Network Flow Analytics: Multi-Class Classification of DDoS Attacks Based on OKNN

Mungwarakarama Irenee[1]
Faculty of Computer Science and Engineering
Xi'an University of Technology
Xi'an, China
[1]E-mail: fezanava@stu.xaut.edu.cn

Xinhong Hei[*], Yichuan Wang, Wenjiang Ji, Xinyu Jiang
Faculty of Computer Science and Engineering
Xi'an University of Technology
Xi'an, China
[*]E-mail: heixinhong@xaut.edu.cn

***Abstract*—in the past decade DDoS attacks detection solutions have been one hot area in the research of cyberspace. Different techniques including machine learning and recent complex deep learning models were used and improved. However, only a few have used real network data and multiclass classification. In this paper, we challenged an Optimized K-Nearest Neighbor (OKNN) on a recent public dataset of the real network containing labeled classes of normal network flow and DDoS attacks. While performing minimum preprocessing to keep data original, OKNN with a tune of hyper-parameters such as; n_neighbors, metric, weights, n_jobs, has identified a normal traffic flow and DDoS attacks with high accuracy. The results of the experiment show that our model would perform better than its counterparts if only they are trained in the same conditions.**

Keywords- Network flow analytic; Optimized KNN; DDoS-based attacks; Multi-class classification

I. Introduction

Distributed denial-of-service (DDoS) attack is a malicious effort to interrupt normal traffic of a target server, service, or network infrastructure in a public or private cloud. The attacker who is either a person or a process sweeps over the target with a flood of Internet traffic disguised in different types of normal traffic. To achieve its goal and effectiveness, DDoS attacks exploit various compromised computer systems as sources of traffic attack. There are numerous types of devices exploited by DDoS ranging from computer servers, routers, firewalls, IoT devices, and other networked resources. If we consider the Internet as a highway from a bird's view, a DDoS attack is like a traffic jam preventing regular traffic from arriving at its intended endpoint. The DDoS attacker uses various sources to send enormous data to the target server, which will cause inaccessibility for legitimate users. "Distributed Denial-of-Service (DDoS) attacks have drawn extensive attention in cyberspace during the last few years" [16]. A reasonable work has been done in the intrusion detection and classification but as far as the authors are aware, only a few have tackled the classification of the normal data and DDoS-based attacks on real network datasets. Those who have contributed in this area have focused on the binary classification to see if traffic flow is normal or abnormal. A few of them of which we have compared with our work have used multiclass classification on small datasets, but they have used synthetic data, others have forgotten the fact that the real data is imbalanced and they tried to balance the data which can cause the model to over-fit when it is used in the real environment. The goal of this work is to use a simplistic but powerful machine learning model; OKNN to predict and classify the traffic flow using a highly imbalanced dataset of the real data collected from the real network environment with different DDoS based attacks [1]. The reason behind choosing OKNN is due to its simplicity but also it's less biased when it comes to classifying highly imbalanced and multiclass data. With the dataset of $\sim$ 216 million samples, OKNN has not only shown potentiality in terms of accuracy but also the speed to train and predict is good considering the amount of data it trained on with the tune of some key parameters. For all the research works we have consulted the course of this research though have contributed a lot in the same research area, nevertheless, they lack a sense of real-world scenario. Many of them have embarked on the recent complex deep learning models and incorporated them into classifying network traffic with synthetic and emulated data, therefore we wanted to prove that it doesn't always take a complex model to solve the matter. Our optimized model though looks simple, however, has shown success on a minimally preprocessed data, which is what awaits every model in real-life.

