Exploring Complexity Reduction in Deep Learning

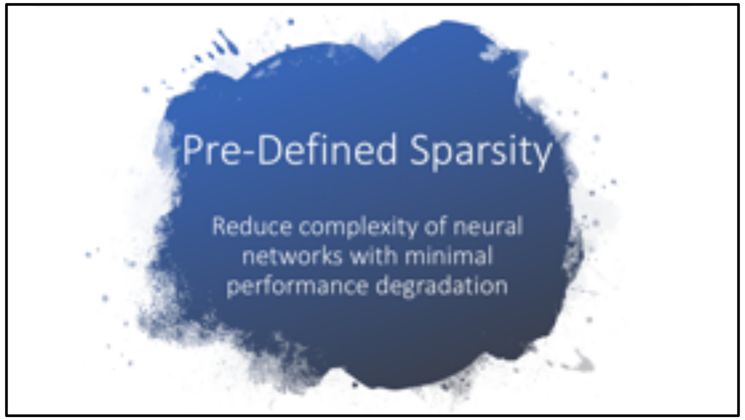
Sourya Dey

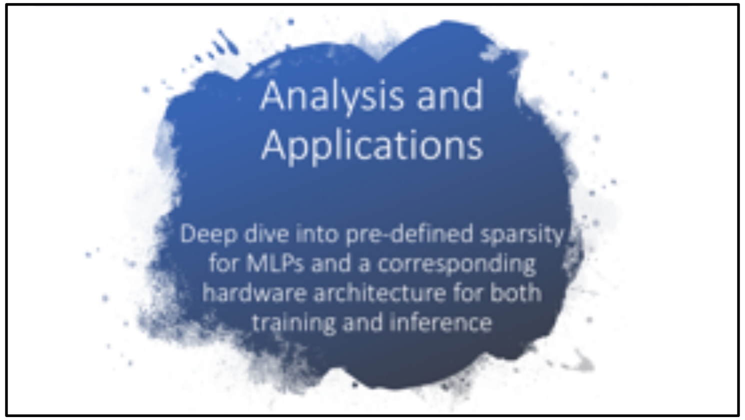
PhD Candidate, University of Southern California
Advisors: Peter A. Beerel and Keith M. Chugg

November 18, 2019

USC Viterbi
School of Engineering
Ming Hsieh Department of Electrical and Computer Engineering

Outline


Pre-Defined Sparsity
Reduce complexity of neural networks with minimal performance degradation


Analysis and Applications
Deep dive into pre-defined sparsity for MLPs and a corresponding hardware architecture for both training and inference


Model Search
Automate the design of CNNs with good performance and low complexity

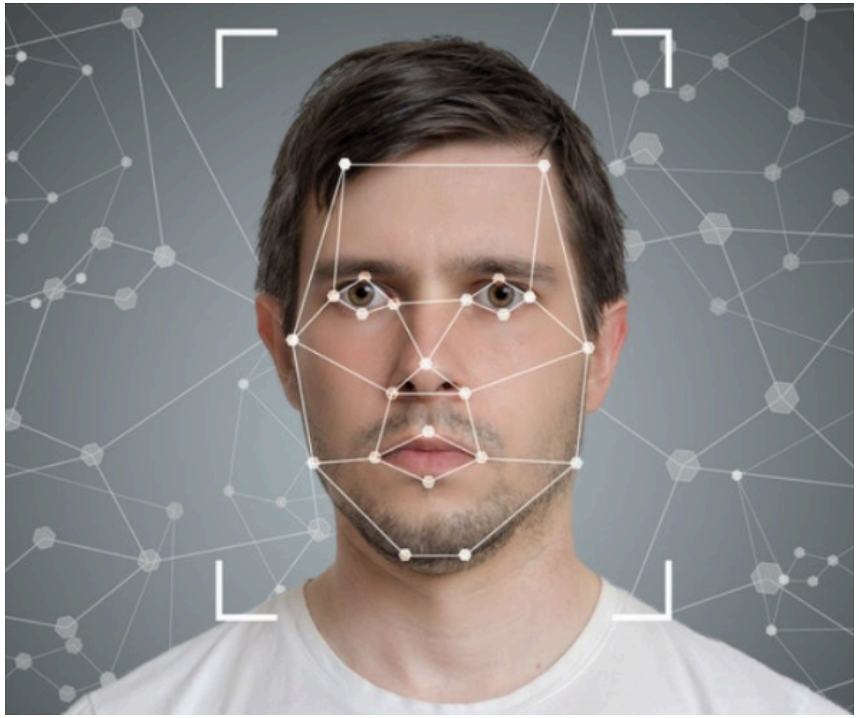
Pre-Defined Sparsity

Reduce complexity of neural networks with minimal performance degradation

Overview

Neural networks (NNs) are key machine learning technologies

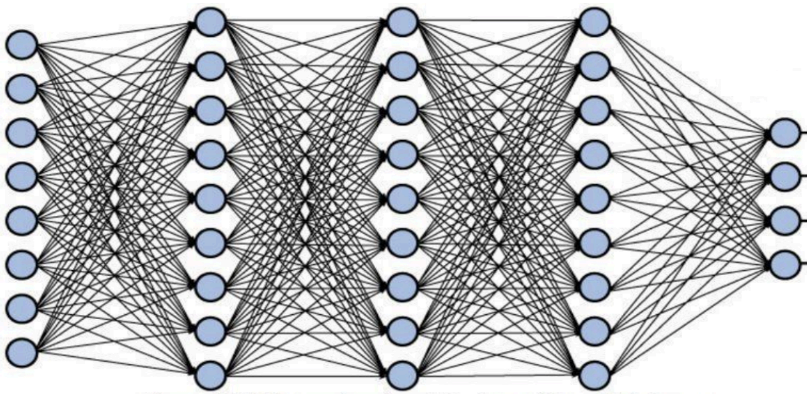
- ➢ Artificial intelligence
- ➢ Self-driving cars
- ➢ Speech recognition
- ➢ Face ID
- ➢ and more smart stuff …



Motivation

Modern neural networks suffer from parameter explosion

Typical deep **CNN**



Fully connected **(FC)** Multilayer Perceptron **(MLP)**

Training can take weeks on CPU

Cloud GPU resources are expensive

amazon web services

Microsoft Azure

Google Cloud Platform

The Complexity Conundrum

Storage and computational complexity dominated by weights

All the weights are used in all 3 operations

Feedforward (FF) $$\sum_{\forall i,j} W_{ij} a_j$$

Backpropagation (BP) $$\sum_{\forall i,j} W_{ij} \delta_i$$

Update parameters (UP) $$W_{ij} - \eta \nabla_{W_{ij}} C \quad \forall i,j$$

Our Work: Pre-defined Sparsity

Pre-define a sparse connection pattern **prior to training**

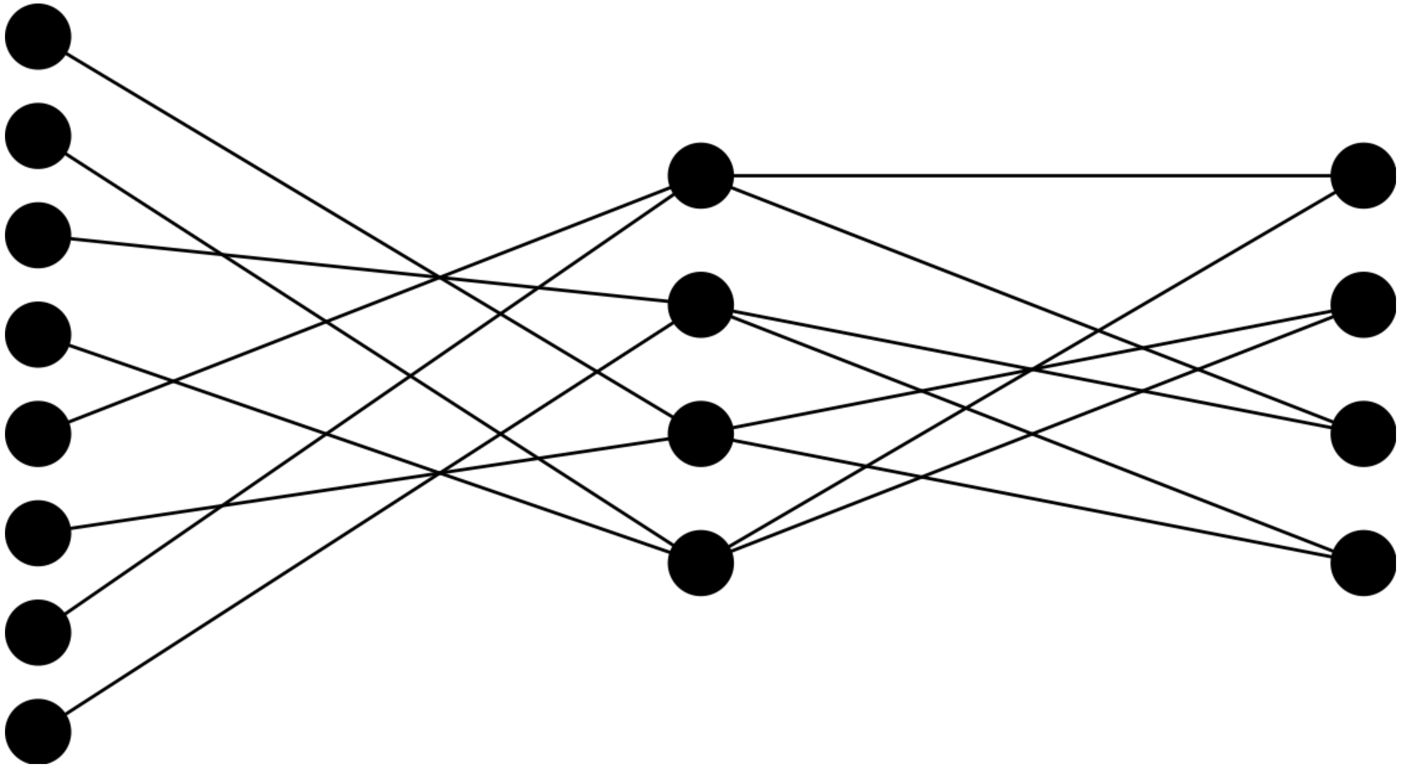
Use this sparse network for both training and inference

