Replicating The Lottery Ticket Hypothesis

Kevin Ammouri
kammouri@kth.se

Youssef Taoudi
yousseft@kth.se

Abstract

In this work, we examine how the lottery ticket hypothesis algorithm affects neural networks for different architectures. We explore the behaviour on both Fully-Connected networks and Convolutional Networks with different sizes and applied regularisation techniques. The pruning techniques used in this report are magnitude-based iterative pruning and one-shot pruning. Iterative pruning showed to be the most effective on both FC-networks and convolutional networks on the MNIST and CIFAR10 datasets respectively. The lottery ticket hypothesis shows that low-density level subnetworks with high accuracy and good training performance, referred to as winning tickets can be found consistently in fairly shallow fully connected and convolutional networks that use batch normalization or dropout regularization by permanently disabling low magnitude weights and retaining the structure of the original unpruned network.

1 Introduction

This report will aim at replicating the lottery ticket hypothesis by Jonathan Frankle and Michael Carbin published 2019[1]. The entirety of the report consists of different pruning techniques that eliminate unnecessary parameters (weights) in a neural network. A lottery ticket is a subnetwork that achieves at least the same performance than the original network even though it has fewer parameters (having a large portion of the weights disabled). The reasoning behind pruning a network is "[...], making inference more efficient"[1, p. 1]. In this report, the focus will be on two pruning techniques, iterative and one-shot pruning. The goal of the project is to find extensively pruned lottery tickets that may be retrained quicker than the original network but still maintain the same accuracy for some widely-used neural network types and architectures.

In this project, the results were similar to the ones shown in the original paper by Jonathan Frankle and Michael Carbin[1]. This means that these pruning techniques are legitimate and can be applied to almost all network types and architectures. However, when the original network is pruned more than approximately 96% one will start to see a decrease in accuracy, this behaviour is also shown throughout the original paper this project aimed to replicate.

It is important to note that no specific pruning library, for instance, Keras-Surgeon[2], was used during this project. The entire pruning procedure, both iterative and one-shot, was done from scratch to get a better understanding of the underlying computations behind pruning.

2 Related Work

The main paper related to this work is, as mentioned before, the Lottery Ticket Hypothesis by Jonathan Frankle and Michael Carbin[1]. The paper shows how one can prune a neural network efficiently without sacrificing performance. This is done by randomly initializing a neural network, $f(x;\theta_0)$, train the network for j iterations, where the network now will contain parameters θ_j, and prune some percentage of the parameters trained, creating a mask m which is the structure of the subnetwork[1, p. 2]. Finally, reset the structure's parameters to the values in θ_0 and re-train the subnetwork (the

Table 1: Distribution of images for all datasets

Dataset	Training set	Test set	Validation set
MNIST	54000	10000	6000
CIFAR-10	45000	10000	5000

winning ticket)[1, p. 2]. Frankle explains that networks can be pruned up to 96.6% and still maintain similar performance levels as the original network at a comparable amount of training iterations[1, pp. 2–3]. The paper thoroughly examines the behaviour of the pruning techniques on different network types and architectures namely Fully-Connected Networks, Convolutional Networks, VGG and ResNet. The results are different for each network type, however, the general trend holds, that pruning networks results in the subnetwork training better as well as showing better inference than that of the original network.

3 Data

The datasets used for this report are MNIST[3] and CIFAR-10[4]. The fully-connected network uses MNIST as its evaluating dataset and the convolutional networks use CIFAR-10. The number of training images, test images and validation images can be found in Table 1. The validation sets are always 10% of the training set throughout the experiments for both datasets.

The datasets were downloaded via the Keras library and pre-processed accordingly. Since the pixel values for the images range between 0 and 255, normalisation of the data is done by dividing all elements in the set with 255, hence, giving all elements a value between 0 and 1. It is also important that the elements are of floating points. The procedure was identical for both the datasets used.

