ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: MÁY HỌC

ĐỀ TÀI:

PHÂN LOẠI CHỮ VIẾT TAY TIẾNG VIỆT CÓ DẤU

Lớp: CS114.L11.KHCL

Giảng viên hướng dẫn:

1. Lê Đình Duy
2. Phạm Nguyễn Trường An

Sinh viên thực hiện:

1. Phan Nguyễn Thành Nhân 19521943
2. Lê Quang Huy 19521617
3. Nguyễn Thành Nghĩa 19521899

Thành phố Hồ Chí Minh, ngày 21 tháng 01 năm 2021

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: MÁY HỌC

ĐỀ TÀI:

PHÂN LOẠI CHỮ VIẾT TAY TIẾNG VIỆT CÓ DẤU

Lớp: CS114.L11.KHCL

Giảng viên hướng dẫn:

1. Lê Đình Duy
2. Phạm Nguyễn Trường An

Sinh viên thực hiện:

1. Phan Nguyễn Thành Nhân	19521943
2. Lê Quang Huy	19521617
3. Nguyễn Thành Nghĩa	19521899

Thành phố Hồ Chí Minh, ngày 21 tháng 01 năm 2021

MỤC LỤC

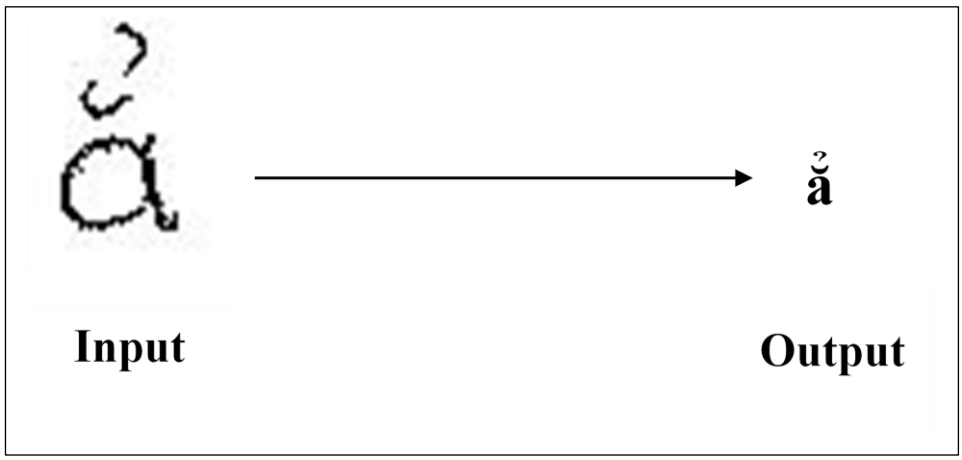
I. Giới thiệu đề tài

1. Tổng quan về đề tài

- Bài toán nhận diện chữ cái viết tay mang lại một lợi ích rất lớn cho con người trong các hoạt động thường ngày. Nhưng trước khi giải quyết được bài toán lớn đó thì chúng ta phải giải quyết được một bài toán nhỏ khác đó là phân biệt chữ cái viết tay. Đã có khá nhiều nghiên cứu về đề tài phân biệt chữ viết tay nhưng nhận diện chữ cái Tiếng Việt viết tay vẫn chưa nhiều. Đó là lý do nhóm quyết định thực hiện đề tài này.

2. Mô tả bài toán

- Bài toán này thuộc lớp bài toán phân loại, có tổng cộng 89 lớp đại diện cho 89 chữ cái tiếng Việt viết thường bao gồm cả các dấu phụ (sắc, huyền, hỏi, ngã, nặng).
- Đầu vào của bài toán là một tấm ảnh trong đó có chứa đúng một chữ cái tiếng Việt viết thường.
- Đầu ra là kết quả dự đoán chữ cái tương ứng với tấm ảnh đó.

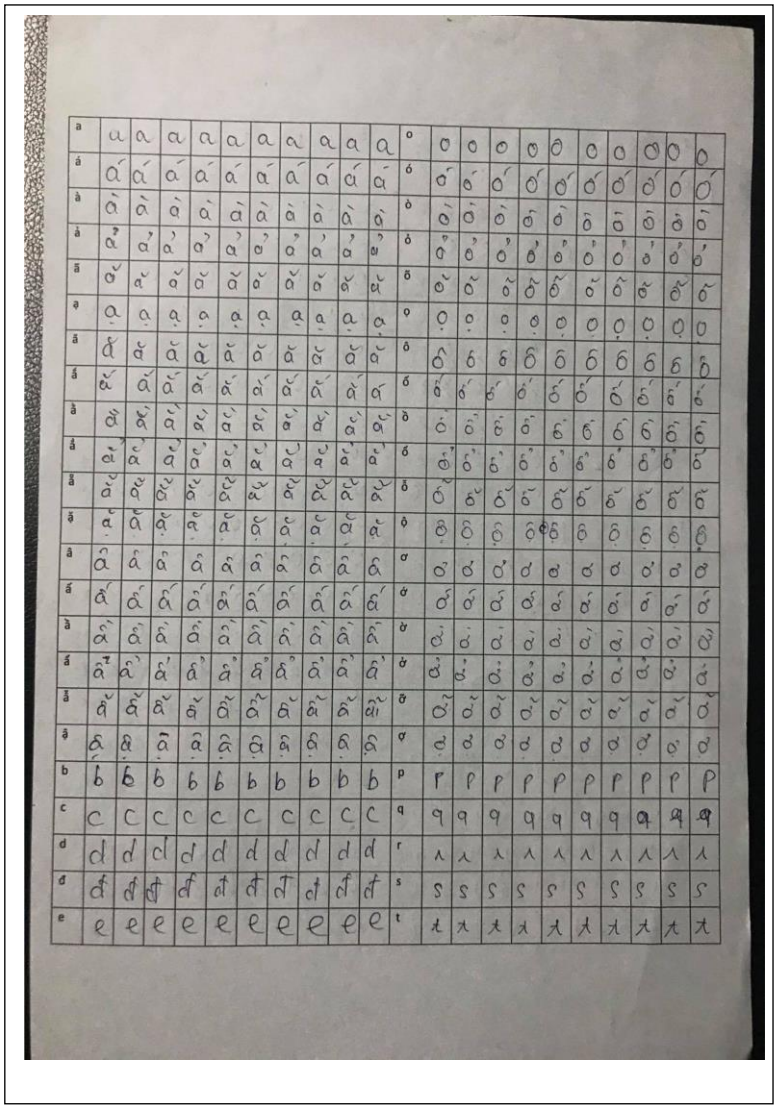


Hình 1. Mô phỏng bài toán

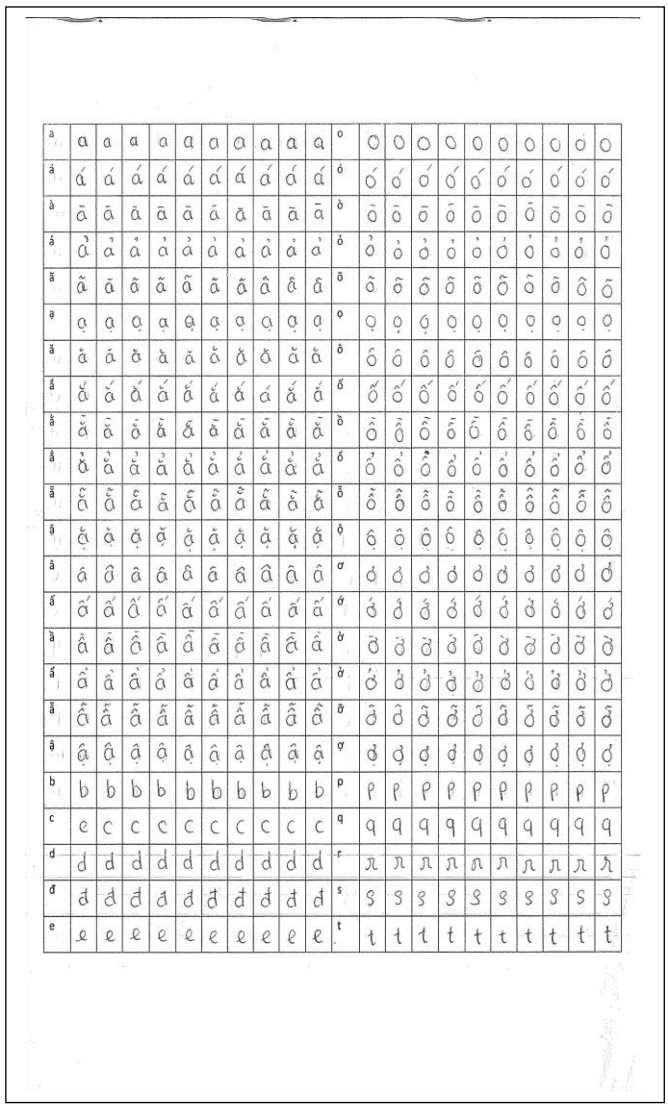
II. Mô tả về bộ dữ liệu

1. Cách thu thập dữ liệu

- Dữ liệu được nhóm thu thập từ hơn 20 sinh viên tình nguyện của Trường Đại học Bách Khoa và Trường Đại học Công Nghệ Thông tin (ĐHQG Thành phố Hồ Chí Minh).
- Nhóm sẽ chuẩn bị sẵn các mẫu giấy và sẽ nhờ các tình nguyện viên viết những con chữ vô tờ giấy như hình 2, sau đó nhóm sẽ mang các tờ giấy đó đi scan ra thành những tấm ảnh như hình 3.



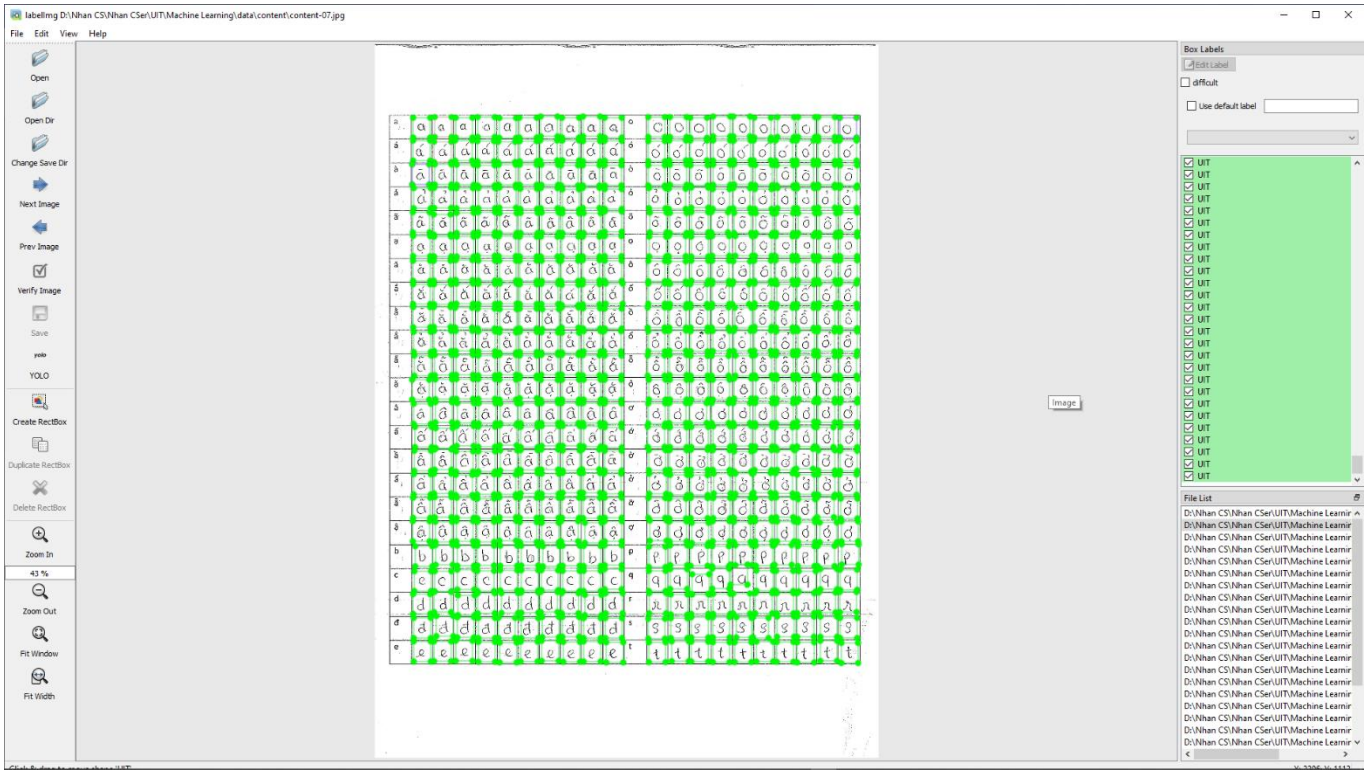
Hình 2: Mẫu thu thập dữ liệu



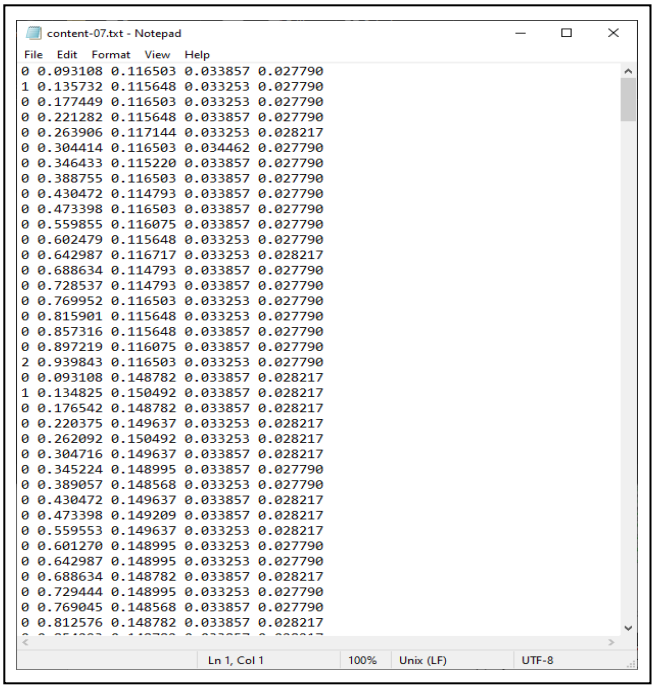
Hình 3: Ảnh đã được scan

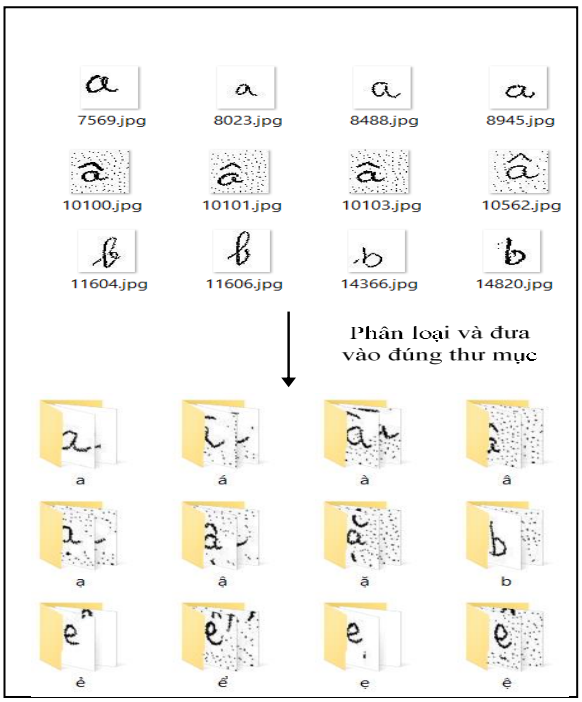
- Để có thể lấy được từng con chữ trong tấm ảnh đã được scan, nhóm đã thực hiện theo các bước sau đây:

  - Bước 1: Sử dụng tool labelimg của tzutalin để label từng con chữ như trong hình (link git: github.com/tzutalin/labelImg).



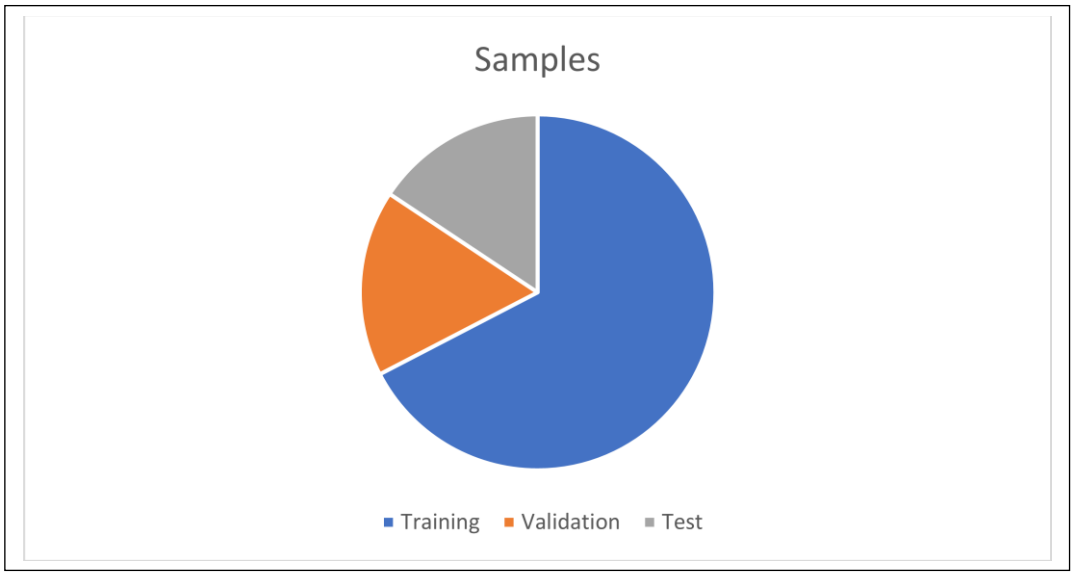
  - Bước 2: Sau khi quá trình label hoàn tất nhóm sử dụng các thông số trong file annotate do tool labelimg tạo ra để cắt ra thành những tấm ảnh chỉ chứa 1 chữ cái rồi sau đó phân loại các tấm ảnh về thành những thư mục riêng.





2. Phân chia dữ liệu

- Sau khi phân loại và gán nhãn cho dữ liệu, nhóm thống kê được tổng cộng 17.607 mẫu với 89 class, trung bình mỗi class sẽ có khoảng 150 tấm ảnh.
- Nhóm chia dữ liệu thu thập được thành 3 tập training set, validation set và test set với số lượng như sau:
    - Training set với 11.865 mẫu, các mẫu từ training set và validation set có thể do cùng một người viết.
    - Validation set với 2.984 mẫu, dùng để đánh giá mô hình sau khi train. Các mẫu ở tập này không được dùng để huấn luyện mô hình.
    - Test set với 2.758 mẫu được thu thập riêng biệt với hai tập trên.



Hình 2. Sự phân bố của các tập dữ liệu

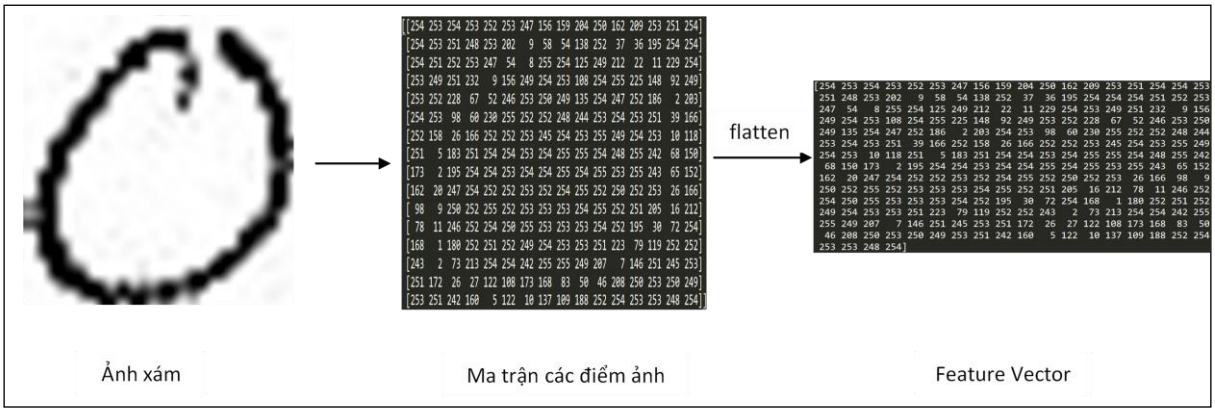
III. Xử lý dữ liệu và rút trích đặc trưng

- Tiền Xử lý dữ liệu:
    - Mỗi tấm ảnh trong tập train và validation được scan để loại bỏ nhiễu trong quá trình thu thập.
    - Loại bỏ các yếu tố gây nhiễu khi cắt hình như bị dính viền đen.
    - Cắt bớt các khoảng trắng dư thừa xung quanh chữ.
    - Resize về kích thước 16x16 pixel.

