This is the author's accepted version of the article. The final version published by IEEE is M. Gucciardo, B. Bütün, A.T-J. Akem and M. Fiore, "Evaluating the Impact of Flow Length on the Performance of In-Switch Inference Solutions,"IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1-6, doi: 10.1109/INFOCOMWKSHPS61880.2024.10620832.

# Evaluating the Impact of Flow Length on the Performance of In-Switch Inference Solutions

Michele Gucciardo*‡, Beyza Bütün*†‡, Aristide Tanyi-Jong Akem*† and Marco Fiore*
*IMDEA Networks Institute, Spain, †Universidad Carlos III de Madrid, Spain
{michele.gucciardo, beyza.butun, aristide.akem, marco.fiore}@imdea.org
‡*Equal contributors*

***Abstract*—As modern networks evolve into complex systems to support next-generation applications with strict latency requirements, in-switch machine learning (ML) has emerged as a candidate technology for minimizing ML inference latency. Multiple solutions, mostly based on Decision Tree (DT) and Random Forest (RF) models, have been proposed in that regard for inference at packet level (PL) or flow level (FL) or simultaneously at PL and FL. Such heterogeneity in the inference target leads to the use of varying performance metrics for evaluating the solutions, rendering a fair comparison between them difficult. In this paper, we perform a comprehensive evaluation of 5 leading solutions for DT/RF-based in-switch inference. We replicate and deploy the solutions into a real-world testbed with Intel Tofino switches, and run experiments with measurement data from 4 datasets. We then evaluate their performance using (*i*) the original metric used in the solution's paper, and (*ii*) a novel FL metric which evaluates every solution at FL. This FL metric enables us to delve into an extensive analysis of how the solutions perform on flows of different lengths in diverse use cases. Results show that while some solutions perform similarly across use cases and flow sizes, others show inconsistent behaviours that we discuss.**

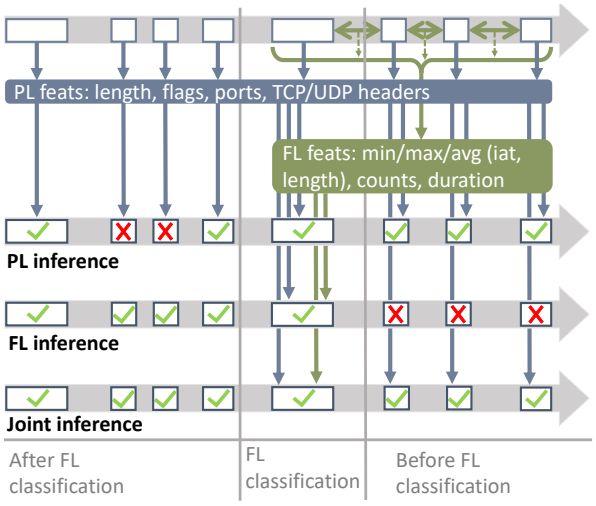



Figure 1: Exemplification of packet-level (PL), flow-level (FL) and joint inference performed by in-switch ML models.

## I. Introduction

Next-generation mobile and computer networks are evolving into complex systems that can support modern applications like self-driving cars, the digital twin and remote mobile health care, which all have very stringent latency requirements [1]. Guaranteeing low latency is thus a key requirement for modern network operation and management, and for the applications they support. In SDN networks, latency is minimized by implementing several applications within the network like anomaly detection and routing optimization in the control plane using ML [2]. However, these models do not run at line rate and would not support applications requiring sub-millisecond latency [3].

Over the past decade, off-the-shelf programmable data planes like Intel Tofino ASICs [4] have become available, alongside dedicated domain-specific programming languages like P4 [5]. This has led to a proliferation of in-network intelligent applications running entirely in the data plane [6], which can achieve sub-microsecond latency, several orders of magnitude less than that of control plane applications. In the case of ML, the strict constraints of programmable switches in terms of low available memory, limited support for mathematical operations, and limited number of allowed per-packet operations have made model training not feasible, and restricted applications to the deployment of trained models into programmable hardware [7].

In the face of the constraints highlighted above, Decision Tree (DT) and Random Rorest (RF) models are commonly used for in-switch ML applications due to their simplicity in terms of the operations and logic they involve [8]–[16]. A major dichotomy exists between these solutions, based on whether they infer at packet level (PL) *i.e.*, on individual packets [8]–[11], or at flow level (FL) *i.e.*, on traffic flows [12]–[14]. Recently, a third approach has also emerged, where the deployed model [15], [16] can infer on both packets and flows.

These approaches are illustrated in Figure 1. In PL inference solutions, the features used for model training and inference are packet header fields like the packet length, which are extracted from individual packets. The resulting models are very simple and do not always attain high accuracy in classification tasks, leading to errors on some packets, as exemplified by the $6^{\text{th}}$ and $7^{\text{th}}$ packets in PL inference row of Figure 1, counting from right (first packet) to left (last packet).

On the other hand, FL solutions employ techniques to compute, store and use stateful features (*e.g.*, inter-arrival times in Figure 1), which are computed over multiple packets of a flow. These features capture better the relationships between packets, and the resulting models grant higher accuracy, albeit at the cost of added complexity and resource usage for state management. Despite this improved accuracy, FL solutions delay classification until they have received enough packets to compute reliable FL features, as shown in Figure 1 where FL inference is delayed until the $4^{\text{th}}$ packet. This means that

FL solutions leave early packets unclassified: in the case of short flows consisting of only a few packets, the entire flow might be missed, unlike with PL solutions that would classify the individual packets of such short flows.

To remedy the low accuracy limitation of PL solutions and the missed early packets issue with FL solutions, hybrid solutions like NetBeacon [15] deploy multiple PL and FL models in the switch to classify either packets or flows. Yet, deploying separate models impacts the overall performance since the PL model is generally less performant than its FL counterpart and will bring down the overall score if many packets are classified at PL. Also, such multiple model deployment leads to substantial increases in the consumption of critical switch resources. Jewel [16] alleviates the resource consumption issue by training and deploying a single model that performs joint flow and packet classification as depicted at the bottom of Figure 1.

While all these in-switch solutions based on DTs and RFs have been evaluated on several use cases at PL or FL, the FL classification is more relevant in networking scenarios. This is apparent in that network policies, like those for Quality of Experience (QoE), are typically designed for groups of packets and not individual packets [17]. This raises the question of how effective PL solutions are in classifying flows, envisioning that FL policies can be designed based on the classification of individual packets. Under these premises, the length or size of the flow, *i.e.*, the number of packets it contains, could be a critical factor to consider when assessing the performances of these models.

Yet, the comparison of PL solutions to FL or hybrid and joint solutions in existing works is still biased by at least three factors: $(i)$ while the accuracy of PL solutions is naturally evaluated over packets, that of FL solutions is assessed by considering entire flows as the atomic unit; $(ii)$ FL solutions evaluations generally overlook the fact that early packets and flows with fewer packets than the minimum required to compute FL features are missed; $(iii)$ hybrid and joint solutions are evaluated over packets to take into account the performance of the PL part of the solution. As such, *no existing work considers an equitable evaluation of the existing models, using consistent metrics and also exploring the existence of dependencies of the accuracy on the length of the flow*.

In this paper, we evaluate five representative recent works that implement RF models into programmable switch hardware for inference on packets (Planter [9] and Mousika [18]), flows (Flowrest [14]), or both packets and flows (Netbeacon [15] and Jewel [16]). We exploit the artifacts provided by the various authors to reproduce each solution and run experiments in a real-world testbed with commercial Intel Tofino switches, where we compare their performances on various classification tasks, using two different metrics to shed light on their strengths and limitations in various use cases. Our results shed light on the differences between these solutions, exposing their strengths and weaknesses in different scenarios, and revealing under what conditions they are best suited for tackling in-switch inference tasks.

## II. Overview of Existing Solutions

There is a wide variety of existing works that implement DT and RF models in programmable switches for line rate inference as detailed in recent surveys [7], [19]. As we cannot evaluate all existing works, we focus on five works that we carefully selected to represent the state-of-the-art across PL, FL, and hybrid or joint inference solutions. Our selection criteria are $(i)$ feasibility in hardware and not just software or emulated environments, $(ii)$ availability of source code, and $(iii)$ generalizability, *i.e.*, the possibility to apply the solution to any classification task. The five selected solutions are detailed next.

**Planter** [9] builds upon IISy [8] by extending its DT implementation to include support for RFs. It was originally proposed as a general-purpose PL inference solution with a mapping scheme that encodes features into feature tables that assign codes that, when concatenated, map paths to tree leaves. These paths are summarized in code tables, one per tree, that assign a classification result to each packet. This scheme decouples the number of features or depth of the trees from the number of Match and Action (M/A) stages in the switch and is thus scalable in switch hardware. It is thus adopted by several other solutions like Henna [10] and in modified forms by NetBeacon [15]. We pick Planter to represent PL solutions and use the public source code [20] to replicate the solution.

**Mousika** [18] introduces a general-purpose PL classifier based on knowledge distillation. A teacher RF, neural network or DT is trained and then distilled into a single binary DT (BDT). While the BDT is mostly compact and memory efficient, the distillation can lead to a loss of classification accuracy. Other times, the BDT could grow too large that it becomes memory-hungry. The distillation is what differentiates Mousika from other PL solutions like Planter. We employ the source code [21] made available by the authors to reproduce the solution for all the use cases we consider for evaluation.

**Flowrest** [14] proposes a complete framework for deploying RF models in programmable switches for FL inference. It introduces a comprehensive flow management strategy for tracking and classifying flows fully in the data plane. While it brings substantial advances in the area of in-switch inference, it leaves several packets unclassified during the feature collection and computation phase. As such, it fails to assign a classification result to very short flows with insufficient packets to have an FL classification. To shed light on the impact of such missed packets, we pick Flowrest to represent FL solutions in our evaluation and use the available code [22] for replication.

**NetBeacon** [15] is the first hybrid solution to classify individual packets and flows by deploying multiple tree-based models fully into the switch. It deploys three sets of models. The first is an XGBoost model based on a DT, that serves as a PL flow-size predictor. Such a predictor identifies packets that belong to short flows for which no FL features will be computed or long flows for which memory is allocated and FL features are computed. The second is a PL classifier, which

classifies all the packets of short flows, those that suffer a hash collision, and those that arrive before an inference point, which is the number of packets at which an FL decision is made. Several inference points are used to classify a flow at different stages in its life. At each point, a DT is deployed to classify flows of that length. We replicate NetBeacon using the public code [23] provided by the authors.

**Jewel** [16] is the most recent RF-based in-switch inference solution. It enables joint PL-FL inference using a single model, unlike NetBeacon which uses multiple models. If FL classification occurs after the first `n` packets are observed, then the model is trained on the first `n` packets, with stateless PL features for the first `n-1` packets and FL features for the $n^{th}$ packet. During inference, the first `n-1` packets are classified using PL features and with constant values assigned to the FL features. At the $n^{th}$ packet, where FL features are deemed reliable, the model switches to FL inference, and an FL decision is made, which then applies to all subsequent packets of the flow. We use the released source code [24] for our evaluation.

## III. Experimental Evaluation

We perform our evaluation on a real-world experimental research platform with state-of-the-art equipment. We implement the five solutions described in Section II in the testbed and run tests with four use cases, collecting performance metrics that enable us to perform a comprehensive evaluation.

### A. Testbed setup

Our experimental testbed comprises 3 Edgecore Wedge 100BF-QS programmable switches, with Intel Tofino BFN-T10-032Q chipsets and 32 100GbE QSFP28 ports, and 2 off-the-shelf servers with Intel 8-core Xeon processors at 2GHz, 48GB of RAM, and QSFP28 interfaces. All components are connected via optical fiber links at 100 Gbps. The switches run the Open Network Linux (ONL) operating system, and the Intel Software Development Environment (SDE) version 9.7.0, for compiling P4 programs. The SDE is also equipped with the Intel P4 Insight tool, which provides details about compiled P4 programs, including their consumption of switch resources and the expected latency they will induce. We implement the control plane component of our setup in one of the servers alongside a traffic sink that captures traffic via Tcpdump for analysis after classification. The controller sets up each experiment by loading the corresponding model and its table entries into the switch, setting up forwarding ports and receiving packet digests containing classification results from the switch. The other server is a traffic source that injects traffic into the switch using Tcpreplay. On the other hand, it employs the MoonGen packet generator to inject background traffic at 100 Gbps, simultaneously to the use case traffic. This background traffic is not a target for inference, but it is leveraged to demonstrate that the tested models can achieve line-rate inference even in the presence of substantial concurrent non-target traffic.

| Dataset | #Flows | #Packets | Flow Length Max. | % Short flows [1] | % Packets in long flows | Very short flows | |
|---|---|---|---|---|---|---|---|
| | | | | | | Flowrest / NetBeacon | Jewel |
| ToN-IoT | 60200 | 1576856 | 999 | 92.82% | 70.31% | 9.97% | 6.64% |
| UNSW | 61170 | 1522271 | 158574 | 92.60% | 81.96% | 40.19% | 40.19% |
| UNIBS | 19657 | 1693245 | 345817 | 79.09% | 91.26% | 0.11% | 0.81% |
| IoT-23 | 60010 | 652529 | 239484 | 97.13% | 58.44% | 0.67% | 0.67% |

Table I: The statistics of the datasets and the proportion of very short flows, unclassifiable at FL because having less than `n` packets. For Flowrest/NetBeacon, `n` is 4 in ToN-IoT, 3 in UNIBS and UNSW, and 2 in IoT23. For Jewel, `n` is 4 in UNIBS, 3 in ToN-IoT and UNSW, and 2 in IoT23.

