Dynamic Adaptation of In-Band Network Monitoring via Meta Learning

Mingyuan Zang*, Eder Ollora Zaballa‡, Lars Dittmann‡ and Jie Wu*†
*Cloud Computing Research Institute, China Telecom, China
†Department of Computer and Information Sciences, Temple University, USA
‡Department of Electrical and Photonics Engineering, Technical University of Denmark, Denmark
Email: *zangmy1@chinatelecom.cn,‡{eoza, ladit}@dtu.dk, †jiewu@temple.edu

***Abstract*—Wide Area Network (WAN) management faces significant complexity in characterizing heterogeneous traffic from diverse services, which complicates unified feature extraction for operational tasks like anomaly detection. Key challenges persist in difficulty of creating generic feature sets across diverse traffic patterns and insufficient historical data for machine learning (ML) generalization. Meta-learning algorithms can be applied to learn dynamic feature sets to improve model generalizability on a limited number of data records. However, prior research focuses on algorithm design, with a lack of study on its application to in-band network monitoring. This work proposes a workflow to adaptively collect new feature sets and promptly learn from them. An adaptive in-band feature selection and extraction method is proposed for programmable switch. Meta learning algorithm is introduced for prompt decision in controller based on few-shot records. Evaluation results have shown that it outperforms prior methods in accuracy and inference time on public datasets.**

***Index Terms*—Programming Protocol-Independent Packet Processors (P4), Software-Defined Networks (SDN), Machine Learning (ML)**

I. Introduction

Wide Area Network (WAN) management faces unprecedented complexity in network traffic characterization due to the coexistence of diverse services with different requirements. The extraction of distinguishable traffic features is critical to characterize traffic for advanced operational services such as network optimization, anomaly detection, and traffic engineering. Yet, existing feature characterization and extraction methods remain three main challenges in efficient deployment. First, the high-dimensional heterogeneity of network traffic complicates unified feature representation. For instance, IoT devices generate periodic patterns tied to sensing intervals, enterprise virtual private networks (VPNs) demand stable low-latency transmission, and video streaming exhibits inherent burstiness due to variable bitrate encoding. This multidimensional variability makes it difficult to design a generic feature set that simultaneously captures temporal, statistical, and protocol-specific attributes. Second, dynamic configuration interference caused by operator-driven policy adjustments, such as quality of service (QoS) reconfiguration or traffic rerouting, introduces distributional shifts in observed features. These changes often invalidate pre-trained models that assume static network environments. Third, insufficient data prevent the deployment of ML-based methods (e.g., Decision Tree, Autoencoder) from generalizing to novel traffic patterns represented by new feature sets.

Traditional methods of network traffic collection are limited in parsing useful information from dynamic IoT traffic and evolving traffic patterns. These methods typically collect fixed fields of interest and rely on predefined protocols, which may not be effective in detecting novel or sophisticated attacks. On the other hand, Software-Defined Networking (SDN)-based solutions offer centralized control and programmability, enabling more flexible and scalable traffic monitoring over the network devices. SDN-based approaches allow for dynamic configuration and management of network resources, but they still face challenges in efficiently collecting the required network traffic features for machine learning-based analysis.

Another emerging approach is the use of programmable data planes, such as P4 (Programming Protocol-Independent Packet Processors) language [1], which enables fine-grained control over packet processing in network devices. Programmable data planes provide the ability to perform in-network computing and real-time analysis, making them well-suited for machine learning-based traffic analysis [2]. By collecting these features directly in the data plane, in-band feature collection offers several advantages. It reduces the need for external monitoring infrastructure, since the data plane itself can perform the necessary analysis and processing tasks. This approach can also enable faster and more efficient monitoring since the analysis is performed in real-time, without the need for data to be sent to external systems [3]. Additionally, in-band feature collection provides [4], [5] more granular and fine-grained information compared to traditional out-of-band methods (as depicted in Figure 1).

However, despite their advantages in programmability, it is still challenging to utilize runtime reconfiguration to improve the flexible in-band feature collection process, as all features need to be predefined in P4 code. Any changes in features need compile-time update, i.e. recompilation that will disrupt the network service. Additionally, all features that could be potentially used for collection may be parsed and sent at once, and let the ML model decide which features to use, but that could bring extra communication overhead and burden

the network bandwidth.

Moreover, once a monitoring task is deployed on programmable switch to collect traffic statistics based on a new group of feature sets, it poses another challenge to the ML-based classifier to give a prompt classification decision based on these new collected statistics. Prior methods are based on pretrained ML models and can only classify traffic based on a fixed set of features [6], [7]. A new group of feature sets will bring compatibility issues that hinder the ML models from performing, resulting in another round of ML model training and tuning. Meta-learning [8] could be a potential solution to address this challenge following the "learning to learn" paradigm, acquiring transferable knowledge across tasks during meta-training for quick adaptation to new scenarios with a few data samples. At the same time, there are gaps in the way to accommodate the workflow design to diverse tasks and services. The prior work focuses on algorithm design for anomaly detection tasks [9], without further study on workflow deployment in network systems.

Thereby, our question is: *How can in-band traffic feature analysis be efficiently and automatically reconfigured to efficient ML learning based on few-shot samples?* To answer this question, we propose a workflow to adaptively collect the new feature sets and promptly learn from them. It is featuring of 1) adaptive in-band feature selection and extraction in P4 program for programmable switch deployment; 2) support of common packet/flow-level features; 3) prompt classification decision in controller based on few-shot records collected from the data plane in programmable switches. The design is prototyped in software switches and evaluation results have shown its high efficiency in classification accuracy on few-shot samples and advantages in low-latency response compared with the state-of-the-art solutions.

Our main contribution to this paper can be summarized as follows.

- An adaptive in-band feature selection and extraction scheme in the programmable data plane is proposed by identifying its necessity.
- A meta-learning-based algorithm is introduced into the control plane to learn from the few-shot traffic statistics collected with new feature sets.
- A workflow is designed to optimize the end-to-end latency for prompt traffic analysis. Evaluation results have verified its efficiency over the state-of-the-art solutions.

The remainder of the paper is organized as follows. Section II explains the background of the work and presents an overview of related work. Sections III and IV demonstrate the key methods introduced in the work. Section V elaborates on the proposed end-to-end workflow design for dynamic adaptation of in-band network monitoring. Section VI describes the experimental setup for evaluation and lists the results. Section VII gives a discussion. Section VIII concludes the work and discusses future work.

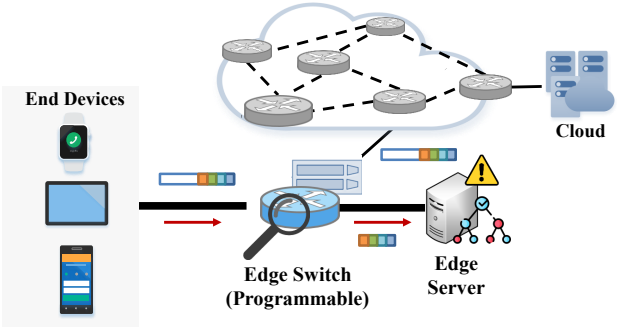


Fig. 1: A sample deployment of in-band feature extraction with the programmable device on network edge.

II. Background And Related Work

A. Background

1) Telemetry and Feature Extraction: Telemetry has been widely used for network monitoring and management. In the context of in-band feature collection, the programmable data plane can be programmed to identify and extract various types of information, such as packet headers, flow statistics, service indicators, or any other relevant network traffic attributes. This extracted information can then be used for real-time analysis, monitoring, debugging, or decision-making purposes within the network.

Generally, extractable traffic features can be classified into two types: stateless features and stateful features. These features are useful for traffic analysis to gain insight into the characteristics of the network. Table I lists the features that can be extracted in programmable data plane.

Stateless features are extracted directly from individual network packets. These features do not rely on previous packets to provide context information. They give fine-grained inspection of the properties of each packet and packet headers. Examples of stateless features include: a) packet header fields like address information, b) protocol information like versions or flags, c) QoS indicators, and d) detailed payload by conducting deep packet inspection into the metadata of packet payload.

Stateful features, on the other hand, involve analyzing the behavior and properties of a group of related packets over time. The grouping is usually based on 5-tuple information (i.e., source/destination IP, source/destination port, transport protocol). It identifies the sessions of the transport layer, taking into account the context of the packets exchanged in the sessions. These features describe the overall flow or session behavior. Examples of stateful features include: a) flow-level metrics such as packet counts and inter-packet arrival times (IAT); b) TCP connection like sequence/acknowledgment numbers; c) tracking of application layer services like session establishment/termination; and d) traffic statistics like minimum/maximum number of statistic counts.

2) P4-Enabled In-Band Telemetry: In-band feature collection with the programmable data plane refers to the capability of collecting information related to network traffic directly within the data plane of a network device. Such a technique

TABLE I: Examples of Stateful/Stateless Features

Stateless Features	Packet Header Fields	Source and destination addresses, Port numbers, Protocols, Packet size, Time-to-Live (TTL)
	Network Protocol Information	IP version, ICMP messages, ARP requests, TCP flags
	QoS	DSCP values, Class of Service (CoS) bits
	Payload	HTTP URLs, DNS query names
Stateful Features	Flow-level Information	Start and end times, Byte and packet counts, Duration, Inter-packet arrival times
	TCP Connection	Sequence/acknowledgement numbers, window sizes
	Application Layer	Transactions tracking , session establishment/termination
	Traffic Statistics	Minimum, maximum, total

is enabled by programmable Application-Specific Integrated Circuits (ASICs) and the Programming Protocol-Independent Packet Processors (P4) language. To support network programmability on the data plane, an abstract architecture PISA is proposed to perform custom operations on incoming packets in real time. This programmability allows for the extraction of specific information or features from the packet headers or payloads as they traverse the network device.

