# NeuroFabric: A Priori Sparsity for Training on the Edge

Mihailo Isakov, Michel A. Kinsy
Adaptive and Secure Computing Systems (ASCS) Laboratory
Boston University

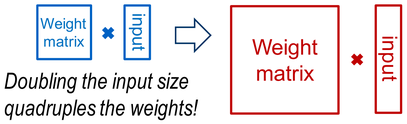
## Introduction

Training networks on the edge is hard, due to low HW performance and power and memory bottlenecks of edge devices. **We propose to reduce the computational complexity of networks by pruning networks ahead of time.**

A dense layer has two properties:

- Information bandwidth (number of output neurons in a layer)
- Layer expressivity (number of connections in the layer)

Dense layers control both properties with one parameter – the number of neurons in the layer. This causes unsustainable growth of the layers as we increase information bandwidth.




*Doubling the input size quadruples the weights!*

Bigger applications will require wider nets - DNNs will grow quadratically! CNN and RNNs also suffer:

- **CNNs:** Doubling the number of input and output channels in a convolutional layer will quadruple the number of convolutions
- **RNNs:** If an RNN layer needs (1) a wider input window, or (2) more memory, the number of connections grows quadratically

**We propose to disentangle layer bandwidth and expressivity.** With separate parameters for each property, the user can select a network that is wide enough to solve a task, but uses the minimal amount of compute power needed.

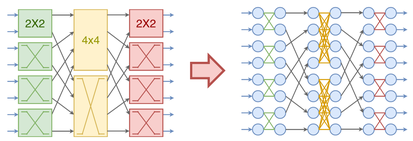
## Previous Work

Pruning neural networks can reduce their size by 95%

- **Only applicable after training**
- Pruned connections waste all energy used to train them

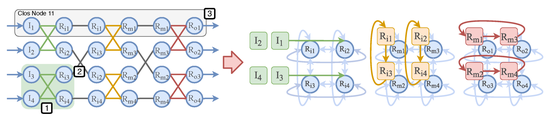
We take inspiration from Network on Chip architectures

- Instead of all-to-all connections, we create a "network" for each layer



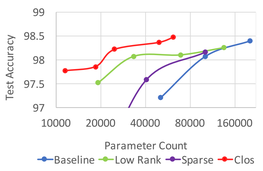

ClosNets break a dense layer into a cascade of three sparse layers with the Clos topology. We use the Clos topology as:

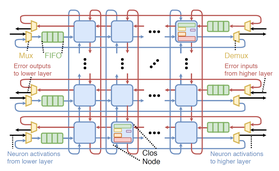
- Guarantees full connectivity
- Is shallow
- Has high path diversity
- Has efficient HW implementation



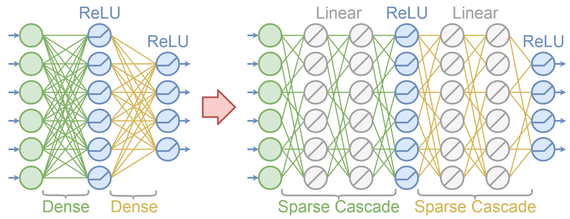

Our ClosNets work did not answer:

- If there existed better topologies
- How to train deeper, sparser topologies




## NeuroFabric Approach




NeuroFabric is a method for compressing linear and convolutional layers **ahead** of training. It reduces both the computation and memory requirements of layers by breaking them down into **a priori structured sparse cascades**.

Since we don't know the training data, we cannot have a dataset specific topology – the topology must work on all datasets.

**NeuroFabric therefore allows networks to decouple layer information bandwidth (number of output neurons in a layer) from layer expressivity (number of parameters in the layer).**

By breaking dense layers into sparse cascades, we:

- Minimize memory and compute bottlenecks
- Allow scaling networks without quadratic growth in size

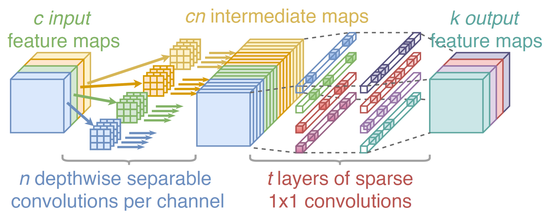
By knowing the topology in advance:

- There is no need for storing element indices
- Memory accesses can be made sequential
- We can minimize data movement by selecting good topologies

The main questions we ask are:

1. **How can we train very deep, sparse topologies?**
2. **What topology should we choose?**
3. **How deep should the cascade be?**

NeuroFabric can also be applied to CNNs!

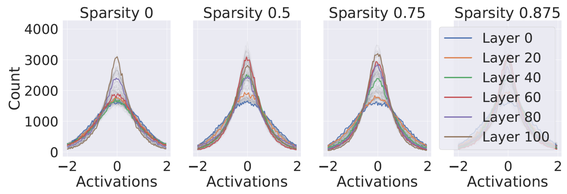



## Sparse Glorot Initialization

Conventional Glorot (Xavier) initialization leads to vanishing gradient problems when used in a priori sparse networks. After diagnosing this issue, we update the initialization as follows:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{(n_{in}+n_{out})}\sqrt{(1-s)}}, \frac{\sqrt{6}}{\sqrt{(n_{in}+n_{out})}\sqrt{(1-s)}}\right]$$

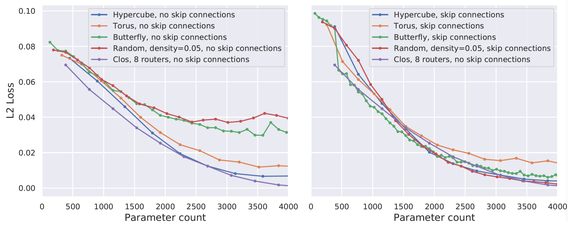
This "Sparse Glorot" initialization allows us to train linear sparse networks of any depth.