Our Work: Pre-defined Sparsity

Pre-define a sparse connection pattern **prior to training**

Use this sparse network for both training and inference

$$\boldsymbol{N}_{\text{net}} = (8, 4, 4)$$
$$\boldsymbol{d}_{\text{net}}^{\text{out}} = (1, 2)$$
$$\boldsymbol{d}_{\text{net}}^{\text{in}} = (2, 2)$$



Our Work: Pre-defined Sparsity

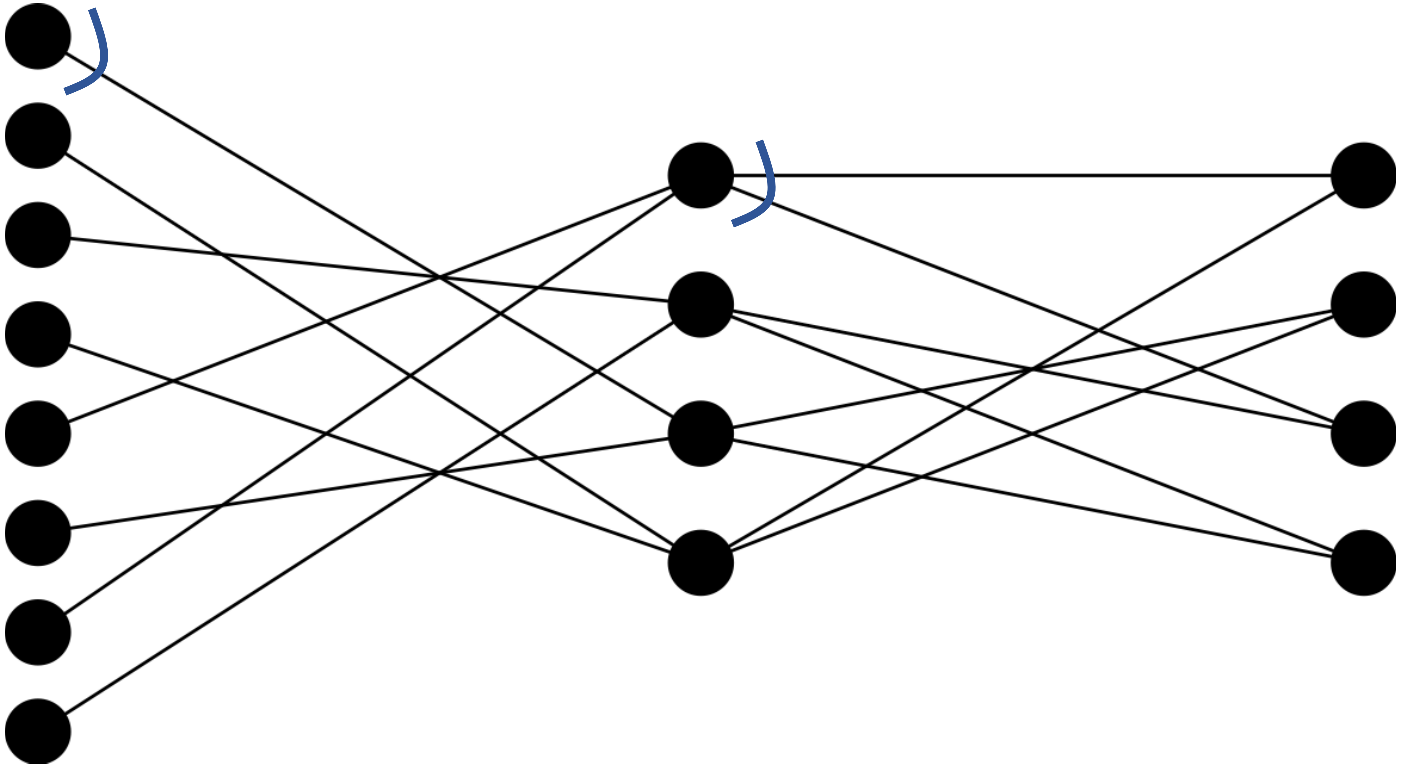
Pre-define a sparse connection pattern **prior to training**

Use this sparse network for both training and inference

$$\boldsymbol{N}_{\text{net}} = (8, 4, 4)$$

$$\boxed{\boldsymbol{d}_{\text{net}}^{\text{out}} = (1, 2)}$$

$$\boldsymbol{d}_{\text{net}}^{\text{in}} = (2, 2)$$



Our Work: Pre-defined Sparsity

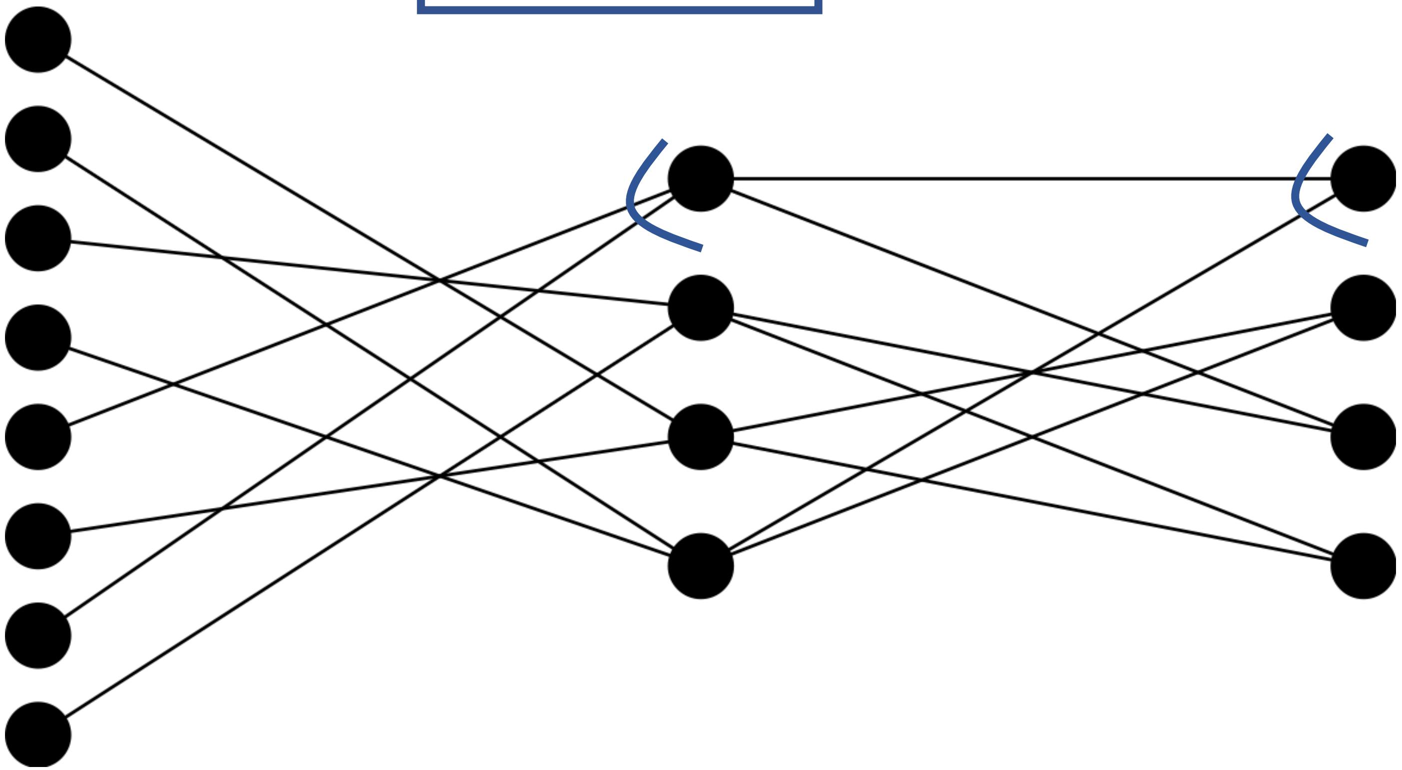
Pre-define a sparse connection pattern **prior to training**

