Brody Montag

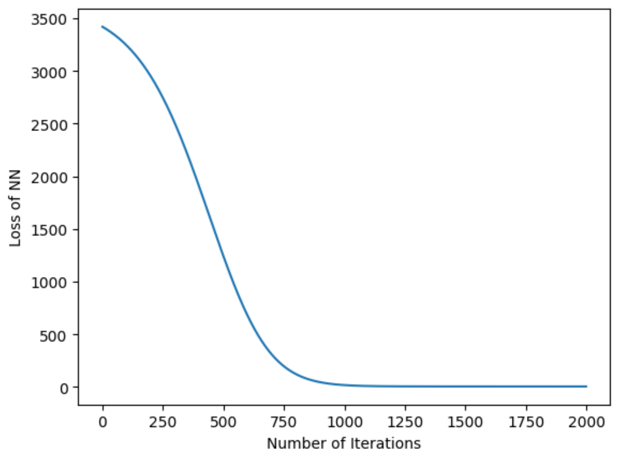
November 19th, 2023

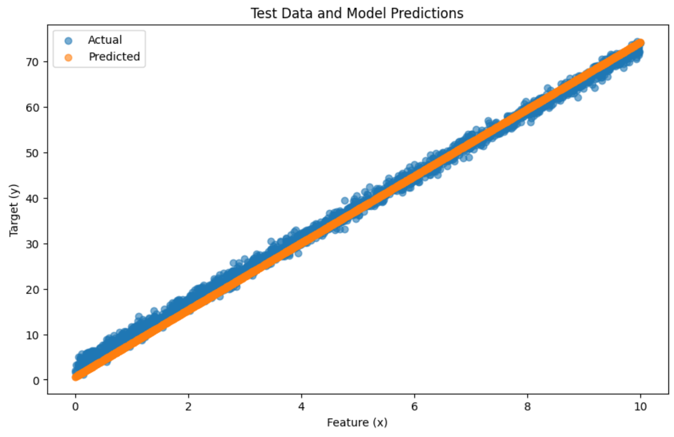
CMSC 421

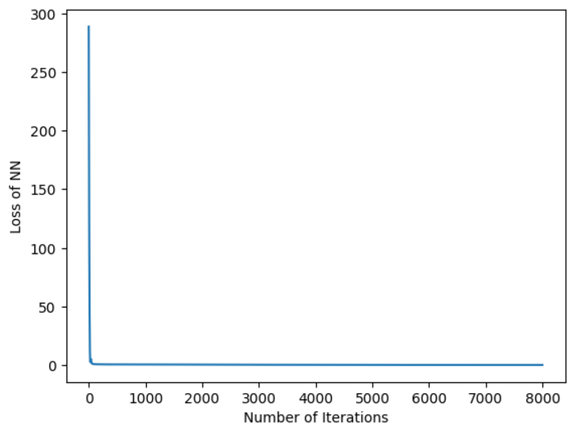
Convolutional Neural Networks Report

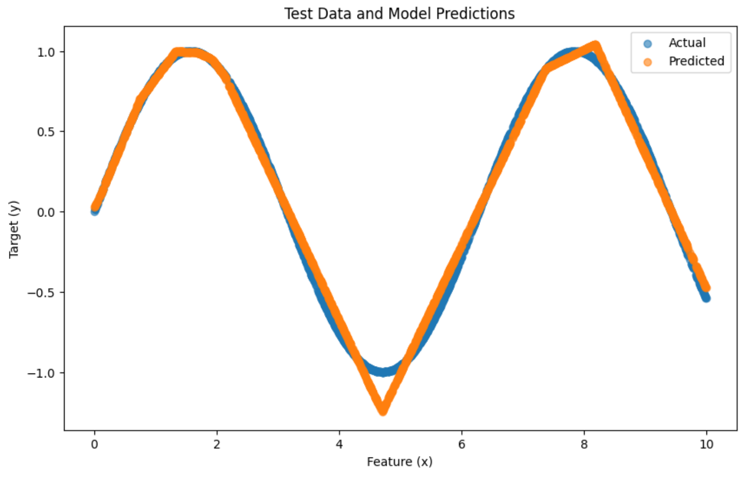
1.4 - Discuss the difficulty level in finding an appropriate set of hyperparameters:

