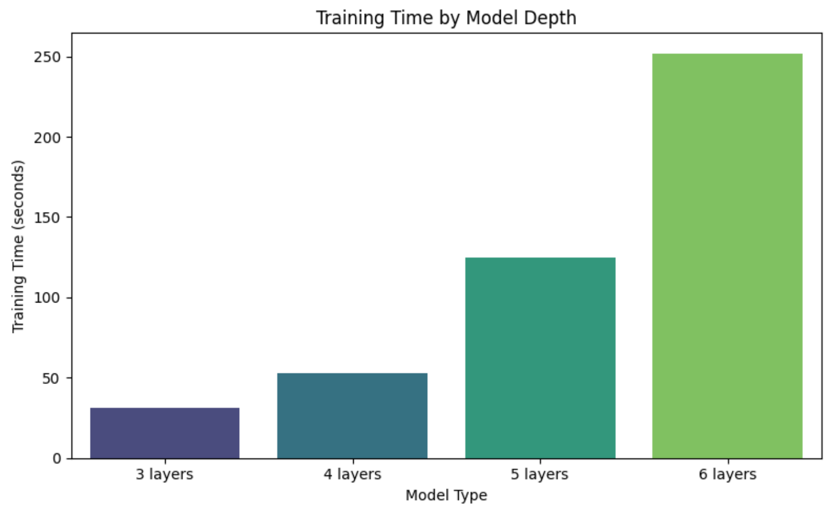
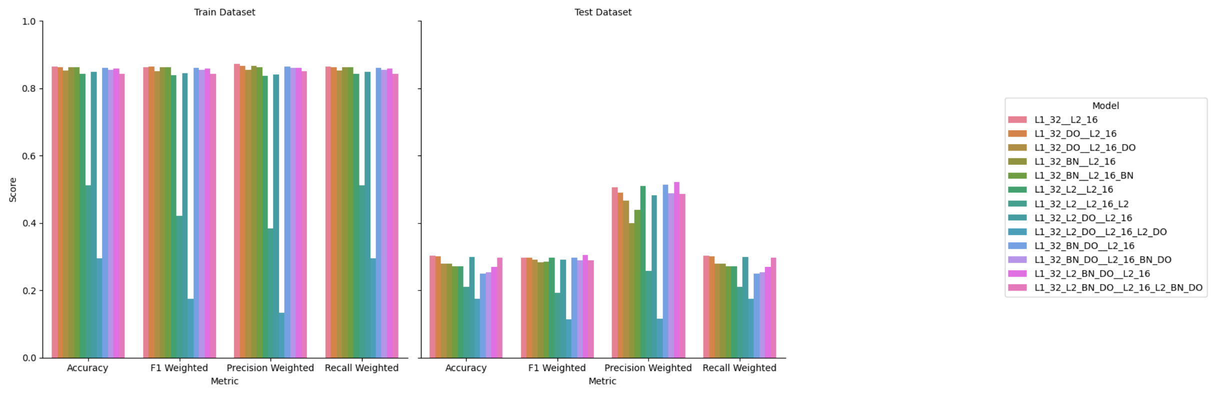
Total number of parameters in the initial model: 1,250,888
Number of layers used in the initial architecture: 3 layers (2 hidden layers 1 output layer)
Model improvement analysis:

- Training time increases when the hidden layer increases.

