Ternary CNN

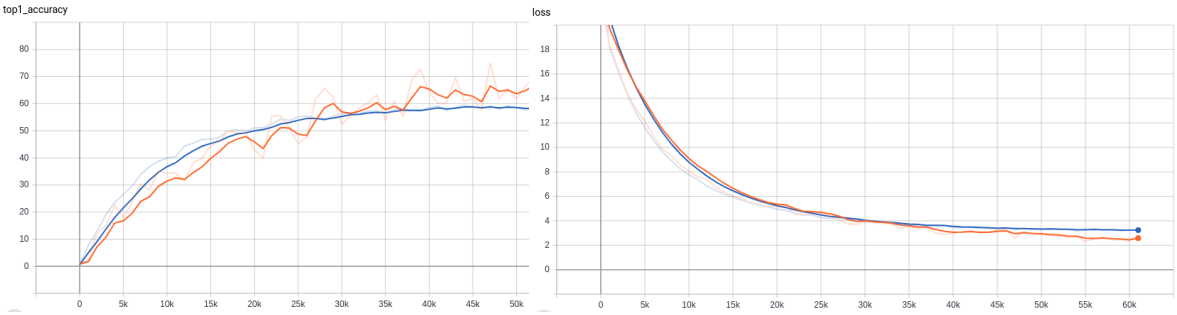
I examined the incremental filter-wise trained ternary quantization method that make use of two independent learnable scaling coefficients for each set of filter parameters together with incremental strategy. Incremental strategy involves several round of ternarization. During each round, a proportion of filter parameters with absolute values higher than a certain threshold is ternarized and the rest of the parameters are trained to make up for accuracy degradation. As the method moves forward, the threshold value decreases. In the final round it becomes zero and all the filter parameters will be ternarized. More importantly, the incremental strategy was planned to help bridge between full precision model and ternarized model, that is, to provide good initial learnable scaling factor at the beginning of each round. Robust initial factors cause less distortion in the architecture when the model parameters are ternarized to these initial value. The rationale is that by each round, part of the parameters are already ternarized and have a uniform value, as a result, taking the average of the parameter set that is going to be ternarized as the the new scaling factors at the beginning of each round, does not move the model drastically away from the optimum state. Actually this hypothesis is justified by experimental results that come later in this report, but the problem is that after quite a few number of training iteration we do not achieve competetively acceptable accuracy.

The experiments are conducted on ResNet101 architecture with tiny-imagenet dataset rather than imagenet dataset to first examine the effectiveness of the method. Tiny-imagenet has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. However, I realized that the relatively lightened training dataset gives rise to the overfitting problem even though the model is downsized by limiting the parameters to ternary values. I will go through this problem and possible solution later in this report. Furthermore, it will take too much time for Compute Canada to allocate the required resources to the code if the model set to be trained on imagenet because the estimated run time increases considerably. To gain an insight into the functionality of each phase of the method, I planned to measure the efficacy of filer-wise ternary operation and further the incremental filter-wise ternarization, separately, and compared the results with layer-wise trained ternary method proposed in Zhu et al. (2016). Also, I found that Li et al. (2016) has developed a filter-wise ternary network, but instead of learning scaling factor, it approximates the distribution of filter parameters as Gaussian and figure

out the scaling factor that minimize the Euclidian distance between the full precision weights and the ternary-valued weights. This method is implemented and used as the benchmark to see how competitive our results are.

