Journal of Computer and Knowledge Engineering, Vol. 5, No. 2, 2022. (31-44)
DOI: 10.22067/cke.2022.75815.1055

A Deep Neural Network Architecture for Intrusion Detection in Software-Defined Networks*

Research Article

Somayeh Jafari Horestani[1] Somayeh Soltani[2] Seyed Amin Hosseini Seno[3]

Abstract: For more comprehensive security of a computer network as well as the use of firewall and anti-virus security equipment, intrusion detection systems (IDSs) are needed to detect the malicious activity of intruders. Therefore, the introduction of a high-precision intrusion detection system is critical for the network. Generally, the general framework of the proposed intrusion detection models is the use of text classification, and today deep neural networks (DNNs) are one of the top classifiers. A variety of DNN-based intrusion detection models have been proposed for software-defined networks (SDNs); however, these methods often report performance metrics solely on one well-known dataset. In this paper, we present a DNN-based IDS model with a 12-layer arrangement which works well on three datasets, namely, NSL-KDD, KDD99, and UNSW-NB15. The layered layout of the proposed model is considered the same for all the three datasets, which is one of the strengths of the proposed model. To evaluate the proposed solution, six other DNN-based IDS models have been designed. The values of the evaluation metrics, including accuracy, precision, recall, F-measure, and loss function, show the superiority of the proposed model over these six models. In addition, the proposed model is compared with several recent articles in this field, and the superiority of the proposed solution is shown.

Keywords: Intrusion Detection, Software-defined Network, Deep Learning, Network Security

1. Introduction

In computer systems and networks, the attackers exploit security vulnerabilities to attack the network; therefore, there is a need for some methods to detect intrusions into a computer system or network. An intrusion detection system (IDS) is the software or hardware that detects and reacts to intrusions. An IDS prevents illegal access and tampering with the resources of a computer system or network [1-3]. Generally, the IDS monitors the activities of the host computer or the entire network and reports the violations of management and security policies to the network administrator [4-6].

With the growing use of the Internet, network traffic is becoming increasingly complex, and the challenge is becoming more difficult for IDS to detect attacks or anomalies more accurately and quickly. Therefore, researchers leverage machine learning techniques to improve the capability of IDSs.

In the category of machine learning, artificial neural network (ANN) is one of the most widely used models. It is a computational technique widely used in data processing, pattern recognition, and information classification. Deep learning, a subset of machine learning, attempts to extract high level features from the raw input using several hidden layers. Deep neural networks are used in the design of IDSs for software-defined networks (SDNs).

In recent years, several approaches for intrusion detection have been proposed using machine learning techniques; however, each of the methods has its challenges and problems. For example, most studies have reported good accuracy rates, while they have not reported other metrics such as precision or recall. Some methods have reported relatively low values for these performance measures. Another weakness of these methods is that they work only on one dataset and do not evaluate their methods on larger and newer datasets. On the other hand, some studies have compared their methods with only simple classifiers. However, it is clear that this kind of comparison does not have the necessary quality. In this paper, we offer an intrusion detection method for software-based networks using deep neural networks; the proposed method achieves high performance on several datasets.

The contributions of this work can be summarized as follows:

- It provides a comprehensive and complete classification (Research Tree) in the field of intrusion detection systems.
- It follows a deep learning approach to IDS using deep neural networks in software-defined networks.
- It provides seven neural network-based IDS models and evaluates them on three datasets, namely NSL-KDD, KDD99, and UNSW-NB15. The best model, which has the best accuracy, precision, recall, and F-measure values on all datasets, is then introduced.
- One of the strengths of this solution is that the layered layouts of the proposed models are the same for all three datasets.

The paper then presents the theoretical background and research motivation, discusses the proposed model, evaluates the proposed model, and finally concludes the work.

2. Research background

In general, intrusion detection systems can be categorized in terms of various aspects, such as detection method (or analysis technique), type of architecture, how to respond and react to intrusion, information source, and many others [7-12]. For example, intrusion detection systems can be divided into two types of continuous monitoring and periodic

* Manuscript received: 13 March 2022; Revised, 01 July 2022, Accepted, 01 August 2022.
[1]. MSc, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
[2]. Corresponding Author, PhD, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
Email: somayeh.soltani@mail.um.ac.ir
[3]. Associate Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

analysis in terms of continuity [13-15]. They can also be divided into active and passive responses [16-19].

Chalapathy and Chawla [7] categorized the deep learning-based anomaly detection techniques using three criteria: application, type of anomaly, and type of model. Then they defined nine applications, that is, fraud detection, cyber intrusion detection, medical anomaly detection, sensor network anomaly detection, video surveillance, IoT big data anomaly detection, log anomaly detection, and industrial damage detection. They defined three types of anomalies: collective, contextual, and point. Moreover, they considered four types of detection models: unsupervised, semi-supervised, hybrid, and one-class neural networks.

Kwon et al. [9] classified the anomaly-based IDSs into two groups: programmed and self-learning. Then they classified the programmed IDSs into two categories of simple-rule and statistical-based, and they categorized the self-learning IDSs into four categories: cognition-based, computation-intelligence, data mining, and machine learning. In the next step, they classified the machine learning-based IDSs into six groups: Bayesian network, genetic algorithm, fuzzy logic, artificial neural network (ANN), supervised vector machine (SVM), and outlier detection. Furthermore, they defined two types of ANNs: supervised and unsupervised. The supervised ANN IDSs can be free-forward ANN or recurrent ANN. The unsupervised methods include deep learning, adaptive resonance theory, and self-organizing maps. Finally, the deep learning methods include AutoEncoder, sum-product network, recurrent neural network (RNN), Boltzmann machine (BM), convolutional neural network (CNN), and deep neural network (DNN).

Lee et al. [18] categorized deep learning-based IDS schemes into nine classes: AutoEncoder-based, RBM-based, DBN-based, DNN-based, CNN-based, GAN-based, LSTM-based, RNN-based, and hybrid. They then classified the AutoEncoder-based schemes into six groups: Stacked AutoEncoder, Denoising AutoEncoder, NonSymmetric AutoEncoder, Sparse AutoEncoder, Variational AutoEncoder, and Convolutional AutoEncoder. They also defined several hybrid schemes: AE+CNN, AE+DBN, AE+DNN, AE+GAN, AE+LSTM, CNN+LSTM, CNN+RNN, and DNN+RNN.

Having reviewed various articles in the field of intrusion detection systems, we categorized these systems in different ways. In terms of continuality, we classified intrusion detection systems into two categories: continuous monitoring and periodic analysis. Concerning reaction to influence, we divided these systems into two groups: active response and passive response. Regarding the architecture, we divided the IDSs into two groups, centralized and distributed. In addition, we defined two types of real-time or offline forecasting.

In terms of the knowledge base, we considered three classes: Boltzmann machine, descriptive languages, and expert systems. We classified the IDSs into three groups: one variable, multivariate, and time series model. Moreover, the IDS systems are categorized into two classes: anomaly-based and signature-based. We considered three signature-based techniques: data mining, state transition, and expert systems.

Anomaly-based techniques are divided into two groups: self-learning, and programming. The self-learning techniques are cognition-based and relate to computation intelligence, data mining, or machine learning. The machine learning techniques can be semi-supervised, supervised, unsupervised, or reinforcement learning. Each of these techniques has so many subcategories.

We summarize various categorizations in a tree named Research Tree in the field of intrusion detection systems. Figure 1. shows the comprehensive classification tree.

In the following, we categorize previous research works into two main groups in terms of the model architecture: 1) works done on shallow architectures, 2) works done on deep architectures.

