Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning

Pegah Golchin *, Chengbo Zhou *, Pratyush Agnihotri *,
Mehrdad Hajizadeh †, Ralf Kundel *, Ralf Steinmetz *
*Multimedia Communications Lab (KOM), Technical University of Darmstadt, Germany
†Communication Networks, Technical University of Chemnitz, Germany
Contact: pegah.golchin@kom.tu-darmstadt.de

***Abstract*—The centralized control plane in Software-Defined Networking (SDN) offers significant advancements in network management capabilities. However, SDN is also susceptible to cybersecurity risks and vulnerabilities. Deploying the Machine Learning (ML) approach in an Intrusion Detection System (IDS) can facilitate early detection of potential vulnerabilities. However, deploying an ML-based IDS solely in either the SDN control plane or the data plane has its benefits and drawbacks. For instance, a high-capacity ML model deployed in the control plane can enhance the detection performance but may increase network latency and the risk of overwhelming the control plane. In contrast, lightweight ML models deployed in the data plane could accelerate intrusion detection with lower detection performance. However, a functional IDS should provide a good detection performance at a line rate. To accomplish these objectives, we introduce a novel method called Collaborative ML-based IDS (CML-IDS), which involves deploying ML models in both the control and data planes to detect network attacks collaboratively. To facilitate this collaboration, we assess the confidence of the classification model, which is flexibly deployed within the programmable data plane. Our evaluation results demonstrate that the CML-IDS enhances the average intrusion detection performance to 93.46% and reduces the misclassification rate by 54.66% when compared to an IDS that solely relies on the ML model deployed in the data plane. Furthermore, CML-IDS effectively reduces network latency caused by forwarding flows to the control plane.**

***Index Terms*—Intrusion Detection System, Machine Learning Models, P4 Switches, Software Defined Networking**

I. INTRODUCTION

Software-Defined Networking (SDN) provides automated and flexible network control compared to the traditional legacy network by decoupling the functionalities of the data plane and the control plane. The centralized control plane offers a comprehensive network view, resulting, e.g., in improved routing decisions and flexibility in general. This comprehensive network view is achieved by providing flow rules to the data plane switches, enabling them to process packets based on these rules.

However, the centralized nature of the control plane in SDN renders it vulnerable to various cyber-attacks, including Denial of Service (DoS) or Distributed DoS (DDoS) attacks [1]. These malicious activities aim to disrupt the desired network behavior. An attack can be directed toward either the data plane or the control plane. In the case of the data plane, the switch can be inundated with numerous flows, resulting in the depletion of the limited flow table memory within the data plane switch. Consequently, the switch becomes incapable of receiving new flows. Additionally, attackers can flood the switch with a multitude of new packet flows in a short time frame, leading to the exhaustion of the switch's buffer. In this scenario, the switch forwards all buffered packets to the control plane, potentially overwhelming it and impeding the control plane's bandwidth [2]. Network intrusions can be detected by analyzing the traffic within each flow via an Intrusion Detection System (IDS). In these approaches, a flow represents all packets sharing the same source and destination IP addresses, L4-ports, and protocols [3]. The two primary approaches for implementing a network IDS are signature-based and anomaly-based [4]. Leveraging signature-based intrusion detection involves employing pattern-matching techniques with a database of known attacks to detect attack flows. On the other hand, anomaly-based intrusion detection involves leveraging statistical methods to classify the malicious patterns of attack and benign (non-attack) flows, allowing the detection of zero-day attacks in comparison to the signature-based approach [5].

ML models are effective for anomaly-based IDS and can be deployed in the SDN control plane with sufficient computational resources [6]. Consequently, in order to classify each flow using an ML-based IDS deployed in the control plane, it is necessary to send flow features from the data plane to the control plane, where the IDS resides. The control plane can then determine the appropriate network policy (forwarding/mitigation rules) based on each flow's classification result (attack/benign). These policies will be installed in the data plane. However, sending flows to the control plane and populating network policies increase network latency and may cause the control plane to become overwhelmed when dealing with a large number of flows at high speed. Therefore, alternative approaches need to be explored to enhance the efficiency and effectiveness of ML-based IDS in identifying attack traffic flows while reducing network latency and resource utilization.

Recently, SDN concepts have been extended by programmable switches [7] to program the data plane behavior using domain-specific programming languages such as P4 [8]. Developing an IDS in the data plane using programmable switches can reduce the latency caused by sending flows to the control plane and decrease the attack detection time.

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

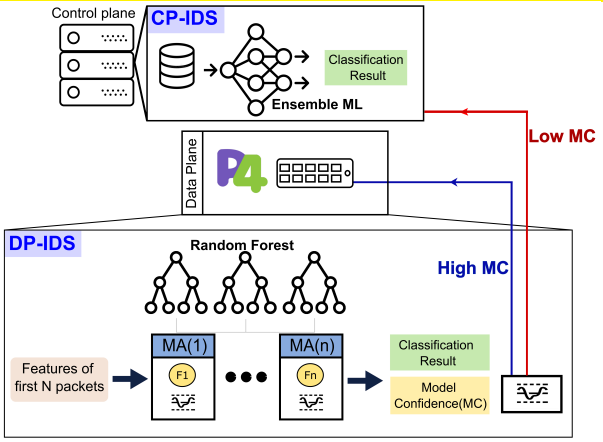

Fig. 1: CML-IDS architecture. The DP-IDS and CP-IDS collaboration is based on the DP-IDS Model Confidence (MC).

Currently, training an ML-based IDS in the programmable data plane is challenging due to the resource and computation limitations of the switch. Instead, a previously trained ML model can be integrated into the Match-Action (MA) pipeline of P4 switches for traffic classification. To this aim, the initial phase involves extracting traffic features that are compatible with the switch limitations. Furthermore, it is required to utilize a lightweight ML model (with a reduced number of learnable parameters) aligned to the switch structure. These two constraints (utilizing only compatible flow features and leveraging a lightweight ML model) can decrease the detection performance and increase the possibility of misclassification, potentially affecting network QoS and security requirements. However, in the context of designing an effective IDS, it is crucial to improve intrusion detection performance while minimizing detection time and forwarding latency.

