Report Exercise 2

The Network was derived from the Tensorflow Tutorial for MNIST. The size of the filters was adapted according the the assignment, the drop-out regularisation was taken out, and the Adam optimiser was replaced by a plain vanilla gradient descent optimiser.

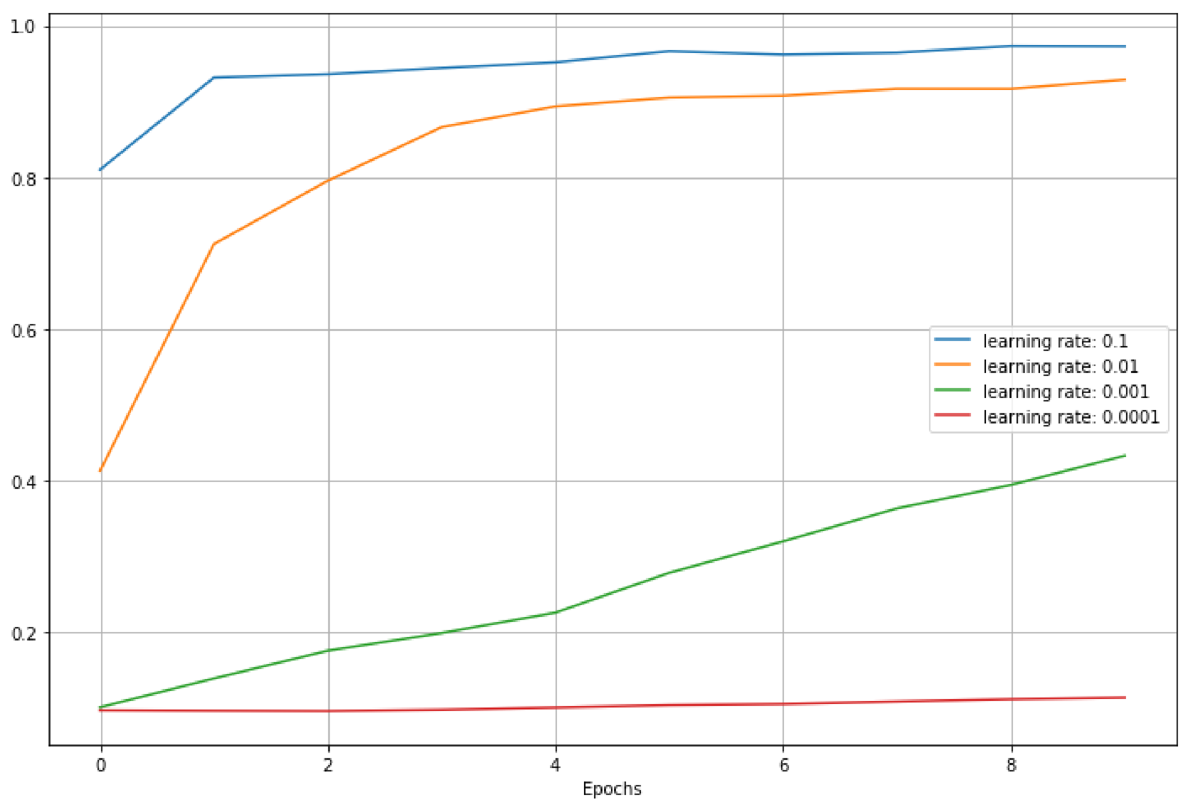
The definition of the graph and its execution was packaged into functions to facilitate the tests.

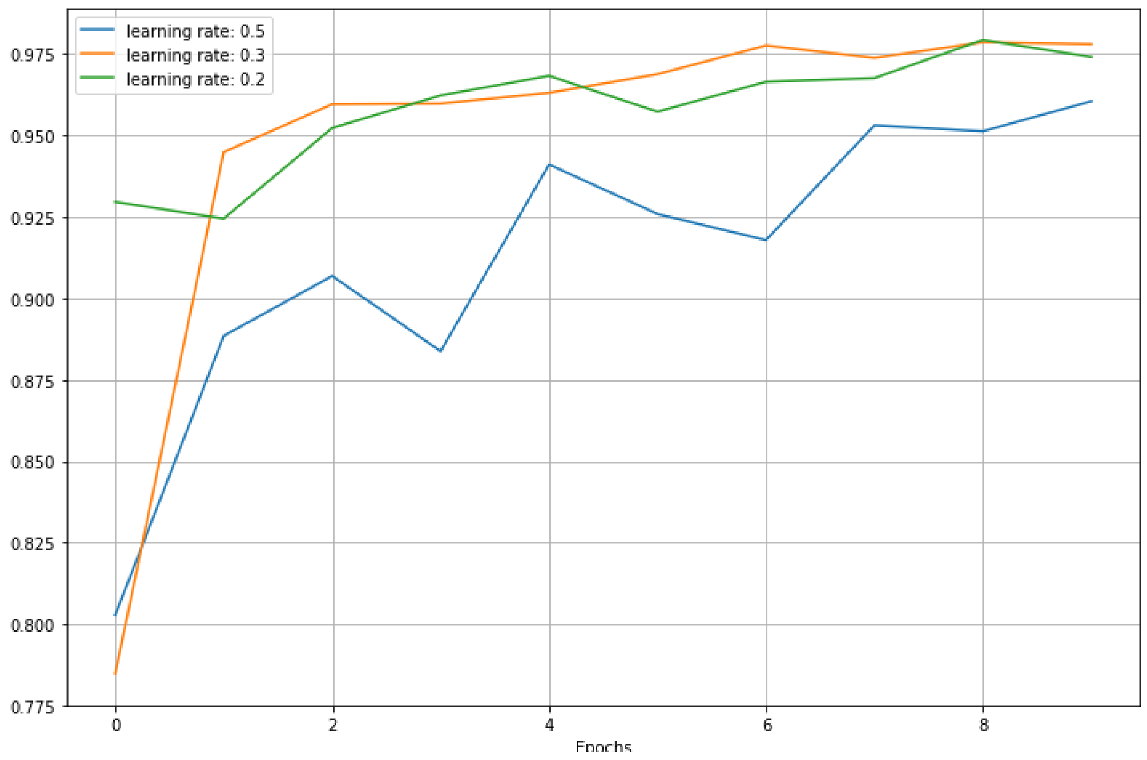
During the test with the different filter-sizes, and resulting total number of parameters, we realised, that before building a new graph "tf.reset_default_graph()" must be called. Otherwise the total number of parameters increases with every new graph created.

Task 2, learning rates

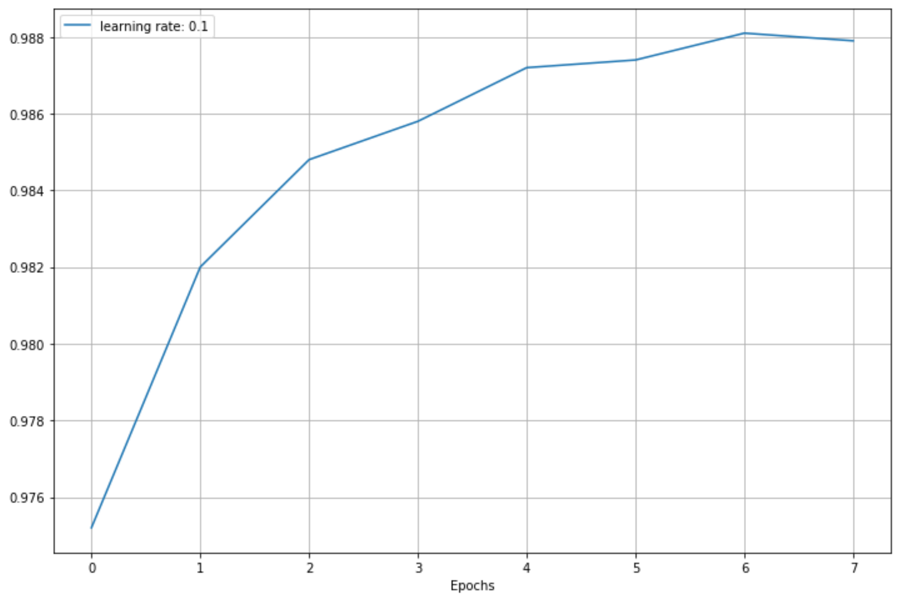
In order to limit the run-times, during training, one epoch was defined by 100 batches of 50 samples, and thus about 10% of the total training dataset. Total Training was 10 such epochs.

The performance declined rapidly with the learning rate, the largest rate 0.1 showed by far the best results (see graphik). The optimal learning rate thus seemed to be located outside the range of values of the assignment, therefor a second run with a set of higher learnig rates was performed.





An optimum seems to be somewhere in the region of 0.3. A longer testrun with 8 epochs over 1.000 batches of 50 samples led to an accuracy of 98,8%. This seems to be the best achievable result with this configuration.

