Dataset *Information

```
#The training archive contains 25,000 images of dogs and cats .Train your alg
#(1=dog, 0 = cat)
```

Download Dataset

```
!wget https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6
```

```
--2022-07-06 09:45:51--  https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-486
Resolving download.microsoft.com (download.microsoft.com)... 23.72.44.106, 2600:1417:3f
Connecting to download.microsoft.com (download.microsoft.com)|23.72.44.106|:443... conne
HTTP request sent, awaiting response... 200 OK
Length: 824887076 (787M) [application/octet-stream]
Saving to: ‘kagglecatsanddogs_5340.zip’

kagglecatsanddogs_5 100%[===================>] 786.67M   204MB/s    in 3.9s

2022-07-06 09:45:55 (199 MB/s) - ‘kagglecatsanddogs_5340.zip’ saved [824887076/82488707
```

Unzip the Dataset

```
!unzip  kagglecatsanddogs_5340.zip
```

```
inflating: PetImages/Dog/9913.jpg
inflating: PetImages/Dog/9914.jpg
inflating: PetImages/Dog/9915.jpg
inflating: PetImages/Dog/9916.jpg
inflating: PetImages/Dog/9917.jpg
inflating: PetImages/Dog/9918.jpg
inflating: PetImages/Dog/9919.jpg
inflating: PetImages/Dog/992.jpg
inflating: PetImages/Dog/9920.jpg
inflating: PetImages/Dog/9921.jpg
inflating: PetImages/Dog/9922.jpg
inflating: PetImages/Dog/9923.jpg
inflating: PetImages/Dog/9924.jpg
inflating: PetImages/Dog/9925.jpg
inflating: PetImages/Dog/9926.jpg
inflating: PetImages/Dog/9927.jpg
```

```
inflating: PetImages/Dog/9928.jpg
inflating: PetImages/Dog/9929.jpg
inflating: PetImages/Dog/993.jpg
inflating: PetImages/Dog/9930.jpg
inflating: PetImages/Dog/9931.jpg
inflating: PetImages/Dog/9932.jpg
inflating: PetImages/Dog/9933.jpg
inflating: PetImages/Dog/9934.jpg
inflating: PetImages/Dog/9935.jpg
inflating: PetImages/Dog/9936.jpg
inflating: PetImages/Dog/9937.jpg
inflating: PetImages/Dog/9938.jpg
inflating: PetImages/Dog/9939.jpg
inflating: PetImages/Dog/994.jpg
inflating: PetImages/Dog/9940.jpg
inflating: PetImages/Dog/9941.jpg
inflating: PetImages/Dog/9942.jpg
inflating: PetImages/Dog/9943.jpg
inflating: PetImages/Dog/9944.jpg
inflating: PetImages/Dog/9945.jpg
inflating: PetImages/Dog/9946.jpg
inflating: PetImages/Dog/9947.jpg
inflating: PetImages/Dog/9948.jpg
inflating: PetImages/Dog/9949.jpg
inflating: PetImages/Dog/995.jpg
inflating: PetImages/Dog/9950.jpg
inflating: PetImages/Dog/9951.jpg
inflating: PetImages/Dog/9952.jpg
inflating: PetImages/Dog/9953.jpg
inflating: PetImages/Dog/9954.jpg
inflating: PetImages/Dog/9955.jpg
inflating: PetImages/Dog/9956.jpg
inflating: PetImages/Dog/9957.jpg
inflating: PetImages/Dog/9958.jpg
inflating: PetImages/Dog/9959.jpg
inflating: PetImages/Dog/996.jpg
inflating: PetImages/Dog/9960.jpg
inflating: PetImages/Dog/9961.jpg
inflating: PetImages/Dog/9962.jpg
inflating: PetImages/Dog/9963.jpg
inflating: PetImages/Dog/9964.jpg
inflating: PetImages/Dog/9965.jpg
inflating: PetImages/Dog/9966.jpg
```

