A Framework for Resource-aware Online Traffic Classification Using CNN

Wanqian Zhang
Dalian University of Technology
Dalian, China
zhangwanqian@mail.dlut.edu.cn

Junxiao Wang
Dalian University of Technology
Dalian, China
wangjunxiao@mail.dlut.edu.cn

Sheng Chen
Tianjin University
Tianjin, China
chensheng@tju.edu.cn

Heng Qi*
Dalian University of Technology
Dalian, China
hengqi@dlut.edu.cn

Keqiu Li
Dalian University of Technology
Dalian, China
likeqiu@gmail.com

ABSTRACT

As a fundamental problem in network security and management, traffic classification has attracted more and more research interest. In existing work, machine learning based traffic classification methods are mainstream in recent years. With the development of deep learning, Convolutional Neural Network (CNN) is widely used in traffic classification, achieving promising results. However, prior work only focuses on how to improve the accuracy of classification tasks without considering the time efficiency. As we know, the deep learning models require a lot of computational overhead. Therefore it is necessary to realize realtime CNN-based traffic classification with limited computational resources. In this paper, we propose a new framework for online traffic classification using CNN. By detecting CPU occupancy in real time, the proposed framwork can seek optimal window sizes using a regression model of meta-parameters to achieve accuracy at a lower cost of resources. The simulation experiments with real trace data show that the proposed framework significantly reduces processing latency by about 77%, while achieving matchable accuracy of classification compared to the state-of-the-art method.

CCS CONCEPTS

• **Networks** → **Network monitoring**.

KEYWORDS

Traffic Classification, CNN, CPU Occupancy, Low-Latency, Parameter Regression

*Heng Qi is the corresponding author.

CFI'19, August 7–9, 2019, Phuket, Thailand

ACM ISBN 978-1-4503-7238-1/19/08…$15.00
https://doi.org/10.1145/3341188.3341195

ACM Reference Format:
Wanqian Zhang, Junxiao Wang, Sheng Chen, Heng Qi, and Keqiu Li. 2019. A Framework for Resource-aware Online Traffic Classification Using CNN. In *The 14th International Conference on Future Internet Technologies (CFI'19), August 7–9, 2019, Phuket, Thailand*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3341188.3341195

1 INTRODUCTION

Internet traffic classification plays an important role in the field of network management and security. For example, online traffic classification can identify network applications for service quality guarantee or network anomaly detection [12]. Existing classification mainly relies on four technical methods: 1) port number 2) deep packet inspection 3) heuristic technology 4) machine learning.

The port number based method associates well-known port number with application registering the port, and cannot be applied to the identification of dynamic port number [13]. Deep packet inspection performs feature matching and classification by analyzing payload of the packet, but its tight requirements for payload inscription limits its versatility [20]. The heuristic method is not widely accepted due to its low accuracy of classification [8]. The machine learning based method seems to be promising [2, 6, 19]. Most learning models perform an accurate classification on the basis of feature set [4, 5, 9, 15]. However, if the appropriate feature set is not available, the accuracy of these models fails significantly [7]. Recently, there are some deep learning methods proposed not relying on building feature set [11, 17]. They directly learn features from raw packet data taking the benefits of Convolutional Neural Network (CNN), which turns out to be much more accurate for online classification task [16]. The input of these state-of-the-art methods is the packets with a fixed window size. For instance, when window size is set to M, each coming flow will be collected the first M bytes [11] or M packets [17] as the input of model. Because CNN models usually require a lot of computational overhead, the choice of window size is an important issue for reducing overhead and latency.

To analyze the issue of window size, we perform some test on real trace data. We find the following three points:

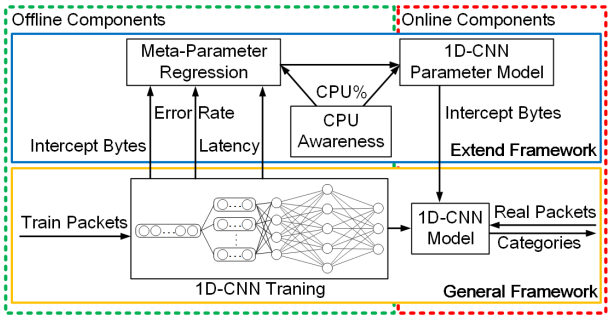

Figure 1: Overview of system framework. We extend the framework based on general ones which split into offline components and online components.

