimized using the extended version of the proposed implementation in this paper. In light of this data, implementing Gaussian Kernelized Correlation on a platform with higher PL resources can increase the number of DFTs processed per unit time and reduce the input-output transfer count, resulting in higher performance.

References

[1] J.F. Henriques, R. Caseiro, P. Martins, et al., "High-speed tracking with kernelized correlation filters", I*EEE Transactions on Pattern Analysis and Machine Intelligence*, 37,(3), pp. 583-596, 2015.

[2] M. Mueller, N. Smith, B. Ghanem, "A benchmark and a simulator for UAV tracking". *European Conf. Computer Vision*, pp. 445-461, Amsterdam, Netherlands, October 2016.

[3] H. Yang, J. Yu, S. Wang, X. Peng, et al. "Design of airborne target tracking accelerator based on KCF". J. Eng., 2019, Vol. 2019 Iss. 23, pp. 8966-8971. IET Journals doi: 10.1049/joe.2018.9159, 2019

[4] X. Liu, Z. Ma, M. Xie, J. Zhang, T. Feng, et al., "Design and implementation of scale adaptive Kernel correlation filtering algorithm based on HLS", *IEEE ICSPCC.* doi:10.1109/ICSPCC52875.2021.9564815, 2021

[5] P. Cong, M. Xie, K. Yang, X. Zhang, H. Su and X. Fu, "Design and implementation of multi-feature fusion kernel correlation filtering algorithm based on HLS," *IET International Radar Conference (IET IRC 2020)*, Online Conference, 2020, pp. 645-649, doi: 10.1049/icp.2021.0761.

[6] J. Faro, "C++ KCF Tracker" 2021. [Online]. Available: https://github.com/joaofaro/KCFcpp ,[Accessed: 30.08.2023]

[7] G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities" . *AFIPS Conference Proceedings* (30): pp. 483–485, doi:10.1145/1465482.1465560, 1967
[8] M. Kristan, J. Matas, A. Leonardis, and et al., "The visual object tracking vot2014 challenge results.", ECCV Workshop, 2014
[9] S.G. Raju, "Accessing BRAM in LINUX" 2021. [Online]. Available: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842412/Accessing+BRAM+In+Linux, [Accessed: 01.04.2023]
[10] J. Hassan, M. Bilal, and S. Masud, A Hardware-Software co-design framework for real-time video stabilization. *J. Circuits Syst. Comput.*, 29, 2020. 2050027:1-2050027:18. doi:10.1142/S0218126620500279.
[11] C. Chien, C. Chien, and C. Hsu, "Hardware-software co-design of an image feature extraction and matching algorithm. *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*, 37-41, 2019. https://doi.org/10.1109/ICoIAS.2019.00013.
[12] T. Dang, and Y. Hoshino, "Hardware/Software co-design for a neural network trained by Particle Swarm optimization algorithm". *Neural Processing Letters*, 49, 481-505, 2019. doi: 10.1007/s11063-018-9826-4.
[13] E. Gudis, P. Lu, D. Berends, K. Kaighn, G. Wal, G. Buchanan, S. Chai, and M. Piacentino, "An embedded vision services framework for heterogeneous accelerators. *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 598-603, 2013. doi: 10.1109/CVPRW.2013.90.
[14] A. Ruta, R. Brzoza-Woch, and K. Zielinski, "On fast development of FPGA-based SOA services—machine vision case study". *Design Automation for Embedded Systems*, 16, 45-69, 2012. https://doi.org/10.1007/s10617-012-9084-z.
[15] Y. Pan, M. Zhu, J. Luo, and Y. Qiu, "A Hardware/Software co-design approach for real-time Binocular Stereo Vision based on ZYNQ (Short Paper)", 719-733, 2018. https://doi.org/10.1007/978-3-030-12981-1_50.

Conflict of Interest Notice

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

Ethical Approval and Informed Consent

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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 22-35
DOI: 10.35377/saucis...1341082
SAUCIS
SAKARYA UNIVERSITY

RESEARCH ARTICLE

Classification of Malware Images Using Fine-Tunned ViT

Oğuzhan Katar [1], Özal Yıldırım [2]

[1] Department of Software Engineering, Faculty of Technology, Fırat University, Elazığ, Türkiye
[2] Department of Artificial Intelligence and Data Engineering, Faculty of Engineering, Fırat University, Elazığ, Türkiye

Corresponding author:
Oguzhan Katar, Department of Software Engineering, Faculty of Technology, Firat University, Elazig, Türkiye
okatar@firat.edu.tr

Article History:
Received: 10.08.2023
Accepted: 25.01.2024
Published Online: 25.01.2024

ABSTRACT
Identifying and classifying malware has become a critical task in ensuring the security and resilience of computer systems and networks. Traditional techniques for malware assessment often rely on signature-based methods, which struggle to keep up with the constantly evolving landscape of malware variations. Recently, the application of advanced deep learning methods has shown promising results in automating the malware classification process. This study presents an innovative strategy for classifying malware images using the Vision Transformer (ViT) architecture. The ViT model is adapted to the domain of malware analysis by representing malware images as input tokens. A comprehensive dataset of 14,226 malware samples from 26 families was used to evaluate the effectiveness of this approach. A comparative analysis was performed between the performance of the ViT-based classifier, traditional machine learning approaches and other deep learning architectures. Our experimental results demonstrate the potential of ViT in handling malware images, achieving a classification accuracy of 98.80%. The presented approach establishes a strong foundation for further research in utilizing cutting-edge deep learning architectures for enhanced malware analysis and detection techniques.

Keywords: Malware detection, Vision Transformer, Deep learning, Network Security.

1. Introduction

The relentless advance of Internet technology has brought about a period of rapid expansion in the computer software sector. This has led to the development of a wide range of software applications that have become seamlessly integrated into the fabric of everyday life [1]. Nevertheless, this technological advancement has concurrently led to a concerning issue: the rampant proliferation of detrimental malware. This presents a substantial threat to the security of users' personal information, causing significant disruptions to computers, servers, and cloud infrastructures [2]. Malware is a type of software that threatens computer systems today and is designed to steal user data, exploit systems or for other malicious purposes [3]. This malware often runs without the user's permission or awareness and compromises personal security, privacy and the integrity of computer systems [4]. The most common types of malware today can be summarized as follows:

- ***Viruses:*** Malware that can infect other files and spread by making copies of themselves. They usually attach themselves to other files, infect them and then spread [5].
- ***Worms:*** Malicious software that self-replicates and spreads rapidly across computer networks to other systems. They do not infect files but spread by sending copies of themselves across the network to other devices [6].
- ***Trojans:*** Malicious software hidden in seemingly innocuous and useful programs that users tend to download. While installed by the user, its real purpose may be to covertly cause damage or steal information [7].
- ***Spyware:*** Software designed to secretly steal information by monitoring a user's computer activity. This information can often be sensitive, such as a user's online habits, personal or financial information [8].
- ***Adware:*** Malware designed to generate revenue by constantly displaying advertisements to the user. The ads appear without the user's consent and often have a negative impact on the user's experience [9].
- ***Ransomware:*** Malware that encrypts the user's files or system and demands a ransom to get them back [10].

Malware detection methods can be divided into two main categories: traditional static and dynamic approaches [11]. Static methods involve analyzing the structural characteristics of software to determine the presence of malware. In contrast, dynamic methods monitor the behavior of executing programs to identify potential malware instances [12]. These strategies offer distinct advantages and disadvantages in the quest for effective malware detection.

Dynamic detection is particularly accurate because it actively monitors the behavior of programs as they run, allowing it to quickly identify malicious software [13]. However, this approach is time-consuming because it requires continuous monitoring of running processes. This real-time analysis may not be conducive to the timely detection of emerging malware threats. In contrast, static detection can serve as a valuable complement to dynamic methods, helping to overcome their time-consuming aspect. By analyzing the structural attributes of software without executing it, static detection can quickly assess potential threats [14]. However, traditional static detection techniques rely on powerful antivirus engines and extensive virus databases [15]. This reliance on known signatures and patterns poses a significant challenge in detecting unseen or previously unknown malware, often resulting in the limited performance of traditional static approaches [16]. Efforts to strike a balance between static and dynamic detection techniques remain critical to improving the overall effectiveness of malware detection [17].

To overcome the limitations of traditional static detection approaches, researchers have looked at innovative ways to detect malware using visualization technology. These innovative techniques have shown promising performance in malware detection [18]. In many cases, malware variants are created through automation or reuse of critical function modules, resulting in a degree of similarity in their binary or assembly code [19]. Visualization technology is proving to be very useful in capturing these similarities and presenting them in a visual form. Interestingly, the challenges of malware detection are similar to those of image recognition, as both require the identification of variants or patterns within the original samples. By visually representing the structural characteristics and behavioral patterns of malware, visualization-based malware analysis reveals unique features that improve detection accuracy. By revealing hidden relationships and commonalities between malware variants, this approach promises to strengthen defenses against both known and previously unseen threats [20].

With the rapid advancement of artificial intelligence technology, researchers are increasingly using deep learning models to detect and classify malware. Yadav et al. [21] proposed a novel deep learning based two-stage framework for detecting and classifying DEX files images. The framework uses the EfficientNetB0 model to extract relevant features from malware images. These features are then processed through a stacking classifier, utilizing linear support vector machine (SVM) and random forest (RF) algorithms as base-level classifiers and logistic regression. The proposed method achieves impressive results, obtaining 100% accuracy in binary classification and 92.9% in 5-class classification. Khan et al. [22] proposed a malware detection framework called Deep Squeezed-Boosted and Ensemble Learning (DSBEL). The proposed DSBEL framework incorporates a novel Squeezed-Boosted Boundary-Region Split-Transform-Merge (SB-BR-STM) CNN that employs multi-path dilated convolutional, boundary and regional operations to capture global malicious patterns. The performance evaluation of the DSBEL framework and the SB-BR-STM CNN is performed on the IOT_Malware dataset, yielding results of 98.50% accuracy, 97.12% F-1 score, 95.97% recall and 98.42% precision. Xing et al. [23] proposed a state-of-the-art malware detection method. The method introduces a novel approach that involves representing malware as grey-scale images and incorporating an auto-encoder network for analysis. The viability of the grey-scale image representation is assessed by evaluating the reconstruction error of the auto-encoder. Furthermore, the dimensionality reduction capabilities of the auto-encoder are exploited to classify malware from benign software. Experimental evaluations conducted on an Android-side dataset demonstrate the effectiveness of the model, which achieves an impressive accuracy rate of 96% and a stable F1-score of around 96%. Asam et al. [24] introduced a novel CNN-based architecture called IoT malware detection architecture (iMDA) for effective detection. The iMDA architecture is designed with modularity and incorporates various feature learning schemes such as edge exploration and smoothing, multi-path dilated convolutional operations, and channel squeezing and boosting within the CNN framework. The performance evaluation of iMDA on a benchmark IoT dataset demonstrates its promising capabilities in malware detection, achieving 97.93% accuracy, 93.94% F-1 score, 98.64% precision, and 88.73% recall. Kumar and Janet [25] proposed deep transfer learning for malware image classification (DTMIC). By converting portable executable files (PEs) into grayscale images, DTMIC exploits the visual characteristics of similar malware families. The effectiveness and robustness of DTMIC are evaluated using MalImg and Microsoft BIG dataset. DTMIC achieves high detection accuracies of 98.92% for MalImg and 93.19% for Microsoft datasets, outperforming established CNN architectures.

Within the realm of malware detection and classification, CNN-based architectures, together with classical machine learning methods, have achieved significant success in image processing and are widely used in the literature. However, CNN-based classifiers often have limitations such as special architectural designs and input data with predetermined dimensions [26]. To overcome such problems, Vision Transformer (ViT) is a new approach that has recently attracted attention [27]. It is a transformer architecture based on this attention mechanism and designed for image classification problems. By treating the data as an irregular array of pixels, ViT provides a more flexible approach to better understand the relationships of objects in images and to detect important patterns [28]. ViT's particular scalability and ability to deal with large datasets and complex classification problems make it a suitable and promising candidate for malware image classification. Thanks to its ability to learn distant relationships between data, ViT can detect subtle differences between different types and subtypes of malware

and achieve higher classification accuracy. In addition, ViT's attention mechanism prevents significant information loss in the feature extraction process, allowing for more comprehensive and detailed analysis.

This paper proposes a method for fine-tuning the default ViT architecture for automatic classification of malware images. The proposed model initially divides the image into patches and extracts features through the encoder network. Subsequently, these features are classified using an MLP (Multi-Layer Perceptron) head to determine the malware class. Moreover, to the best of the authors' knowledge, this is the first study on the classification of the MaleVis dataset utilizing the ViT model.

The main contributions of this study can be summarized as follows:

- The ViT model achieved high performance values using the MaleVis dataset, which contains 25 different types of malware.
- The ViT model demonstrated effective prediction on input images of different classes, eliminating the need for any feature engineering.
- A system that can classify malware images without depending on any resolution value.
- Leveraging the transfer learning method, the study attained higher performance values compared to similar research with fewer hardware requirements.

The remaining sections of this paper are structured as follows: Section 2 outlines the proposed method, explains the dataset used for model training, introduces the classifier model and describes the performance metrics. Section 3 explains the experimental setup and presents the results obtained, and Section 4 contains the discussion section of the study. The conclusions of the study are presented in Section 5.

2. Material and Methods

This paper proposes a ViT model for classifying malware images. Instead of taking the input images directly as input, the model processes them by dividing them into patches and vectorizing them. This approach allows the model to work faster and more efficiently by processing smaller parts of the input images by dividing them into patches. A schematic diagram of the proposed malware image classification method is given in Figure 1.

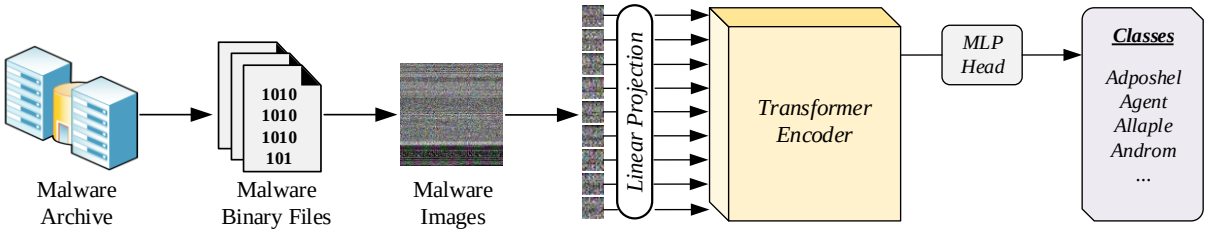


Figure 1 Schematic diagram of the proposed method.

The effectiveness of the deep learning models is highly dependent on the quality, diversity and size of the dataset. A well-curated and comprehensive dataset of malware images plays a crucial role in enabling the ViT model to learn meaningful representations and features that distinguish between different types of malware. By using a large and diverse dataset of malware images, the ViT model can effectively learn to extract relevant patterns and structures from the input images. The model's ability to divide the input images into smaller patches and vectorize them allows it to exploit the inherent spatial relationships in the dataset. This process facilitates the capture of fine-grained details and local features within each patch, enabling the model to make accurate and efficient classifications. Furthermore, the use of a diverse dataset can help improve the generalization capabilities of the ViT model. By exposing the model to a wide variety of malware samples with different characteristics, the model can better adapt to unseen and real-world scenarios. Consequently, this increases the overall robustness and reliability of the proposed malware image classification method.

2.1 Dataset

The public dataset used in this study is called Malware Evaluation with Vision (MaleVis) [29]. The MaleVis dataset consists of 25 different malware families, collected from samples that appeared between 2017 and 2018. These samples were created on PE files prepared by a cybersecurity company. These binary code files were converted into 224×224-pixel PNG images using the 'Bin2png' library. The samples that were randomly selected from the dataset are shown in Figure 2.

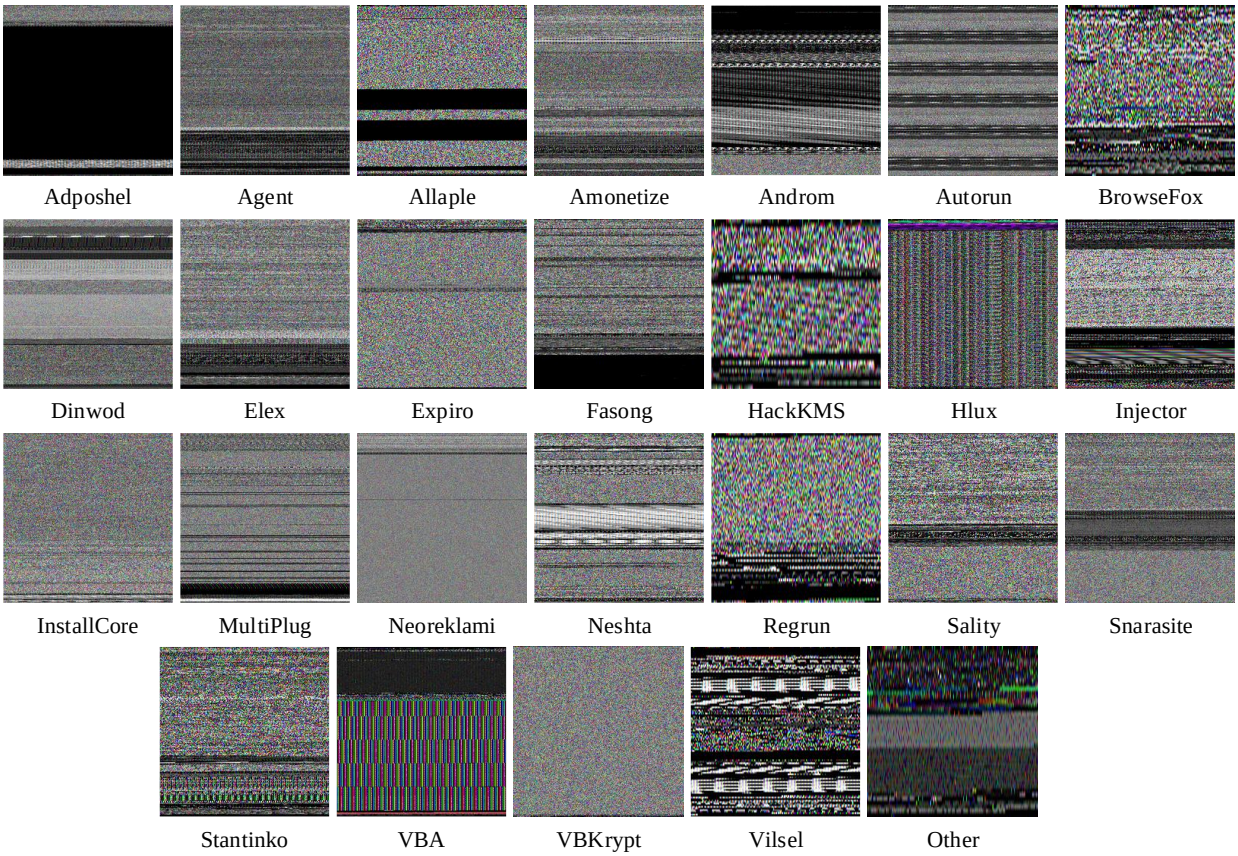


Figure 2 MaleVis Samples[29]

Table 1 Distribution of the dataset samples

Class Name	Category	Number of Samples in MaleVis	Number of Samples in Subsets (Train/Validation/Test)
Adposhel	Adware	494	396/49/49
Agent	Trojan	470	376/47/47
Allaple	Worm	478	382/48/48
Amonetize	Adware	497	397/50/50
Androm	Backdoor	500	400/50/50
Autorun	Worm	496	396/50/50
BrowseFox	Adware	493	395/49/49
Dinwod	Trojan	499	399/50/50
Elex	Trojan	500	400/50/50
Expiro	Virus	501	401/50/50
Fasong	Worm	500	400/50/50
HackKMS	Trojan	499	399/50/50
Hlux	Worm	500	400/50/50
Injector	Trojan	495	397/49/49
InstallCore	Adware	500	400/50/50
MultiPlug	Adware	499	399/50/50
Neoreklami	Adware	500	400/50/50
Neshta	Virus	497	397/50/50
Regrun	Trojan	485	389/48/48
Sality	Virus	499	399/50/50
Snarasite	Trojan	500	400/50/50
Stantinko	Backdoor	500	400/50/50
VBA	Virus	500	400/50/50
VBKrypt	Trojan	496	396/50/50
Vilsel	Trojan	496	396/50/50
Other	Legitimate	1832	1466/183/183

The MaleVis dataset, which consists of 26 classes, only possesses legitimate content in the form of the "Other" class, while the remaining classes, constitute malware. The dataset is divided into five categories of malware, including Adware, Trojan, Worm, Backdoor, and Virus. With a balanced distribution of approximately 500 samples per class, no data augmentation was deemed necessary. The dataset was divided into training, validation, and testing subsets, with 80% of the samples allocated for training, 10% for validation, and 10% for testing. The class-wise distribution of the dataset is presented in Table 1.

2.2 ViT Model

The profound impact of transformer networks on natural language processing tasks has been widely acknowledged. Building upon the success of the original transformer architecture, Dosovitskiy et al. [30] introduced the ViT model, specifically tailored for image processing tasks. The ViT model consists of self-attention blocks and MLP networks, equipped with linear projection and positional embedding mechanisms to handle input images effectively. The ViT architecture is based on the process of dividing the input image into fixed size, non-overlapping patches [31]. These patches are then flattened and a spatial embedding step is performed using linear projection. The purpose of spatial embedding is to preserve the spatial information of the original image within the flattened patches. The resulting vector is then fed into a stack of N transform encoder blocks. The architecture of these encoder blocks used for feature extraction in the ViT model is given in Figure 3.

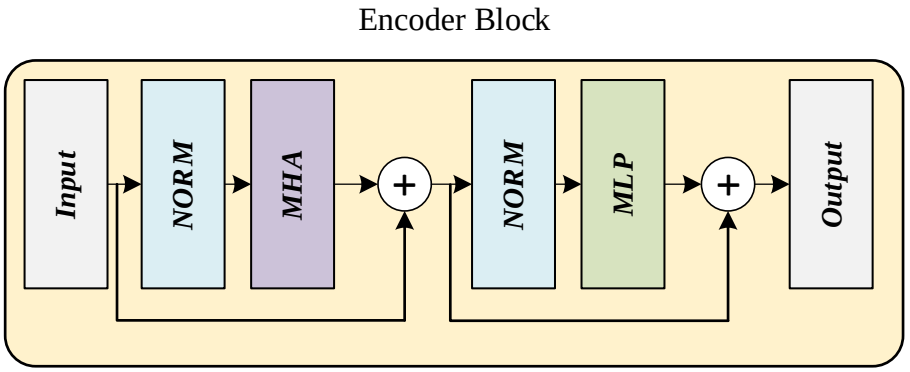


Figure 3 Architecture of the ViT encoder block

The basic components of a transformer encoder block include multi-head self-attention (MHA) and MLP layers. Each component is complemented by a normalization layer and a residual connection for improved training stability. Within MHA, self-attention is applied to each patch individually, producing three distinct vectors: query (Q), key (K) and value (V) [32]. To measure the importance or saliency of each embedded patch, a dot product operation is performed between the Q and K vectors, producing a score matrix. This matrix is then passed through the SoftMax activation function, which converts the scores into attention weights [33]. Finally, the output is obtained by element-wise multiplication of the attention weights and the V vector. This process produces the self-attention result as seen in Equation 1, where dk denotes the dimensionality of the key vector K.

$$Self\ Attention\ (Q,K,V) = SoftMax(\frac{QK^T}{\sqrt{d_k}}) \times V \quad (1)$$

The self-attention matrices are concatenated and then passed to a linear layer followed by a regression head. This application of self-attention allows the model to recognize relevant semantic features at different locations in the image, facilitating accurate classification. Within the transformer encoder, there can be MHA blocks, each contributing to the overall understanding of the image. Following the MHA layer, the transformer block contains an MLP. These MLP layers are equipped with a GeLU activation function [34]. The final output of the transformer block is calculated as shown in Equation 2. K.

$$Transformer\ Encoder_{out} = MLP(NORM(MHA_{out})) + MHA_{out} \quad (2)$$

2.3 Performance Metrics

Evaluating the performance of deep learning models is critical to understanding their effectiveness in solving real-world problems. Performance metrics play a vital role in quantifying the quality and reliability of these models, and some of the

most important metrics include true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These metrics can be summarized as follows.

- TP signifies the number of instances that are correctly classified as belonging to that specific class. It's the count of correctly identified positive samples for each class.
- TN signifies the number of instances that are correctly classified as not belonging to that specific class. It's the count of correctly identified negative samples for each class.
- FP signifies the number of instances that are incorrectly classified as belonging to that specific class when they actually don't belong to it. It's the count of incorrectly identified positive samples for each class.
- FN signifies the number of instances that are incorrectly classified as not belonging to that specific class when they actually belong to it. It's the count of incorrectly identified negative samples for each class.

In addition to the basic metrics of TP, TN, FP and FN, the evaluation of deep learning models includes the construction of a confusion matrix, which provides a comprehensive summary of the model's performance across all classes. The confusion matrix is a table that represents the predicted labels against the true labels, allowing a more detailed analysis of the model's behavior. The confusion matrix for the performance evaluation of a 26-class deep learning model is shown in Figure 4.

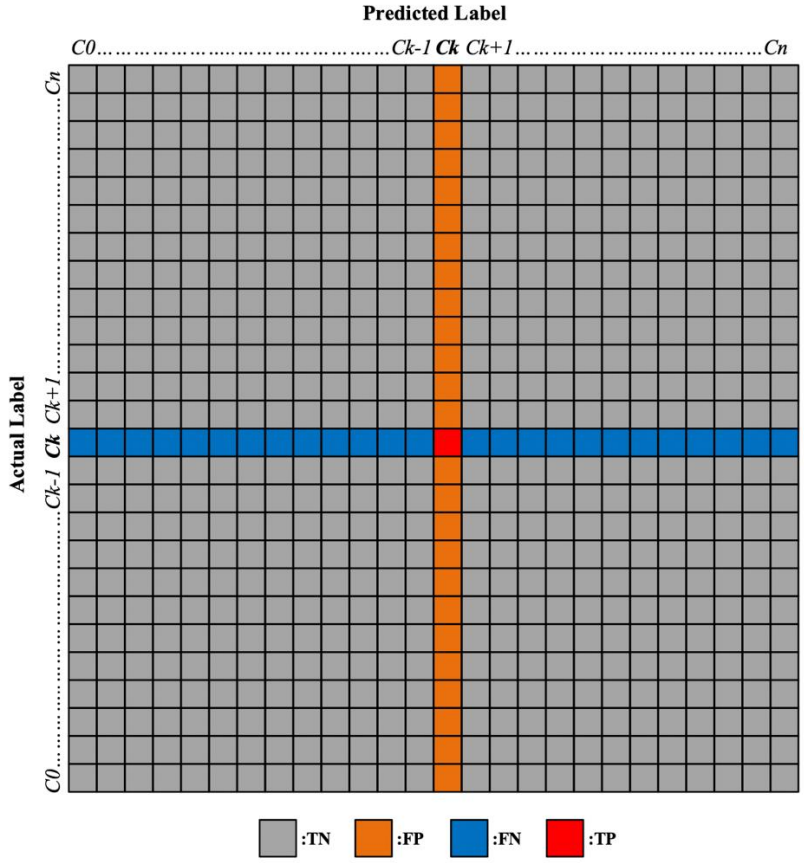


Figure 4 A confusion matrix with 26 classes

The main performance measures and mathematical equations used to evaluate deep learning models can be summarized as follows:

1. ***Accuracy (Acc):*** This metric quantifies the overall correctness of the model's predictions. It is the ratio of correctly classified samples to the total number of samples. The Acc can be calculated as seen in Equation 3.

$$Acc = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{3}$$

2. ***Precision (Pre):*** Premeasures the proportion of positively labelled samples that the model correctly identifies. It focuses on the ability of the model not to misclassify negative samples as positive, reflecting its ability to make accurate positive predictions. The Pre can be calculated as seen in Equation 4.

$$Pre = \frac{TP}{(TP + FP)} \quad (4)$$

3. ***Recall (Rec):*** Rec, also known as sensitivity, assesses the ability of the model to correctly identify true positive samples. It is particularly relevant when the aim is to minimize false negatives and avoid missing positives. The Rec can be calculated as seen in Equation 5.

$$Rec = \frac{TP}{(TP + FN)} \quad (5)$$

4. ***Specificity (Spe):*** This metric assesses the model's ability to correctly identify negative samples. It gauges the model's performance in avoiding false positives and accurately recognizing negative instances. The Spe can be calculated as seen in Equation 6.

$$Spe = \frac{TN}{(TN + FP)} \quad (6)$$

5. ***F-1 Score (F1):*** The F1 is the harmonic mean of Pre and Rec. It is particularly useful when there is an imbalance between positive and negative samples, as it balances the trade-off between precision and recall. The F1 can be calculated as seen in Equation 7.

$$F1\ Score = \frac{(2 \times Pre \times Rec)}{(Pre + Rec)} \quad (7)$$

3. Experiments

The experiments conducted to evaluate the performance of the ViT model are presented in this section. Additionally, the following sections present the analyses of the experimental results, along with the performance metrics.

3.1 Experimental Setups

In this study, we used the ViT-B/16 model, which has a resolution of 224×224 pixels and was pre-trained on the ImageNet [35] dataset. The ViT-B models include a hidden size of 768, an MLP size of 3072, and an overall parameter count of 86 million [30]. The input-output shapes and trainability of the layers of the model are summarized in Table 2.

Table 2 Details of the ViT model

Count	Layer	Input Shape	Output Shape	Trainable
×1	Patch Embedding	(1, 3, 224, 224)	(1, 196, 768)	True
×1	Dropout (pos_drop)	(1, 197, 768)	(1, 197, 768)	False
×1	Identity (patch_drop)	(1, 197, 768)	(1, 197, 768)	False
×1	Identity (norm_pre)	(1, 197, 768)	(1, 197, 768)	False
×12 (Encoder)	LayerNorm (norm1)	(1, 197, 768)	(1, 197, 768)	True
	Attention (attn)	(1, 197, 768)	(1, 197, 768)	True
	Identity (ls1)	(1, 197, 768)	(1, 197, 768)	False
	Identity (drop_path1)	(1, 197, 768)	(1, 197, 768)	False
	LayerNorm (norm2)	(1, 197, 768)	(1, 197, 768)	True
	Mlp (mlp)	(1, 197, 768)	(1, 197, 768)	True
	Identity (ls2)	(1, 197, 768)	(1, 197, 768)	False
	Identity (drop_path2)	(1, 197, 768)	(1, 197, 768)	False
×1	LayerNorm (norm)	(1, 197, 768)	(1, 197, 768)	True
×1	Identity (fc_norm)	(1, 768)	(1, 768)	False
×1	Dropout (head_drop)	(1, 768)	(1, 768)	False
×1	Linear (head)	(1, 768)	(1, 26)	True

The model was implemented using the timm library. During model training, 11,380 samples were randomly selected from the dataset, representing 80% of the total dataset samples. The remaining samples were divided equally to be used for the validation and testing phases. A visual representation of the proposed experimental setup framework is shown in Figure 5.

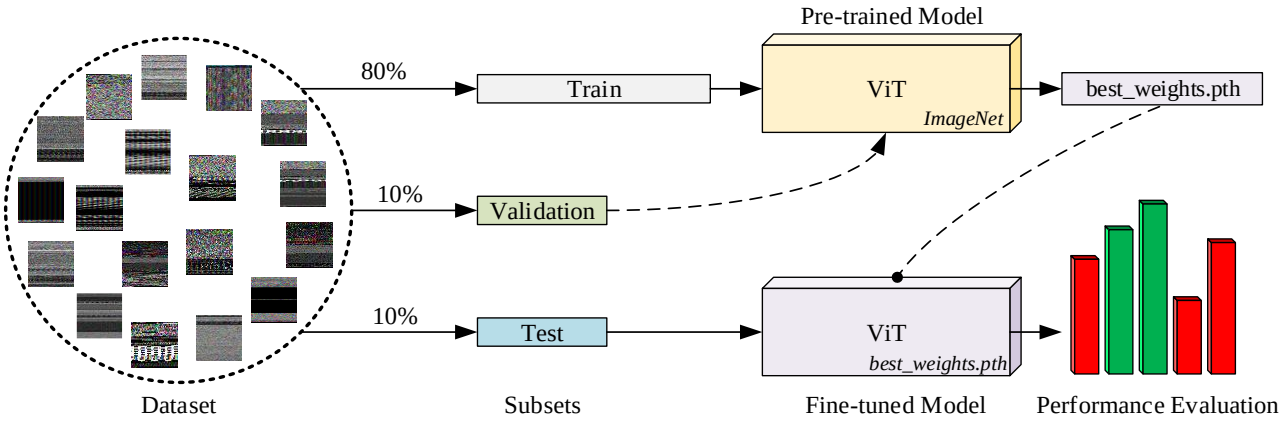


Figure 5 The proposed experimental setup framework

The training set was used to learn the parameters of the model, while the validation set was used to adjust the hyper-parameters and check for overfitting. The training and test samples were included in the training with a batch size of 32. The learning rate of the model was set to 0.00002 and the AdamW optimizer was used. The maximum number of epochs defined for the training and validation processes is 100. The CrossEntropyLoss function was used to calculate the loss value at each epoch. The software, hyperparameters and library versions used in the study are summarized in Table 3.

Table 3 Experimental environment and parameters

Name	Type	Version / Value
Python	Programming Language	3.10.11
Timm	Library	0.9.2
Torch	Library	2.0.1
Torchvision	Library	0.15.2
Transformers	Library	4.32.1
Batch Size	Hyperparameter	32
Learning Rate	Hyperparameter	0.00002
Max Epoch	Hyperparameter	100
Epsilon	Hyperparameter	0.000001

When training the ViT model with ImageNet weights, an early stopping function is defined to monitor the training phase. This function monitors the validation accuracy value and stops training if there is no improvement for twenty consecutive epochs. In this way, the weights of the epoch with the highest validation accuracy value were saved as 'best_weights.pth'. The best_weights.pth values were transferred to the ViT model, and the fine-tuned model was obtained. The fine-tuned ViT model was given test examples as input, which the model had never encountered before, and the performance of the model was determined by analyzing the predictions obtained.

3.2 Results

The ViT model was trained in the Google Colab environment using samples from the MaleVis dataset. By using optimized ImageNet weights as initial parameters, rather than random weights, the model achieved high accuracy rates in a relatively short time. The early stopping function set the validation accuracy value of 97.78% obtained in the 37th epoch as the stopping point, as there was no improvement in the subsequent twenty epochs. The weights obtained were saved for use during the test phase. The graphs showing the performance curves of the ViT model during the validation and training phases are shown in Figure 6.

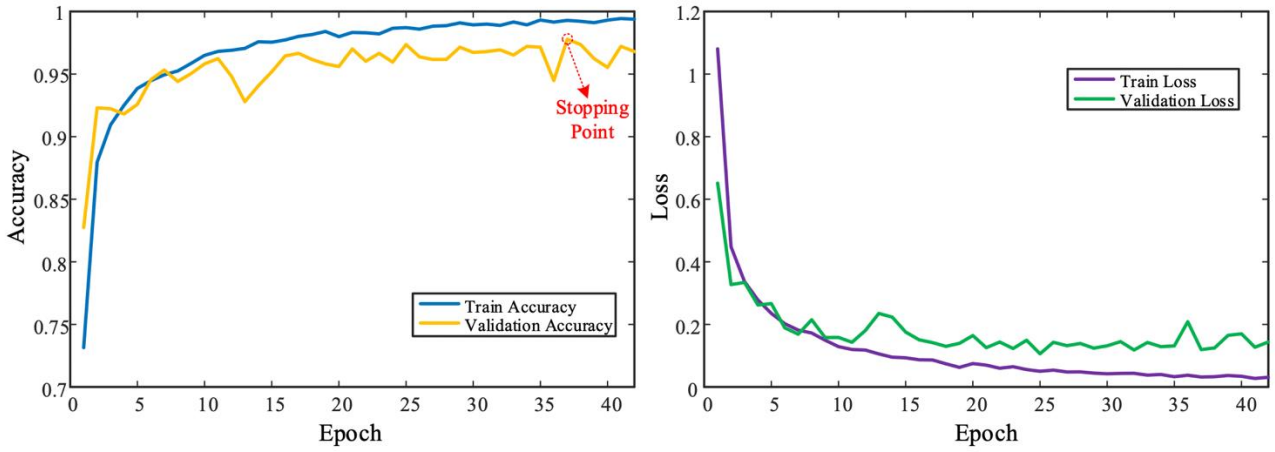


Figure 6 Graphs of the training and validation performance

Using the stopping point weights of the model, predictions were analyzed on randomly selected samples from the validation dataset. These predictions and their class-based probabilities are given in Figure 7.

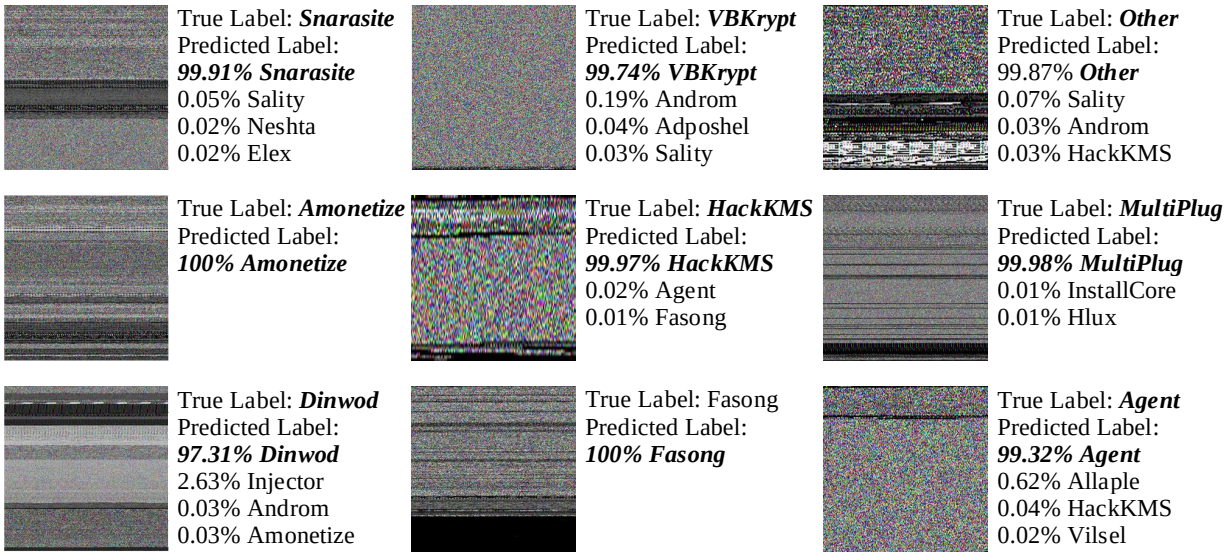


Figure 7 The validation predictions and their class-based probabilities

As the accuracy of the ViT model was at the desired level during training, the weights were transferred to the model. Test images were used as input to the model to verify the robustness of the model. The model achieved an accuracy rate of 98.80% with only 17 misclassifications on 1423 test images. The confusion matrix resulting from the model's predictions on the test images is shown in Figure 8. When analyzing the test results of the model, it can be seen that the predicted class for most of the misclassifications is 'Other'. The main reason for this is that the features of the 'Other' class, which contains more examples than other classes, are dominant.

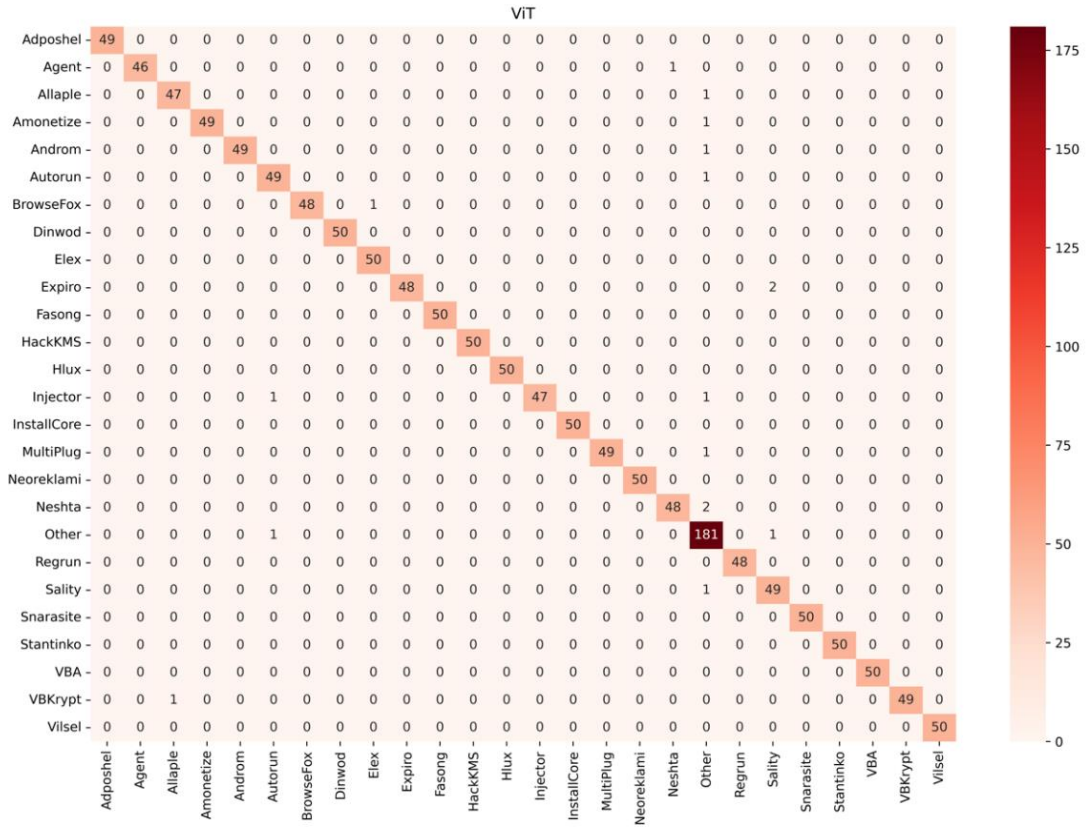


Figure 8 The confusion matrix for the test dataset

The block plots illustrating the class-based values of the performance metrics obtained during the test phase are presented in Figure 9.

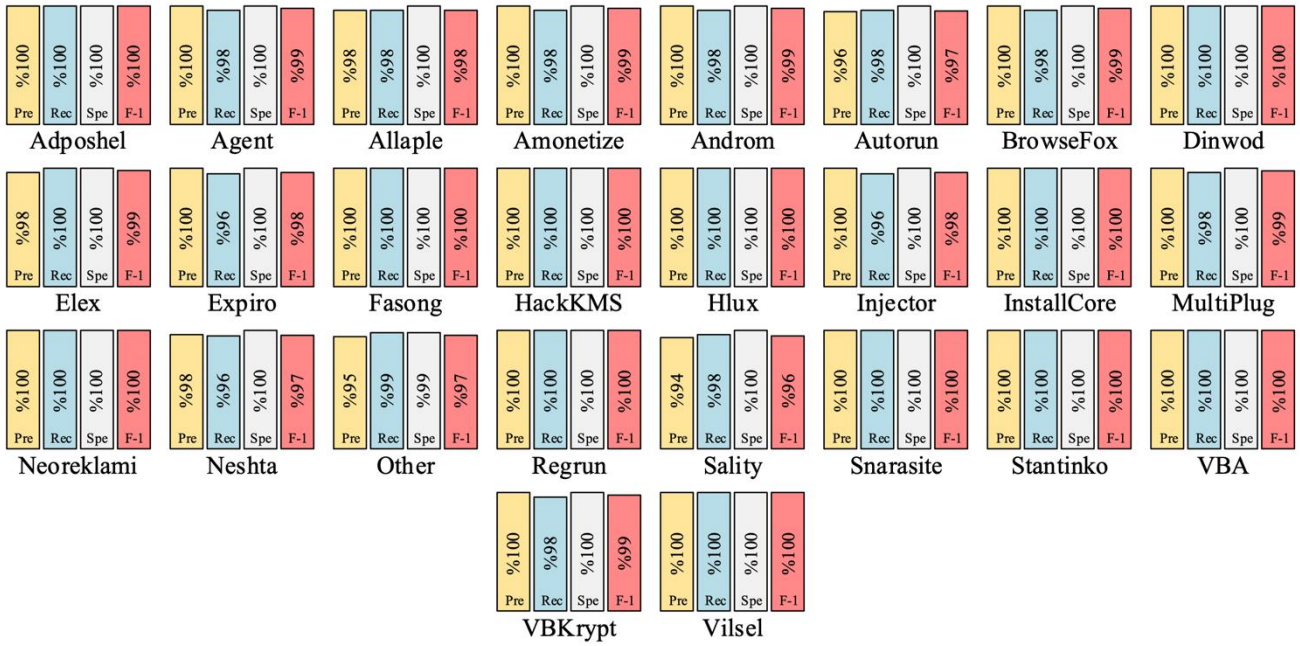


Figure 9 Class-based performance values obtained from the test samples

When analyzing the experimental results, it is observed that the Pre rate reaches 100% for twenty classes. Similarly, the remaining six classes are also characterized by high Pre values. The lowest Pre rate was observed in the Sality class with a

value of 94.23%. Of the classes, only thirteen achieved a Rec rate of 100%, while the lowest recall value was obtained in the Injector class, with a rate of 95.91%. In terms of the Specificity metric, which boasts the highest average percentage, 25 classes reached a specificity of 100%. The Other class is the sole exception, exhibiting a specificity value that is distinctively lower than that of the other classes. Upon scrutinizing the class-based F-1 scores, it becomes evident that 25 classes exhibit scores surpassing 97%. However, the Sality class displays the lowest F-1 score at 96%. These statistical findings collectively demonstrate the model's proficient performance in the multiclass classification task, effectively identifying various classes.

4. Discussion

In the field of cybersecurity, the accurate classification of malware images is of paramount importance, serving as a central tool for identifying and mitigating a wide range of digital threats. While the classification of malware images has traditionally involved labor-intensive processes, the integration of machine learning methods to automate fundamental tasks has become essential in various domains, including information security. Leveraging advances in deep learning, a path pioneered by CNNs, the field continues to flourish with a number of novel architectures, each contributing to the evolving threat detection landscape. This study presents an implementation of ViTs, which have recently gained popularity, to classify malware with high accuracy. Table 4 lists similar research studies with the same dataset. Patil et al. [36] proposed a novel approach for malware image classification using machine learning, achieving accuracy rates of 93.00% for RF, 93.70% for EfficientNet-B0, and 92.00% for VGG-16 models. Ilyas and Mohammad [37] proposed a method for malware image classification. Their approach incorporated the employment of MobileNetV2, InceptionV3, ResNet50, and LittleVGG architectures. Notably, MobileNetV2 exhibited high performance compared to the other models, achieving an accuracy rate of 95.19%. Fathurrahman et al. [38] introduced a lightweight CNN model designed for malware image classification in IoT applications, particularly suitable for embedded systems. The proposed model achieved an average accuracy of 96.22%. Atitallah et al. [39] proposed a novel vision-based approach for classifying IoT malware images, utilizing deep transfer learning with ensembling strategies. The proposed approach, which fuses ResNet18, MobileNetV2, and DenseNet161 CNNs using an RF voting strategy, achieves exceptional performance with an accuracy of 98.68%.

Table 4 Comparison of our work with studies developed on the same dataset

Study	Year	Architecture	Model	Performance
Patil et al. [36]	2021	CNN	EfficientNet-B0	Acc = 93.70%
Iyas and Mohammad [37]	2021	CNN	MobileNetV2	Acc = 95.19%
Fathurrahman et al. [38]	2022	CNN	Custom CNN	Acc = 96.22%
Atitallah et al. [39]	2022	CNN	ResNet18+MobileNetV2+DenseNet161	Acc = 98.68%
The proposed study	2023	Transformer	ViT-B/16	Acc = 98.80%

In this study, the ViT model was used to classify malware images, achieving an impressive accuracy rate of 98.80% across 26 different classes. This performance surpasses the accuracy rates reported in other studies listed in Table 4. The ViT model's superiority can be attributed to its enhanced ability to comprehend pixel relationships, thanks to its attention mechanism, allowing it to effectively capture and represent crucial image features. However, in the context of malware classification, it's imperative to carefully consider the implications of both FP and FN classifications. FP can trigger unnecessary alarms or resource-intensive benign file investigations, while FN may pose significant security risks by allowing malicious files to evade detection. When examining the studies outlined in Table 4, it's evident that our research resulted in fewer FP and FN predictions. Additionally, the Sality class consistently displayed the lowest accuracy across all the studies in Table 4. By understanding the unique challenges associated with this class and exploring the potential reasons for its lower accuracy, can enhance the model's robustness and contribute to more reliable real-world malware detection systems. The ViT model's superior ability to grasp pixel relationships through its attention mechanism makes it especially advantageous for images with intricate structures, such as malware.

The advantages of our transformer-based model can be summarized as follows:

- The conceptual foundation of the proposed model is built upon ViTs, which are presently a prominent area of research. This is a pioneering work to investigate the performance of transformer-based image classifier models in the cybersecurity domain.
- Since the transfer learning method is used, the model achieves high performance values with low cost.
- The proposed model improves computational efficiency by vectorizing the input images with 16×16 patches, while minimizing important information lost in feature extraction.
- The proposed model can be fine-tuned and easily used to detect different types of malwares.

While our research demonstrates the encouraging possibilities of the ViT model in classifying malware images, it is important to acknowledge the inherent limitations and potential challenges associated with its application. Specifically, our ViT-based classifier requires a significant amount of labeled data for training, which may not be readily available in certain malware

analysis scenarios, particularly for rare or emerging malware families. Furthermore, while the ViT model has impressive accuracy in classifying malware images, its real-time response time has not been evaluated. Another consideration is the computational requirements and scalability of the ViT model. Our study primarily focuses on its classification accuracy, but it's important to recognize that deploying the ViT model for real-time malware analysis on a large scale may demand substantial computational resources. Therefore, a thorough evaluation of the computational requirements and an assessment of the feasibility of deploying the ViT model for real-time malware analysis on a large scale are necessary. However, the black box nature of the proposed method may create difficulties in understanding how it makes classification decisions. Without the ability to gain insight into the features and attributes that the ViT model uses for classification, it may be difficult to gain meaningful information about the characteristics and behavior of different malware families. These limitations highlight the need for complementary methods or tools that can provide transparency and interpretability in the context of malware analysis, offering a more comprehensive and reliable approach to cybersecurity. In future work, we will extend our efforts by exploring malware detection using alternative state-of-the-art transformer-based architectures. Additionally, we intend to develop an explainable method that visualizes the specific pixel areas to which these models' pay attention in their predictions.

