Pruning Neural Networks

Neural networks can be made smaller and faster by removing connections or nodes

Rohit Bandaru Sep 1, 2020 · 5 min read ★

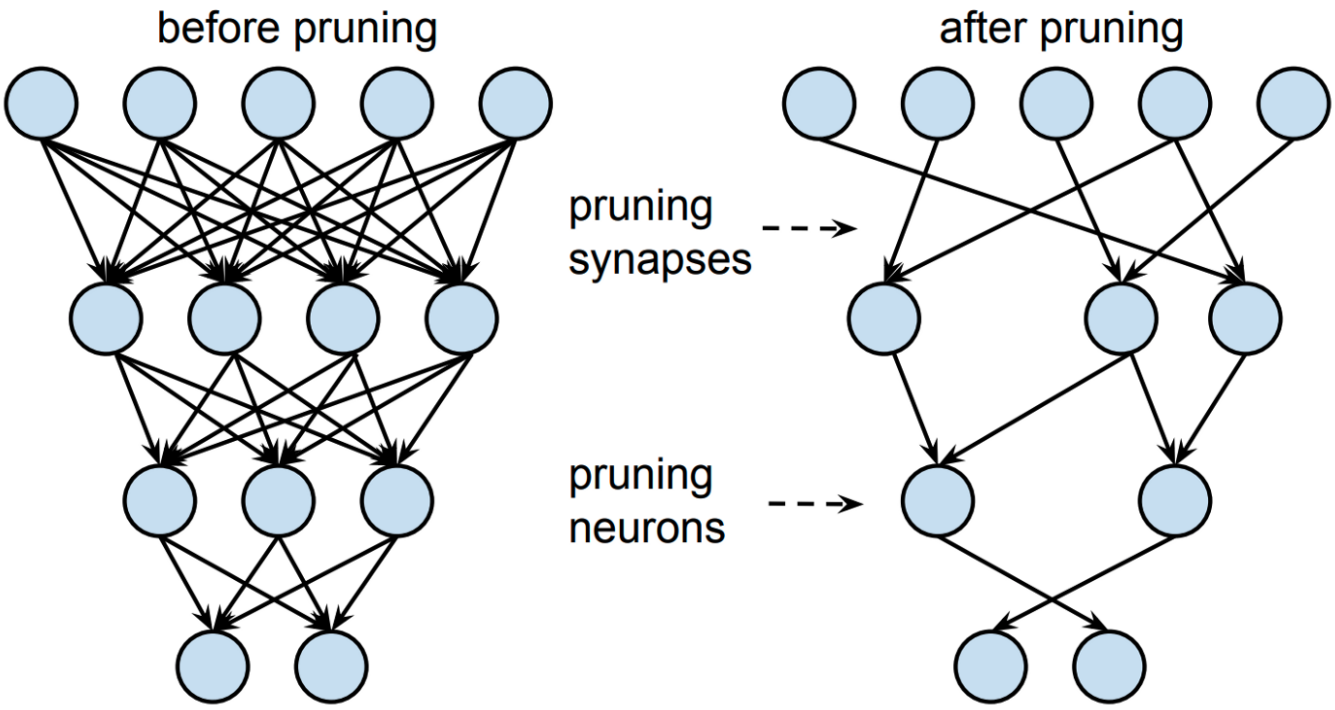
Much of the success of deep learning has come from building larger and larger neural networks. This allows these models to perform better on various tasks, but also makes them more expensive to use. Larger models take more storage space which makes them harder to distribute. Larger models also take more time to run and can require more expensive hardware. This is especially a concern if you are productionizing a model for a real-world application.

Model compression aims to reduce the size of models while minimizing loss in accuracy or performance. Neural network pruning is a method of compression that involves removing weights from a trained model. In agriculture, pruning is cutting off unnecessary branches or stems of a plant. In machine learning, pruning is removing unnecessary neurons or weights. We will go over some basic concepts and methods of neural network pruning.

Remove weights or neurons?

There are different ways to prune a neural network. (1) You can prune weights. This is done by setting individual parameters to zero and making the network sparse. This would lower the number of parameters in the model while keeping the architecture the

same. (2) You can remove entire nodes from the network. This would make the network architecture itself smaller, while aiming to keep the accuracy of the initial larger network.



Visualization of pruning weights/synapses vs nodes/neurons (Source)

Weight-based pruning is more popular as it is easier to do without hurting the performance of the network. However, it requires sparse computations to be effective. This requires hardware support and a certain amount of sparsity to be efficient.

