Anne Trinh
CS3B

Assignment 5 Write-Up

Summary of Functions

run_iris()

1. **run_iris() was run with the parameters given in the assignment prompt.**

Parameters		Results	
Training factor	0.7	Start Training RMSE	0.56
Epoch	10001	Final Training RMSE	0.29
Hidden layers (#nodes)	1 (3)	Testing RMSE	0.27
Verbosity	2		

The testing RMSE was similar to the final training RMSE, which indicates that the parameters given did not result in any overfitting or underfitting of the data during the training. This means the training factor used was most likely appropriate for the dataset passed into the function. While the RMSE did decrease over time, the RMSE is still relatively high and the model could be further optimized.

2. **run_iris() with more hidden layers.**

Parameters		Results	
Training factor	0.7	Start Training RMSE	0.59
Epoch	10001	Final Training RMSE	0.13
Hidden layers (#nodes)	**3 (3)**	Testing RMSE	0.35
Verbosity	2		

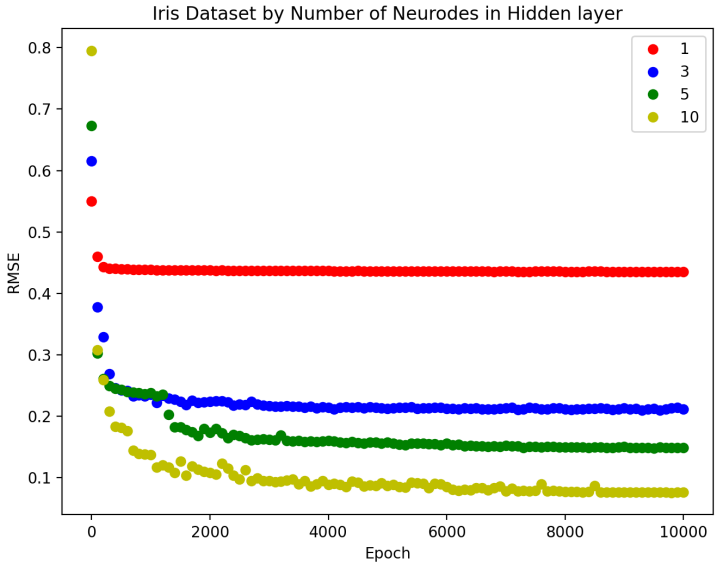
Adding more hidden layers improved our final training RMSE but not our testing RMSE. This shows that our model was likely overfitted with the introduction of 2 more hidden layers.

3. **run_iris() with more neurodes added and only one hidden layer.**

Parameters		Results	
Training factor	0.7	Start Training RMSE	0.70
Epoch	10001	Final Training RMSE	0.13
Hidden layers (#nodes)	**1 (5)**	Testing RMSE	0.12
Verbosity	2		

Anne Trinh
CS3B

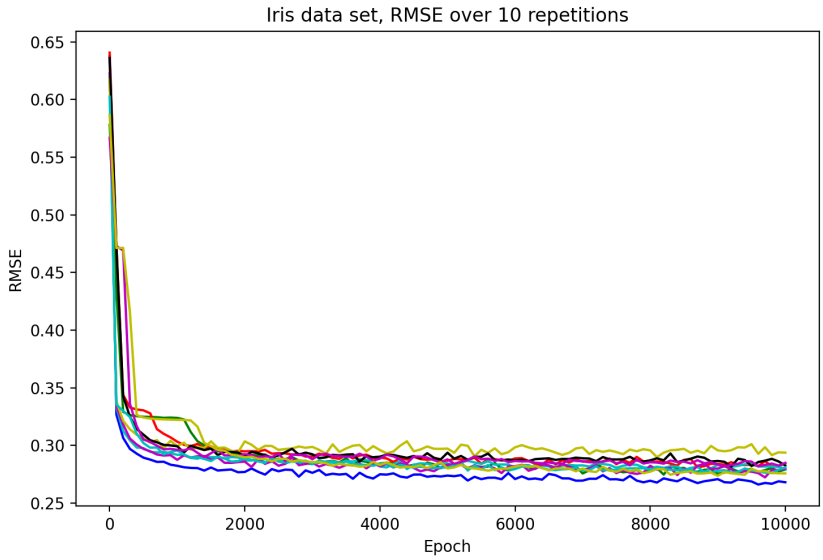
Based on the final training RMSE and the testing RMSE, it seems like our model benefits from having only one hidden layer with an increased number of neurodes. Both RMSEs are the lowest observed so far and the testing and training RMSE are similar in value, indicating no overfitting. To confirm that the number of neurodes did impact the model in such a way, the average RMSE of 3 runs of different models with various numbers of neurodes in its hidden layer was recorded and plotted onto one graph for comparison.



