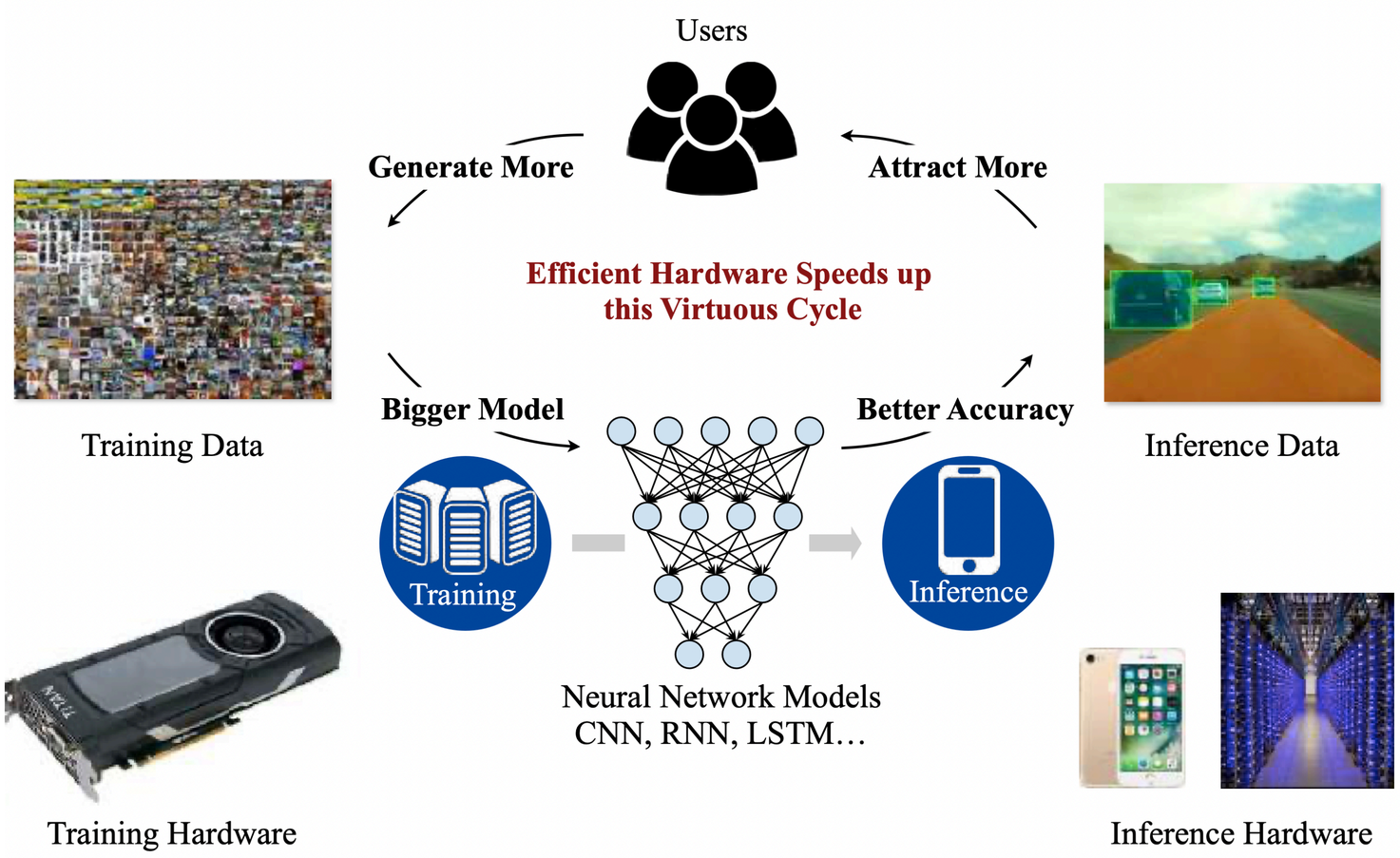
Deep Compression: Compressing Deep Neural networks with Pruning, Trained Quantization and Huffman Coding

Song Han, Huizi Mao, and William J. Dally

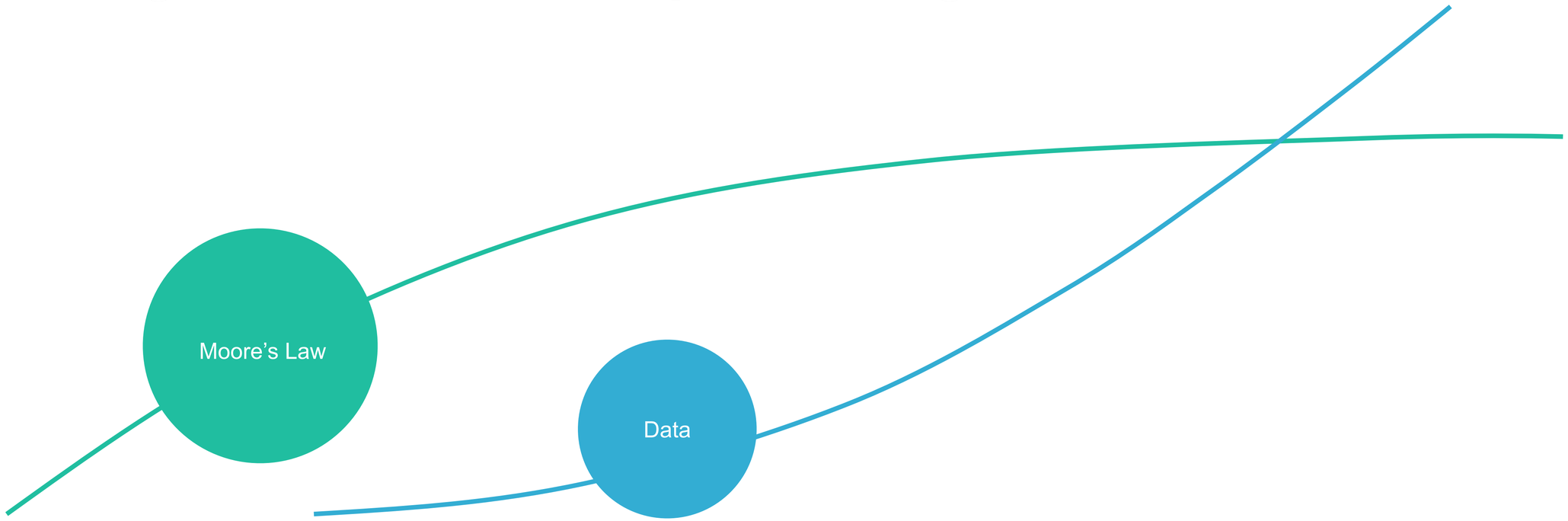
Presenter: Muyang Li

Hardware Plays an Important Role in Modern Deep Learning



Song Han, Stanford PhD thesis. https://purl.stanford.edu/qf934gh3708

A Challenge for Modern Deep Learning



- We are solving more complicated AI problems with larger datasets, which **requires more computation**.

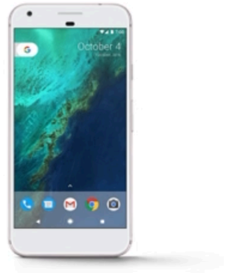
- However, Moore's Law is slowing down; the amount of computation per unit cost is **no longer increasing** at its historic rate.

We Do Need AI on Edge Devices

Low Latency

Low Cost

User Privacy

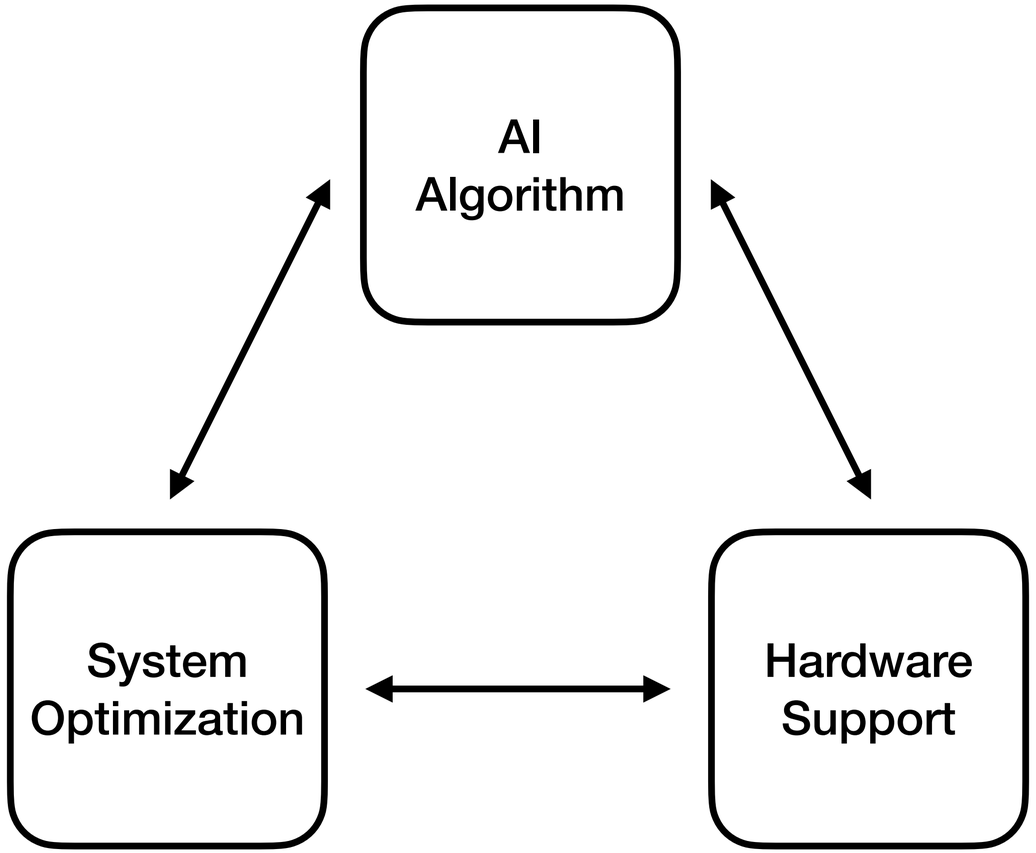


Edge devices only have limited resources!!!

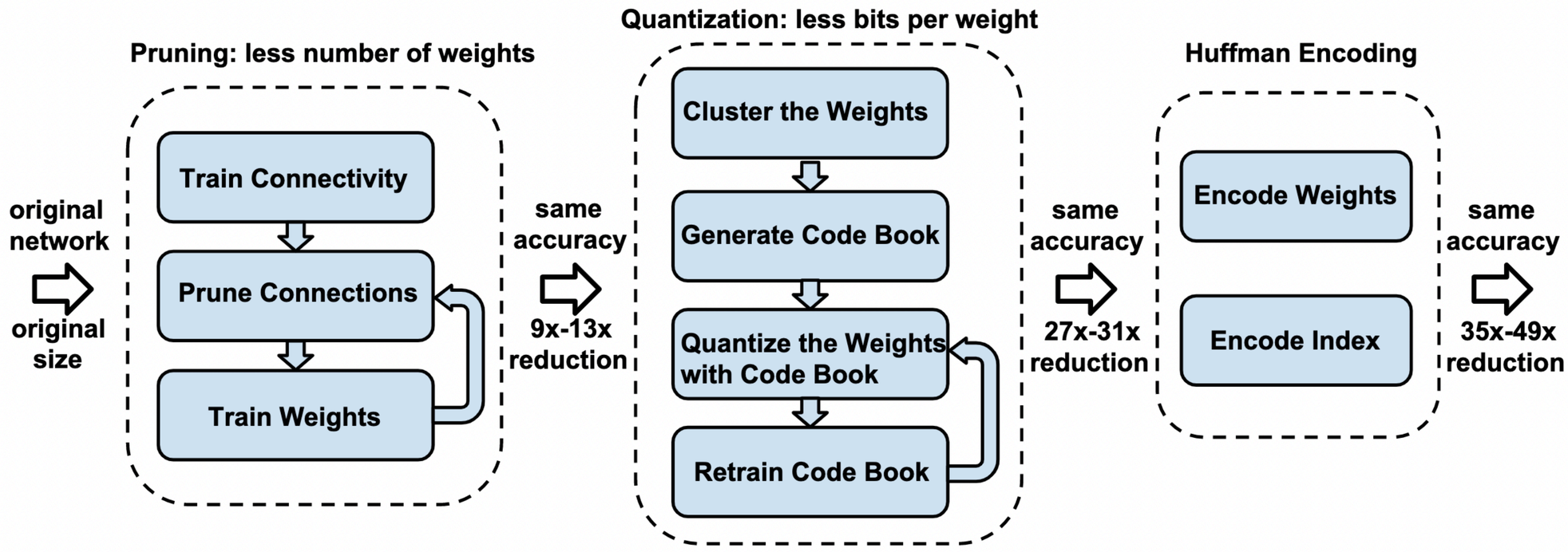
Power-Hungry

AI is good, but AI is slow!

