EPL442

Assignment 2

Loukas Papalazarou 1013129

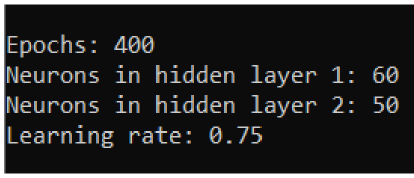
1. Design choices

For my neural network I followed an object-oriented approach, modeling the Neurons, Connections and Network as separate objects

- Neurons have their value, delta value, inbound and outbound connections (lists of Connections) as well as functions for calculating new value and delta.
- Connections have a weight and two neurons
- The Network a list of Neurons for each layer and methods for initialization, single input prediction, learning (backpropagation) and entire training as well as some other utility functions.
- Errors are calculated based on the average absolute value of the difference between expected bit and actual bit for all bits. For example, if wanted is [0, 0, 1, 0] and actual is [0.3, 0.1, 0.9, 0.2], then the error will be:
  error = avg(abs(0–0.3), abs(0-0.1), abs(0.9-1), abs(0-0.2))
- For the success rate I took the correctness rate for each epoch. A result is correct if the highest valued bit is the bit of the expected output that is equal to 1.
- For every epoch, the input data was fed in a **random order**. The goal is to avoid overfitting the network. That is the reason for drops in success rate from one epoch to the next.

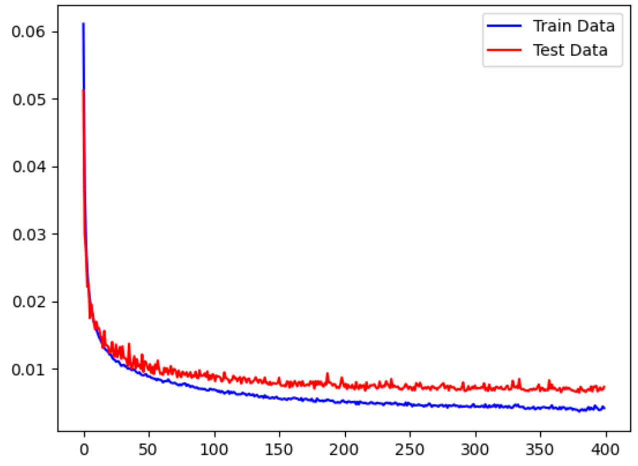
2. Control Run

This is a control run. An execution of training and testing with the parameters I found to be optimal.

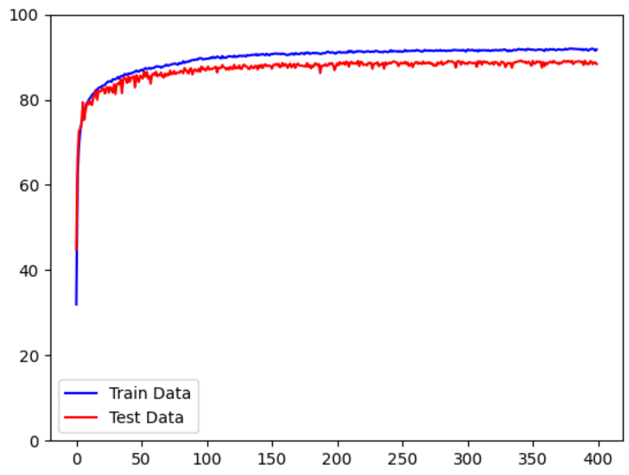


```
Epochs: 400
Neurons in hidden layer 1: 60
Neurons in hidden layer 2: 50
Learning rate: 0.75
```

Errors



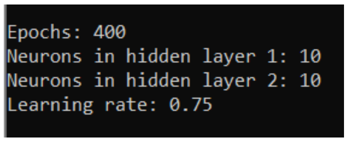
Success rate



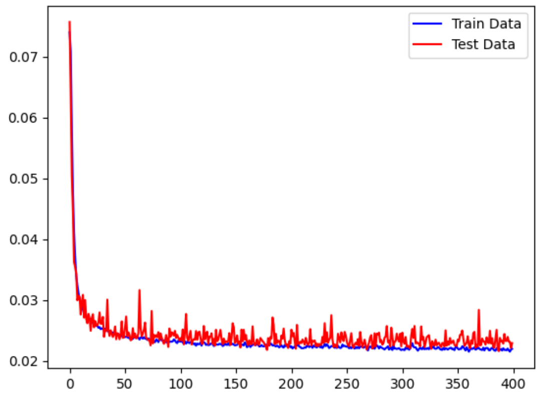
As we can see, errors begin at around 0.06 and by epoch 50 are under 0.01. At this point, the training errors settle at around 0.005 whereas the testing error remains at just under 0.01. This is expected since the model has never been trained on the testing data and will of course make some mistakes. Same goes for the success rate. Training success rate stabilizes around 90%, while testing success rate is around 85%.

