1. File data cho training:

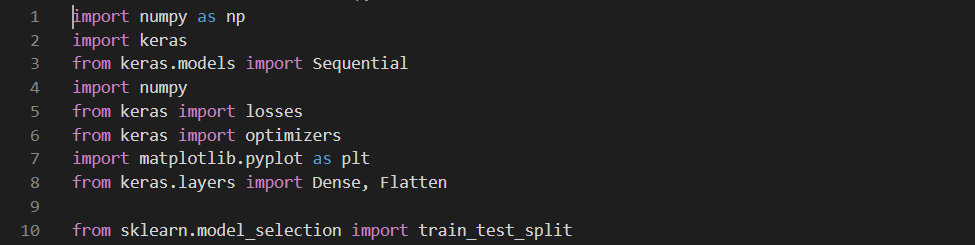
+ trainingData.csv: File data collect cho tay phải, chỉ bao gồm các ký tự đơn, mỗi dòng có 23 phần tử, 22 phần tử đầu là các góc tương đối của tay phải, phần từ cuối là label=> cho training nhận diện riêng ký tự đơn (không bao gồm các chữ)

+ twoHandData.csv: File data collect cho 2 tay phải và trái cho các chữ “need”, “more”, “help”. Mỗi dòng gồm 45 phần tử, 22 phần từ đầu là các góc của tay phải, 22 phần tử sau là các góc của tay trái, phần tử cuối là label=>cho training nhận diện riêng các chữ (không có các ký tự đơn)

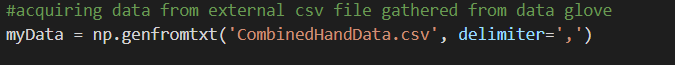
+ CombineHandData.csv: File data ghép bởi 2 file phía trên, cho việc training nhận diện cả ký tự đơn cả từ cùng 1 lúc. Mỗi dòng gồm 45 phần tử, 22 phần từ đầu là các góc của tay phải, 22 phần tử sau là các góc của tay trái (nếu là ký tự đơn thì 22 phần tử sau lúc này sẽ đều là giá trị 0 vì ký tự đơn chỉ đc biểu diễn bằng 1 tay,tay trái sẽ không connect nên chỉ có 22 giá trị 0), phần tử cuối là label. => Đây là file chính dùng để training. Có tất cả 32321 samples được collect

2. File code cho training:

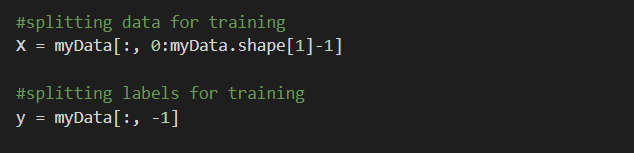
a.Trainer\_MPL.py: Chạy file này để thực hiện việc training dùng MLP



Import các thư viện cần thiết.

