Securing the IoT: An Efficient Intrusion Detection System Using Convolutional Network

Harith Yas[1], Manal M. Nasir [2,*]

[1]Faculty of Management, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia

[2]Gwinnett Technical College, 5150 Sugarloaf Pkwy, Lawrenceville, GA 30043, USA

Emails: Harith.albayati@yahoo.com ; mnasir@gwinnetttech.edu

Abstract

The Internet of Things (IoT) is an ever-expanding network of interconnected devices that enables various applications, such as smart homes, smart cities, and industrial automation. However, with the proliferation of IoT devices, security risks have increased significantly, making it necessary to develop effective intrusion detection systems (IDS) for IoT networks. In this paper, we propose an efficient IDS for complex IoT environments based on convolutional neural networks (CNNs). Our approach uses IoT traffics as input to our CNN architecture to capture representational knowledge required to discriminate different forms of attacks. Our system achieves high accuracy and low false positive rates, even in the presence of complex and dynamic network traffic patterns. We evaluate the performance of our system using public datasets and compare it with other cutting-edge IDS approaches. Our results show that the proposed system outperforms the other approaches in terms of accuracy and false positive rates. The proposed IDS can enhance the security of IoT networks and protect them against various types of cyber-attacks.

Keywords: IoT; Intrusion Detection; Convolutional Network; Secure IoT Systems

1. Introduction

The Internet of Things (IoT) is a rapidly growing network of interconnected devices that are embedded with sensors, software, and other technologies, allowing them to collect and exchange data with each other and with other systems over the Internet. The IoT has the potential to revolutionize various industries, including healthcare, transportation, and manufacturing, by enabling real-time monitoring, automation, and optimization of processes. However, the widespread adoption of IoT devices has also brought new security challenges. IoT devices are often designed with limited computing power and memory, which makes it difficult to implement robust security measures. Furthermore, IoT devices are often deployed in uncontrolled environments, making them vulnerable to various types of attacks, such as malware, denial-of-service (DoS), and man-in-the-middle (MiTM) attacks.

These security vulnerabilities pose a significant threat to the integrity, confidentiality, and availability of the IoT infrastructure and the data it handles. For instance, an attacker who gains access to an IoT device can use it as a foothold to launch further attacks on other devices or systems. Furthermore, IoT devices often collect sensitive data, such as personal health information or financial data, making them an attractive target for cybercriminals. Moreover, IoT devices are often used in critical infrastructure, such as energy grids and transportation systems, which could lead to significant disruptions and even physical harm if compromised. Therefore, it is crucial to develop effective security mechanisms to protect IoT devices and the data they handle.

DOI: https://doi.org/10.54216/JCIM.010105

Intrusion Detection Systems (IDS) play a crucial role in addressing security vulnerabilities in IoT environments. IDS can monitor network traffic and identify suspicious activities, such as unauthorized access attempts, malware infections, and abnormal traffic patterns. By detecting these security threats, IDS can alert network administrators and take appropriate actions to mitigate the risks. Machine learning (ML) has emerged as a powerful tool for developing more effective and efficient IDS. ML algorithms can analyze large volumes of network traffic and learn from the patterns and anomalies in the data to identify security threats that may be missed by traditional rule-based IDS. ML-based IDS can leverage various types of algorithms, including supervised, unsupervised, and semi-supervised learning, to detect and classify security threats. ML-based IDS can also adapt to the changing threat landscape by continuously learning from new data and updating the detection models. ML algorithms can also be used to optimize the IDS performance by reducing false positives and false negatives, improving the accuracy and reliability of the system.

This study contributes to securing the IoT infrastructure by developing a novel ML-based IDS that adapts and applies a convolutional neural network to learn attacking patterns from streams of IoT traffic. Our system alleviates the need for complex feature engineering, as the usual ML model. Extensive evaluation of public datasets demonstrated valuable insights about the performance of our system, and its opportunity to secure real IoT systems.