4 Methods

4.1 Network Types and Architectures

The networks used throughout the experiments are Fully-Connected Dense Networks, specifically LeNet-300-100 architecture[5], and Convolutional Networks, Conv-2, Conv-4 and Conv-6 which are from the VGG family[6]. The structure of networks can be found in table 2. Other settings such as type of optimizer, size of the batch, pruning rates as well as the number of trainable parameters (number of weights) can be found in table 3. Each convolutional module has a batch normalization[7] layer in front. Batch normalization is a technique that shifts and rescales output from the previous module and sends it as input to the next convolutional layer which generally boosts generalization[7]. Gaussian Glorot initialization[8] was used by all networks in the project.

For the most part, the architectures in this project mimicked that of Frankle and Carbin in the original paper[1, p. 2] due to their extensive hyperparameter exploration (over 3000 trained networks!) along with the general rules of deep convolutional architectures presented by Simonyan in [6]. However, none of the two papers mentioned the early stopping criteria for the networks. Thus, the project also explored and defined early stopping criteria for each network.

For some experiments, dropout was implemented into the architecture of the ConvNets. Dropout is a regularisation technique that improves accuracy by disabling a portion of the units during gradient descent [9]. The networks that experimented with dropout regularisation had a dropout rate of 0.5 for the first two fully-connected layers as suggested by Simonyan in [6, p. 4], however, in this report a dropout rate of 0.25 was used instead for the first two fully-connected layers. Furthermore, because batch normalization generally does not cooperate well with dropout [7], all batch normalization layers were disabled for the dropout experiments.

Table 2: Architecture of all the networks used during the experiments, table layout inspired by Simonyan[6].

Network Configurations			
FC-Network using MNIST (input: 28x28 image)	ConvNet using CIFAR-10 (input: 32x32 RBG image)		
LeNet-300-100	Conv-2	Conv-4	Conv-6
	BatchNorm		
	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
	maxpool		
		BatchNorm	
		conv3-128 conv3-128	conv3-128 conv3-128
		maxpool	
			BatchNorm
			conv3-256 conv3-256
			maxpool
FC-300 FC-100 FC-10	FC-256 FC-256 FC-10		

Table 3: Parameter settings of all the networks used during the experiments.

Network	**Optimizer**	**Pruning rate (fc/conv)**	**Batch size**	**Number of trainable parameters (All/Conv)**
LeNet-300-100	Adam 1.2e-3	20%	60	266K
Conv-2	Adam 2e-4	20% / 10%	60	4.3M / 38K
Conv-4	Adam 3e-4	20% / 10%	60	2.4M / 260K
Conv-6	Adam 3e-4	20% / 15%	60	1.7M / 1.1M

4.2 Pruning methods

Pruning a neural network is the means of systematically eliminating parameters from an already existing network [10]. The pruning process of a neural network generally involves creating a binary data-structure known as a mask that when applied to a network, permanently shuts down the parameters corresponding to the zeros of mask [10]. Although there are many different pruning techniques applicable to different types of neural networks, this report focuses on the two pruning methods used in the original paper by Jonathan Frankle and Michael Carbin[1] on convolutional and fully connected networks; one-shot pruning and iterative pruning.

One-shot pruning, also known as single-shot pruning[11] is a pruning technique where parameters or connections are removed in a single swipe given a sparsity level and an already-trained network. One-shot pruning only requires pruning once compared to iterative pruning methods that could require many 'prune - retrain cycles' which would bring about an extensive amount of time for finding a winning ticket [11].

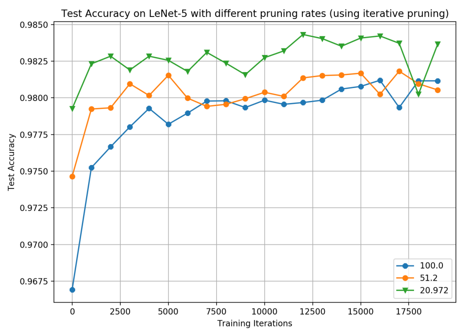
Contrary to the one-shot pruning technique, iterative pruning requires multiple prune-retrain cycles when dropping parameters. Two types of iterative pruning methods were discussed in the original winning lottery paper [1], pruning with resetting and pruning with continued training. Pruning with resetting is the means of iteratively pruning a network into a subnetwork by some small pruning rate followed by reinitializing and retraining the subnetwork. The process of pruning, reinitializing and retraining until the given sparsity level is satisfied are the main properties of pruning with resetting [1, p. 14]. Pruning with continued training is very similar to pruning with resetting but it does not include the reinitializing step. Instead, the network is continuously pruned and trained without ever being reinitialized [1, p. 14]. Iterative pruning in this paper is limited to pruning with resetting as it was the best performing pruning technique in the original paper[1, p. 15].

