DL-Based DDoS Detection with High Accuracy and Low False Positive Rate

Kartik Balu[1] **& Anoop R S**[2]

Received January 26, 2025
Accepted April 30, 2025
Electronic access May 15, 2025

Computer Networks are responsible for ensuring the availability of internet services, in addition to providing confidentiality and integrity of the data being transmitted. These services should be available on demand in critical applications such as healthcare. Unfortunately, attacks may hinder networks from offering essential services, which may affect human lives in medical scenarios for example. Distributed Denial of Service (DDoS), is one of the most dangerous cyber security attacks affecting the availability of networks. With increasing attacks, it has become very important to find ways to detect them to ensure uninterrupted operation of networks. In order to combat these attacks, many statistical, machine learning (ML), and deep learning (DL) models have been proposed and implemented. Most of these methods have high false positive rates (FPR), as a result of which a lot of normal traffic is detected as malicious, leading to the loss of important data in networks. To overcome this issue, a methodology using Bi-directional LSTM (Bi-LSTM) is proposed to reduce false positive rates while maintaining high accuracy, which leads to improved network performance. A thorough evaluation has been conducted by comparing our methodology against other learning models like LSTM (Long Short-Term Memory), accuracy (99.84%), FPR (0.071%), GRU (Gated Recurrent Unit) accuracy (99.85%), FPR (0.069%), and SVM (Support Vector Machine) accuracy (99.01%), FPR (0.85%) and found that our methodology yields high accuracy (99.850%) and low FPR (0.053%) on the standardized dataset CICIDS2017 from the Canadian Institute of Cybersecurity.

Introduction

In today's "Internet of Things" world, all devices from kitchen appliances to the most sophisticated servers are on the internet. With so many devices constantly communicating with each other along with ubiquitous technologies like cloud based computing and big data, the network is now vulnerable to attacks by hackers and cybercriminals. One example is the Distributed Denial of Service (DDoS) attack. DDoS is an attack that comes from various sources generating very high network traffic with the intention of flooding and eventually bringing down the network. According to Gcore reports*, DDoS attacks in early 2024 have increased by 46% compared to the attacks in a similar period in 2023. DDoS attacks have reached 445,000 in the second quarter of 2024 alone, indicating a rapid increase in these attacks. Hence, there is a need to deploy effective detection methods that have high accuracy and low false positive rates. False positive cases happen when a legitimate network traffic is incorrectly reported as malicious, which prevents it from reaching its destination. Several papers have studied the accuracy of detection methods; however, fewer have studied the issue of reducing false positives while maintaining high accuracy, which makes it a point of interest (Table 1).

There are 2 types of detection mechanisms: signature based and anomaly-based detection. In signature-based detection, the network packet is checked against a database of attacks to determine if it is malicious or not. The drawback of this method is that it is difficult to detect evolving attacks. In an anomaly-based method, the model learns normal network traffic patterns and is able to detect attack packets based on deviation from normality. Machine learning and deep learning models, which often employ anomaly-based detection, can be used to detect DDoS attacks[1,2]. However, these models need to be trained and tested on a dataset that contains normal and attack data for these models to recognize new attack pattern changes. In this paper, the standardized dataset CICIDS2017 is used since it contains a balance of both normal and attack traffic, making it ideal for anomaly-based detection.

Distributed Denial of Service (DDoS) Attack

Network security remains the primary concern for today's networks[3]. DDoS attacks pose a significant impact to network security. Attackers utilize bots or zombies to flood the network and exhaust the host's resources rendering it incapable of handling any new requests. These attacks are easy and low

[1] *Cupertino High School, Cupertino, CA, USA*
[2] *Computer Science and Engineering, Carmel College of Engineering and Technology, Kerala, India*
* https://gcore.com/blog/radar-q1-q2-2024-insights/

cost making them an increasingly critical threat. Attackers may infect several computers with viruses. These distributed computers then serve as bots or zombies to continuously consume the resources of the network bringing services down and leaving very little or no room for legitimate traffic. There are 3 main types of attacks: volumetric attacks like ICMP (Internet control message protocol) attacks that consume the bandwidth of the network, protocol attacks like SYN (synchronize) flood that attack the weakness of the network protocols bringing servers and other resources down, and application layer attacks like "http flood" or "slowHttpTest attacks" that bring down applications.

Dataset

In order for (Machine Learning) ML and (Deep Learning) DL models to be effective, it is necessary to train and test them against standardized data containing malicious and benign traffic. Once the model has been properly evaluated under a standardized dataset, it can be tested against real time data. This paper uses the CICIDS2017 dataset from the Canadian Institute of Cybersecurity. It is a labeled dataset containing eight different files with five days (Monday to Friday) of normal and malicious flow of traffic. In this dataset, there are fifteen distinct classes, including Benign data, Bot attacks, DDoS attacks, DoS GoldenEye attacks, Dos Hulk attacks, Heartbleed attacks, Slowhttptest attacks, slowloris attacks, Infiltration attacks, SSH Patator attacks, Brute Force attacks, FTP Patator attacks, SQL injection attacks, PortScan attacks, and XSS attacks (Table 2). Because of this diversity, researchers can test and evaluate Intrusion Detection Systems against various different threats. For our model, only the Friday dataset was used since it contains DDoS attacks, which is the focus of this paper. It is confirmed that the Friday dataset was well balanced since 60% of its rows contain DDoS attack data and 40% of its rows contain benign data.

Related Works

In recent times much work has been done on DDoS attacks on SDN (Software Defined Network). Kumar et al. use a deep learning method called LSTM (Long Short-Term Memory) to detect DDoS attacks[4]. The accuracy of their model was 98% as compared to 89% for k-nearest neighbors (KNN) method and 93% for Artificial Neural Networks (ANN) method. This model proved more accurate than the ML models. However, this model can only detect learned attacks. Time domain correlation is a key factor in choosing LSTM for this paper. Keros framework was used to create the model and manage the data.Python 3.9, Tensorflow, and Sklearn were used for testing the model. For future works, they proposed capturing network traffic which will incorporate incremental learning. Using this method, the limitation of detecting only learned attacks can be eliminated and the model can now detect new types of attacks.

Jogin et al compared Convolutional Neural Networks (CNN) to other classifier algorithms like KNN (28.2% accuracy), Support Vector Machine SVM (37.4% accuracy), softmax classifier (34.1% accuracy), and fully connected Neural networks (46.4% accuracy)[5]. They reported the CNN algorithm had 85.97% accuracy. They attributed the better performance using CNN to its ability to alter the shape of the output unlike the other models where it is fixed. An ensemble CNN is implemented for more efficient attack detection in reference[6]. In that work, a set of 10 features were extracted using the CICFlowmeter and a random forest regressor was used for selecting the best features. They achieved 99.45% accuracy, better than SVM (95.24%) and RNN (83.28%). While CNN and ensemble CNN have high accuracy, they need large amounts of data to train them effectively so training time is a drawback for this method. The downside of this model is that every single packet is analyzed for computation and detection instead of just sampling some flows. Since all the incoming traffic is analyzed, the False Positive Rate (FPR) will be reduced, but this method is computationally complex and highly resource intensive. Similarly, another intrusion detection system called DDOSNet was proposed[7], which combines RNN with encoders using dataset CICIDS2019 resulting in an accuracy of 99% as compared to the SVM (95%), Random Forest (86%), and Decision Tree (77%). However, in this model, false alarm rates are extremely high when the traffic becomes very large, making it unfit for such situations. A combination of statistical and machine learning models[8] yielded an accuracy of 98.71% using SVM. However, accuracy of SVM models is very sensitive to noise. To address this, Sahoo et al proposed using SVM with kernel principal component analysis (KPCA) with a genetic algorithm1. In order to decrease noise caused by feature differences an improved kernel function (N-RBF) was used. Other investigations like those done by Ramzan et al[9] propose a model where DL methods like LSTM and Gated Recurrent Units (GRU) are used. They report faster attack detection time for GRU with high accuracy (99%) using the dataset CICIDS2019. Sequential steps used to implement the model are data pre-processing, data normalization using standard scalar, one-hot encoding and finally feature selection. Overfitting is prevented using early stopping along with dropout layers. Furthermore, when loss and accuracy graphs are examined, then RNN does not exhibit signs of overfitting while LSTM and GRU do which indicates that RNN will perform better to unseen data and thus is more effective in binary classification. In addition, GRU is not efficient in catching long term dependencies so for larger amounts of data, GRU may not detect the attacks as accurately as RNN.

Wang et al[10] implemented a safeguard scheme on the control plane which consists of two modules: anomaly traffic detection and controller dynamic defense. These modules serve as identification and mitigation mechanisms. The anomaly traffic detection module in the data plane uses a flow monitoring

approach to distinguish normal flows and attack flows. The controller dynamic defense module remaps the load from the master controller to slave controllers. The implementation will be complicated in this method since the data plane is resource restricted.

Chartuni et al [11], used a multi-classification method with high accuracy (94.57%). Multi-classification is a good method since it is able to detect not only attack traffic, but also is able to classify it to the specific attack. However, multi-classifiers rely on data of already existing attacks, so they won't be able to classify a new type of attack. To detect DDoS traffic, a technique is proposed that integrates sflow-RT application and snort rules [12]. In this implementation, multiple RYU controllers are utilized. Detection and mitigation rules are shared among these controllers using Redis simple message queue(RSMQ). A multi-controller deployment has advantages of detecting and mitigating attacks much quicker than a single controller. More controllers can also lead to complexity in the inter communication between controllers and also increase cost of deployment.

After comparing 34 research papers Mittal et al concluded that CNN, DNN and CNN-LSTM show greater than 99% accuracy [13]. They highlighted the importance of advanced pre-processing and feature selection techniques used in parallel with DL methods to achieve high accuracy in DDoS attack detection, despite the reduction in the number of features. Thus, pre-processing plays a key role in detecting higher amounts of attack data even if not previously trained. Zhao et al [14] showed that a CNN-BiLSTM method using 4 memory modules with 2 cells in each module gave optimum results and high accuracy (92.50%).

