Deep Learning Model for Cancer Type vs Drug Effectiveness/Sensitivity

Tianyi Chen

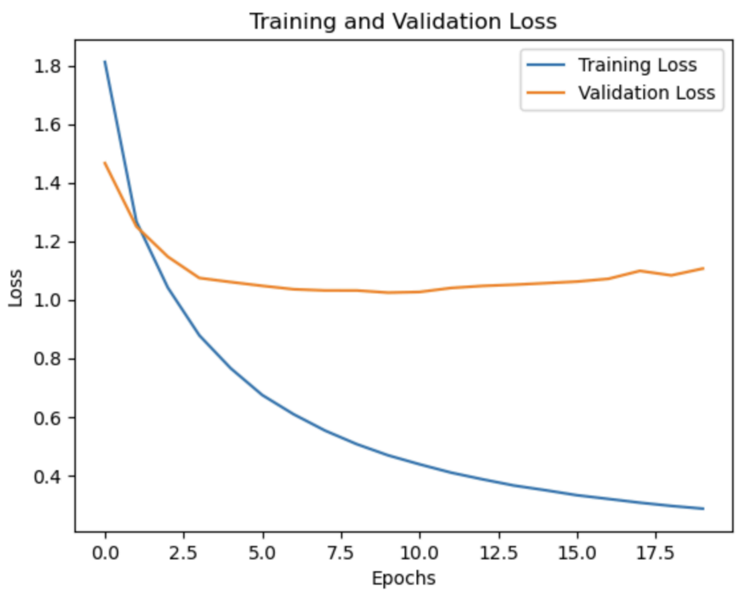
Given two features– Cell Line (Cancer Type) and Drug Name, we aim to create a Machine Learning model that predicts the IC50 value (measure of drug effectiveness on cancer cells) given a drug type and a cancer type.

Initially I had planned to utilize a Random Forest Regressor model on our dataset with SciKit Learn. However, due to the sheer size of our dataset, it was not possible to effectively create a model– the execution would overload my CPU and crash the Jupyter Notebook Interface. Instead, I decided to utilize TensorFlow to create a deep neural network that learned by itself.

The methodology was fairly simple: extract the columns in the dataset pertaining to the Cell Line and Drug Type and append those into a single "features" dataframe; we also extract the IC50 values (dependent variable). To make the data easier to process for the model, we one-hot encode the data and normalize it, then feed both the features and the dependent variable into a train-test-split function. After splitting our data for testing and training, we fit a sequential deep neural network with three layers to the data.

Now we can analyze the model performance based on several metrics:

I. Baseline Loss versus Test Loss– the test loss is determined by the mean squared error, the difference between the predicted values of the model and actual values in our test data. The baseline loss is a standardized comparison loss value which is the mean squared error of the differences between each of the y_test values and the mean of the y_test values.

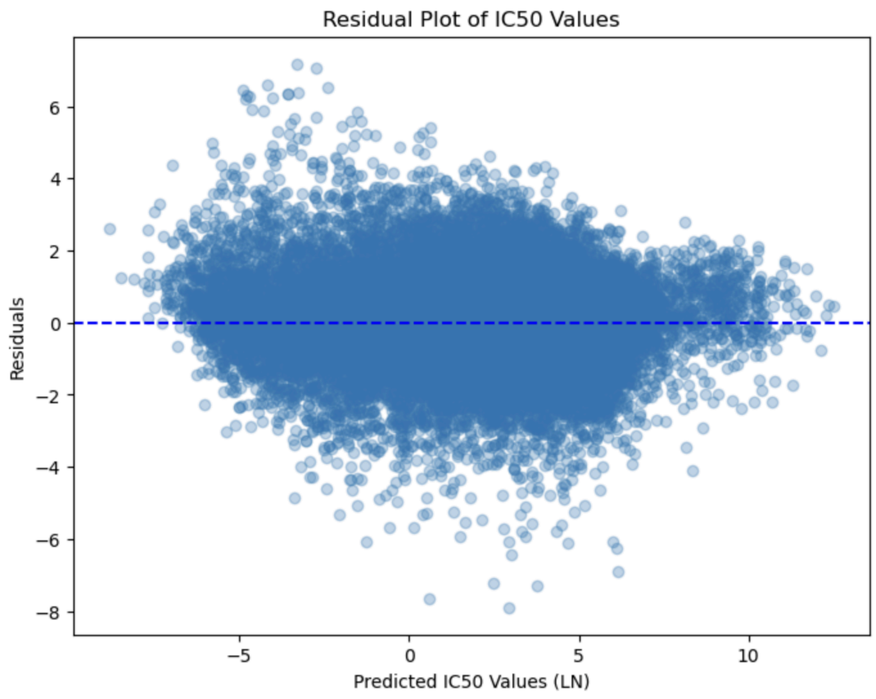


Baseline Loss:LN_IC50 7.662797
Test loss =1.1370331048965454

With a test loss that is significantly smaller than the baseline loss, it can be inferred that the model is performing well.

II. Training vs. Validation Loss– We have taken 20% of our training data and split it into a validation set which is used to verify the accuracy of our model. After plotting the training loss versus the validation loss, we see that training loss decreases significantly over epochs (every time we run through the training set in entirety) while the validation loss stays about the same. This is a good sign, as it means that our model increasing in accuracy over epochs and is not overfitting to the training set (if that were the case, the validation loss would increase drastically as the model increases in accuracy for the training set, diverging from the values of the validation set).

III. Residual Plot– An ideal residual plot of the data would be a random scattering of points close to the horizontal axis at zero. This randomness is a good indicator that the model is making accurate predictions and not overfitting or underfitting to the data. Looking at our model, we see that the residuals are centered at zero, indicating an unbiased model. However, the elongated “potato” shape of the model could possibly indicate an underlying relationship between the data that the model has failed to capture. This should be a minimal issue, however, because there is no visible pattern in the oval such as general clusters of points or extreme curvatures. One possible explanation could be that the model works better on smaller IC50 values than large IC50 values, creating a denser clustering near zero. Investigating this further could yield more information.


Residual Plot of IC50 Values
Residuals
Predicted IC50 Values (LN)