As shown in the graph, the final training RMSE decreases considerably the more neurodes there are in the hidden layer. This is because each hidden neurode works to adjust its weight and delta in every backpropagation, fine-tuning the predicted outputs to the expected outputs. This introduces the potential for overfitting though. Unfortunately I was only able to get the average testing RMSE for the model with 10 hidden neurodes, which was 0.12. This is a bit higher than where the training RMSE converged (<0.1).

4. **run_iris() run 10 times to see RMSE variability.**
   I also ran the model 10 times with 10,000 epochs and a training factor of 0.7 to see how much the RMSE trends would vary between each repetition. Here are the results:

Anne Trinh
CS3B



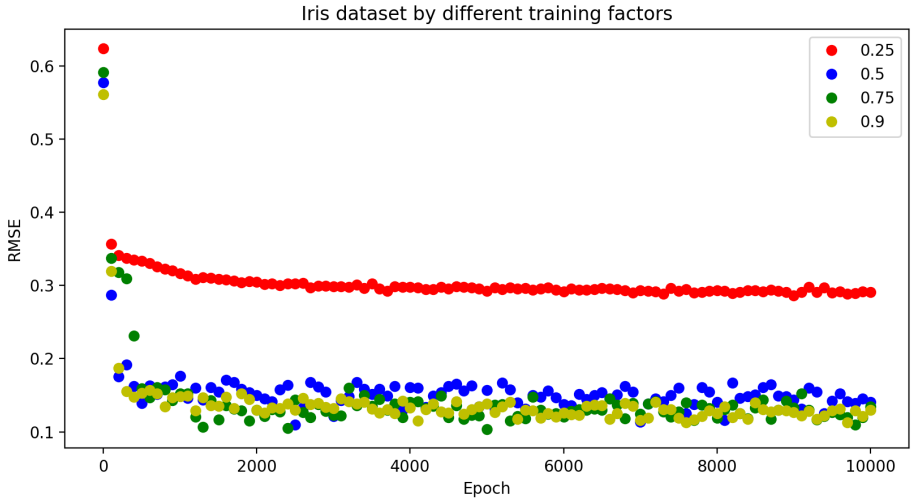
Overall, I was surprised to see that the RMSE trends didn't fluctuate too much between each replicate. This also means that for prior tests that I ran testing out other variables (i.e. training factors, number of epochs) where there was only one replicate of each variable, the results of the tests were valid enough to be compared to each other.

5. **Iris data with different training factors**

The Iris dataset was fed through the neural network with the following conditions to see the effect of different training factors on the model.

Number of Epochs	Training Factor	Learning Rate	Hidden Layers
10001	**0.25**	0.05	3
10001	**0.5**	0.05	3
10001	**0.75**	0.05	3
10001	**0.9**	0.05	3

Anne Trinh
CS3B



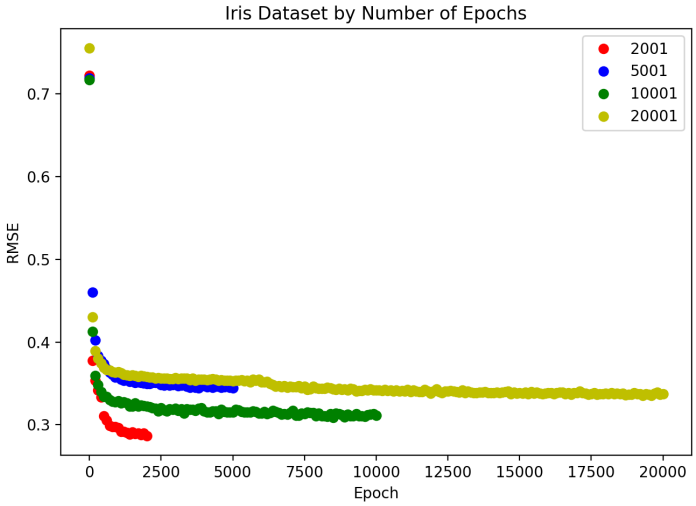
For all the different models, the RMSE starts off relatively high and converges within the first 1000 epochs. Over the next ~9,000 epochs, the RMSE stabilizes within a band. With a training factor of 0.25, the RMSE levels out around 0.3 whereas training factors of greater than 0.5 result in RMSEs between 0.1 and 0.2. In general, the higher the training factor (or ratio of training examples to testing examples), the lower the RMSE. This makes sense- as the model is presented with not only more examples but different ones too, the more it'll learn. However, at some point, we go past the point of optimization. Note that a training factor of 0.9 doesn't necessarily yield a significantly lower RMSE than a training factor of 0.5.

6. **Iris data with different # of epochs**

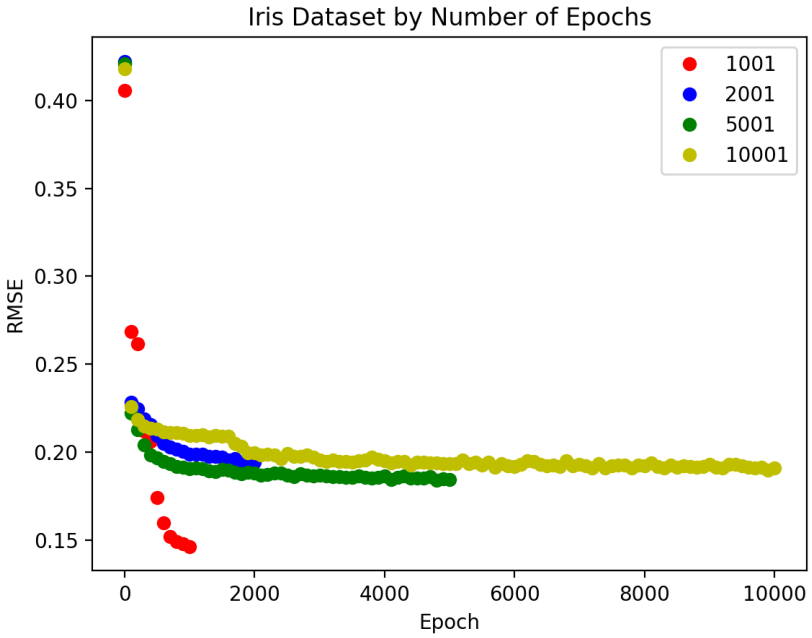
The Iris dataset was also fed through the neural network with the following conditions to see the effect of different training epoch cycles on the model.

Number of Epochs	Training Factor	Learning Rate	Hidden Layers (# of nodes)
5001	0.7	0.05	1 (3)
10001	0.7	0.05	1 (3)
20001	0.7	0.05	1 (3)
40001	0.7	0.05	1 (3)

Anne Trinh
CS3B



Each line represents an average of 5 training regiments with 2001, 5001, 10001, and 20001 epochs respectively. Interestingly, I would expect the model with the most amount of epochs to end up with the lowest RMSE over time, but in this case, we saw the lower RMSE with only 2001 models. To double check, I ran the same test with 1001, 2001, 5001, and 10001 epochs and got similar trends (depicted below).



The graphs above show that at some point, it no longer becomes effective to increase the number of epochs used in a NN model. In my testing, I also went up to 40001 epochs, whereby it took over 15 minutes to train a model with the iris dataset. The epochs seem to always converge with its RMSE relatively early only, negating the need to go even beyond 10001 cycles.

Anne Trinh
CS3B

7. **Iris dataset with different # of nodes hidden layers**
   The Iris dataset was also fed through the neural network with the following conditions to see the **effect of amounts of hidden layers** on the model.

Number of Epochs	Training Factor	Learning Rate	Hidden Layers (nodes in layer)
10001	0.7	0.05	1 (**1**)
10001	0.7	0.05	1 (**3**)
10001	0.7	0.05	1 (**5**)
10001	0.7	0.05	1 (**10**)

