Ilkogretim Online - Elementary Education Online, 2020; 19 (4): pp. 947-956
http://ilkogretim-online.org
doi:10.17051/ilkonline.2020.04.203

# A study on comparison analysis of the dnn, cnn, and rnn models for network anomaly detection

**Jiahang Ren,** *Department of Math, Computer Science, and Physics, Rockford University, 5050 E. State, Rockford, IL, 61108, U.S.A., jr170718@rockford.edu*
**Jiayuan Cui,** *Department of Math, Computer Science, and Physics, Rockford University, 5050 E. State, Rockford, IL, 61108, U.S.A., jc171508@rockford.edu*
**Mishaal Shah,** *Department of Math, Computer Science, and Physics, Rockford University, 5050 E. State, Rockford, IL, 61108, U.S.A., ms167347@Rockford.edu*
**Jeong-Tak Ryu,** *School of Electronic and Communication Engineering, Daegu University, 201 Daegudae-ro, Gyeongsan-si, Gyeongsangbuk-do, 38453, Republic of Korea, jryu@daegu.ac.kr*
**Donghwoon Kwon,** *Department of Math, Computer Science, and Physics, Rockford University, 5050 E. State, Rockford, IL, 61108, U.S.A., dkwon@rockford.edu*

**Abstract.** With the widespread use of the Internet, network technology is used in a large amount in daily life, and the Internet and network are currently suffering from severe threats of network attacks. Network anomaly detection is one of the most significant issues in network security, and it is a core method to prevent cyber-attacks because it monitors network traffic data to figure out whether they are normal or abnormal. A variety of research frameworks have been proposed for network anomaly detection, and nowadays, deep learning-based methodologies are in the spotlight. For this reason, this research employed three deep learning models, i.e., Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) models, with the public dataset, which is CICIDS 2017 dataset to examine their effectiveness for network anomaly detection. After evaluating the three deep learning models with the CICIDS 2017 dataset, the experimental results show that all three deep learning models show satisfactory results and have high detection accuracy, precision, recall, and F1 Score. This means that they could facilitate a more in-depth analysis of network data and identify anomalies faster. Besides, we observed that the DNN model outperformed the other two deep learning models, which achieved 98.14% of the overall detection accuracy. It proves that deep leaning models seem to be a robust potential tool for network anomaly detection in the cybersecurity field.

**Keywords:** Deep neural network (DNN), convolutional neural network (CNN), recurrent neural network (RNN), intrusion detection evaluation dataset (cicids 2017), deep learning.

Received: 05.10.2020 Accepted: 16.11.2020 Published: 10.12.2020

## INTRODUCTION

In recent years, our network technology has been developed and evolved rapidly. With the widespread use of the Internet, network technology is used in a large amount in daily life; for example, the Internet is used as the medium to communicate and transmit data between the physical devices or between the virtual domains [1]. Although the application of network-based technology is widely used in life, and the threat of network attacks has also been significantly increased. 122 million attacks were detected in 2016, and the number of cyber attacks is increasing every year [2]. By 2021, the cost of cybercrime is expected to reach $6 trillion annually, and cybersecurity spending may exceed $ 1 trillion from 2017 to 2021 [3]. Thus, a variety of models based on deep learning need to be used for network attack detection technologies to achieve reliable detection of network anomalies because of increasing severity, diversification, and complexity of the network attacks. In [4], the authors evaluated how three Convolution Neural Network (CNN) models with different internal depth such as shallow, moderate, and deep take effect on the performance in network anomaly detection. In this research, the evaluation models with three different preprocessed traffic datasets showed that deeper structures did not improve any performance compared to Fully-Connected-Network (FCN) and Long-Short-Term Memory (LSTM).

This point led us to the following research question: which deep learning model does show the best performance with a pure binary classified dataset if the CNN model does not represent performance improvement over other deep learning models?

In the field of deep learning model development related to network anomaly detection, a variety of deep learning models like Deep Neural Network (DNN), CNN, Recurrent Neural Network (RNN) with

LSTM and Gated Recurrent Units (GRU), etc. have been introduced [5-8]. Among them, we decided to adopt the DNN, CNN, and RNN models to answer our research question mentioned above using the pure binary classified dataset in this research because we think these are the three common models in the field of deep learning. The basic concepts of the three deep learning model we adopted are as follows: i) the DNN model is good at handling large-sized data problems. The purpose is to provide basic guidance in the field of deep learning, which has completely changed the natural language processing (NLP) field, ii) the CNN model has an advantage for processing grid-like data. It extracts position-invariant features, and iii) the RNN model is a neural network about spatial depth, and it is good at processing sequence-related data. Thus, it can model units well [9]. The pure binary classified dataset that we decided to use is the Intrusion Detection Evaluation (CICIDS 2017) dataset [10]. The CICIDS2017 data set contains two main labels, such as benign and Distributed Denial-of-Service (DDoS). Note that benign is a normal traffic label, and DDoS is a network attack label. Not only two labels, but it also includes other attributes like timestamps, source and destination IP, source and destination ports, etc. [10]. In the analysis of evaluating the three deep learning models with the CICIDS 2017 dataset, our experimental results show that the DNN model outperformed the other two deep learning models.

This paper is organized as follows: Section 2 briefly discusses a description of the related studies with respect to deep learning models for network anomaly detection. Section 3 presents each deep learning methodology we examined, and evaluation results are discussed in Section 4. Section 5 summarizes our research and described the conclusions and future work.

## MATERIAL AND METHOD

The work in [11] proposed a taxonomy of the Intrusion Detection System focusing on shallow machine-learning and deep learning. Artificial Neural Network (ANN), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), etc. belong to the shallow machine-learning category, and DNN, CNN, RNN, Generative Adversarial Networks (GANs), etc. belong to deep learning. Furthermore, the authors pointed out the pros and cons between shallow models and deep learning models shown in Table 1 below.

**Table 1.** *Pros and Cons Between Shallow Models and Deep Learning Models*

| Category | Shallow Models | Deep Learning Models |
|---|---|---|
| Running Time | Short training and testing time | Long training and testing time due to the high complexity |
| Number of Parameters | Less learnable parameters and hyperparameters | More learnable parameters and hyperparameters |
| Feature Representation | A feature vector is required as an input | A feature vector is not required and is able to learn feature representations from raw data |
| Learning Capacity | Do not have stronger fitting ability, but low risk of overfitting | Have stronger fitting ability, but high risk of overfitting |
| Interpretability | Weak interpretability | Strong interpretability |

Yet, the interesting point the authors mentioned is that deep learning models could have an ability to outperform shallow machine-learning models in most applications.

As the example of deep learning-based network anomaly detection, both LSTM and CNN-based framework in conjunction with constructing block-based features for efficient feature extraction was proposed [12]. A variety of datasets were employed for extensive experiments, and the dataset that draw our attention was the CICIDS 2017 dataset. The authors reported that their proposed deep learning framework with the block-based features achieved 99.56% of the F1 score.

### DNN Model

The fundamental architecture of the DNN model is to be composed of an input layer, one or multiple hidden layers, and an output layer [13]. There are numerous neurons in each layer, and they are connected with each other. Here, connections between the neurons in each layer refer to propagation. Specifically, the neurons in the input layer propagate values or features to the neurons in one or more hidden layers, and the hidden layer(s) propagate the weighted sums to the output layer. Moreover, the outputs of the neurons are activated by an activation function [13-14].

The brief concepts mentioned above could be mathematically explained in detail as follows; There are $x$ inputs, and each neuron performs learning by assigning a weighted value ($w$) to each of its inputs ($x$) through the equation (1) below [15]:

$$Y(x) = w_i * x_i + \mathrm{b} \quad (1)$$

where $Y(x)$ is the summation from multiplying the weights to the inputs, and $w$ and $b$ are weights and bias, respectively. Moreover, there is an activation function to transform the weighted inputs into the outputs. One of the many activation functions, i.g. sigmoid, tanh, Rectified Linear Unit (ReLU), etc. could be employed, and the primary role of the activation function is to convert the linear inputs into the non-linear outputs [16].

The sigmoid non-linear activation function is mainly used in a feedforward neural network which is the skeleton of deep learning and is given by the following equation (2) [16]:

$$f(z) = \left(\frac{1}{1 + exp^{-z}}\right) \quad (2)$$

where $z$ is $Y(x)$, and the *exp* is an exponential calculation. The expected output from the sigmoid activation function is a probability in the range of 0 and 1, which means 0% through 100%.

Natural language processing and speech recognition in the RNN mostly adopt the tanh activation function because it provides better training performance for multi-layer neural networks than the sigmoid function. Better training performance results from a zero-centered output in the range of -1 and 1, and it is given by the following equation (3) [16]:

$$f(z) = \left(\frac{exp^{z} - exp^{-z}}{exp^{z} + exp^{-z}}\right) \quad (3)$$

where $z$ is $Y(x)$, and the exp is an exponential calculation. Yet, the tanh function has a limitation that causes a vanishing gradient issue like the sigmoid function.

