ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

DETECTION OF UDP FLOODING DDOS ATTACKS ON IOT NETWORKS USING RECURRENT NEURAL NETWORK

Warcita[1], Kurniabudi[2], Eko Arip Winanto[3]

[1,2,3]Faculty of Computer Science, Universitas Dinamika Bangsa, Jambi, Indonesia

email: waaarcitaaa@gmail.com[1], kurniabudi@unama.ac.id[2], ekoaripwinanto@unama.ac.id [3]

Abstract

Internet of Thing (IoT) is a concept where an object can transfer data through a network without requiring human interaction. Complex IoT networks make it vulnerable to cyber attacks such as DDoS UDP Flood attacks, UDP Flood attacks can disrupt IoT devices. Therefore, this study proposes an attack detection method using a deep learning approach with the Recurrent Neural Network (RNN) method. This study uses Principle Component Analysis (PCA) to reduce the feature dimension, before learning using RNN. The purpose of this study is to test the combined performance of the PCA and RNN methods to detect DDoS UDP Flood attacks on IoT networks. The testing in this study used 10 datasets sourced from CICIOT2023 containing UDP Flood and Benign DDoS traffic data, and the testing was carried out using three epoch parameters (iterations), namely 10, 50, and 100. The test results using RNN epoch 100 were superior, showing satisfactory performance with an accuracy value of 98%, precision of 99%, recall of 99%, and f1-score of 99%. Based on the experimental results, it can be concluded that combining PCA and RNN is able to detect UDP Flooding attacks by showing high accuracy.

Keywords : DDoS, UDP Flood, IDS, Deep Learning, RNN

Received: 03-06-2024 | **Revised:** 28-09-2024 | **Accepted:** 02-11-2024
DOI: https://doi.org/10.23887/janapati.v13i3.79601

INTRODUCTION

Internet of things or commonly known as IoT is a concept where an object can transfer data over a network without the need for human interaction. With the development of IoT technology in the near future it will become something commonly used in the future[1]. The development of IoT is often interpreted by the term smart integrated with existing and conventional infrastructure, such as smart city, smart health, smart transportation, and smart home[2].

The presence of IoT makes it easier and speeds up the process of interaction between humans and objects. IoT can be applied in various fields including health, industry, automotive and others[3]. The application of IoT in everyday life, for example, a remote control that can notify the owner via short message about the use of the AC in their house which has not been turned off when the owner is away from home, another example is if there is a gas leak at home, the home owner will receive a warning via direct message automatic[4].

However, with all the convenience provided by IoT, it often experiences attacks due to the large number of devices connected to the internet network and connected to each other, making the network vulnerable to cyberattacks[5] such as phishing, malware[6], and Distributed Denial-of-Service (DDoS) attacks[7]. These attacks can damage IoT devices, access personal data stored on the device, and can even exploit security weaknesses in the device so they can access other networks[8]. DDoS is an attack that is capable of paralyzing a server by flooding network traffic which causes the network to go down[9]. Therefore, a DDoS attack detection system is needed on the Internet of Things network.

To detect DDoS attacks that occur, one common approach is to use an Intrusion Detection System (IDS) which functions to observe suspicious activity on the network[10]. Various IDS methods have been developed, starting from the use of machine learning algorithms, to Deep Learning, which is a machine learning method that combines artificial neural networks by imitating the way the human brain works and the algorithms used are inspired by the structure of the human brain[11],[12]. Deep Learning has also been widely used to detect DDoS attacks. In research [13] discussing the development of an attack detection system on cloud computing where cloud computing is also vulnerable to DDoS attacks, the research uses

the KDDCup 99 dataset and uses the Deep Learning method, namely RNN with accuracy results of 94.12%. RNN has the best accuracy compared to other methods. Furthermore, In the study [14] using deep convolutional neural networks (CNNs) to detect DDoS attacks. The test results show that the proposed model is able to detect attacks with an accuracy above 91%.

One type of DDoS attack is UDP Flood. UDP Flood attacks can cause significant disruption to IoT devices, which often have limited resources and unstable connectivity[15]. In addition, Specifically, when compromised devices were bombarded with TCP floods, 39.22% more reads and 34.68% more writes were performed compared to normal devices[16]. Therefore, it is important to identify UDP Flooding attacks. To identify UDP Flood attacks, researchers applied various methods and environments. Machine learning is one of the methods commonly used to detect UDP Flooding attacks. However, the Machine Learning method still has shortcomings, besides low throughput, it also has a high false positive rate[17].

The research aims to apply Deep Learning with the RNN method to detect DDoS attacks with the UDP Flood type. RNN was chosen because it has high accuracy and strong modeling for detection and the performance of this method is superior to CNN[18].

In this research, RNN was tested to detect UDP Flood attacks on the CICIoT2023 dataset, which is a dataset that has large dimensions. Apart from that, in this research Principal Component Analysis (PCA) was used to extract features. The purpose of feature extraction is to reduce the dimensions of the data so that it can reduce computing time on the RNN. This research is expected to provide insight into the performance of PCA combined with RNN to detect UDP Flood attacks in terms of Accuracy, Precision, Recall and F1-Score parameters. This insight can be used as a reference for other researchers, especially in the field of IDS on IoT infrastructure.

