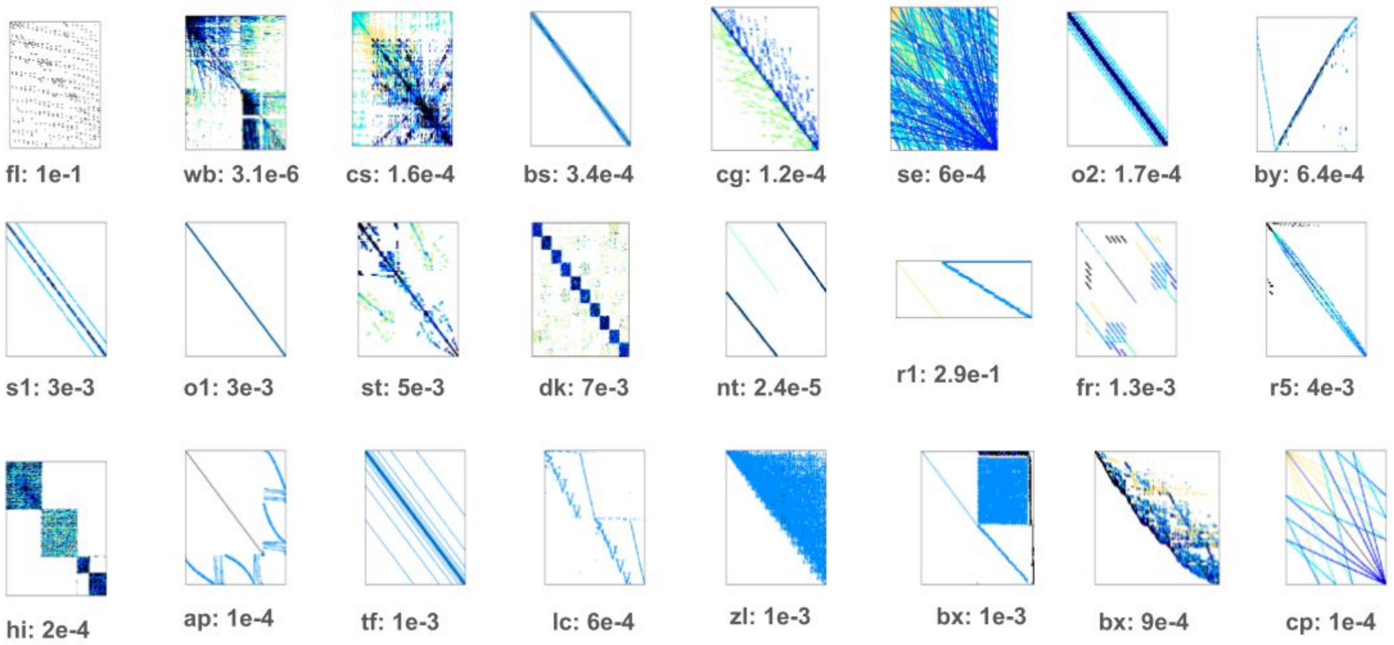
Misam: Using **M**L **i**n Dataflow Selection of **S**p**a**rse-Sparse **M**atrix Multiplication