To achieve these goals, we propose CML-IDS, a collaborative ML-based IDS approach for SDN. Through the collaborations of two proposed ML-based IDSs in both planes, we can effectively leverage advantages of each, thereby enhancing the overall detection performance. The general architecture of CML-IDS is illustrated in Figure 1. Within the CML-IDS framework, we investigate the flow features compatible with the programmable switch structure. Furthermore, the ML model deployed in the data plane (DP-IDS) is effectively embedded within the programmable switch, offering more flexibility for updates and changes.This flexibility ensures that the DP-IDS model can be readily modified and improved. Ultimately, considering the DP-IDS Model Confidence (MC), further collaboration with the IDS deployed in the control plane (CP-IDS) will be pursued to improve the detection performance.

The paper is structured as follows: Background information on the programmable switches is given in Section II. Section III explains related works. The CML-IDS architecture is presented in Section IV. In Section V, evaluation results are presented and discussed. Finally, Section VI provides a short summary.

II. BACKGROUND

This section provides a brief overview of the key concepts related to the groundwork for the design and implementation of the proposed CML-IDS.

A. Programmable Switches

Programmable switches enable network flexibility and in-network control. P4 [8] is a programming language by which the data plane can be tailored to customized packet processing. The P4 language provides packet processing abstractions [9]. The packet processing in a programmable switch begins with a parser that extracts information from the received packets directly behind the ingress port. Match-Action (MA) tables apply actions to the received packets that match to the stored keys in the table. The control plane can configure and modify the entries of MA tables in the programmable switch. Registers are utilized for storing metadata over an extended period, allowing for the computation of stateful features. These registers are readable and writable from both the data and control planes. However, the limited resource capacity of a switch means that a limited number of registers can be used in a programmable switch. In addition, the P4 language lacks support for iterative constructs and certain mathematical operations, such as division, logarithm, and exponentiation. That limits the available ML models which can be implemented within the programmable switch leveraging the P4 language.

B. Mapping Tree-based Models into MA Pipelines

A tree-based ML model classifies an input (e.g., features of an arriving flow) by following a path from the root node through intermediate nodes until it reaches the appropriate leaf node, representing a decision (classify the arrival flow as an attack or benign). Different selections of nodes result in different classification paths. Two procedures are required to construct the classification path. At the root and intermediate nodes, the value of a certain feature is compared against a specific threshold. The comparison result decides the next node to be selected. At the leaf nodes, the prediction of the input is performed, and the classification process is completed.

The MA pipeline in the programmable switch comprises multiple MA tables that are applied sequentially. Each table contains numerous table entries consisting of various key values to be compared and a corresponding action to be carried out. The sequential execution of MA tables is similar to the classification process of a tree-based ML model, where each level of the tree is reached exactly once in sequence and only a single node at each level is selected to construct the classification path. Therefore, each level of the tree can be mapped to an MA table, whereas the table entries can represent the nodes at each level. The two procedures can be implemented by the actions carried out in the table entries, while the keys are responsible for the selection of nodes.

III. RELATED WORK

This section presents an overview of ML-based IDS deployed in either the SDN control plane or data plane, highlighting the distinctions from the proposed CML-IDS.

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

A. ML-based IDS in the Control Plane

In [10] and [11], different ensemble ML-based IDSs deployed in the SDN control plane are presented. It is stated that utilizing multiple ML models in an IDS can improve the detection performance. However, their proposed flow-based IDS poses a limitation in minimizing the detection time of the attack flows since the IDS relies on predicting completed flows, leaving the victim host or controller susceptible to attacks prior to detection. The authors in [12] introduced an ensemble feature selection method to boost the detection speed and to reduce the required computational resources for an ML-based IDS in the control plane by reducing the feature space dimensions. Although their method improves detection performance and reduces detection time, it does not operate at a line rate, and it could not effectively reduce packet forwarding latency as it necessitates sending flows to the control plane. The approach presented in [13] addresses GPU-accelerated network function acceleration, e.g., for IDS systems in the control plane; however the achievable performance is still not comparable to programmable data plane switches.

B. ML-based IDS in the Data Plane

To address the drawbacks of an ML-based IDS deployed in the control plane, the idea of embedding a ML model directly into programmable switches was proposed.

Embedding an ML model in a P4 switch is challenging because the P4 language does not support floating values and certain mathematical operations, and the switch usually provides limited memory resources. The authors of [14] investigated the deployment of an ML-based traffic classification scheme within the programmable data plane. They implemented different ML models within a P4 switch, such as Decision Tree, K-means, and Support Vector Machine. However, their proposed system, IIsy, has limitations w.r.t. accuracy and feature extraction due to the utilization of look-up tables for estimating computational results. In [15], RF and DT model structure alignment to a P4 switch was considered. The authors also compared packet-based and flow-based approaches in terms of attack detection performance, observing a higher detection performance for the flow-based approach. The authors in [16] proposed a flow-based IDS, called SwitchTree, in which an RF algorithm was embedded into the programmable switch by hard-coding. They trained the RF model using pre-recorded flow data with twelve stateless and stateful features. A total of 20 registers are used in SwitchTree to store the features and metadata. However, using too many registers can slow down detection because each one needs to be updated with each new received packet. Moreover, using many registers is incompatible with the switch hardware due to the memory size limitation. In [17], a flow-based IDS named pForest was proposed, in which multiple RFs are embedded into the P4 switch to perform intrusion detection for the flows with different number of packets. The predicted outcome of each RF estimator is only accepted if the accuracy exceeds a predefined minimum threshold. The final prediction result of a flow is determined by the RF estimator that reaches the threshold first. To overcome the limited memory resources, pForest concatenates all flow features into a bit string and stores it in a single register, significantly reducing the number of used registers.

IV. CML-IDS SYSTEM DESIGN

This section presents a comprehensive description of the system design of CML-IDS along with relevant implementation details of each module. The source code of CML-IDS is available on GitHub for public access[1].

A. Flow & Sub-flow Scope

In CML-IDS, a traffic flow is identified by employing the CRC32 hash function, which takes into account the 5-tuple comprising the source and destination IP addresses, source and destination port numbers, and protocol type. To prevent potential delays in attack detection caused by waiting for the entire flow to complete, we introduce a sub-flow approach using the initial N packets only for the attack detection decision. The determination of N is based on its ability to extract informative flow features. Consequently, in the evaluation section (V-A), the performance of ML models is evaluated for different N values. Based on our investigation and the findings reported in [17], utilizing the first 8 packets yields optimal results in terms of informativeness and classification performance.

To extract the statistical features of a network flow, we employ NFStream, a Python framework renowned for its ability to extract post-mortem and early statistical flow features [18]. In this study, we extend NFStream through the use of NFPlugin to enable the extraction of sub-flow information. The extracted statistical features are categorized into three directions: source to destination (*src2dst*), destination to source (*dst2src*), and *bidirectional* (involving packets flowing in both *src2dst* and *dst2src* directions).

B. Sub-flow Features in the DP-IDS

In the following, we will elucidate the process by which we extract the compatible features to train the DP-IDS model.

1) ***Investigating Incompatible Sub-Flow Features****:* This section provides the reasoning behind defining certain features as incompatible ones. These features are removed for training the ML models.

Architecture-based Features: To ensure the general application of CML-IDS regarding any network traffic traversing a switch, it is necessary to eliminate network architecture-based features (e.g., IP addresses, port numbers, and protocols).

Time-related Features: These features can be computed within DP-IDS by retrieving the values of *ingress_global_timestamp* and *egress_global_timestamp* of the P4 language, that reflect the ingress and egress timestamps when a packet appears and leaves the switch. These timestamps are measured in *microseconds* within the programmable switch (BMv2), while NFStream measures time-related features in *milliseconds*. To ensure consistency of time units, the obtained results by NFStream are multiplied

[1]https://github.com/golchinpg/CML-IDS

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

TABLE I: Comparison of Packet Inter Arrival Time (PIAT) measurements between BMv2 and NFStream.

Time-Related Features	**Measured in BMv2** (μs)	**Measured by NFStream** (μs)
Min. PIAT (*dst2src*)	3410.7	2000
Min. PIAT (*bidirectional*)	255.6	0
PIAT (between 1st and 2nd packet)	729.4	0

by 1000. Nonetheless, a significant bias persists due to the lower precision of NFStream's time measurement. Specifically, NFStream assigns a value of 0 to time-related features that are less than 1 *millisecond*. Table I illustrates the difference between the measured values of time-related features in BMv2 and NFStream. The values measured in BMv2 represent the average of ten individual measurements. Moreover, it shows that time-related features obtained by NFStream are imprecise compared to BMv2. Consequently, we eliminated the time-related features from the training dataset.

Complexity-intensive Features: The limitations of the P4 language pose a challenge for calculating complex features requiring loops or divisions (e.g., calculating the packet size standard deviation) within DP-IDS. This limitation makes it currently infeasible to compute most of these features within the programmable switch. To calculate the mean value of a feature, [17] proposed to estimate it utilizing the Exponential Weighted Moving Average (EWMA) method instead of calculating the exact mean value. For example, the *src2dst_mean_ps* (*ps* refers to the packet size) feature estimation is performed according to equation 1, with a smoothing factor of 0.5.

$$\bar{PS}^{t}_{src2dst} = \frac{\bar{PS}^{t-1}_{src2dst} + Payload^{t}_{src2dst}}{2} \tag{1}$$

where $\bar{PS}^{t}_{src2dst}$ is the current mean packet size and $\bar{PS}^{t-1}_{src2dst}$ is the previous mean packet size. As the division operation in equation 1 is not supported in the P4 language, the division by 2 is achieved by right-shifting the binary value by one position. However, the accuracy of the estimated mean value using EWMA may be low if the payloads of the packets within a sub-flow vary significantly. Since in our implementation, the maximum number of packets per sub-flow is constant, i.e., 8, the value of the feature *bidirectional_mean_ps* can be accurately calculated by right-shifting the binary value of the feature *bidirectional_bytes* (the sum of packet sizes of the extracted packets within a sub-flow) by three positions. Consequently, all features related to the mean value, except *bidirectional_mean_ps*, are removed from the training dataset. Additionally, the computation of features related to the standard deviation is infeasible within the current P4 switch pipeline architecture.

After removing incompatible features, a total of 59 features will remain. To reduce the feature dimensionality, the 20 most relevant features are selected leveraging an ensemble feature importance algorithm, including Mutual Information, Impurity score, RF feature selection, and Permutation score algorithms [19]. This decreases the required computational resources and the risk of overfitting, which arises when the model is overtrained on a particular dataset, leading to inadequate performance when evaluated on the unseen data [20]. The list of 20 selected features is available in the CML-IDS GitHub repository.

*2) **Flow Storage Mechanism**:* To calculate stateful features (depending on the previous state) in BMv2, we optimize storage efficiency by using a single register as a buffer to store all sub-flow entries. This strategy is inspired by pForest [17]. The compatible sub-flow features and metadata are serialized into a bit string using the bit concatenation operation supported by the P4 language and stored in the buffer indexed by the sub-flow ID. A sub-flow entry is expired after a certain timeout, allowing the buffer to store the newly extracted sub-flows. The sub-flow features are updated in the following steps: 1) The bit string is retrieved from the buffer based on the sub-flow ID computed from the incoming packet. This bit string is then converted to the *struct* data structure, which enables to read each sub-flow feature value. 2) These values are updated by the extracted header of the incoming packet. 3) The updated sub-flow features are converted to the bit string and written into the buffer based on the sub-flow ID.

*3) **Incoming Packet Analysis & Packet Direction Recognition**:* Updating the sub-flow features requires knowing the direction of the incoming packet mentioned in section IV-A. In this regard, the CML-IDS utilizes an efficient flow identification procedure within the programmable switch to identify the packet direction category, initialize new sub-flows, and detect potential hash collisions which are caused by using the CRC32 hash function to compute the sub-flow IDs. Upon receiving a packet p, a sub-flow entry F_i is retrieved from the buffer based on the sub-flow ID computed from p. p is identified as a *src2dst* packet of F_i when all the values of the 5-tuple in p match the corresponding values in F_i. Otherwise, the other sub-flow entry F_j is read from the buffer based on the sub-flow ID computed in the swapped order of the IP addresses and port numbers. The details of this procedure are presented in the Algorithm 1.

