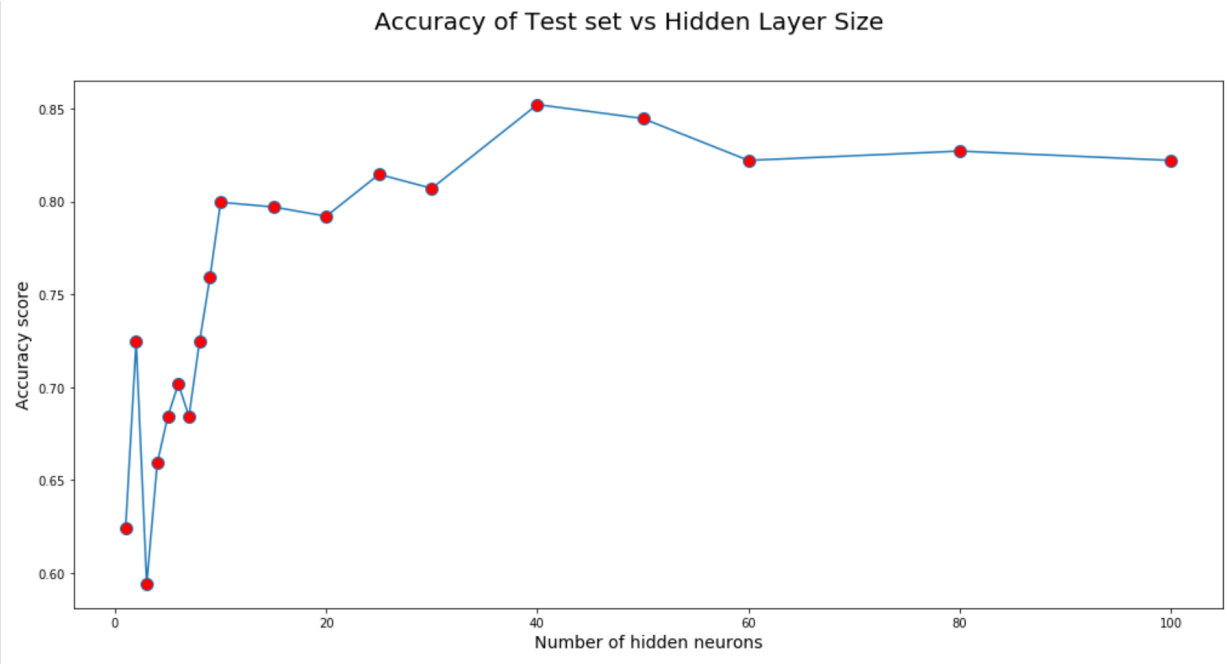
train1.csv

1. How does the number of hidden units impact the results?

With fewer hidden layer neurons, the model performs badly, but as more neurons are added it starts to perform better. However, increasing even more results in the same performance. It is likely that network starts to overfit to the training set and as a result the test accuracy suffers. Increasing the number of neurons in the hidden layer increases accuracy until it starts to overfit.