II. RELATED WORK

Wherever Network traffic analytics has been hot research in the area of network management and monitoring. Many authors have used different techniques trying to understand how emerging technologies maybe be able to identify, classify, and mitigate the degree of attacks in the network traffic for different purposes. We have acknowledged different authors' viewpoints on this matter. Most of the recent research [2] has leveraged the new centralized network device management strategies SDN and they have been focusing on the security based on the control plane. For this purpose, several methods based on information entropy, machine learning, and statistical analysis have been proposed by various researchers [2]. As the same authors also indicated, flow-based and packet-based are some of the classes of DDoS attack mechanisms and they are the two different methods to study and examine the network traffic flow among others. On one hand, during the study of flow-based traffic [4], the authors collected and used the flow statistics aggregates from the OpenFlow switches. On the other hand, the network packet features such as headers and payloads helped to learn some patterns in packet-based traffic. Here the network flow is defined as a combination of packet field values that make up a connection between two end devices. So while studying the packet-based traffic, it is promising to analyze the data deeply to differentiate the malicious traffic from the normal one.

The focusing technique of this research is limited to machine learning methods.

A. *Machine learning (ML)-based DDoS attacks detection methods*

The ML methods are fundamentally mathematical models relatively intelligent enough to learn by themselves on a large dataset to recognize hidden patterns and decide without explicit instructions. Many different machine learning models have been used extensively to classify, predict, and cluster the normal traffic and abnormal traffic, and they are more promising than signature-based detection solutions. [5] On synthetic data using Hping3 and Nping flow traffic generators extracted some flow characteristics and they compared the performance of KNN and SVM classifiers to classify the studied network traffic as malicious or benign. KNN predicted well on the dataset however, since the dataset was synthetic, it does not represent the real-life network where we have highly imbalanced traffic. [6] Proposed two new methods "K-means ++ and Fast K-Nearest Neighbors (K-FKNN)" for DDoS detection in SDN to improve classical KNN. However, even though in his study classical KNN is outperformed, the study does not show the prediction results of each attack, and the dataset is incomparable in size.

B. *Deep learning (DL)-based DDoS attacks detection methods*

Even though both classical machine learning and deep learning have been used in this research area, we see that deep learning due to its newness has been toping journals in recent years. A few of the most recent deep learning models are acknowledged. In their experiments [7] authors train backpropagation neural network (BPPNN) on synthetic data after extracting the time-based behaviors of an attack to learn the patterns which are later used to classify or detect the attack. In the research done by [8], the authors try deep leaning models CNN, LSTM, and RNN to identify DDoS attacks in the network. In their experiment, they use a trained DDoS detector module built with deep learning capability to analyze the data packets entering in the OpenFlow switch. [9] Used SOM on synthetic data to classify the network traffic as normal or attack. In the research done by [10], the entropy metric is used to calculate the flow attributes which are further used for the application of LSTM to predict the normal behavior signature of each attribute.

Advantages: Flexibility – able to train on a huge number of functional forms. Strength – it does not assume or (rarely assumes) the underlying function. Performance – the prediction capability for the model is high.

Disadvantages: More data – to approximate the mapping function requires a huge amount of data. Overfitting – more chances to overfit the training data some predictions are not easy to explain.

III. CLASSIFICATION MODEL

A. *OKNN and its mathematical characteristics*

Generic K-Nearest Neighbor (KNN) on which OKNN is built, "is an efficient lazy learning algorithm" [11] and has been successfully developed in many applications [16]. It is used for both classification and regression problems. Fig. 1, 2, and 3 respectively show how generic KNN learns patterns using a distance metric. The task here is to determine the position of a new data point whether it falls in the red category or the green category according to Fig. 1. In Fig. 2 we choose the number of neighbors i.e. *k*=5, we calculate the distance of 5 nearest neighbors and allocate the new data point to the corresponding class where the number of neighbor points counted is most presented. In our example it is red category, so the new data point is allocated to the red category. Fig. 3 shows that the new data point is allocated to the red category i.e. Category 1 because among the five nearest neighbors three of them are from the red category.

The process of predicting a new data point by KNN starts by searching for the *k* most similar data points known as neighbors into the training dataset with the help of distance measure which results in a variable holding the summary of *k* neighbors.