Sanjali Yadav, Bahar Asgari

University of Maryland

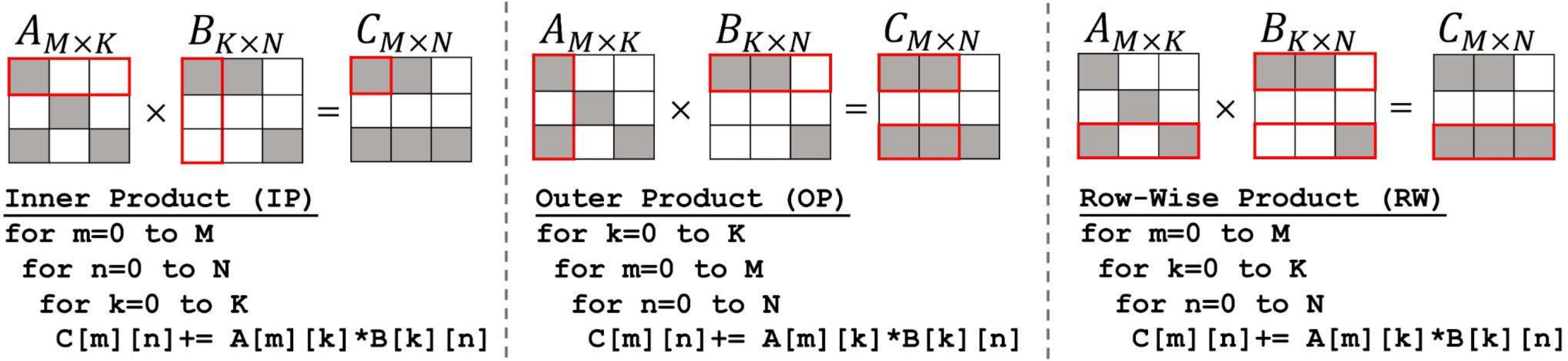
Sparse-Sparse Matrix Multiplication



Irregular structure of sparse workloads across various domains

Current Approaches

- Hardware accelerators customized for the three widely recognized SpGEMM execution dataflow schemes: inner product (IP), outer product (OP) and row-wise product (RW).



Issues with the current approaches

- Employ a fixed execution dataflow, which optimizes the input or output data reuse, at the expense of the other.
- The performance is sub-optimal if the sparsity of the workload does not align with the rigid design of the accelerator.

To achieve a more universal hardware, we require a mechanism to select the best dataflow for diverse workloads across different domains.

Selecting Optimal Dataflow

- Works like Spada and Flexagon acknowledge the limitations of fixed dataflow designs
- Spada: Uses window-based profiling to determine efficient dataflows; risks sub-optimal choices due to insufficient profiling.
- Flexagon: Simple profiling based on SpGEMM characteristics to select dataflows; acknowledges need for more comprehensive methods.

Current profiling methods lack accuracy and generalization.

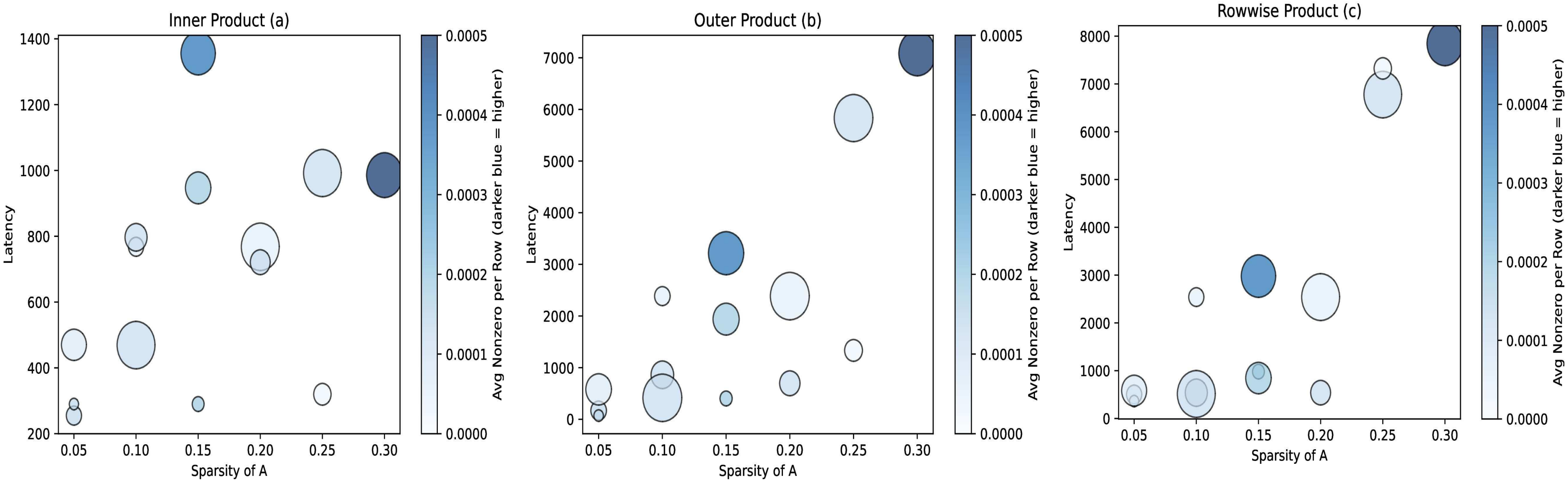
Using Machine Learning in Dataflow Selection

- The characteristics of this problem are well-suited to machine learning (ML) techniques commonly employed in data classification – ***given the features of the input matrices, we can categorize them into classes corresponding to different data flow schemes.***

Objective

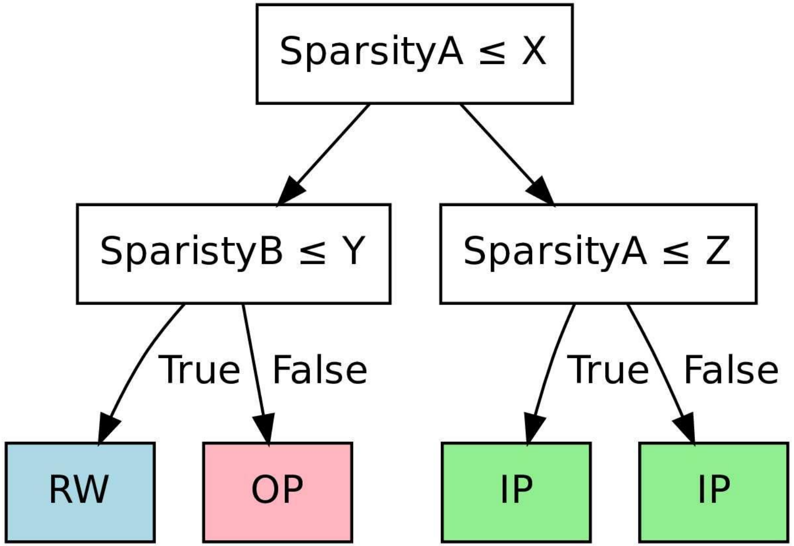
- Assess whether employing machine learning (ML) techniques can provide a more viable and efficient solution compared to traditional approaches.
- Determine which ML techniques offer the best balance between prediction accuracy and model efficiency.

