```
import tensorflow as tf
print(tf.__version__)
```

```
2.4.0
```

Application of Neural Network to Binary Output Classification

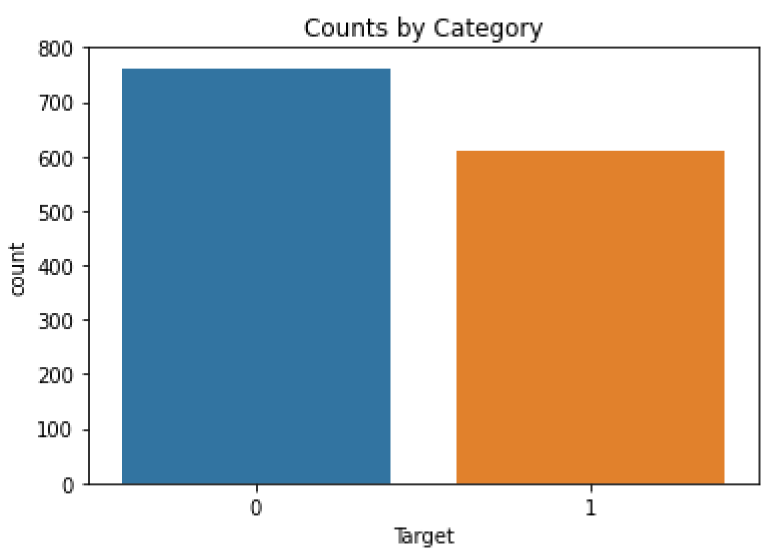
```
import seaborn as sns
import pandas as pd
import numpy as np
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
banknote_data = pd.read_csv('https://raw.githubusercontent.com/AbhiRoy96/Banknote-Authentication-UCI-Dataset/master/bank_notes.csv')
banknote_data.head()
```

	variance	skewness	curtosis	entropy	Target
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

```
banknote_data.shape
```

```
(1372, 5)
```

```
import matplotlib.pyplot as plt
%matplotlib inline
ax=plt.axes()
sns.countplot(x='Target', data=banknote_data, ax=ax)
ax.set_title('Counts by Category')
plt.show()
```




```
# Determine X and y
X = banknote_data.drop(['Target'], axis=1).values
y = banknote_data[['Target']].values
print(X.shape)
print(y.shape)
```

```
(1372, 4)
(1372, 1)
```

```
# Create train and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
# Standardize the variables
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

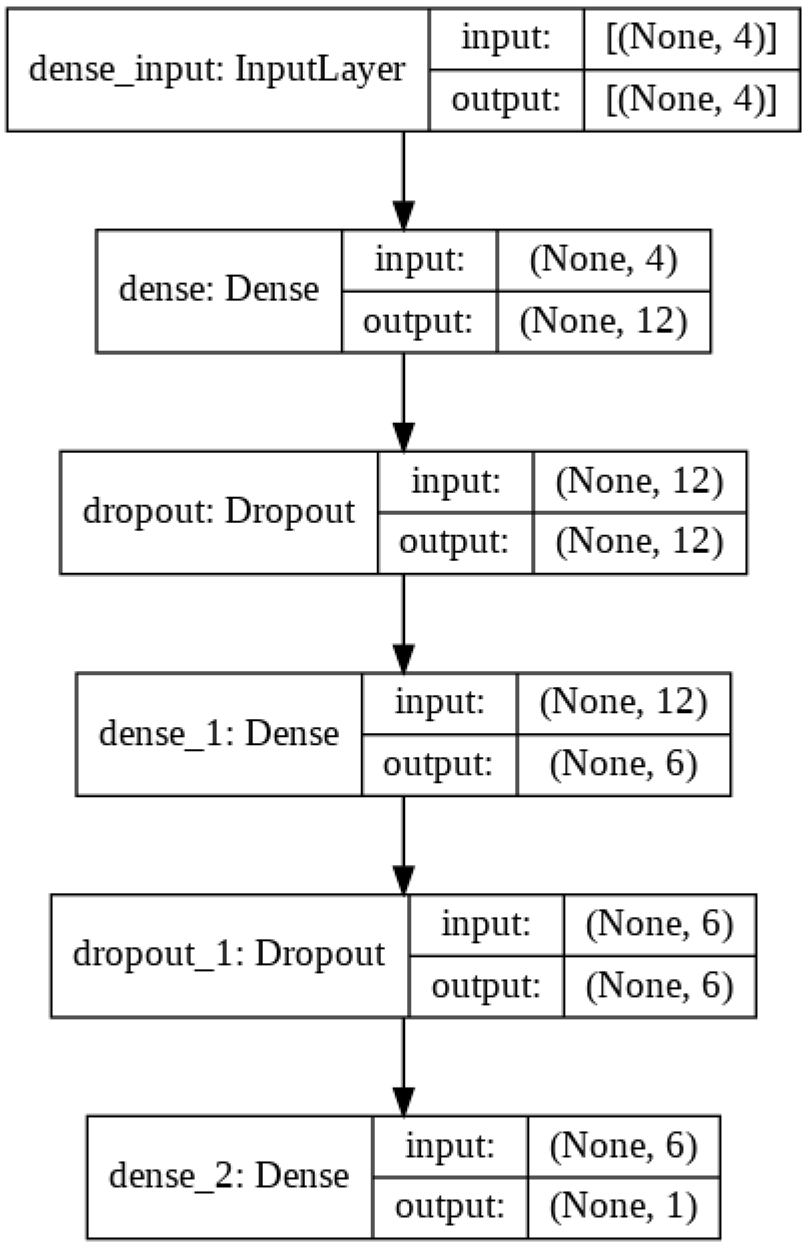
```
# Create the Neural Network
```

