Pruning Deep Neural Networks

Sean Hooker

University of Warwick

December 2, 2019

The use of Deep Neural Networks (DNN) in various areas and applications has allowed for state-of-the-art performance to be achieved. The most notable areas where DNN excels include image recognition, speech recognition and natural language processing. With more data becoming available for these models to learn off of and improve, these models are also becoming harder to deploy in resource-constrained environments (such as mobile phones) due to their magnitude.

Researchers are looking extensively to try and compress these neural networks to improve performance and efficiency whilst also preventing any significant loss in model quality/accuracy. Methods which have been proposed thus far to help solve this issue [1] include approximation of weights, weight quantization, knowledge distillation and network pruning. Out of these methods, network pruning has been seen as one of the most popular due to its impressive performance and capabilities in reducing parameter size within a model with little or no loss in model quality.

Pruning is a method which induces sparsity within an DNNs. This essentially means that a selection or subset of parameters within the network will have a value equal to zero. One of the biggest computational costs which dominate these networks is multiplication [2]. Hence, inducing sparsity through pruning reduces these costs through avoiding these multiplications. On top of this, storing these sparse models can be done in a more compact manner using sparse matrix formats.

This paper aims to evaluate how pruning can be applied to DNNs, methods of pruning and the limitations of pruning.

I. How is Pruning Implemented and what is the Winning Lottery Ticket Hypothesis?

Pruning within DNNs can be generally applied in a three stage process [1] shown in Figure 1.

Figure 1: *Procedure for implementing pruning within a Deep Neural Network.*

Initially a model is trained to the data, this model is most likely over-paramterized and hence computationally expensive. The next stage would be to apply a pruning technique to improve performance before finally fine-tuning the model in order to regain any significant loss in performance.

Knowing that pruning can help achieve desired attributes of DNNs, researchers have looked at ways in which they can improve the efficiency of the process (Figure 1) itself. Frankle and Carbin [3] proposed the idea that the training stage could be made to be more efficient by using a smaller model/architecture rather than using the original full model. This led to the **Lottery Ticket Hypothesis.** *A randomly-initialized, dense neural network contains a sub-network 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 [3].

What they were looking for was a subnetwork, when used instead of the full model would be more efficient in the training process but would also prove to yield results as good, if not better, than the full model (in terms of test accuracy). This desired subnetwork is referred to as the *winning ticket*. They arrived at a *winning ticket* through randomly initializing the neural network, training the network over a number of iterations (to arrive a certain parameters), performing iterative pruning on these parameters and then finally resetting the remaining parameters to what they were in the original model in order to create the *winning ticket*.

This dense subnetwork proved to be roughly 10-20% of the size of the original network whilst still maintaining competitive test-accuracy results. Not only this, but they were also able to conjecture that these dense subnetworks also performed better in training than sparse networks that were produced through pruning.

Frankle and Carbins work demonstrated an improvement in training performance but also allowed for a more in-depth knowledge of network structure and improvement in design. But their work was also limited. They found that through random initialization, these desired dense neural network learned more slowly and achieved a poorer accuracy, hence highlighting the importance of initialization. Another limitation was that they only performed these experiments on relatively small learning tasks (MNIST, CIFAR10) and not larger ones. However, this theory is still an active area of research [8].

II. Random Initialization

Frankle and Carbin saw that their *winning ticket* was dependent on the original initialization needed for it to achieve a competitive performance. *Rethinking the Value of Network Pruning [1]* looked further and saw that in fact in some cases, the reuse of the original initialization is not needed and that using random initialization in pruning is enough to achieve competitive performances.

Like Frankle and Carbin, these authors set out to improve the efficiency associated with training the model. They too found that training a large (most likely over-paramatized) model was counter intuitive. Their approach was to train a target model/architecture from scratch using structured pruning. Structured pruning is when pruning occurs at each channel or layer of the DNN. Their target model/architecture was either predefined by a human (i.e. human specifies 50% pruning at each channel/layer) or achieved through automatic pruning (i.e. pruning algorithm automatically determines the pruning criterion by analysing the entire structure of the DNN). They then analysed how long to train a model. They implied that it would be unfair to train a pruned model the same number of times as that of a full one due to it requiring significantly less computational power for one run through. They therefore decided to denote two *scratch* models. *Scratch-E* which denotes training a small pruned model the same number of times as the full model and *Scratch-D* which denotes training a small pruned model to the same computational requirements as that required by the full model. They trained these models from scratch on two large scale learning tasks. They found that training the model from scratch avoided implementing the pruning criterion (Figure 1) and hence the fine-tuning. Not only this, but training the models from scratch produced test-accuracy results that were better than fine-tuned models in most cases.