1) If window size is set too large, processing latency will become very high correspondingly. This is caused by too many operations of data preprocessing involved. Even if the packet may not contain so many bytes, a large number of zero-padding operations are still required to satisfy the window size. 2) If window size is set too small, processing latency will be shorter though, frequent thread calls in parallel will result in tightness of CPU, even packet loss and system crash in extreme cases. 3) The distribution of flows and the occupancy of CPU change in real time, so the method of selecting fixed window size in advance may not be able to adapt to dynamics in practice.

To further reduce the computation overhead and latency of CNN-based traffic classification, we propose a new framework for online traffic classification, which can dynamic select suitable window sizes via detecting CPU occupancy in real time. Overall, the proposed framework consists of an offline training stage and an online classification stage. In the offline training stage, we train not only a learning model used for classification tasks, but also another model used for predicting window sizes. In the online classification stage, by inputting CPU occupancy of system in real time, the prediction model can give the window suitable size for the current situation. The classification model loads the suitable window sizes dynamically and achieves low-latency processing. Concretely, the contributions of this paper are summarized as follows:

- We design a novel online traffic classification framework. With a brand-new window size prediction integrated, dynamic window sizes can be loaded onto classification model in real time. The processing latency of classification is thus greatly reduced.
- We implement a prototype system using the proposed framework and 1D-CNN. The prototype system can be used by real network traffic and output traffic classification online.
- We evaluate the prototype system in the settings representing real world scenarios. Results show that our solution outperforms the state-of-the-art method and decreases the processing latency by about 77%. In terms of accuracy of classification, our solution can match the state-of-the-art method.

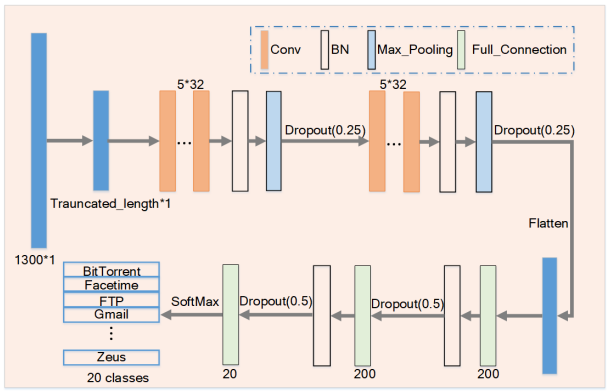

Figure 2: Overview of 1D-CNN model. We employ the model to conduct online traffic categorization task.

2 SYSTEM FRAMEWORK

Figure 1 illustrates the overview of system framework. Basically, we have extended existing learning based traffic classification framework which consists of offline components and online components [14]. In existing offline training stage (left lower side), a model designed for online classification task can be obtained by inputting training traffic into the model and iterative training. In existing online classification stage (right lower side), the system utilizes a well-trained model to conduct with fixed window size and to classify coming flows in real time.

Different from existing framework (lower side), we have added three new components (upper side): the CPU awareness component, the parameter regression component, and the parameter conditioning component. Among them, the CPU awareness component works in both offline training stage and online classification stage, and is responsible for providing the system's CPU vacancy rate to other components in real time. The parameter regression component works in offline training stage, and is responsible for receiving the meta-parameters (window size, accuracy rate, processing latency, CPU occupancy) during the training of classification model. By performing regression training, a model for dynamic parameter conditioning is obtained. The parameter conditioning component works in the online classification stage, and is responsible for receiving CPU idleness rate during the online processing of classification model, and gives appropriate parameters (i.e., window sizes) for the classification model.