2. Related works

The literature on ML-based IDS has gained significant attention in recent years due to the increasing sophistication of cyber-attacks. Many researchers have proposed various ML-based IDS architectures to detect various types of network intrusions effectively. For instance, in [1], Zhou et al. developed a novel approach for building an efficient IDS using feature selection and ensemble classification techniques. The feature selection method used a combination of correlation-based feature selection (CFS) and principal component analysis (PCA) to select the most relevant features for the IDS. The ensemble classification method was designed to combine the strengths of different classification algorithms, such as decision trees, k-nearest neighbor, and support vector machines, to improve the accuracy and reliability of the IDS. In [2], Almseidin et al evaluated and compared the performance of various of six different machine learning algorithms for building IDS, including Naïve Bayes (NB), Decision Trees (DT), Random Forests (RF), Support Vector Machines (SVM), K-Nearest Nearest Neighbor (KNN), and Artificial Neural Networks (ANN), using the NSL-KDD dataset. They also discussed the impact of feature selection techniques, such as Principal Component Analysis (PCA) and Mutual Information (MI), on the performance of the IDS. In [3], Wang et al proposed an IDS framework based on Support Vector Machines (SVM) and a feature augmentation technique that combined the original features of network traffic data with the domain knowledge features, such as packet size, protocol type, and destination port, to enrich the feature space and improve the discrimination ability of the IDS. In [4], Javaid et al investigated a deep learning (DL) based approach for building an IDS, where a convolutional model was applied to automatically learn the complex patterns in network traffic data without relying on handcrafted features or domain knowledge. Liu and Lang [5] provided an overview of the types of security threats that are commonly encountered in the IoT infrastructure and then highlighted the importance of IDS in detecting these threats. They then presented a detailed analysis of various ML methods that have been used for building IDS, including DT, SVM, ANN, deep belief networks, and convolutional neural networks. Moreover, Vinayakumar et al [6] developed an Intelligent IDS based on the DL model that learned the complex features and patterns in the data through a stack of convolutional and pooling layers. In [13], Bamakan et al proposed an efficient IDS based on the two-stages framework. In the early stage, the raw network traffic data was preprocessed to extract the relevant features using the Minimum Cluster Perceptron algorithm, which reduced the dimensionality of the data and improved the classification accuracy. In a later stage, the SVM classifier is used to classify the preprocessed data into normal and malicious traffic. Ahmad et al. [15] developed a comparative analysis of three different machine learning algorithms, namely SVM, RF, and ELM, for intrusion detection in computer networks. In [18], Lin et al provided an overview of intrusion detection and the various techniques used for it. It then introduces the proposed approach, called CANN (Combining Cluster Centers and Nearest Neighbors). The CANN algorithm consisted of two main steps: clustering and KNN. The clustering step was used to group similar instances into clusters, while the KNN classification step is used to identify the class of each instance based on the nearest neighbor in each cluster. The use of ML algorithms in IDS has shown promising results in improving the detection rate, reducing the false alarm rate, and improving the overall performance of the system. However, some challenges remain, such as the difficulty of achieving high accuracy while minimizing computational

DOI: https://doi.org/10.54216/JCIM.010105

resources, handling unbalanced datasets, and ensuring the robustness of the system against adversarial attacks. DL promising direction to address these gaps, but DL-based IDS is still in its early stage, which motivates the proposal of this work.

3. Methodological Design

Before diving into building our CNN-based, we started applying several data preprocessing steps to get the IoT data ready for training. Since IoT traffic data may contain missing values, duplicates, or other errors. Therefore, data cleaning should be performed to remove or correct these errors. Data cleaning is a process of identifying and correcting or removing errors and inconsistencies in a dataset. given that original data $D = \{x_1, x_2, \ldots, x_n\}$, we clean the data by removing duplicates as follows:

$$D = \{x_1, x_2, \ldots, x_n\} \rightarrow D' = \{x_1', x_2', \ldots, x_{n'}'\} \quad (1)$$

where x_i' represents a unique data point and $n' \leq n$. Thus, we are removing duplicates from the original dataset D and creating a new dataset D' that contains only unique data points. Then, we handle missing values as follows:

$$D' = \{x_1', x_2', \ldots, x_{n'}'\} \rightarrow D'' = \{x_1'', x_2'', \ldots, x_{n''}''\} \quad (2)$$

where x_i'' represents a data point with no missing values and $n'' \leq n'$. This way, we are creating a new dataset D'' that contains only data points with no missing values. This is done by removing data points with missing values. Following this, we perform data normalization to scale the data to have zero mean and unit variance. This can be expressed mathematically as:

$$\tilde{x} = \left\{ \frac{(x_i'' - min(x_i''))}{max(x_i'') - min(x_i'')} \right\}_{i=1}^{n''} \quad (3)$$