5. Conclusions

In this study, we proposed a ViT model designed for the automated classification of malware images. The proposed model is trained and validated on a public dataset with 26 different classes consisting of 14,226 samples. The ViT model with ImageNet weights is fine-tuned on malware images. As a result of the training phase using the early stopping function, the weights of the epoch with the highest validation accuracy value were recorded and implemented to the model. This model achieved 98.80% accuracy on test images it had never seen before. When analyzing the model's predictions on the test images, it can be seen that all performance metrics reach 100% for 12 classes. The performance values obtained for the other classes are also quite high. However, the model showed the lowest classification performance on the Sality class samples. The main limitation of the study is that it does not evaluate the performance of real-time applications. The proposed model can perform the classification of malware images automatically and can be effectively used by experts due to its high accuracy rates. Moreover, the proposed model can be easily fine-tuned for similar tasks to achieve high performance with low training costs.

References

[1] M. Wazid, A. K. Das, J. J. P. C. Rodrigues, S. Shetty, and Y. Park, "IoMT malware detection approaches: analysis and research challenges," *IEEE Access*, vol. 7, pp. 182459–182476, 2019.

[2] A. Chakraborty, A. Biswas, and A. K. Khan, "Artificial intelligence for cybersecurity: Threats, attacks and mitigation," *arXiv preprint arXiv:2209.13454*, 2022.

[3] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics (Basel)*, vol. 12, no. 6, p. 1333, 2023.

[4] C. S. Yadav *et al.*, "Malware analysis in iot & android systems with defensive mechanism," *Electronics (Basel)*, vol. 11, no. 15, p. 2354, 2022.

[5] M. Z. Hasan, M. Z. Hussain, and Z. Ullah, "Computer viruses, attacks, and security methods," *Lahore Garrison University Research Journal of Computer Science and Information Technology*, vol. 3, no. 3, pp. 20–25, 2019.

[6] B. S. Rawal, G. Manogaran, and A. Peter, "Malware," in *Cybersecurity and Identity Access Management*, Springer, 2022, pp. 103–116.

[7] P. M. Datta, "Cybersecurity threats: Malware in the code," in *Global Technology Management 4.0: Concepts and Cases for Managing in the 4th Industrial Revolution*, Springer, 2022, pp. 155–170.

[8] K. Geldenhuys, "Spyware: Spying on everything you do," *Servamus Community-based Safety and Security Magazine*, vol. 114, no. 10, pp. 15–17, 2021.

[9] M. Agrawal, K. D. S. Mann, R. Johari, and D. P. Vidyarthi, "Cyber Risks and Security—A Case Study on Analysis of Malware," in *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 3*, Springer, 2022, pp. 339–349.

[10] S. Thakur, S. Chaudhari, and B. Joshi, "Ransomware: Threats, identification and prevention," *Cyber Security and Digital Forensics*, pp. 361–387, 2022.

[11] S. Li, Q. Zhou, R. Zhou, and Q. Lv, "Intelligent malware detection based on graph convolutional network," *J Supercomput*, vol. 78, no. 3, pp. 4182–4198, 2022.

[12] A. Razgallah, R. Khoury, S. Hallé, and K. Khanmohammadi, "A survey of malware detection in Android apps: Recommendations and perspectives for future research," *Comput Sci Rev*, vol. 39, p. 100358, 2021.

[13] N. Galloro, M. Polino, M. Carminati, A. Continella, and S. Zanero, "A Systematical and longitudinal study

of evasive behaviors in windows malware," *Comput Secur*, vol. 113, p. 102550, 2022.
[14] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020.
[15] Y. Yang *et al.*, "GooseBt: A programmable malware detection framework based on process, file, registry, and COM monitoring," *Comput Commun*, vol. 204, pp. 24–32, 2023.
[16] U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, "Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier," *Sci Rep*, vol. 12, no. 1, p. 15647, 2022.
[17] B. Y. Sathwara, "A hybrid approach based on boosting algorithm for effective android malware detection," *International Journal of Computing and Digital Systems*, vol. 13, no. 1, pp. 189–206, 2023.
[18] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications*, vol. 47, pp. 377–389, 2019.
[19] S. Alrabaee, M. Debbabi, and L. Wang, "A survey of binary code fingerprinting approaches: taxonomy, methodologies, and features," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–41, 2022.
[20] H. Naeem *et al.*, "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, p. 102154, 2020.
[21] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based android malware detection and variant classification," *Comput Intell*, vol. 38, no. 5, pp. 1748–1771, 2022.
[22] S. H. Khan *et al.*, "A new deep boosted CNN and ensemble learning based IoT malware detection," *Comput Secur*, p. 103385, 2023.
[23] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A malware detection approach using autoencoder in deep learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.
[24] M. Asam *et al.*, "IoT malware detection architecture using a novel channel boosted and squeezed CNN," *Sci Rep*, vol. 12, no. 1, p. 15498, 2022.
[25] S. Kumar and B. Janet, "DTMIC: Deep transfer learning for malware image classification," *Journal of Information Security and Applications*, vol. 64, p. 103063, 2022.
[26] Z. Lu, S. Liang, Q. Yang, and B. Du, "Evolving block-based convolutional neural network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–21, 2022.
[27] M. Chen *et al.*, "Searching the search space of vision transformer," *Adv Neural Inf Process Syst*, vol. 34, pp. 8714–8726, 2021.
[28] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Adv Neural Inf Process Syst*, vol. 34, pp. 23296–23308, 2021.
[29] A. S. Bozkir, A. O. Cankaya, and M. Aydos, "Utilization and comparision of convolutional neural networks in malware recognition," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2019, pp. 1–4.
[30] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[31] J. Wu, R. Hu, Z. Xiao, J. Chen, and J. Liu, "Vision Transformer-based recognition of diabetic retinopathy grade," *Med Phys*, vol. 48, no. 12, pp. 7850–7863, 2021.
[32] P. S. Thakur, P. Khanna, T. Sheorey, and A. Ojha, "Explainable vision transformer enabled convolutional neural network for plant disease identification: PlantXViT," *arXiv preprint arXiv:2207.07919*, 2022.
[33] Y. Wu, S. Qi, Y. Sun, S. Xia, Y. Yao, and W. Qian, "A vision transformer for emphysema classification using CT images," *Phys Med Biol*, vol. 66, no. 24, p. 245016, 2021.
[34] S. Illium, R. Müller, A. Sedlmeier, and C.-L. Popien, "Visual transformers for primates classification and covid detection," *arXiv preprint arXiv:2212.10093*, 2022.
[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
[36] S. Patil *et al.*, "Improving the robustness of ai-based malware detection using adversarial machine learning," *Algorithms*, vol. 14, no. 10, p. 297, 2021.
[37] I. Alodat and M. Alodat, "Detection of Image Malware Steganography Using Deep Transfer Learning Model," in *Proceedings of International Conference on Data Science and Applications: ICDSA 2021, Volume 2*, Springer, 2021, pp. 323–333.

[38] A. Fathurrahman, A. Bejo, and I. Ardiyanto, "Lightweight convolution neural network for image-based malware classification on embedded systems," in *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, IEEE, 2022, pp. 12–16.

[39] S. Ben Atitallah, M. Driss, and I. Almomani, "A novel detection and multi-classification approach for IoT-malware using random forest voting of fine-tuning convolutional neural networks," *Sensors*, vol. 22, no. 11, p. 4302, 2022.

Conflict of Interest Notice

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

Ethical Approval and Informed Consent

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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 36-45
DOI: 10.35377/saucis...1391636
RESEARCH ARTICLE

Classification of Electronics Components using Deep Learning Methods

Emel Soylu [1], İbrahim Kaya [2]
[1] Department of Software Engineering, Faculty of Engineering, Samsun University, Samsun, Türkiye
[2] Department of Software Engineering, Graduate School, Samsun University, Samsun, Türkiye

Corresponding author:
Emel Soylu, Department of Software Engineering, Faculty of Engineering, Samsun University, Samsun, Türkiye
emel.soylu@samsun.edu.tr

Article History:
Received: 16.11.2023
Accepted: 30.01.2024
Published Online: 30.01.2024

ABSTRACT

In this study, we present an electronic component classification system with a classification accuracy exceeding 98%, achieved by utilizing state-of-the-art deep learning architectures. We employed EfficientNetV2B3, EfficientNetV2S, EfficientNetB0, InceptionV3, MobileNet, and Vision Transformer (ViT) models for the classification task. Our dataset comprises various electronic components, and it has been meticulously organized and labeled to provide high-quality training data. We conducted extensive experiments, utilizing data augmentation techniques and transfer learning, to fine-tune and optimize the models for the given task. The high classification accuracy achieved by our system indicates its readiness for real-world applications. It can be applied to advance automation and efficiency in the electronics industry.

Keywords: Electronic component classification, Deep learning, Transfer learning

1. Introduction

Fundamental to electronic circuitry and systems, electronic components represent indispensable units, meticulously engineered to fulfill precise functionalities. They are classified into two primary categories: active components, exemplified by transistors and diodes, and passive components, encompassing resistors and capacitors. These elements play pivotal roles in the processing, storage, and transmission of electrical signals [1]. Resistors, for instance, regulate the flow of electric current, often used to control voltage and current within a circuit. Capacitors, on the other hand, store and release electrical energy, proving valuable for tasks like energy storage and timing. Transistors are versatile, functioning as amplifiers or switches for electronic signals and serving as the backbone of modern electronics, including amplifiers and processors. LEDs (Light Emitting Diodes) are ubiquitous for their light emission when current flows through them, commonly applied in displays, indicators, and lighting. Potentiometers, or variable resistors, are useful for tasks like volume control and tuning in electronic devices. Buttons, which also go by the name of switches, control the electrical current's flow, often used for user input and control. In addition, ultrasonic sensors make use of sound waves to measure distance or detect objects. They have applications in robotics, automotive systems, and distance measurement. These components are deployed across various industries, from consumer electronics and automotive systems to industrial automation, telecommunications, medical devices, aerospace and defense, and renewable energy solutions. In essence, electronic components are the foundational elements that power the world of modern technology, enabling the development of advanced electronic devices and systems that have revolutionized everyday life and various industrial sectors [2]–[10].

Image classification is the process of assigning a specific class to an image, and within this domain, various techniques are employed [11]. Deep learning, particularly leveraging architectures such as Convolutional Neural Networks (CNNs), stands out as a robust approach [12]. Additionally, machine learning algorithms like Support Vector Machines (SVM) [13], decision trees [14], and random forests [15], as well as straightforward methods like K-Nearest Neighbors [16], are commonly utilized. Image features and descriptors, including color histograms, edge detectors, and Histogram of Oriented Gradients (HOG), contribute to the diverse array of methods. These techniques are often combined or customized based on factors such as dataset size, complexity, and specific requirements. The selection of a particular method is contingent upon the distinct usage scenarios and objectives.

Deep learning, a subfield of machine learning, involves training artificial neural networks with multiple layers to perform complex tasks. Its importance lies in its remarkable ability to automatically learn and extract intricate patterns and representations from large datasets, enabling the development of highly accurate predictive models. Deep learning has found diverse applications, one of which is in classification. It is used to categorize and identify objects or data, such as images, audio, or text, in various domains. For instance, in computer vision, deep learning is employed for image recognition, object detection, and facial recognition. In natural language processing, it aids in sentiment analysis, language translation, and chatbot development. Deep learning also has applications in healthcare for disease diagnosis, in autonomous vehicles for object detection and navigation, in finance for fraud detection, and in manufacturing for quality control. Its capacity to handle large and complex datasets makes deep learning a transformative technology with wide-ranging implications for automation, precision, and decision-making across industries [17]–[20]. Deep learning methods for electronic component classification involve the use of advanced neural networks, such as CNNs and transformers, to categorize electronic components based on their visual attributes and features. These methods are particularly valuable in automating the identification and sorting of electronic components, which can vary significantly in size and appearance. They are widely employed in quality control processes in electronics manufacturing, ensuring that the correct components are used in assembly. Additionally, this technology finds applications in inventory management, making it easier to track and manage the vast array of components used in various products. The importance of deep learning in this context lies in its ability to achieve high accuracy and speed in classification, reducing human error and increasing efficiency in the electronics industry. It also paves the way for the automation of tedious and time-consuming tasks, allowing human resources to be redirected to more complex and creative aspects of electronic design and production [21]–[28].

Our study focused on an extensive comparison of state-of-the-art deep learning models, including EfficientNet-V2B3 [29], EfficientNet-V2S [30], EfficientNet-B0 [31] , Inception-V3 [32], MobileNet [33], and Vision Transformer (ViT) [34], in the realm of electronic component classification. We evaluated their performance across various electronic component classes, such as capacitor, LED, potentiometer, button, resistor, transistor, and ultrasonic sensor. The significance of this research lies in its potential to bring about a significant transformation in the electronics industry by providing a robust and highly accurate automated solution for classifying electronic components. Such a system has the capacity to greatly enhance quality control, reduce errors, and expedite manufacturing processes. Furthermore, the unique value of our work is evident in its thorough examination of these advanced models in a practical, industrial context, highlighting their real-world applicability. By demonstrating the capabilities of these models in achieving exceptional accuracy in component classification, we contribute to the ongoing efforts aimed at advancing automation, efficiency, and precision in electronic component management, offering a compelling pathway to redefine modern electronics manufacturing. In the subsequent sections of this study, we will delve into the existing body of work within this domain, the dataset employed, the methodology employed, the experimental endeavors, and the outcomes obtained. We aim to provide a comprehensive overview of related research, illuminate the specifics of our dataset, elucidate the methods applied, chronicle our experimental investigations, and ultimately present the findings and results that have emerged from our efforts.

2. Relevant Work

The recognition of electronic components has been extensively studied, with methodologies that integrate image processing, machine learning, and deep learning techniques. Image processing methods involve the use of edge detection algorithms to outline the contours and edges of electronic components, with color and intensity analysis playing a crucial role, especially in identifying components on printed circuit boards. Machine learning approaches such as SVM leverage component features for classification, and decision trees and forests are employed for effective feature extraction. Deep learning methodologies, particularly CNN, demonstrate effectiveness, especially in the recognition of components on printed circuit boards. Transfer learning, utilizing pre-trained models from extensive datasets, enhances component recognition performance, even with smaller datasets. Object detection methods like R-CNN and its derivatives, as well as YOLO (You Only Look Once), offer effective strategies for recognizing components within images [28]. Tailored methods, specific to component characteristics, involve geometry analysis and Optical Character Recognition (OCR) for labels or text on components. This dynamic field continues to evolve, holding substantial potential, particularly in applications such as industrial automation, electronic manufacturing, and maintenance [21]–[28]. Table 1 represents various studies in the literature, each detailing their dataset, task, methodology, and the achieved results. For instance, reference [21] employed the ERFAM-YOLOv3 method for object detection on a dataset consisting of 1000 images with 29 instrument categories, achieving a notable 95.03% average accuracy. Similarly, other references provide insights into different approaches and outcomes in the field of electronic component recognition. According to the values provided in this table, the performance rates vary within the range of 90% to 100%.

3. Dataset

The images of various electronic components, including capacitors, LEDs, potentiometers, push buttons, resistors, transistors, and ultrasonic sensors, were collected from publicly available datasets for the purpose of classification in a research project. These datasets contain visual representations of these components, which are essential in electronic circuitry and various applications [35]–[37]. The dataset also includes images that we captured ourselves and images obtained from Google image search. Figure 1 illustrates randomly chosen samples within the dataset. Table 2 shows the number of training and testing

samples for each component category in the study. The dataset has been partitioned with approximately 25% of the total data allocated for testing and 75% for training purposes. Since the dataset contains images of varying quality and from different perspectives, and it possesses enough data for classification, no augmentation process was performed. In Figure 2, a block diagram illustrating the data set preparation process is provided.

Table 1 Relevant work

Ref.	Dataset	Task	Method	Result
[23]	483 images, 5000 labeled IC instances	Object detection	VN-Siamesev2 network containing the backbone of VGG16 architecture	92.31% accuracy
[21]	1000 images, 29 instrument categories, 182900 electronic components	Object detection	ERFAM-YOLOv3	95.03%average accuracy
[22]	8000 images	Object detection	ECLAD-Net	90%-100%
[24]	60 images, 172 labeled components	Object detection	Image processing	91.28%
[25]	3094 images	Classification	Siamese network	99%
[26]	200340 images	Classification	Multilayer perceptron	92.3%
[27]	-	Classification	Back Propagation Neural Network	95.8%
[28]	1026 images, 4 categories	Object detection	YOLOv2 Network	0.27 error rate on test set 0.8743% on evaluation set

Figure 1 Sample images from the dataset.

4. Method

We used transfer learning based deep learning models in our study. Transfer learning in the context of deep learning refers to the practice of leveraging a pre-trained neural network model for a new, related task. It's a technique where a model developed for a particular task is adapted for a second related task. Transfer learning can significantly speed up the training process and often leads to better performance compared to training a model from scratch. Transfer learning typically involves starting with a pre-trained model that has been trained on a large dataset for a similar or related problem. These models are often trained on massive datasets and have learned useful features from them. After obtaining a pre-trained model, you fine-tune it for your specific task. Training deep neural networks from scratch can be computationally expensive and time-

consuming, especially when dealing with large datasets and complex architectures. Transfer learning allows you to start with a pre-trained model, saving a significant amount of training time.

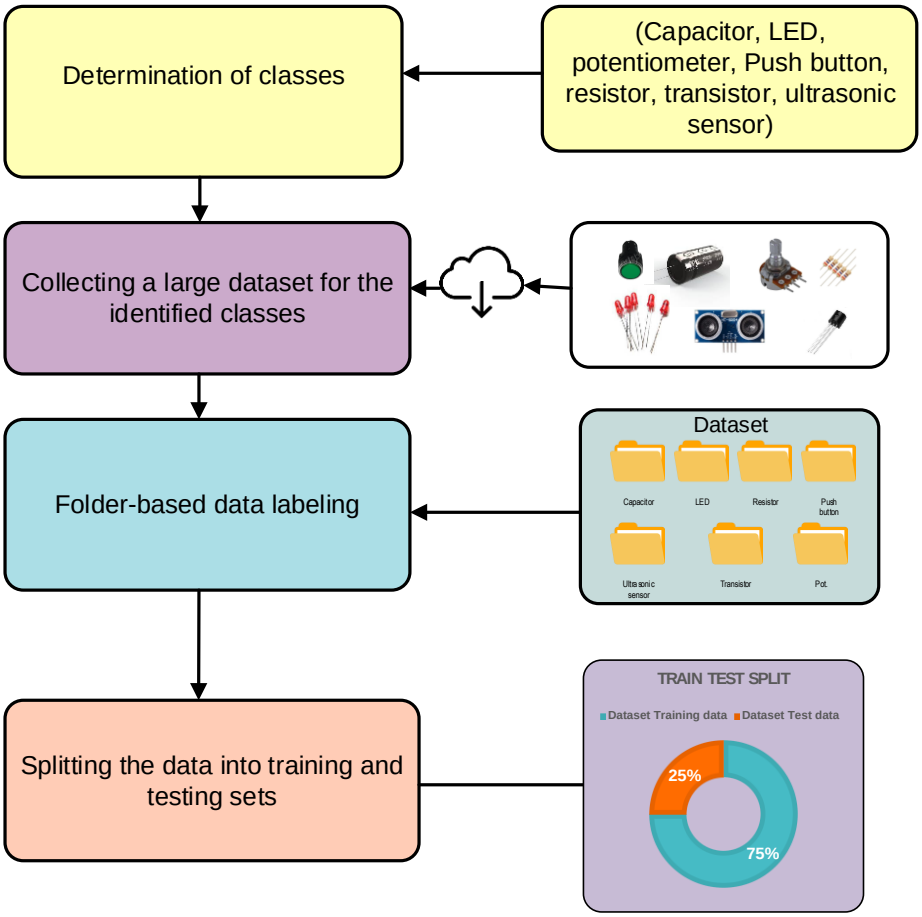


Figure 2 Dataset preparation diagram

Table 2 Component Classification Data: Training and Testing Split

Components	**Train**	**Test**
Capacitor	600	200
Led	375	100
Poentiometer	302	100
Push Button	301	100
Resistor	355	100
Transistor	281	100
Ultrasonic Sensor	300	100
Total	2514	800

We have leveraged a selection of pre-trained CNN architectures, including EfficientNet-V2B3, EfficientNet-V2S, EfficientNet-B0, Inception-V3, MobileNet, and Vision transformer based neural network (ViT), for the purpose of electronic component classification. These models can be readily accessed in Keras, an open-source neural network library written in Python [38].

Inception, also recognized as GoogleNet, represents a deep learning architecture tailored for CNNs, meticulously crafted to tackle the complexities of training exceptionally deep networks without compromising computational efficiency. Pioneered by researchers at Google, Inception introduces an ingenious 'inception module' that integrates multiple convolutional filter

sizes and pooling operations within a single layer. This innovative approach empowers the network to capture features across various scales, thereby enhancing the robustness and precision of feature extraction. Inception has exerted a profound influence on the domain of computer vision, particularly in tasks such as image classification and object detection. Its aptitude for harmonizing model depth with computational efficiency has solidified its status as a widely adopted architectural solution in the realm of deep learning [39].

EfficientNet is a family of deep learning models specifically designed to achieve state-of-the-art performance with high efficiency in terms of computational resources. These models use a novel scaling method that uniformly scales the network's depth, width, and resolution. This approach ensures that the model adapts to different computational constraints while maintaining excellent performance on a wide range of computer vision tasks, such as image classification and object detection. EfficientNet's architecture efficiently balances model size and accuracy, making it a popular choice for various real-world applications where computational efficiency is a priority, such as edge devices and resource-constrained environments [40].

MobileNet is a CNN architecture designed for efficient and lightweight deep learning applications, particularly optimized for mobile and edge computing devices. Introduced by Google researchers in 2017, MobileNet addresses the challenge of deploying complex neural networks on resource-constrained platforms. It achieves computational efficiency through the use of depth wise separable convolutions, a key architectural element that significantly reduces the number of parameters and computations required. The network's core idea is to factorize a standard convolution into a depth wise convolution and a 1x1 pointwise convolution. The depth wise convolution applies a single filter per input channel, followed by a 1x1 pointwise convolution that combines the outputs from the depth wise convolution. This separation of spatial and channel-wise filtering allows MobileNet to maintain a good balance between accuracy and computational efficiency. With its lightweight design, MobileNet has become a popular choice for real-time image classification and object detection tasks on devices with limited computational resources.

The Vision Transformer (ViT) represents a groundbreaking architecture in the realm of computer vision and image processing. Introduced in a seminal paper by researchers from Google in 2020, ViT diverges from conventional CNN structures by exclusively relying on self-attention mechanisms. The architecture leverages the Transformer model, originally designed for natural language processing, to capture intricate hierarchical features within images. In ViT, the input image is divided into fixed-size non-overlapping patches, which are linearly embedded and flattened into sequences. These sequences serve as input tokens for the Transformer encoder, allowing the model to attend to relationships between different image patches. This mechanism enables ViT to grasp both local and global contextual information, crucial for understanding complex visual patterns. Additionally, ViT employs positional embeddings to preserve spatial information within the flattened sequences. Notably, ViT has demonstrated exceptional performance on various computer vision tasks, often surpassing traditional CNNs. Its remarkable ability to scale to large datasets and capture long-range dependencies positions ViT as a versatile architecture for vision-based applications, showcasing its potential impact on the evolution of deep learning models for image understanding.

The metrics employed for the comparative assessment of the performance of these architectures include Accuracy (Acc.), Precision, Recall, and F1 Score values. These metrics serve as quantitative indicators to evaluate the effectiveness and capabilities of the different models in a rigorous and systematic manner. Precision measures the model's ability to accurately identify positive instances among the instances it predicts as positive as given in Eq.1. Recall, also known as the true positive rate or sensitivity, assesses the model's ability to correctly identify all positive instances, as defined in Equation 2. The F1 score, a harmonic mean of precision and recall, strikes a balance between precision and recall, proving valuable for imbalanced datasets according to Equation 3. Accuracy, measured by the formula in Equation 4, evaluates the overall correctness of the model's predictions. A block diagram of the deep learning-based classification system is given in Figure 3.

$$\boldsymbol{P} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \tag{1}$$

$$\boldsymbol{R} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \tag{2}$$

$$\boldsymbol{F1\ Score} = \boldsymbol{2} \cdot \frac{\boldsymbol{P \cdot R}}{\boldsymbol{P+R}} \tag{3}$$

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \tag{4}$$

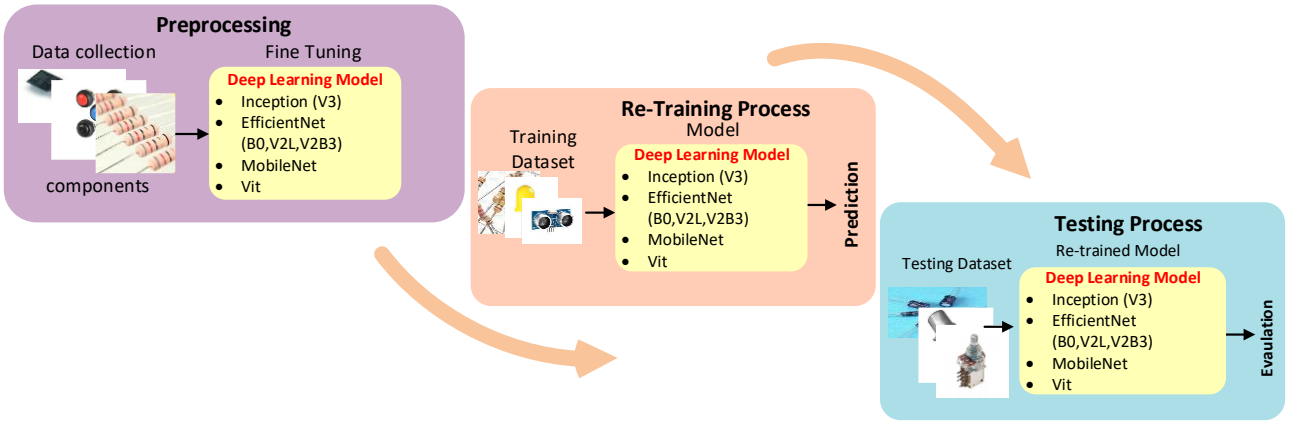


Figure 3 Block diagram of the deep learning-based classification system

5. Results

Table 3 presents the training performance metrics for various base models used in the first approach. The metrics include training accuracy, acc-loss (accuracy loss), validation accuracy, and val-loss (validation loss). The presented data illustrates the training accuracy, accuracy loss, validation accuracy, and validation loss for each base model after 100 epochs. Notably, the EfficientNet-0V2S model achieved the highest training accuracy of 99.50%, with a relatively low accuracy loss of 0.0234. However, each model's performance is comprehensively evaluated based on both training and validation metrics, providing a comprehensive overview of their effectiveness in the first approach. Table 4 presents the performance metrics of re-trained models for the first approach, considering various classification tasks. The metrics include accuracy (Acc.), precision, recall, F1 score, and overall accuracy. The provided table details the performance metrics for re-trained models in the first approach across various classes. For instance, the MobileNet model demonstrates high accuracy for LED classification (96.68%), while ViT achieves perfect accuracy (100%) across all classes, indicating excellent overall performance. Precision, recall, and F1 score metrics offer insights into the models' ability to correctly classify instances, providing a comprehensive evaluation of their effectiveness in differentiating electronic components. The overall accuracy metric presents a consolidated measure of each model's performance across all classes, facilitating a holistic assessment of their classification capabilities.

Table 3 Training performance table for the first approach

Base Model	Training Accuracy	Acc-loss	Validation accuracy	Val-loss
EfficientNet-V2B3	0.9876	0.0475	0.8044	0.1107
EfficientNet-V2S	0.9930	0.0210	0.8523	0.6433
EfficientNet-B0	0.9911	0.0299	0.8184	1.0668
Inception-V3	0.9892	0.0442	0.6617	2.5616
MobileNet	0.9958	0.0167	0.7143	2.4491
ViT	0.9850	0.2477	0.9621	0.41

Table 4 Performance metrics of re-trained models for the first approach

Method	Class	n truth	n classified	Acc.	Precision	Recall	F1 Score	Overall Acc.
MobileNet	Capacitor	184	200	89.26%	0.74	0.81	0.78	87.27%
	LED	99	100	99%	0.95	0.97	0.96	
	Potentiometer	87	100	98.13%	0.86	0.99	0.92	
	Button	136	100	94.01%	0.94	0.69	0.80	
	Resistor	89	100	98.63%	0.89	1.0	0.94	
	Transistor	113	100	97.13%	0.95	0.84	0.89	
	Ultrasonic Sensor	93	100	98.38%	0.90	0.97	0.93	
Inception-V3	Capacitor	153	200	89.39%	0.67	0.88	0.76	84.39%
	LED	112	100	97.88%	0.97	0.88	0.92	
	Potentiometer	102	100	97.5%	0.91	0.89	0.90	
	Button	87	100	96.38%	0.79	0.91	0.84	
	Resistor	101	100	96.63%	0.87	0.86	0.87	
	Transistor	109	100	96.63%	0.91	0.83	0.87	
	Ultrasonic Sensor	137	100	94.38%	0.96	0.70	0.81	
EfficientNet-B0	Capacitor	227	200	88.13%	0.83	0.73	0.78	85.13%
	LED	102	100	99%	0.97	0.95	0.96	
	Potentiometer	85	100	97.63%	0.83	0.98	0.90	
	Button	92	100	95.5%	0.78	0.85	0.81	
	Resistor	78	100	97%	0.77	0.99	0.87	
	Transistor	126	100	96%	0.97	0.77	0.86	
	Ultrasonic Sensor	90	100	97%	0.83	0.92	0.87	
EfficientNet-V2B3	Capacitor	153	200	88.88%	0.66	0.86	0.75	86.63%
	LED	96	100	99%	0.94	0.98	0.96	
	Potentiometer	112	100	97.75%	0.97	0.87	0.92	
	Button	110	100	97.5%	0.95	0.86	0.90	
	Resistor	105	100	98.63%	0.97	0.92	0.95	
	Transistor	144	100	94%	0.98	0.68	0.80	
	Ultrasonic Sensor	80	100	97.5%	0.80	1.0	0.89	
EfficientNet-V2S	Capacitor	80	200	85%	0.40	1.0	0..57	82.5%
	LED	106	100	97.25%	0.94	0.89	0.91	
	Potentiometer	117	100	97.13%	0.97	0.83	0.89	
	Button	144	100	94%	0.98	0.68	0.80	
	Resistor	115	100	97.88%	0.99	0.86	0.92	
	Transistor	138	100	95.25%	1.0	0.72	0.84	
	Ultrasonic Sensor	100	100	98%	0.92	0.92	0.92	
ViT	Capacitor	206	200	99%	0.99	0.97	0.98	**98.5%**
	LED	100	100	100%	1.0	1.0	1.0	
	Potentiometer	101	100	99.88%	1.0	0.99	1.0	
	Button	100	100	99.28%	0.97	0.97	0.97	
	Resistor	98	100	99.75%	0.98	1.0	0.99	
	Transistor	101	100	99.88%	1.0	0.99	1.0	
	Ultrasonic Sensor	94	100	99.25%	0.94	1.0	0.97	

6. Conclusions

In this study, we successfully implemented an accurate electronic component classification system using state-of-the-art deep learning architectures. The models, including EfficientNet-V2B3, EfficientNet-V2S, EfficientNet-B0, Inception-V3, MobileNet, and Vision Transformer (ViT), achieved a classification accuracy of over 98%. The comprehensive evaluation across various electronic component classes demonstrated the models' effectiveness in complex visual recognition tasks within the electronic components' domain. Training metrics further confirmed the models' efficiency, displaying high accuracy and minimal loss during both training and validation phases. Given the achieved high classification accuracy, we recommend considering the real-world deployment of the developed electronic component classification system. This system has the potential to significantly improve automation and efficiency in the electronics industry, particularly in tasks related to quality control, manufacturing, and inventory management. To enhance the system's generalization capability, expanding the dataset to include a wider variety of electronic components and variations in environmental conditions is advised. This expansion ensures the model's effectiveness in recognizing a broader range of components under various circumstances. As technology and industry standards evolve, continuous monitoring, feedback loops, and model updates become crucial. Regular assessments and updates are essential to ensure the system's adaptability to changing requirements and emerging technologies in the electronics sector. In conclusion, the successful implementation of this electronic component classification system opens doors for transformative applications in the electronics industry. The combination of advanced deep learning models and meticulous experimental methodologies positions this system as an asset for driving innovation, precision, and efficiency in electronic component management and manufacturing processes.

References

[1] A. A. Almubarak, "The effects of heat on electronic components," *Int. J. Eng. Res. Appl*, vol. 7, no. 5, pp. 52–57, 2017.

[2] M. Pecht, P. Lall, G. Ballou, C. Sankaran and N. Angelopoulos, "Passive components," in *Circuits, Signals, and Speech and Image Processing*, CRC Press, 2018, p. 1.

[3] Z. Fu, J. Wang, A. Bretas, Y. Ou and G. Zhou, "Measurement method for resistive current components of metal oxide surge arrester in service," *IEEE Trans. Power Deliv.*, vol. 33, no. 5, pp. 2246–2253, 2017.

[4] P. Hauptmann, N. Hoppe and A. Püttmer, "Application of ultrasonic sensors in the process industry," *Meas. Sci. Technol.*, vol. 13, no. 8, p. R73, 2002.

[5] X. D. Zhang, L. Y. Kang and W. F. Diao, "The principle of the potentiometer and its applications in the vehicle steering," in *IEEE International Conference on Vehicular Electronics and Safety, 2005.*, 2005, pp. 20–24.

[6] Y. Yang, X. Tong, L.-T. Yang, P.-F. Guo, L. Fan and Y.-C. Yeo, "Tunneling field-effect transistor: capacitance components and modeling," *IEEE Electron Device Lett.*, vol. 31, no. 7, pp. 752–754, 2010.

[7] A. De Donatis, "The Button Component," *Adv. ActionScript Components Mastering Flash Compon. Archit.*, pp. 275–293, 2006.

[8] Q. J. Harmer, P. M. Weaver and K. M. Wallace, "Design-led component selection," *Comput. Des.*, vol. 30, no. 5, pp. 391–405, 1998.

[9] B. Eisenberg, N. Gold, Z. Song and H. Huang, "What current flows through a resistor?," *arXiv Prepr. arXiv1805.04814*, 2018.

[10] W. J. Sarjeant, I. W. Clelland and R. A. Price, "Capacitive components for power electronics," *Proc. IEEE*, vol. 89, no. 6, pp. 846–855, 2001.

[11] E. Soylu, "A Deep Transfer Learning-Based Comparative Study for Detection of Malaria Disease," *Sak. Univ. J. Comput. Inf. Sci.*, vol. 5, no. 3, pp. 427–447, 2022.

[12] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *J. Big data*, vol. 6, no. 1, pp. 1–18, 2019.

[13] M. A. Chandra and S. S. Bedi, "Survey on SVM and their application in image classification," *Int. J. Inf. Technol.*, vol. 13, pp. 1–11, 2021.

[14] C.-C. Yang *et al.*, "Application of decision tree technology for image classification using remote sensing data," *Agric. Syst.*, vol. 76, no. 3, pp. 1101–1117, 2003.

[15] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi and S. Homayouni, "Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 6308–6325, 2020.

[16] J. Kim, B.-S. Kim and S. Savarese, "Comparing image classification methods: K-nearest-neighbor and support-vector-machines," in *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*, 2012, pp. 133–138.

[17] J. Schmidhuber, "Deep learning," *Scholarpedia*, vol. 10, no. 11, p. 32832, 2015.

[18] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, p. 436, May 2015.

[19] N. Rusk, "Deep learning," *Nat. Methods*, vol. 13, no. 1, p. 35, 2016.

[20] L. C. Yan, B. Yoshua and H. Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] J. Li, W. Li, Y. Chen and J. Gu, "A PCB Electronic Components Detection Network Design Based on Effective Receptive Field Size and Anchor Size Matching," *Comput. Intell. Neurosci.*, vol. 2021, p. 6682710, 2021.

[22] M. A. Mallaiyan Sathiaseelan, O. P. Paradis, S. Taheri and N. Asadizanjani, "Why is deep learning challenging for printed circuit board (pcb) component recognition and how can we address it?," *Cryptography*, vol. 5, no. 1, p. 9, 2021.

[23] M. A. Reza, Z. Chen and D. J. Crandall, "Deep neural network--based detection and verification of microelectronic images," *J. Hardw. Syst. Secur.*, vol. 4, no. 1, pp. 44–54, 2020.

[24] A. Bhattacharya, S. Roy, N. Sarkar, S. Malakar and R. Sarkar, "Circuit Component Detection in Offline Handdrawn Electrical/Electronic Circuit Diagram," in *2020 IEEE Calcutta Conference (CALCON)*, 2020, pp. 80–84.

[25] Y. Cheng, A. Wang and L. Wu, "A Classification Method for Electronic Components Based on Siamese Network," *Sensors*, vol. 22, no. 17, 2022.

[26] D. Lefkaditis and G. Tsirigotis, "Intelligent optical classification system for electronic components," *Elektron. ir Elektrotechnika*, vol. 2, no. 2, pp. 10–14, 2010.

[27] Y. J. Wang *et al.*, "An Artificial Neural Network to Support Package Classification for SMT Components," *2018 3rd Int. Conf. Comput. Commun. Syst. ICCCS 2018*, pp. 173–177, 2018.

[28] J. Huang and Y. Lu, "A Method for Identifying and Classifying Resistors and Capacitors Based on YOLO Network," in *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, 2019, pp. 1–5.

[29] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour and N. A. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE access*, vol. 9, pp. 14078–14094, 2021.

[30] Y. Chen, H. Liang and S. Pang, "Study on small samples active sonar target recognition based on deep learning," *J. Mar. Sci. Eng.*, vol. 10, no. 8, p. 1144, 2022.

[31] X. Chen *et al.*, "Application of EfficientNet-B0 and GRU-based deep learning on classifying the colposcopy diagnosis of precancerous cervical lesions," *Cancer Med.*, vol. 12, no. 7, pp. 8690–8699, 2023.

[32] C. Wang *et al.*, "Pulmonary image classification based on inception-v3 transfer learning model," *IEEE Access*, vol. 7, pp. 146533–146541, 2019.

[33] Y. Nan, J. Ju, Q. Hua, H. Zhang and B. Wang, "A-MobileNet: An approach of facial expression recognition," *Alexandria Eng. J.*, vol. 61, no. 6, pp. 4435–4444, 2022.

[34] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *arXiv Prepr. arXiv2106.10270*, 2021.

[35] "Resistor Dataset." [Online]. Available: https://www.kaggle.com/datasets/eralpozcan/resistor-dataset.

[36] "Electronic Components and devices." [Online]. Available: https://www.kaggle.com/datasets/aryaminus/electronic-components/code.

[37] "Transistor BC BD." [Online]. Available: https://www.kaggle.com/datasets/josevitormichelin/transistor-bc-bd/code.

[38] "Keras Applications." [Online]. Available: https://keras.io/api/applications.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[40] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019, pp. 6105–6114.

Conflict of Interest Notice

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

Ethical Approval and Informed Consent

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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 46-60
DOI: 10.35377/saucis...1402414
RESEARCH ARTICLE

An Analysis of Intelligent Turkish Text Classification Models for Routing Calls in Call Centers: A Case Study on the Republic of Turkiye Ministry of Trade Call Center

Muammer Özdemir [1], **Yasin Ortakcı** [2]

[1] Department of Computer Engineering, Karabuk University, Karabük, Türkiye
[2] Department of Computer Engineering, Karabuk University, Karabük, Türkiye

Corresponding author:
Yasin Ortakcı, Karabük Üniversitesi, Department of Computer Engineering
yasinortakci@karabuk.edu.tr

Article History:
Received: 09.12.2023
Accepted: 14.03.2024
Published Online: 15.03.2024

ABSTRACT
Call centers play a key role in the management of customer relationships in the modern business world. However, the growing demand for their services presents significant challenges, particularly in terms of staffing and handling increasing call volumes. This paper addresses these issues by presenting an AI-driven text classification framework tailored for the Republic of Turkiye Ministry of Trade Call Centre (MTCC), with the aim of automatically routing calls to relevant departments. Using a specific dataset of 20,000 phone call texts collected from the MTCC, the study employs TF-IDF, Word2Vec, and GloVe text vectorization techniques and applies various machine learning algorithms such as K-Nearest Neighbours, Naive Bayes, Support Vector Machines, Adaptive Boosting, Decision Tree and Random Forest for text classification. Through a comprehensive analysis, the study answers key research questions regarding optimal classifiers and vectorization methods. The proposed solution not only improves the efficiency of MTCC's call routing but also provides researchers with practical insights regarding Turkish text classification. The results indicate that a combination of the Random Forest classifier and Word2Vec text vectorization method is the optimal model that can manage to route calls in real-time.

Keywords: Text classification, Word2Vec, GloVe, TF-IDF, Call center

1. Introduction

Call centers play a crucial role in modern business in order to enhance customer relation management. Their goal is to ensure customer satisfaction by providing them with accurate information, responding quickly to inquiries, and dealing effectively with service requests. In recent years, there has been a surge in demand for call center services as a result of technological advances and increases in trading volumes. However, many companies and institutions face challenges in customer relations management due to a lack of qualified staff to handle initial inquiries. This situation leads to delayed and inaccurate problem resolution, resulting in longer wait times and customers having to make repeated calls, even for simple issues. The result is an increase in customer dissatisfaction and a rise in complaints. Furthermore, call center representatives, who are frequently overburdened, are exposed to excessive stress and have to deal with angry customers. Therefore, there is an urgent need for intelligent assistance systems to improve call center representatives' performance and motivation. These factors underline the significance of integrating automation and intelligent support tools into call centers.

In this paper, we develop an AI-driven text classification framework to address the challenges faced in call centers. As a case study, we have focused on the routing of incoming calls to the Republic of Turkiye's Ministry of Trade Call Centre (MTCC) to the relevant departments, taking into account the call text. MTCC receives an average of 10,000 daily calls, which are first answered by call center representatives. These representatives decide whether to handle the call themselves or forward it to the relevant department. However, due to the diverse call content and varying call center representative knowledge and experience levels, they sometimes provide customers with the wrong solutions or direct them to the wrong departments. These errors result in unresolved problems or delayed solutions. These delays not only disrupt the operation of the Ministry of Trade but also lead to economic losses for the country.

To tackle these challenges, this study introduces an innovative solution that utilizes various machine learning techniques to route incoming calls to the relevant departments automatically in the MTCC. The objective is to enhance customer satisfaction by delivering quicker and more precise solutions. To achieve this, we used a dataset of 20,000 sample call texts

received by the MTCC and their correctly routed departments. These texts were pre-processed and transformed into numerical representations using TF-IDF (Term Frequency- Inverse Document Frequency), Word2Vec, and GloVe text vectorization techniques. We then applied a variety of machine learning algorithms, including K-Nearest Neighbors (K-NN), Naive Bayes (NB), Support Vector Machines (SVM), Adaptive Boosting (AdaBoost), Decision Tree (DT), and Random Forest (RF) to classify these digitized call transcripts and route the calls to the appropriate departments.

Throughout our experiments, we conducted extensive analysis of text classification algorithms and text vectorization methods. Our primary objective was to find the most effective combination of classifiers and text vectorization methods, which demonstrate superior performance in classifying MTCC call text. Additionally, we investigated the responses to the following research questions (RQ): (RQ1) What is the most appropriate classifier to use in text classification? (RQ2) Which text vectorization technique yields better performance across the classifiers? (RQ3) Which classifier and text vectorization method combination produces the highest text classification results? (RQ4) Which combination of classifiers and text vectorization methods yields optimal results in terms of both runtime efficiency and accuracy? Our observations and findings regarding these matters are elaborated in section 4.4 in detail. Consequently, the key contributions of this study can be summarized as follows:

- We propose an automated system designed to streamline the routing of phone calls within the MTCC, thereby improving the efficiency of call center operations.
- This study presents a comprehensive comparative analysis of text classification performance on Turkish texts using classifiers such as K-NN, NB, SVM, AdaBoost, DT, and RF together with TF-IDF, Word2Vec, and GloVe text vectorization methods.
- Some practical information is provided for customer service representatives to enhance the performance of the proposed intelligent system.

The rest of the paper is organized as follows. Section 2 provides the current literature related to this study. Section 3 presents the methodology of this study, describing the data pre-processing, text vectorization, and classification steps. Section 4 provides comprehensive experiments on the MTCC dataset with its implications. Finally, Section 5 concludes our study and provides the scope for future work.

2. Literature Review

Customer satisfaction (CS) is a crucial objective of marketing research, which focuses on evaluating the satisfaction levels of customers with products and services through their experiences with companies [1-2]. Recent developments in information technology have yielded significant advances in assessing CS. Namely, to ensure CS, much research has focused on enhancing the performance of call center departments through various tools. For instance, Park and Gates conducted research demonstrating the ability to automatically measure CS by analyzing call transcripts. This method allows companies to evaluate satisfaction for each call in almost real-time [3]. Similarly, Chowdhury et al. investigated the predictive capacity of turn-taking as a key factor influencing user satisfaction in verbal conversations [4]. Luque et al. carried out a study aimed at predicting CS through the analysis of various acoustic features extracted from customer service conversations. The study utilized convolutional neural networks to measure satisfaction, with a focus on accentuation-related data in the dialogues. Their proposed method demonstrated superior performance in comparison to traditional systems, in terms of AUC and F-score criteria [5]. Chatterjee et al. categorized problematic and non-problematic phone calls by employing an SVM classifier based on audio features such as meal-frequency cepstral coefficients, energy, voice, and zero-crossing rate. Their system identified the problematic calls with an 87,5% accuracy rate [6].

Customer relation management studies on call centers have mainly concentrated on utilizing machine learning techniques for text classification. In this context, Meinzer et al. used four different machine learning classifiers, namely AdaBoost, K-NN, SVM, and RF, to quantify levels of customer dissatisfaction in the automotive sector. The study found that the SVM method, which utilized the radial basis function as its kernel, demonstrated the greatest accuracy among all classifiers in detecting dissatisfaction, with an accuracy of 88.8% [7]. Liu et al. introduced a model that integrates key utterance analysis and logistic regression algorithms for the classification of service conversations in call centers. The effectiveness of this model was demonstrated in their experiments, particularly with a limited training dataset [8]. Busemann et al. introduced a systematic method to categorize emails in call centers according to their content by leveraging shallow text processing and various machine learning techniques such as lazy learners, SVM, and symbolic eager learners. Their model was seamlessly integrated into an assistance system for call center representatives, improving the overall quality of responses [9]. Galanis et al. developed a technique to extract emotional segments from a call center speech database by incorporating SVM [10]. Emmanuela et al. measured customer satisfaction by applying text classification techniques to user surveys of 549 customers organized in different marketplaces. The study employed K-NN, DT, and NB machine learning methods and found that the DT algorithm had the highest accuracy [11].

The first step in text classification studies is transforming texts into numerical representations [12]. Salminen et al. presented

a comprehensive classification analysis for online hate detection using comments from various platforms such as YouTube, Reddit, Wikipedia, and Twitter. They employed a combination of classification algorithms, including LR, NB, SVM, XGBoost, and Artificial Neural Network (ANN), along with diverse feature extraction methods like Bag of Words (BoW), TF-IDF, Word2Vec, and BERT. Their findings revealed that XGBoost emerged as the top-performing model, achieving an F1 Score of 0.92 [13]. Alaoui and Nfaoui developed four different deep learning-based text classification models using CNN and LSTM networks with Word2Vec to detect malicious HTTP web requests. They concluded that LSTM has better performance than the others in terms of classification metrics and training time [14]. In another study, Cahyani and Patasik compared the efficacy of TF-IDF and Word2Vec methods for sentiment analysis of commuter line tweets using SVM and Multinomial NB methods. Their two-step approach first categorized tweets as having or not having emotion, followed by classification into five emotion types: happy, angry, sad, scared, and surprised. The results demonstrated that TF-IDF outperformed Word2Vec across various metrics, enhancing classification performance compared to previous studies [15]. Akuma et al. compared the performance of BoW and TF-IDF feature extraction methods for hate speech detection from live tweets. They combined machine learning methods such as NB, DT, LR, and K-NN with TF-IDF and BoW, evaluating them on the Kaggle Hate Speech and Offensive Language dataset. When the results were evaluated according to precision, recall, f1-score, and accuracy metrics, DT was the most successful machine learning, while TF-IDF outperformed BoW as a feature extraction technique [16].

For example, Öğe and Kayaalp performed sentiment analysis on IMDB comments using BoW, TF-IDF, FastText, and Word2Vec text representation methods. LR and SVM, which produce similar results, achieved 86%, 87%, 87%, and 83% accuracy for BoW, TF-IDF, Word2Vec, and FastText, respectively [17]. Ekici and Takcı integrated Word2Vec and TF-IDF methods with the Gradient Boosting algorithm and compared their performance on a Turkish spam dataset. According to the results of the study, the TF-IDF and Gradient Boosting pair were more successful than the Word2Vec&Gradient Boosting pair and CNN model [18]. Koruyan and Ekeryilmaz employed the TF-IDF approach, alongside LR, SVM, and Stochastic Gradient Descent classifiers, in their study, which categorized the complaints from three prominent consumer electronics retailers in Turkey that were received via the Sikayetvar.com website. The study showed that LR achieved the highest accuracy rate of 80% [19]. Çelik and Koc carried out a study that categorized Turkish news collected from various sources into six distinct groups. The study employed several classifiers such as SVM, NB, LR, RF, and ANN and used TF-IDF, Word2Vec, and FastText word representation methods. The FastText and SVM combination presented the highest accuracy rate of 95.75%, outperforming other methods [20].

Our study distinguishes itself from prior studies through a comprehensive exploration of various text classification techniques, particularly on a Turkish corpus. Additionally, our contribution to the literature includes an in-depth examination of the synergy between these classifiers and diverse text vectorization methods applied to Turkish texts. Furthermore, we introduce an intelligent assistant application within the developed MTCC framework, capable of autonomously directing calls to the pertinent units.

3. Methodology

In this section, we describe our framework, presenting a systematic approach for vectorization and classification of text within call conversation. Our methodology starts with the pre-processing of incoming call text at the MTCC. We applied three different techniques for text digitization: TF-IDF, Word2Vec, and GloVe. After generating text representation, we proceeded with the classification of call text using six distinct machine-learning algorithms: K-NN, NB, SVM, AdaBoost, DT, and RF. We then evaluated the performance of each classifier with respect to the text vectorization methods to identify the most compatible classifier and text vectorization method. Figure 1 depicts the general overview of our framework.

3.1 Data Pre-processing

Our dataset consists of transcripts of telephone conversations between customers and call center representatives at the MTCC. The data pre-processing step of this study encompasses the tasks of data cleaning, tokenization, and lemmatization.

