DD2424 Deep Learning in Data Science

Assignment 2

Ramona Häuselmann

April 16, 2021

1 Gradient Computation

I managed to successfully write the functions to correctly compute the gradient analytically. I tested my implementation by computing the gradients using both ComputeGradsNum.m and ComputeGradsNumSlow.m. Then I compared my results with the numerical approaches by calculating the absolute difference of each gradient element as in equations below. Then I checked those against a threshold (1e-5). When using the reduced set with `d=20` and `n=2` and using ComputeGradsNumSlow.m I get a maximum error of

- `diff_W1_max = 3.9846e-11`
- `diff_b1_max = 2.9749e-11`
- `diff_W2_max = 3.3175e-11`
- `diff_b2_max = 3.2699e-11`

Using ComputeGradsNum.m I get

- `diff_W1_max = 1.9299e-07`
- `diff_b1_max = 1.3112e-07`
- `diff_W2_max = 6.4177e-07`
- `diff_b2_max = 5.6976e-07`

These errors are small enough to conclude that my implementation works.

$$diff\_W = abs(ngrad\_W - grad\_W) \tag{1}$$

$$diff\_b = abs(ngrad\_b - grad\_b) \tag{2}$$

After that I trained the network with 100 examples, `lambda=0` for 200 epochs, which resulted in overfitting:

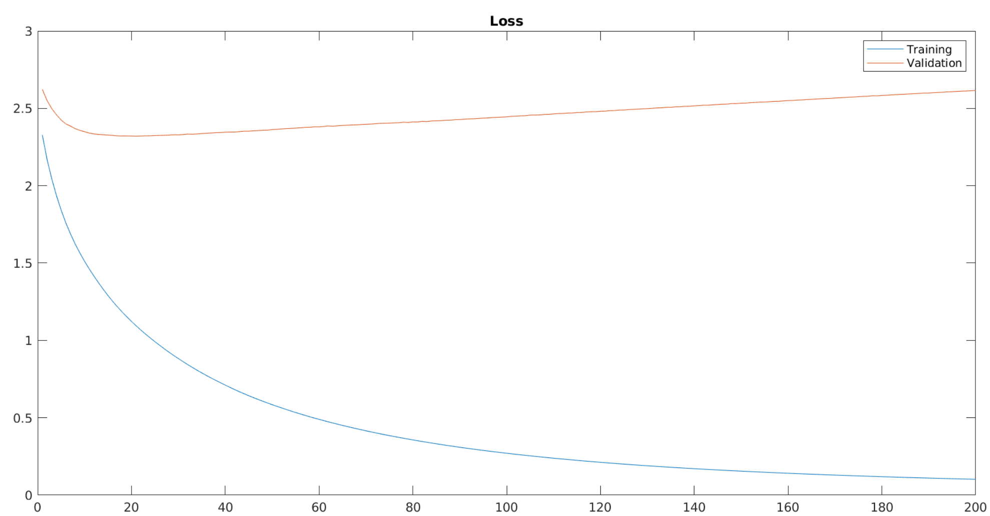

Figure 1: Exercise 2: sanity check, overfitting to training data

2 Exercise 3

In exercise 3 I trained the network with `eta_min = 1e-5`, `eta_max = 1e-1` and `n_s=500`, `lambda = 0.01` and a batch size of 100 for 1 cycle. Then I compared my results with the result given in the assignment instructions. I concluded that my result is similar enough and therefore my implementation should be correct. For the diagrams I recorded 10 values per cycle and I only used data_batch_1.mat for training. After one cycle my network achieves a training accuracy of 60.55%, validation accuracy of 45.62% and a test accuracy of 46.16%.

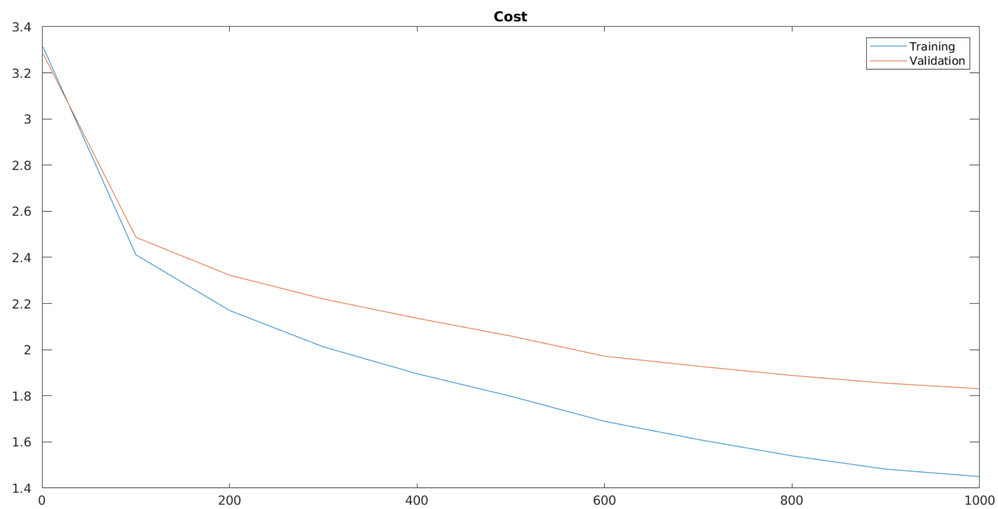

Figure 2: Exercise 3: cost

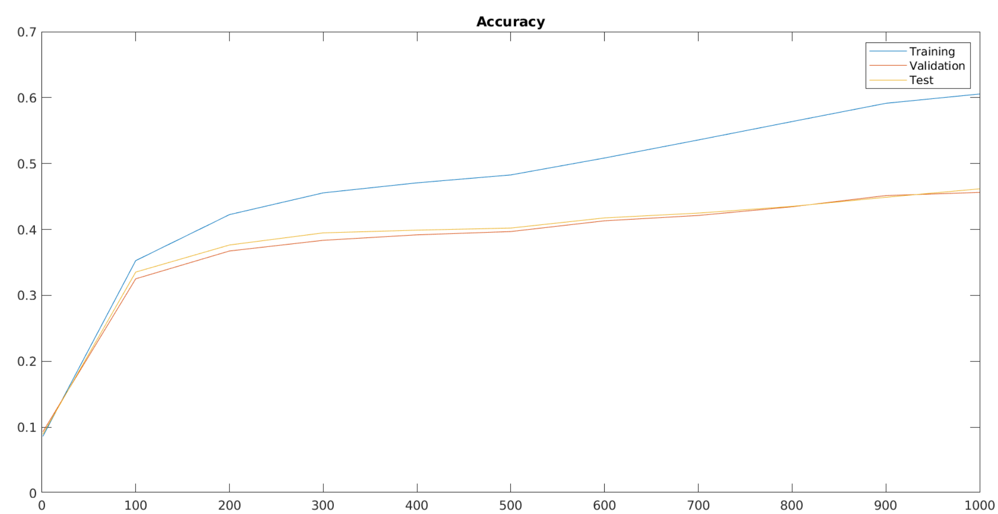

Figure 3: Exercise 3: accuracy

3 Exercise 4

3.1 Sanity check

For another sanity check I first ran the training with `n s=800` for 3 cycles. I got similar results as in the assignment instructions. After 3 cycles my network achieves a training accuracy of 71.68%, validation accuracy of 46.16% and test accuracy of 46.88%.

