Identifying DDoS Attacks in-Kernel via eBPF/XDP and Knowledge Distillation

Kezhuo Chen
School of Software Engineering
Huazhong University of Science and Technology
Wuhan, China
M202176561@hust.edu.cn

Ding Yuan
School of Software Engineering
Huazhong University of Science and Technology
Wuhan, China
ragazzo_27@qq.com

Dehong Qiu
School of Software Engineering
Huazhong University of Science and Technology
Wuhan, China
qiudehong@hust.edu.cn
*Corresponding author

***Abstract*—Identifying Distributed Denial of Service (DDoS) plays a vital role in network security as DDoS attacks have grown rapidly and become one of the most serious threats to network security. However, most existing DDoS identification methods run in user space on hardware servers, which increases the traffic load and delays the response. In this work, we propose a method for fast identification of DDoS attacks in the Linux kernel. This method uses the eXpress Data Path (XDP) as a hook to attach an extended Berkeley Packet Filter (eBPF) program to the device driver to process data packets at a very early stage and at high speed. Furthermore, to break through the computing/memory limitations of the devices and ensure the accuracy of DDoS attack identification, we use Knowledge Distillation (KD) techniques to transform the DDoS identification knowledge of a complex Multi-Layer Perceptron (MLP) into a simple Decision Tree (DT) model, and implement the DT model as an eBPF program. Experimental results show that the proposed method can quickly and accurately identify DDoS attacks. Compared to the baseline models, the Macro F1 Score has increased by 1.1%, reaching 97.6%.**

Keywords—DDoS attack, eBPF/XDP, in-kernel identification, knowledge distillation

I. INTRODUCTION

A DDoS attack is a cybercrime in which the attacker floods a server with internet traffic to prevent users from accessing connected online services and sites. Since the first known DDoS attack occurred in July 1999 [1], targeting the University of Minnesota's computer network, DDoS attacks have become one of the most harmful threats to network security. In 2024, DDoS attacks saw a significant increase in frequency and intensity, with some reports indicating a 49% rise quarter-over-quarter and a 55% increase year-over-year [2]. Identifying DDoS attacks is an important task that is critical in network security.

Due to its importance, identifying DDoS attacks has attracted much attention since the first appearance of DDoS. Traditional methods for identifying DDoS attacks generally include two steps: first, statistically analyzing the characteristics of the data packets, and then determining whether it is an attack. For example, the features like packet rate, size, source/destination IPs, and protocol types are widely used to differentiate between normal network traffic and malicious DDoS traffic. The machine learning classifiers, such as support vector machine (SVM) [3], Decision Tree (DT) [4] and Neural Networks (NN) [5, 6] are widely used to identify DDoS attacks. These traditional methods usually run in user space on hardware servers. Although they have achieved certain results, they also have some obvious shortcomings. For example, these methods deployed on servers increase data traffic load and cannot response in time. Besides, the accuracy of these methods is not satisfactory because dropping a large number of packets will affect the statistical characteristics of the data traffic load.

Recently, with the rapid development of network programmability [7, 8], researchers have been trying to efficiently check and filter arriving data packets directly in-kernel to guarantee the high-speed processing. Network programmability refers to the ability to control and manage the forwarding of data by utilizing programmable switches and interfaces. One major advantage brought about by network programmability is that programmable networks can be configured to mitigate security threats and implement security policies more effectively. Among the proposals made to realize network programmability, eBPF/XDP is the most highlighted [9]. XDP enables the execution of eBPF programs within the Network Interface Card (NIC) driver level, which makes it possible to identify DDoS attacks at high speed in-kernel.

However, identifying DDoS attacks in kernel is significantly impacted by hardware limitations. Specifically, the memory space is limited; Only simple instructions like integer additions and bit shifts are allowed; Unbounded loops are forbidden; The program size is limited; Decimal numbers are not supported, and so on. These restrictions induce challenges to the deployment of complicated DDoS attack identification models in kernel.

To overcome these challenge, this work proposes to leverage the programming capabilities offered by eBPF with the high-performance of XDP hook to execute a DDoS attack identification DT model in the kernel space. With this design, XDP establishes high performance data paths within the kernel and processes packets closer to the NIC, realizing fast packet processing, while the DDoS attack identification DT model is executed as a kernel module. To guarantee the performance of the DT model, we use KD technology that distill the knowledge of a MLP built for DDoS attack identification into the DT model. The KD is a machine learning technique where knowledge from a larger, more complex model (the "teacher") is transferred to a smaller, simpler model (the "student"). This allows the student model to achieve performance comparable to the teacher [10]. We first build an MLP in user space and train it with historically collected DDoS attack data to achieve a sufficient DDoS attack identification accuracy. Subsequently, we use the soft labels

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61873100.

generated by the MLP to generate train data with mixed labels to train the simple DT model. As a result, the DT model performs well on the task of DDoS attack identification. Extensive experiments demonstrate that the proposed in-kernel method achieves better accuracy in DDoS attack identification than the compared baseline models.

II. Related Work

DDoS attack identification has attracted attention due to its importance to network security. A large number of diagnosis methods have been proposed, such as statistical methods and machine learning-based methods. The paper [11] gives a good comprehensive overview of the study of DDoS attack identification. Below we present a brief overview of the machine learning-based methods that are closely related to our method.

With the rapid development of machine learning, it has been proven to be one of the most effective methods for identifying DDoS attacks. Various machine learning methods, for example, SVM[3], DT[4], NN [5, 6], CNN[12], and GRN[13, 14] have been widely utilized to DDoS attack identification. Although promising results have been produced, it is also found that machine learning-based methods suffer some limitations. For example, they need a huge number of labeled training data. Additionally, they are usually complex and require a lot of computing resources. Therefore, they can only be deployed in servers at the user end, which limits their use in identifying DDOS attacks in high-speed network environments such as data centers and cloud infrastructure.

The advent of programmable network devices and the eBPF/XDP technology enables high-speed programmable packet processing in the kernel at runtime [7-9]. However, implementing complex models in such an environment with limited computing resources poses enormous challenges. The KD is effective for model compression, and has been widely used in many fields, including DDoS identification [15-19]. For example, Mousika [19] enables the knowledge translation from a teacher model NN to a student model BDT to tackle the drawbacks of offloading models to the switch. While the future is promising, it remains difficult to deploy sophisticated machine learning models in kernel due to the inherent constraints of network devices, such as lack of support for floating-point operations. In this work, we distill the knowledge of a MLP and use the knowledge to generate a training set with mixed labels to train a small DT. The depth of the DT is well controlled, so it can be easily deployed in the kernel and identify DDoS attacks with satisfactory performance.

III. Background and Motivation

A. eBPF/XDP Technology

The eBPF [9] is a revolutionary technology that allows programs to run safely and efficiently within the Linux kernel. The eBPF system is composed of a series of components, such as the verifier, Just-In-Time (JIT) compilation engine, maps and helper functions, as illustrated in Figure 1.

An eBPF program is written in a high-level language (mainly restricted C). The clang compiler transforms it into an ELF/object code. An ELF eBPF loader can then insert it into the kernel using a special system call. During this process, the verifier analyzes the program and upon approval the kernel performs the dynamic translation. Maps are generic key-value stores available to eBPF programs. An eBPF program just specifies the size of the key and the size of the value at map-creation time. The helper functions are special functions offered by the kernel infrastructure to enable an eBPF program to perform tasks such as interacting with maps, modifying packets, and printing messages to the kernel trace, and so on.

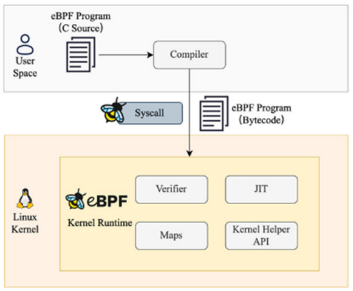

Fig. 1. eBPF components and workflow.

To execute an eBPF program, it is necessary to attach it to an interface that allows custom programming. This interface is called a hook. The XDP is a type of hook, which is present only on the RX path, inside a device's network driver and allows packet processing at the earliest point in the network stack, as illustrated in Figure 2.

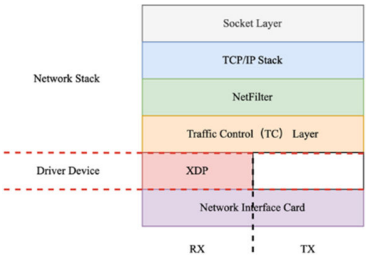

Fig. 2. Linux kernel network stack.

