Sakarya University

Journal of Computer and Information Sciences

e-ISSN 2636-8129

VOLUME 6 ISSUE 1 APRIL 2023

SAUCIS

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES

ISSN 2636-8129

April 2023 Volume 6 Issue 1

SAUCIS

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES

ISSN 2636-8129

April 2023 Volume 6 Issue 1

SAUCIS

SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES

ISSN 2636-8129

April 2023 Volume 6 Issue 1

CONTENTS

Sakarya University Journal of Computer and Information Sciences
http://saucis.sakarya.edu.tr/

DOI: 10.35377/saucis...1172027 RESEARCH ARTICLE

LSTM Hyperparameters optimization with Hparam parameters for Bitcoin Price Prediction

I. Sibel Kervancı [1], M. Fatih Akay [2]

[1]Gaziantep University; IT Department Gaziantep/Türkiye
[2] Çukurova University, Faculty of Engineering, Department of Computer Engineering Adana/Türkiye

Corresponding author:
I.Sibel Kervancı , Gaziantep University; IT Department Gaziantep/Türkiye
E-mail address:
skervanci@gantep.edu.tr

Received: 7 September 2022
Revised: 27 November 2022
Accepted: 02 January 2023
Published Online: 22 March 2023

Citation: Kervancı S. and Akay MF. (2023). LSTM Hyperparameters optimization with Hparam parameters for Bitcoin Price Prediction. *Sakarya University Journal of Computer and Information Sciences*. 6 (1). https://doi.org/10.35377/saucis...1172027

ABSTRACT

Machine learning and deep learning algorithms produce very different results with different examples of their hyperparameters. Algorithm parameters require optimization because they are not specific to all problems. This paper used Long Short-Term Memory (LSTM) and eight different hyperparameters (go-backward, epoch, batch size, dropout, activation function, optimizer, learning rate, and the number of layers) to examine daily and hourly Bitcoin datasets. The effects of each parameter on the daily dataset on the results were evaluated and explained. These parameters were examined with the hparam properties of Tensorboard. As a result, it was seen that examining all combinations of parameters with hparam produced the best test Mean Square Error (MSE) values with hourly dataset 0.000043633 and daily dataset 0.00061806. Both datasets produced better results with the tanh activation function. Finally, when the results are interpreted, the daily dataset produces better results with a small learning rate and dropout values. In contrast, the hourly dataset produces better results with a large learning rate and dropout values.

Keywords: LSTM, optimization, Bitcoin, prediction, hparam

1. Introduction

For a prediction problem optimization method, dataset's properties, and learning algorithm's parameters affect the prediction accuracy and learning process. However, there is no theoretical approach to how they should be selected. Instead, parameters are determined by experimental approaches using optimization methods [1]. This paper seeks the most appropriate dataset and hyperparameters to get the best results for Bitcoin price prediction with LSTM. For this reason, this study consists of Bitcoin prices at different time intervals and examines datasets with LSTM. Also, this study detects dominant hyperparameters values and interactions by using important hyperparameters and speeding up the optimization process. An LSTM is a particular type of recurrent neural network (RNN) with long-term memory. There are different types of LSTMs that are not required to be present in the three doors mentioned. But in general, some gates hold or forget that allow data to be transferred. The number of doors can vary in different LSTM examples. For LSTM, there are layers in Keras, the number of hidden neurons, and parameters for each gate. Besides, extra dropout layer parameters can be optimized to avoid exploding gradient problems. The training of LSTM networks in solving several problems, as in neural networks, depends on several hyperparameters that determine several appearances of algorithm reactions [2]. Basically, hyperparameter optimization techniques are manual and automatic.

Manual approaches: Different combinations of values need to be tried repeatedly to obtain optimal values for each of these hyperparameters. For this, a specialist is required.

Automatic approaches: There are several hyperparameter optimization algorithms. Grid Search, Random Search, Particle swarm optimization, simulated annealing, and automated hyperparameter optimization methods. Automatic approaches are

challenging to apply because of their high computational cost [2]. But now, computer clusters or graphics processor unit (GPU) processors enable more experimentation to use to cheaper than last.

Based on the reference [3], it is stated that the result of the manual search can be reliably matched with Random Search for some datasets. As proposed in the same reference Gaussian Process (GP) and Tree Structured Parzen Estimator (TPE) hyperparameter optimization algorithms are close to or surpass the performance of manual and Brute-Force Random Search algorithms. In this paper, the parameters go-backward, epoch, batch size, dropout, activation function, optimizer, learning rate and the number of layers were manually evaluated to understand the effect of hyperparameters on learning. The results of all combinations of parameters and their values are obtained and visualized by using the hparam parameter feature of the Tensorboard. The manual approach of the hyperparameters shows the success that cannot be ignored. The relatively small time series Bitcoin price was used as the manual approach and the automated approach will be used in our future study.

2. Literature review

This paper is divided into two parts: the effect of the dataset and hyperparameters on the learning.

2.1 The effect of the datasets

According to reference [4], bitcoin is primarily based on historical datasets, and seasonality can be weekly, daily or hourly. They used two model time-length 30, 60, 120 minutes and 180, 360, 720 minutes. As a result, there were raising the threshold and the mean holding time, the number of trades reduced while the average profit per trade rose. At the end of 50 days, it almost doubled the investment. Based on the reference [5] , predictions were made using the 1-minute and 30-minute datasets. In the 1-minute datasets, the price fluctuation between two different markets (CNY and USD) is quite high. But in the 30-minute dataset, the fluctuation has improved. Based on the reference [6], review article, different articles were evaluated according to the frequency of the time intervals, which are daily, hourly and different minute intervals. The minute and second datasets are used for instant trading, while the daily dataset is used for further dates. Based on the reference [7], used bitcoin price and S&P500 datasets with daily, 1 hourly, and 15-minute time intervals. Performance was compared with LSTM and selected hyperparameters for datasets in different time intervals. Since dataset from different periods yielded similar results, they combined them into a single ensemble model. As a result, both hourly and daily prices in this paper were used in univariate LSTM.

2.2. The effect of hyperparameters

Based on the reference [7], they used a number of layers, neurons in each layer, dropout rate, learning rate, l2 kernel regularization, optimizer, and momentum. In place of testing each hyperparameter singly, they used Hyperband, which provided an understanding of the effect of changes made in several parameters simultaneously on network performance. They found the best optimizer is 'Adam', and the best activation is 'tanh', which aligns with our results. In contrast, others are different because of parameter intervals. Based on the reference [8], most performance variations depend only on a few hyperparameters, even in very high-dimensional situations related to the relationship between hyperparameter settings and performance. Based on the reference [9], they investigated a subset of possible hyperparameters with Grid Search. Improperly selected learning rate or dropout can make it difficult for the model to learn effectively. When using hparam or Grid Search to overcome this, parameters will be scanned in all possible ranges, resulting in more efficient learning.

3. Methods

This section will describe the methods; Python (version 3.9.7) in Spyder (version 5.1.5), keras library (version 2.7.0), Tensorflow (version 2.7.0) backend and Tensorboard (version 2.7.0). Keras has two main models; one is sequential (), and the other is the model class used with the functional API. This paper was used as a model, sequential (). Keras has two main models; one of them is sequential (), and the other is the model class used with the functional API. This paper was used as a model, sequential (). Downloaded the daily dataset from https://www.investing.com/ and the hourly dataset from https://www.cryptodatadownload.com.Two datasets were used for the daily datasets, consisting of 2700 lines from 01 January 2015 to 22 May 2022, 1469 lines from 15 May 2018 to 22 May 2022. The hourly dataset consists of 36957 records from 06:00 15 May 2018 to 23:00 22 May 2022. While comparing the daily and hourly datasets in section 4.9, it aims to ensure fairness by using the date range of 15 May 2018 to 22 May 2022.

MSE was used to evaluate the results and its formulation is as follows.

$$\mathrm{MSE}=\frac{\sum_{j=1}^{m}(\mathrm{yj}-\mathrm{xj})2}{m} \qquad (1)$$

The dropout technique prevents overfitting and leaves unrelated information from the network to improve performance [10]. Education of deep neural networks is complicated as the parameters of the previous layer and the distribution of the inputs of each layer change during training[11]. For this reason, the effect of the parameters (go-backward, activation function, optimizer, batch size, learning rate, and epochs) with a single layer LSTM has been observed in this paper. For all experiments, if the dropout is not specified, dropout=0.2 was taken to prevent overfitting. At the end of each experiment, the kernel reset so that no results affect each other. In experiments using the hparam feature of Tensorboard, the parameters and interactions that yield better results are more clearly seen as the measurements are logged and visualized. Tested the hparam parameters of the tensorboard by giving the values in Figure 1.

```
HP_NUM_UNITS=hp.HParam('num_units', hp.Discrete([ 32, 64, 128]))
HP_DROPOUT=hp.HParam('dropout', hp.RealInterval(0.1, 0.2))
HP_LEARNING_RATE= hp.HParam('learning_rate', hp.Discrete([0.1, 0.05, 0.001,0.2]))
HP_OPTIMIZER=hp.HParam('optimizer', hp.Discrete(['adam', 'sgd','Adadelta',
                                                 'Adagrad','rmsprop','Adamax']))
HP_ACTIVATION=hp.HParam('activation', hp.Discrete(['tanh', 'softmax', 'relu',
                                                   'sigmoid']))
METRIC_MSE = 'mean_squared_error'
```

Figure 1 Hparam parameters

While discrete in Figure 1 shows that the interval is discrete, Realinterval represents the number interval. Each result of go-backward, activation function, optimizer, batch size, learning rate, and epochs are evaluated separately below.

4. Results

The results for each parameter are evaluated separately below.

4.1. “go-backward” parameter

In our tests with go-backwards parameter, it was observed that the go-backwards='False' value gave better results with the increase of the epoch. Recommended go-backwards should be 'False' for time series prediction. If the go-backwards parameter is suitable for your dataset for instance if natural language processing (NLP) is being studied, go-backward=True should be used, which allows both forward and backward contexts to be used. As shown in Table 1, when 'True' value is compared to 'False', train MSE gives better results, while test MSE gives worse results. Based on Table 1, we conclude that there may be excessive learning in the 'True' value train process.

Table 1 The effects of the go-backward on the RMSE and MSE.

Epoch	Go-backward	Activation function	Optimizer	Train MSE	Test MSE
10	False	tanh	Adam	0.000346	0.013109
10	True	tanh	Adam	0.000354	0.013603
100	False	tanh	Adam	0.000020	0.000672
100	True	tanh	Adam	0.000019	0.000694

Figure 2 True and predicted value of Bitcoin price

As shown in Figure 1, the difference between the actual and estimated values from the test dataset is more reasonable with go-backward=False.

4.2. Activation function

Based on the reference [12], 23 different activation functions evaluated. But in this paper tanh, relu, sigmoid, and softmax were examined.

Table 2 The effects of the optimizer and the activation function on MSE (epoch=100).

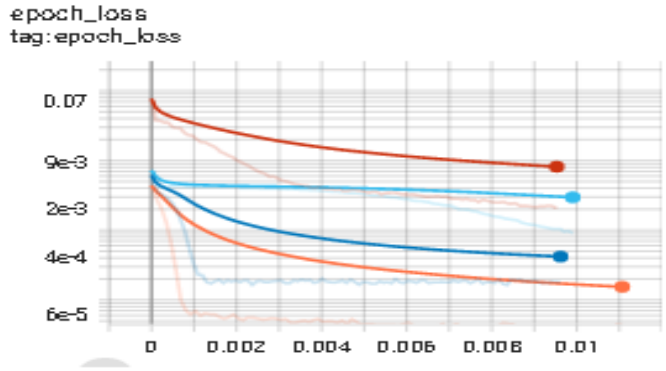
Batch size	Activation function	Optimizer	Train MSE	Test MSE
64	tanh	Adam	0.000019	**0.000667**
64	relu	Adam	0.000033	0.002130
64	sigmoid	Adam	0.000573	0.080025
64	softmax	Adam	0.000460	0.101000
64	tanh	RMSprop	0.000024	**0.000858**
64	relu	RMSprop	0.000023	0.000969
64	sigmoid	RMSprop	0.000279	0.053592
64	softmax	RMSprop	0.000481	0.095665
64	tanh	Adamax	0.000021	**0.001542**
64	relu	Adamax	0.000027	0.003094
64	sigmoid	Adamax	0.002187	0.191153
64	softmax	Adamax	0.003035	0.270137
64	tanh	SGD	0.003852	**0.328543**
64	relu	SGD	0.003741	0.342115
64	sigmoid	SGD	0.004248	0.363644
64	softmax	SGD	0.004231	0.360037
64	tanh	Adagrad	0.004515	0.384147
64	relu	Adagrad	0.003808	**0.331651**
64	sigmoid	Adagrad	0.004130	0.352627
64	softmax	Adagrad	0.004264	0.367113
64	tanh	Adadelta	0.006401	0.420758
64	relu	Adadelta	0.006896	0.417276
64	sigmoid	Adadelta	0.004808	**0.346178**
64	softmax	Adadelta	0.009292	0.446577

As shown in Table 2; root mean square propagation (RMSprop), Adaptive moment estimation (Adam), Stochastic gradient descent (SGD), adaptive moment estimation with maximum (Adamax), adaptative gradient (Adagrad) and adaptive learning rate method (Adadelta) are selected as optimizers. Optimizers (Adam, SGD, RMSprop, and Adamax) achieved the best results with tanh activation function. Adadelta with sigmoid and Adagrad with relu pairs obtained better MSE values than other activation functions. The results with Adagrad and Adadelta optimizers are quite poor. Sigmoid and softmax activation functions generally produce worse results in trials. Sigmoid and softmax activation functions were tried with 1000 epochs as they may require more epochs. Sigmoid with Adam, and epoch=1000 result Train MSE: 0.000026, Test MSE: 0.040009. Softmax with Adam, and epoch=1000 results Train MSE: 0.000023, Test MSE: 0.061527. Even with sigmoid and softmax epoch=1000, it could not approach the Test MSE of the tanh activation function.

4.3. Learning rate

Based on the reference [13], SGDs require manual adjustment of optimization parameters such as learning rates. If a person does not understand the task at hand, it is very difficult to find a good learning rate. In this case, running the learning algorithm with many optimization parameters and choosing the best performing model in a validation set should be selected.

The learning rate was applied to the same LSTM structure in Table 1, 2.



In order to see the results better, the relative property of the horizontal axis epoch value has been applied. Optimizer=Adam; activation functions tanh=orange , relu=darkblue, softmax= blue, sigmoid=brown.

Figure 3 Graph of loss values of activation functions for Adam optimization.

Table 3 The effect of the learning rate on test MSE.

Optimizer	Activation function	Learning rate(Lr)	Test MSE
rmsprop	tanh	0.001	0.00070297
adam	tanh	0.2	0.00070604
adam	tanh	0.001	0.00070997
rmsprop	relu	0.001	0.00078362
Adamax	tanh	0.2	0.00094503
Adagrad	tanh	0.2	0.00101260
Adamax	tanh	0.001	0.00105760
Adamax	tanh	0.1	0.00144720
rmsprop	tanh	0.001	0.00070297

When we examine the results in Table 3, the lowest MSE values are obtained with a 0.001 learning rate. When Table 2 and Table 3 are compared, it is observed that the test MSE value improves with the use of the learning rate parameter except for adam-tanh.

As seen in the Table 2; Adam method enables the adam-tanh pair to obtain the best test MSE 0.000667 without using the Lr. Due to the square root effect, Lr effect gradually diminishes and changes slowly around the globally minimum. In this way, the globally minimum is found faster and it is not requiring more Lr optimization with Adam.

4.4. Optimizer

The selection of the optimizer is important to training. SGD, Ftrl, Adam, Adadelta, RMSprop, Adagrad, Nadam, and Adamax are optimizers. SGD and its variants such as Adam [14], Adagrad [15] and RMSprop [16], are amongst the most popular training methods. Adam was found to be widely applicable despite of requiring less regulation of its hyperparameters [17].

Based on the reference [9], they concluded that Adadelta, Adagrad, and SGD required more epochs, our result supports this conclusion. SGD epoch=1000 Train MSE: 0.000020, test MSE: 0.000859 as a result, concluded that SGD want higher epochs to produce approximate results for other optimizers like Adam, RMSprop. Adam was found to be widely applicable despite requiring less regulation of its hyperparameters. The results in Table 3 were created by taking the best 11 results of that make up the tensorboard hparam parallel coordinate view in Figure 4.

Based on the reference [18], SGD and SGD variants such as momentum have proved to be an effective way of training deep networks. Our results are supported by references [18] , [9]; tanh, relu activation functions produced the best results. In general, Rmsprop and Adam showed the best performances.

4.5. Batch size

Batch is a set of dataset samples. The dataset group in each batch size independently of each other processed in parallel. If we increase the batch size, our approach will be the better, but the process takes a long time and will still result in only one

backward weight update. The number of times the backward parameters are updated is determined by the number of an epoch. It is advised to pick a batch size that is as large as you can afford without going out of memory [19]. Based on the reference [20], they said that the training dataset size should be divisible by batch size to ensure safe termination because they use cyclical learning rates. However, cyclical learning did not apply in this study, and batch size fixed 64 in Tables 1, 2, and 3.

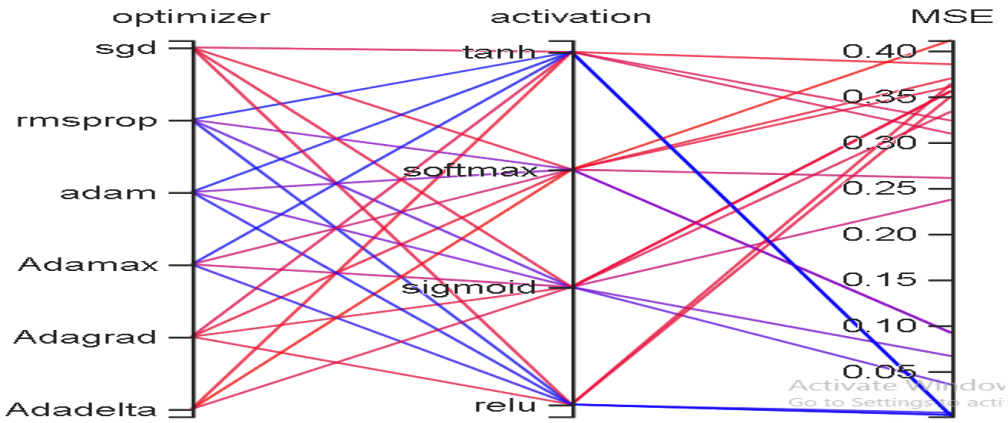


Figure 4 Optimizer and activation function parallel coordinates

Based on reference [21] the impact of batch size on performance, the higher the batch size, the better train and test MSE value like our result in Table 4 (daily dataset). However, it is not a valid generalization for all hyperparameter combinations and datasets, as seen in Tables 7 and 8.

Table 4 The effect of the batch size with daily dataset on test MSE values.

Batch size	Test MSE
32	0.00075906
64	0.00074598
128	0.00066059

4.6. Number of layers

If multi-layer LSTMs are used, usually the first LSTM layer is allowed to output all output (not just one output). This output is fed as input to the next LSTM layer. Since the output of the first layer will be used as input for the next layer, the “return_sequences” parameter of the LSTM layer must be "true".

Table 5 The effect of the number of layers on the train and test MSE values.

Dropout after each layer	Activation function	optimizer	Number of layers	LSTM unit in each layer	Epoch	Train MSE	Test MSE
0.2	tanh	Adam	1	64	100	0.000040	0.005751
0.2	tanh	Adam	2	64	100	0.000021	0.001992
0.2	tanh	Adam	3	64	100	0.000022	0.007281
0.2	tanh	Adam	4	64	100	0.000038	0.015507

Although better train and test MSE values are obtained with increasing layers, the values get worse after a certain number of layers as it continues to increase. As shown in Table 5, while the 2-layer structure produced the best MSE result, the values deteriorated with more than 2 layers.

4.7. Epoch

Epoch shows how many times a dataset is given to the network to train the network. In deep learning algorithms, we need to update the weights and thus pass the whole dataset multiple times to obtain a better and more accurate prediction model to optimize gradient descent. However, it is not clear how many epochs are needed to train a model with the same dataset to achieve optimum weights. For the best train to the network, different datasets proceed differently therefore epoch numbers are different [22]. Based on the reference [9], they measured the number of epochs for an optimizer, convergence observed huge variations in terms of the number of epochs until Nadam converged the fastest, and only requires few training epochs

to achieve good performance. SGD, Adadelta, and Adagrad require more epoch. As seen from Table 1 and Figure 2 the increase in the number of an epoch improved the MSE values. However, after a certain increase, the number of epochs does not contribute to improving the performance, which varies according to the training you have a dataset. As seen in Figure 3, as the epoch increases, the loss decreases, and after a certain epoch value, it doesn't effect on the loss.

4.8. Dropout

Dropout is implemented on any or all hidden layers in the network except the output layer. The dropout technique removes irrelevant information from the network to prevent overfitting and improve performance [10]. Based on the reference [23], they offered dropout and defined how specific hyperparameters should be.
Based on the reference [24], the accuracy of CNN dropout was clearly better than without dropout CNN. It is increased by 58.15% accuracy on a small images' dataset, but our dataset is time series.
According to our experiments, if the dropout value is used, the difference between the train and test MSE values will decrease.

Table 6 The effect of the dropout on the train and test MSE values.

Batch size	Optimizer	Activation function	Dropout	Dataset	Train MSE	Test MSE
64	Adamax	tanh	-	hourly	0.000079	0.000042
64	Adamax	tanh	0.1	hourly	0.000080	0.000045
64	Adamax	tanh	-	daily	0.000020	0.001268
64	Adamax	tanh	0.1	daily	0.000026	0.000969

4.9. The effect of the daily and hourly datasets

The same code with the daily and hourly datasets was executed for the same date range of 15 May 2018 to May 2022. These tests were carried out to see the contribution of daily and hourly datasets to Bitcoin price prediction. Based on the reference [25], g is a noise scale and has an effect on a training and test accuracy.

$$g = Lr(\frac{train_{dataset\ size}}{batch_{size}} - 1) \quad (2)$$

Table 7 The results of the hourly dataset with hparam parameters (epoch=100).

Batch size	Optimizer	Activation function	Dropout	Learning rate(Lr)	Test MSE
64	Adamax	tanh	0.2	0.001	**0.000043633**
64	Adadelta	tanh	0.1	0.05	0.000045415
64	Adadelta	tanh	0.1	0.1	0.000047583
64	Adadelta	tanh	0.2	0.05	0.000047906
32	Adagrad	tanh	0.1	0.1	0.000048397
64	Adamax	tanh	0.1	0.001	0.000048572
32	Adamax	tanh	0.1	0.001	0.000050041
64	Adagrad	tanh	0.2	0.1	0.000051697
64	Adadelta	tanh	0.1	0.2	0.000052008
32	Adagrad	tanh	0.2	0.2	0.000052436

Table 7 shows the best 10 values and parameters out of 576 combinations (Figure 1) for the hourly dataset.

Table 8 The results of the daily dataset with hparam parameters (epoch=100).

Batch size	Optimizer	Activation function	Dropout	Learning rate(Lr)	Test MSE
64.000	tanh	0.1	Adamax	0.001	**0.00061806**
128.00	relu	0.1	adam	0.001	0.00061871
128.00	tanh	0.1	Adagrad	0.100	0.00062113
128.00	tanh	0.2	adam	0.001	0.00062767
128.00	tanh	0.2	Adagrad	0.100	0.00062931
64.000	tanh	0.1	Adadelta	0.200	0.00063148
64.000	tanh	0.2	Adadelta	0.200	0.00063465
128.00	relu	0.2	Adagrad	0.100	0.00063694
32.000	relu	0.1	Adamax	0.001	0.00063716

32.000	softmax	0.1	adam	0.001	0.00063835

Table 8 shows the best 10 values and parameters out of 576 combinations (Figure 1) for the daily dataset.
The daily dataset produces better results with a small learning rate and small dropout values, whereas the hourly dataset produces better results with a large learning rate and large dropout values.

5. Conclusions and Recommendations

The contribution of this paper, the Bitcoin price prediction problem is to infer which of the hyperparameters work better with each other.

We observed that the test MSE values of the hourly dataset gave better results than the daily dataset.

For numerical time series, it is recommended to set the go-backward parameter to False.

Tanh and relu activation functions are suitable for these datasets.

It is quite clear that the SGD optimizer gives better results with the learning rate parameter, and we recommend they to be used together.

In general, the impact of batch size on performance, the higher the batch size, the better the test MSE value.

When all other parameters were fixed, it was seen that tanh and relu activation functions with SGD, Adam, RMSprop and Adamax optimizers, and sigmoid activation function with Adadelta optimizer gave better results.

Achieved the two-layer LSTM with the parameters we optimized, better than the 3 and 4-layer LSTM, which shows us the importance of hyperparameter optimization.

As seen in Tables 7 and 8 the result of "g", better MSE values were obtained when Lr decreased, and batch size was constant, or Lr was constant and batch size increased, or dropout increases, and the other parameters fixed.

When the test MSE values of the daily and hourly datasets are examined, the hourly dataset is recommended since the hourly dataset produces better test MSE values.

References

[1] D. Choi, J. C. Shallue, Z. Nado, J. Lee, J. C. Maddison and E. G. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," *Computing Research Repository (CoRR),* vol. abs/1910.05446, Oct 2019.

[2] B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire and V. Chandra, "Long Short Term Memory Hyperparameter Optimization for a Neural Network Based Emotion Recognition Framework," *IEEE Access,* vol. 6, pp. 49325 - 49338, Sept 2018.

[3] J. Bergstra, R. Bardenet, Y. Bengio and B. Kegl, "Algorithms for Hyper-Parameter Optimization," *Advances in Neural Information Processing Systems 24 (NIPS 2011),* 2011.

[4] R. K. Rathore, D. Mishra, P. S. Mehra, O. pal, A. S. Hashim, A. Shapi'i, T. Ciano and M. Shutaywi, "Real-world model for bitcoin price prediction,"*Information Processing and Management,* vol. 59, no. 4, 2022.

[5] T. Shintate and L. Pichl, "Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning, " *Journal of Risk Financial Management,* vol 12, no. 1, 2019.

[6] I. S. Kervancı and M. F. Akay, "Review on Bitcoin Price Prediction Using Machine Learning and Statistical Methods," *Sakarya University Journal of Computer and Information Sciences,* vol. 3, no. 3, pp. 272-282, 2020.

[7] J. Michańków, P. Sakowski and R. Ślepaczuk, "LSTM in Algorithmic Investment Strategies on BTC and S&P500 Index," *Sensors* , vol. 22, no. 3, 2022.

[8] F. Hutter, H. Hoos and K. Leyton-Brown, "An Efficient Approach for Assessing Hyperparameter Importance," *Proceedings of the 31st International Conference on Machine Learning,* 2014.

[9] N. Reimers and I. Gurevych, "Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks," *EMNLP 2017,* 2017.

[10] A. U. Rehman, A. K. Malik, B. Raza and W. Ali , "A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis," *Multimedia Tools and Applications,* no. 78, pp. 26597–26613, June 2019.

[11] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training Reducing Internal Covariate Shift," *Proceedings of the 32 nd International Conference on Machine Learning,* Lille, France, 2015.

[12] A. Farzad, H. Mashayekhi and H. Hassanpour , "A comparative performance analysis of different activation functions in LSTM networks for classification," *Neural Computing and Applications volume,* vol. 31, pp. 2507–2521, 2019.

[13] Q. V. Le , J. Ngiam, A. Coates, A. Lahiri, B. Prochnow and A. Y. Ng, ”On Optimization Methods for Deep Learning,” *ICML*, 2011.

[14] D. P. Kingma and J. Ba, ”Adam: A Method for Stochastic Optimization,” *Published as a conference paper at the 3rd International Conference for Learning Representations*, San Diego, 2015.

[15] J. Duchi , E. Hazan and Y. Singer, ”Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011.

[16] G. Hinton, N. Srivastava and K. Swersky, ”Lecture 6.5 Rmsprop- Divide the Gradient by a Running Average of its Recent Magnitude,” *Toronto Uni*, 2012.

[17] S. Merity, N. S. Keskar and R. Socher, ”Regularizing and Optimizing LSTM Language Models,” *ICLR 2018*, Vancouver, BC, Canada, 2018.

[18] I. Sutskever, J. Martens and G. D. Geoffr, ”On the Importance of Initialization and Momentum in Deep Learning,” *Proceedings of the 30th International Conference on Machine Learning, PMLR*, 2013.

[19] keras, ”keras,” Feb. 2020. [Online]. Available: https://keras.io/layers/recurrent/.

[20] Smith, Leslie N.;U.S. Naval Research Laboratory, ”Cyclical Learning Rates for Training Neural Networks,” *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Santa Rosa, CA, USA, 2017.

[21] N. Golmant, N. Vemuri, Z. Yao, V. Feinberg, A. Gholami, K. Rothauge, M. W. Mahoney and J. Gonzalez, ”On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent,” *arXiv preprint arXiv:1811.12941*, Nov 2018.

[22] S. SIAMI NAMIN and A. SIAMI NAMIN, ”Forecasting Economic and Financial Time Series: ARIMA vs. LSTM,” *arXiv preprint arXiv: 1803.06386*, 2018.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol.15, pp. 1929-1958, 2014.

[24] K. Pasupa ve W. Sunhem, "A Comparison Between Shallow and Deep Architecture Classifiers on Small Dataset,” *8th ICITEE*, Yogyakarta, Indonesia, 2016.

[25] S. L. Smith, P. J. Kindermans, C. Ying and Q. V. Le, ”Don't decay the learning rate, increase the batch size,” *ICLR (International Conference on Learning Representations)*, Vancouver, 2018.

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/

DOI: DOI: 10.35377/saucis...1209519 RESEARCH ARTICLE

Time-series Forecasting of Energy Demand in Electric Vehicles and Impact of the COVID-19 Pandemic on Energy Demand

Pınar Cihan[1]

[1] Tekirdag Namık Kemal University, Faculty of Corlu Engineering, Department of Computer Engineering, Tekirdag, Türkiye

Corresponding author:
Pınar Cihan, Tekirdag Namık Kemal University, Faculty of Corlu Engineering, Department of Computer Engineering, Tekirdag, Türkiye
E-mail address:
pkaya@nku.edu.tr

Received: 24 November 2022
Revised: 27 December 2022
Accepted: 19 January 2023
Published Online: 30 April 2023

Citation: Cihan, P. (2023). Time-series Forecasting of Energy Demand in Electric Vehicles and Impact of the COVID-19 Pandemic on Energy Demand. *Sakarya University Journal of Computer and Information Sciences*. 6 (1) https://doi.org/10.35377/saucis...1209519

ABSTRACT

The increase in environmental problems such as climate change and air pollution caused by global warming has risen the popularity of electric vehicles (EVs) used in the smart grid environment. The increasing number of EVs can affect the grid in terms of power loss and voltage bias by changing the existing demand profile. Effective predicting of EV's energy demand ensures reliability and robustness of grid use, as well as aiding investment planning and resource allocation for charging infrastructures. In this study, the electricity demand amounts in Boulder and Perth cities are modeled by Support Vector Regression, Random Forest, Gauss Process, and Multilayer Perceptron algorithms. In addition, the impact of the COVID-19 pandemic on energy demand in electric vehicles and the energy demand behavior of EV owners were analyzed. The findings reveal that electric vehicle owners usually start to charge their vehicles during the daytime, the COVID-19 pandemic causes a severe decrease in EVs energy demand, and the support vector regression (SVR) is more successful in energy demand forecasting. Furthermore, the results indicate that the decline in electricity demand during the COVID-19 pandemic caused reductions in the prediction accuracy of the SVR model (a decrease of 17.1% in training and 12.6% in test performance, $P<0.001$).

Keywords: Electric vehicles, energy demand, time series, machine learning, COVID-19

1. Introduction

Energy and fuel consumption has a major impact on climate change. The transportation sector ranks second after electricity generation in terms of fossil fuel use [1]. Global warming levels are predicted to be 2°C above if measures are not taken to reduce greenhouse gas emissions [2]. CO2 is one of the most common greenhouse gases, with around 25% of total global CO2 emissions from transportation [3]. The transportation sector is one of the fastest-growing sectors and the automotive sector has sought an alternative energy source due to the effects of greenhouse gases released into the atmosphere. In addition, according to the UK Department of Transport, 97% of energy consumption in transport comes from the use of oil [4]. The amount of use of the currently limited oil reserves is expected to cause the complete depletion of this resource in the near future. These disadvantages necessitate the use of alternative energy sources.

Electric vehicles (EVs) are a reliable alternative to conventional internal combustion engine vehicles as they are more environmentally friendly and energy efficient. Electric vehicles offer lower carbon emissions compared to gasoline-powered vehicles. The popularity of electric vehicles has increased as environmental problems such as climate change caused by global warming and air pollution has become a global problem. Green transportation vehicles such as e-scooter, e-bikes, and bus rapid transit provide significant advantages in terms of energy security, environmental sustainability, combating air pollution, cost savings, and fuel economy [3]. Because of these advantages, many governments and government agencies are developing targets and policies to incentivize electric vehicles. This has further escalated the boom in the EV market in recent

years and achieved commercially successful [3]. According to the report of the International Energy Agency (IEA), the number of electric vehicles in the world was approximately 3.1 million in 2017, and it is expected to increase to 125 million by 2030 [5, 6].

The Chinese government offers EV manufacturers many incentives for the commercialization of electric vehicles, such as tax reductions, suspension in the purchase of electric vehicles, and increasing charging opportunities. China has the largest electric vehicle market today. A decade ago, the Chinese government established a medium-term strategy aiming to sell a total of 5 million vehicles by the end of 2020. By the end of 2020, an important milestone was reached with the launch of 4.92 million new energy vehicles, including battery-electric, plug-in hybrid, and fuel cell vehicles [7]. The United States is the second-largest country in the world with EV sales around 30% of the global market. Nationwide sales of plug-in hybrid EVs (PHEVs) and battery powered EVs increased nearly 5x from 2012 to 2018 (increased from 50,000 to 360,000). For many years, California has spent millions of dollars driving EV adoption. It also plans to allocate more funds to future infrastructure work [3, 8]. Energy consumption in the transportation sector in South Korea in 2015 constitutes approximately 18% of the country's energy consumption. About 97% of the country's primary energy consumption is based on imports. The government is promoting the development and use of electric vehicle infrastructure to overcome reliance on oil [3]. In light of the Paris Agreement, it is crucial to continue to promote better modeling techniques for the successful adoption of EVs [9].

The significant increase in the number of electric vehicles shows that the countries efforts to promote green transportation vehicles have been carried out successfully. This increase in the number of vehicles also leads to an increase in electricity demand. The increase in electricity demand may cause an interruption if the balance in the grid is not maintained effectively. An uninterrupted electricity supply is essential for the functioning of modern civilization. Therefore, energy demand/load forecasting plays an important role in the operation and planning of electric power [4].

