MATADOR: ML-based Cloud Gaming Traffic Detection entirely in Programmable Hardware

Suneet Kumar Singh, NTNU, Christian Esteve Rothenberg, Alireza Shirmarz
Fábio Luciano Verdi, Israat Haque, Gyanesh Patra, Gergely Pongrácz
Universidade Estadual de Campinas (UNICAMP), Brazil
NTNU Norwegian University of Science and Technology, Norway
Federal University of São Carlos, Brazil Dalhousie University, Canada
Ericsson Research, Hungary

***Abstract*—Cloud gaming (CG) is gaining increasing interest as a new approach to delivering high-quality gaming anywhere, anytime, and on diverse devices. To improve the gaming experience, CG traffic needs to be monitored and prioritized to avoid high latency and jitter. Recent attempts to classify CG traffic on cloud nodes are time- and resource-consuming. Improving CG detection time and accuracy is critical to ensuring a responsive and seamless experience. This paper proposes MATADOR, a machine learning system that classifies CG traffic entirely in P4 programmable hardware. Our design and implementation using a Tofino switch overcome P4 challenges for hardware targets. The performance is analyzed for Tofino and a Python-based simulator. The obtained results demonstrate that, with an accuracy of around 97%, our approach significantly reduces the CG detection time when compared against alternative approaches.**

***Index Terms*—Machine learning, Cloud gaming, P4, SDN**

I. Introduction

Cloud gaming (CG) is a server-based gaming solution that runs games on remote servers in data centers. CG tries to bring high Quality of Experience (QoE) to users by incorporating streaming technologies and network infrastructure optimization. Nvidia GeForce Now, Microsoft xCloud, PlayStation Now, and Amazon Luna are well-known CG platforms. However, prior research [12] also indicates that QoE may be adversely affected by network conditions, including bandwidth, latency, and jitter. Even with high bandwidth network conditions, CG can suffer QoE degradation primarily due to latency and jitter. To overcome this issue, CG traffic must be continuously monitored and given priority over non-latency-sensitive applications.

To prioritize latency-sensitive traffic across queues in network devices, we typically employ packet header information, such as the ECN bit in the packet's IP header. Nevertheless, these classification methods might be less precise because of errors in configuration or malevolent actions by the end host. To circumvent these issues, we employ an intelligent classifier that analyzes traffic and discerns latency-sensitive traffic from millions of flows. According to current initiatives aimed at using machine learning (ML) for traffic categorization, particular application flows can be identified by their features [4]. The feature computations and ML models are typically carried out on network functions virtualization (NFV) nodes running on commodity servers because they involve complicated functions and calculations [4]. These software-based approaches take longer to identify lower latency traffic because they need more CPU cycles to implement rules, apply ML models, and gather dataplane information [7]. In contrast, in-networking computing solutions can significantly enhance the CG experience by shortening detection time.

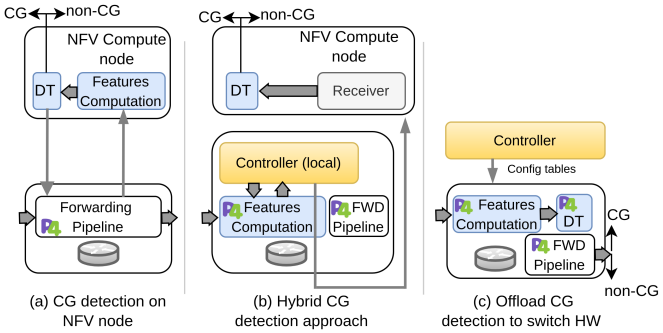

Fig. 1: Cloud Gaming detection approaches.