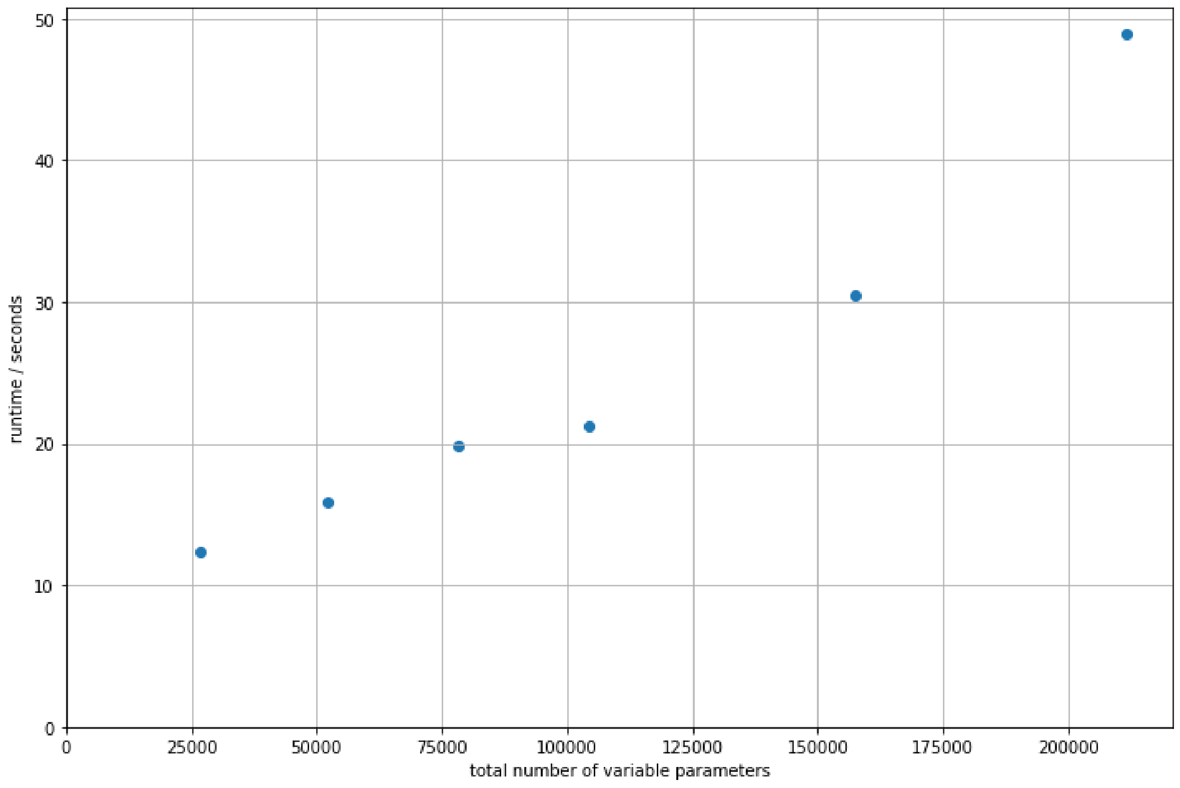


Task 3 a, runtime as a function of number of parameters (CPU)

Running the trials on different filter sizes on the CPU machine (Intel I7 with 8GB ram) led to “ressources” errors with Filters larger than 32 (even 40 was not possible). So in order to obtain a meaningful number of samples a test array of (4,8,12,16,24,32) was chosen.

