TL-NID: Deep Neural Network with Transfer Learning for Network Intrusion Detection

Mohammad Masum
Analytics and Data Science Institute
Kennesaw State University
Kennesaw, USA
mmasum@students.kennesaw.edu

Hossain Shahriar
Department of Information Technology
Kennesaw State University
Marietta, USA
hshahria@kennesaw.edu

2020 15th International Conference for Internet Technology and Secured Transactions (ICITST) | 978-1-913572-21-1/20/$31.00 ©2020010.23919/ICITST51030.2020.9351317

Abstract*- Network intrusion detection systems (NIDSs) play an essential role in the defense of computer networks by identifying a computer networks' unauthorized access and investigating potential security breaches. Traditional NIDSs encounters difficulties to combat newly created sophisticated and unpredictable security attacks. Hence, there is an increasing need for automatic intrusion detection solution that can detect malicious activities more accurately and prevent high false alarm rates (FPR). In this paper, we propose a novel network intrusion detection framework using a deep neural network based on the pretrained VGG-16 architecture. The framework, TL-NID (Transfer Learning for Network Intrusion Detection), is a two-step process where features are extracted in the first step, using VGG-16 pre-trained on ImageNet dataset and in the 2nd step a deep neural network is applied to the extracted features for classification. We applied TL-NID on NSL-KDD, a benchmark dataset for network intrusion, to evaluate the performance of the proposed framework. The experimental results show that our proposed method can effectively learn from the NSL-KDD dataset with producing a realistic performance in terms of accuracy, precision, recall, and false alarm. This study also aims to motivate security researchers to exploit different state-of-the-art pre-trained models for network intrusion detection problems through valuable knowledge transfer.

Keywords-Transfer learning, Pre-trained model, VGG-16, Deep neural network, Network intrusion detection

I. Introduction

Network intrusion detection systems (NIDSs) play an essential role in the defense of computer networks by identifying a computer networks' unauthorized access and investigating potential security breaches. Traditional NIDSs encounters difficulties to combat newly created sophisticated and unpredictable security attacks. The number of security threats on traffic data are increasing exponentially and the newly created attacks has become more sophisticated and variants. Hence, traditional intrusion detection techniques such as signature-based detection, heuristic detection or behavior-based detection are not adequate to combat malicious activities [1].

Machine learning (ML) algorithms have been showing promising results in classifying network intrusions. The algorithms can overcome the limitations of traditional detection methods and provide a rewarding accuracy score. Traditional machine learning approaches like Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), and Decision Tree (DT) were previously proposed for intrusion detection. However, deep convolutional neural networks (DCNN), an advanced ML technique, is gained popularity and widely used for many applications like computer vision, speech recognition, and natural language processing. Extracting automated features, capability of highly non-linear systems, and flexibility in architecture design are the highlight of the DCNN. DCNN essentially follows two distinct approaches to train: 1. training from scratch-training the model of randomly initialized weights, and 2. transfer learning-pre-training a model on a related task and then optimizing the model for target task [5]. Transfer learning from a pre-trained model is a popular approach, yet it is unexplored for network intrusion detection problem. A conventional DCNN, learning from scratch, that is developed on a specific train data demonstrate quite different performance on different test datasets. An example was demonstrated based on KDD-NSL dataset where a DCNN, was trained on NSL-KDD training data, poorly performed on KDDTest-21 dataset since a number of attacks in KDDTest-21are not covered in the training set [6]. Moreover, training time increases exponentially with the increasing/deepening of architecture of DCNN. In general, it could take hours/days to train a 3–5 layers DCNN with a large scale dataset. Consequently, deploying VGG from scratch on a large scale dataset is a tiresome and computationally expensive task due to the depth and number of fully connected layers/nodes in the models' architecture.

Another challenge is that building VGG from scratch requires considerably large memory space and bandwidth since the size of ImageNet trained VGG-16 weights is *528* MB. However, instead of building a VGG from scratch, we can perform transfer learning i.e. —utilizing the knowledge like weights of the previously trained (e.g. pre-trained VGG) models' to solve a similar kind of problem.