Use this sparse network for both training and inference

$$\boldsymbol{N}_{\text{net}} = (8,4,4)$$

$$\boldsymbol{d}_{\text{net}}^{\text{out}} = (1,2)$$

$$\boxed{\boldsymbol{d}_{\text{net}}^{\text{in}} = (2,2)}$$



Our Work: Pre-defined Sparsity

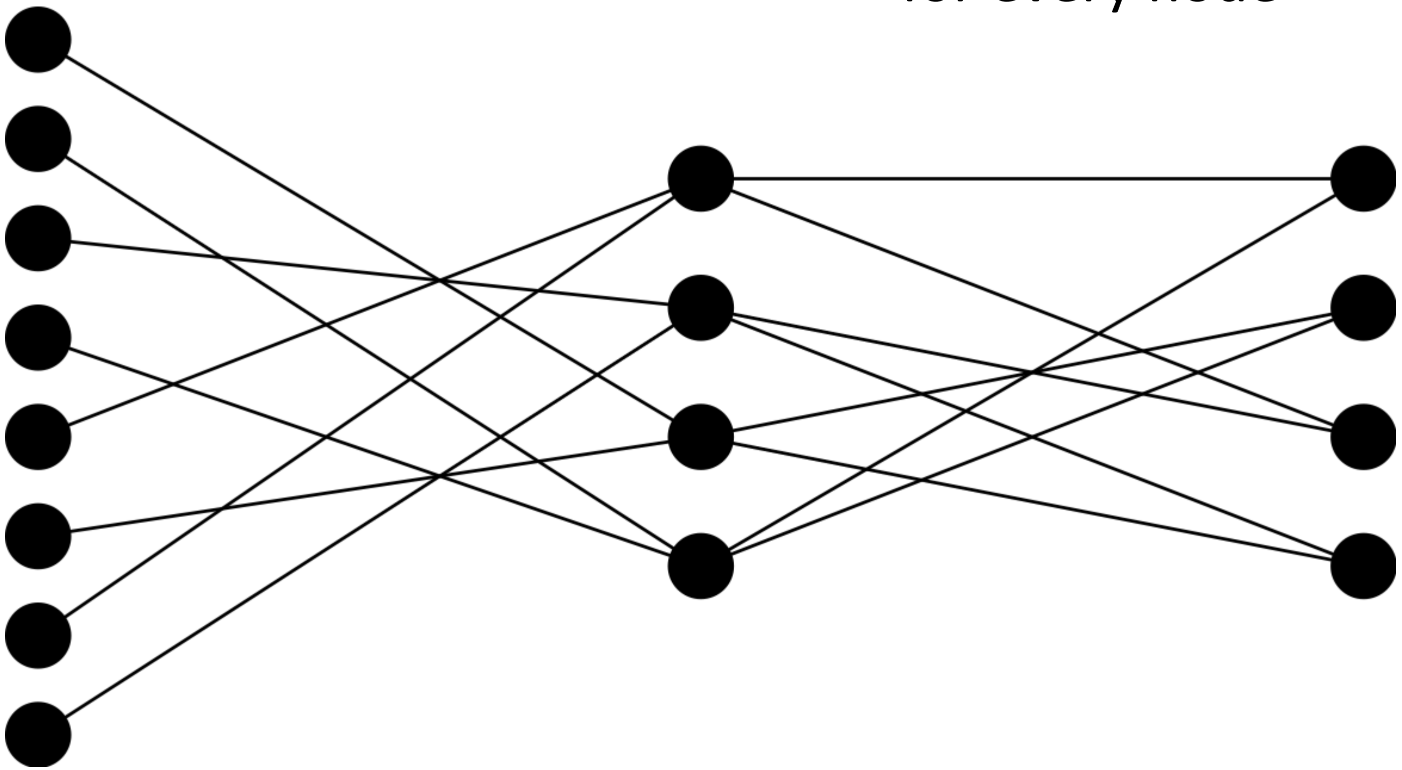
Pre-define a sparse connection pattern **prior to training**

Use this sparse network for both training and inference

$$\boldsymbol{N}_{\text{net}} = (8, 4, 4)$$

$$\left.\begin{aligned} \boldsymbol{d}_{\text{net}}^{\text{out}} &= (1, 2) \\ \boldsymbol{d}_{\text{net}}^{\text{in}} &= (2, 2) \end{aligned}\right\}$$

Structured Constraints:
Fixed in-, out-degrees for every node



Our Work: Pre-defined Sparsity

Pre-define a sparse connection pattern **prior to training**

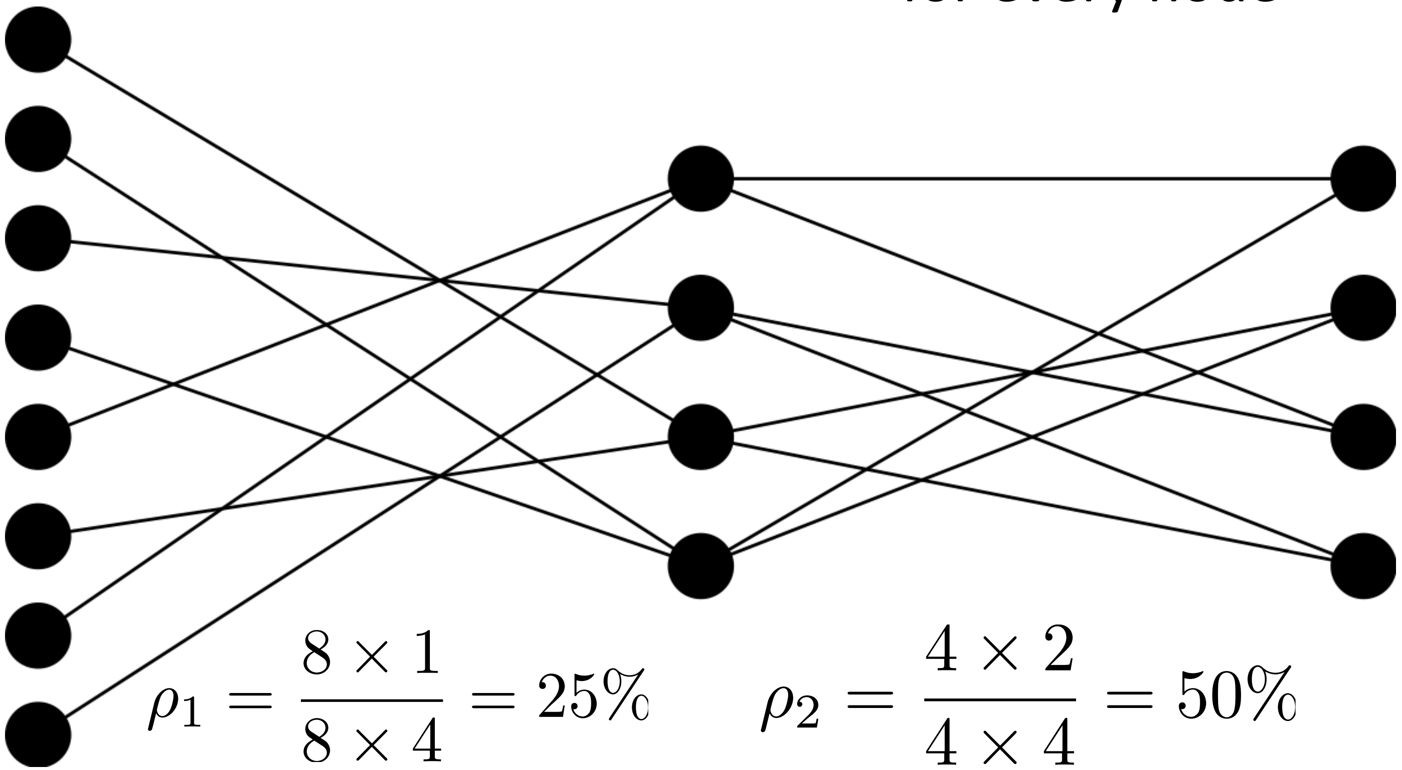
Use this sparse network for both training and inference

$$\boldsymbol{N}_{\text{net}} = (8,4,4)$$

$$\left.\begin{aligned}\boldsymbol{d}^{\text{out}}_{\text{net}} &= (1,2)\\ \boldsymbol{d}^{\text{in}}_{\text{net}} &= (2,2)\end{aligned}\right\}$$

Structured Constraints:

Fixed in-, out-degrees for every node



$$\rho_1 = \frac{8\times 1}{8\times 4} = 25\% \qquad \rho_2 = \frac{4\times 2}{4\times 4} = 50\%$$

$$\rho_{\text{net}} = \frac{8+8}{32+16} = 33\%$$

Overall Density compared to FC

Our Work: Pre-defined Sparsity

Pre-define a sparse connection pattern **prior to training**

Use this sparse network for both training and inference

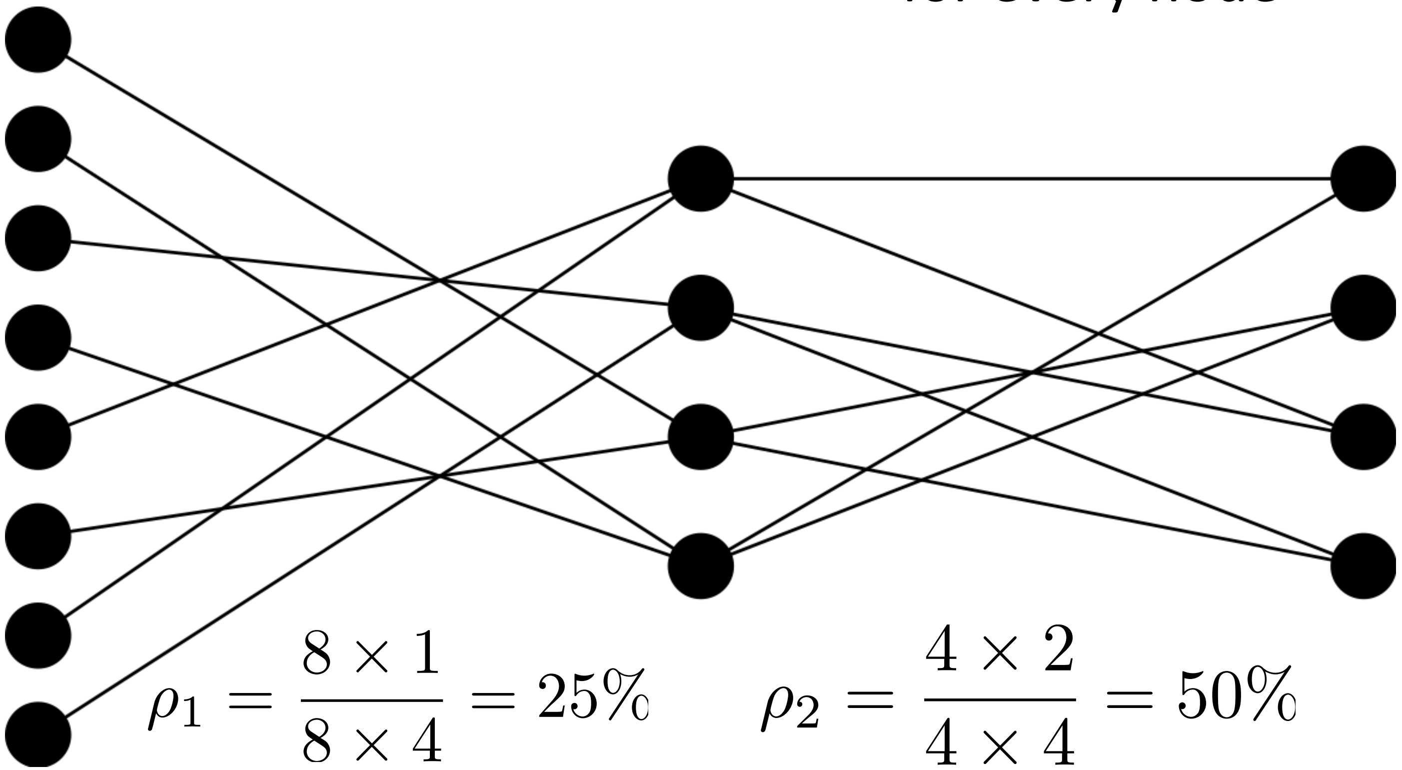
Reduced training *and* inference complexity