C. ML Model Deployed in the DP-IDS

*1) **DP-IDS Model**:* As Section II-B discusses the similarities between the tree-base ML model and the programmable switch structure, a lightweight RF model with three individual trees with a maximum depth of five is deployed in the DP-IDS. The preprocessed dataset is split into training, validation, and test sets, with proportions of 70%, 10%, and 20%, respectively. The validation set is leveraged to optimize hyperparameters of ML models utilizing the GridsearchCV function provided by the scikit-learn library in Python [21]. After the training phase of the DP-IDS, the subsequent task involves extracting the RF model inference, which entails converting the model into rules that align with the P4 program. This extracted RF model is then deployed within the DP-IDS, ensuring compatibility with the P4 switch's format. This process is automated and executed within the control plane, thereby granting the CML-IDS the flexibility to integrate new RF models with distinct

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

Algorithm 1: Incoming packet processing

Input: An incoming packet p

$(src2dst\_no\_match, src2dst\_empty) \leftarrow False$;
$id_i \leftarrow$ CRC32$(p_{s_ip}, p_{d_ip}, p_{s_port}, p_{d_port}, p_{proto})$;
$F_i \leftarrow$ Buffer.read(id_i);
if F_i *is not empty* **then**
  if $p.5\_tuple = F_i.5\_tuple$ **then**
    p is the *src2dst* packet of F_i;
  else
    $src2dst\_no\_match \leftarrow True$;
  end if
else
  $src2dst\_empty \leftarrow True$;
end if
if $src2dst\_no\_match$ *or* $src2dst\_empty$ *is* $True$ **then**
  $id_j \leftarrow$ CRC32$(p_{d_ip}, p_{s_ip}, p_{d_port}, p_{s_port}, p_{proto})$;
  $F_j \leftarrow$ Buffer.read(id_j);
  if F_j *is not empty* **then**
    if $p.swapped\_5\_tuple = F_j.5\_tuple$ **then**
      p is the *dst2src* packet of F_j;
    else
      if $src2dst\_no\_match$ *is* $True$ **then**
        Hash collision happens;
      else if $src2dst\_empty$ *is* $True$ **then**
        p results in a new sub-flow F_i;
      end if
    end if
  else
    if $src2dst\_no\_match$ *is* $True$ **then**
      Hash collision happens;
    else if $src2dst\_empty$ *is* $True$ **then**
      p results in a new sub-flow F_i;
    end if
  end if
end if

hyperparameters into the switch. Consequently, it is no longer necessary to hard code a single trained ML model into the programmable switch. A detailed description of the procedure for embedding the RF model in the programmable switch is provided in Section IV-D.

*2) **Interpretability of the Model's Inference**:* To better comprehend the implementation of the RF model within a P4 switch in the CML-IDS architecture, we provide an example of a two-level Decision Tree (DT) model (as RF consists of multiple DTs) in Figure 2. An ID number uniquely identifies each node of a DT. In each node, the sub-flow feature is denoted as f_i, the threshold is represented by t_i, and the Gini impurity value, displayed as g_i, represents a measure of the purity for each node in the DT model. The CML-IDS initiates the classification process in the programmable switch immediately after the completion of feature extraction from a sub-flow. In Figure 2, the classification path (1-3-6) is highlighted to explain the decision making process of a DT

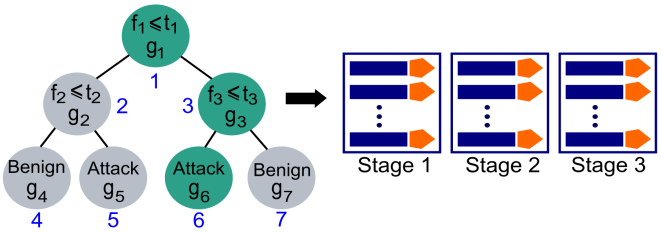

Fig. 2: Example of embedding a DT into a P4 switch.

model. Depending on the comparison result between the feature value and the threshold value at each node, the decision-making process proceeds to the next node or terminates at a leaf node, resulting in the classification of the sub-flow as an attack or benign. The same decision rules are embedded into the MA pipeline of the programmable switch to classify each incoming sub-flow.

D. Embedding DP-IDS in a Programmable Switch

This pipeline comprises three key modules which are explained in the following.

*1) **P4 Program Generator**:* The proposed CML-IDS architecture utilizes this module to integrate the trained RF model into the programmable switch. The module receives the feature set chosen in Section IV-B1 and the inference of the RF model described in Section IV-C1 as inputs. Utilizing them, the compatible P4 code and MA table entries are automatically generated. The generated P4 code contains four blocks, including feature storage and update, MA table entries, feature comparison logic, and classification logic of the RF model.

To control the detection path, the MA table uses the previous node ID and the previous feature comparison result as keys. The action taken depends on the current node type, where root and intermediate nodes compare feature values with thresholds to select the next node, while leaf nodes use their prediction to classify a sub-flow. Furthermore, the model confidence (MC) is computed by averaging the Gini impurity of three DTs in the embedded RF.

*2) **Data Plane Control Module**:* This module is designed to facilitate interaction with the data plane. In our implementation, the P4Runtime API [22] serves as the interface for establishing a connection between the switch and the controller. The P4Runtime API is utilized for controlling the behavior of the BMv2 switch during runtime. The controlling process includes installing the P4 program in the BMv2 switch, transmitting the packet containing the sub-flow information from the switch to the controller, sending the packet with the final predicted label of the sub-flow from the controller to the switch, and populating the MA table entries into the switch for representing the RF model.

*3) **Programmable Switch**:* The programmable switch plays a crucial role in the data plane, running a P4 program received from the control plane. Upon receiving packets, their header fields are extracted by the P4 parser, and the corresponding sub-flows are retrieved from the buffer based on the computed 5-tuple hash value that represents the sub-flow ID. To

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

TABLE II: CP-IDS hyperparameters.

