BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

----------Δ ----------

BÁO CÁO CUỐI KỲ
XÂY DỰNG THUẬT TOÁN NHẬN DẠNG
CỬ CHỈ BÀN TAY

Môn học: Trí tuệ nhân tạo
Họ và tên sinh viên: Lê Minh Trí
MSSV: 19146038
Lớp: Chiều thứ 2 (AI_CLC3A)

Giảng viên hướng dẫn: PGS.TS Nguyễn Trường Thịnh

TP HỒ CHÍ MINH NGÀY 22 THÁNG 06 NĂM 2022

Mục Lục

CHƯƠNG I: ĐẶT VẤN ĐỀ

A/ Đặt vấn đề:

Ngày này việc sử dụng các ngôn ngữ ngày càng phổ biến, trao đổi giao tiếp là một điều gì đó không thể thiếu trong xã hội ngày nay. Ngôn ngữ càng đa dạng càng nhiều loại phổ biến hơn. Nhưng làm sao để một người không thể nói được đúng hơn là một số trường hợp người có hoàn cảnh khó khăn trong việc giao tiếp thông qua ngôn ngữ âm thanh.

B/ Ý tưởng và giải pháp:

1. Ý tưởng

Dựa trên vấn đề đã được đặt ra, cần xây dựng một giao diện đúng hơn là một thuật toán giúp những trường hợp trên để họ có thể giao tiếp với những người xung quanh dễ dàng hơn, tránh trường hợp bị cô lập với xã hội.

2. Giải pháp:

Thông qua model được tạo ra từ quá trình huấn luyện dữ liệu (Dữ liệu ở đây là hình ảnh về các cử tay khác nhau đã được thu thập.) Thuật toán thông qua model này và thông qua webcam hoặc hình ảnh mà người dùng chụp cử chỉ bàn tay đưa lên chương trình, từ đó đánh giá và đưa ra kết quả gần nhất với các ngôn ngữ đã thiết lập từ trước. Qua đó, đưa ra kết ngôn ngữ người đó muốn nói là gì.

3. Giới thiệu về Tensorflow

Tensorflow là một thư viện JavaScript để đưa các model mà ta huấn luyện được từ Python để có thể chạy trên trình duyệt. HDF5 → Json.

Trong dự án này, ta sẽ sử dụng thư viện Tensorflowjs để đưa model đã được train bằng Python (model.h5, best.hdf5) sang model.json nhằm để phục vụ cho quá trình xử lý nhận dạng hình ảnh và đưa ra dự đoán.

CHƯƠNG II: TÍNH TOÁN THIẾT KẾ VÀ XÂY DỰNG SẢN PHẨM

A: Quy trình nghiên cứu:

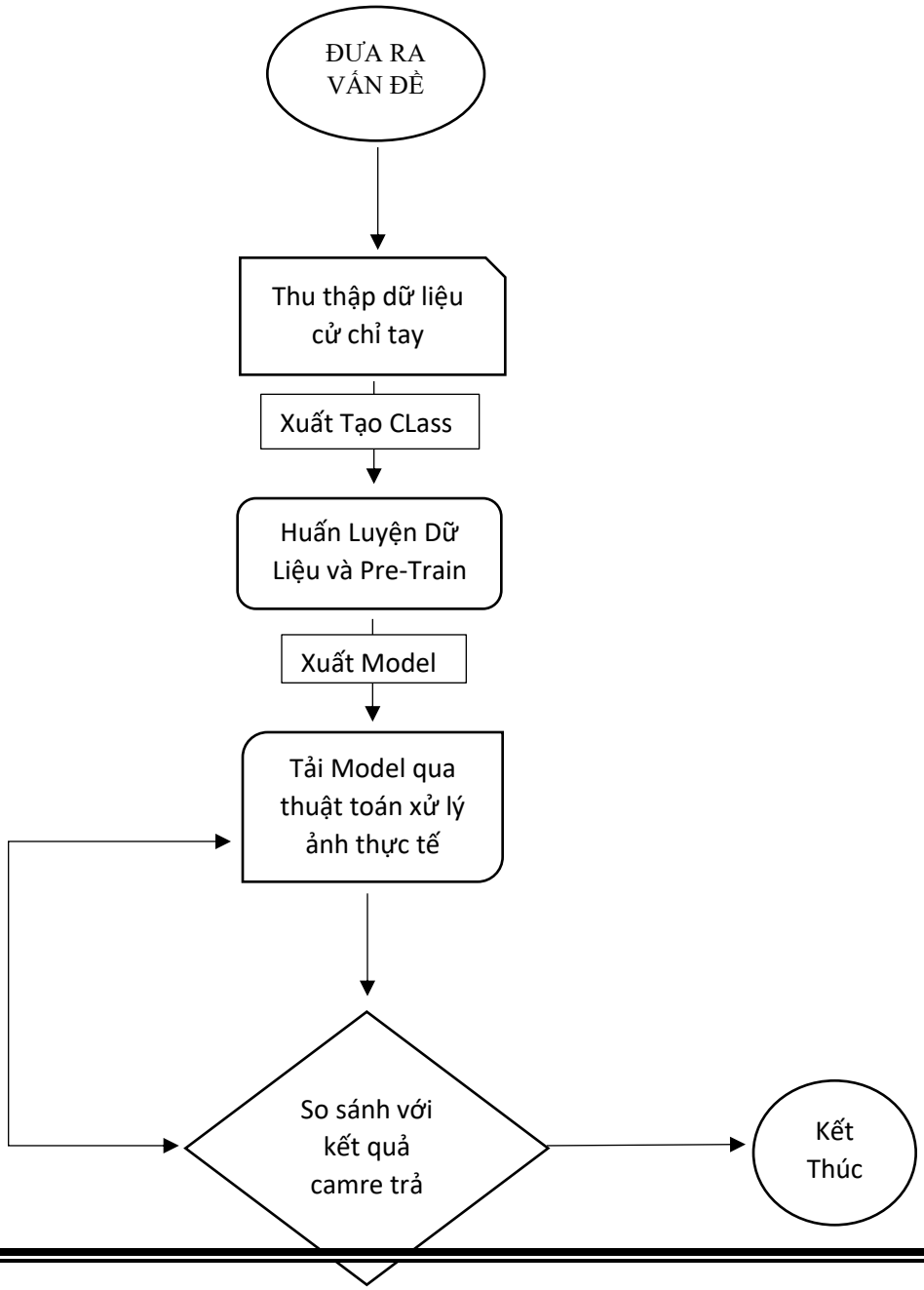
Vấn đề → Ý tưởng → Giải pháp → Nguyên lý hoạt động → Thiết kế → Thi công → Vận hành và thử nghiệm → Thu kết quả → Đánh giá → Viết báo cáo → Ứng dụng sản phẩm vào thực tế.

B: Thiết kế và thi công sản phẩm

I/ Nguyên lý hoạt động:

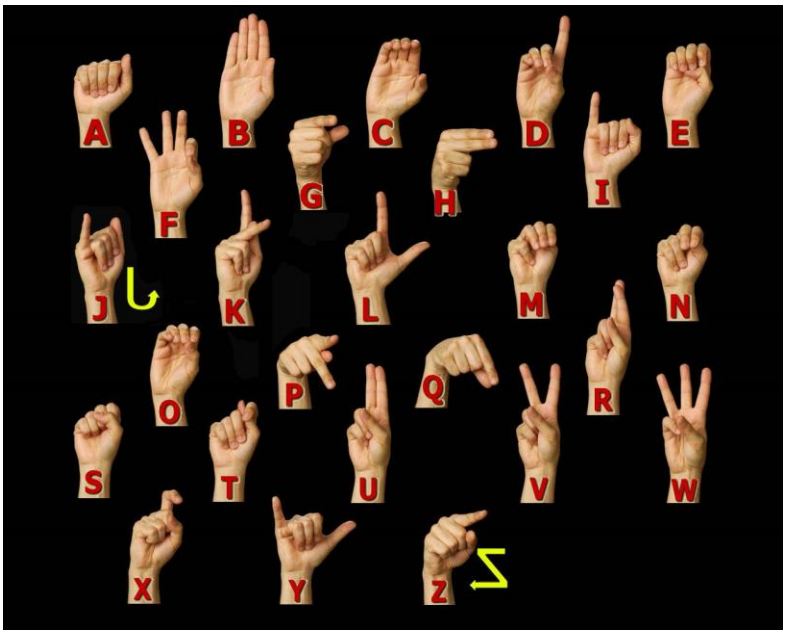
Bàn tay con người có 5 ngón và các đốt cử động thuật toán dựa vào các điểm ở đốt ngón tay co duỗi mà đưa ra phương án giải quyết là từ những cử chỉ đó là ký tự gì.

