PHEW: Constructing Sparse Networks that Learn Fast and Generalize Well **Without Training Data**

Shreyas Malakarjun Patil, Constantine Dovrolis

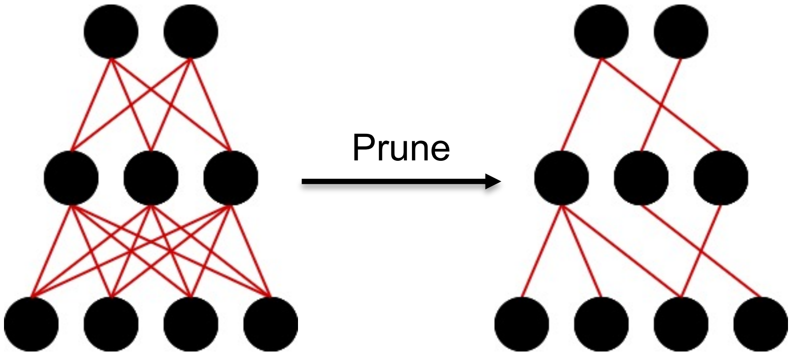
Georgia Tech

Georgia Tech School of Computer Science College of Computing

ICML International Conference On Machine Learning

Sparse neural networks at initialization

- Similar performance to dense neural networks
- Lower training and inference costs

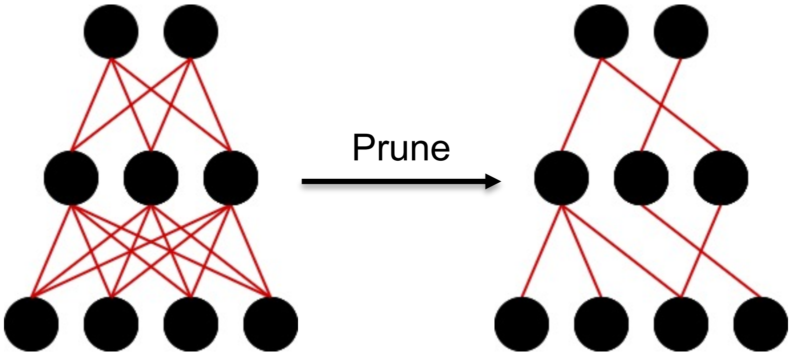


Sparse neural networks at initialization

Pruning networks prior to training :

- **Using Training Data :** SNIP [Lee et al. ICLR 2019], GraSP [Wang et al. ICLR 2020] etc.

- **Without Using Training Data :** SynFlow [Tanaka et al. NeurIPS 2020], SynFlow-L2

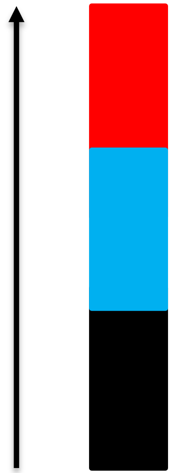


Generalized :

- Subnetworks Across Datasets
- Methods Across Tasks

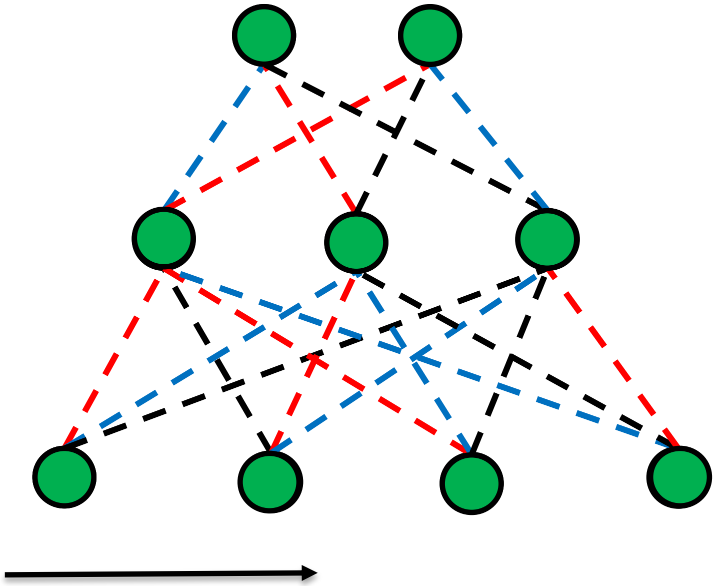
PHEW: P*aths with* H*igher* E*dge*-W*eights*

Consider a randomly initialized neural network and a target number of weights / parameters ($m = 12$)



Increasing Weight Magnitude

Number of Weights : 0 / 12

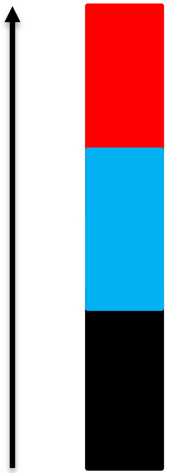


Starting unit selected through round robin

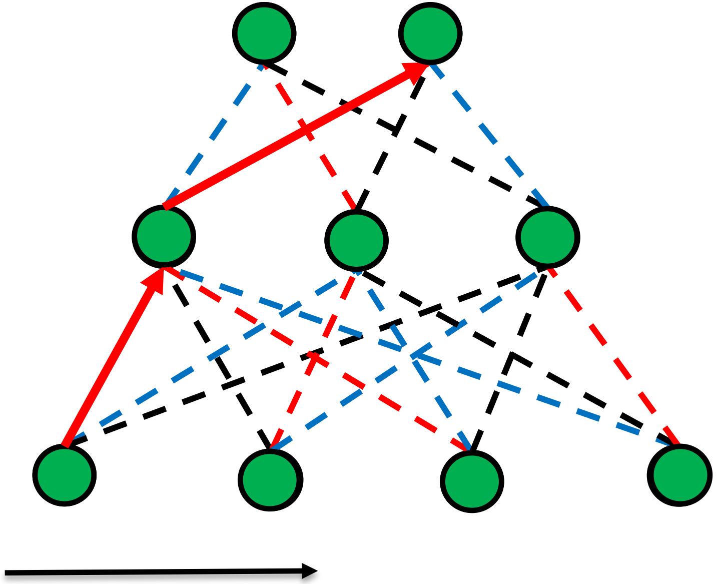
PHEW: P*aths with* H*igher* E*dge*-W*eights*