A limited number of research studies have investigated transfer learning from pre-trained models' in the field of network intrusion detection. In this paper, we present an approach for intrusion detection leveraging DCNN model based on the VGG-16 architecture with pre-trained convolutional layers. We converted the intrusion samples into gray-scale image at first and later transformed into RGB image format to feed the samples to the deep neural network. Our proposed framework transfers convolutional layers' parameters of VGG-16 pre-trained on ImageNet dataset. Two additional fully connected layers are added to the framework. The fully connected layers including the final sigmoid layer are designed to adopt our problem of network intrusion. In this study, we also applied conventional machine learning algorithms like SVM, DT, RF, and LR to compare results with our proposed TL-NID framework. The experimental results on NSL-KDD datasets show that our TL-NID can effectively learn from the train data and provide an accuracy of 89.3% for test data that outperform other reference work on this dataset suggesting that detecting network intrusion using transfer learning technique is promising.

The rest of the paper is organized as follows: In Section II, we introduce the related work of network intrusion detection. Section III describes the methodology of our proposed framework TL-NID along with the other three classifiers that are implemented in this paper. The experimental setting and results are explained in Section IV. Finally, Section V concludes the paper.

II. Related Work

Addressing the constraints of the traditional methods, researchers have proposed both conventional machine learning (ML) algorithms and deep learning for network intrusion detection. A least square SVM applied to representative samples collected from pre-determined arbitrary subgroups to detect network traffic data [13]. The method was applied to KDDVUP'99 data and achieved a reasonable performance in terms of accuracy. Different ML methods like RF, DT, Gaussian naïve bayes, and SVM were applied to NSL-KDD dataset [4, 15, 16].

Deep neural networks are successfully applied for classifying malware activities and network intrusion detection [16, 17]. A 2- layer convolutional neural network (CNN) was applied to NSL-KDD data by converting the raw data into image format [7]. The authors reported accuracy of 99.46%, 79%, and 60% on KDDTrain+, KDDTest+, and KDDTest-21 datasets, respectively. Another CNN approach based on Lenet-5 was applied to the KDDCUP'99 dataset that only evaluated their proposed method on the KDDCUP'99 training data and reported 99.9% accuracy after experimenting with different epoch sizes and learning rates [8]. A multilevel classifier fusion approach was proposed for network intrusion detection, containing two layers wherein the upper layer, an unsupervised feature extraction method was applied using a non-symmetric deep autoencoder, and random forest classifier was then applied with the extracted features [10]. The method experimented with two different datasets including the KDDCUP'99 and the NSL-KDD datasets. For a 5-class classification, the proposed method achieved 97.85% accuracy with the KDDCUP'99 data, while 85.42% accuracy was obtained for the NSL-KDD dataset. A framework based on CNN was proposed experimenting with different number of hidden layers for both binary and multi-class classification of network traffic data [11]. The framework applied to 6 different datasets including KDDCUP'99 and NSL-KDD datasets. The experimental results in the study show that the optimal framework, consists of 5 hidden layers, achieved reasonably low accuracy for both binary and multi-class classification. The framework provided 92.7% and 78.9% on the KDDCUP'99 and NSL-KDD datasets for binary classification, respectively. It produced 92.5% and 78.5% accuracy on the KDDCUP'99 and NSL-KDD datasets, respectively, with respect to multi-class classification.

Limited numbers of published works are available that are based on transfer learning for network intrusion detection. A transfer learning-based method TL-ConvNet was introduced that learns from a base dataset and transfers the learned knowledge to the learning of the target dataset [9]. The TL-ConvNet used UNSW-NB15, and NSL-KDD as based dataset, and target dataset, respectively. Experimental results in this study showed significant performance improves by using transfer learning instead of conventional CNN. TL-ConvNet achieved 87.30%, and 81.9% accuracy on KDDTest+, and KDDTest-21 datasets, respectively.

Our proposed framework TL-NID, a deep learning-based framework, was optimized with different parameters and hyperparameters and experimented with two different test datasets from a real-world network intrusion dataset. The experimental results show the effectiveness of our proposed method.

