Time Series Analysis for Encrypted Traffic Classification: A Deep Learning Approach

Ly Vu[1], Hoang V. Thuy[1], Quang Uy Nguyen[1], Tran N. Ngoc[1],
Diep N. Nguyen[2], Dinh Thai Hoang[2], Eryk Dutkiewicz[2]
[1] Faculty of Information Technology, Le Quy Don Technical University, Hanoi, Vietnam
[2] School of Electrical and Data Engineering, University of Technology Sydney, Australia

Abstract—**We develop a novel time series feature extraction technique to address the encrypted traffic/application classification problem. The proposed method consists of two main steps. First, we propose a feature engineering technique to extract significant attributes of the encrypted network traffic behavior by analyzing the time series of receiving packets. In the second step, we develop a deep learning-based technique to exploit the correlation of time series data samples of the encrypted network applications. To evaluate the efficiency of the proposed solution on the encrypted traffic classification problem, we carry out intensive experiments on a raw network traffic dataset, namely VPN-nonVPN, with three conventional classifier metrics including Precision, Recall, and F1 score. The experimental results demonstrate that our proposed approach can significantly improve the performance in identifying encrypted application traffic in terms of accuracy and computation efficiency.**

Keywords- Traffic classification, LSTM, encrypted applications, deep learning, and feature engineering.

I. Introduction

Traffic classification is crucial to network resource management and security, e.g., policing, firewall, filtering, anomaly, and intrusion detection [1], [2]. To protect users' privacy (e.g., Viber, Whatsapp), most of applications come with an option to encrypt users' traffic. Moreover, malicious users also want to hide their behaviors through encrypted or covert tunnels [3]. As such traffic identification of encrypted applications plays a more and more pivotal role in cyber security.

Dealing with encrypted traffic also brings many challenges for existing network traffic classifiers. Encryption algorithms try to embroil data structure of packets, leading to reducing the effects of payload-based methods [4]. Alternatively, the flow-based methods usually exploit statistical features of network flows [5], [6], [7], resulting in increasing the computation resource and decreasing accuracy comparing with payload-based methods. Therefore, conventional payload-based and flow-based techniques are not effective in both computation efficiency and representing encrypted packets for classification purpose.

Although many machine learning-based classification methods have been introduced to overcome the current limitations of conventional methods, their effectiveness strongly depends on the accuracy and effectiveness of the feature extraction process. However, as aforementioned, traffic of most applications has recently been encrypted [8], making the feature extraction from the raw traffic extremely challenging.

In this paper, we propose a novel deep learning-based classification approach using an effective feature engineering technique to represent encrypted network traffic. First, we rely on the time dependency of packets (i.e., time series samples) to better represent the behavior of traffic. We then leverage the advantage of Long Short-Term Memory (LSTM) network [4] [9] to extract the time dependency from the encrypted traffic. To verify the efficiency of the proposed solution, we deploy experiments on the real packet dataset, i.e., VPN-nonVPN dataset [10], with 12 kinds of applications. The experimental results show the efficiency of the our proposed approach in terms of accuracy, reliability, and robustness.

The main contributions of the paper are as follows:

- We propose a feature engineering technique that combines the payload-based and flow-based methods to efficiently represent the behavior of encrypted network traffic.
- We develop a deep learning approach using LSTM to retain the time dependency of receiving network traffic through time series analysis for data of network application traffic.
- We perform experiments to demonstrate the impact of the feature engineering technique to find the best feature set that achieves the high accuracy with low computation requirement (for identifying encrypted application traffic).

The rest of paper is organized as follows. Related works in the network traffic classification with unencrypted and encrypted traffic are discussed in Section II. Section III presents the fundamental background of the LSTM model. Our feature engineering technique is then described in Section IV. The testing dataset and experimental settings are in Section V. Section VI presents experiment results. Conclusion and future works are discussed in Section VII.

II. Related Work

Flow-based, packet-based, and flow-packet combination are three major techniques to analyze network traffic in the literature.

1) Flow-based method: A flow is a set of packets that have the same source Internet Protocol (IP) address, destination IP address, source transport layer port, destination transport layer port, and transport protocol. Previous works [5], [6], [7] showed the effectiveness of the flow-based method for intrusion detection systems in classifying malicious traffic from normal traffic. Accordingly, this method is efficient in analyzing the behaviors of the set of packets. Gil et al. [11] introduced time-based attributes of packets for traffic analysis. However, these attributes require a high volume of storage to store packets over a period of time [7]. As a result, the collection of many packets into a flow before extracting features is time-consuming.

2) Payload-based method: The payload-based method has a high processing speed because it processes one packet at a time. However, this method is ineffective in presenting user behaviors which are crucial in classifying network traffic. Analyzing encrypted packet payload without decrypting has received more attentions recently [12], [13]. However, it is only efficient for some specific applications which have extremely different packet size and transmission time, e.g., HTTP, VoIP, Video streaming, and P2P. Deep learning was also introduced for feature extraction of raw packets [4]. However, the method in [4] extracts entire raw packets to put into deep learning networks. This method can work for unencrypted traffic, but it is inefficient with encrypted traffic (see detailed discussion in Section III).

3) Payload-flow based combination method: The payload-flow based combination method which consists of both packet and flow features is widely used in many well-known benchmark dataset, e.g., UNSW-NB15 [14] and NSL-KDD [15]. This method extracts features of both packets and the corresponding flow, leading to rapid presentation of characteristics of traffic behaviors. This method is appropriate for encrypted network traffic because it can extract unencrypted packet headers along with encrypted payload. Furthermore, this method can use the flow-based technique to accurately represent network traffic behaviors. As a result, the packet-flow based combination method is widely implemented in intrusion detection systems to identify network malicious traffic [16].

There have been few works using the deep learning approach such as Stack Auto Encoder (SAE), CNN, LSTM network [4] [9] to classify network traffic. However, these works only focus on the structure of the deep network model to extract a large number of payload bytes from (large) raw packets of the flow. Consequently, the deep network requires a very large number of hidden layers with many neurons to enhance its accuracy.

The work of Zhang et al. [2] showed the effectiveness of the time series analysis of network traffic for the network traffic classification problem. They extracted the hand-craft statistical features of packets and flow to represent traffic application, leading to the heavy dependence of accuracy on human knowledge and high computation resource [1]. In our work, we only extract packet features which strongly represent encrypted packets (according to applications) and then arrange receiving packet samples in a time-series order. Furthermore, we leverage the LSTM network to extract the time dependence of time-series packets automatically. This will not only efficiently retain the characteristic of traffic, but also improve the accuracy and reduce the delay in classifying encrypted traffic/applications.

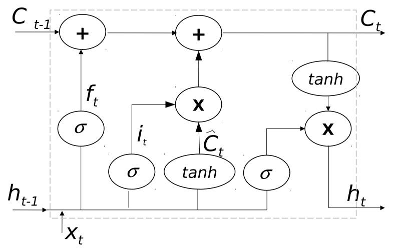

Fig. 1: The description of LSTM node.

III. Long Short Term Memory

In this section, we describe a specific type of Recurrent Neuron Networks (RNN) [17], i.e., Long Short Term Memory (LSTM). LSTM is designed to avoid the long-term dependency problem which cannot be resolved in RNN. This network structure was originally introduced by Hochreiter et al. [18], and has been refined as a powerful technique to handle the problem of time series prediction [19]. The difference of LSTM comparing with RNN resides in their nodes or cells. The basic structure of cell is presented in Fig. 1. One of the advantages of the LSTM is its ability to remove or add information to the cell state by a gate structure. There are three gates in each cell, i.e., input gate, forget gate and output gate, as shown in Fig. 1. Based on the strength of the information which is received, each node will decide to block or pass the information. The calculations of three gates are briefly shown in equations (1)-(6).

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f). \tag{1}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i). \tag{2}$$

$$\hat{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C). \tag{3}$$

$$C_t = f_t C_{t-1} + i_t \hat{C}_t. \tag{4}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o). \tag{5}$$

$$h_t = o_t \tanh(C_t). \tag{6}$$

The forget gate decides which information from previous cell is passed to the current cell. Besides, the input gate determines which information of the current cell will be updated and the output of the current cell is calculated by the output gate. Each gate is represented by a neural network which has input layers, hidden layers, and output layers.

Firstly, the forget gate is used to decide which information from the previous cell would be passed to this cell. This gate uses a sigmoid function with input of h_{t-1} presenting the output of the previous cell and x_t presenting the input of the current cell and output f_t as in (1). The cell will forget this information if f_t is zero and remember otherwise. Secondly, the input gate decides which values will be updated in this cell. As presented in the Fig. 1, the sigmoid layer will decide

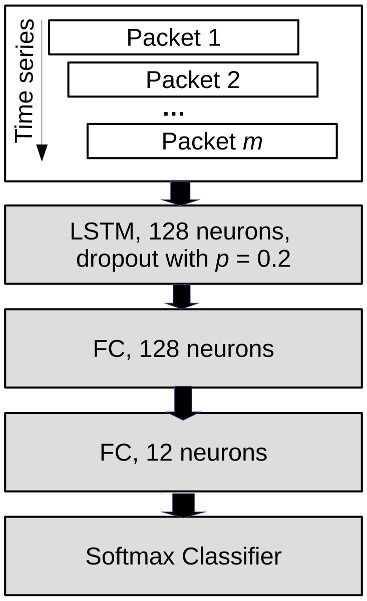

Fig. 2: The network architecture of our model.

values which will be updated, i.e., i_t calculated in (2), and the *tanh* layer creates a vector of new candidate values, i.e., $\hat{C}_t$ calculated in (3). The state of the cell is computed based on output of the forget gate and the input gate which is described in (4). Finally, the cell has to decide the output value by the output gate. The inputs h_{t-1} and x_t pass though the sigmoid layer to decide which parts of cell state being output as o_t presented in (5). The Output gate is the combination of o_t and the tanh layer of cell state C_t as shown in (6). If the time sequence length is m, the final output will be h_m which is the value of h_t when $t = m$.

One hidden layer has many LSTM cells as described above and one deep learning model is usually the combination of many hidden layers. By computing the partial derivatives of outputs, weights, and input values of hidden layers, the system can move backward to trace the error between real output values and predicted output values, and uses Gradient Descent method to update the weights concurrent in order to reduce the predicted errors. Therefore, the LSTM cell is able to use back propagation through the time. Furthermore, the LSTM cell considers one sample based on two characteristics, i.e., the value and the position in time series of the sample. This means that two input samples at different times may have the same values; however, the output will likely differ. Accordingly, the LSTM network can understand the context of samples better.

The network structure of our work is presented in Fig. 2. In this model, the packet trace is analyzed to continuous packets based on the time stamp inside each packet. Then, we use the transport ports and protocol information to group packets into a flow. Each flow has the size of m which means that a flow is represented by m continuous packets. The attribute of packet is described in the next Section. As shown in Fig. 2, the input data sample is the time sequence data, i.e., the network flow, which has m continuous packets. Our network model includes three hidden layers: LSTM (128 neurons), two Fully Connected (FC) layers with 128 and 12 neurons, respectively.

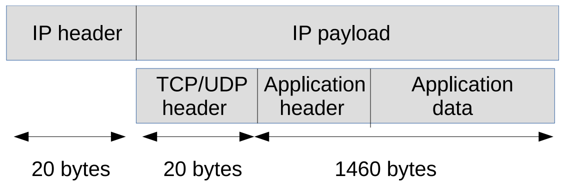

Fig. 3: The packet structure.

The softmax classifier layer with 12 neurons (according to 12 network applications in the dataset) is used as the last layer to classify input data samples.

IV. Feature engineering method

In this section, we first analyze characteristics of an encrypted network packet, then propose the feature set that can best represent encrypted network packets.

A packet is a data unit of the network layer in the Transmission Control Protocol/ Internet Protocol (TCP/IP) model. Most of previous works use the entire payload of packets as features such as 1500 bytes [4] and 1000 bytes [9]. Furthermore, Fig. 3 shows that the packet includes 20 bytes of IP header, 20 bytes of transport layer header, and all the rest of bytes for application header and application data. In ubiquitous encrypted applications, the application layer data is encrypted using symmetric encryption algorithms to ensure the speed of the encryption and decryption process such as AES256 and RC4 [20]. In other words, the encryption keys are pseudo-random numbers which are changed in different sessions [20]. Therefore, the values of encrypted application layer data are completely different between sessions, although they belong to one type of application. As a result, the application layer data is unable to represent the application type. However, the header of application layer describes the encryption algorithm names and protocols which are significant for identifying network traffic.

Fig. 4 shows features which we extract from IP packets. In this figure, features from 1 to 3 describe the identification of a flow including the source transport layer port, the destination transport layer port, and the protocol. The packet samples are ordered in a time series, so that we don't need IP address in collecting its flow. In this work, we use these information to recognize the flow of continuous packets. The fourth feature presents the size of application data which strongly represents the difference among network traffic applications. Furthermore, in the encrypted Internet application, the IP header and transport header of the IP packets are not encrypted; thus, we can represent these headers by bytes values. Features from 5 to 44, which are totally 40 features, present exactly the values of the 20 bytes IP header and 20 bytes transport header. For transport layer header, there are two popular protocols, i.e., User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The size of UDP header is 8 bytes, while the size of TCP header is usually 20 bytes. Therefore, we pad zero bytes at the end of UDP header to achieve the header size of 20 bytes. The rest of features are the first n bytes of the application which present application layer data. In the experiment, we conduct the feature sets of data with various

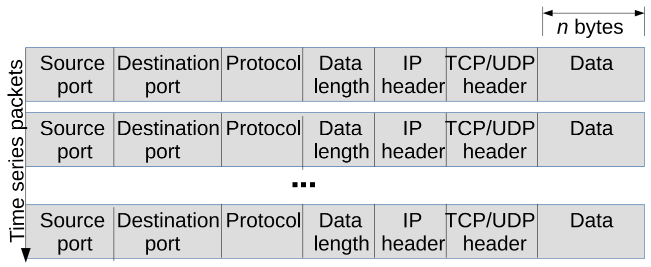

Fig. 4: The description of feature extraction.

TABLE I: Number of class samples for training and testing set

Class	Training set	Testing set
Chat	948	270
Email	328	76
File transfer	739	195
P2P	1946	482
Streaming	2304	563
VoIP	495	112
VPN-Chat	1807	476
VPN-Email	372	91
VPN-File transfer	1306	304
VPN-P2P	191	50
VPN-Streaming	657	155
VPN-VoIP	299	74
Sequence total	**11392**	**2848**

values of n, i.e., 10, 30, 50, 100, 200, 500, 1000, and 1500. Packets with application layer data smaller than n are padded by zero bytes to ensure the feature length.

V. Experimental Settings and Evaluation Metrics

This section presents the dataset and evaluation metrics which we use in the experiments.

A. Dataset

In order to test the effectiveness of the proposed method, we use well-known encrypted network traffic datasets, i.e., VPN-nonVPN [10], which was created by the Canadian Institute for Cybersecurity. To generate the dataset, they created accounts for two users, i.e., Bob and Alice, who execute network transmissions to generate traffic for applications. The captured pcap files are labeled by applications which they engaged to. There are 12 distinct labels shown in Table I. To compare with other works on VPN-nonVPN dataset, we divide the dataset into training set (80%) and testing set (20%), yielding to the number of each class sample displayed in Table I. For each training epoch, 10% number of samples are chosen for validating the best model.

B. Evaluation Metrics

We use three popular evaluation metrics in the classification problem to assess the impact of our proposed method. The reported evaluation metrics include Precision score, Recall score, and F1-score [21]. These metrics are calculated for each class i in the dataset by considering the one with all classifiers. Equations (7) and (8) present the precision and recall scores for one class. The final values of metrics are the weighted

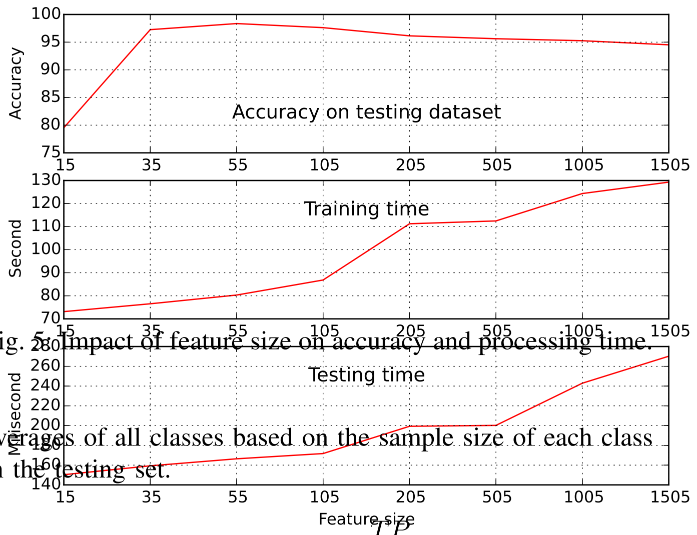

Fig. 5: Impact of feature size on accuracy and processing time.

averages of all classes based on the sample size of each class in the testing set.

$$Precision = \frac{TP}{TP + FP}. \tag{7}$$

$$Recall = \frac{TP}{TP + FN}. \tag{8}$$

The precision and recall scores are calculated as in Equations (7) and (8), where TP and FP are the number of correct and incorrect predicted samples for a class i respectively, and FN is the number of incorrect predicted samples of the rest of classes. The advantage of these metrics is that they are very intuitive and easy to implement. However, they make no distinction between classes, that is sometime insufficient to measure a classifier, especially for imbalanced datasets.

The F1-score overcomes the disadvantage of the precision and recall score. F1 score is Harmonic mean [21] of Precision and Recall, where Harmonic mean is an appropriate way to average ratios. Therefore, it is not effected by the difference of the size of classes. Precisely, F1 score is computed in Equation (9). The F1-score is often considered as a reliable metric to evaluate the performance of classification algorithms in many kinds of dataset. Thus, we will use this metric for measuring our model and compare with other works in the next section. Precision and Recall will be presented as references.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{9}$$

VI. Results and Discussion

In this section, we conduct two kinds of experiments to measure the impacts of our feature engineering technique and the accuracy of our proposed model.

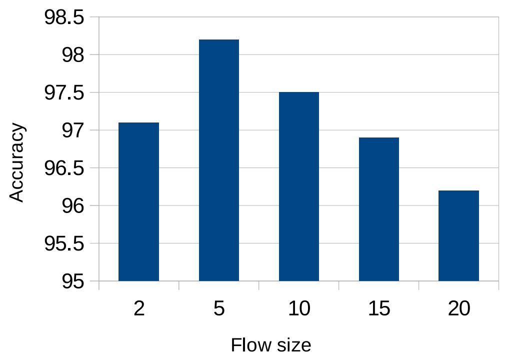

Fig. 6: Impact of flow size on accuracy.

A. Impacts of Feature Engineering Technique

In these experiments, we analyze the impacts of the feature engineering technique to the accuracy of the deep learning model. Fig. 5 presents the effects of feature size on classification accuracy and processing time. In this figure, the training time measures the processing time to execute 50 epochs in seconds and the testing time represents the processing time to predict all samples in the testing dataset. As shown in Fig. 5, the training time and predicting time increase along with the expansion of the feature size. This is reasonable for executing a deep learning model because larger feature size always requires higher computation resource and more time consuming. In addition, Fig. 5 shows that the feature size 55 which contains 50 bytes of application layer data is the most effective to present encrypted network applications. As mentioned in Section IV, each network transaction is encrypted with a different key; therefore, using the large number of encrypted payload reduces the ability to represent network applications. Hence, we consider simulations on the small feature sizes. The experimental results show that the feature size of 55 gives the highest accuracy with an appropriate processing time.

We then extract packets with 55 features, and arrange the time series packets into a flow. Fig. 6 presents the impact of the time sequence length, i.e., the number of packets in a flow. This figure shows that the time sequence length 5 enhances the highest F1-score of the classifier as 98.17% in our network model. If we increase the time sequence length, we will have to enlarge the number of hidden layers in the deep network model to attain a high accuracy of classifiers as in the work [9].

Generally, an appropriate feature set can help the training process converge faster. Fig. 7 presents the convergence ability of our model with the input size of the feature set as $n \times 5 \times 55$, where n is the time sequence length in the dataset. As shown in this figure, our network model converges quickly at 50 epochs. This conjectures that our feature set is appropriate to represent encrypted network applications for improving the performance of the application classification problem. Additionally, in the real network monitoring, if we choose the large number of packets presenting for a sequence, the model will only predict applications after passing many packets, leading to huge computation. Therefore, choosing sequence size of 5 is one of reasons that helps to rapid our model.

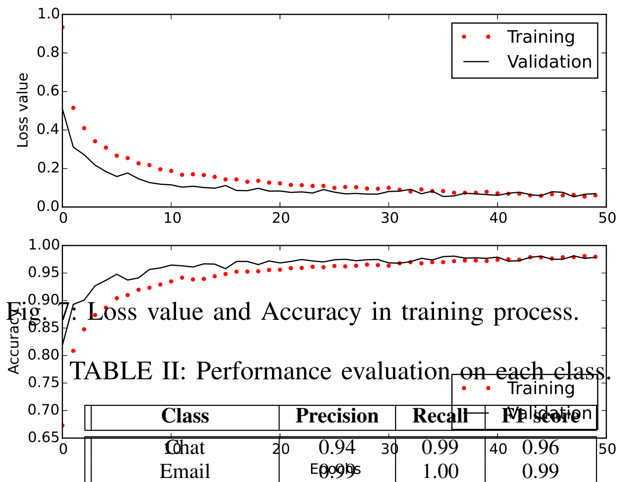

Fig. 7: Loss value and Accuracy in training process.

TABLE II: Performance evaluation on each class.

Class	Precision	Recall	F1 score
Chat	0.94	0.99	0.96
Email	0.99	1.00	0.99
File transfer	0.99	0.88	0.93
P2P	1.00	0.99	0.99
Streaming	0.99	1.00	0.99
VoIP	0.86	0.90	0.88
VPN-Chat	0.99	1.00	0.99
VPN-Email	1.00	0.98	0.99
VPN-File transfer	0.98	0.99	0.99
VPN-P2P	0.98	0.92	0.95
VPN-Streaming	1.00	0.99	1.00
VPN-VoIP	1.00	1.00	1.00
Average	**0.98**	**0.98**	**0.98**

B. Performance evaluation

This section presents the effectiveness of our model on the dataset by comparing with other works in the same domain. The performance of our model is evaluated by the evaluation metrics which are described in Section V-B, i.e., Precision, Recall, and F1-score. Table II shows the performance of our model which are calculated on each application class. As shown in this Table, both unencrypted applications and encrypted applications are identified accurately with the average F1-score up to 0.98. This demonstrates that our deep network model and feature engineering technique are very efficiency in identifying encrypted network applications.

Finally, we compare our proposed solution with some previous works in the same domain which extract the payload of packets as features. These works use deep network models to classify encrypted network traffic applications. By this experiment, we aims to prove that our technique can achieve the higher classifier accuracy with less complicated network architecture. We experiment the model proposed by Martin et al. [9] on VPN-nonVPN dataset to compare performance reliability. In Table III, the network architecture shows the number of hidden layer and the number of neurons in each

TABLE III: Comparision of deep learning methods based on packet payload features

Network	Feature size	Time sequence length	Network Architecture	F1 score
CNN+LSTM [9]	1000	20	CONV.(32), CONV.(100), LSTM(100), FC (100), FC(108)	0.96
CNN-1D [4]	1500	1	CONV.(200), CONV.(100), FC(600), FC(400), FC(300), FC(200), FC(100), FC(50)	0.97
SAE [4]	1500	1	FC(400), FC(300), FC(200), FC(100), FC(50)	0.97
Our work	**55**	**5**	**LSTM(128), FC(128),FC(64)**	**0.98**

hidden layers of models. For example, the model [9] uses the deep network with 5 hidden layers, i.e., two convolutional neuron network (CNN) layers, one LSTM layer, and two FC layers, with the number of filters (for CNN layer) or neurons (for LSTM and FC layers) at each layer as 32, 100, 100, 100, and 108 respectively. Furthermore, M. Lotfollahi et al. [4] designed two deep network models, i.e., Stack Auto Encoder (SAE) and one dimension CNN (CNN1D) with the accuracy as 0.97 on VPN-nonVPN dataset. As shown in this Table, our feature engineering technique combining with LSTM model achieves the highest accuracy of 0.98 with less complicated deep learning architecture which has only three layers, i.e., LSTM (128), FC (128), and FC (64). This verifies that our model is the most effective model for the encrypted network traffic classification problem in terms of computation resource and accuracy.

VII. Summary

In this paper, we propose a novel time series analysis for network traffic classification in order to effectively represent encrypted network applications. In particular, we first exploit significant features of network packets, then represent the flow of packets as the time series data. After that, we reconfigure data samples from raw packets to time sequence samples which can represent the behavior of network traffic. Furthermore, we take the advantage of LSTM network to design the deep network model which can learn feature effectively from time series data samples. The experimental results clearly show that using time series analysis of encrypted network traffic to represent encrypted network application combining with LSTM network can help the classifier achieve better performance for the encrypted network traffic classification. Our work can establish fundamental principle for utilizing time series features of encrypted network traffic and the deep learning approach to represent network traffic. This is the most important step in analyzing network traffic in order to provide valuable performance for various research works on network traffic analysis such as anomaly detection and traffic classification.

References

[1] T. T. T. Nguyen, and G. Armitage. “A survey of techniques for Internet traffic classification using machine learning,” *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56-76, Sept. 2008.

[2] F. Zhang, W. He, X. Liu, and P. G. Bridges,“Inferring Users Online Activities Through Traffic Analysis,” in *ACM Conference on Security and Privacy in Wireless and Mobile Networks*, Germany, Jun. 2011.

[3] P. Velan, M. Cermak, P. Celeda, and M. Draar. “A survey of methods for encrypted traffic classification and analysis,” *International Journal of Network Management*, vol. 25, pp. 355-374, Jul. 2015.

[4] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian. “Deep packet: A novel approach for encrypted traffic classification using deep learning,” arXiv preprint arXiv:1709.02656. Sept. 2017.

[5] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. “An evaluation framework for intrusion detection dataset,” *International Conference on Information Science and Security*, pp. 1-6, Thailand 2016.

[6] I. Sharafaldin, A. H. Laskari, and A. A. Armitage “Toward generating a new intrusion detection dataset ans intrusion traffic characterization,” *International Conference on Information Systems Security and Privacy*, Purtogal, Jan. 2018.

[7] R. Alshammari and A. N. Z. Heywood, “Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?,” *Elsevier*, vol. 22, pp. 1326-1348, ISBN 1326-1350, 2010.

[8] D. Nayor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafo, K. Papagiannaki, and P. Steenkiste, “The cost of the S in HTTPS,” *CoNEXT ’14 Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies* , pp. 133-140, Sydney, Australia, Dec. 2014.

[9] M. L. Martin, A. S. Escuevillas, and J. Lloret. “Network traffic classifier with convolutional and recurrent neural networks for internet of things,” *IEEE Access*, vol. 5, no. 18042-18050, 2017.

[10] http://www.unb.ca/cic/datasets/vpn.html

[11] G. D. Gerard, A. H. Lashkari, M. Mamun, and A. A. Ghorbani. “Characterization of encrypted and VPN traffic using time-related features,” *International Conference on Information Systems Security and Privacy*, 2016.

[12] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, “Blindbox: Deep packet inspection over encrypted traffic,” *ACM SIGCOMM Computer Communication Review*, 45(4), pp.213-226, 2015.

[13] S. Leroux, S. Bohez, P. J. Maenhaut, N. Meheus, P. Simoens, and B. Dhoedt. “Fingerprinting encrypted network traffic types using machine learning,” *NOMS2018, the IEEE/IFIP Network Operations and Management Symposium*,(pp. 1-5), 2018.

[14] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *Military Communications and Information Systems Conference*, 2015.

[15] http://nsl.cs.unb.ca/NSL-KDD/

[16] M. Finsterbusch, C. Richter, E. Rocha, J. A. Muller, and K. Hanssgen, “A Survey of payload-based traffic classification approaches,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1135-1156, Aug. 2014.

[17] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions Neural Networks*, pp.157-166, 1994.

[18] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory,” *Journal Neural Computation*, vol. 9, no. 8, pp.1735-1780, Nov. 1997

[19] Z. C. Lipton, J. Berkowitz, and C. Elkan. “A Critical Review of Recurrent Neural Networks for Sequence Learning,” *Neural and Evolutionary Computing*, May 2015.

[20] N. Singhal and J. P. S. Raina. “Comparative Analysis of AES and RC4 Algorithms for Better Utilization” in *International Journal of Computer Trends and Technology*, ISSN: 2231-280, July to Aug Issue 2011, pp. 177-181.

[21] D. M. W. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” *Journal of Machine learning Technologies*, vol. 2, pp. 37-63, 2011.