Byte Segment Neural Network for Network Traffic Classification

Rui Li, Xi Xiao, Shiguang Ni, Haitao Zheng, Shutao Xia
Graduate School at Shenzhen
Tsinghua University
Shenzhen, China
lir16@mails.tsinghua.edu.cn, {xiaox, ni.shiguang, zheng.haitao, xiast}@sz.tsinghua.edu.cn

***Abstract*—Network traffic classification, which can map network traffic to protocols in the application layer, is a fundamental technique for network management and security issues such as Quality of Service, network measurement, and network monitoring. Recent researchers focus on extracting features for traditional machine learning methods from flows or datagrams of the specific protocol. However, as the rapid growth of network applications, previous works cannot handle complex novel protocols well. In this paper, we introduce the recurrent neural network to network traffic classification and design a novel neural network, the Byte Segment Neural Network (BSNN). BSNN treats network datagrams as input and gives the classification results directly. In BSNN, a datagram is firstly broken into serval byte segments. Then, these segments are fed to encoders which are based on the recurrent neural network. The information extracted by encoders is combined to a representation vector of the whole datagram. Finally, we apply the softmax function to use this vector for predicting the application protocol of this datagram. There are several key advantages of BSNN: 1) no need for prior knowledge of target applications; 2) can handle both connection-oriented protocols and connection-less protocols; 3) supports multi-classification for protocols; 4) shows outstanding accuracy in both traditional protocols and complex novel protocols. Our thorough experiments on real-world data with different protocols indicate that BSNN gains average F1-measure about 95.82% in multi-classification for five protocols including QQ, PPLive, DNS, 360 and BitTorrent. And it also shows excellent performance for detection of novel protocols. Furthermore, compared with two recent state-of-the-art works, BSNN has superiority over the traditional machine learning-based method and the packet inspection method.**

***Index Terms*—recurrent neural network, traffic classification, network management, network measurement.**

I. Introduction

Traffic classification, which can classify network traffic to network protocols in application layers, is a basic module in the field of network management and security [1]. It plays a very significant role in many areas, such as Quality of Service (QoS), network measurement, intrusion detection and prevention systems. Hence, many researchers have paid their attention to this problem [2]–[5]. Traditional methods solve this problem by using protocol port information or protocol specifications [6], [7]. However, with the increase of new applications which do not have a fixed port number and well-documented format, traditional methods cannot accomplish traffic classification tasks anymore.

In this condition, recent researchers try to apply machine learning methods to this issue [8]. By using machine learning techniques, unique patterns of specific protocols are mined in datagram or flow level. Then, these patterns can be used to create a classifier for traffic classification. Various supervised and unsupervised methods are utilized in recent year [8], and they have achieved high accuracy in many network protocols. However, the exponential growth of network applications brings more protocols whose features are hard to be extracted by traditional machine learning methods. Moreover, the huge number of new protocols makes it difficult to find a series of features which can work for all of them.

In general, although machine learning methods have made a great contribution to traffic classification, there are still two unsolved challenges. One is that the number of new applications is enormous, and it is impossible to find a series features which can work on all of them. The other is the modern applications are so complex that traditional mining methods cannot achieve the pattern of specific protocol accurately.

In this paper, we try to apply some deep learning methods to overcome the above challenges. Deep learning methods have developed quickly and draw a plenty of attention in different areas [9], such as computer vision, natural language processing, machine translation and etc. Some deep neural networks show impressive performance for dealing nature language. Since the network datagram can be seen as a kind of character language between network applications, neural networks might have good performance on traffic classification too. Motivated by the above idea, we present a novel deep learning neural network, the Byte Segment Neural Network (BSNN), for traffic classification in this paper. The byte segment network is based on Recurrent Neural Network (RNN), which is expert in dealing with sequences. In BSNN, a network datagram is divided into some short segments at first, because the different part of the datagram may have different importance in traffic classification. Then these short segments are feed into two-level attention encoders which is made of RNN unit with attention layers. At last, BSNN gives output to predict label

This work is supported by the NSFC projects (61773229, 61771273, 61202358), the National High-tech R&D Program of China (2015AA016102), the RD Program of Shenzhen (JCYJ20160531174259309, JCYJ20170307153032483, JCYJ20160331184440545, JCYJ20170307153157440).

978-1-5386-2542-2/18/$31.00 ©2018 IEEE

of the datagram. Furthermore, in BSNN, the Focal Loss [10] is used as the loss function to counter the quantity imbalance and unequal training difficulty of different protocols.

We do extensive experiments on a traffic dataset with datagrams caught from the real world including ten protocols. BSNN shows high accuracy in both traditional and new application protocols and has acceptable classifying speed. In addition, we have some comparison with some traditional machine learning methods, our method shows advantages on testing protocols.

The key contributions of this work are summarized as follows:

- We apply recurrent neural network to traffic classification issue and discuss its feasibility. To the best of our knowledge, this is the first attempt in this field.
- We design a novel deep learning network, the Byte Segment Neural Network, for network datagrams, which has a reasonable structure based on the characteristic of datagrams.
- Our system can classify single packet without assembling IP packets into flows. And it is capable of unbalanced multi-classification tasks.
- The experiments are carried out in both traditional and new application protocols. Besides, we have some comparison with the state-of-the-art method in our experiments.

The rest of this paper is organized as follows: Section II discusses related work; Section III describes the proposed method; Section IV gives our experiment results, and we conclude this paper in Section V.

II. Related Work

Previous works for network traffic classification have involved many different techniques in this area. In this section, we focus on methods that apply machine learning approaches. According to classification levels, these methods can fall into two categories: 1) flow-level methods, 2) datagram-level methods. Besides, network traffic is a kind of character sequence which is used to send messages between network applications. In this condition, the network protocol can be seen as languages between applications, which means we can learn some ideas from some classic applications of deep learning in natural language classification. Thus, we introduce some methods about the application of deep learning in natural language classification area.