There are two additional properties for pruning methods: creating the mask and how subnetworks should be reinitialized after pruning. There are many ways to determine which parameters should be dropped from the network. Some examples of techniques are magnitude-based [1][12], random[12], filter-based (absolute kernel sums in CNN's)[13] and unit/node-based pruning[12]. In this project, only magnitude-based pruning was used where the smallest weights after training were used to create a mask for dropping weights. An additional note is that pruning was applied on a layer-per-layer basis for both types of networks explored in this report.

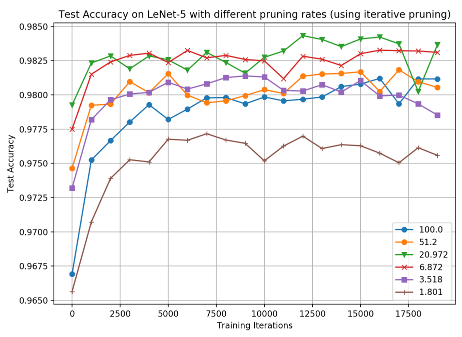
When pruning, a mask is generated based on the lowest magnitude weights. The mask is applied on a lottery ticket to permanently disable weights at initialization. The choice of initialization might, therefore, be an important factor for pruning. For that reason, this report will explore how the efficiency in terms of predicting and training is affected when a pruned network is reinitialized randomly versus if it is reinitialized with the exact same structure as the original network (without the disabled weights). The idea is that the experiment might give insight into if the same mask can be reused for different initializations of a network and into how important initialization is for a winning ticket[1]. Initializing using the original network's structure will be referred to as winning ticket initialization.

5 Experiments

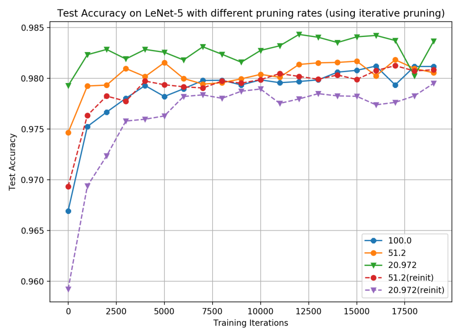
5.1 Winning Tickets in Fully-Connected Networks



(a) Plots of the largest density level winning tickets



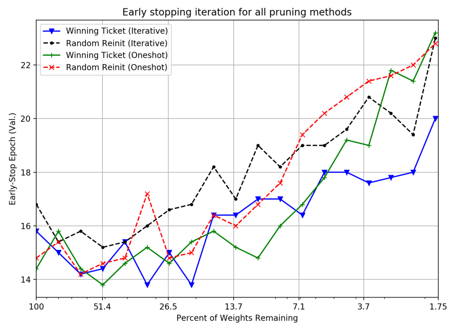
(b) Plots of all tested density level winning tickets



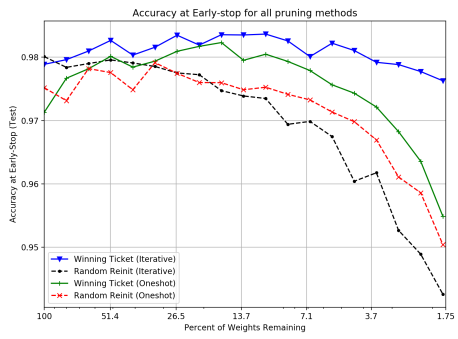
(c) Plots of winning tickets vs randomly reinitialized lottery tickets

Figure 1: Evolution of test accuracy on the MNIST dataset during iterative pruning for different density levels using the LeNet-300-100 architecture. Each curve represents a density level. Dashed lines are curves that have been achieved through random initialization as opposed to initializing with the original network's weights. Y-axis is the test accuracy and the X-axis is the time elapsed in training (in terms of training iterations). Each plot is the average of 5 trials, random reinitializations were only run once per trial contrary to the original paper where they were reinitialized thrice per trial[1, p. 4], effectively running the iterative process 15 times per dashed graph.

