**SOURCE CODE**

import os

import sys

# Flask

from flask import Flask, redirect, url\_for, request, render\_template, Response, jsonify, redirect

from werkzeug.utils import secure\_filename

from gevent.pywsgi import WSGIServer

# TensorFlow and tf.keras

import tensorflow as tf

from tensorflow import keras

# Preprocessing utilities

from tensorflow.keras.applications.imagenet\_utils import preprocess\_input, decode\_predictions

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

# Model building

from keras import layers

from keras.optimizers import Adam

from keras.models import Sequential

from keras.applications import DenseNet121

from keras.callbacks import Callback, ModelCheckpoint

from PIL import Image

from models.model import build\_model, preprocess\_image

# Some utilites

import numpy as np

from utils import base64\_to\_pil

# Creating a new Flask Web application.

app = Flask(\_\_name\_\_)

# Model saved with Keras model.save()

MODEL\_PATH = './models/model.h5'

# Loading trained model

model = build\_model()

model.load\_weights(MODEL\_PATH)

print('Model loaded. Start serving...')

def model\_predict(img, model):

## Preprocessing the image

x\_val = np.empty((1, 224, 224, 3), dtype=np.uint8)

img = img.resize((224,) \* 2, resample=Image.LANCZOS)

x\_val[0, :, :, :] = img

preds = model.predict(x\_val)

return preds

@app.route('/', methods=['GET'])

def index():

# Main page

return render\_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])

def predict():

if request.method == 'POST':

# Get the image from post request

img = base64\_to\_pil(request.json)

# Make prediction on the image

preds = model\_predict(img, model)

# Process result to find probability and class of prediction

pred\_proba = "{:.3f}".format(np.amax(preds)) # Max probability

pred\_class = np.argmax(np.squeeze(preds))

diagnosis = ["No DR", "Mild", "Moderate", "Severe", "Proliferative DR"]

result = diagnosis[pred\_class] # Convert to string

# Serialize the result

return jsonify(result=result, probability=pred\_proba)

return None

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000,debug=True, threaded=False)