ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

Máy Học

Học kỳ II (2019-2020)

NHẬN DIỆN CÁC TOÀ NHÀ TRONG TRƯỜNG UIT

Sinh viên :	Trần Đình Khang
MSSV:	18520072
Giảng viên:	Lê Đình Duy
	Phạm Nguyễn Trường An

Thành phố Hồ Chí Minh, tháng 8 năm 2020

MỤC LỤC

PHẦN I MỞ ĐẦU

1 Giới thiệu đề tài

Bài toán " Nhận diện các tòa nhà trong trường UIT " là một bài toán thuộc lớp bài toán phân lớp (Classification).

Với bài toán này, dữ liệu được thu thập bằng việc chụp ảnh các toà nhà trong trường sau đó sử dụng các thuật toán trong máy học cho bài toán phân lớp để xử lý.

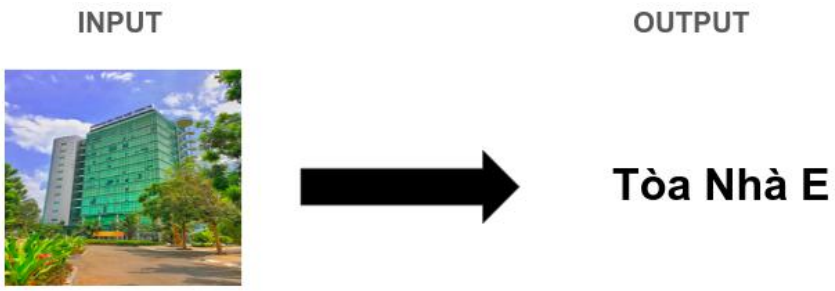
Bài toán này nếu được phát triển và mở rộng sẽ giúp đỡ những sinh viên mới hoặc phụ huynh đến trường có thể dễ dàng hơn trong việc tìm các toà nhà, tìm đường.

2 Mô tả bài toán và bộ dữ liệu

2.1 Mô tả bài toán

- Input: Một bức ảnh về một toà nhà nào đó trong trường UIT.

- Output: Tên của toà nhà đó



Hình 1: Mô tả bài toán

2.2 Mô tả bộ dữ liệu

- Dataset thu thập cùng với các bạn Trung, An, Toàn bằng cách đi chụp các toà nhà trong trường.
- Bộ dữ liệu Train gồm 1096 ảnh file jpg được chụp bằng điện thoại Iphone 7. Các ảnh được chia vào 5 class là : B(Tòa B), C(Tòa C), D(Tòa D), E(Tòa E), CT(Căn Tin).

Trong đó:

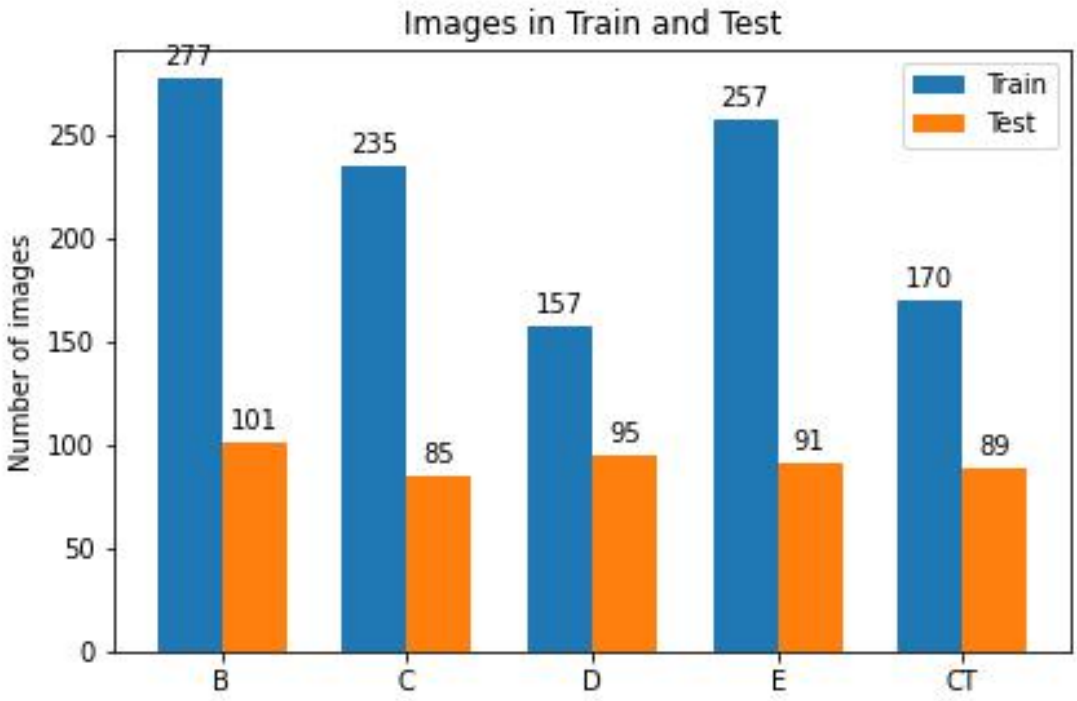
+ Class B(Tòa B): 277 ảnh
+ Class C(Tòa C): 235 ảnh
+ Class D(Toà D): 157 ảnh
+ Class E(Tòa E): 257 ảnh
+ Class CT(Căn Tin): 170 ảnh

- Bộ dữ liệu Test gồm 461 ảnh file jpg chia thảnh 5 class giống như tập train mỗi loại gồm:

+ Class B(Tòa B): 101 ảnh
+ Class C(Tòa C): 85 ảnh

+ Class D(Tòa D): 95 ảnh
+ Class E(Tòa E): 91 ảnh
+ Class CT(Căn Tin): 89 ảnh

- Chia 2 Folder , một Folder lưu tập ảnh để Train và cái còn lại để Test.