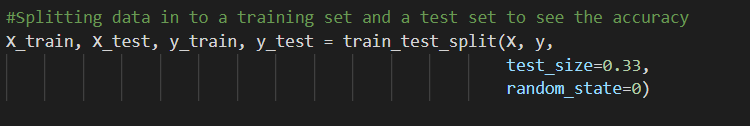


Đọc data từ file dữ liệu

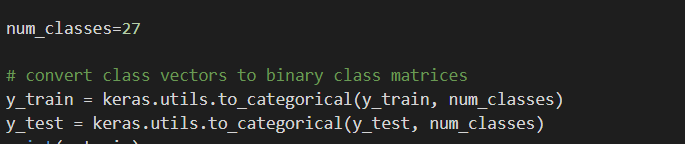


X là mảng chứa dữ liệu đưa vào training, là 1 mảng chứa 44 phần tử

Y: là mảng chứa các label tương ứng

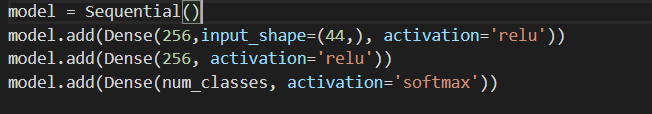


Chia dữ liệu thành dữ liệu train và dữ liệu valid với tỷ lệ 7:3

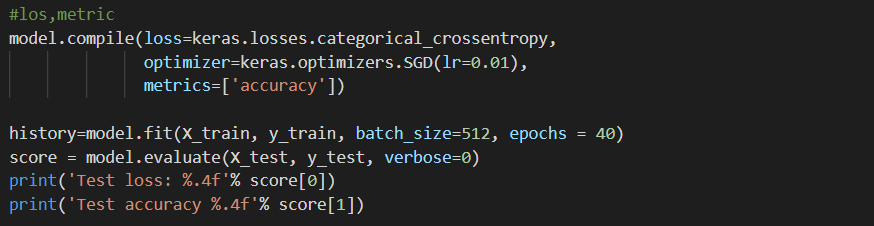


Sau khi loại bỏ 1 số ký tự có hình dạng giống nhau gây nhiễu mô hình => có 27 classes được đưa vào training

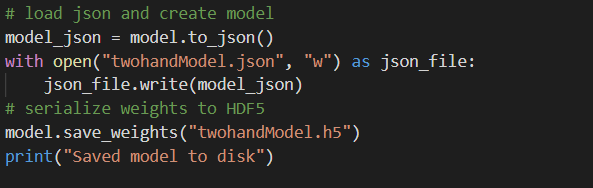
Dùng one-hot coding chuyển đổi label từ giá trị số sang vector cùng kích thước với output model.Ví dụ label được đánh là 1 sẽ được chuyển thành [0,1,0,0,…n], n=27



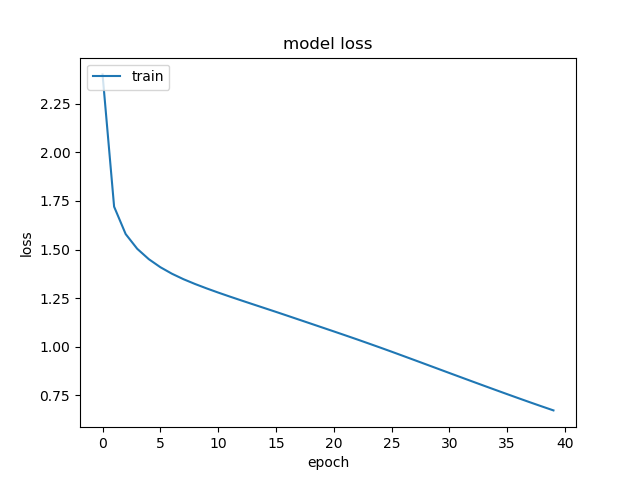
Xây dựng model sử dụng keras. input layer gồm 44 nodes, hidden layer thứ 1 và thứ 2 đều gồm 256 node, hàm activation là relu, lớp ouput gồm 27 lớp, dùng softmax function để chuyển giá trị thực trong các node ở output layer sang giá trị phần trăm. Index của node có giá trị phần tram lớn nhất sẽ là output predict dc



Sử dụng loss function là cross entropy, learning rate là 0,01, epochs=40. => Chạy training



