MACHINE INTELLIGENCE COMMUNITY

The Lottery Ticket Hypothesis: Training Pruned Neural Networks

Jonathan Frankle, Michael Carbin

Summary By Devin de Hueck

1. Introduction

- It has been shown that neural networks can be dramatically pruned without a loss of accuracy.
    - If a network can be pruned and retain accuracy then the function being learned can be represented by a far smaller network.
    - Previous researchers have stated they do not believe these smaller networks can be readily retrained.
- **The lottery ticket hypothesis states:** *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.*
- Subnetworks and winning tickets
    - 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.
    - Large neural networks are more likely to train successfully due to having combinatorially more subcomponents (i.e. more lottery tickets).
- Methodology
    - To extract a winning ticket:
        1. Randomly initialize neural network.
        2. Train until convergence
        3. Prune
        4. Reset weights to original initialization
    - Winning tickets should be able to be retrained successfully at sizes too small for a randomly initialized network to do so.
- Research questions
    - How effectively do winning tickets train in comparison to the original network and to randomly sampled networks of similar size?
    - How big are winning tickets relative to the size of the original network?
    - How sensitive are our results to particular pruning strategies?

2. Learning the XOR Function

- A neural network with two hidden units is sufficient to perfectly represent the XOR function, but a network with much more (around 10 total) hidden units is needed to achieve a high likelihood of a perfect representation of the XOR function when trained from random initialization.

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

- The figure above contains the overall success rate of networks that found the right decision boundary (DB) or reached zero loss (ZL) in 1000 iterations *with random initializations*.

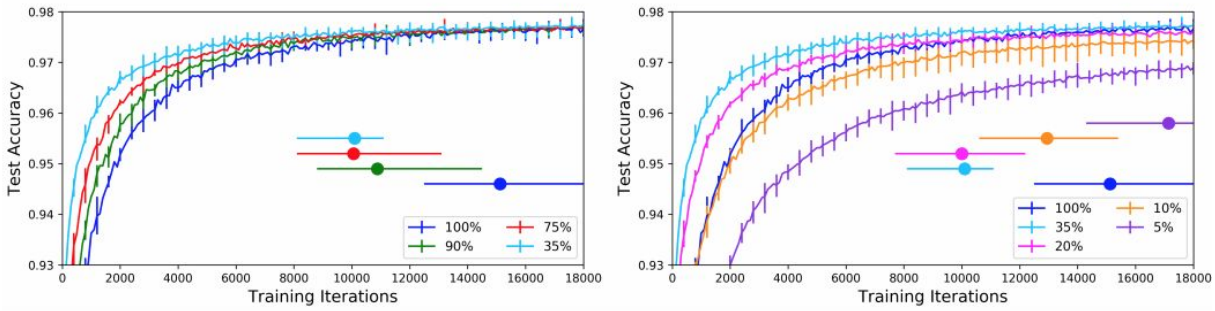
Pruning Strategy	10 Units		4 Units (Pruned)		2 Units (Pruned)	
	DB	ZL	DB	ZL	DB	ZL
One-shot Product	99.2	93.3	98.0	90.3	82.4	65.3
Input Magnitude	98.9	93.5	97.9	92.2	83.8	76.5
Output Magnitude	99.0	93.3	96.9	85.9	78.6	56.1
Product	98.5	92.9	97.6	90.3	91.5	79.4

- The figure above displays the results of pruning and then retraining the pruned network with the original initializations as is consistent with the hypothesis.
    - **One-shot pruning** - pruning the network in a single pass.
    - **Iterative pruning** - repeating steps 2 through 4 in the methodology laid out in the introduction.
        - In this experiment the networks were pruned by 2 units in each iteration.
- Pruning heuristics
    - **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
        - This provides the best results, and is used throughout this paper unless otherwise stated.
- Results

- With one-shot pruning a pruned 2 unit network that finds the correct decision boundary increased from 49.1% to 82.4% when the network was initialized as a winning ticket.
- With iterative pruning a pruned 2 unit network that finds the correct decision boundary increased from 49.1% to 91.5% when the network was initialized as a winning ticket.
    - While iterative pruning is more computationally expensive the results indicate that this method finds winning tickets at a higher rate than one-shot pruning
- These results also reflect the findings of Han et al. in that the magnitude product heuristic provides the best results.
    - Could be due to the fact that in the XOR case when all inputs are either 0 or 1, the product of the input and output magnitudes should mimic the activation of the function.

3. MNIST (One-shot Pruning)

- LeNet-300-100 architecture used. By default, biases were initialized to 0 and weights were randomly sampled from a normal distribution with a mean of 1 and a standard deviation of 0.1.
- *In this experiment only individual weights are removed instead of the entire unit.* The pruning heuristic is as follows: remove those weights with the lowest magnitude within each hidden layer. Weights connecting to the output layer are pruned by half the percentage at which the rest of the network is pruned in order to avoid severing connectivity to the output units.
- The methodology follows the same schema as described in the introduction.
- Results:



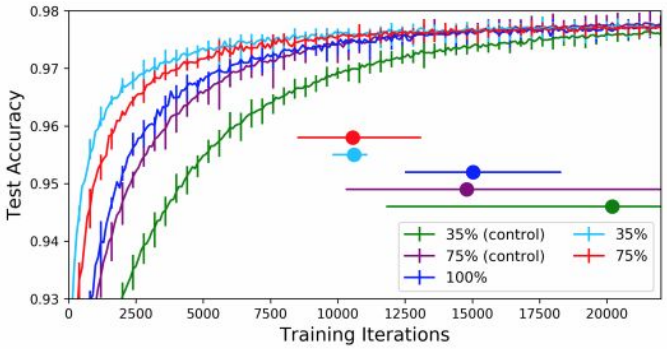
- In the figure above the percentages correspond to the amount of the network remaining after pruning. Each curve shows the average progression of five trials of training at a percentage pruned with vertical error bars marking the minimum

and maximum accuracy at the number of iterations . The horizontal error bars mark the minimum to maximum convergence times of the trials with the average convergence time marked by the dot.

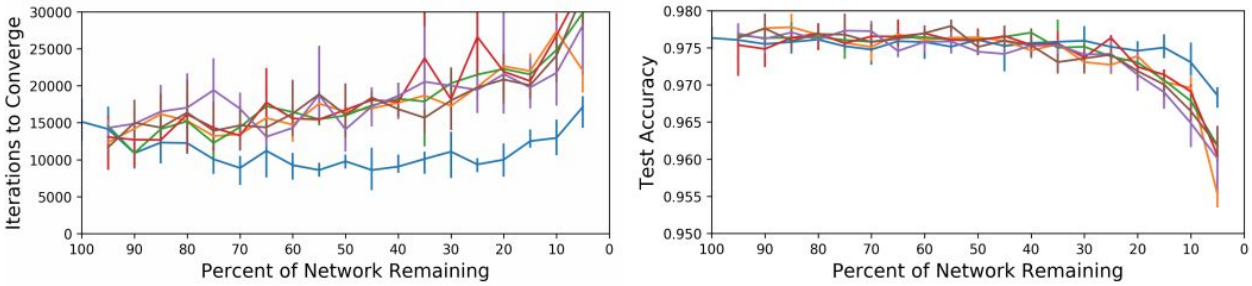
- The left graph indicates that for the first few pruning levels the convergence times are faster and accuracy increases as more of the network is pruned. This pattern continues until a pruning level of around 55%
- In the right graph the patterns observed in the left graph are no longer in play. Convergence times flatten then become slower and accuracy drops.

- Pruning had the greatest impact on convergence times. When pruned to between 75% - 25% of the network, size convergence times increased on average by at least 25%. Accuracy was maintained within 0.15% of the original networks accuracy.
  - This is attributed to the removal of unnecessary, noisy parts of the network.
  - There appears to be a tipping point as pruning eventually removes weights that are essential to a winning ticket. After this tipping point convergence times slow down and accuracy falls.
  - The lottery ticket hypothesis predicts this behavior is largely attributable to a confluence of a networks initialization and architecture.
    - Two control experiments are run to test this assertion.

