TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA KHOA HỌC MÁY TÍNH

MÁY HỌC

Đề Tài: **NHẬN DIỆN MỘT NGƯỜI ĐIỀU KHIỂN XE MÁY CÓ ĐỘI MŨ BẢO HIỂM KHÔNG**

GVHD: Lê Đình Duy

Phạm Nguyễn Trường An

Lớp: CS114.K21

Nhóm: Vũ Minh Luân - 18521067

Hoàng Xuân Thắng - 18521391

Hứa Văn Sơn - 18521344

TP. HỒ CHÍ MINH, NGÀY 12 THÁNG 08 NĂM 2020

MỤC LỤC

Contents

I. Mở đầu

1.1 Giới thiệu chung

Theo số liệu thống kê của Ủy ban An toàn giao thông Quốc gia thì mỗi năm ở nước ta có hơn 8000 người chết vì tai nạn giao thông. Trong đó, số người đi môtô xe máy gây tai nạn giao thông chiếm 60%, số người đi xe máy là nạn nhân tai nạn giao thông là 85%. Một thống kê khác cho thấy đội mũ bảo hiểm giúp giảm 40% nguy cơ thương vong khi bị tai nạn giao thông. Do đó cần xây dựng một ứng dụng giám sát người tham gia giao thông có đội mũ hay không để có các biện pháp xử phạt và ngăn chặn việc tham gia giao thông không đội mũ bảo hiểm, giúp giảm được một phần thiệt hại do tai nạn giao thông gây ra.

1.2 Mục tiêu

Sau cùng có thể tìm ra một cách rút trích đặc trưng từ ảnh hiệu quả nhất, lựa chọn model phù hợp, tinh chỉnh parameter sao cho Accuracy đạt được là cao nhất. Đồng thời có thể áp dụng cho các tình huống thực tế cụ thể.

1.3 Mô tả bài toán

Input: bức ảnh về 1 người điều khiển xe máy tham gia giao thông.

Output: 0 nếu người đó không đội mũ , 1 nếu người đó có đội mũ.

II. Hướng giải quyết

2.1 Mô tả dữ liệu

Data được thu thập bằng cách quay video người tham gia giao thông trên các con đường với tầm nhìn từ trên tầng 2 của một quán cà phê để đảm bảo được góc nghiêng cao phù hợp với camera an ninh giao thông.

Sau đó dùng app Frame Grabber để trích xuất thành ảnh theo từng frame. Ảnh sau khi được trích xuất từ video thu được khoảng 910 tấm ảnh. Trong đó 357 ảnh không đội mũ, còn lại có đội mũ.

2.2 Tiền xử lý dữ liệu

a) Crop, resize ảnh

Ảnh sẽ được crop bớt phần rìa để tập trung vào chủ thể, với mục đích tối ưu cho bước rút trích đặc trưng. Đồng thời ảnh cũng sẽ resize về 200x200 để giảm kích thước, giảm thời gian đọc ảnh.

b) Làm mở, tìm cạnh

Dùng kỹ thuật Gaussian Blur để làm mờ ảnh, sau đó dùng Canny Edge để tìm cạnh cho ảnh.

```
#làm mờ ảnh
im2 = cv2.GaussianBlur(im1, (5, 5), 0)
#sử dụng canny edge để tìm ra các cạnh có trong ảnh
a = cv2.Canny(im2, 150, 200)
```

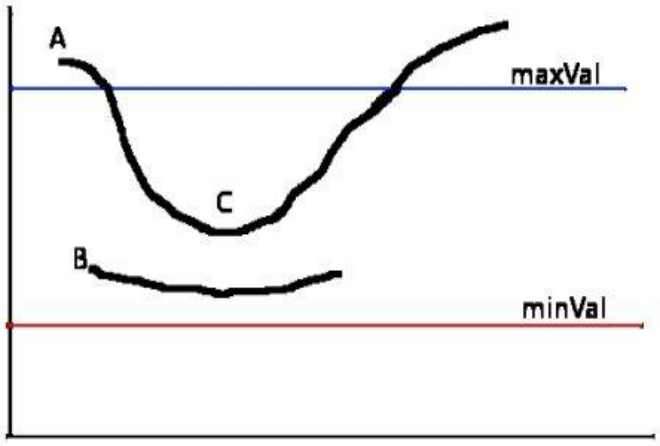
Giải thuật phát hiện cạnh Canny gồm 4 bước chính sau:

- **Giảm nhiễu**: Làm mờ ảnh, giảm nhiễu dùng bộ lọc Gaussian kích thước 5x5.Tất nhiên là có thể thay đổi kích thước bộ lọc nhưng theo em tham khảo thì kích thước 5x5 thường hoạt động tốt cho giải thuật Canny.
- **Tính Gradient và hướng gradient**: ta dùng bộ lọc Sobel X và Sobel Y (3x3) để tính được ảnh đạo hàm Gx và Gy. Sau đó, ta tiếp tục tính ảnh Gradient và góc của Gradient theo công thức. đạo hàm Gx và Gy là ma trận như kích thước ảnh đầu vào là 200x200, thì kết quả tính ảnh đạo hàm Edge Gradient cũng là một ma trận (200x200), mỗi pixel trên ma trận này thể hiện độ lớn của biến đổi mức sáng ở vị trí tương ứng trên ảnh gốc.

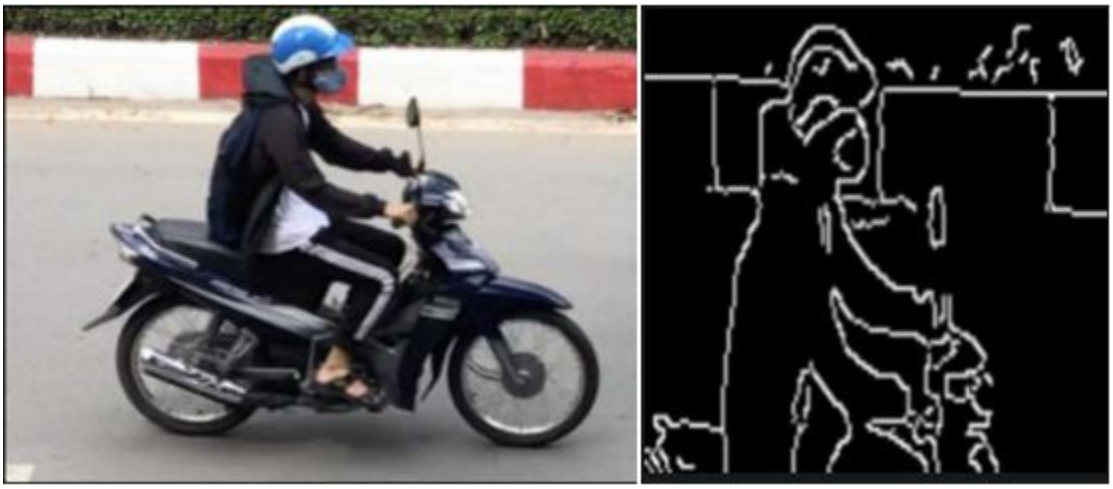
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- **Lọc ngưỡng**: ta sẽ xét các giá trị gradient . Nếu giá trị gradient vượt ngưỡng **max_val** thì pixel đó **chắc chắn là cạnh**. Các pixel có độ lớn gradient nhỏ hơn ngưỡng **min_val** sẽ bị loại bỏ. Còn các pixel nằm trong khoảng 2 ngưỡng trên sẽ được xem xét rằng nó có nằm liên kề với những pixel được cho là "chắc chắn là cạnh" hay không. Nếu liền kề thì giữ, còn không liền kề bất cứ pixel cạnh nào thì ta loại. Ảnh minh họa về ngưỡng lọc:


A
maxVal
C
B
minVal

Kết quả thu được sau khi áp dụng 2 kỹ thuật trên như sau:



c) Feature Extraction

Ảnh sau khi được xử lí qua canny edge thì sẽ được chia thành các block nhỏ hơn . Ảnh có kích thước đầu vào là 200x200 với block có kích thước là 40x40 thì sẽ có tổng cộng 25 block. ta cũng có thể thay đổi kích thước các block này sao cho phù hợp với từng data khác nhau. sau khi thử nghiệm thì em thấy kích thước block 50x50 cho kết quả tốt nhất.

Từ mỗi block này, tiến hành tính toán giá trị trung bình (mean) và độ lệch chuẩn (std) của các giá trị pixel trong các block đó. Như vậy, từ một ảnh ta có thể rút trích 32 giá trị mean và std để làm feature cho ảnh đó. các giá trị này sẽ được thêm vào một data frame để chuẩn bị cho việc train model.

III. Chọn Model, Training/Predict Data

3.1 Chọn Model

Nhóm chọn Logistic Regression và Random Forest Classifier là 2 model để training tập training data, vì đây là các model phù hợp với bài toán Classification đồng thời đã được các thầy dạy trên lớp cũng như nhóm tìm hiểu đã kỹ nhất.

3.2 Training data

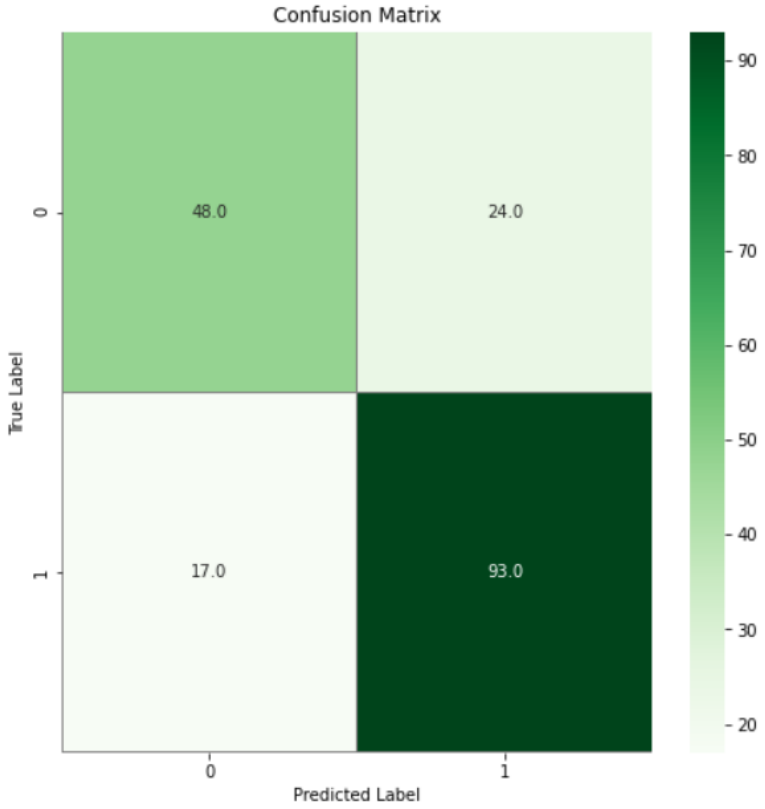
Tập data trước khi đưa vào training sẽ được chia theo tỉ lệ 8-2

3.3 Đánh giá Model

Sử dụng AccuracyScore để kiểm tra độ chính xác của Model dựa trên tập train / valid và gọi thêm Confusion Matrix để xem số ảnh dự đoán đúng/sai.

Kết quả thu được như sau: độ chính xác cao nhất là xấp xỉ 81%, số ảnh dự đoán sai là 41/182 ảnh

```
LogisticRegression : 0.7747252747252747
RandomForestClassifier : 0.8186813186813187
```



3.4 Tinh chỉnh tham số, cải thiện Accuracy

Dùng hàm GridSearchCV để tinh chỉnh tham số cho các model sử dụng ở trên. Tuy nhiên kết quả thu lại không có sự thay đổi đáng kể.

```
# max_depth là chiều sâu của cây
# n_estimators là số nhánh
param1 = {'n_estimators' : [50,60,65,70], 'max_depth' : [10,15,20], 'random_state' : [0]}
model1 = GridSearchCV(RandomForestClassifier(), param1)
model1.fit(X_train, y_train)
print(model1.best_params_)
```

```
{'max_depth': 10, 'n_estimators': 60, 'random_state': 0}
```

```
param2 = {'C' : [0.1, 1.0, 10.0, 100.0]}
model2 = GridSearchCV(LogisticRegression(), param2)
model2.fit(X_train, y_train)
print(model2.best_params_)
```

Kết quả thu được như sau:

```
LogisticRegression :  0.7747252747252747
Random Forest :  0.8241758241758241
```

Như kết quả ở trên, Độ chính xác chỉ tăng khoảng 1%. Chính vì thế nhóm đã tham khảo thêm mô hình của Deep Learning để rút trích đặc trưng, cụ thể ở đây là VGG16

a) Feature Extraction with VGG16

Để cải thiện độ chính xác, nhóm đã dùng mô hình VGG16 để rút trích đặc trưng. Sau đó lại đưa vào Logistic Regression và Random Forest để kiểm tra kết quả.