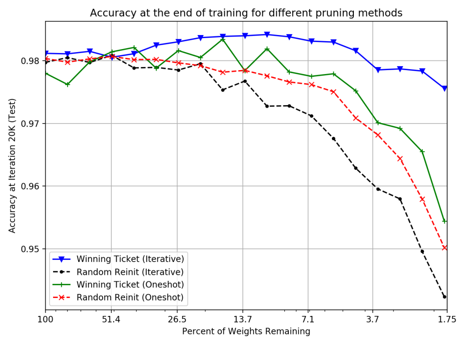
The first experiment revolved around analysing how the network performed on test data for varying sizes of winning tickets. The accuracy for winning tickets of different density levels is presented in Figures 1a and 1b respectively. An interesting find is that every subnetwork above a 3.5% density level performs better than the original network (100% of weights remaining) while still retaining the same training speed. Another observation is that the test accuracy is at it's highest for winning tickets with a density level of around 21%. Finally, in 1c, the randomly reinitialized graphs were compared to the retained weight initialization of the original networks. The results show that the randomly reinitialized plots perform worse, especially for lower percentages where the percentage unit difference exceeds over 0.05. Furthermore, the randomly reinitialized tickets seem to converge slower than the winning tickets.



(a) Early Step Criterion on validation data.



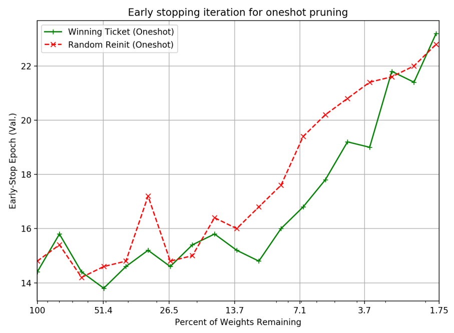
(b) Test accuracy when early stopped.



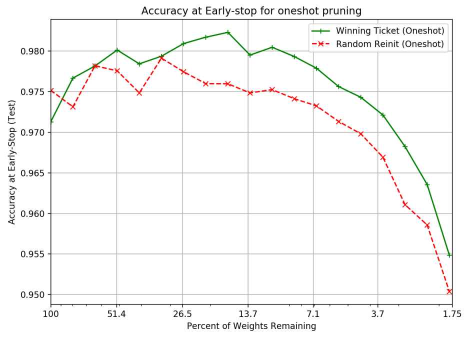
(c) Test accuracy after exactly 20 epochs (20k iterations).

Figure 2: Training time and test accuracy (on MNIST) for subnetworks of different density levels using one-shot and iterative pruning with either random initialization or winning ticket initialization. Each line is the average over 5 trials. The X-axis is the percentage of weights remaining (density) and the Y-axis represents at which epoch early-stopping occured or the test accuracy at a given density level (for a and b or c respectively).

The purpose of the second experiment on the LeNet-300-100 network, found in figure 2, was to analyze the rate of convergence and test performance of subnetworks when early stopped using the two pruning methods. Figures 2a and 2b encapsulate the behaviour at early stopping for different pruning methods on different levels of density. One-shot pruning and iterative pruning using the winning ticket approach of initialization generally performed better than the original network in terms of test accuracy until only around 3.5-4% of the weights remained, peaking at around 16% of the weights remaining for winning ticket pruning methods. Overall, iterative pruning performs better than one-shot for winning tickets. The early stopping criterion for the winning tickets remained fairly consistent down to roughly 10% of the weights remaining for both winning ticket methods, shooting up after the 10% mark. In summary, the winning tickets seemed to perform better than the original network in terms of test accuracy while the percentage of weights remaining was larger than



(a) Early stopping criterion for one-shot pruned FC-networks



(b) Test accuracy for one-shot pruned FC-networks

Figure 3: Comparison convergence of winning ticket initialization and random initialized networks using one-shot pruning. The X-axis is the percentage of weights remaining (density level) and the Y-axis represents at which epoch early-stopping occured or the test accuracy at a given density level (for a and b respectively).

approximately 3.5%. Even when purposely overtraining (figure 2c), the subnetworks performed just as well as when they hit their early stopping criterion showing signs of good generalization.

