Gradient Based Pruning

Yang Yang (yyn@google.com), Rina Panigrahy (rinap@google.com)

Thanks to Suyog Gupta and Badih Ghazi for crucial discussions.

Motivation

Neural networks have shown great potential in tasks such as image classification, speech recognition, et al. Along with accuracy improvement, model size also grows significantly. This poses challenges for deploying these models on resource constrained hardware. Pruning is one of the main methods that reduce the memory footprint and computation requirements for neural networks. Studies such as [1] have shown that removing the connections with **small magnitude of weights** can be a very effective approach. Here we explore that if we take gradients (first order and second order) into account, can we get more signals on the salient of each connection in neural networks, and therefore helping us further compress models.

Intuition

First Order Gradient Pruning

Loss function L is a function of weights W in the model. So based on Taylor expansion and **first order approximation**, we have:

$$L(W + \Delta W) = L(W) + L'(W) * \Delta W = L(W) + g^T * \Delta W \text{ ... (1)}$$

Where:

$$g^T = \left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n}\right] \text{ ... (2)}$$

$$\Delta W = [\Delta w_0, \Delta w_1, \Delta w_2, \ldots, \Delta w_n]^T \text{ ... (3)}$$

Replacing (2) and (3) into (1), we have:

$$\Delta L = L(W + \Delta W) - L(W) = \sum_i \left(\frac{\partial L}{\partial w_i}\right) * \Delta w_i \text{ ... (4)}$$

We can see in (4) that ΔL is not only related to ΔW, which [1] uses to prune away connections, but also related to its first order derivatives. Therefore, we argue that, instead of using $|\Delta W|$ as the only salient connection indicator, what if pruning based on $|\Delta W| * |g|$.

Second Order Gradient Pruning

The core idea is based on [2]. In order to avoid calculating Hessian matrix for the second-order gradient per step, we used an approximation below for second order gradients, which is much easier to calculate.

$$\frac{\partial^2 L}{\partial {w_i}^2} = w'_i + w''_i - 2 * w'''_i \text{, where}$$

w'_i is weight snapshot at current step;

w''_i is weight snapshot at current step - 2;

w'''_i is weight snapshot at current step - 1.

Implementation and Usage

We extend the model pruning library in Tensorflow (link) so that the API changes required to adopt the gradient based pruning methods is minimal. User still needs to add pruning op in the training graph exactly the same way as the original model pruning library.

In addition to that, two new pruning hyperparameters are added.

`prune_option` hyperparameter is used to specify which method is used for pruning. Valid options are `'weight', 'first_order_gradient', 'second_order_gradient'`.

`gradient_decay_rate` is the other hyperparameter that is used to control the decay strength when calculating moving average for gradients.

Currently block sparsity is not supported for gradient based pruning.

Results

Cifar10

On Cifar10, we have observed consistent benefits (>1% test accuracy improvement) for the two gradient based pruning approaches when the sparsity level is high (> 95%). When the target sparsity level is low, we don't see much benefit with this approach.

Sparsity	*Prune Option*	AVERAGE of Test Accuracy (5 runs)
90%	weight	85.16%
90%	first_order_gradient	85.21%

90%	second_order_gradient	85.20%
98%	weight	78.93%
98%	first_order_gradient	79.65%
98%	second_order_gradient	79.49%
99%	weight	71.92%
99%	first_order_gradient	72.80%
99%	second_order_gradient	73.20%

References

[1] To prune, or not to prune: exploring the efficacy of pruning for model compression
[2] Optimal Brain Damage
[3] Optimal Brain Surgeon
[4] Dynamic Network Surgery for Efficient DNNs