Network Anomaly Detection using Deep Learning

Adelola Adebo
MSc. Data Analytics
School of Computing
National College of Ireland
Dublin, Ireland
x19134711@student.ncirl.ie

***Abstract*—Every year, the number of cyber-attacks is increasing, causing huge financial and reputational damage to both small and large enterprises. Intrusion detection systems are one of the methods used to solve this issue. In practice, anomaly-based intrusion detection systems have been widely used to detect and protect against network threats. There are, however, some weaknesses, such as high false positive rates, which means that it is prone to intrusion alarm when, in fact, no intrusion has occurred, placing a major burden on security administrators. In this research, we propose a deep learning approach, the Deep Neural Network, to develop a more sophisticated intrusion detection system to address the limitation of the traditional intrusion detection system. Accuracy, Precision, Recall and F-Measure metrics were used to assess the efficiency of the proposed model. The results of the experiment showed that the proposed model was able to achieve high detection accuracy of 98% while at the same time reducing false positives. In the end, this study is expected to strengthen intrusion detection systems to defend companies from cyber-attacks.**

***Index Terms*—machine learning, deep learning, intrusion detection system, anomaly detection, cybersecurity**

I. Introduction

Cybersecurity risks are rising with increased global connectivity and widespread use of networked computer systems. According to the 2017 Accenture Cost of Cyber Crime study, successful cyber-attack breaches have increased by 27% on average from 13% annually across organizations, costing an average of US$11.7 million to cyber-security threats alone. One such attack was the famous WannaCry ransomware attack on the Taiwan Semiconductor Manufacturing Company, which spread over 10,000 of its computers, forcing the company to shut down some of its factories in 2018. Another example is the attack on Equifax, the largest credit bureau in the U.S., which has put some 145 million people's personal information at risk. Equifax's shares dropped 13% the day after the breach, and the breach brought multiple lawsuits against them. Not to mention the loss of reputation of Equifax [1]. Cyber-attacks are increasing in scale, effect and severity. Organizations are, therefore turning to machine learning to improve their malware detection systems, and defend their companies from cyber attacks.

Defending cybersecurity attacks requires a suite of sophisticated tools to allow network administrators to build defences against such attacks. Intrusion Detection Systems (IDS) is one such tool used to identify malicious network activities launched by outside attackers that compromise security in the form of integrity, confidentiality or availability. Traditional IDS approaches use either signature-based or anomaly-based IDS to detect attacks. Signature-based defines a set of network attacks already known before and then searches for attacks for which the technique has already been established. As a result, new attacks are often missed, as the method only knows the behaviour of the attacks defined earlier [2].

On the other hand, the anomaly-based IDS builds a user behaviour model that is considered normal and then detects any novel attacks that deviate from normal behaviour. The shortcoming, however, is that they are susceptible to high false alarms. For example, a sudden increase in network usage, which has not been previously considered, may have been labelled abnormal, even if it is legitimate. This increases the pressure on security analysts and may lead to negligence in the event of severe, harmful attacks [3]. Although anomaly-based IDS produces false alarms, it's potential to detect both known and unknown attacks has led to widespread acceptance in the research community [4].

To address the weakness of the conventional IDS, researchers have begun to focus on developing IDS using machine learning methods. Machine learning IDSs have been successful in reducing false alarms while still achieving a high level of malware detection accuracy. Deep learning, a subset of machine learning can deliver outstanding performance compared to traditional machine learning methods, mainly when dealing with the big data nature of network traffic. Besides, deep learning methods can automatically learn valuable features from a raw data set and perform malware classification simultaneously [2] [3]. This study aims to explore how deep learning can be used to detect network anomalies in network traffic to enable organizations to strengthen their malware detection systems in the fight against cyber-attacks.

The rest of this paper is structured as follows: Section 2 describes related work on machine learning approaches to the IDS. Section 3 details the experimental procedure used to achieve the objective of this study. Section 4 discusses the findings and Section 5 concludes this paper.

II. Related Work

A. Machine learning for network anomaly detection

From a machine learning standpoint, network anomaly detection is a classification problem. Several machine learning methods such as k-Nearest Neighbour (KNN), Support Vector Machine (SVM), Naïve Bayes, Decision trees, k-Means clustering have been widely used. For example, [5] used kNN and k-means clustering algorithms to identify and detect network anomalies on the Coburg Intrusion Detection Data Sets (CIDDS-001). The algorithms have been used to classify and cluster network instances into five classes; normal, attacker, victim, suspicious and unknown. The results of the research showed that the models performed reasonably good, achieving up to 99% accuracy in the classification of attack groups.

