Assignment 2

Computational Intelligence SEW, SS2017

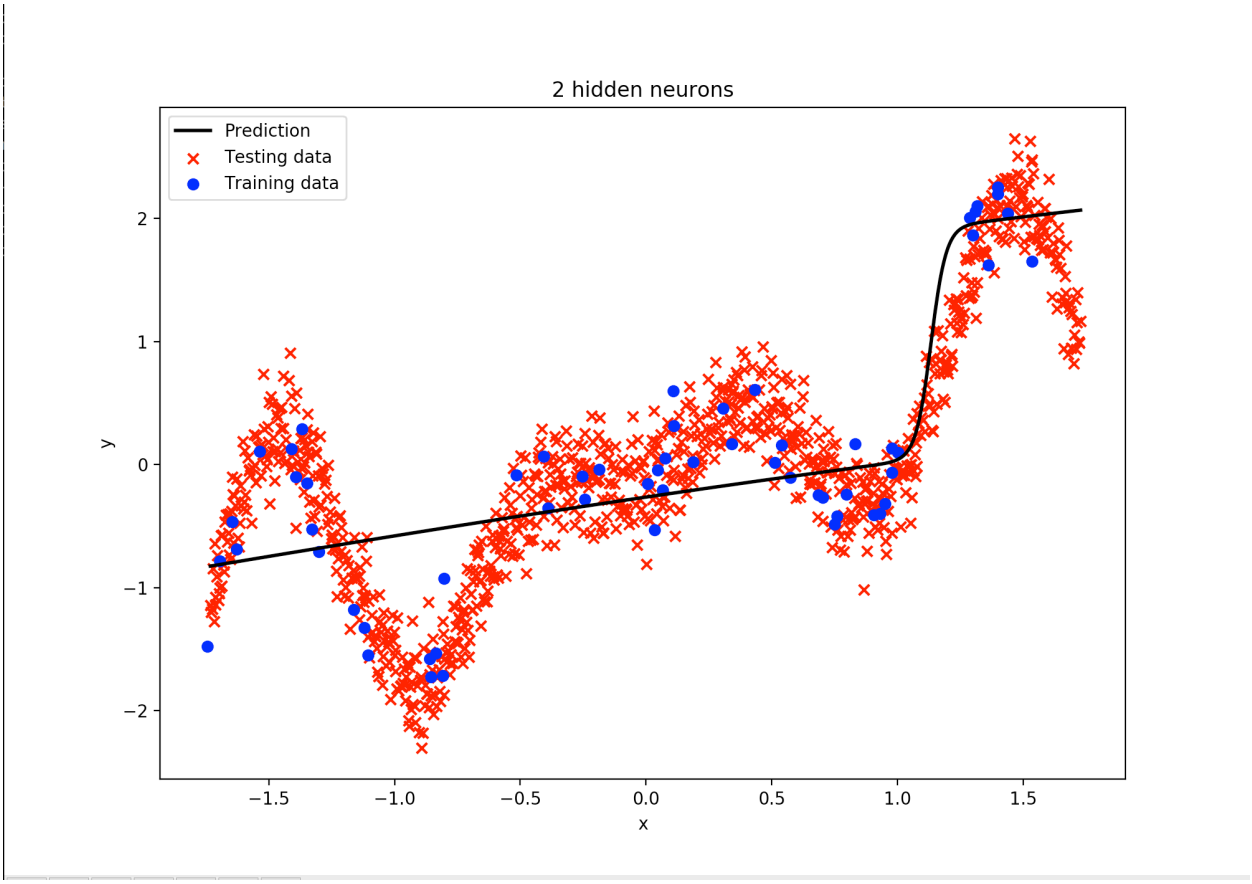
Team Members		
Last name	First name	Matriculation Number
Papst	Stefan	1430868
Guggi	Simon	1430534
Perkonigg	Michelle	1430153

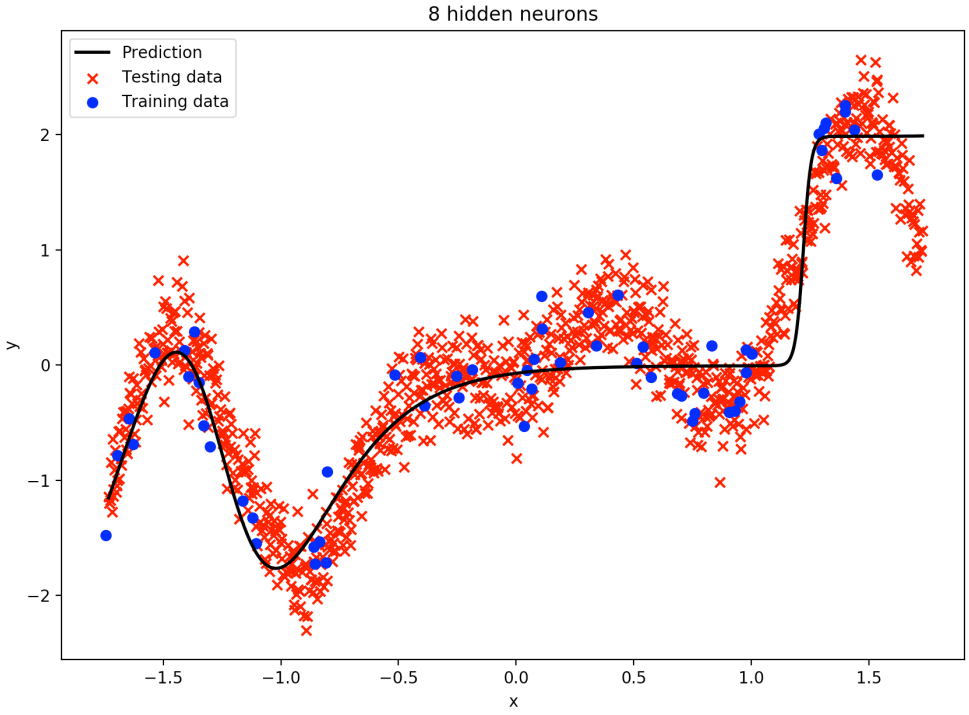
1.1 Simple Regression with Neural Networks

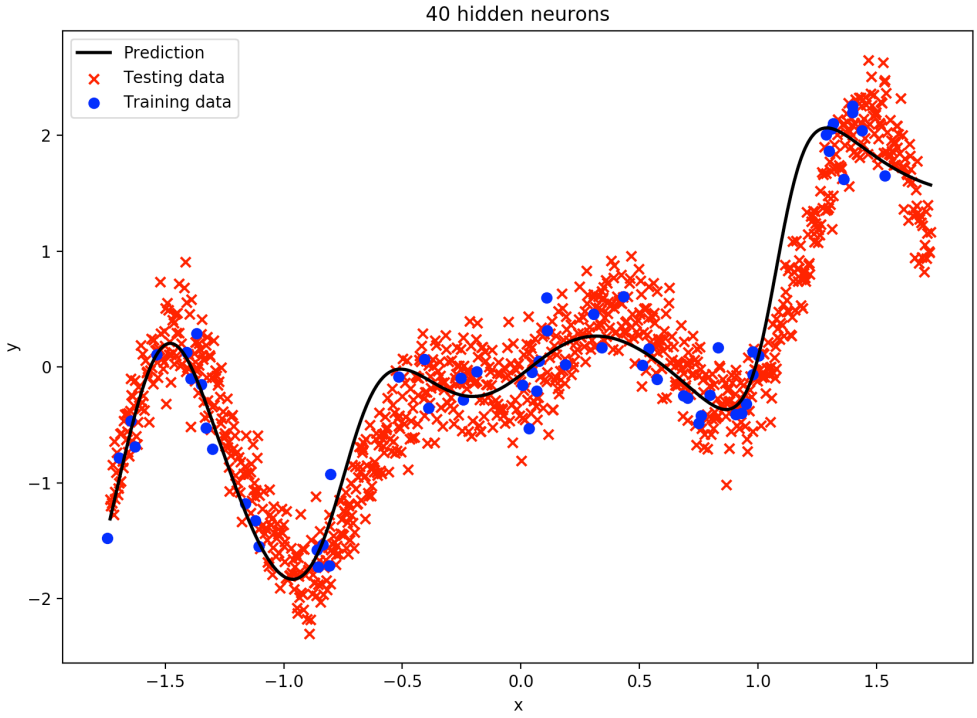
. a) Learned function

. $n_h = 2$, $n_h = 8$ and $n_h = 40$

As you can see in the plot with 2 hidden neurons that under fitting is an existing problem in neural networks. The training set is fitted with a too "uncomplex" function, which results in high errors in the training and testing data. 8 hidden neurons are a better choice, but 40 one fitting the training data as well as the testing data. Overfitting could also happen, for example there are too much hidden neurons, which fit the training set very well, but the testing one not.




8 hidden neurons
Prediction
Testing data
Training data
y
x
2
1
0
−1
−2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


40 hidden neurons
Prediction
Testing data
Training data
y
x
2
1
0
−1
−2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

. b) Variability of the performance of deep neural networks

Here are the minimum, maximum, mean and standard deviation of the mean square error we obtained on the training set for a sample run:

min_train: 0.0517864946589

max_train: 0.102915924071

mean_train: 0.0714639291011

std_train: 0.0183697649393

Is the min MSE obtained for the same seed on the training and on the testing set ?

No, it is a different one.

Explain why you would need a validation set to choose the best seed ?

With the validation data you usually try to find the best performing approach after training the neural network with the training data. In our case this would be the best seed.

Unlike with linear-regression and logistic regression, even if the algorithm converged the variability of the MSE across seeds is expected. Why ?

The seed sets the initial weight values of the neural network. Although the algorithm converges and there is almost the same overall result, the individual weights may differ from each other, because the weights are updated according to their values. This gives us a variability of the MSE for different seeds.