The data cleaning process includes a series of steps. Firstly, it corrects spelling errors and removes empty text, duplicates, and punctuation from the dataset. Moreover, it filters out user-specific data such as identity numbers, phone numbers, and declaration numbers. Additionally, we eliminate stop words, which are typically short, high-frequency words that do not have significant meaning on their own and are often used for sentence structure and cohesion [21-22]. Stop words are generally excluded from Natural Language Processing (NLP) analysis to reduce dimensionality and speed up processing since they have limited semantic contribution to the text [23]. As the MTCC dataset includes Turkish text records, we used the 'Turkish' package of the Natural Language Toolkit (NLTK) library to remove the stop words, such as "ama", "gibi", "fakat",

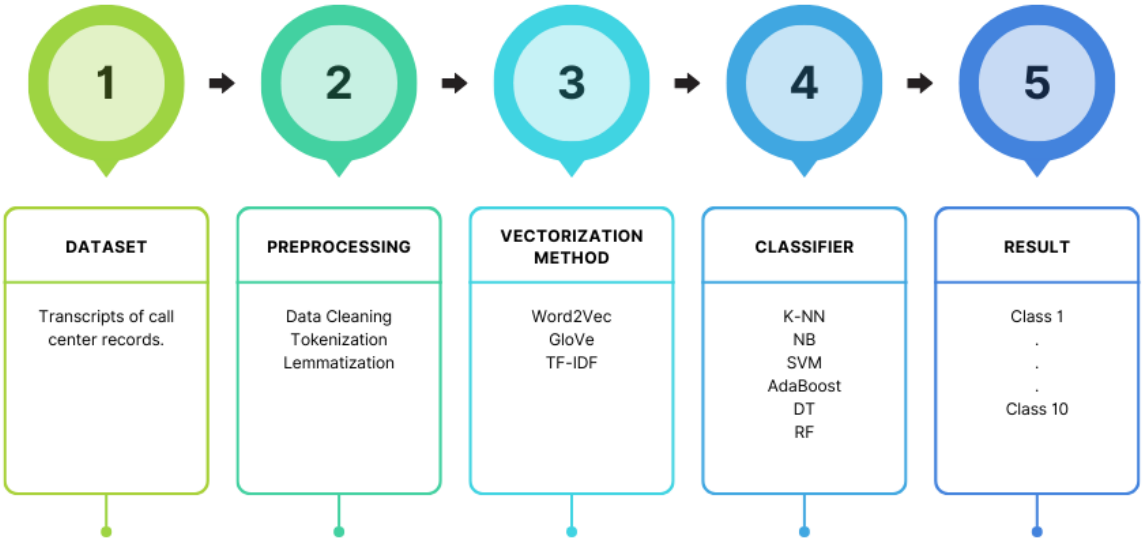


Figure 1 The Implementation Stages of The Proposed Framework for Call Text Classification

"yani", "ya", "mı", "mi", "ne", "ve" and so on. NLTK is an open-source Python library that offers a comprehensive set of NLP functions. Data cleaning ends with the standardization of all text to lowercase.

In the second step of pre-processing, we tokenized the text by segmenting it into words based on spaces. Tokenization is an essential pre-processing technique that breaks text into meaningful units referred to as tokens, such as words, phrases, or symbols [24]. For tokenization, we also utilized the NLTK library.

Finally, we implemented lemmatization to extract the root of the words. Lemmatization is a fundamental NLP technique that modifies or eliminates suffixes on a word to obtain its basic form, known as the 'lemma' [25]. In this study, we used the 'tr' class from Simplemma, an open-source Python library designed to identify the root or basic form of words [26].

3.2 Text Vectorization

After completing the data pre-processing, we applied three distinct methods for converting text to vector representations: TF-IDF, Word2Vec, and GloVe. These methods transform the textual data into numerical representations, allowing us to address NLP problems through mathematical approaches.

3.2.1 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is one of the most widely used text representation methods, providing a statistical metric to quantify the mathematical significance of words within a given corpus. TF-IDF considers both the frequency of a word within a specific document and how rarely it appears across all documents. The TF-IDF score for a word in a document is calculated by multiplying its Term Frequency (TF) and the Inverse Document Frequency (IDF) scores, resulting in a score that ranges from 0 to 1.

To illustrate the computation of a basic TF-IDF value in a given corpus, let's consider an example. TF refers to the division of the frequency of a given word (t) in a particular document (*d*), divided by the total number of words in that document (*Z*), as shown in Eq. (1). On the other hand, the IDF involves the calculation of the logarithm of the total number of documents (*N*) in the corpus (*C*), divided by the number of documents in which the word (*t*) appears, as shown in Eq. (2). The TF-IDF value is then obtained by multiplying these two values, as in Eq. (3). Consequently, words that frequently appear within a specific document but rarely across all documents receive a high score, indicating their significance in the text representation. Conversely, words that frequently appear across all documents receive a low score, showing their limited influence [27].

$$TF(t,d) = count(t,d) \,/\, Z \tag{1}$$

$$IDF\ (t,D) = log\left(\frac{N}{count\ (\ d\ \in\ D:\ t\ \in\ d\ )}\right) \tag{2}$$

$$TF\ -IDF\ (t,d,D) = TF(t,d) * IDF(t,D) \tag{3}$$

where:

$count(t, d)$: *Number of a given word (t) in a specific document.*

Z: Total number of words in a particular document

D: The corpus including all documents

N: Total number of documents in the corpus

$count(d \epsilon D: t \epsilon d)$*: Number of documents where the t term appears.*

For instance, if the term "kapikule" appears four times in a document that is composed of a total of 50 words, then the TF value would be 0.08. Additionally, if "kapikule" is found in 10 out of 500 documents in the corpus, then the IDF value would be calculated as 1.69. By multiplying both values, the resulting TF-IDF value equals 0.135.

3.2.2 Word2Vec

Word2Vec, developed by Mikolov et al. in 2013, is a text representation technique designed to improve the effectiveness of the Google search engine [28]. Its primary objective is to generate word representations using ANNs in a prediction-based manner. Unlike TF-IDF, which encodes text as high-dimensional sparse matrices, Word2Vec produces dense vector word embeddings for each word in the text. It also places similar words close together in the vector space. Word embeddings offer two key advantages: a reduction in computational cost due to lower dimensionality and improved performance in NLP tasks by effectively grouping words that are semantically similar [29-30]. Word2Vec generates word embeddings by capturing the semantic meaning of words based on their context within the text. To achieve this, it employs two different neural network models: Continuous Bag-of-Words (CBOW) and Skip-gram [31].

The CBOW model aims to forecast the central word by considering the surrounding contextual words, which usually include a few words preceding and following the targeted word in the text. CBOW employs a defined window size that specifies the number of neighboring words taken into account. The words within this window collectively contribute to the prediction of the corresponding word. Figure 2 visually represents the CBOW architecture, with w(t) denoting the current word and terms ranging from w(t-2) to w(t+2) representing the surrounding words within the window size of the corresponding word, w(t). In this neural network, the hidden layer is a typical, fully connected, dense layer, whereas the output layer calculates the probabilities for the target word in the vocabulary.

The skip-gram model takes a word in a text as the center of the window size, then predicts the neighboring words in this window as outputs. The central word in the window is denoted as w(t), while the surrounding words are represented with a range from w(t-2) to w(t+2) as shown in Figure 3. The skip-gram model predicts the neighboring words around a single word, considering the context. This characteristic makes the skip-gram model advantageous in representing less common or

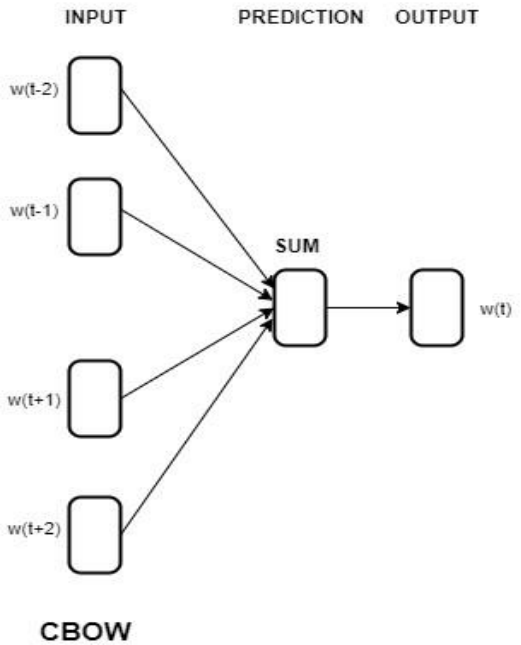


Figure 2 The General Architecture of CBOW

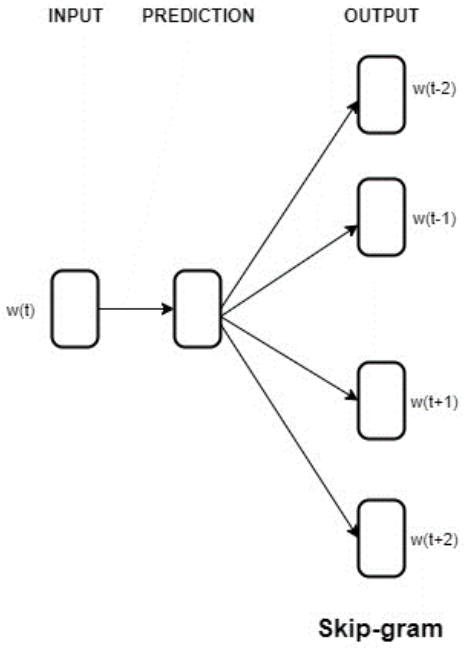


Figure 3 The General Architecture of Skip-Gram

terms, enhancing its efficiency compared to CBOW. Skip-gram is also more effective than CBOW in handling large datasets. These reasons and the fact that Skip-Gram produces better results in a shorter time than CBOW in our preliminary test led us to use Skip-Gram in this study. In our preliminary test, the runtimes for Skip-gram were 4.64, 0.65, 1.6, 4.61, 5.29, and 2.54 seconds respectively for K-NN, SVM, NB, AdaBoost, DT, and RF, whereas for CBOW, the corresponding times were 4.86, 0.66, 2.1, 4.70, 13.03, and 6.66 seconds. These results clearly show that Skip-gram exhibits shorter execution times across all classifiers compared to CBOW.

3.2.3 GloVe

GloVe, an acronym for "Global Vectors," refers to a word embedding model that was introduced by Pennington et al. [32]. In GloVe, the co-occurrence matrix (X) is created from a corpus containing v words and presented in a $v \times v$ matrix format. Each element in this matrix (X_{ij}) denotes the frequency of co-occurrence between $word\ i$ and $word\ j$. For instance, the co-occurrence matrix for the given sentence "The apple is on the table" with a window size of one is given in Figure 4.

This model distinguishes itself from Word2Vec by presenting a novel unbiased objective function that employs probability statistics for word prediction. GloVe generates precise word vectors by minimizing the prediction errors through an error function, which combines both local statistics derived from the corresponding sentence and global statistics obtained from the entire corpus.

Words	*The*	*apple*	*is*	*on*	*table*
The	0	**1**	0	**1**	**1**
apple	**1**	0	**1**	0	0
is	0	**1**	0	**1**	0
on	**1**	0	**1**	0	0
table	**1**	0	0	0	0

Figure 4 Glove Co-Occurrence Matrix of Given Sample Sentence

3.3 Text Classification

This stage focuses on routing the call center records, the numerical representations of which have been generated in the preceding phase, to the respective departments. Within this framework, six different machine learning algorithms, K-NN, NB, SVM, AdaBoost, DT, and RF, are trained using the MTCC dataset to establish text classification models. This study presents a comprehensive analysis of the effectiveness of each classification model, taking into account its harmony with TF-IDF, Word2Vec, and GloVe methods.

3.3.1 K-Nearest Neighbors (K-NN)

The k-NN algorithm, first formulated by Evelyn Fix and Joseph Hodges in 1951 and later extended by Thomas Cover, is a non-parametric supervised learning approach [33]. This algorithm uses a sample dataset with known classes to determine the class membership of a new sample. The calculation is performed by considering the distances between the new data point and the existing dataset, ultimately determining the final class of the point through the application of the k nearest neighbor majority approach. Specifically, the K-NN classifier uses the Euclidean distance metric to quantify the proximity between data points.

3.3.2 Naïve Bayes (NB)

NB, named after the mathematician Thomas Bayes, classifies data using the principles of probability [34]. NB calculates the probability of each class separately for a given sample and assigns the sample to the class with the highest probability. This technique is often used in various scenarios based on text classification, such as spam filtering and sentiment analysis. In general, versions of Multinomial NB, Bernoulli NB, and Gaussian NB are commonly used NB types.

3.3.3 Support Vector Machines (SVM)

SVM, proposed by Cortes and Vapnik (1995), has been extensively employed by researchers to address classification problems. SVM is a supervised machine learning algorithm that is used for classification and regression tasks [35]. This algorithm projects the data onto a high-dimensional feature space using kernels and identifies hyperplanes that divide the data into separate categories [36]. When identifying these hyperplanes, the aim is to maximize the distance between the closest points within different categories. This approach provides a robust classification model that is more resistant to noise and outliers.

3.3.4 Decision Tree (DT)

DT is a hierarchical tree-like structure based on the features of the input data. Each node in the tree represents a particular feature, while the edges leading from the node represent grouping components [37]. DTs can capture non-linear relationships between the input features and the target class. However, they can suffer from overfitting in cases where the tree is too complex or the training data is too noisy, leading to poor classification performance. To overcome these problems, pruning or ensembling methods can be applied.

3.3.5 Adaptive Boosting (AdaBoost)

AdaBoost is a popular ensemble machine-learning algorithm used for classification problems. It creates a robust classification model by combining multiple weak classifiers. AdaBoost is known for its ability to solve complex classification problems and handle noisy and imbalanced data. It has been successfully applied in various fields, such as object detection, face recognition, and medical diagnosis [38].

3.3.6 Random Forest (RF)

RF is an ensemble learning algorithm consisting of multiple DTs, each trained on a different randomized subset of the dataset [39]. RF runs the standard DT algorithm on each subset and makes its final prediction by combining the results of the DT sets. In this combining process, RF applies the majority voting or the weighted voting methods to the class labels obtained from all trees. RF is a popular and powerful algorithm and has been successfully applied to many classification problems, such as image and speech recognition.

4. Experimental Results

This section presents a set of experiments and their results to evaluate the effectiveness of the proposed text classification framework. To ensure the reliability of the results, each classifier, in combination with three text vectorization methods, was

run 10 times, and the averages were calculated along with their respective standard deviations.

To increase the robustness of the classifier models, we also incorporated k-fold cross-validation, a statistical technique essential for impartially evaluating the performance of machine learning algorithms. Rather than assessing a model solely on a single training/test subset, cross-validation employs k-fold validation, dividing the dataset into k different training/test subsets. This approach effectively mitigates the overfitting issues, a common concern in machine learning, and produces more reliable results. By training the model on varying training/test partitions of the same dataset, cross-validation ensures a more uniform and robust training process. In this study, we applied 5-fold cross-validation, meaning that the model creation was repeated five times. Within each fold, the dataset was divided into training and test subsets, with a split of 80% to 20%.

4.1 Setup

The experiments were carried out on a PC with the following specifications: an Intel (R) Core (TM) i7-10700 CPU @ 2.90GHz, with 8 cores and 16 logical processors, supported by 32 GB of RAM and a 500 GB SSD. The proposed framework was developed using the Python programming language within the Spyder environment (Anaconda3 version 4.10.3). We leveraged the sci-kit-learn library version 0.24.2[1] to implement the classification algorithms and cross-validation technique and to calculate evaluation metrics. The optimal parameters of the classifiers used in the experiments are listed in Table 1.

Table 1 Optimal Parameter Values of Each Classifier Used in The Experiments

Method	Parameter	Value
K-NN	n-neighbors	5
NB	alpha	1.0
SVM	kernel	poly
	degree	3
	c	1.75
	gamma	scale
	coef0	0.0
AdaBoost	base_estimator	ExtraTreeClassifier
	n_estimators	50
	learning_rate	1.0
DT	min_samples_split	default=2
	min_samples_leaf	default=1
RF	min_samples_split	default=2
	min_samples_leaf	default=1
	n_estimators	15

4.2 Dataset

Our dataset consists of 20,000 records, including transcripts of phone calls to the MTCC. These records are obtained from actual calls made to the call center and are converted into text and forwarded to the relevant department by customer representatives. The department to which the call is directed becomes the label of the records in the dataset. The records and labels used in this study were validated by experts from the Ministry of Commerce to ensure their accuracy and suitability as a dataset. However, these records are in their raw and unedited form, with no imposed character limit. This comprehensive collection is sourced equally from 10 different departments within the Ministry of Trade. The distribution of these records across the departments and the average number of words and characters in the records of each department are shown in Table 2. Our dataset consists of 20,000 records, including transcripts of phone calls to the MTCC.

Table 2 The Distribution of Call Records Across the Department

Department Label	Department Name	Number of Calls	Average Word Count	Average Character Count
0	Center of Ministry	2000	31	231
1	Retail Trade	2000	29	226
2	MERSİS	2000	37	281
3	Subscription Agreements	2000	46	329
4	Unfair Commercial Practices	2000	31	239
5	After Sales Services	2000	44	320
6	Department Of Exemptions	2000	34	272
7	Consumer Arbitration Committee App Processes	2000	43	318
8	Distance Sales	2000	44	325
9	Defective Goods and Services	2000	33	248

[1] https://scikit-learn.org/stable/user_guide.html

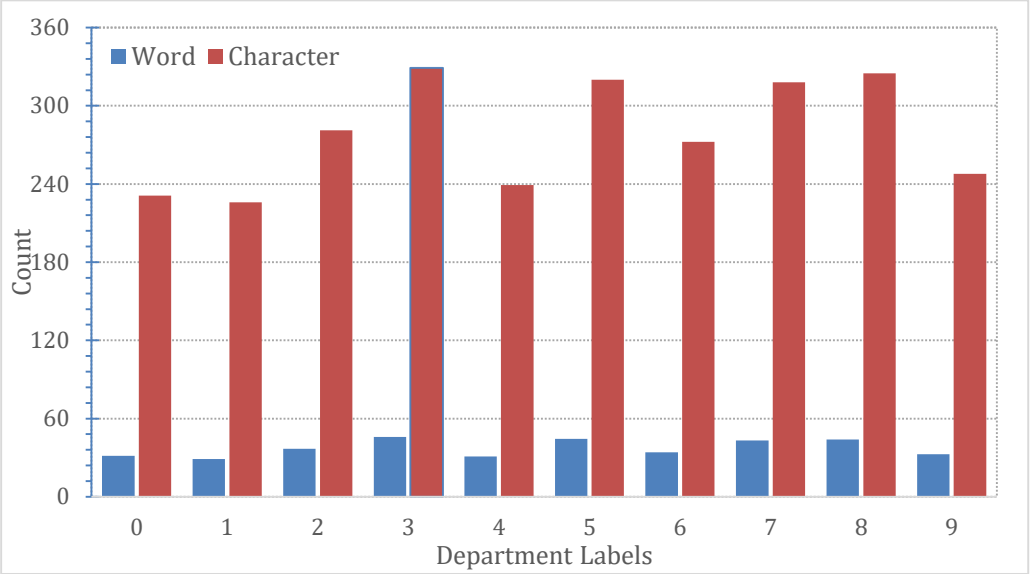


Figure 5 Distribution of the Average Count of Words and Characters in the Call Text across the 10 Departments

Figure 5 illustrates the average number of words and characters present in the texts of phone calls across the 10 departments outlined in Table 2. When we examine the average word and character counts across all departments, it is evident that there is no significant difference that might negatively affect the classification of call texts. This reasonable distribution within our newly created dataset contributes to improving the prediction accuracy of text classification and building a robust model.

4.3 Performance Metrics

To evaluate the effectiveness of text classification algorithms in our experiments, we employed accuracy, precision, recall, and f1-score metrics, which are derived from the confusion matrix. The confusion matrix is a versatile tool, applicable to both binary and multi-class classification problems, that provides a quantitative representation of predicted values compared to their actual values [40]. The confusion matrix enumerates True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) instances, encapsulating the sample counts associated with the results of a classification process. TP represents the number of correctly classified positive examples, while TN denotes the number of correctly classified negative examples. FP signifies the number of examples that are classified as positive but are negative, and FN indicates the number of examples that are classified as negative but are positive.

The accuracy metric, as outlined by Eq. (4), determines the proportion of accurate predictions made by the model out of the total number of predictions. Precision signifies the positive predictive value, which measures the ratio of true positive samples to the total number of positives predicted by the model as in Eq. (5). Recall represents the degree to which positive examples are accurately predicted (Eq. 6). The f1-score is the harmonic mean of recall and precision (Eq. 7).

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{4}$$

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

$$F1-Score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{7}$$

4.4 Results and Discussion

In the experiments, we ran a total of 18 text classification models, each a combination of a classifier algorithm and a text vectorization method, 10 times each on the MTCC call text. Figure 6 displays the average accuracy values of all models with their standard deviations (SDs) on the bar chart. The small size of the SD bars for all models underlines the consistent results of these models. In addition, we conducted a two-way ANOVA statistical test of 6 (classifiers: AdaBoost, DT, KNN, NB, RF, and SVM) x 3 (vectorizers: Glove, TF-IDF, and Word2Vec) to analyze the variance between the models. We only considered the accuracy as other performance metrics have parallel results with accuracy. The results of the variance analysis were as follows:

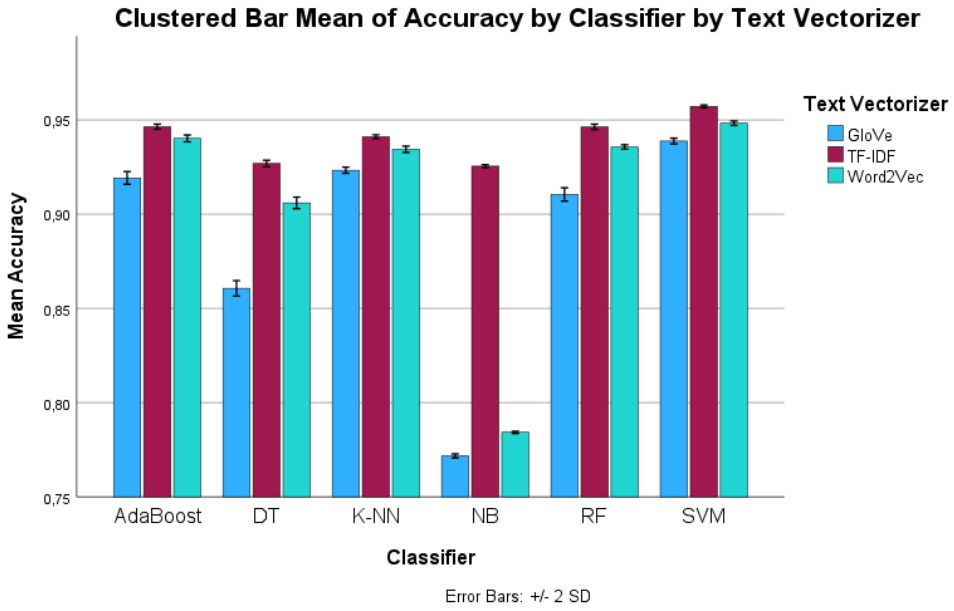


Figure 6 Mean Accuracy Values with Standart Deviation Bars for Each Model

- The results for the classifier showed a significant main effect with **$F(5,162)=58209.99$, $p<0.001$, partial $\eta^2=0.999$**, indicating that the accuracy varies between different classifiers.
- The main effect of the vectorizer method was also significant with **$F(2,162)=41684.37$, $p<0.001$, partial $\eta^2=0.998$**. Namely, each vectorizer has significantly different accuracy results from each other with **$Mean_{GloVe}=0.887$, $SD_{GloVe}=0.057$; $Mean_{TF\text{-}IDF}=0.940$, $SD_{TF\text{-}IDF}=0.011$; and $Mean_{Word2Vec}=0.908$, $SD_{TF\text{-}IDF}=0.057$** values.
- The interaction effect for classifier × vectorizer was also significant with **$F(10,162)=9342.70$, $p<0.001$, partial $\eta^2=0.998$**, meaning that the effect of a classifier on accuracy depended on the text vectorizer method.

After confirming that there were significant differences between the models, we performed comparative analyses of these models. When comparing the accuracy values of the classifiers for TF-IDF, the models perform in the following order: SVM, AdaBoost, RF, K-NN, DT, and NB. This ranking remains the same for Word2Vec. In the case of GloVe, the order changes to SVM, K-NN, AdaBoost, RF, DT, and NB. In addressing RQ1, it can be concluded that SVM consistently achieved the highest accuracy, while NB consistently yielded the lowest accuracy regardless of the text vectorization method. Furthermore, it is evident in Figure 6 that SVM, K-NN, AdaBoost, and RF exhibit comparable classification performance, with SVM consistently demonstrating superior performance.

In addition to the accuracy, Table 3 elaborates on the classification results with the metrics precision, recall, f1-score, as well as the runtime of each model obtained when testing a sample text set. The values presented in Table 3 are the averages derived from 10 distinct experimental runs. Regarding the efficiency of the text vectorization methods, TF-IDF consistently produced the highest metrics across all classifiers, followed by Word2Vec and GloVe. The performance order of the text vectorization methods in all classifiers was consistently TF-IDF, Word2Vec, and GloVe as shown in Figure 7. This finding provides a reliable insight into the selection of text vectorization methods as well as answers to our RQ2.

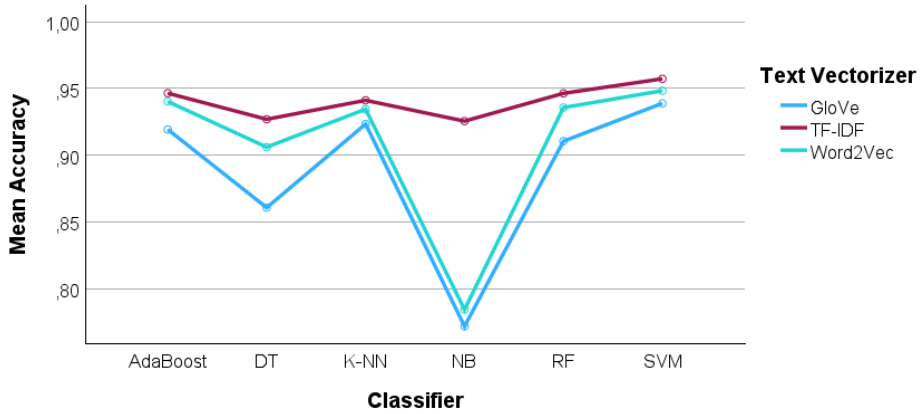


Figure 7 Mean Accuracy Value of Classifiers across the Text Vectorizers

To find the answer to RQ3, we compared the experimental results of 18 different classifier and text vectorizer combinations. As a result, the SVM&TF-IDF combination showed superior performance, achieving 95.7% accuracy, 95.8% precision, 95.6% recall, and 95.7% f1-score in accurately classifying the call text into the corresponding departments. Conversely, the NB classifier showed the lowest performance of all three text vectorization methods for all classification metrics. In particular, the combination of NB with the Word2Vec and GloVe methods fell significantly below the average performance observed with the other methods. On the other hand, the classification performance of the models within our framework closely aligns, except for the combinations of NB&Word2Vec, NB&GloVe, and DT&GloVe.

When comparing runtimes, it is evident that the Word2Vec method has the shortest runtime for all classifiers except K-NN. The DT&Word2Vec combination has the shortest runtime (0.095 sec) among the models, but its accuracy (0.906) is approximately 5% lower than the maximum accuracy observed in the experiments. On the other hand, K-NN and AdaBoost were the least efficient classifiers in terms of runtime. The combination of SVM&TF-IDF achieved the highest accuracy rate of 0.957 with a reasonable runtime averaging 0.937 seconds. The SVM&Word2Vec pair reached an accuracy rate of only about 1% less than SVM&TF-IDF, with an average accuracy of 0.948. Notably, its runtime averages 0.373 seconds, nearly three times less than SVM &TF-IDF. Therefore, SVM&Word2Vec also emerges as one of the most optimal models in terms of both runtime efficiency and accuracy. Regarding RQ4, the experimental results indicate that combinations such as SVM&Word2Vec or SVM&TF-IDF demonstrate the highest levels of accuracy and the most efficient runtimes across all models.

Table 3 Mean Performance Metric Results for 10 Different Runs of Each Model and the Mean Runtimes of Each Model at the Test on a Sample Text Set.

Classifier	Vectorizer	Accuracy ±SD	Precision	Recall	F1-Score	Runtime (sec)
AdaBoost	TF-IDF	0,946 ±0,0007	0,950	0,947	0,948	5.792
	Word2Vec	0,940 ±0,0009	0,944	0,940	0,941	1.321
	GloVe	0,919 ±0,0017	0,927	0,917	0,920	1.201
DT	TF-IDF	0,927 ±0,0009	0,927	0,927	0,927	0.294
	Word2Vec	0,906 ±0,0015	0,906	0,906	0,906	0.095
	GloVe	0,861 ±0,0020	0,860	0,862	0,860	0.119
K-NN	TF-IDF	0,941 ±0,0005	0,940	0,941	0,940	1.863
	Word2Vec	0,934 ±0,0008	0,935	0,934	0,934	1.884
	GloVe	0,923 ±0,0008	0,923	0,923	0,921	1.959
NB	TF-IDF	0,926 ±0,0004	0,925	0,926	0,925	0.128
	Word2Vec	0,784 ±0,0003	0,796	0,784	0,788	0.109
	GloVe	0,772 ±0,0005	0,786	0,771	0,775	0.131
RF	TF-IDF	0,946 ±0,0007	0,948	0,946	0,947	1.122
	Word2Vec	0,936 ±0,0006	0,938	0,936	0,937	0.706
	GloVe	0,910 ±0,0018	0,918	0,910	0,912	0.724
SVM	TF-IDF	**0,957** ±0,0004	**0,958**	**0,957**	**0,957**	0.937
	Word2Vec	0,948 ±0,0006	0,949	0,948	0,948	0.373
	GloVe	0,939 ±0,0008	0,939	0,939	0,939	0.443

Figures 8, 9, and 10 present the confusion matrices for each classifier, obtained by using the TF-IDF, Word2Vec, and GloVe methods, respectively. Within these matrices, numerical labels ranging from 0 to 9 are assigned to the departments outlined in Table 2. The rows indicate the actual department numbers, while the columns represent the predicted department numbers. These matrices provide a comprehensive overview of the classification models' outcomes, considering the correct and incorrect distribution of calls across the departments. For instance, in Figure 7. a, corresponding to the SVM&TF-IDF combination that achieved the highest accuracy in the experiments, 1793 out of 2000 calls for Department 0 (Centre of Ministry) are correctly classified. In contrast, the other 207 calls were misclassified. Specifically, 201 of these calls were assigned to Department 1 (Retail Trade), while 6 calls were assigned to Department 5 (After Sales Services), as incorrect predictions.

Furthermore, these confusion matrices provide some practical information for customer service representatives to improve the performance of the proposed intelligent system. Namely, upon examining all the confusion matrices in Figures 8, 9, and 10, a consistent pattern emerges; each model has the highest error rates in predicting Department 1 (Retail Trade). This finding suggests that call center representatives should use more precise terminology when referring to the Retail Trade Department. Thus, the models' error rates are likely to decrease, leading to more effective and accurate classification results.

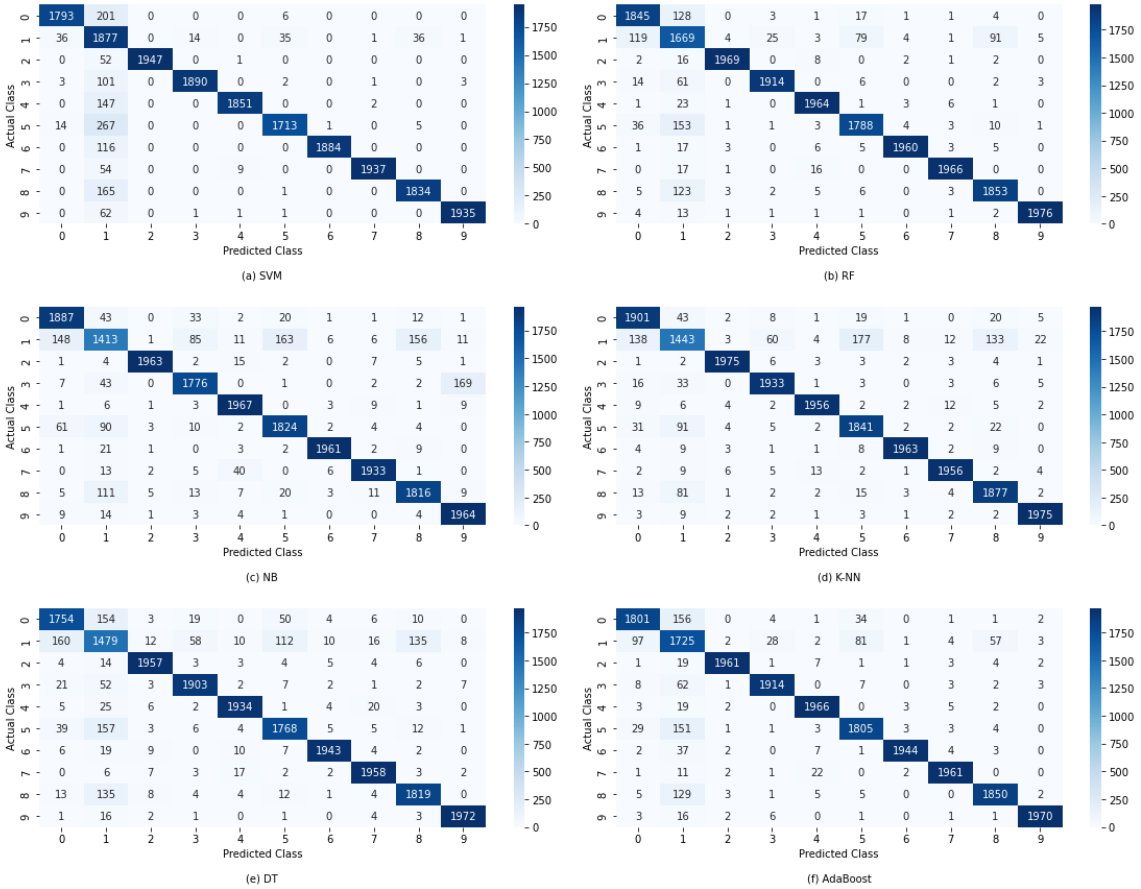


Figure 8 TF-IDF Confusion Matrices for All Classifiers

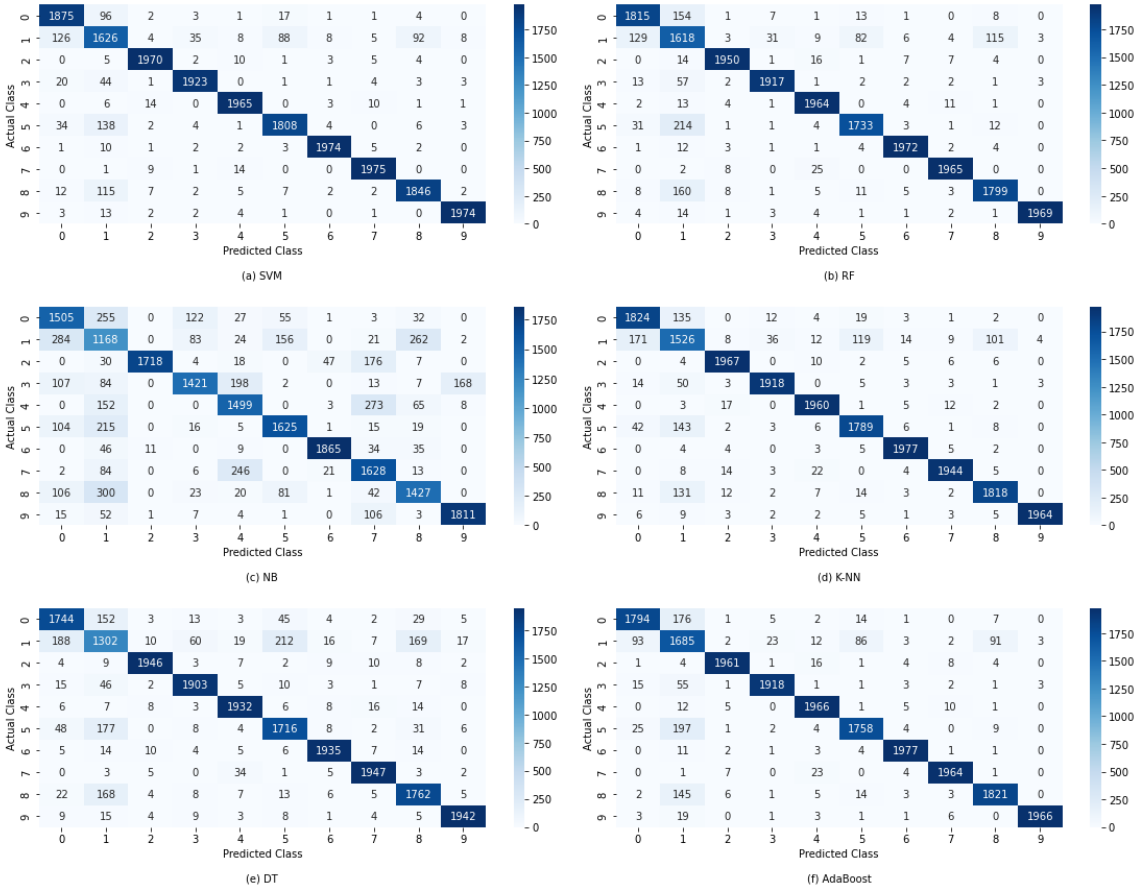


Figure 9 Word2Vec Confusion Matrices for All Classifiers

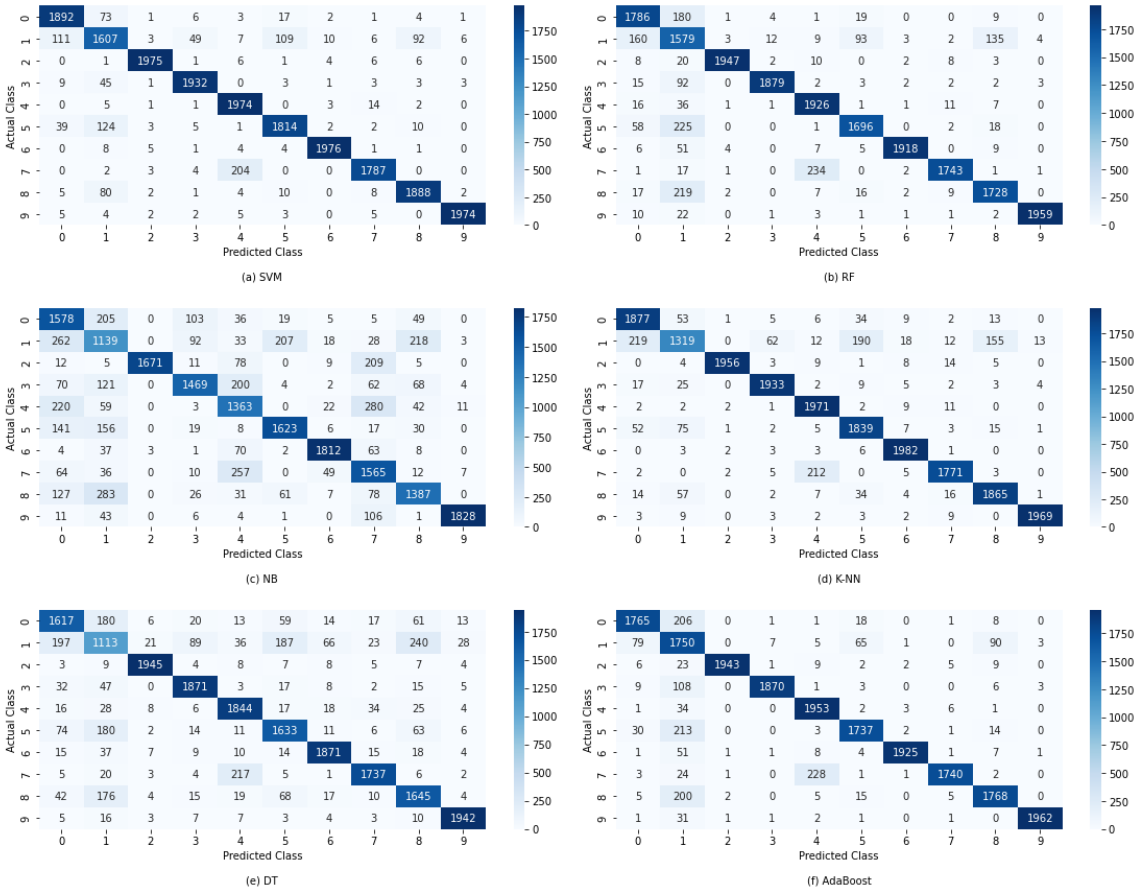


Figure 10 Glove Confusion Matrices for All Classifiers

4. Conclusion

The manual routing of incoming calls in call centers often results in errors, posing challenges to effective customer relationship management. To address these issues, this study introduces an intelligent call center framework. This framework takes the data of MTCC, which receives an average of 10,000 daily calls, as a case study and transforms the transcripts of the incoming calls into numerical representations with TF-IDF, Word2Vec, and GloVe text vectorization methods. Subsequently, six different machine learning algorithms, such as K-NN, NB, SVM, AdaBoost, DT, and RF, are used to classify the call text records to be directed to the relevant department.

In this study, we conducted a series of comprehensive experiments to identify the most successful and practical combination of classifier and text vectorization methods. The results provide valuable insights into classification algorithms and text vectorization techniques applicable to call centers. While SVM emerges as the most successful classifier, TF-IDF outperforms the other text vectorization methods. Considering both runtime and classification performance, the combination of SVM&TF-IDF and SVM&Word2Vec emerge as the most suitable models to serve as an intelligent assistant in MTCC. As a result, the proposed system was able to provide a real-time, automated, Turkish language-oriented solution for call centers.

The complexity of Turkish, in which suffixes can change the meanings of words, poses a significant constraint on word embedding techniques such as Word2Vec and GloVe. Further research is required to address this limitation in processing Turkish text. On the other hand, as a future plan, large language models, which have become very popular recently, can be used as text embedding methods to investigate their success in the classification of Turkish texts.

Acknowledgments

We would like to thank the General Directorate of Information Technologies of the Ministry of Trade in Turkiye for generously providing access to the call center data for the purposes of this study.

References

[1] P. G. Patterson, L. W. Johnson and R. A. Spreng, "Modeling the Determinants of Customer Satisfaction for Business-to-Business Professional Services," *J. Acad. Mark. Sci.*, vol. 25, no. 1, pp. 4–17, 1996, doi: 10.1177/0092070397251002.

[2] V. Pallotta, R. Delmonte, L. Vrieling and D. Walker, "Interaction Mining: The new frontier of Call Center Analytics," in Proc. CEUR Workshop in DART@ AI* IA., vol. 771, Sep. 2011, pp. 1-12.

[3] Y. Park and S. C. Gates, "Towards real-time measurement of customer satisfaction using automatically generated call transcripts," in *Proc. Int. Conf. Inf. Knowl. Manag.*, vol. 24754, 2009, pp. 1387–1396, doi: 10.1145/1645953.1646128.

[4] S. A. Chowdhury, E. A. Stepanov and G. Riccardi, "Predicting user satisfaction from turn-taking in spoken conversations," presented at the INTERSPEECH, San Francisco, USA, Sept. 8-12, 2016, pp. 2910–2914, doi: 10.21437/Interspeech.2016-859.

[5] J. Luque, C. Segura, A. Sanchez, M. Umbert and L. A. Galindo, "The role of linguistic and prosodic cues on the prediction of self-reported satisfaction in contact centre phone calls," presented at the INTERSPEECH, Stockholm, Sweden, Aug. 20-24, 2017, pp. 2346–2350, doi: 10.21437/Interspeech.2017-424.

[6] J. Chatterjee, A. Saxena and G. Vyas, "An automatic and robust system for identification of problematic call centre conversations," presented at the *Int. Conf. Micro-Electronics Telecommun. Eng. (ICMETE),* Ghaziabad, India, Sept. 22-23, 2016, pp. 325–330, doi: 10.1109/ICMETE.2016.48.

[7] S. Meinzer, U. Jensen, A. Thamm, J. Hornegger and B. M. Eskofier, "Can machine learning techniques predict customer dissatisfaction? A feasibility study for the automotive industry," *Artif. Intell. Res.*, vol. 6, no. 1, p. 80-90, Dec. 2016, doi: 10.5430/air.v6n1p80.

[8] Y. Liu, B. Cao, K. Ma, and J. Fan, "Improving the classification of call center service dialogue with key utterences," *Wirel. Networks*, vol. 27, no. 5, pp. 3395–3406, 2021, doi: 10.1007/s11276-021-02573-7.

[9] S. Busemann, S. Schmeier, and R. G. Arens, "Message classification in the call center", in *Proc. Sixth Applied Natural Language Processing Conference*, 2000, pp. 158–165, doi: 10.3115/974147.974169.

[10] D. Galanis, S. Karabetsos, M. Koutsombogera, H. Papageorgiou, A. Esposito and M. T. Riviello, "Classification of emotional speech units in call centre interactions," in *Proc. 4th IEEE Int. Conf. Cogn. Infocommunications (CogInfoCom)*. Proc., 2013, pp. 403–406, doi: 10.1109/CogInfoCom.2013.6719279.

[11] E. P. Emmanuela, F. K. Tjendra, S. Kezia and D. Suryani, "Classification of Customer Satisfaction in Marketplace," presented at the *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE), Jakarta,* ID, Feb. 16, 2023, doi: 10.1109/ICCoSITE57641.2023.10127788.

[12] A. Mousavi, M. Rezaee and R. Ayanzadeh, "A survey on compressive sensing: Classical results and recent advancements," *J. Math. Model.*, vol. 8, no. 3, pp. 309–344, 2020, doi: 10.22124/jmm.2020.16701.1450.

[13] J. Salminen, M. Hopf, S. A. Chowdhury, S. gyo Jung, H. Almerekhi and B. J. Jansen, "Developing an online hate classifier for multiple social media platforms," *Human-centric Comput. Inf. Sci.*, vol. 10, no. 1, pp. 1–34, Jan. 2020, doi: 10.1186/s13673-019-0205-6.

[14] R. L. Alaoui and E. H. Nfaoui, "Web attacks detection using stacked generalization ensemble for LSTMs and word embedding," in *Proc. Comput. Sci.*, vol. 215, 2022, pp. 687–696, doi: 10.1016/j.procs.2022.12.070.

[15] D. E. Cahyani and I. Patasik, "Performance comparison of tf-idf and word2vec models for emotion text classification," *Bull. Electr. Eng. Informatics*, vol. 10, no. 5, pp. 2780–2788, 2021, doi: 10.11591/eei.v10i5.3157.

[16] S. Akuma, T. Lubem and I. T. Adom, "Comparing Bag of Words and TF-IDF with different models for hate speech detection from live tweets," I*nt. J. Inf. Technol*., vol. 14, no. 7, pp. 3629–3635, 2022, doi: 10.1007/s41870-022-01096-4.

[17] B. C. Öğe and F. Kayaalp, "Farklı Sınıflandırma Algoritmaları ve Metin Temsil Yöntemlerinin Duygu Analizinde Performans Karşılaştırılması", *DUBİTED,* vol. 9, no. 6, pp. 406–416, 2021, doi: 10.29130/dubited.1015320.

[18] B. Ekici and H. Takcı, "Spam Tespitinde Word2Vec ve TF-IDF Yöntemlerinin Karşılaştırılması ve Başarı Oranının Artırılması Üzerine Bir Çalışma," *Bilecik Şeyh Edebali Üniversitesi Fen Bilim. Derg.*, vol. 8, no. 2, pp. 646–655, 2021, doi: 10.35193/bseufbd.935247.

[19] K. Koruyan and A. Ekeryılmaz, "Makine Öğrenmesi ile Müşteri Şikayetlerinin Sınıflandırılması," *AJIT-e Acad. J. Inf. Technol.*, vol. 13, no. 50, pp. 168–183, 2022, doi: 10.5824/ajite.2022.03.004.x.

[20] Ö. Çelik and B. C. Koç, "TF-IDF, Word2vec ve Fasttext Vektör Model Yöntemleri ile Türkçe Haber Metinlerinin Sınıflandırılması", DEUFMD, vol. 23, no. 67, pp. 121–127, 2021, doi: 10.21205/deufmd.2021236710.

[21] H. Saif, M. Fernandez, Y. He, and H. Alani, "On stopwords, filtering and data sparsity for sentiment analysis of twitter," in *Proc. 9th Int. Conf. Lang. Resour. Eval. Lr.*, 2014, pp. 810–817.

[22] C. Silva and B. Ribeiro, "The Importance of Stop Word Removal on Recall Values in Text Categorization," in *Proc. Int. Jt. Conf. Neural Networks*, vol. 3, Aug. 2003, pp. 1661–1666, doi: 10.1109/ijcnn.2003.1223656.

[23] Y. Fan, C. Arora and C. Treude, "Stop Words for Processing Software Engineering Documents: Do they Matter?," in *Proc. IEEE/ACM 2nd Int. Work. Nat. Lang. Softw. Eng. (NLBSE),* 2023, pp. 40–47, doi: 10.1109/NLBSE59153.2023.00016.

[24] G. Gupta and S. Malhotra, "Text Document Tokenization for Word Frequency Count using Rapid miner," *Int. J. Comput. Appl.*, vol.12, pp. 24-26, Aug. 2015.
[25] T. Korenius, J. Laurikkala, K. Järvelin and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proc. Int. Conf. Inf. Knowl. Manag.*, 2004, pp. 625–633, doi: 10.1145/1031171.1031285.
[26] A. Barbaresi, "Simplemma". *Zenodo, Jan.* 20, 2023. doi: 10.5281/zenodo.7555188.
[27] W. Aljedaani et al., "Sentiment analysis on Twitter data integrating TextBlob and deep learning models: The case of US airline industry," *Knowledge-Based Syst.*, vol. 255, 2022, Art. no. 109780, doi: 10.1016/j.knosys.2022.109780.
[28] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector," in Proc. *1st International Conference on Learning Representations*, 2013, pp.1-12.
[29] A. K. Singh and M. Shashi, "Vectorization of text documents for identifying unifiable news articles," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 7, pp. 305–310, 2019, doi: 10.14569/ijacsa.2019.0100742.
[30] G. Yeşiltaş and T. Güngör, "Intrinsic and Extrinsic Evaluation of Word Embedding Models," in Proc. *Innovations in Intelligent Systems and Applications Conference* (ASYU), 2020, pp. 1-6, doi: 10.1109/ASYU50717.2020.9259855.
[31] D. Jatnika, M. A. Bijaksana and A. A. Suryani, "Word2vec model analysis for semantic similarities in English words," in *Proc. Comput. Sci.*, vol. 157, Sept.. 2019, pp. 160–167, doi: 10.1016/J.PROCS.2019.08.153.
[32] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global vectors for word representation," in Proc. Conference on Empirical Methods in Natural Language Processing, Oct. 2014, pp. 1532–1543. doi: 10.3115/v1/d14-1162.
[33] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
[34] J. Lakoumentas, J. Drakos, M. Karakantza, G. Sakellaropoulos, V. Megalooikonomou and G. Nikiforidis, "Optimizations of the naïve-Bayes classifier for the prognosis of B-Chronic Lymphocytic Leukemia incorporating flow cytometry data," *Comput. Methods Programs Biomed*., vol. 108, no. 1, pp. 158–167, 2012, doi: 10.1016/j.cmpb.2012.02.009.
[35] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn*., vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1023/A:1022627411411.
[36] S. Metlek and K. Kayaalp, "Derin Öğrenme ve Destek Vektör Makineleri İle Görüntüden Cinsiyet Tahmini", DUBİTED, vol. 8, no. 3, pp. 2208–2228, 2020, doi: 10.29130/dubited.707316.
[37] M. Kantardzic, "Decision Trees and Decision Rules," in Data Mining: Concepts, Models, Methods, and Algorithms, 3rd ed. Columbia, MD, U.S.A.: Wiley-IEEE Press, 2019, sec. 6, pp. 197-229
[38] R. Wang, "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review," in *Proc. Phys. Procedia*, vol. 25, 2012, pp. 800–807, doi: 10.1016/j.phpro.2012.03.160.
[39] E. Scornet, G. Biau and J. P. Vert, "Consistency of random forests," *Ann. Stat.*, vol. 43, no. 4, pp. 1716–1741, Aug. 2015, doi: 10.1214/15-AOS1321.
[40] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000, doi: 10.1093/bioinformatics/16.5.412.

