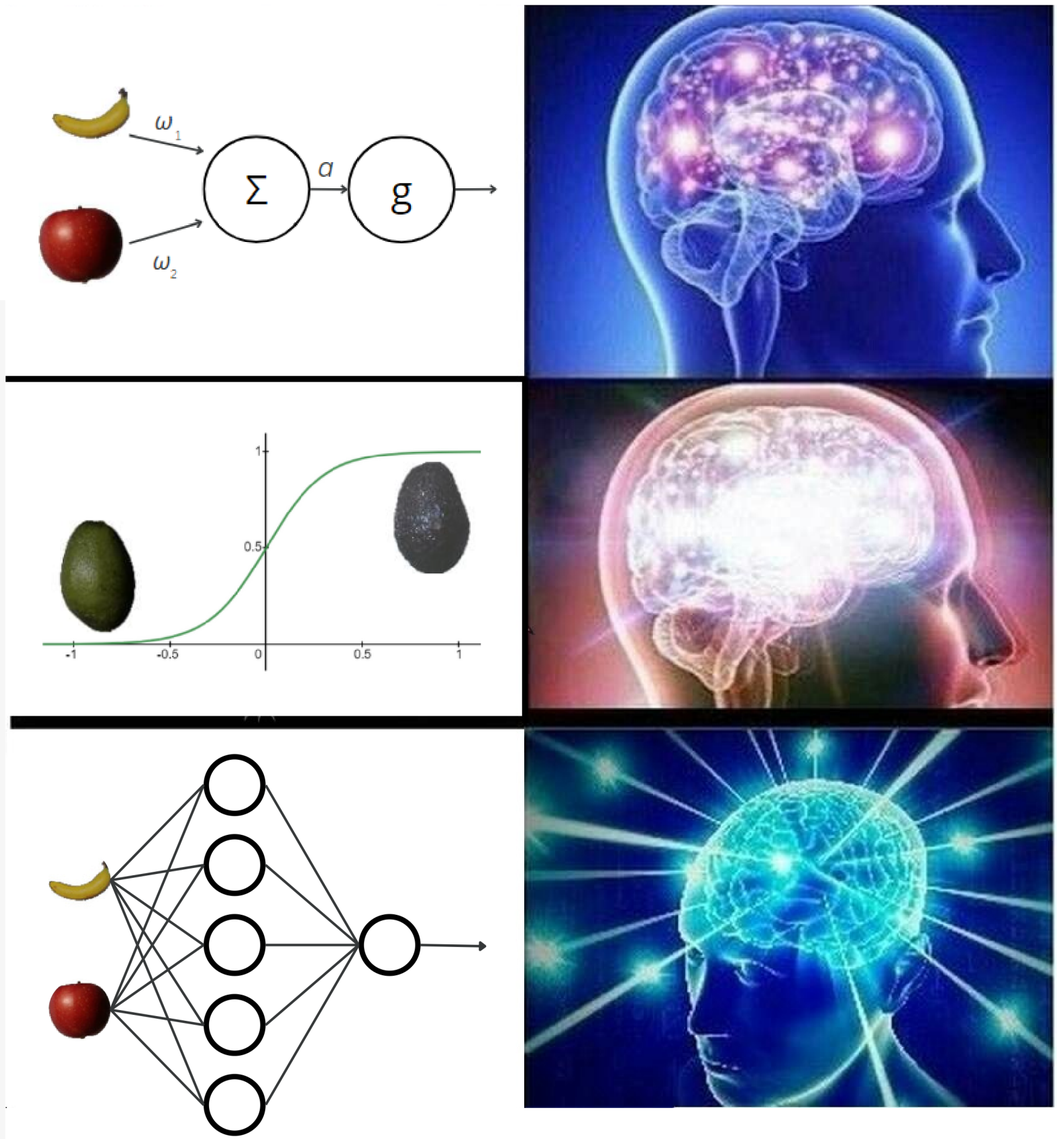
ACTIVITY 9. NEURAL NETWORKS

Julian Christopher L. Maypa
2020-07587
App Physics 157 WFY-FX-1

Objectives

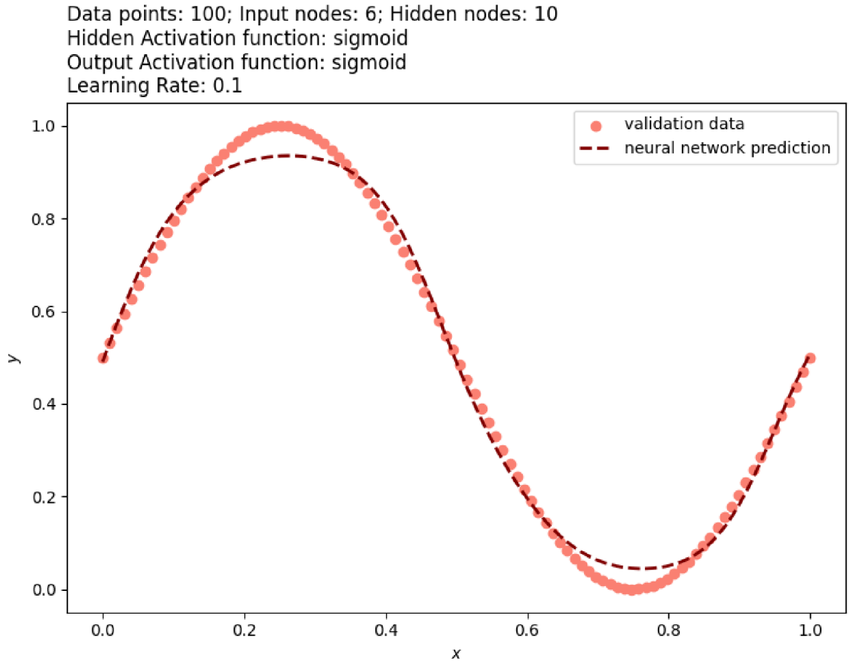
- Create a neural network to learn a sine function.
- Create a neural network to classify the fruits that I used in activity 8.

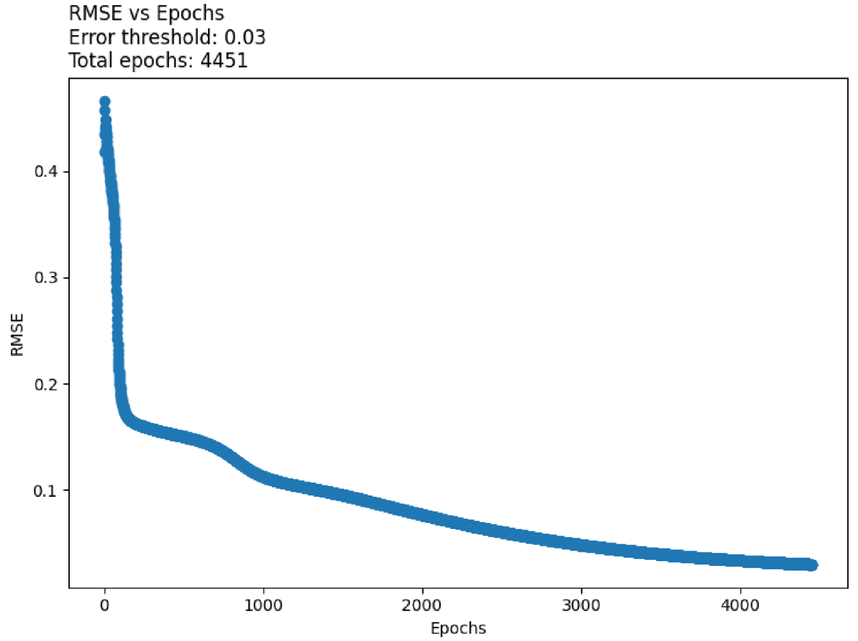


Learning a Sine Function

For this part of the activity, I created a neural network that has 3 layers (input, hidden, and output). To learn a sine function, I created a sine function with an angular frequency of 2π and the *x* values that are used are 100 equally spaced points in the range [0,1). I also normalized the values of the sinusoid from 0 to 1. I used 6 input nodes, 10 hidden nodes, 1 output node, a learning rate of 0.1, an error (RMSE) threshold of 0.03, and a sigmoid activation function for both the hidden and output layers. The 6 input nodes correspond to *x^0, x^1, x^2, x^3, x^4*, and *x^5*. With this, my results are as follows:

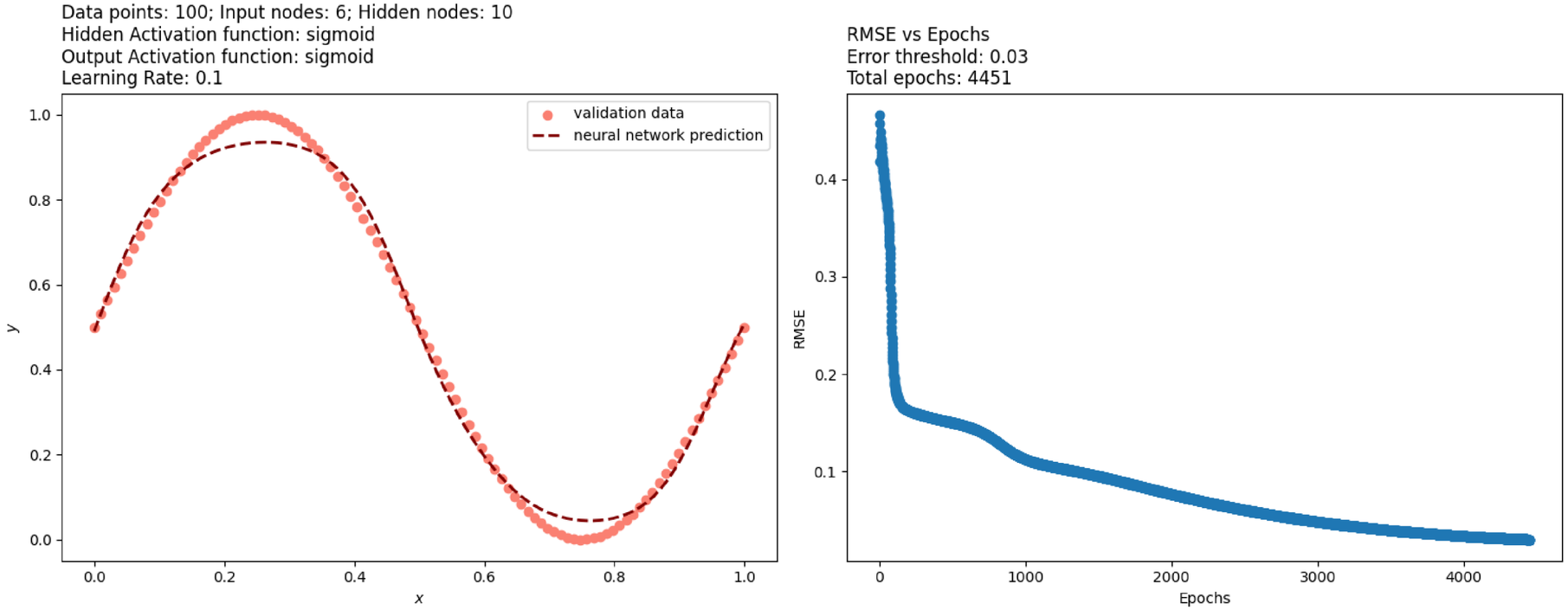
Looking at my result, it can be seen that my neural network has successfully predicted the values of the sine wave. Although the prediction values did not fully align with the validation data (sine wave data points), I can still say that my program has successfully learned the values of the sine wave. If I let my code run longer and lower the error threshold, then the predicted values will be mroe accurate. But in the interest of time. I believe that the error threshold of 0.03 is enough.