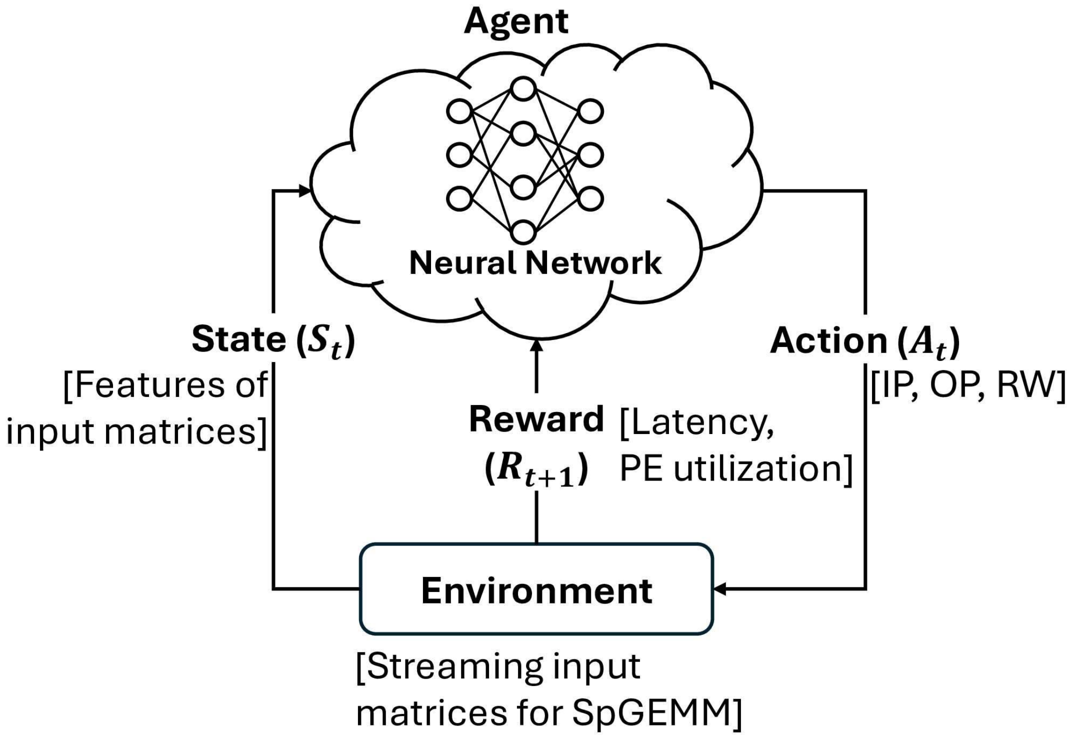
Sparsity Analysis



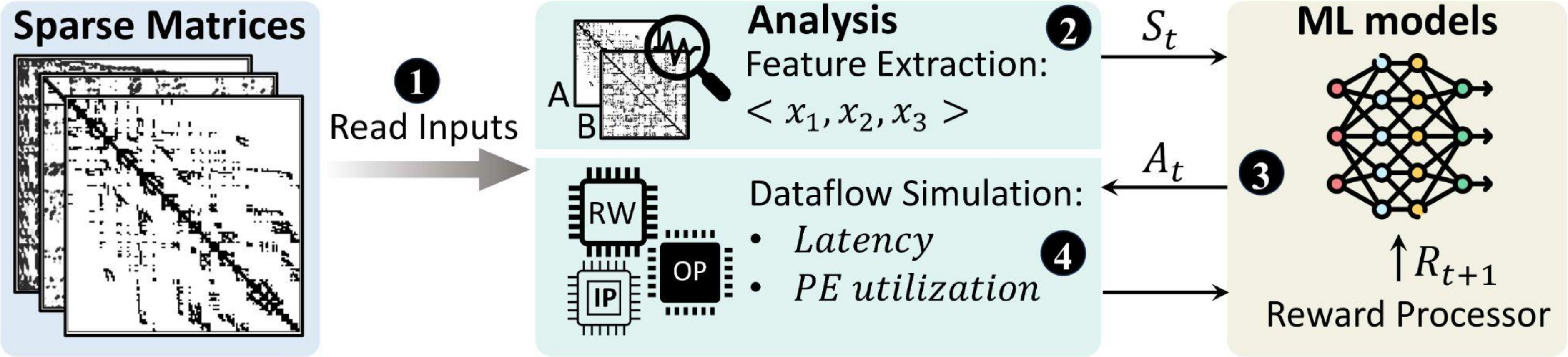
Illustrates the relationship between sparsity of input matrix A (x-axis), sparsity of input matrix B (size of the bubble), average number of nonzero per-row in matrix A (color depth) and the latency (total number of cycles).