Conflict of Interest Notice
The authors declare that there is no conflict of interest regarding the publication of this paper**.**

Ethical Approval and Informed Consent
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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 61-76
DOI: 10.35377/saucis...1418505
SAKARYA UNIVERSITY
RESEARCH ARTICLE

Automatic Maize Leaf Disease Recognition Using Deep Learning

Muhammet ÇAKMAK [1]

[1]Sinop University, Department of Computer Engineering, Sinop, Türkiye

Corresponding author:
Muhammet Çakmak, Sinop University, Department of Computer Engineering
mcakmak@sinop.edu.tr

Article History:
Received: 12.01.2024
Accepted: 18.03.2024
Published Online: 18.03.2024

ABSTRACT
Maize leaf diseases exhibit visible symptoms and are currently diagnosed by expert pathologists through personal observation, but the slow manual detection methods and pathologist's skill influence make it challenging to identify diseases in maize leaves. Therefore, computer-aided diagnostic systems offer a promising solution for disease detection issues. While traditional machine learning methods require perfect manual feature extraction for image classification, deep learning networks extract image features autonomously and function without pre-processing. This study proposes using the EfficientNet deep learning model for the classification of maize leaf diseases and compares it with another established deep learning model. The maize leaf disease dataset was used to train all models, with 4188 images for the original dataset and 6176 images for the augmented dataset. The proposed models were compared with ResNet50, VGG19, DenseNet121 and Inception V3 models according to their accuracy, sensitivity, F1-Score and precision values. The EfficientNet B6 model achieved 98.10% accuracy on the original dataset, while the EfficientNet B3 model achieved the highest accuracy of 99.66% on the augmented dataset.

Keywords: Deep learning, Transfer learning, Plant disease classification

1. Introduction

Precisely detecting diseases in maize leaves is critical to sustain food policies and ensure proper agricultural practices. In addition, early detection of diseases in the leaves of maize plants is of great importance in preventing time and financial losses. Some maize leaf diseases are difficult to diagnose because they have no outward signs of disease. However, most maize leaf diseases show visible symptoms. Typically, an expert pathologist diagnoses diseases on maize leaves through visual observation [1]. When diagnosing maize leaf diseases, a plant pathologist must observe the characteristic symptoms of the disease. Experienced pathologists may still struggle to diagnose certain diseases, as climate change and the rapid spread of maize leaf diseases to previously unaffected regions can alter disease courses and make accurate diagnosis challenging [2].

Applications of the machine and deep learning models in many fields, such as insect detection [3], fungus detection [4], healthcare [5], [6], [7], [8], and education [9], are rapidly increasing. Developing intelligent systems capable of automatically and precisely diagnosing maize leaf diseases benefits engineers seeking to boost production. Moreover, creating a mobile application that can assist farmers struggling with diseases and lacking technical support infrastructure is a significant advancement [10]. Recent advances in deep learning models have enabled the creation of systems that can accurately and quickly classify plant species and diagnose plant diseases. Currently, artificial intelligence techniques in plant disease classification and diagnosis are widespread [11]. Over the last ten years, numerous artificial intelligence models have been suggested for identifying and detecting plant diseases [12], [13], [14]. In their study, the authors employed the Support Vector Machine (SVM) algorithm to identify and classify diseases in sugar beet crops. [11]. Al-Hiary et al. deduced the texture and color characteristics of the diseased areas in 5 different plant leaves using K-means. They then classified the diseases from the obtained features using an Artificial Neural Network [15]. The authors of a different study proposed a Particle Swarm Optimization approach for classifying cotton leaf diseases. This method selects features based on texture, edge, and color using particle swarm optimization, and a cross-information neural network is used to classify the six types of cotton leaf diseases [16]. Mokhtar et al. identified the disease-causing virus species in tomato leaves using the Support Virtual Machine [17]. The authors used SVM to identify the disease in three grapevine leaves[18]. Johannes et al. proposed a mobile-based software that uses a Naive Bayes classifier to detect images of wheat diseases [19]. Chen et al proposed an automated disease recognition logistic algorithm using the group method to detect plant diseases [20]. The feature extraction process is a critical

issue in machine learning, as it can significantly impact classification accuracy. Advances in technology have resulted in significant increases in the speed and capacity of graphics processing units and central processing units, which have facilitated the development of deep learning methods that can achieve high performance without the need for manual feature extraction [21], [22].

Many processing layers and neurons in deep neural networks allow them to efficiently process large and complex data, such as image and voice recognition tasks [23]. As a result, deep learning methods are frequently used to detect medical diseases [24], [25], [26]. Bozkurt F. used a handcrafted features-based framework to diagnose COVID-19 [27]. On the other hand, there is a growing trend in utilizing deep learning techniques for detecting and classifying plant diseases [28]. Chen et al. proposed an ensemble network named Es-MbNet, which was developed by combining three lightweight CNNs, utilizing transfer learning and a two-stage training approach to enhance the identification of subtle plant lesion features, achieving an impressive average accuracy of 99.37% on a local dataset and 99.61% on the PlantVillage dataset [29]. Another study conducted by Chen et al. utilized VGG deep learning architecture to detect diseases in maize and rice leaves, achieving accuracy rates of 91.83% and 92%, respectively [30]. The study introduces CoffeeNet, a novel deep-learning model tailored for the early detection and categorization of various coffee plant leaf infections, addressing challenges posed by image distortions such as color variations, lighting changes, and size alterations. Leveraging a spatial channel attention strategy based on the ResNet-50 model within the CenterNet framework, CoffeeNet achieves an impressive classification accuracy of 98.54% and a mean Average Precision (mAP) of 0.97, demonstrating its efficacy in localizing and categorizing complex coffee leaf anomalies [31]. Too et al. employed VGG16, ResNet152, ResNet101, ResNet50, and DenseNets121 deep learning methods to detect leaf diseases. Among these methods, DenseNets121 achieved the highest accuracy of 99.75%, owing to its efficient computation time and reduced number of parameters [32]. In a separate study, the authors proposed a 9-layer CNN architecture for classifying plant diseases. They compared this method with logistic regression, SVM, K-NN, and decision trees. The authors used a dataset of 55,636 images and 39 classes for testing and training. The proposed CNN network achieved a classification accuracy of 96.46% in identifying plant diseases [33]. In another research, vision transformer (ViT)-like techniques are employed for plant disease identification, introducing an innovative edge-feature guidance (EFG) module that enhances the extraction of localized features. Through integration with leading methods like ViT, PVT, and Swin, the proposed ViT-based EFG module demonstrates superior feature extraction performance and outperforms existingodels across Paddy, Wheat, Cabbage, and Coffee datasets [34]. Kusumo et al. (year) employed speeded-up robust features (SURF), Oriented FAST, scale-invariant feature transform (SIFT), and object detector methods such as histogram of oriented gradients. They rotated BRIEF (ORB) to detect RGB colors in maize leaves. The authors compared these features with Naive Bayes (NB), SVM, Random Forest (RF), and Decision Tree (DT) methods [35]. Hassan et al. proposed two classification methods for diseases of maize, potato, and tomato plants: shallow VGG with Xgboost and shallow VGG with RF and deep learning networks. The authors found that Xgboost yielded the highest accuracy rate in classifying maize, potato, and tomato leaf defects with rates of 94.47%, 98.74%, and 93.91%, respectively [36]. Atilla et al. utilized various CNN models, including AlexNet, ResNet50, VGG16, Inception, and EfficientNet, to classify 54,305 images of plant diseases with an accuracy of 98.42% [37]. Fayyaz et al. proposed a CNN architecture that combines SqueezeNet and ShuffleNet for early detection of leaf blight in plants. The authors also employed SVM for classification and the CIELAB color space to enhance accuracy. They achieved a 98% accuracy rate in classifying leaf blights [23]. Elaraby and colleagues classified 25 plant leaf diseases using AlexNet and Particle Swarm optimization. The proposed deep learning architecture achieved an accuracy rate of 98.93% in classifying plant diseases [39]. As noted in the literature, the utilization of machine learning and deep learning techniques for diagnosing plant diseases is rapidly expanding. However, there are still gaps in applying new deep-learning architectures to detect diseases in maize leaves. Specifically, there is a need for models that can be trained quickly, have fewer parameters, and exhibit high performance.

The current study presents a deep learning architecture for classifying maize leaf diseases, utilizing a CNN EfficientNet. The proposed CNN architecture is then compared to ResNet50, VGG19, DenseNet121, and Inception V3 CNN architectures. The remaining sections of this study are structured as follows: Section 2 describes the dataset used and the deep neural network architectures employed. In contrast, Section 3 outlines the experimental methodology. Section 4 presents the study's results and provides a detailed discussion of the findings, and the study is ultimately concluded in Section 5.

2. Materials and Methods

2.1 Dataset

This research used a dataset of maize leaf diseases, which comprised 4,188 images of colored leaves with varying sizes. The dataset included four categories of maize leaves, of which 1306 images represented Common Rust, 574 images represented Gray Leaf Spot, and 1146 images represented Blight maize leaf disease. Additionally, 1162 images represented healthy maize leaves. The dataset was composed of three diseased maize leaves and one healthy maize leaf. Figure 1 displays original dataset images of four types of maize leaf diseases as well as healthy and diseased leaves.

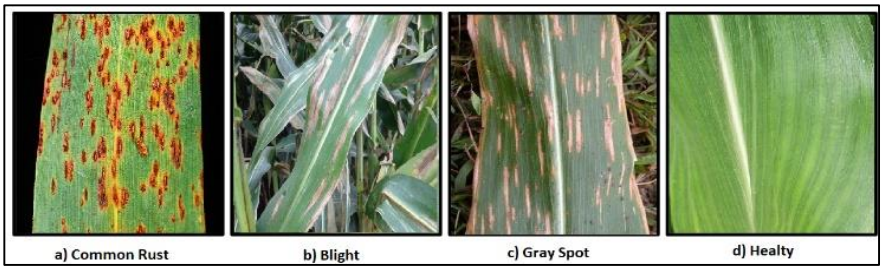


Figure 1 Original dataset: a) Common rust disease, b) Blight disease, c) Gray spot disease, d) Healthy leaves

The dataset used in this study was augmented using various techniques, resulting in 5932 images. Horizontal flip, 20% rotation, 20% width shift, 20% height shift, and zoom were applied to create the augmented dataset. Figure 2 shows visual representations of the maize leaf images in the augmented dataset.

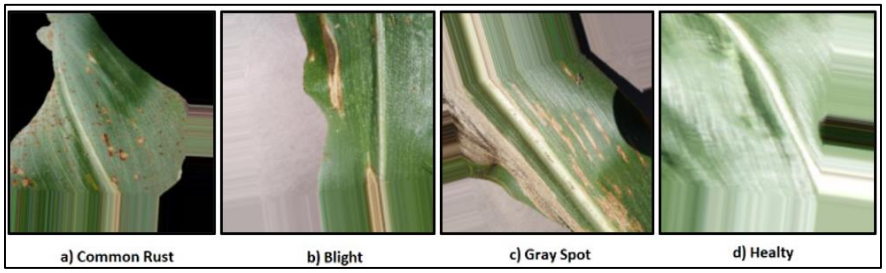


Figure 2 Augmented dataset: a) Common rust disease, b) Blight disease, c) Gray spot disease, d) Healthy leaves

2.2 Transfer Learning

Transfer learning is a machine learning technique that involves leveraging the knowledge acquired from solving a previous problem to tackle a new and similar problem. In traditional machine learning, the learning process occurs while performing different tasks [40]. However, transfer learning involves utilizing source tasks obtained from machine learning methods for new tasks [41]. Figure 3 illustrates the schematic representation of traditional and transfer learning. Transfer learning utilizes the knowledge gained from a pre-trained network, leading to higher accuracy and time savings than training the model from scratch.

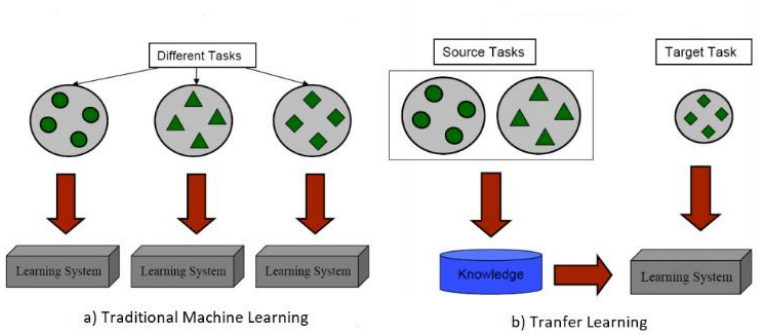


Figure 3 Schematic representation of traditional and transfer learning [41]

2.3 Deep Learning Models

In this study, the comparative performance of the proposed EfficientNet deep learning architecture has been evaluated against several state-of-the-art CNN architectures, including ResNet50, VGG19, DenseNet121, and Inception V3.

2.3.1 ResNet50

The Residual Networks (ResNets) were developed to overcome the challenges posed by numerous non-linear layers, such as not being able to learn identity maps and the problem of degradation. The ResNets architecture aims to ease the network's

training process. ResNet50 is a model with many stacked units consisting of pooling and convolution layers. This model has a depth of 50 layers and 26M parameters and employs 3×3 filters for input images of 224×224 pixels [42]. It uses skip connections to allow the propagation of information across layers. Additionally, the ResNet50 architecture has one MaxPooling layer, one Average Pooling layer, and 48 Convolution layers. Figure 4 illustrates the schematic ResNet50 architecture.

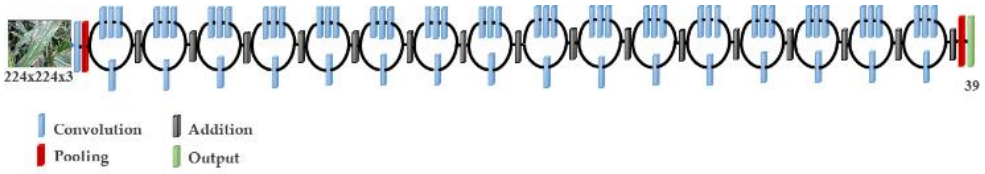


Figure 4 ResNet50 schematic architecture [42]

2.3.2 VGG19

The VGG architecture was developed to enable deep convolutional networks to recognize large-scale images. The VGG19 model, a variant of the VGG architecture, comprises 5 MaxPooling layers, 16 convolution layers, 3 Fully Connected layers, and 1 SoftMax layer. VGG19 achieved the top rank in the Large-Scale Visual Recognition Competition (ILSVRC) in 2014 [43]. It has 138 million parameters and was trained on more than one million images. To reduce the number of parameters, VGG19 uses 3x3 kernels. The architecture of VGG19 consists of 19 layers, and its input layer image size is 224x224 pixels. A schematic of the VGG19 architecture is shown in Figure 5.

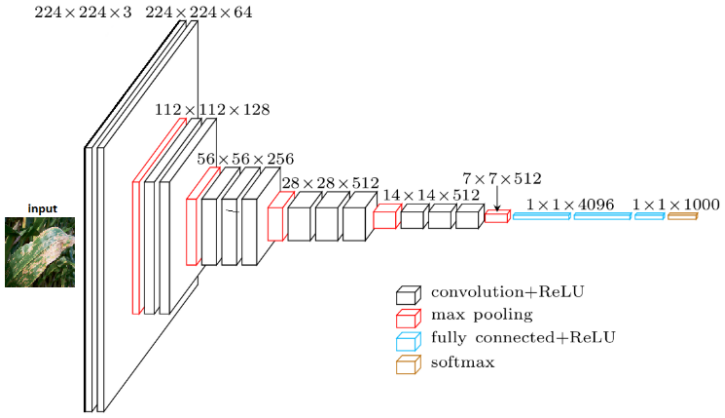


Figure 5 Schematic representation of VGG19 [44]

2.3.3 DenseNets121

DenseNets [45] is an architecture that aims to increase the depth of deep convolutional networks and train the network better by establishing short connections between layers. DenseNets uses fewer parameters than other CNN architectures, as there is no need to learn extra feature maps. Also, its layers are very narrow, and only those layers add a small feature map. DenseNets connects directly between layers to improve the flow of information between layers. Figure 6 shows the 5-layer DenseNet block diagram.

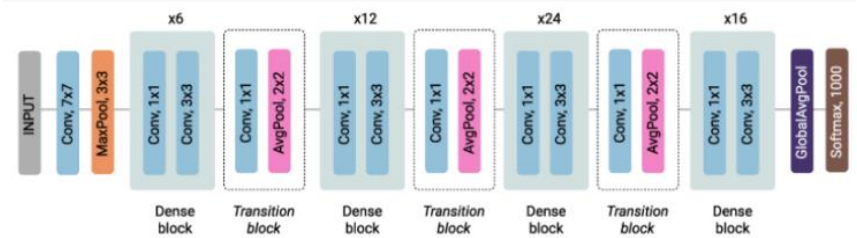


Figure 6 A 5-layer dense block with an expansion rate of k= 4 [45]

The implementation of DenseNet architecture consists of three types of blocks, namely the convolution block, dense block, and transition block. The convolution block, or the main block, connects the dense blocks [46]. The thick block is the main component of DenseNet. Transition blocks are situated between the dense blocks and serve to decrease the dimensionality of the feature map. A schematic illustration of the block structure of the DenseNet architecture is provided in Figure 7. The input layer image size for DenseNet121 is 224x224.

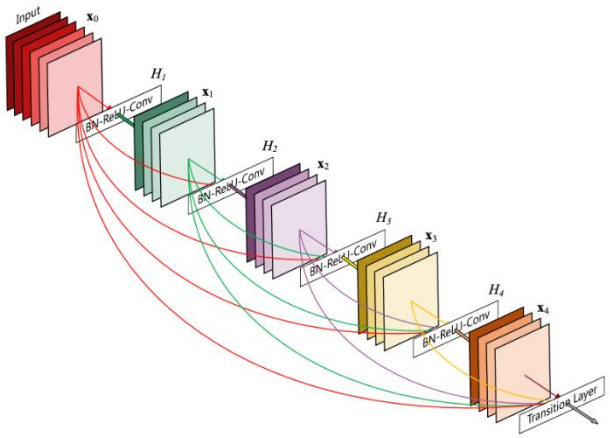


Figure 7 DenseNets block architecture [47]

2.3.4 InceptionV3

The Inception architecture initially called GoogleNet in 2014, is a pre-trained network model [25]. Google developed the 3rd Generation of this deep learning architecture, known as Inception V3. Inception V3 uses a factorization approach to improve the deep learning network's performance by reducing the number of parameters and connections [48]. The network structure of Inception V3 comprises various components, such as convolutions, average and maximum pooling, dropouts, concerts, and fully connected layers. This model has a depth of 48 layers and can process images of size 299x299 pixels. The model's architecture is illustrated in Figure 8.

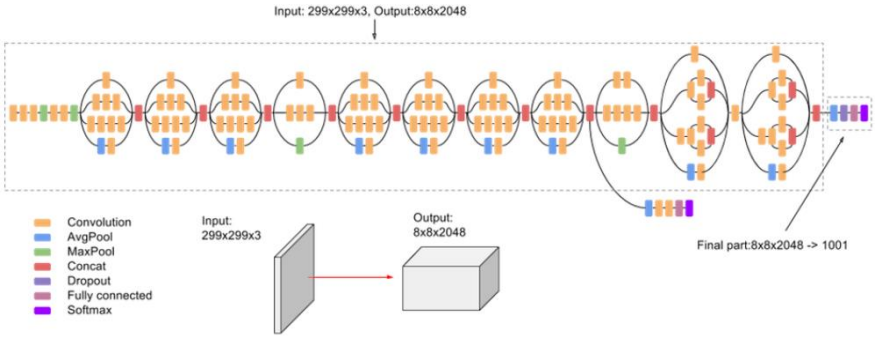


Figure 8 Schematic Inception V3 architecture [49]

2.3.5 EfficientNet

The Inception architecture, also known as GoogleNet, is a pre-trained network model that was introduced by Google in 2014 [49]. Inception V3, the third Generation of this architecture, utilizes the factorization method to enhance the deep learning network's performance by minimizing the number of parameters and connections. The network comprises convolutions, average and maximum pooling layers, dropout layers, concatenation layers, and fully connected layers. The Inception V3 model has 48 layers and requires input images of size 299x299 pixels [50]. The model's architectural representation can be seen in Figure 9.

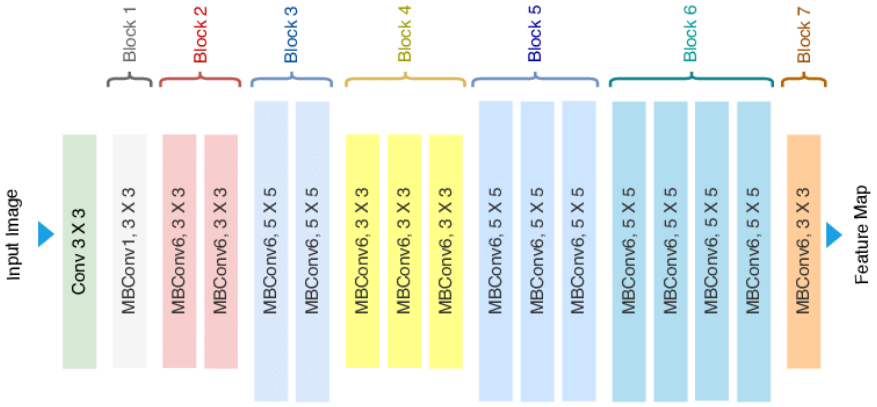


Figure 9 Schematic EfficientNet B0 architecture [41], [51]

3. Experimental Study

3.1. Experimental Setup

The deep learning models were trained in the Google cloud environment using a GPU-accelerated system. The training was performed on a Tesla T4 GPU and an Intel Xeon 2.20 GHz CPU with 16 GB RAM. For the transfer learning design, all programs were written in Python 3 programming language, and the Keras 2.3.1 training framework was utilized.

3.2. Training

In this research, we evaluated deep learning models' efficacy in categorizing maize leaf diseases such as Blight, Common Rust, Gray leaf spot, and Healthy, using both original and augmented datasets. Table 1 shows the distribution of training, validation, and test data between these two datasets. The original dataset encompassed 4,188 images, segmented into training, validation, and test groups. Specifically, the training group comprised 3,769 images, accounting for 90% of the dataset, while the validation and test groups had 209 and 210 images, respectively. To enhance model accuracy, we expanded our dataset through diverse image augmentation methods like rotation, scaling, and mirroring. Consequently, the augmented dataset contained 5,932 images partitioned into the same three categories. Notably, the augmented training set consisted of 5,338 images (90% of the total), while both validation and test subsets included 297 images each.

Table 1 Original and augmented data in the dataset

	Total data	Training (90%)	Validation (5%)	Test (5%)
Original dataset	4.188	3.769	209	210
Augmented dataset	5.932	5.338	297	297

In this study, we adopted transfer learning techniques by repurposing established CNN architectures and fine-tuning them to expedite the learning process. To facilitate this, we incorporated the ImageNet dataset, which boasts approximately 1.2 million images spanning 1000 distinct categories, as a foundational basis for transfer learning. Utilizing pre-existing weight values significantly streamlined our deep learning models' training phase. The final Fully Connected (FC) layers across all models were reconfigured to yield four specific outputs tailored to address the objectives of our study. We designated the CNN layers for training while employing Softmax as the chosen activation function and categorical cross-entropy to quantify the loss. Furthermore, to optimize the training regimen, we implemented an early stopping mechanism, maintaining a consistent threshold of 3 and a loss threshold set at 1e-3.

The pre-trained models in this study were optimized using the same optimization method as the ImageNet dataset. Adam's learning rate was 0.001, while SDG was set to 0.01. A validation value limit of 1 was used for all models. Normalization was applied to all image data used in the dataset. The image data was resized to different sizes for each transfer learning model. ResNet50, VGG19, DenseNet121, and EfficientNet B0 were resized to 224x224 pixels, while InceptionV3 was resized to 299x299 pixels. The EfficientNet network models had different input image sizes. To ensure a fair evaluation of all EfficientNet models, a pixel value of 224x224 was selected for our experimental study. The resolution values of the selected models can be found in Table 2. In this research, the batch mechanism was utilized to update the bias and weights in training the models. To comply with the hardware resources, the maximum value of the batch was established as 32.

Table 2 Transfer learning model input value and parameters

Transfer Learning Models	Image Input Values	Model Parameters
ResNet50	224x224	25,636,712
VGG19	224x224	138,357,544
DenseNet121	224x224	7,978,856
InceptionV3	299x299	23,851,784
EfficientNet		
B0	224x224	5,330,571
B1	240x240	7,856,239
B2	260x260	9,177,569
B3	300x300	12,320,535
B4	380x380	19,466,823
B5	456x456	30,562,527
B6	528x528	43,265,143
B7	600x600	66,658,687

We set the bias l1 kernel regularizer value to 0.006 and the bias l2 kernel regularizer value to 0.016 for fine-tuning. ReLu was used as the activation function in the layers, while Softmax was used as the output activation function. We applied a dropout rate of 40%. For batch normalization, we chose a momentum of 0.99 and an epsilon value of 0.001.

3.3. Performance Metrics

A multi-class assessment was carried out on the maize leaf dataset, comprising four categories. Model performance was assessed using True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values derived from the confusion matrix, as depicted in equations (1), (2), (3), and (4). Model comparison was based on F1-Score (F1_Score), Accuracy (ACC), Sensitivity (Sen), and Precision (Precision) metrics, calculated using equations (1), (2), (3), and (4). For a given class x,

$$\boldsymbol{Sen}(\mathbf{x}) = \frac{\boldsymbol{TP}(\mathbf{x})}{\boldsymbol{TP}(\boldsymbol{x}) + \boldsymbol{FN}(\mathbf{x})} \tag{1}$$

$$\boldsymbol{F1\_Score}(\boldsymbol{x}) = \frac{\mathbf{2} * \boldsymbol{Pre}(\boldsymbol{x}) * \boldsymbol{Sen}(\boldsymbol{x})}{\boldsymbol{Pre}(\boldsymbol{x}) + \boldsymbol{Sen}(\boldsymbol{x})} \tag{2}$$

$$\boldsymbol{Acc}(\boldsymbol{x}) = \frac{\boldsymbol{TP}(\boldsymbol{x}) + \boldsymbol{TN}(\boldsymbol{x})}{\boldsymbol{TP}(\boldsymbol{x}) + \boldsymbol{FN}(\boldsymbol{x}) + \boldsymbol{TN}(\boldsymbol{x}) + \boldsymbol{FP}(\boldsymbol{x})} \tag{3}$$

$$\boldsymbol{Pre}(\boldsymbol{k}) = \frac{\boldsymbol{TP}(\boldsymbol{k})}{\boldsymbol{TP}(\boldsymbol{k}) + \boldsymbol{FP}(\boldsymbol{k})} \tag{4}$$

4. Results and Discussions

Maize is a staple food in many countries, and its cultivation is essential for food security. However, the crop is susceptible to various diseases that can significantly affect its yield. To address this issue, researchers have developed deep-learning models to classify maize leaf diseases from images. In this article, we compare the performance of several deep learning models for maize leaf disease classification, including ResNet50, VGG19, DenseNet121, InceptionV3, EfficientNet B0, B1, B2, B3, B4, B5, B6, and B7.

The research conducted involved utilizing both the original and augmented datasets in all experimental studies. The average results of the original dataset for each model are presented in Table 3, while the outcomes of the augmented dataset are presented in Table 3.

We use the same original dataset to evaluate the models, which contains images of maize leaves affected by Blight, Common Rust, Gray leaf spot, and Healthy. We report the accuracy, sensitivity, F1-score, and precision of each model.

Firstly, ResNet50 achieved an accuracy of 96.67%, making it one of the best-performing models in our comparison. It also achieved a sensitivity of 95.70%, an F1-score of 95.67%, and a precision of 95.66%. These results suggest that ResNet50 is a reliable maize leaf disease classification model.

In contrast, VGG19 had a lower accuracy of 91.43%, a sensitivity of 87.99%, an F1-score of 89.34%, and a precision of 92.94%. Although these results are lower than those of ResNet50, VGG19 still provides reasonable accuracy and precision for maize leaf disease classification.

DenseNet121 outperformed ResNet50 in terms of accuracy, achieving an accuracy of 97.14%, a sensitivity of 96.60%, an F1-score of 96.80%, and a precision of 97.02%. This suggests that DenseNet121 is a highly accurate and reliable maize leaf disease classification model.

InceptionV3 also achieved high accuracy, with an accuracy of 97.62%, a sensitivity of 96.05%, an F1-score of 96.87%, and a precision of 98.01%. These results suggest that InceptionV3 is a reliable maize leaf disease classification model, particularly when high precision is required.

EfficientNet B6 achieved the highest accuracy of 98.10%, a sensitivity of 98.28%, an F1-score of 97.87%, and a precision of 97.60%. This indicates that EfficientNet B6 is a highly accurate and reliable maize leaf disease classification model. Among the models tested on the original dataset, EfficientNet B6 achieved the highest accuracy of 98.10%, followed by EfficientNet B2 with 97.62% accuracy. VGG19 achieved the lowest accuracy of 91.43%.

Table 3 Performance metrics of deep learning models for the original dataset

Transfer Learning Models	**Avg Acc (%)**	**Avg Sen (%)**	**F1-Score (%)**	**Avg Pre (%)**
ResNet50	96.67	95.70	95.67	95.66
VGG19	91.43	87.99	89.34	92.94
DenseNet121	97.14	96.60	96.80	97.02
InceptionV3	97.62	96.05	96.87	98.01
EfficientNet B0	95.71	95.25	95.33	95.51
EfficientNet B1	96.19	95.17	95.65	96.23
EfficientNet B2	97.62	96.92	97.18	97.46
EfficientNet B3	97.14	97.00	96.81	96.64
EfficientNet B4	95.24	93.84	94.09	94.38
EfficientNet B5	96.67	96.52	96.64	96.89
EfficientNet B6	**98.10**	**98.28**	**97.87**	**97.60**
EfficientNet B7	97.14	96.03	96.27	96.54

The augmented dataset originated from the original dataset by integrating diverse image augmentation strategies. This augmented dataset served as the training and evaluation set for models identical to those used with the original data. Notably, when tested on the augmented dataset, the ResNet50 model exhibited an impressive accuracy rate of 98.32%. Additionally, the model showcased a commendable sensitivity of 98.55%, underscoring its proficiency in accurately detecting diseased leaf images. Nonetheless, the precision of this model stood at 97.22%, suggesting instances where it misclassified healthy leaves as diseased, leading to certain false positives.

VGG19, achieved an accuracy of 97.64% on the augmented dataset. Its sensitivity was 96.64%, lower than ResNet50, but its precision was higher at 97.27%. This suggests that VGG19 was better at correctly identifying diseased leaves but had a higher chance of incorrectly classifying healthy leaves as diseased.

DenseNet121 achieved an accuracy of 96.97% on the augmented dataset, with a sensitivity of 96.36% and a precision of 95.84%. Its F1-score was 96.08%, which measures the balance between accuracy and sensitivity. DenseNet121 had a lower sensitivity than ResNet50 and VGG19, but it had a higher precision.

InceptionV3 achieved an accuracy of 97.31% on the augmented dataset, with a sensitivity of 97.36% and a precision of 96.26%. Its F1-score was 96.75%, similar to VGG19 but lower than ResNet50. InceptionV3 had a higher sensitivity compared to DenseNet121 but a lower precision.

EfficientNet models, including B0, B1, B2, B3, B4, B5, B6, and B7, achieved high accuracies ranging from 97.98% to 99.66% on the augmented dataset. The models had high sensitivities ranging from 97.26% to 99.71%, which indicates that they could correctly identify a high percentage of the images affected by the diseases. Among the models tested on the augmented dataset, EfficientNet B3 achieved the highest accuracy of 99.66%, followed by EfficientNet B6 with 98.99%

accuracy. VGG19 achieved the lowest accuracy of 97.64%.

Table 4 Performance metrics of deep learning models for the augmented dataset

Transfer Learning Models	Avg Acc (%)	Avg Sen (%)	F1-Score (%)	Avg Pre (%)
ResNet50	98.32	98.55	97.79	97.22
VGG19	97.64	96.64	96.93	97.27
DenseNet121	96.97	96.36	96.08	95.84
InceptionV3	97.31	97.36	96.75	96.26
EfficientNet B0	97.98	97.26	97.41	97.57
EfficientNet B1	98.32	98.19	97.89	97.62
EfficientNet B2	97.98	97.26	97.41	97.57
EfficientNet B3	**99.66**	**99.71**	**99.55**	**99.39**
EfficientNet B4	97.98	97.92	97.47	97.07
EfficientNet B5	98.32	98.22	97.90	97.62
EfficientNet B6	98.99	98.79	98.63	98.48
EfficientNet B7	98.32	97.54	97.69	97.85

The accuracy values of all models in the original dataset are presented in Figure 10, while Figure 11 displays the accuracy values in the augmented dataset. Accuracy is measured by dividing the number of correctly classified samples by the total number of samples. The EfficientNet B6 model recorded the highest accuracy of 98.10% in the original dataset, while the EfficientNet B3 model achieved the highest accuracy of 99.66% in the augmented dataset. Conversely, VGG19 and DenseNet121 had the lowest accuracy values in both datasets. These findings indicate that the accuracy value in the augmented dataset is greater than that of the original dataset.

In summary, the deep learning frameworks examined in this research exhibit encouraging outcomes in identifying maize leaf diseases through image analysis. Notably, the EfficientNet architectures consistently manifest elevated accuracy levels across the initial and augmented datasets. Furthermore, the augmented dataset notably enhances the efficacy of most models, underscoring the pivotal role of image augmentation methodologies in refining the accuracy of deep learning frameworks.
Figure 10 showcases the accuracy metrics for all models based on the original dataset, whereas Figure 11 illustrates the accuracy figures from the augmented dataset. Accuracy is computed by the ratio of accurately classified samples to the total sample count. Noteworthy, the EfficientNet B6 model led with a peak accuracy of 98.10% when evaluated against the original

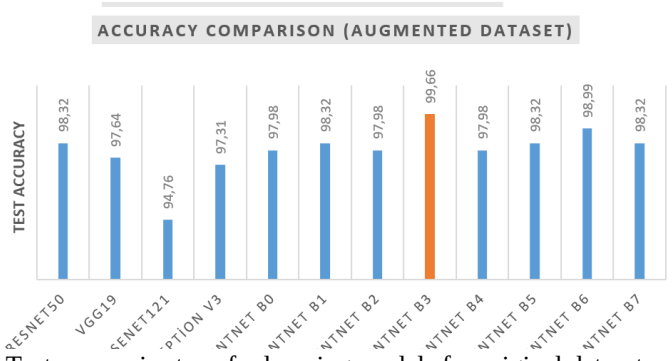


Figure 10 Test accuracies transfer learning models for original dataset.

dataset, whereas the EfficientNet B3 model excelled with an accuracy of 99.66% on the augmented dataset. In contrast, VGG19 and DenseNet121 consistently exhibited the least accuracy across both datasets. Such results strongly suggest that the augmented dataset consistently yields higher accuracy rates than its original counterpart.

Figure 1 Test accuracies transfer learning models for the augmented dataset.

Table 5 presents the performance metrics, including TP, FP, TN, FN, sensitivity, accuracy, F1-Score, and precision values, for each class of the EfficientNet B6 model, which demonstrated superior performance in the original data set. On the other hand, Table 6 displays the corresponding performance metrics for each class of the EfficientNet B3 model, which exhibited the best performance in the augmented data set, including TP, FP, TN, FN, sensitivity, F1-Score, and precision values.

The first model, EfficientNet B6, was trained on the original dataset, while the second model, EfficientNet B3, was trained on an augmented dataset. Let's first look at the results of EfficientNet B6 on the original dataset. For Blight, the model achieved a TP of 54, TN of 152, FP of 0, and FN of 4, resulting in a sensitivity of 93.10, F1-score of 96.43, and precision of 100. For Common Rust, the model achieved a TP of 66, TN of 140, FP of 2, and FN of 0, resulting in a sensitivity of 100, F1-score of 98.51, and a precision of 97.06. For the Gray leaf spot, the model achieved a TP of 28, TN of 178, FP of 2, and FN of 0, resulting in a sensitivity of 100, F1-score of 96.55, and precision of 93.33. Lastly, for Healthy, the model achieved a TP of 58, TN of 148, FP of 0, and FN of 0, resulting in a sensitivity of 100, F1-score of 100, and precision of 100.

Table 5 EfficientNet B6 Model original dataset classification performance

Class	**TP**	**TN**	**FP**	**FN**	**Sen(%)**	**F1-Score(%)**	**Pre(%)**
Blight	54	152	0	4	93.10	96.43	100.00
Common Rust	66	140	2	0	100.00	98.51	97.06
Gray Leaf Spot	28	178	2	0	100.00	96.55	93.33
Healthy	58	148	0	0	100.00	100.00	100.00

Now, let's look at the results of EfficientNet B3 on the augmented dataset. For Blight, the model achieved a TP of 84, TN of 212, FP of 0, and FN of 1, resulting in a sensitivity of 98.82, F1-score of 99.81, and precision of 100. For Common Rust, the model achieved a TP of 91, TN of 205, FP of 0, and FN of 0, resulting in a sensitivity of 100, F1-score of 100, and precision of 100. For Gray leaf spot, the model achieved a TP of 40, TN of 256, FP of 1, and FN of 0, resulting in a sensitivity of 100, F1-score of 98.77, and precision of 97.56. Lastly, for Healthy, the model achieved a TP of 81, TN of 215, FP of 0, and FN of 0, resulting in a sensitivity of 100, F1-score of 100, and precision of 100.

Table 6 EfficientNet B3 Model Augmented Dataset Classification Performance

Class	**TP**	**TN**	**FP**	**FN**	**Sen(%)**	**F1-Score(%)**	**Pre(%)**
Blight	84	212	0	1	98.82	99.81	100.00
Common Rust	91	205	0	0	100.00	100.00	100.00
Gray Leaf Spot	40	256	1	0	100.00	98.77	97.56
Healthy	81	215	0	0	100.00	100.00	100.00

Comparing the two models, we can see that the model trained on the augmented dataset, EfficientNet B3, outperformed the model trained on the original dataset, EfficientNet B6. In particular, EfficientNet B3 achieved higher sensitivities for all four classes, indicating a better ability to lassify diseased leaves correctly. Additionally, EfficientNet B3 achieved higher F1 scores for three out of four classes, indicating a better balance between precision and recall. Lastly, EfficientNet B3 achieved perfect precision for all four classes, indicating that the model made no false positive predictions. In conclusion, EfficientNet B3 trained on an augmented dataset showed superior performance in classifying maize leaf diseases compared to EfficientNet B6 trained on the original dataset. The results demonstrate the importance of data augmentation in increasing the quality of the dataset and improving the performance of the model. The confusion matrices for the models EfficientNet B6 for the original dataset and EfficientNet B3 for the augmented dataset are given in Figure 12 and Figure 13, respectively. Confusion matrices of both models were compared to evaluate the classification performance. The original dataset model, EfficientNet B6, had a sensitivity of 93.10%, 100%, 100%, and 100% for Blight, Common Rust, Gray leaf spot, and Healthy, respectively. The model correctly classified Blight, Common Rust, and Healthy leaf diseases with high accuracy. However, it struggled with the Gray leaf spot, with only 28 out of 30 images correctly classified, resulting in a sensitivity of 93.10%.

On the other hand, the augmented dataset model, EfficientNet B3, achieved a sensitivity of 98.82%, 100%, 100%, and 100% for Blight, Common Rust, Gray leaf spot, and Healthy, respectively. This model showed better performance than the original dataset model in all four classes of diseases, with higher sensitivity and F1-score. The model correctly classified all images of Common Rust and Healthy, and all but one image of Blight. The Gray leaf spot classification improved significantly, with 40 out of 40 images correctly classified, resulting in a sensitivity of 100%.

Overall, the augmented dataset model, EfficientNet B3, showed better performance in classifying maize leaf diseases than the original dataset model, EfficientNet B6. The improved sensitivity and F1-score of the augmented dataset model are particularly notable for Gray leaf spot classification. The results suggest that the use of augmented datasets can improve the performance of deep learning models in image-based classification of maize leaf diseases.

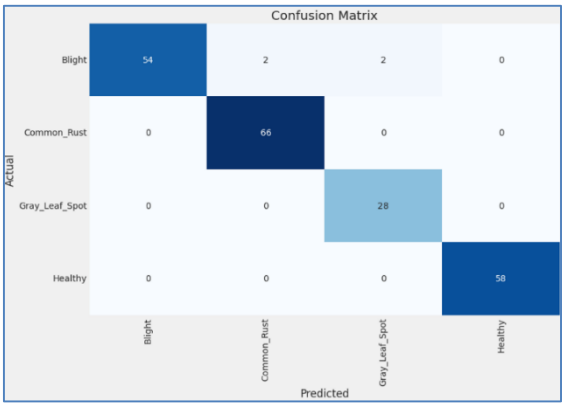


Figure 2 EfficientNet B6 Confusion Matrix for original dataset

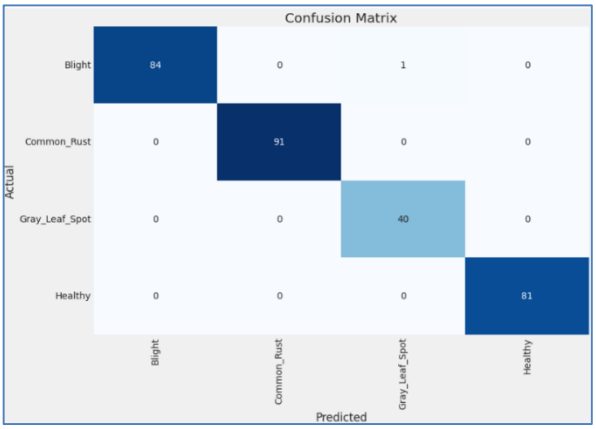


Figure 3 EfficientNet B3 Confusion Matrix for augmented dataset

The results for the original dataset and enriched dataset models of EfficientNet B6 and B3 are presented in Figures 14 and 15, respectively. The effectiveness of the early stopping approach in maintaining higher performance values is demonstrated by the point at which the validation loss begins to decrease. The EfficientNet B6 model achieved its best validation loss and accuracy values in the 42nd and 29th epochs, respectively, as illustrated in Figure 14. Similarly, the EfficientNet B3 model attained its optimal validation loss and accuracy values in the 50th and 26th epochs, respectively, as shown in Figure 15.

As shown in Figure 14 a and Figure 15 a, as the number of epochs increases, the decrease in both training and validation loss is generally due to the increasing learning capacity of the model. With more epochs, the model is exposed to more data, allowing it to gain more insights, resulting in better generalization and lower loss values overall. Additionally, long-term training enables the model to learn both general patterns and finer details over time. Also, increasing epochs increases the model's resistance to overfitting, thus helping to reduce validation loss. It is very important to stop training at the point where the model is performing at its best. The EfficientB6 model trained with the original data set reached its best value in the 42nd epoch, while the EfficientNetB3 model using the augmented data set reached its best value in the 50th epoch.

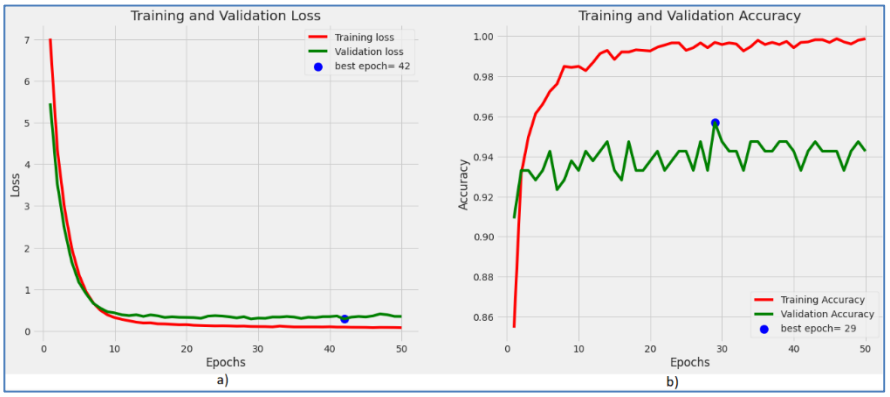


Figure 4 a-b) EfficientNet B6 training and validation loss and accuracy in original dataset

As the number of epochs in Figures 14 b and 15 b increases, training and validation accuracy values increase due to the improved learning capacity of the model. With more epochs, the model is exposed to more data, which allows it to learn general patterns and data properties better, resulting in higher accuracy values. In addition, increasing epochs often contribute to better generalization of the model, initially better adjusting the training data and subsequently improving the generalization ability, leading to higher training and validation accuracy values. Monitoring accuracy values during training and stopping at the optimum performance point ensures the best results. The EfficientB6 model trained with the original data set reached its best value in the 29th epoch, while the EfficientNetB3 model using the augmented data set reached its best value in the 26th epoch.

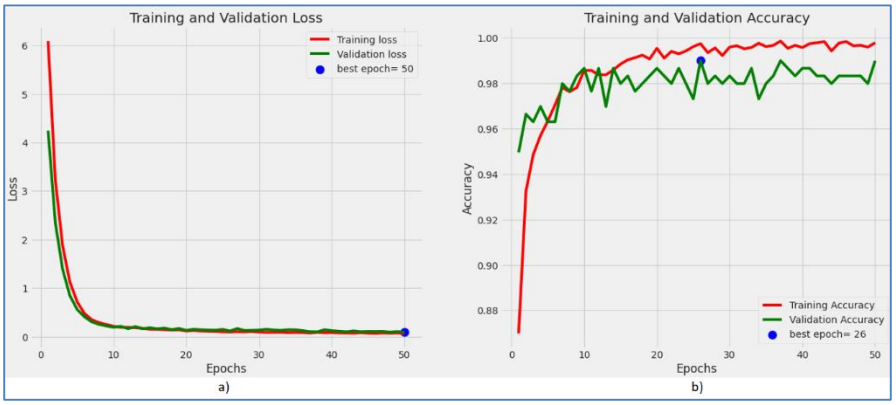


Figure 5 a-b) EfficientNet B3 training and validation loss and accuracy in the augmented dataset

The findings indicate that both the EfficientNet B3 and EfficientNet B6 models achieved high accuracy values of 98% and 99%, respectively, on both the original and augmented datasets. Furthermore, the sensitivity values of these models were also high at 98% and 99%, respectively. These two models demonstrated the highest level of performance among all the models evaluated. The augmented dataset led to an increase in accuracy and sensitivity values across all models. The EfficientNet B3 model, which performed the best on the augmented dataset, exhibited a 2% increase in accuracy and a 3% increase in precision, highlighting the positive impact of increased data on model predictions.

The total number of classification errors for all models is shown in Table 7. In this study, various models, including ResNet50, VGG19, DenseNet121, InceptionV3, and EfficientNet B0 to B7, were evaluated using an original dataset of 210 images and an augmented dataset of 297 images. The aim was to compare the models' classification accuracy and false prediction rate on both datasets.

Results from the original dataset showed that EfficientNet B6 exhibited the best performance with only four false predictions out of 210 images. EfficientNet B3, InceptionV3, and DenseNet121 followed closely with 5 to 6 false predictions. ResNet50, EfficientNet B5, and EfficientNet B7 had a moderate false prediction rate, with 7 to 9 false predictions. VGG19 and EfficientNet B1 had the highest false prediction rate, with 18 and 8 false predictions, respectively.

However, the augmented dataset produced slightly different results. EfficientNet B3 demonstrated the best performance, with only one false prediction out of 297 images. EfficientNet B6 and VGG19 also performed well,, with only 3 and 7 false predictions. DenseNet121 and InceptionV3 had a moderate false prediction rate with 8 to 9 false predictions, while ResNet50 and EfficientNet B0 to B2, B4, and B7 had a higher false prediction rate ranging from 5 to 6.

Overall, results from both datasets demonstrate that EfficientNet B3 and B6 models are suitable for classifying maize leaf diseases. EfficientNet B3 is particularly promising as it achieved the lowest false prediction rate on the augmented dataset.

In general, the augmented dataset improved the models' accuracy and reduced their false prediction rate compared to the original dataset.

In the discussion, we can highlight several key points:

Performance discrepancies between models: The data reveals variations in false prediction rates across different models. For instance, while ResNet50 and DenseNet121 exhibit relatively low false prediction rates in the original dataset compared to VGG19 and InceptionV3, the situation changes in the augmented dataset where VGG19 and InceptionV3 show a decrease in false predictions. Impact of data augmentation: Comparing false prediction rates between the original and augmented datasets sheds light on the effectiveness of data augmentation techniques. In some cases, such as with DenseNet121 and B5-B7 models, false prediction rates increase in the augmented dataset, suggesting that certain augmentation strategies may not universally improve model performance. Model robustness and generalization: The discrepancy in false prediction rates among different models highlights variations in model robustness and generalization capabilities. Models that exhibit consistent performance across both datasets, such as B2 and B3, may indicate more robust architectures that generalize well to augmented data. Potential for further investigation: The observed differences in false prediction rates present avenues for further investigation. Researchers could delve deeper into understanding why certain models perform better in augmented datasets while others do not, leading to insights that could enhance model training strategies and data augmentation techniques. Implications for real-world applications: Discussing the impact of these findings for real-world applications is essential. Understanding model performance under different conditions, such as augmented datasets, is crucial for deploying reliable and robust systems in practical scenarios, such as wildlife monitoring or medical imaging. By incorporating these points into the discussion, the paper can provide a comprehensive analysis of the observed results and their implications for the field of machine learning and computer vision.

