CMSC421 Assignment One Neural Networks

George Witt

November 2023

1 Introduction

In the following sections, I lay out my text responses for questions 1 through 3. I default to reporting the figures only for the successful run (I report the tested hyperparameters in text). This is to save space in this document; I do not wish the report document to be tens of pages long. Overall, I report a significant increasing difficulty in determining optimal hyperparameters for more complex problems due to the increasing functional complexity as well as increasing computation time and number of tunable parameters. I note that I complete this work on a personal Macbook, meaning that many of my runs took a significant amount of time. This implies many of my hyperparameter searches are inherently less extensive than those that could be run on a NVIDIA GPU machine, or cluster, though I do still feel my searches to be fairly effective at determining useful hyperparameters.

I used the new codebase to present these results.

Please note all images are also provided in the zip file itself! Separate images for my own separate hyperparameter sweeps are NOT provided in the zip file to save space.

Please note that all graphs are presented in the document as they appear, but quick notes on the results from the figures are given in the text. Please use both the figures and the text to read results. The figures may show up at slower rates than the text as the figures are large and take up fair chunks of the available space.

2 Question 1

I filled out the code as instructed in the zip file. I present my results in figures below.

2.1 Hyperparameter Search

2.1.1 Part A

I achieved these results by sweeping through learning rates of 0.03, 0.3, 0.6, 0.9, and 1.0 for iterations from 10, 100. I used 10 to validate initial ideas then 100

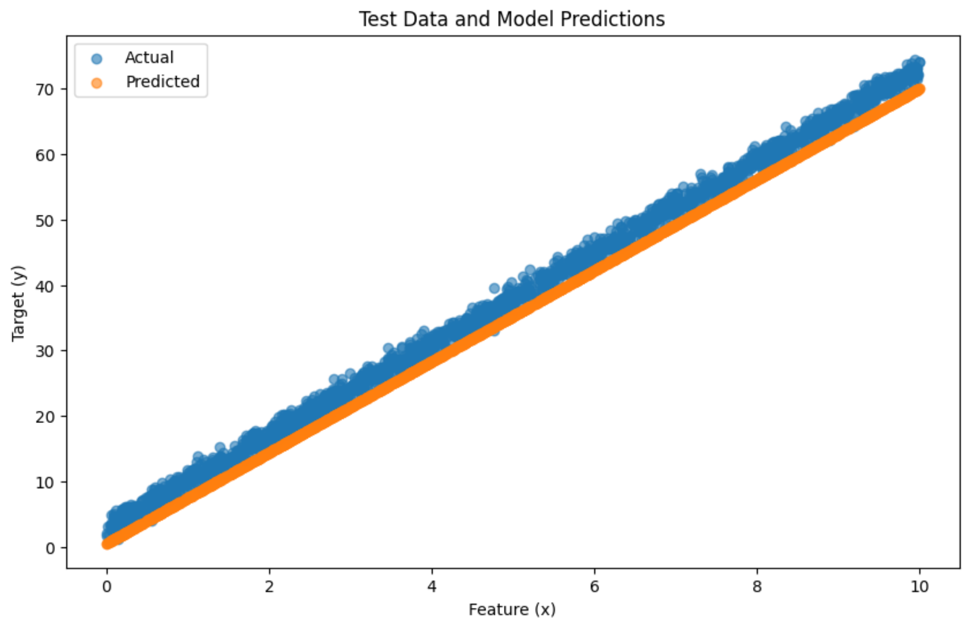
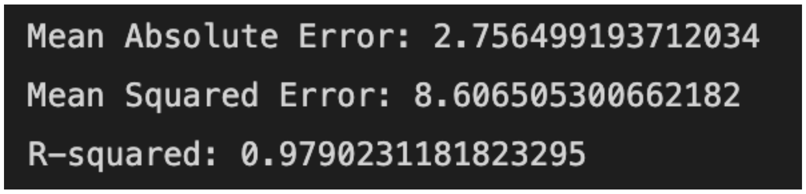


Figure 1: 1A Linear Best Fit (see hyperparameters in text)



```
Mean Absolute Error: 2.756499193712034
Mean Squared Error: 8.606505300662182
R-squared: 0.9790231181823295
```