$$\boldsymbol{N}_{\text{net}} = (8,4,4)$$

$$\boldsymbol{d}^{\text{out}}_{\text{net}} = (1,2)$$

$$\boldsymbol{d}^{\text{in}}_{\text{net}} = (2,2)$$

Structured Constraints: Fixed in-, out-degrees for every node

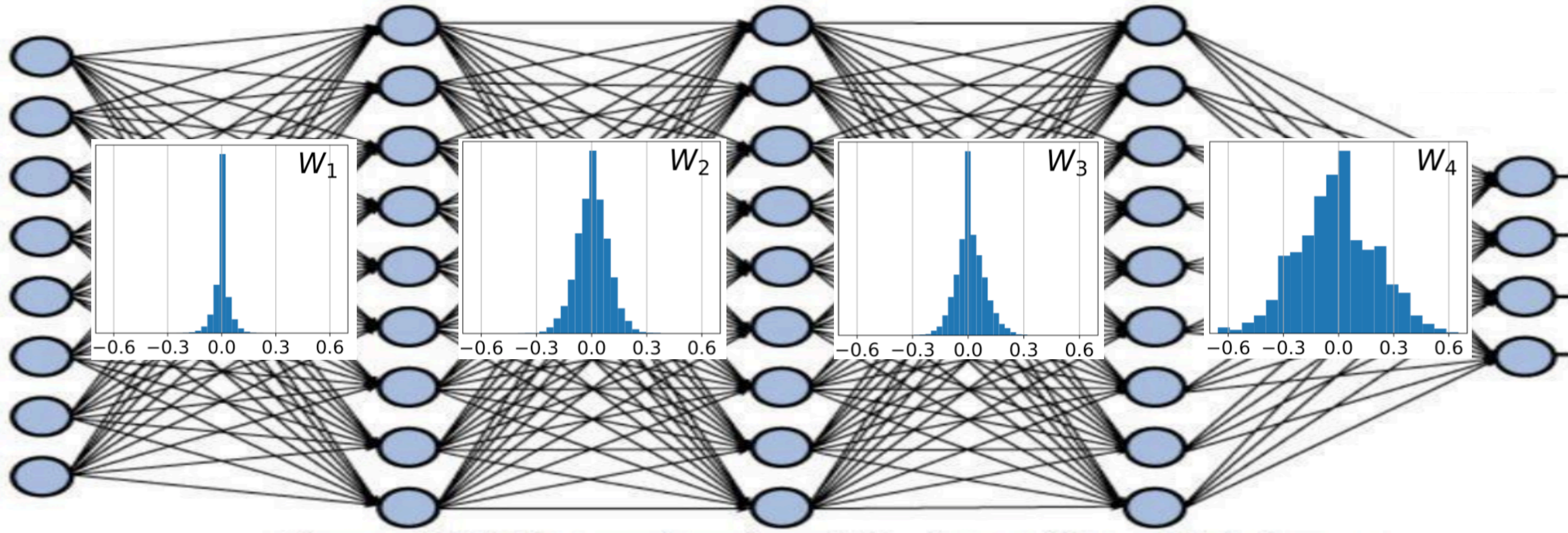


$$\rho_1 = \frac{8 \times 1}{8 \times 4} = 25\% \qquad \rho_2 = \frac{4 \times 2}{4 \times 4} = 50\%$$

$$\rho_{\text{net}} = \frac{8+8}{32+16} = 33\%$$

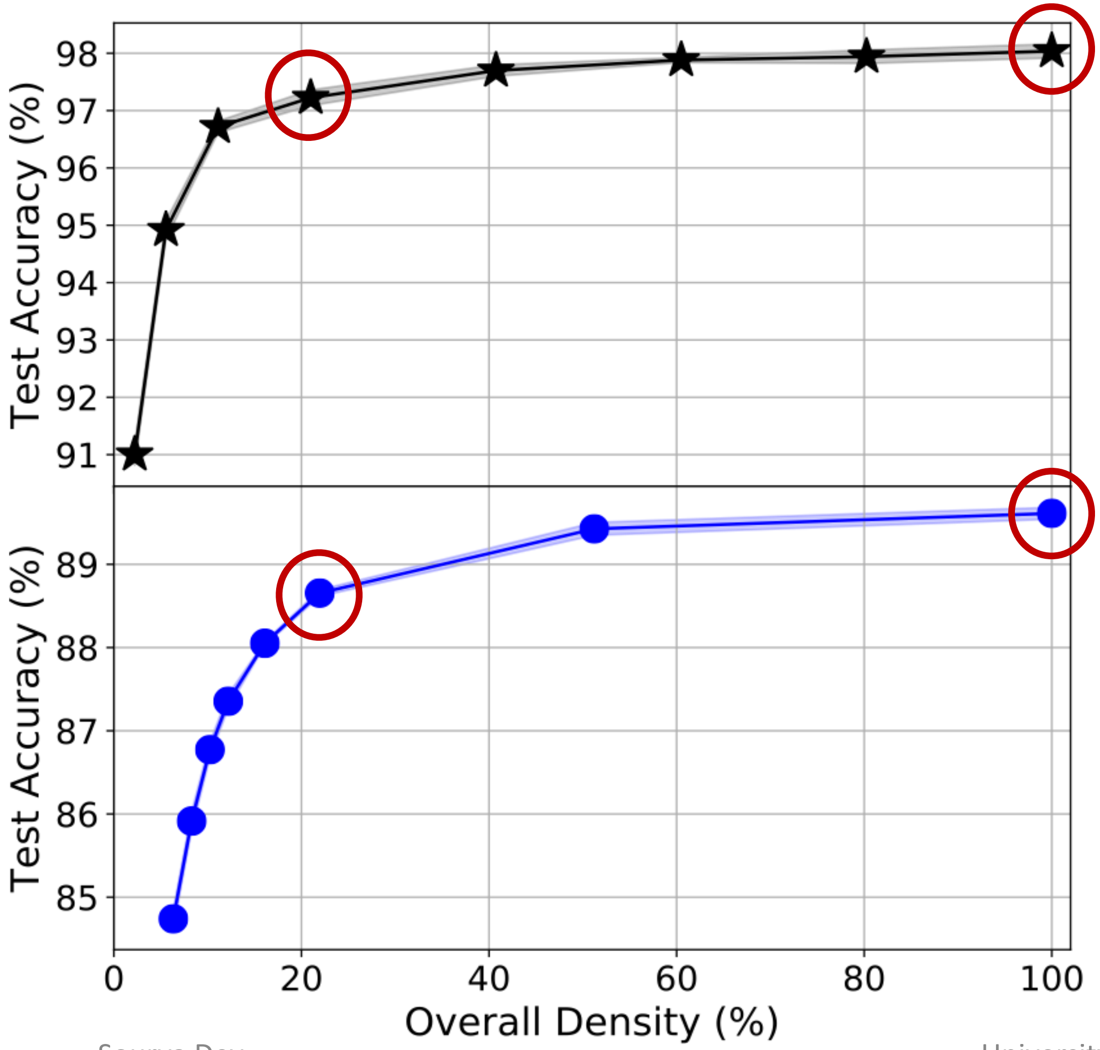
Overall Density compared to FC

Motivation behind pre-defined sparsity



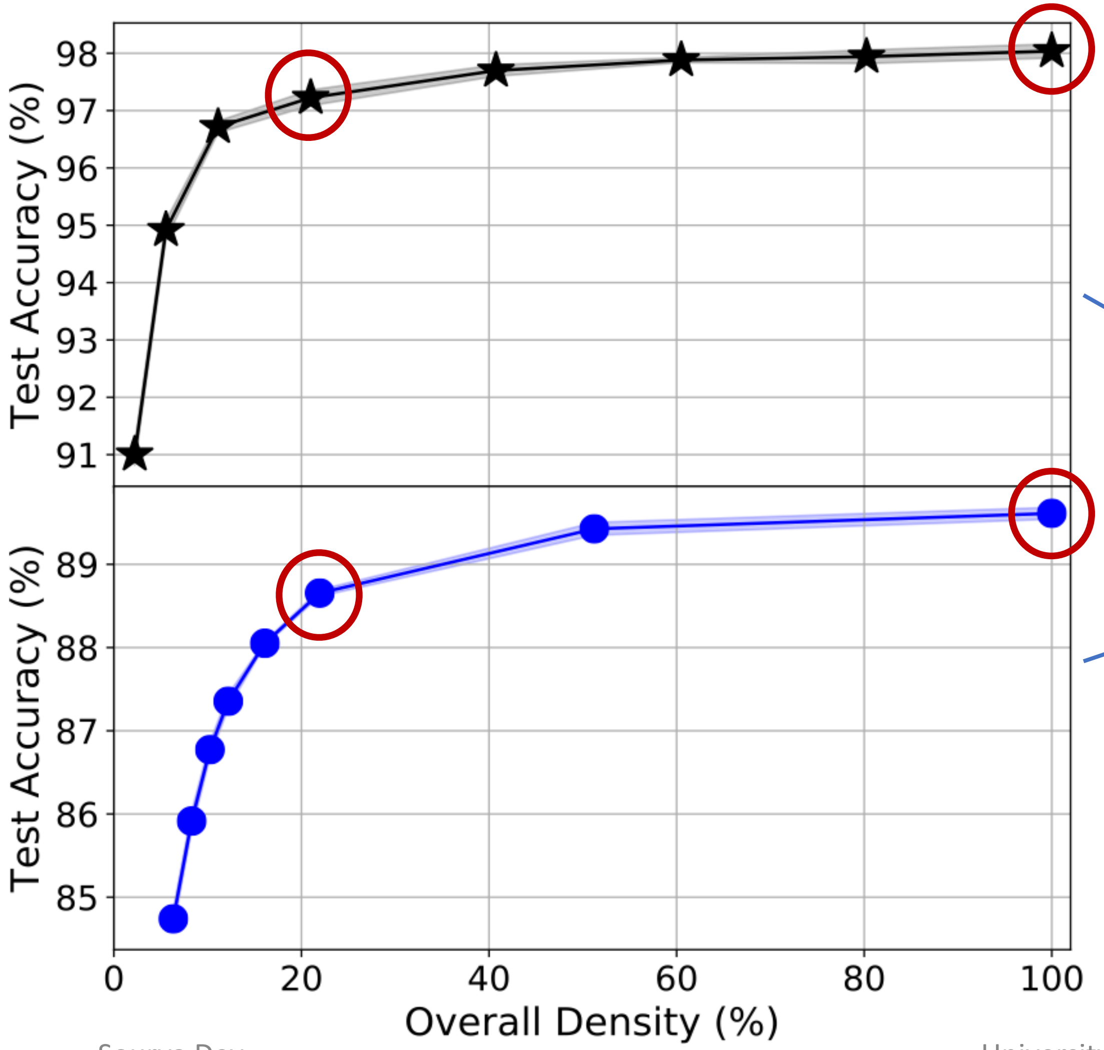
In a FC network, most weights are very small in magnitude after training

