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
09.11.2020

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

Vincent Rolfs
Matrikelnummer: 6789106

# Contents

# 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 [**braindamage**]. Different methods have been proposed that can be used to achieve a decrease in network size of up to 90% while retaining performance [**braindamage**] [**learning-weights-connections**] [**brainsurgeon**] [**pruning-filters**].

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 [**lottery**]. If it is possible to retrain the smaller network from scratch, this may result in less overfitting because the number of parameters has decreased [**learning-weights-connections**]. 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 [**lottery**]. 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" [**lottery**], the authors present a hypothesis that essentially claims that pre-training pruning nonetheless has a lot of potential:

**The Lottery Ticket Hypothesis [lottery].** *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 [**lottery**].

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 [**lottery**]. 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. [**supermask**]. They realized that the subnetworks obtained through pruning show a performance that is 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 describe a method for using the Supermask approach as a tool for choosing the pruning treshold during lottery ticket pruning. As part of this, we reproduce results from both [**lottery**] and [**supermask**], showing that 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. Furthermore, we compare our method to a simple hyperoptimization approach for decreasing the network size, and show that our method can yield a significantly smaller size while retaining the same accuracy and training speed.

# 2 Approach

Our basic setup follows closely the one in [**supermask**].

## 2.1 Dataset

All experiments are done with the well established MNIST dataset [**mnist**]. It contains a collection of grayscale images of size $28 \times 28$, 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.

## 2.2 Baseline neural network

In order to apply the pruning techniques and supermasks, we need a baseline neural network. As in [**supermask**], 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 [**supermask**] we chose a smaller value 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.

## 2.3 Training and early stopping

Since we need a meaningful trained baseline that does not overfit, we use the same early stopping criterion as in [**supermask**]. After saving the initial, random parameters, the neural network is first trained for 20.000 iterations, using the Adam optimizer [**adam**] with learning rate 0.0012 (as in [**supermask**]) and the PyTorch cross entropy loss [**pytorch**]. 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, which is evaluated every 100 iterations.

Because of this, we then record at which iteration the minimal validation loss was achieved. For all random seeds, this always happens around iteration 5000. We then retrain the network from scratch – starting from the saved, initial parameters – for that many iterations. 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 [**supermask**], yielding a trained, non-overfit baseline network.

The whole training procedure is performed ten times using different random seeds. Before training, the validation accuracy is $0.101 \pm 0.016$. After training, the validation accuracy is $0.977 \pm 0.001$.

## 2.4 Training and early stopping

The next step of our methodology is the application of the supermasks. To that end, we apply the following procedure taken from [**supermask**]. First, we reset the baseline network to its initial parameters, which were randomly initialized, but saved before the training started. Next, we choose a treshold $t$ and iterate through the parameters $w_1, \ldots, w_n$ of the network. We set a new value for each parameter according to the formula

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

In effect, we keep parameters the same which had a large magnitude after training, and still had the same sign as the initial value. The other parameters are set to zero. The resulting neural network is identical to the initial, randomly

initialized network, with some parameters set to zero. Effectively, a zero-one mask was applied to the initial, untrained network – the so-called Supermask.

We then evaluate the accuracy of this masked network on the validation set. Of course, the performance depends on the threshold: For very large thresholds, all parameters are set to zero, so the accuracy will be close to 0.1. For very small (negative) thresholds, all parameters are kept at their initial value, which was randomly chosen at the beginning, so we still expect an accuracy close to 0.1. For intermediate tresholds, something remarkable happens: The masked network achieves a validation accuracy that is significantly better than 0.1, even though all the parameter values come from a random distribution, and the only training information lies in the decision which parameters to set to zero. The performance for different tresholds of the masked networks across all ten random seeds are shown in XXX.

## 2.5 Lottery tickets

The next step is to apply the knowledge we have about the performance of the masked networks. We wish to find a "Lottery Ticket" – this is essentially a masked network which has been trained, where the parameters that are zero are frozen during training. When trying to prune a network using lottery tickets, one is faced with the one question that always arises during pruning: How small should the resulting network be? Here, we use the validation results of the supermasks as a guideline: Out of all the masked networks resulting from different tresholds, we choose the masked network that has the highest validation accuracy before training. We then train this masked network for 5000 iterations and observe the validation loss.

## 2.6 Hyperoptimization

In order to have a comparization for the quality of the pruning produced by the lottery ticket approach, we employ a simple hyperoptimization scheme to optimize the network size. Whereas the initial sizes for layers 1 and 2 were 200 and 30 respectively, for the hyperoptimization we allow all combinations of $50, 100, 150$ and $5, 15, 25$ respectively. For each of these size combinations, we take a randomly initialized network of that size, and train it for 5000 ietrations, observing the validation loss.

# 3 Conclusion