The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 11 (2025) 1–6

www.elsevier.com/locate/icte

A filter-and-refine approach to lightweight application traffic classification

Ui-Jun Baek[a], Jee-Tae Park[a], Yoon-Seong Jang[a], Ju-Sung Kim[a], Yang-Seo Choi[b], Myung-Sup Kim[a,*]

[a] *Department of Computer and Information Science, Korea university, Sejong, Republic of Korea*
[b] *Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea*

Received 11 April 2024; received in revised form 27 May 2024; accepted 10 June 2024
Available online 13 June 2024

Dataset link: https://github.com/pb1069/Network-Traffic-Classification

Abstract

As application traffic becomes increasingly complex and voluminous, the need for accurate and fast traffic classification is emphasized, leading to proposals for lightweighting DL-based classifier. Nevertheless, there is still a need for faster and more accurate classification methods for practical deployment. We propose a new traffic classification mechanism using the Filter-and-Refine approach. The proposed method was evaluated public dataset using seven baselines and showed 4%p higher accuracy and about 39 times faster classification speed compared to the state-of-the-art. The source code and dataset are available at https://github.com/pb1069/Network-Traffic-Classification.

Keywords: Filter-and-refine; Application traffic classification; Lightweight; Ensemble

1. Introduction

Application traffic classification (TC) technology is one of the most important techniques in the field of network management, which distinguishes traffic generated in networks into various applications or services. By identifying and classifying application traffic, network administrators can understand traffic patterns, manage network resources efficiently, and detect malicious applications or attacks. Traffic classification technology has evolved from traditional methods such as port-based classification, deep packet inspection (DPI), and behavior-based analysis to methods utilizing machine learning (ML) and deep learning (DL). Recently, a method applying pre-training, which learns prior information of traffic modality and transfers it to various downstream tasks, has been proposed, showing high performance on various datasets. However, despite these achievements, applying the TC model to real-world networks still requires meeting desiderata such as effectiveness, deployability, trustworthiness, robustness, and adaptivity [1].

One of the important challenges related to the aforementioned desiderata is real-time processing, which requires ensuring high accuracy while enabling fast classification. The importance of real-time processing as a significant challenge stems from the rapid growth of internet traffic. Gitnux [2] mentions that global internet traffic is expected to grow at a compound annual growth rate (CAGR) of 24% between 2021 and 2026, while Cloudflare [3] states that globally, Internet traffic grew 25% in 2023. Due to the rapid growth of internet traffic, network administrators need to adopt cost-effective classification models, leading to a rise in research about lightweight. According to our investigation, Research on lightweight techniques can be broadly divided into two categories: the first is feature optimization, and the second is model compression. Feature optimization is a method of increasing the speed of classification by removing or compressing unnecessary features in classification, utilizing expert knowledge, feature compression and removal techniques. Recently, methods utilizing attention mechanisms to remove features have also been proposed. Model compression encompasses techniques such as pruning, quantization, knowledge distillation, and neural architecture search, which reduce model size by eliminating weight redundancy or enhancing efficiency. Many studies aim to improve classification speed by applying various lightweight techniques, sacrificing some accuracy in the process. This reflects the trade-off relationship between accuracy and classification speed. While it is generally difficult to improve both accuracy and classification speed simultaneously, continuous exploration in this area is necessary.

In this paper, we propose a sequential traffic classification method that can improve both accuracy and classification

* Corresponding author.
E-mail address: tmskim@korea.ac.kr (M.S. Kim).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2024.06.003

