The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

Gaia Vaccarezza

3073561

Network Pruning

- Definition

Network Pruning is to reduce the extend of a neural network by **removing superflous parts**

- Why?

It reduces the **energy costs, computations and storage constraints** during inference time
how? by identifying well-performing smaller networks.

→ **Put one in a small devices**

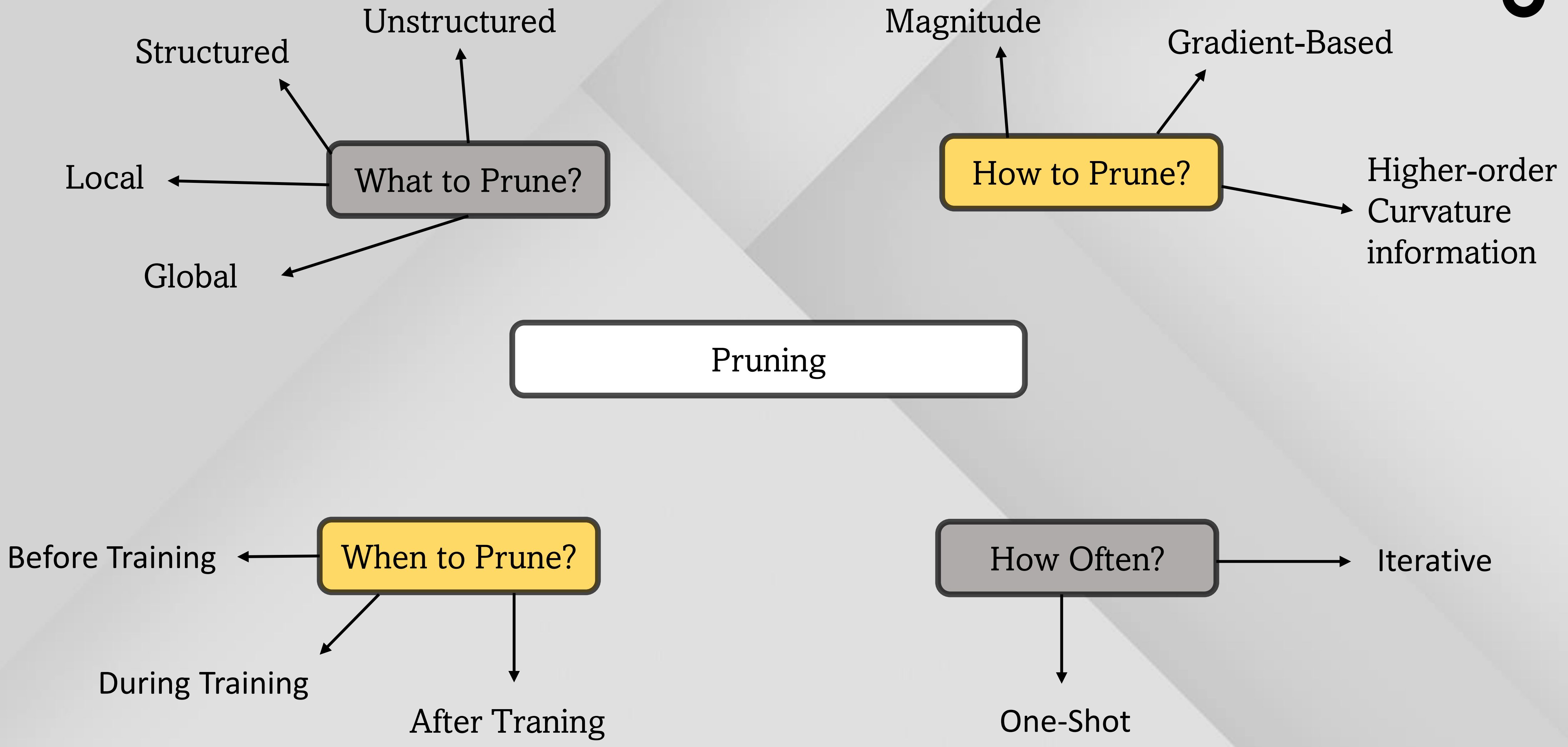
like smartphone, medical devices, sensors, smart cameras
why?

- Limited computation, memory, energy
- Always require internet connection

→ **Reduce training cost**

e.g. OpenAI's GPT-3 has been calculated to have a training cost of $4.6M using the lowest-cost cloud GPU on the market

Network Pruning



Pruning in Literature

Results obtained by papers before:

Pruning is able to reduce significantly amount of parameters without loss in accuracy.

→ **So why don't we train the pruned network from the start?**

Train the pruned in isolation:
1 - Randomly re-initalize pruned network
2 - Train it to convergence

Results:
reaches lower accuracy and learning rate drops with respect to original network

Learning both Weights and Connections for Efficient Neural Networks - Han et al.

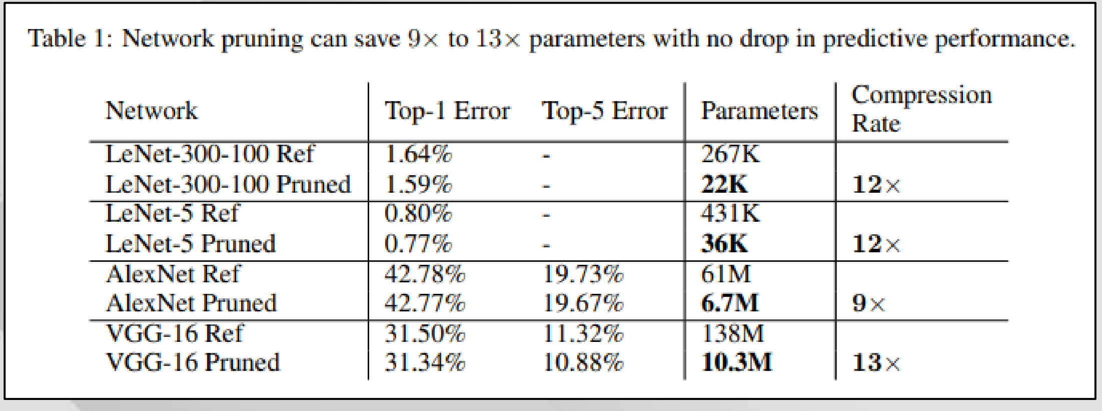


Table 1: Network pruning can save $9\times$ to $13\times$ parameters with no drop in predictive performance.

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**$\times$
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**$\times$
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**$\times$
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13**$\times$

Pruning Filters for Efficient ConvNets - Hao et al.