Ways to Accelerate AI

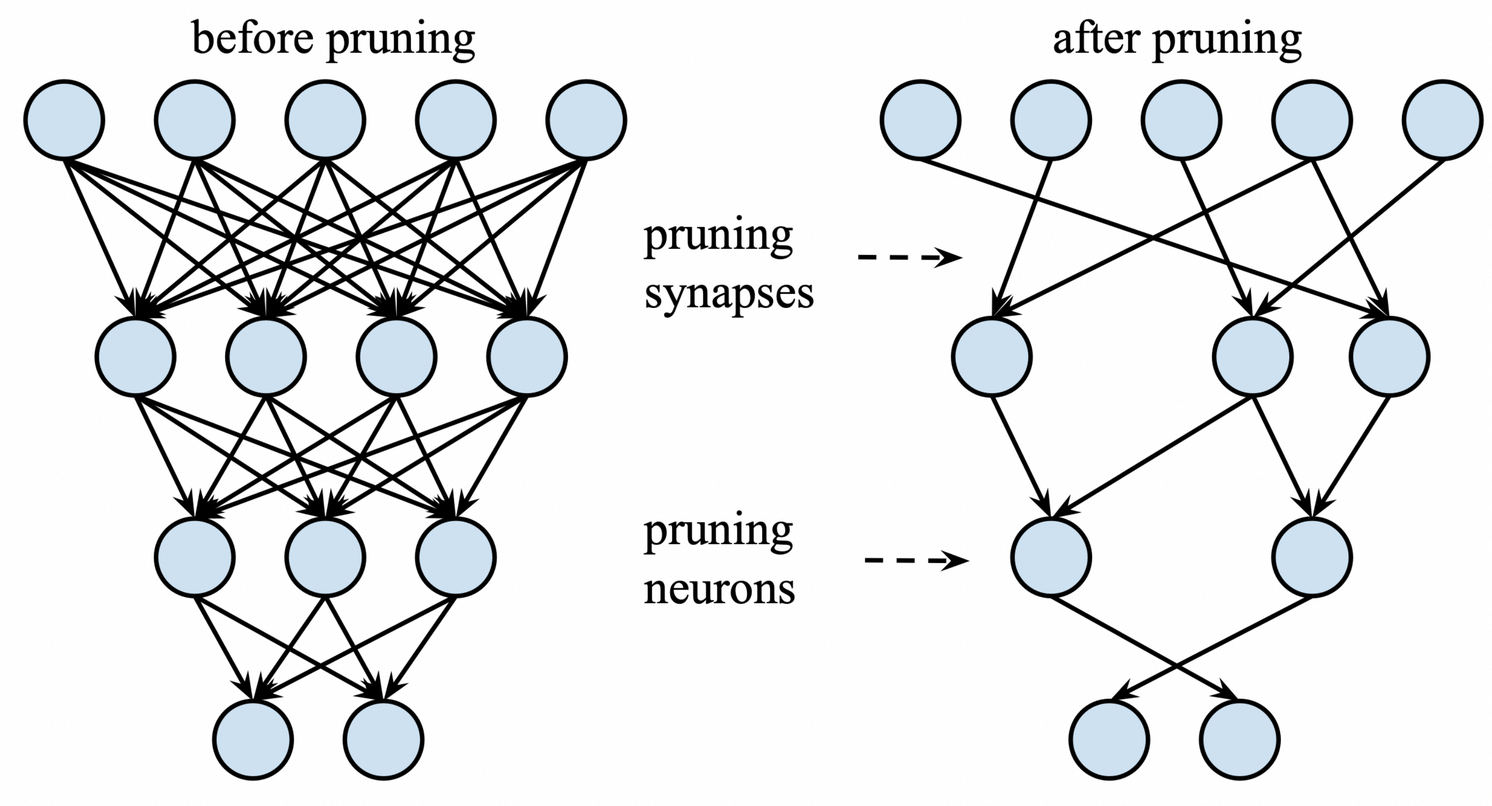


Deep Compression Overview



Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning



Song Han, Stanford PhD thesis. https://purl.stanford.edu/qf934gh3708

Pruning: Sparse Representation

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 40 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Original Matrix

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 40 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Original Matrix

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 5 \quad 2 \quad 3 \quad 4 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation ($2a + n + 1$ numbers)

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 40 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Original Matrix

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 5 \quad 2 \quad 3 \quad 4 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation ($2a + n + 1$ numbers)

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

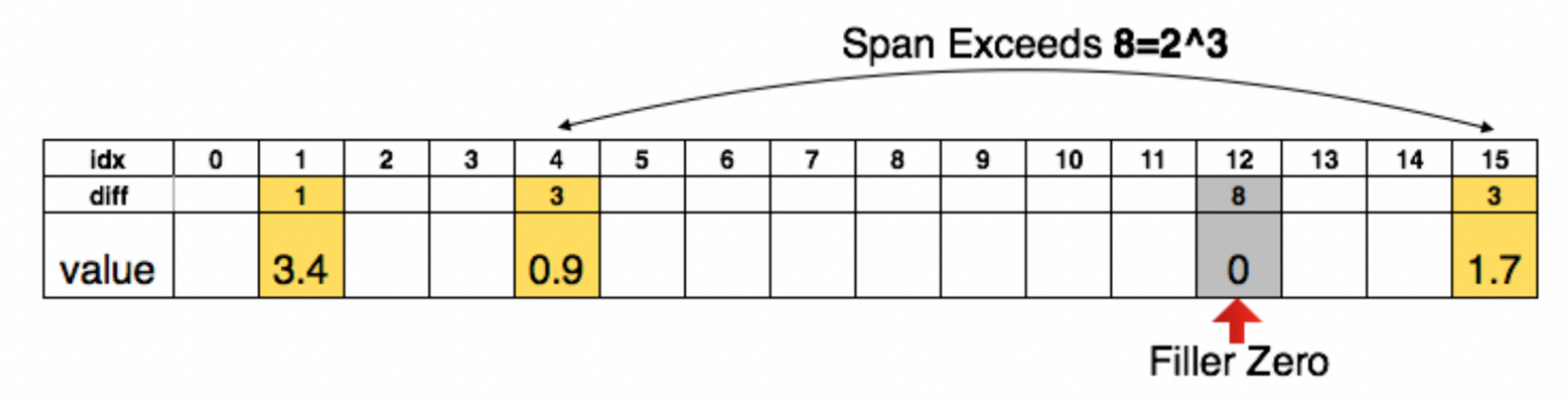
$$COL = [0 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 1 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation with Difference

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation



$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 1 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation with Difference

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 40 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Original Matrix

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 5 \quad 2 \quad 3 \quad 4 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation ($2a + n + 1$ numbers)

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 1 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation with Difference

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Pruning: Sparse Representation

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 40 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Original Matrix

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 5 \quad 2 \quad 3 \quad 4 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation ($2a + n + 1$ numbers)

$$V = [10 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70 \quad 80]$$

$$COL = [0 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 1 \quad 5]$$

$$ROW = [0 \quad 2 \quad 4 \quad 7 \quad 8]$$

CSR Representation with Difference

$$V = [10 \quad 20 \quad 30 \quad 0 \quad 40 \quad 50 \quad 60 \quad 70 \quad 0 \quad 80]$$

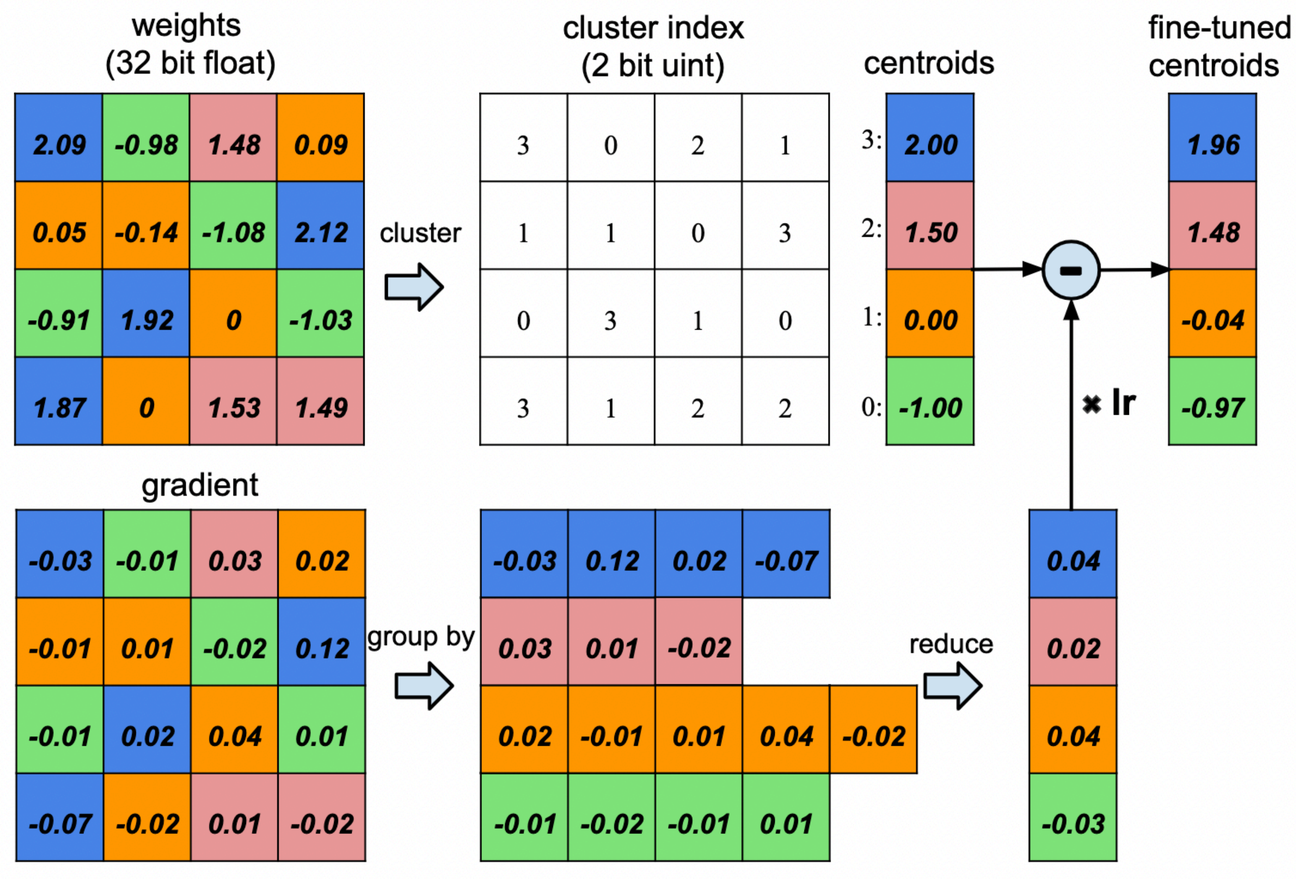
$$COL = [0 \quad 1 \quad 1 \quad 3 \quad 1 \quad 2 \quad 1 \quad 1 \quad 3 \quad 2]$$

$$ROW = [0 \quad 2 \quad 5 \quad 8 \quad 10]$$

CSR Representation with Difference (encoded in 2 bits)

https://en.wikipedia.org/wiki/Sparse_matrix and Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