Hình 2: Mô tả bộ dữ liệu

PHẦN II NỘI DUNG

1 Tiền xử lí dữ liệu (Preprocessing)

1.1 Resize mỗi bức ảnh thành mỗi bức có kích thước 128 x128.

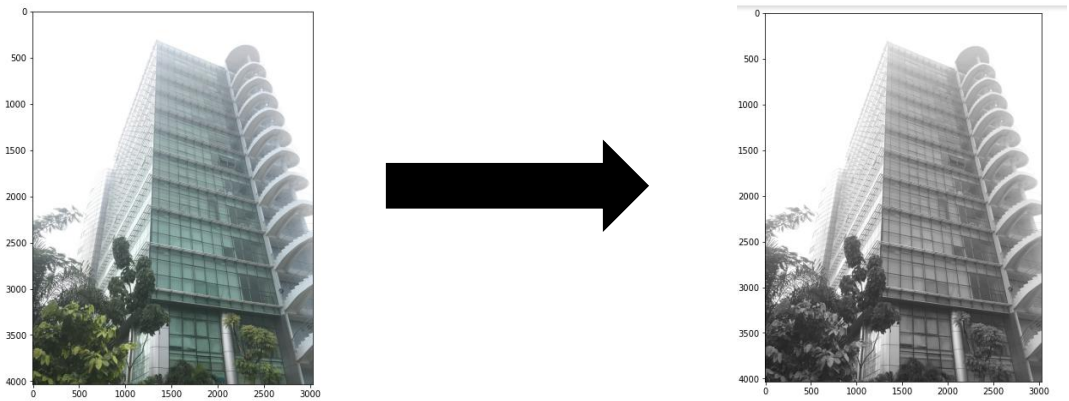
- Sử dụng resize với aspect ratio làm cho ảnh sau khi resize có cảm giác ít bị méo mó hơn.

Hình 3: Aspect Ratio

1.2 Chuyển ảnh màu về grayscale

- Sử dụng hàm cv2.cvtColor trong thư viện cv2.

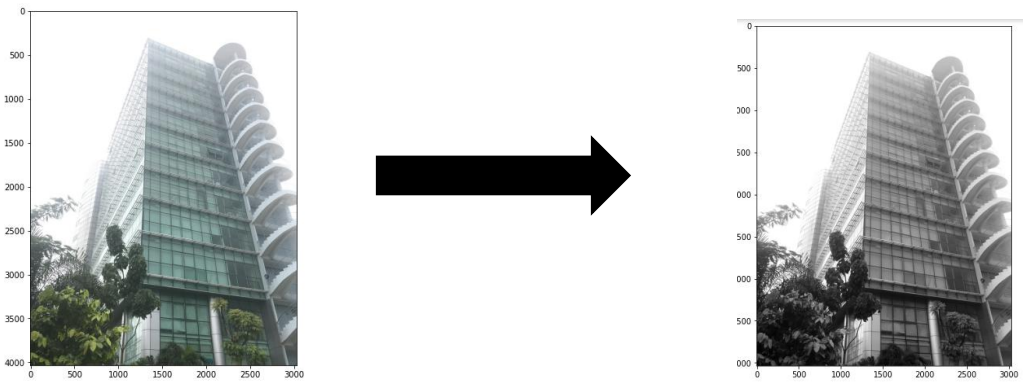
```
ImgTest = cv2.cvtColor(ImgTest,cv2.COLOR_RGB2GRAY)
```



1.3 Lọc nhiễu ảnh, làm mịn ảnh sử dụng Gaussian Blur

- Sử dụng cv2.bilateralFilter trong thư viện cv2.

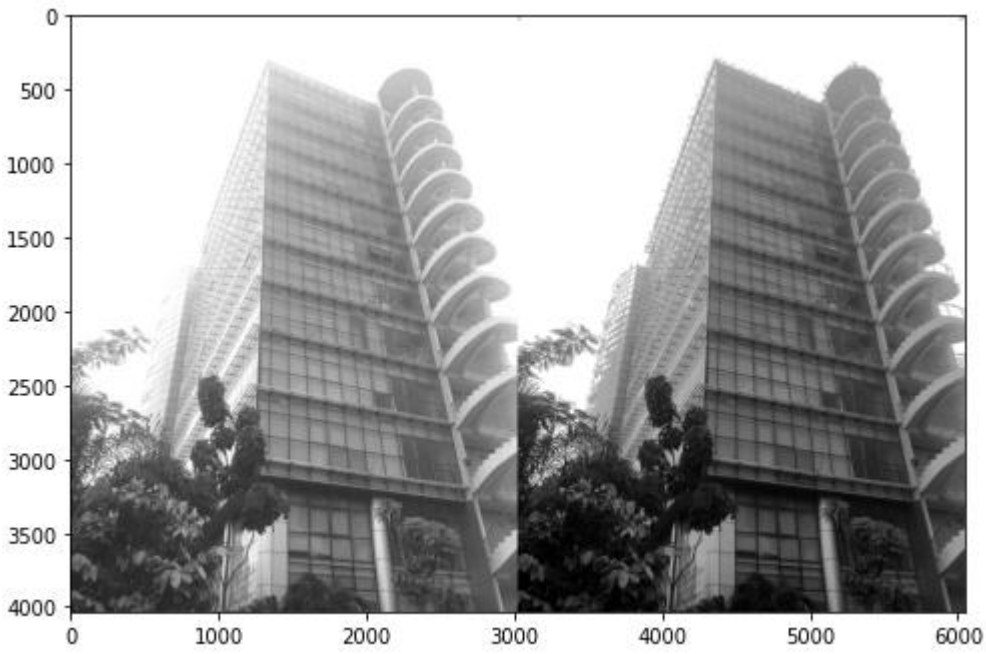
```
Img_Bilateral = cv2.bilateralFilter(ImgTest,9,75,75)
```



