Assignment 2:

Outline:

The goal is to get familiar with Tensorflow by implementing a small CNN and training on MNIST. The deliverables are plots of the learning curves with different learning rates and a scatter plot with execution times over filter sizes.

Tensorflow:

My first version which I implemented as the Tensorflow estimator library was very fast. I soon realized that with that architecture I had no chance to get the validation error after every epoch (efficiently) so I changed my network completely. I then used a Tensorflow tutorial from the Tensorflow website. This tutorial uses loops and the from the tutorial known session style. It was easy to adjust the tutorial and throw out the parts which aren't necessary for these tasks. After I did some checks if the network works properly and added the logic for task 2.2 and 2.3.

Result:

As described before was it easy to set up the network. I think one reason for that is, that the last exercise forced me to understand what I do completely. So almost everything was setup in a short amount of time. The really time consuming part was the training of the network. I trained it at home on my linux machine with Nvidia 1060. For the exercise 2.2 it worked 7 seven hours (I think calculating the validation error after each epoch was the real breaking part (In my first solution with the estimator it was easy to calculate 20.000 epochs in little time)). Both results, from 2.2 and 2.3 were as I expected (knowledge from the ML course).

Parameter Calculation:

With the formulas from the Stanford lecture:
Example with 16 filters:
First Layer:
3*3*16*1 + 16 = 160
Second Layer:
3*3*16*16 + 16 = 2320
Fully connected layer:
7*7*16*128 + 128 = 100480
Sum: 102960 parameters

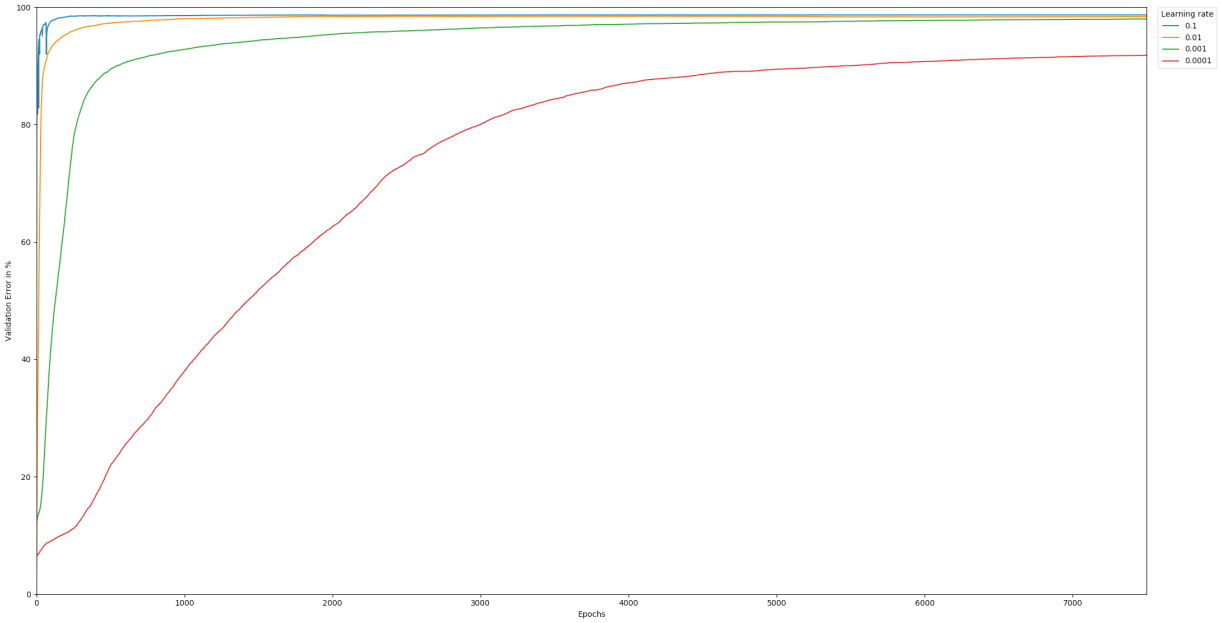
Table 1: Number of parameters for number of filters

Filters	8	16	32	64	128	256
Parameters	50968	102960	210400	439104	951808	2220218

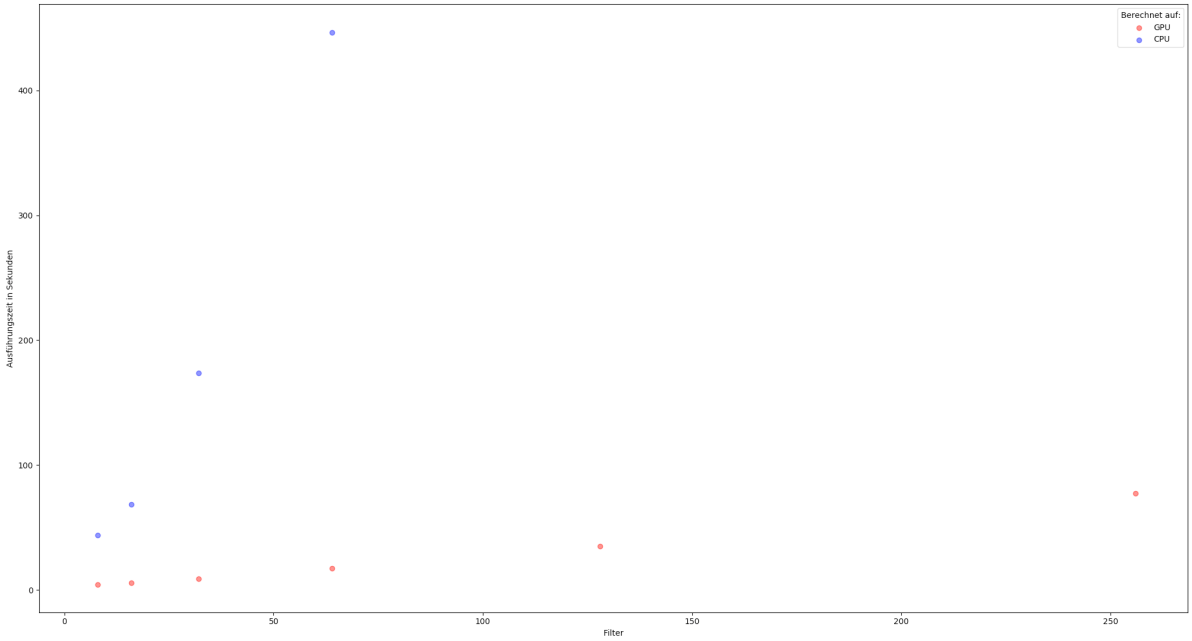
From Table 1 we can see that the number of parameters is proportional to the number of filters.

Runtime GPU/CPU comparison:

The runtime on the CPU is increasing exponentially for the measured number of filters (See Figure F1). For the GPU it seems to be also an exponential growth but way slower than on CPU. That the GPU has an advantage over the CPU is obvious because it can do way more operations in parallel.



L1: Learning curve with different learning rates (the png file is also in the repo)



F1: Execution time in seconds over number of filters (png is also in the repo)