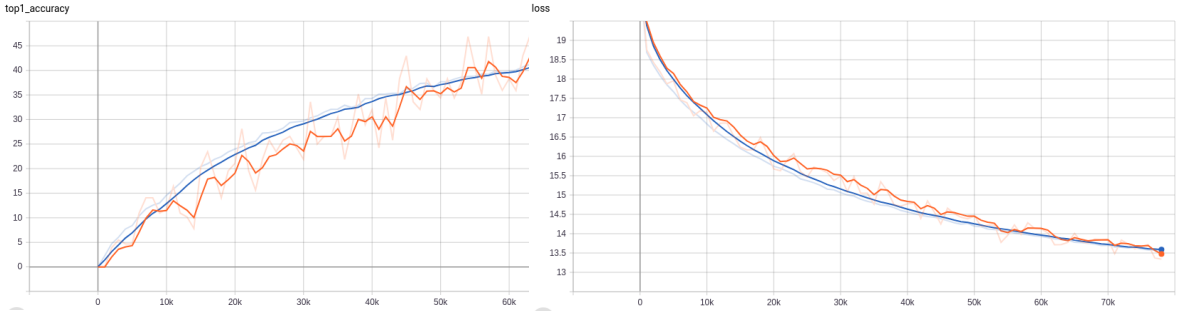
In the following figures that showcase the experimental results the red graph relates to training dataset and the blue one relates to validation dataset. The results for full-precision ResNet101 on tiny-imagenet dataset is shown in Fig1. Although the results are sensible and satisfying to some extent, the model seems too big for this dataset such that we cannot exploit the full capacity of the model. The best hyper-parameters for filter-wise double-factor trained ternary model, resulted from exhaustive search in parameters space, are threshold = 0.05, learning_rate = 2e-5 for ternarized and full-precision parameters, and learning_rate = 1e-7 for scaling factors. Fig2 shows the obtained results. There are two issues that we need to pay attention to, first, notable accuracy degradation compared to full-precision model. Second, it does not outperform the layer-wise approach shown in Fig3 (almost the same results). A potential reason for both issues is overfitting, i.e., the model is yet too big even after it get ternarized, as a result, we can cannot benefit from full capacity of the model and distinguish between the layer-wise and filter-wise schemes. Furthermore, Fig5 and Fig6 show the top1 accuracy and loss value for incremental approach. For the case of loss value, graphs from top to bottom relate to round one to three, and for the accuracy, the graphs are tied in reverse direction. The most suitable quantiles for incremental startegy are [0.5, 0.75, 1]. As the incremental approach moves forward the experimental results indicate decent starting phase at next round that somehow justify the idea that this approach gently traverse between subsequent training courses by providing good initialization for each round. However, after quite a few training iteration it fail to achieve satisfying performance by the end of last round. This degradation might be caused by overfitting as discussed earlier. Moreover, I implemented the approach proposed by Li et al. (2016) as it also performs filter-wise ternarization but does not learn the scaling factor. The obtained results show that it work well in these circumstances and does not indicate any overfitting problem.

A potential solution to cope with this situation might be examining and training the ternaty model on imagenet rather than tiny-imagenet to avoid overfitting. To carry out training on imagenet dataset and get the required resources from Compute Canada in a reasonable amount of time, it needs distributed programming. I am working on this possible solution and running the code on Comute Canada at the moment, but not optimistic that the final results will be satisfying and competitive.



(a) top1 accuracy (b) loss value

Figure 1: Full precision ResNet101 trained on tiny-imagenet dataset



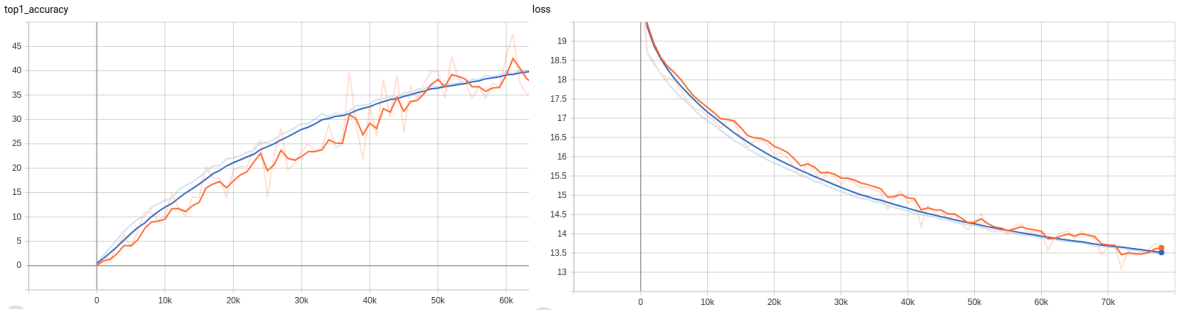
(a) top1 accuracy (b) loss value

Figure 2: Filter-wise double-factor trained ternary ResNet101 trained on tiny-imagenet dataset