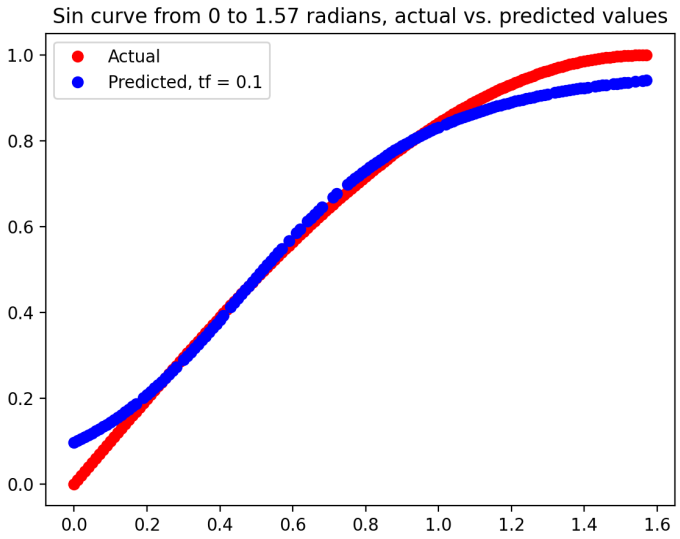
Summary:
After running the models, some supplementary research was done looking into optimization of the models. According to this source, many models can be optimized using just 1 hidden layer with a number of neurodes that is the mean of the neurodes in the input and output layers. The number of nodes plays a more significant role than the number of layers when training the model. This is because at each node is where weights and deltas are calculated.

Anne Trinh
CS3B

run_sin():

1. **run_sin() was run with the parameters given in the assignment prompt.**
   The RMSE results are listed below, as well as a graph of the predicted outputs during the testing set compared to the target outputs.

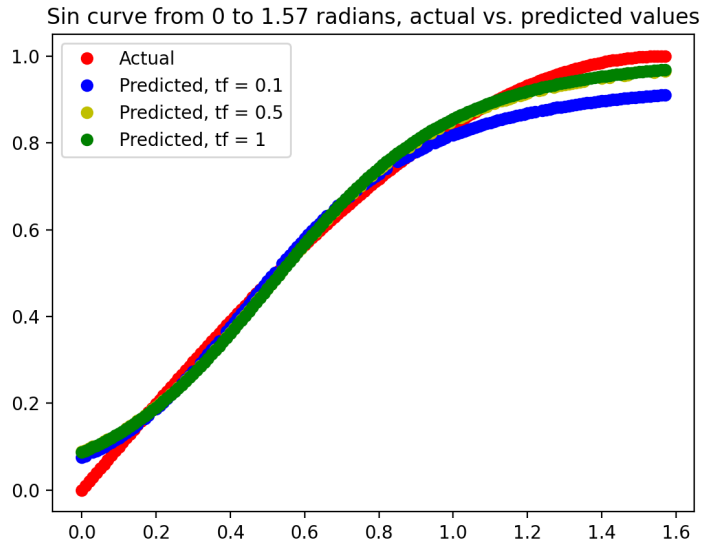
Parameters		Results	
Training factor	0.1	Start Training RMSE	0.25
Epoch	10001	Final Training RMSE	0.036
Hidden layers (#nodes)	1 (3)	Testing RMSE	0.036
Verbosity	2		



While the final training RMSE and the testing RMSE were both closer to each other and relatively low (0.036), the graph above shows that the model's predictions could be more accurate.

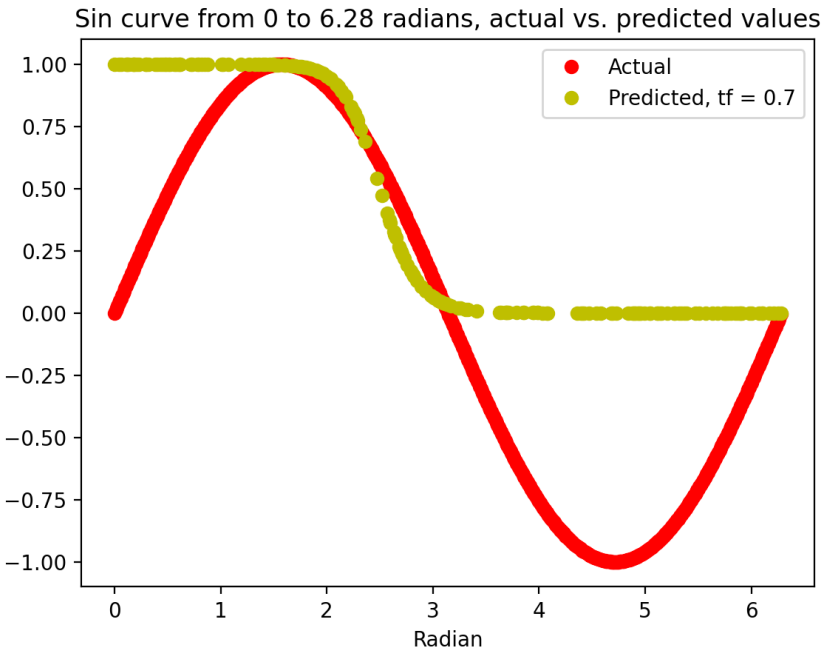
2. **run_sin() with different training factors**
   I tested the effect of different training factors on the model as depicted in the graph below.

