

Accelerating Training of DNNs via Sparse Edge Processing

Sourya Dey, Yinan Shao, Keith Chugg, Peter Beerel

ICANN, September 2017



USCUniversity of
Southern California

Overview of Current DNNs

- ▶ Key machine learning technologies

Overview of Current DNNs

- ▶ Key machine learning technologies
- ▶ Lot of parameters - **Memory intensive**
- ▶ Slow to train - **Computationally intensive**

Overview of Current DNNs

- ▶ Key machine learning technologies
- ▶ Lot of parameters - **Memory intensive**
- ▶ Slow to train - **Computationally intensive**
- ▶ Training done **offline** in CPU/GPU
- ▶ Custom hardware used for **inference only**

Highlights of our Research

- ▶ Predefined sparsity - **Memory friendly**
 - ▶ *30x less parameters in FC layers*

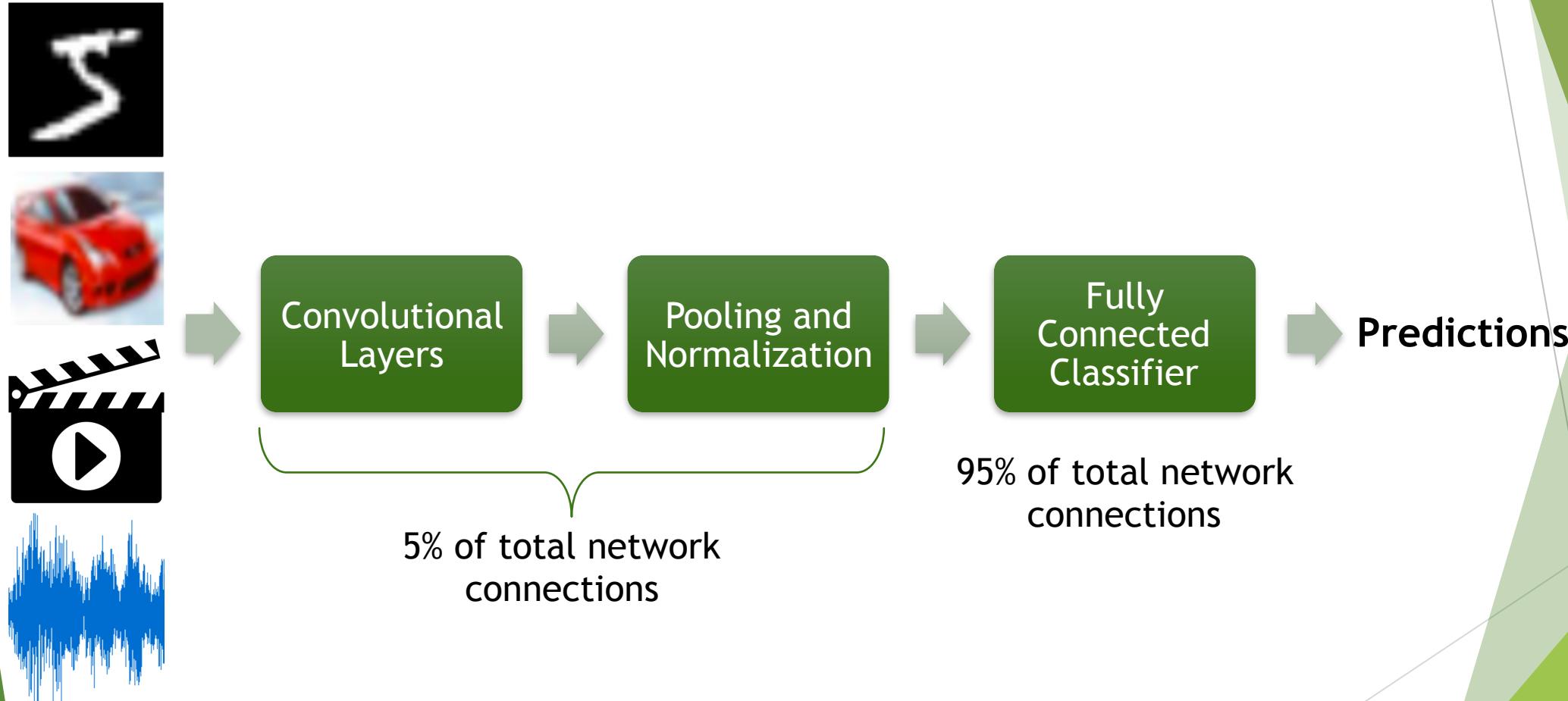
Highlights of our Research

- ▶ Predefined sparsity - **Memory friendly**
 - ▶ *30x less parameters in FC layers*
- ▶ Edge-based processing - **Computationally flexible**
- ▶ Hardware optimizations - **Fast training**
 - ▶ *35x estimated speedup over GPUs*

Highlights of our Research

- ▶ Predefined sparsity - **Memory friendly**
 - ▶ *30x less parameters in FC layers*
- ▶ Edge-based processing - **Computationally flexible**
- ▶ Hardware optimizations - **Fast training**
 - ▶ *35x estimated speedup over GPUs*
- ▶ FPGA based architecture - **Online training** and inference

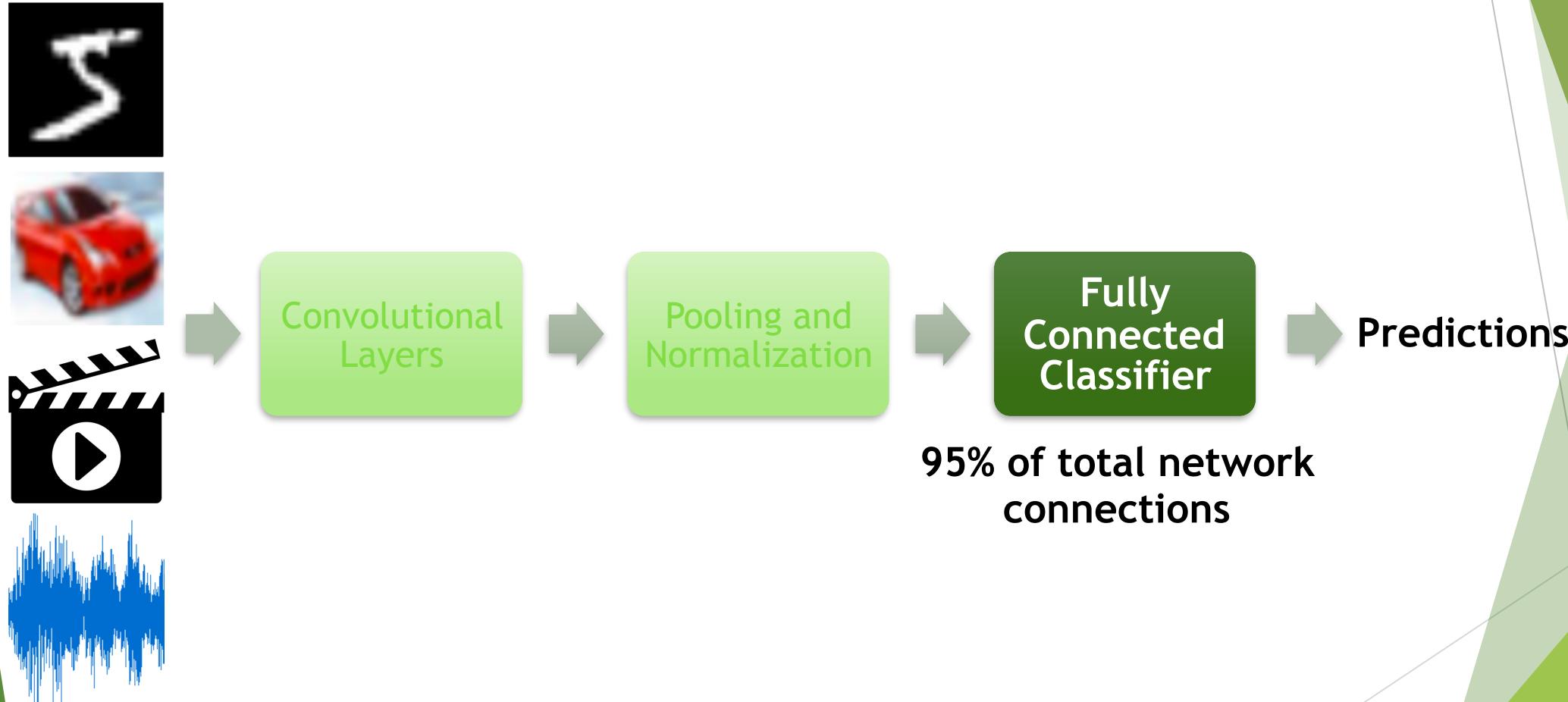
Typical Supervised Network



Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS-2012, pp. 1097-1105 (2012)

