Roger Wienaah

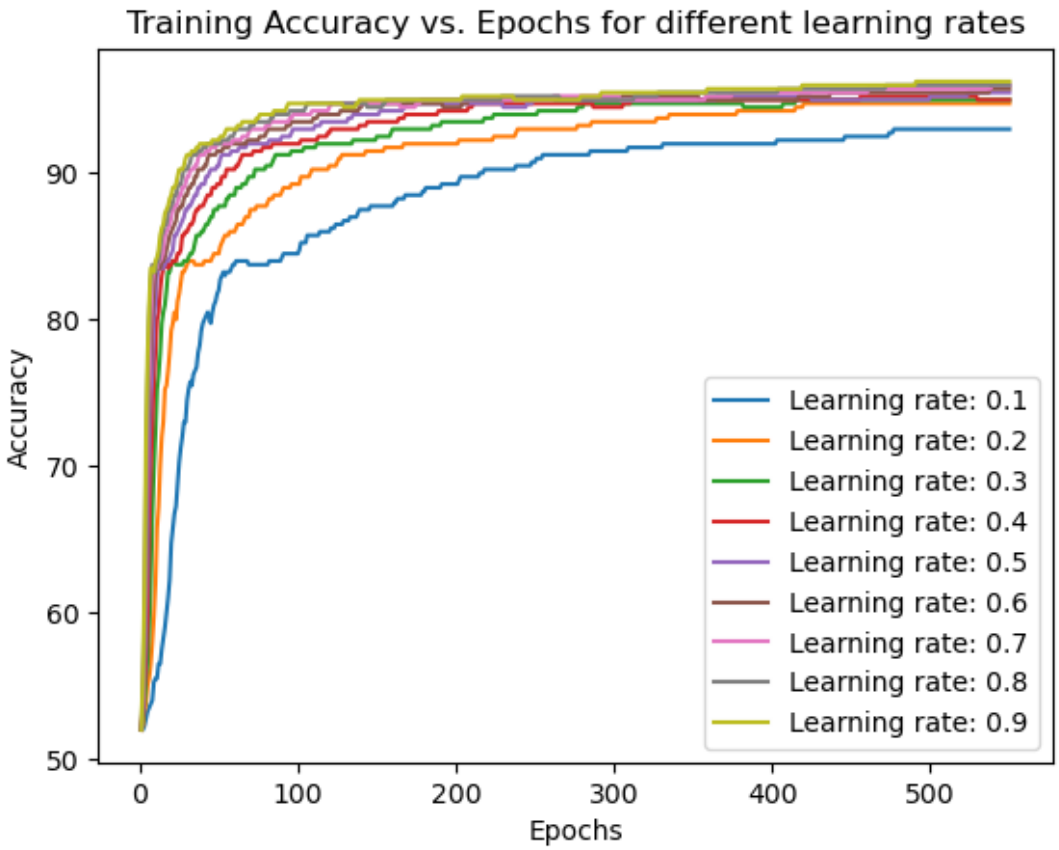
ROB 537 Homework 1

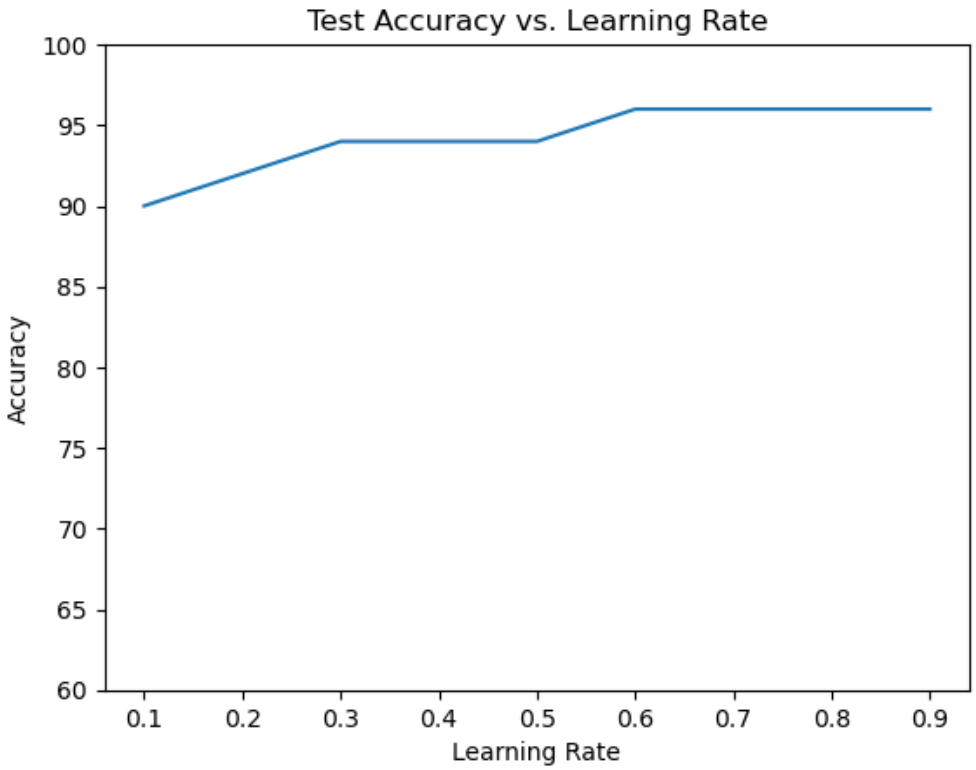
Question 1: Performance of the network using train1.csv and test1.csv

1.1 Impact of different learning rates on the performance of the network.

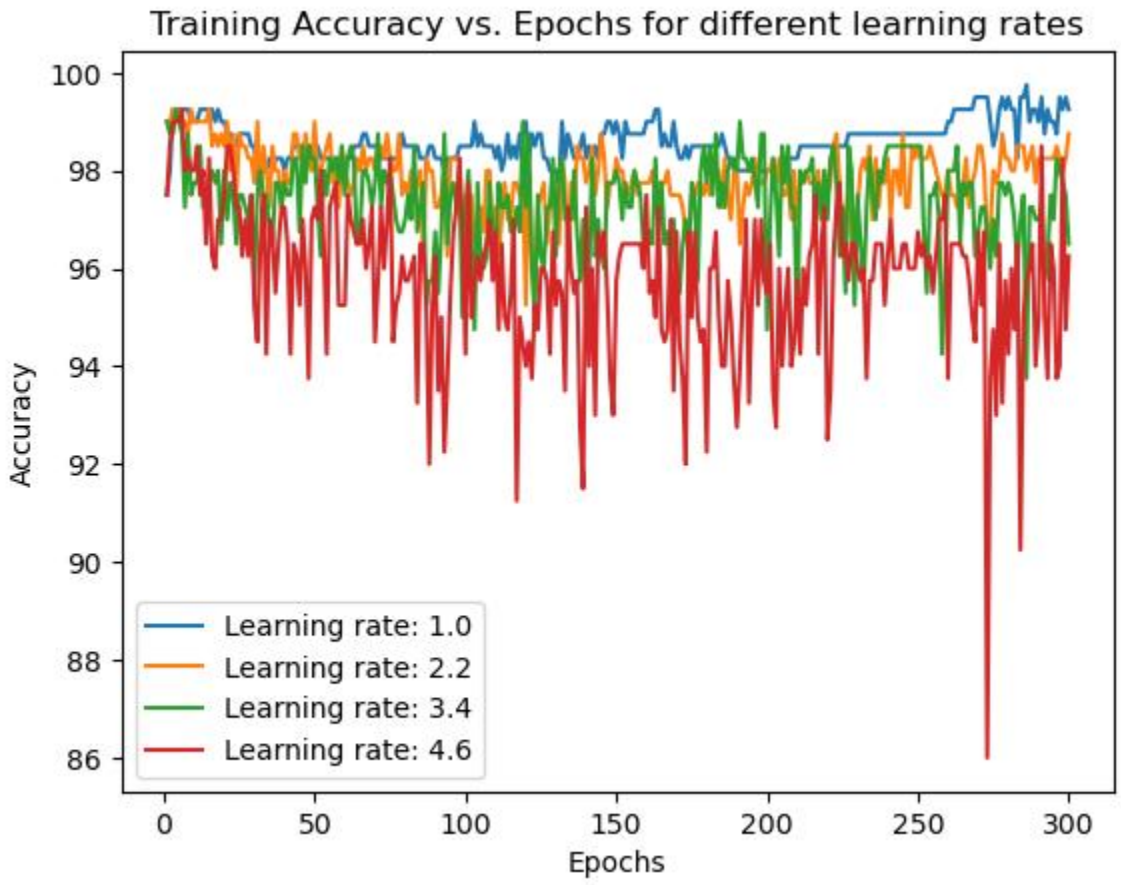
To evaluate the impact of different learning rates on the performance of the neural network, I experimented with learning rates ranging from 0.1 to 1 with a step size of 0.1, and 1 to 5 with a step size of 1.5. The number of hidden units in my network was 10, and I used the mean squared error as my loss metric. As shown in the plots below, I observed that smaller learning rates took longer number of epochs to converge compared to larger learning rates. This is because the learning rate causes the gradients to take very small steps for smaller values, while big steps are taken for larger learning rates, which can lead to instability during training (gradients updates bouncing around and failing to find the minimum when very large learning rates are used).

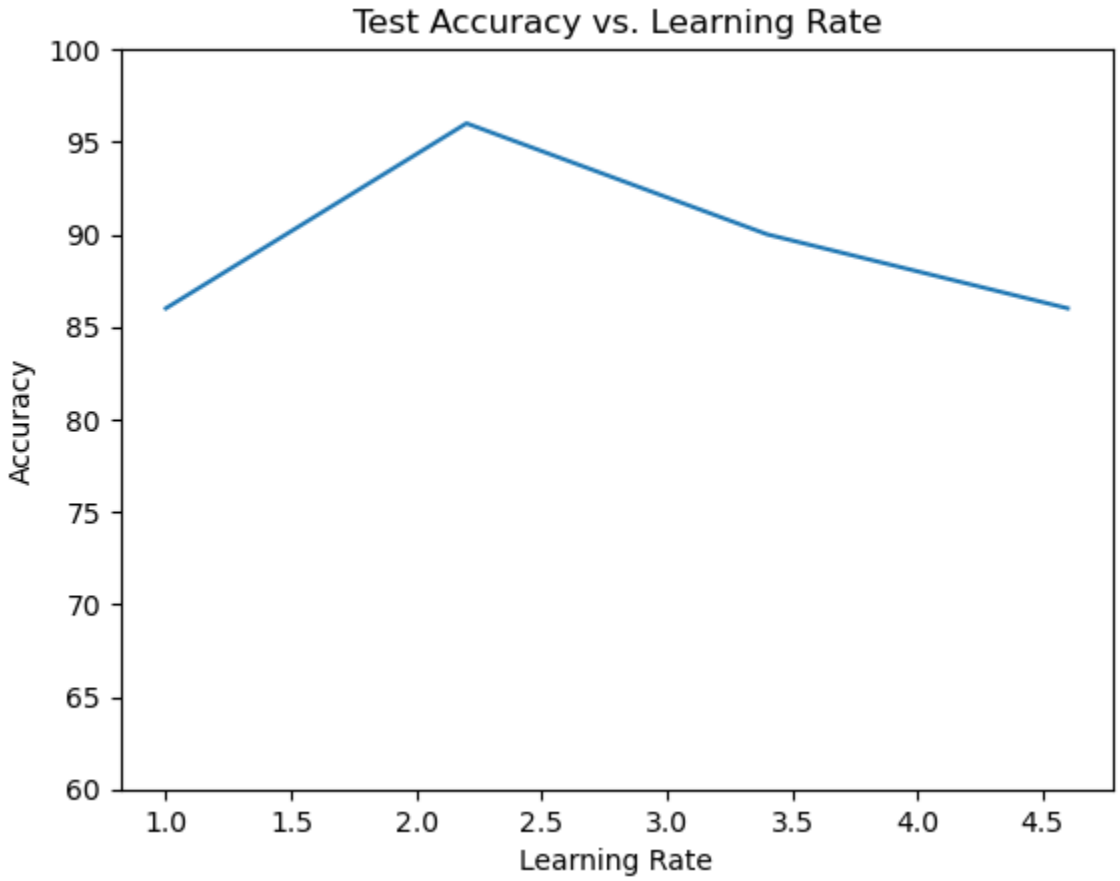
Learning rates plots (0.1 to 1):





Bigger learning rates plots:





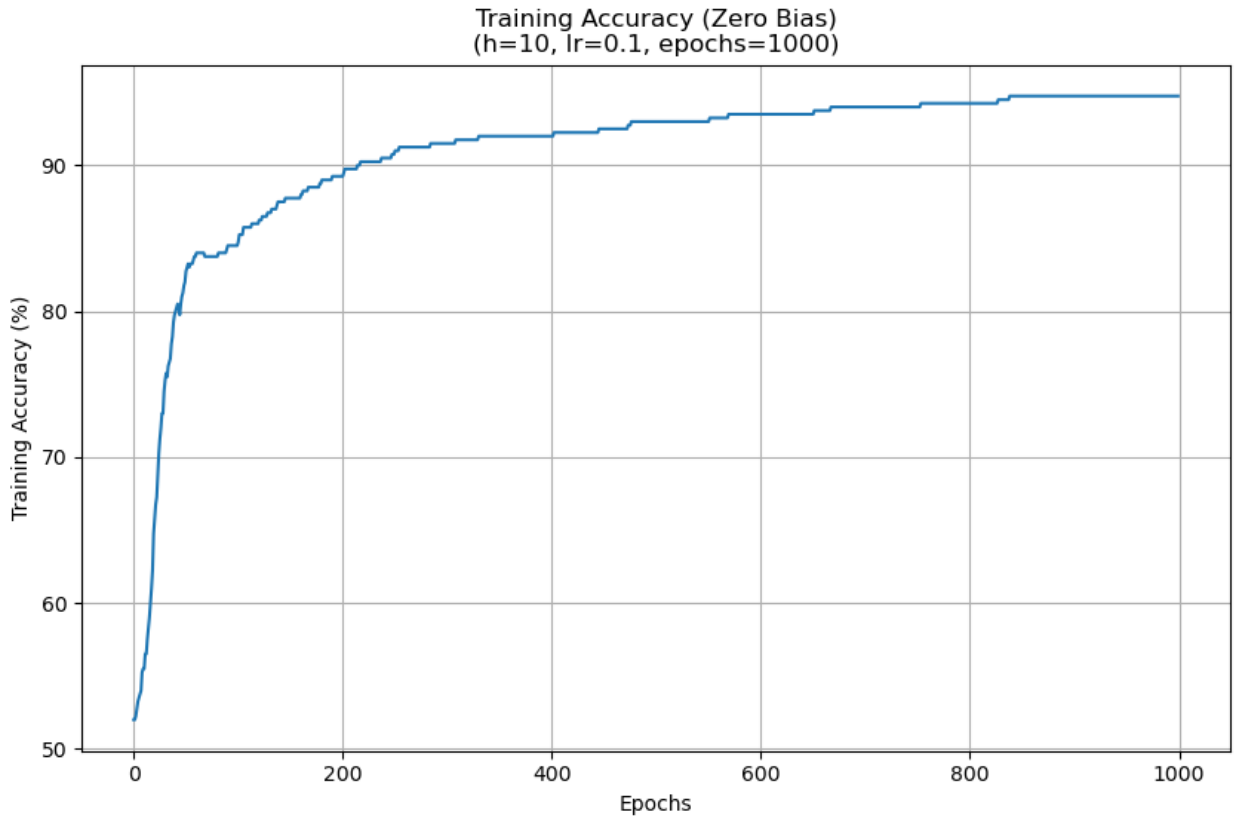
1.2 Impact of bias terms on the network performance

When the bias terms were set to zero, I did not see any significant change in the performance of the network. I think this is because the data is centered around the origin, so the bias term's ability to shift the decision boundary away from the origin to better fit the distribution of the data was not significant. Plots of the training accuracies, code outputs of the mean squared error and performance on the test1 dataset are also shown below.

Statistics of training data:

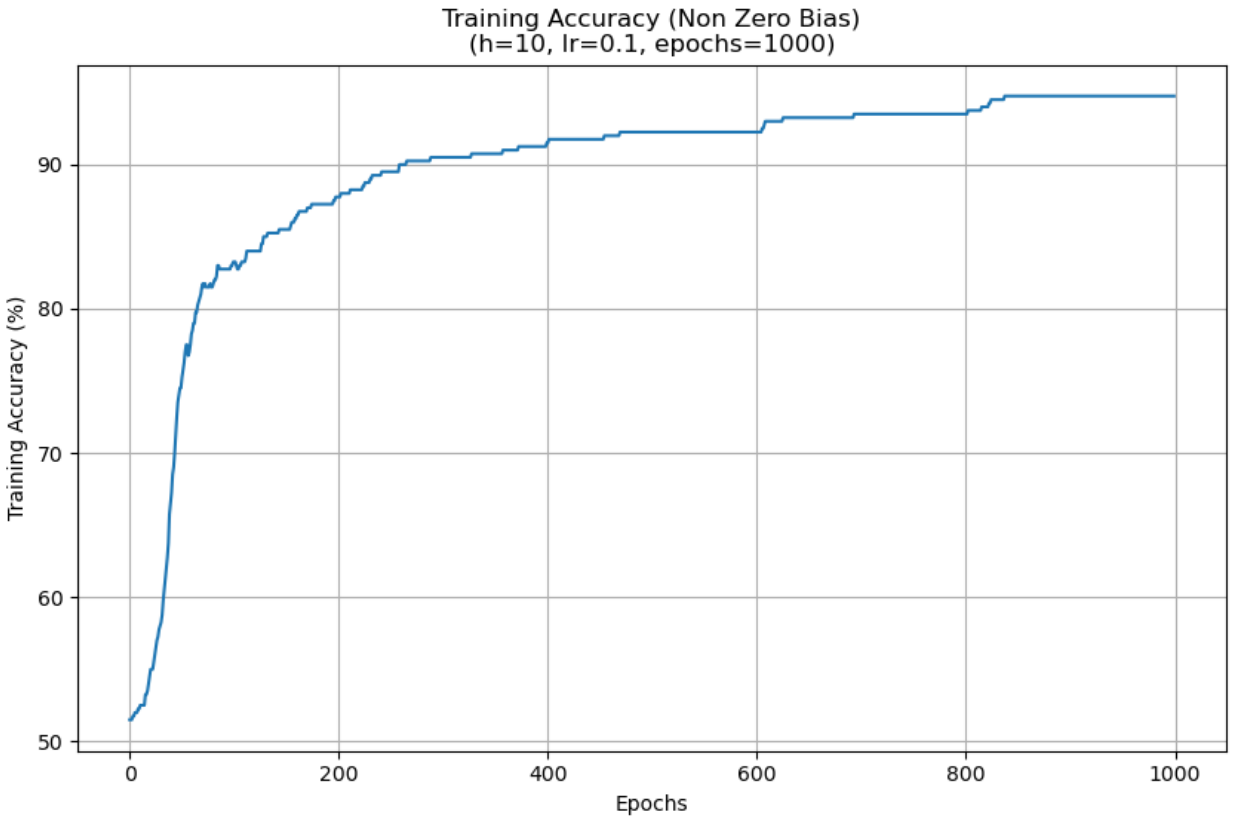
	x1	x2	x3	x4	x5
count	400.000000	400.000000	400.000000	400.000000	400.000000
mean	0.255725	-0.010525	0.432325	0.395400	-0.057250
std	0.559887	0.577314	0.641851	0.575964	0.576344
min	-0.990000	-0.990000	-1.000000	-1.000000	-1.000000
50%	0.400000	-0.020000	0.820000	0.670000	-0.070000
max	1.000000	0.970000	1.000000	1.000000	0.990000