The machine learning approach is widely used to find solutions to problems such as demand forecasting, power quality, and price forecasting in the smart grid. Ensuring sustainable transportation to meet future energy needs is now a vital mission of the countries. Through computer-aided forecasting methods, electrical power distribution can predict demand and power consumers accordingly.

The main motivation of this study is to propose a robust and highly accurate system that can predict electricity demand through real-world EV consumption values in two different regions. The innovations and contributions brought by the study are as follows.

- To propose an effective machine learning-based approach for energy demand forecasting in EVs,
- To observe the energy demand behavior of EV owners,
- To observe the impact of the COVID-19 pandemic on energy demand in EVs,
- To analyze the impact of the COVID-19 pandemic on the energy demand prediction performance of the machine learning model,
- To contribute to EV charging station planners and operators with a successful energy demand forecast.

2. Related Work

Some studies in this field in the literature, including statistical, machine learning, artificial intelligence, and deep learning in energy estimation in electric vehicles, are summarized in Table 1.

3. Materials and Methods

The use of simulated data or information in modeling studies hinders repeatable work and slows down research in the field. Therefore, it is important to work with real-world data in modeling studies. There is a lot of publicly available real-world data on electric vehicles. Open databases of governments are presented in Table 2.

3.1 Research area and data

In this study, energy demand data for electric vehicles in Perth & Kingdon, and Boulder were used. The reason for using two different real-world datasets is to generalize the success of prediction models. The reason for selecting the Perth dataset is containing many samples and data before the COVID-19 outbreak. The reason for choosing the Boulder dataset is that the number of samples is high, and it contains energy demand samples (COVID-19 data) until the near future. Thus, the effects of COVID-19 curfews on energy demand can be observed, and how forecast models affect forecast performance can be analyzed statistics of Perth and Boulder datasets are given in Table 3. In this study, only the energy demand variable was used since it was estimated based on the time series. This variable includes the energy demand amounts used at different times of the day. However, in this study, the total amount of energy demand per day was calculated and used in learning models. The models were trained with 80% of the datasets, and the prediction performances of the models were tested with 20% of the dataset.

Table 1 Some of the studies in the literature about energy prediction in electric vehicles

Author(s)	Method(s)	Accurate Method	Evaluation Criteria(s)	Data	Performance
Unterluggauer 2021 [10]	LSTM	LSTM	MAE	demand forecast	$MAE_{shopping}$=4.35 $MAE_{residential}$=1.53 MAE_{public}=2.7 MAE_{work}=1.85
Yi 2021 [3]	Seq2seq, LSTM	Seq2seq	R^2, MAE, RMSE	charging demand	R^2=0.85 MAE=10.6 RMSE=14.73
Na 2020 [11]	LSTM, DBN, LSTM-DBN	LSTM-DBN	MAPE, RMSE	demand forecast	MAPE=1.03 RMSE=5.67
Huber 2020 [12]	QR, MLPs, KDE	MLP	MAPE, MdAPE, Pinball	parking duration, trip distance	Parking duration=13.7% trip distance= 0.56%
Zhang, 2020 [13]	BPNN, SVM, SAE, TDNN, RNN, DBN, CNN	CNN	MAE, MAPE, RMSE	demand forecast	MAPE=3.21
Zhu, 2019 [14]	ANN, RNN, GRU, SAEs, Bi-LSTM, LSTM	LSTM	MAE, RMSE, R^2	demand forecast	MAE=0.29, RMSE=0.44
Zhu, 2019a [15]	DNN, RNN, LSTM, GRU	GRU	NRMSE, NMAE	demand forecast	NRMSE=2.89 NMAE=0.77
Tat 2018 [16]	RNN, LSTM, MLP	LSTM	Accuracy	energy consumption	-
Louie 2017 [17]	SARIMA	SARIMA	BIC, MSE	demand forecast	BIC_{SD}=13150 MSE_{SD}=610.5 BIC_{WA}=13446 MSE_{WA}=808.4
Amini, 2016 [18]	ARIMA	ARIMA	MAE, MAPE,	power consumption	MAE=1.277 MAPE=1.44
Majidpour, 2016 [19]	CNN, CNN+LSTM, T-GCN	T-GCN	RMSE	energy consumption	RMSE=161
Majidpour, 2014 [20]	SVR, RF, kNN, MPSF	MPSF	MAE, SMAPE	energy consumption	MAE=13.05 SMAPE=14.06
Xydas, 2013 [4]	SVM, Monte Carlo	SVM	MAPE, RMSE	demand forecast	MAPE=3.69 RMSE=50.13

MPSF: Modified Pattern-based Sequence Forecasting, SAE: Stacked autoencoder, TDNN: Time-delayed Neural Network, BPNN: Backpropagation Neural Network, RNN: Recurrent Neural Network, DBN: Deep belief network, QR: Quantile regression, MLP: Multi-layer perceptron, KDE: Kernel density estimator, Seq2Seq: Sequence to Sequence, LSTM: Long short-term memory, SeqST-GAN: Seq2Seq Generative Adversarial Nets, SD: San Diego State, WA: Washington State.

Table 2 Electric vehicle charging station datasets by country.

Countries	Data	Reference
Canada	Historical charging	[21]
Finland	Real-Time charging	[22]
France	Real-Time and Historical charging	[23, 24]
Germany	Real-Time charging	[25]
Netherlands	Real-Time and Historical charging	[26, 27]
Norway	Real-Time charging	[22]
Sweden	Real-Time charging	[22, 28]
UK	Historical charging	[29, 30]
USA	Historical charging	[31-34]

Table 3 Statistics of used datasets in this study

Dataset		Perth	Boulder
Transaction Date	First	09 Jan 2016	01 Jan 2018
	Last	31 Aug 2019	31 Aug 2021
Total Transactions	Weekdays	44664	21198
	Weekends	16393	8022
	All	61057	29220
Park Duration (h)	Mean	1.15	-
	Standard Deviation	2.09	-

3.2 Machine learning models

In this study, different machine learning models are used for energy demand forecasting in Perth and Boulder datasets. These models used are briefly described below.

Support Vector Regression (SVR), is the version of the support vector machine [35] method used for numerical data, that is, for regression. In SVR, there are two situations where the data is linearly separable or linearly inseparable. In linearly separable data structures, two classes can be separated from each other by a line, while to classify data that cannot be separated linearly, the data is moved to a different dimension in various ways, and the data is classified by finding the best separating hyperplane. In nonlinear problems, transformations are performed using kernel functions to analyze the data by moving it to a higher dimensional feature space. The most commonly used are the linear, polynomial, radial basis, and sigmoid kernel functions [36]. Data can be separated linearly by determining the optimum hyper-plane as a result of transformations [37]

Random Forest (RF) [38], consists of two-stage. At the stage of building the RF structure; (i) selection of k features from all features, (ii) compute the node d presenting the best split point among selected features, (iii) create daughter nodes by splitting the node d, (iv) repeat the previous three steps until reaching the desired number of nodes, (v) repeat all the previous steps to build a forest with n number of decision trees. In the estimation stage from the RF model; (i) apply test data to the created decision tree and predict new outputs, (ii) for each predicted value, the votes (importance) are computed, (iii) high voted prediction is considered as the final output value from RF model.

Gauss Process (GP), is a stochastic process in which any finite number of random variables follow a multivariate Gaussian distribution [39]. Various kernel functions are defined on the input samples and the output is created with their weighted sums. The output function is calculated as in Equation (1).

$$\boldsymbol{f(x) = \sum_{j=1}^{H} w_j \Phi_j(x) = w^T \Phi(x)} \quad (1)$$

Here $\Phi(x)$ is a kernel function defined on the input data, w is the weights of the functions, and H is the number of kernel functions.

The process of finding the weights over the predetermined kernel functions is performed with the function minimization optimization in Equation (2).

$$\boldsymbol{J(w) = \frac{1}{2}\sum_{i=1}^{N} (w^T \Phi(x_i) - y_i)^2 + \frac{\lambda}{2} w^T w} \quad (2)$$

Here y_i is the actual value of instance i, λ is the regularization parameter, and N is the number of samples in the training set.

Multilayer Perceptron (MLP), is a fully connected, feedforward type of neural network [40]. MLP consists of at least an input, a hidden, and an output layer. Input neurons transmit the information they receive to neurons in the hidden layer. Hidden layer neurons gather the information they receive from the input layer by weighting, passing it through a function, and transmitting it to the output layer. Output neurons, on the other hand, gather the information they receive from the hidden layer by weighting, passing it through a function, and producing its output. Network training is initially performed by iteratively changing the weights of randomly selected interneuron connections by presenting each training sample to the network. When a training example is given to the network in the training phase, the output of the network is found. Then, for this training example, using the differences between the output of the network and the actual output value, the value of the weights between the neurons is changed to produce values that approximate the actual output. After the outputs of the network

are found in the training process, the error value is calculated for each neuron in this layer. After the errors of the neurons in the output layer are found, they are used to calculate the error values of the neurons in the hidden layer. The weights are updated after the error value of all neurons in the network except the input layer is found. This update process for each sample in the training set is called the epoch. The effect of the update process made in the previous step on each update process is expressed with the momentum term.

3.3 Evaluation metrics of models

Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) metrics [41, 42, 43] were used to evaluate the accuracy of machine learning models in forecasting EV electricity demand. Statistical metrics used in this study and their information are presented in Table 4.

Table 4 Performance metrics used in this study.

Metric	Abbrev.	Equation	Values	Preferred
Mean Absolute Error	MAE	$\frac{\sum_{i=1}^{n}(y_i - x_i)}{n}$	$[0, +\infty[$	Smaller MAE
Root Mean Square Error	RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$	$[0, +\infty[$	Smaller RMSE
Mean Absolute Percentage Error	MAPE	$\frac{1}{n}\sum_{i=1}^{n}\left\|\frac{x_i - y_i}{x_i}\right\| \times 100$	$[0, +\infty[$	Smaller MAPE

n is the number of samples, x_i, y_i : actual and predicted values.

4. Experiments

The total number of vehicles by the time of the Perth and Boulder datasets is given in Figure 1 to observe the charging start time behavior of EV owners. When the graphs are examined, the total number of vehicles that started to be charged between 12:00 and 13:00 for the Perth dataset is the highest. In the Perth dataset, there are 61057 EVs in total, of which 55825, that is, approximately 91.4%, were determined to be starting to charge their vehicles during the daytime (07:00-19:00). When the graph of the Boulder dataset is examined, it is seen that there are three peaks for EVs charge start time. These are the at starting work (08:00-09:00), at lunch (12:00-13:00), and at the end of the working hour (17:00-18:00). It was also observed that approximately 12% of the EVs were charged between 19:00 and 07:00.

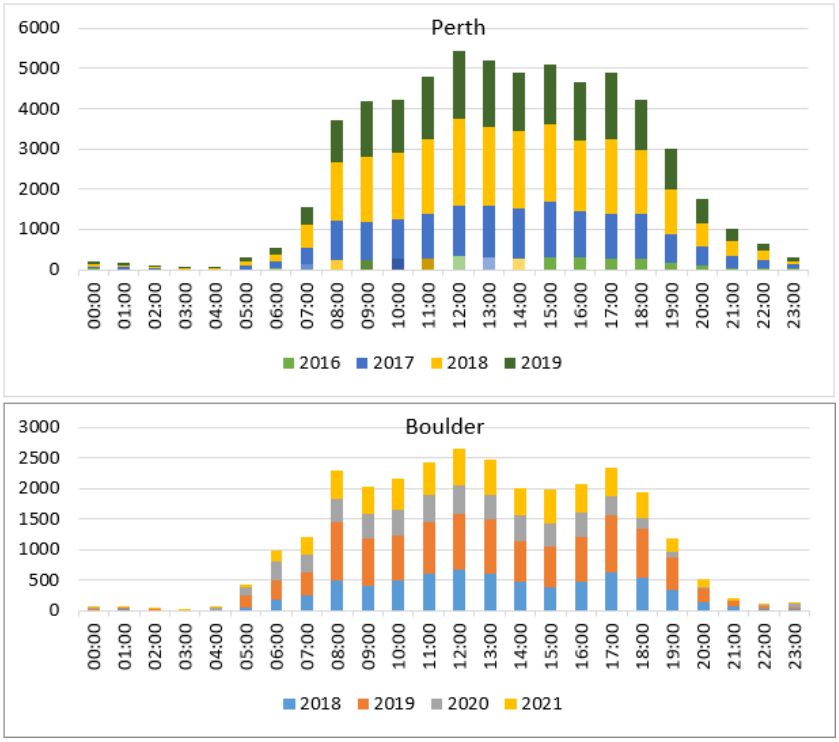


Figure 1 Charging start times of EVs for the Perth and Boulder datasets

The total electricity demand amounts of electric vehicles have been analyzed quarterly and annually, and the graphs are presented in Figure 2.

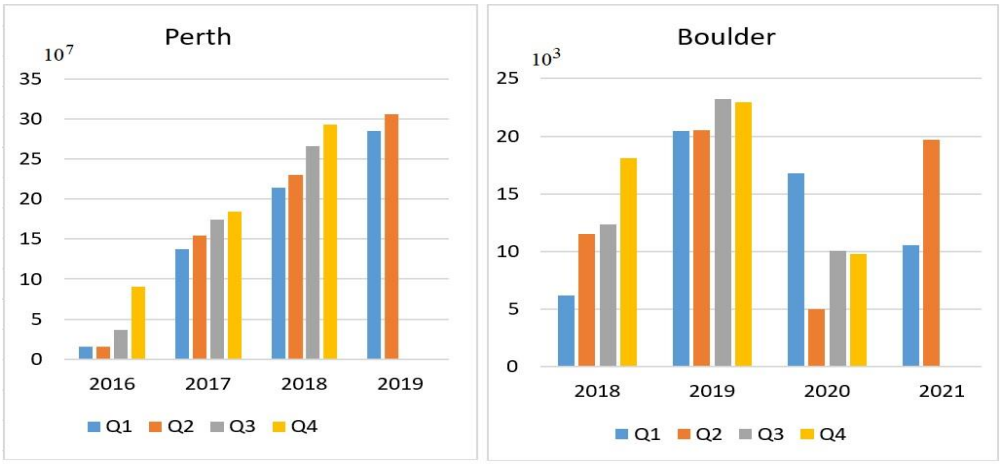


Figure 2 Quarterly total energy demand (kWh) of EVs

The increase in the EV market causes an increase in the amount of electricity demand day by day. When the total energy demand of the Perth dataset is examined, it is seen that the amount of energy demand increases linearly every year. In the Boulder dataset, the total energy demand amount increased considerably in 2019 compared to 2018. However, the COVID-19 epidemic, which started in China in December 2019, caused curfews and thus a decrease in energy demand. The negative impact of the COVID-19 pandemic on energy demand is clearly visible in the 2nd, 3rd, and 4th quarters of 2020 in the Boulder dataset. As a matter of fact, with the relaxation or expiration of the curfews, energy demand increased to its previous levels in the 2nd quarter of 2021. The daily total energy demand amounts in the Perth and Boulder dataset are illustrated in Figure 3. When the daily energy demand amounts are examined, a linear increase is observed in general. However, the curfews due to the COVID-19 outbreak in 2020 caused a significant decrease in the daily energy demand for EVs.

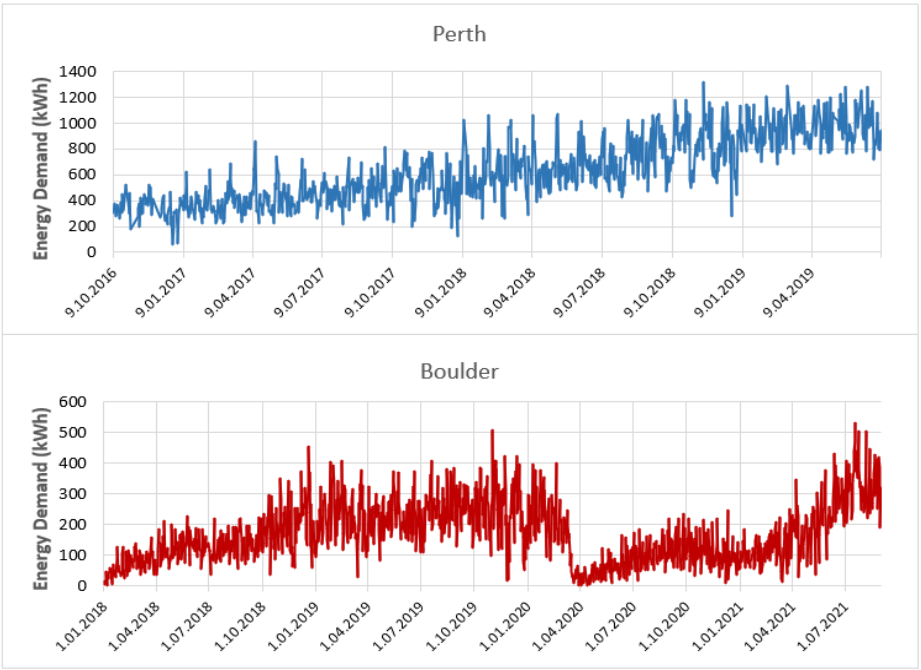


Figure 3 Daily total energy demand (kWh) for the Perth and Boulder datasets

In this study, two different real-world datasets were used to analyze and generalize the energy demand forecasting capability of models. For this, machine learning methods are first trained and then applied to predict at each time point (respectively) by stepping through the data. These forecasts are aggregated and summarized using the MAE (kWh), MAPE (%), and RMSE metrics for each predicted future time step. In this study, 3-time units for forecasting, namely 1-step-ahead (1-s-a), 2-step-ahead (2-s-a), and 3-step-ahead (3-s-a), forecasts are collected and summarized. In this study, each time unit is a day. This allows us to see, to a certain extent, how future predictions change over time relative to closer ones. Models were trained

with 80% of the dataset and tested with 20%. Table 5-8 shows the performance of the prediction models on the training and test data for the Perth dataset. Likewise, Table 9-12 shows the energy demand estimation performances for the Boulder dataset.

Table 5 RF model results on the training set (1017 instances) and test set (203 instances) – Perth dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1005	1004	1003	–	203	202	201	–
MAE	40.10	48.35	55.01	47.82	181.51	182.81	187.99	184.10
MAPE	7.50	9.07	10.41	**8.99**	16.75	16.83	17.29	**16.96**
RMSE	51.39	63.61	72.87	62.62	232.87	235.72	242.63	237.07

Table 6 SVR model results on the training set (1017 instances) and test set (203 instances) – Perth dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1005	1004	1003	–	203	202	201	–
MAE	102.35	106.83	107.89	105.69	122.31	120.27	119.82	120.80
MAPE	19.11	20.13	20.37	**19.87**	12.40	12.16	12.10	**12.22**
RMSE	133.03	138.16	140.00	137.06	153.11	150.21	151.95	151.76

Table 7 GP model results on the training set (1017 instances) and test set (203 instances)– Perth dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1005	1004	1003	–	203	202	201	–
MAE	109.26	114.15	114.89	112.77	127.87	132.26	138.31	132.81
MAPE	21.25	22.29	22.38	**21.97**	13.59	14.23	14.97	**14.26**
RMSE	139.01	144.52	146.12	143.22	159.01	165.02	173.48	165.84

Table 8 MLP model results on the training set (1017 instances) and test set (203 instances) – Perth dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1005	1004	1003	–	203	202	201	–
MAE	117.86	139.19	154.19	137.08	381.07	375.82	588.13	448.34
MAPE	19.99	23.21	25.81	**23.00**	37.44	37.78	57.71	**44.31**
RMSE	148.18	173.65	193.62	171.82	514.84	500.78	743.03	586.22

When the energy forecasting performances of the models are compared, it is seen that the SVR method is more successful than other methods for both the Perth and the Boulder datasets. While the estimation methods made more successful estimations for the Perth dataset, it was seen that the estimation error was higher in the Boulder dataset. The higher model error is thought to be due to the sudden decrease in electricity demand caused by the COVID-19 outbreak. For this reason, the energy demand estimation was made again after the period in which the sudden decrease in the Boulder dataset was experienced was removed from the dataset. The dataset was ended in November 2019 so that the performance of the estimation methods would not be affected by the energy drop caused by the COVID-19 outbreak (Figure 4).

Table 9 RF model results on the training set (1068 instances) and test set (267 instances) – Boulder dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1056	1055	1054	–	267	266	265	–
MAE	16.69	20.59	23.63	20.30	74.38	76.93	81.45	77.59
MAPE	18.06	21.81	25.01	**21.63**	48.32	46.79	47.72	**47.61**
RMSE	21.95	27.77	32.58	27.43	100.30	102.47	108.59	103.79

Table 10 SVR model results on the training set (1068 instances) and test set (267 instances) – Boulder dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1056	1055	1054	–	267	266	265	–
MAE	44.02	46.12	46.79	45.64	53.64	57.64	58.70	56.66
MAPE	47.17	49.75	50.31	**49.08**	39.88	38.13	38.19	**38.73**
RMSE	59.20	62.12	63.33	61.55	70.81	76.05	77.61	74.82

Table 11 GP model results on the training set (1068 instances) and test set (267 instances) – Boulder dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1056	1055	1054	–	267	266	265	–
MAE	50.22	53.87	55.09	53.06	68.23	85.25	100.13	84.54
MAPE	66.81	74.37	77.58	**72.92**	46.91	53.56	63.89	**54.79**
RMSE	63.85	68.39	70.36	67.53	83.99	102.42	119.40	101.94

Table 12 MLP model results on the training set (1068 instances) and test set (267 instances) – Boulder dataset

	Training set				**Test set**			
	1-s-a	**2-s-a**	**3-s-a**	**Avg.**	**1-s-a**	**2-s-a**	**3-s-a**	**Avg.**
#instances	1056	1055	1054	–	267	266	265	–
MAE	41.62	46.04	48.99	45.55	364.21	385.87	305.93	352.00
MAPE	35.15	37.68	39.97	**37.60**	200.35	231.60	178.39	**203.45**
RMSE	52.92	58.60	62.53	58.02	470.39	515.60	432.13	472.71

In this study, energy demand estimation was made for Boulder without COVID-19 samples (non-CoV-Boulder) with SVR, which is the most successful estimation method. In Figure 5, the train and test MAPE results of the datasets are presented comparatively. The MAPE metric is the percentage of error. It is seen that the MAPE value decreases for both train and test sets. This shows that the decrease in energy demand negatively affects forecast performance.

5. Discussion

Nowadays green transportation vehicles have been adopted by all governments due to the many advantages they provide, and various incentives are being carried out by State institutions to increase their use. The use of electric vehicles is increasing significantly every year. The increase in the number of electric vehicles brings with it many problems such as insufficient

charging stations and the inability to meet the electricity demand. For this reason, many studies have been carried out in the literature on the analysis of the behavior of EV owners and the forecasting of energy demand [44-47].

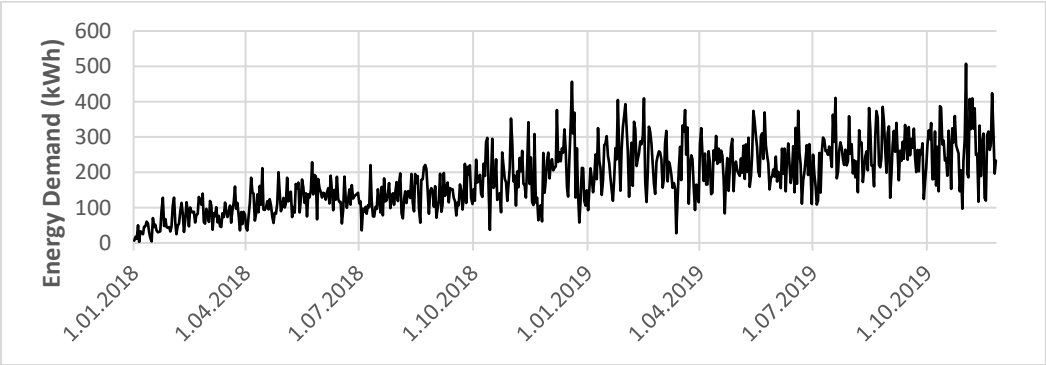


Figure 4 Time series plot of the Boulder dataset without COVID-19 data.

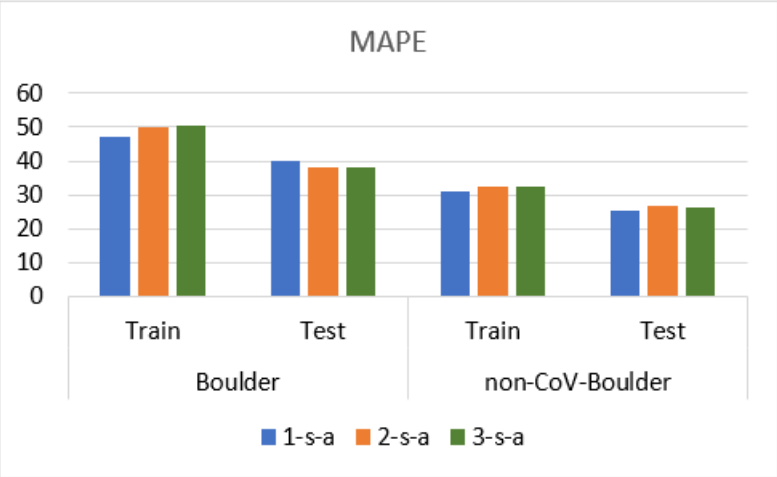


Figure 5 MAPE results of the SVR model for Boulder and n non-CoV-Boulder dataset.

In our study, the total number of vehicles at the start of charging times for two different real-world datasets (Perth and Boulder) was examined to observe the charging behavior of EV owners. In both cities, the starting behavior at night appears to be lower compared to daytime charging (Figure 1). These findings show parallelism with previous studies in the literature [44, 45].

In this study, the most successful forecasting model was determined by comparing the forecasting performances of different machine learning methods for energy demand estimating. Both training and test prediction successes of prediction models are given in detail in Section 4. For a model to be considered successful, the prediction error for the test dataset must be the lowest. The estimation performance of the methods for the test data is summarized in Table 13. The experimental results obtained that the SVR method has the lowest average MAPE value in both test sets.

Table 13 Test average MAE (kWh), MAPE, and RMSE results of all models.

	Perth dataset				**Boulder dataset**			
	RF	**SVR**	**GP**	**MLP**	**RF**	**SVR**	**GP**	**MLP**
MAE	184.10	**120.80**	132.81	448.34	77.59	**56.66**	84.54	352.00
MAPE	16.96	**12.22**	14.26	44.31	47.61	**38.73**	54.79	203.45
RMSE	237.07	**151.76**	165.84	586.22	103.79	**74.82**	101.94	472.71

In our experience, the reason why the energy demand forecast error in the Boulder dataset is higher than in the Perth dataset is that the Boulder dataset contains the COVID-19 data. Because the drastic decrease in energy demand during curfews (see Fig. 3) reduces the predictive ability of the models. For this reason, the samples in the COVID-19 period were removed from the Boulder dataset, and analyses were performed again. Thus, it was possible to measure the effect of the COVID-19 pandemic on energy demand prediction. The energy demand was predicted with the SVR model, which is the most successful forecasting method, and the MAPE values for 1-s-a, 2-s-a, and 3-s-a are presented in Fig. 6. The Post Hoc test was applied

to measure the statistical significance of the mean MAPE values. Experimental results show the COVID-19 pandemic caused a decrease in energy demand for the SVR model of 17.1% in training performance and 12.6% in test performance, P<0.001.

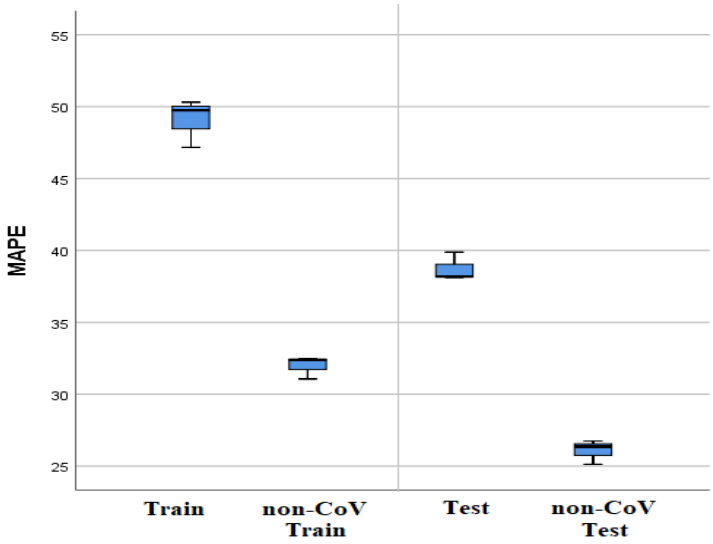


Figure 6 Box-plot of MAPE results

6. Conclusion

Increasing the number of EVs in traffic to reduce global warming and air pollution is among the priority targets of the governments. For this, major incentives are implemented by governments to encourage EV uptake. The serious increase in the number of EVs puts pressure on power system operators. Estimating EV energy demand with good accuracy and reliability is important for the control of the power system. The development of computer-aided energy demand systems will assist in decision-making for electricity market trading. This article contributes to EV energy demand forecasting by comparing machine learning time series models on two different real-world datasets. In addition, the impact of the COVID-19 pandemic on energy demand in electric vehicles and the energy demand behavior of EV owners were analyzed. In the study, the electric vehicle energy consumption of Boulder and Perth city was estimated by SVR, RF, GP and MLP methods. The amount of electric vehicle energy demand in both regions was estimated more successfully by the SVR method ($MAPE_{Perth}$ is 12.2%, and $MAPE_{Boulder}$ is 38.7%). A serious decrease in energy demand has been observed during the COVID-19 pandemic. It was observed that this decrease in energy demand negatively affected the forecast performance of the SVR model. The decrease in demand during the COVID-19 period resulted in a 17.1% decrease in the educational success of the SVR model and a 12.6% decrease in the test pajamas (P<0.001). Finally, it has been observed that electric vehicle owners usually start charging their vehicles during daylight hours.

References

[1] E. P. Agency, "Inventory of US greenhouse gas emissions and sinks: 1990-2005," ed: United States Environment Protection Agency, 2005.

[2] X. Zheng, D. Streimikiene, T. Balezentis, A. Mardani, F. Cavallaro, H. Liao, "A review of greenhouse gas emission profiles, dynamics, and climate change mitigation efforts across the key climate change players," *Journal of Cleaner Production,* vol. 234, pp. 1113-1133, 2019.

[3] Z. Yi, X. C. Liu, R. Wei, X. Chen, J. Dai, "Electric vehicle charging demand forecasting using deep learning model," *Journal of Intelligent Transportation Systems,* vol.26, no.6, pp. 1-14, 2021.

[4] E. Xydas, C. Marmaras, L. M. Cipcigan, A. S. Hassan, N. Jenkins, "Forecasting electric vehicle charging demand using support vector machines," *in 2013 48th International Universities' Power Engineering Conference (UPEC),* pp. 1-6, 2013.

[5] H. Li, Z. Wan, H. He, "Constrained EV charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid.* vol. 11, no. 3 pp. 2427-2439, 2019.

[6] Global EV Outlook, "OECD/IEA," [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2018. [Accessed: 22- Sep-2021].

[7] Global EV Outlook, "Trends and developments in electric vehicle markets," [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets. [Accessed: 22- Sep-2021].

[8] E. Muehlegger, D. Rapson, "Subsidizing mass adoption of electric vehicles: Quasi-experimental evidence from California," *NBER Working Paper*, 2018.

[9] Y. Amara-Ouali, Y. Goude, P. Massart, J. M. Poggi, H. Yan, "A review of electric vehicle load open data and models," *Energies,* vol. 14, no. 3, p. 2233, 2021.
[10] T. Unterluggauer, K. Rauma, P. Järventausta, C. Rehtanz, "Short-term load forecasting at electric vehicle charging sites using a multivariate multi-step long short-term memory: A case study from Finland," *IET Electrical Systems in Transportation,* vol. 11, pp. 405-419, 2021.
[11] Z. Na, T. HanZhen, L. YuTong, C. Jia, Y. JunYou, G. Wang, "Short-term load forecasting algorithm based on LSTM-DBN considering the flexibility of electric vehicle," *in IOP Conference Series: Earth and Environmental Science,* 2020, 042001.
[12] J. Huber, D. Dann, C. Weinhardt, "Probabilistic forecasts of time and energy flexibility in battery electric vehicle charging," *Applied Energy,* vol. 262, p. 114525, 2020.
[13] X. Zhang, K. W. Chan, H. Li, H. Wang, J. Qiu, G. Wang, "Deep-learning-based probabilistic forecasting of electric vehicle charging load with a novel queuing model," *IEEE transactions on cybernetics,* vol. 51, no. 6, pp. 3157-3170, 2020.
[14] J. Zhu, Z. Yang, M. Mourshed, Y. Guo, Y. Zhou, Y. Chang, Y. Wei, S. Feng, "Electric vehicle charging load forecasting: A comparative study of deep learning approaches," *Energies,* vol. 12, p. 2692, 2019.
[15] J. Zhu, Z. Yang, Y. Guo, J. Zhang, H. Yang, "Short-term load forecasting for electric vehicle charging stations based on deep learning approaches," *Applied sciences,* vol. 9, p. 1723, 2019.
[16] T. H. C. Tat and P. Fränti, "Real-time Electric Vehicle Load Forecast to Meet Timely Energy Dispatch," *in 2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI),* pp. 148-153, 2018.
[17] H. M. Louie, "Time-series modeling of aggregated electric vehicle charging station load," *Electric Power Components and Systems,* vol. 45, pp. 1498-1511, 2017.
[18] M. H. Amini, A. Kargarian, O. Karabasoglu, "ARIMA-based decoupled time series forecasting of electric vehicle charging demand for stochastic power system operation," *Electric Power Systems Research,* vol. 140, pp. 378-390, 2016.
[19] M. Majidpour, C. Qiu, P. Chu, H.R. Pota, R. Gadh, "Forecasting the EV charging load based on customer profile or station measurement?," *Applied energy,* vol. 163, pp. 134-141, 2016.
[20] M. Majidpour, C. Qiu, P., Chu, R., Gadh, H. R. Pota, "A novel forecasting algorithm for electric vehicle charging stations," *in 2014 International Conference on Connected Vehicles and Expo (ICCVE),* pp. 1035-1040, 2014.
[21] City Of Edmonton, "Public Charging Stations for Electric Vehicles," [Online]. Available: data.edmonton.ca. [Accessed: 22- Sep-2021].
[22] NOBIL, "NOBIL Database," [Online]. Available: info.nobil.no/eng. [Accessed: 22- Sep-2021].
[23] Data Gouv, "Charging Sessions Apr–May 2017 in Paris," [Online]. Available: www.data.gouv.fr. [Accessed: 22-Sep-2021].
[24] Paris Data, "Belib' Availability in Real-Time," [Online]. Available: opendata.paris.fr. [Accessed: 22- Sep-2021].
[25] Data Bonn, "Charging Point Locations and Usage in Real-Time in Bonn," [Online]. Available: opendata.bonn.de. [Accessed: 22- Sep-2021].
[26] Elaad NL, "Data Analytics," www.elaad.nl/research/data-analytics/. [Accessed: 22- Sep-2021].
[27] Rotterdam Open Data, "Charging Point Locations and Usage in Rotterdam," [Online]. Available: https://rotterdamopendata.nl/#/data. [Accessed: 22- Sep-2021].
[28] Elbil Sverige, "Charging Point Locations in Nordic Countries," [Online]. Available: www.elbilsverige.se. [Accessed: 22- Sep-2021].
[29] Transport Team, "Electric Vehicle Charging Sessions Dundee," [Online]. Available: data.dundeecity.gov.uk. [Accessed: 22- Sep-2021].
[30] OpenData Team, "Electric Vehicle Charging Station Usage in Perth and Kinross," [Online]. Available: data.pkc.gov.uk. [Accessed: 22- Sep-2021].
[31] L. Makram, "Electric Vehicle Charging Stations: Energy Consumption & Savings," [Online]. Available: open-data.bouldercolorado.gov. [Accessed: 22- Sep-2021].
[32] City of Palo Alto, "Electric Vehicle Charging Station Usage," [Online]. Available: data.cityofpaloalto.org. [Accessed: 22- Sep-2021].
[33] City of Evanston, "City-owned Electric Vehicle Charging Station Usage," [Online]. Available: data.cityofevanston.org. A[Accessed: 22- Sep-2021].
[34] ACN-Data, "A Public EV Charging Dataset," [Online]. Available: ev.caltech.edu/dataset. [Accessed: 22- Sep-2021].
[35] V. Vapnik, "The nature of statistical learning theory," *Springer science & business media,* 1999.
[36] M. O. Elish, "A comparative study of fault density prediction in aspect-oriented systems using MLP, RBF, KNN, RT, DENFIS and SVR models," *Artificial Intelligence Review,* vol. 42, pp. 695-703, 2014.
[37] A. J. Smola, B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing,* vol. 14, pp. 199-222, 2004.
[38] L. Breiman, " Random forests," *Machine learning,* vol. 45, pp. 5-32, 2001.
[39] D. J. MacKay, "Introduction to Gaussian processes," *NATO ASI series F computer and systems sciences,* vol. 168, pp. 133-166, 1998.