Compared to the two activation functions mentioned above, the ReLU activation function is state-of-the-art and gives a faster computation and better performance. In addition to these advantages, the ReLU function is powerful for eliminating the vanishing gradient issue, unlike both sigmoid and tanh functions. The main idea of this function is to make the input values zero if the values of the inputs are less than zero through the following equation (4) [16]:

$$f(z) = \max(0, z) = \begin{cases} \infty, if\ z \geq 0 \\ 0, if\ z < 0 \end{cases} \quad (4)$$

The following Fig. 1 depicts a difference between these three activation functions graphically.

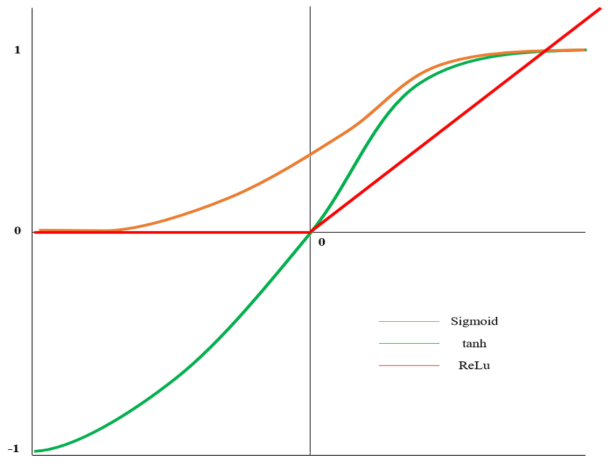

**FIG. 1.** *Activation functions*

The last activation function is based on a conditional probability distribution called the softmax function which produces an output in the range of 0 and 1 [16-17]. Here, the conditional probability means $P_\theta(y|x)$ where $\theta$ is a parameter, $y$ is $y \in R^N$ that is generated by a classification model for a problem with $N$ classes, and $x$ is the inputs [17]. Furthermore, the softmax function is computed by the following equation (5) [16-17]:

$$P(y = j|z^{(i)}) = \left(\frac{exp^{z^{(i)}}}{\sum_{j=0}^{k} exp^{z_k^{(i)}}}\right) \quad (5)$$

where $j$ is a class, and $z$ is $Y(x)$ as mentioned above. The softmax function is used in a multi-class model that returns the probabilities of each class. The target class has the highest probability, and it is mostly seen at almost every output layer in the deep learning architecture [16].

Among multiple activation functions that we introduced above, we decided to employ both ReLU and softmax activation functions in this research, and as shown in Fig. 2 below, our DNN architecture is as follows: i) one input layer with seventy-eight features including two labels, ii) one hidden layer with sixty-four neurons and the ReLU activation function, and iii) one output layer with a softmax layer.

**CNN Model**

One of the main deep learning types is CNN consisting of an input layer, one or more convolutional, pooling, and FCN layers, and an output layer [4], [18].

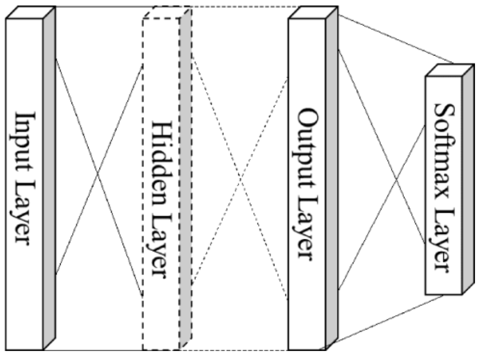

**Fig. 2.** *Architecture of the DNN model*

In this model, each layer has a unique role; for instance, i) the input layer represents features, ii) the convolutional layer is used to learn feature representations of the inputs. Feature representation, also known as a feature map, is generated by a filter with a string and padding. The filter spatially scans the inputs based on a stride that controls the amount of movement over the inputs to create a feature map that summarizes the presence of detected features. However, since the generated feature map through the filter causes the issue of information loss, padding is applied to around the feature map to avoid such a problem. Note that zero-padding is widely used. Moreover, either ReLU or tanh activation function could be adopted to calculate the feature map, iii) two different pooling layers such as average pooling and max pooling reduce the size of the feature representations to control overfitting by down-sampling the inputs. Both pooling layers are computed by the following equations (6) and (7):

- Average pooling

$$f_{avg}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (6)$$

- Max pooling

$$f_{max}(x) = \max(x_i) \quad (7)$$

where *x* denotes the inputs with activation values and *N* indicates a local pooling region. Note that max pooling is more commonly used than average pooling, iv) the FCN layer with an activation function performs high-level reasoning, and v) the output layer is for classification [4], [18-20]. There are two things to keep in mind. The first thing is related to activation functions. While typical activation functions used in the convolutional layer are sigmoid, tanh, and ReLU, the ReLU activation guarantees faster computation compared to the other two activation functions as mentioned earlier [18], [21]. The second thing is that the CNN model could be categorized differently like shallow, moderate, and deep depending on the number of the convolutional layer, pooling layer, and FCN layer as mentioned in Section I.

