Deep Learning

EndSem Exam Part - 1

Name	Addepalli Bharat Sai
Roll No.	B19BB002

Task - Choose a research paper in the time of 2019-Till Date from the prestigious journals in Deep Learning like NeurIPS, ICLR, ICML, and AAAI. Reproduce their results and use their model on any different database where the model can be used.

Paper Title - DropNet: Reducing Neural Network Complexity via Iterative Pruning (ICML/2020)

Summary of the Algorithm -

Iterative Pruning -

Model: Consider a dense feed-forward neural network f(x; n) with weights and biases $\theta = \theta_0$ and initial configuration of nodes/filters n. Let f reach minimum validation loss l with test accuracy a when optimizing with stochastic gradient descent (SGD) on a training set. Consider also training $f(x; m \odot n)$ with a mask $m = \{1\}^{|n|}$ on its nodes/filters such that its initialization is $f(x; m \odot n)$, where $m \odot n$ is an element-wise multiplication between m and n. Let f with the mask reach minimum validation loss l′ with test accuracy a′.

Algorithm:- Iterative Pruning Algorithm

Input: Neural network with initial state θ_0, initial nodes/filters n, pruning metric.

Hyperparameters: Training iterations j, pruning fraction $p \in (0, 1]$, maximum loss factor

Initialize starting mask

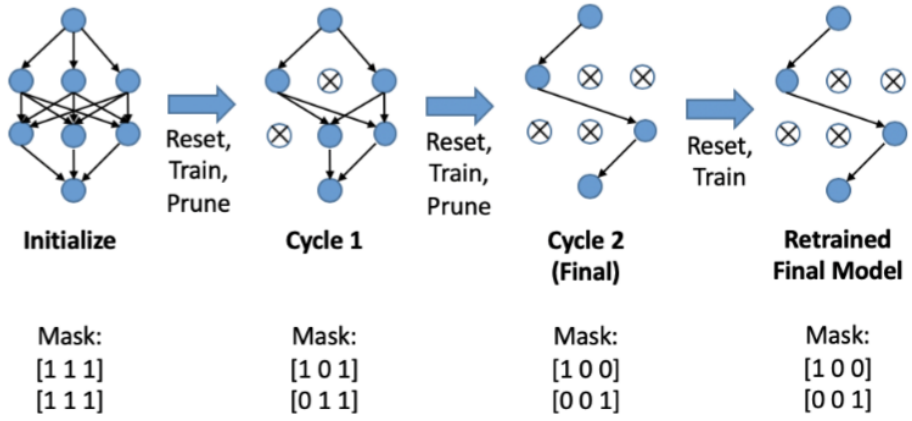
Repeat

1. Revert network to initial state θ_0
2. Apply mask to nodes/filters: $f(x; m \odot n)$
3. Train network for at most j iterations until early stopping
4. Apply pruning metric to choose a fraction p of nodes/filters to drop and update m

until validation accuracy $a' \leq \kappa a$

Run steps 1 to 3

DropNET example of Algorithm 1 -



The mask is initially set to all 1s, meaning all nodes/filters are present. As the training cycle progresses, more and more nodes/filters are dropped. The final model at cycle 2 is then reset to the initial state and retrained to give the final parameters.

Key Contributions -

The researchers proposed a method called DropNet, which drops nodes/filters, thereby reducing the complexity of the neural network of the model while keeping the accuracy the same. They have shown using their proposed model, and They reduced the network size up to 90% while keeping the accuracy the same. Also, there does not need to be a particular initialization required when dropping up to 70% of the nodes/filters.

DropNet, utilizing either the minimum or minimum layer metric, proves to be competitive over a variety of model architectures. The minimum metric seems to work robustly well for smaller models such as Models 1 (**FC40 - FC40**) and Model 2 (**Conv64 - Conv32**), while the minimum layer metric appears to be better for larger models such as Model 3 (**Conv64 - Conv64 - Conv128 - Conv128**), ResNet18 and VGG19, in the configurations we considered.