Zero biases plot:

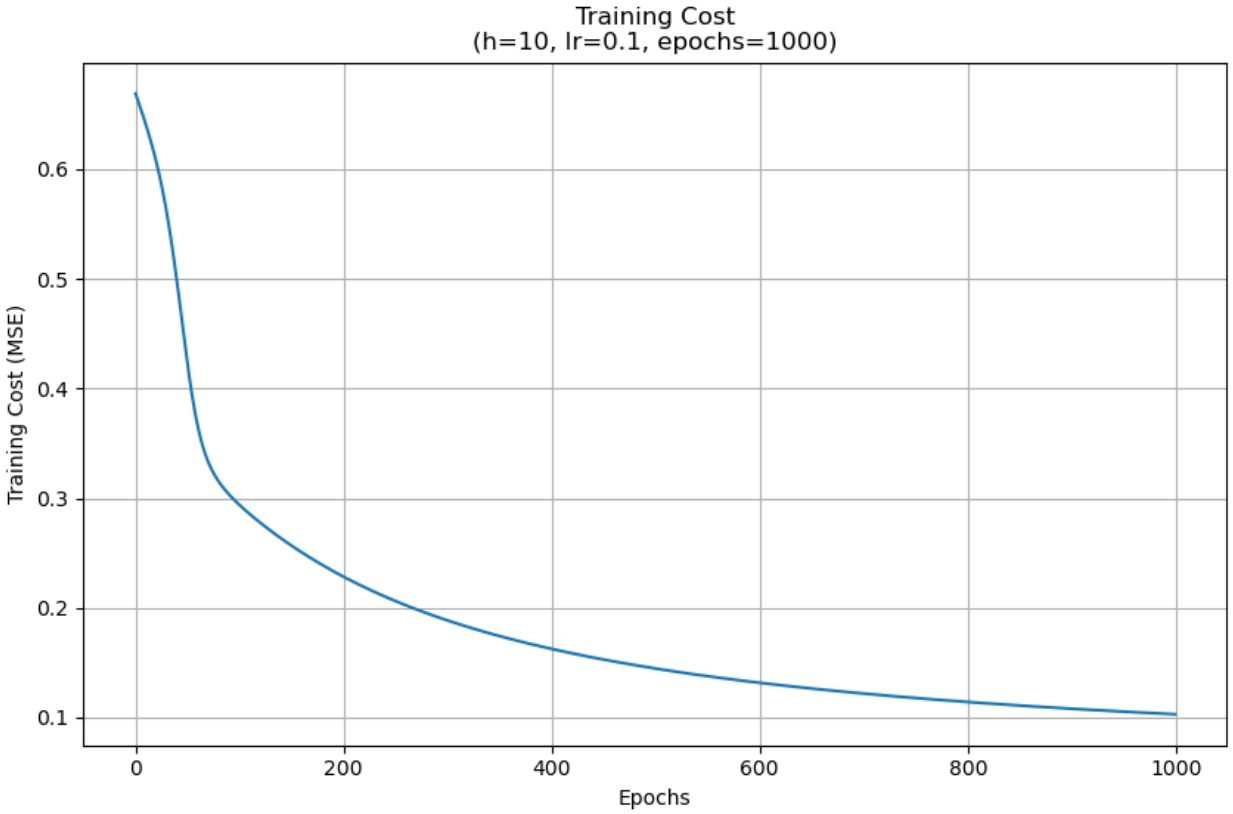



```
--- Starting Training, zero biases ---
Epoch 1/1000 | MSE: 0.590485 | Acc: 52.00%
Epoch 101/1000 | MSE: 0.257153 | Acc: 84.75%
Epoch 201/1000 | MSE: 0.200241 | Acc: 89.25%
Epoch 301/1000 | MSE: 0.167803 | Acc: 91.50%
Epoch 401/1000 | MSE: 0.147045 | Acc: 92.00%
Epoch 501/1000 | MSE: 0.132683 | Acc: 93.00%
Epoch 601/1000 | MSE: 0.122170 | Acc: 93.50%
Epoch 701/1000 | MSE: 0.114136 | Acc: 94.00%
Epoch 801/1000 | MSE: 0.107785 | Acc: 94.25%
Epoch 901/1000 | MSE: 0.102628 | Acc: 94.75%
Epoch 1000/1000 | MSE: 0.098389 | Acc: 94.75%

--- Testing on test1.csv data ---

Test Results -> Cost: 0.123750 | Accuracy: 92.00%
```

Random biases plot:




```
--- Starting Training, Non zero biases ---
Epoch 1/1000 | MSE: 0.668924 | Acc: 51.50%
Epoch 101/1000 | MSE: 0.293718 | Acc: 83.25%
Epoch 201/1000 | MSE: 0.228300 | Acc: 87.75%
Epoch 301/1000 | MSE: 0.188449 | Acc: 90.50%
Epoch 401/1000 | MSE: 0.162529 | Acc: 91.50%
Epoch 501/1000 | MSE: 0.144625 | Acc: 92.25%
Epoch 601/1000 | MSE: 0.131626 | Acc: 92.25%
Epoch 701/1000 | MSE: 0.121799 | Acc: 93.50%
Epoch 801/1000 | MSE: 0.114124 | Acc: 93.50%
Epoch 901/1000 | MSE: 0.107970 | Acc: 94.75%
Epoch 1000/1000 | MSE: 0.102978 | Acc: 94.75%

--- Testing on test1.csv data ---

Test Results -> Cost: 0.127157 | Accuracy: 92.00%
```

