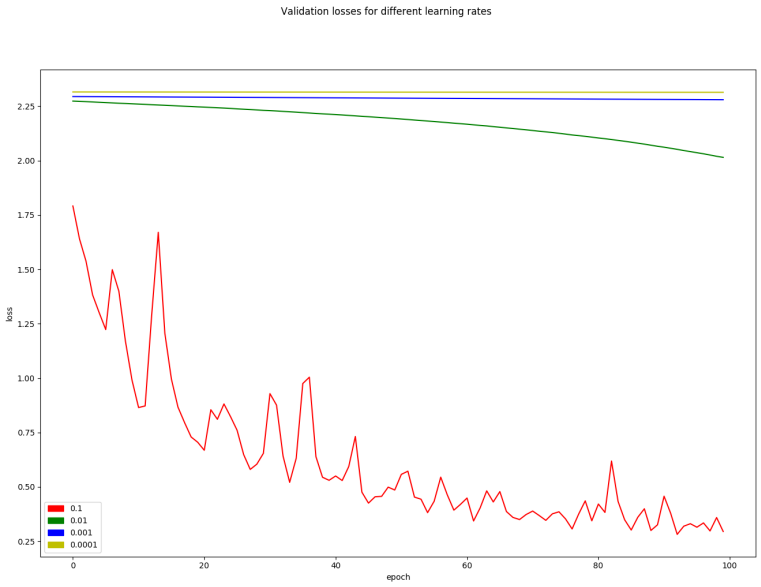
Exercise 2: CNNs

To build the CNN, I chose to extend the example in the tensorflow documentation using an estimator. The adapted model function takes the learning rate and number of filters as parameters to use it for the experiments. The convolutional layers are padded to retain their size and and the pooling layers are max pools of size 2x2 with a stride of 2, thus cutting the layer size in half. As specified in the exercise, the fully connected layer has 128 units.
To obtain the MNIST data, the function from exercise 1 was used, as suggested. For the experiments themselves, a new estimator per changed attribute is created with its own model directory. Then it's trained using the estimator's built-in version of Stochastic Gradient Descent with a batch size of 64 and a manual iteration through the training epochs. This is done to be able to calculate the validation loss for each epoch in an easy way. All visualization is done using Matplotlib.

Experiment 1: Different learning rates

For this experiment, the number of filters was 16 and 100 epochs were used. The learning rates that were tested were 0.1, 0.01, 0.001 and 0.0001. Here's the result of the validation losses plotted against the epoch they were obtained in:

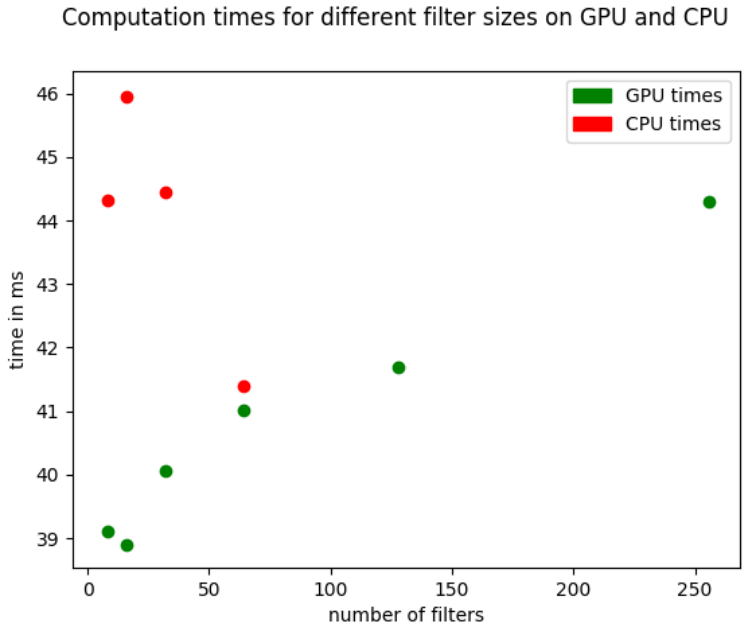


The graph shows that 0.001 and 0.0001 are extremely slow and the validation loss hardly changes. 0.01 is still slow, but there's a recognizable downwards trend of the loss. It's, however, not even close to a learning rate of 0.1. This

value provides a less smooth graph, but a much faster minimization of the validation loss down to near zero, whereas the loss using the other learning rates hardly moved past 2 from a starting point of roughly 2.3.

Experiment 2: Filter sizes of GPU vs. CPU

Here, the learning rate was set at 0.1 (as we don't really care for the training result here) with 100 epochs for each number of filters to. The results were computed on a single GPU and CPU respectively. The scatterplot of GPU and CPU times looks like this:



As expected, the GPU was faster than the CPU by quite a bit. In fact, the GPU needed about as much time for 256 as the CPU for 16. So this result is exactly what could be expected from comparing GPUs and CPUs.
Additionally, there's quite a bit of variance in the CPU times that can't really be found in the GPU times. The computer wasn't used for anything else but the script at the time, but it's entirely possible that some processes in the background caused the computation to be a bit slower than it necessarily was.