METHOD

In this research, several stages were carried out. The first stage is collecting datasets, the second stage is preprocessing data, the third stage is cleaning data, the fourth stage is one-hot-encoding, the fifth stage is feature extraction, the sixth stage is data division, the seventh stage is building a model, the eighth stage is training the model, the ninth stage is testing, and the tenth stage is detection results. For a more detailed explanation, you can see the experimental flow presented in Figure 1.

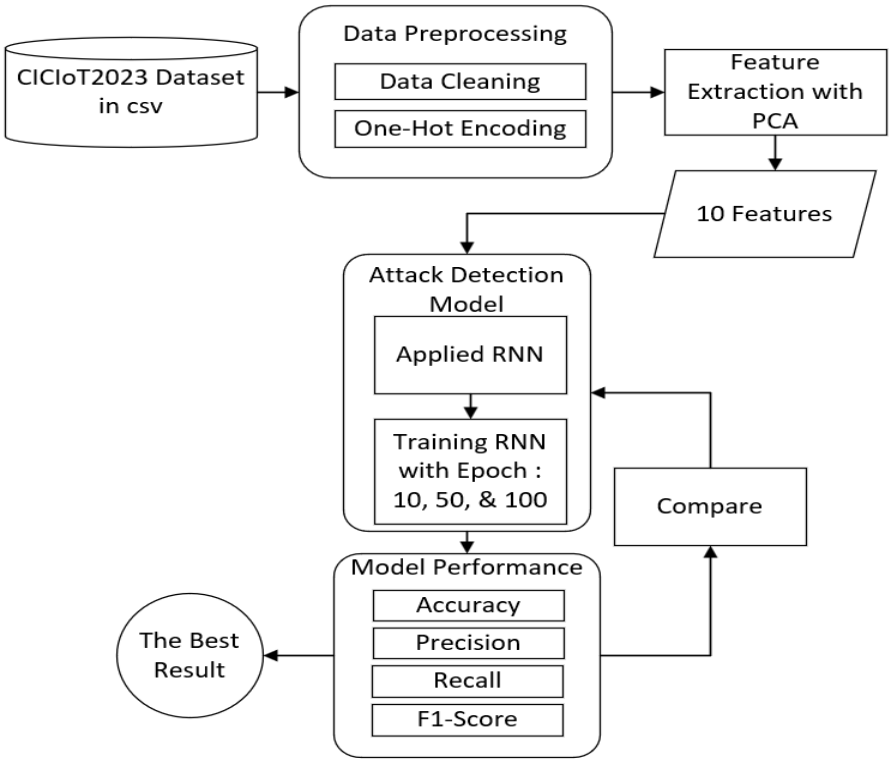

Figure 1. Flow of Experiments

Based on Figure 1, it can be explained that the research experiment was carried out through the following stages:

a) Data preprocessing. At this stage, data cleaning, such as deleting missing values, is carried out. Then, one-hot encoding is carried out to convert the data into binary. The dataset used in the experiment is the CICIoT2023 dataset, where only benign traffic and UDP Flood attacks are used. The data is in .csv format.
b) Feature extraction. At this stage, feature extraction is carried out with PCA. The aim is to reduce features so that it can improve the detection system's performance.
c) Attack detection model. At this stage, RNN is used to detect UDP Flood attacks. Epochs 10, 50, and 100 are applied to the training model.
d) Performance model. At this stage, the performance of the RNN model is measured using the Accuracy, precision, recall, and F1-Score values. Furthermore, the model performance for each epoch is compared.
e) The best result. The best results will be obtained from the results of the experiment.

CICIoT2023 Dataset

The dataset used in this research is CICIoT2023 which is sourced from the Canadian Cyber Security Institute (CIC), in the CICIoT2023 dataset there are seven types of attacks, namely DDoS, Brute Force, Spoofing, Dos, Recon, Web-Based, and Mirai[19]. The author uses a UDP Flood type DDoS attack dataset which has 168 files and 46 features in it. In this study, only 10 data files from the CICIoT2023 dataset were used. Table 1 presents a list of features in the CICIoT2023 Dataset. In table 2 the profile of the dataset used in this research is presented. The detailed data used for the experiment is presented in Table 3.