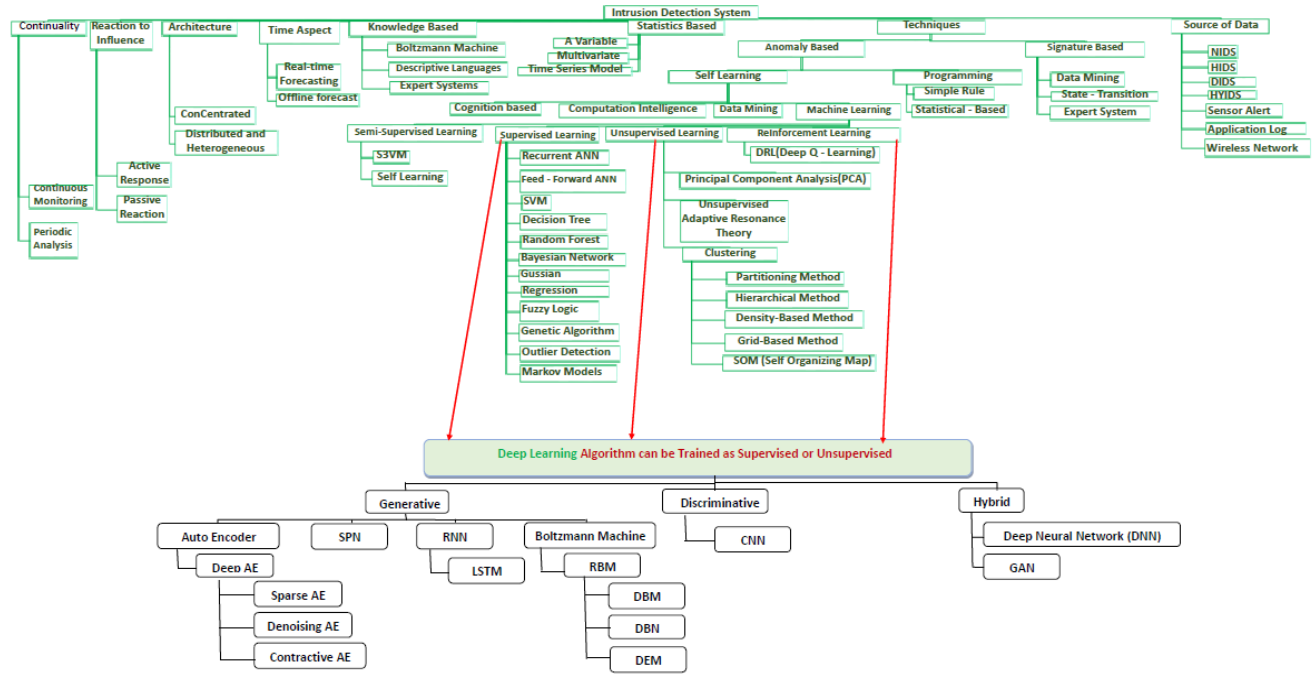

Figure 1. Research Tree in the field of intrusion detection systems

2.1. Shallow Learning IDSs

Some intrusion detection methods use shallow architectures, such as support vector machine (SVM), decision tree (DT), random forests (RF), clustering, K nearest neighbor (KNN), particle swarm optimization (PSO), simulated annealing (SA), ANN, and ensemble methods [4, 20-27].

Lin et al. [4] used the SVM, the decision tree, and the simulated annealing and reached 99.96% accuracy. Wang et al. [23] used the SVM algorithm and reached 99.31% accuracy. Baek et al. [22] achieved an 88% accuracy rate using several simple classifications.

These methods take advantage of the mentioned algorithms and use KDD99 or NSL-KDD datasets to evaluate their solutions and report good accuracy or precision rates. However, these methods report only one metric of accuracy or precision and no other metrics. They use only one dataset for evaluation, and they compare the results with only ordinary classifications.

2.2. Deep Learning IDSs

In this section, we describe intrusion detection models based on deep learning methods.

A. Convolutional Neural Network (CNN)

This category includes research works which have based their intrusion detection techniques on convolutional neural networks [28-32]. Zhu et al. [28] considered 6 layers of the neural networks and used the pooling layer among them. Moreover, they used a learning rate of 0.5 and achieved 80.34% of Accuracy. Li et al. [29] used convolution architecture and data-to-image conversion techniques to detect intrusion but provided a relatively low accuracy rate (about 80%). Nguyen et al. [32] used a deep convolutional network and used 4 main layers of CNN networks. They reported 99.87% accuracy on the KDD99 dataset.

B. Recursive Neural Network (RNN) or Gated Recurrent Unit RNN (GRU–RNN)

Research works in this category have used recursive neural network techniques [33-37]. For example, Yin et al. [33] proposed a binary classification method based on a deep recursive neural network to detect intrusions. They first performed pre-processing (such as normalization) on the input dataset and then attempted to weigh the deep network layers using a recursive neural network with forward propagation, reporting 99.81% accuracy. Tang et al. [34] proposed a gated recurrent unit (GRU) over SDN-based networks. They compared their method with DNN classifiers having different layouts, support vector machines, and simple Bayesian, and reported 89% accuracy and 87% precision. Zhong et al. [37] presented an IDS for IoT servers using text-CNN and GRU methods. They reported the F-score criterion on the KDD99 and ADFA-LD datasets.

C. Long Short-Term Memory (LSTM)

Ponkarthika and Saraswathy [38] developed an intrusion detection system based on the RNN and its specific type, and LSTM networks. They achieved 82% accuracy for the RNN and 83% accuracy for the LSTM on the KDD99 dataset with a learning rate of 200.

D. CNN-RNN

Vinayakumar et al. [39] proposed an intrusion detection technique using the convolutional network for feature extraction and the RNN network for classification. They proposed a CNN-based model and showed that the CNN network would perform better than MLP, CNN-LSTM, and CNN-GRU in extracting and presenting features from network traffic. Their model could report the highest accuracy and recall on single-layer CNN and the highest precision on almost all CNN combinations with other networks on the KDD99 dataset at 99.9%. Chawla et al. [40] proposed a technique using a combined convolutional network and GRU RNN; they also could achieve 81% accuracy on the ADFA-LD dataset with a learning rate of 0.0001.

E. CNN-LSTM

The intrusion detection method proposed by Wang et al. [41] uses the convolution filter to extract the feature and the LSTM network for classification. That is, it uses CNN deep networks to learn low-level features and LSTM networks to learn high-level features. This method reported 99.89% accuracy on the ISCX2012 dataset. Furthermore, Hsu et al. [42] used a hybrid method based on LSTM and convolution network to detect intrusion and reported 94.12% accuracy on a larger dataset. Lee et al. [43] designed an intrusion detection system to prevent SSH and DDOS attacks in software-defined networks, which used four deep learning models, including MLP, CNN, LSTM, and SAE. Malik et al. [44] designed an Efficient Reconnaissance and Surveillance Detection in SDN using CNN and LSTM; however, they evaluated their model using only one dataset, namely CICIDS 2017.

F. RNN-LSTM

Jiang et al. [6] developed a multi-channel intelligent attack detection technique based on a combination of LSTM and RNN networks. In this LSTM-RNN architecture, multiple feature channels are given to the network input layer. Then, the LSTM layer, the Mean Pooling layer and finally the logistic regression layer are used. Finally, a majority vote is taken on the results obtained. Jiang et al. reported a detection rate of 99.23 and an accuracy of 98.94% on the NSL-KDD dataset.

G. Auto Encoder

The articles in this category [45-50] use deep Auto Encoder neural networks. Mohammadi and Namadchian [45] first performed normalization and then used a deep Auto Encoder method to reduce the error rate. Finally, on the NSL-KDD dataset, they achieved 92.72% accuracy and 98.11% detection rate in the classification of R2L attacks.

Papamartzivanos et al. [49] provided a comprehensive framework based on self-taught learning and MAPE-K methodology. The framework included plan, monitor, analyze, and execute activities that are applied to a knowledge base. Their model was a Sparse Auto Encoder and a Feedforward Auto Encoder. Their tests on the KDD99 and NSL-KDD datasets reported 99.8% and 99.6% accuracy.

H. Deep Neural Network (DNN)

The models in this category [5,51-58] use deep neural networks to detect network intrusions. In 2019, Vinayakumar et al. [5] proposed an IDS based on deep neural network and tested it on six datasets. The model layers

included the fully connected layer, the normalization layer, and the drop-out layer with a coefficient of 0.01. They used 16 consecutive layers, several ReLU activation functions, and learning rates between 0.01 and 0.5. This model reported good accuracy on six well-known datasets (e.g. 96.3% on the CICIDS dataset or 93% on the KDD99 dataset).

Tang et al. [51] also proposed a DNN-based intrusion detection method which was performed on the SDN environment and the NSL-KDD data set, and its experiments with a learning rate of 0.0001 reported an accuracy of 75.75. Using the Boltzmann neural network on the KDD99 dataset, Roy et al. [52] were able to report a very high 99.99% accuracy for two-class mode (attack or normal).

Ustebay et al. [53] used both the deep neural network and the shallow neural network (SNN) to detect abnormalities. They used these two models to reduce the feature set. They also trained the models on the CICIDS2017 dataset and reported a 98.45% accuracy rate on the deep models. They showed that deep models would achieve higher accuracy, precision, and recall than shallow models.

