ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

Machine Learning

Học kỳ II (2019-2020)

NHẬN DIỆN CỬ CHỈ NGÓN TAY ÁP DỤNG VÀO TRÒ CHƠI RẮN SĂN MỒI

Sinh viên :	Nguyễn Lâm Quỳnh
MSSV	18521326
Giảng viên:	Lê Đình Duy
	Phạm Nguyễn Trường An

Thành phố Hồ Chí Minh, tháng 7 năm 2020

MỤC LỤC

I. Mở đầu

Giới thiệu đề tài

Rắn săn mồi là trò chơi cổ điển với lối chơi đơn giản bằng cách thực hiện các mũi tên trái, phải, lên, xuống từ bàn phím để di chuyển con rắn sao cho đi đến chấm thức ăn. Trong đề tài này, tôi đề cập đến một cách giải quyết mới – áp dụng Machine Learning vào trong trò chơi, thay vì sử dụng các mũi tên trái, phải, trên, xuống tôi sử dụng một model để dự đoán hình ảnh hướng ngón tay từ camera và con rắn sẽ di chuyển theo hướng đó.

Mục tiêu

Mục tiêu của đề tài là từ phân tích dữ liệu hiện có đưa ra mô hình dự đoán tốt nhất, có

tính khái quát đối với bàn tay của nhiều người. Chương trình sẽ áp dụng mô hình được lựa chọn để đưa vào trò chơi nhằm đưa ra hướng đi của rắn.

Cấu trúc bài báo cáo

Bài báo cáo gồm có 4 phần:

- Phần 1: Giới thiệu đề tài và mục tiêu nghiên cứu
- Phần 2: Quy trình làm việc của một ứng dụng Machine Learning gồm 7 bước:
    - Thu thập dữ liệu
    - Tiền xử lí dữ liệu và rút trích đặc trưng
    - Chọn model
    - Huấn luyện
    - Đánh giá
    - Tinh chỉnh tham số
    - Dự đoán
- Phần 3: Kết luận

– Phần 4: Các tài liệu tham khảo

II. Quy trình xây dựng mô hình

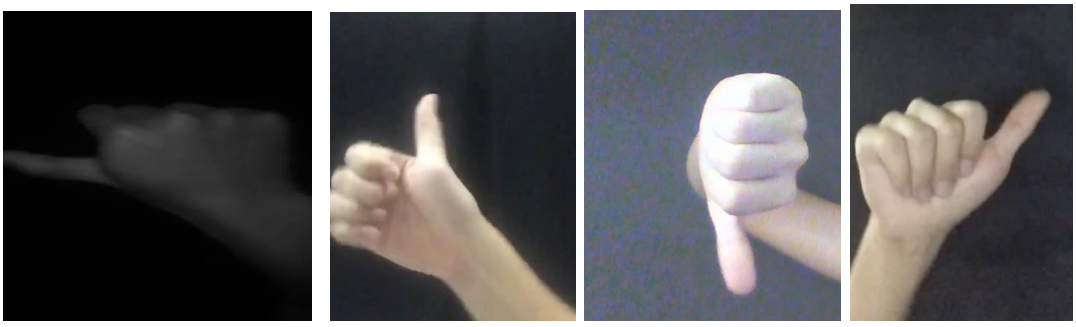
Ở bài toán này, chúng tôi xây dựng mô hình dự đoán hướng ngón tay. Với input là bàn tay phải có ngón tay chỉ hướng là ngón cái chỉ các hướng trái, phải , trên, xuống. Output là hướng đi của con rắn được dự đoán dựa trên mô hình. Chương trình sẽ sử dụng webcam để nhận input và con rắn trong cửa sổ trò chơi sẽ di chuyển theo hướng dự đoán.

1. Thu thập dữ liệu

Dữ liệu được thu thập theo hai cách:

– Thực tế: Dữ liệu được lấy từ bàn tay của bốn người. Thu thập bằng cách quay một video bằng camera trước của điện thoại hoặc máy tính. Background là màu đen, với bàn tay phải nằm trọn trong khung hình, ngón cái chỉ các 4 hướng trái, phải, trên, dưới thay đổi.

