Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 6 No. 3 (2022) 387 - 396 ISSN Media Electronic: 2580-0760

Implementation of CNN-MLP and CNN-LSTM for MitM Attack Detection System

Hartina Hiromi Satyanegara[1], Kalamullah Ramli[2]
[1,2] Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia
[1]hartina.hiromi@ui.ac.id, [2]kalamullah.ramli@ui.ac.id

Abstract

Man in the Middle (MitM) is one of the attack techniques conducted for eavesdropping on data transitions or conversations between users in some systems secretly. It has a sizeable impact because it could make the attackers will do another attack, such as website or system deface or phishing. Deep Learning could be able to predict various data well. Hence, in this study, we would like to present the approach to detect MitM attacks and process its data, by implementing hybrid deep learning methods. We used 2 (two) combinations of the Deep Learning methods, which are CNN-MLP and CNN-LSTM. We also used various Feature Scaling methods before building the model and will determine the better hybrid deep learning methods for detecting MitM attack, as well as the feature selection methods that could generate the highest accuracy. Kitsune Network Attack Dataset (ARP MitM Ettercap) is the dataset used in this study. The results prove that CNN-MLP has better results than CNN-LSTM on average, which has the accuracy rate respectively at 99.74%, 99.67%, and 99.57%, and using Standard Scaler has the highest accuracy (99.74%) among other scenarios.

Keywords: MitM, Kitsune Network Attack Dataset, CNN-MLP, CNN-LSTM

1. Introduction

Technology growth is more sophisticated and rapid in this era, and it makes many people can get the desired information easier and more efficient. But unfortunately, some people misuse the technology itself to get profit for them and loss for the victim at the same time. The attacks could be occured on digital assets, such as websites, email, social media, information systems, etc.

MitM (Man in the Middle) attacks is one of the attacks by accessing the target system, and usually, the attackers are in between the data transmission or conversations between 1 (one) user to another user or server. Subsequently, they could conduct another attack, such as phishing, by sending an email to the victim to access the link from the attacker, as if it is from an official company or institution [1].

It has a sizeable impact because it could create a way for other attacks. Based on X-Force Threat Intelligence Index 2022 created by the IBM Security team, it is mentioned that Mozi botnet attacks that target IoT (Internet of Things) and OT (Operational Technology), the attackers can conduct MitM attacks by infecting routers to spread ransomware to IoT and OT [2].

Usually, to prevent and minimalize the incoming attacks on our system, we could be using NIDS (Network-based Intrusion Detection System). NIDS is 1 (one) type of the IDS (Intrusion Detection System), besides anomaly-based and signature-based, that monitors the network traffic, and it can detect whether the traffic is benign or malicious and suspicious for our system [3]. Besides that, NIDS could detect the network traffic by comparing the pattern or signature of the database. NIDS is also referred to as packet-sniffer [4].

Deep Learning (DL) is a subset of Machine Learning (ML), where deep learning is an important part of Artificial Intelligence (AI) [5]. Deep learning can process the data from any discipline, gives decision-making, and also gives prediction results of data. Deep learning has 3 (three) major categories: unsupervised, partially supervised (semi-supervised), and supervised [6]. The examples of deep learning methods are Convolutional Neural Network (CNN), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Generative Adversarial Networks (GAN), Deep Reinforcement Learning (DRL), Self-Organizing Maps (SOM), etc.

Accepted: 21-04-2022 | Received in revised: 07-06-2022 | Published: 30-06-2022

This study cannot be separated from the prior research. There are many kinds of research about malware detection with deep learning method approaches.

In 2018, the research entitled "Evaluation of Convolutional Neural Networks Features for Malware Detection" conducted by Kemal Özkan, et al. This research aims to search for the connection between malware binaries on the grayscale color representative that developed with feature extraction and classifiers (Softmax, KNN, and SVM) with Convolutional Neural Networks (CNN). The structure used in this research is VGG16. They are using SARVAM as the used dataset. The result is CNN is a method that can be used to classify malware, which generates the accuracy of 85% when implemented to 36 malware families that consist of 12.279 malware samples, and when implemented to 25 malware families that consist of 9.339 malware families, the accuracy into 99% [7].

In the same year (2018), the research entitled "Intrusion Detection via MLP Neural Network using an Arduino Embedded System" was conducted by Felipe de Almeida Florencio, et al. This research aims to implement MLP for IDS using NSL-KDD as the dataset and using Weka software. They are using nominal features to numerical values and feature elimination from 41 to 26 and have 2 (two) experiments: Test 1 (75% of the sample) and Test 2 (25% of the sample). The measurements are divided into model and time performance. Model measurements are consisting of accuracy, precision, coverage, and f1-score, and performance measurements are consisting of mean, standard deviation, variance, and confidence interval. The results are the value of Test 1 and Test 2 such as accuracy (97.14% (Test 1) and 59.02% (Test 2)), precision (98.6% (Test 1) and 96.1% (Test 2)), coverage (95.2% (Test 1) and 52.1% (Test 2)), f1-score (96.9% (Test 1) and 67.6% (Test 2)), mean (5716.04 µs), standard deviation (66.38 µs), variance (4406.86 µs), and confidence interval (5711.88 µs; 5720.20 µs) [8].

In 2019, the research entitled "LSTM deep learning method for network intrusion detection system" was conducted by Alaeddine Boukhalfa, et al. This research aims to implement the Long Short-Term Memory (LSTM) method to Network Intrusion Detection System with NSL-KDD dataset, then it will be compared with other 3 (three) classifiers that consists of Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Decision Tree. The metrics that will be used are accuracy, sensitivity, recall, precision, and f-measure. The results are LSTM with 2 (two) classes and 4 (four) classes have the highest accuracy (99.98% and 99.93%), sensitivity (99.986% and 99.738%), recall (Normal: 99.973% and 99.938%; Attack: 99.998% (2 classes), 99.906% (4 classes-DoS), 99,106% (4 classes-U2R, R2L), and 100% (4 classes-Probe), precision (Normal: 99.999% and 100%; Attack: 99.969% (2 classes), 99.924% (4 classes-DoS), 95.896% (4 classes-U2R, R2L), and 99,863% (4 classes-Probe), and f-measure (Normal: 99.986% and 99.969%; Attack: 99.983% (2 classes), 99.915% (4 classes-DoS), 97.475% (4 classes-U2R, R2L), and 99.931% (4 classes-Probe)) value, while SVM has the highest False Positive Rate (1.765% and 0.493%) value [9].