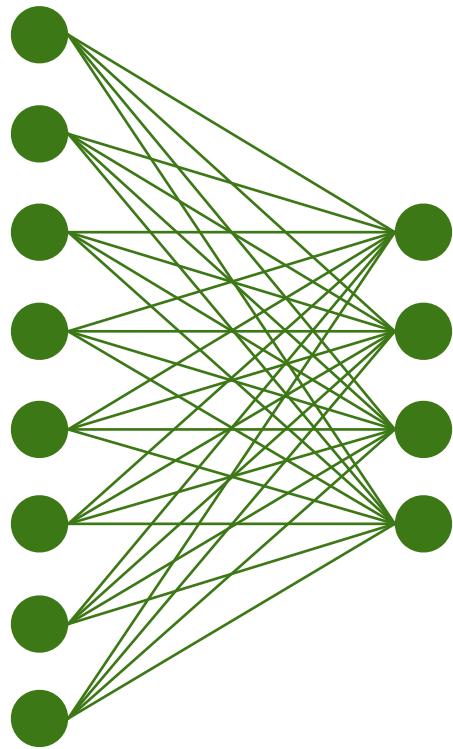
Zhang, C., Wu, D., Sun, J., Sun, G., Luo, G., Cong, J.: Energy-efficient CNN implementation on a deeply pipelined FPGA cluster. In: ISLPED-2016. pp. 326- 331. ACM, New York (2016)

Focus of the Present Work



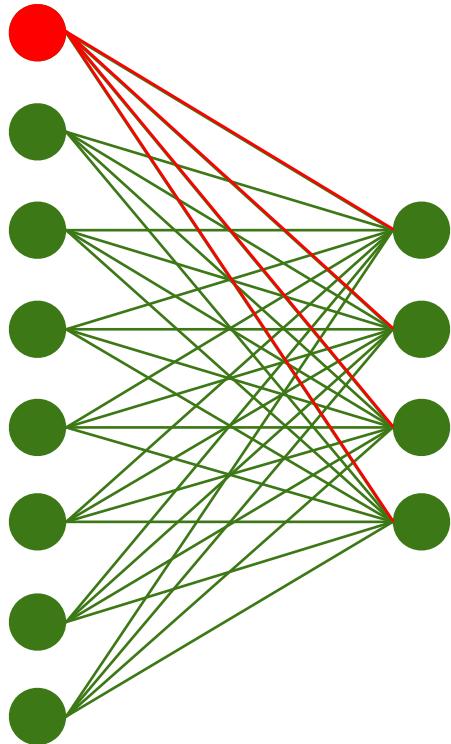
Sparsity

Sparsity



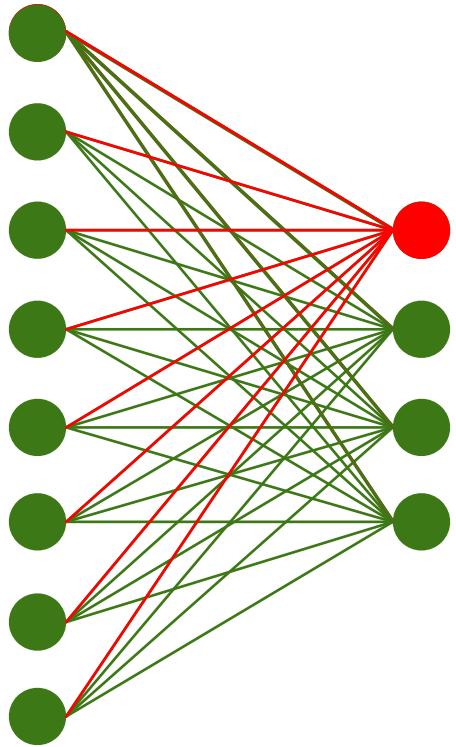
Fully connected (FC) network

Sparsity



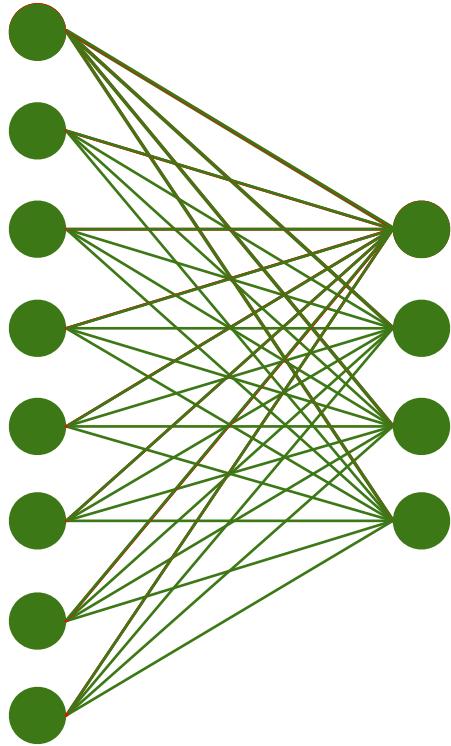
Fully connected (FC) network
Fanout (fo) = 4

