

INTRODUCTION

Machine learning is increasingly being deployed in RMT switches to enable real-time, data-driven decision-making directly in the data plane

Decision trees (DTs) offer a powerful approach for these tasks due to their interpretability and transparency. They are simple, interpretable and verifiable.

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.

We present **SpliDT: Partitioned Decision Trees for Scalable Stateful ML Inference at Line Rate**, designed to support 5x more features than SOTA and scale effectively across millions of flows.

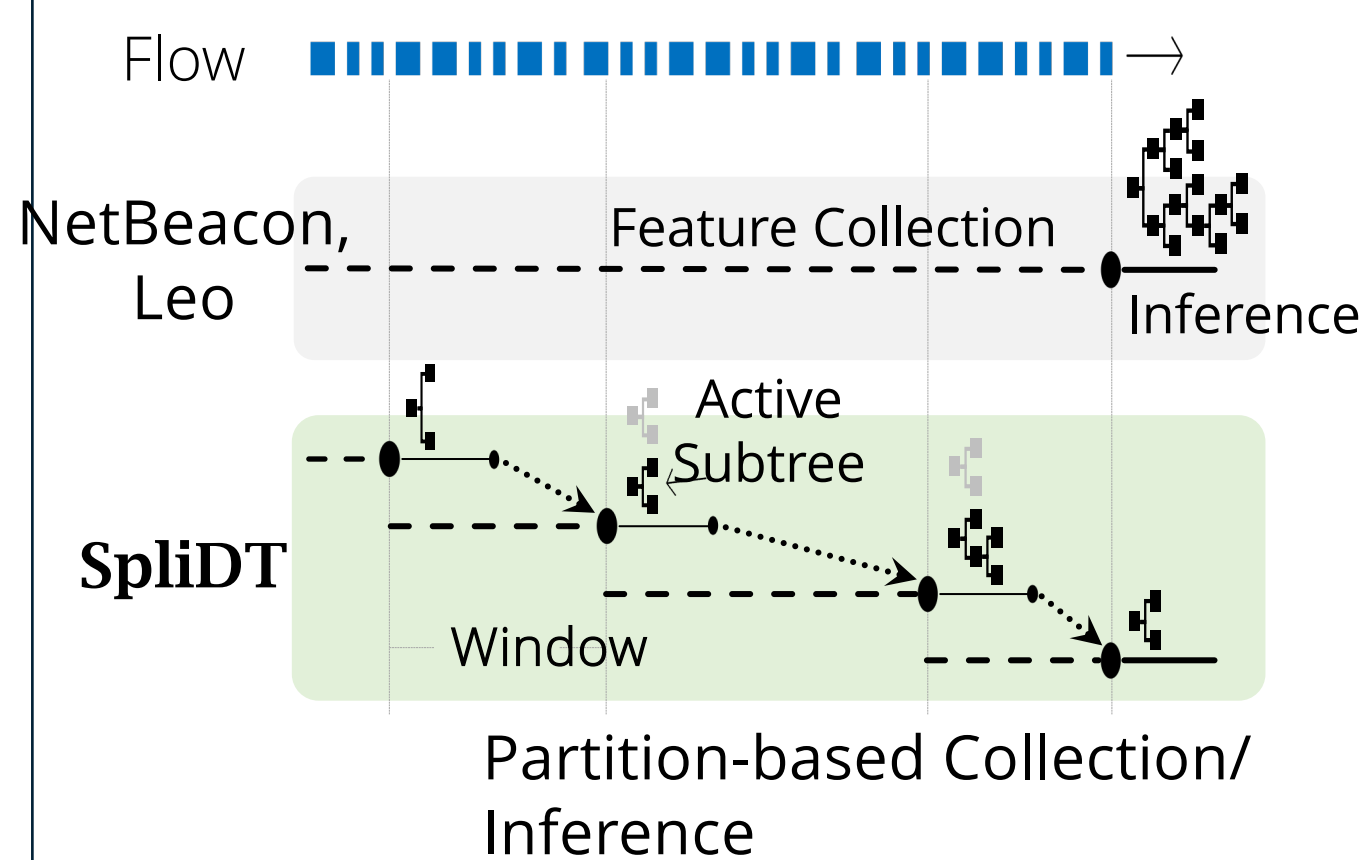
Instead of requiring the same fixed features for all decisions, SpliDT assigns different sets of features to different parts of the tree and dynamically selects them as needed.

