Assignment 2

Computational Intelligence SEW, SS2018

Team Members		
Last name	First name	Matriculation Number
Hohensinner	Richard	00273237
Kim	Sathearo	01430376
Kofler	Lorenz	01430321

1 Regression with Neural Networks

1.1 Simple Regression with Neural Networks

a) Learned function

The first three figures show the results by using a different number of neurons in a single hidden layer (including predicted output function, testing and training data).
As we can see in Figure 1, a number of 2 hidden neurons results in an underfitting model. It is definitely not possible to fit the output function.
By using 8 hidden neurons (Figure 2) it is reached a better result but it is still underfitting. With 40 hidden neurons the results are best. It seems that this amount of hidden neurons leads to almost seeming to overfit (e.g. between -1.0 and -0.5).

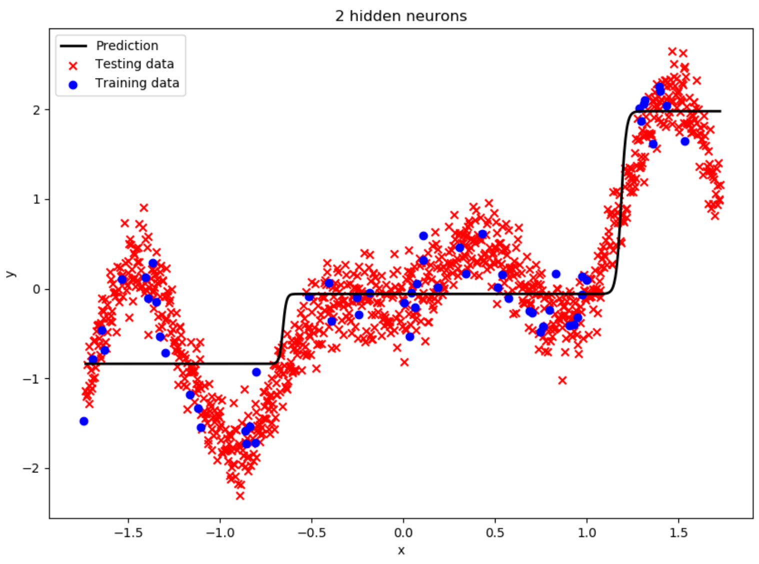


Figure 1: 2 hidden neurons

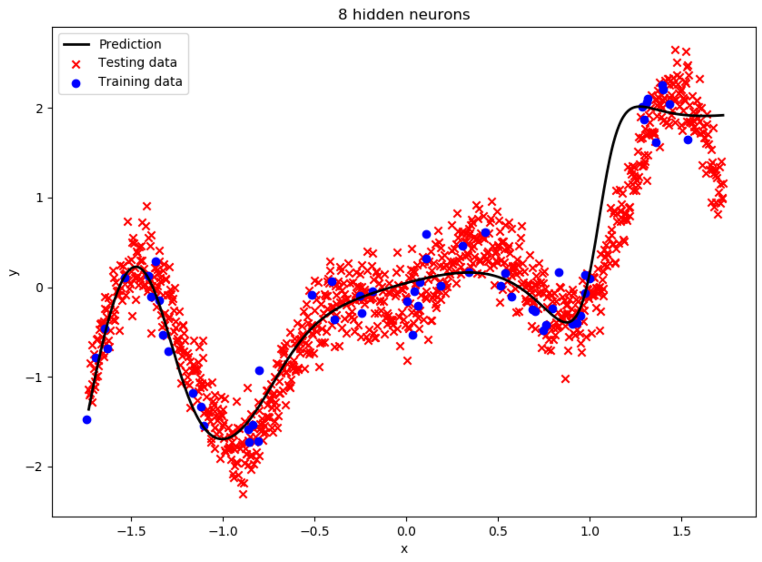


Figure 2: 8 hidden neurons

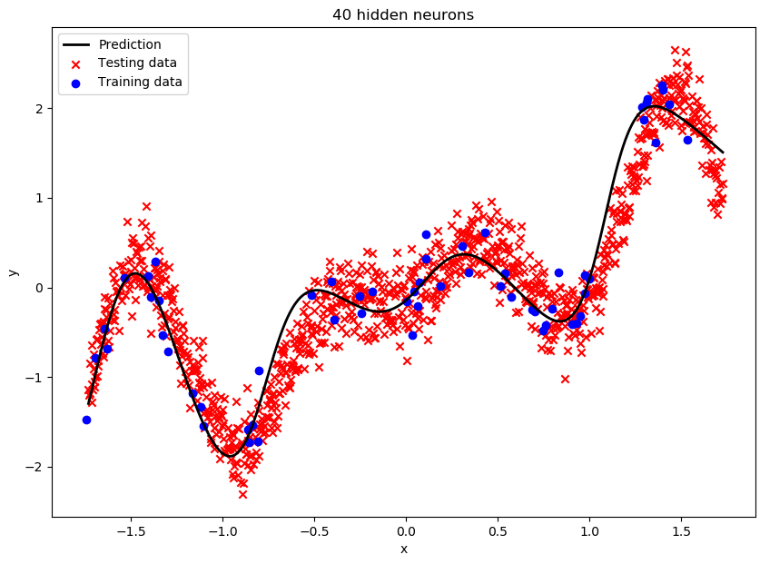


Figure 3: 40 hidden neurons

b) Variability of the performance of deep neural networks

Here are the results of the training set:

- Minimum: 0.04443679
- Maximum: 0.07666421
- Mean: 0.05497159066069205
- Standard Deviation: 0.009648747344143687

Even if they are not asked, here are the results of the test set:

- Minimum: 0.11682291
- Maximum: 0.32886157
- Mean: 0.1956105516802004
- Standard Deviation: 0.06109058959323914

Yes, the seeds are the same at number 3.

Because there is a high difference between the deviations of the errors, it is quite dangerous to use the test set as it is. So there is a validation set needed to be sure. It is best to to use the seed that results in the lowest validation error.

Neural networks use cost functions that are not convex as they have many local minima. In contrast to, linear regression and logistic regression have convex cost functions (convergence is constant).

In Stochastic Gradient Descent the weights are updated after each training point (online learning), so the randomness is introduced. Even if it is replaced by Standard Gradient Descent there is some randomness as the initial weights are random.

c) Varying the number of hidden neurons

As we can see in Figure 4, the best value of n_h is reached at value 4. At this point, the MSE is very low and also the variance is also quite low. A lower number of hidden neurons yield a result that is not as good as the result reached at value 4. Also increasing the number of hidden neurons does not improve the results. As we have discussed within the lecture, increasing the iterations does not yield a better result. Considering the computation time the result it is also not worth to increase the number of iterations. Compared to Figure 6, where 1000 iterations were used, this variant seems to be better. Here the best value of n_h seems to be around 20 as increasing the number of neurons does not give a better result but more effort.

It seems that increasing the number of neurons leads to overfitting.