Có được video, tôi sẽ dùng ffmpeg để tách frame từ video với số frame cố định là 50 frame/s cố định. Sau đó loại bỏ các hình ảnh không đạt chất lượng. Cuối cùng thu lại được tổng cộng 14800 ảnh.



Mô tả data thực tế

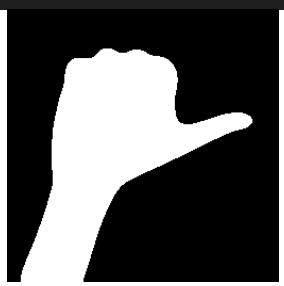
– Internet: Dữ liệu được thu thập trên kaggle với 4000 ảnh

  - Lấy foler thumbs ở link (https://www.kaggle.com/dylanmendonca/hand_gestures?) ta thu được 1000 ảnh với label tương ứng trong bài toán là left.

left

Sau đó xoay ảnh với trục Oy ta sẽ thu được label là right với số ảnh cũng là 1000 ảnh.

```
img_flip_lr = cv2.flip(src, 1)
```



right

- Lấy hai folder thumbsup và thumbsdown trên https://www.kaggle.com/sarjit07/hand-gesture-recog-dataset tương ứng với hai folder up và down tương ứng

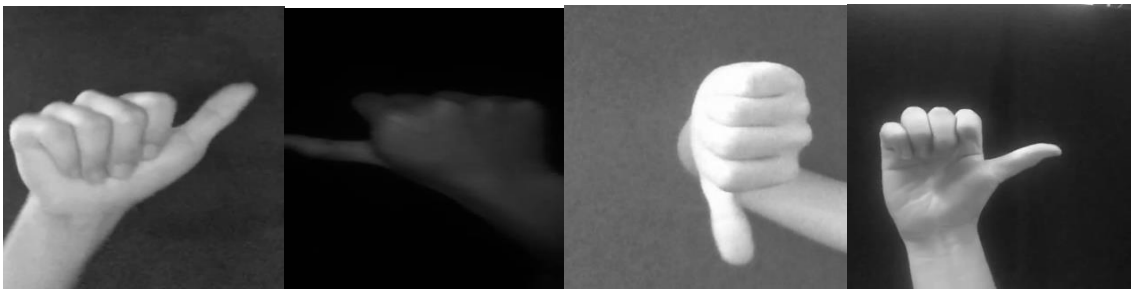
Up and down

Tổng cộng số lượng data gồm 18800 ảnh.

2. Tiền xử lí dữ liệu và rút trích đặc trưng

2.1. Tiền xử lí dữ liệu

- Các hình ảnh được resize về kích thước 320x320 và gán nhãn cho ảnh.
- Sau đó chuyển về ảnh xám.



Chuyển về ảnh xám

- Chuyển về ảnh nhị phân với nền màu đen và hình ảnh bàn tay màu trắng. Muốn chuyển về phải xác định threshold của mỗi người sau đó áp dụng threshold đó vào tất cả ảnh mà người đó quay.

Chuyển về ảnh nhị phân

2.2. Rút trích đặc trưng

Các đặc trưng cần chú ý ở đây là hình dáng bàn tay và hướng ngón tay. Để thực hiện điều đó tôi sử dụng HOG để rút trích đặc trưng. Trước khi áp dụng nó, tôi tìm contours của bàn tay. Điều này cv2 hỗ trợ rất tốt

```
  contours, hierarchy = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APP
ROX_SIMPLE)
```

sau đó vẽ hình chữ nhật đứng bao quanh contours đó, resize hình chữ nhật thành kích thước 150x150. Áp dụng HOG vào hình chữ nhật đứng đó với orientations=9, pixel per cell=(15,15) và block per cell(2,2). Số lượng feature rút trích được là 2916.

Labels của ảnh tôi sử dụng encode để mã hóa nó với ban đầu là down,left,right, up thành 0,1,2,3.