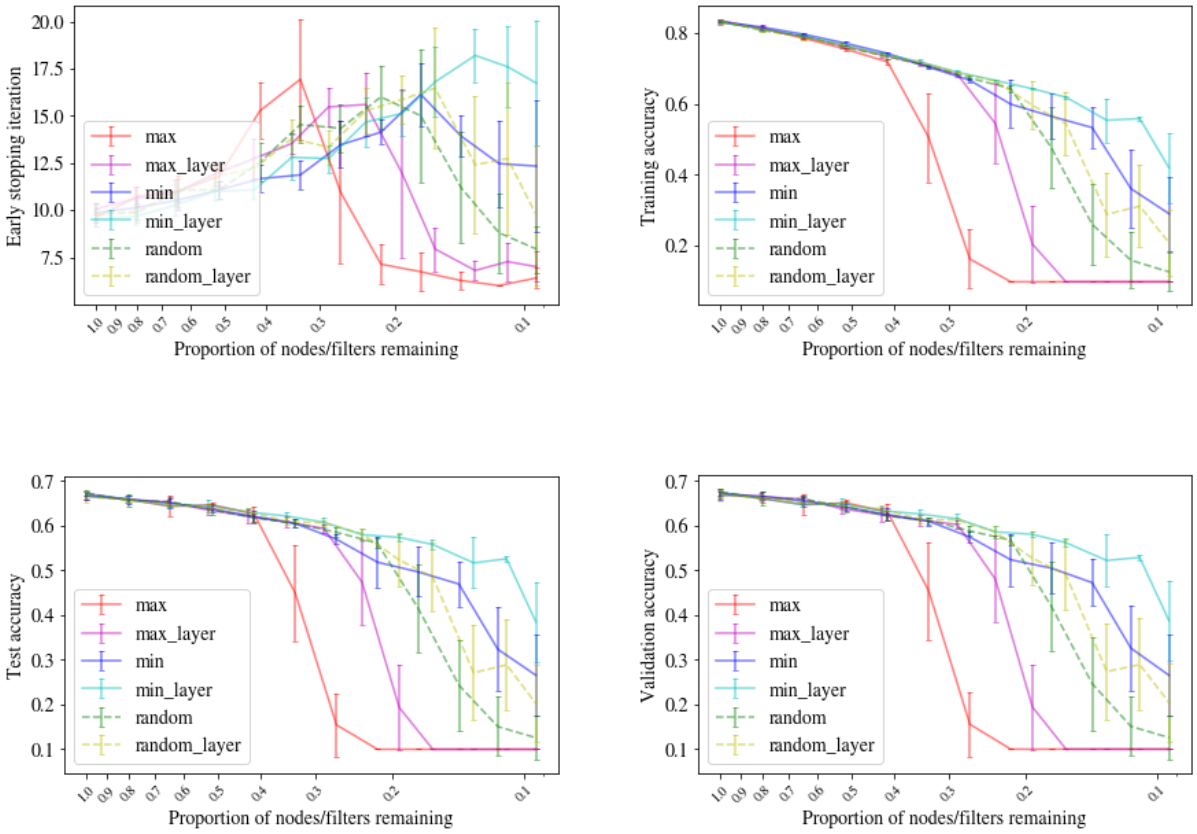
Also, the reason for the better performance of the minimum layer metric in larger models is that as the number of layers increases, the statistical properties of the post-activation values of the nodes/filters may be significantly different in each layer. Hence, using a single metric to prune globally may not be as good as pruning layer-wise. The exception is when using skip connections, as empirical results suggest that the minimum metric is also competitive for larger models for ResNet18. This seems to suggest that skip connections work well with DropNet.

Reproduced Results for MNIST Dataset -

I have used my personnel laptop's GPU to run the code provided by the respected paper authors. The results were more sort of comparison of various pruning metrics while using the DenseNet model. Pruning metrics evaluate the importance score for each node/filter and drop the nodes/filters with the lowest importance scores. These metrics are minimum, maximum, random, minimum layer, maximum layer, and random layer metrics.