References

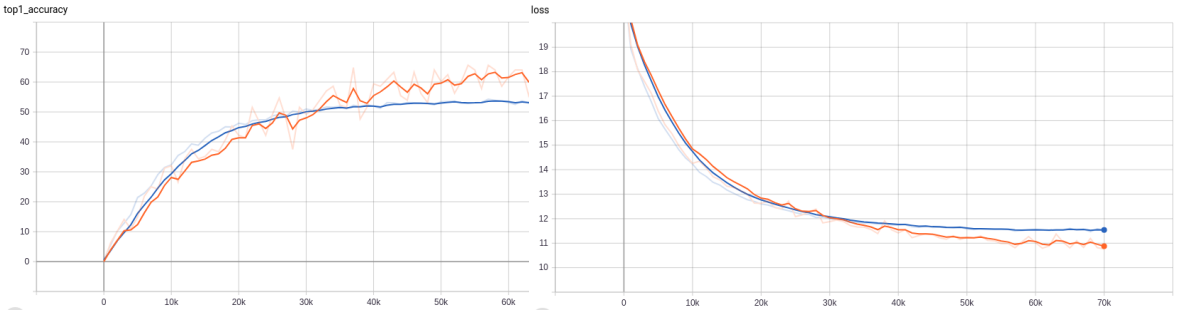
Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.



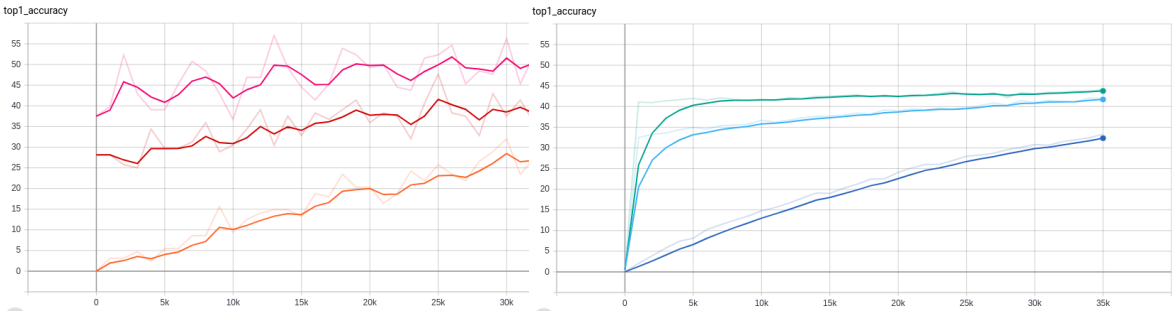
(a) top1 accuracy (b) loss value

Figure 3: Layer-wise double-factor trained ternary ResNet101 trained on tiny-imagenet dataset



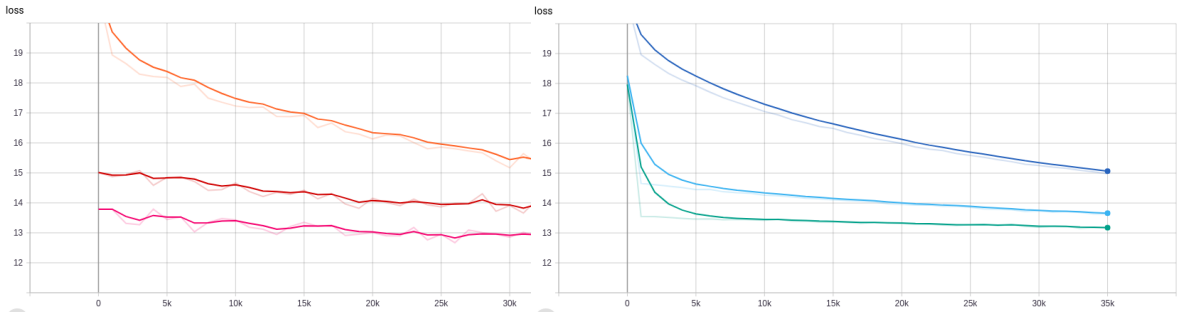
(a) top1 accuracy (b) loss value

Figure 4: Filter-wise single-factor ternary ResNet101 trained on tiny-imagenet dataset



(a) train (b) validation

Figure 5: Top1 accuracy of incremental filter-wise double-factor trained ternary ResNet101 on tiny-imagenet dataset



(a) train (b) validation

Figure 6: Loss value of incremental filter-wise double-factor trained ternary ResNet101 on tiny-imagenet dataset