Sparsity



Fully connected (FC) network
Fanout (fo) = 4 Fanin (fi) = 8

Sparsity

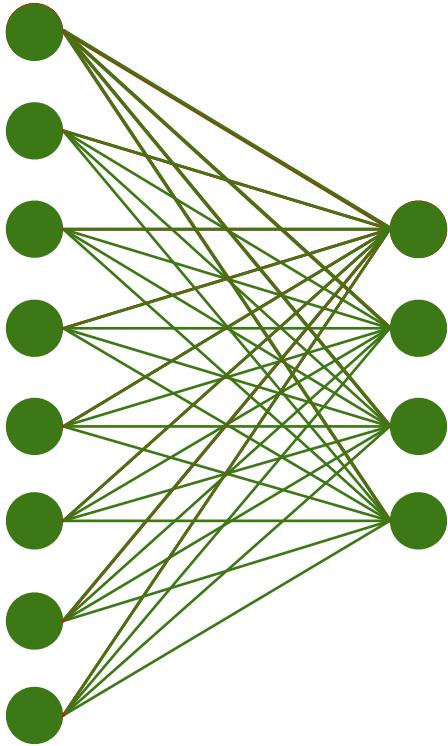


Fully connected (FC) network

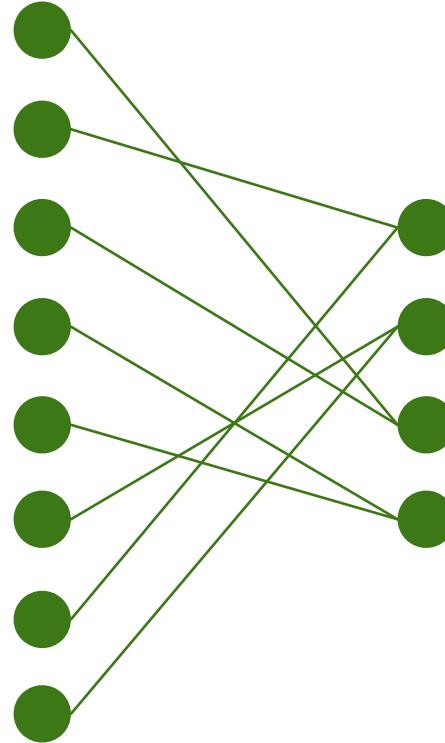
Fanout (fo) = 4 Fanin (fi) = 8

Connectivity = 100%

Sparsity

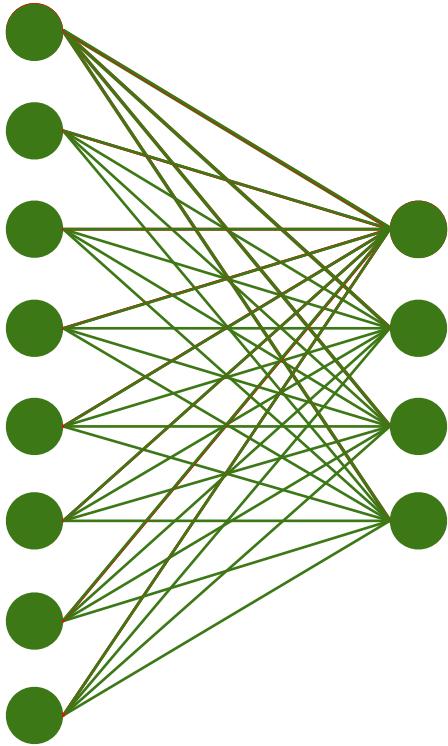


Fully connected (FC) network
Fanout (fo) = 4 Fanin (fi) = 8
Connectivity = 100%

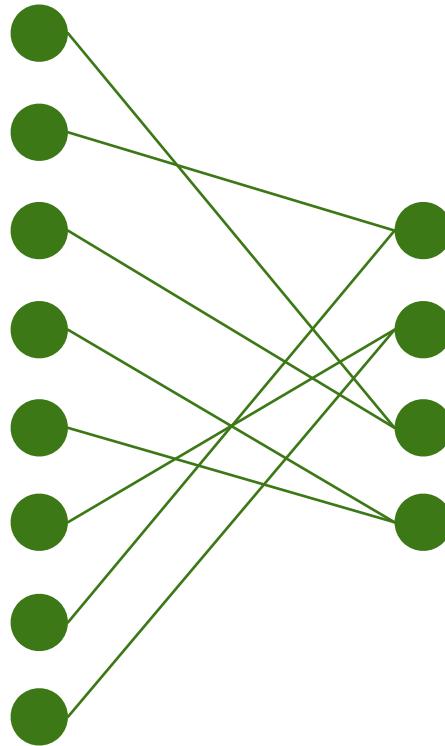


Sparse network
 $fo = 1, fi = 2$
Connectivity = 25%

Sparsity - Predefined

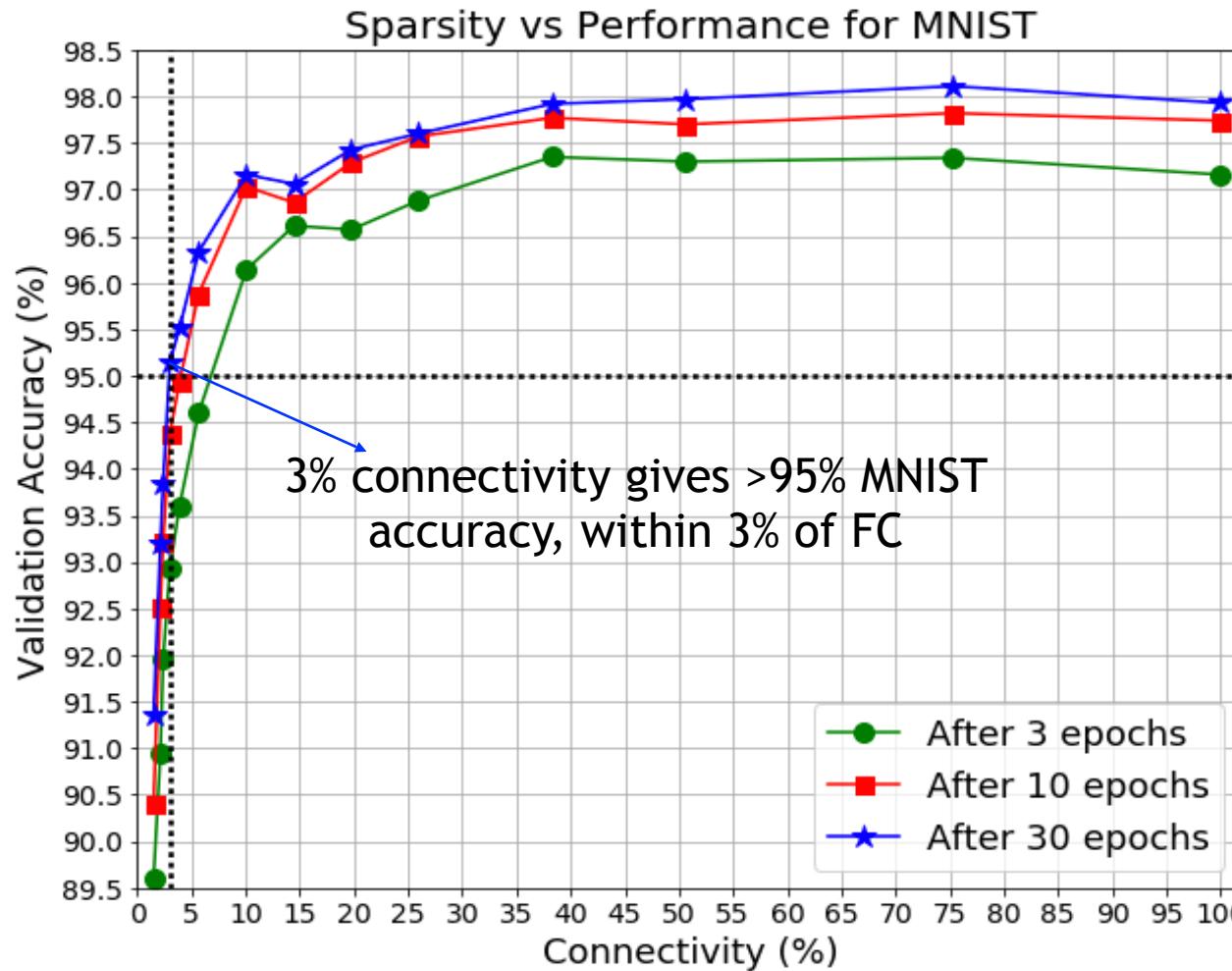


Fully connected (FC) network
Fanout (fo) = 4 Fanin (fi) = 8
Connectivity = 100%

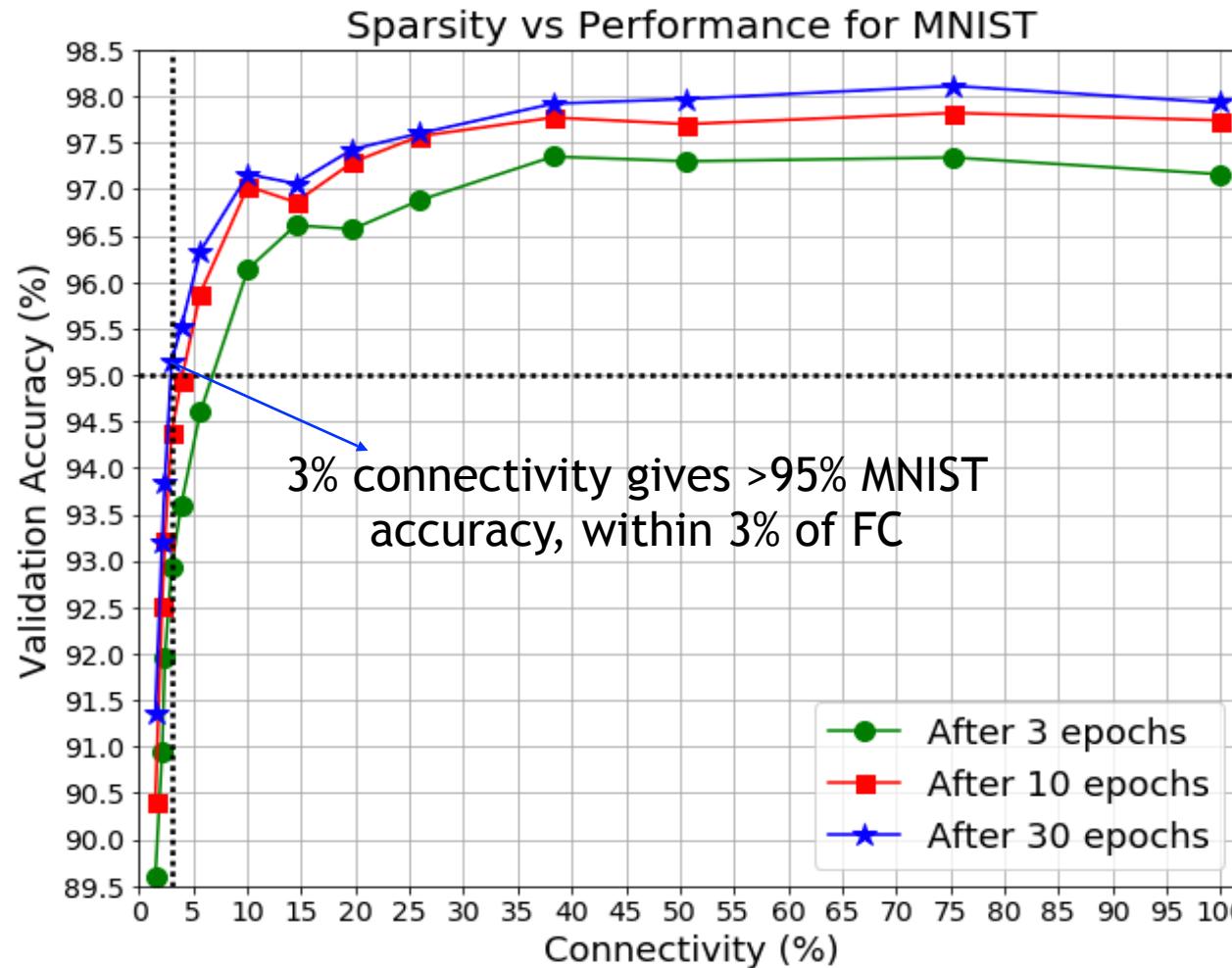


Sparse network
 $fo = 1, fi = 2$
Connectivity = 25%

Does predefined sparsity work?