Dựa vào cử chỉ ngón tay đã được thu thập từ trước, các tập dữ liệu này được chia thành từng nhóm và đánh nhãn gọi là các "Class" (Ở dự án này có tổng cộng 24 Class). Mỗi Class là một loại cử chỉ tay khác nhau. Từ đó, thông qua hình ảnh được người dùng đưa lên từ camera để chuẩn đoán thực tế là chữ cái hay ký hiệu gì mà người khác muốn truyền đạt. Sơ đồ giải thuật sau đây sẽ cho chúng ta thấy rõ.



II/ Tạo dữ liệu:

Cử chỉ tay người thường có thể đưa ra nhiều ký hiệu khác tương đương nhiều ngôn ngữ đa thể loại người đối phương muốn nói gì nhưng trên dự án nghiên cứu thu thập tượng trưng này vì số lượng trường hợp các cử chỉ tay rất lớn, do đó, ta chỉ đưa ra 24 trường hợp làm ví dụ tương đương 24 chữ cái như sau:



Trong dự án này, vì nhiều lý do nên ta không thể đi thu thập cử chỉ tay người trên thực tế của từng người. Do đó, để đảm bảo đủ dữ liệu cho quá trình train cho kết quả chính xác cao, ta đã dựa trên dữ liệu từ kaggle với người dùng dữ liệu đó khá lớn và tin tưởng để huấn luyện cho bài toán này đặt ra. Dữ liệu là dưới dạng toạn độ điểm cử chỉ đã được trích xuất ra.

Ở đây là quá trình thu thập dữ liệu, tiếp theo chúng ta sẽ đến với việc viết chương trình cho hệ thống ở phần tiếp theo.

III/ Viết chương trình cho hệ thống:

Ở phần nay, chúng ta sẽ viết chương trình hoạt động cho quá trình : Huấn luyện và Kiểm tra, Chương trình tạo thuật toán dự đoán hình thực tế từ .Trong đó:

STT	Tên chương trình	Mục đích	Ngôn ngữ
1	Chương trình huấn luyện (Train) AI_LEMINHTRI_19146038 .ipynb	Huấn luyện dữ liệu đã thu thập	Python
2	Chương trình tạo thuật toán dự đoán hình thực tế từ camera và thời gian thực tế Testcamera. ipynb	Kiểm tra kết quả đã huấn luyện qua camera thực tế.	Python

1. Chương trình huấn luyện (Train):

Đây là chương trình dùng để huấn luyện dữ liệu đã thu thập ở bước trên, nhằm mục đích , xuất ra Model có giá trị Loss và Accuracy phù hợp nhất cho dự án này.

a. Tải các thư viện cần thiết:

Để sử dụng các thuật toán huấn luyện, ta cần tải một số thư viện bằng cú pháp như sau:

- Pip install opencv-python:
- Pip install keras:
- Pip install tensorflow==2.5.0:
- Pip install keyboard:
- Pip install tensorflowjs:
- Pip install spellchecker
- Pip install time:
- Pip install textblob:

Sau đó, chúng ta sẽ import các thư viện này vào chương trình:

```
import cv2
import tensorflow as tf
import keras
from keras.models import load_model
import pickle
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
from tensorflow.keras.utils import to_categorical
import numpy as np
import random
import keyboard
from time import sleep
from textblob import TextBlob
from spellchecker import SpellChecker
```

b. Định nghĩa tham số:

Tham số ở đây là số trường hợp nhận dạng trong dự án này, ở đây là 24. Mỗi class tương đương 24 bảng chữ cái.

