This is the author's accepted version of the article. The final version published by IEEE is B. Bütün, D.A. Hernandez, J.M.Aguilar Polo, M. Gucciardo and M. Fiore, "Demonstrating Distributed Inference in the User Plane with DUNE," IEEE INFOCOM 2025 Demo Session - IEEE International Conference on Computer Communications, 2025, doi: TBD.

Demonstrating Distributed Inference in the User Plane with DUNE

Beyza Bütün*†, David de Andres Hernandez*†, Josue Miguel Aguilar Polo*, Michele Gucciardo‡, Marco Fiore*
*IMDEA Networks Institute, Spain, †Universidad Carlos III de Madrid, Spain, ‡NEC Laboratories Europe, Spain
*{beyza.butun, david.deandres, josue.aguilar, marco.fiore}@imdea.org, ‡michele.gucciardo@neclab.eu

***Abstract*—Deploying Machine Learning (ML) models in the user plane enables low-latency and scalable in-network inference, but integrating them into programmable devices faces stringent constraints in terms of memory resources and computing capabilities. In this demo, we show how the newly proposed DUNE, a novel framework for distributed user-plane inference across multiple programmable network devices by automating the decomposition of large ML models into smaller sub-models, mitigates the limitations of traditional monolithic ML designs. We run experiments on a testbed with Intel Tofino switches using measurement data and show how DUNE not only improves the accuracy that the traditional single-device monolithic approach gets but also maintains a comparable per-switch latency.**

I. Background and motivation

User-plane programmability is revolutionizing traditional network functions like telemetry, load balancing, caching, and intrusion detection by fostering innovation and improving efficiency. Notably, programmable network hardware has made it possible to deploy Machine Learning (ML) models in the user plane for in-network inference. ML models, trained offline, can be integrated into programmable switches or smart Network Interface Cards (smartNICs), allowing packet-level processing at line rate with ultra-low latency.

This approach significantly improves delays, scalability, and cost-efficiency compared to traditional control-plane ML, which involves cross-plane interactions and additional hardware overhead. Yet, embedding ML models into programmable devices remains challenging due to limited compute capabilities, memory constraints, and internal architectures not accounting for ML operations.

To address these challenges, prior solutions have focused on tailoring ML models for single devices, such as switches or smartNICs, by leveraging techniques like Decision Trees (DTs), Random Forests (RFs), or Neural Networks (NNs). These models typically operate at the packet-level [1], flow-level [2], or both [3], [4], but their implementation has remained monolithic, confined to a single piece of network hardware. While effective in certain scenarios, this approach underutilizes the inherently distributed nature of modern networks, which consist of multiple switches and middleboxes, and limits the classification accuracy achieved. A more flexible and scalable strategy is distributing ML inference tasks across multiple devices, allowing packets and flows to be processed collaboratively as they traverse the network. By decomposing ML models into smaller sub-models, different devices can perform portions of the inference task, leading to more efficient resource usage and enhanced scalability.

II. Proposed solution

DUNE [5] addresses the shortcomings of traditional monolithic solutions by proposing a novel framework for executing distributed user plane inference in real-world programmable hardware. The fully automated framework breaks down large ML models, trained for complex inference tasks, into simpler sub-models that (i) are compatible with the programmable network constraints and (ii) jointly execute different portions of the desired inference task. Through original optimization, the framework aims to preserve accuracy while minimizing sub-model complexity. Moreover, it supports any class of input ML model, and enables joint packet- and flow-level inference.

The design of DUNE is automated with the following steps: (i) Labeled historical measurement data is used to create a highly accurate monolithic ML model for the target traffic analysis task, independent of network hardware constraints. (ii) The monolithic ML model is analyzed to generate an importance matrix $\boldsymbol{W}$ that expresses the relevance of features in explaining output variables and an accuracy vector $\boldsymbol{f}$ of the inference quality for each class. (iii) The information in $\boldsymbol{W}$ and $\boldsymbol{f}$ is used to partition the original inference task into sub-tasks, each focusing on a subset of input features and corresponding output variables. DUNE optimizes this partitioning through a custom heuristic that groups variables explainable by compact feature sets. (iv) Dedicated ML sub-models are trained for each sub-task using its input features, ensuring compatibility with user-plane hardware constraints. (v) The sub-models are executed sequentially to perform the original inference task, with their order optimized based on their reciprocal accuracy to minimize misclassifications and maximize overall inference performance, as errors can propagate through the chain.

Once the ordered ML sub-models are prepared, they are deployed as P4 programs on programmable user-plane devices. As shown in Figure 1 in three distinct blocks, each switch performs three key operations: *inference-aware forwarding* for traffic filtering, *flow management* for storing stateful information on target flows, and *inference* for classifying packets using packet-level features and incorporating flow-level features once sufficient packets are observed. Each sub-model classifies a distinct portion of the traffic. As illustrated in Figure 1, the flows classified by the first sub-model (green) are not reclassified by subsequent sub-models, while unclassified flows by the first sub-model (grey) are classified by the next one (red). Upon classification, the control plane updates forwarding rules and release flow management resources. When sub-models are simple, multiple models can be deployed on the same switch.