```
def create_model(learning_rate, dropout_rate):
  model = Sequential()
  model.add(Dense(12, input_dim=X_train.shape[1], activation='relu'))
  model.add(Dropout(dropout_rate))
  model.add(Dense(6, activation='relu'))
  model.add(Dropout(dropout_rate))
  model.add(Dense(1, activation='sigmoid'))
  adam = Adam(lr=learning_rate)
  model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
  return model
```

```
# Set the hyperparameters
dropout_rate = 0.1
epochs = 20
batch_size = 4
learn_rate = 0.001
```

```
model = create_model(learn_rate, dropout_rate)
```

```
# Visualize model structure
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_plot1.png', show_shapes=True, show_layer_names=True)
```




```
model_history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.2, verbose=1)
```

```
Epoch 1/20
220/220 [==============================] - 1s 3ms/step - loss: 0.5743 - accuracy: 0.6900 - val_loss: 0.3585 - val_accuracy: 0.9045
Epoch 2/20
220/220 [==============================] - 0s 2ms/step - loss: 0.3187 - accuracy: 0.8971 - val_loss: 0.1971 - val_accuracy: 0.9500
Epoch 3/20
220/220 [==============================] - 0s 2ms/step - loss: 0.2402 - accuracy: 0.9300 - val_loss: 0.1274 - val_accuracy: 0.9591
Epoch 4/20
220/220 [==============================] - 0s 2ms/step - loss: 0.1540 - accuracy: 0.9644 - val_loss: 0.0883 - val_accuracy: 0.9864
Epoch 5/20
220/220 [==============================] - 0s 2ms/step - loss: 0.1323 - accuracy: 0.9700 - val_loss: 0.0634 - val_accuracy: 0.9864
Epoch 6/20
220/220 [==============================] - 0s 2ms/step - loss: 0.1258 - accuracy: 0.9678 - val_loss: 0.0513 - val_accuracy: 0.9955
Epoch 7/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0949 - accuracy: 0.9716 - val_loss: 0.0394 - val_accuracy: 0.9955
Epoch 8/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0758 - accuracy: 0.9811 - val_loss: 0.0316 - val_accuracy: 0.9955
Epoch 9/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0934 - accuracy: 0.9739 - val_loss: 0.0261 - val_accuracy: 0.9955
Epoch 10/20
```

