Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Achieving results with untrained neural networks using Supermasks

Independent Study Proposal
im Arbeitsbereich Knowledge Technology, WTM
Prof. Dr. S. Wermter

Department Informatik
MIN-Fakultät
Universität Hamburg

vorgelegt von
Vincent Rolfs
am
TODO TODO TODO

Gutachter: Prof. Dr. S. Wermter
Dr. M. Kerzel

Vincent Rolfs
Matrikelnummer: 6789106

Contents

1 Introduction **1**

2 Approach **2**
2.1 Main idea 2
2.2 Choosing the threshold 3
2.3 Summary of steps 3

3 Evaluation **4**
3.1 Dataset 4
3.2 Baseline neural network 4
3.3 Training and early stopping 4
3.4 Supermasks 6
3.5 Lottery tickets and hyperoptimization 7

4 Conclusion **8**

References **9**

1 Introduction

Interest in decreasing the size of neural networks that are used to solve certain problems has existed since at least the year 1990 [1]. Different methods have been proposed that can be used to achieve a decrease in network size of up to 90% while retaining performance [1] [4] [5] [7].

This has multiple advantages, which can be classified according to the type of pruning that is applied. If the network is pruned after training while otherwise keeping the trained weights constant, this may result in reduced storage size, less energy consumption and faster computation during the application phase [3]. If it is possible to retrain the smaller network from scratch, this may result in less overfitting because the number of parameters has decreased [4]. And if a smaller but still effective network topology can be chosen *before training*, this can reduce the training time, because less parameters have to be optimized.

However, conventional pruning techniques have the problem that, when retraining from scratch on the smaller topology, the performance gets considerably worse [3]. These pruning techniques are only amenable for pruning the weights computed by an inital training run, and do not work well if the aim is to retrain once again on the smaller topology. Therefore, they cannot be used to choose a good topology before the training starts, because the correct parameters for the smaller network can only be discovered by training the larger network.

In contrast to this negative state of affairs, in their paper "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks" [3], the authors present a hypothesis that essentially claims that pre-training pruning nonetheless has a lot of potential:

The Lottery Ticket Hypothesis [3]. *A randomly-initialized, dense neural network contains a subnetwork that is initialized such that – when trained in isolation – it can match the test accuracy of the original network after training for at most the same number of iterations.*

The authors further show that the subnetwork mentioned in the hypothesis can be uncovered by training the dense network and then setting a percentage $p\%$ of the parameters with smallest magnitude to zero (and freezing them, so that they are not trained anymore). If the remaining parameters are then set to their inital values (the random values they had before the training started), and one retrains the smaller network starting with these initial values, then the training time will usually be less and the test performance better compared to the original, dense, trained network [3].

It is imperative here that the non-frozen parameters are reset to their *initial*, random values, not newly chosen random values. If the latter is chosen, the performance is much worse and the "Lottery Ticket" has not been uncovered [3]. In other words, we should not only search for efficient network topologies, but also for good initial values.

This curious fact has been further investigated by Zhou et al. [9]. They realized that the subnetworks obtained through pruning show a performance that is

already significantly higher than chance *before they are trained*. More specifically, a randomly-initialized, dense network is first trained. The trained parameters are then ranked by a value given by

$$\text{sign}\left(w_{initial}\right) \cdot w_{trained}$$

so that parameters with large magnitude that retained their sign are ranked high. A percentage $p\%$ of the lowest-ranking parameters is set to zero and frozen, the rest is reset to a constant with the same sign as their initial value. The resulting network is **not** trained again, rather it is being evaluated directly. Note that the values of the parameters at that point come directly from the initial random initialization, the only training information comes from deciding which of these random values to set to zero. Nonetheless, this technique achieves a remarkable 86% accuracy on MNIST, and 41% on CIFAR-10. The authors call the corresponding 0-1-masks "Supermasks".

