1D-CNN based Model for Classification and Analysis of Network Attacks

Kuljeet Singh[1]
Research Scholar
Department of Computer Science & IT, University of Jammu
Jammu, India

Amit Mahajan[2]
System Analyst
Department of Computer Science & IT, University of Jammu
Jammu, India

Vibhakar Mansotra[3]
Professor
Department of Computer Science & IT, University of Jammu
Jammu, India

***Abstract*—With the advancement in technology and upsurge in network devices, more and more devices are getting connected to the network leading to more data and information on the network which emphasizes the security of the network to be of paramount importance. Malicious traffic must be detected in networks and machine learning or more precisely deep learning (DL), which is an upcoming approach, should be used for better detection. In this paper, Detection of attacks through a classification of traffic into normal and attack data is done using 1D-CNN, a special variant of convolutional neural network (CNN). For this, the CICIDS2017 dataset consisting of 14 attack types spread across 8 different files, is considered for evaluating model performance and various indicators like recall, precision, F1-score have been utilized. Separate 1D-CNN based DL models were built on individual sub-datasets as well as on combined datasets. Also, an evaluation of the model is done by comparing it with an artificial neural network (ANN) model. Experimental results have demonstrated that the proposed model has performed better and shown great capability in detecting network attacks as the majority of the class labels had achieved excellent scores in each of the evaluation indicators used**.

Keywords*—1D-CNN; CICIDS2017; network attacks; deep learning

I. INTRODUCTION

The Internet has become a major aspect in today's society with people using the services of WWW for most of their day-to-day activities. People use the Internet for both personal as well as professional purposes and the majority of tasks include sharing data, information access, sending files, connecting with friends or colleagues through social media and most importantly e-commerce activities which include saving passwords, credit/debit card info. Not only individuals but organizations too depend heavily on the Internet and its services. Network attacks in the form of malicious traffic results in loss of data, privacy violation for individuals; monetary, financial and political impact on big organizations and interrupted businesses for all its shareholders [1]. Nowadays, with the effect of Covid-19 and the ongoing pandemic, work from home is becoming a new normal. This has led to personal devices being vulnerable with not so sophisticated protection mechanisms as compared to the organization's resources. The effect of the pandemic could lead to attack mechanisms getting more diverse which means cyber-security will remain verticals of critical importance in the times to come.

There are many approaches to provide cyber-security ranging from authentication, encryption to a firewall, IDS (intrusion detection system). With IDS providing monitoring and behavior analysis of network traffic and further identifying attacks from network flow, it has proven itself a better alternative to other approaches [2]. Detection of cyber-attacks is like a classification approach where it categorizes whether it belongs to benign or different types of attacks. Traditional Machine learning (ML) techniques, also known as shallow learning, have been used for intrusion detection by classifying the network traffic [3]. As the real world data gets bigger by time resulting in high dimensional space, the drop in performance of ML techniques can be observed due to its over-dependence on the features selected by the human experts. DL, with its complex architecture, overcame this limitation by automatically learning features through a massive amount of data. In this paper, we propose a 1D-CNN as a DL technique for effective feature representation and categorizing traffic into normal and different attack types. 1D-CNN's or 2D-CNN are almost identical in architecture as the core process in both of them is convolution operation.

Convolution, a mathematical operation operates on two signals by convolving them with one being input signal or data and the other known as kernel or filter. It is the process between input and kernel/filter which includes element-wise multiplication followed by summation resulting in single/scalar value. Convolution can be 1-d, 2-d or multi-dimensional depending on the problem in hand but the traditional CNNs developed [4] and the popular ones employed uses 2-d convolution which became the de facto standard for most applications in image processing and other deep learning tasks [5] [6] [7]. CNNs consist of input layer, convolution layers in the initial stages and MLP or fully connected layers in the final stages of a model preceding the output layer. The other optional but mostly used layers include sub-sampling (pooling) layer and dropout (regularization technique). General convolution operation is shown in equation 1:

$$O_t = (X*F)_t \quad (1)$$

Where X is the input vector and F is the filter or kernel used and * is the convolution operation employed. The dimensions of both X and F are 1 dimensional in 1D-CNN and subsequently vary with the CNN used.

Generally, the architecture of 1D-CNN and 2D CNN remains the same with the main difference between the two is the use of 1d array or tensor in the former and 2d matrix or tensor in the latter. This means both input data and the kernel used for convolution are in 1d array form and the kernel moves over input in 1d direction. These minor but strategic changes led to certain advantages of 1D-CNN over 2D-CNN like 1) Reduced computational complexity due to 1D tensor over 2D tensor, 2) Well suited for low-cost applications but can be used for complex problems [8]. These advantages of 1D-CNN and better compatibility with certain problem domain has led to many areas where it has been applied or can be applied such as:

- Most extensively used in Natural Language Processing (NLP), where it is quite helpful in extracting sub-sequences from sequences of words [9].
- Human activity recognition task which involves time series of sensor data [10].
- Analysis of signal data over a fixed-length period, for example, an audio recording, real-time motor fault detection [11].
- Data in tabular form.