Figure 2: 1A Linear Best Fit evaluation metrics (see hyperparameters in text)

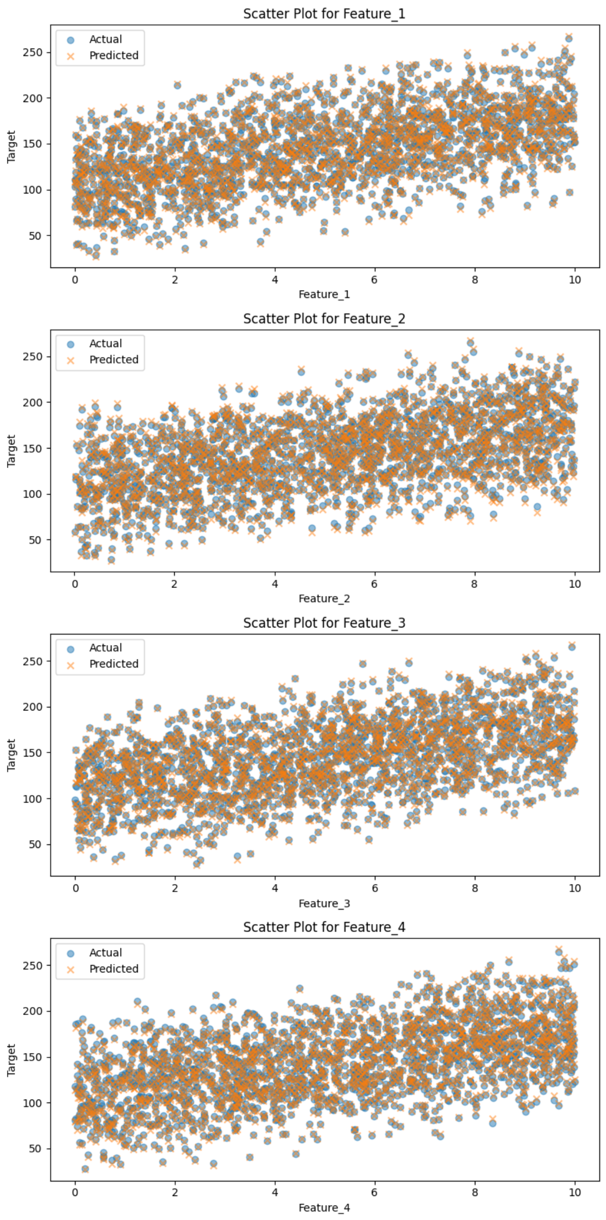


Figure 3: 1B Linear Best Fit Feature by Feature performance (see hyperparameters in text)

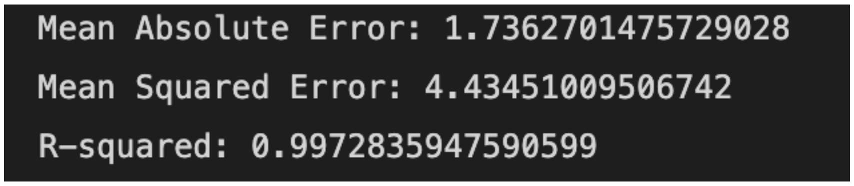

```
Mean Absolute Error: 1.7362701475729028
Mean Squared Error: 4.43451009506742
R-squared: 0.9972835947590599
```

Figure 4: 1B Linear Best Fit evaluation metrics (see hyperparameters in text)

for the final fit. The results always increasingly became closer to these final results, leading to my decision to run a learning rate of 1.0 for as long as I could (1000 iterations). This achieved a near perfect fit.

For this particular problem with a very narrow focus determining the best hyperparameter set was not challenging. This is because the only effective hyperparameter I need to compare models with is the learning rate (length of training just affects final strength of fit). However, this problem is only a simple linear problem, so if the model will fit at all then the choice of hyperparameters only affects how long I will have to wait for the fit to complete.