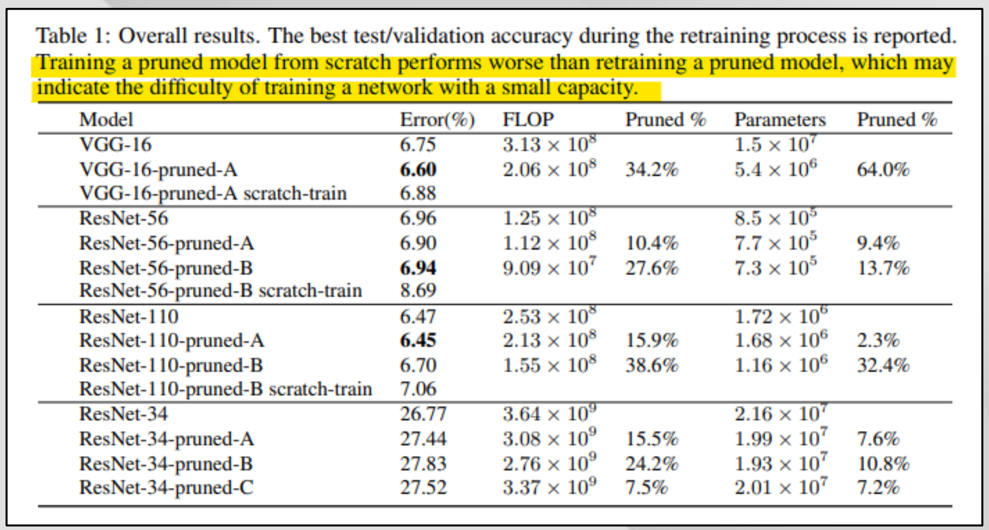


Table 1: Overall results. The best test/validation accuracy during the retraining process is reported. Training a pruned model from scratch performs worse than retraining a pruned model, which may indicate the difficulty of training a network with a small capacity.

Model	Error(%)	FLOP	Pruned %	Parameters	Pruned %
VGG-16	6.75	3.13×10^8		1.5×10^7	
VGG-16-pruned-A	**6.60**	2.06×10^8	34.2%	5.4×10^6	64.0%
VGG-16-pruned-A scratch-train	6.88				
ResNet-56	6.96	1.25×10^8		8.5×10^5	
ResNet-56-pruned-A	6.90	1.12×10^8	10.4%	7.7×10^5	9.4%
ResNet-56-pruned-B	**6.94**	9.09×10^7	27.6%	7.3×10^5	13.7%
ResNet-56-pruned-B scratch-train	8.69				
ResNet-110	6.47	2.53×10^8		1.72×10^6	
ResNet-110-pruned-A	**6.45**	2.13×10^8	15.9%	1.68×10^6	2.3%
ResNet-110-pruned-B	6.70	1.55×10^8	38.6%	1.16×10^6	32.4%
ResNet-110-pruned-B scratch-train	7.06				
ResNet-34	26.77	3.64×10^9		2.16×10^7	
ResNet-34-pruned-A	27.44	3.08×10^9	15.5%	1.99×10^7	7.6%
ResNet-34-pruned-B	27.83	2.76×10^9	24.2%	1.93×10^7	10.8%
ResNet-34-pruned-C	27.52	3.37×10^9	7.5%	2.01×10^7	7.2%

Different weight-initialization

Procedure:

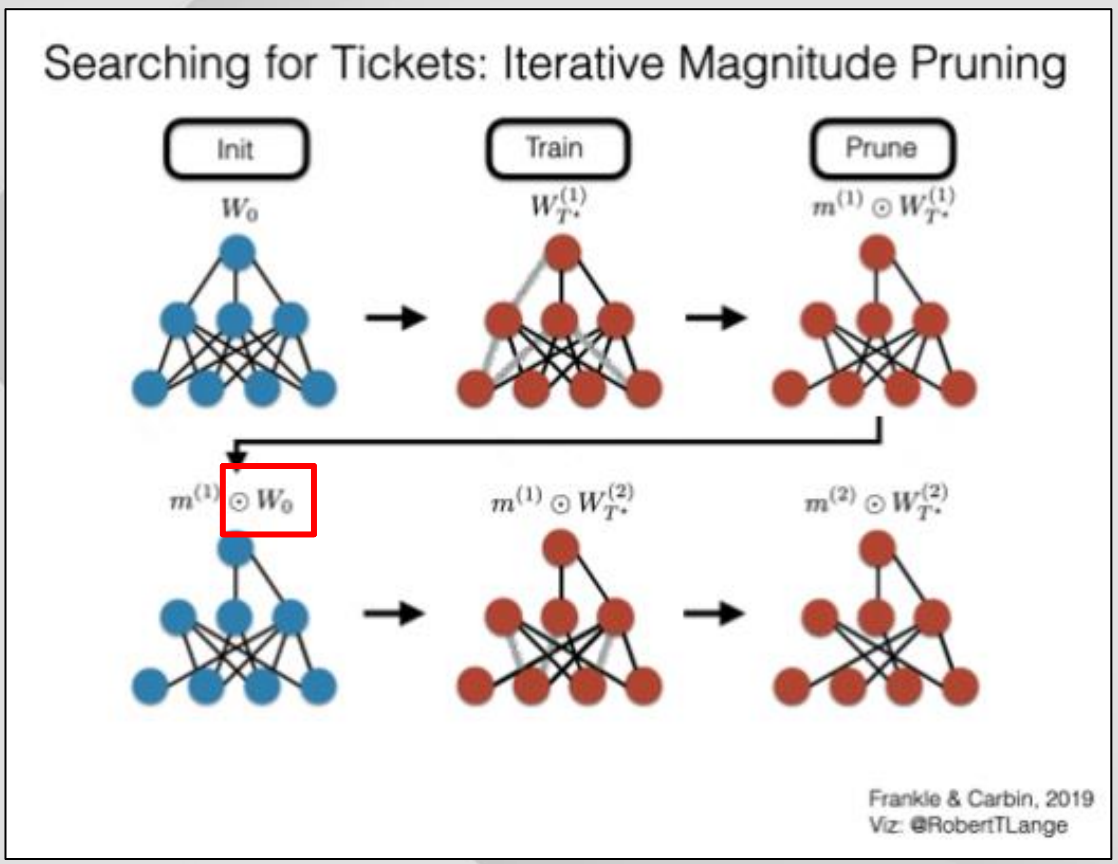
1) Randomly initialize the full network
2) Fully train the model for j iterations
3) Prune superfluous structure
4) **Reset each remaining weights to value before it was trained the first time** instead of re-randomize

$$m^{(1)} \odot W_0.$$

5) May repeat 2, 3 and 4 iteratively (Iterative Pruning)

→ what remains at the end is the **winning ticket**:

$$m^{(n)} \odot W_0$$


Searching for Tickets: Iterative Magnitude Pruning
Init
W_0
Train
$W_{T^*}^{(1)}$
Prune
$m^{(1)} \odot W_{T^*}^{(1)}$
$m^{(1)} \odot W_0$
$m^{(1)} \odot W_{T^*}^{(2)}$
$m^{(2)} \odot W_{T^*}^{(2)}$
Frankle & Carbin, 2019
Viz: @RobertTLange

