2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

TLS/SSL Encrypted Traffic Classification with Autoencoder and Convolutional Neural Network

Ying Yang[1,2], Cuicui Kang[1,2]*, Gaopeng Gou[1,2], Zhen Li[1,2], Gang Xiong[1,2]
1.Institute of Information Engineering, Chinese Academy of Sciences
2.School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
{yangying2, kangcuicui, gougaopeng, lizhen, xionggang}@iie.ac.cn

***Abstract*—With the increasing demand for privacy protection, the amount of encrypted traffic tremendously raises. Precise traffic analysis and monitoring has become a challenge since the traditional algorithms do not work well any more. To deal with the problem, many researchers extract a number of statistical features and propose some machine learning algorithms on the field of traffic analysis. In this paper, we utilize more distinctive representation of packet length and packet inter-arrival time. Meantime, we propose two deep learning approaches for better feature learning and compare them with the existing state-of-the-art machine learning algorithms. One model is Autoencoder for the purpose of extracting representative features. Another model is Convolutional Neural Network. It learns high dimensional features, improves the accuracy of classification and has been popularly used. The evaluation results show that the Convolutional Neural Network outperformed competing algorithms.**

***Keywords*—encrypted traffic classification; autoencoder; convolutional neural network**

I. Introduction

Traffic classification is an important problem of network operation and management. A network administrator can effectively implement Quality of Service (QoS) and trigger automated re-allocation of network resources for priority customers after identifying the number and type of applications and websites. It is also a core part of automated intrusion detection systems, which is used to detect patterns indicative of denial of service attacks. There is a great need to apply classification algorithm to discriminate the set of malicious and normal traffic. In recent years, the increasing demand for users privacy and data encryption has tremendously raised the amount of encrypted traffic. The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) techniques are popularly used in the cyber space to protect the privacy for end users. The encryption procedure turns the original data into a pseudo-random-like format in order to make it hard to decrypt. As a result, the encrypted data scarcely contain any discriminative patterns to identify network traffic. Therefore, the precise classification of encrypted traffic has become a challenge in modern network.

Port-based classification is known to be among the simplest and fastest method for network traffic identification. However, the pervasiveness of port obfuscation, network address translation, port forwarding, protocol embedding and random ports assignments have significantly reduced the accuracy of this approach. Traffic encryption protocol has a specific packet format that differs from others. With knowledge of these formats, it is possible to distinguish and identify individual protocols by inspecting the packet payload. However, the need for updating patterns and user privacy issues are the most important drawbacks of payload-based methods. Statistical and behaviour-based identification methods are less affected by encryption than port-based and payload-based methods[1]. The statistical-feature-based classification method does not require any knowledge of the encryption protocol packet structure. This approach depends on the features of traffic transmission, such as the time interval between packets, packet size and repeating pattern. It can be easily generalized to work with new versions and types of encryption protocols.

With the traffic statistical features, many machine learning algorithms are proposed on the field of traffic analysis and intrusion detection, such as Naive Bayes[2], Support Vector Machine[3], K-Nearest Neighbors[4], and so on. Based on the simulation of a hierarchical structure existing in human brain, deep learning can establish the mapping between the low-level signals and the high-level semantics for achieving the hierarchical expression of data characteristic[5]. It has been widely used for image recognition and classification. The traffic feature is usually one dimension vector. In this paper, we transform it into the form of two-dimensional matrix and apply it with the Convolutional Neural Network. The contributions of our work lie in two folds.

- Two deep learning models namely Autoencoder Neural Net and the Convolutional Neural Net are utilized for the traffic feature learning and classification. Autoencoder model[6] is used to compress and extract representative features. Convolutional Neural Network is a class of deep, feed-forward artificial neural networks. The depth model obtains the high level features of the data after the layer by layer abstraction, and has a stronger ability to express and identify the traffic data.
- On the other hand, two popularly used statistical features based on packet size and packet inter-arrival time are

* Corresponding Author. Email: kangcuicui@iie.ac.cn.

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00079

analyzed and discussed in the work. We compared the proposed methods with five competing algorithms under three feature sets. And the experimental results show that the feature based on packet size is more impactful than the feature based on packet inter-arrival time under the circumstance of encrypted traffic classification.

The remains of the paper are organized as follows. In Section II we briefly review the related work on traffic identification and analysis and the application of the machine learning algorithms. Then, Section III presents the two proposed algorithm models in detail. Section IV displays the experimental result. Finally, the paper is concluded in Section V.

II. Related Work

Most of the feature-based traffic classification methods use statistical or machine learning methods. A vast number of machine learning approaches have been published to classify traffic. Some of the methods focus on identifying encrypted traffic, whereas others try to identify the underlying application protocol.

The Bayesian classifier was firstly applied to internet traffic classification problem by Andrew Moore et al.[7]. In the work, the simple Naive Bayes estimator achieved about 65% accuracy on per-flow classification, and with two powerful refinements the performance was able to be improved to 95%. Auld et al.[2] proposed a Bayesian neural network that was trained to classify the same set of applications, and achieved 99% accuracy.

Gil et al.[8] used time related features such as the duration of the flows, flow bytes per second, forward and backward inter-arrival time to characterize the network traffic using K-Nearest Neighbor(K-NN) and C4.5 decision tree algorithm. They achieved approximately 92% recall, characterizing six major classes of traffic including Web browsing, email, chat, streaming, file transfer and VoIP using C4.5 algorithm. Yamansavascilar et al.[9] manually selected 111 flow feature described in[10] and achieved 94% of accuracy for 14 class of applications using K-NN algorithm. Bagui et al.[11] focused on classification of network traffic which is encrypted, where one is tunneled through a VPN, and the other one is normally encrypted (non-VPN transmission). Six machine learning algorithms with time-related features were evaluated on the databases, including logistic regression, support vector machine, Naive Bayes. K-Nearest Neighbor and the ensemble methods, the Random Forest classifier and Gradient Boosting Tree(GBT). Shen et al.[12] proposed a new method by incorporating the attribute bigrams into the second order homogeneous Markov chains. The work started by exploring approaches that can further imporove the performance of existing methods in terms of discrimination accuracy. Alshammari and Zincir-Heywood have published several papers[13], [14], [15], [16] on traffic classification using various supervised machine learning methods. They focused on recognising SSH, Skype, and in one case, Gtalk traffic using flow features without port numbers, IP addresses or payloads. The following algorithms for traffic classification were compared: AdaBoost, RIPPER, Support Vector Machine (SVM), Naive Bayes, C4.5 and Genetic Programming.

There are also some researches about website fingerprinting. Cai et al.[17] used an SVM with a custom kernel based on an edit-distance and achieved more than 86% accuracy for 100 sites. The edit distance allowed for delete and transpose operations, that are supposed to capture drop and retransmission of packets respectively. Following a similar approach, Wang and Goldberg[18] experimented with several custom edit distances and improved Cai et als attack to 91% accuracy for the same dataset . Wang et al.[19] proposed a website fingerprinting attack which was based on a K-NN classifier with more than 3000 traffic features. The large amount of features was obtained by varying the parameters of set of fewer feature families. Their results showed that this attack achieved 90% to 95% accuracy on 100 websites. Hayes et al.[20] proposed K-Fingerprinting which was based on Random Forest. Their feature sets included 175 features developed from features available in prior work. The random forest was not used to classify but as a way to transform these features into a different feature space. They used the leafs of the random forest to encode a new representation of the sites they intended to detect that was relative to all the other sites in training set. Then, the new representation of the data was fed to a K-NN classifier for the actual classification. Andriy et al.[3] used the cumulative sum of packet sizes and sampled the piecewise linear interpolant of the cumulative sums at 100 equidistant points. They ended with the 100 points and other four numbers, the number of incoming packets, the number of outgoing packets, the sum of incoming packet sizes and the sum of outgoing packet sizes, as the features to represent a traffic instance. Their evaluations demonstrate this attack success that ranges between 90% and 93% for 100 websites and more than 300,000 samples.

The methods we describe above are supervised machine learning methods. Semi-supervised machine learning methods are also used to classify encrypted traffic. Bernaille and Teixeira[21] used traffic clustering to detect applications encrypted by SSL. Their method had three steps. Firstly, they detected SSL connections using a clustering algorithm (Gaussian Mixture Model) on packet sizes and directions of initial packets of a connection. The first three packets and 35 clusters provided good accuracy in detecting SSL traffic. After the SSL traffic was identified, the first data packets of the connections are identified. The sizes of the data packets are used by a clustering algorithm to detect an underlying application in the third step. The evaluation was done on traffic traces from live networks and a manually generated packet trace. The dataset contained five application protocols encrypted using SSL. The authors showed that using a combination of clustering and port numbers to differentiate between applications in clusters provided better results than clustering alone. Deep learning is also applied to traffic identification. Wang[22] used Artificial Neural Network and Stacked Autoencoder. The results showed their approach works well on the applications

of feature learning, protocol classification, anomalous protocol detection and unknown protocol identification. In this paper, we explored two deep learning methods and achieved a higher accuracy than the state-of-the-art machine learning algorithms.

III. Algorithm

A. The Proposed Algorithm

1) Autoencoder: An autoencoder neural network is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs. The simplest form of an autoencoder is a non-recurrent neural network, having an input layer, an output layer and one or more hidden layers connecting them[23]. If the hidden layers are smaller than the input layer, a compressed representation can be learned, for the purpose of dimensionality reduction. Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. Figure 1 shows our architecture of autoencoder . It consists of two parts, the encoder and the decoder, which can be defined as transition ϕ_1, ϕ_2 and ψ_1, ψ_2:

$$\phi_1 : \mathcal{X} \to \mathcal{F}_1, \quad \phi_2 : \mathcal{F}_1 \to \mathcal{F}_2 \tag{1}$$

$$\psi_1 : \mathcal{F}_2 \to \mathcal{F}_3, \ \ \psi_2 : \mathcal{F}_3 \to \mathcal{X} \tag{2}$$

$$\phi_1, \phi_2, \psi_1, \psi_2 = \underset{\phi_1,\phi_2,\psi_1,\psi_2}{\arg\min} \ \|X - (\psi_2 \circ \psi_1 \circ \phi_2 \circ \phi_1)X\|^2 \tag{3}$$

The encoder stage of autoencoder takes the input $\mathbf{x} \in \mathcal{X}$ and maps it to Hidden layer1 $\mathbf{z}_1 \in \mathcal{F}_1$, then maps $\mathbf{z}_1$ to Hidden layer2 $\mathbf{z}_2 \in \mathcal{F}_2$. After that, the decoder stage maps $\mathbf{z}_2$ to Hidden layer3 $\mathbf{z}_3 \in \mathcal{F}_3$ and then maps $\mathbf{z}_3$ to reconstruction $\mathbf{x}' \in \mathcal{X}$ (Output layer), which has the same shape as $\mathbf{x}$. $\mathbf{z}_2$ was utilized as feature vector. This model is trained to minimize reconstruction errors calculated by the formulae:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 \tag{4}$$

Hyperparameter tuning is an important work to achieve the best classification performance. The same kind of deep learning model can require different constraints, weights or learning rates to generalize different data patterns. The hyperparameters should yield an optimal model which minimizes a predefined loss function on given data and avoid overfitting. We trained our models using Root Mean Square Propagation(RMSProp) with a batch size of 128 examples. RMSProp is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. RMSProp is a robust optimizer which has pseudo curvature information. Additionally, it can deal with stochastic objectives very nicely, making it applicable to mini batch learning. The learning rate was set to 0.001. In our architecture, we used sigmoid function as activation function.

Finally, we took compressed feature vector $\mathbf{z}_2$ as the input of classifier. In this model, we used Random Forest as the classifier. The dimension of three features which we described in next subpart are different. Thus, we used different hidden unit numbers for three input features. For Feature I, the unit number of three hidden layers are 128, 64 and 128. For Feature II and Feature III, the unit number of hidden layers are both 256, 64 and 256.

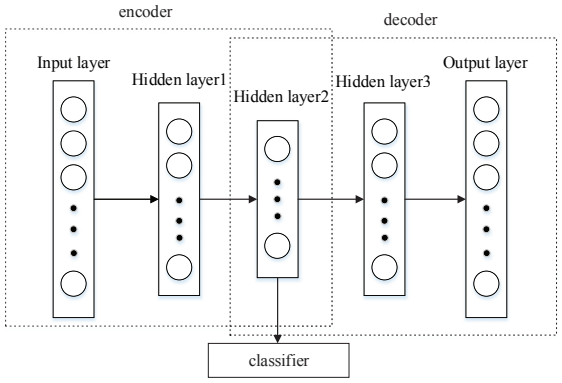

Fig. 1. The structure of Autoencoder model. Dimensionality reduction contains two stage, one is the encoder and another is decoder. After that the compressed feature vector is fed to a classifier.

2) Convolutional Neural Network: Convolutional Neural Network (CNN) is extension of traditional Multi-layer Perception, based on local receive fields, shared weights and spatial or temporal sub-sampling. A CNN consists of an input layer and an output layer, as well as multiple hidden layers[24]. The hidden layers includes convolutional layers, pooling layers, fully connected layers and normalized layers. Convolutional layers apply a convolution operation to the input, passing the result of the next layer. Pooling layers combine the outputs of neural clusters at one layer into a single neuron in the next layer. Fully connected layers connect every neuron in one layer to every neuron in another layer. The architecture of a CNN is designed to take advantage of the two-dimensional structure of an input image or other two-dimensional input such as a speech signal. This is achieved with local connections and tied weights followed by some form of pooling. We can reshape the traffic feature vector into two-dimensional structure. We reshaped Feature I, II, III into 66×10, 81×10 and 76×10, respectively. Another benefit of CNNs is that they are easier to train and have fewer parameters than fully connected networks with the same number of hidden units.

The architecture of our network is imported from Alexnet[25], and is summarized in Figure 2. The net contains eight layers with weights. The first five are convolutional and the remaining three are fully-connected. The output of the last fully-connected layer is fed to a 10-way or 50-way softmax which produces a distribution over the 10 or 50 class labels. For web10 dataset we describe in section IV, it is fed to a 10-way softmax. For web50 dataset, it is fed to a 50-way softmax.

Local-response-normalization layers follow the first and second convolutional layers. Max-pooling layers follow both response-normalization layers as well as the fifth convolutional layer. The ReLu non-linearity is applied to the output of every convolutional and fully-connected layer. The first convolu-

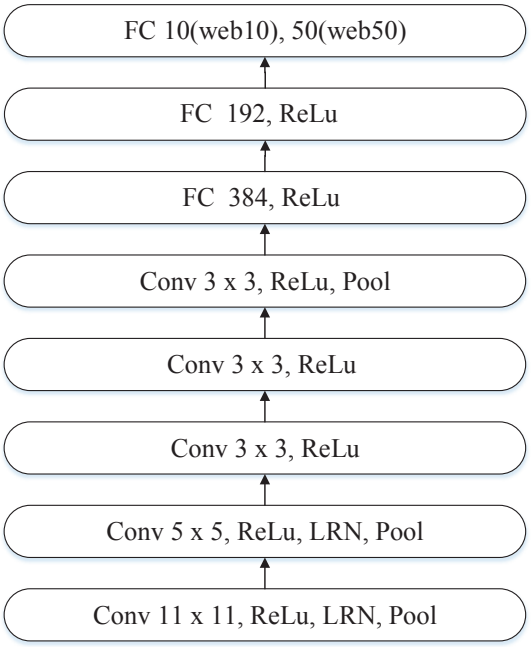

Fig. 2. The structure of Convolutional Neural Network model, consists of five convolutional layers and three full-connected layers.

tional layer filters the 66×10×1 or 81×10×1 input image with 64 kernels of size 11×11×1 with a stride of 1 pixel. The second convolutional layer takes the response-normalized and pooled output of the first convolutional as input and filters it with 192 kernels of size 5x5x64. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layers has 384 kernels of size 3×3×192 connected to the response-normalized and pooled output of the second convolutional layer. The fourth convolutional layer has 256 kernels of size 3x3x384, and the fifth convolutional layer has 256 kernels of size 3x3x256. The first two fully-connected layer have 384 neurons and 192 neurons, respectively.

We trained our model using Adaptive Moment Estimation(Adam) with a batch size of 128 examples. We used an equal learning rate for all layers and the learning rate was initialized at 0.01. We initialized the weights in three fully-connected layers from a zero-mean truncated normal distribution with standard deviation 0.04, 0.04 and 1/192.0, respectively. We initialized the neuron biases in the first two fully-connected layers and the remaining layers with the constant 0.1 and 0.0, respectively.

B. Feature Extraction

The features are extracted with the flow metadata, packet sizes, packet time-arrival times and the Unencrypted TLS Header information, which results in 811 dimensions. The flow metadata, packet length Markov Random Filed (MRF) transition matrix, packet inter-arrival time MRF transition matrix, byte distribution and the unencrypted TLS header information are the same as work[26] proposed by Blake et al., we proposed the histogram distribution which is build with the packet length and packet inter-arrival time.

1) Blake's Feature:

- Flow metadata, includes source port, destination port, total inbound bytes, total outbound bytes, total inbound packets, total outbound packets.
- Packet Length MRF Transition Matrix, uses the sizes of the payloads of the first 50 packets of a session. They assumed a 1500 byte Maximum Transmission Unit(MTU), and created 10 states of 150 bytes each, and estimated the transition probabilities between states with the collected packet sizes, finally formed a 10 by 10 matrix.
- Packet Inter-arrival Time MRF Transition Matrix, uses the packet inter-arrival time of the first 50 packets of a session. They assumed 1200 milliseconds as the maximum inter-arrival time, and created 10 states of 120 milliseconds each, and estimated the transition probabilities between states with the collected packet inter-arrival time, finally also formed a 10 by 10 matrix.
- Byte Distribution Data. The byte distribution is a length-256 array that keeps a count for each value encounted in the payloads of the packets for each packet in the flow.
- Unencrypted TLS Header Information. The 176 offered cipher suite hex codes, 21 unique extensions and TLS version information were observed in full dataset, so 198 TLS client-based features were constructed and used as a kind of feature.

2) Packet Length Distribution: The sizes of the payloads for the first 100 packets of a session are recorded. We assumed a 1500 byte MTU, and created 150 bins of 10 bytes each. Any packet size in the range [0,10) would go into the first bin, any packet size in the range [10, 20) would go into the second bin, etc. We got the number of packets that fell into different intervals in a flow. Finally, we constructed a length-150 array which represented the packet length distribution.

Figure 3 shows the packet length histogram distribution of the first flow when browsing different websites, including *wordpress.com*, *linkedin.com*, *live.com*, *mail.ru*, *microsoft.com* and *mozilla.org*. The flow of *linkedin.com* contains more packets and those packets fall into more bins. The packets of *mail.ru* fall into less bins. Different websites fall into different range of bins. From the figure we can see that the packet length histogram distribution may be a discriminative feature.

3) Packet Inter-arrival Time Distribution: The packet inter-arrival time for the first 100 packets of a session are recorded. We created 100 bins of 5 millisecond each. Any packet time-arrival time in the range[0,5) would go into the first bin, any packet time-arrival time in the range[5,10) would go into the second bin, and any packet time-arrival time longer than 495 millisecond would go into the last bin. Then, we got the number of packets that fell into different time-arrival time intervals. Finally, we constructed a length-100 array which represented the packet inter-arrival time distribution.

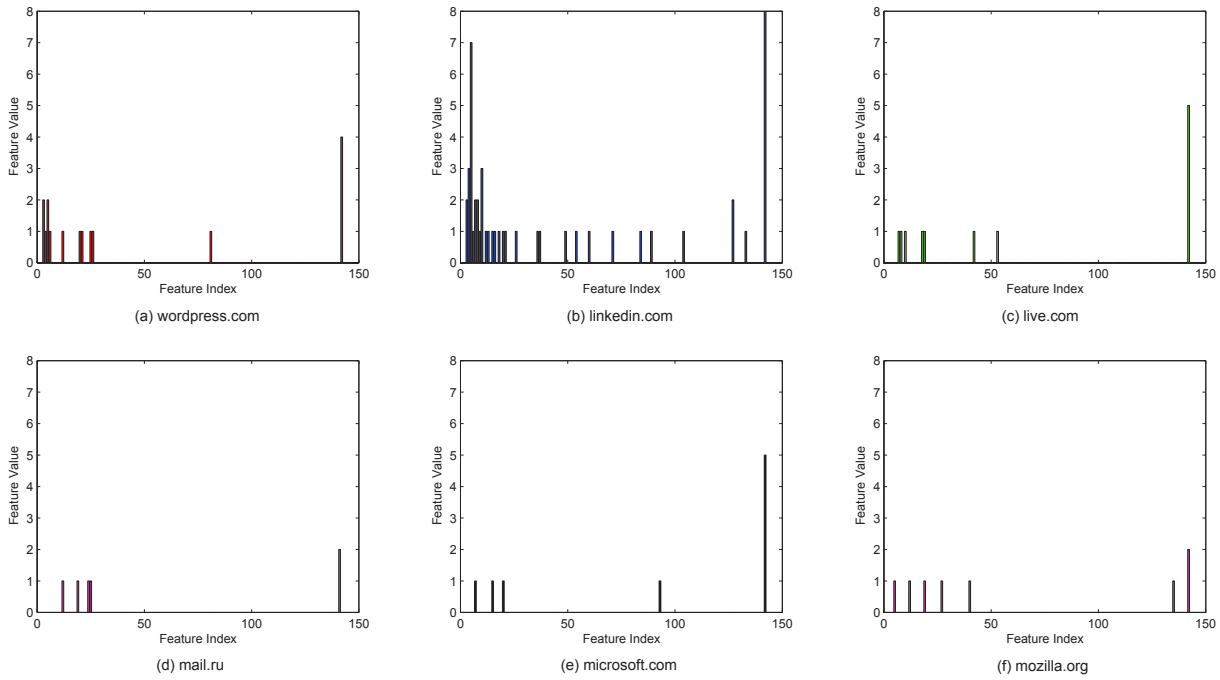

Fig. 3. The packet length histogram distribution of the first flow when we browse the website *wordpress.com, linkedin.com, live.com, mail.ru, microsoft.com and mozilla.org*, respectively.

Finally, three feature sets were conducted from these features, as you can see in Table I. PL MRF transition matrix represents packet length MRF transition matrix and PIT MRF transition matrix represents packet inter-arrival time MRF transition matrix.

TABLE I
THE DATABASE INFORMATION

Feature Type	Feature I	Feature II	Feature III
byte distribution	√	√	√
flow metadata	√	√	√
PL MRF transition matrix	√	√	√
PIT MRF transition matrix	√	√	√
unencrypted TLS header information	√	√	√
packet length distribution		√	
packet inter-arrival time distribution			√
dimension	661	811	761

IV. EXPERIMENT

A. Experimental Setting

Several classic and popularly used machine learning algorithms are used for comparison including Logistic Regression, Naive Bayes, K-NN, Decision Tree and Random Forest.

- **Logistic Regression**
  Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. In the paper, we use $\ell 1$- logistic regression.
- **Naive Bayes**
  All Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, regardless of any possible correlations between features. In the paper, we use multinomial Naive Bayes.
- **K-Nearest Neighbors**
  In K-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.
- **Decision Tree**
  The generation algorithm of decision tree includes ID3, C4.5 and C5.0. Decision tree is a tree structure, where each internal node represents the judgement on an attribute, each branch represents a judge result, finally each leaf node represents a classification result.

- **Random Forest**
  Random Forest is an important ensemble learning method based on bagging. It creates forest in a random manner, consisting of many decision trees. There is no correlation between decision trees. When a new input comes in, each decision tree makes a separate decision. The most selected class is predicted to be the output.

We used 10-fold-cross-validation and randomly selected 9 folds as training set and the left data as testing set. Evaluation criterion is accuracy and precision-recall curve. The accuracy is the raw count of flows that are classified correctly divided by the total number of flows. We used average of 10 round values as the final result. Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. The precision-recall curve shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning precise results, as well as returning a majority of all positive results.

B. Database

We selected 50 web sites encrypted with TLS/SSL protocol from the top 500 webs in the Alexa web ranking list. The traffic flows were captured in pcap file format through ChromeDriver. The experiment is conducted in two dataset, which are summarized in Table II. The 50-web dataset refers to the data of all 50 websites we captured, and the 10-web dataset includes data of 10 web sites that are randomly selected from the whole 50 websites. In the experiment, each flow is treated as a sample. Thus, there are 26017 instances in the 50-web dataset.

The dataset is collected based on the ChromeDriver, which provides the APIs that can be used to launch the Chrome browser with a particular extension installed. TShark is used to capture the traffic flows. It captures packet data from a live network, and reads packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. The native capture file format of TShark is pcap format. Then, the Joy[1]is utilized to extract the network data into a JSON format that containing the relevant data characters and features. Joy is a libpcap-based open source package, which can process both the live traffic and the packet capture files.

TABLE II
THE DATABASE INFORMATION

Feature Dataset	Number of flows	Number of packets
web10	4902	158878
web50	26017	2114551

TABLE III
THE ACCURACY OF THE CLASSIFICATION OF 10 WEB SITES UNDER DIFFERENT ALGORITHMS AND DIFFERENT FEATURES

algorithm	Feature I	Feature II	Feature III
Naive Bayes	0.6565	0.7593	0.6790
Logistic Regression	0.8815	0.9689	0.8953
K-NN	0.9009	0.9523	0.8725
Decision Tree	0.9178	0.9510	0.9170
Random Forest	0.9249	0.9578	0.9074
Autoencoder	0.9293	0.9619	0.8712
CNN	0.9646	0.9790	0.9733

C. On Web10 Database

Table III and the above three figures in Figure 4 show the accuracy results and precision-recall curves of seven algorithms under three features. CNN outperformed competing methods. CNN had the highest accuracy and its effect remained stable. It was 3.53%, 1.01%, 5.63% higher than the second best algorithm using three features, respectively. Using Feature II, CNN achieved the highest value 97.9% in all cases. CNN is a kind of multi-layer neural network specially designed for processing two dimensional data. The convolution neural network can excavate the spatial correlation in the data. The worst results are obtained when using Naive Bayes classifier. Using Feature I, Naive Bayes achieved the lowest value 65.56% in all cases. The reason might be all Naive Bayes classifier assume that the value of a particular feature is independent of the value any other feature regardless of any possible correlations between the features. Autoencoder got similar performance to the best-performing machine learning algorithm using Feature I and II. But Autoencoder was the second worst algorithm using Feature III. The reason may be that the initialization of the weights or the setting of parameters is not accurate enough, or the optimization of this Autoencoder model is insufficient.

Adding the feature packet length distribution makes results of all algorithms improved. The accuracy of Naive Bayes and Logistic Regression has improved dramatically, achieving 10.28% and 8.74% higher than just using Blake's feature. Logistic Regression was the second best algorithm using Feature II, while it got the second worst result using Feature I. Adding the packet inter-arrival time distribution makes results of four algorithms worse. Other three algorithms just improved 2.25%, 1.38% and 0.87%, respectively. The packet length distribution has better classification effect than the packet inter-arrival time distribution.

D. On Web50 Database

Table IV and the three figures in the below of Figure 4 show the accuracy results and precision-recall curves of seven algorithms under three features. CNN also got the best result. This is consistent with the result on web10 dataset. It was 3.59%, 2.42%, 3.72% higher than the second best algorithm

[1]https://github.com/cisco/joy

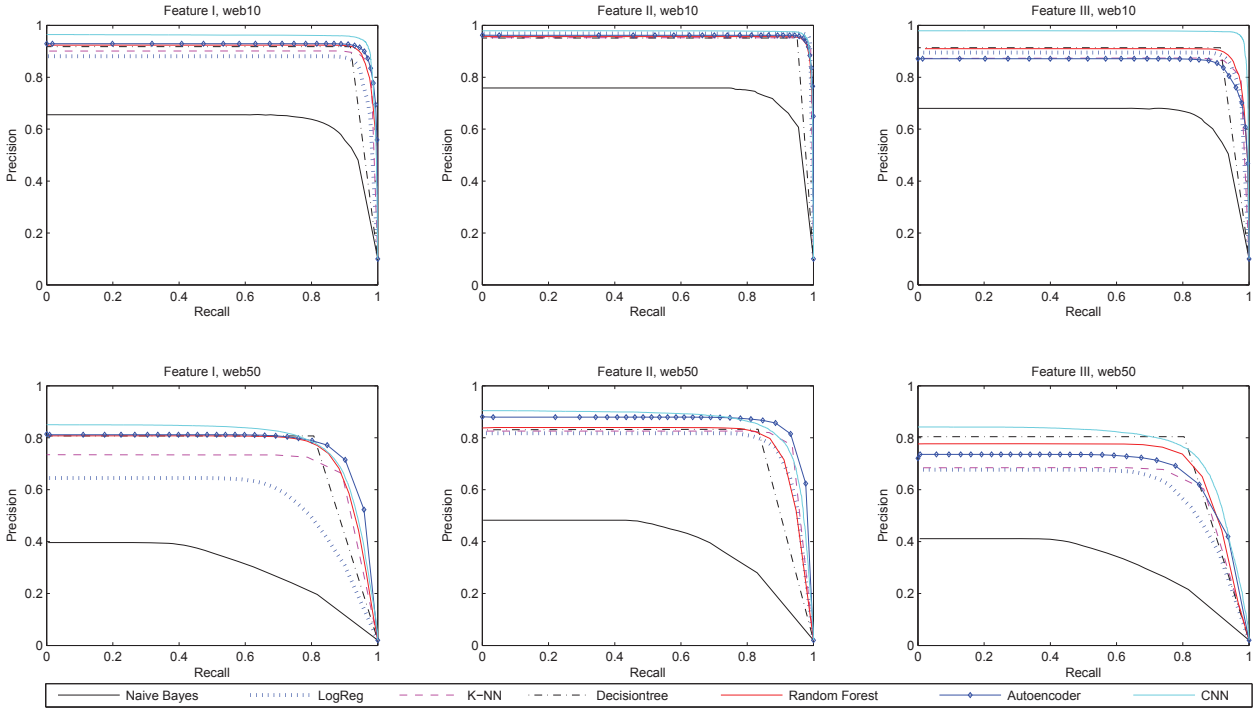

Fig. 4. Precision-Recall curves using Feature I, Feature II and Feature III on web10 and web50 dataset.

TABLE IV
THE ACCURACY OF THE CLASSIFICATION OF 50 WEB SITES UNDER DIFFERENT ALGORITHMS AND DIFFERENT FEATURES

algorithm	Feature I	Feature II	Feature III
Naive Bayes	0.3962	0.4830	0.4106
Logistic Regression	0.6458	0.8196	0.6756
K-NN	0.7345	0.8258	0.6843
Decision Tree	0.8068	0.8305	0.8041
Random Forest	0.8065	0.8378	0.7735
Autoencoder	0.8141	0.8804	0.7209
CNN	0.8500	0.9046	0.8413

using three features, respectively. And it was 6.68% higher than the best-performing machine learning algorithm under Feature II. Naive Bayes classifier also got the worst result. The accuracy values are all lower than 50% in three conditions and Naive Bayes method can't classify website properly.

As a perspective of the three feature set, Feature II got the best classification effect. Naive Bayes and Logistic Regression got 8.68% and 17.38% higher than the accuracy values under Feature I. Five out of seven algorithms achieved worse result when adding the feature packet inter-arrival time distribution. Autoencoder model got 6.33% lower when adding the feature based on the packet inter-arrival time. We assumed that the packet length distribution is more effective than the packet inter-arrival time distribution for the purpose of classifying encrypted website traffic.

V. CONCLUSION

Traffic classification represents an essential task for both network management architectures and network security solutions. In this paper, we discuss the application of Autoencoder and CNN on the field of encrypted traffic classification. The proposed algorithms are evaluated on the encrypted traffic database of fifty websites. Five state-of-the-art algorithms are included for comparison in the experiment, namely the Naive Bayes, Logistic Regression, K-Nearest Neighbors, Decision Tree and Random Forest. The results show that our proposed CNN model works very well and outperforms other competing methods. The Autoencoder model performs better than the competing machine learning algorithms under some feature sets, while it gets relatively poor result under other feature sets. We propose two types of feature extracted from packet sizes and packet inter-arrival time. Comparing the results of the algorithms on three different feature sets, it has been proved that the packet length distribution is more discriminative than the packet inter-arrival time distribution. We believe that the application of deep learning algorithms is a significant step forward in designing traffic classification methods.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (No. 2016QY05X1000,

2016YFB0801600) and the National Natural Science Foundation of China (No. 61702501).

REFERENCES

[1] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[2] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on neural networks*, vol. 18, no. 1, pp. 223–239, 2007.

[3] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.

[4] M.-Y. Su, "Using clustering to improve the knn-based classifiers for online anomaly network traffic identification," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 722–730, 2011.

[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[6] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[7] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 50–60.

[8] G. D. Gil, A. H. Lashkari, M. Mamun, and A. 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)*, 2016, pp. 407–414.

[9] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 2017, pp. 843–848.

[10] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Tech. Rep., 2013.

[11] S. Bagui, X. Fang, E. Kalaimannan, S. C. Bagui, and J. Sheehan, "Comparison of machine-learning algorithms for classification of vpn network traffic flow using time-related features," *Journal of Cyber Security Technology*, vol. 1, no. 2, pp. 108–126, 2017.

[12] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order markov chains and application attribute bigrams," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

[13] R. Alshammari, P. I. Lichodzijewski, M. Heywood, and A. N. Zincir-Heywood, "Classifying ssh encrypted traffic with minimum packet header features using genetic programming," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2539–2546.

[14] R. Alshammari and A. N. Zincir-Heywood, "A flow based approach for ssh traffic detection," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. IEEE, 2007, pp. 296–301.

[15] ——, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–8.

[16] ——, "Can encrypted traffic be identified without port numbers, ip addresses and payload inspection?" *Computer networks*, vol. 55, no. 6, pp. 1326–1350, 2011.

[17] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[18] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013, pp. 201–212.

[19] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting." in *USENIX Security Symposium*, 2014, pp. 143–157.

[20] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique." in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[21] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *International Conference on Passive and Active Network Measurement*. Springer, 2007, pp. 165–175.

[22] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015.

[23] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[24] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 1135–1139.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[26] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1723–1732.