Exercise 2 deep learning lab 2018

Max Fuchs
Matriculation number: 4340529
maxfuchs@gmx.de

1. Introduction

The task in exercise 2 was to setup an constitutional neural network using tensor flow. This report will try to explain the effects of hyper parameter changes on the validation performance. In the last part random search is used to find a incumbent set of parameters.

2. Implementation

The CNN was implemented using keras. Where in a first step a model-class is created, which is being filled in a second step with alternating constitutional layers and max-pooling layers. After the second set the data is flattened and connected with a fully connected layer with 128 nodes. As loss function the categorical cross entropy is used with stochastic gradient descent. In the last step the model is fit to the training data and tested on the validation data. In the following the validation error is used to measure the validation performance.

3. Exercise 2

The learning rate is a hyper parameter that regulates the update step size in the back propagation. If the rate is set to low, the valid. error will take a long time to converge if it does at all. This can be seen in figure 1 where the learning rates for 0,0001 and 0,001 take very long until they converge. Where convergence is a sign for a local minima, and therefore at least in an local area a good setup for weights and biases. The rates with 0.1 and 0.01 perform well, with a fast reduction of the validation error and convergence after about 6 epochs with 0.1 and 8 epochs for 0.01. In this set of hyper parameters none of the rates is to high. If the learning rate is set to high, the network does fast progress in improving the validation performance, but it is likely to not find the optimal parameter setting. This is due to the fact that large update steps may “step” over a local minima. From table 1. it can be seen that learning rate 0.1 performed the best. Typically 0.01 is a good starting point for a parameter search. In this case it performed worse than 0.1 because of the relative high batch size of 128. If the batch size would be smaller the errors for both learning rates would be in the same range.

Learning rate	Final validation error
0,1	0,0127
0,01	0,029
0,001	0,0915
0,0001	0,78

table 1: final validation error with: epochs:15;number filters:16;filter size:3x3;batch size:128

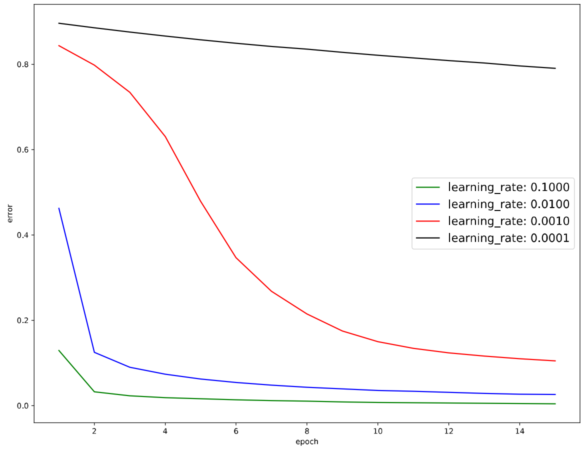

figure 1: comparison learning rate, with hyper parameters: epochs:15; number filters:16; filter size:3x3;batch size:128

4. Exercise 3

The filter size determines how locally we are searching for low level features in our data. Where locally is defined by our filer size. Thus a 1x1 filter would assume our low level features are per pixel and they don’t effect their neighboring pixels. Whereas the other extreme, the kernel is of the image size, would lead to a fully connected layer and no assumptions on low level features are made any more.

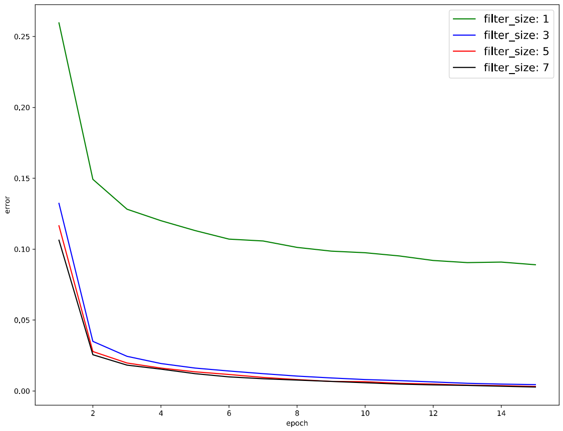

figure 2: comparison filter size, with hyper parameters: epochs:15; number filters:16; learning rate:0.1;batch size:128

