Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 6, No. 6, June : 2023

DEEP LEARNING CONVOLUTIONAL NEURAL NETWORKS BASED INTRUSION DETECTION FRAMEWORK IN INDUSTRIAL INTERNET OF THINGS

Ashalatha Koppineni, M. Tech, Department of Computer Science and Engineering, Swarnandhra College of Engineering and Technology (A), Narasapur, Andhra Pradesh, India
Dr. Srinivasulu Pamidi, Professor & Head, Department of Computer Science and Engineering, Swarnandhra College of Engineering and Technology (A), Narasapur, Andhra Pradesh, India.
Tulasi Raju Nethala, Associate Professor, Department of Computer Science and Engineering, Swarnandhra College of Engineering and Technology (A), Narasapur, Andhra Pradesh, India.

Abstract

With the widespread adoption of the Industrial Internet of Things (IIoT) in various sectors, ensuring the security of IIoT systems has become a critical concern. Intrusion detection plays a vital role in safeguarding IIoT networks from malicious activities. In this study, we propose a deep learning framework based on Convolutional Neural Networks (CNNs) for intrusion detection in the IIoT environment. We utilize the NSL-KDD dataset, which is a benchmark dataset widely used for evaluating intrusion detection systems. The proposed framework leverages the inherent capability of CNNs to extract features from network traffic data without the need for manual feature engineering. We preprocess the dataset, perform data augmentation, and partition it into training and testing sets. The CNN model is then trained on the training set and evaluated on the testing set. Experimental results demonstrate that our CNN-based intrusion detection framework achieves superior performance in terms of accuracy, precision, recall, and F1-score compared to traditional machine learning approaches. The proposed framework provides an effective and efficient solution for detecting intrusions in IIoT systems, enhancing their security and robustness.

Keywords: Deep Learning, Convolutional Neural Networks, Intrusion Detection, Industrial Internet of Things, NSL-KDD dataset.

1. Introduction

Systems and networks dealing with information and communications technology (ICT) [1] are susceptible to various attacks from internal and external threats, making them vulnerable to data breaches. For example, Yahoo experienced a data breach resulting in a loss of around $350 million, while Bitcoin suffered a breach resulting in a loss of approximately $70 million. These attacks are becoming more complex and challenging to detect due to advancements in hardware, software, and network topologies, especially with the emergence of new technologies like the Internet of Things (IoT) [2]. Therefore, there is a growing need for a new and dependable intrusion detection system (IDS) [3] to enhance network security. An IDS is a proactive technology that automatically identifies and categorizes intrusions, attacks, or security breaches in both network-level and host-level infrastructures. Network-based intrusion detection systems (NIDS) [4] and host-based intrusion detection systems (HIDS) [5] are two categories used to classify intrusion detection. NIDS analyzes network traffic by recording and analyzing activities using network devices such as switches, routers, and network taps. On the other hand, HIDS monitors the actions of a local host computer and records events in log files, falling under the broader category of intrusion prevention.

Companies often use a combination of NIDS and HIDS [6]in their data gathering and evaluation processes. Three approaches are commonly employed: analysis of stateful protocols, anomaly detection, and misuse detection. Misuse detection identifies inappropriate use of network resources by utilizing previously configured signatures and filters. Anomaly detection uses heuristic methods to identify previously unknown instances of malicious behavior but often produces false positives. Commercial solutions often combine abuse detection and anomaly detection since they complement each other. Stateful protocol analysis operates on different levels (network layer, application layer, and transport layer) [7] simultaneously, resulting in more accurate findings by checking variations in relevant protocols and applications. While recent research explores deep learning approaches to enhance the intelligence of intrusion detection systems, there is a scarcity of research assessing machine learning algorithms using publicly available datasets. This scarcity poses a challenge in developing effective algorithms. Common issues with current machine learning models include a significant risk of false positives across a wide range of attacks, lack of generalizability due to limited datasets used in earlier research, high false positive rates, and limited threat detection capabilities. Additionally, these models have not considered the tremendous amount of network traffic present today, and they must adapt to increasingly complex network settings.

To address these challenges, this work focuses on evaluating the effectiveness of various machine learning classifiers and deep neural networks (DNNs) [8] in NIDS and HIDS. The study models DNN as an efficient and effective strategy to detect cyberattacks in advance [9], combining the capabilities of host and network intrusion detection systems. The research evaluates the performance of traditional machine learning techniques and DNNs on a wide range of NIDS and HIDS datasets to determine normal and abnormal network traffic behavior in response to different attack categories. Additionally, sophisticated text representation approaches using natural language processing (NLP) are examined for host-level events (system calls) to capture contextual[10] , semantic, and sequential information. Benchmark datasets, such as NSL-KDD, are utilized for comparative experiments due to the challenges faced in real-world data collections.

2. Literature survey

In [11] authors focus on the implementation of machine learning techniques for detecting different types of attacks on network and host systems. The authors discuss the security issues that arise when using machine learning in intrusion detection and provide an overview of various attacks and their unique features. They evaluate the strengths and weaknesses of different machine learning algorithms, including evolutionary strategies, and compare the performance of single classifier techniques with other classifier approaches. In [12] authors proposed a hybrid approach that combines feature-based and data-based visualizations for accurately detecting and classifying malicious software, specifically zero-day malware. They use similarity mining to compare malware samples based on distance scores and generate similarity matrices. The approach successfully identifies and categorizes previously undetectable malware by analyzing the behavioral patterns of viruses. In [13] authors address the detection of botnet traffic, which poses a significant risk to individuals and businesses. The authors present various strategies, including dynamic packet inspection, DNS request behavior, temporal analysis, correlation, and machine learning, for identifying botnet traffic. They introduce a novel strategy called CONIFA, which leverages machine learning and aims to bridge the gap between trained and untrained versions of malicious software.

