Classifying DDoS Attack Categories Using Machine Learning Algorithms

Jiankai Xu
The Ohio State University

Peng Zhou
University of California, Irvine

Hanqi Zhang
University of North Carolina, Chapel Hill

Abstract

Network providers and users have been suffering from distributed denial-of-service (DDoS) attacks for quite a long time. Networking communications via massive data interchange create a favorable platform for us to use statistic machine learning algorithms to detect, classify, and analyze the features of DDoS attacks when it happen. This paper applies multiple machine learning algorithms to detect and classify the categories of DDoS attacks, including in Logistic Regression, Random Forest, and Neural Network classifiers, using datasets captured from real-world network packets. The performance of different algorithms is evaluated and compared with accuracy, precision, recall, and F1 score to show their effectiveness in detecting DDoS attacks.

Keywords: DDos attack, machine learning, logistic regression, random forest, neural network

1 Introduction

With the rapid development of internet services and theincrease of internet users, cybersecurity problems have gradually become urgent issues for IT companies. Computer virus, data breach, and other forms of attack severely damage internet environment and deal huge harm to innocent internet users. Among those online attacks, a Denial of Service (Dos) attack affects users' experience in a high level under which the user cannot access to the site they want. Even worse, a Distributed Denial of Service (DDos) attack denies the access of other legitimate users to shared services or resources [8]. Usually, in this attack, legitimate users are deprived of using web-based services by a large number of compromised machines.

DDoS attacks paralyze servers, make the user unable to access their desired resources on the internet, and make the maintainer difficult to resolve the issue in time. For example, on February 28, 2018, GitHub was hit with a sudden onslaught of traffic that clocked in at 1.35 terabits per second. The amount of traffic is record-breaking, which is 15 times greater than the normal traffic. According to GitHub, the traffic was traced back to "over a thousand different autonomous systems (ASNs) across tens of thousands of unique endpoints."[18]

DDoS attacks can be implemented in network, transport and application layers using different protocols, such as TCP, UDP,ICMP and HTTP [23]. To defend network security, companies and researchers have been conducting researches on DDoS attack prevention and detection. They proposed different useful methods to strengthen the defence of networks in order to prevent their servers from DDoS attacks. Huang et al. [12] propose an extension in software-defined wireless sensor networks called "FuzzyGuard" to protect the normal forwarding of data flow in the attacked state and has a good defensive effect on the control plane saturation attack with lower resource requirements. Saxena et al.[21] implement an application based on Hadoop and MapReduce framework and show the effectiveness of the method for DDoS attack prevention and mitigation.

However, when countermeasures for DDoS attacks have been developed, newer methods of attacks are being emerged to bypass older defensive measures, and the number of attacks has been larger than ever [5]. Therefore, it is still necessary to enable servers to detect and classify newest forms of DDoS attack packets, so that when DDoS attacks happen, the foremost action a server could take is to detect and classify them, then the server can take corresponding measures to defend against each category of attack. In this paper, we applied machine learning algorithms to build classifiers of DDoS attack based on real-world datasets and made comparisons of their performance.

2 Related Work

2.1 Related Intrusion Detection Datasets

There are several datasets available for DDoS attack detection and classification that are widely used by researchers. Kdd Cup 99 dataset provides six types of DoS attack, along with 16 categories of User to Root (U2R) attack, Remote to Local (R2L) attack, and probing attack [14]. However, this dataset is generated artificially using scripts, and the data is not similar to real network traffic [25]. Similarly, DARPA 2000 dataset provides various types of attacks in three different scenarios, but this dataset was produced by naïve attackers and naïve defenders, and it does not resemble the behaviors of modern advanced attackers [16]. The CAIDA DDoS Attack 2007 dataset provides an hour of different types of real-world attack data gathered in diverse locations [4], but it does not contain modern intrusion techniques.

In this paper, we are using DDoS Evaluation Dataset (CICDDoS2019) [23], it contains benign and currently most common attacks collected from real-world data, including in modern reflective DDoS attacks such as NTP, NetBIOS, SSDP, UDP-Lag, and TFTP that are not included in earlier datasets. In this dataset, the taxonomy of DDoS attacks is explained in term of reflection-based attacks and exploitation-based attacks, both can be carried out based on TCP or UDP. Figure 1 illustrates the details of taxonomy of DDoS attacks.