After reviewing related works in the field, the conclusions are as follows. DL methods are likely superior to ML methods because of their ability to handle large scale dynamic data patterns and implement automatic feature engineering. Within DL methods, Bi-LSTM may be best suited for detecting DDoS attacks with high accuracy and low false positive rates. The uniqueness of Bi-directional LSTM lies in its ability to process the data sequence in both the forward as well as the backward direction. DDOS attacks don't happen at a fixed frequency; they occur at irregular times with spikes that could be missed if data is considered from a single direction. But if data is analyzed in both directions (future data and past data), then patterns are easier to identify. DDoS attacks can be identified not only by understanding how the data has changed but also by understanding how future data relates to the past and this can only be done by having a backward direction of analyzing data which Bi-LSTM incorporates. Also, because Bi-LSTM also studies the dependencies of current inputs with future states, it has the ability to correctly predict the attack before it occurs. If a DDoS attack is being launched to slowly increase in intensity, when the attack pattern starts, Bi-LSTM can recognize the patterns immediately like packet frequency, size, input frequency etc. and block the attack in its early stages. Also, Bi-LSTM has the ability to learn normal traffic patterns like spikes in data because of, for example, promotional sales or a special event and recognizing patterns in that traffic to not accidentally characterize them as attack traffic. Another very important feature is the time dependency feature where sequence dependent features are captured and the evolution of data over time is studied and understood. This can easily be missed by models like SVM since they only look in one direction. As a result, if a corrupt bot keeps sending signals in an irregular interval, it can easily be detected by Bi-LSTM because of its understanding of data dependencies in a sequential context. Thus, given the characteristics of DDoS attacks which are a form of time series, involving high frequency, non-linear data with very complex structures and relationships and large range dependencies, Bi-LSTM can catch these intrinsic details and effectively predict the attacks. In addition, Bi-LSTM also has the ability to use its gated cell state, which makes it act like a computer's memory to make decisions on what data is allowed to be written to it, read from it, and stored on it. This allows Bi-LSTM to keep track of features of attacks learned from the training process and make detection decisions based on the stored information on gated cells, increasing its accuracy of detection.

Our Proposal

In our work, data pre-processing and use of the Bi-directional LSTM model is proposed in order to deliver high accuracy and low false positives since Bi-LSTM works very well in processing temporal dependencies and accurately recognizing attack patterns and also recovering any loss of data. The proposed methodology is then compared with other deep learning models like GRU, SVM and LSTM to test our hypothesis that Bi-LSTM performs better.

Results

The labeled dataset CICIDS2017 was used for this model since it was well-balanced in terms of rows containing DDoS attack data and rows containing benign data. 60% of the rows contained DDoS data and 40% of the rows contained benign data. Since the dataset is well balanced, there was no need to carry out class imbalance adjustments. Through random sampling, it can be assumed that there will be an even split of DDoS and benign data in both the training and testing data.

The overall flow used for data analysis is shown in Figure 1. To start the pre-processing stage, rows containing NaN and infinite values were removed from the dataset. Then, by classifying pieces of data that were more than three standard deviations away from the mean, new columns were created to distinguish between populations and outliers. For example, for the feature "Fwd Packet Length Max" (Figure 2), a column "FwdPacketLengthMaxPopulation" was created for outlier classification,

Citation	Year	Network Architecture	Method	False Positive Rates	Accuracy
Becerra et al.[15]	2024	Analysis paper	RF, DT, ADA, XGB, MLP, DNN	-	RF 99.97%
Kumar et al.[4]	2023	-	LSTM	-	98%
Zhao et al.[14]	2023	-	CNN-BiLSTM	-	95.67%
Ramzan et al.[9]	2023	SDN	RNN, LSTM, GRU	-	96% (RNN), 97% (LSTM), 98% (GRU)
Chartuni et al.[11]	2021	-	Custom model	-	94%
Haider et al.[6]	2020	SDN	Ensemble CNN	-	99.45%
Elsayed et al.[7]	2020	SDN(multi-controller)	RNN with encoder	1%	99%
Tayfour et al.[12]	2020	SDN	Sflow-RT	-	-
Singh et al.[8]	2020	SDN	Statistical and ML(SVM)	-	98-99%
Wang et al.[10]	2019	SDN	SGS	-	-
Jogin et al.[5]	2018	SDN	CNN	-	85.97%
This Paper	2025	-	Bi-LSTM	0.053%	99.850%

Table 1 Displays the accuracy and false positives along with methodology used

SI no.	Class Label	Number of Rows
1	BENIGN	2,359,087
2	Bot	1,966
3	DDoS	41,835
4	DoS GoldenEye	10,293
5	DoS Hulk	231,072
6	Heartbleed	11
7	Slowhttptest	5,499
8	Slowloris	5,796
9	Infiltration	36
10	SSH Patator	5,897
11	Web Attack - Brute Force	1,507
12	FTP Patator	7,938
13	Web Attack - SQL Injection	21
14	PortScan	158,930
15	Web Attack - XSS	652

Table 2 Displays the fifteen distinct classes in the CICIDS2017 dataset

which indicated that everything within 3 standard deviations of the mean was part of the main population and everything outside of 3 standard deviations of the mean was an outlier for that feature. Some features had bimodal or trimodal populations, which required a more in-depth classification. For example, the feature “Bwd Packet Length Max” had 3 different populations

(Figure 3), which prompted the creation of the columns "Bwd Packet Length Max Less Than 2250", "Bwd Packet Length Max Between 2250 And 5750", and "Bwd Packet Length Max More Than 5750" to distinguish between these groups.

