Exploring The Lottery Ticket Hypothesis for Pruning Neural Networks

Implementation track, Two members

Abstract

A recent work proposed an iterative pruning methodology which retrains a model while routinely setting setting the lowest magnitude weights to 0, hence *pruning* the network [1]. We applied the same iterative pruning methodology as proposed in [1], and confirmed its results with neural networks on a small data set. We test using modifications proposed in [2] to extend the pruning methodology to deeper networks, and again confirm that the methodology can find sparse networks. We additionally applied the iterative pruning methodology to graph convolutional networks, with a citation network dataset, and again find that the iterative pruning can find very sparse subnetworks which match performance of the unpruned model. Finally, we attempt to improve upon the iterative pruning process by incorporating knowledge distillation into training. Our results are preliminary, and no clear conclusions can be drawn, suggesting further study into the effects of knowledge distillation on the pruning process.

1 Introduction

Modern neural networks have many learnable parameters, which may make operation in some settings prohibitive. There's motivation to reduce the number of parameters while maintaining or only slightly decreasing performance. Pruning methods for neural networks are standardized ways to remove parameters, and in many cases can greatly reduce parameter counts while minimally reducing performance. One possible benefit of educing parameter counts is using less energy [8], improving the efficiency of network inference (applying the model after training).

Thus, the goal of a pruning method is to be able to prune as many parameters while maintaining performance, usually as measured on hold-out data sets. Generally, pruning methods rank the parameters using some measure of importance, like magnitude, and pruning the lowest ranked weights. These are *unstructured* pruning methods, as they correspond to pruning edges in the network. This can be extended to an iterative process, whereby $p\%$ of weights are pruned repeatedly, resulting in a model with $(100 - p)^n)\%$ parameters remaining after n pruning iterations.

These are in contrast to *structured* pruning methods, which prune nodes from the network which is equivalent to collectively pruning all edges incident on a node. Structured pruning methods reduce computational costs, where unstructured pruning does not in general, but structured pruning trades computational efficiency for worse accuracy for a given pruning amount.

1.1 The Lottery Ticket Hypothesis

The Lottery Ticket Hypothesis argues it is possible to reliably prune $p\%$ of parameters and still acquire similar levels of accuracy of the original network [1]. The pruning method proposed in the Lottery Ticket Hypothesis paper is a type of unstructured iterative pruning which introduces a novel strategy: after pruning, rewind parameters to their initial values, then continue training. This strategy has since been generalized to rewinding to parameter values from *early* in training, rather than initial values [2].

Preprint. Under review.

Using the method proposed in the Lottery Ticket Hypothesis, a sparse subnetwork of parameters which can train to comparable accuracy as the original network is called a *winning ticket*. If the network were sagaciously initialized with these weights (particularly, with some weights pruned to 0), it would achieve similar performance as an unpruned model with more parameters. However, discovering the winning ticket subnetwork necessarily requires training the model to convergence, as parameters are pruned based on their values after training.

Empirically, this method has managed to find very small sub-networks [1], and for complicated tasks like image classification on ImageNet, where the state of the art models are extremely large [6, 2]. As compared to other pruning methods, the Lottery Ticket Hypothesis strategy produces models which are generally sparser for equivalent reduction in performance.

In this paper, we explore the Lottery Ticket Hypothesis by implementing the proposed iterative pruning method and applying it to several neural network architectures and data sets. We repeat pruning experiments from [1], including pruning the ResNet18 architecture [3] trained on the CIFAR10 data set to only 21% of its parameters with an *increase* in test performance. We also demonstrate the efficacy of this pruning method on a graph neural network trained on a standard graph benchmark data set. We find that the standard graph convolutional network (GCN) can be heavily pruned with similar performance, providing evidence for the Lottery Ticket Hypothesis outside of image classification.

Finally, we begin exploring the potential of using knowledge distillation techniques to improve iterative training. Knowledge distillation is a means of leveraging an existing model labeled the teacher to aid in training a new model, labeled the student. Knowledge distillation has been found to be an effective technique for training students which are less complex than the teachers [4].

In iterative pruning, models from earlier pruning iterations are trained to convergence, but traditionally are not used in training for subsequent pruning iterations. We begin to investigate a strategy for incorporating knowledge distillation into iterative pruning by using a model from an earlier pruning iteration as the teacher network. We have two hypothesis: first, that knowledge distillation can potentially improve the rate of learning, requiring less iterations over the data to train to convergence. Second, we hypothesize that knowledge distillation can improve the quality of the student network, resulting in improved performance as measured against a hold-out data set.

