Phi: Leveraging Pattern-based Hierarchical Sparsity for High-Efficiency Spiking Neural Networks

Chiyue Wei*, **Bowen Duan***, Cong Guo, Jingyang Zhang, Qingyue Song, Hai “Helen” Li, Yiran Chen

June 24th, 2025

Duke

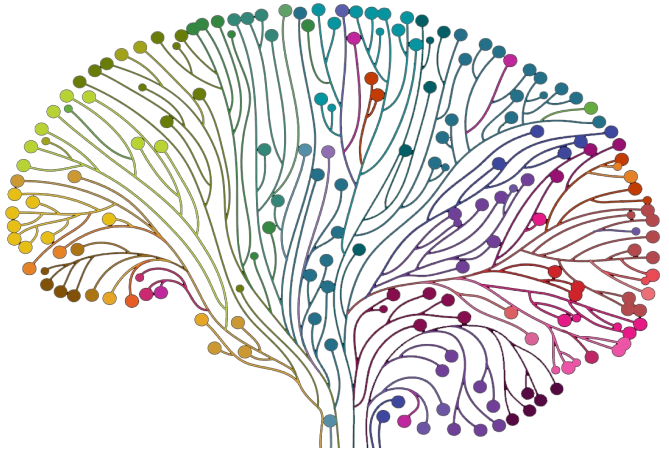
*Both author contributed equally to this work

Contents

Duke

Spiking Neural Networks (SNNs)

- Biologically inspired neural networks
- Various applications
- Higher energy efficiency



Robotics

Brain-computer Interface

Autonomous Driving

Duke

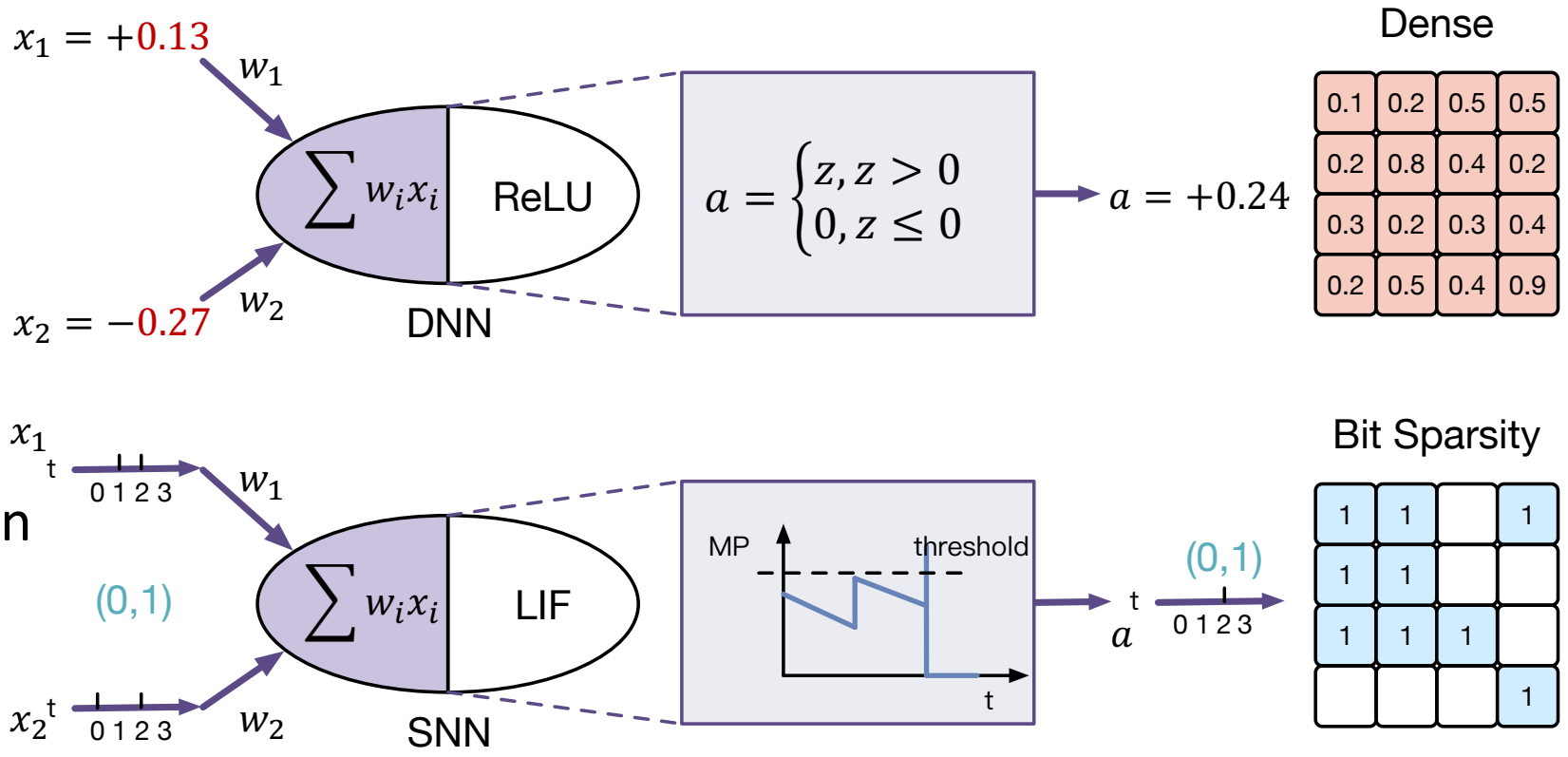
Comparison Between SNNs and DNNs

Similarity:

- Main Operand: Matrix multiplication

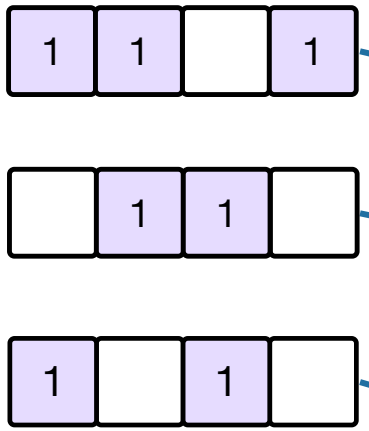
Difference:

- Single-bit spike activation
- Inherent Bit Sparsity



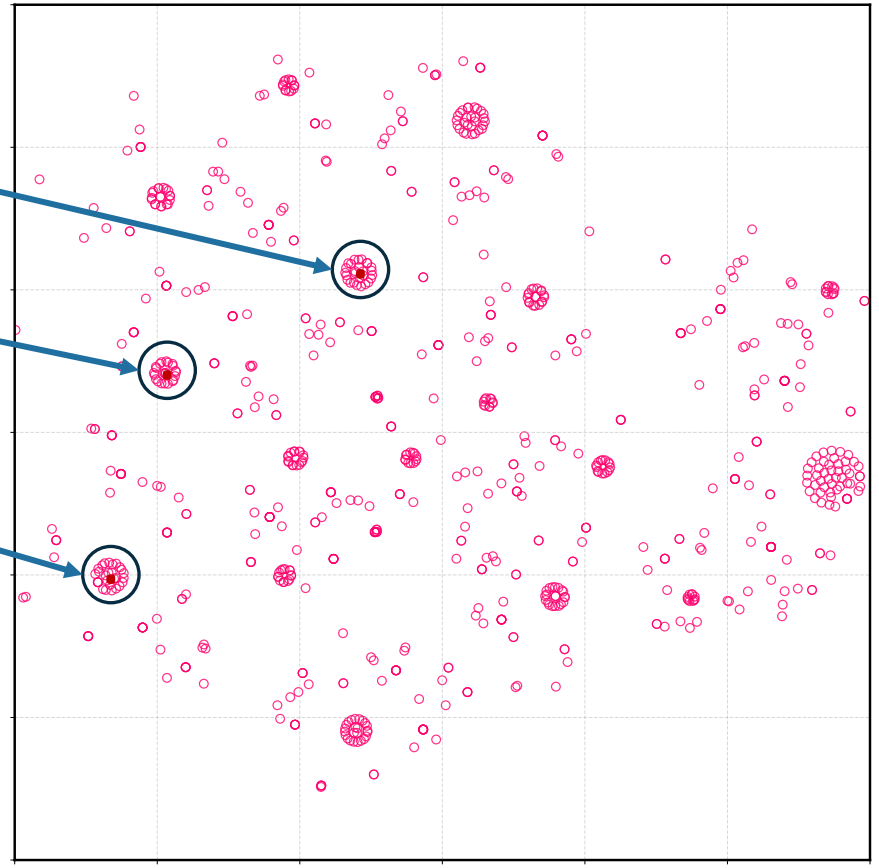
Distribution of Activations

- Regularity in SNNs activations
  - Patterns



  - Can we take advantage of these patterns?