Recent developments in programmable data planes [3] allow supervised and unsupervised ML algorithms like decision trees (DT), K-means, Support Vector Machines (SVM), and Naïve Bayes to be mapped with switch pipelines for classification [17]. Even though P4 for Tofino Native Architecture (TNA) is capable of executing these algorithms, fitting ML models with features that need to be calculated and not just taken from packet headers and forwarding action within the restricted switch resources remains a challenging task. Additionally, choosing features is crucial in avoiding complicated P4 calculations like mean value and standard deviation. As seen in Figure 1(a) and 1(b), the majority of state-of-the-art proposals [4] either use a hybrid approach or execute the entire process on an NFV node [5].

Our work proposes MATADOR, a CG classification system that uses the Tofino switch to execute feature calculation and the DT method, as shown in Figure 1(c). We demonstrate its feasibility using two real datasets. To make processing compatible with P4, we consider exponential weighted moving average (EWMA) for feature computation instead of relying

on mean or standard deviation. Also, we discuss P4 language's constraints and implementation difficulties. Our proposal includes features necessary for CG detection that are compatible with P4-TNA. Furthermore, we optimize the DT method to fit into fewer Tofino stages by reducing dependencies between match-action tables. MATADOR performs the quick and accurate classification of CG traffic at line rate, with an accuracy of around 97%. For reproducibility, we share all pertinent code and documentation in a public repository[1].

II. Background and Related Work

ML has been progressively included in networking to address intricate issues, improve network efficiency, fortify security, and automate related management tasks [8] [9]. Recent attempts to successfully offload ML algorithms to programmable data planes allow to satisfy the requirements for 5G and 6G use cases. Programmable data planes permit customized packet processing logic, which aids in achieving low-latency data processing. With these advantages, numerous applications, including 5G User Plane Function (UPF) and Heavy Hitter (HH) detection, are specified in programmable switch hardware [10]. Deploying ML models directly onto dataplane devices can be difficult because to their limited computing and memory resources. This is because ML models can be resource-intensive. Additionally, ML models that are employed in dataplane applications need to balance complexity and accuracy. Resources-constrained contexts may pose challenges for the deployment and management of complex models.

ML is widely utilized in traffic classification, namely in CG, heavy-hitting flows, and Denial-of-Service (DoS) attacks. For CG categorization, software (SW) and hybrid-based methods are employed. In software solutions similar to that found in [4], DT and Random Forest (RF) models are employed for CG classification using flow-level properties (e.g., mean packet size, average and standard deviation of inter-arrival times). However, an unsupervised machine learning model is employed to categorize CG traffic using a hybrid P4/NFV technique in hybrid solutions like in [5]. The dataplane does the feature calculation (e.g., packet size and inter-arrival times), and it reports the results to the local P4 controller regularly. The NFV node does further calculations to determine CG flows. Currently, no fully programmable hardware classification for CG is available due to the difficulties in executing ML models and simultaneously carrying out complicated feature calculations.

In addition to CG classification, using P4 switches, [11] computes the features required for image classification using the DT method and extracts pixel information. However, a software switch (BMv2) is used to assess the solution. A similar approach to long-lived flow classification is used in [6]. P4 switches (Bmv2) are used in the Mininet environment to assess the detection. In [2] and [13], an Intel Tofino switch ASIC is used to implement the RF model and binary decision tree, respectively, for traffic classification, including elephant flows and IoT device identification. However, compared to performing complex calculations for feature computations, packet header information are used as an input for the ML model, and thus fits easily within the constraints of programmable hardware.

Our work differs from previous implementation attempts in the following ways. CG classification on switch line rate is the main emphasis of MATADOR. We choose features that are simple to compute in the dataplane and provide high accuracy. Within the constraints of Tofino ASICs, we employ these features as an input to the ML model in the same switch pipeline for CG categorization. For feature computations, MathUnit, hash functions, and registers are utilized.

III. MATADOR Design and Implementation

The proposed architecture of MATADOR is shown in Figure 2. The P4 implementation (one thousand lines of P4 code) comprises several stages, which will be covered in more detail below, and is coordinated through a programmable switch and an SDN controller.

