One Initialization to Rule Them All?

Stefan Wijnja, Ellis Wierstra, Thomas Hamburger

February 2, 2020

Abstract

As universal function approximators, neural networks have proven to be a useful tool in many areas of research and industry. However, the computational efficiency of deep learning leaves much to be desired. On this topic, Frankle and Carbin (2019) have presented the lottery ticket hypothesis, which states that on initialization, large networks contain weight combinations that make up subnetworks that can rival the larger versions in performance when trained in isolation. In this paper a comparison between different weight initialization algorithms is made to find out how a network's starting point affects the search for these subnetworks. Results point toward a pattern where drawing weights from Xavier-based, narrower distributions may outperform drawing from Kaiming-based, wider distributions on this task, while the difference in shape of a normal versus uniform distribution may not matter much. More research is needed to confirm or falsify these preliminary conclusions.

Contents

Introduction

As universal function approximators, neural networks have proven to be a useful tool in many areas of research and industry. To meet the challenge of modeling complex functions, network sizes – in terms of the number of parameters – have increased as well, and these large networks require significant amounts of computational power and time to train and use.

On this topic, Frankle and Carbin (2019) have presented the Lottery Ticket Hypothesis, which states:

> 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. (Frankle and Carbin 2019)

It has been shown before that by removing weights from a network in a structured way (*pruning*), network sizes can be reduced by up to 90% without loss in performance (Frankle and Carbin 2019: 1). The novelty here lies in the idea that a reduced network is not simply used as-is, but its weights are reset to the values post-initialization and then retrained.

Indeed, Frankle and Carbin (2019)'s research has shown that the reduced and retrained networks can be up to 96 percent smaller than the original, full-sized network without loss in test accuracy (Frankle and Carbin 2019: 4). This proves that training the full network is only relevant in *finding* the configuration of weights that make up the performant subnetwork – the winning ticket –, not in training and using it.

Naturally this leads one to wonder if there is a way to skip training the large network and initialize the winning ticket in one shot. To inch closer to this holy grail of pruning, we compare different initialization algorithms to see how a network's starting point affects the search for winning tickets.

Method

To find out how initialization algorithms compare when looking for winning lottery tickets, a large number of experiments were done with the following specifications.

Dataset. The MNIST dataset as used, described and published in Lecun et al. (1998). This is a labeled collection with 28×28 grey-scale images of handwritten digits, 60k of which were used as a training set, and 10k as a testing set.

Model. A neural network following the specifications in Lecun et al. (1998), as is also used in Frankle and Carbin (2019) in their paper about the lottery ticket hypothesis. It is a fully connected network containing two hidden layers with 300 and 100 neurons, in that order. With an 784-neuron input layer – sized to the MNIST's 28×28 images, this sums up to 266.2k weights.

Initialization methods. As a starting point, four initialization algorithms were tested, stemming from two distinct approaches to the initialization process: one named *Xavier*, coined by Glorot and Bengio (2010), and one called *Kaiming*, originating from He et al. (2015). Each approach offers a formula to supply the specifications for both a normal and a uniform distribution, which is how we arrive at four algorithms in total.

Table 1: The first four initialization algorithms that were tested. The formulas in the table supply the missing ingredients for the matching distributions to the left. `fan_in` and `fan_out` stand for the number of incoming and outgoing connections to a neuron, respectively.

	Xavier	Kaiming
$\mathcal{U}(-a, a)$	$\sqrt{\frac{6}{\text{fan\_in+fan\_out}}}$	$\sqrt{\frac{3}{\text{fan\_in}}}$
$\mathcal{N}(0, \text{std}^2)$	$\sqrt{\frac{2}{\text{fan\_in+fan\_out}}}$	$\sqrt{\frac{1}{\text{fan\_in}}}$

In table 1, the equations specify the bounds of a uniform ($\mathcal{U}$) or standard deviation of a normal ($\mathcal{N}$) distribution. The intuition here is that both formulas scale the size of the weights in a layer inversely proportional to the number connections to that layer, in order to keep the activation from exploding or vanishing on the forward pass. The Xavier method incorporates information about the number of connections *from* the layer to additionally prevent the gradient from doing the same on the backward pass (Xavier).