- *Control Experiment 1*
  - Evaluates the extent to which initialization is a necessary component of a winning ticket.
    - Train a winning ticket with random initializations instead of the original initializations to isolate the impact of the architecture using the original networks random initialization distribution - $N(0,\ 0.1)$
  - Results:



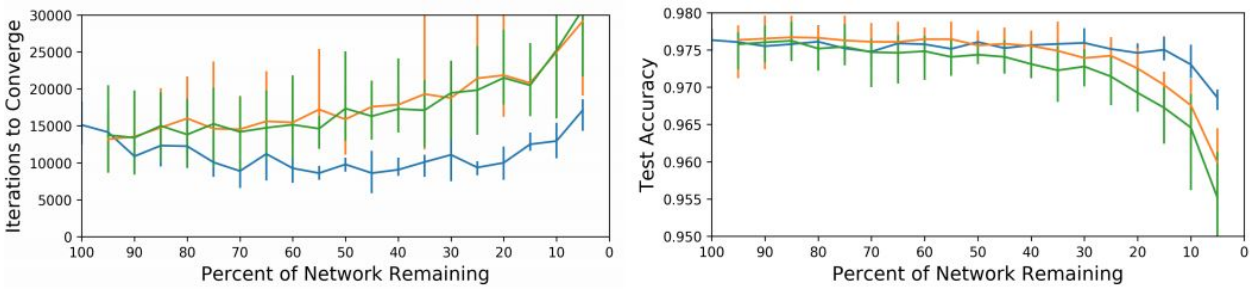
- In the graph above control networks are the pruned networks reinitialized with random weights. Each control curve is the average of 15 trials.
- The control networks converged on average more slowly than the original network and achieved lower levels of accuracy. These differences were non-trivial.
    - The average 35% and 25% winning tickets converged 1.91 and 2.28 times as fast as their respective averaged controls.
    - As the horizontal errors bars reflect, the control networks exhibited a much higher level of variance.



- In the graphs above the blue line is the average of the five winning ticket trials at each level. Each of the multi-colored lines represents three randomly initialize control trials.
- The graphs provide a larger perspective on the differences between the control trials and winning ticket trials in the context of convergence and accuracy.

- This experiment provides great insight into the components of a winning ticket by showing that the structure alone is insufficient to explain the success of a winning ticket.

- *Control Experiment 2*
    - Evaluates the extent to which the architecture of the winning ticket is necessary to the success of a winning ticket.
        - Randomly shuffle the location of a winning ticket's weights within each hidden layer while maintaining the original initializations to isolate impact of the architecture from the initializations.
    - Results:

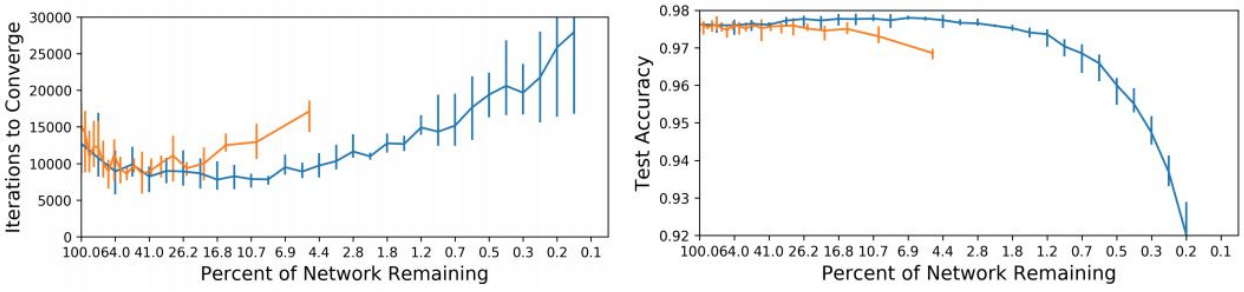


- In the graphs above the blue line is the average of the five winning ticket trials at each level. The orange line is the average of all 15 control trials in *control experiment 1*. The green line is the average of all 15 trials from this experiment - *control experiment 2*.
- It is relevant to note that both control experiment 1 and control experiment 2 approximately follow the same trend. So, just as in control experiment 1 this experiment shows that initialization alone is insufficient to explain the success of a winning ticket.

- Overall, this set of experiments shows that even when pruned to sizes much smaller than the original network winning tickets are able to converge and often do so faster without losing accuracy. In addition, it was shown that winning tickets emerge from a confluence of both initialization and architecture.

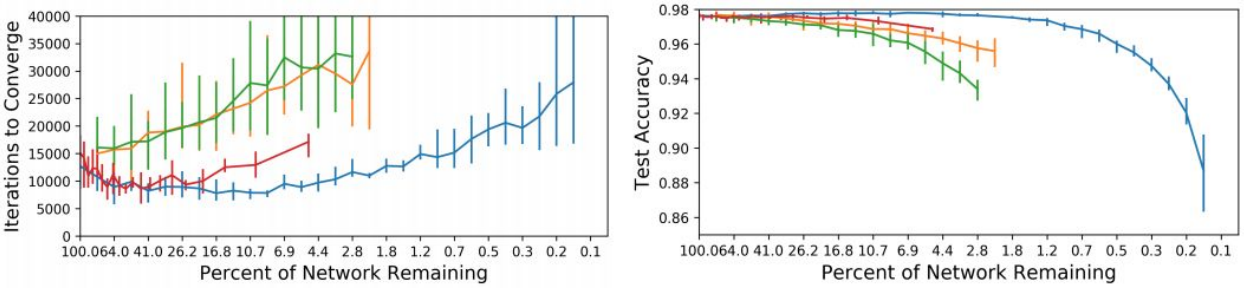
4. MNIST (Iterative Pruning)

- Here it is demonstrated that iterative pruning can find far smaller winning tickets than one-shot pruning.
- Methodology recap and MNIST specific procedures:
    1. Randomly initialize the network
    2. Train for 50,000 iterations
    3. Prune 20% from each hidden layer, and 10% from the output layer. Remove those with lowest magnitudes.
    4. Reset to original weights.
    5. Repeat steps 2 through 4 until network is pruned to desired size.
- Comparison to one-shot pruning



- In the graphs above the blue line represents iterative pruning and the orange line represents one-shot pruning. Note: x-axis is logarithmic
- With iterative pruning, convergence times continue to speed up and then remain flattened longer compared to one-shot pruning.
  - The average iteratively pruned network returns to the original networks convergence time when pruned to 1.8% of the network compared to between 5% and 10% for a one-shot pruned network.
- Accuracy slightly increases before matching the original networks accuracy once the network has been pruned to 1.8% of the original. Comparatively, one-shot pruning accuracy begins to drop off at 15% of the original network.

- While iterative pruning can find much smaller winning tickets it is much more computationally expensive compared to one-shot pruning. For example, when iteratively pruning 20% of the network down to around 5% of the original network the network needs to be trained 14 times.
- Control Experiments
  - These experiments are a a re-run from the section: MNIST (One-shot pruning)

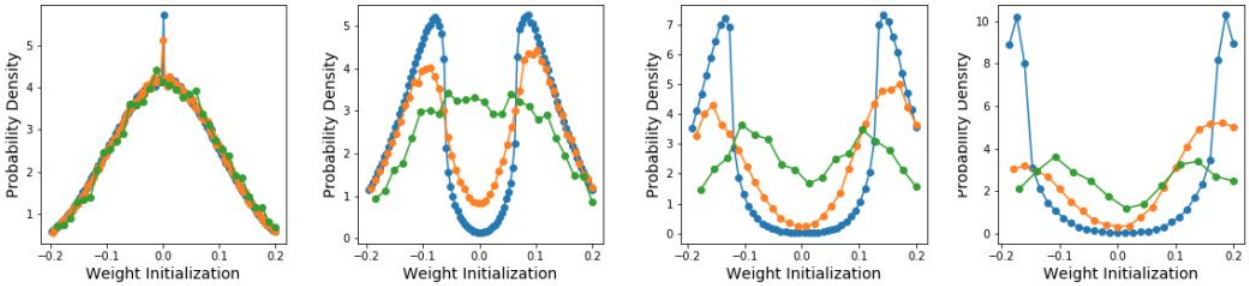


- In the graphs above the blue line is the average of five winning tickets, the orange line is control experiment 1, the green line is control experiment 2, and the red line is the performance of one-shot pruning .
- *Control Experiment 1 - testing initializations*
  - Control trial's convergence times begin to slow down as the network is pruned and continues to follow this trend.

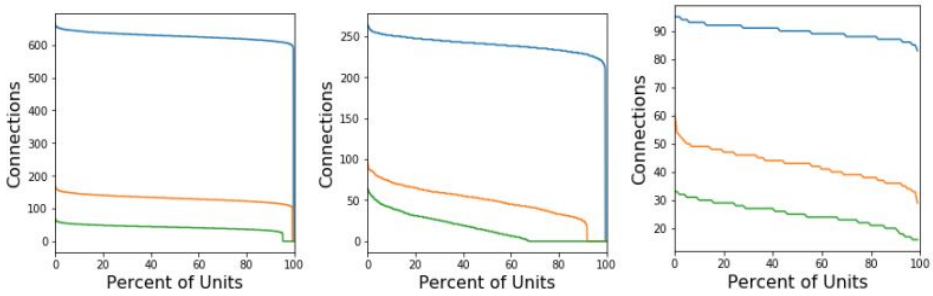
        - The average control trial accuracy begins to drop off once pruned to 37% of the original network. Meanwhile, the average iteratively pruned network drops off when pruned to 1.5% of the original network.
    - *Control Experiment 2 - testing architecture*
        - Convergence slows as patterned by control experiment 1.
        - Accuracy begins to drop off earlier that control experiment 1, around 52% , suggesting that the architecture of a winning ticket may be more important that initialization to a winning ticket.

5. Examining Winning Tickets

- Exploring the internal structures of winning tickets that result from iteratively pruning the MNIST network.
- Initializations



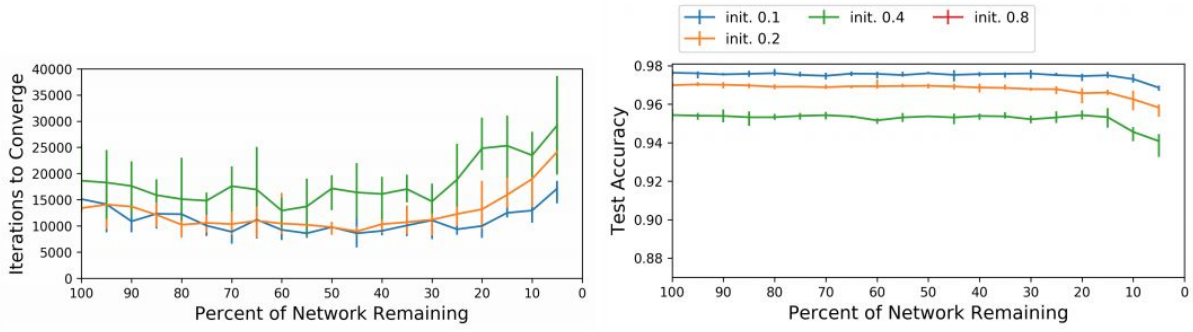
    - The graphs above represent the distribution of the initialization weights *before training* that were left after pruning over 10 iterative prunings. From left to right the respective networks sizes are 100%, 51.2%, 16.8%, 5.5%. The blue, orange, and green lines are the distributions of initial weights for the first hidden layer, second hidden layer, and output layer, respectively.
    - As is readily viewable in the graphs, iterative pruning gradually results in approximate bimodal distributions. As pruning occurs after training and these are the weights from before training it appears the tail values of the original normal distribution survive training.
    - When the hidden layers still resemble the original distribution it is likely the weights moved more during training.
- Architecture



