APSIPA Transactions on Signal and Information Processing, 2023, 12, e11

Original Paper

Malicious Network Traffic Detection for DNS over HTTPS using Machine Learning Algorithms

Lionel F. Gonzalez Casanova and Po-Chiang Lin*

Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan

ABSTRACT

Machine learning is an effective analysis tool to tackle the challenges to detect any suspicious events in the network traffic flow. In this paper, our major contribution is to process and transform the CIRA-CIC-DoHBrw-2020-time series dataset to train deep learning models for network intrusion detection. The main focus of our detection algorithms is to classify the data in a two-layer network approach. At the first layer, we classify DNS over HTTPS (DoH) and non-DoH traffic, and at the second layer, we characterize benign-DoH and malicious-DoH. We use 26 features out of the 34 features describing every pattern of network traffic. We use the DoH predictions in the first layer and pass it to the second layer for characterization of benign or malicious DoH. We then feed data to a fully connected neural network and four types of Recurrent Neural Networks. They are the Long Short-Term Memory, Bidirectional Long Short-Term Memory, Gated Recurrent Unit, and Deep Recurrent Neural Network. The proposed methods are simple and efficient, so that they can be applied to computer systems with limited resources. The generated models are small, so that they can be easily and quickly deployed into the internet network environment.

*Corresponding author: Po-Chiang Lin, pclin@saturn.yzu.edu.tw. This work was supported in part by Ministry of Science and Technology (MOST), Taipei, Taiwan, R.O.C. under grant number MOST 110-2221-E-155-002.

Received 30 August 2022; Revised 18 November 2022

ISSN 2048-7703; DOI 10.1561/116.00000058

Keywords: Network attack, Anomaly detection, Machine learning, Recurrent neural network.

1 Introduction

Intrusion detection aims to identify the anomaly access or attacks in network environments. This research field has been approached by applying machine learning (ML) techniques and has grown exponentially in recent years. Needless to say, the internet has become intrinsic in our daily lives. It helps individuals in a myriad of areas, such as business, entertainment, education, health care, to name a few. We can definitely find internet applications such as websites and email systems at various establishments of business operations. With that said, information security of the internet is of paramount priority and importance. Recently we have been hearing of cyber-attacks to key business areas in America and other wealthy nations. Not to exclude ordinary people or households that use internet services. People privacy are also at risk of attacks. All these attacks fall under the umbrella of the vulnerabilities in network security.

To counter these network attacks under the internet environment, there are various systems designed to block them. In particular, intrusion detection systems (IDS) help the network to counterattack external attacks [7]. According to [2], intrusion detection is based on the assumption that the behavior of intruders is different from a legal user. This, in turn, provides us with two categories derived from IDSs. They are anomaly and misuse (signature) detection based on their detection approaches [18].

In this paper, we clean and process the first ever Canadian Internet Registration (CIRA)-Canadian Institute for Cybersecurity (CIC) DNS over HTTPS (DoH) University of Brunswick (Brw) 2020 (CIRA-CIC-DoHBrw-2020) dataset. It's a two-layered approach that is used to capture benign and malicious DoH traffic along with non-DoH traffic. We train five deep learning models. In so doing, we investigate what algorithm works best for the CIC-DoHBrw-2020 dataset. Our dataset is well balanced for our model's best performance because reviews in machine learning research state that many researchers have overlooked this important step. We use techniques to robustly evaluate our models and use algorithms to evaluate the dataset. We use the resampling technique for splitting the data into training and testing sets to improve the robustness of the result. This is the holdout validation method that uses the Train-Test split function. We also implement well performing techniques such as hyper parameter tuning to acquire optimal parameters for a better performing solution. Our contribution in this research is two-fold. First, we propose a different approach on how to classify DoH and Non-DoH traffic in the first layer, and Benign-DoH and Malicious-DoH in the second layer. In the latter network layer, we create a dataframe that is the same length as

our predicted data values $(1, 0)$. Then we extract the DoH predicted values (0 s representing DoH) and merge it with the Benign-DoH and Malicious-DoH datasets. A detail to bear in mind is that the DoH dataset is the combination of the Benign-DoH and Malicious-DoH datasets.

The second contribution is to apply deep learning neural networks such as a Fully-Connected Convolutional Neural Network (FCN) and four different types of Recurrent Neural Networks (RNNs). FCN and its variant architectures have significantly performed well in comparison to the traditional machine learning classifiers. In regards to the RNNs, these models have limitations whereby they fail to capture the context as time steps increases. Knowing these limitations, we use other improved types of the RNNs. We use the Long-Short Term Memory (LSTM) networks that comprise of memory cells that can store information about previous time steps. The second type is the Gate Recurrent Unit (GRUs) that uses a set of gates to control the flow of information, instead of separate memory cells. In the bidirectional LSTM (third type of the RNNs), instead of training a single model, we introduce two. The first model learns the sequence of the input provided, and the second model learns the reverse of the sequence. The other type of the RNNs is the deep recurrent neural network (deepRNN). Here we stack the RNNs on top of each other. The deepRNN functions by given a sequence of length T, the first RNN produces a sequence of outputs, also of length T. These, in turn, constitute the inputs to the next RNN layer. In a nutshell, the FCN and LSTM neural network-based algorithms for anomaly detection have been investigated and reported to produce significant performance gains over conventional methods.

The rest of this paper is organized as follows. In Section 2 we describe the related work in the literature. The problem description is presented in Section 3. The proposed method is described in Section 4, followed by the performance evaluation results and discussions in Section 5. Finally, conclusions are presented in Section 6.

2 Related Work

As prior definitions lay the groundwork for our study of network anomaly detection, Fernandes Jr. *et al.* [8] categorize network anomalies under two relevant properties. They discussed and dealt with the property that how the network anomalies are characterized. There are three categories; however, the "point anomalies" category is looked at more closely. It is considered the simplest category by researchers. Ultimately, no matter what is the underlying nature of an anomaly, the bottom line is identifying these anomalies in a network helps to prevent malicious attacks from occurring.

In retrospect, the research on anomaly detection methods has seen significant advancements [3, 5, 8, 10, 12, 13]. Powerful computers and processors

have also contributed to a wider audience of researchers to undergo data analysis. To secure or bring an alternative to maintaining order in the domain of network security came about DNS over HTTPS (DoH) [17]. A number of web browsers have invested and implemented DoH support in their application technological structure. However, despite all this effort and determination to stop network intruders, there is a significant security risk of DoH that is related to the decreased visibility for the security tools and applications. Many researchers are being motivated to analyze DoH traffic. Bagnall *et al.* [1] announced the first occurrence of malware that intentionally uses DoH to hide its communication with Command and Control servers. Haddon et al. described possible ways of data exfiltration using DoH, which is more difficult to detect using current tools [9]. [6] studied and analyzed DoH encrypted traffic. They implemented and experimented with five machine learning models: Naïve Bayes, K-Nearest Neighbors, Random Forest, C4.5 Decision Tree, and Ada-Boosted Decision Tree. Moreover, they stated that the mentioned algorithms are commonly used in Network applications.

Banadaki [2] studied a systematic two-layer approach for detecting DNS over HTTPs (DoH) traffic and distinguishing Benign-DoH traffic from Malicious-DoH traffic using a number of machine learning algorithms. The author evaluated the DoHBrw-2020 dataset using the Decision tree, Extra trees, Gradient Boosting, Random Forest, Light Gradient Boosting Machine (LGBM) and XGBoost algorithms considering their accuracy, precision, recall, F-score, confusion matrices, ROC curves, and feature importance. Two algorithms outperformed the other four machine learning algorithms. LGBM and XGBoost algorithms show the maximum accuracy of 100% in the classification tasks of layers 1 and 2. The author explains that LGBM algorithms misclassified one DoH traffic test as non-DoH out of 4000 test datasets. In addition, out of 34 features extracted from the CIRA-CIC-DoHBrw-2020 dataset, Source IP is the critical feature for classifying DoH traffic from non-DoH traffic in layer one followed by the DestinationIP feature. Interestingly, the feature DestinationIP is an important feature for LGBM and gradient boosting when classifying Benign-DoH from Malicious-DoH traffic in layer 2.

According to [13], computer networks have fallen easy prey to cyberattacks in the fast-moving internet services. Recently, the Domain Name System (DNS) has been targeted with malicious intent such as cybercrime, data theft or the like. The new protocol DNS over Encrypted HyperText Transfer Protocol (HTTP) traffic over Secure Socket Layer (SSL), known as HTTPS, has succeeded to prevent DNS attacks, significantly. Hence, the cybersecurity community has introduced the concept of DNS over HTTPS (DoH) to improve user privacy and security by combating eavesdropping and DNS data manipulation on the way to prevent Man-in-the-Middle attacks. The authors studied covert channels by tunneling data through DNS packets. They identify tunneling activities that utilize DNS communications over HTTPS

by presenting a two-layered approach to detect and characterize DoH traffic using time-series classifiers. Their classifiers such as Random Forest (RF) and C4.5 produced equivalent classification results with equal precision, recall and f-score value. It is followed by the Support Vector Machine (SVM) and Naïve Bayes (NB) at 0.877 and 0.84 precision, recall and f-score value, respectively. The researchers also studied two deep learning models such as the LSTM and two-dimensional (2D) CNN via classification by statistical features of the flows with 0.97 and 0.98 precision, respectively. They introduced packet clumps and clump segments to find patterns in a limited window of traffic which in turn reduces detection latency. In this regard, they use the LSTM via classification by time-series features of the flows with precision hikes above 0.99 after six clumps in layer 1 and three clumps at layer 2. They argue that such precision values are comparable with most of the accurate statistical classifiers.

Other literature emphasizes the urgent outcry of keeping vulnerable network protocols secured from various security gaps that have exploited repeatedly over several years. DNS abuse is one of the most challenging threats for cybersecurity specialists [11]. To counterattack threats from attackers using complicated methodologies to inject malicious software in DNS inquiries is a challenging task. As a consequence, many researchers have explored different machine learning (ML) techniques to encounter this challenge. Jafar *et al.* [11] introduces a systematic approach identifying malicious and encrypted DNS queries by examining the network traffic and deriving statistical characteristics. The authors implemented several ML methods such as Random Forest (RF), Decision Tree Classifier (DT), Gaussian Naïve Bayes (GNB), K-nearest neighbor (KNN), Logistic Regression (LR), Support Vector Classifier (SVM) and Quadratic Discriminant Analysis (QDA). The CIRA-CIC-DoHbrw-2020 dataset is used to evaluate their ability to detect malicious DNS traffic. The results report that the machine learning models, RF, SVM, DT, and KNN have an accuracy of almost 99.9%. SVM and KNN are the slowest machine learning models in the training phase whereas, GNB is the fastest one yet has the worst results in the detection phase.

Another important research study sought the solution of implementing an encrypted DNS, called DNS-over-HTTPS (DoH) to counter measure the problem of privacy issues in networks. DoH guarantees privacy and security to prevent various attacks such as eavesdropping and manipulating DNS data by using the HTTPS protocol to encrypt the data between DoH client and DoH-based DNS resolver [14]. The authors emphasize once again that DoH is one of the best security options for an enterprise network where more sensitive data protection is required. Despite this, DoH may cause an unintended security breach, that is, information leakage via malicious DoH tunneling. There exist some limitations in DoH that have been addressed previously. The authors argue that collection and labeling data in this area is an impossible task while the data processing to feed to the Supervised Machine Learning methods

rely heavily on human-engineered feature extraction which makes classifying encrypted DoH traffic difficult. The authors explain there is no complete functional DoH detection application to network infrastructure. They propose a detection system for DoH tunneling attacks based on Transformer to detect a malicious DoH tunneling and build a fully functional DoH detection system that can be integrated with the security operation system of an enterprise network.

Upon using the Transformer architecture as a classifier method to detect malicious DOH, the authors emphasize that its more complex than other Supervised Machine Learning models. Their results show a significant improvement in the number of labeled data used. The accuracy achieved is 0.994 by using a small number of labeled data, only 20% compared with existing research. This advantage of the Transformer architecture makes it more suitable in malicious DoH tunneling detection because, in practice, labeling a large amount of encrypted network traffic is very complex and requires a lot of resources [14].

This research reiterates the importance of Internet security and the reality that the Domain Name System is under constant attack and daily its vulnerability increases. The authors remind readers that the DNS is the ideal target for most cyberattacks. There is no robust solution to this pressing issue; however, DNS over HTTPS as well as DNS over TLS are introduced to reduce the visibility of DNS requests. DNS over HTTPS has been designed to mitigate the DNS security issues but it has its own drawbacks like bypassing the local firewalls [15]. The authors present a Machine Learning approach to detect DNS over HTTPS traffic and to filter it into Benign-DNS over HTTPS traffic and Malicious-DNS over HTTPS traffic using ensemble machine learning algorithms. These are the Decision Tree, Logistic Regression, k-nearest neighbor, and Random Forest. Several evaluation metrics are considered to analyze the performance such as Precision, Recall, F1-score and confusion matrix. The CIRA-CIC-DoHBrw-2020 dataset is used for analysis against the machine learning algorithms. An ensemble learning-based RF classifier emerges as the best-suited model with 100% accuracy. The k-nearest neighbor and Decision Tree classifiers perform well, too.

Vekshin, Hynek, and Cejka explained that the new protocol DNS over HTTPS (DoH) have been engineered to improve users' privacy on the Internet [16]. DoH is used instead of traditional DNS for domain translation with encryption. This paper focuses on the possibilities of encrypted traffic analysis, especially on the accurate recognition of DoH. The authors aim to evaluate what information (if any) is gained from HTTPs extended IP flow data using Machine learning. They evaluated five popular ML methods to find the best DoH classifier.

Chalapathy and Chawla [4] surveyed deep learning for anomaly detection and provide key insights into convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The authors found that the CNNs ability to extract complicated hidden features from high dimensional data with

complex structure has enabled its use as feature extractors in outlier detection for both sequential and image datasets. Recurrent Neural Networks are shown to capture features of time sequence data. The reviewers explain the RNNs limitations whereby these models fail to capture the context as time steps increases. However, LSTM networks serve as an effective way to reduce these drawbacks. RNNS comprise of memory cell that can store information about previous time steps. The Gate Recurrent Unit (GRUs) are similar to LSTMs, the only difference is that it uses a set of gates to control the flow of information, instead of separate memory cells. In a nutshell, LSTM neural network-based algorithms for anomaly detection have been investigated and reported to produce significant performance gains over conventional methods.

3 Problem Description

Many researchers and data scientists are using anomaly detection algorithms to mitigate harmful attacks against internet service providers and the like. A recurrent enhanced neural network (RNN) is described in different contexts and at different levels of abstraction. For example, it can be said that an RNN is any neural network containing one or more recurrent (or cyclic) connections. People always tend to use RNN for time series data. However, we argue that a simple network architecture would be suitable for the application of the DNS over HTTPS problem. In addition, we train a number of deep learning models on the CIC-DoHBrw-2020 dataset, especially considering the computer systems with limited resources.

When working with time series data we need to understand that it is a series of data points ordered in time. There is a dimension which means it adds an explicit order dependence between observations. In a normal machine learning dataset, the dataset is a collection of observations treated equally when predicting the future. Furthermore, the order of observations provides a source of additional information that should be studied and used in the prediction process. Our data is a time series network traffic records captured by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick. It's the first-ever released DNS over HTTPS (DoH) dataset funded by CIRA's Community Investment Program. Time series data is quite challenging and difficult to train. The class imbalance problem is also another crucial aspect to train a model. It can also affect the model's performance.

4 Proposed Method

In this section, we first describe the data preprocessing, followed by the design of the machine learning models.

4.1 Data Preprocessing

First, we show the observations of the CIC-DoHBrw-2020 dataset. The nature of the CIC-DoHBrw-2020 dataset results from the implementation of DoH protocol within an application using five different browsers and tools and four servers to capture Benign-DoH, Malicious-DoH, and non-DoH traffic. Non-DoH traffic is generated by accessing a website handled by HTTPS protocol and its labeled as non-DoH traffic whereas benign-DoH traffic uses the same technique by browsing the web with Mozilla Firefox and Google Chrome. DNS tunneling tools such as dns2tcp, DNSCat2 and Iodine are used to generate malicious-DoH traffic. The scenario here is that such tools create tunnels of encrypted data to send TCP traffic encapsulated in DNS queries using TLS-encrypted HTTPS requests to special DoH servers. Four csv files are provided as the dataset in this paper. There are a total of 1,167,136 data instances in the dataset. The dataset is highly imbalanced. Table 1 shows the count of each label in the dataset. Most of the data instances are "nonDoH". The anomalies only occupy 3.59% of the dataset. Readers are referred to the CIC-DoHBrw-2020 dataset to find the feature names, types, and descriptions of the dataset.

Table 1: Count of each label.

Label name	Count
nonDoH	897,493
DoH	269,643
Benign	19,801
Malicious	249,836

4.1.1 Data Processing Pipeline

The data processing pipeline consists of the following crucial parts: (1) Feature Selection: to select appropriate features without fitting to some specific environments or time period. (2) Missing Data Handling: to use the fillna() mode method for columns separately. (3) Train-Test Split: to split the original dataset into two parts, including the training set and test set. In this paper, we split 20% of the dataset as the test set, and take the other 80% as the training set. (4) Data Imbalance Handling: to balance the numbers of data instances of different labels. (5) One Hot Encoding: to convert categorical features to numerical features in order to facilitate machine learning methods. (6) Feature Scaling: to scale and shift the feature values to some ranges that are suitable for machine learning methods. In this paper, we use the min-max scaling to transform the features by scaling each feature to $[0, 1]$. Detailed descriptions of the critical parts of the data processing pipeline, including

the feature selection and the data imbalance handling, are provided in the following subsections.

4.1.2 Feature Selection

Among all features in the CIC-DoHBrw-2020 dataset, we argue that the first four features are environment-specific. Different network environments would certainly have different IP addresses and port numbers. Moreover, it is very easy for network attackers to modify IP addresses in network packets. On the other hand, the time stamp feature and the duration feature are time-specific. A model trained by using all the 34 features would not be generalized to other network environments. Therefore, we drop the first six features in this work. Note that by dropping these features it would lead to performance degradation, since the test set to evaluate the model performance also comes from the original dataset. A more accurate model is required to compensate the performance degradation caused by the dropping of the features.

4.1.3 Handling Outliers

We observe the outliers in the data by using the visualization technique such as box plots. Outliers are points that are outside of the minimum and maximum values. We use the Interquartile range to measure the limits of the outliers because the data doesn't follow a Normal Distribution or it's either right-skewed or left skewed. The formula for the outlier boundary is calculated using the following:

- Interquartile range (IQR) = Q3 (75th percentile) – Q1 (25th percentile)
- Lower boundary = First Quartile (Q1/25th percentile) – (1.5*IQR)
- Upper boundary = Third Quartile (Q3/75th percentile) + (1.5*IQR)

In this case, all the values smaller than the Lower boundary is assigned to the value of the Lower boundary. The values greater than the Upper boundary is assigned to the value of the Upper boundary.

4.1.4 Handling Multicollinearity

After finding the correlation among the features in the dataset, we found out that some features were highly correlated. Thus, we did the Multicollinearity handling of the dataset. Multicollinearity affects the learning of neural networks, since the dependent variable is very less compared to the other variables, the neural network will take more time to converge. Some literature mentions that neural networks don't suffer from multicollinearity because they tend to be over-parameterized. Regardless of this, we applied the variance

indicator factor (VIF) to see the results obtained by the traditional machine learning models and deep learning models. As result, we dropped two more features such as "PacketTimeMode" and "PacketTimeTimeStandardDeviation", because of NaN values in the former feature and very high VIF values in the latter feature. We saw the VIF values in the other features lower significantly. Consequently, we chose to use 26 features to build our models.

4.1.5 Handling Data Imbalance

As Table 1 shows, the CIC-DoHBrw-2020 dataset is highly imbalanced. We use the resampling technique to deal with the data imbalance problem. The resampling technique consists of removing instances from the majority class (under-sampling) and adding more instances to the minority class (over-sampling). The data imbalance handling is performed after the train-test split mentioned above. In the second layer, the DoH set and non-DoH set are resampled to 161,796 data instances. In the second layer, the Benign set and Malicious set are both resampled to 3,269 data instances.

4.1.6 Feature Scaling

As part of the data pre-processing implementation in our research, we use one of the most common methods to feed the data to our machine learning models. This is the feature scaling method such as Min/Max scaling. In our case, we use normalization which is the rescaling of the data from the original range so that all values are within the range of 0 and 1. This method is applied to most of the statistical features except those time-series features such as SourceIP, DestinationIP, SourcePort, DestinationPort, TimeStamp, and Duration. The statistical traffic features are FlowBytesSent, PacketBytesReceived, PacketLengthVariance, PacketTimeVariance, to name a few. It's important to note that normalization can be useful, and even required in machine learning when time series data has input values with differing scales.

4.2 Deep Learning-Based Classification Models for DoH Traffic

In this paper, we design the following five deep learning models:

- Fully Connected Convolutional Neural Network (FCN)
- Long Short-Term Memory (LSTM)
- Bi-directional Long Short-Term Memory (biLSTM)
- Gated Recurrent Unit (GRU)
- Deep RNN (deepRNN)

We use the grid search method to optimize the models' corresponding hyper-parameter combinations. Tables 2 to 11 show the summaries of our proposed models after model selection and hyper-parameter tuning.

We use TensorFlow v2.4.1, pandas v1.1.5, and scikit-learn v0.24.1 to preprocess the dataset and to implement our deep learning models.

5 Performance Evaluation

Among the four CSV files provided by the Canadian Institute of Cybersecurity website, we use the DoH and nonDoH datasets as the training set in Layer 1, and take the predicted DoH values to be used as data instances to align them with the Benign and Malicious data instances for the training set in Layer 2. We use the holdout validation set to get the training set, validation set and the testing set. In addition, we run the model five times to get the average result for each performance metric.

We use a confusion matrix to give us a better idea of each model's performance. The following four are the basic criteria that help us determine the metrics we are looking for. These are the true positives, true negatives, false positives and false negatives. In the confusion matrices the DoH class is represented by binary number 0, and nonDoH class by 1 in layer 1. In layer 2, the benign class is represented by the binary number 0, and the malicious class by 1. We also use the classification measure to help us achieve a better understanding and analysis of each model and its performance. Precision is a measure of correctness that is achieved in true prediction. It tells us how many predictions are actually positive out of all the positive predicted. Out of all the predicted positive classes, we predicted a high percentage correctly for each of our deep learning models. Precision should be high; ideally 1. Recall is the measure of actual observations which are predicted correctly. That is, how many observations of positive class are actually predicted as positive. Recall should ideally be equal to 1. The F1 score sort of maintains a balance between the precision and recall for a model. If the precision is low, the F1 score is low and if the recall is low again your F1 score is low. The F1-score should be high; ideally 1.

We consider the following performance metrics:

- Confusion Matrix
- Recall
- Precision
- F1-Score

Figure 1 shows the confusion matrix of the testing set for the Fully Connected Convolutional network in Layer 1. There are a total of 233,428 data

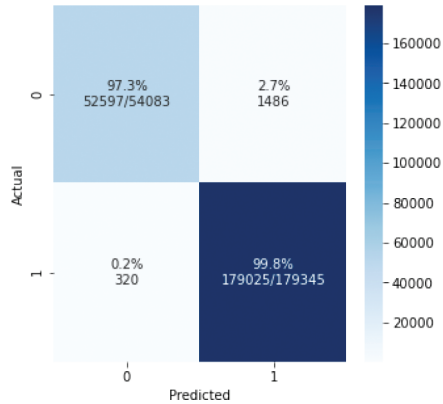

Figure 1: FCN layer 1 confusion matrix.

instances used in the test dataset after applying the holdout validation set. There are 54,083 data instances for DoH, and 179,345 data instances for nonDoH. Out of the 233,428 data instances from the test dataset there are: true negatives: 52,597, false positives: 1,486, false negatives: 320, and true positives: 179,025. FCN results show in Table 13 that almost all the recall for every class is high. The Recall is 0.992.

The precision values for the DoH class and that for the nonDoH class are high. Here we can see that the confusion matrix for the model in Layer 1 shows high percentages of true negatives and true positives for the classes of DoH and nonDoH, respectively. We can also see low percentages of false positives and false negatives in the predictions indicating a good model performance. Although the miss-classified portions are small, the vast amount of the nonDoH data instances affects the precision values. The Precision is 0.991, and F1-Score is 0.991.

Figure 2 shows the confusion matrix of the testing set for the Fully Connected Convolutional network in Layer 2. In Layer 2, we can see the confusion matrix indicating a good model performance due to the high percentages of benign and malicious values correctly predicted. The total number of data instances in the test dataset is 10,584 after using the holdout validation sets in Layer 2. The number of true negatives: 883, false positives: 32, false negatives: 19, and true positives: 9,650. It can be seen in Table 15 that almost all the recall for every class is high. The Recall is 0.975. The Precision is 0.976, and the F1-Score is 0.975.

The Precision is 0.976, and the F1-Score is 0.975. The precision values for the Benign class and that for the Malicious class are high. We can see a small portion of miss-classified class data instances.

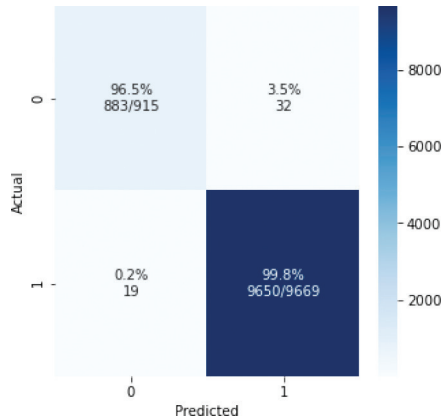

Figure 2: FCN layer 2 confusion matrix.

In working with the LSTM based algorithms (LSTM, biLSTM, GRU and deepRNN), there are 107,858 data instances in the test datasets that is fed to these models individually. Figure 3 shows the confusion matrix of the testing set for the LSTM network model in Layer 1. We can see that the confusion matrix for the model in Layer 1 shows high percentages of true negatives and true positives for the classes of DoH and nonDoH, respectively. The LSTM model predicted true negatives: 53310, false positives: 707, false negatives: 561 and true positives: 53,080. This indicates that the LSTM model performs well against the dataset. We can also see low percentages of false positives and false negatives in the predictions is indicating a good model performance. It

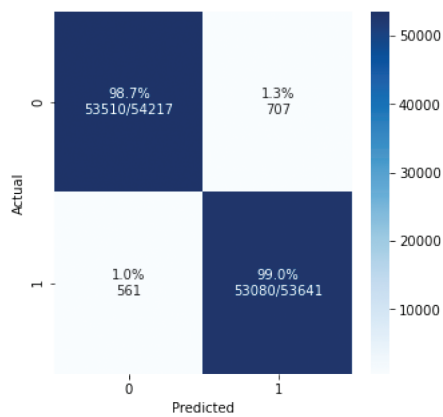

Figure 3: LSTM layer 1 confusion matrix.

can be seen in Table 14 that almost all the recall for every class is high. The Recall is 0.987. The Precision is 0.987, and the F1-Score is 0.987.

Figure 4 shows the confusion matrix of the testing set for the LSTM network model in Layer 2. We can see the confusion matrix indicating a good model performance due to the high percentages of benign and malicious values correctly predicted. The test dataset contains 3,986 data instances fed into the LSTM model in Layer 2. It can be observed that there are true negatives: 1,746, false positives: 32, false negatives: 49, and true positives: 1,921. Table 15 shows that almost all the recall for every class is high. The Recall is 0.922. The Precision is 0.924, and F1-Score is 0.918.

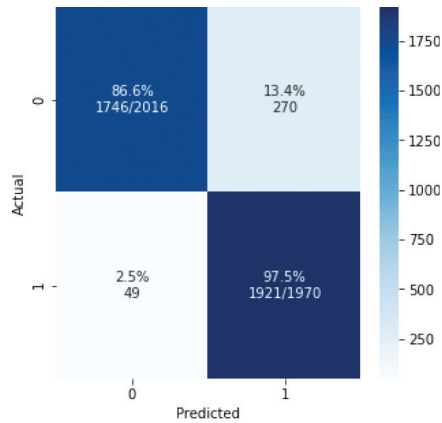

Figure 4: LSTM layer 2 confusion matrix.

Figure 5 shows the confusion matrix of the testing set for the bidirectional Long-Short Term Memory (biLSTM) network model in Layer 1. We can see that the confusion matrix for the model in Layer 1 shows high percentages of true negatives and true positives for the classes of DoH and nonDoH, respectively. Out of the number of test data instances in use, there are true negatives: 53,608, false positives: 458, false negatives: 217, and true positives: 53,575. This indicates that the biLSTM model performs well against the dataset. Table 14 shows that almost all the recall for every class is high. The Recall is 0.994. The Precision is 0.994, and the F1-Score is 0.994.

In Layer 2 of the network, Figure 6 shows the confusion matrix of the testing set for the bidirectional Long-Short Term Memory (biLSTM) network model. In Layer 2, we can see the confusion matrix is indicating a good model performance due to the high percentages of benign and malicious values correctly predicted. The number of test data instances in use is 3,899. There is a slight difference in the sample size for the biLSTM; however, it has to do with the predicted values from Layer 1. There are true negatives: 1,928, false

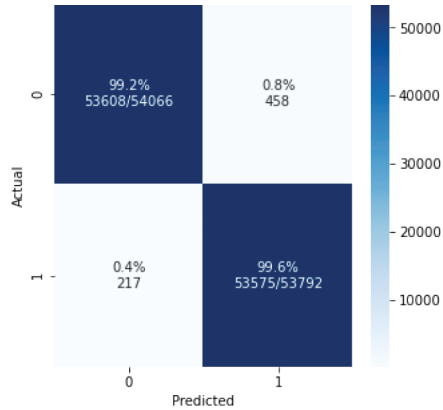

Figure 5: biLSTM layer 1 confusion matrix.

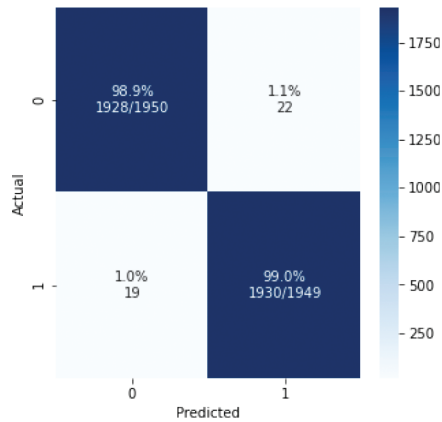

Figure 6: biLSTM layer 2 confusion matrix.

positives: 22, false negatives: 19, and true positives: 1,930. Table 15 shows that almost all the recall for every class is high. The Recall is 0.990. The Precision is 0.990, and the F1-Score is 0.990.

The third type of Recurrent Neural Network is the Gated Recurrent Unit (GRU) model. Figure 7 shows the confusion matrix of the testing set for the GRU network model in Layer 1. For the GRU model, we can see that the confusion matrix for the model in Layer 1 indicates that out of the 107,858 test data instances, there are true negatives: 53,360, false positives: 534, false negatives: 340, and true positives: 53,624. The confusion matrix shows high percentages of true negatives and true positives for the classes of DoH and nonDoH, respectively. This indicates that the GRU model performs well

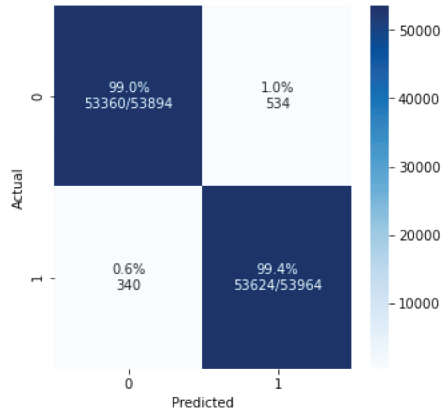

Figure 7: GRU layer 1 confusion matrix.

against the dataset. Table 14 shows that almost all the recall for every class is high. The Recall is 0.992. The Precision is 0.992, and the F1-Score is 0.992.

Figure 8 shows the confusion matrix of the testing set for the GRU network model in Layer 2. We can observe a sample number of 3,923 data instances that is fed to the model. The confusion matrix is indicating a good model performance due to the high percentages of benign (true negatives) and malicious (true positives) values correctly predicted. Table 15 shows that almost all the recall for every class is high. The Recall is 0.976. The Precision is 0.976, and the F1-Score is 0.976.

The fourth type of Recurrent Neural Network is designed as a deep RNN. Figure 9 shows the confusion matrix of the testing set for the deep RNN

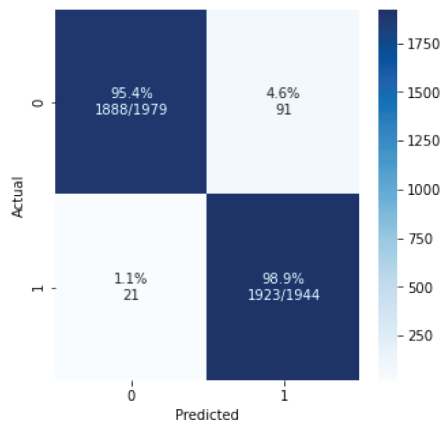

Figure 8: GRU layer 2 confusion matrix.

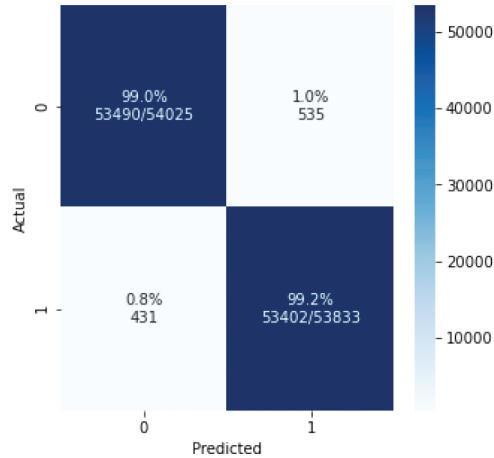

Figure 9: deepRNN layer 1 confusion matrix.

network model in Layer 1. We can see that the confusion matrix for the model shows high percentages of true negatives and true positives for the classes of DoH and nonDoH, respectively. Out of the 107,858 test data instances, there are true negatives: 53,490, false positives: 535, false negatives: 431, and true positives: 53,402. This indicates that the deepRNN model performs well against the dataset. It can be seen in Table 14 that almost all the recall for every class is high. The Recall is 0.992. The Precision is 0.991, and the F1-Score is 0.990.

Figure 10 shows the confusion matrix of the testing set for the deep RNN network model in Layer 2. The sample number of data instances that is fed to

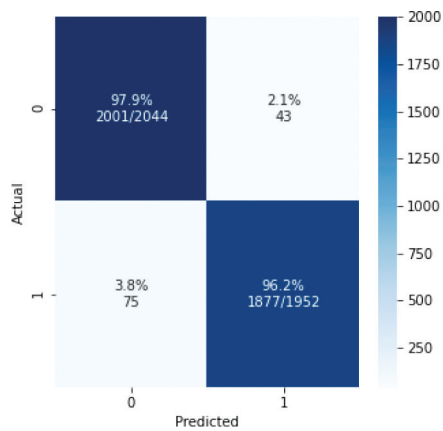

Figure 10: deepRNN layer 2 confusion matrix.

the deepRNN is 3,996. We can see the confusion matrix is indicating a good model performance due to the high percentages of benign (true negatives) and malicious (true positives) values correctly predicted. The recall for every class is high in Table 15. The Recall is 0.960. The Precision is 0.960, and the F1-Score is 0.960.

Since our proposed models are simple, the training and inference of the proposed methods are efficient. On our desktop computer with NVIDIA GeForce RTX 1080Ti GPU, the training for each epoch costs only about 6 to 17 seconds. The generated models are not big in size, so these models could be easily and quickly deployed into target devices.

6 Conclusion

In this paper, we investigate the CIC-DoHBrw-2020 dataset. We observe, analyze and preprocess the dataset. In data preprocessing it is pivotal to identify and correctly handle the missing values. We observe missing data instances in the variable, "ResponseTimeTimeSkewFromMedian", and we use mode (simple interpolation) to replace the missing values. Categorical data is encoded to numerical values using One Hot encoding which separate columns for both DoH, nonDoH, benign-DoH, and Malicious-DoH in the respective network layers. We solve the class data imbalance problem by applying the under-sampling technique whereby we balance the uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class. The splitting of the data is done into two separate sets – training set and test set, which is also known as the holdout validation method. The data is split into an 80–20 ratio. The former is used for training and the latter is used for testing. Last but not least, we apply feature scaling to standardize the independent variables within a specific range using Min/Max scalar.

The post-processing part involves hyper-parameter tuning to help us find a set of optimal parameters for the learning procedure to enable fast convergence leading to a better performing solution. We design a fully connected neural network model and four types of recurrent neural networks to solve the network anomaly detection problem using CIRA-CIC-DoHbrw-2020 dataset. These RNNs are the Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Deep Recurrent Neural Network (deepRNN) and bidirectional LSTM (biLSTM). As far as literature on intrusion network detection is concerned, there are few journal papers that have been published studying this dataset using the RNNs. We used 26 features of the 34 features that make up the CIRA-CIC-DoHbrw-2020 dataset, and achieved good results with the deep learning models. One of our contributions is the characterization of the benign-DoH and malicious-DoH classes in Layer 2. We came up with the idea to classify the DoH class and non-DoH class in Layer 1, and then only extract

the DoH predictions (only binary numbers, 0). In doing so, we pass the DoH predicted values to Layer 2. To guarantee the body of the whole dataset, we merge the DoH predicted values to the benign-DoH and malicious-DoH datasets for alignment and correct indexing. We then split the data into a ratio of 80–20, and feed the sample data to the deep learning models.

The proposed methods are simple and efficient, so that they could be applied to systems with limited resources. The FCN and the RNNs models perform very well against our dataset. The biLSTM outperformed all of the other models in both layers. The GRU is the second-best performer against the dataset in classifying DoH and nonDoH in Layer 1 as well as the characterization of Benign-DoH and malicious-DoH in Layer 2. The FCN and deepRNN models share third place in terms of good performance against the dataset. The RNNs are designed to use sequential data such as our time-series data. However, RNNs suffer from the gradient vanishing problem. To remedy this, we use a specialized RNN known as the LSTM that can mitigate this problem. We observe that our LSTM model learned very well, and predicted excellent results. Our best model performer known as the bidirectional LSTM show outstanding resilience in terms of aggregating input information in the past and the future of the specific time in the LSTM model. The preservation of information from both the past and future makes the biLSTM model outstanding against our dataset. We also use a new generation of LSTM based algorithms known as the Gated Recurrent Unit. The GRU shows excellent performance due to the fact that it is a better version of the LSTM model. Last but least, we implement a deep RNN whereby we stack the RNNs on top of each other. We use an input layer, hidden layer and an output layer; thus, reducing the complexity of our model. In this regard, we mitigate the exploding and vanishing gradients. The deepRNN performed very well against our dataset. Furthermore, it's appropriate to mention that by cleaning and preparing our dataset, we manage to acquire excellent results from our five models.

There exist several directions to further investigate into the network anomaly detection problem. We can apply both the cross-validation (k-fold) for the FCN, and Time-Series validation for the Recurrent Neural Network models. Cross-validation provides the model the opportunity to train on multiple train-test splits resulting in a better indication of how well our model will perform on unseen data. On the other hand, the holdout validation is dependent on just one train/test split. We can also design a hybrid deep learning model to effectively predict normal or malicious attacks to the internet network environment.

Table 2: FCN layer 1 model summary.

Layer 1		
Layer (Type)	Output shape	Num. parameters
Input (InputLayer)	[(None, 28)]	0
hidden1 (Dense)	(None, 30)	870
dropout1(Dropout)	(None, 30)	0
hidden2 (Dense)	(None, 30)	930
Output (Dense)	(None, 2)	62

Table 3: FCN layer 2 model summary.

Layer 2		
Layer (Type)	Output shape	Num. parameters
Input (InputLayer)	[(None, 28)]	0
hidden1 (Dense)	(None, 30)	870
dropout1(Dropout)	(None, 30)	0
hidden2 (Dense)	(None, 30)	930
Output (Dense)	(None, 2)	62

Table 4: LSTM layer 1 model summary.

Layer 1		
Layer (Type)	Output shape	Num. parameters
lstm (LSTM)	[(None, 20)]	3,920
dropout (Dropout)	(None, 20)	0
dense (Dense)	(None, 20)	420
dropout1(Dropout)	(None, 20)	0
dense1 (Dense)	(None, 2)	42

Table 5: LSTM layer 2 model summary.

Layer 2		
Layer (Type)	Output shape	Num. parameters
lstm1 (LSTM)	[(None, 30)]	7,080
dropout2 (Dropout)	(None, 30)	0
dense2 (Dense)	(None, 30)	930
dropout3 (Dropout)	(None, 30)	0
dense3 (Dense)	(None, 2)	62

Table 6: biLSTM layer 1 model summary.

Layer 1		
Layer (Type)	Output shape	Num. parameters
bidirectional (Bidirectional)	[(None, 1, 60)]	14,160
dropout2 (Dropout)	(None, 1, 60)	0
bidirectional1	(None, 60)	21,840
dense(Dense)	(None, 2)	122
Activation (Activation)	(None, 2)	0

Table 7: biLSTM layer 2 model summary.

Layer 2		
Layer (Type)	Output shape	Num. parameters
bidirectional2 (Bidirectional)	[(None, 1, 60)]	14,160
bidirectional3 (Bidirectional)	[(None, 60)]	21,840
dense1 (Dense)	(None, 2)	122
Activation1 (Activation)	(None, 2)	0

Table 8: GRU layer 2 model summary.

Layer 1		
Layer (Type)	Output shape	Num. parameters
gru (GRU)	[(None, 1, 30)]	5,400
dropout (Dropout)	(None, 1, 30)	0
gru1 (GRU)	[(None, 30)]	5,580
dropout1(Dropout)	(None, 30)	0
dense (Dense)	(None, 2)	62

Table 9: GRU layer 2 model summary.

Layer 2		
Layer (Type)	Output shape	Num. parameters
gru2 (GRU)	[(None, 1, 30)]	5,400
gru3 (GRU)	[(None, 30)]	5,580
dense1 (Dense)	(None, 2)	62

Table 10: deepRNN layer 1 model summary.

Layer 1		
Layer (Type)	Output shape	Num. parameters
simplernn (InputLayer)	[(None, 1, 30)]	1,770
simplernn1 (Dense)	[(None, 1, 30)]	1,830
simplernn2 (Output)	(None, 2)	66

Table 11: deepRNN layer 2 model summary.

Layer 2		
Layer (Type)	Output shape	Num. parameters
simplernn3 (InputLayer)	[(None, 1, 20)]	980
simplernn4 (Dense)	[(None, 1, 20)]	820
simplernn5 (Output)	(None, 2)	46

Table 12: Layer 1 training model results.

Layer 1 training				
Models	Precision	Recall	f1-score	Time
FCN	0.993	0.993	0.993	04:39.4
LSTM	0.988	0.988	0.988	25:22.7
biLSTM	0.995	0.995	0.995	15:18.8
GRU	0.992	0.992	0.992	10:22.4
deep RNN	0.993	0.993	0.993	10:36.3

Table 13: Layer 1 testing model results.

Layer 1 testing				
Models	Precision	Recall	f1-score	Time
FCN	0.991	0.992	0.991	00:16.5
LSTM	0.987	0.987	0.987	00:29.5
biLSTM	0.994	0.994	0.994	00:03.7
GRU	0.992	0.992	0.992	00:30.3
deep RNN	0.991	0.992	0.990	00:03.3

Table 14: Layer 2 training model results.

Layer 2 training				
Models	Precision	Recall	f1-score	Time
FCN	0.982	0.982	0.982	00:40.9
LSTM	0.926	0.920	0.920	01:56.5
biLSTM	0.999	0.998	0.998	00:42.8
GRU	0.984	0.984	0.984	00:41.5
deep RNN	0.964	0.964	0.964	00:38.4

Table 15: Layer 2 testing model results.

Layer 2 testing				
Models	Precision	Recall	f1-score	Time
FCN	0.976	0.975	0.975	00:01.6
LSTM	0.924	0.922	0.918	00:00.8
biLSTM	0.990	0.990	0.990	00:01.5
GRU	0.976	0.976	0.976	00:01.5
deep RNN	0.960	0.960	0.960	00:01.8

References

[1] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series Classification with COTE: The Collective of Transformation-based Ensembles," *Telecommunication Systems*, 2016, 1548–9.

[2] Y. M. Banadaki, "Detecting Malicious DNS Over HTTPS Traffic in Domain Name System Using Machine Learning Classifiers," *Journal of Computer Sciences and Applications*, 8, 2020, 46–55.

[3] V. Barnett and T. Lewis, *Outliers in Statistical Data*, Third Edition, Wiley, 1994.

[4] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," 2019.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, 41(3), 2009.

[6] L. FatimaEzzahra, D. Samira, D. Khadija, and H. Badr, "Intrusion Detection Systems Using Long-Short Term Memory (LSTM)," *Journal of Big Data*, 2021.

[7] K. Fazle, M. Somshura, D. Houshand, and C. Shun, "LSTM Fully Convolutional Networks for Time Series Classification," 1, 2017, arXiv:1709.05206v1.

[8] G. Fernandes Jr., J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença Jr., "A Comprehensive Survey on Network Anomaly Detection," *Telecommunication Systems*, 70, 2019, 447–89.

[9] D. A. E. Haddon and H. Alhateeb, "Investigating Data Exfiltration in DNS Over HTTPs Queries," in *In 2019 IEEE 12th International Conference on Global Security, Safety, and Sustainability (IGGS3)*, 2019.

[10] N. Hoque, M. H. Bhuyan, R. Baishya, D. Bhattacharyya, and J. Kalita, "Network Attacks: Taxonomy, Tools and Systems," *Journal of Network and Computer Applications*, 40, 2014, 307–24.

[11] M. Jafar, M. Al-Fawa'reh, Z. Al hrahsheh, and S. Jafar, "Analysis and Investigation of Malicious DNS Queries Using CIRA-CIC-DoHBrw-

2020 Dataset," *Manchester Journal of Artificial Intelligence and Applied Sciences (MJAIAS)*, 2, 2021, 65–70.

[12] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," *SIGCOMM Computer Communication Review (CCR)*, 34(4), 2004, 219–30.

[13] M. MontaseriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of DoH Tunnels using Time Series Classfication of Encrypted Traffic," in *2020 IEE International Conference of Dependable, Autonomic and Secure Computing, International Conference of Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress*, 2020.

[14] T. A. Nguyen and M. Park, "DoH Tunneling Detection System for Enterprise Network Using Deep Learning Technique," *Applied Sciences*, 12(5), 2022.

[15] S. K. Singh and P. K. Roy, "Malicious Traffic Detection of DNS Over HTTPS Using Ensemble Machine Learning," *International Journal of Computing and Digital Systems*, 11(1), 2022, 1061–9.

[16] D. Vekshin, K. Hynek, and T. Cejka, "DoH Insight: Detecting DNS Over HTTPS by Machine Learning," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, *ARES '20*, Virtual Event, Ireland: Association for Computing Machinery, 2020.

[17] H. Wang, W. Sun, and P. X. Liu, "Adaptive Intelligent Control of Nonaffine Nonlinear Time-Delay Systems with Dynamic Uncertainties," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47, 2017, 1474–85.

[18] K. Xie, X. Li, X. Wang, J. Cao, G. Xie, J. Wen, D. Zhang, and Z. Qin, "On-line Anomaly Detection with High Accuracy," *IEEE/ACM Transactions on Networking*, 26(3), 2018, 1222–35.