Vd:

dic1={0:"A",1:"B",2:"C",3:"D",4:"E",5:"F",6:"G",7:"H",8:"I",10:"K",11:"L",12:"M",13:"N",14:"O",15:"P",16:"Q",17:"R",18:"S",19:"T",20:"U",21:"V",22:"W",23:"X",24:"Y"}

dic2={0:"A",1:"E",2:"M",3:"S"}

c. Tạo model:

Ở đây, ta viết một chương trình con để phục vụ cho việc tạo model như sau:

```
# tạo mô hình để huấn luyện sử dụng thuật toán CNN để huấn luyện mô hình
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Conv2D(64, kernel_size=(3,3), activation = 'relu', input_shape=(28, 28 ,1) ))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(128, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(256, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.20))

model.add(Dense(num_classes, activation = 'softmax'))
```

Sau khi chạy chương trình train:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 256)         295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 24)                3096
=================================================================
Total params: 405,656
Trainable params: 405,656
Non-trainable params: 0
```

d. Chỉnh sửa lại dữ liệu đã thu thập để chuẩn bị cho việc huấn luyện:

```
# tạo nhãn huấn luyện
labels = train['label'].values
```

```
df=data[(data['label']==0) | (data['label']==4) | (data['label']==12) | (data['label']==18)]
```

```
#xem các nhãn tổng cộng có 24 nhãn tương đương bảng chữ cái 24 chữ
unique_val = np.array(labels)
np.unique(unique_val)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24], dtype=int64)
```

```
# bỏ nhãn để dữ liệu về dạng chuỗi số để dễ dàng huấn luyện
train.drop('label', axis = 1, inplace = True)
```

```
# Trích xuất những hình ảnh dữ liệu csv từ mỗi hàng của chúng tôi trong lưu trữ nhớ nó trong một hàng của 784 cột
images = train.values
images = np.array([np.reshape(i, (28, 28)) for i in images])
images = np.array([i.flatten() for i in images])
```

e. Huấn luyện dữ liệu:

```
# Biên dịch mô hình
model.compile(loss = 'categorical_crossentropy',
              optimizer= Adam(),
              metrics=['accuracy'])
```

```
# mã hóa dữ liệu các nhãn được tách ra trước đó
from sklearn.preprocessing import LabelBinarizer
label_binrizer = LabelBinarizer()
labels = label_binrizer.fit_transform(labels)
```

```
# xem các nhãn được mã hóa
labels
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0]])
```

```
# Chia dữ liệu của chúng tôi thành x_train, x_test, y_train và y_test
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size = 0.3, random_state = 101)
```

```
# sử dụng thư viện tensorflow để xác định kích thước của từng tập dữ liệu huấn luyện
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout

batch_size = 200
num_classes = 24
epochs = 10
```

```
# chia tỉ lệ hình ảnh
x_train = x_train / 255
x_test = x_test / 255
```

```
batch_size = 200
num_classes = 24
epochs = 10
```

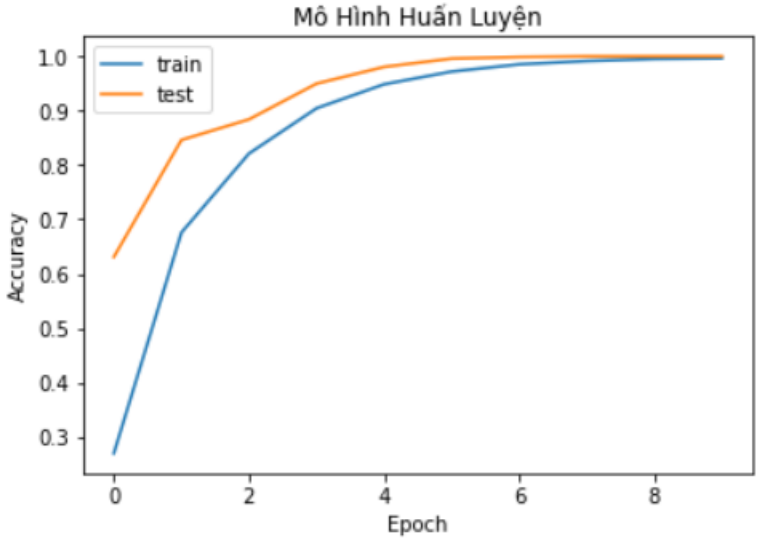
```
# Huấn Luyện Mô hình''
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=epochs, batch_size=batch_size)
```

f. Lưu model và các kết quả của quá trình huấn luyện:

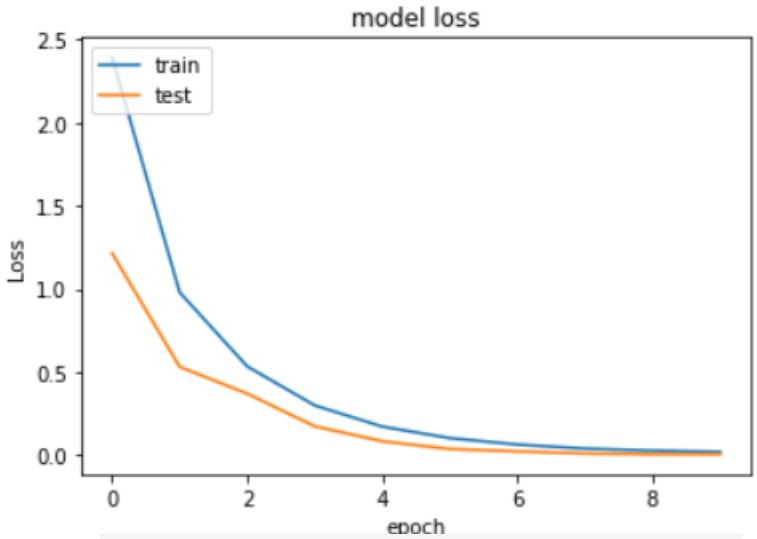
• *Lưu model:*