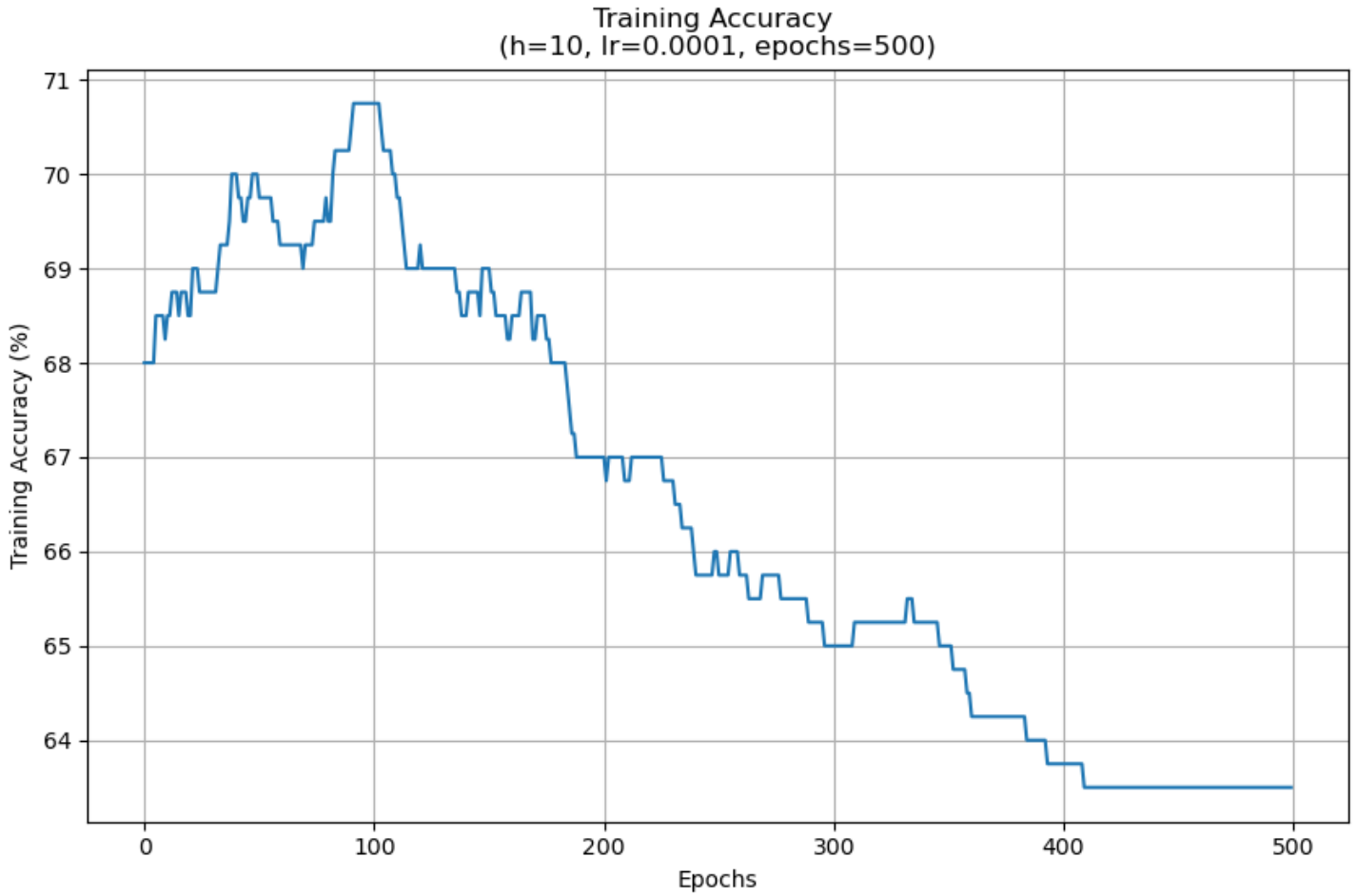


1.3 Impact of a linear activation function on the network performance

When I replaced the sigmoid function with an identity activation function f(x) = x, the network performed poorly (accuracy of 62%). This is because the network couldn't learn the non-linear boundaries separating the data and behaved as a simple linear classifier. A non-linear activation function enables the network to become a good function approximator and use more complex boundaries to separate the various classes of the data, ensuring good performance. Code output and the plot of the network training accuracy vs the number of epochs are shown below.

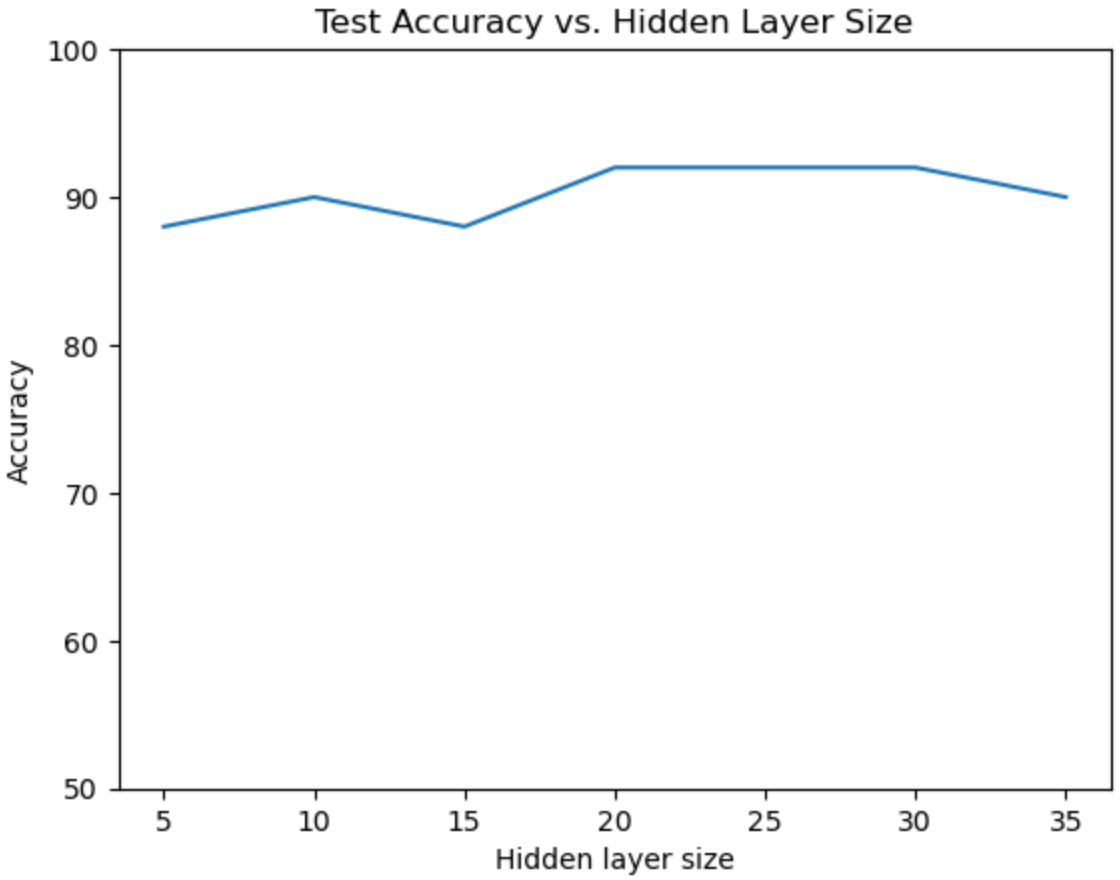
Performance with linear activation function

