An Analysis of Neural Network Pruning in Relation to the Lottery Ticket Hypothesis

M.J. Havinga R.S. Sawhney

Abstract—In this paper, we analyze the novel "lottery ticket" approach for pruning neural networks and compare it to traditional techniques on dense feed-forward neural networks by the means of identifying "winning tickets". We explore the precise conditions required to identify and make use of these winning tickets and explore perspectives on the hypothesis from other work. We conclude that there is demonstrable evidence in support of the hypothesis, but revealing winning tickets in any neural network remains a hard task. We then discuss the continued research in this topic and suggest some research directions of our own.

Index Terms—neural networks, lottery ticket hypothesis, network pruning, network compression, architecture search

1 Introduction

Neural network pruning refers to the process of removing weights from (dense) neural networks with the goal of improving efficiency in terms of energy and memory consumption, as well as to simplify the model [5, 21]. An early example of neural network pruning is named *Optimal Brain Damage* and was proposed in 1990 by LeCun et al. [14]. This method takes a reasonably sized network and selectively deletes half of the weights based on a special metric, *saliency*, in order to come up with a sub-network which could perform just as well if not better than the main network. Many more methods for pruning have since been proposed as illustrated in Sec. 2.1. The common goal of these methods is the reduction of the number of weights in the model, while sustaining an approximately equal model accuracy. As we will see later, pruning can also improve the accuracy. Figure 1 shows an illustration of pruning in general.

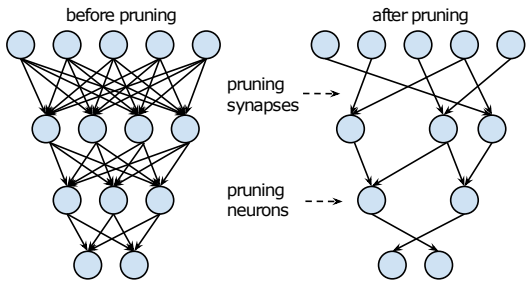

Fig. 1: Illustration of unstructured pruning. Weights are removed by the pruning algorithm based on some metric resulting in a sparser network. Taken from [5], page 3.

In general, pruning is performed *while* training the network. The resulting model is already trained at that point and reduced in the number of parameters. An open question remains whether it is possible to train a network from scratch, once it has already been pruned to a sparse architecture [15, 5]. So far, It has been shown that training a pruned neural network again from a new random initialization is a hard task. When re-initializing such a sparse network with random weights it generally achieves a lower accuracy than the original trained network. For example, Li et al. noticed this in [15] and claimed a "difficulty of training a network with a small capacity". As can be seen in Table 1 in [15], pruned models that have been retrained from scratch perform systematically worse than their pruned and unpruned original model.

M.J. Havinga and R.S. Sawhney are students at the University of Groningen, Computing Science, Master programme.

If such a sparse network is hard to train, why has it proven successful to train an over-parameterized network and prune it until a sparse network remains? Han et al. explain this in [5] through the existence of "fragile co-adapted features" in the network as described in [24]. Essentially, it is implied that the configurations of the weights co-depend and can not be trivially re-initialized and retrained. Following up on these findings, Frankle and Carbin proposed the *lottery ticket hypothesis* and an accompanied conjecture, stated as follows.

Hypothesis. *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.*
Conjecture. *Stochastic gradient descent seeks out and trains a subset of well-initialized weights. Dense, randomly-initialized networks are easier to train than the sparse networks that result from pruning because there are more possible subnetworks from which training might recover a winning ticket.*

Frankle and Carbin (2019) in [3]

From this point forward, we will refer to both the hypothesis and the accompanied conjecture as "the lottery ticket hypothesis". In this paper, we explore this hypothesis. We task ourselves with investigating the following questions.

1. What is the precise meaning and implication of the lottery ticket hypothesis?
2. Can we find experimental evidence of the lottery ticket hypothesis and if so under which conditions?
3. How can we explain the evidence supporting the lottery ticket hypothesis and with which mechanisms?

To be able to adequately answer these questions, we will provide some background information in Sec. 2. After this, we will discuss the methods and experimental results found by Frankle and Carbin [3], Liu et al. [16] and ourselves in Sec. 3 and 4. In our discussion in Sec. 5, we implicitly answer our research questions. Finally in Sec. 6, we suggest some directions for future research in this topic.