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 Sub-Tree ID (SID) tracking mechanism for managing inference across partitions
- (2) a custom training framework using HyperMapper and Bayesian Optimization to optimize decision tree structure and feature allocation.

CURRENT METHOD/STATUS



Systems like IISY and Mousika using only per-packet features — but that loses flow context.

Other systems, like Leo and NetBeacon, added flow features. But with limited switch state, they can only store a few — the so-called top-k features, chosen by hardware, not by the model. That restricts both features and accuracy.

RELEVANCY

SpliDT significantly increases the total number of stateful features that a DT can utilize over its entire execution. Unlike prior approaches that apply a fixed top-k feature set across the entire DT, SpliDT dynamically assigns each subtree—resulting from tree partitioning—its own relevant feature set, allowing features to vary between subtrees.

We demonstrate that SpliDT supports an order of magnitude more stateful features (i.e. the total number of unique features across all subtrees) than state-of-the-art approaches, while maintaining efficient execution in the data plane.

This approach improves accuracy by enabling more richer in-network inference.

METHOD

SpliDT framework first uses the **subtree ID** as match key to **collect** the top-k **features** for each subtree, **dynamically updating** them at runtime as the active subtree changes. It uses the **subtree inference** to select the **next subtree** until a label is found, using **recirculation** to reset state and activate the next subtree in the switch.

