Alphabet Soup Charity – Deep Learning Challenge Analysis

Overview:

Alphabet Soup has funded many organizations in the past and is interested in a tool to help choose future opportunities with the best chance for success. The purpose of this analysis is to model how successful Alphabet Soup investments have been and create a binary (Successful/Not Successful) conclusion based on the inputs. The past data contained over 34,000 organizations and their funding information for the purposes of this analysis.

Results:

* Data Preprocessing
  + The target variable for the analysis is “IS\_SUCCESSFUL” which tells us if the investment was successful or not.
  + The initial feature variables are:
    - “APPLICATION\_TYPE”
    - “AFFILIATION”
    - “CLASSIFICATION”
    - “USE\_CASE”
    - “ORGANIZATION”
    - “STATUS”
    - “INCOME\_AMT”
    - “SPECIAL\_CONSIDERATIONS”
    - “ASK\_AMT”
  + The “EIN” and “NAME” variables were dropped before the analysis began as they were insignificant to the outcome of the target.
* Building the Model
  + The initial model was built with two layers, both with the ‘tanh’ activation function.
  + The first layer had 79 neurons and the second had 30.
    - I knew I wanted to start with a higher number of neurons for the first layer because there were a lot of input features and we needed enough neurons to handle that much data. The second layer was a little less than half the first.
  + The initial model failed to hit a 75% accuracy.
    - Loss: 0.5589
    - Accuracy: 0.724, adjustments needed to be made.
* Optimizing the Model
  + On the second attempt at a proficient model, some perceived extraneous features were dropped from the data set.
    - The dropped feature variables:
      * “APPLICATION\_TYPE”
      * “STATUS”
      * “SPECIAL\_CONSIDERATIONS”
    - With fewer feature inputs now, the neurons were lowered to 45 for the first and 20 for the second. The activation function remained ‘tanh’.
    - The second model performed poorer than the first.
      * Loss: 0.5813
      * Accuracy: 0.7053
      * something was changed for the worse rather than better
  + A third attempt at optimization was made, this time with the Keras Tuner.
    - Using the same data as the initial model, the Keras Tuner was used to find the optimal hyperparameters and the “best” model with that data set.
    - Again, however, the model failed to beat the 75% accuracy threshold.
      * Loss: 0.5516
      * Accuracy: .7279, just slightly better than the original guess at parameters
  + Further optimization could possibly be done by trial and error, likely but using the Keras Tuner and dropping/re-adding features one-by-one, but that would take many iterations of this process and the results might only be marginally better.