Table 1
List of abbreviations and their meanings

Acronym	Meaning
1D	One Dimensional
2D	Two Dimensional
BERT	Bidirectional Encoder Representations from Transformers
CAGR	Compound Annual Growth Rate
CAM	Class Activation Mapping
CNN	Convolutional Neural Network
DL	Deep Learning
DPI	Deep Packet Inspection
ET-BERT	Encrypted Traffic-BERT
HAST	Hierarchical Spatial–Temporal features
IP	Internet Protocol
ISCX	Information Security Center of Excellence
LSTM	Long Short-Term Memory
ML	Machine Learning
SAM	Self-Attentive Method
SOTA	State-Of-The-Art
TC	Traffic Classification
Tor	The Onion Router
XAI	eXplainable Artificial Intelligence
XGB	eXtreme Gradient Boosting

speed. Inspired by the Filter-and-Refine approach, our research involves a sequence of DL and ML models of various sizes to classify traffic. To enhance the overall classification speed, multiple models are deployed at each stage based on their size, and they determine whether to classify the flow or pass it to the next stage according to the inspection conditions at each stage. The proposed method was evaluated on publicly available application datasets, achieving an average 38 times faster speed while also improving accuracy by 3.9 percentage points. Our contributions are as follows:

Effective classification. The proposed method enables effective classification by employing sequential classifier deployment and applying unique conditions checks, ensuring high accuracy and fast classification. It demonstrates superior performance compared to the state-of-the-art (SOTA).

Enhancing robustness. The sequential deployment of multiple models and condition checks based on model reliability help prevent overfitting and improve the generalization capability of the classification system.

Improving deployability. The proposed method can be vertically (Replacing with lightweight or heavyweight classification models) or horizontally (Adding and removing classification models) scaled according to the physical constraints of various networks (e.g., time, memory, resources).

The remaining sections of the paper include related works, dataset descriptions, the proposed method, experimental results, and conclusion. Acronyms used in this article are listed in Table 1.

2. Related works

The purpose of the Filter-and-Refine method is to remove a significant portion of irrelevant data in a computationally efficient manner, thereby exchanging some computations in the refinement step for greater computational time savings in the filter step [4]. [5] proposed an approximate subsequence

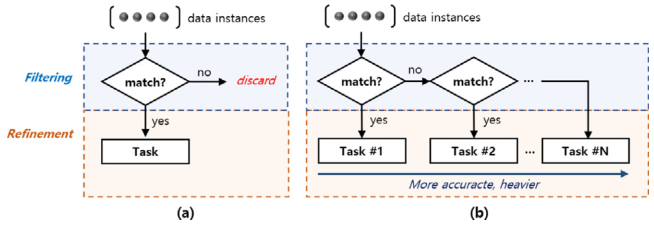


Fig. 1. The difference between the original approach and the proposed method; (a) Original approach, (b) Proposed method.

matching method with Filter-and-Refine applied to improve the query speed of subsequence matching in databases. The proposed method filters out most of the subsequences that do not match the submitted query string using a short index sequence generated by hashing the data sequences, significantly reducing the time for subsequence matching (i.e., refinement step) of the data sequences.

While our approach is inspired by the Filter-and-Refine approach, it differs in perspective from the original approach, as described in Fig. 1. For example, in [5], the task is to retrieve the necessary data from a large dataset, whereas in application traffic classification, inference results are required for all input data. Therefore, in the application traffic classification task, inference results are accepted for data that meet the conditions, while the remaining data that do not meet the conditions pass through the filter. The data that pass through the filter undergo the same process in subsequent models, and ultimately, the inference results of all data that pass through all filters follow the inference results of the heaviest model, which performs best. In the field of application traffic classification, there are two studies similar to Filter-and-Refine approach. [6] introduces the Waterfall, which is a staged classifier. Each flow undergoes inspection for predefined features at each stage, and if the conditions are met, the classification result is outputted. If the conditions are not met, the flow moves to the next stage and undergoes the same process. [7] proposed a Chain architecture where two modules sequentially perform classification. In the first stage of Chain, a classifier that performs consensus based on the results of port-based classification and ML classification is deployed, and in the second stage, a DPI-based classifier is deployed. Chain achieved a similar classification accuracy to baselines but led to a 45% improvement in classification speed. Both studies ensure high classification speed by sequentially deploying classifiers and terminating the classification when consensus conditions are met at each stage. However, neither study considers the reliability of the deployed classifiers, and they utilize overly simplistic structures or features that are unsuitable for handling complex traffic.

To address this issue, the method proposed in this paper utilizes multiple classifiers that extract various features to handle diverse characteristics of traffic, achieving fast and accurate classification through an appropriate combination of ML and DL models.

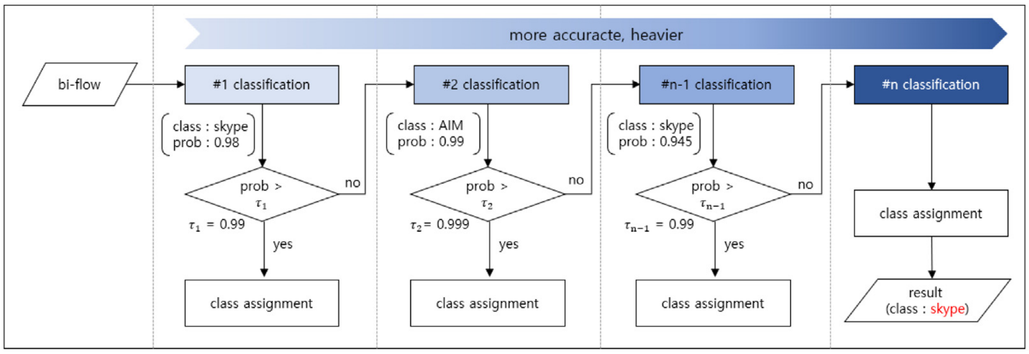


Fig. 2. An example of the proposed method for application traffic classification.

3. Proposed method

3.1. Overall operation

The proposed method is a multi-stage modular architecture consisting of a combination of DL-based models and ML-based models. Each ML-based model and DL-based model is arranged and placed based on its computing resources (i.e., classification speed) in the proposed architecture, and to evaluate the effectiveness of the architecture, publicly available DL models were used. The overall operation of the proposed method is illustrated in Fig. 2.

The bi-flow is first input to the lightest classification model, which outputs the predicted class and the probability of belonging to that class. Subsequently, the bi-flow is filtered or passed through based on whether the probability value for the predicted class exceeds a predefined threshold τ. In the example shown in Fig. 2, the bidirectional flow did not satisfy the conditions of stages 1 through $n-1$; as a result, it is assigned the class output by the model placed at the rearmost stage.

3.2. Baselines

We adopted six publicly available DL-based baselines and implemented one ML-based model for the evaluation of our proposed method. Wang et al. proposed four DL-based models for application or malicious traffic classification, including 2D-CNN [8], 1D-CNN [9], HAST-1 and HAST-2 [10]. [8] presents a simple 2D-CNN consisting of two convolution layers with pooling layers and one fully-connected layer. It extracts two-dimensional spatial features from consecutive packet bytes within a flow to classify applications. [9] presents a simple 1D-CNN consisting of two convolution layers with pooling layers and one fully-connected layer. It extracts one-dimensional spatial features from consecutive packet bytes within a flow to classify applications. HAST-1 [10] uses one-hot encoding for consecutive packet bytes in a flow and then extracts one-dimensional spatial features through 1D-convolution. HAST-2, on the other hand, one-hot encodes each packet byte within a flow and then extracts a 1D-spatial feature time series through 1D-convolution. The LSTM layer then generates temporal features from the spatial feature time series, which is used to predict the class to which the application belongs through a fully-connected layer. Xie et al. [11] proposed SAM, an application classifier that treats the initial 40 bytes of network packets as a language and applies the self-attention mechanism. SAM consists of one self-attention layer and one 1D-convolution layer. Lin et al. [12] introduced ET-BERT, which pretrains on large-scale traffic data using BERT [13]. The pretrained model is evaluated through transfer learning on various downstream tasks. To use the output values of the DL models in the filtering process, a sigmoid function is added to the back of the baseline models to normalize the output values to the range of 0 to 1, and these values are used in the filtering process.

The ML-based baseline uses XGB [14], which is still widely used due to its strong classification performance, and the XGB is used with key parameters such as n_estimators set to 100, objective set to binary:logistic, and booster set to gbtree. Additionally, XGB outputs logits, which represent the probability of belonging to each class, and these are used in the filtering process.

3.3. Datasets

To evaluate the proposed method, two publicly available datasets are used. The first is the ISCX-VPN 2016 dataset [15], which contains 20 applications and includes encapsulation task, application type task, and application task. The second dataset is the ISCX-Tor 2016 dataset [16], which includes 19 applications and has tasks identical to those of the ISCX-VPN 2016. We preprocessed both datasets according to predefined rules [17] with an additional rule applied to remove flows with a destination IP of 255.255.255.255., extracting 8764 flows from the ISCX-VPN 2016 dataset and 7402 flows from the ISCX-Tor 2016 dataset, and split them into training and testing datasets in a 7:3 ratio.

4. Experiments

4.1. Overall evaluation

This section provides a comparison of the proposed method and baselines for the application type task of the ISCX-VPN 2016 dataset. Table 2 compares the accuracy, f1-score,

Table 2
Results of application type classification in ISCX-VPN 2016.

Model		XGB [14]	2DCNN [8]	1DCNN [9]	SAM [10]	HAST1 [11]	HAST2 [12]	ETBERT [13]	Accuracy	f1-score	Precision	Recall	Inference time (ms)
Baselines		✓							0.853	0.852	0.855	0.853	15
			✓						0.791	0.788	0.788	0.791	113
				✓					0.806	0.805	0.805	0.806	113
					✓				0.812	0.812	0.812	0.812	146
						✓			0.867	0.859	0.861	0.861	1646
							✓		0.847	0.847	0.848	0.847	2384
								✓	0.846	0.845	0.846	0.846	6663
Use all model	Threshold	0.87	0.86	0.95	0.89	0.85	0.82	✓	0.881	0.881	0.882	0.881	659
	Accuracy	95%	82%	68%	61%	0%	–	**52%**					
	Coverage	71%	12%	2%	8%	1%	–	**5%**					
Best feature	Threshold	0.87	0.86	0.95	0.89	✓			**0.886**	**0.887**	**0.886**	**0.886**	**171**
	Accuracy	95%	82%	68%	61%	59%							
	Coverage	71%	12%	2%	8%	5%							

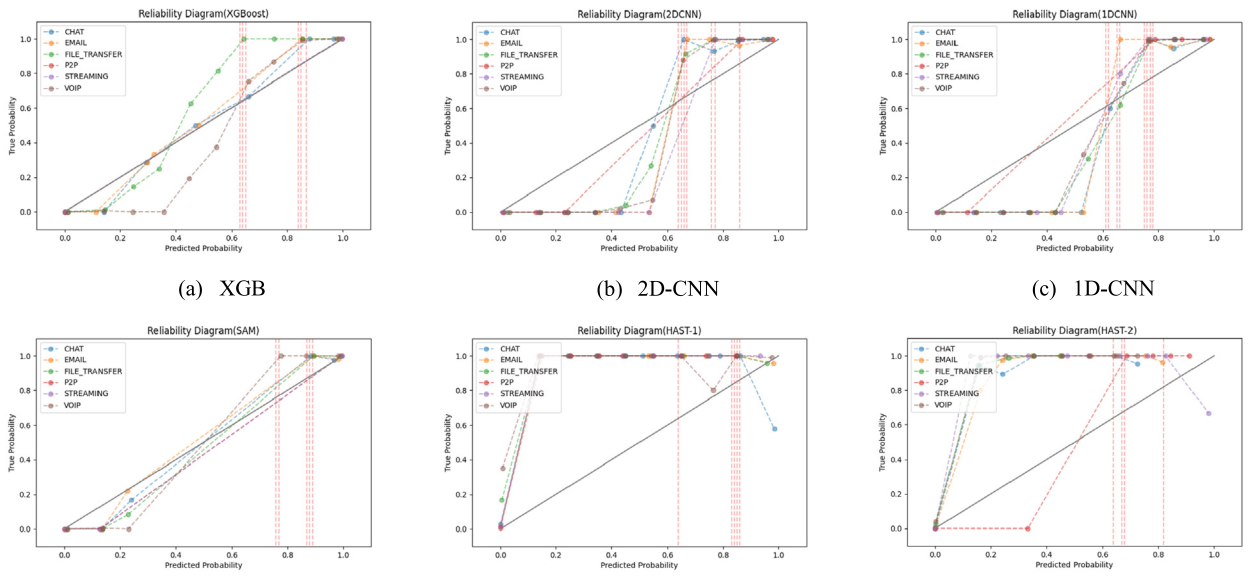
Table 3
Comparison of classification performance of classification models by dataset and task.

