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

1. Summary

In this supplementary material, we present the following:

1. Results using more variants of Model C (Figs. 2, 3, 4, 5, 6, 7) on the CIFAR-10 dataset
2. Results using ResNet18 (Figs. 8, 9) and VGG19 (Figs. 10, 11) on the CIFAR-10 dataset
3. Results of random initialization of ResNet18 (Figs. 12, 13) and VGG19 (Figs. 14, 15) on the CIFAR-10 dataset
4. Results using ResNet18 (Figs. 16, 17) and VGG19 (Figs. 18, 19) on the Tiny ImageNet dataset
5. Performance comparison to Average Percentage of Zeros (APoZ) (Hu et al., 2016) for ResNet18 (Figs. 20, 21) and VGG19 (Figs. 22, 23) on the CIFAR-10 dataset

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. DropNet also has better empirical performance than prior data-driven approach APoZ and is able to achieve better test accuracy for the same amount of pruning.

2. Methodology

The supplementary experiments performed use the same methodology as the main paper. In addition, to show *DropNet*'s scalability, we also perform experiments on the Tiny ImageNet dataset. 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 follows closely from the original papers (Simonyan & Zisserman, 2014; He et al., 2016) and are detailed in Fig. 1.

Algorithm 1, which is used throughout the supplementary material, is detailed in the main paper.

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.

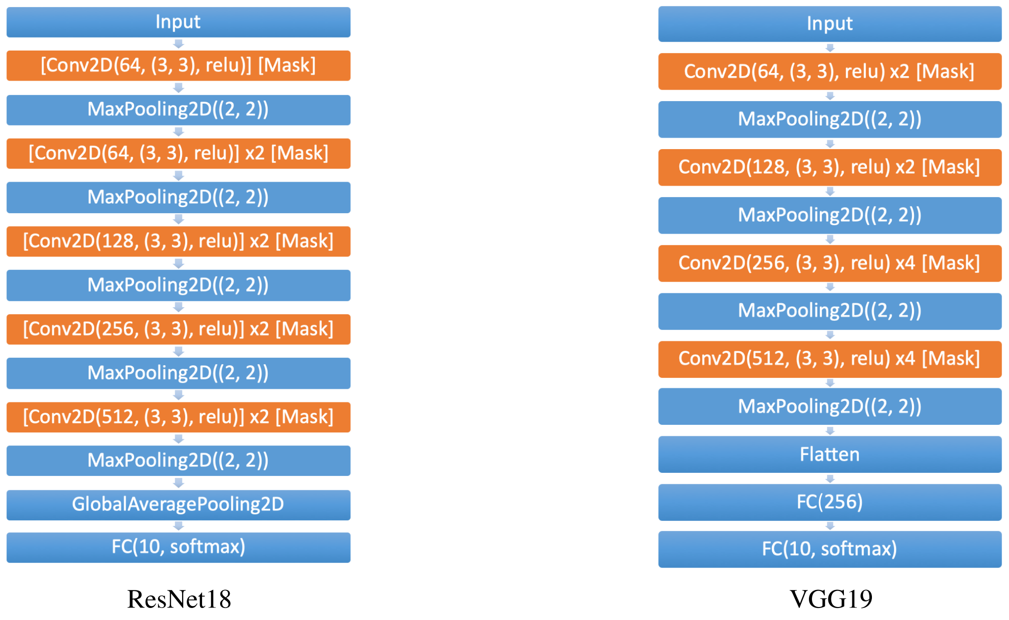
Input
[Conv2D(64, (3, 3), relu)] [Mask]
MaxPooling2D((2, 2))
[Conv2D(64, (3, 3), relu)] x2 [Mask]
MaxPooling2D((2, 2))
[Conv2D(128, (3, 3), relu)] x2 [Mask]
MaxPooling2D((2, 2))
[Conv2D(256, (3, 3), relu)] x2 [Mask]
MaxPooling2D((2, 2))
[Conv2D(512, (3, 3), relu)] x2 [Mask]
MaxPooling2D((2, 2))
GlobalAveragePooling2D
FC(10, softmax)
ResNet18
Input
Conv2D(64, (3, 3), relu) x2 [Mask]
MaxPooling2D((2, 2))
Conv2D(128, (3, 3), relu) x2 [Mask]
MaxPooling2D((2, 2))
Conv2D(256, (3, 3), relu) x4 [Mask]
MaxPooling2D((2, 2))
Conv2D(512, (3, 3), relu) x4 [Mask]
MaxPooling2D((2, 2))
Flatten
FC(256)
FC(10, softmax)
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.