The authors then go further to test their *scratch* models (small target model trained from scratch) against the *winning ticket* model. The key differences in the results obtained thus far by both models are 1) The *scratch* model consists of structured pruning whilst the *winning ticket* is unstructured 2) The *scratch* model was trained on larger data-sets whilst the *winning ticket* was trained on smaller ones 3) They both use different learning rates. To compare both models the authors decided to apply both models to VGG-16 and ResNet-50 neural networks

with iterative and one-shot pruning applied to both models (both methods of pruning are unstructured). They also varied the learning rates from 0.01 to 0.1. Results can be shown in Figure 2.

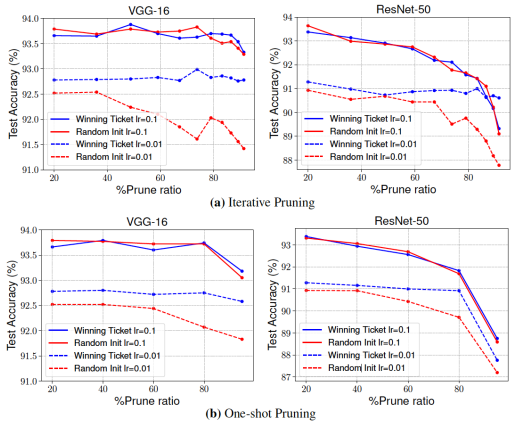

Figure 2: *Winning ticket vs Random Initialization [1]*

From the results we can see that the model trained with original initialization (*winning ticket*) only outperforms the model with random initialization (*scratch* model) when the learning rate is low. However, setting the learning rate to low levels is not desirable as it generally results in a lower test-accuracy. Hence this leads us to believe that for larger learning rates within larger data-sets, the *scratch* model produces more competitive results than both the original full model and the *winning ticket* model.

III. Methods For Inducing Sparsity

Having considered some of the open questions about how pruning should be implemented, we now look more in depth at the various methods of pruning. There are many different types of pruning. Some are based on simple rule based approaches such as Magnitude pruning, whilst others are more complicated in terms of both theory and application, such as Variational Dropout and L_0 Regularization [2].
Magnitude pruning revolves around the weight associated with each parameter. The parameters with the lowest associated weights will be removed (pruned) according to the criterion set out during the training. A successful implementation of this theory can be seen in the work done by Zhu and Gupta [4]. Their method essentially analyses the given weights in a layer within a DNN and sorts them in terms of their magnitude. The lower weights will then begin to have a *mask* placed over them until a desired level of sparsity is reached. This *mask* does not remove these lower weights but ensures that they are not included in the model. This is useful as these weights can then be "re-activated" at a later stage if the user desires. This method is also beneficial as it allows for a user specified level of sparsity.
Variational dropout essentially performs variation inference on the parameters of a Gaussian posterior over the weights of the model which were under a log-uniform prior [2]. This method analyses which of the parameters hold the highest dropout rates through training. These parameters are then removed from the model. *Variational Dropout Sparsifies Deep Neural Networks [5]* provides a method of variational dropout where all possible values of dropout rates are evaluated and hence this leads to a sparse solution.
Finally, L_0 Regularization is a method which encourages all weights during training to become exactly zero hence inducing sparsity [6]. However the norm of weights (L_0) is non-differentiable and hence this regularization term can not be directly applied to an objective function (a function which is desired to optimise in this case). *Learning Sparse Neural Networks through L_0 Regularization [6]* addresses this issue. The authors set out stochastic-gates sampled from hard concrete distributions which determine which of the parameters to set to zero. This is done through reparameterizing the weights withing the DNN as the product of the weight and the stochastic-gate variable which has been sampled from the hard concrete distribution.

IV. Which Method To Use?

The State of Sparsity in Deep Neural Net-works [2] is a paper which analyses the methods stated above in order to find the most desirable one. It applies all three methods to two large scale learning tasks: Transformer trained on WMT 2014 English-to-German, and ResNet-50 trained on ImageNet [2]. Doing this, this paper was one of the first to explore these methods on such large scale problems. Methods such as L_0 Regularization and Variational Dropout have already been proven to produce state-of-the-art compression on a much smaller scale but never applied to larger scale tasks.
The results are shown for both data-sets in Figures 3 and 4.

