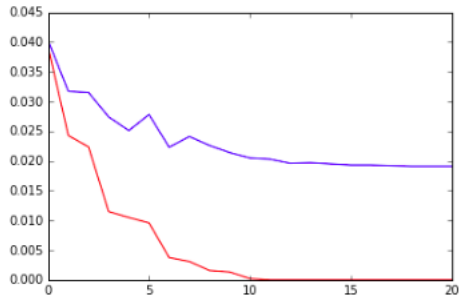
Report: Feed-Forward Neural Net

Theresa Eimer

The best result I achieved with the network was a validation error of 1,91% and a training error of 2,75%. These are the results from the training cycle, the graph shows the training (red) and validation error (blue):

```
epoch 1.0000, loss 0.0812, train error 0.0243
epoch 1.0000, validation loss 0.1176, validation error 0.0317
epoch 2.0000, loss 0.0697, train error 0.0224
epoch 2.0000, validation loss 0.1157, validation error 0.0315
epoch 3.0000, loss 0.0374, train error 0.0115
epoch 3.0000, validation loss 0.1012, validation error 0.0274
epoch 4.0000, loss 0.0326, train error 0.0105
epoch 4.0000, validation loss 0.1048, validation error 0.0251
epoch 5.0000, loss 0.0298, train error 0.0096
epoch 5.0000, validation loss 0.1186, validation error 0.0278
epoch 6.0000, loss 0.0111, train error 0.0038
epoch 6.0000, validation loss 0.1075, validation error 0.0223
epoch 7.0000, loss 0.0099, train error 0.0031
epoch 7.0000, validation loss 0.1122, validation error 0.0241
epoch 8.0000, loss 0.0054, train error 0.0016
epoch 8.0000, validation loss 0.1087, validation error 0.0226
epoch 9.0000, loss 0.0042, train error 0.0013
epoch 9.0000, validation loss 0.1174, validation error 0.0214
epoch 10.0000, loss 0.0010, train error 0.0002
epoch 10.0000, validation loss 0.1131, validation error 0.0205
epoch 11.0000, loss 0.0004, train error 0.0000
epoch 11.0000, validation loss 0.1114, validation error 0.0203
epoch 12.0000, loss 0.0002, train error 0.0000
epoch 12.0000, validation loss 0.1129, validation error 0.0196
epoch 13.0000, loss 0.0002, train error 0.0000
epoch 13.0000, validation loss 0.1149, validation error 0.0197
epoch 14.0000, loss 0.0002, train error 0.0000
epoch 14.0000, validation loss 0.1164, validation error 0.0195
epoch 15.0000, loss 0.0002, train error 0.0000
epoch 15.0000, validation loss 0.1175, validation error 0.0193
epoch 16.0000, loss 0.0001, train error 0.0000
epoch 16.0000, validation loss 0.1185, validation error 0.0193
epoch 17.0000, loss 0.0001, train error 0.0000
epoch 17.0000, validation loss 0.1193, validation error 0.0192
epoch 18.0000, loss 0.0001, train error 0.0000
epoch 18.0000, validation loss 0.1200, validation error 0.0191
epoch 19.0000, loss 0.0001, train error 0.0000
epoch 19.0000, validation loss 0.1206, validation error 0.0191
epoch 20.0000, loss 0.0001, train error 0.0000
epoch 20.0000, validation loss 0.1211, validation error 0.0191
```




```
Duration: 493.4s
```

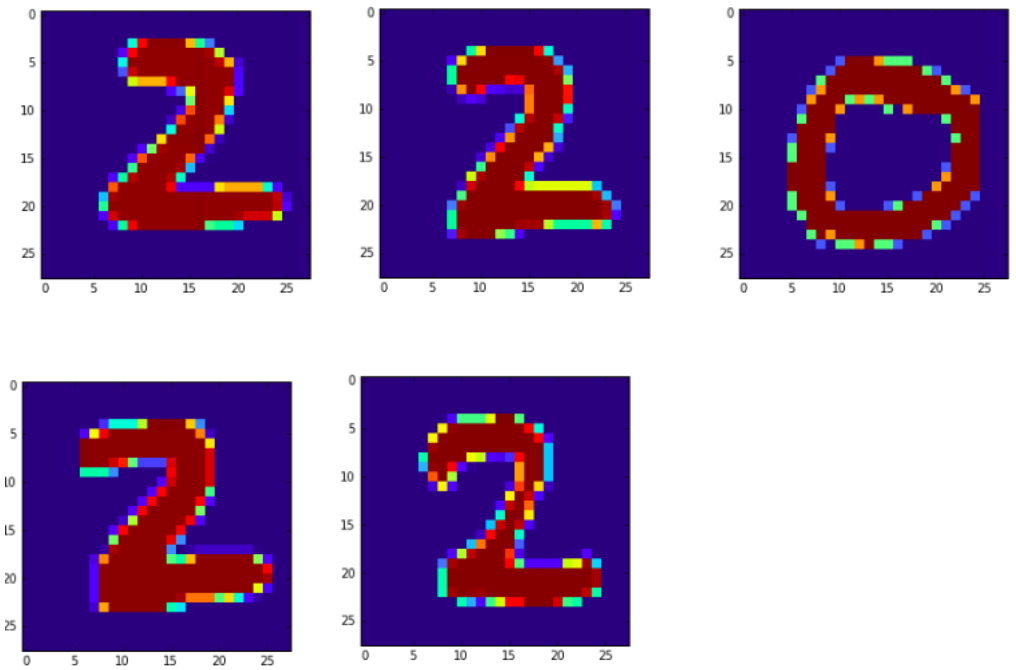
The net used here was comprised of four hidden layers, an input layer and a Softmax output layer, as it made the most sense while using one-hot encoding. The first three hidden layers had 350 units each and the last one 10 to

