Pruning

Q&A Session

11/18

TA : 김성년, 안재연

Covering papers as follows:

- Learning both Weights and Connections for Efficient Neural Networks, 15`NIPS
- The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, 19`ICLR
- Pruning Filters for Efficient ConvNets, 17`ICLR
- The Generalization-Stability Tradeoff In Neural Network Pruning, 20`NIPS

Pruning can be categorized into two :

- Unstructured pruning :

  solely based on numerical value; does not consider its structure

- Structured pruning :

  whereas here we consider what/where we’re pruning (e.g. filters, channels)

  hence, we need to prune in group

1. Learning both Weights and Connections for Efficient Neural Networks, 15`NIPS

- Summary: This paper presents a conventional method of pruning which dragged out a lot of subsequent studies; their algorithm is as follows: (1) train the network to convergence, (2) cut out the connections based on its learned connectivity, and then (3) retrain the network to regain the stability. The last two steps can either be carried on for once or iteratively.

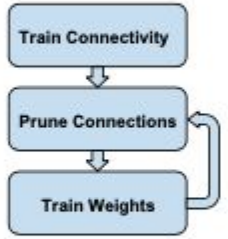


Figure 2: Three-Step Training Pipeline.

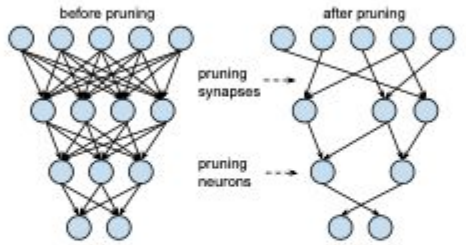


Figure 3: Synapses and neurons before and after pruning.

very simple, typical method; indeed, many paper stems from this algorithm

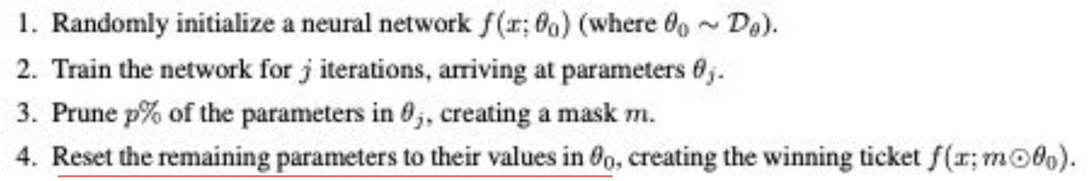
2. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, 19`ICLR

- Proposed Algorithm: The paper states that the standard pruning technique is in fact equivalent to revealing the inherent small subnetwork, which they dubbed as ‘lottery ticket’, and empirically proved that it performs at a similar level compared to the original network. In detail, they proposed an iterative pruning where the network is fully trained for one iteration, pruned based on the computed weights, reset the remaining parameters to its initial values, and then repeat the process over again.

$$\downarrow$$

in short, it introduces a new perspective of pruning !

Draft of the proposed algorithm:



1. Randomly initialize a neural network $f(x; \theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j.
3. Prune $p\%$ of the parameters in θ_j, creating a mask m.
4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

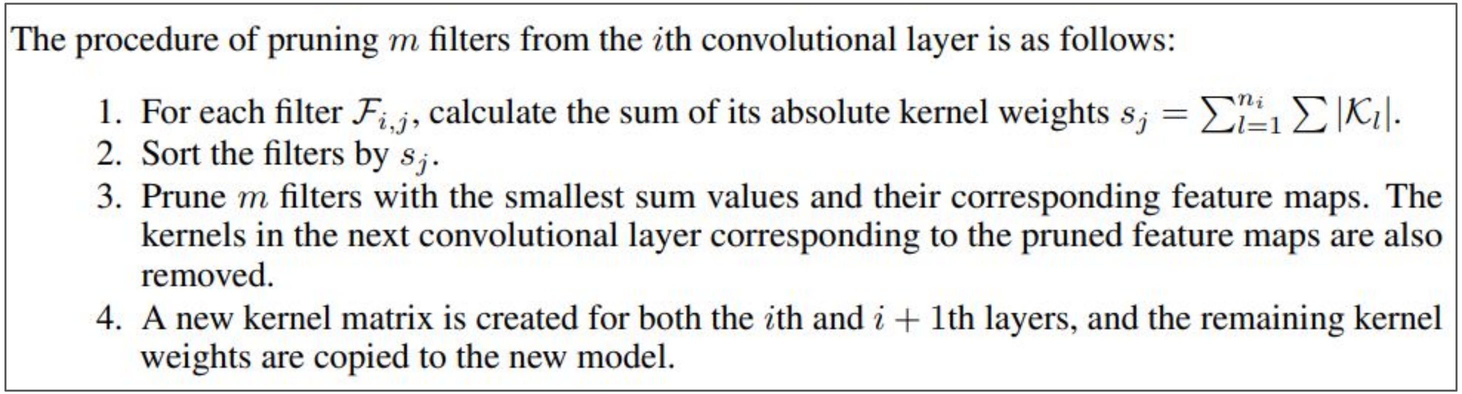
note that it reset to the initial value; further, emphasizes that reinitialization is not appropriate

this is one-shot;
repeating 2,3,4 would be
an iterative pruning
which turns out to be a better choice!

3. Pruning Filters for Efficient ConvNets, 17`ICLR

- Problem Statement: Weight-based pruning mostly reduces parameters from the fc-layers, while computation cost is highly dependent on the convolutional layers.
- Proposed Algorithm: They determine each filter's importance as the sum of absolute kernel weights, and remove those filters with small sum values. The next layer's connected weights to the corresponding filter are also eliminated. They also proposed ways of filter selection, pruning in a residual block, and retraining strategies.