Sau đó tôi lưu vector features và vector labels vào file h5 với kích thước của

vector features này là (18800,2196) và vector labels là (18800,1).

3. Chọn model

Các model dùng để kiểm thử tôi sử dụng trong đề tài này là Logistic Regression, Linear Dicriminant Analysis, K Neareast Neighbor, Decission Tree, Random Forest Tree, Gausian NB của thư viện tensorflow.

4. Huấn luyện

Trước khi huấn luyện tôi đã chia tách vector features và vector labels thành 2 tập là traindata và testdata với tỉ lệ là 80/20.

Đưa tập traindata gồm train labels và train features vào từng model để huấn luyện. Kết quả thu được 6 mô hình dự đoán.

5. Đánh giá

Sử dụng testdata để đánh giá mô hình có hoạt động tốt không. Ở đây, các chỉ số tôi dùng để đánh giá độ chính xác của mô hình dự đoán là recall, precision và F1.

```
====================LogisticRegression=====================

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

Đánh giá model Logistic Regression

```
====================LinearDiscriminantAnalysis=====================

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

Đánh giá model Linear Discriminant Analysis

```
====================KNeighborsClassifier=====================

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

Đánh giá model K Neareast Neighbor

```
====================DecisionTreeClassifier=====================

accuracy =  0.9936170212765958
Confusion matrix:
[[938   0   2   0]
 [  5 932   0   3]
 [  2   3 946   3]
 [  2   1   3 920]]
precision: [0.9904963  0.9957265  0.99474238 0.99352052]
recall: [0.99787234 0.99148936 0.99161426 0.99352052]
fscore: [0.99417064 0.99360341 0.99317585 0.99352052]
support: [940 940 954 926]
```

Đánh giá model Decision Tree

```
====================RandomForestClassifier=====================

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

Đánh giá model Random Foest Tree

```
====================GaussianNB=====================

accuracy =  0.9960106382978723
Confusion matrix:
[[940   0   0   0]
 [  3 934   0   3]
 [  1   2 946   5]
 [  0   1   0 925]]
precision: [0.99576271 0.99679829 1.         0.99142551]
recall   : [1.         0.99361702 0.99161426 0.99892009]
fscore   : [0.99787686 0.99520511 0.99578947 0.99515869]
support  : [940 940 954 926]
```

Đánh giá model Gaussian NB

Trong các model thử nghiệm Linear Discriminant Analysis cho kết quả tốt nhất trên cả tập train và tập test.

6. Tinh chỉnh tham số

Ở đây tôi sử dụng model của thư viện tensorflow nên các tham số đã được fit, nên tôi không thực hiện bước này.

7. Dự đoán

Ở đây tôi đưa model vào game rắn săn mồi để thực hiện dự đoán real-time, code game sử dụng được thu thập trên internet.Trong quá trình chạy sẽ có một khung hình camera để nhận hình ảnh bàn tay đưa vào. Sau đó xử lí ảnh đưa vào đê rút trích đặc trưng và dự đoán, kết quả của dự đoán sẽ là hướng đi của rắn hiện tại.

7.1. Môi trường:

Scikit-image 0.14.2

Python3

Numpy 1.15

Chạy chương trình:

python3 run_demo.py

7.2. Quy trình thực hiện:

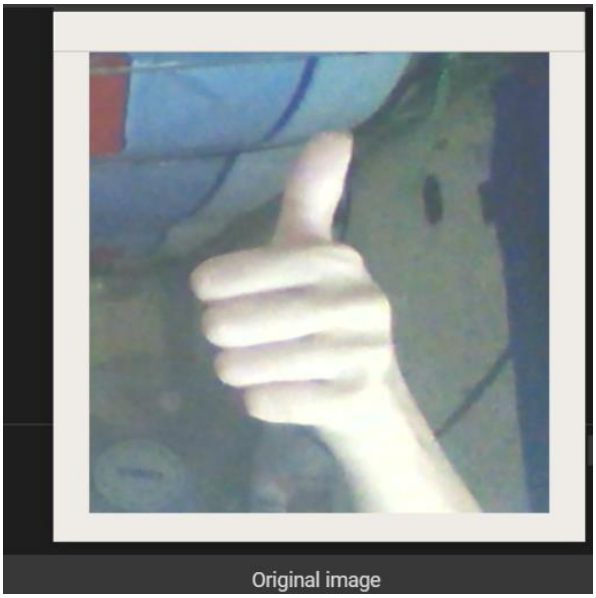
– Lấy ảnh gốc:

Quay video từ màn hình và chọn vùng quan tâm

```
frame=frame[100:400,400:700]
```

Code chọn vùng quan tâm

Đưa bàn tay vào nằm trọn trong khung hình



– Loại bỏ nền:

Sử dụng BackgroundSubtractor của cv2 để loại bỏ nền, chỉ lấy các đối tượng chuyển động. Ở đây, đối tượng chuyển động là bàn tay.

```
bgModel = cv2.BackgroundSubtractorMOG2(0, bgSubThreshold)
```

BackgroundSubtractor

Áp dụng vào từng frame

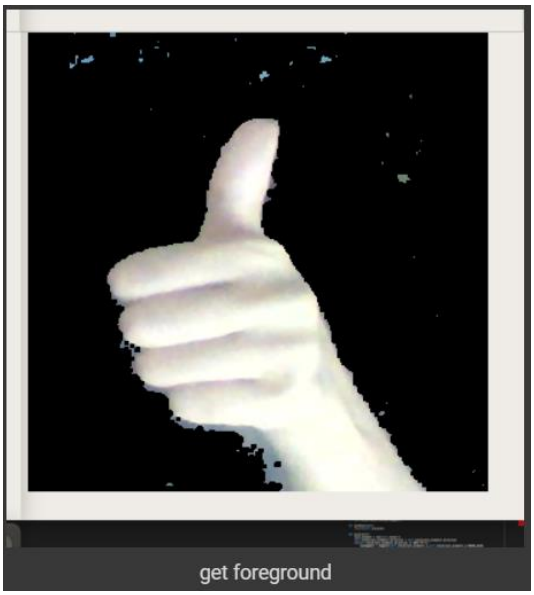
```
fgmask = bgModel.apply(frame,learningRate=learningRate)
```

Trừ nền cho frame

Lấy foreground(hand) image

```
res = cv2.bitwise_and(frame, frame, mask=fgmask)
```

Foeground



Ảnh có được khi trừ nền

- Gausian Blur& Threshold

  Chuyển về ảnh grayscale

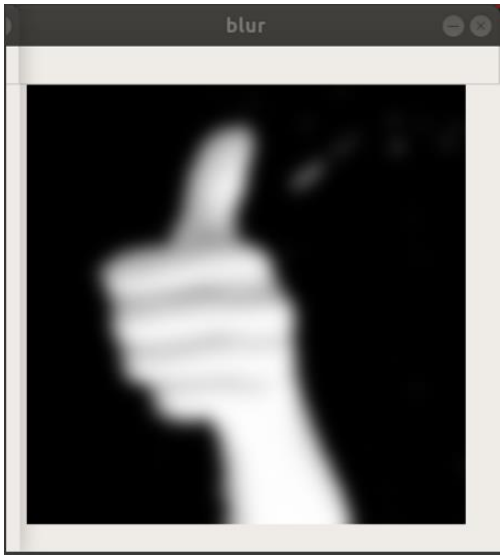
```
gray = cv2.cvtColor(fram, cv2.COLOR_BGR2GRAY)
```

Covert gray scale

Dùng Gausian Blur

```
blur = cv2.GaussianBlur(gray, (blurValue, blurValue), 0)
```

