Hindawi
Security and Communication Networks
Volume 2020, Article ID 8891185, 16 pages
https://doi.org/10.1155/2020/8891185

Research Article

A Mean Convolutional Layer for Intrusion Detection System

Leila Mohammadpour, T.C. Ling, C.S. Liew, and Alihossein Aryanfar

Department of Computer System & Technology, Faculty of Computer Science & Information Technology, University of Malaya, W. Persekutuan Kuala Lumpur 50603, Malaysia

Correspondence should be addressed to Leila Mohammadpour; le.vesal@gmail.com

Received 29 March 2020; Revised 16 September 2020; Accepted 5 October 2020; Published 24 October 2020

Academic Editor: Tom Chen

The significant development of Internet applications over the past 10 years has resulted in the rising necessity for the information network to be secured. An intrusion detection system is a fundamental network infrastructure defense that must be able to adapt to the ever-evolving threat landscape and identify new attacks that have low false alarm. Researchers have developed several supervised as well as unsupervised methods from the data mining and machine learning disciplines so that anomalies can be detected reliably. As an aspect of machine learning, deep learning uses a neuron-like structure to learn tasks. A successful deep learning technique method is convolution neural network (CNN); however, it is presently not suitable to detect anomalies. It is easier to identify expected contents within the input flow in CNNs, whereas there are minor differences in the abnormalities compared to the normal content. This suggests that a particular method is required for identifying such minor changes. It is expected that CNNs would learn the features that form the characteristic of the content of an image (flow) rather than variations that are unrelated to the content. Hence, this study recommends a new CNN architecture type known as mean convolution layer (CNN-MCL) that was developed for learning the anomalies' content features and then identifying the particular abnormality. The recommended CNN-MCL helps in designing a strong network intrusion detection system that includes an innovative form of convolutional layer that can teach low-level abnormal characteristics. It was observed that assessing the proposed model on the CICIDS2017 dataset led to favorable results in terms of real-world application regarding detecting anomalies that are highly accurate and have low false-alarm rate as opposed to other best models.

1. Introduction

Worldwide economic and business advancement is closely tied with Internet and enterprise networks. Organizations have stronger ties with computer networks than before within everyday operations and the ways that customers share and store personal information [1]. This rate of progress is closely tied with the complicated management of these networks, where network administrators are responsible for any issues such as flash crowds, network elements failures, mistakes with configurations, malicious attacks, and more. By ensuring prevention or quick fixes of problems, administrators protect the quality of connections for all involved and prevent end users from having their services disrupted [2, 3].

Governments and private organizations require solutions offering stable performance in protecting the information assets they hold from any unlawful or unwanted accesses and attempt to prevent and detect intrusions [4]. Network intrusion detection system (NIDS) describes overseeing and categorizing network flows based on whether they are normal behavior occurring often in a network or if they are movements which could endanger safety of information systems.

Denning [5] suggests building an intrusion detection system (IDS) which would use a number of artificial intelligence (AI) approaches to detect abnormal movements and potential intrusions. This approach established a new wing of intrusion detection systems, developed using learning algorithms. In the last 30 years, machine learning (ML) approaches have been used as a traditional way of creating a network anomaly detection model.

A category of machine learning algorithms known as deep learning has become more widely used in classification and pattern recognition. Deep learning uses information processing layers within a hierarchical architecture to create the deep model. Deep learning has distinctive differences from conventional machine learning, as it can find the ideal features needed within raw data via certain nonlinear transformations, wherein every transformation achieves a greater complexity [6]. Using deep learning to counter-information security problems has not been investigated for very long, and so there are few research studies on the topic, none of which use deep learning techniques to their full potential [7].

Two major gaps were seen during the literature review of anomaly detection problems. Firstly, there is a high false-alarm rate [8–10] for the methods used in anomaly detection. Secondly, training datasets were used in the training as well as the testing of models, employing cross-validation processes. Most modern studies adopt this methodology, and detection rates were extremely high. This is seen in the work of Kim et al. [11], who used a four-layer DNN with 100 units for intrusion detection on the KDD-CUP99 dataset, showing 99% accuracy. However, these practices are not considered reliable for anomaly detection problems since models can be overflowed to perform at these extreme levels [4].

The key aim of this research study is to cover the gaps described above, through the creation and implementation of anomaly detection models, using cutting edge deep learning models, and present an evaluation of these through standardized classification quality metrics. Of the varied practices existing in deep learning, CNN has had exceptional performance when it comes to computer vision, including face and object recognition. CNNs are a category of standard neural networks, as they employ convolution and pooling layers rather than completely linked hidden layers as seen in traditional neural networks [12]. This paper puts forward a newer type of CNN, by learning the anomalies' content features, which can be beneficial when used in intrusion detection. Furthermore, deep learning anomaly detection models have been contrasted with popular classification systems such as support vector machine (SVM), K-nearest neighbor (KNN), decision tree, random forest, adaptive boosting classifier, and gradient boosting. In order to further cover the existing research gaps, some models were trained on the training dataset, without any exposure to the test dataset throughout this process. Following this, the models were evaluated on the testing datasets, allowing for a more accurate and fair evaluation of the model's advantages, through the use of previously unknown data instances used at the time of testing.

This study will put forward an innovative CNN architecture type, created to learn content features of anomalies, while subsequently pinpointing the specific abnormality. The suggested CNN is used to design a robust NIDS, which involves a new type of convolutional layer able to be taught low-level abnormal characteristics. The suggested CNN-MCL produces the lower false alarm compared to the original CNN.

In this paper, Section 2 will provide background information on the study topic, while Section 3 explains an in-depth description of the suggested CNN-MCL layer. Section 4 will present the network architecture for the CNN-MCL model. Section 5 will include the experimental performances of the proposed algorithm on CICIDS2017 [13], while Section 6 will share the conclusion of this study.

2. Background

In this section, related works and motivation, dataset description, and conventional CNN are presented.

2.1. Related Works and Motivation. Because of its efficiency in finding ideal solutions within a finite amount of data, deep learning has gathered significant research attention. Javaid et al. [14] use a deep learning method in the context of a deep neural network for flow-based anomaly detection, and it is seen through the results that deep learning is able to be used for anomaly detection in software-defined networks (SDNs). Tang et al. [15] suggest a deep learning-based approach involving the use of self-taught learning (STL) within the benchmark NSL-KDD [16] dataset, in the context of a network intrusion detection system. In the work of paper [17], an RNN-based model is implemented for the purposes of classification instead of pretraining. In addition, the NSL-KDD dataset is employed for independent training and testing sets, in order to appraise performance in pinpointing network intrusions in both binary and multiclass classifications. The results are then contrasted with J48, ANN, RF, SVM, and other machine learning methods suggested in earlier research. The study of Zhao et al. [18] offers a cutting-edge survey of deep learning applications in the context of machine health monitoring. Experiments were conducted to contrast conventional machine learning methods with four widely employed deep learning methods (autoencoders, restricted Boltzmann machine (RBM), CNN, and RNN). This study found that deep learning methods provide greater accuracy over their conventional counterparts. In the work of Alrawashdeh and Purdy [19], it is suggested that using a RBM with a single hidden layer can undertake unguided feature reduction. The weights are then transferred to another RBM in order to create a deep belief network (DBN), and the pretrained weights are moved into a fine-tuning layer made up of a logistic regression classifier (trained with 10 epochs) with multiclass SoftMax. In the study of Kim et al. [11], a DNN using 100 hidden units is put forward, in conjunction with the rectified linear unit (RLU) activation function and the ADAM optimizer.

The study by Cordero et al. [20] suggested another unsupervised method to train models the normal network flows. RNN, autoencoder, and dropout concepts of deep learning are employed to achieve this. The performance of these suggested methods is not fully released. Along the same lines, Tang et al. [15] suggest a way of overseeing network flow data. In addition, Kang and Kang [21] put forward the notion of using an unsupervised DBN to train certain features to initialize the DNN, offering greater classification performance, even though specific details of the approach are not provided. Their appraisal depicts superior outcomes when it comes to classification error detection.

In the study by Bontemps et al. [22], a real-time collective anomaly detection model using neural network learning and feature operating was described. Here, a LSTM-RNN is trained using normal time series data, prior to making a live prediction for every time step. Furthermore, Ma et al. [23] used the method of spectral clustering (SC) to find the key properties of network traffic, and a multilayer DNN was used to pinpoint attack types. The findings denote that superior performance was seen with the SC-DNN over the SVM, backpropagation neural network (BPNN), random forest (RF), and Bayesian methods, with the highest level of accuracy. On the contrary, weight parameters and thresholds for every DNN layer must be established experimentally and not theoretically. Erfani et al. [24] put forward a mixed model, which used a DBN alongside a one-class SVM. An unsupervised DBN was trained to pinpoint common properties, and a one-class SVM was trained using features taken through the DBN.

A NIDS using a supervised CNN-IDS has been proposed, in which a datapreprocessing step normalizes the dataset; the CNN is trained, optimal features are extracted, and, finally, a SoftMax classifier is used to classify attacks [8]. To decrease computational costs, the traffic input vector is reconfigured into an image format. This model is evaluated using the KDD-CUP99 dataset. Although the study sees a reduction in detection time, the detection rate should be increased and feature learning should be improved for the model to learn the features with a small number of attack categories.