Trong Keras hiện tại có hỗ trợ 2 pre-trained model của VGG: VGG-16 và VGG 19. Có 2 params chính cần lưu ý là include_top (True / False): có sử dụng các Fully-connected layer hay không và weights ('imagenet' / None): có sử dụng pre-trained weights của ImageNet hay không. Ở đây chúng em chọn là include_top=False vì các lớp Fc layer cuối không phục vụ cho việc phân loại binary nên được loại bỏ và weights=’imagenet’ là có sử dụng pre-trained weights của ImageNet

In [ ]:
```
# ảnh đầu vào có kích thước quy định là 224x224x3
input_shape=(224,224,3)
# loại bỏ các lớp fully connected layers (include top) , với trọng số là imagenet
vgg = VGG16(input_shape=input_shape, include_top=False, weights="imagenet")
for layer in vgg.layers:
    layer.trainable = False # đóng băng tất cả các layers vì đây là pre_trained model
```

- Trước khi sử dụng Pre-trained model để rút trích đặc trưng thì cần phải xử lí ảnh trong bộ dataset về đúng dạng của model cần :
  1. Resize ảnh về kích thước 224x224x3
  2. Chuyển ảnh về ma trận có dạng 224x224x3
  3. Thêm chiều cho ma trận để phù hợp đầu vào của model

4. Loại bỏ các pixel RGB cho phù hợp với định dạng trên tập ImageNet

Code minh họa :

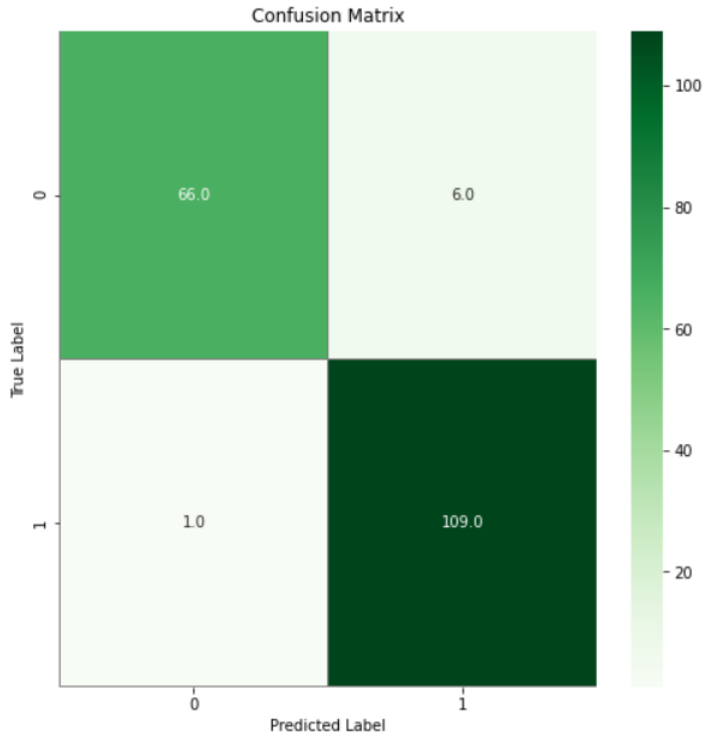
```
feature_list = []
for imgpath in filenames:
    x = load_img(path+"/"+imgpath,target_size=(224,224)) #resize ảnh về 224,224
    img_array = img_to_array(x) # chuyển ảnh về ma trận có shape 224*224*3
    img_array = np.expand_dims(img_array, axis=0)# thêm chiều cho ma trận
    img_array = imagenet_utils.preprocess_input(img_array) # loại bỏ các pixel RGB
    features = vgg.predict(img_array)
    feature_list.append(features)# thêm các feature đã được rút trích vào một mảng
feat_lst = np.reshape(feature_list,(-1,7*7*512))  #reshape lại dạng (n,7*7*512) với n là số ảnh đã qua xử lí
```

b) Đánh giá Model sau khi rút trích đặc trưng với VGG16

Model khi sử dụng VGG16 rút trích cho độ chính xác rất cao lên đến(~98%)

```
Random Forest :  0.9615384615384616
Logistic Regression :  0.978021978021978
```

Confusion matrix:

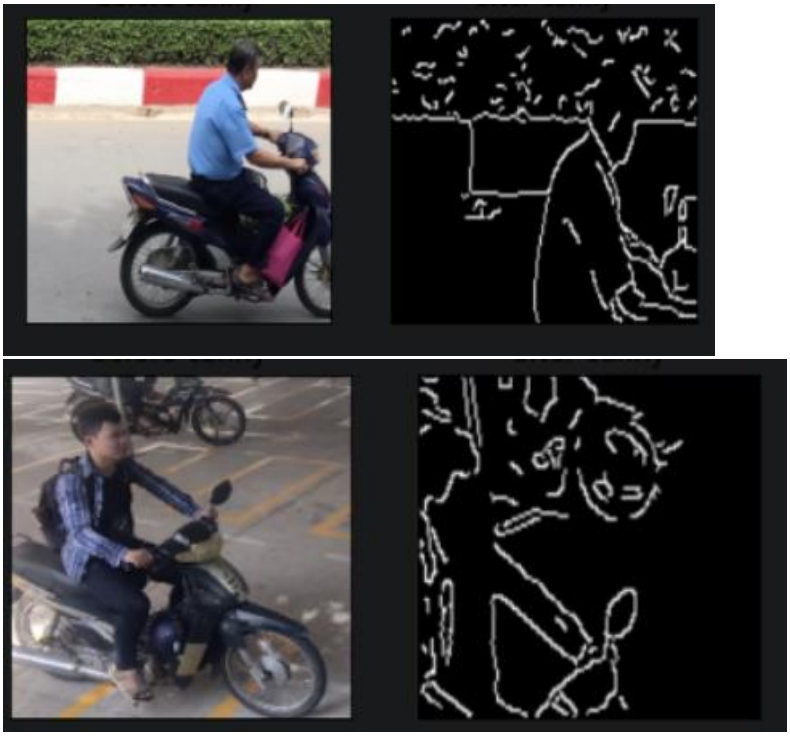


Có thể thấy model chỉ sai 7 hình trong đó 6 hình(dự đoán 1 nhưng thực chất là 0) và 1 hình(dự đoán là 0 nhưng thực chất là 1)

IV. Kết luận

4.1 Nhận xét chung

Đối với cách rút trích đặc trưng bằng cách tính toán độ lệch chuẩn, trung bình sau khi tiền xử lý thì đem lại kết quả tương đối vì nhóm đã crop bớt phần rìa để loại bỏ các vật thể gây nhiễu xung quanh. Tuy nhiên một vài hình vẫn còn nhiều vật gây nhiễu sau khi crop, và đôi khi có thể làm mất đi các chi tiết có giá trị trong việc phân loại. Ví dụ: chỉ thấy mặt người, mất nón bảo hiểm



Còn với mạng VGG16 đem lại kết quả khá cao vì đây là mô hình Deep Learning đã được train sẵn. Ảnh sẽ được rút trích đặc trưng qua từng lớp chập. Vì vậy nhóm đã quyết định chọn VGG16 để làm hàm input cho người dùng.

Link input:
https://github.com/xuanthang482/CS114.K21_CourseProject/blob/master/Input.ipynb

4.2 So sánh hai cách Feature Extraction dựa trên tập data bên ngoài.

Canny Edge : co doi mu

VGG : co doi mu

Canny Edge : Khong doi mu

VGG : co doi mu

Canny Edge : Khong doi mu

VGG : Khong doi mu

Canny Edge : co doi mu

VGG : co doi mu

4.3 Hướng phát triển

Tìm 1 thư viện hoặc phần mềm hỗ trợ bỏ đi backround không cần thiết mà chỉ quan tâm đến chủ thể trong bức ảnh (phần đầu người) .

Sử dụng thêm một số cách feature extraction khác để phát hiện hình dáng hay vật thể trong ảnh.

Hướng phát triển khác em muốn đề cập nữa là thiết kế và sử dụng hoàn toàn 1 mạng tích chập neural network dùng chung cho việc từ tiền xử lí ảnh ,rút trích cho đến việc phân loại ảnh.

V. Tài liệu tham khảo

https://towardsdatascience.com/age-detection-using-facial-images-traditional-machine-learning-vs-deep-learning-2437b2feeab2

https://gist.github.com/gauravgola96/2ef146bc3b9d4f63a0665f679ec703d1?short_path=9e31afa

https://minhng.info/tutorials/xu-ly-anh-opencv-hien-thuc-canny-edge.html?fbclid=IwAR246Zz5ANME9DgKxyh6MpLBx91ly0DEbT6JQXqUaqDaKFaBvou-pXUa3xY