The focus of this study is the network traffic data which is stored in tabular form where each record is represented by an individual row which is in a one-dimensional shape. Applying 2D-CNN over this type of data requires converting each 1d row into a 2d matrix shape before convolution between input and kernel can be performed. Application of 2D-CNN over images seems justifiable since images are already in 2d shape but in the case of tabular data, it takes an additional effort of converting the 1d input data (each row) into 2d matrix shape which might include padding as well. This overhead can be avoided by using 1D-CNN over 2D-CNN with the only notable difference between the two being the shape of input data and kernel vector as 1d array (or tensor) is used in the former and 2d matrix (or tensor) in the latter.

The Rest of the manuscript is organized as follows: Section 2 discusses the related work in the field of intrusion detection, Section 3 explains the methodology part which comprise sub-sections 1) dataset description 2) model architecture and 3) model evaluation. Section 4 presents the results and analysis while Section 5 concludes the paper.

II. Related Work

Research on intrusion detection has been going on for many decades, still a lot of work needs to be done and lots of issues must be examined. Several Data mining/ML techniques whether supervised or unsupervised learning have been applied for the identification of malicious traffic [12] [13]. More recently DL techniques have been used for the detection of Cyber-attacks and it has achieved significant results. So our literature review revolves mostly around the DL technique used (especially CNN) or the CICIDS2017 dataset which has been utilized in the proposed work.

Detection and mitigation of the common DDoS attacks using DDoS detectors employed for network traffic monitoring have been carried out using ANN structures, which were designed for different protocols separately [14]. In [15], authors uses NSL-KDD and Kyoto dataset for implementing their work which contains two important concepts: online sequential extreme learning machine which is the methodology used for classification and traffic profiling which makes up the preprocessing part. DL based intelligent framework have been implemented using Long short term memory (LSTM) to lessen DDoS attack in fog environment [16]. ISCX and CTU-13 were the datasets considered along with attack launching tool Hping3 for model evaluation. In [17], the applicability of restricted boltzmann machine (RBM) to differentiate between normal and abnormal Netflow traffic have been demonstrated in the ISCX dataset. A hybrid approach has been adopted in the form of a Double-Layered Hybrid Approach (DLHA) where the first layer uses naïve Bayes (NB) to detect DoS and probe attacks while the second layer adopts SVM for detecting the remaining attacks in the NSL-KDD dataset [18]. In [19], authors proposed a model based on 5-layer autoencoder (AE) for detection of network anomalies. Their work also includes data preprocessing for removing outliers and error reconstruction for effective network traffic classification.

In the detection of network attacks using CNN, the majority of the academic research has been done using 2D-CNN in which input data in the linear form is transformed into a matrix form. In [20] and [21], the proposed approach revised the established LeNet-5 model for classification of attacks in the KDD99 dataset, and input data is converted into 32*32 matrix shapes for input to the model. DNN based IDS was built with 4 hidden layers and evaluated the model using the NSL-KDD dataset [22]. Dimensionality reduction using principal component analysis (PCA) and AE has been performed on the KDD99 dataset before the classification technique CNN is applied [23]. The input shape of 1*122 is transformed into 1*121 and 1*100 before being converted to 10*10 and 11*11 matrix shapes.

Both shallow and deep learning have been combined through the random forest (RF) and non-symmetric deep auto-encoders (NDAE) [24]. They exercised the NDAE technique for unsupervised learning of features, and for classification tasks, a model constructed from a combination of stacked NDAEs and the RF algorithm was implemented. Separate architectures or models were built in the form of CNN, RNN, and different variants of AE [25]. NSL-KDD dataset has been used and each record was converted into 32*32 2d form. Long short term memory (LSTM) is the variant of RNN used while Sparse, Denoising, Contractive, and Convolutional are different variants of AE used in the experiments. In [26], authors utilized the 1D-CNN based model for intrusion detection further evaluated using the NSL-KDD dataset. They compared the performance of their proposed model with different ML/DL techniques like J48, NB, RF, MLP, and RNN. In [27], authors proposed BAT as a traffic anomaly detection model for effective feature representation and network classification. The BAT model is a combination of a Bidirectional LSTM and attention mechanism.

The use of the CICIDS2017 dataset for intrusion detection has also been found in the literature. The author in his thesis

has done integration of open-source anomaly-based IDS Zeek (Bro), which uses scripts for feature extraction, and developed a model using various algorithms like RF, DT, and KNN on the CICIDS2017 dataset [28]. An ML-based hybrid model was recommended which comprises DT and RF in a stacked manner for classifying attacks in CICIDS2017 and NSL-KDD dataset [29]. The author incorporates the Fisher score as the feature selection method and performed the analysis of Supervised Learning techniques like DT, KNN, and SVM in detecting DDOS attacks from the CICIDS2017 dataset [30]. Experimental results have shown a good detection rates for DT and KNN but mediocre classification results for models built using SVM. In [31], authors applied and performed comparative analysis of 10 common ML/ DL techniques for detecting web attacks. The employed techniques include ANN, DT, KNN, SVM, CNN, NB, RF, k-means, expectation maxim and SOM. The results of the experiment conducted have shown that the NB, KNN and DT has outperformed the other models. Table I summarizes the key existing studies done in the detection of network attacks using ML or DL.