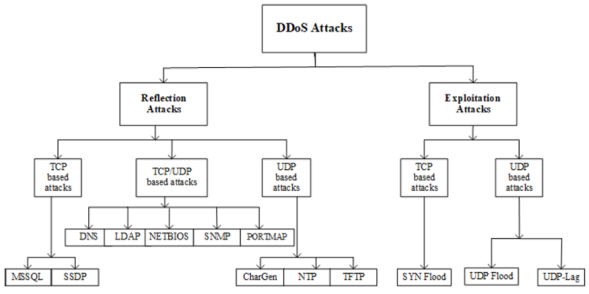

Figure 1: DDoS Attack Classification

2.2 Previous work on DDoS Attack Detection

Many solutions have been proposed to prevent and detect DDoS attacks, such as the trace back scheme and traffic filtering autonomous system[15]. The trace back scheme finds the locations of sources for attacks, while a traffic filtering autonomous system uses a traffic filter to drop traffic that does not originate from the network or is destined to the network[20]. Gong et al. propose an "Improved Quantum Genetic Algorithm" which combines improved quantum genetic algorithm with BP neural network, and uses KDD-Cup 1999 data set to detect DDoS attacks [9].

Some other researchers apply machine learning to this topic. Dong et al.[6] use K nearest neighbor(KNN) to discover DDoS attacks. Amair et.al[2] apply feature engineering to the learning process and receive increased effectiveness. He again works on using clustering[1] to elevate the performance of DDoS detection. Meejoung Kim[15] uses hyperparameter tuning to obtain a decent result which suggests that adjusting models could improve the detection rate. Manjula et al.[24] makes a shallow comparison among classic machine learning algorithms; however, his work only provides results but neglects the detailed implementation of those algorithms, which makes his work less convincing. Although those abovementioned works show some strength in detecting DDoS attacks, the amount of works regarding DDoS attack classification using machine learning algorithms is relatively fewer than those focusing on detecting DDoS attacks.

3 Methodology

3.1 Logistic Regression

Logistic Regression [11] is a machine learning model that uses logistic function to model dependent variable according to input variables. In a basic logistic function, we define

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1}$$

where "z" is the product of feature values and parameter vector

$$z = x^T\theta \tag{2}$$

The logistic function forms a “S” shape in coordinates; the output greater than 0 are classified as class “1” and the outputs less than 0 are classified as class “0”. In logistic regression algorithms, the parameters θ are optimized by minimizing loss function.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i log(\sigma(\theta x^i)) + (1-y^i) log(1-\sigma(\theta x^i)) \tag{3}$$

where "m" is the number of training samples [13].

Logistic regression models are most commonly used for binary classification, but since we classify multiple categories of DDoS attacks in this paper, we conducted multinomial logistic regression, where we define the probability in each class

$$p(Y=y|X=x) = \frac{e^{\theta_y x^T}}{\sum_c e^{\theta_y x^T}} \tag{4}$$

and the log-likelihood function is

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} log(p(y^i|x^i))$$
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[\theta_{y^i}x^i - log\sum_{c} e^{\theta_c x^i}] \tag{5}$$

then we make prediction on input values by predicting the most probable class in decision function [11].

There are several optimization algorithms to minimize the loss function, including in stochastic gradient descent, mini-batch gradient descent, and Newton's method. In this paper, we conducted experiments with limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [7]. To reduce overfitting, we applied a L2 regularization on the model.

3.2 Random Forest

A random forest classifier consists an ensemble of large number of decision trees. Each decision tree splits the input data into subsets, based on a set of splitting rules based on classification features. Decision tree learning algorithm applies greedy algorithms to find the best split on all possible features and splits, it two common criterions includes Gini impurity and entropy. Gini impurity is defined as

$$I_G(n) = 1 - \sum_{i=1}^{c} p_i^2 \tag{6}$$

Entropy is defined as

$$I_H(n) = -\sum_{i=1}^{c} log(p_i) \tag{7}$$

where p_i is the probability of samples in class c in a node of the tree. The tree is constructed at the splits with highest information gain [22]. The experiment conducted on this paper made a comparison between these algorithms.

Training a random forest applies bagging or bootstrap aggregating, which randomly select subsets of data with replacement of the training set and fit trees to those samples. After training, predictions can be made by taking the majority prediction of all decision trees. This model results in better performance by decreasing variance of the model without increasing bias [3].