The classification classifier of a generic KNN is set to default parameters which many researchers have adopted as the best parameters. Only when these default parameters are tuned to reflect the problem, OKNN which is built based on KNN can outperform many powerful models with its simplicity. [12] Indicated that even though some previous researchers have tried to answer the parameter tuning questions of generic KNN, most of them have in this regard used a small number dataset with reasonable or nearly balanced classes as will be shown in Table IV. The most commonly used distance metric *Minkowski* with p set to two i.e. *Euclidean distance* and the default *weights* parameter set to *Uniform*. These parameters have only worked on the constrained problems and have performed well, but we are doubtful that they could do the same on a large and highly imbalanced dataset. A mathematical representation of Minkowski distance measures is explained in (1), (2), (3), and (4).

Minkowski distance measures: It is a family of three distance metrics that are special cases of Minkowski distance, consisting of different values of p denoting the power. "The Minkowski distance, which is also known as the L_p norm, is a generalized metric". It is defined as:

$$L_p(X,Y) = \sqrt[p]{\sum_{i=1}^{n} (|x_i - y_i|)^p} \qquad (1)$$

"Where p is a positive value. When p = 2, the distance becomes the Euclidean distance. When p = 1 it becomes Manhattan distance. x_i is i th value in the vector X and y_i is i th value in the vector Y" [18].

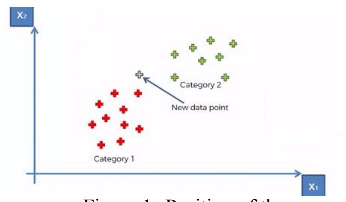

Figure 1. Position of the new point

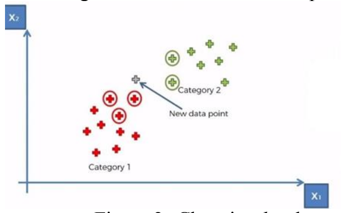

Figure 2. Choosing *k* value

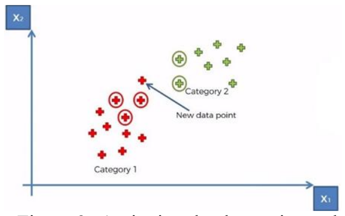

Figure 3. Assigning the data point to the class

Manhattan (MD): In the 19th-century a German scientist by the name of Hermann Minkowski studied the Manhattan distance, which is known as an L_1 norm, City block distance, Rectilinear distance, or Taxicab norm. It represents the sum of the absolute differences between the points in vectors.

$$L_1(X,Y) = \sum_{i=1}^{n} |x_i - y_i| \qquad (2)$$

Euclidean (L_2): This distance represents the root of the sum of the square of differences between the points in vectors. It is known as L_2 norm or Ruler distance, represented as the Pythagorean Theorem.

$$L_2(X,Y) = \sqrt{\sum_{i}^{n} (x_i + y_i)^2} \qquad (3)$$

In the literature according to [13] Euclidean distance function among others is known to measure the distance between points A and B in a feature space.

"Let A and B be represented by feature vectors $A = (x_1, x_2\ldots x_m)$ and $B = (y_1, y_2\ldots y_m)$, where m is the dimensionality of the feature space" [19]. To calculate the distance between A and B, the normalized Euclidean metric L_2 is used in (3) and (4).

$$L_2(A,B) = \sqrt{\frac{(x_1 - y_1)^2 + \cdots + (x_m - y_m)^2}{m}} \qquad (3)$$

$$L_2(A,B) = \sqrt{\frac{\sum_{i=1}^{m}(x_i - y_i)^2}{m}} \qquad (4)$$

Based on the properties of your data it is possible to select the best metric for your problem. In case it's unclear to you, you can try different distance metrics and some k different values all together to see a combination with the best outcome. A recommendation use of the Euclidean distance is when the input variable types are closely related such as all heights and widths, whereas if the input variable types are unrelated such as age, height, gender, etc Manhattan distance is recommended. The best way to find the best value of k is to use hyper parameter tuning on your algorithm.

The main purpose of OKNN in this research is to challenge a generic KNN classifier with different parameters tuned, on a large real network dataset and do a little work of data preprocessing to simulate the real-world scenario.

B. Description of the methodology

The following steps are required to implement the OKNN algorithm.