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 the fraction of filters remaining is 0.5 and above, and may even outperform the original model's accuracy. We posit that 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 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

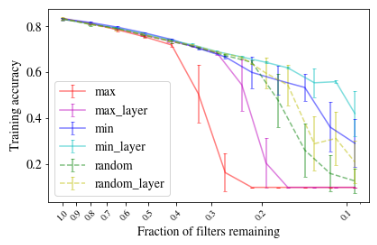

Figure 2. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv64 - Conv64 - Conv128 - Conv128 on CIFAR-10

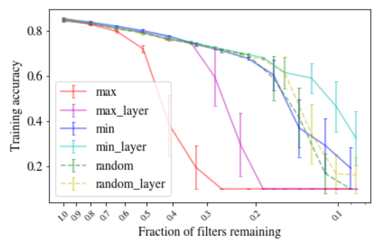

Figure 3. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv128 - Conv128 on CIFAR-10

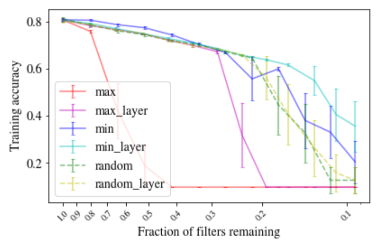

Figure 4. Plot of training accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv64 - Conv64 on CIFAR-10

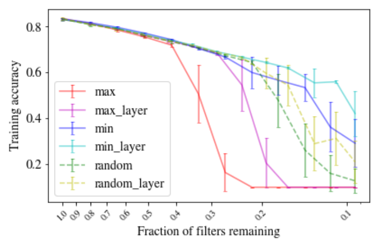

Figure 5. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv64 - Conv64 - Conv128 - Conv128 on CIFAR-10

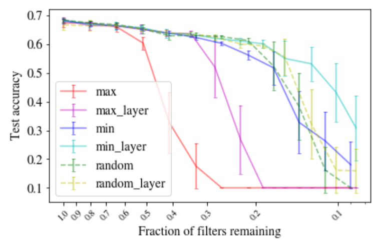

Figure 6. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv128 - Conv128 on CIFAR-10

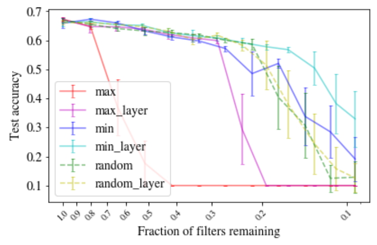

Figure 7. Plot of test accuracy against fraction of filters remaining for various metrics in Model C: Conv128 - Conv128 - Conv64 - Conv64 on CIFAR-10

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, `minimum_layer` is the most competitive. It can also be seen that with the exception of `minimum` and `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.

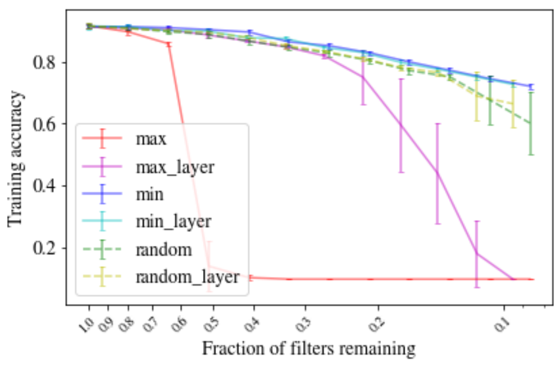

Figure 8. Plot of training accuracy against fraction of filters remaining for various metrics in ResNet18 on CIFAR-10

