SDN based DDoS Detection using Deep Learning Techniques

Software-Defined Networking (SDN) is an emerging paradigm, which evolved in recent years to address the weaknesses in traditional networks. The SDN's main goal is to separate the control and data planes, making network management easier and enabling for more efficient programmability. The centralized structure of SDN brings new vulnerabilities. Distributed Denial-of-Service (DDoS) are the most prevalent and sophisticated threat. DDoS attack tries to disrupt the available services of a victim to block the victim from providing service to the legitimate users, by sending massive malicious requests from a large number of hijacked machines. DDoS attacks are easy to initiate, hard to defend and has strong destructive effect. So that accurate detection of DDoS attack is necessary.

Traditional machine learning approaches are impacted by lower detection rates and higher false-positive rates. Deep Learning (DL) is capable of automatically finding correlations in raw data, and so it can improve the DDOS detection rate. The DL approaches, such as the DNN, Auto Encoders, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) would increase the performance of DDoS detection.

- **Deep Neural Network (DNN)** is used for DDoS detection, goal is to tackle the problem of binary classification in the DDoS Detection System. Classify the network traffic into normal and DDOS attack traffic.

- **Auto Encoders (AE):** AE can be used as an anomaly detection algorithm. Here the DDoS attack traffic is anomaly. AE tries to minimize the reconstruction error part of its training. DDoS attacks are detected by checking the magnitude of the reconstruction loss. The DDoS attack will have higher reconstruction error than normal network traffic.

- **LSTM Auto Encoder:** The main issue in RNN is the vanishing gradient problem. To solve this and extracting long and short-term dependencies, as well as trends in DDoS attack sequences LSTM Auto Encoder can be used.

Dataset :

- **CICDDoS 2019:** CICDDoS2019 is a collection of benign and up-to-date popular DDoS attacks that closely resemble real-world data. This dataset includes a broad range of Distributed Denial of Service attacks. Newest publicly available dataset, which contain a comprehensive variety of DDoS attacks and addresses the gaps of the existing current dataset.

- DDoS attack SDN dataset– The SDN specific dataset created using the SDN architecture and includes UpToDate SDN DDoS traffic data.

Literature Survey :

NO	TITLE	TECHNIQUES USED	ADVANTAGES	DISADVANTAGES
1.	Machine Learning Approach Equipped with Neighborhood Component Analysis for DDoS Attack Detection in Software-Defined Networking [2021]	NCA KNN Decision Tree ANN SVM	NCA gives most relevant features by feature selection, UpToDate dataset	Does not give optimal number of features to be selected, The performance depends on the selected features
2.	Clustering based semi-supervised machine learning for DDoS attack classification [2019]	Agglomerative clustering K means clustering KNN, SVM, Random Forest	Optimizing and validating the model improves the performance	Clustering approach leads to high false positive values

3.	Detection of DDoS attacks with feed forward based deep neural network model [2021]	DNN	High accuracy	Failed to detect adversarial attack
4	A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks [2020]	RNN LSTM CNN RNN+LSTM	Improved accuracy Minimal computational complexity	Failed to detect adversarial DDoS attacks
5	DDoSNet: A Deep-Learning Model for Detecting Network Attacks [2020]	RNN AutoEncoder	DDoSNet gives the highest evaluation metrics in terms of recall, precision, F-score, and accuracy compared to the existing well known classical ML techniques	Vanishing gradient problem

Design Steps :

- ❑ Preprocessing of Dataset
- ❑ Implement DNN for DDoS detection
- ❑ Implement Auto Encoder for the DDoS detection
- ❑ Implement LSTM Auto Encoder for the DDoS detection
- ❑ Performance evaluation

**Framework : ** Keras and tensorflow

Initial Work :

- Apply Machine learning algorithms such as Logistic regression, SVM, MLP, Random Forest and gradient Boosting for the DDoS detection and evaluate the performance of each model.

Detailed Design Steps:-

- **Dataset Preprocessing:**

  - ❑ CICDDoS 2019 dataset in csv format is used for the experiment has been reduced to make it easier to train since it contains a large number of packages.
  - ❑ Eight features (Flow ID, SourceIP, SourcePort, DestinationIP, DestinationPort, Protocol, Timestamp, SimillarHTTP) that do not contribute to the training and 9 features (Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, Fwd Bulk Rate Avg, Bwd Bytes/Bulk Avg, Bwd Packet/ Bulk Avg, Bwd Bulk Rate Avg) containing only ’0’ value were removed from the dataset and the model was trained with 66 features.
  - ❑ ’Normal’ is labeled ’0’ and DDoS attacks are labeled ’1’ in the dataset created to detect DDoS on network traffic.

Features Selected:

No	Feature Name	Decsription
1	Flow Duration	Duration of the flow in Microsecond
2	Tot Fwd packets	Number of forward packets per second
3	Tot Bwd packets	Number of backward packets per second
4	Tot len Fwd packets	Total size of packet in forward direction
5	Tot len Bwd packets	Total size of packet in backward direction

6	Fwd packet len max	Maximum size of packet in forward direction
7	Fwd packet len min	Minimum size of packet in forward direction
8	Fwd packet len mean	Mean size of packet in forward direction
9	Fwd packet len std	Standard deviation size of packet in forward direction
10	Bwd packet len max	Maximum size of packet in backward direction
11	Bwd packet len min	Minimum size of packet in backward direction
12	Bwd packet len mean	Mean size of packet in backward direction
13	Bwd packet len std	Standard deviation size of packet in backward direction
14	Flow byte/s	Number of flow bytes per second
15	Flow packet/s	Number of flow packet per second
16	Flow IAT mean	Mean time between two packets sent in the flow
17	Flow IAT std	Standard deviation time between two packets sent in the flow
18	Flow IAT max	Maximum time between two packets sent in the flow
19	Flow IAT min	Minimum time between two packets sent in the flow
20	Fwd IAT Total	Total time between two packets sent in the forward direction
21	Fwd IAT mean	Mean time between two packets sent in the forward direction
22	Fwd IAT std	Standard deviation time between two packets sent in the forward direction
23	Fwd IAT max	Maximum time between two packets sent in the forward direction
24	Fwd IAT min	Minimum time between two packets sent in the forward direction

25	Bwd IAT Total	Total time between two packets sent in the backward direction
26	Bwd IAT mean	Mean time between two packets sent in the backward direction
27	Bwd IAT std	Standard deviation time between two packets sent in the backward direction
28	Bwd IAT max	Maximum time between two packets sent in the backward direction
29	Bwd IAT min	Minimum time between two packets sent in the backward direction
30	Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
31	Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
32	Fwd Header len	Total bytes used for headers in the forward direction
33	Bwd Header len	Total bytes used for headers in the backward direction
34	Fwd packets/s	Number of forward packets per second
35	Bwd packets/s	Number of backward packets per second
36	Packet len min	Minimum length of a packet
37	packet len max	Maximum length of a packet
38	packet len mean	Mean length of a packet
39	packet len std	Standard deviation length of a packet
40	Packet len variance	Variance length of a packet
41	FIN flag count	Number of packets with FIN
42	SYN flag count	Number of packets with SYN
43	RST flag count	Number of packets with RST

44	PSH flag count	Number of packets with PSH
45	Ack flag count	Number of packets with Ack
46	URG flag count	Number of packets with URG
47	CWE flag count	Number of packets with CWE
48	ECE flag count	Number of packets with ECE
49	Down/up ratio	Download and upload ratio
50	Packet size avg	Average size of a packet
51	Fwd seg size avg	Average size observed in the forward direction
52	Bwd seg size avg	Average size observed in the backward direction
53	Subflow Fwd packets	The average number of packets in a sub flow in the forward direction
54	Subflow Fwd bytes	The average number of bytes in a sub flow in the forward direction
55	Subflow Bwd packets	The average number of packets in a sub flow in the backward direction
56	Subflow Bwd bytes	The average number of bytes in a sub flow in the backward direction
57	Fwd Act Data packets	Count of packets with at least 1 byte of TCP data payload in the forward direction
58	Fwd seg size min	Minimum segment size observed in the forward direction
59	Active mean	Mean time a flow was active before becoming idle
60	Active std	Standard deviation time a flow was active before becoming idle
61	Active max	Maximum time a flow was active before becoming idle
62	Active min	Minimum time a flow was active before becoming idle

63	Idle mean	Mean time a flow was idle before becoming active
64	Idle std	Standard deviation time a flow was idle before becoming active
65	Idle max	Maximum time a flow was idle before becoming active
66	Idle min	Minimum time a flow was idle before becoming active
67	Label	‘0’ denotes normal and ‘1’ denotes DDoS attack

- **DDoS Detection Using Supervised Machine Learning Algorithms**
  - Load the dataset
  - Splitting the dataset into features and label
  - Splitting the dataset in to training and testing (40% for testing and 60% for training)
  - First create a Machine Learning model using **Logistic Regression** for DDoS detection
  - Evaluate the logistic Regression model
  - Create a Machine Learning model using **Gaussian Naïve bayes** algorithm
  - The Gaussian Naïve bayes model is trained and tested in the dataset
  - Evaluate the performance of Gaussian Naïve bayes model
  - Create a Machine Learning model using **Support Vector Machine (SVM)** algorithm
  - Train the SVM model on the dataset and test the model
  - Evaluate the performance of SVM model for DDoS detection
  - Create a Machine Learning model using **Decision Tree** algorithm
  - Train the Decision Tree model in the dataset and test the model
  - Evaluate the performance of Decision Tree model for DDoS detection
  - Create a Machine Learning model using **Random Forest** algorithm
  - Train the Random Forest model in the dataset and test the model
  - Evaluate the performance of Random Forest model for DDoS detection
  - Create a Machine Learning model using **Gradient Boosting** algorithm
  - Train the Random Forest model in the dataset and test the model
  - Evaluate the performance of Random Forest model for DDoS detection

- Results:

Evaluation Metrics	LR	GNB	SVM	DT	RF	GB
Accuracy (%)	77.11	67.35	78.44	99.95	100	98.52
Precision (%)	72.68	57.54	76.16	99.93	100	100
Recall (%)	66.29	62.41	65.17	99.95	100	100
F1 score (%)	75.54	66.17	76.67	99.95	99.95	98.46

- **Deep Neural Network (DNN) for DDoS Detection**

  - Load the dataset
  - Splitting the dataset into features and label
  - Splitting the dataset in to training and testing (20% for testing and 80% for training)
  - Feature Scaling or Standardization using Standard Scaler
  - Create DNN with sequential model having input layer, 3 hidden layer and output layer
  - The input layer consist of 19 neurons corresponds to 19 features selected. The relu activation function used
  - The hidden layers consist of equal number of 50 neurons and relu activation function is used
  - There are also two dropout layers with dropout ratio 0.2
  - The output layer consists of 2 neurons with sigmoid activation function because it is a binary classification problem
  - The output label ‘0’ indicates the network traffic is normal and ‘1’ indicates the traffic is DDoS attack
  - Compile the model using ‘Sparse categorical cross entropy’ loss function, ‘accuracy’` metrics and ‘adam’ optimizer
  - Fit the model with 50 epochs and batch size selected is 16
  - Evaluate the DNN model
  - Obtain the test loss and test accuracy
    - Result:- The accuracy obtained is **99.65 %**

- **Auto Encoder (AE) for DDoS Detection**

The Auto Encoder accepts high dimensional input data, compresses down to the latent space representation in the bottleneck hidden layer. The Decoder takes the latent representation of the data as an input to reconstruct the original input data. AE tries to minimize the reconstruction error part of its training. Anomalies are detected by checking the magnitude of the reconstruction loss.

- Import the required libraries and load the dataset
- The dataset consists of label '0' and '1', 0 denotes normal and 1 denotes the attack (Anomaly)
- Split the data for training and testing (20% for testing)
- Scale the data using Minmax scaler
- Use normal data only for training