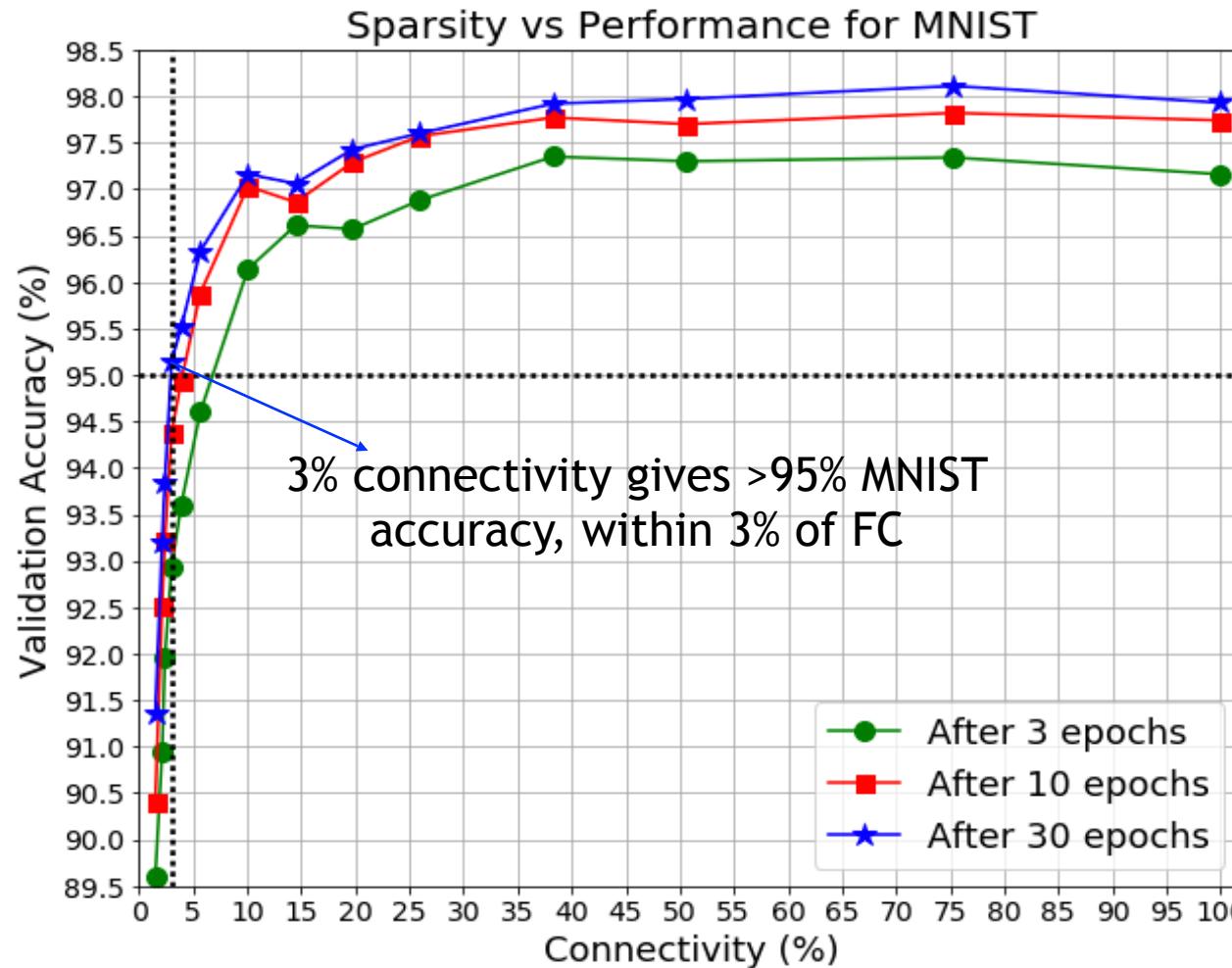


Does predefined sparsity work?



Ongoing research shows:

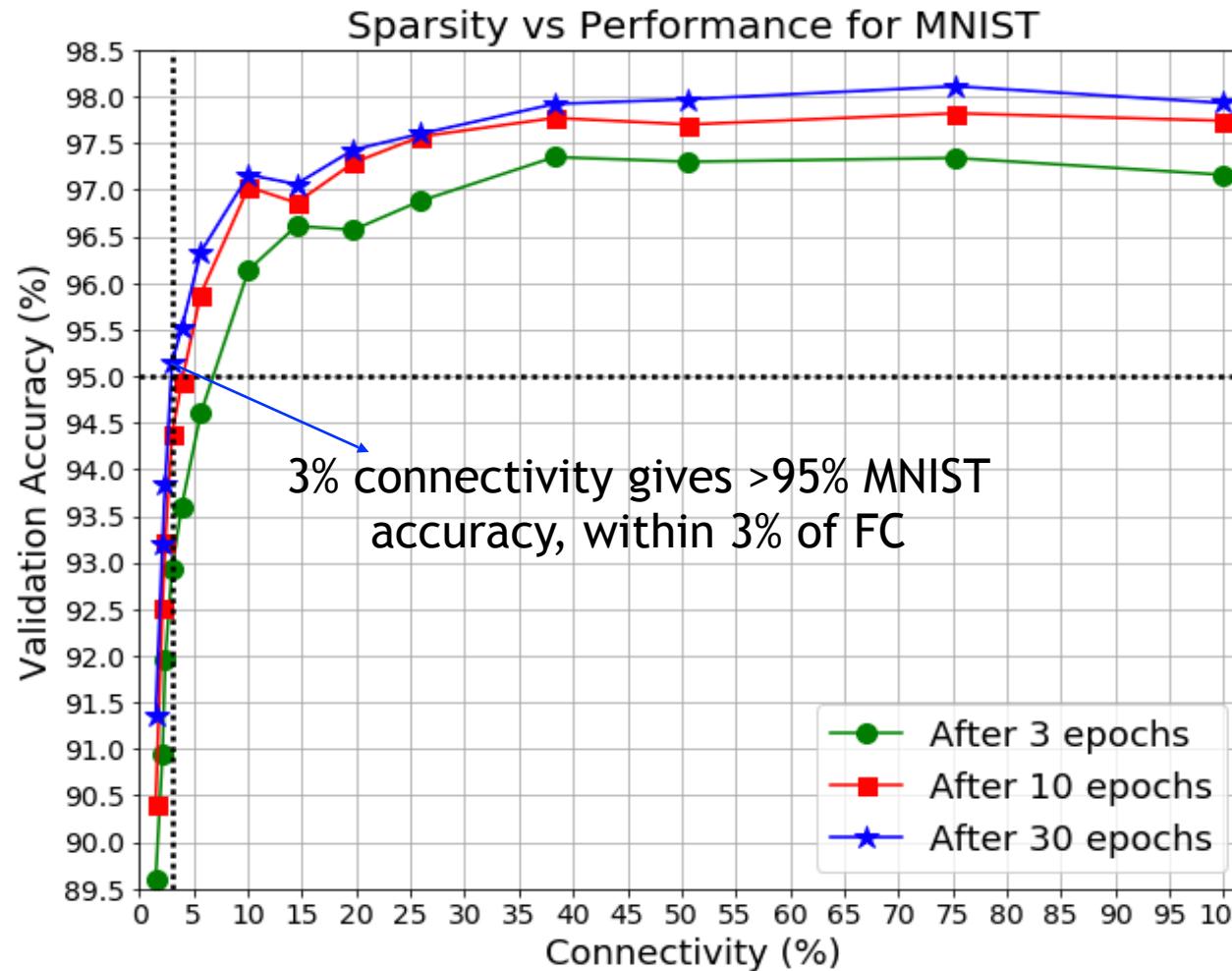
Does predefined sparsity work?



Ongoing research shows:

- ▶ Results can be further improved by planning connections

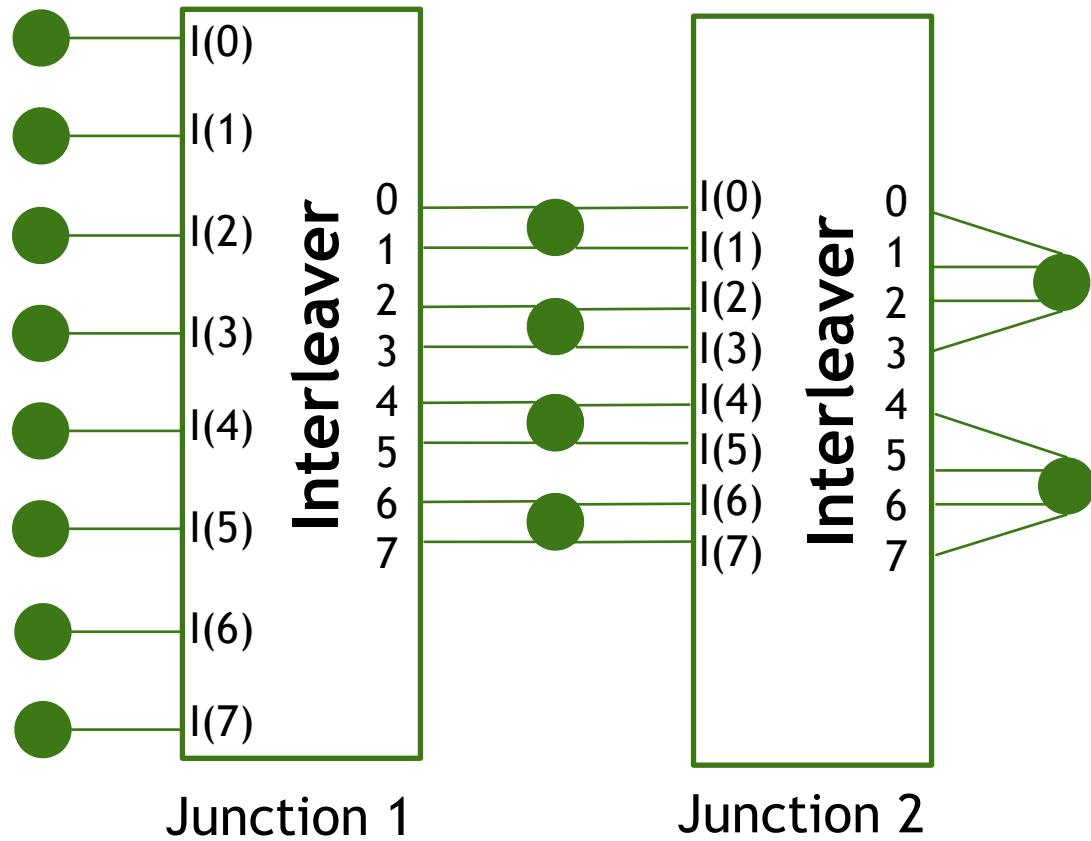
Does predefined sparsity work?



