Finding Reduced Parameter Count Network But Keeping Same Accuracy

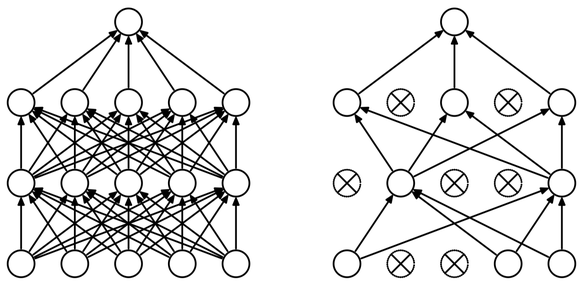
Hasan Unlu, hunlu@stanford.edu

Problem

Demonstrating to prune approximately %90-%80 of the weights of a neural network and to preserve same accuracy with faster training time. Two main pruning techniques are compared in this work. The novel part mentioned in the paper **[1]** is that after each pruning cycle, reset the remaining weights with original random initialization. The other method is randomly initializing the remaining weights after each prune cycle **[2]**.

Other Pruning Techniques

Neural network pruning techniques usually prune weights and neurons(which corresponds entire column to disappear in weight matrix). The mainly used criteria are norm of the weight, gradient checking for the particular weights or removing randomly selected neurons in a layer during the training. One of the known pruning technique is dropping out neurons with predefined probability. The Main idea is to prevent over-fitting. This method reduces size of the network up to %50 for some datasets and networks with same accuracy. However, number of iteration per epoch increases. The other method is removing low norm weights and reinitialize the remaining weights randomly.



(a) Standard Neural Net (b) After applying dropout.

The Dropout **[3]**

Neural Network and Dataset Selection

-Neural Networks

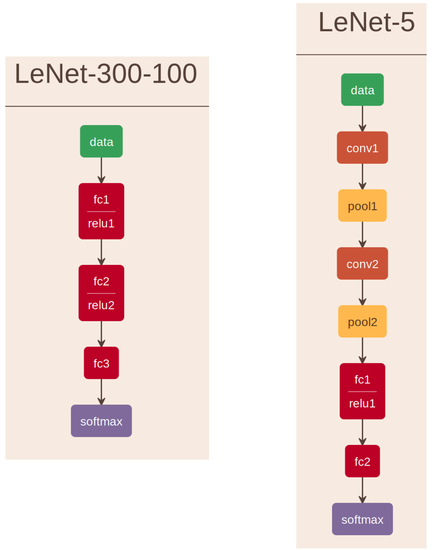
- LeNet-300-100 is for fully connected focused pruning
- LeNet-5 is for convolution network pruning

-Dataset

- MNIST is primary dataset used in these experiments

-Pruning Techniques

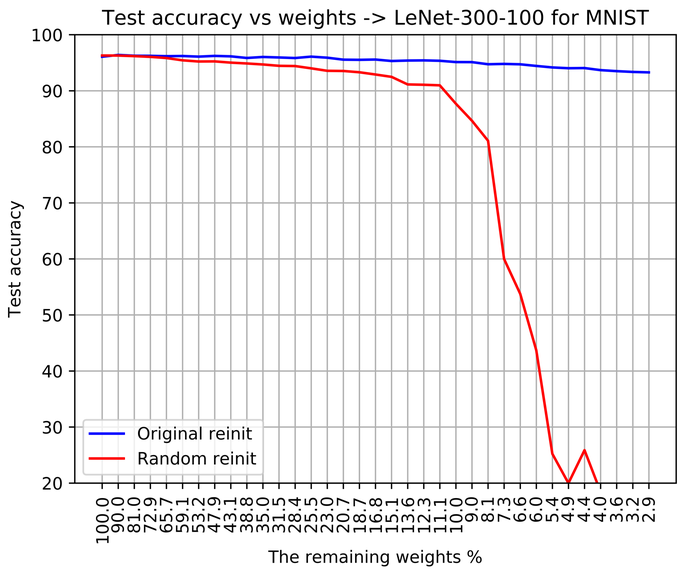
- Original random reinitialization every pruning cycle **[1]**
- New random reinitialization each pruning cycle **[2]**



Neural Network Architectures

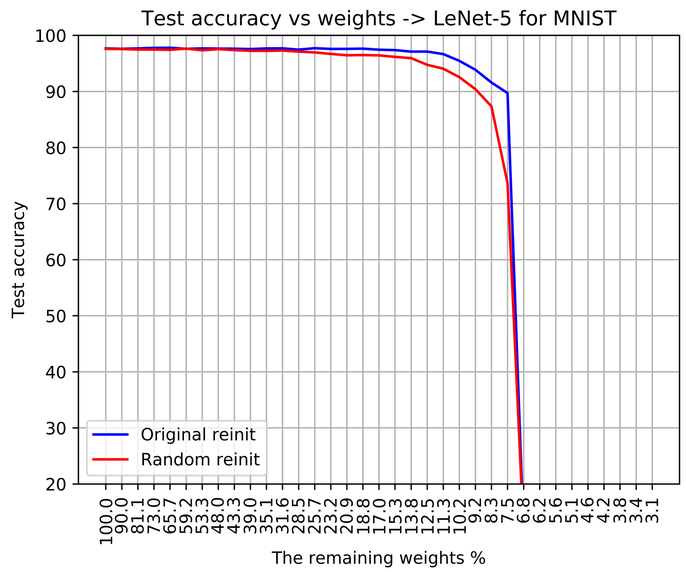
Implementation

Each prune iteration pre-determined amount of weights are removed from original network. %9 is selected initially. Because, we need to see accuracy change on gradual reduction of the weights. We could select lower pruning rates, but it takes longer time to reach final pruning rate. Each prune iteration, weights are sorted in their magnitudes. Then, the least **%p** of them are removed. Two different reinitialization methods are applied for the remaining weights. The first one is to load the original initialized weights **[1]**. The second one is to load newly generated random weights to them **[2]**.



Results

As we see that initializing with original random weight values after each pruning step performs better than random initialization for convolution and fully connected pruning. Original reinit performs way better than random reinit in fully connected pruning. They perform almost same in convolution pruning. For both types of network, there is at least one pruned network which performs better test accuracy than original network. Convolution network is sensitive to pruning. If the percentage was selected less on convolution part, the results would be similar to fully connected pruning.



Intuitive Explanation

For random initialization, we change neural network intrinsic in each pruning step and it may not be represent subset of the original network. Every random initialization step, we increase the probability of the divergence. If we initialize with the original initialization of the network, we fast-forward the prune steps. The last pruned step is always subset of the network, because it has the original initialization.

References

[1] Frankle, J. and Carbin, M. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. ICLR 2019.

[2] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural networks. In NIPS, 2015b.

[3] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. J. Machine Learning Res. 15, 1929–1958 (2014).