Laboratory - Deep Learning Lab, WS 2018/2019 - Exercise 1

Submitted by: Amadeus Hovekamp,
amadeus.hovekamp@rwth-aachen.de,
Matr. no: 4603934

The repository with the source code can be found under the following link:
https://github.com/Schokokugel/RoboticsLab

1 Activation Functions

First, let's look at the activation functions. All implemented derivatives seem to be correct. The gradient check function agrees and shows differences between numerical and analytical derivative smaller than 10^{-7}, so there are no assertion errors.

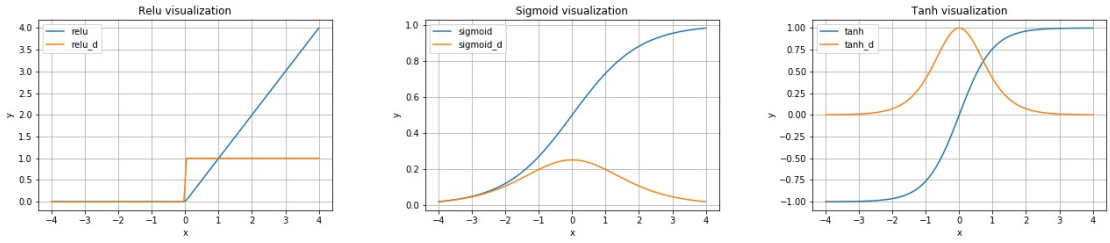

Figure 1: ReLU, Sigmoid and Tanh Activation Functions

2 Network architectures

After implementing all ToDos that needed to build the network and the stochastic gradient descent, I trained a relatively small network with 3 hidden layers containing 40 neurons each. Having heard that ReLU is widely used in neural networks nowadays, I chose to use only ReLU activation functions first.

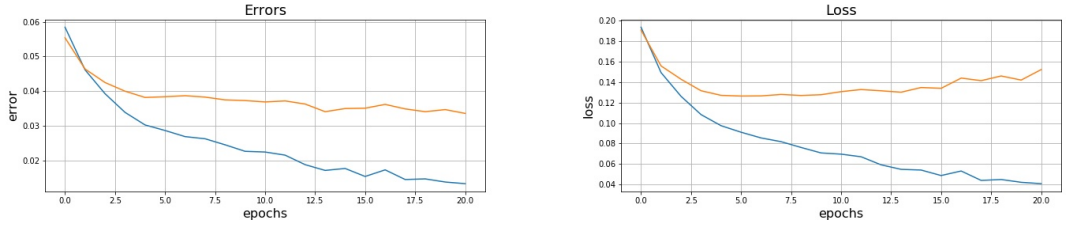

Figure 2: 40 ReLU, 40 ReLU, 40 ReLU

It took some time to train this network with all the 50000 training data points but the performance was kind of good already. The training error goes down fast to 1.1 % and the validation error more slowly to 3.5 %. But while the training loss goes down to 0.04, the validation loss actually increases after epoch 4 and continues to go up until the end of training.

Because it took so long to run the training with all data points, I took a subset of 5000 points to get a feel how the other activation functions perform. While the Tanh activation function was pretty similar to the performance of ReLU, the remaining error of the MLP using the Sigmoid activation

function was higher. This may be ok to reduce overfitting but the Sigmoid network did not seem to learn too much even after two rounds training. For this reason I continued playing arround for the most part with ReLUs.

After using both training sets with 5000 and 10000 data points I noticed a big difference in validation error for small networks. The training error went down faster with fewer data points but the validation error remained large. So I realized that 5000 and most probably 10000 are too few data points to get good results.

Due to that I run 3 networks with more neurons over night, (200, 100), (300, 200) and (300, 100, 100). All had kind of good results but stopped improving after about 35 epochs as the training error went down to 0.

