A Transfer learning-based Approach towards Detection of DDOS Attacks (Botnets)

Tejas Pradhan
Computer Department
Vit Pune
Pune , India

tejas.pradhan18@vit.edu

Sakshi Oswal
Computer Department
Vit Pune
Pune , India

sakshi.oswal18@vit.edu

Pranjal Rane
Computer Department
Vit Pune
Pune , India

rane.pranjal18@vit.edu

Saurabh Rane
Computer Department
Vit Pune
Pune , India

saurabh.rane18@vit.edu

***Abstract*—Distributed Denial of Service Attacks (DDOS) are a common threat to large network servers akin to that of E-commerce websites like Amazon, Flipkart, PayPal, etc. A brute force attack by distributed botnets can completely disrupt the servers and lead to data loss and major security breaches. Our system aims to detect such systems at an early stage by intercepting the first few HTTP requests and analyzing the signatures. The data obtained will be further analyzed through deep learning-based pattern recognition algorithms to provide a full proof and accurate method for detection and reporting at the server side. Our solution consists of 2 components, the first component consists of a Network Analyzer which collects live network parameters which will be given to our prediction model using Deep Learning Techniques to predict botnet traffic from benign traffic and alert the customer about existence of Botnet Network.**

Keywords—Botnet, Detection, DDOS Attacks, Deep Learning, Network Analyzer.

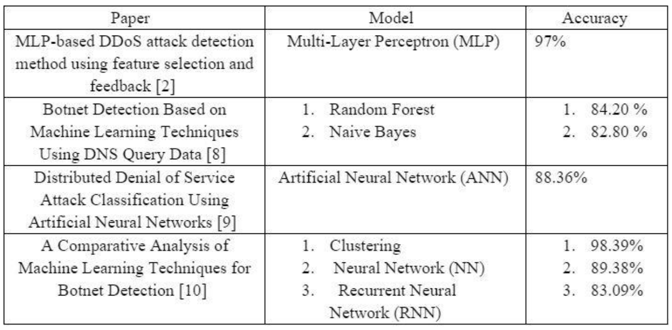
I. Introduction

DDoS attacks aim to stop legitimate users from normally accessing a specific network service by continuously sending a large amount of traffic to the target system. To achieve this attempt, hackers usually use Botnets to launch a DDoS attack. Botnets are the network formed by forming a network of host computers, called bots, that are controlled by one or more attackers, called botmasters, with the intention of performing malicious activities, in this research paper we are going to focus on the DDOS attacks. Botnets, as one of the foremost formidable cyber security threats, are getting more sophisticated and immune to detection. In spite of specific behaviors each botnet has, there are enough similarities inside each botnet that separate its behavior from benign traffic. Several botnet detection systems that are been proposed show these similarities. However, offering an answer for differentiating botnet traffic (even those using same protocol, e.g. IRC) from normal traffic is not trivial. Extraction of features in either host or network level to model a botnet has been one among the foremost popular methods in botnet detection. A subset of features, usually selected supported some intuitive understanding of botnets and is employed by the machine learning algorithms to classify/cluster botnet traffic.

II. Literature Review

DDoS attacks aim to stop legitimate users from normally accessing a specific network service by continuously sending a large amount of traffic to the target system. To achieve this attempt, hackers usually use Botnets to launch a DDoS attack. Botnets are the network formed by forming a network of host computers, called bots, that are controlled by one or more attackers, called botmasters, with the intention of performing malicious activities, in this research paper we are going to focus on the DDOS attacks. One of the latest research paper we referred was - A dynamic MLP-based DDoS attack detection method using feature selection and feedback (2020) [1] It claims that removing srcIP or TCP flag leads to a sharp degradation on the performance, but on the contrary, removing the other three features does not cause significant changes. According to this research paper changing architecture of model is not affecting accuracy of detection significantly whereas feature selection plays a very prominent role in the accuracy of model. This paper has used five features, including source IP address, TCP sequence, TCP flag, TCP source and destination port.