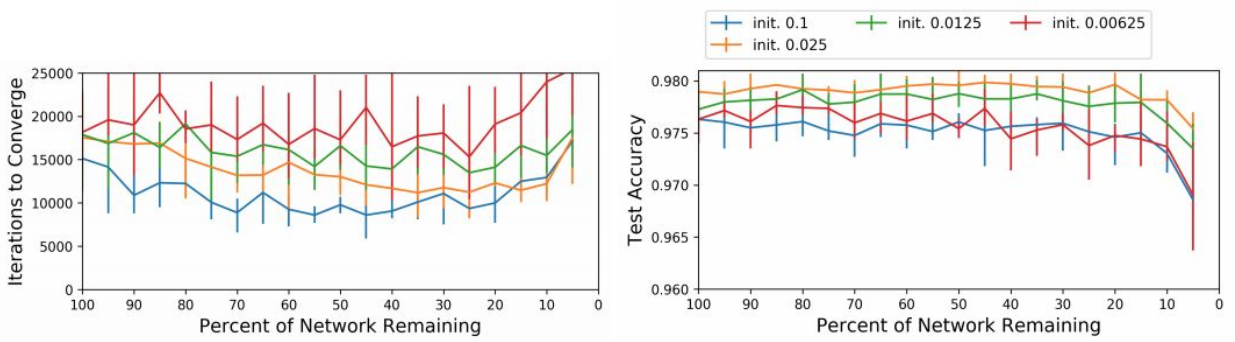
- In the graphs above the number of connections from the previously layer is plotted. From left to right the first hidden layer, second hidden layer, and output layer are represented. The blue, orange, and green lines indicate winning tickets when iteratively pruned to 80%, 16.8%, and 5.50%, respectively.
- Each point on the line represents a single unit; the units have been sorted in descending order by the number of connections they have.
- Winning tickets are quite sparse in terms of connections but only a fraction of the units have been eliminated entirely.
- Connections nearly decrease in proportion to the pruning rate.

6. Exploring MNIST Parameters

- The roles of initialization and network size are explored to better understand the sensitivity of the MNIST results.
- Initialization
  - While the default initializations have been selected from a normal distribution with mean 0 and standard deviation 0.1 other standard deviations were used to explore the effects of larger and smaller weights on winning tickets.

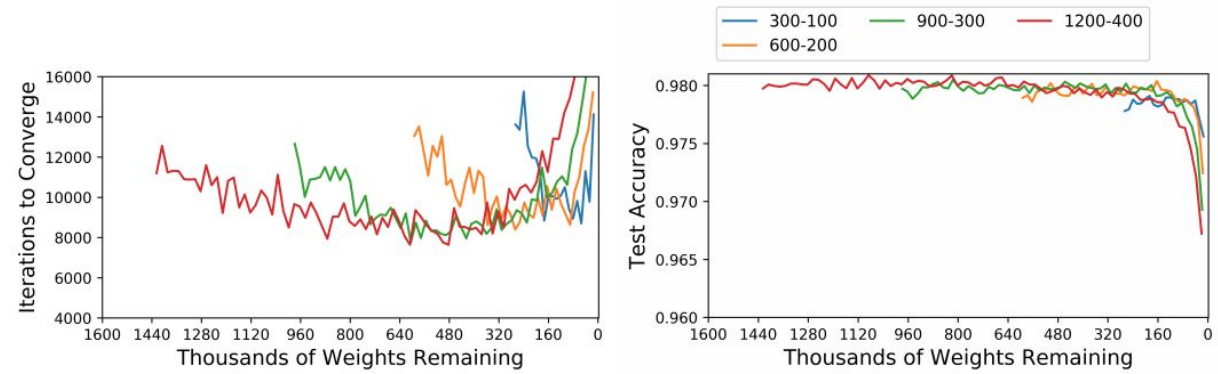


- The graphs give the convergence times and accuracy of winning tickets initialized with standard deviations ≥ 0.1.
- Convergence times become slower and accuracy decreases with larger standard deviations.