3 OFFLINE TRAINING

The classification task in this paper deploys 1D-CNN model which is shown in Figure 2. The jobs of offline training stage include training of 1D-CNN model, testing of 1D-CNN model, and training of parameter model. For training and testing of 1D-CNN model, we divide traffic data into two parts: training set and testing set, where 80% of traffic data is as training

set and 10% is as one testing set, and the rest 10% is as the other testing set.

3.1 Training and testing of 1D-CNN

The data is first subjected to convolution operations of convolutional layer of size (5, 32), followed by activation of Relu layer. Then, BN layer normalizes activated feature layer, and standardized results are sent to maximum pooling layer. The pooling layer performs feature compression and uses a dropout layer with a retention ratio of 0.25 to prevent over-fitting in training. The above process is performed twice in sequence, transforming raw traffic data into abstracted features. To ensure overall performance, we add batch normalization layer to speed up convergence. Then, the full connection is to linearly map features extracted in previous steps, and also combines BN layer with corresponding dropout layer (retention ratio as 0.5) to prevent over-fitting of the model again. The sequential execution of full connected layer, BN layer, and dropout layer is also twice. The full connection weight is set to 200 here. At last, the number of outputs is adjusted by full connection layer and let it be equal to the number of traffic categories.

The loss function is designed as SoftMax logic values corresponding to traffic categories, and its cross-entropy calculation is performed with one-hot code of ground truth in traffic data (See Equation 1),

$$Loss = -\sum_{k=1}^{C} y'_k \log y_k \tag{1}$$

where C is the number of traffic categories. y_k is the predicted results of 1D-CNN model, corresponding to probabilities for C traffic categories. y'_k is the target categories labeled by one-hot code. The training is optimized by adaptive learning rate based on RMSprop algorithm. The number of epoches is set to 10 to achieve a fast convergence of the model.

3.2 training of parameter model

In the process of testing well-trained 1D-CNN model, by incrementing model parameters (i.e., window size) within a certain range, meta-parameters including window size, accuracy of classification and processing latency can be obtained. These meta-parameters are input into the regression algorithms together with current CPU vacancy rate.

The process of parameter regression involves two algorithms, which are least squares regression and LSTM regression, respectively. The optimization objection of least squares regression is represented by Equation 2,

$$\min_{\varphi} \sum_{i=1}^{m} \delta_i^2 = \sum_{i=1}^{m} (\varphi(x_i) - y_i)^2 \tag{2}$$

where $\varphi(x_i)$ is the polynomial predictions fitted by least squares regression. y_i is the observed values. The fitting of least squares regression is achieved by minimizing the interpolation δ_i. The LSTM regression refers to Equation 3,

$$Acc = Lstm(l; \theta) \tag{3}$$

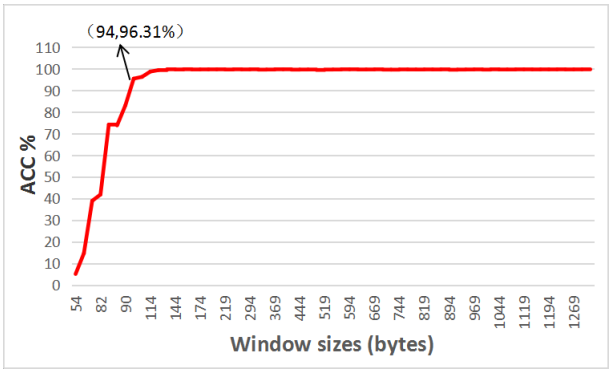

Figure 3: Accuracy vs. window size.

where l is the model parameters (i.e., window size) input into LTSM model. θ is the variables of LTSM model. Acc corresponds to the accuracy of prediction. The LSTM model here uses a memory unit to train 5,000 epoches. At last, via meta-parameter regression, three functions are obtained: 1) $t = f_1(c)$, 2) $l = f_2(t)$, and 3) $A = f_3(l)$. c, t, l, A indicate CPU vacancy rate, processing latency, window size and accuracy of categorization, respectively. For the training of the parameter model, we will show more details in the section 5.

4 ONLINE CLASSIFICATION

The jobs of online classification stage include awareness of CPU occupancy, generation of parameters (i.e., window size), and traffic classification with parameters loaded.