Table 1. Features Name, Data Type, and Example Data on Dataset CICIoT2023

No	Fitur *Dataset*	*Data Type*	*Data Example*
1	*Flow_Duration*	*Numerical*	00.00
2	*Header_Lenght*	*Numerical*	54.00.00
3	*Protocol Type*	*Numerical*	06.00
4	*Duration*	*Numerical*	64.00.00
5	*Rate*	*Numerical*	0.3298071530725829
6	*Srate*	*Numerical*	0.3298071530725829
7	*Drate*	*Numerical*	00.00
8	*Fin_Flag_Number*	*Numerical*	01.00
9	*Syn_Flag_Number*	*Numerical*	00.00
10	*Rst_Flag_Number*	*Numerical*	01.00
11	*Psh_Flag_Number*	*Numerical*	00.00
12	*Ack_Flag_Number*	*Numerical*	00.00
13	*Ece_Flag_Number*	*Numerical*	00.00
14	*Cwr_Flag_Number*	*Numerical*	00.00
15	*Ack_Count*	*Numerical*	01.00
16	*Syn_Count*	*Numerical*	00.00
17	*Fin_Count*	*Numerical*	01.00
18	*Urg_Count*	*Numerical*	00.00
19	*Rst_Count*	*Numerical*	00.00
20	*HTTP*	*Numerical*	00.00
21	*HTTPS*	*Numerical*	00.00
22	*DNS*	*Numerical*	00.00
23	*Telnet*	*Numerical*	00.00
24	*SMTP*	*Numerical*	00.00
25	*SSH*	*Numerical*	00.00
26	*IRC*	*Numerical*	00.00
27	*TCP*	*Numerical*	00.00
28	*UDP*	*Numerical*	00.00
29	*DHCP*	*Numerical*	00.00
30	*ARP*	*Numerical*	00.00
31	*ICMP*	*Numerical*	00.00
32	*IPv*	*Numerical*	01.00
33	*LLC*	*Numerical*	01.00

No	Fitur *Dataset*	*Data Type*	*Data Example*
34	*Tot_Sum*	*Numerical*	567.00.00
35	*Min*	*Numerical*	54.00.00
36	*Max*	*Numerical*	54.00.00
37	*AVG*	*Numerical*	54.00.00
38	*Std*	*Numerical*	00.00
39	*Total Size*	*Numerical*	54.00
40	*IAT*	*Numerical*	8.334.383.192.013.870
41	*Number*	*Numerical*	09.05
42	*Magnitue*	*Numerical*	10.392.304.845.413.200
43	*Radius*	*Numerical*	00.00
44	*Covariance*	*Numerical*	00.00
45	*Variance*	*Numerical*	00.00
46	*Weight*	*Numerical*	141.55.00

Table 2. Dataset Profile

No	Label	Numbers of packages
1	DDoS UDP Flood	320.069
2	Benign Traffic	65.715
	Total Package	385.784

Table 3. Data Spesification for Experiment

File#	*Dataset*	Type of Traffic	
		UDP Flood	***Benign***
1	*UDP_01.csv*	27.626	5.600
2	*UDP_02.csv*	26.301	5.464
3	*UDP_03.csv*	28.639	5.874
4	*UDP_04.csv*	26.979	5.537
5	*UDP_05.csv*	30.995	6.427
6	*UDP_06.csv*	50.129	10.060
7	*UDP_07.csv*	24.932	5.046
8	*UDP_08.csv*	51.820	10.731
9	*UDP_09.csv*	26.735	5.513
10	*UDP_10.csv*	25.913	5.463
	Sub. Total	**320.069**	**65.715**
	Total	385.784	

Data Preprocessing

In the preprocessing stage, data selection will be carried out, where the CICIoT2023 dataset has several attack classes and 1 benign class. In this study, only 2 classes of data will be used, namely DDoS-UDP Flood and BenignTraffic (normal attacks). In addition, the preprocessing stage also removes duplicate data and data that has no value. To delete duplicate data, pseudocode 1 is used. To check data that has missing values, use pseudocode 2.

Pseudocode 1 : Displaying and Deleting Duplicate Data

```
data.duplicated().sum()
df = pd.DataFrame(data)
df_no_duplicates_subset =
df.drop_duplicates(subset=['flow_duration',
'Header_Length', 'Protocol Type',
'Duration', 'Rate', 'Srate', 'Drate',
'fin_flag_number','syn_flag_number',
'rst_flag_number', 'psh_flag_number',
'ack_flag_number', 'ece_flag_number',
'cwr_flag_number', 'ack_count',
'syn_count', 'fin_count',
'urg_count','rst_count', 'HTTP', 'HTTPS',
'DNS', 'Telnet', 'SMTP', 'SSH', 'IRC', 'TCP',
'UDP', 'DHCP', 'ARP', 'ICMP',
'IPv', 'LLC', 'Tot sum', 'Min', 'Max', 'AVG',
'Std', 'Tot size','IAT', 'Number', 'Magnitue',
'Radius', 'Covariance', 'Variance', 'Weight',
'label'])
print("\nDataFrame Setelah Menghapus
Duplikasi (Subset):")
print(df_no_duplicates_subset)
```

***Pseudocode* 2.** check data that has missing values

```
print(data.isnull().sum())
data =
data.dropna(subset=['flow_duration',
'Header_Length', 'Protocol Type',
'Duration',
'Rate', 'Srate', 'Drate',
'fin_flag_number', 'syn_flag_number',
```

```
'cwr_flag_number', 'ack_count',
'syn_count', 'fin_count', 'urg_count',
'rst_count',
'HTTP', 'HTTPS', 'DNS', 'Telnet',
'SMTP', 'SSH', 'IRC', 'TCP', 'UDP',
'DHCP', 'ARP', 'ICMP',
'Radius', 'Covariance', 'Variance',
'Weight', 'label'])
display(data)
```

One-Hot Encoding

One-Hot Encoding is a binary representation where each vector is converted into binary where the class value (label) is represented with the value 1 and other values can be represented with the value 0.