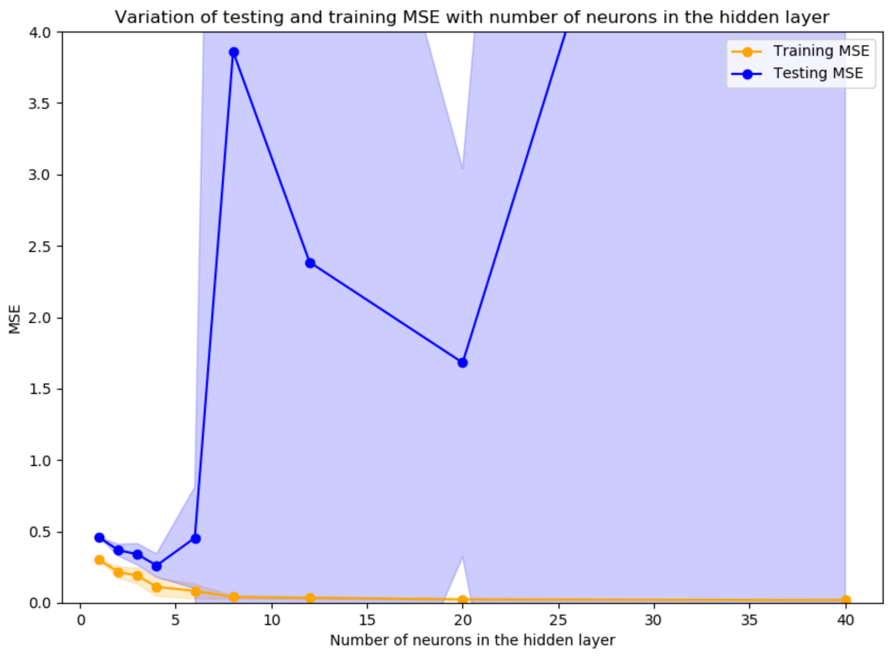


Figure 4: 10000 iterations

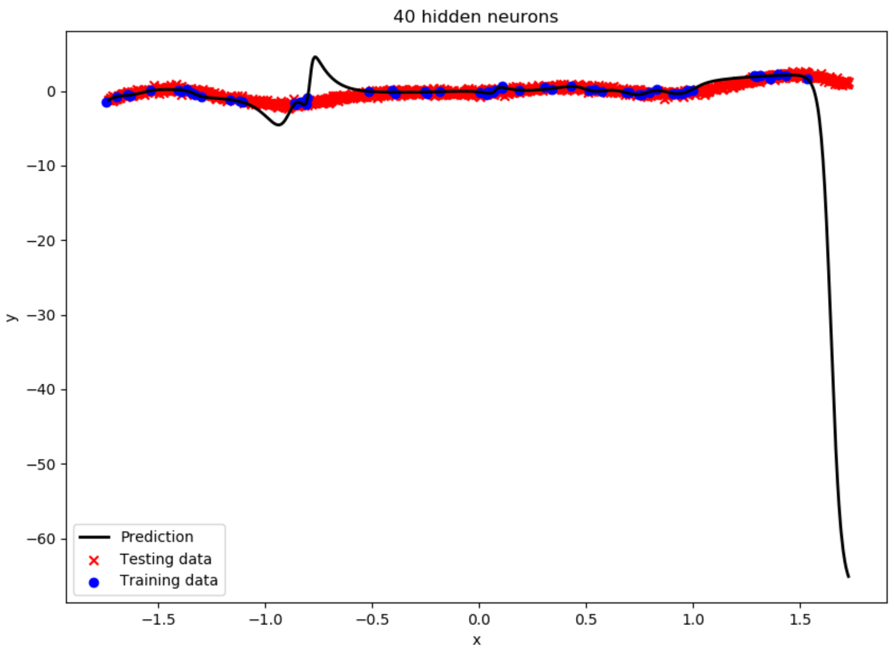


Figure 5: 10000 iterations with 40 hidden neurons

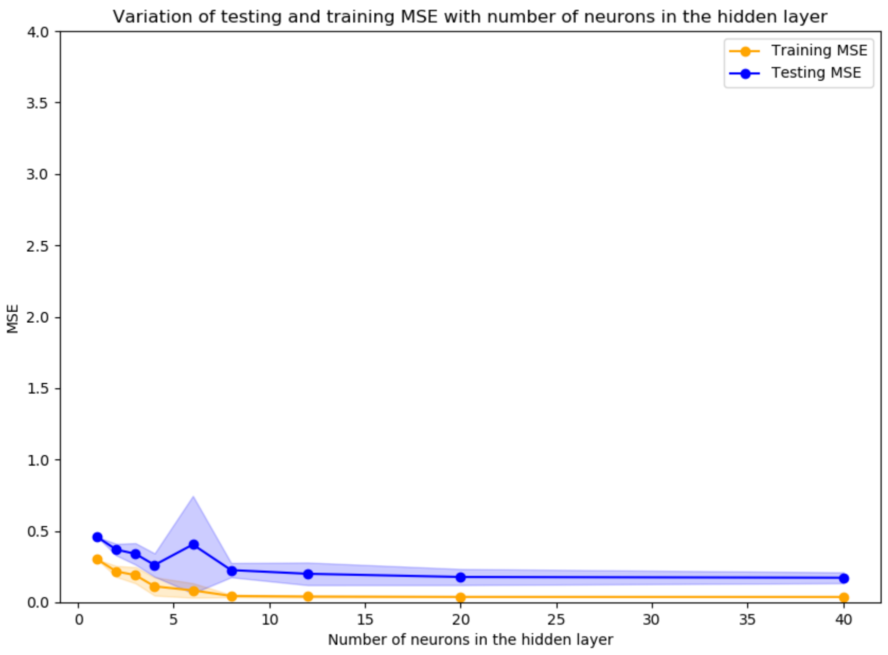


Figure 6: 1000 iterations

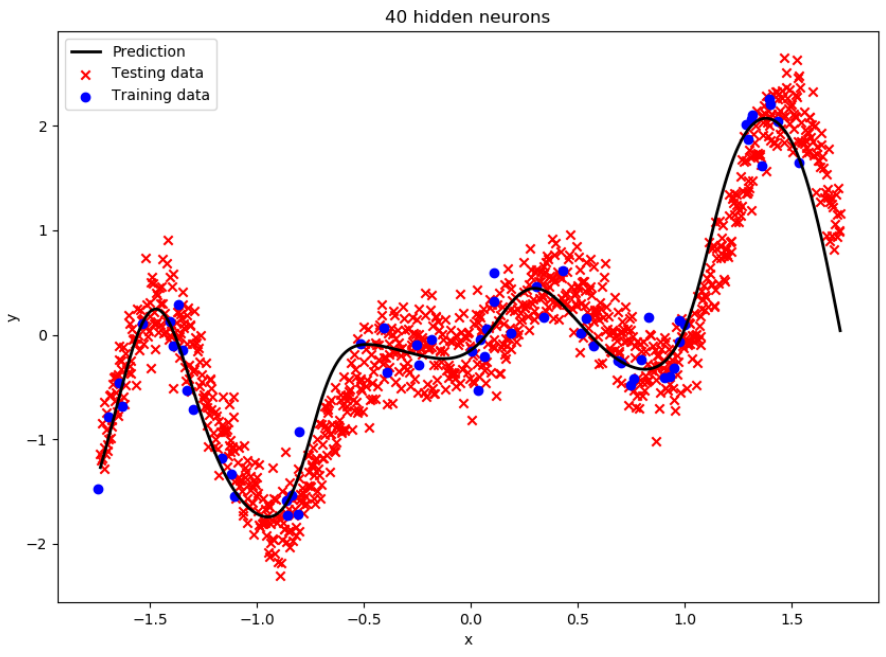


Figure 7: 1000 iterations with 40 hidden neurons

d) Variations of MSE during training

The risk of overfitting seems to increase with the number of hidden neurons. Interestingly, the overfitting exists not from a specific value on as there are some short periods where overfitting occurs and then converges again (e.g. see Figure 9 at around 1500 iterations and 8 hidden neurons).

Regarding the best solver lbfgs seems to do best (on average lowest MSE values). Above all sgd performs not as good as the other two methods.

Stochastic Gradient Descent overcomes overfitting because of the noise in the gradient. It helps to jump to probably better, new local minima.

It seems that lbfgs is better with an increasing number of neurons. It does not only use gradients (second order method). Furthermore, it converges faster.

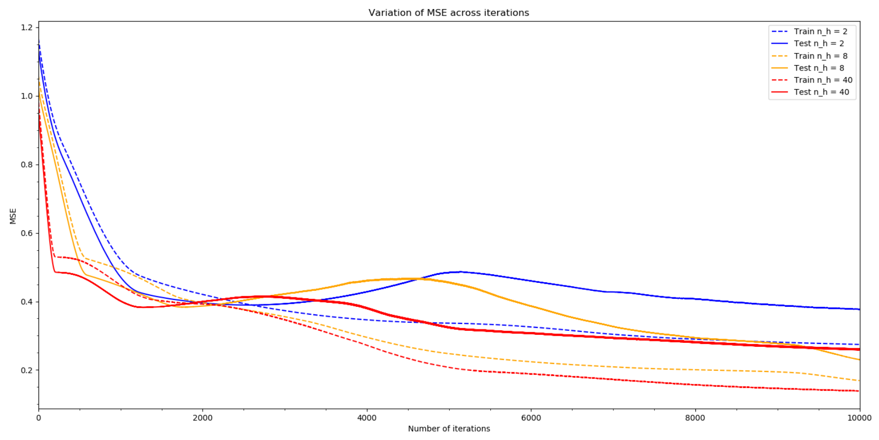


Figure 8: adam

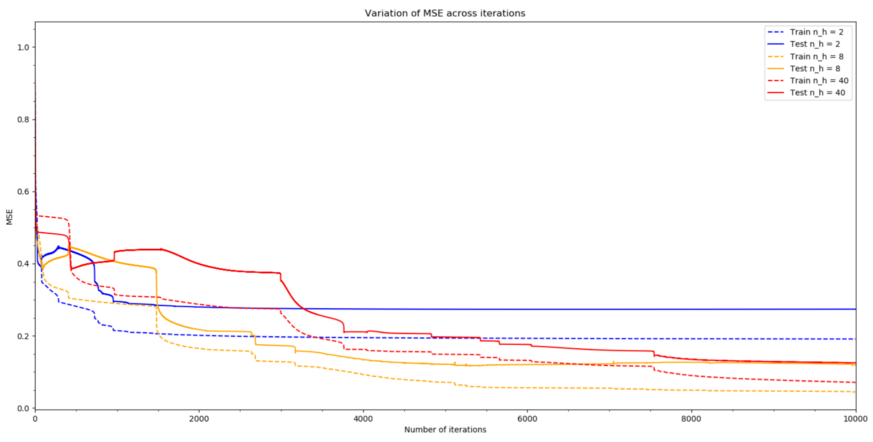


Figure 9: lbfgs

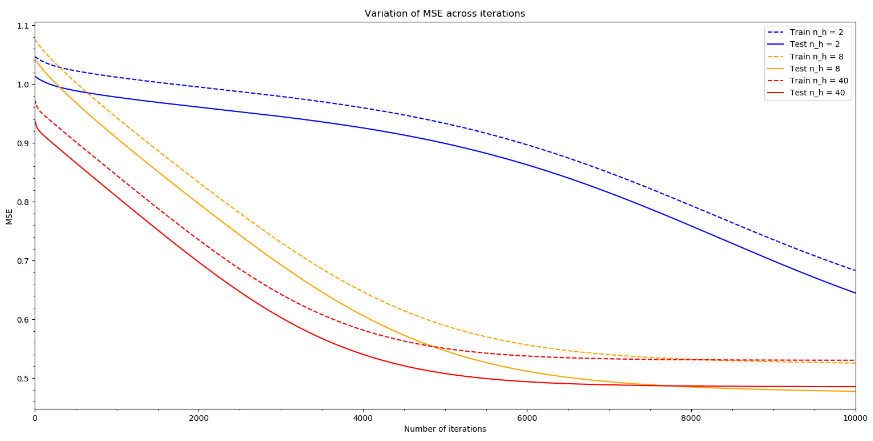


Figure 10: sgd

1.2 Regularized Neural Networks

a) Weight Decay

Figure 11 shows the dependency of the regularization parameter α which is best at a value of 10^{-3}. It helps to overcome overfitting. Its main usage is to simplify the neural network and to keep the weights as low as possible.

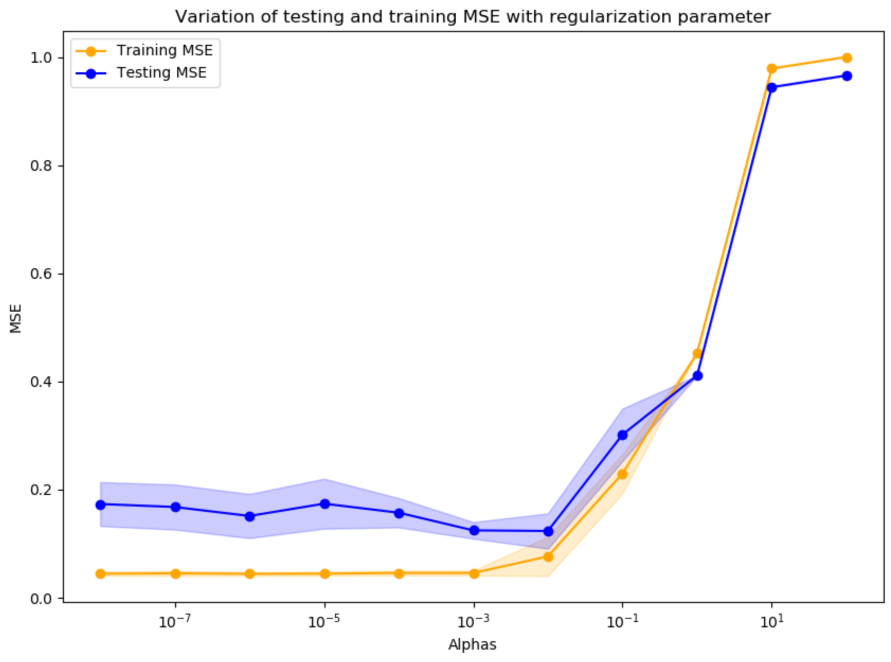


Figure 11

b) Early Stopping

The initial values are always different because of the random seeds, so the iteration numbers will vary for random seeds. This assumption seems to be true in our case.

