An Online Network Traffic Classification Method Based on Deep Learning

Qing Liao
Beijing University of
Posts and Telecommunications
Beijing, China
Email: liaoqing@bupt.edu.cn

Tianqi Li
Beijing University of
Posts and Telecommunications
Beijing, China
Email: litq@bupt.edu.cn

Wei Zhang
CSSC (Zhe Jiang)
Ocean Techonolgy CO., LTD
Zhoushan, China
Email: zhangwei@shiplinker.com

***Abstract*—Traffic classification plays an important role in network traffic analysis. In this paper, an online network traffic classification method based on deep learning was proposed. By pre-training CNN and fine-tuning the fully connected layer, the complexity of model update is greatly reduced, thus enabling online learning. To deal with the stability-plasticity dilemma, we introduce a proficiency mechanism. The proposed method uses the raw binary data of packet header as input of the model instead of statistical features, which avoid complex feature selection process. And it can identify traffic by a small number of packets, which meets the needs of early stage identification. Besides, our model has the ability of online learning, and can adapt dynamically to the network environment. Experimental results show good performance of the proposed method in both accuracy and speed on traffic classification task.**

***Keywords*—Network traffic classification, Machine learning, Deep learning, Early stage, Online learning**

I. Introduction

Network traffic classification/identification is an essential technology for many fields. With the development of network, there have been three technologies for traffic classification. The first one is port-based, which identifies the applications by default port numbers. The port-based method is simple and fast, but it cannot provide reliable results as an application can use any port number instead of the default port, and more and more applications tend to use port obfuscation techniques to evade detection. To address this shortcoming, the Deep Packet Inspection (DPI) technology emerged. The DPI-based method is not affected by port obfuscation and gets excellent accuracy by analyzing internal patterns of the payloads. However, it can only identify the traffic whose signatures are available and can not deal with the encrypted traffic. Recently, an increasing number of researchers focus on the study of traffic classification based on Machine Learning (ML).

The ML-based method often only uses the features of packet header to classify traffic, and therefore can overcome the deficiencies of DPI. Multiple machine learning techniques were proposed to identify the network traffic. Most existing approaches are based on offline learning models. Due to the dynamic nature, the network context often changes and new applications may appear over time. The offline trained models cannot effectively adapt to these dynamic changes. In addition, many methods use the statistics of long-term or even entire flow to identify traffic, which also limits the practicality.

In this paper, we propose an Online Network Traffic Classification (ONTC) method based on Convolutional Neural Network (CNN) to identify the network flow. The proposed method learns the basic features of packet header through a pre-trained CNN, and learns how the basic features of different traffic are combined through an online adjusted fully connected layer. To solve the problem of catastrophic forgetting and the trade-off between stability and plasticity, we introduced a parameter in the online model, namely *"proficiency"*, which means *how skillful to identify a category*. Furthermore, our model classifies traffic by only a small number of packets in the early stage of a flow, which also improves the practicality of ML-based network traffic classification technology.

The rest of this paper is organized as follows: Section II briefly reviews the related works about traffic classification. Section III presents the ONTC method. Section IV describes the experiment and analyzes the results and finally, Section V provides the discussion and conclusions.

II. Related Work

The statistical-based method can overcome the shortcomings of port-based and payload-based methods, such as the problems of port obfuscation and traffic encryption [1]. The statistical-based methods often use ML to classify traffic.

Moore et al. [2] applied ML method to classify traffic in 2005. They used Naive Bayesian technology to classify 10 types of traffic data, and proposed 248 flow-level statistical features that can be used for traffic classification [3]. Since then, more and more ML models have been introduced into traffic classification tasks. Williams et al. [4] compared the performance of 5 statistical-based methods. And Kim et al. [5] evaluated a variety of traffic classification methods, including port-based CoralReef, behavior-based BLING and 7 ML-based methods. In recent years, deep learning has developed rapidly, and it has also been applied to the field of traffic classification. Lopez-Martin et al. [6] combined CNN and Recurrent Neural Networks (RNN), using the combination of models to identify traffic. In addition, identifying traffic at the early stage is more important. Chen et al. [7] studied the method based on Flexible