In the same year (2019), the research entitled "Detection and Prevention of Man-in-the-Middle Spoofing Attacks in MANETs Using Predictive Techniques in Artificial Neural Networks (ANN)" was conducted by Robert A. Sowah, et.al. This research aims to detect MitM attacks in MANET with ANN, by using NS2 as the simulation platform. The performance metrics that had been used are recall, precision, accuracy, and f-measure. They are using 7 to 18 nodes as the experiment scenarios. The result stated that it could generate accuracy rates in the range of around 79%-93% from 7-18 nodes [10].

In 2020, the research entitled "Android Malware Detection Based on a Hybrid Deep Learning Model" was conducted by Tianliang Lu, et al. This research aims to detect Android malware using the combination of Deep Belief Network (DBN) and Gate Recurrent Unit (GRU). The dataset used in this research is divided into benign and malware samples. The number of benign samples is 7000, while the number of malware samples is 6298. The results are with using DBN-GRU to detect Android malware could generate higher accuracy (96.82%), precision (Benign-97.79% and Malware-95.79%), and recall (Benign-96.09% and Malware-97.62%) among deep learning or machine learning method, even if using DBN or GRU only, and have the higher accuracy when did repackage on the malware among the antivirus software [11].

In 2021, the research entitled "Intrusion Detection System using MLP and Chaotic Neural Networks" was conducted by Pooja Shettar, et al. This research aims to implement Multilayer Perceptron (MLP) method and the combination of MLP and Chaotic Neural Networks, using KDD Cup'99 dataset. The metrics consist of accuracy, precision, FPR, and FNR. The result is that the hybrid value from MLP and Chaotic Neural Network has higher accuracy (99.21% and 94.8%) and precision (99.91% and 90.3%), and lower FPR (0.00401 and 0.00478) and FNR (0.00213 and 0.00233) from MLP itself [12].

Deep Learning could be able to predict various data well, especially for detecting MitM [10]. Hence, in this study, the MitM attack data is being processed and analyzed [10]. Since the result of hybrid deep learning method research can generate higher values than using only 1 (one) deep learning method only [11][12], therefore we are using an approach from the combination of another 2 (two) deep learning methods, which are CNN-MLP and CNN-LSTM, so we could

DOI: https://doi.org/10.29207/resti.v6i3.4035

know whether by using a combination from CNN [7], MLP [8], and LSTM [9] methods still could generate the great values. We choose CNN, MLP, and LSTM methods to be combined in this study based on the research from [7][8][9], which is the methods in their research that can handle data in large amounts with sizeable accuracy and other performance metrics. Most of these methods could generate an accuracy rate above 99%, and we will know whether using combining CNN with MLP and LSTM still could generate 99%. The dataset that will be used in this research is ARP MitM Ettercap, which is from Kitsune Network Attack Dataset. The type of MitM that will be used in this research is ARP Spoofing. There are 3 (three) scenarios to process the data, which are based on the Feature Scaling that consists of Standard Scaler, Min-Max Scaler, and Maximum Absolute Scaling (MaxAbsScaler). The Feature Scaling will be set on data preprocessing. The purpose of this study is to determine the accuracy, recall, precision, and f1-Score using CNN-MLP and CNN-LSTM methods for processing the ARP MitM attack detection, then we could compare which method has the highest accuracy rate after doing model testing.

The rest of this journal is organized as follows: Section 1 explains the Introduction. Section 2 explains research methods. Section 3 explains the result and discussion of the study. Section 4 explains the conclusion of the study and future works.

2. Research Methods

This section will be discussed about the dataset information, the research model flow, data preprocessing, and the methods (CNN-MLP and CNN-LSTM).

2.1 Dataset and Research Model

In this study, we are using the ARP MitM Ettercap dataset from Kitsune Network Attack Dataset. This dataset is developed by Yisroel Mirsky, et al. in 2019, from their journal entitled "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection". It has 9 (nine) network attacks that consist of OS Scan Nmap, Fuzzing Sfuzz, Video Injection, ARP MitM Ettercap, Active Wiretap, SSDP Flood, SYN DoS, SSL Renegotiation, and Mirai Telnet. Each attack has 3 (three) files: PCAP file, dataset file, and Label file. The dataset is available on UCI Machine Learning Repository website since October 16th, 2019 [13][14]. We are using 2 (two) files of ARP MitM for data processing: **data** and **label** files. It has a total of 116 features, where the dataset file has 115 features and the label file have 1 feature. Then, these files will be merged into 1 (one) file. The information about the dataset is shown in Table 1.

Table 1. Dataset Information

Attack file	Samples	Features	File Size
ARP MitM Ettercap Dataset	2504267	115	7.04 GB
ARP MitM Ettercap Labels	2504267	1	30.7 MB

The data in this dataset is divided into 2 (two) traffic statuses: Benign and Malicious. This status is shown in the 'label' column, where '0' value is stated as Benign or normal traffic, whereas '1' value is stated as Malicious traffic or got MitM attack. We will combine the benign and malicious traffic data to process in CNN-MLP and CNN-LSTM. The traffic (benign and malicious) percentage and sample number information is shown in Table 2.