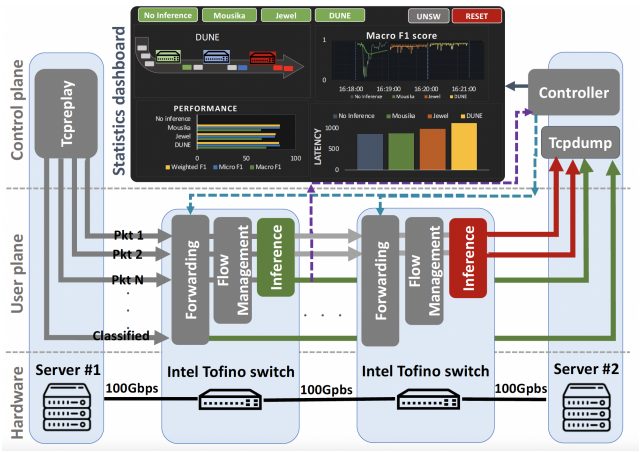

Figure 1: Demo setup, and mapping of the logical components of the control and user planes into the testbed hardware. Two switches are shown for ease, but there are three in the demo.

III. Demonstration

This demo demonstrates how distributed inference executed by DUNE outperforms existing traditional monolithic solutions.

A. Demonstration setup

Figure 1 shows an overview of the components of the demo which are mapped into five hardware devices *i.e.*, two servers and three programmable switches, shown as two in the diagram for the sake of simplicity, linked via QSFP28 interfaces at 100 Gbps, depicted at the bottom of the figure.

The two off-the-shelf servers have AMD EPYC 24-core processors at 2.8GHz and 128GB of RAM. Server #1, hosting the traffic source, replays captured pcap traces using Tcpreplay to inject target traffic for inference (shown in gray) into the switch. The traffic from the source is forwarded through Edgecore switches equipped with an Intel Tofino BFN-T10-032Q chipset, where each runs a sub-model of DUNE for distributed line rate inference. Each switch hosts all the important modules of the DUNE solution, the inference-aware forwarding block, the flow management block, and the inference module. Finally, server #2, serving as the traffic sink, captures outgoing traffic from the switch with a dedicated Tcpdump instance. It generates a pcap file for PL solutions, which is then analyzed by the controller.

The controller, running on server #2, has multiple roles. First, for each experiment, it sets up the switch by loading the corresponding model's table entries. Second, it updates the forwarding table right after the classification of a flow. Third, after each experiment, it analyzes the captured pcap traces and the received packet digests to calculate the classification performance of the benchmarks and DUNE in terms of macro, weighted, and micro F1-score. It also gets latency information from an instance of the Intel P4Insight tool it runs. Finally, the controller feeds the statistics dashboard with data for on-screen display, as shown in Figure 1.

The performance of DUNE and benchmarks are displayed in a dashboard fed by the controller. The dashboard consists of two main areas, as depicted in Figure 1. We show the topology of the displayed solution and live performance results in terms of macro F1 score during each evaluation in the first area (top) and then compare the overall performance results and the total latency of each solution in the second area (bottom). The dashboard displays the results of a solution after selecting the solution and use case by clicking the corresponding buttons.

	Performance			Latency			
Dataset	Mousika	Jewel	Dune	No inference	Mousika	Jewel	Dune
UNSW	64.921%	65.718%	70.263%	852.54	871.64	981.47	1137.70
ToN-IoT	47.099%	61.718%	67.541%	852.54	869.18	1015.08	1183.61

Table I: Performance of DUNE and the benchmarks in terms of macro F1-score and based on *flow-level* metric, and latency (ns) across three switches.

B. Experiments

We use the 2 datasets considered in [5], namely ToN-IoT (attack classification) and UNSW (device identification), for evaluation. We train models for DUNE and two benchmarks, *i.e.*, Mousika [1] and Jewel [4] and implement them in P4. We then run 60-second experiments per use case and solution to evaluate classification performance and end-to-end latency. For a fair comparison, the reported latency accounts for the three switches, reflecting DUNE's distributed architecture. For monolithic models, the first switch incurs both forwarding and inference latency, while subsequent switches only experience forwarding latency. In DUNE, all switches combine the latency of forwarding and inference.

We first run legacy forwarding just to calculate the total latency across three switches, which is 852 ns, as shown in Table I. As no inference is involved, performance metrics are not displayed on the dashboard. Next, we sequentially run Mousika, Jewel, and DUNE. The results show that DUNE outperforms all the benchmarks with the gain of 4.5% and 5.8% macro F1 score compared to the second-best model in UNSW and ToN-IoT, respectively, as shown in Table I. DUNE improves classification performance and enables packet- and flow-level ML inference on production-grade hardware, adding only in average 308 ns, 290 ns, and 162 ns of latency across three switches compared to no inference, Mousika, and Jewel, respectively. The total inference latency remains approximately 1μs across three switches, which is negligible.

Acknowledgments

This research was supported by ORIGAMI project (GA 101139270) funded by SNS JU and the European Union. M. Fiore is Talent Attraction fellow (2023-5A/TIC-28944) and B. Bütün predoctoral fellow (PIPF-2022/COM-24867), both being co-financed by Comunidad de Madrid.

References

[1] G. Xie et al., "Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation," in *INFOCOM*, 2022.
[2] A.T-J. Akem et al., "Flowrest: Practical flow-level inference in programmable switches with random forests," in *INFOCOM*, 2023.
[3] G. Zhou et al., "An efficient design of intelligent network data plane," in *32nd USENIX symposium on security*, 2023.
[4] A.T-J. Akem et al., "Jewel: Resource-efficient joint packet and flow level inference in programmable switches," in *INFOCOM*, 2024.
[5] B. Bütün et al., "Dune: Distributed inference in the user plane," in *INFOCOM*, 2025. [Online]. Available: hdl.handle.net/20.500.12761/1883