In [14] authors focus on the identification and modeling of cybersecurity vulnerabilities, particularly in networked and intelligent systems. The authors propose a statistical technique to analyze the disclosure process of vulnerabilities and investigate the long-term effects using various GARCH models. The research aims to understand the dynamics of vulnerability disclosure and mitigate potential harm caused by exploiting these vulnerabilities. In [15] authors present a literature review that examines the application of machine learning and deep learning techniques in network security, with a focus on intrusion detection. The authors discuss the challenges and limitations of different intrusion detection approaches, the need for representative datasets, and the continuous learning requirements for network information. They emphasize the importance of holistic approaches and lifelong learning in cybersecurity. In [16] authors propose a host-based intrusion detection system (HIDS) framework based on analyzing the frequency of n-gram phrases in system call trace files. By transforming system call traces into n-gram vectors and applying dimensionality reduction, the framework effectively distinguishes between normal and intrusive system activities. The experiments conducted using the ADFA-LD dataset demonstrate the framework's accuracy in detecting intrusions while reducing processing costs.

In [17] authors present three data-mining-based frameworks for network intrusion detection using random forests, an algorithm capable of automatically generating patterns. The frameworks utilize random forests to detect hybrids, abuse, and anomalies in network traffic. The authors apply feature selection, parameter tuning, and sampling techniques to improve the performance of the frameworks. They also propose an unsupervised anomaly identification method that overcomes the limitations of rule-based systems. In [18] authors focus on the detection of malicious URLs using machine learning techniques. The authors propose a framework that combines feature extraction and classification algorithms to accurately identify and classify malicious URLs. They extract various features such as domain-based, content-based, and host-based features, and employ algorithms like decision trees, random forests, and support vector machines for classification. The experimental results demonstrate the effectiveness of the proposed framework in detecting malicious URLs. In [19] authors present a comprehensive review of machine learning techniques applied to email spam detection. The authors discuss various approaches, including content-based, header-based, and hybrid methods, and analyze the performance of different machine learning algorithms such as Naive Bayes, Support Vector Machines, and Random Forests. They also highlight the challenges in email spam detection and suggest future research directions in this field. In [20] authors discuss the importance of integrating machine learning techniques with existing security systems to improve threat detection, classification, and response. The paper covers various machine learning algorithms, including decision trees, neural networks, and clustering techniques, and their application in detecting malware, identifying attack patterns, and predicting future threats. The authors emphasize the need for continuous monitoring and adaptive learning to counter evolving cyber threats.

3. Proposed system

The development of an IDS that can recognize and classify cyberattacks in a timely and automated way at both the network level and the host level is making extensive use of machine learning technology. This is being done on both levels, the network level, and the host level. On both levels, the network level and the host level, this activity is being carried out. On the other hand, because damaging attacks are continually adapting to new forms and taking place in very large numbers, it is necessary to develop a system that can be scaled. Various Intrusion datasets are accessible for further study by the cyber security community and is accessed via the public domain. On the other hand, there is no research that can be accessible at this time that has offered a full examination of the performance of a variety of machine learning algorithms on a variety of datasets that are available to the public. This is because there is currently no study that has conducted such an analysis. Because of the dynamic nature of intrusion, which includes continually shifting methods of attack, the publicly accessible intrusion datasets need to be updated on a regular basis and evaluated. This study examines the effectiveness of many traditional algorithms, including SVM, Random Forest, and DNN, CNN, among others, in identifying attacks on networks by using KDD and NSL datasets. The current classical algorithms (SVM, Random Forest, and DNN) are unable to foresee dynamic assaults and need to be trained in advance. This is because the classical algorithms were not designed to handle such scenarios.

The suggested technique is shown as a block diagram in Figure 1. At first, the NSL-KDD dataset is partitioned such that 80% is used for training and 20% is used for testing. After that, a dataset pre-processing procedure is carried out to standardize the whole dataset. In addition, the CNN classifier is used for the purpose of threat assessment based on test samples. The purpose of the performance assessment is to demonstrate the superiority of the suggested approach. The CNN is a well-known algorithm that has a high predictive ratio in a variety of domains, including data processing and data categorization, amongst others. Therefore, the CNN model can identify these kinds of attacks and to solve the difficulties caused by assaults using dynamic attack signatures. The suggested model for the CNN has a significant number of layers. The CNN approach uses the hidden layer as a filter for the training process on an ongoing basis to build the most accurate model possible to predict the testing class. Although the dataset has additional names, the most common types are Normal, Remote to user (R2L), Denial-of-Service (DOS), User to Root (U2R), and Probe. Even though the dataset contains other names, all those names fall into one of the categories indicated above.

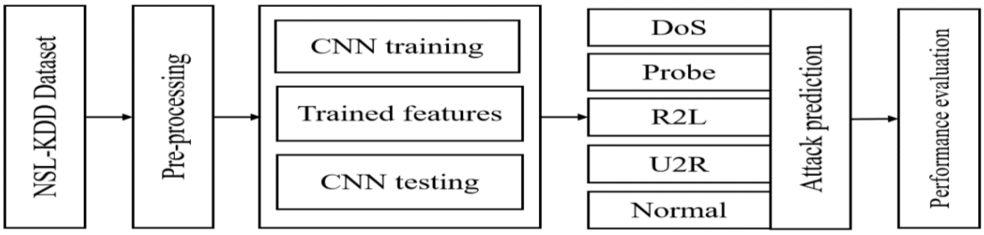


Fig. 1: Proposed methodology.

3.1 Preprocessing

The process of cleaning and formatting raw data so that it is used in a machine learning model is referred to as "data preprocessing." This phase, which is both the first and most important step, kicks off the process of developing a model for machine learning, which is the process. We do not always come across data that is both clean and organized when we are utilizing machine learning to develop a project since it is not always the case. This is since it does not always hold true. In addition, before carrying out any action that includes data, it is essential to thoroughly clean the data and arrange it in the right manner. This should be done before carrying out any activity that involves data.

3.2 CNN Classifier

Figure 2 depicts the deep CNN model used for the detection of intrusions. It is a straightforward graphical formalism that is use for the purpose of representing a system in terms of the data that is fed into the system, the different operations that are performed on this data, and the data that is produced by this system. It is one of the most essential tools for model building. It is use in modelling the various components of the system. This component set is comprised of the following components: an external entity that communicates with the system; the information flows that take place inside the system; the process that runs on the system; and the data that are used by the process. Additionally, this component set includes the information flows within the system. In addition to this, it illustrates how the information moves across the system and how the status of the information is changed because of a variety of different shifts. It is a sort of graphical representation that illustrates the flow of information and the transformations that occur as it moves from the input to the output of the system. Flowcharts are one kind of graphical representation that is used. Additionally, it is used to represent a system at any level of abstraction, and it can be partitioned into layers that reflect increasing information flow and functional complexity. Additionally, it can be used to represent a system at any degree of abstraction. In addition to this, it is used to depict a system at whatever degree of abstraction desired. In addition to this, it is used to describe a system at any level of abstraction, making it very flexible.

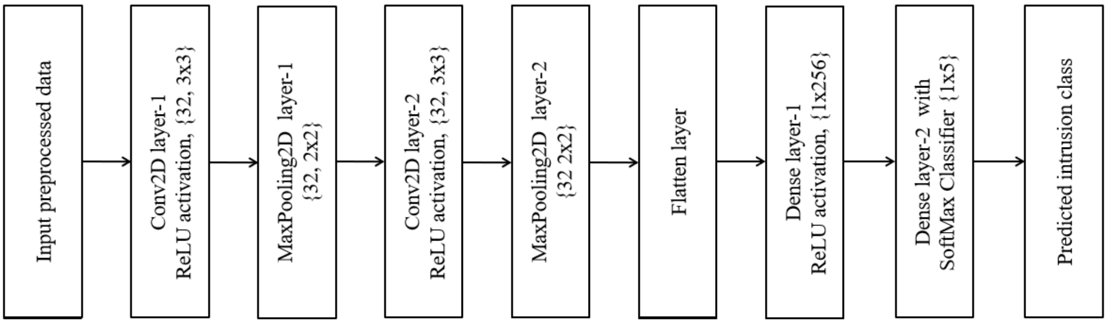


Fig. 2: Proposed deep CNN model.

According to the information that is presently available, training and testing a CNN involves allowing all the source data to pass through a series of convolution layers that are implemented by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer, and utilizing SoftMax layer with classification layer to categorize the objects with probabilistic values that range from [0,1]. Figure 3 demonstrates that the main layer that is used in the process of extracting features from a given data set is the convolution layer. This layer also preserves the link between pixels by learning the characteristics of data using small blocks of source data. This layer is called the "convolution layer." It is a mathematical function that takes into consideration two inputs, such as the source data $I(x, y, d)$, in which x and y represent the spatial coordinates, often referred to as the quantity of rows and columns. d is the dimension of a data collection, which in this example is equal to three since the source data, and the expression for a filter or kernel with a size of input data that is equivalent to d's size is written as $F(k_x, k_y, d)$. k_x and k_y are the sizes of the input data, and d is the size of d.

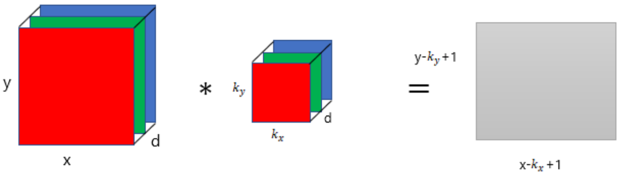


Fig. 3: Process representation for convolution layers.

The result of the convolution process involving the input data and the filter has a size of $C\left((x - k_x + 1), (y - k_y + 1), 1\right)$ and is referred to as the feature map.

ReLU layer: Rectified linear units, more often known to as ReLUs, are a particular kind of network that use the rectifier operation for its hidden layers. Rectified linear units are more commonly referred to as ReLUs. Rectified linear units are also referred to by their acronym ReLUs. This basic operation, which is known as the ReLU function $\mathcal{G}(\cdot)$, will return the value that was provided as input directly if the value of the input is higher than zero; however, it will return zero if the value of the input is lower than zero. In mathematical terms, this is stated as the application of the function $max(\cdot)$ over the set 0 and the input x in the way that is explained below. In other words, this can be written as [[mathematical expression.

$$\mathcal{G}(x) = \max\{0, x\}$$

Max pooing layer: If there are data sets of a larger size, this layer reduces the necessary number of parameters by a significant amount. Subsampling, which is also known as down sampling, is a technique that reduces the dimensionality of each feature map while still maintaining the integrity of the data that is crucial to the investigation. When max pooling is used, the largest element from the corrected feature map is taken into consideration.