TABLE I. SUMMARY OF THE EXISTING STUDIES

Ref	Algorithm or model	Dataset used	Key points
[15]	LSTM	ISCX, CTU-13	• Detection of DDoS attack in fog environment has been done. • Attack launching tool Hping3 utilized for evaluation.
[19], [20]	CNN	KDD99	• Established LeNet-5 model has been implemented. • Each record is converted into a 32*32 matrix shape.
[22]	CNN, AE	KDD99	• Dimensionality reduction using PCA and autoencoders. • Input shape of 1*122 is transformed into 1*121 and 1*100 before converted to 10*10 and 11*11 shape.
[23]	RF, NDAE	KDD99,NSL-KDD	• NDAE is utilized for unsupervised feature learning. • For classification, stacked NDAE and RF have been combined.
[25]	CNN, RF, MLP, RNN	NSL-KDD	• Comparative analysis of different models has been done.
[26]	RF, DT, KNN	CICIDS2017	• Integration of Zeek IDS with ML models done. • Zeek is used to extract features while models for classification.
[29]	DT, KNN, SVM	CICIDS2017 (only DDoS)	• Fisher score is used for selecting optimal features. • Different ML models evaluated for a reduced set of features for detecting DDoS attacks.
[27]	BLSTM, CNN	NSL-KDD	• Combination of BLSTM and attention mechanism is done. • CNN captures local features from traffic data.

From the literature review, it can be observed:

- Majority of the academic research is done using KDD99 and NSL-KDD dataset despite criticism from researchers about it being outdated [32].
- Applicability and deployment of DL in detecting network anomalies is still in infancy stage.
- While implementing CNN, the preferred choice is 2D-CNN although 1D-CNN has better applicability.

III. PROPOSED METHODOLOGY

The proposed 1D-CNN model for classification of attacks consists of four steps:

Step 1: Data preprocessing - This step involves methods to make data suitable for model training.

Step 2: Model Training - Includes specifying the architecture of a model and then train the model.

Step 3: Testing - Testing the model on unobserved data separated from training dataset.

Step 4: Evaluation – Evaluating the model using multiple metrics mentioned.

These steps form the basis for the overall process demonstrated in Fig. 1. First, the dataset is split into 80:20 train/test samples and then preprocessing of data is done on both. Model with basic initial architecture has been built upon which optimization is performed and training samples are then used to train the optimized model. Final model is tested using a test dataset with the help of various evaluation metrics.

These stages in the proposed 1D-CNN model along with description of the dataset used in the process are further elaborated in detail in following sub-sections.

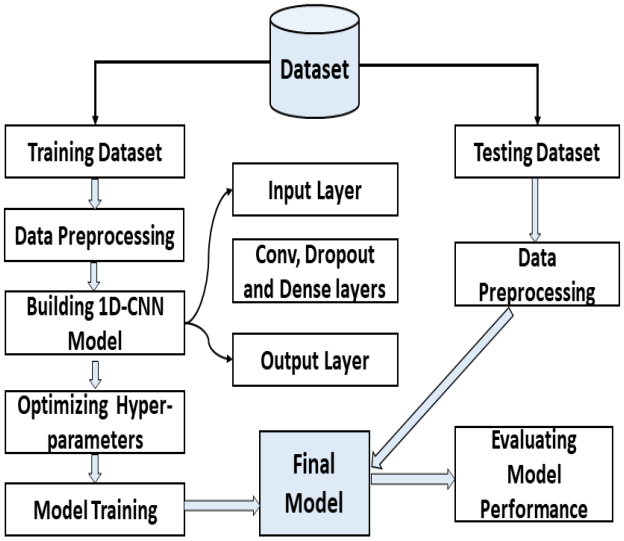

Fig. 1. Flow of the Proposed Methodology.

A. Dataset Description

As already mentioned, the CICIDS2017 dataset, created by the Canadian Institute for Cybersecurity consists of data scattered across eight files both in pcap and csv format [33]. It contains two directories containing 8 files each; GeneratedLabelledFlows has 85 features (including label) per record in each file and MachineLearningCSV, mostly used for ML/DL tasks and focus of this study, has 79 features. These features have been extracted using CICFlowMeter which is a network flow generator and most of the features extracted are time-based statistic features [34]. Csv files are the result of flow-based features extracted from pcap files using an analyzer. Data files used in our experiments contain time-related features embedded in them are further classified as:

Iat: the inter-arrival time between packets sent in backward, forward, or either direction; Psec: includes packets or bytes per second; Active/idle: specifies time a flow was active/idle before going idle/active; other: like duration, Flag count, etc.

As evident from Table II, there are 8 files out of which one file includes only benign data while the other 7 files contain benign and attack data. File1 contains two types of brute force attacks used for logging attempts and file1 includes application layer based Dos attacks launched using different tools like GoldenEye, Hulk, slowhttptest, and slowloris. Furthermore, file3 contains web related attacks like SQL injection, brute force, and XSS while file4 incorporates infiltration attack records. Lastly file5, file6, file7 include records of the bot, PortScan, and DDoS respectively.

1) Data preprocessing: It involves techniques for data preparation or transformation of values before data is fed to the model for training.it further consists of these steps:

a) Handling of missing data: There are two approaches for handling missing data; either drop the rows containing the missing value; or fill the cell with a new value. As the dataset contains a large number of missing values, the former approach looks irrational due to which the latter approach of filling these values is chosen. There are four options to select a new value ranging from a constant value like zero to the mean, mode, or median of the selective attribute. Either one could be okay but we carried out a pre-experiment with a small portion of the dataset before major experiments to find out the best replaced value.

b) Feature scaling: On reviewing the dataset, one can find huge disparities between values from different columns with attributes like SYN, PSH flag count have a smaller range on values while attributes like duration, total length have large magnitude values. To scale these values we use standardization which works on continuous numeric features and makes sure data in a column has 0 mean and unit variance. It is done to ensure each feature has equal weightage and let gradient descent converge quickly in the model training. The formula for standardization is given in equation 2:

$$newval = (val - mean\_val) / sd \quad (2)$$

TABLE II. DATASET DESCRIPTION

S.no	Filename	Label	Records
File0	Monday-WorkingHours.pcap_ISCX.csv	Benign (Normal activities)	529918
File1	Tuesday-WorkingHours.pcap_ISCX.csv	Benign,FTP-Patator,SSH-Patator	445909
File2	Wednesday-WorkingHours.pcap_ISCX.csv	Benign, DoS GoldenEye DoSHulk, DoS slowhttptest, DoS slowloris, Heartbleed	692703
File3	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX	Benign, Web attacks (BruteForce, Sql injection, XSS	170366
File4	Thursday-WorkingHoursAfternoon-Infilteration.pcap_ISCX.csv	Benign, Infiltration	288602
File5	Friday-WorkingHoursMorning .pcap_ISCX.csv	Benign, Bot	191033
File6	Friday-WorkingHours-AfternoonPortScan.pcap_ISCX.csv	Benign, PortScan	286467
File7	Friday-WorkingHours-AfternoonDDos.pcap_ISCX.csv	Benign, DDoS	225745

Where val is actual value, mean_val and sd are mean and standard deviation of respective attribute.

c) One hot encoding: The last column/attribute representing class label in train dataset is one hot encoded to make it compatible with 1D-CNN model while training which expects target vector in said form. This results in additional columns for the output vector which is equal to the number of class labels (attacks and normal labels).

B. Model Architecture

The overall general architecture used in the experimental setup has been shown in Fig. 2. As we deal with different files the architecture of these separate models is uniform/identical albeit with minor changes. It consists of an input layer sequentially connected to 2 or 3 CNN layers intermixed by dropout and followed by flatten layer which further connects to a fully connected (FC) or dense layer and finally output layer. Input shape provided to the first Conv layer is (1* C) with 1 specifying the steps which is one row at a time and C states the number of features. With Conv layer mapping input to high dimension space, its output with dimension 1*C*f1 is the feature map containing f1 number of filters which learns network information from input data. This output is then applied to the activation function and for that purpose, the one used mostly with the Conv layer, ReLu is used. Dropout is then used to minimize the interaction of feature detectors switching off some connections randomly in the network thereby preventing model overfitting [35]. Dropout doesn't decrease the number of parameters in the model, it only prevents some of them from participating in the weight update process. The Softmax activation function is combined with an FC layer to output the classified results. The mathematical

formulae for ReLu and Softmax activation function are given in equation 3 and 4 where x and xi are the input values while f(x) and Softmax(x) being the output values passed to the next layer respectively.

$$f(x) = \max(0, x) \quad (3)$$

$$softmax(x) = \frac{e^{xi}}{\sum_j^n e^{xj}} \quad (4)$$

1) Parameters and Hyper-parameters: Another important aspect of model architecture are the parameters, which are learned through training and hyper-parameters, selected or chosen manually. In the 1D-CNN model building, the type of hyper-parameters ranges from general hyper-parameters like batch size, number of iterations to the model-specific hyper-parameters like a number of layers, filters, size of the kernel, an initial rate of learning, loss function, optimizer, and activation function used. The total parameters depend on certain hyper-parameters like number of layers, filters or nodes in a certain layer and size of filter which might vary from model to model. The general architecture of the proposed model would be like: "$Conv_1(f_1,k_1)$-$Dr_1(r_1)$-$Conv_2(f_2,k_2)$-$Dr_2(r_2)$-----$Conv_n(f_n,k_n)$-$Dr_n(r_n)$-$FC_1(nd_1)$-$FC_2(nd_2)$".

Here Conv, Dr, FC are convolutional, dropout, and fully connected layers respectively. The fi, ki refers to the number of filters and kernel-size in the ith convolutional layer whereas ri signifies the rate of dropout. The nodes in the FC layers are nd1 and nd2 with the latter related to the nodes in the output layer and equal to the number of classes. As hyper-parameters are selected manually, the number of trainable parameters can be calculated as:

a) No of parameters in first Conv layer= $C*f_1*k_1+b_1$. (b represents bias)

b) No of parameters in other Conv layer= $f_{i-1}*f_i*k_i + bi$.

c) No of parameters in Dense layers = $nd_{i-1}*nd_i + bi$.

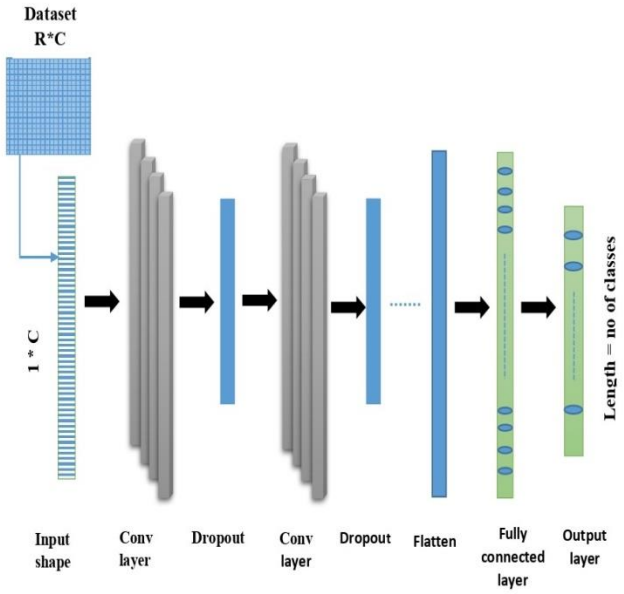

Fig. 2. Architecture of the 1D-CNN Model.

Thus, the total number of parameters in the particular model architecture is equal to the sum of parameters in all the layers. It is to be noted that the use of dropout is optional and has no effect on the number of parameters. Consider a model, for instance, with configuration "Conv(80,1)-Dr(0.2)-Conv(50,1)-Dr(0.2)-FC(50)-FC(2)". Total number of trainable parameters could be calculated as: (78*80*1+80) + (80*50+50) + (50*50+50) + (50*2+2) = 13022 trainable parameters.

C. Model Evaluation

As our work is based on classification of multiple classes, multi-class confusion matrix is used to find or display correct/ incorrect instances and its constituents are TP (True positive), TN (True negative), FP (False positive) and FN (False negative). Using these various evaluation indicators like Precision (Pr), Recall (Rc) and F1_score (F1_sc) can be derived to be further used for evaluation of model.

For classifying attack data, Pr or PPV (positive predicted value) specifies how many attack predictions actually belong to the attack data.

$$PPV = TP / (TP + FP) \quad (5)$$

Also Rc or TPR (true positive rate) specifies the ratio of predicted attack instances to the actual attack instances.

$$TPR = TP / (TP + FN) \quad (6)$$

Both PPV and TPR are suitable in their own way as former tells how attack predictions are relevant and latter tells the relevant records being predicted. Instead of choosing one over other there is another single metric F1_score calculating the harmonic mean of the both.

$$F1\_sc = (2 * PPV * TPR) / (PPV + TPR) \quad (7)$$

IV. Experimental Results and Analysis

A. Experimental Setup and Model Configuration

Experiments are conducted on google colab platform using python language and keras is the framework used for building 1D-CNN model with tensorflow as backend. Other important libraries used are pandas, numpy for loading/storing dataset and sklearn for preprocessing tasks and evaluating model and calculating results.

Different models built and evaluated might have distinct configurations of their architecture resulting in a different number of parameters and hyper-parameter values. The number of epochs and batch-size is not unique for each model but there are still some hyper-parameter values that are identical for all the models implemented in experiments and they are shown in Table III. Table IV shows the configuration parameters for each model, built during experimentation, with its complete model architecture.

B. Results

The overall experimental process is divided into two phases:

Phase1: Separate models built on individual files of dataset.

TABLE III. HYPER-PARAMETERS UNIFORM FOR ALL MODELS

Hyper-parameter	Value
Optimizer used	Adam
Kernel size	1
Learning rate	0.001 decay after certain epochs
Loss	Categorical Crossentropy
Activation function in all Conv layer	ReLu
Activation function in Dense layer	Softmax

TABLE IV. CONFIGURATION PARAMETERS FOR MODELS USED IN EXPERIMENTS

Model	Epochs	Batch size	Model Architecture	Trainable parameters
Model1	50	100	Conv(50,1)-Conv(40,1)-Conv(30,1)-FC(20)-FC(3)	7903
Model2	50	100	Conv(50,1)-Conv(40,1)-Conv(30,1)-FC(20)-FC(6)	7966
Model3	100	100	Conv(40,1)-Dr(0.1)-Conv(30,1)-Dr(0.1)-Conv(30,1)-Dr(0.2)-FC(20)-FC(4)	6024
Model4	20	100	Conv(80,1)-Dr(0.2)-Conv(50,1)-Dr(0.2)-FC(50)-FC(2)	13022
Model5	100	70	Conv(50,1)-Conv(50,1)-FC(25)-FC(2)	7827
Model6	60	40	Conv(40,1)-Dr(0.2)-Conv(30,1)-Dr(0.2)-FC(20)-FC(2)	5052
Model7	40	40	Conv(40,1)-Dr(0.1)-Conv(30,1)-Dr(0.1)-FC(20)-FC(2)	5052
Model(phase2)	50	100	Conv(60,1)-Conv(50,1)-Conv(40,1)-FC(20)-FC(8)	10818

Phase2: Model built on combined dataset except file0. Also some labels are combined and renamed to make it more balanced.