In conclusion, the study highlights the importance of choosing an appropriate deep-learning model for classifying maize leaf diseases. The findings suggest that using an augmented dataset can help to improve the accuracy of the models.

Table 7 False prediction values for Transfer Learning Models

	ResNet50	VGG19	DenseNet121	InceptionV3	B0	B1	B2	B3	B4	B5	B6	B7
Total False Prediction in the original dataset	7	18	6	5	9	8	5	6	10	7	4	6
Total False Prediction in Augmented Dataset	5	7	9	8	6	5	6	1	6	5	3	5

5. Conclusion

Within the agricultural sector, identifying and categorizing plant diseases holds paramount significance. Timely identification averts extensive crop devastation and safeguards farmers from substantial economic setbacks. Consequently, the adoption of machine learning techniques to autonomously discern plant diseases has surged in prominence recently.

In the present study, we conducted a performance comparison of EffiicientNet, ResNet50, VGG19, DenseNet121, and Inception V3 models for the classification of maize leaf diseases, including Blight, Common Rust, Gray leaf spot, and Healthy images. The assessment was carried out using both the original and augmented datasets.

The results of the original dataset trained EfficientNet B6 model showed good performance in detecting all four categories of maize leaf diseases, with a sensitivity of 100% for Common Rust and Healthy images. However, the model showed a relatively lower sensitivity of 93.10% for Blight and 96.55% for Gray leaf spot images. The F1 scores were relatively high for all four categories, with a maximum of 100% for Healthy images. The precision of the model was perfect for Blight and Healthy photos, while it was slightly lower for Common Rust and Gray leaf spot images.
On the other hand, the EfficientNet B3 model trained on the augmented dataset showed better results, with a higher sensitivity for all four categories of maize leaf diseases. The model's sensitivity was 98.82% for Blight, 100% for Common Rust and Healthy images, and 97.56% for Gray leaf spot images. The F1 scores were high for all categories, with a maximum of 100% for Healthy images. The precision of the model was perfect for all four categories of maize leaf diseases.
Overall, our results demonstrate that using augmented datasets can significantly improve the performance ofdeep-learning models for the classification of maize leaf diseases. The EfficientNet B3 model trained on the augmented dataset showed better sensitivity and precision results than the EfficientNet B6 model trained on the original dataset. These findings highlight the importance of using augmented datasets in deep learning algorithms for accurate and efficient classification of plant diseases, which can ultimately help in the early detection and prevention of widespread crop damage.

References

[1] S. Sankaran, A. Mishra, R. Ehsani and C. Davis, "A review of advanced techniques for detecting plant diseases," *Comput Electron Agric*, vol. 72, no. 1, pp. 1–13, Jun. 2010, doi: 10.1016/j.compag.2010.02.007.

[2] S. Mishra, R. Sachan and D. Rajpal, "Deep convolutional neural network based detection system for real-time corn plant disease recognition," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 2003–2010. doi: 10.1016/j.procs.2020.03.236.

[3] D. Ozdemir and M. S. Kunduraci, "Comparison of deep learning techniques for classification of the insects in order level with mobile software application," *IEEE Access*, vol. 10, pp. 35675–35684, 2022, doi: 10.1109/ACCESS.2022.3163380.

[4] M. Akın, A. Dağdelen, R. N. Eğinme and D. Özdemir, "doğada kendiliğinden yetişen mantar türlerinin derin öğrenme ile tespiti," *Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi*, Dec. 2023, doi: 10.53608/estudambilisim.1319221.

[5] D. Özdemir and N. N. Arslan, "Analysis of Deep transfer learning methods for early diagnosis of the Covid-19 disease with Chest X-ray images," *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, vol. 10, no. 2, pp. 628–640, Apr. 2022, doi: 10.29130/dubited.976118.

[6] E. Şahin, D. Özdemir and H. Temurtaş, "Multi-objective optimization of ViT architecture for efficient brain tumor classification," *Biomed Signal Process Control*, vol. 91, May 2024, doi: 10.1016/j.bspc.2023.105938.

[7] N. N. Arslan, D. Ozdemir and H. Temurtas, "ECG heartbeats classification with dilated convolutional autoencoder," *Signal Image Video Process*, Feb. 2023, doi: 10.1007/s11760-023-02737-2.

[8] B. Guler Ayyildiz, R. Karakis, B. Terzioglu and D. Ozdemir, "Comparison of deep learning methods for the radiographic detection of patients with different periodontitis stages," *Dentomaxillofac Radiol*, vol. 53, no. 1, pp. 32–42, Jan. 2024, doi: 10.1093/dmfr/twad003.

[9] D. Ozdemir and M. Emin UGUR, "Deniz Yildizlari Vocational and Technical Anatolian High School Ministry of National Education Kocaeli."

[10] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Comput Electron Agric*, vol. 145, pp. 311–318, Feb. 2018, doi: 10.1016/j.compag.2018.01.009.

[11] F. Jiang *et al.*, "Artificial intelligence in healthcare: Past, present and future," *Stroke and Vascular Neurology*, vol. 2, no. 4. BMJ Publishing Group, pp. 230–243, Dec. 01, 2017. doi: 10.1136/svn-2017-000101.

[12] P. Dong, K. Li, M. Wang, F. Li, W. Guo and H. Si, "Maize Leaf Compound Disease Recognition Based on Attention Mechanism," *Agriculture (Switzerland)*, vol. 14, no. 1, Jan. 2024, doi: 10.3390/agriculture14010074.

[13] R. Ahila Priyadharshini, S. Arivazhagan, M. Arun and A. Mirnalini, "Maize leaf disease classification using deep convolutional neural networks," *Neural Comput Appl*, vol. 31, no. 12, 2019, doi: 10.1007/s00521-019-04228-3.

[14] V. Sharma, A. K. Tripathi, P. Daga, M. Nidhi and H. Mittal, "ClGanNet: A novel method for maize leaf disease identification using ClGan and deep CNN," *Signal Process Image Commun*, vol. 120, Jan. 2024, doi: 10.1016/j.image.2023.117074.

[15] H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik and Z. Alrahamneh, "Fast and Accurate Detection and Classification of Plant Diseases," 2011.

[16] P. Revathi and M. Hemalatha, "Advance computing enrichment evaluation of cotton leaf spot disease detection using Image Edge detection," in *2012 3rd International Conference on Computing, Communication and Networking Technologies, ICCCNT 2012*, 2012. doi: 10.1109/ICCCNT.2012.6395903.

[17] U. Mokhtar, M. A. S. Ali, A. E. Hassanien and H. Hefny, "Identifying two of tomatoes leaf viruses using support vector machine," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2015, pp. 771–782. doi: 10.1007/978-81-322-2250-7_77.

[18] X. E. Pantazi, D. Moshou, A. A. Tamouridou and S. Kasderidis, "Leaf disease recognition in vine plants based on local binary patterns and one class support vector machines," in *IFIP Advances in Information and Communication Technology*, Springer New York LLC, 2016, pp. 319–327. doi: 10.1007/978-3-319-44944-9_27.

[19] A. Johannes *et al.*, "Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case," *Comput Electron Agric*, vol. 138, pp. 200–209, Jun. 2017, doi: 10.1016/j.compag.2017.04.013.

[20] J. Chen, H. Yin and D. Zhang, "A self-adaptive classification method for plant disease detection using

GMDH-Logistic model," *Sustainable Computing: Informatics and Systems*, vol. 28, Dec. 2020, doi: 10.1016/j.suscom.2020.100415.

[21] Y. Lecun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[22] H. C. Altunay and Z. Albayrak, "A hybrid CNN + LSTMbased intrusion detection system for industrial IoT networks," *Engineering Science and Technology, an International Journal*, vol. 38, Feb. 2023, doi: 10.1016/j.jestch.2022.101322.

[23] M. Çakmak and Z. Albayrak, "AFCC-r: Adaptive Feedback Congestion Control Algorithm to Avoid Queue Overflow in LTE Networks," *Mobile Networks and Applications*, vol. 27, no. 5, 2022, doi: 10.1007/s11036-022-02011-8.

[24] R. İncir and F. Bozkurt, "A study on effective data preprocessing and augmentation method in diabetic retinopathy classification using pre-trained deep learning approaches," *Multimed Tools Appl*, vol. 83, no. 4, pp. 12185–12208, Jan. 2023, doi: 10.1007/S11042-023-15754-7/TABLES/7.

[25] D. Shen, G. Wu and H.-I. Suk, "Deep Learning in Medical Image Analysis," 2017, doi: 10.1146/annurev-bioeng-071516.

[26] S. Nahzat, F. Bozkurt and M. Yağanoğlu, "White Blood Cell Classification Using Convolutional Neural Network," *Journal of Scientific Technology and Engineering Research*, 2022, doi: 10.53525/jster.1018213.

[27] F. Bozkurt, "A deep and handcrafted features-based framework for diagnosis of COVID-19 from chest x-ray images," *Concurr Comput*, vol. 34, no. 5, 2022, doi: 10.1002/cpe.6725.

[28] A. Ahmad, D. Saraswat and A. El Gamal, "A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools," *Smart Agricultural Technology*, vol. 3. 2023. doi: 10.1016/j.atech.2022.100083.

[29] J. Chen, A. Zeb, Y. A. Nanehkaran and D. Zhang, "Stacking ensemble model of deep learning for plant disease recognition," *J Ambient Intell Humaniz Comput*, vol. 14, no. 9, 2023, doi: 10.1007/s12652-022-04334-6.

[30] J. Chen, J. Chen, D. Zhang, Y. Sun and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Comput Electron Agric*, vol. 173, Jun. 2020, doi: 10.1016/j.compag.2020.105393.

[31] M. Nawaz, T. Nazir, A. Javed, S. Tawfik Amin, F. Jeribi and A. Tahir, "CoffeeNet: A deep learning approach for coffee plant leaves diseases recognition," *Expert Syst Appl*, vol. 237, 2024, doi: 10.1016/j.eswa.2023.121481.

[32] E. C. Too, L. Yujian, S. Njuki and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput Electron Agric*, vol. 161, pp. 272–279, Jun. 2019, doi: 10.1016/j.compag.2018.03.032.

[33] G. Geetharamani and A. P. J., "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers and Electrical Engineering*, vol. 76, pp. 323–338, Jun. 2019, doi: 10.1016/j.compeleceng.2019.04.011.

[34] B. Chang, Y. Wang, X. Zhao, G. Li and P. Yuan, "A general-purpose edge-feature guidance module to enhance vision transformers for plant disease identification[Formula presented]," *Expert Syst Appl*, vol. 237, 2024, doi: 10.1016/j.eswa.2023.121638.

[35] A. L. Latifah, Lembaga Ilmu Pengetahuan Indonesia. Research Center for Informatics, Institute of Electrical and Electronics Engineers. Indonesia Section and Institute of Electrical and Electronics Engineers, *2018 International Conference on Computer, Control, Informatics and its Applications : "Recent Challenges in Machine Learning for Computing Applications" : proceedings : November 1st-2nd, 2018, Tangerang, Indonesia*.

[36] S. M. Hassan, M. Jasinski, Z. Leonowicz, E. Jasinska and A. K. Maji, "Plant disease identification using shallow convolutional neural network," *Agronomy*, vol. 11, no. 12, Dec. 2021, doi: 10.3390/agronomy11122388.

[37] Ü. Atila, M. Uçar, K. Akyol and E. Uçar, "Plant leaf disease classification using EfficientNet deep learning model," *Ecol Inform*, vol. 61, Mar. 2021, doi: 10.1016/j.ecoinf.2020.101182.

[38] A. M. Fayyaz *et al.*, "Leaf blights detection and classification in large scale applications," *Intelligent Automation and Soft Computing*, vol. 31, no. 1, pp. 507–522, 2022, doi: 10.32604/IASC.2022.016392.

[39] A. Elaraby, W. Hamdy and M. Alruwaili, "Optimization of deep learning model for plant disease detection using particle swarm optimizer," *Computers, Materials and Continua*, vol. 71, no. 2, pp. 4019–4031, 2022, doi: 10.32604/cmc.2022.022161.

[40] A. N. Özalp and Z. Albayrak, "Detecting cyber attacks with high-frequency features using machine learning algorithms," 2023.
[41] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2018, pp. 270–279. doi: 10.1007/978-3-030-01424-7_27.
[42] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition." [Online]. Available: http://image-net.org/challenges/LSVRC/2015/
[43] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: http://www.robots.ox.ac.uk/
[45] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks." [Online]. Available: https://github.com/liuzhuang13/DenseNet.
[46] A. Sardar, A. Issa and Z. Albayrak, "DDoS Attack Intrusion Detection System Based on Hybridization of CNN and LSTM," 2023.
[47] Q. Ji, J. Huang, W. He and Y. Sun, "Optimized deep convolutional neural networks for identification of macular diseases from optical coherence tomography images," *Algorithms*, vol. 12, no. 3, 2019, doi: 10.3390/a12030051.
[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.00567
[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.00567
[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.04381
[51] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," May 2019, [Online]. Available: http://arxiv.org/abs/1905.11946

Conflict of Interest Notice
The authors declare that there is no conflict of interest regarding the publication of this paper**.**

Ethical Approval and Informed Consent
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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 77-91
DOI: 10.35377/saucis...1381522
SAUCIS
SAKARYA UNIVERSITY
RESEARCH ARTICLE

Systematic analysis of speech transcription modeling for reliable assessment of depression severity

Ergün Batuhan Kaynak [1], Hamdi Dibeklioğlu [2]

[1] Department of Computer Engineering, Bilkent University, Ankara, Türkiye
[2] Department of Computer Engineering, Bilkent University, Ankara, Türkiye

Corresponding author:
Ergün Batuhan Kaynak,
Department of Computer Engineering,
Bilkent University, Ankara, Turkey
ebatuhankaynak@gmail.com

Article History:
Received: 26.10.2023
Accepted: 22.03.2024
Published Online: 22.03.2024

ABSTRACT
In evaluating the severity of depression, we rigorously investigate a segmented deep learning framework that employs speech transcriptions for predicting levels of depression. Within this framework, we examine the effectiveness of well-known deep learning models for generating useful features for gauging depression. We validate the chosen models using the openly accessible Extended Distress Analysis Interview Corpus (E-DAIC) as a dataset. Through our findings and analytical commentary, we demonstrate that valuable features for depression severity estimation can be achieved without leveraging the sequential relationships among textual descriptors. Specifically, temporal aggregation of latent representations surpasses the current best-performing methods that utilize recurrent models, exhibiting an 8.8% improvement in Concordance Correlation Coefficient (CCC).

Keywords: Depression severity assessment, Text analysis, Deep learning, Speech transcription

1. Introduction

Depression is a mental disorder that negatively affects the feelings, behaviors, and thoughts of individuals. Overwhelming feelings caused by depression can hinder the individual by leading to disinterest in daily activities and reduced concentration. It can even manifest itself as physical pain. Diagnosis of depression is very important as individuals, in the worst case, can be driven to suicide without proper treatment. Depression has many challenges, both regarding its diagnosis and treatment. Mental health issues are mistakenly not taken as seriously as physical health issues, and most people can show reluctance to accept they are suffering from an illness and seek professional help. This is exacerbated in the case of depression since depressed individuals generally do not have the motivation to perform simple daily tasks, let alone seek treatment. The difficulty of understanding the human psyche is also a primary concern. This can cause misdiagnosis of the severity of depression, as the symptoms can vary depending on individual differences of the patient.

To control this uncertainty, standardized tests are proposed. A popular test is the Hamilton Depression Rating Scale (HDRS) [1]. This test contains point scales in many depression cues, such as sleep quality, physical activity, guilt, and anxiety. The expert is expected to score the individual on these cues to understand their depression severity. As another means of assessment, individuals are also asked to self-assess using simple questionnaires, such as Physical Health Questionnaire Depression Scale (PHQ) [2]. In this study, we use a dataset with PHQ labels to evaluate our architectures.

The recent COVID-19 pandemic acted as a figurative breeding ground for depression. With many people stuck in their homes, deprived of their daily routines, anxiety and depression increased by 25% according to recent statistics by World Health Organization[1]. In light of this, many studies are conducted to further investigate the effects of the pandemic and depression [3]. This recent surge in depression, along with the discussed challenges, makes it clear that the need for good automated detection of depression severity systems is more important than ever. Advances in automatic assessment of depression severity would lead to helping over 300 million [4] people suffering from depression and even save their lives.

[1] WHO (2022). COVID-19 Pandemic [online] https://www.who.int/news/item/02-03-2022-covid-19-pandemic-triggers-25-increase-in-prevalence-of-anxiety-and-depression-worldwide. [accessed 09 09 2023]

Cite as: Kaynak E. B. and H. Dibeklioğlu, "Systematic analysis of speech transcription modeling for reliable assessment of depression severity", *Sakarya University Journal of Computer and Information Sciences*, vol. 7, no. 1, pp. 77-91, 2024. doi: 10.35377/saucis...1381522

To this aim, this study systematically analyzes different network architectures for depression severity assessment from speech transcriptions and discusses the implications of empirical results. Both temporal and non-temporal modeling approaches are investigated. Our experimental results show that the use of temporal pooling of latent representations, rather than recurrent modeling, provides state-of-the-art performance.

2. Related Work

Most depression severity assessment literature focuses on audiovisual modalities and their fusion. In contrast, this study focuses text modality as a single modal.

Studies show that many syntactic and statistical measures regarding language correlate with depression, such as the decrease in syntactic complexity [5] or the use of first-person pronouns [6]. Depressed individuals also display slower speech rates and longer pauses [7, 8]. However, compared to other modalities, fewer studies use text modality. When text modality is used, it is usually used as an additional modality rather than the main focus. Due to this lack of focus on text, most studies use rudimentary processing methods. Kaya et al. create 42 functionals using low-level descriptors such as word count and speech duration, along with a bag of words representation for each participant using term frequencies. Both these text-based features are then evaluated both by themselves and the use of weighted fusion networks [9]. Ye et al. choose to use the top 10 most frequent words to differentiate between healthy and depressed groups [10].

Deep learning based natural language processing embeddings are becoming more and more popular. Consequently, depression assessment networks also started utilizing these high performance semantic information descriptors. Studies [11, 12] use Word2vec [13] and its variants to extract representations. Recently, more powerful sentence embeddings are utilized with Universal Sentence Encoder [14-16] or BERT [17] models. These embeddings are usually used without finetuning the embedding network. An overwhelming majority, if not all, of recent deep learning networks process word and sentence embeddings using a recurrent architecture to explore temporal relationships [15, 16, 18, 19]. Differently, Yang et al. [12] format the text as a two-dimensional matrix of words and embeddings and process it using a TextCNN [20] variant with k-Max-pooling. This study also investigates the performance of temporal architectures on modeling text representations. Contrary to the literature, we also propose non-temporal modeling of sentences.

Even though it is not used as much as audiovisual modalities, text modality has several advantages. Channels like social media and messaging apps contain an abundance of text data. Although audiovisual modalities are also present in such channels, they are mostly used for other purposes and do not shed light on an individual's psyche as much as text modality does. Several studies document the potential of the social media domain [21]. Singh et al. show that for attention enabled ensemble learning models are effective methods of detecting depression symptoms on social media data, such as ones from Reddit and Twitter [22]. Some studies also combine text with social media metadata, such as posting habits [23, 24]. Even in such recent studies, non-deep learning based models, such as decision trees [23] are utilized. We also see this in the case of Liu et al., in which case the authors use an ensemble of support vector machine, naive bayes and regression approaches to detect depression cues [24]. Another property of text modality is that it is significantly harder to identify a person using non-handwritten text only. In contrast, a short audio recording or a single image can be enough to identify an individual. Such arguments create great motivation for the use of text modality for depression analysis.

3. Methodology

In this study, we propose a modular natural language processing pipeline to predict a PHQ-8 score, given sequential sentences of an individual. Figure 1 illustrates the schema for the proposed modular pipeline. Remaining subsections within this section detail the modules that will be used during experiments along with where they fit in our pipeline. Each module touches on a different consideration that emerges while building a regression network. Methods within each module have very similar, if not the same, input and output dimensions and considerations. This results in a highly modular experiment setup where each method of a module can be seamlessly interchanged with one another. After all modules are introduced, we present our investigated architectures in Section 3.2. These architectures are formed according to ablation studies performed in Section 4.

Main processing flow of our pipeline is as follows: First, each sentence for a participant is converted from text to a numeric vector representation. It is required for any form of natural language processing algorithm to apply this step first (unless we are considering a rule-based approach based on the actual string content). Section 3.1.1 describes the method we follow to achieve this conversion. These sentence embeddings can go through an individual processing step using residual blocks (Section 3.1.2) and/or the attention module (Section 3.1.3). These modules allow us to learn an intermediate representation before modeling the sequences as a whole. Our next task is to reduce these variable number of intermediate representations into a single representation, i.e. a summary of the participant. To this aim, Section 3.1.4 discusses both temporal and non-temporal summarization methods. Temporal methods use the order information of each sentence (i.e. each sentence is processed within the context of its preceding sentences), while non-temporal methods are not concerned with when the

sentence is uttered. Ultimately, the summary representation for each participant is regressed into a single value, which is our prediction, using linear regression layers.

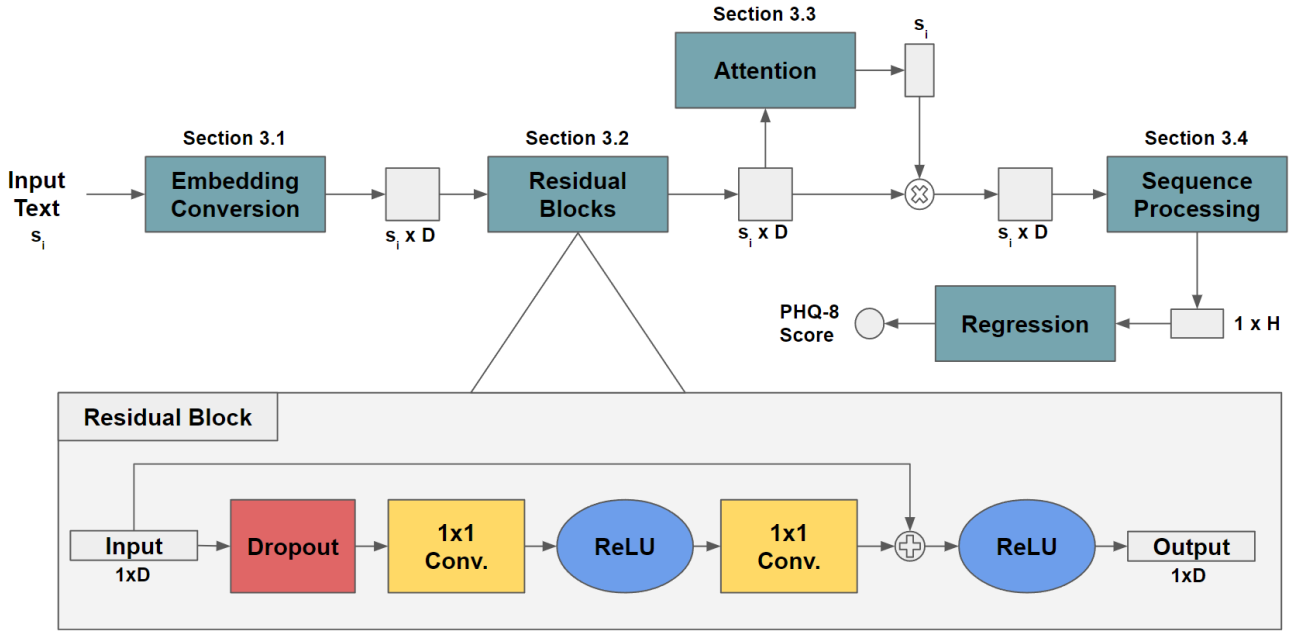


Figure 1: Overview of the proposed pipeline and the residual blocks. Data representations at several stages are depicted with their shapes. Shapes are given for a single participant and not batched input. For participant p_i , s_i is the number of sentences, D is the size of the sentence embedding (depends on embedding choice). H is the output dimension of the sequence processing module ($H = D$ for non-temporal modules, but it is a hyperparameter for temporal ones). Dropout is used at the beginning of each block for regularization. Two 1x1 convolutions with ReLU activation are used to process sentence embeddings and the skip connection happens after the second convolution, before activation.

3.1 Architectural Modules

3.1.1 Transcript Representation

To process text data, we first convert them into numerical representations. Historically, handcrafted algorithms are used to map words or sentences into numeric vectors. More recently, deep learning based architectures are used. This is a required step for our architectures.

For participant p_i, we obtain a sequence of sentence embeddings $P^{(i)} \epsilon\, s_i \times D$, where s_i is the number of sentences and D is the embedding size. We define the single sentence embedding uttered within the time period t as x_t. Embedding size D depends on the choice of embedding, but it does not create any considerations while building our architectures.

We start by introducing our main embedding, all-mpnet-base-v2 [25]. This embedding is a finetuned version of Microsoft's mpnet-base model [26]. Finetuning was applied with a contrastive loss objective using over 1 billion training pairs. Among other embeddings from the same framework, all-mpnet-base-v2 has the best average performance on 14 diverse sentence embedding performance tasks and 6 various semantic search tasks. Due to its performance and popularity, we believe this embedding is a good starting point for our ablation studies.

It should be noted that the embedding architecture is not appended to our end-to-end depression severity assessment network. To elaborate, output sentence embeddings are frozen, and the error from our prediction is not propagated back to these networks. Embeddings for every sentence are the same throughout training and validation procedures. This method was chosen to reduce computation and memory costs. Due to this, it is worth bearing in mind that our performance is reliant on the performance of our chosen embedding.

3.1.2 Residual Blocks

Residual blocks are the next module after converting sentences to sentence embeddings. Here, each sentence embedding x_t is processed through a variable amount of connected residual blocks. These residual blocks use the residual learning idea

from [27], where we add the block input to the output of that block. This skip-connection from the input to the end of the last convolutional layer, as seen in Figure 1 helps the network by both reducing vanishing gradients and reducing the possibility that new blocks degrade previously learned information. Since this module is optional, output representations are also called x_t for ease of notation.

3.1.3 Attention

In this section, we present our attention mechanism that can be used to introduce an additional scaling in between modules for intermediate vectors. Attention weight $a_t \epsilon R$ is calculated by regressing a scalar value from each intermediate representation x_t. This regressor is identical to the one we utilize to regress an output after sequence processing. x_t is then scaled with their respective attention score a_t before being pooled into a summary representation. We include a dropout layer before linear layers for regularization. Each linear layer reduces the input dimension by a multiple of 4 (i.e. $R^d \rightarrow R^{\frac{d}{4}}$using $a_t^i = ReLU(W_i a_t^{i-1}) + b_i$. Herein, $W_i \epsilon R^{s_i \times d}$ is the transformation matrix between layers and b_i is the bias term for the i-th layer. First a_t is the intermediate representation x_t. The last layer outputs a single scalar no matter the input dimension. Output attention weights can be normalized to the 0-1 range using the sigmoid function or min-max normalization $a_t = \frac{a_t - min(a_1, a_2, \ldots, a_t)}{max(a_1, a_2, \ldots, a_t) - min(a_1, a_2, \ldots, a_t)}$. We use such normalization functions to better regularize our network weights, and also provide contextual information across representations for a given participant.

3.1.4 Sequence Processing

Regressing a single scalar value requires some form of summarization of the many sentences a participant utters during their interviews. To this aim, we investigate recurrent and non-recurrent architectures. Transformer architecture was also considered, but our preliminary experiments showed that multi-head attention with positional encoding saturated our embeddings, and we could not reach decent performance. Therefore, it was not included in our in-depth analysis. In this section, we will provide the details of the various summarization methods we explored.

Temporal Modeling of Sequences To temporally model sentence embeddings, we use bidirectional GRUs. GRU architecture is selected due to its documented performance on temporal problems over RNNs and LSTMs, which we also replicated in our cross-validated preliminary experiments, and due to their ability to process variable length sequences. GRUs are also, on average, faster than RNN and LSTMs and use less parameters. Bidirectionality incorporates information from both directions (forward and backward) of the sequence. We define the forward direction of our GRU using the following equations:

$$\begin{aligned} z_t &= sigmoid(W_z x_t + U_z \vec{h}_{t-1}) \\ r_t &= sigmoid(W_r x_t + U_r \vec{h}_{t-1}) \\ c_t &= tanh(W_c x_t + U_c(r_t \odot \vec{h}_{t-1})) \\ \vec{h}_t &= (1 - z_t)\vec{h}_{t-1} + z_t c_t \end{aligned} \tag{1}$$

Where r_t and z_t are the reset and update gates, respectively. The activation of the hidden state $\vec{h}_t$ at time t is the linear interpolation between previous activation $\vec{h}_{t-1}$ and the candidate activation c_t. Weight matrices W and U with subscripts z, r, c are the parameters of the GRU. Each subscript defines another translation from the input sentence embedding x_t. $\odot$ is the operation for element wise multiplication. We concatenate the hidden states of the forward and the backward passes for each t to obtain h_t. The resulting s_i many timesteps are reduced to a single vector using either last-pooling, mean-pooling, or max-pooling. Last-pooling simply assigns the output to the last hidden state; mean-pooling takes the average of all hidden states over the t dimension, while max-pooling takes the maximum over the t dimension. Formally, the output $h^{(i)} \epsilon R^H$ for $P^{(i)}$ is obtained by:

$$\begin{aligned} \text{LAST\_POOL} &\rightarrow h^{(i)} = h_{s_i - 1} \\ \text{MEAN\_POOL} &\rightarrow h^{(i)} = \frac{1}{s_i} \sum_{t=1}^{s_i} h_t \\ \text{MAX\_POOL} &\rightarrow h^{(i)} = max[h_1; h_2; \ldots; h_t] \end{aligned} \tag{2}$$

Non-Temporal Modeling of Sequences Temporal methods are not the only way to process ordered sequences. Temporal architectures have the inherent assumption that there is generalizable information to be found in the order in which we find our sentence embeddings and process each sentence embedding within the context of previous ones. While such methods

are widely used in depression severity assessment literature, the idea that sentences can be good indicators by themselves has not been explored much. Theoretically, if sentences themselves are sufficient, additional context information could even be causing overfitting or significant prediction error for participants with underrepresented PHQ-8 scores.

The proposed non-temporal sequence processing module aims to reduce a sequence $P^{(i)}$ with a variable number of sentence embeddings into a single vector. To this aim, we employ several pooling methods. Similar to Section 3.1.4, mean-pooling takes the average of sentence representations, while max-pooling filters the maximum activations along the t dimension. Formally, the output $h^{(i)} \epsilon R^D$ for $P^{(i)}$ is obtained by:

$$\text{MEAN\_POOL} \rightarrow h^{(i)} = \frac{1}{s_i} \sum_{t=1}^{s_i} x_t$$

$$\text{MAX\_POOL} \rightarrow h^{(i)} = max[x_1; x_2; \ldots; x_t] \quad (3)$$

3.2 Investigated Architectures

Through our experiments and ablation studies, we combine and examine the sub-modules detailed in this chapter. Some of these complex models are proposed as good candidate architectures for the depression severity assessment task, while others will be disqualified through our validation process and discussions on behavioral aspects and generalization to real-world scenarios. This section reveals the details of such candidate architectures:

- NT-MEAN: **N**on-**T**emporal model using **MEAN**-pooling
- NT-MEAN-ATT: **N**on-**T**emporal model using **MEAN**-pooling with **Att**ention
- T1-MEAN: **T**emporal model using **one** GRU with **MEAN**-pooling
- T2-MEAN.MAX: **T**emporal model using **two** GRU's with **MEAN**-pooling on first level and **MAX**-pooling on second level

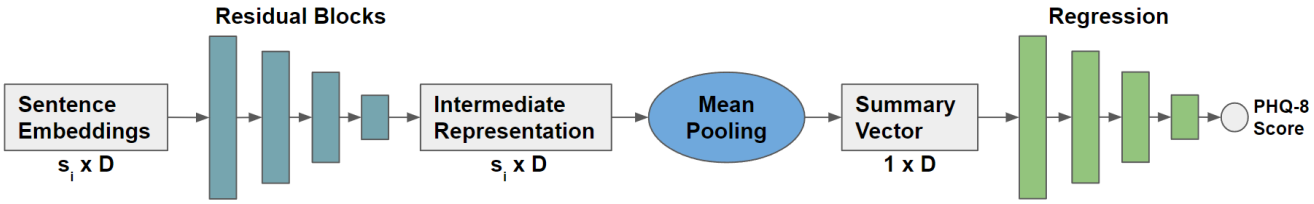


Figure 2: Overview of the NT-MEAN architecture. Shapes are given for a single participant and not batched input. For participant p_i , s_i is the number of sentences, D is the size of the sentence embedding (depends on embedding choice).

3.2.1 NT-MEAN and NT-MEAN-ATT

NT-MEAN is our simplest model. Sentence embeddings are passed through residual blocks. These output representations $x_t \epsilon R^{s_i \times D}$ are then averaged for each participant to obtain a summary representation for that participant. Resulting summary representation of shape $1 \times D$ is regressed with linear layers to obtain the final score. NT-MEAN-ATT is a non-temporal model similar to NT-MEAN. Their difference lies in the addition of a feed forward attention module. This attention module calculates attention scores a_t for each x_t. x_t is then scaled with their respective attention score before being pooled into a summary representation. Figure 2 shows the detailed NT-MEAN architecture.

3.2.2 T1-MEAN and T2-MEAN.MAX

T1-MEAN and T2-MEAN.MAX are both temporal models, meaning that they both use a recurrent architecture, namely a GRU, to process intermediate representations x_t created by the residual blocks. T1-MEAN uses a single level GRU followed by mean-pooling to create a summary representation of shape $1 \times H$. On the other hand, T2-MEAN.MAX utilizes a two-level GRU with mean-pooling in the first level and max-pooling in the second level to create the same representation. As with our non-temporal models, these output summaries are regressed to PHQ-8 scores.

4. Experimental Setup

4.1 Dataset

4.1.1 Overview

We use the Extended Distress Analysis Interview Corpus dataset (E-DAIC) dataset, which is the dataset for the Detecting Depression with AI Sub-challenge (DDS) during The Audio/Visual Emotion Challenge (AVEC) 2019 Workshop and Challenge [28]. This is an actively used benchmark dataset for this task. The dataset contains 275 interviews with unique

participants, and it is collected in an effort to create an AI agent that can interview and identify mental health problems. The interviews consist of a participant's dialogue with an interviewer. The interviewer is either a human or a fully automated AI. Regardless of the nature of the interviewer, the participant sees an animated virtual avatar on the screen in front of them. Among the 275, 163 subjects are used for training purposes, while validation and test splits contain 56 each. The splits are balanced in terms of age, gender distribution, and PHQ-8 scores. We use the predetermined splits during our experiments.

The dataset contains four video and six audio features along with raw audio and speech transcripts. The text is transcribed using Google Cloud's speech recognition service. Due to the private nature of the data, raw video footage is not available. This study uses only transcribed text portion of the data. Label for each participant is a self-reported 8-item Patient Health Questionnaire (PHQ-8) score [2]. PHQ-8 is a self-assessed depression severity measure. The questionnaire provides insight into the degree of impairment an individual goes through on eight different depression cues. A higher score means it is more likely that the individual is suffering from depression.

4.1.2 Challenges

Label imbalance While the splits are balanced in terms of the PHQ-8 score, there is a high imbalance of scores within each split, i.e. People considered non-depressive (PHQ-8 < 10) make up 69%, 73% and 63% (ordered training, validation and test) of all data. This imbalance is increased when we compare participants with severe depression (PHQ $\geq$ 20) to the remainder of the data (non-depressive PHQ-8 < 10 and depressive 10 $\leq$ PHQ-8 < 20). In that case, participants with severe depression only make up 4%, 2% and 7% (ordered training, validation and test) of all data.

Transcription Noise It should be noted that the transcriptions of sessions are not perfect. There are many sentences that do not exactly match the raw audio, and sometimes the voice of the therapy AI or a technician is also transcribed as sentences from the participant. There are also cases where sentence breaks are not recognized, and several sentences are transcribed as a single long sentence. Since it is not feasible to dynamically correct these mistakes, we left the faulty data in its original state. The transcribed text also contains a confidence level (a real number between 0 and 1) for each transcribed sentence. We empirically see that inclusion of this value in our training is generally detrimental to performance. Manual inspection of the dataset shows us that the confidence level is not very reliable, as it often gives low confidence to correctly transcribed words while giving high confidence for bad transcriptions. In light of these inspections, we opt not to use this information.

4.2 Evaluation Criteria

Dataset used in this study was first introduced in The Audio/Visual Emotion Challenge and Workshop 2019 (AVEC 2019) [28]. Organizers of the challenge picked Lin's Concordance Correlation Coefficient (CCC) [29] as the evaluation metric. CCC is a statistical measure of how well a set of predictions compares to the ground truth labels. Since it is a correlation measure, the value of CCC ranges from -1 to 1, where 1 signifies complete correlation between two sets. Since we are dealing with a sample of the total population, we use an approximation of CCC:

$$\widehat{CCC} = \frac{2S_{YX}}{S_X^2 \quad + S_Y^2 + (Y - X)^2} \tag{4}$$

Organizers choose this metric due to its invariance to scale, as well as its ability to include information on accuracy and precision [28]. We also use this metric in our training and evaluation. Organizers also propose Root Mean Square Error (RMSE) as a secondary metric. RMSE computes the numeric difference between prediction and target without any complex statistics. Taking the square of the error makes it so that higher errors are punished more. When used as a loss function, this property of RMSE can help reduce overfitting that can occur in our dataset due to label imbalance.

Alternative to these two metrics, we also propose reporting Mean Absolute Error (MAE). While CCC and RMSE have great properties during training, they cannot be easily interpreted. Even though MAE is ubiquitous within the literature for regression tasks, we see that it is scarcely reported for this dataset. We believe this metric is important to better understand and discuss our results and should also be reported for this dataset.

We follow the traditional training-validation-test scheme using the predetermined splits of the AVEC competition [28]. To reduce selection bias, and mimic the conditions of AVEC competition, we do not evaluate our models on the test set until we finalize model selection through ablation studies on the validation set. We evaluate four models in the test set during this study, to discuss and compare generalization performances. Implementations are done using PyTorch and models are optimized with optuna library using a Tree-structured Parzen Estimator (TPE) for hyperparameter selection.

We use Adam optimizer with 10^{-4} learning rate. Training data is shuffled each iteration, but no augmentation is applied. The training is terminated if our validation loss doesn't improve for 25 epochs. When the training is terminated, the checkpoint with the lowest validation loss is taken as the trained model. We empirically see that Batch Normalization [30]

is generally detrimental for our training, and do not include it in our models. We use ReLU as our activation function of choice due to its popularity and performance over other activation functions. After our optimization and validation procedures, we opt for using four regression blocks, four linear layers for regression and 0.5 dropout probability in our proposed NT-MEAN model. All experiments are conducted on an Nvidia RTX 2080 Ti GPU.

5. Experimental Results

5.1 Temporal Modeling

Temporal models include the order of sentence representations within the sequence as additional information. We now analyze the effects of different pooling methods on these sequences, and the implications of processing overlapping sequence chunks in a two-level GRU setup. After this discussion, we pick the best-performing single-level and two-level models and evaluate them in the test set in Section 5.1.2.

Table 1: Comparison of pooling methods for temporal architectures. Having no pooling on the second level means that the model utilizes a single-level GRU.

First Level Pooling	Last	Last	Last	Last	Max	Max	Max	Max	Mean	Mean	Mean	Mean
Second Level Pooling	-	Last	Max	Mean	-	Last	Max	Mean	-	Last	Max	Mean
Validation CCC	0.589	0.632	0.649	0.634	0.646	0.655	0.649	0.637	0.650	0.658	0.659	0.624

5.1 Assessment of the Number of Recurrent Layers and Pooling Methods

Results in Table 1 show that some configurations of the two-level model perform better than the single-level GRU, while others are still behind single-level with mean or max pooling. Compared to using last-pooling for a single-level GRU, using last-pooling at the first level of the hierarchy does not have any obvious detrimental effects on performance. This is possibly due to the increased information stored within the last hidden state of each chunk in the first level. Also, two of the top three results in this analysis use last-pooling in the second level, providing evidence that last-pooling thrives with shorter sequences. These findings about last-pooling show us that temporal information about depression is not retained for a long time. The performance of mean-pooling in the first level is also noteworthy. We see that for configurations where the second level uses either last or max-pooling, having mean-pooling in the first level is always better. This does not hold when mean-pooling is used for the second level. This could mean that the local scope is better used to understand the overall depressiveness of small conversation episodes, and the global scope does better at forming a final representation using these summaries.

Table 2: Results for different recurrent structures. Results are given for three pooling methods in the validation set.

	Pooling Method		
Recurrent Structure	Last	Max	Mean
GRU	**0.589**	**0.646**	**0.650**
LSTM	0.557	0.625	0.643
RNN	0.363	0.614	0.603

In the single-level setup, we compare three different temporal models, each of which uses a different pooling method to obtain a single summary vector. With the last-pooling method, the performance of the sequence reduction depends on the assumption that h_t holds the information for the entire sequence. This assumption may not hold well based on the length of the sequence and the decisions of gates within the architecture. Also, our knowledge of the nature of interviews and manual inspection of the data shows us that the last couple of sentences are reserved for farewells (e.g. "goodbye", "bye-bye", "okay bye") or small talk about the interview (e.g. "a real life person is really looking at me", "I was expecting", "that was cool"). Also, as we discussed in Section 4.1.2, an operator's voice can be mistaken as the interviewee's and transcribed into text. This usually happens at the start or the end of the interview. Since hidden states hold more information on recent timesteps, these noisy data points can pollute the hidden state and, therefore, reduce the information contained within our summary vector.

Temporal model performance is significantly improved using max or mean pooling instead of last-pooling. Both mean and max pooling has been used extensively in the literature. Theoretically, max-pooling works best when the existence of certain peak values is very important for inference, and completely saturating other activations is not a problem. Conversely, mean-pooling is a better choice when losing minima and maxima is not important, but keeping the overall activation is. The slightly superior performance of average pooling over maximum pooling in the proposed case is because for a participant to be high on the PHQ-8 scale, cues need to be salient throughout the entire interview, not just in parts of the interview. To make sure we are covering a wider range of temporal models when we talk about them, we also include experiments conducted on RNN and LSTM architectures (Table 2). GRU outperforms the other architectures in every pooling configuration. For brevity, we do not go into details for them.

Table 3: Results for best performing temporal (T1-MEAN and T2-MEAN.MAX) and non-temporal (NT-MEAN and NT-MEAN-ATT) models. Results are given for three metrics in both validation and test sets.

Model	Validation			Test		
	CCC	MAE	RMSE	CCC	MAE	RMSE
T1-MEAN	0.650	3.393	4.66	0.598	5.232	6.656
T2-MEAN.MAX	0.659	3.321	4.33	0.572	4.464	5.616
NT-MEAN	0.673	3.214	4.217	0.729	3.304	4.353
NT-MEAN-ATT	0.654	3.269	4.412	0.708	3.801	4.925

5.1.2 Best-Performing Temporal Models

When we compare our single-level and two-level GRU experiments, we see that the models are relatively close in performance. The best-performing single-level model beats 6 of the 9 hierarchical configurations. To better understand the effects of using a two-level approach, we examine the best-performing model from both single-level and hierarchical experiments in Table 3.

In this section, we examined the temporal dynamics regarding depression cues. While learning such relationships result in good models, it is unclear how much of our performance can be ascribed to temporal dynamics. This makes us question the reliability and benefit of temporal architectures. Indeed, if we think about our data, if we know that the participant said: "I haven't been happy at my jobs for at least 10 years", do we need to relate that information to the sentence "New York"? (sentences taken from a participant within our dataset). There is no denying the importance of contextual information, especially for audiovisual modalities [31]. However, we believe they are not as strong for text modality. Given more data, it may be possible to form contextual relations. But for our case, forcing the model to create such relations might result in noise most of the time.

5.2 Non-Temporal Modeling

Following our findings regarding temporal dynamics, we experiment with the simpler non-temporal approach. In this approach, each sentence embedding is passed through several residual blocks before they are pooled into a single vector. Similar to the temporal models, we start by experimenting with different pooling methods. The next subsection uses different attention methods and comments on their differences. We again finalize our discussions by evaluating the best-performing models from each subsection.

5.2.1 Comparison of Pooling Methods

We compare two different non-temporal models, each of which uses a different pooling method to obtain a single summary vector. In Section 5.1, we hypothesize that temporal information could hinder performance. To this aim, we discard recurrent modules from our architecture and replace them with simple pooling operations, and achieve 0.673 for mean-pooling and 0.629 for max-pooling in terms of validation CCC. Observing the performance of mean-pooling, it appears that the exclusion of temporal information leads to a performance increase. As with the temporal pooling experiments in Section 5.1.1, mean-pooling is superior, this time with a bigger margin compared to max-pooling. Observations we can make here regarding the comparison of max and mean pooling are similar to the ones made in Section 5.1.1. It seems that individual high activations are less impactful while forming a summary vector compared to computing the overall activations.

5.2.2 Effect of Weighting Embeddings

As per our discussions, mean and max pooling both make different assumptions on the relative weights of sentence representations. The reason for the performance of mean-pooling is unclear. Since we know that not every sentence is a cue for depression, we would expect such sentences to provide noise to the averaging process. For this reason, incorporating other modules to have a better representation selection process could result in a better average summary. To this aim, we experiment with Softmax Weighted Mean-Pooling and Attention Weighted Mean-Pooling (named SWM and AWM, respectively).

SWM simply takes the softmax of intermediate representations. Softmax values are then multiplied with their corresponding representations to scale them before mean-pooling is applied. This incorporates feature importance to each representation and by proxy to the summary vector. AWM technique calculates an attention score a_t for each representation. a_t for each sentence representation can be any real number. Since this can cause scaling instabilities, we also experiment with applying two normalization techniques before we multiply it with its respective sentence embedding: min-max normalizing a_t to the range 0-1 and passing a_t through a sigmoid function. As with SWM, each a_t is multiplied with its corresponding representation before being pooled to create a summary representation.

Table 4: Comparison of different representation weighing methods for non-temporal architectures.

Pooling Method	Val CCC
SWM	0.581
AWM /wo Norm	0.642
AWM /w Min-Max Norm	0.654
AWM /w Sigmoid Norm	0.645

Our results in Table 4 show that AWM with min-max normalization does better than the alternatives, and AWM in general performs better than SWM. Since AWM contains an additional network to compute individual weights for each representation, each weight is calculated independently of the other embeddings within the sequence. Weights from Softmax, on the other hand, depend highly on the length of the sequence. To elaborate, the same representation can have a very different softmax weight for different participants since softmax distributes a probability of 1 among all representations of that participant. While this can create better relative weights within the sequence, it is a source of high variance for the model in general.

Sigmoid function introduces additional non-linearity to our network, and it could give saturated weights for some embeddings due to its shape. Although it has such qualities, it doesn't have a way of incorporating information from other representations within the sequence. One could argue that min-max normalization is better in that regard, as it does a better job of distributing weights linearly among other representations. Even though AWM with min-max normalization is the best among weighted models, it is still behind simple mean-pooling (Section 5.2.1). Scaling each representation with learned weights seems to result in less representative embeddings. The reason could be similar to the arguments we made for temporal information in Section 5.1. There, we argued that while temporality could be informative by incorporating contextual information, it could cause more noise than information. While we use a relatively simple way to add context information in this section, it still causes noise to our model.

Since we argued that some sort of selection should happen for a good model, we analyze our non-temporal mean-pooling model (NT-MEAN) to better understand its inner workings. To this aim, we come up with a way to relatively weigh the intermediate representations used by a trained NT-MEAN model. We opt for using a measure of magnitude. Namely, we take the average of each representation over the embedding dimension to obtain s_i magnitude averages. These averages are then min-max normalized to the 0-1 range. These weights are called feature importance for a given representation. These weights are not directly a measure of depression per se, but rather signify how much a sentence is deemed important for giving a PHQ-8 prediction. Whether the model prediction is high or not depends on the interaction of residual block outputs with the linear regression head and is not easily interpretable.

Table 5: Feature importances assigned by NT-MEAN model, along with the corresponding raw sentence data.

Prediction	PHQ-8	Importance	Corresponding Text
13	7	1.0000	stressed out
		0.9214	yeah I would say for the past several months
		0.8391	I can't function as well
		0.0002	take my dog for a walk
		0.0002	while I was in a car accident where a drunk driver hit me and I had to
		0.0000	I like the weather I like the beach
21	15	1.0000	I don't know I I I developed anxiety and I freaked out you know if I think I'm going to run out of gas I get short of breath and
		0.9127	sometimes I just give up and I don't even try anymore
		0.8360	hello I've lost all the ability to trust and I'm numb to all feelings partly
		0.0012	I I guess I could erase my big pts State when I was 18 a serial killer
		0.0012	I love the weather people are generally more friendly than where I've lived on the East Coast the scenery the environment the beach the mountains the
		0.0000	that's not my PTSD thing though if you're wondering
0	2	1.0000	I've been feeling fine
		0.5020	I'm pretty easy over the last two to three weeks I think there was one night or I had so much on my mind I just find it hard to fall asleep but in general I do sleep well
		0.4371	ragging
		0.0017	I wish that I would argue with my husband less especially in front of our kids
		0.0011	one of my most memorable experiences in terms of travel I guess was the time that my luggage got lost in front of Vallarta and I spent the week I'm wearing my husband shorts and t-shirt
		0.0000	I've been feeling fine this summer the work stress is still there but my kids are out of school so our household is a lot more relaxed

Table 5 shows the PHQ-8 prediction from NT-MEAN model side-by-side with the PHQ-8 label for the participant. Then, it lists the highest and lowest three feature importance weights for each participant. These weights are presented alongside their corresponding original text. All participants are from the test set. We see that sentences with high feature importance weights are indeed good indicators for either depressive or healthy behaivour. Sentences with lower feature importance seem to be either neutral or longer and more convoluted sentences. The weights are not perfect; as we can see, several sentences depicting unfortunate life events have low relative weights. Nonetheless, this shows that there is an inherent selection process. The context information is possibly applied by using mean-pooling: If the average of representations is more leaning toward a behaviour (i.e. either depressive or healthy), this means that the participant contains relatively more important representations that point to that behavior.

5.2.3 Best-Performing Non-Temporal Models

We conclude our non-temporal model analysis by proposing two networks. Mean-pooling network without attention achieves the best validation score among non-temporal methods. AWM /w MinMax Norm is the best-performing weighted model. We previously argued that an attention-based model could generally find good weights for embeddings, but mislead the model on edge cases. While we show that attention is not required for good representation selection, we believe it is possible for such a model to be less susceptible to overfitting and have good generalization. We also check the MAE and RMSE metrics for these two models in the validation set and observe that they are very similar. Due to these reasons, we believe the attention model should also be evaluated with the test partition and its results should be discussed.

