MicroNet Challenge

CIFAR-100 Classification Task Submission

Model: Sparse Googlenet

By: ID56 (Mashrur Mahmud Morshed)

Overview

My submission is a sparse, weight-pruned Googlenet trained for 200 epochs. I used PyTorch for my implementation. I found the accuracy for Googlenet on CIFAR-100 to be around 78%, which is still a bit lower than the threshold of 80%. Pruning also causes some drop in the accuracy. However, warm-up training, correct learning rate adjustment, label-smoothing and cutout boosts the accuracy significantly.

The submitted checkpoint has an accuracy of 80.18%.

Choosing the Baseline

I reviewed several baseline models initially which had a decent accuracy to parameter count ratio. The models that stood out were the Densenet121 (7M params and ~77% top-1 accuracy), Googlenet (6.2M params and ~78% top-1 accuracy) and the NASNet-A[6@768] (3.3M params and ~76.5% top-1 accuracy).

Densenet121 had trouble pushing up to the 80% threshold, while I found it difficult to replicate the results of NASNet-A [6@768]. However, I was able to tune the hyper-parameters to gain more than 80% top-1 accuracy on the Googlenet, after which I attempted to iteratively prune it to reduce parameters.

Preprocessing:

I used the following preprocessing steps:

- Padding with 4 zeros on each side, resulting in a 40x40 image
- 32x32 random crops
- Random Horizontal Flips
- Channel Normalization
- Cutout of 8x8 pixels

Pruning:

I used weight pruning as described in Han et al. 15 and 16. I used masks of ones and zeros to apply sparsity, and wrote wrappers around Conv2d and Linear layers to define MaskedConv2d and MaskedLinear layers. The pruning threshold was chosen as a quality parameter multiplied by the standard deviation of a layer's weights; the quality parameter is set to 0.25. Although the quality parameter is different for different layers in the paper, I used a single value for all layers.

Training and Iterative Pruning Details:

The model was trained under the following settings:

- Total: 200 epochs
- Stochastic Gradient Descent with a Nesterov Momentum of 0.9 and a weight decay of $5x10^{-4}$
- An initial learning rate of 0.1
- Two initial phases of warm-up training
- A learning rate schedule with milestones 60, 120, 160, 180, 200 with gamma = 0.1 (reduced by $1/10^{th}$ every milestone)
- Label Smoothing of 0.1

The first hundred epochs were trained without pruning to reinforce the connections. Then pruning is done on epochs [101, 103, 105, 107, 109, 111], on Conv and Linear layers alternatingly.

Again, another round of alternating Conv and Linear pruning is applied on epochs [151, 153, 155, 157, 159, 161]. Then the network is trained till epoch 200, where it reaches 80.18 top-1 accuracy.

Scoring:

I built my scoring method from the provided example in the google group, adapting it to work for PyTorch. The Googlenet baseline (with 0 sparsity) contains 6258500 params (6.26M). After pruning and assuming freebie quantization, the model has total 1845672 params (1.845M) and 1627530482 mult-adds (1.627 B). The final score stands 0.2057170217506562, or 0.2057 rounded down to four decimals.