2.1.2 Part B

I reused the parameters from part (a) for search in part b. I found the same parameters, 100 iterations and 1.0 learning rate, to be most effective. This is again because this is a linear problem, implying that if the data is fitable almost any hyperparameter combination will only affect how long it takes to fit the data. In a few sensitive cases, learning rate may have an effect (if the fit were to be unstable, say), but this is not one of those cases. If the data is not fittable, learning rate will or iteration time will not make a difference. My results achieve, again, a near perfect fit with an R^2 of 0.997 (1.0 representing a perfect positive linear correlation, of course).

Solving the multi-dimensional problem was more complex in terms of run time, but again the hyperparameter search here was fairly simple due to the simplistic nature of the model that we are fitting.

3 Question 2

3.1 Hyperparameter Search

3.1.1 Part A

The hyperparameter search in this question was very challenging. Not only can we tune the learning rate and iterations, but also the number of neurons. We may also play with the activation function; the question specifies to use ReLU, but out of curiosity I tested both ReLU and Sigmoid activation functions. In the end I found my best results with 1024 neurons trained for 5000 iterations with a learning rate of 0.02. I tried variations of parameters in a pattern with

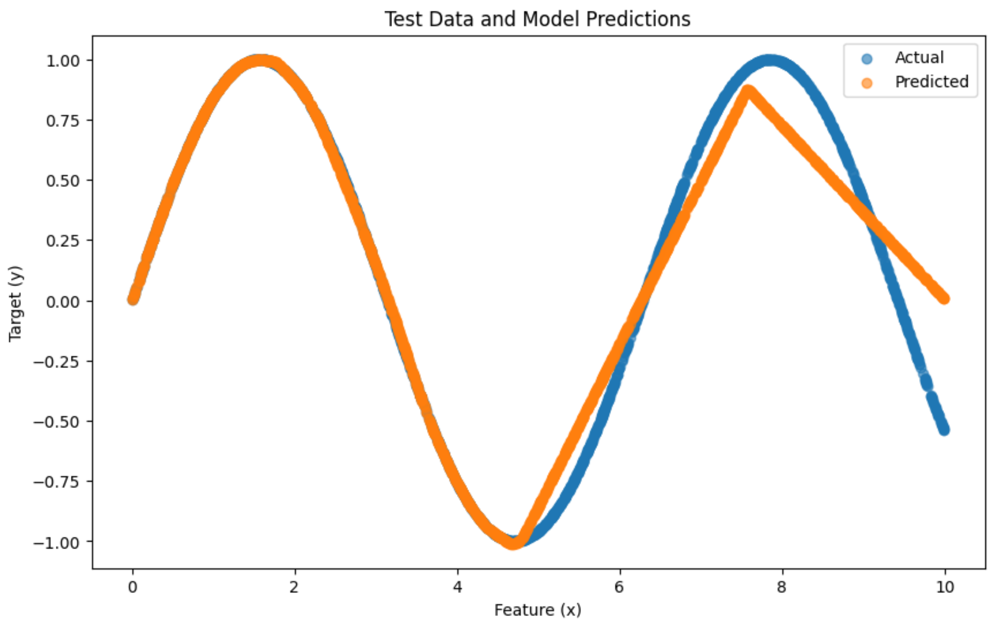


Figure 5: 2A Best Fit long run (see hyperparameters in text). Receives an R^2 of 0.955.

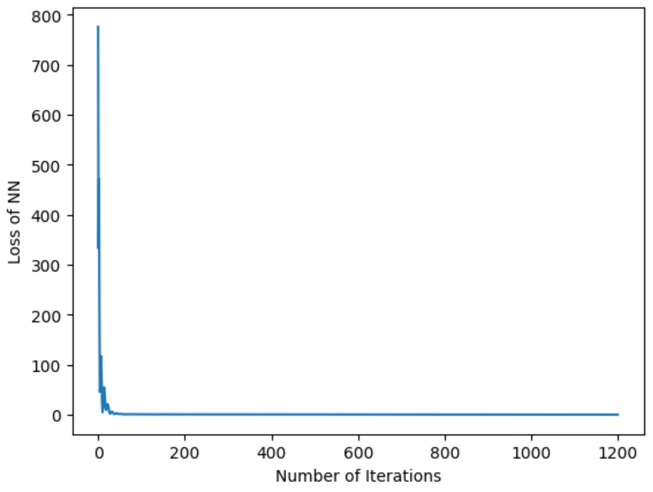
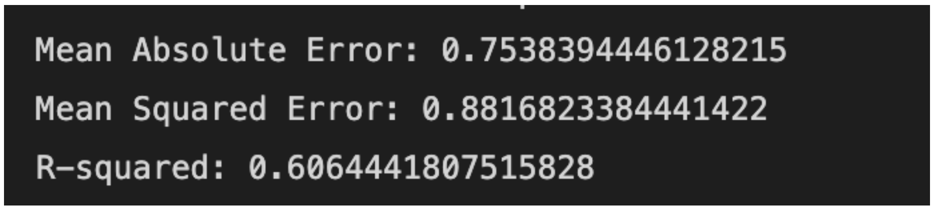


Figure 6: 2B loss plot over time (square loss) (see hyperparameters in text)



```
Mean Absolute Error: 0.7538394446128215
Mean Squared Error: 0.8816823384441422
R-squared: 0.6064441807515828
```

Figure 7: 2B evaluation metrics for best fit model.

learning rates of 1, 0.3, and 0.03 for both ReLU and Sigmoid for 100, 200, and 1000 iterations. In the end, I adjusted these hyperparameters to the reported final values, and achieved reasonable performance that fits the given expected results with an R^2 of 0.955.

The difficulty with fitting these hyperparameters is that the number of possible combinations increases $O(\alpha^N)$ where α represents the number of desired possible values for a given hyperparameter and N represents the number of hyperparameters. Therefore, this section took a significant amount of time. Especially as training for only 100 iterations, say, could lead to an R^2 of 0.7, whereas 1000 iterations would lead to an R^2 closer to 0.9. Moreover, the learning rate is not a simple thing to assign to 1.0, as higher learning rates can be very unstable in multi-dimensional weight space. Therefore, it takes time to validate each combination, and each combination is not simple to predict. This is the challenge with tuning neural networks.

3.1.2 Part B

Solving the multi-dimensional case was significantly more challenging, however. This part was perhaps the most challenging in the assignment. Overall, I found my best performance 1200 iterations and a learning rate of 0.02. It achieved an R^2 of only 0.61. This is still a decent linear fit (linear here in functional space, of course, the model is a neural network and therefore may be non-linear), but a far less effective one than in the other cases. Moreover, the final output seems to be highly accurate but concentrated tightly around the centerline of the data for the different features.

I tuned with the same parameters as Part A's case, as well as a 0.02 learning rate variation with iterations of 100, 500, 900, and 1200. Overall, I report the 1200 iteration case for brevity as it was the best performing. The 2D problem was far more difficult to solve as it required me not only sweeping through more parameters, but also using more time for computation. This implies that the sensitivity of a hyperparameter sweep will grow greatly with the dimensionality of the problem. Since higher dimensional problems inherently scale the number of parameters in a neural network (number of input parameters scales), I therefore can make the assumption that higher parameter neural networks are likely very sensitive to small adjustments in their hyperparameters.

So while this fit isn't perfect, it is still a fairly strong fit given the instability