Table 3 presents our non-temporal model results. Both models achieve good generalization across all three metrics. NT-MEAN outperforms NT-MEAN-ATT in each metric, especially so in terms of generalization to the test set, providing evidence that the attention module in NT-MEAN-ATT is detrimental to performance. With that being said, NT-MEAN-ATT has better generalization compared to our best-performing temporal models. This provides evidence that incorporating contextual information via recurrent architectures could prove challenging, and simpler methods can perform better. Four residual blocks with 0.4 dropout probability, followed by four linear layers with 0.5 dropout probability to obtain the best-performing NT-MEAN model.

5.3 Experiment on Word Count per Sentence

Our experiments thus far take the semantic meaning of sentences to predict depressive behaviour. In this section, we focus solely on statistics regarding sentences rather than their meaning. During our literature reviews, we find that text modality is not well analyzed in the literature. We aim to expand the literature by connecting statistical findings with learning-based results. Every experiment is performed by running inference on already trained models, unless otherwise stated.

During our analysis of feature importance in Section 5.2.2, we observe that the length of individual sentences differs for different participants. As we recall from Table 5, high feature importance can be assigned to sentences with only 2 words, as well as to sentences with around 30 words. Before we analyze the relationship of word count with depression classes, we look at its effect on general performance. Using the trained version of our model NT-MEAN, we re-evaluate the validation set. The model is not trained again or finetuned for each individual word count configuration, but only reevaluated. Manual observation of the dataset shows that in cases where there is not much time between sentences, speech-to-text AI transcribes some answers by the participant as long sentences. Separation of these sentences is non-trivial, and we believe that the benefits of keeping such sentences as they are outweigh potential problems that can occur if we are to separate them.

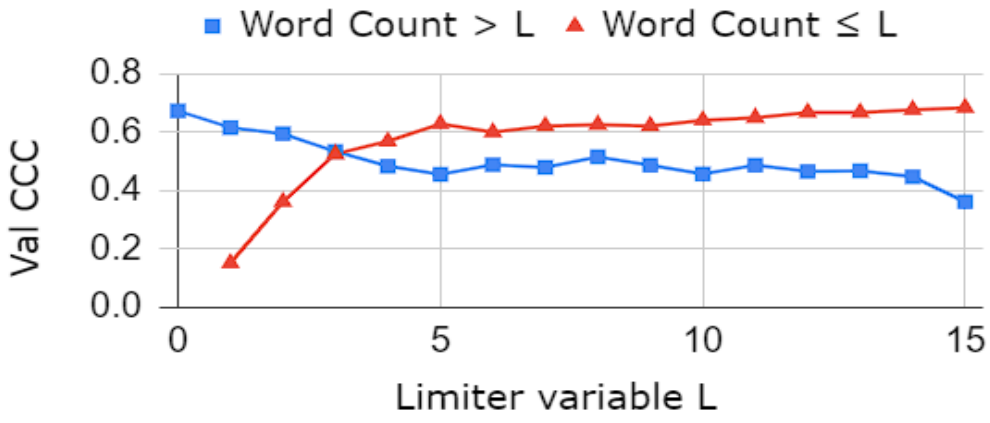


Figure 3: Additive (Red-Triangle) and Subtractive (Blue-Square) experiments on word count, using NT-MEAN model. For Limiter variable L, Additive experiment evaluates NT-MEAN on all sentences with word count ≤ L . Subtractive experiments do the same for word count > L.

We conduct two experiments: subtractive and additive. Subtractive word count experiment includes all sentences whose word counts are bigger than our word count limiter variable L. Formally, a sentence S is included in evaluation if the predicate $|S| > L$ is true, where $|S|$ is the number of words in S. Conversely, additive experiments include sentences that obey $|S| \leq L$. Figure 3 shows the results of these experiments. We only experiment up to L = 15 because subtractive experiments become noisy after that point, due to the lack of such long sentences for each participant.

Our performance drops continuously when we do not include sentences with word counts up to but excluding 5. After that point, the performance has fluctuating changes, but they are not as close to the change and consistency we see with the first 5. Similarly, performance continuously increases as we include sentences with word counts up to and including 5, but the rest is not consistent. This analysis shows that model performance highly depends on shorter sentences. Before we can clearly understand the effect of individual sentences, we examine the percentages of sentences with a certain number of words in our data.

Coincidentally, sentences with at most 6 words (L = 6) make up almost 50% of our data. Since this point separates our data equally, we compare our two experiments using this data point. Comparison of additive and subtractive experiments show that at point L = 6, additive experiment with CCC value of 0.6 performs significantly better than its subtractive counterpart with CCC value of 0.489. Since both these experiments contain very similar amounts of data, we argue that the difference likely depends on whether we include shorter sentences.

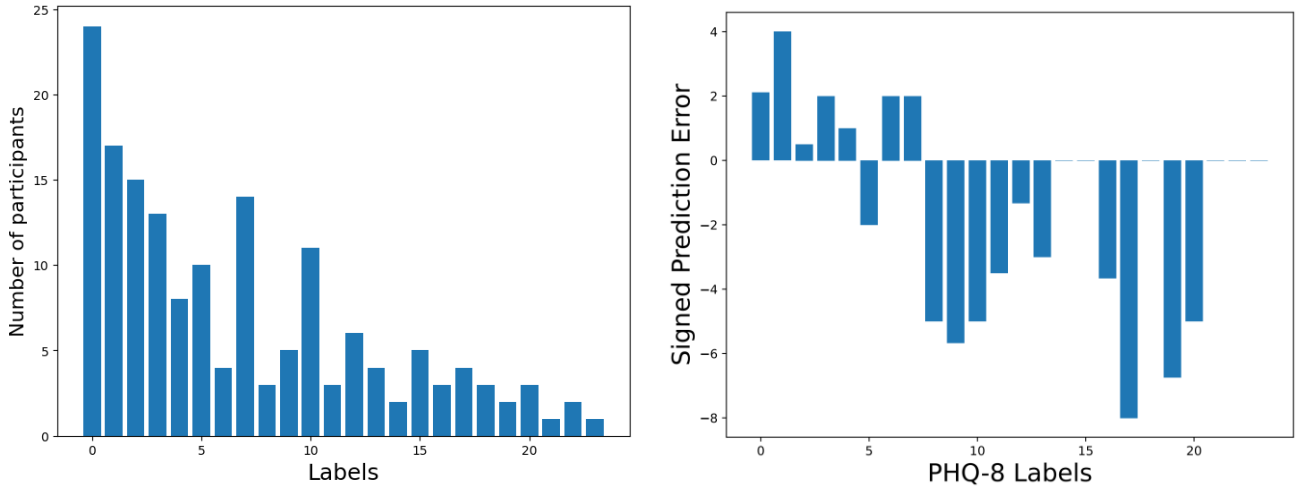
Our discussions up to this point originated from a trained model. We examine the applicability of our argument to learning by training two models. Each of these models perform their training and validation using half of the data, one uses sentences such that $|S| > 6$ while the other uses $|S| \leq 6$. We observe that these models achieve a CCC score of 0.568 and 0.575 respectively. This means that both models perform significantly worse than NT-MEAN (0.673), which was trained with all sentences (i.e. with $|S| > 0$). Also, both models perform similarly using their respective data. Although our data separation took into consideration the information gain from data on both sides, it seems that losing close to 50% of data for training is something we cannot ignore. Per our previous finding in this section, one would expect the $|S| \leq 6$ model to perform better. We argue that not including longer sentences could be detrimental to training, irrespective of the information loss argument. It is known that some amount of noise in training is good for regularization and reduces overfit in some networks [32, 33]. Some architectures purposefully focus their training on optimizing hard examples [34], most popularly in the case of most novel deep metric learning networks [35]. In our case, longer sentences could act as regularization by providing hard examples to the network. This way, the network does not overfit by using only specific information. In a sense, more informative sentences are easier to learn, probably because their semantic content is more obviously a member of one class. It is no coincidence that shorter sentences have easier to learn semantic content since longer sentences are more convoluted and may contain more than one emotion. This also points out a problem with our simple word count-based approach, as the effects of longer sentences with a single emotion are completely omitted. We can see evidence of this behaviour by inspecting important sentence examples in Table 5. We can see that sentences that were deemed important by the model focus on a certain topic or emotion, compared to unimportant sentences. All in all, it seems that certain sentences are not very informative during inference, and they can even hinder prediction at times, but we should use as much data as possible during training. Admittedly, this finding is not out of the ordinary for neural networks, but we state it regardless to explain our findings better.

We stress that these findings are dependent on the dynamics of this dataset, and future work should be conducted to better understand this behaviour. We are also making our arguments based solely on analytical findings, and the argument holds only for a specific part of the pipeline, the inference. In that sense, the reliance argument on shorter sentences is purely for computational purposes, and not necessarily an argument on the linguistic properties of the disease. With that being said, longer pauses between words and sentences that we observe with depressed patients [7, 8] could mean that the automatic transcription tool creates shorter, less convoluted sentences, and these sentences are more impactful during inference.

5.4 Experiment on the Effects of Imbalanced Data

The scarcity of data for highly depressed individuals in the dataset is clearly a challenge. This problem also applies to scores in the 10 $\leq$ PHQ-8 < 20 segment due to label imbalance, and makes overfitting towards the scores in the PHQ-8 < 10 segment a possibility. However, since our problem is modeled as a regression problem, we believe this issue is applicable for every PHQ-8 score. To demonstrate that the dataset is not suffering from high degrees of overfitting, we present 4. This figure shows (a) the PHQ-8 distributions in the training set and (b) the labeled prediction errors in the test set. With this figure, we hope to observe the generalization performance in the test set for PHQ-8 scores that have the potential to be memorized during training. We would like to draw your attention to a few points in these figures. First, although the PHQ-8 < 10 segment constitutes a very large portion of the data, it also varies within itself. For instance, there are many underrepresented PHQ-8 scores in this segment. Specifically, scores like PHQ-8=6 and PHQ-8=8 have much less data compared to PHQ-8=1. The error margins for these individuals vary. For example, PHQ-8=1, which has 24 representatives, and PHQ-8=6, which has 4 representatives, have the same average error margin. PHQ-8=12, which has 6 representatives, has achieved a better error margin than both of these groups. Another example is that PHQ-8=22, which has two representatives, was predicted with a 0 error margin in the test set (since the error margin is 0, it did not form a line

in the graph; there is one data point in the test set that was predicted with a 0 error for the PHQ-8=22 score). We do not believe we've encountered a situation that would suggest a likelihood of memorization between the number of representatives and error margins.



(a) (b)

Figure 4: Figures depicted to show the effect of (a) the number of PHQ-8 representatives during training on the (b) generalization error. Our argument is that the imbalance in the data distribution does not lead to significant memorization

Table 6: Details regarding the modalities and performance of other, most recent studies in the literature that use AVEC2019 dataset, to the best of our knowledge. Modalities are abbreviated as A = Audio, V = Vision, T = Text.

Model	Modalities	Val CCC	Val RMSE	Val MAE	Test CCC	Test RMSE	Test MAE
NT-MEAN (Proposed)	T	0.673	4.22	3.21	**0.729**	**4.35**	**3.30**
Fang et al. (2023) [36]	AVT	-	-	-	-	5.17	-
Wang et al. (2022) [37]	AVT	-	4.03	3.05	-	-	-
Han et al. (2022) [38]	A	-	5.56	4.65	-	6.29	5.38
Sun et al. (2022) [19]	AVT	-	-	-	0.583	-	4.37
Saggu et al. (2022) [16]	AVT	0.662	4.32	-	0.457	5.36	-
Sun et al. (2021) [39]	AV	0.733	3.78	-	-	-	-
Yin et al. (2019) [40]	AVT	0.402	4.94	-	0.442	5.50	-
Makiuchi et al. (2019) [18]	AT	0.696	3.86	-	0.403	6.11	-
Kaya et al. (2019) [9]	AT	0.481	-	-	0.344	-	-
Ray et al. (2019) [15]	AVT	-	4.37	-	0.670	4.73	4.02
Ringeval et al. (2019) [28]	AV	0.336	-	-	0.111	-	-

5.5 Comparison with Other Methods

We finalize our analysis by comparing the performance of our best-performing model, NT-MEAN, to other studies within the literature. Table 6 is a compilation of studies from the literature that use the AVEC 2019 dataset. Listed modalities describe all modalities that the corresponding study explores, and is not necessarily what their final model is based on). In the case where multiple models are proposed, the one with the higher test set performance is chosen. To increase the comparability of our model for future works, we provide both validation and test set evaluations for three metrics. To the best of our knowledge, we are the only study that does not utilize a recurrent architecture in their proposed model. Comparing our performance, we see that NT-MEAN improves the state of the art by Ray et al. [15] on all metrics. The relative improvements are 8.8% for CCC, 8.7% for RMSE, and 21.8% for MAE.

6. Conclusion

In this paper, we have proposed temporal and non-temporal architectures to predict PHQ-8 depression scores. Compared to the majority of studies in the literature, we have only used text modality as a single modal. Our non-temporal model NT-MEAN has improved the state of the art by 8.8%, using a simpler architecture. To shed more light on the inner workings of this non-temporal network, we have extracted sentences that are deemed important by the network by examining network activations. Through this, we have shown that our model successfully learns to select important representations. As we have compared temporal and non-temporal architectures, we have realized that temporal relationships of individual sentences are tenuous at best, and not using the temporal information is better for performance.

We have also expanded the literature on natural language processing and depression severity assessment by presenting our empirical findings regarding participant sentence statistics, such as word count of sentences. We have displayed that a well-trained model shows less reliance on longer sentences. To put it in another way, longer sentences are not as informative for depression assessment compared to shorter ones. We believe this is because shorter sentences usually focus on a very specific semantic information or emotion and are therefore better captured by the sentence embeddings. We should stress that this is an analytical finding regarding only our inference step and could depend highly on the dynamics of the dataset, and we are not making linguistic arguments about the disease. More work is needed for a more solid understanding of this occurrence.

Motivated by the properties of text modality, we hope that the discussions we have started and improvements we have proposed in this paper will open new directions for feature work on depression assessment. We have high hopes that through such conversations, we will understand this insidious illness better.

References

[1] M. Hamilton, “A rating scale for depression”. *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 23, no. 1, pp. 56–62, 1960.

[2] K. Kroenke, T. Strine, R. Spitzer, J. Williams, J. Berry and A. Mokdad, “The phq-8 as a measure of current depression in the general population”, *Journal of Affective Disorders*, vol. 114, pp. 163–73, 09 2008.

[3] A. Gupta, P. Mathur, S. Bijawat and A. Dadheech, “A novel work on analyzing stress and depression level of indian population during Covid-19”, *Recent Advances in Computer Science and Communications*, vol. 13, no. 11, 2020.

[4] World Health Organization. “Depression and other common mental disorders: global health estimates” *Technical Report*, 2017. License: CC BY-NC-SA 3.0 IGO.

[5] J. Zinken, K. Zinken, J. Clare Wilson, L. Butler, and T. Skinner, “Analysis of syntax and word use to predict successful participation in guided self-help for anxiety and depression”, *Psychiatry Research*, vol. 179, no. 2, pp. 181–186, 2010.

[6] S. Rude, E. M. Gortner, and J. Pennebaker, “Language use of depressed and depression-vulnerable college students”. *Cognition Emotion*, vol. 18, pp. 1121–1133, 12 2004.

[7] M. P. Caligiuri and J. Ellwanger, “Motor and cognitive aspects of motor retardation in depression,” *Journal of Affective Disorders*, vol. 57, no. 1, pp. 83–93, 2000.

[8] Psychomotor symptoms of depression. *American Journal of Psychiatry*, vol. 154, no. 1, pp. 4–17, 1997. PMID: 8988952.

[9] H. Kaya et al. Predicting depression and emotions in the cross-roads of cultures, para-linguistics, and non-linguistics. In *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop*, AVEC ’19, page 27–35, New York, NY, USA, 2019. Association for Computing Machinery.

[10] J. Ye, Y. Yu, Q. Wang, W. Li, H. Liang, Y. Zheng and G. Fu, “Multi-modal depression detection based on emotional audio and evaluation text”, *Journal of Affective Disorders*, 295:904–913, 2021.

[11] N. Aloshban, A. Esposito and A. Vinciarelli, “What you say or how you say it? depression detection through joint modeling of linguistic and acoustic aspects of speech”, *Cognitive Computation*, 02 2021.

[12] C. Yang, X. Lai, Z. Hu, Y. Liu and P. Shen. “Depression tendency screening use text based emotional analysis technique”, *Journal of Physics: Conference Series*, 1237:032035, 06 2019.

[13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, “Distributed representations of words and phrases and their compositionality”, In *Proceedings of the 26th International Conference on Neural Information Processing Systems* - Volume 2, NIPS’13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[14] D. Cer et al. “Universal sentence encoder for English”, In *Proceedings of the 2018 Conference on Empirical*

Methods in Natural Language Processing: System Demonstrations, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[15] A. Ray, S. Kumar, R. Reddy, P. Mukherjee and Ritu Garg. "Multi-level attention network using text, audio and video for depression prediction", In *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop*, AVEC '19, pp. 81–88, New York, NY, USA, 2019. Association for Computing Machinery.

[16] G. Singh Saggu, K. Gupta and K. V. Arya, "Depressnet: A multimodal hierarchical attention mechanism approach for depression detection", *International Journal of Engineering Sciences*, 2022.

[17] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding". In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] M. Rodrigues Makiuchi, T. Warnita, K. Uto and K. Shinoda, "Multimodal fusion of bert-cnn and gated cnn representations for depression detection", *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop*, 2019.

[19] H. Sun, H. Wang, J. Liu, Y.-W. Chen and L.n Lin, "Cubemlp: An mlp-based model for multimodal sentiment analysis and depression estimation", arXiv:2207.14087, 2022.

[20] Y. Kim. "Convolutional neural networks for sentence classification", In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[21] S. Yadav, J. Chauhan, J. Prakash Sain, K. Thirunarayan, A. Sheth and J. Schumm, "Identifying depressive symptoms from tweets: Figurative language enabled multitask learning framework". In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 696–709, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[22] J. Singh, N. Singh, M. M. Fouda, L. Saba and J. S. Suri, "Attention-enabled ensemble deep learning models and their validation for depression detection: A domain adoption paradigm," *Diagnostics*, vol. 13, no. 12, 2023.

[23] L. Tong et al. "Cost-sensitive boosting pruning trees for depression detection on twitter", *IEEE Transactions on Affective Computing*, pages 1–1, 2022.

[24] J. Liu and M. Shi, "A hybrid feature selection and ensemble approach to identify depressed users in online social media", *Frontiers in Psychology*, 12, 01 2022.

[25] N. Reimers and I. Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[26] K. Song, X. Tan, T. Qin, J. Lu and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding". arXiv preprint arXiv:2004.09297, 2020.

[27] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pages 770–778, 2016.

[28] F. Ringeval et al. "Avec 2019 workshop and challenge: State-of-mind, detecting depression with ai, and cross-cultural affect recognition", In *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop, AVEC '19,* page 3–12, New York, NY, USA, 2019. Association for Computing Machinery.

[29] L. I. Lin. "A concordance correlation coefficient to evaluate reproducibility", *Biometrics*, 45 1:255–68, 1989.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", In *Proceedings of the 32nd International Conference on International Conference on Machine Learning* - Volume 37, ICML'15, page 448–456. JMLR.org, 2015.

[31] H. Dibeklioglu, Z. Hammal and J. Cohn, "Dynamic multimodal measurement of depression severity using deep autoencoding", *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 03 2017.

[32] S. Sukhbaatar, J. Bruna, M. Paluri, L. D. Bourdev and R. Fergus, "Training convolutional networks with noisy labels", arXiv: Computer Vision and Pattern Recognition, 2014.

[33] M. Zhou, T. Liu, Y. Li, D. Lin, E. Zhou and T. Zhao, "Towards understanding the importance of noise in training neural networks", arXiv, abs/1909.03172, 2019.

[34] Q. Dong, S. Gong and X. Zhu, "Class rectification hard mining for imbalanced deep learning", In *2017 IEEE*

International Conference on Computer Vision (ICCV), pages 1869–1878, 2017.

[35] M. Bucher, S. Herbin and F. Jurie, "Hard negative mining for metric learning based zero-shot classification". In *ECCV Workshops*, 2016.

[36] M. Fang, S. Peng, Y. Liang, C.-C. Hung and S. Liu, "A multimodal fusion model with multi-level attention mechanism for depression detection", *Biomedical Signal Processing and Control*, 82:104561, 2023.

[37] C. Wang, D. Liu, K. Tao, X. Cui, G. Wang, Y. Zhao, Z. Liu, "A multi-modal feature layer fusion model for assessment of depression based on attention mechanisms". In *2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2022.

[38] Z. Han, Y. Shang, Z. Shao, J. Liu, G. Guo, T. Liu, H. Ding, Q. Hu, "Spatial-temporal feature networkfor speech-based depression recognition", *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2023.

[39] H. Sun, J. Liu, S. Chai, Z. Qiu, L. Lin, X. Huang and Y.-W. Chen, "Multi-modal adaptive fusion transformer network for the estimation of depression level", *Sensors (Basel, Switzerland)*, 21, 2021.

[40] S. Yin, C. Liang, H. Ding and S. Wang, "A multi-modal hierarchical recurrent neural network for depression detection", *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop*, 2019.

Conflict of Interest Notice
The authors declare that there is no conflict of interest regarding the publication of this paper.

Support/Supporting Organizations
Not applicable.

Ethical Approval and Informed Consent
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. Since the data used in this research is publicly available, an ethics committee approval is not required / not applicable.

Availability of data and material
Not applicable.

Plagiarism Statement
This article has been scanned by iThenticate ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 92-102
DOI: 10.35377/saucis...1339931
SAKARYA UNIVERSITY
RESEARCH ARTICLE

The Role of Attention Mechanism in Generating Image Captions: An Innovative Approach with Neural Network-Based Seq2seq Model

Zeynep Karaca [1], Bihter Daş [2]

[1] Department of Software Engineering, Technology Faculty, Firat University; Elazığ, Türkiye
[2] Department of Software Engineering, Technology Faculty, Firat University; Elazığ, Türkiye

Corresponding author:
Bihter Das, Firat University
Department of Software Engineering, Technology Faculty, Elazig/Turkey
bihterdas@firat.edu.tr

Article History:
Received: 09.08.2023
Accepted: 26.03.2024
Published Online: 26.03.2024

ABSTRACT
This study addresses important contributions to generating text from images, aiming to create meaning in various fields such as entertainment, communication, commerce, security, and education by establishing a connection between visual and textual content. This process aims to increase the accessibility, understandability, and processability of content by converting image data into meaningful text. Therefore, advances and studies in this field are extremely important. This study focuses on the effect of the combination of deep neural network models and attention mechanisms in creating more meaningful captions from images. Experiments performed on the Flickr8k dataset highlight the abilities of Seq2seq and VGG19 models to generate titles compatible with reference sentences. By using the dynamic focusing feature of the attention mechanism, the model effectively captures detailed aspects of images. The findings of this study have the potential to push the boundaries of multimodal data processing and representation with the effective integration of visual and textual information by adding information that the attention mechanism works more effectively together with the Seq2seq model.

Keywords: Image-to-text, Attention mechanism, Seq2seq model, VGG19, Image Capturing, Deep learning

1. Introduction

Natural Language Processing (NLP) has evolved into a significant field within the realm of artificial intelligence, focusing on the automatic analysis and representation of human language through computational algorithms [1]. NLP finds effective applications in various domains, such as understanding, translation, summarization, and sentiment analysis of texts. Particularly in today's rapidly advancing technological landscape, innovations in NLP hold paramount importance [2].

In this context, the creation of image captions stands as a critical domain at the intersection of Natural Language Processing, computer vision, and artificial intelligence [3]. Image captioning involves the ability to narrate image content using language, and research in this field serves essential purposes, such as improving the quality of life for visually impaired individuals and enhancing overall human-computer interaction [4].

With the proliferation of images on the internet and the advancements in artificial intelligence technologies, the significance of generating image captions has surged. Among the fundamental challenges in this domain is not only accurately and meaningfully describing the content of an image but also expressing the relationships between objects present within the image [5].

The process of generating image captions involves a combination of components from deep learning. Convolutional Neural Networks (CNNs) are employed for image processing, encoding images to extract features, while models like Recurrent Neural Networks (RNNs) are used for natural language processing to generate descriptions. This necessitates a successful integration of both visual and linguistic models [6,7].

The manuscript aims to show how much the use of the attention mechanism with a deep learning-based model contributes to the performance of the system. The proposed approach is demonstrated on a data set widely used in the literature because an accurate comparison with existing methods was desired. In the study, the performance of the attention mechanism was shown with two different deep neural network models. The attention mechanism was used with the VGG19 deep learning model in addition to the Seq2seq model, and the contribution of the proposed approach to the performance of the system was tested

with different methods. This study proposes a combined approach with a neural network-based model and attention mechanism for generating image captions. The suggested first neural network-based model employs an encoder-decoder architecture based on the Sequence-to-Sequence (Seq2seq) framework, with the incorporation of an attention mechanism to enhance performance. The second model is the VGG19 model, which is a deep model with 19 layers. The evaluation of the models is carried out using widely adopted metrics such as BLEU, METEOR, and ROUGE. In this study, current developments in image captioning are examined in depth, and the performance of the proposed approach and different methods are discussed.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the existing research conducted in this field. In Section 3, the utilized dataset, ESA model, Seq2Seq model, encoder-decoder architecture, and attention mechanism employed in the experimental methodology are discussed. Section 4 presents the experimental results concerning the proposed method, and finally, the overall contributions of the study are presented in the Conclusion section.

2. Literature Review

Previous studies on image-to-text generation highlight the effective integration of deep learning and natural language processing techniques. This section extensively examines these approaches in the literature. Various studies have explored the generation of image captions, highlighting diverse approaches.

Yue et al. [8] proposed a method for Thangka image recognition based on an attention mechanism and encoder-decoder architecture. They utilized the ResNet101+ Convolutional Block Attention Module (CBAM) to extract image features and incorporated an attention mechanism into the encoder for improved feature extraction. Using Long Short-Term Memory (LSTM) as the decoder, they validated the model using the Thangka dataset and Flickr8k dataset. The results demonstrated the higher success of the ResNet101 + CBAM model compared to ResNet101.

Chandaran et al. [9] suggested a YOLOv5 model combined with Bidirectional Long Short-Term Memory (Bi-LSTM) for object recognition and feature extraction. They evaluated performance using the Bleu metric and achieved favorable outcomes.

Bhadauria et al. [10] combined two different LSTM networks with CNN using the Flickr8k dataset. Due to the small dataset size, accuracy levels were not significantly high.

Shaikh et al. [11] adopted an alternative approach using an encoder-decoder architecture. They employed Convolutional Neural Networks (CNN) as the encoder and Gated Recurrent Unit (GRU) as the decoder. The results indicated that the GRU model with an open neighborhood connection exhibited higher success.

Xue et al. [12] proposed a model combining the cloze-style approach with neural networks. They employed WGAN to generate sentence templates with ,broader visual coverage and utilized CNN to fill in gaps in visual regions using object detectors. The model was trained on Flickr8k and MSCOCO datasets, showcasing superior performance.

Singh et al. [13] utilized a hybrid model with CNN and LSTM, trained on the Flickr8k dataset, achieving a Bleu score of 0.52.

Han et al. [14] employed an interpretable image captioning generator model based on determining why specific objects are present in an image. Their model featured a generation module and an explanation module, using an encoder-decoder architecture to generate captions. The explanation module constructed a weight matrix for all words in the generated Caption from detected regions in the image.

Wang et al. [15] utilized an end-to-end deep learning model with a semantic attention mechanism for caption generation. They calculated the similarity between end-to-end frame image feature sequences and semantic word sequences using a derived structure. The model was applied by transferring English information from the Flickr8k dataset to Chinese, showcasing a 3.9% improvement over state-of-the-art approaches. In Chinese captions, the model achieved Bleu-1 63.7, Bleu-2 49.4, Bleu-3 37.2, Bleu-4 28.7, Rouge_L 53.34, and CIDEr 51.45, generally higher than English except for Bleu-1. On the MS COCO dataset, the model achieved Bleu-1 73.1, Bleu-2 55.9, Bleu-3 43.4, Bleu-4 32.8, Rouge_L 25.49, and CIDEr 95.1 values.

Chen et al. [16] introduced the SGGC (Scene Graph Guiding Captioning) model, aiming to bridge the semantic gap between scene graphs, images, and captions to generate better sentences. The model employed scene graphs for decoder generation and used an attention mechanism for caption production. It was evaluated on MS COCO and Flickr30k datasets, with MS COCO achieving Bleu-1 77.2, Bleu-2 60.7, Bleu-3 46.2, Bleu-4 36.3, Meteor 27.8, CIDEr 116.5, and Rouge_L 56.7 scores. For the Flickr30k dataset, it obtained Bleu-1 66.9, Bleu-2 49.4, Bleu-3 35.1, Bleu-4 24.8, Meteor 20.3, CIDEr 116.5, and Rouge_L 53.3 scores.

Rafi et al. [17] utilized Linear Substructures, a model understanding word relationships and sequential orders for caption generation. They focused on regions of motion in images using a common variance shift and employed Inceptionv3

architecture with LSTM encoders to extract image features. Flickr8k dataset was used, resulting in improved performance compared to CNN-based approaches, with Bleu-1 at 81.1 and Bleu-2 at 38.6.

Sharma et al. [18] employed a Lightweight Transformer architecture with a GRU-integrated decoder. They reduced the standard encoder-decoder architecture to an Inceptionv3 + Transformer encoder and Transformer decoder. Model evaluation on the MS COCO dataset yielded Bleu-1 81.0, Bleu-2 65.2, Bleu-3 65.2, Bleu-4 37.8, Meteor 27.9, CIDEr 123.1, and Rouge 58.0 scores.

Panigrahi et al. [19] proposed a model leveraging relationships between image regions, caption words, and RNN model states. VGG16 was used as an encoder to convert image regions to feature vectors, which were then used as inputs to an LSTM-based decoder. The LSTM decoder predicted word sequences to generate descriptions. The model, trained for 20 epochs on the Flickr8k dataset, achieved 20.28% loss and 85% accuracy.

Sun et al. [20] proposed the Bidirectional Beam Search (BiBS) method for bidirectional inference. They introduced the method of Gap-filling Image Captioning, considering past and future sentence structures to obtain accurate image captions. The Visual Madlibs dataset was used. Tested on the MS COCO dataset, the method outperformed baseline techniques. The BiRNNBiBS method achieved Bleu-1 0.470 and Bleu-2 0.389 scores.

Keneshloo et al. [21] combined Reinforcement Learning and the seq2seq model to integrate decision-making and long-term memory. They employed their own models for training the DCA (SCPG) based model, which achieved Rouge-1 41.69, Rouge-2 19.47, and Rouge-L 37.92 scores.

Sahrial Alam et al. [22] evaluated performance using five different models (VGG16, ResNet50, InceptionV3, DenseNet201, Xception) on the Flickr8k dataset. Model accuracies were: VGG16 0.83, ResNet50 0.87, InceptionV3 0.80, DenseNet201 0.87, and Xception 0.81.

Zhou et al. [23] proposed the Triplet Sequence Generative Adversarial Networks (TSGAN) model for unsupervised image captioning. Tested on the MSCOCO dataset, it achieved Bleu-1 46.2, Bleu-2 26.8, Bleu-3 13.5, Bleu-4 6.9, Meteor 13.0, Rouge 32.3, CIDEr 28.9, and Spice 8.3 scores.

Kushwaha et al. [24] utilized the VGG19+LSTM model, extracting image features using the VGG19 model. Compared with other models on the Flickr8k dataset, the VGG16+LSTM model received the highest Bleu score of 55.9.

Liu et al. [25] proposed the Vocabulary-Critical Sequence Training (VCST) method, a novel Reinforcement Learning approach assigning distinct values to words at each step. MS COCO 2014 dataset was used, and the VCST method was applied with the SCST training method for ATT2in, Top-Down, Up-Down, and SGAE models. The UpDown+SCST+VCST model achieved Bleu-4 38.0 and CIDEr-D 125.0 scores.

Zheng et al. [26] proposed the Di-vCon method, which generates explanations with various semantic concepts. The method consists of two steps. In the first step, a concept sequence generator is developed to automatically generate concept sequences in reverse order. The second step involves a sentence generator that takes concept sequences as input and produces descriptions for each sequence. The model focuses more on less frequent objects and achieves optimal performance on the MS COCO dataset with beam size two and group size 1.

Table 1 Some of the Studies for Text Production from Images

Author	Dataset	Method	Results
Singh et al. [13]	FLICKR8K	CNN+LSTM	BLEU=0.52
Chen et al. [16]	MSCOCO	SGGC+Attention Mechanism	BLEU1=77.2 BLEU2=60.7 BLEU3=46.2 BLEU4=36.3 METEOR=27.8 CIDEr=116.5 ROUGE-L=56.7
	FLİCKR30K		BLEU1=66.9 BLEU2=49.4 BLEU3=35.1 BLEU4=24.8 METEOR=20.3 CIDEr=116.5 ROUGE-L=53.3
Rafi et al. [17]	FLICKR8K	InceptionV3+LSTM	BLEU1=81.1 BLEU2=38.3
Sharma et al. [18]	MSCOCO	InceptionV3+ Transformer	BLEU1=81.0 BLEU2=65.2 BLEU3=65.2 BLEU4=37.8 METEOR=27.9 CIDEr=123.1 ROUGE-L=58.0
Zhou et al. [23]	MSCOCO	TSGAN	BLEU1=46.2 BLEU2=26.8 BLEU3=13.5 BLEU4=6.9 METEOR=13.0 CIDEr=28.9 ROUGE=32.3 SPICE=8.3
Kushwaha et al. [24]	FLICKR8K	VGG16+LSTM	BLEU=55.9

Birmingham et al. [27] introduced the KENGIC model for image captioning based on keyword and n-gram graph. A series of image keyword nodes are connected through overlapping n-grams to form a directed graph, generating titles from the most probable n-gram sequences. MS COCO dataset was used. The KENGIC model's performance results on the COCO Karpathy dataset are Bleu-1 66.3, Bleu-4 18.6, Meteor 22.6, Rouge 40.4, CIDEr 67.8, and Spice 18.5.

Dwivedi et al. [28] used a 5-layer CNN for extracting image features and an RNN for processing text data. The proposed CNN-5 model was compared with transfer learning models VGG-16 and VGG-19 using the MNIST dataset. CNN-5 model achieved Bleu-1 0.5348, Bleu-2 0.2357, Bleu-3 0.1488, and Bleu-4 0.0660 scores. VGG-16 model achieved Bleu-1 0.5362, Bleu-2 0.2566, Bleu-3 0.1432, and Bleu-4 0.0591 scores. VGG-19 model achieved Bleu-1 0.5240, Bleu-2 0.2430, Bleu-3 0.1370, and Bleu-4 0.0511 scores. The highest Bleu-1 and Bleu-2 values were in the VGG-16 model, while the highest Bleu-3 and Bleu-4 values were in the CNN-5 model. Table 1 lists some of the existing works for image-to-text generation.

In the general evaluation of studies in the literature, the diversity observed between different methods and data sets is striking. The change in performance values depending on the methods used in the studies shows that Transformer-based models, such as InceptionV3+Transformer [18], are more successful than other models. Transformer architecture has shown impressive results in the field of generating text from images, especially with its success in natural language processing tasks. This success can be specifically attributed to the ability to better capture meaning in language. On the other hand, the data sets used in the studies appear to have a significant impact on performance. For example, larger or more diverse data sets generally provide better generalization ability. This situation has been observed especially in studies using data sets such as FLICKR8K and MSCOCO. More diverse data sets allowed the model to better adapt to different scenarios and produce more universal results. It is important to note that although Transformer-based models are generally more successful, this is not the case for every dataset. For example, studies with high BLEU scores obtained with the VGG16+LSTM model [24] show that different models can be effective in certain situations. As a result, the interaction between the methods and data sets used in studies of producing text from images has a complex structure. Current studies show that different approaches are needed to improve the performance of the system. For this reason, this article shows how the use of the attention mechanism, together with deep learning-based models, increases the performance in text generation.

3. Material and Methods

In this study, image captions were created using the Sequence to Sequence (Seq2seq) model, which is based on the encoder and decoder architecture and incorporates the attention mechanism. The experiment is performed on the Flickr8k dataset, and the file containing the images and their corresponding captions goes through the necessary preprocessing steps. It is desired to measure how much the use of the attention mechanism with a neural network-based model will affect performance. The proposed system with the Attention mechanism was also compared with the VGG19 model. The Seq2seq model architecture comprises an encoder and a decoder. The encoder processes the image features and encodes them into a fixed-size representation. On the other hand, the decoder takes this representation as input and generates captions. The encoder is responsible for extracting image features, and the decoder utilizes these extracted features to create image captions. The inclusion of the Attention Mechanism enhances the decoder's capability to access all hidden states of the encoder, thereby facilitating the generation of more meaningful and comprehensive sentences. The VGG19 model, a remarkable Convolutional Neural Network, was used in this study together with the attention mechanism to increase the overall performance of the text generation process from the image.

3.1 Dataset

Various datasets are utilized in image captioning research. In the literature, datasets such as MS COCO (Microsoft Common Objects in Context), Flickr30k, and Flickr8k are commonly employed for generating English captions. The MS COCO dataset contains high-quality images, consisting of 118,287 images for training and 5,000 images for testing. In total, there are 123,287 images in this dataset, each accompanied by five captions. The Flickr8k dataset consists of 8,091 images typically depicting humans and animals, with corresponding captions for each image. The Flickr30k dataset focuses on images primarily showing people in various events and is an extended version of the Flickr8k dataset. It contains 31,783 images. In this study, the Flickr8k dataset is used. This dataset comprises a total of 8,091 images, with 6,000 for training, 1,000 for testing, and 1,000 for validation. Multiple captions are available for each image in the Flickr8k dataset. Figure 1 shows some tagged images from the Flickr8k dataset.

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of neural network that is effective in computer vision tasks, particularly image processing. It aims to extract high-level features using a structure based on n-grams or words, typically used for feature extraction. It finds applications in various domains like image recognition, classification, and text generation. CNN primarily consists of specially designed convolutional layers to process image data [29]. These layers emphasize specific features in the image, aiming to obtain more meaningful and lower-dimensional representations of these features. Pooling layers reduce the dimension of these feature maps while minimizing significant information loss. Activation layers nonlinearly adjust the computed features. The structure of CNN is particularly efficient in extracting and representing features from images (Figure

2). These features can then be used by a decoder for various tasks such as caption generation, classification, or detection. The different layers of convolutional neural networks represent different stages of image processing, while fully connected layers transform these features into actionable results [30].

Figure 1 The used Flickr8k dataset images

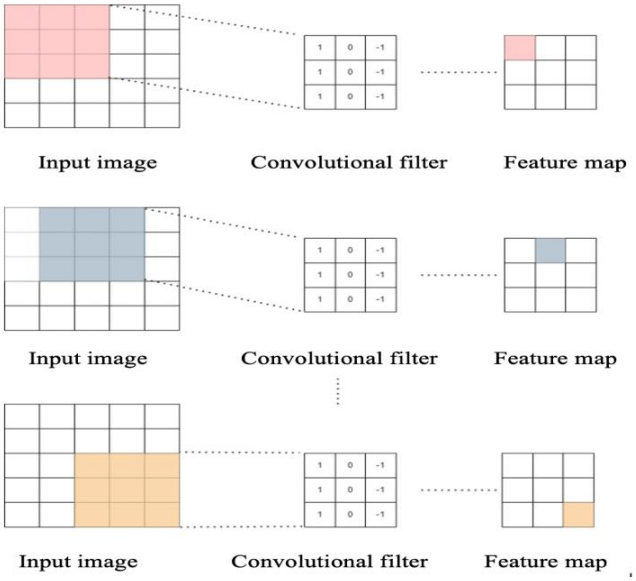


Figure 2 The CNN Architecture Internal Structure

3.3 Sequence-to-Sequence (Seq2Seq) Model

The Sequence-to-Sequence (Seq2seq) model is a deep learning architecture used for tasks such as translating, transforming, or generating sequences of varying lengths between input and output. It consists of two fundamental components: an encoder and a decoder. The encoder converts the input sequence into a fixed-size vector representation [31]. The decoder takes this vector and generates the output sequence. The Seq2seq model has achieved notable success in tasks like language translation, text generation, and speech recognition. Advanced versions incorporating techniques like attention mechanisms have also been developed. In the Seq2seq model, the encoder reads the sequence of input data and passes it to a sequence, which is then fed as input to the decoder to generate the output sequence. As Seq2seq deals with sequential data, both the encoder and decoder typically involve a recurrent structure (RNN, LSTM, GRU). An RNN, for instance, takes into account both the current time step's input and the input from the previous time step. The output at time step "t" is generated based on both the input at time step "t" and the input at time step "t-1". The hidden state in the model retains sequential information and is used in the next step of the process. The encoder takes a sequence as input data. The entire input data is compressed by the encoder into a fixed-size vector, which is then passed to the decoder. The decoder takes this sequence as input. To predict the output data, the decoder progresses from the previous time step's (t-1) hidden state, using the information present until the process is completed. This way, the encoder's hidden state, composed only of the final outputs, struggles less in fully forming sentences [32]. Figure 3 shows the used Seq2seq architecture.

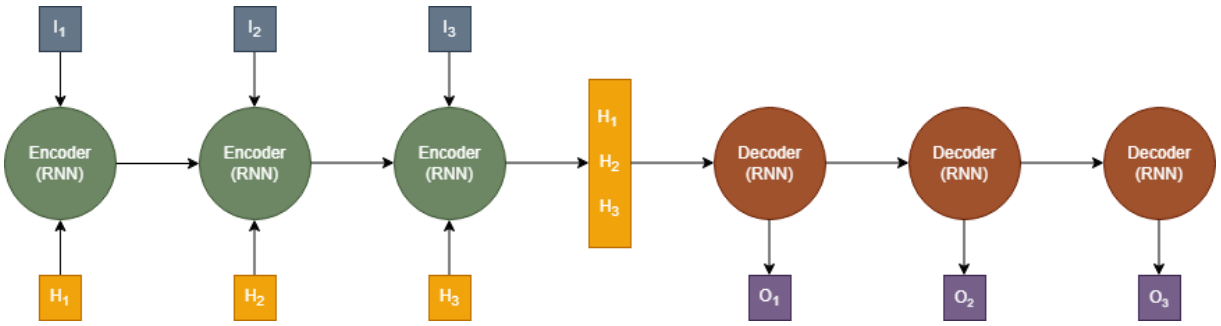


Figure 3 The Used Seq2seq Deep Learning Model Architecture

3.4 Encoder-Decoder

The captions for the input images in the Flickr8k dataset are generated using a Seq2seq model, which is built upon the encoder-decoder architecture. The image is first fed into the encoder. The images are resized to 224x224xRGB dimensions. The encoder extracts features from the image and creates a feature vector. This generated vector is then provided as input to the decoder. As a preprocessing step for the captions, punctuation marks between words are removed, and <s> and <e> tags are added at the beginning and end of each sentence. Tokenization processes are then applied. Using the image feature vector received from the encoder, the decoder constructs the captions.

3.5 Attention Mechanism

An attention mechanism is employed to generate comprehensive captions that encompass all details of the image. The attention mechanism emphasizes relevant information present in the image. This mechanism is based on two fundamental aspects [33]. Firstly, it determines which parts of the input need to be focused on. Secondly, it extracts the features of significant regions. The attention mechanism modifies the context vector at each time step in the decoder based on the similarity of the decoder hidden states. This mechanism is integrated into the encoder-decoder architecture to enhance the generation of image captions with rich details [34].

3.6 VGG19 model

VGG models are CNN models used for image classification. The VGG19 model is named after the Visual Geometry Group and is a model trained using more than 1 million images in the ImageNet database. It is a 19-layer model. The VGG19 model is the VGGNet model with 19 weight layers [35]. It consists of 16 convolutional layers, 3 fully connected layers, 5 max-pool, and 1 softmax layer. The model takes images as input. Once the image is ready for processing, the final image features are extracted.

4. Experimental Results

In text processing studies, different evaluation metrics are employed. BLEU, METEOR, and ROUGE are the most commonly used metrics. These metrics take values between 0 and 100, where a value closer to 100 indicates that the machine translation is similar to the reference translation, while a value closer to 0 suggests that the generated machine translation diverges from the reference translation. The performance evaluation results for Seq2seq and VGG19 models with attention mechanism obtained for text generation on the Flickr8k dataset using the proposed approach are shown in Table 2. Also, Text in English created using images from the Flickr8k dataset is shown in Table 3. Additionally, Table 4 shows a comparison of the proposed approach with existing studies in the literature.

Table 2 Experimental Results for Seq2seq and VGG19 Models

Evaluation Metrics	Performance of the Seq2seq model	Performance of the VGG19 model

BLEU-1	71,42	66,66
BLEU-2	59,76	51,63
BLEU-3	41,85	1,92…e-100
BLEU-4	6,31…e-76	1,07…e-152
METEOR	98.13	62,5
ROUGE_L	83,33	91,42

Table 3. The real captions and prediction captions results

Image	Captions
	Real Caption: black dog playing with green toy Seq2seq Prediction Caption: black dog plays with a green toy VGG19 Prediction Caption: black dog with a green object
	Real Caption: girl with long hair flying in the breeze while she swings Seq2seq Prediction Caption: The girl in the pink top is swinging with her hair flying everywhere VGG19 Prediction Caption: The girl is swinging with her hair flying everywhere
	Real Caption: A man in a red shirt sits on his dirt bike and points at the camera Seq2seq Prediction Caption: The man with the bike is wearing a helmet is on the bike and pointing at the camera VGG19 Prediction Caption: male bikes through the middle of the mountain
	Real Caption: Real Caption: dirt bike rider jumping down the hill Seq2seq Prediction Caption: A person on a BMX bike is riding on an outdoor course VGG19 Prediction Caption: person rides a vehicle
	Real Caption: The hiker is shadowed by the time of day near an open body of water Seq2seq Prediction Caption: backpacker looks at the ocean sky above the ocean VGG19 Prediction Caption: hiker standing on the shore of the lake
	Real Caption: Young men playing basketball in a competition Seq2seq Prediction Caption: four men playing basketball with the team are in action VGG19 Prediction Caption: A basketball player in white is running with behind him
	Real Caption: an adult and child on bleachers near the water Seq2seq Prediction Caption: A man in a cowboy hat sits on bleachers in the park VGG19 Prediction Caption: Man is sitting on bleachers in front of the lake

When the performance of the Seq2seq and VGG19 models is evaluated, we observe that the Seq2seq model achieves more impressive results with the effective impression method. While the Seq2seq model exhibited a high similarity with the BLEU-1 score (71.42), it also achieved remarkable success with the METEOR score (98.13) and ROUGE_L score (83.33). On the other hand, the VGG19 model performs slightly lower on similar metrics. Especially in the second image, we see

that the Seq2seq model successfully captures details such as the "pink top." However, he omitted the "red shirt" detail in the third image. In general, the Seq2seq model is better at preserving visual details. The VGG19 model has a more general and abstract perspective. This shows how the attention mechanism, in combination with the Seq2seq model, is more effective, especially in preserving visual details. While the Seq2seq model attracts attention with its ability to integrate visual and textual information more effectively, the VGG19 model focuses on a more general perspective. This study shows that the combination of the Seq2seq model and the attention mechanism can combine visual and textual information in a more meaningful and comprehensive way. More specifically, the attention mechanism of the Seq2seq model tends to preserve visual details better, making it preferable to the VGG19 model.

Table 4 Comparison of Proposed Approach with Other Studies

Author	Dataset	Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L
Singh et al. [13]	FLICKR8K	CNN+LSTM	0.52	--	--	--	--	--
Rafi et al. [17]	FLICKR8K	InceptionV3+LSTM	81.1	38.3	--	--	--	--
Kushwaha et al. [24]	FLICKR8K	VGG16+LSTM	55.9	--	--	--	--	--
Proposed approach with Seq2seq	FLICKR8K	Seq2seq	71,42	59,76	41,85	6,31…e-76	98.13	83,33
Proposed approach with VGG19	FLICKR8K	VGG19	66,66	51,63	1,92…e-100	1,07…e-152	62,5	91,42

In the study, Seq2seq and VGG19 models, evaluated with common metrics such as BLEU, METEOR, and ROUGE, were used to measure the performance of the proposed approach. Looking at Table 4, the Seq2seq model has higher values in BLEU-1 and BLEU-2 metrics compared to other studies. Especially in BLEU-1, it achieved a success of 71.42%. This indicates that the Seq2seq model contributes to a better matching of words under the proposed approach than similar studies in previous literature. In the ROUGE_L metric, the VGG19 model showed a higher performance compared to other studies (91.42%). This may indicate that VGG19, enhanced by the attention mechanism, has better similarity with reference texts. When we look at the comparisons in Table 4, it can be seen that the proposed approach, Seq2seq+Attention mechanism, has a competitive advantage compared to previous studies. However, the VGG19 model performed poorly on certain metrics, especially when compared to other models. The use of the attention mechanism showed a particularly pronounced effect on the Seq2seq model. The Seq2seq model achieved better results when supported by the attention mechanism and stood out, especially in BLEU-1 and BLEU-2 metrics. This indicates that the attention mechanism helps in making better word alignments under the proposed approach in the text generation task.

5. Conclusion

This study aims to make a substantial contribution to the existing literature by thoroughly investigating the integration of deep neural network models and the utilization of the attention mechanism in the realm of image-to-text conversion. The proposed approach, implemented in conjunction with the Seq2seq model, renowned for its proficiency in sequential data transformation, and VGG19, a widely adopted convolutional neural network model, was subjected to rigorous comparisons with other studies in the literature. Our experiments, conducted on the Flickr8k dataset, underscore the exceptional capability of our proposed approach in generating captions that closely align with reference sentences. The dynamic focusing facilitated by the attention mechanism on various parts of the images enhances the captions by capturing intricate details. Future endeavors will leverage larger and more diverse datasets, delve into advanced attention mechanisms, and explore transfer learning and fine-tuning techniques to enhance adaptability across different domains. Our findings emphasize the potential of our approach in visual understanding, content creation, and human-computer interaction. The proficient integration of visual and textual information positions our model as a valuable asset in the ever-evolving landscape of multimodal data processing. With the continuous advancements in deep learning, our proposed approach holds the promise of pushing the boundaries of efficient multimodal data representation and processing, underlining the superiority of Seq2seq coupled with the attention mechanism in achieving compelling results.

References

[1] T. Alqahtani et al., “The emergent role of artificial intelligence, natural learning processing, and large language models in higher education and research," *Research in Social and Administrative Pharmacy*, vol. 19, no. 8, pp. 1236–1242, Aug. 2023, doi: 10.1016/j.sapharm.2023.05.016.

[2] J. J. Cavallo, I. de Oliveira Santo, J. L. Mezrich, and H. P. Forman, “Clinical Implementation of a Combined Artificial Intelligence and Natural Language Processing Quality Assurance Program for Pulmonary Nodule Detection in the Emergency Department Setting”, *Journal of the American College of Radiology*, vol. 20, no. 4, pp. 438–445, Apr. 2023, doi: 10.1016/j.jacr.2022.12.016.

