SEYBOLD Report
ISSN: 1533 - 9211
https://seyboldreport.net/

AN ENHANCED CONVOLUTIONAL NEURAL NETWORK BASED CLASSIFICATION METHOD FOR CYBER THREAT DETECTION

T.Elangovan
Ph.D. Research Scholar, Dept. of Computer Science Erode Arts and Science College, Erode-638 009, Tamilnadu, India. E-mail: elangovaneasc@gmail.com

Dr.V.Sheshathri
Assistant Professor, Department of Computer Applications, Erode Arts and Science College, Erode-638 009, Tamilnadu, India. E-mail: sheshathrieac@gmail.com

Dr.S.Sukumaran
Associate Professor in Computer Science, Erode Arts and Science College, Erode-638 009, Tamilnadu, India. E-mail: prof_sukumar@yahoo.co.in

ABSTRACT
Objectives: Cyber security is the provision of an effective cyber threats detection technique based on deep neural networks. Cyber attacks is highly challenging to protect the systems against threats and malicious behaviors in networks. Methods/Statistical Analysis: The proposed technique input data first preprocessed, that initial data format convert into image format. Then data normalization process eliminates the different dimensional data. Then mean convolution layer and convolutional neural network based layer to find the cyber threats from normal data. This method recommends a new Enhanced Convolutional Neural Network architecture type known as Mean Convolution Layer (ECNN-MCL) developed for learning the anomalies content features and then identifying the particular abnormality. Findings: The recommended ECNN-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. The system emphases on discriminating between true positive and false positive alerts, thus helping security analysts to fast respond to cyber threats. Applications/Improvements: The experiment in this method is performed using NSLKDD data set. The performance comparison is compared with proposed ECNN-MCL and existing methods of SVM, k-NN, and CNN. The experimental results ensure that our proposed method is capable of being employed as learning-based model for network intrusion-detection. Finally, the experimentation results are shown through Mat Lab R2013a.
Index Terms: Intrusion Detection, Deep Neural Networks and Cyber Threat Detection, ECNN-MCL, Image Format, Normalization,

1. INTRODUCTION
With the emergence of Artificial Intelligence (AI) techniques, learning-based approaches for detecting cyber attacks, have become further improved, and they have achieved significant results in many studies [1]. However, owing to constantly evolving cyber attacks, it

is still highly challenging to protect IT systems against threats and malicious behaviors in networks. Because of various network intrusions and malicious activities, effective defenses and security considerations were given high priority for finding reliable solutions [2]. Traditionally, there are two primary systems for detecting cyber-threats and network intrusions. An Intrusion Prevention System (IPS) is installed in the enterprise network, and can examine the network protocols and flows with signature based methods primarily. It generates appropriate intrusion alerts, called the security events, and reports the generating alerts to another system. The Security Information and Event Management (SIEM) have been focusing on collecting and managing the alerts of IPSs.

The SIEM is the most common and dependable solution among various security operations solutions to analyze the collected security events and logs [4]. Moreover, security analysts make an effort to investigate suspicious alerts by policies and threshold, and to discover malicious behavior by analyzing correlations among events, using knowledge related to attacks. However, it is quiet challenging to recognize and detect intrusions compared to intelligent network attacks owing to their high false alerts and the huge amount of security data. Hence, the most recent studies in the field of intrusion detection have given increased focus to machine learning and artificial intelligence techniques for detecting attacks [5]. Advancement in AI fields can facilitate the investigation of network intrusions by security analysts in a timely and automated manner. These learning-based approaches require to learning the attack model from historical threat data and use trained models to detect intrusions for unknown cyber threats.

Convolutional Neural Network

A CNN is a deep learning architecture that has shown achievement in computer vision and image processing applications [6]. A CNN typically consists of a number of alternate layers, as well as convolution, pooling, and fully connected layers.

Convolution Layer

Convolution refers to the application of K filters also called as kernels. Filter is used to arrest features such as corners and edges in images. A filter is a grid of size $N\,X\,N\,X\,C$, where N is the height and width of the filter, and C is the number of channels in the input image. Convolution is applied such that each filter slides over all pixels in the input image [16]. The pixel value is multiplied by the corresponding value in filter. The multiplication results are calculated to produce a single number. The result of each convolution is called a feature map. Thus, applying K convolutional filters yields a set of K feature maps, each of size ((N-F)/stride) +1, where stride is the number of pixels the filter is moved across in one step.