3. Training with various parameters

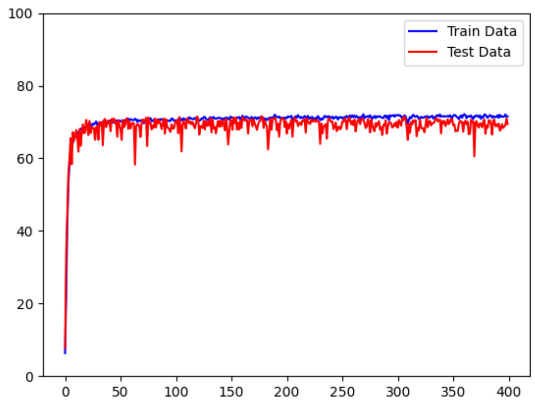
3.1 Fewer neurons on both hidden layers


```
Epochs: 400
Neurons in hidden layer 1: 10
Neurons in hidden layer 2: 10
Learning rate: 0.75
```

Errors

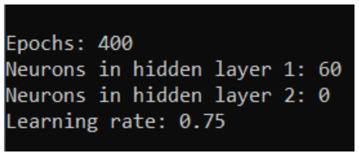


Success rate

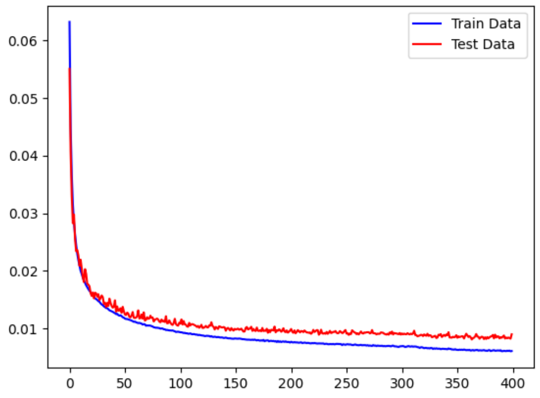


The most interesting observation when using very few neurons is that the network is not being trained properly. It only reaches a success rate of ~70% and the errors remain at a high 0.02. We can also see that both errors and success testing fluctuate a lot. This is probably caused by the unpredictability of the network since it is not properly trained.

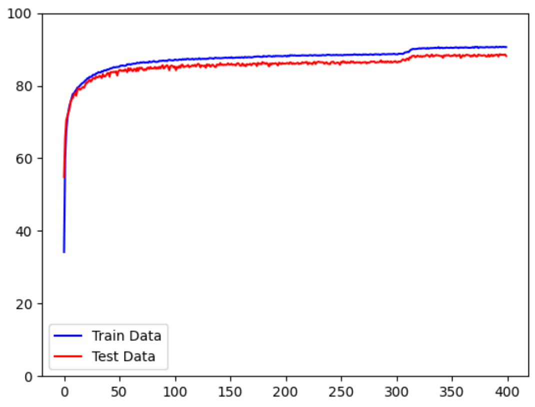
3.2 Only 1 hidden layer


```
Epochs: 400
Neurons in hidden layer 1: 60
Neurons in hidden layer 2: 0
Learning rate: 0.75
```

Errors

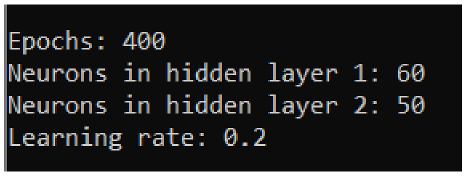


Success rate



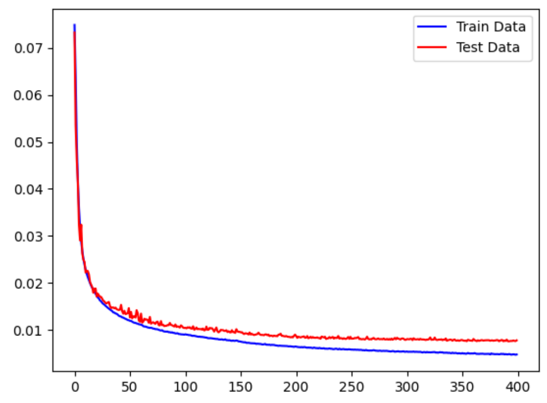
What we can see here is that having only one hidden layer does not have a huge impact on the model which is what I expected. Instead, it manages to get properly trained, reaching a success rate of ~90%

3.3 Small learning rate

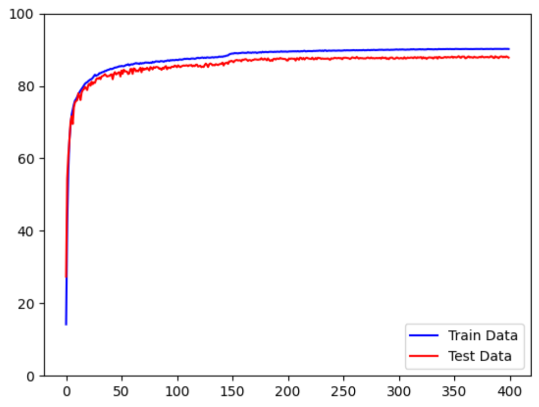


```
Epochs: 400
Neurons in hidden layer 1: 60
Neurons in hidden layer 2: 50
Learning rate: 0.2
```

Errors



Success rate



Here, where the learning rate is so low but the other parameters are optimal(?), the network get properly trained with a good 90% success rate, as expected. What is noteworthy here is how smooth the training curve is. This is due to the small learning rate allowing the network to not be thrown of by some inevitable mispredictions.