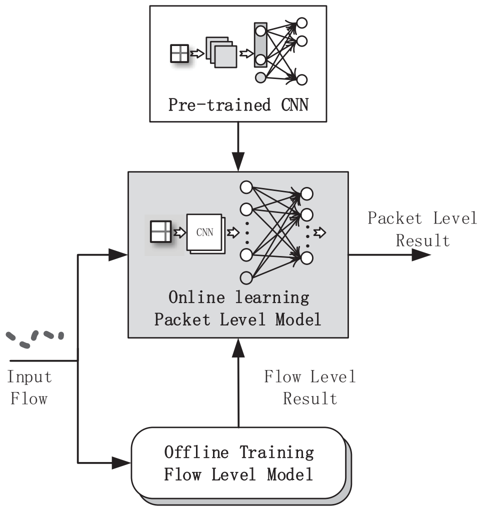
Pre-trained CNN
CNN
Online learning Packet Level Model
Packet Level Result
Input Flow
Flow Level Result
Offline Training Flow Level Model

Fig. 1. The structure of ONTC system.

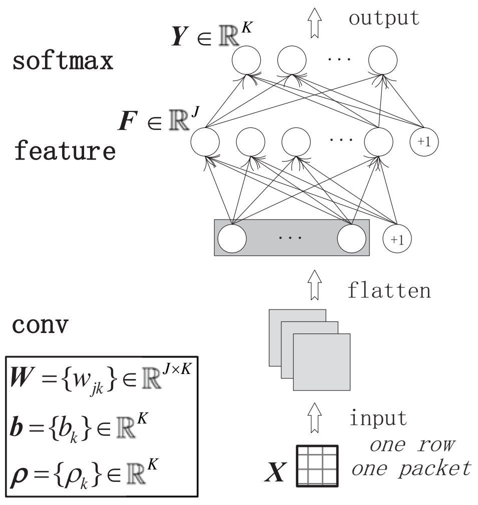
output
$Y \in \mathbb{R}^K$
softmax
$F \in \mathbb{R}^J$
feature
+1
+1
flatten
conv
$W = \{w_{jk}\} \in \mathbb{R}^{J \times K}$
$b = \{b_k\} \in \mathbb{R}^K$
$\rho = \{\rho_k\} \in \mathbb{R}^K$
input
one row
one packet
X

Fig. 2. The main model of ONTC method.

Neural Trees, which identifies traffic at the early stage by the first 10 packets of a flow.

Most ML-based methods take the flow statistics as features, so the selection of features is crucial, and it is difficult to determine whether they are optimal. In our work, we use the raw binary data of packet header as input of a CNN instead of statistical features. And our approach identifies traffic by a small number of packets, which meets the needs of early stage identification.

Another problem with existing methods is that the offline model cannot automatically adapt to the network context changes. Therefore, models with online learning capabilities are needed. Jain et al. [8] and Pérez-Sánchez et al. [9] respectively summarized the online learning of neural networks. In this paper, we propose an online learning method to improve the traffic identification capabilities. The detailed description of the ONTC method will be presented in the next section.

III. Online Network Traffic Classification Method

As illustrated in Fig. 1, an ONTC model is proposed to improve the performance of ML-based traffic classification, which combines the online learning model with an offline training model. The online model is a packet-level classifier, which identifies traffic with only a small amount of packets and more timeliness, while the offline model is a flow-level classifier that obtains the category results through the information of the entire flow and more accuracy. The online model takes the results of offline model as the ground truth for online learning, which achieves a perfect combination of timeliness, accuracy and adaptability. In the following, we describe the proposed method in detail.

A. Inspired by CNN for Images

Since Alex Krizhevsky [10] won the 2012 ImageNet competition, CNN shines in Computer Vision (CV) and many other fields. A CNN consists of a stack of layers, including convolutional (Conv) layers, nonlinear layers, pooling layers and Fully Connected (FC) layers. The very first Conv layer detects the lowest level features, and the deeper layer detects the higher level features. The FC layers combine the basic features extracted by Conv layers into different categories. The success of image caption, object detection and many other CV tasks shows that features extracted by CNN are highly abstract. Therefore, many tasks can be simplified by transfer learning (i.e. freeze the layers in front of CNN and fine-tune the last few layers).