In this research, we adopted the shallow CNN model for the purpose of consistency, and as shown in Fig. 3 below, its architecture is as follows: i) one convolution layer with 64 filters, and the filter size is 3*1, ii) one max pooling layer with a stride. A stride is set to 1, iii) one FCN layer with 64 neurons, and iv) one softmax layer. Besides, both DNN and CNN models in this research perform training with the binary cross-entropy loss function, Adam optimizer, batch normalization, ReLU activation, and dropout. Note that we set 0.5 to the dropout rate.

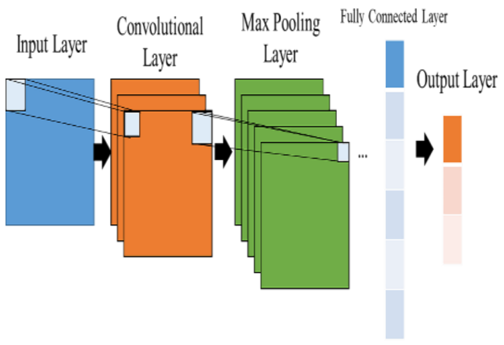

**Fig. 3.** *Architecture of the CNN model*

## RNN Model

RNN is an extension of feed forward neural networks, which is cyclic connections-based for modelling sequences in time series [22]. There are three main layers such as input, hidden, and output layers like the DNN model [23]. The hidden layer refers to a recurrent hidden state, and it is denoted as [24]:

$$h_t = g(Wx_t + Uh_{t-1} + b) \quad (8)$$

where $x_t$ is the inputs at time $t$, $h_t$ is the hidden state, $g$ is the activation function, and $W$, $U$, and $b$ are the sized parameters like input-hidden matrix, hidden-hidden matrix, and bias [24]. The RNN model has definite pros and cons. For example, a variety of different types of data could be used in the RNN model, but despite the RNN model uses a memory cell to store data, they cannot be stored for a long time due to a vanishing gradient issue [23]. For this reason, the idea of using the GRU and the LSTM as the memory cell to overcome such an issue has been proposed.

The GRU RNN model consists of an update gate $z_t$ that determines how much the unit updates its activation and a reset gate $r_t$ that is used to forget the previously computed state [25]. Both gates are computed by the following equations (9) and (10) below [25-26].

$$z_t = g(W_z x_t + U_z h_{t-1}) \quad (9)$$

$$r_t = g(W_r x_t + U_r h_{t-1}) \quad (10)$$

where $g$ is the activation function, $x_t$ is the inputs at time, and $W_z$, $U_z$, $W_r$, and $U_r$ are weight matrices.

The LSTM RNN is a special type of RNN architecture. It is composed of a chain structure, but four interaction layers as the repeating module or cell are used with a unique communication method [27]. Its architecture is illustrated in Fig. 4 below, and the following Table 2 describes the meaning of each notation in Fig. 4.

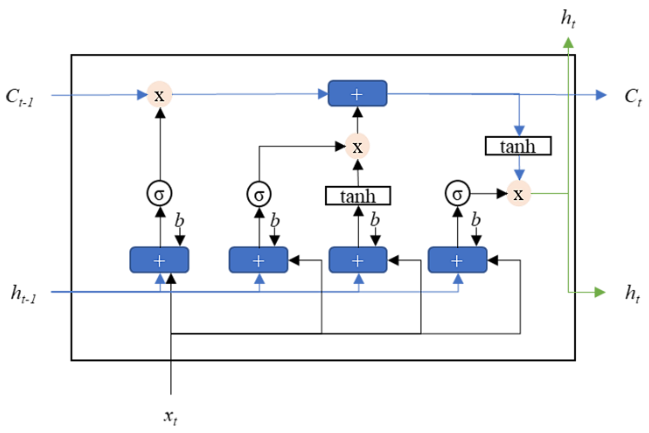

**Fig. 4.** *Architecture of the LSTM RNN model [27]*

**Table 2.** *Meaning of Each Notation [27]*

| Category | Notation | Description |
|---|---|---|
| Inputs | $x_t$ | Current inputs |
| | $h_{t-1}$ | Output of the previous LSTM unit |
| | $c_{t-1}$ | Memory cell state from the previous LSTM unit |
| Outputs | $h_t$ | Current output |

| | | |
|---|---|---|
| | $c_t$ | New updated memory cell |
| Non-linearity | $\sigma$ | Sigmoid activation function |
| | *tanh* | tanh activation function |
| | *b* | bias |
| Vector operations | x | Scaling of information |
| | + | Adding information |

There are three main steps in LSTM architecture [27]. The first step is to identify and exclude data. Specifically, the sigmoid function determines which data need to be identified from the outputs of the previous LSTM unit at time *t-1* and the current inputs at time *t*, as well as, which one from the old outputs needs to be excluded. This step is performed by the forget gate, and it is computed by the following equation (11):

$$f_t = \sigma(W_f\,[h_{t-1}, x_t + b_f) \quad (11)$$

where $W_f$ and $b_f$ are the weight matrices and bias, respectively. The second step with respect to the cell state is to determine which information needs to be updated or ignored, and its importance level through the sigmoid and tanh. The cell state is updated by the following equation (14) resulted from equations (12) and (13):

$$i_t = \sigma(W_i\,[h_{t-1}, x_t + b_i) \quad (12)$$

$$g_t = tanh\,(W_g\,[h_{t-1}, x_t + b_g) \quad (13)$$

$$C_t = C_{t-1} \odot f_t + g_t \odot i_t \quad (14)$$

where $C_t$ is a memory cell, and ⊙ is element-wise multiplication.

The last step is to compute the output cell state through the following equations (15) and (16):

$$O_t = \sigma(W_o\,[h_{t-1}, x_t + b_o) \quad (15)$$

$$h_t = o_t \odot \tanh\,(c_t) \quad (16)$$

After analyzing RNN architectures, we realized that the most challenging part of this research is to decide RNN model architecture because various architectures could be built depending upon a memory cell and an activation function. For instance, simple RNN with the sigmoid function, simple RNN with the softmax function, LSTM with the sigmoid function, etc. are candidate architectures for the RNN model. However, note that the fundamental idea to build a simple architecture is the same as the DNN and CNN models, which means one input layer, one hidden layer, and one output layer. Thus, we subjectively decided to employ the simple RNN model with the sigmoid function at our discretion, and its architecture is as follows: i) one input layer with seventy-eight features including two labels, ii) one hidden layer based on the simple RNN with sixty-four hidden units, and iii) one fully-connected output layer with the softmax function. Furthermore, the RNN model performs training with the binary cross-entropy loss function, Adam optimizer, and dropout like the other two deep learning models. Note that we set 0.5 to the dropout rate, and the architecture of the RNN is depicted in Fig. 5 below.

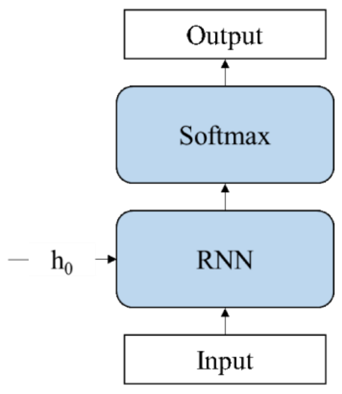

**Fig. 5.** *Architecture of the RNN model*

## RESULT AND DISCUSSION

Before performing experiments, we built each deep learning model with a computer that has the following hardware specifications: Intel i5-7300U 2.6GHz with 8GB RAM. Moreover, each deep learning model was implemented by Python Keras package and was evaluated against the CICIDS 2017 datasets.

As already mentioned earlier, we did not preprocess the dataset like classifying multiple labels into binary labels because the CICIDS 2017 dataset already has two labels only. However, two features out of a total of seventy-nine features, i.e., Flow Bytes/s and Flow Packets/s features, were dropped due to the technical difficulty of applying one-hot encoding. After dropping those two features, the next thing was to split the dataset into training and testing datasets. Specifically, we divided 80% and 20% of the entire dataset into the training and testing datasets. After this step, one-hot encoding was applied so that a total of seventy-eight features were fed into each deep learning model as input data. All the models performed training with five epochs, and the F1 score was employed to measure detection accuracy for binary classification.

Figs. 6 and 7 below depicts the first experimental results in terms of training loss, validation loss, training accuracy, and validation accuracy of the DNN model.