A. Line Rate Feature Extraction

The selection of features is primarily based on CG traffic. However, the same features may also be considered for similar delay critical application traffic (e.g., virtual and augmented reality). Large packet sizes and frequent packet arrivals in the downlink direction are expected for CG traffic. Multimedia flows are regarded as downlink, and player commands are regarded as uplink. Six features are taken into consideration: the number of packets per time window (TW) of a flow for both directions, as well as the moving average of the packet size (PS) and the inter-packet gap (IPG) for both directions.

To compute these features in the data plane, we consider a set of network flows $f \in \mathcal{F} = \{f_1, f_2, ..., f_N\}$, where N indicates the total number of flows. The IPG and PS metrics are updated for each incoming packet entering the switch of $f \in \mathcal{F}$. The last ingress timestamp ($TS_f^l \in \mathbb{N}^+$) is deducted from the current timestamp ($TS_c \in \mathbb{N}^+$) to compute the current IPG_f^c of network flow f. Then, an EWMA metric IPG_f^w and PS_f^w are calculated:

$$IPG_f^w = \alpha \cdot IPG_f^{w-1} + \ (1-\alpha) \cdot IPG_f^c, \quad f \in \mathcal{F} \qquad (1)$$

$$PS_f^w = \alpha \cdot PS_f^{w-1} + \ (1-\alpha) \cdot PS_f^c, \quad f \in \mathcal{F} \qquad (2)$$

where $\alpha \in [0, 1]$ is the smoothing factor and IPG_f^{w-1} and PS_f^{w-1} are the last noted weighted IPG and PS, respectively. The packet header information is used to compute PS_f^c. Using Equations 1 and 2, we calculate the EWMA of IPG and PS with greater α (i.e., range 0.90 to 0.99) for both directions. Increasing the weight assigned to the last observed weighted IPG can help mitigate any abrupt variations in traffic patterns. The Tofino switch ASIC has seven stages dedicated to feature processing, as depicted in Figure 2.

[1]https://github.com/intrig-unicamp/cg-classifier

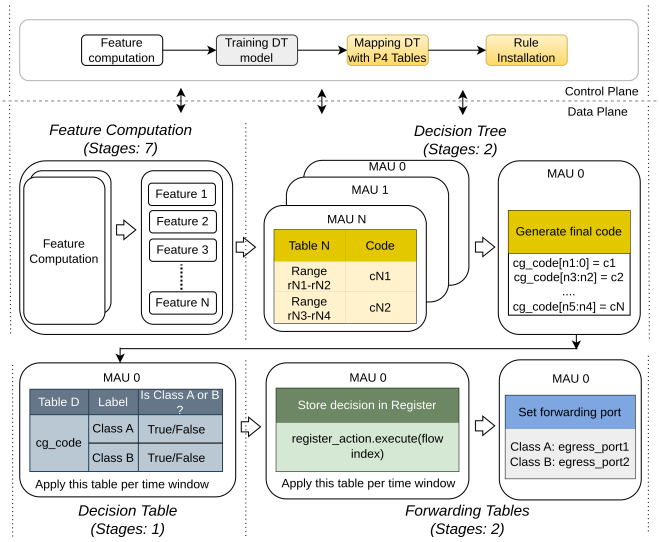

Fig. 2: Overview of MATADOR programmable pipeline.

B. P4-TNA Switch DT Realization

The proposed design of the DT algorithm for P4-TNA is inspired by the idea discussed in [17]. Due to the limited number of stages in Tofino switch hardware, executing if-else conditions on each DT node in P4 is not feasible. We match each feature with all of its possible values in a single Tofino stage as a solution. As an example, Figure 2 shows how matches are made for each feature and its possible ranges once all feature computations are finished, resulting in a unique code. The produced codes are kept in metadata, then utilized to categorize the traffic into distinct groups. The final two blocks, as shown in Figure 2, can be used to map the traffic class to distinct priority queues or to configure the egress port. Furthermore, no modifications to the DT are necessary for the P4 pipeline. All that is needed is to map the model in the dataplane at runtime and configure the control plane.