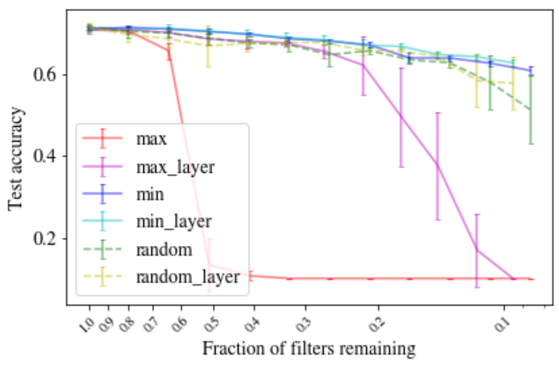

Figure 9. Plot of test accuracy against fraction of filters remaining for various metrics in ResNet18 on CIFAR-10

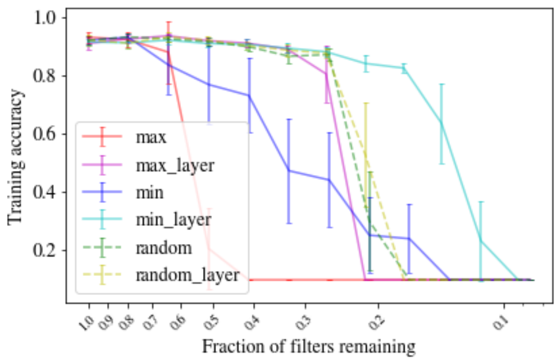

Figure 10. Plot of training accuracy against fraction of filters remaining for various metrics in VGG19 on CIFAR-10

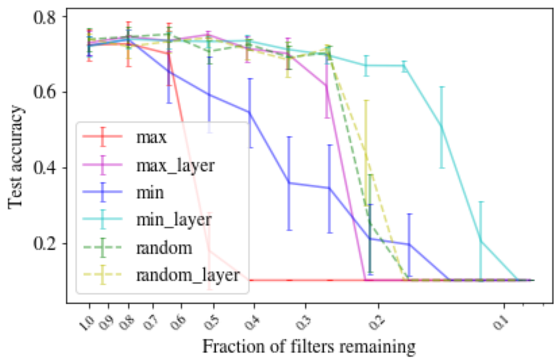

Figure 11. Plot of test accuracy against fraction of filters remaining for various metrics in VGG19 on CIFAR-10

3.3. CNN - CIFAR-10: ResNet18/VGG19 (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 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.

3.4. CNN - Tiny ImageNet: ResNet18/VGG19

Q4. Can DropNet perform well for even larger datasets such as Tiny ImageNet?

In order to show the generalizability of the *DropNet* algorithm on larger datasets, we utilize the Tiny ImageNet dataset, which is a smaller-resolution parallel of the larger

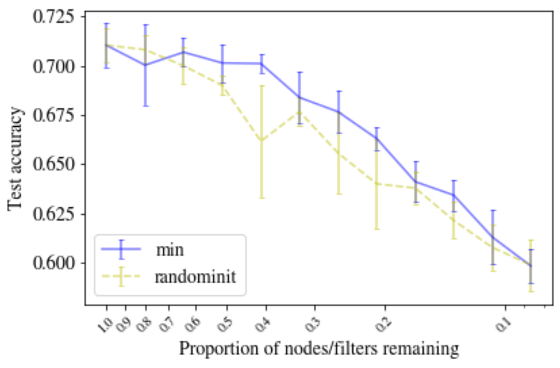

Figure 12. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in ResNet18 using pruned model from `minimum` metric on CIFAR-10