3.3 Artificial Neural Network

An artificial neural network (ANN) is an interconnected group of layers of nodes, example illustrated in figure 2, each node represents an "artificial neuron" [10]. Each neuron receives one or multiple weighted inputs, passes their sum into an activation function and produce an output. Let there be n inputs, with each input x_1 to x_n and a bias input $x_0 = 1$, and their weight w_0 to w_n, then the output of the neuron is

$$z = \sum_{i=1}^{n} x_i w_i; y = f(z) \tag{8}$$

where f is the activation function. Common activation functions in neural network includes sigmoid, softmax, ReLU, etc. The calculation from input layer through neurons to output layer is called forward propagation. An artificial neuron is illustrated in figure 3.

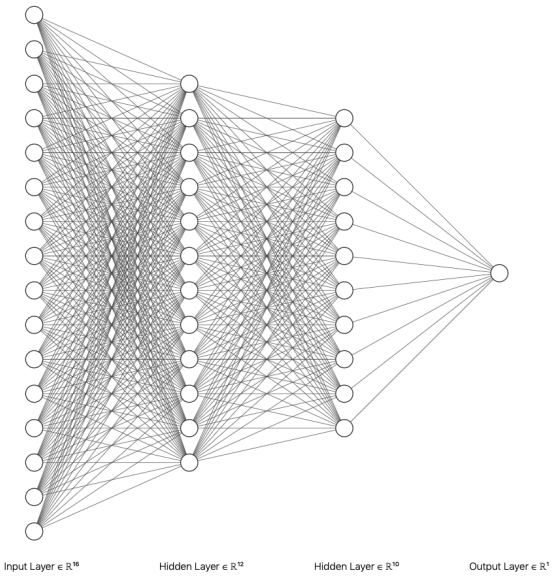

Figure 2: Neural Network

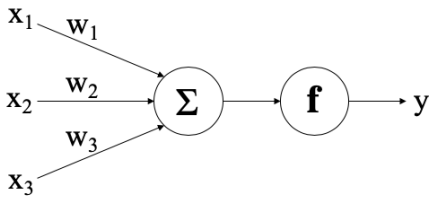

Figure 3: Artificial Neuron

The activation function we used in this papers are rectified linear unit (ReLU) and softmax. ReLU function is defined as

$$y = max(0, x) \tag{9}$$

Softmax function takes input of N-dimensional vector x and returns N-dimensional vector S, defined as

$$S_j = \frac{e^{x_j}}{\sum_{k=1}^{N} e^{x_k}} \tag{10}$$

Backpropagation (BP) algorithm is used in training neural networks, it uses loss function to propagate loss values backwards from output layer to input layer. We optimize parameters of neurons during backpropagation and iterate this process until convergence. In a single neuron, after we calculate z from the product of input x and weights w and pass through activation function $f(z)$ to get predicted value a, the loss function is

$$J(w, x, y) = \frac{1}{2}(y - a)^2 \tag{11}$$

To optimize w, we need to calculate gradient of J on w. By chain rule, we have

$$\frac{\partial}{\partial w} J = \frac{\partial J}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w} \tag{12}$$

Then we can optimize w using gradient descent algorithms and propagate to upper layers by calculating gradient of their parameters using chain rule.

3.4 Evaluation Metrics

We evaluate classifiers by metrics of accuracy, precision, recall, and F1 score. The latter three metrics evaluate performance of each class. Accuracy measures corrected predicted instances in a classifier:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

where TP represents true positives; TN represents true negatives; FP represents false positives; FN represents false negatives. Precision, also called positive predictive value, represents the ratio of actual positives along all the predicted positives:

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

Recall represents true positive rate, shows how many actual positives are predicted to be positive:

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

Label	Raw Data	Training set	Test set
BENIGN	113,828	70,205	29,795
DNS	5,071,011	70,136	29,864
LDAP	4,095,052	69,813	30,187
MSSQL	10,309,945	70,256	29,744
NTP	1,202,642	69,943	30,057
NetBIOS	7,750,776	69,651	30,349
Portmap	186,960	70,063	29,937
SNMP	5,159,870	70,005	29,995
SSDP	2,610,611	69,814	30,186
Syn	6,473,789	70,087	29,913
TFTP	20,082,580	70,001	29,999
UDP	7,001,800	70,047	29,953
UDPLag	368,334	69,979	30,021
WebDDoS	439	307	132

Table 1: Data Sampling and Splits

F1 score of a classifier is the mean of precision and recall.