SNN (0,1) Activation



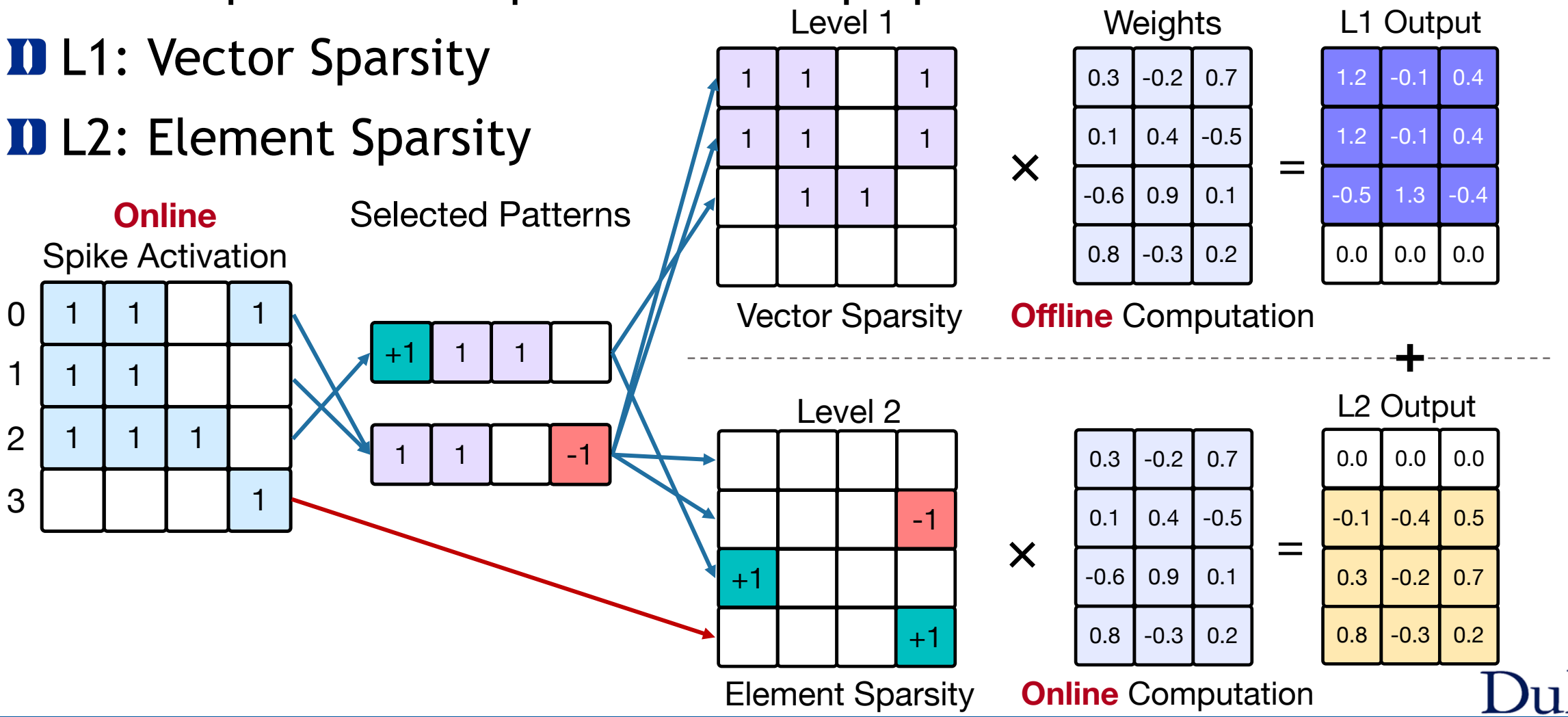
Contents

- Background & Motivation
- Pattern-based Hierarchical Sparsity
- Phi Architecture Design
- Evaluation & Discussion

Duke

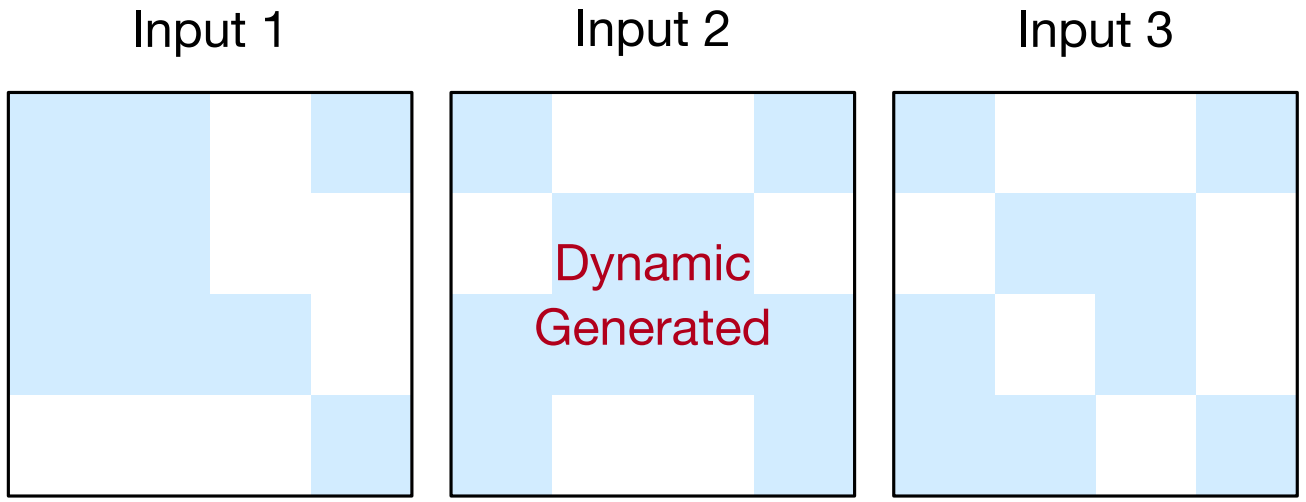
Pattern-based Hierarchical (Phi) Sparsity

- Utilize pre-defined patterns to skip operations
- L1: Vector Sparsity
- L2: Element Sparsity