1.4 Cân bằng ánh sáng và độ tương phản (Histogram Equlization)

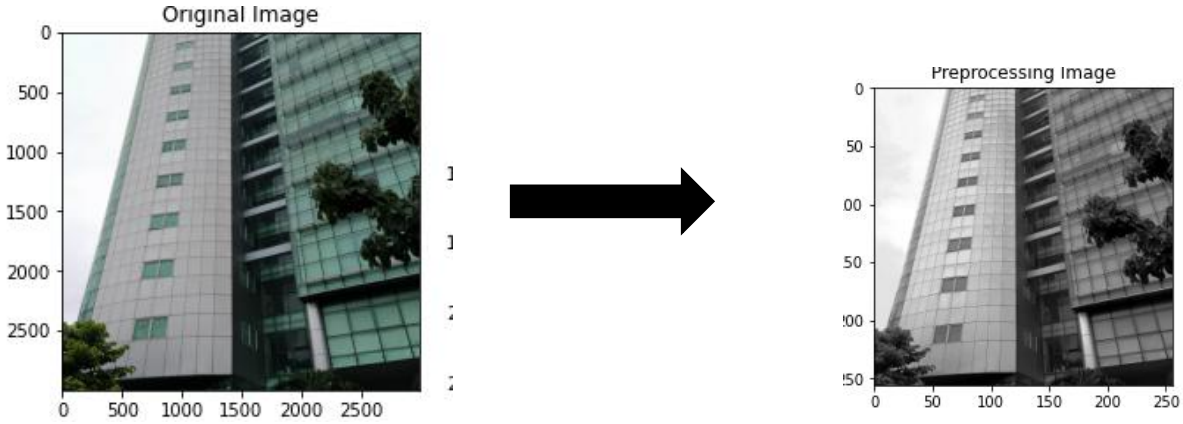
- Sử dụng cv2.equalizeHist() trong thư viện cv2.

```
ImgE = ImgTest
equ = cv2.equalizeHist(ImgE)
```



Hình 3: Trước và Sau Histogram Equlization

Sau khi thực hiên hết các bước tiền xử lí dữ liệu, ta được:



2 Trích xuất đặc trưng (Extract Feature)

2.1 Mỗi pixel trong ảnh grayscale là một feature

Sau khi resize tất cả ảnh về cùng một kích thước (128*128). Coi mỗi pixel sau khi gayscale trong 1 bức ảnh là 1 feature. Resize bức ảnh về vector có kích thước (16384,1)

```
print('+ 2D pixel: \n {}'.format(ImgTest))

Img_to_vec = np.array(ImgTest).flatten()/255.
print('\n+ Vectorize Image: \n {}'.format(Img_to_vec))
```

Ảnh sau khi Vectorize:

```
+ 2D pixel:
 [[255 255 255 ... 255 255 255]
 [255 255 255 ... 255 255 255]
 [255 255 255 ... 255 255 255]
 ...
 [ 67  22  35 ... 153 154 156]
 [ 11  11  47 ... 152 154 155]
 [ 22  17  16 ... 152 153 154]]

+ Vectorize Image:
 [1.         1.         1.         ... 0.59607843 0.6        0.60392157]
```

2.2 LBPs

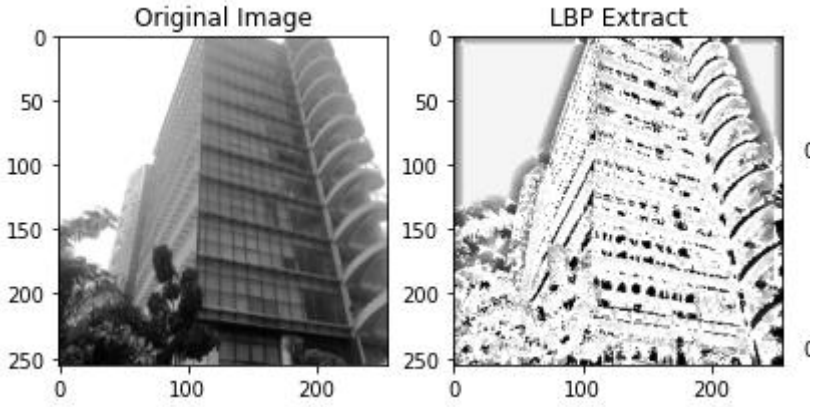
Local binary pattern nó là một thuật toán mô tả texture(cầu trúc) của một image. Ý tưởng cơ bản của nó là mô phỏng lại cấu trúc cục bộ (local texture) của image bằng cách so sánh mỗi pixel với các pixel lân cận nó(neighbors).Ta sẽ đặt một pixel là trung tâm(center) và so sánh với các pixel lân cận với nó, nếu pixel trung tâm lớn hơn hoặc bằng pixel lân cận thì nó sẽ trả về giá trị 1, ngược lại 0.

```
ImgLBP = feature.local_binary_pattern(ImgTest, 24, 8, method="uniform")
```

```
(hist, _) = np.histogram(ImgLBP.ravel(),bins=np.arange(0, 24 + 3),range=(0, 24 + 2))
hist = hist.astype("float")

hist /= (hist.sum() + 1e-7)
```