Vigneswaran et al. [54] evaluated DNN and SNN models on NIDS. They performed experiments on the DNN architecture with 1 to 5 layers at a learning rate of 0.1, considered 1000 Epochs on the KDD99 dataset, and finally compared the results with shallow machine learning algorithms. The results showed that the three-layer DNN had the best accuracy of 93% and precision of 99% among all these algorithms.

Duy et al. [57] designed a framework called DIGFuPAS which creates attack examples and acts like deep learning-based IDS in SDN in a Black-Box manner. They used Wassertein Generative Adversarial (WGAN) Model, a generative model based on deep learning. Bouria and Guerroumi [58] presented an IDS based on a deep learning approach to strengthen SDN network security. The communication channel between the control layer and the infrastructure layer of the SDN is protected against various attacks. Moreover, they evaluated their model only on the CICIDS 2017 dataset.

2.3. Software-Defined Networks

Software-defined network is a new type of network architecture in which one or more central servers are responsible for controlling all network elements, whereas the rest of the elements only direct network traffic [59, 60]. Traditional networks were suitable for a static client-server structure. But today's modern networks, including data centers, cloud services, mobiles, and IoT devices, demand new requirements.

As you know, in traditional networks, each network device calculates routes and makes decisions on network policies. However, in SDN networks, the network operating system (the controller) is responsible for deciding how to route packets and applying network policies. The most essential concept in SDN networks is to separate the control plane and data plane. While the control plane decides how to route the packets, the switches and routers merely forward packets and are not involved in decision-making.

Apart from the controller and the network devices, some other components constitute the SDN architecture. For example, the SDN applications express their desired network behavior to the controller using some interfaces. Moreover, the OpenFlow protocol communicates between the control and the data planes.

While SDN provides easy, flexible, and integrated management, it imposes several security issues. As the control logic in SDN is centralized, it is more vulnerable to cyber-attacks such as DDoS; therefore, the design of security appliances for SDN networks is crucial [61, 62].

3. The proposed model

The proposed model consists of three phases: 1) preprocessing phase, 2) neural network design phase, and 3) intrusion detection phase. In the first phase, the necessary pre-processing is performed on the raw data collected from the SDN network traffic. In the second phase, the neural network is designed with the appropriate layer arrangement and the proper activating function. The model is trained on the training dataset with the required number of repetitions. In the third phase, the trained model is tested on a test dataset, and the performance of the model is evaluated using various metrics such as accuracy, precision, and recall.

In deep learning, the goal of training is to increase the performance of the model using the defined training set. To measure the performance, we defined a loss function and reduced it in the hope that it would improve the overall performance of the model. While there are many loss functions to compute the distance between the true value and the estimated one, Cross entropy [3] is the most popular. In this research, we used the Cross entropy and the Adam optimizer for all three datasets. Cross entropy for a classification problem with n classes is defined as (1):

$$CE = -\sum_{i=1}^{n} t_i \, log(p_i)\,, \tag{1}$$

where t_i is the true value and p_i is the probability for the i^{th} class.

In this research, we considered seven different configurations for the neural network and evaluated all these seven models on three datasets, NSL-KDD, KDD99, and UNSW-NB15. It should be noted that the architecture and layered layout of the proposed models are the same for all three datasets, which is one of the strengths of our solution. To achieve proper performance, many previous models [5,24,28,59] have offered different layout layers for each data set, but our proposed architecture achieved good performance for all three datasets without manipulation.

Each of these seven models had a unique layout consisting of several layers, such as embedding, Dense, Drop out, and activation layers. The first model was a model based on dense layers and had nine layers. The second model was based on the convolutional neural network (CNN) and had the largest number of layers (22 layers). The third model was a 10-layer LSTM-CNN hybrid network. The fourth model was based on the dense network with the least number of layers. The fifth model, like the second one, was a CNN-based model with a relatively large number of layers. The sixth model was based on the LSTM-CNN hybrid network and had 12 layers. Finally, the seventh model was based on the dense network and had 12 layers. The designs of these seven models are described in Table 1. The number of neurons in each layer is represented in parentheses.

Table 1. The layout of the seven proposed ANN-based IDSs

Layers	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Layer 1	Dense(128)	Embedding	Embedding	Dense(128)	Embedding	Embedding	Dense(64)
Layer 2	Dense(64)	Dropout	Dropout	Dense(64)	Dropout	Dropout	Dense(32)
Layer 3	Dense(128)	Conv(256)	Conv(32)	Dense(128)	Conv(256)	Conv(64)	Activation
Layer 4	Dropout	Dense(100)	Conv(64)	Dense(64)	Dense(100)	Dropout	Dropout
Layer 5	Dense(64)	Dropout	Conv(128)	Dense(6)	Dropout	Dropout	Dense(32)
Layer 6	Dropout	Conv(128)	LSTM(128)	Dense(128)	Conv(128)	Conv(64)	Activation
Layer 7	Dense(6)	Dense(100)	Dense(100)	FC	Dense(100)	Dropout	Dropout
Layer 8	Dropout	Dropout	Dropout	---	Dropout	LSTM(300)	Dense(32)
Layer 9	FC	Conv(256)	Dense(100)	---	Conv(128)	Dense(100)	Activation
Layer 10	---	Dense(100)	FC	---	Dense(100)	Dropout	Dropout
Layer 11	---	Dropout	---	---	Dense(100)	Dense(10)	Dense(32)
Layer 12	---	Dense(200)	---	---	Dropout	FC	FC
Layer 13	---	Dropout	---	---	Dense(200)	---	---
Layer 14	---	Dense(100)	---	---	Conv(32)	---	---
Layer 15	---	Conv(32)	---	---	Conv(64)	---	---
Layer 16	---	Max-pool	---	---	Conv(128)	---	---
Layer 17	---	Dropout	---	---	Dense(100)	---	---
Layer 18	---	Dense(256)	---	---	Dropout	---	---
Layer 19	---	Dropout	---	---	Dense(256)	---	---
Layer 20	---	Dense(100)	---	---	Dropout	---	---
Layer 21	---	Dropout	---	---	FC	---	---
Layer 22	---	FC	---	---	---	---	---

We tested all seven proposed models against three datasets NSL-KDD, KDD99, and UNSW-NB15 and selected the best one (i.e., Model 7).

The best-proposed model (Model 7) was a unique 12-layer deep neural network with the following layer topology: dense, dense, activation, drop out, dense, activation, drop out, dense, activation, drop out, dense, and finally activation or fully connected (FC) layer which is used to select the appropriate class using SoftMax or Tanh functions. From now on, we will call Model 7 the proposed model. The proposed model improves the evaluation metrics without changing the number and layout of layers on the three datasets. This is the superiority of our solution over other works, which provides a different network architecture for each dataset.

However, it should be noted that since these three datasets are different in terms of the number of parameters and the number of output classes, our model also considers different parameters and final activation functions for selecting output classes. Also, the initial values for the dense layers are slightly different for each dataset. In the following, we will examine the layers of the proposed model.

Dense Layer: The values of the dense layers in the proposed model are different for each dataset and depend on the number of dataset properties. For example, for the UNSW-NB15 dataset, 64 values are provided for the first layer. The activation function is also one of the best-tested functions for the neural network. The non-linear ReLU function is defined in the following (2):

$$ReLU(x) = max(0, x), \tag{2}$$

where x is the input.

Drop out layer: The drop out layer accidentally removes and releases some neurons, preventing the network overfit. Therefore, it does not allow the network to retain data and to be disturbed in predicting the testing data. In the proposed model, a drop out layer with values of 0.15 to 0.5 is considered after each dense layer.

Fully connected (FC) layer: A fully connected layer (unlike a dense layer) is a layer that connects to all the neurons in the previous layer. It considers the trained inputs in the previous layers and assigns them to the appropriate class using an activation function such as SoftMax or Tanh, as defined in (3) and (4). Figure 2 illustrates the proposed model diagram:

$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \tag{3}$$

$$Tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}, \tag{4}$$

where x is the input.