Pooling Layer

Pooling is an operation that is used to decrease the spatial size of an image. The pooling layer is applied on each feature map autonomously. The most common type of pooling is *max*-pooling.

Fully Connected Layer

A fully connected layer is usually used at the end of the architecture for the purpose of classification. It is collected of several neurons, where each neuron is connected to all neurons

in the former layers. Fully connected layers deliver the final nonlinear combinations of features.

2. LITERATURE REVIEW

A Self-Taught Learning (STL) deep learning model for network intrusion detection proposed by Javaid et al. [7]. The first component of the model was the unsupervised feature learning, in which a sparse auto encoder used to obtain feature representation from a large unlabeled data set. Then, the second component is an ANN classifier which used Softmax regression classification. Using the NSL-KDD data set, the model obtained accuracy values of 88.39% and 79.10% for two-class and five-class classification, respectively.

Yin et al. [17] proposed a model for intrusion detection using recurrent neural networks (RNNs). RNNs are especially suited to data that are time dependent. This model contained of forward and back propagation stages. Forward propagation calculates the output values, although back propagation passes residuals accumulated to update the weights. The model contained of 20 hidden nodes, with Sigmoid as the activation function and Softmax as the classification function. Performance results using the NSL-KDD data set presented the accuracy values was 83.28% and 81.29% for binary and multiclass classification.

Deep learning and traditional machine learning techniques can be hybridized to increase intrusion detection accuracy. A combination of sparse auto encoder and SVM was proposed by Al-Qatf et al [1]. The sparse auto encoder used to capture the input training data set, whereas the SVM used to build the classification model. This model trained and evaluated using the NSL-KDD data set. The obtained accuracy values were 84.96% and 80.48% for two-class and five-class classification, respectively.

Altwaijry et al. [2] developed an intrusion detection model using DNN. The proposed model consisted of four hidden fully connected layers and trained using NSL-KDD data set. The DNN model acquired accuracy values of 84.70% and 77.55% for the two class and five-class classification difficulties. The proposed model outperformed traditional machine learning algorithms, including NB, Bagging, and Adaboost in terms of accuracy and recall.

Wang et al. [14] suggested a Hierarchical Spatial and Temporal features-based Intrusion Detection System (HAST-IDS) that automatically learns network traffic features. The main idea is that the spatial features of network traffic are first learned using deep CNNs and then learns the temporal features are learned LSTM networks. The experiments were conducted by DARPA and ISCX datasets.

Vinayakumar et al. [13] developed a hybrid intrusion detection system which has the capability to analyze the network and host-level activities. It employed distributed deep learning model with DNN for processing and analyzing very large scale data in real-time. The DNN model selected by comprehensively evaluating their performance in comparison to classical machine learning classifiers on various benchmark IDS datasets such as NSLKDD and UNSW-NB15.

Khan et al. [8] suggested a novel two-stage deep learning model, based on a stacked auto-encoder with a soft-max classifier, for efficient network intrusion detection. The experiments on two public datasets: the benchmark KDD99 and UNSW-NB15 datasets. This study achieved results, up to 89.1% for the UNSW-NB15 dataset.

Du et al. [3] developed a new algorithm based on the k-NN classifier method using TF-IDF for modeling program behavior in intrusion detection regarding system calls. With the k-NN classifier, the frequencies of system calls are used to describe the program behavior. The text categorization methods, such as TF-IDF, are approved to transform every system call data to a vector and measure the relationship between two program system call activities. The TF-IDF-based k-NN classifier performs to be well appropriate to the domain of intrusion detection in the field of malware detection.

3. PROPOSED METHODOLOGY

3.1 Data Preprocessing

The first step of data preprocessing is convert the initial data format into image format. The NSL-KDD dataset has four symbolic data types: the protocol_type feature, flag feature, service feature and attack label. The protocol_type feature has three categories: TCP, UDP and ICMP, which could be mapped into three three-dimensional binary vectors (1, 0, 0), (0, 1, 0), (0, 0, 1). Therefore, one protocol_type feature fits into three features, one flag feature turns into 11 features, and one service feature variations into 70 features. Therefore, the 41 initial features change into 122 features. Especially, the attack label should be classified into four types, DOS, Probing, R2L and U2R, before using the one-hot encoder.

