Data-free Parameter Pruning for Deep Neural Networks

Suraj Srinivas
surajsrinivas@ssl.serc.iisc.in
R. Venkatesh Babu
venky@serc.iisc.in

Video Analytics Lab,
Supercomputer Education and Research Centre,
Indian Institute of Science

Deep Neural nets (NNs) with millions of parameters are at the heart of many state-of-the-art computer vision systems today. However, recent works have shown that much smaller models can achieve similar levels of performance. In this work, we address the problem of pruning parameters in a trained NN model. Instead of removing individual weights one at a time as done in previous works, we remove one neuron at a time. We show how similar neurons are redundant, and propose a systematic way to remove them. Unlike previous works, our pruning method does not require access to any training/validation data.

Wiring similar neurons The main principle that we use in this paper is the fact that similar neurons are redundant, as shown in Figure 1. That is, if we find such a similar weight pair anywhere in a neural network, one of them can effectively be removed. Of course, while doing this we also need to account for the weights in the next layer, as shown in Figure 1. This observation also resonates with the well-known Hebbian principle, which roughly states that neurons that fire together ($W_1 = W_2$), wire together ($a_1 = a_1 + a_2$).

Wiring dis-similar neurons The above principle cannot be used as is in real NNs, for one simple reason - weight-sets are seldom equal in value. What do we do when $\|W_1 - W_2\| = \|\varepsilon_{1,2}\| \geq 0$? Let z_n be the output neuron when there are n hidden neurons. Let us consider two similar weight sets W_i and W_j in z_n and that we have chosen to remove W_j to give us z_{n-1}. Using some approximate analysis, we derive a simple rule to find which weight-sets to remove. The final equation is

$$min(E(z_n - z_{n-1})^2) \leq min_{i,j}(a_j^2 \, \|\varepsilon_{i,j}\|_2^2) \, E\|X\|_2^2 \qquad (1)$$

We aim to minimize the expected value of the squared difference between the output neurons. Using the expected error instead of the empirical error is what makes it a **data-free** parameter pruning method. We define the saliency of two weight-sets in (i, j) as $s_{i,j} = \langle a_j^2 \rangle \, \|\varepsilon_{i,j}\|_2^2$, which is exactly the term inside the $min(\cdot)$ in Equation 1. Intuitively, saliency between two weight-sets is low when they have very similar values. Equation 1 tells us that we need to start removing lowest-saliency neuron to minimize the expected squared difference.

We elucidate our procedure for neuron removal here:

1. Compute the saliency $s_{i,j}$ for all possible values of (i, j). It can be stored as a square matrix M, with dimension equal to the number of neurons in the layer being considered.

2. Pick the minimum entry in the matrix. Let it's indicies be (i', j'). Delete the j'^{th} neuron, and update $a_{i'} \leftarrow a_{i'} + a_{j'}$.

3. Update M by removing the j'^{th} column and row, and updating the i'^{th} column (to account for the updated $a_{i'}$.)

Connections to other methods Our method relates to the popular weight-pruning method called Optimal Brain Damage (OBD) [3]. In fact, our method is equivalent to OBD if change in output activation produces proportional change in test error. Unfortunately, this is almost never the case for neural networks. Our method also weakly relates to Knowledge Distillation (KD) [1]. The idea in KD was to minimize the empirical difference in output neurons between a large "teacher" network and a smaller "student" network. In our case the small student network is the pruned version of the larger teacher network.

Determining number of neurons to prune One way to use our technique would be to keep removing neurons until the test accuracy starts going below certain levels. However, this is quite laborious to do for large networks with multiple layers. We ask whether it is possible to somehow determine the number of removals automatically. Specifically, we ask whether the saliency values of pruned neurons can be used as a proxy for test error. Figure 2 shows that it is indeed true. Unfortunately, we found that this does not hold for large networks.

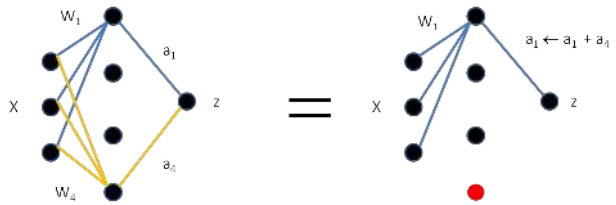

Figure 1: A toy example showing the effect of equal weight-sets ($W_1 = W_4$). Weights of the same colour in the input layer constitute a weight-set.

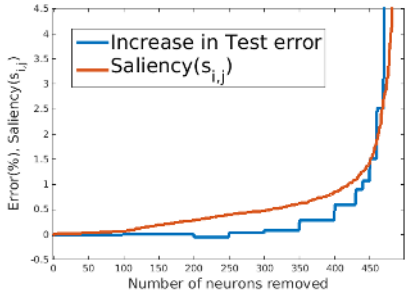

Figure 2: Scaled appropriately, the saliency curve closely follows that of increase in test error

Experiments

Toy dataset We show results on a toy dataset, comparing our method to Optimal Brain Damage and Optimal Brain Surgery. We see that our method performs better than the other two approaches, and is about 5000× faster than OBD, and 180000× faster than Optimal Brain Surgery.

MNIST-trained network We show results on an LeNet-like network trained on MNIST dataset. We were able to remove about 85% of the weights in the network, reducing the accuracy by only 1%.

AlexNet We try our method on AlexNet [2], which is a large-scale network trained on Imagenet database. We were able to remove about 35% of the total weights in the network, reducing the accuracy by 2.2%. In other words, we were able to remove about 21 million weights in a network of 60 million weights, with little effect on accuracy.

References

[1] Geoffrey E Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[3] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, 1989.