Speeding up ResNet training

Konstantin Solomatov (06246217), Denis Stepanov (06246218)
Project mentor: Daniel Kang

Abstract

Time required for model training is an important limiting factor for faster pace of progress in the field of deep learning. The faster the model training, the more options researchers are able to try in the same amount of time, and the higher the quality of their results. In this work we stacked a set of techniques to optimize training time of the ResNet model with 20 layers and achieved a substantial speed up relative to the baseline. Our best stacked model trains about 5 times faster than the baseline model.

Dataset

We used CIFAR10[1] dataset. It consists of 60000 32x32 small images of 10 different categories. We choose this dataset for several reasons:

- It's small and thus we can iterate quickly
- It's widely known, and used, and thus often used as a benchmark in papers

Baseline

As a baseline we choose the vanilla implementation of ResNets with 20 layers. Our version of the ResNet implementation achieved 93% accuracy on the test set in 2:42 time. We used the following parameters:

- batch size of 128 items
- SGD training schedule with 200 epochs with starting learning rate of 0.1 reducing 10 times, after 80 and 160 epochs

Related work

During the last two years many approaches were used by researchers to increase learning speed and led to different significant speedups:

- Boosted ResNet [2] (∼3x speedup)
- Stochastic depth [3] (∼2x speedup)
- YellowFin auto-tuning optimizer [4] (∼1.5x speedup)
- Half-precision [5] (∼2x speedup)

Hardware

We had 2 machines at our disposal: one with 4x1080Ti and the other with 1x1080Ti GPUs which were generously provided by our employer JetBrains, Inc.

Multiple GPUs

To distribute the load across several GPUs we used a set of techniques described in [6]. We achieved 3.3 speedup compared to the baseline model.

Half Precision

We used the standard way of working with half precision in PyTorch [7]. Unfortunately, naive usage of the API didn't lead to improved training time and we achieved more or less the same results in approximately the same amount of time.

Pre Activation

Preactivation is a slight improvement of ResNet architecture proposed in [8]. We decided to implement pre-activated ResNet for 2 reasons: it was reported to achieve a better accuracy than a baseline ResNet and this variant of ResNet architecture happens to be more natural in boosted ResNets [2].

Boosted ResNet

We weren't able to completely reproduce results of Boosted ResNets paper [2]. We achieved accuracy of approximately 88% but wasn't able to improve further from that.

Stochastic Depth

Stochastic depth is a method of training deep neural networks in which we throw away ResNet blocks with some probability. Since the model which we train on every step is smaller than the original model, it achieves a speedup dependent on the probability of throwing layers away. We were able to reproduce the results in original paper and achieved a speed up of 1.8 compared to the baseline model.

The Best Model

We achieved the best result with combination of pre-activation, stochastic depth with probability range of 0.5-0.5 and multi GPU training with the batch size of 1024. We used the training schedule from ResNet paper [9]. The model was trainable in 32 minutes on 4 GPUs achieving 93.4% accuracy which is higher than the baseline accuracy.

Results

Model	Time	Train	Test
BaseLine	2:42	1.00	0.93
Half precision	2:42	1.00	0.928
Pre activation	2:42	1.00	0.935
x4 GPU	0:49	1.00	0.92
Stochastic Depth	1:30	0.99	0.931
Best Model	0:32	0.991	0.934

Discussion

We achieved substantial improvement comparing to the baseline. Optimizations which we used stack but they don't stack perfectly, i.e. we got 5 times improvement instead of 5.94 (1.8 x 3.9).

Future Work

Due to limited amount of time devoted to the project we weren't able to completely reproduce the following techniques:

- Boosted ResNets [2]
- Yellowfin auto-tuning optimizer [4]

We didn't have time to try the following techniques:

- MeProp [10]

In future work, it would be nice to finish reproducing the methods described, try methods which we didn't have time to try and to see how they stack together.

References

References

[1] URL https://www.cs.toronto.edu/ kriz/cifar.html.
[2] J. Langford R. Schapire F. Huang, J. Ash. Learning deep resnet blocks sequentially using boosting theory. URL https://arxiv.org/pdf/1706.04964.pdf.
[3] Z. Liu D. Sedra K.Q. Weinberger G. Huang, Y. Sun. Deep networks with stochastic depth. URL https://arxiv.org/pdf/1603.09382.pdf.
[4] C. Ré J. Zhan, I. Mitliagkas. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.
[5] URL https://devblogs.nvidia.com/parallelforall/mixed-precision-training-deep-neural-networks/.
[6] P. Dollar et al. P. Goyal. Accurate, large minibatch sgd: Training imagenet in 1 hour. URL https://arxiv.org/pdf/1706.02677.pdf.
[7] URL http://pytorch.org/.
[8] Sh. Ren J. Sun K. He, X. Zhang. Identity mappings in deep residual networks, . URL https://arxiv.org/abs/1603.05027.
[9] Sh. Ren J. Sun K. He, X. Zhang. Deep residual learning for image recognition, . URL https://arxiv.org/abs/1512.03385.
[10] Sh. Ma H. Wang X. Sun, X. Ren. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. URL https://arxiv.org/abs/1706.06197.