Data normalization can eliminate the differences between different dimensional data and is therefore widely used in computing. As the 122 features of the NSL-KDD dataset have different codomains, in order to ensure the reliability of the training result normalize these features into the range [0, 1]. Apply the min-max normalization to address the data.

$$x_{i2} = \frac{x_{i1} - x_{min}}{x_{max} - x_{min}} \quad \ldots (1)$$

where x_{i1} represents the initial data i of the feature, x_{max} represents the maximum data of the feature, x_{min} represents the minimum data of the feature, and x_{i2} represents the data after normalizing x_{i1}.

The initial data format is a 1∗122 dimension vector. The converting data format as described in the first step, the feature set includes 122 dimensions. Therefore, convert the 1∗122 vector into n ∗n image data. When more actual data are used, the result of the experiment is more accurate, and n should achieve the maximum value. As 121=11∗11, the optimal result is ‘‘n=11’’. Therefore, the 122-dimension feature set should remove a feature. When we remove a feature, usually calculate the correlation or dispersion of features. A suitable neutral vector variable decorrelation approaches with the serial nonlinear transformation and parallel nonlinear transformation. Then filter the features preliminarily at this step, use CNN to select features further. Choose the removed feature, and the technique is Coefficient of Variance (CV). The function of CV is defined as follows:

$$CV = \frac{\sigma}{\mu} \quad \ldots (2)$$

Where σ is the standard deviation of the feature, μ is the average of the feature, and CV is the CV of the feature.

3.2 Enhanced CNN-Mean Convolutional Layer (ECNN-MCL)

The CNN-based layer to separate anomalies from normal data and this method used

involves using the data to directly learn changes occurring through abnormal data. The foremost issue was that normal and abnormal flows are not greatly different, and so the CNN must be compulsory to detect abnormalities variations. It is seen that if standard form CNNs are used to detect, features are learned which represent an images content, primarily normal flow, meaning that the classifier identifying data content is linked with training data instead of learning data variations. The approach is 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 (ECNN-MCL), established for use with intrusion detection system tasks. These errors are employed as low-level abnormal/normal features, where further advanced abnormal detection features are generated thereafter. In order to echo these actions, the recommended layer aims to fully learn prediction error filters. The feature maps created are then linked with prediction error fields employed as low-level abnormal traces. The ECNN-MCL is able to be positioned differently from the CNN aimed to undertake IDS tasks. This act 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 ECNN-MCL, where L denotes the L^{th} ECNN-MCL, the subscript k describes the k^{th} convolutional filter within a layer, and that the central value of a convolutional filter is well-defined by (cx, cy). The CNN is then forced to learn prediction error filters through actively implementing specific constraints:

$$\mu^{(L)} = Mean\left(w^{(L)}(c_x, c_y)\right), \quad \ldots (3)$$

$$\{w_k^{(L)}(x,y) = \frac{w_k^{(L)} \times \mu^{(L)}}{\sum \quad w_k^{(L)}}, \quad (x,y) \neq (c_x, c_y), w_k^{(L)}(x,y) = -\mu^{(L)}, \quad (x,y) = (c_x, c_y)$$

Predictions of ECNN-MCL are established in a specific training process. Following this, updates of filter weights $w_k^{(L)}$ at each iterations are made, with the Adam algorithm in the back propagation stage. Then, the updated filter weights are set into the feasible set of prediction error filters by ECNN-MCL reinforcement, and projection is undertaken at every training iteration. This is attained 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. There are two steps to this process. Firstly, the remaining weights are multiplied with the mean value, and then, the collected weights are dividing in contradiction of 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. To provide intuition into this, suppose the prediction is formed by using some function f(X) to predict normality or abnormality of input data. 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 $\left(g(I)\right)$. For simplicity, assume a network with one ECNN-MCL and one CNN in the feature extraction section. Regarding equation (1), using conventional CNN, the classifier generates the output based on the following equation:

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

where f is the classifier function, g is the feature extraction process, I is the input data, $*$ is the 2d convolution, k is the number of channels, 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:

$$f(X) = f\big(g(I)\big) = f\big(\sum_{k=1}^{K} \quad I * \widetilde{w}_{kj} + b_{kj}\big) \qquad \dots (5)$$