CNN is powerful for images, and fortunately, we found similarities between traffic and images. As we all know, an grayscale image is a 2-D matrix of pixel values, and each element is in $[0, 255]$. For traffic or a flow, it consists of a series of packets, and each packet header contains a fixed number of bytes representing different field values (e.g. a packet header containing a frame header, an IP header and a TCP header is 54 bytes in addition to the option fields). If we combine p packets of a flow, one packet per line, each packet is q bytes, and convert each byte to an unsigned integer (in $[0, 255]$), then we get a 2-D matrix $\boldsymbol{X} \in \mathbb{R}^{p \times q}$. For flows using different services, the packet sequence has different patterns, and we can get matrices with different features. So we can treat flows as images, extract the basic abstract features using CNN, and then use the FC layer to learn how the basic features of different traffic are combined.

As mentioned above, we can pre-train a CNN to detect basic features, and only fine-tune the FC layer to adapt to different network contexts. In this way, it not only can reduce training costs significantly, but also provides a basis for online learning.

B. Basic Model

As shown in Fig. 2, the main body of ONTC model is a typical CNN. In the pre-training phase, one input passes through several Conv layers, a flatten layer, and two FC layers, all parameters in the model are updated, and after training,

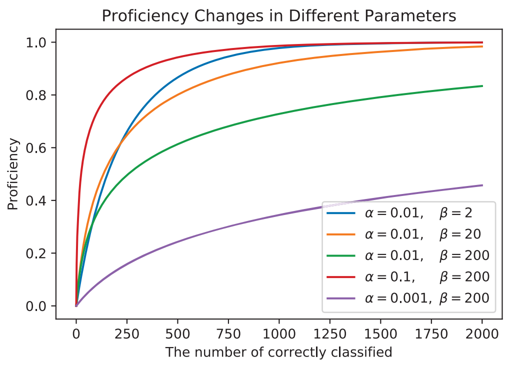

Fig. 3. Proficiency changes in different parameters.

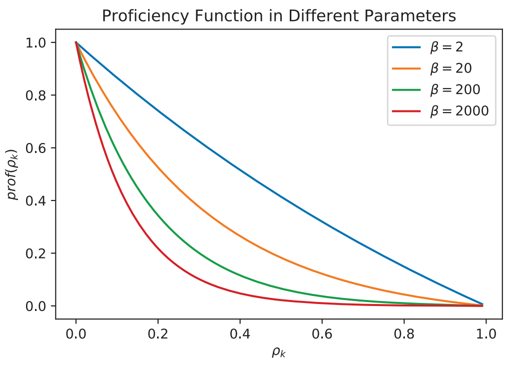

Fig. 4. Proficiency function in different parameters.

obtain the model M_p without the last FC layer. Then in the online-learning stage, for an input $\boldsymbol{X}$ of online model M_o, the basic feature map $\boldsymbol{F}$ is obtained directly through M_p, and the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ between feature map $\boldsymbol{F}$ and output $\boldsymbol{Y}$ are adjusted in an online manner.

Suppose the output layer of M_p contains J nodes, and the output layer of M_o has K nodes. Then we have feature map $\boldsymbol{F} = \{f_j\} \in \mathbb{R}^J$, output $\boldsymbol{Y} = \{y_k\} \in \mathbb{R}^K$, weights $\boldsymbol{W} = \{w_{jk}\} \in \mathbb{R}^{J\times K}$ and biases $\boldsymbol{b} = \{b_k\} \in \mathbb{R}^K$. Since the last layer is the softmax layer, the cross entropy loss function can be derived as follows:

$$L = -\sum_{k=1}^{K} t_k \cdot \log y_k \tag{1}$$

where $\boldsymbol{T} = \{t_k\} \in \mathbb{R}^K$ is the one-hot encoding of the target category, and $y_k = g(z_k)$ is the activation value of the kth output node. The $g(\cdot)$ is softmax function as follows:

$$g(z_k) = \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)} \tag{2}$$

where $z_k = \sum_{j=1}^{J} w_{jk} \cdot f_j + b_k$.

The increments of parameters can be obtained by calculating the partial derivative of (1) :

$$\delta_{w_{jk}} = \frac{\partial L}{\partial w_{jk}} = -f_j \cdot \sum_{i=1}^{K} t_i \cdot (I\{i=k\} - y_k) \tag{3}$$

$$\delta_{b_k} = \frac{\partial L}{\partial b_k} = -\sum_{i=1}^{K} t_i \cdot (I\{i=k\} - y_k) \tag{4}$$

where $I\{\cdot\}$ is the indicator function, if '$\cdot$' is true, the value is one, otherwise is zero.