Different weight-initialization

Procedure:

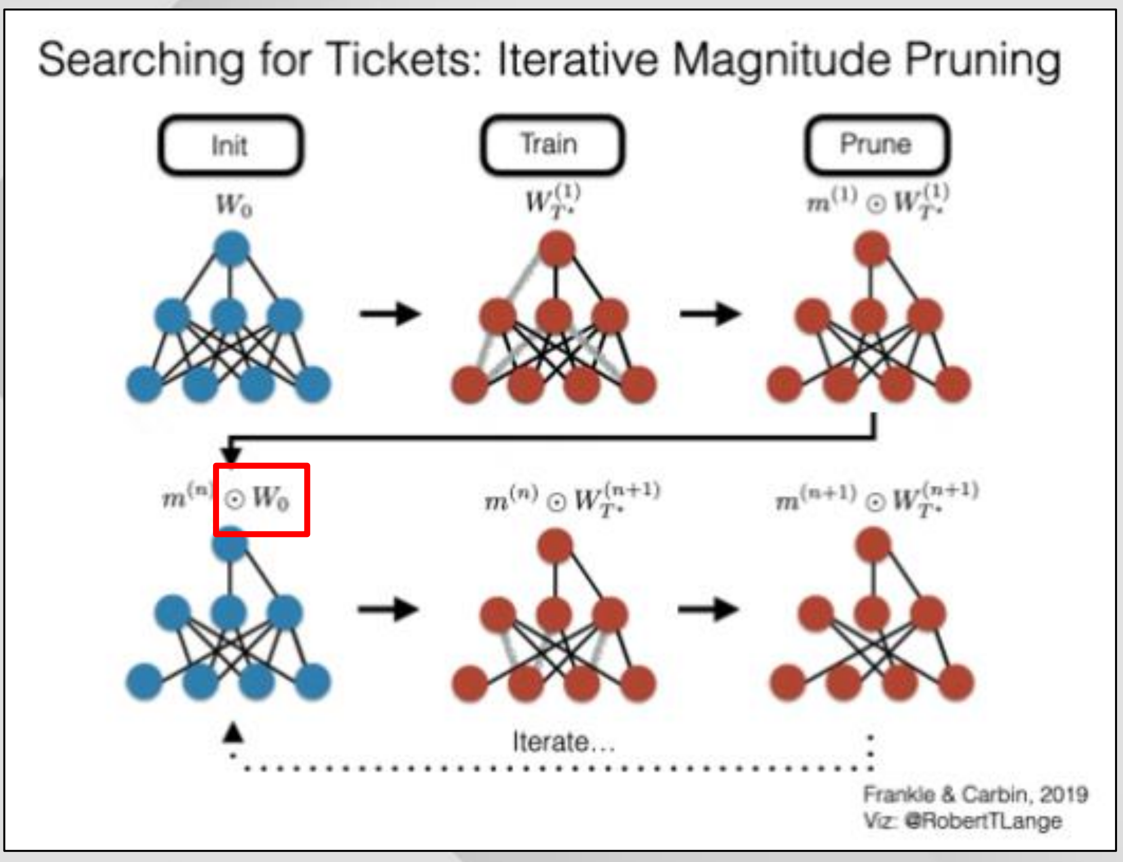
1) Randomly initialize the full network
2) Fully train the model for j iterations
3) Prune superfluous structure
4) **Reset each remaining weights to value before it was trained the first time** instead of re-randomize

$$m^{(1)} \odot W_0.$$

5) May repeat 2, 3 and 4 iteratively (Iterative Pruning)

→ what remains at the end is the **winning ticket**:

$$m^{(n)} \odot W_0$$


Searching for Tickets: Iterative Magnitude Pruning
Init
Train
Prune
W_0
$W_{T^*}^{(1)}$
$m^{(1)} \odot W_{T^*}^{(1)}$
$m^{(n)} \odot W_0$
$m^{(n)} \odot W_{T^*}^{(n+1)}$
$m^{(n+1)} \odot W_{T^*}^{(n+1)}$
Iterate...
Frankle & Carbin, 2019
Viz: @RobertTLange

Different weight-initialization

Procedure:

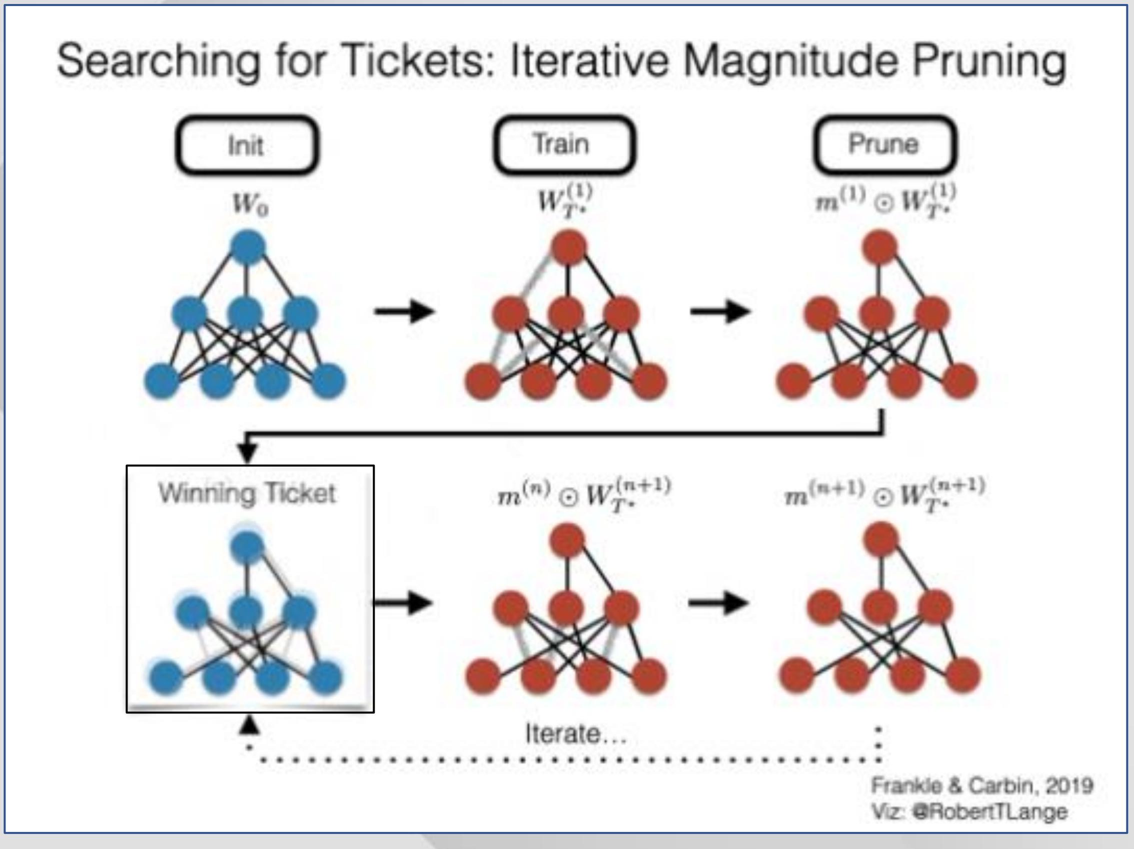
1) Randomly initialize the full network
2) Fully train the model for j iterations
3) Prune superfluous structure
4) **Reset each remaining weights to value before it was trained the first time** instead of re-randomize