Table 2. Percentage of Benign and Malicious Samples in ARP MitM Dataset (Label)

Traffic	Percentage	Number of Samples
Benign	54,3%	1358995
Malicious	45,7%	1145272
Total		2504267

Firstly, we collected the dataset from Kitsune Network Attack Dataset, and process the ARP MitM Ettercap Dataset; On data preprocessing, it contains NA values drop, INF values drop, determine x and y values, Train-Test Split, Feature Scaling (Standard Scaler, Min-Max Scaler, and Maximum Absolute Scaling (MaxAbsScaler)), One Hot Encoding, and reshape the x_train and x_test. The Model is divided into 2 (two) hybrid deep learning methods, which are CNN-MLP and CNN-LSTM, then we will train the models to know the accuracy and loss rates. The last step is to test the model by evaluate the model and compare with model training results, and then we were setting up the prediction to know the rates of accuracy, recall, precision, and F1-Score. The workflow of this research is shown in Figure 1.

2.2 Data Preprocessing

After the dataset and label files have been merged, the next step is data processing. In this step, we are managing the dataset that will be used before the dataset is being implemented in the models. There are 7 (seven) steps of data preprocessing for this study, such as drop NA values, drop INF values, determine x and y values, train-test split, feature scaling, one hot encoding, and reshape the x_train and x_test shapes.

We removed NA (Not Available) or null and INF (Infinite) values first, then determining the x and y values, which are the x values will be implemented to 115 columns (main data), whereas the y values will be implemented to 1 column (label data).

On Train-Test Split [15], we are splitting the data train and test from the merged dataset, in which the percentage of training data is 60% and testing data is 40% [16], and the data would be randomly shuffled.

DOI: https://doi.org/10.29207/resti.v6i3.4035

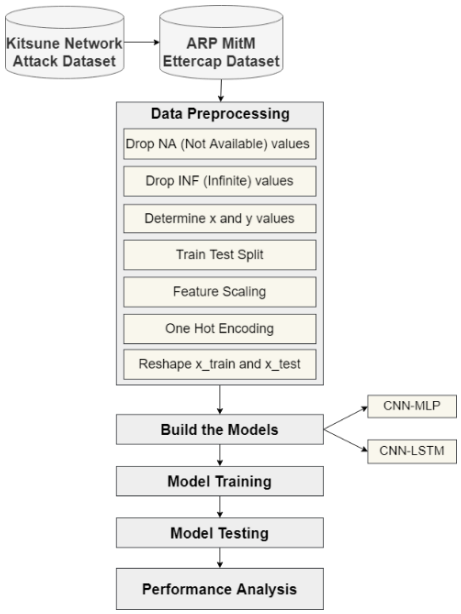

Figure 1. Research Flow Model

Feature Scaling [17] is the next method used for merging the self-variables or feature ranges in data [18], and it is involving value rescaling to make the data processing easier [19]. The StandardScaler, MinMaxScaler, and MaxAbsScaler [19] are used as the form of feature scaling for x_train and x_test values. StandardScaler is a feature scaling method that adjusts the information inside of each component and will scale them in such a way that makes the current dissemination is revolved around 0, with a standard deviation of 1 [19]. MinMaxScaler is a feature scaling method that widely recognized approach to standardizing the data, where the base estimation for each component changes to 0 and the most extreme estimation changes to 1, and each other is changed in the range of 0 and 1 [19]. MaxAbsScaler is a feature scaling method that contrast difference from other feature scaling that the outright quality is mapped in the range of 0 and 1 [19]. By using feature scaling, the accuracy rate could be increased. Based on the research from [19], before implementing any of the feature scaling methods, the accuracy rate is in the range of 67%-69%, but when the feature scaling methods has been applied, respectively, the accuracy rates are increased to the range of 70%-75% (StandardScaler); 72%-78% (MinMaxScaler); and 71%-77% (MaxAbsScaler). It also exist on f1-score, recall, and precision the before implement any of feature scaling method, respectively, the f1-score, recall, and precision rates are in range of 0.3398-0.6449, 0.5-0.6417, and 0.4046-0.6431, and after the feature scaling methods has been applied, respectively, the f1-score, recall, and precision rates are increased to the range of 0.6539-0.7193, 0.648-0.6827, 0.6505-0.6934 (StandardScaler); 0.6833-0.7843, 0.58-0.6943, 0.5689-0.7122 (MinMaxScaler); 0.6945-0.7785, 0.5679-0.6807, 0.5397-0.6972 (MaxAbsScaler).

One Hot Encoding is transforming a single variable with n observations and d distinct values, to d binary variables with each of n observations [20]. On One Hot Encoding, it is applied to y_train and y_test values, which is y_train and y_test values are contains label column, so it could recognize the label whether the condition is 0 or 1.

The last step in data preprocessing is reshaped the x_train shape from (x_train.shape[0], 115) into (x_train.shape[0], 115, 1), and reshaped the x_test shape from (x_test.shape[0], 115) into (x_test.shape[0], 115, 1). The total of x_train.shape[0] is 1502560, whereas the total of x_test.shape[0] is 1001707. The shape of x_train and x_test needs to be changed, so it could be process when build the models.

2.3. CNN-MLP and CNN-LSTM

CNN-MLP and CNN-LSTM are the models that will be implemented in this study. Convolutional Neural Networks (CNN) is a deep learning model for processing the data that has grid patterns, and it is designed to learn spacial feature hierarchy automatically and adaptive, from low-level patterns to high-level patterns. CNN typically has 3 (three) layers that consist of convolution, pooling, and a fully connected layer [21]. Multilayer Perceptron (MLP) is a deep learning model that is categorized as a feed-forward neural network with 1 (one) or more hidden layer(s). The layer of MLP consists of an input layer, hidden layer(s), and an output layer. The backpropagation training algorithm is also used for training MLP [22]. MLP has the ability in developing non-linear models with high complexity, and the layer consist of input, hidden, and output [23]. Long Short-Term Memory (LSTM) is the form of deep Recurrent Neural Network that learns order dependence within sequential data and usually used for time-series data classification [24].

Either using CNN-MLP or CNN-LSTM has the same input shape: (None, 115, 1). The first model that will be build is CNN-MLP. On CNN-MLP, in the CNN part, we are using 2 (two) 1D CNN layers [25], 2 (two) Max Pooling layers [26], 1 (one) Dropout layer [27], and 1 (one) Dense layer by using ReLU as the activation [28], whereas in MLP part, we are using 2 (two) hidden layers, 1 (one) Dropout layer [27], and 1 (one) Dense layer by using ReLU as the activation [28], then we are going to Fully Connected layer process, which is going to Flatten layer [29] first, then using Sigmoid as Activation in Dense layer [30]. The Output layer is the last layer of CNN-MLP that will be through the training phase using some parameters that are needed and will determine the performance results in the testing phase. The model flow of CNN-MLP is shown in Figure 2.

DOI: https://doi.org/10.29207/resti.v6i3.4035

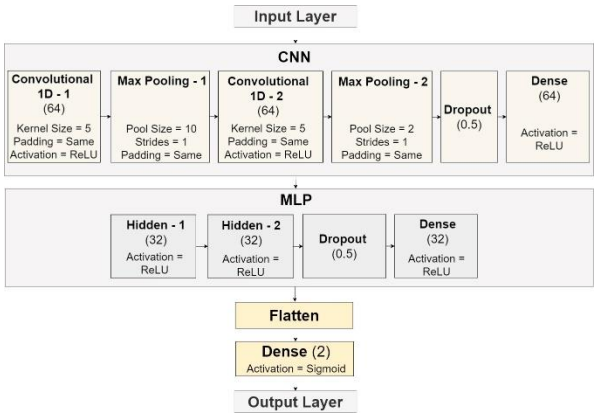

Figure 2. CNN-MLP Model Flow

The CNN-LSTM is the second model that will be build. On CNN-LSTM, in the CNN part, same as in CNN-MLP, we are using 2 (two) 1D CNN layers [25], 2 (two) Max Pooling layers [26], 1 (one) Dropout layer [27], and 1 (one) Dense layer by using ReLU as the activation [28], whereas in LSTM part, we are using 2 (two) LSTM layers, 1 (one) Dropout layer [27], and 1 (one) Dense layer by using ReLU as the activation [28], then we are going to Fully Connected layer process, which is also going to Flatten layer [29] first, then using Sigmoid as Activation in Dense [30]. The Output layer is the last layer of CNN-LSTM that will be through the training phase using some parameters that are needed and will determine the performance results in the testing phase. The model flow of CNN-LSTM is shown in Figure 3.

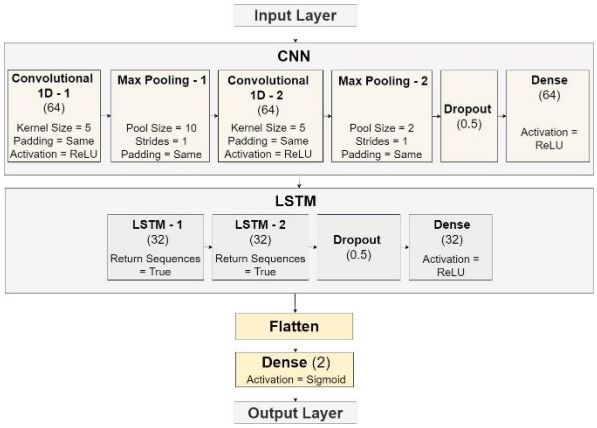

Figure 3. CNN-LSTM Model Flow

By implementing these models (CNN-MLP and CNN-LSTM) for detecting MitM attacks, we will know which model has generating the better performance metrics with different proposed Feature Scaling methods. After building the models, the next steps in data processing are training the models by using the parameter and its values, and then we are going to test the models to perform the results.

3. Results and Discussions

This section will be discussed about the experimental setup, model training and testing process, the performance metrics that will be used, and the results of this study.

3.1. Experimental Setup

To implementing the CNN-MLP and CNN-LSTM models using the ARP MitM Ettercap dataset, we are using Python (ver. 3.10.2) – Jupyter Notebook [31] in Visual Studio Code, with TensorFlow and Keras [32] to create CNN-MLP and CNN-LSTM models. This software runs in a PC with the specifications such as Windows 10 Pro 64-bit, 11th Gen Intel® Core (TM) i5-11400 @ 2.60GHz (12 CPUs) ~2.6 GHz, 16 GB RAM. We have implemented 3 (three) scenarios in our experiments, which is using different Feature Scaling that consist of Standard Scaler (StandardScaler), Min-Max Scaler (MinMaxScaler), and Maximum Absolute Scaling (MaxAbsScaler) with CNN-MLP and CNN-LSTM models.

3.2. Model Training

The next step after data preprocessing is building the CNN-MLP and CNN-LSTM models, using (None, 115, 1) as the inputs. After each model has been created, the next step is the model training phase, using the parameters to support this model process, consisting of Epoch, Batch Size, Optimizer, Loss, Learning Rate, Validation Split, and Metric. as shown in Table 3. We also use Validation Split (0.2), for displaying the validation accuracy and loss in each epoch iteration in the training phase.

Table 3. Parameter Values of Model Training

Parameter	Value
Epoch	5
Batch Size	512
Optimizer	Adam
Loss	Binary Cross-Entropy
Learning Rate	1e-4
Validation Split	0.2
Metric	Accuracy and Loss

While doing training phase on the ARP MitM dataset using these parameters, the training time of CNN-MLP is way faster than CNN-LSTM, using various Feature Scaling methods. CNN-MLP needs around 50 minutes, while CNN-LSTM needs around 200 minutes overall. For each epoch iteration, using CNN-MLP needs around 500 seconds, while CNN-LSTM needs around 3000 seconds.

The Model accuracy during the training session using CNN-MLP and CNN-LSTM along with various Feature Scaling methods will be combined into a plot graph that has been generated and it consists of training (blue line) and validation (orange line), that shows the

DOI: https://doi.org/10.29207/resti.v6i3.4035

accuracy value on each epoch iteration in Figure 4 until Figure 9.