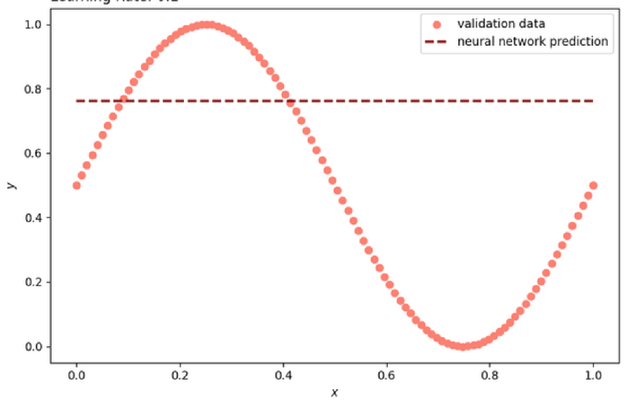


Varying Activation Functions

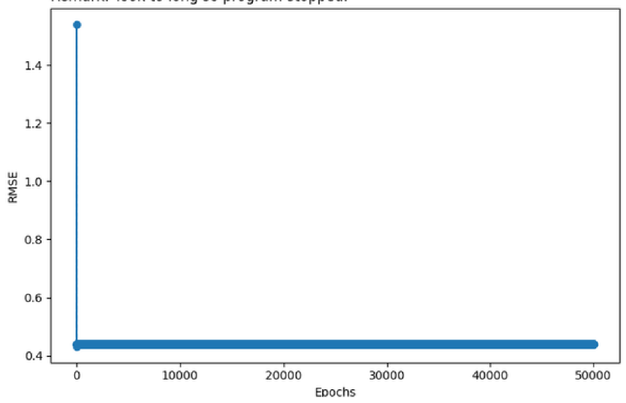
Now that I know my neural network works, I then tried different combinations of activation functions for both hidden and output layers. My results are as follows:



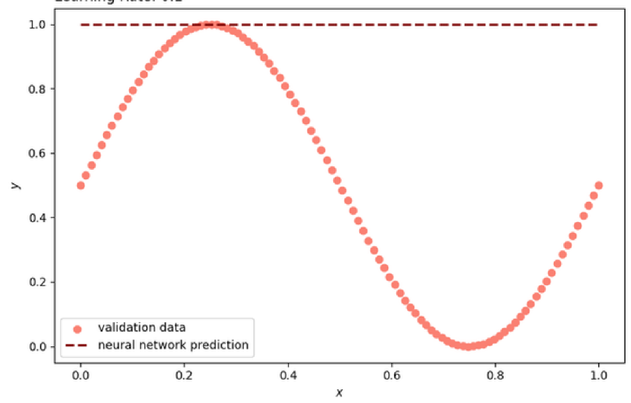
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: tanh
Learning Rate: 0.1



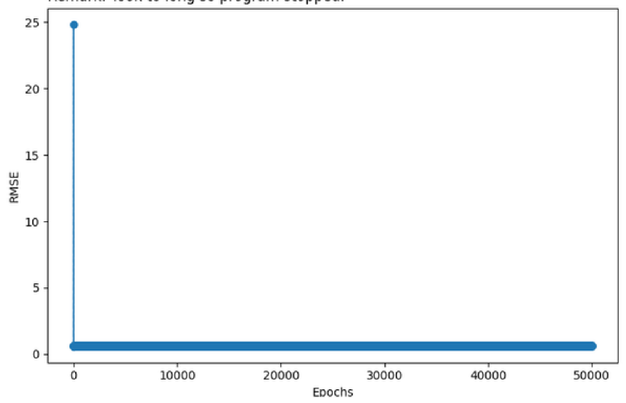
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.4384207328258201
Remark: Took to long so program stopped.



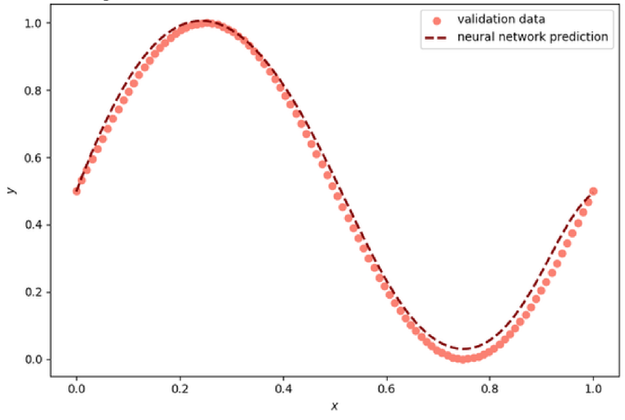
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: linear
Learning Rate: 0.1



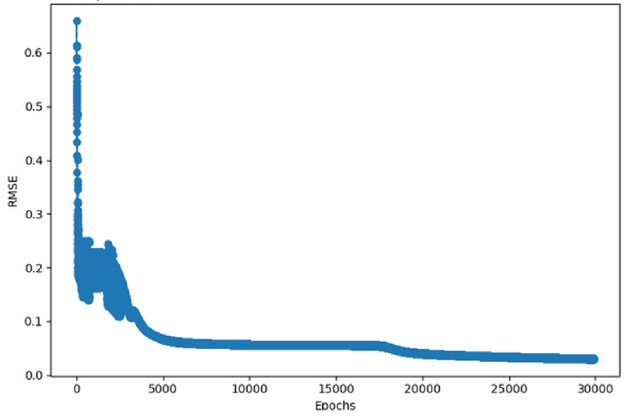
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.6113764464580964
Remark: Took to long so program stopped.



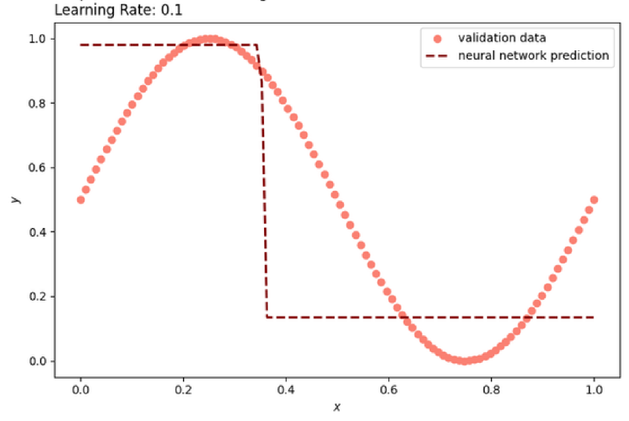
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: rectified_linear
Learning Rate: 0.1



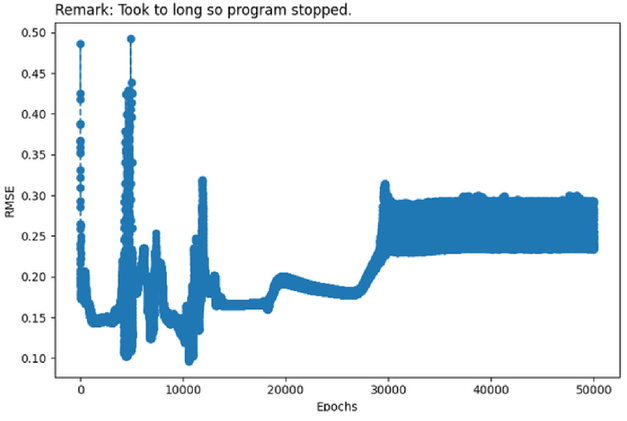
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 29896



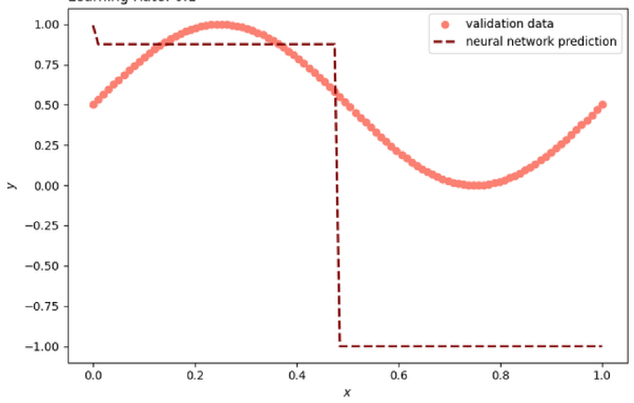
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: sigmoid
Learning Rate: 0.1



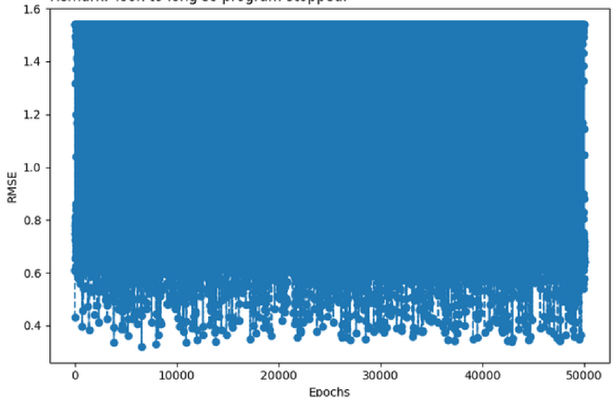
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.27370634097186725
Remark: Took to long so program stopped.



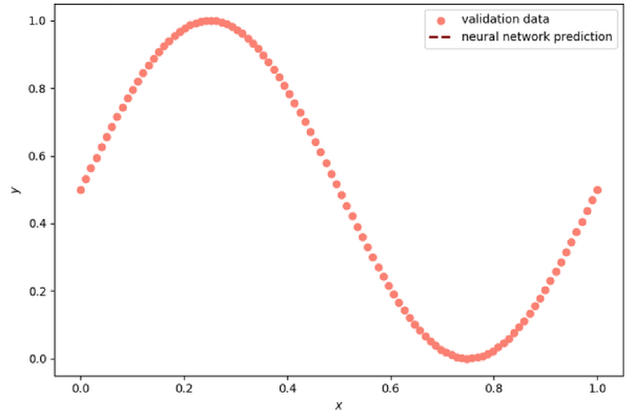
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: tanh
Learning Rate: 0.1



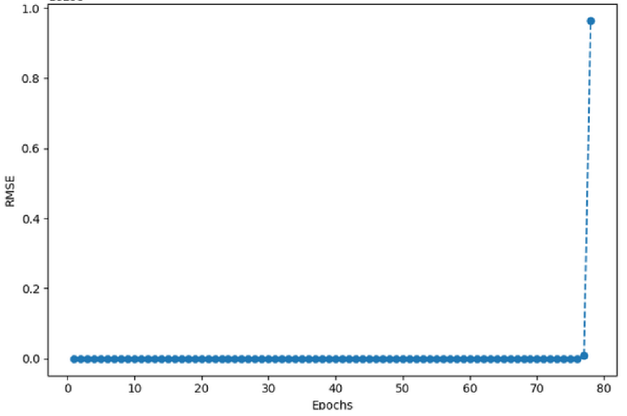
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.8796777199044635
Remark: Took to long so program stopped.