Dataset	Task	Model	Accuracy	Inference time (ms)
ISCX-VPN 2016	Application type	ET-BERT	0.846	6663
		best feature	0.886	171
	Application	ET-BERT	0.812	6663
		best feature	0.851	298
ISCX-Tor 2016	Application type	ET-BERT	0.944	5626
		best feature	0.961	217
	Application	ET-BERT	0.903	5626
		best feature	0.93	336

precision, recall, and inference time between seven baselines and two architectures for the task of application type classification. One of the two architectures shown in Table 2 is an architecture where all models are placed, while the other is an architecture where heavy models placed at the rear are removed. In the table, "√" marks the last classification model used, and real values represent the threshold applied to each model. For example, in the best feature architecture, XGB first performs classification for all flows, assigning classes only to flows with a class probability of 0.87, and passes the rest to 2D-CNN. Models including 2D-CNN follow a similar process, and the final classification is completed by HAST-1. Threshold setting methods are described in Section 2. The architecture using all models achieves approximately 4% higher accuracy and 10 times faster classification speed compared to ET-BERT. Looking at the coverage at each stage, over 90% of flows are classified by the first four models, with the remaining 6% classified by the last three models. Additionally, the accuracy drops sharply as we move to later stages. Even in the case of ET-BERT, the accuracy for classifying the remaining 5% of flows barely exceeds 50%. This indicates that flows remaining after classification in each model are challenging to classify, and if accurate classification is not possible, it might be better to quickly classify them with lighter models. The best feature architecture removes the two heavy models at the rear, resulting in approximately 3.8 times faster classification and a slight increase in accuracy. Furthermore, the best feature architecture achieves about 4% higher accuracy and 38 times faster classification speed compared to ET-BERT. Table 3 provides overall comparison of classification performance of classification models by datasets and tasks. There was a significant improvement in classification accuracy not only for the ISCX-VPN 2016 dataset but also for the ISCX-Tor 2016 dataset, along with a substantial enhancement in classification speed.

4.2. Analysis of model reliability and threshold setting

In an architecture with multiple models, the threshold of one model can influence the inference results of other models. Therefore, careful threshold setting is necessary, but this is a challenging task. We propose a method to set the threshold based on the reliability of each model. Fig. 3 shows the reliability diagram of all models except for ET-BERT. The reliability of a deep learning model ensures that users can trust the model's outputs, while confidence indicates how certain the model is about its outputs. This is illustrated in a reliability diagram that evaluates how well-calibrated the predicted probabilities of a classification model are. A perfectly calibrated model is represented by a diagonal line, meaning that if the model predicts a class with 70% probability, the frequency of successful predictions for that class should be 70%. According to the results predicted by SAM for VoIP flows, all flows within the set predicted with over 80% probability were successful, indicating that the model is "overconfident" in its predictive ability. If the threshold is set low, the model will have high acceptance capacity and low accuracy; if set high, it will have low acceptance capacity and high accuracy. The proposed method aims to maximize the actual prediction accuracy by selecting a threshold candidate where the model's confidence slope increases and then decreases, represented by the red vertical lines in each plot. Although the proposed architecture aims for efficiency, it must guarantee high accuracy,



(a) XGB (b) 2D-CNN (c) 1D-CNN

(d) SAM (e) HAST-1 (f) HAST-2

Fig. 3. Reliability diagram for baseline models with train dataset for application type task; The red vertical dashed lines represent threshold candidates based on the slope of the average True Probability. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