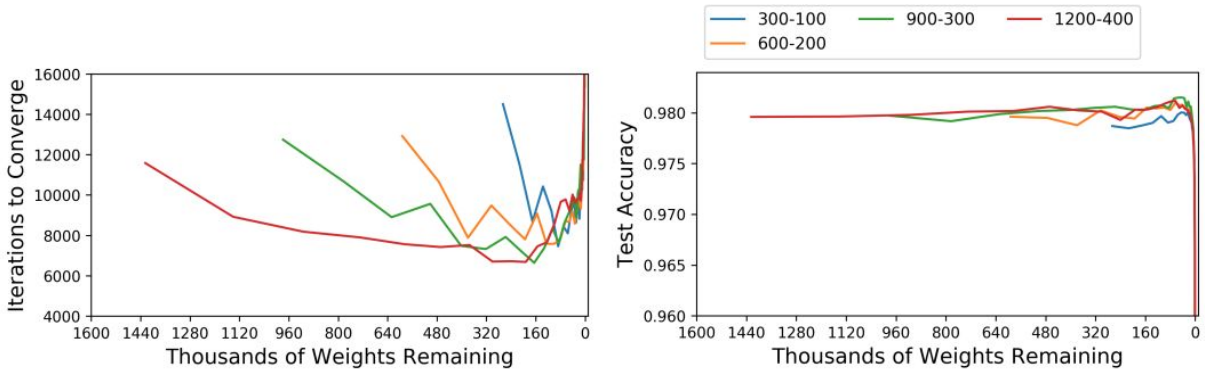
- The graphs give the convergence times and accuracy of winning tickets initialized with standard deviations ≤ 0.1.
- While accuracy increases as standard deviations decrease convergence times become slower.

- Network size
  - The size is increased in order to test if there is a fixed winning ticket size for a learning problem or if a larger network leads to a larger winning ticket. The size of the ticket is considered in two different definitions:
    - A winning ticket is the smallest network that minimizes convergence time.
    - A winning ticket is the smallest network that minimizes accuracy times.
  - One-shot pruning



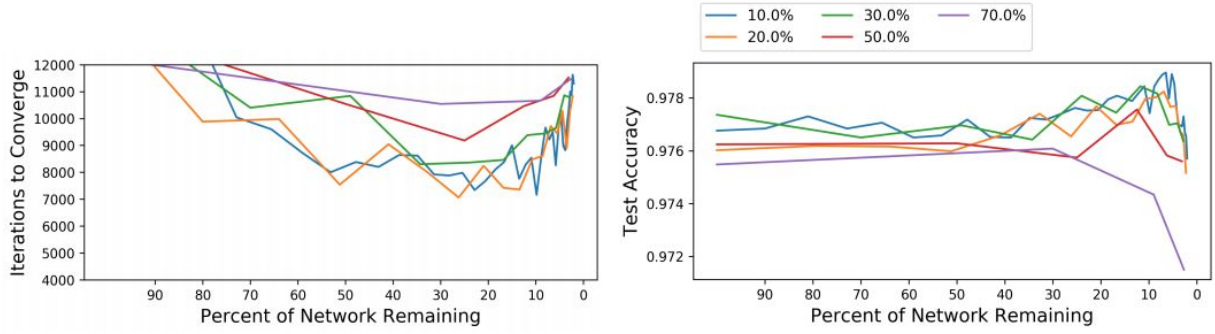
- Each line in the graphs above refer to the average convergence or accuracy of a group of 5 winning tickets. The legend in the figure above refers to network size (e.g. 300-100 corresponds to a network with hidden layers of 300 and 100 units.)
- According to the definition of size based on convergence times, the winning ticket size gradually increases with the size of the original network. The accuracy based definition found similar results although the differences were smaller.