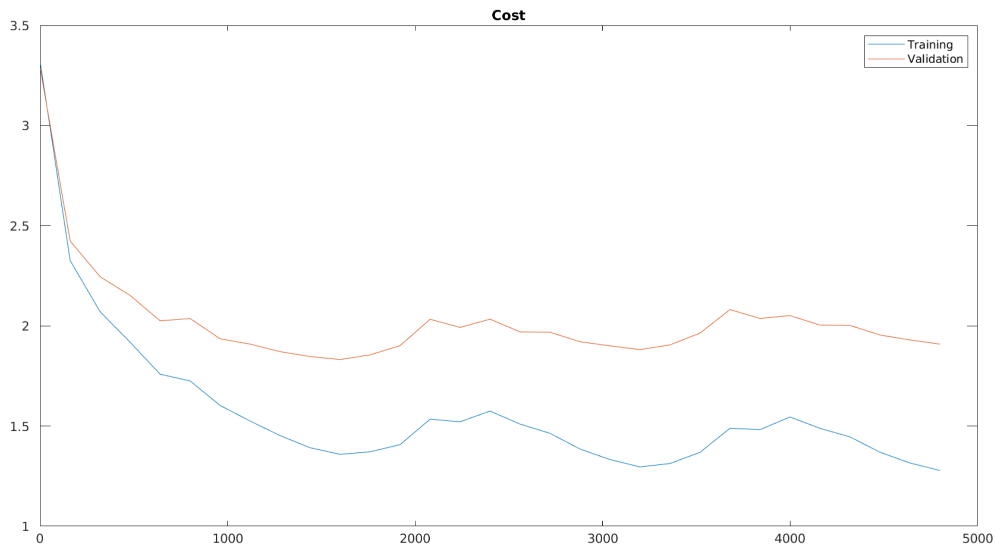

Figure 4: Exercise 4 sanity check: cost

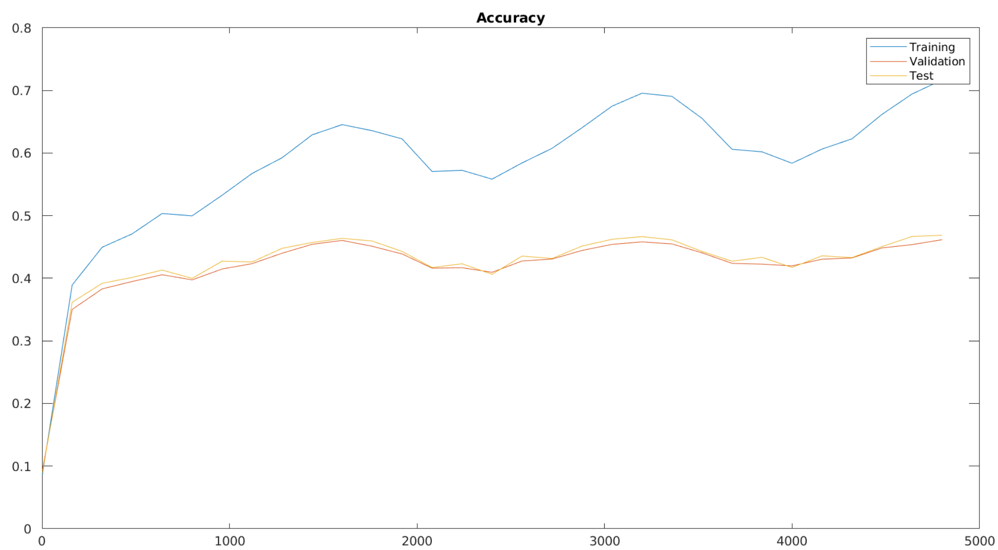

Figure 5: Exercise 4 sanity check: accuracy

3.2 Lambda search

3.2.1 Coarse search (uniform)

For the coarse search I sampled 20 values uniformly in the range `l_min=0.00001` and `l_max=0.10000`. For each lambda I trained for 2 cycles and I used `n_s=900`. I use all available data for training, except for 5000 that I use for validation.