2 Background

As briefly explained previously, pruning refers to process of removing weights from a (dense) neural network. Christopher Bishop [2] describes pruning as a method for finding optimal network architectures, as a counterpart to network growing which adds weights rather than remove them. On the other hand, Simon Haykin [6] describes pruning as a regularization method for neural networks, i.e. to improve the model's generalization abilities. Sec. 2.2 further analyzes pruning in this context. We will see that it is reasonable to regard pruning

as both a method of architecture search and regularization. There are many different pruning methods using several completely different approaches. This large and diverse family of methods is illustrated in the next section.

2.1 Pruning methods

Pruning methods can be classified as either pre-defined by the user or automatically derived by the network. Another division of methods can be made based on whether the pruning algorithm takes the network architecture into account. Methods that do this are called structured methods, while methods that only consider individual weights are called unstructured methods.

Unstructured pruning

Unstructured pruning methods find individual weights that are least relevant and can be pruned. This method was used by Frankle and Carbin to perform their experiments for the lottery ticket hypothesis [3]. There is an important distinction between one-shot pruning and iterative pruning.

One-shot pruning. This is the basic type of unstructured pruning method which prunes the network as as function of the percentage of the weights [3]. It involves the following steps.

1. Randomly initialize a neural network.
2. Train the network.
3. Set p% of weights from each layer (based on some metric) to 0.

Iterative Pruning. This is similar to the one-shot pruning method. The major difference is that the weights are trained and pruned over n rounds. In each round, $p^{\frac{1}{n}}\%$ of the weights are removed. This is the main method used in [3].

To find winning tickets, Frankle and Carbin propose to reset the weights of the pruned network to their original random initialization [3]. If the found model is indeed a winning thicket, it can then be retrained to gain an equivalent accuracy as the unpruned model.

Pre-defined structured pruning

The methods which employ structured pruning involve removing the weights in the channels while keeping the target architecture in mind before performing the pruning. The way in which pruning is performed is set by the user and the manner is dictated by the pruning algorithm itself. These methods were employed by Liu et al. to counter the unstructured pruning methods performed by Frankle and Carbin [16]. This technique of pruning is briefly explained in Figure 2. The different pre-defined structured pruning methods as used in Liu et al.'s work are given below.

L1-norm based Filter Pruning. One of the earliest pruning methods on channel pruning for convolutional neural networks. In this method, the L1-norm for the filters is considered in each layer. A pre-defined percentage of filters with smaller L1-norm is eliminated. [15]

ThiNet. This method is greedy as compared to the L1-norm. It prunes the channel that has the least effect on next layer's activation values [17].

Regression Based Feature Reconstruction. The pruning is performed by reducing the error during reconstruction of feature map of the next layer. The optimization problem in this method is solved with the help of LASSO Regression. [8]

Automatic Structured Pruning

Unlike the pre-defined structured pruning methods, here the target architectures for the sub-network are derived automatically. Liu et. al. argue that training these models from scratch can lead to comparable

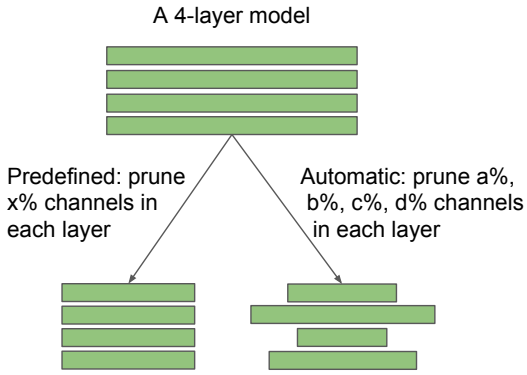

Fig. 2: A visualization for the pre-defined and automatic structured pruning method. Figure 2 taken from [16].

or even better performance, concluding that the architecture derived in these methods is much more important than the weights themselves. The different automatic pruning methods are given below.A brief methodology for this technique of pruning is explained in Figure 2.

Network Slimming. As described in [16], "imposes L1-sparsity on channel-wise scaling factors from Batch Normalization layers during training, and prunes channels with lower scaling factors afterward."

Sparse Structure Selection. This method is a generalization of the network slimming method as it also makes use of the sparsified scaling factors to be used for pruning. It has been shown to be applicable on the residual blocks in ResNet if required. [10]

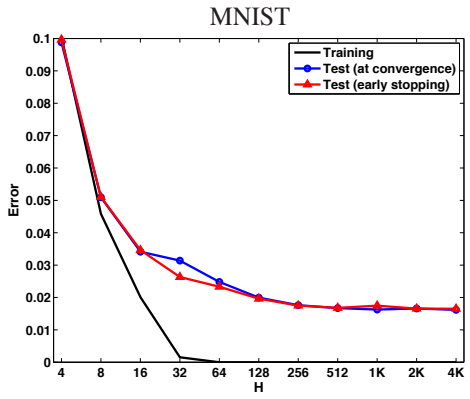

Fig. 3: Relationship between the training/test error and the size of a 2-layer neural network trained by momentum stochastic gradient descent. Here, H represents the network size in terms of the number of hidden units. Figure 1 taken from [20].

2.2 Pruning in relation to the generalization error

Neural network pruning has been observed to improve the generalization capabilities of the model. However, it is still debated how pruning achieves this [1]. Historically, the network's size in terms of the number of free parameters has been used as a metric of the complexity of a network [14]. Since increased complexity implies an increased risk of overfitting, it has been conjectured that the shrinking of a neural network reduces this risk and improves the generalization capabilities of the network. This theory however, has not been generally proven. In fact, contradictory evidence has been found for specific network architectures which implies that larger networks can generalize better than smaller networks [20]. As can be seen in Figure 3, a larger network always improves both the training and test error for this specific experiment. It has been suggested that it is the pruning method itself

from which the improved generalization arises. This theory, as postulated by Bartoldson et al. [1], states that the "destabilizing effects" of the pruning method allow the learning algorithm to leave potentially sub-optimal local minima.

3 Methods

To answer our research questions, we have evaluated the results found by Frankle and Carbin in support of their hypothesis. We also analyzed evidence found by others in relation to the hypothesis, specifically by Liu et al. [16]. In their work, Liu et al. make certain arguments against the findings of Frankle and Carbin. As we will see, the results of the experiments depend strongly on the precise conditions of the experiment. To help clarify this, we add results from our own experiments as extra evidence.
In most of the experiments, four neural network models can be distinguished.

1. The original network with dense fully connected layers.
2. The network after training and pruning down to a $p\%$ density. This is where classical pruning ends.
3. The pruned sparse network, re-initialized with new random weights and trained again.
4. The pruned sparse network, re-initialized with its original weights and trained again. This is the method proposed by Frankle and Carbin to reveal winning "lottery tickets".

In many of the experiments by Frankle and Carbin, Liu et al. and ourselves, the MNIST [13] and CIFAR10 [12] data sets are used. MNIST is a data set consisting of 60,000 training images (grayscale of handwritten digits of 28x28 resolutions. CIFAR10 consists of the same number of images, but of a slightly higher resolution (32x32), in color, and of 10 different real-life object classes such as cats and airplanes. CIFAR10 is arguably a larger and more complex data set than MNIST, which is important to take into account in these experiments.
To find winning tickets, Frankle and Carbin make use of convolutional neural network models with fully connected layers. They use optimization strategies such as stochastic gradient descent (SGD), momentum SGD and Adam [11] also making use of dropout [23, 9] and weight decay. The used pruning technique is mainly iterative unstructured pruning such that the winning tickets are sparse.
In [16], Liu et al. compare the performance of the pruned and the non-pruned network with the help of different methods of pruning as discussed in Sec. 2.1. They employ different techniques such as *Scratch-B* and *Scratch-E* which involve training the small pruned models for the same number of epochs (Scratch -E) or training over the same amount of computation budget (Scratch-B). If pruning the model decreases the number of FLOPs[1] by at least a factor of 2, the number of epochs is doubled. Note that since Liu et al. use automatic structured pruning methods, their approach differs fundamentally from Frankle and Carbin's approach.
In our own experiments, we use a small convolutional neural network applied to the MNIST data set. Our network is based on LeNet [13] and most notably contains three fully connected layers, two of which together make for nearly 95% of the network's parameters[2]. The model is trained using Adam [11]. With this, we recreate and train the four network models as mentioned before for different pruning factors p. We use iterative magnitude-based weight pruning to find our sparse networks. This simple method, contributed by Zhu and Gupta [26], prunes the weights of the lowest magnitudes until a desired sparsity is reached. All our experiments are implemented with the use of the Python programming language[3] using the libraries TensorFlow[4] and Keras[5] for implementing our neural networks.

[1] Floating point operations, a measure of computational load.

[2] These fully connected layers consist of 48,120 and 10,164 weights respectively. The full network consists of 61,706 weights.

[3] `https://www.python.org/`

[4] `https://www.tensorflow.org/`

[5] `https://keras.io/`

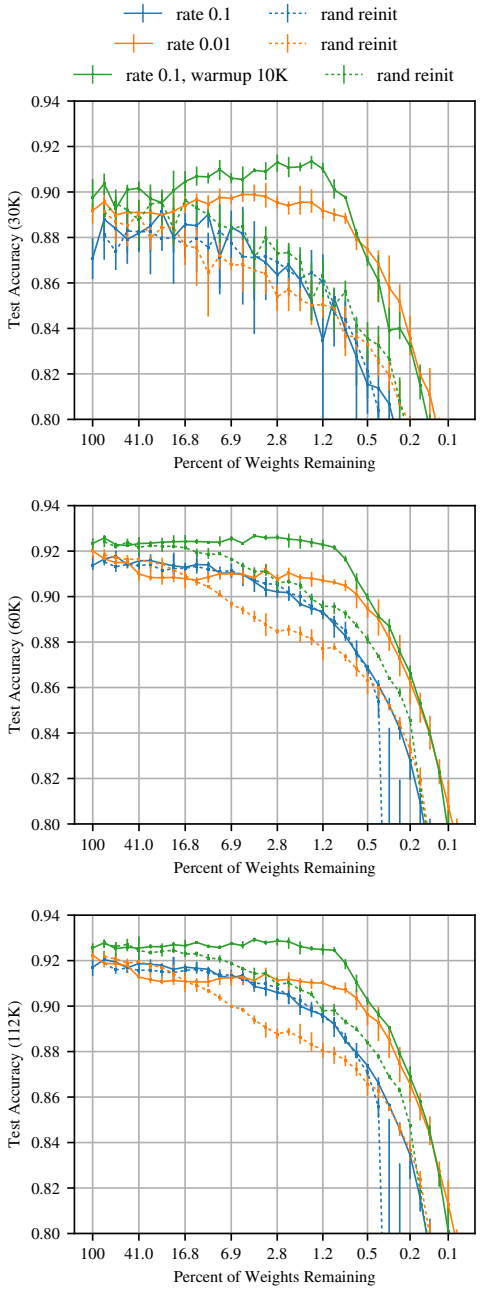

Fig. 4: Adapted from [3], page 7, Figure 7: "Test accuracy (at 30K, 60K, and 112K iterations) of VGG-19 when iteratively pruned."

4 Results

In this section, we give an overview of some of the results from different sources, as described in Sec. 3. We highlight only some results which we think are important, relevant or exemplary. Equipped with this knowledge, we will discuss our research questions in Sec. 5.

4.1 Experiments by Frankle and Carbin

The pruning technique performed on LeNet [13] is a simple layer-wise pruning heuristic. A certain percentage of weights with the lowest magnitudes in each layer is removed. Connections to output are pruned at half the rate compared to the rest of the network. Here, the MNIST data set is used for training on a LeNet 300-100 architecture. From these experiments, it is concluded that in the case of a fully connected network such as LeNet for a relatively small data set (such as MNIST), the initialization is significant for the performance of the winning ticket.

Using iterative pruning, the winning tickets are found to learn faster than the original network. The winning tickets are pruned iteratively to various extents. It is observed that a "winning ticket" with 51.3% of the weights remaining gives a higher test accuracy than the original network. When the density goes down to 21.1% it achieves this higher test accuracy faster, in other words the training early-stops earlier. At the point where the density becomes 3.6%, the winning ticket performs on par again with the original network.
It is to be noted that winning tickets optimize more effectively but do not generalize better in the case of an early stopping criterion. However at 50,000 iterations, the test accuracy still seems to improve, while the training error is already down to 0. This implies that winning tickets do generalize better.
When the model is randomly re-initialized, it is observed that for a pruning density between 51.3% and 21% the winning tickets learn at a slower rate in the case of random re-initialization and lose test accuracy after a little pruning, which supports the lottery ticket hypothesis. One-shot pruning helps to identify the winning tickets without repeated training. It is observed from the experiments that the iteratively pruned tickets learn faster and reach higher test accuracy at smaller network sizes. Frankle and Carbin emphasize the need for iterative pruning for their experiments on fully connected networks.
Moving on to slightly larger convolutional neural networks, Frankle and Carbin use scaled down VGG and VGG-19 networks [22] as well as ResNet [7] for experiments on the CIFAR10 data set [12]. The basic experiment is the same as in the case of LeNet. As the network is pruned, it tends to learn faster and the test accuracy is improved. Iterative pruning and random re-initialization for different layers of the ConvNet was performed. The results can be found in [3], page 6, Figure 5. The winning tickets seem to have a higher test accuracy. It is found that the gap between the test and training error is smaller for winning tickets implying that they generalize better. As in the case of LeNet, in ConvNets, the test accuracy is worse with random reinitialization of the winning tickets. However, the test accuracy at early stopping remains steady and may even improve for some layers in the network.
It is noted by Frankle and Carbin that using dropout [23, 9] helps increasing the test accuracy of the network. Here, dropout refers to updating only a random part of the network for each training iteration.
When training on larger VGG and ResNet networks, Frankle and Carbin found the architecture to be sensitive to the learning rate, and warmup was required to find the tickets at a higher learning rate. When using warmup, the learning rate is initially built up from 0 to the initial learning rate over the course of a specified number of iterations. In the case of a lower learning rate, the winning tickets seem to learn faster than the original network in the beginning. This however soon slows down due to the lower initial learning rate. On the other hand, they perform faster when randomly reinitialized. This can be improved by the introduction of the warmup strategy which in turn helps finding winning tickets at a higher learning rate.
Figure 4 shows an overview of results on VGG-19 when iteratively pruned and re-initialized, either randomly or with its original weights. From these graphs, it can be seen that the lottery ticket approach at an initial learning rate of 0.1 and with a 10K iterations warmup performs consistently better than the other approaches, and is able to maintain the same level of accuracy down to approximatly 1.2% of the weights remaining.

4.2 Experiments by Liu et al.

Liu et al. [16] claim that it is not necessarily better to reuse the original weights of the network. Instead, random re-initialization of the weights of the winning tickets can be enough to perform on par with the original unpruned network and the "lottery ticket" configuration
The main difference according to Liu et al. is the result when comparing unstructured pruning on CIFAR. For structured pruning, using either higher or lower learning rates, the winning ticket approach does not outperform the randomly re-initialized network.
In order to prove this, Liu et al. compare the models with Frankle and Carbin's approach as well as randomly re-initializing weights using initial learning rates of 0.1 and 0.01, also using a step-wise delay schedule and momentum SGD. These are then used on the CIFAR and ImageNet datasets, the latter of which is a larger dataset. The pruning methods used here are iterative pruning and one-shot pruning as used by Frankle and Carbin and comparing them with L1-norm based filter pruning. The values for the experiment are shown in Table 8 in [16].

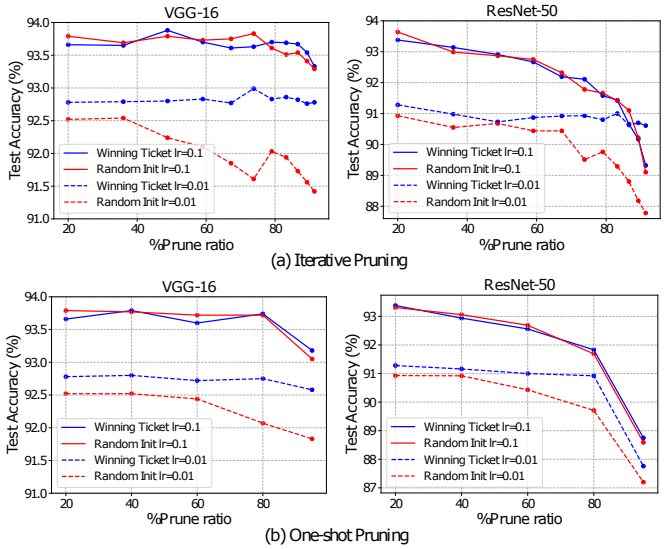

Fig. 5: From [16], page 11, Figure 7: "Comparisons with the Lottery Ticket Hypothesis [3] for iterative/one-shot unstructured pruning [5] with two initial learning rates 0.1 and 0.01, on CIFAR-10 data set."

From Figures 4 and 5 we observe that in the case of unstructured pruning, the winning ticket method of assigning the original weights to the subnetwork works better in comparison to the randomly reinitialized subnetwork if the learning rate is small. In the case of structured pruning, the winning ticket performs on par with the randomly reinitialized subnetwork.
Liu et al. argue that the reason why the winning ticket approach is helpful at low learning rate may be that the weights of the final trained model are not too far from original initialization due to the small parameter step-size.

4.3 Additional experiments

In our own experiments, we put the two main re-initialization approaches to the test. The test (validation) accuracies of our LeNet-based model as described in Sec. 3 are plotted in Figure 6. The accuracy of the fully connected model, i.e. without pruning whatsoever, is shown in the background as reference for the methods under investigation. The pruned network that was used to potentially find winning "lottery ticket" configurations is also regarded a model for reference here as it represents classical pruning. We see that for all pruning methods, the accuracy improves when pruning down to any density of 5% or higher. At 2% and 1% the network seems be too sparse to keep up with the accuracy of the original dense model. As for finding "lottery tickets", we see that only between densities 10% and 2% this approach clearly stands out from the rest. Presumably, a network that is still quite dense can easily find a good solution again with a new random initialization. "Aggressive" pruning seems to be required to make the winning lottery tickets reveal themselves. At the other extreme however, we see that at 1% density the difference between the two methods fades entirely. Both methods under-perform strongly here, which is not surprising because this level of pruning almost guarantees that information from the input is lost to some extent. Another interesting observation is that the pruning network seems to outperform all other models at the higher densities. We think this might be

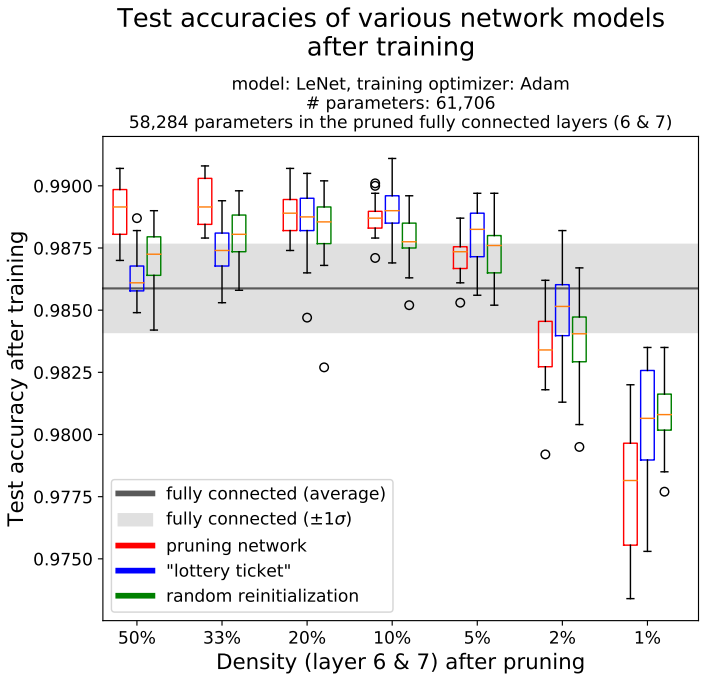

Fig. 6: Results of training and retraining our model for different levels of sparsity after pruning. The measured test accuracies are displayed as boxplots, where the box encompasses 50% of the measurements, with a line denoting the median within. The "whiskers" represent the full range of the data, except for outliers which are shown as separate circles.

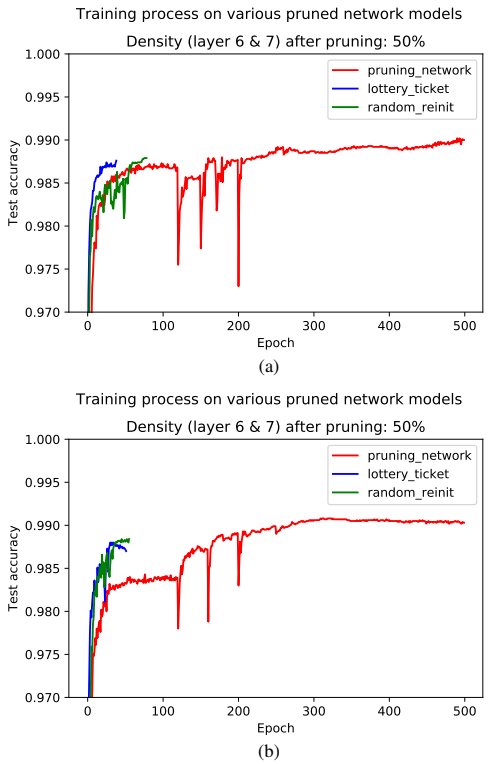

Fig. 7: Training process of the different pruned neural network models. Two instances of training a 50% pruned network were selected which showed interesting features as described in Sec. 4.3.

because the pruning network is not allowed to early-stop, while the re-initialized models are. The extra training on the pruning network may improve the test error, even though the training error is stable at 0. A similar observation was made by Frankle and Carbin (see Sec. 4.1) and we already discussed this general concept in Sec. 2.2. We can substantiate this claim by looking at the training curves in terms of the validation accuracy over the course of the training epochs. As can be seen in Figure 7a, it is possible for the test accuracy to increase in the pruning network over the course of many more epochs, while the re-initialized models are not able to do this because they early-stopped. In other cases, we observe that the pruning itself contributes to an improved test accuracy. As can be seen in Figure 7b, the pruning network benefits from the temporary disruption of the pruning iteration and reaches higher accuracies than it would have if no pruning would have been applied. This observation that pruning and the instability it generates may contribute to training was made earlier in [1] as mentioned in Sec. 2.2.

5 Discussion

Frankle and Carbin [3] observed that their approach of re-initializing weights with their original weights was more effective than re-initializing with random weights when using unstructured pruning. For large models and higher learning rates, the warmup technique was necessary to get better results than with random re-initialization. In case of the lower learning rates, the pruned network learnt faster initially but slowed down over the course of learning due to the small learning step.

Liu et al. [16] performed experiments to gain more evidence and observed that the accuracy achieved when the subnetwork is randomly reinitialized actually performs similar to that of the winning tickets. Thus it is implied that the structure of the network is important rather than the weights. They conjecture that this may occur because in Frankle and Carbin's experiments, the data set was small and the target weights were closer to the values in the original network. Liu et al. argue that the weights may introduce a large bias over the selection of the winning ticket.

Drawing conclusions from both authors and our own experiments, we can substantiate Frankle and Carbin's hypothesis with the found evidence. However, it is a challenging task to find the lottery tickets as described. For various neural network models of different sizes and complexities, more sophisticated learning methods are required. It is unresolved whether it is always possible to find winning tickets in any dense neural network, and if they exist as is claimed by Frankle and Carbin. As was claimed by Frankle and Carbin, it is essentially only possible to find lottery tickets beyond a certain level of sparsity. In other words, networks that remain too dense after pruning will not reveal lottery tickets. This is something that we also saw in our own experiments in Sec. 4.3.

So far, we have only seen evidence of applicability in relatively small networks and when using a relatively low learning rate or a warmup strategy. Indeed, applying the lottery ticket approach to larger and deeper networks has proven harder than for their smaller and more shallow counterparts. Frankle et al. recently published research [4] in which they are able to create well-initialized sub-networks in a deep Resnet-50 network, but only by re-initializing the weights to their values after they had already been trained for a short while. The required training to get the weights in this desired state is admittedly quite little, but it shows that it is significantly harder to apply the hypothesis to these networks.

The lottery ticket hypothesis has already generated a decent amount of follow-up research. This has helped to understand better by which mechanisms the observed effects of the winning tickets work. A notable finding by Zhou et al. [25] is that the process of pruning itself should be considered learning. This is well illustrated in Figure 8, where we see that a "lottery ticket" that was found with Frankle and Carbin's method performs far above the statistical expectation, even before any training. Merely selecting the lottery ticket from the over-arching dense network is therefore a form of training. Zhou et al. name these untrained lottery tickets "Supermasks". Malach et al. [18] worked on a similar theory: "a sufficiently over-parameterized neural network with random weights contains a subnetwork with roughly

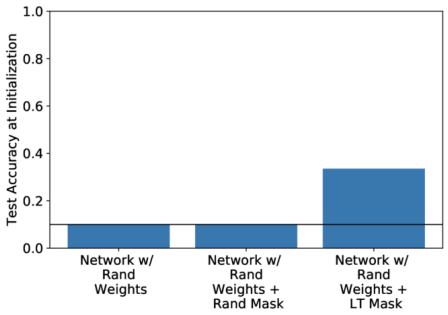

Fig. 8: Networks tested on the MNIST data set. The networks are untrained, i.e. have random weights, but a pruning mask is applied to the second and third model. Taken from `https://eng.uber.com/deconstructing-lottery-tickets/`, a web page about [25].

the same accuracy as the target network, without any further training." This can be thought of as a stronger version of the lottery ticket hypothesis, and Malach et al. claim to prove this stronger hypothesis in their paper. Finding such a perfect subnetwork however, is a computationally hard problem and only works in very over-parameterized models. This problem can therefore be interpreted as a different take on neural network training.

6 Suggestions for future work

The models on which the hypothesis has been applied are quite limited. The concepts contributed by Frankle and Carbin can be extended and applied to many other situations. An interesting example of this is the work by Rahul Mehta [19], who adapted the lottery ticket hypothesis to transfer learning, creating the "Ticket Transfer Hypothesis".
An open question remains the applicability of the hypothesis. In a practical setting, it is usually not necessary to re-initialize the network. Simple pruning and fine-tuning is sufficient to compress the model and achieve the associated benefits. It would be interesting to see more practical applications of the findings of Frankle and Carbin and those inspired by it.
As discussed in Sec. 5, we can interpret pruning as a novel form of neural network training. An important direction for future work should be to see how we can find ways to make use of both types of learning (i.e. "regular" weight optimization and pruning) in the most effective way to train models more efficiently and come up with better architectures.

Acknowledgements

The authors wish to thank Michael Biehl and Allessandro Pianese for reviewing this paper. We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

References

[1] B. R. Bartoldson, A. S. Morcos, A. Barbu, and G. Erlebacher. The generalization-stability tradeoff in neural network pruning. *CoRR*, abs/1906.03728, 2019. URL `https://arxiv.org/abs/1906.03728`. Accessed: February 2020.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.

[3] J. Frankle and M. Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL `http://arxiv.org/abs/1803.03635`. Accessed: February 2020.

[4] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611, 2019. URL `http://arxiv.org/abs/1903.01611`. Accessed: March 2020.

[5] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[6] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, Inc.,, Upper Saddle River, New Jersey 07458, USA, 3rd edition, 2009.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `https://arxiv.org/abs/1512.03385`. Accessed: February 2020.

[8] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. *CoRR*, abs/1707.06168, 2017. URL `https://arxiv.org/abs/1707.06168`. Accessed: February 2020.

[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL `https://arxiv.org/abs/1207.0580`. Accessed: February 2020.

[10] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. *CoRR*, abs/1707.01213, 2017. URL `https://arxiv.org/abs/1707.01213`. Accessed: February 2020.

[11] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*, 2015.

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, Toronto, 2009.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[14] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990. URL `http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf`. Accessed: February 2020.

[15] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL `https://arxiv.org/abs/1608.08710`. Accessed: February 2020.

[16] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. *CoRR*, abs/1810.05270, 2018. URL `http://arxiv.org/abs/1810.05270`. Accessed: February 2020.

[17] J. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. *CoRR*, abs/1707.06342, 2017. URL `http://arxiv.org/abs/1707.06342`. Accessed: February 2020.

[18] E. Malach, G. Yehudai, S. Shalev-Shwartz, and O. Shamir. Proving the lottery ticket hypothesis: Pruning is all you need, 2020. URL `https://arxiv.org/abs/2002.00585`. Accessed: March 2020.

[19] R. Mehta. Sparse transfer learning via winning lottery tickets. *CoRR*, abs/1905.07785, 2019. URL `https://arxiv.org/abs/1905.07785`. Accessed: March 2020.

[20] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[21] R. D. Reed and R. J. Marks. *Neural smithing: supervised learning in feedforward artificial neural networks*, chapter 13: Pruning Algorithms. MIT Press, Cambridge, Massachusetts, 1999.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014. URL `https://arxiv.org/abs/1409.1556`. Accessed: February 2020.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[24] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL `http://arxiv.org/abs/1411.1792`. Accessed: February 2020.

[25] H. Zhou, J. Lan, R. Liu, and J. Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *CoRR*, abs/1905.01067, 2019. URL `https://arxiv.org/abs/1905.01067`. Related web page: `https://eng.uber.com/deconstructing-lottery-tickets/`. Accessed: March 2020.

[26] M. Zhu and S. Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview, 2018. URL `https://openreview.net/pdf?id=Sy1iIDkPM`, Accessed: March 2020.