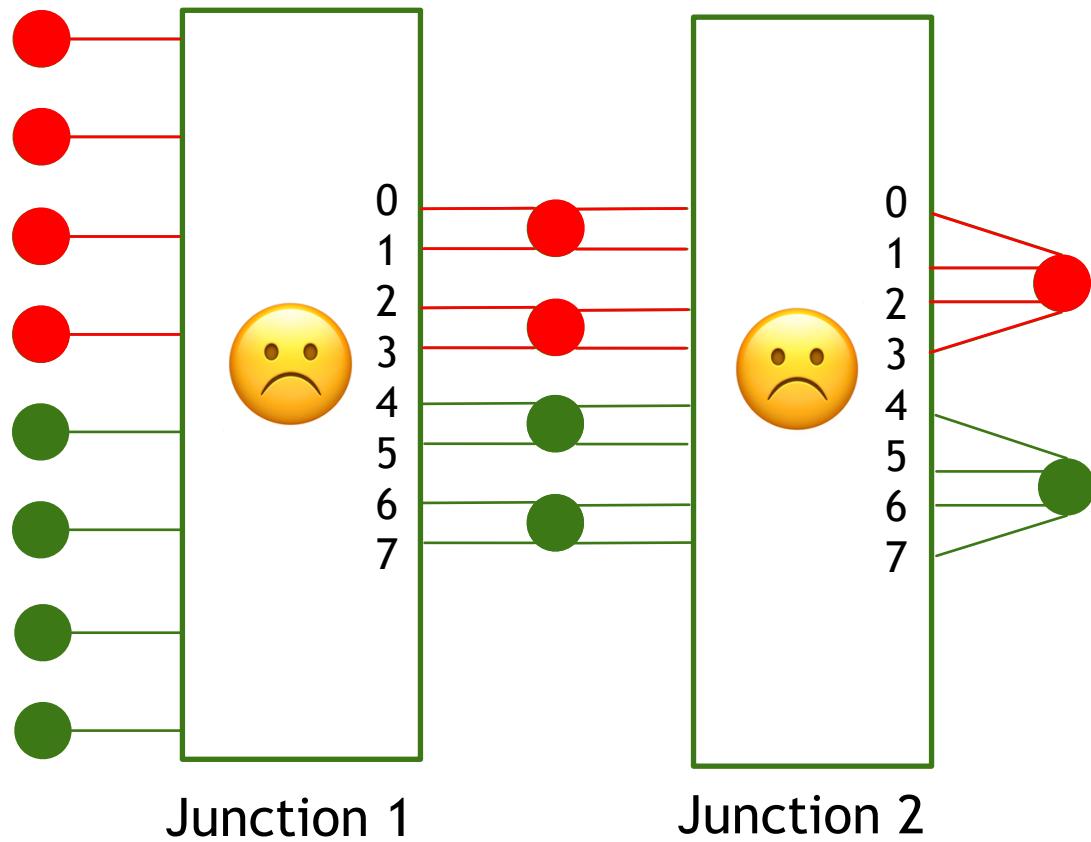
Ongoing research shows:

- ▶ Results can be further improved by planning connections
- ▶ Trend holds for other datasets like CIFAR-10

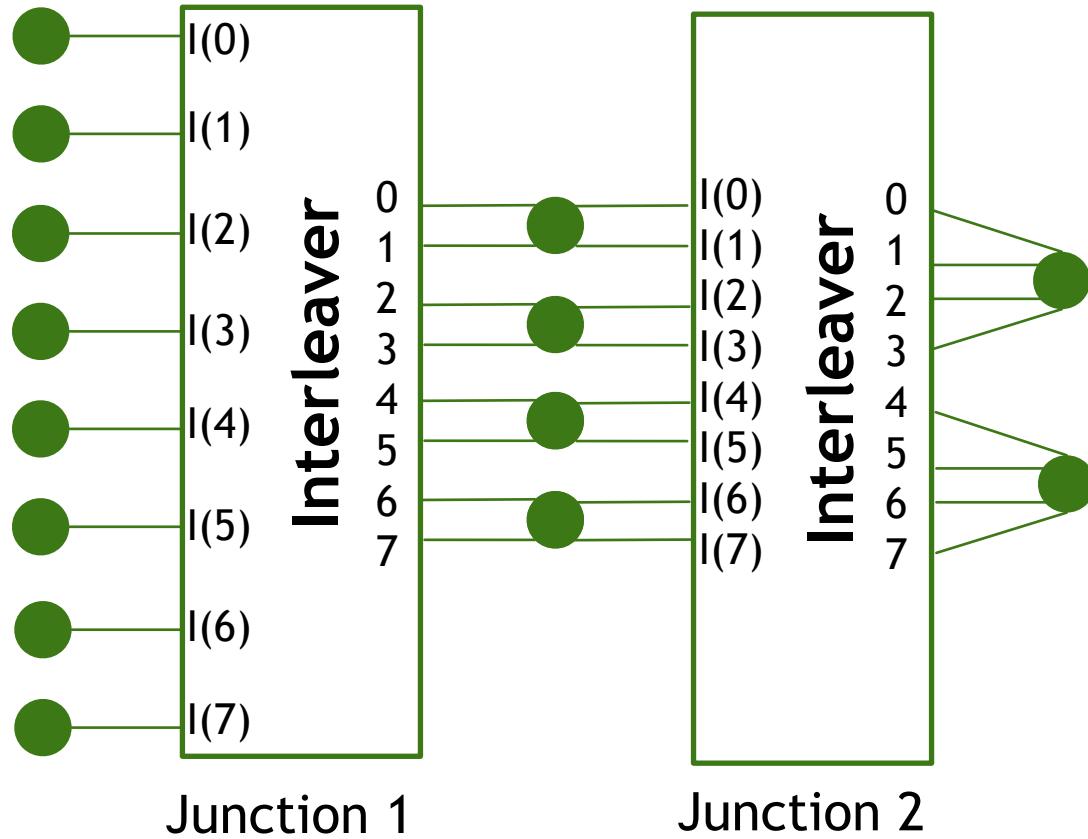
Interleaving and Spread



Interleaving and Spread



Interleaving and Spread



- Interleaver algorithm ensures:
- ▶ Each output connected to a *good chunk* of different inputs
 - ▶ No neuron unconnected

Edge Processing

- ▶ *Concurrent Network Processes*
 - ▶ Feedforward (FF) - Weights and activations
 - ▶ Backpropagation (BP) - Weights, deltas and activation derivatives
 - ▶ Parameter Update (UP) - Weights, deltas and activations
- ▶ Weights (edges) used in all processes
 - ▶ Single weight memory bank
- ▶ Process z sets of parameters together

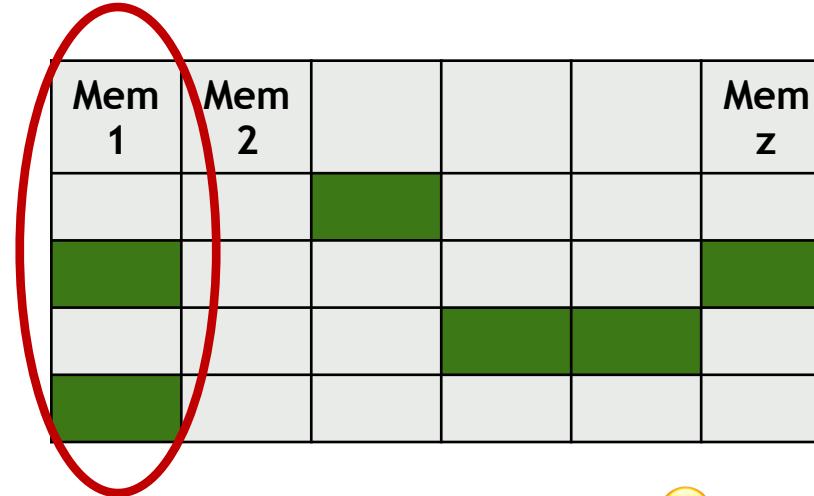
$z = \text{Degree of parallelism}$

Memory Organization

- ▶ z memories for all parameters
- ▶ Read 1 entry from each memory at a time

Mem 1	Mem 2				Mem z

Clash-free access 😊

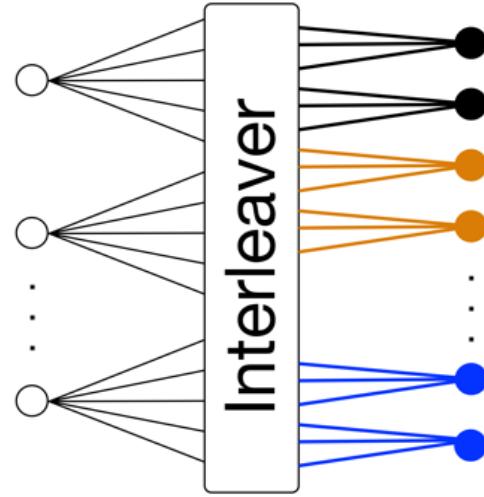


Clash stalls processing 😞

Avoid clashes!