  AE are trained to minimize the reconstruction error. The reconstruction errors are used as the anomaly score. When we train the AE on normal data, we can hypothesize that the anomalies will have higher reconstruction errors than normal data.
- Create an Auto Encoder class with output units equal to number of input data and the number of units in the bottleneck (code size) equal to 16
- The encoder of the model consists of 4 layers that encode the data into lower dimensions
- The decoder of the model consists of 4 layers that reconstruct the input data
- The model is compiled with metrics equal to mse and adam optimizer
- The model is trained with 200 epochs with a batch size of 16
- The reconstruction errors are considered as anomaly score. The Threshold is then calculated by summing the mean and standard deviation of the reconstruction errors
- The reconstruction error above this threshold is DDoS attack (anomalies)
  - Result : Accuracy obtained is **68.27 %**
- Optimize the model with keras tuner
  - Accuracy improved to **71.39 %**

- **Auto Encoder and XGBOOST for DDoS Detection**

  Auto Encoder is a type of neural network that can be used to learn a compressed representation of raw data. An autoencoder is composed of an encoder and a decoder sub-model. The encoder compresses the input, and the decoder attempts to recreate the input from the compressed version provided by the encoder. After training, the encoder model is saved, and the decoder is discarded. The encoder can then be used as a data preparation technique to perform feature extraction on raw data that can be used to train a different machine learning model.

  - Importing the required libraries
  - Load the dataset
  - Split the dataset in to train and test
  - Scale the dataset using Min Max scaler
  - Define the Auto Encoder model with encoder model, bottleneck, and decoder model
  - Fit the autoencoder model to reconstruct input
  - Define and save the encoder model without decoder
  - Compress the input data using encoder model
  - Import the XGBoost classifier
  - Encode the train data
  - Encode the test data
  - Fit the XGBoost classifier model on the training set
  - Make predictions on the test data
  - Calculate classification accuracy
    - Result :

Evaluation Metric	Obtained Value
Accuracy	97.58 %
Precision	97.41 %
Recall	96.25 %
F1 score	97.43 %

- **K Means Clustering for DDoS Detection**

  K means clustering is an iterative clustering method that segments data into K clusters in which each observation belongs to the cluster with the nearest mean (Cluster Centroid)

  - Importing the required libraries
  - Load the data
  - Transform the data
  - PCA for dimensionality reduction
  - Here we have 2 clusters, Normal and Attack cluster
  - KMeans.fit_predict method returns the array of cluster labels each data point belongs to
  - Visualize cluster with label ‘0’ (Normal) using matplot library
  - Plot all clusters
  - Plot cluster centroids
  - Calculate the labelling accuracy of the model
    - Result :

Evaluation Metric	Obtained Value
Accuracy	51.01 %
Silhouette Coefficient	0.706
Homogeneity	0.046
Completeness	0.070
V-measure	0.056

- **DBSCAN for DDoS Detection**

  - Import the required libraries
  - Load the dataset
  - Compute DBSCAN
  - Set epsilon = 1 and min samples = 100
  - Fit the model
  - Evaluate the metrics
    - Results:-

Evaluation Metric	Obtained Value
Silhouette Coefficient	0.706
Homogeneity	0.146
Completeness	0.070
Adjusted Rand Index	-0.021
V-measure	0.094
Adjusted Mutual Information	0.094

References:-

[1]. Elsayed, M.S., Le-Khac, N.A., Dev, S. and Jurcut, A.D., 2020, August. Ddosnet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)* (pp. 391-396). IEEE.

[2]. Cil, A.E., Yildiz, K. and Buldu, A., 2021. Detection of DDoS attacks with feed forward based deep neural network model. *Expert Systems with Applications*, *169*, p.114520.

[3]. Haider, S., Akhunzada, A., Mustafa, I., Patel, T.B., Fernandez, A., Choo, K.K.R. and Iqbal, J., 2020. A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *Ieee Access*, *8*, pp.53972-53983.

[4]. Tonkal, Ö., Polat, H., Başaran, E., Cömert, Z. and Kocaoğlu, R., 2021. Machine Learning Approach Equipped with Neighbourhood Component Analysis for DDoS Attack Detection in Software-Defined Networking. *Electronics*, *10*(11), p.1227.

[5]. Catak, F.O. and Mustacoglu, A.F., 2019. Distributed denial of service attack detection using autoencoder and deep neural networks. *Journal of Intelligent & Fuzzy Systems*, *37*(3), pp.3969-3979.

Guide Details:

Name : Prof. Sumod Sundar

Email: sumodsundar@tkmce.ac.in