Project code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import cv2
import random
import numpy as np
from tensorflow.keras.utils import to_categorical  # Updated import for to_categorical
from tensorflow.keras.layers import MaxPooling2D, Dense, Dropout, Activation, Flatten, Conv2D
from tensorflow.keras.models import Sequential, model_from_json
import pickle
import os



main = tkinter.Tk()
main.title("Non-Binary Image Classification using Convolution Neural Networks")
main.geometry("1300x1200")

global filename
global classifier

names = ['Palm', 'I', 'Fist', 'Fist Moved', 'Thumb', 'Index', 'OK', 'Palm Moved', 'C', 'Down']
bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)

def remove_background(frame):
    fgmask = bgModel.apply(frame, learningRate=0)
```

```
    kernel = np.ones((3, 3), np.uint8)
fgmask = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=fgmask)
    return res

def uploadDataset():
    global filename
    global labels
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END, filename + " loaded\n\n")



def trainCNN():
    global classifier
    text.delete('1.0', END)
    X_train = np.load('model/X.txt.npy')
    Y_train = np.load('model/Y.txt.npy')
    text.insert(END, "CNN is training on total images : " + str(len(X_train)) + "\n")

    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)
        classifier.load_weights("model/model_weights.h5")
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
```

```
        acc = data['accuracy']
        accuracy = acc[-1] * 100  # Updated to fetch the last epoch's accuracy
        text.insert(END, "CNN Hand Gesture Training Model Prediction Accuracy = " + str(accuracy))
    else:
classifier = Sequential()
        classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2, 2)))
        classifier.add(Conv2D(32, (3, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2, 2)))
        classifier.add(Flatten())
        classifier.add(Dense(256, activation='relu'))
        classifier.add(Dense(len(names), activation='softmax'))  # Updated for dynamic output size
        print(classifier.summary())
        classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
        hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=10, shuffle=True, verbose=2)
        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)
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[-1] * 100  # Updated
        text.insert(END, "CNN Hand Gesture Training Model Prediction Accuracy = " + str(accuracy))



def classifyGesture():
```

```
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename, cv2.IMREAD_COLOR)
    img = cv2.flip(img, 1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (41, 41), 0)
    ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    thresh = cv2.resize(thresh, (224, 224))
    thresh = np.array(thresh)
    frame = np.stack((thresh,) * 3, axis=-1)
    frame = cv2.resize(frame, (64, 64))
    frame = frame.reshape(1, 64, 64, 3)
    frame = np.array(frame, dtype='float32')
    frame /= 255
    predict = classifier.predict(frame)
    result = names[np.argmax(predict)]
    img = cv2.imread(filename)
    img = cv2.resize(img, (600, 400))
    cv2.putText(img, 'Hand Gesture Classified as : ' + result, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,
0.7, (255, 0, 0), 2)
    cv2.imshow('Hand Gesture Classified as : ' + result, img)
    cv2.waitKey(0)



def webcamPredict():
    videofile = askopenfilename(initialdir="video")
    video = cv2.VideoCapture(videofile)
    while video.isOpened():
        ret, frame = video.read()
        if ret:
            img = frame
            img = cv2.flip(img, 1)
```

```
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray, (41, 41), 0)
            ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            thresh = cv2.resize(thresh, (64, 64))
            thresh = np.array(thresh)
            img = np.stack((thresh,) * 3, axis=-1)
            img = cv2.resize(img, (64, 64))
            img = img.reshape(1, 64, 64, 3)
            img = np.array(img, dtype='float32')
            img /= 255
            predict = classifier.predict(img)
            print(np.argmax(predict))
            result = names[np.argmax(predict)]
            cv2.putText(frame, 'Gesture Recognized as : ' + str(result), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
            cv2.imshow("video frame", frame)
            if cv2.waitKey(950) & 0xFF == ord('q'):
                break
        else:
            break
    video.release()
    cv2.destroyAllWindows()



font = ('times', 16, 'bold')
title = Label(main, text='Hand Gesture Recognition using Convolution Neural Networks', anchor=W,
justify=CENTER)
title.config(bg='yellow4', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0, y=5)
```

```
font1 = ('times', 13, 'bold')
upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)
upload.place(x=50, y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='yellow4', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50, y=150)

markovButton = Button(main, text="Train CNN with Gesture Images", command=trainCNN)
markovButton.place(x=50, y=200)
markovButton.config(font=font1)

lexButton = Button(main, text="Upload Test Image & Recognize Gesture", command=classifyGesture)
lexButton.place(x=50, y=250)
lexButton.config(font=font1)

predictButton = Button(main, text="Recognize Gesture from Video", command=webcamPredict)
predictButton.place(x=50, y=300)
predictButton.config(font=font1)


font1 = ('times', 12, 'bold')
text = Text(main, height=15, width=78)
scroll = Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=450, y=100)
text.config(font=font1)
```

```
main.config(bg='magenta3')
main.mainloop()
```