In [25], a hybrid model leverages a grey wolf optimizer (GWO) to propose a CNN for network anomaly detection, and the GWO improves initial population generation, exploration, exploitation, and revamped dropout functionality. In the first step, the GWO selects desired features to establish optimal trade-off between the two main objectives of a minimized feature set and reduced false-alarm rate. In the second step, an improved CNN (ImCNN) is utilized for anomaly classification, and the proposed model is subsequently evaluated on the DARPA98, KDD-CUP99, and synthetic datasets.

To discriminate between normal and abnormal traffic, and to auto-profile traffic patterns, D-PACK has been proposed [9]. This approach integrates an unsupervised CNN model to investigate just the first few bytes of the first few packets in each flow, therefore detecting abnormal traffic early using raw packet-level data. D-PACK is assessed using the USTC-TFC2016 dataset [26].

A combination of bidirectional long short-term memory (BLSTM), attention mechanism, and multiple convolutional (MC) layers has been suggested as the BAT-MC model [27]. This approach uses the structured network traffic information to generate time series features. The MC layers extract the local features, the BLSTM generates the packet vectors, the attention mechanism screens the network flow composed of packet vectors, and a SoftMax classifier is used for final classification. This model is tested with the NSL-KDD and KDD-CUP99 datasets.

Zheng [28] propose two convolution and pooling layers with batch normalization appended to each convolution layer to reduce computational costs and speed up detection. To determine the optimal model, different numbers of convolution and pooling layers are examined and a SoftMax classifier assesses the CNN-extracted features. Evaluation is conducted using the KDD-CUP99 dataset.

An improved CNN for wireless network intrusion detection has been proposed using stochastic gradient descent (SGD) classification and KDD-CUP99-based evaluation although this method demonstrated problems with gradient dispersion and local optima [10]. An alternative CNN model that uses a SoftMax classifier on the KDD-CUP99 dataset is proposed [29] and shows that increasing the number of epochs improves the accuracy of the model. In addition, this approach demonstrates that a CNN model achieves better performance as compared to SVM and DBN.

CNNs offer potential benefits in learning image content to achieve object detection, but they are not yet ideal for anomaly discovery. Expected input flow content is easily found in a CNN, while abnormalities exhibit only small differences from these normal data, and this means that a specific method is needed to detect these slight changes. IDS researchers therefore work to explore whether or not CNNs, for example, can learn to detect these abnormal characteristics as well as normal content features.

Normal and abnormal data flows are not significantly different, and the CNN must distinguish the variations. A standard CNN learns the features that represent flow content; this is primarily normal data so that learning is based on content and not on variation. Although CNN-based models have been used to address key challenges in anomaly detection [8–10, 25, 27–29], the mentioned problem of CNN remains unresolved. Solutions proposed thus far have focused on feature selection, model structure, and fine-tuning, while the main objective of MCL-CNN developed in the present study is the detection of the minor differences between the abnormalities and the normal content. Furthermore, most existing studies have evaluated models using the NSL-KDD or KDD-CUP99 datasets; the CICIDS2017 set will be used here to evaluate the model with novel attacks in the testing phase.

2.2. Convolutional Neural Networks. A CNN is a type of neural network which aims to learn appropriate feature representations of the input data. Under this type of architecture, generally, the initial layers are a collection of convolutional feature extractors used alongside an image through a number of learnable filters. The filters used act as a sliding window, which moves across all areas of an input image, where the overlapping distance is known as the stride, and the outputs created are known as feature maps. Every CNN layer is made up of numerous convolution kernels employed to produce a different feature map. Neighboring neurons areas are linked to a neuron of a feature map of the next layer. To produce the feature map, all spatial locations of the input share the kernel. Following convolution and pooling layers, one or multiple full connected layers complete the classification [30–33].

The convolutional operation across input feature maps and a convolutional layer within the CNN architecture is provided through the following equation:

$$\mathbf{h}_j^{(n)} = \sum_{k=1}^{K} \mathbf{h}_k^{(n-1)} * \mathbf{w}_{kj}^{(n)} + b_{kj}^{(n)}, \qquad (1)$$

where $*$ is the a $2d$ convolution, $\mathbf{h}_j^{(n)}$ is the j^{th} feature map's output in the n^{th} hidden layer, $\mathbf{h}_k^{(n-1)}$ is the k^{th} channel in the $(n-1)^{th}$ hidden layer, $\mathbf{w}_{kj}^{(n)}$is the weights of the k^{th}channel in the j^{th}filter in the n^{th} layer, and $b_{kj}^{(n)}$is its corresponding bias term.

For every layer, the filter coefficients are seeded with random values to start and then learned through the backpropagation algorithm [34]. In addition, convolutional layers also involve an activation function to establish nonlinearity. The collection of convolutional layers produces a substantial volume of feature maps. To help limit the dimensionality of these properties, convolutional layers are followed by an additional layer, called pooling, in order to limit the computational expense of training within the network and reduce the potential for overfitting. A number of pooling operations exist, including max, average, and stochastic pooling. For the max-pooling layer, this acts as a sliding window with a stride distance in place to set the maximum value inside the dimension of a sliding window.

A CNN's training is completed with an iterative algorithm moving between feedforward and backpropagation data movements. At every iteration of the backpropagation, the convolutional filters and fully connected layers are updated. A key aim is to limit average loss E across the true class labels and the network outputs, i.e.,

$$E = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{c} y_i^{*(k)} \log\left(y_i^{(k)}\right), \qquad (2)$$

where $y_i^{*(k)}$ and $y_i^{(k)}$ are, respectively, the true label and the network output of the i^{th} input at the k^{th} class with m training input and c neurons in the output layer. A number of solutions have been suggested to limit average loss [35–37], and this paper employs the adaptive moment estimation (Adam) [37] to train the model.

2.3. Dataset Description. This paper has presented numerous experiments regarding the CICIDS2017 [13] dataset which is an intrusion detection as well as prevention dataset. The Canadian Institute for Cyber Security (CIC) obtained this dataset, and it is publicly available for researchers and students. Recently, prominent IDS research has used this dataset because of its criteria and because its size is the largest available dataset concerning real-world data. The fact that the majority of other datasets have outdated data is a well-known problem as intrusion attack types are constantly changing and becoming increasingly sophisticated. Some datasets also have other problems such as no metadata and features, while some do not include adequate diversity in known attacks. All criteria required for developing a precise dataset, according to Gharib et al. [38], are fulfilled by the CICIDS2017 dataset. Sharafaldin et al. [13] noted that this dataset is the most complete one thus far. Furthermore, this dataset includes favorable as well as recent common attacks that are similar to the true real-world data (PCAPs). The results concerning the network traffic analysis conducted through CICFlowMeter with labelled flows as per the time stamp, source and destination ports, source and destination IPs, and protocols and attack (CSV files) are also included.

The CICIDS2017 benchmark dataset includes 25 users' abstract behavior as per the SSH, HTTP, FTP, HTTPS, as well as e-mail protocols. The data gathering period began at 9 a.m. on Monday, 3 July 2017, ending at 5 p.m. on Friday 7 July 2017, thus lasting five days. As only Monday has normal activity, this day only has benign traffic. On other days, besides benign traffic, various attacks are implemented such as brute force SSH, brute force FTP, DoS, web attack, heartbleed, infiltration, DDoS, and Botnet.

Following data cleaning which involves eliminating the records that have missed values, it is noted that the total collected data include 2827876 records, with 2271320 normal records, and 556556 abnormal records. Every specified attack's labelled records are stored in a specific CSV file format with every CSV file formed of a particular number of labelled records described by 78 features and 1 label. All records include two types of features which are nominal and numerical. The five features are nominal data-type features, while the remaining are numerical data-type features.

However, a set of experiments were conducted based on the CICIDS2017 dataset for verifying the CNN-MCL performance on the intrusion detection task. Because, unlike other IDS datasets, the CICIDS2017 was not divided by the provider into training and test datasets, and it was divided into training and test records in this paper based on the nature of the experiment as presented in each experiment.

3. Mean Convolutional Layer (CNN-MCL)

This paper suggests a CNN-based layer to separate anomalies from normal data. The method used involves using the data to directly learn changes occurring through abnormal data. The main issue was that normal and abnormal flows are not greatly different, and so the CNN must be forced to detect abnormalities variations. It is seen that if standard form CNNs are used to detect, features are learned which represent an image's content (flow's content), primarily normal flow, meaning that the classifier identifying data content is linked with training data instead of learning data variations.

However, the approach used here was designed to hold back the content and adaptively learn abnormality traces. In order to achieve this, an innovative convolutional layer is proposed, known as the mean convolutional layer (CNN-MCL), established for use with intrusion detection system tasks. In turn, these errors are employed as low-level abnormal/normal features, where more advanced abnormal detection features are created thereafter. In order to echo these actions, the suggested layer aims to exclusively learn prediction error filters. The feature maps created are then linked with prediction error fields employed as low-level abnormal traces.

The CNN-MCL is able to be positioned differently from the CNN aimed to undertake IDS tasks. This acts as a way of holding back the content, as prediction errors primarily do not include flow content, and this offers the CNN low-level IDS features. Deeper layers of the CNN are able to learn higher level of features as a result of the low-level abnormal characteristics.