1) Phase1: During the first phase of experiments, individual files of the dataset have been used for building different models which means we have separate models for many different types of attacks. This means model1 is built on file1, model2 upon file2 and so on. This will be helpful if one wishes to detect a certain specific type of attack. For instance if you are interested in detecting DDoS attacks then model build using file7 will be useful and likewise for identifying bots model created using file5 is selected. Also processing individual files separately is good for attacks with less instances as they have better prevalence in their respective files rather than in combined dataset. It should be emphasized that file0 is not used in the experimental process as it contains only benign traffic which means seven models were trained and evaluated. Each model built is used to classify normal and corresponding attacks in the individual files and further tested on 20% test data of their respective classes.

Table V shows the detailed evaluation of each model as their overall metrics results has not been displayed but detailed result for each class in every model as huge imbalance in the dataset would always results in better overall model performance. From the detailed analysis we can observe that attacks like XSS, Sql Injection and Bot have not performed well as compared to other attacks.

2) Phase2: For the second phase of experiments, combined dataset is considered for classification and all files except file0 is taken into account. As other files too containing benign records leading to large number of normal records in combined dataset, inclusion of records of file0 could led to more imbalanced data. So dataset is combined with seven files and this combined dataset contains 2,300,825 overall records. Model built on this could classify all attacks (14) in the dataset.

TABLE V. RESULTS (PHASE1)

Model	Class Label	PPV (%)	TPR (%)	F1_sc (%)
Model1	Benign	99.97	99.99	99.98
	FTP-Patator	99.87	99.62	99.75
	SSH-Patator	99.2	98.41	98.8
Model2	Benign	99.98	99.71	99.85
	DoS GoldenEye	99.8	99.06	99.43
	DoS Hulk	99.51	99.99	99.75
	DoS Slowhttptest	96.44	99.2	97.8
	DoS slowloris	99.13	99.22	99.18
	Heartbleed	100	80	88.89
Model3	Benign	99.98	99.8	99.89
	Web Attack – Brute Force	61.39	99.32	75.88
	Web Attack – XSS	100	60	75
	Web Attack – Sql Injection	22.22	1.56	2.92
Model4	Benign	100	100	100
	Infiltration	100	77.78	88.72
Model5	Benign	99.62	100	99.8
	Bot	100	64.82	79.53
Model6	Benign	99.98	99.99	99.99
	PortScan	99.99	99.98	99.99
Model7	Benign	99.96	99.98	99.97
	DDoS	99.98	99.97	99.98

As evident from the Table VI records containing benign traffic constitute 75.76% of all the instances in the concatenated dataset while attacks barring Dos/DDoS or Portscan are low prevalent. The combined dataset suffers from a class imbalance situation with some labels like Heartbleed, XSS having very few records which often results in a low detection rate for these labels [36]. We ran an experiment to build a model that would classify all 15 classes (14 attacks) in the dataset and the results are shown in Table VII where it can be easily observed that attacks with few testing records owing to their low prevalence are not classified properly. Attacks with sufficient training instances have performed satisfactory but for minority label attacks some attacks have zero correctly classified instances while others too have low detection accuracy.

TABLE VI. SHOWS PERCENTAGE OF OCCURRENCE OF EACH LABEL IN COMBINED DATASET

Sno	New Attack label	Old Attack label	No. of Records	% of Total Records	No. of Records (new)	% of Total Records (new)
1	Benign	Benign	1743179	75.76	1743179	75.76
2	Brute Force	FTP-Patator	7938	0.345	13835	0.60
		SSH-Patator	5897	0.2562		
3	DoS	DoS GoldenEye	10293	0.447	252672	10.98
		DoS Hulk	231073	10.04		
		DoS slowloris	5476	0.239		
		DoS slowhttptest	5499	0.251		
		Heartbleed	11	0.0004		
4	Web Attacks	Web Attack– Brute Force	1507	0.0654	2180	0.0947
		Web Attack – XSS	652	0.028		
		Web Attack– Sql Injection	21	0.00091		
5	Bot	Bot	1966	0.085	2002	0.087
		Infiltration	36	0.0015		
6	PortScan	PortScan	158930	6.9075	158930	6.9075
7	DDoS	DDoS	128027	5.5643	128027	5.5643

To solve the class imbalance situation relabeling is done by merging minority class labels into one class label which proves to be a good measure for improving model performance. It is not done randomly but in a strategic way by merging similar categories of attacks. For example, SQL injection, XSS, and web attack-brute force are all types of web attacks so they are merged together and given new labels (web attacks). Full details of the new attack label along with the percentage of occurrence are shown in the Table VI. After relabeling it now contains 7 classes including 6 attack labels and a model based on 1D-CNN is trained and then evaluated. The results for the same are shown in Table VIII and Table IX with the former displaying the confusion matrix based on all the labels and the latter illustrating the detailed results in metrics for all class labels. Analyzing the confusion matrix in Table VIII, the number of classifications or misclassifications with a particular class label predicted as another label can be properly seen. The same can be analyzed from Table IX as a high number of true positives were achieved for all class labels with the exception of the Bot and Web attacks label. The overall performance of the model is better as more than 99.6% output has been achieved in PPV, TPR, and F1_sc. Bot and Web attacks are the two labels with gloomy detection rate resulting in low values of TPR and F1_sc.