III. Methods

Transfer learning is one of the state-of-the-art techniques in machine learning that has been widely used in image classification. VGG is a convolutional neural network with a specific architecture that was proposed by a group of researchers (visual geometry group) from the University of Oxford [12]. The VGG model was trained on the ImageNet dataset, a major computer vision benchmark dataset that includes more than 14 million images belonging to 1000 classes, for object localization (detecting objects within an image coming from 200 classes) and image classification tasks (1000-class classification). The VGG has two different architecture: VGG-16 that contains 16 layers and VGG-19 that contains 19 layers. In this paper, we applied VGG-16 that mainly contains three different parts: convolution, pooling, and fully connected layers— it starts with two convolution layers followed by pooling, then another two convolutions followed by pooling, after that repetition of three convolutions followed by pooling and then finally three fully connected layers. Table II displays the summary of the VGG-16 model where convolutional, max pooling, and fully connected layers are mentioned by Conv. MP, and FC, respectively. The model weights of the VGG-16 are available on different platforms like Keras and can be used for further analysis — developing models and applications.

Table I: Summary of VGG-16 architecture

Layer	Kernel size	Output Size	Acti-vation	Param-eters
input	-	$224 \times 224 \times 3$	-	0
Conv	3×3	$224 \times 224 \times 64$	relu	1792
Conv	3×3	$224 \times 224 \times 64$	relu	36928
MP	2×2	$112 \times 112 \times 64$	relu	0
Conv	3×3	$112 \times 112 \times 128$	relu	73856
Conv	3×3	$112 \times 112 \times 128$	relu	14784
MP	2×2	$56 \times 56 \times 128$	relu	0
Conv	3×3	$56 \times 56 \times 256$	relu	295168
Conv	3×3	$56 \times 56 \times 256$	relu	590080
Conv	3×3	$56 \times 56 \times 256$	relu	590080
MP	2×2	$28 \times 28 \times 256$	relu	0
Conv	3×3	$28 \times 28 \times 512$	relu	1180160
Conv	3×3	$28 \times 28 \times 512$	relu	2359808
Conv	3×3	$28 \times 28 \times 512$	relu	2359808
MP	2×2	$14 \times 14 \times 512$	relu	0
Conv	3×3	$14 \times 14 \times 512$	relu	2359508
Conv	3×3	$14 \times 14 \times 512$	relu	2359508
Conv	3×3	$14 \times 14 \times 512$	relu	2359508
MP	2×2	$7 \times 7 \times 512$	relu	0
FC	-	4096	relu	102764544
FC	-	4096	relu	16791312
Out-put	-	1000	Soft-max	

We leveraged pre-trained VGG-16 weights in the process of developing the TL-NID framework. Fig. 1 illustrates architecture of the TL-NID framework. At the initial stage, the framework prepares the given network intrusion data. Each of the intrusion samples is then converted into its corresponding grayscale images which are later transformed into RGB color images to feed to the pre-trained VGG-16. First 14 layers of the VGG-16 model was utilized for weights transfer as transfer learning. The TL-NID utilized the transferred weights to extract features from the input images. The extracted features are then trained with a DNN based on the VGG-16 architecture. The DNN consists of three layers: input layer, two hidden layers and output layer. The two hidden layers are fully connected layers that contain 64 and 8 number of nodes, respectively. The final layer of the DNN is the sigmoid layer as we considered binary classification for network intrusion detection.