$m^{(1)} \odot W_0.$

5) May repeat 2, 3 and 4 iteratively (Iterative Pruning)

→ what remains at the end is the **winning ticket**:

$m^{(n)} \odot W_0$


Searching for Tickets: Iterative Magnitude Pruning
Init
W_0
Train
$W_{T^*}^{(1)}$
Prune
$m^{(1)} \odot W_{T^*}^{(1)}$
Winning Ticket
$m^{(n)} \odot W_{T^*}^{(n+1)}$
$m^{(n+1)} \odot W_{T^*}^{(n+1)}$
Iterate...
Frankle & Carbin, 2019
Viz: @RobertTLange

Lottery Ticket Hypothesis

Results:
These pruned subnetworks:

- Are between 15% and 1% of original size
- Learn **at least as fast** as the original full network – when trained in isolation
- Are also **at least as accurate** as the original full network – when trained in isolation

Lottery ticket hypothesis:

”Dense, randomly-initialized, feed-forward networks **contain** sparse subnetworks (“Winning Tickets") that, when **trained in isolation**, reach test accuracy after training comparable to the original network in a similar number of iterations.”

Why “Winning Tickets”?
They won the lottery because they are characterized by a **successful combination of structure and initialized weights** have won the initialization lottery

Methodology

- **Pruning**
  - Unstructured
  - Smallest-magnitude weights set to 0
  - Iterative
  - Local

- **Main body of paper's hyperparameters:**
  - Optimization strategies:
    Adam optimizer or SGD with momentum
  - Gaussian Glorot initialization
  - Techniques like dropout, weight decay, batch norm, and residual connections

- **The hyperparameter selection** are evaluated using the validation set for:
  - the iteration of early-stopping (proxy for learning speed)
  - the accuracy at early-stopping

  BUT <u>use as generic as possible parameters</u>

- **Consider other hyperparameters:**
  - Optimization algorithms:
    SGD with and without momentum
  - Initialization strategies:
    Gaussian distributions with various standard deviations
  - Network sizes:
    larger and smaller hidden layers
  - Pruning strategies:
    faster and slower pruning rates

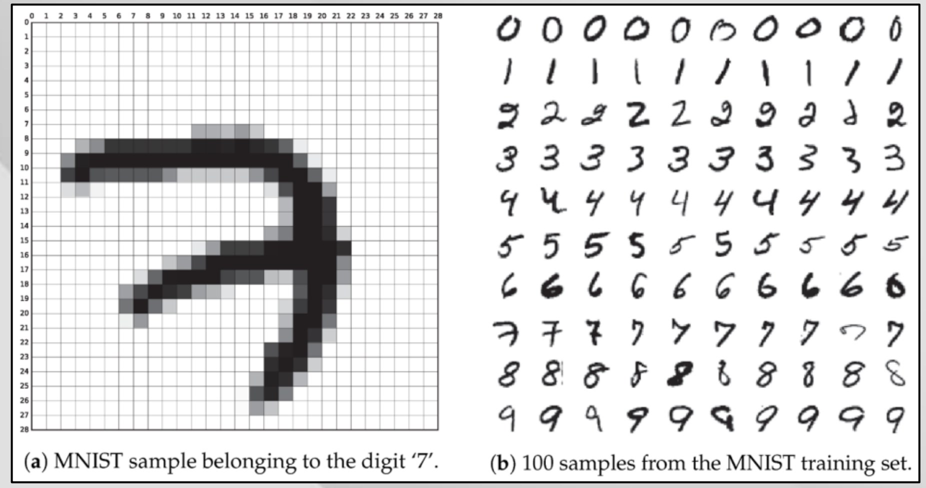
A fully-connected network for MNIST

The MNIST dataset:

- images of handwritten digits
- consists of 60,000 training examples and 10,000 test examples
- randomly sampled a 5,000-example validation set from the training set
- The training set is presented in mini-batches of 60 examples at each epoch
- Trained for 50,000 training iterations regardless of when early-stopping occurs

The full-connected networks:

- use a Lenet-300-100 architecture
  which comprises:
  - two fully-connected hidden layers (of 300 and 100 units each)
  - a ten-unit output layer
- 266K parameters


(a) MNIST sample belonging to the digit '7'. (b) 100 samples from the MNIST training set.

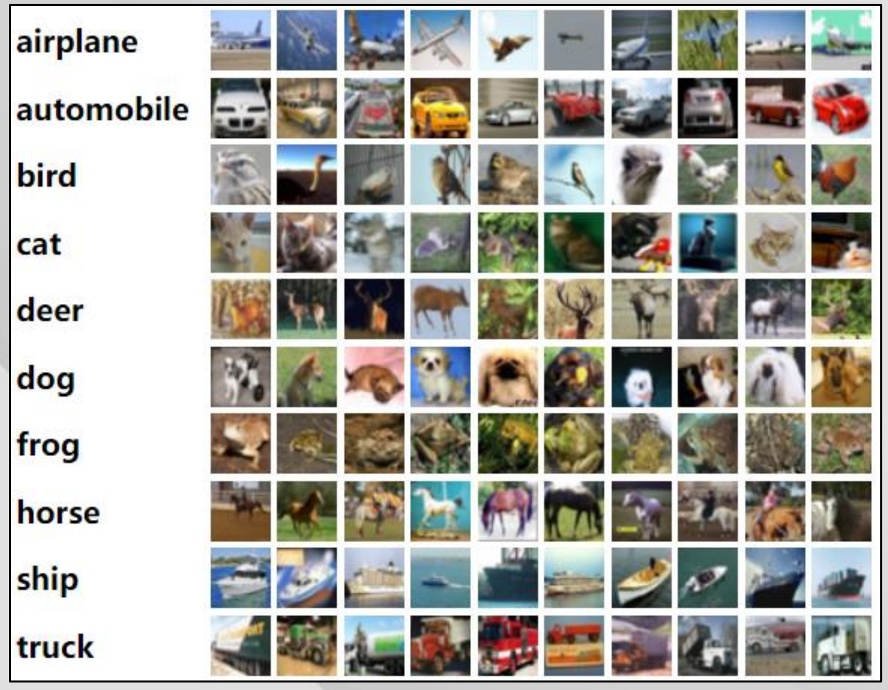
Convolutional Networks for CIFAR10

- **The CIFAR10 dataset:**
- 32x32 color (three-channel) images from 10 different categories
- Consist of 50,000 training examples and 10,000 test examples
- Randomly sampled a 5,000-example validation set from the training set
- training set is presented to the network in mini-batches of 60 examples at each epoch

- **Conv-2/4/6**
- scaled-down variant of VGG
- have 1, 2, and 3 modules, respectively
- Each module has:
  - two layers of 3x3 convolutional filters
  - followed by a maxpool layer with stride 2.