3) Experiment with deep neural network: To compare and further validate our proposed model, a DNN based on an artificial neural network has also being used. The experimental setup is identical with 1D-CNN i.e., the same preprocessing steps and evaluation metrics. DNN comprises of a) input layer with 78 nodes; b) 3 hidden layers with 60, 50, and 20 nodes, respectively; c) output layer with 8 nodes(like phase2). Also, dropout with 0.1 value is used between hidden layers to prevent overfitting. The results are depicted in Tables X and XI.

Table X displays the confusion matrix evaluated from the DNN-model and Table XI shows the comparative analysis of 1D-CNN with DNN. It can be analyzed from the latter table that 1D-CNN model has outperformed the model built using DNN in detection of network attacks.

TABLE VII. INITIAL RESULTS-15 CLASS CLASSIFICATION (PHASE2)

Label	PPV (%)	TPR (%)	F1_sc (%)
BENIGN	99.71	99.78	99.75
Bot	90.61	41.41	57.33
DDoS	99.98	99.92	99.96
DoS GoldenEye	99.57	98.47	99.12
DoS Hulk	99.09	99.20	99.15
DoS Slowhttptest	91.28	98.69	95.11
DoS slowloris	98.71	98.88	98.85
FTP-Patator	99.60	99.27	99.47
Heartbleed	100.00	66.67	80.00
Infiltration	50.00	100.00	67.00
PortScan	99.36	99.95	99.65
SSH-Patator	95.70	99.14	97.66
Web Attack Brute Force	100.0	11.89	21.58
Web Attack Sql Injection	nan	0.00	0.00
Web Attack XSS	0.00	0.00	0.00

TABLE VIII. CONFUSION MATRIX USING 1D-CNN FOR 7 CLASS CLASSIFICATION

True / Predicted	BENIGN	Bot	Brute Force	DDoS	DoS	PortScan	Web Attacks	All
Benign	**347743**	3	30	11	791	205	0	348783
Bot	253	**145**	0	0	0	0	0	398
Brute Force	1723	0	**2665**	0	0	2	0	2683
DDoS	99	0	0	**25534**	1	0	0	25558
DoS	136	0	3	0	**50043**	0	0	50509
PortScan	5	0	0	0	11	**31807**	0	31823
Web Attacks	353	0	24	0	0	0	**34**	411
All	350112	148	2712	25545	50846	32014	34	**460165**

TABLE IX. DETAILED RESULTS- 7 CLASS CLASSIFICATION

Label	TP	FN	FP	PPV (%)	TPR (%)	F1_sc (%)
BENIGN	347743	1040	787	99.77	99.70	99.74
Bot	145	253	3	97.97	36.43	53.11
Brute Force	2665	18	56	97.94	99.33	98.63
DDoS	25534	24	11	99.96	99.91	99.93
DoS	50371	138	804	98.43	99.73	99.07
PortScan	31807	16	205	99.36	99.95	99.65
Web Attacks	34	377	0	100.00	8.27	15.28

TABLE X. CONFUSION MATRIX USING DNN FOR 7 CLASS CLASSIFICATION

True / Predicted	BENIGN	Bot	Brute Force	DDoS	DoS	PortScan	Web Attacks	All
Benign	**345697**	6	69	9	2767	234	1	348783
Bot	257	**141**	0	0	0	0	0	398
Brute Force	31	0	**2650**	0	2	0	0	2683
DDoS	39	0	0	**25517**	2	0	0	25558
DoS	222	0	5	2	**50280**	0	0	50509
PortScan	51	0	0	1	11	**31760**	0	31823
Web Attacks	375	0	23	0	3	0	**10**	411
All	346672	147	2742	25529	53065	31994	11	**460165**

TABLE XI. COMPARISON OF 1D-CNN WITH DNN

	PPV (%)		TPR (%)		F1_sc (%)	
Label	**CNN**	**DNN**	**CNN**	**DNN**	**CNN**	**DNN**
BENIGN	99.77	99.72	99.70	99.12	99.74	99.42
Bot	97.97	95.92	36.43	35.43	53.11	51.74
Brute Force	97.94	96.47	99.33	98.77	98.63	97.61
DDoS	99.96	99.95	99.91	99.84	99.93	99.90
DoS	98.43	94.75	99.73	99.55	99.07	97.09
PortScan	99.36	99.27	99.95	99.80	99.65	99.53
Web Attacks	100.00	90.91	8.27	2.43	15.28	4.74

Analysis: While analyzing the results, it can be observed.

- In both phases, attacks with reasonable instances for training have produced exceptionally better results on testing data. Attacks like DoS, Brute force, DDoS, and Portscan have specific attack pattern and they are better detected using flow based features.
- Overall Bot and Web attacks have shown poor performance in both phases.
- Analyzing the results of Bot in phase2 (Table VIII), one can see similarities between Bot and Benign traffic as all FN and FP in case of Bot attack label belongs to benign label which indicates Bot is not classified as any other attack by the model and no other attack has been classified as Bot attack. This signifies the resemblance between the two as the distinction between bot and normal behavior is blurred.
- As for web attacks, their comparatively lower performance could be attributed to the fewer training instances in the dataset as they have less than 0.1 of total instances. Or these attacks don't have a specific pattern and they could be better detected using payload content.
- Also our proposed 1D-CNN model has outperformed the model built using DNN (Table XI).