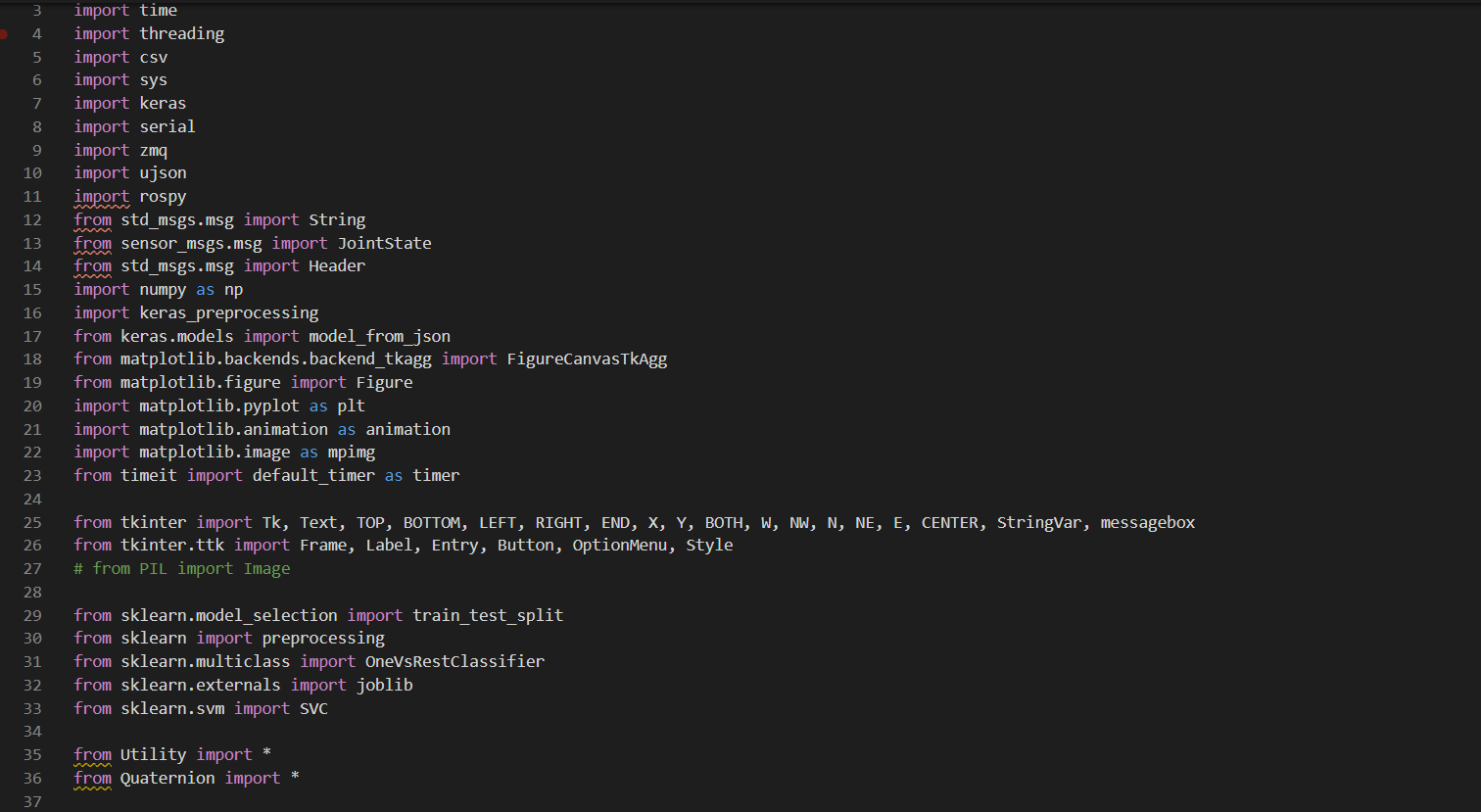
Save model



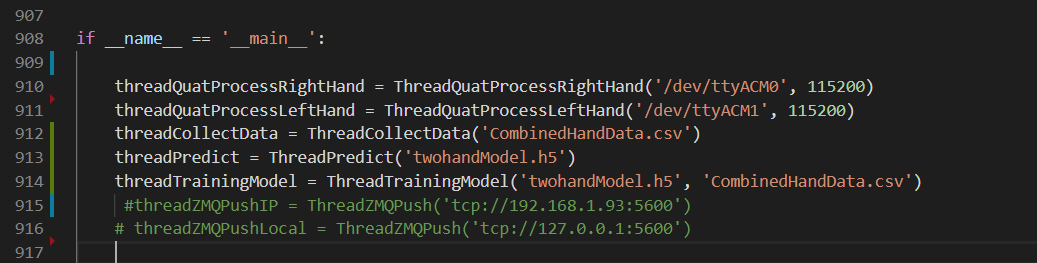
Đồ thị loss

3. File code chương trình chính:

VRGlove\_Full.py: Thực hiện kết nối gang tay, tính toán góc, chạy giao diện cho việc dự đoán ký tự



Import thư việc và các module cần thiết



Các luồng chạy trong chương trình. Chương trình sẽ được chạy đa luồng. Có thể đọc data từ tay phải, tay trái, collect data và predict đồng thời cùng lúc

- ThreadQuatProcessRightHand('/dev/ttyACM0', 115200)

Luồng xử lý cho tay phải

Đọc tay phải qua cổng com ttyACM0 và bound rate 115200

Giá trị 22 góc tính được sẽ lưu vào mảng JointAngleRight

class ThreadQuatProcessRightHand(threading.Thread):

def \_\_init\_\_(self, serialPort, baudRate):

threading.Thread.\_\_init\_\_(self)

self.shutdown\_flag = threading.Event()

self.serialPort = serialPort

self.baudRate = baudRate

self.num=0

# def changeSerialPort(self, serialPort):

# self.ser.close()

# self.serialPort = serialPort

# self.seor = serial.Serial(self.serialPort, self.baudRate)

def run(self):

try:

with serial.Serial(self.serialPort, self.baudRate) as self.ser:

string = ''

while not self.shutdown\_flag.is\_set():

byte = self.ser.read()

if byte != '$'.encode('ascii'):

string += byte.decode('ascii', 'ignore')

if byte == '\n'.encode('ascii'):

print(string)

string = ''

else:

quatPacket = read\_packet(self.ser)

if quatPacket is None:

print('Failed to receive string start with $')

else:

def updateAngle():

nonlocal quatPacket

global jointAngleRight

def calcAngle(joint, bone1, bone2, axis):

nonlocal quatPacket

global jointAngleRight

if bone1 in quatPacket and bone2 in quatPacket:

roll, pitch, yaw = getRelativeAngle(quatPacket[bone1], quatPacket[bone2])

if axis == 'roll':

jointAngleRight[joint] = round(roll, 3)

elif axis == 'pitch':

jointAngleRight[joint] = round(roll, 3)

elif axis == 'yaw':

jointAngleRight[joint] = round(roll, 3)

def copyAngle(destination, source):

global jointAngleRight

jointAngleRight[destination] = jointAngleRight[source]

def scaleAngle(joint, offset, scale):

global jointAngleRight

jointAngleRight[joint] = round(scale\*(jointAngleRight[joint]) + offset, 3)

calcAngle(joint\_R\_forearm\_carpal, hand\_wist, index\_mc, 'yaw')

calcAngle(joint\_R\_carpal\_hand, hand\_wist, index\_mc, 'roll')

calcAngle(joint\_R\_hand\_thumb , thumb\_mc, index\_mc, 'pitch')

scaleAngle(joint\_R\_hand\_thumb, 0, 1.1)

calcAngle(joint\_R\_thumb\_tp\_mc, thumb\_mc, index\_mc, 'yaw')

scaleAngle(joint\_R\_thumb\_tp\_mc, 0, 1)

calcAngle(joint\_R\_thumb\_mc\_pp, thumb\_mc, thumb\_pp, 'roll')

scaleAngle(joint\_R\_thumb\_mc\_pp, 1.5, 0.7) #

calcAngle(joint\_R\_thumb\_pp\_dp, thumb\_pp, thumb\_dp, 'roll')

scaleAngle(joint\_R\_thumb\_pp\_dp, 0, 2)

calcAngle(joint\_R\_hand\_index , index\_mc, index\_pp, 'yaw')

scaleAngle(joint\_R\_hand\_index,0,0.6)

calcAngle(joint\_R\_index\_mc\_pp, index\_mc, index\_pp, 'roll')

calcAngle(joint\_R\_index\_pp\_mp, index\_pp, index\_mp, 'roll')

copyAngle(joint\_R\_index\_mp\_dp, joint\_R\_index\_pp\_mp)

calcAngle(joint\_R\_hand\_middle , index\_mc , middle\_pp, 'yaw')

scaleAngle(joint\_R\_hand\_middle,0,0.6)

calcAngle(joint\_R\_middle\_mc\_pp, index\_mc , middle\_pp, 'roll')

calcAngle(joint\_R\_middle\_pp\_mp, middle\_pp, middle\_mp, 'roll')

copyAngle(joint\_R\_middle\_mp\_dp, joint\_R\_middle\_pp\_mp)

