A Transfer Learning with Deep Neural Network Approach for Network Intrusion Detection

Mohammad Masum[1], Hossain Shahriar, Hisham M. Haddad[3]
School of Analytics and Data Science[1], Department of Information Technology[2], Department of Computer Science[3]
Kennesaw State University, USA

Abstract

Traditional Network Intrusion Detection Systems (NIDS) encounter difficulties due to the exponential growth of network traffic data and modern attacks' requirements. This paper presents a novel network intrusion classification framework using transfer learning from the VGG-16 pre-trained model. The framework extracts feature leveraging pre-trained weights trained on the ImageNet dataset in the initial step, and finally, applies a deep neural network to the extracted features for intrusion classification. We applied the presented framework on NSL-KDD, a benchmark dataset for network intrusion, to evaluate the proposed framework's performance. We also implemented other pre-trained models such as VGG19, MobileNet, ResNet-50, and Inception V3 to evaluate and compare performance. This paper also displays both binary classification (normal vs. attack) and multi-class classification (classifying types of attacks) for network intrusion detection. The experimental results show that feature extraction using VGG-16 outperforms other pre-trained models producing better accuracy, precision, recall, and false alarm rates.

Keywords: *Transfer learning, Pre-trained model, Feature extraction, VGG-16, Deep neural network, Network intrusion detection, Inception, MobileNet, ResNet*

1. Introduction

Network Intrusion Detection Systems (NIDS) play an essential role in computer networks' defense by identifying computer networks' unauthorized access and investigating potential security breaches. Traditional NIDS encounters difficulties to combat newly created sophisticated and unpredictable security attacks. The number of security threats on traffic data increases exponentially, and the newly created attacks have 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, have gained popularity and are widely used for many applications like computer vision, speech recognition, and natural language processing [12]. Extracting automated features, the capability of highly non-linear systems, and flexibility in architecture design highlight 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 the target task [5].

Transfer learning is an advanced machine learning approach, where knowledge (e.g., weights) of previously trained (pre-trained) models' is transferred for learning improvement in a new task. Over the years, transfer learning from pre-trained models has been applied successfully for image classification to avoid the computational cost and ability to produce rewarding results. AlexNet, VGG-16, VGG-19, GoogLeNet or Inception, MobileNet, Inception, and ResNet are the most widely used pre-trained models that were mainly introduced for image classification challenges ILSVRC− ImageNet large scale visual recognition challenge [19]. These models leverage more in-depth network architecture training on the ImageNet dataset to produce better classification performance. AlexNet, an 8-layers deep architecture, was introduced in 2012 and resulted in a top-5 classification error of 16.4% on the ImageNet challenge. VGG-based architecture (VGG-16 and VGG-19 with 16- and 19-layers architecture, respectively) were placed in top-5 in 2014 with a 7.3% classification error. GoogLeNet with 22 layers and ResNet with 152 layers produced a classification error of 6.7% and 3.57%, respectively, and were placed in top-5 performed models. This paper studies advanced pre-trained models VGG-16, VGG-19, Inception, ResNet, and MobileNet for future extraction for intrusion classification.

Transfer learning from a pre-trained model is a popular approach, yet it is unexplored for network intrusion detection. A conventional DCNN, learning from scratch, developed on a specific train data demonstrates quite different performance on different test datasets. An example was demonstrated based on the KDD-NSL dataset, where a DCNN was trained on NSL-KDD training data, poorly performed on the KDDTest-21 dataset since several attacks in KDDTest-21are not covered in the training set [6]. Moreover, training time increases exponentially with the increasing/deepening of the architecture of DCNN. In general, it could take hours/days to train a 3–5 layers DCNN with a large scale dataset [6]. Consequently, deploying VGG like deep network architecture models 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 developing this kind of deep network from scratch requires considerably large memory space and bandwidth. For instance, the size of ImageNet trained VGG-16 weights is *528* MB. However, instead of building a deep network 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. This paper presents an approach for intrusion detection leveraging the DCNN model based on the VGG-16 architecture with pre-trained convolutional layers. We converted the intrusion samples into a gray-scale image and later transformed it into an 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 the ImageNet dataset. Two additional fully connected layers are added to the framework. The fully connected layers, including the final sigmoid layer, are designed to adapt to network intrusion. This study also applied conventional machine learning algorithms like SVM, DT, RF, and LR to compare our proposed framework results. The experimental results on NSL-KDD datasets show that the proposed approach effectively learns from the train data and provides an accuracy of 89.3% for test data that outperform other reference work on this dataset detecting network intrusion using transfer learning technique. In this paper, we applied the VGG-16 pre-trained model. In summary, the contribution of the paper is as follows:

- We proposed a novel network intrusion detection framework by using transfer learning from the pre-trained VGG-16 model.
- We applied the proposed framework considering both binary and multi-class classification problems.
- We evaluated the proposed framework and compared it with other advanced pre-trained models using a benchmark network intrusion dataset.

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

2. 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 was 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].

Deep neural networks are successfully applied for classifying malware activities and network intrusion detection [16]. 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. The upper layer, an unsupervised feature extraction method, was applied using a non-symmetric deep autoencoder 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 a different number of hidden layers for both binary and multi-class classification of network traffic data [11].

The framework was applied to 6 different datasets, including KDDCUP'99 and NSL-KDD datasets. The study's experimental results show that the optimal framework consists of 5 hidden layers and achieved reasonably low accuracy for 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, concerning 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 to learn from a base dataset and transfer the learned knowledge to the target dataset's learning [9]. The TL-ConvNet used UNSW-NB15, and NSL-KDD as a based dataset, and target dataset, respectively. Experimental results in this study showed significant performance improvements by using transfer learning instead of conventional CNN. TL-ConvNet achieved 87.30% and 81.9% accuracy on KDDTest+, and KDDTest-21 datasets, respectively. Transfer learning from ResNet-50 pre-trained model was applied for malicious software classification using the Malimg dataset, where samples are presented as byteplot grayscale images [18]. The presented method froze all layers in the ResNet-50 structure in the training phase, except for the final layers. The final layer included 25 neurons, considering the 25-malware class classification problem. Through experiments, the papers showed that feature extractor learned by ResNet-50 outperforms hand-crafted GIST feature extractor process. A transfer learning approach based on GoogLeNet pre-trained model was applied to NSL-KDD dataset and produced 77% and 81% of accuracy for KDDTest+, and KDDTest-21 datasets, respectively [15].

Our presented framework, 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.

3. 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 contains 16 layers and VGG-19 contains 19 layers. In this paper, we applied VGG-16 that mainly contains three different parts: standard 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 1 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.

We leveraged pre-trained VGG-16 weights in the process of developing the proposed framework. Figure 1 illustrates the architecture of the 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. The first 14 layers of the VGG-16 model were utilized for weights transfer as transfer learning. The framework 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: an input layer, two hidden layers, and an output layer. The two hidden layers are fully connected layers that contain 64 and 8 nodes, respectively. The final layer of the DNN is the sigmoid layer, as we considered binary classification for network intrusion detection.

We also applied other advanced pre-trained models— VGG-19, MobileNet, Inception V3, and ResNet, for transfer learning to compare with the presented framework:

VGG-19: VGG-19, an extended version of VGG-19, is a 19-layers deep convolutional neural network architecture. The input shape of the VGG-19 is $(224 \times 224 \times 3)$, which is similar to the VGG-16 design. In designing VGG-19 architecture one extra convolutional layer was included in the 3rd, 4th, and 5th blocks of VGG-16, which is the significant difference between the two VGG models [20].