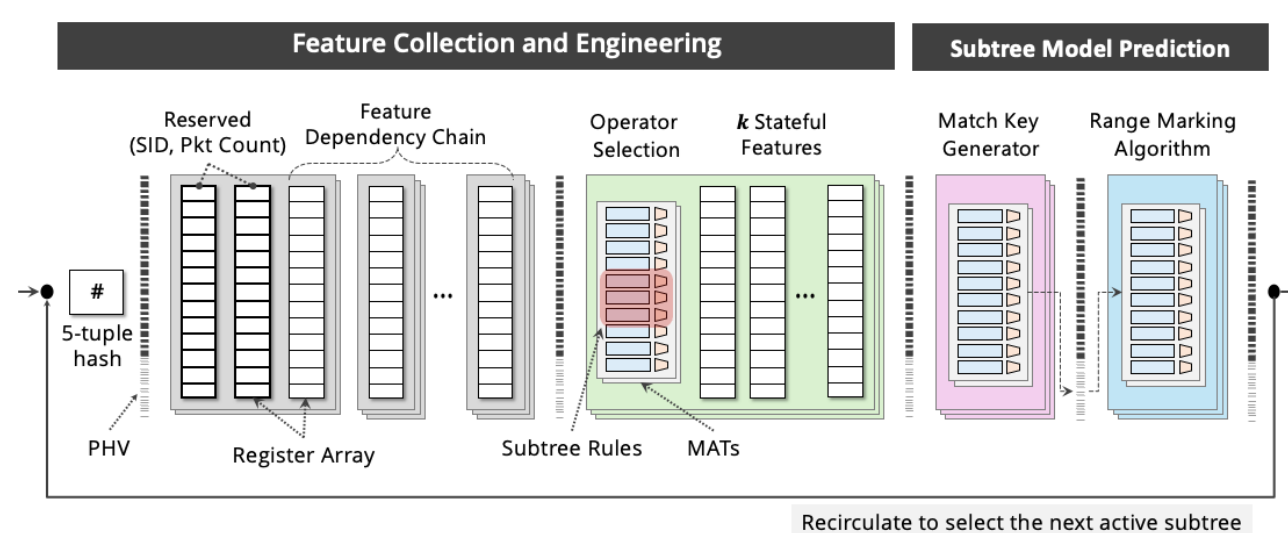
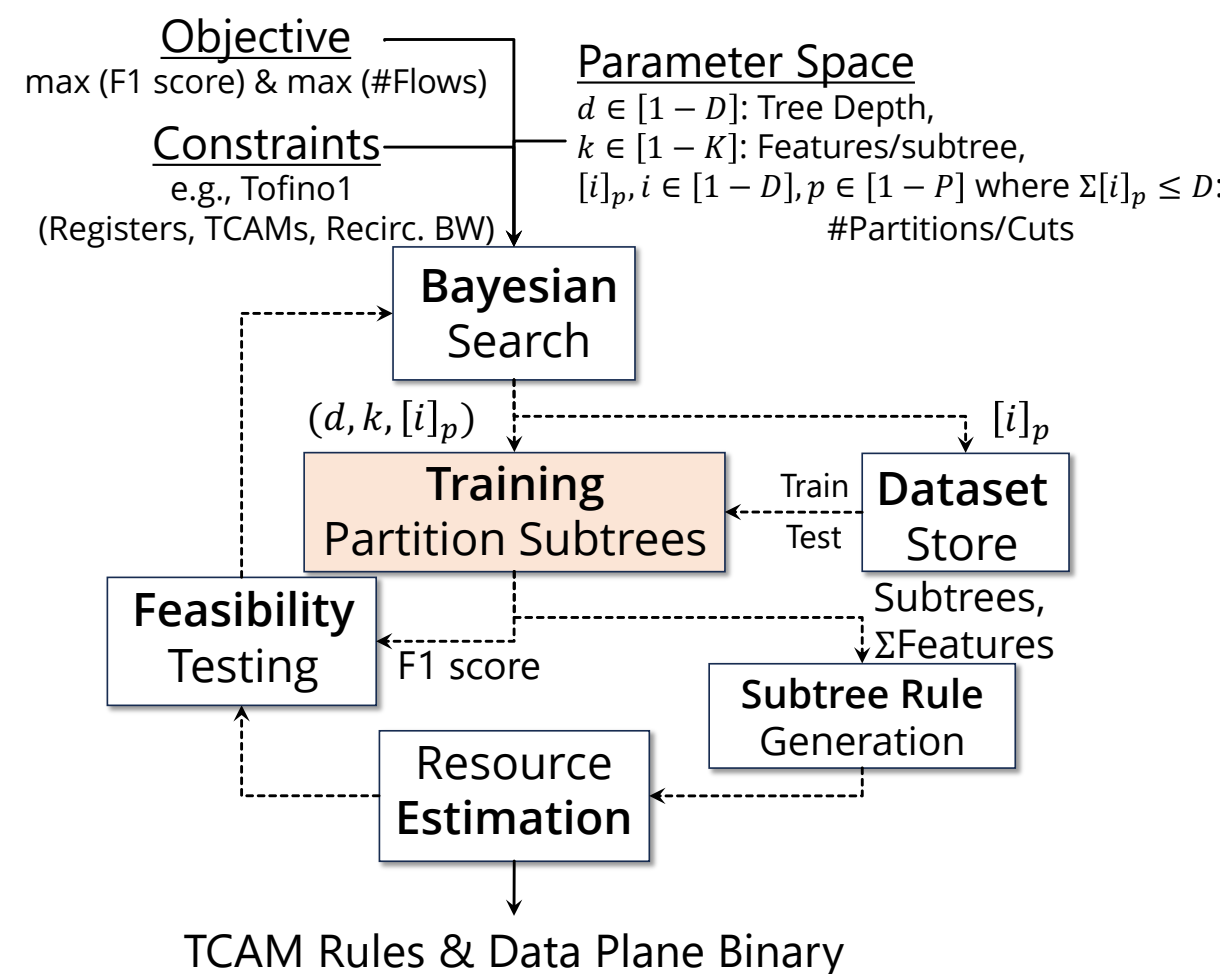


Fig. A high-level overview of SpliDT inference architecture

SpliDT uses **Bayesian Optimization** to efficiently train DTs, tuning **feature count**, **tree depth**, and **partitioning strategy** to find Pareto-optimal models balancing accuracy and flow count.