After getting the models for q1_a and q1_b running, I really enjoyed tuning hyperparameters since it was a working representation of some of the concepts we've looked at in lectures so far. Because these were relatively basic network architectures, there weren't a lot of moving parts to work with. The main challenge was finding the ideal values for step size and number of iterations so I used an iterative guess-and-check approach to optimizing values. After experimenting with a few step sizes I settled on 0.1 which helped inform the number of iterations to choose based on when the model was converging on 0.1 step size. Even after finding some value that I perceived as optimal (eg. 0.02), I thought it was important to explore the possibility that it was a local minima, and there was another step value that would more efficiently converge on the dataset.



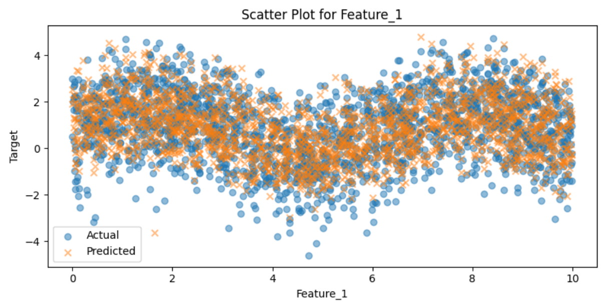


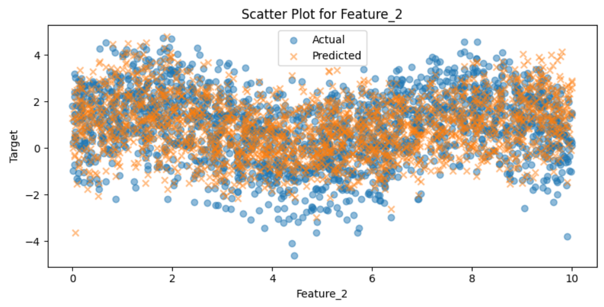


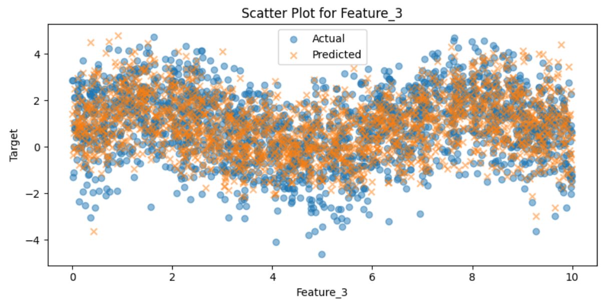


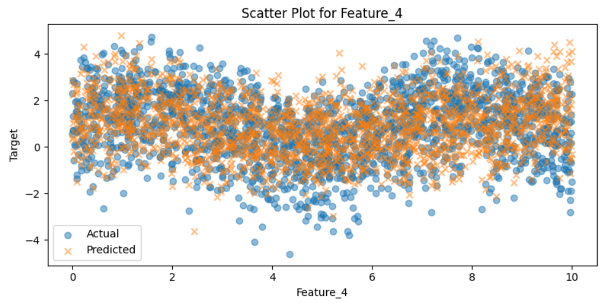
2.4 - Discuss the difficulty in finding an appropriate set of hyperparameters and compare the difficulty level between solving the 1D problem and the higher-dimensional problem.

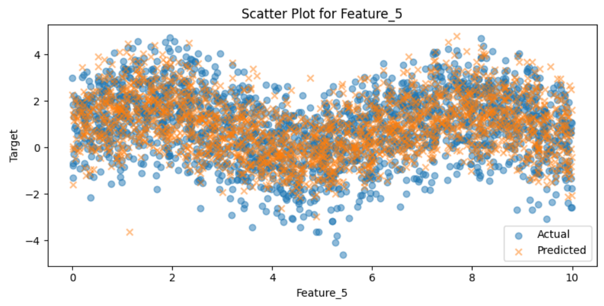
Compared to the 1D problem, much more intensive tweaking was necessary since the number of training iterations greatly affects model performance. Too few iterations did not allow the model to converge, leading to underfitting. Conversely, too many iterations led to overfitting, where the model captures noise. Further I found that if the learning rate is too high or low, the model may respectively overshoot the optimal solution or take too long to converge. In the 1D problem, the feature space is much simpler, with only one variable influencing the output, thus capturing the underlying data patterns is a less sensitive process. As the dimensionality increases, the complexity of the feature space grows exponentially. This increased complexity means more intricate interactions between features, which begets a more robust model to capture these relationships effectively. As such, I had more hyperparameters to work with – meaning fine tuning these values was a more involved process than in previous problems. For example, the inclusion and subsequent adjustment of the number of hidden units greatly affected the model's ability to generalize to new data. One unexpected challenge I encountered was runtime for training the model. The first few models with 1D data only took a few minutes to train, but with so much training data and computations to be processed with multiple dimensions, this training ended up taking roughly 8 hours each epoch. This wasn't really something I could get around since the performance was greatly affected by the number of iterations, thus I landed on 10,000 iterations in my final approach.