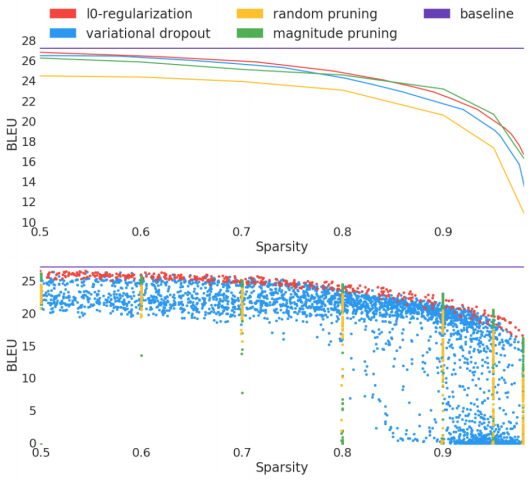

Figure 3: *Pruning methods applied to Transformer Data [2]*

From the Transformer data (Figure 3) we can see that L_0 Regularization and Variational Dropout outperform Magnitude pruning at low to mid sparsity levels, however magnitude pruning then achieves the best results at high sparsity levels.

When looking at the ResNet-50 results (Figure 4), one surprising omission is L_0 Regularization. This was due to the model quality being significantly damaged when L_0 Regularization was applied to the ResNet-50 neural network. We can also see that Variational dropout

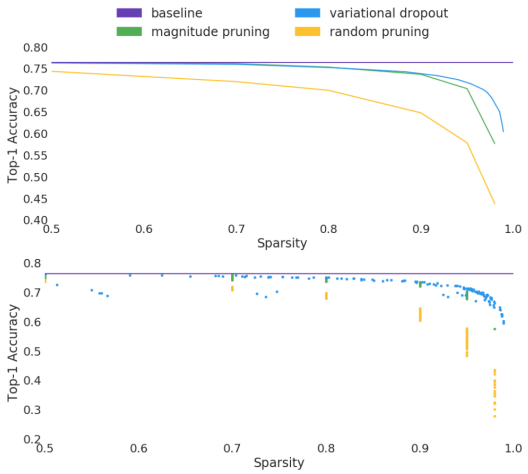

Figure 4: *Pruning methods applied to ResNet-50 Data [2]*

outperforms magnitude pruning. However, when Resnet-50 is trained using this method the training time requires around 9.75 hours [2] to complete compared to the 3.15 hours taken by magnitude pruning. As a result a new training set up was introduced for ResNet-50 regarding the use of magnitude pruning in order to see how well this method could perform if it distributed the non-zero weights more carefully and increased its training time. The number of training steps for magnitude pruning were therefore increased 1.5x.

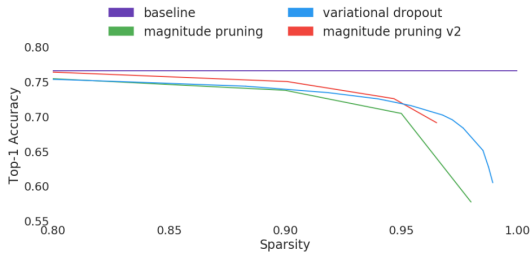

Figure 5: *Pruning methods applied to ResNet-50 with enhanced Magnitude Pruning [2]*

Figure 5 shows us that magnitude pruning now outperforms variational dropout until the highest sparsity levels whilst also exercising fewer computational resources. Overall this paper is informative in showing us that not only is magnitude pruning a simpler concept

and approach to pruning, but also yields highly competitive results.

This paper also went and looked at the two theories we have previously covered, *scratch* and *winning lottery ticket*, and compared their performance against a model learned through sparsification as part of the optimization process (magnitude pruning). Hence, these three methods were applied to the ResNet-50 neural network.

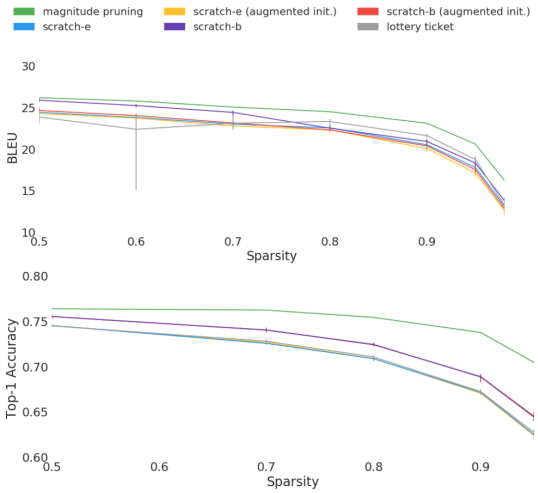

Figure 6: *Pruning methods applied to ResNet-50 with enhanced Magnitude Pruning [2]*

As we can see in Figure 6, magnitude pruning outperforms both previously stated theories. This paper therefore highlights strong counterexamples to the *scratch* and *winning lottery ticket* theories whilst also establishing a solid foundation for large scale pruning tasks.

V. The Impact of Pruning

Having looked at various ways for implementing pruning and different methods of pruning itself, *Selective Brain Damage: Measuring the Disparate Impact of Model Pruning* looks at the effects of pruning. The overall test-set accuracy after pruning may be at a high level but beneath that, this paper looked at how different classes and images are adversely affected by pruning compared to others.

This paper gauges the audience well by introducing an interesting comparative. It notes that during our childhood (from 2-10) we lose roughly 50% of our synapses in our brain and yet we are still able to function. They apply this logic to DNN pruning and ask the question *"What is lost when we prune a Deep Neural Network?" [7]* whilst also describing pruning on DNN as *"selective brain damage" [7]*.
Through their studies they denote classes systematically more impacted by pruning as *pruning identified exemplars (PIE) [7]*. Throughout their studies they used the ResNet-50 neural network trained on the ImageNet dataset (having applied magnitude pruning). One of their most interesting discoveries was that both pruned and unpruned (full) models struggled when it came to identifying *PIEs*.

true: cloak
baseline model: gasmask
pruned model: breastplate

ladle
ladle
perfume

Figure 7: *Visualization of PIE for ImageNet dataset [7]*

From Figure 7 we can see that although the unpruned (baseline) model correctly identifies some classes/images, there are some classes which it still struggles with. The pruned model incorrectly classifies both, highlighting the original theory that different classes and images are adversely affected by pruning compared to others. Reasons for both models struggling may be down to multiple objects being present in the image, image corruption or an abstract representation (these all constitute as PIE).
Not only does this paper highlight that extra caution should be taken when applying pruning (especially in real world applications which may affect human welfare [7]), but it also introduces an interesting topic which requires further research. How do we deal with PIE for

both pruned and unpruned models?

VI. Conclusion

We have seen how important pruning can be for the improvement in computational efficiency associated with Deep Neural Networks. However we have also seen that multiple research articles have provided counterexamples to previous (recent) work. This highlights a need for a rigorous benchmark to be set within this field of research relating to the implementation and choice of pruning.
Further research is also needed to address the issues associated with *pruning identified exemplars*.

VII. Q&A With Researcher, Sara Hooker [2,7]

The full audio interview between me and Google Brain researcher, Sara Hooker, can be found here: Sara Hooker Audio Interview.

References

[1] Liu, Z., Sun, M., Zhou, T., Huang, G. and Darrell, T. (2018). Rethinking the Value of Network Pruning. *CoRR*, abs/1810.05270.

[2] Gale, S., Elsen, E. and Hooker, S. (2019). The State of Sparsity in Deep Neural Networks. *CoRR*, abs/1902.09574.

[3] Frankle, J. and Carbin, M. (2018). The Lottery Ticket Hypothesis: Training Pruned Neural Networks. *CoRR*, abs/1803.03635.

[4] Zhu, M. and Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. *CoRR*, abs/1710.01878.

[5] Molchanov, D., Ashukha, A. and Vetrov, D. (2017). Variational Dropout Sparsifies Deep Neural Networks. *CoRR*, abs/1701.05369.

[6] Louizos, C., Welling, M. and Kingma, D. (2018). Learning Sparse Neural Networks through L_0 Regularization. *CoRR*, abs/1712.01312.

[7] Hooker, S., Courville, A., Dauphin, Y. and Frome, A. (2019). Selective Brain Damage: Measuring the Disparate Impact of Model Pruning. *CoRR*, abs/1911.05248.

[8] Evci, U., Gale, T., Menick, J., Castro, P. and Elsen E. (2019). Rigging the Lottery: Making All Tickets Winners. *CoRR*, abs/1911.11134.