PISA architecture has three main components: parser, Match-Action tables, and deparser. P4 language is used to program this architecture to instruct packet processing. In detail, protocol stacks are defined in P4 language in a customized manner. To identify the protocol headers from the incoming traffic, parser runs a state machine to parse and extract packet header information layer by layer. The extracted information is temporarily saved in metadata. Match-Action (M/A) tables consist of Reconfigurable Match Tables (RMT) which decide how to handle the packet information by comparing the match conditions. If a table entry is matched, the corresponding action is taken to process the packet or instruct the forwarding to the next hop. Deparser will reconstruct the parsed packet header at the egress to make sure the packet contains the correct information after processing.

Specifically, RMT in M/A tables plays a crucial role in defining reconfigurable packet processing. RMT table consists of match fields, actions, and priorities. The packet processing pipeline performs a lookup operation in the Reconfigurable Match Table to determine the appropriate action for the packet. With P4Runtime interface, P4 allows dynamic updates to the table entries from the control plane, enabling runtime reconfiguration of packet processing behavior.

3) Meta-Learning Algorithm: Meta-learning algorithms offer a promising pathway to address these challenges. By leveraging few-shot feature adaptation, meta-learning algorithm like MAML (Model-Agnostic Meta-Learning) [8] can quickly fine-tune feature extractors for new services with minimal labeled data. For example, a meta-model trained in diverse services such as IoT could be adapted to extract jitter-sensitive features for a new AR application using only tens of labeled samples.

B. Related Work

Telemetry collection methods vary in different services and network infrastructures. Conventional collection methods for telemetry data mainly collect flow information to reduce the monitoring overhead to the network and volume of logs. When P4-based monitoring techniques are studied, packet-based solutions emerged because: 1) flow collection requires extra processing power; 2) packet-based telemetry gives more traffic details which reflects more problems. Collection Method The collection methods used in the related work are diverse, with a notable trend towards leveraging P4-enabled data planes. both utilize P4-enabled data planes for data collection, which allows for programmable and flexible data extraction directly from the network hardware. This method is highly efficient and can be tailored to specific needs, making it suitable for real-time applications such as anomaly detection and device recognition. Similarly, [13] and [14] employ P4-enabled data planes, highlighting its popularity in the field. However, [12] and [9] opt for a more traditional capture-based method, which may be less efficient but simpler to implement. The method proposed in [15] also uses a P4-enabled data plane, but their work does not specify a particular machine learning model. Our approach also leverages the P4-enabled data plane, benefiting from its flexibility and performance. This choice of collection method provides a strong foundation for real-time and adaptive feature extraction and analysis.

With a programmable data plane introduced to the network, the reconfigurable processing pipeline provides the potential for reconfiguration of collected telemetry. Conventional methods entail administrators manually reconfiguring the collection by modifying the collection rules, which adds workload and is error-prone. SDN or programmable data plane provides potential automation of the reconfiguration via automatic applications or services. Both [10] and [11] extract both flow and packet data, providing a comprehensive view of the network traffic. This dual-level extraction is beneficial for capturing both high-level trends and detailed packet information, which is essential for tasks like anomaly detection and device recognition. Both [12] and [14] focus solely on packet-level data, which can provide detailed insights but may miss broader patterns. The authors in [13] extract flow data only, which is more efficient but may lack the fine-grained details needed for certain analysis.

Programmable data plane has been widely used for In-network Telemetry (INT) services to collect telemetry information in an in-situ manner. INT embeds telemetry information directly into the data packets as they traverse the network. With programmable data planes, it is possible to collect fine-grained traffic features inside the network device and encode them within the packet headers. This enables detailed monitoring of network behavior. Note that this method is used in a group of network devices along the routing path, different from the single-device deployment

TABLE II: Comparison with Related Work

	Service	Collection Method	Flow	Packet	Reconfigurable	ML Model
[10]	Anomaly Detection	P4-enabled data plane	✓	✓	✓	Random Forest
[11]	Device Recognition	P4-enabled data plane	✓	✓	✓	Logistic Regression
[12]	IoT Fingerprint	Captures	x	✓	×	Random Forest
[13]	Anomaly Detection	P4-enabled data plane	✓	×	✓	Random Forest, K Nearest Neighbor, Support Vector Machine
[9]	Anomaly Detection	Captures	✓	✓	×	Meta-Learning
[14]	Encrypted Traffic Classification	P4-enabled data plane	×	✓	×	Random Forest
[15]	Application	P4-enabled data plane	✓	×	✓	×
Ours	Anomaly Detection, IoT Fingerprint, etc.	P4-enabled data plane	✓	✓	✓	Meta-Learning

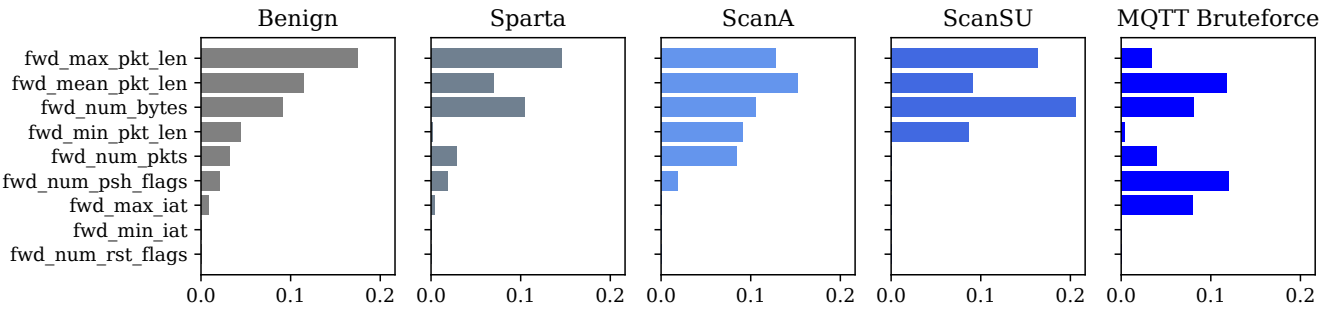


Fig. 2: Top 10 feature importance among 31 flow-level features in MQTT2020 [16].

