from PyQt5 import QtCore, QtGui, QtWidgets

- **PyQt5** is a set of Python bindings for Qt libraries used to create desktop applications. QtCore, QtGui, and QtWidgets are modules in PyQt5 that provide core functionalities, graphical interfaces, and widget tools, respectively.

import numpy as np

from keras.preprocessing import image

from keras.models import Sequential

from keras.layers import Dense

from keras.models import model_from_json

from keras.layers import Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout

from keras.preprocessing.image import ImageDataGenerator

- **numpy** is a library used for numerical computations in Python.
- **keras.preprocessing.image** provides utilities for preprocessing images before feeding them into a neural network.
- **keras.models.Sequential** is a linear stack of layers, used to create the neural network model.
- **Dense** is a fully connected neural network layer.
- **model_from_json** allows loading a model's architecture from a JSON file.
- **Conv2D** is a 2D convolution layer used to extract features from input images.
- **MaxPooling2D** reduces the spatial dimensions of the feature maps.
- **Flatten** converts the 2D feature maps into a 1D vector for the fully connected layer.
- **BatchNormalization** normalizes the activations of the previous layer at each batch.
- **Dropout** is a regularization technique to prevent overfitting.
- **ImageDataGenerator** generates batches of tensor image data with real-time data augmentation.

Additional Libraries

import os

import cv2

from imutils import contours

from sklearn.cluster import KMeans, spectral_clustering

from sklearn.neural_network import MLPClassifier

```
import csv

import scipy.io as sio

import imutils

import mahotas as mt
```

- **os** is used for interacting with the operating system, like reading or writing files.
- **cv2** is the OpenCV library, which is used for image processing.
- **imutils** is a library for image processing, mainly providing convenience functions.
- **sklearn.cluster** provides clustering algorithms like KMeans and spectral clustering.
- **MLPClassifier** is a neural network classifier from sklearn.
- **csv** is used for handling CSV files.
- **scipy.io** is used to read and write MATLAB files.
- **mahotas** is a library for image processing and computer vision tasks.

UI Class Definition

```
class Ui_MainWindow(object):

  def setupUi(self, MainWindow):

    ...
```

- **Ui_MainWindow** is a class that defines the user interface for the main window. The setupUi method is responsible for creating and arranging all the UI elements in the window.

```
MainWindow.setObjectName("MainWindow")

MainWindow.resize(800, 600)
```

- **setObjectName** sets the name of the window object.
- **resize** sets the dimensions of the main window to 800x600 pixels.

```
self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")
```

- **QWidget** is the base class for all UI objects in PyQt5. centralwidget is the main widget placed inside the main window.

Adding Buttons, Labels, and TextBox

```
self.BrowseImage = QtWidgets.QPushButton(self.centralwidget)

self.BrowseImage.setGeometry(QtCore.QRect(160, 370, 151, 51))
```

self.BrowseImage.setObjectName("BrowseImage")

- **QPushButton** creates a button widget. setGeometry sets the button's position and size (x=160, y=370, width=151, height=51).

self.imageLbl = QtWidgets.QLabel(self.centralwidget)

self.imageLbl.setGeometry(QtCore.QRect(200, 80, 361, 261))

self.imageLbl.setFrameShape(QtWidgets.QFrame.Box)

self.imageLbl.setText("")

self.imageLbl.setObjectName("imageLbl")

- **QLabel** creates a label widget. This label is used to display the image selected by the user. setFrameShape sets the frame style, and setText sets the text of the label, initially empty.

self.label_2 = QtWidgets.QLabel(self.centralwidget)

self.label_2.setGeometry(QtCore.QRect(110, 20, 621, 20))

font = QtGui.QFont()

font.setFamily("Courier New")

font.setPointSize(14)

font.setBold(True)

font.setWeight(75)

self.label_2.setFont(font)

self.label_2.setObjectName("label_2")

- **label_2** is a label used to display the title "COVID-19 DETECTION" with the specified font properties.

self.Classify = QtWidgets.QPushButton(self.centralwidget)

self.Classify.setGeometry(QtCore.QRect(160, 450, 151, 51))

self.Classify.setObjectName("Classify")

- **Classify** is a button that, when clicked, will classify the selected image as either "Covid" or "Normal".