Feature Extraction Using PCA

The feature extraction technique used in this research is Principal Component Analysis (PCA), which functions to reduce the value dimensions of the data and to improve the performance of the model used[20]. PCA is a technique used in reducing the size of data. Principal Component Analysis (PCA) calculates the covariance matrix of the data, then looks for the eigenvector and eigenvalue. The Principal Component Analysis (PCA) method is suitable for use on data that has a large number of attributes and is correlated with each other[21]. Feature extraction using PCA is applied with pseudocode 3.

Pseudocode 3: *Feature Extraction*

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_scaled)
explained_variance_ratio =
pca.explained_variance_ratio_
print("Rasio Varians yang Dijelaskan oleh
Komponen Utama
selected_features =
data.columns[np.argmax(np.abs(pca.components_
), axis=1)]
print("Fitur yang Dipilih:", selected_features)
```

RNN Model

After splitting the data, the next stage involves building the Recurrent Neural Network (RNN) model, which will be used for the training process. The model is constructed using the simple RNN library, with a sigmoid activation function applied due to the binary classification task, which involves two classes. The RNN library provides flexibility in adjusting model parameters, such as the number of layers, units, and activation functions, allowing for fine-tuning to achieve optimal performance. This enables the model to effectively learn from the training data and generalize well to the testing data.

An RNN is a type of neural network that is specifically designed to process sequential data, such as time-series data or any data where temporal relationships are important. Unlike traditional feedforward neural networks, RNNs have connections that form cycles, allowing them to maintain a memory of previous inputs in the hidden layers. This characteristic enables RNNs to capture patterns in sequential data over time, making them highly effective for tasks like time-series forecasting, natural language processing, and, in this case, detecting patterns in network traffic to identify DDoS attacks.

One of the key advantages of RNNs is their ability to handle sequential dependencies, meaning they can retain information from previous time steps and use it to inform current predictions in Figure 2.

This is particularly beneficial when working with data where context matters, such as in IoT network traffic, where identifying an attack might require an understanding of past traffic patterns. Additionally, RNNs can work with variable-length input sequences, making them versatile for a wide range of applications. The recurrent nature of the network also allows for more efficient training on temporal data, leading to more accurate predictions in tasks involving time-dependent features. Pseudocode 4 is used to model the RNN.

***Pseudocode** 4 : Recurrent Neural Network Model*