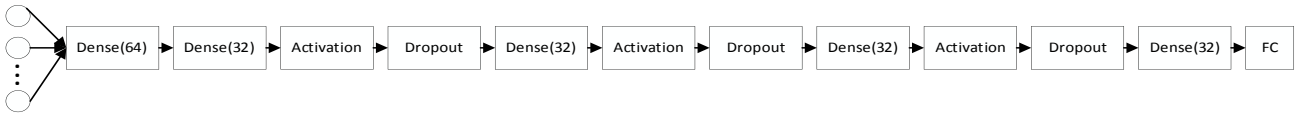

Figure 2. The layer layout of the proposed solution

The implementation of the proposed solution is described below. The data collected from the data sets of SDN networks were prepared and modeled. The layout, number, and type of network layers were set, the activation functions were selected, and the specified attack classes were converted into vectors. Then, the dataset was tagged with attack and non-attack labels. Finally, the input data, the output classes, layers, and weights of the data set in CSV format would make the DNN neural networks. We used TensorFlow and Keras deep learning package and Python programming language. To train the model faster and use powerful GPUs, we used the Google Colab service. We trained the model and the network with a suitable number of iterations and made sure of avoiding overfitting in each epoch. Finally, we tested the model using the test data set and evaluated the improvement of the model's performance in terms of accuracy, precision, recall, and cost function.

4. The evaluation of the proposed model

4.1. Datasets

In this study, three datasets, including NSL-KDD, KDD99, and UNSW-NB15, were used as benchmarks to select the best model among seven models and to compare the proposed model with other methods.

A. KDD99 dataset

The KDD99 dataset is an old dataset containing 41 features and five different classes: normal, DoS, remote-to-local (R2L), user-to-root (U2R), and Prob. It includes 494,021 records for training and 311,029 for testing sets. Some of the derived features include duration, protocol_type, service, src_bytes, dst_bytes, flag, urgent, and so on. One drawback of KDD99 is that the sets of classes in the training and testing sets are imbalanced. Moreover, there are many duplicates in the dataset.

B. NSL-KDD dataset

The NSL-KDD dataset is one of the most widely used datasets for intrusion detection research; it is a subset of the original KDD99 and is designed to solve some of the drawbacks of the KDD99 dataset. This dataset does not have duplicate records in the training and testing sets, and the number of records is considered more reasonable and appropriate. The feature set and the type of classes are the same as the original KDD99.

C. UNSW-NB15 dataset

UNSW-NB15 is a relatively new dataset with a hybrid of real normal activities and synthetic contemporary attacks. It has 175,341 records in the train set and 82,332 records in the test set. The dataset has ten classes (normal and nine types of attacks). The attack types are DoS, backdoors, fuzzers, analysis, exploits, generic, shellcode, reconnaissance, and Worms. Moreover, there are 49 derived features.

4.2. Evaluation Metrics

The most important and widely used metrics to evaluate the quality of the results of intrusion detection methods are: 1) accuracy, 2) precision, 3) recall, 4) F-measure, and 5) loss function [49, 63-65]. At first, it is necessary to define the four basic terms used in the mentioned metrics [66]:

- True Positive (TP) indicates the number of records in the dataset that our method correctly classified in the attack class.
- True Negative (TN) is the number of records in the dataset that our method rightly classified in the normal category.
- False Positive (FP) indicates the number of records in the dataset that our method incorrectly classified in the attack class.
- False Negative (FN) is the number of records in the dataset that our method mistakenly classified in the normal category.

In the following, we will explain the application of these basic terms in the mentioned evaluation metrics.

Precision: This metric estimates the ratio of correctly identified attack records to the total number of detected attack records (5):

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

Recall: This metric estimates the ratio of correctly classified attack records to the total number of attack records (6):

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

Accuracy: This metric estimates the ratio of correctly classified records to the total records. In other words, the accuracy metric shows the percentage of the data that are correctly categorized in (7):

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

F-measure: This metric establishes a tradeoff between precision and recall. It is the harmonic mean of precision and recall (8):

$$F = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{8}$$

Loss function: This metric indicates the amount of output error, and we can get good results by optimizing it [36, 51, 67].

4.3. The evaluation of the Proposed Solution

To implement the proposed models, we used the Jupyter Notebook in the free Google Colaboratory service. In particular, we used the Tensorflow 1.0 deep learning package [68] along with Keras Backend and the Adam optimizer with different learning rates. Moreover, we used the most popular cost function, that is, Cross Entropy. The seven neural network models were examined and evaluated on the three datasets with the same conditions.

Table 2. The performance of the seven proposed models on the KDD99 dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	53.96	98.93	50.87	61.1	98.9	88.22	99.02
Precision	83.79	88.67	78.54	98.27	88.38	63.17	99.14
Recall	76.19	86.34	79	41.24	85.94	58.77	98.93
F-measure	78.78	87.49	78.74	58.08	87.14	60	99.04
Loss function	0.046	0.03	0.054	0.415	0.031	0.077	0.107

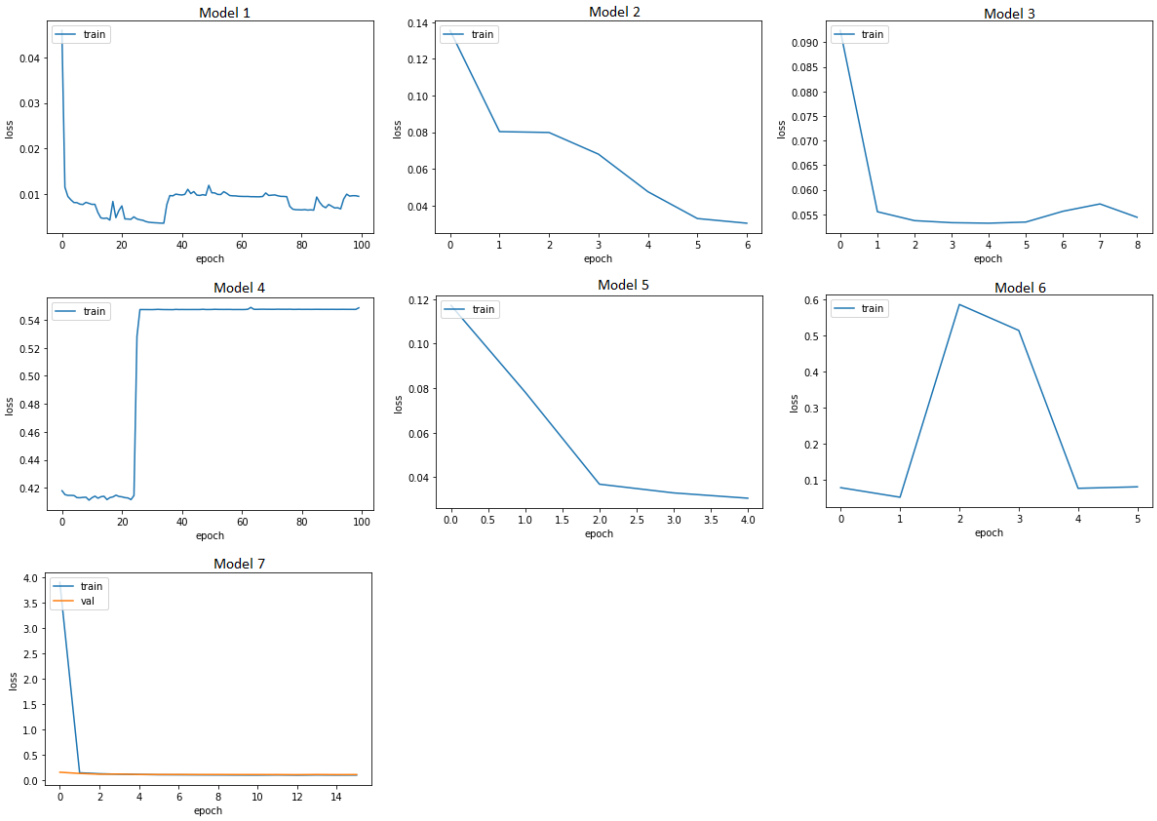

Figure 3. The evaluation of the loss function of the seven proposed models on the KDD99 dataset

A. The evaluation of the seven models on the KDD99 dataset

We examined all seven proposed models with the KDD99 dataset. Table 2 shows the accuracy, precision, recall, F-measure, and loss functions of all the models. The performance of Model 7 was better than the other models in terms of accuracy, precision, recall, and F-measure; however, the loss function of Model 7 was higher than most models. Model 2, with a value of 0.03, had the lowest loss function.

Figure 3. shows the evaluation of the loss function on our seven models for the KDD99 dataset. In Model 7, the cost function for validation data (orange line) was approximately tangent to the cost function for training data (blue line). For other models, only training data were examined due to the imbalanced training and validation data.