Ảnh sau khi trích xuất LBPs:



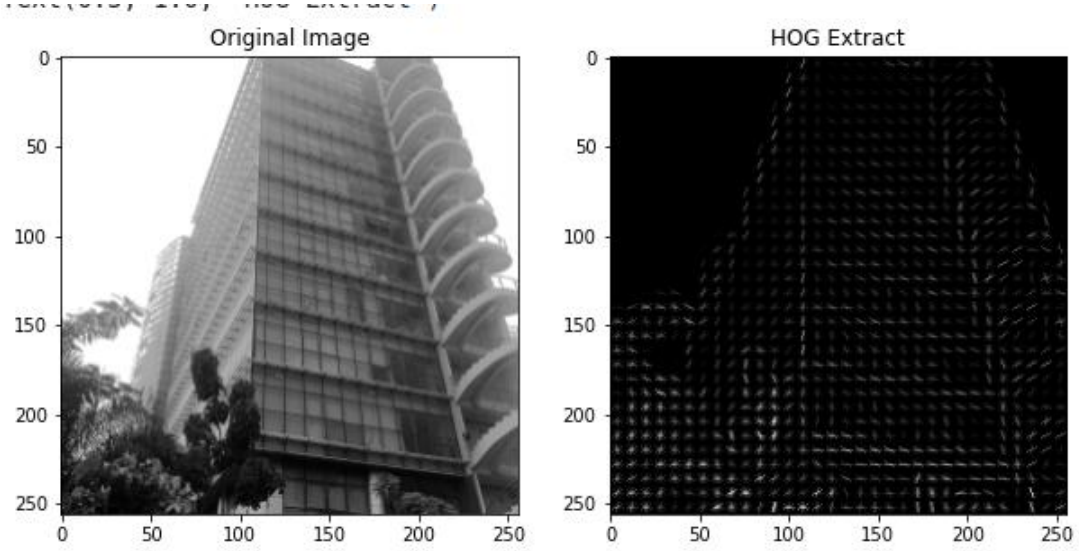
2.3 HOG

HOG là viết tắt của Histogram of Oriented Gradient - một loại "feature - descriptor". Mục đích của "feature descriptor" là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích.

Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.

```
_,ImgHOG = feature.hog(ImgTest, orientations=9, pixels_per_cell=(8, 8),
                       cells_per_block=(2, 2), transform_sqrt=True,
                       block_norm="L2-Hys",visualize=True)
ImgHOG = exposure.rescale_intensity(ImgHOG, out_range=(0, 255))
```

Ảnh sau khi trích xuất HOG:



3 Chọn mô hình

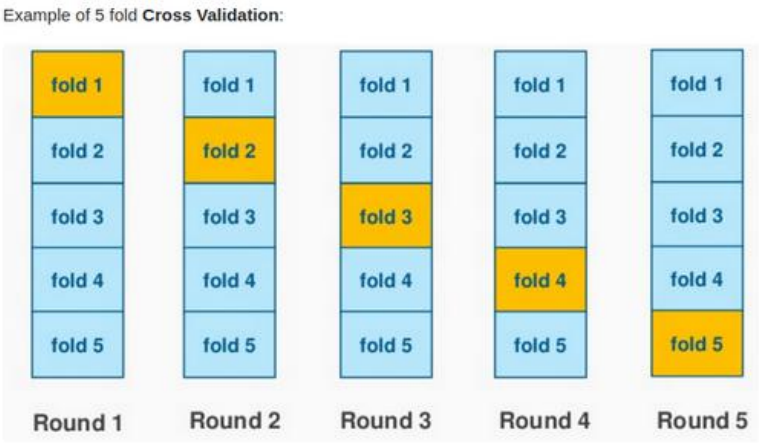
Thử dùng các model có sẵn trong sklearn :