In this paper, we provide multiple contributions to the new theory of Lottery Ticket pruning: We describe a complete methodology for applying Lottery Ticket pruning to a trained neural network. The methodology crucially uses the Supermask idea for finding the pruning threshold. We further show that the methodology applied on MNIST outperforms hyperoptimization of the network layer sizes: Our method can yield a significantly smaller size while retaining the same accuracy and training speed. And as part of this, we reproduce the main results regarding Lottery Tickets [3] and Supermasks [9] on MNIST: Supermasks do indeed yield untrained networks with higher-than-random accuracy, which can be trained to achieve faster training and higher accuracy with less parameters, compared to the unpruned networks.

2 Approach

In this section we describe a complete methodology for applying Lottery Ticket pruning to a trained neural network. We assume that we have a fully connected multilayer perceptron which has already been trained to satisfaction and should be pruned now. Importantly, we also assume that the weights with which the network had been initialized before training have been saved.

2.1 Main idea

Following the Lottery ticket approach, we want to take our initial network parameters, set some of them to zero and freeze them there, and then train the resulting network. The result will effectively have a (much) smaller size. There are different methods for choosing which of the initial parameters to prune: In the original Lottery ticket paper [3], this is done by setting all parameters to zero which had a small magnitude after training. How many parameters are set to zero depends on the specific threshold – a higher threshold corresponds to more aggressive pruning. For example, suppose the initial parameters are $0.1, 0.2, 0.3$ and the parameters

after training are $-0.2, 0.05, 0.3$. If we prune with a threshold of $t = 0.1$, we would get the parameters $0.1, 0, 0.3$. Starting with these values, we would then train the network, but the second value would be frozen at 0.

In [9], the authors show that instead of setting those parameters to zero whose magnitude after training is minimal ($|w_{trained}|$ is minimal), one gets better results by setting those parameters to zero for which the value

$$\text{sign}\,(w_{initial}) \cdot w_{trained}$$

is minimal. In other words, we keep those values which both have a large magnitude after training and retain their sign, and set the other parameters to zero. Following the example above, if the initial parameters are $0.1, 0.2, 0.3$ and the parameters after training are $-0.2, 0.05, 0.3$, then pruning with a threshold $t = 0.1$ yields $0, 0, 0.3$ – in this case, the first value is pruned as well, because it changed signs. In any case, it is necessary to choose a good threshold to apply the method effectively.

2.2 Choosing the threshold

We will describe a way of choosing a good pruning threshold based on another crucial insight from [9]. It is based on the following observation: When choosing the parameters as outlined above, the result is just a version of the initial, random parameters with some set to zero. However, without further training, the resulting networks nonetheless already show a suprisingly good performance when evaluated. This is where the term Supermask comes from: We take the initial, random parameters and multiply them with a mask of zeros and ones, and suddenly we get suprisingly great performance. For example, in [9], the authors report an accuracy of 86% on MNIST.

This leads us to a straightforward way of choosing a threshold: We simply compute the masked networks for different thresholds, and then (without training!) check their performance on the validation set. We then choose the threshold that yielded the highest performance. More to the point, we choose the masked network corresponding to that threshold, and train it to yield our pruned network.

This is especially useful since both the computation of the masked networks and the evaluation of them are computationally cheap, since no training is required.

2.3 Summary of steps

The complete recipe for applying Lottery Ticket pruning looks like this:

1. Take a fully connected multilayer perceptron, initialize it with random values and save these initial values.
2. Train the network to satisfaction.
3. For many different thresholds t, compute the parameters for a masked network M_t using the formula

$$w_i^{\text{new}} := \begin{cases} w_i^{\text{initial}} & \text{if } \text{sign}\left(w_i^{\text{initial}}\right) \cdot w_i^{\text{trained}} \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

4. Evaluate all the networks M_t on a validation set and choose the best performing network M_*.
5. Freeze all the parameters of M_* which are exactly zero, so that they don't change during training.
6. Train M_* to satisfaction.

3 Evaluation

In order to evaluate the outlined methodology, we will apply it to a standard dataset ten times using ten different random seeds. For each of these runs, we also perform a standard hyperoptimization grid search for layer size reduction as a comparison. As part of this evaluation, we reproduce the main results regarding Lottery Tickets [3] and Supermasks [9].

3.1 Dataset

All experiments are done with the well-established MNIST dataset [2]. It contains a collection of grayscale images of size 28×28, which each correspond to exactly one of ten possible digits 0 to 9. The dataset has already been split into a training and a testing part; it contains 60.000 training examples and 10.000 testing examples. We are using the training examples for training the network, and the testing examples for validation of the network performance during training.

3.2 Baseline neural network

In order to apply the pruning techniques and supermasks, we need a baseline neural network. As in [9], we chose a fully connected neural network with three layers. The input size is $28 \times 28 = 784$, the first layer has size 200, the second has size 30 and the output layer has size 10, since there are ten classes in the dataset. Compared to [9] we chose smaller values for the first and second layer size: In that paper, the sizes are 300 and 100 respectively. This was done because early experimentation with simple hyperoptimization showed that we could achieve the same performance with our smaller architecture during training. Since we are testing pruning capabilities, it is good to start with a network size that cannot be reduced trivially.

3.3 Training and early stopping

Since we need a meaningful trained baseline that does not overfit, we use the same early stopping criterion as in [9]. After saving the initial, random parameters, the neural network is first trained for 20.000 iterations, using the Adam optimizer [6] with learning rate 0.0012 (as in [9]) and the PyTorch cross entropy loss [8]. Each iteration corresponds to the processing of one batch of training data; one batch

contains 60 training examples. This initial training certainly leads to overfitting, which can be seen by plotting the validation loss (Figure 1).

Validation loss during initial training without early stopping

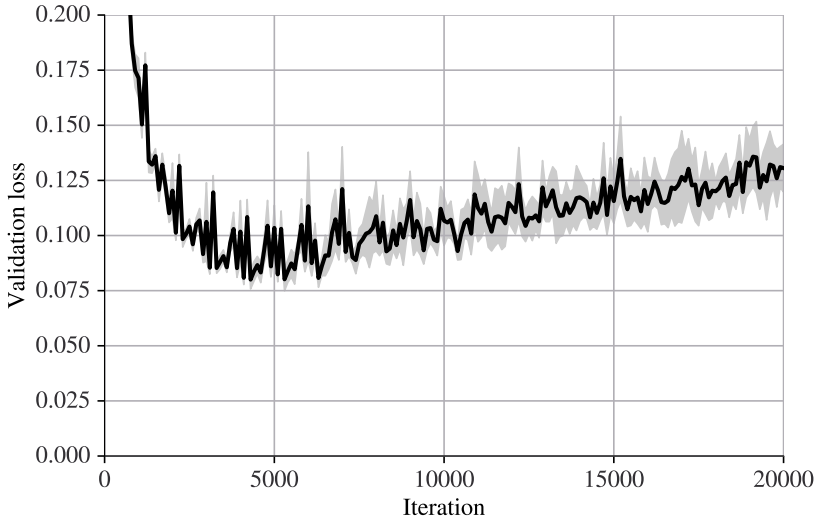

Figure 1: The validation loss during the initial training run, evaluated every 100 iterations across all ten random seeds. The indicated intervals show the standard deviation. Overfitting is clearly apparent: After about iteration 5000, the validation starts increasing again.

Because of this, we record at which iteration the minimal validation loss was achieved, and retrain the network from scratch – starting from the saved, initial parameters – for that many iterations. For all random seeds, the minimal validation loss is always achieved around iteration 5000. While the training will never go exactly the same the second time around due to randomness, this seems to work well in practice both in our experiments and in [9], yielding a trained, non-overfit baseline network. The validation loss during the second training run is visualized in Figure 2.

Across random seeds, the validation accuracy is 0.101 ± 0.016 before training and 0.977 ± 0.001 after training, where the $\pm$ sign denotes the standard deviation.

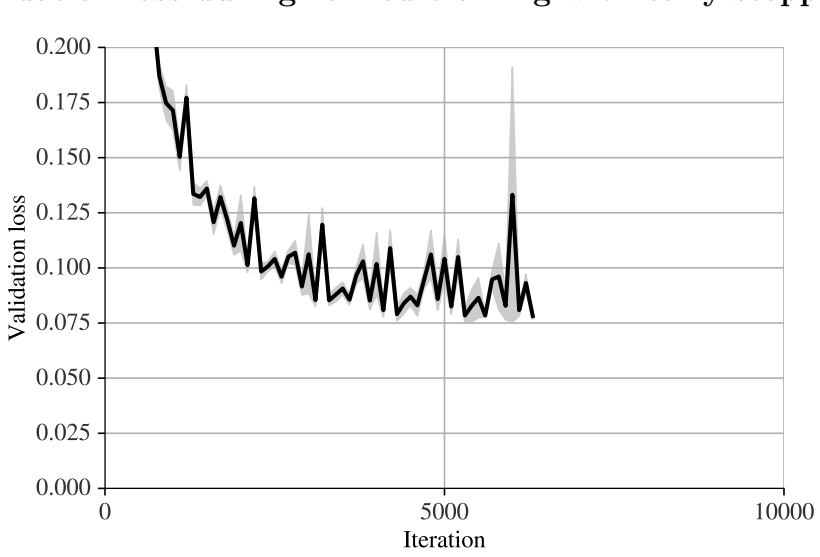

Figure 2: The validation loss during the second training run with early stopping, evaluated every 100 iterations across all ten random seeds. The indicated intervals show the standard deviation. For each seed, the training was stopped at the iteration that achieved the minimal validation loss during the initial training run. We can see that this approach results in a non-overfit, small validation loss when the training finishes.

3.4 Supermasks

We are now at step three of our methodology: The computation of the masked networks. To do this, we consider thresholds t from the set $\{0, 0.01, 0.02, \ldots, 0.2\}$. As described above, we then take the initial parameters of the network, and set some of them to zero, using the formula

$$w_i^{\text{new}} := \begin{cases} w_i^{\text{initial}} & \text{if } \operatorname{sign}\left(w_i^{\text{initial}}\right) \cdot w_i^{\text{trained}} \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

The exact size reduction achieved by choosing a particular threshold depends on the initialization, but the variance is low accross seeds: $t = 0$ results in a relative size of $64.6\% \pm 0.4$, while $t = 0.2$ results in $4.1\% \pm 0.6$. The size for different thresholds is visualized in Figure 3.

With step four, we now choose the threshold which resulted in the highest validation accuracy, before any further training is done. Here, the variance is very high across seeds. The ideal threshold is found to be at 0.049 ± 0.049, resulting in a relative network size of $38.7\% \pm 17.9$. The resulting accuracy of these masked and otherwise untrained networks is $42.6\% \pm 15.2$, where the smallest value observed is 19.1% and the largest is 70.0%. This astonishing fact marks our first reproduced result: As claimed in [9], the strategy of masking the initial, random values