```
model = Sequential()
model.add(SimpleRNN(units = 10,
activation='sigmoid', return_sequences=True,
input_shape= (X_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 10,
activation='sigmoid', return_sequences=True))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 10,
activation='sigmoid', return_sequences=True))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 10))
model.add(Dropout(0.2))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=[10,50,100]
batch_size=512)
predictions = model.predict(X_test)
```

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

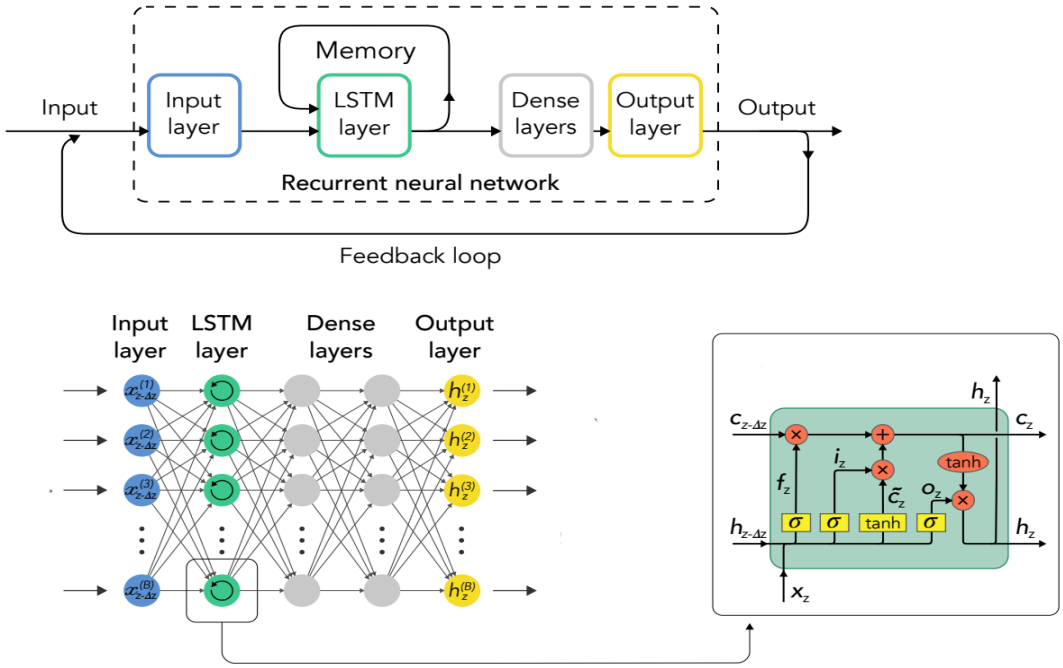

Figure 2 RNN Architecture

RNN Model Training

At this stage, the training process of the recurrent neural network model is carried out using the training data. This phase aims to train the model so that it can recognize more complex attack patterns. In the RNN Model training phase, several epoch parameters are used, namely 10, 50, and 100. In table 4 the training process using the epoch parameter 10 is presented. Based on the training results using RNN epoch 10 mode which have been presented in table 4, it can be seen an increase in the performance of the RNN model. The first epoch, the loss value is 0.6469 and the accuracy is 0.6226. In the RNN model training process, the loss value continues to decrease and the accuracy value continues to increase. It can be seen in epoch 10, the loss value decreases with a value of 0.3514, while the accuracy increases to 0.8342. And it has been presented in table 5, the training process using epoch 50, the loss value continues to decrease and the accuracy value increases, the same as in table 6, the training process for epoch 100 using RNN, the loss value continues to decrease and the accuracy increases. So it can be concluded that the RNN model is successful in carrying out training effectively and is able to perform pattern recognition well.

Detection System Testing

After the model training process has been carried out, the next stage is the testing process using testing data where the model will learn patterns of normal attacks from network traffic and carry out tests. In the testing process three epoch parameters are used, including 10, 50 and 100.

Detection Results

The next step after testing is to measure the detection results using a confusion matrix.Confusion Matrix is a measurement technique used in Deep Learning to view the performance of a model or algorithm to see the accuracy, precision, recall and f1-score of the model used[22]. At this stage, it is to see how well the performance of the model used to detect attacks is. The following are the equations for accuracy, precision, recall, and f1-score[23].

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

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

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

$$F1score = 2 \times \frac{Recall \times Presisi}{Recall + Presisi} \quad (4)$$

Tabel 4. Proses Training Epoch 10

Training Model
history = model.fit(X_train, y_train, epochs=10, batch_size=512, validation_split=0.3)
Epoch 1/10 32/32 [==============================] - 8s 86ms/step - loss: 0.6469 - accuracy: 0.6226 - val_loss: 0.5041 - val_accuracy: 0.8313

Epoch 2/10 32/32 [==========] - 2s 68ms/step - loss: 0.5060 - accuracy: 0.8136 - val_loss: 0.4540 - val_accuracy: 0.8313
Epoch 3/10 32/32 [==========] - 2s 67ms/step - loss: 0.4731 - accuracy: 0.8283 - val_loss: 0.4528 - val_accuracy: 0.8313
Epoch 4/10 32/32 [==========] - 2s 68ms/step - loss: 0.4720 - accuracy: 0.8295 - val_loss: 0.4525 - val_accuracy: 0.8313
Epoch 5/10 32/32 [==========] - 4s 119ms/step - loss: 0.4708 - accuracy: 0.8303 - val_loss: 0.4509 - val_accuracy: 0.8313
…
Epoch 10/10 32/32 [==========] - 3s 94ms/step - loss: 0.3514 - accuracy: 0.8342 - val_loss: 0.2560 - val_accuracy: 0.9359

Tabel 5 Proses Training Epoch 50

Training Model
history = model.fit(X_train, y_train, epochs=50, batch_size=512, validation_split=0.3)
Epoch 1/50 32/32 [==========] - 7s 98ms/step - loss: 0.8879 - accuracy: 0.3681 - val_loss: 0.6088 - val_accuracy: 0.8313
Epoch 2/50 32/32 [==========] - 3s 80ms/step - loss: 0.5946 - accuracy: 0.7005 - val_loss: 0.4847 - val_accuracy: 0.8313
Epoch 3/50 32/32 [==========] - 4s 129ms/step - loss: 0.5159 - accuracy: 0.7932 - val_loss: 0.4578 - val_accuracy: 0.8313
Epoch 4/50 32/32 [==========] - 3s 79ms/step - loss: 0.4910 - accuracy: 0.8156 - val_loss: 0.4526 - val_accuracy: 0.8313
Epoch 5/50 32/32 [==========] - 2s 76ms/step - loss: 0.4825 - accuracy: 0.8245 - val_loss: 0.4522 - val_accuracy: 0.8313