### B. Use cases and datasets

We select a variety of classification tasks based on measurement data that are available as public datasets, and which are employed in most of the solutions under evaluation. For each dataset, we use the same train and test data for all the solutions, following the model training process outlined in each paper.

**ToN-IoT [25]** is used to build a 10-class classification task to differentiate benign traffic and 9 cyberattacks. It is collected from a medium-scale testbed based on several virtual machines with different operating systems such as Windows, Linux, and Kali Linux. The measurements were obtained from seven IoT and industrial IoT (IIoT) devices including a thermostat, weather and Modbus sensors, a motion light, a GPS tracker, a fridge, a garage door, and some non-IoT devices such as a Smart TV and two iPhone 7. We use a ratio of $75-25$ to split the data into train and test sets.

**UNSW-IoT [26]** is made up of traffic flows generated by a wide variety of IoT and non-IoT devices *e.g.*, Dropcam, Amazon Echo, Netatmo Welcome, iHome, Laptop, HP Printer, *etc.* The data is provided as pcap files captured within a living lab at UNSW Sydney that emulates a smart environment. We design a 26-class device identification use case based on this data, and we use 15 days of data to train the RF models and test them on 1 day.

**UNIBS [27]** provides Internet traffic traces as pcap files. The dataset contains mail *e.g.*, POP3, IMAP4, and SMTP; web *e.g.*, HTTP and HTTPS; peer-to-peer applications *e.g.*, BitTorrent and Edonkey, and other protocols *e.g.*, FTP, SSH, and MSN. The traces comprise traffic generated by 20 workstations, which go through an edge router on the University of Brescia's campus network and are captured over 3 consecutive days. We design a service fingerprinting task with 8 different traffic classes. We then train RF models with the first day of traffic and test them with the second day.

**IoT-23 [28]** makes available an extensive malicious traffic dataset comprising real and labeled IoT malware infections and also benign IoT traffic. The dataset comprises 20 malicious and 3 benign captures, collected between 2018 and 2019. From these scenarios, 10 malicious bots generate benign and malicious traffic, while regular real IoT devices, *i.e.*, a Philips HUE smart LED lamp, an Amazon Echo, and a Somfy smart doorlock generate benign traffic. We build a bot classification use case with 14 classes, 4 benign and 10 malicious. We

[1] We consider flows with less than 20 packets as short; similar percentages can be obtained with flows of tens of packets.

extract a representative portion of the dataset and use a 75-25 ratio for the train-test split.

### C. Classification performance metrics

We evaluate the classification accuracy of the five solutions using the widely adopted F1 score (F1) metric. This metric can be computed out of three principal measures of a classification problem, *i.e.*, true positives (TP), false positives (FP), and false negatives (FN), as F1=2TP/(2TP+FP+FN). We average F1 in three different ways: ($i$) a *micro* average that is computed by summing up the TP, FP, and FN from all classes and then calculating the metric; ($ii$) a *macro* average that is the mean of individual class scores; and, ($iii$) a *weighted* average which weights individual class scores by the number of samples present in the dataset. Each average has a different meaning: micro, macro and weighted values quantify how accurate a model is in classifying individual packets, whole classes, and classes weighted by their support, respectively.

The measures TP, FP and FN are naturally calculated on a per-packet basis for Planter and Mousika. In Jewel and NetBeacon, the scores are calculated over packets as well. Indeed, the PL classification is used for the first `n-1` packets and the FL classification of the $n^{th}$ is extended to the following packets. The process is repeated in NetBeacon, as multiple FL classifications occur. Flowrest, has been evaluated at FL in the original paper, not considering the first `n-1` packets nor flows with less than `n` packets, and at PL in [16] to be compared with PL and hybrid or joint solutions.

Although PL evaluation ensures that all solutions are judged on the same number of samples, it fails to assess the capability of the solutions to classify flows. In fact, for a fixed use case, the number of test samples considerably changes from PL to FL, with the number of packets being at least one order of magnitude higher than that of flows, as shown in Table I. Besides, while short flows are the vast majority, long flows carry most of the packets. Indeed, Table I shows that, as an example, while the percentage of flows with less than 20 packets (say short flows) ranges from 79% to 97%, the portion of packets in flows with at least 20 packets (say long flows) ranges from 58% to 91%.

These statistics show that the overall performance at PL may be biased by the classification results of long flows only, making a PL score inappropriate to evaluate the capability of the solutions of classifying flows. We thus design a metric that operates on packets but removes that bias, by applying a normalization to the per-packet TP, FP, and FN. Specifically, we assign to each test packet a weight $w = \frac{1}{l}$, where $l$ is the length of the flow the packet belongs to. We will refer to this metric as the *flow-level* (or FL) metric in the following.

In addition, we also use a *native* metric for each solution, which is the metric adopted in the paper where the solution was originally introduced.

## IV. Experimental Results

Table II summarizes the native and FL F1-score of the solutions in the four use cases. For PL solutions, *i.e.*, Planter and Mousika, and hybrid/joint solutions, *i.e.*, NetBeacon and Jewel, the native score is calculated on a per-packet basis, as detailed in Section III-C, taking into account all the packets of the test sets. In contrast, the native model score of Flowrest is calculated on a per-flow basis, excluding the very short flows that could not be classified because of an insufficient number of packets. The portion of these flows ranges from 0.1% to 40.2% in the four use cases, as shown in Table I.

| Dataset | Metric | F1 score | Planter | Mousika | Flowrest | NetBeacon | Jewel |
|---|---|---|---|---|---|---|---|
| ToN-IoT | Native | Macro | 51.304% | 21.041% | **60.403%** | 50.347% | **54.571%** |
| | | Micro | 79.104% | 27.780% | **79.116%** | 78.172% | **85.347%** |
| | | Weighted | **81.079%** | 33.840% | 78.560% | 80.725% | **87.192%** |
| | Flow-level | Macro | 43.511% | 29.136% | **44.940%** | 42.573% | **49.055%** |
| | | Micro | 44.396% | 36.590% | **57.836%** | 45.351% | **65.014%** |
| | | Weighted | 42.375% | 36.293% | **57.688%** | 42.025% | **67.192%** |
| UNSW | Native | Macro | 67.844% | **77.030%** | 68.918% | 63.537% | **78.584%** |
| | | Micro | 85.834% | 84.649% | **85.864%** | 85.744% | **91.441%** |
| | | Weighted | 85.882% | 84.330% | **86.574%** | 86.502% | **91.592%** |
| | Flow-level | Macro | 49.016% | **64.921%** | 40.100% | 46.057% | **65.718%** |
| | | Micro | 71.192% | **83.568%** | 41.918% | 58.192% | **79.132%** |
| | | Weighted | 70.639% | **83.960%** | 40.614% | 59.278% | **79.776%** |
| UNIBS | Native | Macro | 90.316% | 87.834% | **96.676%** | 93.196 % | **98.023%** |
| | | Micro | 92.387% | 92.565% | **99.546%** | 95.571% | **98.528%** |
| | | Weighted | 91.978% | 90.653% | **99.551%** | 94.943% | **98.512%** |
| | Flow-level | Macro | 83.760% | 86.333% | **88.897%** | 88.072% | **95.978%** |
| | | Micro | 95.874% | **98.230%** | 88.269% | 96.912% | **99.538%** |
| | | Weighted | 96.395% | **98.452%** | 88.215% | 97.152% | **99.535%** |
| IoT-23 | Native | Macro | 79.300% | 74.557% | **93.040%** | 72.935% | **83.045%** |
| | | Micro | 92.587% | 92.136% | **99.393%** | 92.512% | **95.635%** |
| | | Weighted | 92.553% | 91.469% | **99.359%** | 92.780% | **95.261%** |
| | Flow-level | Macro | 82.926% | **87.460%** | 73.560% | 77.749% | **86.501%** |
| | | Micro | 98.588% | **99.243%** | 84.032% | 98.809% | **99.145%** |
| | | Weighted | 98.487% | **99.232%** | 83.616% | 98.742% | **99.095%** |

Table II: Performance of packet-level and flow-level solutions computed based on native model score and flow-level metric. The best value on each row is shown in **black bold** and the second best solution in **purple bold**.

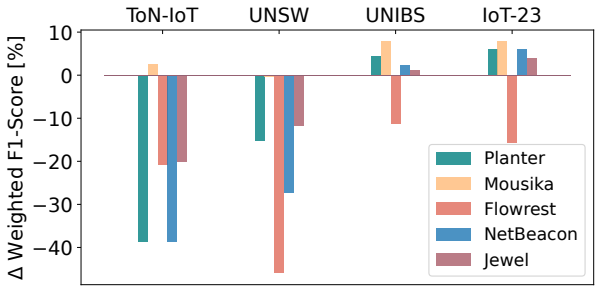
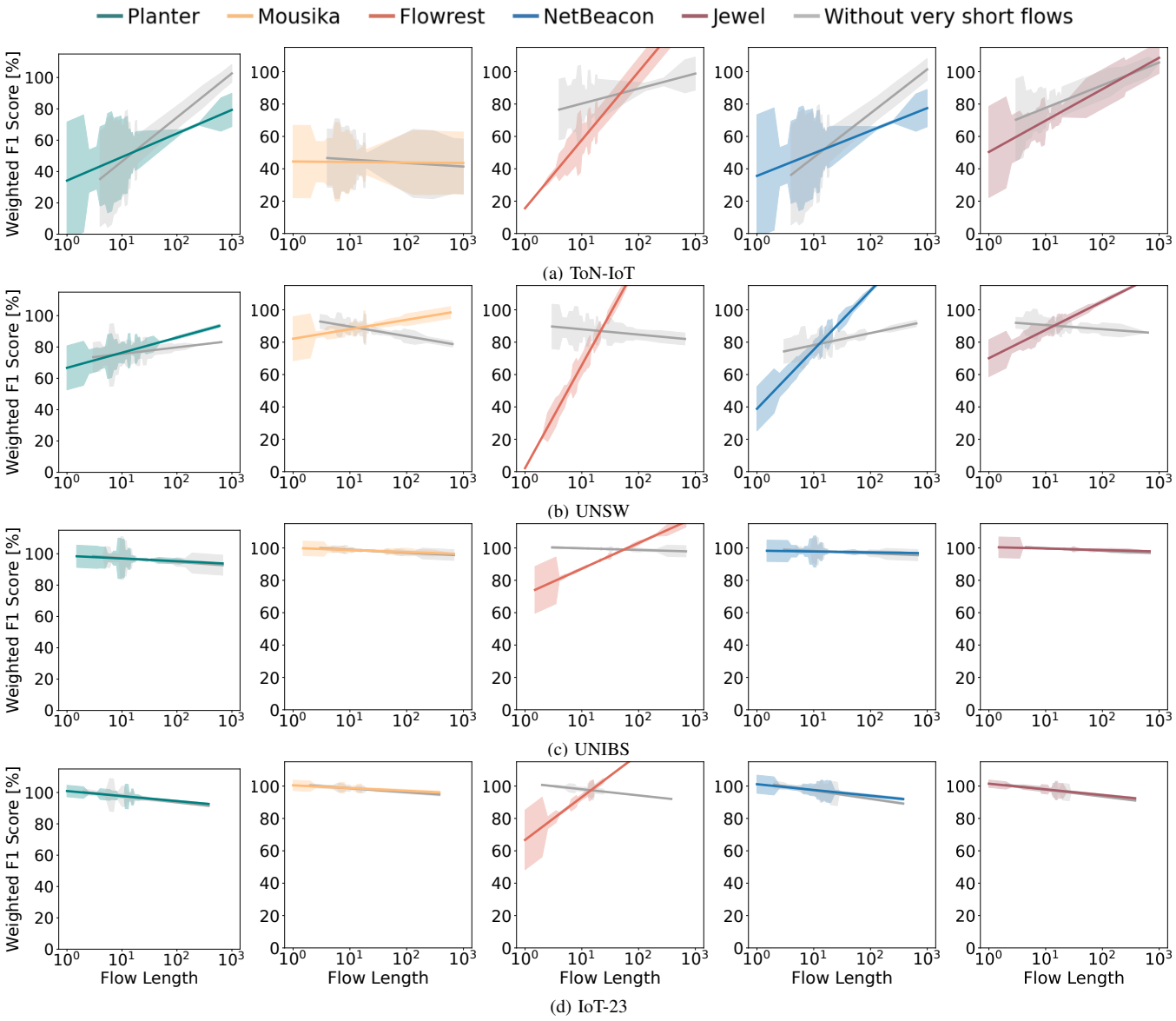



Figure 2: The difference in the weighted F1-score obtained with our flow-level metric and the solution's native metric.

The first observation is that Jewel is either the best or the second-best performing solution in all use cases and with both metrics. With the native metric, Jewel outperforms the competitors in ToN-IoT and UNSW, with the exception of the Macro score of ToN-IoT, where Flowrest is the best. In UNIBS and IoT23, instead, Flowrest outperforms all the competitors in all the F1 scores, except in the Macro score of UNIBS, where Jewel is the best.