What is the source of randomness introduced by Stochastic Gradient Descent (SGD) ?

It is a property which was introduced by SDG to escape local minima to find better minima.

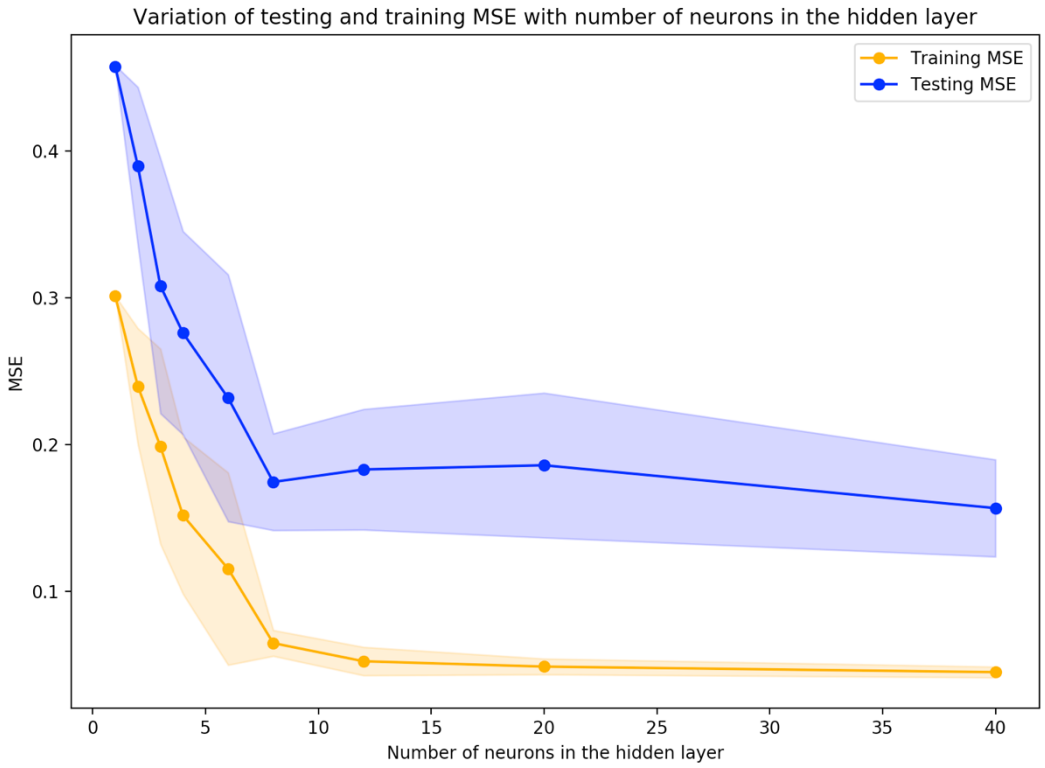
What source of randomness will persist if SGD is replaced by standard Gradient Descent ?

The way how the weights are initialized.

c) Varying the number of hidden neurons:

What is the best value of n_h independently of the choice of the random seed ?