…
Epoch 50/50 32/32 [==========] - 4s 118ms/step - loss: 0.1102 - accuracy: 0.9591 - val_loss: 0.0799 - val_accuracy: 0.9713

Tabel 6. Proses Training Epoch 100

Training Model
history = model.fit(X_train, y_train, epochs=100, batch_size=512, validation_split=0.3)
Epoch 1/100 32/32 [==========] - 7s 85ms/step - loss: 0.4983 - accuracy: 0.8158 - val_loss: 0.4608 - val_accuracy: 0.8313
Epoch 2/100 32/32 [==========] - 2s 67ms/step - loss: 0.4773 - accuracy: 0.8299 - val_loss: 0.4575 - val_accuracy: 0.8313
Epoch 3/100 32/32 [==========] - 2s 67ms/step - loss: 0.4702 - accuracy: 0.8303 - val_loss: 0.4549 - val_accuracy: 0.8313
Epoch 4/100 32/32 [==========] - 3s 87ms/step - loss: 0.4686 - accuracy: 0.8300 - val_loss: 0.4548 - val_accuracy: 0.8313
Epoch 5/100 32/32 [==========] - 3s 96ms/step - loss: 0.4678 - accuracy: 0.8302 - val_loss: 0.4531 - val_accuracy: 0.8313
…
Epoch 100/100 32/32 [==========] - 3s 96ms/step - loss: 0.0944 - accuracy: 0.9689 - val_loss: 0.0621 - val_accuracy: 0.9854

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

RESULT AND DISCUSSION

In this section, the results and discussion of the experiments that have been carried out are presented, where this discussion section contains the results of feature extraction and an evaluation of the performance of the recurrent neural network model in carrying out detection.

Below in table 7 are the feature extraction weights using principal component analysis (PCA). The aim of feature extraction is to reduce the feature dimensions in the dataset. Based on Table 7, it shows the results of feature extraction using the PCA technique. From the table presented, this research reduced 46 features to 10 features. The results of feature extraction are used to process training data using the RNN method, in this process it is successful in extracting features from the range -0 to 1. The next stage is presented with attack detection testing using the Recurrent Neural Network (RNN) method, in this research testing using three epoch parameters include 10, 50, and 100.

Based on Figure 3 which has been presented, it can be seen that the results of the "Sequential" model have 9 layers, for the first layer, namely simpleRNN, where this first layer has 10 units (neurons) with a sigmoid activation function and return sequence indicating that the output from the layer will be the sequence that will be used. returned by the next layer, the second layer is dropout to reduce overfitting.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 46, 10)            120

 dropout (Dropout)           (None, 46, 10)            0

 simple_rnn_1 (SimpleRNN)    (None, 46, 10)            210

 dropout_1 (Dropout)         (None, 46, 10)            0

 simple_rnn_2 (SimpleRNN)    (None, 46, 10)            210

 dropout_2 (Dropout)         (None, 46, 10)            0

 simple_rnn_3 (SimpleRNN)    (None, 10)                210

 dropout_3 (Dropout)         (None, 10)                0

 dense (Dense)               (None, 1)                 11

=================================================================
Total params: 761 (2.97 KB)
Trainable params: 761 (2.97 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3. RNN Model

Tabel 7. Weight PCA Result

Fitur	PCA Result
10	-0.037199, -1.207362, 0.014371, -0.179166, -0,064135, 0.015371, -0.052441, 0.041760, -0,053336, -0.051410
10	-0.014397, -0.777617, 0.499403, -0.335046, 1.393951, -0282846, 1.144556, -0.504602, 0.088795, -0.170807
10	-0.045452, -1.218391, 0.053087, -0.596143, -0.059501, -0.051573, 0.005627, 0.070024, 0.206097, 0.017802
10	-0.067031, -1.145960, 0.033895, -0.460764, 0.55125, -0.007650, 0.018137, -0.003043, 0,045564, 0.274592
10	-0.051226, -1.246096, -0.614934, 0.218699, -0.019531, -0.019282, -0.034572, 0.068057, -0.051146, 0.061969
10	-0.032959, 0.543165, -0.235668, -0.570090, 0.247616, 0.126126, 0.967699, 0.075070, -0.858132, -0.970213

The third layer, namely simpleRNN1, has the same configuration as the first layer, the fourth layer, namely dropout1, is used again to reduce overfitting, then the fifth layer, namely simpleRNN2, has the same configuration as the first layer and is followed by the sixth layer, namely dropout2. Likewise, the seventh and eighth layers have the same configuration as the first layer, and the ninth layer, namely dense, has 1 unit (neuron) in it with a sigmoid activation function which functions to produce output in the range 0 and 1.