Consider a randomly initialized neural network and a target number of weights / parameters ($m = 12$)

Number of Weights : 2 / 12



PHEW selects a set of **input-output paths** to be conserved



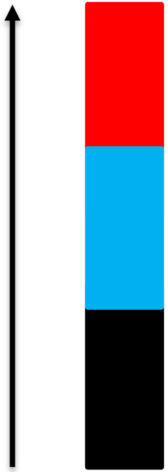
Increasing Weight Magnitude

Starting unit selected through round robin

PHEW: P*aths with* H*igher* E*dge*-W*eights*

Consider a randomly initialized neural network and a target number of weights / parameters ($m = 12$)

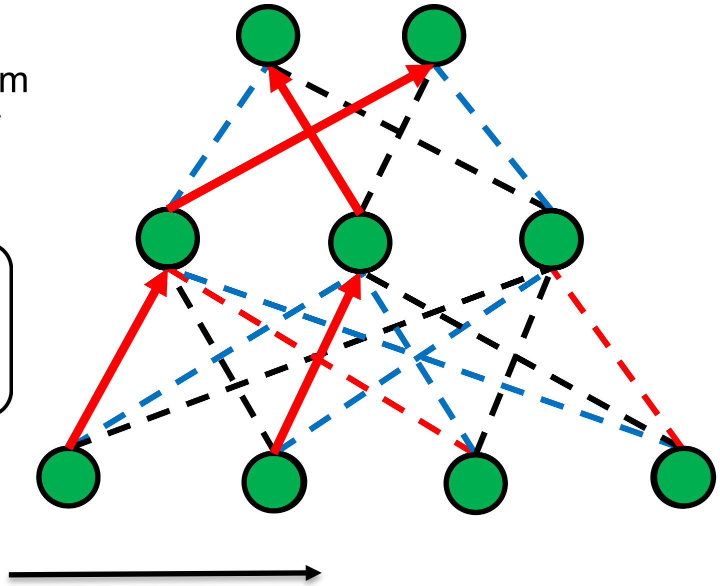
Number of Weights : 4 / 12



Increasing Weight Magnitude

Path selection through random walks, biased towards higher weight magnitudes

$$Q(j,i) = \frac{|\theta(j,i)|}{\sum_j |\theta(j,i)|}$$

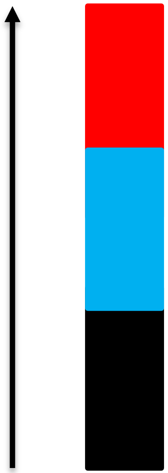


Starting unit selected through round robin

PHEW: P*aths with* H*igher* E*dge*-W*eights*

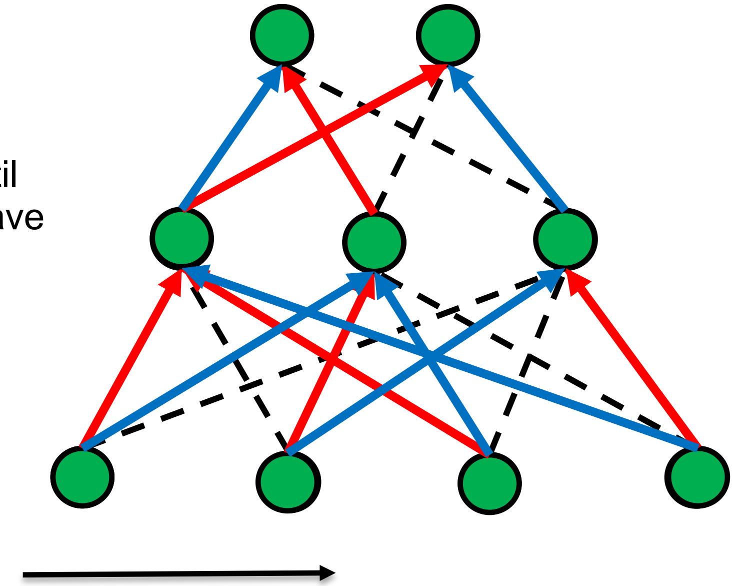
Consider a randomly initialized neural network and a target number of weights / parameters ($m = 12$)

Number of Weights : 12 / 12



Increasing Weight Magnitude

Random walks continue until target number of weights have ben selected.

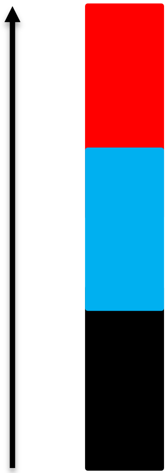


Starting unit selected through round robin

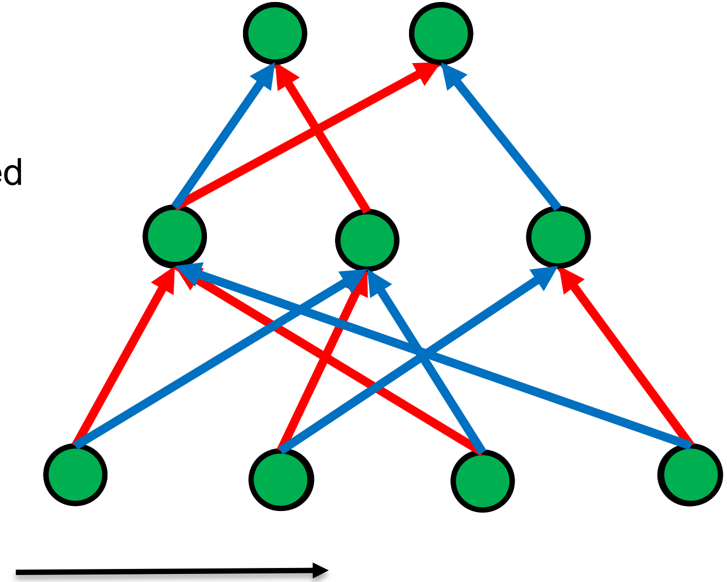
PHEW: P*aths with* H*igher* E*dge*-W*eights*