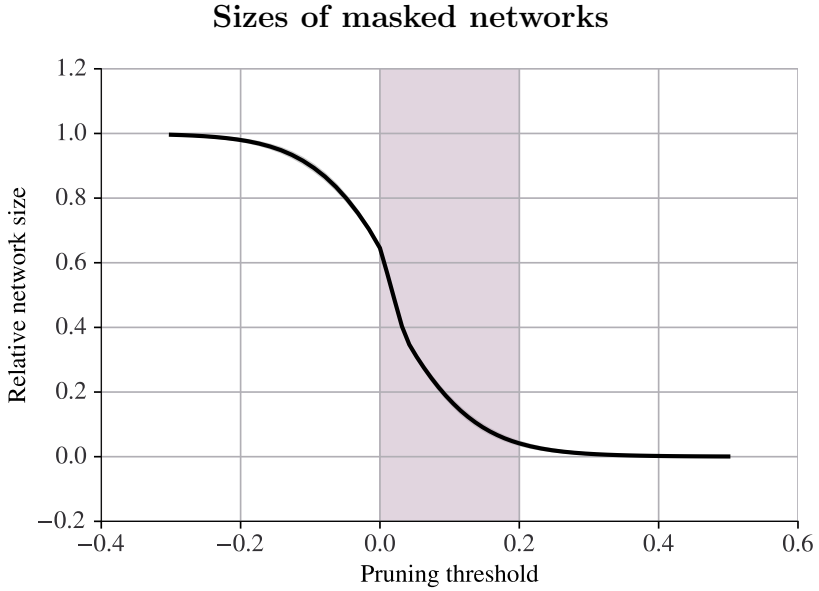

Figure 3: The relative network sizes after pruning with different thresholds. The variance across seeds is too small to be visible. The colored area indicates the thresholds that are considered as part of the evaluation of our methodology. The other parts of the graph are shown to visualize the convergence behaviour.

does indeed lead to a high accuracy, even though there is no explicit training of the masked networks involved. This is shown in Figure 4, where the accuracies obtained by pruning with different thresholds are visualized across seeds.

3.5 Lottery tickets and hyperoptimization

After having chosen a threshold using the Supermask evaluation, it is now time to train the corresponding masked network and thus create the "Lottery Ticket". This corresponds to steps 5 and 6 of our methodology. In order to have a good comparison for the newly created network's performance, we also use a simple hyperoptimization grid search which tries out smaller values for the baseline network's layer sizes. While the baseline uses sizes of 200 and 30 neurons for the first and second layers respectively, the hyperoptimization additionally considers all possible combinations of $(50, 100, 150)$ for the first layer and $(5, 15, 25)$ for the second layer. We are thus comparing the Lottery Ticket network to $1 + 3 \cdot 3 = 10$ other architectures.

To create the lottery ticket, we take the best-performing masked network from the previous step and train it for 5000 iterations. We also train the ten different layer size configurations for 5000 iterations, starting from random initializations. The validation loss for all these training runs is visualized in Figure 5 accross all seeds. We have grouped the different hyperoptimization into three groups by visually gauging their performance according to the graph. The graph shows that the lottery ticket trains at least as quickly and effectively as all the hyperoptimization

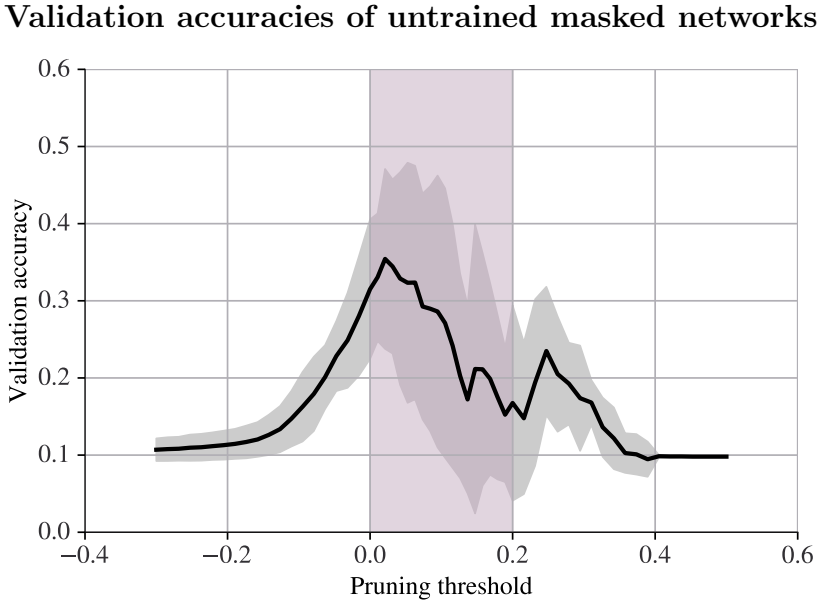

Figure 4: The validation accuracies obtained when evaluating the untrained, masked networks for different thresholds, aggregated across all ten seeds. The indicated intervals show the standard deviation. Even though the masked networks were never trained explicitly, they reach an accuracy that is significantly better than chance. The colored area indicates the thresholds that are considered as part of the evaluation of our methodology – the rest of the graph serves to illustrate that the accuracy indeed converges to 0.1 for very small or large thresholds, as predicted by the theory.

approaches. After 5000 iterations, the lottery ticket achieves a validation loss of 0.091 ± 0.007 while the original baseline network architecture (without any layer size reduction) achieves 0.109 ± 0.019. The "High performer" group of hyperoptimizations as a whole achieves a final validation loss of 0.106 ± 0.016.

At the same time, the Lottery Ticket networks that were found by our method are generally smaller (pruned more aggressively) than the results from the hyperoptimization. This can be seen in Figure 6, which shows the relative network sizes of all the resulting networks. As described in the previous section, the optimal Supermasks, and therefore also the Lottery Tickets, have a relative network size of $38.7\% \pm 17.9$. Thus, our method produces networks that are comparable in size to the low-performing hyperoptimizations, while they tend to outperform even the high-performing hyperoptimizations. The detailed data for the validation losses and sizes of all the different architectures can be found in Table 1.

4 Conclusion

Todo.

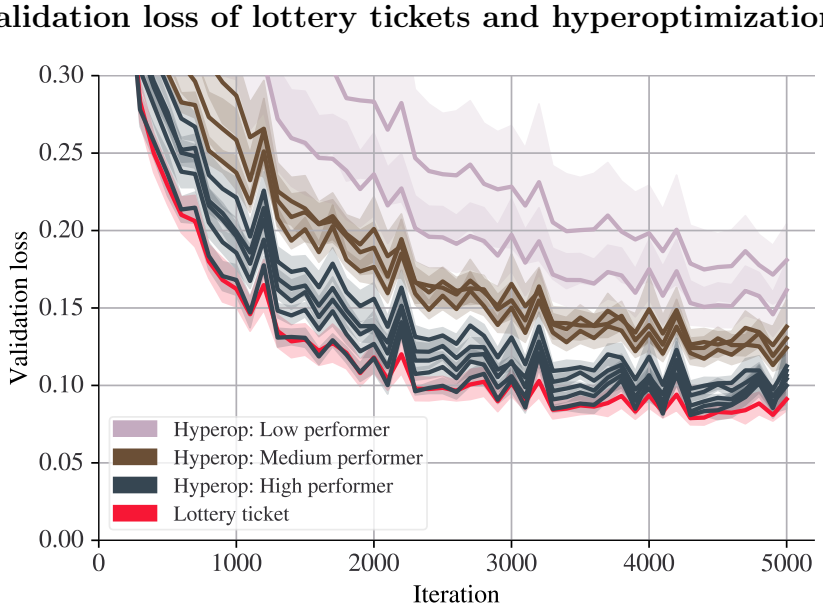

Figure 5: The validation loss during training of Lottery Tickets and hyperoptimized networks, evaluated every 100 iterations across all ten random seeds. The indicated intervals show the standard deviation. The hyperoptimized networks were visually grouped into three categories based on their training performance. It is apparent that the performance of the Lottery Tickets is at least as good as all other hyperoptimizations.

References

[1] Yann Le Cun, John S. Denker, and Sara A. Solla. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 598–605. ISBN: 1558601007.

[2] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[3] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019.

[4] Song Han et al. "Learning both Weights and Connections for Efficient Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: `https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf`.

[5] Babak Hassibi and David Stork. "Second order derivatives for network pruning: Optimal Brain Surgeon". In: *Advances in Neural Information Processing Systems*. Ed. by S. Hanson, J. Cowan, and C. Giles. Vol. 5. Morgan-

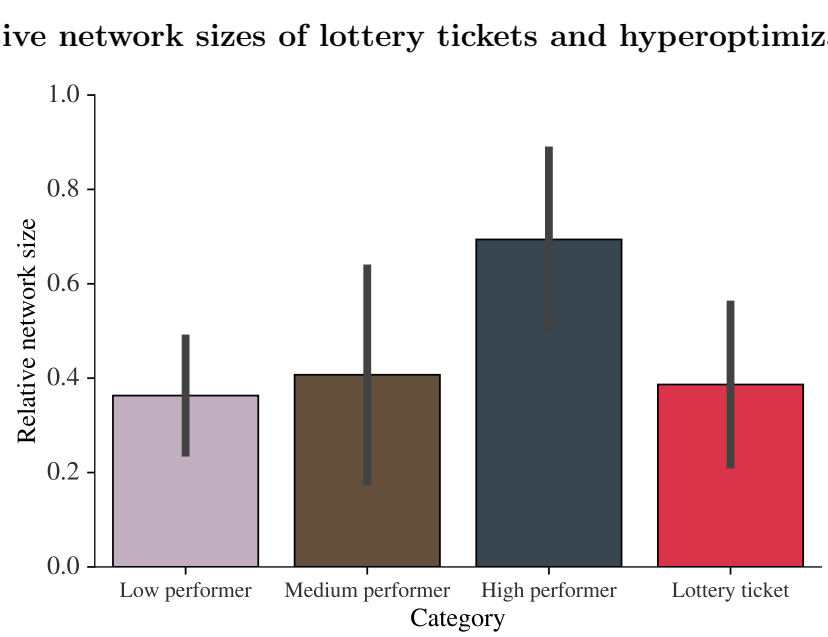

Figure 6: The relative network sizes of the Lottery Tickets and the three groups of hyperoptimizations. The indicated intervals show the standard deviation. We can see that the sizes of Lottery Tickets are similar to the low and medium performers, indicating that the method prunes rather aggressively while achieving a good performance as shown in Figure 5.

Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

[6] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[7] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *International Conference on Learning Representations*. 2017.

[8] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[9] Hattie Zhou et al. "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 3597–3607.

Label	**Category**	**Relative size**	**Final validation loss**	**Minimal validation loss**	**Iteration of minimal validation loss**
Lottery ticket	Lottery ticket	$38.7\% \pm 17.9$	0.091 ± 0.007	0.078 ± 0.004	4340.0 ± 222.1
Hyperop (200, 30) [baseline]	High performer	100.0%	0.109 ± 0.019	0.078 ± 0.001	4360.0 ± 171.3
Hyperop (100, 15)	High performer	49.1%	0.112 ± 0.015	0.095 ± 0.005	4600.0 ± 326.6
Hyperop (100, 25)	High performer	49.8%	0.104 ± 0.008	0.090 ± 0.003	4540.0 ± 340.6
Hyperop (150, 15)	High performer	73.6%	0.106 ± 0.021	0.085 ± 0.003	4660.0 ± 275.7
Hyperop (150, 25)	High performer	74.6%	0.100 ± 0.012	0.082 ± 0.003	4330.0 ± 249.7
Hyperop (50, 15)	Medium performer	24.6%	0.130 ± 0.010	0.119 ± 0.006	4700.0 ± 326.6
Hyperop (50, 25)	Medium performer	25.0%	0.124 ± 0.012	0.113 ± 0.004	4870.0 ± 94.9
Hyperop (150, 5)	Medium performer	72.6%	0.138 ± 0.017	0.122 ± 0.010	4640.0 ± 231.9
Hyperop (100, 5)	Low performer	48.4%	0.161 ± 0.019	0.146 ± 0.014	4870.0 ± 94.9
Hyperop (50, 5)	Low performer	24.2%	0.181 ± 0.023	0.170 ± 0.020	4770.0 ± 258.4

Table 1: Comparison of performance and size between the Lottery Tickets and the hyperoptimized networks. "Hyperop (a, b)" indicates a network architecture with first layer size a and second layer size b. All networks were trained for 5000 iterations. The data indicates that the Lottery Tickets reach a better performance while having a smaller sizes than the hyperoptimized networks.