The unique part about this paper was the Feedback Mechanism where if we use the new labelled samples in this duration to retrain the detection model, the detection accuracy of the retrained model on test data will appear distinguishable reduction compared with the original model. On the contrary, if the detector is well-performed, the accuracy reduction should only fluctuate in a limited range. Therefore, they determined whether the detector is making considerable misclassification by comparing the accuracy reduction with a pre-set threshold, which represents the normal accuracy reduction profile under the stable condition when the detector performs well. If current accuracy reduction is bigger than the threshold, the feedback mechanism will trigger a signal to update and reconstruct the detection model. Another one we referred was - Machine Learning DDoS Detection for Consumer Internet of Things Devices (11 April 2018) [2] Botnets such as Mirai have used insecure consumer IoT devices to conduct distributed denial of service (DDoS) attacks on critical Internet infrastructure. In this paper, they demonstrated that using IoT-specific network behaviors (e.g. limited number of endpoints and regular time intervals between packets) to inform feature selection can result in high accuracy DDoS detection in IoT network traffic with a variety of machine learning algorithms, including neural networks. They compared a variety of classifiers for attack detection, including random forests, K-nearest neighbors, support vector machines, decision trees and neural network all giving an accuracy higher than 0.9. IoT traffic is often distinct from that of other Internet connected devices (e.g. laptops and smart phones) like IoT devices often communicate with a small finite set of endpoints rather than a large variety of web servers. IoT devices are also more likely to have repetitive network traffic patterns, such as regular network pings with small packets at fixed time intervals. Hence, they claimed that IoT traffic is different from other types of network behavior, as laptops and smart phones access a large number of web endpoints due to web browsing activity, IoT devices tend to send automated pings to a finite number of endpoints. Anomaly detection aims to identify patterns in data that do not conform to expected behavior. In the context of our work, anomaly detection techniques may be used to discern attack traffic from regular traffic. We found that simple threshold-based techniques are prone to incorrectly classifying normal traffic as anomalous traffic and are unable to adapt to the evolving nature of attacks. More sophisticated anomaly detection algorithms, particularly those using machine learning can help minimize false positives. Such approaches include deep neural networks, which promise to outperform traditional machine learning techniques for sufficiently large datasets. Fig.1 shows the comparison of our results with the other related work done before.

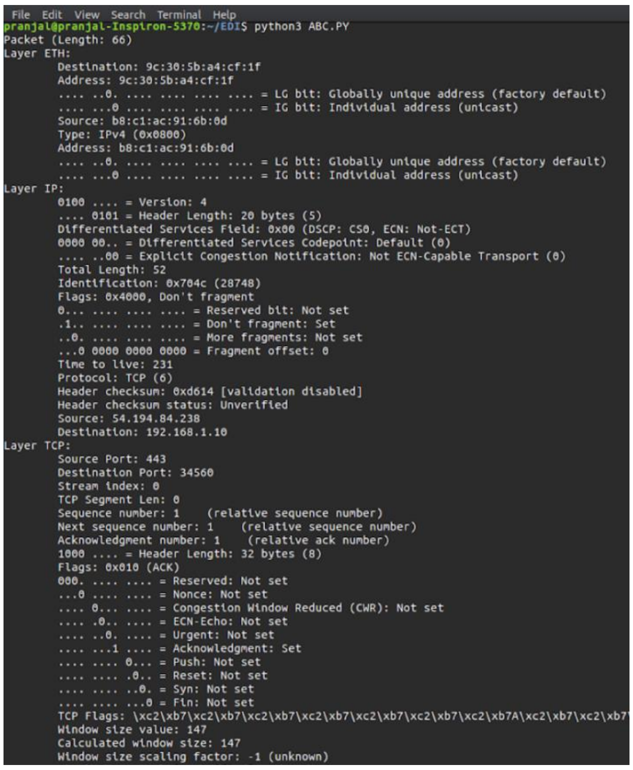


Paper	Model	Accuracy
MLP-based DDoS attack detection method using feature selection and feedback [2]	Multi-Layer Perceptron (MLP)	97%
Botnet Detection Based on Machine Learning Techniques Using DNS Query Data [8]	1. Random Forest 2. Naive Bayes	1. 84.20 % 2. 82.80 %
Distributed Denial of Service Attack Classification Using Artificial Neural Networks [9]	Artificial Neural Network (ANN)	88.36%
A Comparative Analysis of Machine Learning Techniques for Botnet Detection [10]	1. Clustering 2. Neural Network (NN) 3. Recurrent Neural Network (RNN)	1. 98.39% 2. 89.38% 3. 83.09%

Fig. 1. Comparison of Results with Previous approaches

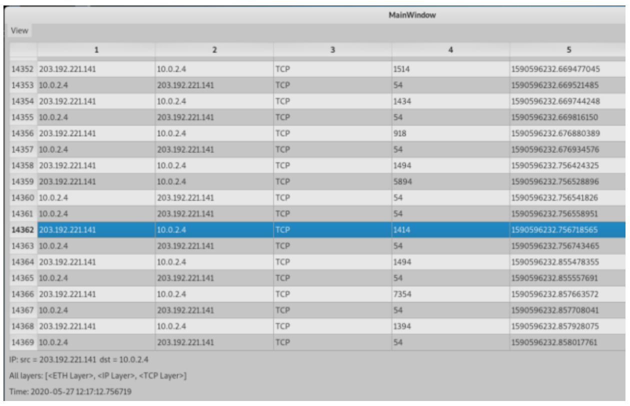
III. Methodology

Our solution mainly consists of 2 steps, the first part is to design a network analyzer and retrieve the essential feature from the system and then feed this data to model to do the prediction. The first step to detect a distributed denial of service (DDOS) is to be able to analyze a network. To analyze the network, we developed our own network analyzer. We developed it in Python using the pyshark library [6]. Pyshark is a python wrapper for tshark, allowing python packet parsing using Wireshark dissectors. The first step to this was retrieving the packets transferred. After being able to retrieve the packet contents as shown in fig.2 we then, filter it out and display only the essential contents, i.e. Source address, destination address, packet size and timestamp as shown in Fig. 23 These parameters are important for our detection and thus predicting a DDOS attack. Our system also identifies ARP spoofing which is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network. This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol. We used the arp_table library [10] for this purpose. Technlogies used are PyQt fir UI backend [11]and QT4 designer for frontend [12] for our desktop application.



```
 File  Edit  View  Search  Terminal  Help
pranjal@pranjal-Inspiron-5370:~/EDI$ python3 ABC.PY
Packet (Length: 66)
Layer ETH:
        Destination: 9c:30:5b:a4:cf:1f
        Address: 9c:30:5b:a4:cf:1f
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        Source: b8:c1:ac:91:6b:0d
        Type: IPv4 (0x0800)
        Address: b8:c1:ac:91:6b:0d
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
Layer IP:
        0100 .... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
        Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
        Total Length: 52
        Identification: 0x704c (28748)
        Flags: 0x4000, Don't fragment
        0... .... .... .... = Reserved bit: Not set
        .1.. .... .... .... = Don't fragment: Set
        ..0. .... .... .... = More fragments: Not set
        ...0 0000 0000 0000 = Fragment offset: 0
        Time to live: 231
        Protocol: TCP (6)
        Header checksum: 0xd614 [validation disabled]
        Header checksum status: Unverified
        Source: 54.194.84.238
        Destination: 192.168.1.10
Layer TCP:
        Source Port: 443
        Destination Port: 34560
        Stream index: 0
        TCP Segment Len: 0
        Sequence number: 1    (relative sequence number)
        Next sequence number: 1    (relative sequence number)
        Acknowledgment number: 1    (relative ack number)
        1000 .... = Header Length: 32 bytes (8)
        Flags: 0x010 (ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        TCP Flags: \xc2\xb7\xc2\xb7\xc2\xb7\xc2\xb7\xc2\xb7\xc2\xb7\xc2\xb7A\xc2\xb7\xc2\xb7
        Window size value: 147
        Calculated window size: 147
        Window size scaling factor: -1 (unknown)
```

Fig. 2 data packet shared over a network



MainWindow

View

	1	2	3	4	5
14352	203.192.221.141	10.0.2.4	TCP	1514	1590596232.669477045
14353	10.0.2.4	203.192.221.141	TCP	54	1590596232.669521485
14354	203.192.221.141	10.0.2.4	TCP	1434	1590596232.669744248
14355	10.0.2.4	203.192.221.141	TCP	54	1590596232.669816150
14356	203.192.221.141	10.0.2.4	TCP	918	1590596232.676880389
14357	10.0.2.4	203.192.221.141	TCP	54	1590596232.676934576
14358	203.192.221.141	10.0.2.4	TCP	1494	1590596232.756424325
14359	203.192.221.141	10.0.2.4	TCP	5894	1590596232.756528896
14360	10.0.2.4	203.192.221.141	TCP	54	1590596232.756541826
14361	10.0.2.4	203.192.221.141	TCP	54	1590596232.756558951
14362	203.192.221.141	10.0.2.4	TCP	1414	1590596232.756718565
14363	10.0.2.4	203.192.221.141	TCP	54	1590596232.756743465
14364	203.192.221.141	10.0.2.4	TCP	1494	1590596232.855478355
14365	10.0.2.4	203.192.221.141	TCP	54	1590596232.855557691
14366	203.192.221.141	10.0.2.4	TCP	7354	1590596232.857663572
14367	10.0.2.4	203.192.221.141	TCP	54	1590596232.857708041
14368	203.192.221.141	10.0.2.4	TCP	1394	1590596232.857928075
14369	10.0.2.4	203.192.221.141	TCP	54	1590596232.858017761