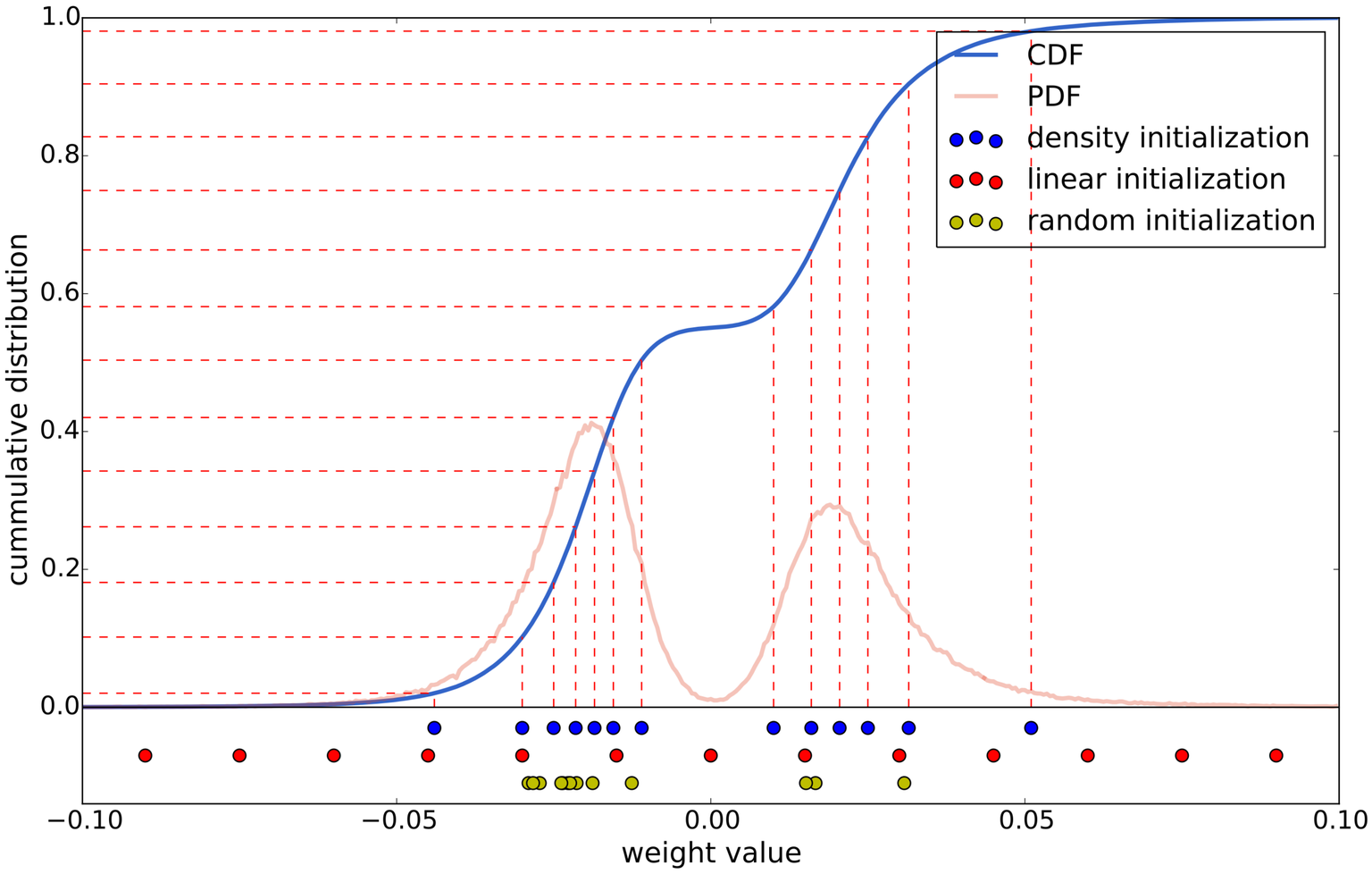
Quantization



$$\text{Compression Ratio } \frac{nb}{n \log_2 k + kb}$$

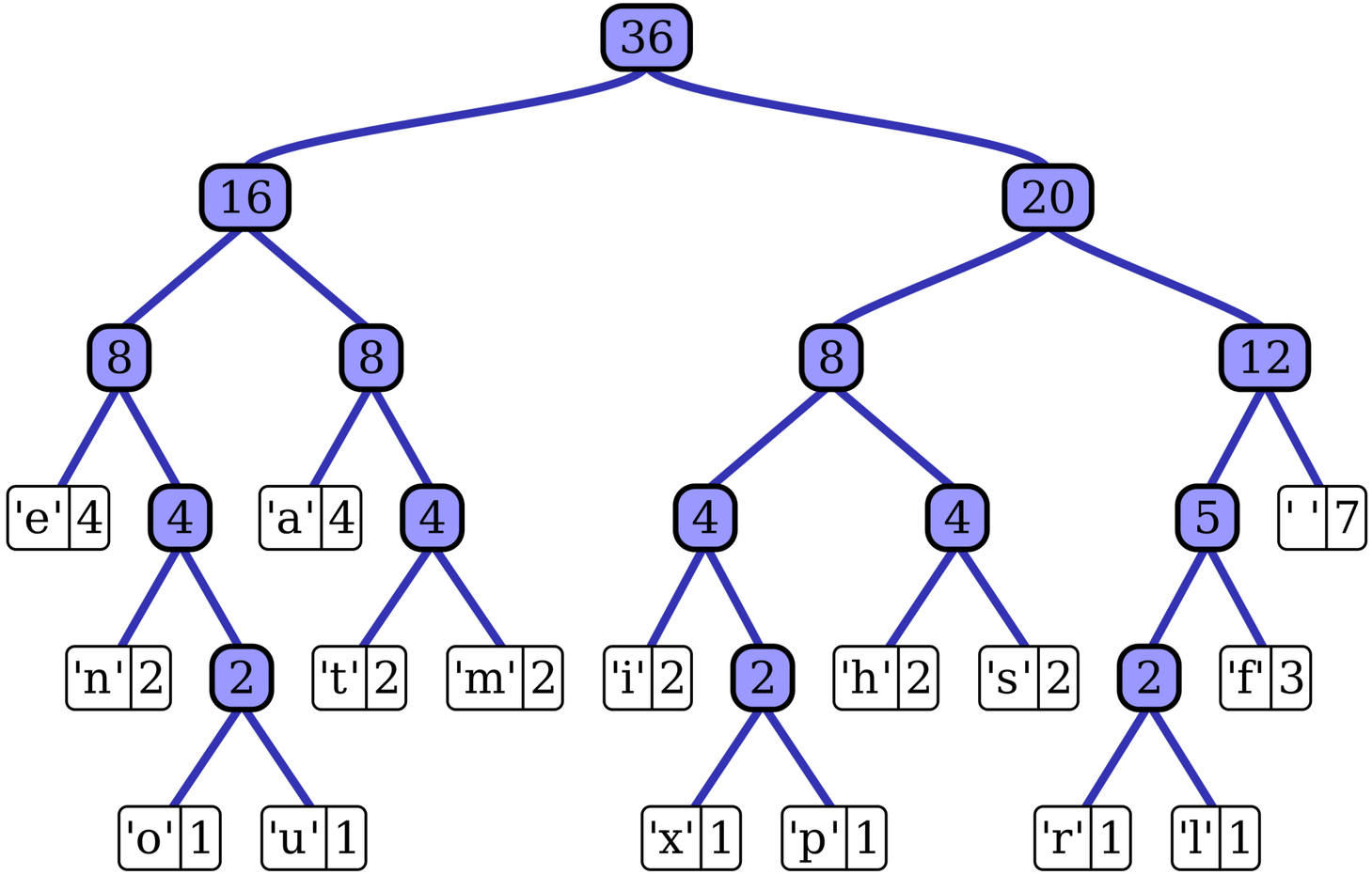
Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Quantization: Initialization of the Shared Weights



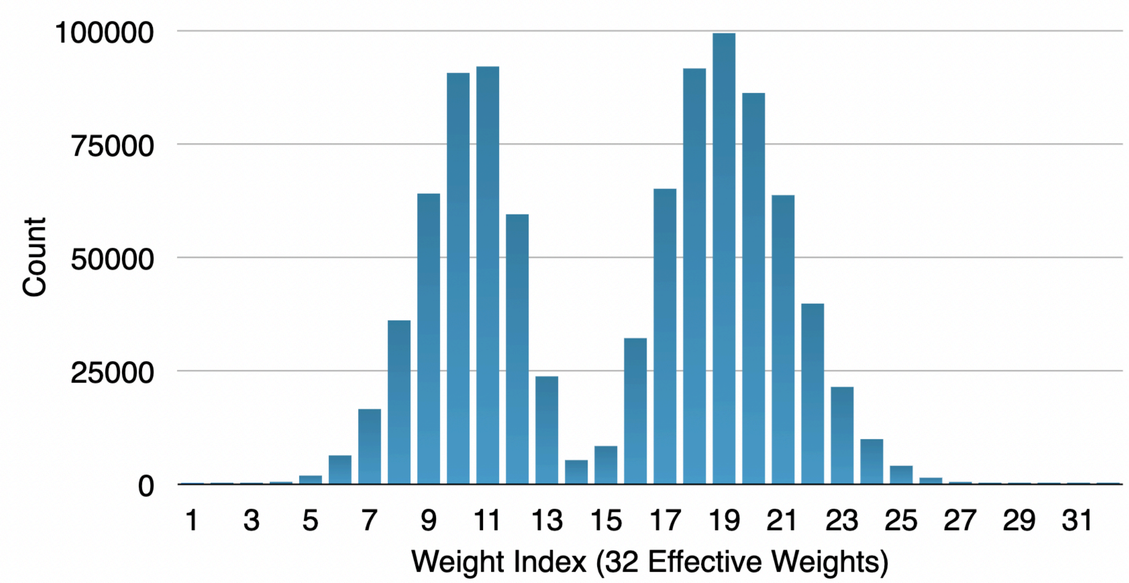
Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

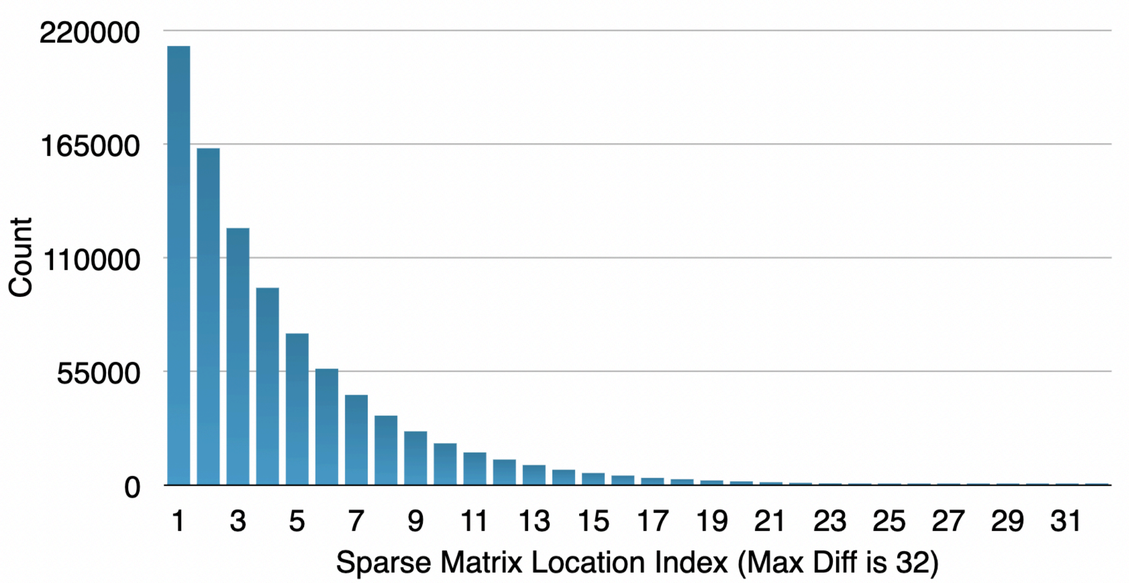
Huffman Coding



https://en.wikipedia.org/wiki/Huffman_coding

Huffman Coding





Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Results

Table 1: The compression pipeline can save 35× to 49× parameter storage with no loss of accuracy.

Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	**40**×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39**×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35**×
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49**×

Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Results

Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
Baseline Caffemodel (BVLC)	42.78%	19.73%	240MB	1×
Fastfood-32-AD (Yang et al., 2014)	41.93%	-	131MB	2×
Fastfood-16-AD (Yang et al., 2014)	42.90%	-	64MB	3.7×
Collins & Kohli (Collins & Kohli, 2014)	44.40%	-	61MB	4×
SVD (Denton et al., 2014)	44.02%	20.56%	47.6MB	5×
Pruning (Han et al., 2015)	42.77%	19.67%	27MB	9×
Pruning+Quantization	42.78%	19.70%	8.9MB	27×
Pruning+Quantization+Huffman	**42.78%**	**19.70%**	**6.9MB**	**35×**

Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Results

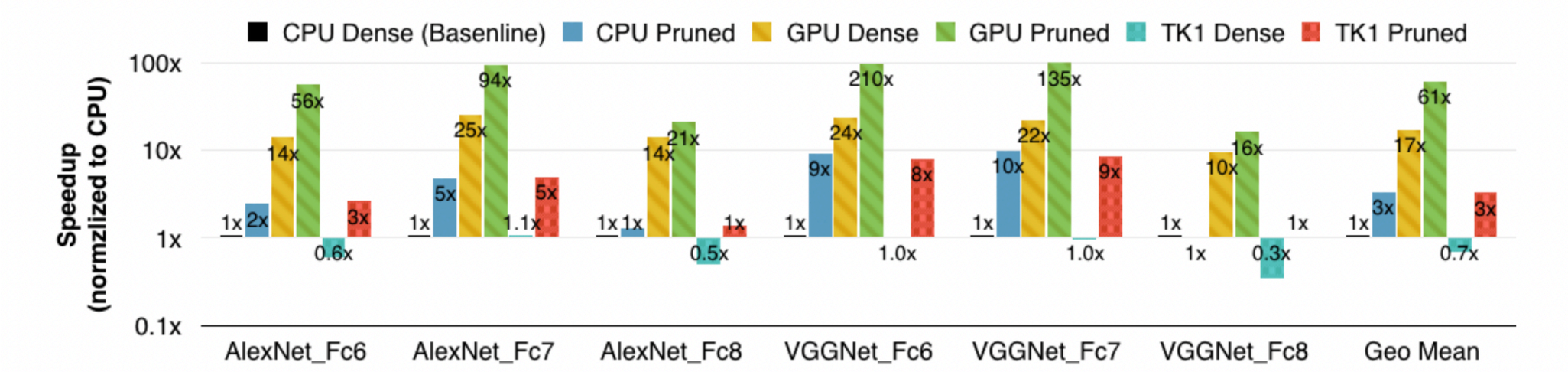


Figure 9: Compared with the original network, pruned network layer achieved 3× speedup on CPU, 3.5× on GPU and 4.2× on mobile GPU on average. Batch size = 1 targeting real time processing. Performance number normalized to CPU.

Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Results

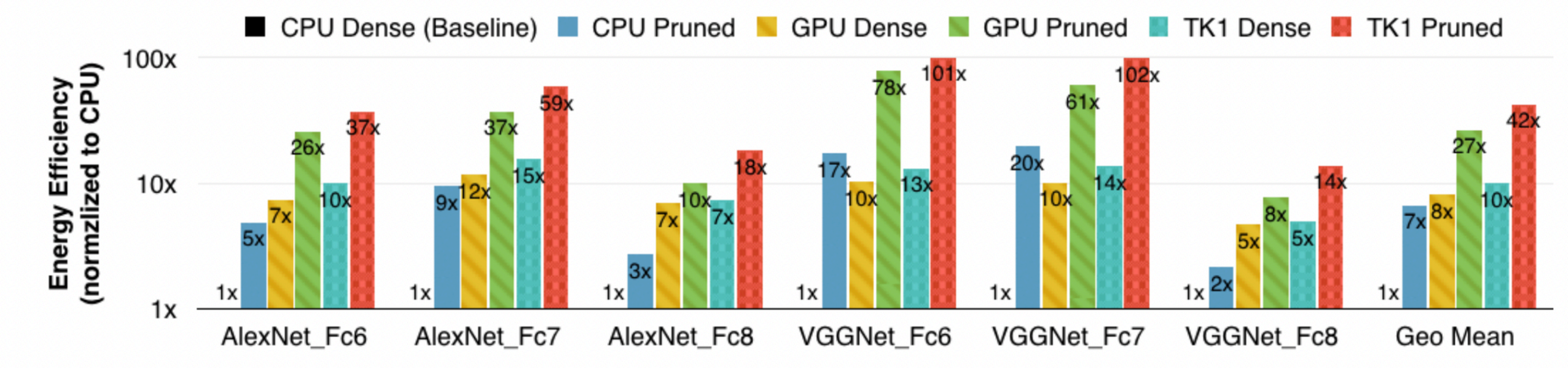


Figure 10: Compared with the original network, pruned network layer takes 7× less energy on CPU, 3.3× less on GPU and 4.2× less on mobile GPU on average. Batch size = 1 targeting real time processing. Energy number normalized to CPU.

Deep Compression, Han et al., 2015, https://arxiv.org/pdf/1510.00149.pdf

Thank you!

Questions

- In fact, the unstructured sparsity (e.g., pruning synapses) in this paper is a little bit hard to convert to real speedup. Is there some way to regularize the sparsity?

- In this paper, the authors set thresholds for each layer manually, which requires much human effort. Besides, the layer sparsity may also have some dependency. For example, you may prune the first layer 90% or the second layer 90% without loss of performance. However, you could not prune both of them 90%. Is there some way that could automatically prune the network as a whole to a targeted compression ratio while achieving the best accuracy?

- Is there some way that you could codesign your pruning and quantization with the system-level optimization (e.g., TVM)? For example, sometimes, you may find pruning more does not indicate lower latency due to some system optimization. Generally, multipliers of 8 are more friendly for system optimization.

Carnegie Mellon University

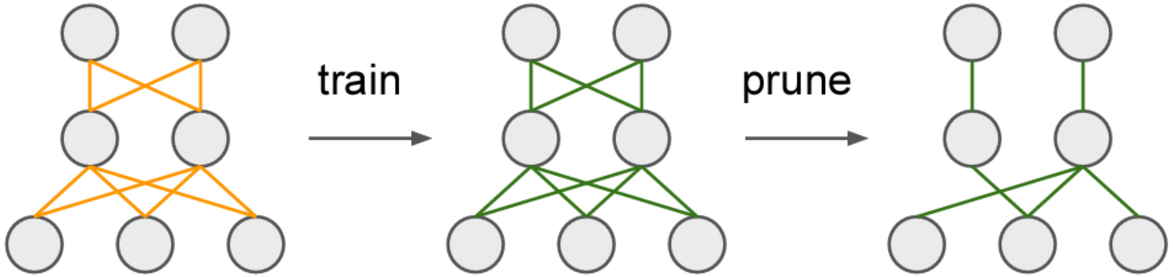
THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS

Authors: Jonathan Frankle (MIT), Michael Carbin (MIT)

Presenter: Rulin Shao, Machine Learning Department

Motivating Question

We learned from the previous paper that model compression techniques like pruning can reduce the number of parameters by more than 90% without harming accuracy.



Then we may wonder:

Since a smaller model can perform well on the task, can we train this sparse model **from scratch** to achieve **comparable performance** but with less computations?

LTH: Promising!

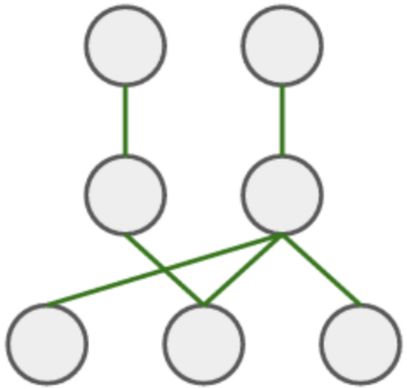
Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 1: Random Sparse Network?

What if we randomly sample a sparse subnetwork and train it from scratch?

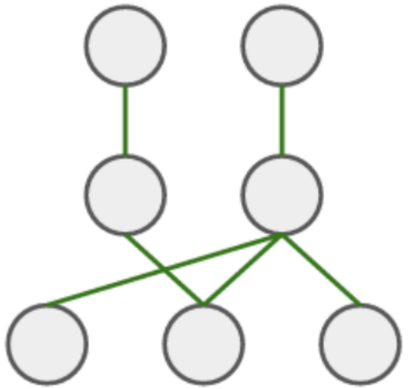
Random Subnetwork



Random Initialization

Train from Scratch

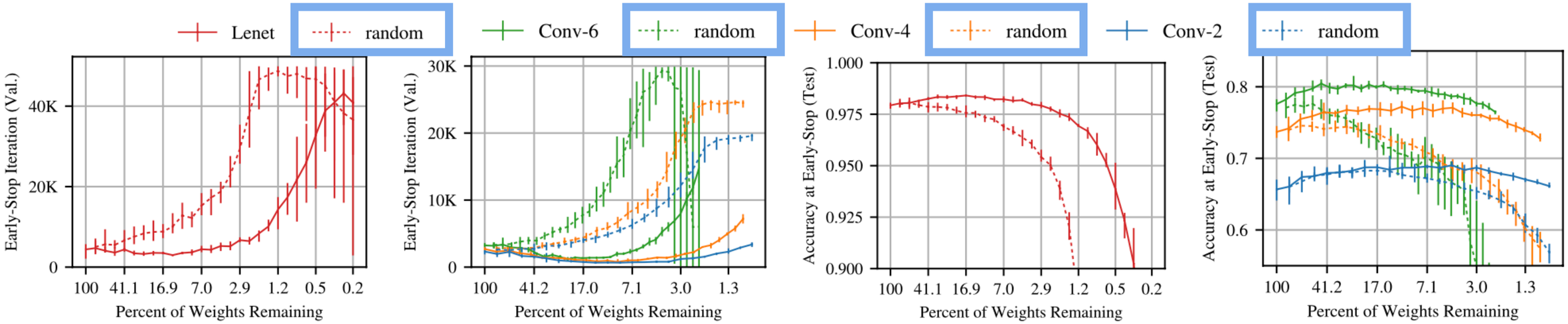
“Random”



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 1: Random Sparse Network?



The sparser the network, the slower the learning and the lower the eventual test accuracy.

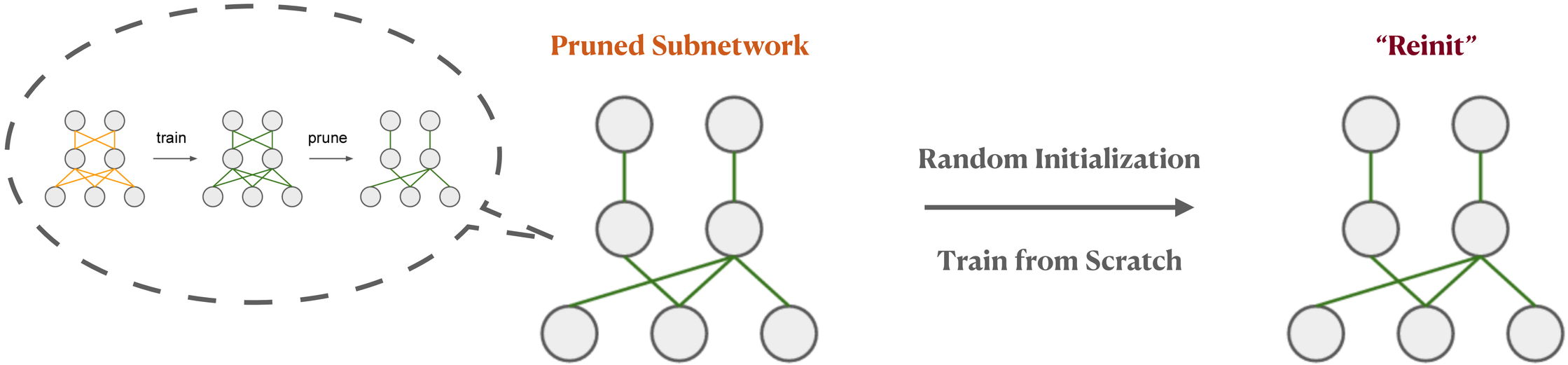
Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 2: Topology Connections from Pruning?

Well, if an arbitrary network doesn't work, what can we do next?