  After all of the modules there are:
  - two fully-connected layers of size 256
  - followed by an output layer of size 10

- VGG-19 and Resnet-18
- Deep convolutional networks
- Residual networks



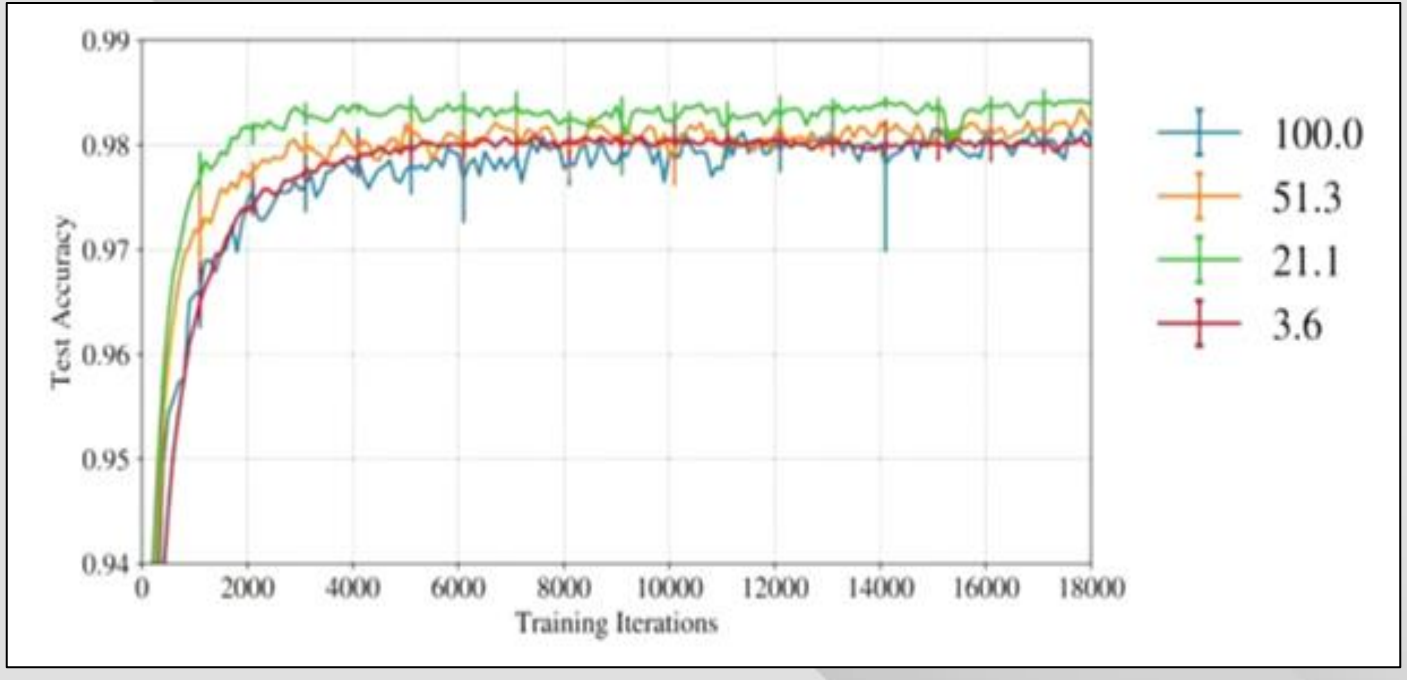
Pruning in LeNet architecture for MNIST

- Neural network with Pm > **3.6%** sparsity learn **faster** and it is **more accurate then the original** full network

- **Test accuracy** increases with pruning, when sparsity of mask is Pm >= 13.5%
improving by more than 0.3 percentage points (wrt original network)
after this point, accuracy decreases

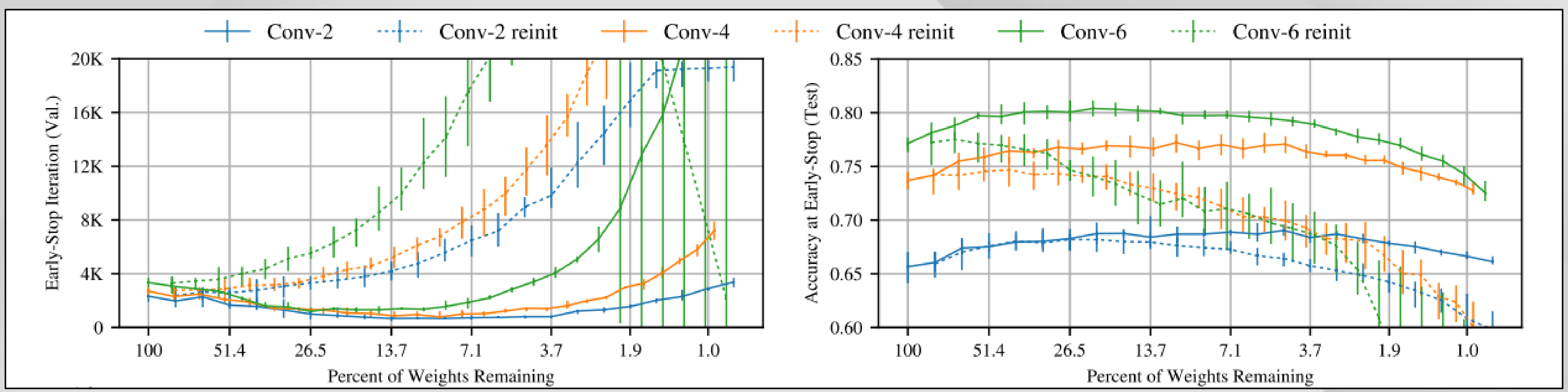
- **Learning rate** increases with pruning
until sparsity of mask is Pm = 21%
then it decreases

- The gap between training accuracy and
test accuracy is smaller for winning tickets,
meaning that they **improve generalizations**



Pruning in Conv-2/4/6 For CIFAR10

- **As network is pruned it learns faster and test accuracy rises**
- All three networks remain above their original average **test accuracy** when Pm > 2%.
  Test accuracy improves at best 3.4 percentage points for Conv-2 (Pm = 4.6%), 3.5 for Conv-4 (Pm = 11.1%), and 3.3 for Conv-6 (Pm = 26.4%).
- Tickets reach minimum validation loss at best 3.5x **faster** for Conv-2 (Pm = 8.8%), 3.5x for Conv-4 (Pm = 9.2%), and 2.5x for Conv-6 (Pm = 15.1%).
- The gap between test and training accuracy is smaller for winning tickets, indicating they **generalize better**.