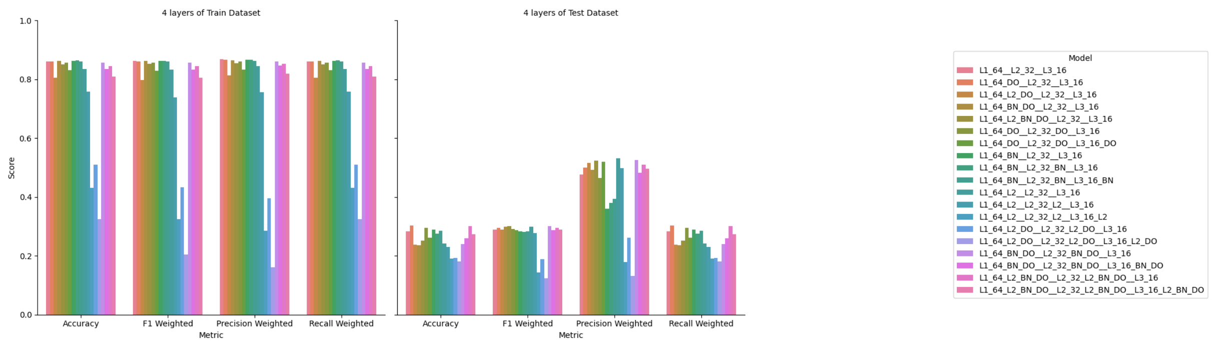


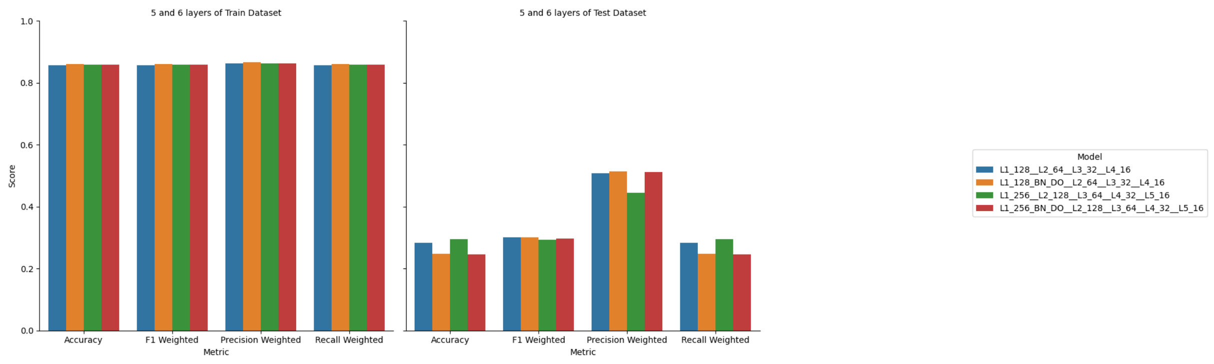
I tested various configurations on a 3-layer model by adding dropout, batch normalization, and L2 regularization. Based on the results, none of these techniques improved model performance compared to the original architecture. The figure below summarizes the results of using 3 layer models with different hyperparameter, using the naming convention:
L1_<units>[_L2][_BN][_DO]__L2_<units>[_L2][_BN][_DO]
Each suffix indicates whether L2 regularization (L2), batch normalization (BN), or dropout (DO) was applied to that layer. Overall, the original model configuration (L1_32__L2_16), without any of these additional hyparameter, achieved the best performance.

- 

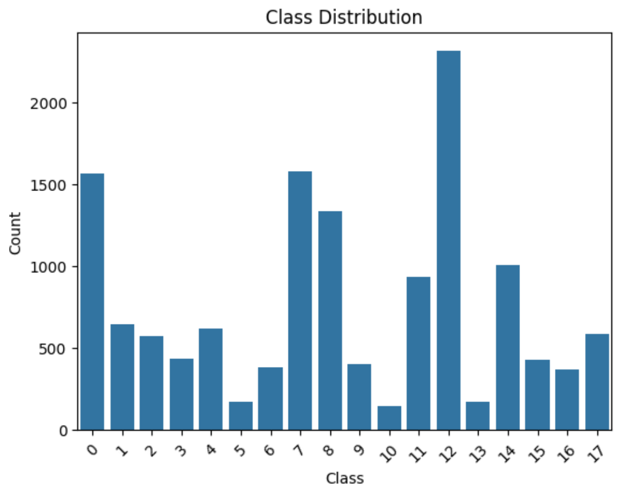
I also tested deeper models with 4, 5, and 6 layers, but the performance did not improve. The training metrics—including accuracy, F1 score, precision, and recall—remained around 80%, while the test performance stayed low, with accuracy, F1 score, and recall around 23%, and precision around 40%. These results are comparable to the performance of the 3-layer model, indicating that increasing the number of layers did not lead to better generalization.

All the models are overfitting. Training is better than test performance.





Since the model side cannot improve the performance and the root cause is due to imbalanced data because some classes have significantly more samples (e.g., ~1500) while others have very few (near 0), creating an unequal distribution.



I think the training data is needed to add more data for those imbalance class to improve the model.