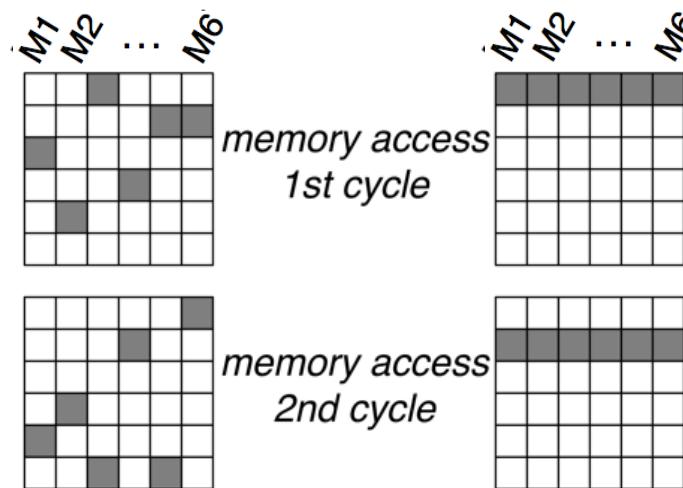
Order of Accesses

Interleaver must prevent clashes



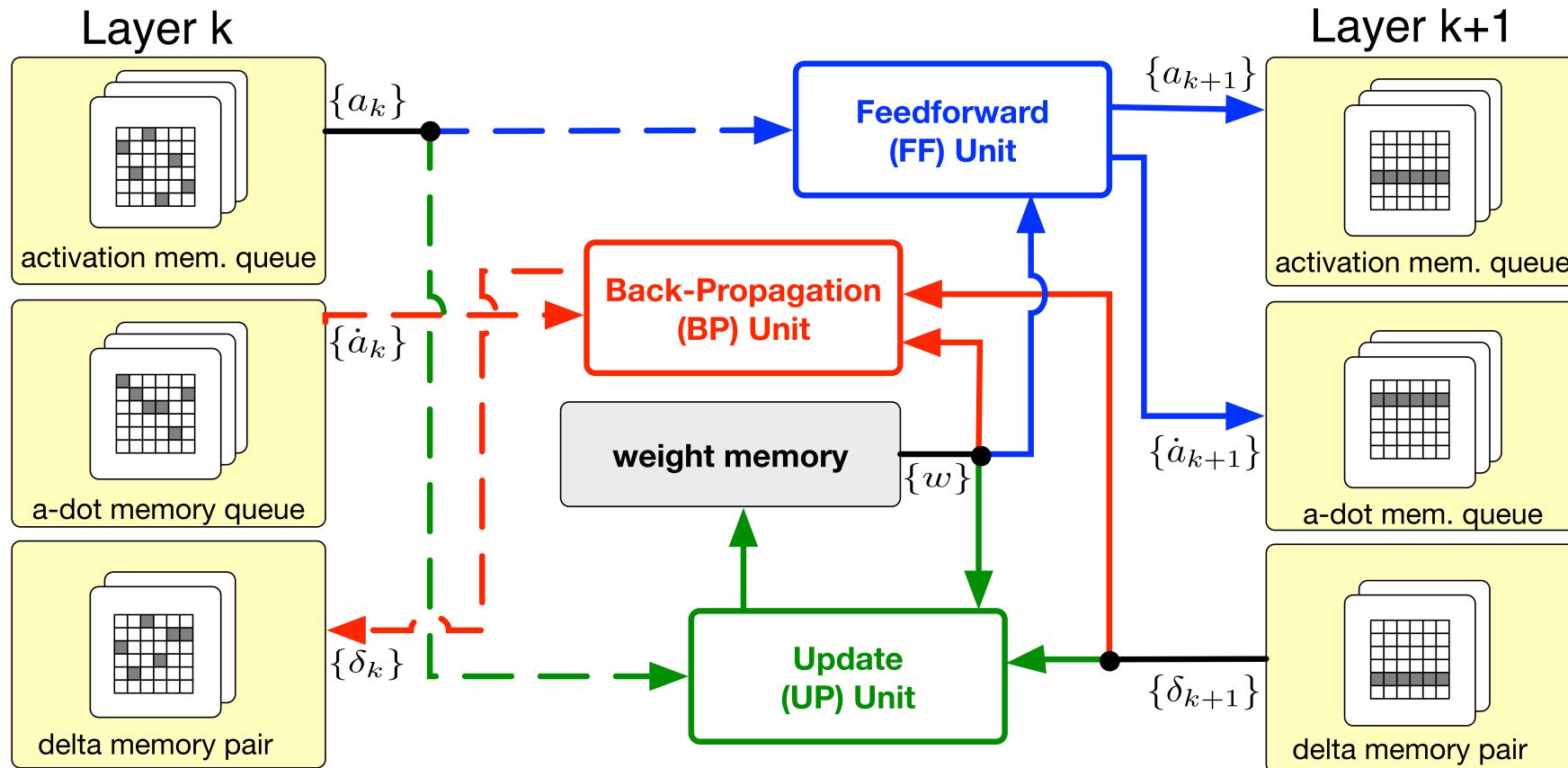
Example: $z=6$

Permuted order
accesses for previous
layer parameters

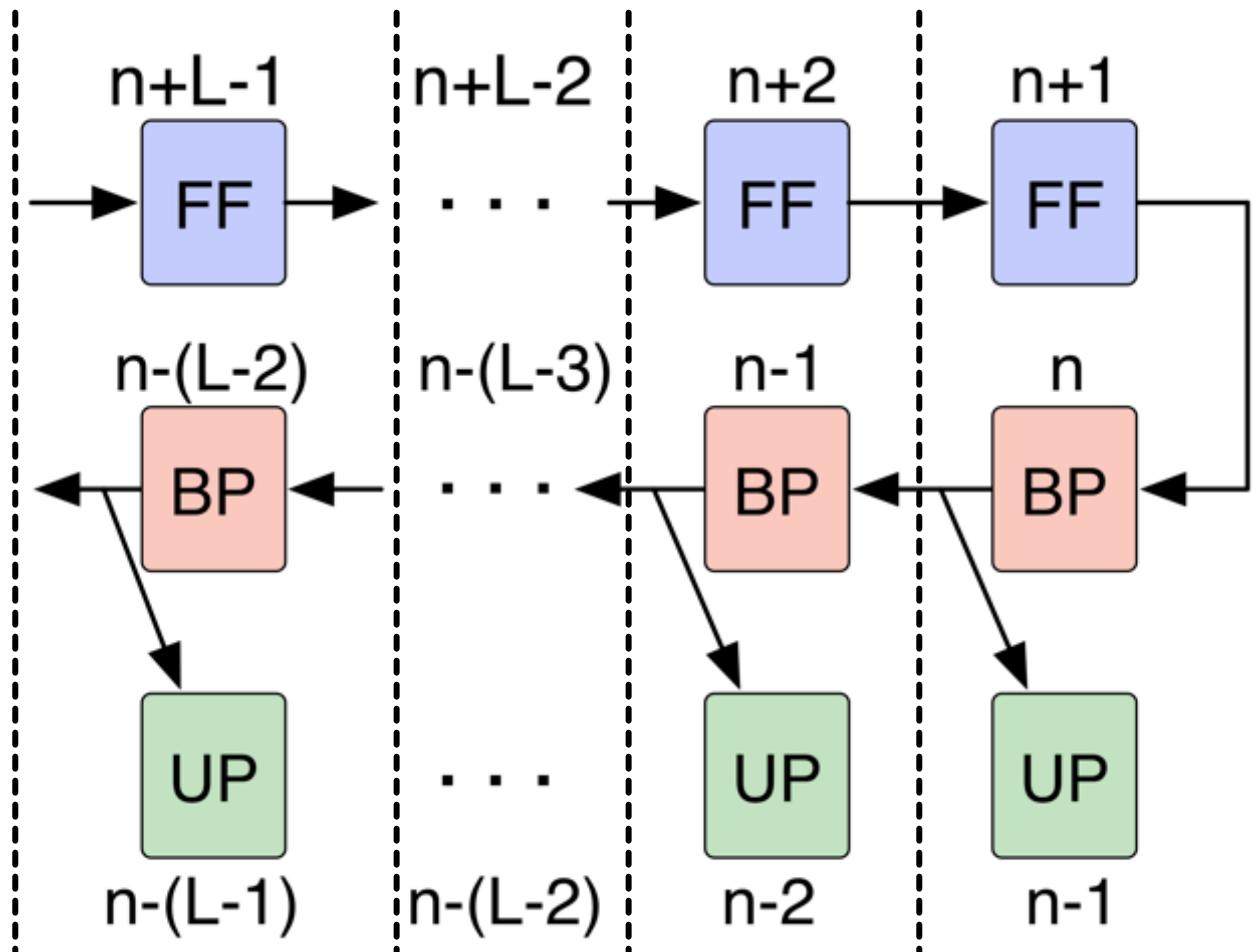


Natural order accesses
for junction weights and
next layer parameters

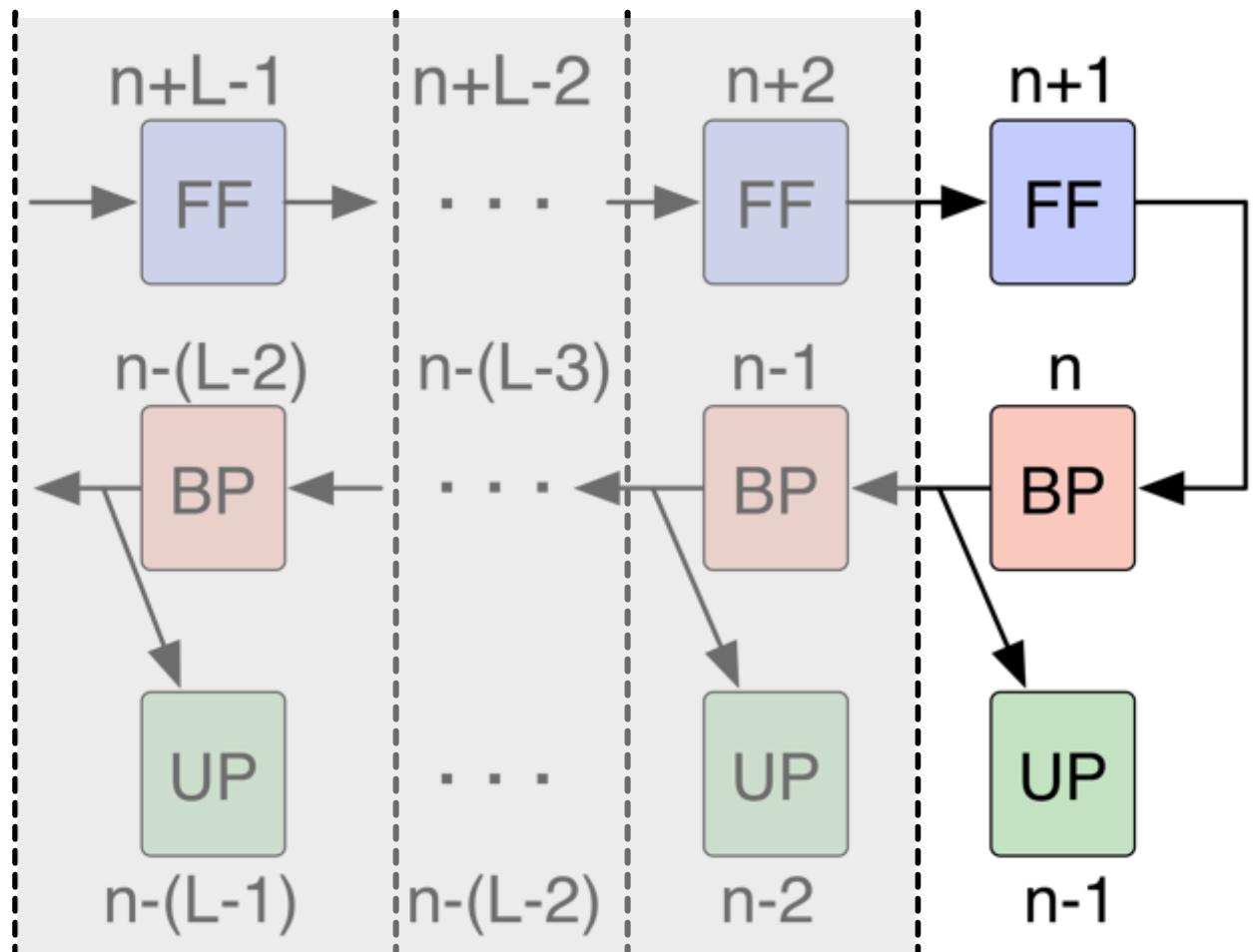