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

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

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

It can be seen from equation (7) that the Mean $(w^{(L)}(c_x, c_y))$ is calculated from all channels because it does not have the subscript k. Consequently, the mean value is shared among all channels. Then, for each individual channel, the weights are normalized using $w_k^{(L)}/ w_k^{(L)}$. These operations decrease the effect of usual context to be extracted as useful features, and they are progressing the outcome of abnormal variations to be deliberated as the extracted features. To show the advantage of ECNN-MCL in the learning process compared to standard CNN, and evaluated a simple model using a CNN layer and a classifier. Our goal is to visualize the difference between the extracted features from the ECNN-MCL and conventional CNN. Therefore, for this objective, generate a simple dataset to control the location and value of anomalies in dataset. However, Experimental have evaluated our proposed ECNN-MCL with a real-world dataset.

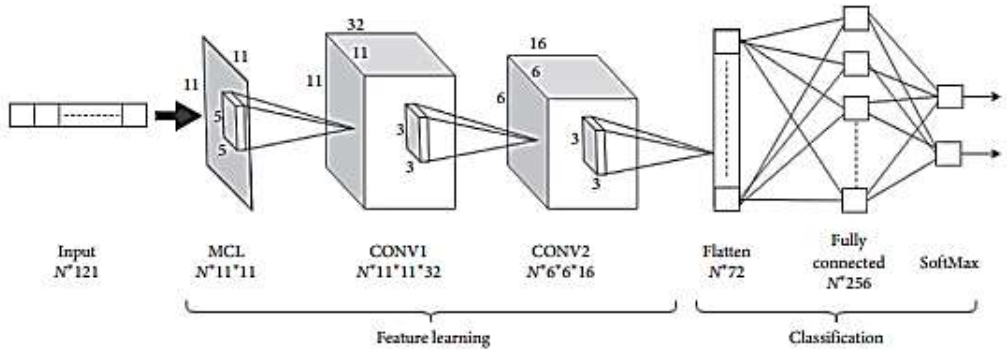

Figure 1. ECNN-MCL Architecture

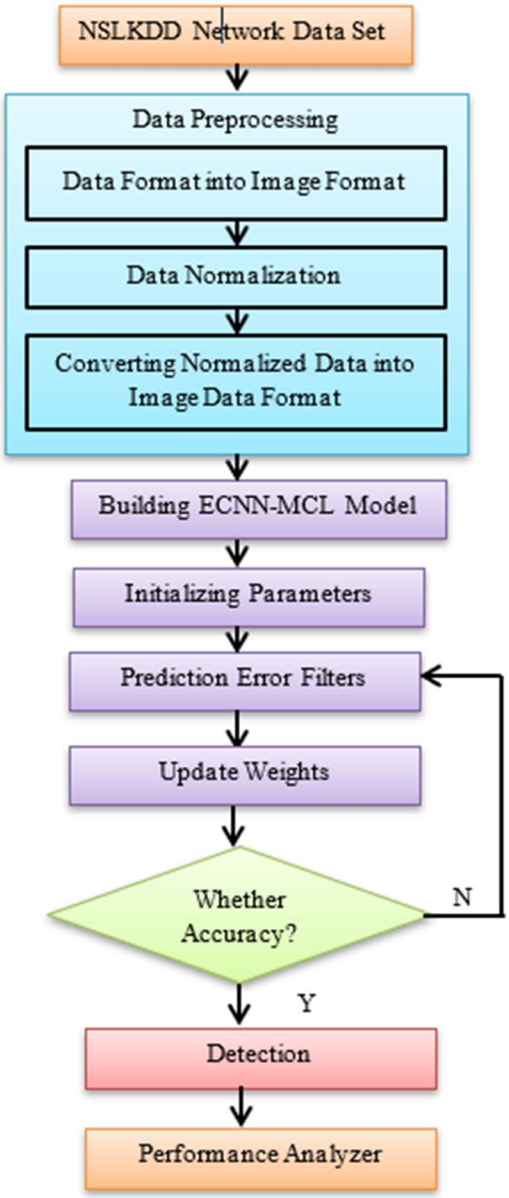

Figure 2. Block Diagram of Proposed ECNN-MCL Method

Algorithm

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

Step 1: Start the Process
Step 2: Select the NSLKDD Network Dataset
Step 3: Perform Data preprocessing
Step 4: Transfer data format into Image Format
Step 5: Apply min-max Normalization
Step 6: Converting normalized Data into Image Data Format
Step 7: Transfer image data into matrix format whose size is 11 * 11
Step 8: Initialize w_k' using randomly draw n weights
    i=1
    While i ≤ Maximum_Iteration do
        Do feed forward pass
        Update forever weights through Adam and back propagation errors
        Set the $\mu^{(L)}$ = mean of central points of all filters of layer L
        Update the weights of layer L using the $w_k^{(L)}(x,y) = ((\frac{w_k^{(L)} * \mu^{(L)}}{t}) \sum \quad w_k^{(L)})$
        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
Step 9: Classified Data.