C. Control Plane

As seen in Figure 2, training the DT model and setting up the P4 table for mapping with the match-action tables are done in the control plane (CP). There are two sets in the dataset: test and training. The decision tree with all the nodes and feature values on each node with criteria is generated by CP using the training data to train the DT model. The DT algorithm is applied using Python scripts, and the Barefoot runtime (BFRT) APIs are called in order to populate the P4 match-action tables.

D. P4-TNA Deployment Limitations

EWMA Computation. According to Equation 1 and 2, the switch hardware presents a challenge for EWMA computation. P4 switch hardware, like P4 Tofino, only supports basic division and multiplication operations. For these kinds of computations, there is a feature called Low Pass Filter (LPF) available. LPF comes in two flavors: Sample and Rate. Rate mode offers the traffic rate within a given time interval, whereas sample mode provides an approximation EWMA

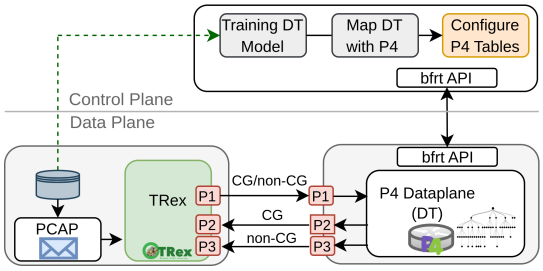

Fig. 3: Testbed for evaluating CG classification.

calculation. Fixing the smoothing factor in sample mode is difficult, but it is necessary for improving accuracy. Without utilizing LPF, our technique is predicated on an approximative EWMA computation. As a next step, we examine the application of LPF with a variable smoothing factor.

Number of Stages. To ensure reduced latency, P4 switch hardware (Tofino) has a packet forwarding pipeline with a restricted number of stages. The number of stages is contingent upon the feature computation, DT algorithm, and forwarding operations, as expounded in Section III-B. We cannot parallelize both processes because the DT algorithm applies only after feature computations. Although Tofino 2 has more flexibility than Tofino 1 regarding the number of stages and resources, we still need to carefully choose the features and attempt to limit them without sacrificing overall accuracy.

IV. Experimental Evaluation

A. Datasets

CG training dataset. The ML model is trained using 20% of PCAP datasets[2] in the same manner as described in [4] [5]. The four main CG platforms—xCloud (cg_xc), PlayStation (cg_psn), GeForce (cg_gfn), and Stadia (cg_std)—are used to train the ML model. The CG dataset contains a wide range of frame rates and resolutions. While user input has led to moderate packet sizes for the uplink, multimedia streams result in enormous packet sizes for the downstream. For the non-CG traffic class, the following applications are considered: video streaming (live streaming: non_cg_ls, game: non_cg_g, remote desktop protocol: non_cg_rdp, youtube: non_cg_vod), video conferencing (non_cg_vc), and Facebook navigation (browsing: non_cg_b). These applications usually have higher bitrates than CG flows, but latency needs are less stringent.

CG test dataset. The evaluation process makes use of two distinct datasets. One is the same as previously mentioned (i.e., 80% of the dataset). Another test dataset, collected using PCAP format and entirely taken in a different lab setup, displays cloud gaming activity from the Xbox cloud gaming platform. This dataset contains two scenarios: one using 5G wireless technology and the other using cable internet technology. Over two days, Leris Lab gathered the data, which includes single-player and multiplayer games like Fortnite, Mortal Combat, and Forza[3].

[2]https://cloud-gaming-traces.lhs.loria.fr
[3]https://github.com/dcomp-leris/VR-AR-CG-network-telemetry

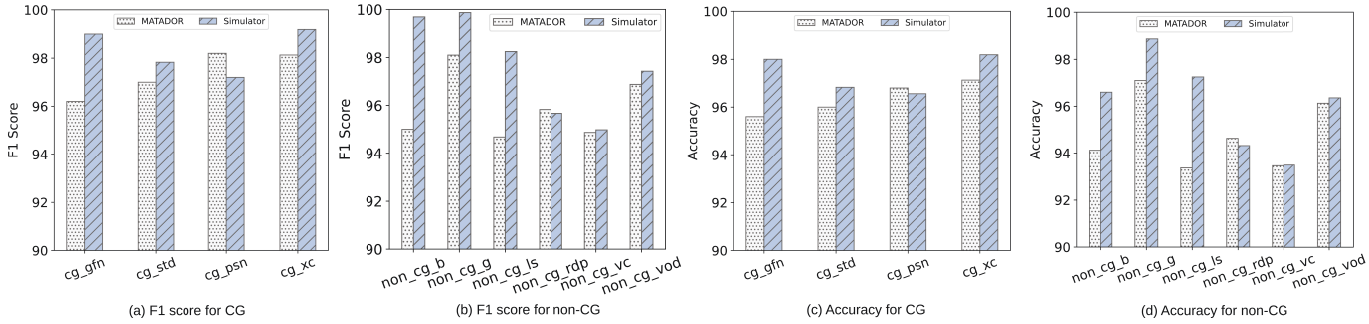

Fig. 4: Performance comparison of MATADOR with simulator outcomes.

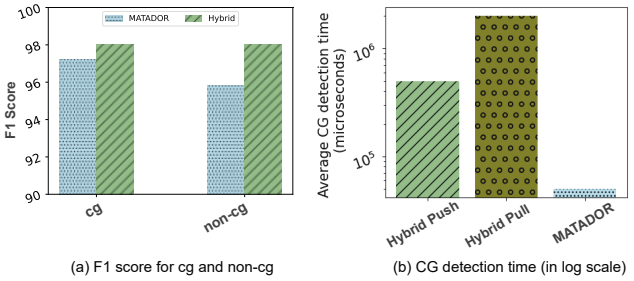

Fig. 5: Performance comparison with the state-of-the-art.

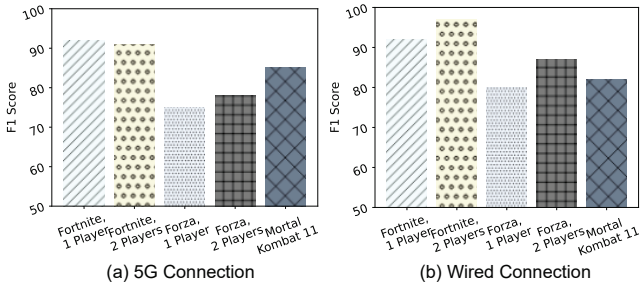

Fig. 6: Performance using a new cloud gaming dataset.

B. Experiment Setup

Testbed. An x86 server equipped with an Intel Xeon D-1518 processor (4 CPU cores, 2.20 GHz) and a *Barefoot Tofino switch ASIC* (Edgecore Wedge 100BF-32X) for CG classification are used to analyze the performance. A testbed with an x86 server connected to a P4 Dataplane (Tofino) via 10G SFP+ ports is depicted in Figure 3. The traffic is sent via TRex, which operates on an x86 server. A local Python-based controller maps the learned DT model with match-action tables in Tofino in order to configure the necessary table entries.

Evaluation Metrics. Accuracy, F1 Score and detection time. TP (True Positive) are CG flows that are correctly detected, while FP (False Positive) are those flows that are non-CG but detected as CG, and FN (False Negative) are the flows that are CG but detected as non-CG. TN (True Negative) are non-CG flows that are correctly classified. We evaluate the Precision ($Pr = \frac{TP}{TP+FP}$) and Recall ($R = \frac{TP}{TP+FN}$) metrics. Accuracy is evaluated as ($\frac{TP+TN}{TP+FP+TN+FN}$) and the F1 score as $\frac{2\cdot(R\cdot Pr)}{(R+Pr)}$. We use a microsecond scale to measure CG detection time.