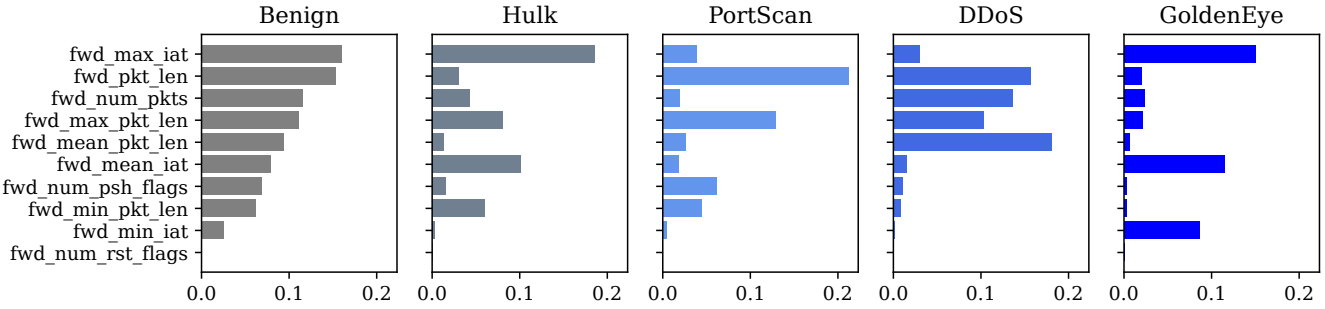


Fig. 3: Top 10 feature importance among 108 flow-level features in CICIDS 2017 [17].

discussed in this work. In this work, the focal lens is on feature collection and analysis in WAN on network edge.

C. Challenges and Gaps

Despite the advancements in the field, there are several gaps in the existing work. First, while some studies focus on either flow or packet data, few provide a comprehensive analysis using both. This can lead to incomplete insights, as flow data captures high-level trends while packet data provides detailed information. Second, the use of reconfigurable data planes is limited, with only a few studies leveraging this technology for real-time adaptability. Third, the choice of ML models is often narrow. Some studies use only one or two models, which may not be sufficient for diverse and complex datasets.

Our method addresses these gaps by integrating a P4-enabled data plane for flexible and efficient data collection, extracting both flow and packet data for a comprehensive analysis, and employing a meta-learning-based algorithm for adaptability to diverse monitoring services based on different feature sets. This holistic approach ensures that our method can handle various tasks, providing robust and reliable feature extraction and analysis. By leveraging reconfigurable data planes, our approach also supports real-time adaptability, making it suitable for dynamic network environments.

III. Feature Analysis

With programmable data plane flexibly parsing the packet header and processing the packets, various features and statistics are possible to be collected directly from the data plane. It can allow customizable feature extraction based on service demands, varying from low-level packet-based features and upper-level application-based features related to application sessions and flows.

Having so many types of features available to be extracted, traffic analysis and ranking is needed to select the meaningful ones. Meaningful features can contribute to more accurate learning and inference results of machine learning models, while meaningless features may hurt model learning performance. To quantify the significance among available features, feature importance is one classical method. Permutation importance-based feature importance is scored by randomly shuffling values of a single feature and observing how much the model's performance drops. The greater the performance drop, the more important the feature is.

To give an example analysis, features in two public datasets are analyzed and compared: CICIDS 2017 [17] and MQTT-IoT-IDS2020 [16]. These datasets include both benign and malicious traffic from various services and protocols in multiple types of IoT scenarios. Both flow-based and packet-based features are listed in these datasets and each feature is of different importance in revealing the patterns. Thus, feature importance is computed and compared to understand how feature importance may vary in each case.

Feature importance in different traffic patterns. Figure 3 and Figure 2 present the rank of feature importance scores to evaluate how the features can reveal the traffic patterns, i.e., malicious attacks in this case. MQTT-IoT-IDS2020 [16] includes traffic captures specifically recording MQTT-related attacks in IoT scenarios, while CICIDS 2017 [17] includes daily traffic captures suffered from DoS/DDoS attacks. In these figures, we use the feature types shared in both datasets for comparison and analysis. The comparison indicates that: The same group of features shows different levels of importance in identifying different attacks in the same dataset. For instance, to identify volumetric attacks (e.g., aggressive scanning (ScanA), UDP scanning (ScanSU), or brute-force attempts) packet-level features like IP TTL and TCP/UDP flags receive high importance scores. Conversely, for stealthy attacks that target specific protocols (e.g., Sparta SSH brute-force and MQTT brute-force), features that closely reflect the intricacies of the MQTT protocol itself are assigned greater importance.