4. EXPERIMENTS AND RESULTS

Data Set

The Network Security Laboratory - Knowledge Discovery in Databases (NSL-KDD) data set is an improved version of the KDD'99 data set. It is a minor data set that delivers best evaluation of classifiers since redundant records are removed. Redundant records cause learning classifiers to be biased toward the more frequent records during training, as well as increasing classification accuracy whenever these same records appear in the test set. The training set $KDDTrain^+$ contains 125,973 records, and the testing set $KDDTest^+$ contains 22,544 records.

The data set simulates the following types of attacks.

1. DoS: where an attacker attempts to make some resource too demanding to handle valid requests or denies legal users access to a machine.

2. Probing Attack: in which an attacker attempts to collect data on a network of computers to find a way around an obstacle to violate the network's security controls.

3. U2R (User to Root Attack) : an attack in which an attacker gains access as a normal user account on the system, then searches to find any vulnerability to exploit and gain root access to the system.

4. R2L (Remote to Local Attack): send packets to a machine via a network, where the attacker does not have an account on that machine, to exploit some breaches to make local access as a user of that machine.

The data set has mutually normal and anomaly traffic. Anomaly traffic is in one of four categories: DoS, Probe, R2L, and U2R. The number of instances and attribute names of different attack classes in the NSL-KDD data set are shown in Table 1.

Table 1. Distribution of attacks in the NSL-KDD Data set

Attack	**Training Set Testing Set**	**Attribute**
Dos	45,927	back, land, teardrop, neptune, pod, smurf
	7,458	back, land, teardrop, neptune, pod, smurf, udpstorm, apache2, processtable, worm, mailbomb
Probing Attack	11,656	ipsweep, nmap, portsweep, satan
	2,421	ipsweep, nmap, portsweep, satan, mscan, saint
U2R	52	loadmodule, buffer-overflow, perl, rootkit
	200	buffer-overflow, loadmodule, perl, rootkit, sqlattack, xterm, ps
R2L	995	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster
	2,754	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named

Metrics

For the performance evaluation, four metrics are adopted: Accuracy, TPR, FPR, and F-Measure, which are all commonly used for learning-based methods in the field of cyber threat detection. TPR is used to estimate the systems performance with respect to its threat detection. FPR is used to evaluate misclassifications of normal data. F-measure is the harmonic mean of the precision and TPR (recall), where Precision = TP / (TP+FP) is the percentage of true attacks among all attacks classified. The explanations for accuracy, TPR, FPR, and F-measure are offered below:

$$TPR\ (Recall) = \frac{TP}{TP + FN} \quad \ldots (8)$$

$$FPR = \frac{FP}{TN + FP} \quad \ldots (9)$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad \ldots (10)$$

$$F - measure = 2 * \frac{Precison \cdot Recall}{Precision + Recall} \quad \ldots (11)$$

Table 2. Performance Comparison Table

Metrics	TPR	FPR	Accuracy	F-Measure
SVM	0.819	0.035	0.897	0.881
k-NN	0.905	0.088	0.909	0.903
CNN	0.909	0.045	0.920	0.912
ECNN-MCL	0.919	0.025	0.950	0.943