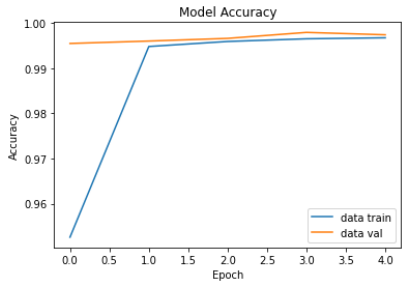

Figure 4. Model Accuracy of CNN-MLP using StandardScaler

In Figure 4, with 5 (five) epoch iterations on the CNN-MLP model using the StandardScaler method, the accuracy rates of training keep increasing from 1st until 5th epoch iterations, while the accuracy of validation had increased from 1st until 4th epoch but had decreased from 4th until 5th epoch iterations. The accuracy rates of training are still stable. However, the accuracy rates of validation are unstable, but still good enough. The average of accuracy rates between training and validation are still above 0.99. This model accuracy graph is most stable among other accuracy graphs.

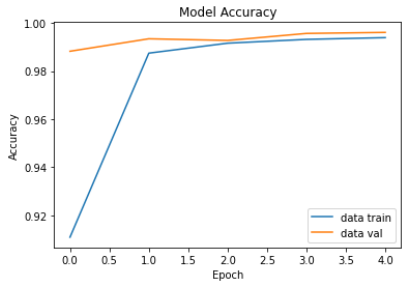

Figure 5. Model Accuracy of CNN-MLP using MinMaxScaler

In Figure 5, with 5 (five) epoch iterations on the CNN-MLP model using MinMaxScaler, the accuracy of training keeps increasing from 1st until 5th epoch iterations, while the accuracy of validation had increased from 1st until 2nd epoch iterations, but had decreased from 2nd until 3rd epoch iterations, and from 3rd until 5th epoch iterations, it has increased. The accuracy rates of training are still stable. However, the accuracy rates of validation are unstable. The average of accuracy rates is still above 0.99.

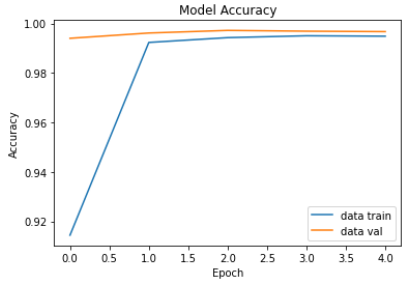

Figure 6. Model Accuracy of CNN-MLP using MaxAbsScaler

In Figure 6, with 5 (five) epoch iterations on the CNN-MLP model using MaxAbsScaler, the accuracy of training had increased from 1st until 4th epoch iterations but had decreased from 4th until 5th epoch iterations, while the accuracy of validation had increased from 1st until 4th epoch iterations but had decreased from 4th until 5th epoch iterations. Overall, the rates between training and validation graph are quite stable, although it has been decreased at once on training and validation, and the accuracy rates are still above 0.99.

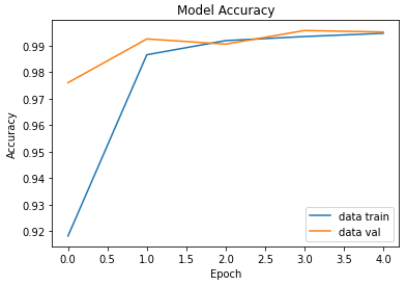

Figure 7. Model Accuracy of CNN-LSTM using StandardScaler

In Figure 7, with 5 (five) epoch iterations on the CNN-LSTM model using StandardScaler, the accuracy of training keeps increasing from 1st until 5th epoch iterations, while the accuracy of validation had increased from 1st until 2nd epoch iterations, decreased from 2nd until 3rd epoch iterations, increased from 3rd until 4th epoch iterations, and then had decreased from 4th until 5th epoch iterations. The accuracy rates of training are still stable. However, the accuracy rates of validation are unstable. The average of accuracy rates is still above 0.99 after the 2nd epoch, in training and validation.

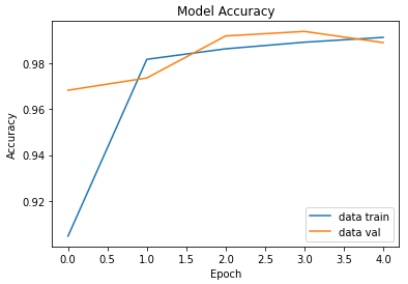

Figure 8. Model Accuracy of CNN-LSTM using MinMaxScaler

In Figure 8, with 5 (five) epoch iterations on the CNN-LSTM model using MinMaxScaler, the accuracy of training keeps increasing from 1st until 5th epoch iterations, while the accuracy of validation had increased from 1st until 4th epoch iteration but had decreased from 4th until 5th. The accuracy rates of training are still stable. However, the accuracy rates of validation are unstable, especially if we compared to the prior graphs because the validation rates have significantly differenced on each epoch iterations. The average of accuracy rates is still above 0.98.

DOI: https://doi.org/10.29207/resti.v6i3.4035

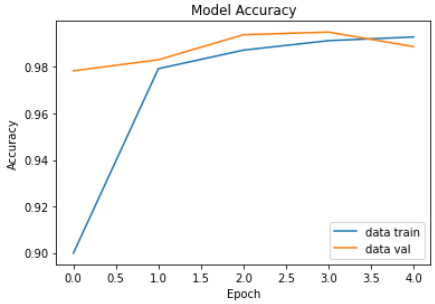

Figure 9. Model Accuracy of CNN-LSTM using MaxAbsScaler

In Figure 9, with 5 (five) epoch iterations on the CNN-LSTM model using MaxAbsScaler, the accuracy of training keeps increasing from 1st until 5th epoch iterations, while the accuracy of validation had increased from 1st until 4th epoch iterations but had decreased from 4th until 5th. The accuracy rates of training are still stable. However, the accuracy rates of validation are unstable, especially if we compared to the prior graphs because the validation rates have significantly differenced on each epoch iterations.