Import Modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
```

```
import os
import tqdm
import random
from keras.preprocessing.image import load_img
warnings.filterwarnings('ignore')
```

Create Dataframe for Input and Output

```
input_path = []
label = []

for class_name in os.listdir("PetImages"):
    for path in os.listdir("PetImages/"+class_name):
        if class_name == 'Cat':
            label.append(0)
        else:
            label.append(1)
        input_path.append(os.path.join("PetImages",class_name,path))
print(input_path[0],label[0])
```

```
PetImages/Cat/8738.jpg 0
```

```
print(input_path[10],label[10])
```

```
PetImages/Cat/4196.jpg 0
```

```
len(input_path)
```

```
25002
```

```
df = pd.DataFrame()
df['images'] = input_path
df['label'] = label
df = df.sample(frac=1).reset_index(drop=True) #we will sufill the data(data w
df.head()
```

	images	label
0	PetImages/Cat/1900.jpg	0
1	PetImages/Cat/3625.jpg	0

```
#delete db files
for i in df['images']:
  if '.jpg' not in i:
    print(i)
```

```
PetImages/Cat/Thumbs.db
PetImages/Dog/Thumbs.db
```

```
import PIL
l = []
for image in df['images']:
    try:
        img = PIL.Image.open(image)
    except:
      l.append(image)
l
```

```
['PetImages/Cat/666.jpg',
 'PetImages/Dog/11702.jpg',
 'PetImages/Cat/Thumbs.db',
 'PetImages/Dog/Thumbs.db']
```

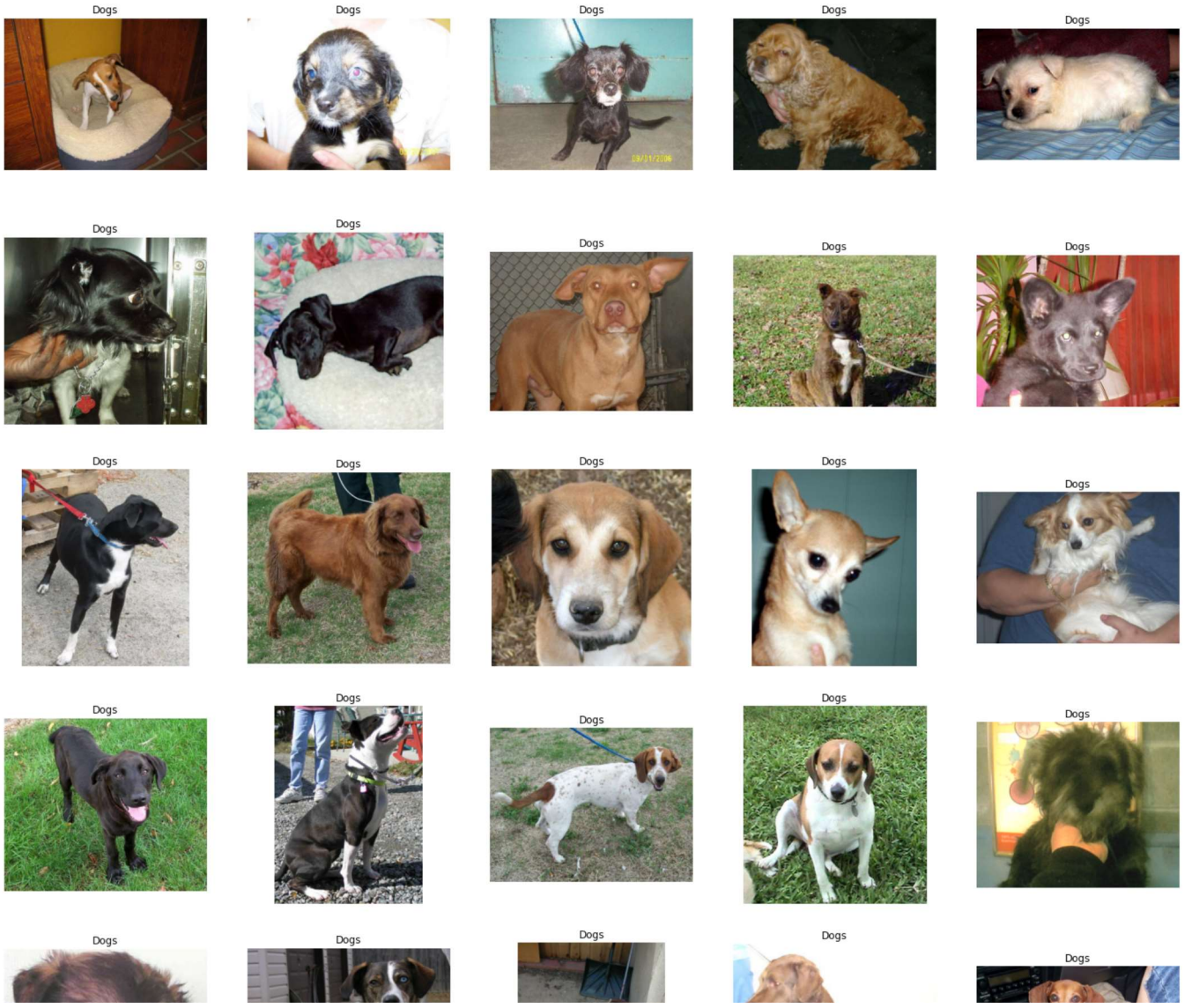
```
df = df[df['images']!='PetImages/Dog/Thumbs.db']
df = df[df['images']!='PetImages/Cat/Thumbs.db']
df = df[df['images']!='PetImages/Cat/666.jpg']
df = df[df['images']!='PetImages/Dog/11702.jpg']
len(df)
```

```
24998
```

Exploratory Data Analysis

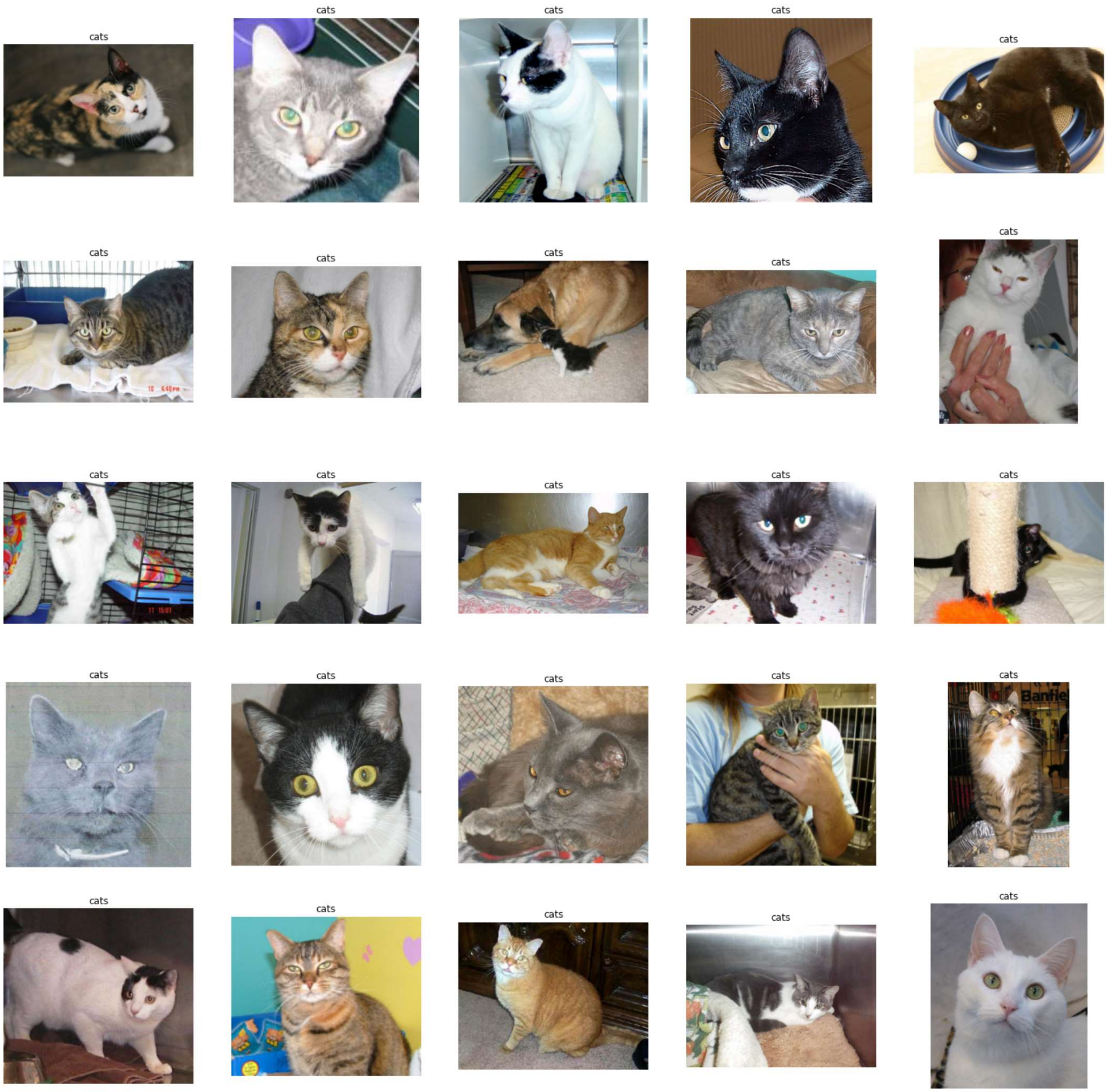
```
#to display grid of images
plt.figure(figsize=(25,25))
temp = df[df['label']==1]['images']
start = random.randint(0, len(temp))
files = temp[start:start+25]
```