The precise result of these functions applied on the model used here can be seen in figure 1, where the probability density functions for each initialization method per layer are plotted.

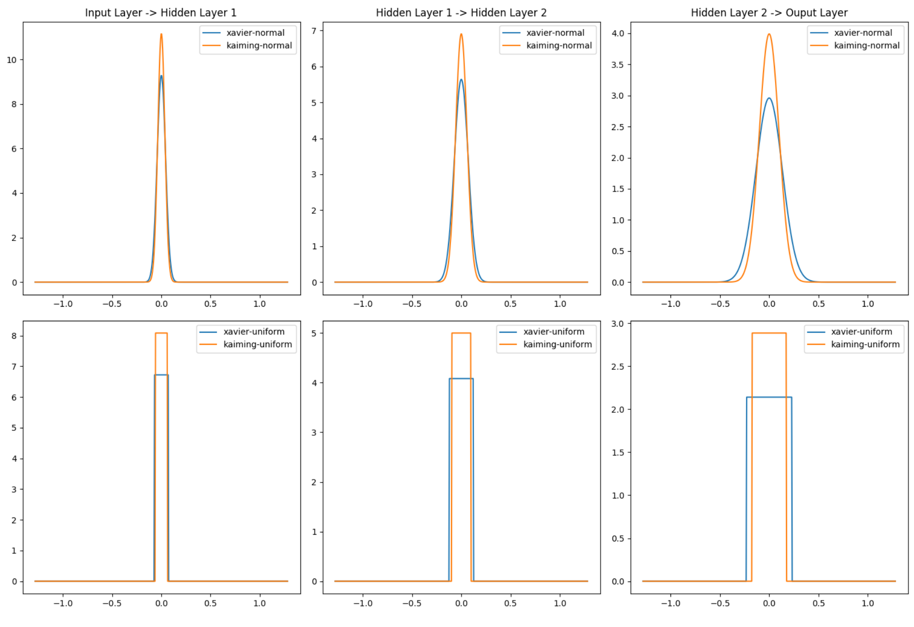

Figure 1: Plots for the probability density functions for each initialization method, per layer.

In addition to these four commonly used initialization methods, four extra algorithms were tested, which can be obtained simply by halving or doubling the outcome of both the Xavier equations, and plugging these values into the the uniform or normal distribution in their stead. This effectively widens (for doubling) or narrows (for halving) the resulting distributions compared to the original version.

Iterative, magnitude-based pruning. In general, pruning is the idea of eliminating weights from a neural network, producing a smaller network. Magnitude-based pruning specifies that the weights that will be nullified are those in the top X percent of weights closest to 0 – the intuition being that the weights that are smallest in absolute terms are the least relevant to the network's outcome and should be the first to go. Iterative, magnitude-based pruning then describes a process where not all pruning is done in one step, but over the course of multiple steps that are spread out over the course of training. (Frankle and Carbin 2019)

Training. For all experiments, the network was trained for 100 epochs at a learning rate of 0.0012, during which it was pruned by 20% and subsequently reset to its initial weights every 5 epochs. See figure 2 for a visualization of the size of the network over the course of an experiment.

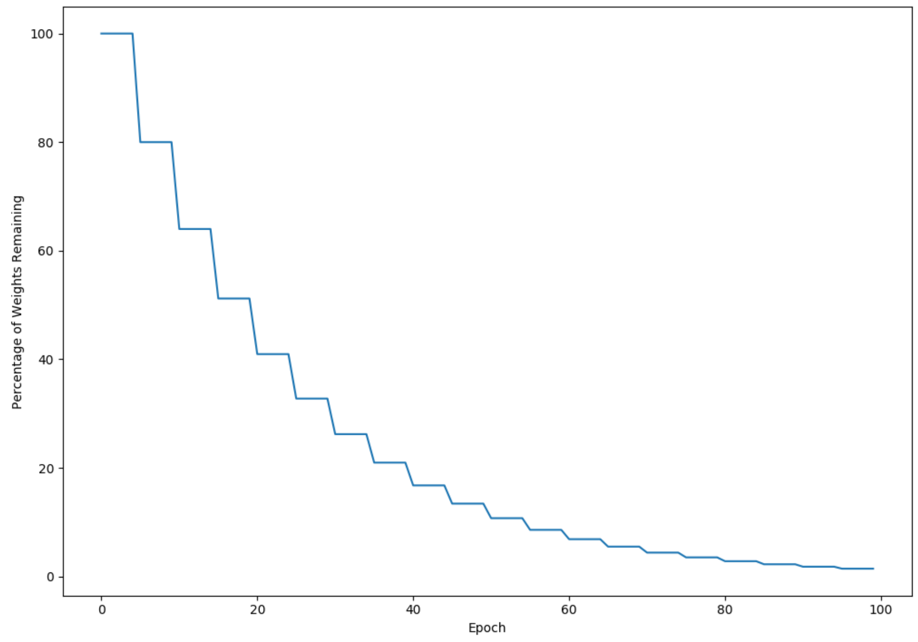

Figure 2: The progression of the size of the network over the course of one experiment.

Testing. To reduce noise in the results from the experiments, each of the eight initialization methods were run 11 times. The primary metric to inform interpretation of the results is the mean test accuracy from these runs at every fifth epoch – right before pruning and resetting occurs –, with the standard deviation over the runs acting as a metric of stability of this resulting mean outcome.

Results

Figure 3 shows the test accuracy of each initialization method as iterative pruning is applied over the course of the experiment. Up to the point where only 3.5% of the weights remain, the mean performance of all four methods is very similar to one another, and every method on its own shows stable results in terms of deviation from its mean when running experiments repeatedly.

It is immediately worth noting that conform to Frankle and Carbin (2019)'s lottery ticket hypothesis, pruned versions of the original network are matching and even outperforming the test accuracy achieved initially. Winning tickets are being found up to 6.9% weights remaining.

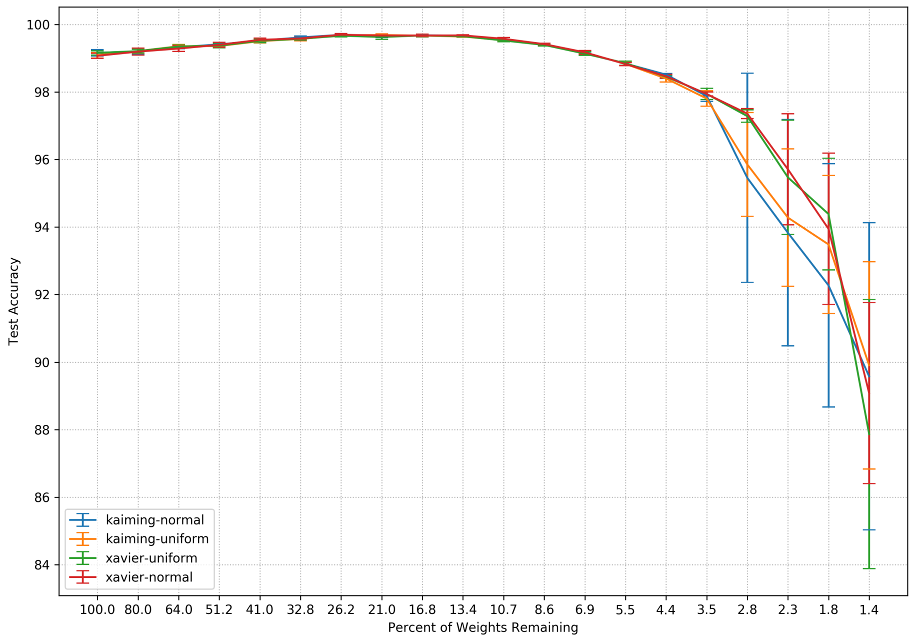

Figure 3: Each line shows the mean test accuracy from 11 differently seeded runs on one initialization method, in order to produce a reliable plot. The error bars represent the standard deviation between the runs; an indication for the stability of the test accuracy at that point. The displayed test accuracy was measured every fifth epoch, right before pruning and resetting.

It could be argued this concludes the research, as we are technically not finding winning tickets anymore beyond this level of sparsity – all of the networks' performances dropped below the accuracy of the original, full-size network at around the 6.9% mark. This disquilifies them from winning the lottery as defined by Frankle and Carbin (2019). However, further inspection uncover regularities that may be useful for further research.

After the 3.5% mark, both Xavier-initalized networks perform better than both Kaiming-initalized ones in the mean and in terms of stability around this mean. Kaiming initalizations led to lower and more erratic performance. This pattern does not hold for long. At 1.8% weights remaining, all the initialization methods show too much internal instability to interpret their performance meaningfully.

Xavier-uniform and Xavier-normal perform in much the same way through the experiment, and the same holds for the Kaiming-normal and Kaiming-uniform pair. Thus, thether the initialization originated from a uniform or normal distribution did not make a difference in these experiments – the deciding difference was the formula that decided the parameters of these distributions.

Further experiments with the Xavier method of initialization have led to the results shown in figure 4. Here both variants of the original Xavier distributions – normal and uniform – have been narrowed or widened by a factor of 2.

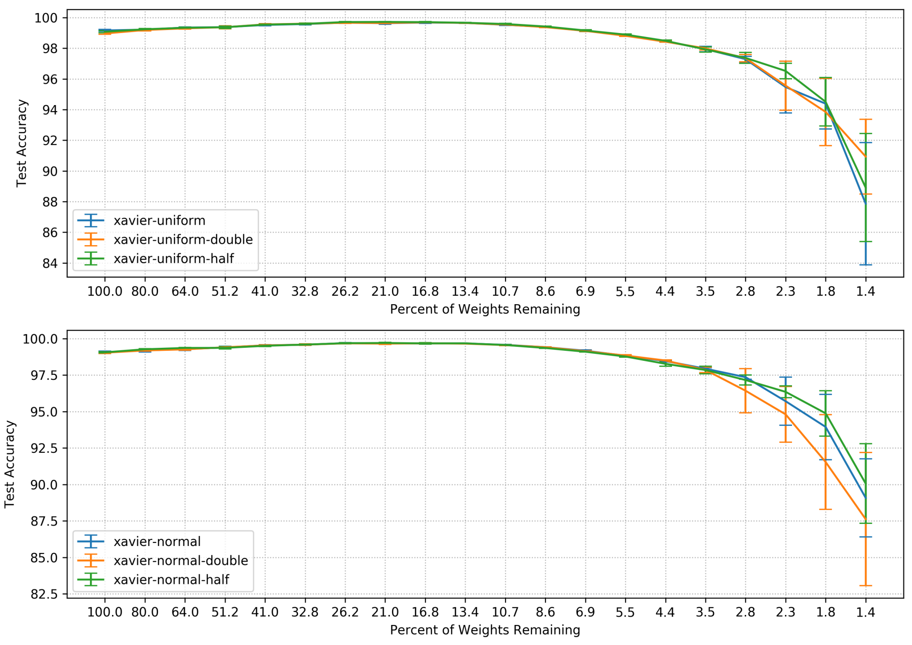

Figure 4: This plot shows the same type of information as in figure 3, but for Xavier distributions that have been widened (`*-double`) or narrowed (`*-half`) by a factor of 2.

The numbers show that both narrowed distributions outperform the other initialization strategies at 2.3% weights remaining in the mean and with a smaller standard deviation – i.e.: they are more stable in their outcome over multiple experiments –, but immediately afterwards the same erratic behavior as appeared in the previous setting re-emerges.

Conclusion

Drawing on the material laid out in the previous section we can inch closer to three conclusions about initialization methods when looking for winning tickets following the lottery ticket hypothesis. In these conclusions, performance is defined as having better test accuracy for lower percentages of weights remaining after pruning.

- Normal distributions and uniform ones have similar performance.
- Distributions that are narrower around 0 outperform wider distributions.
- Xavier methods outperform Kaiming methods.

Although a fair number of experiments have been done per initialization method, and the standard deviations are small enough to have an acceptable level of confidence in the results, more research should be done before these rules of thumb can be set in stone. This project has only examined the application of one pruning method on one neural network model trained on one dataset. That being said, some interpretation for why these results arise from the experiments is in order.

The reason for why normal and uniform distributions perform similarly could be that the network is simply big enough to always give rise to the precise composition of weights that make up the well-performing (although perhaps not winning ticket-level) subnet. In other words: the sample size is so large that the minor difference in the odds in the bell area of a normal distribution versus the odds between the bounds under a uniform distribution simply does not matter enough.

An explanation for why distributions that are narrower around 0 outperform wider distributions could lie in the fact that magnitude-based pruning is used here. Perhaps if the weight initializations are closer together to begin with, the network is more robust to pruning as the other weights can more easily be optimized to fill the gap that was left. This is an interesting notion that should be researched further by experimenting with different pruning methods in combination with narrow and wide initialization methods.

Finally, the reason Xavier initialization methods outperform Kaiming methods here should lie in the size of the weights in each layer relative to weights in other layers. To clarify: notice that although we established narrower distributions outperform wider ones by explicitly testing this factor in isolation, the probability density functions that arise from Xavier methods are in fact wider for every layer than those originating from Kaiming equations. Then notice that *how much wider* Xavier probability density functions are differs per layer. This relationship between weight distributions from layer to layer is an interesting pattern, and there may be a regularity here that is surely worth spending time and resources on to uncover in further research.

Overall, the search for a winning ticket-generating algorithm in the lottery as defined by Frankle and Carbin (2019) is a promising one, and the research presented here shows there may be much to gain from the investigation of different initialization strategies with respect to this search.

References

Frankle, Jonathan, and Michael Carbin. 2019. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.” *ICLR 2019*. https://arxiv.org/pdf/1803.03635.pdf.

Glorot, Xavier, and Yoshua Bengio. 2010. “Understanding the Difficulty of Training Deep Feedforward Neural Networks.” In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 9:249–56. Proceedings of Machine Learning Research. PMLR.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification.” In *2015 Ieee International Conference on Computer Vision (Iccv)*, 1026, 1034. IEEE.

Lecun, Y, L Bottou, Y Bengio, and P Haffner. 1998. “Gradient-Based Learning Applied to Document Recognition.” *Proceedings of the IEEE* 86 (11): 2278, 2324.

Appendix A: Extra Information of Practical Interest

Code and results. We welcome efforts to reproduce and extend our work, and would be happy to get in touch about sharing our repository of code and results. Please contact Stefan Wijnja at s.wijnja@me.com for any queries.

PyTorch. The neural networks in this project are built using the open source machine learning package PyTorch[1]. Among other things, PyTorch includes modules for defining network models, doing backpropagation using automatic differentiation, various optimizers and some built-in datasets. These modules greatly reduce the amount of code and time required to build and train the networks, leaving more time for experiments and analysis of the results.

Randomization. As noted in the report itself, 11 experiments were done for each combination of factors in order to reduce noise in the results. In the interest of reproducibility it is useful for you to know that the seeds used range from 42 to 52 (inclusive).

[1] https://pytorch.org/