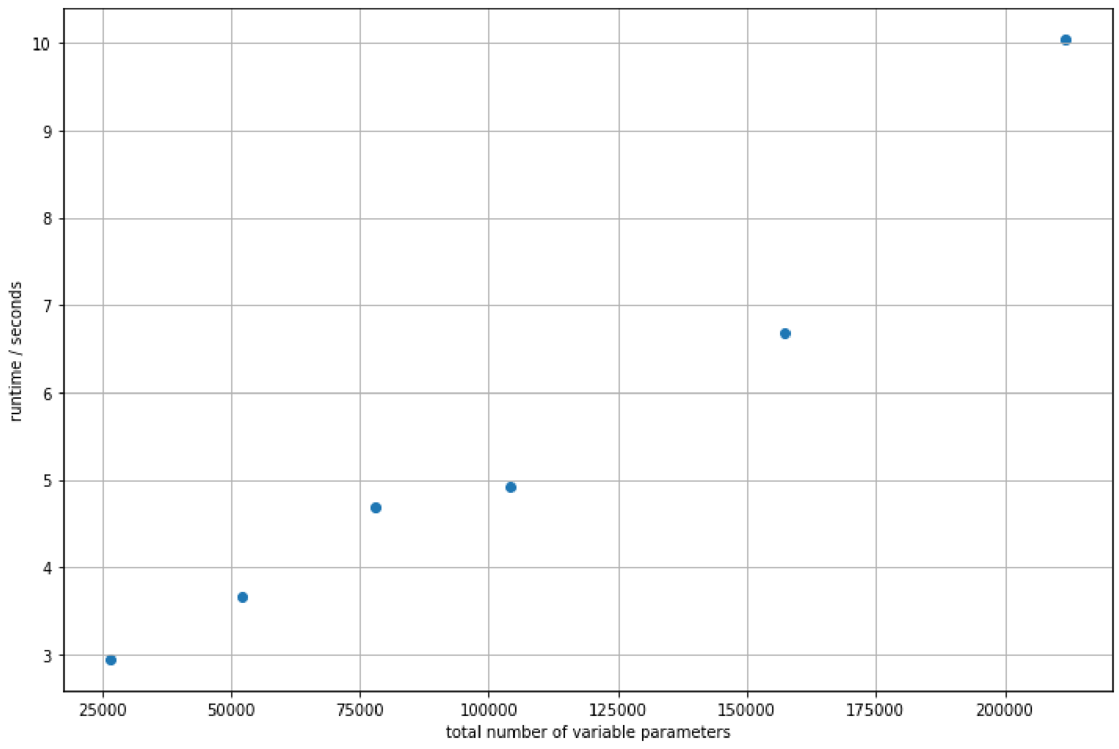
The Training runs on the CPU were performed with 4 epochs, each with 50 batches of size 50. This was a relatively small number (only part of the total training sets was used), but should be sufficient to reveal the relation between the number of parameters and execution time.

The results shows a nearly linear relation between the number of parameters and the runtime. More exaclty, an affine relation with an offset of ~ 8 seconds. The last value (32 filters / >200.000 Parameters) seemed to be ~20% above this approximate linearity.



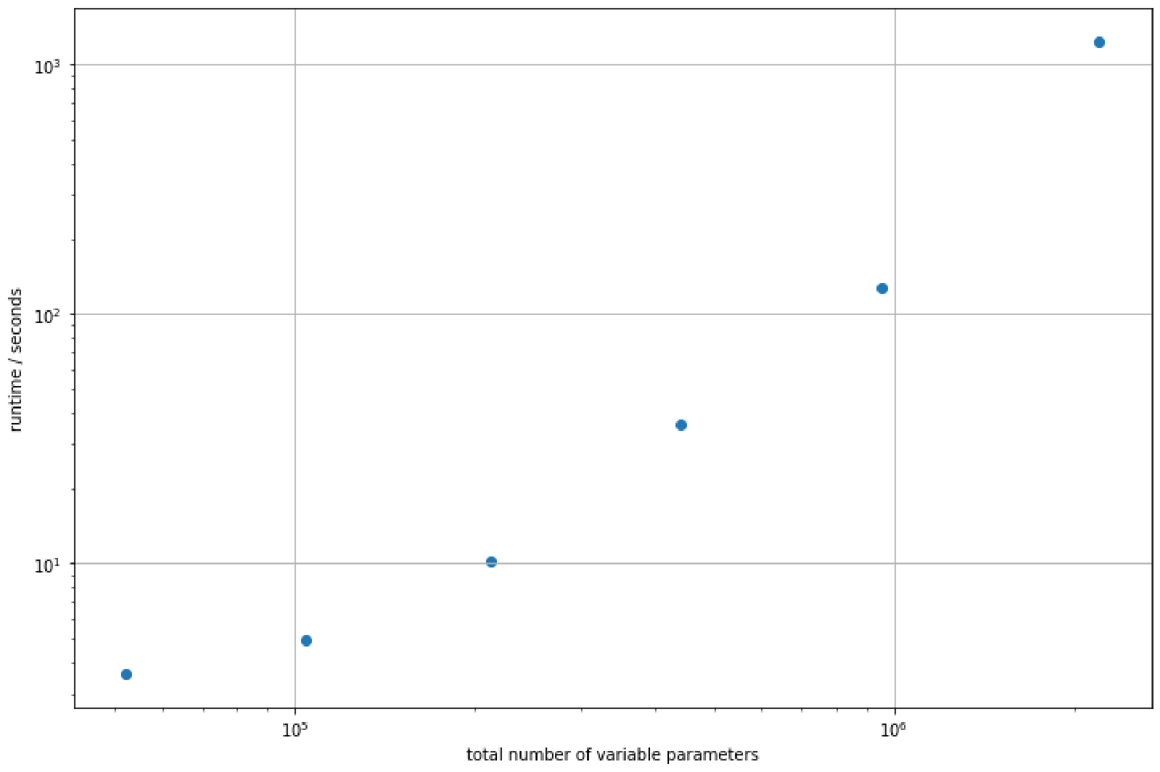
Task 3 b, runtime as a function of number of parameters (GPU)

The same tests were performed on the GPU machine. First the same set of filter sizes was tested as on the CPU machine, to obtain a direct comparison between CPU and GPU.