The standard form of early stopping is much faster. On the one hand, the problem of the used method is that the whole data must be explored so the efficiency suffers from that. On the other hand, it can overcome local minima.

As we can see in Figures 12 to 14, where some examples are plotted, the efficiency of early stopping is (almost) ideal.

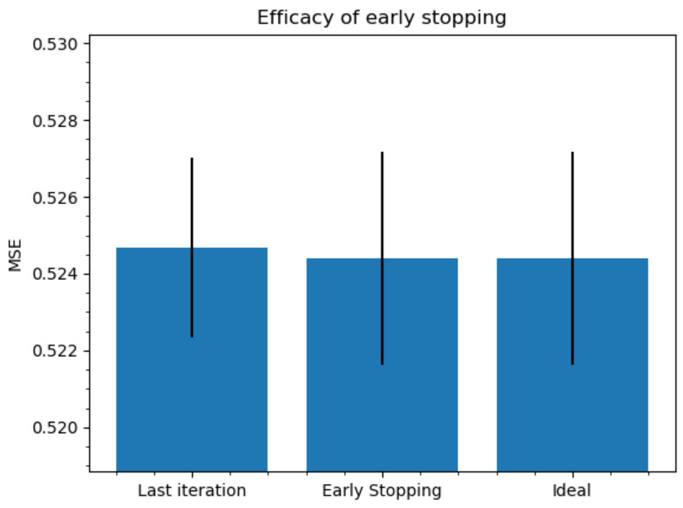


Figure 12

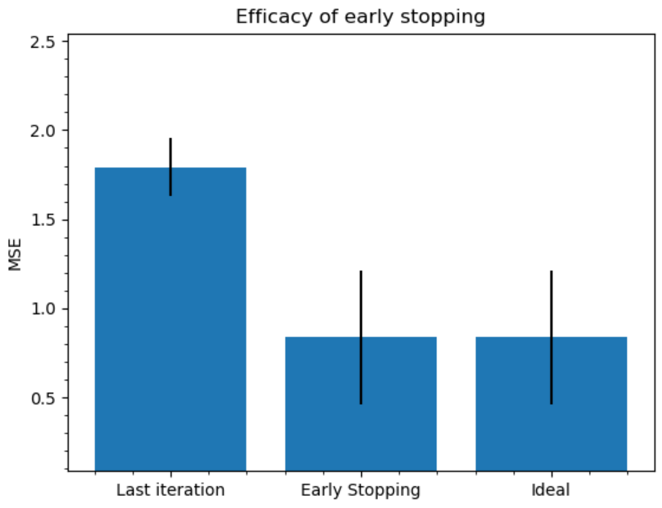


Figure 13

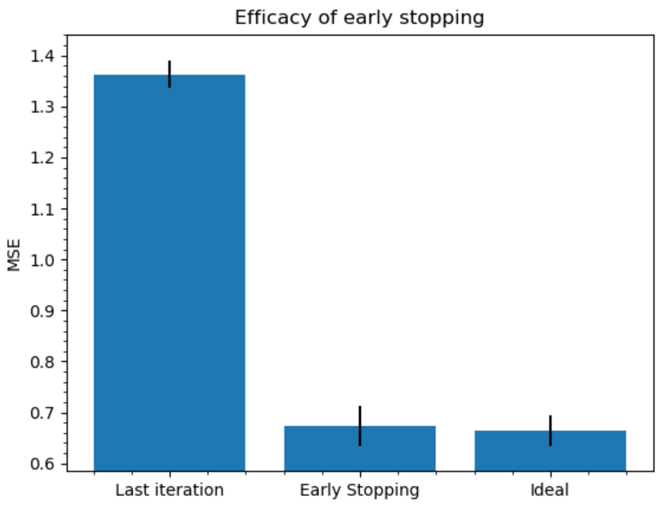


Figure 14

c) Combining the tricks

-

2 Face Recognition with Neural Networks

2.1 Pose Recognition

As we can see in Figure 15, straight and up leads to misclassifications. On the other side, it works best for poses left and right.

```
[[118   5   5  13]
 [  4 135   0   2]
 [  0   0 136   2]
 [  8   4   1 131]]
```