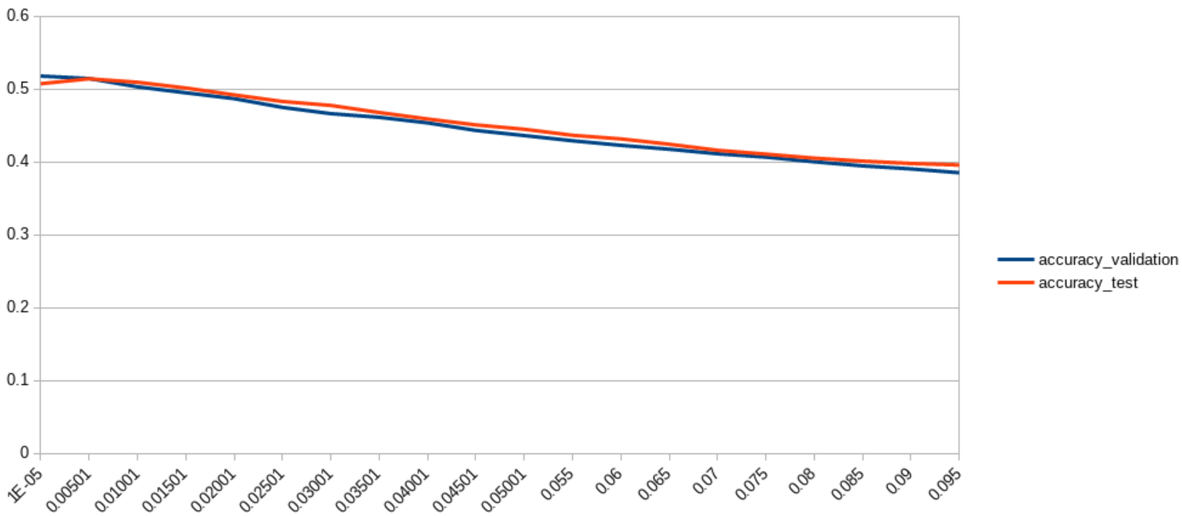

Figure 6: Exercise 4: accuracy for different lambda values, uniform coarse search

The 3 best performing networks based on the validation accuracy were achieved for

(1) `lambda=0.00001` (51.78%)

(2) `lambda=0.00501` (51.44%)

(3) `lambda=0.01001` (50.30%)

From the graph we can see that the best results are achieved in the range `lambda=1e-5` to `lambda=1e-2`. Therefore I used this range for a finer, random search (see next section).

3.2.2 Fine search (random)

For the fine search I sampled 20 values randomly in the range `lambda=1e-5` to `lambda=1e-2`. For each lambda I trained for 5 cycles and I used `n_s=900`. I use all available data for training, except for 5000 that I use for validation.

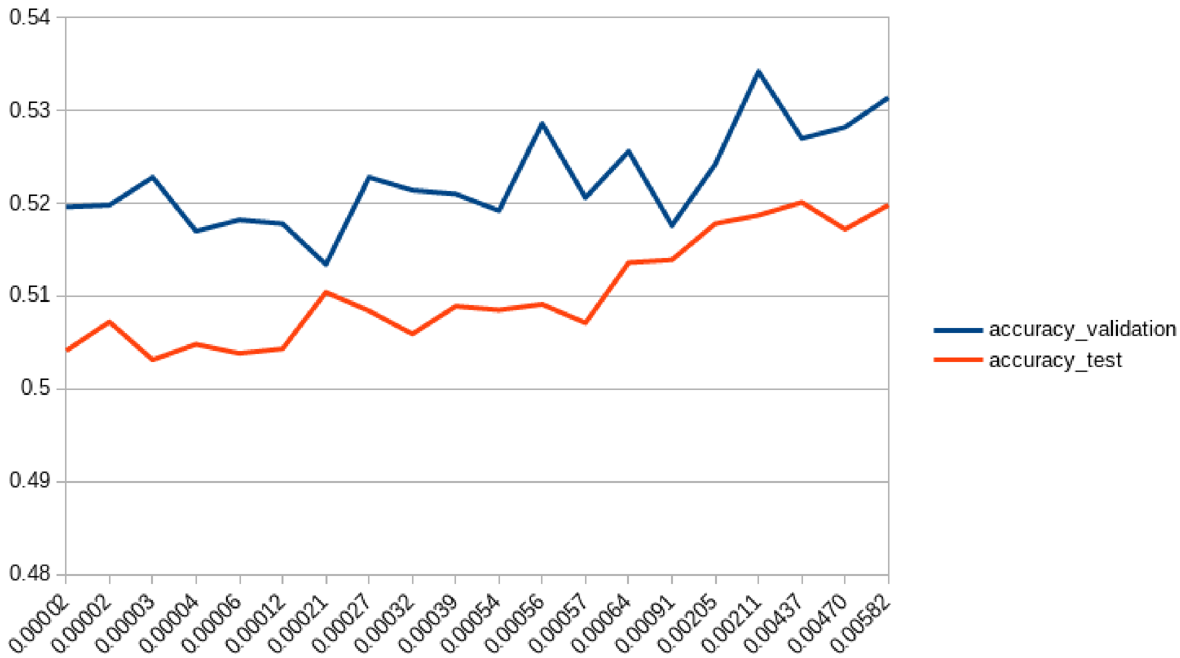

Figure 7: Exercise 4: accuracy for different lambda values, random fine search

The 3 best performing networks based on the validation accuracy were achieved for

(1) `lambda=0.00211` (53.42%)

(2) `lambda=0.00582` (53.14%)

(3) `lambda=0.00056` (52.86%)

As seen in the graph in this experiment I achieved the best results with `lambda = 0.00211`. Therefore I used this value for the training of the network.