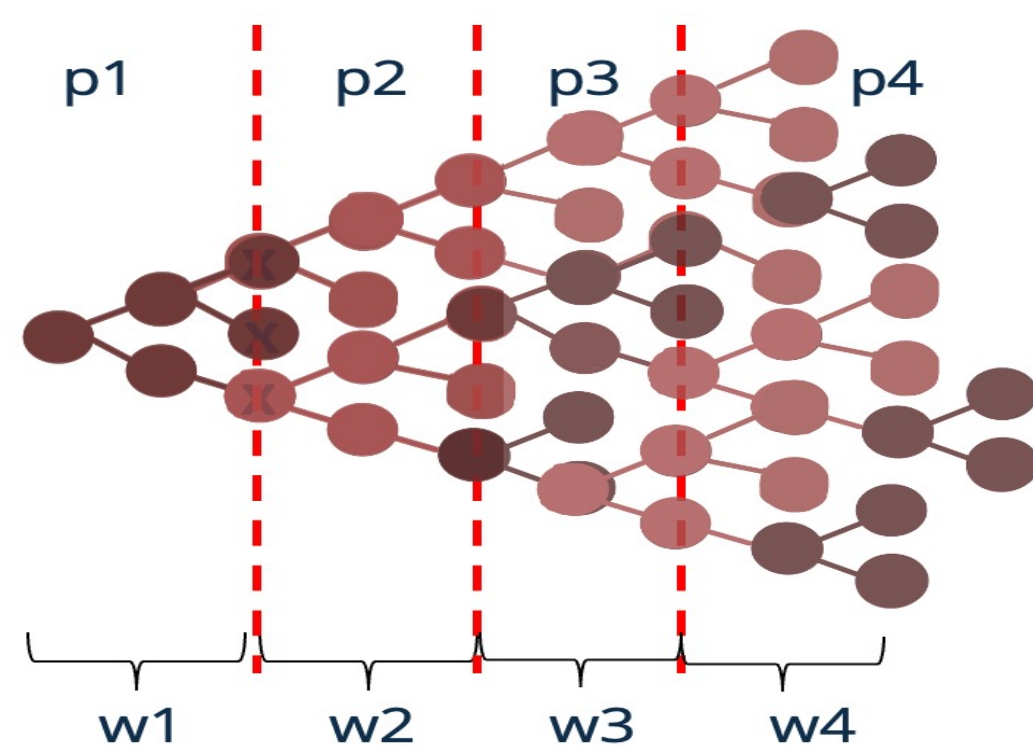


Using the Bayesian optimizer's output — tree depth, top-k features, and partitioning strategy — we train the decision trees sequentially using a **custom SpliDT partitioned DT training algorithm**.

According to the algorithm, Window 1 of flow samples trains the first partition on its features, then flows are split by their leaf.

Window 2 trains the next partition with its features, and so on, until the final depth is reached.

The result is a fully partitioned tree, each containing subtrees with its own features.



Given the trained partitioned DT, we generate the TCAM rules required to represent this model in the data plane. We use the **Range Marking algorithm** to generate TCAM entries for the partitioned decision trees.

These TCAM entries are installed into the feature tables to output range marks as the match key for the model table.

Another set of TCAM entries are generated for the model itself and installed into the model table that matches on these feature range marks to determine the next subtree ID or the predicted class (at the last partition). This representation efficiently encodes each leaf into one TCAM rule for the model table, avoiding the rule explosion problem.

These feature and model entries are generated for each subtree in the partitioned DT and installed into the tables in the data plane inference pipeline and include an exact match on the subtree ID (SID) to select the correct subtree for each partition and flow.