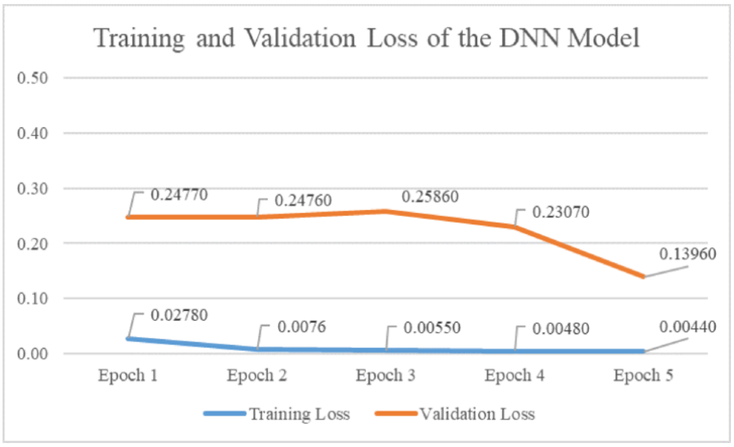

**Fig. 6.** *Training and validation loss of the DNN model*

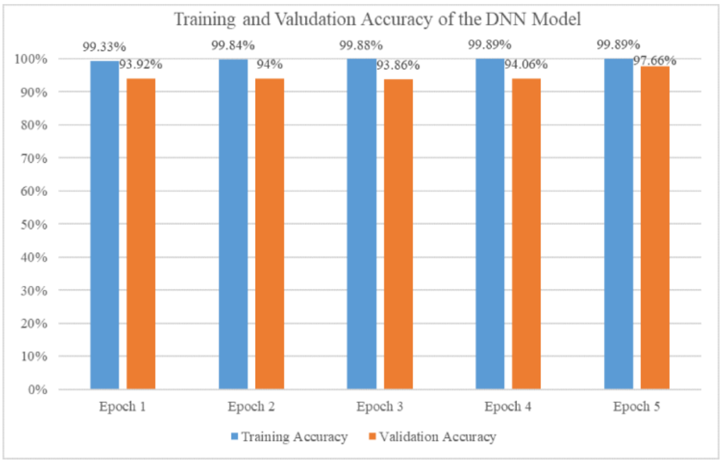

**Fig. 7.** *Training and validation accuracy of the DNN model*

As shown in Figs 6 and 7, we observed that validation loss is higher than training loss, and validation accuracy is lower than training accuracy. The experimental results above seem that it may doubt overfitting, but since both validation results are getting close to training loss and accuracy, we believe that the DNN is not likely overfitting the training data. Based on this, we trained both CNN and RNN models as well, and Figs 8 through 11 depict training loss, validation loss, training accuracy, and validation accuracy of both models.

After training three deep learning models, all models show the same trend that validation results are getting close to training loss and accuracy. Once training our deep learning models is done, we measured the F1 score through the confusion matrix, and the following Table 3 summarizes the detection accuracy of each deep learning model against the CICIDS 2017 dataset.