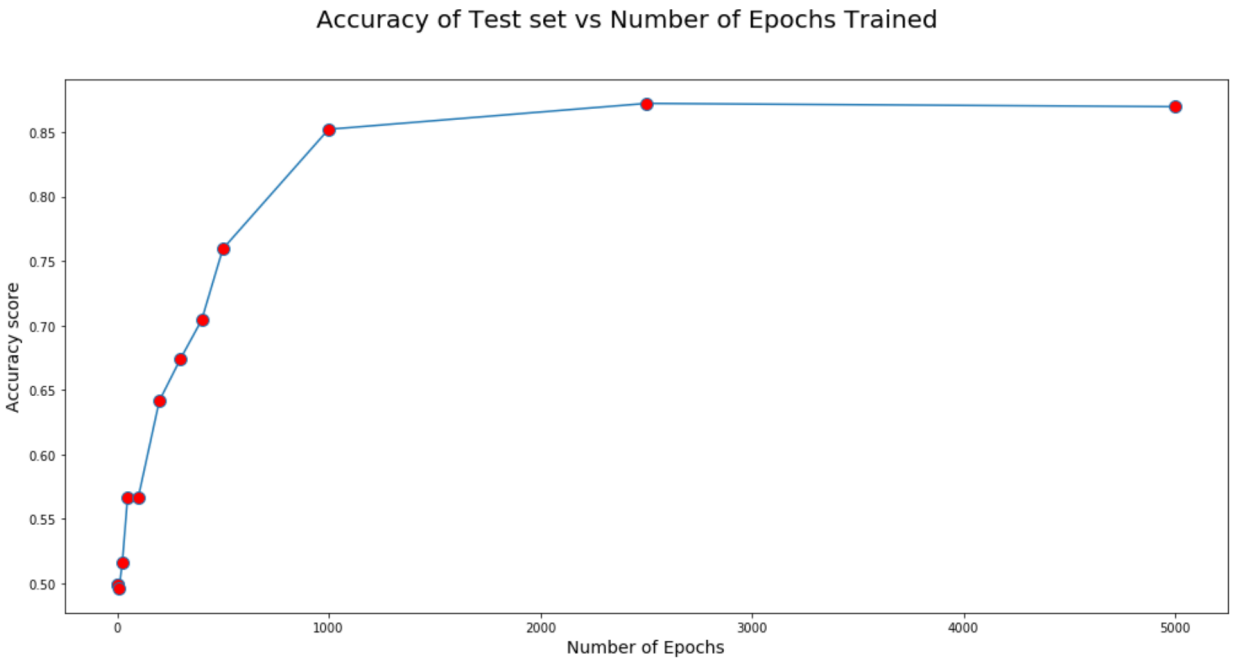


Hidden layer sizes 1,2,3,4,5,6,7,8,9,10,15,20,25,30,40,50,60,80,100

1000 epochs, learning rate of 0.01

2. How does the training time impact the results?

As the training time (number of epochs trained) increases, the accuracy in the test set increases as well up until a point. Increasing the number of epochs didn't affect accuracy and it plateaued beyond that. The amount of compute time goes up at that point with no significant gains in performance. Increasing training time improves results but it will plateau and become more computationally expensive.