Operational Parallelization in a Junction



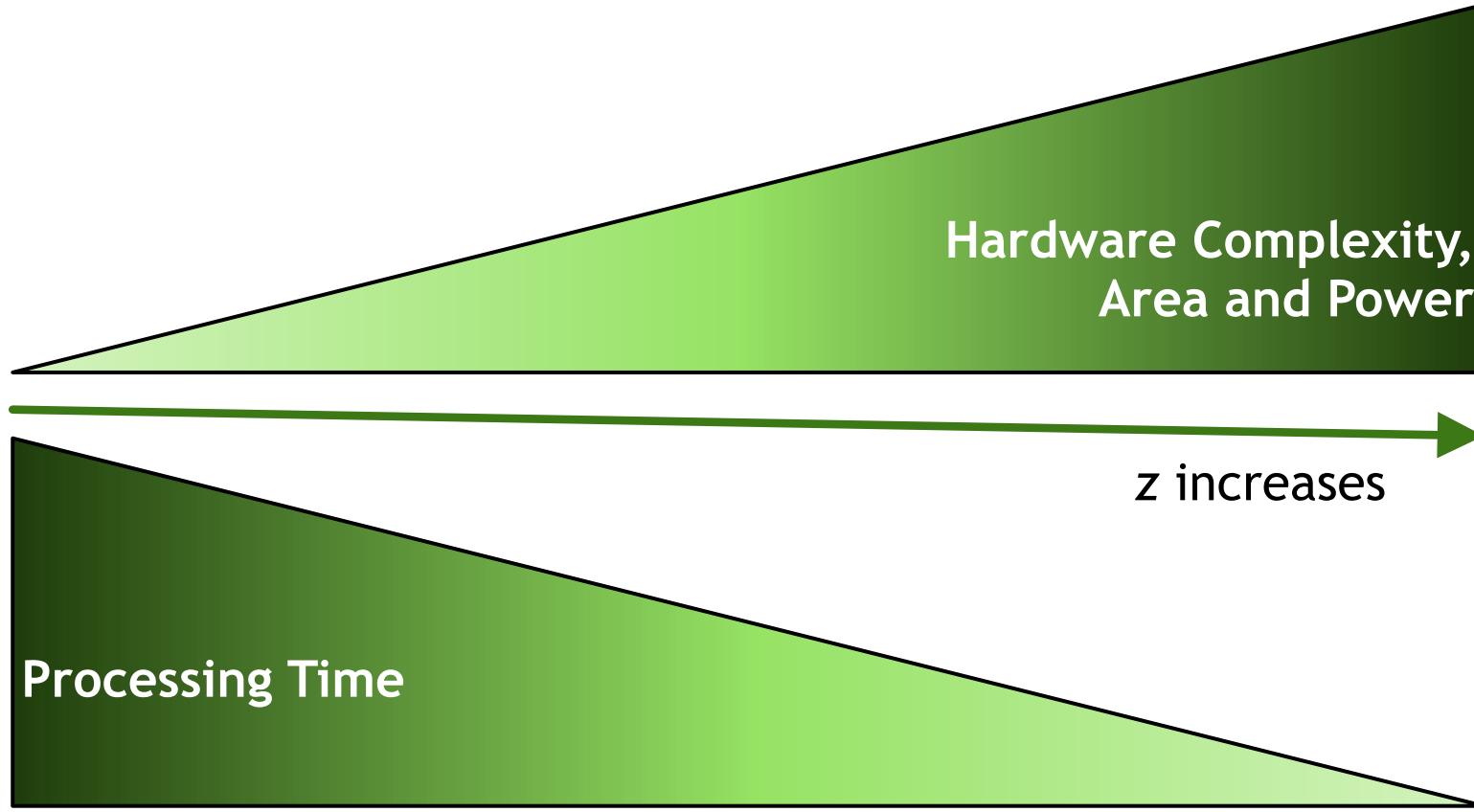
Pipelining across Junctions - Speedup



Pipelining across Junctions - Speedup



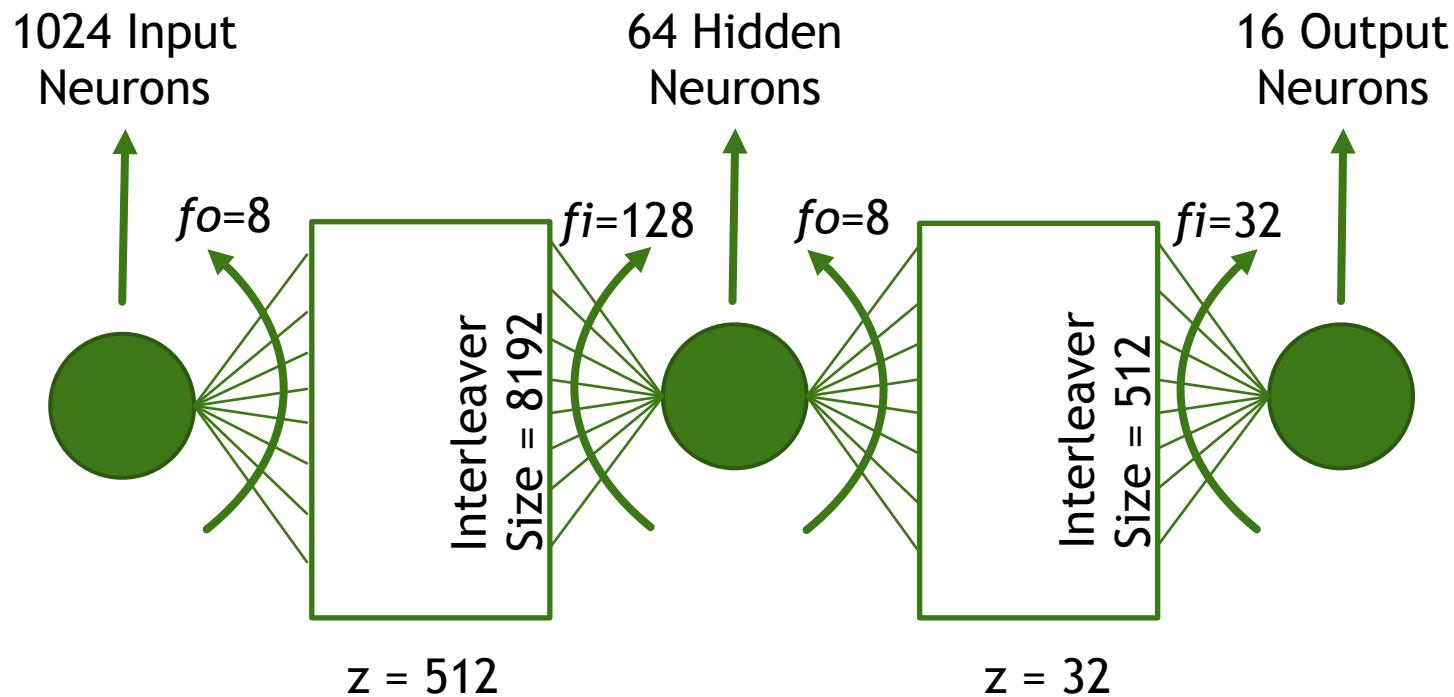
Architecture Flexibility



**Changing z makes architecture automatically adapts to problem size and available hardware.
Suitable for FPGA reconfigurability**