So, let's see the graphs which were the result of this running,

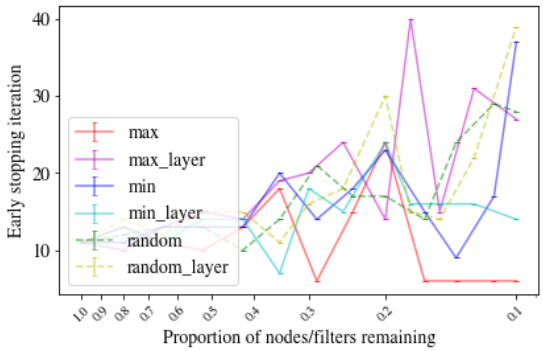
For MNIST data used in the Paper,

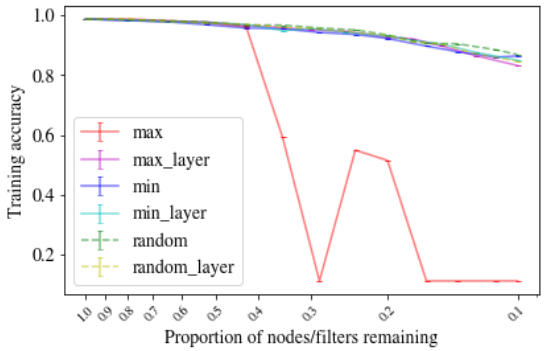


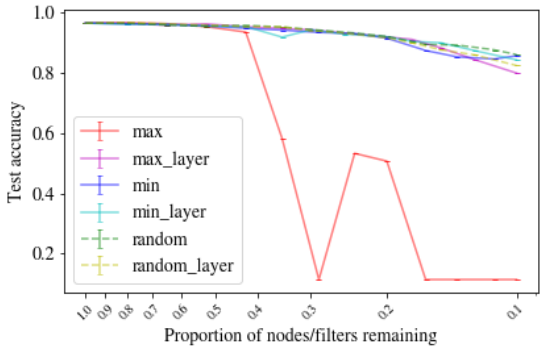
The above graphs are the relation between the nodes/filters with early stopping iteration, training data, testing data, and validation data.

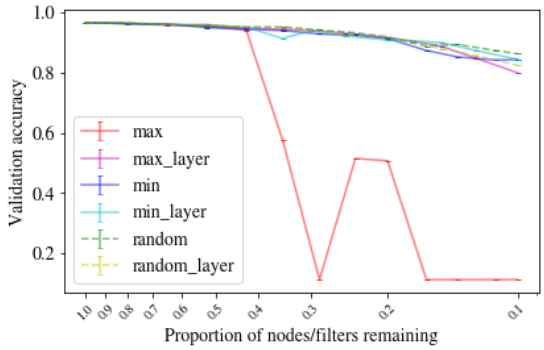
Results of Various models [Reproduced Results] -