IP: src = 203.192.221.141 dst = 10.0.2.4

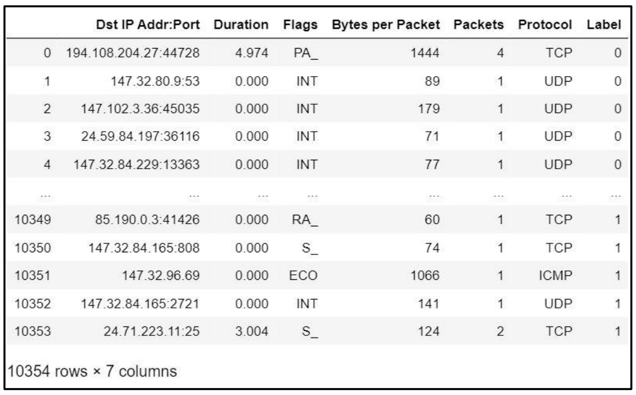
All layers: [<ETH Layer>, <IP Layer>, <TCP Layer>]

Time: 2020-05-27 12:17:12.756719

Fig. 3. Feature Extraction from Network Traffic Analyzer

After extracting the features from network analyzer, we then feed these features to our model to predict botnet traffic from benign traffic. We have trained our model using the CTU-13 Dataset [5]. The CTU-13 is a dataset of botnet traffic that was captured in the CTU University, Czech Republic, in 2011. This dataset has a large capture of real botnet traffic mixed with normal traffic and background traffic. The CTU-13 dataset consists in thirteen captures (called scenarios) of different botnet samples in which we have used the 4th dataset as it captures botnet network targeting DDOS attack [4]. We reduced the features in dataset to the prominent ones as shown in fig. 4. The model training was the most important phase of the research. For the purpose of feature selection – we have selected 4 features namely:

1. Time interval between packets
2. Number of Packets sent
3. Protocol used
4. Destination Port number

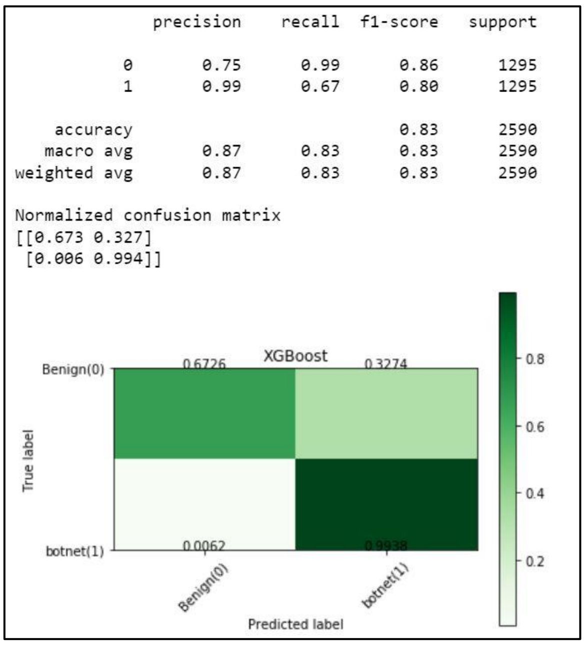


	Dst IP Addr:Port	Duration	Flags	Bytes per Packet	Packets	Protocol	Label
0	194.108.204.27:44728	4.974	PA_	1444	4	TCP	0
1	147.32.80.9:53	0.000	INT	89	1	UDP	0
2	147.102.3.36:45035	0.000	INT	179	1	UDP	0
3	24.59.84.197:36116	0.000	INT	71	1	UDP	0
4	147.32.84.229:13363	0.000	INT	77	1	UDP	0
...	...	...	...	...	...	...	...
10349	85.190.0.3:41426	0.000	RA_	60	1	TCP	1
10350	147.32.84.165:808	0.000	S_	74	1	TCP	1
10351	147.32.96.69	0.000	ECO	1066	1	ICMP	1
10352	147.32.84.165:2721	0.000	INT	141	1	UDP	1
10353	24.71.223.11:25	3.004	S_	124	2	TCP	1