Then we get the formulas for updating parameters by Stochastic Gradient Descent (SGD):

$$w_{jk} \leftarrow w_{jk} - \eta \cdot \delta_{w_{jk}} \tag{5}$$

$$b_k \leftarrow b_k - \eta \cdot \delta_{b_k} \tag{6}$$

where η is the learning rate, and '$\leftarrow$' means assignment.

C. Stability-Plasticity Dilemma and Catastrophic Forgetting

In the previous subsection, we got the basic model, but it cannot be directly used for online classification. For an online model, its weights are constantly changing to adapt to the environment. When learning new knowledge, it has an effect on what it has been learned, and may even result in catastrophic forgetting. So there must be a trade-off between the ability to learn new information and retain old information, that is the so-called stability-plasticity dilemma.

To deal with the stability-plasticity dilemma, we introduce a parameter $\boldsymbol{\rho} = \{\rho_k\} \in \mathbb{R}^K$ called *"proficiency"*, which measures how skillful to identify a category. The $\rho_k \in [0, 1)$ is the proficiency for the kth output category, and initialized to 0. When making the correct classification, the *'proficiency'* increases, and vice versa. For instance, if $\boldsymbol{X} \in C_k$, where C_k is the kth candidate category, and $\boldsymbol{Y} \in C_k$, then ρ_k increases, if $\boldsymbol{X} \in C_k$ but $\boldsymbol{Y} \notin C_k$ or $\boldsymbol{X} \notin C_k$ but $\boldsymbol{Y} \in C_k$, then ρ_k decreases.

The *proficiency* $\boldsymbol{\rho}$ should have the following properties:

1) The Value: The more the number of correctly classified samples, the higher the corresponding *proficiency*.

2) The Update: The higher the *proficiency*, the slower the change in *proficiency*, whether it is increasing or decreasing.

3) The Effect: Proficiency influences the updating of knowledge. The higher the *proficiency*, the slower the change of knowledge, whether it is learning or forgetting.

The first property is obvious. To achieve the latter two properties, we propose a proficiency function as follows:

$$prof(x) = \alpha \cdot (\beta^{-x} - \beta^{-1}) \tag{7}$$

where α and β are two parameters used to control the function. Then we can update $\boldsymbol{\rho}$ with the following method:

$$\rho_k \leftarrow \rho_k \pm prof(\rho_k) \tag{8}$$

Fig. 3 shows the change of $\boldsymbol{\rho}$ in different parameters according to the 2nd property, and it can be seen how α and β control the proficiency function.

Algorithm 1 Online Update Algorithm

```
Input: A flow matrix $\boldsymbol{X}$, the target category $C_x$, the list of candidate category $C_{list}$, and $\boldsymbol{W} \in \mathbb{R}^{J \times K}$, $\boldsymbol{b} \in \mathbb{R}^K$, $\boldsymbol{\rho} \in \mathbb{R}^K$.
 1: # Check if it's a new flow:
 2: if $C_x \notin C_{list}$ then
 3:     Add $C_x$ to $C_{list}$
 4:     # Update structure as follows:
 5:     $\boldsymbol{w_{k+1}} = \{w_{j(k+1)}\} \in \mathbb{R}^{J \times 1}$ initialized randomly
 6:     $b_{k+1} = 0$
 7:     $\rho_{k+1} = 0$
 8:     $\boldsymbol{W} \in \mathbb{R}^{J \times (K+1)} \leftarrow concat(\boldsymbol{W}, \boldsymbol{w_{k+1}})$
 9:     $\boldsymbol{b} \in \mathbb{R}^{K+1} \leftarrow concat(\boldsymbol{b}, b_{k+1})$
10:     $\boldsymbol{\rho} \in \mathbb{R}^{K+1} \leftarrow concat(\boldsymbol{\rho}, \rho_{k+1})$
11: end if
12: Convert $C_x$ to one-hot encoding $\boldsymbol{T}$.
13: Pass $\boldsymbol{X}$ through pre-trained model $M_p$ and get the feature map $\boldsymbol{F}$.
14: Pass $\boldsymbol{F}$ through online model $M_o$ and get the result $\boldsymbol{Y}$.
15: Update $\boldsymbol{W}$ and $\boldsymbol{b}$ according to (9) and (10).
16: $i = \arg\max(\boldsymbol{Y})$
17: $j = \arg\max(\boldsymbol{T})$
18: if $i \neq j$ then
19:     Decrease $\rho_i$ and $\rho_j$ according to (8).
20: else
21:     Increase $\rho_i$ according to (8).
22: end if
```

Similarly, using the 3rd property, (5) and (6) can be improved as follows:

$$w_{jk} \leftarrow w_{jk} - \eta \cdot prof(\rho_k) \cdot \delta_{w_{jk}} \tag{9}$$

$$b_k \leftarrow b_k - \eta \cdot prof(\rho_k) \cdot \delta_{b_k} \tag{10}$$

When $prof(x)$ is used to train $\boldsymbol{W}$ and $\boldsymbol{b}$, we let $\alpha = (1 - \beta^{-1})^{-1}$ to ensure that $prof(0) = 1$. Fig. 4 shows the $prof(\rho_k)$ in different parameters.

We can control the update speed of *proficiency* and weights through α and β, and then achieve the trade-off between stability and plasticity.

D. Algorithm

In the previous subsection, we discussed the approach of updating parameters. Actually, in addition to the parameters update, we also need to update the structure of the online model. Since the last layer of the model is the softmax layer, the number of output nodes is equal to the number of candidate categories. When a new category occurs, the number of output nodes needs to be increased accordingly.

The complete online update pseudo-code is shown in Algorithm 1.

Lines 2-11: determine if the structure needs to be updated and executed, where the $concat(\cdot)$ used to concatenate two tensors along the appropriate dimension.

Line 15: update the weights and biases.

Algorithm 2 Complete Process of ONTC

```
 1: Train the pre-training CNN $M_p$ and flow-level model $M_f$, and prepare the packet-level model $M_o$.
 2: while receive a packet do
 3:     Storing flow $id$ (the 5-tuple) and packet header.
 4:     if already stored $p$ packets with the same $id$ then
 5:         $M_o$ receives the $p$ packets and output the classification result.
 6:     end if
 7:     if a flow is complete then
 8:         $M_f$ receives these packets and return the result $C_x$.
 9:         Pass $\boldsymbol{X}$ of $p$ packets and $C_x$ to $M_o$, and call Algorithm 1.
10:         Release the storage space of this flow.
11:     end if
12: end while
```

Lines 18-22: decide how to update *proficiency* and perform updates.

Finally, combine the pre-training model, flow-level model and packet-level model to get the complete process of ONTC method shown in Algorithm 2.

IV. Experiment

The performance of the proposed method was verified through experiment based on TensorFlow. The details are as follows.

A. Dataset

To test the performance of the proposed method, we choose two open datasets, UNIBS traces [11] provided by F. Gringoli and his team [12], [13] and UPC traces [14] provided by V. Carela-Español and T. Bujlow [15], [16]. Both datasets provide the pcap trace file and corresponding ground truth.

For each packet, 70 bytes were intercepted, including the 16 bytes pcap header and 54 bytes TCP header (the UDP header was padding to 54 bytes with zeros). We converted each byte of the raw binary data into an unsigned integer, and then combined 10 packets into a 10×70 array as the input of model.

We selected six categories from all data, "RDP", "BitTorrent", "Web", "SSH", "eDonkey" and "NTP". The pre-training CNN was trained by only the first 5 kinds of samples, and when testing the online model, the "NTP" category was added to verify the adaptability to new context.

B. Model Structure

In our experiment, we first built a CNN for learning the basic features. As mentioned above, the input size of the CNN is 10×70, and then three Conv layers follow, the kernels are $10 \times 2 \times 2$, $20 \times 2 \times 2$, $40 \times 2 \times 2$ respectively. We do not use any pooling layers, the pooling layer is to provide translation invariance for image, while the traffic data does not have translation invariance. After Conv layers is a flatten layer and a FC layer with 200 nodes. Finally, the output is a

TABLE I
CONFUSION MATRIX

Actual Class	Predicted Class	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

softmax layer. To ease over-fitting, dropout and regularization techniques are applied during training.

Then we use the method proposed in Section III to fine tune the online model. The parameters of proficiency function for updating $\boldsymbol{\rho}$ is set as $\alpha = 0.001, \beta = 200$, and for updating $\boldsymbol{W}$ and $\boldsymbol{b}$ is set as $\beta = 2$.