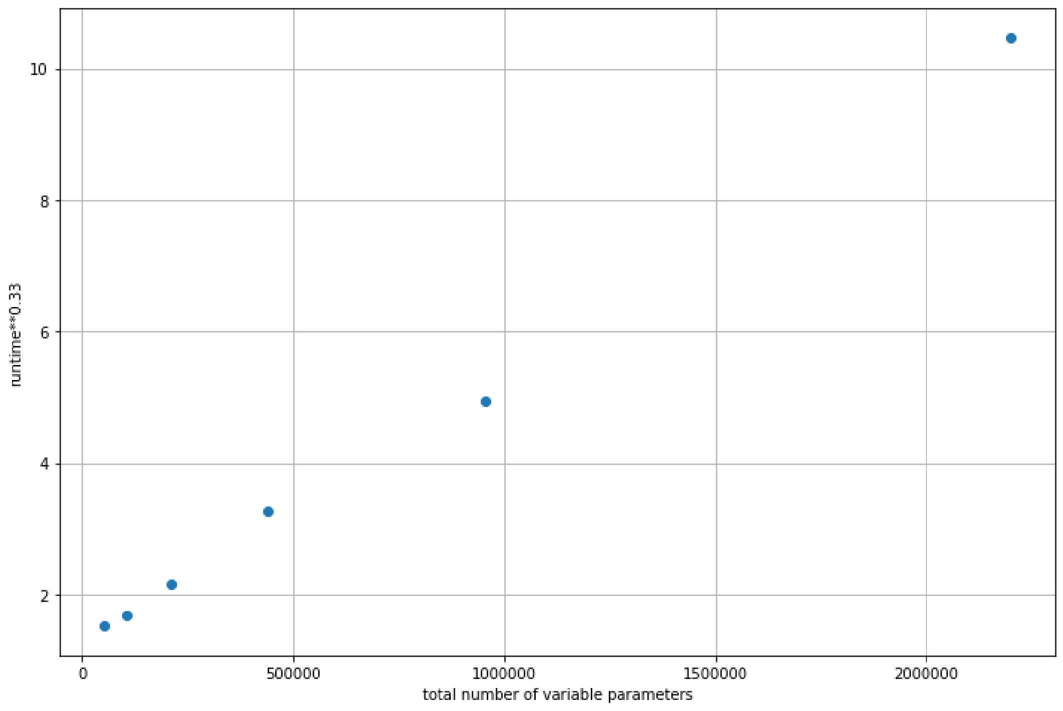


The distribution was highly similar, with the speed approximately 5 times higher.

Next the Filter-sizes (8,16,32,64,128,256) were tested. The result can best be viewed when both axis are displayed at a logarithmic scale. It can be seen, that the runtime scales faster than linear with the number of variable parameters (the same appears with non – logarithmic scales).



In order to understand the relation between the number of parameters and runtime, varoius powers of the runtimes (runtimes**x) were plotted against the number of parameters on non-logarithmic axes. It turns out, that adjusting the runtimes with a power of 1/3 leads to an approximately linear distribution. That means, that the runtime scales in an approximately cubic fashion with the number of parameters.



This behaviour differs from the observations with the lower number of parameters above (up do 32 Fileters). There the number of Parameters was between 25.000 and ~200.000 Parameters, while with the large Filtersizes from 64 to 256 they are by a magnitude higher (up to ~ 2,2mio.)

An explanation for this non linear behaviour might be the increased memory consumption. Eventually the calculations (matrix multiplications with Parameters and data samples) do not fit into the memory of the GPU any more, thus requiring time consuming memory swapping.

<u>Observation about relation between network performance and the Number of Filters</u>

The performance increased significantly from 0.908 to 0.936, when the filter size was augmented from 8 to 16. It then remained at approximately this level for the higher numbers of filters up to 64, and even declined at the filter sizes of 128 and 256 to 0.923 and 0.921.