The Lottery Ticket Hypothesis: Training Pruned Neural Networks

Jonathan Frankle
MIT CSAIL
jfrankle@mit.edu

Michael Carbin
MIT CSAIL
mcarbin@mit.edu

Abstract

Recent work on neural network pruning indicates that, at training time, neural networks need to be significantly larger in size than is necessary to represent the eventual functions that they learn. This paper articulates a new hypothesis to explain this phenomenon. This conjecture, which we term the *lottery ticket hypothesis*, proposes that successful training depends on lucky random initialization of a smaller subcomponent of the network. Larger networks have more of these "lottery tickets," meaning they are more likely to luck out with a subcomponent initialized in a configuration amenable to successful optimization.

This paper conducts a series of experiments with XOR and MNIST that support the lottery ticket hypothesis. In particular, we identify these fortuitously-initialized subcomponents by pruning low-magnitude weights from trained networks. We then demonstrate that these subcomponents can be successfully retrained in isolation so long as the subnetworks are given the same initializations as they had at the beginning of the training process. Initialized as such, these small networks reliably converge successfully, often faster than the original network at the same level of accuracy. However, when these subcomponents are randomly reinitialized or rearranged, they perform worse than the original network. In other words, large networks that train successfully contain small subnetworks with initializations conducive to optimization.

The lottery ticket hypothesis and its connection to pruning are a step toward developing architectures, initializations, and training strategies that make it possible to solve the same problems with much smaller networks.

1 Introduction

Recent work on neural network pruning (e.g., [4, 3, 2]) indicates that neural networks can be dramatically simplified once trained. After training is complete, upwards of 90% of weights can be pruned without reducing accuracy. If a network can be so pruned, then the function that it learned could have been represented by a far smaller network than that used during training. However, researchers believe[1] that these smaller networks cannot be trained as readily as their larger counterparts, in spite of the fact that they are demonstrably capable of representing the desired functions. In this paper, we contend that it is indeed possible to train these smaller networks directly. In fact, these small, trainable networks are embedded within the larger models that we typically train.

This paper articulates a possible explanation for the disconnect between a neural network's representation capacity and its trainability. This conjecture, which we term the *lottery ticket hypothesis*,

[1] [4] mentions that "CNNs contain fragile co-adapted features" and that "gradient descent is able to find a good solution when the network is initially trained, but not after re-initializing some layers and retraining them...When we retrain the pruned layers, we should **keep the surviving parameters instead of re-initializing them**."

states that training succeeds for a given network if one of its subnetworks has been randomly initialized such that it could be trained in isolation—independent of the rest of the network—to high accuracy in at most the number of iterations necessary to train the original network. We refer to these fortuitously-initialized networks as *winning tickets*.

Subnetworks and winning tickets. From the perspective of the lottery ticket hypothesis, a network's initialization procedure can be thought of as drawing many samples from a distribution over initialized subnetworks. Ideally, the procedure manages to draw a subnetwork with the right architecture and weight initializations for optimization to succeed (a winning ticket). If a network of size m (where size is measured in units or weights) is being trained to solve a problem but a network of size n (where $n \leq m$) is sufficient to represent the function to be learned, then the lottery ticket hypothesis views the original network as containing $\binom{m}{n}$ overlapping subnetworks. If the larger network is able to train successfully, it is because one of these subnetworks lucked into an initialization amenable to optimization.

Metaphorically, training a network larger than is necessary to represent the function to be learned is like buying many lottery tickets. Larger networks have combinatorially more subcomponents that could facilitate successful training (i.e., more lottery tickets). The initialization strategy determines which of these subcomponents are well-situated for optimization to succeed (i.e., which tickets are winners). If a subcomponent is initialized favorably (i.e., the network picked a winning ticket), training succeeds.

Identifying winning tickets. In this paper, we demonstrate that it is possible to automatically identify winning tickets by making a small but critical modification to the experiment of Han et al. [4]. We prune a trained neural network's smallest weights (as measured by their magnitudes after training) in the same manner as Han et al.; the set of connections that survives this pruning process is the architecture of a winning ticket as anticipated by the lottery ticket hypothesis. Unique to our work, the winning ticket's weights are the values to which these connections were initialized *before training began*. Where Han et al. aimed to compress networks during the training process, our goal is to find small networks that can be trained independently from the start. We show that a winning ticket extracted in this fashion, when initialized with the original weights from before training, can be trained successfully in isolation at least as fast as (and typically faster than) the full network.

Methodology. To empirically assess the lottery ticket hypothesis, we use the following procedure to extract winning tickets from fully-connected networks of a variety of sizes for MNIST and, as an illustrative example, from small networks for XOR. This procedure is identical to Han et al.'s pruning process with the addition of the crucial last step: resetting the weights to their original values from before training.

1. Randomly initialize a neural network.
2. Train the network until it converges.
3. Prune a fraction of the network.
4. To extract the winning ticket, reset the weights of the remaining portion of the network to their values from (1) (i.e., the initializations they received before training began).

If successful training really does rely on fortuitous initialization of a subcomponent of the network and pruning really does reveal this winning ticket, then the lottery ticket hypothesis predicts that the pruned network—when reset to the original initializations from before training—will train successfully at sizes too small for a randomly-initialized or a randomly-configured network to do so.

Research questions. To test the lottery ticket hypothesis, we evaluate the following research questions:

How effectively do winning tickets train in comparison to the original network and to randomly sampled networks of similar size? For a variety of network configurations and perturbations of winning tickets, we measure both convergence times and test accuracy once the network has converged.

How big are winning tickets relative to the size of the original network? By training on networks of various sizes, we explore whether the size of a winning ticket remains constant for a particular learning problem or grows in proportion to the size of the larger network from which it is derived.

10 Units		8 Units		6 Units		4 Units		2 Units	
DB	ZL	DB	ZL	DB	ZL	DB	ZL	DB	ZL
98.5	92.9	96.8	87.5	92.5	76.8	78.3	55.3	49.1	17.6

Figure 1: Success rates of 1000 random XOR networks, each with the specified number of hidden units. DB = percent of trials that found the correct decision boundary. ZL = percent of trials that reached zero loss.

How sensitive are our results to particular pruning strategies? We test two broad classes of strategies: pruning hidden units with the lowest-magnitude incoming and/or outgoing weights (for XOR) and individually pruning the lowest-magnitude weights (for MNIST). We also study whether networks can be pruned in a single step or whether they must repeatedly be pruned, retrained, and reset in an iterative process.

Results. Our experimental results support the lottery ticket hypothesis.

XOR. We trained a simple network with one hidden layer to learn XOR. The minimal architecture capable of representing XOR, a hidden layer with two randomly-initialized units, reached zero loss 18% of the time. In contrast, when a network with ten hidden units that reached zero loss was iteratively pruned down to a two-unit winning ticket, the winning ticket reached zero loss 80% of the time when trained with its original initializations.

MNIST. Up to a certain point, winning tickets derived by pruning converged faster than and at least as accurately as the original network; after this point, convergence times and accuracy gradually and then rapidly dropped off. In a single step, we could prune networks by 75% while still finding winning tickets that, on average, converged 38% faster than the original network and matched its accuracy. Pruning iteratively by 20% at a time, winning tickets 91% smaller than the original network converged on average 39% faster. Networks iteratively pruned by up to 98% on average still converged as fast as the original network while maintaining accuracy. When winning tickets were randomly reinitialized or their weights were randomly rearranged, convergence times increased and accuracy decreased as compared to the original network. Depending on the metric of winning ticket size, winning tickets grew either gradually or marginally with network size.

Contributions and implications.

- We propose the *lottery ticket hypothesis* as a new perspective on neural network training.
- We further posit that pruning uncovers the winning tickets that the lottery ticket hypothesis predicts, leading to an algorithm for extracting winning tickets from trained networks.
- We apply this algorithm to empirically evaluate these conjectures on small networks. The evidence we find supports both the lottery ticket hypothesis and our contention that pruning can extract winning tickets.

Although this paper focuses mainly on measurement, it has important implications for our understanding of training. The increased representation power of large networks is not necessarily required for gradient descent to learn functions with small representations. Lurking within these large networks are small, fortuitously-initialized winning tickets that are both more efficient to train (as a product of their size) and faster to converge (as a product of their initialization). By examining the initalizations and architectures of successful winning tickets, we might find new ways of designing networks that are smaller but equally-capable (if not superior).

2 Learning the XOR Function

The XOR function is among the simplest examples that distinguish neural networks from linear classifiers. Before presenting our results for MNIST, we summarize the lottery ticket hypothesis as it applies to this simple computation. The XOR function has four data points: the coordinates $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. The first and last points should be placed in class 0 and the middle two points in class 1. Geometrically, this problem requires a nonlinear decision boundary. In this experiment, we consider the family of fully connected networks for XOR with two input units, one hidden layer (ReLU activation), and one output unit (sigmoid activation).

Pruning Strategy	10 Units		4 Units (Pruned)		2 Units (Pruned)	
	DB	ZL	DB	ZL	DB	ZL
One-shot Product	99.2	93.3	98.0	90.3	82.4	65.3
Input Magnitude	98.9	93.5	97.9	92.2	83.8	76.5
Output Magnitude	99.0	93.3	96.9	85.9	78.6	56.1
Product	98.5	92.9	97.6	90.3	91.5	79.4

Figure 2: Success rates of different pruning strategies on 1000 trials each. DB and ZL defined as in Figure 1. The pruned columns include only those runs for which both the original ten-unit network and the pruned winning ticket found the right decision boundary or reached zero loss. The first row of the table was obtained by pruning in one shot; all subsequent rows involved pruning iteratively

Although a network of this form with two hidden units is sufficient to perfectly represent the XOR function,[2] the probability that a standard training approach—one that randomly initializes the network's weights and then applies gradient descent—correctly learns XOR for a network with two hidden units is low relative to that for a larger network.

Figure 1 contains the overall success rates (percent of networks that found the right decision boundary or reached zero loss). In 1000 training runs, a network with two hidden units learned a correct decision boundary in only 49.1% of trials. Cross-entropy loss reached 0 (meaning the network learned to output a hard 0 or 1) in only 17.6% of trials. Meanwhile, an otherwise identical network outfitted with ten hidden units learned the decision boundary in 98.5% of trials and reached 0 loss in 92.9% of trials. Figure 1 charts the loss for these and other hidden layer sizes.[3]

To put the central question of this paper in the concrete terms of the XOR problem, why do we need to start with a neural network with ten hidden units to ensure that training succeeds when a much smaller neural network with two hidden units can represent the XOR function perfectly? We propose the lottery ticket hypothesis as an explanation for this phenomenon.

The Lottery Ticket Hypothesis. *Training succeeds for a given network if one of its subnetworks (a "winning ticket") has been randomly initialized such that it can be trained in isolation to high accuracy in at most the number of iterations necessary to train the original network.*

According to the lottery ticket hypothesis, successful networks with a large number of parameters (e.g., the XOR network with ten hidden units) should contain winning tickets comprising a small number of fortuitously-initialized weights on which training will still succeed.

2.1 Methodology

To test the lottery ticket hypothesis with the XOR function, we instantiated the experiment from Part 1 with the following details:

1. Randomly initialize a network with ten hidden units.
2. Train for 10,000 iterations on the entire training set.
3. Prune a certain number of hidden units according to a particular pruning heuristic.
4. To extract the winning ticket, reset the pruned network to the original initializations.

The first three steps extract the architecture of the winning ticket; the crucial final step extracts the corresponding initializations. We ran this experiment with two different classes of pruning strategies. *One-shot* pruning involves pruning the network in a single pass. For example, one-shot pruning a network by 80% would involve removing 80% of its units after it has been trained. In contrast, *iterative pruning* involves repeating steps (2) through (4) several times, removing a small portion of

[2]Example satisfying weights for the first layer: $\begin{bmatrix} n & -n \\ -n & n \end{bmatrix}$. Satisfying weights for the output unit: $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Satisfying bias for the output unit: $-n/2$. $n \geq 1$. As n grows, the output approaches a hard 1 or 0.

[3]Weights were sampled from a normal distribution centered at 0 with standard deviation 0.1; all values more than two standard deviations from the mean were discarded and resampled. Biases were initialized to 0. The network was trained for 10,000 iterations.

the units (in our case, two units) on each iteration. We find iterative pruning to be more effective for extracting smaller winning tickets; Han et al. [4] found the same for compressing large networks while maintaining accuracy.

We consider three different heuristics for determining which hidden units should be pruned:

- *Input magnitude:* remove the hidden unit with the smallest average input weight magnitudes.
- *Output magnitude:* remove the hidden unit with the smallest output weight magnitude.
- *Magnitude product:* remove the hidden unit with the smallest product of the magnitude of its output weight and the sum of the magnitudes of its input weights.

The magnitude product heuristic achieved the best results, so we use it unless otherwise stated.

2.2 Results

One-shot pruning. We generated 1000 networks with ten hidden units and pruned them down to both four and two hidden units using the magnitude product heuristic. The results of doing so appear in the first row of Figure 2. The winning tickets with two hidden units found the correct decision boundary 82.4% of the time (up from 49.1% for randomly-initialized networks with two hidden units) and reached zero loss 65.3% of the time (up from 17.6% of the time for a random network).

Iterative pruning. We conducted the iterative version of the pruning experiment 1000 times, starting with networks containing ten hidden units that were eventually pruned down (in two unit increments) to networks containing a candidate winning ticket of just two hidden units. Of the 93.5% of ten hidden unit networks that reached zero loss, 84.9% [4] had a two hidden unit winning ticket that also reached zero loss (as compared to 17.6% of randomly-intialized two hidden unit networks). Likewise, of the 98.9% of ten hidden unit networks that found the correct decision boundary, 92.8% had a two-unit winning ticket that did the same (as compared to 49.1% of randomly-initialized two hidden unit networks). The four hidden unit winning tickets almost identically mirror the performance of the original ten hidden unit network. They found the correct decision boundary and reached zero loss respectively in 99% and 97% of cases where the ten hidden unit network did so. Both of these pruned trials appear in Figure 2 (under the *Magnitude Product* row).

These experiments indicate that, although iterative pruning is more computationally demanding than one-shot pruning, it finds winning tickets at a higher rate than one-shot pruning. More importantly, they also confirm that networks with ten hidden units can be pruned down to winning tickets of two hidden units that, when initialized to the same values as they were in the original network, succeed in training far more frequently than a randomly initialized network with two hidden units. The winning tickets with four hidden units succeed nearly as frequently as the ten unit networks from which they derive. Both of these results support the lottery ticket hypothesis—that large networks contain smaller, fortuitously-initialized winning tickets amenable to successful optimization.

In addition to the magnitude-product pruning heuristic, we also tested the input magnitude and output magnitude heuristics. The results of doing so appear in Figure 2. The magnitude product heuristic outperformed both. We posit that this success is due to the fact that, in the XOR case when all input values are either 0 or 1, the product of input and output weight magnitudes should mimic the activation of the unit (and therefore with its influence on the output).

3 MNIST (One-shot Pruning)

In this section and those that follow, we explore the lottery ticket hypothesis as applied to the MNIST dataset. Here, we analyze the behavior of one-shot pruning; in the following section, we show the additional power that iterative pruning offers.

[4]These numbers are derived from the last row of Figure 2. 93.5% of networks with ten hidden units reached zero loss. 79.4% of networks started with ten units, reached zero loss, and were pruned to into two-unit networks that also reached zero loss. 79.4% of 93.5% is 84.9%.

3.1 Methodology

We trained and pruned a network with two fully-connected layers. We used the LeNet-300-100 architecture [6], which has 784 input units (corresponding to the pixels of the 28x28 images in MNIST), a fully-connected hidden layer with 300 units, a fully-connected hidden layer with 100 units, and ten fully-connected output units (one for each class). The hidden units have ReLU activation functions, and the output units have softmax activation functions. By default, biases were initialized to 0 and weights were randomly sampled from a normal distribution with mean 0 and standard deviation 0.1 (values more than two standard deviations from the mean were discarded and resampled). Networks were optimized using stochastic gradient descent with a learning rate of 0.05.

This section follows the experimental template from Section 1:

1. Randomly initialize the network.
2. Train for 50,000 iterations on 100-example mini-batches from the training data
3. Prune a certain percentage of the weights from within each hidden layer, removing those with the lowest magnitudes.
4. To extract the winning ticket, reset the values of the weights of the pruned network to their original initializations from before training.

The pruning strategy we follow for MNIST removes individual weights rather than entire units. In preliminary experiments, we found this strategy to be more effective (Srinivas and Babu explore pruning by unit in [12]). We use the simplest weight-by-weight pruning heuristic possible: remove those weights with the lowest magnitudes within each hidden layer (just as in [4]). Weights connecting to the output layer are pruned by half of the percentage at which the rest of the network is pruned to avoid severing connectivity to the output units.

3.2 Results

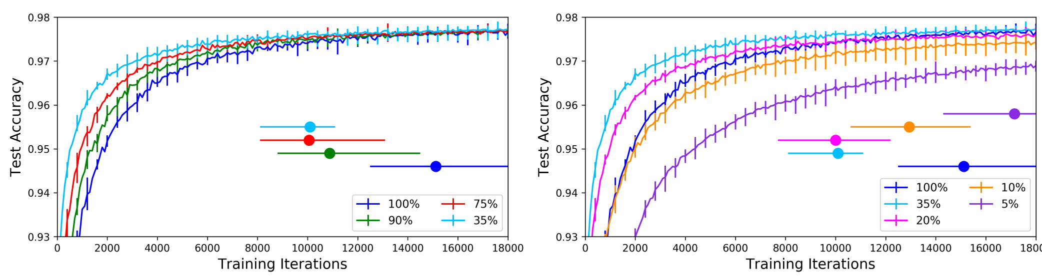

Figure 3: The test set accuracy on MNIST as training proceeds. These charts are zoomed into the very highest levels of accuracy. Each curve shows the average progression of five trials of training at the specified pruning level. Percents are the percent of the weights in each layer that remain after pruning. The error bars show the minimum and maximum values of any one of those five trials. Dots signify the moment when the corresponding colored line has converged, with error bars showing the earliest and latest convergence times amongst the five trials.

Pruning's most substantial impact was on convergence times. When pruned to between 75% and 25% of the size of the original network, the winning tickets converged on average at least 25% faster while accuracy remained on average within 0.15% of the original network's accuracy. The winning ticket that was pruned to 25% of the original size converged on average 38% faster than the original network. Further pruning caused convergence times to slowly rise and accuracy to drop.

Figure 3 shows the test set accuracy and convergence behavior of winning tickets pruned to different levels as they were trained.[5] Each curve is the average of five different runs starting from distinct,

[5]We define convergence as the moment at which the 1000-iteration moving average of test accuracy changed by less than 0.002 for 1000 consecutive iterations. We measured test accuracy every 100 iterations. According to this definition of convergence, the one-shot-pruned winning tickets improved their test accuracy by an average of 0.0019 (standard deviation 0.00085) after convergence. We acknowledge that determining convergence times is an imprecise art, but this metric seems to adequately characterize the behavior of convergence for our purposes.

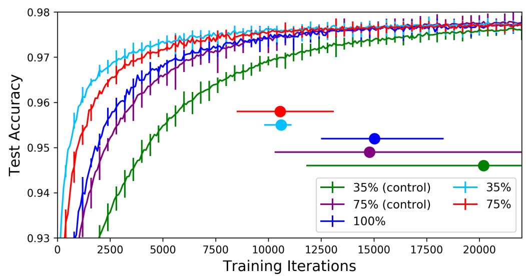

Figure 4: The test set accuracy on MNIST as training proceeds for winning tickets of various sizes and for winning tickets whose weights have been randomly reinitialized (control experiment 1).

randomly initialized networks; error bars indicate the minimum and maximum value that any run took on at each point in the training process. The dots indicate the average convergence times for the curve in the corresponding color; error bars indicate the minimum and maximum convergence times.

The left graph in Figure 3 shows that, for the first few pruning levels, convergence times decrease and accuracy increases. A winning ticket comprising 90% of the weights from the original network converges slightly faster than the original network but slower than a winning ticket with 75% of the original weights. This pattern continues until the network is pruned to about 55% of the original size, after which (as the right graph in Figure 3 shows) convergence times flatten and then, after about 35%, increase. When the winning ticket is between 10% and 15% of the original size of the network, it returns to the performance of the unpruned network.

In the terms of the lottery ticket hypothesis, we attribute improving convergence times to the removal of unnecessary, noisy parts of the network as pruning hones in on the winning ticket. Convergence times reach a tipping point as pruning begins to remove weights that are essential to the winning ticket, after which convergence times increase and accuracy decreases.

The lottery ticket hypothesis also predicts that this behavior is largely attributable to a confluence of initialization and architecture. To test this conjecture, we ran two control experiments: (1) retain the winning ticket's architecture but randomize its weights and (2) retain the winning ticket's weights but randomize its architecture.

Control experiment 1. This experiment evaluates the extent to which initialization is a necessary component of a winning ticket. Figure 4 shows this experiment. The curves for the original network and winning tickets that are 75% and 35% of the original network's size are the same as in Figure 3. Two curves have been added for the control experiments. Each control experiment entailed training a network that used a winning ticket's architecture but randomly reinitialized its weights from the original initialization distribution ($\mathcal{N}(0, 0.1)$). We trained three control experiments for each winning ticket, so the control curves are the average of 15 experiments. Unlike the winning tickets, the control experiments converged on average more slowly than the original network, simultaneously achieving lower levels of accuracy. These differences were substantial: the average 35% and 25% winning tickets converged 1.91 and 2.28 times as fast as the corresponding average controls.

The error bars on convergence times reflect that the control trials exhibited a much wider variance in behavior. For example, the earliest-converging of the 35% control trials converged faster than the average unpruned network; however, the average 35% control trial convergence time converged 27% slower than the average original network.

This experiment further supports the lottery ticket hypothesis' emphasis on fortuitous initialization. Using the same pruned architecture, the original initialization not only withstood but benefited from pruning, while performance of the reinitialized network immediately suffered and steadily diminished as the network was further pruned. This outcome mirrors, on a larger scale, the result of the XOR experiment, in which networks with many hidden units could be pruned to smaller winning tickets that found the right decision boundary at a much higher rate than randomly-initialized small networks.