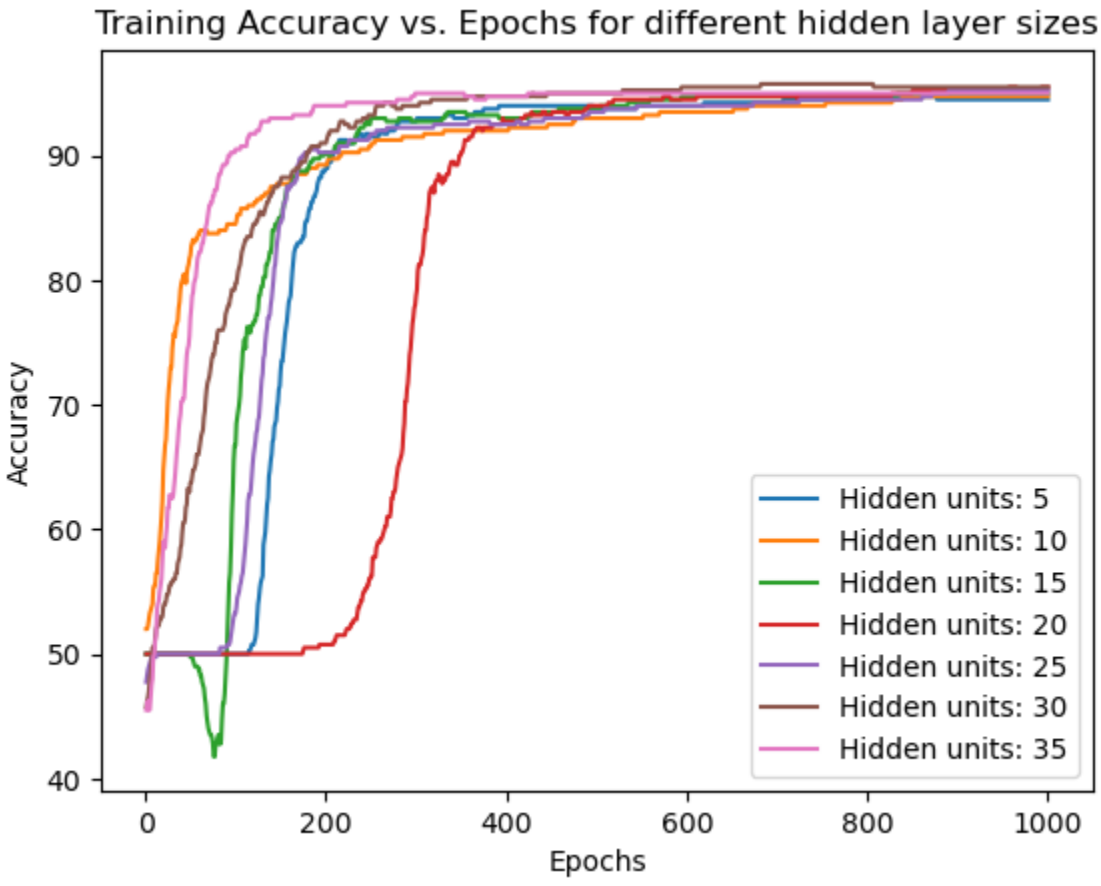
```
--- Starting Training ---
Epoch 1/500 | MSE: 89.377895 | Acc: 68.00%
Epoch 51/500 | MSE: 52.606932 | Acc: 69.75%
Epoch 101/500 | MSE: 37.146467 | Acc: 70.75%
Epoch 151/500 | MSE: 29.475183 | Acc: 69.00%
Epoch 201/500 | MSE: 24.985484 | Acc: 67.00%
Epoch 251/500 | MSE: 21.944954 | Acc: 65.75%
Epoch 301/500 | MSE: 19.652281 | Acc: 65.00%
Epoch 351/500 | MSE: 17.801178 | Acc: 65.00%
Epoch 401/500 | MSE: 16.245028 | Acc: 63.75%
Epoch 451/500 | MSE: 14.905499 | Acc: 63.50%
Epoch 500/500 | MSE: 13.757500 | Acc: 63.50%

--- Testing on test1.csv data ---

Test Results -> Cost: 17.082128 | Accuracy: 62.00%
```



1.4 Effect of different hidden units in the hidden layer on network performance

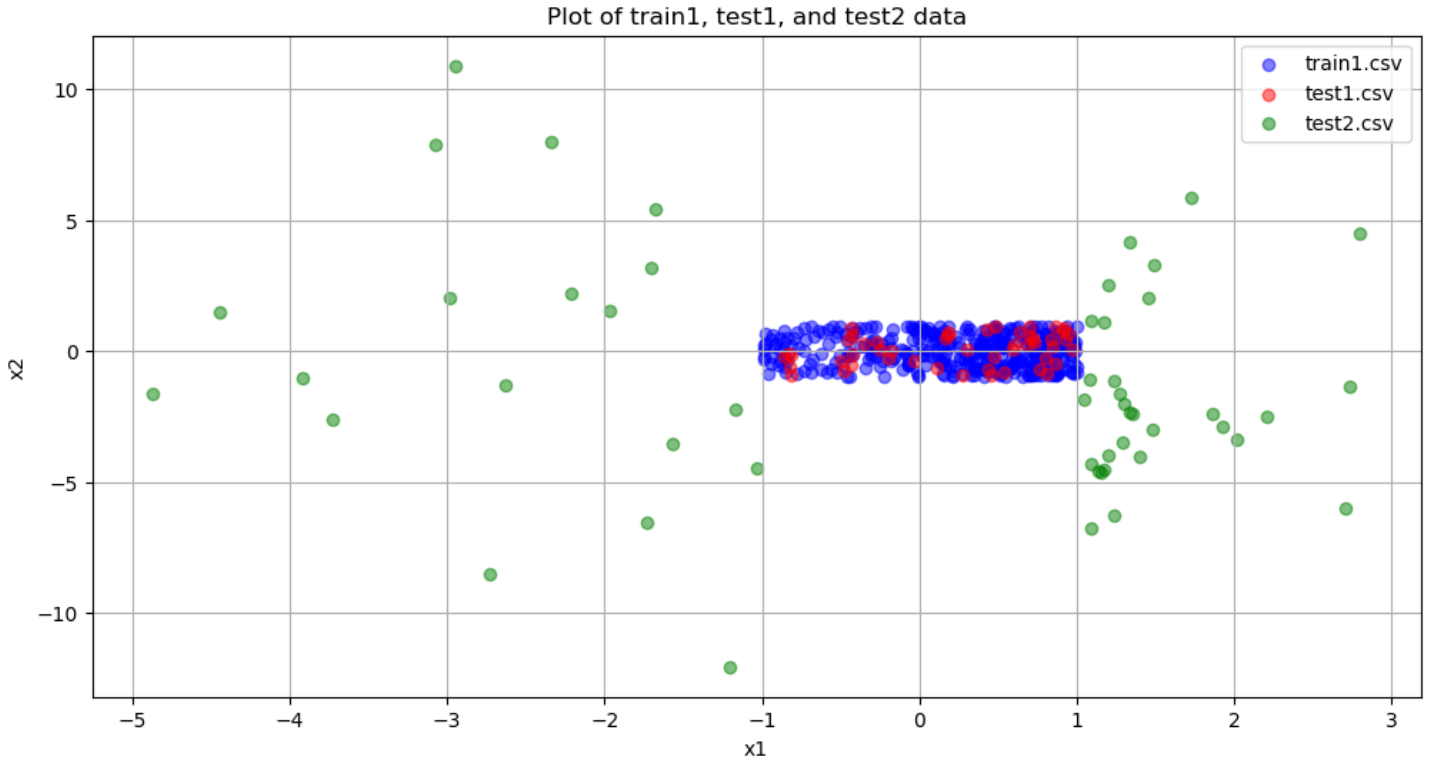
I plotted a graph showing the training accuracy over the epochs for various numbers of hidden units. With a higher number of hidden units, the network achieved higher accuracy in less training time. However, the performance of the network on the test data began to decline with the increased number of hidden units. This indicated that the model did not generalise well when too many hidden units were used. While adding more units increases the model's ability to learn complex functions, it also raises the risk of overfitting.