What if Winning Tickets are re-initialized?

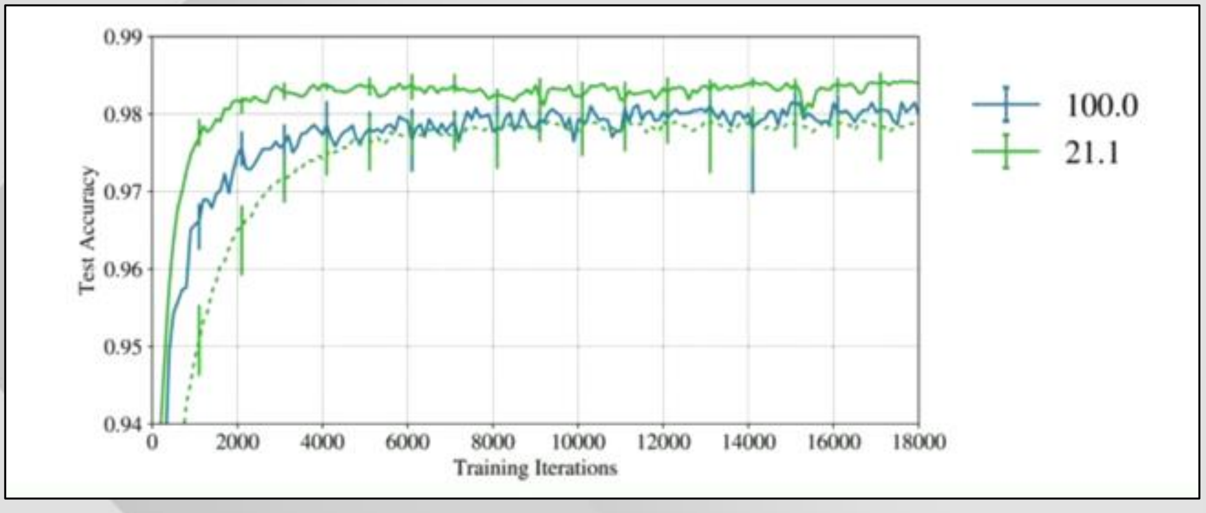
Results for <u>fully connected neural networks</u>:

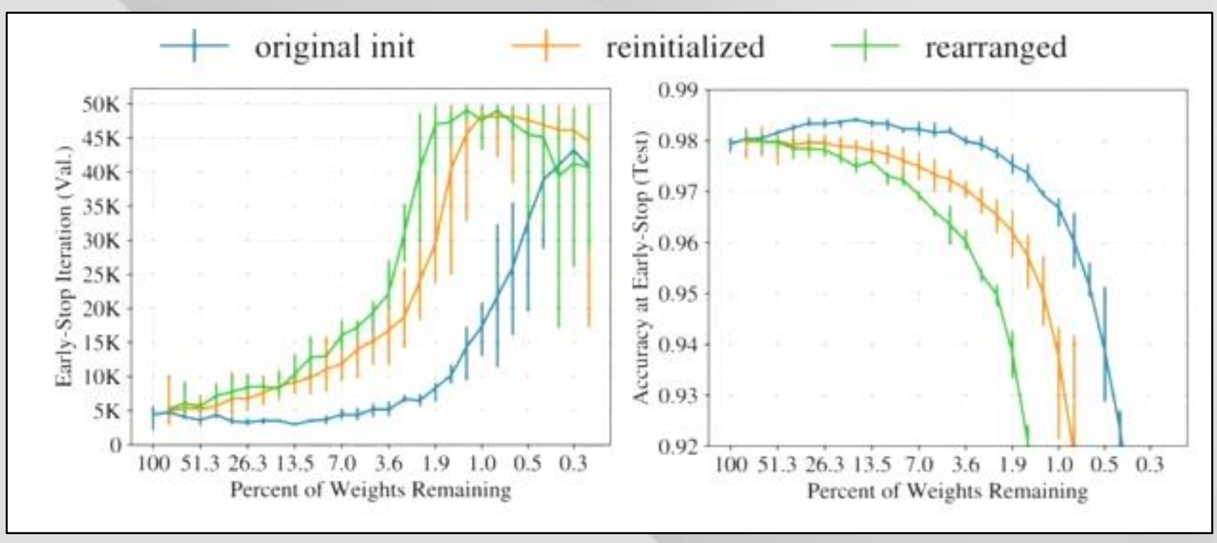
- **The reinitialized networks learn increasingly slower** and **lose test accuracy** after little pruning than the original network

- For example, when Pm = 21% the winning ticket reaches minimum validation loss 2.51x faster than when reinitialized and is half a percentage point more accurate.

- The winning tickets **generalize** substantially better than when randomly reinitialized.

→ it is not just a matter of the fact that they are smaller in size but it is a **matter of right weights and combination**





What if Winning Tickets are re-initialized?

Results for Conv-2/4/6:

- Networks take increasingly **longer to learn** upon continued pruning
- But unlike Lenet:
  test accuracy at early-stopping time initially remains steady and even improves for Conv-2/4
  at moderate levels of pruning the structure of the **winning tickets alone may lead to better accuracy**
  → structure of the winning ticket may alone lead to better accuracy

Why initialization is important?

× Hypo n1: initial weights are close to their final values after training
BUT the winning ticket weights move further than other weights.

✓ Hypo n2: benefit of the initialization is **connected to the optimization algorithm, dataset, and model.**
e.g. the winning ticket initialization might land in a region of the loss landscape that is particularly amenable to optimization by the chosen optimization algorithm

Other results:

- In **deeper networks** (like VGG-19 and Resnet-20) requires linear **learning rate warmup** to find winning tickets at higher learning rates.
- In deeper networks they do not prune layer by layer:
  → **global pruning** identifies smaller winning tickets for Resnet-18 and VGG-19
- **Dropout** increases initial test accuracy, and iterative pruning increases it further.
  → These improvements suggest that our iterative pruning strategy interacts with dropout in a **complementary way**.
- **One-shot pruning** does indeed find winning ticket and it less computational-intensive but iterative pruning find smaller tickets that learn faster and reach higher test accuracy.

Limitation of the Paper

The paper proves the existence of smaller subnetworks that

- can be trained in isolation without loss in accuracy or learning rate
- generalize better than original bigger network

BUT..

- No way of finding them before training (are **found retroactively**)
  → still need to train full model from 1 to 12 times to find them

- Fails on **deeper networks**
  they require linear learning rate warmup

- Work **with small visions tasks** (MNIST, CIFAR10)

- Use **only small-magnitude unstructured pruning**
  no structured pruning or non-magnitude pruning
  → the resulting architectures are not optimized for modern libraries or hardware

Solutions - Need to find retroactively

- **Drawing Early-Bird Tickets: Towards more efficient training of deep networks - You, Li, Xu, Fu, Wang, Chen, Lin, Wang, & Baraniuk (2020)**
  - Lottery tickets can be found early in training (early birds).
  - Empirically observe that the pruning masks change significantly during the first epochs of training but appear to converge soon.
  - How? Each training iteration, they compute a pruning mask. If the mask in the last iteration and this one have a mask distance (using Hamming distance) below a certain threshold, the network stops to prune.

- **Pruning Neural Networks Without Any Data by Iteratively Conserving Synaptic Flow – Tanaka, Kunin, Yamins, Ganguli (2020)**
  - This paper focuses on calculating pruning at initialization with no data.
  - It outperforms existing state-of-the-art pruning algorithms at initialization.
  - They developed an algorithm (SynFlow) define an iterative procedure which generates a mask that preserves the flow of synaptic strengths through the initialized network.

- **Proving the Lottery Ticket Hypothesis: Pruning is All You Need - Malach , Yehudai, Shalev-Shwartz , Shamir (2020)**
  - For every bounded distribution and every target network with bounded weights, a sufficiently over-parameterized neural network with random weights contains a subnetwork with roughly the same accuracy as the target network, without any training.

Solutions

-Issue with deeper networks

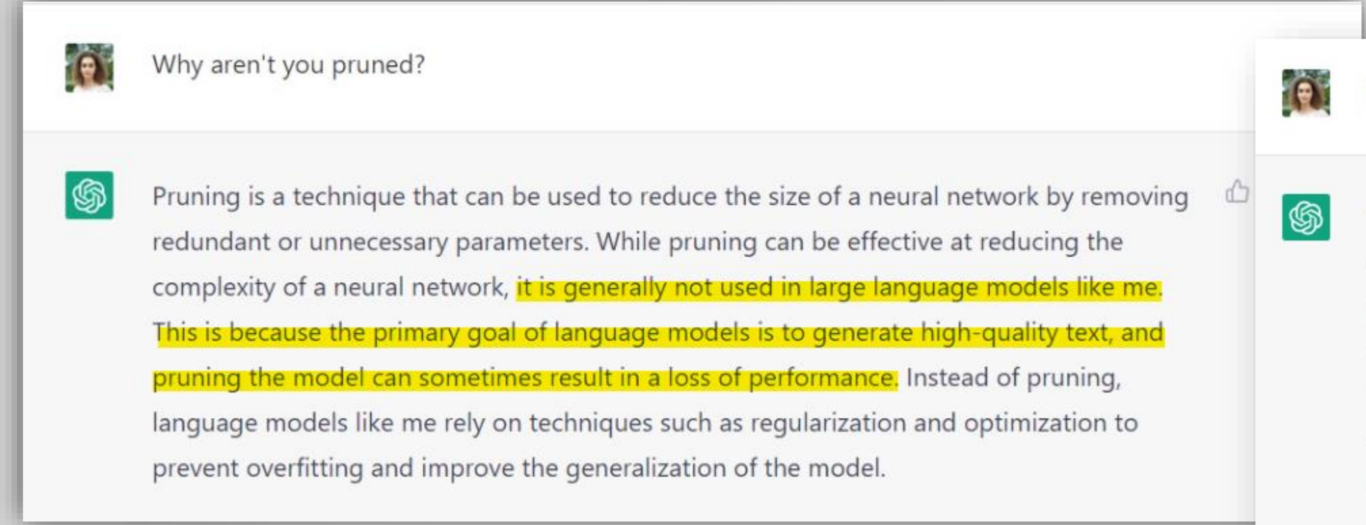
- **Stabilizing the Lottery Ticket Hypothesis – Frankle, Dziugaite, Roy, Carbin (2019)**
  - Modify IMP to search for subnetworks by <u>resetting the weights to those found after a small number of k training iterations</u> (0.1% to 7% through) rather than at iteration 0.
  - Find small subnetworks of deeper networks (e.g., 80% sparsity on Resnet-50) that match the accuracy of the original network on more challenging tasks (e.g., ImageNet).
  - No need of learning rate warmup.

-Other Procedures

- **Comparing Rewinding and Fine-tuning in Neural Network Pruning - Renda, Frankle, & Carbin (2020)**
  - Compare fine-tuning, weight rewinding, learning rate rewinding.
  - Learning rate rewinding outperforms weight rewinding.
- **Zhou, Lan, Liu, & Yosinski (2019) - Deconstructing lottery tickets: Zeros, signs, and the supermask**
  - Compare different scoring methods
  - Compare different initialization (random reinitialisation, reshuffling of kept weights and a constant initialization)

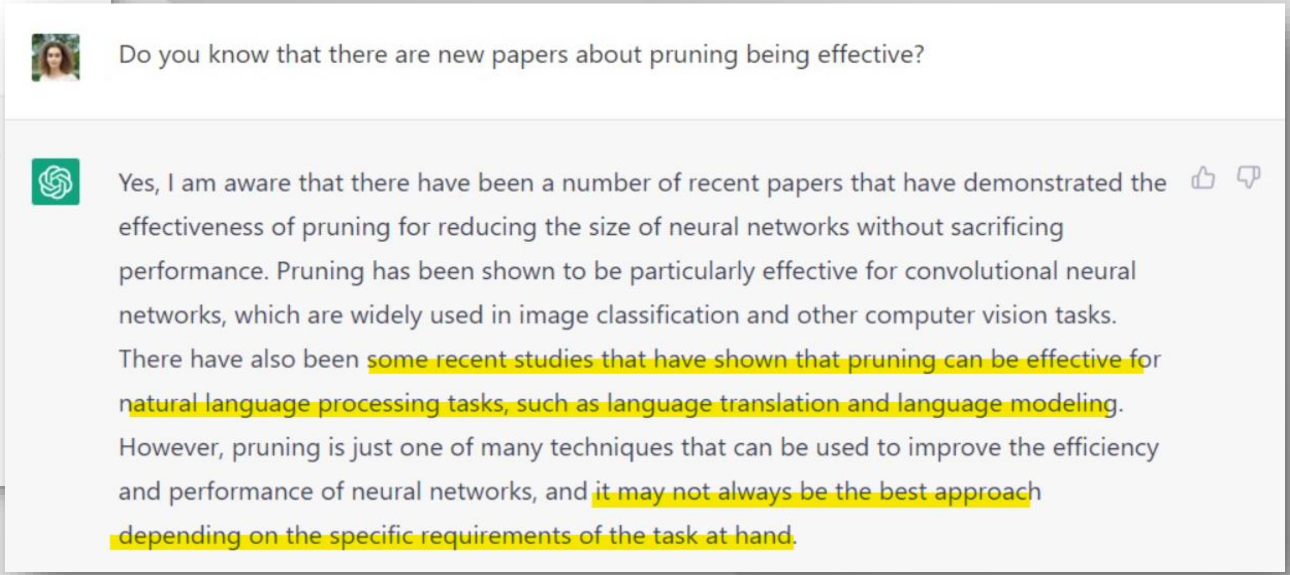
Solutions – Other types of tasks

- **Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP - Yu, Edunov, Tian, & Morcos (2019)**
  - For NLP, examine both recurrent LSTM models and large-scale Transformer models
  - For RL, analyze a number of discrete-action space tasks, including both classic control and pixel control.
  - Confirm that winning ticket initializations generally outperform parameter-matched random initializations, even at extreme pruning rates for both NLP and RL.



Why aren't you pruned?

Pruning is a technique that can be used to reduce the size of a neural network by removing redundant or unnecessary parameters. While pruning can be effective at reducing the complexity of a neural network, it is generally not used in large language models like me. This is because the primary goal of language models is to generate high-quality text, and pruning the model can sometimes result in a loss of performance. Instead of pruning, language models like me rely on techniques such as regularization and optimization to prevent overfitting and improve the generalization of the model.



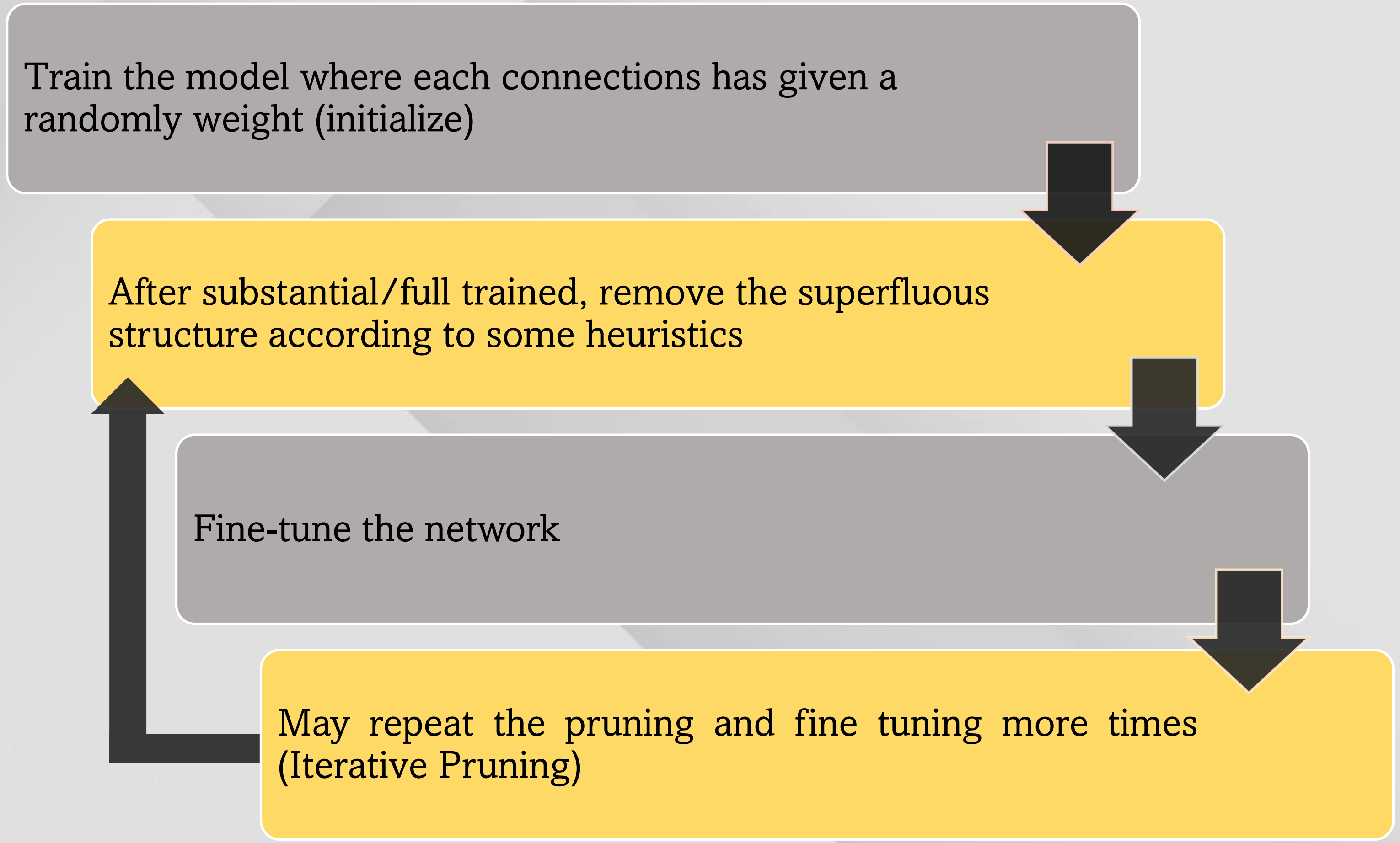
Do you know that there are new papers about pruning being effective?

Yes, I am aware that there have been a number of recent papers that have demonstrated the effectiveness of pruning for reducing the size of neural networks without sacrificing performance. Pruning has been shown to be particularly effective for convolutional neural networks, which are widely used in image classification and other computer vision tasks. There have also been some recent studies that have shown that pruning can be effective for natural language processing tasks, such as language translation and language modeling. However, pruning is just one of many techniques that can be used to improve the efficiency and performance of neural networks, and it may not always be the best approach depending on the specific requirements of the task at hand.

Thanks for the attention!

Pruning in Literature

Process of Pruning after training



Train the model where each connections has given a randomly weight (initialize)

After substantial/full trained, remove the superfluous structure according to some heuristics

Fine-tune the network

May repeat the pruning and fine tuning more times (Iterative Pruning)

Solution - Generalizations

- **Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask – Zhou, Lan, Liu, Yosinski**
    - Study the three critical components of the Lottery Ticket (LT) algorithm, showing that each may be varied significantly without impacting the overall results
    - Finally, we discover the existence of Supermasks, masks that can be applied to an untrained, randomly initialized network to produce a model with performance far better than chance (86% on MNIST, 41% on CIFAR-10).

- **Linear Mode Connectivity & the Lottery Ticket Hypothesis - Frankle, Dziugaite, Roy, & Carbin (2020a)** -
    - When we train a neural network, we usually do so on a random ordering of data batches. Each batch is used to evaluate a gradient of the loss with respect to the network parameters. After an epoch the batches are usually shuffled and we continue with the next epoch. The sequence of batches can be viewed as a source of noise which we inject into the training procedure.

  → we would like the network training procedure to be somewhat robust to such noise