Besides the accuracy rates, we also knew the loss rates of each scenario. The Model Loss during the training phase with CNN-MLP and CNN-LSTM models along with various Feature Scaling methods will be combined into plot graphs that has been generated, and it consists of training (blue line) and validation (orange line) results. Model loss is one of the important measurements in model training, because the model loss could be determined if the model is overfitting, underfitting, or good fit. Therefore, the plot graph shows the loss value on each epoch iterations in Figure 10 until Figure 15.

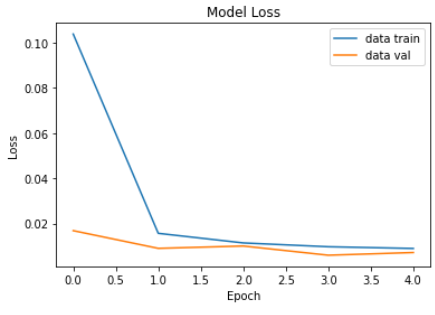

Figure 10. Model Loss of CNN-MLP using StandardScaler

In Figure 10, with 5 (five) epoch iterations on the CNN-MLP model using StandardScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations, while the loss of validation had decreased and increased on every epoch iteration, from 1st until 5th epoch iterations. The loss rates of training are still stable. However, the loss rates of validation are unstable. The average of loss rates is still below 0.02.

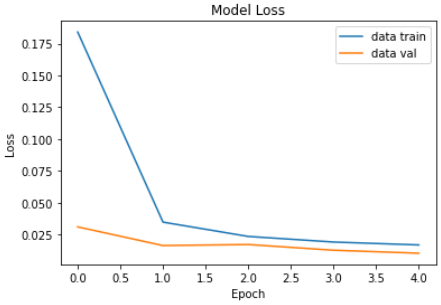

Figure 11. Model Loss of CNN-MLP using MinMaxScaler

In Figure 11, with 5 (five) epoch iterations on the CNN-MLP model using MinMaxScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations, while the loss of validation had decreased from 1st until 2nd epoch iterations, then increased from 2nd until 3rd epoch iterations, and from 3rd until 5th, it keeps decreasing. The loss rates of training are still stable. However, the loss rates of validation are unstable. The average of loss rates is still below 0.05.

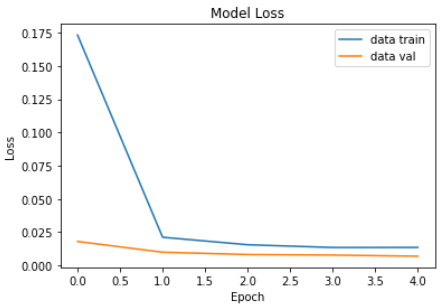

Figure 12. Model Loss of CNN-MLP using MaxAbsScaler

In Figure 12, with 5 (five) epoch iterations on the CNN-MLP model using MaxAbsScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations, while the loss of validation also keeps decreasing as well. Overall, the rates between training and validation graph are still stable, and the loss rates are still below 0.025. This graph is the most stable among other loss graphs because training and validation rates are keep decreasing as well on each epoch iterations.

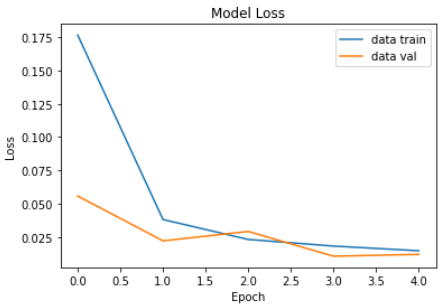

Figure 13. Model Loss of CNN-LSTM using StandardScaler

In Figure 13, with 5 (five) epoch iterations on the CNN-LSTM model using StandardScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations,

DOI: https://doi.org/10.29207/resti.v6i3.4035

while the loss of validation had decreased and increased on every epoch iteration. The loss rates of training are still stable. However, the loss rates of validation are unstable, and it has significantly differenced on the rates in each epoch iterations.

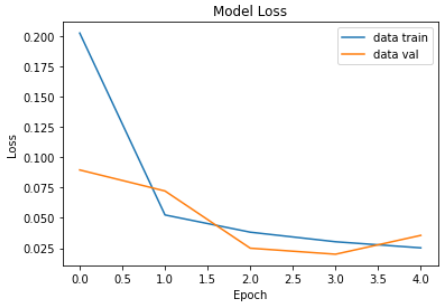

Figure 14. Model Loss of CNN-LSTM using MinMaxScaler

In Figure 14, with 5 (five) epoch iterations on the CNN-LSTM model using MinMaxScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations, while the loss of validation had decreased from 1st until 4th epoch iterations and increased from 4th until 5th epoch iterations. The loss rates of training are still stable. However, the loss rates of validation are unstable, and it has significantly differenced on the rates in each epoch iterations.

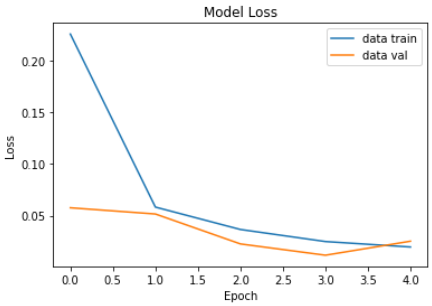

Figure 15. Model Loss of CNN-LSTM using MaxAbsScaler

In Figure 15, with 5 (five) epoch iterations on the CNN-LSTM model using MaxAbsScaler, the loss of training keeps decreasing from 1st until 5th epoch iterations, while the loss of validation had decreased from 1st until 4th epoch iterations and increase from 4th until 5th epoch iterations. The loss rates of training are still stable. However, the loss rates of validation are unstable, and it has significantly differenced on the rates in each epoch iterations.

From overall graphs (model accuracy and loss) that have been generated, most of the model that has been built is indicated as the overfitting models, although it has some slight difference between training and validation. It occurred because the model could be complicated when training its data, that could make the performance results of validation or test become poor [33].

3.3. Performance Metrics

The performance metrics used in this study are accuracy, precision, recall, and f1-Score, based on the research from [7], [8], [9], [10], [11], and [12]. Accuracy is the percentage of samples that are correctly classified above the total number of samples. Recall or sensitivity is to reflects the ability of systems to detect anomalies. Precision is the predictably positive predictor ratio to total positive observation predictions. Whereas F1-Score is the weighted average precision and recall [34].

3.4. Results

After model testing was conducted, the Accuracy, Precision, Recall, and F1-Score results were obtained from CNN-MLP and CNN-LSTM to ARP MitM attack detection, based on Standard Scaler, Min-Max Scaler, and Abs Max Scaler, shown in Table 4, Table 5, and Table 6. We separated the Precision, Recall, and F1-Score based on the label that contains 1 (MitM Attack) and 0 (Benign).

Table 4. Performance Results from CNN-MLP and CNN-LSTM using Standard Scaler

Model	Label	Accuracy	Precision	Recall	F1-Score
CNN-MLP	0	**99.74%**	1.00	1.00	1.00
	1		1.00	1.00	1.00
CNN-LSTM	0	99.44%	1.00	0.99	0.99
	1		0.99	0.99	0.99

Based on the testing results by using Standard Scaler on CNN-MLP and CNN-LSTM models, both models got well rates of accuracy, precision, recall, and f1-Score, which is generated above 99%. CNN-MLP generated a higher accuracy rate than CNN-LSTM with a percentage of 99.74%.

Table 5. Performance Results from CNN-MLP and CNN-LSTM using Min-Max Scaler

Model	Label	Accuracy	Precision	Recall	F1-Score
CNN-MLP	0	**99.67%**	0.99	1.00	1.00
	1		1.00	0.99	1.00
CNN-LSTM	0	99.40%	0.99	1.00	0.99
	1		1.00	0.99	0.99

Based on the testing results by using Min-Max Scaler on CNN-MLP and CNN-LSTM models, both models got well rates of accuracy, Precision, Recall, and F1-Score which is generated above 99%. CNN-MLP generated a higher accuracy rate than CNN-LSTM with a percentage of 99.67%.

Table 6. Performance Results from CNN-MLP and CNN-LSTM using Abs Max Scaler

Model	Label	Accuracy	Precision	Recall	F1-Score
CNN-MLP	0	**99.57%**	0.99	1.00	1.00
	1		1.00	0.99	1.00
CNN-LSTM	0	98.68%	1.00	0.98	0.99
	1		0.97	1.00	0.99

DOI: https://doi.org/10.29207/resti.v6i3.4035

Based on the testing results by using Min-Max Scaler on CNN-MLP and CNN-LSTM models, both models got the good rates of accuracy, precision, recall, and f1-Score which is generated above 99%. CNN-MLP generated a higher accuracy rate than CNN-LSTM with a percentage of 99.57%.

For overall scenarios that have been conducted in this study, using CNN-MLP with StandardScaler got the highest accuracy rate among other methods, which is got a percentage of 99.74%.

4. Conclusion

This study aims to implement the hybrid deep learning methods (CNN-MLP and CNN-LSTM) to detect ARP MitM using Kitsune Network Attack Dataset and has 3 (three) scenarios based on the used feature selection (StandardScaler, MinMaxScaler, and MaxAbsScaler). The results prove that CNN-MLP can generate better a accuracy rate than CNN-LSTM (99.74%, 99.67%, and 99.57%, respectively). For overall scenarios, using CNN-MLP with Standard Scaler could achieve the highest accuracy rate among other scenarios (99.74%). We also generate the accuracy and loss of each scenario using the graphs, which state that mostly, the training has stable values on each epoch iteration and the validation has unstable values on each epoch iteration. Most of the graphs are indicated as the overfitting models, although it has some slight difference between training and validation, especially in CNN-MLP model.

For future works, it could implement other hybrid deep learning or machine learning methods on ARP MitM attack detection. We could also implement feature selection or extraction methods for the dataset to stable the validation results in the training phase and reduce the overfitting or underfitting model, as well as combine the ARP MitM Ettercap dataset with other MitM attacks in the Kitsune dataset (Video Injection and Active Wiretap) or combine the entire dataset (Kitsune NIDS) to be processed.

Reference

[1] A. Mallik, A. Ahsan, M. M. Z. Shahadat, and J. C. Tsou, "Man-in-the-middle-attack: Understanding in simple words," *International Journal of Data and Network Science*, vol. 3, no. 2, pp. 77–92, 2019, doi: 10.5267/j.ijdns.2019.1.001.

[2] C. Singleton *et al.*, "X-Force Threat Intelligence Index 2022," 2022. [Online]. Available: https://www.ibm.com/downloads/cas/ADLMYLAZ

[3] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.

[4] B. Neyole Misiko Jacob, M. Yusuf Wanjala, N. Misiko Jacob α, and M. Yusuf Wanjala σ, "A Review of Intrusion Detection Systems," C, 2017. [Online]. Available: https://globaljournals.org/GJCST_Volume17/3-A-Review-of-Intrusion-Detection.pdf

[5] B. Huy, N. Q. Truong, N. Q. Khiem, K. P. Poudel, and H. Temesgen, "Deep learning models for improved reliability of tree aboveground biomass prediction in the tropical evergreen broadleaf forests," *Forest Ecology and Management*, vol. 508, Mar. 2022, doi: 10.1016/j.foreco.2022.120031.

[6] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[7] K. Özkan, Ş. Işik, and Y. Kartal, "Evaluation of convolutional neural network features for malware detection," in *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, May 2018, vol. 2018-January, pp. 1–4. doi: 10.1109/ISDFS.2018.8355390.