To overcome the limitation of high false alarm rates in anomaly detection systems, researchers have proposed a hybrid solution to address this shortcoming. [6] proposed a combination of k-Means clustering and Naïve Bayes classifier to reduce false alarm rates while achieving high detection accuracy. The authors evaluated their approach on the 1999 KDD Cup data set. k-Means clustering was used to cluster the data into normal and abnormal behaviour, and then Naïve Bayes was used to classify the clustered data into attack groups such as DoS, U2R, Probe, R2L and Normal. The accuracy score reached 99% while decreasing false alarm rates by 0.5%.

A similar approach was proposed in [7], combining SVM and K-Medoids clustering (similar to the k-Means algorithm). The experiments were conducted on the Kyoto+2006 data set and compared with the previous k-Means with Naïve Bayes approach. The results of the analysis showed that the SVM with K-Medoids outperformed the Naïve Bayes with k-Means approach, where SVM with K-Medoids achieved a 4% improvement in accuracy and detection rate and a 1% reduction in false alarm rates.

While the results of the works mentioned above have shown a significant improvement in detection accuracy, while at the same time decreasing false alarm rates, the data sets used are obsolete and do not reflect today's complex and changing network environments [8].

Ye et al. carried out a comparative study of machine learning methods in terms of malware detection systems. They argued that the performance of a good malware detection system critically depends not only on the classification/clustering algorithms but also on the selection of features. A feature selection technique is used to filter out redundant and irrelevant features from a data set. As a result, this reduces computational complexity and makes it easier to interpret and improve the accuracy of the detection algorithm [9].

Mehmood and Rais, introduced a novel SVM method, ASVM, to the anomaly-based IDS. The ASVM uses a variation of the ant colony optimization method to initially remove redundant and irrelevant features from the dataset before fitting the data into the SVM model. Although the results of the approach using the 1999 KDD Cup dataset, have shown an improvement in the accuracy of classification with the extraction of features, the dataset used is still out of date [10].

In [11], the author applied Random Forest Regressor to find the best subset of features from the more recent CIC IDS 2017 dataset, which closely simulates real-world network traffic. The authors analysed the performance of these features with classifiers such as Naïve Bayes, MLP, Random Forest (RF), KNN, Iterative Dichotomiser 3 (ID3), Adaboost and Quadratic Discriminant Analysis (QDA). ID3 and RF had the highest precision score of 98%. However, the approach used was limited to a subset of features for each attack family. Our proposed IDS aims for the combined detection of all attack groups.

B. Deep learning for network anomaly detection

Recently deep learning, a broader field of machine learning motivated from their successful performance in other data-intensive applications such as image processing, natural language processing and self-driving cars has been applied to network anomaly detection systems. Studies have shown that deep learning surpasses the traditional machine learning models as the scale of data increases. Besides, it can automatically learn valuable features from a raw data set and detect malware simultaneously.

Althubiti et al. proposed a deep learning approach to detecting anomalies using Long-Short-Term Memory (LSTM). The approach was evaluated on the CIDDS-001 dataset and compared with classifiers such as SVM, Naïve Bayes and Multilayer Perceptron. The results of the experiments show that both LSTM and MLP achieved higher accuracy and precision than other traditional machine learning models [12]. [13] used a combination of stacked autoencoders and Convolutional Neural Networks (CNN) to automatically learn essential features from large scale unlabelled raw traffic data and at the same time detect network anomalies. [4] proposed a deep neural network to build an IDS to secure an Industrial control system with up to 99.9% accuracy.

In summary, the literature's reviewed have shown impressive high-performance results using deep learning approach. Therefore, we propose a deep learning approach, called a deep neural network, to develop the network intrusion detection model.

III. Experiments

To create a comprehensive analysis, we designed and implemented two deep neural network (DNN) models. A baseline model where pre-processing and class re-sampling is done only before training the neural network with the data. In the second model, feature selection is performed along with class re-sampling. Fig. 1 shows the process flow from the data collection to the evaluation of the approach.

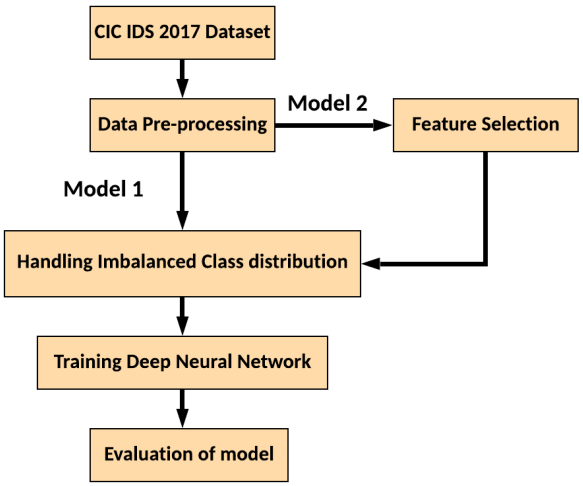

Fig. 1. Methodology flow diagram

A. Data Collection

The Canadian Institute of Cyber Security (CIC) IDS 2017 [11] is used as a data set for our experiment. The data set simulates a realistic network traffic environment and includes the most up-to-date attacks. It captures all requirements for a comprehensive IDS data set, namely labelled data, latest attacks, a variety of attacks, the capture of complete bi-network flow, feature set and metadata [8]. Data records eight files and attacks during five days of daily traffic. We combined the eight CSV files into one file, which becomes a total number of 2, 830, 743 records, 79 features and 15 attack types labelled. Benign traffic accounts for about 80% of the data, while the malicious traffic together accounts for about 20%, creating a class imbalance issue for machine learning models. High-class imbalance causes the machine learning model to be biased towards the majority of the class. In the following sections, we will address the problem of class imbalance. Table I highlights the types and distribution of attacks.

TABLE I
CIC IDS 2017 DATASET ATTACK TYPES AND CLASS DISTRIBUTION

Traffic type	Class distribution
Benign	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack Brute Force	1507
Web Attack XSS	652
Infiltration	36
Web Attack Sql Injection	21
Heartbleed	11

TABLE II
CLASS SAMPLES AFTER BALANCING

Traffic type	Class distribution
Benign	250000
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	5000
Web Attack Brute Force	5000
Web Attack XSS	5000
Infiltration	5000
Web Attack Sql Injection	5000
Heartbleed	5000

B. Data Pre-processing

Data pre-processing is the stage at which we convert raw data to a clean format that will be suitable for further analysis. As we conducted an exploratory study of the data, we found 1358 records with missing values and 3018 records with null values which were removed from the data set. Furthermore, eight features have been found to have zero values, which are *Bwd PSH Flags*, *Bwd URG Flags*, *Fwd Avg Bytes/Bulk*, *Fwd Avg Packets/Bulk*, *Fwd Avg Bulk Rate*, *Bwd Avg Bytes/Bulk*, *Bwd Avg Packets/Bulk*, *Bwd Avg Bulk Rate*. This means that these features have no impact on our analysis. Therefore, they were excluded from the data set. The data set is now made up of 2827876 observations, 70 features and class labels.

C. Handling Class Imbalance

To address the issue of class imbalance, we used two techniques. The first technique, the Benign (normal) traffic class, is down-sampled using the RandomUnderSampler Library in python from 2,273,097 to 250,000 records. Down-sampling is done so that the large volume of the Benign class does not dominate the other classes rendering the learning algorithm bias [14]. For the second technique, the Bot, Web Attack Brute Force, Web Attack XSS, Infiltration, Web Attack SQL Injection, and Heartbleed minority classes were up-sampled to 5000 records using SMOTE (Synthetic Minority Oversampling Technique). Up-sampling is done to increase the sensitivity of the minority class to the learning algorithm [14]. Table IV shows the class distribution after applying the class balancing.

After class balancing, feature scaling is performed on the data before the deep neural network is trained.

D. Deep Neural Network without Feature Selection

Keras, a neural network library in Python, is used for the DNN implementation. The trained DNN used an input layer with seventy neurons corresponding to the number of features in the data set, followed by two hidden layers with Relu activation function and an output layer with 15 neurons using the Softmax activation function. The 15 output neurons

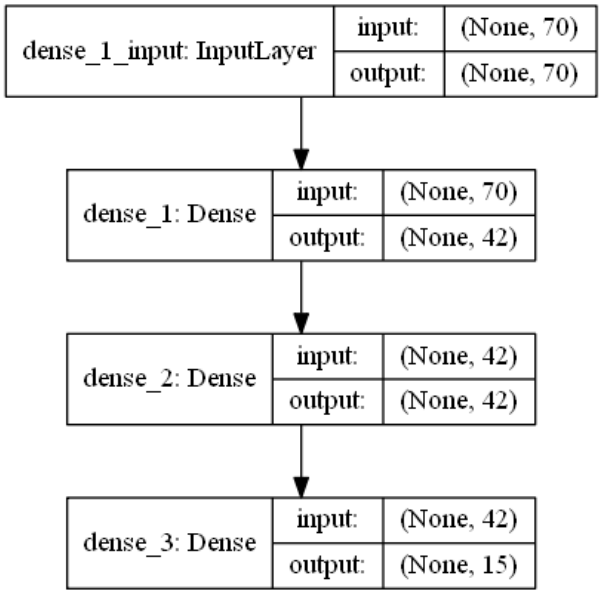

Fig. 2. Structure of the DNN model

correspond to the 15 attack classes in the dataset. Model hyper-parameter settings are: epochs: 100, loss function: "categorical_crossentropy", optimizer: "adam", batch_size: 128. After training the model, we access the performance using k-fold cross validation. Figure 2 shows the structure of the DNN model.

E. Deep Neural Network with Feature Selection

In the second model, after initial pre-processing, we find the best feature set for detecting attack class from 79 features, we used RandomForestClassifier of the scikit-learn library [14]. First, we calculate the importance of each feature in the entire data set, where the algorithm sets the value between 0 and 1. Then we set the threshold of any feature that is less than 0.01 is removed from the data set, which leaves us with 34 features. Next, we applied class balancing and trained the DNN model. The same hyper-parameter setting used in the first model, as mentioned above, is replicated but with 34 input neurons. After training the model, we access the performance using k-fold cross validation. (see Fig. 5 for feature importance).

F. Performance Evaluation

To evaluate the performance of the proposed models, we adopt evaluation metrics such as Confusion matrix, Accuracy, Precision, Recall, ROC-AUC and F1-score, which are commonly used by researchers as considered in the literature review. They are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

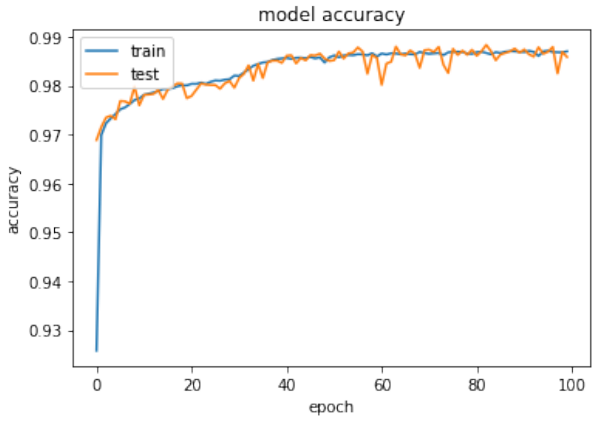

Fig. 3. Model accuracy during training

RESULTS AND DISCUSSION

The findings of the performance evaluation of the models are summarized in Table 3. The results show that the two DNN models achieved high performance in classifying attack groups. The DNN model without feature selection slightly performed better in terms of accuracy 0.985, precision value, 0.98 and F1-score, 0.84 than the DNN model with feature selection. This supports our argument that deep learning is very robust and may not require feature engineering, in particular, when there's a large data set, which relieves the burden of hand-crafted features.

We also looked at the performance of the DNN model during training to ensure that the model was not overfitting. Fig. 3 and Fig. 4 shows the accuracy and loss plots. From the accuracy plot, we can see that the DNN model performed well on both training and test sets. From the loss plot, we also see the loss declining as the number of epochs increases.

It is noted from the confusion matrix for each model, that the models had high numbers of DoS GoldenEye and Heartbleed attack groups misclassified. This was anticipated since the total number of the observations from the original class distribution was 21 and 11, respectively. A better balancing technique should be studied to resolve this in the future. (see Fig. 6 and Fig. 7).

Overall our proposed DNN model has a high detection rate with low false positives.

TABLE III
PERFORMANCE EVALUATION

Metrics	DNN without feature selection	DNN with feature selection
Accuracy	0.9858	0.9811
Precision	0.98	0.96
Recall	0.80	0.81
F1-score	0.84	0.83
AUC score	0.90	0.91

CONCLUSION AND FUTURE WORK

Having a secure network is one of the most critical concerns for businesses. In this study, we proposed a deep learning

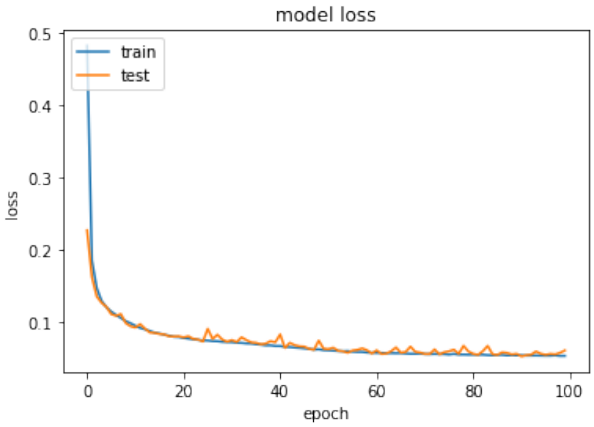

Fig. 4. Model loss during training

approach to develop a network anomaly detection system to classify network attacks into 15 attack groups. The experiment was carried out using the CIC IDS 2017 dataset. The results show that the deep neural network model achieved high performance at classifying attack groups while at the same time decreasing false positive rates. We also noted that the model was weak in classifying attacks with low instances. To address this issue in the future, we propose that a better class balancing technique be investigated. Finally, we believe that this work can help businesses develop sophisticated malware detection systems to protect their networks from future threats better.

REFERENCES

[1] Accenture. (2017) 2017 cost of cyber crime study. [Online]. Available: https://www.accenture.com/gb-en/insight-cost-of-cybercrime-2017

[2] M. Zamani, "Machine learning techniques for intrusion detection," *CoRR*, vol. abs/1312.2177, 2013. [Online]. Available: http://arxiv.org/abs/1312.2177

[3] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[4] A. Hijazi and J.-M. Flaus, "A deep learning approach for intrusion detection system in industry network," 02 2019.

[5] A. Verma and V. Ranga, "Statistical analysis of cidds-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Computer Science*, vol. 125, pp. 709–716, 12 2017.

[6] Z. Muda, W. Mohamed, m. n. Sulaiman, and N. Udzir, "K-means clustering and naive bayes classification for intrusion detection," *Journal of IT in Asia*, vol. 4, pp. 13–25, 04 2016.

[7] R. Chitrakar and H. Chuanhe, "Anomaly detection using support vector machine classification with k-medoids clustering," in *2012 Third Asian Himalayas International Conference on Internet*, 2012, pp. 1–5.

[8] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *CoRR*, vol. abs/1903.02460, 2019. [Online]. Available: http://arxiv.org/abs/1903.02460

[9] Y. Ye, T. Li, D. Adjeroh, and S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, pp. 1–40, 06 2017.

[10] T. Mehmood and H. B. M. Rais, "Svm for network anomaly detection using aco feature subset," in *2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC)*, 2015, pp. 121–126.

[11] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 01 2018, pp. 108–116.

[12] S. A. Althubiti, E. M. Jones, and K. Roy, "Lstm for anomaly-based network intrusion detection," in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, 2018, pp. 1–3.

[13] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Security and Communication Networks*, vol. 2017, pp. 1–10, 11 2017.

[14] M. H. Abdulraheem and N. B. Ibraheem, "A detailed analysis of new intrusion detection dataset," vol. 97, no. 17, 2019, pp. 4519–4537.

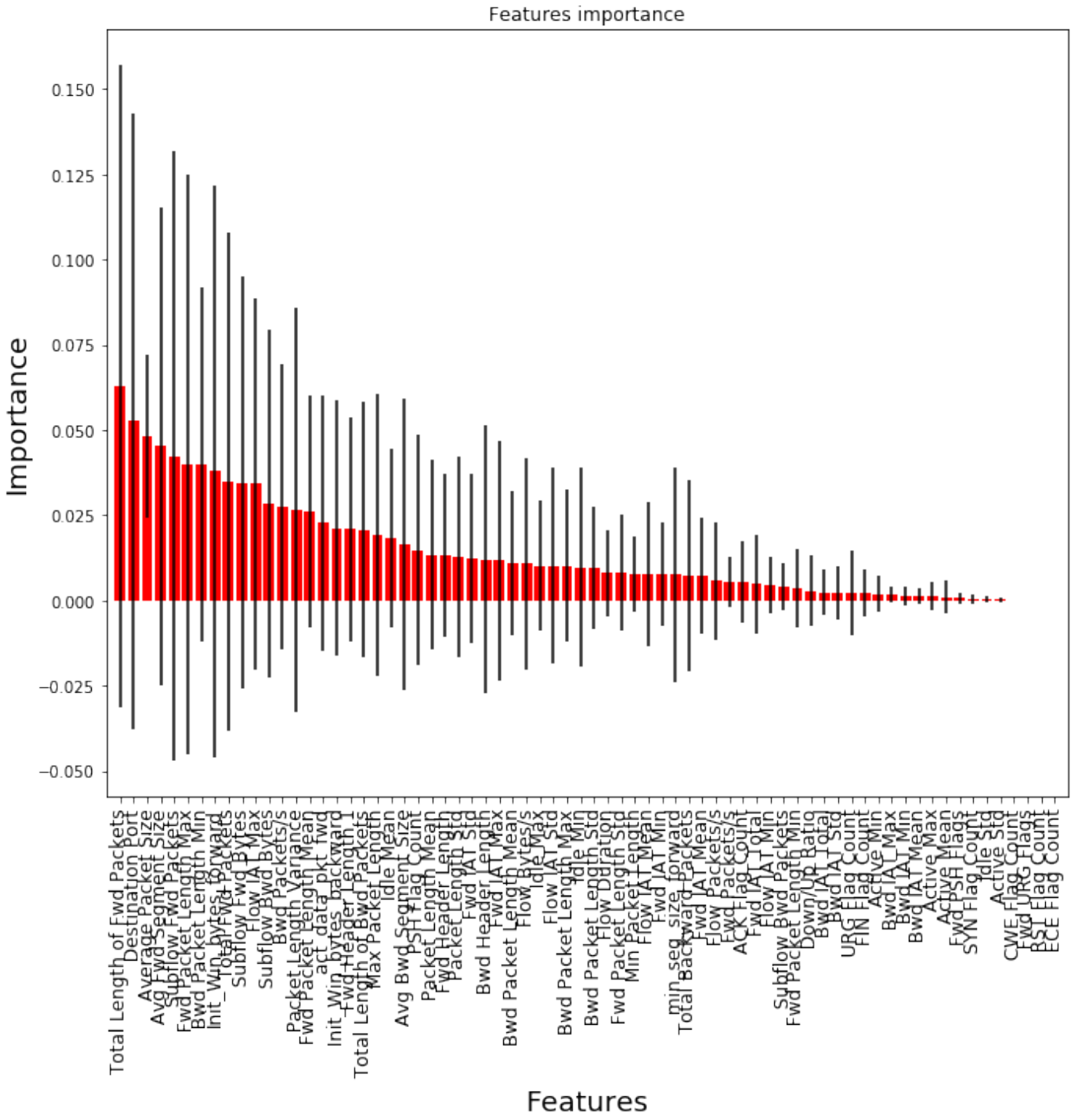

Fig. 5. Feature importance

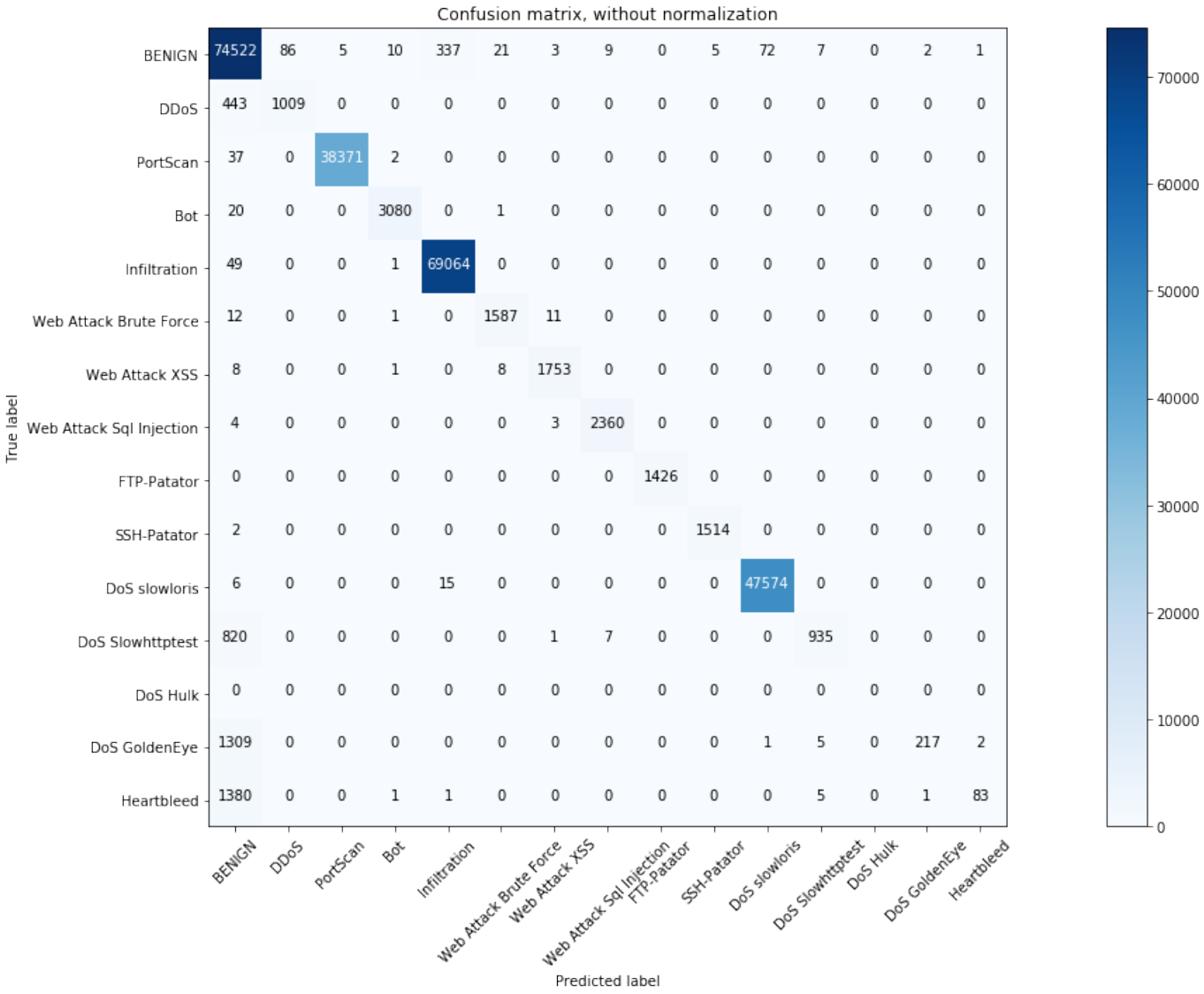

Fig. 6. Confusion matrix for DNN without feature selection

TABLE IV
FEATURES IN THE CIC IDS 2017 DATASET

Destination Port	Flow Packets/s	Packet Length Mean	Subflow Bwd Bytes
Flow Duration	Flow IAT Mean	Packet Length Std	Init_Win_bytes_forward
Total Fwd Packets	Flow IAT Std	Packet Length Variance	Init_Win_bytes_backward
Total Backward Packets	Flow IAT Max	FIN Flag Count	act_data_pkt_fwd
Total Length of Fwd Packets	Flow IAT Min	SYN Flag Count	min_seg_size_forward
Total Length of Bwd Packets	Fwd IAT Total	RST Flag Count	Active Mean
Fwd Packet Length Max	Fwd IAT Mean	PSH Flag Count	Active Std
Fwd Packet Length Min	Fwd IAT Std	ACK Flag Count	Active Max
Fwd Packet Length Mean	Fwd IAT Max	URG Flag Count	Active Min
Fwd Packet Length Std	Fwd IAT Min	CWE Flag Count	Idle Mean
Bwd Packet Length Max	Bwd IAT Total	ECE Flag Count	Idle Std
Bwd Packet Length Min	Bwd IAT Mean	Down/Up Ratio	Idle Max
Bwd Packet Length Mean	Bwd IAT Std	Average Packet Size	Idle Min
Bwd Packet Length Std	Bwd IAT Max	Avg Fwd Segment Size	Label
Flow Bytes/s	Bwd IAT Min	Avg Bwd Segment Size	
Fwd PSH Flags	Fwd Packets/s	Fwd Header Length.1	
Fwd URG Flags	Bwd Packets/s	Subflow Fwd Packets	
Fwd Header Length	Min Packet Length	Subflow Fwd Bytes	
Bwd Header Length	Max Packet Length	Subflow Bwd Packets	

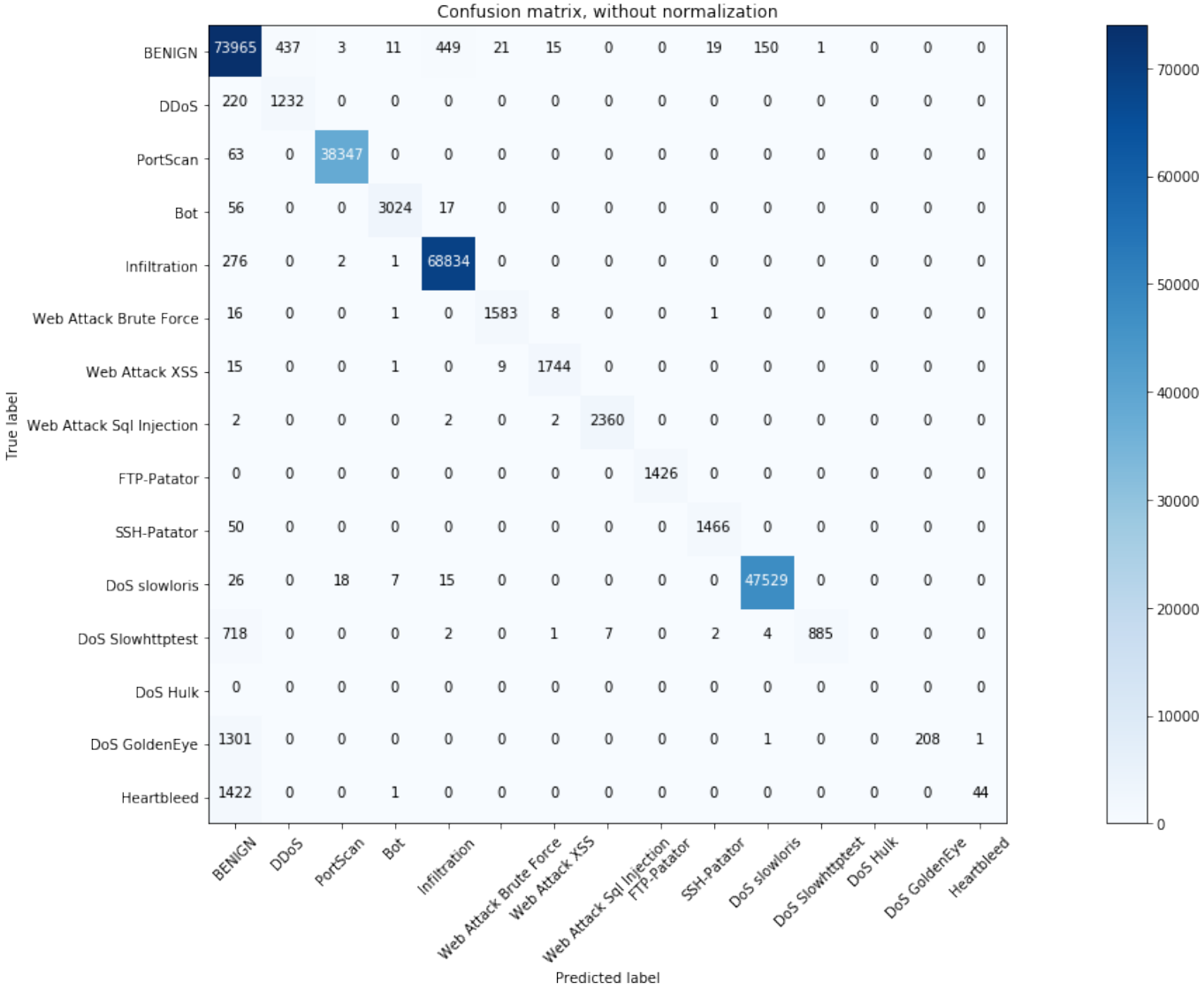

Fig. 7. Confusion matrix with DNN with feature selection