THE LOTTERY TICKET HYPOTHESIS:

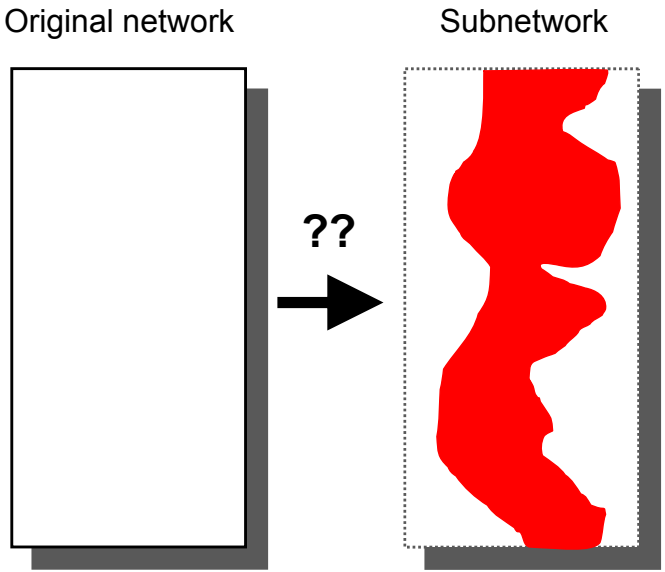
FINDING SPARSE, TRAINABLE NEURAL NETWORKS

Jonathan Frankle, Michael Carbin

Published as a conference paper at ICLR 2019

Aalto University
School of Science

What is the Lottery Ticket Hypothesis about?



- Is there a subnetwork with
  - Better results
  - Shorter training time
  - Notably fewer parameters
  - Trainable from beginning?

Agenda

- The Lottery Ticket Hypothesis
- Winning Tickets
- Pruning
- Identifying Winning Tickets
- Testing the hypothesis
- Winning Tickets in Fully-Connected Networks
- Winning Tickets in Convolutional Networks
- Winning Tickets in VGG
- Winning Tickets in Resnet
- Conclusions

The Lottery Ticket Hypothesis

The Lottery Ticket Hypothesis. *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.*

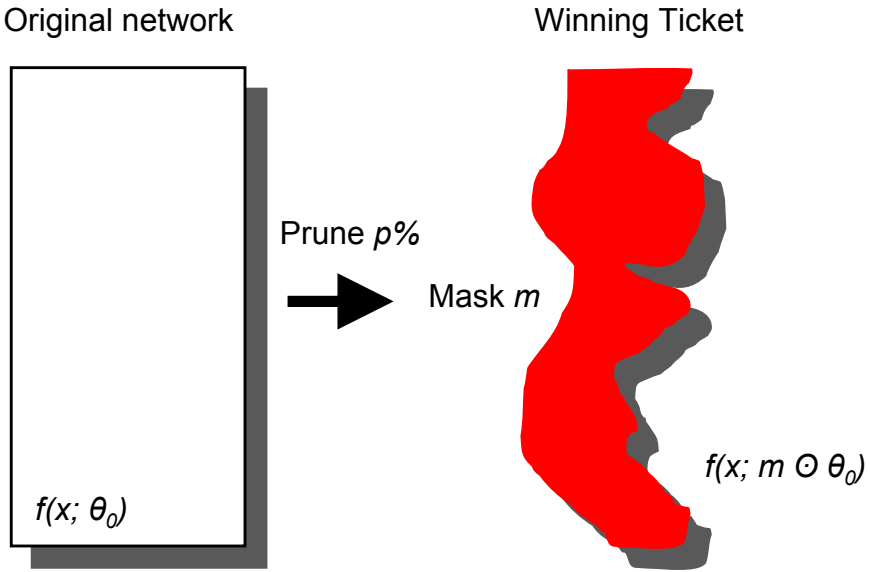
The Lottery Ticket Hypothesis

- The lottery ticket hypothesis predicts that there excists:
  - a subnetwork of the original network that
  - gives as good or better results with
  - shorter or most as long training time and
  - with notably fewer parameters than the original network
  - when initialized with the same parameters (discarding the parameters of the removed part of the network) as the initial network.

Winning Tickets

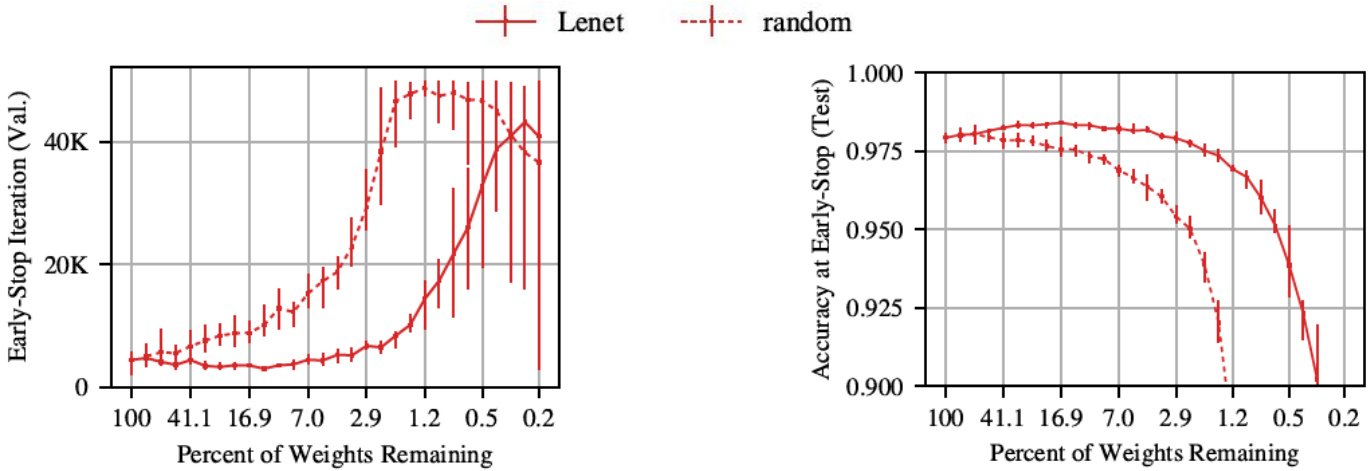
- Subnetworks predicted by the Lottery Ticket Hypothesis
- Found from fully-connected and convolutional feed-forward networks
- Standard pruning technique automatically uncovers them
- Initialized with the same parameters (discarding the parameters of the removed part of the network) as the initial network