At this point, we start building a convolutional model. Let $\tilde{X}$ be the input data of shape (B, T, F_x), where B denotes the number of samples in each batch, T denotes the length of each sequence in the input data, F_x is the number of features in each sequence. The construction of our model is composed of a sequence of convolutional layers, each followed by a pooling layer for down-sampling. The first layer in our model is a 1D convolution that performs a convolution operation on the input data using a set of learnable filters. The output of the convolutional layer can be expressed as:

$$Z^{[1]} = ReLU\left(W_{(3\times1),f}^{[1]} * X + b^{[1]}\right) \quad (4)$$

where $Z^{[1]}$ denote the output of the first layer, $ReLU(\cdot)$ is the rectified linear activation function. $W_{,f}^{[1]}$is the set of f learnable filters of size (3×1). $b^{[1]}$ is the biased term. The output of the convolutional layer is then passed through a max pooling layer, which reduces the dimensionality of the output by taking the maximum value over a sliding window. The output of the max pooling layer can be expressed as:

$$Z^{[2]} = maxpool\left(Z^{[1]}\right) \quad (5)$$

where $Z^{[2]}$ is the output of the max pooling layer.

Additional 1D convolutional and max pooling layers are added to our model to extract more complex features from the input IoT data. The output of each convolutional layer is passed through a rectified linear activation function, and the output of each max pooling layer is obtained by taking the maximum value over a sliding window. The output of the last max pooling layer is flattened and passed through a fully connected layer to perform classification. The output of the fully connected (FC) layer can be expressed as:

$$Z^{[fc]} = ReLU\left(W^{[fc]} \cdot flatten\left(Z^{[K]}\right) + b^{[fc]}\right) \quad (6)$$

Where $Z^{[fc]}$ is the output of the FC layer. $flatten()$ is the function that flattens the output of the last max pooling layer. $W^{[fc]}$ is the weight matrix of the FC layer. $b^{[fc]}$ is the biased term. by the end of our model, the SoftMax probabilities are computed for the last FC layer, as follows:

$$Z = Softmax\left(W \cdot Z^{[fc]} + b\right) \quad (7)$$

DOI: https://doi.org/10.54216/JCIM.010105

Then, cross-entropy is used as a loss function for the and is described as:

$$L = -\frac{1}{C}\sum_{i=1}^{C} y_i \ln Z_i \quad (8)$$

Where C is the number of classes of attacks. Dropout regularization is adopted between layers to avoid overfitting. The code implementation of our model is given as follows:

```
model = Sequential() # initializing model
# input layer and first layer with 32 filters
model.add(Conv1D(32, 3, padding="same",input_shape = (X_train.shape[1], 1), activation='relu'))
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Conv1D(32, 3, padding="same", activation='relu'))
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=50,activation='relu'))
# output layer with softmax activation
model.add(Dense(units=n_classes,activation='softmax'))

# defining loss function, optimizer, metrics and then compiling model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# training the model on training dataset
history = model.fit(X_train, y_train, epochs=100, batch_size=5000,validation_split=0.2)
```

4. Experimental Analysis

To experiment with the proposed model, we chose NSL-KDD [20] dataset to train, evaluate, and analyze the performance of our model. The NSL-KDD dataset is a widely used benchmark dataset for evaluating IDS and was developed as an updated version of the KDD Cup 1999 dataset, which is a widely used dataset for evaluating IDS in the past. The NSL-KDD dataset contains network traffic data generated from a simulated environment with a variety of network attacks, normal traffic, and different types of benign traffic. The dataset consisted of two parts, the training set, and the testing set. The training set contains 125,973 records, and the testing set contains 22,544 records. The records in the dataset are divided into five categories based on the type of traffic: Normal, Probe, DoS, R2L, and U2R. The Normal traffic category contains records of normal network traffic, while the other categories contain records of different types of network attacks. The NSL-KDD dataset contains a total of 41 features or attributes. These features are extracted from network traffic data and are used as inputs to train and test intrusion detection systems. The features can be broadly categorized into three types: basic features, content-based features, and traffic-based features. Summary of class distribution is given in Table 1.

