

FedCSIS 2026 · Riga, Latvia

EnsembleClassifier

Using Machine Learning to Select Exact Packet Classification Structures for Partitioned Rule Subsets

Ashesh Gaur

Sridhar Radhakrishnan · Mohammed Atiquzzaman

School of Computer Science, University of Oklahoma

github.com/asheshgaur/EnsembleClassifier



Exact multi-field packet classification

A router or firewall matches each incoming packet against a ruleset of axis-aligned hyper-rectangles over d header fields and returns the highest-priority matching rule.

- 1 Scale** Operational rulesets contain thousands to tens of thousands of rules
- 2 Latency** Matching is performed per packet, at line rate, in the data plane
- 3 Update rate** SDN and OpenFlow introduce frequent rule updates, so construction cost is also relevant
- 4 Exactness** First-match semantics must be preserved; an approximate result is not acceptable

Definition

Rule $R_i = \langle [f^1_{\min}, f^1_{\max}], \dots, [f^d_{\min}, f^d_{\max}] \rangle$
with priority p_i and action a_i

Packet P matches R_i iff $P[k]$ lies within the interval of R_i in every field k .

$R^* = \operatorname{argmax} p_i$ over all R_i matching P

Geometrically, the problem reduces to the Orthogonal Point Enclosure Query (OPEQ), for which no method is known to be simultaneously optimal in time and space for $d > 3$.

Limitations of single-structure classifiers

Existing classifiers — including HyperCuts, HyperSplit, CutSplit, PartitionSort and tuple-space methods — differ substantially in mechanism, but all apply a single data structure across the entire ruleset. Two consequences follow.

Heterogeneity of rule subsets

Rule distributions vary across the header space. Different regions exhibit different geometric properties:

- dense overlapping ranges favour hierarchical partitioning
- exact-match dominated regions favour hashing
- mixed intervals favour hybrid methods
- small subsets are adequately served by linear scan

A globally uniform choice is therefore a compromise that is suboptimal for at least part of the ruleset.

Rule representation (the CIDR gap)

Many classifiers assume CIDR-aligned prefixes in order to simplify partitioning.

In practice, operational rules frequently specify arbitrary numerical ranges arising from policy aggregation and application constraints — for example a port range 1024–3000, which is not prefix-representable.

Converting such ranges to prefixes is a principal source of rule replication, which increases both memory consumption and lookup latency.

Proposed approach

Decouple rule-space decomposition from structure selection.

1

Decomposition

Connected-component analysis over field intervals partitions the ruleset into disjoint subsets. Each rule appears in exactly one subset, so no replication is introduced.

2

Selection

For each terminal subset, a supervised model selects a structure from a portfolio of exact classifiers on the basis of local geometric features.

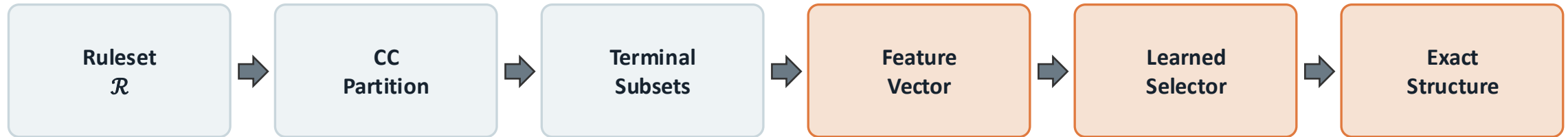
3

Advisory learning

The model determines how packets are searched, not what the answer is. Exactness is a property of the composition tree rather than of the model.

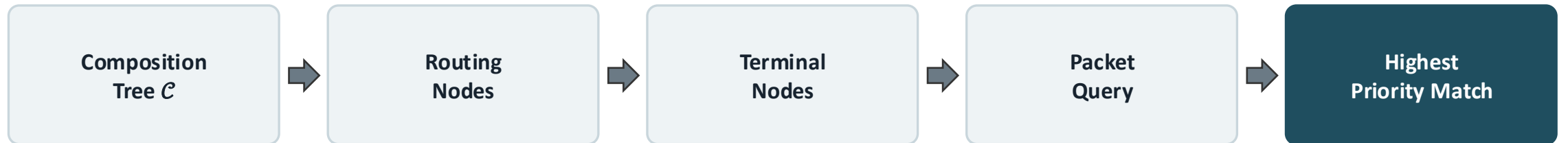
System architecture

CONSTRUCTION PHASE (offline, once per ruleset)