With the FL metric, Jewel outperforms all the solutions in ToN-IoT and UNIBS in all the scores, with gains of up to 7.08%, 7.18%, and 9.50% in terms of Macro, Micro, and Weighted F1 score, respectively. In UNSW, Mousika is the best-performing solution in Micro and Weighted, and Jewel is the best in Macro. In IoT-23, Flowrest has the best scores.

Ultimately, these results validate the superiority of Jewel over existing solutions, as it is the most consistent over use cases, scores and metrics. Such a consistency is due to the




(a) ToN-IoT

(b) UNSW

(c) UNIBS

(d) IoT-23

Figure 3: Linear interpolation curves, showing the performance of 5 different solutions on classifying flows with varying lengths by both including and excluding short flows and the first `n-1` packets of flows in terms of weighted F1 score, plotted on a logarithmic scale and semi-transparent regions surrounding each line that illustrate the deviation in scores. The results with exclusion are shown in gray.

fact that Jewel offers the best of both worlds, *i.e.*, PL and FL, achieving better performances than PL-only and FL-only solutions. In comparison with NetBeacon, the other solution leveraging both PL and FL approaches, we observe a very close performance in UNIBS and IoT-23, but a clear advantage up to 19.66%, 20.94%, and 25.17% in Micro, Macro and Weighted F1 in UNSW and ToN-IoT. One possible explanation for such a gap is that NetBeacon employs DTs, which hardly attain the same performance as Jewel's RF models.

Figure 2 shows how the unifying perspective provided by the FL metric changes the performance of the solutions with respect to using their native score. A negative or a positive value express a drop or an increase over the native score, respectively. We show the change in the weighted F1 score, but the results are similar under the other two scores.

In ToN-IoT, all the solutions experience an absolute loss in the ranges of $20.00\% - 38.70\%$, except Mousika, which shows an absolute gain of $2.45\%$. All the solutions show a lower performance when the weights are introduced in UNSW, with the difference in the range of $0.37\% - 45.96\%$. In UNIBS and IoT-23, all the solutions experience a score improvement in the range of $1.02\% - 7.80\%$ and $3.83\% - 7.76\%$, respectively, except Flowrest that drops by $11.34\%$ and $15.74\%$.

To explain why the score drops in the FL metric, it is important to consider that packets belonging to short flows have more weight because of the normalization with the flow length. At FL, a misclassification of one packet in flows of length 2 will count way more than that in flows of length

100. In the former case, it will correspond to missing half of the flow, in the latter to missing just $1\%$ of the flow. Not only is this intuitive but it is also corroborated by the fact that short flows are the vast majority, as described in Section III-C. We conclude that the score drops when short flows are poorly classified. Such an effect could be especially severe for Flowrest, as the very short flows are oblivious to the solution and are considered wrong classification (same as the first `n-1` packets of each flow).

Figure 3 shows how the FL metric performance varies with the length of the flow. For simplicity, instead of the real maximum flow lengths (shown in Table I), we considered $1,000$ as the maximum flow length, which accounts for $99\%$ of the flows in all the datasets. Since the majority of the flows have less than 20 packets in all the use cases, the curve is plotted mostly depending on the scores of these flows, which causes a trend where we see values greater than $100\%$. The colored lines show the trend considering the full dataset; the curves in gray, instead, exclude flows shorter than `n` and the first `n-1` packets of all other flows.

In ToN-IoT and UNSW, shown in Figures 3a and 3b, the lines grow from left to right, meaning that in all cases the solutions are performing worse over short than over long flows. Such a behavior automatically translates into a score drop, as shown in Table II. The only exception is Mousika in ToN-IoT, which does not loose score but yet shows a poor general performance in this use case. Especially pronounced is the drop of Flowrest, confirmed by the fact that in ToN-IoT and UNSW the very short flows, completely oblivious, account for $9\%$ and $40\%$, respectively.

In UNIBS and IoT-23, shown in Figures 3c and 3d, almost all the solutions are doing well with the short flows and we can see in Table II that their scores increase. The only exception is Flowrest, which shows a slight drop because of the penalty due to the very short flows, which cannot be classified.

The gray lines, which show the trends when excluding the very short flows, generally improve the performance at FL when considering all the flows. In particular, Flowrest stands up in this case as virtually the best solution. Such a result is unsurprising as we think that the FL metric without the very short flows is also Flowrest's native metric.