IV. Meta-Learning-Based Classification

Meta-learning operates on the fundamental paradigm of "learning to learn", where models acquire transferable knowledge across diverse tasks during meta-training to enable rapid adaptation to novel scenarios with minimal data. This approach simulates real-world few-shot conditions through episodic training: each episode presents a synthetic task comprising a small support set (for adaptation) and query set (for evaluation), forcing the model to develop generalizable feature representations and adaptive learning strategies. The core mechanism involves bi-level optimization, an inner loop performs task-specific parameter tuning while an outer loop refines the model's global architecture to facilitate efficient future adaptations. By exposing the model to thousands of such simulated few-shot scenarios, it learns optimal initialization parameters and update rules that minimize the need for extensive retraining when encountering new tasks.

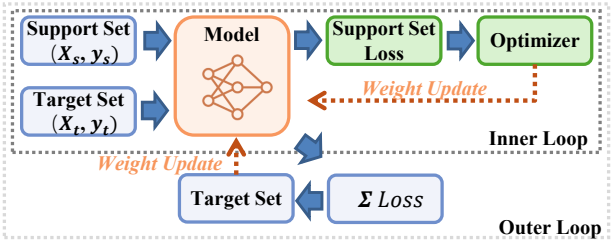


Fig. 4: Overview of the meta-learning algorithm.

This work introduced Model-Agnostic Meta-Learning (MAML) as a meta-learning algorithm for few-shot classification. During meta-training phase, the model processes batches of episodic tasks drawn from base classes [18]. Figure 4 demonstrates the training process of the algorithm. For each task, the inner loop executes rapid adaptation: the classifier module undergoes a few iterations of gradient updates to update the initial parameters θ from the optimized model $f_{\theta_i'}$ using only the support set based on $\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i} \left(f_{\theta_i'}\right)$, while the feature extractor remains frozen to preserve generalized representations. The algorithm incorporates gradient correction, replacing null derivatives with zero tensors, to maintain stability when processing sparse network traffic features. Following inner-loop adaptation, the outer loop computes meta-loss on the query set and back-propagates through the entire computation graph of inner updates, adjusting both feature extractor and classifier weights via mixed-precision training. This dual-loop strategy enables the model for task-specific specialization (inner loop) and cross-task generalization (outer loop).

For few-shot inference phase, the meta-trained model demonstrates its adaptive capabilities. When presented with new traffic classes (e.g., 5 samples of a new class), the feature extractor first generates discriminative embeddings using weights optimized for cross-task feature abstraction [19]. The classifier then performs a few inner-loop updates exclusively on these few samples, dynamically adjusting decision boundaries without changing the foundational feature representations. This process leverages an insight from meta-training: optimal initialization points that lie in parameter regions follows fast adaptation. Enhanced design like *OneCycleLR* scheduling [20] further boost efficiency by automatically tweaking learning rates, initially high for coarse adjustment, then decaying for fine-tuning, while *BatchNorm* layers maintain feature distribution consistency across various tasks.

V. Proposed Design

A. Workflow Overview

The workflow proposed in this paper is built on P4-programmable data planes. Figure 5 illustrates the framework overview, which consists of two primary components: a P4-enabled switch and a controller. When network traffic reaches the P4 switch, the switch extracts stateful in-band features, encapsulates them in a UDP packet (feature report), and

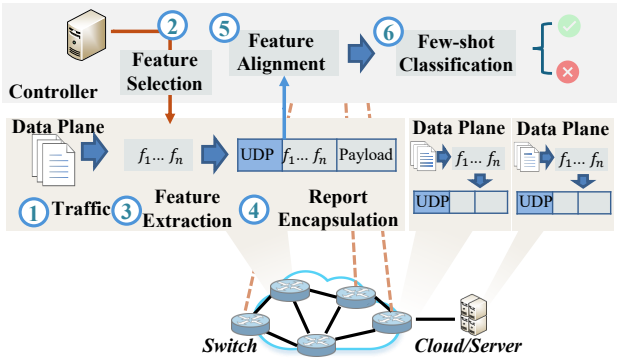


Fig. 5: Overview of the proposed design.

forwards the packet to the controller. The controller then classifies the traffic using a pre-trained ML model.

Detection in the P4 switch proceeds in three stages:

- Packet parsing: Header fields are parsed and protocol information is extracted, as shown in Figure 5, step ①.
- Statistic extraction: Stateful features are computed by hashing into bloom filters and updating counter registers, illustrated in Figure 5, steps ② and ③.
- Report encapsulation: When a packet arrives at the ingress port, a timestamp is recorded. Once the timestamp exceeds time window T, collected features $\{f_1, ..., f_n\}$ are read from registers. This metadata is assembled into a custom header and encapsulated in a UDP report, as depicted in Figure 5, step ④.