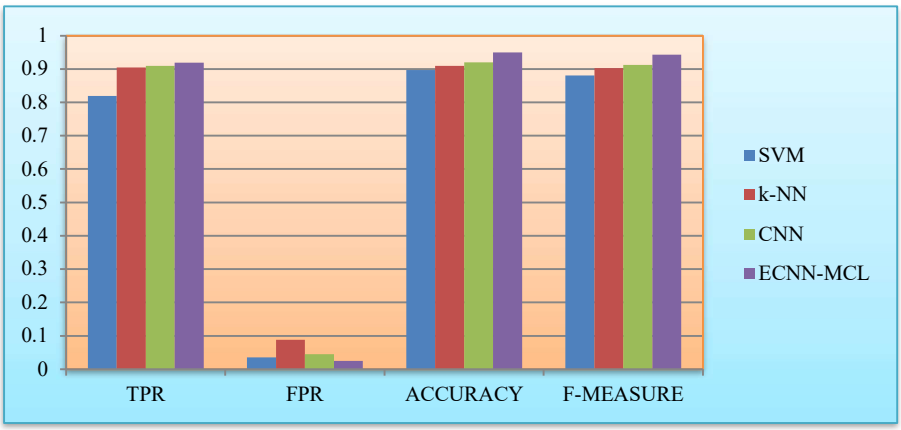

Figure 3. Comparison Graph for Proposed ECNN-MCL

5. CONCLUSION

A new deep learning-based approach suggested in this paper for developing an cyber threat detection system. As compared to general CNN which depends on content features, the proposed ECNN-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 created which could help the CNN in learning prediction error filters that generate low-level general abnormal features. This layer used for designing a new CNN architecture which can identify anomaly accurately in the traffic flow. Numerous experiments conducted for evaluating the proposed ECNNMCL models ability in performing cyber threat detection. The experiments findings showed that it is possible to train the ECNN-MCL so that it can accurately identify normal as well as abnormal flows along with attack types of unknown attacks. For examining the constrained CNN's performance, it compared to well-known method which is the best detector. According to the comparison, the proposed ECNN-MCL is able to detect the anomaly accurately, especially when large-scale training data are used. The experimental results suggest that the ECNN-MCL is able to accurately identify the cyber threats.

REFERENCES

[1] Al-Qatf M, Lasheng Y, Al-Habib M and K Al-Sabahi, "Deep Learning Approach Combining Sparse Auto Encoder with SVM for Network Intrusion Detection", IEEE Access, Vol. 6, Pp. 52843–52856, 2018.

[2] Altwaijry N, Alqahtani A, and Al-Turaiki I. "A Deep Learning Approach for Anomaly-Based Network Intrusion Detection", First International Conference on Big Data and Security, Nanjing, China: Springer, 2019.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning", Proceedings in ACM CCS, Vol. 17, Pp. 1285–1298, Nov. 2017.

[4] 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, Pp. 1–12, 2016.

[5] 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.

[6] 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.

[7] Javaid A, Niyaz Q, Sun W, and Alam M. "A Deep Learning Approach for Network Intrusion Detection System", 9th EAI International Conference on Bio-inspired Information and Communications Technologies, New York, NY: ICST (Institute for Computer Sciences, Social-Informatics), Pp. 21–26, 2016.

[8] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A Novel Two Stage Deep Learning Model for Efficient Network Intrusion Detection", IEEE Access, Vol. 7, Pp. 30373–30385, 2019.

[9] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of Intrusion Detection using Deep Neural Network," IEEE International Conference on Big Data and Smart Computing, pp. 313–316, 2017.

[10] S. Naseer, Y. Saleem, S. Khalid M. K. Bashir1, J. Han, M. M. Iqbal and K. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks", IEEE Access, Vol. 6, Pp. 48231–48246, 2018.

[11] R.Sankarasubramanian and Dr.S.Sukumaran "Encryption and Decryption Techniques using MBECC for Image Data Transfer", International Journal of Innovative Technology and Creative Engineering, ISSN 2045-8711, Vol. 8, No. 02, Pp. 466-471, 2018.

[12] 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 & Bio behavioral Reviews, Vol. 74, Pp. 58–75, 2017.

[13] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System", IEEE Access, Vol. 7, Pp. 41525–41550, 2019.

[14] W. Wang, Y. Sheng, and J. Wang, "HAST-IDS: Learning Hierarchical Spatial-Temporal Features using Deep Neural Networks to Improve Intrusion Detection", IEEE Access, Vol. 6, Pp. 1792–1806, 2018.

[15] 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.

[16] H. Yang and F. Wang, "Wireless Network Intrusion Detection based on Improved Convolutional Neural Network", IEEE Access, Vol. 7, Pp. 64366–64374, 2019.

[17] Yin C, Zhu Y, Fei J and He X, "A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks", IEEE Access, Vol. 5, Pp. 21954–21961, 2017.