[3] J. Doe and A. Smith, "Recent advances in image captioning: A comprehensive survey," *IEEE Transactions on Artificial Intelligence*, vol. 7, no. 3, pp. 210-225, 2022.

[4] M. Johnson, B. Brown, and C. Wilson, "Innovative Approaches for image caption generation using attention mechanisms," *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2021, pp. 750-760.

[5] S. Kim and E. Lee, "Enhancing image captioning performance through multimodal fusion techniques," *IEEE Transactions on Multimedia*, vol. 25, no. 6, pp. 1350-1365, 2020.

[6] L. Wang, H. Chen, and X. Zhang, "Leveraging transformers for improved image captioning," *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, 2018, pp. 240-255.

[7] R. Patel and S. Gupta, "Attention is all you need: Exploring self-attention mechanisms in image captioning," *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, 2019, pp. 1800-1810.

[8] C. Yue, W. Hu, H. Song, and W. Kang, “Thangka image caption method based on attention mechanism and encoder-decoder architecture” In *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Nis. 2022, pp. 1752-1756. doi: 10.1109/ICSP54964.2022.9778737.

[9] S. R. Chandaran, S. Natesan, G. Muthusamy, P. K. Sivakumar, P. Mohanraj, and R. J. Gnanaprakasam, “Image captioning using deep learning techniques for partially impaired people” In *2023 International Conference on Computer Communication and Informatics (ICCCI)*, Oca. 2023, pp. 1-6. doi: 10.1109/ICCCI56745.2023.10128287.

[10] S. S. Bhadauria, D. Bisht, T. Poongodi, and S. A. Yadav, “Assertive vision using deep learning and LSTM” In *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Şub. 2022, pp. 761-764. doi: 10.1109/ICIPTM54933.2022.9754057.

[11] M. K. Shaikh and M. V. Joshi, “Recursive network with explicit neighbor connection for image captioning” In *2018 International Conference on Signal Processing and Communications (SPCOM)*, Tem. 2018, pp. 392-396. doi: 10.1109/SPCOM.2018.8724400.

[12] Z. Xue, L. Wang, and P. Guo, “Slot based image captioning with WGAN” In *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, Haz. 2019, pp. 241-246. doi: 10.1109/ICIS46139.2019.8940218.

[13] A. Singh et al., “Image captioning using python” In *2023 International Conference on Power, Instrumentation, Energy and Control (PIECON)*, Şub. 2023, pp. 1-5. doi: 10.1109/PIECON56912.2023.10085724.

[14] S.-H. Han and H.-J. Choi, “Explainable image caption generator using attention and Bayesian inference” In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, Ara. 2018, pp. 478-481. doi: 10.1109/CSCI46756.2018.00098.

[15] B. Wang et al., “Cross-lingual image caption generation based on visual attention model” *IEEE Access*, vol. 8, pp. 104543-104554, 2020, doi: 10.1109/ACCESS.2020.2999568.

[16] H. Chen et al., “Captioning transformer with scene graph guiding” In *2021 IEEE International Conference on Image Processing (ICIP)*, Eyl. 2021, pp. 2538-2542. doi: 10.1109/ICIP42928.2021.9506193.

[17] S. Rafi and R. Das, “A linear sub-structure with co-variance shift for image captioning” In *2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, Kas. 2021, pp. 242-246. doi: 10.1109/ISCMI53840.2021.9654828.

[18] D. Sharma et al., “ghtweight transformer with GRU integrated decoder for image captioning” In *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Eki. 2022, pp. 434-438. doi: 10.1109/SITIS57111.2022.00072.

[19] L. Panigrahi et al., “Hybrid image captioning model” In *2022 OPJU International Technology Conference on Emerging*

Technologies for Sustainable Development (OTCON), Şub. 2023, pp. 1-6. doi: 10.1109/OTCON56053.2023.10113957.

[20] Q. Sun et al., "Bidirectional Beam Search: Forward-Backward inference in neural sequence models for fill-in-the-blank image captioning" In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Tem. 2017, pp. 7215-7223. doi: 10.1109/CVPR.2017.763.

[21] Y. Keneshloo et al., "Deep reinforcement learning for sequence-to-sequence models", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2469-2489, Tem. 2020, doi: 10.1109/TNNLS.2019.2929141.

[22] M. Sahrial Alam et al., "Arison of different CNN model used as encoders for image captioning" In *2021 International Conference on Data Analytics for Business and Industry (ICDABI)*, Eki. 2021, pp. 523-526. doi: 10.1109/ICDABI53623.2021.9655846.

[23] Y. Zhou et al., "Triple sequence generative adversarial nets for unsupervised image captioning" In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Haz. 2021, pp. 7598-7602. doi: 10.1109/ICASSP39728.2021.9414335.

[24] R. Kushwaha and A. Biswas, "Hybrid feature and sequence extractor based deep learning model for image caption generation" In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Tem. 2021, pp. 1-6. doi: 10.1109/ICCCNT51525.2021.9579897.

[25] H. Liu et al., "Vocabulary-wide credit assignment for training image captioning models" *IEEE Transactions on Image Processing*, vol. 30, pp. 2450-2460, 2021, doi: 10.1109/TIP.2021.3051476.

[26] Y. Zheng et al., "Divcon: Learning concept sequences for semantically diverse image captioning" In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Haz. 2023, pp. 1-5. doi: 10.1109/ICASSP49357.2023.10094565.

[27] B. Birmingham and A. Muscat, "KENGIC: Keyword-driven and N-Gram graph based image captioning" In *2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov. 2022, pp. 1-8. doi: 10.1109/DICTA56598.2022.10034584.

[28] P. Dwivedi and A. Upadhyaya, "A Novel deep learning model for accurate prediction of image captions in fashion industry" In *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Oca. 2022, pp. 207-212. doi: 10.1109/Confluence52989.2022.9734171.

[29] F.Akalin and N. Yumusak, "Detection and classification of white blood cells with an improved deep learning-based approach," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 30, no. 7, article 16. https://doi.org/10.55730/1300-0632.3965

[30] F. Akalin, and N.Yumusak. "Classification of ALL, AML and MLL leukaemia types on microarray dataset using LSTM neural network Approach" , *Journal of Faculty of Engineering and Architecture of Gazi University*, vol. 38, no. 3, 2023, pp. 1299-1306.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks" in *Advances in Neural Information Processing Systems*, (pp. 3104-3112), 2014.

[32] P. Anderson et al., "Bottom-up and top-down attention for image captioning and visual question answering" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 6, pp. 1416-1432, 2018.

[33] J. Xie et al., "A multimodal fusion emotion recognition method based on multitask learning and attention mechanism", *Neurocomputing*, p. 126649, Aug. 2023, doi: 10.1016/j.neucom.2023.126649.

[34] K. Yang et al., "A multi-sensor mapping Bi-LSTM model of bridge monitoring data based on spatial-temporal attention mechanism", *Measurement*, vol. 217, p. 113053, Aug. 2023, doi: 10.1016/j.measurement.2023.113053.

[35] H. Won, B. Kim, I.-Y. Kwak, ve C. Lim, "Using various pre-trained models for audio feature extraction in automated audio captioning," *Expert Systems with Applications*, c. 231, s. 120664, Kas. 2023, doi: 10.1016/j.eswa.2023.120664.

Conflicts of Interest

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

Availability of Data and Material

Not applicable.

Ethical Approval and Informed Consent
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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 103-111
DOI: 10.35377/saucis...1444155
SAUCIS
SAKARYA UNIVERSITY

RESEARCH ARTICLE

Predicting Engine Emissions Using Eco-Friendly Fuels for Sustainable Transportation

Beytullah Eren [1], İdris Cesur[2]

[1] Sakarya University, Faculty of Engineering, Department of Environmental Engineering, Sakarya, Türkiye.
[2] Sakarya University of Applied Sciences, Faculty of Technology, Department of Mechanical Engineering, Sakarya, Türkiye.

Corresponding author:
Beytullah Eren, Sakarya University, Faculty of Engineering, Department of Environmental Engineering
beren@sakarya.edu.tr

Article History:
Received: 28.02.2024
Accepted: 03.04.2024
Published Online: 03.04.2024

ABSTRACT
In recent years, increasing concerns about vehicle emissions' environmental and public health impacts have led to the desire to use eco-friendly fuels as alternatives to traditional fossil fuels. Biofuels, hydrogen, and electric power offer lower greenhouse gas emissions and improved air quality, resulting in their development and adoption globally. Predicting emissions using these fuels is crucial for assessing their environmental benefits. This study proposes using artificial neural networks (ANN), a machine learning technique, to accurately predict emissions associated with eco-friendly fuels across different compositions and engine speeds. The ANN model strongly correlates with predicted and observed emissions values, indicating its effectiveness. The training dataset had an R-value of 0.99928, the test dataset had an R-value of 0.99937, and the validation dataset had an R-value of 0.99904. When all datasets were combined, the overall R-value of 0.99927 confirmed the model's accuracy in capturing the data patterns. This study underscores the importance of adopting innovative approaches to address environmental challenges and promote sustainable transportation solutions. It contributes to reducing the adverse effects of vehicle emissions on air quality and public health by assisting policymakers, car manufacturers, and city planners in making effective decisions. Moreover, It promotes environmental sustainability by providing valuable insights into vehicle emissions prediction and guiding the development of eco-friendly fuels for a more efficient transportation system.

Keywords: Engine emissions prediction, Eco-friendly fuels, Artificial neural network, Environmental sustainability, Sustainable transportation

1. Introduction

In recent years, there has been a growing global concern about environmental pollution and its detrimental effects on human health. Pollution originating from various sources, including industrial emissions, vehicle exhaust, agricultural runoff, and improper waste disposal, has led to air, water, and soil contamination, posing serious risks to the environment and public health. Vehicular emissions, in particular, play a significant role in this issue, contributing substantially to the worsening levels of air pollution. The rising pollution, especially from vehicles on roads, presents a major challenge due to rapid urban growth and increased reliance on private transportation, further exacerbating air quality issues. As a result, the increasing number of vehicles with internal combustion engines is a primary concern for air quality today. A significant portion of urban air pollution comes from vehicles with these engines, and the type of pollutants and their concentrations depend on factors like engine type, tuning, driving habits, fuel makeup, and weather conditions. When fossil fuels are burned in vehicle engines, they release various pollutants into the air, including carbon monoxide (CO), nitrogen oxides (NOx), particulate matter (PM), and volatile organic compounds (VOCs) [1]. Achieving ideal conditions for complete combustion is practically impossible, leading to incomplete combustion and the creation of additional pollutants. Vehicle exhaust gases, responsible for 75% of total pollutants, contain a mixture of unburned hydrocarbons like kinds of paraffin, olefins, and aromatics, as well as partially burned hydrocarbons such as aldehydes, ketones, and carboxylic acids. They also contain CO, NOx, lead compounds, and particulate matter. What makes emissions from combustion noteworthy is their polluting properties and their immediate and significantly harmful toxic effects, distinguishing them from emissions originating from other sources. These emissions, which substantially negatively impact human health and environmental quality, can be classified into six main categories: carbon oxides, nitrogen oxides, sulfur compounds, hydrocarbons, aldehydes, and particulates [2]. These pollutants deteriorate air quality and present significant health hazards to people, resulting in respiratory illnesses, cardiovascular issues, and other harmful health outcomes [3], [4]. Alternative fuels are crucial for reducing vehicle emissions because they emit fewer pollutants, support climate change mitigation, vary energy sources, and often have lower lifecycle emissions than fossil fuels. Using environmentally friendly fuels like biofuels, natural gas, electricity, and propane in vehicles has led to substantial

reductions in emissions. In contrast to fossil-based (FB) fuels, biohydrogen, biomethane, biodiesel, and bioethanol have shown considerable emission reductions of 70%, 63%, 41%, and 54%, respectively, when utilized in vehicles [5]. As urbanization and industrialization expand globally, vehicle emissions and resulting air pollution are increasing, highlighting the urgent need for measures to reduce their impact, as controlling these pollutants has become essential due to their direct or indirect endangerment to human health and the environment. Challenges in predicting engine emissions when using eco-friendly fuels include the need for significant modifications to motor vehicles to accommodate biomethane as a fuel, alongside the potential increase in emissions of certain pollutants. Predicting engine emissions is crucial for several reasons: it aids in effective air quality management by identifying sources of pollution and implementing targeted control measures, allows health professionals to assess potential health risks associated with pollutants emitted from vehicles, enabling the development of appropriate mitigation strategies. Additionally, predicting emissions helps environmental scientists evaluate their impact on ecosystems and biodiversity, contributing to conservation efforts, while also facilitating regulatory compliance by assisting vehicle manufacturers and regulatory agencies in meeting emission standards and developing more efficient emission control technologies. Lastly, predicting emissions is crucial for addressing climate change, allowing policymakers to assess contributions to global warming and develop strategies for mitigation.

In the literature, Chadha et al. [6] underscore the significant contribution of the transportation sector to global CO_2 emissions, amounting to approximately 16.2% of the total. Their study explores the prediction of CO_2 emissions by vehicles using various machine learning (ML) techniques. Through methods such as Lasso Regression, Multiple Linear Regression, XGBoost, Support Vector Regressor (SVR), Random Forest, and Ridge Regression, they achieve promising results with an RMSLE of 0.71 and an accuracy of around 99.8%. This highlights the potential of ML approaches in aiding local authorities in planning effective public transportation infrastructure to mitigate CO_2 emissions. The study conducted by Xu, Kang, and Lv [7] proposes a three-layer artificial neural network model for predicting vehicle exhaust emissions based on remote sensing data. Their approach employs an adaptive lasso algorithm to identify principal factors and establishes the Backpropagation neural network model as the optimal method. Their research aims to reduce inspection costs and establish a prediction model for total pollutant discharge, thereby supporting motor vehicle pollution regulation. Azeez et al. [8] present a hybrid model for predicting vehicular carbon monoxide (CO) emissions in urban areas of Kuala Lumpur, Malaysia. Their model combines correlation-based feature selection (CFS), support vector regression (SVR), and Geographic Information Systems (GIS). Through CFS, they identify seven road traffic CO predictors, and SVR is utilized for emission prediction. The model achieves impressive validation accuracy, correlation coefficient, mean absolute error, and root mean square error, highlighting its effectiveness in assessing traffic-related CO emissions on roads. Singh and Dubey [9] propose a deep-learning model using vehicle telematics sensor data to predict CO_2 emissions. With climate change a significant concern, their scalable model utilizes real-time vehicle sensor data and a Recurrent Neural Network (RNN)-based Long Short-Term Memory (LSTM) model to estimate CO_2 emissions. The system, utilizing On-Board Diagnostics (OBD-II) port data, offers an efficient approach to monitor emissions at the vehicular level, facilitating easy transmission of data to the cloud for analysis. Shobana Bai [10] explores ways to decrease emissions in a low-carbon biofuel-hydrogen dual-fuel engine. They test lemon peel oil, camphor oil, hydrogen induction, and a zeolite-based after-treatment system. Machine learning methods like XGBoost, LGBM, CatBoost, and Random Forest are used to predict engine emissions and performance, with CatBoost showing high accuracy. This research highlights the potential of machine learning in predicting emissions and improving engine efficiency in low-carbon fuel systems. In their study, Hananto et al. [11] explore ways to decrease emissions in a low-carbon biofuel-hydrogen dual-fuel engine. They test lemon peel oil, camphor oil, hydrogen induction, and a zeolite-based after-treatment system. Machine learning methods like XGBoost, LGBM, CatBoost, and Random Forest are used to predict engine emissions and performance, with CatBoost showing high accuracy. This research highlights the potential of machine learning in predicting emissions and improving engine efficiency in low-carbon fuel systems. Ramalingam et al. [12] employ artificial neural network (ANN) modeling to predict the behavior of a non-modified diesel engine fueled by blends of two low viscous biofuels. They find that the B20 blend exhibits improved efficiency, while the B50 blend shows minimal emissions compared to other blends. The trained ANN models demonstrate high accuracy, emphasizing the potential of the B20 blend as an effective alternative fuel for diesel engines.

The aim of this study is to systematically optimize the architecture of the artificial neural network (ANN) and employ data normalization techniques to enhance the accuracy and reliability of predictions for engine emissions. The originality of this study lies in its innovative application of machine learning techniques to predict engine emissions using eco-friendly fuels. While previous research has explored various methods for emissions prediction, this study uniquely focuses on ANN to accurately forecast emissions across different fuel compositions and engine speeds. Integrating ANN-based prediction models into emission control strategies can facilitate the development of cleaner and more sustainable transportation systems, thereby reducing the adverse impacts of emissions on the environment and public health.

2. Material and Methods

2.1. Artificial Neural Networks (ANN)

This section presents the methodology employed in utilizing Artificial Neural Networks (ANN) to predict engine emissions associated with eco-friendly fuels. The Artificial Neural Network (ANN) is a computational model inspired by the human 'brain's information processing mechanism. Structurally, it comprises distinct layers: an input layer, an output layer, and

several hidden layers. Each layer consists of neurons, the fundamental processing units interconnected through communication links governed by connection weights. Signal transmission within the network occurs via these weighted connections [13], [14]. The structure of a neuron and Artificial Neural Network (ANN) is depicted in Figure 1. This illustration highlights the organizational framework of a neuron, along with the interconnected layers and nodes within an ANN.

Development of the ANN model involves two pivotal stages: training/learning and testing/verification. The network is enhanced during training to produce output estimations based on input data. Subsequently, experimental data is compared with predicted outcomes in the testing phase, and training terminates once the test error meets the specified tolerance level. The ""backpropagation algorithm"" (BPA) is the predominant technique in ANN model development. It operates in two phases: forward feed and feedback. In the forward feed phase, information flows from the input layer to the output layer. During the feedback phase, the discrepancy between the achieved output and the target output is evaluated, and this discrepancy is subsequently utilized to update the connection weights, thus improving the model [15], [16]. During the development of the ANN model, 'it's common practice to use metrics like Mean Squared Error (MSE) and coefficient of determination (R^2) to measure the 'model's ability to predict outputs accurately.

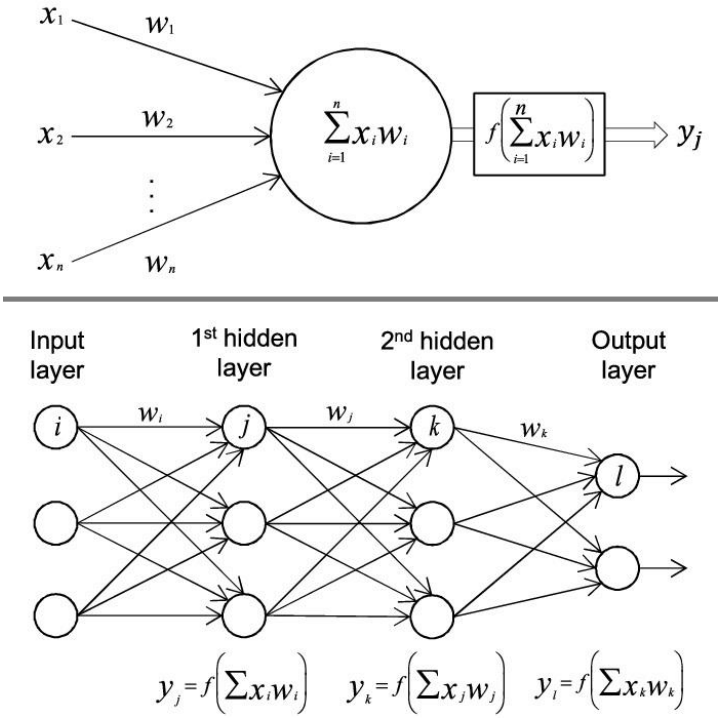


Figure 1 Structure of a Neuron and ANN [17]

Artificial neural network modeling involves several key steps, each crucial for developing an effective model. Firstly, data collection, where relevant data is gathered to train and test the neural network, is essential. This data should accurately represent the problem being addressed, such as historical stock prices for stock prediction tasks. Following data collection, the next step is data preparation, involving tasks like cleaning, outlier removal, and normalization to ensure the data is suitable for use by the neural network. Once the data is prepared, the appropriate neural network architecture is chosen based on the specific requirements of the problem. With the architecture selected, the network is trained using the prepared data, allowing it to learn and recognize patterns. Subsequently, the trained network is tested using a separate dataset to evaluate its accuracy and generalization capability. Finally, upon satisfactory testing results, the neural network can be deployed for real-world use, enabling it to make predictions or decisions in production environments. Each step in this methodology is essential for developing a robust and reliable artificial neural network model [18], [19].

2.2. Dataset

The dataset utilized in this study was acquired through a series of experiments to determine engine emissions. These experiments were conducted under varying conditions, including adjustments to fuel compositions (gasoline and ethanol ratios) and engine speeds (rpm). The primary objective was to develop an ANN model to predict emissions such as NOx (ppm), HC (ppm), and CO (%) levels. The emission concentrations were derived from the experimental data collected during these trials, resulting in a dataset comprising 90 observations. Statistical summaries of this dataset are presented in Table 1. In the model, the input variables include fuel ratio (gasoline and ethanol) and engine speeds, while the output variables consist of NOx, HC, and CO emissions. The dataset reflects a comprehensive evaluation of engine emissions under varying fuel ratios and engine speeds, providing valuable insights into the 'engine's behavior and environmental impact. Gasoline was predominantly used as fuel, with occasional ethanol additives. Engine speeds varied from 1400 to 3400 rpm, indicating

experimentation across different operating conditions. Nitrogen oxide (NOx) emissions ranged from 1328 to 2330 ppm, while hydrocarbon (HC) emissions varied from 176 to 295 ppm. Carbon monoxide (CO) emissions ranged from 1.12% to 1.68%.

Table 1 Statistical information related to the dataset utilized in this study.

	Features		Unit	Minimum	Maximum	Average
Input	Fuel Ratio	Gasoline	-	0.8	1	0.9
		Ethanol	-	0	0.2	0.1
	Engine Speeds		rpm	1400	3400	2400
Output	NOx		ppm	1328	2330	1877
	HC		ppm	176	295	232
	CO		%	1.12	1.68	1.36

2.3. Data Normalization and Model Performance Evaluation

Data normalization is a critical process in preparing datasets for analysis, particularly in machine learning. This technique involves adjusting the scale of data values to bring them within a standardized range. By doing so, we can mitigate the influence of outliers and ensure that different features contribute equally to the analysis. Data normalization is especially crucial for algorithms like artificial neural networks (ANNs), where consistent input ranges can significantly improve model performance. Various methods, such as min-max scaling or z-score normalization, are employed depending on the specific requirements of the dataset and the algorithm being used. In this study, we employ the min-max normalization technique, as outlined in Equation 1 [20].

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}(u - l) + l \quad (1)$$

where:

- x represents the original data.
- x' represents the normalized data.
- x_{max}, x_{min} refer to the maximum and minimum values of the original data vector.
- u, l represent the upper and lower bounds of the new range for the normalized data.

The performance of ANN models has been evaluated based on the utilization of two metrics. Mean Squared Error (MSE) is commonly employed due to its simplicity and effectiveness in measuring the squared difference between actual and predicted values. This metric provides insight into the 'model's overall accuracy by quantifying the average squared distance between predicted and actual values. Utilizing squared differences helps prevent the cancellation of negative terms, contributing to the robustness of MSE as a performance metric. On the other hand, R Squared (R^2) serves as a metric to assess the 'model's performance relative to a baseline model. Unlike MSE, which depends on the context, the R^2 score offers a standardized measure of goodness of fit independent of the specific problem context. It provides a means to compare the performance of the regression model against a simple baseline model, typically represented by the mean line. R^2, also known as the Coefficient of Determination or Goodness of Fit, quantifies how much better the regression line fits the data than a mean line, thereby providing valuable insights into the overall explanatory power of the model. The equations for R^2 and MSE are provided below:

R Squared (R^2) [21]:

$$R^2 = 1 - \frac{x_{res}}{x_{tot}} \quad (2)$$

- x_{res} represents the sum of squared residuals (the squared differences between actual and predicted values).
- x_{tot} represents the total sum of squares (the squared differences between each data point and the mean of the dependent variable).

Mean Squared Error (MSE) [21]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_{act} - x_{pre})^2 \quad (3)$$

Where:

- n is the number of data points.
- x_{act} is the actual value of the dependent variable for the i_{th} data point.
- x_{pre} is the predicted value of the dependent variable for the i_{th} data point.

3. Results and Discussion

In this study, we acquired the dataset through experiments aimed at assessing engine emissions. The dataset consists of three independent variables and three dependent variables. A comprehensive overview of the dataset is presented in Table 1, containing a total of 90 row data records. Subsequently, the dataset was normalized using Equation 1, following which it was randomly partitioned into training (75%, 67 data), validation (15%, 14 data), and testing (10%, 9 data) subsets.

To determine the optimal configuration of the neural network architecture, a single-hidden-layer artificial neural network (ANN) was employed. The number of neurons within the hidden layer was systematically varied from 10 to 50 in increments of 5. Training of the network was performed utilizing the Scaled Conjugate Gradient algorithm, with the sigmoid activation function implemented in the hidden layer and the pure-line function employed in the output layer.

During the training phase, engine parameters were utilized as inputs, while corresponding emission levels served as outputs. Through iterative backpropagation, the network iteratively adjusted its internal weights and biases to minimize the discrepancy between predicted and actual emission values. At the end of the training process, the neural network demonstrated proficient predictive capabilities for unseen input data. Evaluation of the 'network's performance, conducted using the mean square error (MSE) index, revealed that the optimal number of neurons in the hidden layer for predicting engine performance was 50, as delineated in Figure 2.

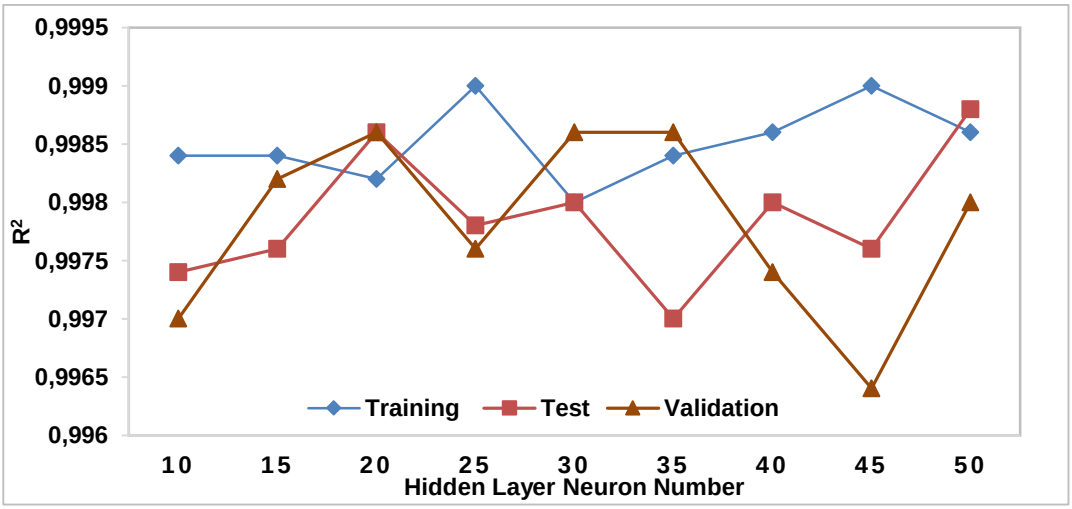


Figure 2 Determination of Neuron Number at the Hidden Layer

Figure 3 depicts an Artificial Neural Network (ANN) architecture comprising three input and output variables. The network includes a hidden layer containing 50 neurons. The schematic representation of the proposed ANN model is presented in Figure 4.

Figure 5 depicts the mean square error (MSE) values corresponding to varying numbers of hidden neurons within an artificial neural network (ANN). These MSE values are computed across training, testing, and validation datasets, offering insights into the ANN's performance across different hidden neuron configurations. This underscores the importance of selecting an optimal number of hidden neurons to attain superior model performance and generalization capability. According to Figure 5, the most suitable number of neurons in the hidden layer appears to be 50, exhibiting the lowest MSE values across all three datasets.

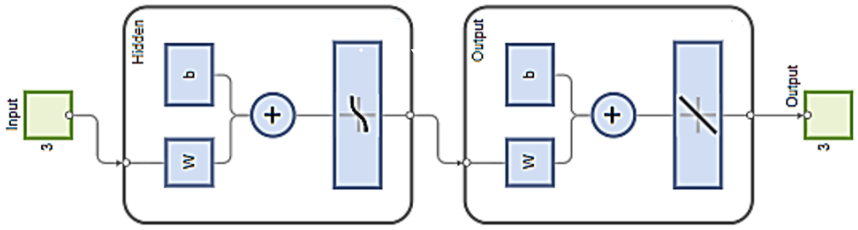


Figure 3 ANN Architecture Utilized for Prediction of the Effective Efficiency of the Engine

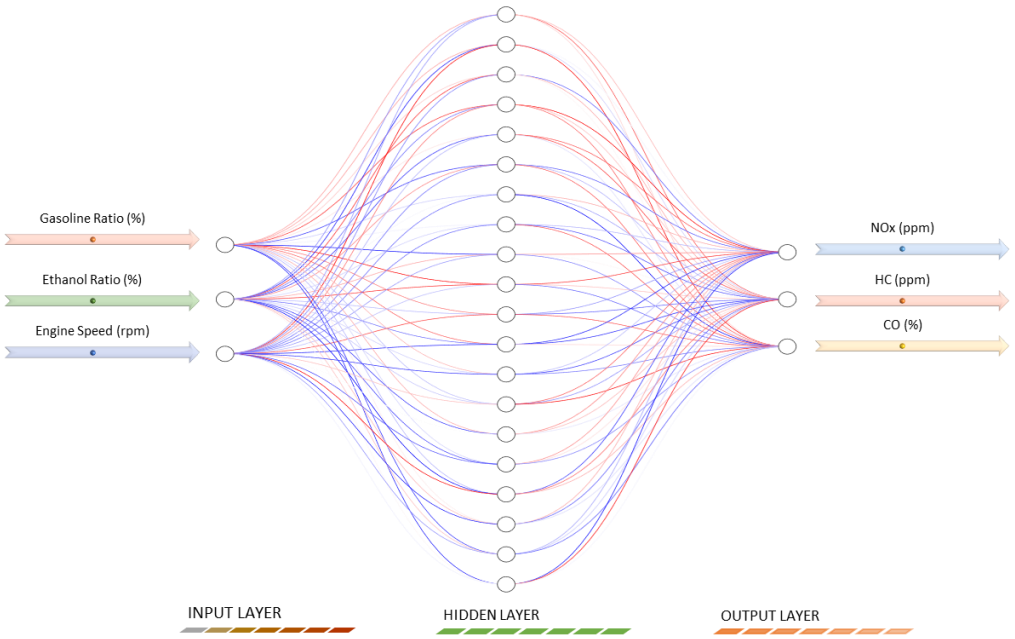


Figure 4 The schematic representation of the proposed ANN model.

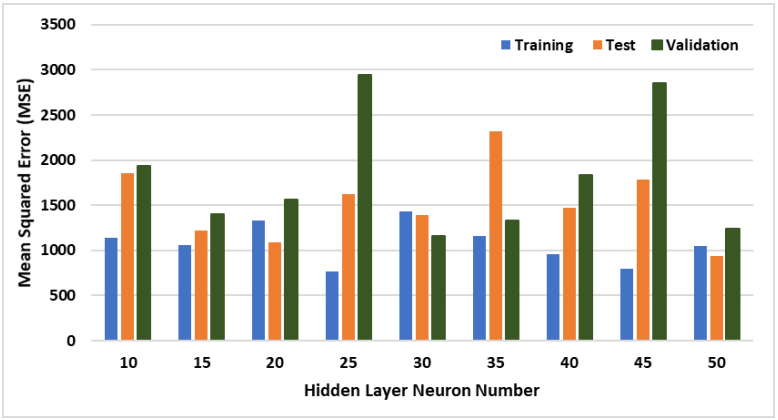


Figure 5 The Mean Squared Error (MSE) Values of the Neural Network

Figure 6 displays a scatter plot comparing predicted data against observed data. This visualization provides insights into the relationship between the predicted values generated by the model and the actual observed values in the dataset. The high R values observed for the training (0.99928), test (0.99937), and validation (0.99904) datasets indicate strong correlations between predicted and observed values, showcasing the 'model's accuracy in predicting engine emissions. When considering all datasets combined, the overall R-value of 0.99927 further reinforces the 'model's effectiveness in accurately capturing the underlying patterns in the data. This suggests that the 'model's predictions closely align with the observed data across various scenarios, highlighting its reliability and robustness in predicting engine emissions.

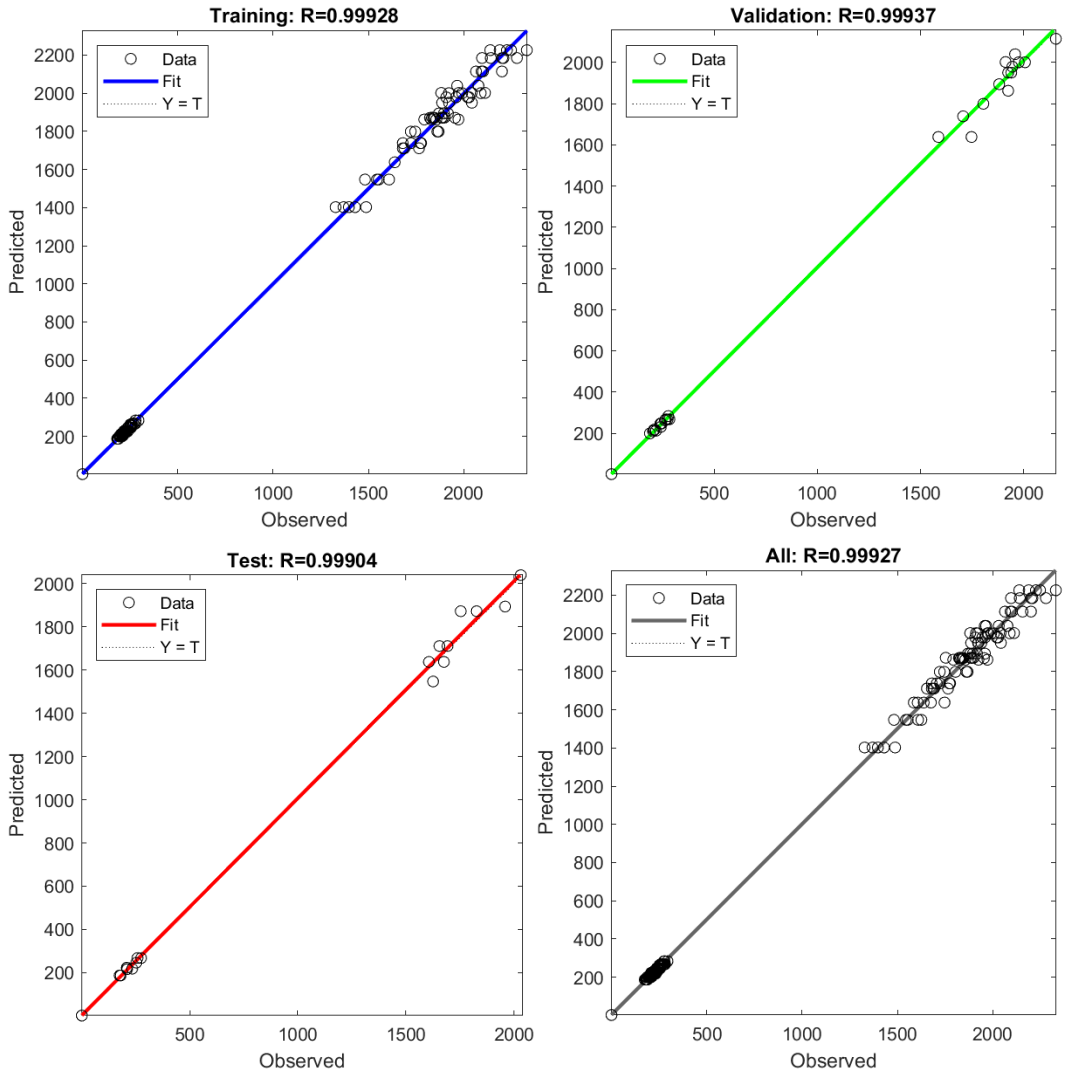


Figure 6 Scatter Plot Representing the Relationship Between Predicted and Observed Data Points.

In our study, we have evaluated the performance of a machine learning model in predicting engine emissions for eco-friendly fuels. The results presented in the Results section showcase strong correlations between predicted and observed emissions values, with high R values across different datasets indicating the accuracy and reliability of our model. This aligns with similar studies conducted in the literature, such as Chadha et al. [6], Xu, Kang, and Lv [7], Azeez et al. [8], Singh and Dubey [9], Shobana Bai [10], and Hananto et al. [11], which also employ various predictive modeling techniques to address emissions prediction in the transportation sector. However, our study stands out due to its focus on predicting emissions specifically for eco-friendly fuels and its utilization of a single-hidden-layer artificial neural network (ANN) architecture optimized through data normalization techniques. This unique approach contributes to the advancement of sustainable transportation solutions by providing accurate predictions for engine emissions under varying fuel compositions and engine speeds.

4.Conclusions

This study employed machine learning techniques to predict engine emissions for eco-friendly fuels. The dataset utilized in this research, acquired through experimental trials, provided valuable insights into engine emissions under varying fuel compositions and engine speeds. The dataset was prepared for analysis through data normalization techniques, such as min-max scaling, facilitating the training of a single-hidden-layer artificial neural network (ANN). The ANN architecture was optimized to determine the optimal number of neurons in the hidden layer. The results indicated that 50 neurons yielded the lowest mean square error (MSE) values across all datasets. Additionally, the scatter plot visualization demonstrated strong correlations between predicted and observed emissions values, further validating the effectiveness of the ANN model. Overall, the findings underscore the potential of machine learning approaches in predicting engine emissions and guiding the development of eco-friendly fuels, contributing to advancing our understanding of sustainable transportation solutions. In

future studies, it would be valuable to explore the application of more advanced machine learning techniques or hybrid models for predicting engine emissions with eco-friendly fuels. Additionally, investigating the impact of other factors, such as ambient temperature, humidity, and driving conditions on emission levels could enhance the predictive accuracy of the models. Furthermore, conducting field experiments or real-world validations to assess the performance of the developed models under diverse operating conditions and vehicle types would provide valuable insights for practical applications. This study has practical implications for multiple fields involved in environmental sustainability and transportation management. By offering a reliable method for predicting engine emissions, the research enables policymakers to make informed decisions about emission control strategies and regulations. Furthermore, automotive manufacturers can control the findings to develop more efficient emission control technologies and create engines that are environmentally friendly. Urban planners and transportation authorities can also benefit by using the predictions to optimize public transportation routes and infrastructure, leading to reduced emissions from engines. Briefly, this research could significantly influence real-world efforts to reduce the environmental impact of transportation systems.

References

[1] H. Aydin and C. İlkiliç, 'Air pollution, pollutant emissions and harmfull 'effects', *J. Eng. Technol.*, vol. 1, no. 1, Art. no. 1, Dec. 2017.

[2] F. Kelen, 'Motorlu Taşıt emisyonlarının insan sağlığı ve çevre üzerine etkileri', *Üzüncü Il Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 19, no. 1–2, Art. no. 1–2, Nov. 2014.

[3] P. Gireesh Kumar, P. Lekhana, M. Tejaswi and S. Chandrakala, 'Effects of vehicular emissions on the urban environment- a state of the 'art', *Mater. Today Proc.*, vol. 45, pp. 6314–6320, Jan. 2021, doi: 10.1016/j.matpr.2020.10.739.

[4] E. Ogur and S. Kariuki, 'Effect of car emissions on human health and the 'environment', *Int. J. Appl. Eng. Res.*, vol. 9, pp. 11121–11128, Jan. 2014.

[5] K. A. Bello, O. Awogbemi and M. G. Kanakana-Katumba, 'Assessment of alternative fuels for sustainable road 'transportation', presented at the 2023 IEEE 11th International Conference on Smart Energy Grid Engineering, SEGE 2023, 2023, pp. 7–15. doi: 10.1109/SEGE59172.2023.10274583.

[6] A. S. Chadha, Y. Shinde, N. Sharma and P. K. De, 'Predicting CO2 Emissions by vehicles using machine 'learning', in *Data Management, Analytics and Innovation*, S. Goswami, I. S. Barara, A. Goje, C. Mohan, and A. M. Bruckstein, Eds., in Lecture Notes on Data Engineering and Communications Technologies. Singapore: Springer Nature, 2023, pp. 197–207. doi: 10.1007/978-981-19-2600-6_14.

[7] Z. Xu, Y. Kang, and W. Lv, 'Analysis and prediction of vehicle exhaust emission using 'ANN', Jul. 2017, pp. 4029–4033. doi: 10.23919/ChiCC.2017.8027988.

[8] O. S. Azeez, B. Pradhan, and H. Z. M. Shafri, 'Vehicular CO Emission prediction using support vector regression model and 'GIS', *Sustainability*, vol. 10, no. 10, Art. no. 10, Oct. 2018, doi: 10.3390/su10103434.

[9] M. Singh and R. K. Dubey, 'Deep Learning Model Based CO2 Emissions Prediction Using Vehicle Telematics Sensors 'Data', *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 768–777, Jan. 2023, doi: 10.1109/TIV.2021.3102400.

[10] F. J. J. Shobana Bai, 'A Machine learning approach for carbon di oxide and other emissions characteristics prediction in a low carbon biofuel-hydrogen dual fuel 'engine', *Fuel*, vol. 341, p. 127578, Jun. 2023, doi: 10.1016/j.fuel.2023.127578.

[11] A. L. Hananto *et al.*, 'Elman and cascade neural networks with conjugate gradient Polak-Ribière restarts to predict diesel engine performance and emissions fueled by butanol as sustainable 'biofuel', *Results Eng.*, vol. 19, p. 101334, Sep. 2023, doi: 10.1016/j.rineng.2023.101334.

[12] K. Ramalingam *et al.*, 'Forcasting of an ANN model for predicting behaviour of diesel engine energised by a combination of two low viscous 'biofuels', *Environ. Sci. Pollut. Res.*, vol. 27, no. 20, pp. 24702–24722, Jul. 2020, doi: 10.1007/s11356-019-06222-7.

[13] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, 'Evolutionary artificial neural networks: a 'review', *Artif. Intell. Rev.*, vol. 39, no. 3, pp. 251–260, Mar. 2013, doi: 10.1007/s10462-011-9270-6.

[14] V. Eyupoglu, B. Eren, and E. Dogan, 'Prediction of ionic cr (vi) extraction efficiency in flat sheet supported liquid membrane using Artificial Neural Networks (ANNs)', *Int. J. Environ. Res.*, vol. 4, no. 3, pp. 463–470, SUM 2010.

[15] Y. Wu and J. Feng, 'Development and Application of artificial neural 'network', *Wirel. Pers. Commun.*, vol. 102, no. 2, pp. 1645–1656, Sep. 2018, doi: 10.1007/s11277-017-5224-x.

[16] J. Zou, Y. Han, and S.-S. So, 'Overview of Artificial Neural 'Networks', in *Artificial Neural Networks: Methods and Applications*, D. J. Livingstone, Ed., in Methods in Molecular Biology™. , Totowa, NJ: Humana Press, 2009, pp. 14–22. doi: 10.1007/978-1-60327-101-1_2.

[17] S. Vieira, W. H. L. Pinaya and A. Mechelli, 'Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and 'applications', *Neurosci. Biobehav. Rev.*, vol. 74, no. Pt A, pp. 58–75, Mar. 2017, doi: 10.1016/j.neubiorev.2017.01.002.
[18] D. J. Livingstone, Ed., *Artificial Neural Networks*, vol. 458. in Methods in Molecular Biology™, vol. 458. Totowa, NJ: Humana Press, 2009. doi: 10.1007/978-1-60327-101-1.
[19] M. Zakaria, M. AL-Shebany, and S. Sarhan, 'Artificial Neural Network : A Brief 'Overview', vol. 4, no. 2, 2014.
[20] E. Dogan, A. Ates, E. C. Yilmaz and B. Eren, 'Application of artificial neural networks to estimate wastewater treatment plant inlet biochemical oxygen 'demand', *Environ. Prog.*, vol. 27, no. 4, pp. 439–446, Dec. 2008, doi: 10.1002/ep.10295.
[21] B. Eren, İ. Aksangür and C. Erden, 'Predicting next hour fine particulate matter (PM2.5) in the Istanbul Metropolitan City using deep learning algorithms with time windowing 'strategy', *Urban Clim.*, vol. 48, p. 101418, Mar. 2023, doi: 10.1016/j.uclim.2023.101418.

Conflict of Interest Notice
The authors declare that there is no conflict of interest regarding the publication of this paper**.**

Ethical Approval and Informed Consent
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 ™.

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 112-126
DOI: 10.35377/saucis...1414742
RESEARCH ARTICLE

Estimation Single Output with A Hybrid of ANFIS and MOPSO_HS

Aref Yelghi [1]

[1] Department of Computer Engineering, İstanbul Topkapı University, Istanbul, Kazlıçeşme/Zeytinburnu 34010, Türkiye

Corresponding author:
Aref Yelghi, Department of Computer Engineering, İstanbul Topkapı University
arefyelghi@topkapi.edu.tr

Article History:
Received: 04.01.2024
Accepted: 29.04.2024
Published Online: 29.04.2024

ABSTRACT
Adaptive Neuro-Fuzzy Inference System (ANFIS) has gained popularity in recent years due to its predictive capabilities. Proper adjustment of ANFIS parameters is an optimization problem but integrating it with traditional optimization techniques has led to challenges such as local minima and slow convergence, resulting in obstacles to its prediction. Additionally, some researchers focusing on incorporating single-objective optimization often face issues with reliability and stability in parameter adjustment. This study, focused on multi-objective optimization, presents an algorithm that integrates ANFIS with MOPSO_HS. The proposed model, compared and applied to three real-world datasets, has demonstrated robustness in prediction problems. A comparative analysis is conducted between the proposed integrated model and well-known integrated algorithms with 20 runs. For further comparison, the Wilcoxon signed-rank test is used to determine whether there is a statistically significant difference in performance. The experimental results indicate the algorithm's accuracy, stability, and reliability in solving integration problems, highlighting its superiority over alternative approaches.

Keywords: Metaheuristic, Multi-Objective Optimization, ANFIS, Exchange Rate, Neuro Fuzzy, RMSE

1. Introduction

Numerous artificial intelligence (AI) techniques have been widely used in practical applications over the past few years. In the field of neuro-fuzzy techniques, the Adaptive Neuro-Fuzzy Inference System, also known as the Adaptive Network-Based Fuzzy Inference System (ANFIS) [1], has become more well-known. Fuzzy logic (FL) and artificial neural networks (ANN) are combined in ANFIS, which has applications in Data Science, Image processing, Finance Technology, traffic control studies, feature extraction, estimate, prediction, and more [2]. Fuzzy logic, introduced by Zadeh [3], defines membership between 0 and 1, while ANN models certain functions of human brain neurons. In ANFIS, the premise and consequence layers play pivotal roles in the network's training process. The setting of ANFIS parameters involves the use of optimization algorithms.

The original ANFIS, proposed by Jang [1], employed hybrid learning using the gradient descent (GD) algorithm for antecedent parameters and the Least Squares Error (LSE) algorithm for consequent parameters. These classical optimizations were applied to the set of ANFIS parameters. However, due to GD and LSE's tendency to get trapped in local minima, researchers turned to metaheuristic algorithms, which explore the global minimum effectively. In the metaheuristic optimization field, two types of algorithms exist: based on the derivative (gradient descent) and not based on the derivative (metaheuristic and heuristic) algorithms. While derivative-based algorithms work only on differentiable functions, ANFIS parameters can be transformed into non-differentiable functions. Hence, in this study, metaheuristic methods can solve ANFIS parameters as non-differential functions.

Fewer studies have investigated multi-objective algorithms for integration, even though some have combined ANFIS with single-objective metaheuristic optimization techniques. This paper focuses on using multi-objective algorithms to carry out tune ANFIS configurations. Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Genetic Algorithms (GA), Simulated Annealing (SA), and Hybrid Approaches that combine ANFIS with various algorithms are some of the strategies that have been investigated in this field.

The objective in this endeavor is to develop a prediction system for complicated data by emphasizing on multi-objective

optimization and ANFIS. Several well-known multi-objective optimization algorithms include Non-dominated Sorting Genetic Algorithm (NSGA) [19], Non-dominated Sorting Genetic Algorithm II (NSGAII) [20], Strength–Pareto Evolutionary Algorithm 2 (SPEA2) [21], Multi-Objective Particle Swarm Optimization (MOPSO) [22], Multi-Objective Artificial Bee Colony (MOABC) [23] and Multi-Objective Grey Wolf Optimizer [24]. When combined with Adaptive Neuro-Fuzzy Inference Systems (ANFIS), these techniques greatly aid in solving practical problems and give results that are highly precise.

Hybrid ANFIS techniques have been increasingly applied to real-life scenarios in recent years, with definite advantages and disadvantages. This study aims to find out how adjusting operational factors, such as engine load and syngas composition, can improve the efficiency of a dual-fuel syngas/diesel engine while reducing pollution emissions. Using a hybrid technique of ANFIS and response surface methodology (RSM), the research simulates engine performance under different syngas compositions and compares the predicting capacities of ANFIS and RSM [25].

The other study used genetic algorithms (GA) to optimize multi-objective age-hardening process parameters while leveraging the improved performance of artificial neural networks (ANN) beyond experimental points. This demonstrated the effectiveness of ANNs [26]. The objective of other work is to precisely calculate the recompression coefficient (Cr) for over-consolidated soil using a hybrid ANFIS-PSO Machine Learning (ML) model. The model compares favorably to benchmark models of single ANFIS and Support Vector Machines (SVM) using PSO and ANFIS techniques [27].

The hybrid ANFIS-GA-PSO model, along with an extreme learning machine (ELM), is used in the research for predictive analysis. The results show that the ANFIS-GA model performs better than ELM in analyzing shear behavior. The comparison shows how well the hybrid model predicts shear strength and is analyzed using regression indices [28]. The objective of this research is to develop prediction models for a range of cut quality variables, including surface roughness, kerf taper, and material removal rate, during abrasive aqua jet cutting (AAJC) of natural fiber composite laminates. The models are created using an ANFIS and the Taguchi-genetic algorithm (TGA) [29].

In general, it has been shown in the studies that layers one and four of ANFIS have been looked at as a problem of setting parameters. So far, no study has been done on different measurements for each solution-providing parameter. In this study, with the efficiency of multi objective, it has been tried to use at least two different measurements or functions for layer one and four so that Anfis can show better accuracy. In line with the proposal, it has been tried to use the existing multi-objective algorithms. Finally, by combining the existing algorithms, efficiency, and improvement in this type of problem can be achieved.