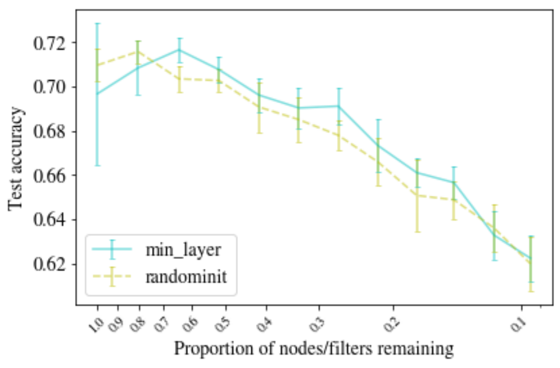

Figure 13. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in ResNet18 using pruned model from `mininum_layer` metric on CIFAR-10

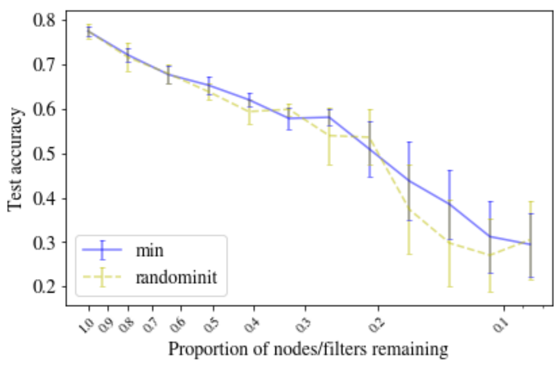

Figure 14. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in VGG19 using pruned model from `minimum` metric on CIFAR-10

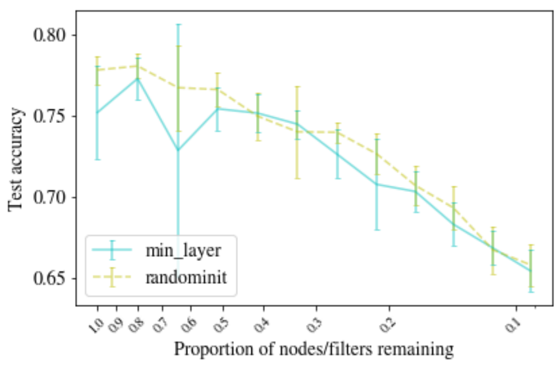

Figure 15. Plot of test accuracy against fraction of filters remaining for original initialization and random initialization in VGG19 using pruned model from `minimum_layer` metric on CIFAR-10

ImageNet dataset. This dataset was taken from `https://tiny-imagenet.herokuapp.com/`.

Dataset Details: Tiny Imagenet has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. Each image has a resolution of 64 pixels by 64 pixels by 3 channels.

Changes to model parameters: We utilize a similar model for ResNet18 and VGG19 as in the earlier experiment with CIFAR-10. We only modify the models slightly in order to cater for the 200 output classes of Tiny ImageNet, which is an increase from the 10 output classes in CIFAR-10. As such, the last linear layer for ResNet18 has 200 nodes (instead of 10 nodes for CIFAR-10), while the last two linear layers for VGG19 has 1024 nodes and 200 nodes respectively (instead of 256 nodes and 10 nodes respectively for CIFAR-10).

Table 1. Image Augmentation Parameters

METRIC	VALUE
ROTATION	40 DEGREE
WIDTH SHIFT RANGE	0.2
HEIGHT SHIFT RANGE	0.2
ZOOM RANGE	0.2
SHEAR RANGE	0.2
FLIPPING	HORIZONTAL

Due to the complexity of this dataset, in order to attain better train/test accuracies, we apply image augmentation to each data sample per epoch. The image augmentation parameters applied are shown in Table 1.

In order to give the model more time to converge for this larger dataset, we also increase the number of epochs before early stopping to 10 (instead of 5 in earlier experiments).

