Topic 5 Assignment By Wilson Peguero Rosario

Problem Statement

In this assignment, practical training issues are considered. There will be three basic issues addressed: the first describes things that need to be done prior to training a network, such as collecting and pre-processing data and selecting the network architecture; the second addresses network training itself; and the final considers post-training analysis. A case study is constructed for pattern recognition.

Assume a produce dealer has a warehouse that stores a variety of fruits and vegetables. When fruit is brought to the warehouse, various types of fruit may be mixed. The dealer wants a machine that will sort the fruit according to type. There is a conveyor belt on which the fruit is loaded. This conveyor belt passes through a set of sensors, which measure three properties of the fruit: shape, texture, and weight. These sensors are somewhat primitive. The shape sensor will output a 1 if the fruit is approximately round and a -1 if it is more elliptical. The texture sensor will output a 1 if the surface of the fruit is smooth and a -1 if it is rough. The weight sensor will output a 1 if the fruit is more than one pound and a -1 if it is less than one pound. The three sensor outputs will then be input to a neural network. The purpose of the network is to decide which kind of fruit is on the conveyor, so that the fruit can be directed to the correct storage bin. To make the problem even simpler, assume that there are only two kinds of fruit on the conveyor belt: banana and pineapple.

Key Notes

- Client Requires algorithm for sorting fruit according to type
  - Categorical data based on whether the fruit is a citrus, etc.
- Conveyor belt passes the fruit through a sensor which extracts three properties from the fruit
  1. Shape
     - output is binary (1 if the fruit is round, -1 if the fruit is elliptical)
  2. Texture
     - output is binary (1 if the fruit is smooth, -1 if the fruit is rough)
  3. Weight
     - output is binary (1 if the fruit is more than 1 lb, -1 if the fruit is less than 1 lb)
- Assume that there are only two kinds of fruit on the conveyor belt
  1. Banana (shape:-1, texture:1, weight:-1)
  2. Pineapple (shape:-1, texture:-1, weight:1)

In [ ]:

```
import pandas as pd
import tensorflow as tf
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Data Collection/Preparation

Now that the relevant libraries have been updated, one should be able to replicate the data necessary to develop the Neural Network. The average Pineapple weighs approximately 2 lbs, whereas a banana weighs less than half a pound. This makes it safe to assume that the weight feature will be consistent throughout the entirety of the data set (there will not be many pineapples whose weight is less than 1 lb and there will not be many bananas that weigh more than 1 lb). Texture is also an inherent feature of pineapples and bananas, meaning that a banana will always be smooth whereas a pineapple will always be rough. The feature with the most variation may be in the terms of shape for pineapples.

To develop the data, numpy and pandas libraries were used to create random selection between the number 1 or -1 for each feature (shape, texture, and weight) and develop the data set.

In [ ]:
```
list__dataset = []
for _ in range(30000):
    shape = np.random.choice([-1,1], replace=False)
    texture = np.random.choice([-1,1], replace=False)
    weight = np.random.choice([-1,1], replace=False)
    if texture == 1 and shape == -1:
        label = "banana"
    elif texture == -1 or (texture == -1 and weight == 1):
        label = "pineapple"
    else:
        label = "banana"
    list__dataset.append({"label":label, "shape":shape, "texture":texture, "weight":wei