▼ **Model 1 - [FC40 - FC40]**

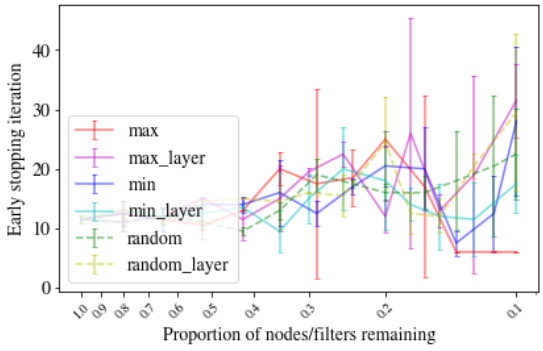


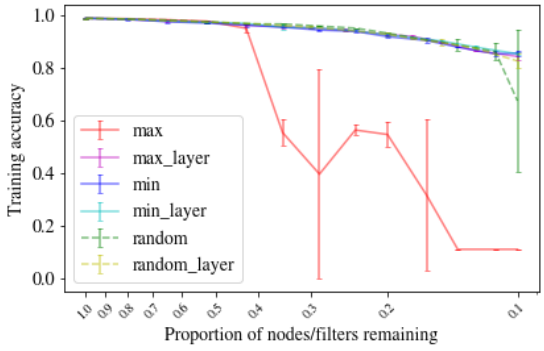


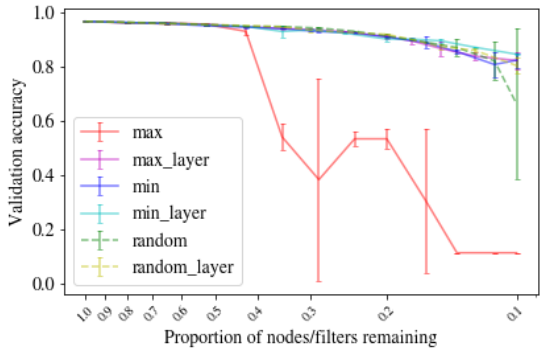


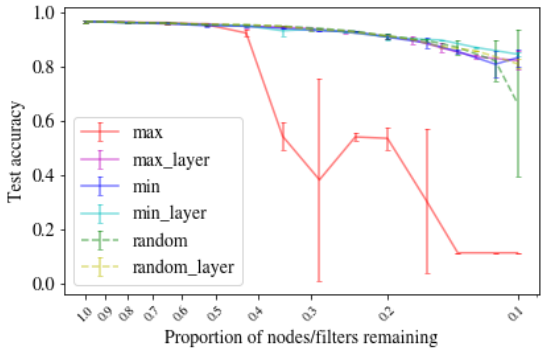


▼ Model 2 - [Conv64 - Conv32]

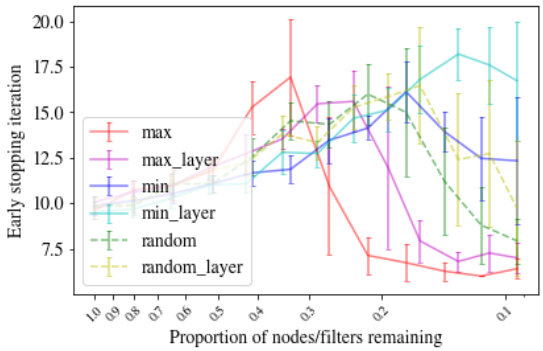


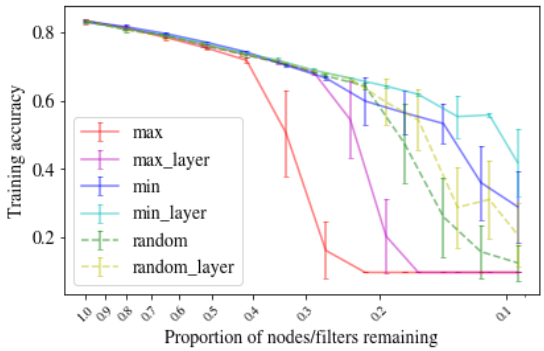


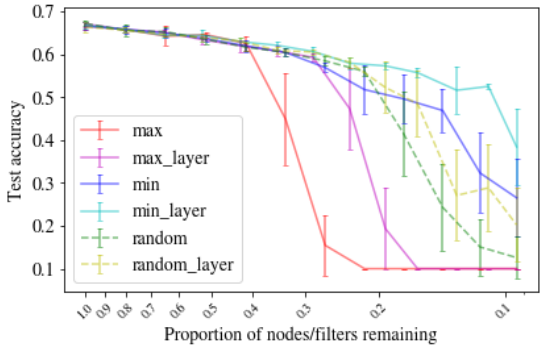


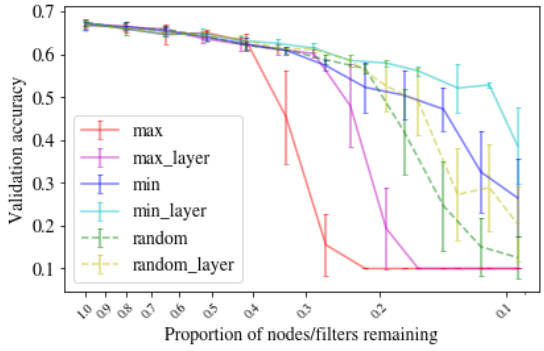


▼ Model 3 - [Conv64 - Conv64 - Conv128 - Conv128]









Overall, we can see that dropping a greater fraction p of nodes/filters per training cycle leads to poorer performance. For one-shot pruning methods which attempt to remove 90% of nodes/filters, it is not ideal as it generally leads to worse performance.

The results also show that larger p(Proportion of nodes/filters remaining) has similar performance when dropping up to 70% of the nodes/filters. Hence, it may be possible to iterate faster through Algorithm 1 by adopting a larger p under certain conditions. Further

experiments need to be done to determine the optimal pruning fraction p, but p = 0.2 seems to be competitive.

The results reproduced are, in conclusion, similar to the results which were given in the paper.

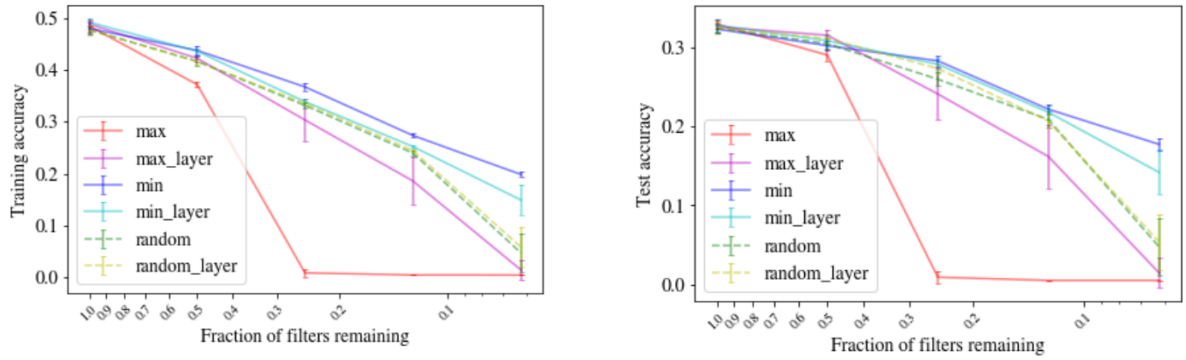
Results from using a different Database [Tiny Imagenet] -

Here lets see weather DropNet can give better results using the same algorithm on the larger dataset which is Tiny ImageNet.

We make use of predefined models ResNet18 and VGG19. We only modify the models which are slightly in order to for the 200 output classes of Tiny ImageNet, which is an increase from the 10 output classes in MNIST. As such, the last linear layer for ResNet18 has 200 nodes (instead of 10 nodes for MNIST), while the last two linear layers for VGG19 has 1024 nodes and 200 nodes respectively (instead of 256 nodes and 10 nodes respectively for MNIST).

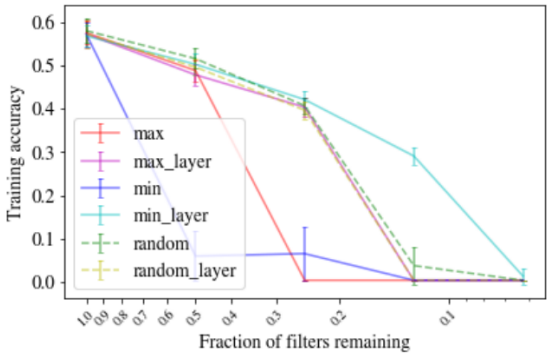
Due to the complexity of this dataset, in order to attain better train/test accuracies, we apply image augmentation to each data sample per epoch. Also, we decrease the pruning fraction 0.5 for due to complexity of running time.

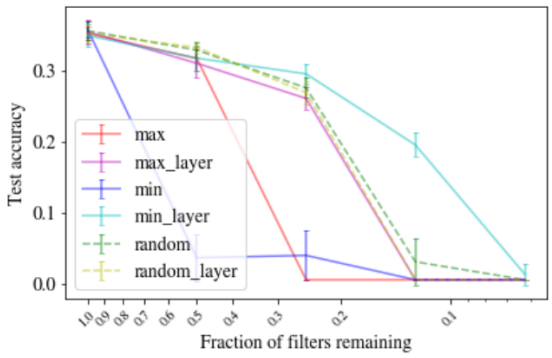
Results from ResNET18 -



For ResNet18, we can 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.

Results from VGG19 -





For VGG19, we can see that the minimum layer metric performs the best, followed by random, random layer, max layer, maximum, 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.

Also, the experiments on the Tiny ImageNet dataset also suggest that the minimum and minimum layer are both competitive in ResNet18.

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.