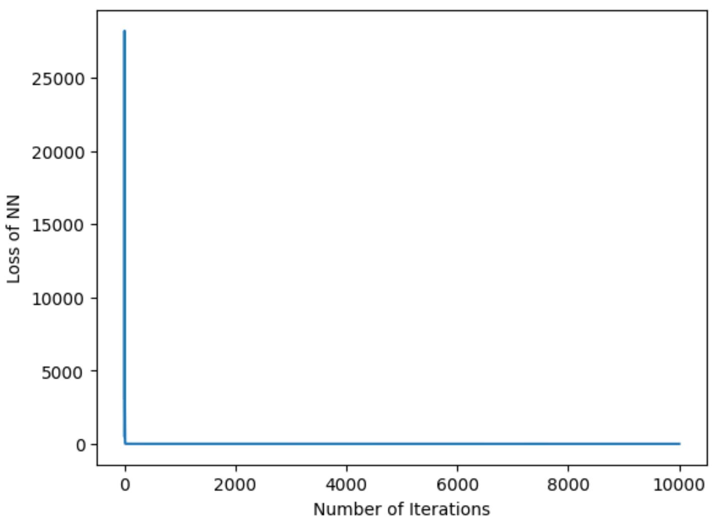






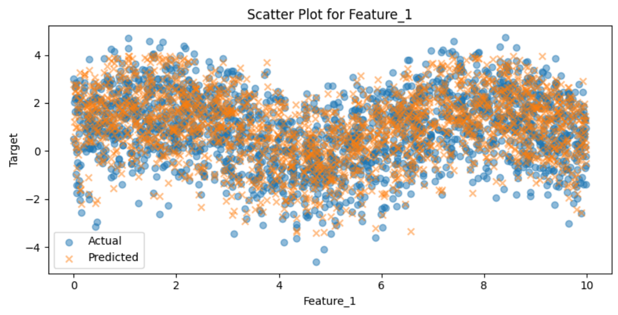


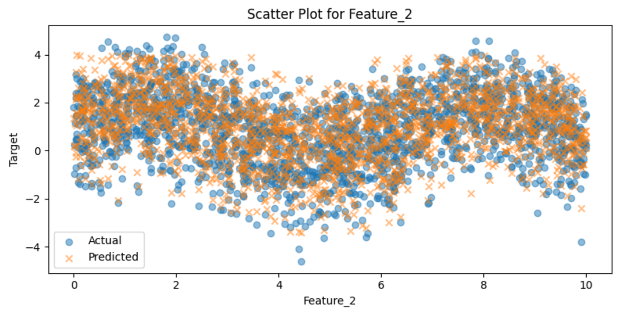


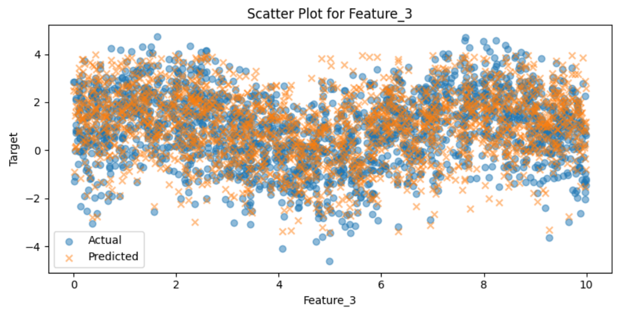


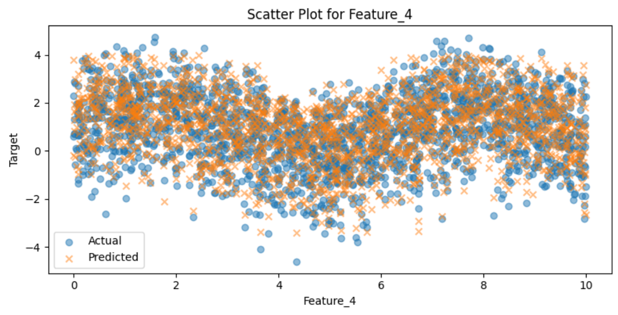
3.4 - Conduct and report on experiments to determine whether the depth of a network has any significant effect on how quickly your network can converge to a good solution. Include at least one

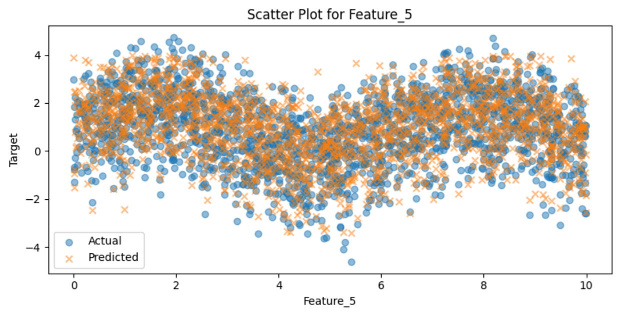
When implementing this architecture in the previous section, I only used one hidden and accompanying bias layer to generalize from the data without a complex layer implementation. After trying a wide array of hyperparameters I was able to get some decent results, but compared to the output with a multi-layer approach, the more fleshed-out system captures outliers more effectively meaning the generalization is more accurate – corroborated by a higher R-squared value as well. In terms of how quickly