calcAngle(joint\_R\_hand\_ring , index\_mc, ring\_pp, 'yaw')

scaleAngle(joint\_R\_hand\_ring,0,0.6)

calcAngle(joint\_R\_ring\_mc\_pp, index\_mc, ring\_pp, 'roll')

calcAngle(joint\_R\_ring\_pp\_mp, ring\_pp , ring\_mp, 'roll')

copyAngle(joint\_R\_ring\_mp\_dp, joint\_R\_ring\_pp\_mp)

calcAngle(joint\_R\_hand\_pinky , index\_mc, pinky\_pp, 'yaw')

scaleAngle(joint\_R\_hand\_pinky,0,0.6)

calcAngle(joint\_R\_pinky\_mc\_pp, index\_mc, pinky\_pp, 'roll')

calcAngle(joint\_R\_pinky\_pp\_mp, pinky\_pp, pinky\_mp, 'roll')

copyAngle(joint\_R\_pinky\_mp\_dp, joint\_R\_pinky\_pp\_mp)

for angle in jointAngleRight:

if angle == 0.0:

self.num=self.num+1

if self.num >=3:

print(jointAngleRight)

print("0")

messagebox.showerror("Error")

self.shutdown\_flag.set()

self.num = 0

print(jointAngleRight)

print("right")

updateAngle()

except:

print("Unexpected error:", sys.exc\_info()[0])



Luồng xử lý cho tay trái

Tương tự như tay phải, đọc qua cổng com ttyACM1, 22 góc được tính là lưu vào mảng jointAngleLeft



Luồng collect data: vì luồng đọc dữ liệu từ tay trái và phải chạy song song=> ghép các giá trị thu được từ 2 mảng jointAngleLeft và jointAngleRight thành 1 mảng jointAngle có 44 phần tử, 22 phần tử đầu là giá trị của jointAngleRight, 22 phần tử sau là giá trị của jointAngleLeft. 44 giá trị góc này + 1 giá trị label sẽ được lư vào file CombineData.csv

class ThreadCollectData(threading.Thread):

def \_\_init\_\_(self,csvFilename):

threading.Thread.\_\_init\_\_(self)

self.shutdown\_flag=threading.Event()

self.csvFilename = csvFilename

self.trainingCharacter = '2'

self.collectSampleEnable = False

self.sampleCount = 0

self.count=0

def run(self):

with open(self.csvFilename, mode='a') as trainerFile:

global fieldnames\_twoHand

global fieldnames

global jointAngle

global jointAngleRight

global jointAngleLeft

writer = csv.DictWriter(trainerFile, fieldnames=fieldnames\_twoHand)

while not self.shutdown\_flag.is\_set():

if(self.collectSampleEnable==True):

jointAngle=jointAngleRight+jointAngleLeft

for angle in jointAngle:

if angle == 0.0:

self.count =self.count+1

if self.count >=4:

print(jointAngle)

print("2 hand")

messagebox.showerror("Error")

else:

print(jointAngle)

newRow = dict(zip(fieldnames\_twoHand, jointAngle))

newRow.update({'character':self.trainingCharacter})

writer.writerow(newRow)

self.sampleCount += 1

print('Sample count: ' + str(self.sampleCount))

time.sleep(0.05)

self.count=0



Luồng predict

class ThreadPredict(threading.Thread):

    def \_\_init\_\_(self,modelFile):

        threading.Thread.\_\_init\_\_(self)

        self.shutdown\_flag=threading.Event()

        self.modelFile=modelFile

        self.json\_file=open('twohandModel.json', 'r')

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

        self.json\_file.close()

        self.load\_model=model\_from\_json(self.loaded\_model\_json)

        self.load\_model.load\_weights(modelFile)

        self.load\_model.compile(loss=keras.losses.categorical\_crossentropy,

              optimizer=keras.optimizers.SGD(lr=0.01),

              metrics=['accuracy'])

        self.predictResult = 'N/A'

        self.predictEnable = True

        self.MultiplePredictEnable = False

        self.timeConstant=timeConstant

        self.Count=0

    def run(self):

        while not self.shutdown\_flag.is\_set() and self.MultiplePredictEnable:

            if self.predictEnable:

                global jointAngleRight

                global jointAngleLeft

                global jointAngle

                global characterList

                jointAngle=jointAngleRight+jointAngleLeft

                self.Count=self.Count+1

                temp = np.array(jointAngle).reshape(1, -1)

                y\_array=self.load\_model.predict(temp)

                index=np.argmax(y\_array)