C. Performance Metric

Two metrics were used to measure the performance of the proposed method, they are *Accuracy* and *F1-score*. The *Accuracy* measures the overall correct rate of predictions for multiple categories, and is defined as:

$$Accuracy = \frac{N_{\texttt{correct}}}{N_{\texttt{total}}} \tag{11}$$

where $N_{\texttt{correct}}$ is the number of correct samples, and $N_{\texttt{total}}$ is the number of total samples.

F1-score is a popular metric in binary classification problem. We use it to measure the ability to identify different categories. *F1-score* is based on two other metrics: *Precision* and *Recall*, which are calculated as follows through the confusion matrix shown in TABLE I:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

And then the *F1-score* is defined as:

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{14}$$

For both *Accuracy* and *F1-score* metrics, larger values mean better performance.

D. The Effect of Proficiency

Firstly, the effect of the *proficiency* $\boldsymbol{\rho}$ is verified through *Accuracy*. We designed two experiments, one with the *proficiency* and another without, the rest of the settings are the same. To explore the adaptability to environment changes, only five categories of traffic were fed to the model at the beginning and now the output layer has five nodes. After about 150000 steps, the 6th kind samples were added to the dataset, and the model can increase output nodes autonomously. We calculate the overall *Accuracy* (covering all results from the start to the current step) to show the overall performance, and the current *Accuracy* (covering only the last 3000 steps results) to show the performance fluctuations. As shown in Fig. 5, the thick lines are overall *Accuracy* and the thin lines are current *Accuracy*. Obviously, the *Accuracy* of model with *proficiency*

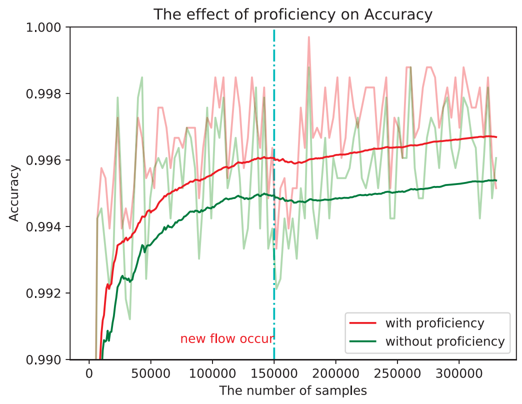

Fig. 5. The effect of proficiency on Accuracy.

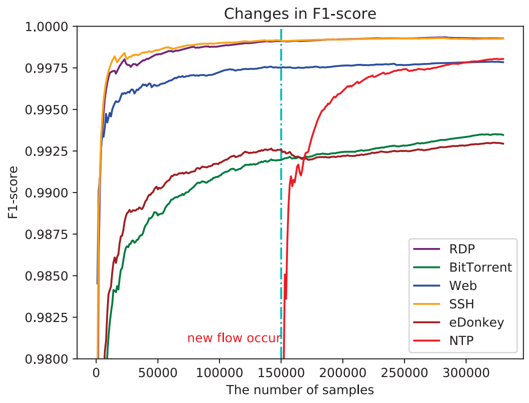

Fig. 6. Changes in the F1-score for different categories.

is always better than model without *proficiency*, and after the new category occurs, the gap between them has further increased. In addition, the current *Accuracy* curves show that the performance of model without *proficiency* is more volatile than the one with *proficiency*. Due to the appearance of new category, the context has changed, so the *Accuracy* of both model has declined. But the model without *proficiency* is more susceptible to catastrophic forgetting problem, so the gap increases. Besides, the fluctuation of the curve without *proficiency* is more obvious, indicating that the model lacks stability.

E. The Performance of ONTC

We further studied the performance of the proposed method through *F1-score*. The model of this experiment is the same as the model with *proficiency* above. To study the overall performance, the *F1-score* for different categories is calculated as follows: when calculating the *F1-score* for category C_k, C_k is the *Positive* sample, and all other categories are *Negative* samples, then the confusion matrix is obtained and the *F1-score* is calculated by (14). Fig. 6 shows the result of *F1-score* for different categories. When the new category occurs, the model can learn how to identify new samples quickly.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT METHODS

Method		NB	NB_10	C&R	C&R_10	**ONTC**
Speed		32.4ms	29.9ms	36.8ms	32.3ms	**5.1ms**
Accuracy		0.912	0.830	0.952	0.823	**0.997**
F1	RDP	0.995	0.970	0.971	0.873	**0.999**
	BitTorrent	0.708	0.579	0.890	0.734	**0.993**
	Web	0.941	0.855	0.982	0.881	**0.997**
	SSH	0.882	0.709	0.941	0.689	**0.999**
	eDonkey	0.327	0.018	0.709	0.334	**0.992**

As can be seen from the curves, when the model learns new categories, the impact on other categories is very small. This shows that while dynamically adapting to changes in the environment, the model retains well-know old knowledge.