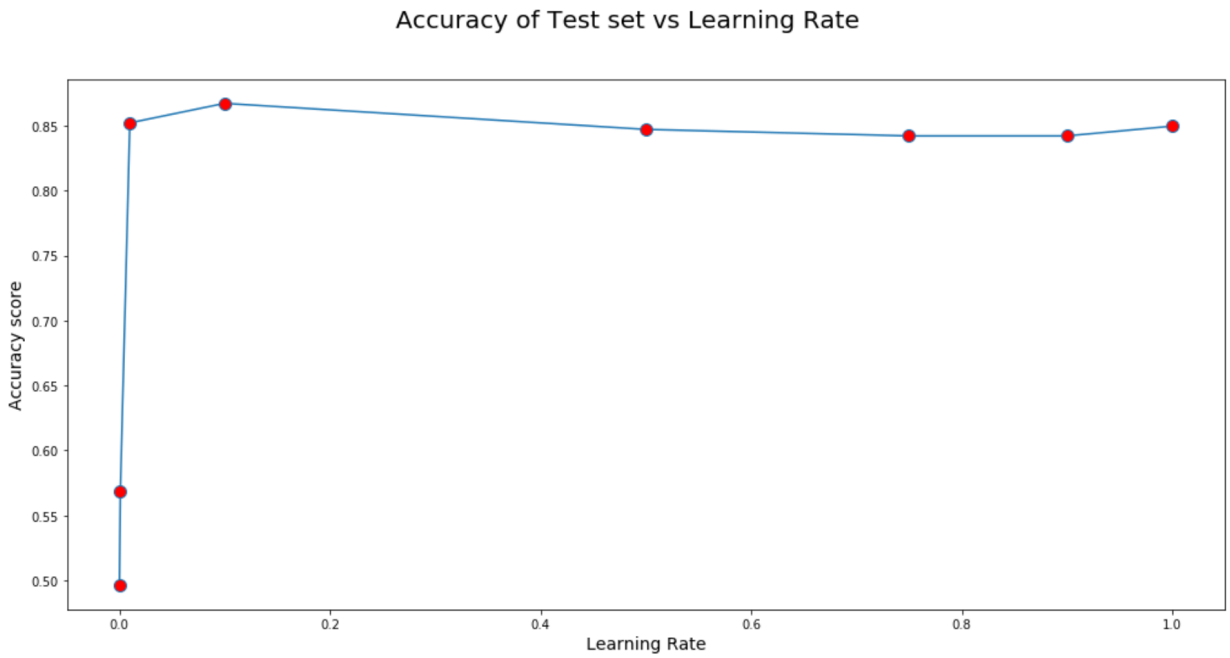


Trained on epoch_sizes of 1, 5, 10, 25, 50, 100, 200, 300, 400, 500, 1000, 2500, 5000

Learning Rate of 0.01, 40 hidden neurons

3. How does the learning rate impact the results?

As the learning rate increased, the test accuracy increased as well. I believe this is because a lower learning rate would be more accurate but requires more epochs to train. In this case since we have so many epochs, the lower learning rate has better accuracy and it starts to get worse as it gets higher.

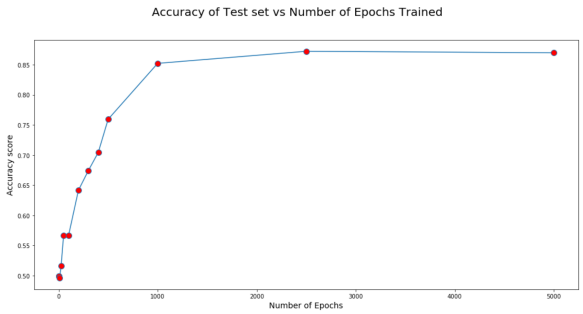


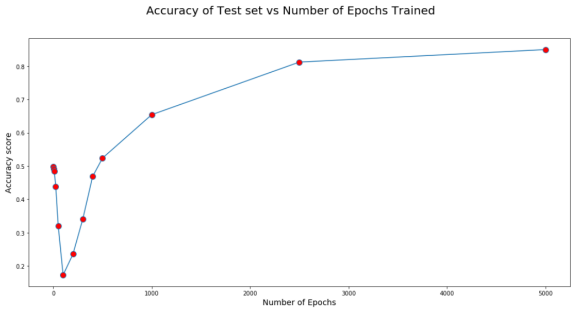
Conducted with 1000 epochs

Learning rate 0.1, Hidden layer neurons 64

4. What other critical parameters impacted the results?

I played with the final activation layer function and in both the training time and the learning rate graphs had some interesting changes. On the left is using softmax as the activation function for the output layer and the right is using sigmoid. Sigmoid decreased accuracy from 1 epoch to around 25, this could be because it learns decently but needs more epochs to get really accurate.