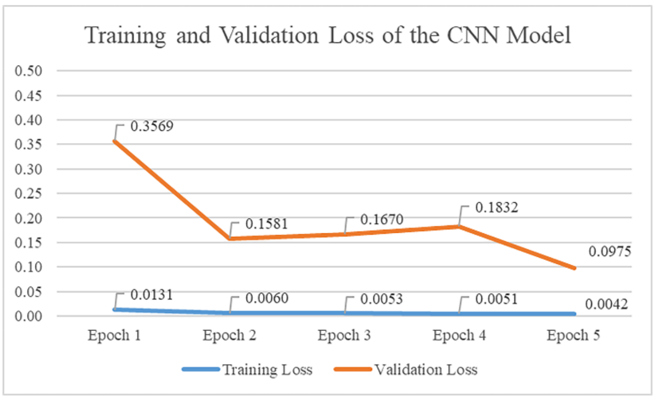

**Fig. 8.** *Training and validation loss of the CNN model*

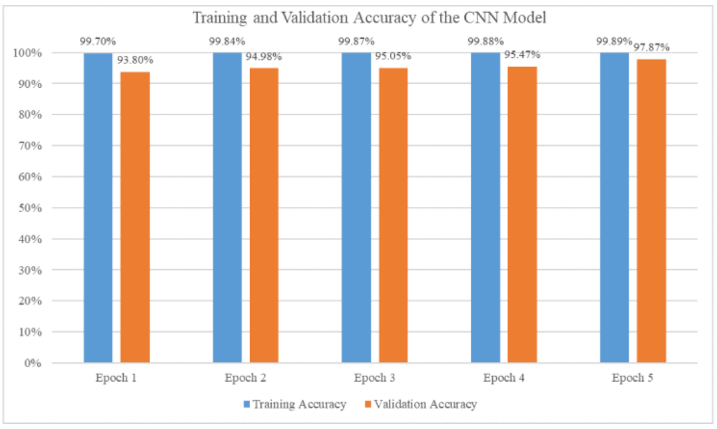

**Fig. 9.** *Training and validation accuracy of the CNN model*

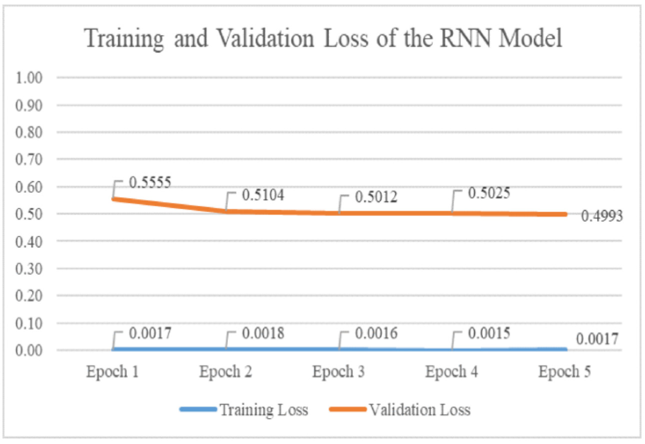

**Fig. 10.** *Training and validation loss of the RNN model*

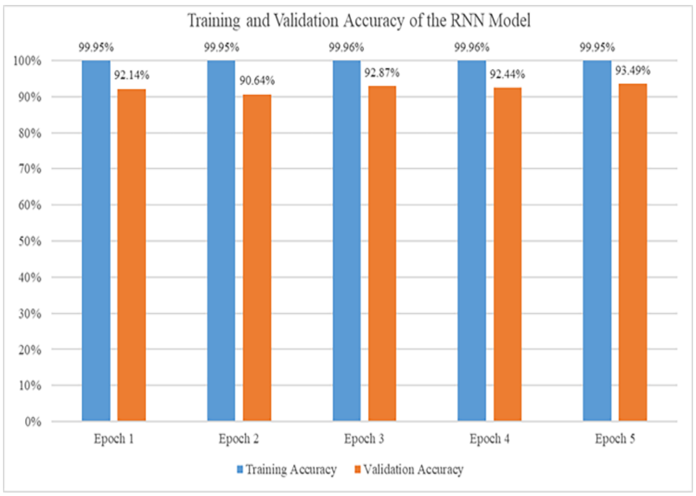

**Fig. 11.** *Training and validation accuracy of the DNN model*

**Table 3.** *A Summary of Experimental Results*

| | | Precision | Recall | F1 Score |
|---|---|---|---|---|
| DNN | Benign | 0.9986 | 0.9762 | 98.14% |
| | DDoS | 0.9336 | 0.9960 | |
| CNN | Benign | 0.9987 | 0.9526 | 96.43% |
| | DDoS | 0.8761 | 0.9932 | |
| RNN | Benign | 0.9988 | 0.9109 | 93.48% |
| | DDoS | 0.7899 | 0.9967 | |

According to Table 3 above, all three deep learning models showed satisfactory results, but interestingly, the DNN model outperformed the other two models. Although no assumptions were made before starting this research, we honestly expected that the RNN model would show the highest accuracy, given that the RNN model was up-to-date compared to the DNN and CNN models.

## CONCLUSION

Network anomaly detection is one of the most significant issues in network security. In this paper, we presented an examination for network anomaly detection with the CICIDS2017 dataset using the DNN, CNN, and RNN models. Our experimental results show that three deep learning models have high detection accuracy, but the DNN model outperformed the other two deep learning models, which achieved an overall accuracy of 98.14%. We think deep learning models eventually seem to be a robust potential tool for network anomaly detection in the cybersecurity field, but some future research works need to be conducted.

In a simple deep learning model architecture, the DNN model showed the highest accuracy, but in fact, the architecture of deep learning may be deeply designed. It is necessary to verify which models show the highest accuracy in a deeply designed deep learning architecture. It is also questionable how well the deep learning model actually works in the field of cybersecurity because only a straightforward binary label dataset was used. Therefore, it is necessary to verify that the deep learning models are good at detecting network anomalies using multi-label datasets. The last future work we are thinking of is to build unsupervised deep learning models. Since our deep learning model is supervised-learning based, showing high accuracy may be a natural result. Thus, it is our final future research work to develop an unsupervised learning-based deep learning model that can expect high accuracy and be applied directly to real life.