Figure 5 provides a broader perspective on these patterns across all of the levels to which we pruned. The left graph shows the convergence time in relation to the percentage of the network remaining after pruning. The blue line is the average of the five winning ticket trials at each level. The convergence

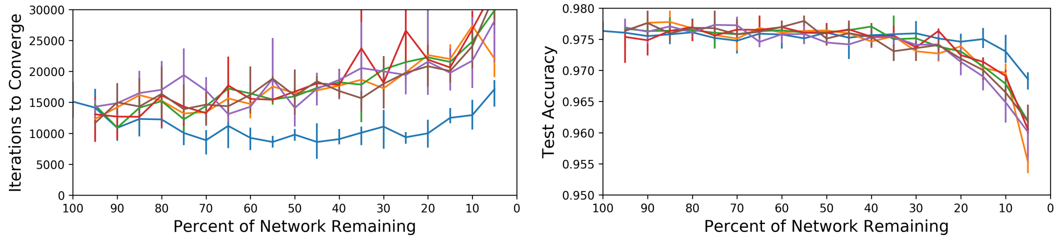

Figure 5: The convergence times (left) and accuracies (right) when running the MNIST pruning experiment to various degrees of pruning. The blue line is the average of five trials with different starting initializations that prune and reuse the original initialization. Each of the multicolored lines represents three randomly reinitialized control trials (one for each trial with the original initialization). The error bars are the minimum and maximum value any trial takes on at each interval.

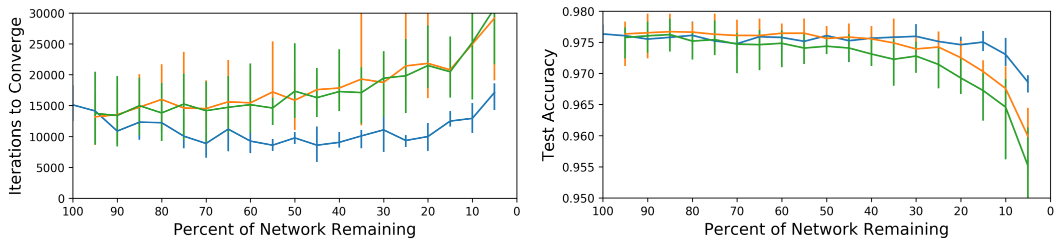

Figure 6: The convergence times and accuracy for the five winning tickets at each level of pruning (blue line), the 15 trials where the winning ticket weights were reinitialized (orange line), and the 15 trials where the winning ticket weights were maintained but shuffled within each layer (green line).

time initially decreases before leveling off between 55% and 35% and then slowly climbing again. In contrast, the multicolored lines, which represent the groups of control trials for each winning ticket, steadly require longer to converge as more of the network is pruned. In the control experiment, the error bars are much larger, suggesting wider variation in convergence times compared to the consistent convergence times of the winning tickets.

The right graph in Figure 5 provides important context: how accurate are the networks at the moment they converge? The average trial that used the original initialization (the blue line) maintans accuracy within 0.15% of the original network when pruned down to 15%, after which accuracy drops off. In contrast, the accuracy of the average control trial drops below this level when the network has been pruned by about 30%, falling precipitously when pruned by 15% (0.6% below the original network's accuracy).

This experiment supports the lottery ticket hypothesis' prediction that fortuitous initialization is a necessary ingredient to make a winning ticket. The winning ticket's structure alone is insufficient to explain its success.

Control experiment 2. This experiment evaluates the extent to which architecture is a necessary component of a winning ticket. For each winning ticket at each level of pruning, we randomly shuffled the locations of the weights in each hidden layer while maintaining their original initializations. The results of doing so appear in Figure 6. Just as in Figure 5, the blue line traces winning tickets pruned to various sizes. The orange line is the average of all 15 of the trials from control experiment 1 (reinitializing the winning tickets). The green line is the average of all 15 of the trials from control experiment 2 (shuffling the winning tickets without reinitializing). The convergence times for the two control experiments are similar: they start increasing immediately and increase more rapidly as the network gets smaller. The accuracy of control experiment 2 drops off slightly earlier than control experiment 1, which itself dropped off before the winning ticket did.

This experiment supports the lottery ticket hypothesis' prediction that winning tickets emerge from a combination of initialization and structure. Neither initialization (control experiment 1) nor structure (control experiment 2) alone is sufficient to explain the better performance of the winning tickets.

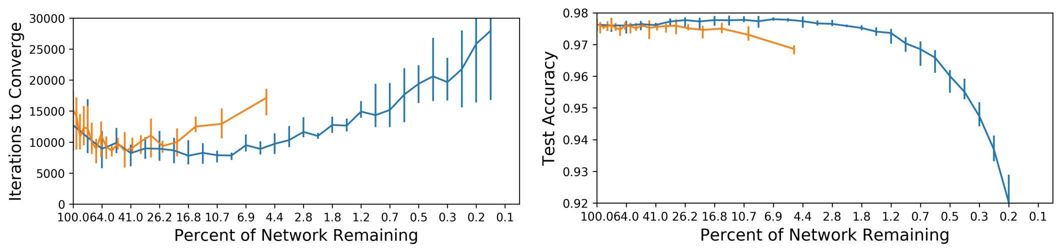

Figure 7: The convergence times and accuracy of winning tickets extracted from fully-connected networks for MNIST using one-shot pruning (orange) and iterative pruning (blue). Note that the x-axis is logarithmic.

Summary. The first notable result from this set of experiments is that, even when pruned to sizes much smaller than the original network, winning tickets are still able to converge at all. This supports the core prediction of the lottery ticket hypothesis: pruning reveals smaller subcomponents that were originally initialized such that they can train successfully in isolation. Not only do these networks train successfully, but they converge faster than and maintain the accuracy of the networks from which they derive. Furthermore, winning tickets emerge from a confluence of both fortuitous initalization and structure.

4 MNIST (Iterative Pruning)

In the XOR experiment in Section 2, iterative pruning [4]—repeatedly training, pruning, reinitializing, and pruning again—arrived at winning tickets that were more likely to train successfully In this section, we find that iterative pruning makes it possible to extract winning tickets from our MNIST network that are far smaller than those generated by one-shot pruning.

4.1 Methodology

We use the same experimental setup (network architecture, initialization strategy, and optimization strategy) as in Section 3. We follow a similar procedure repetitively in order to iteratively prune.

1. Randomly initialize the network.
2. Train for 50,000 iterations on 100-example mini-batches from the training data
3. Prune 20% of the weights from within each hidden layer and 10% of the weights in the output layer, removing those with the lowest magnitudes.
4. Reset the weight values of the pruned network to their initializations from before training.
5. Repeat steps (2) through (4) until the network has been pruned to the desired size. The result of the last iteration of (4) is the winning ticket.

We iteratively prune the incoming weights of the first and second layers of the network by 20% and the weights of the output layer by 10%.[6] We start with a network with two fully-connected hidden layers with 300 and 100 hidden units and prune the network until just under 1% of the original weights remained.

Comparison to one-shot pruning. Figure 7 shows the difference in convergence times and accuracy between one-shot pruning (orange) and iterative pruning (blue). (Note that the x-axis is logarithmic in Figure 7 and in most figures in this section.)

The average iteratively pruned winning tickets reach initially reach lower convergence times. These convergence times flatten when the original network is pruned to between 41% (36% faster than the original network) and 8.5% (38% faster than the original network) of the original network size, as compared to between 55% (44% faster than the original network) and 40% (41% faster than the

[6]As mentioned in Section 3, we prune the output layer at a lower rate to reduce the chances of severing connectivity to any of the output units.

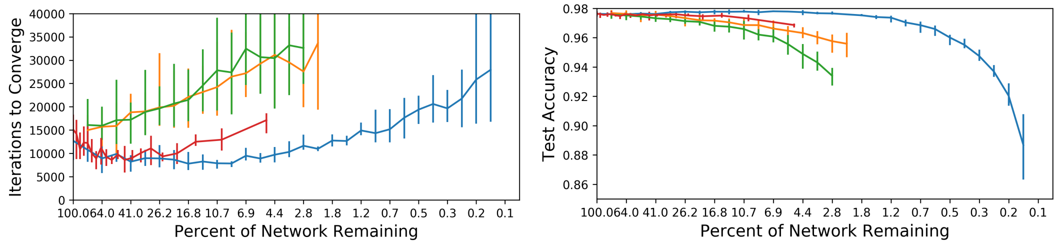

Figure 8: The convergence times and accuracy of winning tickets extracted by iteratively pruning and control experiments. The blue line is the average of five winning tickets. The orange line is control experiment 1: winning tickets that have been reinitialized. The green line is control experiment 2: winning tickets whose weights were randomly shuffled. The red line is the performance of one-shot pruning. The locations where the control trials cut off are those at which, according to our metric, they no longer converged.

original network) for one-shot pruning. The average iteratively pruned network returns to the original convergence time when pruned to 1.8% (as compared to between 5% and 10% for one-shot pruning).

Likewise, accuracy actually increases slightly for many winning tickets, returning to the original network's accuracy at a winning ticket size of 2.8% on average. In contrast, one-shot pruning begins to drop off when the winning ticket is 15% of the size of the original network.

Although iterative pruning can extract much smaller winning tickets than one-shot pruning, it is far more costly to find those winning tickets. Extracting a winning ticket with one-shot pruning requires training the original network a single time, regardless of how much the network is pruned. In contrast, iteratively pruning a network by 20% at each iteration until it is about 5% of the original network's size requires training the network 14 times. However, since our goal is to understand the behavior of winning tickets rather than to find them efficiently, iterative pruning's compelling advantage is that it is able to extract smaller winning tickets that maintain convergence and accuracy performance, placing a tighter upper-bound on the size of a network's winning ticket.

4.2 Results

In this section, we re-run the control experiments from Section 3.1. Just as before, we aim to explore the extent to which architecture and initialization are responsible for a winning ticket's ability to continue to converge at such small sizes. Figure 8 contains the average results of performing control experiment 1 (randomly reinitializing the winning ticket's weights) in orange and control experiment 2 (randomly shuffling the winning ticket's weights) in green. For comparison, the curve in red is the performance of one-shot pruning.

Control experiment 1. Just as with one-shot pruning, average convergence times for control experiment 1 begin increasing as soon as the network is pruned and continue to grow at a steady rate. The error bars in Figure 8 reflect that convergence times vary widely for pruned networks that are reinitialized. The average control trial's accuracy begins dropping off (more than 0.15% lower than the original networks accuracy) when the network is pruned to about 31%, whereas the average iteratively pruned network drops below this level when pruned to 1.5%. Just as with the one-shot experiment, this control trial indicates that initialization plays a critical role in making a winning ticket.

Control experiment 2. Average convergence times for control trial 2 increase steadily in a pattern similar to those from control trial 1. Error bars indicate that these convergence times similarly vary widely. Accuracy begins dropping off earlier, at about 51.2% and more steeply, potentially suggesting that architecture might be more important than initialization.

Summary. The control experiments for iterative pruning put the results from Section 3 in sharper relief. Iterative pruning makes it possible to extract smaller winning tickets than from one-shot pruning that reach lower convergence times than the original network while maintaining or exceeding its level of accuracy. The control experiments show that both initialization and network architecture

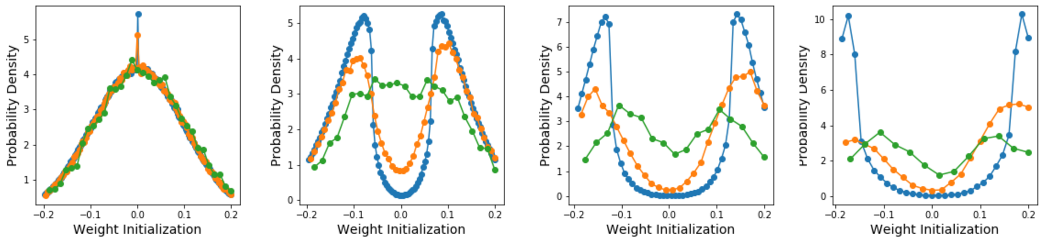

Figure 9: The distributions of the initializations of weights that survived iterative pruning across ten iterative pruning runs. The graphs contain the initializations of the network pruned to (left to rigth) 100%, 51.2%, 16.8%, and 5.50%. The blue, orange, and green lines are the distributions of initial weights for the first hidden layer, second hidden layer, and output layer, respectively.

play factor into creating a winning ticket, with control trial 2 suggesting that network architecture might be slightly more important.

The experiments on XOR and MNIST support the lottery ticket hypothesis. Embedded within larger networks are small subcomponents that are fortuitously initialized in a manner conducive to successful training. We extracted the winning ticket architectures by pruning and determined the corresponding initializations by resetting the winning ticket's connections to their original from before training. These networks not only trained successfully but, in the case of iteratively-pruning with the MNIST network, converged faster and more accurately. Meanwhile, neither architecture nor initialization alone could entirely account for this result.

We next investigate the architecture and initializations of these small winning tickets (Section 5) and the behavior of winning tickets when subjected to a wider variety of parameters (Section 6).

5 Examining Winning Tickets

In this section, we briefly explore the internal structure of winning tickets that result from iteratively-pruning our MNIST network. We have already found evidence to support the claim that winning tickets arise out of a confluence of architecture and initialization. What exactly do those architectures and initializations look like?

Initializations. Figure 9 shows the initialization distributions for winning tickets at four different levels of pruning. Note that these are the values of the winning ticket's weights from *before* training. The graph in the upper left contains the initial weights of the entire network (no pruning), which are initialized according to a normal distribution with mean 0 and standard deviation 0.1. The graph in the upper right contains the weights after iteratively pruning the network down to 51.2% of its original size. The blue, orange, and green lines are the distriutions of initial weights for the first hidden layer, second hidden layer, and output layer, respectively.

At 51.2%, the remaining weights already show the impact of pruning. The first and second hidden layer's distributions are bimodal, with two peaks mirrored opposite 0. Since these distributions plot the original initializations of the weights that survive the pruning process (i.e., the weights before training), these distributions were created by removing samples from a formerly normal distribution. These peaks on the 51.2% graph appear to be the left and right tails of the original normal distribution. The missing weights have been pruned.

Interestingly, pruning occurs *after* training and these are graphs of weights *before* training. In other words, for these distributions to emerge, small weights from before training must have remained small after training. The second hidden layer (orange) retains more of its center than the first hidden layer, indicating that those weights likely moved more during training. The output distribution (green) more closely resembles the original normal distribution, indicating that its weights probably moved significantly during training. One other contributing factor to the output distribution is that we prune it at a slower rate, meaning that the effects of pruning make take longer to appear.

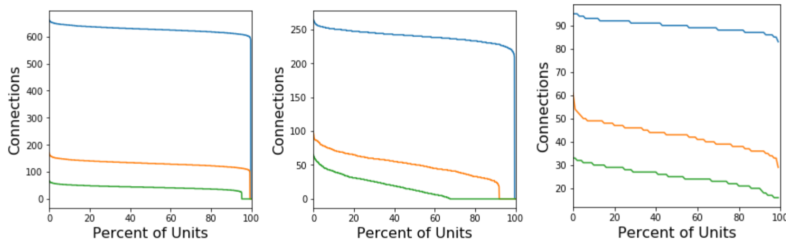

Figure 10: For each unit in the current layer, how many units in the previous layer connect to it? The left graph is for the first hidden layer. The middle graph is for the second hidden layer. The right graph is for the output layer. The blue, orange, and green lines are for the winning tickets when iteratively pruned to 80%, 16.8%, and 5.50%, respectively. Each point on the line represents a single unit; the units have been sorted in descending order by the number of connections they have. These data points were collected over ten trials.

The pattern for 51.2% pruning plays out in more extreme form at 16.8% (lower left graph) and 5.50% (lower right graph). The middles of the first and second hidden layer distributions continue to get hollowed out, and the same happens (albeit more slowly) to the output distribution. Even with these extreme-looking input distributions, the corresponding networks converged faster than the original network and retained the same accuracy.

Considering the extent to which the particular pruning strategy we pursued left its imprint on the winning tickets, it is worth considering the impact that other pruning strategies would have and, more broadly, whether the winning tickets we found are a product of the pruning strategy we pursued or whether the pruning strategy we pursued happens to exploit a deeper reality of the way neural networks behave.

Architecture. As the network is pruned, it becomes sparser. Figure 10 shows the distributions of surviving connections aggregated across ten trials[7] between a unit in layer n and units in layer $n-1$ when the network is pruned to 80% (blue), 16.8% (orange), and 5.50% (green) of its original size. The left, middle, and right graphs show the first hidden layer, second hidden layer, and output layer.

When pruned to 80%, the network remains almost fully-connected, with only slight differences between the units with the most and least connections. As the network is further pruned, units in the first hidden layer continue to have a roughly equal number of connections to units in the input layer. Even when the network is pruned to 5.50%, only a small fraction of the hidden units in the first layer have been eliminated entirely. The second hidden layer becomes less evenly connected as more weights are pruned. By the time the network is pruned to 5.50%, nearly a third of the units in the second hidden layer have been fully disconnected, and there is a steep decline from the best-connected units to the worst-connected. The output layer shows a less severe slope, likely because every output unit serves a clear function and because we prune the output layer at a slower rate.

The winning tickets are quite sparse. Even when the network is pruned by nearly 95%, only a fraction of the units have been eliminated entirely. No units maintain a large number of connections after pruning; instead, nearly all units retain a proportionally small number of connections.

6 Exploring MNIST Parameters

This section explores the sensitivity of the MNIST results to the parameters of the lottery ticket experiment. Namely, we explore the role that initialization and network size play in the properties of the winning tickets that emerge.

6.1 Initialization

Although our default network was initialized from a normal distribution with mean 0 and standard deviation 0.1, we experimented with several other standard deviations to explore the effect of larger

[7] We also removed any edges that did not have a path to an output unit.

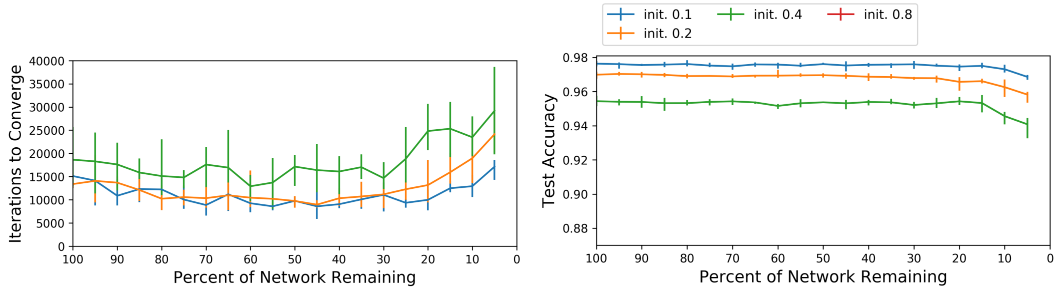

Figure 11: The convergence times and accuracy for groups of five winning tickets initialized with various standard deviations ≥ 0.1.

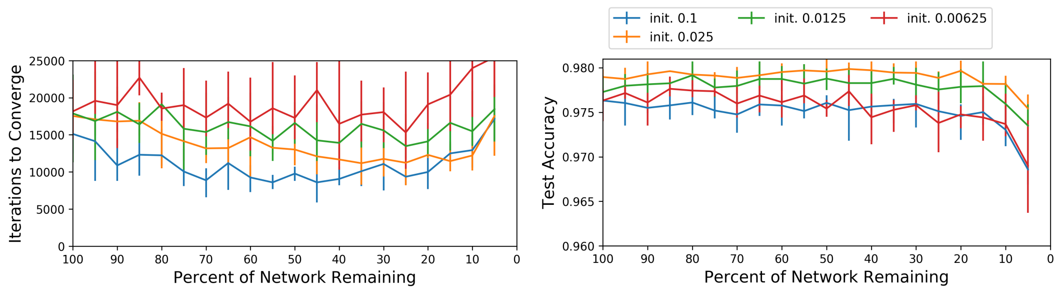

Figure 12: The convergence times and accuracy for groups of five winning tickets initialized with various standard deviations ≤ 0.1.

and smaller weights on the behavior of winning tickets. One might expect that our pruning strategy would be especially vulnerable to initializing network weights too large: by selecting for the highest-magnitude weights, it might exacerbate exploding gradients. Likewise, it might be resilient to initializing network weights too small, since it will select for the largest weights after training.

In this section, we present the results using the one-shot pruning strategy. The results for iterative pruning were similar.

Figure 11 shows the convergence times and accuracy for winning tickets of networks initialized with standard deviations larger than 0.1 (0.2, 0.4, and 0.8). As expected, convergence times increase and accuracy decreases as the standard deviations increase. We did not explore whether the extent to which this behavior resulted from exploding gradients or weaknesses in the pruning strategy.

Figure 12 contains the same information for winning tickets of networks initialized with standard deviations smaller than 0.1. A standard deviation of 0.1 produces the fastest convergence times but cedes a certain amount of accuracy in doing so. In contrast, a standard deviation of 0.025 causes winning tickets to converge more slowly but to higher-accuracy optima. This behavior suggests that there are sweet spots for both convergence times (stddev=0.1) and accuracy (stddev=0.025) and a tradeoff-space in between.

6.2 Network Size

We experimented with increasing the size from the default network (layers of 300 and 100 hidden units) in order to determine whether there is a fixed winning ticket size for a particular learning problem, or whether larger networks naturally beget larger winning tickets. We consider two possible definitions of the "size" of a network's winning ticket:

- A winning ticket is the minimal network that minimizes convergence time. Since convergence times initially decrease with pruning, this heuristic looks for the winning ticket with the lowest possible convergence time.
- A winning ticket is the minimal network that retains the accuracy of the original network. Accuracy remains relatively flat as smaller and smaller winning tickets are created; it then

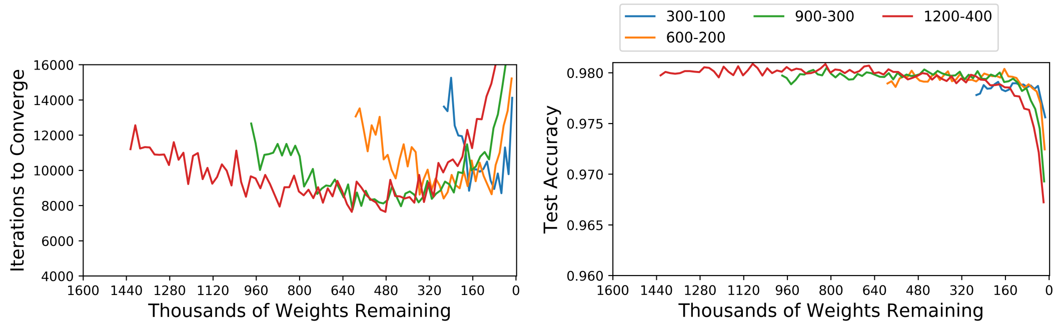

Figure 13: The convergence times and accuracy for groups of five winning tickets extracted from networks of various sizes with the one-shot pruning strategy. Error bars were elided to improve readability. The legend contains the size of the network (e.g., 300-100 means a network with hidden layers of 300 and 100 units). All networks were initialized with a standard deviation of 0.05.

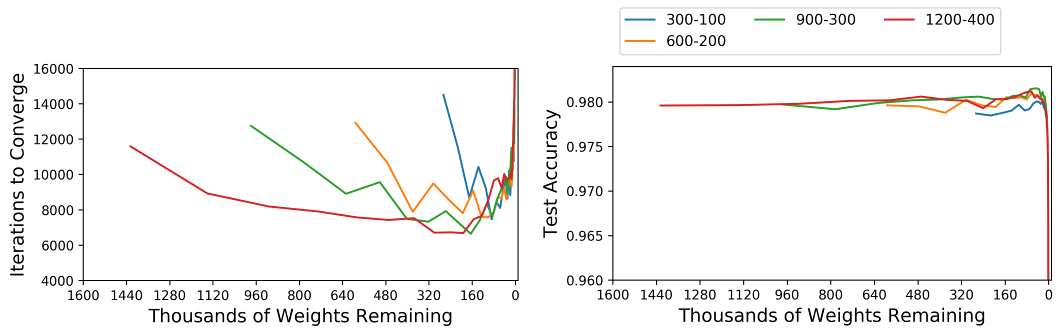

Figure 14: The convergence times and accuracy for groups of five winning tickets extracted iteratively from networks of various sizes. Error bars were elided to improve readability. All networks were initialized with a standard deviation of 0.05.

reaches a tipping point and drops off rapidly. This definition considers the winning ticket to be the last moment before this accuracy drop-off takes place.

One-shot pruning. We trained networks whose sizes were multiples of the original network size. The results of doing so and applying the one-shot pruning strategy appear in Figure 13, which plots convergence times and accuracy according to the number of weights in the winning ticket.

According to the convergence-based definition of a winning ticket, the winning ticket sizes increase gradually with the size of the network. The LeNet-300-100 architecture appears to reach this point at about 140,000 weights, where the LeNet-600-300 does so at about 200,000 weights. The same pattern holds for the larger architectures. Larger networks are capable of representing more sophisticated functions, so pruning larger networks may produce different network architectures that exploit this additional representation capacity to converge faster. Indeed, the larger the network, the lower the convergence times its winning tickets were able to achieve and the larger the size at which it reached them.

The accuracy-based definition of a winning ticket agreed. As the bottom graph of Figure 13 illustrates, the accuracy of larger networks dropped off steeply at slightly earlier times than the accuracy of smaller networks. However, these differences were quite small—on the order of tens of thousands of weights. Although winning ticket size does seem to increase with network size by this definition, the changes were very slight and winning ticket sizes close to uniform.

Iterative pruning. As Figure 14 reflects, the convergence and accuracy trends for iteratively pruning larger networks remains the same as in the one-shot case. Larger networks reach their minimum convergence times at gradually larger sizes, but accuracy plummets in unison. There are two key differences worth noting in the iterative case.

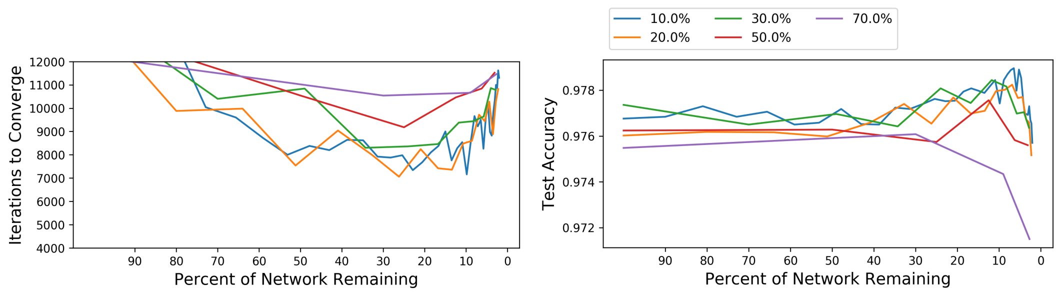

Figure 15: The convergence times and accuracy of winning tickets extracted by iteratively pruning by different rates on each iteration. Error bars have been elided for readability. Note that the x-axis is not logarithmic.

First, both minimum convergence times and accuracy dropoffs occur at much smaller network sizes than in the one-shot experiments. This result coincides with the other iterative experiments, which demonstrate that iterative pruning creates winning tickets that can be pruned to much smaller sizes before convergence times increase and accuracy diminishes. Whereas the accuracy dropoff took place when the networks had about 150,000 weights in the one-shot experiments, it occurs when the iteratively-derived winning tickets have just tens of thousands of weights.

Second, each of the accuracy graphs has a small bulge upwards just before dropping off, indicating that accuracy actually increases slightly when the winning tickets are smallest. These bulges occur at the same winning ticket size in all cases, regardless of the initial size of the network.

Summary. The analysis in this subsection leaves many open questions for future research. Although we do not undertake extensive analysis of the internal structure of winning tickets in this study, comparing equally-sized winning tickets derived from networks of different sizes would shed light on the extent to which the winning tickets themselves are similar or different between various initial network sizes.

6.3 Exploring Iterative Pruning Rates

Choosing the exact rate at which to prune on each iteration of iterative pruning entails balancing the performance of the resulting winning ticket with the number of iterations necessary to extract that winning ticket. Figure 15 shows the convergence times and accuracy for the LeNet-300-100 architecture iteratively pruned at different rates on each iteration. (Note that the x-axis is not logarithmic.) This experiment can be thought of as exploring middle grounds between one-shot pruning and iteratively pruning at a small rate.

Although pruning by a larger percentage (e.g., 70% or 50%) on each iteration reaches smaller winning tickets faster, those winning tickets are pruned too aggressively and fail to match the convergence times or accuracy of winning tickets pruned more slowly. On the other end of the spectrum, iteratively pruning by 10% appears to achieve the best convergence times and accuracy but would require training the network 28 times to extract a winning ticket 5% of the original network's size. For our experiments, we prune by 20%, which balances performance with the amount of training required.

6.4 Weight Resetting

Before each training iteration of our iterative pruning approach, we reset the weights of the unpruned connections to their original values from before training. Doing so is part of our experiment to evaluate the lottery ticket hypothesis: exploring how well winning tickets obtained by pruning train in isolation. We conjecture that resetting before each training iteration makes it easier to find small winning tickets. In effect, each iteration is a recursive pruning problem in which a subnetwork that trains effectively when starting from the original initializations must be pruned to a slightly smaller network that does the same.

In contrast, Han et al. [4] interleave training and pruning without ever resetting weights. After a round of training, low-magnitude weights are pruned and then training continues based on the trained

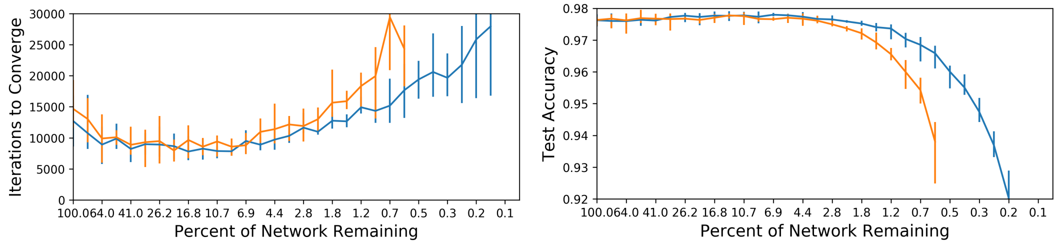

Figure 16: The convergence times and accuracy of winning tickets extracted by iteratively pruning using weight resetting between iterations (our strategy, in blue) and by continuing to use the trained weights after pruning (Han et al [4]'s strategy, in orange).

weights. These differences in approaches reflect two different goals: Han et al. want to produce the smallest possible trained network, while we wish to find a pruned network that trains successfully from the start.

Figure 16 shows the convergence times and accuracy achieved by the winning tickets extracted using these two pruning strategies. To simulate Han et al.'s strategy, we iteratively trained a network, pruned low-magnitude weights, and continued training using the trained weights. At each iteration, we copied the resulting network, reset its weights to the original initializations, and trained the network to obtain the results in Figure 16.