df__dataset = pd.DataFrame(list__dataset)
```

Data Exploration

Now that the data has been collected and formatted into the version apt for developing algorithms, it can be explored and some analytics can be taken from the data set.

In [ ]:
```
df__dataset.describe()
```

Out[ ]:

	shape	texture	weight
count	30000.000000	30000.000000	30000.000000
mean	-0.014467	-0.006000	-0.004067
std	0.999912	0.999999	1.000008
min	-1.000000	-1.000000	-1.000000
25%	-1.000000	-1.000000	-1.000000
50%	-1.000000	-1.000000	-1.000000
75%	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000

From the above, one is able to tell immediately that majority of the samples within the data set belong to the pineapples as the mean texture value is negative. There are indications that out of the fruits, most weigh more than 1 lb and their shape is mostly round. One can confirm this by running a summary of the labels (shown below).

In [ ]:
```
df__dataset['label'].value_counts()
```

Out[ ]:
```
pineapple    15090
banana       14910
Name: label, dtype: int64
```

Now one can create a copy of the original data set and exchange the values based on the table below:

Feature	Value	Definition
Shape	-1	Elliptical
Shape	1	Round
Texture	-1	Rough
Texture	1	Smooth
Weight	-1	<1 lb
Weight	1	>1 lb

In [ ]:
```
df__data_copy = df__dataset.copy()
df__data_copy['shape'].replace({1:"Round",-1:"Elliptical"}, inplace=True)
df__data_copy['texture'].replace({1:"Smooth",-1:"Rough"}, inplace=True)
df__data_copy['weight'].replace({1:">1 lb",-1:"<1 lb"}, inplace=True)
df__data_copy.head(10)
```

Out[ ]:

	label	shape	texture	weight
0	pineapple	Round	Rough	>1 lb
1	banana	Round	Smooth	>1 lb
2	pineapple	Round	Rough	>1 lb
3	pineapple	Round	Rough	>1 lb
4	banana	Elliptical	Smooth	<1 lb
5	pineapple	Round	Rough	>1 lb
6	pineapple	Elliptical	Rough	>1 lb
7	pineapple	Round	Rough	<1 lb
8	pineapple	Elliptical	Rough	>1 lb
9	banana	Elliptical	Smooth	<1 lb

Now that the data set has been converted to a more readable format, crosstabs can be created to estimate how many round/elliptical bananas there are vs pineapples. The same can be done for the

other features as well.

In [ ]:
```
ct01 = pd.crosstab(df__data_copy['label'], df__data_copy['weight'])
ct01
```

Out[ ]:

label	<1 lb	>1 lb
banana	7472	7438
pineapple	7589	7501

In [ ]:
```
ct02 = pd.crosstab(df__data_copy['label'], df__data_copy['texture'])
ct02
```

Out[ ]:

label	Rough	Smooth
banana	0	14910
pineapple	15090	0

In [ ]:
```
ct03 = pd.crosstab(df__data_copy['label'], df__data_copy['shape'])
ct03
```

Out[ ]:

label	Elliptical	Round
banana	7586	7324
pineapple	7631	7459

Now that one has better insight upon the labels and its features, one can create a correlation matrix to observe the correlation between the features.

In [ ]:
```
corr_matrix = df__dataset.corr()
corr_matrix\
    .style\
        .background_gradient(cmap='coolwarm', axis=None, vmin=-1, vmax=1)\
            .format(precision=2)
```

Out[ ]:

	shape	texture	weight
shape	1.00	-0.00	0.00
texture	-0.00	1.00	0.00
weight	0.00	0.00	1.00

As can be observed, there is little to no correlation between the features (this is due to its binary/polar nature).

Data Modelling

Now that one has finished exploring the data, it is time to develop the model. Based on its binary nature, one can develop either a hopfield neural network to classify the data. In this case, the hopfield neural network will not be used as this is meant to be an industrial grade model that predicts different kinds of fruits based on 3 features. In the case that one may wish to distinguish between more fruits than bananas and pineapples, one may experience a dramatic decrease in its ability to recall as well as its accuracy.

Now, the data requires separating the data set between a training set and a testing set.

In [ ]:
```
df__dataset['label'].replace({"banana":0, "pineapple":1}, inplace=True)
X_train,X_test,y_train,y_test = train_test_split(df__dataset.drop('label', axis=1), df_
X_train = np.array(X_train)
y_train = tf.one_hot(y_train, depth=2)
y_test = tf.one_hot(y_test, depth=2)
print(f"The shape of the training data set is {X_train.shape}.\nThe shape of the test d
```

```
The shape of the training data set is (21000, 3).
The shape of the test data set is (9000, 3).
```

Now that the data has been separated into its test and train data set, a machine learning model can be used for classification.

In [ ]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(3,1)),
    tf.keras.layers.Dense(3, activation='sigmoid'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(2, activation= 'softmax'),
])
```

