Developing the Network

Using the functional API and multiple online resources, I was able to get a model that I believe works efficiently well enough for the data set that we were given.

Network Configuration

After the data set was one hot encoded, the dataset contained 17 feature columns and 3 label columns. The training data was indexed into numpy arrays with the 17 feature columns being used as training and each separate index of a label was used as testing data. The network is built with an input shape of 17 to handle all of the feature columns and 3 hidden layers with each giving an output for a label column. With the first layer corresponding to the first label, and the second layer handling the next, etc.

First layer: 24 neurons, outputs math
Second layer: 24 neurons, outputs reading
Third layer: 24 neurons, outpears writing

The following formula was used to decide the amount of neurons.

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

The formula was found in a comment on a forum post asking for the correct number of neurons a hidden layer should have. I defaulted to 128 neurons, but after running this formula, I chose 24.

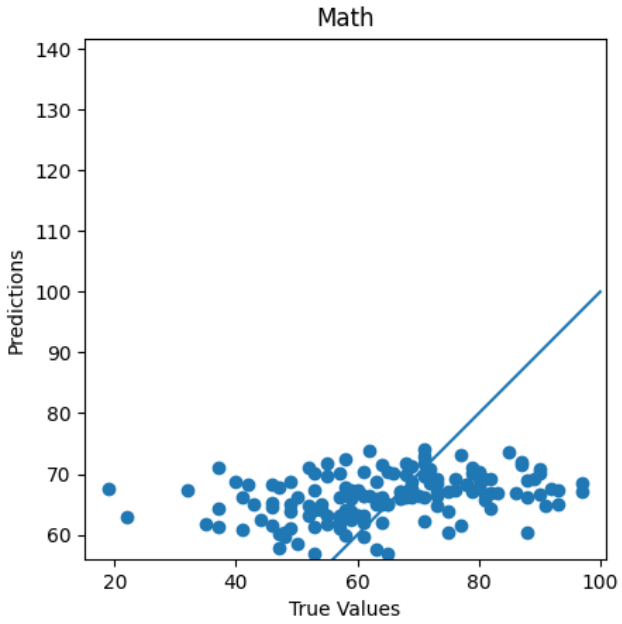
850/(2*(17+3)) = 21.25.

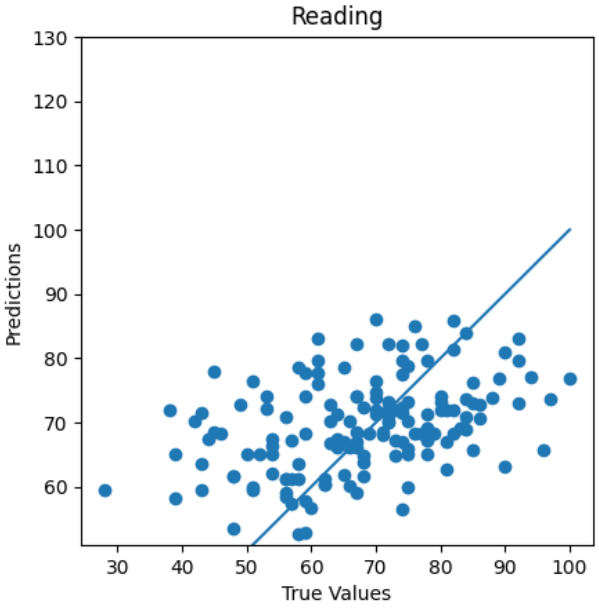
The formula was created by the user after looking at other resources and can be used as a starting point in supervised learning models. This model gets similar results whether the neurons are 10, 24, 128, or more; epochs needed vary slightly. 24 was a good middle ground that needed only ~35 epochs and since it is computationally faster than 128, 256, etc., and it gets the same results, then it seemed like a good solution.