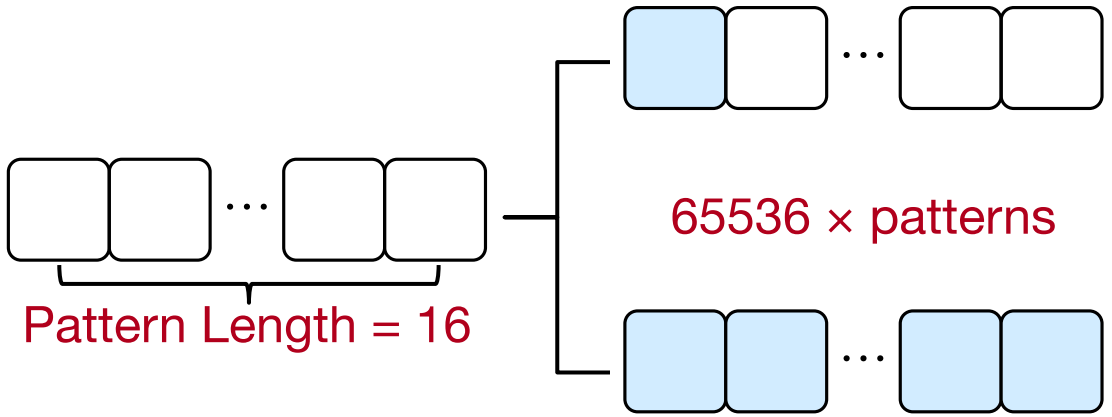


Challenges: Algorithm

- Uncertainty of activations



- Complexity of patterns

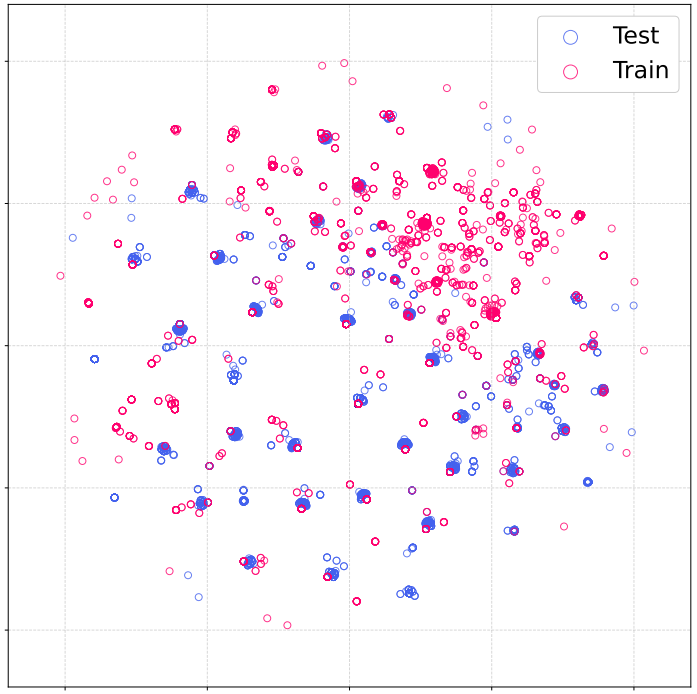


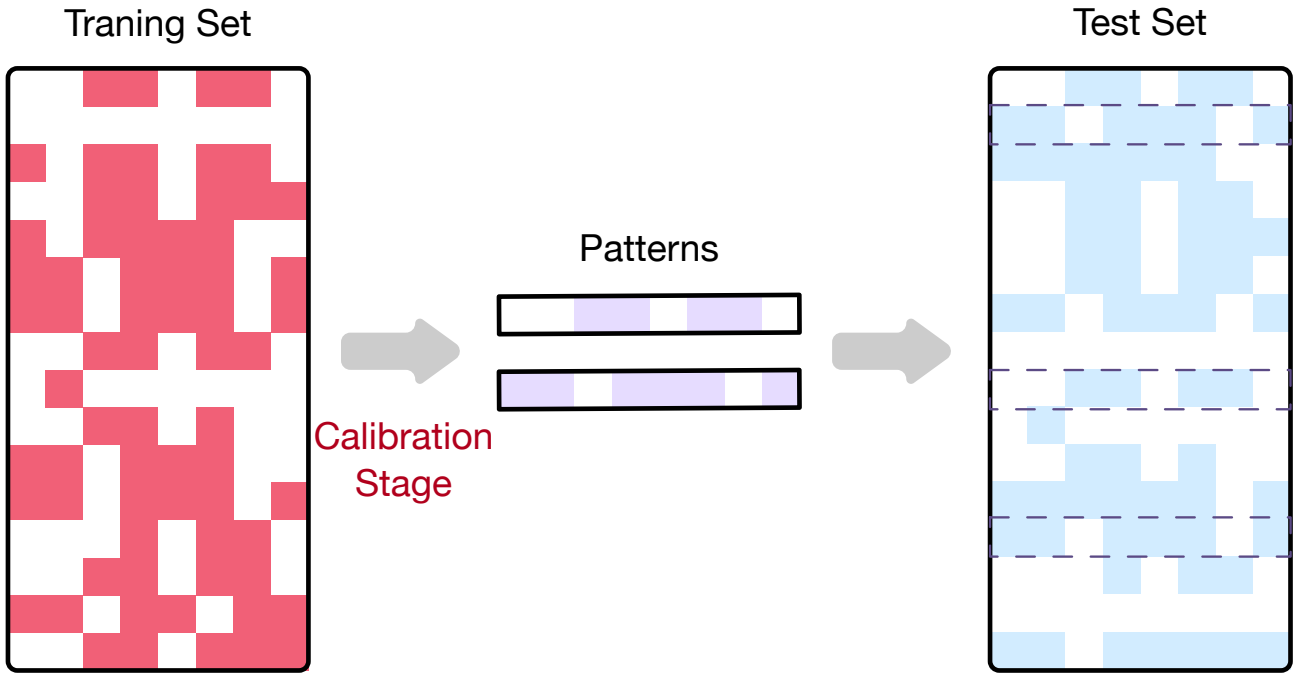
- How to select patterns?

Duke

Phi Sparsity Calibration Set

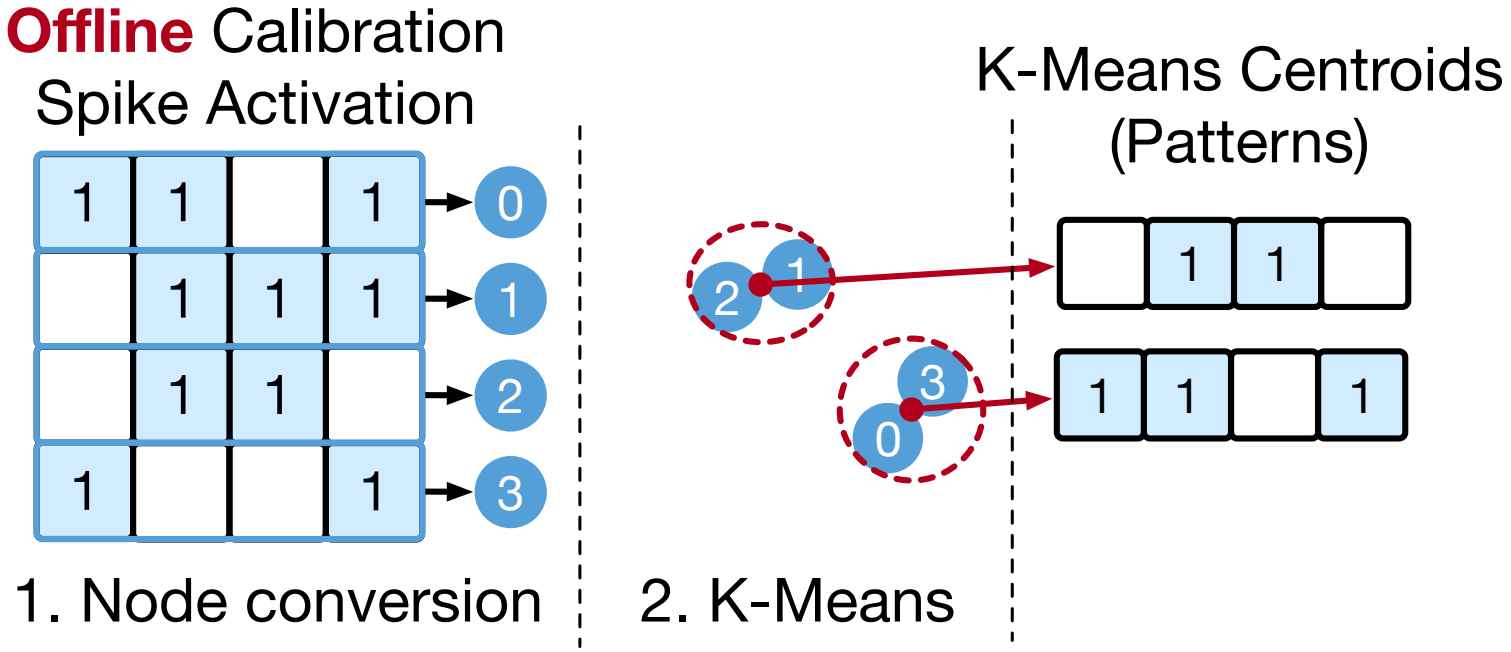
- Similarity between training set and test set
  - Use a calibration set to identify patterns