Learning is confined to these three stages

DEPLOYMENT PHASE (per packet, in the data plane)



No inference is performed at query time. The model is evaluated once per terminal node during construction. A packet then traverses a single root-to-leaf path through routing nodes and is matched by one exact classifier — NPTree, PartitionSort, CutSplit, and so on. The data-plane cost attributable to the learned component is zero.

Rule-space decomposition by connected components

For a candidate field, the interval overlap graph is constructed over the rules at that node. If the graph has more than one connected component, no rule in one component can share a matching packet with a rule in another with respect to that field, so the split is safe.

1

Remove bypass fields

Fields identical across all rules in $\mathcal{R}_v$ provide no discriminatory power and are removed.

2

Evaluate terminal conditions

The node is terminal if $|\mathcal{R}_v| \leq B$, no fields remain, or depth $h \geq H$.

3

Select a split field

Return a remaining field whose overlap graph yields multiple components; if none exists, the node is terminal.

4

Recurse per component

Each component becomes a child node with its own rule and packet subset.

Properties of the decomposition

No rule replication

Components are disjoint, so each rule occurs in exactly one terminal node.

$O(n \log n)$ construction

Partitioning is inexpensive; structure selection adds negligible overhead.

Deterministic

No learned parameters or data-derived thresholds are involved at this stage.

Representation-agnostic

Operates on arbitrary numerical ranges; no CIDR alignment is assumed.

Correctness is independent of the learned model

Definition 1 (Routing invariant)

A routing node ensures that any matching rule is contained in exactly one child, and that packets are forwarded to that child.

Theorem 1 (Compositional exactness)

If all terminal nodes use exact classifiers and all routing nodes satisfy the routing invariant, then classification returns the exact highest-priority matching rule.

Proof: by induction on tree depth.

Neither statement refers to the model.

The portfolio $\mathcal{S}$ contains only exact classifiers, so the terminal node is exact for any structure the selector returns. The routing invariant holds because the components produced by the decomposition are disjoint. The hypotheses of Theorem 1 are therefore satisfied for every possible output of the model.

A misprediction degrades performance; it cannot produce an incorrect result.

This distinguishes the approach from neural classifiers that approximate the matching decision itself. First-match semantics are preserved by construction.

Structure selection as supervised classification

Since the decomposition is independent of learning, structure selection is a local decision problem: each terminal node is treated independently, conditioned only on its own rule geometry. Training instances are terminal nodes produced by applying the same partitioning procedure to sampled ClassBench rulesets.

Feature vector s_v

- Unique endpoints
- Mean overlap ratio
- Range entropy
- Prefix-alignable fraction
- Point-rule fraction

All features are normalised. Each is a property of the local subset; no global ruleset statistic is used.

Architecture

Dense 128 ReLU

Dense 128 ReLU

Dense 64 ReLU

Softmax over portfolio $\mathcal{S}$

Trained with cross-entropy against the empirically fastest structure.

Objective and labels

$$S^*(s_i) = \operatorname{argmin}_{S_j \in \mathcal{S}} L_{ij}$$

Labels are obtained by benchmarking every $S_j \in \mathcal{S}$ on every terminal subset, so the optimum is measured rather than estimated. The objective therefore coincides with the evaluation metric.

70,160 terminal nodes from 420 ClassBench rulesets; disjoint files for training and evaluation.