Consider a randomly initialized neural network and a target number of weights / parameters ($m = 12$)

Number of Weights : 12 / 12



Remove weights not selected through the random walks



Increasing Weight Magnitude

Starting unit selected through round robin

Why do we select edges with high weight magnitudes ?

Larger number of paths and higher weight magnitudes leads to faster convergence

A Unified Paths Perspective for Pruning at Initialization

Thomas Gebhart*
Department of Computer Science
University of Minnesota
gebhart@umn.edu

Udit Saxena*
Sumo Logic
usaxena@sumologic.com

Paul Schrater
Department of Computer Science
University of Minnesota
schrater@umn.edu

Larger number of paths and higher weight magnitudes leads to faster convergence

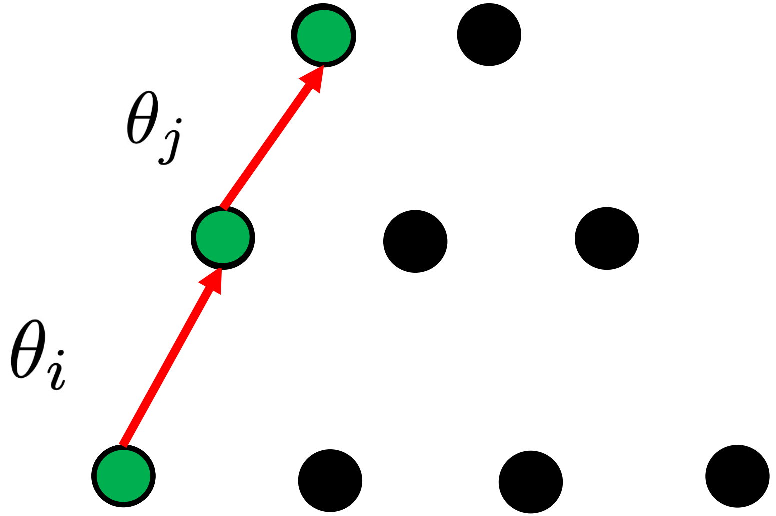
Let us consider a ReLU network at initialization, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}),\ \boldsymbol{\theta} \in \mathbb{R}^m$

An input-output path, p

Edge-Weight Product for path p,

$$\boldsymbol{\pi}_p(\boldsymbol{\theta}) = \prod_{k=1, \theta_k \in p}^{m} \theta_k = \theta_i \times \theta_j$$

k : Edge, θ_k : Weight of Edge



Larger number of paths and higher weight magnitudes leads to faster convergence

Let us consider a ReLU network at initialization, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}),\ \boldsymbol{\theta} \in \mathbb{R}^m$

An input-output path, p

Edge-Weight Product for path p,

$$\boldsymbol{\pi}_p(\boldsymbol{\theta}) = \prod_{k=1, \theta_k \in p}^{m} \theta_k = \theta_i \times \theta_j$$

k : Edge, θ_k : Weight of Edge

Path Kernel element for two paths p and q,

$$\boldsymbol{\Pi}_{\boldsymbol{\theta}}(p, q) = \sum_{k=1}^{m} \frac{\partial \pi_p(\boldsymbol{\theta})}{\partial \theta_k} \frac{\partial \pi_q(\boldsymbol{\theta})}{\partial \theta_k}$$

Larger number of paths and higher weight magnitudes leads to faster convergence

Let us consider a ReLU network at initialization, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}),\ \boldsymbol{\theta} \in \mathbb{R}^m$

An input-output path, p

Edge-Weight Product for path p,

$$\boldsymbol{\pi}_p(\boldsymbol{\theta}) = \prod_{k=1, \theta_k \in p}^{m} \theta_k = \theta_i \times \theta_j$$

Path Kernel element for two paths p and q,

$$\boldsymbol{\Pi}_{\boldsymbol{\theta}}(p, q) = \sum_{k=1}^{m} \frac{\partial \pi_p(\boldsymbol{\theta})}{\partial \theta_k} \frac{\partial \pi_q(\boldsymbol{\theta})}{\partial \theta_k}$$

$$Tr(\boldsymbol{\Pi}_{\boldsymbol{\theta}}) = \sum_p \boldsymbol{\Pi}_{\boldsymbol{\theta}}(p, p) = \sum_p \sum_{k=1, \theta_k \in p}^{m} \left(\frac{\pi_p(\boldsymbol{\theta})}{\theta_k} \right)^2$$

Larger number of paths and higher weight magnitudes leads to faster convergence

$$Tr(\mathbf{\Pi}_{\boldsymbol{\theta}}) = \sum_p \mathbf{\Pi}_{\boldsymbol{\theta}}(p,p) = \sum_p \sum_{k=1,\theta_k \in p}^{m} \left(\frac{\pi_p(\boldsymbol{\theta})}{\theta_k}\right)^2$$

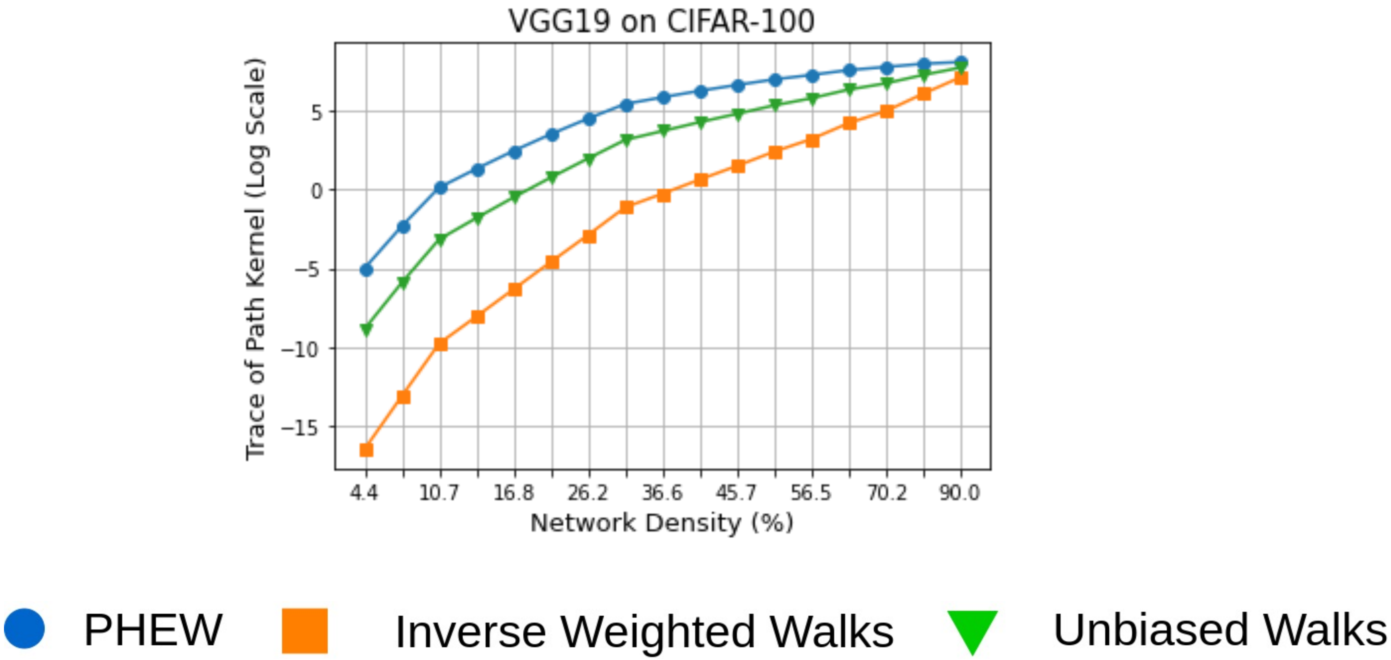
Subnetworks with higher path kernel trace are **expected** to converge faster
[Genhart et al. 2021]

Path kernel trace increases with :

- Number of paths
- Edge-Weight-Product Magnitude of the Paths

[1] : Gebhart, Thomas, Udit Saxena, and Paul Schrater. "A Unified Paths Perspective for Pruning at Initialization." *arXiv preprint arXiv:2101.10552* (2021).

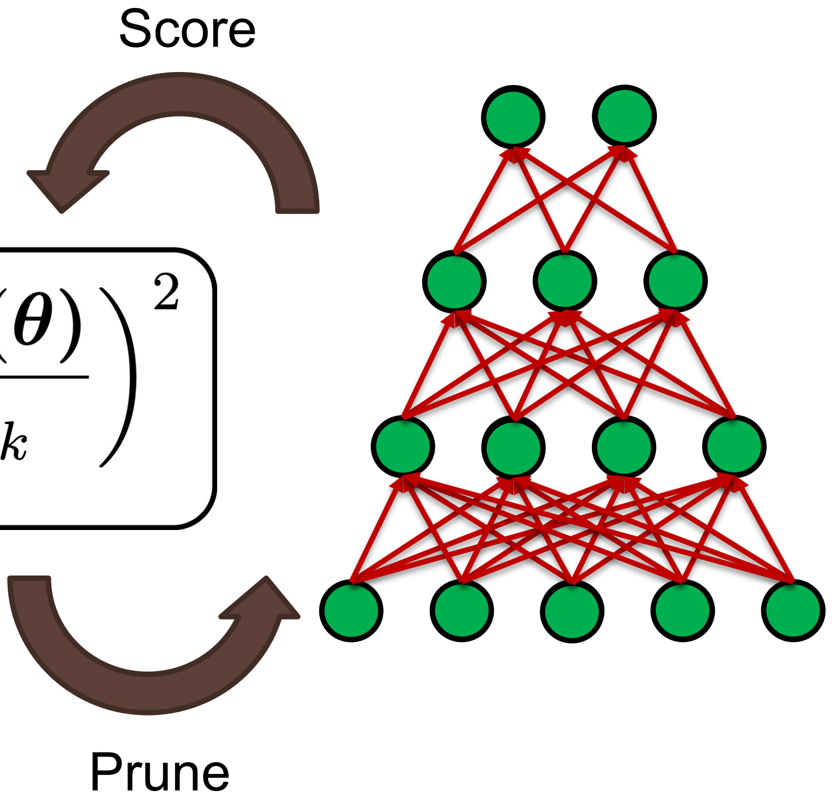
PHEW attains larger path kernel trace than random paths



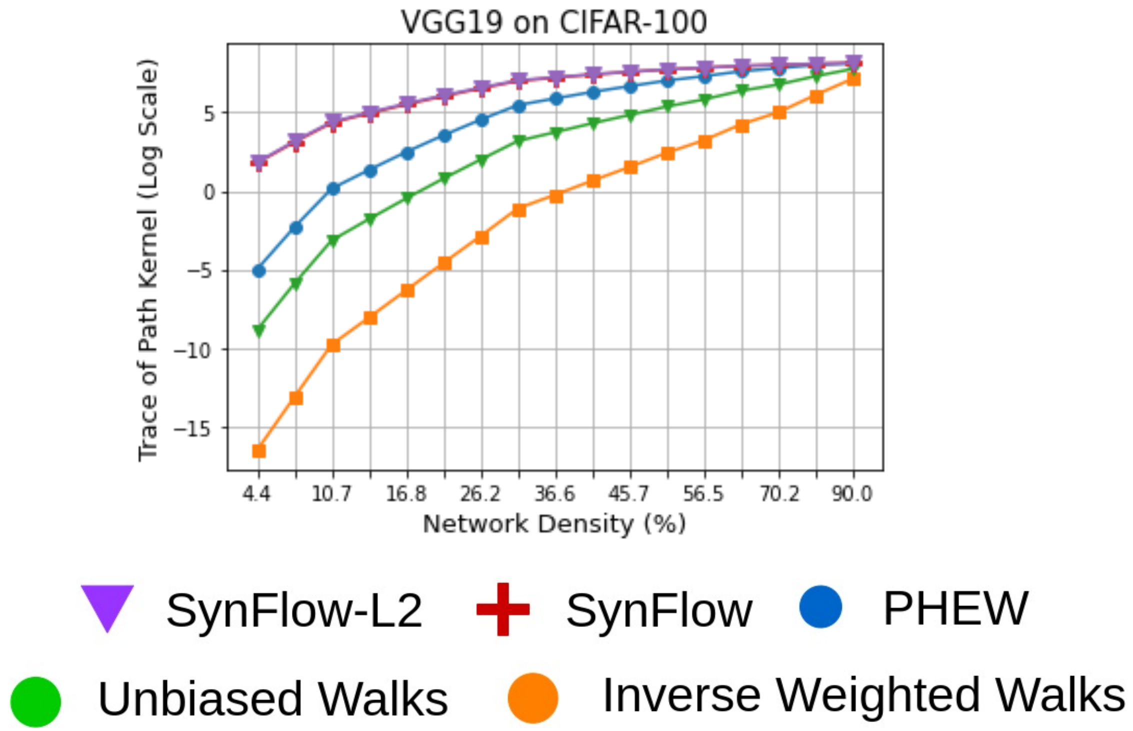
Why not maximize the path kernel trace?

SynFlow-L2 maximizes the path kernel trace

$$S(\theta_k) = \theta_k \odot \sum_{p, \theta_k \in p} \left(\frac{\pi_p(\boldsymbol{\theta})}{\theta_k} \right)^2$$



SynFlow-L2 maximizes the path kernel trace

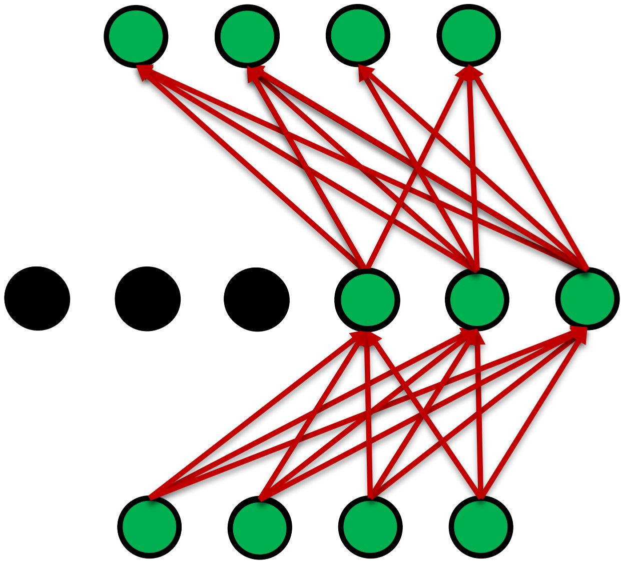


Optimizing just the path kernel trace produces narrow layers

$$Tr(\boldsymbol{\Pi_\theta}) = \sum_p \boldsymbol{\Pi_\theta}(p,p) = \sum_p \sum_{k=1,\theta_k \in p}^{m} \left(\frac{\pi_p(\boldsymbol{\theta})}{\theta_k}\right)^2$$

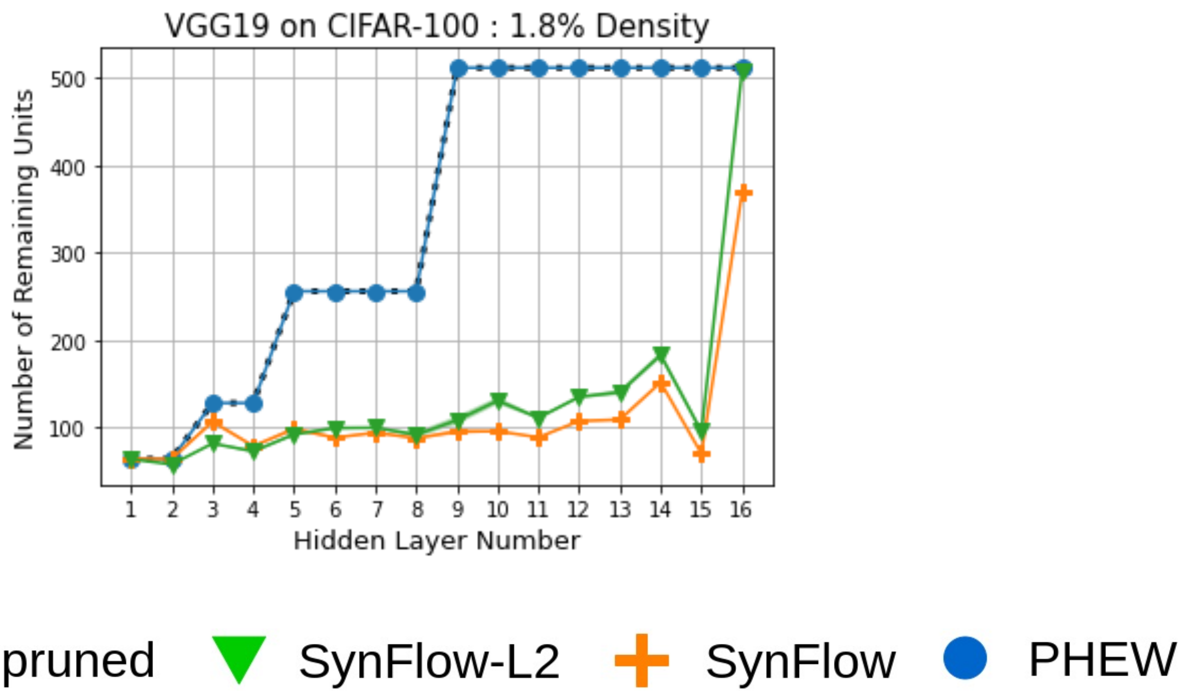
Subnetwork with maximum path kernel trace :

- The lowest possible width
- The highest number of paths



Single hidden layered network

Optimizing just the path kernel trace produces narrow layers



Larger per-layer width improves performance

Published as a conference paper at ICLR 2021

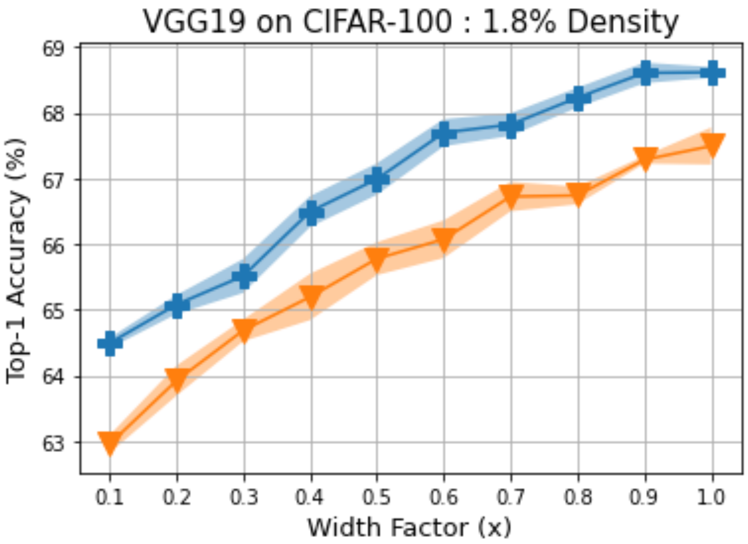
ARE WIDER NETS BETTER GIVEN THE SAME NUMBER OF PARAMETERS?

Anna Golubeva*
Perimeter Institute for Theoretical Physics
Waterloo, Canada
agolubeva@pitp.ca

Behnam Neyshabur
Blueshift, Alphabet
Mountain View, CA
neyshabur@google.com

Guy Gur-Ari
Blueshift, Alphabet
Mountain View, CA
guyga@google.com

Larger per-layer width improves performance

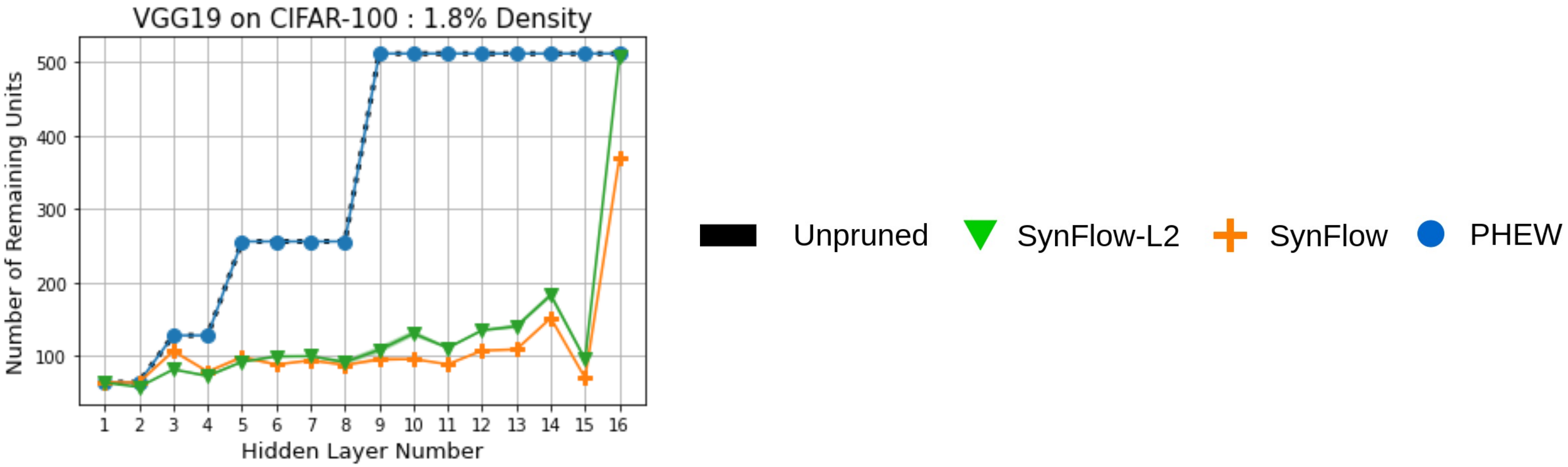


SynFlow : Layer-Wise Mask Shuffling

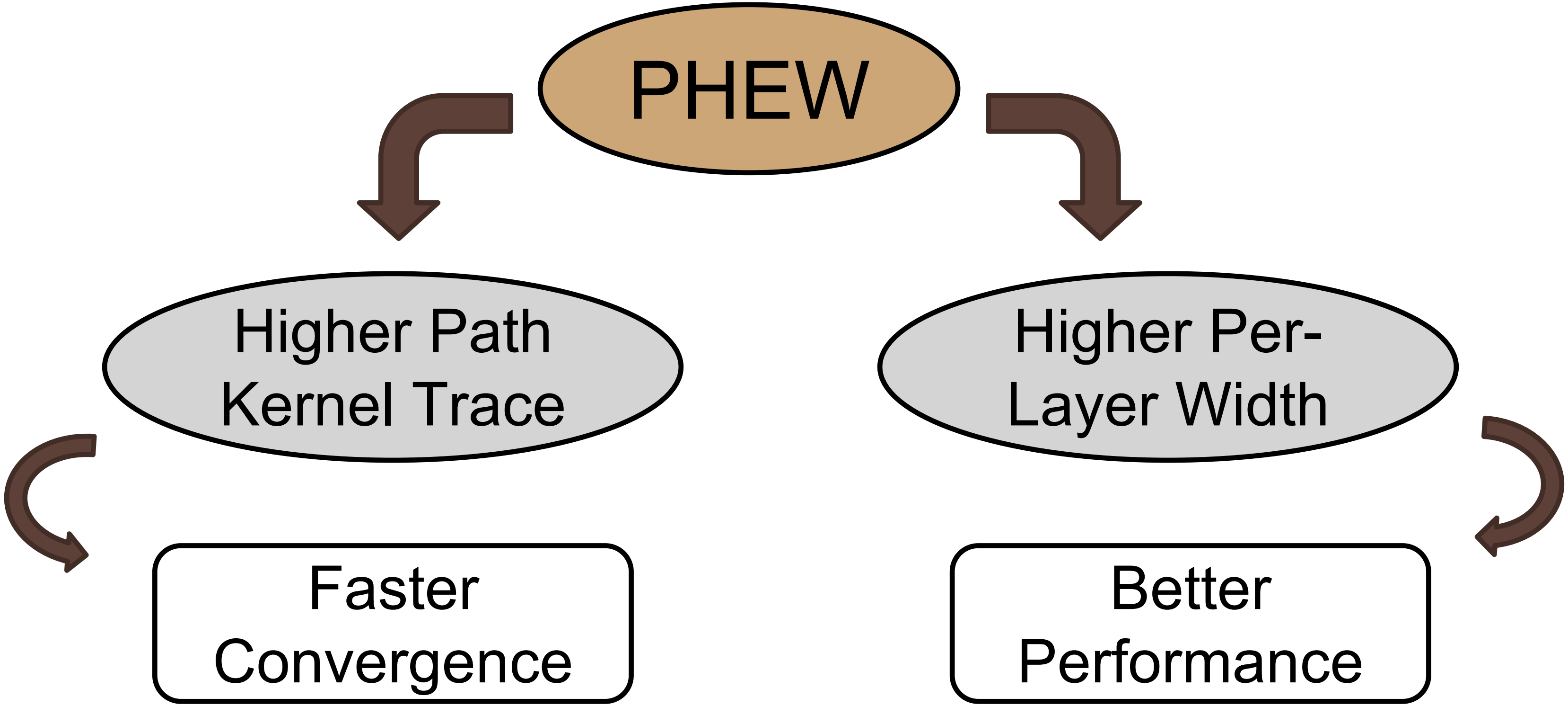
SynFlow-L2 : Layer-Wise Mask Shuffling

PHEW achieves larger per-layer width due to randomization

Given the required number of walks W and N_l, number of units in layer l ,
the expected number of random walks through each unit of a layer l is : $\frac{W}{N_l}$

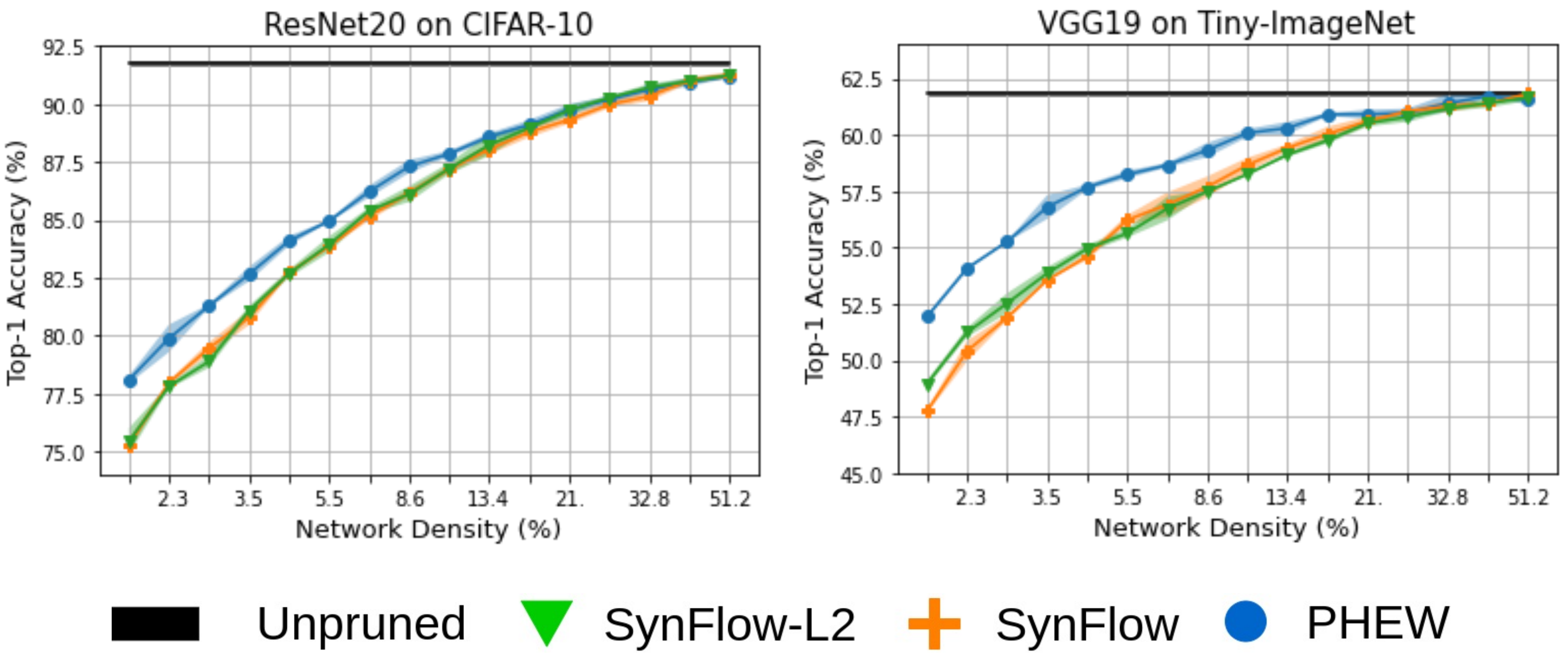


PHEW : faster convergence and better performance

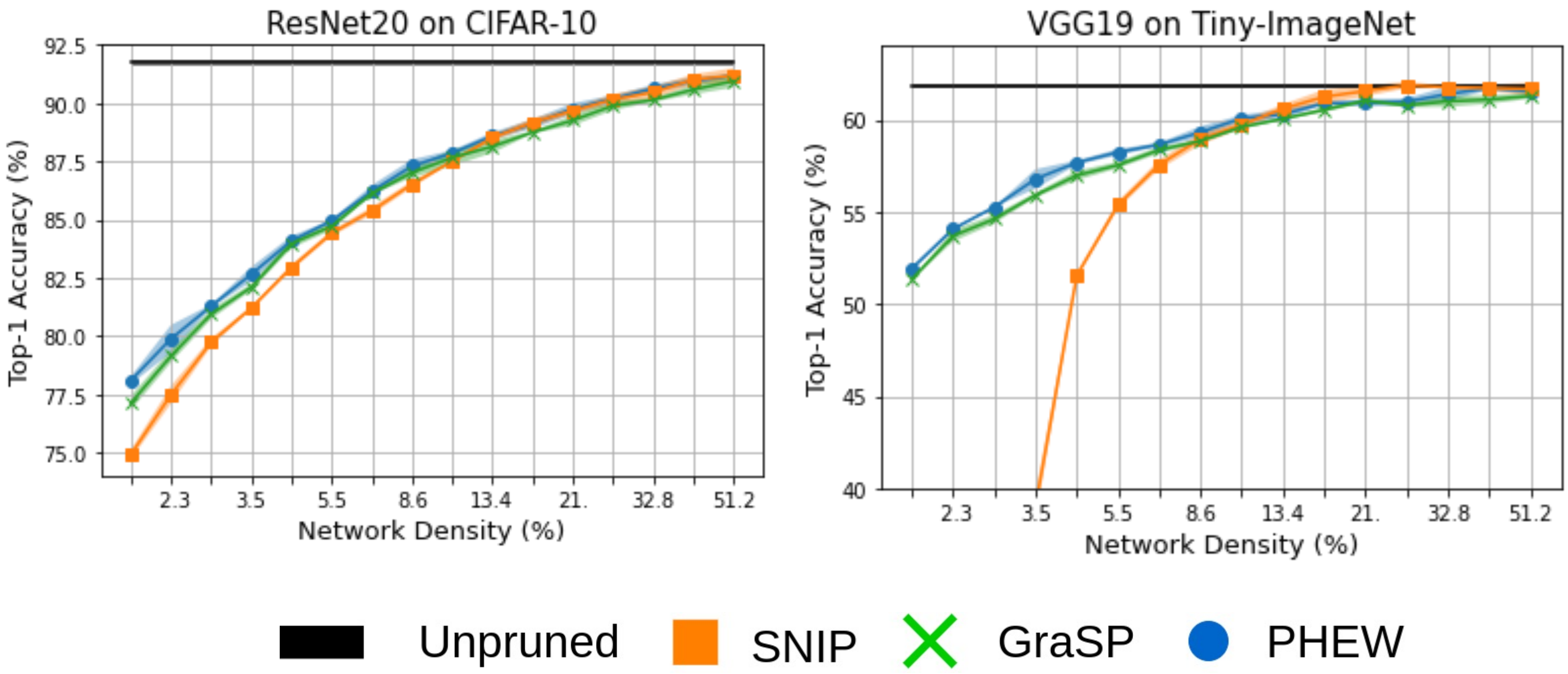


Experiments and Results

Accuracy gap increases with number of classes



PHEW a good alternative to data-dependent SNIP and GraSP



Conclusion and future research questions

- Exploring more **path-based** network construction algorithms at different points in time while training.
  - Using limited amounts of training data
  - Dynamically changing connectivity throughout training
- How to dynamically determine the optimal number of parameters in a sparse network ?
  - Rather than starting with with a given target number of parameters