TABLE I. OKNN FRAMEWORK

Input		:	Training data
Output		:	Classified flows
	Step 1	:	For training data:
	1.1	-	Set the list to hold the error rate for the range of selected k range
	1.2	-	Set the range of k and tune other parameters
	1.3	-	Append the error values with corresponding k values to the list
	1.4	-	Plot the error rate with the respect of k
	Step 2	:	Select the best k value corresponding to 0 or close to 0 error rate
	Step 3	:	**For** each point P_t **in** test data:
	2.1	-	Compute distance D between test data and each row of training data
	2.2	-	Sort distance values D_v in ascending order
	2.3	-	Select top k rows with minimum distance
	2.4	-	Assign class C to the test point P_t based on the most frequent class of these rows. $C \leftarrow P_t$

IV. EXPERIMENTAL METHODS AND RESULTS

In this paper, we divide the experiment activity into three tasks: Data processing, model building, and evaluation.

A. Data processing

The dataset used in this experiment was collected from a large-scale network and was uploaded publically on the Kaggle platform for research purposes. The dataset contains 2,160,668 labeled "network logs from various types of network attacks" [1]. The types of network attack traffic flow logged together with the Normal network flows are: UDP-Flood, Smurf, SIDDOS, HTTP-FLOOD make 5 classes of an imbalanced distribution. Fig. 5 shows the class distribution in the dataset and in Tab. 1 we see the distribution percentage of classes.

We have tried to train the model on almost the features except a few of them that were specific flow-based such as; src_add, des_add, pkt_id, from_node, to_node, fid, node_name_from, node_name_to, which were not relevant in this context. Due to not having missing values in the dataset, we didn't do much work on the data pre-processing except encoding categorical values. We have chosen Label Encoder for its simplicity and dimensional reduction because after training the model on the Label Encoded dataset and One Hot Encoded dataset we have not noticed any reasonable difference.

Since the main objective of the model is to work in a real environment and handle the real data, we have trained the model with reality, i.e.; without data scaling or sampling.

B. Model building

The model used in the experiment is an Optimized K-Nearest Neighbor (OKNN). We have chosen this model for the reason discussed in section III of this paper. Moreover, it has proven to be powerful on this dataset with multiclass classification and imbalanced distribution of classes. Also, it does not take much time to train and test, even though the size of the dataset is big enough. We split the dataset into three subsets generally known as the training set, validation set, and test set with 60%, 20%, and 20% respectively.

C. Model evaluation and results

On Intel Core i7 9th Gen laptop, OKNN has trained on 1296400 samples with 19 features. The results that are shown in Fig. 6 indicate our final model trained on the testing dataset. The model has converged in what we consider a very short time of average ~ 10min compared to other models in the same category of non-parametric machine learning algorithms such as SVM on the same dataset. The later has trained more than 4 hours on the same split and it was stopped before completion because of taking a great amount of time.

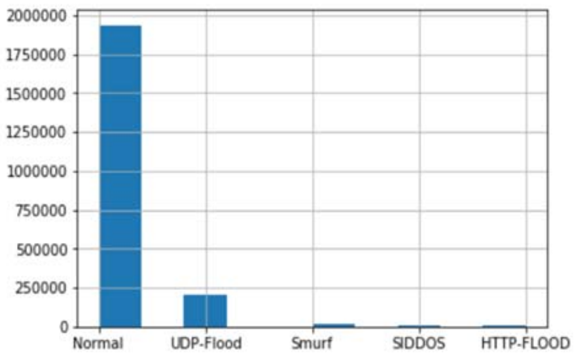

Figure 4. Class distribution in the dataset

The following parameters have been used: it is very hard to find the optimum *k* value due to the size of the dataset. Most researchers [14] use *k* = sqrt (N) as the optimum value of *k*, where N is the training sample size but this does not work well in all cases. In our experiment we computed *k* based on the error rate it produces as shown in Fig. 5 and 6, we observe the increase of *k* value leads to the decrease of the error rate. Initially, we computed the optimum *k* value in the range 1 – 100, but due to the limitation of our computing resources, it took more than a day with a failed memory. Therefore, we started computing in blocks of numbers ranges such as; 1-10, 20-25, 25-50, and 50-100. Since the higher number was promising we have randomly chosen numbers ranging from 100 to 490 since our memory could not train on k higher than 490 without failing.

Table 2 shows the results of *k* values and *time* (training + validation + testing). The *k* value which was used to predict the results in Fig. 6 was *k* =490, but according to the graph, the higher would be better, unfortunately, our memory would not handle it.

Due to the dissimilarity of the input variable types, we have used the distance metric*: Minkowski* with *p* = 1 (Manhattan). Weights are set to '*distance'* because it assigns weights proportional to the inverse of the distance from the query point unlike the default value '*form'* which assigns uniform weights to each neighbor which could be dangerous in our case of a highly imbalanced data. The number of jobs represented as parameter *n_job* is assigned the value -1 to tell the computer to dispatch the computation workload to all CPUs to achieve a short training time.

TABLE I. PERCENTAGE OF CLASS DISTRIBUTION

Flow class	Flows/class	Percentage
NORMAL	1,935,959	89.6%
UDP-FLOOD	201,344	9.3%
SMURF	12,590	0.6%
SIDDOS	6,665	0.3%
HTTP-FLOOD	4,110	0.2%

TABLE II. DDOS ATTACKS CLASSIFICATION RESULTS

	Precision	Recall	F1-score	Support
Http-Flood	98%	77%	86%	810
Normal	99%	100%	99%	3872050
Siddos	88%	96%	92%	1325
Smurf	67%	32%	43%	2567
Udp-Flood	97%	90%	93%	40182

TABLE III. THE VALUE OF *K* WITH *TIME*

K value	49	100	250	350	450	480	490
Time(s)	406	414	555	700	618	695	669

Machine learning models experience poor performance usually when they go through overfitting or underfitting. Overfitting is more likely to happen with non-parametric models that have more flexibility when learning a target function. We cannot rely on the general accuracy score metric since it can be misleading especially when we are dealing with a multiclass and imbalanced dataset. To limit overfitting while training machine learning model two handful of techniques can be used: 1. Evaluate the model accuracy with resampling technique 2. Detain a validation dataset

To achieve this, a well-known most popular resampling technique k-fold cross-validation is a regularly used technique in applied machine learning. However, this technique is only efficient if the dataset is small. We have used the second technique and it shows a great achievement. Besides, we have also evaluated the model using the confusion matrix metric for its efficiency on multiclass imbalanced data. As it is shown in Fig. 4 let us try to understand the metric used by the confusion matrix.

D. Model evaluation metrics

When dealing with a highly imbalanced dataset Precision-recall is the best evaluation metric. In (4) Precision *P* denotes that the number of true positives T_p is inversely proportional to the addition of true positives with false positives F_p.

$$P = \frac{T_p}{T_p + F_p} \tag{4}$$

Recall *R* denotes that the number of True Positives T_p is inversely proportional to the addition of true positives with false negatives F_n.

$$R = \frac{T_p}{T_p + F_n} \tag{5}$$

Both two measures are related to the F_1 score in (6), representing the harmonic mean of precision and recall.

