An Analysis of Neural Network Pruning in Relation to the Lottery Ticket Hypothesis

Thijs Havinga   Rishabh Sawhney

Abstract

In this paper, we analyze the novel "lottery ticket" approach for pruning neural networks and compare it to traditional techniques on dense feed-forward neural networks by the means of identifying winning tickets. We explore the precise conditions required to identify and make use of these winning tickets and explore perspectives on the hypothesis from other work. We conclude that there is demonstrable evidence in support of the hypothesis, but revealing winning tickets in any neural network remains a hard task.

1 Introduction

Neural network pruning refers to the process of removing weights from (dense) neural networks with the goal of improving efficiency in terms of energy and memory consumption (Han et al., 2015). An early example of neural network pruning is named *Optimal Brain Damage* and was proposed by LeCun et al. in 1990. Optimal Brain Damage (OBD) takes a reasonably sized network and selectively deletes half of the weights based on a special metric, *saliency*, in order to come up with a sub-network which could perform just as well if not better than the main network. Many more methods for pruning have since been proposed as illustrated in subsection 2.1. The common goal of these methods is the reduction of the number of weights in the model, while sustaining an approximately equal model accuracy. Figure 1 shows an illustration of pruning in general.

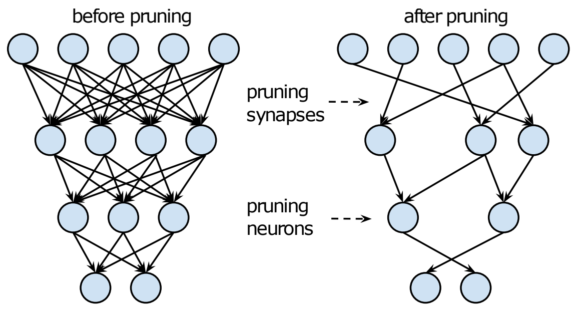

Figure 1: Illustration of unstructured pruning. Weights are removed by the pruning algorithm based on some metric resulting in a sparser network. From Han et al. (2015).

It has been showed to be a hard task to train a pruned neural network again from a new random initialization. When re-initializing such a sparse network with random weights it generally achieves a lower accuracy. For example, Li et al. noticed this in Li et al. (2016) and claimed a "difficulty of training a network with a small capacity". As can be seen in Table 1, pruned models that have been retrained from scratch perform systematically (much) worse than their pruned and unpruned original model.

If such a sparse network is hard to train, why has it proven successful to train an over-parameterized network and prune it until a sparse network remains? Han et al. (2015) explains this through the existence of "fragile co-adapted features" in the network as described in Yosinski et al. (2014). In essence, it is implied that the configurations of the weights co-depend and can not be trivially re-initialized and retrained.

From these previous findings, Frankle and Carbin proposed the *lottery ticket hypothesis* and an accompanied conjecture, stated as follows.

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.*

Conjecture. *Stochastic gradient descent 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.*

(Frankle and Carbin, 2019)

Model	Error(%)
VGG-16	6.75
VGG-16-pruned-A	**6.60**
VGG-16-pruned-A scratch-train	6.88
ResNet-56	6.96
ResNet-56-pruned-A	**6.90**
ResNet-56-pruned-B	6.94
ResNet-56-pruned-B scratch-train	8.69
ResNet-110	6.47
ResNet-110-pruned-A	**6.45**
ResNet-110-pruned-B	6.70
ResNet-110-pruned-B scratch-train	7.06

Table 1: Excerpt from Table 1 from Li et al. (2016). Overall best results in terms of the test error for different models. Models containing "-pruned-" have been pruned using a structured pruning method for CNNs. Models containing "scratch-train" have been re-initialized and re-trained after pruning.

1.1 Research question

In this paper, we explore Frankle and Carbin's *lottery ticket hypothesis* and the accompanied conjecture. We task ourselves with investigating the following questions about the hypothesis.

1. What is the precise meaning and implication of the hypothesis?
2. Can we find experimental evidence of the hypothesis and if so under which conditions?
3. How can we explain the evidence supporting the hypothesis and with which mechanisms?

Similarly, we can ask these questions about their proposed conjecture which tries to answer our third research question.

