Report for Assignment 2

The code committed is based on the tutorial for the MNIST dataset provided by TensorFlow. Two main modifications have been applied:

- The layers in the network have been modified to match the parameters given in the exercise sheet.
  - Two convolutional layers (16 3 × 3 filters and a stride of 1).
  - Each convolutional layer followed by ReLU activations and a max pooling layer.
  - A fully connected layer with 128 units.
  - A softmax layer to do the classification.
  - Optimise the cross-entropy loss with stochastic gradient descent.
- TensorBoard functionality has been added. For this, I instantiated writers and saved for every 100 epoch a summary with the validation accuracy. TensorBoard then automatically reads this data and plots the result, when the following command is run:

  ```
  tensorboard --logdir=/path-to-dir-with-files
  ```

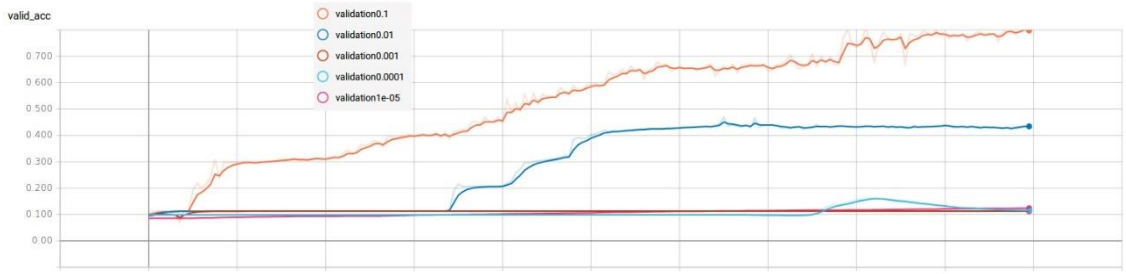
  In the code, the folder is set to `~/lab/tensorflow`, but it can be modified by changing the "`--log-dir`" in the parser. The summaries for validation are in a /validation subfolder.

  Different runs should be in different subfolders. To achieve this, I attached the variable I was studying (learning rate or number of filters) at the end of the path.

Changing the learning rate

I run the network with learning rates of 0.1, 0.01, 0.001, 0.0001 and 0.00001 and 20.000 epochs.

This is the TensorBoard corresponding to those executions



It is easily appreciated how a bigger learning rate usually means faster learning. However, values too high would end up in a worse performance due to jumps too big. This doesn't happen here because the biggest one (0.1) is not too high.

Changing the number of filters

In order to calculate the number of parameters per run, I followed this equation:

$$\sum_{conv_{layers}} \left(\left(\mathrm{h}_{\mathrm{filter}} \cdot \mathrm{w}_{\mathrm{filter}} \cdot \mathrm{d}_{\mathrm{input}} + 1\right) \cdot \mathrm{filters}\right) + \left(input_{fc} + 1\right) \cdot units_{fc}$$

- 8 filters: 50968
- 16 filters: 102960
- 32 filters: 210400
- 64 filters: 439104
- 128 filters: 951808
- 256 filters: 2198400

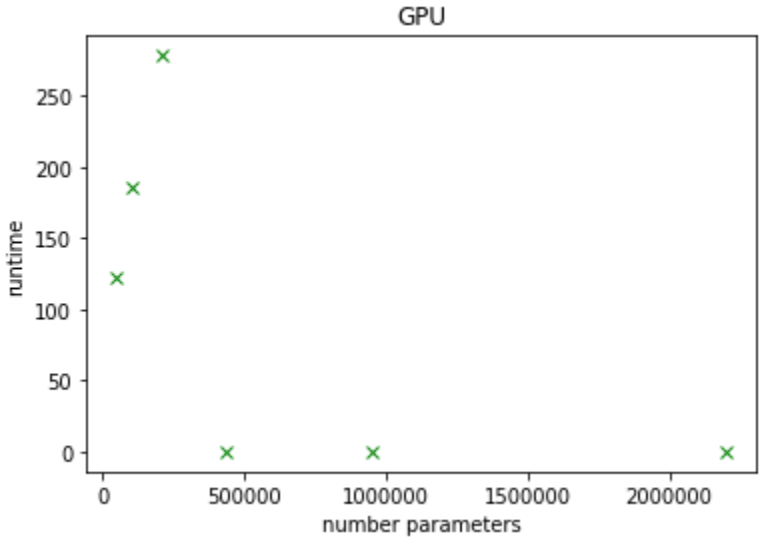
Using GPU

I set a learning rate of 0.001 and run 7000 epochs. The time measures were as follows:

GPU (7000 epochs, lr 1e-3):

- 8 filters, time: 122.705 seconds.
- 16 filters, time: 185.701 seconds.
- 32 filters, time: 278.410 seconds. Crashed just before test
- 64 filters, time: couldn't start
- 128 filters, time: couldn't start
- 256 filters, time: couldn't start

The GPU started running into trouble when going through the test set with 32 filters. For higher filters it couldn't even start

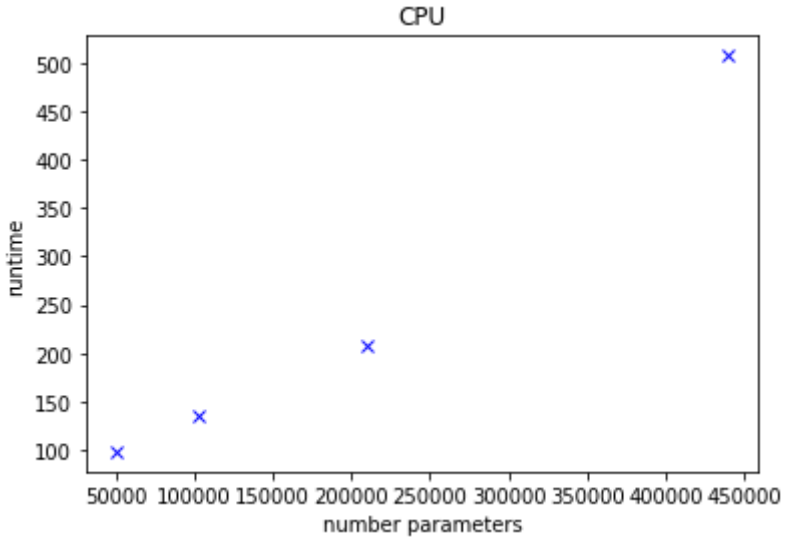


Using CPU

I added the code `os.environ['CUDA_VISIBLE_DEVICES'] = '-1'` to hide the GPU from the program, so it would choose the CPU.

Again, I set a learning rate of 0.001 and run 7000 epochs. The time measures were as follows:

- 8 filters, time: 98.310 seconds.
- 16 filters, time: 134.524 seconds.
- 32 filters, time: 208.262 seconds.
- 64 filters, time: 507.323 seconds.



Comparison

When both settings are compared, it is surprising to see that actually the CPU performs better. I believe this is because the number of filters is small. If the GPU hadn't run out of memory, I believe the GPU would have outperformed the CPU.