Gausian Blur

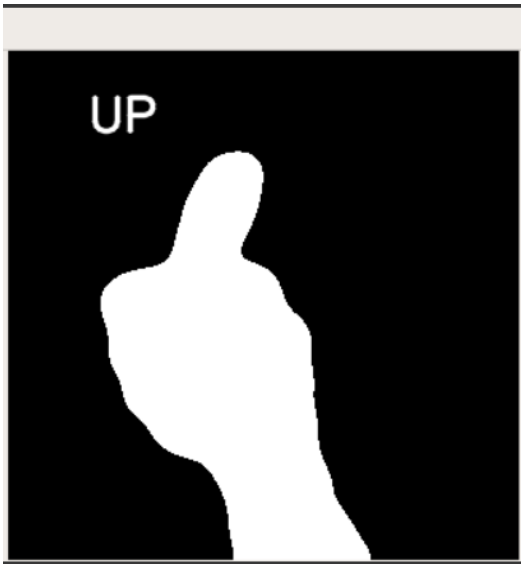


Blur image

Chuyển về ảnh nhị phân:

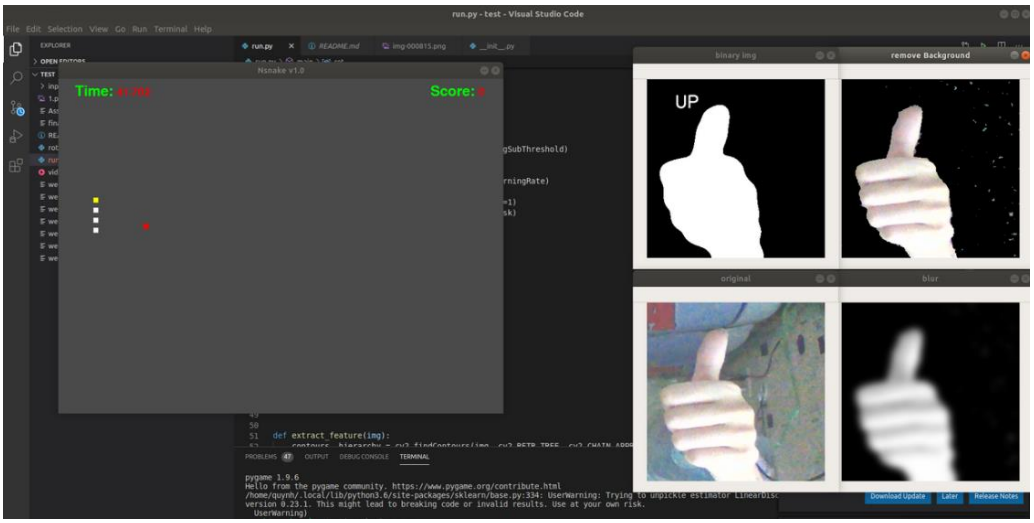
```
ret, thresh = cv2.threshold(blur,threshold, 255, cv2.THRESH_BINARY)
```

Covert binary image



Binary image

– Dự đoán hướng của ngón tay(hướng đi của rắn)



Ảnh khi thực thi chương trình

III. Kết luận

Mô hình có khả năng áp dụng áp dụng cho bài toán nhận diện hướng ngón tay hiệu quả. Tuy nhiên trong lúc thực hiện đề tài cũng có những hạn chế nhất định. Việc chạy real-time chỉ áp dụng được khi background không chuyển động. Vậy nên với những background có sự di chuyển thì model sẽ không thực hiện tốt. Hướng giải quyết có thể thực hiện là sử dụng YOLO để detect bàn tay trong khung hình. Một vấn đề nữa xảy ra là model dự đoán tốt hướng của bàn tay, nhưng trong lúc di chuyển từ hướng cũ thành hướng mới, khoảng cách giữa chúng model dự đoán không thực sự tốt. Cái này có thể khắc phục bằng cách sử dụng các phương pháp rút trích đặc trưng khác như sử dụng CNN.

IV. Nguồn tham khảo

https://www.tensorflow.org/

https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html

https://scikit-learn.org/stable/modules/model_evaluation.html#:~:text=2.-,Classification%20metrics,values%2C%20or%20binary%20decisions%20values.

https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python

Source code game snake: https://gist.github.com/someoneigna/5022021

https://docs.opencv.org/3.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html