暨南大學
JINAN UNIVERSITY

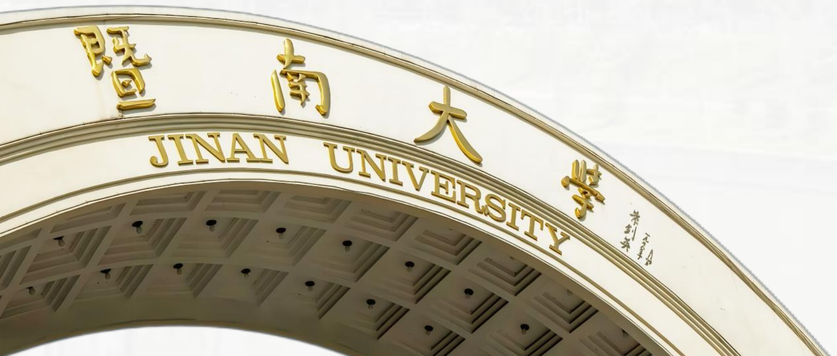
Time Slot based Heavy Changer Prediction on P4 Switches

P4交换机上基于时隙的Heavy Changer预测研究

陈奕铭 2020103006
指导教师 崔林

暨南大學
JINAN UNIVERSITY

Contents



Backgrounds

Heavy Changer, Time Slot, Ahead Definition

- ***Slot_size*: time slot duration (in seconds)**
- ***Ahead*: future slot count**
- *time slots* or *epochs* or *slots:* regular time intervals
- (x, v_x) x *flow identifier,* v_x *packet size*
- $\phi (0 < \phi < 1)$ *predefined heavy changer threshold*
- D *absolute change in total flow* $D(x)$ *absolute change in packets size*
- Heavy changer: $D(x) > \phi D$
- Heavy changers signal congestion or malicious attacks.