V. Conclusion

In this paper, we proposed a novel way of identifying attacks in the dataset using 1D-CNN as a classification approach. The proposed 1D-CNN model has performed better with the least number of misclassifications. Experiments were conducted with a model trained and evaluated on individual files of the dataset as well on a combined dataset which was further relabeled to handle class imbalance situation. Satisfactory performance was recorded in both cases for the majority of labels as more than 99% output achieved in each of the evaluation indicators used. Some attacks with low prevalence like bot and web attacks have a comparatively lower detection rate. Experiments using DNN have also been done for comparative purposes and further validation of the proposed model.

As for future work, other DL algorithms need to be explored for training the model and a study regarding hyper-parameter optimization should be done to find the optimal model configuration. Moreover, other datasets with the latest attack types and real world traffic should be investigated for detection of cyber-attacks. Addition of records of bots and web related attacks needs to be done as more data is needed for training and to improve their detection accuracy.

References

[1] M. Uma, and G. Padmavathi, "A Survey on Various Cyber Attacks and their Classification," Int. J. Netw. Secur. 15(5), pp. 390-396, 2013.

[2] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," ACM Computing Surveys (CSUR), 47(4), pp. 1-33, 2015.

[3] A. L. Buczak, and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications surveys & tutorials, 18(2), pp. 1153-1176, 2015.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11), 1998, pp.2278-2324.

[5] A, Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks" In Advances in neural information processing systems, pp. 1097-1105, 2012.

[6] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv preprint arXiv:1409.1556.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015 pp. 1-9.

[8] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inma, "1D convolutional neural networks and applications: A survey," 2019, arXiv preprint arXiv:1905.03554.

[9] A. Jacovi, O.S. Shalom, and Y. Goldberg, Understanding convolutional neural networks for text classification," 2018, arXiv preprint arXiv:1809.08037.

[10] H. Cho, and S.M. Yoon, "Divide and conquer-based 1D CNN human activity recognition using test data sharpening," Sensors, 18(4), p.1055, 2018.

[11] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," IEEE Transactions on Industrial Electronics, 63(11), pp.7067-7075, 2016.

[12] S. Duque, and M.N. Omar, 2015, "Using data mining algorithms for developing a model for intrusion detection system (IDS)," Procedia Computer Science, 61, pp.46-51, 2015.

[13] S. Aljawarneh, M. Aldwairi, and M.B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," Journal of Computational Science, 25, pp.152-160, 2018.

[14] A. Saied, R.E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," Neurocomputing, 172, pp.385-393, 2016.

[15] R. Singh, H. Kumar, and R.K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Systems with Applications, 42(22), pp.8609-8624, 2015.

[16] R. Priyadarshini, and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," Journal of King Saud University-Computer and Information Sciences, 2019 https://doi.org/10.1016/j.jksuci.2019.04.010.

[17] T. Aldwairi, D. Perera, and M.A. Novotny, "An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection," Computer Networks, 144, pp.111-119, 2018.

[18] T. Wisanwanichthan and M. Thammawichai, "A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM," IEEE Access, 9, pp.138432-138450, 2021.

[19] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, " Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset," IEEE Access, 9, pp.140136-140146, 2021.

[20] Y. Liu, S. Liu, and X. Zhao, "Intrusion detection algorithm based on convolutional neural network," 2017 DEStech Transactions on Engineering and Technology Research, (iceta).

[21] W.H. Lin, H.C. Lin, P. Wang, B.H. Wu, and J.Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," In 2018 IEEE International Conference on Applied System Invention (ICASI), IEEE, pp. 1107-1110, April 2018.

[22] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," IET Information Security, 13(1), pp.48-53, 2018.

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

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

[25] S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," IEEE Access, 6, pp.48231-48246, 2018.

[26] A.K. Verma, P. Kaushik, and G. Shrivastava, "A Network Intrusion Detection Approach Using Variant of Convolution Neural Network," In 2019 International Conference on Communication and Electronics Systems (ICCES), IEEE, July 2019, pp. 409-416.

[27] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," IEEE Access, 8, pp.29575-29585, 2020.

[28] V. Gustavsson, "Machine Learning for a Network-based Intrusion Detection System: An application using Zeek and the CICIDS2017 dataset," 2019.

[29] B. Rababah, and S. Srivastava, "Hybrid Model For Intrusion Detection Systems,"2019, arXiv preprint arXiv:200308585.

[30] D. Aksu, S. Üstebay, M.A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," In International Symposium on Computer and Information Sciences, Springer, Cham, September 2018, pp. 141-149.

[31] Z.K. Maseer, R. Yusof, N. Bahaman, S.A. Mostafa, and C.F.M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset" IEEE Access, 9, pp.22351-22370, 2021.

[32] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security (TISSEC), 3(4), pp.262-294, 2000.

[33] I. Sharafaldin, A.H. Lashkari, and A.A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," In ICISSP, January 2018, pp. 108-116.

[34] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, and A.A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," In Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), February 2016, pp. 407-414.

[35] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, arXiv preprint arXiv:1207.0580.

[36] S. Wang, L.L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," IEEE transactions on neural networks and learning systems, 29(10), pp. 4802-4821, 2018.