SpliDT: Partitioned Decision Trees for Scalable Stateful Inference at Line Rate

Murayyiam Parvez•, Annus Zulfiqar[1•], Roman Beltiukov[2•], Shir Landau Feibish[3], Walter Willinger[4], Arpit Gupta[2], Muhammad Shahbaz[1]

Purdue University [1]*University of Michigan* [2]*UCSB* [3]*The Open University of Israel* [4]*NIKSUN Inc.* —•*Student*

1 MOTIVATION & GOALS

Machine learning is increasingly being deployed in programmable network switches to enable real-time traffic analysis and security monitoring. Decision trees (DTs) offer a powerful approach for these tasks due to their interpretability and transparency. However, existing DT implementations in switches face a critical limitation: they require collecting all features before making a decision. This constraint forces models to use a small, fixed set of features per flow, limiting accuracy and scalability.

This paper introduces SpliDT, a scalable framework that removes this feature constraint through a partitioned inference architecture (Figure 1). Instead of requiring the same fixed features for all decisions, SpliDT assigns different sets of features to different parts of the tree and dynamically selects them as needed. To efficiently manage resources, SpliDT leverages recirculation to reuse registers and match keys at line rate. These capabilities are enabled by two core innovations: (1) a mechanism for tracking and managing inference across partitions using Sub-Tree IDs (SIDs) and (2) a custom training framework using HyperMapper and Bayesian Optimization to optimize decision tree structure and feature allocation. Our evaluation shows that SpliDT achieves higher accuracy while accommodating up to 5× more stateful features and scaling to millions of flows, significantly outperforming state-of-the-art approaches like NetBeacon and Leo. Despite this increased capacity, SpliDT maintains low recirculation overhead (≤ 50 Mbps) and low time-to-detection (TTD), demonstrating that ML models can operate efficiently within the constraints of programmable switches.

• ***Challenges in Scaling Stateful Features.*** Existing approaches such as NetBeacon [5] and Leo [4] strictly limit the number of stateful features to a small, fixed set (e.g., the top-k most important features), achieving higher accuracy than models limited to per-packet features but at the cost of added stateful memory overhead. First, stateful features are stored in registers, which share limited space with match-action tables (MATs) in each pipeline stage—creating a trade-off between feature storage and model complexity.

Corresponding author: Murayyiam Parvez (parvezm@purdue.edu)

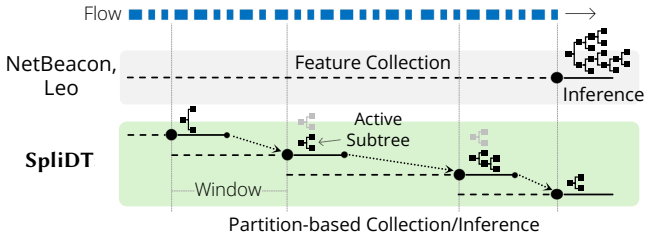

Figure 1: Comparison of in-network DT classification. Prior work (top) performs one-shot inference over full flows. SpliDT (bottom) collects features and infers incrementally across partitions, scaling to more stateful features and achieving higher F1 scores at line rate.

Increasing the number of registers or supporting more flows further reduces available MAT stages, limiting DT depth and restricting feature selection. Second, increasing the number of stateful features also expands match key sizes, inflates table entries, and exacerbates TCAM usage, making it harder to map DTs onto the MATs [3–5].

Insight 1: Traditional designs [4, 5] face a three-way trade-off between feature richness, scalability (flows), and model complexity. We demonstrate that by decoupling stateful feature selection from DT execution, feature richness and scalability can grow independently without sacrificing model complexity. SpliDT achieves this by dynamically selecting and reusing stateful features across inference steps, efficiently managing limited hardware resources.

• ***Domain-Specific Properties of DTs.*** DT inference begins at the root node and proceeds level by level, making decisions based on selected features until reaching a leaf. Instead of evaluating the tree sequentially, we can group consecutive levels into *partitions* and process one partition at a time. Within each partition, inference traverses the *active* subtree, whose outcome determines the next subtree to evaluate.

Insight 2: This subtree-by-subtree execution enables features to be collected incrementally and on demand, as inference progresses one partition at a time—where decisions from the active subtree in one partition determine the next subtree to traverse. With just k available feature slots, we can dynamically load only the relevant features for the current subtree, avoiding the rigid top-k selection imposed by prior systems [4, 5].

