DropNet: Reducing Neural Network Complexity via Iterative Pruning (Supplementary Material)

1. Summary

In this supplementary material, we: (i) present more results for Model C (Figs. 2, 3, 4, 5, 6, 7), ResNet18 (Figs. 8, 9) and VGG19 (Figs. 10, 11) and (ii) explore the effect of random initialization on ResNet18 (Figs. 12, 13) and VGG19 (Figs. 14, 15). The results show that *DropNet* is robust for larger models, and the final pruned model is able to achieve a similar performance even after reinitialization.

2. Methodology

The supplementary experiments performed use the same CIFAR-10 dataset and methodology as the main paper. We demonstrate how effective pruning using *DropNet* can be done on larger models like Model C (Conv4), ResNet18 and VGG19. For ResNet18 and VGG19, the model architecture follow closely from the original papers (Simonyan & Zisserman, 2014; He et al., 2016) and are detailed in Fig. 1.

3. Experiments

3.1. CNN - CIFAR-10: Model C (Conv4)

Q1. Can DropNet perform robustly well on larger CNNs of various starting configurations?

To address this question, we conduct an experiment using Algorithm 1 for various configurations of Model C on CIFAR-10, listed as follows:

1.1) **Model C: Conv64 - Conv64 - Conv128 - Conv128**. The plot of training and test accuracy against fraction of filters remaining for various metrics are shown in Figs. 2 and 5 respectively.

1.2) **Model C: Conv128 - Conv128 - Conv128 - Conv128**. The plot of training accuracy and test accuracy against fraction of filters remaining for various metrics are shown in Figs. 3 and 6 respectively.

1.3) **Model C: Conv128 - Conv128 - Conv64 - Conv64**. The plot of training accuracy and test accuracy against fraction of filters remaining for various metrics are shown in Figs. 4 and 7 respectively.

For 1.1), it can be seen (Figs. 2 and 5) that the `minimum_layer` metric performs the best, followed by `minimum`, `random_layer`, `random`, and `maximum_layer` and lastly `maximum` metric. The `minimum` and `minimum_layer` perform equally well when the fraction of filters remaining is 0.3 and above. The `maximum` metric can be seen to be consistently poor when the fraction of filters remaining is 0.5 and below. The `random` metric is in between the performance of the `minimum` and `maximum` metrics.

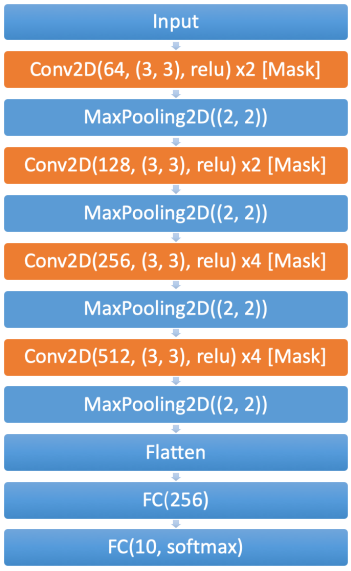
For 1.2), it can be seen (Figs. 3 and 6) that the `minimum_layer` metric performs the best, followed by `random_layer`, `random`, `minimum`, `maximum_layer` and lastly `maximum` metric. The `minimum` and `minimum_layer` perform equally well when the fraction of filters remaining is 0.3 and above. The `maximum` metric can be seen to be consistently poor when the fraction of filters remaining is 0.6 and below.

For 1.3), it can be seen (Figs. 4 and 7) that the `minimum_layer` metric performs the best, followed by `minimum`, `random_layer`, `random`, and `maximum_layer` and lastly `maximum` metric. The `minimum` and `minimum_layer` perform equally well when the fraction of filters remaining is 0.4 and above, with the `minimum` metric performing significantly better when the fraction of filters remaining is 0.6 and above, even outperforming the original model accuracy at some instances. The `maximum` metric can be seen to be consistently poor when the fraction of filters remaining is 0.8 and below.