Anne Trinh
CS3B



When the model was run with training factors of 0.5 and 1, the predicted outputs were closer to the actual outputs. For the training factor of 1, the model was tested with data already seen during the training period (that is, split_set() was called and the data reprimed). Interestingly, even when trained and tested with 100% of the data from the same dataset, the predicted output did not match the actual output.

3. **run_sin() extended to 6.28 radians instead of 1.57.**
When run_sin() was extended to 6.28 radians instead of 1.57, the following results were achieved:



Anne Trinh
CS3B

The predictions were even more inaccurate. I initially hypothesized that this is because in the dataset, some y-values are associated with multiple x-values. The model probably had trouble establishing relationships between the features and the labels due to these multiple occurrences.

run_xor():

1. **run_xor() was run with the parameters given in the assignment prompt.**

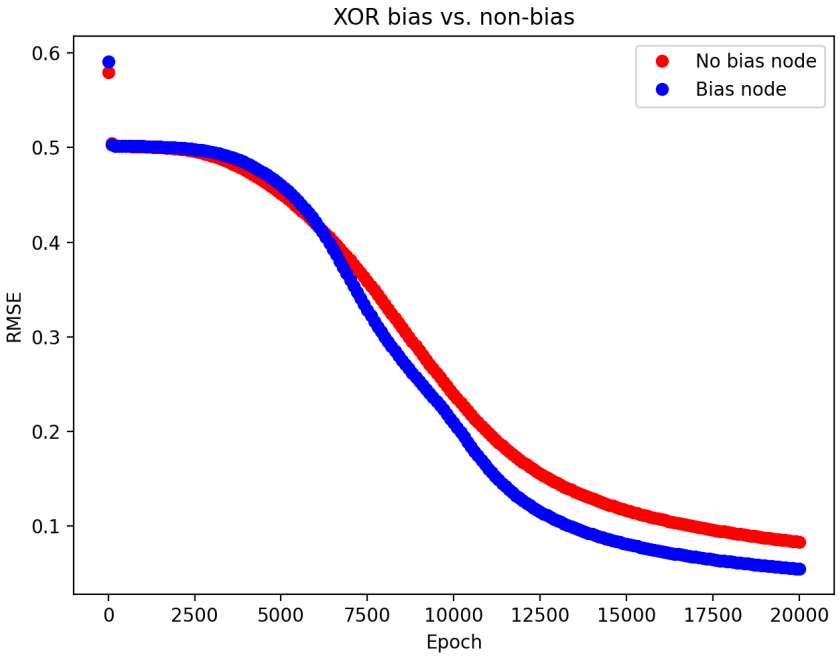
Parameters		Results	
Training factor	1	Start Training RMSE	0.59
Epoch	20001	Final Training RMSE	0.084
Hidden layers (#nodes)	1 (4)	Testing RMSE	0.084
Verbosity	2		

The testing RMSE is in line with the final training RMSE. The testing predicted values are also fairly similar to the target values here as well, listed below.

```
Input Values        Expected Values     Predicted Values
0.00                0.00                0.09
0.00
Input Values        Expected Values     Predicted Values
1.00                1.00                0.92
0.00
Input Values        Expected Values     Predicted Values
0.00                1.00                0.92
1.00
Input Values        Expected Values     Predicted Values
1.00                0.00                0.08
1.00
```