4.1 Awareness of CPU occupancy

Although, computing resources typically include multiple resources, including CPU, GPU, cache, and memory. Prior works [1, 3] pointed out that in real-time traffic processing scenario, CPU is required to perform a series of data acquisition operations (reading and pre-processing packets) with frequent thread calls in parallel, making CPU extremely easy to become a bottleneck of computing performance.

We consider CPU occupancy as a measurement of system resources. In CPU awareness component, it monitors CPU occupancy in real time through `top` command. We denote a to indicate average CPU occupancy of system for the current period, and denote s to indicate total CPU resources (CPU vacancy rate) available to system. The value of s depends on the total number of CPUs in operating system. When the total number of cores is 12, $s = 1200\%$.

4.2 Generation of parameters

We analyze classification accuracy of 1D-CNN model under different window sizes, whose results are depicted in Figure 3, TABLE 1 and TABLE 2. We find that when CPU occupancy is constant, as window size increases, accuracy of classification also increases since information of learning involved is getting larger. As shown in Figure 3, when window

Table 1: Zoom in 78-94 bytes.

window size (bytes)	accuracy (%)
78	38.92
82	41.84
86	74.22
88	73.86
90	83.16
92	95.49
94	96.31

Table 2: Contained fields of packet header against different window sizes.

window size (bytes)	src&dst port	seq num	ack num	offset	window	checksum
78	*	-	-	-	-	-
82	*	*	-	-	-	-
86	*	*	*	-	-	-
88	*	*	*	*	-	-
90	*	*	*	*	*	-
92	*	*	*	*	*	*
94	*	*	*	*	*	*

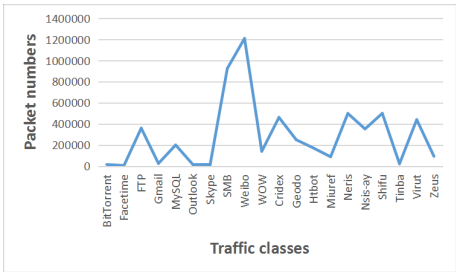

Figure 4: Traffic distribution of different types.

size reaches a certain value, accuracy of classification cannot increase any more. For TABLE 2, contained fields of packet header is zoomed in under different window sizes, which we think should play a leading role on accuracy of classification. Combined with TABLE 1, it can be seen that window size of 94 bytes is an adjacent point. Before this point, the accuracy varies greatly. After that point, accuracy tends to be stable.

Based on these observations, we propose a parameter generation policy to guarantee accuracy of classification, at the same time bringing about processing latency as few as possible. We denote p' the guaranteed accuracy. For current CPU occupancy a, via as trained-well parameter model introduced in Section 3, regressions of window size $l = f_2(f_1(a))$ and accuracy $p = f_3(f_2(f_1(a)))$ are obtained. If $p \geqslant p'$, output l as the selected window size. Otherwise, increment l till $p \geqslant p'$.

4.3 Parameters loaded

We consider packets arrive at system in real time and are cached in the buffer managed by NIC driver. Whenever packets fill up the space of buffer, system parses the part of packets into a session and reads in the first packet of each session. The system converts the binary encoding of these packets into decimal encoding. Then, their first M bytes are intercepted as a batch input into 1D-CNN model for classification. M is the parameter (i.e., window size), which needs to be loaded in current period.

5 EVALUATION

5.1 Simulation setup

Trace: We leverage the USTC-TFC2016 traces used by [17] to conduct our simulation. These traces include 10 classes of public website malware traffic and 10 classes of normal traffic. Figure 4 depicts the traffic distribution of different classes. The number of packets of different classification varies widely. In order to ensure the correctness of experiments, our simulation is based on the Facetime traffic with the least number of packets. Each category intercepts the first 6,000 packets and forms a new data set containing 20 classes of traffic for a total of 120,000 packets. The data set is divided into training set, testing set 1 and testing set 2 in a ratio of 8:1:1. The training set is used to train 1D-CNN model. The testing set 1 is used to test 1D-CNN model and construct the parameter conditioning model. The testing set 2 is used to examine online performance of system in terms of processing latency and accuracy of classification.