Evaluation: The results show that `minimum` and `minimum_layer` are both competitive when less than of half of the filters are dropped. Thereafter, `minimum_layer` performs significantly better. Using *DropNet*, we can reduce the number of filters by 50% or more without significantly affecting model accuracy, highlighting its effectiveness in reducing network complexity.

The results indicate that, for larger convolutional models like Model C, global pruning methods like the `minimum` metric are only good at the early stages of pruning. In fact, for models with non-symmetric layers (see Figs. 4 and 7), `minimum` works the best when fraction of filters remaining is 0.5 and above, and may even outperform the original model's accuracy. This is due to the flexibility of global pruning methods to avoid pruning small layers which pose a bottleneck as compared to layer-wise pruning

ResNet18



VGG19

Figure 1. Architecture of ResNet18 (He et al., 2016) and VGG19 (Simonyan & Zisserman, 2014) used in the experiments. The layers where the masks are applied are written with a postfix '[Mask]', and shown in orange. ***ResNet18 (Left)***: The network architecture closely follows that of ResNet18. It consists of several skip-connection blocks which are shown in square brackets. The model consists of repeated residual blocks comprising two 2D convolutional layers followed by a MaxPooling2D layer of stride 2. Each 2D convolutional layer comprises either 64, 128, 256 or 512 filters, of size 3x3 with 'same' padding. After the multiple residual blocks, the filters are averaged using GlobalAveragePooling2D before passing into the final fully connected layer with 10 nodes. The mask is applied after the convolutional layer and before the MaxPooling2D layer. Batch Normalization is not applied between layers as the model is found to work well even without it. ***VGG19 (Right)***: This is a network with repeated blocks of 2/4 2D convolutional layers followed by a MaxPooling2D layer. The 2D convolutional layer comprises either 64, 128, 256 or 512 filters, of size 3x3 with 'same' padding. The mask is applied after the convolutional layer and before the MaxPooling2D layer. Batch normalization is applied right before every MaxPooling2D layer in order to counter the vanishing gradient effect.

methods. That said, not all global pruning methods can do that - `maximum` and `random` do not display such a trend of avoiding bottlenecks.

One further observation is that the train and test data show similar accuracy trends (the same applies for validation accuracy, although not shown here). This shows that the train-test-validation split is done well and the general distribution of the train dataset is similar to that of the test dataset. Hence, a metric to prune based on the node's post-activation value such as *DropNet* works well using just the post-activation values from the training data only.

3.2. CNN - CIFAR-10: ResNet18/VGG19

Q2. Can DropNet perform robustly well on even larger models such as ResNet18 and VGG19?

To address this question, we conduct an experiment using Algorithm 1 for ResNet18 and VGG19 on CIFAR-10:

2.1) **ResNet18**. The plot of training and test accuracy against fraction of filters remaining for various metrics is shown in Figs. 8 and 9 respectively.

2.2) **VGG19**. The plot of training and test accuracy against fraction of filters remaining for various metrics is shown in Figs. 10 and 11 respectively.

For ResNet18, it can be seen (Figs. 8 and 9) that the `minimum_layer` metric performs the best, followed by `minimum`, then `random_layer`, `random`, `maximum_layer` and and lastly `maximum` metric. The `minimum_layer` and `minimum` are both competitive.

For VGG19, it can be seen (Figs. 10 and 11) that the `minimum_layer` metric performs the best, followed by `random_layer`, `random`, `max_layer`, `minimum`, and and lastly `maximum` metric. The `minimum_layer` metric is the most competitive.

For both ResNet18 and VGG19, `maximum` can be seen to be consistently poor when the fraction of filters remaining is 0.5 and below, while `maximum_layer` is consistently poor when the fraction of filters remaining is 0.2 and below.

Evaluation: The results show that for larger models, the `minimum_layer` is the most competitive. It can also be seen that with the exception of `minimum` and

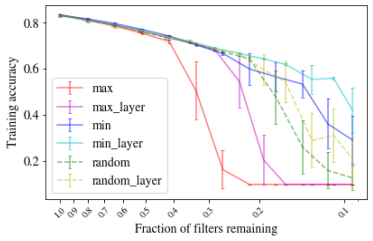


Figure 2. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv64 - Conv64 - Conv128 - Conv128 on CIFAR-10

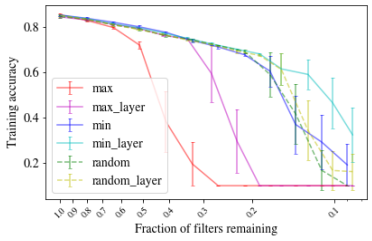


Figure 3. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv128 - Conv128 on CIFAR-10

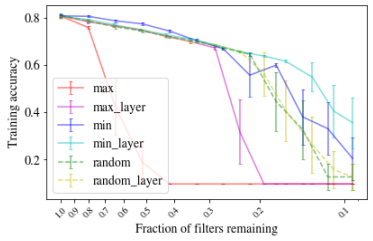


Figure 4. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv64 - Conv64 on CIFAR-10

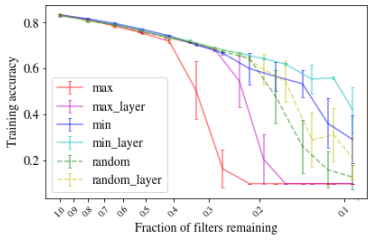


Figure 5. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv64 - Conv64 - Conv128 - Conv128 on CIFAR-10

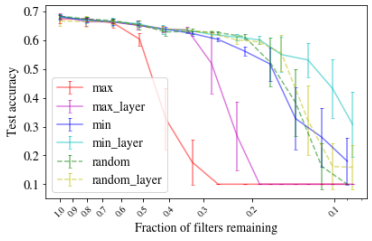


Figure 6. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv128 - Conv128 on CIFAR-10

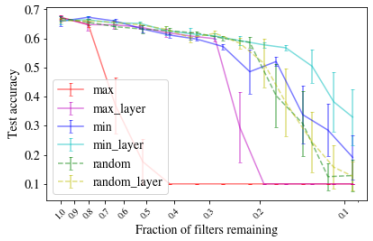


Figure 7. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv64 - Conv64 on CIFAR-10

`maximum_layer`, the layer-wise metrics outperform the global metrics for larger models. This shows that there may be significant statistical differences between layers for larger models such that comparing magnitudes across layers may not be a good way to prune nodes/filters. That said, the `minimum_layer` can be seen to perform very well and consistently performs better than random, which shows promise that it is a good metric.

The `minimum` metric proves to be almost as competitive as `minimum_layer` for ResNet18, but performs worse than random for VGG19. This shows that the skip connections in ResNet18 does help to alleviate some of the pitfalls of global metrics. Interestingly, the `minimum` metric tends to prune out some skip connections completely, which shows that certain skip connections are unnecessary. This means that *DropNet* using the `minimum` metric is able to automatically identify these redundant connections on its own.

In comparison, it can be seen that the `maximum` metric performs the worse in all cases, and shows that filters with high expected absolute post-activate values are generally important in classification and should not be removed.

The `maximum_layer` metric on the other hand, performs poorly in ResNet18, but has comparable performance to the layer-wise metrics in VGG19. This may be due to the fact that VGG19 in the experiments use a Batch Normalization after every change of Conv2D filter size, which helps to normalize the post-activation values and hence, the layer-wise pruning metrics do not differ much in performance.

Using *DropNet*, we can reduce the number of filters by 80% or more without significantly affecting model accuracy, highlighting its effectiveness in reducing network complexity.

3.3. CNN - CIFAR-10: Random Initialization

Q3. Is the starting initialization of weights and biases important for larger models such as ResNet18 and VGG19?