- Linear SVM
- Logistic Regression
- Decision Tree
- Naive Bayes
- KNN

4 Huấn Luyện

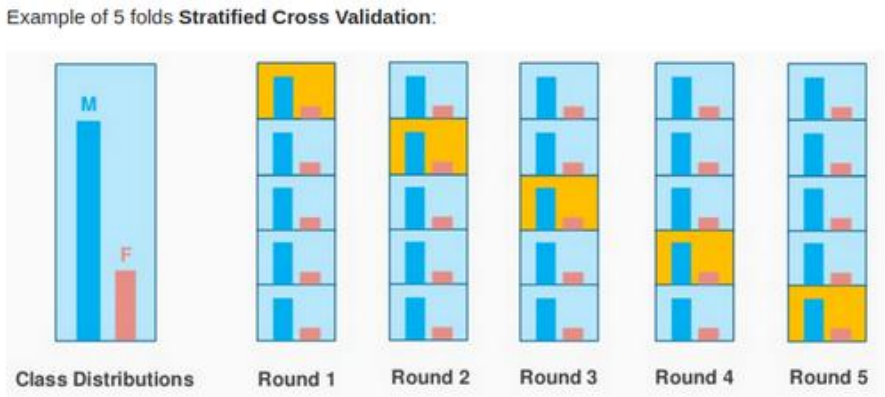
4.1 Cross Validation

Là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường đường sử dụng là chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số k tập con được lấy ra làm validata set. Mô hình sẽ được xây dựng dựa vào hợp của k−1 tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation.



4.2 Stratified Cross Validation

Là phiên bản cải tiến của Cross Validation, thay vì chia ngẫu nhiên tập dữ liệu train thành k fold (Cross Validation), Cross Validation chia tập dữ liệu train thành k fold theo một phân phối xác suất (mean, variance,...).

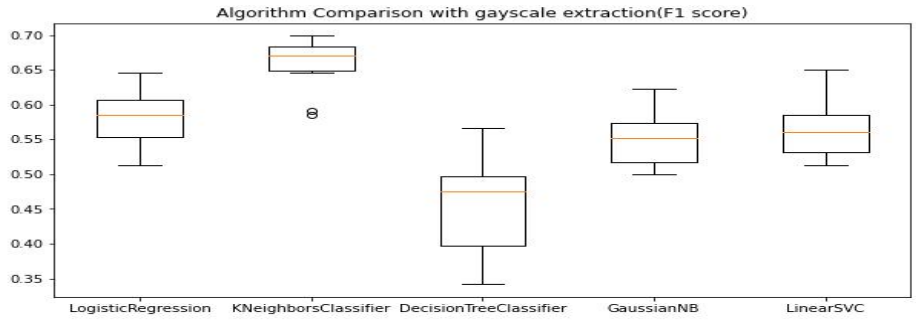


4.3 Chọn mô hình phù hợp với mỗi cách trích xuất

Với mỗi cách trích xuất đặc trưng, ta sẽ so sánh 5 thuật toán đã chọn với kĩ thuật Stratified Cross Validation.

- Đối với trích xuất grayscale là đặc trưng:

- **Kết quả F1 Socre:**

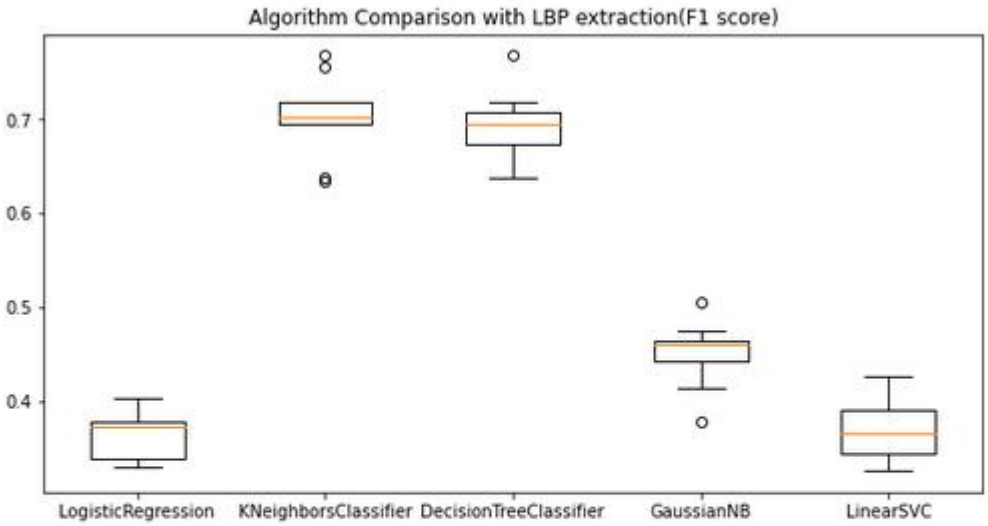



```
LogisticRegression: 0.581487 (0.037614)
KNeighborsClassifier: 0.658184 (0.038204)
DecisionTreeClassifier: 0.454790 (0.064914)
GaussianNB: 0.551117 (0.039171)
LinearSVC: 0.564384 (0.040729)
```

Nhìn vào biểu đồ trên thì ta chọn model KNeighborsClassifier(KNN).

- Đối với trích xuất LBP:

- **Kết quả F1 Socre:**

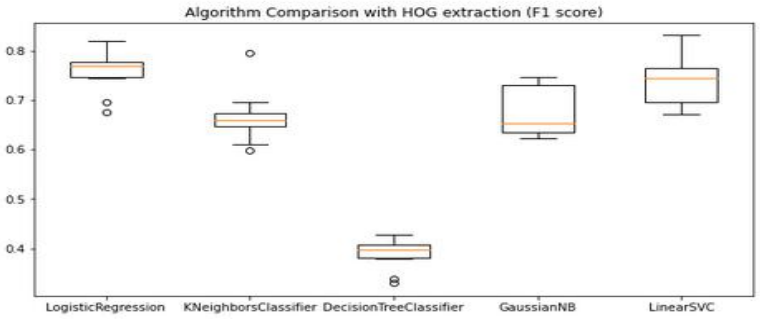



```
LogisticRegression: 0.363738 (0.026137)
KNeighborsClassifier: 0.703232 (0.040950)
DecisionTreeClassifier: 0.693550 (0.034853)
GaussianNB: 0.451264 (0.033060)
LinearSVC: 0.368631 (0.031404)
```

Nhìn vào biểu đồ ta chọn model KNeighborsClassifier(KNN).

