```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob

from sklearn.model_selection import train_test_split
from sklearn import metrics

from zipfile import ZipFile
import cv2
import gc
import os

import tensorflow as tf
from tensorflow import keras
from keras import layers

import warnings
warnings.filterwarnings('ignore')


data_path = 'LungCancer.zip'

with ZipFile(data_path,'r') as zip:
  zip.extractall()
  print('The data set has been extracted.')

base_path = '/content/LungCancer.zip'
classes = []
image_paths = []
labels = []

for main_dir in os.listdir(base_path):
    if os.path.isdir(os.path.join(base_path, main_dir)):
        for sub_dir in os.listdir(os.path.join(base_path, main_dir)):
            if os.path.isdir(os.path.join(base_path, main_dir, sub_dir)):
                classes.append(f'{main_dir}/{sub_dir}')
                image_dir = os.path.join(base_path, main_dir, sub_dir)
                images = glob(f'{image_dir}/*.jpeg')
                image_paths.extend(images)
                labels.extend([f'{main_dir}/{sub_dir}'] * len(images))


# Display sample images (optional, remove if dataset is too large)
# for cat in classes:
#     image_dir = f'{base_path}/{cat}'
#     images = os.listdir(image_dir)

#     fig, ax = plt.subplots(1, 3, figsize=(15, 5))
#     fig.suptitle(f'Images for {cat} category . . . .', fontsize=20)

#     for i in range(3):
#         k = np.random.randint(0, len(images))
#         img = np.array(Image.open(f'{image_dir}/{images[k]}'))
#         ax[i].imshow(img)
#         ax[i].axis('off')
#     plt.show()


IMG_SIZE = 256
SPLIT = 0.2
EPOCHS = 10
BATCH_SIZE = 64

X = []
Y = []

label_map = {label: i for i, label in enumerate(classes)}

for image_path, label in zip(image_paths, labels):
    img = cv2.imread(image_path)
    if img is not None:  # Check if image is loaded successfully
        X.append(cv2.resize(img, (IMG_SIZE, IMG_SIZE)))
        Y.append(label_map[label])

X = np.asarray(X)
one_hot_encoded_Y = pd.get_dummies(Y).values


X_train, X_val, Y_train, Y_val = train_test_split(X, one_hot_encoded_Y, test_size=SPLIT, random_state=2022)
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X, one_hot_encoded_Y, test_size=SPLIT, random_state=2022)


model = keras.models.Sequential([
    layers.Conv2D(filters=32,
                  kernel_size=(5, 5),
                  activation='relu',
                  input_shape=(IMG_SIZE,
                               IMG_SIZE,
                               3),
                  padding='same'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(filters=64,
                  kernel_size=(3, 3),
                  activation='relu',
                  padding='same'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(filters=128,
                  kernel_size=(3, 3),
                  activation='relu',
                  padding='same'),
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(len(classes), activation='softmax')
])
model.summary()


from keras.callbacks import EarlyStopping, ReduceLROnPlateau


class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_accuracy') > 0.90:
            print('\n Validation accuracy has reached upto \
                      90% so, stopping further training.')
            self.model.stop_training = True


es = EarlyStopping(patience=3,
                   monitor='val_accuracy',
                   restore_best_weights=True)

lr = ReduceLROnPlateau(monitor='val_loss',
                       patience=2,
                       factor=0.5,
                       verbose=1)

history = model.fit(X_train, Y_train,
                    validation_data = (X_val, Y_val),
                    batch_size = BATCH_SIZE,
                    epochs = EPOCHS,
                    verbose = 1,
                    callbacks = [es, lr, myCallback()])


history_df = pd.DataFrame(history.history)
history_df.loc[:,['accuracy','val_accuracy']].plot()
plt.show()


Y_pred = model.predict(X_val)
Y_val = np.argmax(Y_val, axis=1)
Y_pred = np.argmax(Y_pred, axis=1)
print(metrics.classification_report(Y_val, Y_pred,
                                    target_names=classes))


print(metrics.classification_report(Y_val, Y_pred,
                                    target_names=classes))
```

```
The data set has been extracted.