We compare the performance of a network retaining its initial weights and biases θ_0 when performing iterative node/filter pruning, as compared to a network with the pruned architecture but with a random initialization (`randominit`). In our experiments, we focus on the pruned architecture produced by *DropNet* metrics, namely `minimum` and `minimum_layer`. The experiments are conducted on CIFAR-10, and are detailed as follows:

3.1) **ResNet18**. The plot of test accuracy against fraction of nodes remaining for original and random initialization using pruned model from `minimum` metric and `minimum_layer` metric respectively are shown in

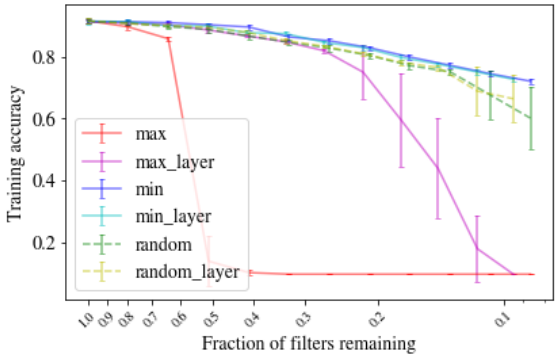


Figure 8. Plot of training accuracy against fraction of filters remaining for various metrics in ResNet18 on CIFAR-10

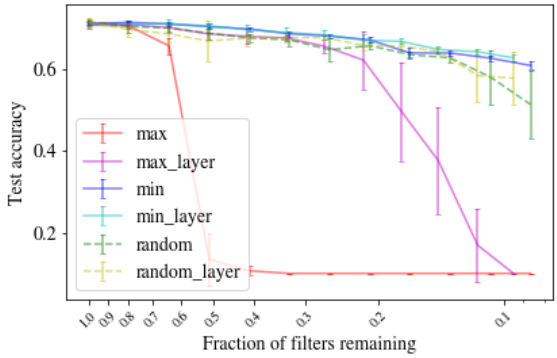


Figure 9. Plot of test accuracy against fraction of filters remaining for various metrics in ResNet18 on CIFAR-10

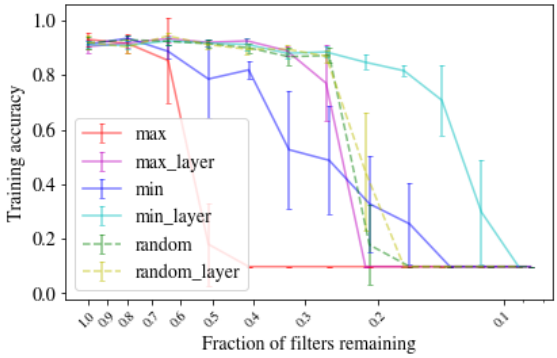


Figure 10. Plot of training accuracy against fraction of filters remaining for various metrics in VGG19 on CIFAR-10

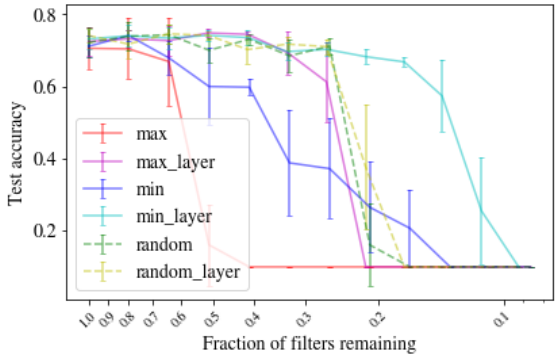


Figure 11. Plot of test accuracy against fraction of filters remaining for various metrics in VGG19 on CIFAR-10

Figs. 12 and 13.

3.2) **VGG19**. The plot of test accuracy against fraction of nodes remaining for original and random initialization using pruned model from `minimum` metric and `minimum_layer` metric respectively are shown in Figs. 14 and 15.

It can be seen (Figs. 12, 13, 14, 15) that unlike the Lottery Ticket Hypothesis (see Figure 4 in (Frankle & Carbin, 2018)), *DropNet* does not suffer from loss of performance when randomly initialized.

Evaluation: This means that for *DropNet*, only the final pruned network architecture is important, and not the initial weights and biases of the network. This is a pleasant finding as it shows that *DropNet* can prune a model down to an ideal structure, from which it can be readily deployed on modern machine learning libraries.

4. Concluding Remarks

The results show that *DropNet* has statistically significant performance over random pruning, even for larger models such as ResNet18 and VGG19. More experiments are underway to test out on larger datasets, but preliminary results show promise that *DropNet* is a highly-effective general-purpose pruning algorithm able to prune up to 90% or more of nodes/filters without significant loss of accuracy.

Global or layer-wise pruning: As shown in the main paper, if we are pruning small models such as Model A or Model B, `minimum` works well. Furthermore, we show here that even in larger models, `minimum` shows promise in avoiding pruning bottlenecks as compared to its layer-wise counterpart `minimum_layer`, and gives significantly better performance if we are just pruning a small fraction of the original model. However, when pruning even larger models such as ResNet18 and VGG19, we show that it is better to use `minimum_layer` instead. One reason for this is that the statistical properties of the post-activation values of each layer may differ significantly as the model grows

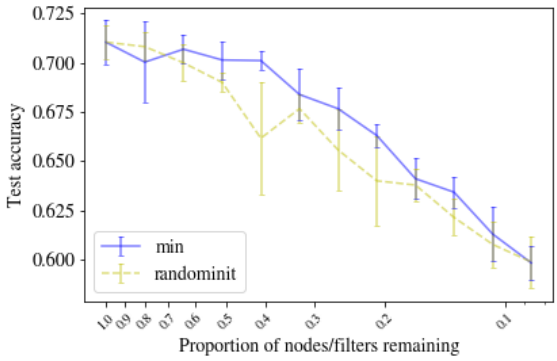


Figure 12. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in ResNet18 using pruned model from `minimum` metric on CIFAR-10

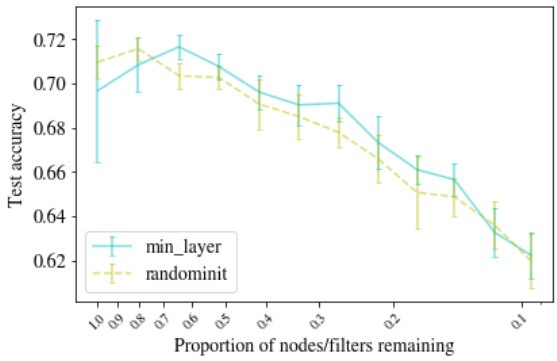


Figure 13. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in ResNet18 using pruned model from `mininum_layer` metric on CIFAR-10

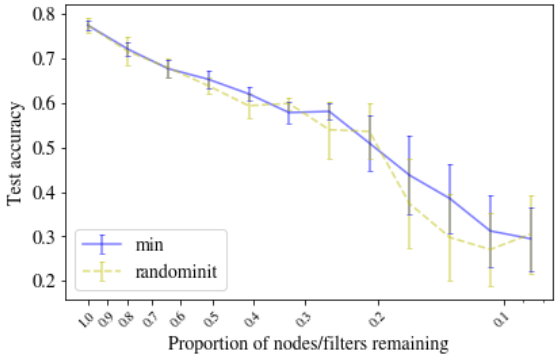


Figure 14. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in VGG19 using pruned model from `minimum` metric on CIFAR-10

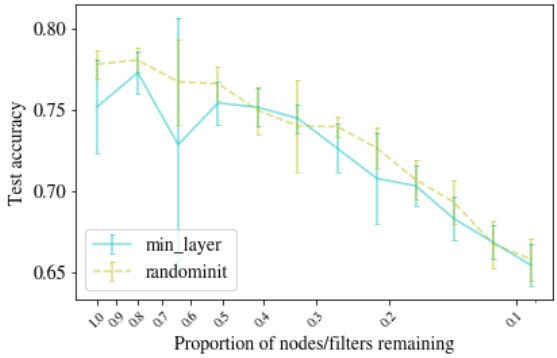


Figure 15. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in VGG19 using pruned model from `minimum_layer` metric on CIFAR-10

large, and a global metric for all the layers may not work as well. That said, the empirical results of ResNet18 show that `minimum` can be competitive as well, which suggests that skip connections may be able to alleviate the pitfalls of global metrics.

References

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.