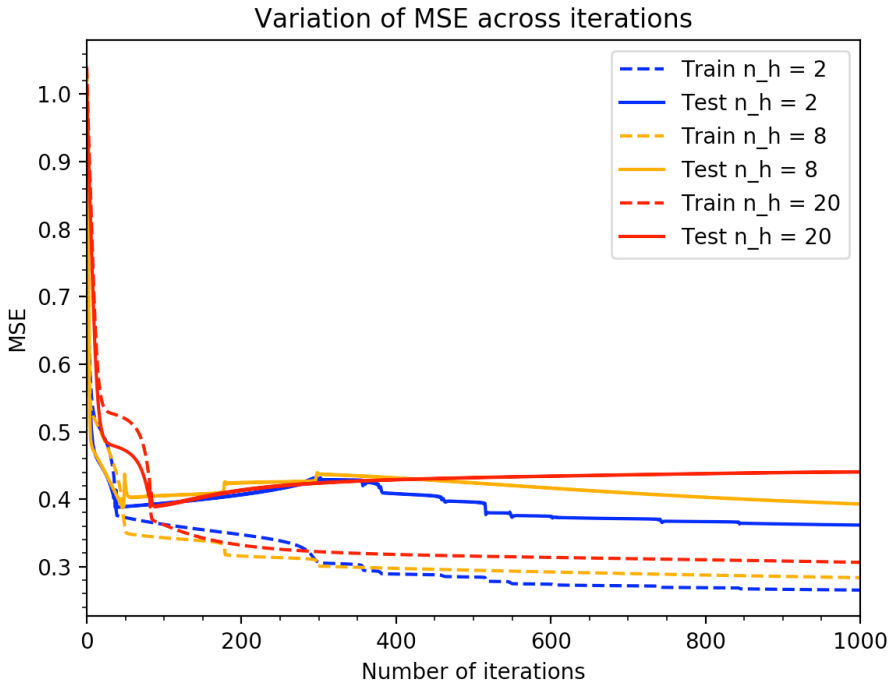
8



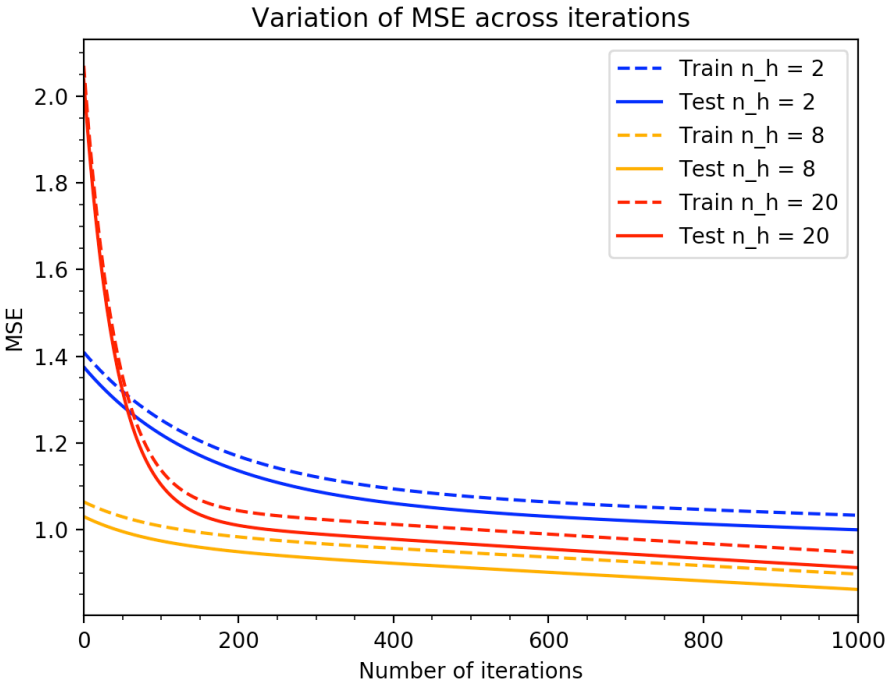
Obviously 8 is the best value for neurons in the hidden layer, so everything above is called over fitting. The training MSE is getting lower, but the training MSE doesn't. With 8+ neurons the neural network is over fitting the training set, which is a drawback for further using with testing sets. In the other hand values under 8 would cause in under fitting. This does mean that the neural network is not trained complex enough to fit either the training set nor the testing set very well.

d) Variations of MSE during training:

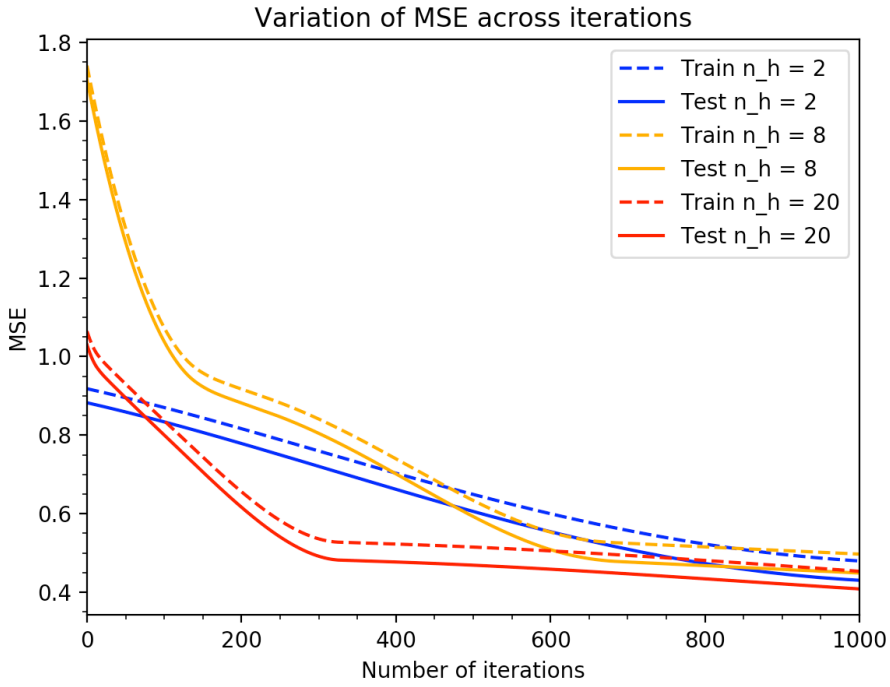
solver: "lbfgs"


Variation of MSE across iterations
Train n_h = 2
Test n_h = 2
Train n_h = 8
Test n_h = 8
Train n_h = 20
Test n_h = 20
MSE
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000
Number of iterations

solver: “sgd”


Variation of MSE across iterations
Train n_h = 2
Test n_h = 2
Train n_h = 8
Test n_h = 8
Train n_h = 20
Test n_h = 20
MSE
2.0
1.8
1.6
1.4
1.2
1.0
0
200
400
600
800
1000
Number of iterations

solver: “adam”



Is the risk of overfitting increasing or decreasing with the number of hidden neurons ?