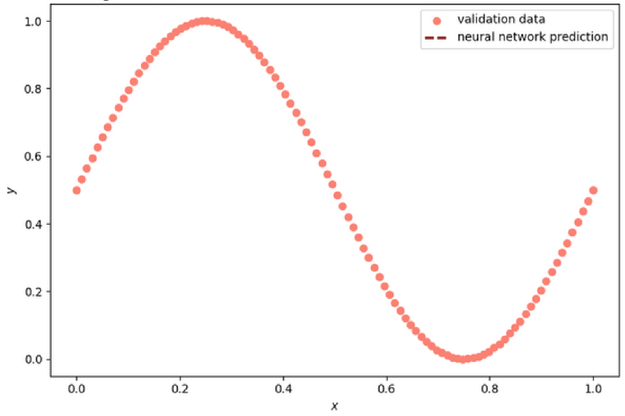
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: linear
Learning Rate: 0.1



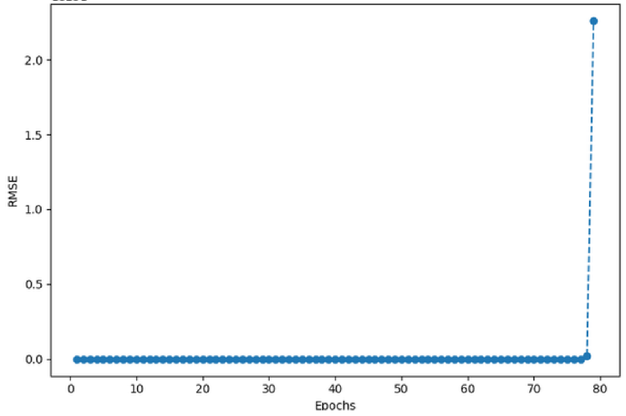
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000



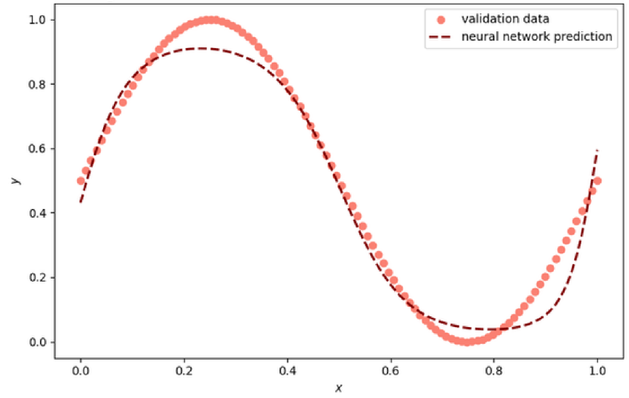
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: rectified_linear
Learning Rate: 0.1



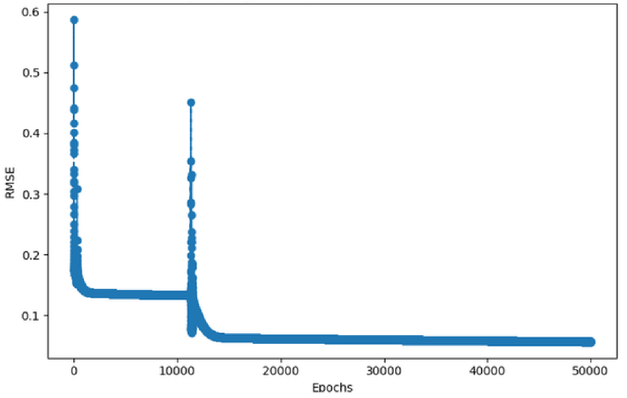
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000

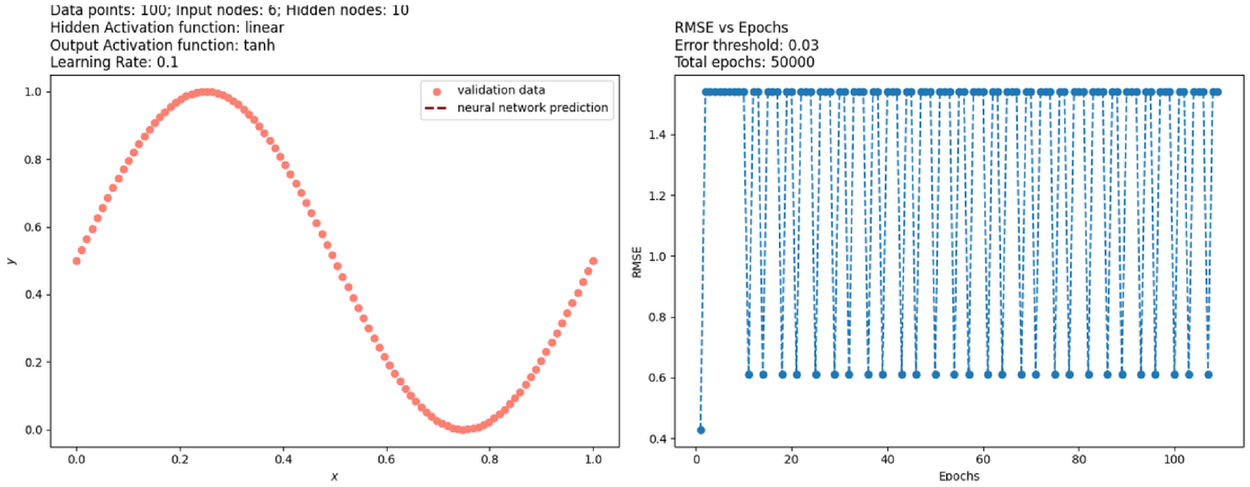


Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: sigmoid
Learning Rate: 0.1

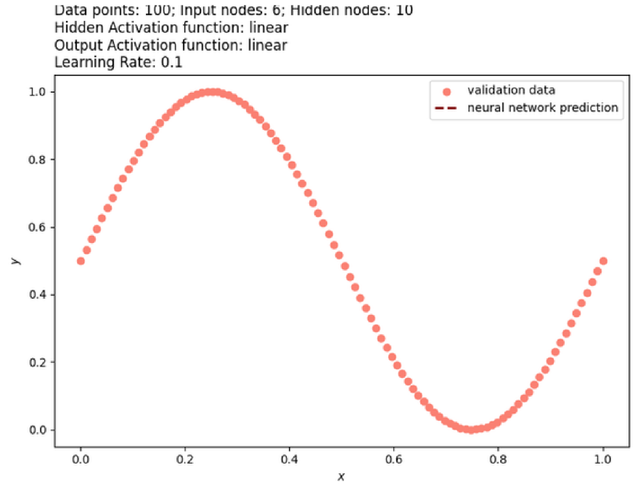


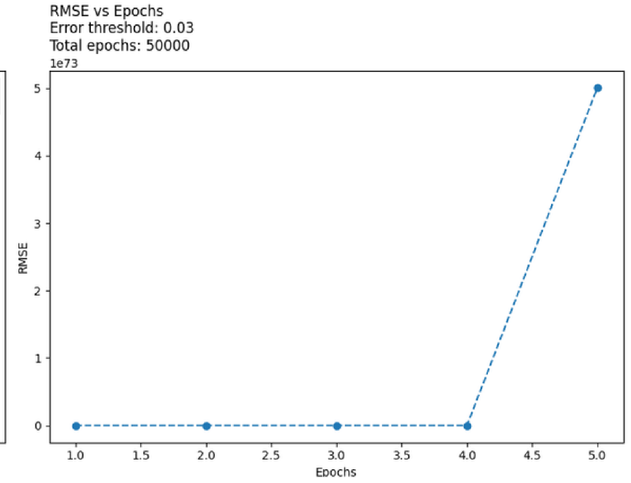
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.057390244183844884
Remark: Took to long so program stopped.

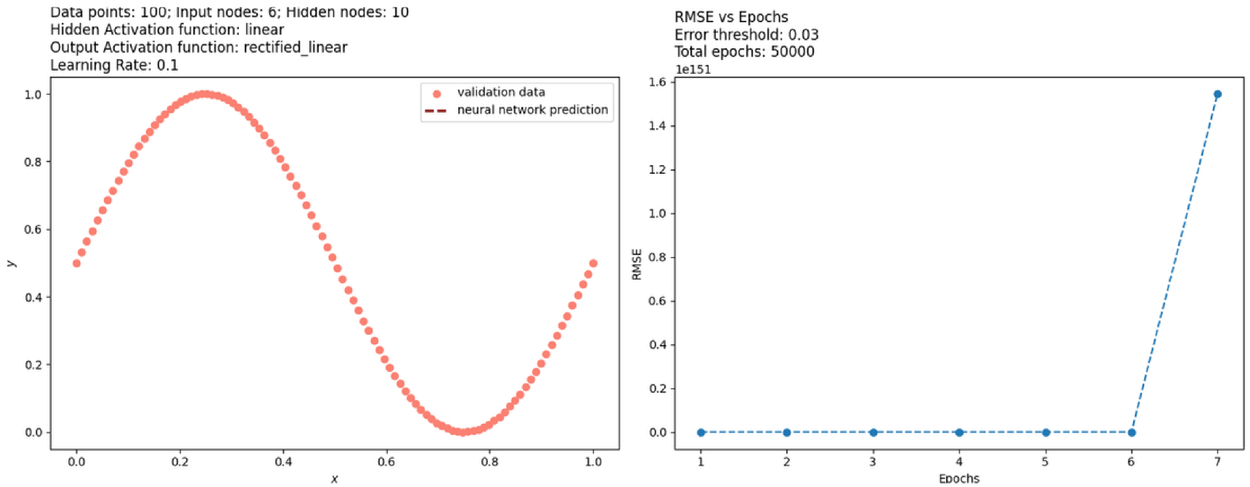



Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: tanh
Learning Rate: 0.1
validation data
neural network prediction
x
y

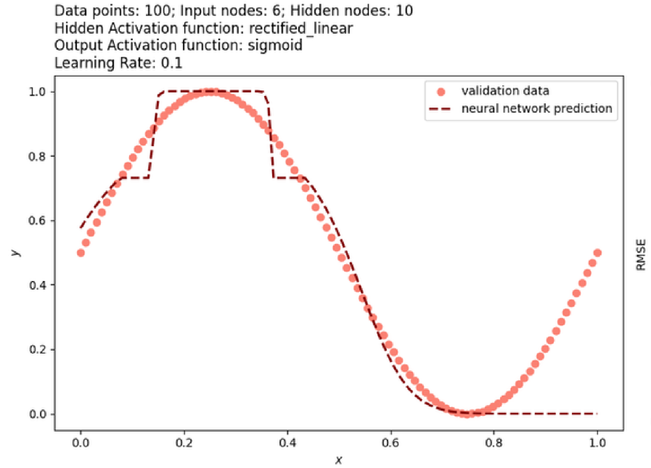
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
RMSE
Epochs

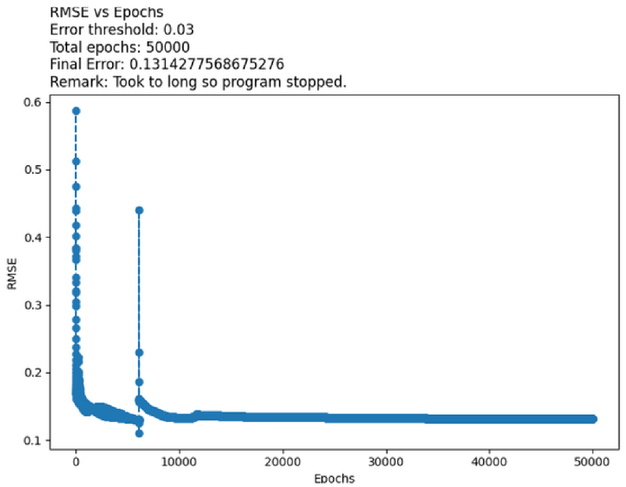

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: linear
Learning Rate: 0.1
validation data
neural network prediction
x
y

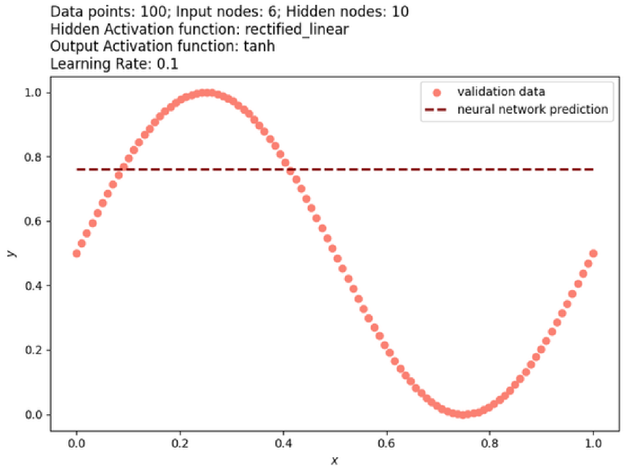

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e73
RMSE
Epochs

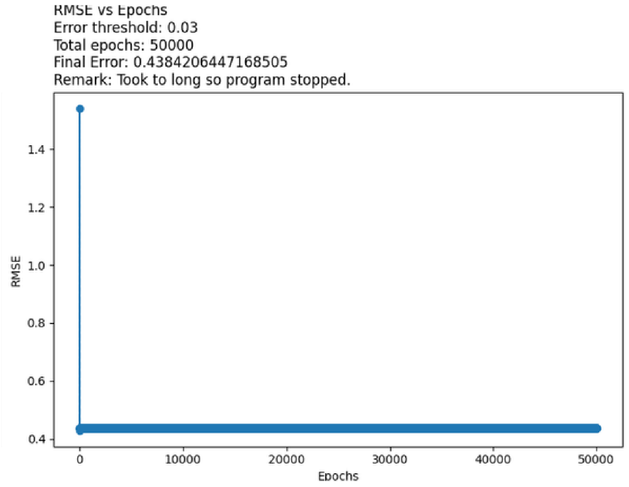

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: rectified_linear
Learning Rate: 0.1
validation data
neural network prediction
x
y

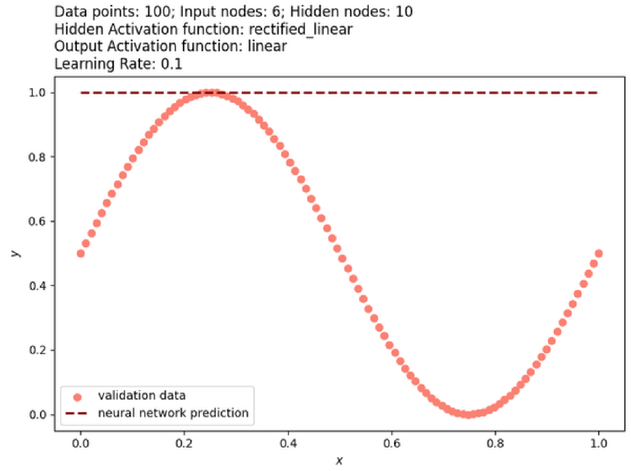
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e151
RMSE
Epochs

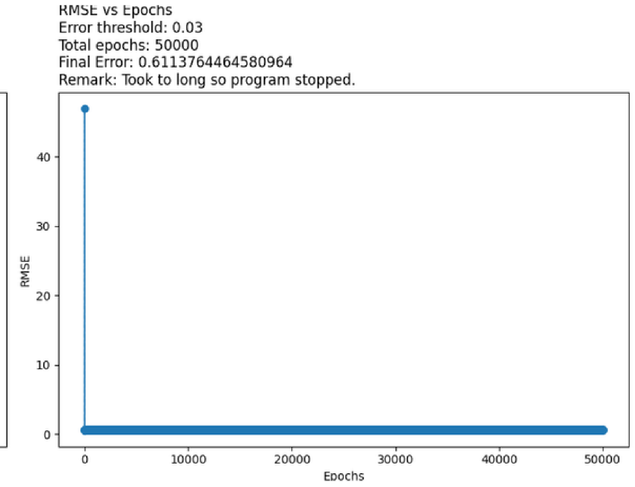

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: rectified_linear
Output Activation function: sigmoid
Learning Rate: 0.1
validation data
neural network prediction
x
y

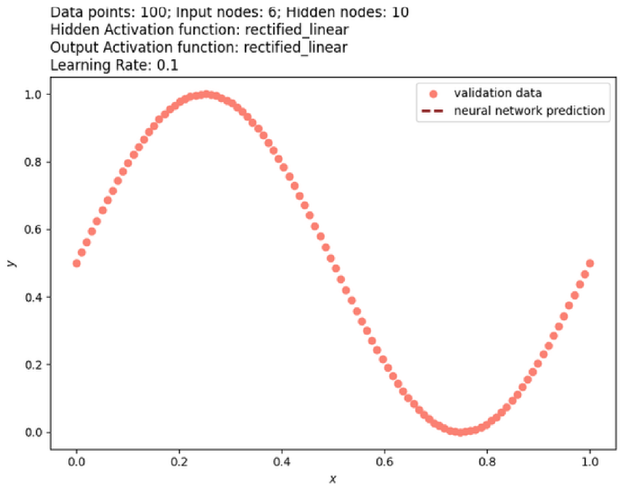

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.1314277568675276
Remark: Took to long so program stopped.
RMSE
Epochs

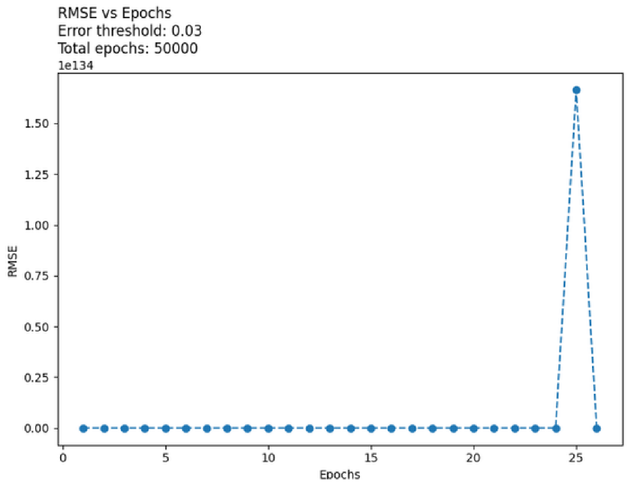








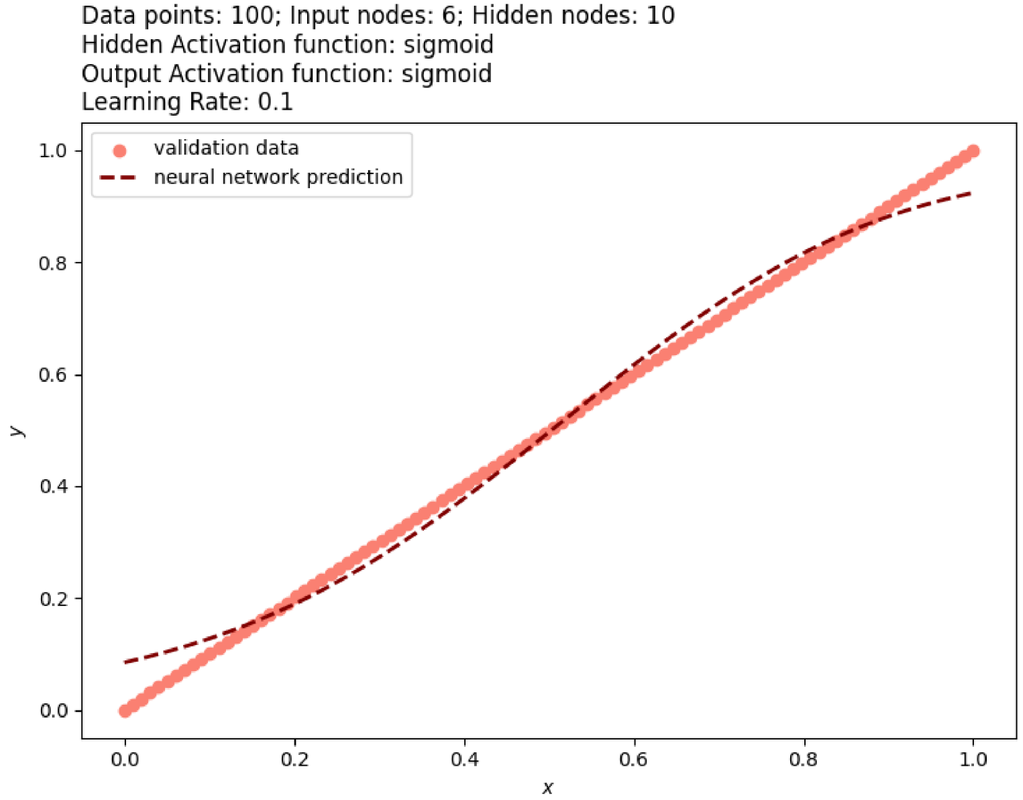


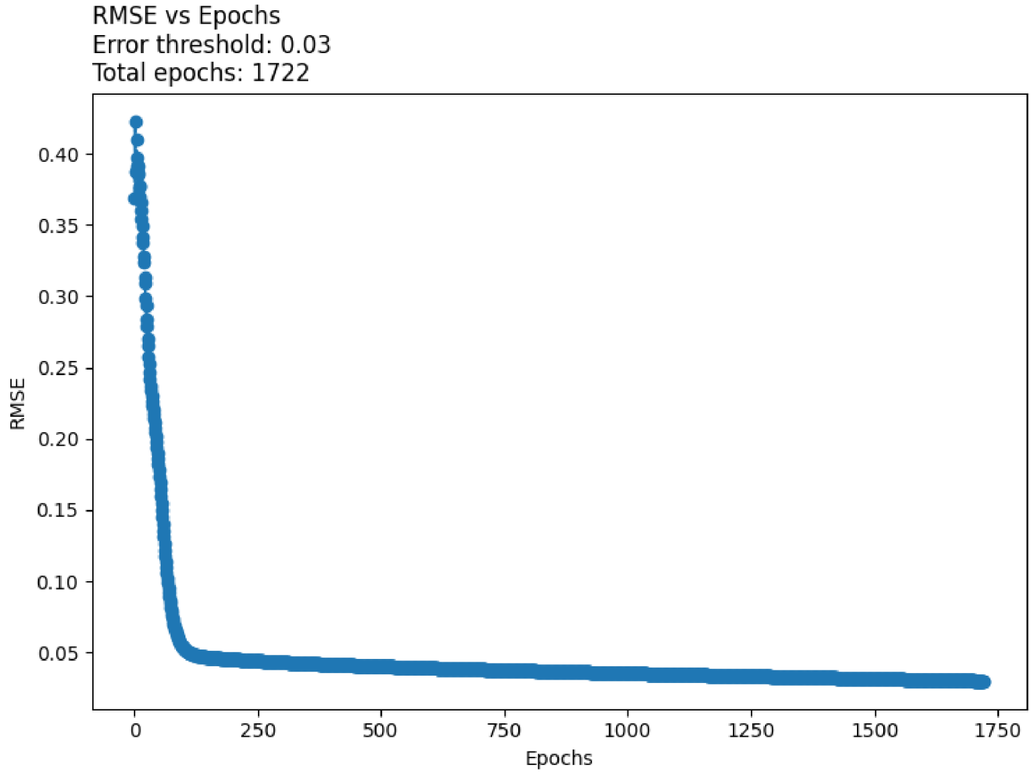


Looking at the results, it can be seen that there are some combinations of activation functions that are successful in learning the sine wave. However, some combinations are unsuccessful because their errors exploded, they were not able to reduce the error below the threshold, or encountered an error in the activation functions such as an overflow error (this happened to combinations that did not plot the prediction line). This is to be expected because the accuracy of the neural network depends on the linearity of the activation functions. So the unsuccessful combinations means that it is a combination that does not work for learning sine functions [2]. However, they might still work for other functions.