Phi Sparsity Generation Stage

- K-means-based clustering algorithm
  - Filter out all zero and one-hot rows
  - Use Hamming distance



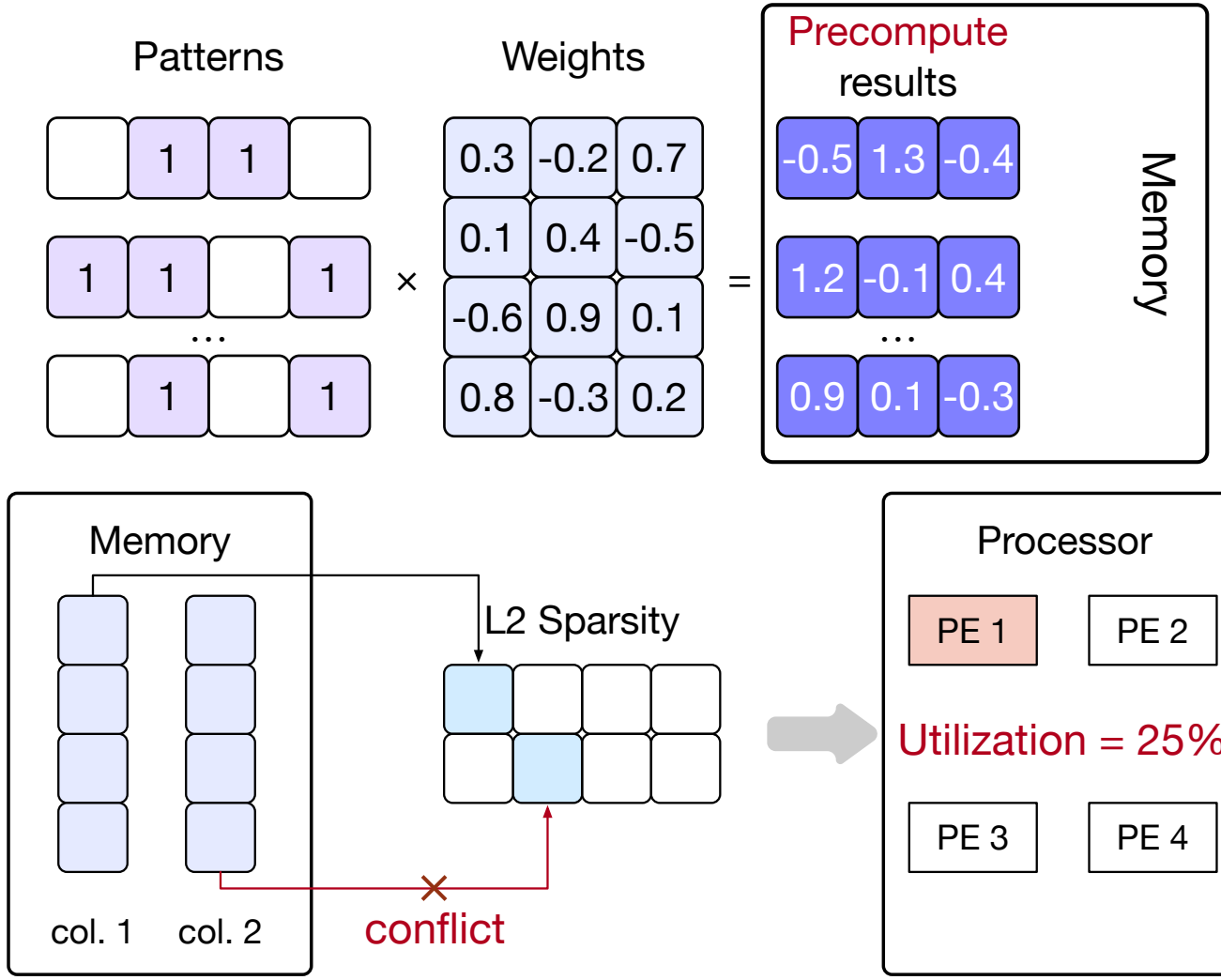
Contents

- Background & Motivation
- Pattern-based Hierarchical Sparsity
- Phi Architecture Design
- Evaluation & Discussion

Duke

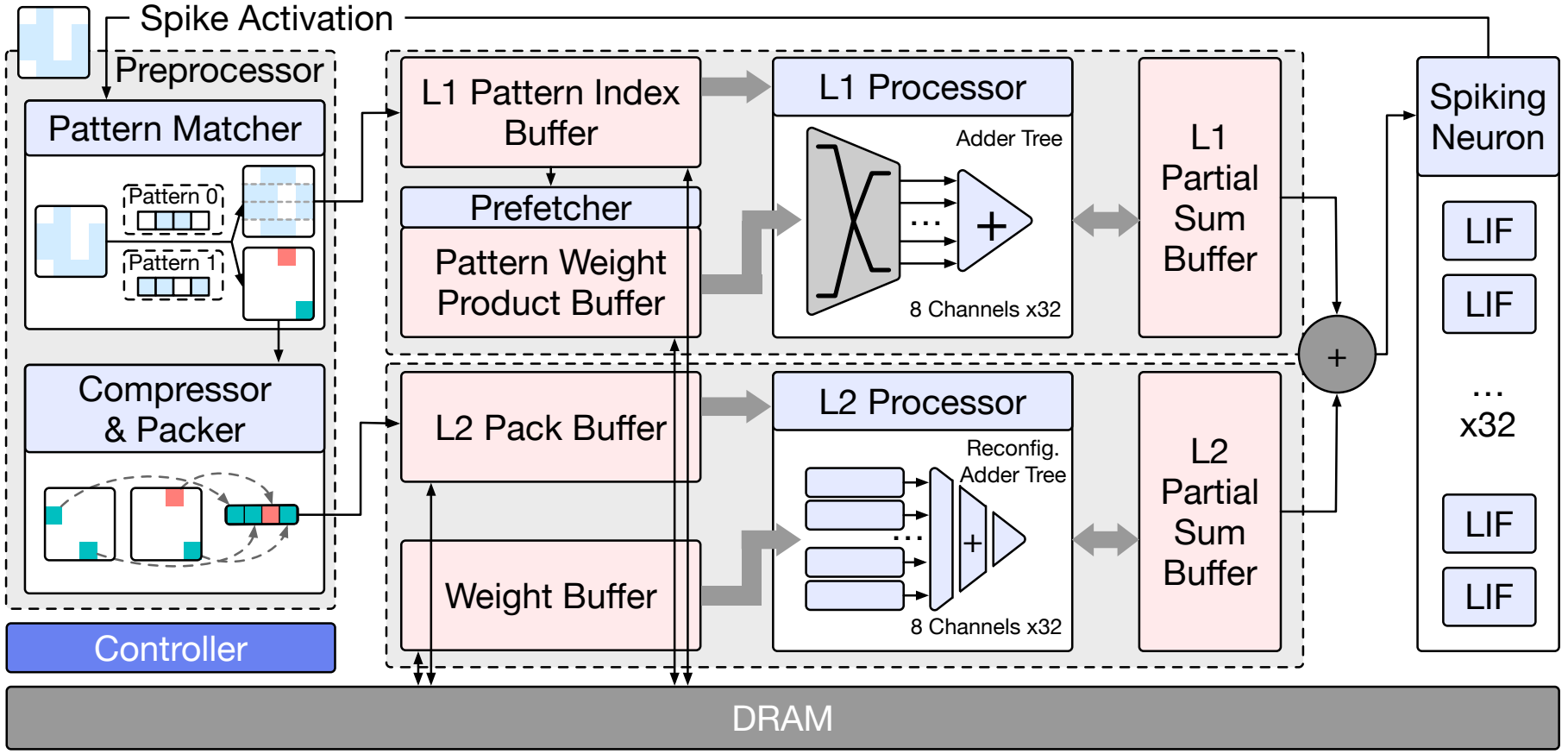
Challenges: Hardware

- Memory Overhead
  - 9x memory access
- Sparsity Processing
  - Data read/write conflicts
  - Low hardware utilization



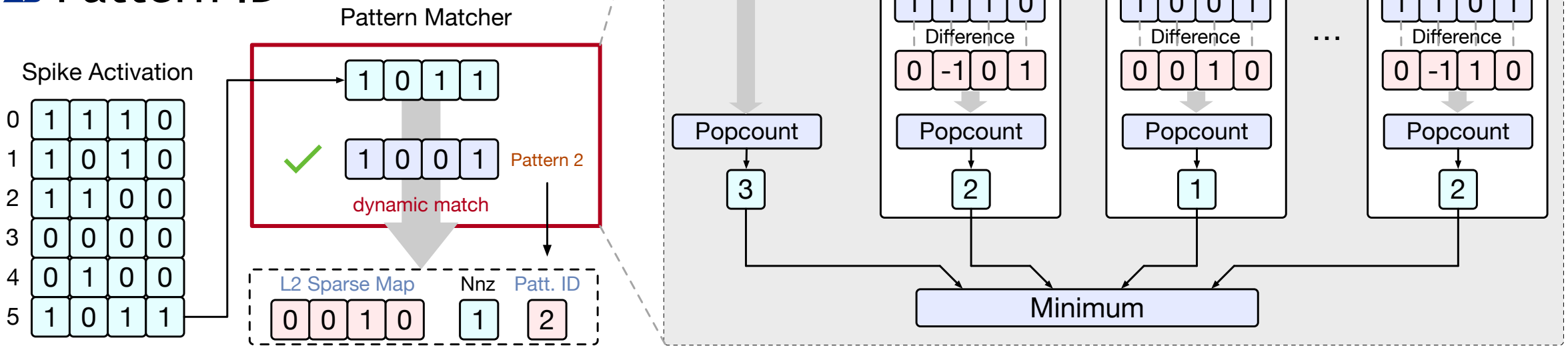
Architecture Overview

- An architecture generates and processes two levels of Phi sparsity
- Main components: Preprocessor, L1 processor, L2 processor