Decision Tree & Reinforcement Learning Model



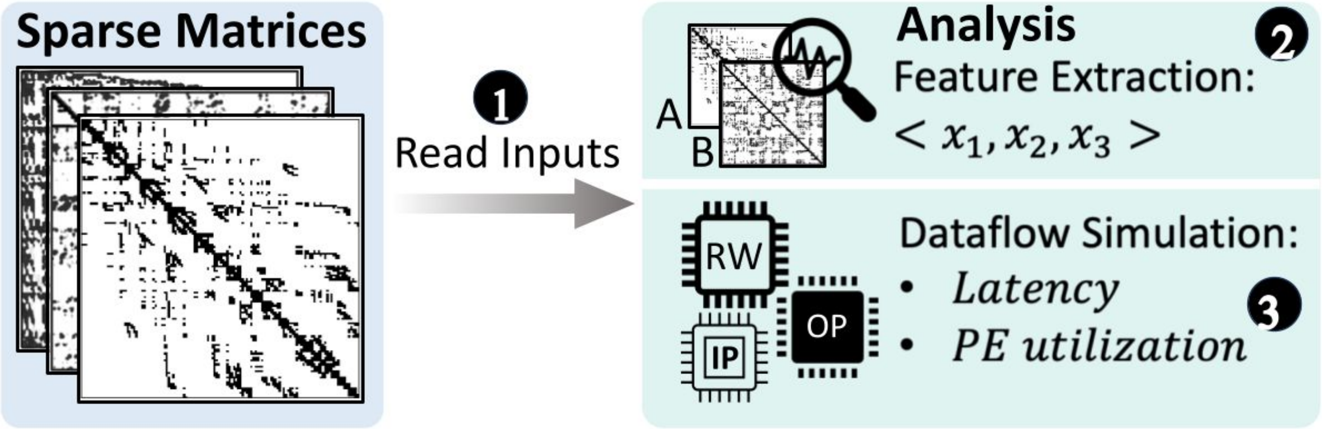


Misam Architecture

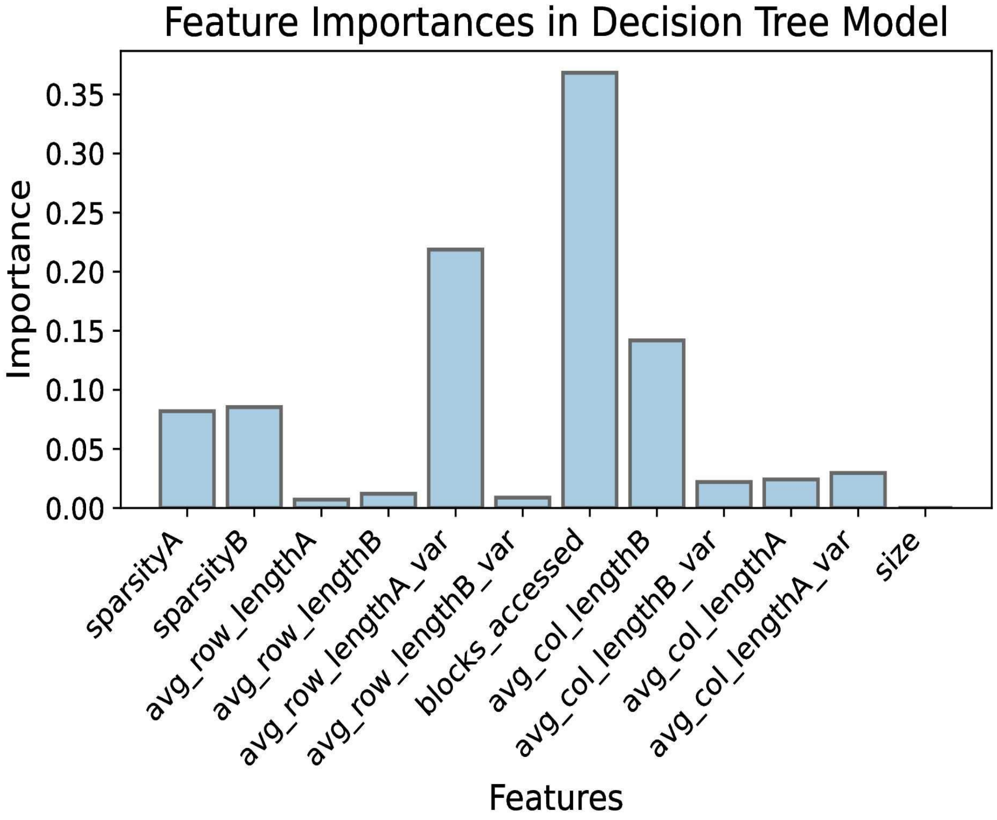


Dataset

- ***Dataset Development***: Created a dataset of 50K matrix multiplication simulations with matrices diverse dimensions and sparsity patterns.



Feature Selection



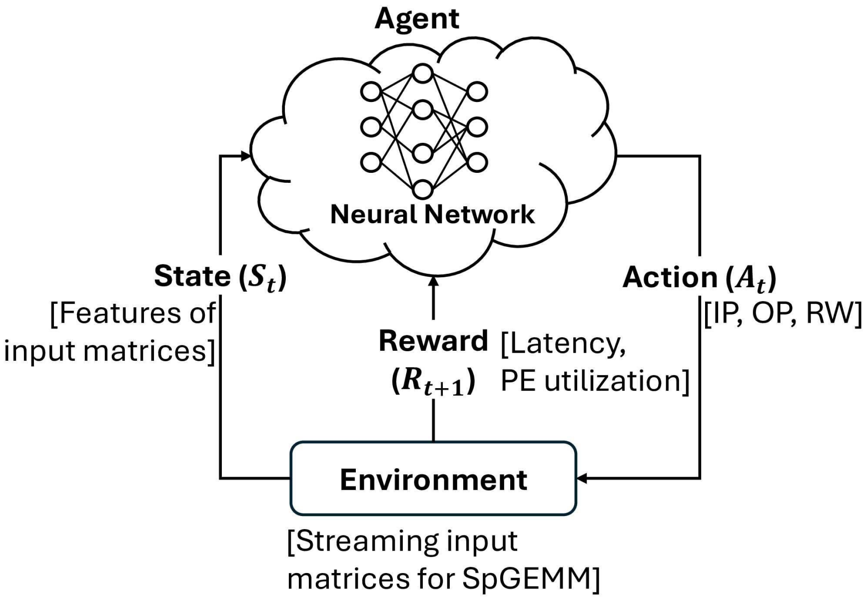
Feature Name	***Description***
sparsity of A & B	Total nonzero in matrix size of matrix
avg_row_length of A & B	Average number of nonzero per row
avg_col_length of A & B	Average number of nonzero per column
avg_row_length_var of A & B	Variance in average number of nonzero per row
avg_col_length_var of A & B	Variance in average number of nonzero per column
blocks_accessed of B	Blocks of rows accessed not in memory
size	Size of matrix block

Decision Tree Evaluation Methodology

- 70:30 dataset split for training and validation
- Select the top five features for our decision tree model as identified in feature selection model: *blocks accessed, avg_col_lengthB, avg_row_lengthA_var, sparsityA, and sparsityB*
- Used *sklearn* library to create the model

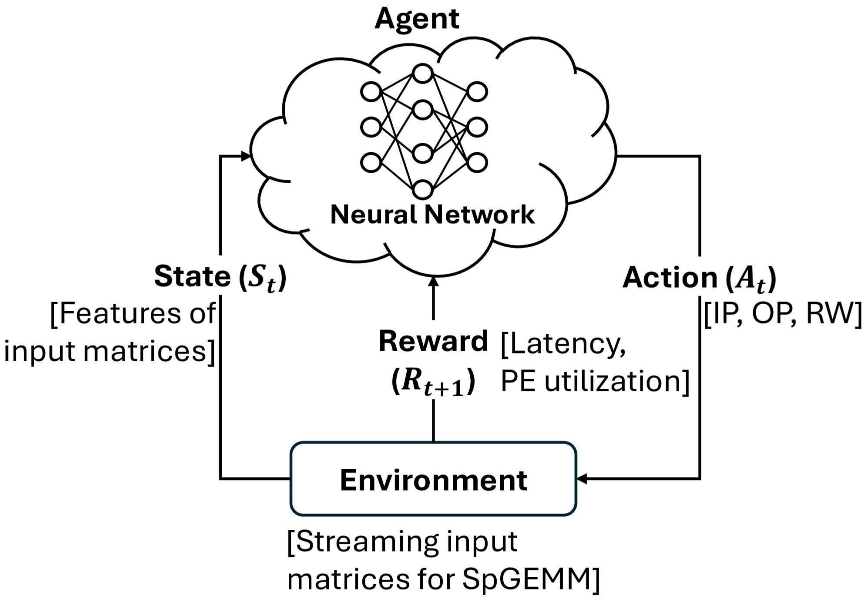
Reinforcement Learning Evaluation Methodology

- ***Environment***: Agent predicts the optimal dataflow scheme for processing streaming input matrices
- ***State***: [*blocks accessed, avg_col_lengthB, avg_row_lengthA_var, sparsityA, and sparsityB]*
- ***Action***: [IP, OP, RW]
- ***Reward***: +1 if action yields minimum latency



Reinforcement Learning Evaluation Methodology

- Neural network features just one hidden layer and contains 9,219 parameters
- Use validation set, similar to decision trees, for evaluation

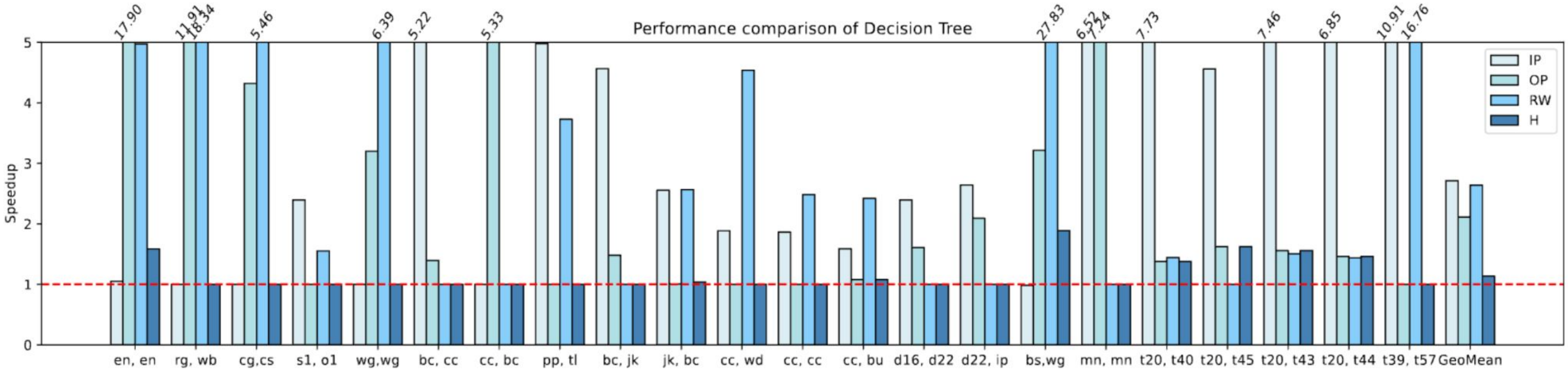


Decision-Tree-Guided Heuristic Evaluation Methodology

- Critical features are positioned at the top of the decision tree
- Created a decision tree, with two levels and transforming it into nested if-else statements