Table 1. 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	

MobileNet: MobileNet, a lightweight 28-layers deep convolutional neural network, is primarily built on depthwise separable convolutions to reduce the computation in the first few layers [2]. The depthwise separable convolution consists of depthwise and pointwise convolutions, where depthwise convolutions utilize a single filter for each input channel and pointwise convolutions then apply a 1×1 convolution to linearly combine the outputs of the depthwise layers. Using depthwise separable convolutions instead of standard convolutions is the primary different between MobileNet and VGG-based architectures. All layers are accompanied by batch normalization and a nonlinear activation function-ReLU except for the final fully connected layer, which has no nonlinearity and feeds to the softmax layer for classification.

Inception V1: Inception V1 is a 27 layers deep convolutional neural network that introduced the inception layer. The inception layer combines (1×1), (3×3), and (5×5) convolution layers that are concatenated into a single output vector forming the next stage's input [17]. The architecture starts with two blocks where each block contains one convolution and one max-pooling layer. After that, two inception layers followed by a max-pooling and then five consecutive inception layers with a max-pooling layer are added. Subsequently, two inception layers, one average pooling, one dropout, one linear, and lastly a softmax layer appended for classification.

ResNet-50: ResNet, short for the residual network, is a deep (152 layers) architecture for image classification and winner in the 2015 ImageNet challenge. ResNet-50 is a version of the residual network of 50-layers consists of 5 stages, where each stage includes one convolutional block and one identity block. Each convolutional block contains three standard convolutional layers, and each identity block contains three convolutional layers. Following the five stages, an average pooling layer for subsampling, a flatten layer, a fully connected layer, and a softmax layer were added for classification.

4. Experiment and Results

4.1. 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 generate biases towards the more frequent records. Therefore, the NSL-KDD dataset was proposed by addressing the drawbacks. Since then, it has been considered the benchmark dataset for the network intrusion detection 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 included in the NSL-KDD. In this paper, we used 10 percent of KDDTrain+, KDDTest+, and KDDTest-21 datasets to develop and validate the proposed framework.

Each of the network connection records in the NSL-KDD dataset consists of 41 features, providing information about the encounter with ID traffic input and a label attribute indicating the connection status

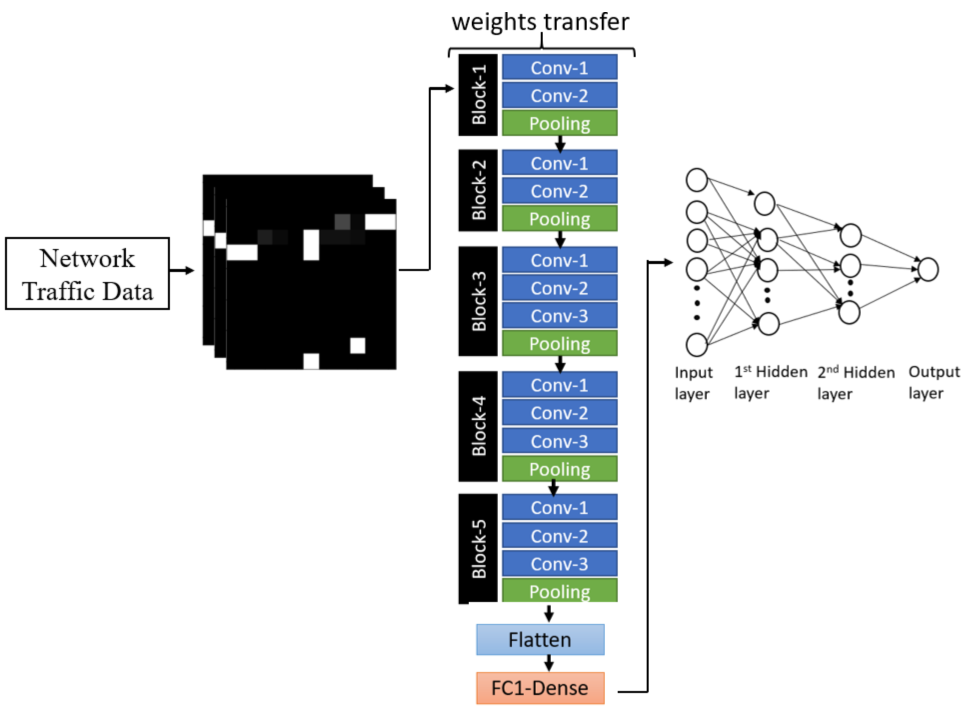

Figure 1. Architecture of proposed framework

Table 2. Sub-classes of different attacks

Types of Attacks	Sub-classes
DoS	apache2, back, land, Neptune, mailbomb, pod, processtable, smurf, teardrop, udstorm, and worm
Probe	ipsweap, mscan, nmap, portsweep, saint, and satan
U2R	buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, and xterm
R2L	ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, spy, snmguess, warezclient, warezmaster, xlovk, and xsnoop

(normal or attack). Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L) are four different types of attacks, where each attack includes several subclasses- DoS, Probe, U2R, and R2L contain 11, 6, 7, and 15 subclasses, respectively.

Table 2 exhibits subclasses of different attacks. The features in the dataset are divided into groups: intrinsic, content, host-and time-based features. The intrinsic features contain necessary information about the packet derived from the header of the connection, while the content features comprise information related to the original packets. Time-based features include information about the traffic input over a two-second window. On the other hand, host-based components contain information over a series of connections, designed to access attacks that are longer than a two-second window. Table 3 displays the distribution of normal and attacks data. Moreover, table 3 shows the distribution of data for different types of attacks: DoS, Probe, U2R, and R2L for the three datasets. We used 10% of each of the datasets to avoid computational complexity.

Table 3. Distributions of the NSL-KDD data

Type	KDDTrain+	KDDTest+	KDDTest-21
Normal	6641	973	199
Attack	5956	1281	986
DoS	4627	754	322
Probe	1233	283	256
U2R	87	237	199
R2L	9	7	5
Total Attacks	5956	1281	986

4.2. 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 the 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.

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

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 a 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.

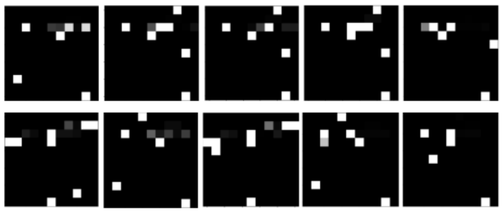

Figure 2. Grayscale images of network connection samples

4.3. Model evaluation metrics

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

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

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

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

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

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

4.4. Experimental design and Results

We evaluated our model performance by comparing it with the performance of traditional machine learning algorithms such as 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 a pre-trained VGG-16 model and a DNN. The DNN is trained with the features that are extracted using the pre-trained VGG-16 model. The DNN consists of four layers: an input layer, two hidden layers, and an 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.

We also applied other pre-trained models like VGG-19, Inception, MobileNet, and ResNet-50 to extract features from the converted RGB images and employ a DNN on the extracted features. We dropped the final classification layer, i.e., the softmax layer, from the mentioned pre-trained models' architecture, added a sigmoid layer of one neuron in binary classification, and added a softmax layer of five neurons for multi-class classification. We followed an identical structure for the DNN to maintain consistency of performance.

4.5. Results

We applied nine different methods for the binary classification problem, on the NSL-KDD dataset including traditional machine learning methods−SVM, DT, LR, and RF; transfer learning approaches− VGG-16+DNN, VGG-19+DNN, Inception V3+DNN, MobilNet+DNN, and ResNet-50+DNN. Table 4 represents the results of the methods with two different test datasets KDDTest+, and KDDTest-21 considering binary classification. The transfer learning-based methods express that the pre-trained model was use for feature extraction and DNN applied for classification. For instance, VGG-16+DNN: VGG-16 was applied for feature extraction from the image data, and the DNN was applied to the extracted features for intrusion classification. The experimental results show that our presented method (VGG-16+DNN) achieved the highest accuracy 89.30%, and F-score 90.26% for network detection for KDDTest+ while the decision tree achieved the second maximum accuracy of 80.65. The highest precision rate and lowest false alarm rate were achieved by DT. The highest accuracy (70.9%), recall (82.15%), and F-score (82.48) were achieved by the presented method, whereas maximum precision was obtained by SVM for the KDDTest-21 dataset. Other feature extractors VGG-19, MobileNet, Inception, and ResNet-50 also provides good performance, close to the VGG-16 methods.

The Table 5 shows the results for each of the classes of multi-class classification of network intrusion. Table 6 represents the results for multi-class classification for both the datasets. The VGG-16 feature extractor outperform others in terms of accuracy. VGG-16 + DNN achieved 78.39% and 52.56% for KDDTest+, and KDDTest-21 datasets, respectively. VGG-19 and MobileNet provides reasonable performance as well.

Table 4. Comparison of results for binary classification

	KDDTest+					KDDTest-21				
Methods	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
VGG-16+ DNN	**89.30**	**93.55**	**87.19**	**7.9%**	**90.26**	**81.77**	**81.03**	**96.49**	**14.57**	**88.09**
VGG-19 + DNN	86.60	83.52	92.16	9.35	87.63	81.43	83.26	93.72	27.63	88.18
Inception V3 + DNN	85.71	78.84	95.19	5.2	86.25	75.61	79.00	90.47	41.20	84.35
MobileNet + DNN	85.71	78.84	95.19	5.2	86.25	79.57	79.81	94.81	21.60	86.67
ResNet50 + DNN	84.29	76.03	95.39	4.83	84.62	83.12	99.89	83.19	100.0	90.78

Table 5. Classification results for each class using VGG-16 + DNN

	KDDTest+			KDDTest-21		
	precision	recall	f1-score	precision	recall	f1-score
Normal	0.81	0.92	0.86	0.57	0.74	0.64
DoS	0.78	0.95	0.86	0.40	0.81	0.54
Probe	0.71	0.62	0.66	0.52	0.55	0.54
U2R	0.00	0.00	0.00	0.00	0.00	0.00
R2L	0.00	0.00	0.00	0.00	0.00	0.00

Table 6. Comparison of results for multi-class classification

	KDDTest+ Accuracy (%)	KDDTest-21 Accuracy (%)
VGG-16 + DNN	**78.39**	**52.56**
VGG-19 + DNN	74.62	38.90
Inception V3 + DNN	68.90	32.91
MobileNet + DNN	72.98	50.98
ResNet50 + DNN	69.38	41.85

5. Conclusions

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 for network intrusion detection that first extracts higher level features leveraging VGG-16 pre-trained on ImageNet dataset by transferring wights. We trained the presented framework on KDDTrain+ dataset and evaluated performance on KDDTest+, and KDDTest-21 datasets, respectively. We also evaluated and compared performance of the framework with other advanced transfer learning from pre-trained models like VGG-19, MobileNet, Inception V3, and ResNet 50. We also compared the results with traditional machine learning models— LR, SVM, and DT methods. The experimental results show that using VGG-16 in the feature extraction process outperforms other pre-trained models that were used for feature extraction in terms of accuracy, precision, recall, false alarm, and f1-score.

6. References

[1] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., and 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] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv: 1704.04 861.

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

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

[5] Xie, Y., and 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., and 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., and 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., and 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., and 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., and Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.

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

[13] Kabir, E., Hu, J., Wang, H., and 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., and 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] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, Intrusion detection using convolutional neural networks for representation learning, Lect. Notes Comput. Sci. (including

Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10638 LNCS, pp. 858866, 2017.

[16] Masum, M., and 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] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[18] Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., and De Geus, P. (2017, December). Malicious software classification using transfer learning of resnet-50 deep neural network. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1011-1014). IEEE.

[19] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[20] Jaworek-Korjakowska, J., Kleczek, P., and Gorgon, M. (2019). Melanoma Thickness Prediction Based on Convolutional Neural Network with VGG-19 Model Transfer Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).