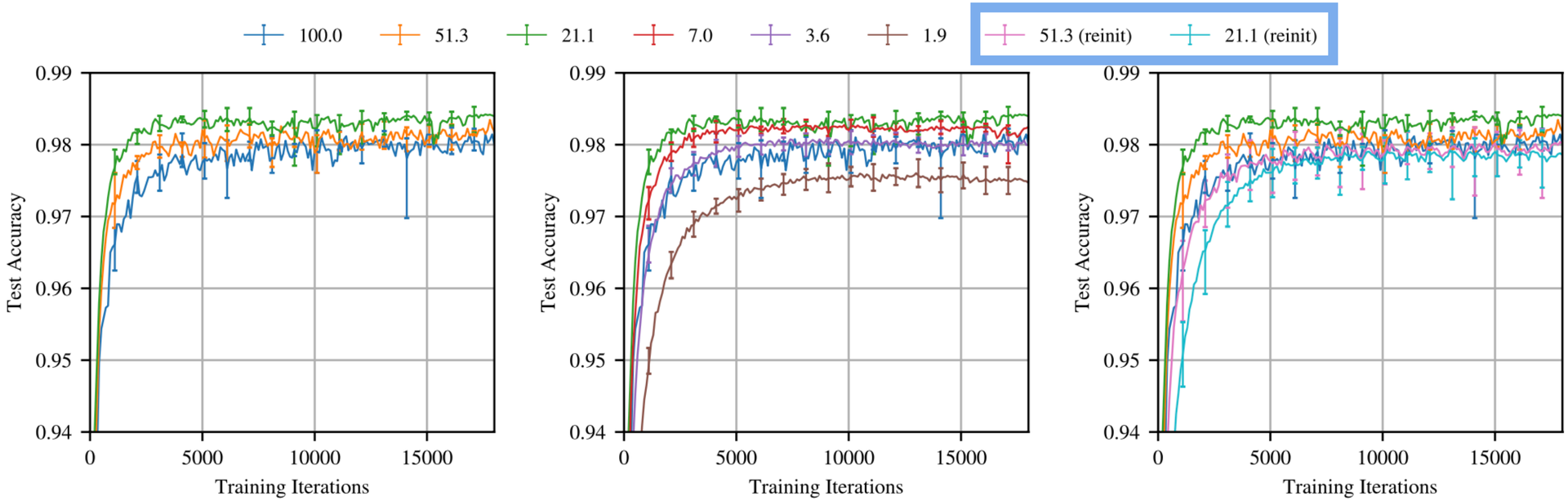
What if we take advantage of the connection topology found by pruning?



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 2: Topology Connections from Pruning?



Not as good as the dense neural network.

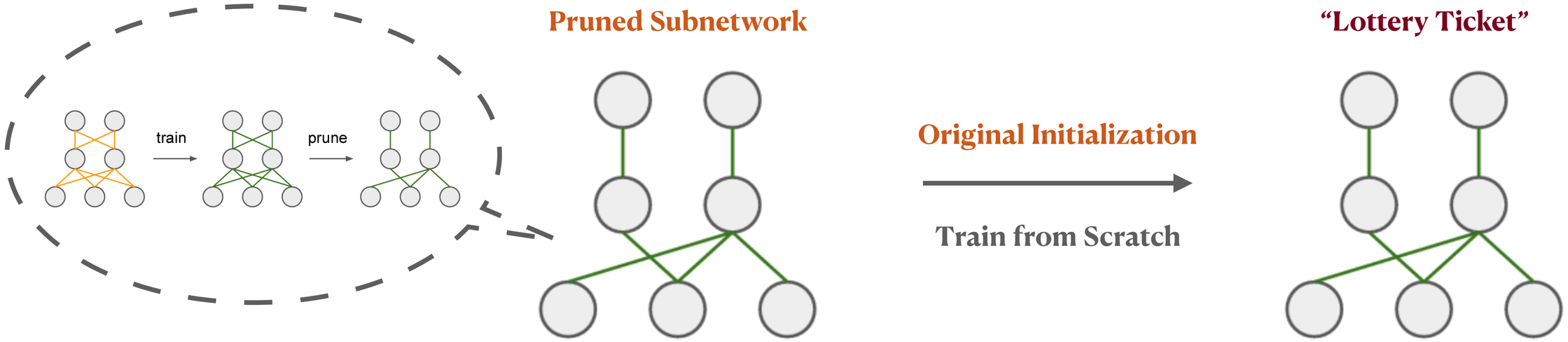
Structure alone cannot meet our goal.

Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 3: Topology Connections + Initialization?

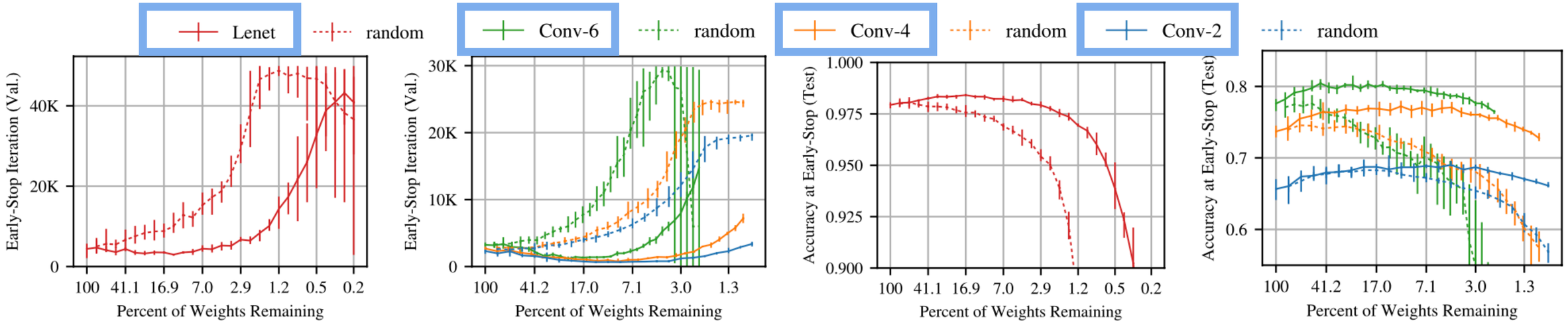
What if we further preserve the original initialization?



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

Proposal 3: Topology Connections + Initialization?



Lottery tickets have **faster learning** and **higher test accuracy** than the dense network!

Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

We reached the goal!

Carnegie Mellon University

Lottery Ticket Hypothesis (LTH)

Formally speaking:

Found by unstructured pruning

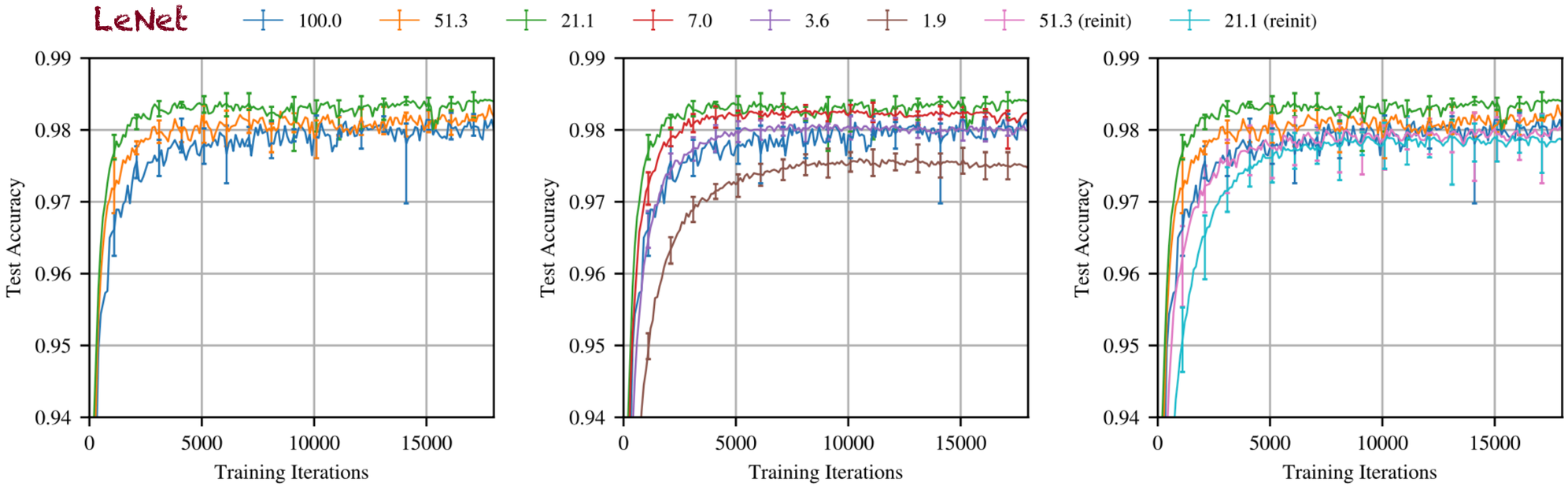
A randomly-initialized, dense neural network contains a subnetwork that is initialized such that — when trained in isolation — it can match the test accuracy of the original network after training for at most the same number of iterations.

Using the original initialization

Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

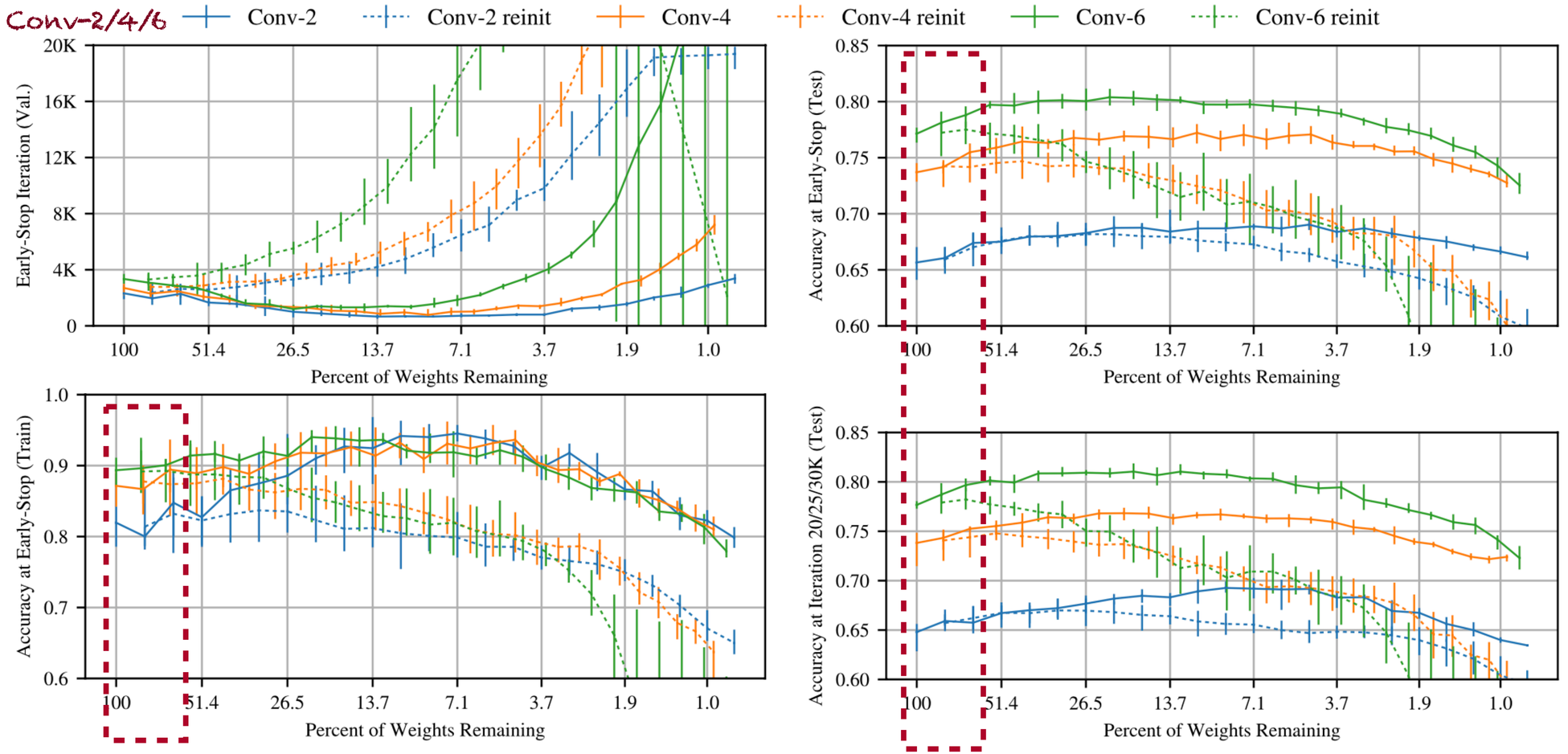
LTH: Experiments



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

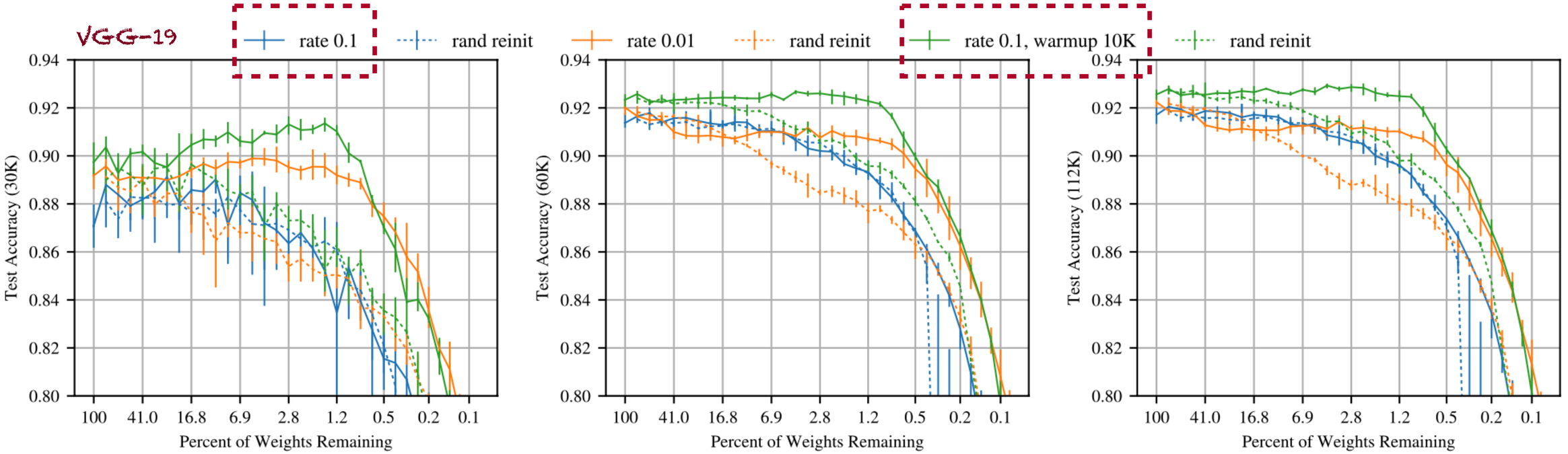
LTH: Experiments



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

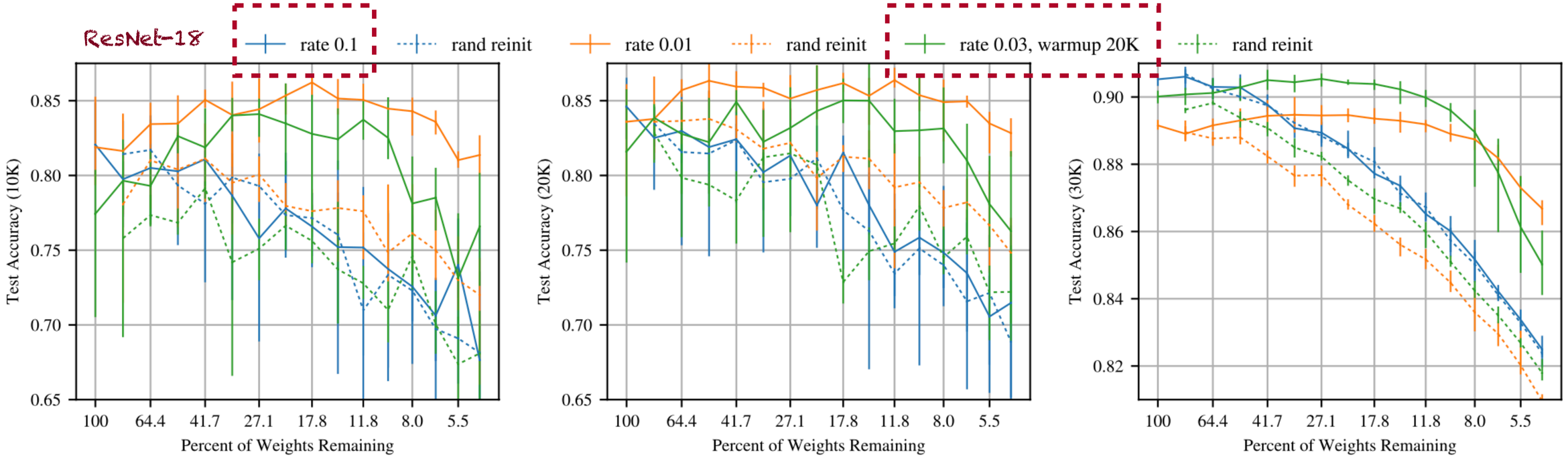
LTH: Experiments



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

LTH: Experiments



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

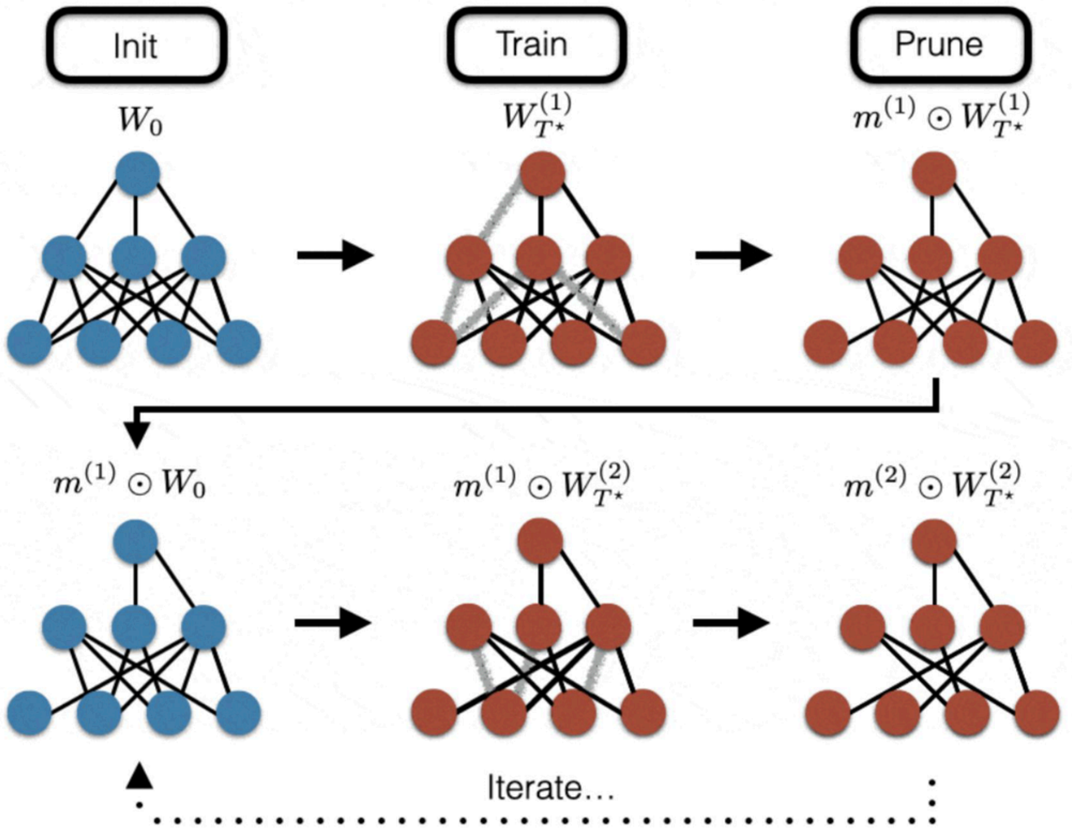
LTH: one-shot v.s. iterative pruning

How does LTH find the winning tickets?

—— Pruning: one-shot v.s. iterative

Iterative Pruning:

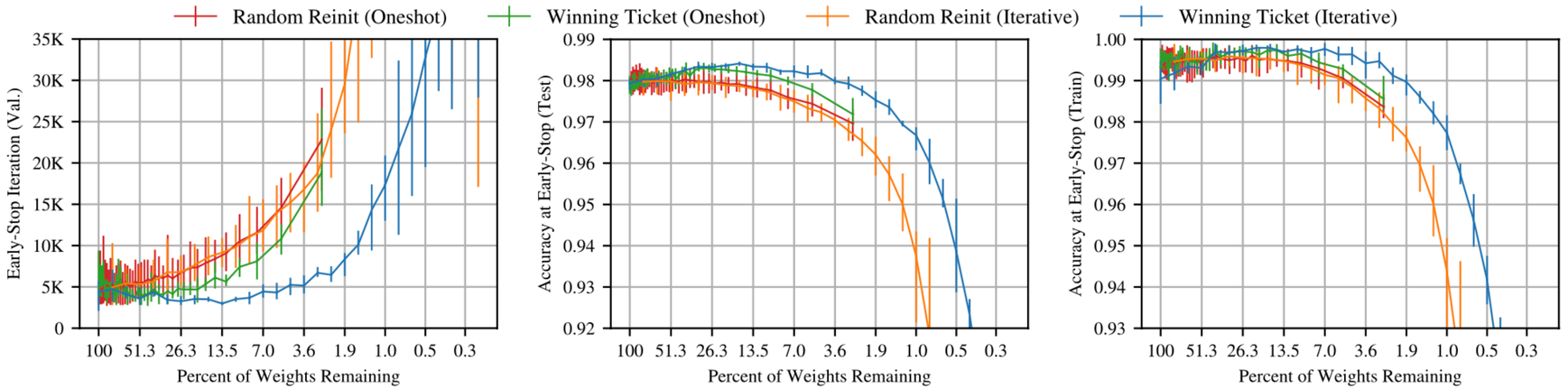
1. Initialize the model
2. Train the model
3. Prune $p^{\frac{1}{n}}\%$ of weights with smallest magnitudes
4. Reset values of remaining weights to their previous initial values
5. Repeat 2 - 4 until the desired sparsity is met



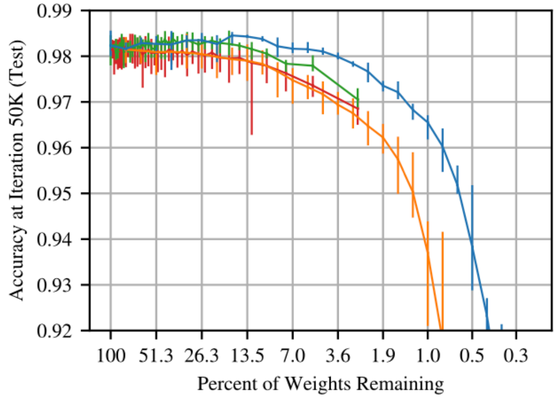
Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

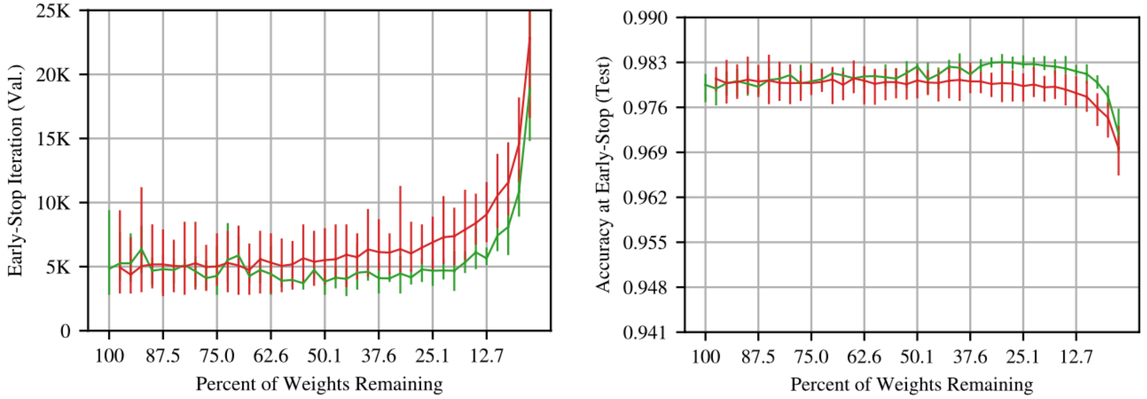
LTH: one-shot v.s. iterative pruning



(a) Early-stopping iteration and accuracy for all pruning methods.



(b) Accuracy at end of training.



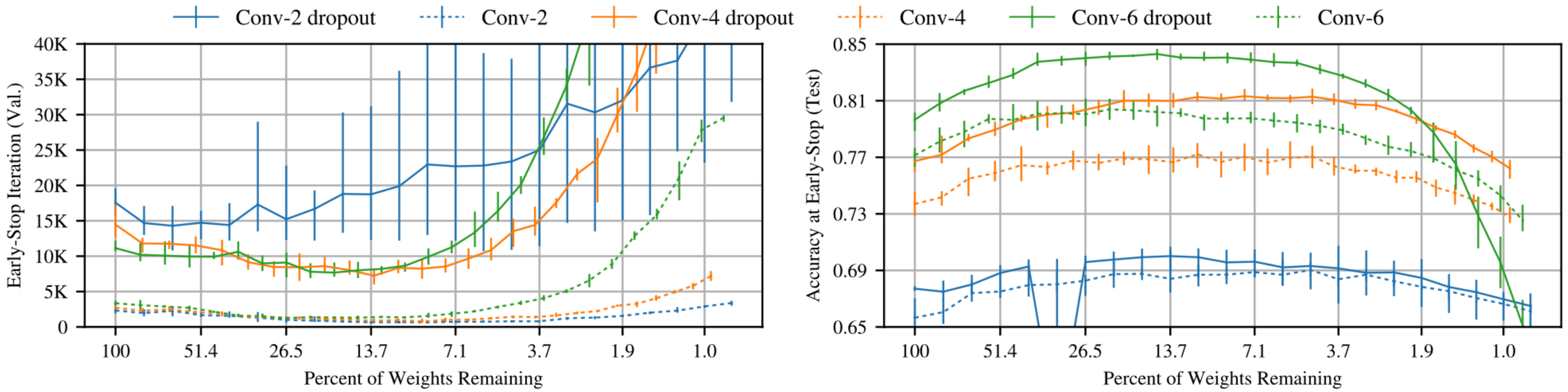
(c) Early-stopping iteration and accuracy for one-shot pruning.

Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

LTH: effect of dropout

Dropout is simultaneously training the ensemble of all subnetworks.
Dropout induce sparsity in the activations.
It is possible that dropout-induced sparsity primes a network to be pruned.



Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

LTH: Issues and Related Works

Have we really reached the goal? Not really...

- Finding the winning tickets (i.e., the pruning process) is too expensive. [2,3,4,5]
- Unstable: not useful for large learning rate and structured pruning (e.g., l_1-norm pruning). [6]
- The original initialization contains the pre-training information. LT is not strictly training from scratch ...

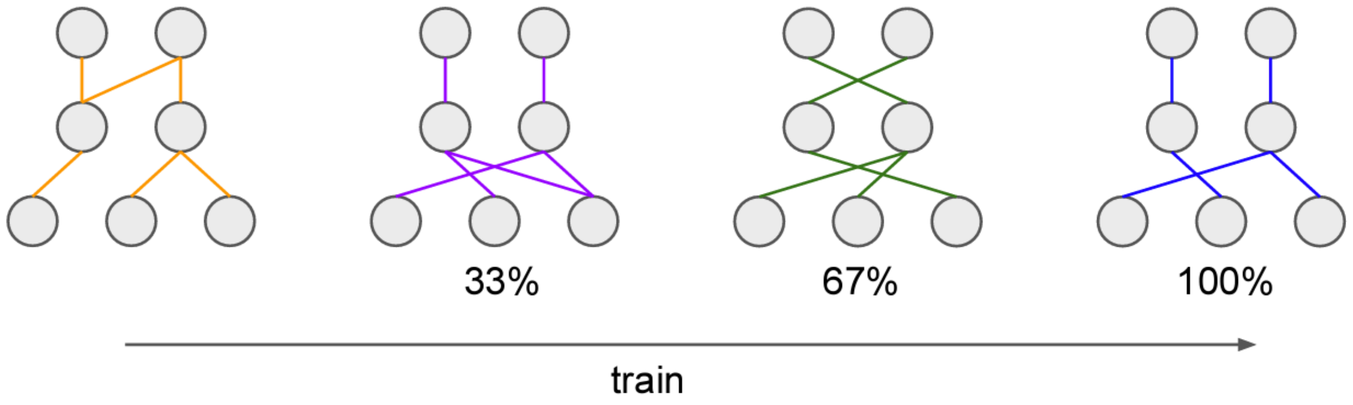
Limitations:

- Only involved small detests (MNIST and CIFAR-10) [2,3,4,5]
- Only considered unstructured pruning. [6]
- The resulting architectures are not optimized for modern libraries or hardware. [5]

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).
[2] Evci, Utku, et al. "Rigging the lottery: Making all tickets winners." *(ICML* 2020).
[3] Jayakumar, Siddhant, et al. "Top-kast: Top-k always sparse training." (NeurIPS 2020).
[4] Dettmers, Tim, and Luke Zettlemoyer. "Sparse networks from scratch: Faster training without losing performance." (2019).
[5] Zhou, Aojun, et al. "Learning n: m fine-grained structured sparse neural networks from scratch." (ICLR 2021).
[6] Liu, Zhuang, et al. "Rethinking the value of network." (ICLR 2019)

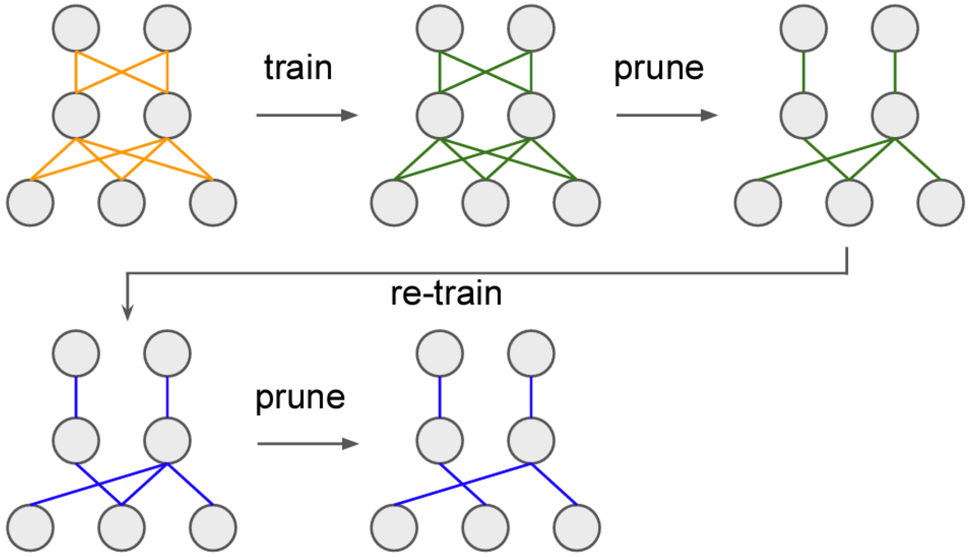
Follow-up Works: Sparse Training

Sparse-to-sparse



- Sparsity is maintained throughout the whole training process
- Connection topology can change during training
- Goal: Achieving the same performance as dense training with minimal computations

Dense-to-sparse

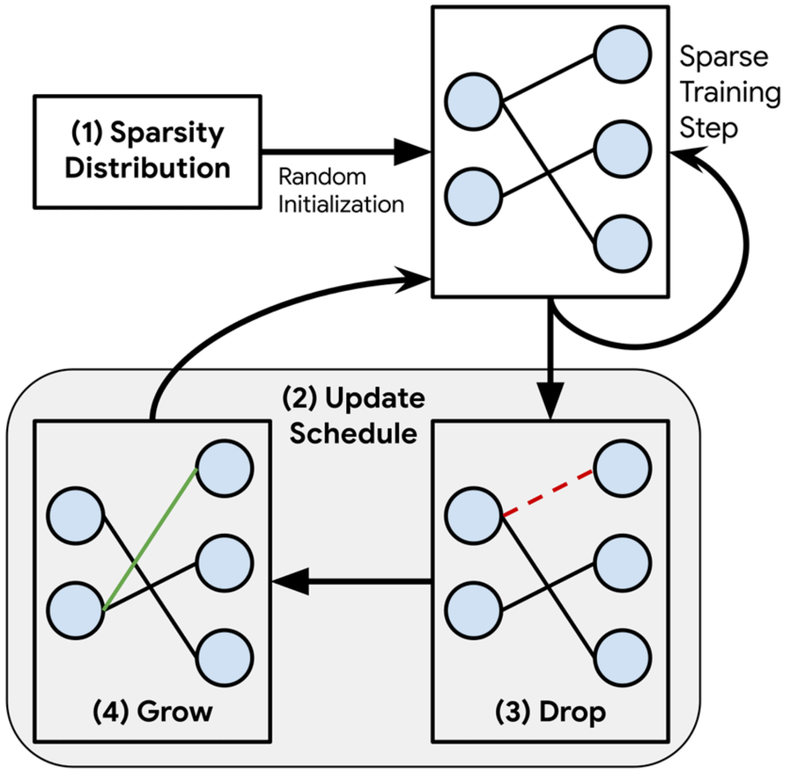


- (Iteratively) train the dense model, and then sparsify the “trained” dense model

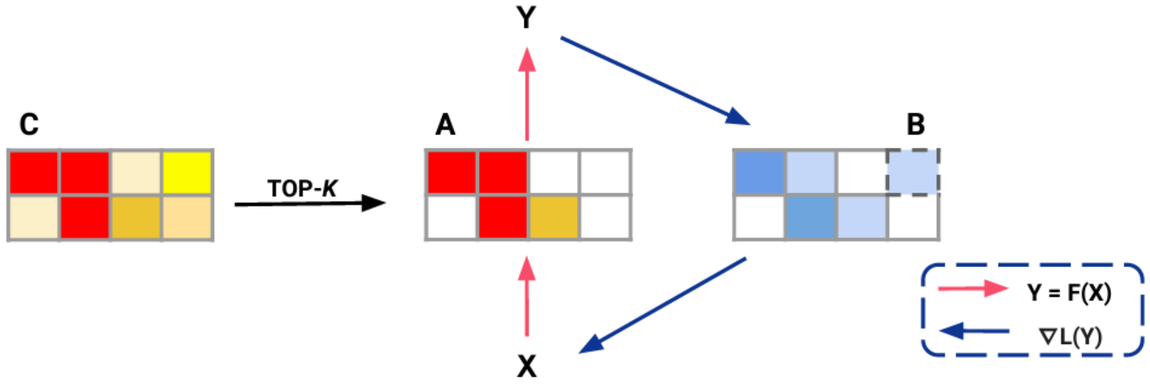
Carnegie Mellon University

Follow-up Works: Sparse Training

Rigging the lottery [1]



Top-KAST [2]



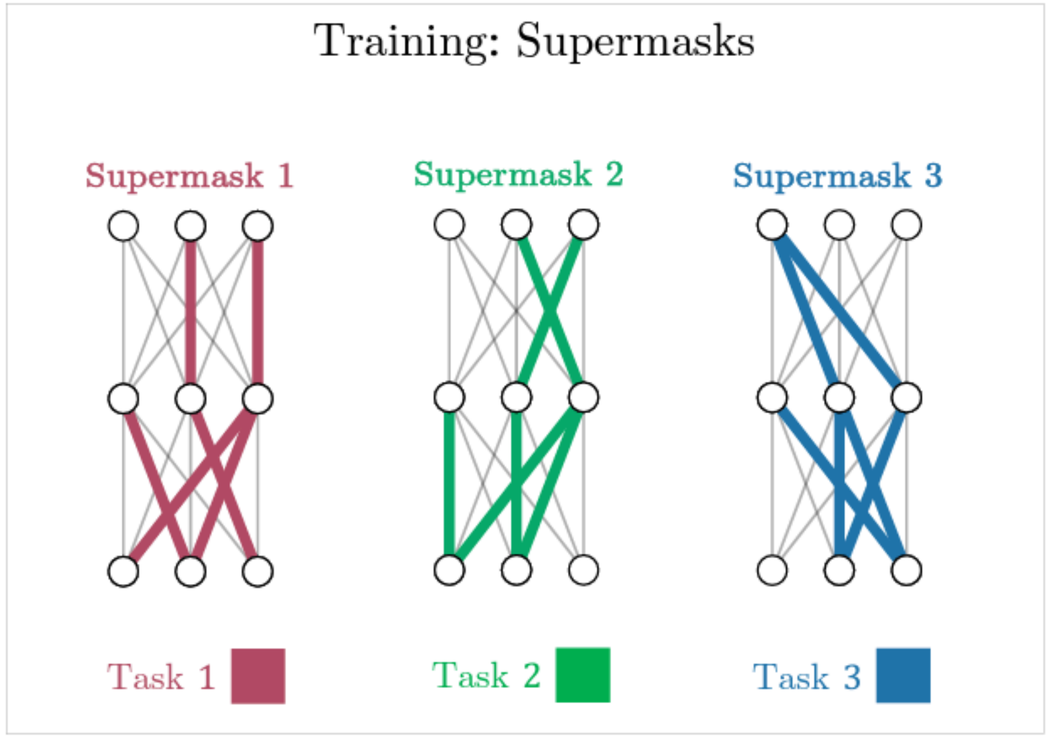
Sparse Momentum [3]: redistribute removed weights using sparse momentum.
N:M [4]: only N weights are non-zero for every continuous M weights.
Rethinking [5]: consider structured pruning.
...