With the equation below, one can define the CNN-MCL, where L denotes the L^{th} CNN-MCL, the subscript k describes the k^{th} convolutional filter within a layer, and that the central value of a convolutional filter is defined by (c_x, c_y). The CNN is then forced to learn prediction error filters through actively implementing specific constraints:

$$\mu^{(L)} = \text{Mean}\left(w^{(L)}\left(c_x, c_y\right)\right),$$

$$\begin{cases} w_k^{(L)}(x, y) = \dfrac{w_k^{(L)} \times \mu^{(L)}}{\sum w_k^{(L)}}, & (x, y) \neq \left(c_x, c_y\right), \\ \\ w_k^{(L)}(x, y) = -\mu^{(L)}, & (x, y) = \left(c_x, c_y\right). \end{cases} \quad (3)$$

Predictions of CNN-MCL are established via a specific training process. Following this, updates of filter weights $w_k^{(L)}$ at each iteration are made, with the Adam algorithm in the backpropagation stage. Then, the updated filter weights are set into the feasible set of prediction error filters by CNN-MCL reinforcement, and projection is undertaken at every training iteration. This is achieved by firstly setting the central filter weight to the negative mean value of the middle values of all k filters in the layer, and then, the filter weights left over are normalized using equation (3). There are two steps to this process. Firstly, the remaining weights are multiplied with the mean value, and then, the collected weights are dividing against the sum of all filter weights, without including the central value. In layer L, the midpoints of all k filters are set to the negative mean value. The pseudocode of this process is seen in Algorithm 1.

To provide intuition into this, suppose the prediction is formed by using some function $f(X)$ to predict normality or abnormality of input data (flow). In particular, $f(X)$ is a classifier which predicts based on the extracted features from Feature Extraction. Moreover, suppose $g(I)$ is the extracted feature; therefore, the full operation is formed as $f(g(I))$. For simplicity, we assume a network with one CNN-MCL and one CNN in the feature extraction section. Regarding equation (1), using conventional CNN, the classifier generates the output (predicts) based on the following equation:

$$f(X) = f(g(I)) = f\left(\sum_{k=1}^{K} I * \mathbf{w}_{kj} + b_{kj}\right), \quad (4)$$

where f is the classifier function, g is (or is part of) the feature extraction process, I is the input data, $*$ is the $2d$ convolution, k is the number of channels, $\mathbf{w}_{kj}$ is the weights of the k^{th}channel in the j^{th}filter, and b_{kj} is its corresponding bias term.

The classifier detects the anomaly based on the following equation:

$$f(X) = f(g(I)) = f\left(\sum_{k=1}^{K} I * \tilde{\mathbf{w}}_{kj} + b_{kj}\right), \quad (5)$$

where $\tilde{\mathbf{w}}_{kj}$ is the weights generated by the CNN-MCL from the k^{th}channel in the j^{th} filter. Then, regarding equations (3) and (5), we have

$$f(X) = f(g(I)) = f\left(\sum_{k=1}^{K} I * \frac{w_k^{(L)} \times \mu^{(L)}}{\sum w_k^{(L)}} + b_{kj}\right), \quad (6)$$

$$f(g(I)) = f\left(\sum_{k=1}^{K} I * \frac{w_k^{(L)} \times \text{Mean}\left(w^{(L)}\left(c_x, c_y\right)\right)}{\sum w_k^{(L)}} + b_{kj}\right). \quad (7)$$

It can be seen from equation (7) that the $\text{Mean}(w^{(L)} \times (c_x, c_y))$ is calculated from all channels because it does not have the subscript k. Therefore, the mean value is shared among all channels. Then, for each individual channel, the weights are normalized using $w_k^{(L)} / \sum w_k^{(L)}$. These operations reduce the effect of normal context to be extracted as useful features, and they are progressing the effect of abnormal variations to be considered as the extracted features.

To show the advantage of CNN-MCL in the learning process compared to standard CNN, we have evaluated a simple model using a CNN layer and a classifier. Our goal here is to visualize the difference between the extracted features from the CNN-MCL and conventional CNN. Therefore, for this objective, we generate a simple dataset to control the location and value of anomalies in our dataset. However, in Main Experimental, we have evaluated our proposed CNN-MCL with a real-world dataset.

The dataset is generated based on the specifications below. Moreover, we have tested with different specifications and got similar behaviors for all evaluations:

- The input size: 11×11
- Normal data: random uniform number in the range [0, 1]
- Abnormal train data: normal data + random integer between [5, 10].
- Abnormal test data: normal data − random integer between [5, 10].
- Number of training records: 100000
- Number of testing records: 10000
- The ratio of abnormality records in the training data in the evaluation 1 : 10%.
- The ratio of abnormality records in the training data in the evaluation 2 : 30%
- The ratio of abnormality records in the testing data in the evaluation 1 and evaluation 2 : 50%
- Abnormal location: first row of the input and first to fifth values

Initialize $\mathbf{w}'_k$ using randomly draw n weights
$i = 1$
while $i \le$ maximum_iter **do**
Do feedforward pass
Update forever weights through Adam and backpropagation errors
Set the $\mu^{(L)} =$ mean of central points of all filters of layer L
Update the weights of layer L using the $w_k^{(L)}(x, y) = ((w_k^{(L)} * \mu^{(L)})/t \sum w_k^{(L)})$
Update the weights of central points of k filters of layer L using the $w_k^{(L)}(c_x, c_y) = -\mu^{(L)}$
$i = i + 1$
If training accuracy converges **Then**
Exit
End.

Algorithm 1: Pseudocode of CNN-MCL.

It can be seen from the above dataset that the values of abnormality of test data are different from the abnormality of train data.

We test a simple model with one CNN-MCL and one CNN model. Accordingly, in one model, the proposed CNN-MCL is applied, and in the other model, only CNN is used. We train the model on 20 epochs, and we plot three channels of the output (weights) of CNN-MCL and CNN layers to compare the CNN-MCL and CNN visually. For visualization purposes, we have created pseudocolor plots of the 2D array applying quadrilaterals using Matplotlib [39], as shown in Figures 1–5. In Figure 1, samples of (a) normal, (b) abnormal train, and (c) abnormal test record with data in the ranges [0, 1], [5, 11], and [−4, −10], respectively.

In the first evaluation, we have only 10% abnormal records in the training data. We choose the first three channels of the output of the layers (weights) and plot them. Figures 2 and 3 show the output of the extracted features (weights) of an abnormal input after the CNN and CNN-MCL layers. It can be seen from Figure 2 that the CNN-MCL model distinguishes the abnormality location and that the extracted features (values) in the abnormality neighborhood are significantly different from normal, whereas all values in Figure 3 are in the range [0, 1].

We then increase the abnormality rate of the training data from 10% to 30% and repeat the evaluation. The output weight plots of the CNN and CNN-MCL models for one of the abnormal records are presented in Figures 4 and 5, respectively. The figures show that the CNN-MCL and CNN layers distinguish the abnormality location effectively; the extracted features (values) in the CNN-MCL abnormality neighborhood are significantly different from normal values but less so in the CNN model.

Comparison between Figures 2 and 3 or between Figures 4 and 5 shows that the CNN-MCL has more distinct abnormal feature presentation than CNN, and the abnormality has been diagnosed very well. On the contrary, from the visual perspective, CNN-MCL has extracted weights to distinguish between normal and abnormal parts more accurately than CNN.

The abnormal data in the test data can be thought of as the unseen abnormal data because the range of abnormal data in training data is [5, 11] (random normal data + random integer between [5, 10]), but the range of abnormal test data is [−4, −10] (random normal data − random integer between [5, 10]).

For the more realistic experiment, we set the location of abnormality randomly and evaluated the models. Based on this explanation, two datasets with 10% and 30% abnormality rates have been generated. The results in Table 1 illustrate the accuracy of the CNN-MCL and CNN layers in detecting abnormal records in the test data with the CNN-MCL model outperforming the CNN.

4. Network Architecture

The CNN-MCL layer is used for devising a CNN, which can make the distinction between NIDS' normal and abnormal flows. Figure 6 illustrates the overall design of the suggested architecture including every layer in detail.

This architecture can gain information about new associations between deeper layers' feature maps by extracting a higher level of representation regarding the previously learned normal/abnormal features. The final convolution layer's output is flattened and then fed to the classification block that includes a fully connected and a SoftMax layer. A detailed overview of the suggested architecture and the different layers used in the CNN's architecture is presented.

4.1. Reshaping Layer (Layer 0). The input vector's (one flow) shape in NIDS is typically indicated by a $N \times V$ or $N \times 1 \times V$ vector, in which V refers to the number of features that indicate the flow and N is the batch size. As implemented in a previous work [40], the input shape is changed from 1D to 2D. Hence, this architecture's first layer is one which can reshape the input vector into an 11×11 2D matrix and has less than 121 features regarding the input vector size, and thus, the remaining values are zeros. Thus, layer 1 is sent an $N \times 11 \times 11$ patch. Because in the tested dataset, the feature number is more than 100 and less than 121, we choose 11×11 patches. This architecture can handle other sizes, such as $n \times m$, but the feature arrangement in the test and train must be the same.

