GRADIENT AND MAGNITUDE-BASED PRUNING FOR SPARSE DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

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 paramters of MobileNet by a factor of 9X, from 14 million to 1.5 million, with just a 3.8% drop in accuracy.

1 INTRODUCTION

In recent years, Deep Neural Networks (DNNs) have become ubiquitous in applications ranging from computer vision to speech recognition and natural language processing (Han et al., 2015). The advent of massively parallel GPUs, TPUs and other hardware customized for the use of DNNs has opened the way for the creation of deeper and denser network architectures. State-of-the-art pretrained models have immense computational and memory demands that cannot be satisfied in low-resource environments. For instance, training VGG-16 with a batch size of 128 will require about 14 GB of global GPU memory (Rhu et al., 2016). Meanwhile, the latest NVIDIA Titan X Pascal GPU has a total of 12 GB DRAM.

Various methods have been proposed to ameliorate these problems. In a closely related work, Han et al. (2015) propose a magnitude-based pruning 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 entirety of the original 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 suffer from similar performance issues.

We present a method that avoids having to retrain the model by learning the importance of a specific weight in the network through its gradient against a validation dataset. Weights which have gradients below a specified threshold are assumed to have settled close to their resting values and can be pruned based on their magnitudes. Pruning is carried out iteratively according to a schedule that is one of the model hyperparameters.

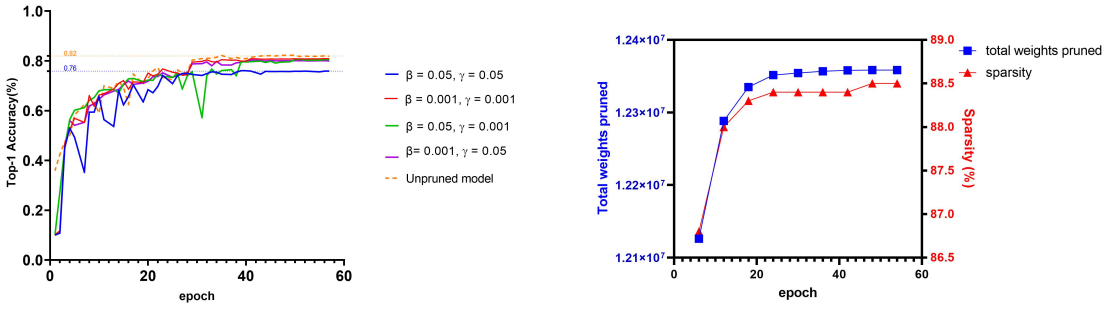
2 THE PROPOSED APPROACH

We present our 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 β and its gradient is less than the gradient threshold γ, we set it's weight and it's corresponding bitmask to zero. Using NumPy's array vectorization operations (van der Walt et al., 2011), this algorithm takes less than a second to implement on a network with 13.8 million parameters.

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, $\boldsymbol{B} \circ \bigtriangledown_W C$.

Algorithm 1 Gradient and magnitude based weight pruning

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



(a) (b)

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

3 Empirical Evidence

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 α of 0.5 and 1, and width multipliers 1, 2 and 4 on the CIFAR-10 dataset. Image augmentation was not used during training. We vary the pruning parameters β and γ as well as the pruning schedule δ to determine how their interaction affects 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 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 sufficient changes in accuracy or compression rate.

Figure 1a shows the accuracy of a MobileNet model pruned with different values of β and γ. All convolutional layers are pruned using the same pruning criteria. Most pruning parameters yielded similar results, with the model pruned with β = 0.05 and γ = 0.05 performing worse than the others. Table 1 offers more insight into the performance of the various pruning parameters. The model pruned with β = 0.05 and γ = 0.001 had the best accuracy-compression tradeoff, achieving 89.1% sparsity and a Top-1 accuracy of 78.2%.

4 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

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%

pruning parameters. We believe these results will encourage the use of model pruning in low-resource environments.

REFERENCES

Hong-Yen Chen and Chung-Yen Su. An enhanced hybrid mobilenet. *2018 9th International Conference on Awareness Science and Technology (iCAST)*, pp. 308–312, 2017.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

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.

Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13:22–30, 2011.

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