**CovidDetectionXray.py**

from tkinter import \*

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

from tkinter import simpledialog

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

from sklearn.preprocessing import LabelEncoder

main = tkinter.Tk()

main.title("Deep Learning Model for Detecting COVID‑19 on Chest X-Ray Using Convolutional Neural Networks")

main.geometry("1000x650")

global filename

global model\_acc

global classifier

global X, Y

disease =['Covid 19 Detected','X-Ray Normal']

def upload():

global filename

global dataset

filename = filedialog.askdirectory(initialdir = ".")

text.delete('1.0', END)

text.insert(END,filename+' Loaded\n\n')

def getLabel(label):

index = 0

for i in range(len(disease)):

if disease[i] == label:

index = i

break

return index

def preprocess():

global X, Y

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

test = X[3]

test = cv2.resize(test,(400,400))

cv2.imshow("test image",test)

cv2.waitKey(0)

X = X.astype('float32')

X = X/255

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

text.insert(END,"Total dataset processed image size = "+str(len(X)))

def buildCNNModel():

text.delete('1.0', END)

global classifier

global model\_acc

if os.path.exists('model/model.json'):

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

json\_file.close()

classifier.load\_weights("model/model\_weights.h5")

classifier.\_make\_predict\_function()

print(classifier.summary())

f = open('model/history.pckl', 'rb')

model\_acc = pickle.load(f)

f.close()

acc = model\_acc['accuracy']

accuracy = acc[9] \* 100

text.insert(END,"Covid-19 CNN Prediction Accuracy : "+str(accuracy))

else:

print(X.shape)

print(Y.shape)

classifier = Sequential()

#defining model with 32, 64 and 128 and due to system memory limit we have restrict image size to 64 X 64

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu')) #define CNN layer with image input size as 64 X 64 with 3 RGB colours

classifier.add(MaxPooling2D(pool\_size = (2, 2))) #max pooling layer to collect filter data

classifier.add(Convolution2D(64, 3, 3, activation = 'relu')) #defining another layer to further filter images

classifier.add(MaxPooling2D(pool\_size = (2, 2))) #max pooling layer to collect filter data

classifier.add(Flatten()) #convert images from 3 dimension to 1 dimensional array

classifier.add(Dense(output\_dim = 128, activation = 'relu')) #defining output layer

classifier.add(Dense(output\_dim = 21, activation = 'softmax')) #this output layer will predict 1 disease from given 21 disease images

print(classifier.summary())

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy']) #compile cnn model

hist = classifier.fit(X, Y, batch\_size=32, epochs=20, shuffle=True, verbose=2) #build covid detection CNN model with given Xtrain and Ytrain images

classifier.save\_weights('model/model\_weights.h5')

model\_json = classifier.to\_json()

with open("model/model.json", "w") as json\_file:

json\_file.write(model\_json)

json\_file.close()

f = open('model/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/history.pckl', 'rb')

model\_acc = pickle.load(f)

f.close()

acc = model\_acc['accuracy']

accuracy = acc[9] \* 100

text.insert(END,"Covid-19 CNN Prediction Accuracy : "+str(accuracy))

def predict():

global classifier

text.delete('1.0', END)

file = filedialog.askopenfilename(initialdir="testImages")

image = cv2.imread(file)

img = cv2.resize(image, (64,64))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,64,64,3)

img = np.asarray(im2arr)

img = img.astype('float32')

img = img/255

preds = classifier.predict(img)

predict\_disease = np.argmax(preds)

img = cv2.imread(file)

img = cv2.resize(img, (600,400))

cv2.putText(img, 'Disease predicted as : '+disease[predict\_disease], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 255), 2)

cv2.imshow('Disease predicted as : '+disease[predict\_disease], img)

cv2.waitKey(0)

def graph():

accuracy = model\_acc['accuracy']

loss = model\_acc['loss']

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Iterations/Epoch')

plt.ylabel('Accuracy/Loss')

plt.plot(accuracy, 'ro-', color = 'green')

plt.plot(loss, 'ro-', color = 'blue')

plt.legend(['Accuracy', 'Loss'], loc='upper left')

#plt.xticks(wordloss.index)

plt.title('Covid-19 Detection CNN Accuracy & Loss Graph')

plt.show()

def close():

main.destroy()

font = ('times', 14, 'bold')

title = Label(main, text='Deep Learning Model for Detecting COVID‑19 on Chest X-Ray Using Convolutional Neural Networks', justify=LEFT)

title.config(bg='mint cream', fg='olive drab')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=100,y=5)

title.pack()

font1 = ('times', 11, 'bold')

uploadButton = Button(main, text="Upload Covid-19 Chest Xray Dataset", command=upload)

uploadButton.place(x=10,y=100)

uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset", command=preprocess)

preprocessButton.place(x=300,y=100)

preprocessButton.config(font=font1)

contextButton = Button(main, text="Build CNN Covid-19 Detection Model", command=buildCNNModel)

contextButton.place(x=480,y=100)

contextButton.config(font=font1)

predictButton = Button(main, text="Upload Test Data & Predict Disease", command=predict)

predictButton.place(x=780,y=100)

predictButton.config(font=font1)

graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)

graphButton.place(x=10,y=150)

graphButton.config(font=font1)

closeButton = Button(main, text="Close Application", command=close)

closeButton.place(x=300,y=150)

closeButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=160)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='gainsboro')

main.mainloop()