Due to limited time and computational resources, we only present initial findings for knowledge distillation. We perform few experiments again on ResNet18 and CIFAR10, each for only a single trial. We tentatively find that the rate of learning is unchanged by knowledge distillation, but that classification accuracy as measured on a validation set appears to improve, measured during an early epoch in training. The experiments are detailed in section 4.2.

This paper is organized as follows. We review some related work in section 2. In section 3, we present a mathematical description of our methodologies. We detail our experiments in section 4, and finally conclude in section 5.

2 Related Work

The Lottery Ticket Hypothesis and the iterative pruning method are initially presented in [1]. A further work does an ablation-style study to understand which parts of the original pruning strategy are needed, as well as presenting some alternative heuristics for pruning weights [9]. An interesting alternative to the pruning heuristic used here is to prune the weights with the lowest change from their initial value. In some tests, they found that pruning by lowest change rather than lowest final magnitude improved performance. We do not consider any heuristics outside of the lowest final magnitude, but it would be interesting to compare across more data sets and architectures.

In the previous two works, only relatively shallow networks are studied, and it is noted that the strategy presented in [1] is unable to find winning tickets for deeper networks without significant performance sacrifices. A simple modification to the original algorithm is presented in [2]. The notable change to finding winning tickets in large networks is to reinitialize to the parameters of an *early iteration*, rather than the *first initialization*.

Finally, this modified strategy is used to find winning tickets for deeper models in [6], where they also demonstrate that pruned networks can be used for transfer learning. Evidence that pruned networks

can still be used for transfer learning indicates that the pruned model still learns useful embeddings in early layers, suggesting a similarity between pruned networks and their unpruned counterparts.

Generally, these works all explore methodologies for finding winning tickets. We continue this trend by applying their methods in new situations, and exploring ways to improve the iterative pruning methodology.

There is much more work in the area of pruning neural networks in general. We focus more solely on the work related specifically to the Lottery Ticket Hypothesis, mostly in the interest of time. In future work, it would be interesting to compare to other classes of pruning methods.

3 Methodology

We denote a neural network as $f(x;\theta)$, where the network has initial learnable parameters θ_0. Applying a mask operator $m \in \{0,1\}^{|\theta|}$ to the parameters results in a pruned network[1] $f(x; m \odot \theta)$, which is a subnetwork of the original $f(x;\theta)$. Note that the sparsity of the mask operator determines the number of parameters pruned, i.e. a sparse pruned network will have[2] $||m||_0 << |\theta|$. The lottery ticket hypothesis posits that it is possible to find a mask such that the *winning ticket* subnetwork $f(x; m \odot \theta_0)$ will perform at least as well as the unpruned model $f(x;\theta_0)$ in at most the same number of training iterations [1].

The proposed strategy to finding these winning tickets is to iteratively train a network, prune the lowest $p\%$ of parameters, and then reset them to their initial values. Intuitively, the parameters with the largest magnitude have the most effect on the output, while the lowest magnitude parameters have little effect (and a parameters of 0 has no effect). It was found in [1] that only performing the pruning step performs worse than rewinding weights and then retraining. Additionally, resetting the weights to random initial values after pruning does worse than resetting to the parameter's initial values. This suggests that the value in the winning ticket subnetwork is in both its architecture and the parameter initialization, i.e. the winning ticket finds an architecture for the particular parameter initialization that is well suited to the task. Formally, the procedure for performing iterative pruning is:

1. Select a random initial set of parameters θ_0 for a neural network $f(x;\theta_0)$.
2. Train f until convergence, resulting in parameters $\hat{\theta}$.
3. Create a mask $m_i(\hat{\theta}) \in \{0,1\}^{|\theta|}$ s.t. $m_i = 0$ for the lowest $p\%$ of parameters in $\hat{\theta}$ by magnitude (i.e. prune the lowest magnitude weights), excluding those parameters already pruned.
4. Rewind parameters to θ_0. Repeat from step 2 using the pruned model $f(x; m_i \odot \theta_0)$.