Pre-defined sparsity performance on MLPs



Starting with only 20% of parameters reduces test accuracy by just 1%

Pre-defined sparsity performance on MLPs



Starting with only 20% of parameters reduces test accuracy by just 1%

MNIST handwritten digits

Reuters news articles

TIMIT phonemes

CIFAR images

Morse symbols

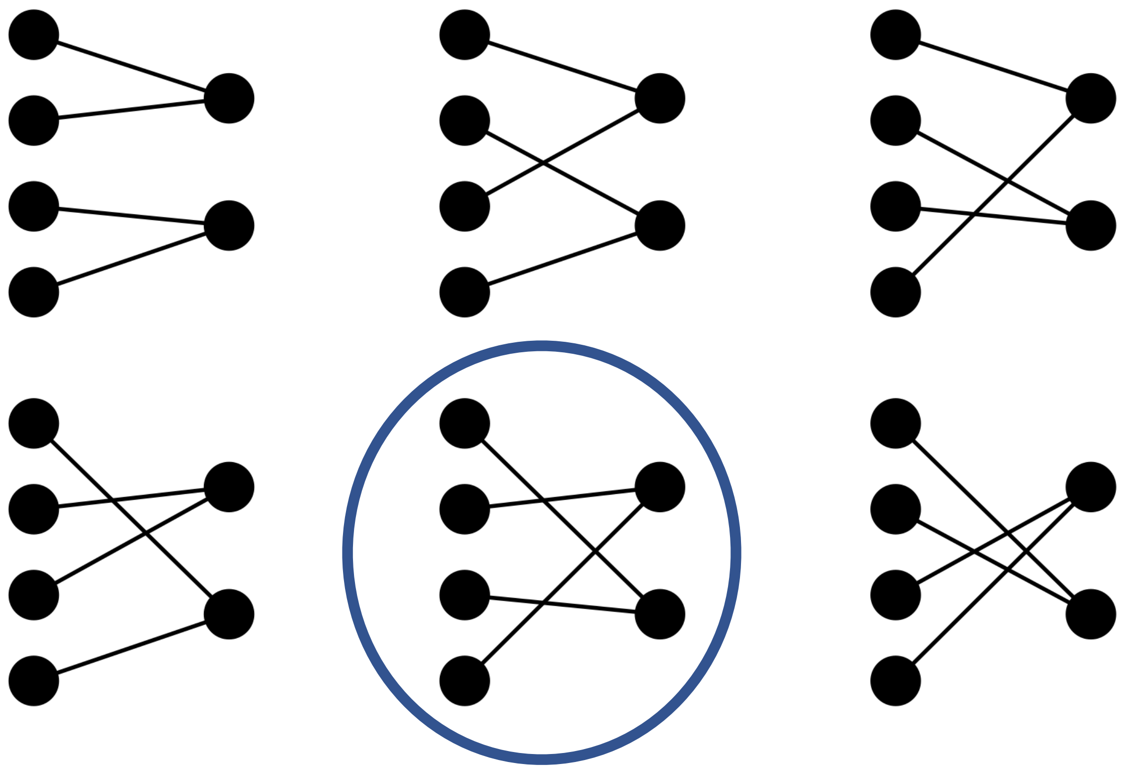
S. Dey, K. M. Chugg and P. A. Beerel, "Morse Code Datasets for Machine Learning," in ICCCNT 2018.
Won Best Paper award.
https://github.com/usc-hal/morse-dataset

Analysis and Applications

Deep dive into pre-defined sparsity for MLPs and a corresponding hardware architecture for both training and inference

Designing pre-defined sparse networks

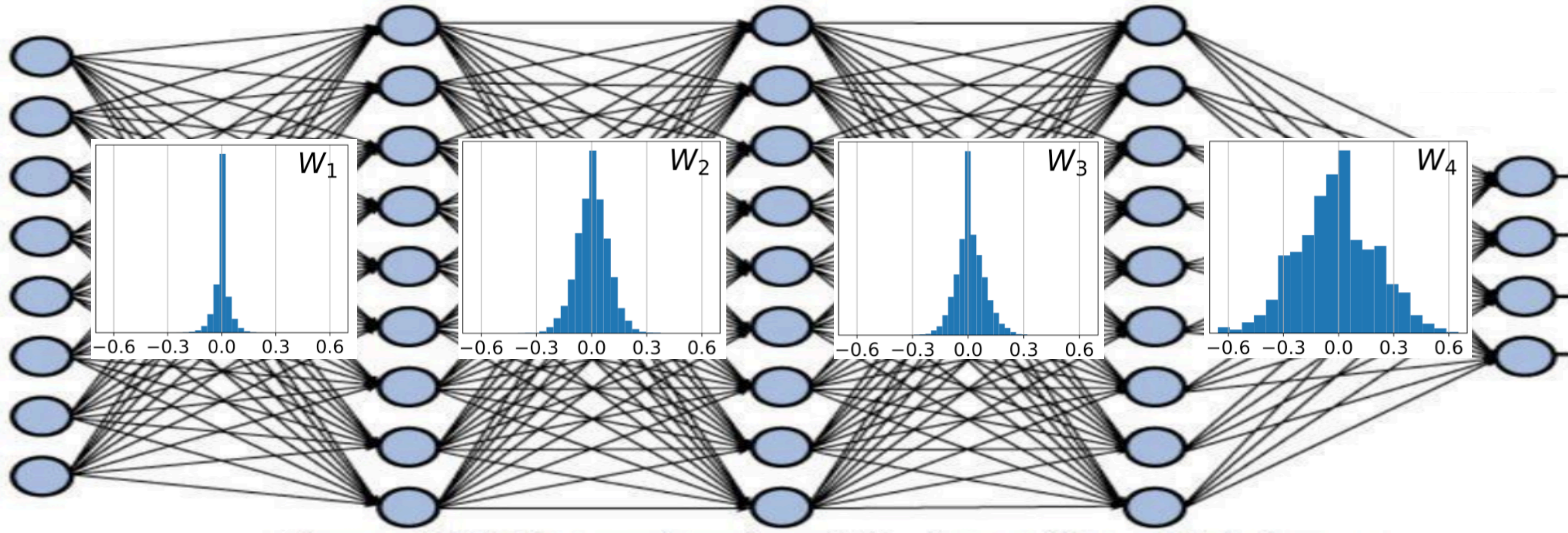
*A pre-defined sparse connection pattern is a **hyperparameter** to be set prior to training*



Find trends and guidelines to optimize pre-defined sparse patterns

S. Dey, K. Huang, P. A. Beerel and K. M. Chugg, "Pre-Defined Sparse Neural Networks with Hardware Acceleration," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 332-345, June 2019.

Individual junction densities



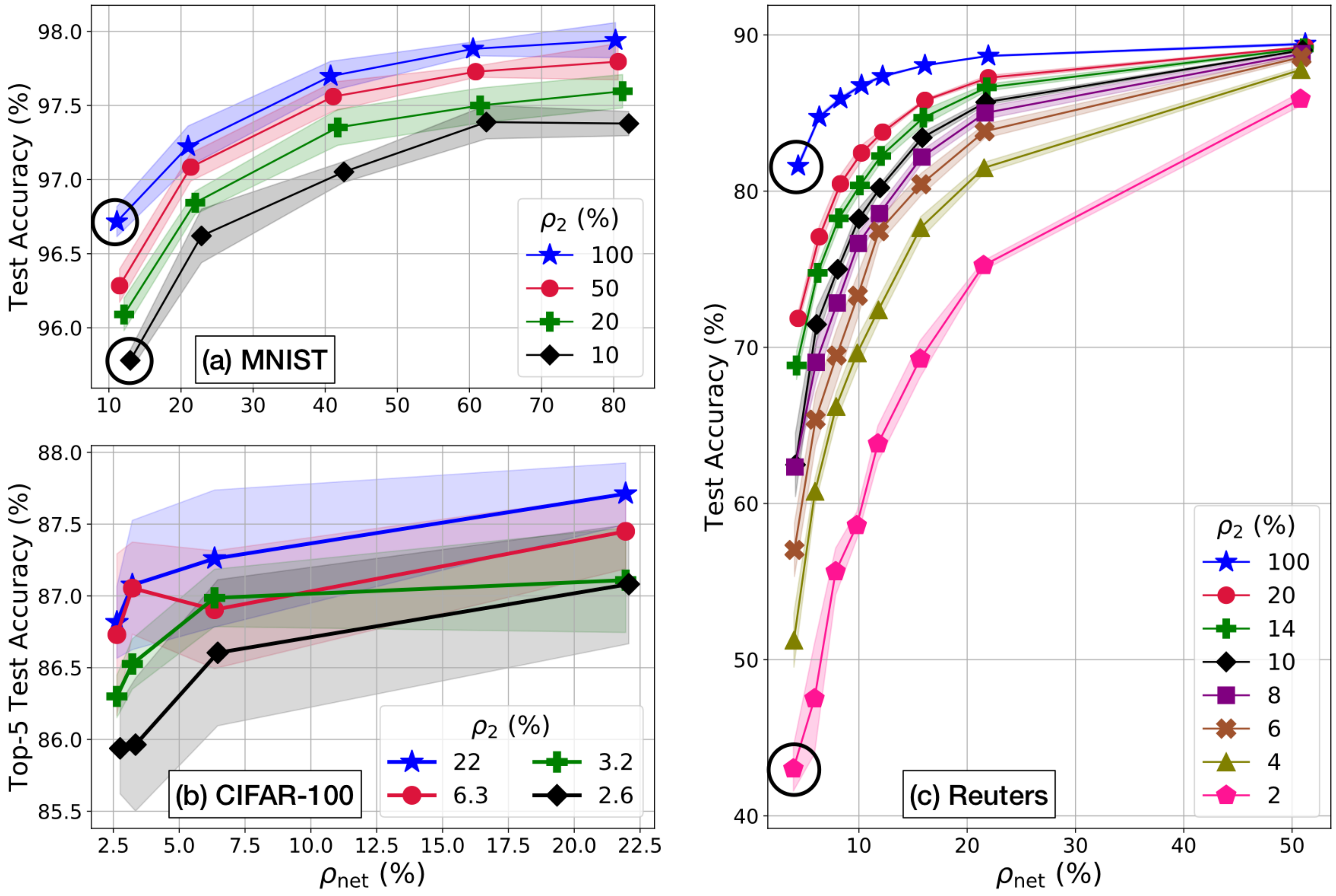
Latter junctions (closer to the output) need to be denser