```
for index,file in enumerate(files):
     plt.subplot(5,5,index+1)
     img = load_img(file)
     imp = np.array(img)
     plt.imshow(img)
     plt.title('Dogs')
     plt.axis('off')
```



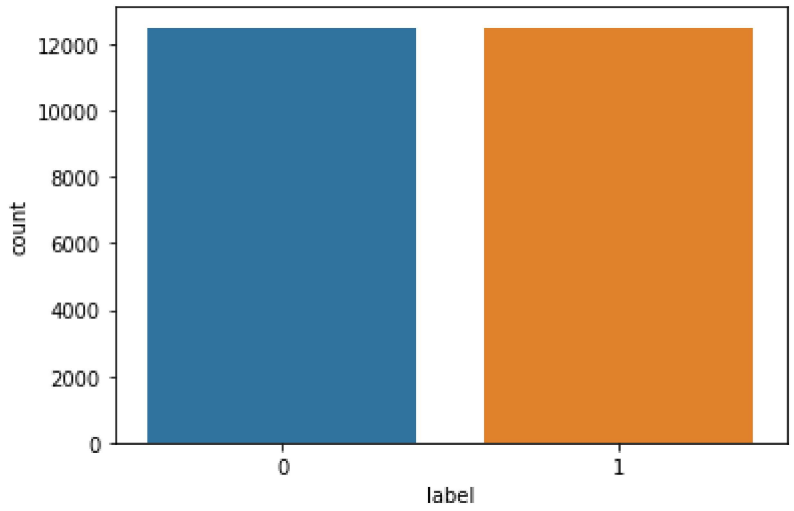