Motivation and Challenges

Motivation

- **Real-time identification** of heavy changers

Detection methods need packets

- **Avoids extra** control-data plane **communication**

Method Implement on the control plane need extra communication

Challenges

- **Skewed Data Distribution:**

  Imbalance data problem

  Overrepresentation of normal flows

- **Data Plane Constraints:**

  Limited memory and processing power

  No floating-point operations, loops, division

- **P4 Language Limitations:**

  Access only to packet metadata

  Cannot predict across multiple flows

System Overview

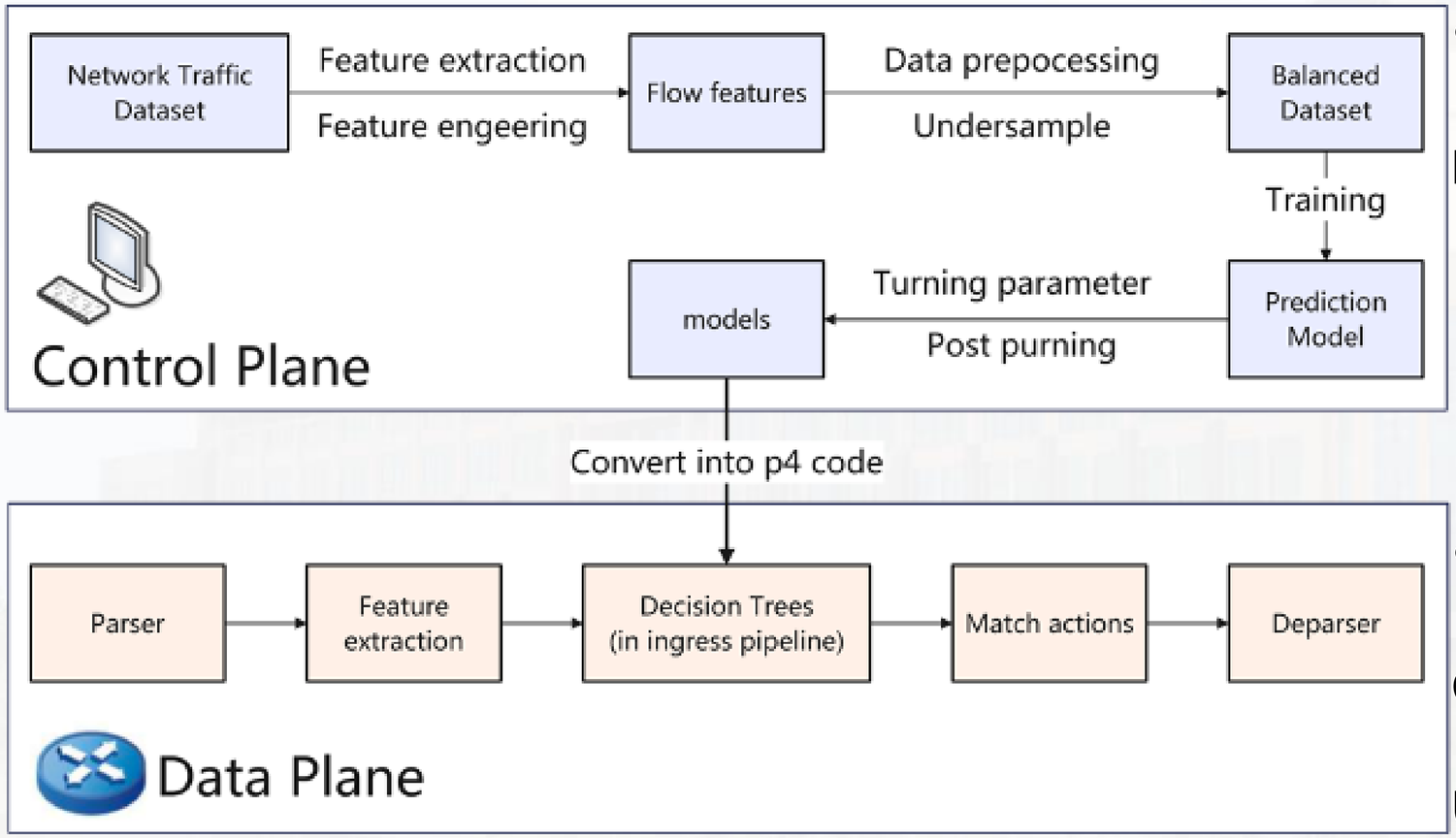


Figure 1. pChanger Procedure

Two Stages

- Training in the control plane

Mitigate data imbalanced problem

- Implement in the data plane

Convert not allowed operation

Using register to manage slots, timestamp

Training Process

Feature

- Two Types of Features:

Packet-level: Dport, Sport, Proto

Extracted from metadata in P4 parse

Flow statistics:

Require register

Length_max, Length_min, Length_mean

IAT_max, IAT_min, IAT_mean

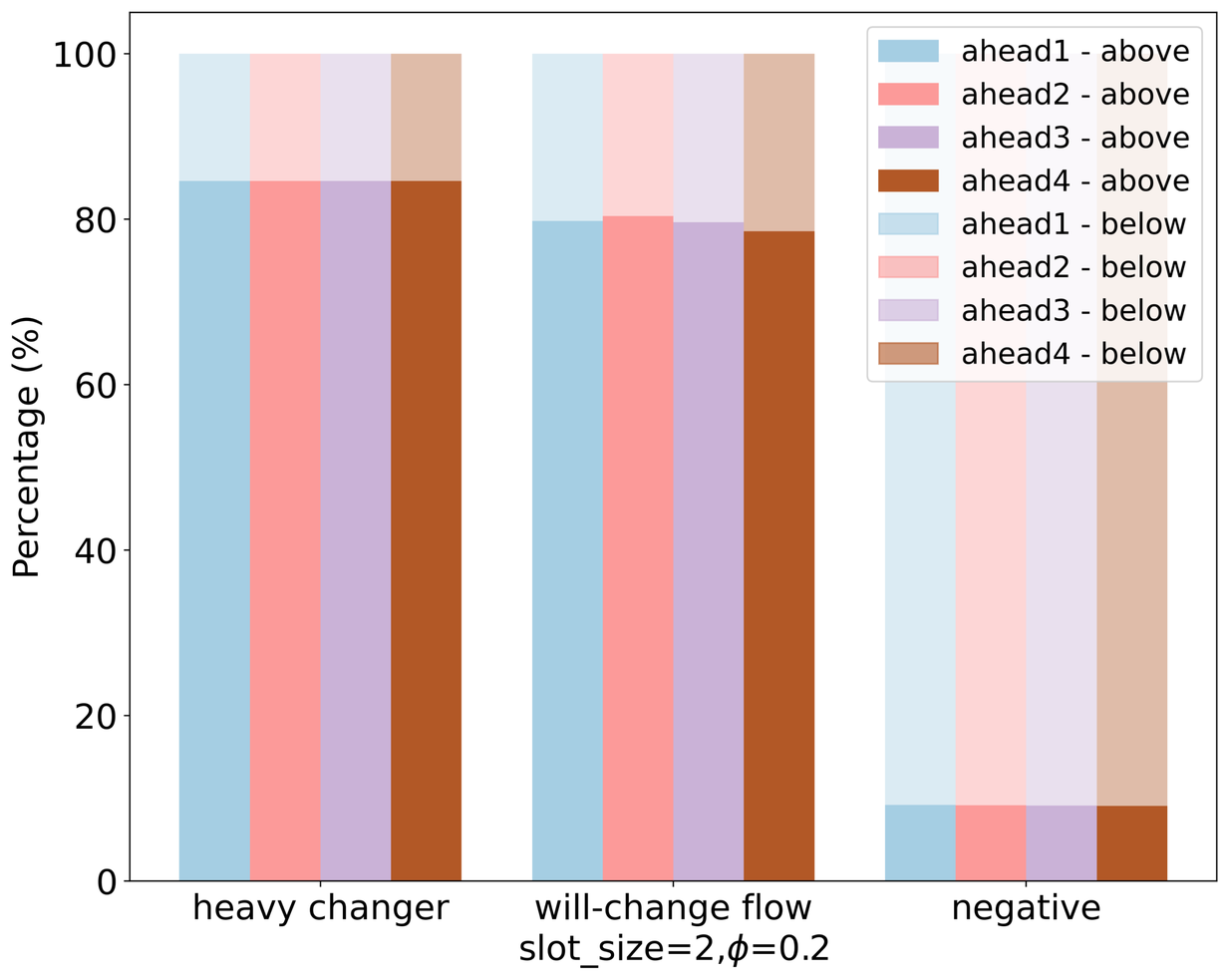
ACK_flag, FIN_flag, SYN_flag

PSH_flag, RST_flag, ECE_flag

Packet_count

- **Removed Identifiers**: Excluding destination IPs, and source IPs.
- **Eliminated Timestamps**: Ensure heavy flow detection is time-independent.

Data Filtering Strategies



- **Strategy 1**: Eliminate flows with fewer than 10 packets.
- **Strategy 2**: Eliminate flows with fewer than 20 packets.
- **Result**: Higher percentage of significant flows (heavy changers and will-change flows) retained.

Figure 2. Strategy 2 example: slot_size 2, threshold 0.2

Methods to mitigate imbalanced data problem

•One-sided Selection

Algorithm One-sided Selection

Input:

S: the initial training set.

C: a stable subset composed of every positive from S alongside a negative chosen randomly

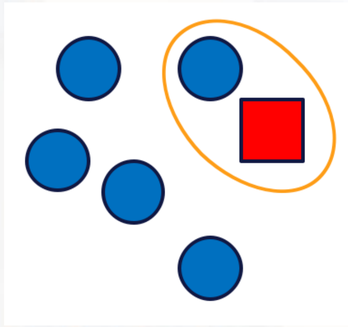
Output:

T: reduced cleaner training subset

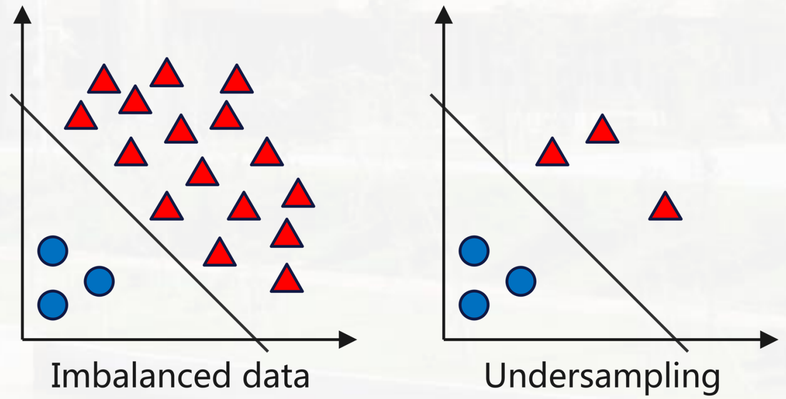
1. **For** sample s in S **do**
2. Classify s with *C* using 1-NN and verify label accuracy
3. **If** s is misclassified, move s into *C*
4. Eliminate all negative examples in C involved in Tomek links
5. *T*←*C*

Tomek Link to identify borderline and noisy examples (either marginal or noisy)

Two samples with different labels and no third sample are closer to them than they are to each other.