Individual junction densities

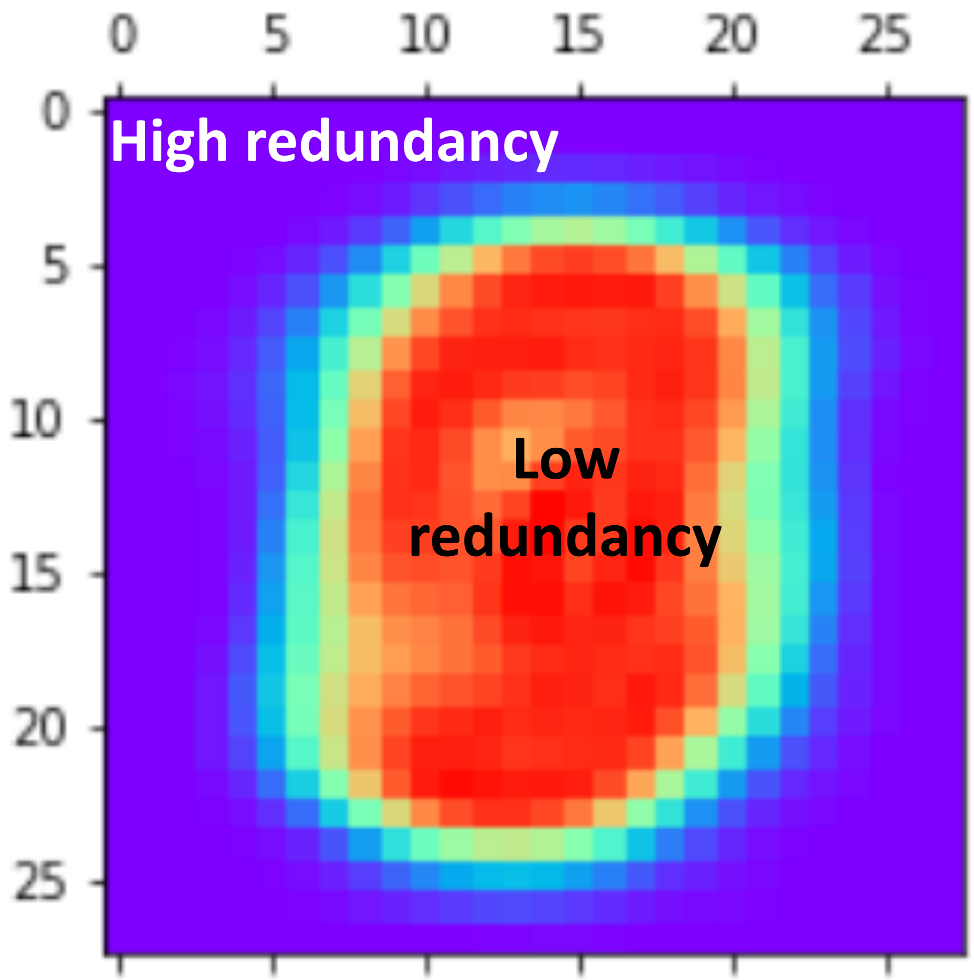
Each curve keeps ρ_2 fixed and varies ρ_{net} by varying ρ_1

For the same ρ_{net}, $\rho_2 > \rho_1$ improves performance

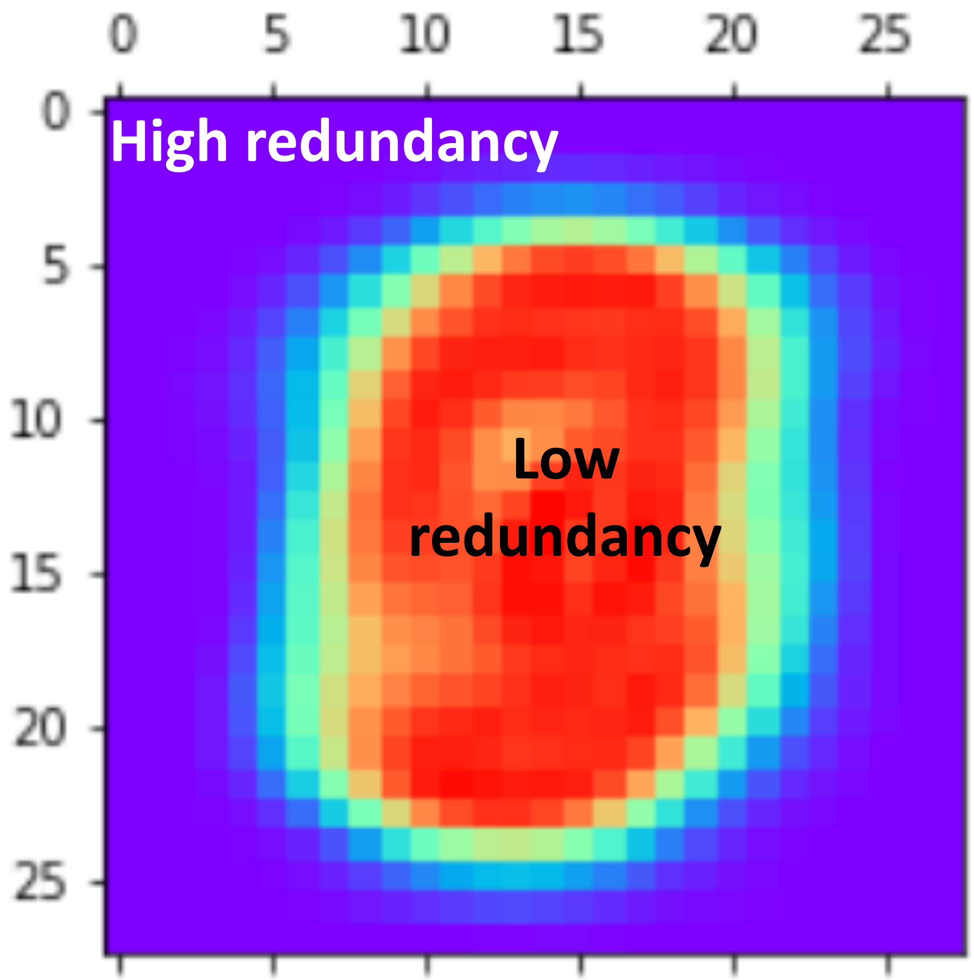
Mostly similar trends observed for deeper networks