SoftMax classifier: The SoftMax function is tacked on to the very end of the output in most applications. This is because it is the site where the nodes eventually meet, and therefore, they can be classified. This is the reason why it is important. Here, X acts as the input for each of the models; the layers that are located in-between X and Y are referred to as the hidden layers, and data travels from X via each of the layers on its way to Y. X and Y are both located on a two-dimensional plane. Let's say there are 10 different classes, and our goal is to determine which one the input information falls into. Considering this, the action that we do is to assign a certain expected output to each class. We have ten outputs that correspond to ten distinct classes, and we anticipate the class based on the probability that is greatest for that class.

4. Result and discussion

This study conducts an analysis to determine how well several conventional algorithms, such as SVM and Random Forest, can identify malicious activity on a computer network. However, these conventional algorithms are unable to forecast dynamic assaults (if an attacker presents new attacks with changes in attack parameter), and therefore need previous training to identify attacks of this kind.

In addition, these traditional algorithms are not able to detect attacks of this type. This research assesses the performance of a CNN model that includes dynamic attack signatures so that the issue is solved. The results of CNN's detection accuracy tests have shown that they are superior to those of all traditional techniques.

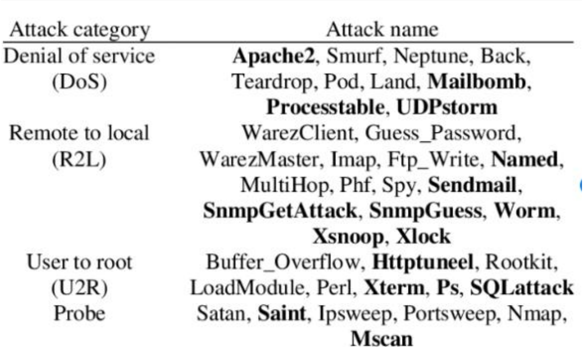
4.1 Dataset

To put this idea into practice, we have NSL-KDD dataset and used the SVM, Random Forest, DNN, and CNN modeling techniques. CNN models continuously filter training algorithms using hidden layers to produce the most accurate model possible to predict testing class. It is a well-known model that has a high accuracy rate of prediction in a variety of domains, including image processing, data categorization, and other areas. The columns of the dataset columns are listed below.

duration,protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,label

The names of the assaults are indicated by the column labels in the preceding table; the names of request signatures are shown above in bold format and are separated by commas.

Figure 4 shows the dataset description. This study describes Normal, R2L, DOS, U2R, DOS, and Probe; nevertheless, the dataset contains additional names; yet, all these other names fall into one of five categories, including Normal, R2L, DOS, U2R, DOS, or Probe. Figure 5 shows the number of records of each class in dataset. The x-axis in the above graph displays the attack name that was discovered in the dataset, and the y-axis displays the count of instances of that attack type. We can see that the dataset has a total of 10137 records by looking at the screen that was just shown. The program is utilizing 8109 records for training, and it is using 2028 records to assess the accuracy of its algorithm prediction.



Attack category	Attack name
Denial of service (DoS)	**Apache2**, Smurf, Neptune, Back, Teardrop, Pod, Land, **Mailbomb**, **Processtable**, **UDPstorm**
Remote to local (R2L)	WarezClient, Guess_Password, WarezMaster, Imap, Ftp_Write, **Named**, MultiHop, Phf, Spy, **Sendmail**, **SnmpGetAttack**, **SnmpGuess**, **Worm**, **Xsnoop**, **Xlock**
User to root (U2R)	Buffer_Overflow, **Httptuneel**, Rootkit, LoadModule, Perl, **Xterm**, **Ps**, **SQLattack**
Probe	Satan, **Saint**, Ipsweep, Portsweep, Nmap, **Mscan**

Fig. 4: Dataset description.

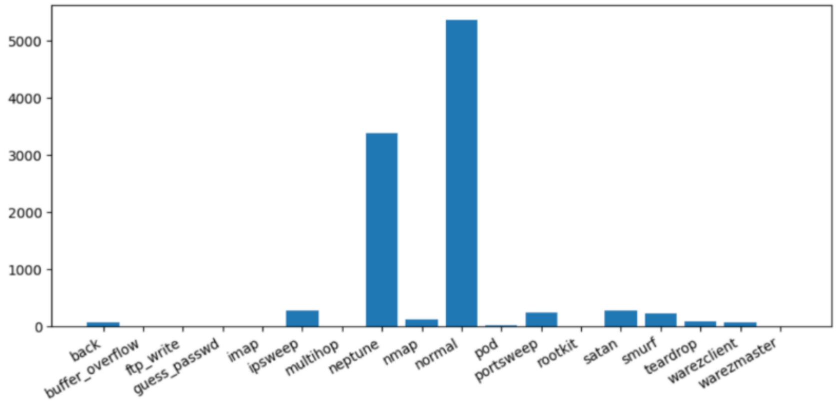


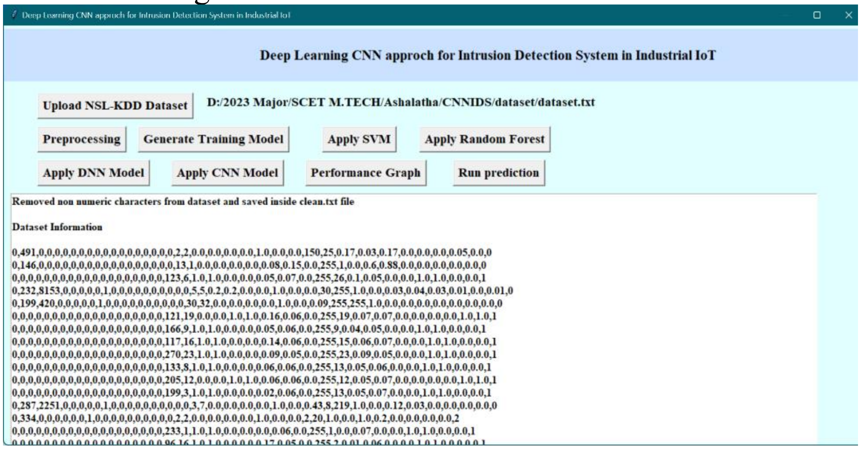
Fig. 5: Count of each class.

