Gradient and Magnitude based pruning for Sparse Deep Neural Networks

Kaleab B. Belay

What is pruning?

- Pruning a Neural Network is the process of removing edges (weights) and/or nodes (neurons) from the network.
- It is usually done to reduce the number of parameters of the Neural Network so that it consumes less memory or does not overfit to training samples.
- Pruning is done in an iterative fashion to avoid the deletion of essential neurons and connections. As the important nodes and edges are identified gradually, iterative pruning ensures that the important elements of the network are not prematurely removed.

The need for pruning

- Deep Neural Networks (DNNs) have immense computational and memory requirements that make them difficult to use in low-resource environments. For instance, training VGG-16 with a batch size of 128 will require about 14 GB of global GPU memory while the latest NVIDIA Titan X Pascal GPU has a total of 12 GB DRAM (Rhu et al., 2016).
- Highly dense networks are overparameterized and prone to overfitting. Traditional techniques for preventing overfitting such as dropout do not improve computational and memory usage efficiency.
- Removing unimportant weights from a Neural Network is also expected to speed up learning and classification, and requires fewer training samples.

Literature Review

- There are numerous effective DNN pruning schemes that have gained wide acceptance in the Machine Learning community. In their paper *Optimal Brain Damage*, Lecun et al. (1989) propose a pruning mechanism whereby the second derivative of the objective function with respect to each parameter is computed to calculate its saliency. Low-saliency parameters are removed after this step.
- In a closely related work, Han et al. (2015) propose a magnitude-based pruning scheme in which all weights whose values lie below a specific threshold are removed and re-adjusts the remaining weights in the next round of training.
- Zhu and Gupta (2017) also use magnitude-based pruning to move the network from an initial sparsity level s(i) to a final sparsity level s(f) in n pruning steps.

Drawback of these approaches

- The major drawback inherent to all the pruning methods mentioned above is that they require the entire model to be trained initially to be able to distinguish the connections that matter from those that do not, resulting in more than 100% performance overhead. In other words, all of the above methods need pre-existing knowledge about which connections matter and which do not to function properly.

Gradient and magnitude based pruning

- Our proposed method avoids having to retrain the entire model by learning the importance of each connection while pruning is ongoing. Instead of relying solely on the magnitude of the weights to determine whether they should be pruned or not, we observe the values of their gradients to determine the rate of change of their magnitudes. Weights whose gradients fall below a specific threshold are assumed to have settled relatively close to their final values, thus they can be pruned based on their magnitudes.
- Pruning is carried out iteratively according to a schedule that is one of the model hyperparameters.

The Algorithm

Algorithm 1 Gradient and magnitude based weight pruning

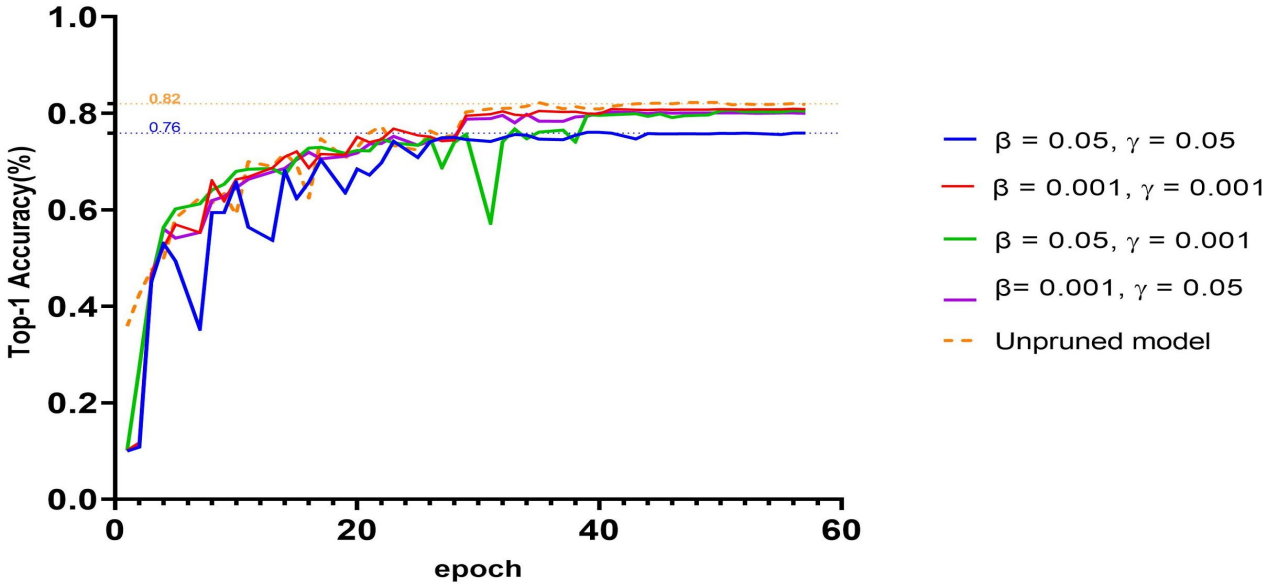
Require: $\nabla_W C$, tensors of weight gradients taken against validation dataset
$\boldsymbol{W}$, weight tensors of the network
$\boldsymbol{B}$, bitmask tensors of the network
β, weight threshold
γ, gradient threshold
δ, pruning schedule
$epoch$, current epoch
for each: $w \in \boldsymbol{W}, g \in \nabla_W C, b \in \boldsymbol{B}$ **do**
    if $|w| < \beta$ **and** $|g| < \gamma$ **and** $b \neq 0$ **then**
        $b \leftarrow 0$
        $w \leftarrow 0$
Ensure: $epoch \% \delta = 0$

Performance

Table 1: Model size and accuracy tradeoff for sparse-MobileNet ($\alpha = 1$, depth=4) with $\delta = 6$

β	γ	**sparsity**	**Non-zero params**	**Top-1 Accuracy(%)**
0.001	0.001	7.5%	12,913,302	81%
0.001	0.05	7.8%	12,879,444	80.1%
0.05	0.001	89.1%	1,520,242	78.2%
0.05	0.05	88.4%	1,607,959	76%

Performance



Key Insights

- In our experiments, we varied the pruning parameters β and γ as well as the pruning schedule δ to determine how their interaction affects model size and accuracy. We saw that moderately aggressive values assigned to the pruning parameters yield significant reduction in model size while incurring minimal drop in accuracy.
- Pruning is executed according to a specified schedule δ. So long as the model received sufficient number of training steps to recover from the changes introduced during pruning, varying the δ didn't yield any notable changes in accuracy or compression rate.
- On the CIFAR-10 dataset, our method reduced the number of parameters of MobileNet by a factor of 9X, from 14 million to 1.5 million, with just a 3.8% drop in accuracy.

References

Lecun, Yann & Denker, John & Solla, Sara. (1989). Optimal Brain Damage. Advances in Neural Information Processing Systems. 2. 598-605.

Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen Keckler. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design. pp.1–13, 10 2016. doi: 10.1109/MICRO.2016.7783721

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. ArXiv, abs/1710.01878, 2017