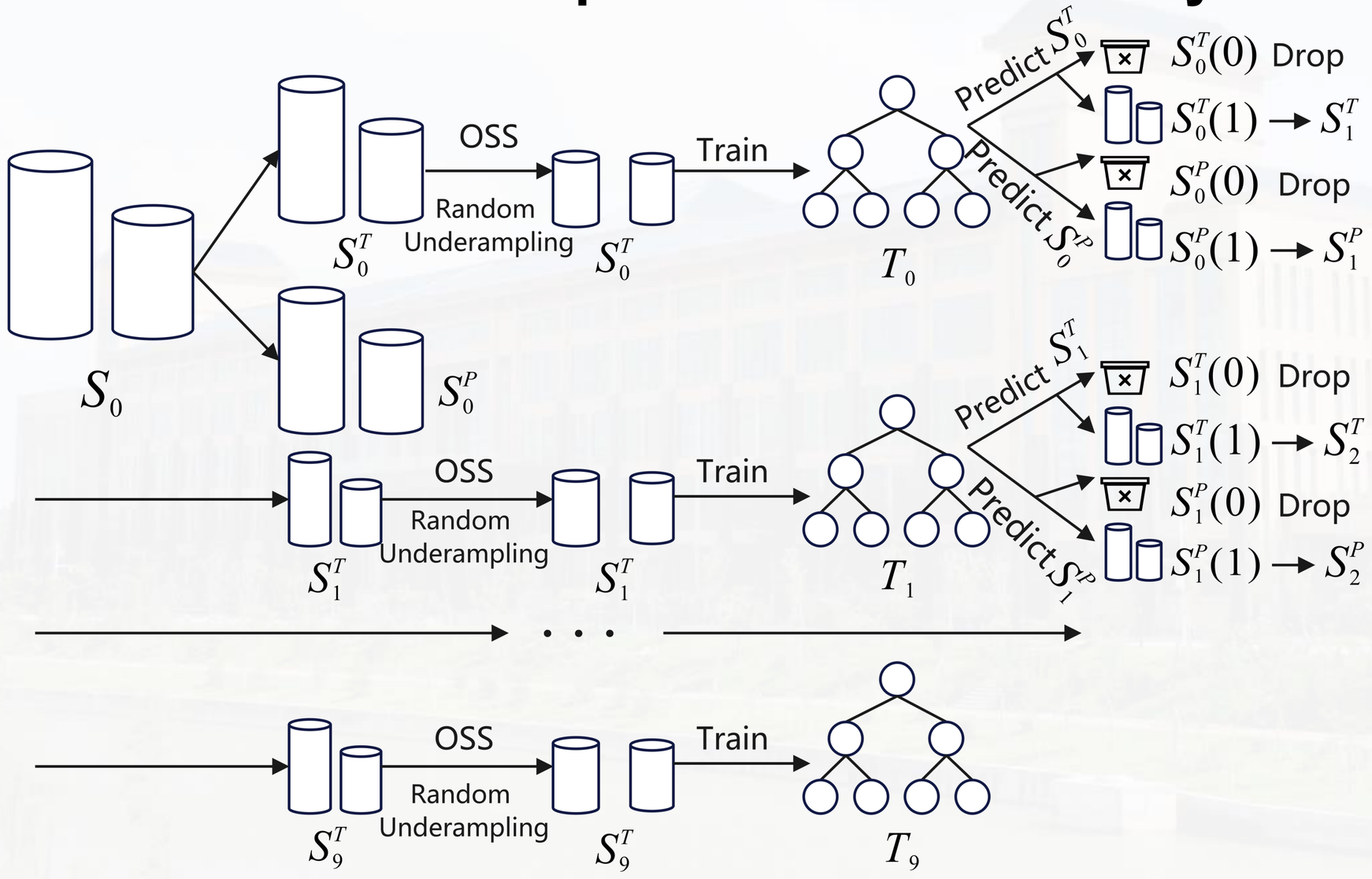


• Under Sampling



Multi-stage model training process Post pruning

Aim: Enhance subsequent models' accuracy



Solution of Data Plane Constraints Challenges

Eliminate floating-point operations and division

- **Eliminate floating-point:**

  Round down to the nearest integer

- **Eliminate division:**

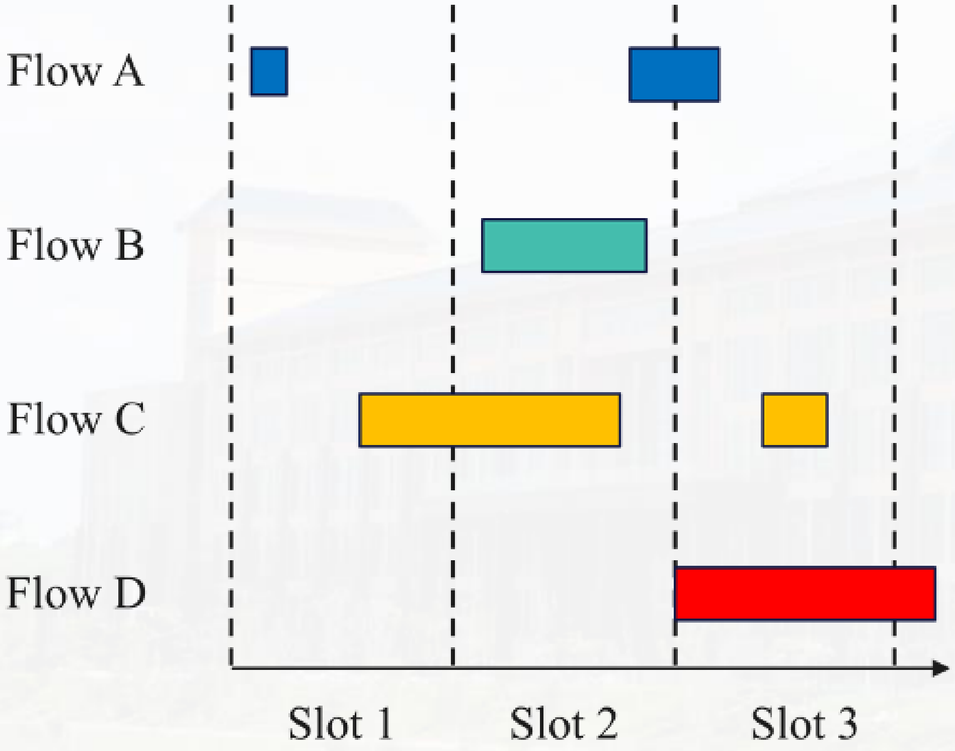
  Use exponentially weighted moving average (EWMA) and Set α=0.5

  Formula becomes $S_t = \frac{1}{2} y_t + \frac{1}{2} S_{t-1}$ i.e., $S_t = (y_t + S_{t-1}) >> 1$