Test Accuracy vs. Hidden Layer Size
Accuracy
Hidden layer size
100
90
80
70
60
50
5
10
15
20
25
30
35


Training Accuracy vs. Epochs for different hidden layer sizes
Accuracy
Epochs
Hidden units: 5
Hidden units: 10
Hidden units: 15
Hidden units: 20
Hidden units: 25
Hidden units: 30
Hidden units: 35
90
80
70
60
50
40
0
200
400
600
800
1000

2. Performance of the network on test2 dataset.

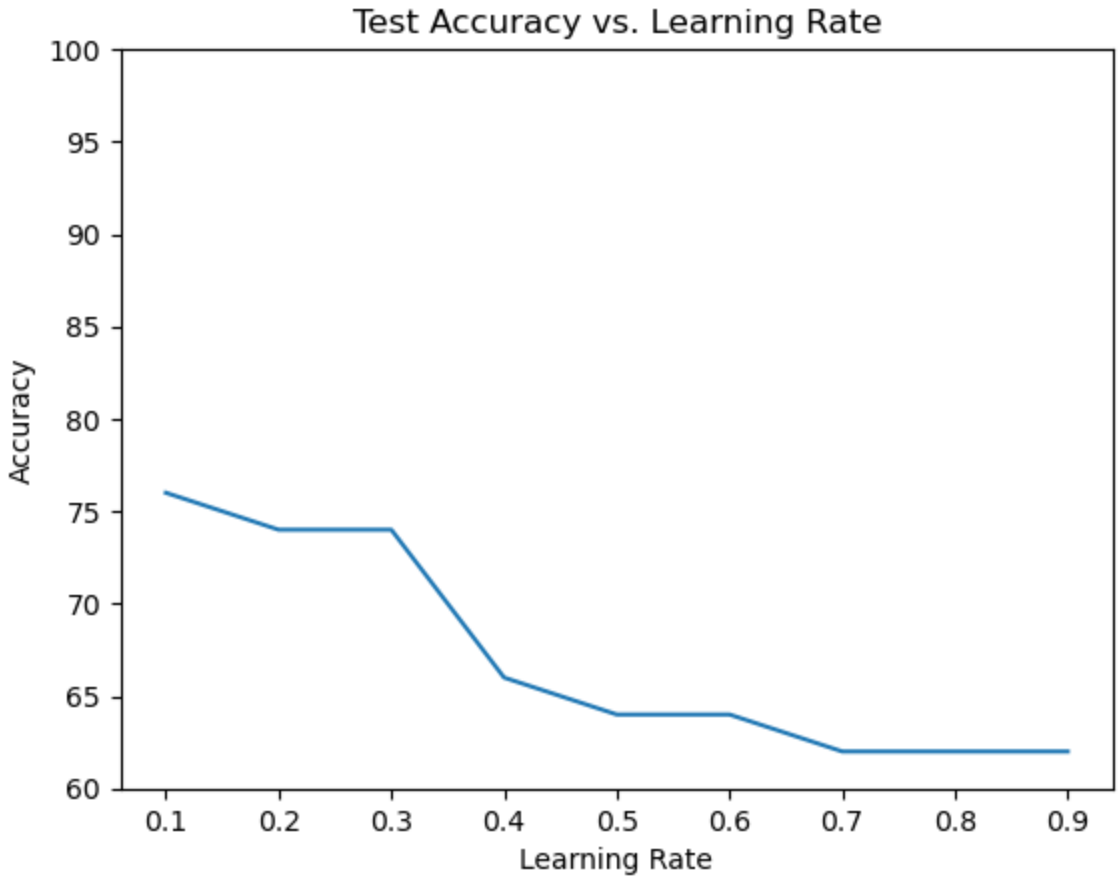
The data in test2 is distributed differently from the training data and test1 data, as shown in the plot below. This causes the neural network to perform poorly on the test2 data compared to the test1 data, as the network learned a function around the distribution of the training data (range for train1 is [-1, 1]). Most of test2 data is outside the range seen in the training data, resulting in lower performance.



2.1 Impact of different learning rates on the performance of the network.

The accuracy of the network decreased on test2 data as the learning rates were increased. This is because the network struggles to adapt to the distribution of test2 data. For higher learning rates, the gradient descent steps are bigger, leading to bad weights configuration and poor performance.

A plot of the network's performance on the dataset with different learning rates is shown below.



2.2 Impact of bias on the network performance

The network had a lower accuracy (74%) when trained with zero biases compared to random biases (76%). This is because the training data and test1 data were centered around the origin, so the bias terms were less important. However, test2 data is not centered around the origin , making non-zero bias terms important. The bias terms allow the model to shift the decision boundary away from the origin to better fit the distribution of the data, leading to slightly better performance.