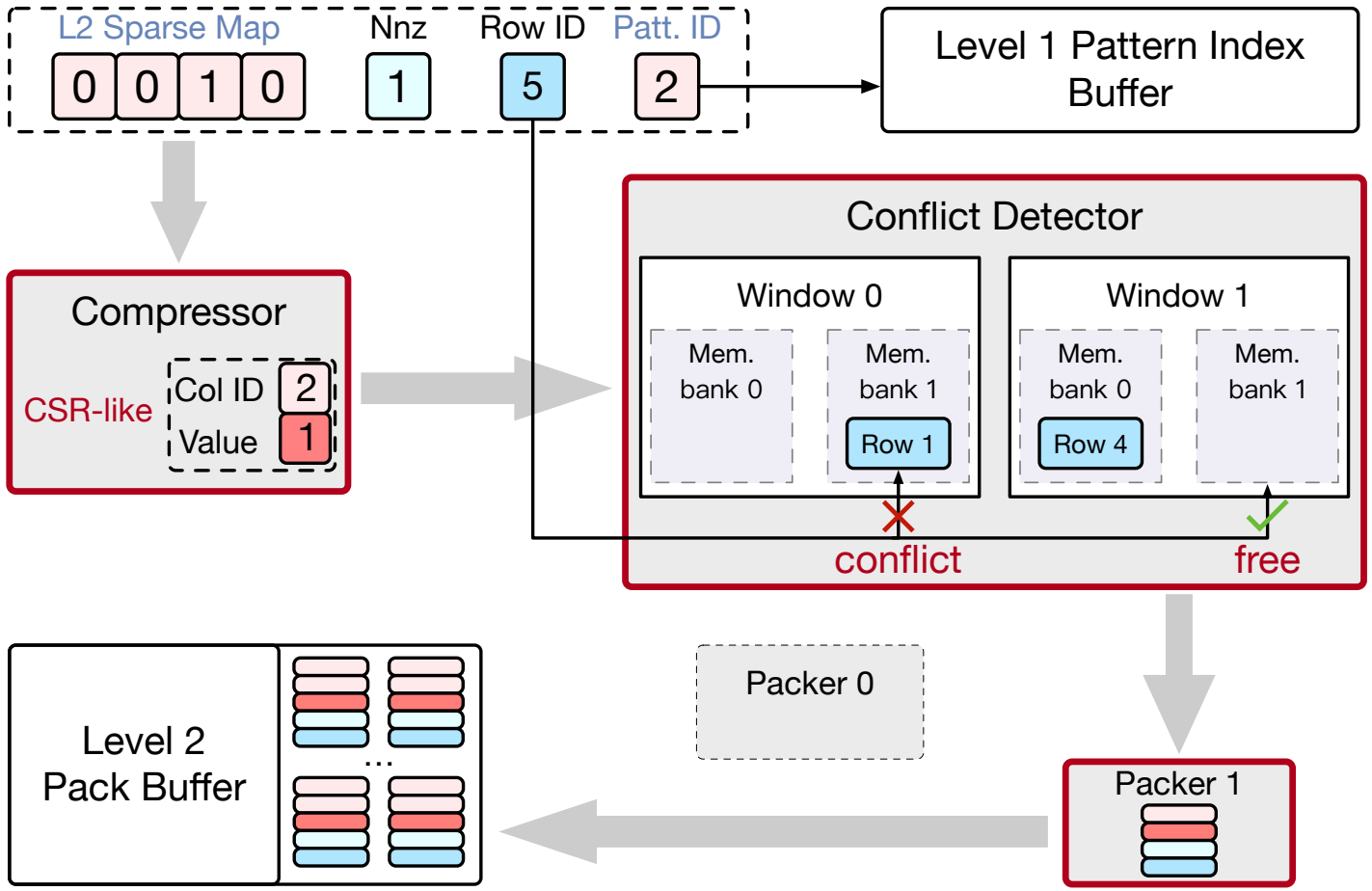
Pattern Matcher

- Dynamic matching pattern
- Get L2 sparse map
- Pattern ID



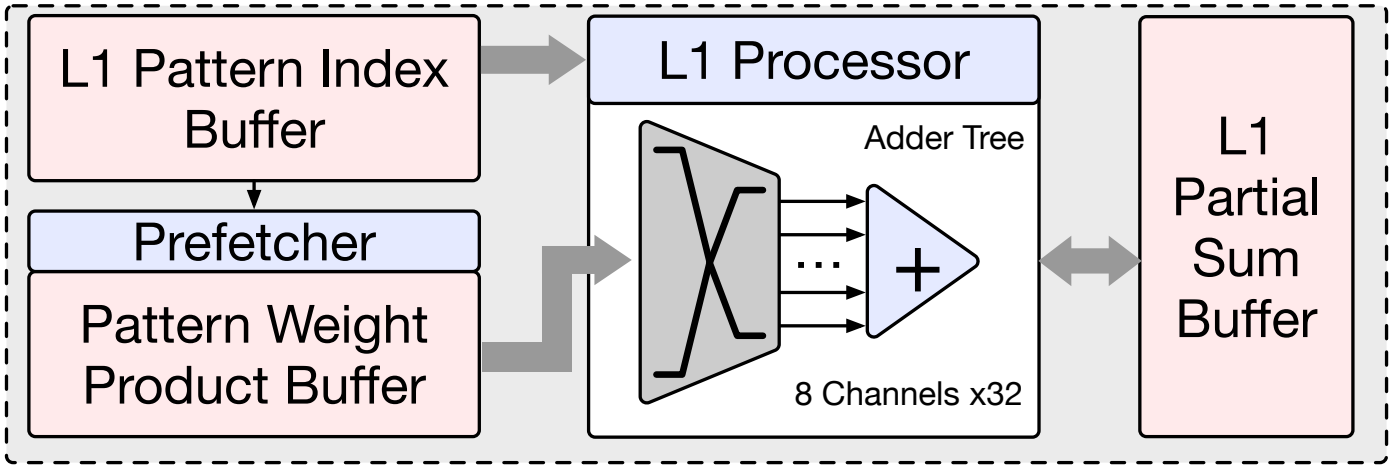
Compressor & Packer

- CSR-like format
  - Reduce memory access
- Multi window packing
  - Conflict free



Computation: structured sparsity

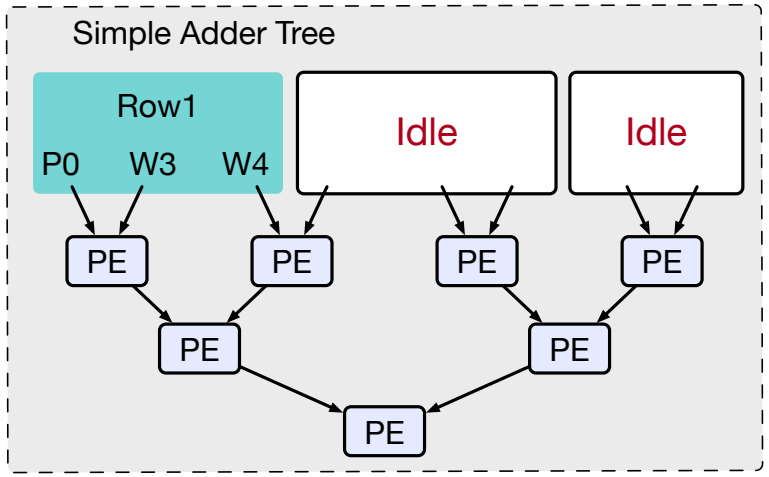
- Prefetcher
  - Only read the necessary PWPs
  - Reduce memory traffic
- L1 Processor
  - 16:8 structured sparsity
  - 8 channels adder tree



