Gradient and Magnitude Based Pruning
for Sparse Deep Neural Networks

Kaleab Belay
Addis Ababa Institute of Technology
belete.kaleab@gmail.com

Abstract

Deep Neural Networks have memory and computational demands that often render them difficult to use in low-resource environments. Also, highly dense networks are over-parameterized and thus prone to overfitting. To address these problems, we introduce a novel algorithm that prunes (sparsifies) weights from the network by taking into account their magnitudes and gradients taken against a validation dataset. Unlike existing pruning methods, our method does not require the network model to be retrained once initial training is completed. 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.

Related Work

Various pruning methods have been proposed to reduce model size while incurring minimal loss in accuracy. In a closely related work, Han et al. (2015) propose a magnitude-based pruning approach in which the algorithm removes all weights below a specific threshold and re-adjusts the remaining weights in the next round of training. The drawback of this method is that it requires the model to be trained first so that the connections that matter can be distinguished from those that do not, resulting in more than 100% performance overhead. The method suggested by Zhu and Gupta (2017) also suffers from similar performance issues.

Method

We present the pseudocode for our proposed pruning scheme, which sparsifies a network given a set of model hyperparameters. For every weight in the model’s weight tensors and its corresponding elements in the gradient and bitmask tensors, if the absolute value of the weight is below the weight threshold $\beta$ and its gradient is less than the gradient threshold $\gamma$, we set its weight and its corresponding bitmask to zero. Using NumPy’s array vectorization operations, this algorithm takes less than a second to implement on a network with 13.8 million parameters.
Results

We define our own MobileNet in Keras based on the instructions provided by Chen & Su (2017). We compare the accuracy and compression rate achieved by pruning networks trained with the scaling factor $\alpha$ of 0.5 and 1, and width multipliers 1, 2 and 4 on the CIFAR-10 dataset. We vary the pruning parameters $\beta$ and $\gamma$ as well as the pruning schedule $\delta$ to determine how their interactions affect model size and accuracy. We see that moderately aggressive values assigned to the pruning parameters yield significant reduction in model size while incurring minimal drop in accuracy.

In the experimental results presented in the paper, pruning is executed according to a specified schedule $\delta$. So long as the model received sufficient number of training steps to recover from the changes introduced during pruning, varying the $\delta$ didn’t yield any sufficient changes in accuracy or compression rate. Figure 1a shows the accuracy of a MobileNet model pruned with different values of $\beta$ and $\gamma$. Most pruning parameters yielded similar results, with the model pruned with $\beta = 0.05$ and $\gamma = 0.001$ had the best accuracy-compression tradeoff, achieving 89.1% sparsity and a Top-1 accuracy of 78.2%. The model had a higher accuracy when compared to the approach by Chen et al., who were able to achieve an accuracy of 73.8% for similar levels of sparsity.

Conclusion

This work introduces an iterative pruning scheme that reduces model size during training. We demonstrate the efficacy of this scheme and its compression-accuracy tradeoff by using different pruning parameters. We believe these results will encourage the use of model pruning in low resource environments.

References

Han, S.; Pool, J.; Tran, J. and Dally W. J. 2015. Learning both weights and connections for efficient neural networks. In NeurIPS.

Algorithm 1: Gradient and Magnitude Based Pruning
Parameters: $\nabla w_C$, tensor of weight gradients
$W$, weight tensor of the network
$B$, bitmask tensor of the network
$\beta$, weight threshold
$\gamma$, gradient threshold
$\delta$, pruning schedule
$\text{epoch}$, current epoch
Output: Sparse Neural Network
1: for each: $w \in W, g \in \nabla w_C, b \in B$ do
2: if $|w| < \beta$ and $|g| < \gamma$ and $b \neq 0$ then
3: $b \leftarrow 0$
4: $w \leftarrow 0$
Ensure: $\text{epoch} \mod \delta = 0$

Figure 1: (a) Top-1 accuracy for different values of $\beta$ and $\gamma$ on MobileNet (b) Total weights pruned and sparsity for $\beta = 0.05$, $\gamma = 0.05$

The bitmask tensors were introduced so as to prevent weights that have already been set to zero from being updated in future backpropagation operations. This is done by performing the Hadamard product of the training gradients with the bitmasks.