The values of the columns in the dataset that was just presented are shown below.

0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal

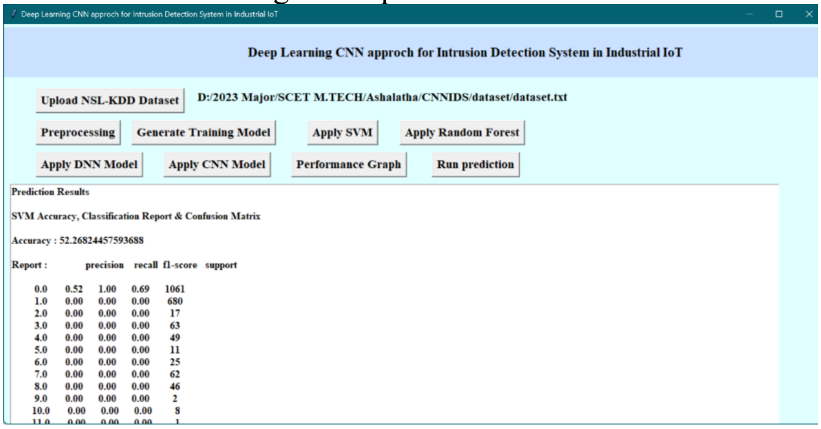
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,Neptune

The first two records represent the signature values, while the third record carries the class label. Examples of class labels include attack signature and typical request signature. The second record has the name "Neptune," which is the name of the assault. In a similar vein, the dataset contains close to thirty distinct names for various kinds of assaults. The above dataset records show that some of the values are stored in string format, such as tcp and ftp_data; however, these values do not contribute significantly to the accuracy of the prediction. Because of this, the PREPROCESSING Concept will be used to remove these string-formatted values from the dataset as shown in Figure 6. If the method is presented in text format, then the algorithm will not be able to identify any of the attack names; thus, we need to assign a numeric value to each assault. All of this will be accomplished via a series of PREPROCESS stages, and after that, a new file with the name "clean.txt" will be created. This file will be used to produce a training model.


Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Upload NSL-KDD Dataset
D:/2023 Major/SCET M.TECH/Ashalatha/CNNIDS/dataset/dataset.txt
Preprocessing
Generate Training Model
Apply SVM
Apply Random Forest
Apply DNN Model
Apply CNN Model
Performance Graph
Run prediction

```
Removed non numeric characters from dataset and saved inside clean.txt file

Dataset Information

0,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.0,0.0,0.0,0.0,1.0,0.0,0.0,150,25,0.17,0.03,0.17,0.0,0.0,0.0,0.05,0.0,0
0,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0.0,0.0,0.0,0.0,0.08,0.15,0.0,255,1,0.0,0.6,0.88,0.0,0.0,0.0,0.0,0.0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1.0,1.0,0.0,0.0,0.05,0.07,0.0,255,26,0.1,0.05,0.0,0.0,1.0,1.0,0.0,0.0,1
0,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,5,5,0.2,0.2,0.0,0.0,1.0,0.0,0.0,30,255,1.0,0.0,0.03,0.04,0.03,0.01,0.0,0.01,0
0,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,30,32,0.0,0.0,0.0,0.0,1.0,0.0,0.09,255,255,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.0,0.0,1.0,1.0,0.16,0.06,0.0,255,19,0.07,0.07,0.0,0.0,0.0,0.0,1.0,1.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.0,1.0,0.0,0.0,0.05,0.06,0.0,255,9,0.04,0.05,0.0,0.0,1.0,1.0,0.0,0.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1.0,1.0,0.0,0.0,0.14,0.06,0.0,255,15,0.06,0.07,0.0,0.0,1.0,1.0,0.0,0.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1.0,1.0,0.0,0.0,0.09,0.05,0.0,255,23,0.09,0.05,0.0,0.0,1.0,1.0,0.0,0.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1.0,1.0,0.0,0.0,0.06,0.06,0.0,255,13,0.05,0.06,0.0,0.0,1.0,1.0,0.0,0.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,205,12,0.0,0.0,1.0,1.0,0.06,0.06,0.0,255,12,0.05,0.07,0.0,0.0,0.0,0.0,1.0,1.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,199,3,1.0,1.0,0.0,0.0,0.02,0.06,0.0,255,13,0.05,0.07,0.0,0.0,1.0,1.0,0.0,0.0,1
0,287,2251,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,3,7,0.0,0.0,0.0,0.0,1.0,0.0,0.43,8,219,1.0,0.0,0.12,0.03,0.0,0.0,0.0,0.0,0
0,334,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0.0,0.0,0.0,0.0,1.0,0.0,0.0,2,20,1.0,0.0,1.0,0.2,0.0,0.0,0.0,0.0,2
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,233,1,1.0,1.0,0.0,0.0,0.0,0.06,0.0,255,1,0.0,0.07,0.0,0.0,1.0,1.0,0.0,0.0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,96,16,1.0,1.0,0.0,0.0,0.17,0.05,0.0,255,2,0.01,0.06,0.0,0.0,1.0,1.0,0.0,0.0,1
```