```
# Lưu mô hình được huấn luyện
model.save("CNN.h5")
print("Model Saved")
```

• *Các thông số kết quả của quá trình huấn luyện:*



Sau khi huấn luyện dữ liệu, ta có được các thông số để đánh giá quá trình huấn luyện như: Time train, Time train / Steps, Loss, Accuaracy. Dưới đây là các biểu đồ về các giá trị trên với Epoch = 10 và Batch Size =200. Ở đây ta cần chọn giá trị Epoch và Batch Size phù hợp để đạt được giá trị Accuaracy tốt nhất và giá trị Loss nhỏ nhất.




```
# Độ chính xác sau khi huấn luyện
from sklearn.metrics import accuracy_score

accuracy_score(test_labels, y_pred.round())
```

```
0.838957055214724
```

Đây là kết quả độ chính xác sau quá trình huấn luyện.

IV/ Tăng độ chính xác bằng cách pre-train

1/ Huấn luyện mô hình

Để tăng độ chính xác mô hình ta dung thuật toán để pre-train mô hình lần nữa

Đầu tiền phải tạo lại dữ lữ liệu và sử dụng lại model khi nãy đã huấn luyện.

```
import keras
model2=keras.models.load_model("CNN.h5")
dic={0:0,1:4,2:12,3:18}
def model_pred(data,new_model):
    img = np.array(data)/255
    img = img.reshape(1,28,28,1)
    result = new_model.predict(img)
    result=np.argmax(result)
    if result>=9:
         result=result+1
    return result
```

```
def model2_pred(data,new_model):
    img = np.array(data)/255
    img = img.reshape(1,28,28,1)
    result = new_model.predict(img)
    result=np.argmax(result)
    if result>=9:
        result=result+1
    return result
```

Ở đây dùng cách pre-train dữ liệu cũ nhưng phần hình ảnh data vẫn phải khai báo lại để tạo lại dữ liệu mới có độ chính xác cao hơn.

```
lst=[]
for x in x_test:
    prediction=model2_pred(x,model2)
    if prediction ==0 or prediction==4 or prediction ==12 or prediction ==18 :
#         print(prediction)
        prediction=model_pred(x,model)
    lst.append(prediction)
```

Sau đó huấn luyện chương trình một lần nữa và in ra kết quả sau quá trình pre-train.

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, lst)
```

```
0.9060234244283324
```

Như kết quả pre-train đã cho thấy kết quả đã tăng khá nhiều so với trước đó từ đó ta có thể đưa cả hai dữ liệu vào chương trình dự đoán chính tạo độ chính xác cao hơn nhiều lần