[40] J. Zurada, "Introduction to artificial neural systems," *West Publishing Co*., 1992.
[41] P, Cihan. "Fuzzy rule-based system for predicting daily case in covid-19 outbreak." *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE, 2020.* https://doi.org/10.1109/ISMSIT50672.2020.9254714
[42] P, Cihan and O, Kalipsiz, "Öğrenci proje anketlerini sınıflandırmada en iyi algoritmanın belirlenmesi." *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi,* vol. 8, no. 1, pp. 41-49, 2016.
[43] P, Cihan, H, Ozel, and H. K. Ozcan, "Modeling of atmospheric particulate matters via artificial intelligence methods." *Environmental Monitoring and Assessment*, vol. 193, no. 5, pp. 1-15, 2021. https://doi.org/10.1007/s10661-021-09091-1
[44] R. Van Den Hoed, J. Helmus, R. De Vries, D. Bardok, "Data analysis on the public charge infrastructure in the city of Amsterdam," *World Electric Vehicle Journal,* vol. 6, pp. 829-838, 2013.
[45] J. C. Spoelstra and I. J. Helmus, "Public charging infrastructure use in the Netherlands: A rollout-strategy assessment," *in Proc. of European Battery, Hybrid and Fuel Cell Electric Vehicle Congress*, 2016.
[46] N. Sathaye and S. Kelley, " An approach for the optimal planning of electric vehicle infrastructure for highway corridors," *Transportation Research Part E: Logistics and Transportation Review*, vol. 59, pp. 15-33, 2013.
[47] J. Liu, "Electric vehicle charging infrastructure assignment and power grid impacts assessment in Beijing," *Energy policy,* vol. 51, pp. 544-557, 2012.

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this manuscript.

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/

DOI: 10.35377/saucis...1170902 RESEARCH ARTICLE

Deep Learning-based Road Segmentation & Pedestrian Detection System for Intelligent Vehicles

Gozde Yolcu Oztel [1], Ismail Oztel [2]

[1]Software Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Türkiye
[2]Computer Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Türkiye

Corresponding author:
Gozde Yolcu Oztel, Software Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Türkiye
E-mail address:
gyolcu@sakarya.edu.tr

Received: 4 September 2022
Revised: 23 February 2023
Accepted: 24 February 2023
Published Online: 30 April 2023

Citation: Oztel G and Oztel I.. (2023). Deep Learning-based Road Segmentation & Pedestrian Detection System for Intelligent Vehicles. *Sakarya University Journal of Computer and Information Sciences*. 6(1) https://doi.org/10.35377/saucis...1170902

ABSTRACT

Correctly determining the driving area and pedestrians is crucial for intelligent vehicles to reduce fatal road accident risk. But these are challenging tasks in the computer vision field. Various weather, road conditions, etc., make them difficult. This paper presents a vision-based road segmentation and pedestrian detection system. First, the roads are segmented using a deep learning-based consecutive triple filter size (CTFS) approach. Then, pedestrians on the segmented roads are detected using an object detection approach. The CTFS approach can create feature maps for small and big features. According to the experiments, the segmentation and object detection results have achieved successful results compared to the literature. The Jaccard index value is 95.84% for segmentation and the average precision value is 65.50% for the people detection task. The proposed system is a low-cost road segmentation and pedestrian detection system for intelligent vehicles.

Keywords: Pedestrian detection, road segmentation, convolutional neural networks, intelligent vehicles, deep learning

1. Introduction

Intelligent transportation systems have been very popular recently. These systems allow efficient and safe planning of the traffic. According to International Road Federation, the goals of Intelligent Transportation Technologies are as follows: (1) Safe roads and driving. (2) Providing sustainable road transportation. (3) Collecting the data. (4) Transferring, processing, and analyzing the data. (5) Smart decision. These technologies can be applied in many fields like vehicle navigation [1], [2], traffic signal control systems [3], [4], [5], and pedestrian detection systems [6], [7]. This technology can be helpful for all humanity, especially visually impaired and disabled people.

To ensure safety, first, the intelligent vehicle's road must be correctly determined. On the other hand, road segmentation is difficult to work in the computer vision field. Because different road and weather conditions, etc., make these tasks difficult. Potholes on the roads may reduce the segmentation performance. Also, for efficient traffic management, roads are designed with various patterns like rectangular, radial, and hexagonal. So, the roads contain small and big features in terms of computer vision. Correctly mapping these features is significant for a high-performance system.

In order to prevent loss of life, the pedestrians on the road must also be correctly determined. Pedestrian detection on the road is also a challenging task because pedestrians may wear clothes in a wide variety of colors. Also, some environmental factors such as light and/or weather conditions, and complex backgrounds can camouflage the pedestrians. In addition, in some environments, the pedestrian body may not be completely observed due to the occlusions.

This study presents a deep learning-based road segmentation and pedestrian detection system. The system first segments road images taken by a vehicle-mounted camera. Then, the system starts to detect pedestrians on the segmented road images. The main contribution of the study is given below:

1) The low-cost proposed system allows robust road segmentation and pedestrian detection tasks using a simple camera.

2) A deep learning-based multi-task system is developed, which applies both segmentation and detection tasks.

The system architecture is shown in Figure 1. As can be seen in the figure, the system includes two subsystems (a and b). The first subsystem(a) has been trained to segment road images as road vs. background. In this stage, a CNN-based algorithm has been developed. The second subsystem(b) has been trained for the human detection task. For this purpose, a pre-trained YOLOv7 network has been used. In the realization stage, the system takes an image using a vehicle-mounted camera. Then, this image is used as an input for both trained subsystems, and results have been obtained. Both system results have been combined in a single final image.

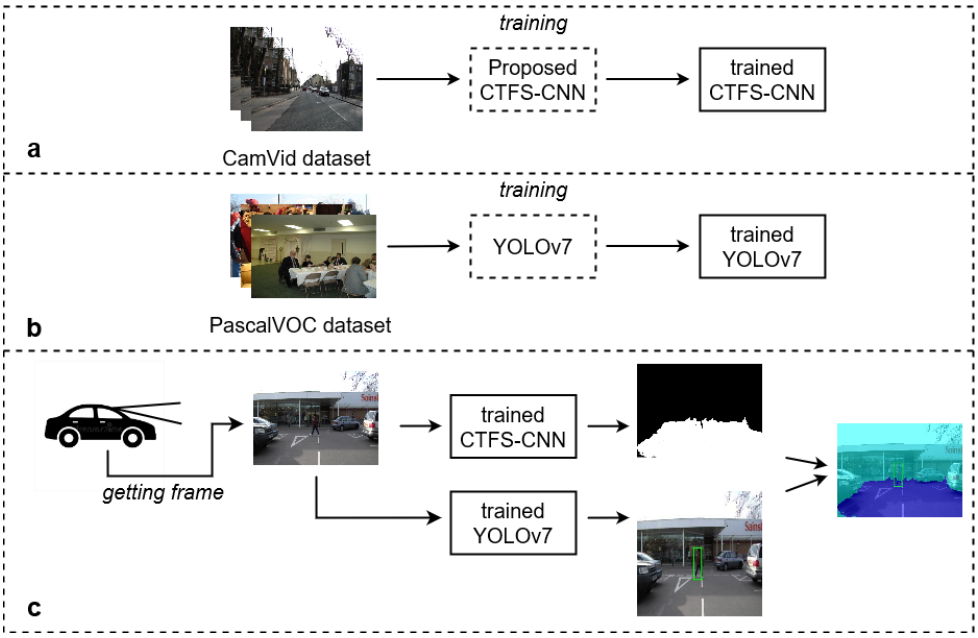


Figure 1 The system architecture: training step for segmentation task (a) and human detection task (b), system pipeline (c)

2. Related Works

2.1 Road segmentation

Many road segmentation studies used radar [8], laser scanners [9], stereovision [10], etc. for detecting road markings or boundaries.

Vision-based road segmentation studies are mainly based on three approaches, namely feature-based, model-based, and neural network-based approaches [11].

Feature-based approaches use color, texture, and edge information to detect and segment the road area [12], [13]. Although this method has the advantage of not requiring prior knowledge of the road shape, it is quite disadvantageous in road conditions such as shadow and water [14]. According to [15], studying road detection using texture information has two disadvantages. One of them is the strong perspective effect of road scenes. Also, random aperiodic textures usually are presented by roads, and these are not easily characterizable.

Model-based approaches build a road model taking into account the shape of the road. They extract the lanes with edge detection and match the lanes with the road model. The disadvantage of this approach is establishing a road model is quite difficult [11].

Neural network-based approaches perform using input and output data. If there is sufficient input data, it can produce successful results.

Recently, deep learning-based approaches have been studied in road segmentation tasks. In [16], the authors modified the structure of a deep network for an effective process in the case of memory and running time. In [15], the authors proposed a CNN approach to segment road scene images. In [17], the authors proposed a Siamese deep neural network based on FCN-8s to detect the road region. They collected the data from a LIDAR sensor and a monocular camera. In [17], the authors proposed a supervised deep Auto-Encoder model for road segmentation tasks. In [18], the authors proposed a CNN model to distinguish different image patches.

2.2 Human detection

Human detection systems are basically developed in three steps. Firstly, the regions which are potentially covered by human components are extracted. Secondly, extracted regions are described. The last step is the classification process for human vs. non-human areas.

In the literature, three basic feature extraction methods have been used for human detection. These are shape-based, appearance-based, and motion-based methods [19]. Shape-based methods use edge-based features for detecting human objects. In appearance-based methods, color and texture information is used. If the object's motion patterns are different, motion monitoring can be used to discriminate objects. For detecting human motion, firstly, temporal features are determined using the temporal difference or optical flows. If the human features are detected, the classification step starts [19].

Also, deep learning has been mostly used for human detection tasks [20][21]. In the proposed study, the pedestrian detection task has been performed using a deep transfer learning approach.

3. Methodology

Recently, deep learning has been popular owing to the increasing data sources and improvement of powerful computer equipment. Convolutional Neural Network (CNN) is a deep learning method that shows successful results in computer vision problems. CNNs can be designed using a different combination of convolution, pooling, ReLU, fully connected, softmax etc. layers.

In convolution layers, filters operate on the input images. The mathematical definition of convolution is given in Equation (1) [22].

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \quad (1)$$

In the Equation: K is a kernel, I is an input image, S is the production result after the convolution operation, and i, j, m, n are the index values.

The pooling layer is not an obligatory layer. It can be used to reduce the input size. It causes a reduction in the size of width and height [23]. Thus, less processing load is provided for the next layers. Also, the pooling layer can help reduce overfitting.

In [24], the authors suggest using a non-linear activation function for quick learning in CNNs. In this paper, ReLU is used for this purpose. The Mathematical Equation of ReLU is given in Equation (2). In the Equation, x is the input of the activation function.

$$f(x) = max(0, x) \quad (2)$$

In order to avoid overfitting in the CNNs, the dropout layer is used [25]. In this layer, some nodes are deleted. Thus, dependence on a particular neuron is prevented. The fully connected layer connects to all nodes in the previous layer. It is used before the classification layer.

The CNNs are trained based on the feed-forward technique. In this technique, information is transmitted through the network. After that, the error value is calculated based on the produced result and the corresponding target. In the proposed study, the error is calculated using the Least Mean Square Error (LMSE). The mathematical definition of LMSE is given in Equation (3) [22].

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 \quad (3)$$

In the Equation, t is the target vector, z is the produced result vector, t and z are vectors with the length of c, w represents the weights, and k represents the index.

In order to increase the network segmentation performance based on the error value Backpropagation Algorithm is used.

3.1 Road segmentation using Consecutive Triple Filter Size Approach - CTFS

Roads may include different patterns such as rectangular, radial, etc. Thus, they have different sizes of features. For effective road segmentation, these features must be correctly detected. For this purpose, in the proposed system, a CNN architecture is designed that contains consecutive 3x3, 5x5, and 7x7 filters in convolution blocks. The structure of convolutional layers with the CTFS approach is shown in Figure 2. In the proposed network, the properties of all consecutive triple convolution blocks are the same. These properties are given in Table 1.

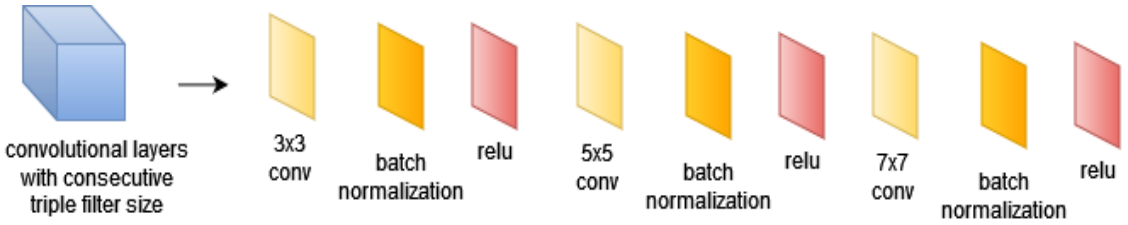


Figure 2 The structure of convolutional layers with CTFS

Table 1 Layer properties of proposed CTFS blocks

Type of layer	#Filter	Filters	Padding
convolution	64	3x3	1
convolution	64	5x5	2
convolution	64	7x7	3

The proposed CNN architecture contains 2 maxpool layers with 2x2, 5 CTFS blocks, 2 transposed-convolution layers with 4x4 size 64 filters, 1 fully connected layer, 1 normalization layer, 1 softmax and 1 pixel classification layer. It also includes 15 batch normalization layers after each convolution layer, and 15 relu layers after each batch normalization layer. The proposed CNN architecture is illustrated in Figure 3.

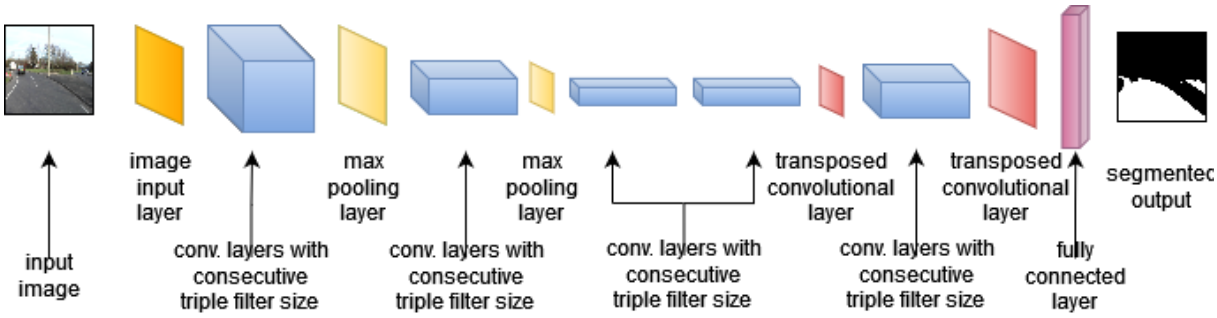


Figure 3 Proposed CNN architecture

Road segmentation is considered as a binary classification problem. Thus, the image pixels have been classified as background vs. road. As the learning rate of 0.001 is used.

The proposed network also includes an encoder-decoder subnetwork. With the encoder, the inputs are downsampled; then, with the decoder, the output of the encoder is upsampled.

3.2 Pedestrian detection using YOLOv7

The YOLO [26], [27] is an object detection network. In the YOLO, the detection problem is structured as a regression problem. The bounding box coordinates and probabilities of each class are produced through regression.

In this network, training images are divided into S× S grids. If the center of the target ground truth is in a grid; this grid is responsible for detecting the target.

Grids predict bounding boxes, their confidence scores, and conditional probabilities of the classes [28]. The mathematical definition of confidence is shown in Equation (4) [28]:

$$confidence = p_r(object) x IoU_{pred}^{truth}, p_r(object) \in \{0,1\} \quad (4)$$

If the target is in a grid p_r(object)=1; otherwise, it becomes 0. In order to represent the confidence between the reference and the predicted bounding box, IoU_{pred}^{truth} is used. The confidence shows if the grid includes objects. If it includes objects, the confidence also represents the predicted bounding box accuracy.

In July 2022, the YOLO family introduced a new version called YOLOv7. [29] claims that this version is the fastest and most accurate real-time object detector to date. The E-ELAN is the computational block in the YOLOv7 backbone. This E-ELAN architecture enables the framework to learn better. It has been designed by analyzing the performance factors such as impact speed and accuracy. YOLOv7 is based on a compound model scaling approach. In YOLOv7, width and depth are scaled in coherence for concatenation-based models.

After the training stage, the re-parameterization method is used to improve the model. Due to this operation, training time may increase, but results can be better. There are 2 types of re-parametrizations: Model level and Module level. For model level re-parametrization, there are 2 options: (1) Train multiple models with different training data but the same settings. Then average weights are calculated to obtain the final model. (2) Calculate the average weights of models at different epochs. In Module level re-parameterization, the model training process is divided into multiple modules. The outputs are combined to obtain the final model [29].

Classic YOLO architecture includes a backbone, a neck, and a head. The head includes the predicted outputs. Differently, YOLOv7 has multiple heads. The head responsible for the final output is called as Lead Head. The head which helps with training in the middle layers is called as Auxiliary Head. Using an assistant loss, the auxiliary heads' weights are updated. Thus, the model learns better [29].

4. Experimental Results

4.1 Database

For the road segmentation task, The Cambridge-driving Labeled Video Database (CamVid) [30] was used. The dataset contains 701 images that were taken by a driving automobile. The database has the original frames and their corresponding labeled frames. There are few images of people on the road in this database. Thus, for pedestrian detection, training could not be done in this database.

The pedestrian detection system has been trained using the last version of the Pascal VOC dataset [31]. This dataset includes four main labels. These are person, vehicle, animal, and indoor. For the pedestrian detection task, the person labels with corresponding coordinate information have been selected. The pedestrian detection system has also been visually tested using CamVid images.

The Pascal VOC dataset contains data with 20 labels. The Roboflow platform [32] was used to use only 'person' labeled data. Also, a YOLOv7 repository [33] was used to train and test the process for person detection.

4.2 Experiments

In the proposed system, firstly road segmentation task has been applied. Using the proposed CTFS approach, the system has been trained and tested with CamVid road images and their ground truths.

Then, the pedestrian detection task has been applied. For pedestrian detection tasks, the system has been trained using YOLOv7. YOLOv7 has been trained with the Pascal VOC dataset for the human detection task. This trained network has been used for CamVid.

Some visual results from CamVid have been shown in Figure 4. In this figure, columns a, and b show original images and their ground truths, respectively. Column c illustrates the segmentation results. Column d shows pedestrian detection results. Finally, column e shows the complete system result. As seen, the system produced visually successful results.

In Figure 5, the automated road segmentation result for a test image has been compared to its ground truth. In this figure, black pixels indicate true negatives (TN), yellow pixels show true positives (TP), red pixels represent false negatives (FP) and green pixels are true negatives (FN). As seen in the figure, FP and FN are mostly produced in boundary fields that separate the two classes. Because the ground truths are manually created in the database, a small margin of error in these fields can be expected.

To better examine the benefits of the proposed consecutive triple filter size approach, an object detection method has been applied to the same road segmentation task. Vgg16 [34] pre-trained network has been adapted to the road segmentation task and retrained. Finally, the results have been compared.

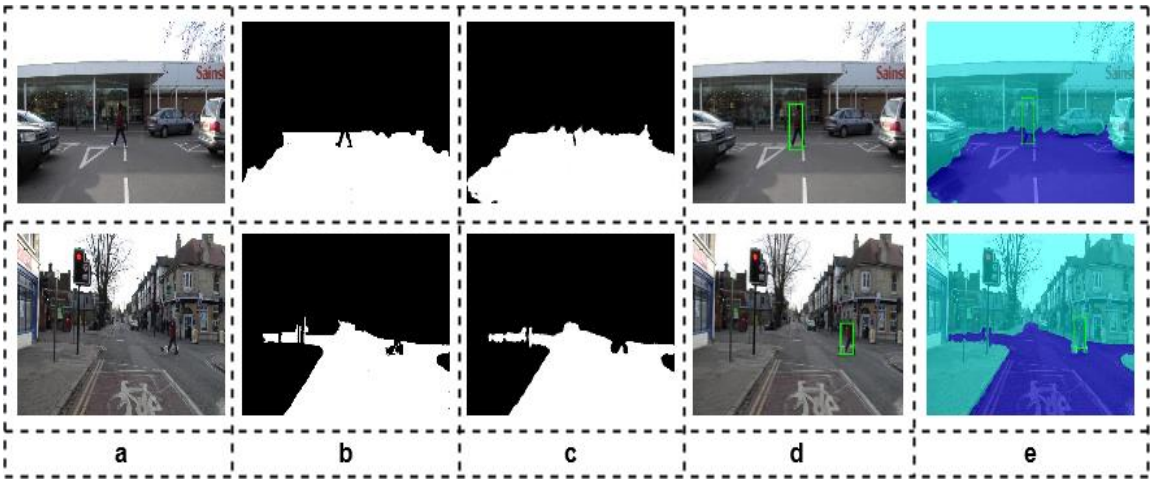


Figure 4 Visual results from the pedestrian detection system. a: original image b: ground truth for road segmentation c: road segmentation result d: pedestrian detection result e: complete system result

To evaluate the proposed CTFS approach, different criteria indexes have been used. These are False Positive Rate (FPR), TPR (True Positive Rate), and IoU (Intersection-Over-Union) metrics. To evaluate the pedestrian detection part; the Average precision (AP) metric has been used. Their mathematical definition is given in Equations (5), (6), (7), (8) and (9).

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

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

$$IoU = \frac{TP}{TP + FP + FN} \tag{7}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{8}$$

$$AP = \sum_n (R_n - R_{n-1)} P_n \tag{9}$$

where P_n and R_n are the precision and recall at the nth threshold.

Figure 5 Visual results from the pedestrian detection system

Table 2 compares the success rate of the CTFS approach with other studies that use the same CamVid dataset. In the different literature studies, different number of classes were used according to their own scenario. The success rates which are reported

in Table 2 belong to the" road" label success rate. These are obtained from the study's confusion matrices. As can be seen in Table 2, the proposed study shows the best performance in terms of the FPR index. In addition, the proposed system outperformed most of the approaches and ranked second behind Vgg16 in terms of IoU and TPR indexes. Vgg16 is a pre-trained network, and it was trained with more than one million images. In Vgg16, just 3x3 filter size was used. Although the proposed CNN architecture was trained with fewer data (1200 images), it produced a close result to Vgg16. Also, for higher quality, the number of data can be increased, and the methods presented in [35] can be implemented to speed up the runtime.

Table 2 Performance comparison of the road segmentation subsystem to other studies that used the same CamVid dataset.

Approach	FPR	TPR	IoU	Inference time
K means [36]	15.80	78.6	63.50	0.25 s
Supervised Deep AE [36]	3.30	97.1	95.40	0.03s
Resnet [37]	-	-	75.8	-
HyperSeg [38]	-	-	78.1	-
Vgg16 (transfer learning)	3.05	97.94	95.98	1.45s
Proposed approach	2.34	97.86	95.84	0.88 s

The comparison table of the pedestrian detection part is shown in Table 3. As can be seen in this table, the proposed pedestrian subsystem gives successful results in terms of the average precision index. Also, the precision-recall curve of the pedestrian detection part is given in Figure 6. A high area under the curve shows high recall and high precision.

Table 3 Performance comparison of the people detection subsystem to other studies that used the same Pascal VOC dataset.

Method	Av. precision	Inference time
HOG III Feature [39]	52.10	2s
G Feature [40]	51.30	-
Fusion of G and T Feature [41]	52.10	-
Faster R-CNN [42]	65.00	-
YOLOv7	65.50	0.2ms

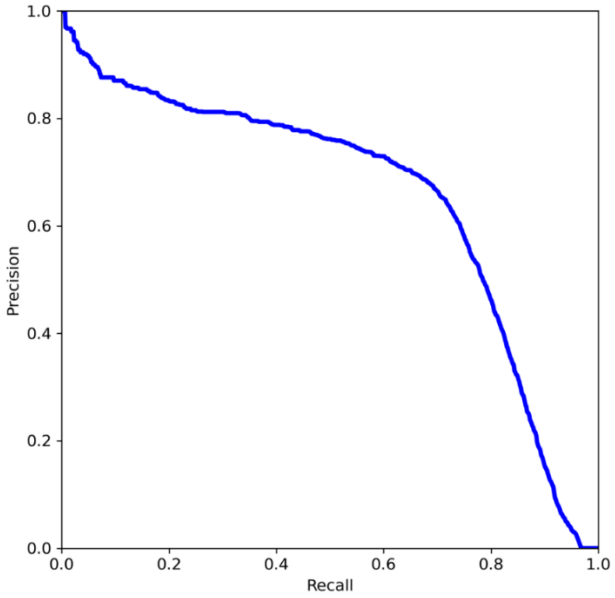


Figure 6 Precision-recall curve of the pedestrian detection task

5. Conclusions

In this study, a deep learning-based road segmentation and pedestrian detection system for intelligent vehicles has been presented. The system includes two main steps. In the first step, the system segments road images and detects roads. In the second step, it detects pedestrians on the roads.

For the road segmentation task, a deep learning-based consecutive triple filter size approach has been developed. This model has been trained and tested using the CamVid dataset. Owing to consecutive 3x3, 5x5 and 7x7 filter blocks, small and big features have been mapped. Also, Vgg16 was retrained for the road segmentation task using the transfer learning method.

Then, the results of both networks were compared. Although Vgg16 was trained using more than one million data, the proposed approach reached close performance using just 1200 data.

After that, the YOLOv7 network has been used for pedestrian detection tasks on the road. In this study, YOLOv7 network has been retrained with the Pascal VOC dataset to perform human detection tasks. Additionally, the trained network has been tested using the CamVid dataset.

Both road segmentation and pedestrian detection subsystems have been evaluated using various metrics. They also compared to literature studies that used the same database. The comparison tables have been reported. They produced promising results compared to the literature. The visual results of the system have also been reported.

In this study, it is focused on artificial intelligence and software to realize the system. This system can be integrated into any embedded system. With a simple camera fixed to the vehicle, the system can take images and transfer these images to a system and produce instant results. We are planning to expand this system for road anomaly detection in the near future.

References

[1] K.-W. Chiang and Y.-W. Huang, "An intelligent navigator for seamless INS/GPS integrated land vehicle navigation applications," *Appl Soft Comput*, vol. 8, no. 1, pp. 722–733, Jan. 2008, doi: 10.1016/j.asoc.2007.05.010.

[2] X. Zhang and M. M. Khan, "Intelligent Vehicle Navigation and Traffic System," in *Principles of Intelligent Automobiles*, Singapore: Springer Singapore, 2019, pp. 175–209. doi: 10.1007/978-981-13-2484-0_5.

[3] J. Jin and X. Ma, "A group-based traffic signal control with adaptive learning ability," *Eng Appl Artif Intell*, vol. 65, pp. 282–293, Oct. 2017, doi: 10.1016/j.engappai.2017.07.022.

[4] J.-Z. Yuan, H. Chen, B. Zhao, and Y. Xu, "Estimation of Vehicle Pose and Position with Monocular Camera at Urban Road Intersections," *J Comput Sci Technol*, vol. 32, no. 6, pp. 1150–1161, Nov. 2017, doi: 10.1007/s11390-017-1790-3.

[5] C. Ma, W. Hao, A. Wang, and H. Zhao, "Developing a Coordinated Signal Control System for Urban Ring Road Under the Vehicle-Infrastructure Connected Environment," *IEEE Access*, vol. 6, pp. 52471–52478, 2018, doi: 10.1109/ACCESS.2018.2869890.

[6] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards Reaching Human Performance in Pedestrian Detection," *IEEE Trans Pattern Anal Mach Intell*, vol. 40, no. 4, pp. 973–986, Apr. 2018, doi: 10.1109/TPAMI.2017.2700460.

[7] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware Fast R-CNN for Pedestrian Detection," *IEEE Trans Multimedia*, pp. 1–1, 2017, doi: 10.1109/TMM.2017.2759508.

[8] B. Ma, S. Lakshmanan, and A. O. Hero, "Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 3, pp. 135–147, 2000, doi: 10.1109/6979.892150.

[9] J. Sparbert, K. Dietmayer, and D. Streller, "Lane detection and street type classification using laser range images," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, pp. 454–459. doi: 10.1109/ITSC.2001.948700.

[10] M. Bertozzi and A. Broggi, "GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 62–81, 1998, doi: 10.1109/83.650851.

[11] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image Vis Comput*, vol. 22, no. 4, pp. 269–280, Apr. 2004, doi: 10.1016/j.imavis.2003.10.003.

[12] Luo-Wei Tsai, Jun-Wei Hsieh, Chi-Hung Chuang, and Kuo-Chin Fan, "Lane detection using directional random walks," in *2008 IEEE Intelligent Vehicles Symposium*, Jun. 2008, pp. 303–306. doi: 10.1109/IVS.2008.4621271.

[13] Q. Li, N. Zheng, and H. Cheng, "Springrobot: A Prototype Autonomous Vehicle and Its Algorithms for Lane Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 300–308, Dec. 2004, doi: 10.1109/TITS.2004.838220.

[14] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image Vis Comput*, vol. 22, no. 4, pp. 269–280, Apr. 2004, doi: 10.1016/j.imavis.2003.10.003.

[15] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez, "Road Scene Segmentation from a Single Image," 2012, pp. 376–389. doi: 10.1007/978-3-642-33786-4_28.

[16] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *2016 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), Oct. 2016, pp. 4885–4891. doi: 10.1109/IROS.2016.7759717.

[17] H. Liu, X. Han, X. Li, Y. Yao, P. Huang, and Z. Tang, "Deep representation learning for road detection using Siamese network," *Multimed Tools Appl*, vol. 78, no. 17, pp. 24269–24283, Sep. 2019, doi: 10.1007/s11042-018-6986-1.

[18] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler, "Convolutional Patch Networks with Spatial Prior for Road Detection and Urban Scene Understanding," Feb. 2015.

[19] D. T. Nguyen, W. Li, and P. O. Ogunbona, "Human detection from images and videos: A survey," *Pattern Recognit*, vol. 51, pp. 148–175, Mar. 2016, doi: 10.1016/j.patcog.2015.08.027.

[20] Y. Kim and T. Moon, "Human Detection and Activity Classification Based on Micro-Doppler Signatures Using Deep Convolutional Neural Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, Jan. 2016, doi: 10.1109/LGRS.2015.2491329.

[21] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2056–2063, 2013, doi: 10.1109/ICCV.2013.257.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[23] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu, "A Taxonomy of Deep Convolutional Neural Nets for Computer Vision," *Front Robot AI*, Jan. 2016, doi: 10.3389/frobt.2015.00036.

[24] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 807–814.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014, doi: 10.1214/12-AOS1000.

[26] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242. doi: 10.1109/IWSSIP48289.2020.9145130.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[28] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved YOLO-V3 model," *Comput Electron Agric*, vol. 157, pp. 417–426, Feb. 2019, doi: 10.1016/j.compag.2019.01.012.

[29] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022.

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[31] M. Everingham, L. Van~Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge 2012 (VOC2012) Results."

[32] B. , Dwyer, J. , Nelson, J. , Solawetz, and et. al., "Roboflow (Version 1.0)," *https://roboflow.com.*, 2022.

[33] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022.

[34] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, Sep. 2015.

[35] Ö. Dülger, H. Oğuztüzün, and M. Demirekler, "Memory Coalescing Implementation of Metropolis Resampling on Graphics Processing Unit," *J Signal Process Syst*, vol. 90, no. 3, pp. 433–447, Mar. 2018, doi: 10.1007/s11265-017-1254-6.

[36] X. Song, T. Rui, S. Zhang, J. Fei, and X. Wang, "The Road Segmentation Method Based on the Deep Auto-Encoder with Supervised Learning," *Computers & Electrical Engineering*, vol. 68, pp. 381–388, 2018, doi: 10.1007/978-3-319-69877-9_28.

[37] B. L. Priya, S. Jayalakshmy, G. Idayachandran, and S. Kumaran, "Performance Analysis of Semantic Segmentation using Optimized CNN based SegNet," in *2022 International Conference on Smart Technologies and Systems for Next*

Generation Computing (ICSTSN), Mar. 2022, pp. 1–5. doi: 10.1109/ICSTSN53084.2022.9761293.

[38] W. Nagai, T. Katayama, T. Song, and T. Shimamoto, "High Efficiency Dataset Generation for Semantic Video Segmentation on Road Intersection," in *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, Jul. 2022, pp. 1–4. doi: 10.1109/ITC-CSCC55581.2022.9894901.

[39] Yunsheng Jiang and Jinwen Ma, "Combination features and models for human detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 240–248. doi: 10.1109/CVPR.2015.7298620.