The Lottery Ticket Hypothesis and Winning Tickets



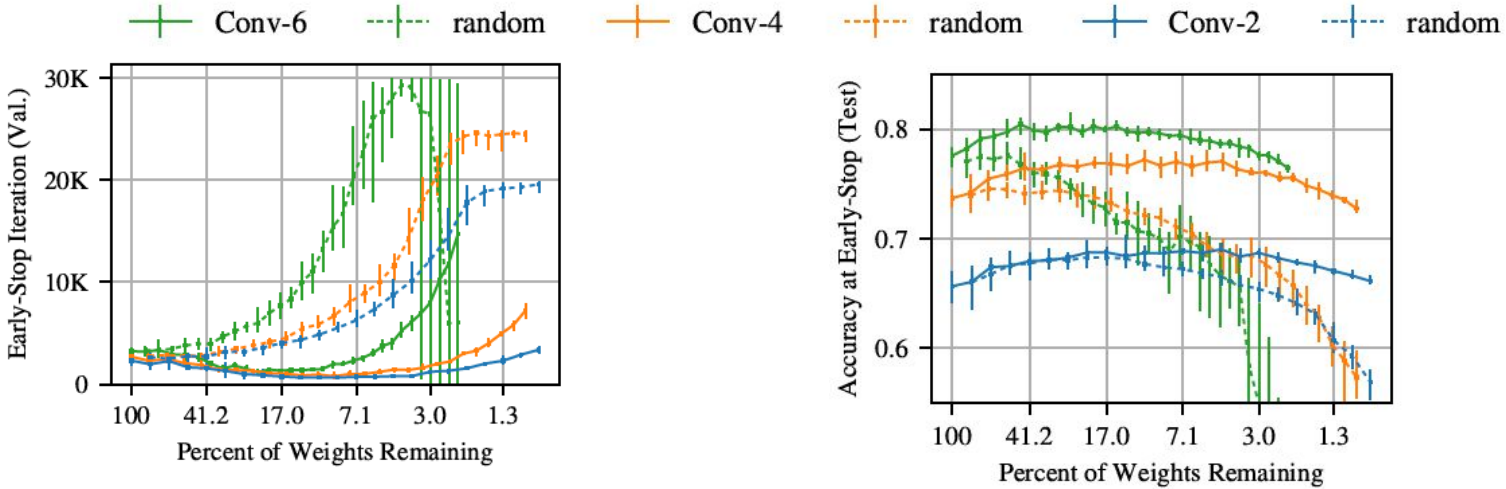
- Winning Ticket gives
  - Better or same results
  - Shorter or same training time
  - Notably fewer parameters
  - Is trainable from the beginning

Winning Tickets and random sampling



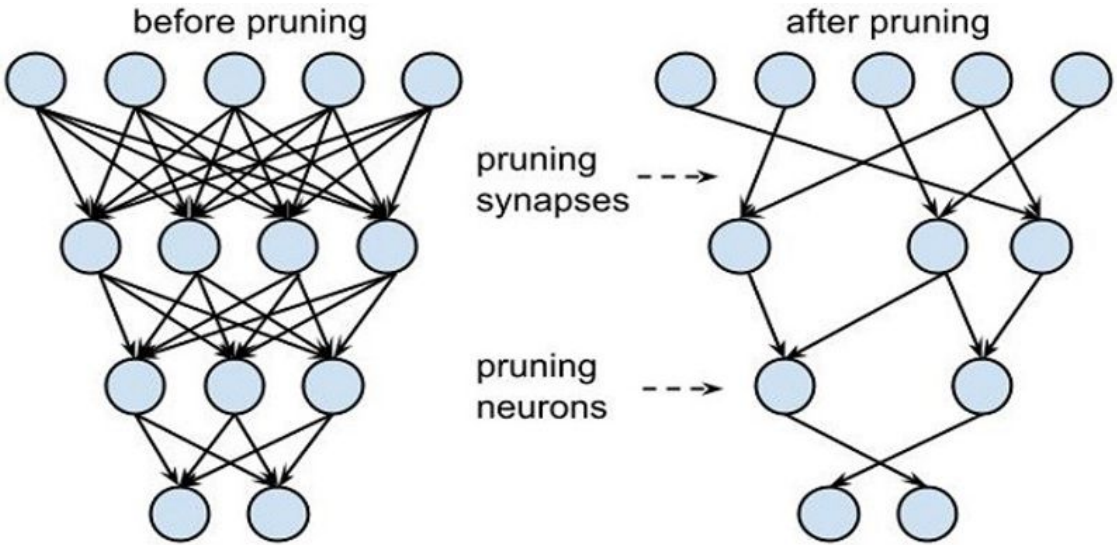
The iteration at which early-stopping would occur (left) and the test accuracy at that iteration (right) of the Lenet architecture for MNIST when trained starting at various sizes. Dashed lines are randomly sampled sparse networks (average of ten trials). Solid lines are winning tickets (average of five trials).

Winning Tickets and random sampling



The iteration at which early-stopping would occur (left) and the test accuracy at that iteration (right) of the Conv-2, Conv-4, and Conv-6 architectures for CIFAR10 when trained starting at various sizes. Dashed lines are randomly sampled sparse networks (average of ten trials). Solid lines are winning tickets (average of five trials).

Pruning



Jesus Rodriguez: How the Lottery Ticket Hypothesis is Challenging Everything we Knew About Training Neural Networks
https://towardsdatascience.com/how-the-lottery-ticket-hypothesis-is-challenging-everything-we-knew-about-training-neural-networks-e56da4b0da27

Pruning Rate and Sparsity

- *p%* is the Pruning Rate
- P_m is the Sparsity of the pruned network (mask)

- E.g. P_m = 25% when *p%* = 75% of weights are pruned

Pruning the network

- Remove random weights
- Remove small weights
- Remove weights which have the least effect on the solution

  => Optimal Brain Damage or OBD

Pruning the network with OBD

- Optimal Brain Damage (OBD) (Le Cun, Denker, and Solla 1990)
- Remove weights with the smallest saliency
- Saliency:

$$\text{saliency} = \delta E_{ji} W_{ji}^2$$

- Sensitivity of error function to small changes of the weight:

$$\delta E_{ji} \approx \frac{\partial^2 E}{\partial W_{ji}^2}$$

LiMin Fu: Neural Networks in Computer Intelligence (1994), page 92

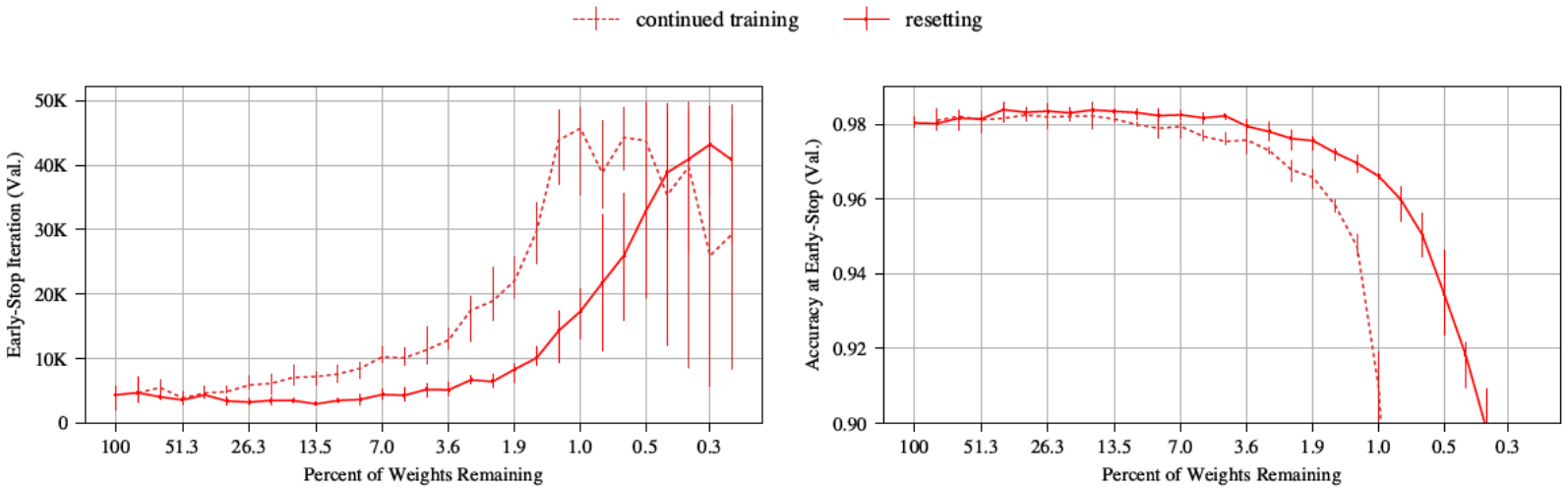
Identifying Winning Tickets

- One-shot pruning:
  1. Randomly initialize a neural network $f(x; \theta_0)$, with initial parameters θ_0
  2. Train the network for j iterations, arriving at parameters θ_j
  3. Prune $p\%$ of the parameters in θ_j, creating a mask m
  4. Reset the remaining parameters to their value in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.
- Iterative pruning:
  1. Randomly initialize a neural network $f(x; \theta_0)$, with initial parameters θ_0
  2. Train the netowork for j iterations, arriving at parameters θ_j
  3. Prune $p^{1/n}\%$ of the parameters in θ_j, creating a mask m
  4. Reset the remaining parameters to their value in θ_0, creating network $f(x; m \odot \theta_0)$
  5. Repeat n times from 2
  6. Final network is a winning ticket $f(x; m \odot \theta_0)$.

Iterative pruning using the resetting and continued training strategies

- Two alternative strategies for executing the iterative pruning
- Iterative pruning with resetting
  - Train and partially prune the network
  - Reset the remaining network weights to their initial walues
  - Continue the process until done
- Iterative pruning with continued training
  - Train and partially prune the network
  - Keep the already trained weights of the remaining network
  - Continue the process until dome
- Iterative pruning with reset maintains higher validation accuracy and faster early-stopping times to smaller network sizes

Iterative pruning using the resetting and continued training strategies: example



The early-stopping iteration and accuracy at early-stopping of the iterative lottery ticket experiment on the Lenet architecture when iteratively pruned using the resetting and continued training strategies.

Testing the hypothesis

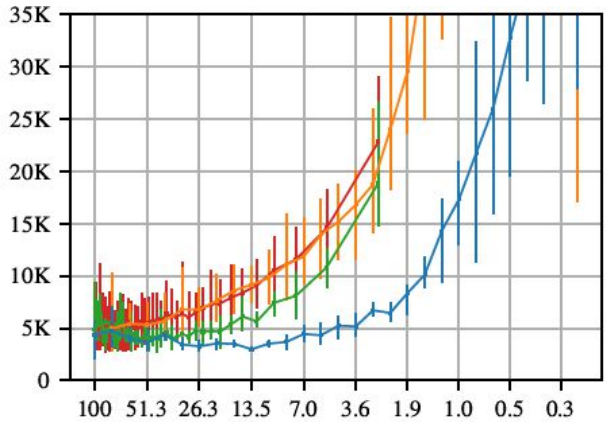
- Empirically study the lottery ticket hypothesis

- Architectures used in the studying
  - Fully connected networks
  - Convolutional networks
  - Networks evocative of the architectures and techniques used in practice

Architectures tested

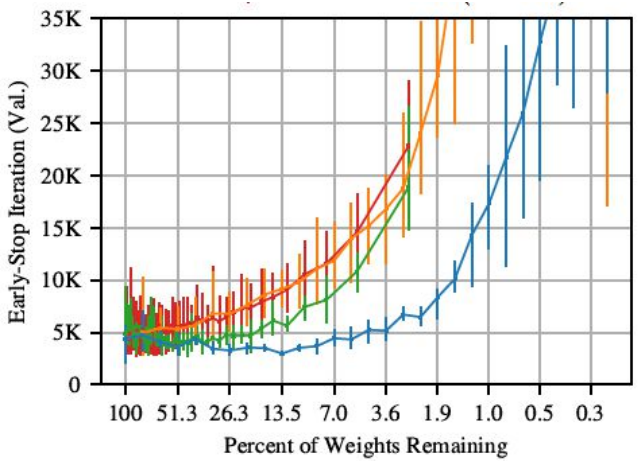
Network	Lenet	Conv-2	Conv-4	Conv-6	Resnet-18	VGG-19
Convolutions		64, 64, pool	64, 64, pool 128, 128, pool	64, 64, pool 128, 128, pool 256, 256, pool	16, 3x[16, 16] 3x[32, 32] 3x[64, 64]	2x64 pool 2x128 pool, 4x256, pool 4x512, pool, 4x512
FC Layers	300, 100, 10	256, 256, 10	256, 256, 10	256, 256, 10	avg-pool, 10	avg-pool, 10
All/Conv Weights	266K	4.3M / 38K	2.4M / 260K	1.7M / 1.1M	274K / 270K	20.0M
Iterations/Batch	50K / 60	20K / 60	25K / 60	30K / 60	30K / 128	112K / 64
Optimizer	Adam 1.2e-3	Adam 2e-4	Adam 3e-4	Adam 3e-4	← SGD 0.1-0.01-0.001 Momentum 0.9 →	
Pruning Rate	fc20%	conv10% fc20%	conv10% fc20%	conv15% fc20%	conv20% fc0%	conv20% fc0%

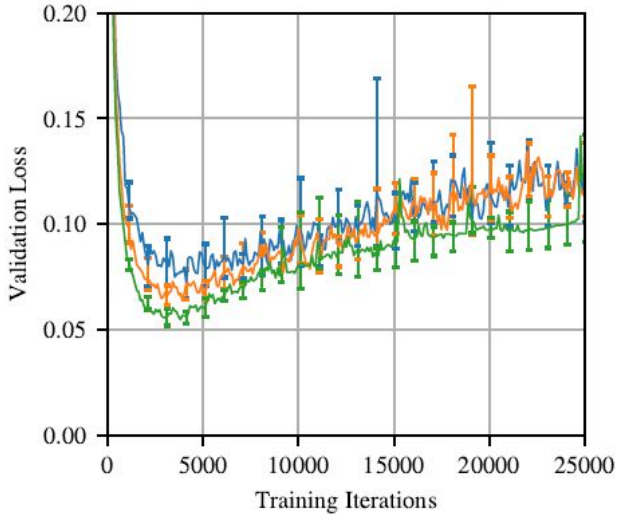
Statistical handling and visualization



- Average of *x* trials
- Error bars for the
  - Minimum value
  - Maximum value

Early-Stopping Criterion identification



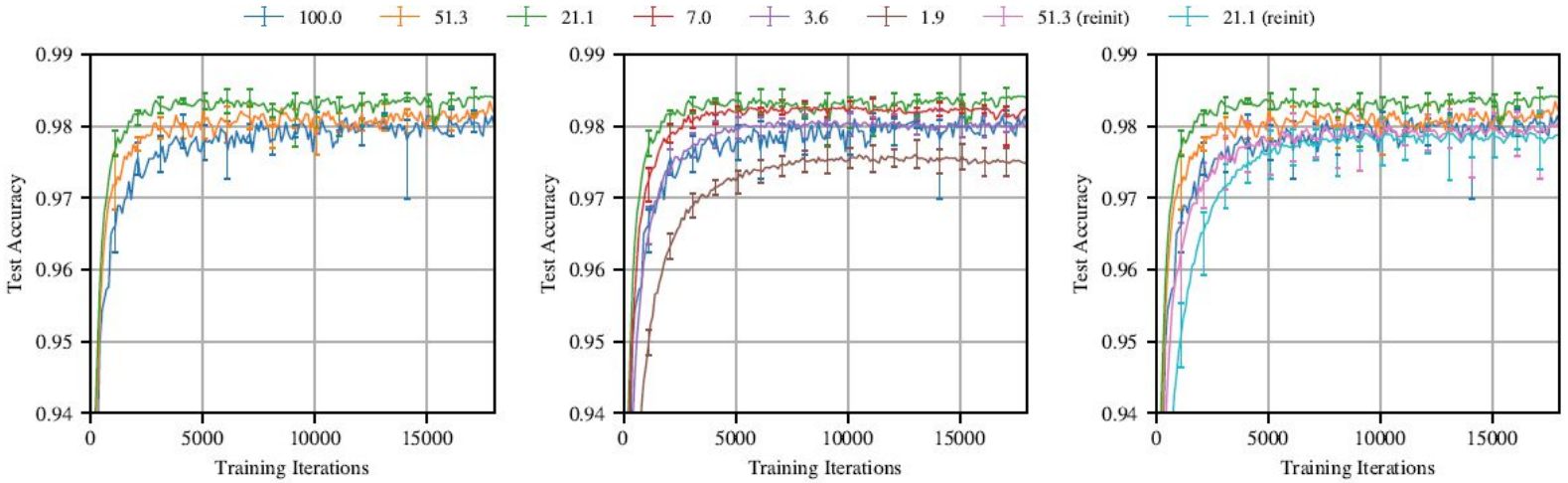


Early-Stoppping Criterion is the iteration of minimum validation loss. Validation loss initially drops, after which it forms a clear bottom and then begins increasing again. Our early-stopping criterion identifies this bottom.

Winning Tickets in Fully-Connected Networks

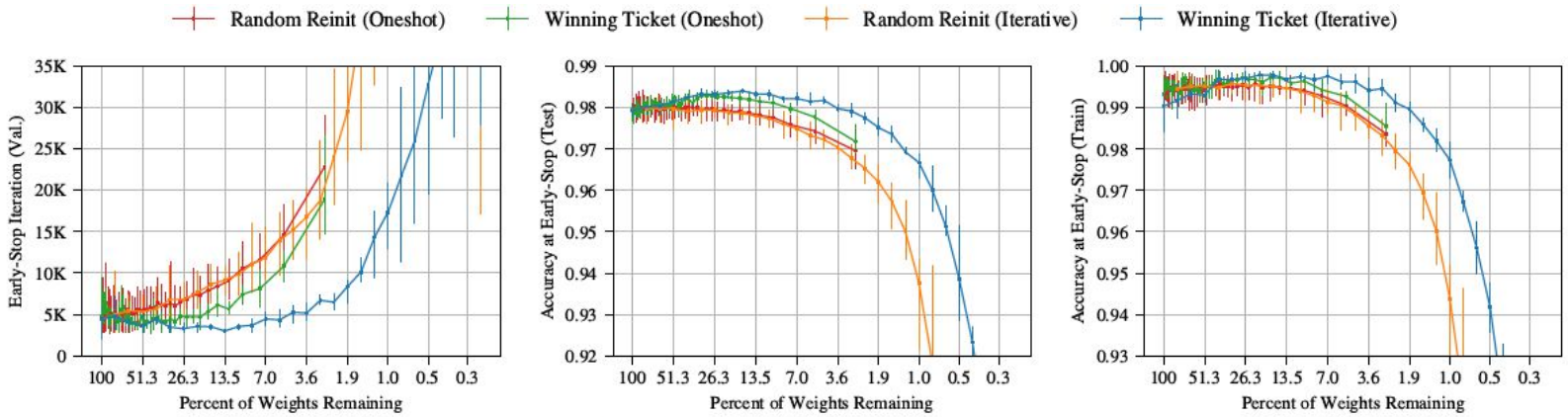
- Fully connected Lenet-300-100 architecture (LeCun at.al., 1998)
- MNIST data
- Layer-wise pruning
- Output layer connections pruned at half the rare

Winning Tickets in Fully-Connected Networks



Test accuracy on Lenet (iterative pruning) as training proceeds. Each curve is the average of five trials. Labels are P_m—the fraction of weights remaining in the network after pruning. Error bars are the minimum and maximum of any trial.

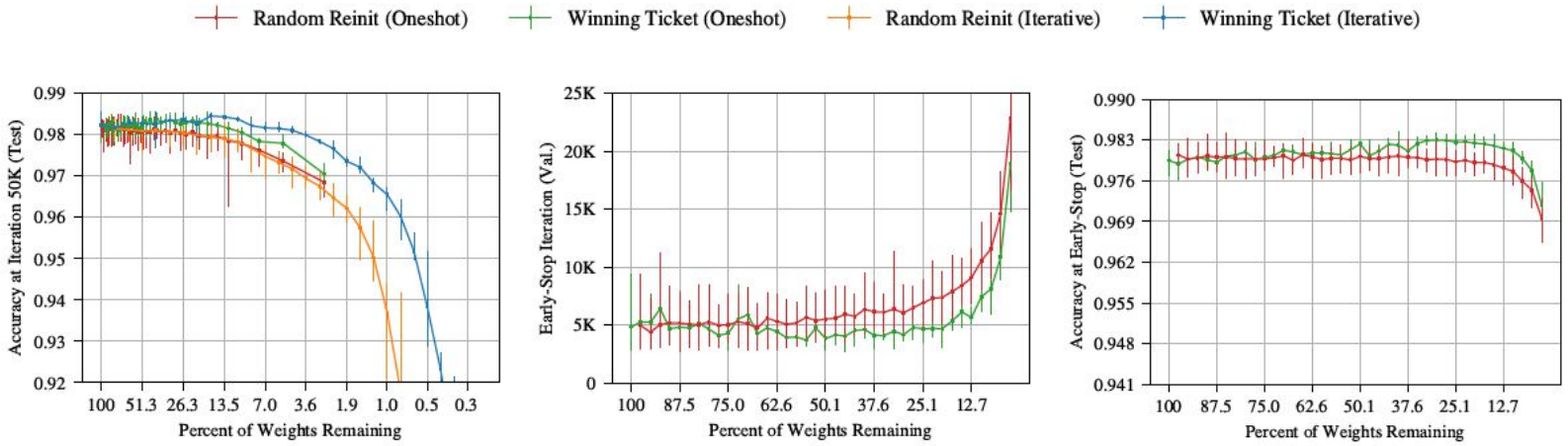
Winning Tickets in Fully-Connected Networks



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

Early-stopping iteration and accuracy of Lenet under one-shot and iterative pruning.

Winning Tickets in Fully-Connected Networks



(b) Accuracy at end of training.

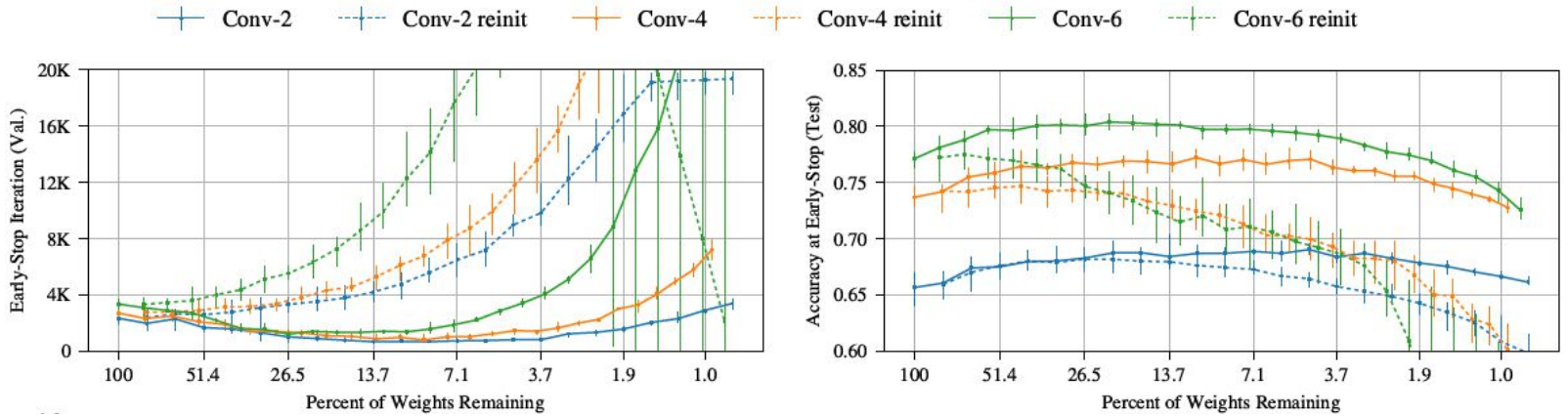
(c) Early-stopping iteration and accuracy for one-shot pruning.

Figure b: At iteration 50,000 (end of training), training accuracy 100% for Pm 2% for iterative winning tickets.
Figure c: Early-stopping iteration and accuracy of Lenet for one-shot pruning.

Winning Tickets in Convolutional Networks

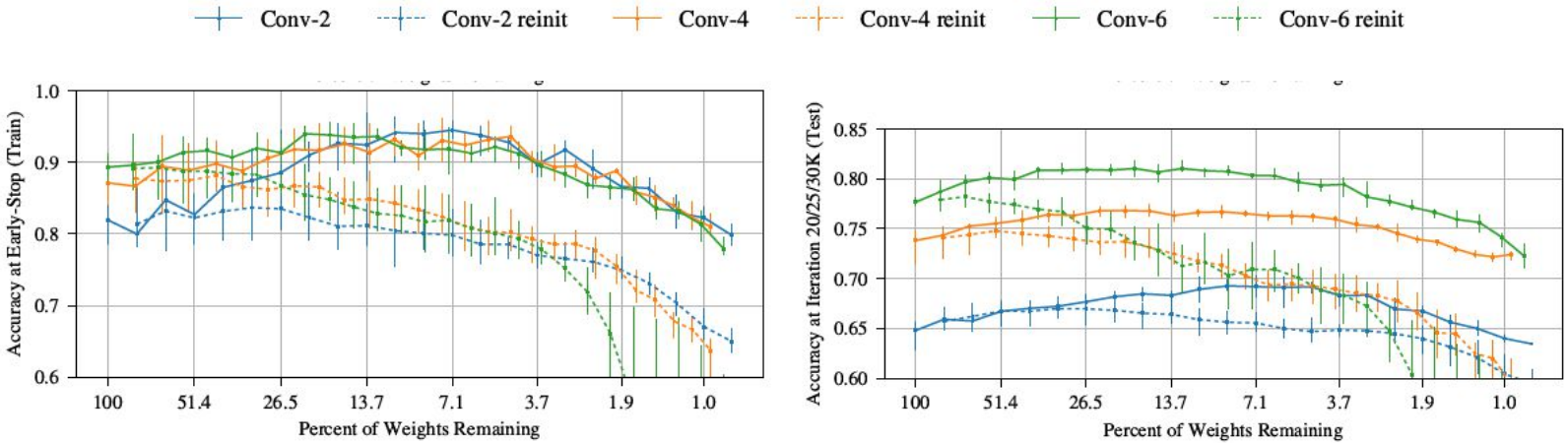
- Convolutional networks Conv-2, Conv-4, Conv-6
  - Scaled-down variants of the VGG (Simonyan & Zisserman, 2014)
  - Architecture:
    - 2, 4 or 6 convolutional layers
    - 2 fully-connected layers
    - Max-pooling after every two convolutional layers
- CIFAR10 data
- Layer-wise pruning
- Output layer connections pruned at half the rare
- Dropout with rate 0.5 tested

Winning Tickets in Convolutional Networks



Early-stopping iteration and test accuracy of the Conv-2/4/6 architectures when iteratively pruned and when randomly reinitialized. Each solid line is the average of five trials; each dashed line is the average of fifteen reinitializations (three per trial).

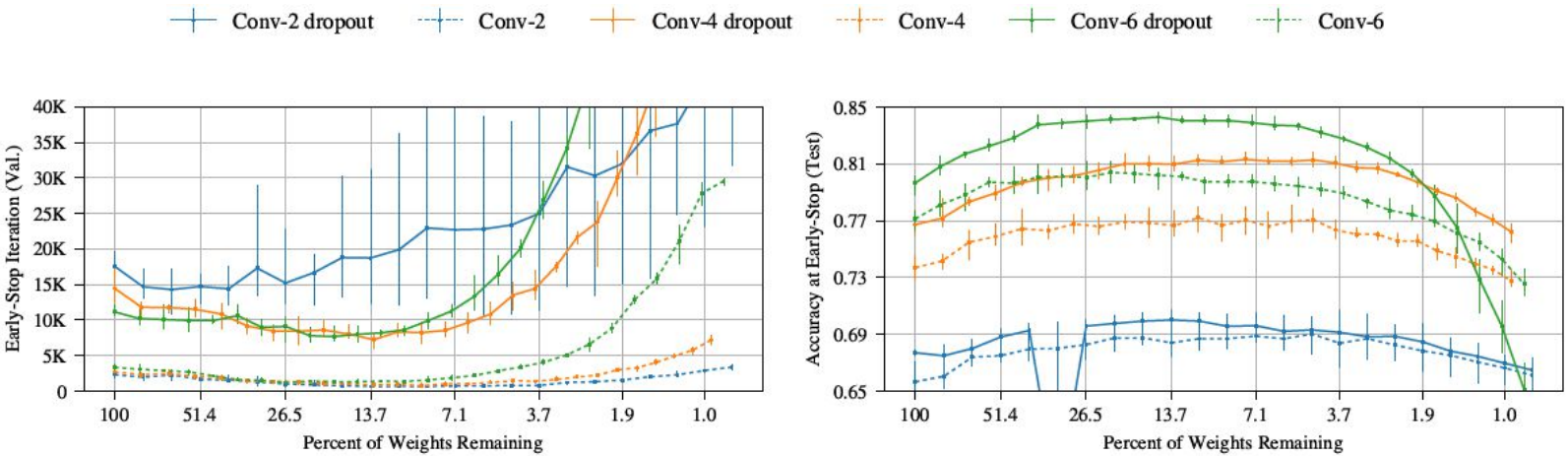
Winning Tickets in Convolutional Networks



Training accuracy of the Conv-2/4/6 architectures when iteratively pruned and when randomly reinitialized. Each solid line is the average of five trials; each dashed line is the average of fifteen reinitializations (three per trial).

Test accuracy of winning tickets at iterations corresponding to the last iteration of training for the original network (20,000 for Conv-2, 25,000 for Conv-4, and 30,000 for Conv-6); at this iteration, training accuracy about 100% for P_m >= 2% for winning tickets

Winning Tickets in Convolutional Networks

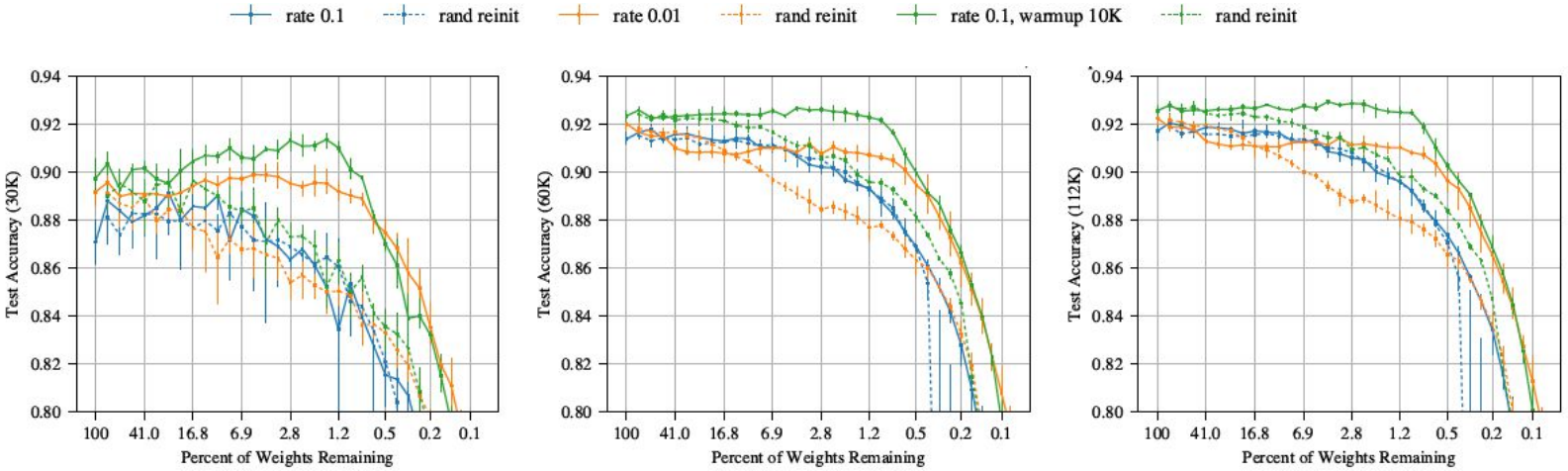


Early-stopping iteration and test accuracy at early-stopping of Conv-2/4/6 when iteratively pruned and trained with dropout. The dashed lines are the same networks trained without dropout (the solid lines in the two previous slides). Learning rates are 0.0003 for Conv-2 and 0.0002 for Conv-4 and Conv-6.

Winning Tickets in VGG

- VGG-19 is a VGG-style deep convolutional network (Simonyan & Zisserman, 2014) and adapted for CIFAR10 (Liu et al. 2019)
- CIFAR10 data
- Global pruning
- Output layer connections pruned at half the rare
- Warmup from 0 to initial learning rate over *k* iterations

Winning Tickets in VGG

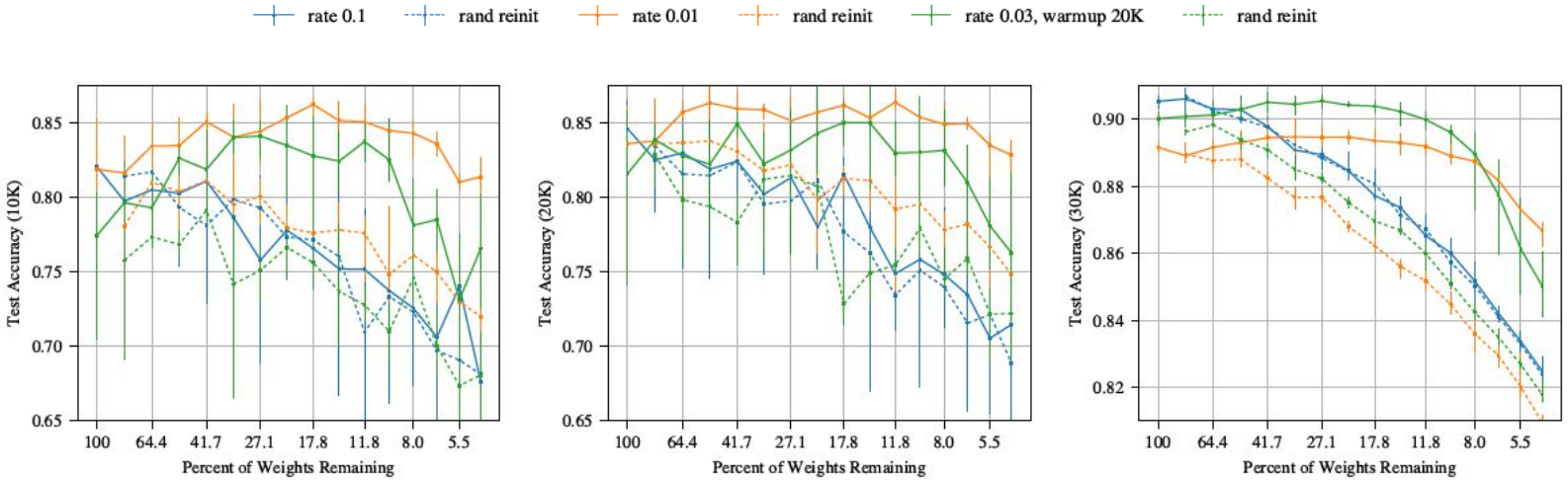


Test accuracy (at 30K, 60K, and 112K iterations) of VGG-19 when iteratively pruned

Winning Tickets in Resnet

- Resnet-18 is a is a 20 layer convolutional network with residual connections designed for CIFAR10 (He et al. 2016)
- CIFAR10 data
- Global pruning
- Output layer connections pruned at half the rare
- Warmup from 0 to initial learning rate over *k* iterations

Winning Tickets in Resnet



Test accuracy (at 10K, 20K, and 30K iterations) of Resnet-18 when iteratively pruned

Conclusions

- Architechtures studied reliably contain Winning Tickets
- Lottery Ticket Hypothesis proposes that this property applies in general
- Follow-up questions:
  - Importance of Winning Ticket initialization
  - Importance of Winning Ticket structure
  - Improved generalization of Winning Tickets
  - Implications for neural network optimization

Discussion!

Thank you!