---------------------------------------------------------------------------
NotADirectoryError                        Traceback (most recent call last)
/tmp/ipython-input-4209922017.py in <cell line: 0>()
     32 labels = []
     33
---> 34 for main_dir in os.listdir(base_path):
     35     if os.path.isdir(os.path.join(base_path, main_dir)):
     36         for sub_dir in os.listdir(os.path.join(base_path, main_dir)):

NotADirectoryError: [Errno 20] Not a directory: '/content/LungCancer.zip'
```

```
import csv

# Assuming your CSV file is named 'my_data.csv'
# and is located in the same directory as your notebook.
# If not, replace 'my_data.csv' with the correct path.

file_path = '/content/survey lung cancer.csv'

try:
    with open(file_path, 'r', newline='') as csvfile:
        # Use csv.reader for simple comma-separated values
        csv_reader = csv.reader(csvfile)

        # If your CSV has a header row, you can skip it
        header = next(csv_reader)
        print(f"Header: {header}")

        # Iterate over each row in the CSV file
        print("Data rows:")
        for row in csv_reader:
            print(row)

except FileNotFoundError:
    print(f"Error: The file '{file_path}' was not found.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Header: ['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE', 'CHRONIC DISEASE', 'FATIGUE ', 'ALLERGY
Data rows:
['M', '69', '1', '2', '2', '1', '1', '2', '1', '2', '2', '2', '2', '2', '2', 'YES']
['M', '74', '2', '1', '1', '1', '2', '2', '2', '1', '1', '1', '2', '2', '2', 'YES']
['F', '59', '1', '1', '1', '2', '1', '2', '1', '2', '1', '2', '2', '1', '2', 'NO']
['M', '63', '2', '2', '2', '1', '1', '1', '1', '1', '2', '1', '1', '2', '2', 'NO']
['F', '63', '1', '2', '1', '1', '1', '1', '1', '2', '1', '2', '2', '1', '1', 'NO']
['F', '75', '1', '2', '1', '1', '2', '2', '2', '2', '1', '2', '2', '1', '1', 'YES']
['M', '52', '2', '1', '1', '1', '1', '2', '1', '2', '2', '2', '2', '1', '2', 'YES']
['F', '51', '2', '2', '2', '2', '1', '2', '2', '1', '1', '1', '2', '2', '1', 'YES']
['F', '68', '2', '1', '2', '1', '1', '2', '1', '1', '1', '1', '1', '1', '1', 'NO']
['M', '53', '2', '2', '2', '2', '2', '1', '2', '1', '2', '1', '1', '2', '2', 'YES']
['F', '61', '2', '2', '2', '2', '2', '2', '1', '2', '1', '2', '2', '2', '1', 'YES']
['M', '72', '1', '1', '1', '1', '2', '2', '2', '2', '2', '2', '2', '1', '2', 'YES']
['F', '60', '2', '1', '1', '1', '1', '2', '1', '1', '1', '1', '2', '1', '1', 'NO']
['M', '58', '2', '1', '1', '1', '1', '2', '2', '2', '2', '2', '2', '1', '2', 'YES']
['M', '69', '2', '1', '1', '1', '1', '1', '2', '2', '2', '2', '1', '1', '2', 'NO']
['F', '48', '1', '2', '2', '2', '2', '2', '2', '2', '1', '2', '2', '2', '1', 'YES']
['M', '75', '2', '1', '1', '1', '2', '1', '2', '2', '2', '2', '1', '2', '2', 'YES']
['M', '57', '2', '2', '2', '2', '2', '1', '1', '1', '2', '1', '1', '2', '2', 'YES']
['F', '68', '2', '2', '2', '2', '2', '2', '1', '1', '1', '2', '2', '1', '1', 'YES']
['F', '61', '1', '1', '1', '1', '2', '2', '1', '1', '1', '1', '2', '1', '1', 'NO']