4.2. CNN-MCL Layer (Layer 1). In their present form, CNNs often learn content-dependent features, and thus, the

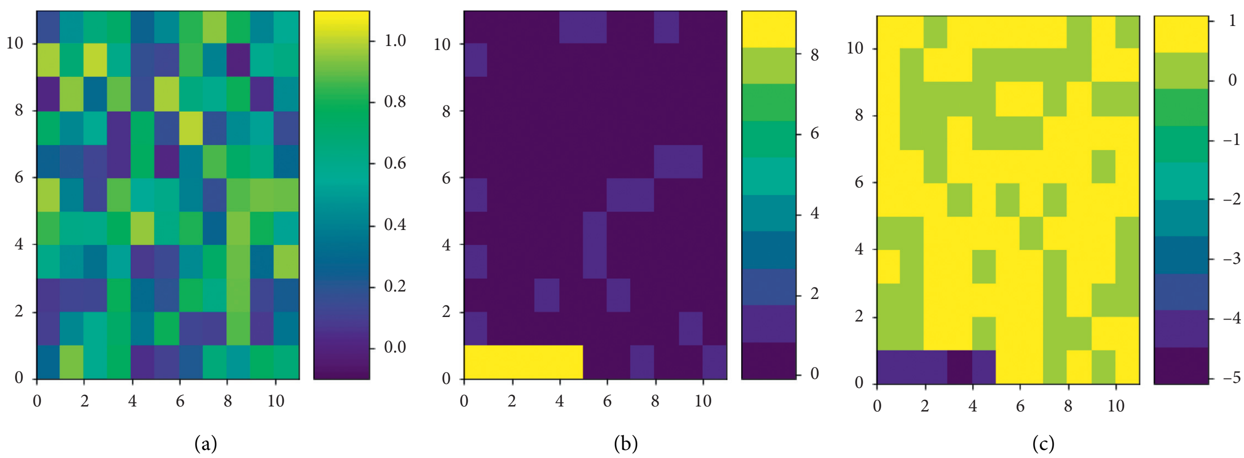


FIGURE 1: Sample of normal (a), train abnormal (b), and test abnormal (c) records.

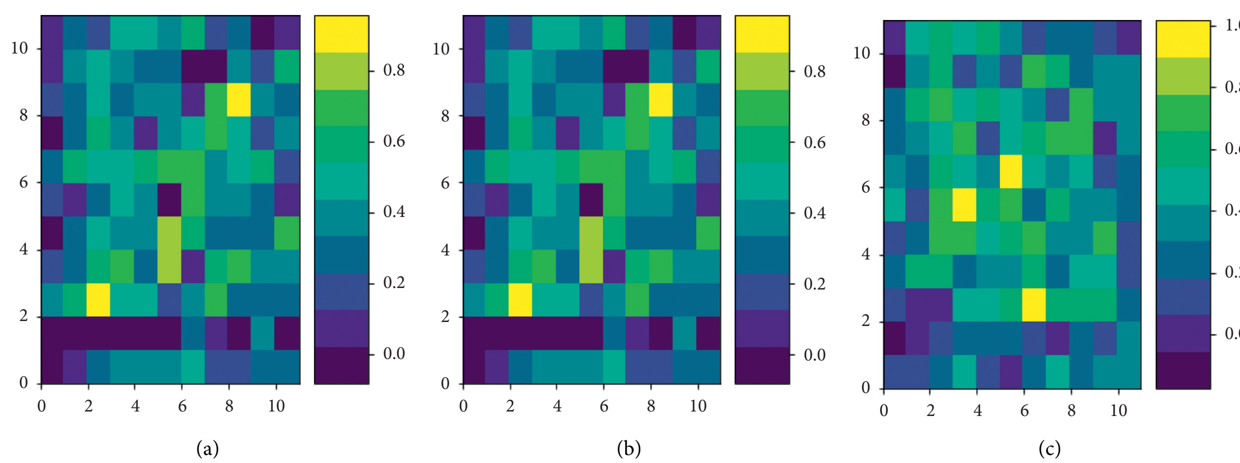


FIGURE 2: Sample output of the CNN layer with 10% abnormal data.

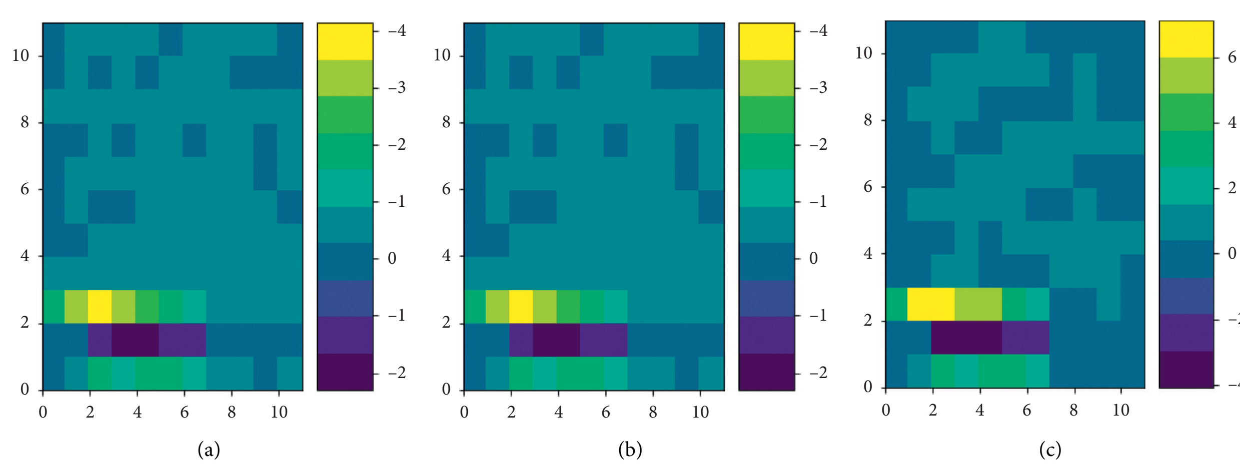
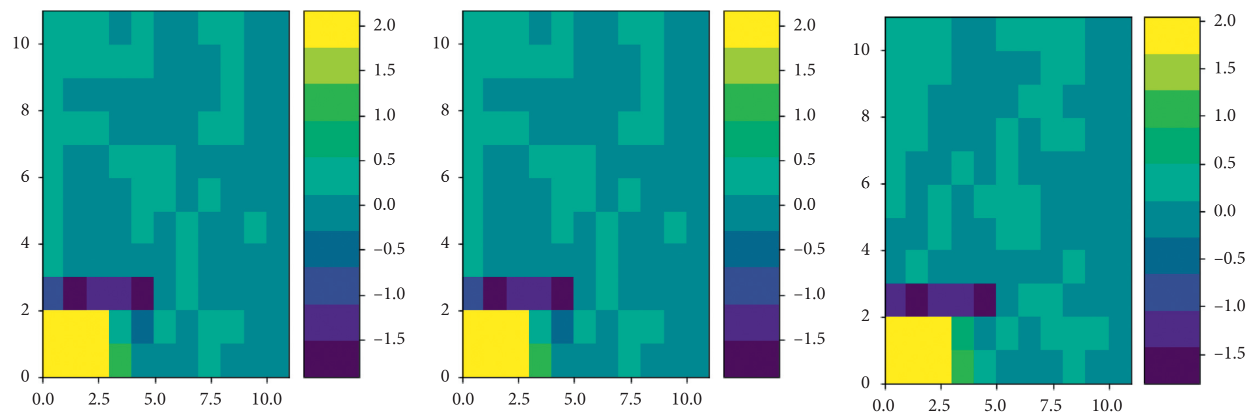


FIGURE 3: Sample output of the CNN-MCL layer with 10% abnormal data.

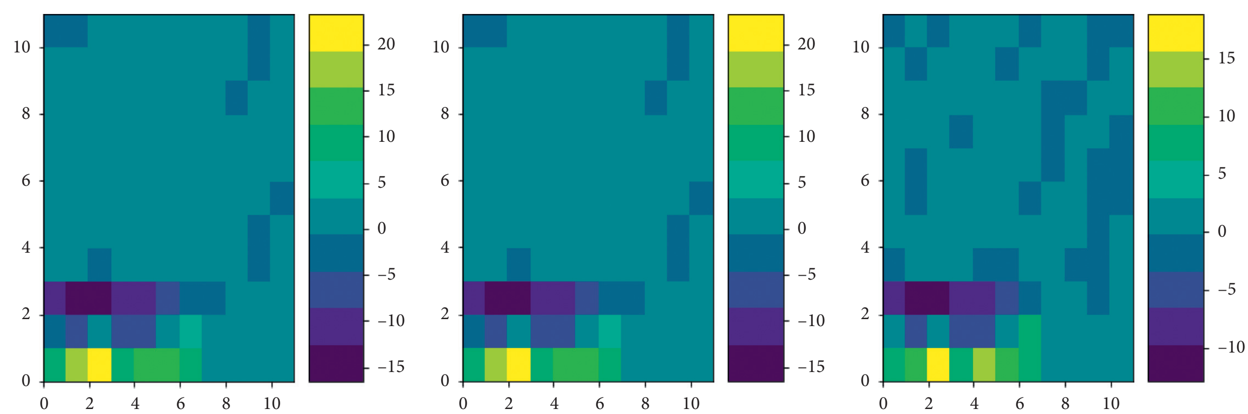
proposed architecture has the CNN-MCL layers forming the second layer (layer 1). This leads to this layer learning value-dependency features which are fragile and can be eliminated by various nonlinear operations [41] including activation layers and pooling. The CNN-MCL layers' output is directly given to a regular convolutional layer.