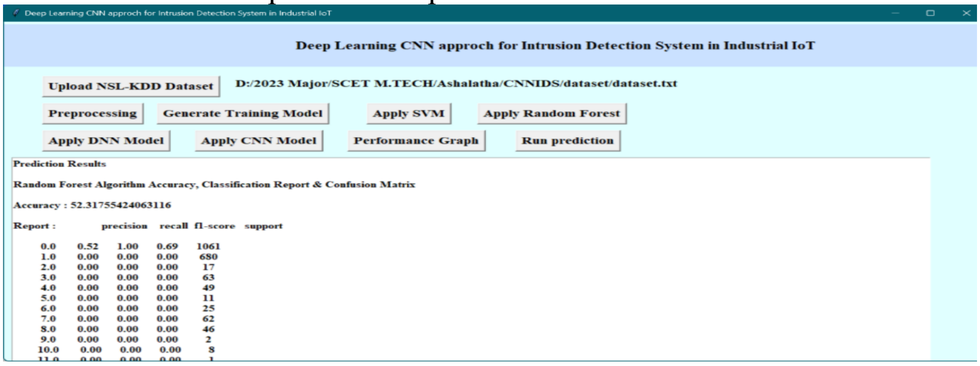
Fig. 6: Preprocessed data.


Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Upload NSL-KDD Dataset
D:/2023 Major/SCET M.TECH/Ashalatha/CNNIDS/dataset/dataset.txt
Preprocessing
Generate Training Model
Apply SVM
Apply Random Forest
Apply DNN Model
Apply CNN Model
Performance Graph
Run prediction

```
Prediction Results

SVM Accuracy, Classification Report & Confusion Matrix

Accuracy : 52.26824457593688

Report :            precision    recall  f1-score   support

        0.0      0.52      1.00      0.69      1061
        1.0      0.00      0.00      0.00       680
        2.0      0.00      0.00      0.00        17
        3.0      0.00      0.00      0.00        63
        4.0      0.00      0.00      0.00        49
        5.0      0.00      0.00      0.00        11
        6.0      0.00      0.00      0.00        25
        7.0      0.00      0.00      0.00        62
        8.0      0.00      0.00      0.00        46
        9.0      0.00      0.00      0.00         2
       10.0      0.00      0.00      0.00         8
       11.0      0.00      0.00      0.00         1
```

Fig. 7: Existing SVM classifier performance evaluation.

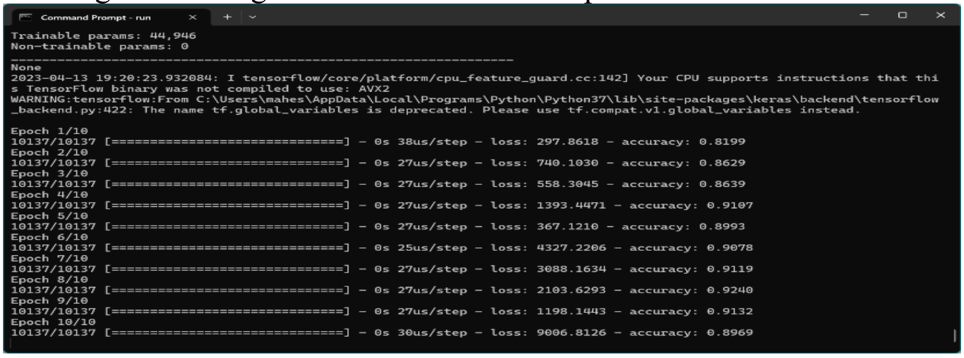
Figure 7 shows the existing SVM performance evaluation. The SVM gave us an accuracy of 52.26 percent. The classification report also presented, where each class precision, recall, and F1-score are measured. Figure 8 shows the existing random forest classifier performance evaluation. The random forest achieved an accuracy of 52.31%. The classification report also presented, where each class precision, recall, and F1-score are measured. Figure 9 shows the existing DNN classifier performance evaluation. The accuracy of the DNN algorithm is shown to be 89.96% in the screen shot that was taken earlier; this score is higher than that of the other two methods. Figure 10 shows the proposed CNN classifier performance evaluation. The accuracy of the CNN algorithm is shown to be 98% in the screen shot that was taken earlier; this score is higher than that of the other two methods. Figure 10 shows performance comparison of various classifiers. Figure 12 shows the prediction of attack from test data. The test results are interpreted as Neptune.


Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Upload NSL-KDD Dataset
Preprocessing
Generate Training Model
Apply SVM
Apply Random Forest
Apply DNN Model
Apply CNN Model
Performance Graph
Run prediction