Increasing.

adam’ is a variant of ‘sgd’ and both are first order methods (the parameter updates are based on the gradient only), whereas ‘lbfgs’ is a second order method (the updates are also based on the Hessian). Which methods seem to perform best in this problem?

The first order methods seem to be more stable, but the second order method getting faster to a lower MSE. For this problem, we would say that the “adam” solver fits the requirements the best.

What feature of stochastic gradient descent helps to overcome over fitting?

Early-stopping.

The neural network is rather small as compared to what is used is real-life

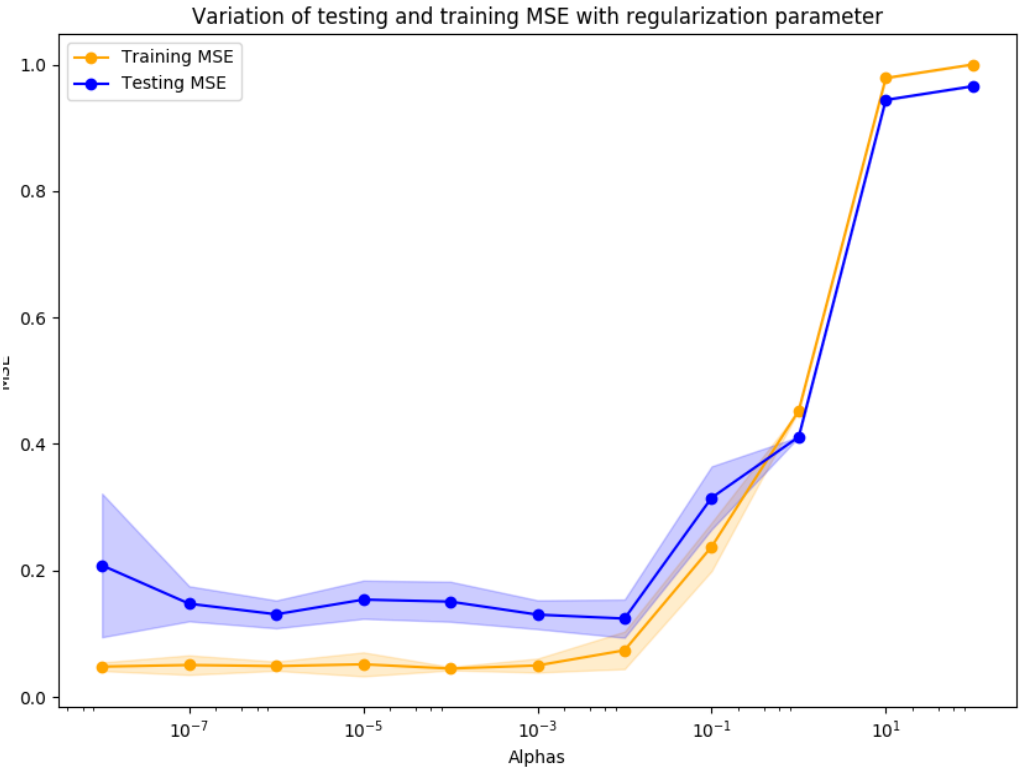
problems, according to your analysis which solver will be more appropriate when the number of neurons increases?

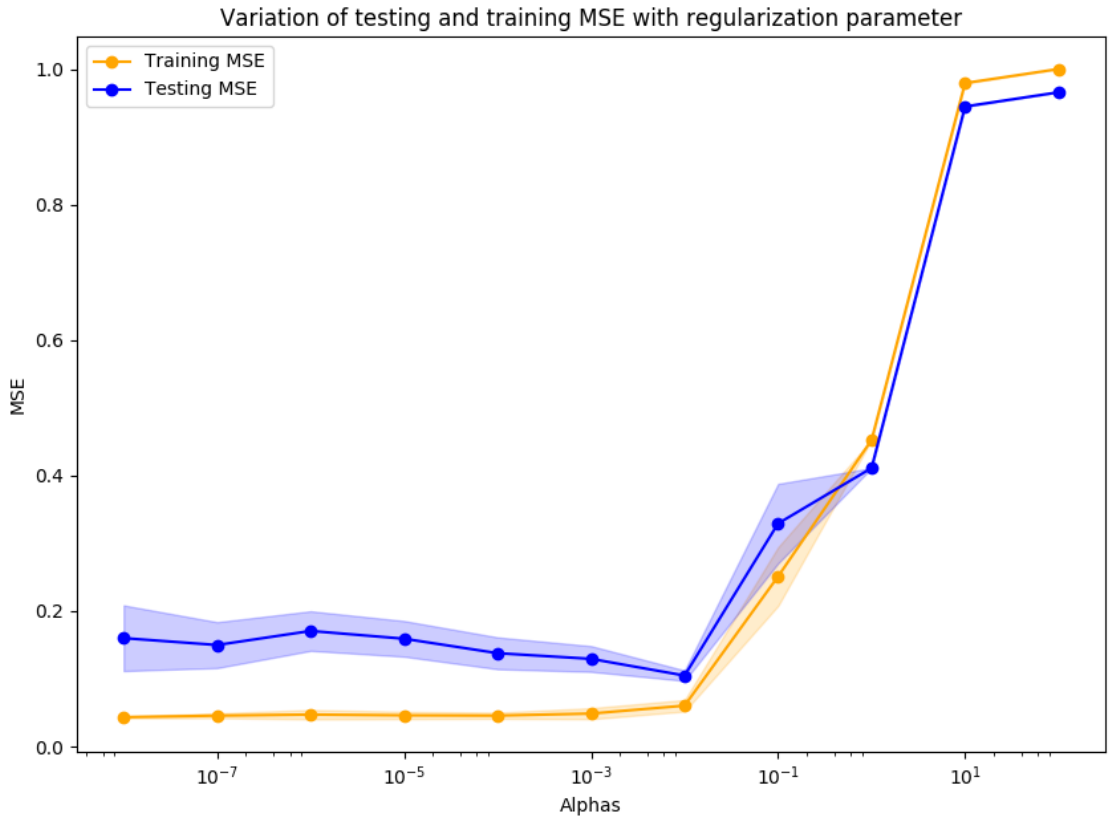
"Adam" would be the best choice, because it is a first order method and do not to calculate the hessian, which lasts very long for big networks.