An eBPF program executes whenever a registered event, for example, sending or receiving a packet, occurs. In the hook of XDP, an eBPF program can take quick decisions about incoming packets and also performing arbitrary modifications on them, avoiding additional overhead imposed by processing inside the kernel. After processing a packet, an XDP program returns an action, which represents the final verdict regarding what should be done to the packet after the program exits.

From the above statement, it can be seen that the eBPF program attached to the hook of XDP inside a device's network driver is one of the best choices in terms of performance speed for applications such as DDoS attack identification.

B. Knowledge Distillation

The KD is a machine learning technique that aims to transfer the learnings of a large pre-trained model, the "teacher model," to a smaller "student model." The KD can be generally divided into the following three categories: response-based KD, feature-based KD, and relation-based KD. Among them, response-based KD trains a smaller, faster student model to produce predictions that closely match a larger, more complex teacher's predictions,

making it more efficient for deployment without a significant loss in accuracy. Figure 3 shows the response-based KD model.

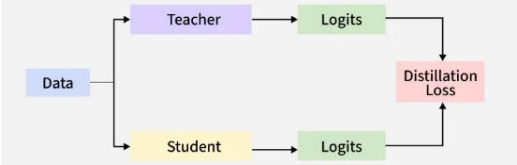

Fig. 3. The generic response-based knowledge distillation

The student model is trained by minimizing a loss function that measures the difference between the outputs of the teacher and the student. Given a vector of logits z_t as the outputs of the teacher model, the distillation loss for the response-based KD can be formulated as

$$L_{ResD}(z_t, z_s) = \mathcal{L}_R(z_t, z_s), \quad (1)$$

where $\mathcal{L}_R(\,)$ indicates the divergence loss of logits, and z_t and z_s are logits of teacher and student, respectively.

The student model is typically smaller, simpler, faster to infer, and easier to deploy. Since the response-based KD is simple yet effective for model compression, it has been widely used in different applications, including the in-kernel DDoS identification that needs simple and effective models.

IV. Methodology

A. Architecture Overview

Figure 4 plots the architecture of our method of in-kernel DDoS identification, which contains three modules, that is, packet filter, feature extractor, and DDoS Detector. These three kernel modules perform their respective functions and share data in kernel through eBPF Maps. In the rest of this section, we describe each module in detail.

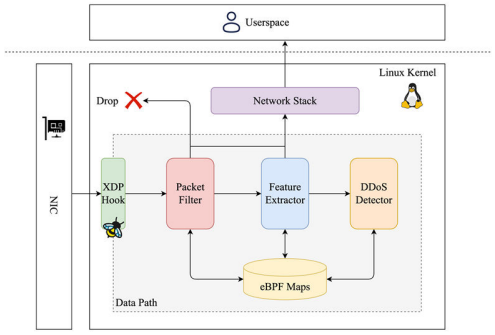

Fig. 4. The architecture of the in-kernel DDoS identification model

B. Packet Filter

The XDP program is event-triggered, it executes whenever an registered event occurs. When a network packet arrives at the network interface, the XDP program attached to the network driver intercepts the data packet, which provides a powerful and efficient way to gain insights into network traffic and enforce network policies at a very low level in the Linux kernel. The module of packet filter examines network packets and allows users to create rules to filter traffic based on IP addresses, MAC addresses, and ports. Based on the user-defined filtering rules, the module of packet filter decides whether to drop or redirect or pass the packets. The malicious traffic associated with DDoS attacks can be quickly identified and dropped at a very early stage in the network stack, preventing it from overwhelming the system. Compared to the packet filters to defend against DDoS attacks in user space, the XDP filters require less resources and can process network packets at a much higher rate.

C. Feature Extractor

The task of the feature extractor is to calculate the characteristics of data traffic. Since data packets travel independently across the network, taking different paths and potentially arriving at different times, the arrived data packets are reassembled in the correct order as Figure 5 shows, which is crucial to extracting the characteristics of traffic flow correctly. The feature extractor computes the following five types of features: 1) Flow Duration, 2) Flow Rate, 3) Packet Size, 4) Flow Inter Arrival Time and 5) TCP protocol flags like FIN, SYN, RST, and PSH. For each of the 1)-4) types of features, their Max, Min, Mean, and Variance are calculated as follow:

$$Max_n = \max(Max_{n-1}, X_n), \quad (2)$$

$$Min_n = \min(Min_{n-1}, X_n), \quad (3)$$

$$Mean_n = Mean_{n-1} + \frac{(X_n - Mean_{n-1})}{n}, \quad (4)$$

$$V_n = \frac{n-1}{n^2}(X_n - Mean_{n-1})^2 + \frac{n-1}{n}V_{n-1}, \quad (5)$$

where, the subscript n means the $n-th$ data packet. X_n means the corresponding statistics of the $n-th$ data packet. It can be observed that there is no need to store the information of the first (n-1) packets, only the current statistics need to be kept, and the subsequent statistics can be computed by recursion. Therefore, the network flow features can be extracted at a linear rate.

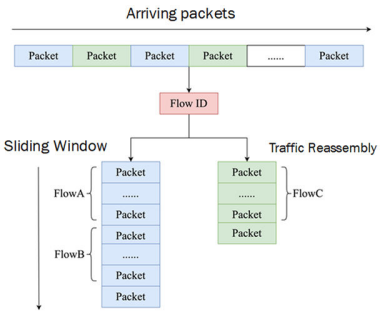

Fig. 5. Network traffic reassembly

D. DDoS Detector DT-MLP

As stated before, the complex learning models such as neural networks cannot be deployed directly in the kernel due to the computational and memory limitations of NIC hardware. The DT model, which is a rule-based learning classifier [20], is one of the most suitable models for deployment in the kernel. However, the learning capability of DT is not as powerful as other complicated models such as MLP. To obtain a DT model with satisfied performance, we distill the knowledge of a high quality MLP into a DT, as Figure 7 shows. Thus, we construct the DDoS detector DT-MLP.

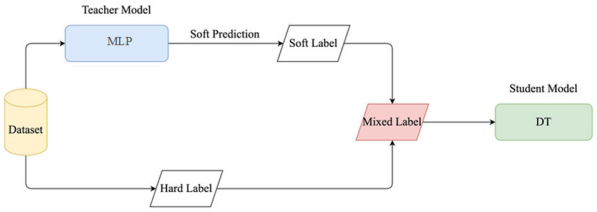

Fig. 6. The knowledge distillation from MLP to DT

The teacher model MLP is trained in user space by a given training dataset $D = (x_i, y_i^{hard})$, x_i is a feature vector and $y_i^{hard} \in \{0, 1, \dots, M\}$, 0 represents the normal flow and $1, \dots, M$ are the labels of different DDoS attacks. The output of MLP is a soft label vector y_i^{soft} for a given input x_i, whose $j-th$ element $y_i^{soft}(j)$ means the probability of x_i belonging to the $j-th$ class of DDoS attack. Furthermore, we define the mixed label y_i^{mixed} as below by combining y_i^{hard} and y_i^{soft}.

$$y_i^{mixed} = \frac{y_i^{soft} + k * y_i^{hard}}{1 + k} \tag{6}$$

where, $k \geq 0$ is used to balance the two kinds of labels.

Subsequently, we use the training dataset $\widehat{D} = (x_i, y_i^{mixed})$ to train the student model, which is a Classification And Regression Tree (CART). CART uses Gini impurity to determine the best features and split points for each node. The purpose of training is to continuously reduce Gini impurity.

$$Gini(\widehat{D}) = 1 - \sum_{j=1}^{M} \left(\sum_{i=1}^{N} \frac{y_i^{mixed}(j)}{N} \right)^2 \tag{7}$$

To deploy the trained CART in kernel efficiently, we represent each node of CART as a quintuple $\{left, right, feature, threshold, value\}$, where left and right represent its left and right child node respectively; feature is the index of characteristics; threshold is the split threshold; For a leaf node, value is the label of DDoS attack; Otherwise, value is set to -1 for other nodes. Thus, we can efficiently deploy the CART by using eBPF Maps. One example is shown in Figure 7.

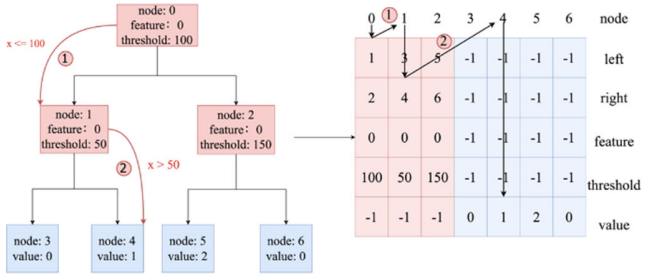

Fig. 7. The CART deployed in kernel

V. Evaluation

A. Bataset, Baseline and Metrics

we evaluate the model DT-MLP on the open Intrusion Detection Evaluation Dataset (CICIDS2017), which is a network traffic dataset comprised of both normal traffic and abnormal data caused by intentional attacks on a test network. The data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. Here, we use the DDoS attack data only. To make the DDoS attack data suitable for evaluating the in-kernel model DT-MLP, we first use the tool CICFlowMeter to transform the dataset CICIDS2017. Table I shows the details of the extracted dataset, which contains the benign traffic and the traffic of five types of DDoS attacks.