In Model 1, with increasing epochs, the error decreased but with fluctuations, which can be attributed to the lack of use of drop out layer for optimal control of overfitting. Model 2 had a stepped decrease. In the first repetitions, it had good learning from training data, however, in the subsequent repetitions, the learning rate decreased. It should be noted that this model had the best value of the loss function.

Model 3 did not reach the minimum value of the loss function but had a good decreasing slope. The loss function of Model 4 not only decreased after a while, but also it showed an increase due to the high learning rate and inappropriate layer arrangement. The lower the learning rate, the greater the possibility of improving the loss function. The loss function of this model was the worst loss function among the design models.

Model 5 initially had a sudden decrease and then reached a slow and relatively uniform decrease. The learning rate gradually improved better in this model. In Model 6, the overfitting fluctuations were uncontrolled, and the error increased and decreased abruptly. The sixth model worked well on the training data. However, after feeding new data, the loss function increased due to not using the drop out layer correctly.

Model 7 (i.e., the best proposed model) did not have the least loss function among all the models, but it was able to control overfitting with the correct arrangement of layers. This model also performed well in the validation dataset.

B. The evaluation of the seven models on the NSL-KDD dataset

Table 3. shows the performance values of the seven models on the NSL-KDD dataset. It is quite clear that the proposed

solution (Model 7) performed much better than other models in all evaluation criteria, even in the loss function. As compared to other models, this showed the excellent performance of the proposed solution and the proper arrangement of the neural network layers in it.

Considering Figure 4, it is clear that Model 7 greatly reduced the loss function. In fact, the loss function had a slight difference in both the training and validation datasets. If the loss function of the training data were close to the loss function of the validation data, it would be safe to say that the over-fitting is well controlled. The loss function in Model 7 reached the lowest possible value among the seven models. One of the reasons for this smooth reduction of the loss function was the correct use of drop out layers between the dense layers.

Model 1 had the lowest loss after model 7. The value of the loss function could be well reduced due to its good learning rate. Of course, the loss function fluctuated with the arrival of some new data, and the network controlled the fluctuations using the appropriate learning rate. The cost function of the second model initially decreased but remained constant after a few iterations. To solve this problem, the learning rate should be adjusted and reduced during the training steps.

The layout of Model 3 was not able to reduce the loss function well. Model 4, like the third model, was subject to fluctuations in new train data. This indicated that the model had learned well from previous data; however, the error fluctuated with new data, which was not very acceptable.

The loss function of Model 5 remained constant very soon and could not reduce the loss function more than this amount. Adjusting the input weights of the next layers was very important. In the fifth model, the input weights of the layers were not well adjusted and had been updated with a constant value, producing a fixed loss function.

Model 6 did not perform well in this dataset. Increasing the learning rate initially reduced the loss function, but the error rate then increased. It seems that by reducing the learning rate in these models, we can solve this problem and improve the model performance. Thus, Model 7 in the NSL-KDD dataset is undoubtedly the best-designed model according to the evaluation criteria under consideration.

Table 3. The performance of the seven proposed models on the NSL-KDD dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	71.75	95.95	96.53	67.08	95.95	61.32	99.39
Precision	83.51	53.26	93.25	91.17	53.48	80.33	99.49
Recall	69.79	53.23	70.23	78.4	53.44	84.11	99.33
F-measure	75.63	53.24	79.98	83.89	53.46	81.74	99.41
Loss function	0.068	0.088	0.084	0.094	0.088	0.076	0.022

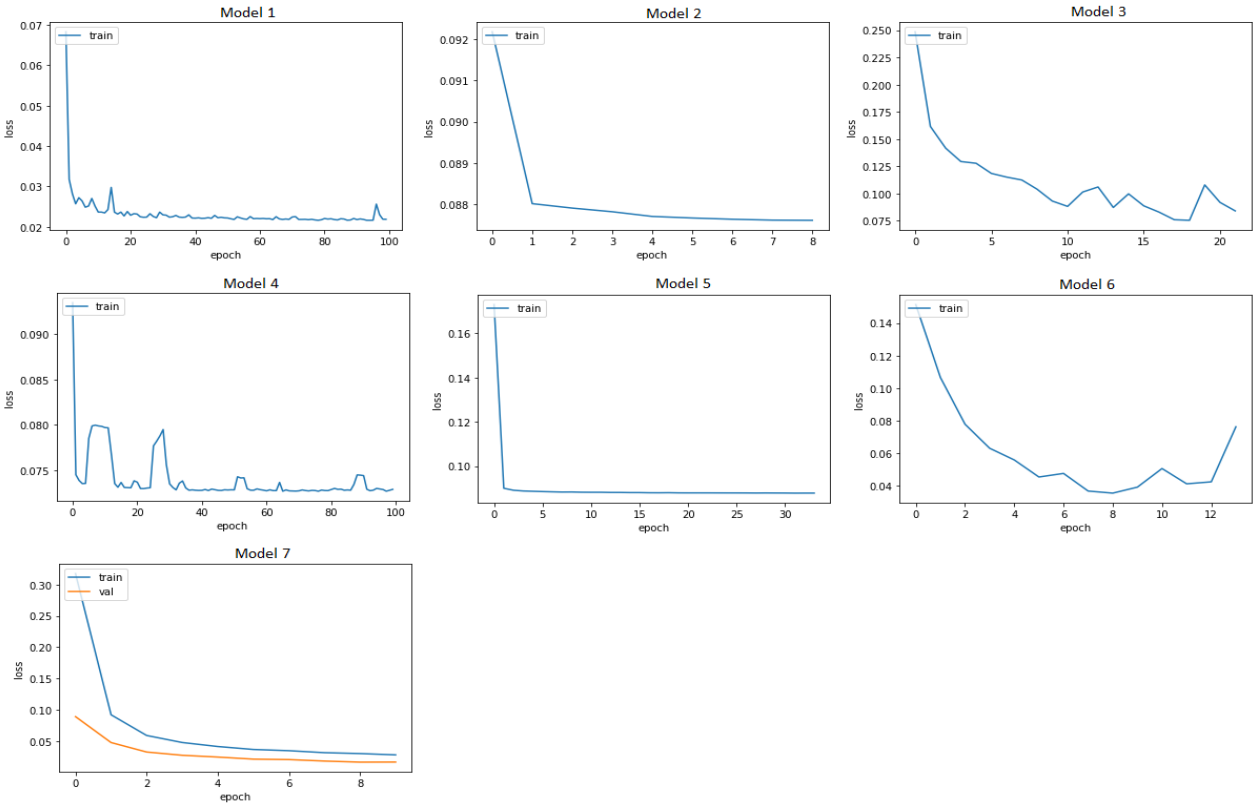

Figure 4. The evaluation of the loss function of the seven proposed models on the NSL-KDD dataset

C. The evaluation of the seven models on the UNSW-NB15 dataset

Table 4. shows the evaluation of our models on the UNSW-NB15 dataset. Model 7 had the highest value in terms of accuracy and F-measure. In addition, this model achieved a recall of 99.98% (approximately one). Although the third and fourth models achieved 100% recall, other performance measures of these two models were lower than Model 7. However, the loss function of Model 7 was higher than Models 2, 4, 5, and 6. It is possible to reduce the error rate by changing the activation function for this dataset. However, changing the activation function is not acceptable, and we consider fixed activation functions for three datasets.

Referring to Table 2, Table 3, and Table 4, it is clear that the highest value for accuracy metric (one of the most important evaluation metrics in intrusion detection systems) on all three datasets of KDD99, NSL-KDD, and UNSW-NB15 belonged to the proposed model (Model 7).

Figure 5 shows the loss functions of the seven models on the UNSW-NB15 dataset. It is clear that the first model increased the error instead of decreasing it and thus had the worst loss function among the seven models. It can be inferred that the final activation function of Model 1 failed to predict the correct class. Given that changing the activation may reduce the error, we did not change it in this study; in fact, we considered fixed activation functions for all three datasets.

The loss function of the second model had a decreasing trend, which implies that the layer arrangement and the final activation function were chosen properly. The third model is almost the same as the second model and has the least loss value. The loss function in the fourth model continuously decreased; however, it did not reach the lowest level and was fixed at approximately 0.6881%.

While the loss functions of the fifth and the sixth models performed similarly, the fifth model acted slightly better. The sixth model had a higher learning rate than the fifth model, but it controlled the overfitting better. The loss function of the proposed model also had a decreasing trend, but its error rate was higher than the other models.