→ **Reducing the computation cost of well-trained CNNs by pruning filters**



The procedure of pruning m filters from the ith convolutional layer is as follows:

1. For each filter $\mathcal{F}_{i,j}$, calculate the sum of its absolute kernel weights $s_j = \sum_{l=1}^{n_i} \sum |\mathcal{K}_l|$.
2. Sort the filters by s_j.
3. Prune m filters with the smallest sum values and their corresponding feature maps. The kernels in the next convolutional layer corresponding to the pruned feature maps are also removed.
4. A new kernel matrix is created for both the ith and $i + 1$th layers, and the remaining kernel weights are copied to the new model.

3. Pruning Filters for Efficient ConvNets, 17`ICLR

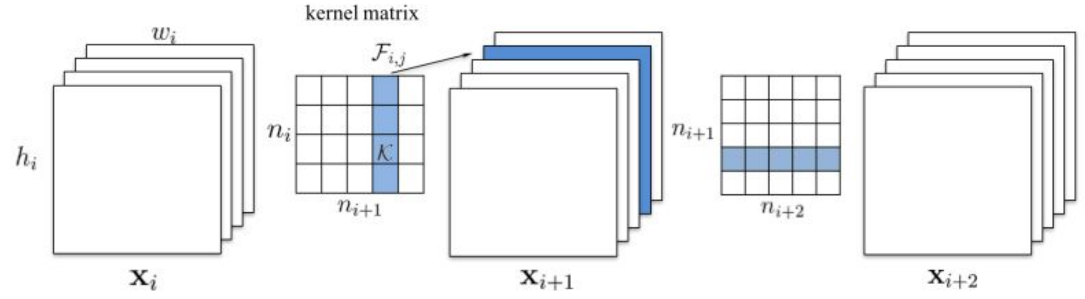
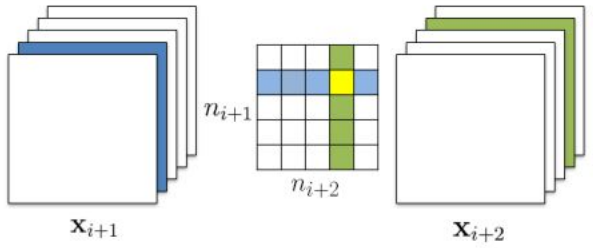
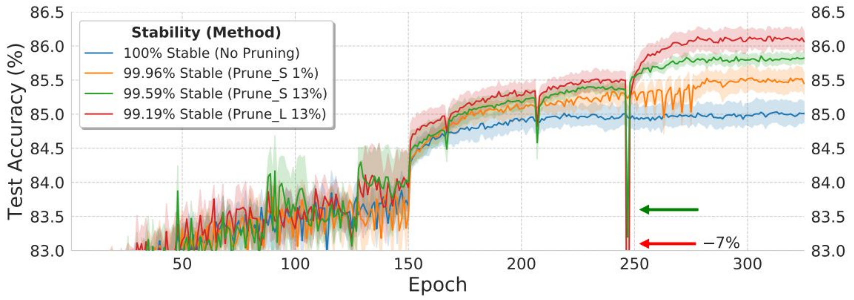


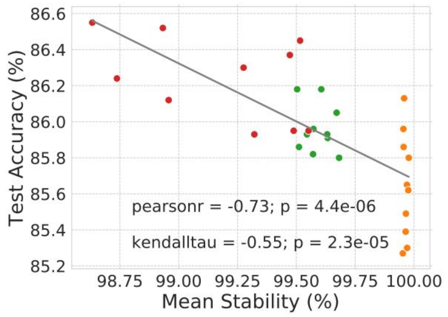
Figure 1: Pruning a filter results in removal of its corresponding feature map and related kernels in the next layer.

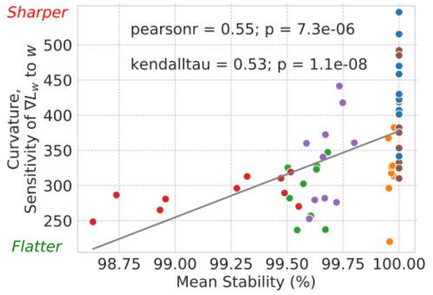


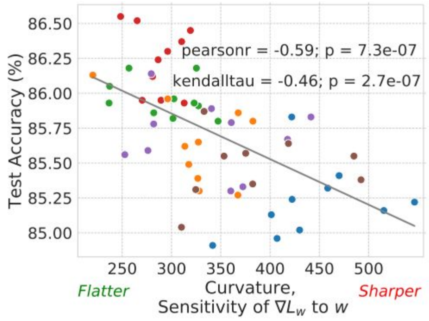
4. The Generalization-Stability Tradeoff In Neural Network Pruning, 20`NIPS

- Summary: Pruning stability is negatively correlated to the generalization level of the pruned model. Instability is measured by the amount of accuracy decrease after pruning. They assume and empirically show that pruning, viewed as noise injection, helps generalization by encouraging more flatness in DNN.


Stability (Method)
100% Stable (No Pruning)
99.96% Stable (Prune_S 1%)
99.59% Stable (Prune_S 13%)
99.19% Stable (Prune_L 13%)
Test Accuracy (%)
Epoch


pearsonr = -0.73; p = 4.4e-06
kendalltau = -0.55; p = 2.3e-05
Test Accuracy (%)
Mean Stability (%)


Sharper
Flatter
pearsonr = 0.55; p = 7.3e-06
kendalltau = 0.53; p = 1.1e-08
Curvature, Sensitivity of ∇L_w to w
Mean Stability (%)


pearsonr = -0.59; p = 7.3e-07
kendalltau = -0.46; p = 2.7e-07
Test Accuracy (%)
Flatter
Sharper
Curvature, Sensitivity of ∇L_w to w