When the controller receives the report encapsulated in UDP packet from the switch (Figure 5, step ④), features are parsed into a *DataFrame* and labeled by a pre-trained meta-learning model designed for few-shot classification. The model is trained offline and then loaded for real-time inference on a small number of examples (Figure 5, step ⑥). Because the collected features evolve when new sets arrive, feature alignment is performed before the classification task (Figure 5, step ⑤).

B. Feature Collection And Alignment

In-Band Feature Collection. With P4-enabled programmable switches deployed, the features discussed in Figures 2 and 3 are collected in band upon arrival of traffic at the data plane. To enable meta-learning based on a few examples, common features $\{f_1, ..., f_n\}$ collected within the time window are flow information based on 5-tuple like port information and protocol {*source port, destination port, protocol*}.

We focus on these features to capture trends of traffic arrival rather than precise source or destination details, because attackers may use spoofed IP addresses. For example, a TCP SYN flood generates many half-open handshakes: the flood of SYN packets causes a sudden rise in TCP traffic and new sessions, and when those sessions time out, a burst of RST packets follows. Such anomalies are visible in the packet count, rate, and TCP-flag distribution observed within each time window.

To collect these features, counters are first stored in temporary 32-bit registers. This width is defined to balance the need to handle high-volume attack traffic against memory consumption. A Bloom filter, a space-efficient, probabilistic data structure, is then used to identify unique TCP port pairs in the heavy traffic without storing the full pair information.

Feature Alignment. When the controller receives the collected feature values from various switches, cross-switch feature alignment is triggered. This module resolves feature schema disparities between heterogeneous data sources, which is critical in WAN environment where diverse services coexist. The key idea is to group the traffic features and find the representative common features. The standardization process normalizes numerical features like flow duration and port numbers, while label encoding converts categorical attack types into unified numerical representations. This phase outputs dimensionally consistent tensors ready for model ingestion, ensuring compatibility between pre-training data (e.g., port scan traffic in the first group of data) and incremental samples (e.g., emerging traffic patterns in incoming data).

C. Adaptive Feature Selection

To allow adaptive feature selection for dynamic demands in operational service, this work introduces an idea of *precomputation and selection* to enable dynamic feature set switching without recompilation of P4. During the initial compilation phase, the P4 program is designed to compute all potential features in parallel, storing them in dedicated registers or metadata fields. This precomputation occurs continuously as packets traverse the data plane, ensuring all features remain updated regardless of current selection. A feature selector table acts as a runtime switchboard, controlled by a bitmap that determines which precomputed feature gets collected and sent to the control plane.

In detail, the control plane populates the feature selector table with default mappings. The incoming network traffic triggers concurrent feature updates such as feature extraction and counter incrementation, while timestamps enable duration calculations. When a flow termination is detected (e.g., via idle timeout), the current selector value indexes the feature table, triggering only the selected feature to be cloned to the controller via packet-out. This design decouples feature selection from feature computation, allowing the latter to be reconfigured on the fly while maintaining line-rate processing performance.

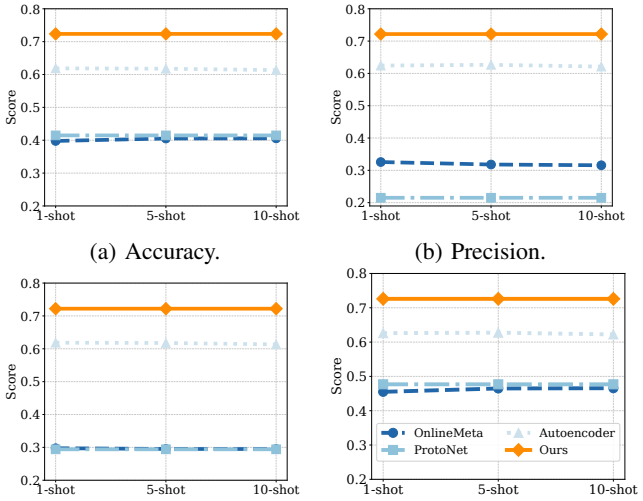
VI. Experimental Setup and Results

A. Experimental Setup

To evaluate the proposed design, we created a Mininet topology with six hosts acting as iperf3 TCP/UDP servers, a seventh host serving as an iperf3 client for each server, and a controller. We implemented our design in $P4_{16}$ [1] using the BMv2 Simple Switch target to enable data collection in the band. We then train the meta-learning model and replay the public datasets for evaluation.

TABLE III: Model Performance

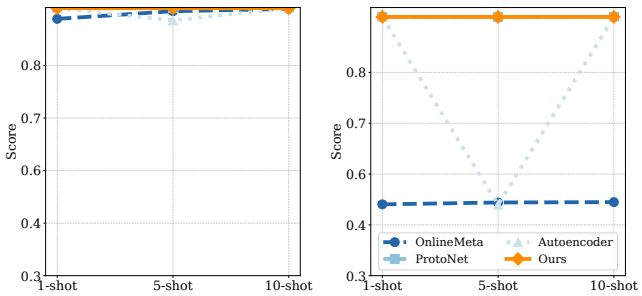
	Accuracy	Precision	Recall	F1	Train Time (ms)	Infer. Time (ms)
AE	0.71	0.72	0.72	0.71	45	612
ProtoNet	0.43	0.33	0.50	0.30	25	189
OnlineML	0.43	0.45	0.49	0.34	47	1068
Ours	0.75	0.75	0.76	0.75	2357	792



(a) Accuracy. (b) Precision.