[40] H. Htet Lin, "Person detection based on fusion histogram of gradients with texture (FHGT) local features," *International Journal of Research in Computer Scientific (IJRCS)*, vol. 5, no. 4, 2018.

[41] H. Htet Lin, "Smart Feature Fusion and Model for Human Detection," *Review of Computer Engineering Research*, vol. 7, no. 1, pp. 38–46, 2020, doi: 10.18488/journal.76.2020.71.38.46.

[42] I. Oztel, "Human Detection System using Different Depths of the Resnet-50 in Faster R-CNN," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2020, pp. 1–5. doi: 10.1109/ISMSIT50672.2020.9255109.

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/

DOI: 10.35377/saucis...1210687 RESEARCH ARTICLE

On Orbit Demonstration of Pointing Accuracy of Ground Antennas by a Flying GEO Satellite

U. Cezmi Yilmaz[1], Umit Guler[2]

[1, 2] TURKSAT AS, Satellite Control Center, Golbasi, Ankara, Türkiye

Corresponding author:
U. Cezmi Yilmaz, TURKSAT AS, Satellite Control Center, Golbasi, Ankara, Türkiye
E-mail address:
cyilmaz@turksat.com.tr

Received: 27 November 2022
Revised: 27 January 2023
Accepted: 09 March 2023
Published Online: 30 April 2023

Citation: Yilmaz UC and Guler U. (2023). On Orbit Demonstration of Pointing Accuracy of Ground Antennas by a Flying GEO Satellite. *Sakarya University Journal of Computer and Information Sciences* 6 (1) https://doi.org/10.35377/saucis...1210687

ABSTRACT

Geostationary Satellites (GEO) are being used commonly in the communication market. The service providers uplink or downlink the signal using their dedicated antennas (whether with or without tracking capability) to the GEO satellite. The satellite down-converts and amplifies the signal before sending it back to the end users on Earth. Usually, the user sets and adjusts their ground antenna to follow or cover the GEO satellite movement as much as possible. As soon as there is no reduction in the link budget, this pointing is assumed to be successful. On the other hand, the input power of the satellite, together with satellite longitude vs. latitude, can give reasonable ideas about the accuracy of the ground antenna pointing. In this study, ground station pointing performance is shown in two different cases: one with tracking and one without tracking capability.

Keywords: GEO satellite, Tracking, Satellite communication, antenna pointing

1. Introduction

Theoretically, the GEO satellites are located at about 42165.8 km from the center of Earth and assumed to be stable with reference to the Earth. But, due to different kinds of perturbations acting on the satellite, this movement cannot be stable and needs to be controlled using propellants on board [1]. The users, who have larger antennas, need to track the GEO satellites since the larger parabolic antennas have narrower beams according to the beamwidth formula [6]. On the other hand, this kind of antenna has more gain that is taken into account in the link budgets. Besides, users with smaller-sized antennas have to point and fix the antennas through the center of the satellite control box, as precisely as possible, to perfectly receive or transmit the power whenever the satellite is inside the dedicated control box. Nominally, for GEO satellites, ±0.1 degrees of the control box in latitude and longitude are used.

In the literature, there are multiple studies regarding optimal tracking of GEO satellites [2], implementation of step-tracking [3], optical calibrations for these systems [4] or finding GEO satellite directions [5]. There are also some studies that detail the importance of pointing accuracy of the ground antenna [7], the movement of the satellite coverage when the inclination of the orbit is not controlled [8] and pointing errors due to perturbation on satellite movement [9]. In [10], the effects of the orbital parameters on the satellite link budget were simulated.

In this study, we figure out, (1) how can we make sure that the tracking performance of the ground antenna is satisfactory? And (2) how the pointing of the fixed antenna can be confirmed? That's why this study focuses on two kinds of users, 'with'

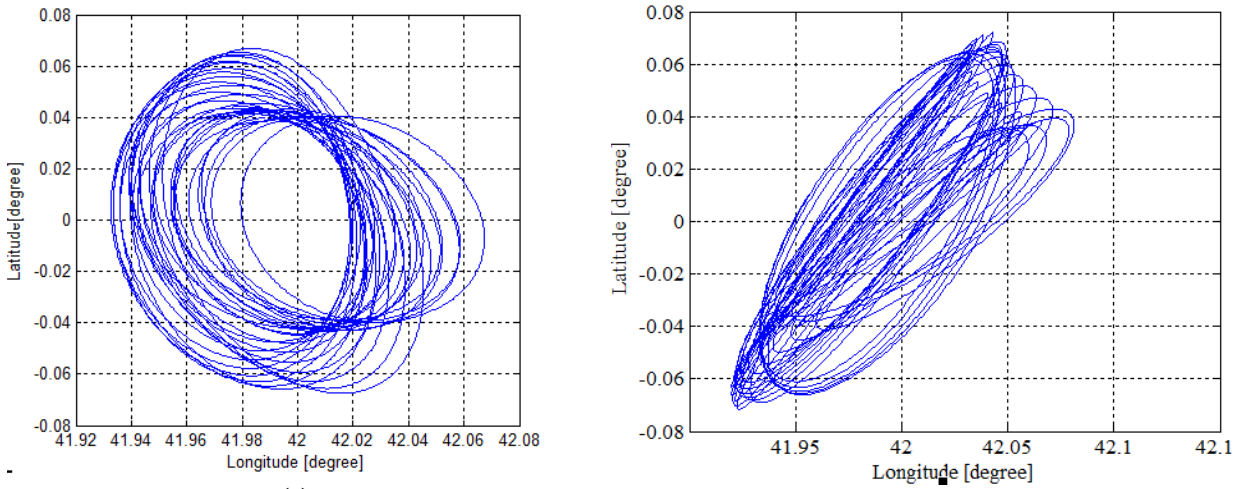
and 'without' tracking antennas in the ground station. The latitude and longitude of the GEO satellite, collected from Flight Dynamics Software, with received power of antenna (on satellite or ground-based on the case study) collected from the ground system, are used together. For the complete analysis, a flying GEO satellite is used.

Case-1 uses a tracking ground antenna with a single uplink carrier to satellite. For case-2, a fixed ground antenna with a single carrier at downlink is used. We paid attention to observing the power level, whether on satellite or on ground, due to a single carrier. The weather was mostly clear sky, sometimes cloudy.

2. Case 1: Tracking antenna received power at satellite

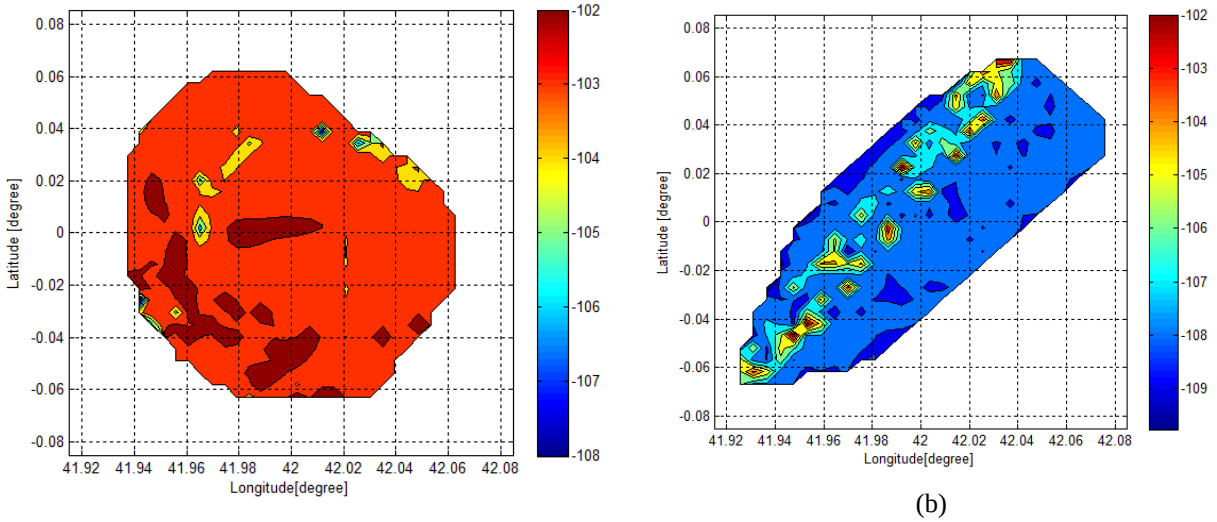
For case 1, an antenna with a diameter of 6.2 meters is used, which has 56.35 dBi gain at 14.425 GHz. The antenna uses the monopulse tracking option to follow the beacon signal coming from the satellite and uplink a single carrier to the satellite receiver. To interpret the tracking accuracy, the input power level of the satellite receiver is used with the same time label of satellite latitude and longitude. Two months of intervals are selected for different cases which show good representations. The GEO satellite, used in this study, had two chemical orbital control maneuvers in 14 days cycles; first, a North/South maneuver to control the inclination and after a couple of days, an East maneuver to control the longitude and drift of the satellite.

The hypothesis in this case-1 is; if the satellite always had the same level of input power reception during its movement inside the box for the selected period, this shows that the ground antenna is following the satellite perfectly. As two examples, Figure 1 below shows the satellite movement in latitude and longitude during (a) February 2020 and (b) May 2020.



(a) (b)

Figure 1 GEO Satellite orbit at 42-degree East in (a) Feb. 2020 (b) May 2020.



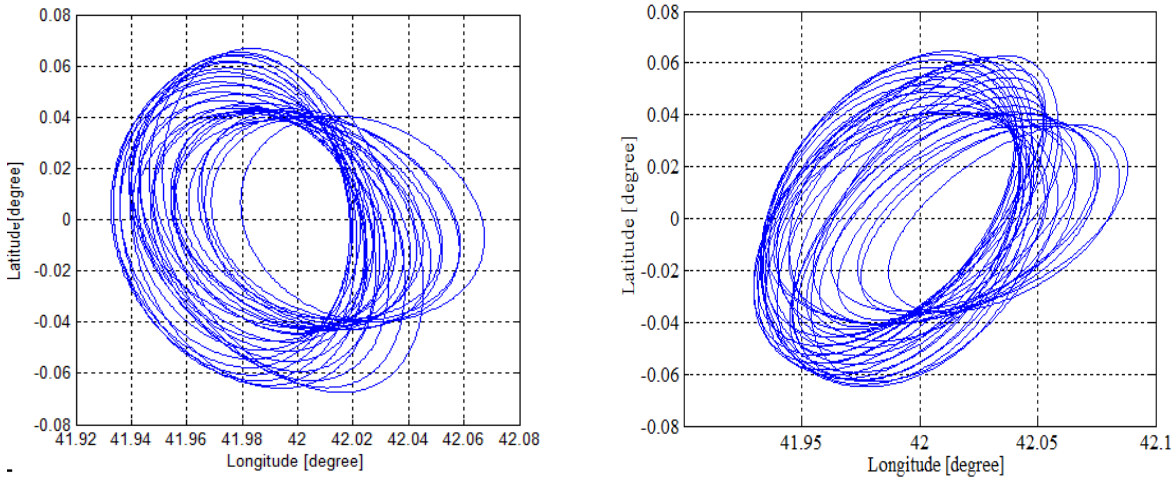
(a) (b)

Figure 2 The input power level at the GEO satellite from tracking ground antenna (a) Feb 2020 and (b) May 2020

Figure 2 below shows the input power level of the satellite receiver in color code while the satellite moves in its dedicated control box. As can be seen, there are some small areas where the link budget degraded during the weather condition and different power levels between the months. But overall, the input power level is almost stable and homogenous at each individual duration, as mentioned in the hypothesis. That shows the excellent performance of the tracking capability of the ground antenna, and we can easily confirm this result from the plot in Figure 2 below. The received power in the color codes is in dBm. The level decreased from February to May 2020 due to operational requests, but the distribution of the power level shows a small variation in time. This means the tracking performance of the ground antenna is successful.

3. Case 2: non-tracking antenna, received power at ground antenna:

For case-2, a non-tracking ground antenna is used with its received power. The hypothesis here is; while the satellite transmits the beacon signal continuously if the ground antenna is correctly pointed towards to center of the control box, it should receive the maximum level, especially when the satellite travels through the center of the box, which is 42-degree East in longitude and 0-degree in latitude in our case. For this case, February 2020 and April 2020 are used, and the satellite movements inside the box are shown in Figure 3 below.

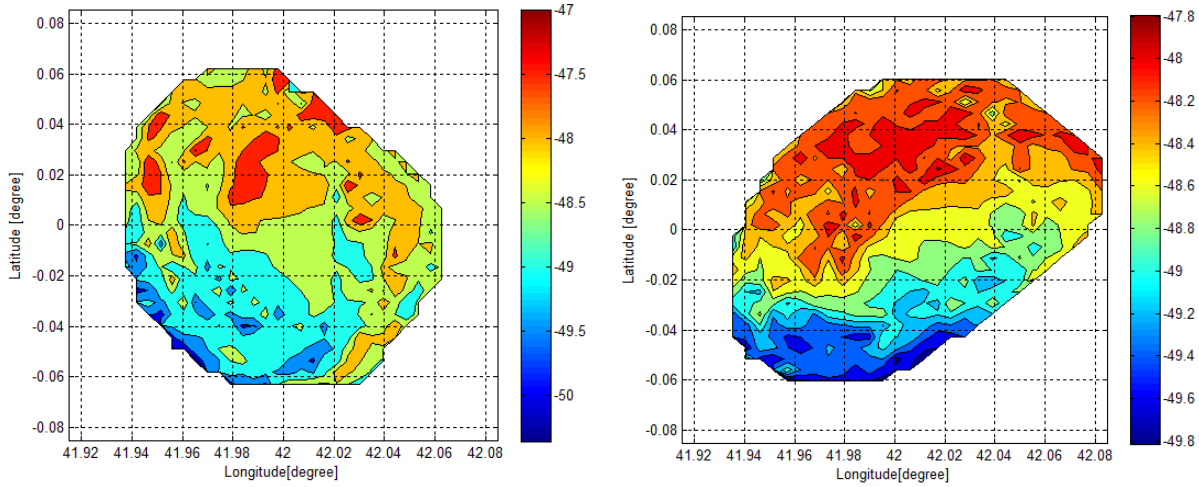


(a) (b)

Figure 3 GEO Satellite orbit at 42-degree East in (a) Feb. 2020 (b) April 2020.

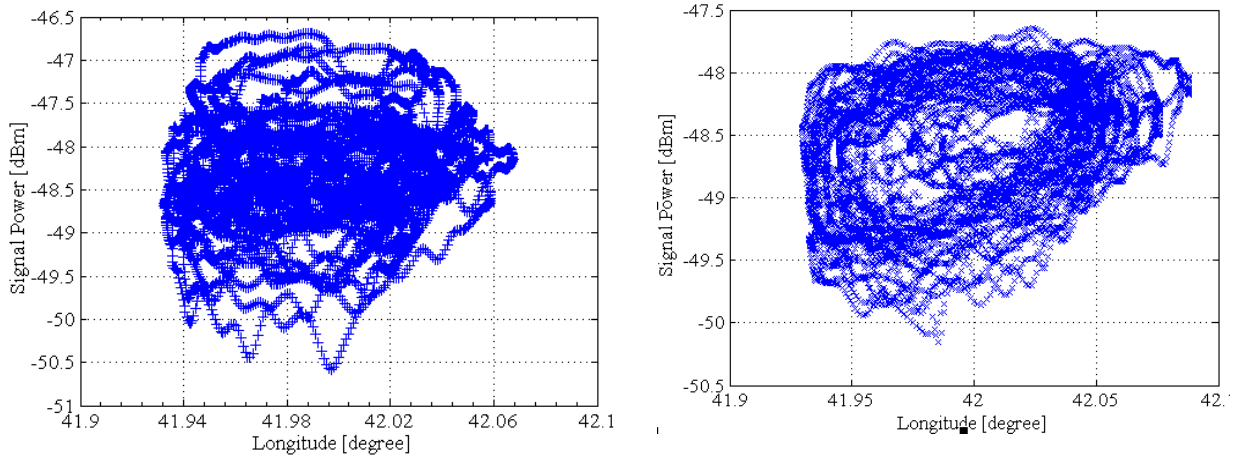
The used ground antenna has 2.4 meters of diameter with 26 dB/K of G/T (@ 20° elevation, 10.7 GHz) and about 0.79545 degrees of theoretical 3 dB beamwidth for 11.0 GHz receiver frequency [6]. The power levels in Figure 4 below are in dBm.

As shown in Figure 4 above, the input power level of the fixed antenna is not homogenous even though the satellite emits no or negligible variation in time. For this example, the ground antenna does not precisely point to the center of the box. Still, it has a minor error in Azimuth (longitude) but more in Elevation (latitude). For more details, Figure 5 and Figure 6 show the power level versus longitude and latitude for given time durations.



(a) (b)

Figure 4 The input power level at the non-tracking ground antenna receiver on (a) Feb 2020 and (b) April 2020



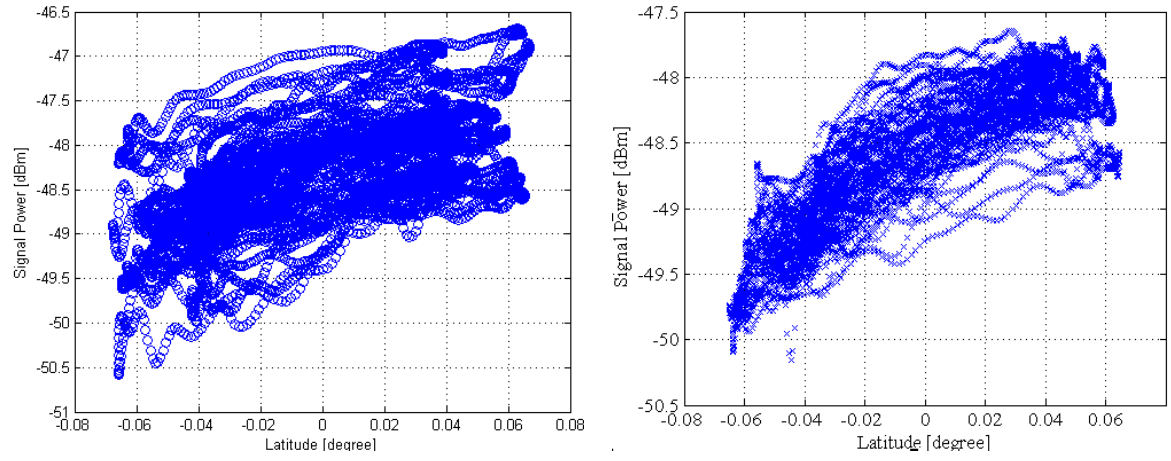
(a) (b)

Figure 5 Power level versus satellite longitude (a) Feb 2020 and (b) April 2020

As seen from Figure 5, there is a variation in signal power level due to the thermal behavior of ground equipment. There is no significant variation in Longitude concerning the signal power.

On the other hand, Figure 6 below shows the observed variations in time, which causes a similar variation as in longitude. As can be seen, there is a trend to increase when satellite travels in the North hemisphere for both time durations. This variation can explain the de-pointing of the ground antenna. When the satellite moves through the North, the input power level of the ground antenna increases and vice versa when it moves to the South.

As can be seen from Figure 5 and Figure 6, the input power distribution inside the box is more stable in Figure 5. But the input power shows some trend behavior in Figure 6. This is evidence that this ground antenna has not been fixed to the center of the box of the related GEO satellite perfectly. It can be seen that the origin of the power distribution diagram is at almost 42.0 degrees in longitude and 0.03 degrees in latitude. This means the azimuth and/or elevation must be corrected to eliminate the error in latitude. It has been noted here that the correction of the latitude for about 30 mdeg requires about -9 mdeg of correction in azimuth and about 34 mdeg of correction in elevation (based on the ground antenna location) by a generic Az/El calculation [6].



(a) (b)

Figure 6 Power level versus satellite latitude (a) Feb 2020 and (b) April 2020

4. Conclusion

For GEO satellite communication links, it is essential to be sure that both the transmitters and receivers are well-pointed to each other. This study shows that the orbital information of the satellite gives reasonable outputs regarding the pointing of the ground antenna if they are used together with the received or transmitted power. This would allow the user to calculate the pointing losses more accurately in link budgets.

It is found in this study that pointing a ground antenna to a GEO satellite requires a good setting of azimuth and elevation angles or good tracking of the GEO satellite. In that sense, the GEO satellite operators may provide more realistic values to the end users regarding the pointing performance of their antennas and possible pointing loss in link budgets.

References

[1] E. M., Soop, “Handbook of Geostationary Orbits”, Kluwer Academic Publishers, Netherlands, 1994.

[2] M. Richharia, “An Optimal Strategy for Tracking of Geosynchronous Satellites”. IETE Journal of Research. 30. 103-108. 10.1080/03772063.1984.11453249,1984

[3] A. Pirhadi, M. Hosseini and M. Hakkak, “A novel implementation of GEO satellite step track subsystem”. Iranian Journal of Electrical and Computer Engineering (IJECE). 4. 71-76, 2005

[4] S. Kawase, N. Kawaguchi, T. Tanaka and K. Tomita, "Optical Calibration of Geostationary Satellite Tracking Systems," IEEE Transactions on Aerospace and Electronic Systems, vol. AES-17, no. 2, pp. 167-172, March 1981, doi: 10.1109/TAES.1981.309142.

[5] S. Kawase, "Radio interferometer for geosynchronous-satellite direction finding", IEEE Transactions on Aerospace and Electronic Systems, vol.43, no.2, pp.443-449, 2007.

[6] D. Roddy, “Satellites Communications”, Mcgrow-Hill, 2001.

[7] S.V. Devika, K. Karki, S. Kotamraju, K. Korada and M.Z.U. Rahman, “A new computation method for pointing accuracy of cassegrain antenna in satellite communication”, Journal of Theoretical and Applied Information Technology. 95. 3062-3074, 2017

[8] İ. Oz, Ü.C. Yılmaz, “Determination of Coverage Oscillation for Inclined Communication Satellite”. Sakarya University Journal of Computer and Information Sciences (2020). 2. 973-983. DOI: 10.16984/saufenbilder.702190.

[9] H. Yuanzhi and C. Ma. "Analysis of the Effect of Antenna Pointing Error Caused by Satellite Perturbation on Space Terahertz Communication" Applied Sciences 12, no. 21: 10772. https://doi.org/10.3390/app122110772, 2022

[10] Z. Peng, R. Bin, W. Youping, F. Yongqiang, J. Bo and Z. Weibo, "The effect of orbital elements on GEO satellite communication quality," 2012 IEEE Aerospace Conference, Big Sky, MT, USA, pp. 1-6, doi: 10.1109/AERO.2012.6187120, 2012

[11] S. Mrak, U. Kuhar and A. Vilhar, "Low-Cost system design for tracking satellites in geosynchronous orbit," 2015 9th European Conference on Antennas and Propagation (EuCAP), Lisbon, Portugal, 2015, pp. 1-5.

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/

DOI: 10.35377/saucis...1233094 RESEARCH ARTICLE

Analysis of Urine Sediment Images for Detection and Classification of Cells

Hilal Atıcı[1], H. Erdinç Koçer[2], Abdullah Sivrikaya[3], Mehmet Dağlı[3]

[1]Department of Computer Engineering, Selcuk University, Konya / Türkiye
[2]Department of Electrical and Electronics Engineering, Selcuk University, Konya / Türkiye
[3]Department of Biochemistry, Selcuk University Faculty of Medicine, Konya / Türkiye

Corresponding author:
Hilal Atıcı, Department of Computer Engineering, Selcuk University
E-mail address: hilalatici@eskisehir.edu.tr

Received: 17 January 2023
Revised: 02 March 2023
Accepted: 17 March 2023
Published Online: 30 April 2023

Citation: Atıcı H. et al. (2023). Analysis of Urine Sediment Images for Detection and Classification of Cells. *Sakarya University Journal of Computer and Information Sciences*.6 (1) https://doi.org/10.35377/saucis...1233094

ABSTRACT

Urine sediment tests are important in diagnosing abnormal diseases related to the urinary tract. The formation of cells such as red blood cells and white blood cells in the urine of patients is important for diagnosing the disease. Therefore, cells need to be fully identified in clinical urinalysis. Urinalysis with human eyes; since it is subjective, time consuming and causing errors, methods have been developed to automate microscopic analysis with the help of image processing. In this study, a deep learning algorithm (Yolov7), which gives successful results in image processing technology, was used as a method. The dataset used in the study was created by using microscopic images of urine sediment taken from the Biochemistry Laboratory of the Faculty of Medicine, Selcuk University. Seven different cell segmentation and classification studies have been carried out, including WBC, RBC, WBCC, Epithelial, Flat Epithelial, Mucs and Bubbles, which have clinical value for diagnosing the disease. Experimental studies were carried out with the Yolov7 algorithm, and the results were presented. As a result of the experiment, the urine cell images were segmented into cells using the deep learning method. The segmentation performance metrics, precision, recall, mAP(0.5) and F1-Score(%) were calculated as 0.384, 0.759, 0.432 and 0.510, respectively. The segmented cells were classified as WBC, RBC, WBCC, Epithelial, Flat Epithelial, Mucs and Bubbles and the classification accuracies were obtained as 0.78, 0.94, 0.90, 0.57, 0.92, 0.68 and 0.97, respectively. A mean classification success of 0.822 was achieved for all classes. Thus, it has been seen that the Yolov7 model can be used by experts as a tool for recognizing cells in the urine sediment. Consequently, it has been shown that suitable deep-learning models can be used to recognize the biometric properties of urinary sediment cells. With the model created using deep learning libraries, urine sediment cells can be easily classified, and it is possible to define many different cells if there is a dataset with a sufficient number of images.

Keywords: Deep learning, urine sediment, classification, segmentation

1. Introduction

Today, the microscope is used by technicians in many laboratories to detect cells or parasites. Urine sediment tests are important in diagnosing abnormal diseases related to the urinary tract. The formation of red blood cells, white blood cells, crystals, bacteria, and other microorganisms in the urine sediments of patients is of great importance for diagnosis.

In 2010, the development of an automatic recognition and counting system for visual components of urine sediment was worked on. With the help of image processing technology, image segmentation, representation and definition, geometric properties and texture properties were obtained. The findings of the experiments showed that the ability to learn and classify the neural network could be effectively improved by combining genetic algorithms with appropriate feature selection [1].

In 2013, a study was conducted on the detection and segmentation of red blood cells and white blood cells in urine sediment images. The algorithm process consists of three main parts. The first step is the segmentation of the urine sediment analysis using a Neural Network applied to the HSV color model image. The next step is to remove the noise with morphology operations. The final step is to detect RBCs and WBCs using the Circle Hough Transform. Experimental results showed the mean error percentage of RBC and WBC detection as 5.28 and 8.35, respectively [2].

In a study conducted in 2014; A comprehensive approach was introduced for detecting microscopic urine particles. Microscopic images include RBC, WBC, Calcium oxalate, Triple phosphate and other cells. 16 shape descriptors and 38 textural features were extracted. When K-NN, Neural Network, Naive Bayes, Decision Tree, and Rule Induction classifiers were tested with 10-fold cross-validation, the evaluation resulted in 96.41% accuracy performance with a minimum f measurement of 93.83% using the Neural Network [3].

A 2017 study focused on automatically identifying malaria-infected cells using deep learning methods. Thin blood images were used to compile a dataset of malaria-infected red blood cells and uninfected cells, labeled by a group of four pathologists. Three types of well-known neural networks were evaluated, namely LeNet, AlexNet, and GoogLeNet. The simulation results showed that all these deep convolutional neural networks achieve classification accuracies of over 95%, higher than the approximately 92% accuracy obtained using the support vector machine method. Moreover, deep learning methods have the advantage of being able to automatically learn features from input data, thus requiring minimal input from human experts for automated malaria diagnosis [4].

Urine sediment examination is an important topic in kidney disease analysis and is often a prerequisite for subsequent diagnostic procedures. In a study conducted in 2018, the DFPN (Feature Pyramid Network with DenseNet) method was proposed to overcome the problem of class confusion in urinary sediment examination images. The urine sediment examination dataset contains 42759 labeled samples in a total of 5377 images and includes 7 cell categories: cast, crystals, epithelium, epithelial nucleus, erythrocyte, leukocytes, and mycete. The importance of two parts of the basic model for urinary sediment examination cell detection was investigated. First, adding the attention module at the beginning of the network and the class-specific attention module increased the mAP by 0.7 points with the pre-trained ImageNet model and 1.4 points with the pre-trained COCO model. Next, DenseNet was introduced to the base model (DFPN) for cell detection in urine sediment examination. The DFPN achieved the best result with a mAP of 86.9% on the urine sediment examination test set after balancing between loss of classification and loss of bounding box regression [5].

Urine cast cells are a particularly important examination material in clinical urinalysis for disease diagnosis. Therefore, the exact identification of cells in clinical urinalysis is of great importance. A 2019 study proposes an effective approach for cast cell detection and recognition in urine sediment images. Urine cast cells were used as detection targets in urine microscopy, and then the ResNet50 network was used. Finally, the target area feature maps for classification and localization were entered into the classification subnet and the regression subnet separately, and the detection results were obtained. The data took only 0.2s per image on the NVIDIA Titan X GPU, with the average precision of the recognition result being %89.4 [6].

In 2019, a study was conducted on image classification using microscopic images of urine sediment. A total of 1670 datasets are used for training and testing different convolutional network models in deep learning. These models are VGG 16, VGG 19, Resnet 50, InceptionV3, Xception, Inception-ResnetV2 and MobileNet. Classification of urine sediment has been successfully applied. In the study, the following were evaluated for these models: confusion matrix, loading, feature extraction and average feature extraction times, accuracy, precision, recall and f-scoring. The models that give the best results through these evaluations are Inception V3 and Inception-Resnet V2. The highest accuracy (99.4%) was also obtained with these models. However, MobileNet achieved remarkable results with 98% accuracy, considering its very light size of 16.82 MB compared to Inception-Resnet V2 of 219.93 MB [7].

2. Material and Method

In this study, the Yolov7 model, which is a machine learning-based algorithm that was previously trained with the COCO dataset, was used. To implement the software, Google Colaboratory Pro [8], Tesla T4 graphics card and Pytorch library were used.

2.1 Yolov7 (You Only Look Once) deep learning model

By looking at the images, people can instantly understand which objects are in the images, their positions, and their interactions with each other. The human visual system works very quickly and accurately. Fast and highly accurate algorithms in object detection have allowed computers to drive without special sensors, and assistive devices to transmit real-time scene information to users [9].

Unlike traditional object detection methods, YOLO has taken the finding of bounding boxes, calculation of class probabilities and all other operations as a single regression problem and brought a new view to the field of object detection. With YOLO, it is sufficient to look at the image only once to determine which objects are where on the image. Multiple bounding boxes are estimated simultaneously with a single convolutional network and class probabilities are estimated for each class. This combined model has several benefits over traditional object detection methods.

The YOLO detection model divides each image in training set into SxS(S=7) square cells (grid). If the center position of any objects to be detected is within any of the dividing cells, the cell where the center is located is responsible for detecting that object. Each cell generates the B bounding boxes and estimates the confidence score for these bounding boxes. The

confidence score reflects how confident the model is that the bounding box it produces contains an object and how accurate the probability that the produced bounding box is produced.

$$Confidence\ Score = P_r(Object) * IoU_{pred}^{truth} \quad (1)$$

It is calculated as Pr(Object)∈ (0,1). If there is no object in the dividing cells, the confidence score of those cells is 0, if there is, it is 1. Each cell also estimates the C conditional class probabilities Pr($Class$i $|Object$), as seen in equation 2. These possibilities are linked to cells containing objects. Regardless of the number of bounding boxes produced B, only one of the class probabilities per cell is estimated. Confidence scores are calculated on a class basis for each box by multiplying the conditional class probabilities and individual confidence score estimates during the test [10].

$$P_r(Class_i \mid Object) * P_r(Object) * IoU_{pred}^{truth} = P_r(Class_i) * IoU_{pred}^{truth} \quad (2)$$

Model	Test Size	AP^{test}	$AP_{50}{}^{test}$	$AP_{75}{}^{test}$	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

Figure 1 YOLO Models [11]

Yolo v7 is the latest version of YOLO. Yolo v7 was released on July 7, 2022, by Wong Kin Yiu. This new version is Yolo's most accurate and fast real-time object detection version available. The Yolo frame has three parts, the Spine, the head and the neck. The spine detects important information in the image and sends this information from the neck to the head. The neck compiles information from the spine and creates feature pyramids. The head consists of the output layers and forms the last part of the structure. Yolov7 is not limited to one head. The auxiliary head assists the training in the middle tiers, while the auxiliary head is responsible for the final output. Additionally, to improve deep network training, a tag assignment mechanism was introduced that takes into account precision and network prediction results and then assigns tags. Among all real-time object detectors with 30 Fps or more, Yolov7 achieved the highest accuracy (56.8% AP). YOLOv7 was trained completely from scratch using the MS COCO [13] dataset without using any pre-trained weights [12].

2.2 Urine Sediment Dataset

This study was approved by the ethics committee of Selcuk University. (Number:2022/192, Date:12.04.2022). In this study, a dataset consisting of a total of 9,004 microscopic images of urine sediment obtained by using the DIRUI FUS-2000 device in the Biochemistry Laboratory of Selcuk University Medical Faculty Hospital was used. Segmentation and classification of cells found in microscopic urine sediment images in the dataset were emphasized. For this reason, each cell in the microscopic images in the dataset was labeled with the help of experts working in the Biochemistry Laboratory of Selcuk University Faculty of Medicine.

The dataset consists of seven classes: White Blood Cell (WBC), White Blood Cell Cluster (WBCC), Red Blood Cell (RBC), Epithelial, Flat Epithelial, Mucs and Bubbles. The images are 800 x 600 pixels in size in bmp format and have been labeled with the help of an expert. The images were obtained from the urine samples of the patients who came to the biochemistry laboratory. The obtained high-quality labeled dataset can be applied to machine learning and deep learning models to recognize different urine cell types, and the models can be trained and tested.

In order to apply the dataset to the Yolo algorithm, firstly, the labeling process was performed. The data labeling process was done with the MakeSense tool [14], which can be labeled online. The labeling process is important in terms of giving deep learning algorithms to the training set, where it can distinguish the desired objects and train itself.

The images in the Data Set were uploaded separately and the necessary labels were made. Seven classes in the dataset were added as labels. The cells in each image are marked with a rectangle and the class label to which they belong is selected. An image can also contain more than one cell. After all the labeling is done, the labeling file is exported. Since the dataset tag information uses tagging in Yolo format, it is exported in VOC XML format.