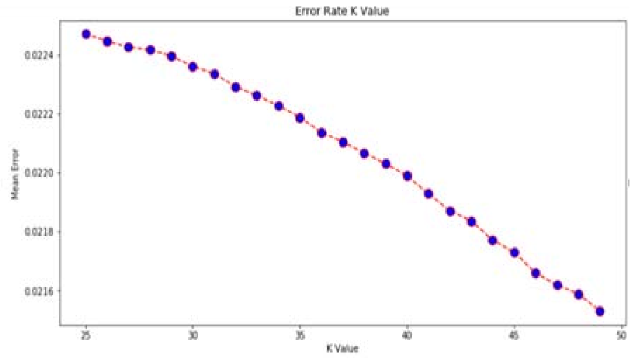

Figure 5. *k* ranges from 25 to 50

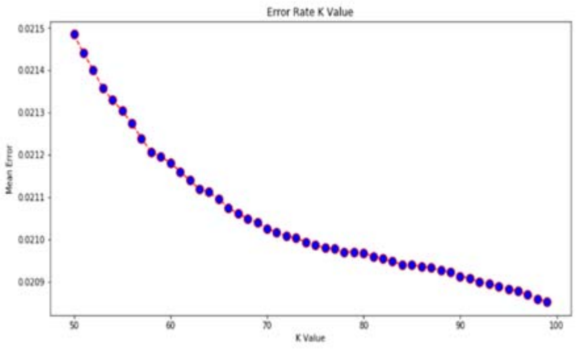

Figure 6. *k* ranges from 50 to 100

$$F1 = 2\frac{P * R}{P + R} \tag{6}$$

In the study of information retrieval, the term precision means the measurement of how relevant the results are returned, while how many true results returned is measured by the recall. A combination of both high results indicates that the model is more accurate with positive results. The problem comes when the model is returning many results but most of them are incorrectly predicted when compared to the training data labels, this model is worse than the one with few results, but most of them are correctly predicted when compared to the training data labels. These alternating precision-recall variations can be seen in our results however when you look at the F1 Score we see that our model is doing well in predicting NORMAL traffic, UDP-FLOOD, SIDDOS, and HTTP-FLOOD attacks with 99$^{\%}$, 93$^{\%}$, 92$^{\%}$, and 86$^{\%}$ respectively. This is because the F1 score seeks to find a balance between Precision and Recall when there is an uneven class distribution i.e; a large number of Actual Negatives as in our case. We can see that this measure shows the overall performance of the model hence it is recommended as the model performance evaluation metric. The reason why the model has poorly predicted SMURF attack with 43$^{\%}$ is yet to be understood.

In a comparison with other authors who conducted researches close in similarity with this one, we have only picked the two recent and compared them with ours as shown in Tab. IV. It is hard to compare them since the researches are being conducted in different conditions, for instance, [11] conducted their experiment on a small simulated network topology with one server and ten clients including one DDoS user, it does not produce enough data. Even though his results look well and promising for KNN and improved KNN (DDAML), in their conclusion they look forward to improving their algorithm on the real network environment which is what we have done and the results are promising.

E. Comparison of OKNN with other models

Other researchers like [6] used what they called Fast KNN (KFKNN) to detect whether a flow is normal or a DDoS attack. Their results are better than ours however, the original public dataset they claim to have used is huge and rich [15]

TABLE IV. RESULTS COMPARISON

Model	DDAML[11]	KNN[11]	KFKNN[6]	OKNN
Data distrib-ution	70%, 20%, 10%	70%, 20%, 10%	50%, 50%	89.6%, 9.3%, 0.6%, 0.3%, 0.2%
Sample size	1400	1400	20000	2160668
Classes	3	3	2	5
Preci-sion Avg.	99.30%	98.30%	97%	89.80%
Recall Avg.	99.40%	98.50%	98.30%	94.60%
F-mea-sure Avg.	99.35%	98.40%	97.70%	82.60%

but surprisingly the authors have decided to use a small amount of data with sampling. This does not reflect the real-world environment like what we have done, which cannot be a proof that this model could perform better than ours.

To conclude our comparison, all the models discussed above have done well in the conditions of which they have been constrained, but when compared with our model in the same condition, we can confidently say that our model would outperform them inaccurate prediction.

V. CONCLUSION

There has been tremendous growth in using neural networks and so many other sophisticated techniques to detect DDoS attacks in recent years, moreover, most DDoS attack-based datasets used were synthetic or small in size and mostly binary classification. In this paper, we have shown that OKNN though simple in structure, with hyper-tuned parameters, is a powerful model. It has efficiently predicted different types of DDoS attacks on an imbalanced multi-class dataset. Nevertheless, the reason why it could not predict the SMURF attack so well can lead to danger in the real life. This issue is considered as the next part of the research.