Testbed: All the simulations are conducted with the server NVIDIA RTX 2080Ti GPU. We leverage the platform of Tensorflow 1.9 and Keras to implement 1D-CNN model. In order to make a fair comparison between different window sizes, we ensure that the running status of other applications other than Python is consistent during the training and prediction process. The 1D-CNN model is trained for 10 epoches, and the learning rate is fixed at 0.01, using the loss function of cross entropy.

Metrics Cared: In our simulations, the evaluation metrics used are overall accuracy and flow classification time. The overall accuracy is defined as follows: $Acc = N + /N$, where $N+$ indicates the amount of traffic identified correctly, and N is the amount of all traffic to be identified. The flow classification time is defined as: $Time = T_I - T_A$, where T_I is the timestamp flow which is identified, and T_A is the timestamp flow which is arriving at system.

Scheme Compared: We compare our solution with two other state-of-the-art methods: RawPower [11] and the method in [17]. RawPower is a 1D-CNN method with raw packets as input and with fixed window size. Another method is one 2D-CNN method and converts the traffic classification problem into an image recognition problem.

5.2 Flow classification time

We first compare our solution with the state-of-art method RawPower. For RawPower, the window size is fixed and is set to 1300 according to [11]. Figure 5 plots the selection of window size under different CPU vacancy rate. Combined with Figure 6, it clearly shows that our solution with dynamic selection of window size performs better in computational

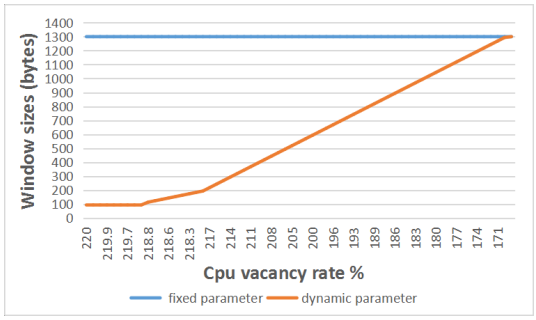

Figure 5: Selection of window size against different CPU occupancies.

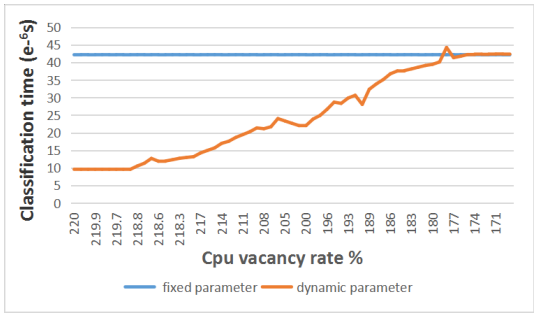

Figure 6: Corresponding flow classification times with dynamic selection of window size.

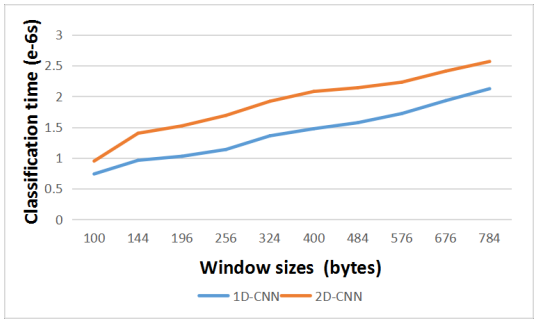

Figure 7: Comparison of flow classification times between 1D-CNN and 2D-CNN.

efficiency. Our solution has about 77% less than flow classification time on average, and the time can be shorten to 22.75% required for RawPower in the extreme case.

Then, we also compare our solution with 2D-CNN method. For 2D-CNN, the different number of bytes are intercepted and converted into n×n gray-scale images and input into the 2D-CNN model for classification. We conduct ten experiments and compare the flow classification time between two methods. We find that 1D-CNN takes less 33.4% time then that is required for 2D-CNN.