10354 rows × 7 columns

Fig. 4. Features of CTU-13 Dataset

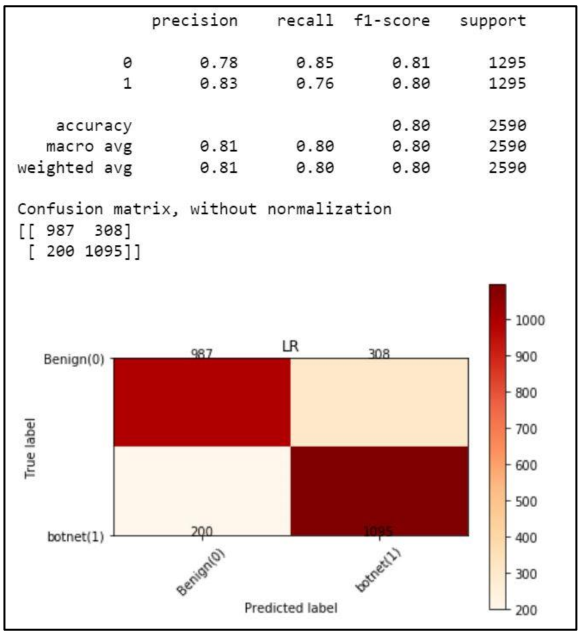
These features were finalized after analysis of the botnet data. The correlation of these features with the possibility of a botnet was the highest, With the help of these features 4 machine learning and 3 deep learning models were trained. As our problem is a classification problem, we have used 4 classification algorithms. For Xtreme Gradient Boost we have learning rate parameter as 0.001 and by tuning this parameter the largest accuracy we received with this algorithm is 83% with precision of 0.99 for Botnet classification and 0.75 for benign traffic. Hence, it works quite good for classifying the botnet traffic but it gives a large false positive for normal traffic which is a disadvantage as described in fig.5.



```
              precision    recall  f1-score   support

           0       0.75      0.99      0.86      1295
           1       0.99      0.67      0.80      1295

    accuracy                           0.83      2590
   macro avg       0.87      0.83      0.83      2590
weighted avg       0.87      0.83      0.83      2590

Normalized confusion matrix
[[0.673 0.327]
 [0.006 0.994]]
```

Fig.5. Confusion Matrix of XGBoost Model

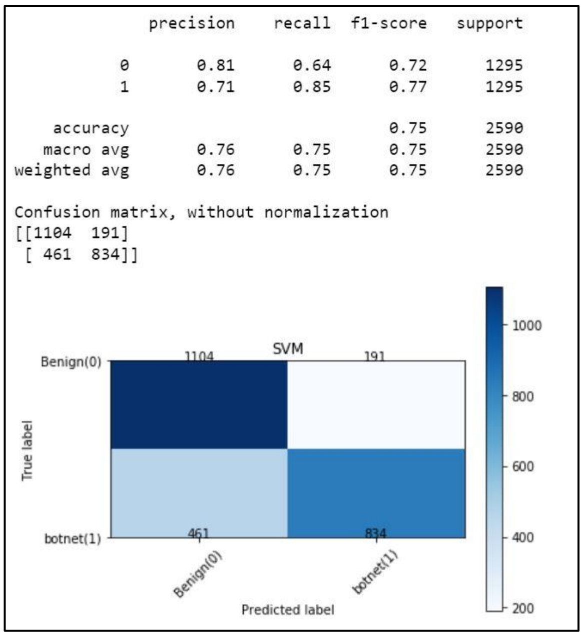
With Logistic Regression the initial accuracy was 76% hence we improved the regularization parameter C to 0.001 and obtained an accuracy of 80% with the solver parameter used is ‘liblinear’.The precision and specificity of model can be referred in fig.6.



```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81      1295
           1       0.83      0.76      0.80      1295

    accuracy                           0.80      2590
   macro avg       0.81      0.80      0.80      2590
weighted avg       0.81      0.80      0.80      2590

Confusion matrix, without normalization
[[ 987  308]
 [ 200 1095]]
```