For the randomly reinitialized tickets, the test performance was generally lower than the winning tickets across the board, especially for the lower density level subnetworks. In figure 2b, one can observe a drastic decrease in test accuracy at around 20% of weights remaining for the randomly reinitialized tickets. Another note is that while the randomly initialized ticket follows the general convergence trend shown by the winning tickets, the drastic increase in convergence for smaller subnetworks is even more substantial as observed in 2c.

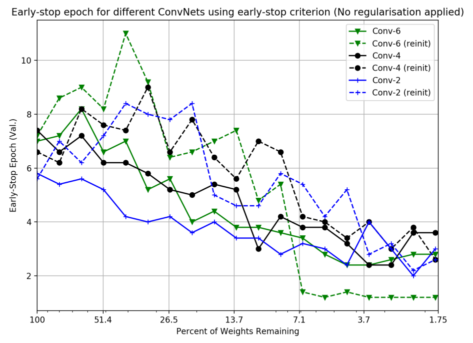
A follow-up experiment for the LeNet network, shown in figure 3, focused on one-shot pruning and how initialization affects the performance of a network. The purpose of the test was to further analyze the extent to which initialization matters in finding winning lottery tickets. The dashed line in figures 3 represent one-shot pruned subnetworks reinitialized with a new Gaussian Glorot initialization while the non-dashed line retained the weight initialization of the original network. By observing the figures, it is clear that the randomly initialized network not only consistently performs worse in terms of test accuracy, but also requires longer training to reach the early stop criterion. It must, however, be stated that the random tickets perform similarly to the winning tickets when only small portions of the networks have been pruned.

The findings in the experiments for the LeNet network are more or less in line with the findings of Frankle and Carbin [1]. The only noticeable difference between the experiments in this report and the original paper is how long it took for the networks to hit the early stopping criterion as the fastest early stop in figure 2a is at approximately 11.7k iterations (13 epochs) while some networks hit early stopping at around 5k iterations in the original paper. This is most likely due to the different early stopping criteria used for the experiments. An interesting note is that the randomly reinitialized network used in our tests had the same test accuracy and early stopping patterns to those used in the original paper even though they were only reinitialized once per trial compared to Frankle and Carbin's three per trial[1, p. 4].

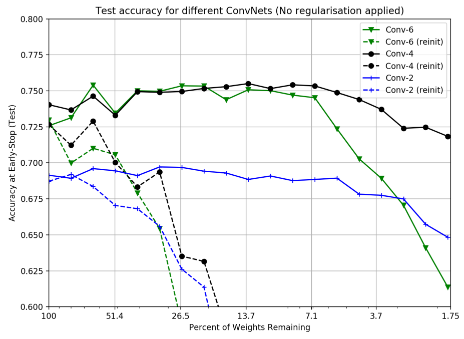
5.2 Winning Tickets in Convolutional Networks

In this section, pruning on the convolutional nets mentioned in section 4.1 will be examined. Furthermore, only iterative pruning will be considered to be consistent with the original paper and because it showed the best results overall as seen in the previous section.