Table 4. The performance of the seven proposed models on the UNSW-NB15 dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	4.3	64.04	55.05	55.04	64.02	66.45	68.11
Precision	3.3	68.5	55.05	55.04	69.15	74.54	68.13
Recall	2.2	65.56	100	100	64.55	59.36	99.98
F-measure	2.2	66.59	71	70.99	66.11	66.06	80.97
Loss function	8.86	0.632	0.34	0.688	0.635	0.739	7.16

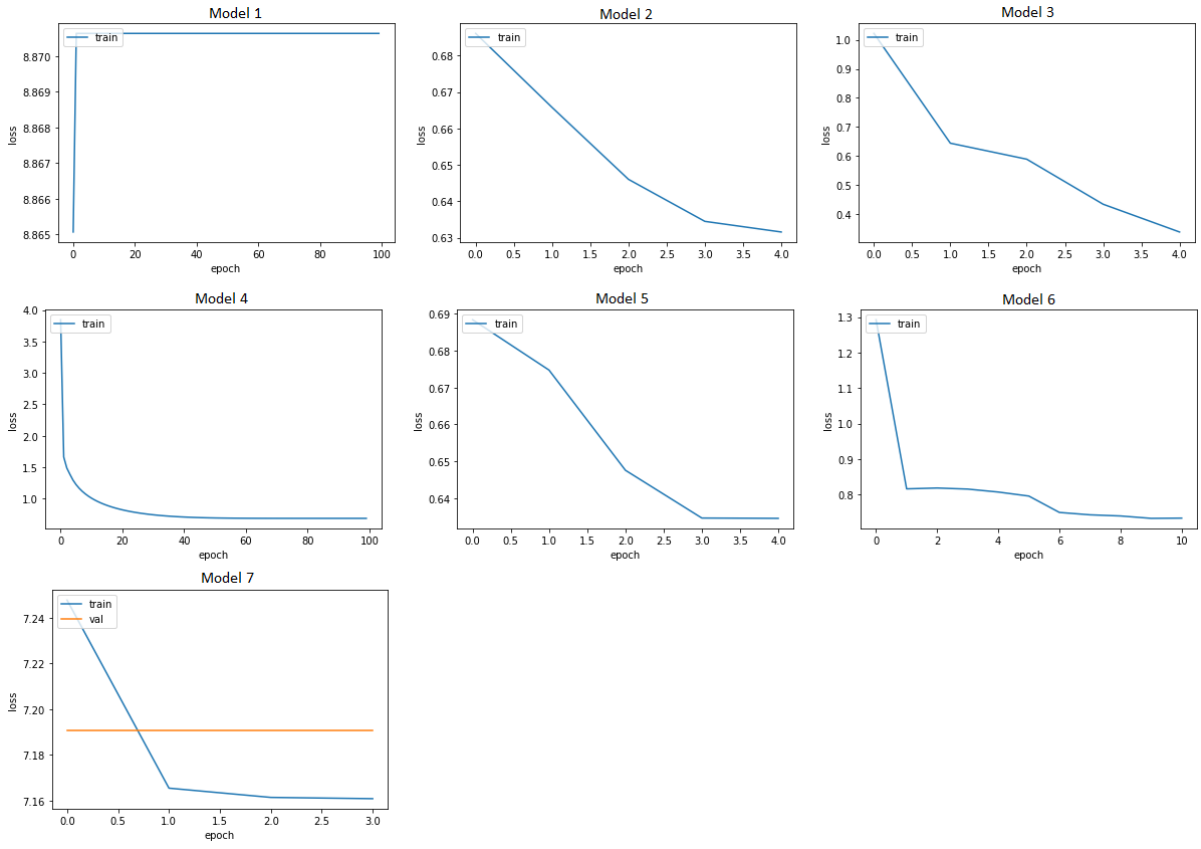

Figure 5. The evaluation of the loss function of the seven proposed models on the UNSW-NB15 dataset

D. Performance Comparison

In this section, we will compare the results of our model with the findings offered by VinayaKumar et al. [5] in which excellent evaluation metrics values are obtained on six datasets.

As shown in Figure 6, the proposed model performs better than VinayaKumar et al.'s work on the KDD99 dataset in terms of all evaluation metrics (accuracy, precision, recall, and F-measure). Figure 7 shows a comparison of the proposed method with VinayaKumar et al.'s method on the NSL-KDD dataset. The proposed method works much better than VinayaKumar et al.'s work. All four criteria in the proposed method are close to 100%, while in VinayaKumar et al.'s method they are about 80%.

The proposed solution and the method offered by VinayaKumar et al. on the UNSW-NB15 dataset were also compared. Figure 8 shows that the accuracy of the proposed model is 68.11%, and the accuracy of the method of VinayaKumar et al. is 65.1%, and therefore the proposed solution works better. Regarding the Precision metric, VinayaKumar et al.'s method is 59.7%, and the proposed method is 68.13%. Also, in the recall metric, the proposed method performs much better than the method of VinayaKumar et al.

It should be noted that the UNSW-NB15 dataset is one of the largest intrusion detection datasets, and the improvement obtained by the proposed method on this dataset is valuable.

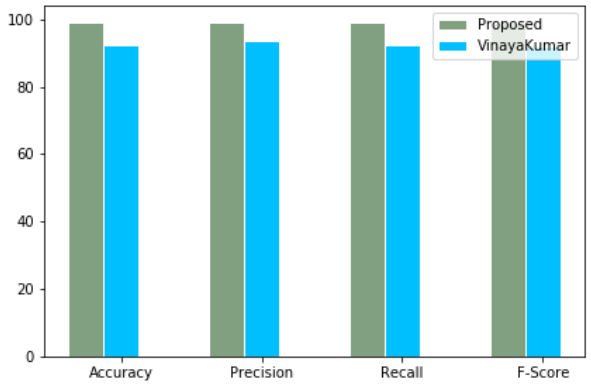

Figure 6. Comparison between the proposed model and VinayaKumar et al.'s work on the KDD99 dataset

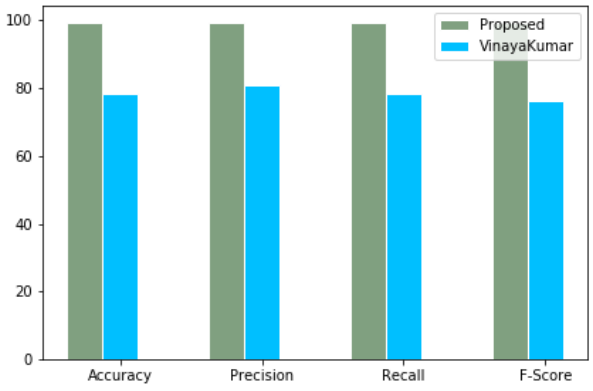

Figure 7. Comparison between the proposed model and VinayaKumar et al.'s work on the NSL-KDD dataset

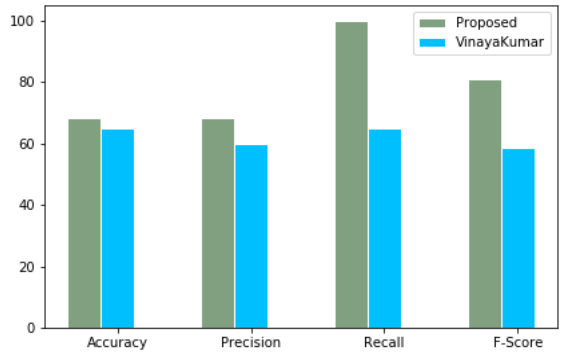

Figure 8. Comparison between the proposed model and VinayaKumar et al.'s work on the UNSW-NB15 dataset

In the following, we will compare the performance of the proposed model with several other models. The models in [69-71] are evaluated using NSL-KDD. As can be seen in Table 5, the accuracy, precision, recall, and F-score of our model are better than these models in this dataset. The model in [37] reports the F-measure on the KDD99 dataset. However, as Table 5 shows, the F-measure of our model is greater than the F-measure of [37]. Finally, the model in [72] reports the precision equal to 93.41 on UNSW-NB15, which is greater than our precision score on this dataset. However, we should mention that our model, unlike [72], has acceptable performance on each of these three datasets.

