Deep Learning Lab – Assignment 2
Hakan Yilmaz

Introduction

The goal of this assignment is the implementation and the investigation of a small convolutional neural network (CNN) in Tensorflow (scaled-down version of LeNet) applied to image classification (MNIST digits). During the training phase of the network, different learning rates and numbers of parameter were used to investigate their effect on learning performance and run times. Herein, only a CPU (AMD A10-5750M APU) was used.

Implementation of the CNN

The network architecture follows the description in exercise 1 of the exercise sheet.
For exercise 2 (learning performance with different learning rates), the default number of filters (16) was used.
For exercise 3 (runtime performance with different numbers of filters), the default learning rate (0.01) was used.
For all exercises, the batch size was 500 resulting in 100 steps per epoch and the number of epochs was 20 (https://www.tensorflow.org/tutorials/layers#create_the_estimator).

Exercise 2: Changing the learning rate

For this exercise, for each value of the learning rate in {0.1, 0.01, 0.001, 0.0001}, the validation error was logged during training of the CNN. The individual learning curves were plotted in one figure.

%%% Learning curves figure %%%

From this observation, we can conclude that higher learning rates lead to faster convergence but lower precision or can even lead to zig-zagging around the local minimum. Small learning rates lead to high-precision local minima and a reduced risk of zig-zagging, but require more training time for that precision.

In this case, the value of %%% for the learning rate works best.

Exercise 3: Runtime

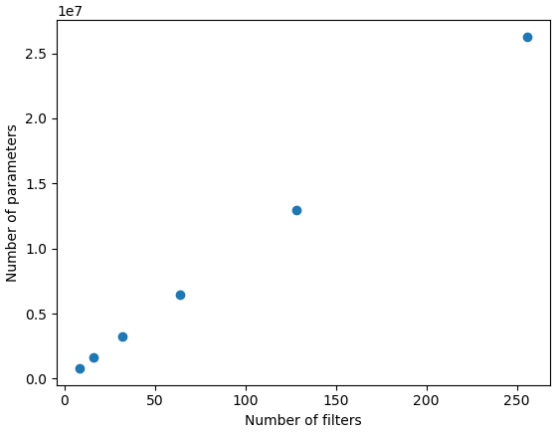
For this exercise, for each value of the number of filters in {8, 16, 32, 64}, the runtime was logged during training of the CNN with the mentioned CPU. First the relation between the number of filters and the number of parameters is visualized, then the runtime w.r.t. the number of parameters is presented.

Number of parameters in the 1st layer: $\#params1 = \#filters \cdot (3 \cdot 3 \cdot 1 + 1)$

Number of parameters in the 2nd layer: $\#params1 = \#filters \cdot (3 \cdot 3 \cdot \#filters + 1)$

Number of parameters in the 3rd layer: $\#params3 = 128 \cdot (28 \cdot 28 \cdot \#filters + 1)$

Total number of parameters: $\#params\_sum = \#params1 + \#params2 + \#params3$


1e7
2.5
2.0
1.5
1.0
0.5
0.0
Number of parameters
0
50
100
150
200
250
Number of filters