KTH Royal Institute of Technology

Deep Learning in Datascience
DD2424

Assignment 3 (Bonus)

Authors:
Ali Banaei Mobarak Abadi

May 5, 2022

1 Introduction

Here, we try to implement some additional code to enhance the performance of batch normalization. For a short explanation of the code, please refer to the original assignment report. Since all the training logs looks like each other, in this file I did not include them.

2 Estimating mean and variance

We first try to provide a better estimate of mean and variance in order to use in training. As mentioned in the assignment instructions, when we are using the moving average, the last values for mean and variance are highly affected by the last batches, which contain an estimate of these values based on only a small number of samples. To enhance the performance, after training, we perform a forward pass on 10K samples and then set the average and var of this big batch as the values used in testing.

For evaluating this method, the three-layer network with the same setting as the mandatory part of the assignment is trained. After training, the final mean and variance were estimated using the above procedure. The final model reached the accuracy of 0.548, which is almost %2 better than the original model. Note that this difference is not very big, and performing multiple tests and investigating the significance of this difference using statistical methods can give us a better insight into the effect of this method.

3 Adaptive Batch Normalization

For the next part, instead of calculating the mean and variance based on the training data, we randomly selected 1K samples from the test data. Then, each sample is flipped with a probability of 0.5. Finally, μ and v are calculated using these data. A model with the same settings as the previous part was trained. After setting the mean and variance and testing the data on the test set, the accuracy of 0.539 was reported on the test set. As mentioned, this method was expected not to make a big difference in CIFAR-10 dataset and this difference is not that big to let us draw any conclusions.

4 Batch Normalization after non-linearity

Before testing the effect of changing the order of batch normalization and non-linearity, we must check if the model is correctly computing the gradients in reverse mode. So, the same procedure for checking the gradient as the mandatory part was used. The results can be found in Listing 4. As we see, the model correctly computes the gradients, and we are ready to test this feature.The final accuracy of the model was 0.544 which is still slightly higher than the original but not significantly bigger.

```
layer 0:
W: mean=0.03795370009733081 std=0.08359284144364894 mean_diff=-1.0061633095182626e-07
  std_diff=3.6250118914596754e-07 max_diff=1.2624478623601831e-07
b: mean=-0.02178190783396793 std=0.0515975068749352 mean_diff=-5.328076545588201e-09
  std_diff=6.157184468716802e-09 max_diff=2.779680943754137e-09
gamma: mean=-0.00528649954768781 std=0.5476752011626725 mean_diff=-3.4412626946506155e-07
  std_diff=4.894571212071012e-07 max_diff=2.8748783070797757e-07
betas: mean=-0.03594082393816264 std=0.2854065908190191 mean_diff=-2.6687645473893965e-07
  std_diff=1.0941742705260775e-07 max_diff=-9.762020187764348e-08
-----------
layer 1:
W: mean=-0.003579468648332461 std=0.21693702397373868 mean_diff=-2.6451992938825165e-08
  std_diff=1.165612816123496e-07 max_diff=1.1426778612211308e-07
b: mean=0.026380109614135776 std=0.10879234708783916 mean_diff=-1.2929260931326958e-08
  std_diff=1.5967298761064426e-08 max_diff=6.722608969500499e-09
gamma: mean=1.398829512805015 std=0.6245202556368488 mean_diff=4.780568123807782e-07
  std_diff=1.0918580578981339e-07 max_diff=6.228875752967156e-07
betas: mean=0.07111945234878532 std=0.5536566704920242 mean_diff=5.883523039040828e-07
  std_diff=1.2616057247957168e-07 max_diff=7.261606881359128e-07
-----------
layer 2:
W: mean=0.002777392946722759 std=0.16058358636772774 mean_diff=3.5793607620869525e-08
  std_diff=2.1912387635505304e-08 max_diff=9.99622090658292e-08
b: mean=-5.551115123125783e-18 std=0.12153292304128933 mean_diff=2.6112445544127644e-08
  std_diff=1.4198992513904881e-08 max_diff=4.0622780181354123e-08
```

5 Adam Optimizer

In the next part, instead of performing vanilla SGD, an ADAM optimizer was used. This optimizer can change and adjust the learning rate. The implementation is the same as the ones provided in the slides. Also, like in the previous assignment, a decay factor was added to the circular learning rate scheduler. After each cycle, the maximum learning rate is multiplied by this decay factor (here, we set it to 0.6). The network was trained for two cycles this time. The resulting network could reach the performance of 0.564 over test data which is the highest accuracy so far.

6 More hidden nodes

We saw that the network with 9 hidden layers could not perform as well as the more shallow network. In this part, we test a network with more number of hidden nodes in each layer. Note that in order to find the best network, one must perform a grid search to fine-tune hyper-parameters such as the number of hidden units and λ. This is very important since by increasing the number of hidden nodes, we are increasing the complexity and capacity of the model, and hence we may encounter overfitting sooner. So, a more complex model may require more regularization, while a strict regularizer may be too limiting for a simpler network. However, since here we did not have access to high-performance computing platforms, this grid search may take a lot of time, and I have just simply used the default settings as in exercise 2. So, a 9-layer model with hidden nodes as [50, 30, 30, 30, 20, 20, 20, 20]. After training this model for 20 epochs on all the data, the model reached an accuracy of 0.546 on validation data and accuracy of 0.551 on test data.