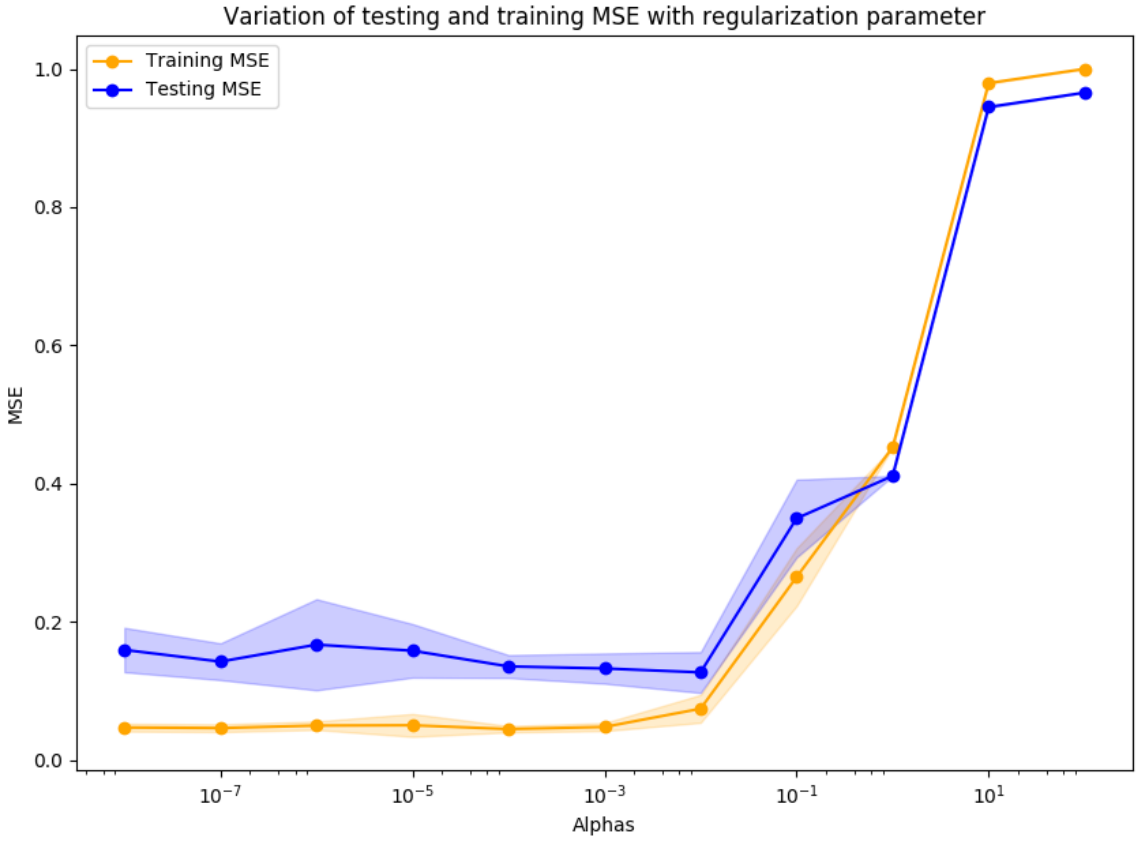
1.2 Regularized Neural Networks

a) Weight Decay:

Include plots of the variation of MSE of the training and test set with the value of α:




Variation of testing and training MSE with regularization parameter
Training MSE
Testing MSE
MSE
Alphas
0.0
0.2
0.4
0.6
0.8
1.0
10^{-7}
10^{-5}
10^{-3}
10^{-1}
10^{1}