Where i above denotes the ith iteration of pruning, in which $(1-p^i)$ percent of parameters have been cumulatively pruned. This procedure can be varied by using different sorting orders, some of which are investigated in [9]. Other variations consider sorting parameters by layer rather than globally, and pruning a number of parameters from each layer in proportion to the size of the layer. This was found to not be as effective as a global ordering of parameters [1].

To reliably identify winning tickets in deeper neural networks, [2] proposes a slight modification: rather than rewinding parameters to their initial values, instead rewind them to an early iteration. In other words, in step 4 parameters are rewound to θ_j for some iteration j.

We train our networks using stochastic gradient descent (SGD) and Adam, with an appropriate loss function L, e.g. cross entropy loss. The type of optimizer and use of special layers for training neural networks like dropout and batch normalization are not thought to impact the presence of winning tickets in networks [6, 1].

Combining knowledge distillation with pruning only changes the training step. Here, the student is the neural network on pruning iteration i, i.e. $f_i = f(x, m_i \odot \theta_j)$, and the teacher is a neural network from a previous pruning iteration $j < i$, i.e. $f_j(x, m_j \odot \hat{\theta}^j)$ where $\hat{\theta}^j$ denotes the converged parameters on the j pruning iteration. The student is trained using a modified loss function $\tilde{L}$ which is a weighted average of the original loss between $f_i(x)$ and hard labels y, and the loss between $f_i(x)$ and soft labels $f_j(x)$. Here, we use the term hard labels to refer to a one-hot vector encoding of the

[1] The $\odot$ operator denotes element-wise multiplication.

[2] This denotes the 0-norm, the number of nonzero elements of a vector: $||x||_0 = |\{x_i \neq 0 \; \forall i\}|$.

target class, and soft labels refers to a vector as a distribution over the classes.

$$\tilde{L}(f_i(x), y) = \alpha L(f_i(x), y) + (1 - \alpha) L(f_j(x), f_j(x; m_j \odot \hat{\theta}_j)) \tag{1}$$

The knowledge distillation loss incorporates the outputs of a previous model $f_j(x)$ as a target for the currently-training student network f_i. For a training pair (x, y), the hyperparameter $\alpha \in [0, 1]$ weights the influence of the traditional loss using the hard labels y as targets and the modified loss using the soft labels $f_j(x)$ as targets. Note that $\alpha = 1$ corresponds to the traditional loss with only hard labels, and $\alpha = 0$ corresponds to a loss considering only the soft labels $f_j(x)$ generated by the teacher network.

The motivation behind this loss function is that the already trained model may be able to provide more informative feedback on training examples through the soft labels [4], and potentially increase the time to convergence for training the student or improve performance on test data. Knowledge distillation has been applied as a means to learn less complex models, where the soft labels provided by the teacher enable the student to learn effectively where a traditional training procedure would be less effective. This matches the pruning scenario: the teacher network has more parameters, $||m_j||_0 > ||m_i||_0$, and is therefore more complex.

However, we might also imagine that the student network already incorporates significant information from the teacher (and from all previous pruning iterations) due to the inductive bias provided by the mask. That is, the student's mask selects parameters which were already found to be effective on this task in previous pruning iterations. If the additional information provided by the soft labels is already included in the inductive bias of the mask, knowledge distillation may not improve the training of the student network. Thus it is unclear whether what benefit knowledge distillation could provide to the iterative pruning process, if any.

4 Experiments

We implemented our work using PyTorch, including the training and pruning procedures. We used PyTorch library code to handle the masking of weights using PyTorch's various `prune` methods[3].

We first present results on a small fully-connected neural network (FNN) trained on the MNIST dataset, as an initial confirmation of our iterative pruning implementation. The FNN has four linear layers, with (784, 64, 32, 10) nodes in each layer as well as ReLU activations and dropout with probability 0.5 for the first three layers. We trained using a batch-size of 32, the Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.99$ (these are additional hyperparameters for Adam), and each network is trained for 2 epochs. We use standard cross entropy loss. At the end of training, we prune the lowest 20% of weights in the first three layers, and prune no weights from the final layer. Figure 1 shows the performance of the model on the test data set across 20 pruning iterations, averaged over five trials.

On this simple neural network, pruning does not appear to improve test accuracy. But the model can still be pruned with relatively minimal decreases in performance, which seems to conform with the lottery ticket hypothesis. However, this network is small and the MNIST task is comparatively simple, so this experiment mostly serves to confirm that our implemented procedure for pruning works.

Next, we consider training the ResNet18 architecture [3] on CIFAR10, a large data set of 60K images with 10 classes. Compared to the previous test, the model has far more parameters and layers, and the data set is more challenging. We follow the modification to iterative pruning and rewind weights to their value after 500 iterations, rather than their initial values, to improve the quality of the pruned subnetworks [2]. Otherwise, training is mostly the same as in [3] for the CIFAR10 data set: we split into training, validation and test sets, with 45K, 5K and 10K examples in each respectively, with a batch size of 128. We use SGD with a learning rate of 0.1, and decrease learning rate by a factor of 10 at epochs 90 and 136 (this learning rate schedule is approximately the same as in [3]). Finally, we use a momentum value of 0.9, and weight decay value of 0.0001. We prune the lowest 20% of weights, but we do not prune the final fully connected layer nor the downsample layers in ResNet

[3] Addressing our group's effort: we did implement working code for pruning, but in all of our experiments (except the basic FFN) we used the PyTorch library version as it was easier to work with. We did have to add to this procedure to handle rewinding weights and saving/reloading masks across multiple sessions.

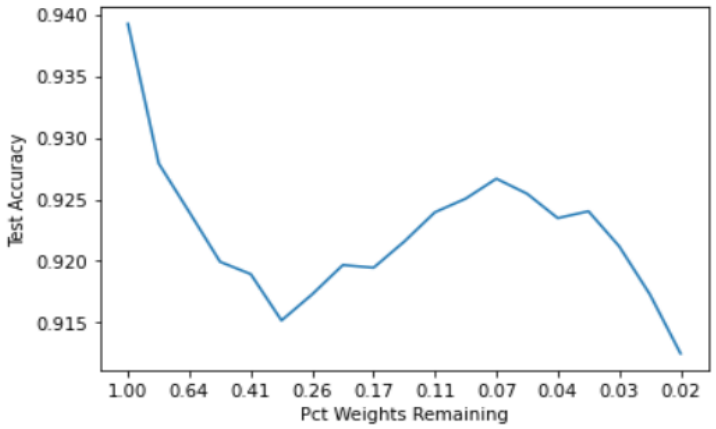

Figure 1: Pruning a 4-layer fully connected network on MNIST: can prune to roughly 2% of weights remaining with only minimal drop in accuracy. Results shown as an average over 5 trials.

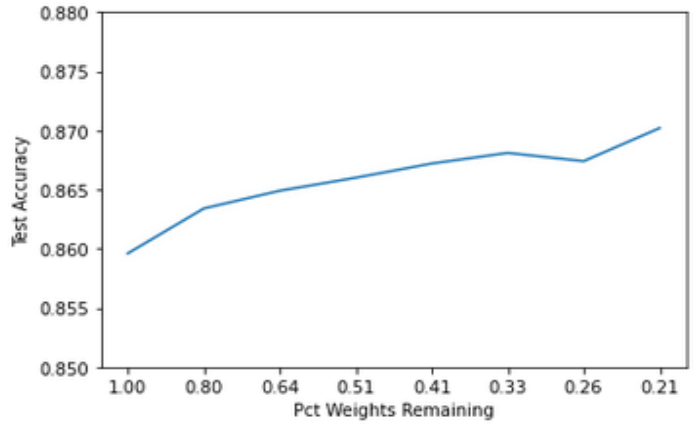

Figure 2: Change of test accuracy during pruning ResNet18 trained on CIFAR10

as these layers represent very few of the overall parameters. This is in accordance with ResNet18 pruning in [1].

Model performance is evaluated on the validation set every 1000 iterations. We then select the model with the highest accuracy on the validation set after 180 epochs for pruning and evaluation on the test set. Figure 2 shows the resulting test accuracy over 7 pruning iterations, giving a final model with only 21% of weights remaining. This constitutes a reduction from about 11M parameters for ResNet18 to about 2.3M parameters, with no decrease in test accuracy.

Interestingly, test performance notably improves as the model is pruned. Although we only prune for 7 iterations, significant pruning past this point would begin to deteriorate performance [2]. A major drawback of our analysis here is we only performed a single trial, so it is possible that the increase in performance is due to noise in training. However, test accuracy improvement is a noted potential outcome for iterative pruning [1].

4.1 Pruning Graph Neural Networks

Thus far we have only examined pruning on image classification tasks. To confirm that the Lottery Ticket Hypothesis generalizes to other tasks, we apply iterative pruning to a graph neural network, particularly the graph convolutional network (GCN) [5]. We use the Cora data set, which is a citation network frequently used in graph benchmarks [7]. Cora consists of 2708 nodes, each a particular

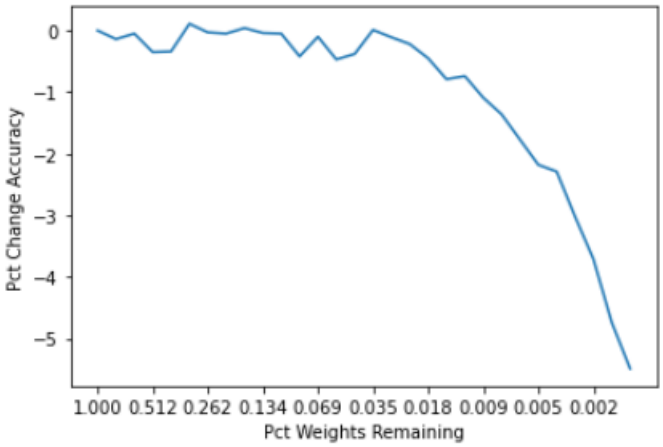

Figure 3: Pruning a graph convolutional network (GCN) results in small deviations from the initial accuracy of 79.97% until about 2% of weights remain. After this, performance decreases very rapidly, although stays far above random even for very few parameters remaining. After 30 pruning iterations, 0.12% of weights remain, and test accuracy is on average 74.48%. Results shown as an average over 10 trials.

scientific publication in the area of machine learning, and 5429 edges, where an edge denotes that one paper cited another. A bag of words feature vector is associated with each node, and each node has one of seven labels corresponding to the specific area within machine learning. The graph structure and all feature vectors are available at training time, but only a subset have nodes. The task is to predict the classes of the remaining nodes.

We train a two-layer GCN with 32 hidden units, with a ReLU activation and dropout with probability 0.5 for the first layer. We use the Adam optimizer, with a learning rate of 0.01, weight decay of 5e-4, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We train for a total of 200 epochs, then prune just the parameters in the first layer (there are very few parameters in the second layer), again using the lowest 20% magnitude rule. Our results for 30 pruning iterations are in figure 3, which represents the average of 10 trials. For reference, the average accuracy before any pruning was 79.97%, which is roughly the performance obtained in [5].

An interesting result of this example is the GCN can be reliably pruned for 30 iterations, resulting in very few nonzero weights. The network achieves an average accuracy of about 75% on the test set after 30 iterations of pruning, meaning only 0.12% of weights remain. While this result is ostensibly remarkable, the data set is very small, and our initialization of GCN may have been overparamaterized for the task. To better understand pruning graph neural networks, we should either try applying the GCN to a larger data set with a more challenging task, or instead investigate a less complex model with fewer parameters that does similarly well on Cora.

4.2 Knowledge Distillation

In this final section, we present our limited results on combining knowledge distillation with iterative pruning. We train an initial model f_0, again using ResNet18 architecture and CIFAR10, using the same training process stated previously. We then prune the model, resulting in f_1 and train the result for 80 epochs, measuring validation accuracy every 1000 iterations. However, we use the modified knowledge distillation loss from 1 with varying values of α, where the student network is f_1 (the pruned model) and the teacher network is f_0 (the unpruned model which has been trained to convergence). Figure 4 shows the validation accuracy over training. Note that $\alpha = 1$ corresponds to the traditional cross entropy loss with hard labels, while $\alpha = 0$ corresponds to cross entropy loss with just soft labels.

