from flask import Flask, render\_template, requestpi

#from PIL import Image

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

#from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

import cv2

import os

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

class\_labels = ['F', 'N', 'Q', 'S', 'V']

model = load\_model('LeNet\_model.h5')

img\_size = (299, 299)

def test\_single\_img(img\_path):

    img = image.load\_img(img\_path, target\_size=img\_size,color\_mode='grayscale')

    img\_array = image.img\_to\_array(img)

    img\_array = np.expand\_dims(img\_array, axis=0)

    img\_array /= 255.0 # Normalize pixel values

    prediction = model.predict(img\_array)

    predicted\_class = np.argmax(prediction)

    return class\_labels[predicted\_class]

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/about')

def about():

    return render\_template('about.html')

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

def prediction():

    prediction\_result = None

    img\_path = None

    if request.method == 'POST':

            img = request.files['my\_image2']

            img\_path = "static/" + img.filename

            img.save(img\_path)

# Get the prediction result

            prediction\_result = test\_single\_img(img\_path)

    return render\_template('prediction.html',

prediction\_result=prediction\_result, img\_path=img\_path)

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

    app.run(debug=True)