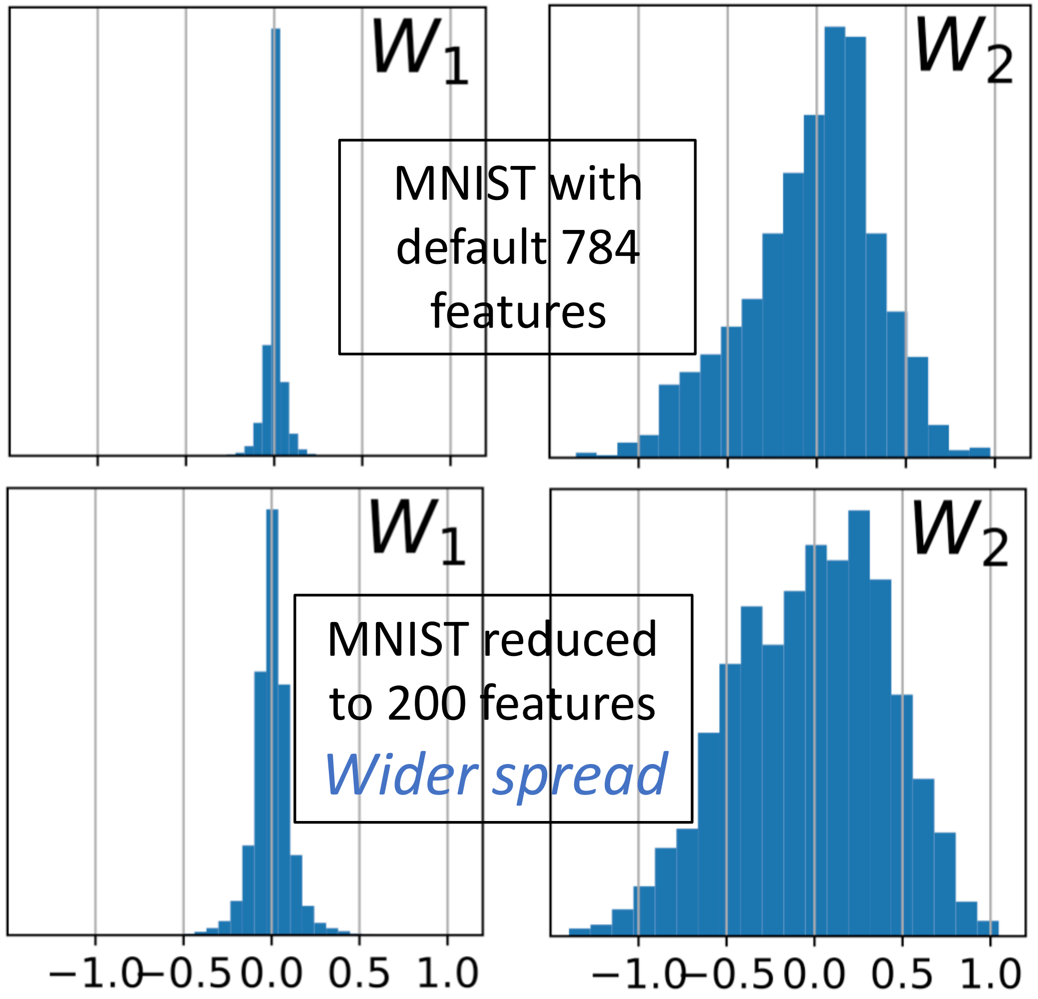


Dataset redundancy



Dataset redundancy

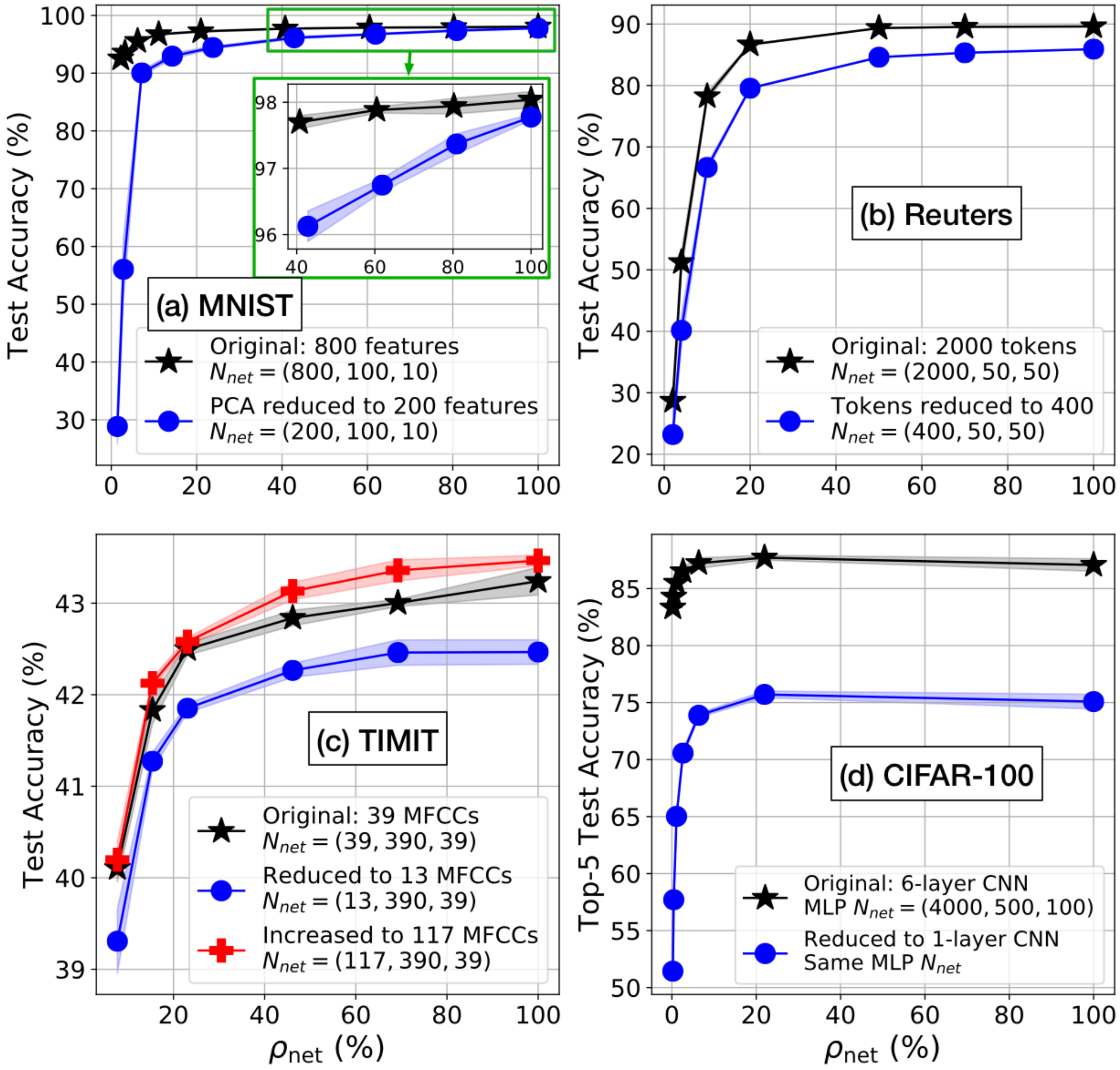




Less redundancy => Less sparsification possible

Effect of redundancy on sparsity

Reducing redundancy leads to increased performance degradation on sparsification

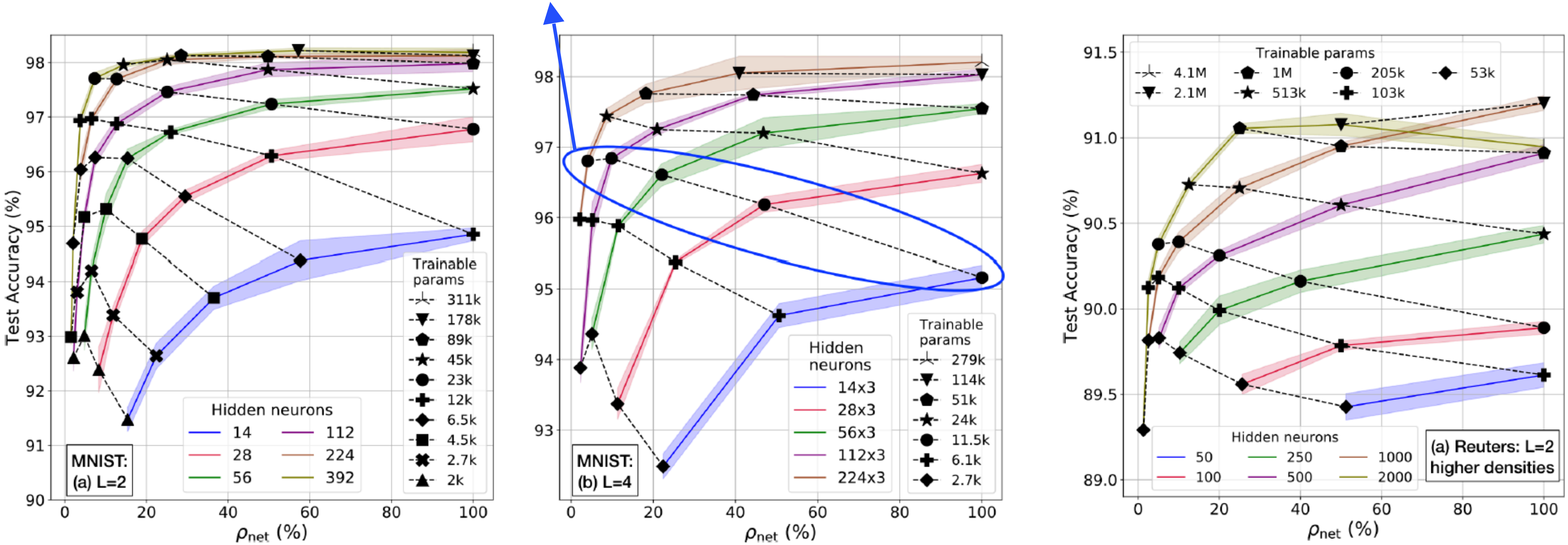


‘Large sparse’ vs ‘small dense’ networks

A sparser network with more hidden nodes will outperform a denser network with less hidden nodes, when both have same number of weights

‘Large sparse’ vs ‘small dense’ networks

Networks with same number of parameters go from bad to good as #nodes in hidden layers is increased



Regularization

$$C(\boldsymbol{w}) = C_0(\boldsymbol{w}) + \lambda \left\| \boldsymbol{w} \right\|_2^2$$

Regularized cost

Original unregularized
cost (like cross-entropy)

Regularization term

Regularization

$$C(\boldsymbol{w}) = C_0(\boldsymbol{w}) + \lambda \left\|\boldsymbol{w}\right\|_2^2$$

Regularized cost

Original unregularized cost (like cross-entropy)

Regularization term

Pre-defined sparse networks need smaller λ (as determined by validation)

Overall Density	λ
100 %	1.1×10^{-4}
40 %	5.5×10^{-5}
11 %	0

Example for MNIST 2-junction networks

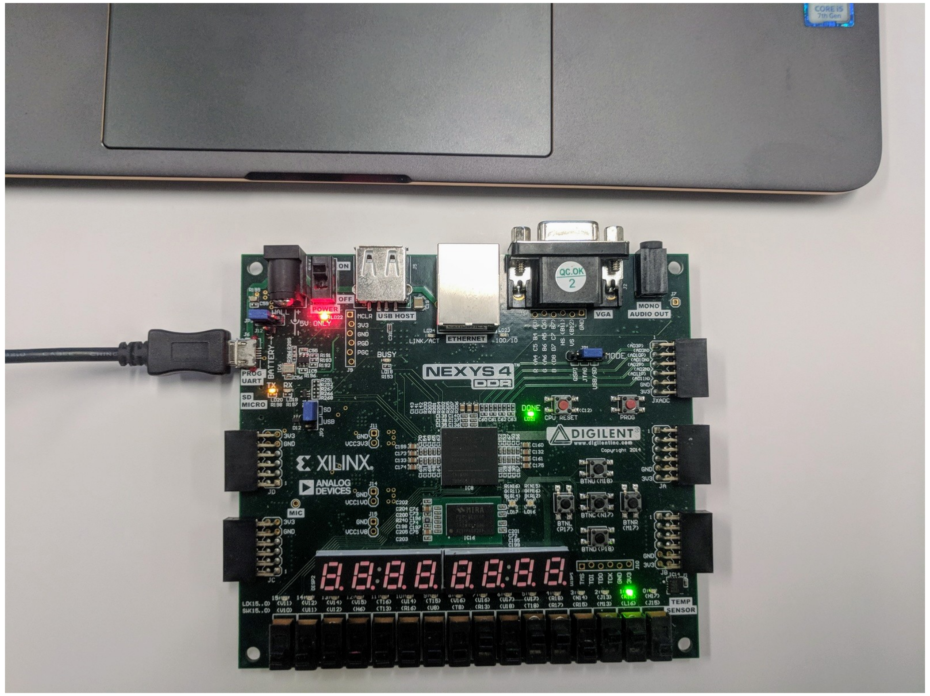
Pre-defined sparsity reduces the overfitting problem stemming from over-parametrization in big networks