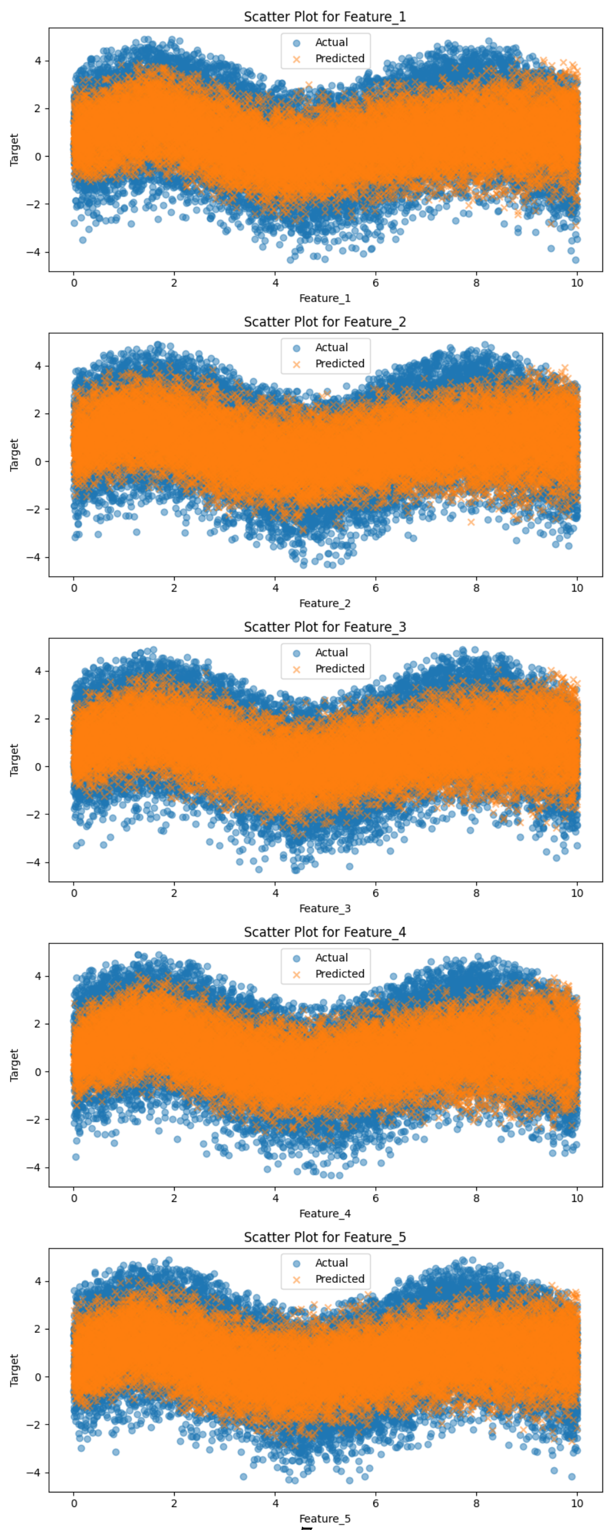


Figure 8: 2B output feature by feature (see hyperparameters in text)