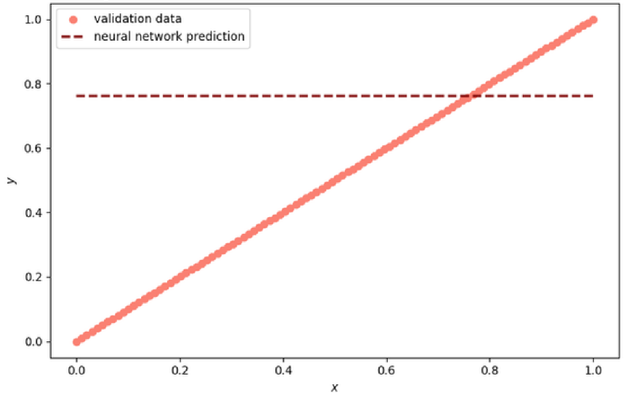
Learning a different function

Using the same parameters as before, I used a different function to learn using my neural network. I used a linear function with a slope of 1. My results are as follows:

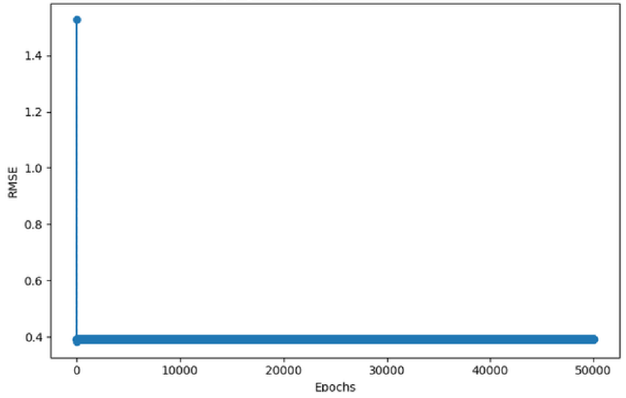




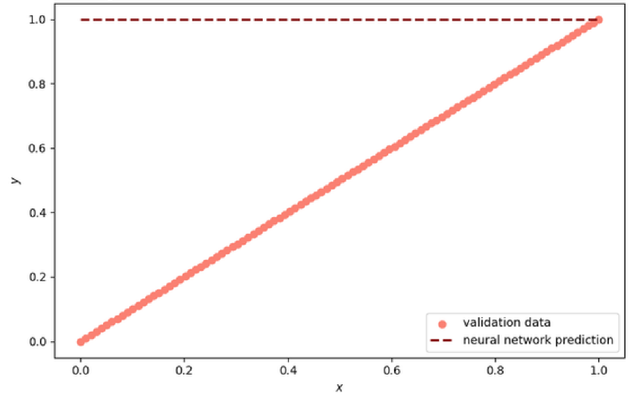
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: tanh
Learning Rate: 0.1



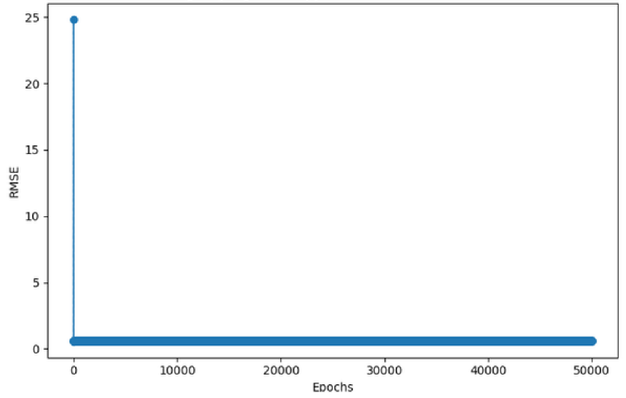
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.39172631585738205
Remark: Took to long so program stopped.



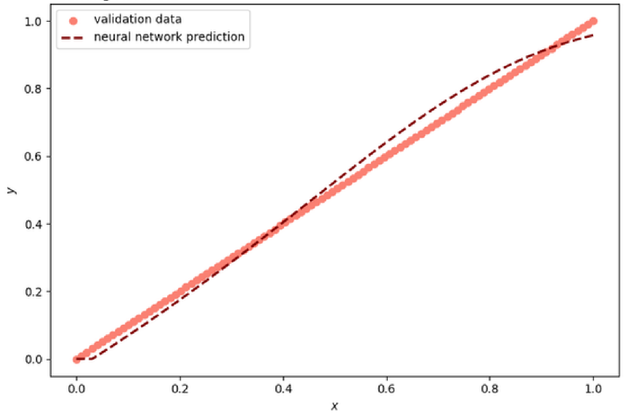
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: linear
Learning Rate: 0.1



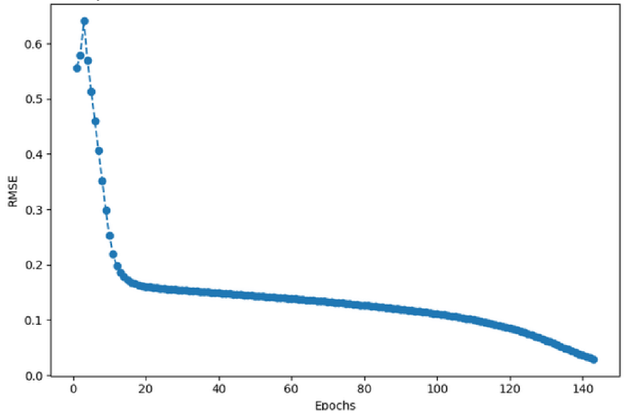
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.5788063881962905
Remark: Took to long so program stopped.



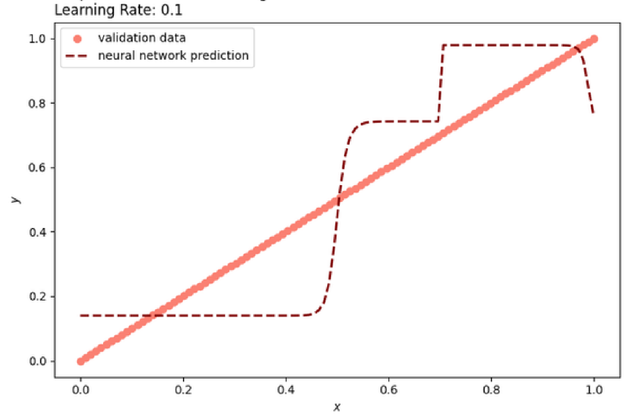
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: sigmoid
Output Activation function: rectified_linear
Learning Rate: 0.1



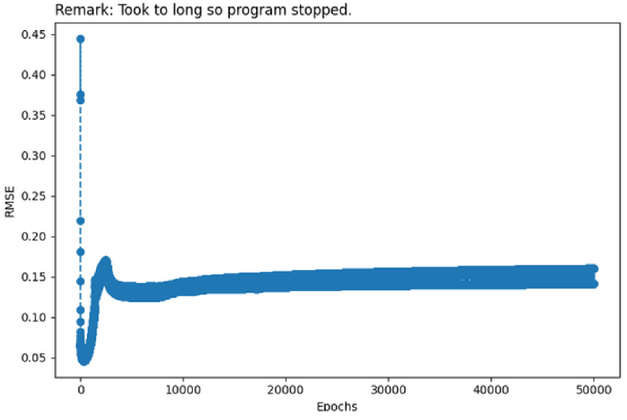
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 143



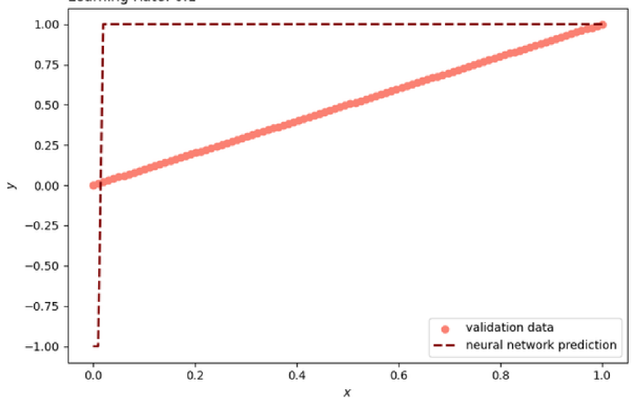
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: sigmoid
Learning Rate: 0.1



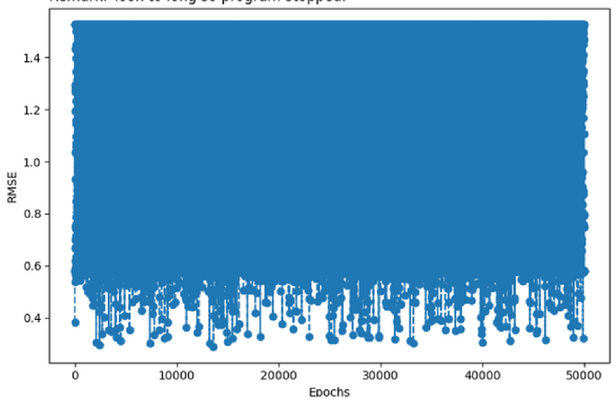
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.15957278994869367
Remark: Took to long so program stopped.



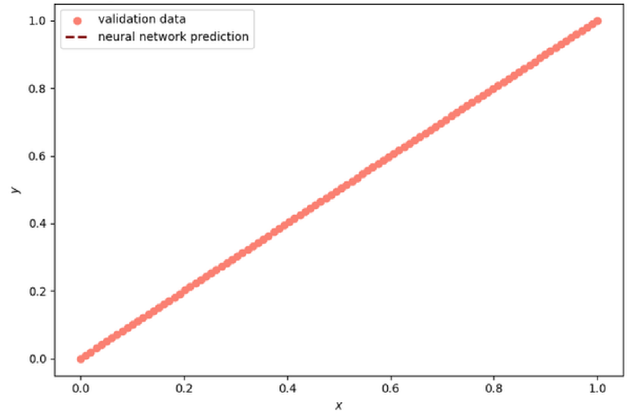
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: tanh
Learning Rate: 0.1



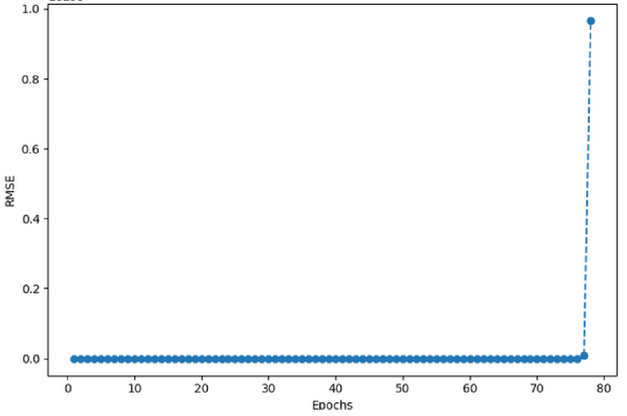
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.57915531200264
Remark: Took to long so program stopped.



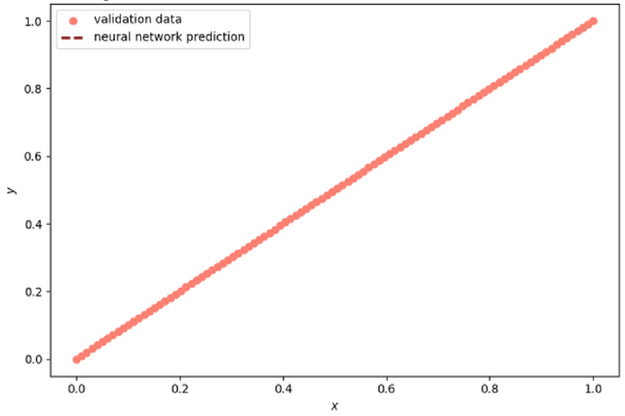
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: linear
Learning Rate: 0.1



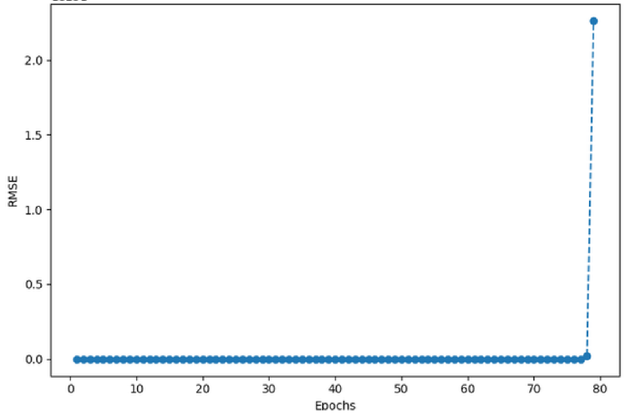
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e153