Filter size	Final validation error
1 X 1	0,1262
3 X 3	0,0126
5 X 5	0,0117
7 X 7	0,0147

table 2: final validation error for changing Filters with: epochs:15;number filters:16; learning rate:0.1;batch size:128

The result of figure2 and table2 supports the assumption that a filter of size 1 is not able to generalize well enough. The error is about 10 times bigger than the error of the other filter sizes. The results for 3,5,7 are very close together, after a 6 epochs there is only minor improvement in the validation error. The curve for filter size 5 performed slightly better than 3 and 7. This spacial dimension seems to fit the low level features of the pictures the best.

5. Exercise 4

In this task the hyper parameters were optimized with random search. For that, ranges were defined, were the optimal hyper parameters are expected. The HpBandster tool randomly chooses from these intervals and evaluates the performance of the network. With the learning rate, batch size, and number of filters on a logarithmic scale, and the filter size as a categorical parameter. For each hyper parameter set the validation performance is calculate for the first 6 epochs. This is done 50 times, namely the intervals. After that the hyper parameter set with the best performance is chosen and evaluated on the test data. The performance of each iteration can be seen in figure4. Figure4 shows the incumbent error on the validation data after each iteration.

Optimal hyper parameters	Validation error	Test error
Learning rate: 0,0692 Filter size: 3 Number of filters: 56 Batch size: 16	0,0106	0,0094

table 3: optimal parameter setting

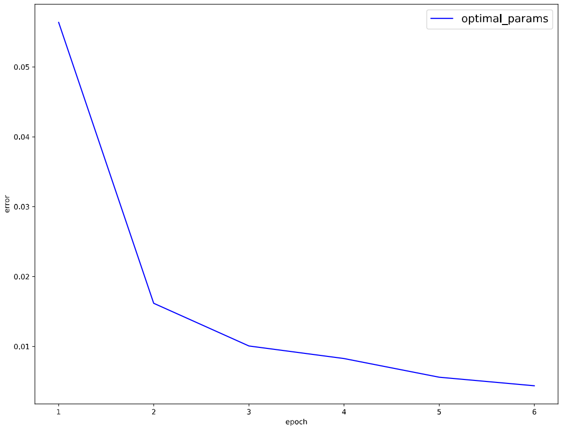

figure 3: learning curve with optimal hyper parametersetting: validation error over epochs

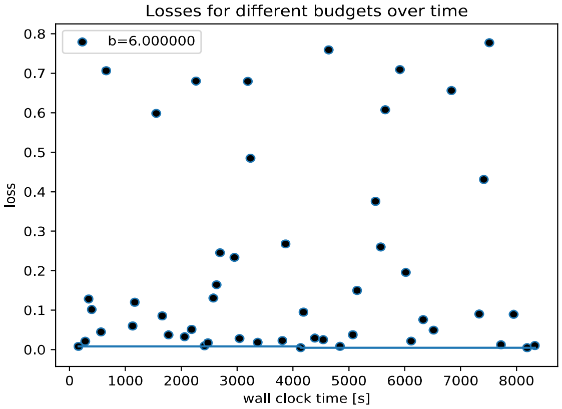

figure 4: best error of each iteration with 6 epochs

The optimal hyper parameter setting achieved a test error slightly under die validation error of around 1 %. This error could be further improved if the network parameters would be further trained on the training data. This assumption is due to figure3 were the learning curve of the net is not fully converged after the 6 training epochs.

The results from ex.2 and 3 match quite well with the results shown in table3. From ex.2 we concluded that most likely the optimal learning rate will be within the interval of 0.1 to 0.01. Which holds true for the final optimal setting. From ex.3 we deduced that a kernel size of 3 X 3 or 5 X 5 would be a good setup.

This assumption also holds true for the results presented in table3. Interestingly as we already expected in ex.2 the batch size is better chosen to be smaller than the batch size setup from ex.2, compare table1 and table3.

6. Conclusion

The results of the measurements show that hyper parameters are dependend on each other and its important to find a well performing combination to save time and on the other hand increase the performance of the network. Random search offers a reasonable good tool to find a good hyper parameter setting, and with GPU calculations the time consumption is justifiable.