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{16}$$

In intrusion attack classification problems, the classifier needs to avoid benign packets to predicted malicious, or attacking packets classified as other categories, so it needs to minimize false positives. Therefore, precision is the most importance metric of evaluation.

In multiclass classification, when evaluating overall performance of the entire classifier, there are different methods to calculate these metrics. Micro average evaluates models by making summation of all classes, it only calculates accuracy of models. Macro average calculate average of evaluation metrics of all classes. Weighted averages weighted metrics of all classes, based on the number of actual instances of each class.

4 Experiment

4.1 Data Preprocessing

CICDDoS2019 dataset [23] provides intrusion data in both PCAP and labeled CSV format. The experiment conducted in this paper used the dataset in CSV format, which contains 13 types of attacks and benign packets, illustrated in figure 1. In total, it has 14 labels with 70,427,637 entries in 88 columns, including in one column for labeling. However, as table 1 shows, because the raw dataset is too large and contains highly unbalanced entries of classes, we randomly selected 100,000 sample of each class, with all 439 WebDDoS entries.

After selecting data sample, we removed columns that are unrelated to model training, including in index, “Flow ID”, and “Timestamp”, then modified “SimilarHTTP” to be binary feature, where all entries that are not “0” in the original dataset are set to be “1”. For “Flow bytes” and “Flow Packets” feature that contains infinite values, we replace those values with a large value greater than the greatest value of the column in the original dataset. We also converted IP addresses in “Source IP” and “Destination IP” into binary format, then transformed them into decimal form. Finally, we encoded the non-numerical classes into numerical forms.

We split the sample dataset into training set and test set, with 7:3 ratio. The number of entries in each set is shown in table 1.

Because the features in this dataset is highly skewed, to improve training performance, we transform this dataset similar to Gaussian distribution, with mean μ of 0 and standard deviation σ of 1. Each feature is transformed with

$$x_i' = \frac{x_i - \mu}{\sigma} \tag{17}$$

We implemented this transformation with StandardScaler class of scikit-learn library [19] by first fitting the scaler with training set and then transform both training set and test set with this scaler.

4.2 Logistic Regression

We conducted experiment this multiclass classification problem in logistic regression with L-BFGS optimization algorithm [7] and L2 regularization. The performance of the classifier is affected by regularization strength. Figure 4 shows the relationship between regularization strength and accuracy of classifier in both training data and test data.

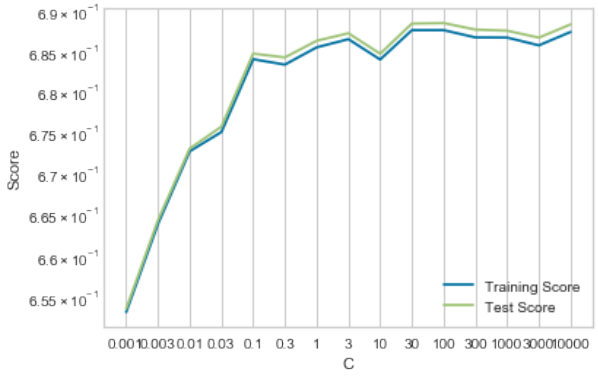

Figure 4: Logistic Regression Accuracy Over Regularization Strength

We applied "LogisticRegressionCV" class in scikit-learn library [19] to select the best regularization strength in the range from 1e-4 to 1e4. The result training accuracy is 0.6939 and the test accuracy is 0.6948.

4.3 Random Forest

The performance of random forest classifier in our experiment is affected by the number of decision trees and criterion of tree splits. Splitting trees by gini impurity performs better than splitting trees by entropy in out experiment. We set the number of trees to be 1000, split by gini impurity, the result training accuracy is 0.8576 and the test accuracy is 0.7930. Figure 5 shows the relationship between the number of trees and classification performance, by splitting trees by gini impurity.

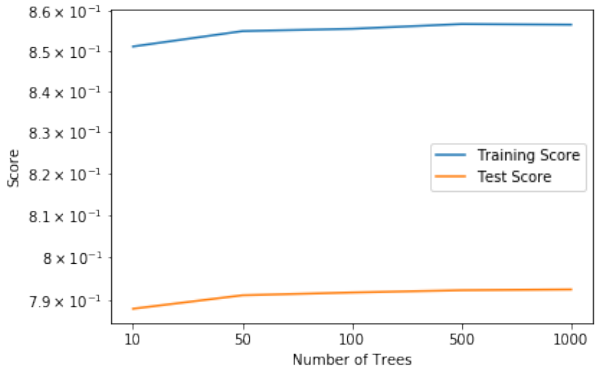

Figure 5: Random Forest Accuracy Over Number of Trees

4.4 Neural Network

We designed and build neural network classifier with TensorFlow Keras framework [17]. The structure of neural network contains data with 84 input features, then a 84-neuron fully connected layer with ReLU activation; a 0.2 dropout layer; a 128-neuron fully connected layer with ReLU activation; a 0.2 dropout layer, and a 14-class output layer with softmax activation function. The structure of the neural network is illustrated in figure 6.

We set batch size to be 1500 and run 100 epochs. The performance of neural network largely improved in first 10 epochs and improved smoothly after first 10 epochs. Training loss and accuracy over epochs are illustrated in figure 7, figure 8, respectively.

After 100 epochs of training, the training accuracy reached 0.7643 and test accuracy is 0.7661.

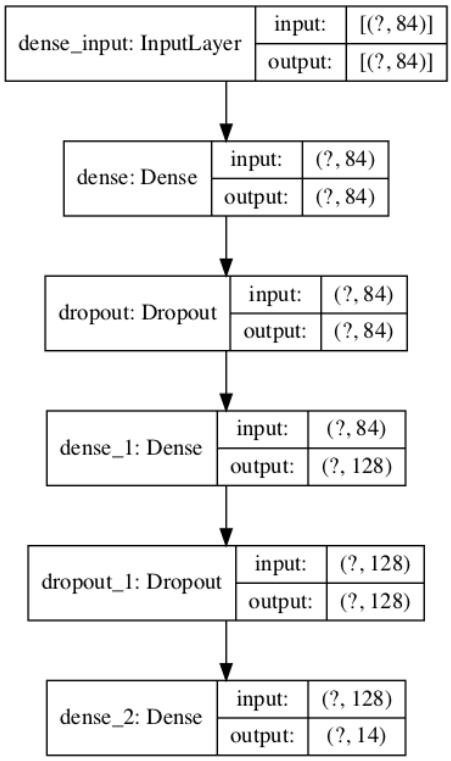

Figure 6: Neural Network Design

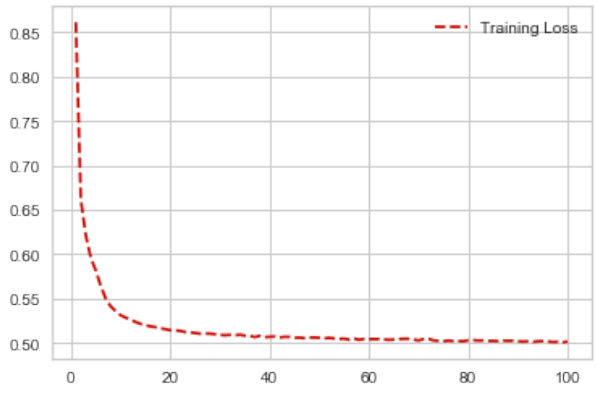

Figure 7: Training Loss Over Epochs

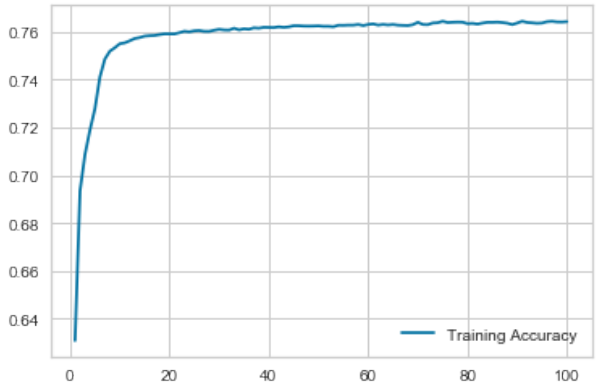

Figure 8: Training Accuracy Over Epochs

Metric	LR	RF	NN
Precision (Macro)	0.722	0.802	0.782
Precision (Weighted)	0.725	0.809	0.795
Recall (Macro)	0.658	0.788	0.781
Recall (Weighted)	0.695	0.793	0.766
F1 (Macro)	0.649	0.783	0.754
F1 (Weighted)	0.677	0.788	0.754

Table 2: Traffic Classification Results

5 Results

5.1 Classification

We tested three machine learning algorithms to classify network flows into 13 types of DDoS attacks and 1 benign class:

1. Logistic Regression with regularization (LR)
2. Random forest using Gini impurity scores (RF)
3. Neural Network (NN): 2-layer fully-connected feedforward neural network, trained for 100 epochs with batch size 1500 using sparse categorical cross entropy loss and Adam optimizer

We trained these three classifiers on a training set with balanced number of observations for each class and calculated the classification accuracy on a test set (Table 2).

The accuracy of the three classifiers ranged from 0.65 to 0.81. Logistic Regression had the worst performance, suggesting that the data may not have a clear decision boundary and is not linearly separable. The neural network performed well, reaching an accuracy of 0.795. It is expected to improve its performance if more amount of data could be provided.

Syn	Precision	F1
LR	0.99	0.88
RF	0.94	0.87

Table 3: Accuracy of LR and RF for Syn attack

The random forest performed the best, achieving an accuracy of 0.81. The confusion matrix of random forest classifier is shown in figure 9.

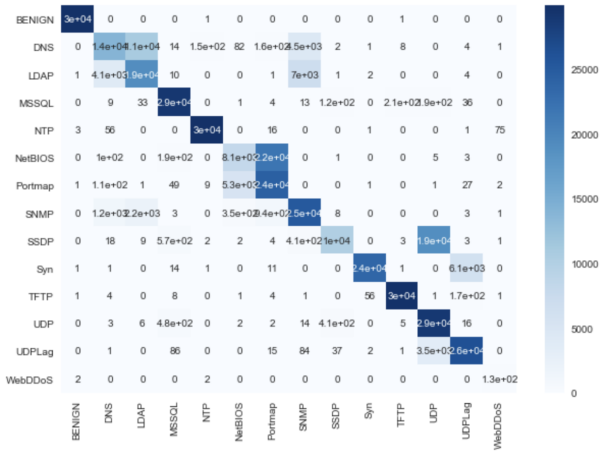

Figure 9: Confusion Matrix of the RF Classifier

Note that although logistic regression had the worst performance on the overall classification, it achieved higher accuracy than that of random forest when deciding whether a traffic is a Syn attack (Table 3).

5.2 Feature importance

The random forest also generated the feature importance based on the Gini impurity score. The results suggested that the source port and some properties of the packets sending through the traffic play significant roles in the DDoS attack multinomial classification, top 20 important features are illustrated in figure 10.

6 Conclusions

6.1 Conclusion

In this work, we showed that machine learning algorithms can effectively distinguish 14 categories of DDoS attack, including the benign traffic. Our dataset came from real-world network. We tested three different machine learning classifiers and random forest produced the best accuracy, which was 0.81. The neural network also achieved an accuracy of

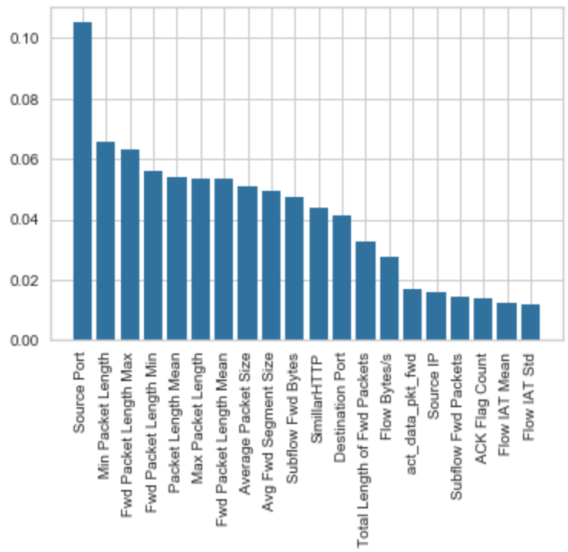

Figure 10: Feature Importance given by the Random Forest

0.79. These results demonstrate that it is feasible to employ machine learning for DDoS attack classification task.

6.2 Future Works

The results in this work motivates follow-up research to classify DDoS attack categories more accurately.