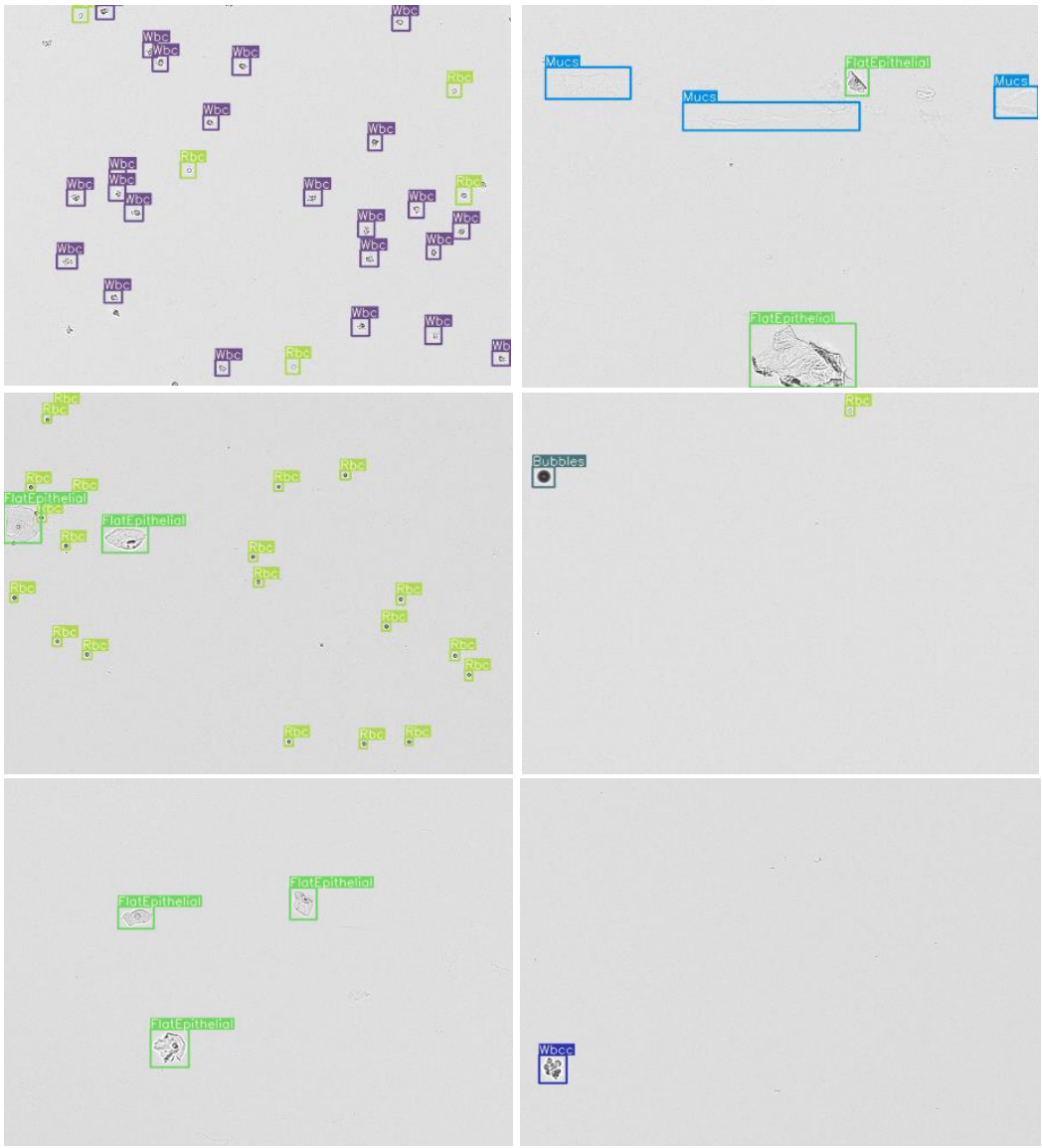


Figure 2 Labeled image examples.

2.2.1 Types of cells found in the data set

The dataset consists of seven classes: White Blood Cell (WBC), White Blood Cell Cluster (WBCC), Red Blood Cell (RBC), Epithelial, Flat Epithelial, Mucs and Bubbles.

Tablo 1 Cell Numbers and Types in each group

Cell Type	**Number of Cells**	**%**
RBC	16124	59,64
WBC	5980	22,12
Flat Epithelial	3574	13,22
Mucs	468	1,73
Epithelial	366	1,35
WBCC	309	1,14
Bubbles	212	0,78
Total	27,033	100

Leukocytes (White Blood Cells – WBC)

Leukocytes, known as white blood cells, play an active role in the body's fight against foreign substances, microbes and infectious diseases. A high incidence of these cells may indicate the presence of an infection or other underlying inflammatory medical problems associated with it.

Pathologically in the urine; more than normal leukocytes are seen in cases of urinary tract infection, kidney stone, kidney infection, urinary tract obstruction, blood diseases such as sickle cell anemia, pregnancy, some cancers such as bladder, kidney or prostate cancer, and consumption of some painkillers [15].

Red Blood Cells (RBC)

The presence of RBCs in the urine means that there is blood in the urine. Blood in the urine is medically referred to as hematuria. Although the sight of erythrocytes in the urine may sound scary, it is usually not life-threatening and can rarely be diagnosed due to serious illness. The presence of blood in the urine may be so small that it can only be diagnosed by looking under a microscope. But when it occurs, it is important to investigate the cause, because sometimes it can occur from several serious conditions [16]. Color change may occur in the urine due to some vitamin drugs or the consumption of carrots or citrus fruits. However, if there is visible bleeding in the urine, a physician should be consulted without delay and the treatment plan should be formed after the effect causing bleeding is found. It is quite normal for a normal person to have 0-3 erythrocytes in their urine. But if it is seen on it, it is useful to investigate the cause.

Among the causes of erythrocytes in the urine; having had a urinary tract infection, infection in the urinary bladder or kidneys, stones in the kidneys, bladder cancer, kidney cancer, glomerulonephritis, damage to the kidney, enlarged prostate, prostate cancer, tumor in the urinary tract, some painkillers, cancer drugs, kidney diseases that develop due to a genetic disease such as sickle cell anemia, inflammations caused by viruses, excessive exercise, blood clots can be counted [16].

Epithelial

Epithelial cells; are cells on the surface of various tissues of the body, especially blood vessels, organs, skin and urinary tract. Epithelial cells act as a barrier between the inside and outside of the body and protect the body against pathogens from outside. The surface of the urinary tract is also covered with epithelial cells. For this reason, it is considered normal to encounter a small number of epithelial cells in urinalysis. If the number of epithelial cells is less than 15-20 in the urine test, the epithelial cell test is considered normal in this urine test [17]. These parameters alone do not make sense. For this reason, a meaningful result can be obtained by looking at other parameters such as erythrocytes, leukocytes, parasites, and crystals in the blood.

Squamous Epithelium: It is the most common epithelial cell found in the urine. They are very large (largest cell seen in urine microscopy), small (very small compared to the cell) and mononuclear, with coarse cytoplasm, and flat, irregular borders. In some, double nuclei can be seen [18].

Renal Tubuler Epithelium: It is more common in urine with high protein content and casts. It is small, round, and polygonal. It is slightly larger than leukocytes and has a larger nucleus than granular and squamous epithelial cells. It is very similar to leukocytes in low-density urine [18].

Pathologically in the urine; In cases such as urinary tract infection, kidney stones, kidney infection, fungal infections, and bladder cancer, more epithelial cells are seen than normal [17].

Mucs

Mucus is a protective substance that covers and moistens the surfaces of structures such as the nose, trachea, urinary tract and stomach. It is normal to have a small number of mucus cells in the urine, while too many may indicate a urinary tract infection or other medical condition. The amount of mucus in the urine is determined by the urine test taken from the patient [19].

In the case of a normal discharge, little or moderate mucus cells are found in the urinalysis. A large number of mucus cells found in the urine may indicate medical conditions such as urinary tract infection, irritable bowel syndrome, kidney stones, and bladder cancer [19].

WBCC

It is formed by the coexistence of leukocytes, known as white blood cells. It is defined as leukocyte aggregation. Clusters of leukocytes are often seen in the urine of patients with urinary tract infections.

Bubbles

These cells are peculiar large cells with mononuclear cells that appear to contain one or more fluid-filled blebs. Air bubbles are very common due to air being trapped between the slide and the coverslip [20].

2.3 Performance measurements

There are many measurement criteria used to evaluate the performance of an object detection model. IoU (Intersection over Union), Precision, Recall, Average Precision (AP) and mean Average Precision (mAP) can be given as examples of the most used criteria.

IoU: IoU is an evaluation metric measuring the similarity between Ground Truth and model prediction. The metric is calculated as given in equation (3).

$$IoU = \frac{Area\ of\ Union}{\text{Area of intersection}} \tag{3}$$

Precision: Precision is calculated by dividing the correctly predicted positive samples against all predicted positive samples. The metric is calculated as given in equation (4).

$$Precision = \frac{TP}{(TP+FP)} \tag{4}$$

Recall: Recall is obtained by dividing the correctly predicted positive samples by all samples in the real class. The metric is calculated as given in equation (5).

$$Recall = \frac{TP}{(TP+FN)} \tag{5}$$

Average Precision (AP): AP is measurement metric that includes precision and recall metrics used to evaluate object detection performance. It is a number metric that summarizes the Precision-Recall curve by averaging the recall values from 0 to 1. The metric is calculated as given in equation (6).

$$AP = \frac{1}{11}\sum_{r\in(0,0.1,0.2,\ldots,1)} P_{interp(r)} \tag{6}$$

mAP: The mAP value is obtained by summing the APs of each class and dividing by the number of classes. The metric is calculated as given in equation (7).

$$mAP = \frac{1}{M}\sum_{j=1}^{M} AP_j \tag{7}$$

3. Experimental Results and Discussion

In this study, the machine learning-based deep learning-based YOLO algorithm, which was previously trained with the COCO dataset, was used. The YOLO algorithm is one of the one-step object detectors. There are six different models of Yolov7: Yolov7, Yolov7-X, Yolov7-W6, Yolov7-E6, Yolov7-D6 and Yolov7-E6E. In this study, the Yolov7 model was used. Because the Yolov7 model was trained with the COCO dataset and produced better results than other models. The Yolov7 model produced 161 fps (frame per second) and 2.8 ms results, surpassing other models. To implement the software, Google Colaboratory, Tesla T4 graphics card and Pytorch library, which provide access to powerful GPUs and do not require configuration, were used. In addition, to determine the error rates at the end of the training, the error rates were observed according to the iterations by using the Tensorboard graphical interface in the Tensorflow library. The training was completed after ***5 days 5 hours 29 minutes***.

Many experimental studies have been conducted to verify the performance of the YOLO-based model, and the results and findings have been analyzed. The training details used for the YOLOv7 model used in this study are presented in Table 2. These settings have been determined with the best results based on testing in the experimental studies conducted. In this study, various improvements have been made for test reviews and training by using the model developed on the open-source framework.

Table 2 Training Details of the Yolov7 Model

İnput Size	Model Parameters	Software Language	Environment	Library	Epoch	Batch Size	Optimizer	Activation Function
800x600	37228920	Python	Google Colaboratory	Torch	300	8	SGD	Leaky RELU & Sigmoid

In this study, the dataset used to train the network was created using microscopic images of urine sediment. To identify the cells in the training set of the network, the dataset was labeled by referring to the expert knowledge. The dataset is divided into three as training, validation and test set. The dataset consists of 9,004 images in total. These images contain a total of 27,033 cells.

The data set contains images of urine sediment and .xml files containing the location information of the cells in that image. However, to train the Yolov7 model, it is necessary to edit the dataset structure. Therefore, images (.jpg, .png etc.) and tags must be converted to .txt. In the format of Yolov7 tag (.txt) files, there is a line for each object, each line consists of class, x_center, y_center, width and height data. The box coordinates surrounding the object must be in the normalized x, y, w, h format (0–1). The classes in the dataset are zero-indexed. Therefore, the data has been converted to the Yolov7 tag file format.

Table 3 Comparison of performance metric results such as mAP, P and R obtained with the Yolov7 model for the validation set.

Class	Precision	Recall	mAP@.5
WBC	0.202	0.874	0.298
RBC	0.252	0.826	0.277
WBCC	0.296	0.583	0.264
Epithelial	0.395	0.507	0.419
FlatEpithelial	0.537	0.985	0.650
Mucs	0.264	0.594	0.270
Bubbles	0.744	0.946	0.848
All	0.384	0.759	0.432

Table 4 Comparison of performance metric results such as mAP, P and R obtained with the Yolov7 model for test set.

Class	Precision	Recall	mAP@.5
WBC	0.220	0.844	0.293
RBC	0.253	0.793	0.276
WBCC	0.303	0.483	0.265
Epithelial	0.438	0.479	0.420
FlatEpithelial	0.545	0.982	0.651
Mucs	0.299	0.479	0.257
Bubbles	0.760	0.865	0.840
All	0.403	0.704	0.429

P (Precision), R (Recall) and mAP (Average Precision) values give information about whether our model is performing well. The weight values obtained as a result of the training were recorded. These saved weights can be used later. For the experiment, the Yolo model was carried out for the segmentation of cells in the Urine Sediment image dataset. At the end of the experiment, Precision, Recall, mAP (0.5) and F1-Score(%)performance metrics were calculated as 0.384, 0.759, 0.432 and 0.510, respectively.

Table 5 Classification performance metric results with the Yolov7 model for test set

Class	WBC	RBC	WBCC	Epithelial	FlatEpithelial	Mucs	Bubbles	All
%Accuracy	%78	%94	%90	%57	%92	%68	%97	%82

In experimental studies carried out with the YOLO model; Classification accuracy for WBC, RBC, WBCC, Epithelial, FlatEpithelial, Mucs and Bubbles cells was calculated as %78, %94, %90, %57, %92, %68 and %97, respectively. A mean classification success of %82 was achieved for all classes. Obtained results are presented in Table 5.

Considering the results obtained, the detection accuracy of epithelial and mucus cells is lower than other cells. This is because; cell numbers are less, the colors of the cells are pale, and the edges are not clear and sharp. In particular, mucus cells are often confused with the background, as they have a flat and indistinct structure as seen in the pictures. The epithelial cell is

similar to Wbc cells, only larger in size, so it is mixed with Wbc cells. This problem can be solved by using more labeled mucus and epithelial cells.

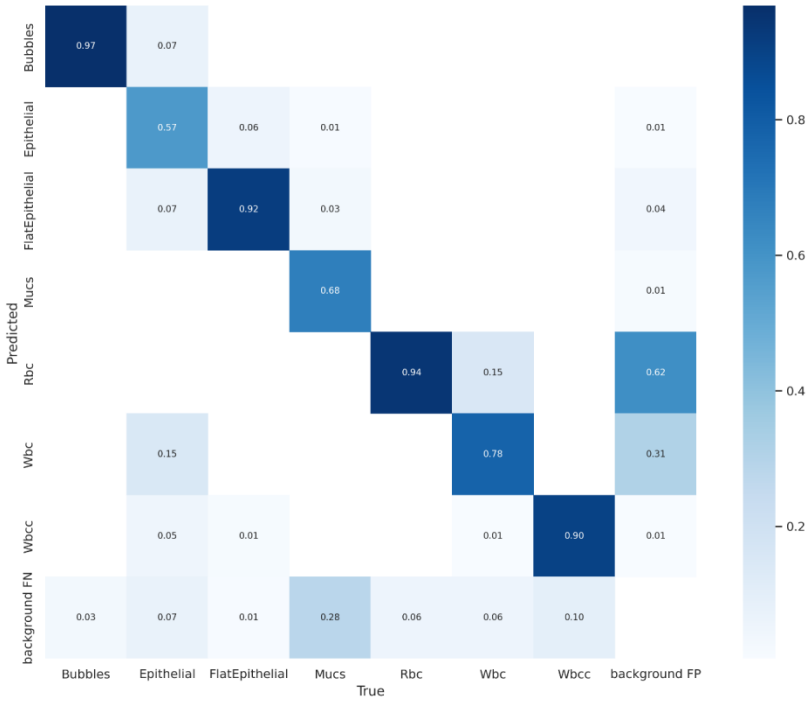


Figure 3 Confusion Matrix for Test Set

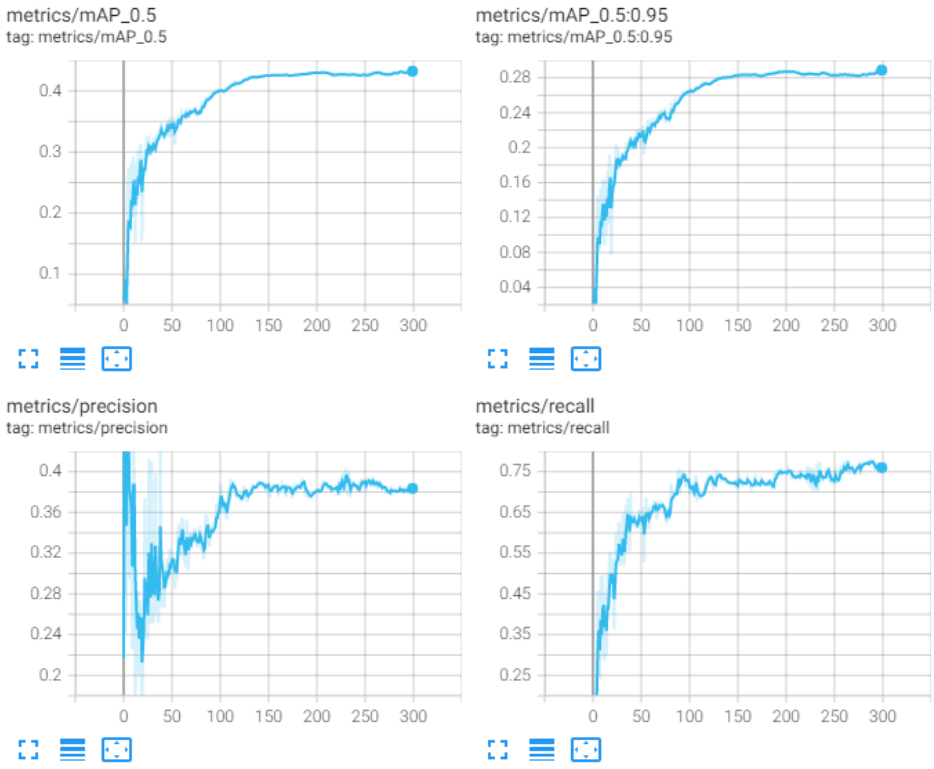


Figure 4 Plots of mAP_0.5, mAP_0.5:0.95, precision and recall values according to the epoch number during the training of the YOLOv7 network

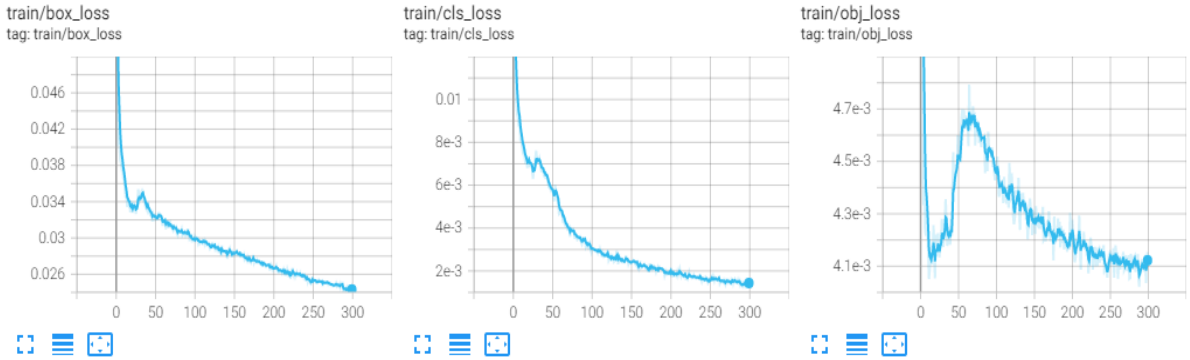


Figure 5 Plots of Box Loss, Class Loss and Object Loss values according to the epoch number during the validation of the YOLOv7 network

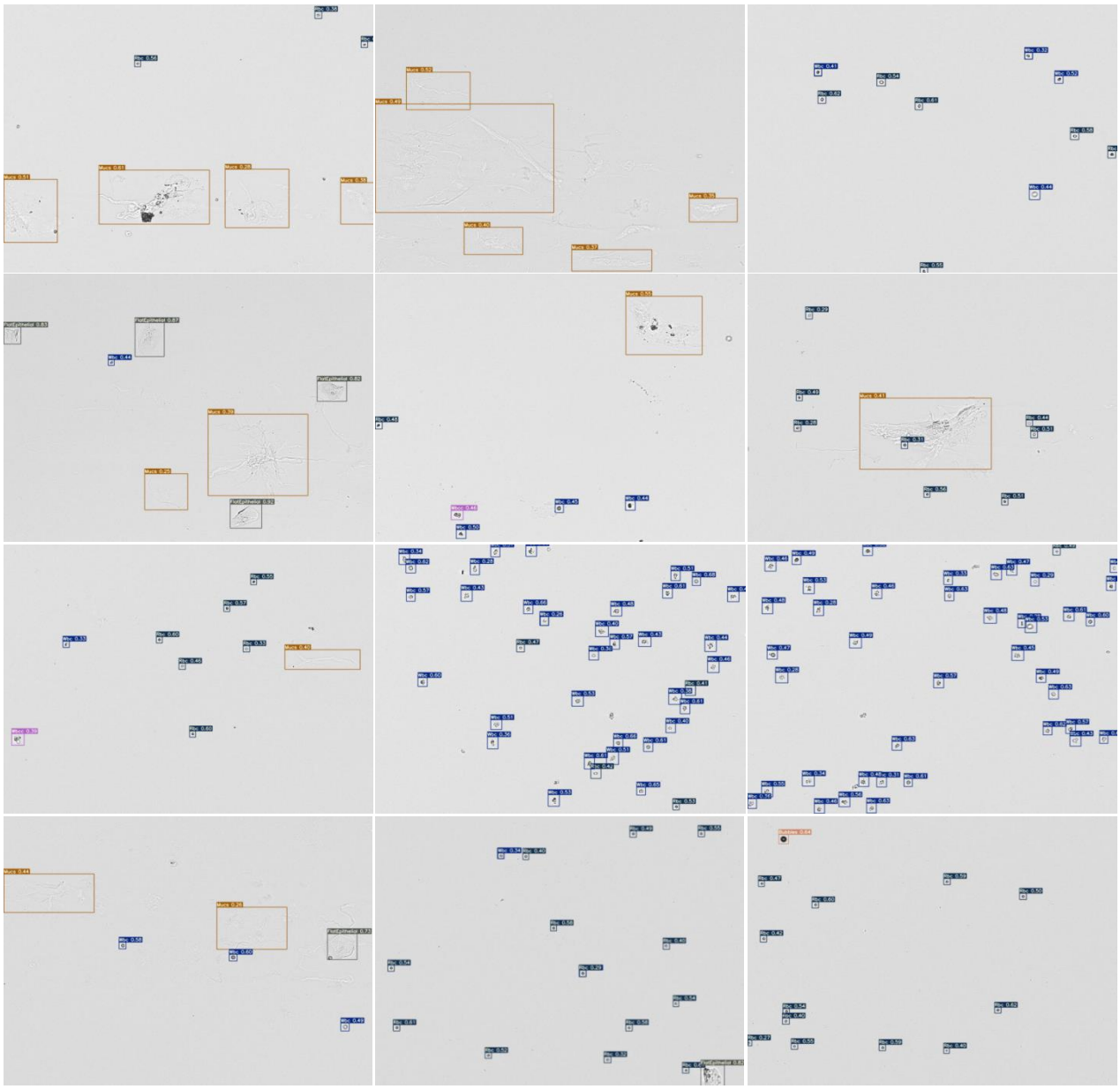


Figure 6 Test Image Samples

4. Conclusions

Analysis of microscopic images is used in many fields of medicine. Today, the microscope is used by technicians in many laboratories and anomalies (defects in the cell, parasites, low/excess cell count, etc.) are detected. Urine sediment tests are important in diagnosing abnormal diseases related to the urinary tract. The formation of red blood cells and white blood cells etc. in the urine sediments of patients is of great importance for diagnosing the disease. In patients' urine sediments, red blood cells, white blood cells, etc. the formation of cells is of great importance for diagnosing the disease. Therefore, cells need to be fully identified in clinical urinalysis. Urinalysis with human eyes; Since it is subjective, time consuming and causing errors, methods have been developed to automate microscopic analysis with the help of computer and software systems. In this study, the Yolov7 algorithm, which gives successful results in image processing technology, was used as a method and model. The dataset used in the study was obtained from the urine sediment microscopic images taken from the Biochemistry Laboratory of the Faculty of Medicine, Selcuk University. Studies have been conducted on seven different cell segmentation and classification, clinically valuable WBC, RBC, WBCC, Epithelial, Flat Epithelial, Mucs and Bubbles.

The contributions of this study can be summarized as follows. (1) It has been shown that urine sediment cells can be successfully classified with a deep-learning approach. (2) It has been observed that the Yolov7 model, which is frequently preferred in image processing applications, can be used as a means of recognizing cells in the urine sediment. (3) It has been seen that the labeling process is an important step in deep learning and recognition applications. (4) Since epithelial cells are similar to WBC cells, recognition success was low. Mucus cells, on the other hand, are perceived as background fluid due to their wide and diffuse appearance. Recognition percentage can be increased by increasing the number of sample images of cells with low classification success.

References

[1] X. Zhou, X. Xiao, and C. Ma. “A study of automatic recognition and counting system of urine-sediment visual components. ” *2010 3rd International Conference on Biomedical Engineering and Informatics*. Vol. 1. IEEE, 2010.

[2] W. Tangsuksant *et al.,* “Development algorithm to count blood cells in urine sediment using ANN and Hough Transform.” *The 6th 2013 Biomedical Engineering International Conference*. IEEE, 2013.

[3] M. D. Almadhoun and Alaa El-Halees. “Automated recognition of urinary microscopic solid particles.” *Journal of medical engineering & technology 38.2* (2014): 104-110.

[4] Y. Dong *et al.,* “Evaluations of Deep Convolutional Neural Networks for Automatic Identification of Malaria Infected Cells”, *2017 IEEE EMBS international conference on biomedical & health informatics (BHI).* IEEE, 2017.

[5] Y. Liang *et al.,* "Object detection based on deep learning for urine sediment examination.” *Biocybernetics and Biomedical Engineering 38.3* (2018): 661-670.

[6] Q. Li *et al.,* “A recognition method of urine cast based on deep learning.” 2019 International Conference on Systems, Signals and Image Processing (IWSSIP). IEEE, 2019.

[7] J.S. Velasco, M.K. Cabatuan, and E.P. Dadios, “Urine sediment classification using deep learning.” *Lecture Notes on Advanced Research in Electrical and Electronic Engineering Technology (2019)*: 180-185.

[8] Colab Pro, Available: https://colab.research.google.com/. [Accessed: 21.11.2022].

[9] J. Redmon *et al.,* “You only look once: Unified, real-time object detection.” *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[10] P. F. Felzenszwalb *et al.,* “Object detection with discriminatively trained part-based models.” *IEEE transactions on pattern analysis and machine intelligence 32.9* (2009): 1627-1645.

[11] Chien-Yao Wang *et al.,* “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv preprint arXiv:2207.02696 (2022).

[12] S. Ren *et al.,* “Faster r-cnn: Towards real-time object detection with region proposal networks.” *Advances in neural information processing systems 28* (2015).

[13] COCO Dataset. Available: https://cocodataset.org/#home, [Accessed: 20.05.2022].

[14] P. Skalski, “Make Sense,” 2019. Available: https://github.com/SkalskiP/make-sense/. [Accessed: 20.05.2022].

[15] Lökosit, Available: https://www.medicalpark.com.tr/lokosit/hg-2070. [Accessed: 21.11.2022].

[16] İdrarda RBC, Available: https://www.acibadem.com.tr/ilgi-alani/eritrosit-rbc/#genel-tanitim. [Accessed: 21.11.2022].

[17] İdrarda Epitel Nedir? Değeri Kaç Olmalı? Nedenleri ve Tedavisi, Available : https://saglik.li/idrarda-epitel-nedir/, [Accessed: 21.11.2022].

[18] Hücreler, Available: https://www.mustafaaltinisik.org.uk/idrar/turkce/hucre.htm, [Accessed: 21.11.2022].

[19] İdrarda Mukus Testi Nedir? Ne İşe Yarar?, Available: https://www.probiyotix.com/idrarda-mukus-testi-nedir-ne-ise-yarar/, [Accessed: 21.11.2022].

[20] S. E. Pambuccian, “Bedside Urinary Microscopy Giovanni Battista Fogazzi Lectures Series”, *Urinary Sediment: Part 6 and last: Contaminants and funny findings*, Milan, İtaly, March 8th, 2007.

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/

DOI: 10.35377/saucis...1259584 RESEARCH ARTICLE

Ischemia and Hemorrhage detection in CT images with Hyperparameter optimization of classification models and Improved UNet Segmentation Model

Mehmet Okuyar[1], Ali Furkan Kamanlı[1]

[1]Sakarya University of Applied Sciences, Faculty of Technology, Electrical and Electronics Engineering, Sakarya/Türkiye

Corresponding author:
Ali Furkan Kamanli, Sakarya University of Applied Sciences, Faculty of Technology, Electrical and Electronics Engineering, Sakarya/Türkiye
E-mail address:
fkamanli@sakarya.edu.tr

Received: 03 March 2023
Revised: 15 March 2023
Accepted: 22 March 2023
Published Online: 30 March 2023

Citation: Okuyar M., Kamanlı A. F. (2023). Ischemia and Hemorrhage detection in CT images with Hyperparameter optimization of classification models and Improved UNet Segmentation Model. *Sakarya University Journal of Computer and Information Sciences*. 6 (1). https://doi.org/10.35377/saucis...1259584

ABSTRACT

Deep learning is a powerful technique that has been applied to stroke detection using medical imaging. Stroke is a medical condition that occurs when the blood supply to the brain is interrupted, which can cause brain damage and other serious complications. Stroke detection is important to minimize damage and improve patient outcomes. One of the most common imaging modalities for stroke detection is CT (Computed Tomography). CT can provide detailed images of the brain and can be used to identify the presence and location of a stroke. Deep learning models, particularly convolutional neural networks (CNNs), have shown promise for stroke detection using CT images. These models can learn to automatically identify patterns in the images that are indicative of a stroke, such as the presence of an infarct or hemorrhage. Some examples of deep learning models used for stroke detection in CT images are U-Net, which is commonly used for medical image segmentation tasks, and CNNs, which have been trained to classify brain CT images into normal or abnormal. The purpose of this study is to identify the type of stroke from brain CT images taken without the administration of a contrast agent, i.e., occlusive (ischemic) or hemorrhagic (hemorrhagic). Stroke images were collected, and a dataset was constructed with medical specialists. Deep learning classification models were evaluated with hyperparameter optimization techniques. And the result is segmented with an improved Unet model to visualize the stroke in CT images. Classification models were compared and VGG16 achieved %94 success. Unet model achieved %60 IOU and detected the ischemia and hemorrhage differences.

Keywords: stroke detection; CT image; Deep learning; medical image processing; segmentation

1. Introduction

Stroke detection is a critical area of research in medical artificial intelligence. In recent studies, deep learning algorithms have been explored to predict hematoma expansion from non-contrast computed tomography (NCCT) scans through external validation [1]. One novel CNN, SkullNetV1, uses CNN for feature extraction and a lazy learning approach to classify five types of skull fractures from brain CT images [2]. Another study formulated intracranial hemorrhage (ICH) detection as a problem of multiple instance learning (MIL), which enables training with only scan-level annotations [3]. Deep learning-based automated analysis of CT scan slices has been proposed for detecting various levels of brain hemorrhages [4]. A combination of deep learning and machine learning classification algorithms has been used to establish an explainable COVID-19 detection system using CT scans and chest X-rays [5]. The proposed research focuses on several deep transfers learning-based CNN approaches for detecting COVID-19 in chest CT images, using foundation models such as VGG16, VGG19, Densenet121, InceptionV3, Xception, and Resnet50 [7]. A method for detecting pulmonary nodules based on multiscale fusion has been shown to have a higher detection rate for small nodules and improve the classification performance of true and false-positive nodules [8,9], with competitive performance in terms of sensitivity compared to state-of-the-art methods [10].

There are also studies proposing deep learning-based methods for CT [11,12], proposing fully automated lesion detection and segmentation systems on whole-body PET/CT scans [13,14], and developing effective segmentation techniques based on deep learning algorithms for optimal identification of regions of interest and segmentation [15-17]. Other influential work includes developing a deep learning model capable of segmenting IVCF from CT scan slices along the axial plane [18] and showing a deep learning model that segments acute ischemic stroke on NCCT at a level comparable to neuroradiologists [19]. Medical images differ from natural images in many ways, and a domain expert should be consulted to assess the model's performance. Detecting stroke in CT images is a challenging task due to its nature. It is important to verify the outcomes of deep learning models on different datasets and with different preprocessing methods, as the models' performance may vary depending on these factors.

The purpose of this study is to identify the type of stroke from brain CT images taken without the administration of a contrast agent, i.e., occlusive (ischemic) or hemorrhagic (hemorrhagic). A deep learning model was used to visualize the important area (ischemia and/or bleeding site) on the CT image, and classification algorithms were used. In addition, a study was carried out on determining the stroke region with the segmentation structure. The most successful classification model accuracy was 94 % and the improved UNET segmentation model IOU metric accuracy was 60 %. The paper is organized as follows, Materials and Methods, Transfer learning, Data Augmentation, Segmentation Mode, Results and Discussion and Conclusion.

2. Materials and Methods

The quantity of the training dataset and the variability of the data are key factors in the network's success. For deep learning models to function well, large datasets are required. Data augmentation and transfer learning techniques were applied. The data was used from Kaggle (Brain MRI dataset) and collected from the hospitals and labeled with the medical professionals.

Due to its structure, learning transfer removes the requirement for additional data and improves model performance by cutting down on learning time. As pre-trained networks already modify weights by learning from many data, they are known to need less input than networks trained from the start.

To create a project on stroke detection using deep learning classification and segmentation models, several steps need to be taken:

- Assembling a database of medical CT scans including unaffected and photos showing the effects of a stroke. Having a varied dataset that encompasses various stroke types and imaging circumstances is crucial.
- Images need to be resized or normalized, and the dataset might be divided into training, validation, and test sets.
- Training the model on a custom dataset, the training needs to be adjusted to the model's parameters to accurately classify and segment the images.
- Evaluating the model's performance using the validation and test sets. To observe how well the model can classify and segment new images that have not been seen before.
- For the model to correctly categorize and segment the images, the parameters need to be adjusted using the training set (Hyperparameter optimization).

2.1 Transfer learning

Transfer learning is a technique in machine learning that allows a model trained on one task to be used for a different but related task. In the context of medical imaging, transfer learning can be used to apply a model trained on a large dataset of general images to a smaller dataset of medical images.

There are two main ways to use transfer learning for medical imaging:

- Feature extraction: In this approach, a pre-trained model extracts features from the medical images. The extracted features are then used as input to a separate classifier to recognize specific medical conditions. This allows the model to take advantage of the pre-trained model's ability to detect low-level features, while still allowing the classifier to learn the specific characteristics of the medical images.