Table 5. Comparison between the proposed model and other state-of-the-art models on the KDD99, NSL-KDD, and UNSW-NB15 datasets

DataSet / References / Metrics		KDD99	NSL-KDD	UNSW-NB15
Proposed Model	Accuracy	99.02	99.39	68.11
	Precision	99.14	99.49	68.13
	Recall	98.93	99.33	99.98
	F-measure	99.04	99.41	80.97
[69]	Accuracy	-	79.08	-
	Precision	-	87.27	-
	Recall	-	94.60	-
	F-measure	-	91.47	-
[72]	Accuracy	-	-	-
	Precision	-	-	93.41
	Recall	-	-	-
	F-measure	-	-	-
[37]	Accuracy	-	-	-
	Precision	-	-	-
	Recall	-	-	-
	F-measure	94.50	-	-
[70]	Accuracy	-	90.73	-
	Precision	-	86.38	-
	Recall	-	93.17	-
	F-measure	-	89.65	-
[71]	Accuracy	-	86.70	-
	Precision	-	89.36	-
	Recall	-	86.70	-
	F-measure	-	87.22	-

5. Conclusion and future work

One of the challenges of SDN networks is to design an intrusion detection system that can prevent various types of attacks. While several methods have provided IDSs for SDNs, none of them has been able to achieve suitable performance values on different available datasets.

In this study, to improve the security level of the network and prevent various attacks, we proposed an intrusion detection system based on a 12-layer deep neural network. This intrusion detection system was trained and tested on three SDN-specific datasets, namely NSL-KDD, KDD99, and UNSW-NB15. We evaluated our model over these datasets. The accuracy, precision, recall, and F-measure of the model on KDD99 were 99.02, 99.14, 98.93, and 99.04, respectively. These measures on the NSL-KDD dataset were 99.39, 99.49, 99.33, and 99.41, respectively. Furthermore, the model on the UNSW-NB15 dataset reached good results. The results on the three datasets show that our model can reduce the loss function significantly. Moreover, we compared our model with six recent works. The experiment results showed the supremacy of the proposed model over these models.

For future work, the authors plan to work on the following:

- Working on different datasets. While only three widely used datasets are examined in this study, we can work on more than 20 publicly available SDN-specific datasets.
- Implementing other neural network architectures, including CNNs, such as MobileNet, AlexNet, or LeNet. Another possible work is to ensemble the proposed model with other deep architectures or meta-heuristic algorithms such as particle swarm optimization (PSO) algorithm.

6. Reference

[1] Heady, R., Luger, G., Maccabe, A., and Servilla, M., "The architecture of a network level intrusion detection system", *Los Alamos National Lab.*, 1990.

[2] Panda, M., Abraham, A., and Patra, M. R., "A hybrid intelligent approach for network intrusion detection", *Procedia Engineering*, Vol. 30, pp. 1-9, 2012.

[3] Sheikhan, M., and Bostani, H., "A hybrid intrusion detection architecture for internet of things", *in 2016 8th International Symposium on Telecommunications (IST)*, IEEE, pp. 601-606, 2016.

[4] Lin, S.-W., Ying, K.-C., Lee, C.-Y., and Lee, Z.-J., "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection", *Applied Soft Computing*, Vol. 12, No. 10, pp. 3285-3290, 2012.

[5] Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S., "Deep learning approach for intelligent intrusion detection system", *IEEE Access*, Vol. 7, pp. 41525-41550, 2019.

[6] Jiang, F., et al., "Deep learning based multi-channel intelligent attack detection for data security", *IEEE transactions on Sustainable Computing*, Vol. 5, No. 2, pp. 204-212, 2018.

[7] Chalapathy, R., and Chawla, S., "Deep learning for anomaly detection: A survey", *arXiv preprint arXiv*,1901.03407, 2019.

[8] Sultana, N., Chilamkurti, N., Peng, W., and Alhadad, R., "Survey on SDN based network intrusion detection system using machine learning approaches", *Peer-to-Peer Networking and Applications*, Vol. 12, No. 2, pp. 493-501, 2019.

[9] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J., "A survey of deep learning-based network anomaly detection", *Cluster Computing*, Vol. 22, No. 1, pp. 949-961, 2019.

[10] Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., and Faruki, P., "Network intrusion detection for IoT security based on learning techniques", *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 3, pp. 2671-2701, 2019.

[11] Zhong, G., Ling, X., and Wang, L. N., "From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 9, No. 1, pp. e1255, 2019.

[12] Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., and Atkinson, R., "Shallow and deep networks intrusion detection system: A taxonomy and survey", *arXiv preprint arXiv*, 1701.02145, 2017.

[13] Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, SH., Rashidi, S., Hosseinzadeh, M., Rahmani, AM., "Deep learning-based intrusion detection systems: a systematic review", *IEEE Access.* No. 14, Vol. 9, pp. 101574-99, Jul, 2021.

[14] Ahmad Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F., "Network intrusion detection system: A systematic study of machine learning and deep learning approaches", *Transactions on Emerging Telecommunications Technologies*, Vol. 32(1), pp. e4150, Jan, 2021.

[15] Kocher, G., Kumar, G., "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges", *Soft Computing*, Vol. 25(15), pp. 9731-63, Aug, 2021.

[16] Jasim, A. D., "A survey of intrusion detection using deep learning in internet of things", *Iraqi Journal For Computer Science and Mathematics*,Vol. 3(1), pp. 83-93, Jan 30, 2022.

[17] Alsoufi, M. A., Razak, S., Siraj, M. M., Nafea, I., Ghaleb, F. A., Saeed, F., Nasser, M., "Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review", *Applied Sciences*, No. 9, Vol. 11(18), pp. 8383, sep, 2021.

[18] Lee, S. W., Mohammadi, M., Rashidi, S., Rahmani, A. M., Masdari, M., Hosseinzadeh, M., "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review", *Journal of Network and Computer Applications*, Aug No. 1, Vol. 187, pp. 103111, Aug, 2021.

[19] Ahmed, M., Shatabda, S., Islam, A. K., Robin, M., Islam, T., "Intrusion detection system in software-defined networks using machine learning and deep learning techniques–a comprehensive survey", 2021.

[20] Wang, G., Hao, J., Ma, J., and Huang, L., "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering", *Expert systems with applications*, Vol. 37, No. 9, pp. 6225-6232, 2010.

[21] Aburomman, A. A., and Reaz, M. B. I., "A novel SVM-kNN-PSO ensemble method for intrusion detection system", *Applied Soft Computing*, Vol. 38, pp. 360-372,

2016.

[22] Baek, S., Kwon, D., Kim, J., Suh, S. C., Kim, H., and Kim, I., "Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks", *in 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE, pp. 205-210, 2017.

[23] Wang, H., Gu, J., and Wang, S., "An effective intrusion detection framework based on SVM with feature augmentation", *Knowledge-Based Systems*, Vol. 136, pp. 130-139, 2017.

[24] Aljawarneh, S., Aldwairi, M., and Yassein, M. B., "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", *Journal of Computational Science*, Vol. 25, pp. 152-160, 2018.

[25] Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., and Lahza, H. F. M., "Improving performance of intrusion detection system using ensemble methods and feature selection", *in Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1-6, 2018.

[26] He, D., Chen, X., Zou, D., Pei, L., and Jiang, L., "An improved kernel clustering algorithm used in computer network intrusion detection", *in 2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp. 1-5, 2018.

[27] Song, J., Takakura, H., Okabe, Y., and Kwon, Y., "Unsupervised anomaly detection based on clustering and multiple one-class SVM", *IEICE transactions on communications*, Vol. 92, No. 6, pp. 1981-1990, 2009.

[28] Zhu, M., Ye, K., and Xu, C.-Z., "Network anomaly detection and identification based on deep learning methods", *in International Conference on Cloud Computing* Springer, pp. 219-234, 2018.

[29] Li, Z., Qin, Z., Huang, K., Yang, X., and Ye, S., "Intrusion detection using convolutional neural networks for representation learning", *in International conference on neural information processing*, Springer, Vol. ???, pp. 858-866, 2017.

[30] Liu, Y., Liu, S., and Zhao, X., "Intrusion detection algorithm based on convolutional neural network", *DEStech Transactions on Engineering and Technology Research*, No. iceta, 2017.

[31] Potluri, S., Ahmed, S., and Diedrich, C., "Convolutional neural networks for multi-class intrusion detection system", *in International Conference on Mining Intelligence and Knowledge Exploration*, Springer, pp. 225-238, 2018.

[32] Nguyen, S. -N., Nguyen, V.-Q., Choi, J., and Kim, K., "Design and implementation of intrusion detection system using convolutional neural network for DoS detection", *in Proceedings of the 2nd international conference on machine learning and soft computing*, pp. 34-38, 2018.