Để có cái nhìn khả quan hơn ta in thử kết quả :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	331
1	1.00	0.98	0.99	432
2	0.94	1.00	0.97	310
3	0.98	1.00	0.99	245
4	0.96	0.97	0.96	498
5	1.00	1.00	1.00	247
6	0.93	0.75	0.83	348
7	0.94	0.95	0.95	436
8	0.80	0.87	0.83	288
10	0.93	0.95	0.94	331
11	0.79	1.00	0.88	209
12	0.82	0.83	0.83	394
13	0.94	0.85	0.89	291
14	1.00	0.83	0.91	246
15	1.00	1.00	1.00	347
16	0.81	0.90	0.85	164
17	0.58	0.95	0.72	144
18	0.83	1.00	0.91	246
19	0.70	0.68	0.69	248
20	0.98	0.91	0.94	266
21	1.00	0.58	0.74	346
22	0.82	1.00	0.90	206
23	0.91	0.91	0.91	267
24	0.93	0.88	0.91	332
accuracy			0.91	7172
macro avg	0.90	0.91	0.90	7172
weighted avg	0.92	0.91	0.91	7172

```
y_test[10:20]
```

```
10     8
11     8
12    21
13    12
14     7
15     4
16    22
17     0
18     7
19     7
Name: label, dtype: int64
```

Khá tốt so với trước pre-train sau đó ta lưu lại model thành một tên khác để sau khi thử nghiệm trên camera real-time có độ chính xác so sánh từ 2 model đồ tốt hơn.

```
model.save("second_model.h5")
```

2. Chương trình tạo thuật toán dự đoán hình thực tế từ camera và thời gian thực tế

Thêm dữ liệu thư viện vào để kiểm tra ký hiệu ngôn ngữ tay qua camera thực tế

```
import cv2
import tensorflow as tf
import keras
from keras.models import load_model
import pickle
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
from tensorflow.keras.utils import to_categorical
import numpy as np
import random
import keyboard
from time import sleep
from textblob import TextBlob
from spellchecker import SpellChecker
```

Chuyển hóa dữ liệu và đưa dữ liệu hình ảnh so sánh với hai model đã huấn luyện trước đó

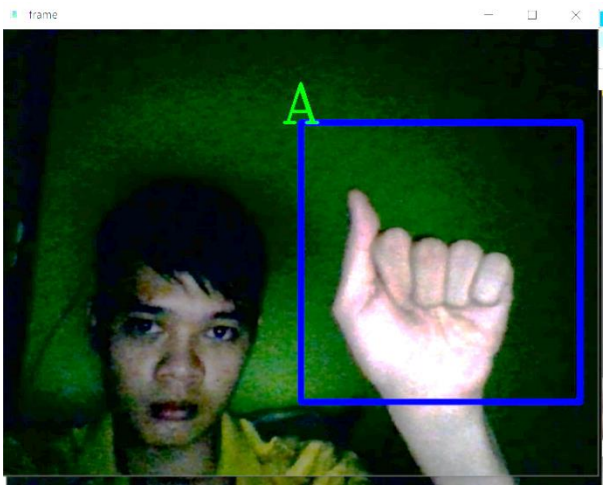
```
model1 = load_model('CNN.h5')
model2=load_model('second_model.h5')
def load_image1(filename,new_model1):
    img = load_img(filename,target_size = (28,28),color_mode = "grayscale")
    img = np.array(img)/255
    img = np.expand_dims(img,axis = 0)
    img = img.reshape(1,28,28,1)
    result = new_model1.predict(img)
    result=np.argmax(result)
    if result>=9:
        result=result+1
    return dic1[result],result

def load_image2(filename,new_model2):
    img = load_img(filename,target_size = (28,28),color_mode = "grayscale")
    img = np.array(img)/255
    img = np.expand_dims(img,axis = 0)
    img = img.reshape(1,28,28,1)
    result = new_model2.predict(img)
    result=np.argmax(result)
    return dic2[result],result
```

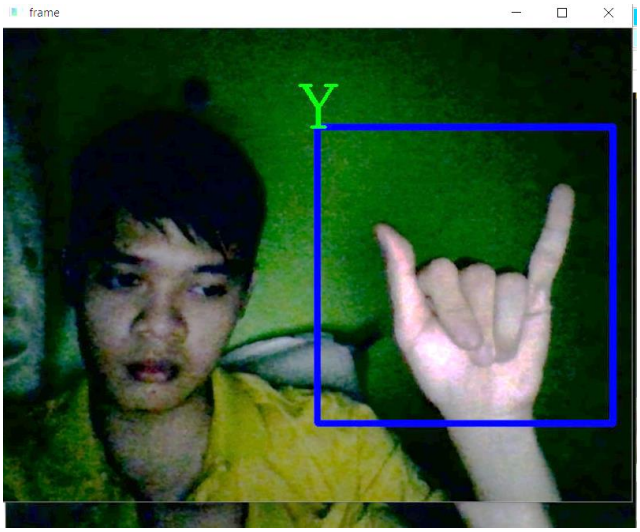
```
vid = cv2.VideoCapture(0)
spell = SpellChecker()
font = cv2.FONT_HERSHEY_SIMPLEX
org = (80, 40)
fontScale = 1.0
color = (255, 255, 0)
thickness = 3
word=""

while True:
    ret,frame=vid.read()
    cv2.rectangle(frame, (20,80), (190,250), (0,200,200),2)

    img1=frame[80:250,20:190]
    cv2.imwrite("test.jpg", img1)
    pred1,prediction=load_image1('test.jpg',model1)
    if prediction ==0 or prediction==4 or prediction ==12 or prediction ==18 :
        pred1,prediction=load_image2('test.jpg',model2)

    try:
        cv2.putText(frame, "HAND PUT IN", (30,32), font,
                            1, (0,0,255), 2, cv2.LINE_AA)
        cv2.putText(frame, pred1, (95,70), font,
                                fontScale, (0, 0,255), thickness, cv2.LINE_AA)
        cv2.imshow("frame",frame)
    except Exception as e:

        cv2.imshow("frame",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

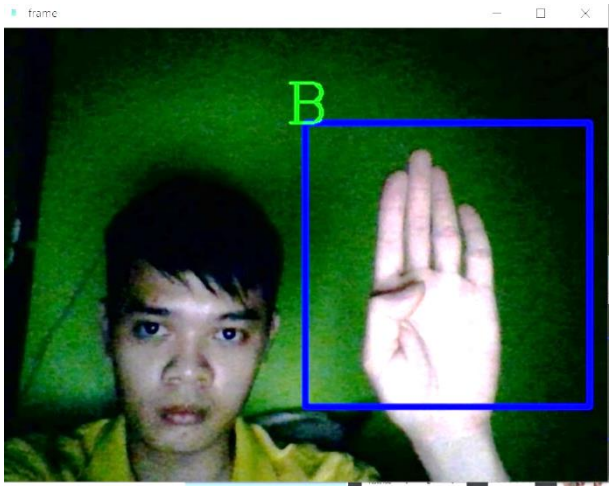
Ở đây ta đưa hình ảnh để kiểm tra từ camera thực tế, lưu ý chỗ khai báo Videocapture có thể khai báo 0 hoặc 1, 0 là lấy camera từ máy tính và 1 là từ cổng usb ngoài. Để tạo khung hình camera sinh động hơn ta tạo thêm các font chữ định dạng và kích thước, sau đó ta sẽ thử đưa hình ảnh bàn tay vào khung giới hạn để nhận diện bàn tay (ret,frame=vid.read()
cv2.rectangle(frame, (20,80), (190,250), (0,200,200),2))

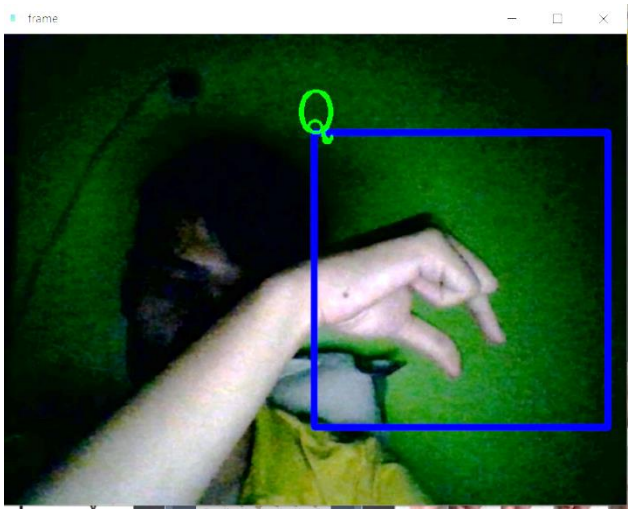
V/ Kiểm tra độ chính xác mô hình qua camera

Hình ảnh sẽ được chụp và so sánh liên tục với 2 model đã được huấn luyện sẵn và được dự đoán chính xác như sau đây là hình ảnh được cắt ra từ việc dự đoán thực tế camera một số cử chỉ tay.

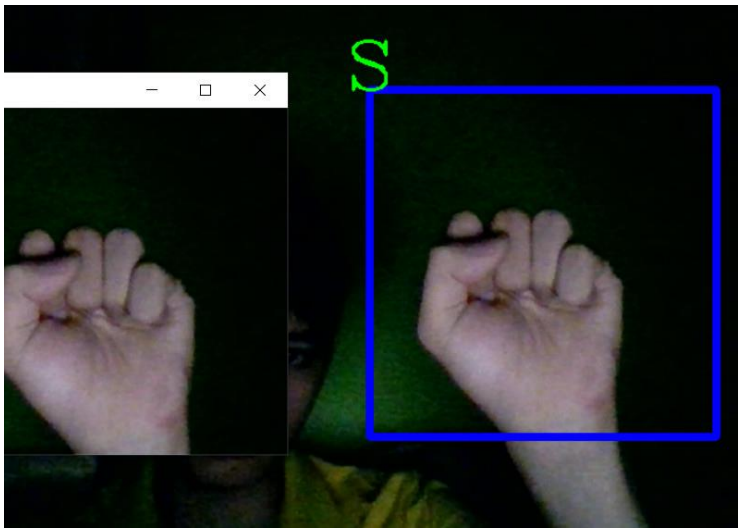








Ở đây để có thể bao quát được hình ảnh nhận diện đặt bàn tay đặt trong khung hình để hạn chế độ nhiễu và sai số phép toán đưa ra, thêm một cửa sổ từ khung nhận diện bàn tay vào để cải thiện cái nhìn từ bàn tay tốt hơn.



D/ Kết Luận:

1.Ưu điểm:

- Kết quả nhận dạng cử chỉ tay có độ chính xác cao: 90 % .
- Có thể trực tiếp nhận diện cử chỉ hình ảnh một cách nhanh và chính xác

2. Nhược điểm:

-Do sử dụng huấn luyện dựa trên tọa độ điểm ảnh dữ liệu thu thập có sẵn nên khi ứng dụng real-time gây hiện tượng nhiễu khá khó cải thiện tốt

3. Hướng phát triển:

- Tự tạo dữ liệu ảnh tọa độ cử chỉ bàn tay từ nhiều dữ liệu bàn tay khác nhau, từ nhiều môi trường khác nhau để hạn chế tối đa quá trình nhiễu khi huấn luyện và khi chạy real-time.

- Chọn lựa giá trị Epoch và Batch Size thích hợp để đạt được giá trị Loss và Accuarcy tốt nhất cho Model.
- Thêm tính năng tự phát âm thanh cho cử thay cho người không nói được.

4. Tài liệu tham khảo:

1. https://www.kaggle.com/datasets/datamunge/sign-language-mnist

2. Convolutional Neural Networks – Nguyen Truong Thinh

5. Ghi chú:

Vì chương trình khá dài nên ta không thể đưa hết vào trong bài báo cáo này. Em xin gửi chương trình trong file đính kèm.
Link youtube về hoạt động của sản phẩm:

1. https://www.youtube.com/watch?v=OYbvbrr2m6E

2. https://github.com/uyminhtri2702/AI_T.THINH

- Các thuật ngữ được sử dụng trong báo cáo:

• Model: Là file có được sau quá trình train, dựa vào đó mà ta xử lý các yêu cầu.
• Epoch:Có thể hiểu đơn giản là số lần mà ta nạp dữ liệu vào mạng Nerual Network
• Batch Size: Là số lượng mẫu dữ liệu trong một batch.
• Loss: Là độ sai số giữa giá trị hiện thực và giá trị mà Model dự đoán. Giá trị Loss càng nhỏ thì Model càng dự đoán chính xác.
• Accuracy:Là độ chính xác giữa giá trị hiện thực và giá trị dự đoán của Model.
• Class:Có thể hiểu đơn giản là Lớp, là các thông tin mà ta cần phân biệt và nhận dạng.

Lời cảm ơn: Em xin chân thành cảm ơn thầy Nguyễn Trường Thịnh đã hỗ trợ em thực hiện dự án này,

----------Hết----------