Particularly, layer 1's input of 11×11 size is first convolved involving 32 diverse 5×5 filters that have a stride



(a) (b) (c)

Figure 4: Sample output of the CNN layer with 30% abnormal data.



(a) (b) (c)

Figure 5: Sample output of the CNN-MCL layer with 30% abnormal data.

Table 1: Accuracy of the CNN-MCL and CNN on the test data to detect the abnormal records.

	Accuracy of detection (30 (%) abnormality)	Accuracy of detection (10 (%) abnormality)
Fixed location (novel abnormality)	91	86
Random location (novel abnormality)	79	50

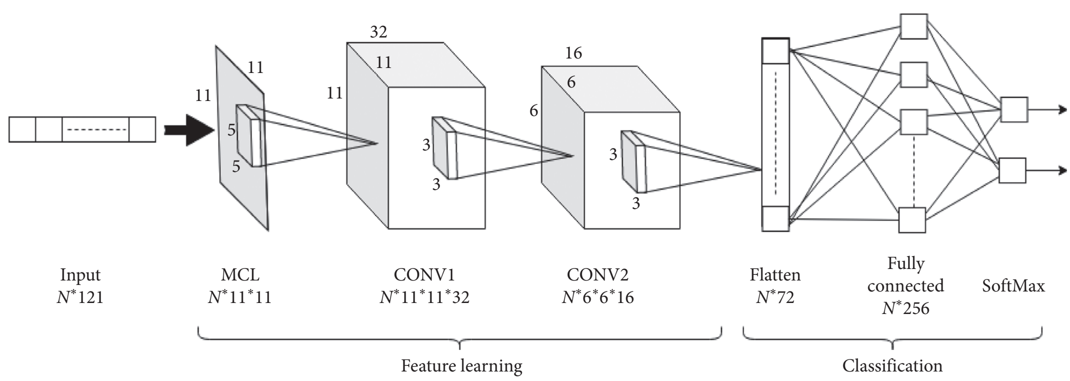


Figure 6: Design of the proposed architecture.

equivalent to 1. Such filters are able to learn the prediction error features regarding the estimated center value as well as its local neighbors. Furthermore, the CNN-MCL layer results in prediction of abnormal feature maps that have dimensions of $11 \times 11 \times 32$.

Generally, the larger-sized filter (such as 5×5) captures generic features and essential components in the inputs. However, the smaller-sized filter (3×3) captures the sophisticated features and has better weight sharing. Therefore, we utilize a 5×5 filter for the CNN-MCL layer (first layer) and a 3×3 filter for the other layers. Furthermore, in this paper, we represent the efficiency of using these filter sizes with some experiments in Experimental Results.

4.3. Convolutional Block. For learning higher level prediction error features, a set of convolutional layers is used, and every layer is followed by an activation function, batch normalization, and pooling layers. Furthermore, every such convolutional layer is referred to in this paper as the convolution block. Every convolutional layer will learn feature maps' new representation which is learned by the previous convolutional layer or lower-level features.

As seen in Figure 6, general convolutional layers (convolution block) are used in third (layer 2) and in fourth (layer 3) layers. Moreover, for learning higher level representative features as well as new associations amongst the prediction feature maps, regular convolutional layers (layer 2 and layer 3) are used. The convolutional blocks' output dimensions are $6 \times 6 \times 32$ and $3 \times 3 \times 8$, respectively. These layers within the proposed structure are further explained below.

4.3.1. Activation Function. Generally, a nonlinear mapping known as an activation function follows a convolutional layer. Such function is then applied to every value within the feature maps of each convolutional block. Activation functions are of various types. Regarding computer vision applications, there has been successful implementation of the ReLU activation function [42, 43]. A different activation function type was recommended by He et al. [44] called PReLU which creates surpass human-level performance concerning visual recognition challenge [45]. Moreover, Clevert et al. [46] suggested the exponential linear units (ELUs) activation function that can speed up learning significantly obtaining below 10% classification error as opposed to a ReLU network having the same architecture.

It is possible to strengthen the capability of CNN for separating feature space by including nonlinearity across the network layers. The proposed CNN recommends restricting the data values range having the ReLU activation function at the network's each stage. On the contrary, it is known that an activation function layer does not follow the feature maps that are learned by the CNN-MCL layer. This is primarily because it is possible to easily eliminate the learned prediction error features using several nonlinear operations such as activation functions.

4.3.2. Batch Normalization. Computer vision researchers have devised numerous methods for normalizing the data in CNN architecture. In early deep learning architectures, the local response normalization (LRN) layer is used that normalizes the central coefficient in a feature map's sliding window concerning its neighbors. Ioffe and Szegedy [47] recommended the batch normalization layer that drastically accelerates the deep networks training. Such a mechanism reduces the internal covariate shift that is the input distribution change regarding a learning system.

For this, a zero-mean and unit-variance transformation of the data is implemented along with the CNN model being trained. The parameters of each previous layer impact every layer's input and amplified even the small changes. Hence, such a layer deals with a significant problem and enhances a CNN model's final accuracy. This is why the proposed architecture implements a batch normalization layer following every regular convolutional layer.

4.3.3. Pooling. This CNN utilized max-pooling of 3×3 size and stride of 2. There is maximum value in the max-pooling layer in the sliding window's local neighborhood. Such a layer aims to minimize the feature maps' dimensionality which, in turn, diminishes the computational cost required for training and reduces the possibility of overfitting. In particular, the set of parallel convolutional operations provides a feature maps volume of high dimension. Thus, pooling layers retain the features that are most representative and help in subsampling and enhancing the accuracy. In this architecture, the two pooling layers that are used have decreased the feature maps dimensions from $11 \times 11 \times 32$ and $6 \times 6 \times 16$ to $6 \times 6 \times 16$ and $3 \times 3 \times 8$, respectively.

$3 \times 3 \times 8$ is then reshaped for layer 4 as 72 outputs. Such a mapping form learns the association throughout feature maps, which is a linear combination of features through channels in the same location. However, the previously learned hierarchical features are developed by learning local spatial association in a field that is receptive (local region/patch convolved with a filter) within the same feature map. Finally, a new association between these feature maps is learned.

4.4. Classification Block. Figure 6 shows that a neural network classifier having a SoftMax activation function within the output layer is implemented for classifying the output features that are learned by the convolutional layers set. Such an activation function maps features in which the last layer that is fully connected to a set of probability values learns in which all neurons' output in this layer equals 1. The abnormal flows can be identified by selecting the editing operation related to the SoftMax layer neuron that has the highest activation level. In particular, this layer which is fully connected includes 256 neurons. Furthermore, this layer learns new relations among CNN's deepest convolutional features.

5. Experimental Results

For examining the performance of the proposed NIDS approach, a set of experiments and analysis were implemented on the well-known datasets called CICIDS2017. This section presents the experimental work to validate CNN-MCL as well as compare its performance with that of two approaches: the state-of-the-art methods (deep learning) DL and the NON-DL methods (which this paper refers to as machine learning methods).

Regarding the IDS task experiment, the criteria derived from estimating a confusion matrix as a classification problem was used. The confusion matrix aims to compare actual labels with predicted labels. It has been known that an intrusion detection problem includes two classes: normal and attack, which is defined by a 2-by-2 confusion matrix for evaluation. Similar to any classification problem, the confusion matrix of IDS task includes the terms TP-true positive, FP-false positive, TN-true negative, and FN-false negative. Generally, in the IDS task, the terms TP, FP, TN, and FN are regarded as an attack data that are correctly classified as an attack, normal data that are incorrectly classified as an attack, normal data that are correctly classified as normal, and attack data that are incorrectly classified as normal, respectively. These four terms help in generating the IDS evaluation measures which are accuracy (ACC), precision (P), recall (R), false alarm (FA), and *F*-score (F1).

Moreover, all mathematical parts related to the evaluation measures are presented in Table 2. The accuracy measures refer to the proportion of the total number of correct classifications. The precision measures the number of correct classifications penalized by the number of incorrect classifications. The recall or sensitivity measures the number of correct classifications penalized by the number of missed entries. The false alarm rate measures the proportion of benign events incorrectly classified as malicious. The *F*-score rate measures the harmonic mean of precision and recall which serves as a derived effectiveness measurement.

All standard machine learning algorithms were applied by Scikit-learn [48] which is an open source machine learning library for the Python programming language. All CNNs were implemented using the Tensorflow [49] which is a Python deep learning framework. The experiments were all conducted using an Amazon AWS EC2 instance (p2.xlarge) with specifications as follows: 4 Intel Xeon E5-2686, 61 GB RAM, 1 NVIDIA K80 GPUs, each with 2,496 parallel processing cores and 12 GiB of GPU memory. Furthermore, Table 3 illustrates the CNN-MCL parameters which are shared between all the experiments.