First, we would like to carefully look at the feature set and do the feature selection for different training models. We could utilize the feature importance generated by the random forest as well as calculate the ANOVA f-score of the features to filter out meaningful features. This could possibly improve the performance of our model.

We would also like to collect more data as it would increase the diversity of network traffic. We could use the larger dataset to prove whether the current classifier is generally useful. Increasing amount of data could also improve the performance of neural network due to the nature of NN's algorithm.

We would try to build more complex neural networks such as the convolutional neural network or apply some advanced network architecture such as the AlexNet. We would also train more classifiers such as the support vector machine (SVM) with nonlinear kernel.

References

[1] Muhammad Aamir and Syed Mustafa Ali Zaidi. "Clustering based semi-supervised machiine learning for DDoS attack classification". In: *Journal of King Saud University-Computer and Information Sciences* (2019).

[2] Muhammad Aamir and Syed Mustafa Ali Zaidi. "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation ". In: *International Journal of Information Security* 5 (2019), pp. 761–785.

[3] Leo Breiman. "Random Forests". In: *Machine Learning* (2001).

[4] CAIDA. *The CAIDA "DDoS Attack 2007" Dataset*. Aug. 2007. URL: https://www.caida.org/data/passive/ddos-20070804_dataset.xml. (accessed: 06.07.2020).

[5] Deloitte. *Defending against distributed denial of service (DDoS) attacks*. URL: https://www2.deloitte.com/ca/en/pages/risk/articles/DDoSattacks.html. (accessed: 06.06.2020).

[6] Shi Dong and MUDAR SAREM. "DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks". In: *Digital Object Identifier* (2019).

[7] Roger Fletcher. *Practical methods of optimization*. 2nd ed. Wiley, 1987.

[8] V. D. Gligor. "A note on denial-of-service in operating systems". In: *IEEE Transactions on software Engineering* 10.3 (1984), pp. 320–324.

[9] Changqing Gong, Tongyao Shi, Ming Mu, Liang Zhao, Abdullah Gani, and Han Qi. "An Improved Quantum Genetic Algorithms and Application for DDoS Attack Detection ". In: *2019 IEEE Intl Conf on (ISPA/BDCloud/-SocialCom/SustainCom)* (2019).

[10] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks*. The MIT Press, 1995.

[11] David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. 3rd ed. Wiley, July 2013.

[12] Meigen Huang and Bin Yu. "FuzzyGuard: A DDoS attack prevention extension in software-defined wireless sensor networks". In: *KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS* 7 (2019).

[13] Avinash Kadimisetty. *Gradient Descent — Demystified*. June 2018. URL: https://towardsdatascience.com/gradient-descent-demystified-bc30b26e432a. (accessed: 06.07.2020).

[14] *KDD Cup 1999*. Oct. 1999. URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. (accessed: 06.06.2020).

[15] Meejoung Kim. "Supervised learning-based DDoS attacks detection: Tuning ". In: *ETRI Journal* (2019).

[16] MIT Lincoln Laboratory. *2000 DARPA Intrusion Detection Scenario Specific Datasets*. July 2000. URL: https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets. (accessed: 06.07.2020).

[17] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[18] A10 Networks. *5 Most Famous DDoS Attacks*. URL: https://www.a10networks.com/blog/5-most-famous-ddos-attacks/. (accessed: 05.28.2020).

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] S.Suresh and N.S.Ram. "A review on various DPM trace back schemes to detect DDoS attacks". In: *Indian J. Sci. Technol* 47 (2016), pp. 1–8.

[21] Rajat Saxena. "DDoS attack prevention using collaborative approach for cloud computing". In: *Cluster computing* (2019).

[22] Shai Shalev-Shwartz and Shai Ben-David. "18. Decision Trees". In: *Understanding Machine Learning* (2014).

[23] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani. "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy". In: *IEEE 53rd International Carnahan Conference on Security Technology* (2019).

[24] Manjula Suresh and R. Anitha. “Evaluating Machine Learning Algorithms for Detecting DDoS Attacks”. In: *Communication in Computer and Infomation Science* (2011).

[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. “A detailed analysis of the KDD CUP 99 data set”. In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 2009, pp. 1–6.