Figure 15: confusion matrix with rows (top to down) and columns (left to right) straight, left, right and up

Considering Figure 16, it seems that the regions within the face are the interesting ones (above all the regions around the nose, forehead, mouth with chops and the neck are white coloured and so they get more weight).

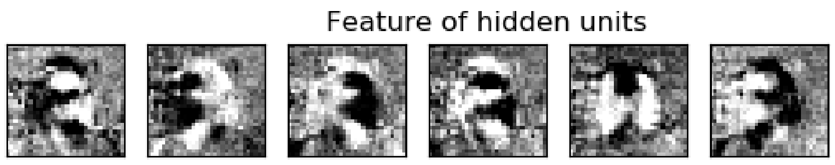


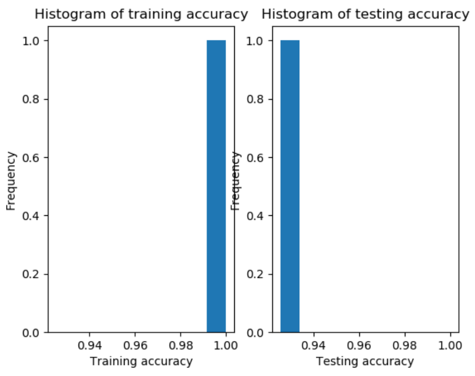
Figure 16: hidden weights

2.2 Face Recognition

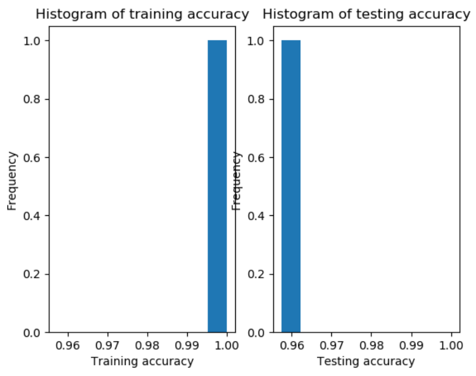
Random initial weights lead to different classifcations as features get weighted differently. With different initial weights also the accuracy changes as can be seen in the next ten pictures after Figure 17. The testing accuracy ranges between 0.92 and 0.96. The training accuracy is always at 1.0.

```
[[27  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 29  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  0 27  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 24  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 28  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  8 19  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 28  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 28  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 23  0  0  0  2  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0 28]]
```

Figure 17: confusion matrix

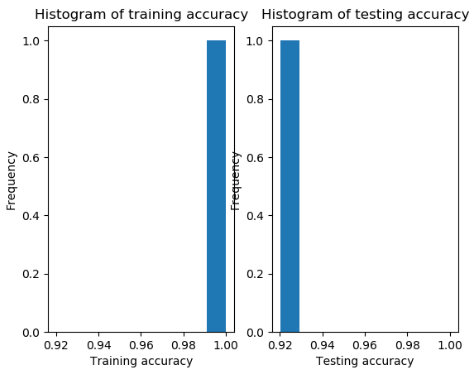


(a)

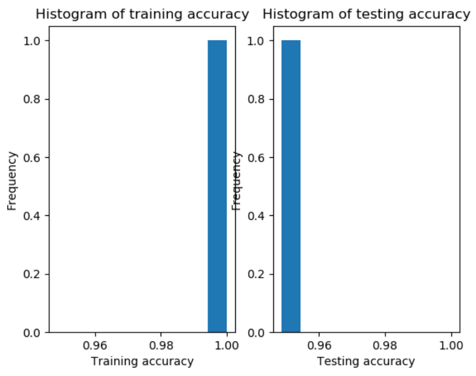


(b)