```
220/220 [==============================] - 0s 2ms/step - loss: 0.0595 - accuracy: 0.9884 - val_loss: 0.0207 - val_accuracy: 1.0000
Epoch 11/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0592 - accuracy: 0.9891 - val_loss: 0.0180 - val_accuracy: 0.9955
Epoch 12/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0700 - accuracy: 0.9818 - val_loss: 0.0162 - val_accuracy: 0.9955
Epoch 13/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0616 - accuracy: 0.9830 - val_loss: 0.0124 - val_accuracy: 1.0000
Epoch 14/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0558 - accuracy: 0.9892 - val_loss: 0.0107 - val_accuracy: 1.0000
Epoch 15/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0505 - accuracy: 0.9872 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 16/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0478 - accuracy: 0.9927 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 17/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0409 - accuracy: 0.9927 - val_loss: 0.0093 - val_accuracy: 1.0000
Epoch 18/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0438 - accuracy: 0.9959 - val_loss: 0.0077 - val_accuracy: 1.0000
Epoch 19/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0425 - accuracy: 0.9883 - val_loss: 0.0065 - val_accuracy: 1.0000
Epoch 20/20
220/220 [==============================] - 0s 2ms/step - loss: 0.0352 - accuracy: 0.9913 - val_loss: 0.0060 - val_accuracy: 1.0000
```

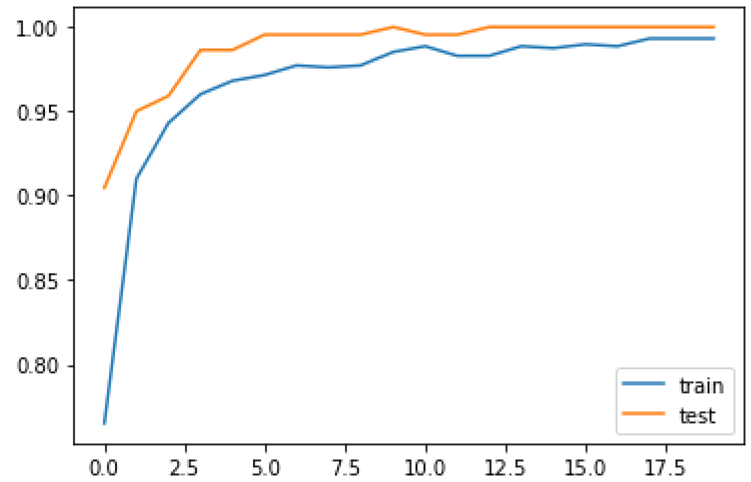
```
# Model Performance
accuracies = model.evaluate(X_test, y_test, verbose=1)
print('Test Score;', accuracies[0])
print('Test Accuracy:', accuracies[1])
```

```
9/9 [==============================] - 0s 3ms/step - loss: 0.0110 - accuracy: 1.0000
Test Score; 0.011000803671777248
Test Accuracy: 1.0
```

Visualize the model performance for the training and test datasets.

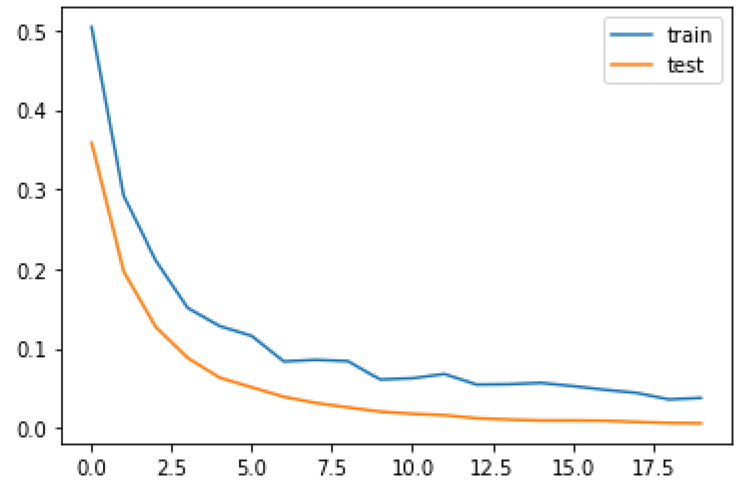
```
plt.plot(model_history.history['accuracy'], label = 'accuracy')
plt.plot(model_history.history['val_accuracy'], label = 'val_accuracy')
plt.legend(['train', 'test'])
```

```
<matplotlib.legend.Legend at 0x7fbbc3b9a278>
```




```
plt.plot(model_history.history['loss'], label = 'loss')
plt.plot(model_history.history['val_loss'], label = 'val_loss')
plt.legend(['train','test'])
```

```
<matplotlib.legend.Legend at 0x7fbbc3b049e8>
```



Application of Neural Network to Multiclass Output

There are three changes compared to the simple class output:

- Change the number of nodes in the final dense layer to the number of target labels in the output.
- Change the activation function in the final dense layer from sigmoid to softmax.

- Change the loss function in the compile method of the model from binary_crossentropy to categorical_crossentropy.

```
import seaborn as sns
import pandas as pd
import numpy as np
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
iris_data = sns.load_dataset('iris')
iris_data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
X = iris_data.drop(['species'], axis=1)
y = pd.get_dummies(iris_data.species, prefix='output')
X.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y.head()
```

	output_setosa	output_versicolor	output_virginica
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0

```
X = X.values
y = y.values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

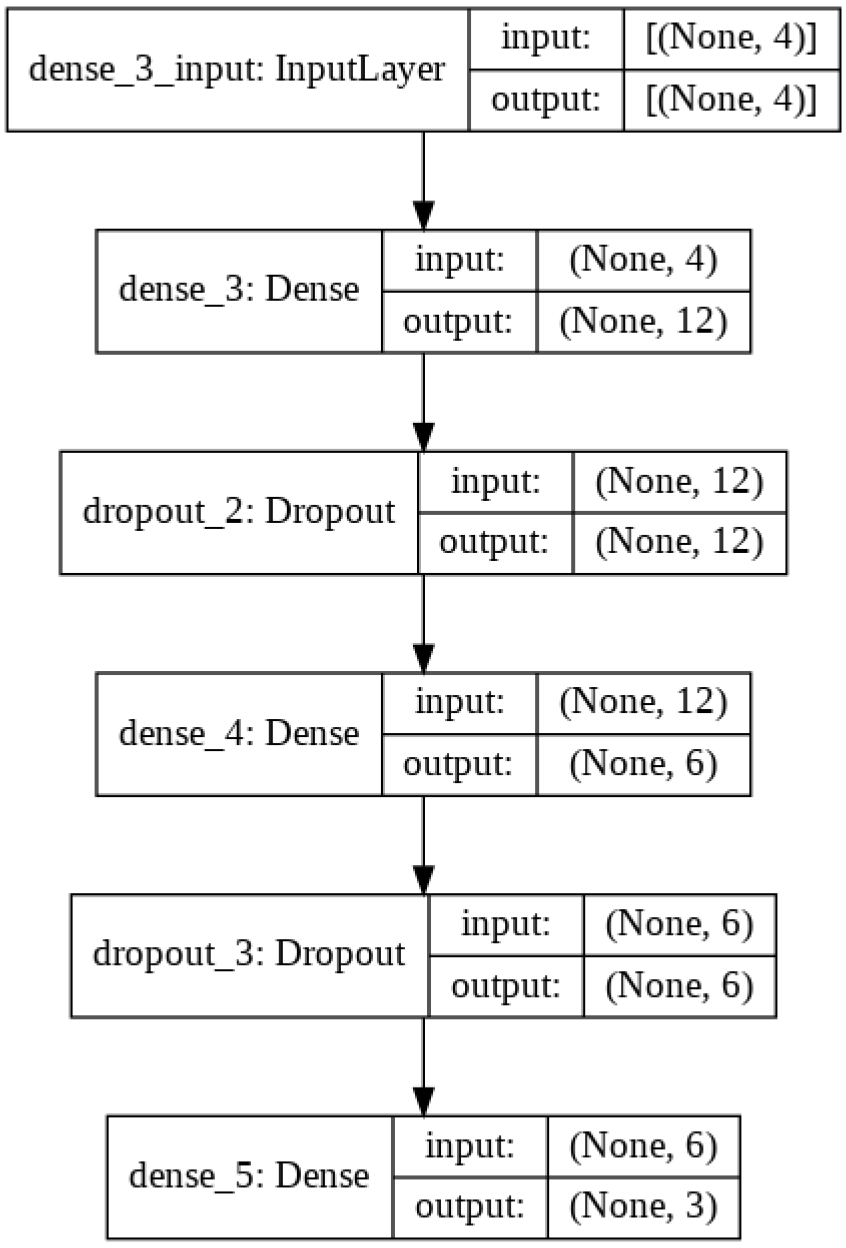
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
def create_model_multiple_outs(learning_rate, dropout_rate):
  model = Sequential()
  model.add(Dense(12, input_dim=X_train.shape[1], activation='relu'))
  model.add(Dropout(dropout_rate))
  model.add(Dense(6, activation='relu'))
  model.add(Dropout(dropout_rate))
  model.add(Dense(y_train.shape[1], activation='softmax'))
  adam = Adam(lr=learning_rate)
  model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
  return model
```

```
dropout_rate = 0.1
epochs = 50
```

```
epochs = 50
batch_size = 1
learn_rate = 0.001
```

```
model = create_model_multiple_outs(learn_rate, dropout_rate)
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_plot1.png', show_shapes=True, show_layer_names=True)
```




```
model_history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.2, verbose=1)
```

```
Epoch 1/50
96/96 [==============================] - 1s 3ms/step - loss: 1.0346 - accuracy: 0.2466 - val_loss: 0.8499 - val_accuracy: 0.5833
Epoch 2/50
96/96 [==============================] - 0s 2ms/step - loss: 1.0029 - accuracy: 0.5238 - val_loss: 0.7893 - val_accuracy: 0.7500
Epoch 3/50
96/96 [==============================] - 0s 2ms/step - loss: 0.9889 - accuracy: 0.6691 - val_loss: 0.7617 - val_accuracy: 0.8333
Epoch 4/50
96/96 [==============================] - 0s 2ms/step - loss: 0.9122 - accuracy: 0.6031 - val_loss: 0.7213 - val_accuracy: 0.8333
Epoch 5/50
96/96 [==============================] - 0s 2ms/step - loss: 0.8673 - accuracy: 0.7538 - val_loss: 0.6375 - val_accuracy: 0.8333
Epoch 6/50
96/96 [==============================] - 0s 2ms/step - loss: 0.7370 - accuracy: 0.7204 - val_loss: 0.5393 - val_accuracy: 0.8333
Epoch 7/50
96/96 [==============================] - 0s 2ms/step - loss: 0.5669 - accuracy: 0.7359 - val_loss: 0.4900 - val_accuracy: 0.8750
Epoch 8/50
96/96 [==============================] - 0s 3ms/step - loss: 0.5290 - accuracy: 0.7288 - val_loss: 0.4462 - val_accuracy: 0.8750
Epoch 9/50
96/96 [==============================] - 0s 1ms/step - loss: 0.5457 - accuracy: 0.6873 - val_loss: 0.4136 - val_accuracy: 0.9167
Epoch 10/50
96/96 [==============================] - 0s 2ms/step - loss: 0.4694 - accuracy: 0.7829 - val_loss: 0.3856 - val_accuracy: 0.9583
Epoch 11/50
96/96 [==============================] - 0s 1ms/step - loss: 0.4567 - accuracy: 0.8017 - val_loss: 0.3483 - val_accuracy: 0.9167
Epoch 12/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3828 - accuracy: 0.8084 - val_loss: 0.3039 - val_accuracy: 0.9167
Epoch 13/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3802 - accuracy: 0.8621 - val_loss: 0.2775 - val_accuracy: 0.9167
Epoch 14/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3695 - accuracy: 0.8745 - val_loss: 0.2568 - val_accuracy: 0.9583
Epoch 15/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3339 - accuracy: 0.8546 - val_loss: 0.2300 - val_accuracy: 0.9583
Epoch 16/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3005 - accuracy: 0.8453 - val_loss: 0.2230 - val_accuracy: 0.9583
Epoch 17/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3664 - accuracy: 0.8138 - val_loss: 0.2085 - val_accuracy: 0.9583
Epoch 18/50
96/96 [==============================] - 0s 2ms/step - loss: 0.2964 - accuracy: 0.8898 - val_loss: 0.1988 - val_accuracy: 0.9583
Epoch 19/50
96/96 [==============================] - 0s 2ms/step - loss: 0.2170 - accuracy: 0.9475 - val_loss: 0.1817 - val_accuracy: 0.9583
Epoch 20/50
```

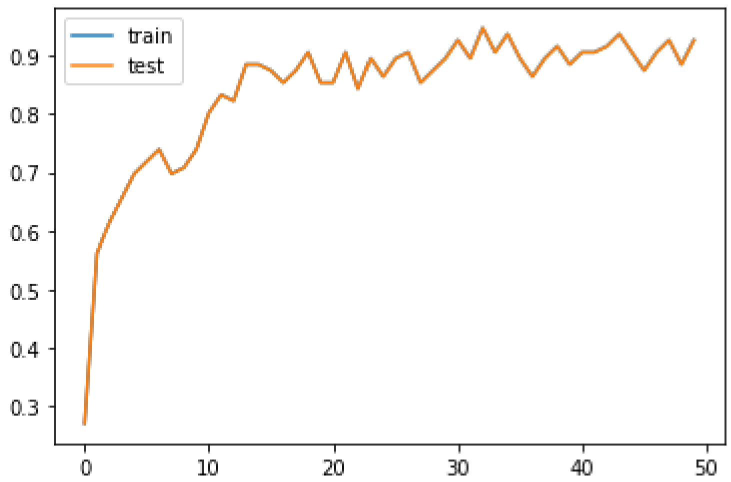
```
96/96 [==============================] - 0s 2ms/step - loss: 0.2811 - accuracy: 0.8506 - val_loss: 0.1735 - val_accuracy: 0.9583
Epoch 21/50
96/96 [==============================] - 0s 1ms/step - loss: 0.2368 - accuracy: 0.8902 - val_loss: 0.1606 - val_accuracy: 0.9583
Epoch 22/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3204 - accuracy: 0.8482 - val_loss: 0.1588 - val_accuracy: 0.9583
Epoch 23/50
96/96 [==============================] - 0s 2ms/step - loss: 0.3222 - accuracy: 0.8216 - val_loss: 0.1589 - val_accuracy: 0.9583
Epoch 24/50
96/96 [==============================] - 0s 2ms/step - loss: 0.2102 - accuracy: 0.8818 - val_loss: 0.1647 - val_accuracy: 0.9583
Epoch 25/50
96/96 [==============================] - 0s 1ms/step - loss: 0.2628 - accuracy: 0.8308 - val_loss: 0.1580 - val_accuracy: 0.9583
Epoch 26/50
96/96 [==============================] - 0s 2ms/step - loss: 0.1925 - accuracy: 0.9269 - val_loss: 0.1588 - val_accuracy: 0.9583
Epoch 27/50
96/96 [==============================] - 0s 1ms/step - loss: 0.1997 - accuracy: 0.9007 - val_loss: 0.1479 - val_accuracy: 0.9583
Epoch 28/50
96/96 [==============================] - 0s 1ms/step - loss: 0.3094 - accuracy: 0.8254 - val_loss: 0.1519 - val_accuracy: 0.9583
Epoch 29/50
96/96 [==============================] - 0s 2ms/step - loss: 0.2318 - accuracy: 0.8953 - val_loss: 0.1337 - val_accuracy: 0.9583
Epoch 30/50
```

```
accuracies = model.evaluate(X_test, y_test, verbose=1)
print('Test Score:', accuracies[0])
print('Test Accuracy:', accuracies[1])
```

```
1/1 [==============================] - 0s 18ms/step - loss: 0.0651 - accuracy: 1.0000
Test Score: 0.06505632400512695
Test Accuracy: 1.0
```

```
import matplotlib.pyplot as plt
plt.plot(model_history.history['accuracy'], label = 'accuracy')
plt.plot(model_history.history['accuracy'], label = 'val_accuracy')
plt.legend(['train', 'test'])
```

<matplotlib.legend.Legend at 0x7fbbc1fc0fd0>




```
import matplotlib.pyplot as plt
plt.plot(model_history.history['loss'], label = 'loss')
plt.plot(model_history.history['val_loss'], label = 'val_loss')
plt.legend(['train','test'])
```

<matplotlib.legend.Legend at 0x7fbbc1f7f4e0>