Experimental methodology

1

Implementation

- C++, built on prior implementations of NPTree-Split, NPTree-Cut, HyperCuts, HyperSplit, SmartSplit, CutSplit, CutTSS and PartitionSort
- 32-core AMD Threadripper, 128 GB RAM

2

Rulesets

- ClassBench ACL, FW and IPC, 1K–64K rules
- 60 files per size: 5 configuration files each for ACL and FW, 2 for IPC, with 5 seeds per configuration
- 420 ruleset files in total

3

Protocol

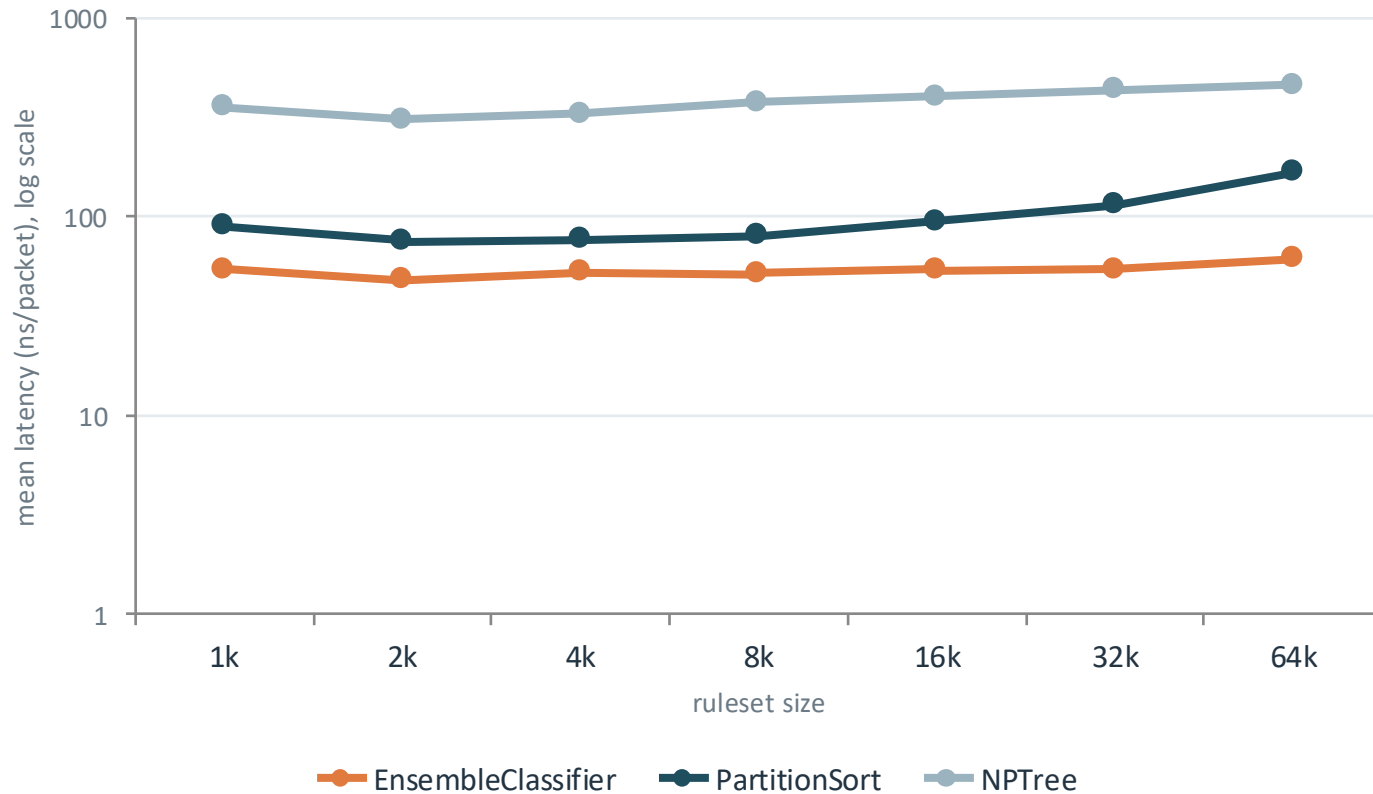
- Separate construction trace $\mathcal{T}$ and disjoint evaluation trace $\mathcal{V}$
- A single packet trace common to all classifiers
- Metrics: construction time, classification latency, memory

Configuration fairness

Each baseline is evaluated under its best global configuration, obtained by sweeping standard parameter ranges. EnsembleClassifier selects configurations per terminal node from learned lookup tables. Per-node configuration is not available to the baselines, as they are single-structure methods; this difference is the capability under evaluation.

RESULTS

Classification latency across ruleset sizes



Mean over all sizes

1.85× speedup relative to PartitionSort, the fastest baseline in the portfolio (45.97% latency reduction).