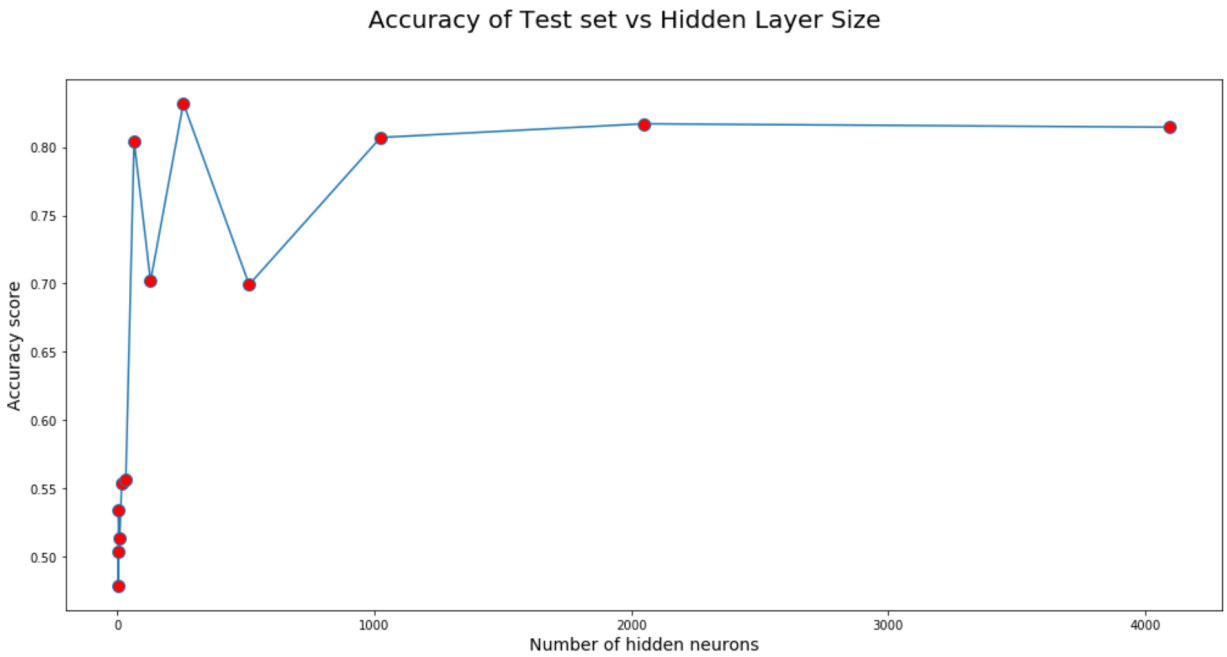




train2.csv

1. How does the number of hidden units impact the results?

With fewer hidden layer neurons, the model performs badly, but as more neurons are added it starts to perform better. We need more hidden neurons than in the first test set. Increasing the number of neurons in the hidden layer increases accuracy until it starts to plateau.

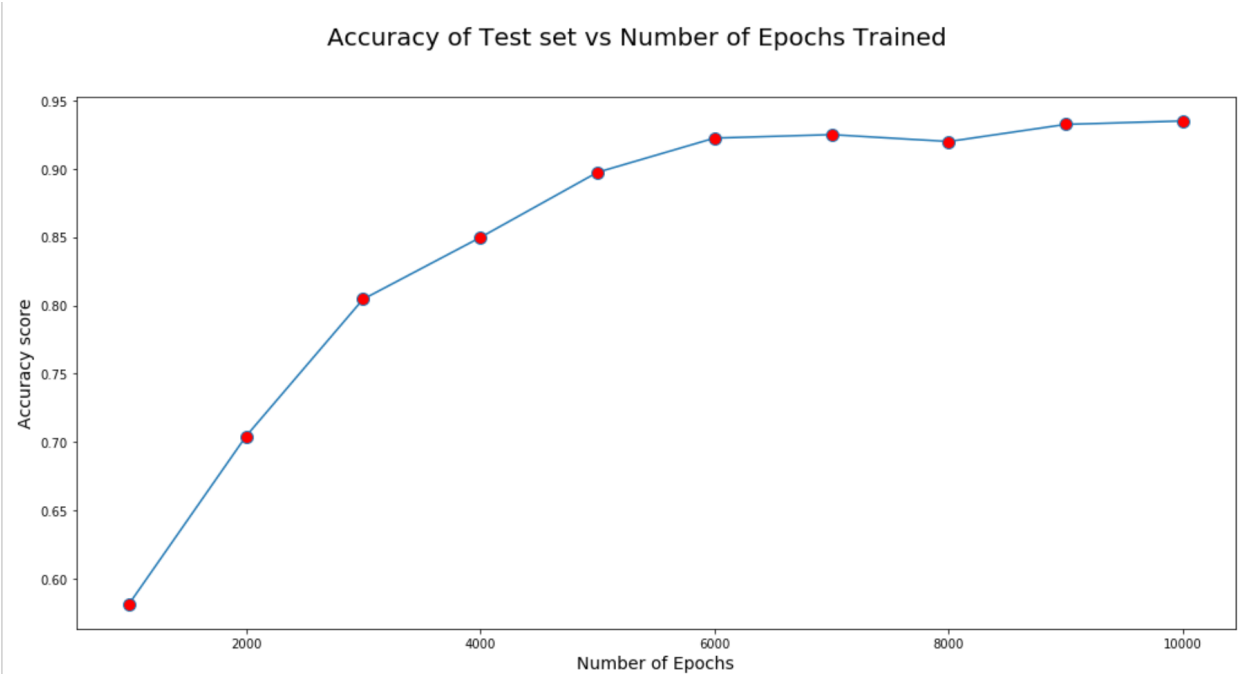


Number of hidden layer neurons 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

3000 epochs, learning rate of 0.1

2. How does the training time impact the results?

As the training time (number of epochs trained) increases, the accuracy in the test set increases as well up until a point. In the experiment I conducted, the graph below shows that at 6000 epochs, increasing the number of epochs didn't affect accuracy and it plateaued beyond that. The amount of time of compute time goes up at that point with no significant gains in performance. Increasing training time improves results but it will plateau and become more computationally expensive.

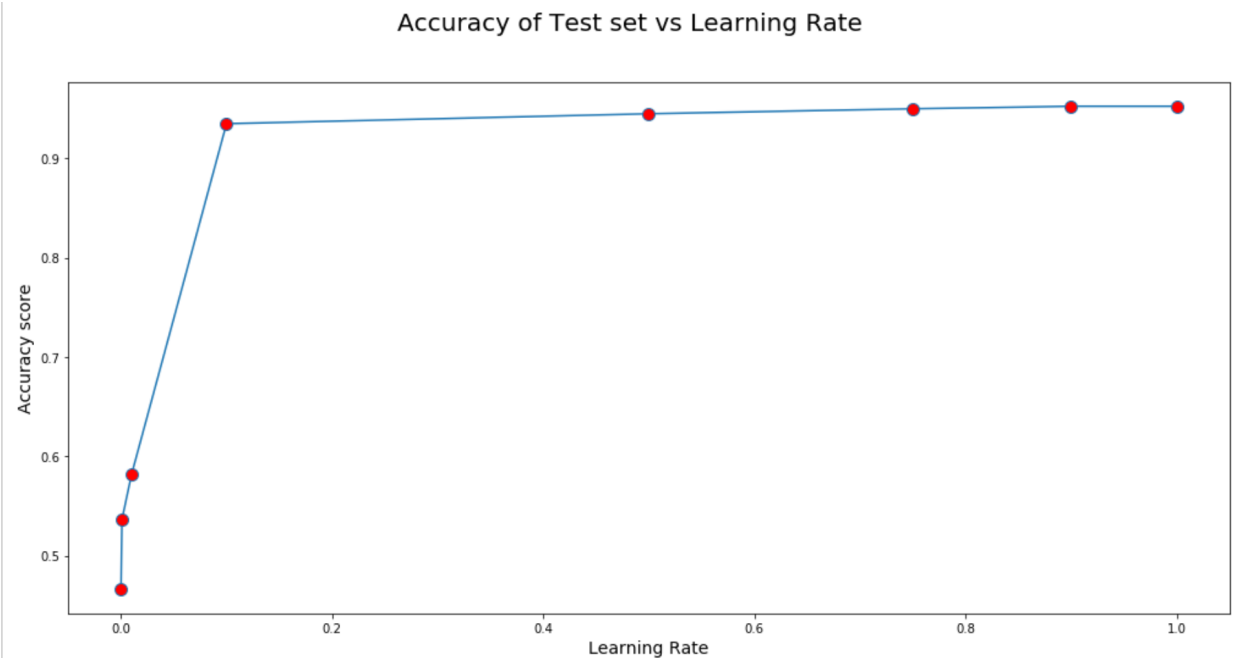


Trained on epoch_sizes of 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000