[8] F. de Almeida Florencio, E. D. Moreno, H. T. MacEdo, R. J. P. de Britto Salgueiro, F. B. do Nascimento, and F. A. O. Santos, "Intrusion detection via MLP neural network using an arduino embedded system," in *Brazilian Symposium on Computing System Engineering, SBESC*, Jul. 2018, vol. 2018-November, pp. 190–195. doi: 10.1109/SBESC.2018.00036.

[9] A. Boukhalfa, A. Abdellaoui, N. Hmina, and H. Chaoui, "LSTM deep learning method for network intrusion detection system," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 3, pp. 3315–3322, 2020, doi: 10.11591/ijece.v10i3.pp3315-3322.

[10] R. A. Sowah, K. B. Ofori-Amanfo, G. A. Mills, and K. M. Koumadi, "Detection and prevention of man-in-the-middle spoofing attacks in MANETs using predictive techniques in Artificial Neural Networks (ANN)," *Journal of Computer Networks and Communications*, vol. 2019, 2019, doi: 10.1155/2019/4683982.

[11] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/8863617.

[12] P. Shettar, A. v. Kachavimath, M. M. Mulla, G. N. D. D. Narayan, and G. Hanchinmani, "Intrusion Detection System using MLP and Chaotic Neural Networks," in *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*, Jan. 2021, vol. 2021-January. doi: 10.1109/ICCCI50826.2021.9457024.

[13] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," Feb. 2018. doi: 10.14722/ndss.2018.23204.

[14] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune Network Attack Dataset Data Set," Oct. 16, 2019. https://archive.ics.uci.edu/ml/datasets/Kitsune+Network+Attack+Dataset

[15] D. S. Prashanth, R. V. K. Mehta, and N. Sharma, "Classification of Handwritten Devanagari Number - An analysis of Pattern Recognition Tool using Neural Network and CNN," *Procedia Computer Science*, vol. 167, no. 2019, pp. 2445–2457, 2020, doi: 10.1016/j.procs.2020.03.297.

[16] T. Acharya, I. Khatri, A. Annamalai, and M. F. Chouikha, "Efficacy of Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection," *2021 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2021 - Proceedings*, no. June, pp. 408–413, 2021, doi: 10.1109/I2CACIS52118.2021.9495864.

[17] M. A. Umar and C. Zhanfang, "Effects of Feature Selection and Normalization on Network Intrusion Detection," pp. 1–25, 2020, doi: 10.36227/techrxiv.12480425.

[18] X. Wan, "Influence of feature scaling on convergence of gradient iterative algorithm," in *Journal of Physics: Conference Series*, Jun. 2019, vol. 1213, no. 3. doi: 10.1088/1742-6596/1213/3/032021.

[19] V. N. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, "Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification," in *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020*, Aug. 2020, pp. 729–735. doi: 10.1109/ICSSIT48917.2020.9214160.

DOI: https://doi.org/10.29207/resti.v6i3.4035

[20] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.

[21] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4. Springer Verlag, pp. 611–629, Aug. 01, 2018. doi: 10.1007/s13244-018-0639-9.

[22] M. H. N. Solhdar, M. J. Solahdar, and S. Eskandari, "An intrusion detection system with a parallel multi-layer neural network," *Journal of Mathematical Modeling*, vol. 9, no. 3, pp. 437–450, 2021, doi: 10.22124/jmm.2021.17362.1502.

[23] S. Abdullah, M. Ismail, and Ahmed, "Multi-Layer Perceptron Model for Air Quality Prediction," 2019. [Online]. Available: https://einspem.upm.edu.my/journal/fullpaper/vol13sdecember/8.pdf

[24] T. D. Pham, "Time–frequency time–space LSTM for robust classification of physiological signals," *Scientific Reports*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-86432-7.

[25] J. Alikhanov, W. Kim, M. Azizjon, and A. Jumabek, "1D CNN based network intrusion detection with normalization on imbalanced data," 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9064976

[26] N. Sabri, "A Comparison between Average and Max-Pooling in Convolutional Neural Network for Scoliosis Classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 1.4, pp. 689–696, Sep. 2020, doi: 10.30534/ijatcse/2020/9791.42020.

[27] J. Yang and G. Yang, "Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer," *Algorithms*, vol. 11, no. 3, pp. 1–15, Mar. 2018, doi: 10.3390/a11030028.

[28] A. M. Javid, S. Das, M. Skoglund, and S. Chatterjee, "A ReLU Dense Layer to Improve the Performance of Neural Networks," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.13572

[29] E. Jeczmionek and P. A. Kowalski, "Flattening layer pruning in convolutional neural networks," *Symmetry (Basel)*, vol. 13, no. 7, Jul. 2021, doi: 10.3390/sym13071147.

[30] H. Pratiwi *et al*., "Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks," in *Journal of Physics: Conference Series*, Mar. 2020, vol. 1471, no. 1. doi: 10.1088/1742-6596/1471/1/012010.

[31] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "Understanding and improving the quality and reproducibility of Jupyter notebooks," *Empirical Software Engineering*, vol. 26, no. 4, 2021, doi: 10.1007/s10664-021-09961-9.

[32] S. P. Pillai, T. Radha Ramanan, and S. D. Madhu Kumar, "Evaluating deep learning paradigms with TensorFlow and Keras for software effort estimation," *International Journal of Scientific and Technology Research*, vol. 9, no. 4, pp. 2753–2761, 2020, [Online]. Available: http://www.ijstr.org/final-print/apr2020/Evaluating-Deep-Learning-Paradigms-With-Tensorflow-And-Keras-For-Software-Effort-Estimation.pdf

[33] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems," 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8927876

[34] Q. R. S. Fitni and K. Ramli, "Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems," in *The 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 2020, pp. 118–124. [Online]. Available: https://ieeexplore.ieee.org/document/9172014

DOI: https://doi.org/10.29207/resti.v6i3.4035