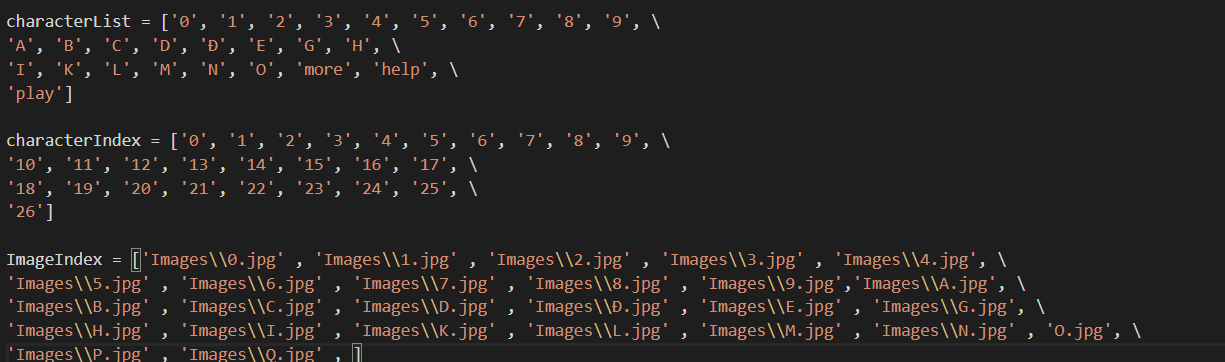
                self.predictResult=characterList[int(index)]

                print(self.predictResult)

                app.predictResult.insert(END, self.predictResult+" ")

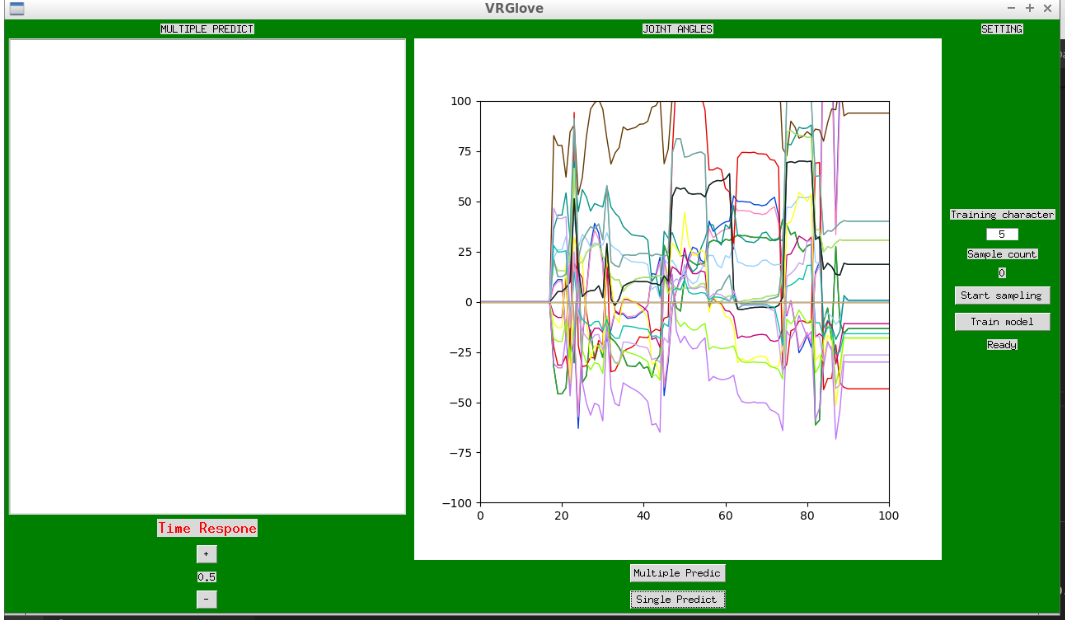
                time.sleep(self.timeConstant)

Cơ chế predict: model sau khi training sẽ được load vào. Đầu vào sẽ giá trị kết hợp của cả tay phải và tay trái như phần collect data. jointAngle=jointAngleRight+jointAngleLeft. Kết quả đầu tien sau hàm **load\_model.predict(temp)** là vector chứa 27 phần tử, mỗi phần tử là xác xuất dự đoán có thể rơi vào class tương ứng. Hàm **np.argmax** trả gia chỉ số của phần tử có xác xuất cao nhất. Chỉ số sẽ được match vào characterList để lấy ra phần tử tương ứng đồng nghĩa với ký tự dự đoán ra. CharacterList là list các ký tự đã được học. characterIndex là list các label được đánh tương ứng với các ký tự trong characterList



4. Hướng dẫn chạy chương trình:

Chạy file VRGlove\_Full.py để chạy chương trình và giao diện



1.Thu thập dữ liệu

characterList = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', \

'A', 'B', 'C', 'D', 'Đ', 'E', 'G', 'H', \

'I', 'K', 'L', 'M', 'N', 'O', 'more', 'help', \

'play']

characterIndex = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', \

'10', '11', '12', '13', '14', '15', '16', '17', \

'18', '19', '20', '21', '22', '23', '24', '25', \

'26']

Điền label cho ký tự chuẩn bị thu thập theo 2 list phía trên. Ví dụ muốn thu thập ký tự “0” thì điền dưới ô Training character số 0, muốn thu thập chữ more thì điền số 24.

Sau đó kết nối tay với máy tính và tạo hình dáng ký tự muốn thu thập, sau đó ấn start sampling. Khi nào muốn dừng thu thập thì ấn stop sampling. Nếu muốn thu thập ký tự đơn, thì chỉ kết nối tay phải, không kết nối tay trái vì ký tự đơn chỉ được biểu diễn bằng 1 tay. Lúc nãy mảng jointAngle sẽ mang 44 phần tử với 22 phần tử đầu là góc tay phải 22 phần tử sau đều là các giá trị 0. Nếu muốn thu thập các chữ “more, help,play”, kêt nối cả 2 tay, lúc này jointAngle sẽ có 44 giá trị từ 2 tay. Dữ liệu thu thập đc sẽ save vào file CombineData.csv

2. Nhận diện: Tương tự như lúc thu thập, muốn nhận diện ký tự đơn chỉ kết nối tay phải, muốn nhận diện từ thì kết nối cả 2 tay.

a. Single predict mode:Ấn button single predict Sẽ show ảnh của ký tự được nhận ra. Dùng chỉ số mà model trả ra để match vào list sau để show ảnh tương ứng:

ImageIndex = ['Images\\0.jpg' , 'Images\\1.jpg' , 'Images\\2.jpg' , 'Images\\3.jpg' , 'Images\\4.jpg', \

'Images\\5.jpg' , 'Images\\6.jpg' , 'Images\\7.jpg' , 'Images\\8.jpg' , 'Images\\9.jpg','Images\\A.jpg', \

'Images\\B.jpg' , 'Images\\C.jpg' , 'Images\\D.jpg' , 'Images\\Đ.jpg' , 'Images\\E.jpg' , 'Images\\G.jpg', \

'Images\\H.jpg' , 'Images\\I.jpg' , 'Images\\K.jpg' , 'Images\\L.jpg' , 'Images\\M.jpg' , 'Images\\N.jpg' , 'O.jpg', \