Anne Trinh
CS3B

Extra credit #1: XOR with and without a bias node



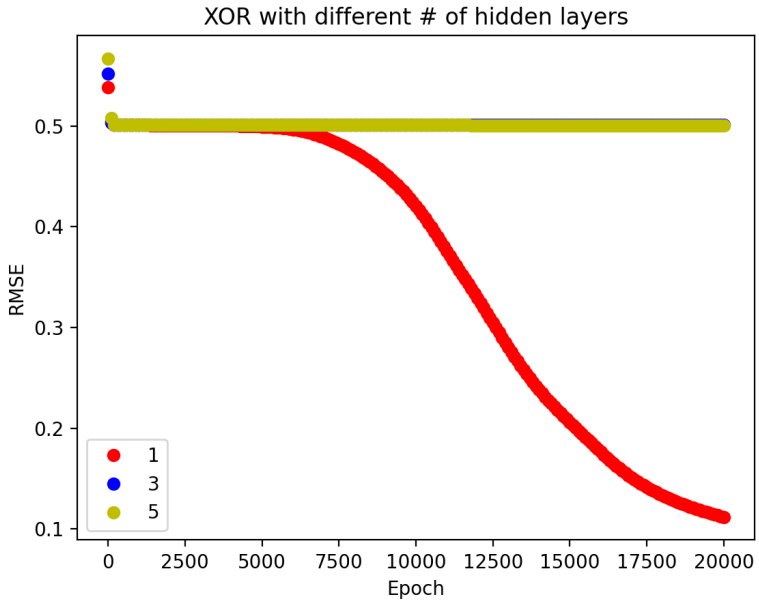
The XOR dataset was put through the neural network model with and without a bias node. In both instances the model cycled through 20000 epochs for 5 training periods. The graph above depicts the average RMSE of the 5 training periods for every 100th epoch for both the dataset with bias node and the dataset without. Both the bias node and the non-bias node follow similar trends, but over time, running XOR with a bias node yields a lower RMSE value, indicating that the model was able to predict the target output more accurately by the 20000th epoch.

The bias node acts as a constant to the data set to help the neural network with cases where the feature values are (0, 0), resulting in a weighted sum that is always 0 instead of being updated to reflect the changing weights over the course of an epoch. The sigmoid function we're using is sensitive to these cases. Adding the constant allows us to get around those cases where the weighted sum wouldn't update due to the presence of 0-value features by ensuring that there are features that have only 0-values.

Anne Trinh
CS3B

Extra credit #2: Vanishing gradient

The XOR dataset (no bias node) was put through 3 neural network models with 1, 3, and 5 hidden layers respectively. Each line depicted on the graph below is an average of 5 training periods with the model, each undergoing 20000 epochs. With 1 hidden layer in the network, the RMSEgradually decreased over the entirety of the training regiment. With 3 and 5 hidden layers, the network stayed at an RMSE of 0.5 throughout the 20000 epochs, exhibiting the "getting stuck behavior" that was described in the assignment prompt.

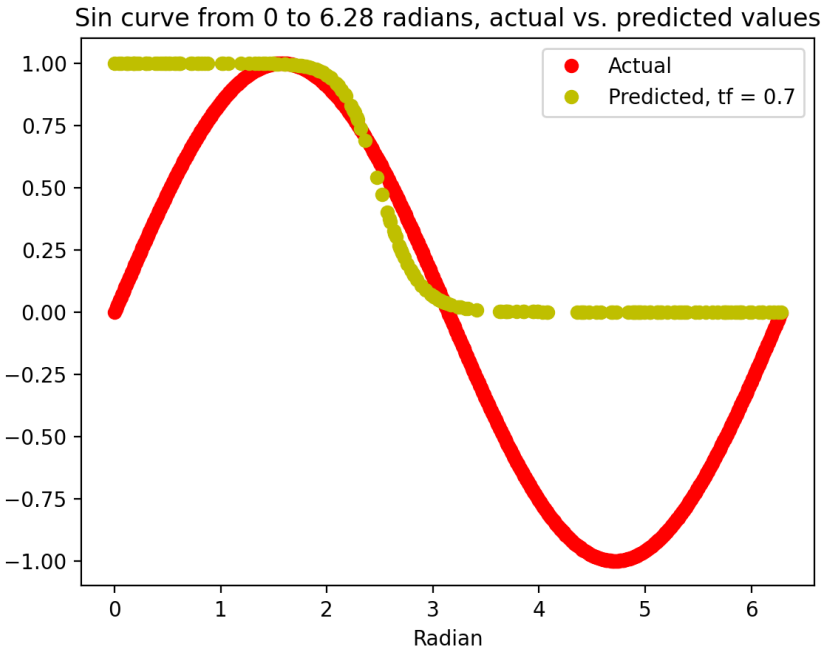


The XOR dataset with the bias node was also put through the 3 NN models with 1, 3, and 5 hidden layers as well for comparison. I got similar results to the graph depicted above, even with the bias node.

Anne Trinh
CS3B

Extra credit #3: Preprocessing data

1. **run_sin() extended to 6.28 radians (2pi) instead of 1.57 (pi/2).**
   When run_sin() was extended to 6.28 radians instead of 1.57, the following results were achieved:



The predictions were even more inaccurate. I initially hypothesized that this is because in the dataset, some y-values are associated with multiple x-values. The model probably had trouble establishing relationships between the features and the labels due to these multiple occurrences.

The model also failed where y-values fell in the negative range.