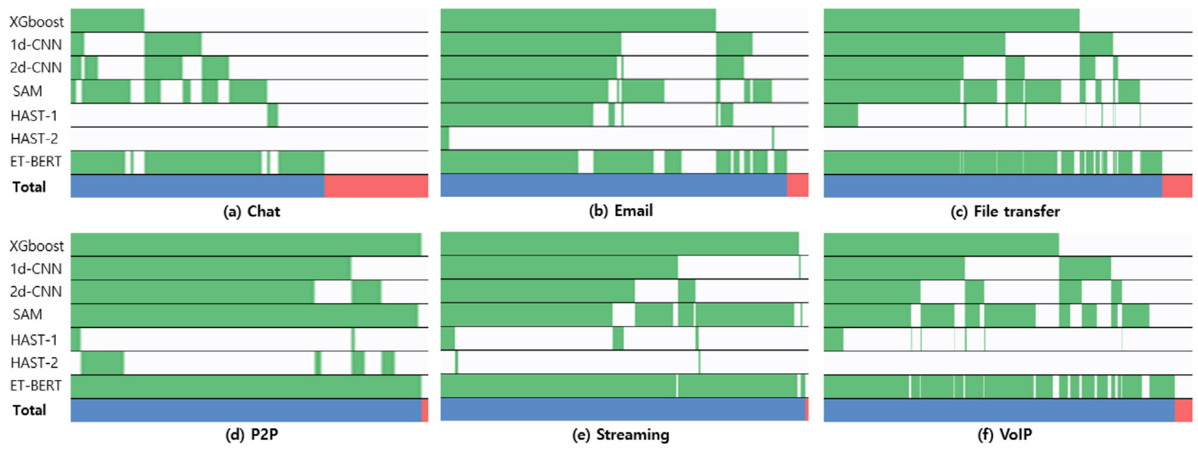


Fig. 4. Comparison of baseline model (w/ threshold) coverage rates by application type.

so the last candidate among the derived threshold candidates is set as the final threshold.

4.3. Analysis of model coverage

This section provides an analysis of the models' coverage. Fig. 4 visualizes the flows that each model correctly predicted with the applied threshold for each application type. The x-axis represents the index of each flow, and the distances between flows are adjusted for convenience. Overall, the error rate is highest for the Chat type, while for the other application types, over 90% of flows are accommodated by at least one model. From a model perspective, it should be noted that ET-BERT has the highest accuracy, but this is considering the results without applied thresholds. XGB shows high acceptance rates in most application types, making it an excellent pre-classifier for less challenging flows. HAST-1 and HAST-2 do not exhibit high accuracy in most application types. However, HAST-1 can classify flows that are not classified correctly by all models in the Chat type. Similarly, SAM can classify some flows that are not correctly classified by other models, which is observable in the Email type and File Transfer. This suggests that the combination of models using different feature extraction methods can enhance the generalization ability of the model.

5. Conclusion

This paper proposes a lightweight method and architecture for sequential deployment and classification of ML and various DL models. Our research is inspired by the filter-and-refine method, successfully achieving lightweighting by processing large amounts of data through lightweight models and handling the remaining, more challenging data in heavier models at the rear. Additionally, we propose a method for generating initial thresholds by analyzing model reliability during the training process to ensure reliable classification by the forward models. The proposed method was implemented and compared considering seven baselines, and evaluated on the publicly available dataset ISCX-VPN 2016. As a result, the best feature architecture showed 4% higher accuracy and 38 times faster classification speed compared to the ET-BERT in application type classification tasks, and 3.9% higher accuracy and approximately 22 times faster classification speed in application classification tasks. The proposed method not only improves accuracy and speed but also greatly reduces the dependency on accuracy–speed trade-offs, making it highly scalable through the application of models with various characteristics.