Computation: unstructured sparsity

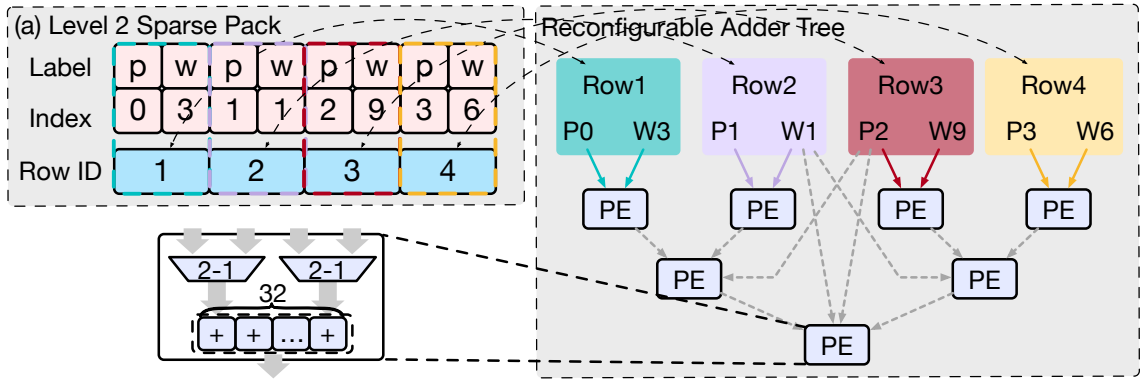
Reconfigurable Adder Tree

- High utilization
- Minimal overhead

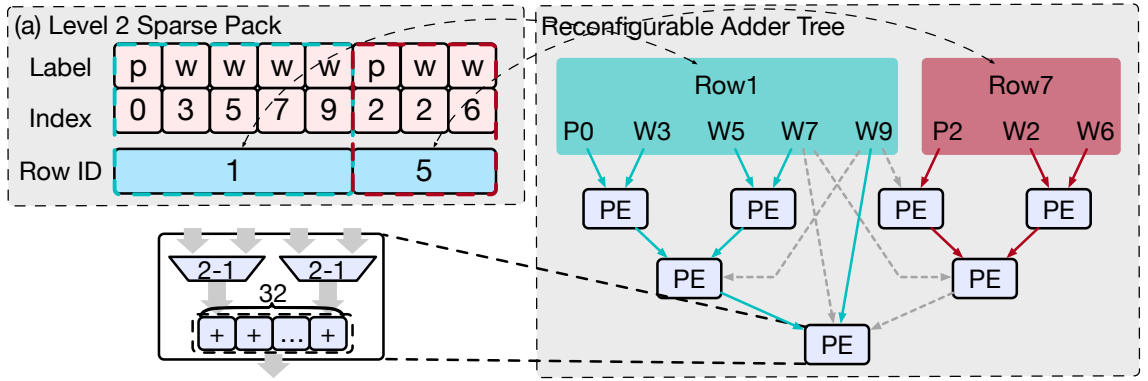


Simple adder tree

Utilization = 37.5%



Input packed in 2-2-2-2



Input packed in 5-3

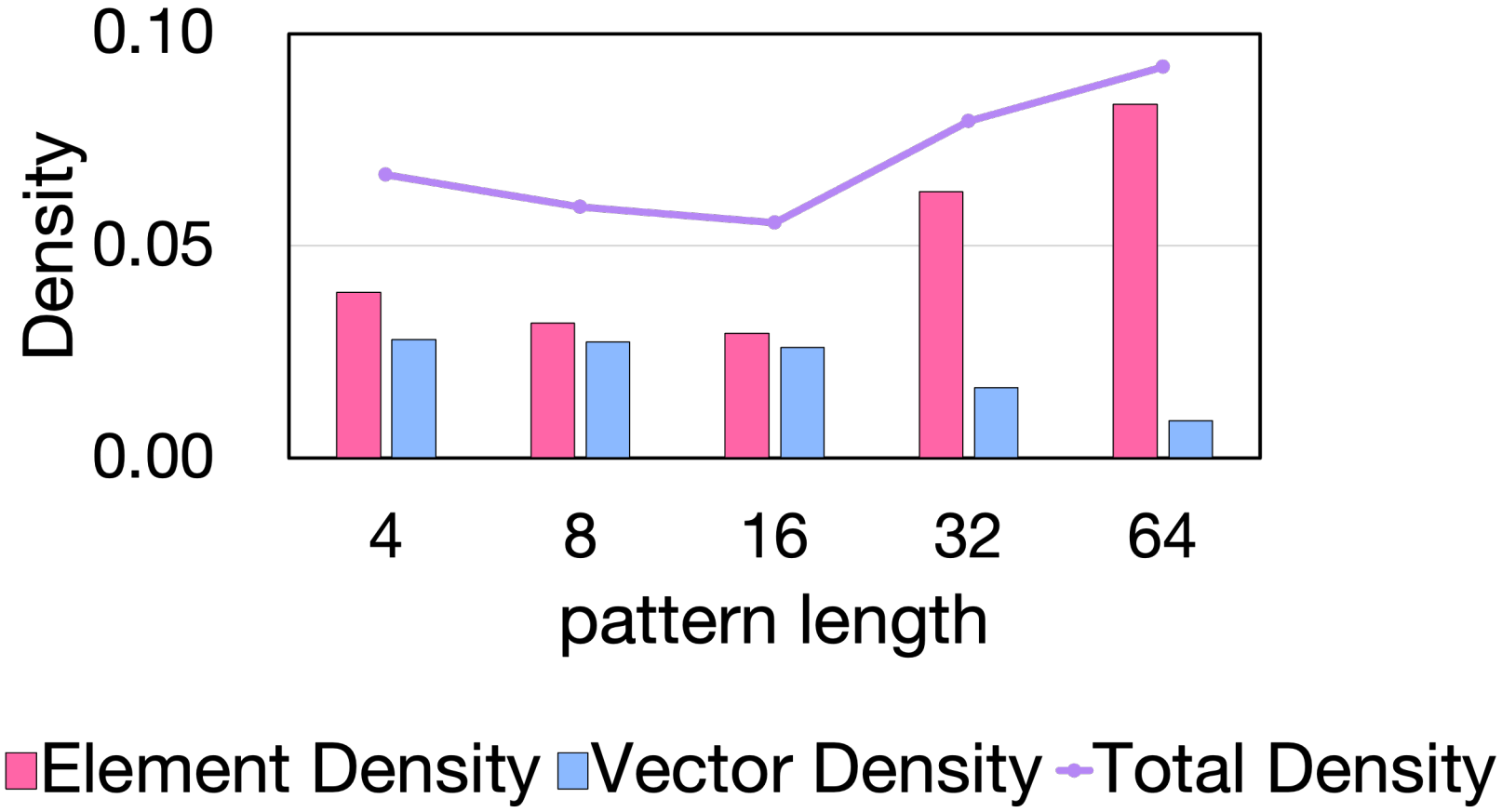
Contents

- Background & Motivation
- Pattern-based Hierarchical Sparsity
- Phi Architecture Design
- Evaluation & Discussion

Duke

Design Space Exploration

- Pattern length
- Balance L1 & L2 density
- Minimize total density
- L = 16

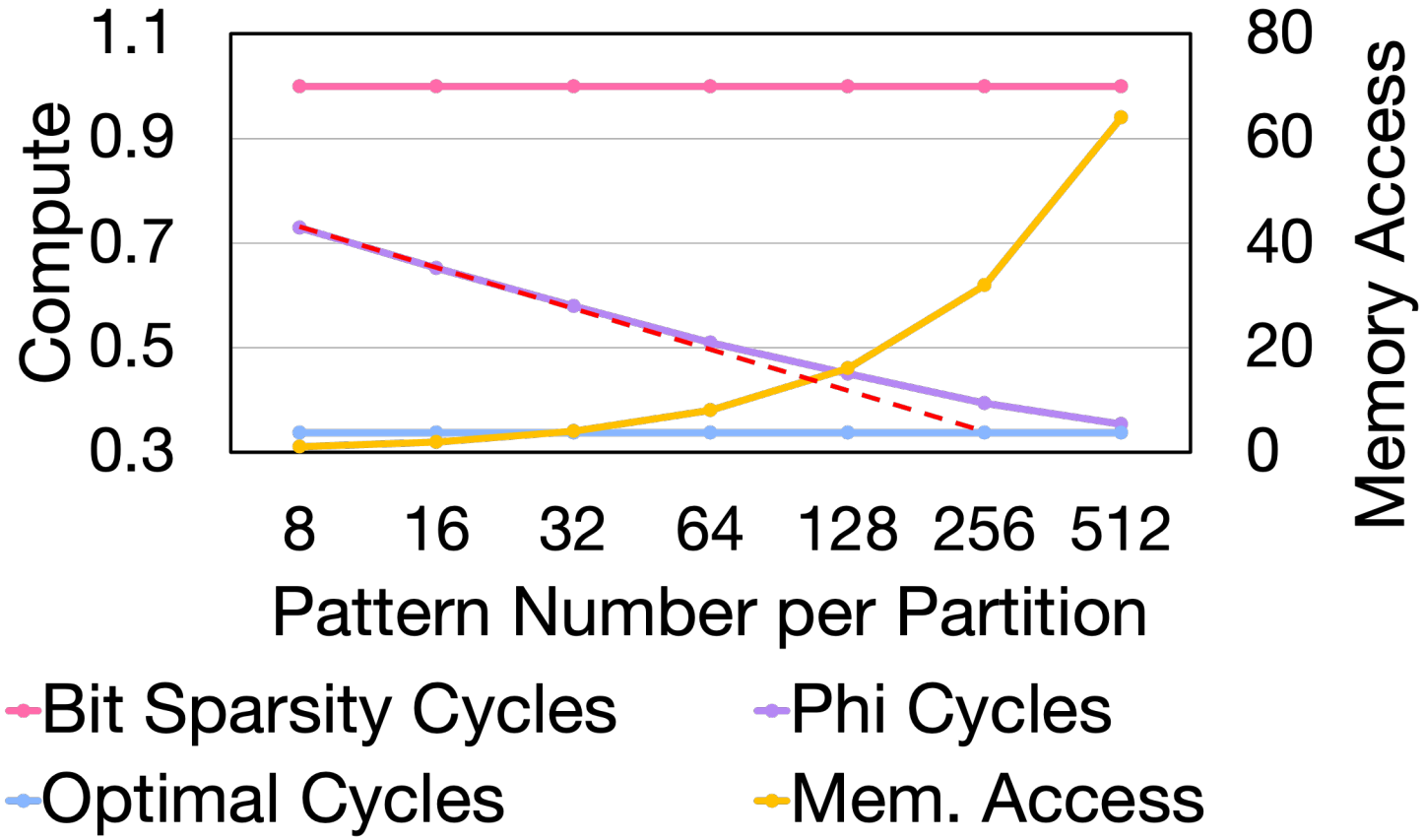


(a) Density with the pattern length.

Duke

Design Space Exploration

- Pattern number
- Balance computation cycles and memory overhead
- #patterns = 128



(b) Compute cycles with # patterns.

