清華大學

Tsinghua University

Deep Learning Course

Homework 1

-

MNIST Digits Classification with Pytorch

André Cavalheiro - 2019403065

Contents

Introduction

In order to experience the effects of batch normalization in the networks a few scenarios were tested. According to the original paper on batch normalization suggested several improvements, and the different scenarios were chosen in order to highlight these improvements. For one this technique allows the usage of higher learning rates, the training of nets in smaller amounts of time for its faster convergence and excludes the necessity of applying dropout for most cases.

MLP

Experimentations

The chosen network architecture was chosen tacking into consideration the project requirements and the best results obtained in the first assignment being this a two-layer network with widths of 1024 and 512 in the first and second layer respectively.

The Learning rate was varied in conjunction with the Momentum since both influence the degree by which the weights are updated. The experimented values are present in table 1 as well as the corresponding results after 5 epochs of training. The weight decayed was fixated at 0.001 since it was never mentioned in the original paper that this parameter had any specific relevance with the presented method.

Batch Normalization	Accuracy (%)	Running time (ms)	Learning Rate
False	95.7	576	0.01
True	96.4	587	0.01
False	97.1	432	0.1
True	97.8	298	0.1
False	96.2	121	0.7
True	93.4	130	0.7

TABLE 1 - RESULTS FOR MLP ACCORDING TO DIFFERENT PARAMETERS

The respective graphs of the situations expressed in table 1 can be seen by the same order in figures 1-3.

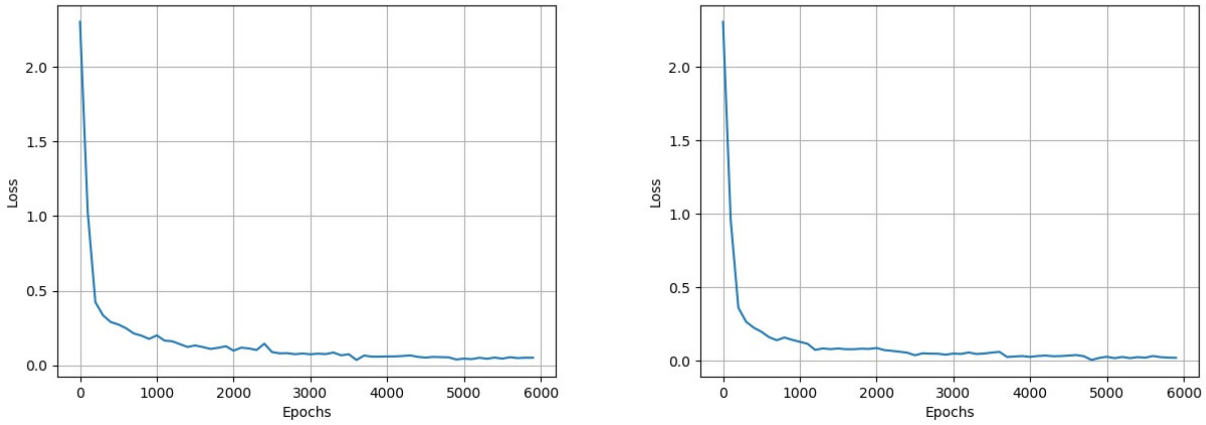

FIGURE 1 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.01)

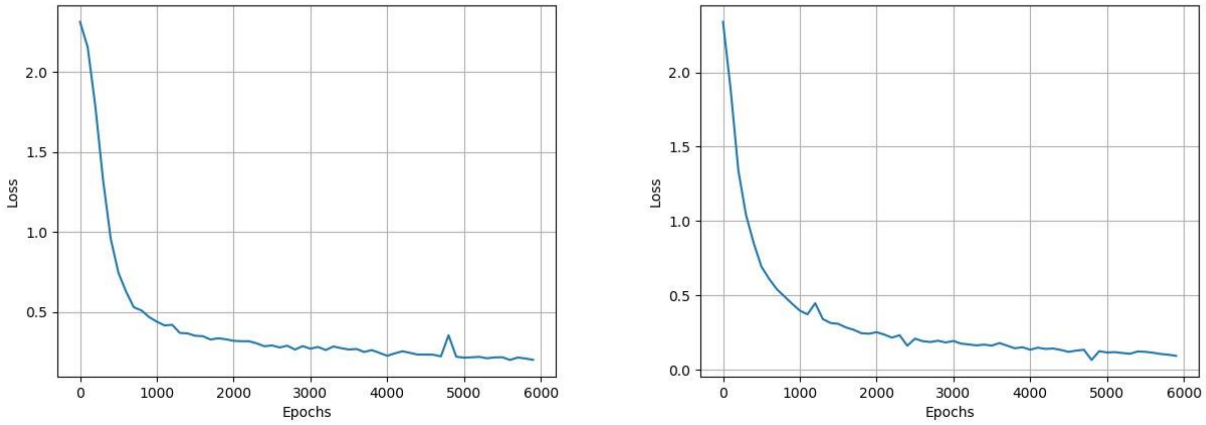

FIGURE 2 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.1)