Variation of testing and training MSE with regularization parameter
Training MSE
Testing MSE
MSE
Alphas
0.0
0.2
0.4
0.6
0.8
1.0
10^{-7}
10^{-5}
10^{-3}
10^{-1}
10^{1}

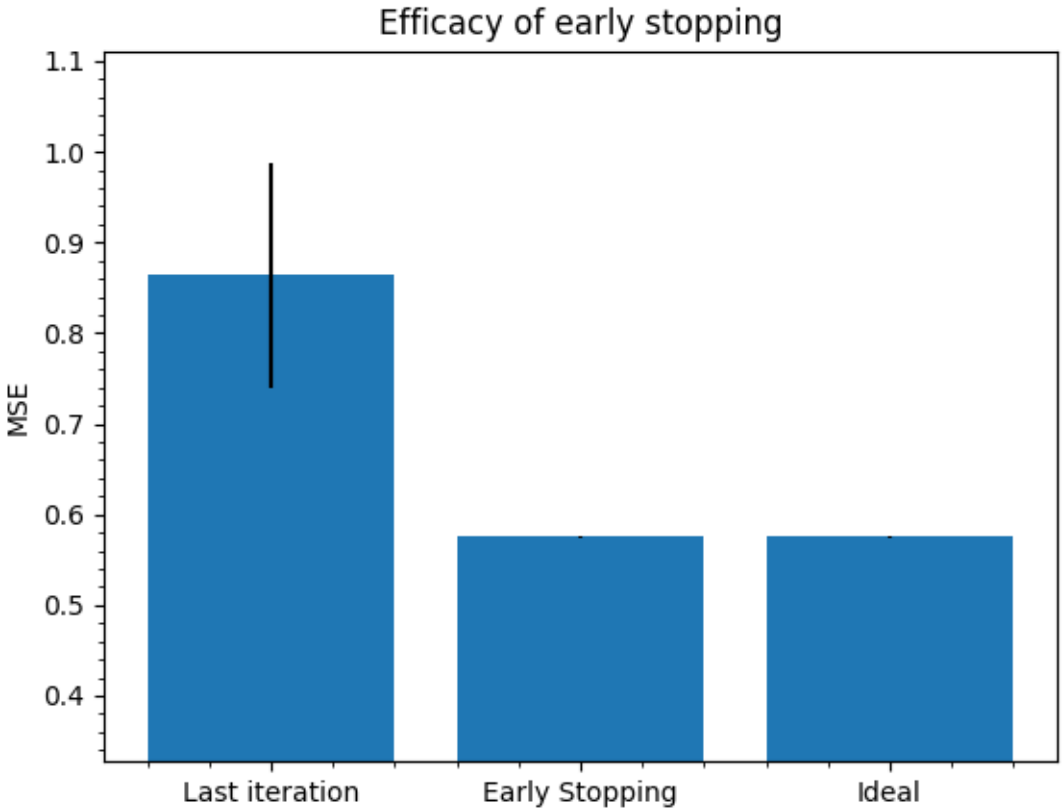
What is the best value of α?

The best value of α as seen in the plots above is 10^-3.

Is regularization used to overcome overfitting or underfitting? Why ?

Regularization is used to reduce overfitting through adding a penalty for complexity to the loss function.

b) Early Stopping:



Include the bar plots to compare the errors on the test sets at the last training iterations, at early stopping and when it is minimal.

In the light of question 1.1.b) is it expected that early stopping happens (validation error is minimized) at the same iteration number for all random seeds? Is it coherent with your results?

No, it should not happen on the same iteration on random seeds, because it compares the MSE, and on different seeds the number of iteration will differ. This is also coherent with our results.

Early stopping in its standard form is a little different, instead of stopping when the validation error is minimized, one stops training as soon as the validation error increases. What are the pros and cons of those standard form of early stopping and the one you implemented?

Stopping when the error is minimized results in a better fitted neuronal network, but the time to get it is longer as with the real early stopping where the result is not as perfect, but faster. The disadvantage is if the validation error increases just for a short bit and decreases later on, we can't detect it, because we stopped to early.

c) Combining the tricks:

Explain your choice of number of hidden neurons, regularization parameter and solver. Then describe in a short paragraph but rigorously the protocol followed to identify the optimal random seed (mention all the parameter you chose such as).

We choose our number of hidden neurons to be as optimal as possible, because too few neurons are a cause for underfitting, to many are a cause for overfitting. We used early stopping to prevent overfitting as means of regularization and we used the "lbfgs" solver because it gets faster to a lower MSE.

To identify the optimal seed, we calculated the minimal validation error for every seed. Then we checked which seed had the smallest minimal validation error and with this information we have the optimal random seed with the corresponding regressor.

Report the mean and standard deviation of your training, validation and testing error. Report the training, validation and testing error of your optimal random seed.