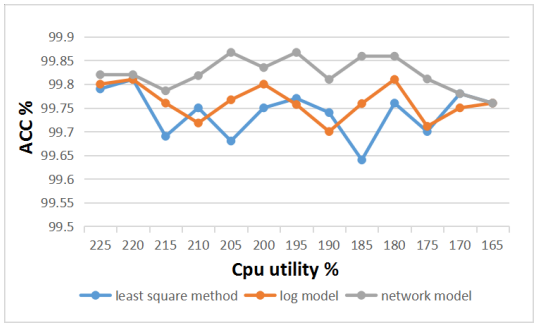

Figure 8: Comparison of accuracy between polynomial fitting, logarithmic fitting and LSTM.

5.3 Meta-parameter regression

The regression of meta-parameters includes three parts: 1) flow classification time and CPU vacancy rate 2) window size and flow classification time 3) accuracy of classification and window size. The former two parts are obtained by least squares regression. They are $y = 6.222e-07x5-0.003167x4+6.445x3-6557x2+3.335e+06x-6.782e+08$ and $y = -9.398e-05x5+0.01429x4-0.8084x3+21.09x2-218.1x+841.2$, respectively. The third part is different from the former two, since the accuracy tends to be stable as window size grows up. We find that polynomial fitting and logarithmic fitting cannot adapt to the trend; but rather the effect of LSTM based RNN is better. Figure 8 depicts the comparison of accuracy between polynomial fitting, logarithmic fitting and LSTM. It clearly shows that LSTM based RNN is more suitable for fitting the third part.

5.4 Overall accuracy

In terms of accuracy of classification, we also examine our solution with RawPower and 2D-CNN. Figure 9 depicts the comparison for accuracy of classification. It can be seen that our solution achieves matchable accuracy compared to state-of-the-art method which we think is very accurate. Figure 9 depicts the comparison between our solution and 2D-CNN. We find that the accuracy of 1D-CNN is slightly higher than the accuracy of 2D-CNN, which is within 0.01%. This validates effectiveness of our solution to achieve much latency reduction with no loss in accuracy of classification.

6 RELATED WORK

The solutions on traffic classification mainly focus on the methods of machine learning. These studies are mainly implemented using stream features and grouping features. Wang et al. [15], Coull et al. [4] and Mauro et al. [5] apply flow features, packet features and flow features, respectively. The corresponding classifiers are C4.5 decision tree, naive Bayes and random forest. In recent years, some methods of deep learning have shown good performance. Wang [18] proposes a network protocol identification method based on stacked automatic encoder (SAE), which uses the raw flow data to achieve high accuracy. Wang et al. [17] propose the method

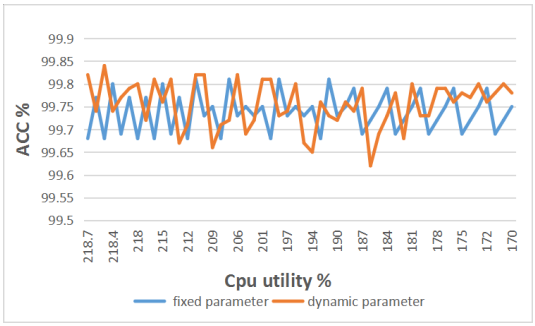

Figure 9: Accuracy of categorization between our solution and RawPower.

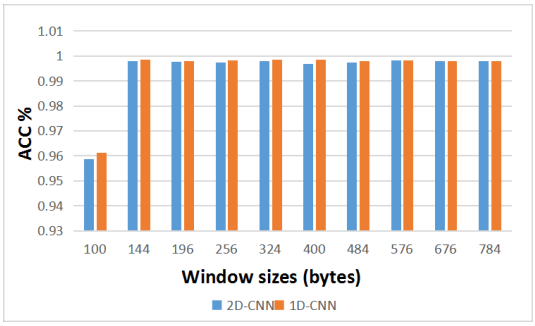

Figure 10: Accuracy of categorization between our solution and 2D-CNN.

of converting sessions into pictures and using CNN to identify them, so as to achieve the purpose of traffic classification. Gonzalo Marn et al. [10] propose a packet-level based approach and use 1D-CNN for learning, which achieves good accuracy. Different from these existing works, this paper pays more attention to the high efficiency of computation. It is hoped that by dynamically conditioning the parameters (i.e., window size), the processing latency can be significantly shortened while ensuring the accuracy.