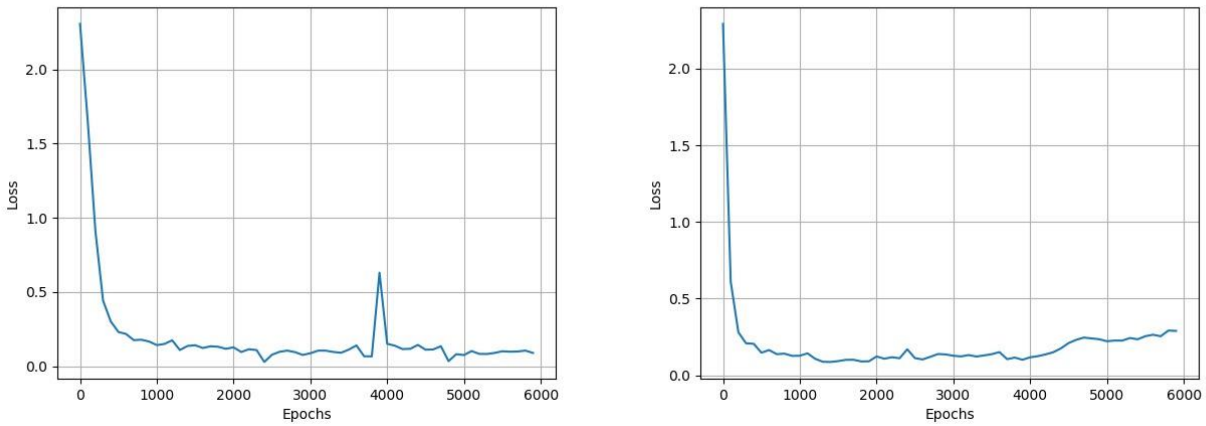

FIGURE 3 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.7)

Conclusions

For the lowest value of learning rate present in all the situations both models perform almost equally, displaying similar accuracies, convergence times and execution times. When we increase the learning rate to more interesting values though we can notice a slight increase in the convergence time for the batch normalization model, especially in the case of figure 3, although for reason I cannot explain after a certain number of time the loss values start to arise again.

CNN

Experimentations

Similarly, to the previous section a network architecture was fixed taking into considerations the assignment requirements and the results from the previous assignment as follows: Two convulsion layers, both with 4 kernels, each with size of 3, stride of 1 and padding of 1.

The weight decay was fixed at a safe value of 0.001 and the momentum at 0.01 for the similar reasons expressed in the first section of this report. The most interesting combinations of hyperparameter values are showed below in table 2 and its corresponding graphs in figures 3-4, all the results are from models trained with 5 epochs.

Batch Normalization	Accuracy (%)	Running time (ms)	Learning Rate
False	91.77	719	0.01
True	94.87	483	0.01
False	94.67	765	0.05
True	96.56	917	0.05
False	11.36	258	0.5
True	97.54	187	0.5

TABLE 2 - RESULTS FOR CNN ACCORDING TO DIFFERENT PARAMETERS

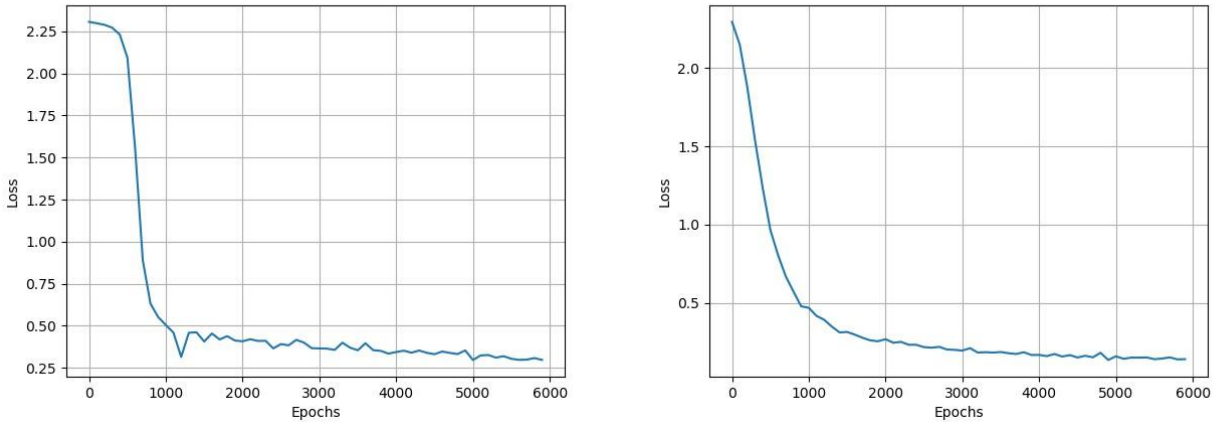

FIGURE 4 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.01)