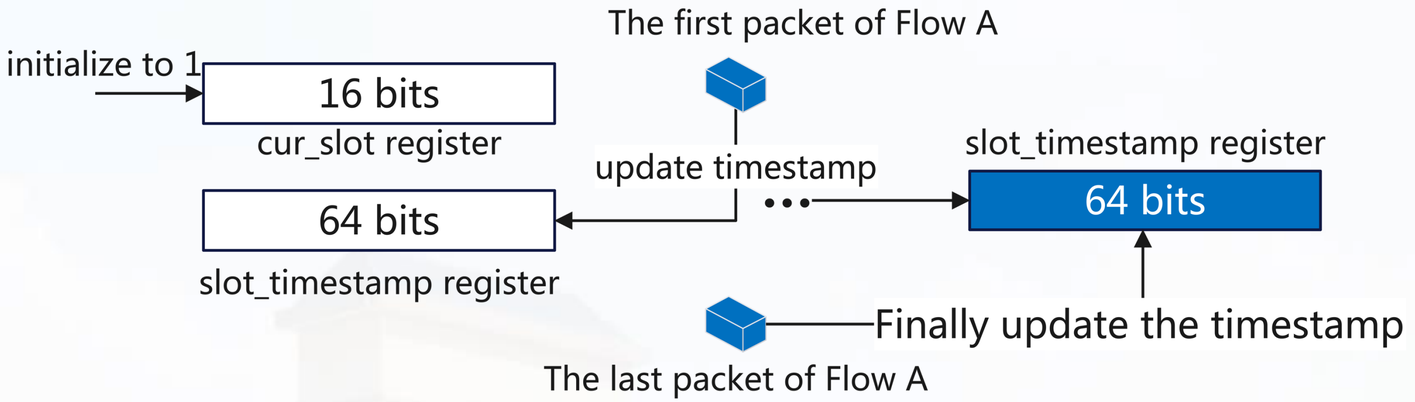
  when $t > 1$

Data Plane Design

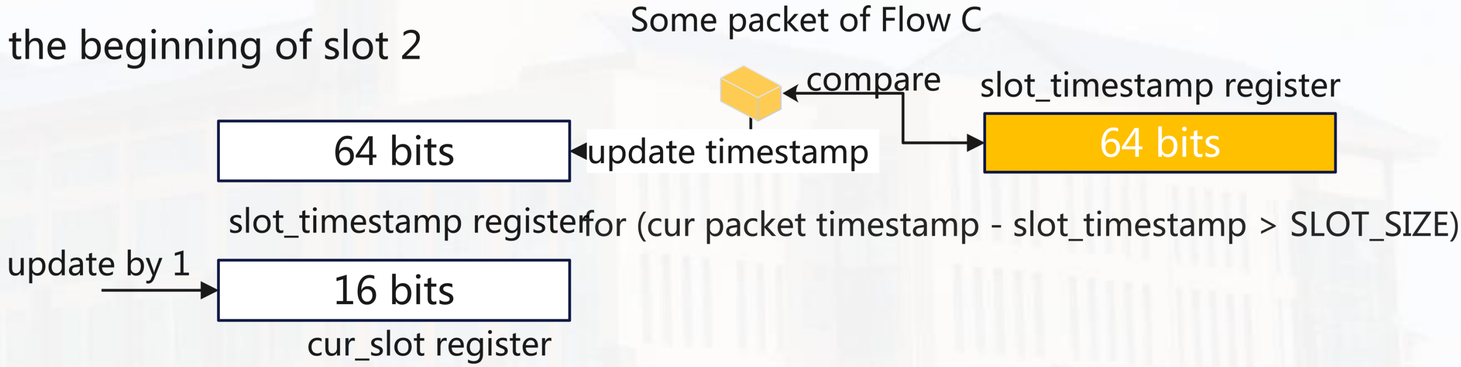
Time-slot based feature prediction in P4 programming