A. Flow-level Methods

The flow-level methods use the information of traffic flow as input. The features and the behavior of the entire network flow are used for classification. There are many flow-level methods based on cluster techniques. Roughan et al. used Nearest Neighbour, Linear Discriminate Analysis, and Quadratic Discriminant Analysis methods in [11]. Features are collected from different levels of traffic and calculated on full flow. Bernaille et. al observed that the size and the direction of the first few packets of the TCP connection are very considerable. According to this, they proposed a model based on simple K-Means [12]. Besides, McGregor et al. proposed a method [13] which applied Expectation Maximization (EM) algorithm. The packet length statistics, byte counts, and some connection and flow information are input to the clustering algorithm.

Zhang et al. have presented several approaches [14]–[17] to doing flow-level traffic classification, include some Bayesian and clustering techniques. The TCC [14] involved flow correlated information into clustering methods. And the work [15] utilized flow correlation information to solve the unknown applications problems with flow label propagation and compound classification techniques. Moreover, works [16] and [17] tried to combine flow-level features and datagram-level features as input for supervised or unsupervised machine learning methods.

Besides ordinary methods, Auld et al. put forward a Bayesian Neural Networks for flow-level traffic classification with the input of a series of flow-level features in [18]. Their work indicates that deep learning methods can be applied in network traffic classification.

Although these methods achieved good results in experiments, it is also shown that they all require the network flow information, which need to assemble IP packets into application-level messages.

B. Datagram-level Methods

The datagram-level methods identify network traffic using datagram information. With the help of machine learning methods, automatically abstracting protocol features comes to reality. Haffner et al. found out that the first n-bytes datagram of a TCP unidirectional flow can be used to automatically extract signatures. Based on this, they applied three machine learning algorithms in their method ACAS [19]. Similarly, Ma et al. also focused on the feature of the first n-bytes datagram. They used unsupervised learning methods including product distribution and Markov processes to identify traffic pertaining to the same protocol in [20]. Moreover, their method took the network traffic issued by one unknown application into account and distinguished it from that by known applications.

Finamore et al. raised an approach Kiss [21], [22] for UDP traffic, which identifies the traffic by automatically extracting payload signatures. And the decision engine is supported by Support Vector Machines (SVM). They are the first researchers who defined the concept of Stochastic Packet Inspection (SPI). Later Mantia et al. extended SPI to support classification for TCP in work [23].

Deri et al. developed nDPI [24], an implementation of Deep Packet Inspection (DPI). By using both packet header and payload, DPI methods can perform string detection in streaming data without degrading network throughput. nDPI can support over 500 different protocols. It has achieved high accuracy and efficiency, but the prior knowledge of protocols must be prepared in nDPI.

Recently, Yun et al. presented Securitas [25], which used Latent Dirichlet Allocation (LDA) and the word bag model to get the feature of protocols. Then, a classifier using traditional machine learning methods such as SVM, C4.5 decision tree and Bayes Network were constructed.

Compared with flow-level methods, datagram-levels methods involve more detailed traffic information. However, since it is very difficult to extract features for traditional machine learning methods and the features are various in different applications, we apply deep learning methods to design new datagram-level traffic classification methods without feature extraction.

C. Deep Learning in Natural Language Classification

As we discussed above, network traffic classification has many similarities with natural language classification. As the rapid development of deep learning and neural networks, many researchers paid their attention to apply deep learning techniques to natural language classification problem.

Blunsom et al. proposed DCNN [26], which used a convolutional neural network to represent sentences. DCNN processes input sentences of different lengths and introduces feature maps on sentences that clearly capture short-term and long-term relationships. Moreover, it does not rely on parse trees and is suitable for any language.

Lai et al. introduced a recurrent convolutional neural network [27] for text classification. A recurrent structure is used to capture contextual information for introducing less noise than traditional neural networks. In addition, this work can automatically judge which words are more important in text classification.

Lin et al. applied recurrent neural network to capture the coherence between sentences in a document and proposed HRNNLM [28] for document modeling. With a two-step training process, the language models are convergent in both the sentence level and the word level.

The model HAN [29] proposed by Yang et al. employed a hierarchical structure to mirror the document structure. And it has two attention layers, which can make sure that every sentence and word has different importance when constructing the document representation.

All of these works achieve a good performance in natural language classification. Motivated by this, we design BSNN for network traffic classification.

III. Proposed Method

In this section, we provide the architecture of byte segment neural network in detail. The overall structure of BSNN can be found in Fig. 1. In our system, every single datagram is fed into segments generator and broken into a series of integer segments. Then, the attention encoder transforms every segment to a representation vector. After this, these vectors are combined by a representation of the whole datagram in another attention encoder. Finally, the representation of the datagram is introduced to a softmax function to calculate the predicted class of the datagram. For the remainder of this section, we explain the mechanism of RNN units applied in our work at first. Then, we introduce every part of BSNN. At last, we discuss the loss function and the classification approach.

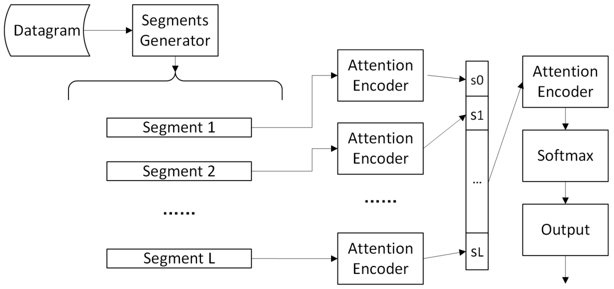


Fig. 1. Overall structure of BSNN

A. RNN Units for Encoder

We apply the recurrent neural network to our attention encoder, which can generate a representation of a sequence. Since the ordinary RNN suffers from gradient vanishing, we involve two RNN units in our encoder: the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

1) LSTM-Based Encoder: LSTM uses a gating mechanism, which can preserve the important features during long-term propagation. There are three gates in LSTM: i, f and o represent the input, forget and output gates. These gates control the updating style of the unit.

At each time t, LSTM holds a memory c_t. The output of the unit h_t is calculated as:

$$h_t = o_t \tanh c_t \tag{1}$$

where o_t is the output gate at time t, which adjusts the amount of memory content output. o_t can be updated like this:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t) \tag{2}$$

where σ is a sigmoid function. x_t denotes the input vector. The memory c_t is updated by forgetting the old memory and adding the new memory $\hat{c_t}$:

$$c_t = f_t c_{t-1} + i_t \hat{c}_t \tag{3}$$

The new memory is:

$$\hat{c_t} = \tanh(W_c x_t + U_c h_{t-1}) \tag{4}$$

In 3, f_t denotes the forget gate which controls how much old memory should be forgotten. And i_t represents the input gate, it decides the amount of new memory that can be added to the memory cell. They can be updated by these:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1}) \tag{5}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1}) \tag{6}$$

2) GRU-Based Encoder: GRU was proposed in [30]. Similar with LSTM, GRU also adopts gates mechanism. While GRU has just two gates and does not have the memory cell independently. z and r denote the update gate and the reset gate.

The output of GRU at time t is denoted by h_t, which can be computed as:

$$h_t = (1 - z_t) h_{t-1} + z_t \hat{h_t} \tag{7}$$

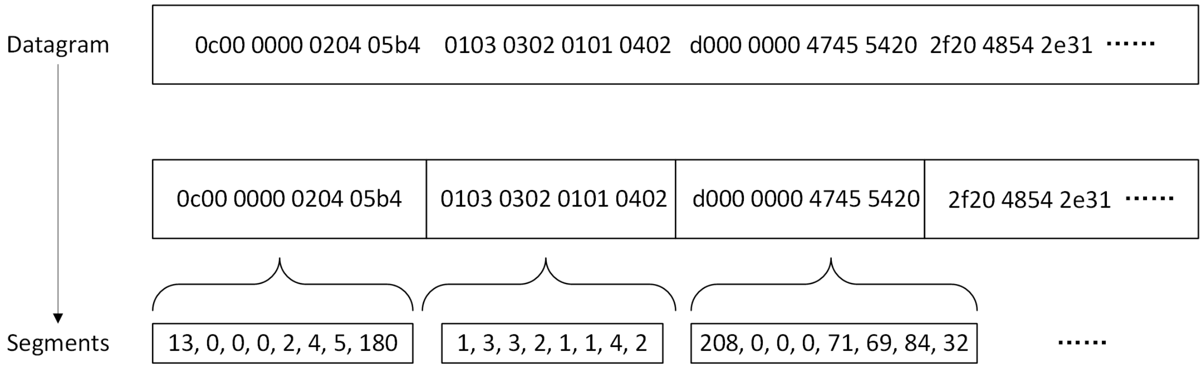


Fig. 2. An example of segments generation

where z_t is the update gate at time t. $\hat{h_t}$ represents the candidate state of the output. The update gate plays a role like the input gate and the forget gate in LSTM, it controls the keeping of the old memory and the adding of new memory. z_t is updated like:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{8}$$

And the candidate state $\hat{h_t}$ is:

$$\hat{h_t} = \tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \tag{9}$$

where r_t is a set of reset gates, which describes how many contributions the old memory does to the candidate state. Similarly, r_t can be computed as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{10}$$

B. Segments Generator

Segments generator is designed to provide a simple representation of a raw network datagram. To this end, it transforms the raw network datagram into serval fixed-length segments. The segment is a vector which consists of integers, which is the input of BSNN.

Packet header information can make some contributions to traffic classification. However, it can be easily forged and even packets from the same application may have different packet header information. To avoid the effect of the packet header information, we remove all the packet header of the datagram, including the Ethernet header, the IP header, and the TCP header. Only the payload can be transformed to segments.

Network datagrams have different types such as text and binary, but we read all the datagrams in binary type for convenience. In segments generator, a byte is transformed to an integer between 0 to 255. Besides, a datagram is divided into segments with fixed length N. Let L denotes the total number of segments from a datagram. A sample of the processing of segments generation is shown in Fig.2.

The first line of Fig.2 is a datagram in hexadecimal, and two numbers represent a byte. At first, the datagram is broken into short segments. Then, a byte is transformed to an integer as the figure shows. For the sake of simplicity, we use the word character represents a byte data.

C. Attention Encoder

The attention encoder, which employs RNN units to combine the contextual information of every character, can achieve a reasonable representation of an input sequence. Moreover, it uses an attention layer to find out which part is important to the whole sequence.

The collection of the whole datagram can be indicated as $A = \{\{a_{i,n}\}_{n=1}^{N}\}_{i=1}^{L}$, where $a_{i,n}$ means the n-th character in segment i. For applying RNN units to datagrams, we need to embed characters to a vector. As we describe above, the character $a_{i,n}$ is an integer between 0 to 255, we can transform it to a one-hot vector in 256-dimension. Let $x_{i,n}$ be the one-hot vector of $a_{i,n}$. For getting information from both directions of the segments, we employ a bidirectional RNN structure which proposed in [31]. The bidirectional RNN structure contains two RNN units, one reads the segment from $x_{i,1}$ to $x_{i,N}$ and the other reads from $x_{i,N}$ to $x_{i,1}$. It can be described in the formula:

$$\overrightarrow{o_{i,n}} = \overrightarrow{RNN}(x_{i,n}), n \in [1, N] \tag{11}$$

$$\overleftarrow{o_{i,n}} = \overleftarrow{RNN}(x_{i,n}), n \in [N, 1] \tag{12}$$

where $RNN()$ means the RNN unit we used, i.e., GRU or LSTM. After this, a high level representation of a character $a_{i,n}$ can be created by combining the forward output $\overrightarrow{o_{i,n}}$ and the backward output $\overleftarrow{o_{i,n}}$. The final output $o_{i,n}$ can be obtained by simply connecting the two direction outputs like: $o_{i,n} = [\overrightarrow{o_{i,n}}, \overleftarrow{o_{i,n}}]$.