4. What is the precise meaning and implication of the conjecture?
5. Can we find experimental evidence of the conjecture and if so under which conditions?

Before we can adequately answer these questions, we will provide some background information in section 2.

2 Background

To thoroughly answer our research questions, it is important to note that there are many different pruning methods using several completely different approaches. This large and diverse family of methods is illustrated in subsection 2.1, while subsection 2.2 explains the overall success of pruning in a broader context of feed-forward neural networks.

2.1 Pruning methods

Pruning methods can be classified as either pre-defined by the network's user or automatically derived by the network. Another division of methods can be made based on the relevance of the network's structure to the pruning algorithm. Methods that take the network's structure into account are called structured methods, while methods that only consider individual weights are called unstructured methods.

Unstructured pruning

Unstructured pruning methods find individual weights that are least relevant and can be pruned. This method was utilized by Frankle and Carbin to perform their experiments for the *lottery ticket hypthesis.* There is an important distinction between one-shot pruning and iterative pruning.

1. **One-shot pruning** This is the basic type of unstructured pruning method which prunes the network as as function of the percentage of the weights(Frankle and Carbin, 2019). It involves the following steps.

    1. Randomly initialize a neural network.
    2. Train the network.
    3. Set p% of weights from each layer (based on some metric) to 0.

    In order to find winning tickets, Frankle and Carbin propose to reset the weights of the pruned network to their original random initialization.

2. **Iterative Pruning** This is similar to the one-shot pruning method. The major difference is that the weights are trained and pruned over n rounds. In each round, $p^{\frac{1}{n}}\%$ of the weights are removed. This is the main method used in Frankle and Carbin (2019).

Pre-defined structured pruning

The methods which employ structured pruning involve removing the weights in the channels while keeping the target architecture in mind before performing the pruning. The way in which pruning is performed is set by the user and the manner is dictated by the pruning algorithm itself. These methods were employed by Liu et. al. in order to counter the unstructured pruning methods performed by Frankle and Carbin.A brief methodology for this technique of pruning is explained in Figure 2 The different predefined structured pruning methods from Liu et al.'s work are given below.

1. **L1-norm based Filter Pruning** One of the earliest pruning methods on channel pruning for convolutional neural networks. In this method, the L1-norm for the filters is considered in each layer. A pre-defined percentage of filters with smaller L1-norm is eliminated. (Li et al., 2016)

2. **ThiNet** This method is greedy as compared to the L1-norm. It prunes the channel that has the least effect on next layer's activation values. (Luo et al., 2017)

3. **Regression Based Feature Reconstruction** The pruning is performed by reducing the error during reconstruction of feature map of the next layer. The optimization problem in this method is solved with the help of LASSO Regression. (He et al., 2017)

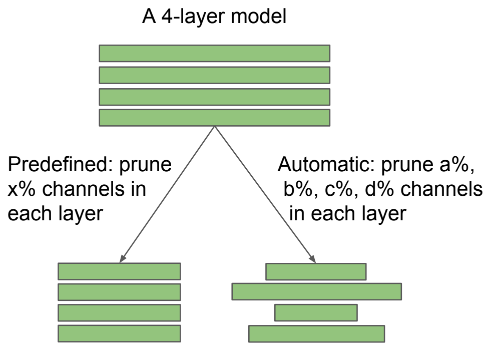
A 4-layer model
Predefined: prune x% channels in each layer
Automatic: prune a%, b%, c%, d% channels in each layer

Figure 2: A visualization for the pre-defined and automatic structured pruning method. Figure 2 in Liu et al. (2019).

Automatic Structured Pruning

Unlike the pre-defined structured pruning methods, here the target architectures for the sub-network are derived automatically. Liu et. al. argue that training these models from scratch can lead to comparable or even better performance, concluding that the architecture derived in these methods is much more important than the weights themselves. The different automatic pruning methods are given below.A brief methodology for this technique of pruning is explained in Figure 2.