[33] Yin, C., Zhu, Y., Fei, J., and He, X., "A deep learning approach for intrusion detection using recurrent neural networks", *Ieee Access*, Vol. 5, pp. 21954-21961, 2017.

[34] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M., "Deep recurrent neural network for intrusion detection in sdn-based networks", *in 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, pp. 202-206, 2018.

[35] Vinayakumar, R., Soman, K., and Poornachandran, P., "A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): deep learning for N-IDSs", *International Journal of Digital Crime and Forensics (IJDCF)*, Vol. 11, No. 3, pp. 65-89, 2019.

[36] Chockwanich, N., and Visoottiviseth, V., "Intrusion detection by deep learning with tensorflow", *in 2019 21st International Conference on Advanced Communication Technology (ICACT)*, IEEE, pp. 654-659, 2019.

[37] Zhong, M., Zhou, Y., Chen, G., "Sequential model based intrusion detection system for IoT servers using deep learning methods. Sensors", No. 5, Vol. 21(4), pp. 1113, Feb, 2021.

[38] Ponkarthika, M., and Saraswathy, V., "Network intrusion detection using deep neural networks", *Asian Journal of Science and Technology*, Vol. 2, No. 2, pp. 665-673, 2018.

[39] Vinayakumar, R., Soman, K., and Poornachandran, P., "Applying convolutional neural network for network intrusion detection", *in 2017 International Conference on Advances in Computing*, Communications and Informatics (ICACCI), IEEE, pp. 1222-1228, 2017.

[40] Chawla, A., Lee, B., Fallon, S., and Jacob, P., "Host based intrusion detection system with combined CNN/RNN model", *in Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 149-158, 2018.

[41] Wang, W., et al., "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection", *IEEE access*, Vol. 6, pp. 1792-1806, 2017.

[42] Hsu, C.-M., Hsieh, H.-Y., Prakosa, S. W., Azhari, M. Z., and Leu, J.-S., "Using long-short-term memory based convolutional neural networks for network intrusion detection", *in International wireless internet conference*, Springer, pp. 86-94, 2018.

[43] Lee T. H., Chang, L. H., Syu, C. W., "Deep learning enabled intrusion detection and prevention system over SDN networks", *In2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Jun 7, pp. 1-6, IEEE, 2020.

[44] Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., Kim, S. W., "Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN", IEEE Access. No. 16, Vol. 8, pp. 134695-706, Jul, 2020.

[45] Mohammadi, S., and Namadchian, A., "A new deep learning approach for anomaly base IDS using memetic classifier", *International Journal of Computers Communications & Control*, Vol. 12, No. 5, pp. 677-688, 2017.

[46] Niyaz, Q., "Design and Implementation of a Deep Learning based Intrusion Detection System in Software-Defined Networking Environment", University of Toledo, 2017.

[47] Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q., "A deep learning approach to network intrusion detection", *IEEE transactions on emerging topics in computational*

intelligence, Vol. 2, No. 1, pp. 41-50, 2018.

[48] Farahnakian, F., and Heikkonen, J., "A deep auto-encoder based approach for intrusion detection system", *in 2018 20th International Conference on Advanced Communication Technology (ICACT)*, IEEE, pp. 178-183, 2018.

[49] Papamartzivanos, D., Mármol, F. G., and Kambourakis, G., "Introducing deep learning self-adaptive misuse network intrusion detection systems", IEEE Access, Vol. 7, pp. 13546-13560, 2019.

[50] A. Abusitta, M. Bellaiche, M. Dagenais, and T. Halabi, "A deep learning approach for proactive multi-cloud cooperative intrusion detection system", *Future Generation Computer Systems*, Vol. 98, pp. 308-318, 2019.

[51] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M., "Deep learning approach for network intrusion detection in software defined networking", *in 2016 international conference on wireless networks and mobile communications (WINCOM)*, IEEE, pp. 258-263, 2016.

[52] Roy, S. S., Mallik, A., Gulati, R., Obaidat, M. S., and Krishna, P. V., "A deep learning based artificial neural network approach for intrusion detection", *in International Conference on Mathematics and Computing*, Springer, pp. 44-53, 2017.

[53] Ustebay, S., Turgut, Z., and Aydin, M. A., "Cyber attack detection by using neural network approaches: shallow neural network, deep neural network and autoencoder," *in International conference on computer networks, Springer*,pp. 144-155, 2019.

[54] Vigneswaran, R. K., Vinayakumar, R., Soman, K., and Poornachandran, P., "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security", *in 2018 9th International conference on computing*, communication and networking technologies (ICCCNT), IEEE, pp. 1-6, 2018.

[55] Javeed, D., Gao, T., Khan, M.T., Ahmad, I., "A hybrid deep learning-driven SDN enabled mechanism for secure communication in Internet of Things (IoT)", Sensors, No. 18, Vol. 21(14), pp. 4884, Jul, 2021.

[56] Hande, Y., Muddana, A., "Intrusion detection system using deep learning for software defined networks (SDN)", *In2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) 2019 Nov 27*, pp. 1014-1018, IEEE, 2019.

[57] Duy, P. T., Khoa, N. H., Nguyen, A. G., Pham, V. H., "DIGFuPAS: Deceive IDS with GAN and Function-Preserving on Adversarial Samples in SDN-enabled networks", *Computers & Security*, No. 1, Vol. 109, pp. 102367, Oct, 2021.

[58] Boukria, S., Guerroumi, M., "Intrusion detection system for SDN network using deep learning approach", *In2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, Dec 15, Vol. 1, pp. 1-6, IEEE, 2019.

[59] Seyedkolaei, A. A., Seno, S. A. H., Moradi, A., and Budiarto, R., "Cost-Effective Survivable Controller Placement in Software-Defined Networks", IEEE Access, Vol. 9, pp. 129130-129140, 2021.

[60] Prajapati, A., Sakadasariya, A., and Patel, J., "Software defined network: Future of networking", *in 2018 2nd International Conference on Inventive Systems and Control (ICISC)*, IEEE, pp. 1351-1354, 2018.

[61] Khairi, M. H., Ariffin, S. H., Latiff, N. A., Abdullah, A., and Hassan, M., "A review of anomaly detection techniques and distributed denial of service (DDoS) on software defined network (SDN)", *Engineering, Technology & Applied Science Research*, Vol. 8, No. 2, pp. 2724-2730, 2018.

[62] Shaghaghi, A., Kaafar, M. A., Buyya, R., and Jha, S., "Software-defined network (SDN) data plane security: issues, solutions, and future directions", *Handbook of Computer Networks and Cyber Security*, pp. 341-387, 2020.

[63] Ludwig, S. A., "Intrusion detection of multiple attack classes using a deep neural net ensemble", *in 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, pp. 1-7, 2017.

[64] Faker, O., and Dogdu, E., "Intrusion detection using big data and deep learning techniques", *in Proceedings of the 2019 ACM Southeast Conference*, pp. 86-93, 2019.

[65] Rawat, S., Srinivasan, A., Ravi, V., Ghosh, U., "Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network", *Internet Technology Letters*, Jan, Vol. 5(1), pp. e232, 2022.

[66] Powers, D. M., "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation", *arXiv preprint arXiv*, 2010.16061, 2020.

[67] Revathi, S., and Malathi, A., "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2, No. 12, pp. 1848-1853, 2013.

[68] Abadi, M., et al., "Tensorflow: A system for large-scale machine learning", *in 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265-283, 2016.

[69] Saritha Reddy, A., Ramasubba Reddy, B., Suresh Babu, A., "An Improved Intrusion Detection System for SDN using Multi-Stage Optimized Deep Forest Classifier", *International Journal of Computer Science & Network Security*, Vol. 22(4), pp. 374-86, 2022.

[70] Fu, Y., Du, Y., Cao, Z., Li, Q., Xiang, W. A., "Deep Learning Model for Network Intrusion Detection with Imbalanced Data", *Electronics*, Mar 14, Vol. 11(6), pp. 898, 2022.

[71] Imrana, Y., Xiang, Y., Ali, L., Abdul-Rauf, Z., "A bidirectional LSTM deep learning approach for intrusion detection", *Expert Systems with Applications*, Dec 15, Vol. 185, pp. 115524, 2021.

[72] Toldinas, J., Venčkauskas, A., Damaševičius, R., Grigaliūnas, Š., Morkevičius, N., Baranauskas, E., "A novel approach for network intrusion detection using multistage deep learning image recognition", *Electronics*, Aug 1, Vol. 10(15), pp. 1854, 2021.