Figure 4 shows the results of the experiments using the three convolutional architectures without any regularization. Compared to the LeNet architecture, the randomly reinitialized tickets performed much worse for the ConvNet architectures. In figure 4b the randomly reinitialized networks started



(a) Early stopping criterion for pruned ConvNets without any regularisation



(b) Test accuracy for pruned ConvNets without any regularisation

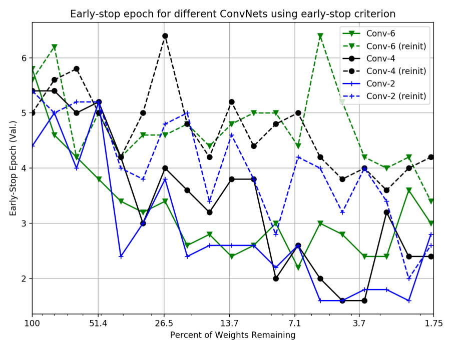
Figure 4: Comparison of convergence and test accuracy achieved by ConvNets without any regularisation applied. The plotted figures shows the results for ConvNets that had reinitialized their parameters randomly and those that had winning ticket initialization. Each graph represents results received from the average of 5 trials.

dipping in performance immediately, with a larger decrease the deeper the network is. The Conv-2 and Conv-4 architectures were able to find winning tickets in density levels all the way down to around 4% while the Conv-6 architecture had a dramatic decrease in prediction performance at around 7.1%. For the training duration in figure 4a, the winning ticket initialized subnetworks needed less time to train fairly consistently until 7.1% of the weights remained where the Conv-6 architecture started decreasing in accuracy. The amount of epochs required for training reduced fairly linearly for all networks until the final 10% density levels where training time stagnated.

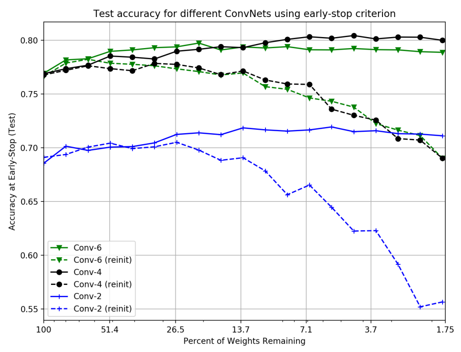
The second experiment for the ConvNet architecture sought to analyze how state-of-the-art technologies, in this case, batch normalization, affects winning tickets. This experiment was very similar to Figure4, the only difference being that the ConvNets used batch normalization on every convolutional layer. The main differences for the experiment, as shown in figures 5b, was that the randomly reinitialized networks did not drop as drastically as in the previous test, not once dropping below 55% for any network as compared to in figure 4b where all randomly reinitialized networks dropped well below 55%. Furthermore, the accuracies are consistently higher for the winning ticket initialized networks than in the previous experiment (as expected when using batch normalization). Using batch normalization, winning tickets were consistently found up until 1.8% of weight remaining (lowest tested density level) meaning it was a significant improvement not only over the experiments in figure 4b but also over the LeNet-300-100 architecture where the lower limit was roughly 4%. The amount of training epochs in figure 5a remained similar to those of experiment 4a.

In the final experiment of the project, the ConvNet winning tickets were tested using dropout regularization. The purpose of the experiment was to find out how network pruning would function with a regularizer that consistently drops parameters from the model and whether or not that would disturb the pruning process. The test accuracies for the dropout networks in figure6b were consistently higher than the networks without dropout but the density level patterns remained very similar. Once again, the Conv-4 and Conv-2 networks were able to find winning tickets all the way down to around 3.6% of weights remaining while the Conv-6 architecture dropped drastically at around 7%. The downside to using dropout was the extensive amount of training that was required as seen in figure6a.

Comparing the results of the three experiment with those of Frankle and Carbin[1], the most obvious difference is the behaviour of Conv-6 over the different density levels. In the original paper, Conv-6 consistently finds winning tickets up to at least 3.6% of weights remaining while the Conv-6 model accuracy in the experiments of this report starts decreasing much earlier, at around 7%. The is most likely a flaw due to the early stopping criterion used in the experiments of this report or due to minor architectural differences such as the number of strides in the max pool layers. Similarly to the LeNet experiments, the training time for the networks in the experiments of this report exceeded those of Frankle and Carbin by roughly double the number of training iterations for non-regularized networks.

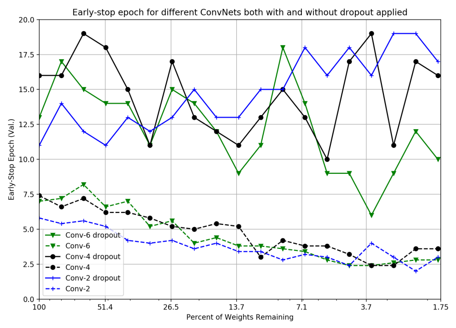


(a) Early stopping criterion for pruned ConvNets

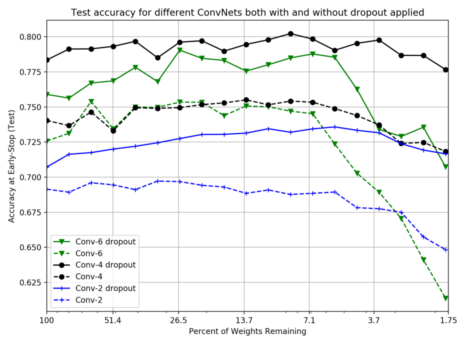


(b) Test accuracy for pruned ConvNets

Figure 5: Comparison of convergence and test accuracy achieved by ConvNets with batch normalization as regularisation for both randomly reinitialized and winning ticket initialized subnetworks. Each graph represents results received from the average of 5 trials.



(a) Early stopping criterion for pruned ConvNets with and without dropout applied



(b) Test accuracy for pruned ConvNets with and without dropout applied

Figure 6: Test accuracy and convergence of networks with and without dropout applied. The networks using dropout has a dropout rate of 0.25 as mentioned in section 4.1. Furthermore, each graph using dropout represents the results from a single trial.

6 Conclusion

All in all, the experiments in the report align with the original paper by Frankle and Carbin and show that the lottery ticket hypothesis algorithm consistently finds very dense winning tickets that outperform the original network both in terms of generalization capabilities and in terms of training performance, even when dropout regularization and batch normalization are applied. It is also important that the lottery tickets retain the initialization structure of the original network.

Although the results looked promising, the networks used in the experiments were fairly shallow. A future extension of the hypothesis would be to use deeper networks and other types of network architectures such as recurrent networks and to perform a more extensive hyperparameter and early stopping criteria search for the convolutional networks.

References

[1] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *arXiv:1803.03635 [cs]* (Mar. 2019). arXiv: 1803.03635. URL: `http://arxiv.org/abs/1803.03635` (visited on 04/16/2020).

[2] *Magnitude-based weight pruning with Keras*. en. Library Catalog: www.tensorflow.org. URL: `https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras` (visited on 05/16/2020).

[3] *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. URL: `http://yann.lecun.com/exdb/mnist/` (visited on 05/17/2020).

[4] *CIFAR-10 and CIFAR-100 datasets*. URL: `https://www.cs.toronto.edu/~kriz/cifar.html` (visited on 05/17/2020).

[5] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". en. In: (1998), p. 46.

[6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: `http://arxiv.org/abs/1409.1556` (visited on 05/14/2020).

[7] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Mar. 2015). arXiv: 1502.03167. URL: `http://arxiv.org/abs/1502.03167` (visited on 05/17/2020).

[8] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". en. In: (), p. 8.

[9] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[10] Davis Blalock et al. "What is the State of Neural Network Pruning?" In: *arXiv:2003.03033 [cs, stat]* (Mar. 2020). arXiv: 2003.03033. URL: `http://arxiv.org/abs/2003.03033` (visited on 05/07/2020).

[11] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H S Torr. "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY". en. In: (2019), p. 15.

[12] Michela Paganini and Jessica Forde. "On Iterative Neural Network Pruning, Reinitialization, and the Similarity of Masks". In: *arXiv:2001.05050 [cs, stat]* (Jan. 2020). arXiv: 2001.05050. URL: `http://arxiv.org/abs/2001.05050` (visited on 05/07/2020).

[13] Hao Li et al. "PRUNING FILTERS FOR EFFICIENT CONVNETS". en. In: (2017), p. 13.

7 Code Repository

Implementation and experiment code is available on our GitHub repository:
`https://github.com/Taoudi/LotteryTicketHypothesis`