## V. Conclusions

We performed an extensive evaluation of leading state-of-the-art in-switch ML solutions based on DT and RF models, running at flow level and/or packet level. We conducted multiple experiments on a real-world testbed comprising production-grade hardware, using measurement data from various use cases. We then derived a new metric that enables an equitable assessment of all solutions in terms of their performance when their results are used to derive scores at the flow level, which is typically used to derive network policies. Results shed light on the peculiarities of each solution, showing that Jewel is the most consistent of all the solutions, taking the best out of PL and FL classification. Flowrest, instead, stands up as virtually the best-performing solution when excluding the impact of the very short flows.

## Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 101017109 "DAEMON", the CHIST-ERA grant no. CHIST-ERA-20-SICT-001 "ECOMOME", via grant PCI2022-133013 of Agencia Estatal de Investigación, from the Spanish Ministry of Science and Innovation through grant no. PID2021-128250NB-I00 "bRAIN", and from the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D project no.TSI-063000-2021-52 "AEON-ZERO".

## References

[1] R. Bai et al. Next generation mobile wireless networks: 5G cellular infrastructure. *J. Technol. Manag. Appl. Eng.*, 36, 7 2020.
[2] J. Xie et al. A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Commun. Surv. Tutor.*, 21(1):393–430, 2019.
[3] K. He et al. Measuring control plane latency in sdn-enabled switches. SOSR '15, NY, USA, 2015. ACM.
[4] Tofino Programmable Ethernet Switch ASIC. https://shorturl.at/bqs47.
[5] P. Bosshart et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, jul 2014.
[6] E. F. Kfoury et al. An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. *IEEE Access*, 9:87094–87155, 2021.
[7] R. Parizotto et al. Offloading machine learning to programmable data planes: A systematic survey. *ACM Comput. Surv.*, jun 2023.
[8] Z. Xiong and N. Zilberman. Do switches dream of machine learning? toward in-network classification. In *HotNets 2019*. ACM, 2019.
[9] C. Zheng and N. Zilberman. Planter: Seeding trees within switches. In *SIGCOMM '21*, pp. 12–14, NY, USA, 2021. ACM.
[10] A. T.-J. Akem et al. Henna: Hierarchical machine learning inference in programmable switches. In *NativeNI 22*, pp. 1–7. ACM, 2022.
[11] B. Bütün et al. Fast detection of cyberattacks on the metaverse through user-plane inference. In *IEEE MetaCom*, pp. 350–354, 2023.
[12] C. Busse-Grawitz et al. pForest: In-network inference with random forests. *CoRR*, abs/1909.05680, 2019.
[13] B. Coelho et al. BACKORDERS: Using random forests to detect DDoS attacks in programmable data planes. In *EuroP4 '22*, 2022.
[14] A. T.-J. Akem et al. Flowrest: Practical flow-level inference in programmable switches with random forests. In *IEEE INFOCOM*, 2023.
[15] G. Zhou et al. An efficient design of intelligent network data plane. In *32nd USENIX symposium on security*, 2023.
[16] A. T.-J. Akem et al. Jewel: Resource-efficient joint packet and flow level inference in programmable switches. In *IEEE INFOCOM*, 2024.
[17] S. C. Madanapalli et al. Reclive: Real-time classification and QoE inference of live video streaming services. In *IEEE/ACM IWQOS*, 2021.
[18] G. Xie et al. Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation. In *INFOCOM*, 2022.
[19] C. Zheng et al. In-network machine learning using programmable network devices: A survey. *IEEE Commun. Surv. Tutor.*, pp. 1–1, 2023.
[20] A.T-J. Akem et al. Henna. https://github.com/nds-group/Henna.
[21] G. Xie et al. Mousika. https://github.com/xgr19/Mousika.
[22] A.T-J. Akem et al. Flowrest. https://github.com/nds-group/Flowrest.
[23] G. Zhou et al. Netbeacon. https://github.com/IDP-code/NetBeacon.
[24] A.T-J. Akem et al. Jewel. https://github.com/nds-group/Jewel.
[25] A. Alsaedi et al. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access*, 8, 2020.
[26] A. Sivanathan et al. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans Mob Comput*, 18(8), 2019.
[27] M. Dusi et al. Detection of encrypted tunnels across network boundaries. *2008 IEEE ICC*, pp. 1738–1744, 2008.
[28] S. Garcia et al. IoT-23: A labeled dataset with malicious and benign IoT network traffic. Zenodo, 2020.