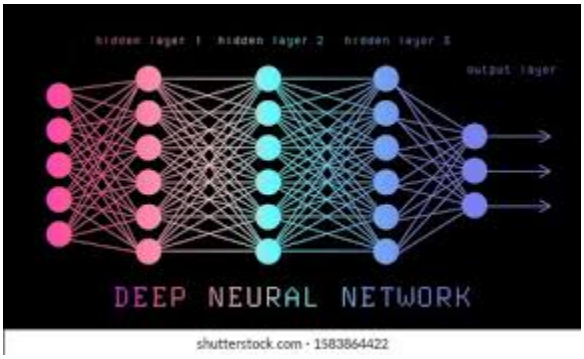
Neural Networks Analysis



Overview of the analysis: Explain the purpose of this analysis.

The purpose of the analysis for Alphabet Soup was to predict which applicants the company would approve for investments. Using machine learning and neural networks to apply target/features on the dataset, I will create a binary classifier capable of predicting whether investors would be successful if funded by Alphabet Soup.

The csv file contains 34,000 organizations that have received funding from Alphabet Soup over the years and has 12 columns showing the metadata regarding each organization and funding outcomes.

Results: Data Preprocessing

What variables are the targets for your model?

IS_SUCCESSFUL is the target that is marked 1 for successful while a 0 indicates an unsuccessful investment

What variables are the features for your model?

The remaining columns not including EIN and NAME were the features for the model

What variables should be removed from the input data because they are neither targets nor features?

EIN and Name were removed from the input data because they have largely irrelevant data

Compiling, Training, and Evaluating the Model

How many neurons, layers, and activation functions did you select for your neural network model, and why?

In the first training model I used 3 layers as these were given to us in the starter code - this gave us almost 6500 trainable parameters with 0 non-trainable parameters. I also used a 'relu' activation for the first two layers and 'sigmoid' for the output layer.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                4000

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 6461 (25.24 KB)
Trainable params: 6461 (25.24 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Result: Using 100 epochs gave us the following results:

```
268/268 - 0s - loss: 0.5616 - accuracy: 0.7270 - 302ms/epoch -
1ms/stepLoss: 0.5616334080696106, Accuracy: 0.7269970774650574
```

Model Optimization: Next I tried adding more hidden nodes to the first layer, increasing it from 80 to 100 while also changing the second activation to tanh, the hyperbolic tangent function, in order to see how this affected the model. I also reduced the number of epochs to 20 to increase model efficiency. This gave the following results:

```
268/268 - 0s - loss: 0.5528 - accuracy: 0.7261 - 406ms/epoch -
2ms/stepLoss: 0.5527952313423157, Accuracy: 0.726064145565033
```

Were you able to achieve the target model performance?

In my third model, I left the NAME variable in the data rather than deleting it and when I first tried to run this I was receiving an error message because the file was too big. After choosing several different cutoff values creating a list of names to be replaced, I settled on counts<100 - this gave 32 different categories/ bins which is a manageable number.I then re-ran the same model as before:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               7800

 dense_1 (Dense)             (None, 30)                3030

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 10861 (42.43 KB)
Trainable params: 10861 (42.43 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

I ran this with 20 epochs and this gave me the target model performance which was greater than 75% accuracy as can be seen in the results below:

```
268/268 - 0s - loss: 0.4967 - accuracy: 0.7523 - 403ms/epoch - 2ms/stepLoss: 0.49669381976127625, Accuracy: 0.7523031830787659
```

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

As noted above, I changed the amount of layers in the models and switched the activation types and the numbers of epochs around to try and increase the model performance. However, it was only after including the NAME variable that the model was able to predict with over 75% accuracy.

Summary

Adding layers and changing activation types can affect the results which a model will obtain. It is worth noting however that there was practically no advantage to using 100 epochs instead of 20 which was surprising to me as using 100 epochs is a lot more inefficient than 20.

It is possible that higher results may be achieved by further adding layers but the results were not getting incrementally higher with each epoch so this may have been a waste of time and resources. Random Forest or linear regression may be much more efficient ways to run the optimization models for Alphabet Soup and we should always start with simpler models before performing Neural Network analysis such as what was conducted in this assignment!