Detecting Network Intrusions Using Convolutional Neural Networks

Nur Mahabub Morshedur Rahman
United International University
Dhaka, Bangladesh
nrahman171291@bscse.uiu.ac.bd

Mahbub-E-Sobhani
United International University
Dhaka, Bangladesh
msobhani171134@bscse.uiu.ac.bd

ABSTRACT

The world's dependency on the internet is ever-growing. Information is the most valuable resource in the world. To protect information from unwanted hands is of utmost importance. Information gets stolen when a network gets breached. An intrusion detection system finds known and unknown attacks that facilitate breaching of a network. In this paper, we model an intrusion detection system trained to detect such types of attack. We do so using neural networks which is a machine learning approach. Furthermore, we study the performance of our model in binary and multiclass classification environments. We look into the performance of other traditional machine learning algorithms such as Support Vector Machine, Naive Bayes, K-Nearest Neighbour and Random Forest on the data set (NSL-KDD) we use. Our experiments show that using Convolutional Neural Networks are better approaches to detect network intrusions.

1 INTRODUCTION

Society and the internet integrate deeply. This dependency is every-growing, and for that enormous quantity of data is being produced. Security of information is a significant concern and to identify attacks on networks, especially attacks that are entirely new and unseen, is a necessary need. An intrusion detection system (IDS) can identify such attacks, whether ongoing or an attack that has already occurred. Signature-based systems are the traditional ways of detecting network intrusions, and they use signatures and patterns of attacks that have already occurred before. They are very accurate when detecting known security threats but are entirely ineffective in identifying unknown malicious threats. So, real-time threat detection scenarios do not use traditional detection systems. The self-learning system is a proactive method that uses machine learning concepts such as supervised and unsupervised algorithms to be on par with the ever-changing network attack techniques. Such a machine learning methods is Convolutional Neural Networks which have already been successful in fields like natural language processing (NLP) and computer vision (CV). Inspired by the performance of the method, we propose an intrusion detection system using it (CNN-IDS). The contributions of this paper are:

(1) We design and implement the CNN-IDS.
(2) We study the performance of our model in binary and multiclass classifications.
(3) We compare the performance of our models with that of Support Vector Machine, Naive Bayes, K-Nearest Neighbour and Random Forest on the data set we use (NSL-KDD) in binary and multiclass classification environments.

2 BACKGROUND

In this chapter we are presenting the primary studies in 2.1 along side with research papers that is aligned with our project work.

2.1 Preliminaries

This precursory segment introduces to the Grammatical Error Correction problem concepts and keywords that are required to have a full idea of understanding the rest of the sections without any difficulty.

Recurrent Neural Network

Recurrent Neural Network, also known as RNNs are a part of neural networks which is used for sequential prediction. RNNs allow previous outputs to be used as inputs and has the ability to process an input of any length by not increasing the model size where weights are shared across time.

Bidirectional Recurrent Neural Network

Bidirectional Recurrent Neural Networks is a variant of RNN. Which has all the advantages of basic RNN architecture with additional speed and capturing the ability of long-term dependencies. The ability of considering the future and past output for processing the current output made Bidirectional RNN so special for Natural Language Processing problems.

Long Short Term Memory

Long Short Term Memory, also known as LSTMs is a modified version of RNN having the ability of learning long-term dependencies. LSTM deals with the vanishing gradient problem encountered by traditional RNN. Remembering the information for long periods of time is the default behavior of LSTM. It has the advantage of forgetting something from the previous information when it needs to, and updates the necessary information for the future time steps.

Convolutional Neural Network

Convolutional Neural Network is most often used for image processing problems, but nowadays it has been discovered that CNN can also be useful for Natural Language Processing problems. Convolutional neural network is widely used for important contributions in deep learning and artificial intelligence. CNN is made up of multiple layers and has the fast processing ability because of powerful feature extraction and reduction techniques. The convolutional neural network is a feed-forward network that filters spatial data whereas recurrent neural networks feed data back to itself, and that is why it works well on sequential data.

3 RELATED WORK

In previous studies, traditional machine learning approaches such as Support Vector Machine, Naive Bayes, K-Nearest Neighbour and Random Forest have been successful in intrusion detection systems.[1] In recent years, a division of machine learning known as deep learning has been famous in this field due to its unparallel performance. Studies show that deep learning is better in terms of performance in detecting unknown network attacks than traditional methods.[2]

The authors of [1] have tested the effectiveness of Naive Bayes, Random Forest, Support vector machine and K-Nearest Neighbors on both encrypted and unencrypted network packets. With a dataset consisting of 1130 instances collected through packets captured by Wireshark, they have determined amongst all other algorithms, Random Forest is the algorithm which more accurate.

In [3], the researcher has experimented with all the methods of [1], but with a different dataset. They have used the popular KDD99 dataset, which is the precursor of the NSL-KDD dataset that we have used. With 21 types of attacks and 41 features in the dataset, they have used Information gain as their feature selection method. They have also concluded like [1] that Random Forest is more accurate than all the traditional machine learning algorithms.

Language	English
Algorithms	Naive Bayes, Random Forest, K-Nearest Neighbor, Decision Table
Dataset	KDD99
Methodology	1. Importing the dataset 2. Classifying and choosing the algorithm. 3. Using the 10-Fold method 4. Again testing using the Percentage Split (70%) 5. Checking the Correctly Classified Instance Percentage.
Evaluation Metrics	10-Fold cross validation, 70% percentage split
Results	10-Fold cross validation Naive Bayes: 76.16% Random Forest: 99.40% K-Nearest Neighbor: 98.94% Decision Table: 98.51%

[4] has applied CNN to the KDD99 dataset. They have found that a fixed learning rate of 0.1 performs the best. The paper claims that a deep learning-based approach such as CNN is suitable at modelling network traffic in comparison to other conventional machine learning classifiers. The authors of this paper say that the NSL-KDD dataset is outdated and have suggested other researches to use UNSW-NB15[5] dataset instead.

Language	English
Algorithms	CNN, CNN-LSTM, CNN-GRU, CNN-RNN
Dataset	KDD99
Methodology	Tensorflow > CNN 1 Layer > CNN 2 Layer > CNN 3 Layer > Evaluation
Evaluation Metrics	Accuracy, Precision, Recall, F-score
Results	Average score of CNN 3 layer-RNN Accuracy: 0.938 Precision: 0.997 Recall: 0.926 F-score: 0.960

In [6], a way of increasing accuracy and performance has been proposed. The paper identified relevant features inside the dataset, and the accuracy rate was improved. The researchers eliminated Redundant and irrelevant features which significantly improved classifier performance.

Language	English
Algorithms	Method "SelectPercentile" in the sklearn.feature_selection module
Dataset	KDD99
Methodology	1. Data Cleaning and Pre-processing 2. Features scaling 3. Features Selection 4. Model 5. Prediction and Evaluation
Evaluation Metrics	10-fold cross-validation
Results	Performance evaluation with 41 features for Dos Accuracy: 99.66 Precision: 99.505 Recall: 99.71 F-measure: 99.61 Performance evaluation with selected features for Dos Accuracy: 99.90 Precision: 99.69 Recall: 99.79 F-measure: 99.74

Finally, in [7], the authors reduced the amount of data in the NSL-KDD dataset by 80.4%, which reduced training time by 40% and testing time by 70%.

We have constructed our intrusion detection system using CNN and the NSL-KDD dataset. Then we compared the accuracy of other machine learning models when applied to this dataset.

4 NORMALIZATION

Although the processed features are already trainable, the numerical differences in the records are large, which will affect the convergence speed and training effect of the model. Therefore, the dataset needs to be normalized so that the data in the sample falls within the range of [0, 1]. Since the datasets contained both normal and anomalous traffic, we need to avoid the negative influence of the

sample mean and variance. For general numerical features we have used min-max scalar.For the features duration, srcbytes, and dstbytes where the data ranges are large, logarithmic normalization is required.

5 DATA CLUSTERING

The existing deep learning based intrusion detection methods usually directly map the preprocessed one-dimensional numerical features into corresponding two-dimensional matrices, and fill the redundant parts with zeros. Although this method is simple and straightforward, it ignores a very important issue – the added relevance. The transformed two-dimensional matrix is similar to a grayscale image and will inevitably impose a correlation in the vicinity of the matrix elements. This will seriously affect the model training and weaken the adaptability of the model. To reduce the impact of imposing correlation, this paper proposes a novel method – data clustering. Specifically, for data containing mfeatures, it is divided into nparts according to prior knowledge or common clustering methods, where m > n. Then the different parts of the data are processed separately. According to the correlation between features, this paper divides the feature data into four parts, the first part is the basic features; the second part is the content features; the third part is the time-based network traffic statistics features; and the fourth part is the host-based network traffic statistics features. The data clustering can help to learn the high level relationships between the global features that would be ignored by other classification algorithms. By separating the less relevant features, the effects of additional additive correlations can be effectively reduced.

6 CONVERT TO MATRIX

Since the convolutional neural network has a better image processing ability, this paper converts the inputs into the form of images. The advantages of CNNs can be better exploited by transforming intrusion detection problems into image classification problems. The first part of the data contains 90 features that can be directly converted into a corresponding 9 x 10 matrix. The second, third and fourth parts respectively contain 11, 9, and 10 features, and the numbers of features are too small to be directly converted into a matrix. Therefore, this paper proposes a simple and effective method to expand these part into a matrix using randomly repeating features. Specifically, with nfeatures, this paper constructs an n x n matrix using a random method that makes each feature appear at each position in the matrix. This method can effectively extract the correlations between features and improve the network's ability to acquire structural information of the feature.

7 TRAINING MODEL

For the different parts of the dataset, we use the same CNN structure. Taking the first part of the dataset as an example, the architecture of the CNN model that is implemented for intrusion detection in the binary classification and the dimensions of each layer are shown in figure 1. The single model consists of an input layer, two convolution and pooling layers, three fully connected layers and an output layer. For regularization, a dropout layer with the dropout rate of 0.5 is adopted between the flattened model and the first fully connected layer to control over-fitting. The activation function of each hidden layer is Rectified Linear Unit (ReLU). The input layer converts the flow data into a 2-dimensional matrix. The second layer is the convolution layer, and the input data are extracted using 32 filters with zero-padding. The third layer is the max-pooling layer, and the data are checked with 22 down sampling with a step size of 1. The fourth layer is similar to the second layer, except that the number of convolution kernels becomes 64. A softmax layer in the last is employed as the output of the classifier, and noting that the layer is different for the binary and multiclass classifications. The cross-entropy cost function is used as the loss function to be minimized by training. The batch size, the optimizer, and the learning rate used in training the networks are 128, Adam, and 0.001, respectively

8 PROPOSED METHODOLOGIES

8.1 Convolutional Neural Network (CNN)

CNN has two operations which are convolution and pooling. Convolution changes input data to output using a set of kernels or filters. The produced output showcases the features of the input data. That is why the output is known as the feature map. An activation function processes the convolution output further and down-sampling trims off irrelevant data using pooling. Pooling removes glitches in the data. That is how the learning improves for the following layers.[8][9] CNN adjusts the kernels/filters using rounds after rounds of learning so that the feature map can functionally represent the input data. We use ID convolution since the network packet in the dataset is represented in an ID format.

$$(f * g)(i) = \sum_{m}^{j=1} g(j).f(i - j + m/2)$$

Here, i denotes the position of the values in sequence data. The activation function is ReLU. Finally, we use max pooling.

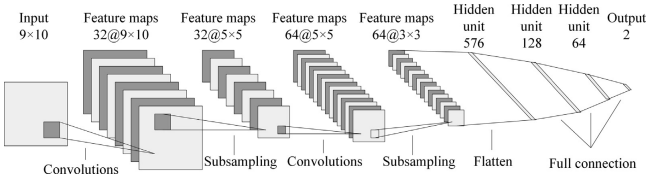

Figure 1: An example of the architecture of a single CNN model

In traditional neural network models, data go from the input layer to the hidden layer to the output layer. The layers are fully connected, and there is no connection between nodes in the same layer. Therefore, traditional neural networks have many problems that cannot be solved. The convolutional neural networks [12], which improve upon the architecture of the common neural network, have already yielded impressive achievements in the fields of speech analysis and image classification in recent years [13]. The CNN consists of one or more convolutional layers and pooling layers at the top, as well as fully connected layers and dropout layers [14] which serve as regularization layer. This structure enables the convolutional neural network to take advantage of the two-dimensional

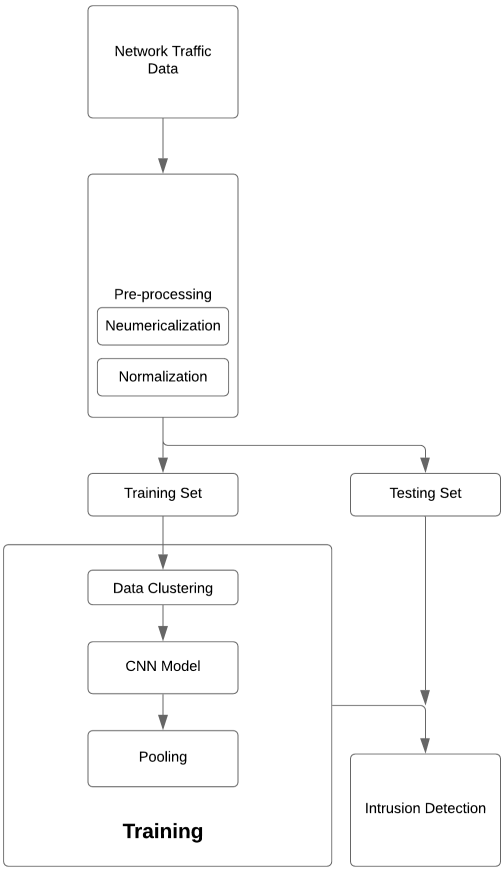

Figure 2: Block diagram of the proposed IDS.

structure of the input data. Therefore, an image can be directly used as an input to the network, thereby avoiding complex feature extraction and data reconstruction operations that exist in traditional recognition algorithms. Through sparse connectivity, shared weights, and pooling, the difficulty of manually processing data can be effectively reduced, and the modeling efficiency is greatly improved. CNN can learn different levels of features from a large amount of unlabeled data. Therefore, the application prospects of CNN in the field of network intrusion detection are very broad. This section detailed describes the multi-CNN fusion architecture that is used in this study and the methodology of its usage for developing intrusion detection models. The diagram of the proposed method is shown in Fig. 2. The steps that are involved in this chapter include dataset description, data preprocessing, specific methodology, and the evaluation metrics. [18]

9 DATASET

The NSL-KDD data set is not the first of its kind. The KDD cup was an International Knowledge Discovery and Data Mining Tools Competition. In 1999, this competition was held with the goal of collecting traffic records. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. As a result of this competition, a mass amount of internet traffic records were collected and bundled into a data set called the KDD'99, and from this, the NSL-KDD data set was brought into existence, as a revised, cleaned-up version of the KDD'99 from the University of New Brunswick.

This data set is comprised of four sub data sets: KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20Percent, although KDDTest-21 and KDDTrain+_20Percent are subsets of the KDDTrain+ and KDDTest+. From now on, KDDTrain+ will be referred to as train and KDDTest+ will be referred to as test. The KDDTest-21 is a subset of test, without the most difficult traffic records (Score of 21), and the KDDTrain+_20Percent is a subset of train, whose record count makes up 20% of the entire train dataset. That being said, the traffic records that exist in the KDDTest-21 and KDDTrain+_20Percent are already in test and train respectively and aren't new records held out of either dataset.

These data sets contain the records of the internet traffic seen by a simple intrusion detection network and are the ghosts of the traffic encountered by a real IDS and just the traces of its existence remains. The data set contains 43 features per record, with 41 of the features referring to the traffic input itself and the last two are labels (whether it is a normal or attack) and Score (the severity of the traffic input itself).

Within the data set exists 4 different classes of attacks: Denial of Service (DoS), Probe, User to Root(U2R), and Remote to Local (R2L). A brief description of each attack can be seen below:

- DoS is an attack that tries to shut down traffic flow to and from the target system. The IDS is flooded with an abnormal amount of traffic, which the system can't handle, and shuts down to protect itself. This prevents normal traffic from visiting a network. An example of this could be an online retailer getting flooded with online orders on a day with a big sale, and because the network can't handle all the requests, it will shut down preventing paying customers to purchase anything. This is the most common attack in the data set.
- Probe or surveillance is an attack that tries to get information from a network. The goal here is to act like a thief and steal important information, whether it be personal information about clients or banking information.
- U2R is an attack that starts off with a normal user account and tries to gain access to the system or network, as a super-user (root). The attacker attempts to exploit the vulnerabilities in a system to gain root privileges/access.
- R2L is an attack that tries to gain local access to a remote machine. An attacker does not have local access to the system/network, and tries to "hack" their way into the network.

It is noticed from the descriptions above that DoS acts differently from the other three attacks, where DoS attempts to shut down a system to stop traffic flow altogether, whereas the other three attempts to quietly infiltrate the system undetected.

In the table below, a breakdown of the different subclasses of each attack that exists in the data set is shown:

@default@default

Figure 3: Multiclass Attacks in NSL-KDD

Although these attacks exist in the data set, the distribution is heavily skewed. A breakdown of the record distribution can be seen in the table below. Essentially, more than half of the records that exist in each data set are normal traffic, and the distribution of U2R and R2L are extremely low. Although this is low, this is an accurate representation of the distribution of modern-day internet traffic attacks, where the most common attack is DoS and U2R and R2L are hardly ever seen.

Dataset	Number of Records:					
	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+20%	25192	13449 (53%)	9234 (37%)	2289 (9.16%)	11 (0.04%)	209 (0.8%)
KDDTrain+	125973	67343 (53%)	45927 (37%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
KDDTest+	22544	9711 (43%)	7458 (33%)	2421 (11%)	200 (0.9%)	2654 (12.1%)

Figure 4: Number of records in NSL-KDD

The features in a traffic record provide the information about the encounter with the traffic input by the IDS and can be broken down into four categories: Intrinsic, Content, Host-based, and Time-based. Below is a description of the different categories of features:

- Intrinsic features can be derived from the header of the packet without looking into the payload itself, and hold the basic information about the packet. This category contains features 1–9.
- Content features hold information about the original packets, as they are sent in multiple pieces rather than one. With this information, the system can access the payload. This category contains features 10–22.
- Time-based features hold the analysis of the traffic input over a two-second window and contains information like how many connections it attempted to make to the same host. These features are mostly counts and rates rather than information about the content of the traffic input. This category contains features 23–31.
- Host-based features are similar to Time-based features, except instead of analyzing over a 2-second window, it analyzes over a series of connections made (how many requests made to the same host over x-number of connections). These features are designed to access attacks, which span longer than a two-second window time-span. This category contains features 32–41.

The feature types in this data set can be broken down into 4 types:

- 4 Categorical (Features: 2, 3, 4, 42)
- 6 Binary (Features: 7, 12, 14, 20, 21, 22)
- 23 Discrete (Features: 8, 9, 15, 23–41, 43)
- 10 Continuous (Features: 1, 5, 6, 10, 11, 13, 16, 17, 18, 19)

A breakdown of the possible values for the categorical features can be seen in the table below. There are 3 possible Protocol Type values, 60 possible Service values, and 11 possible Flag values.

Protocol Type (2)	Service (3)				Flag (4)
• icmp • tcp • udp	• other • link • netbios_ssn • smtp • netstat • ctf • ntp_u • harvest • efs • klogin • systat • exec • nntp • pop_3 • printer • vmnet • netbios_ns	• urh_i • ssh • http_8001 • iso_tsap • aol • sql_net • shell • supdup • auth • whois • discard • sunrpc • urp_i • Rje • ftp • daytime • domain_u • pm_dump	• time • hostnames • name • ecr_i • bgp • telnet • domain • ftp_data • nnsp • courier • finger • uucp_path • X11 • imap4 • mtp • login • tftp_u • kshell	• private • http_2784 • echo • http • ldap • tim_i • netbios_dgm • uucp • eco_i • Remote_job • IRC • http_443 • red_i • Z39_50 • Pop_2 • gopher • Csnet_ns	• OTH • S1 • S2 • RSTO • RSTRs • RSTOS0 • SF • SH • REJ • S0 • S3

Figure 5: Features of NSL-KDD

Unlike Protocol Type and Service whose values are self-explanatory (these values describe the connection), Flag is not very easy to understand. The Flag feature describes the status of the connection, and whether a flag was raised or not. Each value in Flag represents a status a connection had and the explanations of each value can be found in the table below: [17]

Flag	Value	Flag	Description
SF	Normal establishment and termination. Note that this is the same symbol as for state S1. You can tell the two apart because for S1 there will not be any byte counts in the summary, while for SF there will be	RSTO	Connection reset by the originator
REJ	Connection attempt rejected	RSTR	Connection reset by the responder
S0	Connection attempt seen, no reply	OTH	No SYN seen, just midstream traffic (a "partial connection" that was not later closed)
S1	Connection established, not terminated	RSTOS0	Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder
S2	Connection established and close attempt by originator seen (but no reply from responder)	SH	Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was "half" open)
S3	Connection established and close attempt by responder seen (but no reply from originator)	SHR	Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator. (Not in NSL-KDD but still a flag)

Figure 6: The values of the feature "Flag" in NSL-KDD

10 DATA PREPROCESSING

10.1 Numericalization

In the NSL-KDD dataset, there are three features which are non-numeric and 38 numeric features. Since the input values must be numeric, we convert the non-numeric features into numeric. For example, the feature 'protocol_type' can have three different types of attributes which are 'tcp', 'udp', and 'icmp'. We encode them as binary vectors (0,0,1), (0,1,0) and (1,0,0). This way, we convert the 41-dimensional feature map into a 122-dimensional feature map.

10.2 Normalization

There are several features in the dataset in which the difference between the max and min values are large. Such features are dst_bytes[0,1.3 × 109], src_bytes[0,1.3 × 109] and duration[0,58329]. We apply the logarithmic scaling method to lower the differences and then use the formula below to map them to the [0,1] range:

xi = (xi - Min) / (Max - Min)

10.3 Feature Selection

The features in a traffic record provide the information about the encounter with the traffic input by the IDS and can be broken down into four categories: Intrinsic, Content, Host-based, and Time-based. We have considered the whole dataset for training as a four part dataset for classifying the attacks. There are many highly correlated features which hampers the model in classifying in the validation and testing time. So, we split the dataset in four part and took the first part as hybrid

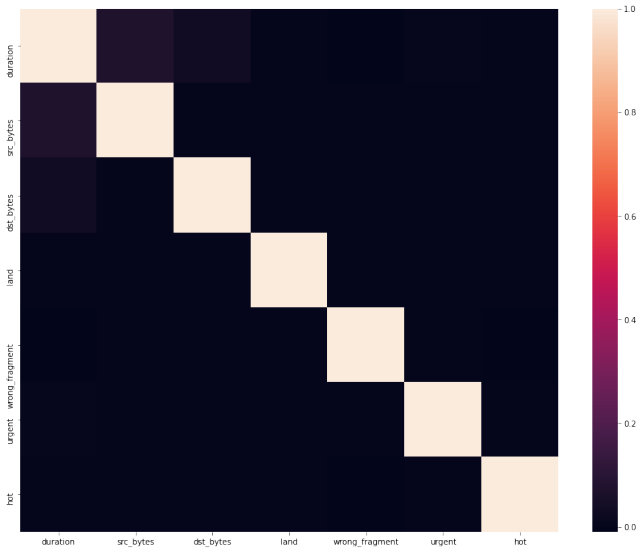

Figure 7: Selecting the most relevant features of NSL-KDD

features. Intrinsic features can be derived from the header of the packet without looking into the payload itself, and hold the basic information about the packet. First 10 features of each packet gives the model more generalization and inference ability which helped us to go through all this way with a better accuracy than other models.

10.4 Evaluation Metrics

We use Accuracy (AC) to measure the performances of our model. Furthermore, we also introduce false positive rate and detection rate. True Positive (TP) denotes the number of records that rejects correctly and identifies as anomalies. Whereas, True Negative (TN) denotes the opposite. True Negative (TN) denotes the correct records that are normal and False Negative (FN) denotes the opposite. We follow the notations given below.

Accuracy, AC = (TP + TN) / (TP + TN + FP + FN)
True Positive Rate, TPR = TP / (TP + FN)
False Positive Rate, FPR = FP / (FP + TN)
True Negative Rate, TNR = TN / (TN + FP)
False Negative Rate, FNR = FN / (FN + TP)
Sensitivity, FPR = TP / (TP + FN)
Specificity, FPR = FN / (TN + FP)

Therefore, our motivation is to get high accuracy and better detection rate with low false positive.

10.5 Binary Classification

We have mapped 41-dimensional features into 83-dimensional features. Therefore, the CNN-IDS model has 122 input nodes and 2 output nodes in the binary classification experiment. We take the number of epochs as 100 and the learning rate as 0.01. To find the better model, let the number of hidden nodes be 60, 80 and 120, respectively. The no. of hidden layers is 2, and the batch size is 64. From the table below, we have determined that a hidden node value of 64 achieves the best result.

	anomaly	normal
anomaly	8721	990
normal	2576	10238

Table 1: Confusion matrix of 2-category classification on KDDTest+

The experiments show that the CNN-IDS model works with a good detection rate (84.16%) when given 100 epoch for the KDDTrain dataset.

The results obtained by J48,Naive Bayesian, Random Forest, Multi-layer Perceptron,Support Vector Machine and the other classification algorithms, and the artificial neural network algorithm also gives 81.2%, which is the recent literature about ANN algorithms applied in the filed of intrusion detection. Fortunately, these results are all based on the same benchmark - the NSL-KDD dataset

In [3], the maximum accuracy that the authors have achieved using the traditional methods is 99.40%. The performance of CNN-IDS model is superior to other classification algorithms in binary classification.

Our model		
Train	Train	Test
96.22%	84.16%	70.26%

Table 2: Convolutional Neural Network Accuracy for Binary Classification

10.6 Multiclass Classification

Inspired by [4], We have used both Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) for multiclass classification. In CNN, the number of hidden layers that we have taken is 2. The number of epochs is 15 and 150 respectively. In RNN, almost all the values are similar to those used in the binary classification experiment. The difference is that the number of hidden nodes is taken as 80 and 120 respectively. It is observable from the table below that CNN with the number of epochs being 150 gains the best results.

	Normal	DoS	R2L	U2R	Probe
Normal	9017	633	53	2	6
DoS	126	980	0	0	0
R2L	93	101	5528	0	0
U2R	33	0	0	2	2
Probe	1823	0	0	0	376

Table 3: Confusion Matrix for Multiclass Classification

In this experiment,the detection rate of the CNN-IDS model gets higher accuracy testing dataset, not only higher than the detection rate on the NSL-KDD dataset, but also higher than other neural network models. The experimental results show that the fully connected model has stronger modeling ability and higher detection rate than the reduced-size CNN model.

Our model		
Train	Validation	Test
99.11	84.70	65.45

Table 4: CNN model Accuracy for Multiclass Classification

The accuracy of 99.11 that this model has achieved is better than both [4]'s 93.8%.

11 EXPERIMENT RESULTS & DISCUSSION

We have used Google Collab's Free Tier plan to conduct all of our experiments. We use two experiments to study the performance of the IDS model. First, binary classification. And secondly, five-category classification such as Normal, DoS, R2L, U2R and Probe.

	KDDTest+	KDDTest-21
Naive Bayes	76.56	55.77
J48	81.05	63.97
NB Tree	82.02	66.16
Random Forest	80.67	63.26
SVM	69.52	42.29
RNN	83.28	68.55
Our Model CNN	84.16	70.26

Table 5: Performance of the CNN model and other traditional machine learning models in the binary classification.

In order to compare the performance of different classification algorithms on the benchmark dataset for the multicalss classification as the binary classification experiments J48, Naive Bayesian, Random Forest, Multi-layer Perceptron, Support Vector Machine and other machine learning algorithms are used to train models through the training set (using 10-layer cross-validation) by mean of the opensource machine learning and data mining software Weka. We then apply the models to the testing set. The results are described in Fig. 8. Compared with the binary classification, the accuracy of classification algorithms is declined in the five-category classification. Table 3 shows the confusion matrix of the CNN-IDS on the test set KDDTest+ in the five-category classification experiments. The experiment shows that the accuracy of the model is 84.70% for the test set KDDTest+ and 65.45% for KDDTest21, which is better than those obtained using J48, naive bayes, random forest, multi-layer perceptron and the other classification algorithms.

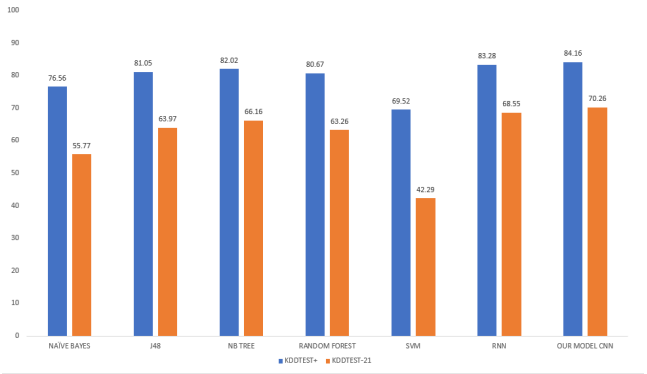

Figure 8: Binary Class Classification

In the binary classification experiments, we have compared the performance with Naive Bayes, Support vector machine, Random Forest and K-Nearest Neighbors models.

	KDDTest+	KDDTest-21
Naive Bayes	74.40%	55.77%
J48	74.60%	51.90%
NB Tree	75.40%	55.77%
Random Forest	74.00%	50.80%
SVM	74.00%	50.70%
RNN	81.29%	64.67%
Our Model CNN	84.70%	65.45%

Table 6: Performance of the CNN model and other traditional machine learning models in the multiclass classification.

We have evaluated the performance of different classification and neural network models to compare with our proposed model and we have found that we have achieved better result then other model by using Adam optimizer and epoch number 500.

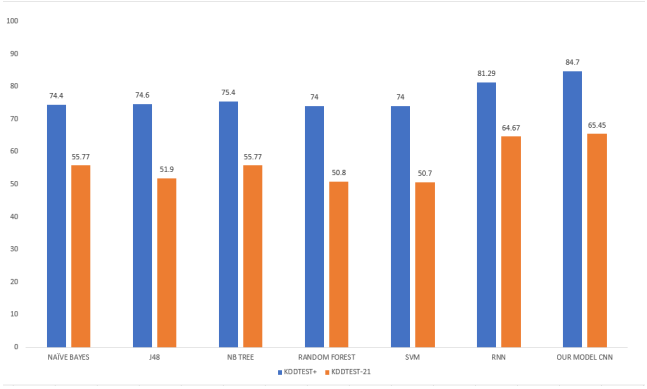

Figure 9: Multi Class Classification

In the same way, we analyze the performance of multi-classification of the CNN-IDS model against Naive Bayes, Support vector machine, Random Forest and K-Nearest Neighbors and RNN.

Based on the same benchmark, using KDDTrain+ as the training set and KDDTest+ and KDDTest-21 as testing set, the experimental result shows that both binary classification and multiclass classification the intrusion detection model of CNN-IDS has better test result and accuracy than the other machine learning models and maintains a high accuracy rate, even in the multiclass classification problem. Of course, the model we proposed will spend more time for training, but using GPU acceleration can reduce the training time.

12 CONCLUSION

Our experiment results say that in both binary classification and multiclass classification, the intrusion detection model using neural networks achieve higher accuracy then tradition machine learning models using the same dataset. Though our models require more computation time, additional hardware can reduce that to a great extent.

13 FURTHER RESEARCH

Using the research done in paper [6], redundant and irrelevant features can be removed, which can significantly improve classifier performance. By identifying relevant features inside the dataset, accuracy increases. Furthermore, the authors of [4] suggested the use of UNSW-NB15, which removes the inherited issues of KDDCup 99 and NSL-KDD.[7] Finally, Principal component analysis (PCA) can be used, as shown in paper [7], to drastically reduce the number of features, training and testing time. Combining these three ideas, we believe that we can get further accuracy and performance improvements to our models.

REFERENCES

[1] I Sumaiya Thaseen, B Poorva, and P Sai Ushasree. Network intrusion detection using machine learning techniques. In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pages 1–7. IEEE, 2020.
[2] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," presented at the 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol. (BIONETICS), New York, NY, USA, May 2016, pp. 21–26.
[3] Kirty, A. D. (2018, March). Network Intrusion Detection using Machine Learning. Website. https://github.com/Anshumank399/Network-Intrusion-Detection-using-Machine-Learning/blob/master/Final%20Report.pdf
[4] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Applying convolutional neural network for network intrusion detection", Proc. Int. Conf. Adv. Comput. Commun. Inform., pp. 1222-1228, 2017.
[5] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6, IEEE, 2015.
[6] Herve. Nkiama, Syed. Zainudeen Mohd Said and Muhammad. Saidu, "A Subset Feature Elimination Mechanism for Intrusion Detection System", International Journal of Advanced Computer Science and Applications, vol. 7, no. 4, 2016.
[7] S. Lakhina, S. Joseph and B. Verma, "Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD", Int. J. Eng. Sci. Technol., vol. 2, no. 6, pp. 3175-3180, 2010.
[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.
[9] Y. Bengio, I. Goodfellow, and A. Courville, Deep learning, vol. 1. Citeseer, 2017.
[10] NSL-KDD dataset. Website. https://www.unb.ca/cic/datasets/nsl.html
[11] Moustafa, N., & Slay, J. (2015b). UNSW-NB15: A comprehensive data set for network intrusion detection. 2015 Military Communications and Information Systems Conference. Canberra, Australia: MilCIS 2015-IEEE Stream.
[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[13] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recogn. 77 (2018) 354–377.
[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[15] Wenjie Zhang, Dezhi Han, Kuan-Ching Li, Francisco Isidro Massetto. "Wireless sensor network intrusion detection system based on MK-ELM" , Soft Computing, 2020
[16] "Cloud Computing – CLOUD 2018" , Springer Science and Business Media LLC, 2018
[17] Saporito, Gerry. "A Deeper Dive into the NSL-KDD Data Set" Medium, https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657. Accessed 17 September 2019.
[18] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, et al., "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion", Measurement, vol. 154, Mar. 2020.
[19] C. L. Yin, Y. F. Zhu, J. L. Fei and X. Z. He, "A deep learning approach for intrusion detection using recurrent neural networks", IEEE Access, vol. 5, pp. 21954-21961, 2017.
[20] Filiz Türkoğlu. "Author Attribution of Turkish Texts by Feature Mining" , Lecture Notes in Computer Science, 2007
[21] Md. Abdul Wahab. "Strain effect on single and double walled carbon nanotubes" , TENCON 2009 - 2009 IEEE Region 10 Conference, 2009
[22] Chidchanok Lursinsap. "Decision tree induction based on minority entropy for the class imbalance problem" , Pattern Analysis and Applications, 2016

[23] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, Chunhua Wang. "Machine Learning and Deep Learning Methods for Cybersecurity" , IEEE Access, 2018

[24] Herve Nkiama, Syed Zainudeen, Muhammad Saidu. "A Subset Feature Elimination Mechanism for Intrusion Detection System" , International Journal of Advanced Computer Science and Applications, 2016