Adrian Stahl
Deep Learning Challenge
Module 21

Deep Learning Challenge: Charity Funding Predictor

(1) Overview of the Analysis:
Alphabet Soup wants to create an algorithm to predict if applicants' funding will be successful or not. Through the use of machine learning and neural networks, we will create a binary classifier that can predict the latter.

(2) Results:

Data Preprocessing

- What variable(s) are the target(s) for your model?
  "IS_SUCCESSFUL" which has a value of 1, and 0 if it was not.
- What variable(s) are the features for your model?
  "APPLICATION" data was analyzed, while "CLASSIFICATION" was used for binning.
- What variable(s) should be removed from the input data because they are neither targets nor features?
  We dropped "EIN" and "NAME" as they contained irrelevant information. Consequently, "NAME" was then added back into a second test for binning purposes, and it was then split for training and testing purposes.

Compiling, Training, and Evaluating the Model
- How many neurons, layers, and activation functions did you select for your neural network model, and why?
  There were 3 layers for each model after applying Neural Networks.

Adrian Stahl
Deep Learning Challenge
Module 21

- Were you able to achieve the target model performance?

Compile, Train and Evaluate the Model

```
[14] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len( X_train_scaled[0])
hidden_nodes_layer1=7
hidden_nodes_layer2=14
hidden_nodes_layer3=21

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 350

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 477
Trainable params: 477
Non-trainable params: 0
_________________________________________________________________
```

```
[17] # Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")

268/268 - 0s - loss: 0.5532 - accuracy: 0.7294 - 241ms/epoch - 898us/step
Loss: 0.5532000660896301, Accuracy: 0.7294460535049438
```

477 parameters were created by the three-layered training model. The first attempt was at 73% Accuracy so it did not reach the 75% goal.

Adrian Stahl
Deep Learning Challenge
Module 21

- What steps did you take in your attempts to increase model performance?

```
[19]
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len( X_train_scaled[0])
hidden_nodes_layer1=7
hidden_nodes_layer2=14
hidden_nodes_layer3=21
nn = tf.keras.models.Sequential()

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 3171

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 3,298
Trainable params: 3,298
Non-trainable params: 0
_________________________________________________________________
```

```
[22] # Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.4643 - accuracy: 0.7915 - 363ms/epoch - 1ms/step
Loss: 0.46428781747817993, Accuracy: 0.7914868593215942
```

Deep learning models typically utilize multiple layers to enable effective prediction and classification of information. By employing a hierarchical structure, these models can efficiently process input data through various layers of computational filters, allowing them to learn and discern complex patterns for accurate predictions and classifications.