Daniel Eftekhari

daniel.eftekhari@mail.utoronto.ca

This (brief) document is written with the following components:

1. **Essentials:** all challenge requirements, as well as the bonus, have been completed, with analysis and investigation where appropriate.
2. **Research Ideas:** I provide some initial theoretical analysis of pruned networks, and demonstrate these properties empirically as well. I hope this will sprout interesting discussion amongst us going forward, both for research in pruned neural networks, and for implications to neural networks in general as well.

My code for the project can be found at https://github.com/DanielEftekhari/neural-network-pruning **.**

This document is written informally (along the lines of a blog post), to convey ideas in an intuitive way, such that further ideas and insights can be exchanged in person during collaboration. A goal of this document, and the code, was to surpass the expectations of the challenge, and to provide a glimpse at research prowess and creativity. The goal is for this work to help cultivate future research.

It was truly a pleasure completing this challenge.

Yours truly,

Daniel Eftekhari

Baseline

For the sake of brevity, consult the challenge outline for details on prescribed model architecture. Figure 1 provides training curves, which will serve as performance benchmarks for comparison to the pruned models. The best model was obtained after epoch 6, with a validation loss of 0.0682.

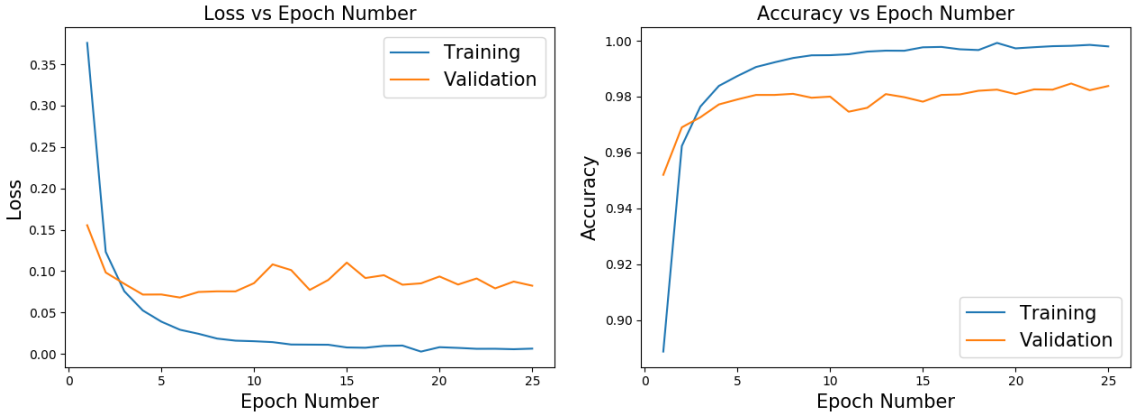

Figure 1 Training curves for the five fully connected layered ReLU-activated neural network. These figures serve as performance benchmarks for comparison to the pruned networks analyzed below. Note that while using a ReLU activation may sparsify a layer's output activations, it does not necessarily sparsify the weights, which is the topic of investigation in this work.

Pruning

Pruning was applied layer-wise for each network (i.e. the smallest k fraction weights/units were eliminated for each layer separately). Figure 2 and Figure 3 provide the performance metrics for the weight and unit pruned networks, respectively. Observe that model performance degrades with increasing sparsity in both weight and unit pruning, the reason being that important information contained in the network is eliminated when pruning. Prior to this work, the author suspected the performance degradation pattern would have a sigmoidal shape. (that is, a low effect on performance for small pruning factors, a sudden change in performance at critical pruning factors, and then again a small incremental effect on performance for pruning factors beyond the critical values) and interestingly the performances do have some semblance to a sigmoidal curve.

Note that this degradation in performance occurs much more quickly in the unit-pruned network than in the weight-pruned network. This phenomena can be attributed to the following hypothesized reasons:

1) Unit-pruning eliminates entire features (neurons) from the network, disabling their ability to pass forward their information. In contrast, weight-pruning allows for all neurons, except in the unlikely scenario where every weight belonging to a neuron is eliminated, to continue passing forward their information. The ability to pass on information despite “local damage” in the weight-pruned network is analogous to the idea of distributed representations, popular in the intersection of computer and cognitive sciences.
2) More subtly the following observation can be made. Notice that the weight-pruned model **necessarily** eliminates the k-fraction lowest magnitude weights. In contrast, the unit-pruned model only eliminates neurons with the lowest total L2-norm, thus not guaranteeing the elimination of the lowest magnitude weights. This results in important information being eliminated in the unit-pruned network, which would not be eliminated in the weight-pruned network. The following Theorem formalize this idea, and is demonstrated empirically in Figure 4.

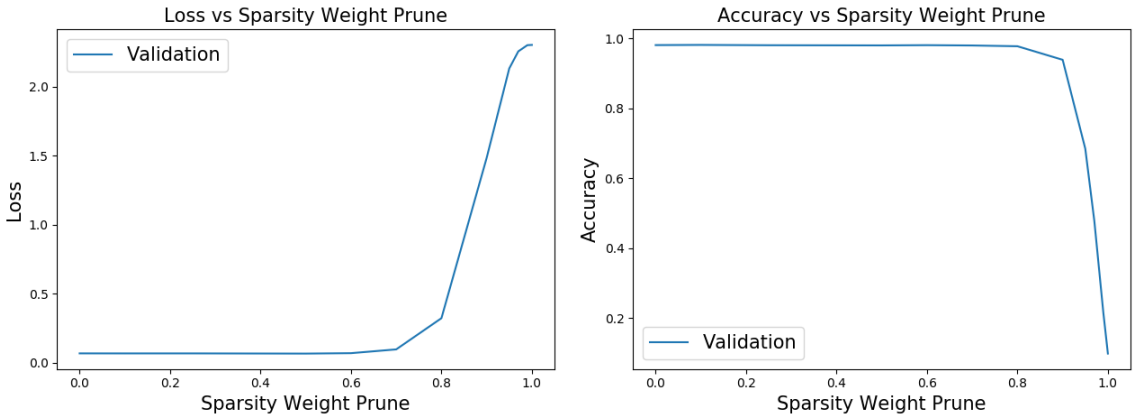

Figure 2 Loss and accuracy for varying sparsity factors for weight-pruned networks. Note that the performance of the model only decreases significantly after a high pruning factor is used, indicating some robustness to pruning. This is in contrast to the unit-pruned network. See text for analysis.

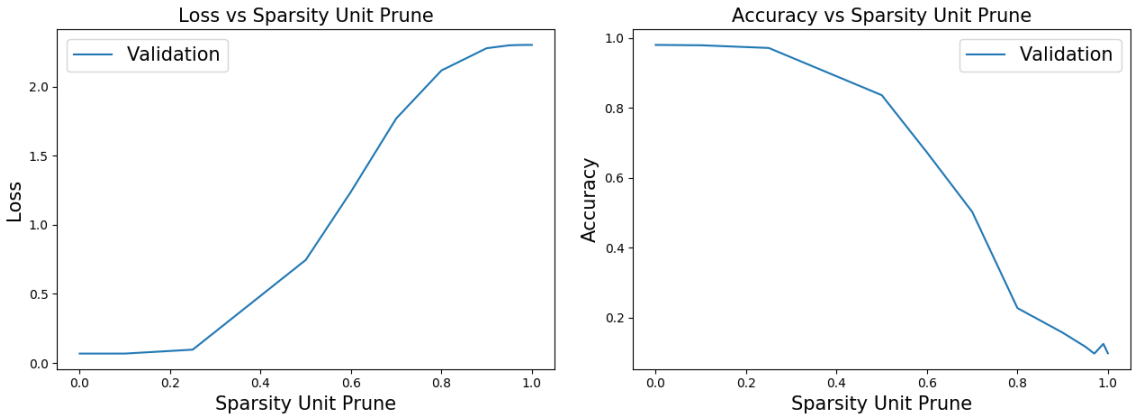

Figure 3 Loss and accuracy for varying sparsity factors for unit-pruned networks. Note that the performance of the model decreases significantly even after a relatively low pruning factor is used. This is in contrast to the weight-pruned network, which demonstrates robustness to pruning. See text for analysis.

Theorem 1 (Can be made rigorous in future work)

Statement

A neural network consisting only of fully connected layers will tend towards eliminating weights randomly with respect to their magnitudes in unit-pruning (assuming the weights were initialized randomly prior to training).

Proof

Observe that in a fully connected layer, the permutation of the neurons in each layer is on average (what this average is with respect to is discussed on page 6) immaterial. For a proof of this idea, consult investigations around the non-convexity of neural network loss functions with respect to the network parameters, which is due to the permutation-invariance of the neurons.

Therefore, each neuron will on average produce weights with L2-norms equal to that of any other neuron in the same layer. It is only a statistical matter that they are not precisely the same. Because each neuron's weights have equal L2-norm on average, the k-fraction lowest L2-norm neurons will tend towards being randomly selected for elimination in unit-pruned networks.

This (perhaps surprising, and beautiful) result is demonstrated empirically in Figure 4. In the weight-pruned network, we can see that the lowest magnitude weights are necessarily eliminated (red), and the surviving weights have higher magnitude (green). But in the unit-pruned network, the weights are eliminated somewhat randomly, as evidenced by the similarity in distribution between the eliminated (red) and surviving weights (brown).

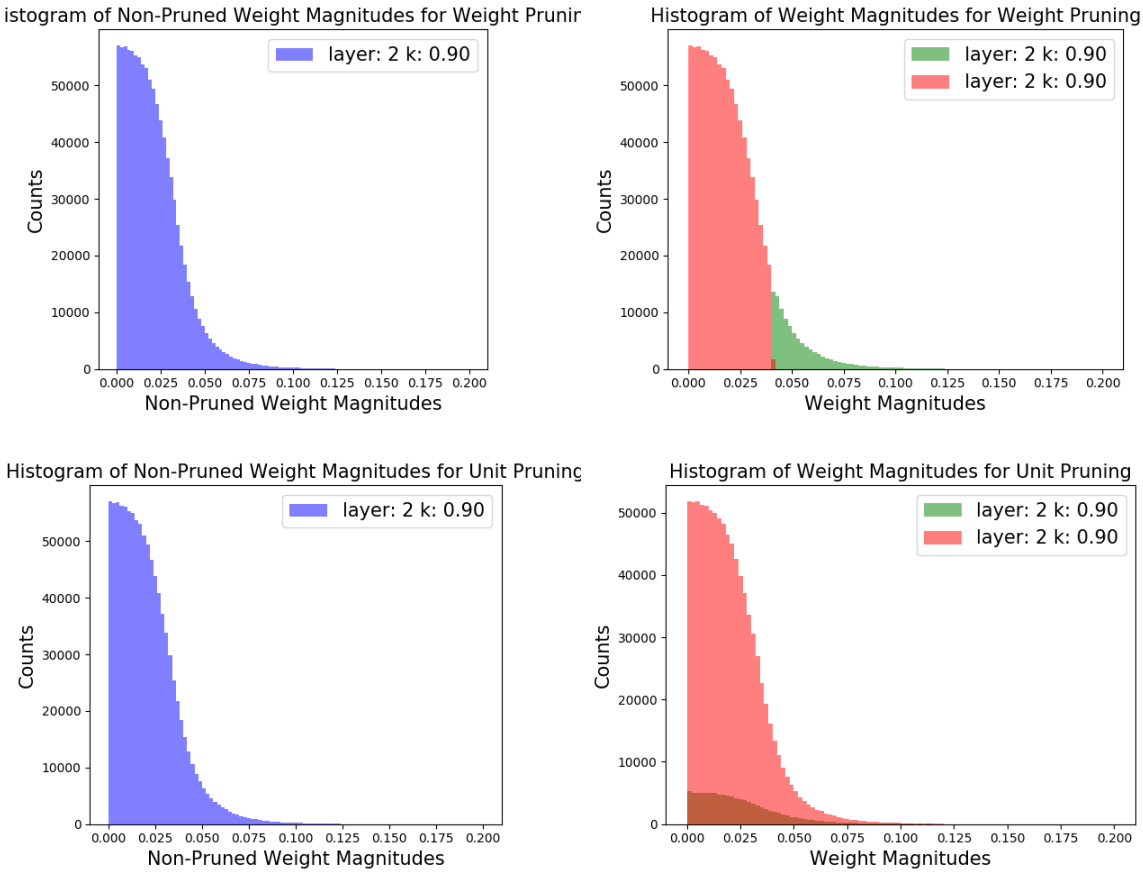

Figure 4 A demonstration of the effect of weight and unit pruning. Weight-pruning necessarily eliminates the k fraction lowest weights in a layer. However, unit-pruning tends towards randomly eliminating weights with respect to their magnitudes, the reasons for which are provided in-text.

The mathematical rate of approach to random elimination of weights w.r.t magnitudes in unit-pruning, and what parameters this rate depends on (ex: number of units in the layer, the type of norm used for ranking, etc..), and the bound approximating randomness, is unknown to the author (Daniel Eftekhari) at the time of writing, but he is pursuing it.

Time Analysis

Figure 5 demonstrates the performance gains in speed of computation with increasing fractional pruning rate (k). These speedups occur due to 1) weight matrices in pruned layers are converted to sparse tensors (in coordinate list (COO) representation) 2) sparse matrix multiplication modules are used for the sparse tensors. The timing results in Figure 5 are produced by running on a CPU, in order to ensure reproducibility of results.

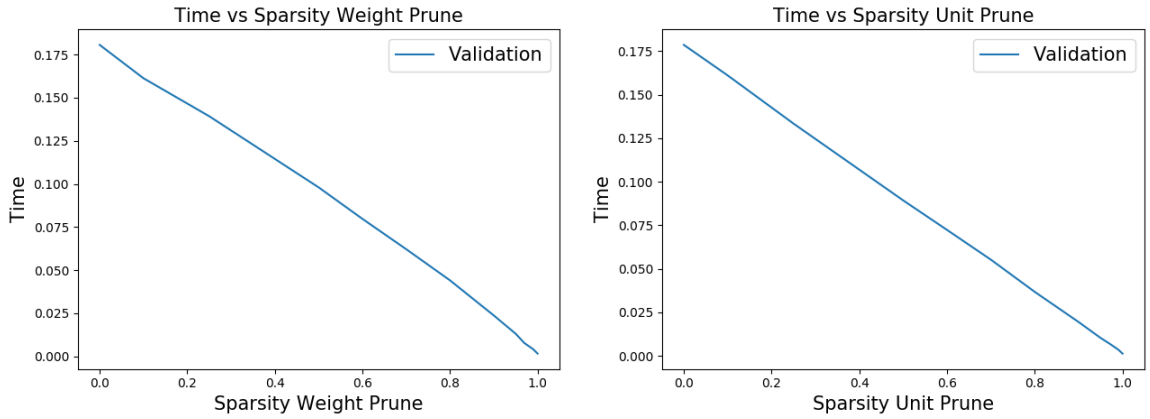

Figure 5 The average time for the model's forward pass on a minibatch of 100 samples decreases significantly with increasing sparsity, for both weight and unit-pruning, when the appropriate deep learning sparse libraries are employed.

Future Work

This work can go on to explore several ideas for future work, such as:

1. Formalize, and rigorously investigate, Theorem 1 and its implications.
2. Investigate alternative pruning approaches, for example as outlined in "Optimal Brain Damage" (LeCun et. al 1990) (http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf), and investigate what their effects on model performance are.
3. Investigate whether fully connected (dense) layers trained on image classification tasks, when pruned, tend towards the weight-connectivity patterns of convolutional neural network (CNN) layers. That is, whether pruning can imitate the prior on CNN networks, which enforces that input neurons are only connected to spatially near output neurons.