After that, the data were normalized using Standard Scaler. The scaling process was divided into 2 parts. The first part was creating new columns of scaled data for entire populations. For example, "Total Fwd Packets Scaled" was a column that contained the scaled version of the "Total Fwd Packets" feature. After doing this for all features, the next step consisted of creating new columns of scaled data just for the main populations, meaning that outliers were excluded in this scaling. To do this, a temporary variable was created that only contained the main population for a certain feature based on the outlier thresholds determined earlier. After that, the temporary variable was scaled, creating a new column that contained the scaled version of the main population.

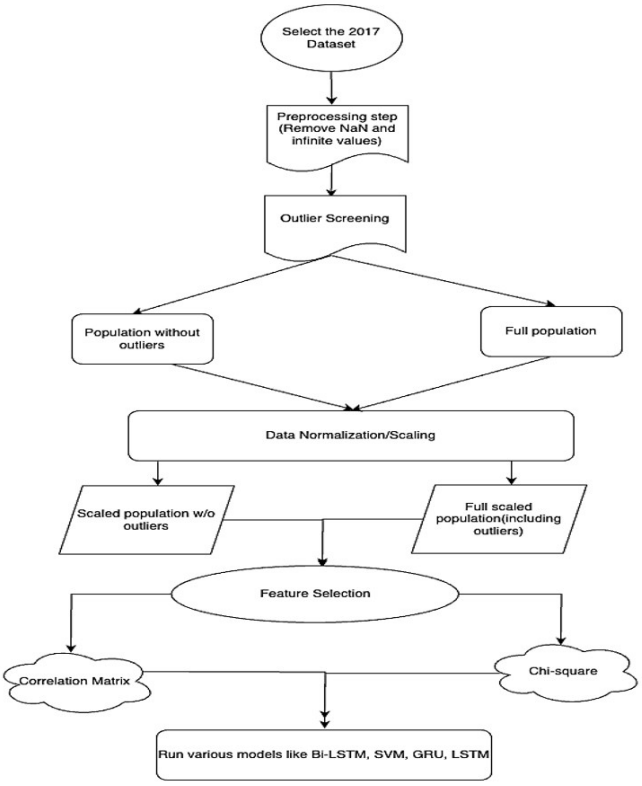

Fig. 1 Flow chart of sequence used for data analysis

Feature selection was then carried out through 2 different methods. The first method of feature selection through a correlation matrix and the second method was through the chi-squared formula. The use of a correlation matrix for feature selection is an established approach that has been commonly utilized. For example [16], makes use of feature selection with a correlation

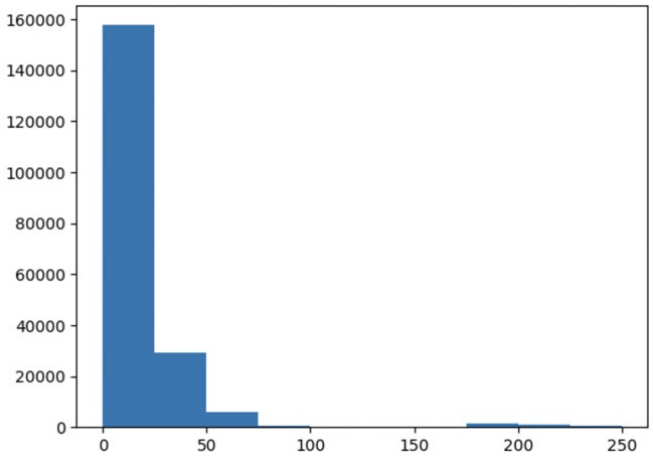

Fig. 2 Histogram for feature Fwd Packet Length Max in CICIDS 2017 Dataset

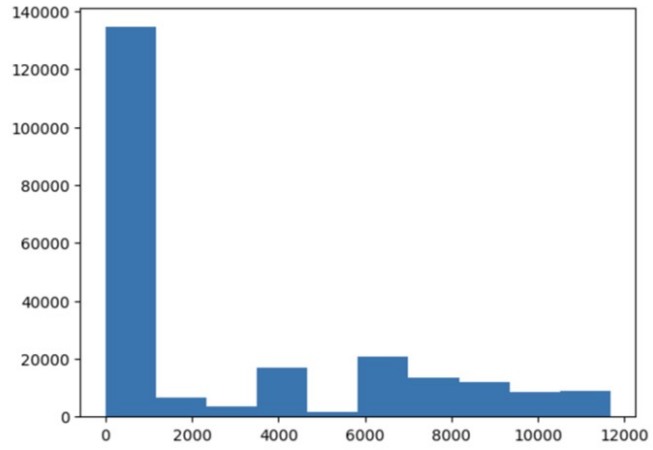

Fig. 3 Histogram for feature Bwd packet length max in CICIDS 2017 Dataset showing trimodal distribution

matrix on an experimental dataset, demonstrating the means by which it can be used for feature selection. To use a correlation matrix for feature selection, features that were correlated with each other by a threshold of over 0.95 were put into the same groups. The threshold of 0.95 was used for feature selection because through sensitivity analysis, it was determined that using a correlation threshold of 0.95 produced the best results for most of the performance metrics, especially for accuracy and FPR. For example, when considering Bi-LSTM, it can be seen that it had an FPR of 0.00142 and an accuracy of 99.775% when the threshold was 0.75, an FPR of 0.00134 and an accuracy of 99.737% when the threshold was 0.85, and an FPR of 0.00053 and an accuracy of 99.850% when the threshold was 0.95, which shows that using the threshold of 0.95 gave the lowest FPR and

highest accuracy (Table 3, 4, 5). This trend can generally be seen with the other models and performance metrics as well, making the 0.95 threshold the most optimal, probably because features with more distinct information were selected at this threshold value.