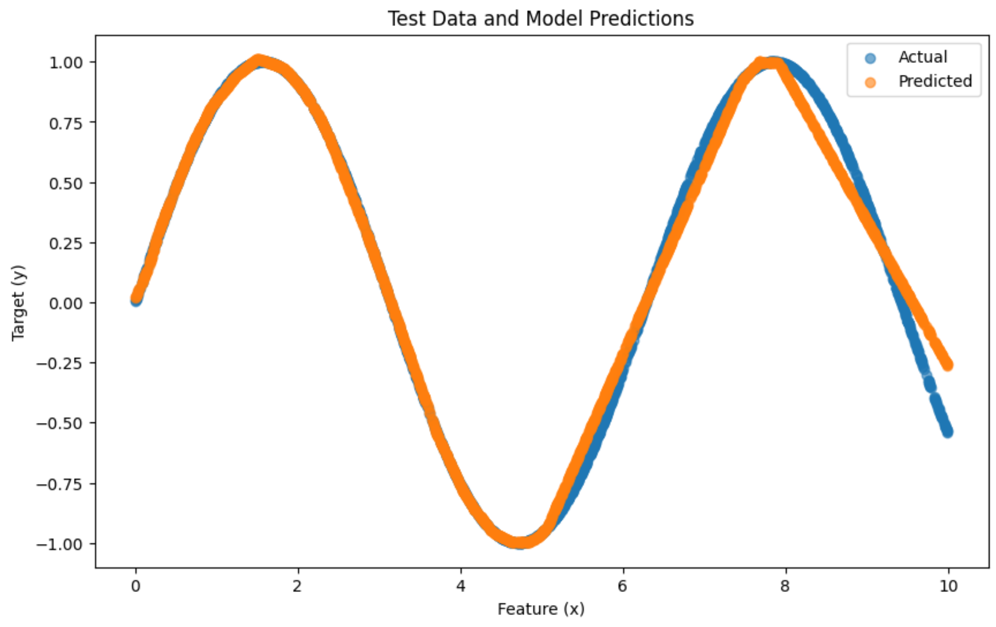


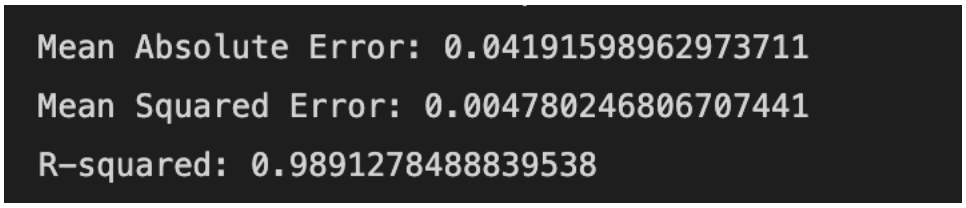
Figure 9: 3A output for the best fit model (see hyperparameters in text)

in the search for hyperparameters as mentioned above.

4 Question 3

4.1 Hyperparameter Search

The deep neural network fit was once again incredibly good in the 1D case as the hyperparameter search is simpler than in the 2D case. I got an R^2 of 0.99. This did take significant sweeping between parameters like in the second case, especially considering the fact that now each layer can vary in size as well. For simplicity, I kept the size of each layer the same and performed the same



```
Mean Absolute Error: 0.04191598962973711
Mean Squared Error: 0.004780246806707441
R-squared: 0.9891278488839538
```

Figure 10: 3A evaluation metrics for the best fit model (see hyperparameters in text)

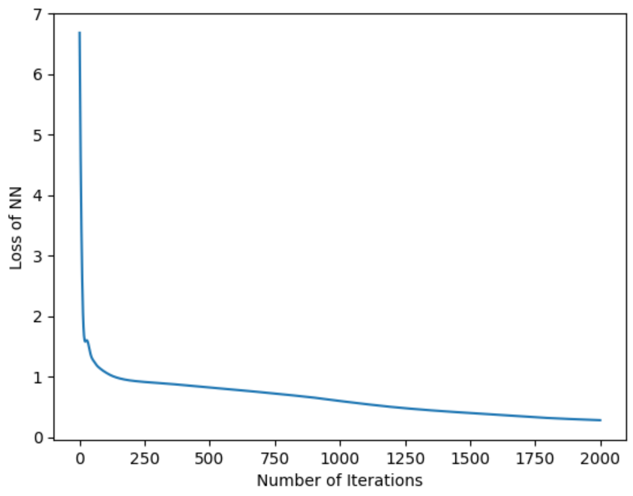


Figure 11: 3B Loss over time, observe this as evidence of the convergence speed mentioned in the text. (see hyperparameters in text)

sweeps as before. My hyperparameter search however was **incredibly** limited by available memory on-board my laptop in the case of the multi-dimensional data. My best hyperparameters were found as 2000 iterations with a learning rate of 0.0003 on a ReLU 2 layer 64 x 64 neuron network.

I did the same thing for the multi-dimensional model, and swept hyperparameters for the layer size from 32 to 64 (2 layers), as around 128 and above units for each layer (or increasing the number of layers), led to visual studio crashing on my small personal machine. I observe that the hyperparameter search becomes significantly more difficult, quicker, by $O(\alpha^N)$ as the problems and functional approximators becomes more complex; please see the earlier description of $O(\alpha^N)$. The multi-dimensional case, due to this resource-limited hyperparameter search only achieved an R^2 of approximately 0.74. This still indicates a strong fit of the model, but like before the model is overfitting its results to the centerline of the trend in the features. I found the best results with the same hyperparameters as in part a.

4.2 Speed of Convergence

Considering the provided loss plots, I do still note that these models converge fairly quickly. I refer to the loss plots for 3B as shown in the document's figures for evidence of this convergence speed, compared to similar loss plots provided in other figures. However the quality of solution that the final model converges to seems to vary strongly by the chosen hyperparameters. As a result, even though a model may converge for a given problem quickly, it will not necessarily perform well unless it is a complex enough model. Sadly, with the resource-limitations