Learning Rate of 0.1, 64 hidden neurons

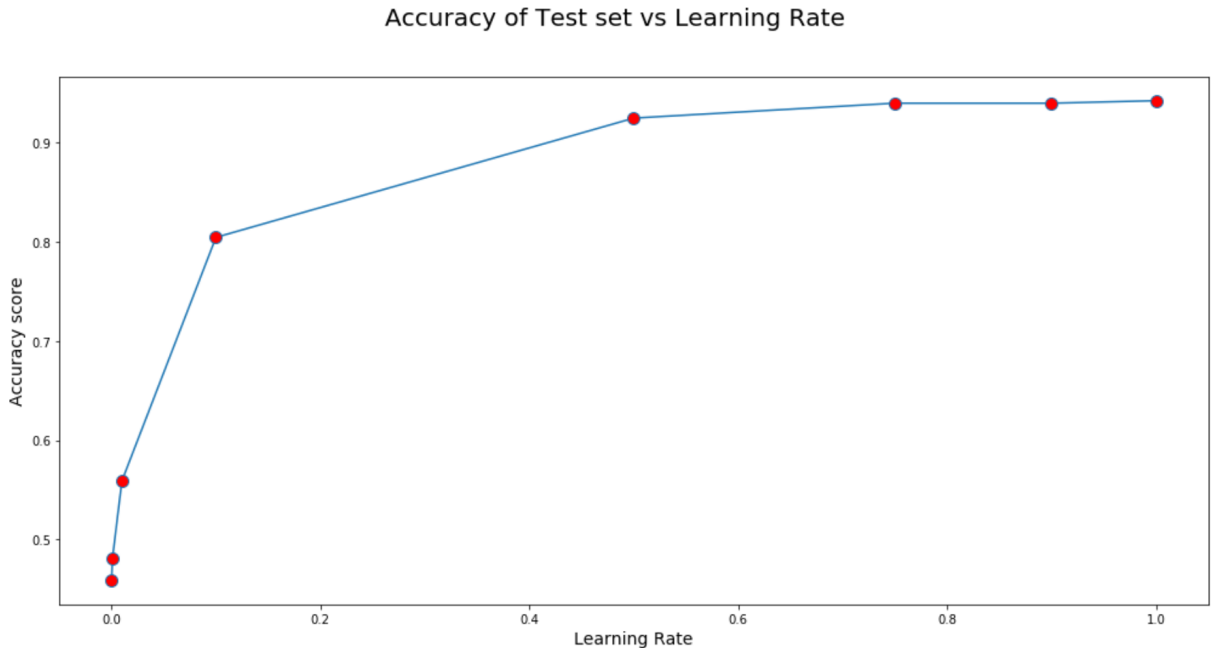
3. How does the learning rate impact the results?

As the learning rate increased, the test accuracy increased as well. I conducted the same test with 3000 and 10000 epochs and you can see that the lower learning rate ends up reaching the same amount of accuracy with more time.