Fig.6. Confusion Matrix of Logistic Regression Model

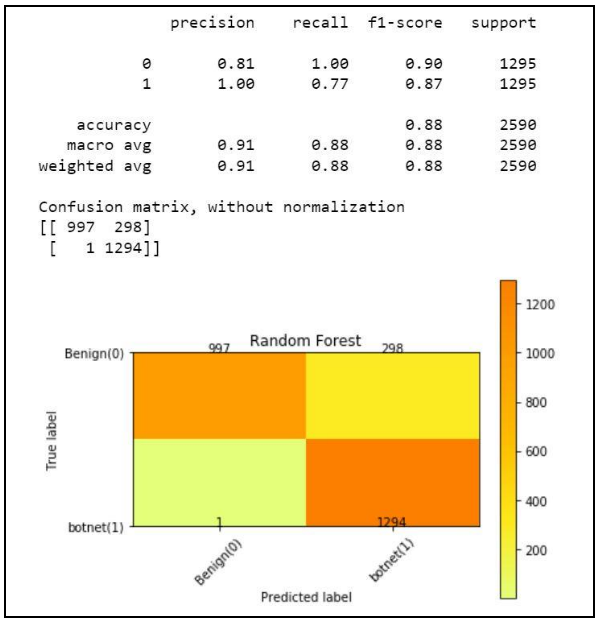
For SVM we had the lowest accuracy of 75% we did try to tune the hyperparameters and found the kernel as radial bias function (RBF) and Regularization parameter(C) as 0.001 as most optimum but this algorithm gave us low precision and recall due to high bias of the dataset as shown in fig.7.



```
              precision    recall  f1-score   support

           0       0.81      0.64      0.72      1295
           1       0.71      0.85      0.77      1295

    accuracy                           0.75      2590
   macro avg       0.76      0.75      0.75      2590
weighted avg       0.76      0.75      0.75      2590

Confusion matrix, without normalization
[[1104  191]
 [ 461  834]]
```

Fig.7. Confusion Matrix of SVM Model

Lastly, the highest accuracy algorithm for our project is Random Forest Classifier which gave us an accuracy of 88% with parameters set as class weight equal to balanced and max depth 4 .It gave us a precision of 1.00 for botnet classification whereas it has 0.81 for benign traffic which is still better than other algorithms as demonstrated in fig.8. We aim to reduce the false positive for normal traffic in our future work.



```
              precision    recall  f1-score   support

           0       0.81      1.00      0.90      1295
           1       1.00      0.77      0.87      1295

    accuracy                           0.88      2590
   macro avg       0.91      0.88      0.88      2590
weighted avg       0.91      0.88      0.88      2590

Confusion matrix, without normalization
[[ 997  298]
 [   1 1294]]
```

Fig.8. Confusion Matrix of Random Forest Model

The first deep learning model we deployed was the Single layer Perceptron. The data was fed into a shallow – relu activated neural network with a sigmoid output neuron. This model gave us an accuracy of 87%. A deep Multi-Layer Perceptron was also trained with 3 hidden layers. This gave an accuracy of 88%. Since the data was numerical, the accuracy did not increase a lot even after increasing the depth of the neural network.The third model was a Convolutional Neural Network. 1D convolutions of the data were fed as an input to the CNN layer followed by a max pooling layer. Output layer was a single sigmoid neuron. The CNN model gave an accuracy of 84 %. The final system consists of the multi-layer perceptron model.

Our final application GUI before and after botnet DDOS attacks is shown in fig.9 and fig.10.Here the green graph denotes time difference between two packets arrived at server while the blue shows length of packets whereas the red one depicts number of packets arriving per second at the server and brown one is plot of number of packets sent by the attacker to the server per second.

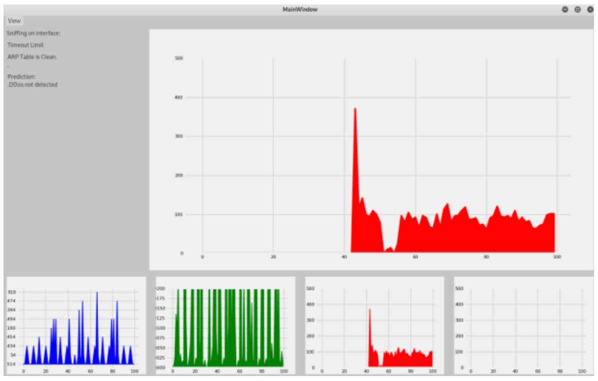


Fig.9. Network Traffic before DDOS Attack.

