Ali Akbar Izadkhah, Seyed-Arman Ghaffari-Zadeh
aizadkha@andrew.cmu.edu, sghaffar@andrew.cmu.edu
Team Bison (submission name on the gradescope)

Methodology

Model compression is the practice of reducing the computational and storage cost of using deep neural networks (NN) by simplifying - or removing- certain layers, nodes, or weights that are of low importance, while minimizing the loss of accuracy associated with the action. The three model compression methods used here can be categorized into two main approaches; first approach removing unimportant weights across the NN regardless of the layer type and functionality, second approach is specifically targeting the convolution layers and pruning the filters. A brief explanation of the three methods is outlined below:

1. *Magnitude-based pruning*[1]: In this method, the magnitude of all weights from different layers are sorted in an ascending order. Starting from the full NN, the bottom percentile of the weights are filtered out and equated to zero, depending on the targeted pruning degree. Upon removal of the weights, the model is retrained to obtain weights associated with the sparsity.

2. *L1-norm based filter pruning*[2]: In this method, only the filters of the CNN are assessed for the pruning purpose. For each filter, the magnitude of the kernel weights is calculated and sorted in the ascending order. Depending on how aggressive the compression model is aimed to be, the bottom percentile associated with it is removed and set to zero. After removing the filters, the NN is retrained to obtain the updated weights

3. *Learning Efficient Convolutional Networks through Network Slimming*[3]: This method is also concerned with reducing the complexity of the CNN, like method number 2. However, here the filters are not directly set to zero. Instead, by adding a batch normalization layer right after the convolution layers, we assign scaling factors to each filter. The scaling factors are then augmented into the loss function to induce sparsity. The most least important filters are determined and the associated kernel in the convolution layer is then set to zero in the original model (the one whiteout batch normalization layer). Then, retraining is performed on the original NN structure and final weights are extracted.

Comparison

There are a variety of parameters that can alter the performance of each method explained above. These hyperparameters range from number of training epochs, batch sizes, regularization, learning rate, etc. To maintain consistency, at a given sparsity value, the number of epochs is limited to 25 for all methods[4]. The NN is trained for 50 epochs prior to pruning.
Regardless of the method, after pruning, the network is trained again, while freezing the pruned parameters. This was achieved via a custom training subroutine- no software package used- with elimination of the gradients of pruned parameters during the retraining step. This extra step, prolonged the run time and combined with the limitation of free Google Colab version, prohibited us from increasing the number of epochs.
The Pareto frontier of sparsity vs accuracy all three methods is shown in Figure 1.a.

In this figure, method 2 and 3 that target only convolution layers (about 11% of total parameters), confirm that these layers are very sensitive and forcing high sparsity on them will significantly deteriorate the model accuracy

[1]Han, Song, et al. "Learning both weights and connections for efficient neural networks." arXiv preprint arXiv:1506.02626 (2015).
[2]Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).
[3]Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." Proceedings of the IEEE international conference on computer vision. 2017.
[4]Please refer to the submitted code for the detail of hyperparameters

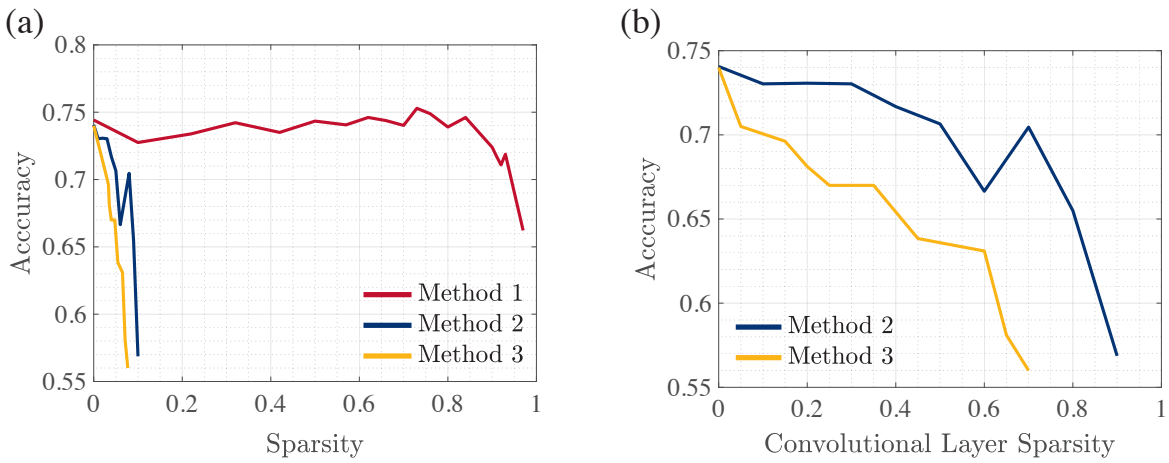

Figure 1: (a) Pareto frontier of sparsity vs accuracy for all three methods. (b) Pareto frontier of convolution layers sparsity vs accuracy for methods 2 and 3 which only prune layers containing the convolution operation.

regardless of retraining. As stated in the literature, the general complexity of convolution layers which usually contain multiple channels and filters lead to such behaviour. Hence, at the limit of full elimination of weights, the maximum sparsity does not surpass this value. Figure 1.b illustrates the accuracy of models 2 and 3 based on the sparsity of CNN, which is the focus of the methods. It can be seen that for up to 0.8 and 0.6 CNN sparsity, method 2 and 3 have accuracy higher than 0.6, respectively.

Method 1 performs well, even at high sparsity values. In this method, we used incremental removal of the weights, as suggested by the literature. This means that the retrained network at each sparsity value is used for further removal of the weights, as opposed to starting from the fully connected - zero sparsity- network. In addition to high accuracy, the implementation of method 1 was the most straight forward. We believe that methods 2 and 3 can be beneficial for networks in which convolution layers dominate the number of parameters, as they provide extra consideration while pruning them. Here, the dense layer, which is a densely-connected NN layer takes up the majority of weights, hence method 1 can achieve high sparsity without greatly compromising the accuracy. We further combined methods 1 exclusively for non-convolution layers and 2 to achieve a better score, however, we were not able to obtain promising results. Nevertheless, we believe this would be the proper approach for this model architecture. We suspect that by tuning the hyperparameters to train the base network, prior to pruning, a better result can be achieved. We hypothesize this based on comparing our scored on the gradescope; the top teams have an accuracy higher than our fully dense and connected network.