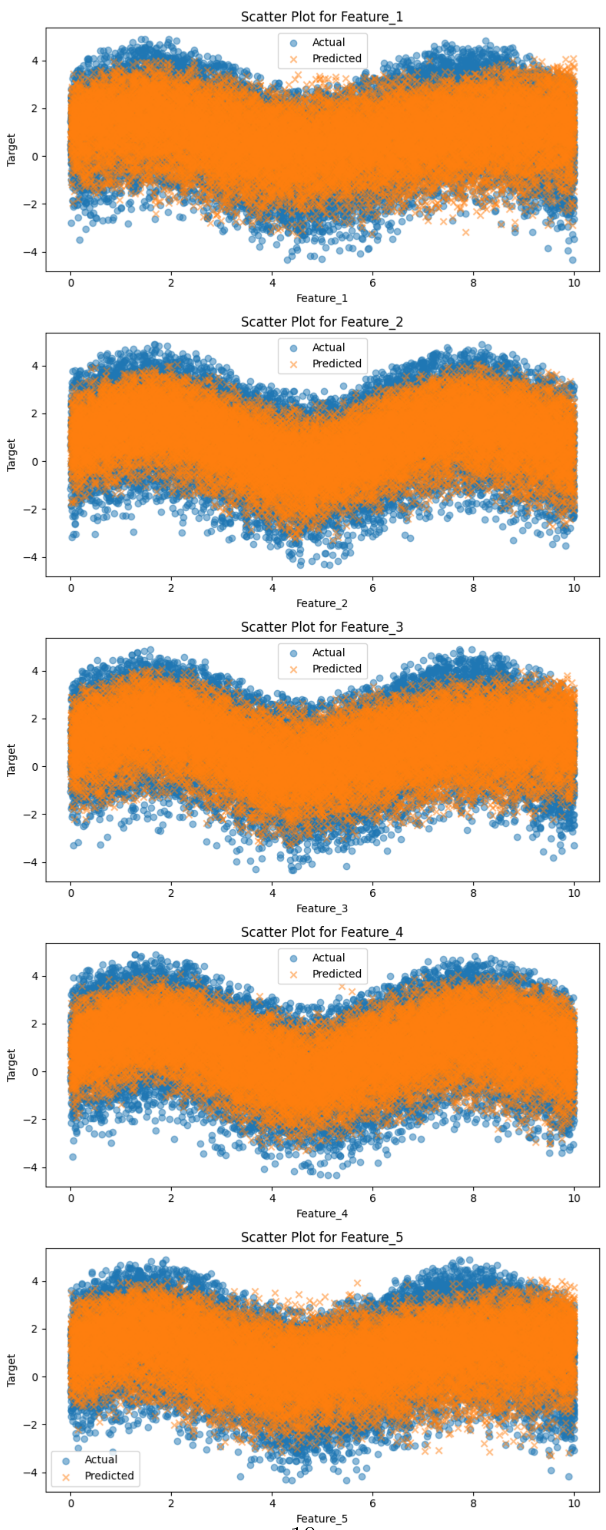
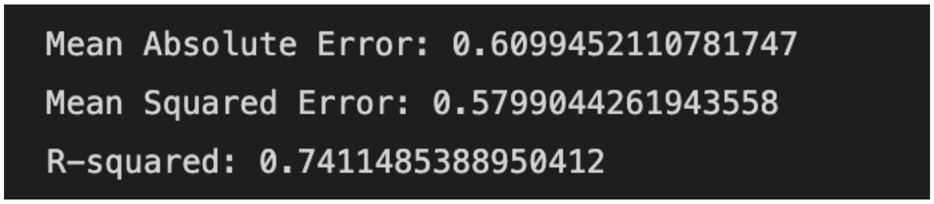


Figure 12: 3B output feature by feature. Observe that the performance is limited here by network limitations and training time limitations, however the final model still achieves a final R^2 of approximately 0.74. (see hyperparameters in text)



```
Mean Absolute Error: 0.6099452110781747
Mean Squared Error: 0.5799044261943558
R-squared: 0.7411485388950412
```

Figure 13: 3B evaluation metrics for the best fit model.

I experience on my older personal laptop, I am unable to perform a far-more extensive hyperparameter grid search to determine the optimal model. I am limited especially (as mentioned) in the number of neurons and layers I can support, as well as the time I cna leave the models running (2000-3000 iterations is already a significant chunk of time).