Model	Hyperparameter Values
XGB	*max_depth:* 25, *tree_method:* 'approx', *scale_pos_weight:* 40
RF	*n_estimators:* 500, *min_samples_leaf:* 5, *max_depth:* 10
MLP	*number of hidden layers:* 14, *batch_size:*2048, *epochs:*200

avoid hash collisions and save memory, sub-flow entries are removed after a specified timeout. If the sub-flow entry is valid and the updated packet number reaches the length of sub-flow, classification is performed by the DP-IDS, and only the predicted labels with high MC are accepted. The data plane control module monitors the performance of DP-IDS by reading counter values related to detection performance.

E. Model Confidence Threshold (MC_{thr})

The CML-IDS aims to enhance the detection performance and keep network latency low by utilizing a pre-defined MC_{thr} to control whether to send sub-flows to the control plane. This threshold is compared against the MC value, represented by the Gini impurity of the leaf node (last stage nodes) in the output of the RF model in the DP-IDS. More specifically, the prediction made by DP-IDS has a higher MC when the Gini impurity is lower. Lower Gini impurity indicates that the sub-flow is more confidently classified by the DP-IDS. After extracting MC value, if all three DT models (DT_i, DT_j, DT_l) classify a sub-flow F into the same class (benign/attack), the average Gini impurity ($\bar{G}i, j, l(F)$) is compared to MC_{thr} to determine whether to forward F to the CP-IDS ($\bar{G}i, j, l(F) \geq MC_{thr}$) or accept the detection result ($\bar{G}i, j, l(F) < MC_{thr}$). If only two DT models (e.g., DT_i, DT_j) classify sub-flow F into the same class, the mean value of their Gini impurities ($\bar{G}i, j(F)$) is compared to the Gini impurity value of the third DT model ($G_l(F)$) and MC_{thr}. If one of these two comparisons meets the condition ($\bar{G}i, j(F) \geq G_l(F)$ or $\bar{G}i, j(F) \geq MC_{thr}$), the sub-flow F is forwarded to the CP-IDS for the further detection process.

F. ML-based IDS in the Control Plane (CP-IDS)

Suppose the MC value fails to meet the specified threshold. In that case, the sub-flow features are subsequently forwarded to the CP-IDS for classification utilizing an ensemble ML classifier which is more complex than the DP-IDS.

1) ***Sub-flow Features in CP-IDS****:* All the 59 features extracted within the programmable switch (without feature selection) are encapsulated into a packet and delivered to the CP-IDS. These features are used to train the ensemble classifier employed in the CP-IDS. A list of these features is available in the CML-IDS GitHub repository.

2) ***Ensemble Classifier in CP-IDS****:* In CP-IDS, an ensemble classifier composed of multi-layer perceptron (MLP), RF, and XGBoost (XGB) classifiers is employed. Ensembling these models offers a method to combine multiple instances of ML models using different learning algorithms, which can complement each other and improve overall system performance with greater accuracy [23].

The preprocessed dataset is divided into training, validation, and test sets, with proportions of 70%, 10%, and 20%, respectively. Through GridsearchCV, we optimize the hyperparameters to ensure the optimal performance on the validation set. Table II provides the hyperparameters for each model of CP-IDS. To optimize the overall performance of the CP-IDS, the XGB model is specifically tailored to minimize the number of false negatives; the RF model focuses on minimizing the number of false positives, while the MLP is designed to strike a balance between false positives and false negatives.

V. EVALUATION AND DISCUSSION

This section aims to present the evaluation results of CML-IDS on various types of attacks, such as Brute Force, DoS/DDoS, and Botnet attack flows, along with benign flows, using the CIC-IDS2017 dataset [24]. To evaluate the framework in real-time, we transmit the previously captured network traffic of the CIC-IDS2017 through the system, using the open-source tool *tcpreplay*. This tool is capable of reading network packets from a Packet Capture (PCAP) file and replaying them onto the desire network interface. To train the DP- and CP-IDS as well as perform the ensemble feature selection, we utilize an Ubuntu server equipped with 128GB RAM and 4 CPUs (Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz). The implementation is in Python, using the Scikit-learn, Pandas, XGBoost, and TensorFlow packages. To evaluate the detection performance of the proposed CML-IDS, we utilized the macro-average of the F1 Score metric. The F1 Score is a suitable evaluation metric as it considers both false positives and false negatives, leading to a more accurate assessment of the detection performance. In this study, the positive class indicates the attack, and the negative class represents the benign traffic. Given that the majority of network traffic typically consists of benign flows, we opted to calculate the macro-average of the F1 Score to ensure equal treatment of each class, regardless of its frequency or potential imbalance in the dataset.

A. Analysis of the Optimized Number of Packets in a Sub-flow

To ascertain the number of initial packets required to define a sub-flow, we examine the effectiveness of different numbers of initial packets in providing informative data to better differentiate between attacks and benign flows. As shown in Figure 3, the average misclassification of sub-flows decreased as the number of packets increased from 4 to 8. Therefore, the overall detection performance is increased. Thus, 8 packets in a sub-flow provide more informative data. However, limited improvement is observed when increasing the number of packets from 8 to 15 and 30. To minimize the delay in initiating the detection process and avoid waiting for additional packets, we consider a sub-flow size of 8 packets.

B. Reducing Potential Hash Collisions

CML-IDS utilizes one single register as the buffer to store all sub-flow entries indexed by their IDs computed using the CRC32 hash function. A hash collision could occur when a new extracted sub-flow has the same ID as a sub-flow that exists in the buffer. Hash collisions can impact both the feature update and inference procedures. When a hash collision takes

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

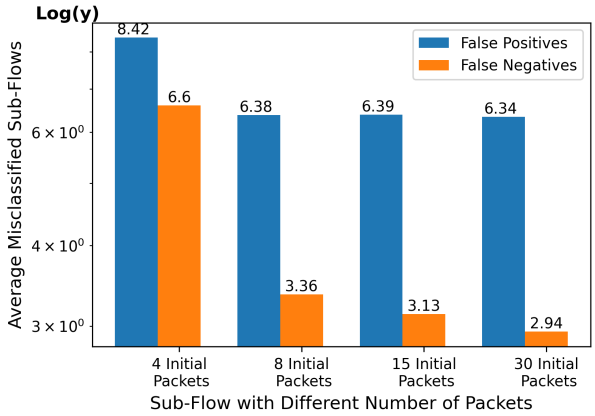

Fig. 3: Comparison of the number of FP and FN across varying numbers of initial packets.

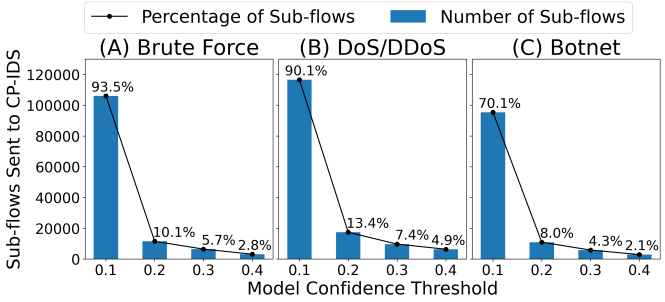

Fig. 4: The number of sub-flows forwarded to the CP-IDS utilizing different MC_{thr} values. The percentage values represent the proportion of the sub-flows forwarded to the CP-IDS.

place during the feature updating process for a flow (consisting of less than 8 packets), it results in an inaccurate feature update. Similarly, if a hash collision occurs for a sub-flow that has already been classified, the subsequent packets belonging to the collided sub-flow are processed based on the predicted class of the existing sub-flow. This can lead to potentially incorrect packet inference. Hence, a greater number of hash collisions indicates a higher level of instability in the system.

To reduce the hash collisions and store the sub-flows as much as possible, a flow expiration mechanism is applied. A sub-flow entry is removed from the buffer after 30 minutes. This choice of a 30-minute expiration timeout was made to align with the NFStream timeout for generating flows. As discussed in Section IV-A, NFStream converted network traffic into sub-flow statistical features. We conducted three independent experiments to measure the occurrence of hash collisions, utilizing the flow expiration mechanism. The results revealed that the proposed flow expiration mechanism effectively reduced hash collisions for each attack dataset. In particular, the observed percentages of flows impacted by hash collisions were 0.13%, 0.13%, and 0.30% for BruteForce, DoS/DDoS, and Botnet attacks, respectively, indicating a minimal impact of hash collisions on the overall performance of the system. Additionally, 20.08%, 23.68%, and 15.83% of the flows for BruteForce, DoS/DDoS, and Botnet attacks, respectively, expired after the expiration timeout.

C. Analysing the Impact of MC_{thr} on Detection Performance

In this study, the primary objective of CML-IDS is to improve detection performance while minimizing potential increases in network latency. As explained in Section IV-E, a predefined MC_{thr} is required to decide whether to forward the sub-flow to the CP-IDS or not. This section aims to evaluate the impact of different MC_{thr} values on the overall detection performance achieved through the collaboration of DP- and CP-IDS compared to solely relying on the DP-IDS (baseline) detection. To highlight the distinctions clearly, we also measure the misclassification rate of CML-IDS using the formula $(FP + FN)/(FP + FN + TP + TN)$. Figure 4 displays the average number of sub-flows and their percentages that are forwarded to the CP-IDS for the BruteForce, DoS/DDoS, and Botnet attacks, with the experiment being repeated three times. As expected, when MC_{thr} is set to a very small value, the majority of sub-flows are forwarded to the CP-IDS since these sub-flows are unlikely to have higher MC (i.e., lower Gini impurity values). As MC_{thr} increases, the number of forwarded sub-flows decreases, reducing the overall network latency. According to Figure 4, for the BruteForce, DoS/DDoS, and Botnet datasets, the percentages of 93.5%, 90.1%, and 70.1% sub-flows are forwarded to the CP-IDS when the value of MC_{thr} is set to 0.1. These are very high percentages compared to the other MC_{thr} values. Therefore, selecting a very low MC_{thr} does not align with our objectives. Table III presents the impact of different MC_{thr} values on the detection performance and the misclassification rate. According to the results, the macro-average F1 Score has improved when utilizing CML-IDS (for all MC_{thr} values) compared to the baseline approach, which only utilizes DP-IDS. Moreover, the results of the misclassification rate demonstrate that CML-IDS effectively reduces the percentage of misclassifications. The maximum difference is for $MC_{thr} = 0.3$, displayed in bold in Table III. At this threshold, the detection performances for BruteForce, DoS/DDoS, and Botnet attacks are improved by 12.2%, 5.8%, 3.6%, and the misclassification rates are reduced by 84.0% $((4.4 - 0.7)/4.4)$, 29.2%, 54.3%, compared to the baseline.

D. Selecting MC_{thr} Value

In order to achieve a balance between high detection performance and low network latency, various metrics are explored to select an appropriate MC_{thr} value. Despite a high detection performance at MC_{thr} values of 0.1, 0.2, and 0.3 (as shown

TABLE III: Relative detection performance (DP) and misclassification rate (MR) of different attacks for different MC_{thr}.

$\mathbf{MC_{thr}}$	**BruteForce**		**DoS/DDoS**		**BotNet**	
	DP(%)	MR(%)	DP(%)	MR(%)	DP(%)	MR(%)
0.1	96.8	0.7	83.0	7.1	95.5	3.7
0.2	96.7	0.7	83.7	7.2	96.6	2.9
0.3	***96.7***	***0.7***	***87.0***	***6.3***	***96.7***	***2.6***
0.4	88.1	3.1	82.0	7.9	94.7	4.4
Baseline	***84.5***	***4.4***	***81.2***	***8.9***	***93.1***	***5.7***

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

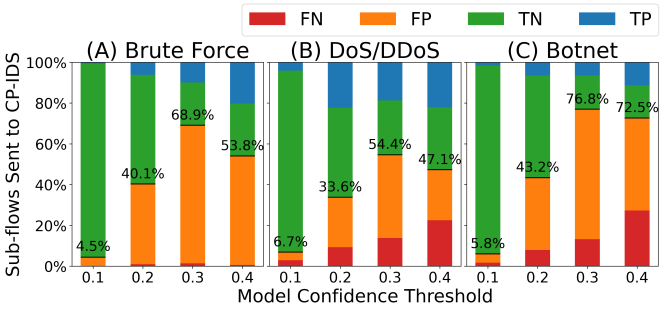

Fig. 5: Percentages of FP, FN, TP, and TN in the DP-IDS which are forwarded to the CP-IDS for different MC_{thr}.

in Table III), Figure 4 indicates that a higher number of sub-flows are forwarded to the CP-IDS for $MC_{thr} = 0.1$ and $MC_{thr} = 0.2$ compared to $MC_{thr} = 0.3$, leading to increased network latency. Thus, selecting $MC_{thr} = 0.3$ is more aligned with the primary goals of CML-IDS.

Furthermore, we conducted a detailed analysis of the number of correctly and incorrectly detected sub-flows in the DP-IDS that are forwarded to the CP-IDS for different MC_{thr} values. Figure 5 provides insight into the percentage of incorrectly detected sub-flows (FP or FN) by the DP-IDS (which are different from FP and FN values in Section V-C) that was subsequently forwarded to the CP-IDS for different MC_{thr} values. The results presented in Figure 5 demonstrate that for all attacks, the percentage of FP and FN sub-flows that are sent to the controller when $MC_{thr} = 0.3$ is higher compared to when $MC_{thr} = 0.2$, with an increase of 28.8%, 20.8%, and 33.6% observed for the BruteForce, DoS/DDoS, and Botnet attacks, respectively. In summary, Table IV presents all the relevant metrics and their values that led to the selection of $MC_{thr} = 0.3$. The table comprises individual cells, each containing three values representing the corresponding metrics for BruteForce, DoS/DDoS, and Botnet attacks. The $MC_{thr} = 0.3$ values are highlighted in bold to ease comparison. The findings indicate that at $MC_{thr} = 0.4$, the percentage of forwarded sub-flows to the controller is lower than at $MC_{thr} = 0.3$. However, at $MC_{thr} = 0.3$, there is a higher number of incorrectly detected sub-flows (FP and FN) forwarded to the CP-IDS compared to $MC_{thr} = 0.4$. As a consequence, the overall detection performance is improved at $MC_{thr} = 0.3$ (Table III). Therefore, $MC_{thr} = 0.3$ is considered an appropriate choice for the CML-IDS to balance achieving high detection performance and maintaining low network latency.

TABLE IV: Comparison of important metrics to select the best MC_{thr}. Each cell represents the corresponding value for the BruteForce, DoS/DDoS, and Botnet attacks, respectively.

$\mathbf{MC_{thr}}$	Forwarded Sub-flows (%)	Macro-Average F1 Score (%)	Misclassified Forwarded Flows (%)
0.1	93.5, 90.1, 70.1	96.8, 83, 95.5	4.5, 6.7, 5.8
0.2	10.1, 13.4, 8	96.7, 83.7, 96.6	40.1, 33.6, 43.2
0.3	***5.7, 7.4, 4.3***	***96.7, 87.0, 96.7***	***68.9, 54.4, 76.8***
0.4	2.8, 4.9, 2.1	88.1, 82.0, 94.7	53.8, 47.1, 72.5

TABLE V: Comparison of detection time between the DP-IDS and the CML-IDS for different MC_{thr} values.

$\mathbf{MC_{thr}}$	Detection Time in the DP-IDS (second)	Detection Time in the CP-IDS (second)
0.1	0.067 (± 0.000)	137.066 (± 61.417)
0.2	0.075 (± 0.023)	0.546 (± 0.050)
0.3	0.057 (± 0.001)	0.361(± 0.040)
0.4	0.057(± 0.001)	0.354 (± 0.044)

E. Detection Time

In network security, the timely detection of malicious activities is crucial to mitigate potential damage and ensure the integrity of the network. In this section, we investigate the detection time for different MC_{thr} values. Moreover, we compare the detection time of the DP-IDS and the CP-IDS. Table V shows the impact of forwarding sub-flows to the CP-IDS on the overall detection time. Regardingly, the detection time increases when $MC_{thr} = 0.1$ is used, as higher percentages (93.5%, 90.1%, and 70.1% for different attacks illustrated in Figure 4) of the sub-flows are forwarded to the CP-IDS. Therefore, it takes more time to classify sub-flows within the CP-IDS for $MC_{thr} = 0.1$. The significant detection delay observed in the CP-IDS for $MC_{thr} = 0.1$ emphasizes the limitations of relying solely on ML models deployed in the control plane for effective intrusion detection. The detection time for the other MC_{thr} values is less than 1 second, indicating a rapid detection time.

F. Comparison between CML-IDS and SwitchTree

In this section, we present a comparison between the CML-IDS and the SwitchTree [16] (in Table VI). We chose to compare with SwitchTree primarily because their open-source code is readily available, and their approach shares the closest similarity to our DP-IDS (not the entire CML-IDS). To enable a fair comparison, it was necessary to evaluate their model using the dataset utilized for this work and convert the network traffic to the feature set using our proposed preprocessing pipeline. Table VI presents a comparison of these two approaches in terms of various metrics. The CML-IDS involves the collaboration of two ML models, DP- and CP-IDS, while SwitchTree uses only an RF model as a DP-IDS. Hence, CML-IDS utilizes a more complex approach in comparison to SwitchTree. However, the proposed DP-IDS in SwitchTree employs a more complex design of an RF model with a maximum depth of 11, which makes it challenging to embed it in a programmable switch, while CML-IDS utilizes a lightweight RF model with a maximum depth of 5. Moreover, regarding hardware resources, the SwitchTree approach requires 20 registers, whereas the CML-IDS employs an efficient register usage technique by

TABLE VI: Comparison between CML-IDS and SwitchTree.

Metric	CML-IDS	SwitchTree [16]
Deployed ML model	CP & DP	DP
DP-IDS complexity	Max. Depth: 5	Max. Depth: 11
Number of Registers	1	20
Macro-Average F1 Score	93.4%	87.7%

Pegah Golchin, Chengbo Zhou, Pratyush Agnihotri, Mehrdad Hajizadeh, Ralf Kundel, Ralf Steinmetz. *CML-IDS: Enhancing Intrusion Detection in SDN through Collaborative Machine Learning*. To appear in Proceedings of the 19th International Conference on Network and Service Management, 2023.

applying bit concatenation operation (as explained in Section IV-B2). Additionally, the average detection performance on all attack datasets demonstrates that the CML-IDS model (with $MC_{thr} = 0.3$) outperforms SwitchTree. Furthermore, in CML-IDS, the RF model is deployed in DP-IDS with the flexibility to facilitate model updates (explained in Section IV-C1). However, SwitchTree rigidly hardcoded the RF model in the software switch.

VI. CONCLUSION & FUTURE WORK

This paper introduces a novel ML-based IDS approach in SDN, leveraging collaboration between distinct ML models in the data (DP-IDS) and control plane (CP-IDS). The collaboration will occur by assessing the confidence of the DP-IDS model in classifying a sub-flow and comparing it to a predefined threshold. Accordingly, if the model confidence is low the sub-flow features will be forwarded to the control plane.The evaluation highlights the importance of choosing the correct predefined threshold (MC_{thr}) value. In this work, a value of 0.3 is chosen for MC_{thr} based on different criteria. The results indicate that utilizing CML-IDS leads to an average reduction of 54.66% in the misclassification rate compared to the baseline, which solely relies on DP-IDS. Moreover, CML-IDS effectively improves detection performance and reduces the latency caused by forwarding flows to the control plane. Furthermore, we compare CML-IDS with SwitchTree, a state-of-the-art DP-IDS, to highlight CML-IDS advantages, including improvement in detection performance and reducing resource usage by decreasing the number of registers and DP-IDS complexity.

In this study, the CML-IDS is deployed on the BMv2, a software-based P4 switch, considering hardware limitations. Our future work aims to expand this approach to a Tofino-based switch, considering the challenges of the Tofino platform and conducting a comparative analysis.

ACKNOWLEDGMENT

This work is funded by the Federal Ministry of Education and Research of Germany (BMBF) through Software Campus Grant 01IS17050 (ML-based NIDS), the CELTIC-NEXT Flagship Project AI-NET-PROTECT and in parts by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1053 MAKI.

REFERENCES

[1] K.-y. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "Sdnshield: Towards more comprehensive defense against ddos attacks on sdn control plane," in *2016 IEEE conference on communications and network security (CNS)*. IEEE, 2016, pp. 28–36.

[2] L. F. Eliyan and R. Di Pietro, "Dos and ddos attacks in software defined networks: A survey of existing solutions and research challenges," *Future Generation Computer Systems*, vol. 122, pp. 149–171, 2021.

[3] R. V. Steiner and E. Lupu, "Towards more practical software-based attestation," *Computer networks*, vol. 149, pp. 43–55, 2019.

[4] S. Einy, C. Oz, and Y. D. Navaei, "The anomaly-and signature-based ids for network security using hybrid inference systems," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–10, 2021.

[5] M. Hajizadeh, S. Barua, and P. Golchin, "Fsa-ids: A flow-based self-active intrusion detection system," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023, pp. 1–9.

[6] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE access*, vol. 9, pp. 22 351–22 370, 2021.

[7] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 99–110, 2013.

[8] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[9] J. Liu, W. Hallahan, C. Schlesinger, M. Sharif, J. Lee, R. Soulé, H. Wang, C. Caşcaval, N. McKeown, and N. Foster, "P4v: Practical verification for programmable data planes," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on data communication*, 2018, pp. 490–503.

[10] T.-H. Lee, L.-H. Chang, and C.-W. Syu, "Deep learning enabled intrusion detection and prevention system over sdn networks," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[11] A. O. Alzahrani and M. J. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, p. 111, 2021.

[12] P. Golchin, R. Kundel, T. Steuer, R. Hark, and R. Steinmetz, "Improving ddos attack detection leveraging a multi-aspect ensemble feature selection," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–5.

[13] R. Kundel, L. Anderweit, J. Markussen, C. Griwodz, O. Abboud, B. Becker, and T. Meuser, "Host bypassing: Let your gpu speak ethernet," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 85–90.

[14] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," in *Proceedings of the 18th ACM workshop on hot topics in networks*, 2019, pp. 25–33.

[15] B. M. Xavier, R. S. Guimarães, G. Comarela, and M. Martinello, "Programmable switches for in-networking classification," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[16] J.-H. Lee and K. Singh, "Switchtree: in-network computing and traffic analyses with random forests," *Neural Computing and Applications*, pp. 1–12, 2020.

[17] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever, "pforest: In-network inference with random forests," *arXiv preprint arXiv:1909.05680*, 2019.

[18] Z. Aouini and A. Pekar, "Nfstream: A flexible network data analysis framework," *Computer Networks*, vol. 204, p. 108719, 2022.

[19] A. Hashemi, M. B. Dowlatshahi, and H. Nezamabadi-pour, "Ensemble of feature selection algorithms: a multi-criteria decision-making approach," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 1, pp. 49–69, 2022.

[20] M.-J. Jun, "A comparison of a gradient boosting decision tree, random forests, and artificial neural networks to model urban land use changes: The case of the seoul metropolitan area," *International Journal of Geographical Information Science*, vol. 35, no. 11, pp. 2149–2167, 2021.

[21] "GridSearchCV. An exhaustive search over specified parameters values for an estimator." (Accessed on 14.03.2023). [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[22] "P4 Language Consortium. (2021). P4Runtime Specification, Version 1.3.0." (Accessed on 04.02.2023). [Online]. Available: https://p4.org/p4-spec/p4runtime/main/P4Runtime-Spec.html

[23] N. Martindale, M. Ismail, and D. A. Talbert, "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data," *Information*, vol. 11, no. 6, p. 315, 2020.

[24] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.