## Evaluating Topologies

We reduce the problem of finding the best intra-layer topology of a network for to a sparse decomposition task, where an original dense matrix $W_o$ is decomposed as a product of sparse matrices:

$$L_r = \left\| W_o - \prod_{i=0}^{l} W_i M_i \right\|_2, \; W_i \in \mathbb{R}^{n \times n}, \; M_i \in [0,1]^{n \times n}$$

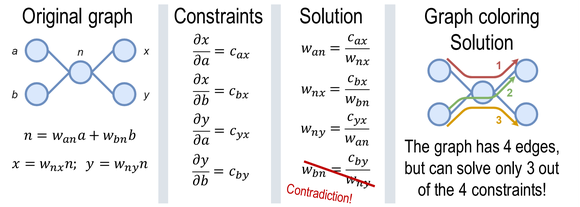



Not all topologies perform equally well. **We seek a unifying heuristic that predicts how a certain topology will behave.**

We reduce the sparse reconstruction problem to a constraint satisfaction problem. By using L1 loss, SGD will exactly reconstruct some of the $W_o$ values. We can count those values and treat each matrix value as a separate constraint, with the goal of satisfying the largest number of constraints. We rewrite the matrix equation for sparse decomposition as:

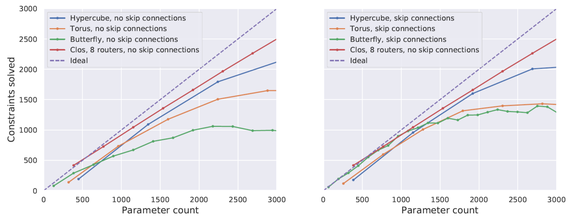
$$W_r = \prod_{i=0}^{l} W_i M_i, \quad W_r[x,y] = \sum_{p}^{P(x,y)} \prod_{e_{ij} \in p}^{p} w_{ij}$$

We interpret the equation as: a reconstructed element $W_r[x,y]$ is a sum of products across all paths from input *x* to output *y*. Since setting just one element on one path is enough to satisfy a constraint, **the number of solvable constraints is lesser or equal to the number of parameters in the topology**. From here, we develop a graph coloring based heuristic for estimating the number of constraints a topology can solve. As an example:



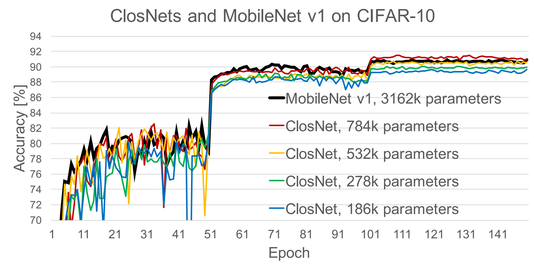

**Graph coloring rules:**

1. If a path is drawn from an input to an output, that constraint is solved.
2. A path can be drawn only if there is at least one uncolored edge on it.
3. A path colors all edges it covers.
4. The intersection of two paths must be a connected component.




## Deriving Better Topologies

- From the constraint solving graph coloring, we see that deep networks can waste edges
  - Skip connections allow paths to connect input and output pairs without using (coloring) multiple edges
- Skip connections help with parameter efficiency (the number of constraints solved per parameter)
- Even with skip connections, some topologies have a hard time satisfying all constraints
  - Solving all constraints is an NP-hard problem
  - Butterfly networks underperform as at high depth, it is difficult to find an open path through the graph
- Clos performs well because of (1) ideal parameter efficiency, (2) shallowness and (3) high path diversity
  - Clos networks with skip connections have ideal parameter efficiency
  - 4-stage Clos networks do not provide a benefit as path diversity is already sufficiently high




## Results & Key Contributions

Our key contributions are:

- We show a method for reducing the memory and compute requirements of dense, convolutional and recurrent layers
  - This allows scaling networks to bigger problems without quadratically growing the network size
- We develop new initialization strategies for sparse networks
- We provide a heuristic for comparing topologies, and show that shallow topologies with high path diversity like the Clos network outperform all other topologies
  - We provide an intuition of why butterfly and other deep and parameter-efficient topologies underperform
- We show how a priori sparse networks can be applied to CNNs
- We propose efficient GPU kernels that can process Clos networks with very little overhead, despite high sparsity

## Future Directions of Research

1. **Densifying networks during training:** the network topology gains edges as training progresses
2. **Network Calcification:** older, trained connections get frozen, and become immutable during future training epochs
3. **On-chip training:** removing DRAM from the equation
4. **Inverse plasticity:** dynamic reconfiguration on FPGAs allows moving old weights from SRAM to FPGA LUTs

## References

M. Isakov, A. Ehret and M. Kinsy, "**ClosNets: Batchless DNN Training with On-Chip a Priori Sparse Neural Topologies**," *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, Dublin, 2018, pp. 55-554.