['F', '44', '2', '2', '2', '2', '2', '2', '1', '1', '1', '1', '2', '2', '1', 'YES']
['F', '64', '1', '2', '2', '2', '1', '1', '2', '2', '1', '2', '1', '2', '1', 'YES']
['F', '21', '2', '1', '1', '1', '2', '2', '1', '1', '1', '2', '1', '1', 'NO']
['M', '60', '2', '1', '1', '1', '1', '2', '2', '2', '2', '2', '2', '1', '2', 'YES']
['M', '72', '2', '2', '2', '2', '2', '1', '2', '2', '2', '2', '1', '2', '2', 'YES']
['M', '65', '1', '2', '2', '1', '1', '2', '1', '2', '2', '2', '2', '2', '2', 'YES']
['F', '61', '2', '2', '2', '1', '1', '2', '2', '1', '2', '1', '2', '2', '2', 'YES']
['M', '69', '1', '1', '1', '2', '1', '2', '1', '2', '1', '2', '2', '1', '2', 'NO']
['F', '53', '2', '2', '2', '1', '1', '2', '1', '2', '2', '1', '2', '2', '2', 'YES']
['M', '55', '1', '2', '1', '1', '1', '2', '1', '2', '2', '2', '2', '1', '1', 'NO']
['F', '57', '2', '2', '1', '1', '1', '1', '1', '1', '1', '1', '2', '1', '1', 'NO']
['M', '62', '2', '1', '2', '1', '1', '1', '2', '2', '2', '1', '2', '2', '2', 'YES']
['M', '56', '2', '2', '2', '1', '1', '1', '1', '1', '1', '1', '2', '2', '1', 'NO']
['F', '67', '2', '2', '2', '1', '2', '1', '1', '1', '1', '1', '2', '2', '2', 'YES']
['M', '59', '1', '2', '2', '1', '1', '1', '1', '1', '1', '1', '2', '2', '2', 'NO']
['F', '59', '2', '2', '2', '1', '2', '1', '1', '1', '1', '1', '2', '2', '1', 'YES']
['M', '60', '1', '2', '1', '1', '2', '1', '1', '2', '1', '2', '2', '1', '2', 'YES']
['F', '56', '1', '1', '1', '1', '2', '1', '1', '2', '1', '1', '2', '2', '1', 'NO']
['M', '56', '2', '1', '1', '1', '2', '1', '1', '2', '1', '1', '2', '1', '2', 'YES']
['M', '60', '2', '1', '1', '1', '2', '1', '2', '2', '2', '1', '1', '2', '2', 'YES']
['M', '68', '2', '1', '2', '1', '1', '2', '2', '1', '2', '2', '2', '1', '2', 'YES']
['M', '63', '1', '1', '1', '2', '1', '2', '2', '2', '2', '1', '1', '2', '1', 'YES']
['F', '77', '1', '2', '2', '2', '2', '2', '1', '2', '2', '1', '1', '1', '1', 'YES']
['M', '52', '2', '1', '1', '2', '1', '2', '2', '2', '2', '1', '2', '1', '2', 'YES']
['F', '70', '2', '2', '1', '2', '2', '1', '1', '1', '2', '2', '1', '2', '1', 'YES']
['M', '72', '2', '2', '2', '2', '2', '2', '1', '2', '2', '2', '2', '2', '2', 'YES']
['M', '62', '2', '2', '1', '1', '2', '1', '2', '1', '1', '2', '2', '2', '2', 'YES']
```

```
['F', '64', '2', '2', '1', '2', '1', '2', '1', '2', '2', '2', '1', '2', '2', 'YES']
['F', '70', '1', '1', '2', '2', '2', '2', '2', '2', '2', '1', '2', '2', '2', 'YES']
['M', '60', '1', '1', '2', '2', '2', '1', '1', '1', '2', '1', '1', '1', '1', 'NO']
['F', '56', '1', '1', '1', '2', '2', '2', '2', '2', '2', '2', '1', '1', '1', '2', 'YES']
['M', '63', '2', '2', '2', '1', '2', '2', '2', '2', '2', '1', '1', '2', '1', '1', 'YES']
['F', '54', '2', '1', '1', '2', '1', '2', '2', '2', '2', '2', '2', '1', '2', '2', 'YES']
['M', '49', '2', '1', '1', '2', '2', '2', '2', '2', '2', '2', '2', '2', '2', '2', 'YES']
['F', '57', '1', '2', '1', '2', '2', '2', '2', '1', '2', '2', '1', '1', '1', 'YES']
```

```
Start coding or generate with AI.
```