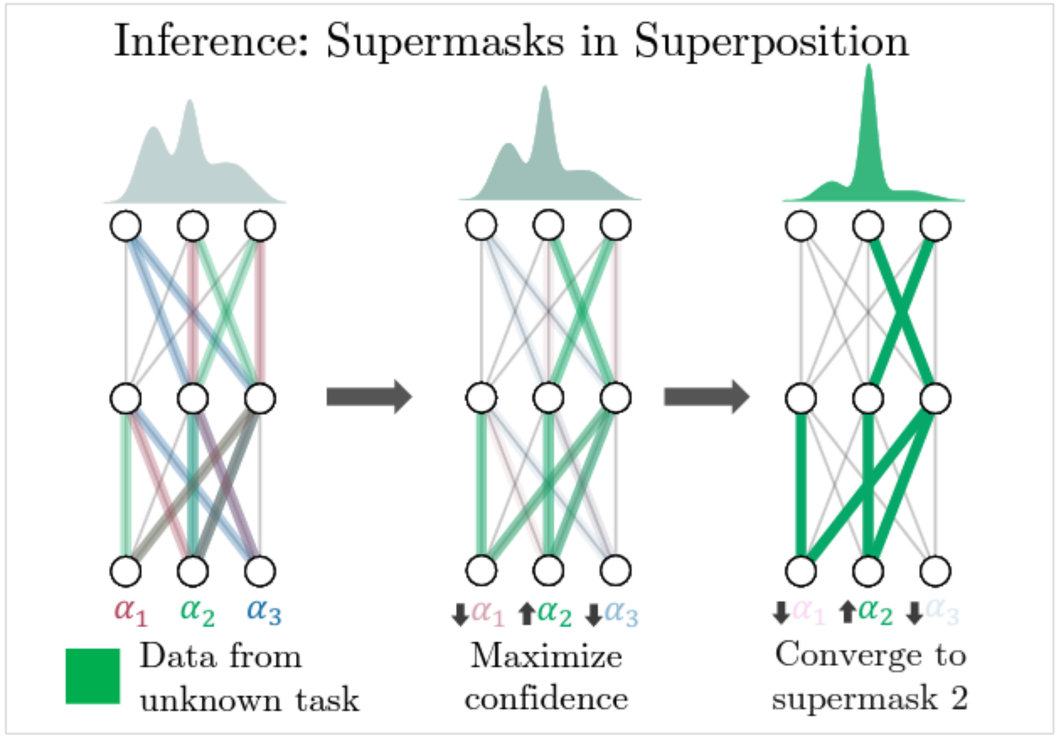
[1] Evci, Utku, et al. "Rigging the lottery: Making all tickets winners." (*ICML* 2020).
[2] Jayakumar, Siddhant, et al. "Top-kast: Top-k always sparse training." (NeurIPS 2020).
[3] Dettmers, Tim, and Luke Zettlemoyer. "Sparse networks from scratch: Faster training without losing performance." (2019).
[4] Zhou, Aojun, et al. "Learning n: m fine-grained structured sparse neural networks from scratch." (ICLR 2021).
[5] Liu, Zhuang, et al. "Rethinking the value of network." (ICLR 2019)

Carnegie Mellon University

Structural Information in Subnetworks

SupSup [1]: The base network is randomly initialized, learn supermasks for different tasks.





Carnegie Mellon University

[1] Wortsman, Mitchell, et al. "Supermasks in superposition." *(NeurIPS 2020)*

Lottery Ticket Hypothesis Conjecture

(Untested) **Conjecture**: SGD seeks out and trains a subset of well-initialized weights. Dense, randomly-initialized networks are easier to train than the sparse networks that result from pruning because there are more possible subnetworks from which training might recover a winning ticket.

Carnegie Mellon University

[1] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (ICLR 2019).

One (Possible) Perspective to Support the Conjecture: Equivalences Between Sparse Models and Neural Networks [1]

The training process of neural networks (e.g. with weight decay or early stopping) are "hunting" for a sparse representation in terms of its learned features!

- Equivalences can be derived between sparse models and neural networks.
- Weight decay and early stopping can both be sparse-inducing in neural networks.

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks

The simplest case: The lasso is equivalent to a two-layer neural network* fit with weight decay:

Lasso

$$\underset{\beta}{\text{minimize}} \quad \sum_{i=1}^{n}(y_i - x_i^\mathsf{T}\beta)^2 + 2\lambda\sum_{j=1}^{d}|\beta_j|$$

$\Longleftrightarrow$

A two-layer neural network fit with weight decay

$$\underset{u,v}{\text{minimize}} \quad \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} x_{ij}u_jv_j\right)^2 + \lambda\sum_{j=1}^{d}(u_j^2 + v_j^2)$$

Optimize in the 2d dimension (instead of d)

$$\forall c,\ \min_{ab=c}(a^2 + b^2) = 2|c| \text{ is achieved at } a = b = \sqrt{|c|}$$

* A two-layer neural network with linear activation functions, no bias terms, and a very simple connectivity structure

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks

A k-layer neural network* that has otherwise the same structure is in turn equivalent to an l_p-penalized regression problem, where $p = 2/k < 1$.

An l_p-penalized regression problem

$$\underset{\beta}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d} x_{ij}\beta_j\right) + k\lambda \sum_{j=1}^{d} |\beta_j|^{2/k}$$

$\Longleftrightarrow$

K-layer neural network

$$\underset{w^{(\ell)},\, \ell=1,\ldots,k}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d} x_{ij} \prod_{\ell=1}^{k} w_j^{(\ell)}\right) + \lambda \sum_{j=1}^{d} \sum_{\ell=1}^{k} (w_j^{(\ell)})^2$$

* A k-layer neural network with linear activation functions, no bias terms, and a very simple connectivity structure since the third layer

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks

Similarly observations can be extended to k-layer group-sparse models* and (2 or 3-layer) fully connected NN with linear/ReLU activations... (Equivalent to a **group lasso problem)**

A 2-layer fully connected NN with ReLU activation

$$\underset{b^{(1)},w^{(1)},w^{(2)}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d_2} \phi(x_i^T w_j^{(1)} + b_j^{(1)}) w_j^{(2)}\right) + \lambda\left(\sum_{j=1}^{d_2} \|w_j^{(1)}\|_2^2 + \|w^{(2)}\|_2^2\right)$$

A group lasso problem

$$\underset{\alpha,\,\beta,\,s^{(2)}\in\{-1,1\}^{d_2}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d_2} \phi(x_i^T \beta_j + \gamma_j) s_j^{(2)}\right) + 2\lambda \sum_{j=1}^{d_2} \|\beta_j\|_2.$$

* A k-layer neural network with linear activation functions, no bias terms, and a very simple connectivity structure since the third layer

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks

A group lasso problem

$$\underset{\alpha,\,\beta,\,s^{(2)}\in\{-1,1\}^{d_2}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d_2} \phi(x_i^\top \beta_j + \gamma_j) s_j^{(2)}\right) + 2\lambda \sum_{j=1}^{d_2} \|\beta_j\|_2.$$

A 2-layer fully connected NN with ReLU activation

$$\underset{b^{(1)},w^{(1)},w^{(2)}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{d_2} \phi(x_i^\top w_j^{(1)} + b_j^{(1)}) w_j^{(2)}\right) + \lambda\left(\sum_{j=1}^{d_2} \|w_j^{(1)}\|_2^2 + \|w^{(2)}\|_2^2\right)$$

Due to what we know about the group lasso penalty, many of the parameters β_j will be sparse at the solution

The same will be true for the feature map:

$$x \mapsto \phi(x_i^\top w_j^{(1)} + b_j^{(1)}) w_j^{(2)},\ j = 1, \ldots, d_2.$$

That is, many of these functions will be either the zero map, or a constant map .

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks

$$x \mapsto \phi(x_i^T w_j^{(1)} + b_j^{(1)}) w_j^{(2)}, \; j = 1, \ldots, d_2$$

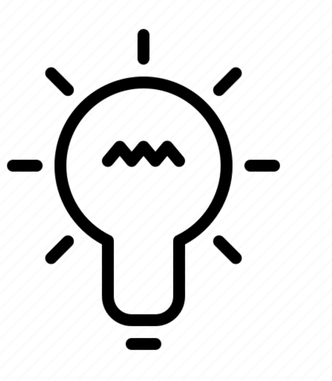
That is, many of these functions will be either the zero map, or a constant map .

Therefore, a standard two-layer neural network, trained with weight decay, is actually “hunting” for a sparse representation in terms of its learned features!

Carnegie Mellon University

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).

Another Perspective: Equivalences Between Sparse Models and Neural Networks



One possible takeaway from this perspective:

If the neural networks are learning sparse representation themselves, it shouldn't be surprising that we can prune a learned network to a sparse structure.

But in lower dimensions, we need to rely on good initializations/ appropriate learning rate/careful construction of the model structure … for current optimizers to find a good local optimal.

E.g. rewinding[2,3]…

- These subnetworks only reach full accuracy when they are stable to SGD noise, which occurs at initialization in some settings but only after some amount of training in others. [2]
- Weight rewinding, learning rate rewinding[3] …

[1] Tibshirani, Ryan J. "Equivalences Between Sparse Models and Neural Networks." (2021).
[3] Lezama, José, et al. "Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning." (CVPR 2018)
[4] Wright, John, and Yi Ma. High-dimensional data analysis with low-dimensional models: Principles, computation, and applications. Cambridge University Press, 2022.

Discussion Questions

1. Why is it hard for SGD to learn sparse representations from scratch? Why initialization is important to SGD (e.g. thinking of rewinding[1,2] techniques...)?
2. How to make the lottery tickets fit better for modern libraries (possibly at the cost of being less sparse)?
3. Can we use graphical models to learn the sparse connection topology?
4. Can we reformulate the sparse training problem as learning low-rank representations for high-dimensional data? E.g. using nuclear norm (OLE loss[3])/rate distortion (MCR loss[4]) to induce more interpretable sparse models instead of heuristically searching for sparse connections using pruning?

[1] Frankle, Jonathan, et al. "Linear mode connectivity and the lottery ticket hypothesis." *(ICML 2020)*
[2] Renda, Alex, Jonathan Frankle, and Michael Carbin. "Comparing rewinding and fine-tuning in neural network pruning." *(ICLR 2020)*
[3] Lezama, José, et al. "Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning." (CVPR 2018)
[4] Wright, John, and Yi Ma. High-dimensional data analysis with low-dimensional models: Principles, computation, and applications. Cambridge University Press, 2022.

Discussion Questions

1. Why is it hard for SGD to learn sparse representations from scratch?
   - These subnetworks only reach full accuracy when they are stable to SGD noise, which occurs at initialization in some settings but only after some amount of training in others. [1]
   - Weight rewinding, learning rate rewinding[2] ...
2. How to make the lottery tickets fit better for modern libraries (possibly at the cost of being less sparse)?
   - M:N, Low-rank decomposition constrain ...
3. Can we use graphical models to learn the sparse connection topology?
4. Can we reformulate the sparse training problem as learning low-rank representations for high-dimensional data? E.g. using nuclear norm (OLE loss[3])/rate distortion (MCR loss[4]) to induce more interpretable sparse models instead of heuristically searching for sparse connections using pruning?

[1] Frankle, Jonathan, et al. "Linear mode connectivity and the lottery ticket hypothesis." *(ICML 2020)*
[2] Renda, Alex, Jonathan Frankle, and Michael Carbin. "Comparing rewinding and fine-tuning in neural network pruning." *(ICLR 2020)*
[3] Lezama, José, et al. "Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning." (CVPR 2018)
[4] Wright, John, and Yi Ma. High-dimensional data analysis with low-dimensional models: Principles, computation, and applications. Cambridge University Press, 2022.