D:/2023 Major/SCET M.TECH/Ashalatha/CNNIDS/dataset/dataset.txt

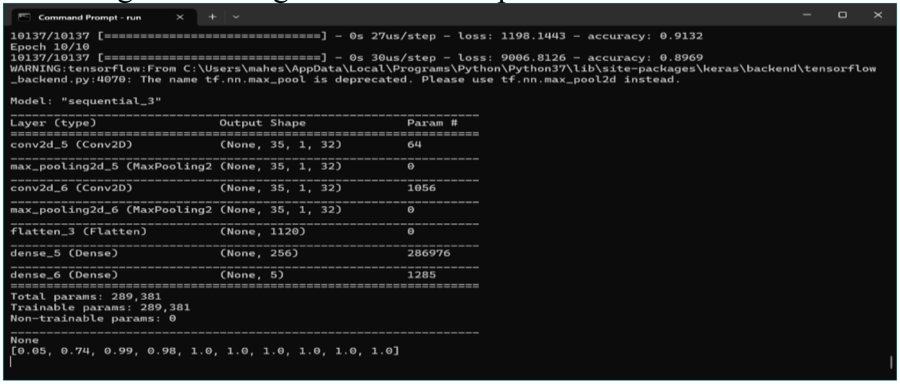
```
Prediction Results

Random Forest Algorithm Accuracy, Classification Report & Confusion Matrix

Accuracy : 52.31755424063116

Report :          precision    recall  f1-score   support

      0.0     0.52     1.00     0.69     1061
      1.0     0.00     0.00     0.00      680
      2.0     0.00     0.00     0.00       17
      3.0     0.00     0.00     0.00       63
      4.0     0.00     0.00     0.00       49
      5.0     0.00     0.00     0.00       11
      6.0     0.00     0.00     0.00       25
      7.0     0.00     0.00     0.00       62
      8.0     0.00     0.00     0.00       46
      9.0     0.00     0.00     0.00        2
     10.0     0.00     0.00     0.00        8
     11.0     0.00     0.00     0.00        1
```

Fig. 8: Existing random forest classifier performance evaluation.



```
Trainable params: 44,946
Non-trainable params: 0
_________________________________________________________________
None
2023-04-13 19:20:23.932084: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:From C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow
_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/10
10137/10137 [==============================] - 0s 38us/step - loss: 297.8618 - accuracy: 0.8199
Epoch 2/10
10137/10137 [==============================] - 0s 27us/step - loss: 740.1030 - accuracy: 0.8629
Epoch 3/10
10137/10137 [==============================] - 0s 27us/step - loss: 558.3045 - accuracy: 0.8639
Epoch 4/10
10137/10137 [==============================] - 0s 27us/step - loss: 1393.4471 - accuracy: 0.9107
Epoch 5/10
10137/10137 [==============================] - 0s 27us/step - loss: 367.1210 - accuracy: 0.8993
Epoch 6/10
10137/10137 [==============================] - 0s 25us/step - loss: 4327.2206 - accuracy: 0.9078
Epoch 7/10
10137/10137 [==============================] - 0s 27us/step - loss: 3088.1634 - accuracy: 0.9119
Epoch 8/10
10137/10137 [==============================] - 0s 27us/step - loss: 2103.6293 - accuracy: 0.9240
Epoch 9/10
10137/10137 [==============================] - 0s 27us/step - loss: 1198.1443 - accuracy: 0.9132
Epoch 10/10
10137/10137 [==============================] - 0s 30us/step - loss: 9006.8126 - accuracy: 0.8969
```

Fig. 9: Existing DNN classifier performance evaluation.



```
10137/10137 [==============================] - 0s 27us/step - loss: 1198.1443 - accuracy: 0.9132
Epoch 10/10
10137/10137 [==============================] - 0s 30us/step - loss: 9006.8126 - accuracy: 0.8969
WARNING:tensorflow:From C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow
_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 35, 1, 32)         64
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 35, 1, 32)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 35, 1, 32)         1056
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 35, 1, 32)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1120)              0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               286976
_________________________________________________________________
dense_6 (Dense)              (None, 5)                 1285
=================================================================
Total params: 289,381
Trainable params: 289,381
Non-trainable params: 0
_________________________________________________________________
None
[0.05, 0.74, 0.99, 0.98, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

Fig. 10: Proposed CNN classifier performance evaluation.

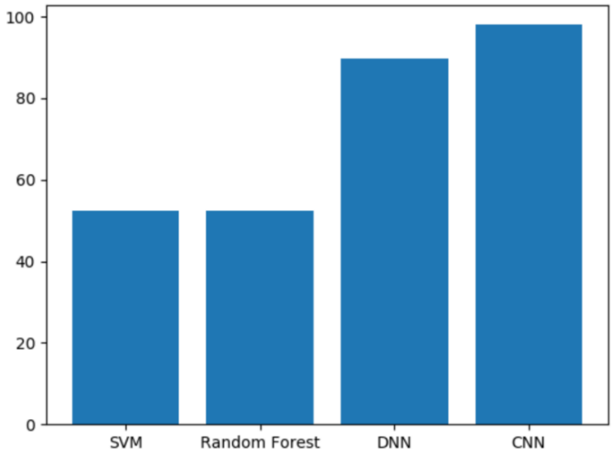
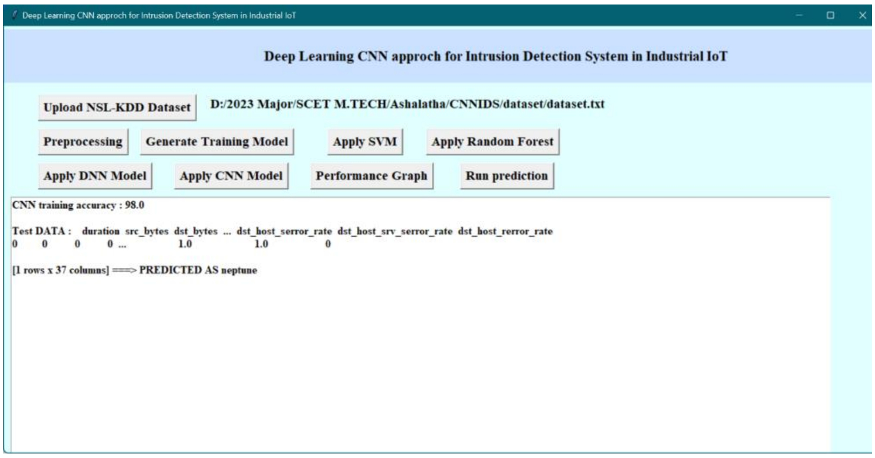


Fig. 11: Performance comparison of various classifiers



Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Deep Learning CNN approch for Intrusion Detection System in Industrial IoT
Upload NSL-KDD Dataset
Preprocessing
Generate Training Model
Apply SVM
Apply Random Forest
Apply DNN Model
Apply CNN Model
Performance Graph
Run prediction

D:/2023 Major/SCET M.TECH/Ashalatha/CNNIDS/dataset/dataset.txt