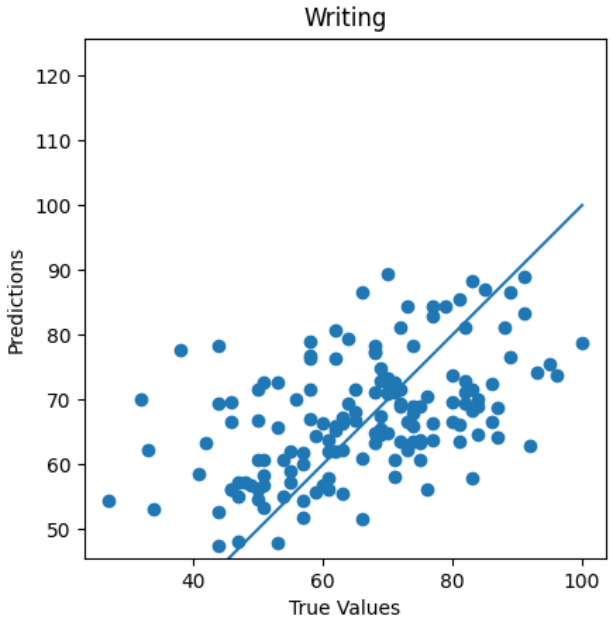
Optimizers also seem to give similar results; SGD, Adam, RMSprop were tried out.

Here are some outputs from the network:

loss: 590.3487548828125
math_loss: 186.17633056640625
math_rmse: 13.644643783569336
reading_loss: 198.557373046875
reading_rmse: 14.091038703918457
writing_loss: 205.6150665283203
writing_rmse: 14.33928394317627







Some printed predictions from the Math section. Full output attached in txt file.

True: 65.0 Prediction: [63.684853] Difference: [1.3151474]
True: 81.0 Prediction: [65.1229] Difference: [15.877098]
True: 58.0 Prediction: [69.22977] Difference: [11.229767]
True: 59.0 Prediction: [69.012314] Difference: [10.012314]
True: 58.0 Prediction: [58.26319] Difference: [0.26319122]
True: 40.0 Prediction: [64.581535] Difference: [24.581535]
True: 65.0 Prediction: [68.60487] Difference: [3.6048737]
True: 49.0 Prediction: [67.479996] Difference: [18.479996]
True: 61.0 Prediction: [61.44631] Difference: [0.44630814]
True: 64.0 Prediction: [62.479736] Difference: [1.5202637]
True: 84.0 Prediction: [70.41287] Difference: [13.587128]
True: 89.0 Prediction: [69.29473] Difference: [19.705269]
True: 50.0 Prediction: [60.889187] Difference: [10.889187]
True: 68.0 Prediction: [61.07917] Difference: [6.92083]
True: 65.0 Prediction: [65.1229] Difference: [0.12290192]
True: 62.0 Prediction: [63.590584] Difference: [1.5905838]

Math Acc: 0.5133333333333333
Read Acc: 0.58
Write Acc: 0.5666666666666667

Accuracy was measured in a separate loop after the program ran, counting as correct if the prediction was within 10% of the true score. The model gets an accuracy of ~50-60%. But the output accuracy of the model itself, measured as the root mean square error, falls between 12-15 after running the program.

After using the min-max normalizer on the dataframe to make better use of mean absolute percentage error. These are the results after running the program.

Math MAPE: 24.579123
Reading MAPE: 31.681267
Writing MAPE: 30.341051

MAPE seems to fall between the values of 20-40.

Future Exploration

The model has a hard time predicting high scores, ~90 or above and low scores, ~40 and below since they are not as common. A RMSE of 12-15 does not seem to be that good, but it also does not seem to be that bad considering training had 850 samples. The main problem that could be affecting the project result is the amount of data available. Having 5,000-10,000+ rows would likely reduce RMSE to ~10 or below. Having 100,000 rows would be interesting to see how low the RMSE can get and if this model will work. Perhaps the model is not set up correctly and is not working as initially intended.

References:

https://www.youtube.com/watch?v=HVJAHAbdG_s

https://www.youtube.com/watch?v=NMGsIsR7gR4 ---> https://colab.research.google.com/drive/1sZqFWkWTmv-htvL7OwiFMHX022Gy0Syf

https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw