arXiv:2001.03665v1 [cs.NI] 10 Jan 2020

Classification of Traffic Using Neural Networks by Rejecting: a Novel Approach in Classifying VPN Traffic

Ali Parchekani, Salar Nouri Naghadeh, Vahid Shah-Mansouri
School of ECE, College of Engineering, University of Tehran, Tehran, Iran
{aliparchekani, salar.nouri, vmansouri}@ut.ac.ir

***Abstract*—Traffic flows are set of packets transferring between a client and a server with the same set of source and destination IP and port numbers. Traffic classification is referred to as the task of categorizing traffic flows into application-aware classes such as chats, streaming, VoIP, etc. Classification can be used for several purposes including policy enforcement and control or QoS management. In this paper, we introduce a novel end-to-end traffic classification method to distinguish between traffic classes including VPN traffic. Classification of VPN traffic is not trivial using traditional classification approaches due to its encrypted nature. We utilize two well-known neural networks, namely multi-layer perceptron and recurrent neural network focused on two metrics: class scores and distance from the center of the classes. Such approaches combined extraction, selection, and classification functionality into a single end-to-end system to systematically learn the non-linear relationship between input and predicted performance. Therefore, we could distinguish VPN traffics from Non-VPN traffics by rejecting the unrelated features of the VPN class. Moreover, obtain the application of Non-VPN traffics at the same time. The approach is evaluated using the general traffic dataset ISCX VPN-nonVPN and the acquired real dataset. The results of the analysis demonstrate that our proposed model fulfills the realistic project's criterion for precision.**

Keywords– Traffic classification, VPN traffic, neural networks.

I. INTRODUCTION

Traffic classification is referred to as the task where traffic flows are categorized based on the class of service. Traffic flows are the set of packets that have the same source and destination IP and port addresses. The class of service indicates the application category the flow belongs to. For example, VoIP, multimedia streaming, and video on demand are examples of traffic classes. Network traffic classification can be used in several applications, including access control, vulnerability assessment, fire-walling, and incident response [1]. Multiple functions, including tracking, identification, control, and optimization, could then be carried out on the traffic classes [2]. Traffic classification approaches must overcome the problems of increasing traffic types as well as increasing transmitting speeds equally. To cope with such pace and scale, researchers are pursuing lightweight algorithms with as little computing requirements as possible for classification purposes.

Virtual private networks (VPN) are employed to connect users over the internet to the enterprise network securely. VPN protects the security of information transmitted across internet using packet-level encryption. Due to the encryption of traffic, it is not very easy to carry out traffic classifications for VPN connections [3]. Traffic encryption methods used in VPN networks are divided into application-layer encryption, presentation layer encryption, and network layer encryption [4]. It is also possible to split the encrypted traffic classification into encrypted traffic classification, encrypted traffic analysis and comprehensive encrypted traffic classification corresponding to the trade-off in the quality of service (QoS) of the network [3]. Encrypted traffic classification is exceedingly complex because of the extensive range of applications and models. Encrypted traffic classification involves affiliating traffic towards a category of application (e.g., email, FTP), and some research has currently been conducted out on this challenge [3], [4].

There are different traffic classification approaches naming port-based, signature-based (also known as deep packet inspection (DPI)), feature-based, and host behavior-based. A port number based approach is simple to implement and very efficient in large networks. However, some applications may not have different ports. DPI based methods have several drawbacks, including significant complexity and processing load, difficult to implement on proprietary protocols, and are not applicable to encrypted traffics. Statistical and behavioral-based techniques are essential techniques for machine learning that identify traffic by utilizing a collection of specific features of the traffic flows [5]. Network traffic has statistical features (such as flow period distribution, packet inter-arrival time, and packet lengths) that are special to certain types of applications and make the distinction between different reference applications. In this paper, we use a supervised learning approach for prediction learning from a set of known features [6].

A supervised learning-based classification algorithm generates a function f, the classification algorithm, which is capable of associating some input data, typically a vector x of numerical characteristics x_i named features, with an output value c, the class name, taken from a list C of possibilities. The machine learning algorithm requires some samples of already classified data, the training set (i.e., a set of pairs $(\vec{x}, c)$) from which it learns how to identify new data, to construct such a mapping feature that can be arbitrarily complex.

In this paper, we present an end-to-end approach using neural networks for encrypted traffic classification. We employ two supervised learning based classification algorithms, most used for traffic classification research, namely multi-layer perceptron (MLP) and recurrent neural network (RNN). MLP is employed as a learning method, and instantaneously learns features specifically from raw traffic at the first step. Traffic classification is discovered layer by layer, and elevated-level characteristics are like the activation function input. In the second step, if the MLP does not identify the traffic classes, we present RNN to discover the labels. The technique is the end-to-end approach, widely utilized in technologies of deep learning. It could also gain knowledge, especially the nonlinear relationship among the raw traffic input and the predicted performance label, rather than splitting a sophisticated issue into the meta-problems.

We would elaborate on the contribution to our research in the following. First, we introduce an encrypted end-to-end system of traffic classification using MLP. Second, we discern the most excellent form of encrypted traffic simplification and the best design of the MLP pattern. If the MLP model does not recognize the features, then the RNN approach is used to define the characteristics and classify them. Eventually, we evaluate the performance of our algorithm on a public traffic dataset (i.e., ISCX VPN-nonVPN (ISCX) [3]) as well as actual data collected from a local ISP network. Our analysis shows substantial enhancements to the state-of-the-art approach.

The rest of the paper is organized as followed. Section II explains related works. The methodology of the suggested approach is presented in Section III. The model of the neural networks used in the paper is demonstrated in Section IV. Section V includes the results and evaluation of the experiments. Section VI lays out the concluding assertions.

II. Related Work

An essential characteristic of VPN traffics is that the VPN flow embeds several traffics flows inside it. Not only detecting tunnels (i.e., VPN like traffic) from non-tunnel traffic is a challenge, investigating flows inside a tunnel is of interest indeed [7]. Seeking out new approaches has always been a clear road towards workable solutions. To classify encrypt traffic, Bacquet C et al. [8] implemented genetic programming. They used an extended MOGA in feature selection and cluster count optimization for K-Means, resulting in an increase of 2% to 5%. In contrast, the FPR did not increase significantly. Xie G et al. [9] employed subspace clustering to instruct the current classification algorithm to classify each program independently using its related features, rather than separating one framework with the other using combined network topologies. The method demonstrated very pinpoint precision and had been versatile to adjust on five traces from various ISPs. The countermeasures for encrypted network traffic processing are also related to mathematical classification. Wright C et al. [10] suggested a mechanism for morphing one traffic type to appear as something in the packet size spread, utilizing convex optimization techniques to change the packets in cleartext. Protocol emulation is commonly seemed to combat traffic classification schemes through authentication software and malware. Furthermore, anti-classification strategies emerged in the coming years, and the existing approaches of classification must change tremendously to meet the problems ahead. Throughout this traffic package, there should be two feature types namely flow feature and packet features. Simple instances of these types are flow bytes per interval time and packet size. The articles, as mentioned earlier, all followed the conventional divide-and-conquer approach that works by recursively splitting apart a problem into two or more sub-problems of the same or similar form until they become sufficiently straightforward to solve. Moreover, the quantity of articles that use the end-to-end principle to conduct traffic assessment is quite limited currently. In networks designed according to the end-to-end principle, application features reside in the network's communicating end nodes, instead of intermediate nodes, like routers, which exist to set up the network. In comparison, the classification accuracy of end-to-end approaches is somewhat weak. In some of them, features apart from actual traffic were utilized as inputs by hand. In other terms, their techniques are not end-to-end processes.

The implemented end-to-end encrypted traffic identification system may exclude conventional measures such as feature architecture, extraction features, and selection features that are widely employed in conventional dividing and conquering methods. This utilizes deep neural networks to acquire further descriptive traffic features instantly.

Traffic Type	Content	Labeled No.
1	Chat	'0'
2	Email	'1'
3	Ftp	'2'
4	Streaming	'3'
5	Voip	'4'
6	VPN	'5'

TABLE I
Structure of dataset and their label.

III. Methodology

A. Data Set

Due to the fact that basic machine learning methods rely on feature selection strategies, several existing public traffic datasets include flow features datasets as well as raw traffic datasets. As an example, KDD CUP1999 and NSL-KDD have forty-one predetermined features in their dataset [11]. Such datasets can not fulfill our specific requirements for raw traffic since these data sets contain few VPN and regular traffic. Several datasets include these features and solve our concern about raw traffic. ISCX data set comprises of six features of encrypted traffic and six features of other network protocols. Nowadays, such datasets are still the most common, and indeed the features are already quite comprehensive.

In comparison, we collected and utilized traffic from the actual traffic dataset. There are six traffic classes in this dataset, including chat, email, FTP, multimedia streaming, VoIP, and VPN. The utilization of such data sets results in finding the best model for classifying traffic and validating it compared to classification aims. The specific content of the data set is shown in Table I.

B. Neural Networks

Neural Networks are widely used for different tasks in the areas of machine learning and machine vision. Their primary usage is for the task of classification. The task of classification is done by extracting features from input data, and then multiplying these features by different weight matrices in different layers of the network and then applying a non-linear function to the result. Neurons do this task.

Neurons consist of a set of weights that are multiplied to the value of different dimensions of the input and then summed together. In the final stage, they add non-linearity by applying an activation function to the sum result. Neurons can be put parallel to form a layer, and succession in the layers will form the complete neural network [12][13].

The last layer of the neural network has the same number of neurons as the number of classes available to be classified. There are different methods to classify an input based on the final layer values and make a decision about their class. The first method is to treat the last layer value of the neurons as scores of each class and assign an input to a class that has the maximum score compared to other classes. Another method of decision making is to assign each class a center, usually a one-hot representation for this center, for instance, in a 2 class classification, first class has the center [1 0], and the other class has the center [0 1]. In this approach, the last layer values of Neurons are considered as a vector, and the final decision is made by this rule: The class of an input is the one which has the minimum distance from the corresponding center.

C. The Proposed Method

In our proposed method, we classify non-VPN flows based on their type of application, and VPN is classified as a kind of flow that does not fit to any application. To evaluate whether a flow fits a particular type of class, we use two metrics, including class scores and distance from the class center. In the first approach, we classify our flows based on their score regarding each class of application, and VPN is the kind of flow that does not get the minimum score required. In the second approach, we use a one-hot representation to represent every individual class center, and we assign each flow to the corresponding class based on their distance from these centers; the VPN class is the one which has the distance that is more than the maximum permitted distance.

1) Score Method: Neural networks typically contain multiple layers, and the last layer is usually devoted to the task of classification. In this way, the number of neurons in the last layer is usually equal to the number of classes to be classified, and input is assigned to a class that its corresponding neuron has the maximum score. If there are m classes available, and the corresponding score of each neuron in the last layer is assigned by y_i, the assigned class i^* is determined based on the following rule:

$$i^* = \underset{i=1..m}{\operatorname{argmax}} y_i, \tag{1}$$

In our method, we tend to modify the rule of classification. We define a parameter λ, which acts as a threshold for the task of classification in the way that input is only assigned to a class that has the score more than the parameter λ. Otherwise, it is rejected:

$$y_i^* = \max_{i=1..m} y_i, \tag{2}$$

$$i^* = \begin{cases} \text{arg } y_i^*, & \text{if } y_i^* > \lambda \\ \text{rejected}, & \text{otherwise} \end{cases}, \tag{3}$$

As mentioned in Section 3.2, we treat VPN traffic as a kind of traffic which does not fit any other traffic type, so we assign each flow to VPN traffic if their corresponding class in the above decision-making rule is rejected.

As there are six different categories of traffic, including VPN, in the dataset, the last layer of neural network models, contains five neurons corresponding to each non-VPN traffic classes. After applying each raw input traffic to the model, there are five values regarding each non-VPN class. The decision-making rule becomes as follows:

$$y_i^* = \max_{i=1..5} y_i, \tag{4}$$

$$i^* = \begin{cases} \text{arg } y_i^*, & \text{if } y_i^* > \lambda \\ \text{VPN}, & \text{otherwise} \end{cases}, \tag{5}$$

To make our method more precise, we tend to apply the classification method in two-phase. The first phase distinguishes VPN traffic from non-VPN ones, and the second phase classifies the non-VPN traffics based on their application. Each phase consists of a neural network model that performs the task of classification. The first network finds VPN traffic based on the proposed method, and the second network classifies non-VPN traffic concerning their score in each output neuron. Since the first network is also classifying the traffic, it is more efficient to use the first network's information for classification. In order to do so, we define a parameter μ, working as a threshold to assign each input's corresponding class based on the first or second network. As the first network's scores are determined, and the maximum score of the classes exceeds λ, then the maximum score is compared to μ. In this case, if the maximum score is more than μ, the class corresponding to the maximum score is assigned as a result. In the case that the maximum score is not more than μ, the second network decides about the class of the input data. The parameter μ should be higher than the parameter λ, and the decision process is as follows:

$$y_i^* = \max_{i=1..5} y_i, \tag{6}$$

$$\gamma_i^* = \max_{i=1..5} \gamma_i, \tag{7}$$

$$i^* = \begin{cases} \text{arg } y_i^*, & \text{if } y_i^* > \mu \\ \text{arg } \gamma_i^*, & \text{if } \mu > y_i^* > \lambda \\ \text{VPN}, & \text{if } y_i^* < \lambda, \end{cases} \tag{8}$$

where y_i is the class score of each class produced by the first network, and o_i is the class score of each class generated by the second network.

2) Distance Method: As mentioned in Section 3.2.1, the last layer of neural networks performs the task of classification, and they have an equal number of neurons as the number of classes. One way to classify data based on the last layer's values is to compare their distance from each classes' center. There are several ways to assign center to each class, but the typical one is providing each class with one-hot representation as to their centers. If there are m classes available, we need an m-dimensional space to assign each class its corresponding one-hot representation as a center, c_i. In an m-dimensional space, each class center resides on the value of one on each dimension, which means the first class center has its first dimension value equal to one and the other dimensions equal to zero in an m-dimensional space, and the second class center has the value one for its second dimension value, and the other dimensions are zero-valued in an m-dimensional space. This rule applies to all classes. The decision-making process treats the values of m neurons of the last year of the model, as a point in an m-dimensional space, z. At first, the distance of the resulted point from all class centers is computed, and then the corresponding class is the one that its related center has the minimum distance from the resulted point. The decision-making process can be summarized as

$$i^* = \underset{i=1..m}{\operatorname{argmin}} \mathrm{d}(\mathbf{y}, \mathbf{c_i}) \tag{9}$$

In our method, we tend to modify the rule of classification. We define a parameter η, which acts as a threshold for the task of classification in the way that input is only assigned to a class that has the minimum distance with its center less than the parameter η. Otherwise, it is rejected:

$$d_i^* = \min_{i=1..m} \mathrm{d}(\mathbf{y}, \mathbf{c_i}), \tag{10}$$

$$i^* = \begin{cases} \arg\ d_i^*, & \text{if } d_i^* < \eta \\ \text{rejected}, & \text{otherwise} \end{cases}. \tag{11}$$

As mentioned in the previous section, we treat VPN traffic as a kind of traffic which does not fit any other traffic type, so we assign each flow to VPN traffic if their corresponding class in the above decision-making rule is rejected.

As there are six different categories of traffic, including VPN, in the dataset, the last layer of the neural network contains five neurons corresponding to each non-VPN traffic classes. After applying each raw input traffic to the model, a point in a five-dimensional space is found, z. The classification task is completed based on the following decision rule:

$$d_i^* = \min_{i=1..5} \mathrm{d}(\mathbf{y}, \mathbf{c_i}), \tag{12}$$

$$i^* = \begin{cases} \arg\ d_i^*, & \text{if } d_i^* < \eta \\ \text{VPN}, & \text{otherwise} \end{cases}, \tag{13}$$

The same procedure as the score method section is used to make the model more accurate. Two-phase classification is used for this method, and two networks are used to do so. The first network distinguishes VPN traffic from non-VPN ones, and the second network classifies non-VPN traffics based on their applications. The information of the first network is also used to classify non-VPN traffic. As mentioned in the score method section, the parameter δ is used to act as a threshold to assign classes based on either the first network or the second one. As the first network produces distances from centers, and the minimum distance is lower than the parameter η, then the minimum distance from the center is compared to the parameter δ. In this case, if the distance is lower than δ, the class corresponding to the minimum distance is assigned as a result; otherwise, the decision is based on the distance from the centers by the second network. The parameter δ should be less than the parameter η, and the decision-making process is as follows:

$$d_i^* = \min_{i=1..5} \mathrm{d}(\mathbf{y}, \mathbf{c_i}), \tag{14}$$

$$d_{2i}^* = \min_{i=1..5} \mathrm{d}(\gamma, \mathbf{c_{2i}}), \tag{15}$$

$$i^* = \begin{cases} \arg\ d_i^*, & \text{if } d_i^* < \delta \\ \arg\ d_{2i}^*, & \text{if } \delta < d_i^* < \eta \\ \text{VPN}, & \text{if } d_i^* > \eta \end{cases} \tag{16}$$

, Where c_{2i} are the classes centers in the second network, and γ is the output of the second network.

IV. Models

In order to evaluate our proposed method of classification, we used the decision-making rule on four different models of neural networks.

A. MLP

The first model that we used for the neural network consists of neurons, which do the task of the feature extracting by multiplying a weight to the input values. In our model, we have a three-layer network that has its activation function in each layer. In the first layer, neurons multiply their weights to the input that has 784 dimensions, x_{784*1}, as it is the number of features selected from each flow. The dimension of the output of this layer is 1000. In this way, the model tries to expand the space to new dimensions to find relationships between different dimensions, and the weight matrix multiplied to the input is $W_{784*1000}$. Therefore, the input to the next layer has a dimension of 1000. The output of this layer is q_{1000*1}, and is computed as $q = Wx$.

There should be some non-linearity between different layers of neural networks to create complex mappings between inputs and outputs. The function that creates non-linearity in each layer is called the activation function. The activation function that we use for the first layer is a rectified linear unit (ReLU). ReLU function is a representative of neurons' spike in a body neural system. ReLU function is defined as

$$ReLU(x) = \max(0, x), \tag{17}$$

In our model, we apply ReLU function to the output of neurons, q_{1000*1}, to get the next layer input, s_{1000*1}, as

$$s = ReLU(q). \tag{18}$$

For the second layer, we put 100 neurons to transfer the input of this layer, s_{1000*1}, to a 100-dimension output, p_{100*1}, by multiplying in the weight matrix, $U_{1000*100}$. After

Layer	Operation & non-linearity	Input Size	Output Size
1	Linear + ReLU	784*1	1000*1
2	Linear + ReLU	1000*1	100*1
3	Linear + Gaussian	100*1	5*1

TABLE II
STRUCTURE OF NETWORK IN MLP MODEL.

projection to the lower dimension, we apply ReLU function to add non-linearity and form the next layer's input, r_{100*1}:

$$p = Us, \tag{19}$$

$$r = ReLU(s). \tag{20}$$

For the last layer, the number of neurons should be chosen as five to match the number of classes available, excluding VPN traffic. So the weight matrix in this layer is V_{100*5}, and the output after multiplication is z_{5*1} as $z = Vr$. For the last layer, we use a different activation function, and we use the Gaussian activation function to produce non-linearity. The Gaussian activation function is defined as:

$$\text{Gaussian}(x) = \exp\{\frac{-||x-c||^2}{2\sigma^2}\}. \tag{21}$$

For simplicity, we assume that $\sigma^2 = 1$ and c is equal to zero.

The result of the model will be y_{5*1}, which is computed by applying Gaussian function on each dimension of z_{5*1}:

$$y = \text{Gaussian}(z) = \exp\{\frac{-z^2}{2}\}, \tag{22}$$

this model is summarized in the table II.

B. Recurrent Neural Networks (RNN)

RNN is widely used to process sequences in the inputs. These models can do this using an internal state, which allows them to store state based on the input. One of the most famous architectures in recurrent neural networks is long short-term memory (LSTM). An LSTM unit is typically composed of a cell, an input gate, an output gate, and a forget gate. The cell tries to save the values, and the gates have an impact on updating the values that the cell remembers [14].

RNN's key and most significant function is the hidden state, which recalls some sequence details. The decision made at that same period phase t-1 by a recurring network impacts the decision achieved at the time step t a moment later. Recurring networks have two types of information, the current and the recent past, which interact to decide how they respond to new data. Recurrent networks are differentiated from feed-forward networks by linking feedback loop to their past decisions, consuming their outputs as data point by moment. The sequential data is stored in the secret phase of the recurrent network, which can cover several time phases as it flows to influence the processing of each new case. It is making similarities among activities divided by many times, and these connections are considered *long-term dependencies*, since an occurrence in time downstream relies on one or more activities that come before, and is a result of them.

We would mathematically explain the mechanism of taking memory forward as

$$h_t = \phi(W\vec{x_t} + U\vec{h_{t-1}}). \tag{23}$$

At the time phase t, the secret state is h_t. This is also a method of input simultaneously time step x_t, updated by a weight matrix W applied to the previous time step h_{t-1} hidden system multiplied by its own hidden-state-to-hidden-state matrix U known as a transition matrix and identical to a Markov chain. The weight matrices are filters that determine the extent to which the current input and the secret past state are to be accorded. The error they create is returned via back-propagation and used to modify their weights until there is no lower error.

The sum of the weight input and the secret state is squashed by the function ϕ, either a logistic sigmoid function is relying on which is a standard tool for compressing quite large or even minimal quantities into a logistic domain, as well as rendering gradients feasible for back-propagation. Since this feedback loop happens at every stage of the sequence, every hidden state includes not just traces of the previous hidden state but also traces of many who followed h_{t-1} as far as memory will survive. The recurrent network will also use the first symbol to help determine the second character's interpretation, so that the initial q may lead it to infer that the next letter is u. In contrast, the initial t may lead it to infer that the next letter is h.

C. LSTM

LSTM is an artificial recurrent model of the neural network employed in the deep learning area. LSTM has feedback connections. LSTM applies to subjects, including unsegmented handwriting recognition, speech recognition, and network traffic abnormality detection. A typical LSTM device consists of a cell, an input gate, an output gate, and a gate that is overlooked. The cell recognizes values over variable amounts of time, and the three gates monitor information flows into and out of the cell. LSTM networks are well equipped to detect, analyze, and make inferences based on time series data, as there may be lags of uncertain length in a time series among significant events. LSTMs have been created to resolve the bursting and disappearing gradient problems that can be found in conventional RNN preparation. LSTM has a benefit over RNNs, secret Markov models, and some other sequence learning approaches in many applications.

The structure consists of a cell and three "regulators" of the information flow inside the LSTM structure, commonly labeled gates: the input gate, an output gate, and a forgotten gate. The cell is accountable for controlling the interactions in the input sequence between the components. The input gate regulates the degree that a new value enters into the cell, the ignored gate regulates the extent to whom a value persists in the cell. The output gate regulates the extent to which the cell value is employed to measure the LSTM unit's output activation.

Layer	Operation & non-linearity	Input Size	Output Size
1	LSTM	784*1	300*1
2	Linear + ReLU	300*1	100*1
3	Linear + Gaussian	100*1	5*1

TABLE III

STRUCTURE OF NETWORK IN LSTM MODEL.

The equation types for an LSTM unit's forward pass with a forget gate are [14]:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f), \tag{24}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \tag{25}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \tag{26}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \tag{27}$$

$$h_t = o_t \circ \sigma_h(c_t), \tag{28}$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the Hadamard product (element-wise product) is indicated by the operator. The subscript t shows the move in time. σ_g, σ_c, and σ_h are sigmoid function, hyperbolic tangent function, and hyperbolic tangent function or $\sigma_h(x) = x$, respectively. Matrices W_q and U_q include the weights of the input and recurrent links, respectively, wherein the subscription q is either the input gate I the output gate o, the forgotten gate f or the memory cell c, relying on the activation measured [14].

An RNN using LSTM units can be trained in a controlled way, on a collection of training sequences, using an optimization algorithm, such as gradient descent, coupled with time back-propagation to determine the gradients provided through the optimization process.

For LSTM units, the error persists in the cell of the LSTM device as error values are back-propagated from the output sheet. The "error carousel" continuously feeds error back to the gates of each LSTM device until the quality is cut off.

V. RESULTS

In order to train and test our proposed model, we separate the first 784 bytes in each traffic flow as inputs which contain several traffic packets. To train our model, we need to define a loss function that the model is trying to minimize during the learning phase. For two different methods, we use different loss functions respectively.

The software framework used is PyTorch [15]. 20% of data are randomly selected as test data, and 80% of data are used for the training phase. The mini-batch size was 64, and the model was trained for 20 epochs by PyTorch built-in Adam optimizer [16]. The learning rate was $1.0e-4$, and the weight decay of 0.05 was used to prevent over-fitting.

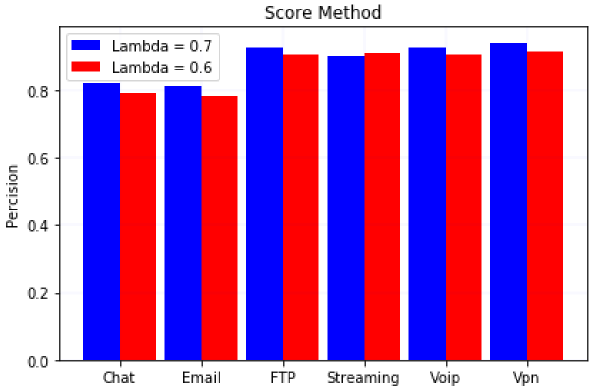

Fig. 1. The Accuracy Comparison of Score method in the two different lambda value and $\mu = 0.9$

A. Score Method Loss

We used mean squared error as a loss function for this method. The objective for this loss is to minimize the squared difference between the final result of the model, y_i, and the desired output of the model, s_i:

$$Loss = ||y_i - s_i||^2. \tag{29}$$

B. Distance Method Loss

As mentioned in the distance method loss, the distance is measured to perform the task of classification. In this approach, the model should make the final result as close as possible to the correct's class center, and as far as possible from other classes. We use the loss function of, as mentioned in [17]:

$$Loss = \sum_{i=1}^{N} \left(d_{y_i}\left(x^{(i)}\right) + \sum_{j \notin y_i} \max\left(0, \eta - d_j(x^{(i)})\right)\right). \tag{30}$$

where $\eta > 0$, $d_{y_i}(x^{(i)})$ is the distance from the correct class, and $d_j(x^{(i)})$ is the distance from the j^{th} class.

C. Performance

Figures display the precision relation of 6 types of authenticated traffic and the recall comparison.

The precision of the two methods is more excellent than 80%, as shown in the figures. The distance method's precision is better than the average, up to 1.95 percent. The accuracy of the four-class distance method loss is 1.45 percent better on average than the score method. In conclusion, on the function of encrypted traffic detection, the loss of the distance system has better performance than the loss of the score method.

VI. CONCLUSIONS

A novel end-to-end encrypted traffic classification approach utilizing deep neural networks was presented in this article focused on the study of a conventional encrypted traffic classification approach utilizing a divide-and-conquer technique. The approach combines feature configuration, extraction of

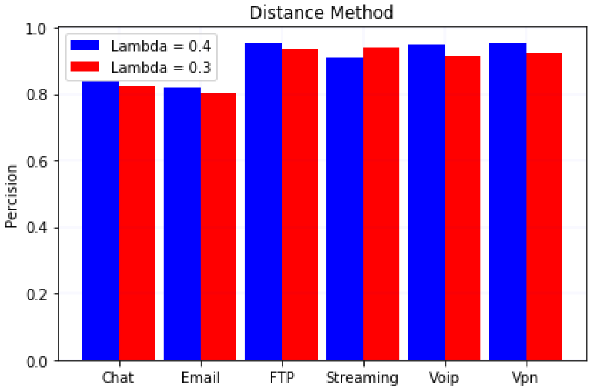

Fig. 2. The Accuracy Comparison of Distance method in the two different lambda value and $\delta = 0.1$

features, and compilation of features into a common structure. Therefore, it can obtain further traffic features efficiently. Contrary to either the divide-and-conquer approach and other strategies of artificial intelligence, the end-to-end approach has a strong adaptive impact. We noticed that the proposed neural networks are far quite suited than prior machine learning solutions to the challenge of encrypted traffic classification. The results on the mentioned datasets brought substantial refinements to all of the state-of-the-art methods, confirming the reliability of our envisaged end-to-end principle. Recent research has shown that deep learning techniques, including MLP and RNN, have excellent prospects in the traffic classification area. We intend to accurately analyze the solution suggested in this article to enhance classification of traffic capabilities.

References

[1] Z. Cao, G. Xiong, Y. Zhao, Z. Li and L. Guo, ”A survey on encrypted traffic classification” in Applications and Techniques in Information Security, Springer, pp. 73-81, 2014.

[2] E. Biersack, C. Callegari and M. Matijasevic, Data traffic monitoring and analysis. Berlin: Springer, 2013.

[3] G. Draper-Gil, A. H. Lashkari, M. S. I. 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), pp. 407-414, 2016.

[4] P. Velan, M. Cermak, P. Celeda and M. Drasar, ”A survey of methods for encrypted traffic classification and analysis”, International Journal of Network Management, vol. 25, no. 5, pp. 355-374, 2015.

[5] E. Biersack, C. Callegari and M. Matijasevic, Data traffic monitoring and analysis. Berlin: Springer, 2013.

[6] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. Informatica (Slovenia), 31(3):249268, 2007.

[7] M. Finsterbusch, C. Richter, E. Rocha, J. A. Muller and K. Hanssgen, ”A Survey of Payload-Based Traffic Classification Approaches”, Communications Surveys & Tutorials IEEE, vol. 16, no. 2, pp. 1135-1156, 2014.

[8] Bacquet C, Zincir-Heywood AN, Heywood MI. An investigation of multi-objective genetic algorithms for encrypted traffic identification. In Computational Intelligence in Security for Information Systems, Advances in Intelligent and Soft Computing, vol. 63, Springer Berlin Heidelberg, 2009; 93100.

[9] Xie, G., Iliofotou, M., Keralapura, R., et al.: SubFlow: towards practical flow-level traffic classification. In: IEEE INFOCOM, pp. 25412545 (2012)

[10] Wright, C., Coulls, S., Monrose, F.: Traffic morphing: an efficient defense against statistical traffic analysis. In: The 14th Annual Network and Distributed Systems Symposium (2009)

[11] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, ”A detailed analysis of the KDD CUP 99 data set”, Proc. 2009 IEEE Int. Conf. Comput. Intell. Security Defense Appl., pp. 53-58.

[12] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.

[13] Rumelhart, D. E., G. E. Hinton, and R. J. Williams, Learning Internal Representations by Error Propagation, Parallel Distributed Processing, 1, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA (1986).

[14] Sherstinsky, Fundamentals of recurrent neural network(rnn) and long short-term memory (lstm) network, ArXivabs/1808.03314

[15] A. Paszke, S. Gross, S. Chintala, and G. Chanan.Pytorch: Tensors and dynamic neural networks inpython with strong gpu acceleration.PyTorch:Tensors and dynamic neural networks in Python withstrong GPU acceleration, 6, 2017.

[16] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprintarXiv:1412.6980 (2014)

[17] P. H. Zadeh, R. Hosseini, and S. Sra, Deep-rbf networks revisited: Robust classification with rejection,arXiv preprint arXiv:1812.03190, 2018