IV. Experiment and Results

A. Dataset specification

NSL-KDD dataset is an updated version of the KDDCUP'99 dataset which was widely used for network intrusion detection problems [12]. Many redundant records in the KDDCUP'99 dataset generates biases towards the more frequent records. Therefore, the NSL-KDD dataset was proposed by addressing the drawbacks and since then it has been considered the benchmark dataset for the NID problem [14]. KDDTrain+, KDDTrain+_20 percent (a 20% subset of the KDDTrain+), KDDTest+, and KDDTest-21(a subset of KDDTest+) are the four datasets that are

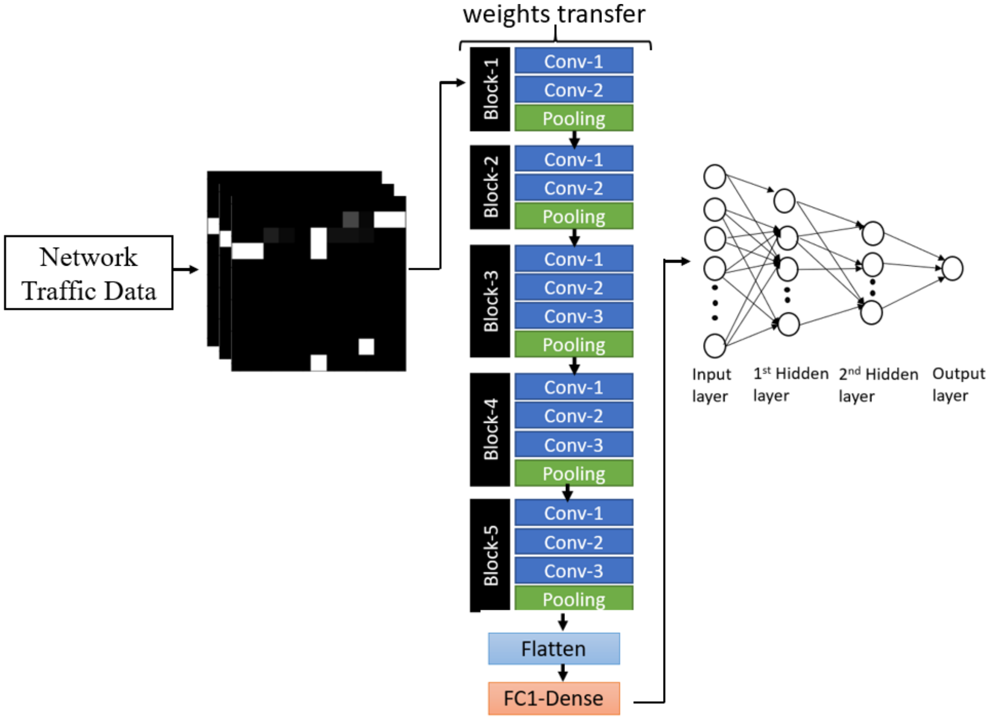

Figure 1: Architecture of TL-NID framework

included in the NSL-KDD. In this paper, we used 10 percent of KDDTrain+, KDDTest+, and KDDTest-21 datasets to develop and validate the TL-NID framework.

Each of the network connection records in the NSL-KDD dataset consists of 41 features, providing information about the encounter with eh traffic input by IDS, along with a label attribute indicating the connection status (normal or attack). Four different types of attacks are included in the dataset: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). The features in the dataset can be divided into groups: intrinsic, content, host-, and time-based features. The intrinsic features contain basic information about the packet that is derived from the header of the connection, while the content features comprise information related to the original packets. Time-based features contain information about the traffic input over a two-second window. Host-based features, on the other hand, contain information over a series of connections, designed to access attacks that are longer than a two second window. Table I displays distributions of the three datasets for binary classification where normal, and attack represent good, and bad network connections, respectively.

Table II: Distributions of the NSL-KDD dataset

Type	KDDTrain+	KDDTest+	KDDTest-21
Normal	6641	973	199
Attack	5956	1281	986

B. Data Preprocessing

We applied the NSL-KDD dataset that contains network connection features to evaluate the DCNN framework. Thus, data preprocessing is required to convert the raw data into image format to feed it to the DCNN. The categorical data in NSL-KDD should be mapped to numeric data at first and then the overall data should be normalized. protocol type, flag, and service are the three categorical features in the NSL-KDD that are converted into numeric data using one hot encoder technique. We applied the min-max normalization technique to NSL-KDD to scale the original data to a fixed range of 0 and 1. The normalization ensures consistency of the data distribution and avoiding the exploding gradients problem in the training phase [3]. Equation (1) represents the min-max normalization formula, where X_{scaled}, and X is the normalized, and original value, respectively. $\min(X)$, and $\max(X)$ are the minimum and maximum values of the data.