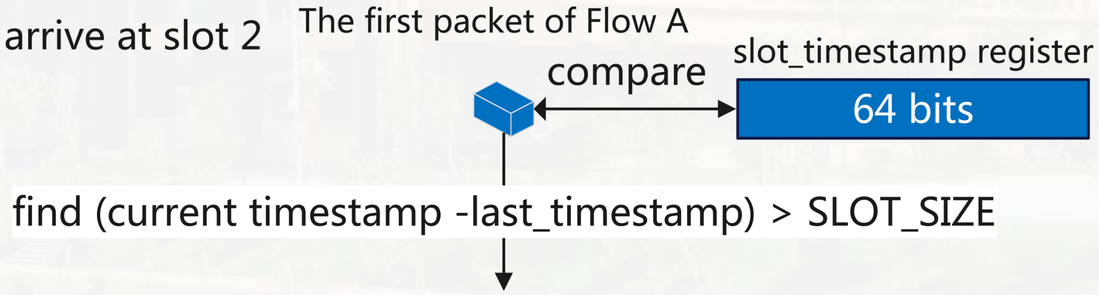
In the beginning of slot 1



In the beginning of slot 2



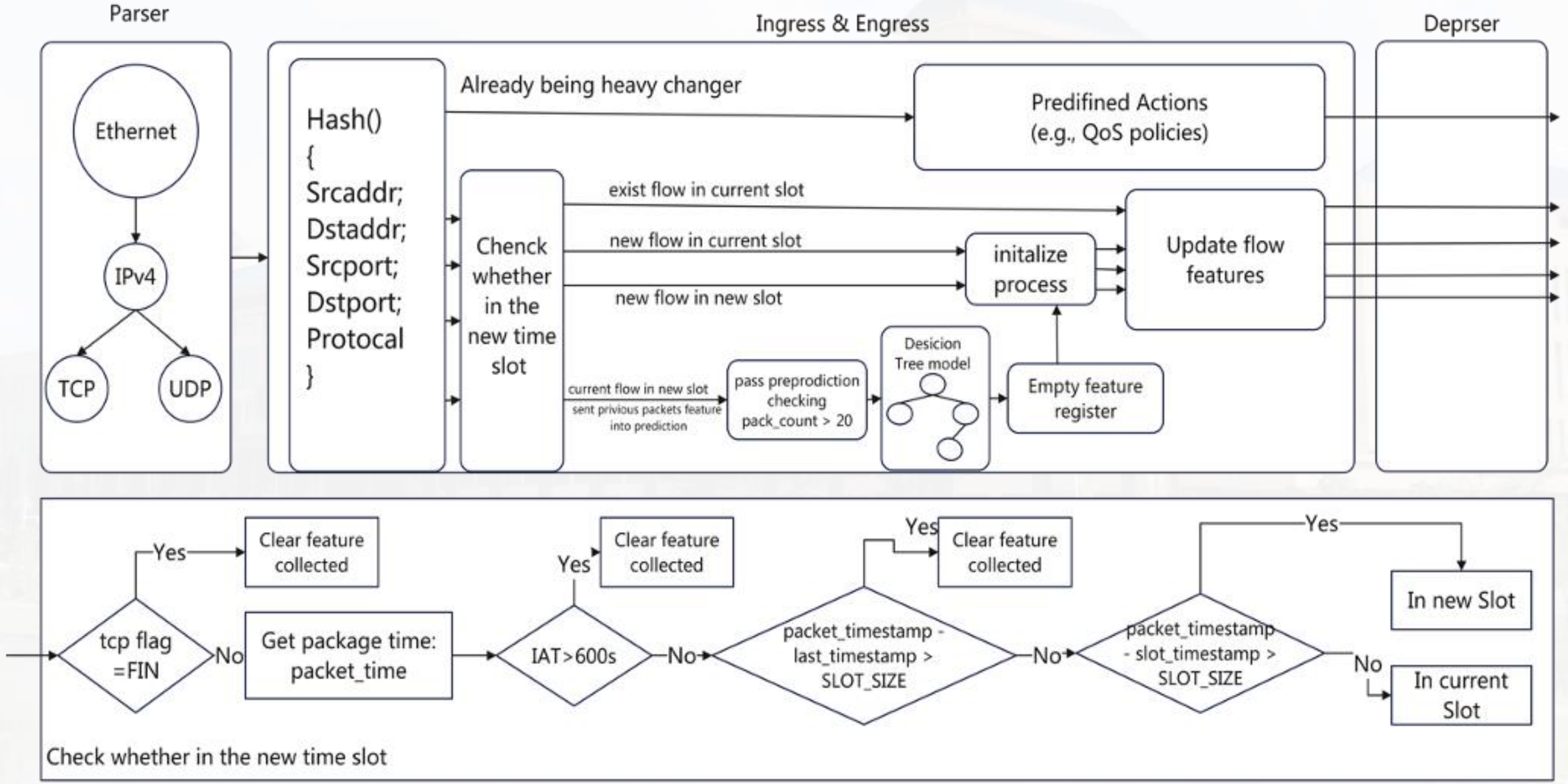
When packet A arrive at slot 2



1. Predict Flow A first
2. Set feature registers to 0
3. then accpt new packets

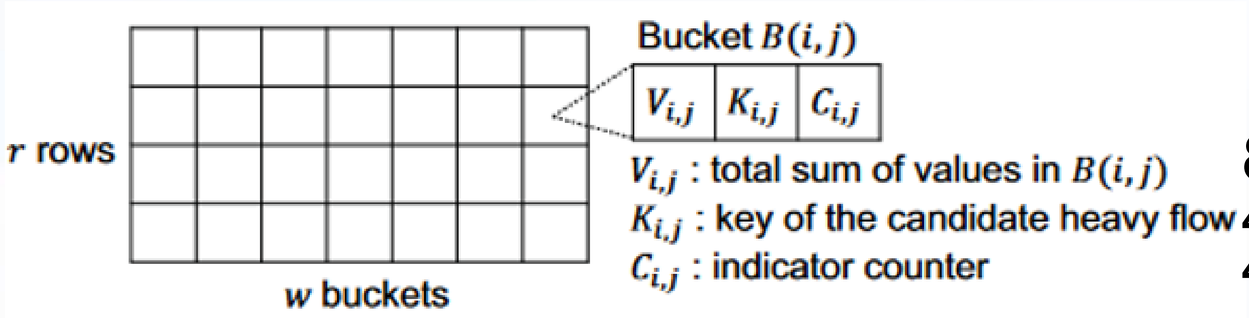
Implementation in the data plane

pChanger Pipeline Overview



4. Evaluation

4.1 Experiment Setup



1) Testbed:

- Server: Virtual server with 4 cores Intel i7-8550U 1.80GHz CPU, 20GB RAM.
- OS: Ubuntu 20.04.6 LTS.
- Datasets: Loaded into memory to avoid I/O overhead.

For P4 evaluation

- Switch: bmv2 simple switch, input: veth2, output: veth4.

2) Dataset:

- Source: UNIV1 dataset from IMC 2010.
- Preprocessing: Payload nulled, IP anonymized with SHA1
- Division: Time slots of 2 seconds

For P4 evaluation

- Experiment Data: First 6 slots with slot_size=2s, total 29,393 packets.

3) Methodology:

- Ground Truth: Mark true heavy changers based on threshold.
- Comparison: MV-Sketch.
- Memory Limiting: Adjust sketch width, use resource.setrlimit for pChanger.
- Feature Extraction: EWMA for averages counting

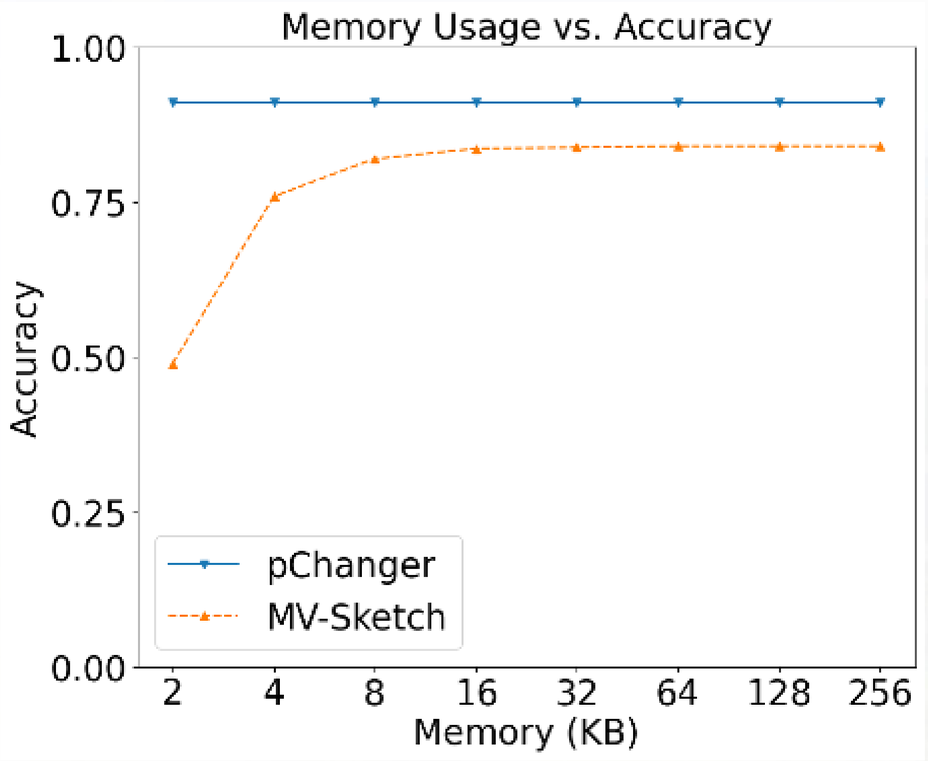
4) Metrics:

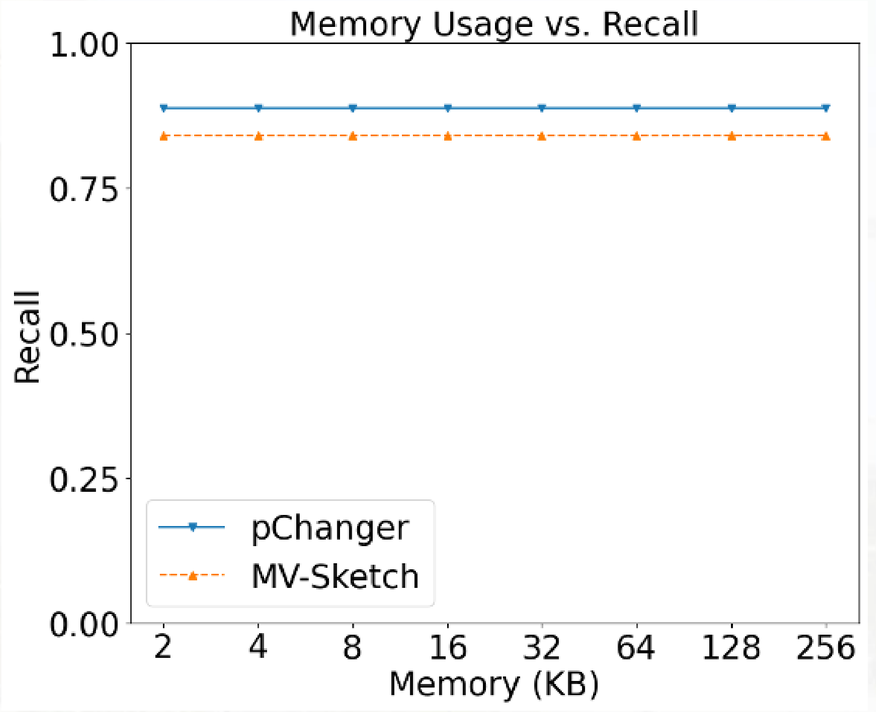
- Accuracy: Correctly identified flows.
- Precision: True heavy changers among reported flows.
- Recall: True heavy changers reported.
- F1-score: Harmonic mean of precision and recall.
- Update Throughput: Packets processed per second (Mbits/s).

Results and analysis

Experiment based on traces

pChanger consistently maintains a high recall of 0.89 across all memory sizes, outperforming MV-Sketch, which improves from 0.81 to 0.84 as memory increases.

