**Neural Network Model Analysis**

**Overview**

We made a deep learning model for a nonprofit foundation called Alphabet Soup. The model is supposed to predict if an organization will use the funding they get effectively. We used different features from the dataset to make these predictions.

### **Step 1: Preprocess the Data**

Data Processing: We first cleaned up the data by removing some unnecessary information such as dropping the NAME columns and the EIN. These were removed because they are neither targets nor relevant/helpful features for the prediction. Then we split the data into training and testing sets.

Target Variables: te target is IS\_Successful, which shows if the funding that the organization received was yes – (1) or no – (0).

Feature: The following coumns from the data set were te features that were used in this model. APPLICATION\_TYPE, INCOME\_AMNT, AFFILIATION, USE\_CASE, ORGANIZATION, STATUS, CLASSIFICAITON, and SPECIAL\_CONSIDERATIONS.

### **Step 2: Compile, Train, and Evaluate the Model**

**Model Architecture**

Neurons & Layers: In this model is a neural network with two hidden layers. The first layer had 80 neurons and the second layer had 30 neurons. You have to play around with the numbers of neurons to balance out the model, so these numbers were an outcome of a trial and error.

Functions: this model uses two types of activation functions. For the hidden layers, we used the Rectified Linear Unit (ReLU) function. This function is good at dealing with data that isn’t straight or linear. For the final output layer, we used the Sigmoid function. This function is good for problems where we have to choose between two things, like in our case where we’re choosing between ‘yes’ (1) and ‘no’ (0).

**Model Performance**

A screenshot of a computer

Description automatically generated

Performance Results: as you can see that 3298 parameters were created by a three-layer training model. This model was able to hit the goal of at least 75% accuracy with this model but it actually has approximately 78% accuracy. This means it’s pretty good at predicting if an organization will use their funding successfully considering the model was right about 78% of the time.

### **Step 3: Optimize the Model**

In order to improve the model performance, we tried a few different things:

**Feature Engineering**: We changed the way we represent some of our data. We used something called one-hot encoding for our categorical variables. This made it easier for the model to understand and use this type of data.

**Hidden Layers and Neurons**: We played around with different numbers of hidden layers and neurons. In the end, we decided on a model with two hidden layers - one with 80 neurons and one with 30 neurons. This seemed to give us the best balance between a model that was too simple and one that was too complicated.

**Activation Functions:** We tested out different activation functions for the hidden layers. We found that the Rectified Linear Unit (ReLU) function worked well for capturing complex patterns in the data.

**Epochs and Batch Size**: We tweaked the number of training epochs and the batch size to fine-tune our model. By increasing the number of epochs, we gave the model more chances to learn from the data.

**Saving the Model:** We set up a system to save the model’s weights every five epochs. This let us go back and look at the best-performing model.

**Results**

The deep learning model we made did a good job at predicting if an organization would use their funding effectively with approximately 78% accuracy. We were able to achieve this by cleaning up the data, choosing the right model architecture, and fine-tuning some things along the way the model was able to create a reliable tool to make informed decisions on effectively allocating the organization funding.