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(430, 370, 111, 16))

self.label.setObjectName("label")

- **label** is a label that will display the recognized class after the classification.

self.Training = QtWidgets.QPushButton(self.centralwidget)

```
self.Training.setGeometry(QtCore.QRect(400, 450, 151, 51))

self.Training.setObjectName("Training")
```

- **Training** is a button that, when clicked, will start the training process of the CNN model.

```
self.textEdit = QtWidgets.QTextEdit(self.centralwidget)

self.textEdit.setGeometry(QtCore.QRect(400, 390, 211, 51))

self.textEdit.setObjectName("textEdit")
```

- **QTextEdit** is a widget for multiline text editing. It will display the result of the classification.

Menu Bar and Status Bar

```
self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))

self.menubar.setObjectName("menubar")

MainWindow.setMenuBar(self.menubar)
```

- **QMenuBar** creates a menu bar at the top of the window.

```
self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)
```

- **QStatusBar** creates a status bar at the bottom of the window to display status messages.

Connecting Buttons to Functions

```
self.BrowseImage.clicked.connect(self.loadImage)

self.Classify.clicked.connect(self.classifyFunction)

self.Training.clicked.connect(self.trainingFunction)
```

- **clicked.connect** is used to link the button clicks to their respective functions (loadImage, classifyFunction, trainingFunction).

Loading an Image

```
def loadImage(self):

    fileName, _ = QtWidgets.QFileDialog.getOpenFileName(None, "Select Image", "", "Image Files (*.png *.jpg *jpeg *.bmp);;All Files (*)")

    if fileName:

        print(fileName)
```

```
    self.file = fileName
    pixmap = QtGui.QPixmap(fileName)
    pixmap = pixmap.scaled(self.imageLbl.width(), self.imageLbl.height(), QtCore.Qt.KeepAspectRatio)
    self.imageLbl.setPixmap(pixmap)
    self.imageLbl.setAlignment(QtCore.Qt.AlignCenter)
```

- **QFileDialog.getOpenFileName** opens a file dialog to select an image. If a file is selected, it is loaded and displayed in imageLbl.

Classifying the Image

```
def classifyFunction(self):
  json_file = open('model.json', 'r')
  loaded_model_json = json_file.read()
  json_file.close()
  loaded_model = model_from_json(loaded_model_json)
  loaded_model.load_weights("model.h5")
  print("Loaded model from disk")
  label = ["Covid", "Normal"]
  path2 = self.file
  print(path2)
  test_image = image.load_img(path2, target_size=(128, 128))
  test_image = image.img_to_array(test_image)
  test_image = np.expand_dims(test_image, axis=0)
  result = loaded_model.predict(test_image)
  print(result)
  label2 = label[result.argmax()]
  print(label2)
  self.textEdit.setText(label2)
```

- **classifyFunction** loads a pre-trained model from model.json and its weights from model.h5. It processes the selected image, predicts whether it is "Covid" or "Normal", and displays the result in textEdit.

Training the Model

```
def trainingFunction(self):
    self.textEdit.setText("Training under process...")
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(128, 128, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))


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


    train_datagen = ImageDataGenerator(rescale=1. / 255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
    training_set = train_datagen.flow_from_directory('dataset/training_set', target_size=(128, 128), batch_size=32,
class_mode='binary')


    model.fit(training_set, epochs=10)


    model_json = model.to_json()
    with open("model.json", "w") as json_file:
        json_file.write(model_json)
    model.save_weights("model.h5")
    self.textEdit.setText("Training Completed!")
```

- **trainingFunction** defines a CNN model and trains it on a dataset of images. After training, the model's architecture and weights are saved to disk.

Completing the UI Setup

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
    self.BrowseImage.setText(_translate("MainWindow", "Browse Image"))
    self.label_2.setText(_translate("MainWindow", "COVID-19 DETECTION"))
    self.Classify.setText(_translate("MainWindow", "Classify"))
    self.label.setText(_translate("MainWindow", "Recognized Class"))
    self.Training.setText(_translate("MainWindow", "Training"))
```

- **retranslateUi** sets the text for all buttons, labels, and the window title.

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

- This block initializes and runs the application. QApplication is the main class that handles the event loop, and QMainWindow is the main window class. The application is started with app.exec_().