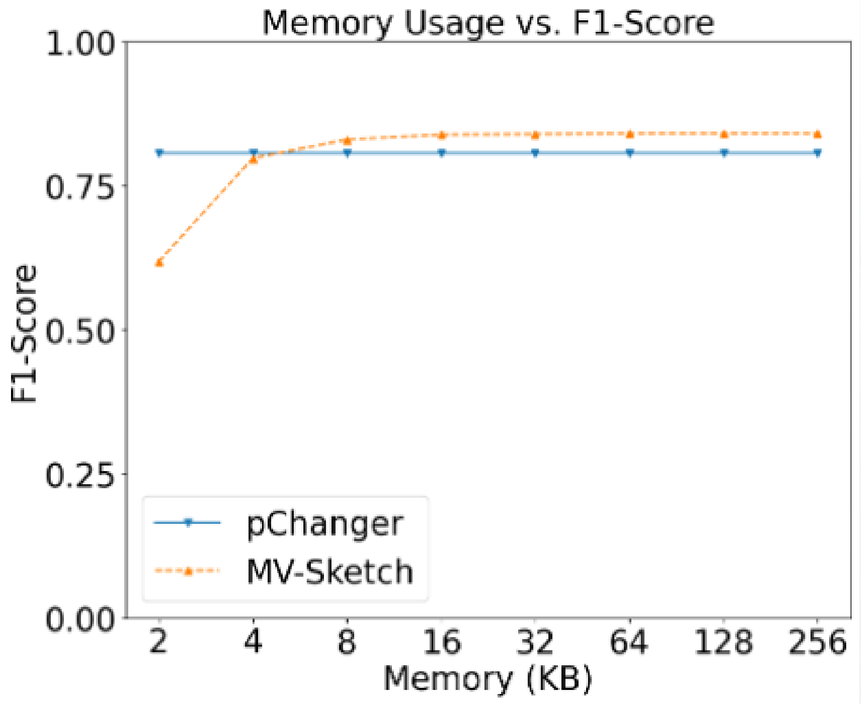


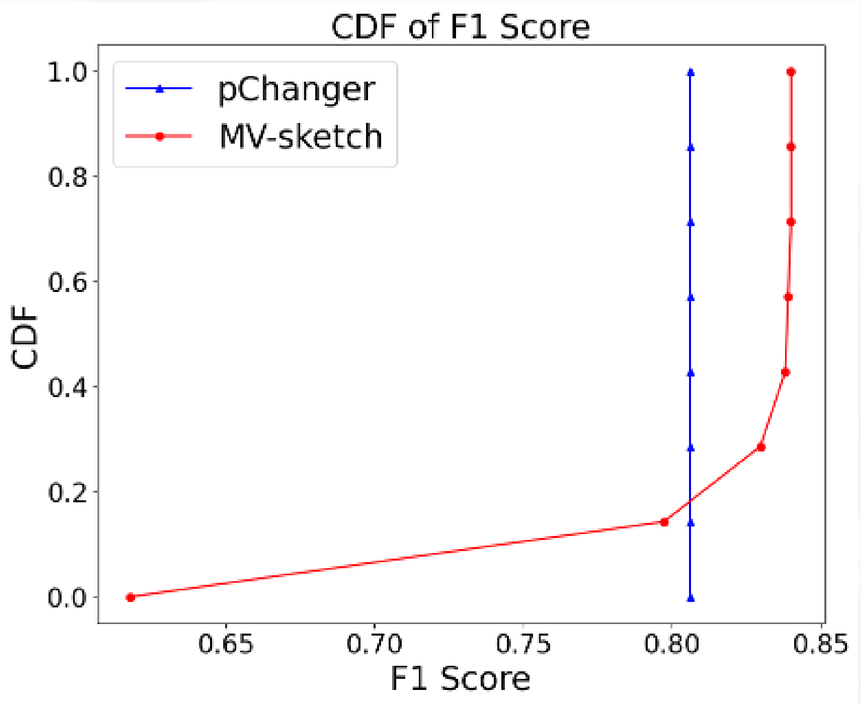


pChanger maintains high accuracy across all memory capacities, outperforming MV-Sketch, which improved accuracy only up to 4KB

Results and analysis

Experiment based on traces





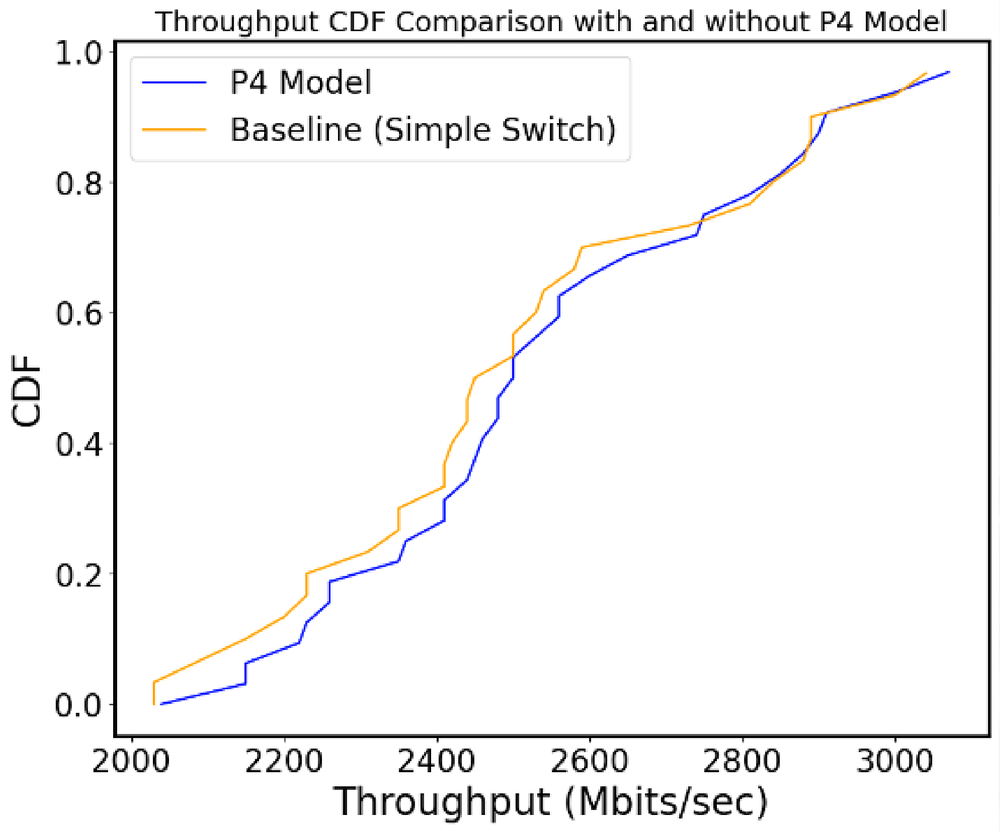
pChanger outperforms MV-Sketch at low memory, but MV-Sketch surpasses pChanger as memory increases.

Results and analysis

Experiment on P4 Simple Switch

pChanger can run prediction at line rate

Method	Accuracy	Recall	Precision	F1-Score
pChanger	0.9997	0.7500	0.6000	0.6667



- **Findings:**
  - High accuracy due to a small number of packets.
  - Rounding floating-point numbers reduces performance.
  - Predicts up to 4 slots in advance

5. Conclusion and Future works

pChanger: programmable heavy changer prediction based on time slots

- **Advantages:**
  - Reduces network congestion by predicting before heavy changers occur.
  - Timeliness: Processes data in time slots.
  - High accuracy and recall.
  - Low memory usage doesn't impact throughput.
  - Enables linear prediction.
- **Limitations:**
  - Can't utilize extra memory effectively.
  - Generalization issues exist

- **Future Work:**
  - Explore multiple decision trees or random forests.
  - Introduce more trees when extra memory is available.
  - Allow controllers to dynamically configure and modify ML models.
  - Combine ML models with sketch-based methods.