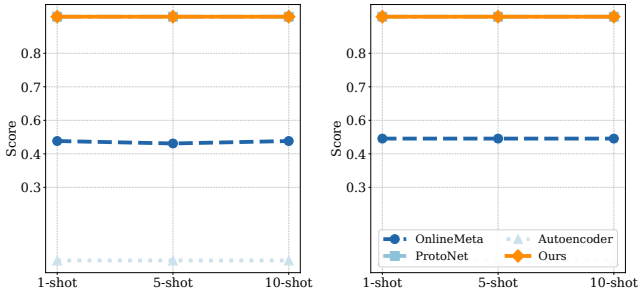
(c) F1 score. (d) Recall.

Fig. 6: Acccuracy vs. number of shots on CICIDS2017 dataset.



(a) Accuracy. (b) Precision.

Fig. 7: Accuracy vs. number of shots on IoT Sentinel dataset.



(a) Accuracy. (b) Precision.

Fig. 8: Accuracy vs. number of shots on MQTT2020 dataset.

Public datasets are used for evaluation: CICIDS 2017 [21], IoT Sentinel [12], MQTT2020 [16]. To evaluate the adaptive performance of the proposed design on emerging traffic pattern, the datasets are replayed in two types of scenarios: a) initial traffic with class 1 and followed by traffic with class 2 in the same CICIDS 2017 dataset; b) initial traffic in dataset CICIDS 2017 and followed by traffic in dataset IoT Sentinel or MQTT2020 [16].

Multiple metrics are used to evaluate detection performance of the proposed design. Here, TP, FN, FP, TN denote the counts of true-positive, false-negative, false-positive, and true-negative results, respectively. Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$ measures the overall proportion of correct classifications. Precision $\frac{TP}{TP+FP}$ indicates how reliable a positive label is. Recall $\frac{TP}{TP+FN}$ reflects the fraction of actual anomalies that are correctly detected [22]. The F1-score $2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ gives a harmonic mean of Precision and Recall. The False Positive Rate (FPR) $\frac{FP}{TN+FP}$ shows the proportion of benign traffic that is incorrectly flagged.

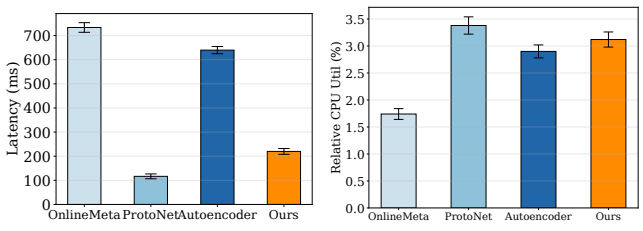
B. Experimental Results

In this section, we evaluated the design from three aspects: (1) classification capability, (2) adaptive classification, (3) system overheads brought by the proposed workflow.

Effects from model selection: Three meta-learning models [23], [24], [9] were evaluated as state-of-the-art solutions. The widely-used Autoencoder (AE) model is also compared and evaluated as baseline [24]. Table III summarizes the performance details of these models. Although AE present a faster inference process, but it takes a relative long training time. ProtoNet shows faster inference but lower accuracy on new traffic pattern. Overall, the algorithm proposed in this work has the best performance, the highest accuracy to 75% with better trade-off between training and inference time.

Effects from the number of shots: Figure 6, 7 and 8 demonstrate the inference accuracy when labeling the traffic representing different number of data shots. Results in Figure 6 show that the proposed model presents an adaptive classification performance with stable and high accuracy performance. When the number of shots is low (e.g., 1-shot), the accuracy decreases by 1% from the 5-shot case for Autoencoder and naive meta-learning algorithm. When it comes to the IoT sentinel dataset, the results present a similar trend. The proposed algorithm still shows stable performance with relatively high accuracy and outperforms other algorithms by 8% to 17% higher accuracy. Similar trend is also presented in other datasets in Figure 7 and 8.

Effects from the system latency: During the detection, the end-to-end latency from feature collection and transmission within the P4 switch until the ML-based inference result in the controller is recorded as Figure 9. While the ProtoNet presents a 1.5× faster latency performance than the algorithm proposed in this work, its accuracy is relatively low. Despite the relatively high CPU utilization of the proposed algorithm, it achieves a better trade-off between accuracy and CPU utilization compared to other SOTA.



(a) (b)

Fig. 9: System performance of (a) end-to-end inference time, (b) Relative CPU utilization.

VII. Discussions

Data Plane Resource Constraints. The dynamic feature extraction capability relies on the limited memory in programmable switches. When high-dimensional features are activated simultaneously, contention for memory resources may occur, potentially affecting forwarding rules and other network functions. This requires advanced feature analysis mechanisms to prioritize critical features. Such priority results can be varied and depend on security-relevant attributes.

Meta-Learning Generalization. During initial deployment, the adaptability of the meta-model may be affected by the pre-training data. For instance, if the controller encounters new traffic distribution, the model may fail to rapidly generalize from minimal samples. More data and updates of model parameters may be needed to enhance model's reliability.

Temporal Latency and Overhead in Feature Statistics. Stateful time-windowed features introduce decision delays for short-lived anomalies. Although real-time stateless features can compensate, dynamically reconfiguring the pipeline to balance batch-statistical features versus real-time analysis increases control-plane complexity. The workflow proposed in this work approaches this problem by introducing the meta-learning-based updates.

VIII. Conclusions

This work proposes a workflow to adaptively collect the new feature sets and promptly learn from them. It demonstrates how in-band feature analysis can be dynamically reconfigured for efficient few-shot learning. By integrating adaptive P4-based feature selection in programmable switches with meta-learning in the control plane, the proposed solution avoids manual recompilation overhead and enables real-time classification of novel traffic patterns. The optimized workflow significantly reduces end-to-end latency while maintaining accuracy with minimal samples, overcoming limitations of static feature sets and pretrained models. This approach advances network automation by providing a responsive, reconfigurable framework for next-generation traffic engineering and operational services in dynamic network environments. Future work regarding adaptive in-band monitoring will likely focus on deploying the meta-learning-based method presented in this paper on hardware programmable switches and investigating on scalable deployment scheme on the programmable switch.

References

[1] "P416 language specification version 1.2.0," https://p4.org/p4-spec/docs/P4-16-v1.2.0.html.

[2] C. Zheng, M. Zang, X. Hong, and et al., "Planter: Rapid Prototyping of In-Network Machine Learning Inference," *ACM SIGCOMM Computer Communication Review*, 2024.

[3] D. Ding, M. Savi, F. Pederzolli, and et al., "In-Network Volumetric DDoS Victim Identification Using Programmable Commodity Switches," pp. 1–16, 2021.

[4] M. Zang, C. Zheng, T. Koziak, and et al., "Federated in-network machine learning for privacy-preserving iot traffic analysis," *ACM Transactions on Internet Technology*, 2024.

[5] D. Barradas, N. Santos, S. Signorello, and et al., "FlowLens : Enabling Efficient Flow Classification for ML-based Network Security Applications," *Proceedings of NDSS*, no. February, pp. 1–18, 2021.

[6] J. Fan, K. Wu, Y. Zhou, and et al., "Fast model update for iot traffic anomaly detection with machine unlearning," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8590–8602, 2022.

[7] M. Zang, E. O. Zaballa, and L. Dittmann, "Sdn-based in-band ddos detection using ensemble learning algorithm on iot edge," in *ICIN*, 2022, pp. 111–115.

[8] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[9] N. Niknami, V. Mahzoon, and J. Wu, "Ptn-ids: Prototypical network solution for the few-shot detection in intrusion detection systems," in *LCN*, 2024, pp. 1–9.

[10] N. Gray, K. Dietz, M. Seufert, and et al., "High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems," *IEEE International Conference on High Performance Switching and Routing, HPSR*, vol. 2021-June, 2021.

[11] M. Jose, K. Lazri, J. François, and et al., "Leveraging in-network real-value computation for home network device recognition," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 734–735.

[12] M. Miettinen, S. Marchal, I. Hafeez, and et al., "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems*, 2017, pp. 2177–2184.

[13] F. Musumeci, V. Ionata, F. Paolucci, and et al., "Machine-learning-assisted DDoS attack detection with P4 language," *IEEE International Conference on Communications*, 2020.

[14] A. T.-J. Akem, G. Fraysse, and M. Fiore, "Encrypted traffic classification at line rate in programmable switches with machine learning," in *NOMS*, 2024, pp. 1–9.

[15] C. Györgyi, S. Laki, and S. Schmid, "P4rrot: Generating p4 code for the application layer," *SIGCOMM Comput. Commun. Rev.*, vol. 53, no. 1, p. 30–37, Apr. 2023.

[16] A. Alatram, L. F. Sikos, M. Johnstone, and et al., "Dos/ddos-mqtt-iot: A dataset for evaluating intrusions in iot networks using the mqtt protocol," *Computer Networks*, vol. 231, p. 109809, 2023.

[17] "Intrusion detection evaluation dataset (CIC-IDS2017)," https://www.unb.ca/cic/datasets/ids-2017.html.

[18] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *ICML*, 2019, pp. 1920–1930.

[19] M. A. Jamal and G.-J. Qi, "Task agnostic meta-learning for few-shot learning," in *IEEE/CVF CVPR*, 2019, pp. 11 719–11 727.

[20] A. Al-Kababji, F. Bensaali, and S. P. Dakua, "Scheduling techniques for liver segmentation: Reducelronplateau vs onecyclelr," in *International conference on intelligent systems and pattern recognition*. Springer, 2022, pp. 204–212.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[22] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *European conference on information retrieval*. Springer, 2005, pp. 345–359.

[23] J. Tian, M. Li, Z. Wang, and et al., "Omlog: Online log anomaly detection for evolving system with meta-learning," *IEEE Internet of Things Journal*, 2025.

[24] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *CoRR*, vol. abs/1802.09089, 2018.