F. Performance Comparison

We implemented two other methods for comparison: one is the most classic Moore's Naive Bayesian (NB) [2], and the other is Lopez-Martin's method of combining CNN and RNN (C&R) [6]. Only 5 kinds of samples are used as our pre-trained CNN. TABLE II shows the comparison of the performance of different methods. The method name with "_10" is 10-packets version (the same as ONTC, identify traffic with only 10 packets). The original NB and C&R method can both achieve accuracy more than 0.9, but the performance of the corresponding 10-packets version is significantly reduced. So they cannot meet the requirements of early stage identification. In addition, although their overall accuracy can exceed 0.9, the performance of some classes (e.g. eDonkey) is very poor. Furthermore, the identification speed is also critical for network traffic classification. The NB and C&R methods use statistical features for identification, so they need to extract features from a flow at first. While our ONTC method uses the raw binary data of packet header as input of the model, this improves the speed of identifying a flow by more than 6 times.

Overall, the results in this section show that the proposed method is effective for ONTC task. Through online learning, it provides the ability to adapt to the network context automatically. The online model with *proficiency* can not only prevent catastrophic forgetting, but also achieve a trade-off between stability and plasticity. In addition, the model used only 10 packets, and achieved accuracy over 0.99, so a flow can be identified at the early stage. The adaptability for context and early stage identification improve the practicality of ONTC greatly.

V. CONCLUSION

The dynamic nature of network and the need for early stage identification present challenge to the network traffic classification technology. In this paper, we proposed a new method, ONTC, to identify traffic adaptively. Compared with other ML-based methods, the ONTC has several advantages: First, it has the adaptability to environment changes. The introduction of *proficiency* and proficiency function solve the catastrophic forgetting and the trade-off between stability and plasticity. Second, it enables the early stage identification. Considerable performance can be achieved by only 10 packets of each flow. Third, it achieves the perfect combination of accuracy, timeliness and adaptability. The flow-level classifier is more accuracy, the packet-level classifier is more timely, and the online-learning has adaptability. With more research on flow-level unsupervised clustering methods, the ONTC model would be more powerful in the future work.

REFERENCES

[1] H. Shi, G. Liang, and H. Wang, "A novel traffic identification approach based on multifractal analysis and combined neural network," *annals of telecommunications-annales des télécommunications*, vol. 69, no. 3-4, pp. 155–169, 2014.

[2] 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.

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

[4] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[5] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*. ACM, 2008, p. 11.

[6] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[7] Z. Chen, L. Peng, C. Gao, B. Yang, Y. Chen, and J. Li, "Flexible neural trees based early stage identification for ip traffic," *Soft Computing*, vol. 21, no. 8, pp. 2035–2046, 2017.

[8] L. C. Jain, M. Seera, C. P. Lim, and P. Balasubramaniam, "A review of online learning in supervised neural networks," *Neural computing and applications*, vol. 25, no. 3-4, pp. 491–509, 2014.

[9] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks," *Artificial Intelligence Review*, vol. 49, no. 2, pp. 281–299, 2018.

[10] 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.

[11] "The UNIBS Anonymized 2009 Internet Traces." [Online]. Available: http://netweb.ing.unibs.it/~ntw/tools/traces

[12] M. Dusi, F. Gringoli, and L. Salgarelli, "Quantifying the accuracy of the ground truth associated with internet traffic traces," *Computer Networks*, vol. 55, no. 5, pp. 1158–1167, 2011.

[13] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso *et al.*, "Gt: picking up the truth from the ground for internet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 12–18, 2009.

[14] "Traffic Classification at the Universitat Politècnica de Catalunya." [Online]. Available: https://cba.upc.edu/monitoring/traffic-classification

[15] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, "Is our ground-truth for traffic classification reliable?" in *International Conference on Passive and Active Network Measurement*. Springer, 2014, pp. 98–108.

[16] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular dpi tools for traffic classification," *Computer Networks*, vol. 76, pp. 75–89, 2015.