Our future research can be summarized into two main points. First, by applying XAI to each classifier, we aim to gain insights into classification results, providing explainability and reliability to network operators. Additionally, feature analysis and selection processes using XAI techniques such as class activation mapping (CAM) [18,19] and self-attention mechanisms can improve the efficiency and generalization performance of each classifier. Second, optimizing the classification order is crucial as the order significantly impacts accuracy and classification speed. While our proposed method prioritizes classification speed by deploying faster models first, the classification order can be adjusted or optimized based on various requirements and constraints.

CRediT authorship contribution statement

Ui-Jun Baek: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Jee-Tae Park:** Validation, Resources, Data curation. **Yoon-Seong Jang:** Software, Resources, Data curation. **Ju-Sung Kim:** Software, Resources, Investigation, Data curation, Conceptualization. **Yang-Seo Choi:** Funding acquisition, Writing – review & editing, Data curation, Supervision, Validation. **Myung-Sup Kim:** Validation, Supervision, Project administration, Funding acquisition, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and dataset are available at https://github.com/pb1069/Network-Traffic-Classification.

Acknowledgments

This work was supported by Institute of Information and communications Technology Planning and Evaluation (IITP) funded by the Korea government (00235509, Development of security monitoring technology based network behavior against encrypted cyber threats in ICT convergence environment), and was supported by the “Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-004)

References

[1] Giuseppe Aceto, et al., AI-powered internet traffic classification: Past, present, and future, IEEE Commun. Mag. (2023).
[2] Must-know internet traffic statistics, 2023, Accessed 14 March 2024, from https://gitnux.org/internet-traffic-statistics/.
[3] Cloudflare 2023 year in review, 2023, Accessed 14 March 2024, from https://blog.cloudflare.com/radar-2023-year-in-review.
[4] W. Jordan, Filter and Refine Strategy, Springer US., Boston, MA, 2008, p. 320, http://dx.doi.org/10.1007/978-0-387-35973-1_415.
[5] Beng Chin Ooi, et al., Fast filter-and-refine algorithms for subsequence selection, in: Proceedings International Database Engineering and Applications Symposium, IEEE, 2002, pp. 243–254.
[6] Paweł Foremski, Christian Callegari, Michele Pagano, Waterfall: Rapid identification of IP flows using cascade classification, in: Computer Networks: 21st International Conference, CN 2014, BrunÓw, Poland, June 23-27, 2014. Proceedings 21, Springer International Publishing, 2014, pp. 14–23.
[7] Hossein Doroud, et al., Speeding-up dpi traffic classification with chaining, in: 2018 IEEE Global Communications Conference, GLOBECOM, IEEE, 2018, pp. 1–6.
[8] Wei Wang, et al., Malware traffic classification using convolutional neural network for representation learning, in: 2017 International Conference on Information Networking, ICOIN, IEEE, 2017, pp. 712–717.
[9] Wei Wang, et al., End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics, ISI, IEEE, 2017, pp. 43–48.
[10] Wei Wang, et al., HAST-IDS: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection, IEEE Access 6 (2017) 1792–1806.
[11] Guorui Xie, Qing Li, Yong Jiang, Self-attentive deep learning method for online traffic classification and its interpretability, Comput. Netw. 196 (2021) 108267.
[12] Xinjie Lin, et al., Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 633–642.
[13] Jacob Devlin, et al., Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[14] XGBoost documentation, 2022, Retrieved 1 April 2024 from https://xgboost.readthedocs.io/en/stable/.
[15] Gerard Drapper Gil, et al., Characterization of encrypted and VPN traffic using time-related features, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, SciTePress, 2016, pp. 407–414.
[16] Arash Habibi Lashkari, et al., Characterization of tor traffic using time based features, in: International Conference on Information Systems Security and Privacy, SciTePress, 2017, pp. 253–262.
[17] Ui-Jun Baek, et al., Preprocessing and analysis of an open dataset in application traffic classification, in: 2023 24st Asia-Pacific Network Operations and Management Symposium, APNOMS, IEEE, 2023, pp. 227–230.
[18] Bolei Zhou, et al., Learning deep features for discriminative localization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2921–2929.
[19] Ramprasaath R. Selvaraju, et al., Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.