The following are the results of attack detection using the RNN method which has been tested using three epoch parameters, the results of attack detection are accuracy, precision, recall and f1 score. Below in table 8 are the detection results using the epoch 10 parameter.

In the detection results using RNN epoch 10, the highest accuracy reached 96%, the highest precision reached 99%, the highest recall reached 100%, and the highest f1 score reached 98%. Next, in table 9, the detection results using RNN epoch 50 are presented.

In the detection results using RNN epoch 50, the highest accuracy reached 98%, the highest precision reached 100%, the highest recall reached 99%, and the highest f1 score reached 99%. Next, in table 10, the detection results using RNN epoch 100 are presented.

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

Table 8. Epoch 10 RNN Detection Result

Dataset	Accuracy	Precission	Recall	F1- score
UDP_01	93%	99%	93%	96%
UDP_02	95%	96%	98%	97%
UDP_03	92%	99%	91%	95%
UDP_04	92%	99%	92%	95%
UDP_05	93%	94%	98%	96%
UDP_06	96%	98%	98%	98%
UDP_07	96%	95%	100%	97%
UDP_08	96%	97%	98%	98%
UDP_09	93%	94%	98%	96%
UDP_10	96%	97%	98%	98%

Table 9. Epoch 50 RNN Detection Result

Dataset	Accuracy	Precission	Recall	F1-score
UDP_01	97%	99%	97%	98%
UDP_02	97%	99%	98%	98%
UDP_03	98%	98%	99%	99%
UDP_04	97%	99%	97%	98%
UDP_05	98%	99%	99%	99%
UDP_06	97%	99%	98%	98%
UDP_07	97%	99%	97%	98%
UDP_08	98%	100%	97%	99%
UDP_09	97%	99%	97%	98%
UDP_10	97%	99%	98%	98%

Table 10. Epoch 100 RNN Detection Result

Dataset	Accuracy	Precission	Recall	F1-score
UDP_01	98%	99%	99%	99%
UDP_02	99%	99%	100%	99%
UDP_03	97%	99%	98%	98%
UDP_04	96%	96%	100%	98%
UDP_05	99%	99%	100%	99%
UDP_06	99%	99%	100%	99%
UDP_07	98%	98%	100%	99%
UDP_08	99%	100%	99%	99%
UDP_09	98%	98%	99%	99%
UDP_10	99%	99%	100%	99%

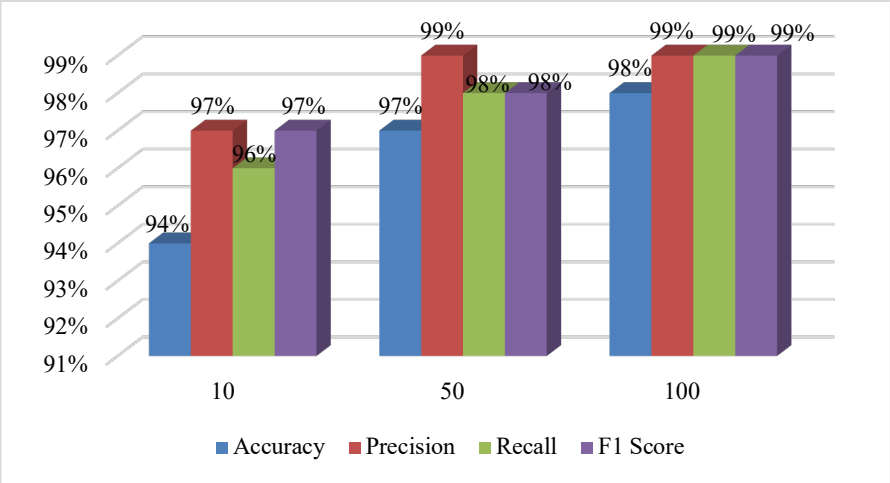

Figure 4. Performance Comparison for Each Epoch

In the detection results using RNN epoch 100, the highest accuracy reached 99%, the highest precision reached 100%, the highest recall reached 100%, and the highest f1 score reached 99%.

Based on Figure 4 that has been presented, it can be seen that the detection results using the epoch 100 parameter have the best performance with an accuracy value reaching 98%, a precision value reaching 99%, a recall value reaching 99%, and an f1 score reaching 99%. Based on the test results, it can be concluded that the Recurrent Neural Network (RNN) method is able to detect UDP Flood DDoS attacks with high accuracy indicating that detecting attacks using deep learning Recurrent Neural Network (RNN) methods can recognize complex attack patterns on IoT network traffic, as can be seen in research[13] detecting DDoS Attacks using the Deep Neural Network method result an accuracy of 94.12%. so it can be compared that RNN is superior in detecting attacks that focus on UDP Flood attacks on IoT networks. However, this study has limitations including only testing UDP Flood attacks, in addition to the use of limited datasets so that if implemented on more complex network traffic, the results may be different.

CONCLUSION

This study proposes a method to detect DDoS attacks of the UDP Flood type on IoT networks. To represent real IoT network traffic, this study uses the CICIoT2023 dataset. This dataset has a large data dimension. To reduce the feature dimension, Principal Component Analysis is applied. This technique successfully reduces 40 features to 10 features. The reduced data is then used as input data for the RNN model. In this study, experiments were conducted using 10 datasets and three epoch parameters, namely 10, 50 and 100. Model performance was measured by looking at the accuracy, precision, recall and f1-score values. The test results show the superiority of the RNN method in recognizing attacks on complex IoT networks. By applying three epoch parameters (iterations), namely 10, 50 and 100, at epoch 100 the accuracy value reaches 98%, precision 99%, recall 99% and f1-score 99%. These results show that by performing feature reduction using PCA, and applying the RNN model, very good UDP Flood attack detection performance is obtained. Although the test results show very good performance, further research is still needed to test the reliability of the model. The model still needs to be tested with a more complex number of attacks (traffic types) and a larger amount of data.

REFERENCES

[1] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, “Internet of things security: A top-down survey,” *Comput. Networks*, vol. 141, pp. 199–221, 2018, doi: 10.1016/j.comnet.2018.03.012.