```
#to display grid of images
plt.figure(figsize=(25,25))
temp = df[df['label']==0]['images']
start = random.randint(0, len(temp))
files = temp[start:start+25]

for index,file in enumerate(files):
     plt.subplot(5,5,index+1)
     img = load_img(file)
     imp = np.array(img)
     plt.imshow(img)
     plt.title('cats')
     plt.axis('off')
```


cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats

```
import seaborn as sns
sns.countplot(df['label'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff20bfb7990>
```



Create DataGenerators for the Images

```
df['label'] = df['label'].astype('str')
```

```
df.head()
```

	images	label
0	PetImages/Cat/1900.jpg	0
1	PetImages/Cat/3625.jpg	0
2	PetImages/Dog/6014.jpg	1
3	PetImages/Dog/189.jpg	1
4	PetImages/Dog/4363.jpg	1

```
#input split
from sklearn.model_selection import train_test_split
train, test = train_test_split(df,test_size=0.2, random_state=42)


train.head()
```

	images	label
23290	PetImages/Cat/11879.jpg	0
17785	PetImages/Dog/10168.jpg	1
1021	PetImages/Dog/4258.jpg	1
12648	PetImages/Dog/10464.jpg	1
1534	PetImages/Dog/11112.jpg	1

```
test.head() #we have separate datas
```

	images	label
6870	PetImages/Cat/51.jpg	0
22914	PetImages/Dog/8680.jpg	1
9671	PetImages/Dog/8228.jpg	1
13642	PetImages/Dog/1935.jpg	1
23341	PetImages/Cat/6544.jpg	0

```
from keras.preprocessing.image import ImageDataGenerator
train_generator = ImageDataGenerator(
    rescale = 1./255, #normalization of images
    rotation_range = 40, #augmention of images to avoid overfitting
    shear_range =0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
```

```
    fill_mode = 'nearest'
)

val_generator =ImageDataGenerator(rescale = 1./255)

train_iterator = train_generator.flow_from_dataframe(
    train,
    x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)

val_iterator = val_generator.flow_from_dataframe(
    test,
    x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)
```

```
Found 19998 validated image filenames belonging to 2 classes.
Found 5000 validated image filenames belonging to 2 classes.
```

```
from keras import Sequential
from keras.layers import Conv2D,MaxPool2D,Flatten,Dense
```

Model Creation

```
from keras import Sequential
from keras.layers import Conv2D,MaxPool2D,Flatten,Dense

model = Sequential([
                    Conv2D(16, (3,3),activation='relu',input_shape=(128,128,3
                    MaxPool2D((2,2)),
                    Conv2D(32, (3,3), activation ='relu'),
                    MaxPool2D((2,2)),
                    Conv2D(64, (3,3), activation='relu'),
                    MaxPool2D((2,2)),
                    Flatten(),
                    Dense(512, activation='relu'),
                    Dense(1, activation='sigmoid')
```

```
])
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 126, 126, 16)      448

 max_pooling2d_6 (MaxPooling  (None, 63, 63, 16)       0
 2D)

 conv2d_7 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_7 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 conv2d_8 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_2 (Flatten)         (None, 12544)             0

 dense_4 (Dense)             (None, 512)               6423040

 dense_5 (Dense)             (None, 1)                 513

=================================================================
Total params: 6,447,137
Trainable params: 6,447,137
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(train_iterator,epochs=10, validation_data=val_iterator)
```

```
Epoch 1/10
40/40 [==============================] - 136s 3s/step - loss: 0.7203 - accuracy: 0.5330
Epoch 2/10
40/40 [==============================] - 131s 3s/step - loss: 0.6183 - accuracy: 0.6632
Epoch 3/10
40/40 [==============================] - 133s 3s/step - loss: 0.5816 - accuracy: 0.6932
Epoch 4/10
40/40 [==============================] - 131s 3s/step - loss: 0.5430 - accuracy: 0.7282
Epoch 5/10
40/40 [==============================] - 132s 3s/step - loss: 0.5311 - accuracy: 0.7346
Epoch 6/10
```

```
40/40 [==============================] - 131s 3s/step - loss: 0.5108 - accuracy: 0.7476
Epoch 7/10
40/40 [==============================] - 130s 3s/step - loss: 0.4919 - accuracy: 0.7627
Epoch 8/10
40/40 [==============================] - 130s 3s/step - loss: 0.4907 - accuracy: 0.7598
Epoch 9/10
40/40 [==============================] - 130s 3s/step - loss: 0.4632 - accuracy: 0.7797
Epoch 10/10
40/40 [==============================] - 130s 3s/step - loss: 0.4569 - accuracy: 0.7857
```

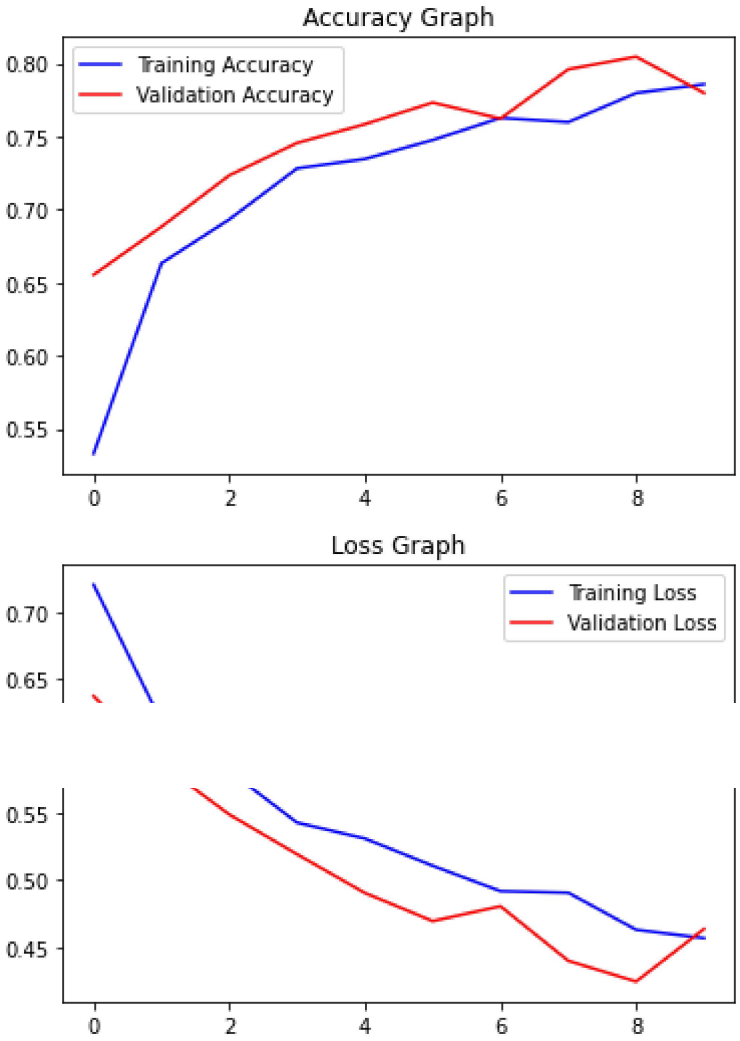
Visualization of Results

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(acc))

plt.plot(epochs,acc,'b',label='Training Accuracy')
plt.plot(epochs,val_acc,'r',label='Validation Accuracy')
plt.title('Accuracy Graph')
plt.legend()
plt.figure()

loss = history.history['loss']
val_loss = history.history['val_loss']
plt.plot(epochs,loss,'b',label='Training Loss')
plt.plot(epochs,val_loss,'r',label='Validation Loss')
plt.title('Loss Graph')
plt.legend()
plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

✓ 0s completed at 5:35 PM