Pruning nodes will allow dense computation which is more optimized. This allows the network to be run normally without sparse computation. This dense computation is more often better supported on hardware. However, removing entire neurons can more easily hurt the accuracy of the neural network.

What to prune?

A major challenge in pruning is determining what to prune. If you are removing weights or nodes from a model, you want the parameters you remove to be less useful. There are different heuristics and methods of determining which nodes are less important and can be removed with minimal effect on accuracy. You can use heuristics based on the weights or activations of a neuron to determine how important it is for the model's performance. The goal is to remove more of the less important parameters.

One of the simplest ways to prune is based on the magnitude of the weight. Removing a weight is essentially setting it to zero. You can minimize the effect on the network by removing weights that are already close to zero, meaning low in magnitude. This can be implemented by removing all weights below a certain threshold. To prune a neuron based on weight magnitude you can use the L2 norm of the neuron's weights.

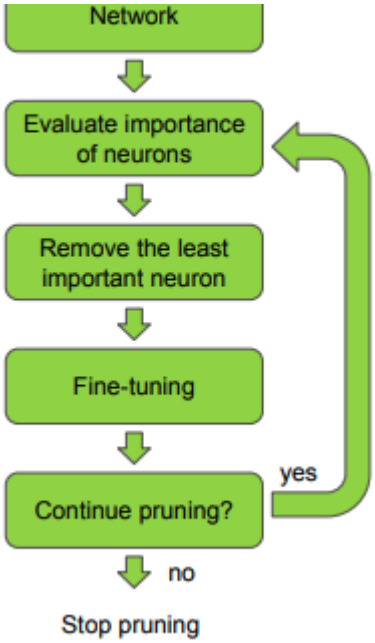
Rather than just weights, activations on training data can be used as a criteria for pruning. When running a dataset through a network, certain statistics of the activations can be observed. You may observe that some neurons always outputs near-zero values. Those neurons can likely be removed with little impact on the model. The intuition is that if a neuron rarely activates with a high value, then it is rarely used in the model's task.

In addition to the magnitude of weights or activations, redundancy of parameters can mean a neuron can be removed. If two neurons in a layer have very similar weights or activations, it can mean they are doing the same thing. By this intuition, we can remove one of the neurons and preserve the same functionality.

Ideally in a neural network, all the neurons have unique parameters and output activations that are significant in magnitude and not redundant. We want all the neurons are doing something unique, and remove those that are not.

When to prune?

A major consideration in pruning is where to put it in the training/testing machine learning timeline. If you are using a weight magnitude-based pruning approach, as described in the previous section, you would want to prune after training. However, after pruning, you may observe that the model performance has suffered. This can be fixed by fine-tuning, meaning retraining the model after pruning to restore accuracy.



Flow of iterative pruning (Source)

The usage of pruning can change depending on the application and methods used. Sometimes fine-tuning or multiple iterations of pruning are not necessary. This depends on how much of the network is pruned.

How to evaluate pruning?

There multiple metrics to consider when evaluating a pruning method: accuracy, size, and computation time. Accuracy is needed to determine how the model performs on its task. Model size is how much bytes of storage the model takes. To determine computation time, you can use FLOPs (Floating point operations) as a metric. This is more consistent to measure than inference time and it does not depend on what system the model runs on.

With pruning, there is a tradeoff between model performance and efficiency. You can prune heavily and have a smaller more efficient network, but also less accurate. Or you could prune lightly and have a highly performant network, that is also large and expensive to operate. This trade-off needs to be considered for different applications of the neural network.

Conclusion

Pruning is an effective method of making neural networks more efficient. There are plenty of choices and areas of research in this area. We want to continue to make

advances in deep learning while also keeping our models energy, time, and space-efficient.

References

[1] Blalock, Davis, et al. "What is the state of neural network pruning?." *arXiv preprint arXiv:2003.03033* (2020).

[2] Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.

[3] PyTorch Pruning Tutorial
https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

[4] Keras / Tensorflow Pruning Tutorial
https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras

[5] Molchanov, Pavlo, et al. "Pruning convolutional neural networks for resource efficient inference." *arXiv preprint arXiv:1611.06440* (2016).

[6] Babaeizadeh, Mohammad, Paris Smaragdis, and Roy H. Campbell. "Noiseout: A simple way to prune neural networks." *arXiv preprint arXiv:1611.06211* (2016).

Thanks to Elliot Gunn.

Machine Learning Deep Learning Pruning Compression Artificial Intelligence