[2] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. A. Maisto, and S. Nacchia, “Internet of things reference architectures, security and interoperability: A survey,” *Internet of Things*, vol. 1–2, pp. 99–112, 2018, doi: 10.1016/j.iot.2018.08.008.

[3] I. Khajenasiri, A. Estebsari, M. Verhelst, and G. Gielen, “A review on Internet of Things solutions for intelligent energy control in buildings for smart city applications,” *Energy Procedia*, vol. 111, no. September 2016, pp. 770–779, 2017, doi: 10.1016/j.egypro.2017.03.239.

[4] R. Kaur, P. Vats, M. Mandot, S. S. Biswas, and R. Garg, “Literature Survey for IoT-based Smart Home Automation: A Comparative Analysis,” in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2021, pp. 1–6. doi: 10.1109/ICRITO51393.2021.9596421.

[5] K. H. Le, M. H. Nguyen, T. D. Tran, and N. D. Tran, “IMIDS: An Intelligent Intrusion Detection System against Cyber Threats in IoT,” *Electron.*, vol. 11, no. 4, pp. 1–16, 2022, doi: 10.3390/electronics11040524.

[6] S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail, and M. M. Hassan, “Malware Threats and Detection for Industrial Mobile-IoT Networks,” *IEEE Access*, vol. 6, no. c, pp. 15941–15957, 2018, doi: 10.1109/ACCESS.2018.2815660.

[7] H. Mustapha and A. M. Alghamdi, “DDoS attacks on the internet of things and their prevention methods,” *Proc. 2nd Int. Conf. Futur. Networks Distrib. Syst. - ICFNDS ’18*, pp. 1–5, 2018, doi: 10.1145/3231053.3231057.

[8] V. Subbarayalu, B. Surendiran, and P. Arun Raj Kumar, “Hybrid Network Intrusion Detection System for Smart Environments Based on Internet of Things,” *Comput. J.*, vol. 62, no. 12, pp. 1822–1839, Dec. 2019, doi: 10.1093/comjnl/bxz082.

[9] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, “Network Intrusion Detection for IoT Security based on Learning Techniques,” *IEEE Commun.*

Surv. Tutorials, vol. PP, no. 0, pp. 1–1, 2019, doi: 10.1109/comst.2019.2896380.

[10] M. Roopak, G. Y. Tian, and J. Chambers, "An Intrusion Detection System Against DDoS Attacks in IoT Networks," *2020 10th Annu. Comput. Commun. Work. Conf. CCWC 2020*, pp. 562–567, 2020, doi: 10.1109/CCWC47524.2020.9031206.

[11] S. Gurung, M. Kanti Ghose, and A. Subedi, "Deep Learning Approach on Network Intrusion Detection System using NSL-KDD Dataset," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 3, pp. 8–14, Mar. 2019, doi: 10.5815/ijcnis.2019.03.02.

[12] B. I. Farhan and A. D. Jasim, "Survey of Intrusion Detection Using Deep Learning in the Internet of Things," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 1, pp. 83–93, 2022, doi: 10.52866/ijcsm.2022.01.01.009.

[13] R. SaiSindhuTheja and G. K. Shyam, "An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment," *Appl. Soft Comput.*, vol. 100, Mar. 2021, doi: 10.1016/j.asoc.2020.106997.

[14] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep Learning-Based DDoS-Attack Detection for Cyber–Physical System Over 5G Network," *IEEE Trans. Ind. Informatics*, vol. 17, no. 2, pp. 860–870, 2021, doi: 10.1109/TII.2020.2974520.

[15] L. Feinstein, "Preventing DDoS attack using Data mining Algorithms," *IEEE Cloud Comput.*, vol. 6, no. 10, p. 390, 2016, [Online]. Available: www.ijsrp.org

[16] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The Impact of DoS Attacks on Resource-constrained IoT Devices : A Study on the Mirai Attack," *arXiv Prepr. arXiv2104.09041*, 2021.

[17] M. A. Al-shareeda, S. Manickam, and M. A. Saare, "DDoS attacks detection using machine learning and deep learning techniques : analysis and comparison," vol. 12, no. 2, pp. 930–939, 2023, doi: 10.11591/eei.v12i2.4466.

[18] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine Learning and Deep Learning Approaches for CyberSecurity: A Review," *IEEE Access*, vol. 10, pp. 19572–19585, 2022, doi: 10.1109/ACCESS.2022.3151248.

[19] E. Carlos *et al.*, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," 2023, doi: 10.20944/preprints202305.0443.v1.

[20] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Comput. Networks*, vol. 148, pp. 164–175, Jan. 2019, doi: 10.1016/j.comnet.2018.11.010.

[21] S. A. Kadom, S. H. Hashem, and S. H. Jafer, "Optimize network intrusion detection system based on PCA feature extraction and three naïve bayes classifiers," *J. Phys. Conf. Ser.*, vol. 2322, no. 1, 2022, doi: 10.1088/1742-6596/2322/1/012092.

[22] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[23] 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," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.