One feature from each of the correlated groups was selected during feature selection since highly correlated features in the same group would essentially give the same results. For example, from the groups created using a threshold of 0.85, it was found that the features "Flow Duration", "Flow IAT Std", "Flow IAT Max", "Fwd IAT Total", and "Fwd IAT Max" were all very highly correlated, so they were considered as a group (Table 6). Only Flow IAT Max was selected from this group for feature selection since picking each feature in this group essentially provides the same information, so picking more than one feature from here wouldn't make sense. In addition, Flow IAT Max's correlation with Flow Duration was closest to the mean of all of the other correlation values with Flow Duration, so by selecting this feature, the already small loss of data caused by not selecting all of the features in the group is minimized.

This process was done three times for the unscaled version of features, the scaled version of entire populations, and the scaled version of only the main populations since it was hard to tell which set of features would perform the best without testing it on a model. Through this method, three sets of different features were selected, each one containing about 10-12 features. In addition to this method, the chi-squared function (Equation 1) was used for feature selection, which produced another set of 12 features. Chi-square is a statistical procedure that helps figure the difference between observed data Oi and expected data Ei mainly to rule out that the observations are not due to chance and that in fact there is a relation between the observed and the expected data. Using chi-squared to create a fourth set allowed us to verify that our feature selection using a correlation matrix was valid, and it gave us an alternative set of features that could potentially perform better than the features selected through the use of a correlation matrix.

For the training and testing stage, the program was run through 4 different models: Bi-LSTM, GRU, SVM, and LSTM. For deep learning neural networks (Bi-LSTM, GRU, LSTM), categorical cross entropy was employed for our loss function, adam as our optimizer, and accuracy as our performance metric. For SVM, a machine learning model, auto was used for the gamma hyperparameter. The models were run on the four different sets of features mentioned previously. In order to make sure all data was used in training and testing at least once, k-fold validation with four folds was used for all models. It was determined that the feature set containing scaled versions of the entire data ended up giving the best results through testing on our models. Using this feature set, Bi-LSTM, GRU, SVM, and LSTM models were then compared.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Where:
χ^2 = Chi-squared statistic
O_i = Observed value
E_i = Expected value

Comparing Bi-LSTM to other models like SVM, GRU and LSTM for the threshold 0.95, it was found that Bi-LSTM had an accuracy of 99.850% while SVM had an accuracy of 99.041%, LSTM had an accuracy of 99.841%, and GRU had an accuracy of 99.845%. In addition, it can be noted that the confidence intervals for each of the models is extremely small, which means that the predictions for the accuracy are extremely precise and would be consistent for several runs. False positive rates were also determined for each of the above methods. Bi-LSTM had a false positive rate of 0.00053 while SVM had a false positive of 0.00848, GRU had a false positive rate of 0.00069, and LSTM had a false positive rate of 0.00071 (Table 5). After running and comparing the models, it is clear that Bi-LSTM is superior since it not only has the highest accuracy, but also results in the lowest false positive. This makes it the best model to choose for DDoS attacks where not only attack traffic is caught accurately, but also normal traffic reaches its destination on time, making it very efficient for the network.

When comparing the proposed Bi-LSTM with a correlation threshold of 0.95 to existing approaches that target low false positives for DDoS attacks, it can be noted that the proposed model generally performs better in terms of accuracy, precision, recall, and FPR. The only exceptions to this are RF and a Conditional Entropy-based Algorithm, which have FPRs of 0%, but this comes at the cost of other metrics including accuracy, which are better for the proposed Bi-LSTM model. (Table 7).

Discussion

Overall, this study aimed to minimize false positive rates and achieve a high accuracy when detecting DDoS attacks. After going through preprocessing, outlier identification, normalization, and feature selection, Bi-LSTM, SVM, GRU, and LSTM were run on our dataset and it was found that Bi-LSTM achieved the highest accuracy and lowest false positive rate.

From our experiment, it is important to recognize the significance of outlier classification. Creating columns for each feature to determine whether a piece of data was considered an outlier or part of the main population and then feeding that information to our model was a key part in achieving a high accuracy and low false positive rate because outliers typically differ from the main population. For example, for the feature Active Max, the main population was well balanced with around 57% of the data being DDoS attacks and 43% being benign. However, for

Model Used	Conf. Interval	FPR	False Negative Rate (FNR)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
Proposed Bi-LSTM	(0.00129, 0.00320)	0.00142	0.00082	99.750	99.855	99.802	99.775
LSTM	(0.00062, 0.00555)	0.00071	0.00238	99.875	99.581	99.727	99.691
GRU	(0.00146, 0.00322)	0.00155	0.00078	99.726	99.862	99.794	99.766
SVM	(0.01155, 0.01164)	0.00627	0.00532	98.896	99.062	98.979	98.841

Table 3 Comparison of models tested for correlation threshold of 0.75

Model Used	Conf. Interval	FPR	False Negative Rate (FNR)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
Proposed Model Bi-LSTM	(0.00170, 0.00356)	0.00134	0.00129	99.763	99.773	99.768	99.737
LSTM	(0.00242, 0.00514)	0.00276	0.00102	99.514	99.820	99.667	99.622
GRU	(0.00165, 0.00577)	0.00271	0.00101	99.524	99.823	99.673	99.629
SVM	(0.02486, 0.02604)	0.00774	0.01771	98.610	96.878	97.736	97.455

Table 4 Comparison of models tested for correlation threshold of 0.85

Model Used	Conf. Interval	FPR	False Negative Rate (FNR)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
Proposed Model Bi-LSTM	(0.00139, 0.00162)	0.00053	0.00097	99.907	99.828	99.868	99.850
LSTM	(0.00140, 0.00179)	0.00071	0.00088	99.874	99.845	99.859	99.841
GRU	(0.00142, 0.00169)	0.00069	0.00087	99.879	99.847	99.863	99.845
SVM	(0.00904, 0.01014)	0.00848	0.00111	98.523	99.805	99.160	99.041

Table 5 Comparison of models tested for correlation threshold of 0.95

the outliers, 100% of the data was benign, demonstrating the significance of outliers since they not only can be removed for scaling, but also tend to fall under the same category of either DDoS or benign.

In addition, the difference in performance between chi-squared feature selection and manual feature selection with a correlation matrix should be noted. The features selected by chi-squared were "Fwd Packet Length Max", "Bwd Packet Length Max", "Bwd Packet Length Mean", "Bwd Packet Length Std", "Bwd IAT Total, Max Packet Length", "Packet Length Mean", "Packet Length Std", "URG Flag Count", "Average Packet Size", "Avg Fwd Segment Size", and "Avg Bwd Segment Size". On the other hand, the features selected after making groups based on the correlation threshold of 0.95 were "Total Forward Packets", "SYNFlagCount", "Bwd Packet Length Max", "Max Packet Length", "Flow IAT Max", "Flow Duration", "Packet Length Mean", "Total Length of Bwd Packets", "Fwd Packet Length Max", "Bwd IAT Max", "Bwd Header Length", and "Active

Feature Name	Correlation Value
Flow Duration	-
Flow IAT Std	0.899179
Flow IAT Max	0.920257
Fwd IAT Total	0.997054
Fwd IAT Max	0.917945

Table 6 Example of a group of correlated features

Min". These features were selected because their correlation values were closest to the mean of all of the other correlation values in the group, as explained earlier in the example with "Flow IAT Max." A key reason why chi-squared produced worse results than feature selection through a correlation matrix is that chi-squared didn't account for how strongly correlated two features were. For example, chi-squared selected both "Bwd Packet Length Max" and "Bwd Packet Length Mean" in its feature set, which according to the correlation matrix, have a high correlation of 0.961442 with each other. This can hinder the amount of information the model gets because if two features have an extremely high correlation, they essentially contain the same information. When carrying out feature selection with a correlation matrix, it is ensured that each feature gives the model distinct information since it is guaranteed that the correlation between features is less than the correlation threshold. Hence, our method of manual feature selection produced better results than chi-squared because it didn't face the pitfall of selecting strongly correlated features.

From our experiment, it is evident that Bi-LSTM performed the best for both metrics (accuracy and FPR) and LSTM performed the worst for both metrics. The architecture of LSTM versus Bi-LSTM can help explain these results. The LSTM network introduces memory cells which have the ability to retain information over long sequences. Each memory cell has 3 main components: an input gate, an output gate, and a forget gate. The memory module of the LSTM is the core of the LSTM module since it plays a crucial role in processing long distance dependent information. It can determine whether the features in the recorded information are forgotten, and thus, the appropriate memory modules can improve the phenomenon of lowering high false positive rates.

Bi-LSTM is a variant of LSTM that consists of two parts: a forward and a backward LSTM. This allows network traffic to be stored both ways at the same time. This resolves the deficiency that LSTM has since it only stores the previous stage's information, demonstrating how Bi-LSTM can reduce false positives more effectively than LSTM can.

DDOS is nothing but an inundation of traffic. To better detect this, the model needs to have a good understanding of data packets and the information they contain over a period of time. In this scenario, having information of both the past and the future is essential to accurately detect attack patterns without incorrectly detecting normal peak traffic flow as attacks. This caters well to a Bi-LSTM architecture. Individual characteristics can be brought out by comparing it to the traditional LSTM model. LSTMs will have only one layer of processing (the forward state). Due to the single layer in only one direction, the model will not be able to properly learn relationships for the future or understand the context of past events. DDOS attack patterns are very complex in nature and understanding packet sequence in both directions are crucial to its detection hence Bi-LSTM will outperform LSTM for DDOS attacks. There are two cases to which Bi-LSTMs can help detect: SYN Flooding and http floods. Both of these attacks require deep connections about the past context for future attacks in addition which lends itself perfectly to the bidirectionality of the Bi-LSTM.