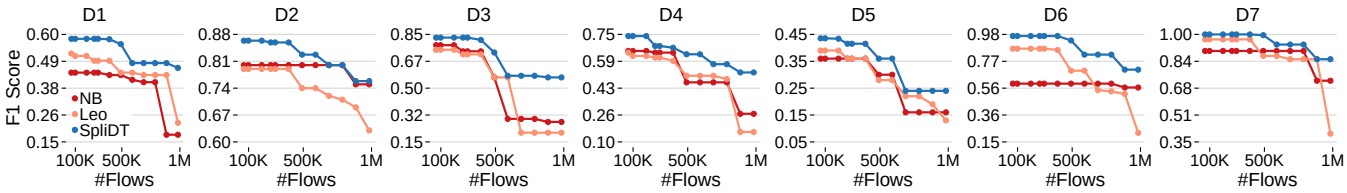

Figure 2: Pareto frontier of SpliDT vs. baselines, indicating the best F1 score for a given #flows in the data plane.

• ***Switch as a Time-Shared Resource.*** Programmable switches are typically seen as spatial architectures with fixed resource limits, where exceeding these constraints causes compilation failures [1–3]. However, this static view overlooks dynamic capabilities like packet recirculation, supported in modern switches (e.g., Tofino1 at 100 Gbps [3]) without impacting line rate.

Insight 3: Recirculation enables temporal execution, allowing different program stages (e.g., in P4 [1]) to run across multiple passes and reuse limited resources such as registers and match keys. By restructuring DTs in P4, we exploit this behavior to scale inference beyond spatial limits—similar to how CPUs reuse registers over time. We observe that the recirculation usage stays within 20 Mbps in different datacenter settings, well below the available 100 Gbps bandwidth.

2 SPLIDT DESIGN

We now show how SpliDT leverages the domain-specific properties of DTs and resource reuse through switch recirculation to optimize the F1 score–flow scalability tradeoff, enabling full stateful feature support at line rate.

• ***Partitioned Inference Architecture.*** SpliDT's partitioned inference architecture operates in two phases: (1) feature collection and engineering, and (2) subtree model prediction—processing flow windows iteratively within each DT partition by reusing resources such as registers and match keys. SpliDT maintains dedicated registers to track subtree ID (SID), compute intermediate states, and store active stateful features. Upon packet arrival, it hashes the 5-tuple to index the correct register set and supports multi-stage dependency chains for hierarchical feature computations. Operator selection is handled through match-action tables (MATs), which dynamically apply the appropriate computation needed for each feature in the current subtree, based on the SID. To avoid unnecessary updates, feature computations can be triggered conditionally (e.g., on SYN packets). During prediction, encoded feature values and subtree IDs are matched using range-marked MATs to determine the next subtree for continued inference unless a flow is classified.

• ***Custom Search/Training Framework.*** SpliDT employs Bayesian Optimization to explore Pareto-optimal decision tree configurations that balance accuracy, flow scalability, and hardware constraints. The search space includes parameters such as tree depth, number of features per partition, and partition sizes, with each iteration guided by feedback on model performance and feasibility. A custom training algorithm recursively builds partitioned trees, specializing each subtree using only the flow windows that reach it during training. For every candidate, the model is evaluated for accuracy, supported flows, and switch resource usage. Using the Range Marking algorithm [5], SpliDT generates TCAM rules for both feature encoding and model logic, installing them per subtree, and determines feasibility through hardware resource analysis.

3 PRELIMINARY RESULTS

We evaluate SpliDT on seven real-world datasets (D1–D7) spanning security applications such as intrusion detection, traffic classification, and detection of attacks like DoS, botnets, and infiltration. Across all datasets, SpliDT consistently outperforms baselines by achieving higher accuracy for the same number of flows. By maintaining a better balance between model performance (i.e., accuracy) and scalability (i.e., flows), SpliDT defines the Pareto frontier (Figure 2). The tradeoff curves are monotonically decreasing: models attain higher accuracy with fewer flows and gradually sacrifice model size and feature coverage to scale. Compared to state-of-the-art methods [4, 5], SpliDT delivers better accuracy while supporting up to 5× more stateful features and scaling to 1 million flows, all while keeping recirculation overhead $\leq$ 50Mbps and time-to-detection low.

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. In *ACM SIGCOMM CCR.*

[2] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. In *ACM SIGCOMM.*

[3] Intel. last accessed: 12/31/2024. Tofino: P4-programmable Ethernet switch ASIC that delivers better performance at lower power. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html.

[4] Syed Usman Jafri, Sanjay Rao, Vishal Shrivastav, and Mohit Tawarmalani. 2024. Leo: Online ML-based Traffic Classification at Multi-Terabit Line Rate. In *USENIX NSDI.*

[5] Guangmeng Zhou, Zhuotao Liu, Chuanpu Fu, Qi Li, and Ke Xu. 2023. An Efficient Design of Intelligent Network Data Plane. In *USENIX Security Symposium.*