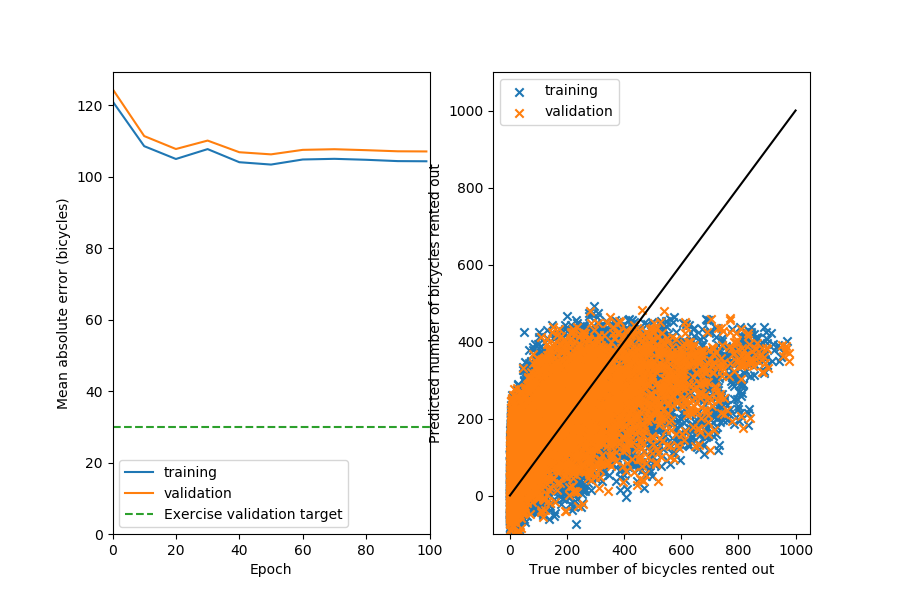
Voorblad

Problem 1: Bicycle rental prediction in TensorFlow

In this section we will used the provided python script ‘bicycle\_predictor.py’ to train a neural network in Tensorflow in order to predict the number of bicycles that will be rented out, based on weather and seasonal data.

Task 1.1  
Initially, the loss is defined as the mean absolute error between the network predictions and the true number of bicycles. According to theory when we use the mean absolute error as cost function the output of the network should converge to the **median** of the dataset.  
Now we change line 75 in ‘bicycle\_predictor.py’ such that the mean squared error is used as the training criterion instead. According to theory the output of the neural network should converge to the **mean** of the dataset.

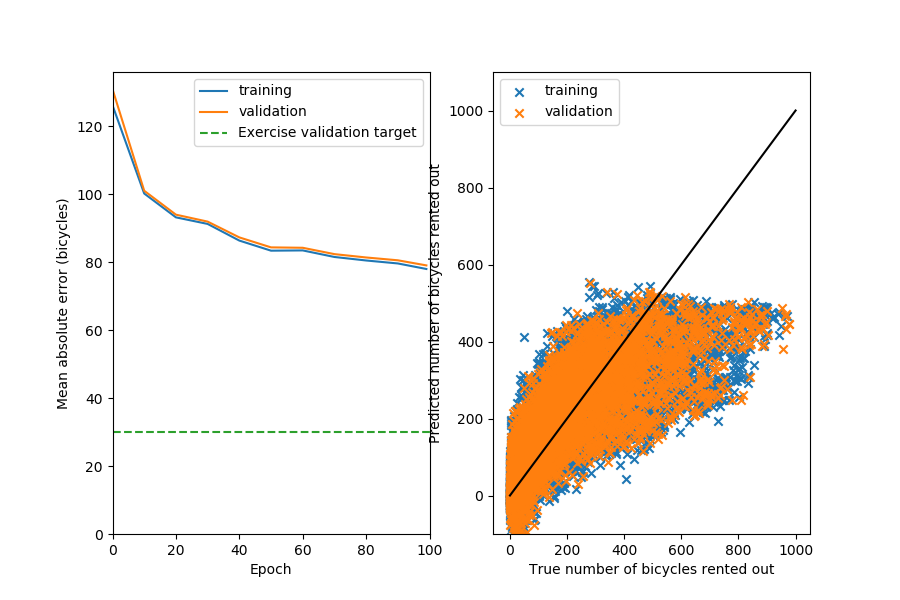
Task 1.2 Linear regression  
Now we change *create\_neural\_network* function (line 84) such that the prediction is an affine function of the input features. We get the following results.

The neural network training converges to a mean absolute error of 104.32 bikes.  
  
If we would increase the amount of linear layers in the neural network by one we get a slight increase in performance (to output of 102.18 bikes). However, if we add too many linear layers, overfitting of the data occurs and we get decreasing performance. The neural network converges slower and to only slightly better output relative to the additional computational power.

If we add extra units to the linear layers, we get only a slight increase in performance compared to the additional computational power.

1.3 Nonlinear regression

Next we changed the hidden layer to use a ReLU (Rectified Linear Unit) activation function. We get the following results.

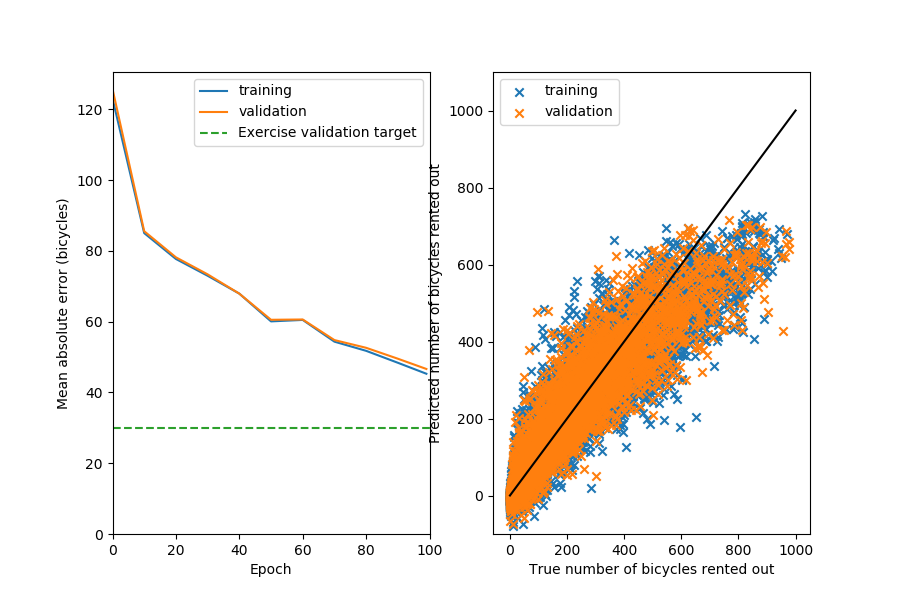


Now the performance has increased to a training value of 77.99 bikes. We observe that the validation value is really close to the training value. Therefore, we conclude that the neural network is currently underfitting the system. This means we could increase the performance by increasing the model complexity and doing more training iterations.

Validation = 79.04

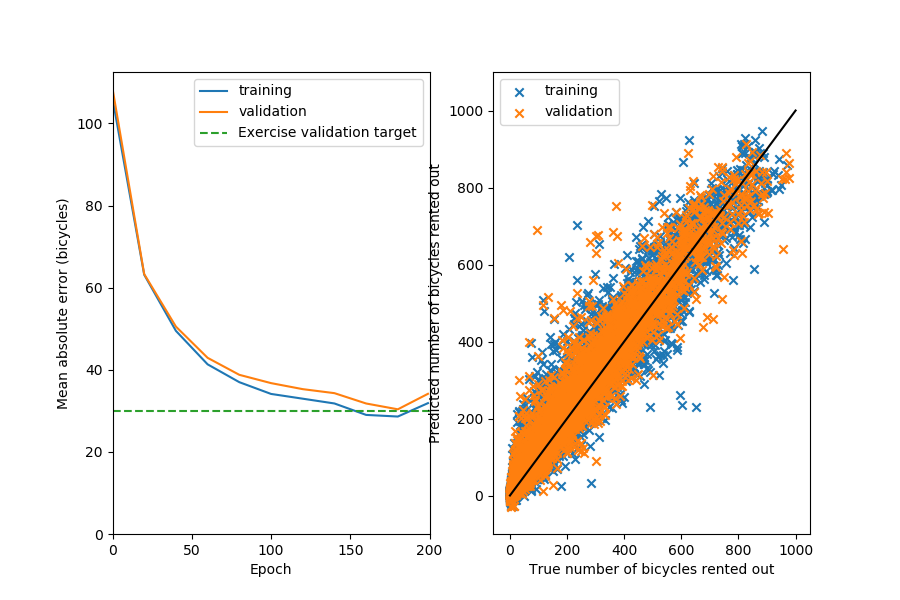
1.4 Going Deep

Next, we’ve changed the neural network such that we have two hidden ReLU layers. Now the training value has improved to a mean absolute error of 45.30 bikes.



If we also change the prediction layer to use a ReLU activation function the mean absolute error doesn’t decrease at all. This may be because a large gradient may cause a unit in the prediction layer to be forever zero. Meaning that neuron will never activate another again, so that the network may stagnate. When we instead use a sigmoidal activation function in the prediction layer by definition the unit will never become zero and the network will not stagnate.

Task 1.5 Hyper-parameters

We increase the number of epochs to 200, (keeping everything else the same) this gives a mean absolute error of 35.62 bikes. Increasing the amount of epochs much further would result in a lot of extra computational effort for only a slight performance increase.  
When we also change the batch\_size to 32 instead of 64, we find a mean absolute error of 31.87 bikes. A smaller batch\_size means that you update your gradients more often, therefore your neural networks is also updated more often and achieves better performance

Bonus:

Using the largest\_data\_point\_errors function we found that on the 23’th of March 2012 a large mistake in the absolute mean error was made. This can explained by the fact that every year on that day there is a Cherry Blossom Festival in Washington DC, therefore more people may opt to rent a bike that day.