```
CNN training accuracy : 98.0

Test DATA :   duration  src_bytes  dst_bytes  ...  dst_host_serror_rate  dst_host_srv_serror_rate  dst_host_rerror_rate
0        0         0         0 ...                  1.0                      1.0                    0

[1 rows x 37 columns] ==> PREDICTED AS neptune
```

Fig. 12: Prediction of attack from test data.

5. Conclusion

In conclusion, the objective of the project was to develop a hybrid intrusion detection system that could effectively detect attacks on both the network and host levels. The system utilized a distributed deep learning model, specifically a CNN, to process and assess large amounts of data in real time. The performance of the CNN model was evaluated and compared to traditional machine learning classifiers using benchmark IDS datasets, and the CNN model consistently outperformed the traditional classifiers. In future work, addressing the challenge of training complex CNN architectures on benchmark IDS datasets should be a key consideration. Additionally, exploring the integration of DNS and BGP event monitoring and utilizing more powerful hardware for distributed learning can further enhance the overall performance of the system. These efforts will contribute to the advancement of hybrid intrusion detection systems, enabling more accurate and efficient detection of attacks in real-time scenarios.

References

Chalé, Marc, and Nathaniel D. Bastian. "Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems." *Expert Systems with Applications* 207 (2022): 117936.

Apruzzese, G., Andreolini, M., Ferretti, L., Marchetti, M., & Colajanni, M. (2022). Modeling realistic adversarial attacks against network intrusion detection systems. *Digital Threats: Research and Practice (DTRAP)*, *3*(3), 1-19.

Apruzzese, G., Pajola, L., & Conti, M. (2022). The Cross-evaluation of Machine Learning-based Network Intrusion Detection Systems. *IEEE Transactions on Network and Service Management*.

Zhang, C., Costa-Pérez, X., & Patras, P. (2022). Adversarial Attacks Against Deep Learning-Based Network Intrusion Detection Systems and Defense Mechanisms. *IEEE/ACM Transactions on Networking*.

Asad, Hafizul, and Ilir Gashi. "Dynamical analysis of diversity in rule-based open source network intrusion detection systems." *Empirical Software Engineering* 27.1 (2022): 1-30.

Sheatsley, R., Papernot, N., Weisman, M. J., Verma, G., & McDaniel, P. (2022). Adversarial examples for network intrusion detection systems. *Journal of Computer Security*, (Preprint), 1-26.

Data, M., & Aritsugi, M. (2022, July). AB-HT: An Ensemble Incremental Learning Algorithm for Network Intrusion Detection Systems. In *2022 International Conference on Data Science and Its Applications (ICoDSA)* (pp. 47-52). IEEE.

Magan-Carrion, R., Urda, D., Diaz-Cano, I., & Dorronsoro, B. (2022). Improving the Reliability of Network Intrusion Detection Systems through Dataset Aggregation. *IEEE Transactions on Emerging Topics in Computing*.

Fu, Y., Du, Y., Cao, Z., Li, Q., & Xiang, W. (2022). A Deep Learning Model for Network Intrusion Detection with Imbalanced Data. *Electronics*, *11*(6), 898.

Antunes, M., Oliveira, L., Seguro, A., Veríssimo, J., Salgado, R., & Murteira, T. (2022, March). Benchmarking Deep Learning Methods for Behaviour-Based Network Intrusion Detection. In *Informatics* (Vol. 9, No. 1, p. 29). MDPI.

Moizuddin, M. D., & Jose, M. V. (2022). A bio-inspired hybrid deep learning model for network intrusion detection. *Knowledge-Based Systems*, *238*, 107894.

Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, *27*(1), 357-370.

Ahmad, Iftikhar, et al. "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection." IEEE access 6 (2018): 33789-33795.

Al-Yaseen, Wathiq Laftah, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for IDS." Expert Systems with Applications 67 (2017): 296-303.

Wang, Huiwen, Jie Gu, and Shanshan Wang. "An effective intrusion detection framework based on SVM with feature augmentation." Knowledge-Based Systems 136 (2017): 130-139.

Thaseen, Ikram Sumaiya, and Cherukuri Aswani Kumar. "Intrusion detection model using fusion of chi-square feature selection and multi class SVM." Journal of King Saud University-Computer and Information Sciences 29.4 (2017): 462-472.

Gao, Xianwei, et al. "An adaptive ensemble machine learning model for intrusion detection." IEEE Access 7 (2019): 82512-82521.

Zhang, Wenjie, et al. "Wireless sensor network intrusion detection system based on MK-ELM." Soft Computing 24.16 (2020): 12361-12374.

Raman, MR Gauthama, et al. "An efficient intrusion detection system based on hypergraph-Genetic algorithm for parameter optimization and feature selection in support vector machine." Knowledge-Based Systems 134 (2017): 1-12.

Ali, Mohammed Hasan, et al. "A new intrusion detection system based on fast learning network and particle swarm optimization." IEEE Access 6 (2018): 20255-20261.