Factors in Data Processing and Training

In our CICIDS2017 dataset with 79 features, there were several instances of infinite and not a number (NaN) values, which needed to be handled in order for the dataset to be used on our model. A factor that potentially influenced our results was the way that these NaN and infinite values were managed. When initially preprocessing the data, rows containing infinite values and NaN values were removed. Rather than removing these rows, it is possible to add placeholder values, which can be calculated by taking the mean of all of the other data in the column for example. However, the number of rows removed was insignificant (4 out of about 200,000 rows) compared to the number of rows in the dataset, so it is likely that this step didn't drastically affect our results.

In addition, the method used to scale our data played a role in the results achieved. When scaling the data, both Standard Scaler and Min-max scaling were viable options. However, it was determined that Standard Scaler would be better for our model since it is possible that not all of the outliers were completely removed within main populations. Min-max scaling would have been significantly impacted by these potential outliers, which makes it less ideal than Standard Scaler for our model.

In order to train our model, accuracy metric was used for evaluation, categorical cross entropy for the loss function, and Adam as our optimizer. K-fold validation with four folds was utilized for all models, which ensured all data was used in both training and testing at least once. In addition, because the balanced dataset contains a 60-40 split of malicious to benign data, there is no need to carry out any sort of restructuring to address class imbalance. It is likely that employing a different metric, loss function, and/or optimizer would have produced different results. However, using the accuracy metric and categorical cross entropy made sense for our model because the binary

Citation	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)
J. Zhao, et al.[14]	CNN-AttBiLSTM	95.670	95.824	95.904	95.864	Low*
U. Garg, et al.[17]	Semi-supervised entropy-based ML	-	84	82	82	Low*
Sutrisno, et al.[18]	RF	99	100	99	99	0
Q. Tian, et al.[19]	Conditional Entropy Based Algorithm	97.2	100	94.2	97	0
M. Idhammad, et al.[20]	Information Theoretic Entropy-Based RF	97	-	-	-	0.33
A. Agrawal, et al.[21]	M-DBNN	87	85	-	84	0.09
This paper	Proposed Model (Bi-LSTM)	99.850	99.907	99.828	99.868	0.053

Table 7 Comparison of performance of proposed model to existing approaches

classification problem was being solved where it had to be determined if data being sent is benign or malicious. In addition, the adam optimizer is efficient, helps reduce the overall loss and improves accuracy.

Significance of Results and Future Work

Our Bi-LSTM model achieved high accuracy of attack detection and low false positive rates, which indicates its success in the simulated environment created using a dataset. This supports our initial hypothesis that Bi-LSTM is the best model to lower false positives with high accuracy, due to its bi-directional capabilities and its memory cells architecture containing an input gate, output gate, and forget gate. False positives can cause the loss of data since it prevents benign data from reaching its destination after being detected as attack data. Situations like these can be especially harmful when important pieces of benign data are being transmitted and detected as an attack. If Bi-LSTM were to be employed in these situations, it would likely help mitigate this issue. In addition, it was found that feature selection with a correlation matrix is more effective than chi-squared feature selection in our simulated environment, which supports our presumption that selecting features which aren't highly correlated produces better results since each feature provides the model with generally distinct pieces of data.

It is important to note that the proposed model's success can only be ensured in this simulated environment, which consists of offline data from a dataset. The CICIDS2017 dataset is limited by the fact that DDoS attack patterns are complex and attack parameters are changing, so detection of newly evolving attacks may be critical. For that, dynamic retraining methods would have to be deployed in such a way that new emerging attack patterns can also be known to the deep learning model, which can be addressed in future work. In addition, further testing in real time environments is necessary in order to expand the scope of our model's abilities. In the future, our hope is to further the scope of our model by running it against real time data by creation of a network architecture, including ethernet, time sensitive networks, siamese networks, etc. These networks can be effectively and practically implemented using Software Defined Networks (SDNs) because of their programmability, flexibility, and scalability. Through the use of real time data, the ability to test our model's performance under varying network conditions would be possible. A simulated environment can be employed such as Mininet, OMNeT++, or NS3 to simulate the SDN architecture so that the real time traffic can be fed into the model and then analyzed. Once the necessary resources are obtained, running our model against real time data using a simulated SDN architecture would be achievable. Through the use of real time data, it will be possible to prevent traffic that may attack critical networks, such as machines and websites in healthcare, by sending a huge volume of data and making these critical websites unavailable.

Materials & Methods

Google Scholar was used for the literature review for this paper and the standardized dataset CICIDS2017 from the Canadian Institute of Cybersecurity was used for DDoS attack analysis. Pandas was used to convert the dataset into a CSV file, which allowed it to be used for code. Google Collab was employed to code and Google sheets to keep track of the data for screening,

feature selection, and analysis. The data was pre-processed using the Python library Matplotlib for outlier screening using histogram analysis, followed by normalization using Standard Scalar from the preprocessing package in the library Scikit-learn. Feature selection was also done using Chi-squared from the Scikit-learn library and a correlation matrix.

For deep learning neural networks (Bi-LSTM, GRU, LSTM), the API Keras in the TensorFlow framework was utilized . For these DL models the following training parameters were used: 100 epochs, a batch size of 32, the adam optimizer, and the categorical crossentropy loss function. The architecture for the Bi-LSTM model was as follows: one Bi-LSTM layer of size 64 with l1 and l2 regularizers of weights 10^{-5} and 10^{-4} respectively and a Dropout layer of 20%. Both the regularizers and Dropout layer were implemented in order to prevent overfitting. The same was done for LSTM and GRU except a LSTM layer and GRU layer were used respectively instead of a Bi-LSTM layer. The final layer used for each model was the Dense Layer of size 2 with the softmax activation. This was to ensure that the output for each DL model was represented as probabilities of the data being benign or malicious. For SVM, a machine learning model, the Scikit-learn library was employed. The hyperparameters for SVM were as follows: gamma was set to auto and C was set to 0.1 for regularization in order to prevent overfitting. For all models (Bi-LSTM, GRU, LSTM, SVM), k-fold validation with 4 folds was employed in order to ensure that all data points were used as a part of both training and testing data at least once. Seaborn was used to plot a confusion matrix, allowing us to determine accuracy and false positive rates.

Acknowledgments

Thanks to my mentor, Prof. Anoop for providing guidance and constant feedback necessary to stay on track with this project and encouraging me throughout the process.

References

1 K. Sahoo, B. Tripathy, S. K.Naik, M. B.Balusamy and DBurgos, *An evolutionary SVM model for DDOS attack detection in software defined networks*, https://www.doi.org/10.1109/ACCESS.2020.3009733.

2 H. Alqahtani and M. Abdullah, *A Review on DDoS Attacks Classifying and Detection by ML/DL Models*, https://www.doi.org/10.14569/ijacsa.2024.0150283.

3 Y. Liu, P. B.Zhao, P. Fan and H. Liu, *A survey: Typical security issues of software-defined networking*.

4 D. Kumar, R. Pateriya, R. Gupta and A. V.Dehalwar, *DDoS detection using deep learning*, https://doi.org/10.1016/j.procs.2023.01.217.

5 M. Jogin, M. Madhulika, G. Divya, R. Meghana and SApoorva, *Feature extraction using convolution neural networks (CNN) and deep learning*, https://www.doi.org/10.1109/RTEICT42901.2018.9012507.

6 A. S.Haider, I. Mustafa, T. Patel, K. A.Fernandez and J. Iqbal, *A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks*, https://www.doi.org/10.1109/ACCESS.2020.2976908.

7 M. Elsayed, N. Le-Khac, S. Dev and A. Jurcut, *Ddosnet: A deep-learning model for detecting network attacks*, https://www.doi.org/10.1109/WoWMoM49955.2020.00072.

8 V. Singh, *DDOS attack detection and mitigation using statistical and machine learning methods in SDN*, https://norma.ncirl.ie/4542/1/vishalkumarsingh.pdf.

9 M. M.Ramzan, A. Altaf, S. Arshad, F. Iqbal and Castilla, *Distributed denial of service attack detection in network traffic using deep learning algorithm*, https://doi.org/10.3390/s23208642.

10 T. Y.Wang, J. G.Tang and J. Lu, *SGS: Safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking*, https://www.doi.org/10.1109/ACCESS.2019.2895092.

11 J. A.Chartuni, *Multi-classifier of DDoS attacks in computer networks built on neural networks*.

12 O. Tayfour and M. Marsono, *Collaborative detection and mitigation of distributed denial-of-service attacks on software-defined network*, https://doi.org/10.1007/s11036-020-01552-0.

13 M. Mittal, K. Kumar and S. Behal, *Deep learning approaches for detecting DDoS attacks: A systematic review*.

14 J. Zhao, Y. Liu, Q. Zhang and X. Zheng, *CNN-AttBiLSTM Mechanism: A DDoS Attack Detection Method Based on Attention Mechanism and CNN-BiLSTM*, https://www.doi.org/10.1109/ACCESS.2023.3334916.

15 F. Becerra-Suarez, I. Fernández-Roman and M. Forero, *Improvement of Distributed Denial of Service Attack Detection through Machine Learning and Data Processing*, https://www.mdpi.com/2227-7390/12/9/1294.

16 A. F. A.F.AlShammari, *Implementation of Feature Selection using Correlation Matrix in Python*.

17 U. Garg, M. Kaur, M. Kaushik, M. Kaushik and N., *Gupta "Detection of DDoS attacks using semi-supervised based machine learning approaches*, https://ieeexplore.ieee.org/abstract/document/9784741.

18 U. Sutrisno, S. Wijono, T. Wahyono, I. Sembiring and I. Widiasari, *Effective DDoS Detection through Innovative Algorithmic Approaches in Machine Learning*, https://doi.org/10.1109/ICCIT62134.2024.10701265.

19 Q. Tian and M., *Sumiko."A DDoS attack detection method using conditional entropy based on SDN traffic*, https://www.mdpi.com/2624-831X/4/2/6.

20 M. Idhammad, K. Afdel and M. Belouch, *Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest*.

21 A. Agrawal, R. Singh, S. M.Khari and SLim, *Autoencoder for Design of Mitigation Model for DDOS Attacks via M-DBNN*.