As Figure 16 shows, Han et al.'s pruning strategy is quite effective at finding small networks that rain successfully, although our strategy of resetting weights at each iteration maintains lower convergence times and higher accuracy for slightly longer. However, since Figure 16 is on a logarithmic scale, these differences appear only at very small network sizes.

7 Related Work

Pruning. LeCun et al. first explored pruning as a way to reduce the size of neural networks [7]; they pruned based on the second derivative of the loss function with respect to each weight. Hassibi et al. [5] build on this approach.

More recently, Han et al. [4, 3, 2] showed that these techniques could be used to substantially reduce the size of modern image-recognition networks. Since then, a rich variety of neural network pruning approaches have emerged (e.g., pruning smallest weights [4], pruning units in a Bayesian fashion [9], pruning entire convolutional filters [8, 10], fusing redundant units to increase network diversity [11]). The goal of this literature on pruning is to compress trained neural networks, reducing the size of a large model such that it can run efficiently on restricted computational platform (e.g., a mobile device) without sacrificing accuracy. In contrast, we aim to make it possible to train small neural networks from the start.

In [4] and follow-up work, network compression takes place in three iterative steps. First, a large network is trained. Second, weights or units are pruned according to a heuristic. Third, the network is further trained using the already-trained weights. Han et al. find that, without this third retraining step, network performance drops off much earlier in the pruning process. Han et al. also caution that the pruned network should not be re-initialized after training, but do not consider reusing the values to which the surviving weights were initialized in the original network as we do.

Our work builds off of the literature on pruning by shedding light on the mechanisms that make pruning possible. The fact that networks can be pruned while maintaining accuracy indicates that the function to be learned can be represented with a much smaller network than the one used for training. We aim to understand why pruning is possible and investigate whether small networks can be trained directly (rather than pruning large networks to smaller sizes after training).

The lottery ticket hypothesis posits that large networks have small, fortuitously-initialized subnetworks that facilitate successful training. From this point of view, neural network pruning finds these winning tickets. To evaluate the lottery ticket hypothesis on small, fully-connected networks, we leverage Han et al.'s experimental approach, except that we make a crucial modification: after pruning we reset each weight to its original value.

Our results explain or complement those of Han et al. The lottery ticket hypothesis offers insight into why Han et al. are able to prune networks. Many of the trends we see (e.g., that the accuracy of iteratively-pruned winning tickets drops off at very small winning ticket sizes or that the original initializations of pruned networks take on a bimodal distribution) parallel those that Han et al. find when continuing to train pruned networks based on their trained weights.

Dropout. Dropout [13] creates a smaller subnetwork for each training iteration by randomly removing a subset of the units (or weights). At inference-time, a unit's activation is reduced by the probability that it was dropped out. Intuitively, dropout is intended to reduce overfitting and improve generalization by forcing units to remain robust to changes in the network. Follow-up work on dropout [1] has characterized training with dropout as "perform[ing] gradient descent...with respect to...the ensemble of all possible subnetworks" and inference with dropout as approximately computing the average over this ensemble.

In the terminology of dropout, our experiment aims to discover a single, particularly successful member of this ensemble of subnetworks. Our dropout heuristic is that, after training the network once without dropout, we drop out the lowest $k\%$ of weights (by magnitude after training) with probability 1 and all other weights with probability 0. In other words, we perform an extremely aggressive, coarse-grained form of dropout based on examining the results of training the network once without dropout.

However, our goal is different. Dropout is designed to regularize a network during training, a process that can be used to produce sparse networks. We aim to directly find small (and, in the case of the networks we found, sparse) networks that can be trained from start to finish without removing further weights.

Our broader formulation of the lottery ticket hypothesis does closely relate to dropout's notion of ensemble learning. The lottery ticket hypothesis views a randomly-initialized large network as a collection of a combinatorial number of small networks (i.e., lottery tickets) of which one (i.e., the winning ticket) must be initialized fortuitously to enable training to succeed. From this point of view, a large network begins with the possibility of coalescing toward one of an exponential number of subnetworks, and gradient descent drives it toward the subnetwork comprising the winning ticket that we find.

8 Limitations

This work is limited in several ways. We only examine fully-connected networks, and for two of the smallest possible examples (XOR and MNIST). We do not consider convolutional networks or larger networks that better reflect real-world examples.[8] Our evidence for the lottery ticket hypothesis is purely experimental; we do not offer any theoretical analysis to formally support this claim. Finally, although we analyze the structure and initialization distributions of winning tickets for MNIST, we have yet to devise a way to turn these observations into useful strategies for training smaller networks. We anticipate exploring these avenues in future work and updating this paper as we do so.

9 Conclusions and Future Work

This paper proposes a new hypothesis to explain why large neural networks are amenable to substantial pruning yet the pruned networks cannot be trained effectively from scratch. This conjecture, known as the *lottery ticket hypothesis*, holds that training succeeds when a subcomponent of the larger network is randomly initialized in a fashion that is suitable for optimization. Furthermore, it conjectures that pruning uncovers these *winning tickets*. To empirically evaluate this hypothesis, we devised an experiment based on the work of Han et al. [4] where, after pruning a trained network, remaining weights are reset to their original initializations. If the lottery ticket hypothesis holds and pruning

[8] However preliminary experiments with CIFAR10 on a convolutional network reflect the same behavior described in this paper for MNIST.

uncovers these winning tickets, then these pruned networks should train successfully in isolation when reset to their original initializations.

On XOR, we found that winning tickets derived from larger networks were able to learn the decision boundary and reach zero loss far more frequently than those that were randomly initialized. On MNIST, winning tickets converged more quickly and reached higher accuracy than the original network. Control experiments supported the claim that winning tickets represent a confluence of fortuitious initialization and network architecture.

This paper articulates a new perspective on neural network training and supports that view empirically. Now that a foundation has been laid, there are numerous research directions to further evaluate the lottery ticket hypothesis and exploit this perspective to improve network design and training.

Larger examples. The largest network that we examine is a fully-connected network for MNIST. Repeating the experiments outlined in this paper for a convolutional network, larger networks, and harder learning tasks would make it possible to understand whether the lottery ticket hypothesis holds more generally and how it manifests in these settings.

Understanding winning tickets. This paper focuses mainly on the behavioral properties of the lottery ticket hypothesis, pruning, and winning tickets. One logical next step is to systematically analyze the architectures and initializations of lottery tickets. To what extent are winning tickets unique artifacts created by randomly initializing large networks and getting lucky? To what extent is there common structure between multiple winning tickets for the same task? What do winning tickets tell us about the functions that neural networks learn for particular tasks?

Lottery ticket networks. The lottery ticket hypothesis and the existence of winning tickets demonstrate that small networks can be trained from start to finish. The most concrete follow-up to this work would be to exploit the lessons learned by leveraging winning tickets to develop new network architectures and initialization regimes that allow smaller networks to be trained for a wider variety of learning tasks. Doing so could reduce the amount of computation needed to train neural networks.

References

[1] Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. In *Advances in neural information processing systems*. 2814–2822.

[2] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). arXiv:1510.00149 `http://arxiv.org/abs/1510.00149`

[3] Song Han, Huizi Mao, and William J Dally. 2015. A deep neural network compression pipeline: Pruning, quantization, huffman encoding. *arXiv preprint arXiv:1510.00149* 10 (2015).

[4] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. 1135–1143.

[5] Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *Neural Networks, 1993., IEEE International Conference on*. IEEE, 293–299.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

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

[8] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[9] Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*. 3290–3300.

[10] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342* (2017).

[11] Zelda Mariet and Suvrit Sra. 2015. Diversity Networks: Neural Network Compression Using Determinantal Point Processes. *arXiv preprint arXiv:1511.05077* (2015).

[12] Suraj Srinivas and R Venkatesh Babu. 2015. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149* (2015).

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.