Hardware Architecture

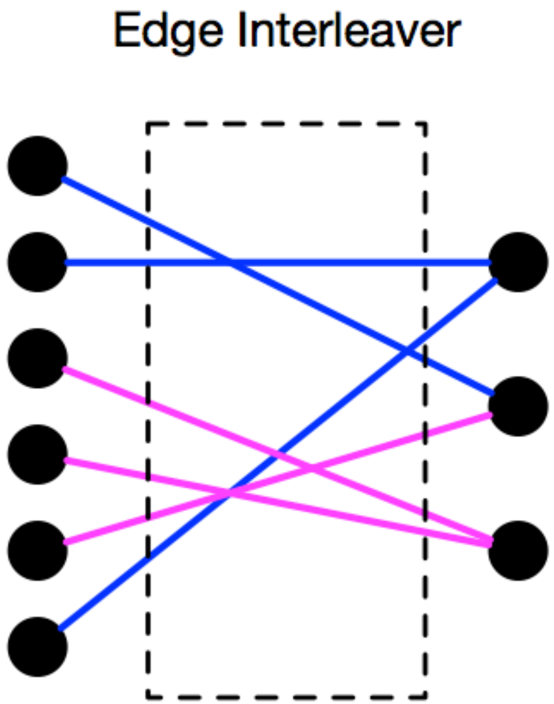
We built a customized hardware architecture to leverage pre-defined sparsity

Key highlights:

- ➢ On-device training
- ➢ Edge-based
- ➢ Customizable amount of parallelism
- ➢ Clash free memory accesses
- ➢ Pipelined processing



Degree of parallelism z



Example $z_i = 3$

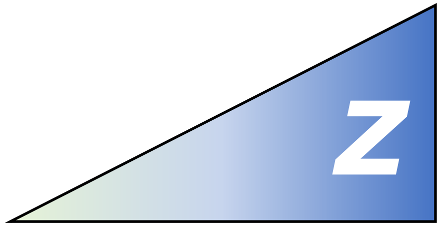
z_i = #edges (weights) processed in parallel in junction i

$$\text{\#clock cycles } (C_i) \text{ to process junction i } = \frac{\text{\#weights } |W_i|}{z_i}$$

Computational complexity depends only on z_i

Decouple hardware required from network complexity

Slow Training



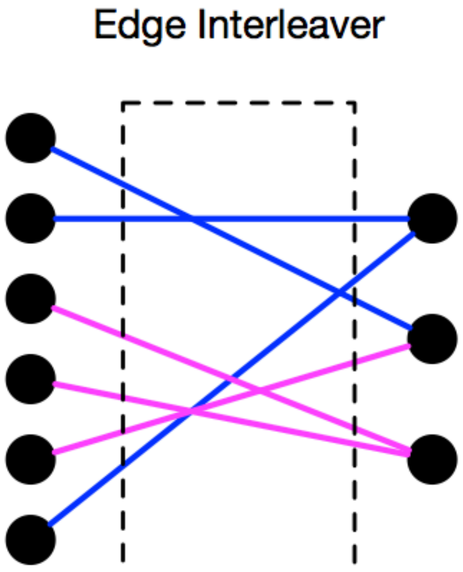
Hardware Intensive

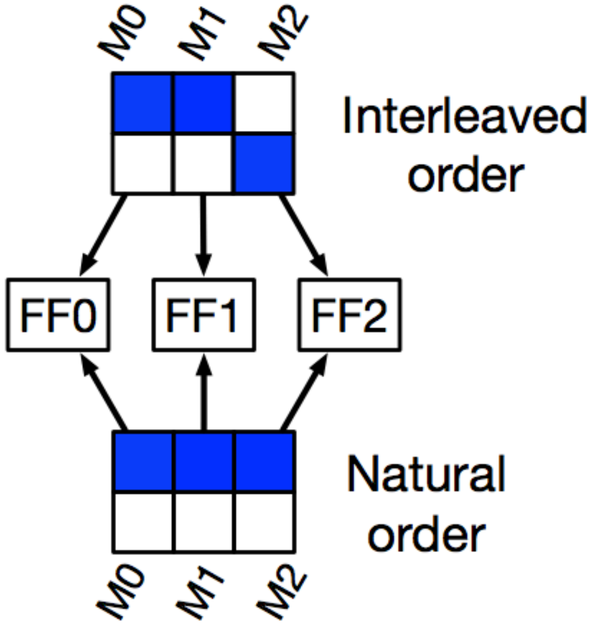
Flexibility

Memory organization and clash freedom

z_i memories for storing each variable in each junction

Must access each memory no more than once per clock cycle, otherwise clash => processing stall





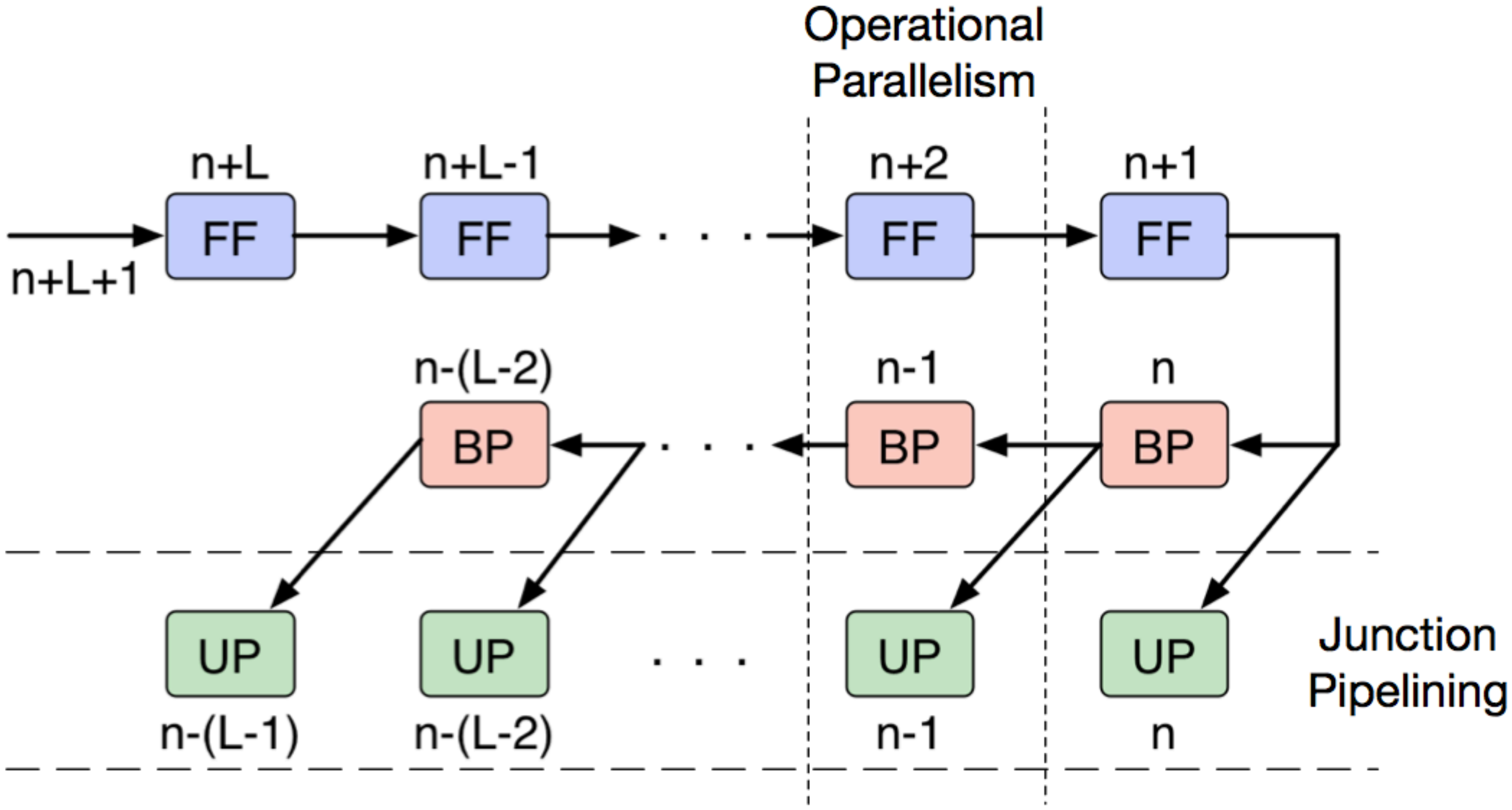
Left side nodes are accessed in arbitrary order due to interleaving

S. Dey, P. A. Beerel and K. M. Chugg, “Interleaver design for deep neural networks,” in *51st Annual Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, pp. 1979-1983, Oct 2017.

Weights are accessed one row at a time

Example z_i = 3

Parallel and Pipelined processing



- *FF, BP, UP operates only on weights which are present*
- *Operational parallelism:* FF, BP, UP simultaneously inside a junction
- *Junction pipelining:* Each operates on different inputs across junctions
- Faster training @ more hardware and storage cost

Model Search

Automate the design of CNNs with good performance and low complexity

Model search is ongoing research, hence currently not available publicly

Thank you!

I plan to graduate in May 2020 and am looking for post-doc openings (primarily on the algorithmic / software side of deep learning)