This section includes datasets description in Section 6.1. Next, the reliability of CNN-MCL is analyzed using various ML approaches in Section 6.2. Then, the CNN-MCL for single attack is explored in Section 6.3. The structural design is explored in Section 6.4. Finally, CNN-MCL for multi-attack is presented in Section 6.5.

5.1. CNN-MCL versus ML. In this section, a set of experiments are conducted for discussing and comparing the performance of IDS as per various machine learning (ML) methods. ML experiments were also applied to compare the proposed CNN-MCL with machine learning approaches. Furthermore, a binary classification was implemented using these ML approaches to predicate the normal and abnormal flows in CICIDS2017.

Table 2: Equations of the performance measures.

Accuracy	$(TP+TN)/(TP+TN+FP+FN)$
Precision	$TP/(TP+FP)$
Recall	$TP/(TP+FN)$
False alarm	$FP/(FP+TN)$
F-score	$2*((Precision\ *\mathrm{Recall})/tPrecision+\mathrm{Recall})$

Table 3: CNN-MCL parameters.

Learning rate	$10e{-}4$
Decay steps	5000
Decay rate	0.99
Optimizer	Adam
Dropout rate	0.5
Regularizer scale	0.01
Regularizer beta	0.01
Activation function	ReLU
Batch size	128

To this end, the best ML classifiers were used which were k-nearest neighbors (K-NN), two types of support vector machine classifiers (SVM and NuSVC), decision tree (DT), random forest (RF), adaptive boosting classifier (AdaBoost), and gradient boosting (GB). Additionally, the default parameters of ML methods (from Scikit-learn) were considered for this experiment. The network configuration and hyperparameters were selected based on the lack of records in the dataset with CNN-MCL characteristics.

To address the problem of unbalanced datasets in CICIDS2017, the abnormal records from the dataset were first divided into training and test sets at 70% to 30% ratio, respectively, and the maximum epoch has set to 50. Second, these training and test sets were increased by adding an equal number of normal records chosen randomly. However, regarding a large-scale dataset classification by ML methods [50], experiments were conducted using a six subsample dataset including 10000, 20000, 40000, 60000, 80000, and 100000 records.

Table 4 presents the accuracies of the models regarding the change in the number of records. The table shows that the number of records is obviously related to the models' generated accuracy as it is evident that the accuracy of all models increased after using a higher number of records. The proposed CNN-MCL model is more superior compared to all other ML models concerning increasing the number of records in terms of both accuracy and *F*-score where both the accuracy and *F*-score of 10000 records case was 96.77%, and when the records number increased to 100000, both the accuracy and *F*-score increased to 99.87%, which outperforms all other models regarding high number of records being used because of the capability of CNN-MCL for recognizing normal and abnormal flows. On the contrary, it shows that when the number of training records increased, the CNN-MCL produced better results compared to the ML methods.

Figure 7 shows that the false alarm rate decreases when the number of training records increases. Furthermore, all

Table 4: Accuracy and *F*-score of the ML and CNN-MCL with different numbers of records.

	10000		20000		40000		60000		80000		100000	
	Acc	F-M	Acc	F-M	Acc	F-M	Acc	F-M	Acc	F-M	Acc (%)	F-M (%)
KNN	96.42	96.39	96.97	96.95	98.10	98.09	98.47	98.46	98.69	98.69	98.89	98.88
NuSVC	96.18	96.31	96.80	96.88	97.80	97.84	97.93	97.97	98.05	98.09	98.06	98.09
DT	97.76	97.79	97.48	97.51	99.58	99.58	99.68	99.68	99.69	99.69	99.74	99.74
RF	97.67	97.71	97.64	97.67	99.77	99.77	99.84	99.84	98.78	98.78	99.16	99.15
AdaBoost	96.97	97.01	96.87	96.89	98.97	98.96	98.19	98.19	98.62	98.62	98.39	98.39
GB	97.57	97.61	97.40	97.44	99.41	99.40	98.90	98.88	99.43	99.43	99.45	99.45
SVM	76.06	80.64	87.35	88.71	97.76	97.80	97.93	97.96	98.05	98.08	98.04	98.08
CNN-MCL	96.77	96.77	96.95	96.96	98.56	98.56	99.34	99.33	99.88	99.88	99.87	99.87

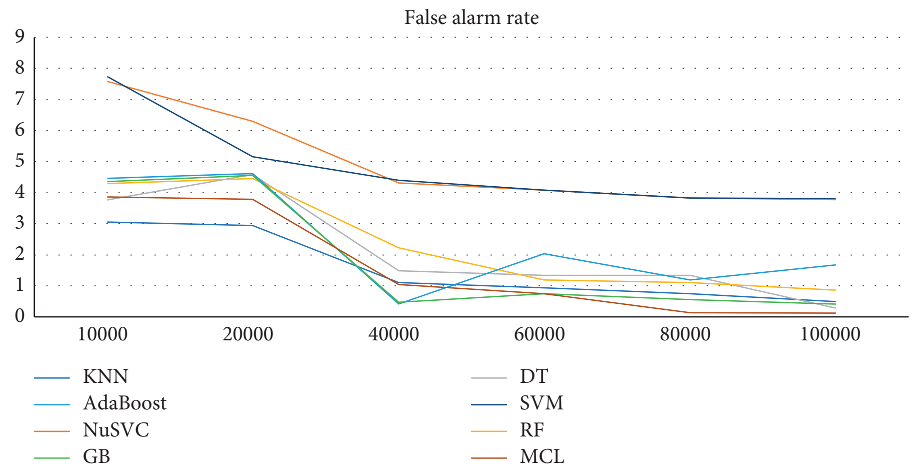


Figure 7: CNN-MCL and NON-DL false alarm rate with respect to different numbers of training records.

models begin high false-alarm rate when the number of records is small (10000), after which the false alarm rate decreased, and the CNN-MCL produces superior results with high number of records (80000–100000). This shows that the CNN-MCL requires more training data compared to the conventional ML methods.

Hence, the CNN-MCL can be considered to be a more suitable method for IDS because of its reliability and validity in detecting an intrusion attack in high-scale dataset.

5.2. CNN-MCL for Unknown Attacks. In this experiment, the capability of the proposed CNN-MCL was evaluated for detecting new types of attacks without pretrained knowledge. Furthermore, the results of CNN-MCL are compared with that of the pure CNN model, for which a set of experiments were conducted in each scenario. One attack was chosen for testing the model that was trained with the remaining attacks and the remaining dataset records. The dataset contains 14 types of attacks, but only 10 of them were used, while four types of attacks were ignored because of the small number of records that prevent the capability of training and testing in such a scenario.

Figure 8 illustrates the accuracy for each type of the 10 attacks, with the x-axis representing the attack type with the number of attacks and y-axis representing the accuracy percentage. The developed model is undeniably superior compared to the CNN in detecting almost all types of new attacks, and these results are interpreted by the capability of CNN-MCL for learning the abnormal features.

Figure 9 shows the same observation that CNN-MCL outperforms CNN in terms of false alarm rate in almost all attack types.

The current section of the study was evaluated through 20 independent experiments. Tables 5 and 6 present the confusion matrices of the classification results using the CNN-MCL and CNN models on DoS HULK and PortScan attacks, respectively. We chose these types of attack because they have the maximum number of records. Although the results in these tables show that the true-positive rate of CNN is higher than CNN-MCL, the false-negative rate of the CNN-MCL model is lower.

For more elaboration, the evaluation measures, precision, recall, and *F*-score are generated for both CNN and CNN-MCL. As shown in Table 7, it is evident that CNN-MCL outperforms CNN in terms of precision in all types of attacks with the highest percentage for DoS HULK attack at 58.64% while that of CNN was 58.5%. Furthermore, it is seen that CNN-MCL has superiority over CNN regarding *F*-score for all types of attacks with the highest percentage for DoS HULK attack at 73.68% while that of CNN was 73.58%. Meanwhile, CNN was slightly superior in terms of recall in certain attack types. However, this is acceptable considering good accuracy and other measures.