C. Test Execution

We employ the CG test dataset given in Sec. IV-A for evaluation. To evaluate MATADOR, we replay these traces at the same rate that pcap recorded them. TRex sends and receives CG and non-CG packets via switch hardware ports to measure performance. Upon receiving traffic (CG or non-CG), the switch uses the source and destination IP addresses to identify the flow. We then compute the features and store them in the registers. The switch identifies flows using the DT method at the end of each time interval and sends packets to the designated hardware port. Receiving CG and non-CG packets on distinct ports is used for performance computation.

D. Performance Evaluation

F1-Score and Accuracy. We assess the accuracy of the suggested method using a simulator built on Python in order to compare MATADOR's accuracy. Using a simulator, we map the trained DT model and identify CG flows for each time window. The assessment is carried out for both CG and non-CG streams. Because of more precise computations on an x86 server to support floating-point numbers, the simulator performs slightly better in terms of accuracy and F1-score for both CG and non-CG scenarios, as expected and illustrated in Figure 4. Meanwhile, MATADOR outperforms on some occasions, too. This may be because, there aren't many differences in packet sizes or IPG, which lessens the influence of approximate calculations. As MATADOR represents the first attempt to be fully implemented in switch hardware, we compare it to the state-of-the-art, which is CG hybrid. Results are comparatively close (Figure 5(a)). MATADOR's overall performance on Tofino HW with *approximation computation* is encouraging, ranging from 92% to 99.99%.

We also evaluate our trained model using a new dataset (as mentioned in Sec. IV-A) obtained in a completely different lab configuration. Findings (Figure 6) verify that the suggested model can accurately (80-97%) categorize CG flows—a completely separate dataset from the one used to train it. Further

work is being done to optimize the depth of the ML model to increase overall accuracy further.

Detection Time. Detection times are improved when CG classification is offloaded to the Tofino HW. Three distinct time frame sizes—0.3, 0.5, and 1 second—are used to assess MATADOR. MATADOR computes features for every incoming packet and applies DT at the end of each time interval. According to the current implementation, as indicated in Figure 5(b), MATADOR can classify each flow in 0.03 seconds into CG or non-CG. When using a hybrid push or pull strategy, the controller reads all of the register values at regular intervals, or digest messages (a maximum of 48 bytes are permitted) are used to send feature calculations. Subsequently, the trained ML model is applied by the NFV node to classify flows and send the necessary entries to the switch. A few seconds are needed for the entire processing, as Figure 5(b) illustrates [14] [7].

E. Resource Utilization

We examine the resources for the DT and feature computations independently to assess the resource usage on the Tofino switch shown in Table I. Because DT uses six feature tables and one decision table with range match, TCAM is filled by about 12%. As mentioned in section III-B, DT occupies three stages—two for feature tables and one for decision tables. Seven stages and 4.7% of hash bits are needed for feature computation to identify the flows. MATADOR fits into ten stages in total, with two more stages for actions. Additionally, a maximum of 16 logical tables for both exact and ternary matches are supported by each match-action unit. 19% of the logical table id is consumed because several tables with actions are run to compute features and determine the uplink or downlink direction.

Table I: Resource usage of MATADOR.

Resource	DT	Features	CG (DT+Features)
Exact Match Result Bus	0.0	14.1	14.1
Exact Match Search Bus	0.0	8.9	9.4
Hash Bit	0.0	4.7	4.7
Logical Table Id	3.1	19.3	22.4
SRAM	0.5	3.4	4.1
TCAM	12.2	0.0	12.2
Stages	3	7	10

V. Conclusions and Future Work

This work investigates ML in a customizable data plane to improve CG traffic detection times and line-rate categorization. Given the restricted CPU capacity and resource limits of the switch hardware, it is essential to choose features that fit the overall design model.