Learning rates of 0.0001, 0.001, 0.01, 0.1, 0.5, 0.75, 0.9, 1

10000 epochs, 64 hidden layer neurons

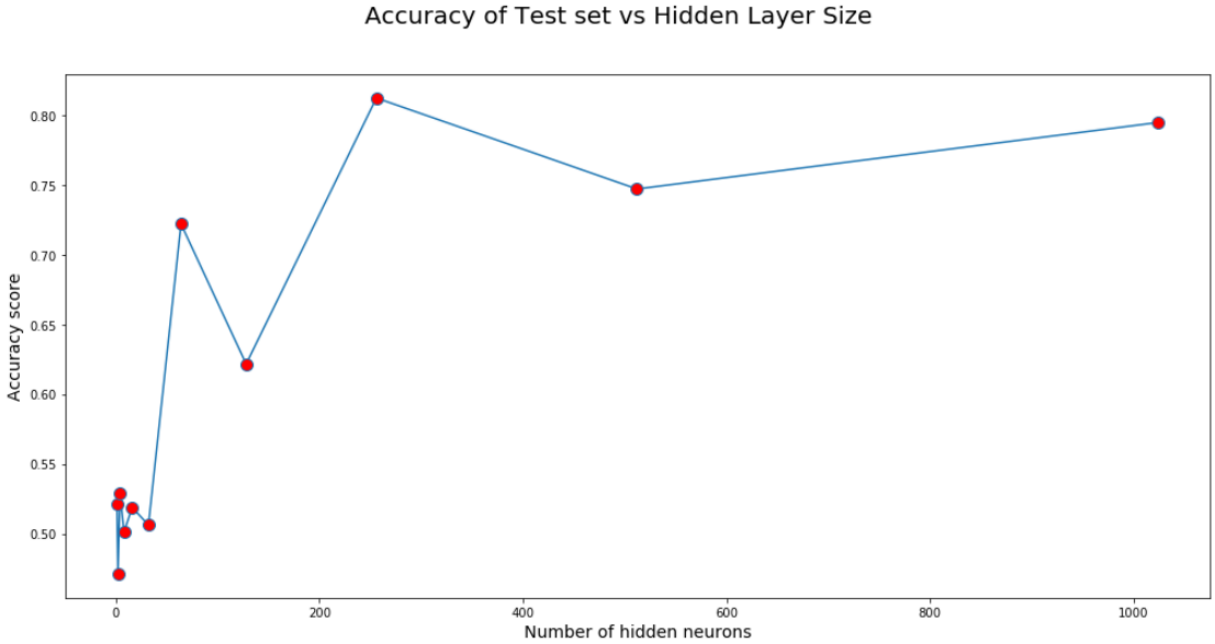


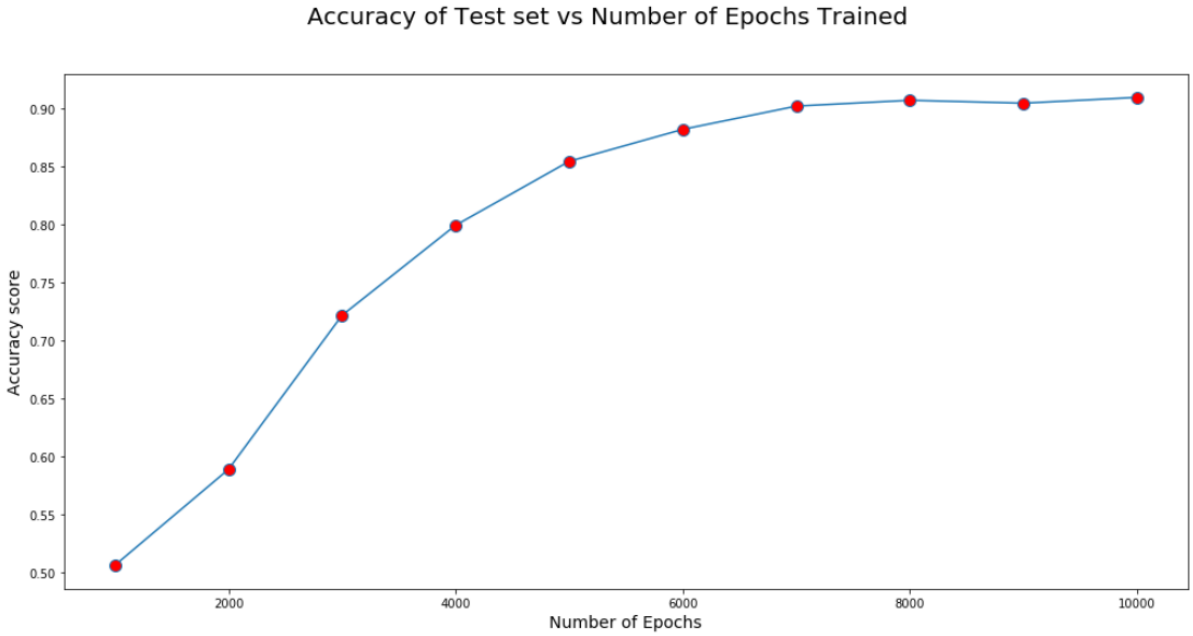
Learning rates of 0.0001, 0.001, 0.01, 0.1, 0.5, 0.75, 0.9, 1