The structure of this research paper is as follows: Section 2 provides a detailed overview of the ANFIS model. In contrast, Section 3 delves into multi-objective optimization and discusses the integration of PSO and HS algorithms for multi-objective optimization. Section 4 presents the experimental results and evaluations. Section 5 Application of Proposed Model is described, and Section 6 concludes the study, offering recommendations for further research.

2. ANFIS Tool

An artificial intelligence method called the adaptive network-based fuzzy inference system (ANFIS) blends fuzzy logic (FL) and artificial neural networks (ANN). There are five levels in the approach for each layer that provides the data set. Figure 1 is a representation of the ANFIS Framework.

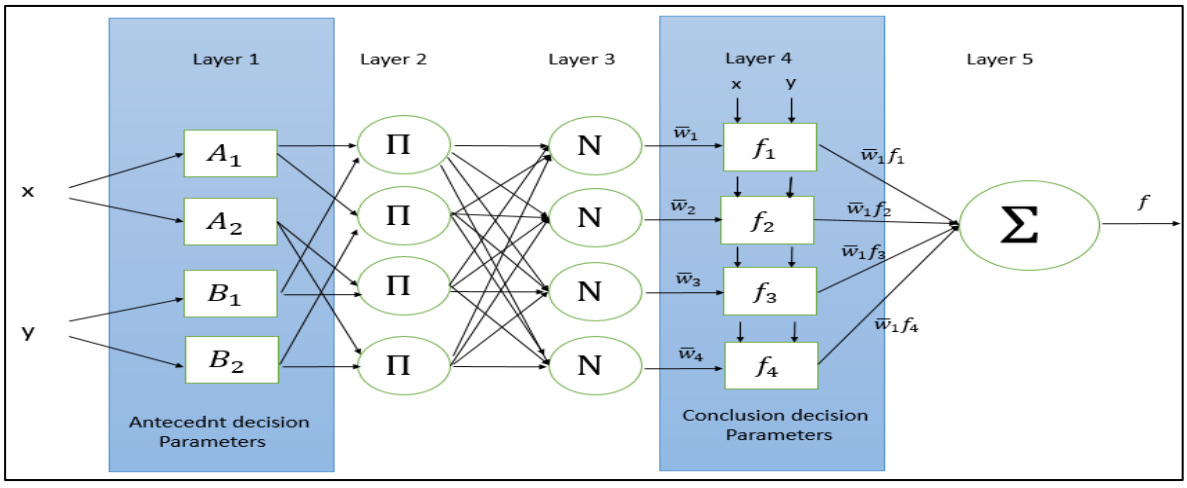


Fig 1. Example: an ANFIS Two-input Model

First Rule: İF x is A_1 and y B_1 then z is $f_1(x,y)$
Second Rule: İF x is A_2 and y B_2 then z is $f_1(x,y)$

x, y: Inputs for ANFIS
A, B: Fuzzy sets
z: $f_i(x,y)$ Outputs for Sugeno Fuzzy inference systems.

The first- and fourth-layer nodes represent parameters that have been optimized. The structure has been fixed at layers 2 and 3. The first layer has responsive nodes with the following characteristics:

$o_{1,i}$-$\mu_{A_x}(x)$ *for i=1,2* (1)
$o_{1,i}$-$\mu_{B_{i-2}}(x)$ *for i=3,4* (2)

$\mu(x)$, $\mu(y)$ are the membership functions, which can be defined as a bell shape. The formula is given as follows:

$$\mu(x) = \frac{1}{1+\left(\frac{x-c_i}{a_i}\right)^{2b_i}} \quad (3)$$

Or

$$\mu(x) = \exp\left\{-\left(\frac{x-c_i}{a_i}\right)^2\right\} \quad (4)$$

a_i, b_i, c_i Parameters are called premise parameters. They will be changed in the training step. The output of them illustrates the power of a rule. Every fixed node in the second layer has been multiplied with the signal input from the previous layer.

$$o_{2,i} = w_i = \mu_{A_i}(x).\mu_{B_i}(y) \text{ for i=1,2} \quad (5)$$

Every node in the third layer normalizes the input data by taking into account all relevant criteria. The following formula is provided:

$$o_{3,i} = \overline{w_i} = \frac{w_i}{\sum w_i} = \frac{w_i}{w_1+w_2} \text{ for i=1,2} \quad (6)$$

Every node in the fourth layer is an adaptive layer, which has the following definition and description:

$$o_{4,i} = \overline{w_i}.f_i \text{ for i=1,2} \quad (7)$$

First Rule: if x is A_1 and y is B_1 then$f_1=p_1x+q_1y+r_1$
Second Rule: if x is A_2 and y is B_2 then$f_2=p_2x+q_2y+r_2$

$p_i, q_i,$ and r_i Consequential parameters are parameters. The final layer assesses the output while it is running and computes the total output [17, 18].

$$o_{5,i} = f_{out} = \sum_i \overline{w_i}.f_i = \text{overall output} \quad (8)$$

3. Multi-Objective Optimization

In mathematical terms, a multi-objective optimization problem can be formulated as follows:

$$\begin{aligned} &Min/Max\ f_m(x), && m=1,2,\ldots,M \\ &Subject\ to\ g_i(x) \geq 0, && j=1,2,\ldots,J \\ &h_k(x) = 0, && k=1,2,\ldots,K \\ &x_i^{(L)} \leq x_i \leq x_i^{(L)}, && i=1,2,\ldots,n \end{aligned} \quad (9)$$

The optimal solution in the single-objective optimization problem is the first or last of the sorted solutions. The comparison of solutions is based on the sorting. Conversely, in the case of a multi-objective optimization issue, a solution's superiority can be determined by its domination over many values of the optimal solution.

3.1. Definition of Dominance

A solution is considered dominant if it does not perform worse than any of the objective values, and the solution on the opposing side performs better than the other in at least one of the objective values. Refer to Figure 2. A set of solutions is called a non-dominated solution set if it contains all of the solutions that are not dominated by any other feasible solution. This is how the term "Pareto front" is defined. (see Fig 2)

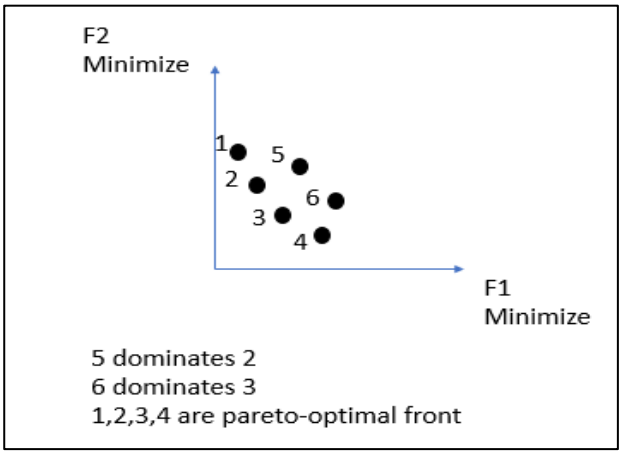


Fig 2. The Concept of Domination and Non-Pareto Front.

3.2. Multi-Objective Particle Swarm Optimization (MOPSO)

Swarm intelligence is used by MOPSO, in which a collection of particles works together to search the solution space. According to their previous best positions and the best positions discovered by their neighbors, each particle represents a potential solution to the optimization issue, and their positions are updated all over iterations. MOPSO aims to provide a set of solutions that gives decision-makers a variety of trade-offs to choose from when dealing with conflicting objectives by combining Pareto dominance with diversity-preserving processes. A sequential algorithm's general form is provided below.

1- Initialization: The population of particles should be initialized with random velocities (v_i) and coordinates (x_i). Decide on your own best positions.P_{best_i} of every particle to its starting location.

2- Objective Evaluation: Evaluate each particle's objective values:$F = \{f_1, f_2, \dots, f_k\}$

3- Pareto Dominance: Pareto dominance is used to compare particles. We have discussed before.

4- Update Individual Best and Archive:

5- Position and Velocity Updates:

Utilizing the following formulas, adjust each particle's position and velocity:V_i(t+1)=w·V_i(t)+ c_1·r_1·(P_{best_i}−x_i(t))+ c_2.r_2·(P_g −x_i (t)) and x_i(t+1)= x_i(t)+ V_i (t+1)

6- Mechanisms of Convergence and Diversity:

Use techniques to promote the dispersion of solutions along the Pareto front, such as crowding distance calculation, in order to achieve a balance between convergence and diversity.

7- Termination Standards:

Search for conditions that indicate when the process should end, like completing a certain number of iterations or approximating the Pareto front to a reasonable degree.

8- Goal:

The collection of non-dominated solutions that indicate the Pareto front in the external archive is the final outcome.

Wherever,
w is the weight of inertia.
The coefficients of acceleration are c_1 and c_2
There are two random variables, $r_1$and r_2 , in [0,1]
The position with the best global ranking among the non-dominated solutions in the external archive is P_g.

3.3 Multi-Objective Particle Swarm Optimization and Harmony Search (MOPSO_HS)

The proposed algorithm combines multi-objective particle swarm optimization (MOPSO) and harmony search (HS), offering flexibility and stability while seeking the global minimum. It seamlessly adapts to the ANFIS model and operates as follows: The main body of algorithms is the probability operators and mutation operators, which are complementary to each other. The Poisson cumulative distribution, the Gaussian distribution, and the mutation operator all work together to keep the balance between exploration and exploitation in our algorithm. They let the suggested algorithm look for new solutions in a

way that is based on probability, get scape of around local optima, and add variety solution, which guarantees a thorough and successful search for the global minimum.

Poisson Cumulative Distribution: The Poisson cumulative distribution represents the probability that several events will take place within a specific window of time or space. About the proposed algorithm: The number of separate occurrences (represented by the variable x) is given. When calculating the Poisson cumulative distribution function, the parameter pm is employed. In point of view of Application for proposed Algorithm: Using the Poisson cumulative distribution function, randomization is added while still following a structured methodology. It enables the algorithm to balance exploitation and exploration in the search space by allowing it to investigate novel solutions probabilistically. The algorithm can take into account different probabilities when developing new solutions because of the cumulative structure of the distribution.

Gaussian Distribution: A continuous probability distribution that is symmetric about its mean and resembles a bell curve is the Gaussian distribution, commonly referred to as the normal distribution. R stands for a random variable that was utilized to choose a particular dimension.

The fret width (FW) is a parameter that affects how widely spaced out the Gaussian distribution is. Application in Algorithm: In order to add randomness to the algorithm's progress through the search space, the Gaussian distribution is used. The program searches a larger search space by producing arbitrary numbers using a Gaussian distribution and exploring the region surrounding the current solutions. This exploration technique helps the algorithm achieve global optimization by assisting in escaping local optima.

Evolutionary algorithms need to include the mutation operator. By adjusting some of the search space parameters used for determining the solutions, it provides genetic diversity to the existing population. By altering the existing solutions, the mutation operator assists in introducing novel solutions and promotes the algorithm to proceed toward uncharted areas of the search space.

MOPSO_HS Algorithm

1. **BEGIN**
2. **Initialize** swarm Positions, Velocities and evaluate fitness value
3. FW=0.02*(VarMax-VarMin); /*Fret Width (Bandwidth)*/
4. MaxIt=100; /*Max generation*/
5. It /*generation number
6. **WHILE** (**Check** Terminate or Maximum Number of Generation is reached)
7. **BEGIN**
    a. **Select** Leader by RouletteWheelSelection method.
    b. **Update** Positions and Velocities.
    c. **Rand** (par);
    d. **If** par>0.7 Then Apply Mutation by Formula pm=(1-(it-1)/(MaxIt-1))^(1/2)
    e. **Else If** ((par<=0.7) && (par>0.3)) Then /*Gaussian*/
        i. R=Random [1..5]
        ii. Delta=FW*randn();
        iii. NewSol.Position (R)=OldSol.Position(R)+ Delta
    f. **Else** /*Poisson cumulative distribution*/
        i. R=Random [1..5]
        ii. x = 0:4;
        iii. pm=(1-(it-1)/(MaxIt-1))^(1/2);
        iv. y = poisscdf(x,pm);
        v. Delta=FW*y;
        vi. NewSol.Position(R)=OldSol.Position(R)+ Delta
    g. **Update** Repository by truncating its member.
    h. **Update** Grid
    i. **Check** if the Repository is Full Remove the bad solution
8. **END**
9. **END**

Fig 3. MOPSO_HS Algorithm for Global Optimization

As viewed in Figure 3, which includes three sections. Roulette Wheel Selection, a fundamental genetic algorithmic technique, is used to choose a leader at the beginning of this repetitive algorithmic process. It then modifies particle placements and velocities according to selected leaders and other variables, allowing efficient search space exploration. To decide whether a mutation should be applied, a random value par between 0 and 1 is generated. If par is greater than 0.7, a decreasing factor is used to carry out the mutation, providing a balanced exploration-exploitation strategy (Figure 3-d). When 0.3<=par<=0.7,

the specified dimension's particle positions are perturbed using a Gaussian distribution (Figure 3-e).
Alternatively, the Poisson cumulative distribution is used for position perturbation if par<=0.3 (Figure 3-f). By updating the repository and grid structures, truncating inappropriate solutions, and controlling solution variety, the algorithm preserves solution quality. It dynamically adjusts to the repository's capacity while ensuring the retention of top-notch solutions. The execution of the algorithm brings this complex process to a successful conclusion, delivering a flexible and efficient optimization framework.

The algorithm is integrated with ANFIS, and then the obtained general model is applied to the data set. The dataset is categorized into 7 systems, where each system includes inputs and targets that are fed to ANFIS. The data set related to the exchange rates in Turkish Lira and Dolar from 2007 to 2020. As you can see in Figure 4, the general system contains two components: ANFIS and the real system, which are fed to ANFIS. For setting the parameters of ANFIS, all parameters in each iteration map to a vector, individual, or particle in multi-objective optimization.

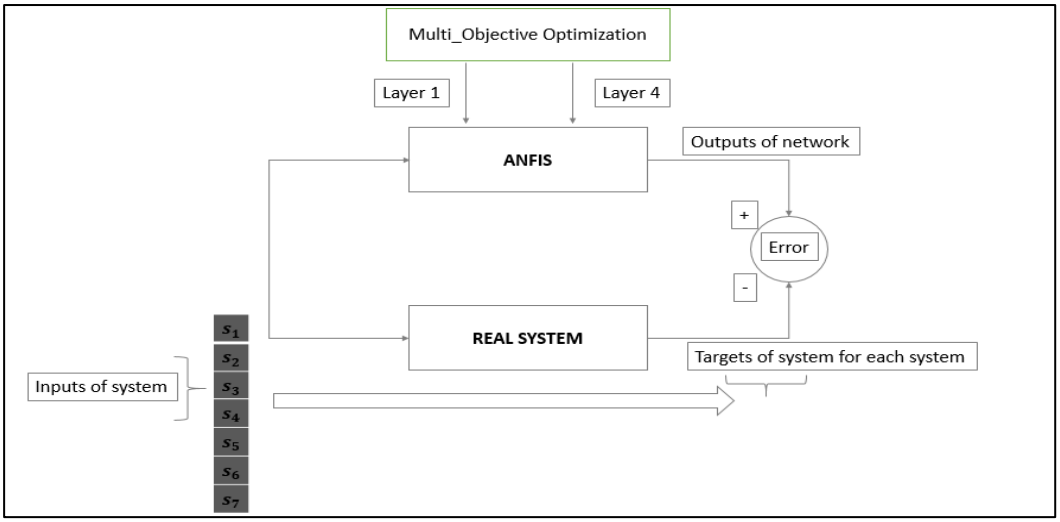


Fig 4. ANFIS and Dataset with Definition of 7 Systems

For integrating the proposed algorithm with ANFIS. Layers 1 and 4 of ANFIS are converted to one vector, which is the proposed algorithm that tries to find the best parameters for ANFIS. As you view Figure 5, parts (a) and (c) of a vector are antecedent decision parameters and conclusion decision parameters, respectively. Seven other systems also have different parameters.

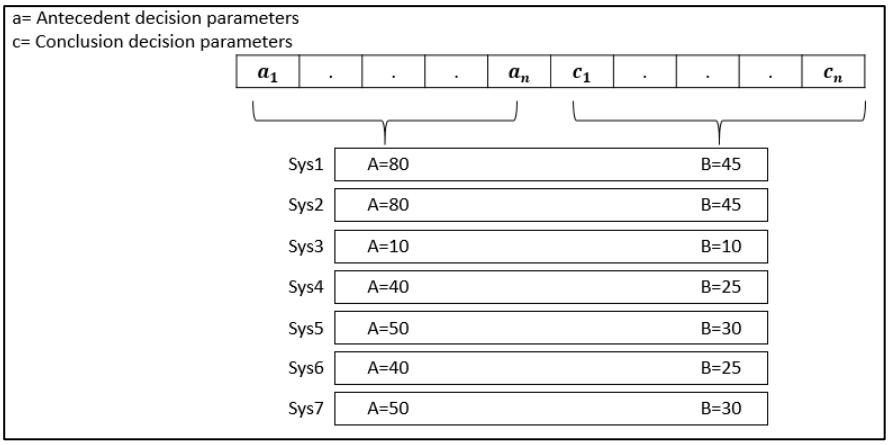


Fig 5. The Decision Variables or Vectors

Tables 1 and 2 are the definitions and descriptions of the components of systems. As you view seven systems generated by two inputs called buy and sell, the systems are generated based on the last few days for easy feed to the ANFIS. By taking this action, it will be clear and easy to extract a pattern from the complex dataset. The data set instance 151 has two inputs and is generated with a supposed target from one of the inputs.

Table 1. Definition of a Finance Problem Utilized in the Application

Name	Definition	Systems
f_1	Calculating the exchange rate between USD and YTL	$S_1..S_7$
x	Buy	x(t), time-based on the day
y	Sell	y(t), time-based on the day

Table 2. Applications Using Data Systems

Systems	Inputs of system	Target of system
S_1	x(t-1),x(t-2),x(t-3),x(t-4),x(t-5),x(t-6),x(t-7)	x(t+1)
S_2	y(t-1),y(t-2),y(t-3),y(t-4),y(t-5),y(t-6),y(t-7)	y(t+1)
S_3	x(t)	y(t)
S_4	x(t-1),x(t-2),x(t-4), x(t-6)	x(t+1)
S_5	x(t-1), x(t-3),x(t-5), x(t-7)	x(t+1)
S_6	y(t-1),y(t-2),y(t-4), y(t-6)	y(t+1)
S_7	y(t-1),y(t-3),y(t-5), y(t-7)	y(t+1)

3.4. Fitness Function

The calculation of Mean Squared Error (MSE) involves averaging the squared deviations between the values $\hat{Y}_i$ that were predicted and those Y_i that were observed. The MSE formula is:

$$\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(Y_i-\hat{Y}_i\right)^2 \quad (10)$$

The square root of the MSE is the RMSE. Because the error metric has the same units as the dependent variable, it is frequently employed to improve its readability.
The RMSE formula is:

$$\text{RMSE}=\sqrt{MSE} \quad (11)$$

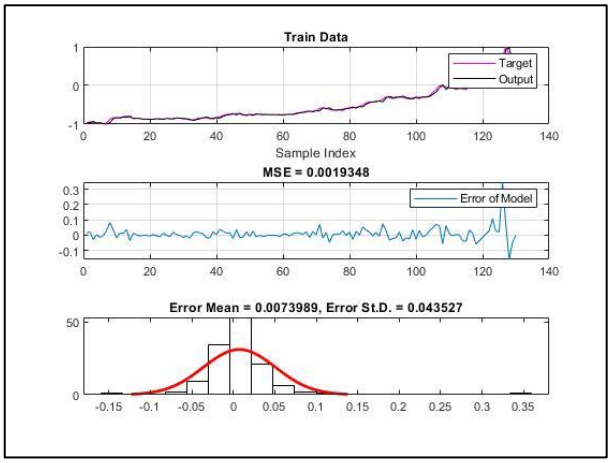
In this case, it can be said that the formulas are of the same type, and both of them are different interpretations of information. Here, the algorithms should provide the minimum value of each as a solution to a problem. The question is, if we consider these two together, the act of dominating does not make sense. By combining a small amount of imaginary noise, the problem can be transformed into a demonization. Finally, after successive execution of multi-objective algorithms, the Pareto front solution of the desired case can be obtained.

4. Result and Experiments

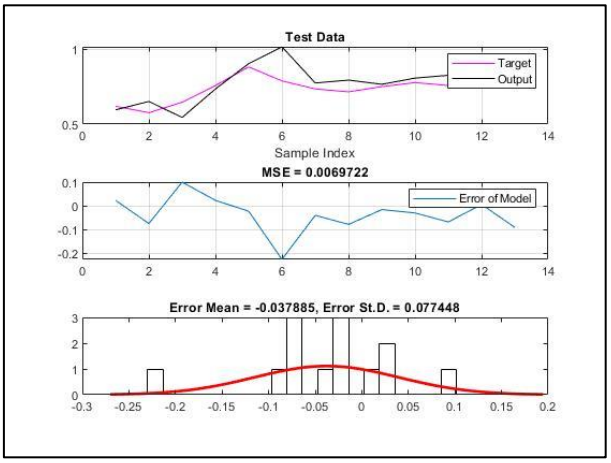
All the parameters of algorithms are tuned according to Table 3, and the other specified parameters are standard. For the input of ANFIS, Using the SUGENO technique, take five cluster inputs based on the Gaussian. Figure 6 shows one example of sys1, which is estimated using some integrated methods with ANFIS. The two basic steps of ANFIS are the train and test steps, with 138 and 13 instances of the dataset, respectively. As you observe each figure, you can find the training dataset, test dataset, mean squared error (MSE) of the target and output of the network for each instance, standard deviation, and error mean based on the bin for proper presentation. As a result, as shown in Table 4, the comparison of performance with 20 runs for the integrated ANFIS with MOPSO, NSGA2, SPEE2, MOGWO and MOPSO_HS. It is demonstrated that MOPSO_HS algorithms with specific features and complex systems outperform the others in sys 3,4,6, which is a significant feed that is odd or even intervals operate well. MOPSO in sys 1,2,7 indicates the best performance and have diversity in solving problem and it is not able to work in sequence interval. NSGA2 also only proposed the best result in sys 5. We try to prove the algorithm that provides the best results according to the given sequence based on Table 2.

Table 3. Parameters of the Metaheuristic Algorithms

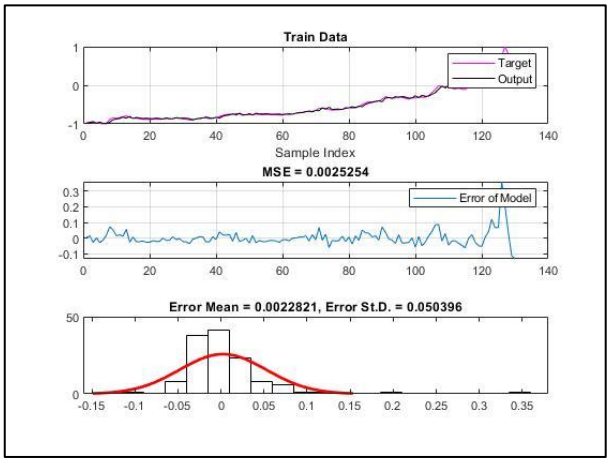
parameters	MOPSO	NSGA2	MOPSO_HS
Population size	25	25	25
iteration	100	100	100
Upper and Lower Bound	[0.1..1]	[0.1..1]	[0.1..1]



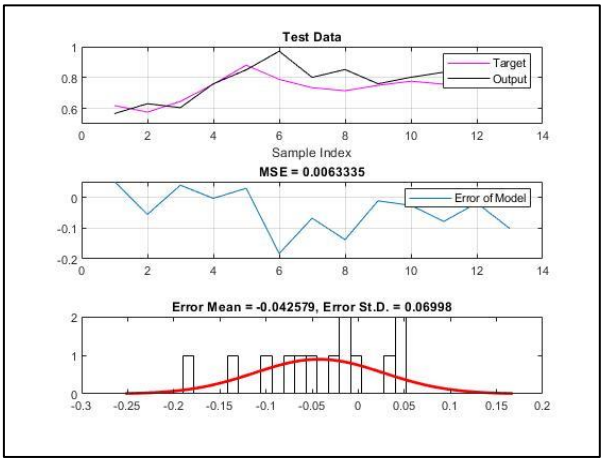
a) Train phase of the MOPSO algorithm for sys1



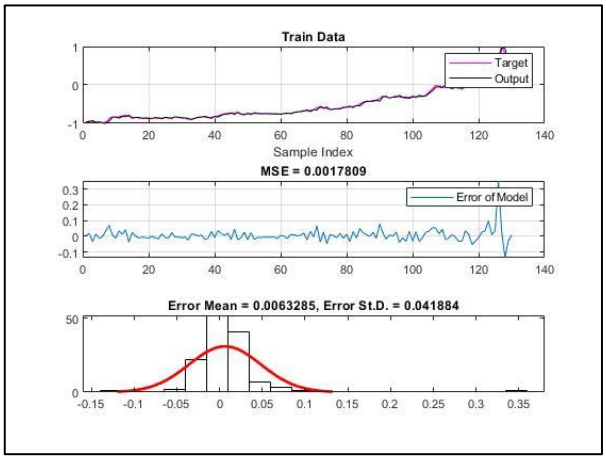
b) Test phase of the MOPSO algorithm for sys1



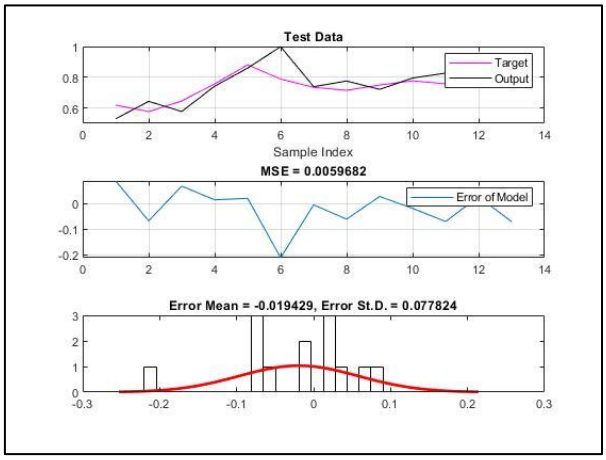
c) Train phase of the NSGA2 algorithm for sys1



d) Test phase of the NSGA2 algorithm for sys1



e) Train phase of the MOPSO_HS algorithm for sys1



f) Test phase of the MOPSO_HS algorithm for sys1

Figure 6. The Integrated Algorithms in Train and Test steps

One of the performance comparisons for each system using algorithms based on the training and testing steps' root-mean-square deviation (RMSE) appears in Table 5. We just give an example of the estimated Sys 3 using algorithms to demonstrate the performance of the models. As you view Table 6, the comparison between the target and the output of ANFIS_MOPSO_HS exhibits strong performance. To further prove the estimation model, we used one real dataset for traffic in the LTE network in 2018, which included 8735 training sets and 167 testing sets (Kaggle community). The traffic of data collected from the 4G cell for mobile phones adjacent to the cell is examined and predicted. Traffic is the total data capacity of all users within an hour that are served by an antenna cell [22]. Example: Cell 01000 is serving 60 users; each user uses an average of 10 Mb in 1 hour. Traffic in cell 01000. So, the traffic of this cell in hours x = 60 * 10 = 600 Mb. The data set

includes 57 cells, and it is gathered in approximately 1 year x 24 hours x 57 cells. In Table 7. As you can see, as in the previous case, three systems have been used for modeling. The results of our algorithm are obtained with less error than other algorithms.

Table 4. Evaluation of Multi-objective Metaheuristic Algorithms' Performances in ANFIS Comparison

Algorithms	Statistic Value	Sys1	Sys2	Sys3	Sys4	Sys5	Sys6	Sys7
MOPSO	Average	4.06E-02	4.18E-02	1.80E-03	4.89E-02	4.51E-02	4.85E-02	4.38E-02
	Minimum	3.09E-02	3.27E-02	1.76E-03	4.37E-02	3.35E-02	4.22E-02	3.17E-02
	Maximum	5.05E-02	5.05E-02	1.88E-03	5.16E-02	4.97E-02	5.24E-02	5.04E-02
	Variance	2.69E-05	1.65E-05	9.16E-10	6.42E-06	1.65E-05	1.15E-05	2.74E-05
NSGA2	Average	4.29E-02	4.35E-02	1.82E-03	4.95E-02	4.34E-02	5.01E-02	4.46E-02
	Minimum	3.30E-02	3.31E-02	1.76E-03	4.32E-02	3.30E-02	4.27E-02	3.25E-02
	Maximum	4.70E-02	5.10E-02	1.99E-03	5.22E-02	4.89E-02	5.46E-02	5.12E-02
	Variance	8.41E-06	1.25E-05	3.98E-09	6.93E-06	3.18E-05	9.99E-06	3.08E-05
SPEA2	Average	4.28E-02	4.20E-02	1.93E-03	4.80E-02	4.40E-02	4.95E-02	4.32E-02
	Minimum	3.45E-02	3.39E-02	1.68E-03	3.76E-02	3.48E-02	3.69E-02	3.45E-02
	Maximum	4.48E-02	7.41E-02	3.62E-03	6.19E-02	6.41E-02	7.75E-02	5.33E-02
	Variance	4.66E-05	1.51E-05	7.13E-08	3.80E-05	3.35E-05	1.52E-05	6.20E-05
MOPSO_HS	Average	4.49E-02	4.45E-02	1.91E-03	4.76E-02	4.38E-02	4.90E-02	4.20E-02
	Minimum	3.11E-02	3.36E-02	1.65E-03	3.59E-02	3.43E-02	3.61E-02	3.39E-02
	Maximum	5.47E-02	5.39E-02	2.64E-03	5.79E-02	5.28E-02	6.05E-02	5.31E-02
	Variance	4.55E-05	1.82E-05	6.63E-08	3.70E-05	3.34E-05	2.56E-05	4.10E-05
MOGWO	Average	3.25E-01	3.60E-02	1.73E-02	3.70E-01	3.50E-04	3.85E-02	3.32E-01
	Minimum	2.20E-02	3.59E-02	1.68E-02	3.63E-02	3.47E-02	3.39E-02	3.30E-02
	Maximum	5.47E-02	4.81E-02	4.72E-02	7.25E-02	5.36E-02	8.45E-02	5.63E-02
	Variance	3.45E-02	2.61E-03	8.14E-03	4.70E-01	1.76E-02	6.22E-01	3.30E-02

* The most statistically significant outcomes are shown in bold.

Table 5. One of the Comparisons of Performance in Each System Based on RMSE

Problem	MOPSO		NSGA2		ANFIS_SPEA2		MOPSO_HS		MOGWO	
System	TR	TS	TR	TR	TR	TS	TR	TS	TR	TS
S_1	0.04	0.104	0.041	0.093	0.04	0.0102	0.093	0.206	0.05	0.098
S_2	0.042	0.102	0.04	0.086	0.051	0.077	0.086	0.219	0.081	0.075
S_3	0.002	0.003	0.002	0.056	0.002	0.052	0.056	0.24	0.030	0.046
S_4	0.052	0.075	0.051	0.072	0.057	0.072	0.072	0.107	0.046	0.064
S_5	0.046	0.101	0.045	0.078	0.046	0.089	0.078	0.118	0.075	0.095
S_6	0.052	0.086	0.052	0.078	0.021	0.091	0.078	0.118	0.039	0.076
S_7	0.044	0.089	0.045	0.07	0.048	0.077	0.07	0.121	0.65	0.47

* The most statistically significant outcomes are shown in bold.

Table 6. Compute the Sys 3 Using Multi-Objective Metaheuristic Algorithms and ANFIS

TARGET	ANFIS_MOPSO	ANFIS_NSGA2	ANFIS_SPEA2	ANFIS_MOPSO_HS	ANFIS_ MOGWO
0.247601	0.2476008	0.247601	0.247745	0.247600768	0.247901
0.328215	0.328215	0.328215	0.328316	0.328214971	0.328314
0.37428	0.3742802	0.37428	0.334587	0.37428023	0.34536
0.754319	0.7543186	0.754319	0.744901	0.754318618	0.75012
1	1	1	1	1	1
0.804223	0.8042226	0.804223	0.806040	0.804222649	0.807200
0.616123	0.6161228	0.616123	0.616421	0.616122841	0.624600
0.589251	0.5892514	0.589251	0.516401	0.58925144	0.560520
0.616123	0.6161228	0.616123	0.616147	0.616122841	0.616201
0.573896	0.5738964	0.573896	0.512606	0.573896353	0.540120
0.642994	0.6429942	0.642994	0.642910	0.642994242	0.650230
0.754319	0.7543186	0.754319	0.769512	0.754318618	0.753201
0.877159	0.8771593	0.877159	0.778418	0.877159309	0.870230
0.785029	0.7850288	0.785029	0.718202	0.785028791	0.790212
0.731286	0.731286	0.731286	0.748601	0.731285988	0.740230
0.712092	0.7120921	0.712092	0.748730	0.712092131	0.720210
0.746641	0.7466411	0.746641	0.746649	0.746641075	0.746452
0.773512	0.7735125	0.773512	0.775124	0.773512476	0.773210
0.754319	0.7543186	0.754319	0.754318	0.754318618	0.754160
0.796545	0.7965451	0.796545	0.796541	0.796545106	0.796402

Table 7. Performance of Algorithms on the Traffic of LTE Network Problem

The traffic of the LTE network				
Algorithms	Statistic Value	Sys1	Sys4	Sys5
MOPSO	Average	0.0078	0.0075	0.0082
	Minimum	0.0071	0.0071	0.0080
	Maximum	0.0079	0.0085	0.0087
	Variance	2.64E-03	2.64E-03	2.64E-03
NSGA2	Average	0.0081	0.0080	0.0082
	Minimum	0.0079	0.0079	0.0078
	Maximum	0.0082	0.0088	0.0086
	Variance	2.32E-03	2.94E-02	2.71E-03
SPEA2	Average	0.0080	0.0082	0.0090
	Minimum	0.0062	0.0069	0.0070
	Maximum	0.0085	0.0086	0.0089
	Variance	2.71E-02	2.69E-02	2.31E-02
MOPSO_HS	Average	0.0050	0.0057	0.0054
	Minimum	0.0049	0.0050	0.0048
	Maximum	0.0057	0.0059	0.0060
	Variance	3.71E-02	6.69E-02	5.61E-03
MOGWO	Average	0.0060	0.0067	0.0076
	Minimum	0.0042	0.0055	0.0068
	Maximum	0.0078	0.0088	0.0091
	Variance	6.23E-02	1.79E-02	2.52E-02

*The most statistically significant outcomes are shown in bold.

Two real benchmark datasets were also used. The first is the lowest daily temperature recorded in Melbourne, Australia, between 1981 and 1990. There are 3650 instances and it is based on degrees Celsius. The Australian Bureau of Meteorology is the data's original source. The second is a monthly total of sunspot observations spanning little more than 230 years, from 1749 to 1983. There are 2,820 instances, and the units are a count. The dataset's original source is given credit to Andrews & Herzberg (1985). The aforementioned datasets are split in half and supplied to the model for testing and training. (Refer to Table 8)

Table 8. Comparison of the Minimum daily temperatures and Sunspots datasets

Minimum daily temperatures						Sunspots			
Algorithms	Statistic Value	S1	S4	S5	Algorithms	Statistic Value	S1	S4	S5
MOPSO	Average	0.1902	0.1916	0.1915	MOPSO	Average	0.1186	0.1202	0.1190
	Minimum	0.1901	0.1917	0.1913		Minimum	0.1185	0.1201	0.1188
	Maximum	0.1904	0.1918	0.1916		Maximum	0.1188	0.1204	0.1194
	Variance	4.50e-10	5.0e-12	4.50e-10		Variance	2.0e-11	4.5e-11	1.8e-10
NSGA2	Average	0.1903	0.1917	0.1914	NSGA2	Average	0.1186	0.1202	0.1191
	Minimum	0.1901	0.1917	0.1913		Minimum	0.1185	0.1201	0.1188
	Maximum	0.1904	0.1918	0.1915		Maximum	0.1188	0.1202	0.1194
	Variance	4.5e-11	5.0e-12	2.e-12		Variance	4.5e-11	5.e-13	1.8e-10
SPEA2	Average	0.1910	0.1917	0.1911	SPEA2	Average	0.1200	0.1208	0.1208
	Minimum	0.1908	0.1917	0.1909		Minimum	0.1194	0.1207	0.1199
	Maximum	0.1912	0.1918	0.1914		Maximum	0.1206	0.1209	0.1217
	Variance	8.0e-11	5.0e-11	1.2e-10		Variance	7.2e-10	2.0e-11	1.62e-08
MOPSO _HS	Average	0.1191	0.1914	0.1905	MOPSO _HS	Average	0.1188	0.1193	0.1188
	Minimum	0.1877	0.1913	0.1905		Minimum	0.1186	0.1188	0.1188
	Maximum	0.1896	0.1916	0.1906		Maximum	0.1190	0.1199	0.1189
	Variance	1.805e-7	4.5e-11	5.e-13		Variance	8.e-12	6.05e-09	5.e-13
MOGWO	Average	0.1289	0.1990	0.1990	MOGWO	Average	0.1195	0.1223	0.1202
	Minimum	0.1280	0.1980	0.1992		Minimum	0.1194	0.1200	0.1199
	Maximum	0.1915	0.1990	0.1999		Maximum	0.1216	0.1235	0.1220
	Variance	7.2e-11	1.1e-10	4.2e-5		Variance	1.2e-05	1.0e-06	2.63e-04

* The most statistically significant outcomes are shown in bold.

4.2. Wilcoxon Signed Rank Test for Comparison

Three algorithms—NSGA2, SPEA2, and MOGWO—with min metric are compared to the algorithm MOPSO_HS in seven different systems (Sys1 through Sys7). The Wilcoxon signed-rank test, which is the foundation of every comparison, indicates whether there is a statistically significant difference in performance. In Table 9. Indicates the result of the Wilcoxon Signed Rank Test for each Sys.

For Sys1, NSGA2 and SPEA2 (p-values of 0.688 and 0.109, respectively) did not show a significant difference when compared to MOPSO_HS. However, with a p-value of 0.031, MOGWO showed a statistically significant difference, indicating better performance in this situation. With p-values of 0.031 and 0.001, respectively, NSGA2 and MOGWO demonstrated statistically significant differences when compared to MOPSO_HS in Sys2. SPEA2, on the other hand, showed no discernible variation ($p = 0.312$). Sys3 showed a similar trend, with MOGWO demonstrating significance ($p = 0.219$), indicating its higher performance, whereas NSGA2 and SPEA2 failed to show significant differences (p-values of 0.109 and 0.031). Sys4 through Sys7 are analyzed, with different results for each system. Interestingly, Sys5 showed substantial differences for each of the three algorithms, suggesting that it performs differently from MOPSO_HS.

Table 9. The result of the Wilcoxon Signed Rank Test for each Sys

Sys 1			
Algorithm	Test Statistic	p-value	Significant
NSGA2	-0.5	0.688	No
SPEA2	-2.0	0.109	No
MOGWO	-3.0	0.031	Yes
Sys 2			
NSGA2	-3.0	0.031	Yes
SPEA2	-1.0	0.312	No
MOGWO	-6.0	0.001	Yes
Sys 3			
NSGA2	-2.0	0.109	No
SPEA2	-3.0	0.031	Yes
MOGWO	-1.5	0.219	No
Sys 4			
NSGA2	-0.5	0.688	No
SPEA2	-1.5	0.219	No
MOGWO	-2.5	0.078	No
Sys 5			
NSGA2	-4.0	0.004	Yes
SPEA2	-3.5	0.031	Yes
MOGWO	-4.5	0.004	Yes
Sys 6			
NSGA2	-2.0	0.109	No
SPEA2	-0.5	0.688	No
MOGWO	-3.0	0.031	Yes
Sys 7			
NSGA2	-2.5	0.078	No
SPEA2	-1.0	0.312	No
MOGWO	-5.0	0.004	Yes

When integrating these algorithms with ANFIS, their own set of disadvantages should be taken into consideration. NSGA-II algorithm maintains a variety of approaches and applies elitism to achieve several goals, but it might need numerous evaluations. SPEA2 algorithm comprises diversity preservation and elitism but requires parameter adjusting and can be computationally expensive. MOPSO_HS algorithm requires parameter tuning but combines the exploration of PSO with the local search of Harmony Search. It can be used for a variety of situations. MOGWO algorithm employs a population-based methodology inspired by the grey wolf; while effective, it necessitates careful parameter selection and may encounter certain issues.

5. Application of Proposed Model

The suggested model provides a wide variety of potential applications in various companies and industries since it combines innovative multi-objective optimization methods with an ANFIS model. The model can be used for business processes for market analysis, business process optimization, and financial forecasting. For example, it can predict stock prices better, improve the efficiency of supply chain operations, and improve marketing strategies by analyzing client behavior. The model's combination of an ANFIS model with multi-objective optimization algorithms may lead to more precise forecasts in the stock market. Using past stock data and market trends, the algorithm may be able to spot linkages and patterns that traditional analysis methods might have missed. This can help investors and financial analysts make better decisions about buying, selling, or holding onto stocks, which will ultimately enhance portfolio management and boost return on investment.

Moreover, the model can be used to assess and enhance several aspects of the supply chain, such as inventory control, production scheduling, and distribution, in terms of supply chain optimization. The approach considers several criteria, such as cost minimization, lead time reduction, and customer satisfaction maximization, to develop optimal solutions that strike a balance between conflicting objectives. This might result in a supply chain that is more adaptable and efficient, which would save costs, raise satisfaction with clients, and increase the company's overall competitiveness.

Additionally, the proposed methodology has the potential to revolutionize the product development process by offering insights into market demands and forecasting product performance. Through analyzing consumer behavior and market trends, the technique helps businesses find opportunities for new products. It fills gaps in the market, leading to more targeted and successful product launches. In order to ensure that the finished products meet customer needs and expectations, the model can also be used to improve product designs for greater usefulness and cost-effectiveness.

Systems and structures can be designed using the model to have the least amount of environmental impact and energy usage. Through the optimization of building design, insulation, lighting, and HVAC (heating, ventilation, and air conditioning) systems, the model has the potential to significantly reduce energy expenditures and carbon emissions. Additionally, the model can help with the development of intelligent energy management systems, which monitor energy consumption and adjust based on occupancy and usage trends. When everything is said and done, companies and the environment will benefit greatly from the implementation of the proposed model for energy efficiency and product development. Reduced expenses, improved competitiveness, and a lesser carbon impact are some of these advantages.

Overall, the suggested model offers a strong strategy for tackling difficult problems in a range of industries and provides a route forward for more effective, competitive, and environmentally friendly business operations. When put into practice, it could lead to beneficial changes in how businesses run and innovate, as well as open up new opportunities.

6. Conclusion and Future Work

The study introduces innovative multi-objective optimization algorithms integrated with an ANFIS model, demonstrating their superiority over other integrated estimation methods. The dataset, sourced from the Central Bank of the Republic of Turkey and spanning TL and dollar exchange rates from 2007 to 2020, served as the basis for the analysis. Notably, predictions in sys 3, 4, and 6 exhibited the best performance. To further validate the algorithm's efficacy, standard datasets including LTE network traffic, minimum daily temperatures, and sunspot dataset temperatures were examined. The investigation revealed that the proposed algorithm (ANFIS_MOPSO_HS) surpassed ANFIS_MOPSO, ANFIS_NSGA2, MOGWO, and ANFIS_SPEA2 in terms of stability and reliability when applied to real data. This indicates its potential applicability across various real-world systems. In conclusion, the study affirms the applicability of the proposed model in different real-world contexts. In order to further enhance the multi-objective optimization algorithms connected to the ANFIS model, the paper's future studies will concentrate on evaluating new features and increasing parameters. We intend to expand the model's use to non-financial domains such energy forecasting and other financial sectors include stock market forecasting. As well as we intend to improve algorithmic performance to handle larger datasets and integrate real-time data streams. To further guarantee the robustness and generalizability of the model, we plan to perform out validation tests utilizing a variety of datasets. In order to enhance prediction, the model also aims to investigate how well the model integrates with cutting-edge technologies, including edge computing, blockchain, and the Internet of Things.

References

[1] Jang JSR, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans Syst Man Cybern*, vol. 23, no. 3, pp. 665–685, 1993. doi: 10.1109/21.256541

[2] S. Kar, S. Das, P. K. Ghosh, "Applications of neuro-fuzzy systems: a brief review and future outline," *Appl Soft Comput*, vol. 15, pp. 243–259, 2014.

[3] L. A. Zadeh, "Fuzzy sets". *Inf Control*, vol. 8, pp. 338–353, 1965.

[4] A. D. Falehi, "MOPSO based TCSC–ANFIS–POD technique: Design, simultaneous scheme, power system oscillations suppression," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 1, pp. 23-34, 2018.

[5] M. Taheri, M.R. Alavi Moghaddam, M. Arami, "Techno-economical optimization of Reactive Blue 19 removal by combined electrocoagulation/coagulation process through MOPSO using RSM and ANFIS models," *Journal of Environmental Management*, vol. 128, 2013, pp. 798-806. doi: 10.1016/j.jenvman.2013.06.029.

[6] D. Karaboga, E. Kaya, "Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2263-2293, 2019.

[7] B. Haznedar, A. Kalinli, A. "Training ANFIS structure using simulated annealing algorithm for dynamic systems identification," *Neurocomputing*, vol. 302, pp. 66-74, 2018.

[8] H. Marzi, A. Haj Darwish, H. Helfawi, "Training ANFIS using the enhanced Bees Algorithm and least squares estimation," *Intelligent Automation & Soft Computing*, vol. 23, no. 2, pp. 227-234, 2017.

[9] H. S. Pannu, D.Singh, A. K. Malhi, "Multi-objective particle swarm optimization-based adaptive neuro-fuzzy inference system for benzene monitoring," *Neural Computing and Applications*, pp. 1-11, 2019.

[10] A. Jamali, H. Babaei, N. Nariman-Zadeh, S. H. Ashraf Talesh, T. Mirzababaie Mostofi, “Multi-objective optimum design of ANFIS for modelling and prediction of deformation of thin plates subjected to hydrodynamic impact loading” Proceedings of the Institution of Mechanical Engineers, Part L: *Journal of Materials: Design and Applications*. vol. 234, no. 3, pp. 368-378, 2020. doi: 10.1177/1464420716660332

[11] V. Seydi Ghomsheh, M. Aliyari Shoorehdeli, A. Sharifi, M. Teshnehlab, "Multi objective optimization of ANFIS structure," *2007 International Conference on Intelligent and Advanced Systems, Kuala Lumpur*, pp. 249-253, 2007. doi: 10.1109/ICIAS.2007.4658384.

[12] E. G. Carrano, R. H. Takahashi, W. M. Caminhas, O. M. Neto, “A genetic algorithm for multiobjective training of ANFIS fuzzy networks,” In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)* pp. 3259-3265, Jun. 2008.

[13] H. Azimi, H. Bonakdari, I. Ebtehaj, S. H. A. Talesh, D. G. Michelson, A. Jamali, “Evolutionary Pareto optimization of an ANFIS network for modeling scour at pile groups in clear water condition,” *Fuzzy Sets and Systems*, vol. 319, pp. 50-69, 2017.

[14] Q. Liu, L. Xiaofeng, L. Haitao, G. Zhaoxia, “Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art,” *Applied Soft Computing*, vol. 93, pp. 1568-4946, 106382, 2020.

[15] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer. ISBN 978-0-7923-8278-2. Retrieved 29 May 2012, 1999.

[16] Jump up to: a b c d e f Ching-Lai Hwang; Abu Syed Md Masud (1979). *Multiple objective decision making, methods and applications: A state-of-the-art survey*. Springer-Verlag. ISBN 978-0-387-09111-2. Retrieved 29 May 2012.

[17] B. Xu, S. Li, A. A. Razzaqi, L. Wang, M. Jiao, "A Novel ANFIS-AQPSO-GA-Based Online Correction Measurement Method for Cooperative Localization," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-17, 2022, Art no. 9504417, doi: 10.1109/TIM.2022.3156997.

[18] A. Lefteh, M. Houshmand, M. Khorrampanah, G. F. Smaisim, "Optimization of Modified Adaptive Neuro-Fuzzy Inference System (MANFIS) with Artificial Bee Colony (ABC) Algorithm for Classification of Bone Cancer," *2022 Second International Conference on Distributed Computing and High Performance Computing (DCHPC)*, 2022, pp. 78-81, doi: 10.1109/DCHPC55044.2022.9731840.

[19] N. Srinivas, K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms”, *Evol Comput*,vol. 2, pp. 221-248, 1994.

[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, *IEEE T Evolut Comput*, vol. 6, pp. 182-197, 2002.

[21] E. Zitzler, M. Laumanns, L. Thiele, “SPEA2: Improving the Strength Pareto Evolutionary Algorithm”, in *ETH Zurich*, 2001, pp. 21.

[22] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, “Handling multiple objectives with particle swarm optimization,” *IEEE T Evolut Comput*, vol. 8, pp. 256-279, 2004.

[23] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, K. Ziarati, “A multi-objective Artificial Bee Colony for optimizing multi-objective problems,” in: *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, pp. V5-277-V275-281, 2010.

[24] S. Mirjalili, S. Saremi, S. M. Mirjalili, L. D. S. Coelho, “Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization,” *Expert Systems with Applications*, vol. 47, pp. 106-119, 2016.

[25] P. Sharma, B. B. Sahoo, “An ANFIS-RSM based modeling and multi-objective optimization of syngas powered dual-fuel engine,” *International Journal of Hydrogen Energy*, vol. 47, no. 44, pp. 19298-19318, 2022.

[26] C. C. Nwobi-Okoye, B. Q. Ochieze, S. Okiy, “Multi-objective optimization and modeling of age hardening process using ANN, ANFIS and genetic algorithm: Results from aluminum alloy A356/cow horn particulate composite,” *Journal of Materials Research and Technology*, vol. 8, no. 3, pp. 3054-3075, 2019.

[27] M. D. Nguyen, D. D. Nguyen, H. N. Hai, A. H. Sy, P. N. Quang, L. N. Thai, ... , B. T. Pham, “Estimation of recompression coefficient of soil using a hybrid ANFIS-PSO machine learning model,” *Journal of Engineering Research*, 2023.

[28] J. Li, G. Yan, L. H. Abbud, T. Alkhalifah, F. Alturise, M. A. Khadimallah, R. Marzouki, “Predicting the shear strength of concrete beam through ANFIS-GA–PSO hybrid modeling,” *Advances in Engineering Software*, vol. 181, 103475, 2023.

[29] R. Devaraj, S. K. Mahalingam, B. Esakki, A. Astarita, S. Mirjalili, “A hybrid GA-ANFIS and F-Race tuned harmony

search algorithm for Multi-Response optimization of Non-Traditional Machining process," *Expert Systems with Applications*, vol. 199, 116965, 2022.

Conflict of interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies on human.
Participants or animals performed by any of the authors.

Informed consent: Informed consent was obtained from all individual
participants included in the study.

Data available on request: The data underlying this article will be shared at a reasonable request by the corresponding author.

Plagiarism Statement
This article has been scanned by iThenticate ™.