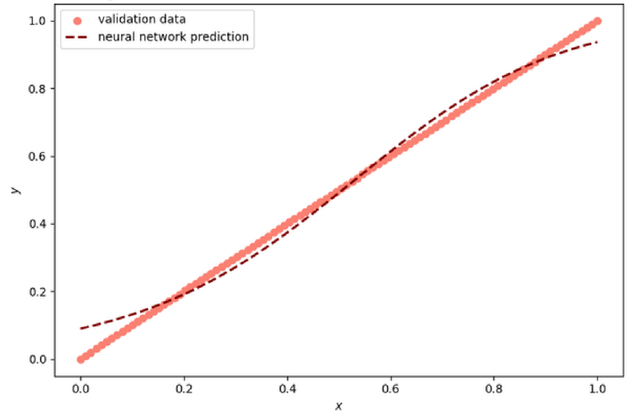
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: rectified_linear
Learning Rate: 0.1



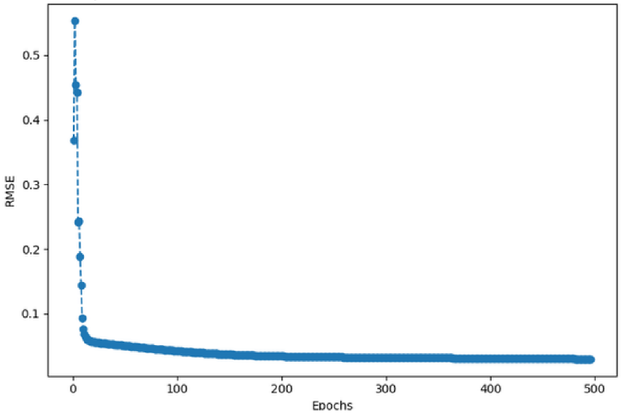
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e151

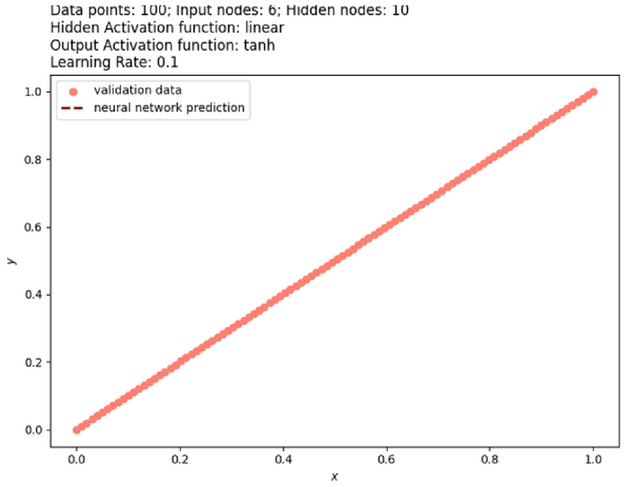


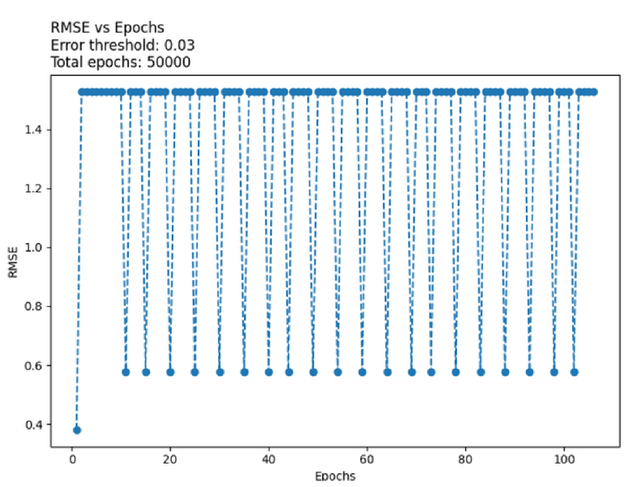
Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: sigmoid
Learning Rate: 0.1

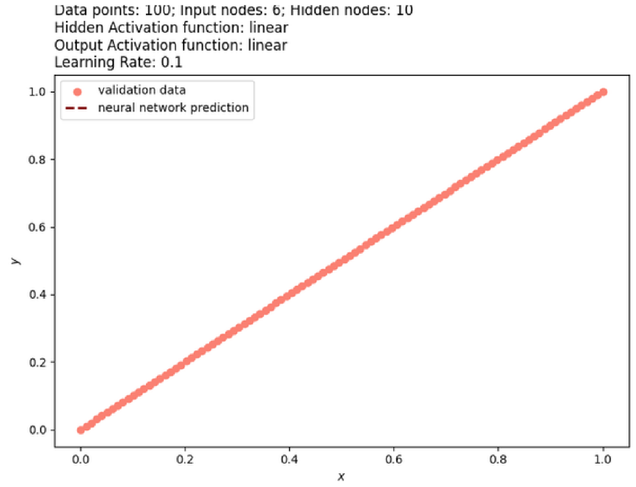


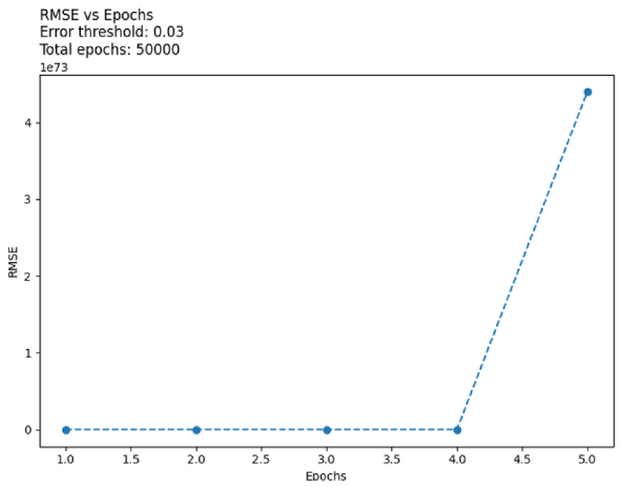
RMSE vs Epochs
Error threshold: 0.03
Total epochs: 496

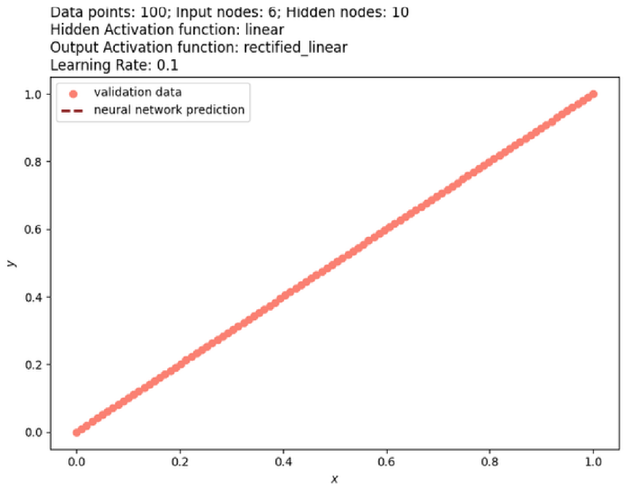


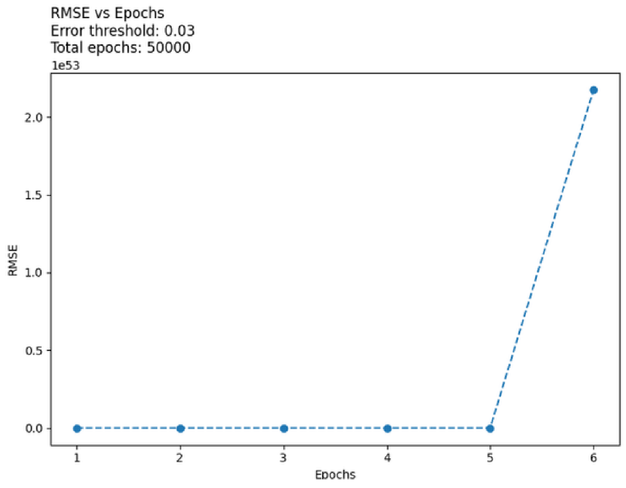

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: tanh
Learning Rate: 0.1
validation data
neural network prediction
x
y

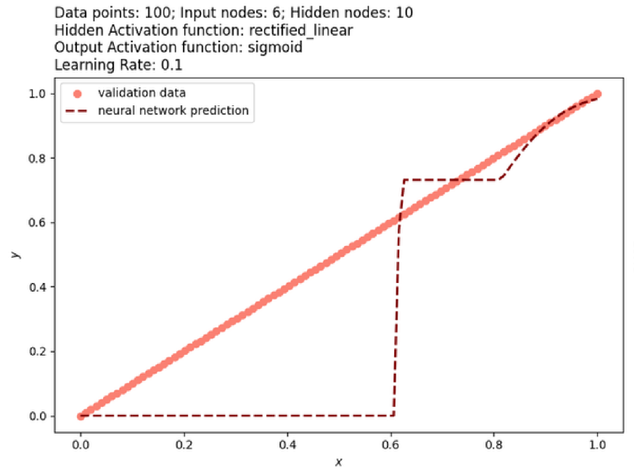

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
RMSE
Epochs

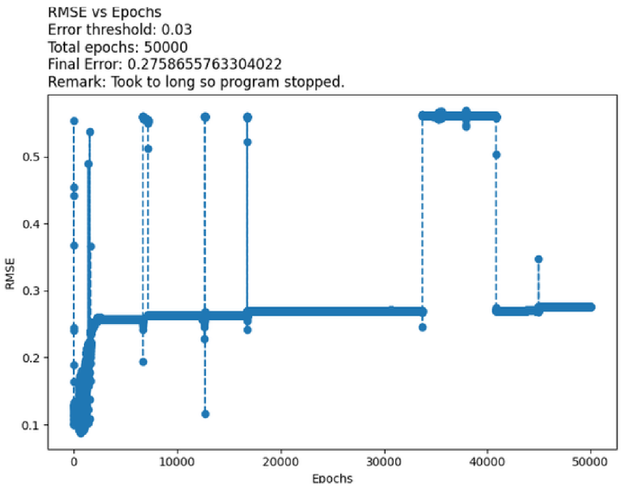

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: linear
Learning Rate: 0.1
validation data
neural network prediction
x
y

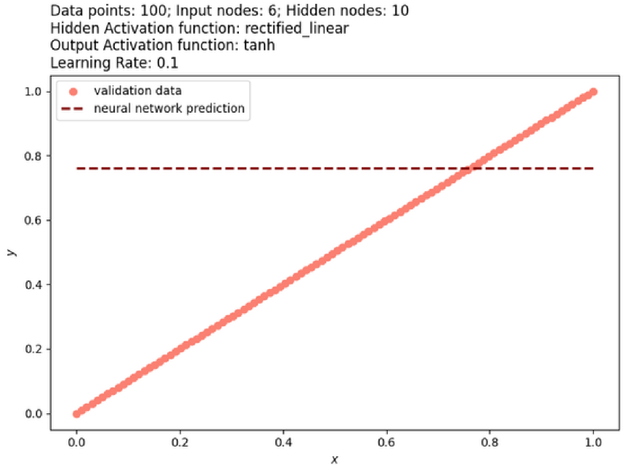

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e73
RMSE
Epochs

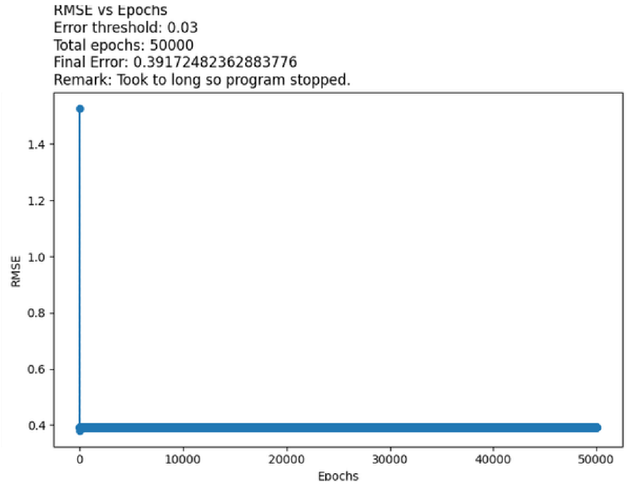

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: rectified_linear
Learning Rate: 0.1
validation data
neural network prediction
x
y