Figure 18

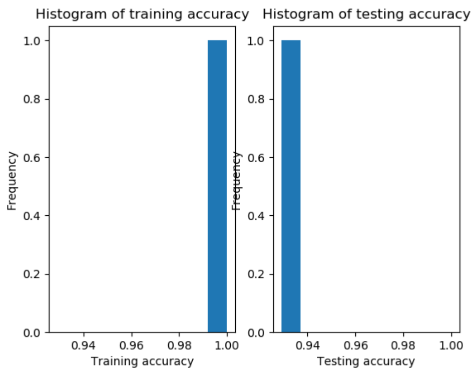


(a)

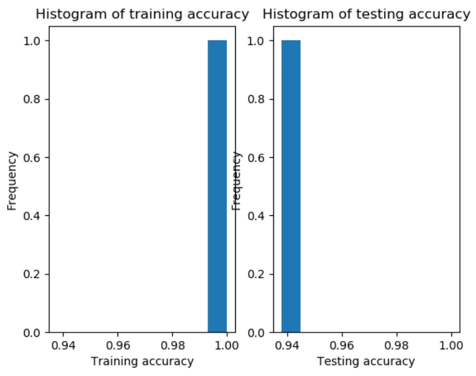


(b)

Figure 19

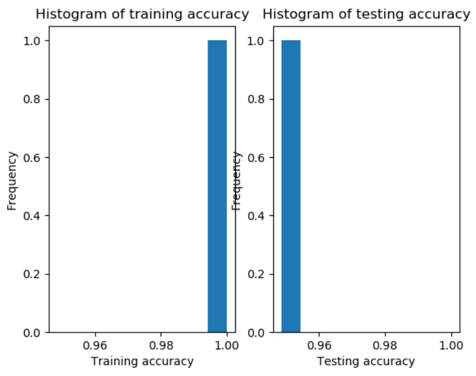


(a)

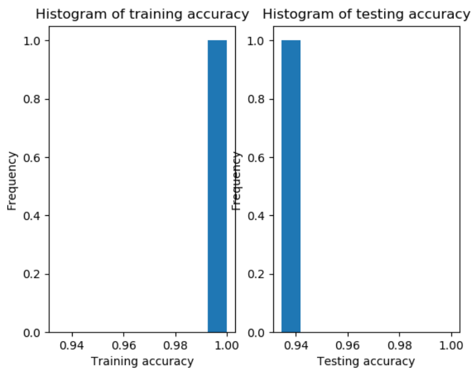


(b)

Figure 20

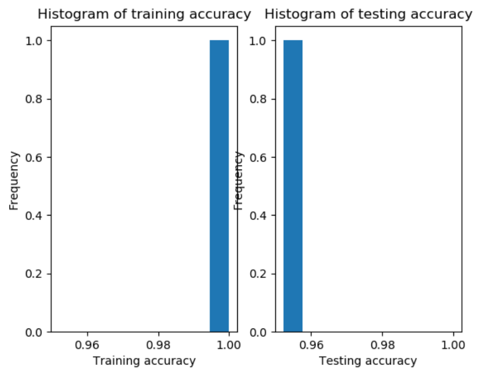


(a)

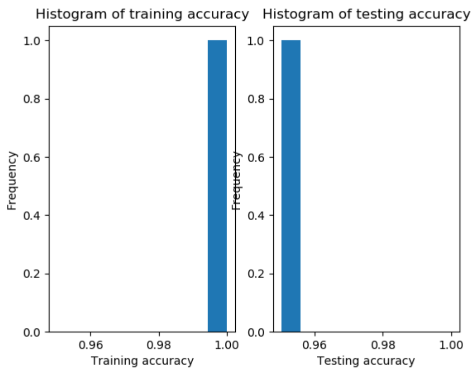


(b)

Figure 21



(a)



(b)

Figure 22

If we look at the misclassified images, we can see, that the people are looking straight forward, sometimes even up to the ceiling or to the right (from their view). Also the sunglasses might be a reason for misclassification.

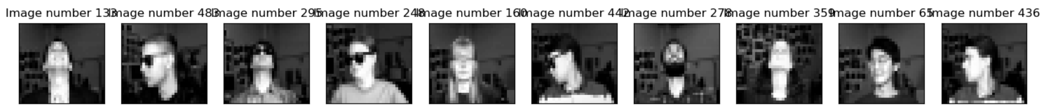


Figure 23