Alonso Salcido, Chris Haub

Neural Networks

**Introduction**

Neural networks are a class of algorithms used in machine learning and data science for modeling complex relationships between inputs and outputs. They are inspired by the structure and function of the human brain. Neural networks consist of layers of interconnected nodes, where each node represents a neuron that receives input, processes it, and generates an output. By adjusting the strength of the connections between neurons and the biases of the neurons, the network learns to make predictions or classifications based on input data.

**Methods and Results**

For this project we were given a compressed CSV dataset that when unzipped was 5 gigabytes. The data shape was 29 columns, 7 million rows. A binary classifier was used from 28 feature variables (predictor variables). The 28 features used were continuous variables. StandardScaler was used to standardize all features. This was done to improve the performance of the machine learning model.

The target variable “label” was balanced almost evenly at 50.01% and 49.99% respectively. Matplotlib, was used to create a bar chart that displays the percentage of different label values in a given variable. This visualization technique is particularly useful when dealing with categorical data where each unique value in a column has a label attached to it.

Chart, bar chart

Description automatically generated

Since the dataset at used was extremely large a smaller dataset was used to test parameters for the classifier. 30% of the dataset was used for efficiency during testing. The reduced dataset was then broken up into 80/20, train/test set. The selection of appropriate activation functions, layers, and optimizers was dependent on the data, and required careful consideration to ensure that the neural network was able to effectively learn and generalize from the input data. Once sufficient parameters were obtained the full dataset was then used.

The final models were run on the full dataset and split 80/20, train/test split. Our training dataset consisted of 5.6 million rows and 28 columns, while the validation dataset had 1.4 million rows. The Keras library in TensorFlow was used to define the neural network model. The model consisted of several 7 hidden layers, and an input and output layer. The input layer expected input data with a shape of (28,), while the output layer had a single neuron with a sigmoid activation function, which is appropriate for binary classification problems. Each dense layer had a different number of neurons, with the first layer having 50 neurons, and the subsequent layers having 80 and 100 neurons respectively. All the dense layers used the ReLU activation function, which helps introduce non-linearity into the model. The kernel\_regularizer argument was set to l2(0.001), which applied L2 regularization to the kernel weights of each dense layer, helping to prevent overfitting. Stochastic gradient descent optimizer was used with binary cross-entropy loss function, the drawback to this is time is takes to run SGD.

The model was set to run an extremely large number of epochs (1000) with a batch size of 15. This was done to make sure that the early stopping criteria is used. The model was set to stop once the validation loss stopped improving. The patience argument was set at 2 for EarlyStopping. Once the loss did not improve in 2 subsequent epochs the run was stopped.

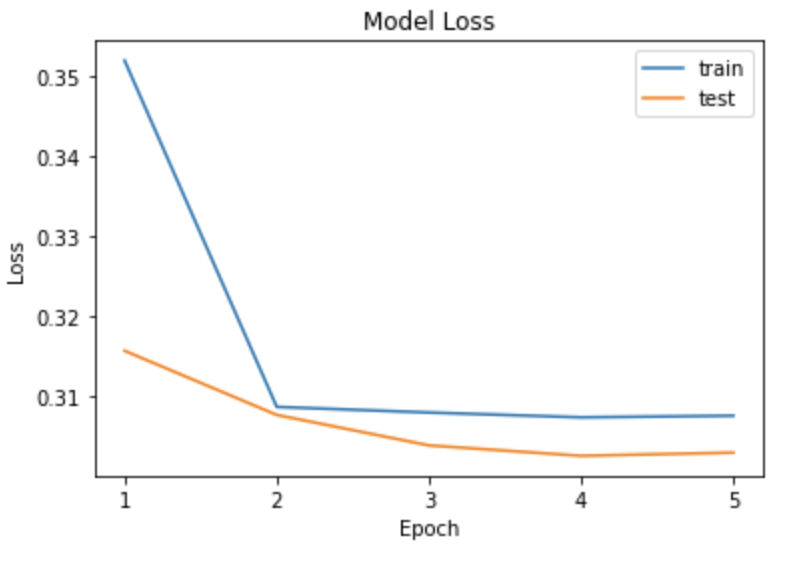
The final model was able to achieve a maximum validation accuracy of 0.8704. This model is the one that had the best accuracy of all the models trained using the reduced dataset. The drawback of this model is that once it was used on the full dataset it was extremely slow to run with a sinlge epoch taking longer than an hour to run. An interesting finding is that the highest climb in accuracy and lowest decline in loss between epochs happened between epochs 1 and 2. The epochs after epoch 2 didn’t improve our model substantially and took a long time to run. In other words, if we were to run this model again on the full dataset, it might be a good idea to stop it after epoch 2, instead of using an early stopping. Gains after epoch 2 are very small compared to the time it takes to run.

A picture containing graphical user interface

Description automatically generated

Chart, line chart

Description automatically generated



**Appendix**Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated