```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping
from PIL import Image
import matplotlib.pyplot as plt


from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
traindata = r"/content/drive/MyDrive/dataset/train"

testdata = r"/content/drive/MyDrive/dataset/test"
train_datagen = ImageDataGenerator(
    rotation_range=40,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    traindata,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    testdata,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')
```

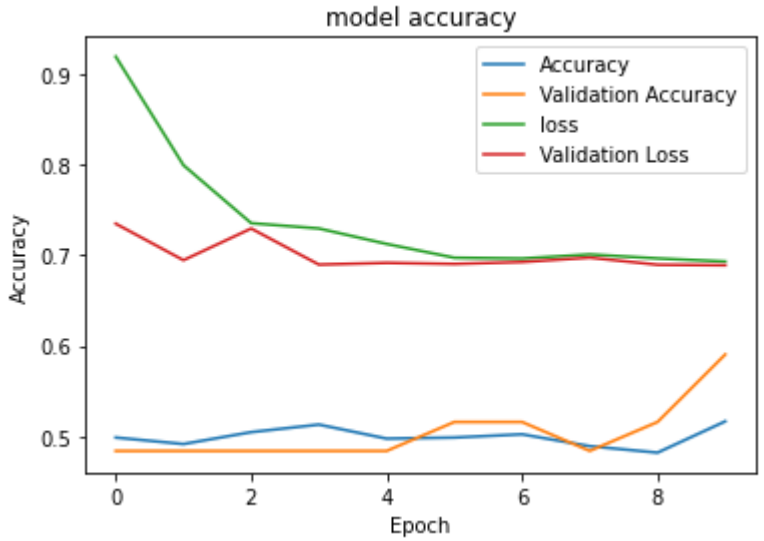
```
Found 840 images belonging to 2 classes.
Found 188 images belonging to 2 classes.
```

✓ 0s completed at 12:26 AM

```
model = Sequential()
model.add(ResNet50(include_top=False, pooling='avg', weights='imagenet'))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))



for layer in model.layers[0].layers:
    layer.trainable = False

model.compile(optimizer=Adam(lr = 0.001), loss='categorical_crossentropy', metri

history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              steps_per_epoch=len(train_generator),
                              validation_steps=len(validation_generator))
```

```
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/adam.p
  super(Adam, self).__init__(name, **kwargs)
<ipython-input-9-952590400691>:15: UserWarning: `Model.fit_generator` is de
  history = model.fit_generator(train_generator,
Epoch 1/10
27/27 [==============================] - 682s 25s/step - loss: 0.9190 - acc
Epoch 2/10
27/27 [==============================] - 14s 527ms/step - loss: 0.7992 - ac
Epoch 3/10
27/27 [==============================] - 15s 570ms/step - loss: 0.7353 - ac
Epoch 4/10
27/27 [==============================] - 14s 522ms/step - loss: 0.7294 - ac
Epoch 5/10
27/27 [==============================] - 14s 533ms/step - loss: 0.7124 - ac
Epoch 6/10
27/27 [==============================] - 14s 525ms/step - loss: 0.6971 - ac
Epoch 7/10
27/27 [==============================] - 14s 531ms/step - loss: 0.6963 - ac
Epoch 8/10
27/27 [==============================] - 14s 523ms/step - loss: 0.7008 - ac
Epoch 9/10
27/27 [==============================] - 14s 529ms/step - loss: 0.6964 - ac
Epoch 10/10
27/27 [==============================] - 14s 529ms/step - loss: 0.6930 - ac
```

```
plt.plot(history.history["accuracy"])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation Loss"])
plt.show()
```




```
model.save('/content/drive/MyDrive/model/elephant_model_rsnet_id.h5')
model.save_weights('/content/drive/MyDrive/model/elephant_weights_rsnet_id.h5')
width, height = 224, 224
model_p = '/content/drive/MyDrive/model/elephant_model_rsnet_id.h5'
weight_p = '/content/drive/MyDrive/model/elephant_weights_rsnet_id.h5'

model = tf.keras.models.load_model(model_p)
model.load_weights(weight_p)

def fpredict(file):
    if not os.path.isfile(file):
        return "Error: file does not exist."
    try:
        test_image = tf.keras.utils.load_img(file, target_size=(width,height))
        test_image = tf.keras.utils.img_to_array(test_image)
        test_image = np.expand_dims(test_image, axis=0)
    except Exception as e:
        return f"Error: {e}"
    if not 'model' in globals():
        return "Error: model is not defined."
    try:
        array = model.predict(test_image)
        result = array[0]
    except Exception as e:
        return f"Error: {e}"
    if result[0] >= 0.5 :
        an = 'African Elephant'
    elif result[1]>= 0.5:
        an = 'Asian Elephant'
    else:
        an = "Not sure"
    return an


fpredict('/content/drive/MyDrive/elephsnt.jpg')
```

```
1/1 [==============================] - 0s 21ms/step
'Asian Elephant'
```

Asian Elephant