In [ ]:
```
loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=False)

metrics = [
    'Accuracy',
    tf.keras.metrics.Recall(),
    tf.keras.metrics.Precision(),
    tf.keras.metrics.TruePositives(name="tp"),
    tf.keras.metrics.TrueNegatives(name='tn'),
    tf.keras.metrics.AUC()
]
model.compile(
    optimizer='adam',
    loss=loss_fn,
    metrics=metrics
)

model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
657/657 [==============================] - 7s 9ms/step - loss: 0.7244 - Accuracy: 0.4750
- recall: 0.4749 - precision: 0.4750 - tp: 9973.0000 - tn: 9976.0000 - auc: 0.4510
Epoch 2/10
657/657 [==============================] - 6s 9ms/step - loss: 0.5984 - Accuracy: 0.7501
```

```
- recall: 0.7501 - precision: 0.7501 - tp: 15752.0000 - tn: 15752.0000 - auc: 0.8497
Epoch 3/10
657/657 [==============================] - 6s 9ms/step - loss: 0.3790 - Accuracy: 0.8714
- recall: 0.8714 - precision: 0.8714 - tp: 18300.0000 - tn: 18300.0000 - auc: 0.9686
Epoch 4/10
657/657 [==============================] - 6s 9ms/step - loss: 0.2549 - Accuracy: 0.9505
- recall: 0.9505 - precision: 0.9505 - tp: 19960.0000 - tn: 19960.0000 - auc: 0.9843
Epoch 5/10
657/657 [==============================] - 6s 10ms/step - loss: 0.1954 - Accuracy: 0.955
4 - recall: 0.9554 - precision: 0.9554 - tp: 20063.0000 - tn: 20063.0000 - auc: 0.9849
Epoch 6/10
657/657 [==============================] - 7s 10ms/step - loss: 0.1656 - Accuracy: 0.954
1 - recall: 0.9541 - precision: 0.9541 - tp: 20036.0000 - tn: 20036.0000 - auc: 0.9845
Epoch 7/10
657/657 [==============================] - 6s 9ms/step - loss: 0.1500 - Accuracy: 0.9538
- recall: 0.9538 - precision: 0.9538 - tp: 20030.0000 - tn: 20030.0000 - auc: 0.9840
Epoch 8/10
657/657 [==============================] - 6s 9ms/step - loss: 0.1365 - Accuracy: 0.9549
- recall: 0.9549 - precision: 0.9549 - tp: 20053.0000 - tn: 20053.0000 - auc: 0.9843
Epoch 9/10
657/657 [==============================] - 7s 10ms/step - loss: 0.1276 - Accuracy: 0.955
3 - recall: 0.9553 - precision: 0.9553 - tp: 20061.0000 - tn: 20061.0000 - auc: 0.9845
Epoch 10/10
657/657 [==============================] - 7s 10ms/step - loss: 0.1224 - Accuracy: 0.954
1 - recall: 0.9541 - precision: 0.9541 - tp: 20037.0000 - tn: 20037.0000 - auc: 0.9842
<keras.callbacks.History at 0x2bed2698f10>
```

Out[ ]:

Although the model has been created, there is still one more step to extract the classification. As shown earlier on, the pineapples can be considered to be classified as 1, whereas bananas can be considered to be classified as 0.

In [ ]:

```
def classify(fshape, ftexture, fweight, model=model):
    prediction = model.predict(np.array([[fshape, ftexture, fweight]]))
    fnum = np.argmax(prediction)
    if fnum == 0:
        fclass = "banana"
    elif fnum == 1:
        fclass = "pineapple"
    return fclass

classify(1,-1,1)
```

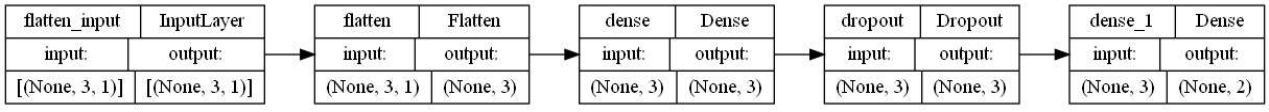
```
1/1 [==============================] - 0s 60ms/step
'pineapple'
```

Out[ ]:

In [ ]:

```
tf.keras.utils.plot_model(model, show_shapes=True, rankdir='LR')
```

Out[ ]:



As shown above, the neural network possesses 3 input nodes, 1 hidden layers each with three nodes and an output of two which will indicate whether the fruit is a pineapple or a banana. The Flatten layer converts the input into a 1D array while the Dropout Layer in tensorflow is used only during

the training step and is meant to prevent overfitting while training the model. A summary of the model can be seen below:

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 3)                 0

 dense (Dense)               (None, 3)                 12

 dropout (Dropout)           (None, 3)                 0

 dense_1 (Dense)             (None, 2)                 8

=================================================================
Total params: 20
Trainable params: 20
Non-trainable params: 0
_________________________________________________________________
```

Now that the neural network is able to appropriately classify, one can create a test that determines whether the neural network performs well.

In [ ]:
```
def test(x_test, y_test, model=model):
    evaluation = model.evaluate(x_test, y_test, batch_size=128)
    return evaluation

test(X_test,y_test)
```

Out[ ]:
```
71/71 [==============================] - 1s 8ms/step - loss: 0.0137 - Accuracy: 1.0000 -
recall: 1.0000 - precision: 1.0000 - tp: 9000.0000 - tn: 9000.0000 - auc: 1.0000
[0.01369396410882473, 1.0, 1.0, 1.0, 9000.0, 9000.0, 1.0]
```

The test above was done using a series of random records that were chosen based on the function above. As shown by the resulting values, the model is very accurate at predicting whether the fruit in question is a pineapple or a banana is a matter of whether the fruit is rough or smooth (pineapples will always have a rough texture, whereas bananas will always have a smooth texture).

Proposition

Automating this process is not very difficult, to supplement the introduction of new fruits into the model, one may want to create a form of data pipeline where the initial data or expected properties are introduced into the model which can be retrained everytime there is a new fruit introduced. Now a better idea may involve obtaining the exact weight, provide more variety selection of shape, and provide a new feature such as color. With this, one should be able to develop an unsupervised model, such as a Kmeans model, that can categorize the fruit. Once the fruit is categorized, an employee may supply the appropriate name of the fruit based on the properties.