account for the 10 different possible outputs in the encoding. In each layer, the weights and biases were randomly initialized with a standard deviation centered at zero with a spread of 0.05. The ReLu activation function got similar results as the Sigmoid or Tanh functions, but needed less iterations, so it was the activation function of choice for all but the last (linear) layer. As descend method, Stochastic Gradient Descend performed better than the simple Gradient Descend. The learning rate was set at 0.3 with a decay of 10% per epoch. The batch size sat relatively low at 20 data points per batch and this allowed for a small number of epochs, namely 20.
Here are the test results and some interesting information from the test set:

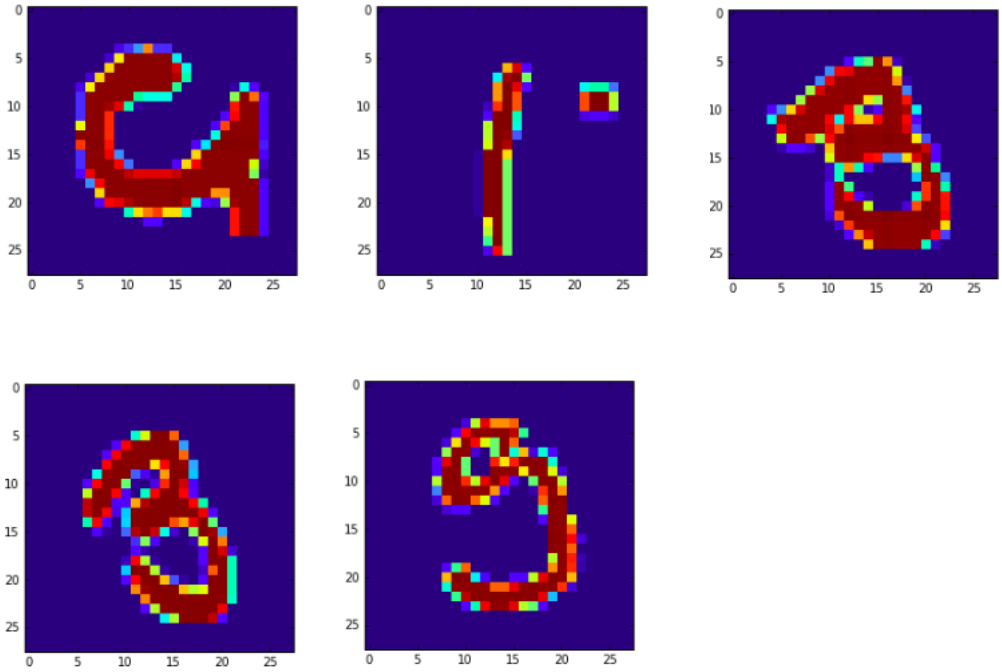
```
Test error: 0.02750
Errors per number:
[ 16.   8.  28.  29.  17.  38.  25.  39.  28.  47.]
Number most often classified correctly: 1
Number most often classified incorrectly: 9
```

In testing, the number that was recognized correctly the most was the number one and the number recognized correctly the least was the nine. This isn't completely unexpected, the one doesn't resemble many other numbers, unlike the nine which may be confused with the eight or the zero and is all together a very simple shape.

The images with the best scores in the test set were (from left to right):



All numbers here are relatively tidy and very easy to recognize for the human eye.
The images with the worst scores were:



These are less tidy and closer to shapes of other numbers. Those results can't be used to conclude that results follow a pattern like "shapes that are easy to predict for humans are easy to predict for neural nets", of course, but they do seem intuitively logical to a human observer.