In the future, we intend to create a multi-class classification that considers not only CG but also other application traffic kinds, like DoS traffic and virtual and augmented reality, leveraging efforts in related work [15]. Moreover, even though our proposed design can manage million flows, we will try to keep only the CG flows in the hash table to free up more resources for other tasks. We want to test our model in multiple circumstances, such as traffic prioritization to higher priority queues, to enhance the customer experience further through slicing architectures [1]. Furthermore, we plan to implement more ML algorithms on switch hardware, including RF and unsupervised learning, which require more calculations but potentially provide higher accuracy. Another track is augmenting the programmable pipeline with a suitable QoE estimation module as in [16].

Acknowledgment

This work was supported by Ericsson Telecomunicações Ltda., and by the Sao Paulo Research Foundation (FAPESP), FAPESP grant 2021/00199-8, CPE SMARTNESS.

References

[1] Galis Alex, Tusa Francesco, Clayman Stuart, Rothenberg Christian, and Serrat Joan. *Slicing 5G Networks: An Architectural Survey*, pages 1–41. John Wiley Sons, Ltd, 2020.

[2] Akem A. T. et al. Flowrest: Practical flow-level inference in programmable switches with random forests. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023.

[3] Bosshart et al. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.*, 44(3), July 2014.

[4] Graff P. et al. Efficient identification of cloud gaming traffic at the edge. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–10, 2023.

[5] J. Ky et al. A hybrid p4/nfv architecture for cloud gaming traffic detection with unsupervised ml. In *2023 IEEE Symposium on Computers and Communications*, pages 733–738, Los Alamitos, CA, USA, jul 2023. IEEE Computer Society.

[6] Kamath R. et al. Machine learning based flow classification in dcns using p4 switches. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–10, 2021.

[7] Kučera J. et al. Enabling event-triggered data plane monitoring. In *Proceedings of the Symposium on SDN Research*, SOSR '20, page 14–26, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Ridwan M. A. et al. Applications of machine learning in networking: A survey of current issues and future challenges. *IEEE Access*, 2021.

[9] S. K. Singh et al. HH-IPG: Leveraging Inter-Packet Gap Metrics in P4 Hardware for Heavy Hitter Detection. *IEEE Transactions on Network and Service Management*, 20(3):3536–3548, 2023.

[10] S. K. Singh. et al. Hybrid p4 programmable pipelines for 5g gnodeb and user plane functions. *IEEE Transactions on Mobile Computing*, 22(12):6921–6937, 2023.

[11] Siddique H. et al. Towards network-accelerated ml-based distributed computer vision systems. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 122–129, 2021.

[12] X. Marchal et al. An analysis of cloud gaming platforms behaviour under synthetic network constraints and real cellular networks conditions. In *JNSM*, pages 1573–7705. Springer, feb 2023.

[13] Xie Guorui et al. Empowering in-network classification in programmable switches by binary decision tree and knowledge distillation. *IEEE/ACM Transactions on Networking*, 32(1):382–395, 2024.

[14] Zhang D. et al. Hypertester: High-performance network testing driven by programmable switches. *IEEE/ACM Transactions on Networking*, 29(5):2005–2018, 2021.

[15] Alireza Shirmarz, Fábio Luciano Verdi, Suneet Kumar Singh, and Christian Esteve Rothenberg. From pixels to packets: Traffic classification of augmented reality and cloud gaming. In *IEEE NetSoft*, 2024.

[16] Francisco Vogt, Fabricio Rodríguez Cesen, Ariel Góes deCastro, Marcelo Caggiani Luizelli, Christian Esteve Rothenberg, and Gergely Pongrácz. Qoeyes: Towards virtual reality streaming qoe estimation entirely in the data plane. In *IEEE NetSoft*, 2023.

[17] 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*, HotNets '19, page 25–33, New York, NY, USA, 2019. Association for Computing Machinery.