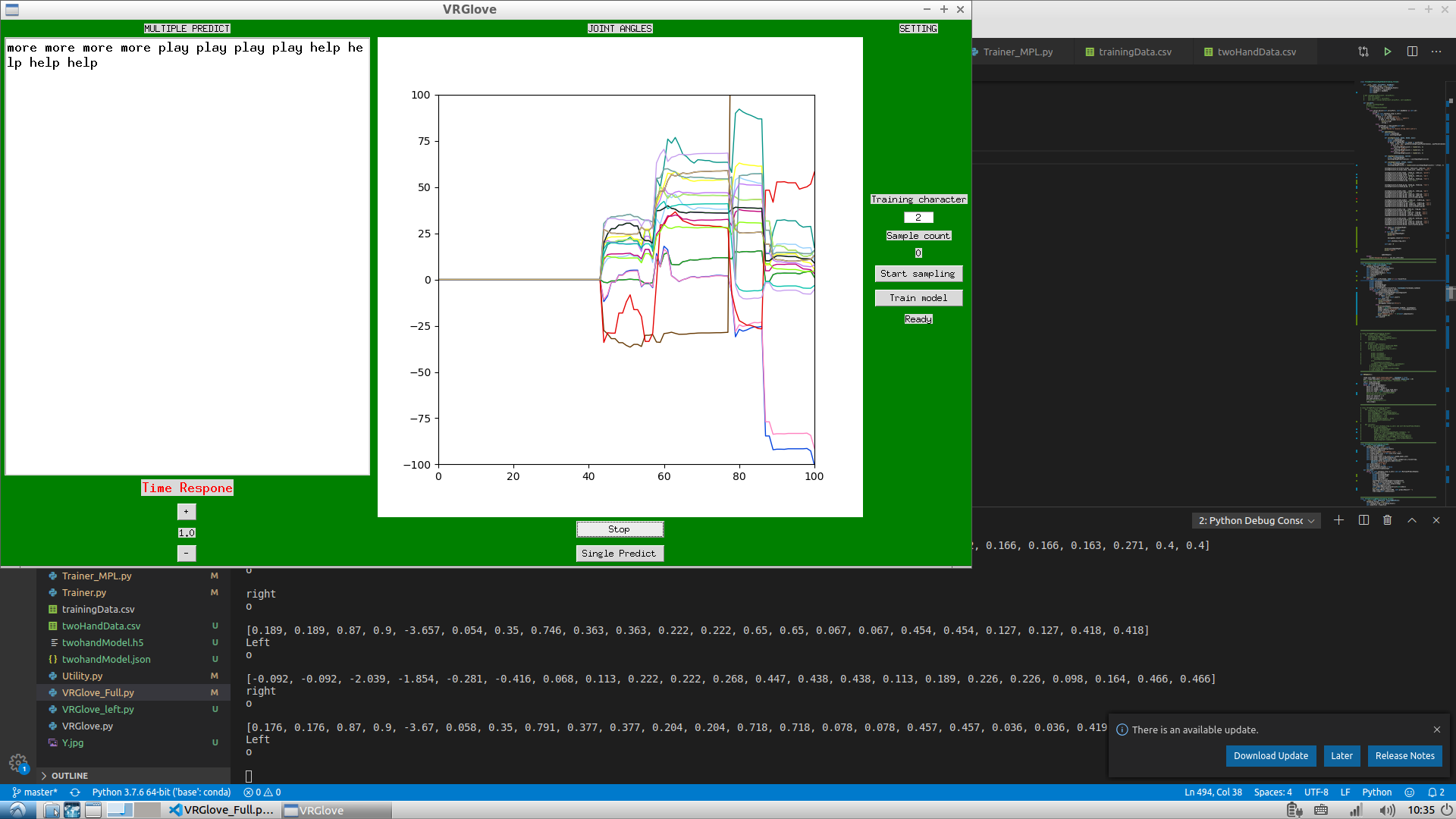
'Images\\P.jpg' , 'Images\\Q.jpg' , ]

Là list chứa đường dẫn ảnh



Kết quả

b. Mode Multiple predict : Ấn button Multiple predict



Mục tiêu để nhận diện liên tục và ghép thành câu và show lên text box. Nhưng hầu hết các từ đều phải kết hợp cả chuyển động nên chỉ mới đưa ra được 3 từ ‘need’,’more’,’help’. Phần điều chỉnh time response để điều chỉnh thời gian dự đoán, tránh bj lặp từ lúc ghép câu

3. Train Model : Ấn button train model sau khi đã thu thập đầy đủ dữ liệu (thay cho việc chạy file Trainer\_MLP.py