ACKNOWLEDGMENT

This research work is supposed by the National Key R& D Program of China(2018YFB1201500), National Natural Science Funds of China (62072368, 61773313, 61702411), National Natural Science Funds of Shaanxi (2017JQ6020, 2016JQ6041), Key Research and Development Program of Shaanxi Province (2020GY-039, 2017ZDXMGY-098, 2019TD-014).

REFERENCES

[1] Steyn, J. V. (2020, April 7). DDoS Attack Network Logs. Retrieved from Kaggle:https://www.kaggle.com/jacobvs/ddos-attack-network-logs.

[2] Mininder, S. P., & Abhinav, B. (2020). New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research. Computer Communications.

[3] Pratibha, K., Naminath, H., & Bodhisatwa, M. (2020). Efficient Keyword Matching for Deep Packet Inspection based Network Traffic Classification. 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS). Bengaluru: IEEE.

[4] Liu, Y., Gao, H., Guo, L., Qin, A., Cai, C., & You, Z. (2020). A Data-Flow Oriented Deep Ensemble Learning Method for Real-Time Surface Defect Inspection. IEEE Transactions on Instrumentation and Measurement.

[5] SUN, G., JIANG, W., GU, Y., REN, D., & LI, H. (2018). DDoS Attacks and Flash Event Detection Based on Flow Characteristics in SDN. IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).

[6] XU, Y., SUN, H., XIANG, F., & SUN, Z. (2019). Efficient DDoS Detection Based on K-FKNN in Software Defined Networks. IEEE ACCESS.

[7] Cui, J., He, J., Xu, Y., & Zhong, H. (2018). TDDAD: Time-Based Detection and Defense Scheme Against DDoS Attack on SDN Controller. *Australasian Conference on Information Security and Privacy*. Springer.

[8] Li, C., Wu, Y., Yuan, X., Sun, Z., Wang, W., Li, X., & Gong, L. (2018). Detection and defense of DDoS attack – based on deep learning in OpenFlow – based SDN. International Journal of Communication Systems.

[9] Nam, T. M., Phong, P. H., Khoa, T. D., Huong, T. T., Nam, P. N., Thanh, N. H., . . . Loi, V. D. (2018). Self-organizing map-based approaches in DDoS flooding detection using SDN. *International Conference on Information Networking (ICOIN).*

[10] Matheus, N. P., Luiz, F. C., Jaime, L., & Mario, P. L. (2020). Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment. *IEEE Access*.

[11] Dong, S., & Sarem, M. (2019). DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. *IEEE Access*.

[12] Haneen ARrafat, A., Ahmad, H. B., Omar, L., Ahmad, T. S., Mahmoud, B., Hamzeh, E. S., & Surya, P. V. (2019). Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. arXiv.

[13] Hu, L. Y., Huang, M. W., Ke, S. W., & Tsai, C. F. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *Springer Plus*.

[14] *K Nearest Neighbor*. (2016, 1). Retrieved from ScienceDirect: https://www.sciencedirect.com/topics/immunology-and-microbiology/k-nearest-neighbor.

[15] Cybersecurity, C. I. (2020, July 5). NSL-KDD dataset. Retrieved from the University of New Brunswick.

[16] HAWLADER. (2020, September 16). *The Application of K Nearest Neighbor Algorithm In Real Life*. Retrieved from Foss Guru Web Site: https://www.fossguru.com/the-application-of-k-nearest-neighbor-algorithm-in-real life/#Application_of_K_Nearest_Neighbor_Algorithm

[17] SHI, D., & MUDAR, S. (2020). DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. IEEE Access.

[18] V. B. Surya, P., Haneen, A. A., Ahmad, B. A., Omar, L., Ahmad, S. T., Mohamoud, B. A., & Hamzeh, S. E. (2019). Effects of Distance Measure Choice on KNN Classifier Performance - A Review. *Mary Ann Liebert*

[19] Magda, B., Ding, X., Wang, H., David H., G., Wang, H., Girijesh, P., . . . Initiative, F. t. (2019). A practical computerized decision support system for predicting the severity of Alzheimer's disease of an individual. Expert Systems with Applications.