Vol.2018-IOT-40 No.6
2018/3/5

Towards Self-Optimizing Network: Applying Deep Learning to Network Traffic Categorization and Identification in the Context of Application-Aware Network

Pongsakorn U-chupala[1,a)] Yasuhiro Watashiba[2,b)] Kohei Ichikawa[1,c)] Hajimu Iida[1,d)]

Abstract: The application-aware routing is a network routing technology optimized for a network with an inconsistent link performance, a problem which is common for a multi-institution research and academic network. Using the application-aware routing, an application-aware network routes each flow independently via the optimal path corresponding to the identified application characteristic. This technology enables the creation of a self-optimizing network. However, an automatic network flow categorization and identification system is required. In the scope of this work, network flow categorization is defined as the process of generating a meaningful classification whereas network flow identification is defined as identifying which class a network flow belongs to. These are challenging problems with various applicabilities. We present a deep learning approach to network flow categorization and identification problems. Deep learning provides several advantages over existing solutions in the context of the application-aware network. According to our experiments, a 3-layer stacked denoising autoencoder trained with CAIDA Internet traffic dataset produces the most meaningful classification and a useful class identifier (classifier). This deep neural network (DNN) model generates three-classes classification: a bandwidth-bound pattern, a latency-bound pattern, and an irregular pattern. A design of a highly scalable implementation of a self-optimizing network using a DNN model is also presented with justification for each design decision. Our findings suggest that a deep learning approach to network flow categorization and identification problems in the context of the application-aware network and the self-optimizing network are promising.

1. Introduction

The application-aware routing is a novel network routing method optimized for a network with a highly inconsistent link performance [1]. The link performance inconsistency is very common problem in a geographically separated wide-area network connecting multiple research or academic institutions. In our previous work, we have implemented the mechanism of application-aware routing using OpenFlow technologies [2] and developed an application-aware network by applying the mechanism. The application-aware network was evaluated on an international OpenFlow network testbed, PRAGMA-ENT [3], which is composed of multiple research and education networks including the United States, Japanese, Taiwanese and Thai resources, and we found that the performance of data-intensive applications can be improved by the application-aware routing.

The application-aware routing prioritizes paths with properties corresponding to application's characteristics. Using a predetermined network flow classification and flow optimization rules, an application-aware network identifies the class each individual network flow belongs to. Each flow is then routed via the most optimal path given its class and the current condition of all links in the network. For example, let there be two network applications, a VoIP application and a video streaming application. The performance of a VoIP application could be defined as latency-bound (performs better under low latency) whereas the performance of a video streaming application could be defined as bandwidth-bound (performs better with higher bandwidth). Even if these two applications are sending data from the same source to the same destination (and vice versa), the optimal path for the flows of each application could be different. Figure 1 illustrates the concept of application-aware network.

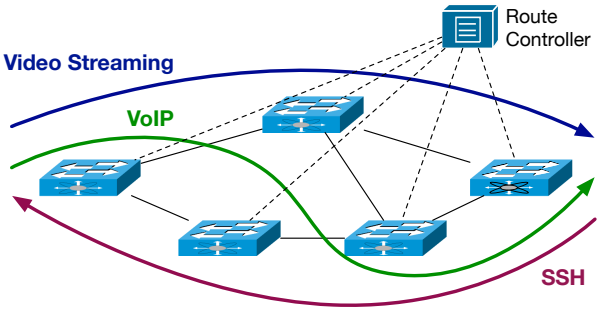


Fig. 1: An example routing with application-aware network

An automatic network flow categorization and identification is essential for realizing a self-optimizing network with an

[1] Nara Institute of Science and Technology, Ikoma, Nara 630–0101, Japan
[2] Osaka University, Suita, Osaka 565–0871, Japan
[a)] pongsakorn.uchupala.pm7@is.naist.jp
[b)] watashiba-y@cmc.osaka-u.ac.jp
[c)] ichikawa@is.naist.jp
[d)] iida@itc.naist.jp

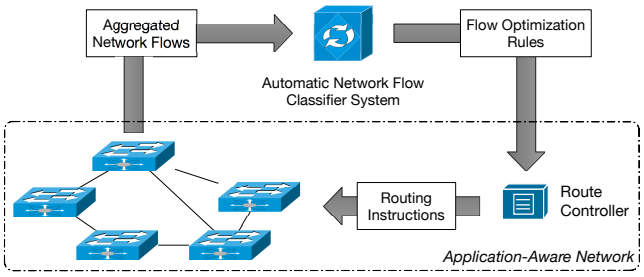


Fig. 2: The concept of a self-optimizing network

application-aware network. Network traffic categorization and identification, in general, is also useful for network monitoring, quality of service (QoS) control, network security, as well as data mining. Deploying an application-aware network involves configuring a predetermined flow classification and flow optimization rules for network flows corresponding to each application using the network. An automated method to create (and update) flow classification and flow optimization rules using communication data from the network would complete a cycle and create an automatic self-optimizing network. Figure 2 illustrates the concept of a self-optimizing network.

Generating a meaningful classification of network flows together with a corresponding class identifier (classifier) is challenging problem. A class of a network flow is defined as a non-linear representation of the network flow. The classifier model creation process is formulated as an unsupervised clustering of a raw network traffic capture. A raw network traffic capture is commonly available as a large unlabeled lightly-preprocessed dataset. While there are many clustering techniques, a deep-learning-based approach was chosen as it is a technique suitable for this kind of dataset.

The rest of the paper is organized as followed. Section 2 defines the problems, describes existing approaches, and describes the proposed deep learning approach. Section 3 describes the classification and the classifier creation process. Section 4 describes the model training process and the resulting model including the interpretation of the classification. Section 5 proposes the design of a self-optimizing network using the created classifier. Section 6 presents the conclusion and outlines the future work.

2. Background

In the scope of this work, network flows categorization and identification are defined as follow. Network flow categorization is a process of generating a meaningful network flow classification. The classification also has to be useful to the application-aware network meaning that the classification should be based on network properties that could be optimized with network routing. Network flow identification is to identify which class a network flow belongs to, corresponding to the classification.

2.1 Existing Approach

There were issues with the previous approaches to automatically categorize and/or identify network flows from network traffic data. Most of the existing works use an application-level classification which is too specific for route optimization with the application-aware network [4], [5], [6], [7], [8], [9], [10], [11]. In the work by X. Wang, a decision tree with the limited number of predetermined features and thresholds were used [12]. While their work is applicable to the application-aware network, they used a predetermined bandwidth-bound and latency-bound classification with the assumption that the classification is appropriate for any network traffic without providing proper justification. A single-model-fit-all approach also limits the applicability of the model.

2.2 Deep Learning based Approach

A deep learning technique refers to a machine learning technique using deep neural network (DNN) model. While the neural network is not a new technology, a recent breakthrough in parallel computation allowed for a training of a much more complex network within a reasonable time. Multiple types of deep neural network are developed for various applications ranging from (unsupervised) clustering fix-sized input vectors to (supervised) classification of sequential time series data.

Using deep learning for network flows categorization and identification has several advantages. Deep learning is a technique that is highly suitable for capturing a non-linear representation from a large unlabeled lightly-preprocessed dataset. Since the classification of a network flow is a non-linear representation of the flow and a large unlabeled lightly-preprocessed network traffic data set is also readily available, deep learning is a natural choice for this task.

While there are some existing neural-network-based approaches, they do not work very well for creating a self-optimizing network with the application-aware network. Specifically, most of the approaches classify a network flow into a predetermined application-level classification [6], [11] which is too specific for application-aware network and does not guarantee to reflect a natural classification of the dataset. Some works also require labeled input [11] which is not readily available.

In this work, a deep neural network clustering is selected as it is a suitable deep-learning-based approach to network flows categorization and identification (for the application-aware network) with the following characteristics. The approach is unsupervised thus allowing the model to capture natural classification of the dataset without requiring prior knowledge which could bias the results. The resulting classification represents application's communication characteristic and is useful for route optimization.

3. Model Creation Process

This section describes training samples preparation and the deep neural network clustering method to generating a network flow classification and a classifier.

3.1 Training Samples Preparation

Raw network traffic capture is transformed into training samples format suitable for clustering with a deep neural network model. Since the objective is to create an online network flow classifier for an application-aware network, the input format have to be highly scalable to be able to process the large amount of network flows in a near-real-time manner. A fixed-size input vector

is chosen as the input format as it allows for parallel inferencing with multiple instances of the classifier. Protocol (TCP or UDP), source TCP/UDP port, and destination TCP/UDP port are also added to the input vector as these are common features that are known to be related to network application performance characteristic.

The training samples preparation process is divided into three steps as followed.

(1) Raw network traffic is grouped by flows. Each flow is identified by a unique combination of source IP address, destination IP address, protocol (TCP or UDP), source TCP/UDP port, and destination TCP/UDP port.
(2) A flow is sliced into multiple intermediate samples with a 1-second sliding window with 10 seconds in length. Each intermediate sample contains 10 seconds of packets from the flow. Figure 3 illustrates the flow slicing window.
(3) An intermediate sample is restructured with a transfer size binning and a packet count binning into a complete training sample. Packets in the intermediate sample are counted and binned into 10 1-second transfer size bins and 10 1-second packet count bins. Protocol (TCP or UDP), source TCP/UDP port, and destination TCP/UDP port are also added. Ultimately, each complete training sample is a vector with 23 dimensions. Table 1 shows an example complete training sample.

3.2 Training Deep Neural Network Model

A 3-layer stacked denoising autoencoder network with sigmoid activation functions is used in this work. Softmax functions are applied at the final layer to "squash" the output into the probability distribution. The error function is a standard mean square error function comparing the clean input and the reconstructed output. Adam optimizer, an optimization algorithm based on stochastic gradient descent, is used for the optimization with all parameters set to its default value [13]. Figure 4 illustrates the structure of the model.

A stack denoising decoder is an unsupervised deep neural network model structure that is commonly used for automatic feature extraction [14], [15]. The denoising process is introduced to the standard autoencoder to address overfitting problem which is prevalent in case the size of the encoded representation is larger than the size of the input vector. A stack denoising autoencoder model is trained on a layer-by-layer basis. For the training of each layer, the model takes a clean input vector and generate a noised input from the clean input. The noised input is fed to the encoder in the encoding step to create an encoded representation of the noised input. The encoded representation is then fed to the decoder in the decoding step to create a reconstructed input. The quality of the encoded representation is defined as the inverse of the error function. The error function is a distance function between the clean input and the reconstructed input. An optimizer is used to minimize the value of the error function as the training step goes on. After the training of each layer, the decoder is discarded and the training continues with the output from the trained encoder as an input to the next layer. Figure 5 illustrates the training process of the first layer of the stack denoising autoencoder model used in this work.

We have also experimented with the other networks such as deep denoising autoencoder. However, they did not produce useful results. Only the stacked denoising autoencoder is discussed in this paper due to space limitation.

4. Experimental Results

Several stacked denoising autoencoder models were trained with varying numbers of hidden layers, size of each hidden layers, and size of the output vector (numbers of output nodes). All models were trained with CAIDA Internet traffic dataset [16]. Not all results are shown due to space limitation.

The following 3-layer stacked denoising autoencoder model produces the best results. The first two hidden layers contain 100 and 10 nodes respectively. The output layer has 3 nodes. Uniform random masking noise is used to create the noised input from the clean input. Figure 4 illustrates the structure of this model.

The resulting model cluster network communication into three classes. Figure 6 and Figure 7 shown properties distribution of classes generated with this model.

Information from Figure 6 and Figure 7 is used to interpret the meaning of the classes. From the Figure 7, we observe that all distributions of class 2 are grouped at the very small value compared to the other classes. According to the Figure 6, all UDP communications are also clustered into this class 2. We interpret these observations as class 2 representing a low-frequency communication considered as an irregular communication pattern. Comparing only class 0 and class 1, class 1 have a relatively higher transfer size and packet rate across the board. Transferred size standard deviation distribution of class 1 also significantly higher the other classes. We interpret these observations as class 0 represents a regular-frequency communication with relatively lower packet size (latency-bound communication pattern) whereas class 1 represents a regular-frequency communication with relatively higher packet size (bandwidth-bound communication pattern). With these interpretations, the classes are interpreted as followed.

(1) Class 0: Regular-frequency communication with relatively low packet size (latency-bound pattern).
(2) Class 1: Regular-frequency communication with relatively high packet size (bandwidth-bound pattern).
(3) Class 2: Low-frequency communication (irregular pattern).

It is also important to note that the three classes discovered by this model are similar to our prior curated classification (bandwidth-bound/latency-bound classification) used in the development of the application-aware network [1].

5. Self-Optimizing Network

The classifier model presented in this work, together with the application-aware network, can be used to develop an automatic self-optimizing network. Figure 2 illustrates the concept of a self-optimizing network.

An implementation of a self-optimizing network has to be able to operate in a high-throughput environment. Raw network traffic contains an enormous amount of data. Ideally, during each optimization cycle, every network flows have to be identified and categorized Then, a rule for each flow are created or updated to re-

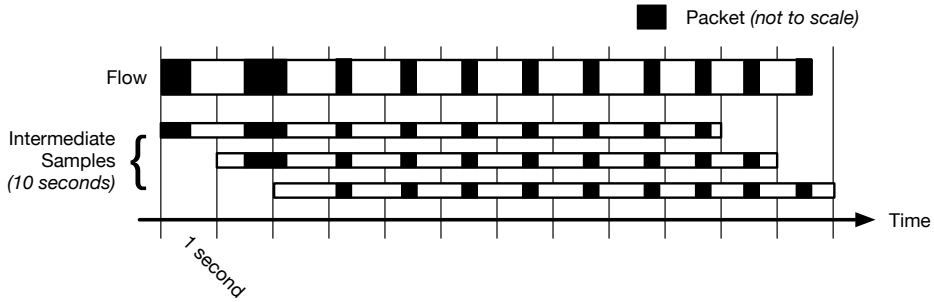


Fig. 3: Example intermediate training samples sliced from a flow

Protocol	Source Port	Destination Port	Transferred Size (second 0-1, Bytes)	...	Transferred Size (second 9-10, Bytes)	Packet Count (second 0-1)	...	Packet Count (second 9-10)
TCP	80	56931	796	...	2,073	6	...	4

Table 1: An example complete training sample

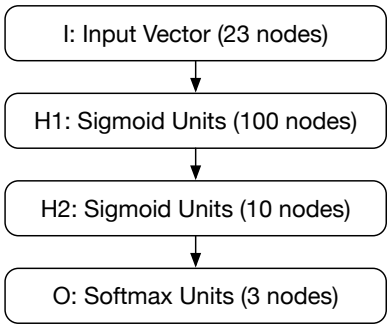


Fig. 4: The 3-layer stacked denoising autoencoder model

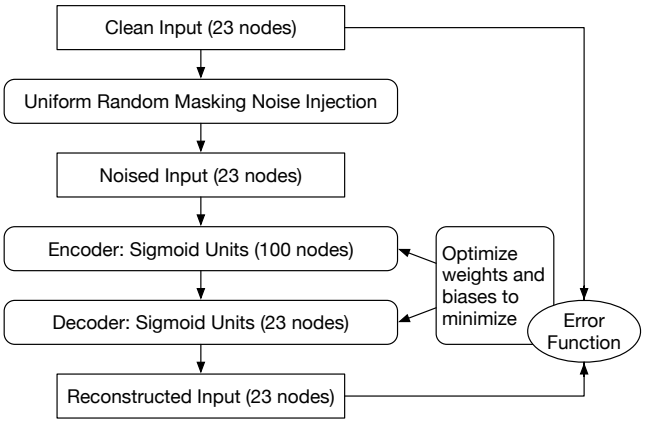


Fig. 5: Training process of the first layer of the stacked denoising autoencoder model used in this work

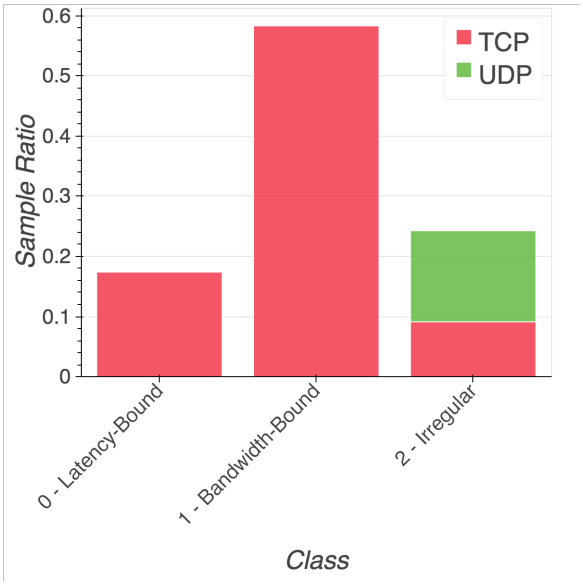


Fig. 6: Class size distribution

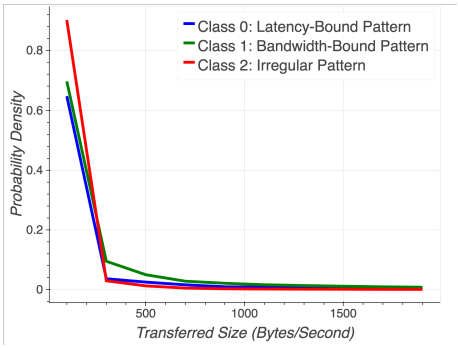
flect the current classification of the corresponding flow. The flow categorization and identification process has to be highly scalable to cope with the amount of the raw network traffic data. Since it is not practical to create a rule for every single flow due to the limitation of OpenFlow, the implementation requires a method to prioritize the creation and maintenance of optimization rules for high-impact flows.

We propose the following design for an implementation of a self-optimizing network. At every edge switch, the ingress communication is mirrored out to a collector port. sFlow [17] could be used to output a uniform sampling of the communication at each switch instead of mirroring everything to reduce the throughput if required. The communication is gathered at the collector and bucketed into each identified flow. The network flow samples are constructed from the latest collected communication together with corresponding duration counter. To prioritize high-impact flows, only flow sample from the flows with the duration longer than a specific threshold value are sent to the classifier model for classification. Flow class identification (classification) can be parallelized for higher throughput with parallel model inferencing. The result of the classification is used to create and update the flow optimization rules for the application-aware network. Figure 8 illustrates the structure of this implementation.

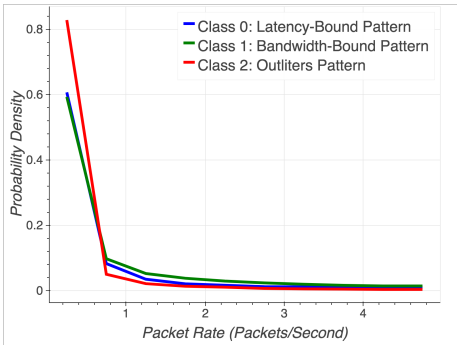
This design is feasible and highly scalable. The sFlow sampling rate and the flow duration threshold can be optimized the achieve the acceptable balance between classification accuracy and scalability unique to each network. Parallel model inferencing also allows for scaling-out by increasing the numbers of the classifiers.

6. Conclusion

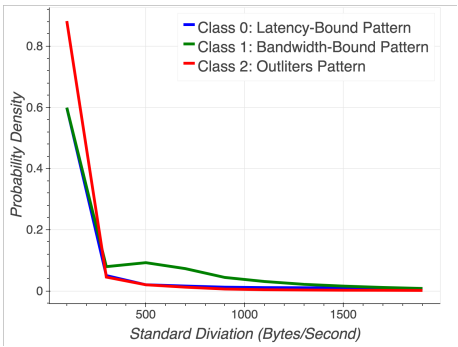
In this work, we describe the development of a network flow categorization and identification model using deep learning technique. The model was developed using a 3-layer stacked denois-



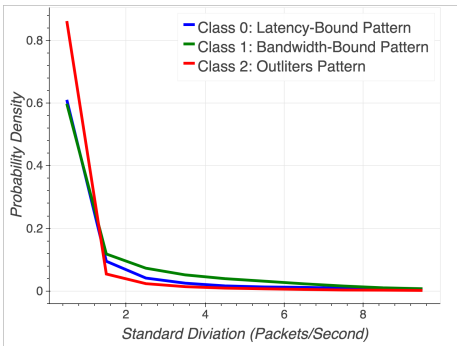
(a) Transferred size distribution per class



(b) Packet rate distribution per class



(c) Transferred size standard deviation distribution per class



(d) Packet rate standard deviation distribution per class

Fig. 7: Properties distribution of the resulting classification

ing autoencoder and trained with CAIDA internet traffic dataset. 3-classes classification is discovered from the dataset. They are a latency-bound pattern, a bandwidth-bound pattern, and an irregular pattern. This result suggested that deep learning is a feasible approach for network flow categorization and identification in the context of the application-aware network.

The design of the self-optimizing network using the developed model is also proposed. The design has to be highly scalable to be feasible given the size of the network traffic workload. All design decisions were described and rationalized in detail to demonstrate its scalability.

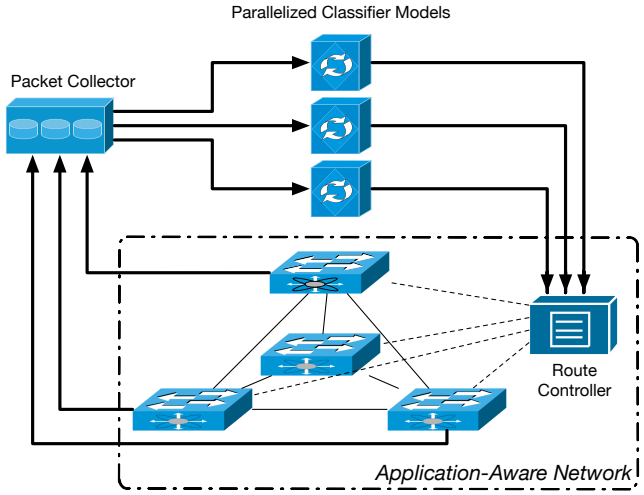


Fig. 8: Structure of the proposed design of a self-optimizing network

In the future, we plan to expand upon this work by continually improving our DNN classifier model as well as looking into the other categorization and identification techniques. The real implementation of the proposed self-optimizing network is also being developed.