5.3. CNN-MCL versus DL Methods. This section assesses the performance of binary classification of normal and abnormal for three DL approaches. In the DL approaches, the

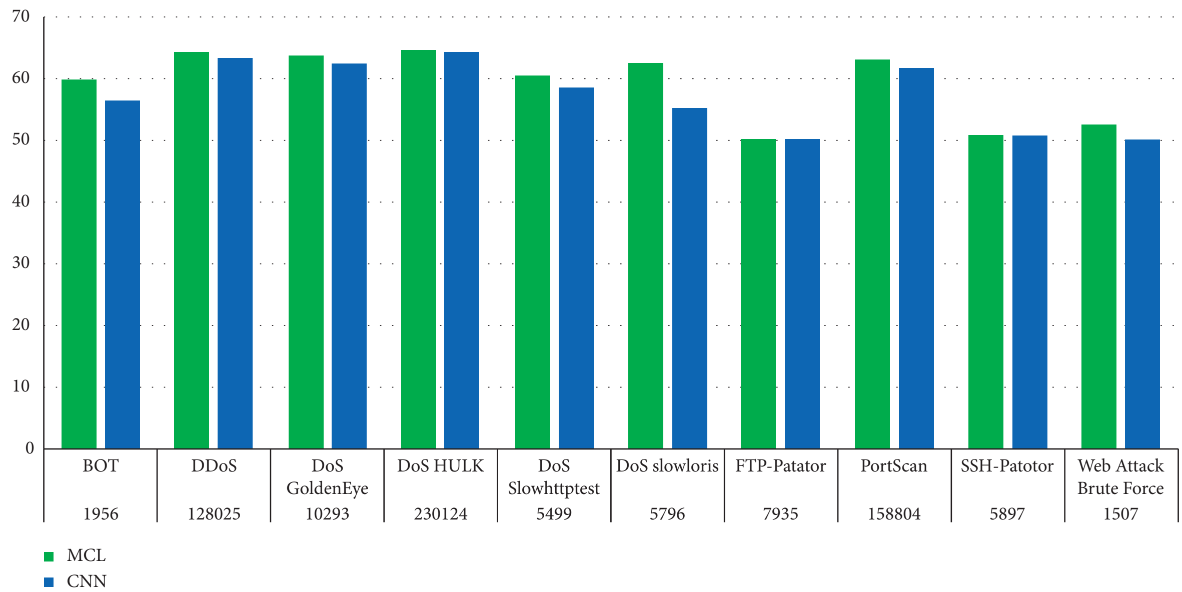


Figure 8: Accuracy of CNN-MCL for different types of attacks.

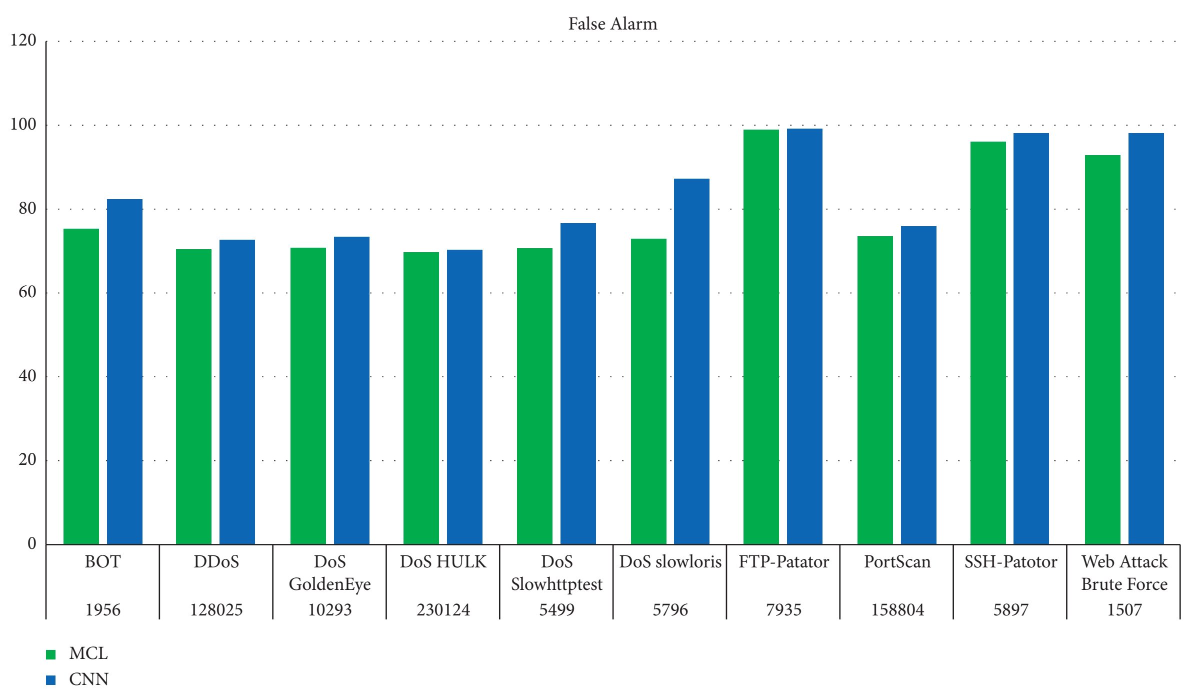


Figure 9: False alarm of CNN-MCL for different types of attacks.

Table 5: Confusion matrix of the CNN-MCL and CNN on DoS HULK attack.

		CNN-MCL		CNN	
		Predicted class		Predicted class	
		Attack	Normal	Attack	Normal
Actual class	Attack	228051	160848	228151	161856
	Normal	2073	69276	1973	68268

baseline method standard CNN, the C-CNN [51] which is the most similar method to the proposed method but in a different domain, and the proposed CNN-MCL model are selected. For fair comparison, the parameters of these three CNN models were unified. Furthermore, the epoch number is set to 50.

Unlike non-DL methods, DLs perform better when training with a large dataset. Subsequently, this section used the full dataset size in training and testing at the 60 :

Table 6: Confusion matrix of the CNN-MCL and CNN on PortScan attack.

		CNN-MCL Predicted class		CNN Predicted class	
		Attack	Normal	Attack	Normal
Actual class	Attack	158583	116637	158214	120484
	Normal	221	42167	590	38320

Table 7: Recall, precision, and *F*-score for both CNN-MCL and CNN.

Attack name	No. of attacks	Recall		Precision		*F*-score	
		CNN-MCL	CNN	CNN-MCL	CNN	CNN-MCL (%)	CNN (%)
Bot	1956	95.19	95.47	55.83	53.67	70.38	68.71
DDoS	128025	99.19	99.80	58.44	57.75	73.55	73.16
DoS GoldenEye	10293	98.71	98.88	58.19	57.37	73.22	72.61
DoS HULK	230124	99.10	99.14	58.64	58.5	73.68	73.58
DoS Slowhttptest	5499	91.50	94.10	56.43	55.06	69.81	69.47
DoS Slowloris	5796	97.96	98.21	57.3	52.91	72.30	68.77
FTP-Patator	7935	99.77	99.94	50.16	50.16	66.80	66.76
PortScan	158804	99.86	99.63	57.62	56.77	73.08	72.33
SSH-Patator	5897	98.16	99.81	50.46	50.43	67.01	66.66
Web Attack Brute Force	1507	97.96	98.38	51.32	50.06	67.35	66.35

40% ratio. Full dataset size is used to simulate the real-world attack scenarios because the missing values are removed from the dataset during data reading. The number of train records is 667866, and number of test records are 445244.

Figure 10 shows that the standard CNN obtained the poorest results in terms of precision, *F*-score, and accuracy. Meanwhile, highest recall at 99.30 was obtained using the C-CNN, and the proposed CNN-MCL obtained close value recall at 99.15. However, the CNN-MCL model outperformed CNN and C-CNN models in terms of precision, *F*-score, and accuracy at 99.76, 99.46, and 99.46, respectively. Although the improvement was slight with the CNN variants, CNN-MCL performed better than other CNNs for IDS.

Moreover, Figure 11 displays the false alarm rate for standard CNN, C-CNN, and the developed CNN-MCL. Standard CNN obtained FAR of 0.75, and approximate false rate of 0.71 was obtained by the C-CNN. The proposed CNN-MCL obtained significantly better FAR at 0.23.

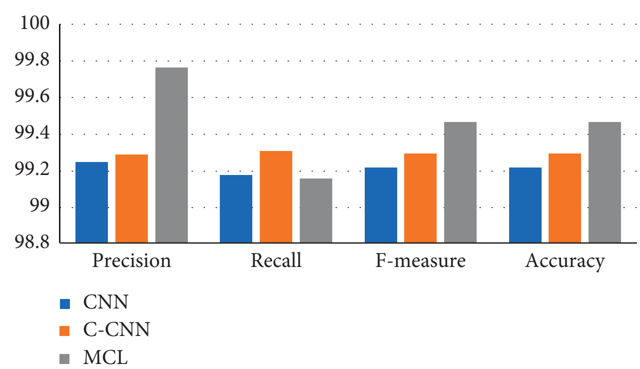


Figure 10: Evaluation measures for CNN, C-CNN, and CNN-MCL.

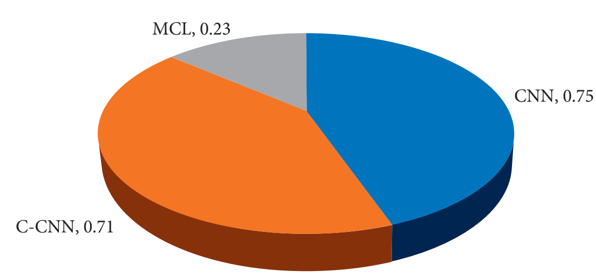


Figure 11: False alarm rate for CNN, C-CNN, and CNN-MCL.

5.4. Structural Design. The structural design of a deep learning model has impacts on the final result of its detection. We present the results of experiments to fit the structural design of the proposed architecture although the objective of this paper is not introducing the best structure for the IDS model. Therefore, we ran several sets of examinations to find suitable architecture for the proposed model. We randomly chose 200 k of test and train data, randomly split to 70 : 30% for train and test, and finally trained models by only 20 iterations. We chose the training accuracy, testing accuracy, average training loss, and average testing loss for the comparison. Furthermore, we believe that metrics can be utilized because of the randomly chosen balanced dataset for this part of the experiments.

We present the experimental results of choosing the different values for filter output size, batch size, and kernel size (filter size) in Tables 8–10, respectively.

Regarding Tables 8–10, the best structure for the proposed model is selected, where the filter output size for layer 1, layer 2, and layer 3 is 32, 18, and 8, respectively; the best batch number is 128; and for the kernels, the best sizes are

Table 8: Accuracy and loss results for filter output size with different values.

			Train		Test	
			Accuracy	Loss	Accuracy	Loss
Filters output size	Layer 1	8	99.4658	0.0221	99.2900	0.0295
		16	99.3555	0.0248	99.1999	0.0303
		32	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		64	99.4338	0.0213	99.0499	0.0335
	Layer 2	8	99.3841	0.024	98.9799	0.0398
		16	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		32	99.3878	0.0241	99.2999	0.0276
		64	99.3588	0.0248	99.0400	0.0346
	Layer 3	**8**	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		16	99.4145	0.0220	99.2399	0.0282
		32	99.4985	0.0215	99.3499	0.0279
		64	99.3401	0.0332	99.2399	0.0304

Table 9: Accuracy and loss results for batch size with different values.

		Train		Test	
		Accuracy	Loss	Accuracy	Loss
Batch size	16	99.375	0.0289	99.2200	0.0294
	32	99.1897	0.0315	99.1999	0.0311
	64	99.0372	0.0352	99.0000	0.035
	128	**99.6829**	**0.0126**	**99.6571**	**0.0137**
	256	99.3292	0.0255	99.1699	0.0325

Table 10: Accuracy and loss results for kernel size with different values.

			Train		Test	
			Accuracy	Loss	Accuracy	Loss
Kernel size	Kernel 1	**3 × 3**	99.4578	0.0212	99.3499	0.0257
		5 × 5	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		7 × 7	99.4065	0.0229	99.2502	0.0279
	Kernel 2	**3 × 3**	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		5 × 5	99.3851	0.0241	99.2931	0.0282
		7 × 7	99.4111	0.0216	99.2038	0.0306
	Kernel 3	**3 × 3**	**99.6829**	**0.0126**	**99.6571**	**0.0137**
		5 × 5	99.4862	0.02	99.3176	0.0257
		7 × 7	99.4505	0.0219	99.1893	0.0337

5×5, 3×3, and 3×3 for kernel 1 (in layer 1), kernel 2 (in layer 2), and kernel 3 (in layer 3).

6. Conclusion

A new deep learning-based approach was suggested in this paper for developing an intrusion detection system. As compared to general CNN which depends on content features, the proposed CNN-MCL can suppress flow content as well as adapt to learn variation detection features from data directly. For this, a new type of layer known as a mean convolutional layer was created which could help the CNN in learning prediction error filters that generate low-level general abnormal features. This layer was used for designing a new CNN architecture which can identify anomaly accurately in the traffic flow. Numerous experiments were conducted for evaluating the proposed CNN-MCL model's ability in performing intrusion detection. The experiments' findings showed that it is possible to train the CNN-MCL so that it can accurately identify normal as well as abnormal flows along with attack types of unknown attacks. For further examining the constrained CNN's performance, it was compared to well-known machine learning methods which are the best detectors at present. According to the comparison, the proposed CNN-MCL architecture is able to detect the anomaly accurately, especially when large-scale training data are used. Hence, the experimental results suggest that the CNN-MCL is able to accurately identify anomalies even in case of no manual feature extraction and unbalanced training data.

Data Availability

The dataset (CICD2017) can be downloaded from https://www.unb.ca/cic/datasets/ids-2017.html. The code will be published on GitHub after paper got published.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Authors' Contributions

All authors were involved in drafting the article or revising it critically for important intellectual content, and all authors approved the final version to be published.

Acknowledgments

This research was partially supported by the Shiraz International School (http://www.shirazschool.com), Shiraz, Iran.

References

[1] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: a review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994–12000, 2009.

[2] J. Zhang, H. Li, Q. Gao, H. Wang, and Y. Luo, "Detecting anomalies from big network traffic data using an adaptive detection approach," *Information Sciences*, vol. 318, no. 11, pp. 91–110, 2015.

[3] G. Fernandes, L. F. Carvalho, J. J. P. C. Rodrigues, and M. L. Proença, "Network anomaly detection using IP flows with principal component analysis and ant colony optimization," *Journal of Network and Computer Applications*, vol. 64, pp. 1–11, 2016.

[4] S. Naseer, Y. Saleem, S. Khalid et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[5] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[6] S. Vieira, W. H. L. Pinaya, and A. Mechelli, "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: methods and applications," *Neuroscience & Biobehavioral Reviews*, vol. 74, pp. 58–75, 2017.

[7] A. Hijazi, E. A. El Safadi, and J. M. Flaus, "A deep learning approach for intrusion detection system in industry network," *CEUR Workshop Proceedings*, vol. 2343, pp. 55–62, 2018.

[8] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.

[9] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

[10] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019.

[11] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing*, pp. 313–316, Jeju, Korea, 2017.

[12] M. Erza and K. Kim, "Deep learning in intrusion detection System : an overview," in *Proceedings of the 2016 International Research Conference on Engineering and Technology (2016 IRCET)*, pp. 1–12, Seoul, South Korea, 2016.

[13] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP 2018-Proceedings of the 4th International Conference on Information Systems Security and Privacy*, pp. 108–116, Funchal-Madeira, Portugal, January 2018.

[14] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, New York, NY, USA, December 2016.

[15] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, pp. 258–263, Morocco, October 2016.

[16] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

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

[18] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[19] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016*, pp. 195–200, Anaheim, CA, USA, December 2016.

[20] C. G. Cordero, S. Hauke, M. Muhlhauser, and M. Fischer, "Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks," in *Proceedings of the 2016 14th Annual Conference on Privacy, Security and Trust, PST 2016*, pp. 317–324, Auckland, New Zealand, December 2016.

[21] M. J. Kang and J. W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *Proceedings of the 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, Nanjing, China, May 2016.

[22] L. Bontemps, V. L. Cao, J. McDermott, and N. A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 141–152, Springer, Berlin, Germany, 2016.

[23] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[24] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[25] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.

[26] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the International Conference on Information Networking*, pp. 712–717, Taipei, Taiwan, March 2017.
[27] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.
[28] W. F. Zheng, "Intrusion detection based on convolutional neural network," in *Proceedings - 2020 International Conference on Computer Engineering and Application, ICCEA 2020*, pp. 273–277, Guangzhou, China, March 2020.
[29] R. U. Khan, X. Zhang, M. Alazab, and R. Kumar, "An improved convolutional neural network model for intrusion detection in networks," in *Proceedings - 2019 Cybersecurity and Cyberforensics Conference, CCC 2019*, pp. 74–77, Melbourne, Australia, May 2019.
[30] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.
[31] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, http://arxiv.org/abs/1207.0580.
[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 818–833, Springer, Berlin, Germany, 2014.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference of Learning Represent. ICLR 2015 - Conference Track Proceedings*, San Diego, CA, USA, May 2015.
[34] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop in neural networks: tricks of the trade," *Lecture Notes in Computer Science*, p. 111, Springer, Berlin, Germany, 1998.
[35] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *Proceedings of the COLT 2010 - 23rd Conference Learning Theory*, pp. 257–269, Haifa, Israel, July 2010.
[36] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," 2012.
[37] "Gnats V. Mosquitoes," *Notes Queries*, vol. s4-VII, no. 176, p. 416, 1871.
[38] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proceedings of the 2016 International Conference on Information Science and Security (ICISS)*, pp. 1–6, Jaipur, India, December 2016.
[39] J. D. Hunter, "Matplotlib: a 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 99–104, 2007.
[40] L. Mohammadpour, T. C. Ling, C. S. Liew, and C. Y. Chong, "A convolutional neural network for network intrusion detection system," in *Proceedings of the 46th Asia-Pacific Advanced Network Meeting*, pp. 50–55, Auckland, New Zealand, August 2018.
[41] B. Bayar and M. Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Electronic Imaging*, vol. 2017, no. 7, 86 pages, 2017.
[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, and D. Anguelov, "Going deeper with convolutions," 2014, http://arxiv.org/abs/1409.4842.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, Santiago, Chile, December 2015.
[45] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[46] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proceedings of the 4th International Conference of Learning Representations ICLR 2016 - Conference Track Proceedings*, San Juan, Puerto Rico, May 2016.
[47] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference of the Machine Learning ICML 2015*, pp. 448–456, Lille, France, July 2015.
[48] F. Pedregosa, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[49] M. Abadi, "TensorFlow: large-scale machine learning on heterogeneous distributed systems," 2016, http://arxiv.org/abs/1603.04467.
[50] S. Suthaharan, "Big data classification," *ACM Sigmetrics Performance Evaluation Review*, vol. 41, no. 4, pp. 70–73, 2014.
[51] B. Bayar and M. C. Stamm, "Constrained convolutional neural networks: a new approach towards general purpose image manipulation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018.