PRUNING CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE EFFICIENT INFERENCE

Gist:

1. Oracle/Brute-force pruning (Remove a feature map/channel, get the change in the training losses) is the best, but slow.
2. Absolute oracle criterion is does not reduce accuracy as much as signed oracle criterion. More stable as the loss shouldn’t decrease OR increase too much by pruning.
3. Use Spearman coefficient to decide a good substitute to oracle criterion.
4. Lecun’s OBD and Taylor expansion performs the best. But OBD requires Hessian (second order derivative) which is expensive to compute.
5. In Taylor expansion, only use the first order expansion. Remainder requires Lagrangian hence it is omitted due to high computation. Substitute first order expansion into absolute loss results in absolute product of loss gradient w.r.t activation and activation itself.
6. Average Taylor criterion over pixels/neurons and also minibatch examples.
7. Scale of activation values varies according to network depth, normalize layer using L2 norm.
8. Use FLOPs regularization to reduce compute time. Prune layers with high FLOPs first.
9. The more training updates are done between pruning iterations, the higher the accuracy.
10. Additional fine-tuning at the end of pruning significantly restores lost accuracy.
11. Generally, shallower layers are more important than deeper layers.

Baseline: U-Net on Carvana Dataset

Training size: 4071 (images are split into left & right halves)

Validation size: 1018

Batch size: 6

Image scale: 0.5

GPU Memory: 8GB

Training Loss (BCE): 0.02259

Validation Metric (Dice Coeff): 0.9853

Ideas:

Since model only has Conv-BN-ReLU modules, only remove previous convs’ filters & biases, corresponding BN parameters, and next conv’s channels (but not the bias).

Cannot remove maxpool or bilinear upsampling since these are needed for output shape consistency and they are parameter-less.

Also, cannot remove final conv layer since it is used to generate mask and it only has 64 parameters.

Prune class:

-Initialise with network to be pruned.

- Need containers for Taylor ranks, activations, gradients.

- Might also want to store layer names and corresponding BN module

- Register forward and backward hooks on construction.

- Hooks add tensors to containers. Tensors must be detached from computation graph, else they would not be deallocated upon exit.

- After each mini-batch, compute Taylor ranks by averaging (activation \* gradient) over the spatial dimensions. Absolute the average and then average across examples in the minibatch. Accumulate into ranks container.

- After sufficient number of mini-batch iterations, rank the channels by first dividing the accumulated ranks by the number of iterations. Perform layer-wise L2-normalization and store normalized ranks in a hierarchy-less container. Also keep track of the layer index of each channel and the channel index wrt to the layer of each channel.

-Get the top-K channels to be removed next.