Acknowledgments This work was partly supported by JSPS KAKENHI Grant Number 15K00170. The first author also gratefully acknowledges the support from the Creative and International Competitiveness Project (CICP) 2017 and the Japanese government scholarship.

References

[1] U-chupala, P., Watashiba, Y., Ichikawa, K., Date, S. and Iida, H.: Application-aware network: network route management using SDN based on application characteristics, *CSI Transactions on ICT*, pp. 1–11 (online), DOI: 10.1007/s40012-017-0171-y (2017).

[2] McKeown, N. and Anderson, T.: OpenFlow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 69–74 (2008).

[3] Ichikawa, K., U-Chupala, P., Huang, C., Nakasan, C., Liu, T.-L., Chang, J.-Y., Ku, L.-C., Tsai, W.-F., Haga, J., Yamanaka, H., Kawai, E., Kido, Y., Date, S., Shimojo, S., Papadopoulos, P., Tsugawa, M., Collins, M., Jeong, K., Figueiredo, R. and Fortes, J.: PRAGMA-ENT: An International SDN testbed for cyberinfrastructure in the Pacific Rim, *Concurrency and Computation: Practice and Experience*, No. February (online), DOI: 10.1002/cpe.4138 (2017).

[4] Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y. and Guan, Y.: Network traffic classification using correlation information, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 1, pp. 104–117 (online), DOI: 10.1109/TPDS.2012.98 (2013).

[5] Wright, C., Monrose, F. and Masson, G. M.: HMM profiles for network traffic classification, *... of the 2004 ACM workshop on ...*, pp. 9–15 (online), DOI: 10.1145/1029208.1029211 (2004).

[6] Soysal, M. and Schmidt, E. G.: Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison, *Performance Evaluation*, Vol. 67, No. 6, pp. 451–467 (online), DOI: 10.1016/j.peva.2010.01.001 (2010).

[7] Erman, J., Mahanti, A., Arlitt, M., Cohen, I. and Williamson, C.: Offline/realtime traffic classification using semi-supervised learning, *Performance Evaluation*, Vol. 64, No. 9-12, pp. 1194–1213 (online), DOI: 10.1016/j.peva.2007.06.014 (2007).

[8] Karagiannis, T., Papagiannaki, K. and Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark, *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for*

computer communications - SIGCOMM '05, Vol. 35, No. 4, p. 229 (online), DOI: 10.1145/1080091.1080119 (2005).

[9] Moore, A. W. and Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques Categories and Subject Descriptors, *Sigmetrics*, pp. 50–60 (online), DOI: 10.1145/1071690.1064220 (2005).

[10] Nguyen, T. T. T. and Armitage, G.: A survey of techniques for internet traffic classification using machine learning, *Communications Surveys and Tutorials, IEEE*, Vol. 10, No. 4, pp. 56–76 (online), DOI: 10.1109/SURV.2008.080406 (2008).

[11] Wang, Z.: The Applications of Deep Learning on Traffic Identification, *Black Hat USA* (2015).

[12] Wang, X.: A Flow-level Monitoring Middleware for Automatic Flow Categorization, Master's thesis, Nara Institute of Science and Technology (2016).

[13] Kingma, D. P. and Ba, J. L.: Adam: a Method for Stochastic Optimization, *International Conference on Learning Representations 2015*, pp. 1–15 (2015).

[14] Vincent, P., Larochelle, H., Bengio, Y. and Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders, *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 1096–1103 (online), DOI: 10.1145/1390156.1390294 (2008).

[15] Vincent Pascalvincent, P. and Larochelle Larocheh, H.: Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol, *Journal of Machine Learning Research*, Vol. 11, pp. 3371–3408 (online), DOI: 10.1111/1467-8535.00290 (2010).

[16] CAIDA: The CAIDA Anonymized Internet Traces 2016 Dataset (2016).

[17] Phaal, P., Panchen, S. and McKee, N.: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, (online), available from ⟨http://dl.acm.org/citation.cfm?id=RFC3176⟩ (2001).