$$X_{scaled} = \frac{X - \min(x)}{\max(x) - \min(x)} \quad \textbf{(1)}$$

The number of features has been expanded from 41 to 121 after the preprocessing step. To feed to the DCNN the 1-dimensional data is translated into 2-d array of size 11×11 or grayscale image at first. Fig shows some network connection samples as grayscale images where images in the 1st row, and 2nd row are for normal, and attack connections, respectively. In the next stage, the grayscale images were converted into 3-d RGB images and resized to the format of $224 \times 224 \times 3$ format as required by the pre-trained VGG-16 input shape. Since the grayscale images have only one-channel, in the process of conversion, a channel augmentation is performed by duplicating the grayscale images into three channels RGB images.

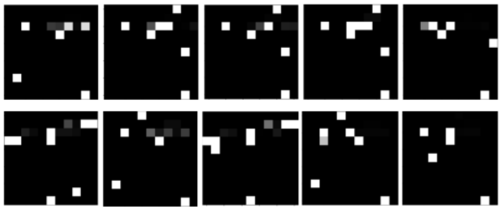

Figure 2: Grayscale images of network connection samples

C. Model evaluation metrics

To evaluate the models' performance, we considered accuracy, precision, recall, false alarm, and F-score metrics. These metrics use properties from confusion matrix such as true positive (TP), false positive (FP), false negative (FN), and true negative (TN). TP is number of attacks that are correctly classified as attacks, while FN is the attacks that incorrectly classified. The number of incorrectly classified normal data is FN, and TN is correctly classified normal data. Equation 2,3,4,5, and 6 are the mathematical definition of the performance metrics accuracy, precision, recall, false alarm, and F-score, respectively.

$$Accuracy = \frac{TP + TN}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$\mathrm{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{False Alarm} = \frac{FP}{FP + TN} \quad (5)$$

$$\mathrm{F-score} = \frac{2 * Precision * recall}{Precision + recall} \quad (6)$$

D. Experimental design & Results

We evaluated our model performance by comparing it with the performance of LR, SVM, and DT methods. Trained data was used to train each of the models we experimented with while test data was used for evaluating the performance of the models. The SVM, LR, and DT classifiers were applied to the dataset for comparing results with our proposed framework. The algorithms were implemented using Python scikit-learn library with available hyperparameter options. 'rbf' (Radial Basis Kernel) were chosen for SVM, 'gini' index was chosen for DT, and L2 penalty was chosen for LR classifier.

Our proposed framework is based on pre-trained VGG-16 model and a DNN. The DNN is trained with the features that are extracted using pre-trained VGG-16 model. The DNN consists of four layers: input layer, two hidden layers, and output layer. We used 'ReLu' activation function in the hidden layer and 'sigmoid' function in the output layer. 'Adam' and 'binary cross-entropy' were used for optimizer and loss function respectively. We implemented an early stopping method to stop training once the model performance stops improving on the test data. We selected validation loss to be monitored for early stopping and set minimum delta to $1e-4$ (checks minimum change in the monitored quantity to qualify as an improvement) and patience to 10 (checks number of epochs that produced the monitored quantity with no improvement after which training will be stopped). Mini-batch gradient descent was considered and a batch size of 64 was chosen to train the model. The initial learning rate was set to 0.001 with a decay of $1e-5$ in every epoch The L^2 regularization technique was applied to the output of the hidden layer to prevent the network from overfitting and the regularization parameter 'lambda' was set to 0.001. All the parameters and hyperparameters used in the model were optimized by grid search.

