Experiments on Super-Convergence

Roman Bachmann
roman.bachmann@epfl.ch
EPFL

Michael Allemann
michael.allemann@epfl.ch
EPFL

***Abstract*— This miniproject is part of the course CS-439 "Optimization for Machine Learning" taught as a part of the EPFL Data Science core courses in the academic year 2017-2018. The goal of this project is to explore the possibility of fast training with a standard consumer GPU, while preserving decent accuracy. The proposed super-convergence method [1] shows the possibility of training a deep neural network 5-10x faster than previously known methods. However this paper has so far not been accepted to an academic publishing venue [2].**

I. INTRODUCTION

Upon searching for a topic for this mini-project the proposed theme "Meta-Learning: Can you learn the learning rate?" caught our eye. We then found the blog "fast.ai"[2] by Jeremy Howard and Rachel Thomas, in which they address the above mentioned paper [1] concerning a technique for faster convergence. Being regular students without access to high end computer clusters with multiple GPUs, a method promising a speedup of up to an order of magnitude is great news and we decided to conduct experiments testing the proposed method. For our experiments we used a standard ResNet-18 modified to the CIFAR-10 dataset [3] [4]. Different than Smith et al. [1] using a ResNet-56, we took a much smaller network to allow the experiments to run in a reasonable amount of time on a mid-range GPU.

We did not use data augmentation, didn't conduct preprocessing except for normalization. We regularized the network with a weight decay of 5e-4. In the next section we will shortly describe the proposed method, then we will show the results of our experiments, before concluding the paper.

II. METHOD

The gist of the proposed method for super-convergence is to very slowly increase the learning rate linearly while decreasing the momentum, then flipping the process going back to the departing state and in a last phase annealing [5] [6]. Through this heuristic Smith et al. [1] promise avoiding over-fitting and faster convergence, due to the large learning rate. Annealing in this context means decreasing strongly the learning rate at the end of the learning phase, with the idea of gracefully ending in the lowest point of the current valley. In Figure 1 the learning rate and momentum of an experiment of ours are made visible. In this experiment the learning rate was gradually scaled from 0.001 to 0.01 and the momentum was scaled between 0.95 and 0.85.

Before starting the experiments we need to establish a baseline learning rate. To do so we follow the blog of Sylvain Gugger describing a method to find a good learning rate [7],

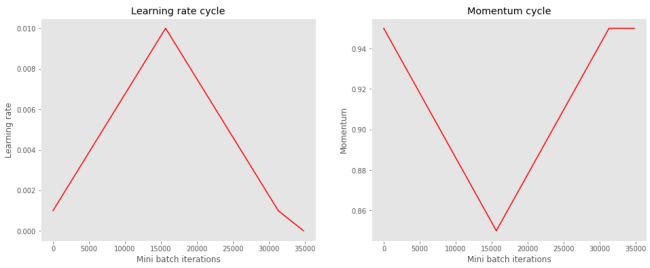


Fig. 1. Learning rate and momentum cycle with annealing

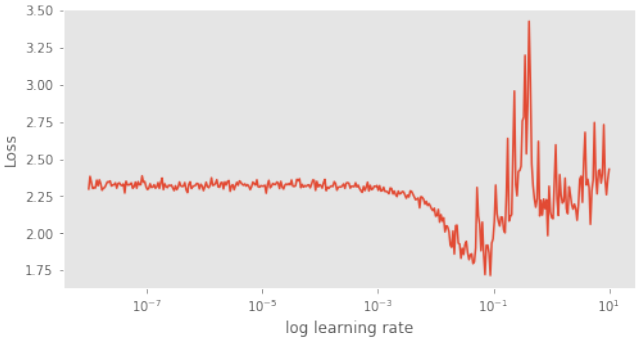


Fig. 2. Learning rate finder for ResNet-18 and CIFAR-10

which is based upon another paper by Leslie N. Smith [8]. Smith proposes to start with a very small learning rate, then gradually increasing over one epoch and plotting the loss with respect to the corresponding learning rate. The loss will first stay fairly constant, then sink quickly and then start jumping all over the place, implying a too large learning rate. The sweet spot lies where the loss decreases quickly, but before it diverges. From Figure 2, we extract 0.01 as our baseline aggressive learning rate. This method is nice, because we just have to go through one epoch, plot and then visually get the learning rate. Starting with this baseline we conducted experiments with varying learning rates, to see if super-convergence holds what it promises.

III. EXPERIMENTS

A. Baseline

As a baseline for our experiments, we trained a ResNet-18 on CIFAR-10 first using the Adam optimizer with the standard suggested learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We chose this baseline with those parameters, because it is a current widely suggested out-of-the-box method.

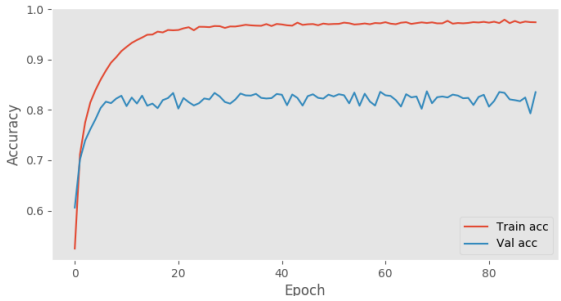


Fig. 3. Adam optimized training and validation accuracies

LR min	LR max	Mom min	Mom max	Ann %	Epochs	Val loss	Val acc
0.001	0.01	0.85	0.95	10	90	0.004281	0.8442
0.001	0.01	0.85	0.95	50	90	0.004103	0.8500
0.01	0.1	0.85	0.95	10	70	0.003112	0.8813
0.001	0.01	0.9	0.9	10	90	0.004340	0.8429
0.01	0.1	0.9	0.9	10	70	0.003018	0.8865
0.001	0.1	0.85	0.95	10	90	0.002912	0.8878
0.01	0.01	0.9	0.9	10	50	0.004067	0.8512
0.1	3	0.85	0.95	10	50	0.004646	0.8018

TABLE I

ALL SUPER CONVERGENCE EXPERIMENTS WITH THEIR LEARNING RATES, MOMENTUMS, ANNEALING PERCENTAGE AND EPOCHS.