TABLE I. The details of the dataset

Traffic	The number of Traffic
BENIGN	364682
DoS Hulk	151059
DDoS LOIT	92304
DoS Slowhttptest	5279
DoS Slowloris	5598
DoS GoldenEye	7746

We compare DT-MLP with DT and MLP, and use the metrics Macro Precision, Macro Recall, and Macro F1 Score to evaluate their performance.

$$Macro\ Precision = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{8}$$

$$Macro\ Recall = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{9}$$

$$Macro\ F1\ Score = \frac{2 \times Macro\ Precision \times Macro\ Recall}{Macro\ Precision + Macro\ Recall} \tag{10}$$

B. Performance of DDoS Identification

Table II summarizes the performance of the models DT, MLP and DT-MLP in DDoS identification. It is obvious that the performance of the model DT-MLP is very close to the teacher model MLP and much better than the traditional model DT, which means the knowledge of the teacher model MLP is well transferred into the student model DT.

TABLE II. The performance of the models DT, MLP and DT-MLP

Model	Macro Precison	Macro Recall	Macro F1
DT	0.964	0.966	0.965
MLP	0.969	0.987	0.978
DT-MLP	0.966	0.987	0.976

C. Influences of hyperparameters

The parameter k in (6) is an important hyperparameters that need to be set in advance. We investigate how it affects the accuracy and the complexity of the model DT-MLP. Table III shows the results. We can observe that the Macro F1 score of the model DT-MLP with $k = 0.5$ is larger than that of the model

DT-MLP with $k = 0$ or $k = 1$. The last column of Table III shows the number of nodes of the model DT-MLP with different value of k. As the value of k increases, the number of the tree nodes also increases, which means that a more complex model is required to learn the hard labels.

TABLE III. THE INFUENCE OF THE PARAMETER K

k	Macro F1	Nodes
0	0.972	296
0.5	**0.976**	484
1	0.968	1285

D. Influences of the Depth of DT-MLP

Since the kernel has limited memory space, the model DT-MLP is expected to perform better with smaller depth. Figure 8 shows the influences of the maximum depth of the model DT-MLP on the Macro F1 score and the Log Loss. It is recommended that the maximum depth of the DT-MLP model be kept between 10 and 15.

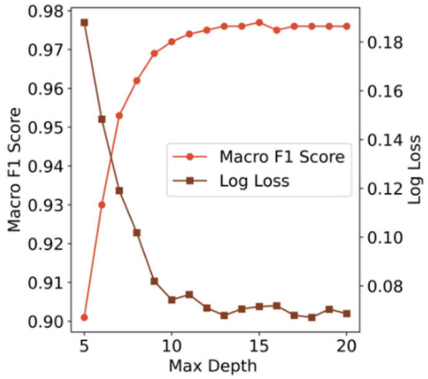

Fig. 8. The influences of the depth of DT-MLP

VI. CONCLUSION

In order to achieve high-precision and high-speed DDoS attack identification, we propose an in-kernel DDoS attack identification model DT-MLP in this paper using eBPF\XDF technology and knowledge distillation. The model DT-MLP receives the knowledge distilled form a powerful model MLP in user space and is deployed in Linux kernel as an eBPF program. As a result, the DT-MLP model not only has high accuracy close to that of the MLP model deployed in user space, but also achieves DDoS attack identification at a higher speed. Its effectiveness is demonstrated by the experiments.

In future work, we are going to deploy the model DT-MLP in practical scenarios to further evaluate its performance. Furthermore, we aim to enhance the kernel model by extracting knowledge from different teacher models. We are also interested in further compressing the size of the kernel model.

REFERENCES

[1] CERT Coordination Center, CERT Incident Note IN-99-04, 1999. https://web.archive.org/web/20081115163511/http://www.cert.org/incident_notes/IN-99-04.html.