EVALUATION AND RESULTS

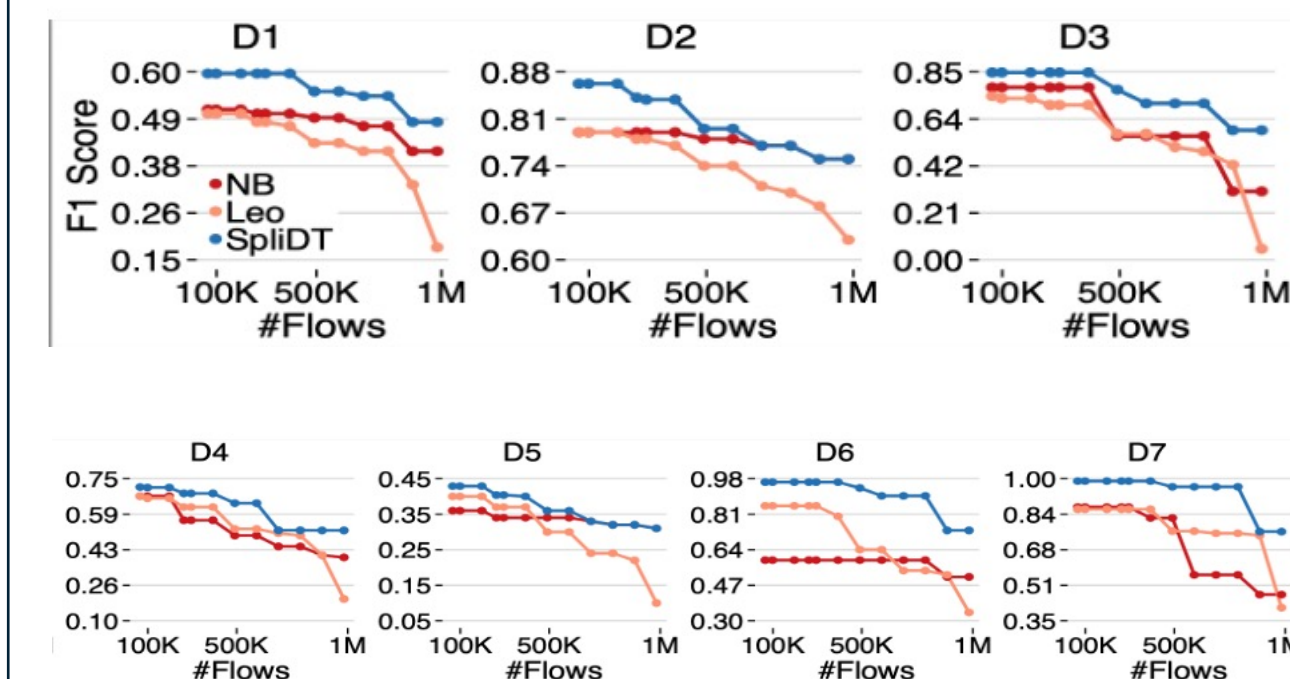


Fig. Pareto frontier of SpliDT vs. baselines, indicating the best F1 score for a given number of flows in the data plane