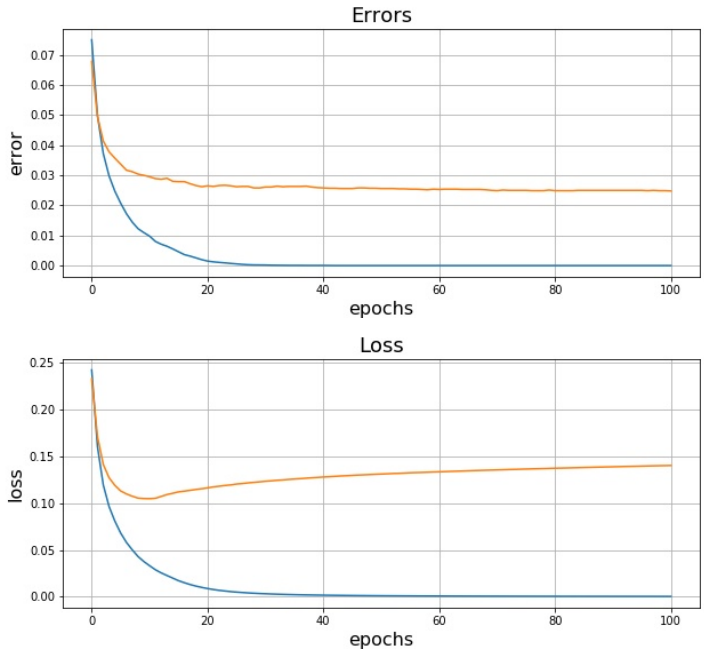

Figure 3: 200 ReLU, 100 ReLU

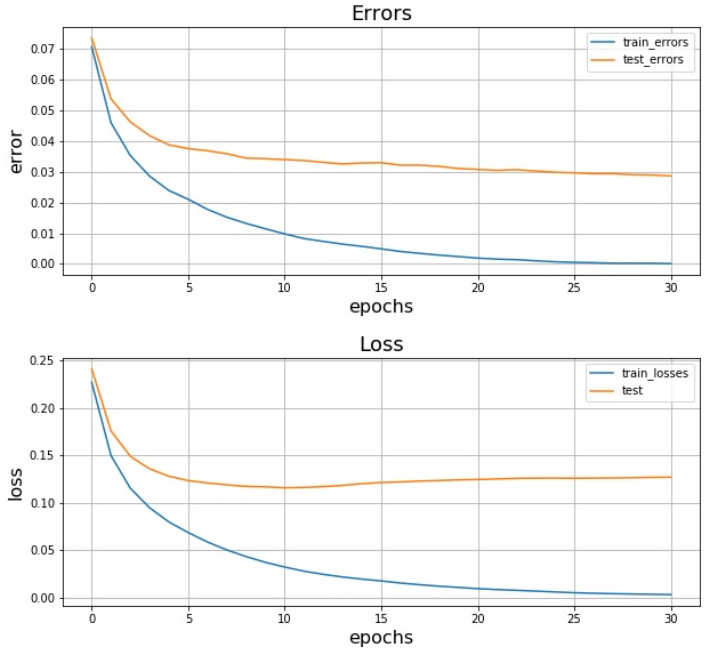

Figure 4: 300 ReLU, 200 ReLU

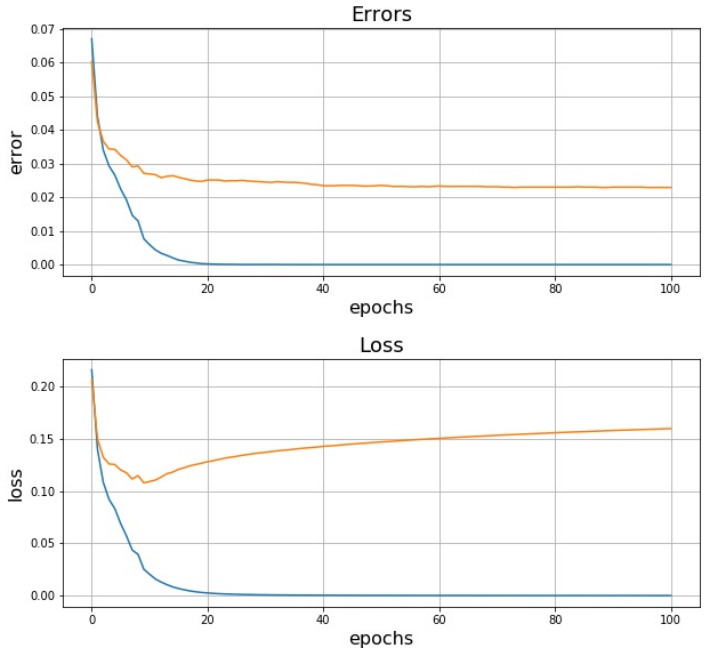

Figure 5: 300 ReLU, 100 ReLU, 100 ReLU

3 Gradient Descent vs. SGD

As Gradient Descent performed way worse than SGD I stopped using it after several runs with different architectures.

4 Evaluation

I trained (300, 200) with ReLUs with 0.08 learning rate, batch size 80 and 30 epochs. It achieved a final test error of 0.0287 which is worse than I expected.