E. Results

We applied five different methods on the NSL-KDD dataset including SVM, DT, LR, RF, and our proposed TL-NID. Table III represents the results of the methods with two different test datasets KDDTest+, and KDDTest-21. The experimental results show that TL-NID achieved highest accuracy 89.30%, and

Table III: Comparison of results

	KDDTest+					KDDTest-21				
	Accuracy (%)	Precision (%)	Recall (%)	False Alarm (%)	F-score (%)	Accuracy (%)	Precision (%)	Recall (%)	False Alarm (%)	F-score (%)
SVM	65.08	68.93	71.32	43.33	70.11	55.44	**97.63**	46.75	**5.1**	63.23
DT	80.65	**95.44**	69.62	**4.41**	80.51	68.10	95.12	64.36	14.95	76.78
LR	80.25	91.49	72.33	9.06	80.79	57.72	89.16	55.09	30.37	68.10
RF	73.91	93.36	58.73	5.60	72.10	65.51	81.39	74.35	77.10	77.71
TL-NID	**89.30**	93.55	**87.19**	7.9%	**90.26**	**70.97**	82.82	**82.15**	8.24	**82.48**

F-score 90.26% for network detection for KDDTest+ while the decision tree achieved second maximum accuracy 80.65. Highest precision rate, and lowest false alarm rate were achieved by DT. Highest accuracy (70.9%), recall (82.15%), and F-score (82.48) were achieved by the TL-NID, whereas maximum precision was obtained by SVM for KDDTest-21 dataset.

V. Conclusion

Computer network attacks are increasingly posing a serious security threat. It is essential to develop an automatic network intrusion detection solution to reduce the risks of malicious activities. Existing traditional methods and machine learning algorithms are not sufficiently effective for network intrusion detection problems. In this paper, we proposed a transfer learning-based framework, TL-NID, for network intrusion detection that first extracts higher level features leveraging VGG-16 pre-trained on ImageNet dataset by transferring wights. We trained the TL-NID on KDDTrain+ dataset and evaluated performance on KDDTest+, and KDDTest-21 datasets, respectively. We also evaluated TL-NID performance by comparing it with the performance of LR, SVM, and DT algorithms. The experimental results show effectiveness of TL-NID with producing rewarding accuracy, precision, recall, false alarm, and f1-score.

References

[1] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-5). IEEE.

[2] Mujumdar, A., Masiwal, G., & Meshram, D. B. (2013). Analysis of signature-based and behavior-based anti-malware approaches. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, *2*(6).

[3] Kumar, V., & Sangwan, O. P. (2012). Signature based intrusion detection system using SNORT. International Journal of Computer Applications & Information Technology, 1(3), 35-41.

[4] Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications surveys & tutorials, 18(2), 1153-1176.

[5] Xie, Y., & Richmond, D. (2018). Pre-training on grayscale ImageNet improves medical image classification. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 0-0).

[6] Wu, P., Guo, H., & Buckland, R. (2019, March). A transfer learning approach for network intrusion detection. In 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA) (pp. 281-285). IEEE.

[7] Wu, K., Chen, Z., & Li, W. (2018). A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access, 6, 50850-50859.

[8] Liu, P. (2019, February). An intrusion detection system based on convolutional neural network. In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering (pp. 62-67).

[9] Wu, P., Guo, H., & Buckland, R. (2019, March). A transfer learning approach for network intrusion detection. In 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA) (pp. 281-285). IEEE.

[10] Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE transactions on emerging topics in computational intelligence, 2(1), 41-50.

[11] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.

[12] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[13] Kabir, E., Hu, J., Wang, H., & Zhuo, G. (2018). A novel statistical technique for intrusion detection systems. Future Generation Computer Systems, 79, 303-318.

[14] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). IEEE.

[15] Belavagi, M. C., & Muniyal, B. (2016). Performance evaluation of supervised machine learning algorithms for intrusion detection. Procedia Computer Science, 89(2016), 117-123.

[16] Masum, M., & Shahriar, H. (2019, December). Droid-NNet: Deep Learning Neural Network for Android Malware Detection. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 5789-5793). IEEE.

[17] Rosay, A., Carlier, F., & Leroux, P. (2020, May). Feed-forward neural network for Network Intrusion Detection. In 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring) (pp. 1-6). IEEE.