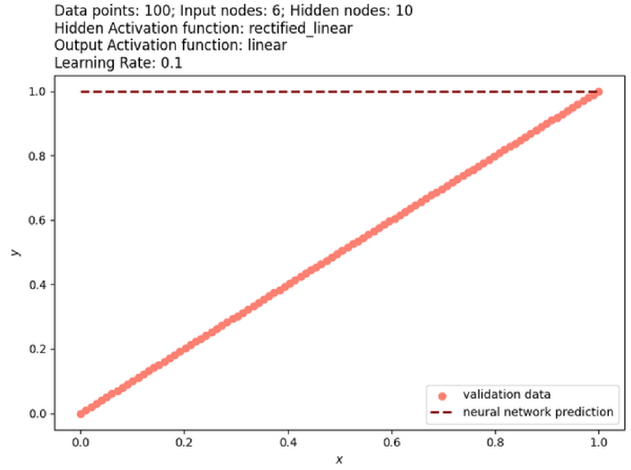

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
1e53
RMSE
Epochs

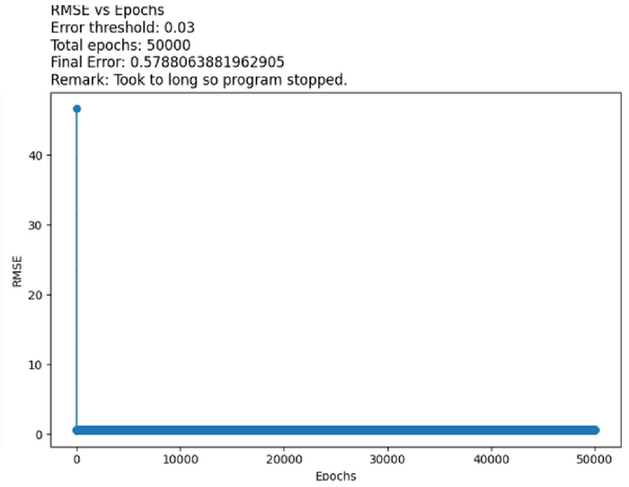

Data points: 100; Input nodes: 6; Hidden nodes: 10
Hidden Activation function: rectified_linear
Output Activation function: sigmoid
Learning Rate: 0.1
validation data
neural network prediction
x
y

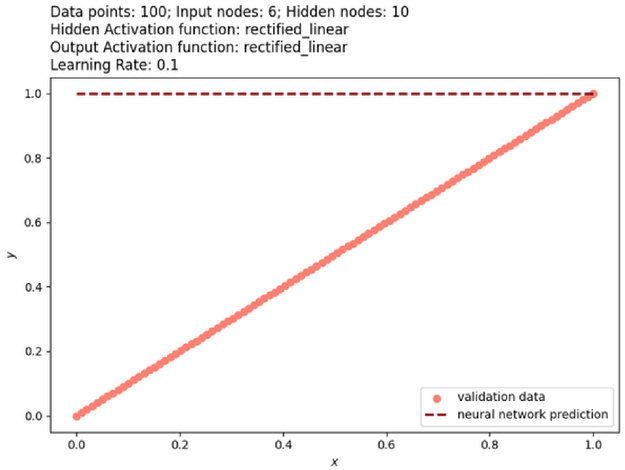

RMSE vs Epochs
Error threshold: 0.03
Total epochs: 50000
Final Error: 0.2758655763304022
Remark: Took to long so program stopped.
RMSE
Epochs

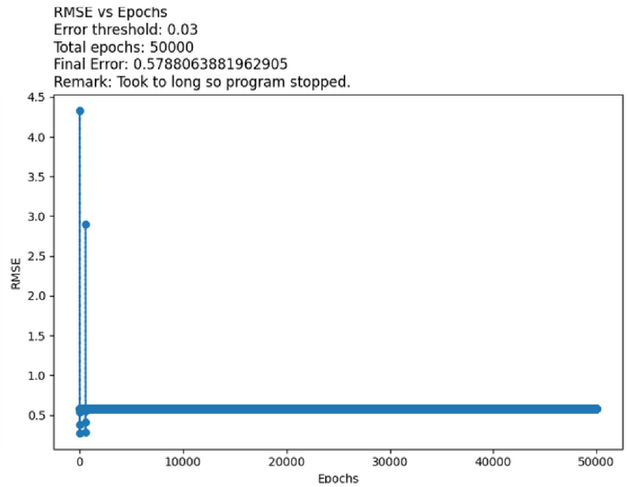








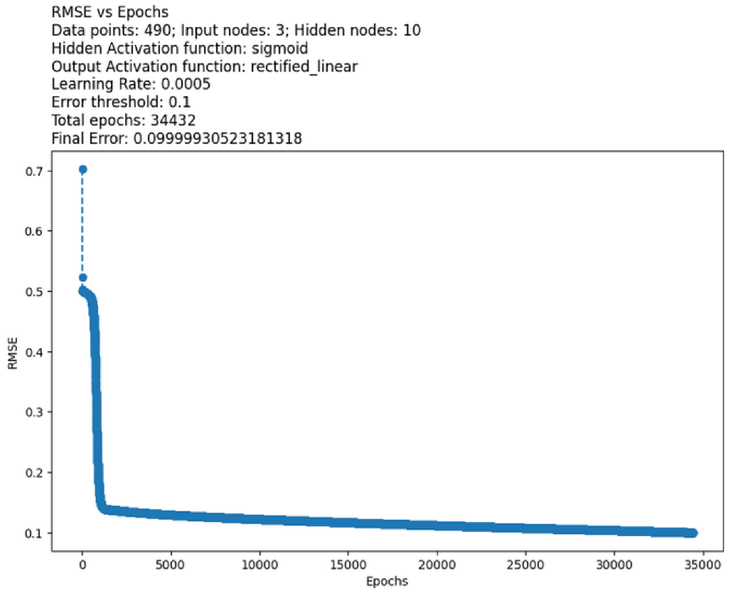




From the resutls, it can also be seen that not all combinations yield an accurate result. There are still unsuccessful learning attempts because the of the error exploding, being unable to go below the threshold error, or because of overflow errors. But there are still successful combinations and these combination are the (in the format of hidden-output): sigmoid-sigmoid, sigmoid-rectified linear, and linear-sigmoid. Also, exactly the same combinations yield successful results when learning the sine function.

Fruit Classification

Using my neural network, I used it to classify apples and bananas. I used the same data in activity 8, which are the eccentricity and the average *g* value of the fruits. With these features, I used them as an input for my neural network. The network has an input of 3 nodes (bias, average *g* value, and eccentricity), 490 data points, 10 hidden nodes, 1 output, 0.0005 learning rate, 0.1 error threshold, and a sigmoid activation function for the hidden layer and a rectified linear activation function for the output layer. I also set the desired value of the banana to be 1 and the apple to be 0. With these parameters, my results can be seen below:



Accuracy of neural network for classifying bananas: 100.0 %

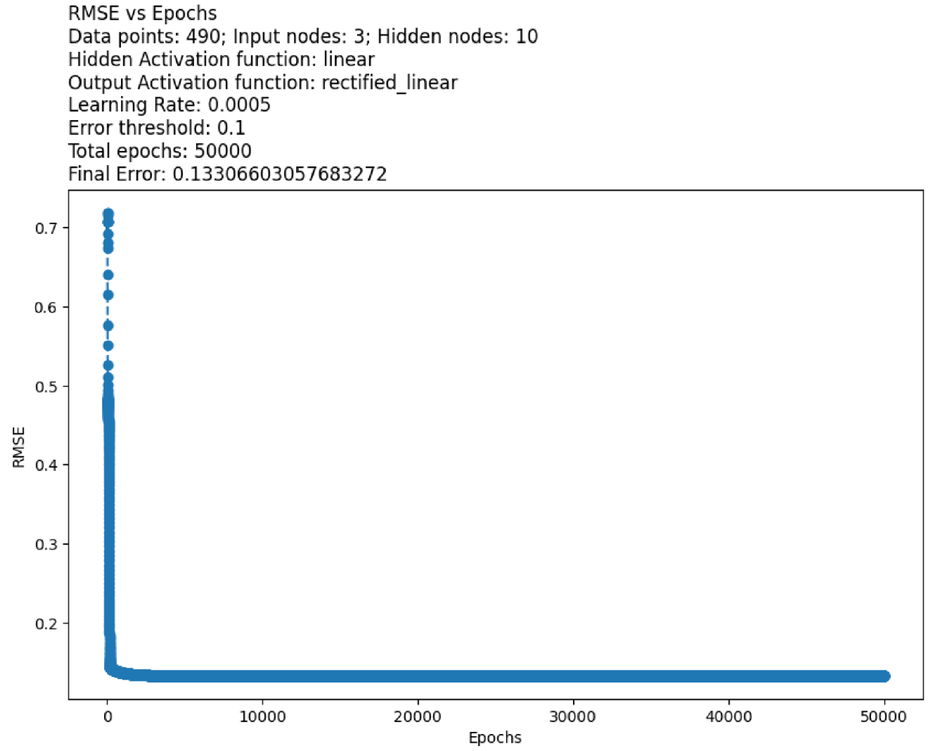
	Test image	Actual Classification	Neural Network Prediction
0	Test_image_1	Banana	Banana
1	Test_image_2	Banana	Banana
2	Test_image_3	Banana	Banana
3	Test_image_4	Banana	Banana
4	Test_image_5	Banana	Banana
...	...	...	...
240	Test_image_241	Banana	Banana
241	Test_image_242	Banana	Banana
242	Test_image_243	Banana	Banana
243	Test_image_244	Banana	Banana
244	Test_image_245	Banana	Banana

Accuracy of neural network for classifying apples: 100.0 %

	Test image	Actual Classification	Neural Network Prediction
0	Test_image_1	Apple	Apple
1	Test_image_2	Apple	Apple
2	Test_image_3	Apple	Apple
3	Test_image_4	Apple	Apple
4	Test_image_5	Apple	Apple
...	...	...	...
240	Test_image_241	Apple	Apple
241	Test_image_242	Apple	Apple
242	Test_image_243	Apple	Apple
243	Test_image_244	Apple	Apple
244	Test_image_245	Apple	Apple

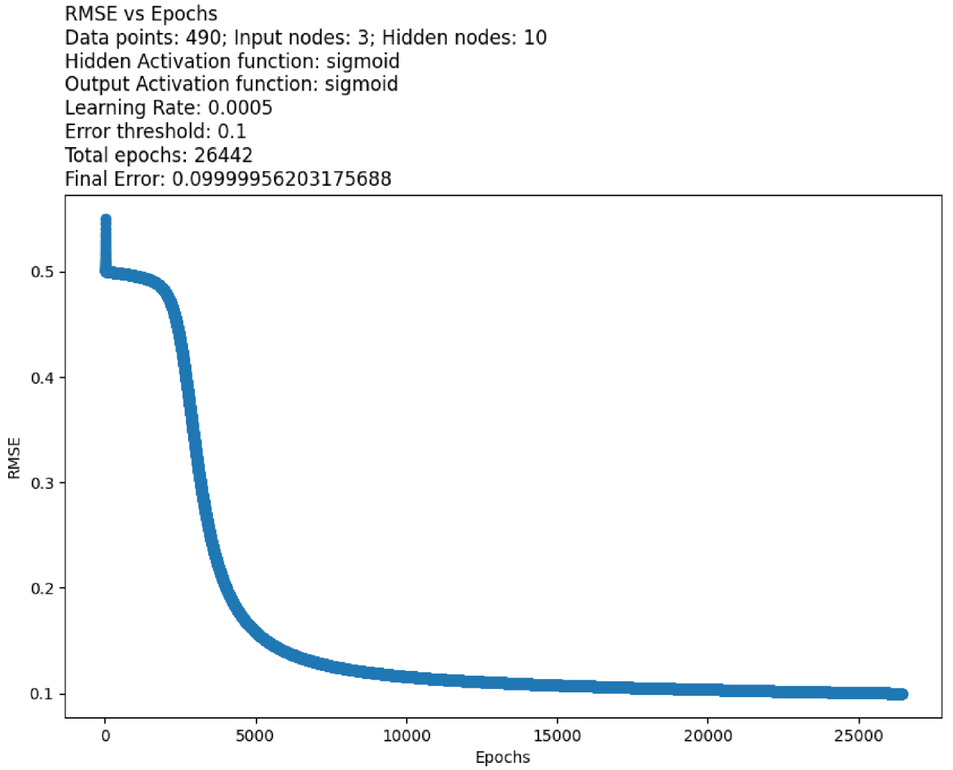
From my results, it can be seen that my neural network has successfully classified apples and bananas based on their eccentricity and average *g* value. I then tried using other different combinations of activation functions based on the ones that worked in learning functions. The following are my results:

linear - rectified linear




```
Accuracy of neural network for classifying bananas:  99.59183673469387 %
Accuracy of neural network for classifying apples:  100.0 %
```

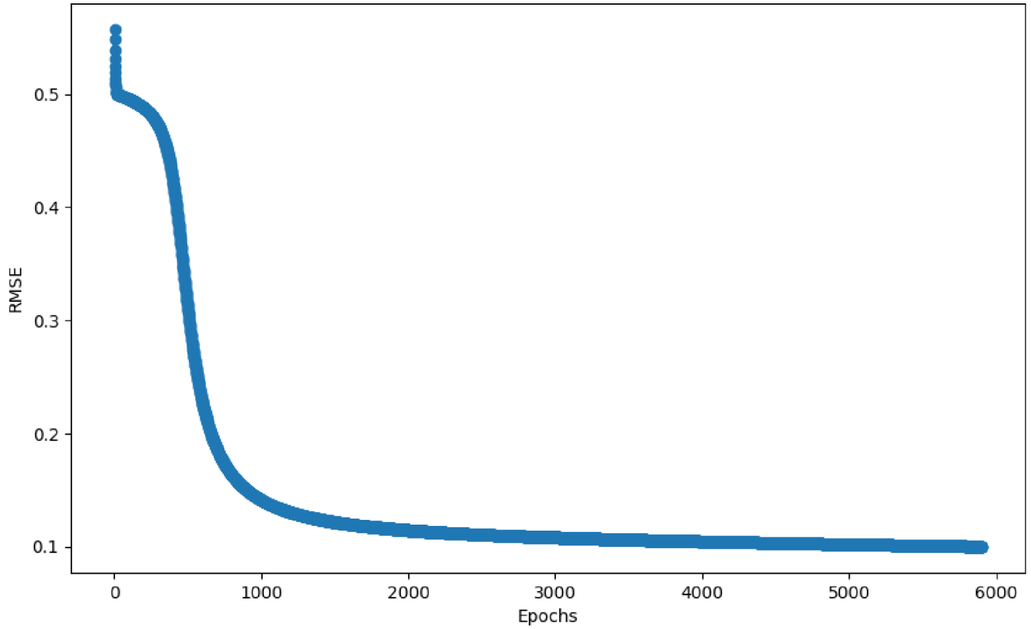
sigmoid - sigmoid




```
Accuracy of neural network for classifying bananas:  100.0 %
Accuracy of neural network for classifying apples:  100.0 %
```

linear - sigmoid

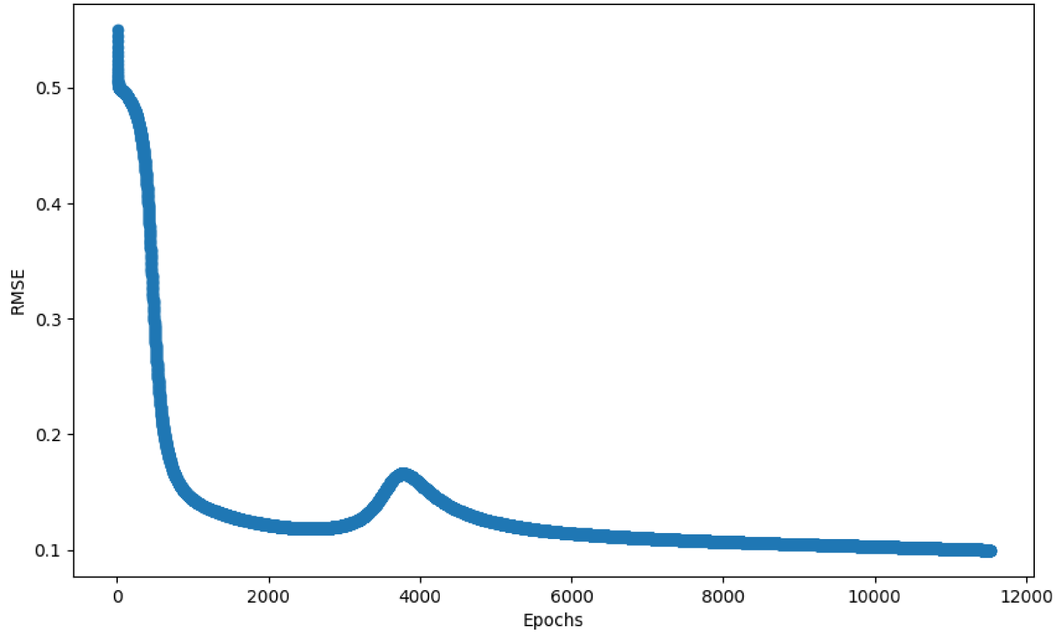
RMSE vs Epochs
Data points: 490; Input nodes: 3; Hidden nodes: 10
Hidden Activation function: linear
Output Activation function: sigmoid
Learning Rate: 0.0005
Error threshold: 0.1
Total epochs: 5904
Final Error: 0.09999877551540132




```
Accuracy of neural network for classifying bananas:  100.0 %
Accuracy of neural network for classifying apples:  100.0 %
```

tanh-sigmoid

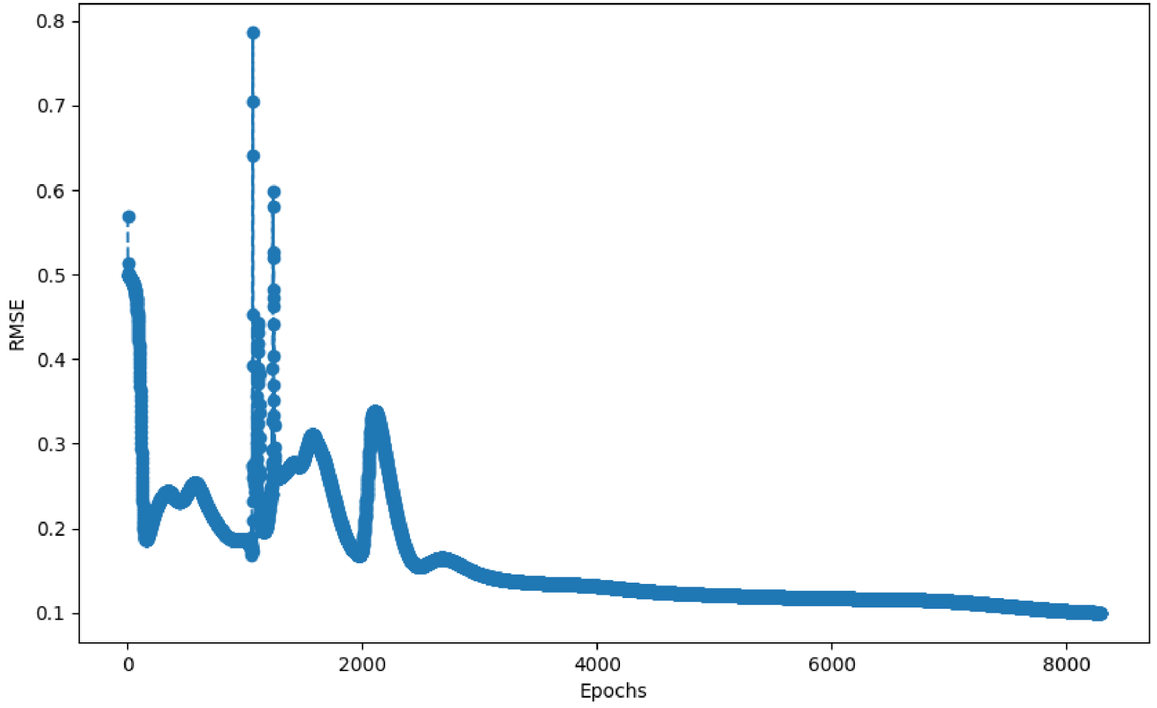
RMSE vs Epochs
Data points: 490; Input nodes: 3; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: sigmoid
Learning Rate: 0.0005
Error threshold: 0.1
Total epochs: 11528
Final Error: 0.09999988818643052




```
Accuracy of neural network for classifying bananas:  100.0 %
Accuracy of neural network for classifying apples:  100.0 %
```

tanh - tanh

RMSE vs Epochs
Data points: 490; Input nodes: 3; Hidden nodes: 10
Hidden Activation function: tanh
Output Activation function: tanh
Learning Rate: 0.0005
Error threshold: 0.1
Total epochs: 8291
Final Error: 0.09999566509388093




```
Accuracy of neural network for classifying bananas:  100.0 %
Accuracy of neural network for classifying apples:  100.0 %
```

Looking at the results, it can be seen that my neural network has successfully classified the apples and bananas. Almost all of the activation function combinations that I tried obtained a 100% accuracy. Only the linear-rectified linear combination got a 99.59% accuracy for the classification of the bananas. But the other combinations, even the ones that did poolry in learning the sine wave and linear function, achieved 100% accuracy. Their only difference is the number of epochs that they took to minimize the error below the threshold. The fastest one so far is the linear - sigmoid combination.

Reflection

Overall, this was a really fun activity. I fully understood how neural networks work and after 12 hours of reading the manual over and over again and reading different online references, I finally understood the math and realized that it was actually really simple and straightforward.

For this activity, I believe that I got the correct results. The neural network was able to learn the sine wave and the linear function and it was able to classify apples and bananas based on their eccentricity and avaerage *g* values with high accuracy. The learning and classifications are correct, because their RMSE converged to low values. But in the cases of the RMSE diverging, it just indicates that the combination of activation functions were not suited for that specific case.

The parts that I got stuck on was understanding the math. It was really hard visualizing the matrices for the error back propagation part. But in the end, I was able to understand it and implement it in python.

I'd like to thank my instructors, Sir Rene Principe Jr. and Sir Kenneth Leo, for guiding me throughout the activity. I would also like to thank my professor, Ma'am Jing, for guiding me in my coding while my classmates and I worked in R202. I would also like to acknowledge my classmates: Abdel, Johnenn, Jonabel, Richmond, Lovely, Hans, Genesis, Jeruine, Rusher, and Ron for helping me complete this activity.

Self Grade

Technical Correctness	I understood the lesson and met all the objectives. My results are complete and I got the expected results.	35
Quality of Presentation	The images I added to this report are of good quality and all the graphs are properly labelled. My code is also properly organized and labelled.	35
Self Reflection	I got the expected results, and acknowledged the contributions of my peers while doing this activity. I also properly cited online references.	30
Initiative	Apart from doing the required tasks, I also helped my classmates with their code and helped them by cross-referencing my results with theirs.	10
	Total	110

References

[1] Soriano, M. (2020). ML5 - Neural Networks. Applied Physics 157 Laboratory Manual

[2] Pramoditha, R. (2022, January 26). How to choose the right activation function for neural networks. Medium. https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c