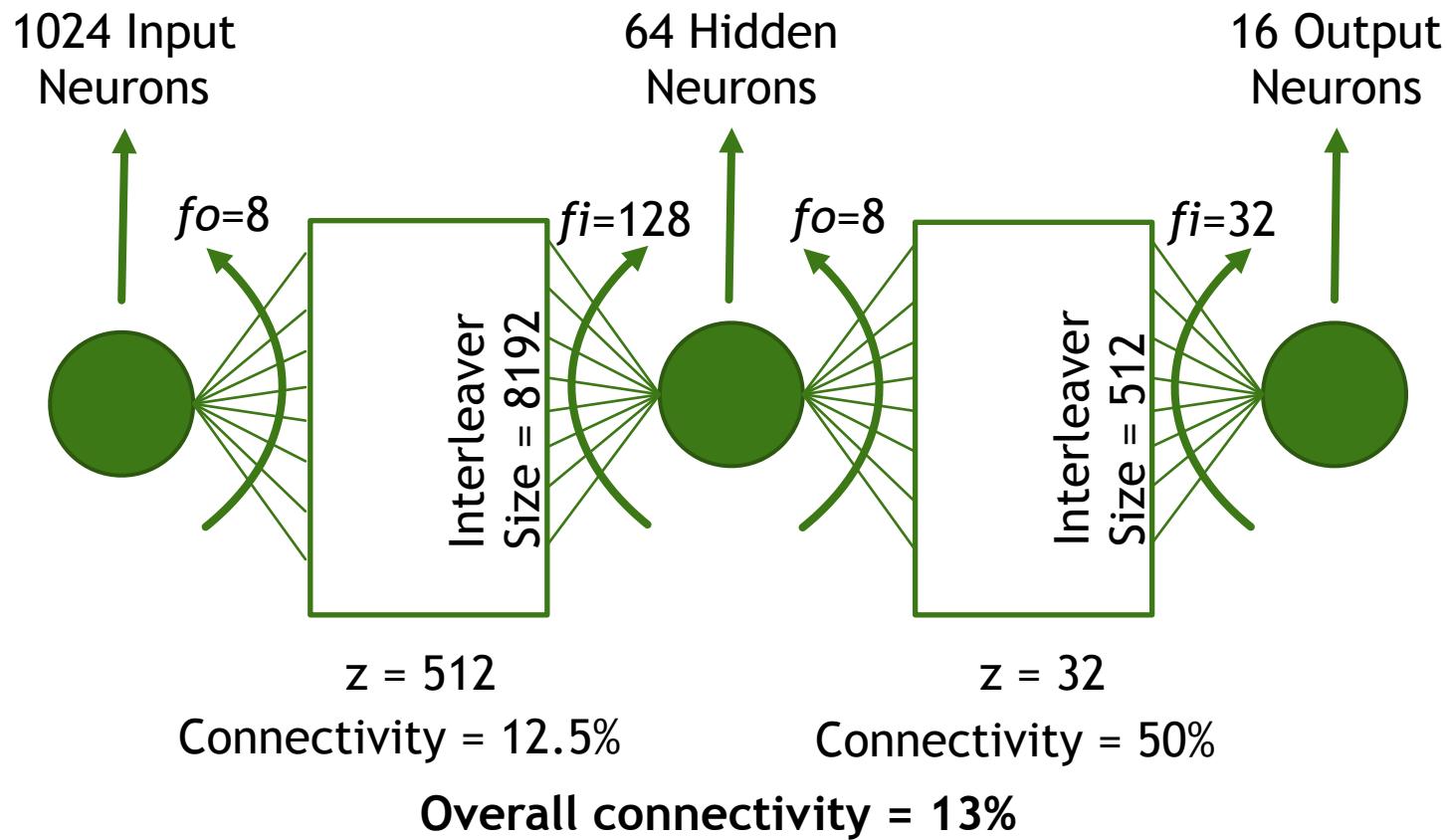
Hardware Simulations

Verilog RTL on MNIST dataset

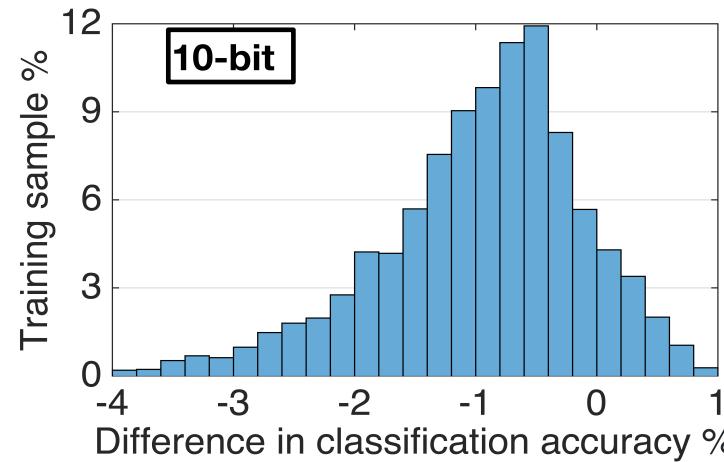
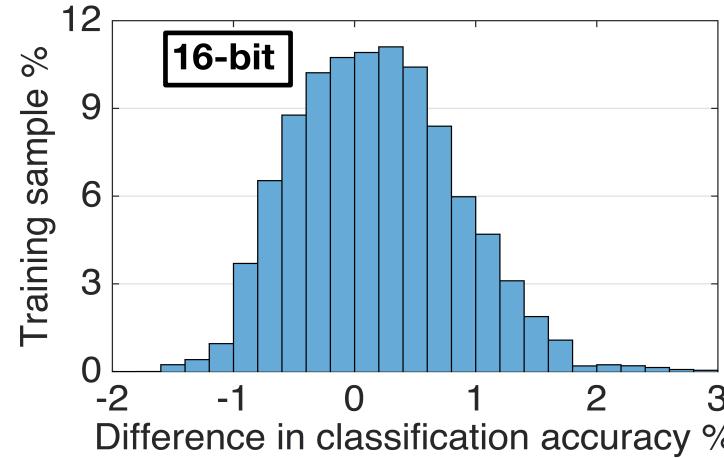


Hardware Simulations

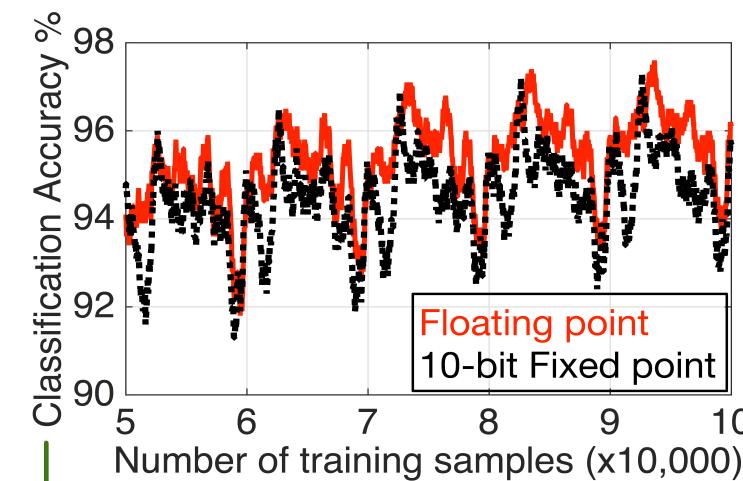
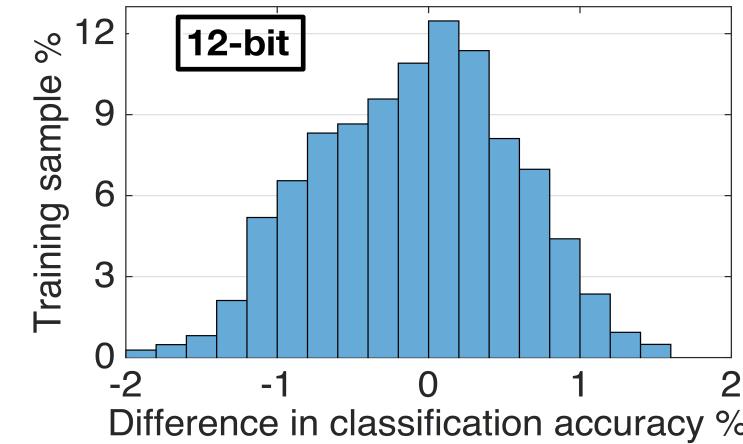
Verilog RTL on MNIST dataset



RTL Fixed Point Results vs Floating Point



Fixed point –
Floating point



Moving average of
last 1000 samples

Total 100K
samples

Summary and Outstanding Issues

- ▶ Flexible hardware architecture for online training and inference
- ▶ Predefined sparsity reduces memory and computational complexity
- ▶ Speedup due to operational parallelization and junction pipelining

- ▶ Extend to other types of neural networks
- ▶ Memory bandwidth bottlenecks
- ▶ Theoretical exploration of connectivity patterns

Thank you!

Questions?