3.2.3 Training with best lambda

For the final training I used all available data for training, except 1000 that I use for validation. The lambda search resulted in the best performance when using `lambda = 0.00211`, so this value is used for the final training. I ran the training with 2 different values for n_s and for 3 cycles.

(1) `n_s=900`: test accuracy: 51.54%

(2) `n_s=980`: test accuracy: 51.91%

Below are the resulting diagrams for `n_s=980`, since it gave a slightly better result.

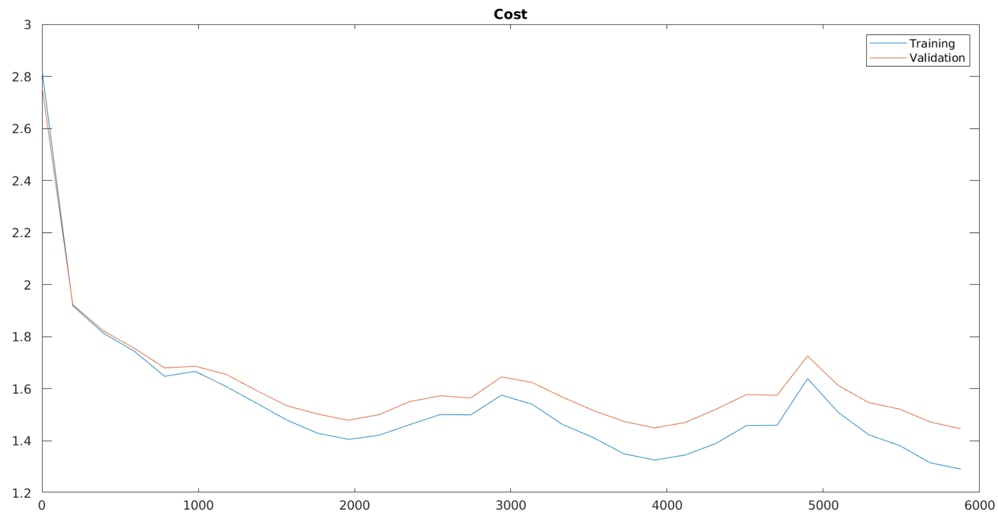

Figure 8: Exercise 4: final training

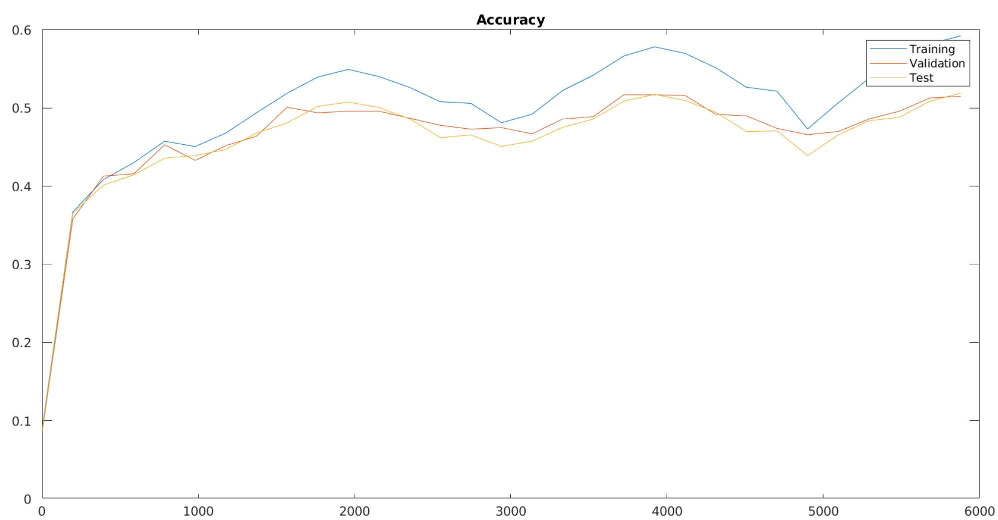

Figure 9: Exercise 4: final training

The final performance is

- training accuracy: 59.22%
- validation accuracy: 51.5%
- test accuracy: 51.91%