As we can observe from Figure 3, the Adam optimized version stopped increasing after epoch 10 and reached a validation accuracy of **83.54%**.

B. Experiments with varying learning rate and momentum

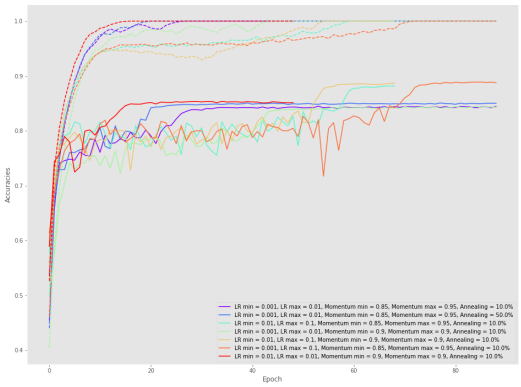
To test the general idea of linearly increasing and decreasing the learning rate and momentum and to see if we can observe super convergence, we set out to train the network with different learning rate and momentum ranges, different percentages of the training we reserve for the annealing phase and a varying number of epochs. Figure 4(a) and 4(b), as well as Table I show the training and validation accuracies and losses for the seven experiments we conducted.

We tested learning rate ranges higher and lower than the baseline in combination with static and changing momentums. For one experiment, we tested a very long annealing phase yielding results on par with methods using a short annealing time. The best performance of over **88%** validation accuracy was achieved by models whose learning rate grows to 0.1, which is a surprisingly high learning rate.

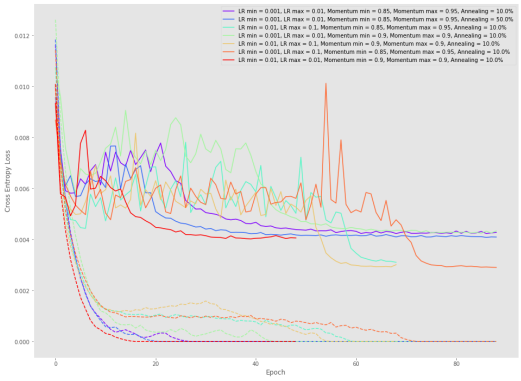
The validation loss perturbations grow quite large during the learning rate peak but completely vanish during the annealing phase.

C. Experiment with very high learning rate

We also experimented with very high learning rates, reaching a maximum of 3 during the cycle. As seen in Figure 5, the training actually gets worse with very high learning rates but quickly recovers. Additionally, we can observe the strong regularizing effect that this kind of training has on the network, since the training and validation curves are not as far apart as in the other experiments.



(a) Training (dashed lines) and validation accuracies



(b) Training (dashed lines) and validation losses

Fig. 4. Training curves from all experiments

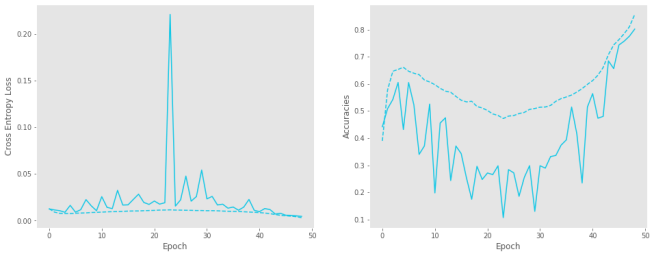


Fig. 5. Experiment with very high learning rate from 0.1 to 3

IV. CONCLUSION

We find that our experiments delivered quite interesting results. For example with a learning rate growing from 0.01 to 0.1 and then back again and momentum scaled between 0.85 and 0.95, 10 percent of annealing and only 70 epochs, we reach a validation accuracy of 0.8813 which by far outperforms the out-of-the-box Adam optimizer over 90 epochs. There seems to be something interesting here, which might be worth looking into more deeply. For now the learning rate and the momentum are just adapted linearly over some fixed number of epochs. Maybe adapting the learning rate and the momentum with more complex and thought through functions could lead to even greater performance.

REFERENCES

[1] Leslie N. Smith and Nicholay Topin. Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates. *CoRR*, abs/1708.07120, 2017.

[2] Jeremy Howard and Rachel Thomas. fast.ai. `http://www.fast.ai/`. Accessed: 24.05.2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[4] kuangliu. Train CIFAR10 with PyTorch. `https://github.com/kuangliu/pytorch-cifar`. Accessed: 24.05.2018.

[5] Sylvain Gugger. The 1cycle policy. `https://sgugger.github.io/the-1cycle-policy.html#the-1cycle-policy`. Accessed: 24.05.2018.

[6] Sylvain Gugger. Cyclical LR and momentums Jupyter Notebook. `https://github.com/sgugger/Deep-Learning/blob/master/Cyclical%20LR%20and%20momentums.ipynb`. Accessed: 24.05.2018.

[7] Sylvain Gugger. How Do You Find A Good Learning Rate. `https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html`. Accessed: 24.05.2018.

[8] Leslie N. Smith. No More Pesky Learning Rate Guessing Games. *CoRR*, abs/1506.01186, 2015.