[2] Vercara, Annual DDoS Report 2024 Trends and Insights, 2025. https://vercara.digicert.com/resources/annual-ddos-report-2024-trends-and-insights.

[3] A. Yılmaz, A. Küçüker, G. Bayrak, D. Ertekin, M. Shafie-Khah, J. M. Guerrero, "An improved automated PQD classification method for distributed generators with hybrid SVM-based approach using un-decimated wavelet transform," Int. J. Electr. Power Energy Syst. Vol.136, 107763, 2022.

[4] V.G. Costa, C.E. Pedreira, "Recent advances in decision trees: an updated survey," Artif. Intell. Rev., vol. 56, pp.4765–4800, 2023.

[5] R. Gopi, V. Sathiyamoorthi, S. Selvakumar, R. Manikandan, P. Chatterjee, N. Z. Jhanjhi, et al, "Enhanced method of ANN based model for detection of DDoS attacks on multimedia internet of things," Multimedia Tools and Applications, vol.81, no. 19, pp. 26739-26757, 2022.

[6] R. Amrish, K. Bavapriyan, V. Gopinaath, A. Jawahar, C. V. Kumar, "DDoS detection using machine learning techniques," Journal of IoT in Social, Mobile, Analytics, and Cloud, vol.4, no.1, pp.24-32, 2022.

[7] M. Dimolianis, A. Pavlidis and V. Maglaris, "Signature-based traffic classification and mitigation for DDoS attacks using programmable network data planes," IEEE Access, vol. 9, pp. 113061-113076, 2021,

[8] A. d. S. Ilha, Â. C. Lapolli, J. A. Marques and L. P. Gaspary, "Euclid: A fully in-network, P4-based approach for real-time DDoS attack detection and mitigation," IEEE Transactions on Network and Service Management, vol. 18, no. 3, pp. 3121-3139, Sept. 2021.

[9] Marcos A. M. Vieira, Matheus S. Castanho, Racyus D. G. Pacífico, Elerson R. S. Santos, Eduardo P. M. Câmara Júnior, and Luiz F. M. Vieira, "Fast packet processing with eBPF and XDP: concepts, code, challenges, and applications," ACM Comput. Surv., vol. 53, no.1, Article 16, 2019.

[10] G. Hinton, O. Vinyals, J. Dean, "Distilling the knowledge in a neural network", arXiv:1503.02531, 2015.

[11] T. E. Ali, Y. W. Chong, S. Manickam, "Machine learning techniques to detect a DDoS attack in SDN: A systematic review," Applied Sciences, vol. 13, no. 5, pp.3183-3209, 2023.

[12] H. Kumar, Y. Aoudni, G. G. R. Ortiz, L. Jindal, S. Miah, R. Tripathi, "Light weighted CNN model to detect DDoS attack over distributed scenario," Security and Communication Networks, pp.1-10, 2022.

[13] S. Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, et al, "DIDDOS: an approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU)," Future Generation Computer Systems, vol. 118, pp.453-466, 2021.

[14] A. Zainudin, L. A. C. Ahakonye, R. Akter, D. S. Kim, J. M. Lee, "An efficient hybrid-DNN for DDoS detection and classification in software-defined IIoT networks," IEEE Internet of Things Journal, vol. 10, no.10, pp. 8491-8504, 2023.

[15] X. Zhang, X. Wang, J. W. Bian, C. Shen, M. You, "Diverse knowledge distillation for end-to-end person search," in: 35th AAAI Conference on Artificial Intelligence, pp. 3412-3420, 2-9 Feb. 2021.

[16] Z. R. Wang, J. Du, "Joint architecture and knowledge distillation in CNN for Chinese text recognition," Pattern Recognition, vol. 111, 1077222021, March 2021.

[17] K. Choi, M. Kersner, J. Morton, B. Chang, "Temporal knowledge distillation for on-device audio classification," in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 486-490, 23-27 May 2022.

[18] C. Zheng, Z. Xiong, T. T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, et. al, "IIsy practical in-network classification," arXiv:2205.08243, 2022.

[19] G. Xie, Q. Li, Y. Dong, G. Duan, Y. Jiang and J. Duan, "Mousika: enable general in-network intelligence in programmable switches by knowledge distillation," in: IEEE Conference on Computer Communications, pp. 1938-1947, London, United Kingdom, 02-05 May, 2022.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees, Wadsworth, 1984.