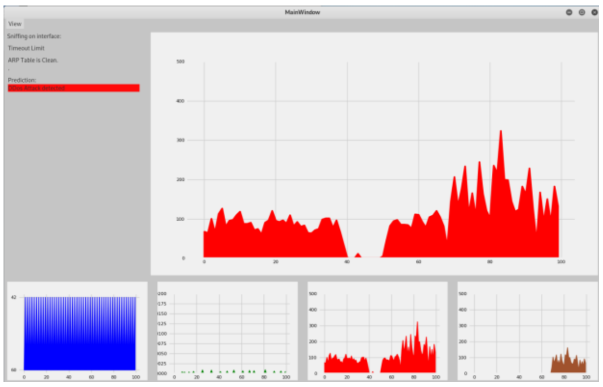


Fig.10. Network Traffic after Botnet DDOS Attack.

IV. Conclusion

By using transfer learning, we were able to model the scenario of a botnet attack and build an AI based system to recognize such attacks. This system can be set up at the server and alert the customer when a DDOS attack occurs due to Botnet Network. Transfer learning has helped in generalization of the botnet scenarios. This has made the classifier more robust and it can be set up at any server regardless of the average traffic and the location of the server. The proposed system will also continuously monitor the traffic which makes it easier to notify the user at an early stage of the attacks.

V. Future Scope

The work presented in this paper focuses only on the detection of botnets attacks. The research can be taken further to develop accurate prevention strategies based on the results of the detection system presented in this paper. This will help in making a complete intrusion detection and prevention system. The work can be taken further to develop a detection system for more attacks related to botnets apart from DDOS.

VI. Acknowledgments

Our project required guidance and assistance from people and we are extremely privileged to have got this all along the completion of the project. all that we have done is somewhere due to such supervision and assistance and we would not forget to thank them. We respect and thanks to our director sir for providing us an opportunity to do the project work in Vishwakarma institute of technology and giving us all the support and guidance, which made guided us to the right path. we owe deep gratitude to our project guide Prof. Saraswati Patil who took keen interest in our project work and guided us all along by providing all the necessary information detecting hate speech.

VII. References

[1] Meng Wang, Yiqin Lu, Jiancheng Qin, A dynamic MLP-based DDoS attack detection method using feature selection and feedback,Computers & Security,Volume 88,2020,101645,ISSN 0167-4048,(https://doi.org/10.1016/j.cose.2019.101645) https://www.sciencedirect.com/science/article/pii/S0167404819301890

[2] R. Doshi, N. Apthorpe and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, 2018, pp. 29-35, doi: 10.1109/SPW.2018.00013. https://ieeexplore.ieee.org/abstract/document/8424629/

[3] Chowdhury, Md. Nasimuzzaman, Ken Ferens and Mike Ferens. “Network Intrusion Detection Using Machine Learning.” (2016). https://www.semanticscholar.org/paper/Network-Intrusion-Detection-Using-Machine-Learning-Chowdhury-Ferens/a30c16f5598ba18ffd7d9c533515cf671d54b382

[4] THE CTU-13 DATASET. A labeled dataset with normal, botnet and background traffic. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-45/

[5] An empirical comparison of botnet detection methods" Sebastian Garcia, Martin Grill, Honza Stiborek and Alejandro Zunino. Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123 http://dx.doi.org/10.1016/j.cose.2014.05.011

[6] PyShark open source library https://github.com/KimiNewt/pyshark

[7] Hoang, X.D.; Nguyen, Q.C. Botnet Detection Based On Machine Learning Techniques Using DNS Query Data. Future Internet 2018, 10, 43. https://www.mdpi.com/1999-5903/10/5/43

[8] Distributed Denial of Service Attack Classification Using Artificial Neural Networks https://wvvw.easychair.org/publications/preprint_download/D1fr

[9] Ankit Bansal and Sudipta Mahapatra. 2017. A comparative analysis of machine learning techniques for botnet detection. In Proceedings of the 10th International Conference on Security of Information and Networks (SIN ’17). Association for Computing Machinery, New York, NY, USA, 91–98. DOI https://doi.org/10.1145/3136825.3136874

[10] Arp_table library for ARP Spoofing Detection https://pypi.org/project/python_arptable/

[11] PyQT used as backend - https://www.tutorialspoint.com/pyqt/index.html

[12] QT4 Designer as frontend - https://doc.qt.io/qt-5/qtdesigner-manual.html