Now we have outputs for every character in a segment and they will be combined to a segment vector which is the representation of the whole segment. Since different character may have different importance in traffic classification, we apply attention mechanism to integrate the outputs. The attention mechanism is from a work in natural language document classification [29]. First, we need to calculate the weight $w_{i,n}$ for every character in segment i.

$$w_{i,n} = \frac{\exp(h_{i,n}^{\mathrm{T}} h_c)}{\sum_{n=1}^{N} \exp(h_{i,n}^{\mathrm{T}} h_c)} \tag{13}$$

where $h_{i,n}$ is:

$$h_{i,n} = \tanh(W_h o_{i,n} + b_n) \tag{14}$$

TABLE I
DATA DETAILS

Protocols	DNS	BitTorrent	PPLive	QQ	STMP
Number[a]	2516	11581	24527	5072	2830
Size(Kb)	473	8514	15264	2913	494
Protocols	**360**	**Amazon**	**Yahoom**	**Cloudmusic**	**foxmail**
Number[a]	20957	7598	37895	55576	7300
Size(Kb)	16035	5384	17306	48050	3590

[a] Number of Datagrams

By a multilayer perceptron of one layer, the hide state $h_{i,n}$ of character n can be achieved. Then, the weight of character n is set as the similarity of $h_{i,n}$ and the context vector h_c. The context vector is proposed in [32], which is an abstract representation of the most informative character in the segment. And it can be measured during the training process of the neural network. Finally, the representation of the segment is:

$$s_i = \sum_{n=1}^{N} w_{i,n} o_{i,n} \tag{15}$$

In this way, we can get representations of all the segments. Let $S = \{s_i\}_{i=1}^{L}$ be the collection of segment representation of a datagram. Note that S is a series of vector just like $\{a_{i,n}\}_{n=1}^{N}$, which means we can feed it to another attention encoder. After this, we can get a representation vector d of the whole datagram.

D. Focal Loss and Classification

As we already have the representation d of a datagram, we can turn to the classification problem. For K classification, d can be transformed to a K-dimension vector $y = W_x d + b_x$.

For training loss, we involve the focal loss in BSNN, because it shows strong advantages on the imbalance multi-classification problem [7]. The key idea of the focal loss is that the instances which are hard to train are significant to the whole neural network. The focal loss for multi-classification can be simply calculated like this:

$$FL(y_l) = -\alpha(1 - softmax(y_l))^{\gamma} \log(softmax(y_l)) \tag{16}$$

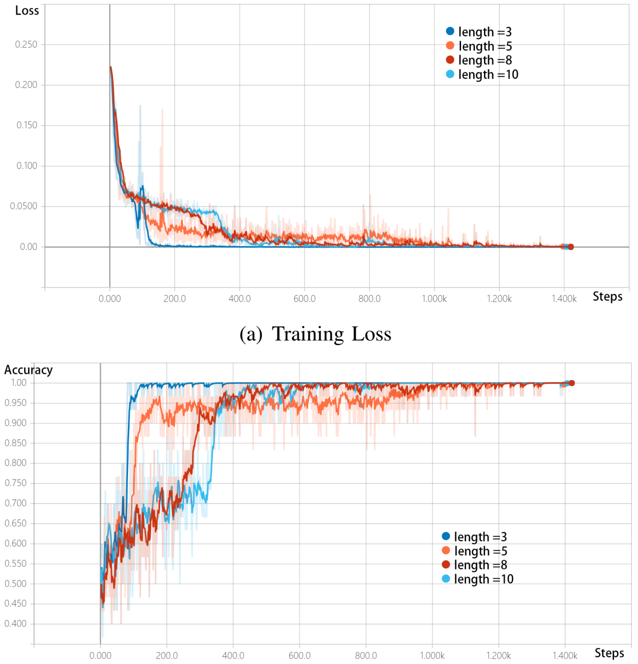
where l is the true label of the datagram. α is the class weight, and γ is the focusing parameter. The softmax, i.e., the normalized exponential function, is:

$$softmax(y_j) = \frac{\exp(y_j)}{\sum_{k=1}^{K} \exp(y_k)} \qquad j \in [1, k] \tag{17}$$

where K is the total number of classes. Furthermore, we provide the backward gradient as follows:

$$\frac{\partial FL(y_l)}{\partial y_i} = \begin{cases} -\alpha(1 - G_l)^{\gamma}(1 - G_l - \gamma G_l log(G_l)) & i = t \\ \alpha G_i (1 - G_l)^{\gamma - 1}(1 - G_l - \gamma G_l log(G_l)) & i \neq t \end{cases} \tag{18}$$

where $G_l = softmax(y_l)$.



(a) Training Loss

(b) Training Accuracy

Fig. 3. Training Performance of Different Segment Lengths

IV. EXPREMENTAL EVALUATION

To test the performance of BSNN, we provide the experimental results and evaluation of BSNN in this section. We carry on our experiment on a datagram dataset caught from the real world. Firstly, we give the description of the dataset and the metrics. Then, the selection of parameters of BSNN is shown. After this, the experiment performances in a multi-classification problem and comparison with the traditional methods are presented. The detection performance for some novel protocols in real network and the discussion of classification speed is provided at last.

A. Dataset

Ten different traditional and novel applications are involved to our experimental dataset, including DNS, BitTorrent, PPLive, QQ, STMP, 360, Amazon, Yahoom, Cloudmusic and foxmail. Since traffic classification is mapping traffic to protocols in the application layer, the word protocol means protocol in the application layer in this section. Note that these target protocols include both connection-oriented protocols (TCP) and connection-less protocols (UDP). Besides, both text type and binary type payload protocols exist in our dataset.

The datagrams of a specific protocol are caught by tracing the corresponding application processes in the real network environment. In this way, we obtain a dataset which is big enough for training and test BSNN. The number and the size of datagrams in each protocol can be found in Table I.