- Iterative pruning



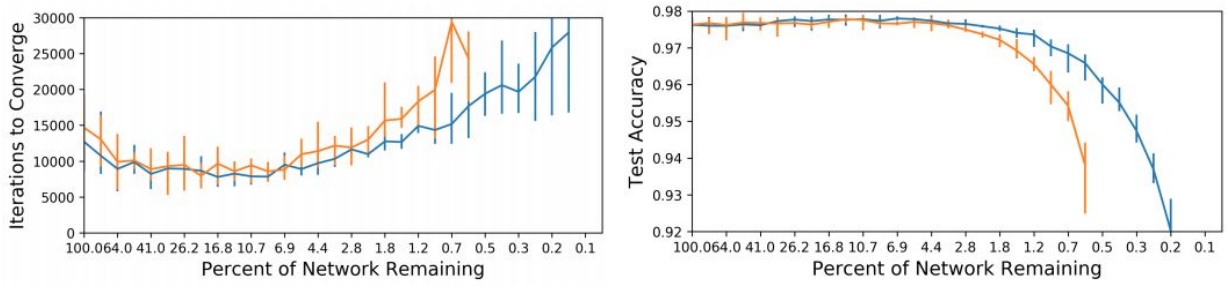
  - The figure above represents the same values as in the one-shot pruning trials.
  - The results reflect the findings of the one-shot pruning method except for two key differences.
    - The minimum convergence times and accuracy drop offs occur at much smaller sizes which reflects the previous findings that iterative pruning results in smaller winning tickets.
    - As seen in the right graph there is a slight bulge right before the drop off at about the same point no matter the original network size.

- Exploring iterative pruning rates
  - Choosing the rate at which to iteratively prune is a balance between performance of the winning ticket and the number of iterations needed to extract the winning ticket.



  - The figure above are the results of iteratively pruning a LeNet-300-100 architecture at the specified rates.
  - Winning tickets extracted by aggressively pruning (e.g. 70% or 50%) have poorer convergence times and accuracy than winning tickets pruned more slowly.
- Weight resetting

- Han et al., whose work is detailed further in the Related Work section, iteratively prunes networks without resetting the weights in order to find the smallest trained network. Meanwhile, this paper has reset the weights to the original initialization in order to find the smallest network that can be trained in isolation. The differences between these two methods are explored.



- In the graphs above this paper's method is represented by blue and Han et al.'s method in orange. Note that the x-axis is logarithmic.
- The results show that this paper's method maintains faster convergence times and holds accuracy longer than Han et al's strategy. However, these differences only appear at very small network sizes.

7. Related Work

- Pruning
  - Han et al. showed that pruning techniques can be applied to greatly reduce the size of modern image recognition networks without sacrificing accuracy.
    - In following work, Han et al. cautioned against reinitializing pruned networks but did not consider reusing the original weights.
    - Many of this paper's findings parallel the work of Han et al.
- Dropout
  - This method creates smaller subnetworks by randomly removing subsets of the units or weights on each iteration. This method is used to reduce overfitting and improve generalization.
    - In this paper's experiments an extremely aggressive form of dropout based on the results of training the network can be said to have been performed. Dissimilarly, this paper has aimed to find smaller subnetworks that can be trained without removing anymore units or weights.
    - This paper and the dropout strategy both view a randomly initialized large network as combinatorial collection of smaller subnetworks.

8. Limitations

- Only examined fully-connected neural networks and some of the smallest examples at that. No consideration of CNNs or larger networks.
- No theoretical analysis to support the hypothesis.
- Have yet to devise useful strategies for training neural networks from these results.

9. Conclusions and Future Work

- With XOR, winning tickets were able to find the correct decision boundary and reach zero loss at a much higher rate than randomly initialize networks of the same size.
- With MNIST, winning tickets reached higher accuracy and converged quicker than the original network.
- Potential research directions:
    - *Larger examples* - Does the hypothesis generalize?
    - *Understanding the winning tickets* - Are winning tickets made by chance in large networks? Is there a common structure between multiple winning tickets for the same task? What do winning tickets tell us about the function being learned?
    - *Lottery ticket networks* - Can winning tickets be leveraged to develop new neural network architectures and initialization strategies that allow for smaller networks to be trained for a wider scope of tasks?