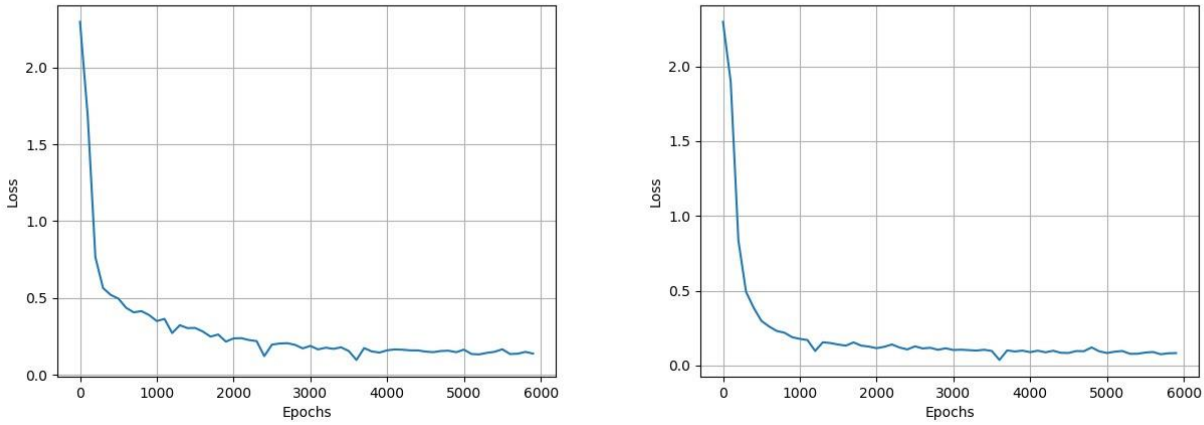

FIGURE 5 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.05)

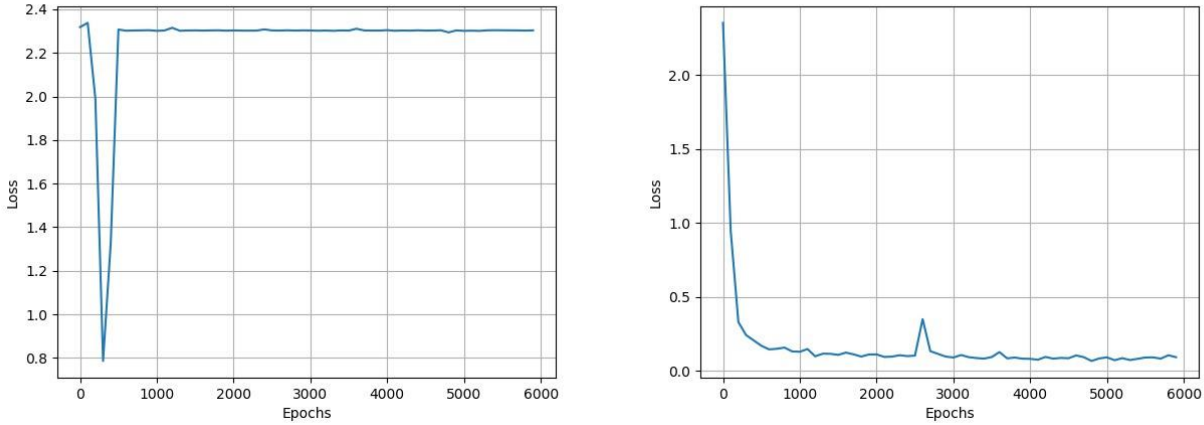

FIGURE 6 - LOSS VALUES EVOLUTION WHILE TRAINING - WITHOUT BATCH NORM ON THE LEFT AND WITH ON THE RIGHT. (LR=0.5)

Conclusions

By analyzing the generated data, we can confirm some of the hypothesis stated by the creators of the batch normalization procedure. From the first 2 situations (represented by pictures 4 and 5) the convergence to acceptable values happened a lot faster than for the situation in which the technique was not applied. We can also verify that from picture 4 to 5, by increasing the learning rate 5 times the effects were a lot more noticeable in the batch normalization model resulting in an even faster convergence. We should also mention that for both situations the batch normalization model performed better showing better accuracy values in both cases. Finally, for the last situation in which the learning rate was increased to 0.5 the model without batch normalization stopped working since the updates are to wild to allow the model to converge to an optimal position. Despite that fact its rival model performed ridiculously well showing the best accuracy of all the three situations. It was also the model that took the least to train in all three situations, also proving that affirmation from the algorithm's creators.