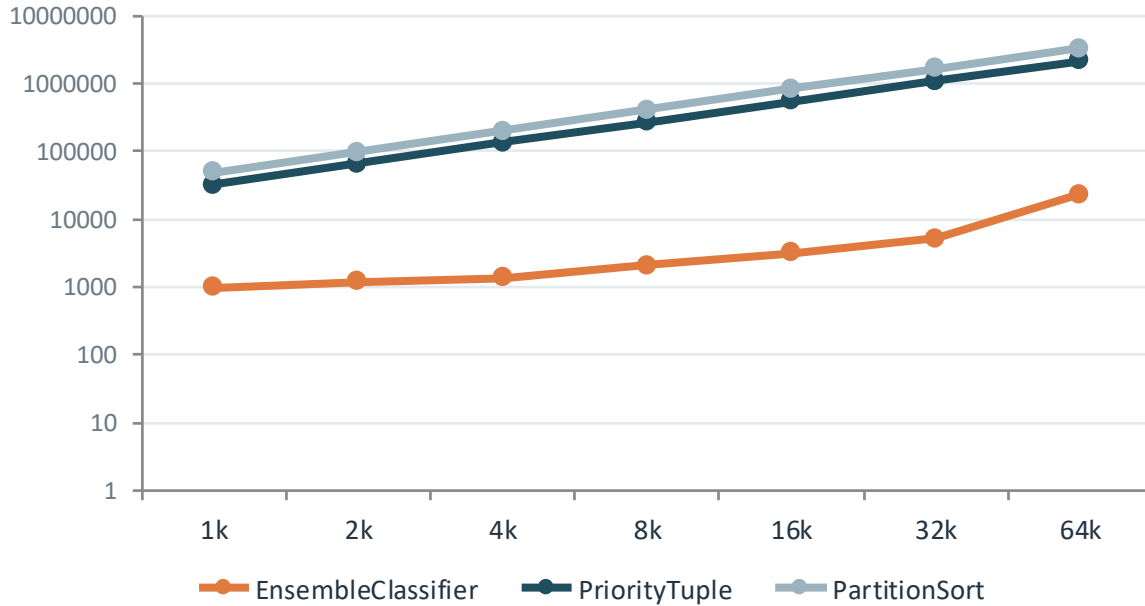
Observations

Baseline latency increases with ruleset size; PartitionSort rises from 89 to 167 ns between 1k and 64k. EnsembleClassifier remains between 48 and 61 ns across the same 64× increase in rule count. Decomposition bounds the work performed per packet: matching occurs within a single terminal subset, and subset size grows more slowly than the ruleset.