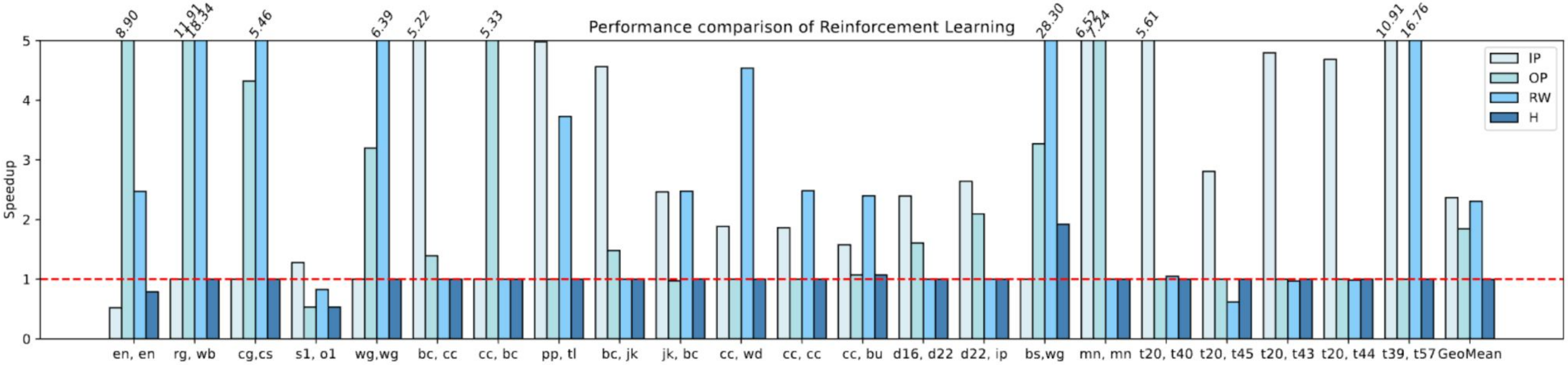
```
def heuristic(input):
   if blocks_accessed <= 0.03894999995827675:
    if avg_row_lengthA_var <= 0.00980000116229057:
      predicted.append('2')
    elif avg_row_lengthA_var > 0.00980000116229057:
      predicted.append('1')
  elif blocks_accessed > 0.03894999995827675:
    if blocks_accessed <= 0.04165000095963478:
      predicted.append('0')
    elif blocks_accessed > 0.04165000095963478:
      predicted.append('0')
```

Performance of Decision Tree Model



Average speedups of 2.7× over the IP, 2.1× over the OP, 2.64× over the RW, and 1.13× over the heuristic approach.

Performance of Reinforcement Learning Model



Average speedup of 2.36× over IP, 1.85× over OP, and 2.3× over RW. The heuristic performance was comparable to that of the RL model.

Storage Comparison

- Storage optimization in ML systems is essential for enhancing resource efficiency, minimizing energy consumption, and improving inference latency.

Model	***Storage Requirement***
Decision Tree	24KB
Reinforcement Learning Model	38KB
Decision-Tree-Guided Heuristic	512B

ML or Heuristic?

Aspect	***ML***	***Heuristic***
Performance Adaptability	Better equipped to adapt to system changes, especially with RL models featuring online learning.	Best suited for static systems that don't undergo significant change
Storage Requirements	Generally higher, varies by model complexity. RL model is around 38KB.	Lightweight around 512B
System Suitability	Ideal for dynamic systems	Ideal for stable systems
Feedback Incorporation	Can adapt by considering additional environmental parameters like PE utilization and system bandwidth.	Need to establish specific thresholds for each environmental parameter targeted for optimization.

Conclusions & Future Work

- **Misam** aimed to investigate the application of ML in selecting dataflows for SpGEMM. It explored three distinct approaches: decision trees, RL models, and heuristics.
    - This study pioneered in its examination of techniques to identify the most optimal dataflow in SpGEMM.
- **Future Direction**
    - Expand our dataset and develop a more sophisticated online reinforcement learning model that remains lightweight.
    - Comparison with Spada and Flexagon
    - Target deployment into a self-reconfigurable hardware system, where a lightweight ML model predicts the optimal dataflow based on the features of sparse matrices streaming from memory