- Đối với trích xuất HOG:

- **Kết quả F1 Socre:**




```
LogisticRegression: 0.755495 (0.040145)
KNeighborsClassifier: 0.665281 (0.051342)
DecisionTreeClassifier: 0.388113 (0.030375)
GaussianNB: 0.676256 (0.051066)
LinearSVC: 0.738393 (0.047281)
```

Nhìn vào biểu đồ ta chọn model LogisticRegression.

5 Tinh chỉnh tham số

5.1 RandomSearchCV

RandomizedSearchCV là một phương pháp tìm bộ hyperparameters bằng cách từ những giá trị parameter đã setting, Random Search sẽ chọn ngẫu nhiên các cặp parameter sao cho độ chính xác của tập cần dự đoán là lớn nhất theo một độ đo nhất định nào đó (F1-score, Accuracy, ROC,..)

Ưu điểm : RandomizedSearchCV thường có thời gian chạy nhanh.

Nhược điểm: Vì random một số bộ parameter nên có khi kết quả không được tối ưu, và dẫn đến không được như ý muốn.

5.2 Tinh chỉnh tham số

- Sử dụng RandomizedSearchCV cùng với StratifiedKFold(n_split=5) để tuning model.

- Đối với K-NN tinh chỉnh bộ tham số:

```
leaf_size = list(range(1,50))
n_neighbors = list(range(1,30))
p=[1,2]
weights = ['uniform','distance']
```

- Đối với Logistic Regression tinh chỉnh bộ tham số:

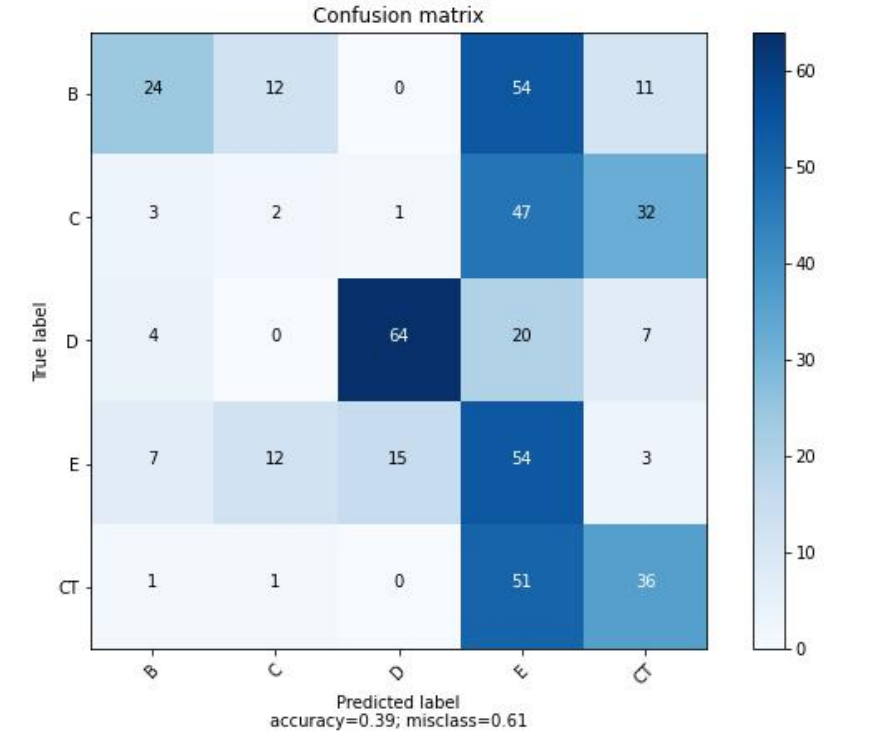
```
solver = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l1', 'l2', 'elasticnet']
C = [10, 4.5, 3, 1.5,1.0, 0.3, 0.1, 0.03, 0.01]
```

6 Đánh giá kết quả, kết luận

Kết quả dự đoán trên tập Test:

- Phương pháp sử dụng ảnh grayscale chuyển sang vector đặc trưng:

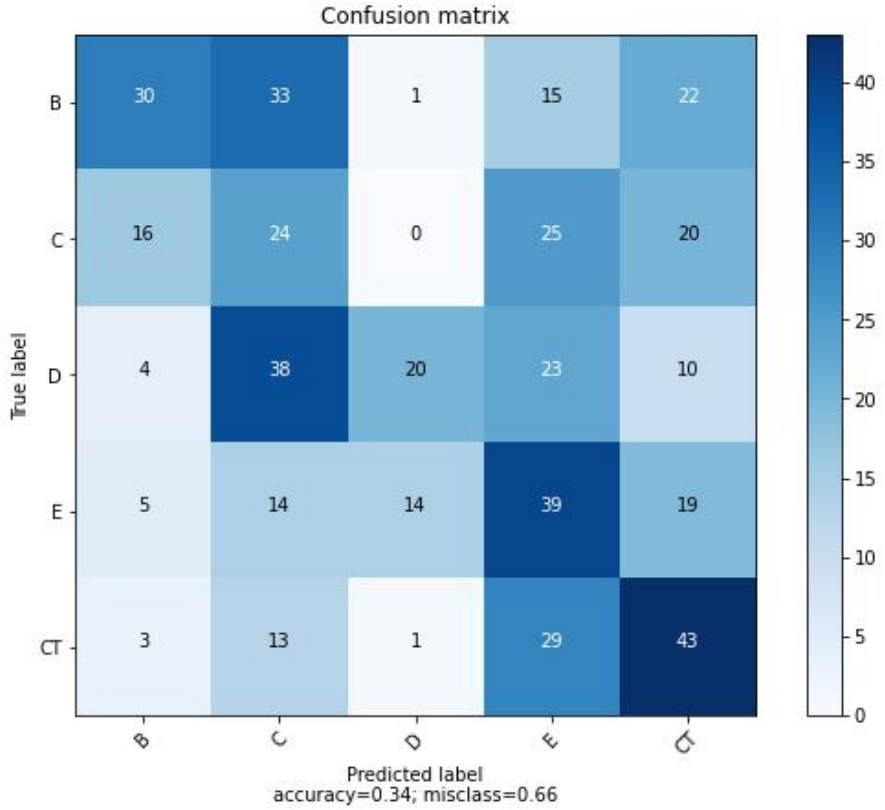
	precision	recall	f1-score	support
B	0.62	0.24	0.34	101
C	0.07	0.02	0.04	85
D	0.80	0.67	0.73	95
E	0.24	0.59	0.34	91
CT	0.40	0.40	0.40	89
accuracy			0.39	461
macro avg	0.43	0.39	0.37	461
weighted avg	0.44	0.39	0.38	461



Đối với Model(K-NN) thì kết quả dự đoán là 0.39, trong đó label E được dự đoán khá nhiều nhưng tỉ lệ dự đoán trúng label E khá thấp (0.26), nhầm lẫn tòa E và tòa B (54 bức ảnh).

- Phương pháp rút trích đặc trưng bằng LBPs:

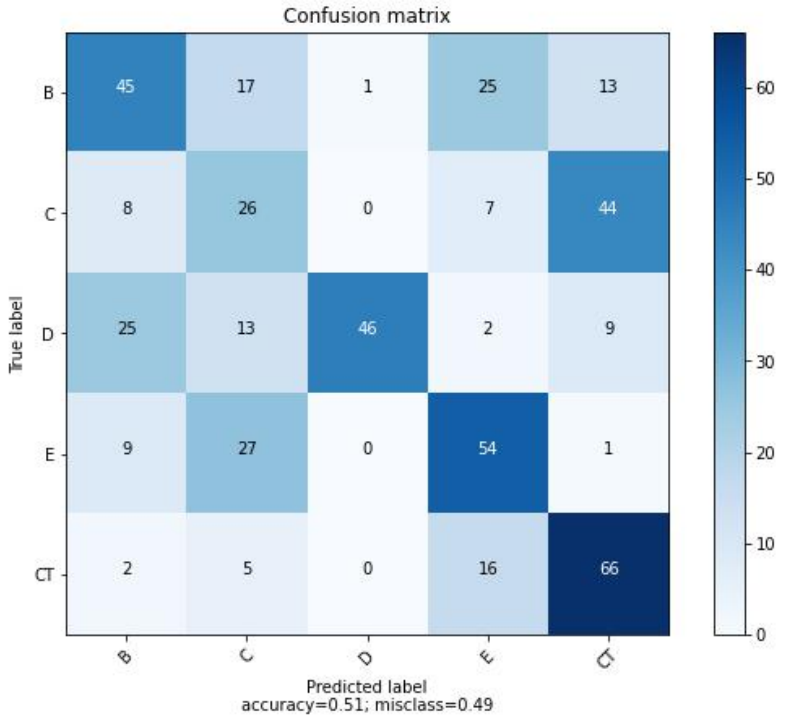
	precision	recall	f1-score	support
B	0.52	0.30	0.38	101
C	0.20	0.28	0.23	85
D	0.56	0.21	0.31	95
E	0.30	0.43	0.35	91
CT	0.38	0.48	0.42	89
accuracy			0.34	461
macro avg	0.39	0.34	0.34	461
weighted avg	0.40	0.34	0.34	461



Đối với Model(K-NN) thì kết quả dự đoán là 0.34, model dự đoán nhiều vào tòa B, nhưng có 38 ảnh bị dự đoán C trong khi các bức ảnh này là tòa D.

- Phương pháp sử dụng ảnh HOG:

	precision	recall	f1-score	support
B	0.51	0.45	0.47	101
C	0.30	0.31	0.30	85
D	0.98	0.48	0.65	95
E	0.52	0.59	0.55	91
CT	0.50	0.74	0.59	89
accuracy			0.51	461
macro avg	0.56	0.51	0.51	461
weighted avg	0.57	0.51	0.52	461



Model logistic regression có accuracy (0.51) cao hơn các model khác nhưng vẫn còn thấp. Dự đoán sai nhiều (44)giữa tòa CT và tòa C.