RESULTS

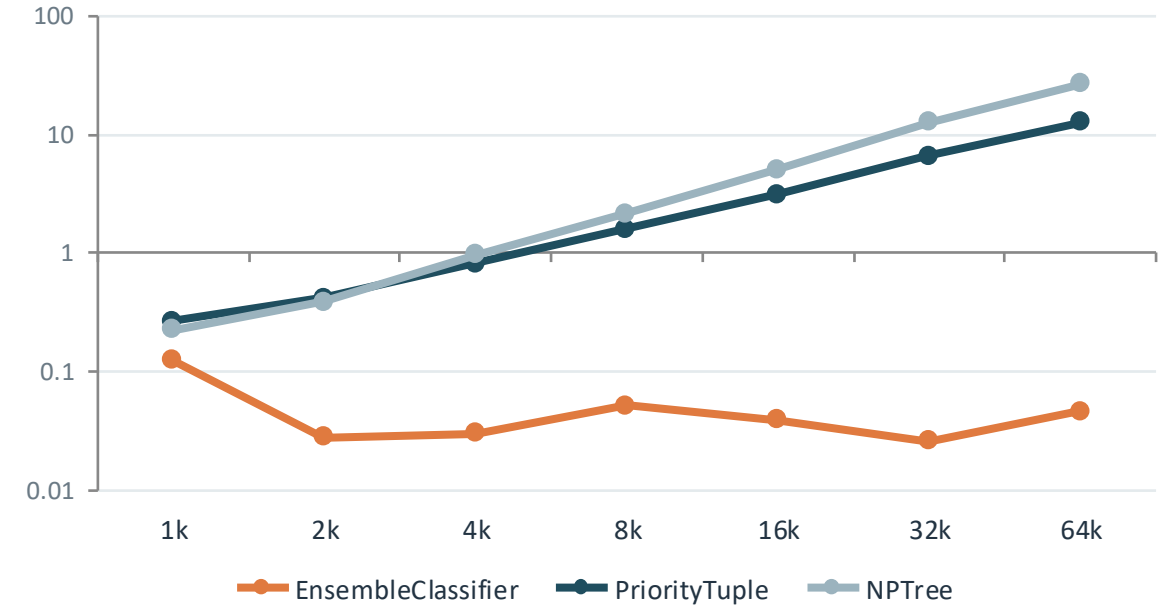
Memory footprint and construction time

MEMORY FOOTPRINT (bytes, log scale)



99.14% reduction in memory. 616,991 → 5,320 bytes on average, relative to PriorityTuple.

CONSTRUCTION TIME (ms, log scale)



98.65% reduction in construction time. 3.654 → 0.049 ms, and approximately constant in ruleset size.

RESULTS

Ablation: fixed structure assignment at terminal nodes

To isolate the contribution of the learned selector from that of the decomposition, we retain the composition tree and assign a single fixed structure to every terminal node. Averages over all 420 rulesets:

Assignment at terminal nodes	Latency (ns)	Memory (B)	Build (ms)
CutSplit (fixed)	53.22	9,813	0.655
Learned selector	53.93	5,320	0.049
TabTree (fixed)	64.17	6,778	0.214
PartitionSort (fixed)	142.89	948,713	11.83
NPTree (fixed)	415.20	5,801,177	6.69
ByteCuts (fixed)	748.12	38,552,958	28,215

Reading the table

On latency alone, a fixed CutSplit assignment is marginally superior (53.22 vs 53.93 ns). The decomposition, not the selector, accounts for most of the latency improvement.

The selector's contribution is in the remaining two metrics: 1.8× lower memory and 13× lower construction time than the best fixed assignment.

Identifying the best fixed assignment requires benchmarking the entire portfolio, which is the procedure selection replaces. A fixed assignment that is adequate on ClassBench also carries no guarantee of transferring to other rule distributions.

Summary of results

Each metric is compared against the strongest baseline in the portfolio for that metric. The second row of each pair reports EnsembleClassifier trained only on 1k rulesets and evaluated across the full range.

Metric	Best baseline	EnsembleClassifier	Improvement
Query latency (s)	$9.98 \cdot 10^{-8}$	$5.39 \cdot 10^{-8}$	1.85×
<i>trained on 1k only</i>		$6.4 \cdot 10^{-8}$	1.55×
Memory (bytes)	616,991	5,320	99.14%
<i>trained on 1k only</i>		6,784	98.9%
Construction time (ms)	3.654	0.0494	98.65%
<i>trained on 1k only</i>		0.236	93.5%

Training restricted to 1k rulesets recovers most of the improvement, suggesting the relevant geometric properties are largely scale-invariant.

Conclusions

Packet classification as a composition problem

1

Performance is governed by local geometry

Classifier performance is determined by the geometric properties of rule subsets rather than by global ruleset characteristics, which motivates a shift from single-structure designs to adaptive composition.

2

Correctness follows from structure, not from learning

Connected-component decomposition yields disjoint subsets without replication; together with the routing invariant and an exact-only portfolio, this establishes first-match semantics for every possible model output.

3

Improvements are concurrent across metrics

Latency, memory and construction time all improve relative to the strongest baseline for each, with the margin increasing in ruleset size. The ablation indicates the decomposition accounts for the latency result and selection for the other two.

More generally, the pattern of a learned component that reorganises a system while a structural invariant preserves its correctness may be applicable to other domains in which systems must operate under hard correctness constraints.

Thank you

I would be glad to respond to questions by email, or in the session if a live slot is available.

Ashesh Gaur

asheshgaur@ou.edu

School of Computer Science, University of Oklahoma

with Sridhar Radhakrishnan and Mohammed Atiquzzaman

Implementation: github.com/asheshgaur/EnsembleClassifier