	Mean Derivation	*Standard Derivation*	*Error of optimal random seed*
Training set	**0.375514996046**	**0.135434513306**	**0.235962582789**
Validation set	**0.493127011783**	**0.0127649652861**	**0.455896404649**
Testing set	**0.453484328187**	**0.0441102316522**	**0.411022684077**

2.1 Pose Recognition

Include the confusion matrix you obtain and discuss. Are there any poses which can be better separated than others?

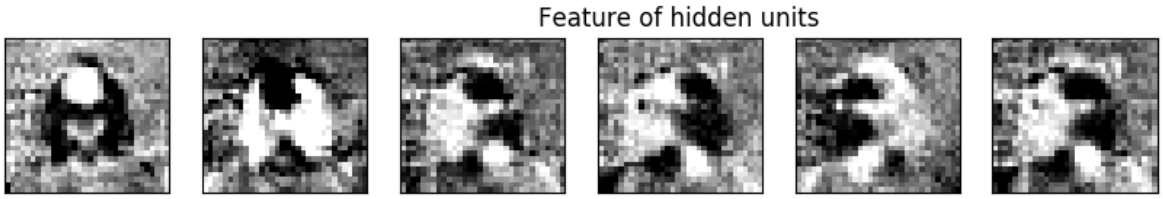
		Predicted Class		
Actual Class	**120**	**4**	**6**	**11**
	2	**134**	**2**	**3**
	1	**3**	**129**	**5**
	8	**3**	**2**	**131**

The rows of the matrix describe the actual instances and the columns the predicted instances. The confusion matrix obtained shows that the most of the predicted instances are correct. This can be seen that the amount of nearly all samples are listed in the diagonal of the confusion matrix. There are only small deviations from the predicted to the actual instance.

Can you find particular regions of the images which get more weights than others?

The white parts of the images indicate the regions with more weight, which are especially the faces of the persons.

Include all plots in your report.



2.2 Face Recognition

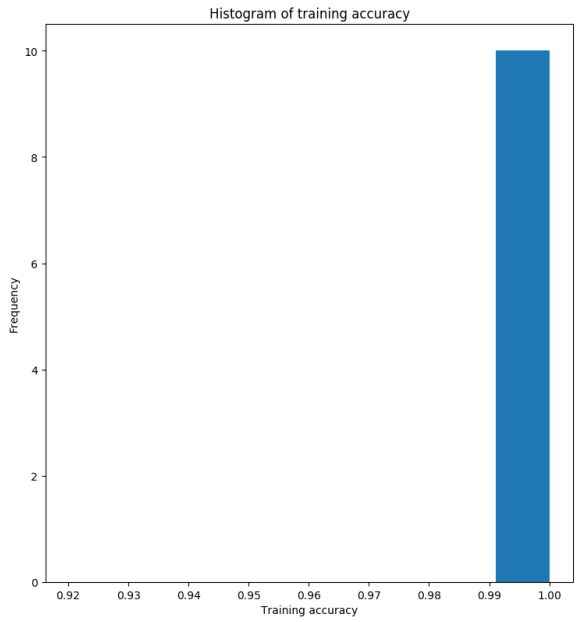
Why do different networks have different accuracies? Explain the variance in the results.

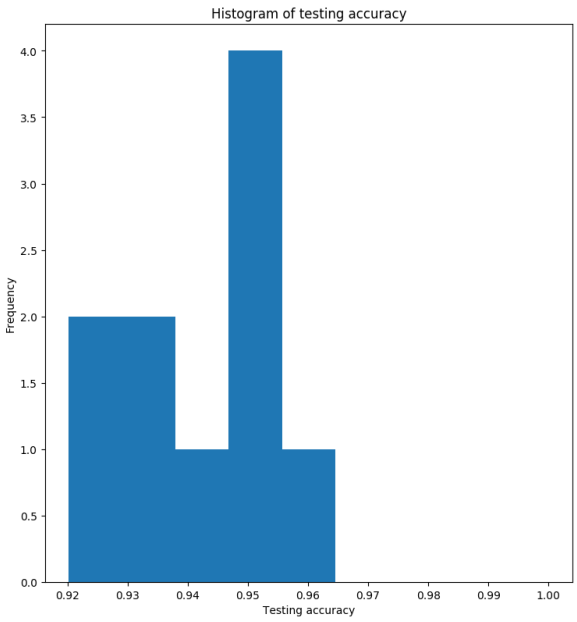
The accuracy is the proportion of correctly classified samples. Different networks will have different amounts of misclassified samples and therefore the accuracy will be also different.

Do the misclassified images have anything in common?

The misclassified images do not include one image where the person's pose is straight. All poses were either left or right, except for one pose is up.

Include all plots in your report.


Histogram of training accuracy
Frequency
Training accuracy
0
2
4
6
8
10
0.92
0.93
0.94
0.95
0.96
0.97
0.98
0.99
1.00


Histogram of testing accuracy
Frequency
Testing accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.92
0.93
0.94
0.95
0.96
0.97
0.98
0.99
1.00