- Fine-tuning: In this approach, a pre-trained model is used as a starting point and is further trained on a medical imaging dataset. This allows the model to adapt to the specific characteristics of the medical images, while still taking advantage of the knowledge learned from the pre-trained model.

Transfer learning can be useful in medical imaging because it allows for the training of models with a smaller amount of data, which is often a limitation in the medical field. Additionally, transfer learning can also help to improve the performance of models by leveraging the knowledge learned from pre-trained models on large datasets.

Even with transfer learning, it's still important to have a diverse and high-quality dataset, as well as to evaluate the model's performance and ensure that it generalizes well to new cases. In this study, classification models combined the feature extraction and fine-tuning methods.

2.2. Data augmentation

Furthermore, a study evaluated various methodologies, deep learning architectures, approaches, bioinformatics, specified function requirements, monitoring tools, artificial neural network (ANN) algorithms, data labeling, and annotation algorithms that control data validation, modeling, and diagnosis of different diseases using smart monitoring health informatics applications [23]. Another study proposed an end-to-end Generative Adversarial Network (GAN) architecture capable of generating high-resolution 3D images [24].

Additionally, an Extreme Gradient Boosting (XGBoost) algorithm was developed to classify four subtypes of brain tumors: normal, gliomas, meningiomas, and pituitary tumors [25]. Tumor segmentation is a specific task that requires clinicians to label every slice of volumetric scans for each patient, which can become impractical for training neural networks with a large dataset. To address this issue, a novel semi-supervised framework was proposed to train any segmentation model using only the presence of a tumor in the image, as well as a few annotated images [26]. The training pipeline of the dataset included histogram equalization and data augmentation [29,30]. Data augmentation is a machine learning technique that applies various transformations to existing data, such as flipping, rotation, scaling, and cropping, to artificially increase the size of the dataset. In medical imaging, data augmentation can be used to increase the training data available for a model, and thus improve its performance.

There are several reasons why data augmentation is important in medical imaging:

- Small datasets: Medical imaging datasets are often small in size due to the high cost and complexity of acquiring medical images. Data augmentation can help to overcome this limitation by artificially increasing the size of the dataset.
- Variability: Medical images can vary greatly depending on the imaging modality, patient population, and imaging conditions. Data augmentation can help to increase the diversity of the dataset and make the model more robust to these variations.
- Overfitting: Deep learning models can easily be overfit to the training data, resulting in poor performance on new data. Data augmentation can help to reduce overfitting by introducing additional variations in the training data.

Common data augmentation techniques used in medical imaging include flipping, rotation, scaling, translation, shearing, and adding noise. Additionally, it is important to keep in mind that data augmentation should be applied carefully and with consideration of the specific characteristics of the medical images, as well as the medical condition being analyzed, to avoid creating unrealistic or misleading images.

The data augmentation techniques applied should be chosen carefully and considering the specific characteristics of the medical images, as well as the medical condition being analyzed, to avoid creating unrealistic or misleading images. Additionally, it is important to consider the regulatory requirements for your project, such as HIPAA compliance, as well as ethical considerations.

2.3. Classification model

Using pre-trained models such as VGG16, InceptionV3, DenseNet, and Xception for medical image classification is a common approach in deep learning. These models have already been trained on large image datasets and can be fine-tuned for a specific medical imaging task. Here is a general outline of the process:

- First, a dataset of medical images labeled with the appropriate class labels.
- Fully connected layers need to be removed, which is used for the original image classification task the model was trained on.
- Then, a fully connected layer with the number of neurons corresponding to the number of classes was added for our study.
- Fine-tune the model by training on a medical image dataset. This can be done by "freezing" the weights of the pre-trained layers and only training the added fully connected layer.

A big, diversified dataset with high-quality photos might be challenging to get in the field of medical imaging. Additionally, it is preferable to have a domain specialist assess the model's performance because medical images differ from natural photos in many ways. Therefore, the dataset was constructed with radiologists.

2.4. Segmentation model

U-Net style convolutional neural networks (CNNs) are a popular choice for the segmentation of medical images. The U-Net architecture is designed specifically for image segmentation, with a contracting path (downsampling) and an expansive path (upsampling). The contracting path is based on a traditional CNN, while the expansive path uses a transposed CNN (deconvolution) to increase the spatial resolution of the feature maps. The two paths are connected via skip connections, which concatenate feature maps from the contracting path with corresponding feature maps from the expansive path. By leveraging information from earlier layers in the contracting path, the U-Net architecture can achieve more precise segmentation results.

Improved UNet is an enhanced version of the original UNet architecture. It aims to improve the performance of the original UNet by incorporating some novel techniques such as:

- Attention Mechanism
- Multi-Scale Feature Fusion
- Residual Connection
- Spatial Dropout
- Batch Normalization
- Weighted Cross-Entropy Loss

These changes help to improve the accuracy and stability of the model. In this study, improved UNet was trained and tested with parameter optimizations.

2.5. Hyperparameter optimization

Hyperparameter optimization is the process of finding the best set of hyperparameters for a given model and dataset. Hyperparameters are parameters that are not learned during the training process, but rather set before the training process begins. Examples of hyperparameters in the UNET model include learning rate, batch size, and number of filters in each layer.

One possible approach to hyperparameter optimization is grid search, which involves evaluating the model's performance for all possible combinations of hyperparameters in a predefined range. Another approach is random search, which involves sampling hyperparameters from a distribution. In practice, it is often useful to use a combination of these methods, along with techniques such as early stopping and cross-validation, to find the best set of hyperparameters for a given task. Pixel-level thresholding is a post-processing technique used in image segmentation tasks to improve the accuracy of the segmentation. In this technique, a threshold value is applied to the predicted probabilities to binarize the output image. The threshold value can be set based on the distribution of the predicted probabilities, or it can be optimized using a validation set. By setting an appropriate threshold value, the output image can be refined to better separate the foreground and background.

In addition to the weighted cross-entropy loss function, there are several other loss functions that can be used in the UNET model for image segmentation tasks. These include Dice Loss: This loss function measures the overlap between the predicted and true segmentation masks using the Dice coefficient. Jaccard Loss: This loss function measures the similarity between the predicted and true segmentation masks using the Jaccard coefficient. Focal Loss: This loss function is designed to give more weight to hard examples in the training data by down-weighting easy examples. Lovasz Softmax Loss: This loss function is based on the Lovasz extension of submodular functions and is designed to optimize the intersection-over-union (IoU) metric directly. The choice of loss function depends on the specific task and the performance metric of interest. In practice, it is often useful to experiment with different loss functions and compare their performance on a validation set. Therefore, Dice loss was used in the dataset to compare very similar stroke types.

Multi-scale feature fusion is a technique used to combine features from different scales to improve the accuracy of the model. In the UNET model, multi-scale feature fusion is typically achieved by concatenating feature maps from different layers of the encoder and decoder networks. This allows the model to capture both high-level and low-level features, which can be particularly useful in tasks where objects of different sizes need to be segmented. Mathematically, the multi-scale feature fusion operation can be represented as follows:

$$F_i = \text{Concatenate}\ (F_i\text{ -1}, G_i) \qquad (1)$$

Where F_i is the i-th feature map in the decoder network, F_i -1 is the corresponding feature map in the encoder network, and G_i is the i-th feature map in the corresponding decoder layer. The resulting feature map is then processed further in the decoder network to refine the segmentation output.

A residual connection is a technique used to improve the training of deep neural networks by allowing gradients to flow more

easily through the network. In the UNET model, residual connections are typically used to connect layers in the encoder and decoder networks. Mathematically, a residual connection can be represented as follows:

$$F_i = G_i + \text{Conv}(F_i - 1) \quad (2)$$

Where F_i is the i-th feature map, G_i is the corresponding feature map in the same layer, Conv is a convolutional layer, and the '+' operator denotes element-wise addition. The resulting feature map is then processed further in the decoder network to refine the segmentation output.

Spatial dropout is a variant of the standard dropout technique used in deep learning to prevent overfitting. In the UNET model, spatial dropout is typically applied to the input feature maps to the decoder network. Mathematically, spatial dropout can be represented as follows:

$$F_i = \text{Dropout}(F_i - 1) \quad (3)$$

Where F_i-1 is the input feature map, Dropout is the spatial dropout layer, and F_i is the resulting feature map with some of its elements set to zero. The resulting feature map is then processed further in the decoder network to refine the segmentation output.

Batch normalization is a technique used in deep learning to normalize the inputs to each network layer. In the UNET model, batch normalization is typically applied to the convolutional layers in both the encoder and decoder networks. Mathematically, batch normalization can be represented as follows:

$$F_i = \text{BatchNorm}(\text{Conv}(F_i - 1)) \quad (4)$$

Where F_i-1 is the input feature map, Conv is the convolutional layer, BatchNorm is the batch normalization layer, and F_i is the resulting feature map with normalized values. The resulting feature map is then processed further in the decoder network to refine the segmentation output.

In the UNET model, the weighted cross-entropy loss function can be used to improve the accuracy of segmentation tasks by giving more weight to certain classes or regions of interest in the image.

The Dice loss is a commonly used loss function in deep learning models for image segmentation tasks. It is named after the Dice coefficient, which is a statistical metric used to measure the similarity between two sets.

The Dice coefficient is defined as follows:

$$\text{Dice coefficient} = (2 * |A \cap B|) / (|A| + |B|) \quad (5)$$

where A and B are two sets, and |A| and |B| represents the number of elements in each set. $|A \cap B|$ represents the number of common elements between A and B.

The Dice loss is derived from the Dice coefficient and is used to measure the dissimilarity between the predicted segmentation mask and the ground truth segmentation mask. The Dice loss is defined as follows:

$$\text{Dice loss} = 1 - (2 * |P \cap G|) / (|P| + |G|) \quad (6)$$

where P and G represent the predicted and ground truth segmentation masks, respectively, and |P| and |G| represent the number of pixels in each mask. $|P \cap G|$ represents the number of common pixels between the predicted and ground truth masks.

The Dice loss ranges from 0 to 1, with a value of 0 indicating complete dissimilarity between the predicted and ground truth masks, and a value of 1 indicating perfect similarity.

In the UNET model, the Dice loss is commonly used as the loss function to optimize the model during training. The UNET architecture is designed for image segmentation tasks. The Dice loss is well-suited for this type of problem because it penalizes false positives and false negatives equally, making it a more balanced loss function than other options like binary cross-entropy.

During training, the UNET model updates its weights to minimize the Dice loss between the predicted and ground truth segmentation masks. By minimizing this loss, the model learns to produce more accurate segmentation masks, which is more

effective for medical imaging.

Intersection over Union (IOU) is a common evaluation metric used in image segmentation tasks to measure the accuracy of the segmentation output. It measures the overlap between the predicted segmentation mask and the ground truth mask.

In conclusion, hyperparameter optimization techniques such as grid search, random search, and Bayesian optimization is used to optimize hyperparameters in a UNET model for image segmentation tasks to maximize IOU metrics. These techniques involve exhaustively searching through all possible combinations, randomly sampling from a predefined distribution, or using a probabilistic model to predict performance, respectively.

3. Results and Discussion

In this study, VGG16, InceptionV3, DenseNet, Xception and InceptionResnetV2 were compared with data augmentation, fine-tuning and transfer learning methods. Also, the hyperparameter-optimized UNet model was trained to segment the stroke type and region from the CT scan. The Dice loss optimization, multiscale feature optimization, batch normalization, learning rate optimization, Residual Connection, and Spatial Dropout Loss were used in our study.

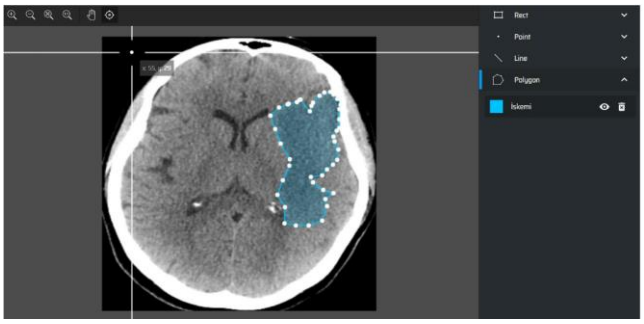


Figure 1 Labeled data example of CT image.

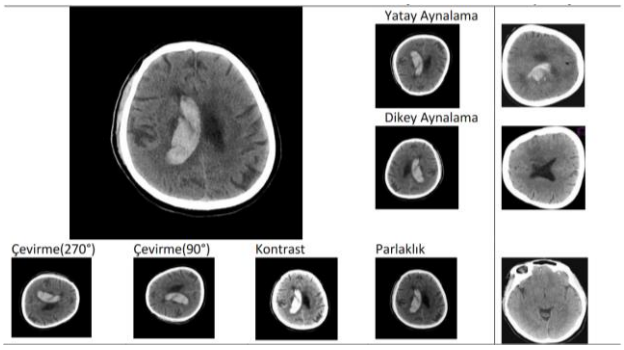


Figure 2 Rotate, contrast, brightness, mirror, and ROI example of data augmentation.

Data augmentation was performed to evaluate the performance of the CT scan. The critical point of the augmentation is that every augmentation type needs to have a reason to add to the dataset. For example, in CT the images can rotate, contrast can change, brightness can change, and the images can be mirrored depending on the application. Therefore, in the study, these parameters were made accordingly. Table 1 shows the augmentation of the data and Table 2 shows split size of the model data for training and test purposes.

Table 1 Dataset size after augmentation, 5-fold of original dataset

Class	**Dataset size**
Positive	8892
Negative	8854

Table 2 Training, validation, and test data size for comparison

Model Data	**Data Size**
Training	12422
Test	3550
Validation	1774

As shown in Table 3, the preprocessed data were prepared for training as ischemia and hemorrhage for the segmentation model. Additionally, data augmentation procedures are shown in Figure 11.

Table 3 The segmentation model data size for ischemia and hemorrhage

Class	**Dataset size**
Ischemia	6780
Hemorrhage	6558

Table 4 Classification model training results for stroke detection

Model	**Accuracy**	**Val_acc**	**Loss**	**Val_Loss**
VGG16	0.9189	0.9336	0.193	0.227
Inceptionv3	0.8793	0.9152	0.277	0.266
DenseNet	0.8025	0.8863	0.454	0.399
Xception	0.9235	0.9256	0.190	0.198
InceptionResnetV2	0.9052	0.9262	0.211	0.234

Table 5 Classification result parameters for detecting the stroke.

Model	**Class**	**Accuracy**	**Precision-Recall**	**F1 score**
VGG16	Negative	0.92	0.96	0.94
	Positive	0.95	0.94	0.92
Inceptionv3	Negative	0.81	0.92	0.87
	Positive	0.97	0.83	0.85
DenseNet	Negative	0.72	0.93	0.86
	Positive	0.98	0.72	0.80
Xception	Negative	0.91	0.91	0.90
	Positive	0.96	0.90	0.92
Inception ResnetV2	Negative	0.92	0.92	0.95
	Positive	0.94	0.88	0.94

Table 6 Improved UNet segmentation results for IOU metric to detect ischemia and hemorrhage.

Model	**Accuracy**	**Val_acc**	**IOU**	**Loss**	**Val_Loss**
U-Net	0.8825	0.8125	0.6505	0.2125	0.2397

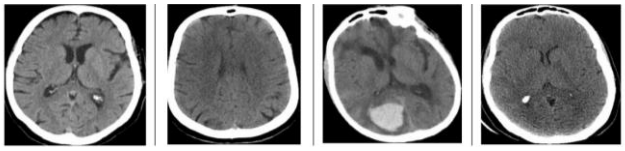


Figure 3. Ischemia and hemorrhage data examples

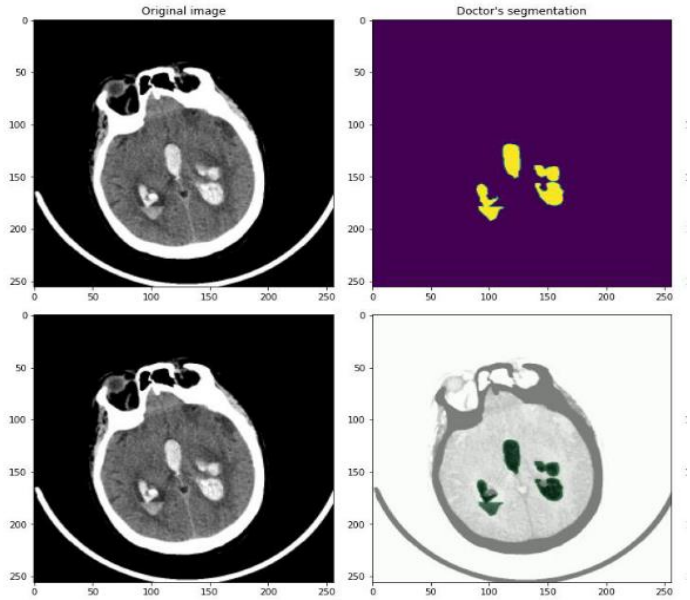


Figure 4 Improved Unet Segmentation results compared to the ground truth.

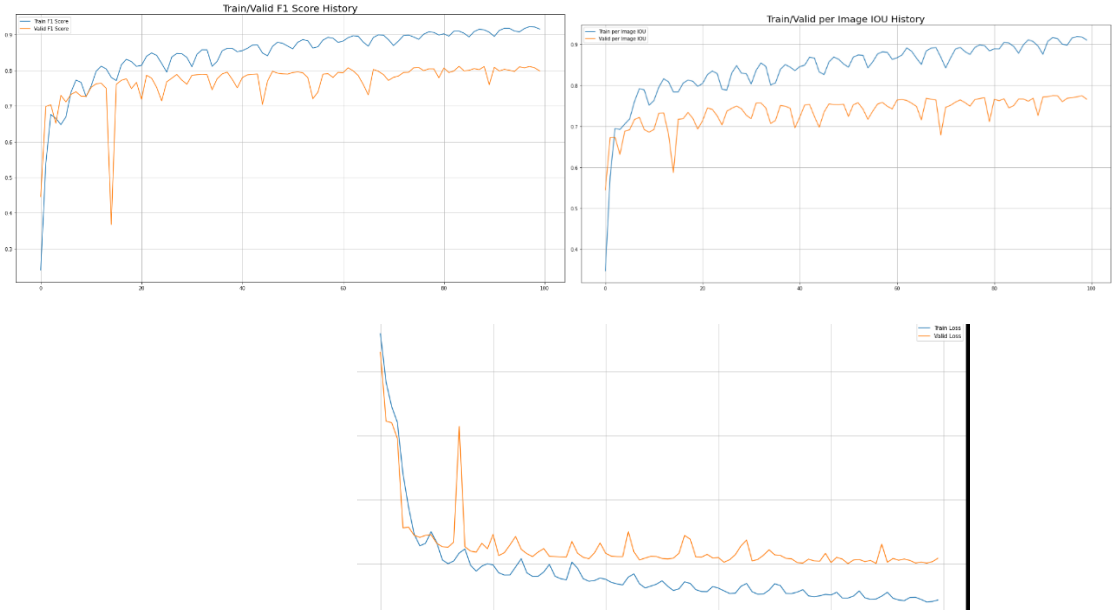


Figure 5 F1, IOU and Loss graphics.

We have demonstrated how classification models can be used to detect strokes in CT images. These models can be trained to recognize patterns in the images that are indicative of a stroke and can provide a more accurate and efficient way to detect stroke compared to traditional image analysis methods.

One of the benefits of using classification models for stroke detection in CT images is that they can provide a binary output, indicating whether a stroke is present in the image. This can help physicians quickly and easily identify patients who require further examination or treatment.

Another benefit is that these models can be trained with a large dataset of CT images, which can improve their accuracy and generalizability. Additionally, these models can be implemented with different architectures such as CNNs, RNNs and others, which can further improve their performance.

Also, in our study segmentation model was used. Segmentation models can be used to detect ischemia and hemorrhage in medical images. These models can segment or label specific regions of an image, such as the brain or blood vessels. They can be trained to identify patterns in the images that are indicative of ischemia or hemorrhage.

One of the benefits of using segmentation models for ischemia and hemorrhage detection is that they can provide more specific information about the location and extent of the condition within the image. This can help physicians to make more accurate diagnoses and treatment decisions. Additionally, these models can be used to automatically segment the regions of interest in an image, which can save time and reduce the need for manual annotation.

4. Conclusion

Ischemia and hemorrhage detection in CT images with deep learning can be challenging for a few reasons:

- Variability in imaging protocols: Different imaging protocols can result in variations in the appearance of ischemia and hemorrhage in CT images. This can make it difficult for deep learning models to learn to recognize patterns that are indicative of these conditions.
- Limited annotated data: Obtaining a large dataset of annotated CT images that contain ischemia and hemorrhage can be difficult. This can make it challenging to train deep learning models that are able to accurately detect these conditions.
- High dimensionality: CT images are high-dimensional, making it difficult for deep-learning models to learn to recognize patterns in the images.
- Overlapping features: Ischemia and hemorrhage can have similar features, making it difficult for deep learning models to differentiate between them.
- Class imbalance: Ischemia and hemorrhage may be rare in some datasets, making it difficult for deep learning models to learn to detect these conditions.

All these challenges could be addressed by using more sophisticated models, more data, and more advanced pre-processing techniques. In our work classification and segmentation models were used to challenge the task of detecting the stroke type automatically. The IOU metric is a very difficult metric to improve given the ischemia and hemorrhage similarities on CT images. Therefore pixel-wise accurate models must be evaluated and given to the medical professional for usage.

References

[1] Dong Chuang Guo, Jun Gu, Jian He, Hai Rui Chu, Na Dong, Yi Feng Zheng, "External Validation Study on The Value of Deep Learning Algorithm for The Prediction of Hematoma Expansion from Noncontrast CT Scans", *Bmc Medical Imaging*, 2022.

[2] Md Moniruzzaman Emon, Tareque Rahman Ornob, Moqsadur Rahman, "Classifications of Skull Fractures Using CT Scan Images Via CNN with Lazy Learning Approach", *Arxiv-Eess.Iv*, 2022.

[3] Miguel López-Pérez, Arne Schmidt, Yunan Wu, Rafael Molina, Aggelos K Katsaggelos, "Deep Gaussian Processes for Multiple Instance Learning: Application to CT Intracranial Hemorrhage Detection", *Computer Methods And Programs In Biomedicine*, 2022.

[4] V Pandimurugan, S Rajasoundaran, Sidheswar Routray, A V Prabu, Hashem Alyami, Abdullah Alharbi, Sultan Ahmad, "Detecting and Extracting Brain Hemorrhages from CT Images Using Generative Convolutional Imaging Scheme", *Computational Intelligence And Neuroscience*, 2022.

[5] Farhan Ullah, Jihoon Moon, Hamad Naeem, Sohail Jabbar, "Explainable Artificial Intelligence Approach in Combating Real-time Surveillance of COVID19 Pandemic from CT Scan and X-ray Images Using Ensemble Model", *The Journal Of Supercomputing*, 2022.

[6] Murugan Hemalatha, "A Hybrid Random Forest Deep Learning Classifier Empowered Edge Cloud Architecture for COVID-19 and Pneumonia Detection", *Expert Systems With Applications*, 2022.

[7] Nirmala Devi Kathamuthu Shanthi Subramaniam, Quynh Hoang Le, Suresh Muthusamy, Hitesh Panchal, Suma Christal Mary Sundararajan, Ali Jawad Alrubaie, Musaddak Maher Abdul Zahra, "A Deep Transfer Learning-based Convolution Neural Network Model for COVID-19 Detection Using Computed Tomography Scan Images for Medical Applications", *Advances In Engineering Software (Barking, London, England ...*, *2022.*

[8] Yue Zhao, Zhongyang Wang, Xinyao Liu, Qi Chen, Chuangang Li, Hongshuo Zhao, Zhiqiong Wang, "Pulmonary Nodule Detection Based on Multiscale Feature Fusion", *Computational And Mathematical Methods In Medicine*, 2022.

[9] Jing Xu, Haojie Ren, Shenzhou Cai, Xiaoping Zhang, "An Improved Faster R-CNN Algorithm for Assisted Detection of Lung Nodules", *Computers In Biology And Medicine*, 2022.

[10] Yashwanth Manjunatha, Vanshali Sharma, Yuji Iwahori, M K Bhuyan, Aili Wang, Akira Ouchi, Yasuhiro Shimizu, "Lymph Node Detection in CT Scans Using Modified U-Net with Residual Learning and 3D Deep Network", *International Journal Of Computer Assisted Radiology And ...*, 2023.

[11] Ujjwal Upadhyay, Mukul Ranjan, Satish Golla, Swetha Tanamala, Preetham Sreenivas, Sasank Chilamkurthy, Jeyaraj Pandian, Jason Tarpley, "Deep-ASPECTS: A Segmentation-Assisted Model for Stroke Severity Measurement", *Arxiv-Eess.Iv*, 2022.

[12] Yang Wang, Junkai Zhu, Jinli Zhao, Wenyi Li, Xin Zhang, Xiaolin Meng, Taige Chen, Ming Li, Meiping Ye, Renfang Hu, Shidan Dou, Huayin Hao, Xiaofen Zhao, Xiaoming Wu, Wei Hu, Cheng Li, Xiaole Fan, Liyun Jiang, Xiaofan Lu, Fangrong Yan, "Deep Learning-Enabled Clinically Applicable CT Planbox for Stroke With High Accuracy and Repeatability", F*rontiers In Neurology*, 2022.

[13] John T Murchison, Gillian Ritchie, David Senyszak, Jeroen H Nijwening, Gerben van Veenendaal, Joris Wakkie, Edwin J R van Beek, "Validation of A Deep Learning Computer Aided System for CT Based Lung Nodule Detection, Classification, and Growth Rate Estimation in A Routine Clinical Population", *Plos One*, 2022.

[14] Ine Dirks, Marleen Keyaerts, Bart Neyns, Jef Vandemeulebroucke, "Computer-aided Detection and Segmentation of Malignant Melanoma Lesions on Whole-body 18 F-FDG PET/CT Using An Interpretable Deep Learning Approach", *Computer Methods And Programs In Biomedicine*, 2022.

[15] Jake Kendrick, Roslyn J Francis, Ghulam Mubashar Hassan, Pejman Rowshanfarzad, Jeremy S L Ong, Martin A Ebert, "Fully Automatic Prognostic Biomarker Extraction from Metastatic Prostate Lesion Segmentations in Whole-body [68 Ga]Ga-PSMA-11 PET/CT Images", *European Journal Of Nuclear Medicine And Molecular Imaging*, 2022.

[16] Chetna Kaushal, Md Khairul Islam, Sara A Althubiti, Fayadh Alenezi, Romany F Mansour, "A Framework for Interactive Medical Image Segmentation Using Optimized Swarm Intelligence with Convolutional Neural Networks", *Computational Intelligence And Neuroscience*, 2022.

[17] T Ahila, A C Subhajini, "E-GCS: Detection of COVID-19 Through Classification By Attention Bottleneck Residual Network", *Engineering Applications Of Artificial Intelligence*, 2022.

[18] Xun Wang, Hanlin Li, Pan Zheng, "Automatic Detection and Segmentation of Ovarian Cancer Using A Multitask Model in Pelvic CT Images", *Oxidative Medicine And Cellular Longevity*, 2022.

[19] Rahul Gomes, Connor Kamrowski, Pavithra Devy Mohan, Cameron Senor, Jordan Langlois, Joseph Wildenberg, "Application of Deep Learning to IVC Filter Detection from CT Scans", *Diagnostics (Basel, Switzerland)*, 2022.

[20] Sophie Ostmeier, Brian Axelrod, Benjamin F. J. Verhaaren, Abdelkader Mahammedi, Li-Jia Li, Greg Zaharchuk, Soren Christensen, Jeremy J. Heit, "Non-inferiority of Deep Learning Acute Ischemic Stroke Segmentation on Non-Contrast CT Compared to Expert Neuroradiologists", *Arxiv-Eess.Iv*, 2022.

[21] Zihui Ouyang, Peng Zhang, Weifan Pan, Qiang Li, "Deep Learning-based Body Part Recognition Algorithm for Three-dimensional Medical Images", *Medical Physics*, 2022.

[22] Chun-Chieh Wang, Pei-Huan Wu, Gigin Lin, Yen-Ling Huang, Yu-Chun Lin, Yi-Peng Eve Chang, Jun-Cheng Weng, "Magnetic Resonance-Based Synthetic Computed Tomography Using Generative Adversarial Networks for Intracranial Tumor Radiotherapy Treatment Planning", *Journal Of Personalized Medicine*, 2022.

[23] Marin Benčević, Marija Habijan, Irena Galić, "Epicardial Adipose Tissue Segmentation from CT Images with A Semi-3D Neural Network", *Arxiv-Eess.Iv*, 2022.

[24] Amin Gasmi, "Deep Learning and Health Informatics for Smart Monitoring and Diagnosis", *Arxiv-Q-Bio.Qm*, 2022.

[25] Li Sun, Junxiang Chen, Yanwu Xu, Mingming Gong, Ke Yu, Kayhan Batmanghelich, "Hierarchical Amortized GAN for 3D High Resolution Medical Image Synthesis", *Ieee Journal Of Biomedical And Health Informatics*, 2022.

[26] Manika Jha, Richa Gupta, Rajiv Saxena, "A Framework for In-vivo Human Brain Tumor Detection Using Image Augmentation and Hybrid Features", *Health Information Science And Systems*, 2022.

[27] Eugene Vorontsov, Pavlo Molchanov, Matej Gazda, Christopher Beckham, Jan Kautz, Samuel Kadoury, "Towards Annotation-efficient Segmentation Via Image-to-image Translation", *Medical Image Analysis*, 2022.

[28] Lisa C. Adams, Felix Busch, Daniel Truhn, Marcus R. Makowski, Hugo JWL. Aerts, Keno K. Bressem, "What Does DALL-E 2 Know About Radiology?", *ARXIV-CS.CV*, 2022.

[29] Sarah Ettinger, Lena Sonnow, Christian Plaass, Alexandra Rahn, Christina Stukenborg-Colsman, Christian von Falck, Gesa Poehler, Christoph Becher, "Arthroscopic Defect Size Measurement in Osteochondral Lesions of The Talus Underestimates The Exact Defect Size and Size Measurement with Arthro-MRI (MR-A) and High-resolution Flat-panel CT-arthro Imaging (FPCT-A)", *Knee Surgery, Sports Traumatology, Arthroscopy : Official ...*, 2022.

[30] Connie Y Chang, Florian A Huber, Kaitlyn J Yeh, Colleen Buckless, Martin Torriani, "Original Research: Utilization of A Convolutional Neural Network for Automated Detection of Lytic Spinal Lesions on Body CTs", *Skeletal Radiology*, 2023.

Acknowledgment
Sakarya University of Applied Sciences BAPK supports this study with project number 078-2022.

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/

DOI: 10.35377/saucis...1207742 RESEARCH ARTICLE

The Effects of Preprocessing on Turkish and English News Data

Bekir Parlak[1]

[1]Amasya University, Türkiye

Corresponding author:
Bekir Parlak, Amasya University
E-mail address:
bekir.parlak@amasya.edu.tr

Submitted:21 Nov 2022
Revised: 27 March 2023
Accepted: 30 March 2023
Published Online: 30 April 2023

Citation: Parlak,B. (2023). The effects of preprocessing on Turkish and English News Data. *Sakarya University Journal of Computer and Information Sciences* 6 (1) https://doi.org/10.35377/saucis...1207742

ABSTRACT

In a standard text classification (TC) study, preprocessing is one of the key components to improve performance. This study aims to look at how preprocessing effects TC according to news text, text language, and feature selection. All potential combinations of commonly used preprocessing techniques were compared on one domain, namely news data, and two different news datasets for this aim. Preprocessing technique contributions to classification performance at multiple feature sizes, possible interconnections among these techniques, and technique dependency on corresponding languages were all evaluated in this way. The effect of two important preprocessing techniques on two different common news datasets was examined. While the highest performance for the Turkish dataset is a 0.781 F1 score, the highest performance for the English dataset is a 0.980 F1 score.

Keywords: Feature selection, news data, preprocessing, text classification

1. Introduction

TC is one of the most challenging study subjects due to the requirement to organize and classify an increasing number of digital text documents globally. TC has been efficiently used in many different fields.

Preprocessing, feature extraction, feature weighting, feature selection, and classification are all steps in a traditional TC system. Stop-word removal, stemming, tokenization, and lowercase conversion, are common tasks in the preprocessing stage. In most cases, the feature extraction step uses the vector space model[1] which employs the bag-of-words technique[2]. Filter methods such as document frequency[3], information gain[4], Gini index[5], Normalized Difference Measure(NDM)[6], Max-Min Ratio(MMR)[7], Extensive Feature Selector[8] and Class-index Corpus-index Measure(CiCi)[9] are used to the feature selection step for TC domain. As a final, the classification step employs successful and well-known classification methods, such as nave Bayesian classifiers, artificial neural networks, decision trees, support vector machines, and among others.

While it has been established that feature extraction, feature weighting, feature selection, and classification method have a significant effect on the performance of TC, the preprocessing step may also have a significant impact. Stemming, stop-word removal, alphabetic tokenization and lowercase conversion are commonly used in text categorization research without thoroughly analyzing their contributions to classification accuracy.

One investigation into the categorization of Turkish news data, Kılınç et al. [10] created a new dataset called TTC-3600 that may be extensively used in TC research of Turkish news and article content. On TTC-3600, different successful classifiers in the TC domain and successful feature selection methods are evaluated. The experimental studies show that the combination of the Random Forest (RF) classifier and filter-based feature selection method achieves the best performance in all

comparisons performed after pre-processing techniques and feature selection steps. In another study[11], a TC study was carried out on the TTC-3600 dataset using Convolutional Neural Networks (CNN) and Word2Vec method and compared with the previous study using the same dataset. In the study, two different CNNs were trained and tested on the raw and stemmed versions of the TTC-3600 with the Zemberek library. CNN and Word2Vec methods showed more performance than classical statistical and machine learning algorithms. Yıldırım et al.[12] were compared traditional bag-of-words approach and neural network based new representation approaches in terms of TC. In this study, it seems that the traditional methods of effective feature selection are still at a level to compete with the new generation word embeddings approach. Experiments are reported by diversifying in terms of these two approaches and successful TC architecture for Turkish is discussed in detail. Safali et al.[13] classified academic studies based on deep learning using the Doc2vec word embeddings method. For the training, 7 different symposiums broadcasting in Turkey were selected. During the classification process, it was ensured that the studies were classified into 9 different categories by using recurrent neural networks (RNNs) and LSTM architectures. Köksal[14] conducted experiments with the TTC-4900 dataset. This dataset is comparable to TTC-3600. The TTC-4900 dataset contains 4900 news documents and 700 examples of Turkish and English news data from seven different classes. The study made extensive use of data correction. Stop words in Turkish and English are then removed. Finally, the process of root separation (lemmatization) is used. The F1 score increased when the original data was corrected, while it decreased when lemmatized. As a result, the original dataset received % 90 F1 score, but the F1 score increased to % 91.77, after correcting the data without using lemmatizing.

One investigation into the categorization of English news data, Dadgar et al.[15] objects to categorize news. It was proposed using Term Frequency-Inverse Document Frequency (TF-IDF) and Support Vector Machine (SVM). The proposed method consists of three steps: 1) data preprocessing, 2) TF-IDF feature weighting, and 3) SVM classification. The proposed method was tested using two datasets. The classification performances for the BBC and 20Newsgroup datasets were 97.84 and 94.93 percent, respectively. When compared to other classification methods, these are very desirable results. Haryanto et al.[16] can enhance the effectiveness of TC using Chi-square feature selection and SVM on preprocessing data results, including lemmatization and stemming. In another study[17], TC is carried out using a new bi-gram approach instead of the unigram approach to construct a feature vector. The proposed method makes significant contributions to the field of TC. In a study[18], researchers focused on identifying sentence level negations in news articles. This work makes use of online news articles from BBC News. Machine Learning Algorithms such as SVM and Nave Bayes are used to analyze the results. SVM has an accuracy of 96.46 percent, while Naive Bayes has an accuracy of 94.16 percent.

The effect of two techniques, which are the most important pre-processing steps, on text classification performance is analyzed in detail in this study. These techniques are stemming and stopwords removal. These two techniques have a serious impact in terms of dimension reduction. Because after these techniques are applied, the feature size decreases as unnecessary and similar words are discarded. In addition, these techniques were applied to Turkish and English datasets to examine how they changed performance on the basis of languages.

The remainder of this research is structured as follows. In section 2, materials and methods are explained. These are datasets, preprocessing, feature selection and representation, and classification algorithms. In section 3, we described experimental settings consisting of classifiers, and accuracy analysis. Finally, a conclusion is given in section 4.

2. Materials and Methods

In this section, the datasets, preprocessing techniques, feature selection methods, feature weighting and pattern classifiers used in the experiments are explained in detail. In Section 2.1, datasets are presented in detail. Preprocessing techniques are explained in Section 2.2. The feature selection techniques used in this study are explained in detail in Section 2.3. Information about feature representation and feature weighting is given in Section 2.4. Finally, the classification algorithms used in this study are given in Section 2.5. The flowchart of the study is shown in Figure 1.

2.1 Datasets

Preprocessing techniques are assessed in two different datasets, and on just one topic, namely news. English is a non-agglutinative language, despite Turkish being one of the world's most often spoken agglutinative languages. The number of documents within the same categories is kept essentially constant to allow for an impartial review. The Turkish news dataset includes 300 training and 300 test samples for each class. The English news datasets include 500 training samples and 500 test examples for each class. The number of features reaches hundreds of thousands in studies in the field of text classification. For this reason, it is sufficient that the training-test rate is 50%-50%. In addition, very high scores are obtained when the training data is above 50%. Tables 2-3 summarize the class distributions for news datasets. Two news datasets are balanced. The preprocessing tasks for the multi-class classification problem are evaluated using the news datasets. The first dataset is TTC-3600[10]. Being user-friendly and well-documented is the most crucial aspect of this dataset, which may be extensively employed in

TC studies pertaining to Turkish news and articles. The dataset contains 3600 documents in total, 600 of which are news stories or texts from six different categories. These articles were gathered from six reputable news portals and agencies. The second dataset is 20Newsgroups. The newsgroups dataset is evenly distributed, with equal documents in all classes. The number of documents in each class of the datasets are presented in Table 1 and Table 2.

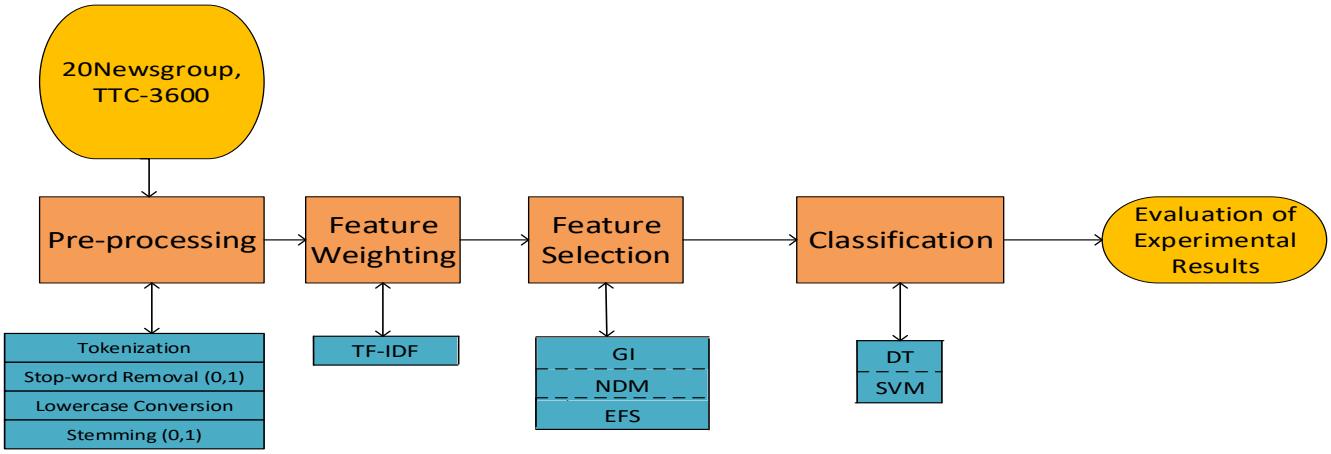


Figure 1 Flowchart of the study

Table 1 TTC-3600 Dataset

Categories	Total	Train	Test
Culture	600	300	300
Economy	600	300	300
Health	600	300	300
Policy	600	300	300
Sport	600	300	300
Technology	600	300	300
Total	3600	1800	1800

Table 2 20Newsgroup

Categories	Total	Train	Test
alt.atheism	1000	500	500
comp.graphics	1000	500	500
comp.os.ms-windows.misc	1000	500	500
comp.sys.ibm.pc.hardware	1000	500	500
comp.sys.mac.hardware	1000	500	500
comp.windows.x	1000	500	500
misc.forsale	1000	500	500
rec.autos	1000	500	500
rec.motorcycles	1000	500	500
rec.sport.baseball	1000	500	500
Total	10000	5000	5000

2.2 Pre-processing techniques

Within the scope of this study, four common preprocessing steps of TC are considered: stop-word removal, stemming, lowercase conversion, and tokenization[19]. Of these techniques, lowercase conversion and tokenization do not vary with the language of the dataset, while stop-word removal and stemming techniques vary with the language of the dataset. Because the method of finding the stem of the words in each language and the stop-word list is different. However, tokenization and lowercase conversion are the same in every language, as no structural processing is applied to the word. Stemming algorithms are customized to the language according to the study. Among the various approaches, the fixed-prefix algorithm [20] is a simple but highly effective stemming technique for Turkish. However, the stemming algorithm, namely Porter-stemmer introduced in a study[21] is widely used by English researchers. In this study, Zemberek was used for the Turkish dataset and Porter-stemmer was used for the English dataset. Zemberek is an open-source natural language processing library that you can use for Turkish languages, developed by Ahmet A. Akın using the Java programming language. By using this library, besides finding the stem words, spelling, checking whether a word is Turkish, correcting spelling mistakes etc. can be done.

There are some rules specific to the Turkish language in the Zemberek library. Today, there is no algorithm and library that works more accurately than the Zemberek library for Turkish in detecting word stem. It is aimed to find the stem of the word by removing the suffixes such as constructional affixes and inflectional affixes.

2.3 Feature selection methods

In this study, it is employed three different feature selection methods. These methods are Gini Index, Normalized Difference Measure, Extensive Feature Selector, Odds Ratio, and Chi-Square. While GI, NDM, and EFS are global methods, OR and CHI2 are local. So, we utilized global and local methods in the experiments. All notations used in these methods are shown in Table 3.

Table 3 Notations

Notation	Meaning
$P(t\|C_j)$	Probability of term t when class C_j exists
$P(\bar{t}\|C_j)$	Probability of absence of term t when class C_j exists
$P(t\|\bar{C}_j)$	Probability of term t when class C_j does not exists
$P(\bar{t}\|\bar{C}_j)$	Probability of absence of term t when class C_j does not exists
$P(C_j\|t)$	Probability of class C_j when term t exists
$P(\bar{C}_j\|t)$	Probability of absence of class C_j when term t exists
$P(C_j\|\bar{t})$	Probability of class C_j when term t does not exist
$P(\bar{C}_j\|\bar{t})$	Probability of absence of class C_j when term t does not exist

2.3.1 Gini Index (GI)

GI is a successful technique for the TC domain. A novel measure of the Gini index is created in order to fit TC. Following an examination of the advantages and disadvantages of the current text feature selection measure functions, the GI formula is as follows:

$$\boldsymbol{GI(t) = \sum_{i=1}^{M} P(t|C_j)^2 \cdot P\left(C_j|t\right)^2} \quad (1)$$

2.3.2 Normalized Difference Measure (NDM)

The NDM[6] technique, which takes into account relative document frequencies. The Balanced Accuracy Measure has been improved with the NDM method. The NDM algorithm computes as follows:

$$\boldsymbol{NDM(t) = \sum_{i=1}^{M} \frac{\left|P\left(t|\ C_j\right) - P\left(t|\bar{C}_j\right)\right|}{\min\left(P\left(t|\ C_j\right), P\left(t|\bar{C}_j\right)\right)}} \quad (2)$$

2.3.3 Extensive Feature Selector (EFS)

EFS is a successful technique for the TC domain. A novel measure of the EFS is created in order to fit TC. Following an examination of the advantages and disadvantages of the current text feature selection measure functions, the EFS formula is as follows:

$$\boldsymbol{EFS(t) = \sum_{j=1}^{M} \left(\frac{P\left(t|C_j\right)}{P\left(\bar{t}|C_j\right) + P\left(t|\bar{C}_j\right) + 1}\right) \cdot \left(\frac{P\left(C_j|t\right)}{P\left(\bar{C}_j|t\right) + P\left(C_j|\bar{t}\right) + 1}\right)} \quad (3)$$

2.4 Feature Representation and Weighting

In general, it has been evaluated text documents as a bag of words (BoW) approach in machine learning classifiers. In an enhanced form of BoW known as the Vector Space Model (VSM), each document is represented as a vector, with each dimension denoting a distinct term (word) or feature. A term's value in the vector changes from zero to non-zero if it appears

in the text. The objective, as viewed from a TC perspective, is to create vectors with features for each class using training documents. Term weighting is a crucial stage in VSM, and there are three main factors that influence how important a term is in a document. These factors are document length normalization, Term frequency factor (TF), and the inverse document frequency factor (IDF).

2.5 Classification algorithms

The goal of TC is the classification of unclassified documents into predetermined classes. Machine learning classifiers for TC are well-documented in the literature. In this study, we employed two successful pattern classifiers, namely Decision Tree (DT) and Support Vector Machine (SVM). The following section provides illustrations of the specific details about each classifier that was chosen.

2.5.1 J48 Decision Tree (DT)

DT learning is a supervised machine learning classifier that uses a DT to classify an input document and determine its category. Internal nodes in the DT represent dataset attributes, leaves represent classification labels, and branches represent attribute values, respectively. J48 grows the DT using a divide-and-conquer approach. J48 is particularly successful in the field of TC.

2.5.2 Support Vector Machine (SVM)

SVM, is a successful classifier based on statistical information theory and structural risk minimization. SVM classifier is split into linear and nonlinear SVM algorithms. An infinite number of hyper-planes are formed to divide the data, and the hyper-plane with the highest margin is chosen from all of them in the linear SVM algorithm. When classes cannot be distinguished linearly and data must be translated into a higher dimensional space, nonlinear SVM is utilized. The data can also be separated linearly as a result. High accuracy and resistance to over-fitting via structural risk minimization by utilizing a regularization parameter are the major benefits of SVM.

3. Experimental Study

In this study, three different filter feature selection methods are used to evaluate according to the performance applied on 20Newsgroups and TTC-3600 datasets. DT and SVM classifiers were fed different sizes of features chosen by each feature selection approach. Additionally, the translation of documents into a term-document matrix has been carried out with the Java programming language. Classification has been carried out using the Weka software tool. The total feature size varies according to the experimental parameters. The application of the stemming algorithm and the removal of stop-words change the feature size. If these processes are not applied for two datasets, feature size increases. However, when the stemming algorithm is not applied in the TTC-3600 dataset, the number of features has increased significantly. The feature numbers are given in Table 4-5 according to the datasets and situations. The resulting F1-Scores are shown in Tables 6-9 for TTC-3600 and 20Newsgroup datasets where the highest performance is underlined and bold.

The highest score for the TTC-3600 dataset was obtained with the EFS method, SVM classifier and 1000 feature size. Also, this score was obtained without a stemming algorithm. Similarly, the highest score for the DT classifier was obtained with the NDM method and 1000 feature size and not applying the stemming algorithm. From here, it has been seen that Zemberek[22], the Turkish stemming algorithm, does not make a serious contribution to the performance for the TTC-3600 dataset. In general, performance increases as the feature size increases for the TTC-3600 dataset.

Table 4 Total number of features for TTC-3600

Situations	Total # of features
Stemming=1, Stopwords=1	19605
Stemming=1, Stopwords=0	19672
Stemming=0, Stopwords=1	62197
Stemming=0, Stopwords=0	62402

Table 5 Total number of features for 20Newsgroup

Situations	Total # of features
Stemming=1, Stopwords=1	39912
Stemming=1, Stopwords=0	40166
Stemming=0, Stopwords=1	49938
Stemming=0, Stopwords=0	50451

Table 6 F1 scores from TTC-3600 dataset with Decision Tree

Feature Selection Methods	50	100	300	500	1000
(Stemming=1,Stopwords=1)					
GI	0.636	0.609	0.683	0.684	0.701
NDM	0.410	0.570	0.687	0.692	0.689
EFS	0.571	0.626	0.670	0.691	0.702
(Stemming=1,Stopwords=0)					
GI	0.581	0.624	0.681	0.683	0.699
NDM	0.410	0.570	0.680	0.689	0.688
EFS	0.546	0.581	0.657	0.687	0.689
(Stemming=0,Stopwords=1)					
GI	0.601	0.644	0.682	0.707	0.700
NDM	0.526	0.569	0.677	0.659	0.709
EFS	0.572	0.627	0.673	0.694	0.699
(Stemming=0,Stopwords=0)					
GI	0.519	0.630	0.655	0.687	0.703
NDM	0.526	0.569	0.677	0.678	0.699
EFS	0.506	0.589	0.639	0.687	0.698

Table 7 F1 scores from TTC-3600 dataset with Support Vector Machine

Feature Selection Methods	50	100	300	500	1000
(Stemming=1,Stopwords=1)					
GI	0.671	0.662	0.728	0.750	0.762
NDM	0.469	0.614	0.716	0.717	0.738
EFS	0.535	0.662	0.720	0.735	0.762
(Stemming=1,Stopwords=0)					
GI	0.634	0.660	0.727	0.758	0.765
NDM	0.469	0.614	0.716	0.728	0.749
EFS	0.595	0.647	0.733	0.754	0.764
(Stemming=0,Stopwords=1)					
GI	0.612	0.681	0.723	0.733	0.764
NDM	0.555	0.636	0.710	0.718	0.732
EFS	0.599	0.652	0.710	0.727	0.781
(Stemming=0,Stopwords=0)					
GI	0.560	0.665	0.729	0.743	0.775
NDM	0.555	0.636	0.710	0.721	0.744
EFS	0.563	0.627	0.717	0.727	0.768

Table 8 F1 scores from 20Newsgroup dataset with Decision Tree

Feature Selection Methods	50	100	300	500	1000
(Stemming=1,Stopwords=1)					
GI	0.977	0.975	0.947	0.947	0.972
NDM	0.689	0.906	0.973	0.972	0.972
EFS	0.977	0.948	0.946	0.945	0.973
(Stemming=1,Stopwords=0)					
GI	0.976	0.975	0.946	0.946	0.970
NDM	0.689	0.906	0.979	0.979	0.980
EFS	0.976	0.976	0.946	0.945	0.945
(Stemming=0,Stopwords=1)					
GI	0.845	0.913	0.951	0.955	0.949
NDM	0.353	0.581	0.720	0.920	0.931
EFS	0.846	0.913	0.928	0.951	0.951
(Stemming=0,Stopwords=0)					
GI	0.899	0.920	0.921	0.952	0.949
NDM	0.353	0.581	0.874	0.910	0.921
EFS	0.878	0.888	0.915	0.923	0.953

Table 9 F1 scores from 20Newsgroup dataset with Support Vector Machine

Feature Selection Methods	50	100	300	500	1000
(Stemming=1,Stopwords=1)					
GI	0.970	0.968	0.965	0.960	0.960
NDM	0.701	0.898	0.976	0.973	0.966
EFS	0.971	0.969	0.963	0.960	0.961
(Stemming=1,Stopwords=0)					
GI	0.969	0.971	0.962	0.963	0.962
NDM	0.701	0.898	0.973	0.974	0.967
EFS	0.972	0.970	0.964	0.963	0.963
(Stemming=0,Stopwords=1)					
GI	0.839	0.922	0.941	0.927	0.902
NDM	0.411	0.604	0.757	0.923	0.931
EFS	0.844	0.927	0.907	0.912	0.904
(Stemming=0,Stopwords=0)					
GI	0.901	0.925	0.910	0.913	0.912
NDM	0.411	0.604	0.886	0.923	0.930
EFS	0.902	0.902	0.892	0.884	0.912

The highest score for the 20Newsgroup dataset [23] was obtained with the NDM method, DT classifier and 1000 feature size. In addition, this score was obtained by performing a stemming algorithm. Similarly, the highest score for the DT classifier was obtained by applying the NDM method and the feature size of 300, the removal of stopwords, and applying the stemming algorithm. From here, it has been seen that Porter, the English stemming algorithm, contributes to the performance [24]. In general, performance improves as the feature size increases for the 20Newsgroup dataset. In some cases, performance decreases as the size increases.

4. Results and Discussion

Experimental results have shown which method produces the highest and lowest performance. In addition, the effect of preprocessing methods on performance has been analyzed in detail. Both preprocessing techniques used significantly change the number of features. In terms of size, it had a positive contribution to performance in both datasets in general. In addition, when the stemming algorithm is not applied for Turkish, the feature size increases more than the English dataset. It was observed that applying the preprocessing technique did not have a positive effect on the performance for the Turkish dataset. However, preprocessing techniques contributed significantly to the performance for the English dataset. For this reason, the development of more effective stemming algorithms for Turkish will be a research topic for researchers in this field.

5. Conclusions

This study looked closely at how commonly used preprocessing tasks affected TC in just one domain and two different languages. The examination was conducted utilizing every possible combination of the preprocessing tasks while considering different factors including accuracy, language, and dimension reduction. Extensive experimental investigation showed that the right preprocessing task combinations, depending on the language, may significantly improve classification accuracy, whereas the wrong preprocessing task combinations may reduce classification accuracy. As a result, the preprocessing stage in the TC process is just as crucial as the other TC steps.

The two datasets examined in this work each have unique characteristics in terms of language, class distribution, and a number of classes. Hence the conclusions drawn from this study may also apply to other text collections.

References

[1] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing". *Communications of the ACM*, 1975. 18(11): p. 613-620.

[2] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features". in *European conference on machine learning*. 1998. Springer.

[3] Y. Yang, and J.O. Pedersen. "A comparative study on feature selection in text categorization." in *ICML*. 1997.

[4] C. Lee, and G.G. Lee," Information gain and divergence-based feature selection for machine learning-based text categorization." *Information processing & management*, 2006. 42(1): p. 155-165.

[5] S.R. Singh, H.A. Murthy, and T.A. Gonsalves, "Feature Selection for Text Classification Based on Gini Coefficient of

Inequality. "*Fsdm*, 2010. 10: p. 76-85.
[6] A. Rehman, K. Javed, and H.A. Babri, "Feature selection based on a normalized difference measure for text classification." *Information Processing & Management*, 2017. 53(2): p. 473-489.
[7] A. Rehman, et al., "Selection of the most relevant terms based on a max-min ratio metric for text classification." *Expert Systems with Applications*, 2018. 114: p. 78-96.
[8] Parlak, B. and A.K. Uysal, A novel filter feature selection method for text classification: Extensive Feature Selector. Journal of Information Science, 2021: p. 0165551521991037.
[9] B. Parlak, "Class-index corpus-index measure: A novel feature selection method for imbalanced text data." C*oncurrency and Computation: Practice and Experience*, 2022: p. e7140.
[10] D. Kilinc, et al., "TTC-3600: A new benchmark dataset for Turkish text categorization." *Journal of InformationScience*, 2017. 43(2): p. 174-185.
[11] A. Çiğdem. and A. Çırak, "Türkçe haber metinlerinin konvolüsyonel sinir ağları ve Word2Vec kullanılarak sınıflandırılması." *Bilişim Teknolojileri Dergisi*, 2019. 12(3): p. 219-228.
[12] S. Yıldırım, and T. Yıldız, "Türkçe için karşılaştırmalı metin sınıflandırma analizi. "*Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 2018. 24(5): p. 879-886.
[13] Y. Safali, et al. "Deep learning based classification using academic studies in doc2vec model". in 2*019 International Artificial Intelligence and Data Processing Symposium (IDAP)*. 2019. IEEE.
[14] Ö. Köksal, "Tuning the Turkish Text Classification Process Using Supervised Machine Learning-based Algorithms". in *2020 International Conference on INnovations in Intelligent SysTems and Applications* (INISTA). 2020. IEEE.
[15] S.M.H. Dadgar, M.S. Araghi, and M.M. Farahani. "A novel text mining approach based on TF-IDF and Support Vector Machine for news classification." in *2016 IEEE International Conference on Engineering and Technology (ICETECH)*. 2016. IEEE.
[16] A.W. Haryanto, and E.K. Mawardi. "Influence of word normalization and chi-squared feature selection on support vector machine (svm) text classification." in *2018 International Seminar on Application for Technology of Information and Communication*. 2018. IEEE.
[17] F. Elghannam, "Text representation and classification based on bi-gram alphabet." J*ournal of King Saud University-Computer and Information Sciences*, 2021. 33(2): p. 235-242.
[18] V.S. Shirsat, R.S. Jagdale, and S.N. Deshmukh, "Sentence level sentiment identification and calculation from news articles using machine learning techniques," in *Computing, Communication and Signal Processing*. 2019, Springer. p. 371-376.
[19] A.K. Uysal, and S. Gunal, "The impact of preprocessing on text classification." *Information Processing & Management*, 2014. 50(1): p. 104-112.
[20] D. Torunoğlu, et al. "Analysis of preprocessing methods on classification of Turkish texts." In: *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, 2011. p. 112-117.
[21] M.F. Porter, "An algorithm for suffix stripping." *Program*, 1980. 14(3): p. 130-137.
[22] A. Akın, M. D. Zemberek, "an open source NLP framework for Turkic languages". *Structure*, 2007, 10.2007: 1-5.
[23] B. Parlak, and A.K. Uysal, "The effects of globalization techniques on feature selection for text classification." *Journal of Information Science*, 2021, 47(6), 727-739.
[24] B. Parlak and A.K. Uysal, "On classification of abstracts obtained from medical journals." *Journal of Information Science*, 2020, 46(5), 648-663.

Acknowledgments
This work was not supported by any fund or project.

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/

DOI: 10.35377/saucis...1223054 RESEARCH ARTICLE

Preprocessing Impact Analysis for Machine Learning-Based Network Intrusion Detection

Hüseyin Güney

Bahçeşehir Cyprus University, Department of Computer Engineering, Nicosia, Northern Cyprus, Türkiye

Corresponding author:
Hüseyin Güney,
Bahçeşehir Cyprus University,
Department of Computer Engineering,
Nicosia, Northern Cyprus
E-mail address:
huseyin.guney@baucyprus.edu.tr

Received: 22 December 2022
Revised: 14 March 2023
Accepted: 03 April 2023
Published Online: 30 April 2023

Citation: Hüseyin Güney (2023). Preprocessing Impact Analysis for Machine Learning-Based Network Intrusion Detection. *Sakarya University Journal of Computer and Information Sciences*. 6 (1) https://doi.org/10.35377/saucis...1223054

ABSTRACT

Machine learning (ML) has been frequently studied to build intelligent systems in many problem domains. For example, one of the application areas of ML in cybersecurity is to develop intelligent intrusion detection systems (IDSs) for malicious network activity detection. However, intelligent IDS development is challenging due to many available methods in the current literature, including different types of classification algorithms and preprocessing techniques. Therefore, revealing the best-fitting methods for intrusion detection would help practitioners develop efficient detection systems. For this purpose, this study has conducted extensive experiments using the support vector machines (SVM) classifier and feature selection (FS) technique, several data normalisation techniques, and a classifier optimisation algorithm to analyse the impact of preprocessing techniques on classification. These methods were tested on three open network intrusion datasets, NSL-KDD, UNSW-NB15, and CICIDS2017. Finally, the results were analysed to investigate each method's impact on model performance and extract insights for building intelligent IDS. The optimised model achieved an accuracy of 81.51% with two features, 85.27% with 32 features, and 99.43% with 16 features for the NSLKDD, UNSW-NB15, and CICIDS2107 testing datasets, respectively. Furthermore, the results exhibited that data preprocessing has improved classification performance, and the log-scaling normalisation technique outperformed the z-score and min-max. Additionally, the results suggested that SVM-based FS improved classification performance and significantly reduced model complexity. In addition, the conclusion was drawn that classifier optimisation could enhance the performance of the classifier-dependent FS technique, such as SVM FS. However, it was observed that an inadequate feature set in the classifier optimisation process could result in worse performance; therefore, this problem must be addressed during the optimisation process for accurate optimisation. In conclusion, this study provided insights into data preprocessing in ML applications and showed the significance of data preprocessing for building accurate and efficient IDSs.

Keywords: Data Preprocessing, Classifier Optimisation, Feature Selection, Network Intrusion Detection System, Support Vector Machines.

1. Introduction

Modern computer applications are essential to daily life, providing a wide range of services. Developments of modern computer applications have led to the exchange of high-volume data over the Internet, including users' sensitive data [1]. Intrusion detection systems (IDSs) are promising for protecting user data due to their ability to monitor computer systems to determine malicious activities [2]. However, an intelligent system that can learn and recognise attacks autonomously is needed because of high-volume network traffic and various attack types. From the point of view of machine learning (ML), this is a classification and dimensionality reduction problem that can be developed to classify malicious activities.

The basic steps for the development of classification models include data preprocessing (encoding, data normalisation, and dimensionality reduction), classifier optimisation, model training, and model evaluation [3]. Encoding is applied if a feature needs to be converted to another type [3]. For example, if a classifier (e.g., support vector machine (SVM)) cannot process a categorical feature, the features must be converted to a nominal feature before model development. Data normalisation aims to transform feature values into a new range to create a better distribution of features and smaller feature space with the aim of improving classification performance and reducing model complexity [4]. Feature selection (FS) also aims at the same

goal as data normalisation by selecting the most relevant features and removing noisy and irrelevant features [5]. Finally, to develop an accurate model, it is necessary to apply preprocessing techniques prior to model development [4,5].

The network intrusion detection benchmark datasets were created in the articles [6], [7], and [8], NSL-KDD, UNSW-NB15, and CICIDS2017, respectively. First, the authors created NSLKDD to overcome the shortcomings of the KDDCUP99 dataset. Next, to tackle the shortcomings of NSLKDD, UNSW-NB15 was created. Finally, CICIDS2017 was created as an up-to-date and modern dataset.

In the study [9], the authors conducted several experiments to evaluate the performance of normalisation techniques, including decimal scaling, min-max normalisation, and z-score normalisation. In addition, Yin et al. proposed a deep learning (DL)-based IDS model using recurrent neural networks (RNN-IDS), which were normalised using the min-max method [10]. In [11], an ensemble model was developed for intrusion detection using min-max normalisation and feature selection techniques. In work [12], the authors designed and implemented several ML settings with different FS techniques using the C4.5 classifier to build intrusion detection methods. Among four FS techniques, Information Gain, Correlation-based FS, ReliefF, and Symmetrical Uncertainty, the InfoGain FS technique achieved the best performance when applied before the C4.5 classifier as a preprocessing technique for dimensionality reduction. The results revealed that the InfoGain-C4.5 pair achieved the best accuracy with 17 features. This study emphasised the importance of feature selection as a preprocessing step. In Article [13], several experiments were conducted that combined three FS techniques with various classifiers to analyse the impact of the filter and embedded FS on intrusion detection. Chi-square, information gain, and SVM recursive feature elimination (SVM-RFE) were used as the two filter and one embedded FS technique, respectively. When the filter methods were compared to SVM-RFE, SVM-RFE achieved the best performance due to the ability of SVM-RFE to select features with respect to their usefulness rather than their relevancy. As a result, this study showed that embedded FS techniques might be more promising for intrusion detection than filter methods. In addition, the authors stated that the best-performing classifier was SVM with all FS techniques. In conclusion, FS helps improve the accuracy of attack detection. However, it should be noted that embedded FS techniques are computationally more complex than filter FS techniques.

Malik et al. [14] proposed a NIDS based on the particle swarm optimisation (PSO) algorithm. In the study, extensive experiments were conducted to show that selecting the relevant features helped improve the classification performance of the proposed method. In [15], another PSO-based study was conducted, showing that the PSO-enabled SVM outperformed the default SVM configuration. Finally, Khammassi and Krichen [16] combined a wrapper FS technique using genetic algorithms with a logistic regression classifier for the detection of network intrusions. They achieved the maximum performance with 18 features for the KDD dataset and 20 for the UNSW-NB15 dataset. In the study, the effectiveness of FS was discussed based on the obtained experimental results. However, wrapper FS techniques have a relatively high computational complexity compared to embedded and filter methods.

In [17], packet preprocessing techniques were used to improve convolutional neural network (CNN) performance for intrusion detection. For this purpose, three preprocessing techniques, direct, weighted, and compressed, were developed and applied to CNN. In addition, CNN with the direct preprocessing technique was evaluated on the NSLKDD dataset. As a result, the authors stated that packet preprocessing had improved the model's performance.

The authors used deep neural networks to compare data preprocessing techniques [18]. In the study, a trial and error approach was conducted to find classifier settings, such as the number of layers, the number of neurons in each layer, and the optimiser algorithm, to obtain the optimised model. UNSW-NB15 was used in the study, whereas two different techniques were employed in this dataset. The first technique was Log transformation and MinMaxScaling, and the second was Z-score encoding and dummy encoding. Finally, it was mentioned that this particular study aimed to prove the concept that data preprocessing improves the performance of ML algorithms.

T. Ahmet and M. N. Aziz conducted an experimental study to classify intrusions in computer networks [19]. First, preprocessing and feature selection were applied, and the obtained data were classified using k-NN, SVM and Naïve Bayes classifiers. The experimental results obtained using the KDDCup99 benchmarking dataset showed that SVM with preprocessing and feature selection achieved the best performance, where Min-max normalisation and Correlation-based FS and Particle Swarm Optimisation were applied as the preprocessing and feature selection techniques, respectively.

P. Nimbalkar and D. Kshirsagar proposed a feature selection method for intrusion detection [20]. The study focused on DoS and DDoS attacks and proposed the FS method based on Information Gain and Gain Ratio FS techniques. The proposed method was evaluated on IoT-Bot and KDDCup99 datasets using the JRip classifier. The proposed system selected 16 and 19 features for the IoT-Bot and KDDCup99 datasets, respectively, and it was concluded that it performed better than the method with the complete feature set.

In [21], Naïve Bayes and KNN classifiers were used to develop a two-tier classifier. In addition, for feature selection, the Discriminant Analysis method was used. The proposed method was evaluated on the NSLKDD dataset. In [22], the authors proposed an intrusion detection framework for botnet using feature selection. According to the obtained results on the CICIDS2017 dataset, the Correlation Attribute Eval FS technique with JRip classifier achieved the best performance for botnet detection.

In order to reduce the time complexity of ML algorithms, fast kNN was proposed [23], and it was shown that fast kNN maintains the model's accuracy and reduces time complexity. Several filter FS techniques were combined for DoS attack detection, including Information Gain Ratio (IGR), Correlation (CR), and ReliefF (ReF). The results showed the significance of feature selection. A deep learning approach was used in the study [24] for intrusion detection to improve detection performance, and the proposed method was evaluated on the NSLKDD dataset [25].

In the present study, several experiments were conducted to measure the impact of preprocessing on ML applications, such as developing intelligent IDSs, using data normalisation, feature selection, and classifier optimisation algorithms. For this purpose, several network intrusion detection benchmark datasets were used, namely NSLKDD, UNSW-NB15, and CICIDS2017. To evaluate data normalisation techniques, three different and frequently used techniques (min-max, z-score, and logarithmic scaling) were applied to the original dataset before model development. Additionally, the SVM classifier was optimised using an exhaustive search algorithm, grid-search, where the optimal kernel function, cost parameter and gamma parameter were selected for all benchmarking datasets.

As a result, this study aimed to measure the impact of several data normalisation techniques to find the most effective technique for ML classification applications. Additionally, a classifier-dependent feature selection method was used to investigate feature selection performance on intrusion classification. Moreover, classifier optimisation was applied to classification and FS methods for analysing the impact of its classification performance with a reduced and complete feature set. Furthermore, the classifiers developed with default settings and optimised with different settings were compared to reveal the impact of feature selection in classifier optimisation. Finally, the ML model settings and chosen preprocessing techniques were presented to provide important insights for practitioners in the field. In summary, the contributions of this study are as follows.

(1) Detailed analysis of normalisation techniques and their impact on classifier performance
(2) Impact of classifier parameter optimisation on feature selection and classification
(3) Important insights into the development of intelligent intrusion detection systems.

The remainder of this paper is organised as follows. Section 2 explains the materials and methods used in this study. Section 3 provides details about the experimental setup. Section 4 presents and discusses the experimental results. Finally, the conclusion of this study is stated in Section 5.

2. Materials and Methods

In this section, the materials and methods used for this study are explained, including data preprocessing techniques and SVM. Furthermore, the used network intrusion benchmark datasets were mentioned in detail.

2.1 Benchmark datasets

NSLKDD dataset [6]: The NSLKDD dataset was extracted from the KDDCup'99 dataset as its enhanced version to overcome the shortcomings of the KDDcup'99 dataset. Although NSLKDD may not be the best representation of real networks, containing some synthetic data, it is a valuable benchmark dataset for evaluating the intrusion detection model. The NSLKDD dataset comprises two partitions: $KDDTrain^{+}$ (training dataset) and $KDDTest^{+}$ (test dataset). The $KDDTest^{-21}$, as a test dataset, is a subset of $KDDTest^{+}$ where the samples correctly classified by all classifiers in [26] have been removed. Thus, $KDDTest^{-21}$ is a more challenging dataset than $KDDTest^{+}$ for classification algorithms. In addition, the binary NSLKDD dataset contains two types of network activities: normal and attack. Table 1 presents the statistics of this dataset. Furthermore, the features in this dataset belong to three categories, namely *basic features* (Feature Nos. 1–10), *content features* (Feature Nos. 11–20), and *traffic features* (Feature Nos. 23–41) [10]. Finally, there are 41 features and 1 class label in this dataset.

Table 1 Record Distribution of the NSLKDD Dataset

Dataset	Normal	Attacks	Total
$KDDTrain^{+}$	67343	58630	125973
$KDDTest^{+}$	9711	12833	22544
$KDDTest^{-21}$	2152	9698	11850

UNSW-NB15 Dataset [7]: Moustafa and Slay published the UNSW-NB15 benchmark dataset for network intrusion detection in 2015 as a comprehensive dataset. They stated that this dataset was developed to overcome the shortcomings of the NSLKDD dataset. This dataset includes 47 features, where categories of features are as follows: *flow features* (1–5), *basic features* (6–18), *content features* (19–26), *time features* (27–35), and *general-purpose features* (36–47). The UNSW-NB15

training dataset and the UNSW-NB15 test dataset are the two predefined splits of this dataset. Statistics for the dataset are listed in Table 2.

Table 2 Record Distribution of the UNSW-NB15 Dataset

Dataset	Total Sample Size	Normal	Attacks
Training Set	175,341	56,000	119,341
Testing Set	82,332	37,000	45,332

CICIDS2017 Dataset [8]: The CICIDS2017 dataset was created by the Canadian Institute of Cybersecurity in 2017 as a network evaluation dataset. The CICIDS2017 dataset contains realistic network activity records extracted from an actual network setup. In addition, it comprises the most up-to-date and widely used attack types. In the dataset, network flows are based on the time stamp, source and destination IPs, source and destination ports, and protocols. The dataset consists of 78 features and a label column where the records were collected over a week. The CICIDS2017 Wednesday dataset used in this study contains normal activity records and DoS attacks. For computational reasons, the dataset was under-sampled by randomly selecting 20% of its records. Then, it was split into two partitions as the training and testing datasets. Statistics are shown in Table 3.

Table 3 Record Distribution of the CICIDS2017 Wednesday Dataset

Dataset	Total Sample Size	Normal	Attacks
Training Set	138,480	88,191	50,289
Testing Set	137,806	87,817	49,989

2.2 Data preprocessing

Data preprocessing in ML can be described as the process of investigating and transforming the dataset in terms of sample distribution (i.e., data normalisation) and dimensionality (i.e., feature selection or extraction) to improve classification performance as the primary objective. The secondary objective of preprocessing is to reduce model complexity in terms of features (dimensionality) and feature value ranges (. In addition, encoding of feature values is only applied when the classifier needs conversion of feature type. For example, SVM does not accept categorical features; thus, a categorical feature must be converted to a nominal one.

The encoding process comprises three main steps: (1) identification of categories in a categorical feature, that is, finding distinct values, (2) creation of new features for each distinct value; encoding will generate new features as many as the number of unique values in the categorical feature, (3) and finally, for each feature, the records are set to 1, and the rest is set to 0 [3].

Data normalisation can be considered as a scaling process that creates a new scale of feature values to transform the distribution of the dataset into a more balanced form. Additionally, this process helps to create a smaller hyperspace for the classifier and can lead to higher classification performance and lower computational complexity [4]. Z-score, min-max, and logarithmic-scaling (log-scaling) are the frequently used techniques in the field of ML. Min-max and z-score normalisation are linear techniques in which these functions perform a linear mapping of the feature space [4]. However, log-scaling is a nonlinear technique that transforms values in a nonlinear space. On the other hand, Min-max normalisation maps feature values into the range of [0, 1] using the minimum and maximum values. Furthermore, z-score normalisation (standardisation) transforms a feature into a new range using mean and standard deviation values, where the standard deviation of the transformed feature is always one. Additionally, log-scaling uses a log function to scale down the feature values non-linearly. Equations are listed in the following equations (eq. 1, eq. 2, and eq. 3), respectively. Note that x_{min} and x_{max} are the minimum and maximum values of the feature x in Equation 1. In equation 2, x_μ and x_σ are the mean and standard deviation values of the feature x. In addition, x_i represents the i^{th} value in feature x for all equations.

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (1)$$

$$x'_i = \frac{x_i - x_\mu}{x_\sigma} \quad (2)$$

$$x'_i = \log(x_i + 1) \quad (3)$$

Finally, dimensionality reduction reduces feature space by removing irrelevant/noisy features (feature selection) or transforming the feature set (feature extraction) into a new space. Feature selection retains the values of the features, but feature extraction changes them while transforming the space [5]. Feature selection and feature extraction are the main categories of dimensionality reduction.

2.3 Support Vector Machine Classifier

Classification is the process of building an ML model using supervision with a dataset that contains input and output data. As a supervised ML algorithm, SVM has several significant characteristics that make it a successful classifier. These are (1) the ability to find an optimal hyperplane between two linearly separable classes by maximising the class boundaries, (2) the ability to construct a high-dimensional hyperplane, (3) embedded in FS techniques as a robust FS technique, and (4) its kernel trick that allows one to create nonlinear boundaries [26, 27]. Furthermore, the kernel functions provide access to the higher space dimensions without explicitly defining the mapping function. Therefore, the performance of the SVM classification is highly dependent on its kernel function and parameters, which implies that classifier optimisation is crucial for SVM [28]. Classifier optimisation aims to find the optimal values for the classifier kernel and parameters. However, finding the best-fitted values is a dataset-specific problem and heuristic algorithms are used to solve this problem, which is time-consuming.

2.4 Feature selection

Feature selection techniques aim to select the most relevant features by removing redundant and irrelevant features, which has been widely applied in ML applications before model construction to improve model accuracy and reduce computational complexity. Embedded/Wrapper FS techniques use a classifier to obtain feature weights for ordering features from the most relevant to the least relevant, aiming to select the optimal subset of features. On the contrary, filter FS techniques check feature relevance for the same purpose. Since embedded/wrapper techniques are classifier-dependent (e.g., SVM, Random Forest), they are more accurate, whereas filter FS techniques are classifier-independent and thus faster than embedded/wrapper FS techniques.

In this study, the SVM feature selection technique (SVM-FS) [5] uses the feature weights vector created by the SVM training process to rank features with the aim of measuring the impact of classifier-dependent FS on intrusion detection. Algorithm 1 represents the algorithm for SVM-FS. Unlike SVM-RFE [5], the used SVM-FS technique runs SVM train function ones to obtain feature weights and rank feature weights accordingly. As it is shown in line 2, SVM training runs to obtain support vectors (SV) and coefficients (coefs) from constructed SVM model, and then this information is used to calculate the weight vector, which is ordered to obtain feature ranks (shown in lines 3 and 4). Note that the generated ranked feature list is in descending order, where the most significant weight represents the best feature; therefore, the first feature in the list is the most relevant, and the last is the least relevant feature.

Algorithm 1. SVM Feature Selection Technique (SVM-FS)

Input: Training Dataset
Output: Ranked Feature List

1. Initialise the training dataset, TrainingDataset.
2. SVM_Model = SVM_Training(TrainingDataset, Kernel, CostValue, Kernel_Parameter_Values)
3. Feature_Weight_List = CreateWeightList(SVM_Model$coefs, SVM_Model$SV)
4. Ranked_Feature_List = argmax(Feature_Weight_List)

3. Experimental Setup

The experiments were carried out using the software implemented with R programming (v4.0.2) [29] and RStudio [30] on a PC, Intel® i7 Core(TM) i7-4790 CPU @ 3.60GHz, 32 GB DDR3 Ram, 256 GB SSD, and Microsoft® Windows 10 Pro x64. In addition, the R programming package, e1071 [31], was used to implement the SVM classifier. This experimental study involves (1) data normalisation, (2) feature selection, (3) classifier optimisation, and (4) model training and evaluation.

First, the model has never seen the test dataset to avoid normalisation, FS, or classification bias. As mentioned in the benchmark datasets section, each dataset was split into two sets: training and testing. The model was constructed using the training dataset, and evaluation was performed on the testing dataset; hold-out validation was used. Furthermore, during the data normalisation process, the required values were obtained from the training dataset and applied to the test dataset to avoid bias can be caused by normalisation. Additionally, FS was performed on the training dataset to avoid FS bias. The model's performance was measured with the top k features after the feature ranking where k = 1, 2, 3, 4, 8, 16, 32, 64, and feature set size. Since the NSLKDD and UNSW-NB15 datasets contain categorical features, encoding was applied. After encoding, the feature size of the datasets was 122 and 194, respectively. The SVM classifier with default settings (c=1, gamma = 1 / |

feature set |, degree = 3) was implemented. Finally, grid search was used for SVM optimisation, where a kernel function, cost parameter, and the value search of the chosen kernel parameter were performed. Thus, the kernel function was selected among linear, polynomial, sigmoid, and radial basis functions (RBF) in the classifier optimisation process. Furthermore, the cost and gamma value spaces were searched where $c > 0$ and $\gamma > 0$. Performance evaluation is the final and critical stage of ML model development since it shows the model's accuracy. For this purpose, accuracy and f-score performance evaluation metrics were used [3]. The equations of these metrics are given in (4) and (7), where (5) and (6) were used to calculate F-score.

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

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

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

$$\text{F} - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{Precision + Recall} \tag{7}$$

Here, TP, TN, FP, and FN are explained below.

- True Positive (TP): Truly classified attack class (positive) sample.
- True Negative (TN): Truly classified normal class (negative) sample.
- False Positive (FP): Falsely classified normal class sample, classified as an attack.
- False Negative (FN): Falsely classified attack class sample, classified as normal activity.

4. Experiment Results

The experiment results in this study are examined in four sections, including SVM kernel function selection, comparison of normalisation techniques, the impact of FS, and the impact of classifier optimisation.

4.1. SVM kernel function performance comparison

The kernel function comparison test showed that the RBF kernel function outperformed the others for all datasets. SVM-RBF with default parameter values fits well with the training datasets and outperforms the other kernels for the KDDTest^{+}, UNSW-NB15, and CICIDS2017 Wednesday testing datasets. The linear kernel achieved the best for the KDDTest^{-21} dataset but did not fit the training dataset as RBF did. The results are shown in Table 4.

Table 4 Performance of the Kernel Functions of the SVM Classifier in terms of Accuracy

Kernel Function	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
Linear	59.3%	64.3%	**69.1%**	78.13%	69.73%	42.44%	42.28%
Polynomial	15.0%	22.5%	38.6%	30.89%	43.38%	63.88%	63.87%
Sigmoid	40.9%	53.2%	55.4%	68.06%	55.06%	63.68%	63.72%
Radial	**99.9%**	**72.4%**	47.5%	**99.24%**	**75.69%**	**99.99%**	**72.45%**

4.2. Normalisation technique selection

This subsection discusses the impact of normalisation techniques by applying min-max, z-score, and log-scaling. However, since CICIDS2017 contains negative values, the log-scaling was not applied to this dataset. In addition, normalisation technique evaluation tests were performed with the entire feature set; that is, no FS prior to classification was applied for this experiment. Moreover, the SVM-RBF classifier with default settings was used for classification. The accuracies obtained for the original and normalised datasets are listed in Table 5.

The results showed that the normalisation of the datasets significantly improved the classification performance. The achieved accuracies were as follows: for KDDTrain^{+}, the accuracy was 99.5%, which was very close to 100%; it retained the model's performance on the original dataset. For the NSLKDD testing datasets, KDDTest^{+} and KDDTest^{-21}, the accuracies were 80.8% and 63.4%, respectively; normalisation increased the model performance by 7.6% on KDDTest^{+} and 15.9% on KDDTrain^{-21}. Furthermore, z-score and log-scaling normalisation methods have achieved almost the same performance for

the UNSW-NB15 training and testing datasets, where the model's performance was improved by approximately 6%. When the min-max and z-score normalisation methods were compared for the CICIDS2017 datasets, it was shown that the z-score method outperformed the min-max. With the z-score normalisation, the accuracy was improved by approximately 26% for the Wednesday testing dataset. In conclusion, the results showed that the normalisation of the data led to a considerable improvement in the performance of SVM-RBF. It was also observed that log-scaling and z-score achieved similar performance and outperformed the min-max technique for all datasets.

In addition, statistical tests were performed on the NSLKDD dataset to further understand the normalisation techniques' impact. First, some features with high variation were randomly selected, and the minimum, maximum, mean, and standard deviation values were calculated before and after min-max, z-score, and log-scaling. In addition, the skewness and kurtosis values were calculated.

Table 5 Impact of Normalisation Techniques on the Performance of SVM-RBF in terms of Accuracy

Normalisation Technique	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
-	**99.9%**	72.4%	47.5%	**99.24%**	75.69%	**99.99%**	72.45%
Min-max	97.4%	74.7%	52.1%	92.92%	78.76%	97.03%	96.85%
Z-score	98.5%	76.7%	55.9%	93.63%	81.47%	98.58%	**98.48%**
Log-scaling	99.5%	**80.8%**	**63.4%**	93.64%	**81.55%**	-	-

Table 6 Statistics of Normalisation Techniques Using KDDTest+ Dataset

		Original	**Min-Max**	**Z-Score**	**Log-Scaling**
Duration - Feature No **1**	Mean	287.14	0.030	-5.890e-18	0.322
	Std. Dev.	2604.51	0.136	1	1.451
	Min. Val.	0	0	-0.110	0
	Max. Val.	42908	1	16.364	10.667
	Skewness	11.88	5.34	11.88	5.34
	Kurtosis	156.07	28.95	156.07	28.95
Source Bytes - Feature No **86**	Mean	45566.74	0.153	-2.023e-20	3.230
	Std. Dev.	5870331	0.142	1	2.982
	Min. Val.	0	0	-0.008	0
	Max. Val.	1379963888	1	235.067	21.045
	Skewness	190.66	0.31	190.66	0.31
	Kurtosis	39351.93	-0.67	39351.93	-0.67
Destination Host Count - Feature No **113**	Mean	128.15	0.858	3.637e-17	4.757
	Std. Dev.	99.21	0.241	1	1.334
	Min. Val.	0	0	-1.836	0
	Max. Val.	255	1	0.734	5.545
	Skewness	-0.83	-1.73	-0.83	-1.73
	Kurtosis	-1.07	1.88	-1.07	1.88

When the min and max values were observed for all normalisation techniques, it was shown that min-max obtained the smallest range, whereas the log-scaling created slightly larger ranges, but the ranges generated by the z-score varied, depending on the level of variation in the features. The z-score transformed the datasets into a distribution where the standard deviation is one, and log-scaling transformation obtained a dataset similar standard deviation to z-score. On the contrary, since min-max creates values within the range [0,1], standard deviation and mean values are also in this range. Additionally, it was observed that the z-score scaled the feature space down; however, it did not change the skewness and kurtosis values. The results showed that min-max and log-scaling alter the distribution of the dataset, and both have transformed the datasets with the same skewness and kurtosis values. Finally, the results in Table 6 suggest that log-scaling has led to a better representation of the dataset for ML applications.

4.3. Impact of feature selection

In this section, the impact of the SVM-FS technique is investigated. For this reason, it was applied with the default parameters before model construction. Then SVM-RBF with the default parameters was trained and evaluated using the ranked feature

list obtained by SVM-FS. This experiment was conducted on the normalised NSLKDD, UNSW-NB15, and CICIDS2017 Wednesday datasets. The results of the experiments are presented in Table 7. The best-achieved accuracy for the normalised NSLKDD test datasets was 85.8% and 74.1% using the top eight features. That is, SVM-FS removed 114 features and improved accuracy by 5% for $KDDTest^{+}$ and 10.7% for $KDDTest^{-21}$. Furthermore, for the UNSW-NB15 dataset, 162 features were removed, and the accuracy achieved was maintained. Finally, for the Wednesday CICIDS2017 Wednesday dataset, SVM-FS removed 14 features. In summary, the SVM-FS technique improved the performance of the SVM classifier by selecting the relevant features, and it reduced the number of features used for the model (reduced model complexity).

4.4. Impact of classifier optimisation on SVM's performance

In this section, the impact of classifier optimisation on the performance of the classifier is investigated. Each classification method has some parameters that need to be optimised for the target dataset to build the best-fitted model. Optimised parameters of a classifier are different for each input dataset, showing that this is a dataset-specific process and time-consuming since the parameters have a wide range of possible values for finding the one that fits well. Therefore, searching parameter spaces for the optimal values is an important aspect of classifier optimisation. This study uses a grid search algorithm for SVM classifier parameter optimisation. Since the radial basis function was found to be the best kernel function for this problem domain, the gamma parameter was optimised along with the cost parameter. For computational reasons, a limited search space was created for cost and gamma values, where 10^{6}, 10^{5}, 10^{4}, 10^{3}, 10^{2}, 10^{1}, 10^{0}, 10^{-1}, 10^{-2}, 10^{-3}, and 10^{3}, 10^{2}, 10^{1}, 10^{0}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6} were the members of cost and gamma list, respectively. After that, SVM with predefined cost and gamma values was run, and the accuracies were recorded. Additionally, to observe the impact of FS on classifier optimisation, two different feature sets were used: a reduced set (top eight features) and a complete feature set of the input dataset. The best classifier was found, and the optimised classifier was used to build the classification model. SVM-FS was used for feature ranking. After ranking, the best k features were selected for model construction, where k is 1, 2, 3, 4, 8, 16, 32, 64, and the complete feature set.

Table 7 Performance Evaluation of the SVM-RBF with Reduced Feature Sets in terms of Accuracy

Feature Subset	$KDDTrain^{+}$ Training Set	$KDDTest^{+}$ Testing Set	$KDDTest^{-21}$ Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
Top **1** Feature	89.2%	76.3%	55.7%	90.15%	75.13%	88.22%	88.34%
Top **2** Features	89.3%	75.4%	54.1%	91.70%	76.30%	89.41%	89.48%
Top **3** Features	89.3%	75.4%	54.1%	92.11%	76.40%	91.42%	91.42%
Top **4** Features	89.1%	75.4%	54.1%	92.28%	76.48%	91.63%	91.61%
Top **8** Features	94.6%	**85.8%**	**74.1%**	93.45%	81.01%	87.99%	87.79%
Top **16** Features	98.1%	81.8%	65.9%	93.48%	81.01%	96.19%	95.87%
Top **32** Features	99.0%	78.2%	58.7%	93.60%	81.50%	99.83%	99.74%
Top **64** Features	99.5%	80.7%	63.4%	93.62%	81.53%	**99.88%**	**99.75%**
Top **78** Features	-	-	-	-	-	99.88%	99.74%
Top **122** Features	99.5%	80.8%	63.4%	-	-	-	-
Top **128** Features	**99.9%**	72.4%	47.5%	**93.64%**	**81.55%**	-	-
Top **194** Features	-	-	-	93.64%	81.55%	-	-

As mentioned, default and optimised SVM models were implemented to compare the performance of the SVM classifier and the significance of classifier optimisation with and without feature selection. The first experiment was the SVM-RBF with default parameter settings as the FS technique and classifier. The obtained results are presented in Table 7. Table 8 presents the performance results of the SVM optimised with a reduced feature set. Additionally, the results for SVM optimised using the complete feature set are shown in Table 9. Two different settings of SVM classifier optimisation were developed to accomplish a further analysis for understanding the impact of feature selection in the classifier optimisation process. Accuracy

was used to measure the performance of all the models. However, another performance evaluation metric was used to evaluate the performance of the optimised SVM. Thus, Table 10 represents the performance of optimised SVM in terms of F-score. As shown in Equation 7, the F-score performance evaluation metric uses precision and recall values to calculate the model's performance, which better represents a model's performance where the benchmarking dataset is not balanced in class distribution. The formulation for precision and recall is given in Equations 5 and 6.

The results exhibited that feature selection helps improve classification performance or reduce feature set size. For example, the classification performance of SVM was improved by around 5% for the NSLKDD dataset, where 120 features were removed. On the other hand, 162 features were removed for the UNSW-NB15 dataset, and the classification performance was maintained. On the other hand, classifier optimisation also improved classification performance or reduced feature set size. For instance, classification performance was improved for the UNSW-NB15 dataset by ~4%. Furthermore, classification performance was improved for the CICIDS2017 dataset.

Table 8 Performance Evaluation of the SVM Optimisation with Reduced Feature Set in terms of Accuracy

Feature Subset	$KDDTrain^{+}$ Training Set	$KDDTest^{+}$ Testing Set	$KDDTest^{-21}$ Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.02%	78.02%	60.15%	68.06%	55.06%	88.22%	88.34%
Top **2** Features	95.26%	**81.51%**	**65.68%**	90.59%	75.43%	89.41%	89.48%
Top **3** Features	96.26%	77.79%	58.13%	90.62%	76.08%	91.42%	91.42%
Top **4** Features	98.57%	79.45%	61.33%	90.63%	76.10%	91.63%	91.61%
Top **8** Features	99.67%	76.01%	54.36%	93.45%	80.98%	87.99%	87.79%
Top **16** Features	99.88%	79.20%	60.46%	94.14%	83.01%	96.19%	95.87%
Top **32** Features	99.94%	79.42%	60.96%	94.78%	84.52%	99.83%	99.74%
Top **64** Features	**99.96%**	78.52%	59.32%	**94.98%**	85.25%	**99.88%**	**99.75%**
Top **78** Features	-	-	-	-	-	99.88%	99.74%
Top **122** Features	99.96%	76.99%	56.59%	-	-	-	-
Top **128** Features	-	-	-	**94.98%**	**85.30%**	-	-
Top **194** Features	-	-	-	94.98%	85.30%	-	-

The results showed that classifier optimisation has a notable impact on FS and classifier performance when applied to both FS and classifier. However, optimisation achieved worse performance for the NSLKDD dataset than the default settings, showing the importance of selecting the correct set of parameter values. On the other hand, when the results of classifier optimisation with a complete feature set were compared with the reduced feature set, it was observed that both achieved similar performance. However, in some cases, the use of reduced feature sets negatively affected performance, which is plausible because the selected feature subset is not the best representation of the informative features. It should be noted that, as shown in this study, neither default classifier nor classifier optimisation guarantees the best performance. For example, when the default SVM used for both FS and classification (SVM-RBF, cost = 1, and gamma = 1/feature size), it outperformed the optimised SVM (optimised using Grid-search, SVM-RBF, cost = 10^2 and γ: 10^{-1}).

Table 9 Performance Evaluation of SVM Optimisation with All Features in terms of Accuracy

Feature Subset	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.02%	78.02%	60.15%	90.57%	75.41%	63.65%	63.67%
Top **2** Features	95.26%	**81.51%**	**65.68%**	92.11%	76.57%	80.78%	80.69%
Top **3** Features	96.26%	77.79%	58.13%	92.46%	78.18%	81.94%	81.83%
Top **4** Features	98.57%	79.45%	61.33%	92.68%	78.83%	84.11%	83.92%
Top **8** Features	99.67%	76.01%	54.36%	92.88%	79.12%	96.30%	96.27%
Top **16** Features	99.88%	79.20%	60.46%	94.31%	83.98%	99.43%	99.36%
Top **32** Features	99.94%	79.42%	60.96%	95.03%	85.27%	99.74%	99.66%
Top **64** Features	**99.96%**	78.52%	59.32%	**95.10%**	**85.31%**	**99.79%**	**99.72%**
Top **78** Features	-	-	-	-	-	99.79%	99.72%
Top **122** Features	99.96%	76.99%	56.59%	-	-	-	-
Top **128** Features	-	-	-	95.10%	85.31%	-	-
Top **194** Features	-	-	-	95.10%	85.31%	-	-

Table 10 Performance Evaluation of SVM Optimisation with All Features in terms of F-score

Feature Subset	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.07%	77.68%	69.93%	89.97%	77.91%	84.89%	85.00%
Top **2** Features	94.40%	**81.54%**	**74.84%**	93.55%	81.79%	86.53%	86.58%
Top **3** Features	96.63%	79.80%	72.14%	94.16%	82.96%	88.87%	88.84%
Top **4** Features	98.20%	78.04%	69.27%	94.88%	83.84%	89.16%	89.11%
Top **8** Features	99.46%	74.59%	63.71%	95.30%	84.96%	80.29%	79.89%
Top **16** Features	99.72%	73.74%	62.42%	95.84%	86.92%	94.96%	94.51%
Top **32** Features	99.87%	76.31%	66.39%	96.38%	87.94%	99.77%	99.64%
Top **64** Features	**99.92%**	73.54%	62.32%	96.45%	87.99%	**99.83%**	**99.66%**
Top **78** Features	-	-	-	-	-	99.83%	99.64%
Top **122** Features	99.92%	74.18%	63.32%	-	-	-	-
Top **128** Features	-	-	-	**96.46%**	**88.01%**	-	-
Top **194** Features	-	-	-	96.45%	88.01%	-	-

Table 11 Summary of the Default and Optimised Model on Testing Datasets

Dataset	Preprocessing			Classifier	Performance
	Data Normalisation	Classifier Optimisation	Feature Selection	Optimised SVM Configuration	Accuracy (Selected Feature Size)
NSLKDD	Log-scaling	-	SVM-FS	RBF Kernel C: 1 and γ: 0.008	85.80% (8/122)
UNSW-NB15	Log-scaling	-	SVM-FS	RBF Kernel C: 1 and γ: 0.005	81.50% (32/194)
CICIDS2017 Wednesday	Z-Score	-	SVM-FS	RBF Kernel C: 1 and γ: 0.013	98.48% (64/78)
NSLKDD	Log-scaling	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-1}	81.51% (2/122)
UNSW-NB15	Log-scaling	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-2}	85.30% (64/194)
CICIDS2017 Wednesday	Z-Score	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^5 and γ: 10^0	99.74% (32/78)
NSLKDD	Log-scaling	Applied with Full Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-1}	81.51% (2/122)
UNSW-NB15	Log-scaling	Applied with Full Feature Set	Optimised SVM-FS	RBF Kernel C: 10^0 and γ: 10^{-1}	85.27% (32/194)
CICIDS2017 Wednesday	Z-Score	Applied with Full Feature Set	Optimised SVM-FS	C: 10^4 and γ: 10^{-1}	99.43% (16/122)

However, the optimised configuration removed more features than the default and obtained worse performance, yet a more accurate selection of features. On the other hand, there is a significant improvement in the performance of the default SVM in the CICIDS2017 dataset after optimisation; that is, a 1% higher accuracy was achieved while removing 48 features. Another critical point to emphasise is the FS problem in classifier optimisation. The results exhibited that using a predefined set of features may not result in a better model since the selected feature subset may contain irrelevant features or may not contain all the relevant features, as seen from the results of the NSLKDD dataset.

In conclusion, the results revealed that a detailed FS process must be performed before classifier optimisation. However, it is known that filter FS techniques are not as accurate as embedded/wrapper techniques, and embedded/wrapper techniques suffer from the mentioned FS problem in classifier algorithms since they are classifier-dependent FS techniques. As a result, a potential solution would be the ensemble of filter methods prior to classifier optimisation to remove irrelevant features or a classifier optimisation algorithm incorporating FS. Finally, a summary of the SVM configuration for all setups are given in Table 11.

5. Conclusion

Cybersecurity is an emerging issue in information technologies, and it has become more critical with modern networking and applications. Since Internet users continuously share their sensitive data on the Internet, it is crucial to protect these data. ML-enabled IDSs are promising for this purpose. However, it is challenging for developers to select the appropriate techniques for this problem domain among many possible candidates. Furthermore, developing such an ML system requires a deep understanding of the problem and the input data. To this end, data preprocessing plays a crucial role in developing accurate and efficient ML models. Therefore, this study aimed to analyse the impact of preprocessing techniques on the ML algorithm to find the best-fitting techniques for intrusion detection datasets.

In this study, the SVM was used as a classifier for preprocessing impact analysis due to its outstanding performance and simple implementation. Additionally, SVM was also used as a feature selection technique. For the evaluation of the model, three binary intrusion detection datasets were used, namely NSL-KDD, UNSW-NB15 and CICIDS2017. First, the most

widely used kernel functions (linear, polynomial, sigmoid, and radial basis function) were tested to find the best-fitting kernel function of SVM for intrusion detection. This setting was then used to select the normalisation technique, where the techniques were applied to all datasets. Next, SVM-RBF was performed on each normalised dataset, and the best-performing normalisation technique was found. Finally, FS and classifier optimisation was performed to observe the impact of each on intrusion detection. In summary, the contributions of this study are (1) a detailed analysis of normalisation techniques and their impact on classifier performance, (2) the impact of classifier parameter optimisation on feature selection and classification, and (3) important insights for developing intelligent intrusion detection systems.

It was observed that log-scaling is the best technique for normalising intrusion detection datasets due to their high variance in features. However, when a dataset contains negative values, log-scaling is not applicable. However, the z-score can be an alternative since it achieved a similar performance and outperformed the min-max technique. As a result, data normalisation improves the classification performance in this problem domain. In addition, removing irrelevant features helps to build efficient methods. In conclusion, FS and classifier optimisation improved classification performance and reduced model complexity. However, it was observed that irrelevant features could affect the performance of optimisation algorithms that FS in classifier optimisation must be addressed for building intelligent and efficient IDSs.

Finally, this study can be repeated with more datasets and various ML algorithms, which can be one future direction of this study. Another future direction of this study would be the development of an algorithm that selects features within the optimisation process. A potential solution to this problem would be the ensemble of filter methods prior to classifier optimisation to remove irrelevant features or a classifier optimisation algorithm incorporating feature selection.

References

[1] Ham, Jeroen Van Der. “Toward a Better Understanding of “Cybersecurity”.” Digital Threats: Research and Practice 2.3 (2021): 1-3.

[2] Khraisat, Ansam, et al. “Survey of intrusion detection systems: techniques, datasets and challenges.” Cybersecurity 2.1 (2019): 1-22.

[3] Ahmad, Zeeshan, et al. “Network intrusion detection system: A systematic study of machine learning and deep learning approaches.” Transactions on Emerging Telecommunications Technologies 32.1 (2021): e4150.

[4] Singh, Dalwinder, and Birmohan Singh. “Investigating the impact of data normalisation on classification performance.” Applied Soft Computing 97 (2020): 105524.

[5] Guyon, Isabelle, et al. “Gene selection for cancer classification using support vector machines.” Machine learning 46.1 (2002): 389-422.

[6] Tavallaee, Mahbod, et al. “A detailed analysis of the KDD CUP 99 data set.” 2009 IEEE symposium on computational intelligence for security and defense applications. Ieee, 2009.

[7] Moustafa, Nour, and Jill Slay. “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set.” Information Security Journal: A Global Perspective 25.1-3 (2016): 18-31.

[8] Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. “Toward generating a new intrusion detection dataset and intrusion traffic characterisation.” ICISSp 1 (2018): 108-116.

[9] Zhang, Xiaoyuan, Daoyin Qiu, and Fuan Chen. “Support vector machine with parameter optimisation by a novel hybrid method and its application to fault diagnosis.” Neurocomputing 149 (2015): 641-651.

[10] Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017;5:21954-21961. doi:10.1109/access.2017.2762418.

[11] Tang, Chaofei, Nurbol Luktarhan, and Yuxin Zhao. “SAAE-DNN: Deep learning method on intrusion detection.” Symmetry 12.10 (2020): 1695.

[12] Pervez, Muhammad Shakil, and Dewan Md Farid. “Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs.” The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014). IEEE, 2014.

[13] Janarthanan, Tharmini, and Shahrzad Zargari. “Feature selection in UNSW-NB15 and KDDCUP’99 datasets.” 2017 IEEE 26th international symposium on industrial electronics (ISIE). IEEE, 2017.

[14] Malik, Arif Jamal, Waseem Shahzad, and Farrukh Aslam Khan. “Network intrusion detection using hybrid binary PSO and random forests algorithm.” Security and Communication Networks 8.16 (2015): 2646-2660.

[15] Kanakarajan, Navaneeth Kumar, and Kandasamy Muniasamy. “Improving the accuracy of intrusion detection using gar-forest with feature selection.” Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015. Springer, New Delhi, 2016.

[16] Khammassi, Chaouki, and Saoussen Krichen. “A GA-LR wrapper approach for feature selection in network intrusion detection.” computers & security 70 (2017): 255-277.

[17] Jo, Wooyeon, et al. "Packet preprocessing in CNN-based network intrusion detection system." Electronics 9.7 (2020): 1151. https://doi.org/10.3390/electronics9071151.
[18] Kumar, VD Ambeth. "An Effective Comparative Analysis of Data Preprocessing Techniques in Network Intrusion Detection System Using Deep Neural Networks." Smart Intelligent Computing and Communication Technology 38 (2021): 14.
[19] Ahmad, Tohari, and Mohammad Nasrul Aziz. "Data preprocessing and feature selection for machine learning intrusion detection systems." ICIC Express Lett 13.2 (2019): 93-101.
[20] Nimbalkar, Pushparaj, and Deepak Kshirsagar. "Feature selection for intrusion detection system in Internet-of-Things (IoT)." ICT Express 7.2 (2021): 177-181. https://doi.org/10.1016/j.icte.2021.04.012.
[21] Pajouh HH, Dastghaibyfard GH, Hashemi S. Two-tier network anomaly detection model: A machine learning approach. Journal of Intelligent Information Systems. 2015;48(1):61-74. doi:10.1007/s10844-015-0388-x.
[22] Jabbar AF, Mohammed IJ. Development of an optimised botnet detection framework based on filters of features and machine learning classifiers using CICIDS2017 dataset. IOP Conference Series: Materials Science and Engineering. 2020;928(3):032027. doi:10.1088/1757-899x/928/3/032027.
[23] Krishna KV, Swathi K, Rao BB. A novel framework for nids through fast knn classifier on CICIDS 2017 dataset. International Journal of Recent Technology and Engineering (IJRTE). 2020;8(5):3669-3675. doi:10.35940/ijrte.e6580.018520.
[24] Kshirsagar D, Kumar S. An efficient feature reduction method for the detection of Dos Attack. ICT Express. 2021;7(3):371-375. doi:10.1016/j.icte.2020.12.006.
[25] Azzaoui H, Boukhamla AZ, Arroyo D, Bensayah A. Developing new deep-learning model to enhance network intrusion classification. Evolving Systems. 2021;13(1):17-25. doi:10.1007/s12530-020-09364-z.
[26] Prajapati, Gend Lal, and Arti Patle. “On performing classification using SVM with radial basis and polynomial kernel functions.” 2010 3rd International Conference on Emerging Trends in Engineering and Technology. IEEE, 2010.
[27] Zhang, Xiaoyuan, Daoyin Qiu, and Fuan Chen. “Support vector machine with parameter optimisation by a novel hybrid method and its application to fault diagnosis.” Neurocomputing 149 (2015): 641-651.
[28] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. “A practical guide to support vector classification.” (2003): 1396-1400.
[29] R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
[30] RStudio Team (2019). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.
[31] Meyer, David, et al. “Package ‘e1071’.” The R Journal (2019).

Conflict of interest

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

Funding

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

Data Availability

The datasets generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

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 ™.