In particular, we have eight connection-oriented protocols, including BitTorrent, QQ, STMP, 360, Amazon, Yahoom, Cloudmusic and foxmail, and two connection-less protocols, including PPLive and DNS. Note that our dataset contains not only traditional protocols like BitTorrent, DNS and SMTP but also some novel applications like PPLive, 360, QQ, Amazon, Yahoom, Cloudmusic and foxmail. Thus, we can evaluate the effectiveness of our method comprehensively. The traditional protocols we chose are well-known, which have been applied to file sharing, domain name services, and e-mail communication. What's more, they have been used in experiments of previous works. While the novel protocols are responsible for audio and video, safety and social network, they are more complex compared with the traditional protocols.

We apply 5-fold cross-validation of the dataset in our experiment. In the training phase, a batch, which consists of thirty datagrams, is used for the backpropagation. The training of a batch is described as a step in the following parts. Besides, an epoch means training with all the batches one time.

B. Metrics

There are several metrics for evaluating our method. Before their definitions, we first introduce four numbers. For each protocol in the test set, True Positives (TP) means the number of datagrams classified to the specific protocol which indeed belong to the protocol. Let True Negatives (TN) be the number of datagrams that are judged as not belonging to the specific protocol and actually not issued by it. False Positives (FP) refers to the number of datagrams which is incorrectly classified to the specific protocol. And let False Negatives (FN) be the number of datagrams classified to the other protocols but actually generated by the specific application.

For evaluation of the test dataset, we define $Precision$, $Recall$ and $F1-measure$ for every target protocols:

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

$$F1-measure = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{21}$$

In order to analyze the training phase of BSNN, we use the training loss and the global training accuracy. The training loss is defined in equation 18, and the global accuracy is defined as :

$$Accuracy = \frac{\sum_{i=1}^{K} TP[i]}{Total} \qquad i \in [1, K] \tag{22}$$

where K is the number of classes and $Total$ is the total number of instances in the training set. Besides, $TP[i]$ means the TP of class i.

C. Parameter Selection

There are several important parameters in BSNN such as the segment length N, the Hyperparameters α and γ for the focal loss, and parameters for the training of the neural network. We provide the selection and setting of parameters in this part.

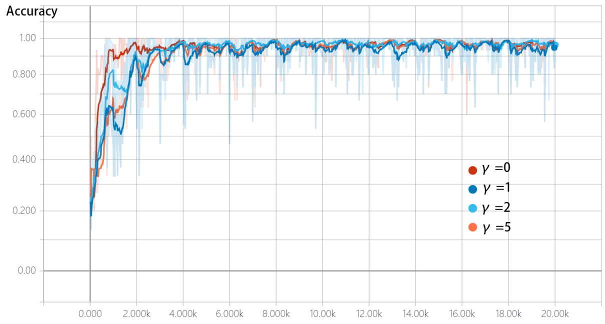


Fig. 4. Training Accuracy of different γ

1) Segment Length N: The segment length is very significant for BSNN, as it can influence the speed and effectiveness of the neural network. Thus, we carry on some experiments to find the proper N. The experimental dataset contains two protocols, DNS and STMP. We create different training and test sets with segment length $N = 3$, 5, 8 and 10, and the model is trained by 20 epochs. The RNN unit we used in this part is GRU. BSNN achieves nearly 100% test accuracy on all the dataset.

The training loss and the training accuracy with different segment length are shown in Fig. 3(a) and Fig. 3(b).The abscissa of the two figures are the training steps. Based on Fig. 3(a), when segment length N increase from 3 to 10, the speed of loss reducing becomes slow. While this difference in reducing speed becomes smaller when N is big enough. Meanwhile, when N is small, it is obvious that the loss curve is more unstable than that with bigger N. Unstable loss curve indicates the danger of trap in local optimal.

The training accuracy depicted in Fig. 3(b) shows that the increasing speed of accuracy comes faster when N is getting bigger. Besides, the varying of the difference of increasing speeds also has the similar trend as the loss figure. The increasing of accuracy comes faster when N becomes smaller. After step 400, all the accuracy curves turn to stable, and the curves with $N = 8$ and $N = 10$ show better performance than that with $N = 5$.

Moreover, considering that short segments lead to more segments which bring more cost in the attention layer and long segments make the encoding time longer, we choose $N = 8$ in following experiments.

2) Hyperparameters for Focal Loss: There are two Hyperparameters for focal loss, α and γ, and α refers to the class weight, which can be calculated by the percentage of every class. Thus, we focus on the selection of γ in this part. γ is the focusing parameter, and bigger γ will let the neural network more focusing on the instances that are hard to train. Generally, γ should be a positive number, and when $\gamma = 0$, the focal loss becomes the ordinary cross-entropy.

We have done some experiments for the selection of γ on an imbalance traffic classification problem with five protocols (QQ, PPLive, DNS, 360 and BitTorrent). Furthermore, we applied GRU and LSTM in different experiments. γ is set

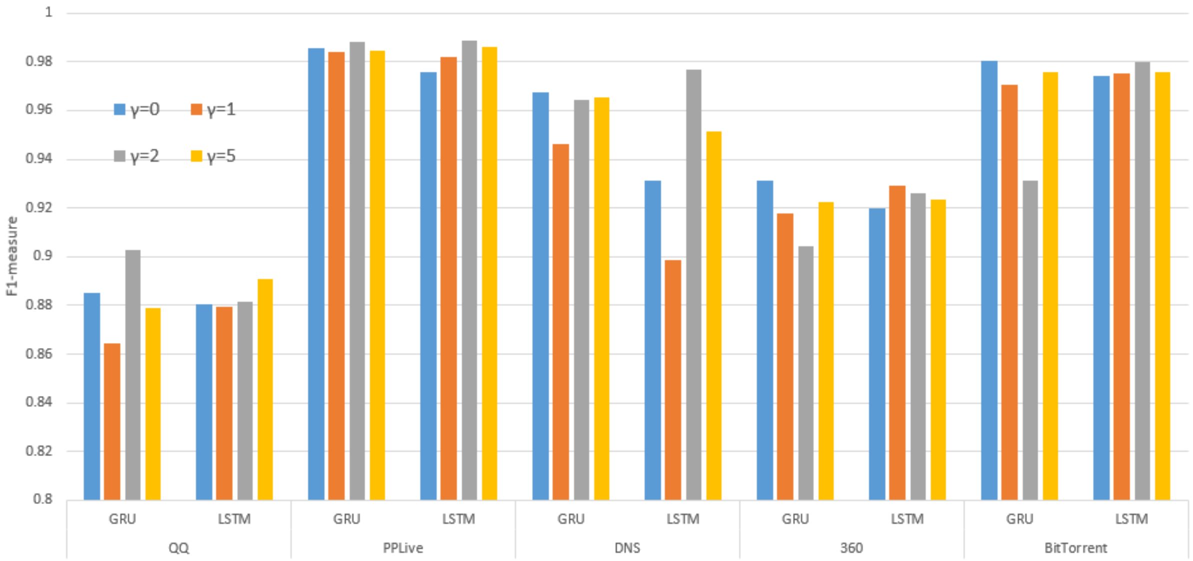


Fig. 5. Test Performances of Different γ

as $0, 1, 2, 5$ and the model is trained by 20 epochs.

Note that models with different γ calculate the training loss in different ways, which demonstrates the comparison of the training loss makes no sense. In this condition, we focus on the training accuracy and the test performances of different γ.

Fig.4 shows the training accuracy of different γ, and these experimental models use GRU unit. Since the accuracy curve figure of models with LSTM unit is similar with Fig.4, we omit it for the sake of simplicity. The accuracy with $\gamma = 0$ has the fastest increasing speed. While as discussed above, when $\gamma = 0$, the focal loss is an ordinary cross-entropy and does not have the ability to overcome the imbalance problem or focus on important instances. Besides, the curve with $\gamma = 2$ is better than those with $\gamma = 1$ and $\gamma = 5$. All the models reach global optimal in 2000 steps, so we should check out the test performances to select a proper γ.

The test performances of different γ are provided in Fig. 5. we use F1-measure to judge the classifying results. GRU and LSTM in abscissa represent BSNN models with GRU or LSTM unit. It is easy to find out that the performances of the model with a fixed γ rely on the target protocols. But $\gamma = 2$ shows best results with regard to the overall performances. Therefore, we choose $\gamma = 2$ in our following experiments.

3) Hyperparameters for Neural Networks: For training BSNN, there are some more hyperparameters should be set. The embedding dimension and the dimension of GRU and LSTM are set as 100. Thus, the output size of the attention encoder is 100. Besides, we use a learning rate of 0.001 and a dropout rate of 0.5.

In the training phase, the size of a batch is set as 30, i.e., BSNN uses 30 datagrams in one training step. And we find out from Fig.3 and Fig.4 that training with 20 epochs is enough for BSNN to achieve global optimal solutions.

D. Multi-classification Comparison

We provide the test effectiveness results of BSNN and the comparison with Securitas and nDPI in this part. Securitas was proposed in 2016 [25] and it is one of the best solutions in the field of traffic classification. Securitas has an impressive performance and it classifies single packet just like our method. nDPI is an implementation of Deep Packet Inspection (DPI) proposed in 2014 [24] which is a classic traffic classification method with high protocol detection accuracy and efficiency. What's more, nDPI is also focused on analyzing information from packets. Thus, we choose Securitas and nDPI for comparison in this part.

Securitas is a datagram-level traffic classification method. This method uses the Latent Dirichlet allocation topic model to extract features of a specific protocol, then feeds these features to traditional supervised machine learning methods such as SVM, C4.5 Decision Tree, and Bayes Network. Securitas is a classic traffic classification method with traditional machine learning, which is dedicated to find proper features and use these features to construct machine learning classifier.

nDPI is based on a traffic classification library using both packet header and payload. Before classifying a specific protocol, the corresponding code must be added to DPI library. For every traffic flow, nDPI will check every protocol in the library to classify it. And it is a classic representation of traffic classification methods which use prior information of protocols.

1) Experiment Setup: In this experiment, we use a dataset mixed with five different protocols (QQ, PPLive, DNS, 360 and BitTorrent). In the test phase, our method takes a datagram as input and outputs its class label. As there are five classes in our experiment, our method can directly classify datagrams to one of them.

Besides, we set parameters of Securitas as the same as

TABLE II
Experimental Results and Comparison

Protocols	QQ			PPLive			DNS			360			BitTorrent		
Metrics	$R.\%$	$P.\%$	$F1.\%$	$R.\%$	$P.\%$	$F1.\%$	$R.\%$	$P.\%$	$F1.\%$	$R.\%$	$P.\%$	$F1.\%$	$R.\%$	$P.\%$	$F1.\%$
nDPI	67.1	82.1	73.8	90.4	100	94.9	76.7	93.8	84.4	-	-	-	81.55	100	89.83
Securitas-SVM	42.2	97.9	59.0	60.6	99.4	75.3	-	-	-	76.0	53.0	62.4	69.2	93.0	79.4
Securitas-C4.5	82.8	99.2	90.3	99.0	99.7	**99.4**	99.0	97.4	**98.2**	64.6	94.9	76.9	96.0	97.9	96.9
Securitas-Bayes	81.2	87.1	84.0	98.0	99.1	98.6	99.8	92.4	96.0	96.3	47.5	63.6	91.6	84.1	87.7
BSNN-GRU	83.5	99.1	90.6	98.0	99.8	98.9	98.8	96.2	97.5	96.7	89.8	93.1	97.7	96.1	96.9
BSNN-LSTM	83.4	99.3	**90.7**	97.8	99.9	98.9	100	95.1	97.5	98.6	89.6	**93.9**	97.6	98.5	**98.1**

suggested in [25]. Note that Securitas is a binary classifier which can only identify a specific protocol. Thus, we build five different Securitas models for every target protocol. Generally, multi-classification is more difficult than binary-classification on the same data. Thus, the comparison results of BSNN and Securitas are convincing.

nDPI can support 512 different protocols now, but this number is still too small for the enormous network applications. In our experiment, nDPI cannot deal with 360 protocol, so there are only experiment results of the other protocols.

The parameters of BSNN is set as we show in former parts. BSNN-GRU means the attention encoder is embedded with GRU unit. And BSNN-LSTM refers to the attention encoder is embedded with LSTM unit.

2) Experimental Results: The experimental results are shown in Table II. And metrics $R.$, $P.$ and $F1.$ mean $Recall$, $Precision$, and $F1 - measure$. Securitas-SVM, Securitas-C4.5 and Securtas-Bayes mean the Securitas classification methods with SVM, C4.5 decision tree and Bayes network.

Although nDPI gets high $Precision$ for some protocols, it is obvious that the overall performance of nDPI is far below Securitas and BSNN. Traditional methods do have disadvantages in accuracy with machine learning based methods. The comparison of Securitas and BSNN is provided in detail as follow.

For QQ protocol, BSNN-LSTM reaches the best $F1 - measure$. Note that the Securitas-C4.5 achieves high $Precision$ 99.2% which is similar with that of BSNN-LSTM and BSNN-GRU. While BSNN methods gain higher $Recall$ and $F1 - measure$.

For PPLive protocol, Securitas-C4.5 gains the best performance with high $F1 - measure$ 99.4%. Both BSNN-GRU and BSNN-LSTM reach the second highest $F1 - measure$ as 98.9%. Although the $Precision$ of BSNN methods nearly come to 100%, they have not got an excellent performance in $Recall$.

For DNS protocol, there are only four results, because the quantity of DNS datagrams in our dataset is very small, and Securitas-SVM classifies all the datagrams as negative, which makes no sense in our comparison. Securitas-C4.5 has the best result for DNS with $Recall$ 99.0% and $Precision$ 97.4%. Besides, BSNN-LSTM reaches a 100% $Recall$.

For 360 protocol, BSNN methods show significant advantages over Securitas methods in classification of 360 protocol. BSNN-GRU achieves $F1 - measure$ as 93.1% and BSNN-LSTM gains $F1 - measure$ as 93.9%. Meanwhile, the best performance of Securitas methods only has 76.9% $F1 - measure$.

For BitTorrent protocol, BSNN-LSTM has the most precise classification performance, it gets highest results in all the metrics. Securitas-C4.5 and BSNN-GRU also arrive at 96.9% in$F1 - measure$, but they still have 1.2% lesser than that of BSNN-LSTM. Comparing with other methods, BSNN-LSTM achieves the best $Precision$.

3) Discussion: Note that Securitas-SVM shows bad performance for all the target protocols. As we point above, instances from different protocols are quite unbalanced in our experiments when it comes to binary classification, and SVM is not good at dealing with imbalance classification problem.

BSNN methods have achieved high accuracy similar to Securitas for traditional protocols, i.e., PPLive, DNS and BitTorrent. Although Securitas is better for PPLive and DNS, the $F1 - measure$ distance between Securitas and BSNN is less than 1%. With regard to novel protocols QQ and 360, BSNN shows advantages over Securitas. Even though Securitas cannot achieve ideal results for 360 protocol, BSNN still gains outstanding performance. What's more, it is worth mentioning that Securitas is a binary classifier and we build five Securitas model for every target protocols. And BSNN can classify all the datagrams to five classes directly. That indicates if we combine Securitas as one multi-classification, BSNN will show more advantages over it. Besides, both Securitas and BSNN show better performance than traditional method nDPI.

The impressive performance of BSNN, which achieves $F1 - measure$ about 95.82%, is owing to the ability of deep learning methods to deal with complex sequences. Comparing with traditional methods and traditional machine learning

TABLE III
DETECTION PERFORMANCE

Protocols	$R.\%$	$P.\%$	$F1.\%$
Amazon	96.8	77.9	86.3
Yahoom	96.3	76.5	85.2
Cloudmusic	97.8	81.1	88.7
foxmail	97.4	93.8	95.6

methods, neural networks can model complex issues better.

E. Detection Performance for Novel Protocols

In this section, we test the detection performance of BSNN with five novel protocols, Amazon, Yahoom, Cloudmusic and foxmail. These protocols are from different application areas including web-application, instant messaging, music and mail service. The experiment target is to detect specific protocols from a set of traffic mixed with different unknown protocols. The parameters of BSNN are set as we discussed in the last section.

As shown in Table III, BSNN achieves $F1-measure$ more than 85% for all the target protocols. The performance of BSNN in detection problem is worse than that in the multi-classification problem (Table II). While in detection problem, $Recall$ is very significant because high $Recall$ can make sure every packet of target protocols can be detected. Note that BSNN reaches high $Recall$ over 96% for all protocols, which proves BSNN is a powerful tool for detection of complex novel protocols.

F. Classification Speed

In the online classification of network traffic, classification speed is an important index. Many researchers think that the deep learning methods suffer from the slow speed. Now, we provide the running speed of BSNN under the multi-classification experiment that we discuss in last part.

The device we used in our experiment is a GTX 1080 graphics card, and the CPU is the Intel Xeon E5-2630 v3 @2.40GHz. In the training phase, the average dealing time of a datagram is 43 millisecond (ms). And the total training time of 20 epochs costs 123 minutes.

In the testing phase, the average time to process a single datagram is $2.97ms$. And the average test dealing time of Securitas is $7.01ms$ [25]. Thus, the classifying speed of BSNN is similar to the traditional machine learning methods and is enough for the online classification.

V. CONCLUSION

In this paper, we propose a new neural network classifying system, the Byte Segment Neural Network, to address network traffic classification. This is the first application of the recurrent neural network in this area to the best of our knowledge. Our method is a multi-classification system and can directly map datagrams to application protocols. Besides, it can deal with protocols of different connection types and different payload types. Moreover, there is no need for prior information of the specific protocol such as protocol format and application codes in the training phase and the classification phase. We validate the effectiveness of BSNN with network traffic caught from the real world. The experimental results show that BSNN has an outstanding performance for all the target protocols. Moreover, the comparison with two state-of-the-art methods proves that our method does have advantages over methods based on traditional machine learning. In general, BSNN is an efficient multi-classifier for network traffic, and it supports both traditional protocols and novel complex protocols.

REFERENCES

[1] Cisco Systems, Inc., San Jose, CA, USA, “Cisco WAN and application optimization solution guide,” Tech. Rep., 2008 [Online].

[2] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Lee, “Internet traffic classification demystified: myths, caveats, and the best practices,” In Proceedings of the International Conference on emerging Networking EXperiments and Technologies, Madrid, Spain, p. 11, 2008.

[3] Y. Wu, G. Min, K. Li, B. Javadi, “Modeling and analysis of communication networks in multicluster systems under spatio-temporal bursty traffic,” IEEE Transactions on Parallel and Distributed Systems 23, 902-912, 2012.

[4] C. S. Sastry, S. Rawat, A. K. Pujari, V. P. Gulati, “Network traffic analysis using singular value decomposition and multiscale transforms,” Information Sciences 177, 5275-5291, 2007.

[5] T. Karagiannis, K. Papagiannaki, M. Faloutsos, “BLINC: multilevel traffic classification in the dark,” ACM SIGCOMM Computer Communication Review 35, pp. 229-240, 2005.

[6] Z. Li, G. Xia, H. Gao, Y. Tang, Y. Chen, B. Liu, J. Jiang, and Y. Lv, “NetShield: Massive semantics-based vulnerability signature matching for high-speed networks,” In Proceedings of the ACM SIGCOMM, pp. 279290, 2010.

[7] R. Pang, V. Paxson, R. Sommer, and L. Peterson, “Binpac: A yacc for writing application protocol parsers,” In Proceedings of the 6th ACM SIGCOMM Conf. Internet Meas., pp. 289300, 2006.

[8] T. T. T. Nguyen, G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” IEEE Communications Surveys & Tutorials 10, 56-76, 2008.

[9] J. Schmidhuber. “Deep learning in neural networks: An overview,” Neural networks 61, 85-117, 2015.

[10] TY Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. “Focal loss for dense object detection,” arXiv preprint arXiv:1708.02002, 2017.

[11] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification,” In Proceedings of the ACM/SIGCOMM Internet Measurement Conference (IMC) 2004, Taormina, Sicily, Italy, October 2004.

[12] L. Bernaille, R. Teixeira, K. Salamatian, “Early application identification,” In Proceedings of the 2006 International Conference on emerging Networking EXperiments and Technologies, Lisboa, Portugal, p. 6, 2006.

[13] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, “Flow clustering using machine learning techniques,” In Proceedings of the Passive and Active Measurement Workshop (PAM2004), Antibes Juan-les-Pins, France, suver-48, April 2004.

[14] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Yong Xiang, and Yong Guan. “Network traffic classification using correlation information,” IEEE Transactions on Parallel and Distributed systems 24, no. 1, 104-117, 2013.

[15] J. Zhang, C. Chen, Y. Xiang, W. Zhou, A. V. Vasilakos, “An effective network traffic classification method with unknown flow detection,” IEEE Transactions on Network and Service Management 10, 133-147, 2013.

[16] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang. “Internet traffic classification using constrained clustering,” IEEE transactions on parallel and distributed systems 25, no. 11, 2932-2943, 2014.

[17] J. Zhang, Y. Xiang, W. Zhou, Y. Wang, “Unsupervised traffic classification using flow statistical properties and IP packet payload,” Journal of Computer and System Sciences 79, 573-585, 2013.

[18] T. Auld, A.W. Moore, and S. F. Gull. “Bayesian neural networks for internet traffic classification,” IEEE Transactions on neural networks 18, no. 1, 223-239 ,2007.

[19] P. Haffner, S. Sen, O. Spatscheck, D. Wang, “ACAS: automated construction of application signatures,” In Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data, Philadelphia, PA, pp. 197-202, 2005.

[20] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. Voelker, “Unexpected means of protocol inference,” In Proceedings of the 6th ACM SIGCOMM on Internet measurement. Rio de Janeriro, Brazil: ACM Press, pp. 313326, October 2006.

[21] A. Finamore, M. Mellia, M. Meo, D. Rossi, “Kiss: Stochastic Packet Inspection,” International Workshop on Traffic Monitoring and Analysis, pp. 117-125. Springer, Berlin, Heidelberg, 2009.

[22] A. Finamore, M. Mellia, M. Meo, D. Rossi, “Kiss: Stochastic Packet Inspection Classifier for UDP Traffic,” IEEE/ACM Transactions on Networking (TON) 18, 1505-1515, 2010.

[23] G. L. Mantia, D. Rossi, A. Finamore, M. Mellia, M. Meo, “Stochastic packet inspection for TCP traffic,” In: 2010 IEEE International Conference on Communications (ICC), Cape Town, South Africa, pp. 1-6, 2010.

[24] L. Deri, M. Martinelli, T. Bujlow, et al. “ndpi: Open-source high-speed deep packet inspection,” 2014 IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), 617-622, 2014.

[25] X. Yun, Y. Wang, Y. Zhang, Y. Zhou, “A semantics-aware approach to the automated network protocol identification,” IEEE/ACM Transactions on Networking (TON) 24, 583-595, 2016.

[26] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” arXiv preprint arXiv:1404.2188, 2014.

[27] S. Lai, L. Xu, K. Liu, and J. Zhao. “Recurrent Convolutional Neural Networks for Text Classification,” In AAAI, vol. 333, pp. 2267-2273, 2015.

[28] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li. “Hierarchical recurrent neural network for document modeling,” In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 899-907, 2015.

[29] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. “Hierarchical attention networks for document classification,” In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480-1489, 2016.

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.

[31] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[32] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. “Ask me anything: Dynamic memory networks for natural language processing,” In Proceedings of the International Conference on Machine Learning, pp. 1378-1387, 2016.