Zero bias experiment.

```
--- Starting Training, zero biases ---
Epoch 1/1000 | MSE: 0.668924 | Acc: 51.50%
Epoch 101/1000 | MSE: 0.293718 | Acc: 83.25%
Epoch 201/1000 | MSE: 0.228300 | Acc: 87.75%
Epoch 301/1000 | MSE: 0.188449 | Acc: 90.50%
Epoch 401/1000 | MSE: 0.162529 | Acc: 91.50%
Epoch 501/1000 | MSE: 0.144625 | Acc: 92.25%
Epoch 601/1000 | MSE: 0.131626 | Acc: 92.25%
Epoch 701/1000 | MSE: 0.121799 | Acc: 93.50%
Epoch 801/1000 | MSE: 0.114124 | Acc: 93.50%
Epoch 901/1000 | MSE: 0.107970 | Acc: 94.75%
Epoch 1000/1000 | MSE: 0.102978 | Acc: 94.75%

--- Testing on test2.csv data ---

Test Results -> Cost: 0.383597 | Accuracy: 74.00%
```

Non zero bias experiment:

```
--- Starting Training ---
Epoch 1/1000 | MSE: 0.590485 | Acc: 52.00%
Epoch 101/1000 | MSE: 0.257153 | Acc: 84.75%
Epoch 201/1000 | MSE: 0.200241 | Acc: 89.25%
Epoch 301/1000 | MSE: 0.167803 | Acc: 91.50%
Epoch 401/1000 | MSE: 0.147045 | Acc: 92.00%
Epoch 501/1000 | MSE: 0.132683 | Acc: 93.00%
Epoch 601/1000 | MSE: 0.122170 | Acc: 93.50%
Epoch 701/1000 | MSE: 0.114136 | Acc: 94.00%
Epoch 801/1000 | MSE: 0.107785 | Acc: 94.25%
Epoch 901/1000 | MSE: 0.102628 | Acc: 94.75%
Epoch 1000/1000 | MSE: 0.098389 | Acc: 94.75%

--- Testing on test2.csv data ---

Test Results -> Cost: 0.384124 | Accuracy: 76.00%
```

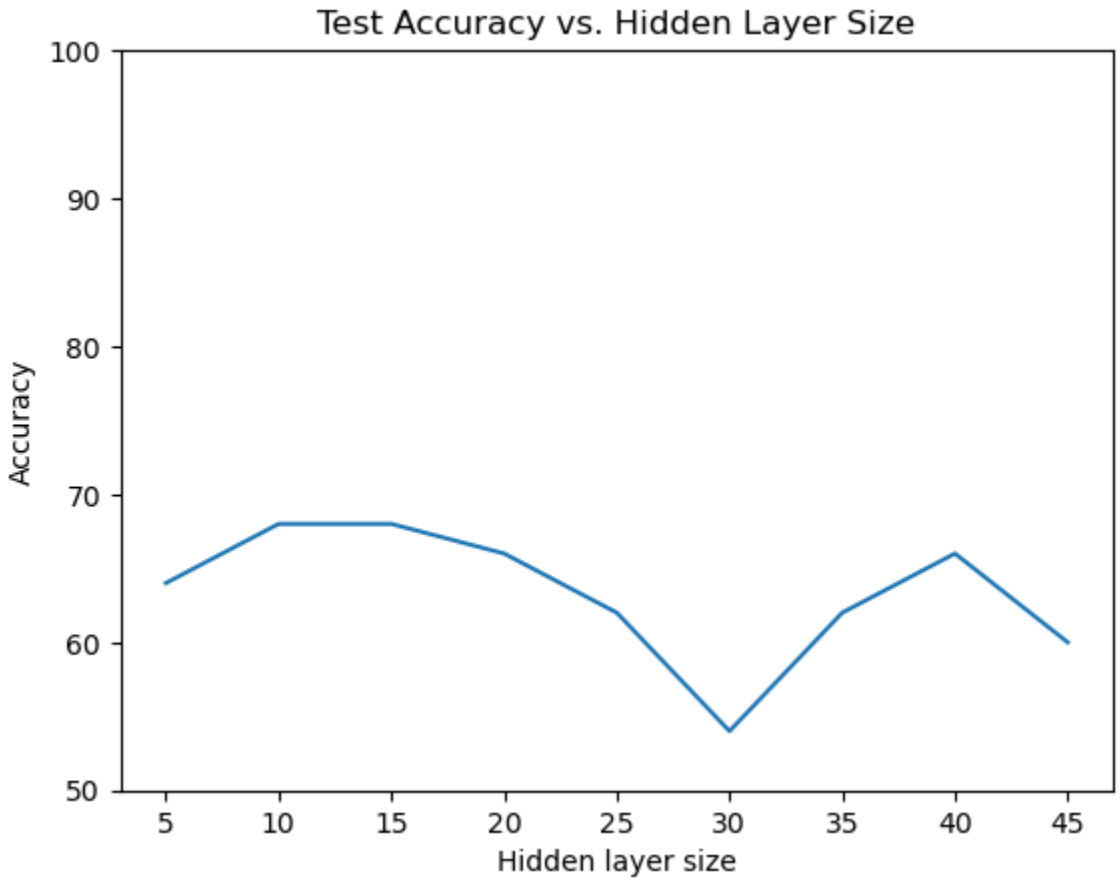
2.3 Impact of a linear activation function on the network performance

The network performed better on test2 data with the linear activation function as compared to test1 data. This is because of the wider distribution of the test2 data set.

```
--- Starting Training ---
Epoch 1/500 | MSE: 89.377895 | Acc: 68.00%
Epoch 51/500 | MSE: 0.628810 | Acc: 74.75%
Epoch 101/500 | MSE: 0.208711 | Acc: 89.75%
Epoch 151/500 | MSE: 0.149011 | Acc: 93.75%
Epoch 201/500 | MSE: 0.137882 | Acc: 95.25%
Epoch 251/500 | MSE: 0.135698 | Acc: 95.25%
Epoch 301/500 | MSE: 0.135266 | Acc: 95.25%
Epoch 351/500 | MSE: 0.135180 | Acc: 95.25%
Epoch 401/500 | MSE: 0.135163 | Acc: 95.25%
Epoch 451/500 | MSE: 0.135160 | Acc: 95.25%
Epoch 500/500 | MSE: 0.135159 | Acc: 95.25%

--- Testing on test2.csv data ---

Test Results -> Cost: 14.505174 | Accuracy: 66.00%
```

2.4 Different number of hidden units on the network performance

The plot below shows that the network performed well when the number of hidden units was between 10 and 15, but its performance started to decrease as the number of hidden units increased. This indicates that as more layers are added to the network, overfitting occurs, and the network fails to generalise well.



Code:

My code is attached as a separate file

AI Usage:

I used Copilot to write these functions: generate_plot() and save_model_parameters()