Table 1: class distribution for NSL-KDD dataset.

Class	Training Set	Testing Set
Normal	67,343	9,730
Probe	10,532	2,416
DoS	11,009	1,940
R2L	995	288
U2R	52	10
Total	125,931	22,384

For performance evaluation, a set of metrics is used to measure the performance of our model during the inference stage. These metrics are defined as follows:

DOI: https://doi.org/10.54216/JCIM.010105

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

$$F1 - measure = 2 * \frac{Recall \times Precision}{Recall + Precision} \quad (12)$$

Statistical analysis plays a crucial role in the design of IDS for IoT scenarios, where networks generate vast amounts of traffic data that can be analyzed to identify patterns and anomalies in network traffic that may indicate the presence of a network intrusion. Statistical analysis for NSL-KDD is presented in Table 2, to establish baselines for normal network traffic behavior.

Table 2: Summary statistics for the NSL-KDD dataset.

	count	mean	std	min	25%	50%	75%	max
duration	**125973**	**287.1447**	**2.60E+03**	**0**	**0**	**0**	**0**	**4.29E+04**
src_bytes	**125973**	**45566.74**	**5.87E+06**	**0**	**0**	**44**	**276**	**1.38E+09**
dst_bytes	**125973**	**19779.11**	**4.02E+06**	**0**	**0**	**0**	**516**	**1.31E+09**
land	**125973**	**0.000198**	**1.41E-02**	**0**	**0**	**0**	**0**	**1.00E+00**
wrong_fragment	**125973**	**0.022687**	**2.54E-01**	**0**	**0**	**0**	**0**	**3.00E+00**
urgent	**125973**	**0.000111**	**1.44E-02**	**0**	**0**	**0**	**0**	**3.00E+00**
hot	**125973**	**0.204409**	**2.15E+00**	**0**	**0**	**0**	**0**	**7.70E+01**
num_failed_logins	**125973**	**0.001222**	**4.52E-02**	**0**	**0**	**0**	**0**	**5.00E+00**
logged_in	**125973**	**0.395736**	**4.89E-01**	**0**	**0**	**0**	**1**	**1.00E+00**
num_compromised	**125973**	**0.27925**	**2.39E+01**	**0**	**0**	**0**	**0**	**7.48E+03**
root_shell	**125973**	**0.001342**	**3.66E-02**	**0**	**0**	**0**	**0**	**1.00E+00**
su_attempted	**125973**	**0.001103**	**4.52E-02**	**0**	**0**	**0**	**0**	**2.00E+00**
num_root	**125973**	**0.302192**	**2.44E+01**	**0**	**0**	**0**	**0**	**7.47E+03**
num_file_creations	**125973**	**0.012669**	**4.84E-01**	**0**	**0**	**0**	**0**	**4.30E+01**
num_shells	**125973**	**0.000413**	**2.22E-02**	**0**	**0**	**0**	**0**	**2.00E+00**
num_access_files	**125973**	**0.004096**	**9.94E-02**	**0**	**0**	**0**	**0**	**9.00E+00**
num_outbound_cmds	**125973**	**0**	**0.00E+00**	**0**	**0**	**0**	**0**	**0.00E+00**
is_host_login	**125973**	**0.000008**	**2.82E-03**	**0**	**0**	**0**	**0**	**1.00E+00**
is_guest_login	**125973**	**0.009423**	**9.66E-02**	**0**	**0**	**0**	**0**	**1.00E+00**
count	**125973**	**84.10756**	**1.15E+02**	**0**	**2**	**14**	**143**	**5.11E+02**
srv_count	**125973**	**27.73789**	**7.26E+01**	**0**	**2**	**8**	**18**	**5.11E+02**
serror_rate	**125973**	**0.284485**	**4.46E-01**	**0**	**0**	**0**	**1**	**1.00E+00**
srv_serror_rate	**125973**	**0.282485**	**4.47E-01**	**0**	**0**	**0**	**1**	**1.00E+00**
rerror_rate	**125973**	**0.119958**	**3.20E-01**	**0**	**0**	**0**	**0**	**1.00E+00**
srv_rerror_rate	**125973**	**0.121183**	**3.24E-01**	**0**	**0**	**0**	**0**	**1.00E+00**
same_srv_rate	**125973**	**0.660928**	**4.40E-01**	**0**	**0.09**	**1**	**1**	**1.00E+00**
diff_srv_rate	**125973**	**0.063053**	**1.80E-01**	**0**	**0**	**0**	**0.06**	**1.00E+00**
srv_diff_host_rate	**125973**	**0.097322**	**2.60E-01**	**0**	**0**	**0**	**0**	**1.00E+00**
dst_host_count	**125973**	**182.1489**	**9.92E+01**	**0**	**82**	**255**	**255**	**2.55E+02**
dst_host_srv_count	**125973**	**115.653**	**1.11E+02**	**0**	**10**	**63**	**255**	**2.55E+02**
dst_host_same_srv_rate	**125973**	**0.521242**	**4.49E-01**	**0**	**0.05**	**0.51**	**1**	**1.00E+00**
dst_host_diff_srv_rate	**125973**	**0.082951**	**1.89E-01**	**0**	**0**	**0.02**	**0.07**	**1.00E+00**
dst_host_same_src_port_rate	**125973**	**0.148379**	**3.09E-01**	**0**	**0**	**0**	**0.06**	**1.00E+00**
dst_host_srv_diff_host_rate	**125973**	**0.032542**	**1.13E-01**	**0**	**0**	**0**	**0.02**	**1.00E+00**
dst_host_serror_rate	**125973**	**0.284452**	**4.45E-01**	**0**	**0**	**0**	**1**	**1.00E+00**
dst_host_srv_serror_rate	**125973**	**0.278485**	**4.46E-01**	**0**	**0**	**0**	**1**	**1.00E+00**