Duke

Evaluation Setup

Baselines	Sparsity
Spiking Eyeriss (ISCA'20)	Dense
SpinalFlow (ISCA'20)	Bit Sparsity
PTB (HPCA'22)	Bit Sparsity
SATO (DAC'22)	Bit Sparsity
Stellar (HPCA'24)	Bit Sparsity
Phi (Ours)	Phi Sparsity

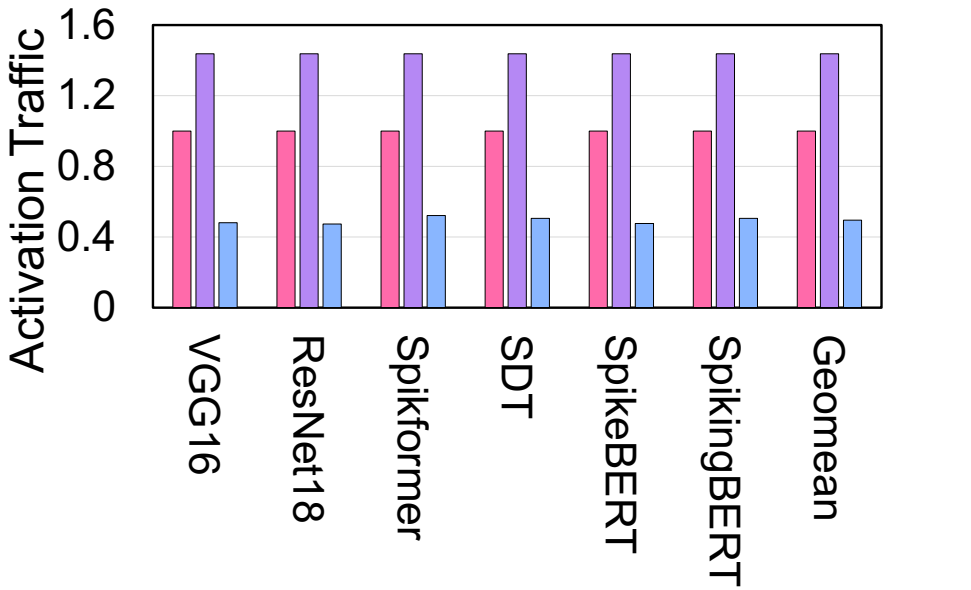
Tools	
Synopsys Design Compiler	Logic synthesis
CACTI	Buffer simulation
DRAMsim3	DRAM simulation

Type	Models	Data set
Spiking CNN	VGG16	Cifar10, Cifar100, Cifar10dvs
	ResNet18	
Spiking Transformer	Spikformer	
	SDT	
	SpikingBERT	SST-2, SST-5, MNLI
	SpikeBERT	

Duke

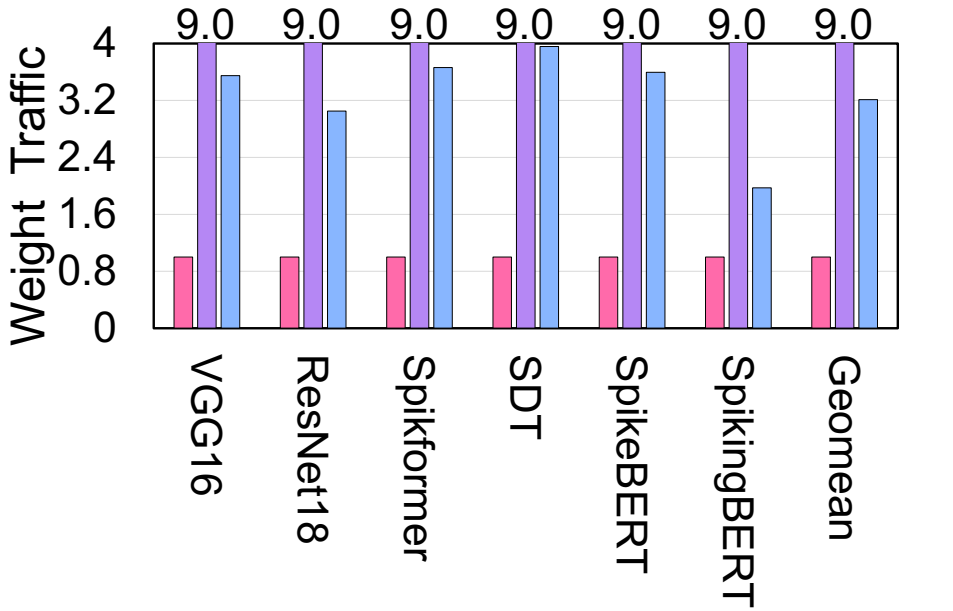
Memory Traffic Reduction

Compact data structure



(a) Activation memory traffic (normalized by dense).

Pattern prefetcher

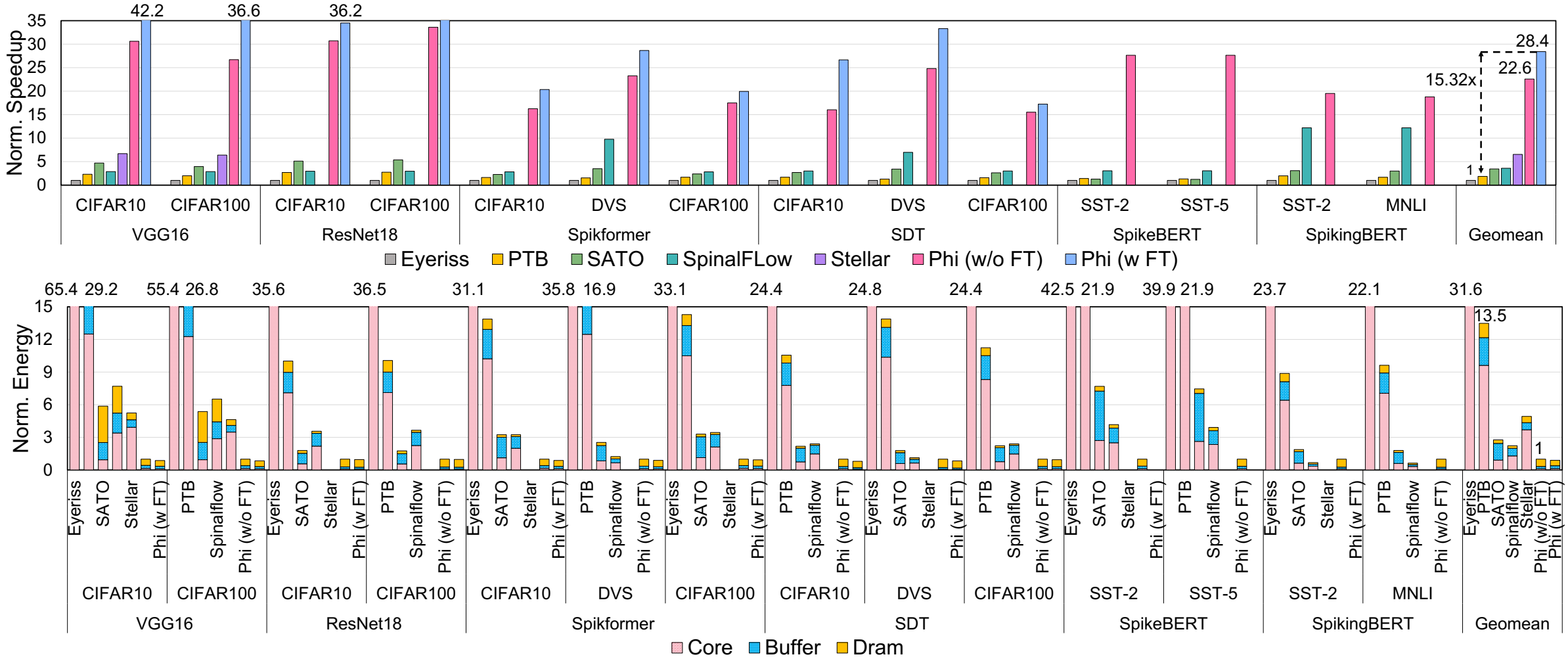


(b) Weight memory traffic (normalized by dense).

Duke

Speedup and Energy

- 3.45× speedup, 4.93× energy efficiency compared to Stellar



Conclusion

- **Phi Sparsity**: A novel hierarchical sparsity framework for SNNs that leverages structural patterns in SNN activations to generate two levels of sparsity.

- Our proposed algorithm: addresses key challenges through k-means-based pattern selection and pattern-aware fine-tuning.

- Our proposed architecture: **Phi**, generates and processes both levels of sparsity on the fly, achieves performance improvements compared to existing state-of-the-art accelerators.

Duke

Thanks & QA