We are limited to only a small amount of data for this experiment, again having only a single trial for each measurement, so we are hesitant to draw strong conclusions from these results. Nonetheless, it

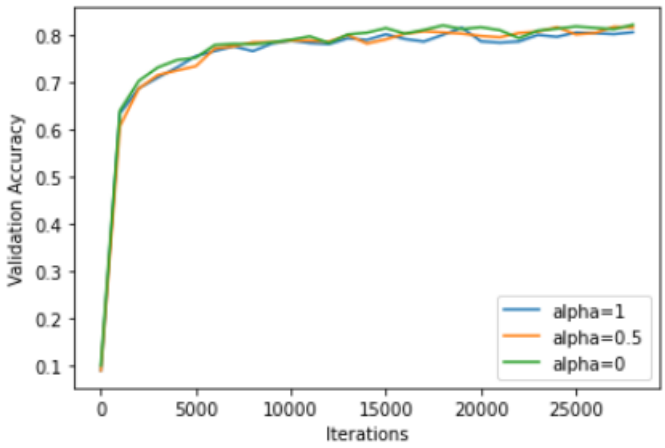

Figure 4: ResNet18 accuracy on the validation set over the first 80 epochs, after pruning 20% of weights, appears to follow the same trend for whether loss considers hard labels ($\alpha = 1$) or soft labels ($\alpha = 1$) or a mix ($\alpha = 0.5$). It is not clear that knowledge distillation has an effect on training in iterative pruning.

would appear that training with knowledge distillation does not improve the rate of convergence: the three procedures all follow roughly the same trend over the 80 epochs.

It is also not obvious from figure 4 that knowledge distillation can improve the performance of pruned models. However, it seems that for $\alpha = 0$, i.e. the soft label case, the final validation accuracy after 29K iterations (roughly 80 epochs) is 1.5% larger than for the $\alpha = 1$ hard label case. Indeed, this validation accuracy at this stage of training is the largest we observed over all instances of ResNet18 on CIFAR10. The validation accuracy[4] after 29K iterations is included in Appendix Table 1 for all ResNet18 models in this paper. In place of commenting on the generality of this result, we instead argue that further investigation should study the potential for knowledge distillation to improve the performance of pruned models. Although training with the teacher network is more computationally expensive, an accuracy improvement of 1.5% would be substantial if it could be obtained reliably. Since we only used a single trial, we cannot reason about the consistency of our result.

Table 1: Table of validation accuracy of ResNet18 after training for 29K iterations (roughly 80 epochs).for varying levels of pruning and knowledge distillation. A teacher model was used for $\alpha = 0, 0.5$ to generate soft labels as a target for the model.

Prop. Unpruned	0.8				0.64	0.512	0.41	0.328	0.262	0.210
KD Value	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$							
Val Acc	**0.822**	0.814	0.807	0.810	0.810	0.805	0.806	0.810	0.819	0.808

5 Conclusion

We successfully applied iterative pruning, confirming our implementation's consistency with the findings in [1]. We were also successful in applying the method to a new architecture and data set, demonstrating that the Lottery Ticket Hypothesis holds even outside of image classification settings.

A notable difficulty with this iterative pruning methodology is the computational resources needed to train models numerous times. Here, performing multiple trials for ResNet18 on CIFAR10 would be needed to better understand the effects of iterative pruning. This constitutes a sort of "con" for the method in general, as results are limited to tractable experiments, which may not be sufficient to establish meaningful results.

[4]Test accuracy would be a more representative measure to report here, but we did not have time to run these models on the test set at this particular stage of training. We still report validation accuracy as the section is more aimed at summarizing our results than presenting an argument.

For our results on knowledge distillation, all the results can imply is that more investigation is needed. It would be interesting to conduct further experiments with different learning strategies, e.g. varying the weight of the soft labels α throughout training. A drawback of our knowledge distillation implementation is the added computational complexity: for each mini-batch of data, the outcomes need to be computed for both the student and teacher networks. While the parameter updates only need to be computed for the student network, the forward pass for the teacher is on a similar order of computational cost. This should be taken into consideration when evaluating the time it takes to train using knowledge distillation.

Further understanding the nature of lottery tickets may lead to new weight initialization schemes, or more efficient training methodologies. For the moment, these pruning strategies are of potential interest in resource constrained settings which may take advantage of the sparsity created by the unstructured pruning.

However, unstructured pruning does not improve performance on modern hardware and with standard libraries, where dense matrix multiplications are much faster than sparse matrix multiplications outside of very sparse examples [1]. Unstructured pruning is thus not immediately practical for reducing time or computational cost, but is still an interesting research area for its potential to help us better understand neural network training and work towards capable models with fewer parameters.

References

[1] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[2] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

[6] A. Morcos, H. Yu, M. Paganini, and Y. Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, pages 4932–4942, 2019.

[7] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[8] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017.

[9] H. Zhou, J. Lan, R. Liu, and J. Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pages 3597–3607, 2019.