Deep Learning for Network Traffic Classification: Feature-Based vs. Raw-Data-Based

***Abstract*—Network traffic classification has been extensively used in Quality-of-Service (QoS) control, intrusion detection, and other areas of network communications and cybersecurity. To classify traffic, Neural Networks (NN) have been adopted and achieved promising performances. There are two major approaches in the NN-based traffic classification, i.e., using raw traffic data and using flow features. This paper first proposed a novel searching algorithm to find the optimal hyperparameters of the NN with the consideration of the characteristics of the network traffic data. With the optimized NNs, a comprehensive comparison was conducted between the raw-date-based and the feature-based traffic classification. The experimental results showed that the former achieved higher accuracy and precision, which means that even with partial information and less data pre-processing, NN performs more effectively and efficiently in extracting features for traffic classification.**

***Index Terms*—network traffic classification, neural networks, deep learning, traffic flow, features**

1. Introduction

Network traffic classification categorizes traffic into various types, such as http, email, stream, etc. It has been playing an important role in Quality-of-Service (QoS) control, intrusion detection, and other areas in network communications and cybersecurity. In the past, traffic flows and packets were inspected, and specific patterns were extracted and used to classify network traffic, known as Data Packet Inspection (DPI) [1]. However, the DPI approach requires human expertise to build up patterns and to relate them to a particular traffic type. With the network communications becoming more complicated and more camouflaged, it is challenging to build patterns and to use them to recognize network traffic accurately. Furthermore, for many network traffic has been encrypted nowadays, it becomes more challenging to use DPI to recognize network traffic.

With the development of machine learning and deep learning, researchers are using Neural Networks (NN) to classify network traffic [2]. As a matter of fact, neural networks play the role of extracting patterns and using them to classify traffic, just as what human experts do in DPI. With the advanced computing capacity, a well designed and trained neural network could make a more accurate prediction of traffic type. However, there are two major challenges in a neural-network-based traffic classification model. First, what information should be fed into a neural network? There are two approaches to this end, i.e., raw traffic flow date or flow features. Raw traffic date contains tremendous information. To limit the complexity of the classification model, researcher extract part of the raw data and feed them to a neural network [3]. With the concern of a huge part of the traffic information has been lost in this approach, other researchers first extract features from the entire traffic flow, then feed the features to the neural network [4] [5]. Both methods have achieved promising performances.

The second challenge of NN-based traffic classification is the neural network architecture design and hyperparameter search. With the consideration of network traffic flow characteristics, various neural networks with different complexity have been studied for traffic classification [6]. However, determining the optimal neural network architecture with the optimal hyperparameters is extremely challenging, for the searching space is tremendous. A one-dimensional Convolutional Neural Network (CNN) has been used in [7]. Meanwhile, Chen et al. converted the time series of traffic data into a two-dimensional structure and used a two-dimensional CNN for the network traffic classification [8]. A Stacked Auto Encoder (SEA) model was used for traffic classification by Wang et al. and achieved improved performance [9]. All those models have studied different NN architectures for network traffic classification and achieved good performance. However, does raw traffic data or flow features produce better classification performance? Which NN architecture works better with raw data and which works better with features? What are the optimal hyperparameters and how to find them? Those questions still remain unanswered.

In this paper, we implemented two network traffic classification models, i.e., raw-data-based and feature-based. The main goal is to conduct a comparison between the two approaches. To make the comparison fair and accurate, we proposed a two-level searching algorithm to find the optimal hyperparameters of the NN. Using the neural networks with quasi-optimal hyperparameters, a performance comparison and analyses were conducted for the raw-data-based traffic classification and feature-based traffic classification.

The rest of the paper is organized as follows. Section 2 introduces our traffic classification methodology. Section 3 proposes two approaches to generate input data for the neural network, i.e., raw data and features. Section 4 introduces the neural network design method and how to search for the optimal hyperparameters. Section 5 presents the experimental results. At the end, section 6 draws the conclusions.

2. Traffic Classification Methodology

There are three steps in our traffic classification model: data collection, data pre-preprocessing, and Deep Learning networks design and implementation. To classify network traffic into various categories using neural networks, significant amount of traffic data with labels is needed for the neural network training purpose. The data collected from either host computers or routers is a sequence of zeros and ones and needs to be transformed and organized into a readable format for the neural network. This procedure is called data pre-processing. Eventually, the formatted traffic data will be fed into a neural work for classification. The work flow is shown in Fig. 1.

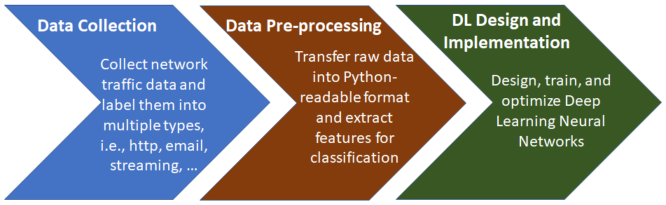

Data Collection
Collect network traffic data and label them into multiple types, i.e., http, email, streaming, …
Data Pre-processing
Transfer raw data into Python-readable format and extract features for classification
DL Design and Implementation
Design, train, and optimize Deep Learning Neural Networks

Figure 1. Methodology of Traffic Classification Using Neural Networks

The data collected for traffic classification could be either packets or flows. A network traffic flow is defined as a sequence of packets with the same 5-tuple: source IP address, destination IP address, source port number, destination port number, and protocol number. ISCXVPN2016 is a free database that provides network traffic flow data with labels [4]. The samples distribute among 14 categories: browsing, email, chat, streaming, file transfer, VoIP, TraP2P, and each type includes non-VPN and VPN traffic. Rather than using ISCXVPN2016 data, we also collected traffic flow data on host devices in our lab using Wireshark software, which generates traffic flow data with a format of pcap or pcapng. When collecting traffic data, we turned off all other applications on the same device and only keep the target traffic running. Once the traffic had been collected by Wireshark, we named the pcap file with the application name and traffic type, such as youtube_stream.pacp. In this way, we labeled each flow data in file name. The self-collected data plus ISCXVPN2016 date generates sufficient, unbiased, and diverse traffic data for NN training.

Once the traffic data has been collected, we need to transfer it into a program-readable format, such as csv files or json files. There are two considerations to this end: feeding neural network raw traffic data vs. traffic features. The information amount of a traffic flow is significant, and it is almost impossible to feed the entire flow information to the neural networks. One of the efficient solutions is to take the first N bytes of a flow and use them as the input of the neural network. This approach decreases the computational complexity of the neural network with the cost of giving up the rest of bytes of a flow. Another approach is to extract features from the entire traffic flow, such as IP address, protocol type, byte distribution, etc. Same as raw-data-based classification, the goal is to decrease the computation of the neural network. However, the classification accuracy heavily relies on the features been used. Both approaches generate input data with various size for the NN, yielding different classification accuracy. Our traffic classification model is as shown in Fig. 2.

Using either raw traffic data or flow features, we designed and implemented neural network, trained them, and tested them. With a large hyperparameter space of the neural network, we proposed a search methodology to find quasi-optimal neural network architecture which yields high classification accuracy. More details will be found in Section 4.

3. Data Pre-Processing

In this section, we introduce how to generate samples that contain raw traffic data or flow features. Both are able to generate samples with various size.

3.1. Raw-Data-Based Traffic Classification

The training data of this paper comes from two resource: ISCXCPN2016 and data collected in our lab using Wireshark software. We have collected a total of more than 200 flow files with 14 categories. The format of the flow data is pcap or pcapng, which is a sequence of bits. We first converted the raw traffic data into a sequence of bytes using a self-developed Python program. To generate enough training samples, we partitioned a flow into multiple samples. Since the flow header yields the administration information and is crucial for traffic classification, all the samples belonging to the same flow share the same flow header. Wireshark captures flow data in network layer, where the header has a maximum size of 60 bytes, sitting at the beginning of a flow sequence. Therefore, all samples take the first 60 bytes of the flow data and put them at the beginning. Assuming the size of each sample is N bytes, each sample then reads $(N - 60)$ bytes from the flow and adds them next to the flow header. The sample generation process is as shown in Fig. 3

Some flows, such as video, contains significant information, therefore can generate a lot of samples. While flows such as email could be relatively short and generates less samples. To make sure the training database contains balanced samples across all the traffic types, we have set a maximum number of samples generated by the same flow. Depending on the threshold, the total number of samples generated by the traffic flows varies. For example, with a threshold of 160, we have generated 16,426 samples from 200 pcap files.

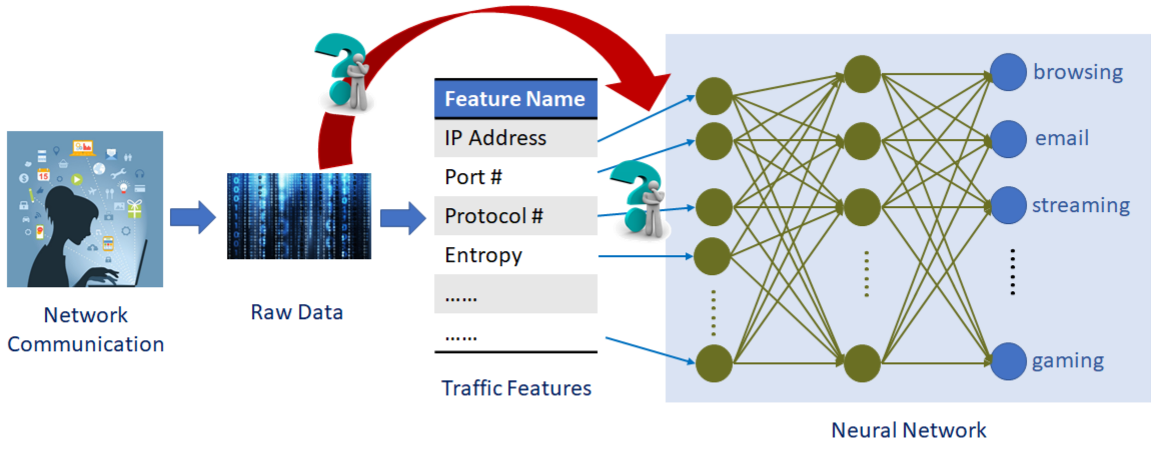
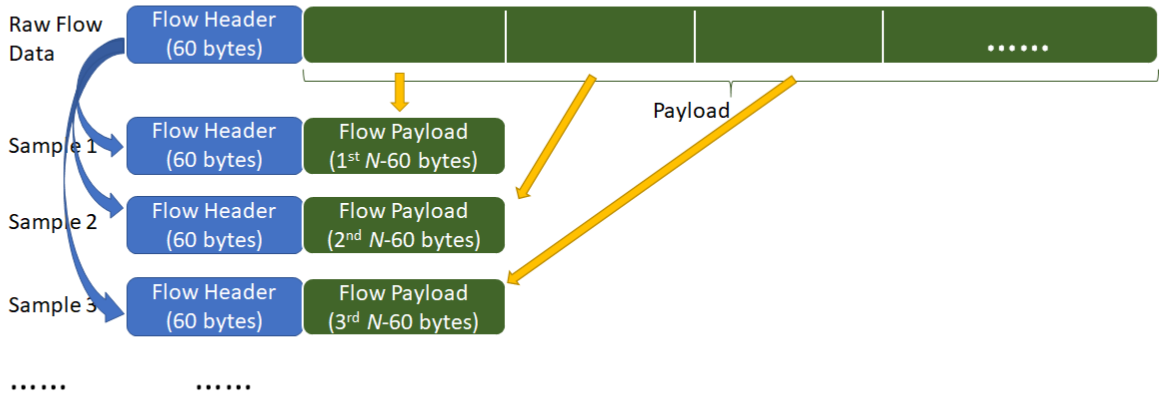


Figure 2. The Traffic Classification Model



Raw Flow Data
Flow Header (60 bytes)
Payload
Sample 1
Flow Payload (1st N-60 bytes)
Sample 2
Flow Payload (2nd N-60 bytes)
Sample 3
Flow Payload (3rd N-60 bytes)
……

Figure 3. Generate Samples from Flow Sequence: Flow Partition

3.2. Feature-Based Traffic Classification

To extract features from the raw traffic data, we used a third-party software, JOY, provided by CISCO. JOY reads a pcap or pcapng file, extracts significant flow features, and writes them into a json file [10]. The json file can be read by our program and used for a feature-based classification.

In our previous work, we had clustered traffic features into five groups depending on their characteristics, i.e., Group A) Basic Flow Information, which includes IP addresses, port numbers, and protocol number; Group B) transport layer security (TLS) information, which provides cybersecurity protocols, encryption key exchange method, authentication method, etc. [11]; Group C) Time-to-Live (TTL) information [12]; Group D) Byte Distribution and Entropy [13], Group E) Packet Sequence information. Depending on the computational complexity limit, various number of traffic features can be selected and fed to the neural network. Table 1 shows some feature selection schemes and their number of features in total. The more features being used, the more computational complexity of the neural networks would be required.

TABLE 1. Feature Selection Schemes

	Selected Features	*Total No. of features*
Scheme 1	Feature Group A	11
Scheme 2	Feature Groups A, B, C	499
Scheme 3	Feature Groups A, B, C, D, E	937

4. Neural Network Design

In this paper, Multilayer Perceptron (MLP) and CNN have been used for traffic classification. One of the challenges of the neural network design is to find the optimal hyperparameters. In this section, an efficient searching strategy for neural networks' hyperparameters is proposed.

For an MLP with an input layer of 937 neurons, two hidden layers, and an output layer of 12 neurons, we have considered various numbers of neurons for each hidden layer, i.e., 64, 320, 576, 832, 1088, 1344, 1600, and 1856 neurons. Therefore, the number of MLP models that need to train is 8^2, i.e., 64 neural networks. Feeding with 937 feature bytes, the MLP will classify the traffic type with a certain accuracy. Fig. 4 shows the classification accuracy with various neuron numbers of each hidden layer. In Fig. 4, we used different color to represent different accuracy and marked the MLPs of which the classification accuracy is higher than 92% as “good”.

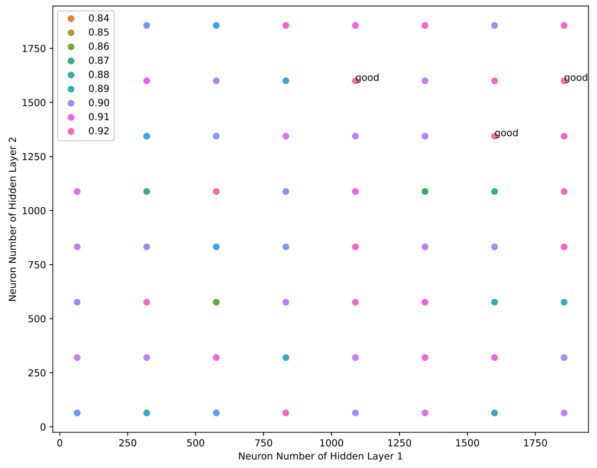


Figure 4. Classification Accuracy with Various Hyperparameters (Two Hidden Layers)

When the number of hidden layers increased to three, with the candidates of 64, 320, 576, 832, 1088, 1344, 1600, and 1856 as the number of neurons of each hidden layer, the number of various combinations is 8^3. That means a total of 512 MLPs need to been trained and tested. Fig. 5 shows the accuracy of each MLP, of which the accuracy is greater than 92% were marked as “good”.

When the number of hidden layers becomes 4, the number of hyperparameter combinations is 8^4, which means a total of 4096 MLPs need to be trained and tested. This process took weeks on a DELL Alienware work station. Fig. 6 shows the search result in a four-dimensional hyperparameter space, where the x-axis and y-axis of the left figure represent the number of neurons of the first and second hidden layers, and the x-axis and y-axis of the right figure represent the number of neurons of the third and fourth hidden layers. In Fig. 6, the neural networks of which accuracy is greater than 93% were marked as “good”.

A brutal search for the hyperparameters in a high dimensional space takes incredibly long time. To fasten the process, we need to narrow down the candidates of the MLP hyperparameters that are considered. From Fig. 4 through Fig. 6 we see that, the neural networks with higher classification accuracy are clustered in the hyperparameter space. Therefore, we can first search the entire space on a coarse-grained base, locate the hyperparameter region that yields higher classification accuracy, then conduct a fine-grained search in that region. For example, for the MLP with 4 hidden layers, the neuron number of 320, 832, 1344, and 1856 for each hidden layer are first considered, therefore,

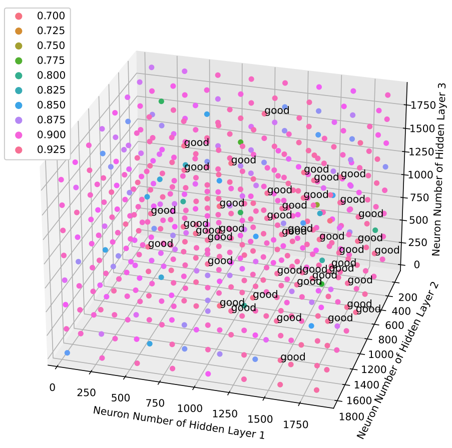


Figure 5. Classification Accuracy with Various Hyperparameters (Three Hidden Layers)

$4^4 = 256$ MLP networks are trained and tested. Among of the MLPs, there are 29 MLPs achieved an accuracy of 92% and above. Table 2 shows the top 10 trials with the highest classification accuracy after a coarse-grained search.

TABLE 2. THE TOP TEN TRIALS WITH A COARSE-GRAINED SEARCH FOR HYPERPARAMETERS (4 HIDDEN LAYERS)

trial No.	*HL 1*	*HL 2*	*HL 3*	*HL 4*	*accuracy*	*precision*
2912	1344	1344	832	1856	0.9248	0.9296
3932	1856	1344	832	832	0.9245	0.9287
4042	1856	1856	320	320	0.9245	0.9300
3930	1856	1344	832	320	0.9239	0.9260
1740	832	832	320	832	0.9236	0.9295
1870	832	1344	320	1344	0.9233	0.9299
3052	1344	1856	1344	832	0.9230	0.9275
3916	1856	1344	320	832	0.9230	0.9289
734	320	832	832	1344	0.9227	0.9312
3962	1856	1344	1856	320	0.9227	0.9289

The coarse-grained search revealed that the region of the top-right corner on the first-second hidden layer chart and the top region of the third-fourth hidden layer chart yield a higher classification accuracy. Therefore, we conducted a fine-grained search in these regions, and found the quasi-optimal number of neurons of the hidden layers 1, 2, 3, and 4 are 1088, 1856, 576, and 1600, respectively. With these hyperparameters, the MLP achieved an accuracy of 93.1%.

With a two-level hyperparameter search, the number of neural networks that need to be trained and tested to find an optimal architecture is dramatically decreased, which is particular important for a neural network with large numbers of hidden layers and neurons.

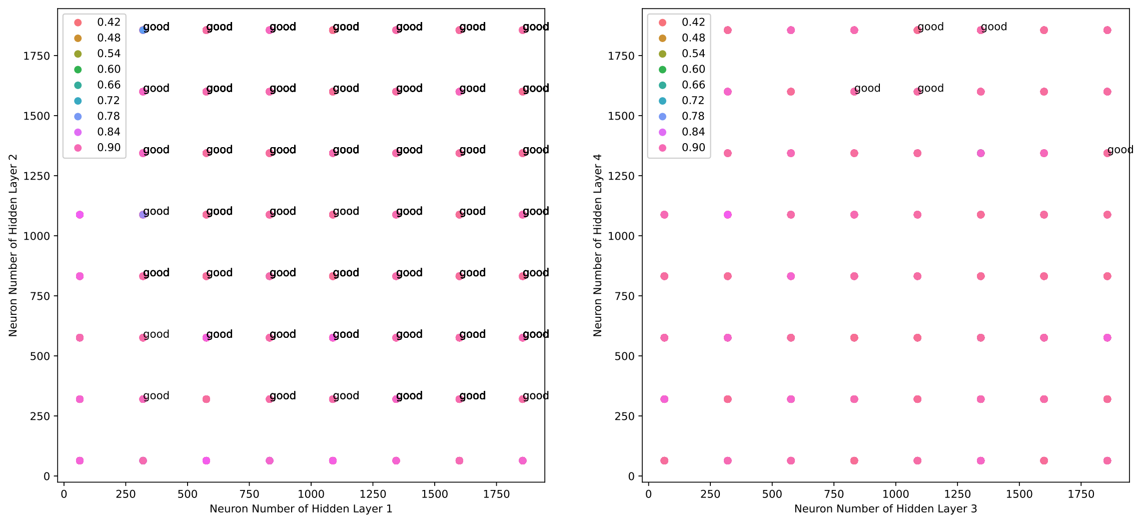


Figure 6. Classification Accuracy with Various Hyperparameters (Four Hidden Layers)

4.1. Experimental Results

In this section, we compare the classification performance of the raw-data-based neural networks and the feature-based neural networks. The comparison was conducted upon various level of computational complexity. To evaluate the classification performance, accuracy and precision are used, which are defined as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (1)$$

$$precision = \frac{TP}{TP + FP}, \quad (2)$$

where TP is the number of True Positive, TN is the number of True Negative, FP is the number of False Positive, and FN is the number of False Negative.

First, we used 499 bytes of the flow features (shown as the scheme 2 in Table 1) and 499 bytes of the raw flow data, respectively, as the input of the neural network. Using an MLP with eight hidden layers (i.e., 1600-1200-1200-880-640-360-220-80) and a 1D-CNN (i.e., CNN(100,100)-CNN(100,80)-220), respectively, the classification accuracy and precision are shown in Table 3.

TABLE 3. Classification Performances with 499 Inputs

	MLP		1D-CNN	
	Accuracy	Precision	Accuracy	Precision
Raw-Data-Based	99.259%	99.259%	98.963%	98.992%
Feature-Based	84.436%	87.019%	87.41%	88.225%

When 937 bytes of the flow features (shown as the scheme 3 in Table 1) and 937 bytes of the raw flow data, respectively, was used as the input of the neural network, the classification performances of an MLP with six hidden layers (i.e., 1088-1088-1856-832-576-1600) and a 1D-CNN (i.e., CNN(100,100)-CNN(100,80)-1200-580-120) are shown in Table 4.

TABLE 4. Classification Performances with 937 Inputs

	MLP		1D-CNN	
	Accuracy	Precision	Accuracy	Precision
Raw-Data-Based	99.449%	99.449%	98.604%	99.113%
Feature-Based	92.422%	92.963%	92.209%	93.037%

Apparently, for both less input (499) and more inputs (937), using raw flow data to classify traffic type performs better than using flow features. This means that neural networks work more effectively in extracting features and using them to recognize traffic. When features are used for the classification, CNN performs better than MLP. This is because that CNN captures the correlation among the packets of a flow (presented by feature group E as we discussed in subsection 3.2). However, when raw data is used, it is hard to find correlations among flow bytes using a 1D-CNN. Therefore, MLP performs slightly better than 1D-CNN when raw traffic data was used.

5. Conclusions

To classify network traffic, this paper first proposed an efficient way to search the hyperparameter space of the neural networks. With the optimized NNs, a comprehensive comparison was conducted between the raw-data-based and the feature-based network traffic classification. The experimental results show that using part of the raw data of a network flow, the classification accuracy and precisions are higher than using pre-designed features. This means that the neural networks work more effectively in extracting features and using them to recognize traffic. In future, we could take a peek at what features the hidden layers of the NN were extracting, which would give us better understanding about what network flow information indicates traffic types.

References

[1] Tomasz Bujlow, Valentin Carela-Espanol, and Pere Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," Computer Networks, vol. 76, pp. 75–89, Jan 2015.

[2] Shahbaz Rezaei and Xin Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," in IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019, doi: 10.1109/MCOM.2019.1800819.

[3] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," In Proceedings of 2017 International Conference on Information Networking (ICOIN), pp. 712–717, Da Nang, Vietnam, Jan. 2017.

[4] Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, and Ali A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), pp. 407–414, Rome, Italy, Feb. 2016.

[5] Qian Mao, Charles O'Neill, and Ke Bao, "A feature-based network traffic classification approach," International Journal of Network Security, vol. 25, No. 5, pp. 821–828, Sept. 2023.

[6] Shahbaz Rezaei and Xin Liu, "Deep learning for encrypted traffic classification: An overview," IEEE Communications Magazine, vol. 57, no. 5, pp. 76–81, 2019.

[7] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," In Proceedings of 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 43–48, Beijing, China, Jul. 2017.

[8] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng, "Seq2img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks," In Proceedings of 2017 IEEE International Conference on Big Data (Big Data), pp. 1271–1276, Boston, MA, USA, Dec 2017.

[9] Zhanyi Wang, "The Applications of Deep Learning on Traffic Identification," Tech. Rep. Blackhat USA 2015.

[10] David McGrew, Blake Anderson, Philip Perricone, and Bill Hudson, Cisco Systems Advanced Security Research Group (ASRG) and Security and Trust Organization (STO), https://github.com/cisco/joy.

[11] RFC5289, https://www.iana.org/assignments/tls-parameters/tls-parameters.xml.

[12] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, https://datatracker.ietf.org/doc/html/rfc791#section-1.4, IETF., Sept. 1981.

[13] Encrypted Traffic Classification, https://github.com/qa276390/ Encrypted_Traffic_Classification.