each converges to a good solution, the deeper network also shines in this category, boasting these outputs shown to the right in only 5000 iterations rather than the 10,000 used in the prior section. As for experimentation in this section, I employed a few different values for the number of hidden units and layers, but interestingly enough I produced the best results with my luck numbers 18 and 41 – although not a very scientifically settling result, compared to a wide array of other value-combinations, this yielded the most efficient and accurate output overall.

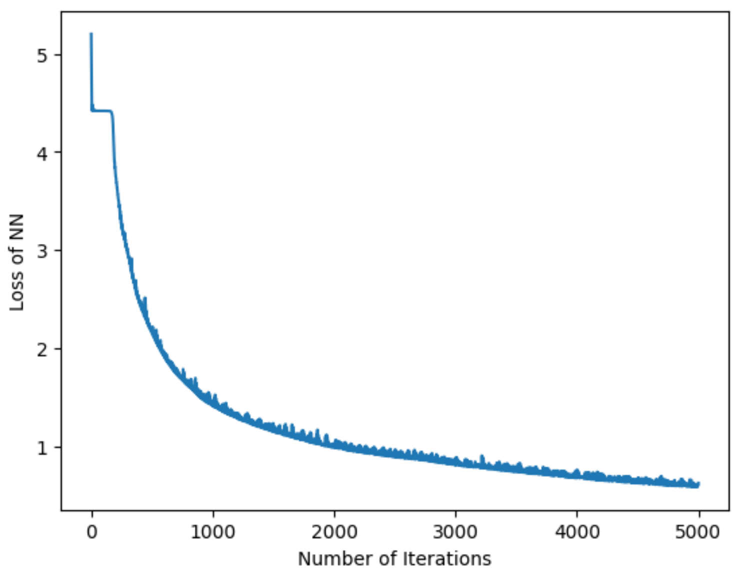

Scatter Plot for Feature_1
Target
Feature_1
Actual
Predicted


Scatter Plot for Feature_2
Target
Feature_2
Actual
Predicted


Scatter Plot for Feature_3
Target
Feature_3
Actual
Predicted


Scatter Plot for Feature_4
Target
Feature_4
Actual
Predicted


Scatter Plot for Feature_5
Target
Feature_5
Actual
Predicted


Loss of NN
Number of Iterations