## REFERENCES

B. Subramanian, K. Nathani, and S. Rathnasamy, “IoT Technology, Applications and Challenges: A Contemporary Survey,” *Wireless Personal Communications*, https://doi.org/10.1007/s11277-019-06407-w, 2019.

N. Al-Suwaidi, H. Nobanee, and F. Jabeen, “Estimating Causes of Cyber Crime: Evidence from Panel Data FGLS Estimator,” *International Journal of Cyber Criminology*, Vol. 12, No. 2, pp. 392-407, 2018.

S. Morgan, “Top 5 Cybersecurity Facts, Figures, Predictions, And Statistics For 2019 To 2021,” https://cybersecurityventures.com/top-5-cybersecurity-facts-figures-predictions-and-statistics-for-2019-to-2021/

D. Kwon et. al, "An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks," *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS),* Vienna, pp. 1595-1598, 2018.

M.Z. Alom et. al, “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches,” *arXiv:1803.01164*. 2018.

M. Shahriar and A. Namadchian, “A new deep learning approach for anomaly base IDS using memetic classifier.”, *International Journal of Computers Communications and Control*, Vol. 12, No. 5, pp. 677-688, 2017.

A.M. Aleesa, B.B. Zaidan, A.A. Zaidan, and N.M. Sahar, “Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions,” *Neural Computing and Applications*, https://doi.org/10.1007/s00521-019-04557-3, 2019.

T.C. Truong, Q.B. Diep, and I. Zelinka, “Artificial Intelligence in the Cyber Domain: Offense and Defense,” *Symmetry*, 12(3), 410, 2020.

W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative Study of CNN and RNN for Natural Language Processing.” ArXiv abs/1702.01923 (2017): n. pag.

“Intrusion Detection Evaluation Dataset (CICIDS2017),” https://www.unb.ca/cic/datasets/ids-2017.html

H. Liu and B. Lang, “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey,” *Applied Sciences*, Vol. 9, Number 20, 2019.

J. Liu et. al., “Deep Anomaly Detection in Packet Payload,” *arXiv preprint arXiv:1912.02549*, 2019.

M.Z. Alom et. al, “A State-of-the-Art Survey on Deep Learning Theory and Architectures,” *electronics*, 8(3), 292, 2019.

D. Kwon, R. Malaiya, G. Yoon, J. Ryu, and S. Pi, “A Study on Development of the Camera-Based Blind Spot Detection System Using the Deep Learning Methodology,” *Applied Science*, Vol. 9, Number. 14, 2019.

D. Rengasamy, M. Jafari, B. Rothwell, X. Chen, and G.P. Figueredo, “Deep Learning with Dynamically Weighted Loss Function for Sensor-Based Prognostics and Health Management,” *Sensors*, Vol. 23, No. 3, 2020.

C.E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,” *arXiv preprint arXiv:1811.03378*, 2018.

B. Gao and L. Pavel, “On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,” *arXiv preprint arXiv:1704.00805*, 2017.

J. Gu et. al., “Recent Advances in Convolutional Neural Networks,” *Pattern Recognition*, pp. 354-377, 2018.

C. Lee, P.W. Gallagher, and Z. Tu, “Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree,” *Artificial Intelligence and Statistics*, pp. 464-472, 2016.

D. Scherer, A. Műller, S. Behnke, “Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,” In *International conference on artificial neural networks, Springer,* Berlin, Heidelberg, pp. 92-101, 2010.

C.E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,” *arXiv preprint arXiv:1811.03378*, 2018.

A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network,” *arXiv preprint arXiv:1808.03314*, 2018.

H. Salehinejad, S. Sankar, J. Barfett, R. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” *arXiv preprint arXiv:1801.01078*, 2017.

R. Dey and F. M. Salem, “Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks,” 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). IEEE, pp. 1597-1600, 2017.

J. Kim and H. Kim, “Classification performance using gated recurrent unit recurrent neural network on energy disaggregation,” 2016 *International Conference on Machine Learning and Cybernetics (ICMLC),* Vol. 1, pp.105-110, 2016.

X. Wang, W. Jiang, and Z. Luo, “Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts,” *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers,* pp. 2428-2437, 2016.

X. Le, H.V. Ho, G. Lee, and S. Jung, “Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting,” *Water*, 11(7), 1387, 2019.