- Kết Luận:

Sau khi thử với cả 3 cách trích xuất và tuning thì model Logistic với trích xuất HOG có kết quả tốt nhất trên tập Test với thang độ đo F1 (0.51).

```
LogisticRegression(C=1.0, class_weight=None, dual=False, auto fit_intercept=True,
          intercept_scaling=1, l1_ratio=None, max_iter=100000,
          multi_class='auto', n_jobs=None, penalty='l2',
          random_state=42, solver='lbfgs', tol=0.0001, verbose=0,
          warm_start=False)
```

Các model sử dụng đã không có độ chính xác cao < 52%. Nguyên nhân dẫn đến bởi các khó khăn sau:

- Bộ dữ liệu còn quá ít (khoảng 1400 ảnh cho 5 class) chưa có được sự tổng quát cho
mô hình. Có nhiều bức ảnh bị nhiễu bởi lá cây.

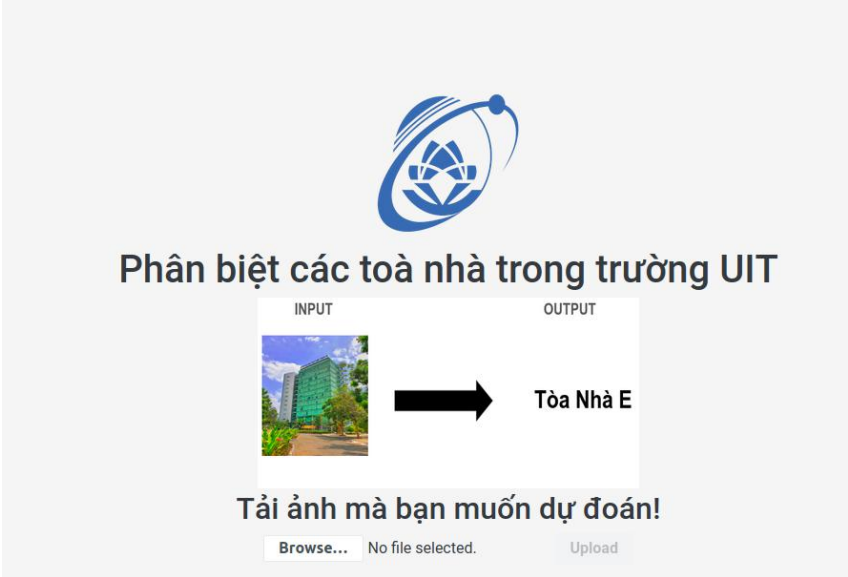
- Phương pháp trích xuất đặc trưng còn hạn chế, chưa loại bỏ được những đối tượng
không mong muốn trong hình. Chưa thử nghiệm với ảnh màu.

- Hạn chế về kỹ năng thu thập dữ liệu. Các tòa nhà có kích thước lớn, không thể chụp toàn bộ một tòa nhà trong một khung hình nên phải chụp từng góc. Khi nhìn vào một vài góc thì sự có sự tương đồng giữa các tòa nhà nên đã gây nên sự nhầm lẫn cho mô hình dự đoán.

PHẦN III DEMO VÀ HƯỚNG PHÁT TRIỂN

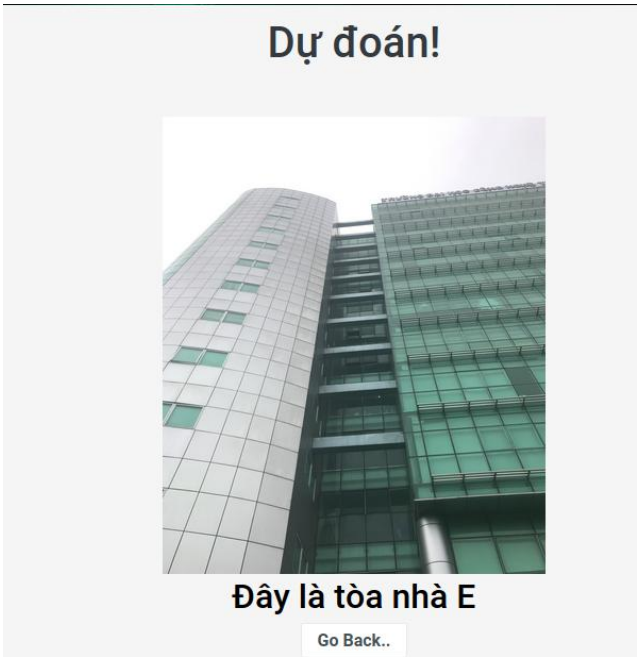
1 DEMO

1.1 Giao Diện



- Phần Upload ảnh:

- Phần dự đoán:



Run Code: export FLASK_APP=application.py && flask run

1.2 Video demo

Link: https://youtu.be/kwNO6GoIqLM

2 Hướng phát triển

- Cải thiện rút trích đặc trưng bằng CNN.

- Chụp thêm các ảnh sao cho ít bị nhiễu, lấy được toàn cảnh tòa nhà, thêm nhiều dữ liệu hơn.

- Sử dụng các mô hình học sâu để giải quyết.

PHẦN IV TÀI LIỆU THAM KHẢO

[1] Blog Machine Learning cơ bản, Vũ Hữu Tiệp

[2] OpenCV, https://opencv.org/

[3] Scikit-image, https://scikit-image.org/