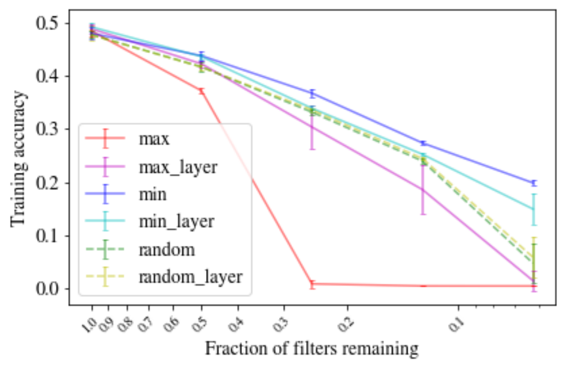

Figure 16. Plot of training accuracy against fraction of filters remaining for various metrics in ResNet18 on TinyImageNet

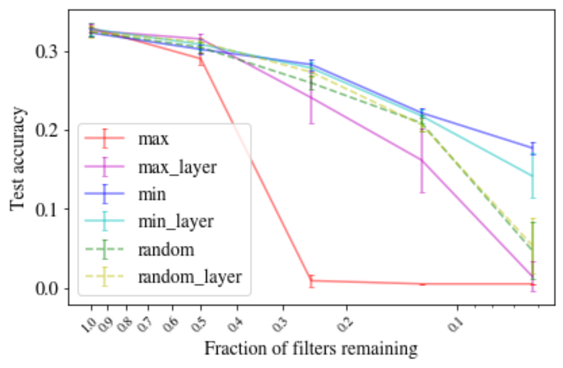

Figure 17. Plot of test accuracy against fraction of filters remaining for various metrics in ResNet18 on TinyImageNet

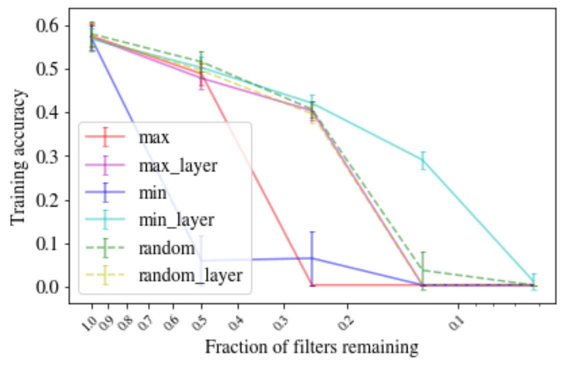

Figure 18. Plot of training accuracy against fraction of filters remaining for various metrics in VGG19 on TinyImageNet

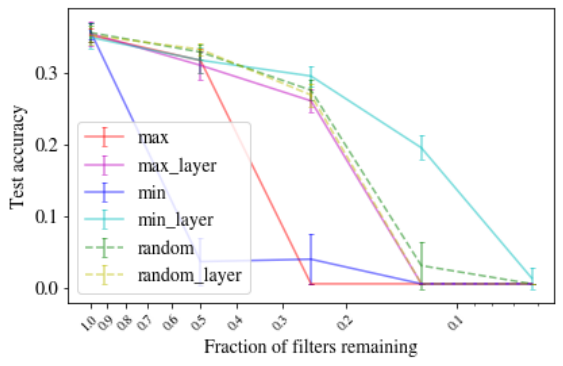

Figure 19. Plot of test accuracy against fraction of filters remaining for various metrics in VGG19 on TinyImageNet

Also, in order to reduce experimental running time for Tiny ImageNet, we drop at each pruning cycle a fraction 0.5 of the filters (instead of 0.2 for CIFAR-10).

Experiments: We conduct an experiment using Algorithm 1 for ResNet18 and VGG19 on Tiny ImageNet:

4.1) **ResNet18**. The plot of training and test accuracy against fraction of filters remaining for various metrics is shown in Figs. 16 and 17 respectively.

4.2) **VGG19**. The plot of training and test accuracy against fraction of filters remaining for various metrics is shown in Figs. 18 and 19 respectively.

For ResNet18, it can be seen (Figs. 16 and 17) that `minimum` metric and `minimum_layer` metrics have the most competitive performance, followed by `random_layer`, `random`, `maximum_layer` and and lastly `maximum` metric. The `minimum` metric is the most competitive.

For VGG19, it can be seen (Figs. 18 and 19) that the `minimum_layer` metric performs the best, followed by `random`, `random_layer`, `max_layer`, `aximum`, and and lastly `minimum` 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 0.3 and below. Surprisingly, `minimum` has the poorest performance for VGG19.

Evaluation: The results show that for ResNet18, `minimum` performs the best. This is similar to the results obtained on the CIFAR-10 dataset (Refer to Figs. 8 and 9). This could be the fact that the skip connections in ResNet18 allow the pruned model to perform well even if majority of the layer is removed, and gives greater redundancy for pruning. With such redundancy, some of the pitfalls of global pruning methods can be alleviated.

For VGG19, `minimum_layer` performs the best, while `minimum` performs the worst. This could again be because there is no redundancy in the layers for VGG19 and if one layer gets pruned aggressively by a global metric, it might affect performance negatively. The drop in performance of `minimum` is worse in the Tiny ImageNet dataset as compared to the CIFAR-10 dataset (Refer to Figs. 10 and 11) likely because the proportion of filters dropped per pruning cycle is larger at 0.5 as compared to 0.2, hence there is a greater chance of a layer getting pruned aggressively.

Similar to the CIFAR-10 experiments, the experiments on the larger Tiny ImageNet dataset also suggest that the `minimum` and `minimum_layer` are both competitive in ResNet18. For most models, the layer-wise `minimum_layer` metric is a general all-round metric to be used.

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.

4. Benchmarking against APoZ

Q5. Does DropNet perform better than prior data-driven pruning methods?

While we utilize an oracle for smaller model sizes such as Model A and B, in order to evaluate the effectiveness of *DropNet* for larger models, we compare its performance to a similar data-driven metric known as Average Percentage of Zeros (APoZ) (Hu et al., 2016).

APoZ measures the percentage of zero activations of a neuron after a ReLU activation function. The neuron/filter with the highest percentage of zero activations is considered least important and is pruned first.

In their original paper (Hu et al., 2016), APoZ used a variant of layer-wise pruning, where they first prune "a few layers with high mean APoZ, and then progressively trim its neighboring layers". In order to keep the methodology consistent to that of *DropNet*, we adapt the same APoZ metric of percentage of zero activations of a neuron/filter after a ReLU activation, but use *DropNet*'s layer-wise and global-wise iterative pruning approaches as depicted in Algorithm 1. APoZ using layer-wise pruning is termed `apoz_layer`, while APoZ using global pruning is termed `apoz`. We compare its performance to DropNet's layer-wise pruning `minimum_layer` and global pruning `minimum`.

We conduct an experiment to compare *DropNet* and APoZ using Algorithm 1 for ResNet18 and VGG19 on CIFAR-10:

5.1) **ResNet18**. The plot of training and test accuracy against fraction of filters remaining for *DropNet* and APoZ is shown in Figs. 20 and 21 respectively.
5.2) **VGG19**. The plot of training and test accuracy against fraction of filters remaining for *DropNet* and APoZ is shown in Figs. 22 and 23 respectively.

For ResNet18, it can be seen (Figs. 20 and 21) that the `minimum_layer` and `apoz_layer` both perform the best, followed closely by `minimum`, then the `apoz`. After a fraction of 0.7 or more filters are pruned, the `apoz` metric suffers a huge performance drop.

For VGG19, it can be seen (Figs. 22 and 23) that the `minimum_layer` performs the best, followed by `apoz_layer`, then `minimum` and finally `apoz`. After a fraction of 0.3 or more filters are pruned, the global metrics `apoz` and `minimum` suffer a huge performance drop.

Evaluation: The results show that DropNet in general outperforms APoZ, both layer-wise and globally. In general, for the same amount of filters pruned, *DropNet* achieves higher test accuracy than APoZ. The `minimum_layer` metric is consistently the best performing metric across both ResNet18 and VGG19 models, outperforming `apoz_layer`. For global metrics, the `minimum` metric also consistently outperforms `apoz`. Notably, while the `minimum` metric has good performance for ResNet18, the same performance is not seen in `apoz`.

This shows that *DropNet* has merit as a data-driven pruning approach, as it captures more information about the importance of a particular node/filter through the use of the expected absolute value. This is an improvement from APoZ, as it also takes into account the magnitudes of the post-activation values, rather than just only relying on the percentage of zero activations of a node/filter.

5. Concluding Remarks

The results show that *DropNet* shows significantly better performance than random pruning, even for larger models such as ResNet18 and VGG19. *DropNet* also manages to achieve higher test accuracy for the same amount of pruning as compared to prior work APoZ, highlighting its competency. *DropNet* is a highly-effective general-purpose pruning algorithm able to work on datasets of varying sizes such as MNIST, CIFAR-10 and Tiny ImageNet. Overall, *DropNet* is 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 in large models such as Model C, `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 may be that the statistical properties of the post-activation values of each layer may differ significantly as the model grows large, and a global metric for all the layers

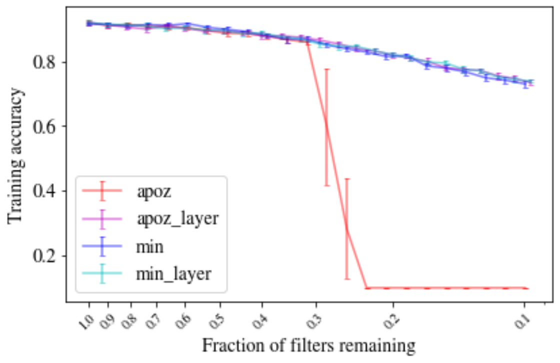

Figure 20. Plot of training accuracy against fraction of filters remaining for DropNet and APoZ in ResNet18 on CIFAR-10

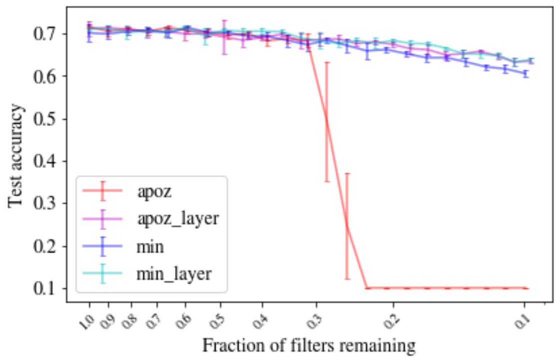

Figure 21. Plot of test accuracy against fraction of filters remaining for DropNet and APoZ in ResNet18 on CIFAR-10

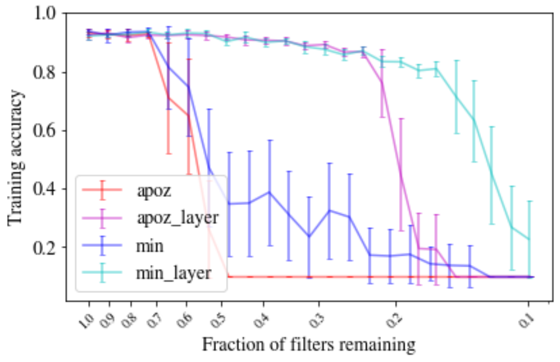

Figure 22. Plot of training accuracy against fraction of filters remaining for DropNet and APoZ in VGG19 on CIFAR-10

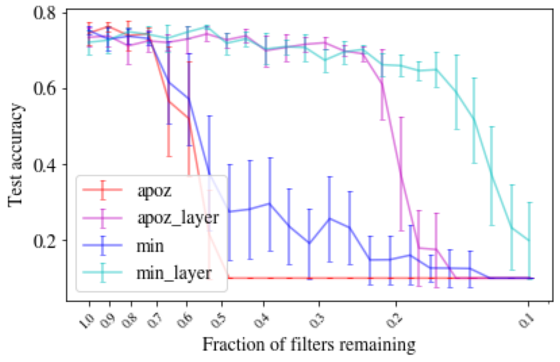

Figure 23. Plot of test accuracy against fraction of filters remaining for DropNet and APoZ in VGG19 on CIFAR-10

may not work as well. That said, the empirical results of ResNet18 show that `minimum` can be competitive as well for these larger models, 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.

Hu, H., Peng, R., Tai, Y., Tang, C., and Trimming, N. A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

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