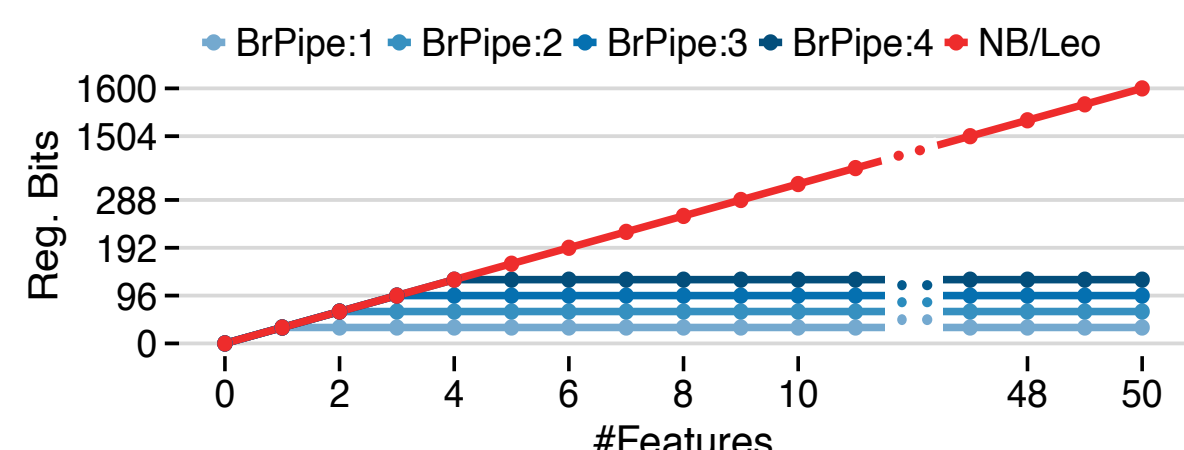


Fig. Register size (bits) vs number of features supported by each model. SpliDT:k is a partitioned tree with k features per subtree.

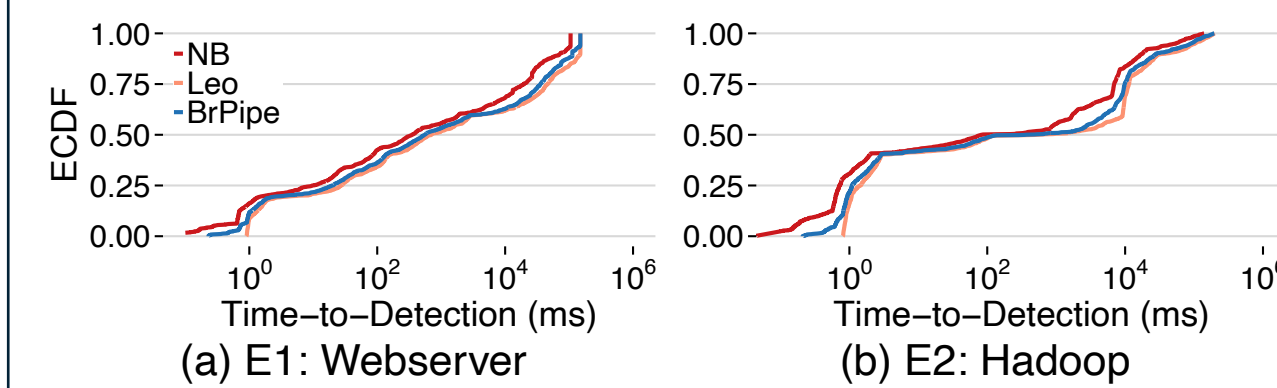


Fig. Flow-level time-to-detection (TTD) of D3 for two datacenter environments (E1-E2)

Dataset	Recirculation Bandwidth (Mbps)	
	E1: Webserver	E2: Hadoop
D1	2.93 ± 2.44	5.99 ± 3.51
D2	6.01 ± 4.01	12.32 ± 5.76
D3	3.58 ± 3.21	7.33 ± 4.62

Fig. Maximum recirculation bandwidth (Mbps) when processing the datasets (D1–3) for two datacenter environments (E1–2)

SUMMARY

- 5× More Stateful Features than SOTA
- Our architecture remains scalable to million of flows.
- Low recirculation overhead (<15 Mbps)
- Preserves flow-level time-to-detection.

NEXT STEPS

Adaptive Window Sizing: Currently, SpliDT uses fixed per-flow window sizes, which may not always align with real-world traffic dynamics. Future work will explore adaptive window sizing across partitions, allowing SpliDT to optimize feature extraction using real flow characteristics.

Security & Robustness: SpliDT relies on flow size metadata stored in packet headers, making it susceptible to spoofing attacks that manipulate window boundaries. If an attacker alters flow size fields, this could lead to incorrect partitioning, misclassifications, resource exhaustion, or even denial-of-service (DoS) attacks. Future work will focus on enhancing security mechanisms, such as redundancy mechanisms or anomaly detection, to protect against header manipulation.

Towards explainable AI with SpliDT: DTs play a central role in explainable AI due to their interpretability and transparency, making them well-suited for high-stakes decision-making. Future work will extend SpliDT to navigate a broader trade-off space, optimizing not just for scalability and efficiency, but also for fidelity with respect to a domain expert-approved and trusted DT model.