7 CONCLUSION

We propose a dynamic parameter (i.e., window size) conditioning method, which can be integrated with learning based traffic classification framework. By detecting CPU occupancy in real time, it provides suitable window sizes for the learning model and helps to satisfy accuracy at a lower cost of latency. Simulation results show that our approach can significantly reduce processing latency by about 77%, and achieves matchable accuracy of classification compared to state-of-the-art method which we think is very accurate.

REFERENCES

[1] Shuichi Asano, Tsutomu Maruyama, and Yoshiki Yamaguchi. 2009. Performance comparison of FPGA, GPU and CPU in image processing. In *Proc. of IEEE International Conference on Field Programmable Logic and Applications*.

[2] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kav Salamatian. 2006. Traffic classification on the fly. *Acm Sigcomm Computer Communication Review* 36, 2 (2006), 23.

[3] Young-kyu Choi, Jason Cong, Zhenman Fang, Yuchen Hao, Glenn Reinman, and Peng Wei. 2016. A quantitative analysis on microarchitectures of modern CPU-FPGA platforms. In *Proc. of ACM Annual Design Automation Conference*.

[4] Scott E Coull and Kevin P Dyer. 2014. Traffic analysis of encrypted messaging services: Apple imessage and beyond. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 5–11.

[5] Mario Di Mauro and Maurizio Longo. 2015. Revealing encrypted WebRTC traffic via machine learning tools. In *Proc. of IEEE ICETE*.

[6] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. 2006. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*. ACM, 281–286.

[7] Zubair Md Fadlullah, Fengxiao Tang, Bomin Mao, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani. 2017. State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2432–2455.

[8] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: multilevel traffic classification in the dark. 35, 4 (2005), 229–240.

[9] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access* 5 (2017), 18042–18050.

[10] Gonzalo Marín, Pedro Casas, and Germán Capdehourat. 2018. Rawpower: Deep learning based anomaly detection from raw network traffic measurements. In *Proc. of ACM SIGCOMM Conference on Posters and Demos*.

[11] Gonzalo Marín, Pedro Casas, and Germán Capdehourat. 2019. DeepSec meets RawPower-Deep Learning for Detection of Network Attacks Using Raw Representations. *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (2019), 147–150.

[12] Andrew W Moore and Denis Zuev. 2005. Internet traffic classification using bayesian analysis techniques. 33, 1 (2005), 50–60.

[13] Thuy TT Nguyen and Grenville J Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials* 10, 1-4 (2008), 56–76.

[14] Da Tong, Yun Rock Qu, and Viktor K Prasanna. 2017. Accelerating decision tree based traffic classification on FPGA and multicore Platforms. *IEEE Transactions on Parallel and Distributed Systems* 28, 11 (2017), 3046–3059.

[15] Dawei Wang, Luoshi Zhang, Zhenlong Yuan, Yibo Xue, and Yingfei Dong. 2014. Characterizing application behaviors for classifying p2p traffic. In *Proc. of IEEE ICNC*.

[16] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. 2017. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 43–48.

[17] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. 2017. Malware traffic classification using convolutional neural network for representation learning. In *Proc. of IEEE ICOIN*.

[18] Zhanyi Wang. 2015. The applications of deep learning on traffic identification. *BlackHat USA* 24, 11 (2015), 1–10.

[19] Nigel Williams, Sebastian Zander, and Grenville J. Armitage. 2006. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *Acm Sigcomm Computer Communication Review* 36, 5 (2006), 5–16.

[20] Jun Zhang, Xiao Chen, Yang Xiang, Wanlei Zhou, and Jie Wu. 2015. Robust network traffic classification. *IEEE/ACM Transactions on Networking* 23, 4 (2015), 1257–1270.