DOI: https://doi.org/10.54216/JCIM.010105

dst_host_rerror_rate	**125973**	**0.118832**	**3.07E-01**	**0**	**0**	**0**	**0**	**1.00E+00**
dst_host_srv_rerror_rate	**125973**	**0.12024**	**3.19E-01**	**0**	**0**	**0**	**0**	**1.00E+00**

Learning curve analysis is conducted here for evaluating the performance of our IDS and optimizing its parameters based on a plot of the performance of our CNN model as a function of the size of the training dataset (See Figure 1). As shown, our model shows stable training behavior without underfitting or overfitting the training data and also shows rapid convergence.

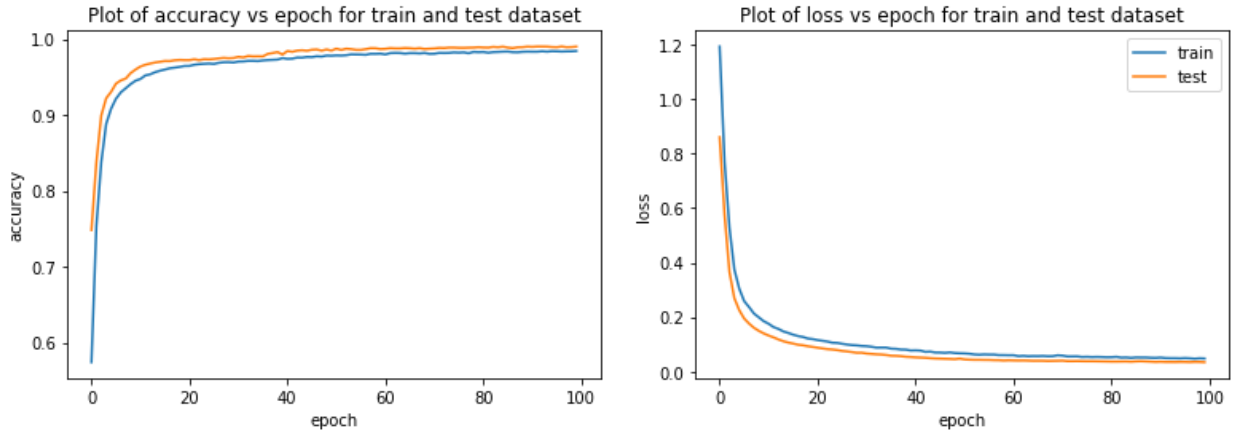

Figure 1: Visualization of the learning curves of our system on NSL-KDD data.

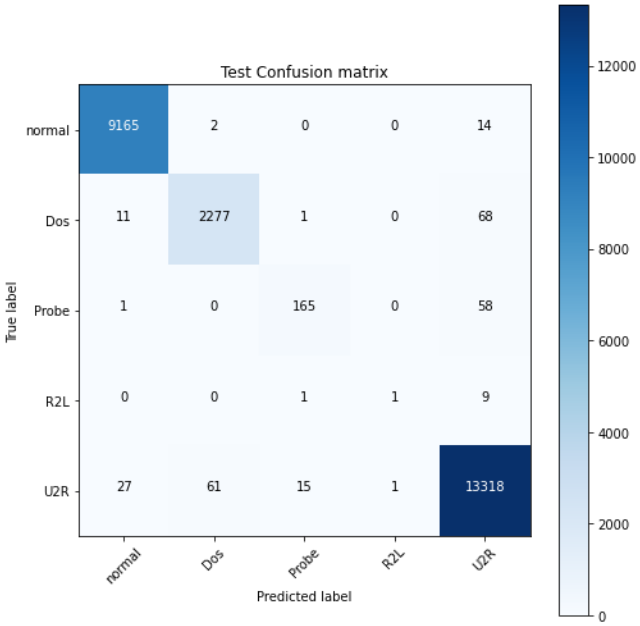

Figure 2: Visualization of confusion matrix of our system on NSL-KDD data.

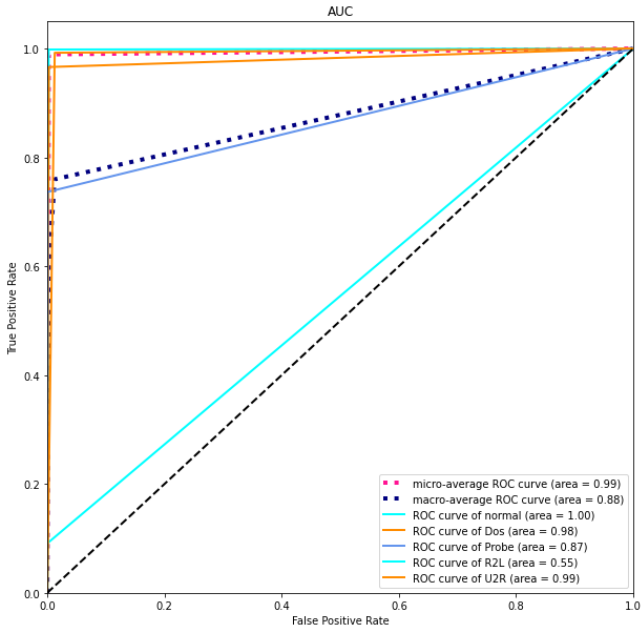

Figure 3: Visualization of ROC curves of our system on NSL-KDD data.

Confusion matrix analysis is a common technique for evaluating the performance of ML-based IDS. Figure 2 shows a confusion matrix of our system as a table that summarizes the number of true positives, true negatives, false positives, and false negatives for a given classification algorithm. The confusion matrix is typically computed using a test set

DOI: https://doi.org/10.54216/JCIM.010105

that is separate from the training dataset. The test set is used to evaluate the performance of the classification algorithm on data that it has not seen during training.

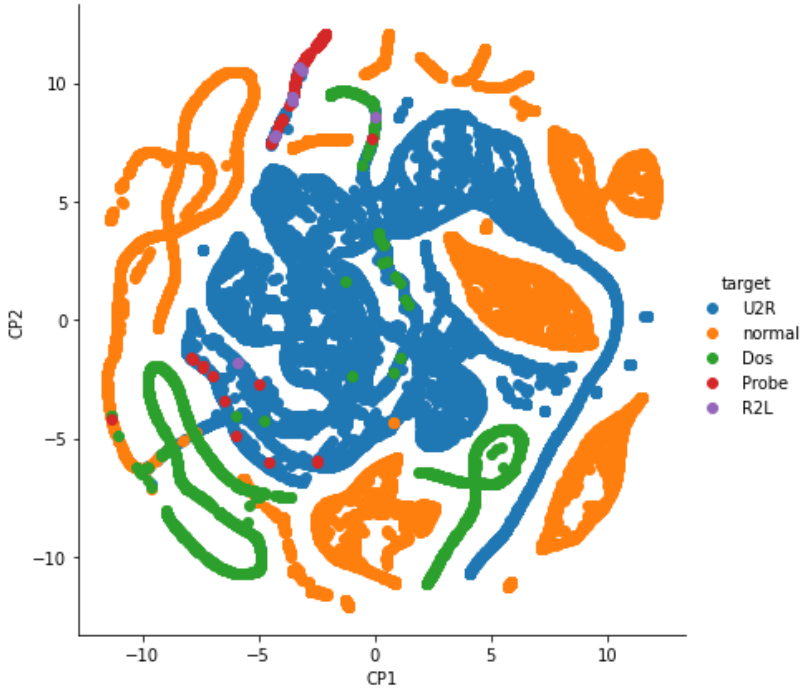

Figure 1: Visualization of T-SNE of our system on NSL-KDD data.

Receiver Operating Characteristic (ROC) curve analysis is conducted as a way of evaluating the performance of the machine in our system, as displayed in Figure 3. A ROC curve is a plot of the true positive rate (TPR) versus the false positive rate (FPR) for our CNN model. The true positive rate (TPR) is defined as the fraction of actual attacks that are correctly identified by the algorithm, computed as TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives. The false positive rate (FPR) is defined as the fraction of benign samples that are incorrectly classified as attacks, computed as FP / (FP + TN), where FP is the number of false positives and TN is the number of true negatives. The area under the ROC curve (AUC) is a commonly used performance metric for IDS. A perfect classifier would have an AUC of 1.0, while a random classifier would have an AUC of 0.5. An AUC value between 0.5 and 1.0 indicates the performance of our model for each type of attack.

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a technique for visualizing high-dimensional data in a lower-dimensional space. It is commonly used for analyzing the performance of ML-based IDS. As shown in Figure 4, t-SNE is used to visualize the distribution of attack and benign samples in a lower-dimensional space. This can help to identify patterns and clusters of attacks that are not easily visible in the original high-dimensional feature space.

5. Conclusion

This research presents a novel approach to address the security challenges in the IoT using a Convolutional Neural Network (CNN) based IDS. The proposed IDS effectively detects different types of attacks on the IoT network by analyzing network traffic patterns in real time. The experimental evaluation of the proposed system shows that it outperforms traditional IDS approaches in terms of accuracy, detection rate, and false alarm rate. Our system makes significant progress in addressing the security challenges associated with IoT networks. The proposed CNN-based IDS provides a reliable and efficient solution to detect various types of attacks in real time, improving the overall security of the IoT network. We hope that this work will inspire further research in this area and contribute to the development of more robust and effective security solutions for IoT networks.

DOI: https://doi.org/10.54216/JCIM.010105

References

[1]. Zhou, Y., Cheng, G., Jiang, S., & Dai, M. (2020). Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer networks*, *174*, 107247.

[2]. Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbeh, M. (2017, September). Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)* (pp. 000277-000282). IEEE.

[3]. Wang, H., Gu, J., & Wang, S. (2017). An effective intrusion detection framework based on SVM with feature augmentation. *Knowledge-Based Systems*, *136*, 130-139.

[4]. Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016, May). 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).

[5]. Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, *9*(20), 4396.

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

[7]. Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, *12*, 493-501.

[8]. Alrawashdeh, K., & Purdy, C. (2016, December). Toward an online anomaly intrusion detection system based on deep learning. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)* (pp. 195-200). IEEE.

[9]. Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, *189*, 105124.

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

[11]. Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019). A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, *7*, 30373-30385.

[12]. Kang, M. J., & Kang, J. W. (2016). Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, *11*(6), e0155781.

[13]. Bamakan, S. M. H., Wang, H., Yingjie, T., & Shi, Y. (2016). An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization. *Neurocomputing*, *199*, 90-102.

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

[15]. Ahmad, I., Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE access*, *6*, 33789-33795.

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

[17]. Da Costa, K. A., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, *151*, 147-157.

[18]. Lin, W. C., Ke, S. W., & Tsai, C. F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, *78*, 13-21.

[19]. Abdel-Basset, M., Chang, V., Hawash, H., Chakrabortty, R. K., & Ryan, M. (2020). Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment. *IEEE Transactions on Industrial Informatics*, *17*(11), 7704-7715.

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

DOI: https://doi.org/10.54216/JCIM.010105