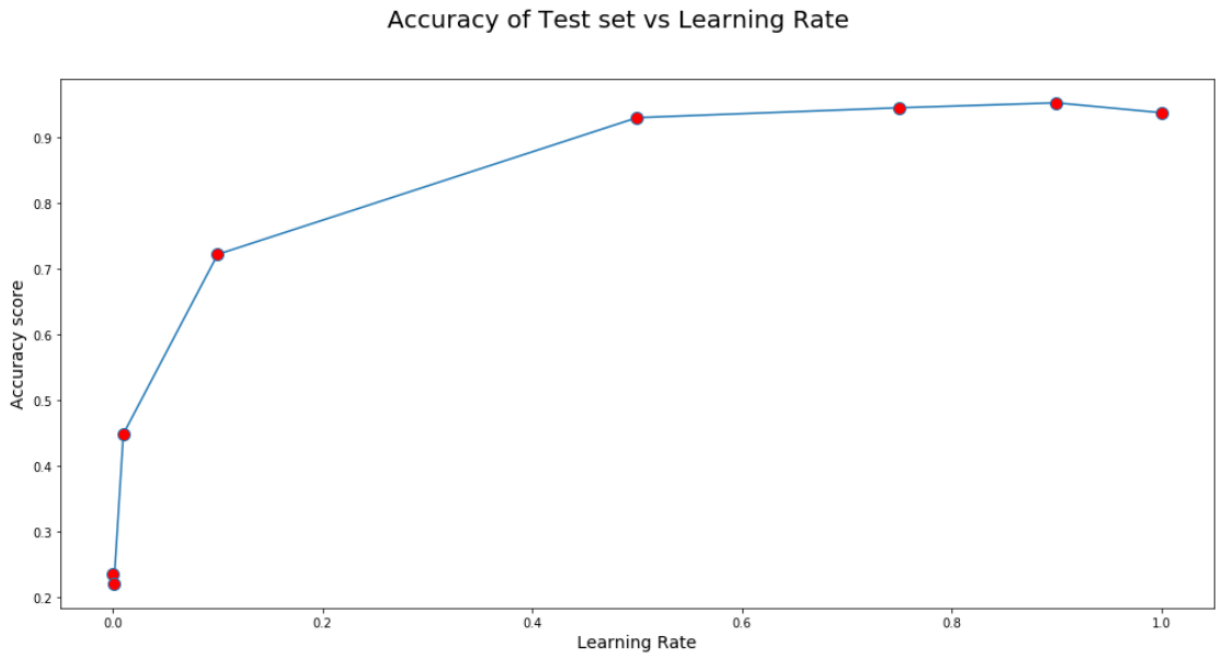
3000 epochs, 64 hidden layer neurons

4. What other critical parameters impacted the results?

Once again I played with the final layer activations. In all the above graphs I used softmax as a final layer, but the below graphs use sigmoid. With sigmoid as a final layer, it appears to prefer fewer hidden neurons, more epochs, and a higher learning rate, with all other parameters kept the same from the above graphs.


Accuracy of Test set vs Hidden Layer Size
Accuracy score
Number of hidden neurons
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
200
400
600
800
1000


Accuracy of Test set vs Number of Epochs Trained
Accuracy score
Number of Epochs
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
2000
4000
6000
8000
10000



5. What conclusions can you draw from your results? What do you think is causing the difference in performance?

The second set seems to consistently >90% test accuracy with tuned parameters but the first set seems to settle at around 85%. It's possible that the first set being simpler that the data isn't as varied so the test set performs worse but the second set provides more varied data which lets us train better. The first set could contain more outliers which are not close to the rest of the data. There could also be strange distributions where it's unequal. There could also be more features in the first set and therefore we would need more hidden layers to classify more accurately.