1. **Network Slimming** "imposes L1-sparsity on channel-wise scaling factors from Batch Normalization layers during training, and prunes channels with lower scaling factors afterward." (Liu et al., 2019)

2. **Sparse Structure Selection** This method is a generalization of the network slimming method as it also makes use of the sparsified scaling factors to be used for pruning. It has been shown to be applicable on the residual blocks in ResNet if required. (Huang and Wang, 2017)

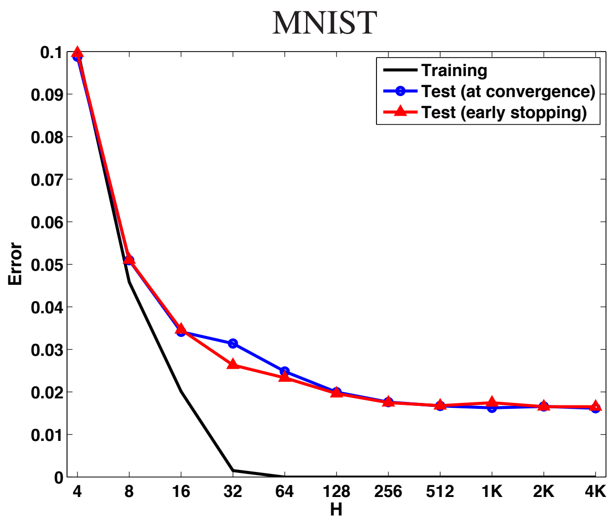

Figure 3: Relationship between the training/test error and the size of a 2-layer neural network trained by momentum stochastic gradient descent. Here, H represents the network size. From Figure 1 in Neyshabur et al. (2015).

2.2 Pruning in relation to the generalization error

Neural network pruning has been observed to improve the generalization capabilities of the model. However, it is still debated in what way pruning achieves this (Bartoldson et al., 2019). Historically, the network's size in terms of the number of free parameters has been used as a metric of the complexity of a network (LeCun et al., 1990). Since increased complexity implies an increased risk of overfitting, it has been conjectured that the shrinking of a neural network reduces this risk and improves the generalization capabilities of the network. This theory however, has not been generally proven. In fact, contradictory evidence has been found for specific network architectures which implies that larger networks generalize better rather than worse (Neyshabur et al., 2015). As can be seen in Figure 3, a larger network always improves both the training and test error for this specific experiment. It has been suggested that it is the pruning method itself from which the improved generalization arises (Bartoldson et al., 2019). This theory, as postulated by Bartoldson et al., states that the destabilizing effects of the pruning method allow the learning algorithm to leave potentially sub-optimal local minima.

3 Experiments

This section provides a broad overview of the experiments conducted by Liu et al. and Frankle and Carbin. The figures and the table in the appendix (Appendix A) are of central importance. We separately explore the gained insights in the following sections.

3.1 Experiments by Frankle and Carbin

To answer our research question, in this section we look at the findings from Frankle and Carbin (2019). In order to find the winning tickets, Frankle and Carbin make use of fully connected neural network models for the dataset MNIST as well as convolutional neural network architectures for the CIFAR10 dataset. They use optimization strategies such as stochastic gradient descent (SGD), momentum SGD and Adam also making use of dropout, weight decay and residual connections. The used pruning technique is mainly iterative unstructured pruning such that the winning tickets are sparse.

3.1.1 Fully Connected Neural Networks

The pruning technique performed for these networks is a simple layer-wise pruning heuristic. A certain percentage of weights with the lowest magnitudes in each layer is removed. Connections to output are pruned at half the rate compared to the rest of the network. Here the MNIST dataset is used for training on a LeNet (LeCun et al., 1990) 300-100 architecture. From these experiments, it is that concluded that in the case of a fully connected network such as LeNet for a relatively small dataset (such as MNIST), the initialization is significant for the performance of the winning ticket.
In the case of iterative pruning, the winning tickets are found are to learn faster than the original network. The winning tickets are pruned iteratively to various extents. It is observed that a "winning ticket" with 51.3% of the weights remaining gives a higher test accuracy than the original network. However, it does so slower than when P_m (percentage of weights remaining in the network) is 21.1% and subsequently slows when P_m is reduced. When $P_m = 3.6\%$, the winning ticket performs on par with the original network.
It is to be noted that winning tickets optimize more effectively but do not generalize better in the case of an early stopping criterion. However at 50,000 iterations, the test accuracy seems to improve thus countering the assumption of the winning ticket not generalizing better.
When the model is randomly re-initialized, it is observed that for P_m between 51.3% and 21% the winning tickets learn at slower rate in the case of random re-initialization and lose test accuracy after a little pruning, which supports the lottery ticket hypothesis.
One-shot pruning helps to identify the winning tickets without repeated training. It is observed from the experiments that the iteratively pruned tickets learn faster and reach higher test accuracy at smaller network sizes. Frankle and Carbin emphasize the need for iterative pruning for their experiments on fully connected networks.

3.1.2 Convolutional Neural Networks

Frankle and Carbin extend the hypothesis to convolutional neural networks using data from CIFAR10 and utilize scaled down VGG networks as well as VGG and ResNet for the experiments. The basic experiment is the same as in the case of LeNet. As the network is pruned, it tends to learn faster and the test accuracy is improved. Iterative pruning and random re-initialization for different layers of the ConvNet is performed and the results can be found in Figure 5 in (Frankle and Carbin, 2019, Page 10). The winning tickets have higher accuracy as they are iteratively pruned. However, it is found that the gap between the test and training error is smaller for winning tickets implying that they generalize better.
As in the case of LeNet, in ConvNets, the test accuracy is worse with random reinitialization of the winning tickets. However, the test accuracy at early stopping remains steady may even improve for some layers in the network. This results in the structure of the network may lead to higher accuracy at moderate levels of pruning. It is noted by Frankle and Carbin that using dropout helps increasing the test accuracy of the network. Here, dropout refers to randomly sampling a subnetwork for each training iteration. (Srivastava et al., 2014)
When training on larger VGG and ResNet networks, Frankle and Carbin found the architecture to be sensitive to the learning rate, and warmup was required

to find the tickets at a higher learning rate. However, in the case of a lower learning rate, the winning ticket seems to learn faster than the original network in the beginning. This however soon slows down due to the lower initial learning rate. On the other hand, they perform faster when randomly reinitialized. This can be improved by the introduction of the warmup strategy which in turn helps finding winning tickets at a higher learning rate.

3.2 Experiments by Liu et al.

In their paper, Liu et al. make certain arguments against the findings of Frankle and Carbin. Liu. et. al. compare the performance of the pruned and the unpruned network with the help of different methods of pruning as discussed in the section 2.1. They employ different techniques such as *Scratch-B* and *Scratch-E* which involve training the small pruned models for the same number of epochs (Scratch -E) or training over the same amount of computation budget (Scratch-B). If the pruned model helps saving 2 times the number of FLOPs, the number of epochs are doubled. Scratch-B is mostly found to be comparable while Scratch-E is used to compare its accuracy with the fine-tuned model.
In the case of automatic pruning methods, the architectures are detected automatically as a means of the feedback from layer by layer pruning and helps find efficient architectures in some cases. Note that since Liu et al. use automatic structured pruning methods, their approach differs fundamentally from Frankle and Carbin's approach.
Although they do not argue over the validity of the Lottery Ticket hypothesis entirely, Liu et al. mentions that it in unnecessary to reuse the original weights of the network and random re-initialization of the weights of the winning tickets is enough to perform on par with the original unpruned network. The main difference according to Liu et al. is the result when comparing unstructured pruning on CIFAR. For structured pruning, using either higher or lower learning rates, random re-initialization does not outperform the original network.
In order to prove this, Liu et al. compare the models with Frankle and Carbin's approach as well as randomly re-initializing weights using learning rates of 0.1 and 0.01, also utilizing a step-wise delay schedule and momentum SGD. These are then used on the CIFAR and ImageNet datasets, the latter of which is a larger dataset. The pruning methods used here are iterative pruning, one-shot pruning as used by Frankle and Carbin and comparing them with L1-norm based filter pruning. The values for the experiment are shown in Table 8 in Liu et al. (2019, page 11).
From figures 4 and 5 (Appendix A) we observe that in the case of unstructured pruning, the winning ticket method of assigning the original weights to the subnetwork works better in comparison to the randomly reinitialized subnetwork if the learning rate is small. In the case of structured pruning, the winning ticket performs on par with the randomly reinitialized subnetwork.
Liu et al. argue that the reason why the *winning ticket* approach is helpful at small learning rate may be that the weights of the final trained model are not too far from original initialization due to the small parameter step-size.

4 Discussion and conclusion

In the case of Frankle and Carbin it is observed that when unstructured pruning is used, the winning tickets work better when they are trained with the original training weights of the original network than when they are randomly re-initialized. For higher learning rates, the warm-up technique was necessary to get better results than with random re-initialization. In case of the lower learning rates, the pruned network learnt faster initially but slowed down over time due to the small learning step. Liu et al. performed experiments to gain more evidence and conclude that the accuracy achieved when the subnetwork is randomly reinitialized it actually performs similar to that of the winning tickets. Thus it is implied that the structure of the network is important rather than the weights. They conjecture that this may occur because in Frankle and Carbin's experiments, the data set was small and the target weights were closer to the values in the original network. Liu et al. argue that the weights may introduce a large bias over the selection of the winning ticket.
In conclusion, we can substantiate Frankle and Carbin's hypothesis with the evidence found by Frankle and Carbin themselves as well as by Liu et al.. However, it is a challenging task to find the lottery tickets as described. For different neural network models of different sizes and complexities, more sophisticated learning methods are required. It is unresolved whether it is always possible to find winning tickets in any dense neural network, and if they exist as is claimed by Frankle and Carbin. We have only seen evidence of applicability in relatively small network and when using a relatively low learning rate or a warmup strategy. More research into this topic is necessary and important.

References

Brian R. Bartoldson, Ari S. Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *CoRR*, abs/1906.03728, 2019. URL `http://arxiv.org/abs/1906.03728`.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf`.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *CoRR*, abs/1707.06168, 2017. URL `http://arxiv.org/abs/1707.06168`.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *CoRR*, abs/1707.01213, 2017. URL `http://arxiv.org/abs/1707.01213`.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL `http://arxiv.org/abs/1608.08710`.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJlnB3C5Ym`.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *CoRR*, abs/1707.06342, 2017. URL `http://arxiv.org/abs/1707.06342`.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6614`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15: 1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL `http://arxiv.org/abs/1411.1792`.

A Appendix – Additional experimental results

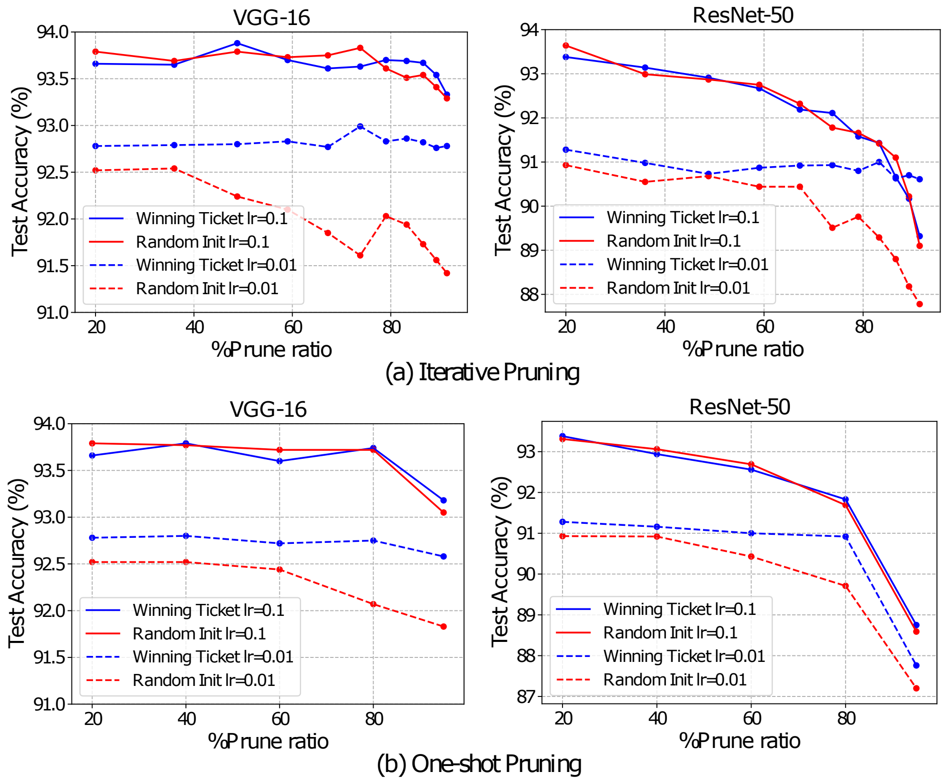

Figure 4: From Liu et al. (2019, page 11, Figure 7): "Comparisons with the Lottery Ticket Hypothesis (Frankle and Carbin, 2019) for iterative/one-shot unstructured pruning (Han et al., 2015) with two initial learning rates 0.1 and 0.01, on CIFAR-10 dataset."

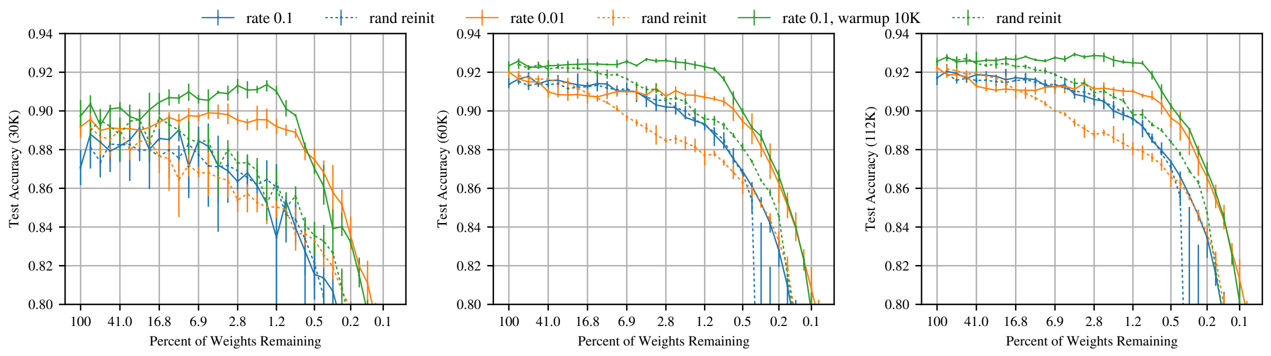

Figure 5: From Frankle and Carbin (2019, page 7, Figure 7): "Test accuracy (at 30K, 60K, and 112K iterations) of VGG-19 when iteratively pruned."

Dataset	Model	Unpruned	Prune ratio	Winning Ticket	Random Init
CIFAR-10	VGG-16	93.76 (±0.20)	20%	93.66 (±0.20)	**93.79** (±0.11)
			40%	**93.79** (±0.12)	93.77 (±0.10)
			60%	93.60 (±0.13)	**93.72** (±0.11)
			80%	**93.74** (±0.15)	93.72 (±0.16)
			95%	**93.18** (±0.12)	93.05 (±0.21)
	ResNet-50	93.48 (±0.20)	20%	**93.38** (±0.18)	93.31 (±0.24)
			40%	92.94 (±0.12)	**93.06** (±0.22)
			60%	92.56 (±0.20)	**92.69** (±0.22)
			80%	**91.83** (±0.20)	91.69 (±0.22)
			95%	**88.75** (±0.18)	88.59 (±0.09)

(a) One-shot pruning with initial learning rate 0.1

Dataset	Model	Unpruned	Prune ratio	Winning Ticket	Random Init
CIFAR-10	VGG-16	92.69 (±0.12))	20%	**92.78** (±0.11)	92.52 (±0.15)
			40%	**92.80** (±0.18)	92.52 (±0.15)
			60%	**92.72** (±0.16)	92.44 (±0.19)
			80%	**92.75** (±0.07)	92.07 (±0.25)
			95%	**92.58** (±0.25)	91.83 (±0.11)
	ResNet-50	91.06 (±0.28)	20%	**91.28** (±0.15)	90.93 (±0.34)
			40%	**91.16** (±0.07)	90.92 (±0.10)
			60%	**91.00** (±0.15)	90.43 (±0.16)
			80%	**90.92** (±0.08)	89.71 (±0.18)
			95%	**87.76** (±0.19)	87.20 (±0.17)

(b) One-shot pruning with initial learning rate 0.01

Table 2: Table 11 from Liu et al. (2019). Test errors are shown after pruning whilst re-initializing with original weights or random weights.