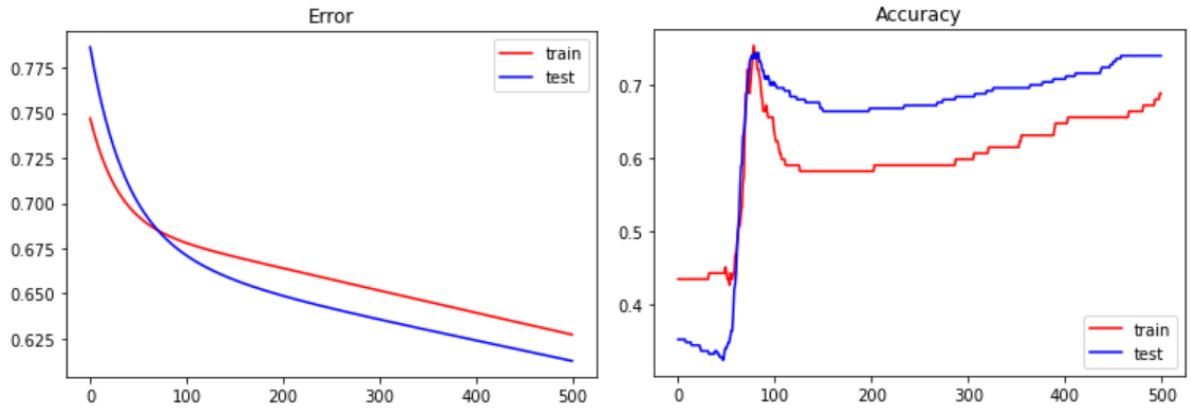
INVESTIGATIONS FROM TASK ONE

The results obtained from Task One are shown by the following graphs



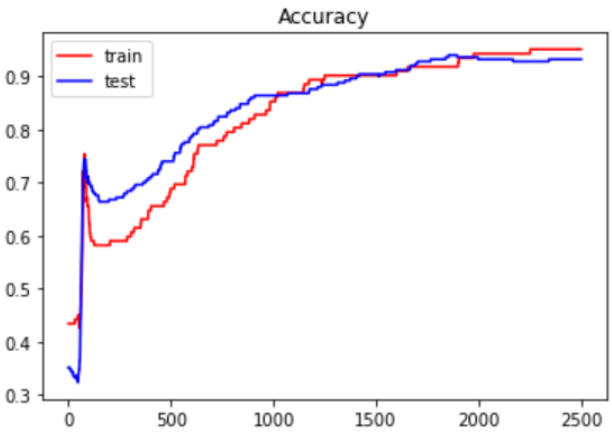
From here we can see, that as expected, the error (calculated using cross entropy) shows a slow and gradual decline with each epoch, indicating the learning is going well and moving towards a small error as possible.

We are also able to see that the accuracy appears to reach its peak around epoch ~100. This indicates that the optimal weights were found around this epoch, and as training continued it moved away from those weights as it was updating them. This is shown as a slight decline shortly after reaching its peak, before it gradually begins to increase again as the updating of the weights gets closer to the values it found at its peak.

Increasing epochs

It is expected that as the number of epochs are increased, the resulting final accuracy and error should improve as the model has had more time to train

Epochs	Final Accuracy	Final Error
500	0.740	0.613
1000	0.864	0.552
1500	0.904	0.482
2000	0.932	0.411
2500	0.932	0.350



As expected, we see a dramatic increase in accuracy and a dramatic decrease in the error values as the number of epochs is increased. The accuracy seems to peak at 2000 epochs, as no improvement was shown from 2000 -> 2500, however the error value did decrease between these two epochs settings

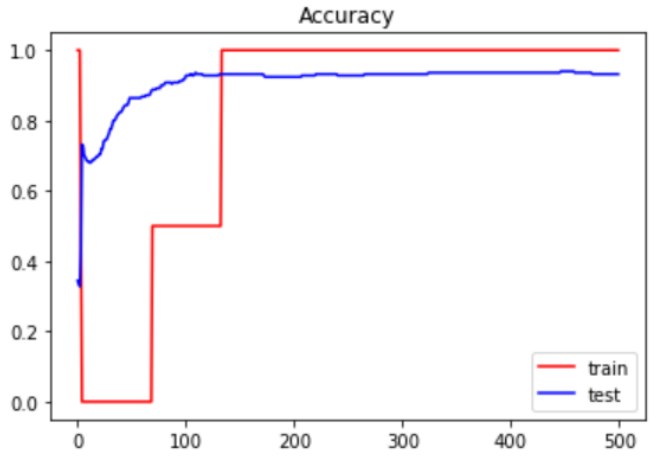
Changes in batch size

Changing the batch size will result in the model updating its weight multiple times per epoch, as it changes its weights every time its trained on a batch. A smaller batch size could increase the accuracy, as the smaller the batch, the more updates to the weights it provides.

All values below were recorded with the default settings from task one (epochs, activation function, neurons etc), the only change being the batch size.

Batch Size	Final Accuracy	Final Error
8	0.932	0.174
16	0.924	0.242
32	0.940	0.408
64	0.864	0.551
128	0.740	0.613

As expected, the model seems to benefit from a smaller batch size, as it is able to make more updates every epoch to find the ideal weights. While the best accuracy was achieved using a batch size of 32, there is a much larger error compared to using a batch size of 8, whilst the difference in accuracy between these two is negligible. The graph below are from batch size 8.



Learning Rate

Learning Rate	Final Accuracy	Final Error
0.001	0.340	0.701
0.005	0.672	0.642
0.01	0.740	0.613
0.05	0.932	0.350
0.1	0.936	0.209

As shown above, decreasing the learning rate gives a significant decrease in accuracy. These results were obtained using the default configuration for the rest of the parameters, however I believe that increasing the epochs and decreasing the batch size will yield better results with a lower learning rate, as it will have many more opportunities to update the weights. The results below were obtained using a batch size of 8, and 2000 epochs

Learning Rate	Final Accuracy	Final Error
0.001	0.924	0.289
0.005	0.936	0.149
0.01	0.940	0.136
0.05	0.940	0.159
0.1	0.948	0.190

As the accuracy change was of negligible difference when using a higher epoch setting, I've decided to keep the initial learning rate of 0.01 as the 'best' option here, whilst a learning rate of 0.1 does provide a higher accuracy, using such a high learning rate does not seem like good practice.

Neurons

Neurons	Final Accuracy	Final Error
5	0.740	0.613
10	0.728	0.601
20	0.812	0.573
30	0.872	0.504
40	0.888	0.442
50	0.848	0.529

Activation Function

Layer One f(x)	Layer Two f(x)	Final Accuracy	Final Error
ReLu	Sigmoid	0.912	0.338

Best Overall Model

From my investigations to all the different parameters above, the best accuracy and loss values should be achieved by using the following parameter values:

Epochs	Batch Size	Learning Rate	Layer One F(x)	Layer Two F(x)	Neurons	Final Accuracy	Final Error
2500	8	0.01	ReLu	Sigmoid	40	0.948	0.184