- Xử lý dữ liệu: nhóm xử lý dữ liệu theo hai cách.
  - Cách 1: grayscale ảnh và flatten ảnh thành một vector
  - Cách 2: sử dụng kỹ thuật hog để mô tả sự xuất hiện của chữ cái trong ảnh và lấy feature

1. Chuyển sang ảnh xám (greyscale) và flatten vector

- Chuyển sang ảnh xám để giảm số chiều của ảnh và flatten ảnh thành một vector có 256 phần tử (16x16) để thuận tiện cho việc huấn luyện mô hình là một kỹ thuật xử lý ảnh khá đơn giản.



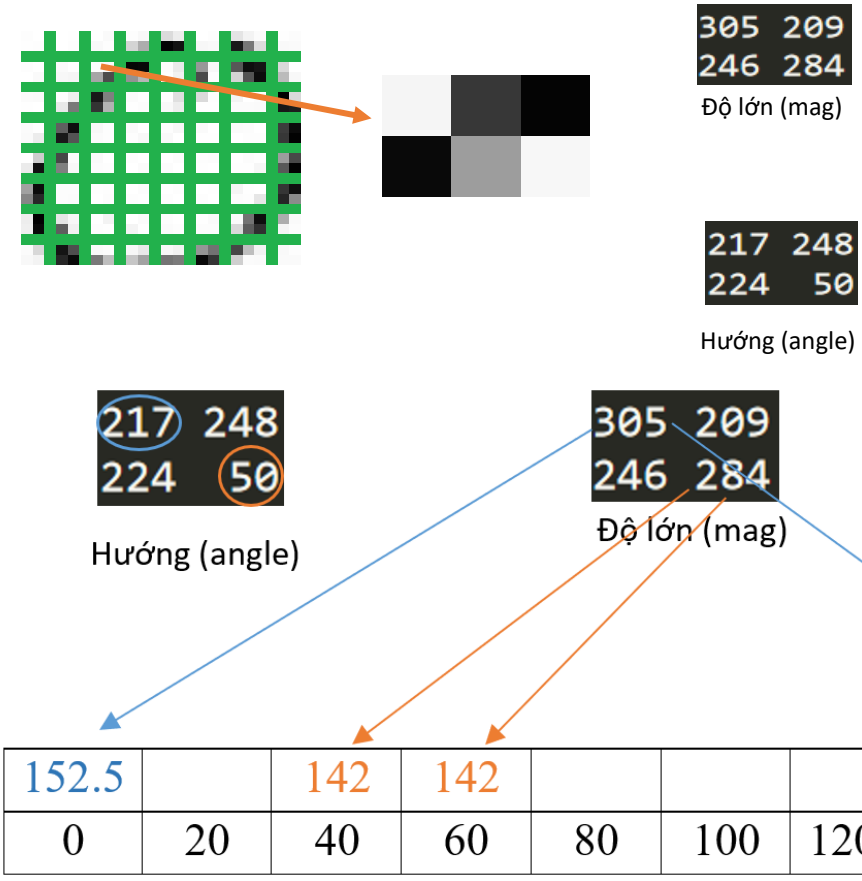
Hình 3. Mô phỏng chuyển sang ảnh xám và flatten vector

- Tuy nhiên với loại dữ liệu phi cấu trúc như hình ảnh cách làm này có phần kém hiệu quả vì ảnh sau khi được chuyển sang vector đơn thuần là một dãy số không chứa nhiều thông tin.

2. Dùng kỹ thuật rút trích đặc trưng HOG

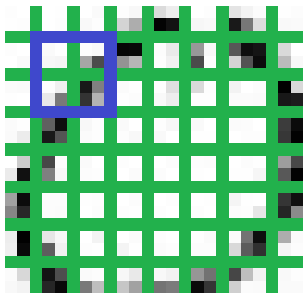
- Một cách làm khác để xử lý ảnh là sử dụng kỹ thuật rút trích đặc trưng HOG (Histogram of Orientation Gradients).
- Hình ảnh được chia thành từng ô nhỏ nối tiếp nhau kích thước 2x2 pixel. Từ giá trị của các pixel ở các ô nhỏ này ta có thể tính ra được độ lớn và hướng của nó bằng các hàm có sẵn của thư viện opencv.

```
# Calculate gradient
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=1)
# Python Calculate gradient magnitude and direction ( in degrees )
mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)
```



152.5		142	142					152.5
0	20	40	60	80	100	120	140	160

- Sau khi tìm được độ lớn và hướng của các khối nhỏ 2x2 pixel, ta thêm giá trị độ lớn(mag) vào trong khoảng hướng (angle) tương ứng của nó để được một vector có 9 phần tử như hình.
- Tiếp theo chúng ta tạo một khối vuông gồm 4 ô, với mỗi ô là 2x2 pixel.



- Vì mỗi ô tạo ra được 1 vector có 9 phần tử nên mỗi khối khi ghép nối tiếp các vector lại sẽ được một vector có 36 phần tử.
- Các khối này dịch chuyển theo từng ô của bức hình. Sau khi đi qua hết 49 vị trí (7x7) có thể trong hình và ghép nối tiếp các vector 36 phần tử lại ta được một vector có 1764 phần tử.
- Vector này chứa thông tin về hướng và độ lớn của các pixel trong hình. Đây chính là Feature vetcor cuối cùng thu được.

IV. Cài đặt và tinh chỉnh tham số

- Với bài toán phân loại chữ viết tay tiếng việt, nhóm đã chọn ra 3 thuật toán để huấn luyện:
  + Logistic Regression
  + SVM (linear and non-linear)
  + Multi layer perceptron

1. Thực nghiệm trên Logistic Regression

```
Lgr1= LogisticRegression(C=0.1, solver = 'liblinear', max_iter = 1000)
Lgr1.fit(x_train, y_train)
y_pred_val = Lgr1.predict(x_val)
y_pred_test = Lgr1.predict(x_test)
```

- Các thông số :
  - C = 0.1 (Inverse regularization parameter): thông số dùng để ngăn chặn sự overfit của mô hình, C càng nhỏ thì mô hình sẽ ít bị overfit (nhưng nếu quá nhỏ có thể làm mô hình bị underfit).
  - solver = ‘liblinear’: Sử dụng Coordinate descent làm thuật toán tối ưu các trọng số của hàm Logistic Regression.
  - max_iter = 1000 : Số lần lặp tối đa để thuật toán có thể tìm ra được local mininum của bài toán.

2. Thực nghiệm trên SVM

```
SVC_model3 = SVC(C=8)
SVC_model3.fit(x_train, y_train)
print(accuracy_score(y_val, SVC_model3.predict(x_val)))
```

```
0.3867292225201072
```

- Các thông số:
    - C = 8 (Regularization parameter) : Thông số dùng để hạn chế sự nhầm lẫn(misclassifying) giữa các bộ dữ liệu trong tập train. C càng lớn thì biên (magin) của đường phân chia các class sẽ càng nhỏ.

3. Thực nghiệm trên MLP

```
MLP2 = MLPClassifier(hidden_layer_sizes=(1000, 1000, 5000), max_iter=500)
MLP2.fit(x_train, y_train)
print(accuracy_score(y_val, MLP2.predict(x_val)))
```

```
0.6437667560321716
```

- Các thông số:
    - hidden_layer_sizes = (1000, 1000, 5000) : Số lượng hidden layer và số lượng neurons mỗi layer.
    - solver = ‘adam’ : sử dụng stochastic gradient descent để tối ưu các trọng số.
    - max_iter = 500, với solver = ‘adam’ thì max_iter đại diện cho số lần sử dụng các bộ dữ liệu trong tập train để huấn luyện cho mô hình.

V. Demo chương trình

- **Clip demo được đính kèm trong CS114.L11.KHCL.FinalReport.**

VI. Đánh giá kết quả

- Độ đo được sử dụng để đánh giá mô hình là accuracy, accuracy càng cao thì mô hình càng tốt.
- Sau quá trình training thì nhận được kết quả predict trên tập train và tập test như sau:

1. **Với bộ dữ liệu sử dụng phương pháp grayscale và flatten vector**
   - **Thuật toán Logistic Regression:**

```
Lgr1= LogisticRegression(C=0.1, solver = 'liblinear', max_iter = 1000
Lgr1.fit(x_train, y_train)
y_pred_val = Lgr1.predict(x_val)
y_pred_test = Lgr1.predict(x_test)
```

- Kết quả đối với tập validation: Accuracy 26%

accuracy			0.26	2984
macro avg	0.30	0.26	0.26	2984
weighted avg	0.29	0.26	0.26	2984

- Kết quả đối với tập test: Accuracy 10%

accuracy			0.10	2758
macro avg	0.10	0.20	0.10	2758
weighted avg	0.26	0.10	0.11	2758

   - **Thuật toán SVM:**

```
SVC_model3 = SVC(C=8)
SVC_model3.fit(x_train, y_train)
```

- Kết quả đối với tập validation: Accuracy 39%

accuracy			0.39	2984
macro avg	0.42	0.38	0.39	2984
weighted avg	0.42	0.39	0.39	2984

- Kết quả đối với tập test: Accuracy 10%

accuracy			0.10	2758
macro avg	0.25	0.10	0.11	2758
weighted avg	0.25	0.10	0.11	2758

- **Thuật toán MLPClassifier:**

```
MLP2 = MLPClassifier(hidden_layer_sizes=(1000, 1000, 5000), max_iter=500)
MLP2.fit(x_train, y_train)
print(accuracy_score(y_val, MLP2.predict(x_val)))
```

```
0.6263404825737265
```

- Kết quả đối với tập validation: Accuracy 63%

accuracy			0.63	2984
macro avg	0.64	0.62	0.62	2984
weighted avg	0.64	0.63	0.63	2984

- Kết quả đối với tập test: Accuracy 20%

accuracy			0.20	2758
macro avg	0.36	0.20	0.21	2758
weighted avg	0.36	0.20	0.22	2758

Nhận xét:

- Các mô hình SVM, Logistic Regression và MLPClassifier đều bị underfit, nguyên nhân ban đầu được đánh giá là do mô hình chưa đủ tốt, ở các mô hình SVM và Logistic Regression thì cả bộ validation và bộ test đều cho kết quả thấp.
- Điều đó cho thấy mô hình không phù hợp để giải quyết bài toán. Với mô hình MLPClassifier thì accuracy của bộ validation tăng lên (63%), mô hình có khả năng dự đoán tốt với các bộ dữ liệu được học tuy nhiên vẫn chưa phân biệt tốt các dữ liệu ở tập test.
- Vấn đề nằm ở dữ liệu chưa được chuẩn bị tốt, cần thay đổi dữ liệu bằng kỹ thuật HOG(Histogram of Oriented Gradients).

2. Với bộ dữ liệu sử dụng phương pháp grayscale và HOG

○ Thuật toán Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(max_iter=500)
clf.fit(X_train, Y_train)
y_pred_vali = clf.predict(X_vali)
y_pred_test = clf.predict(X_test)
```

- Kết quả đối với tập validation: Accuracy 31%

accuracy			0.31	2984
macro avg	0.32	0.31	0.31	2984
weighted avg	0.32	0.31	0.31	2984

- Kết quả đối với tập test: Accuracy 14%

accuracy			0.14	2758
macro avg	0.15	0.13	0.13	2758
weighted avg	0.15	0.14	0.13	2758

○ Thuật toán SVM:

```
from sklearn.svm import SVC
clf = SVC(C=8,kernel='rbf')
clf.fit(X_train, Y_train)
y_pred_vali = clf.predict(X_vali)
y_pred_test = clf.predict(X_test)
```

- Kết quả đối với tập validation: Accuracy 43%

accuracy			0.43	2984
macro avg	0.44	0.42	0.43	2984
weighted avg	0.44	0.43	0.43	2984

- Kết quả đối với tập test: Accuracy 19%

accuracy			0.19	2758
macro avg	0.20	0.19	0.18	2758
weighted avg	0.20	0.19	0.18	2758

- **Thuật toán MLPClassifier:**

```
MLP2_hog = MLPClassifier(hidden_layer_sizes=(1000, 1000, 1000), max_iter=500)
MLP2_hog.fit(X_train, Y_train)
```

- Kết quả đối với tập validation: Accuracy 49%

accuracy			0.49	2984
macro avg	0.49	0.48	0.48	2984
weighted avg	0.49	0.49	0.49	2984

- Kết quả đối với tập test: Accuracy 24%

accuracy			0.24	2758
macro avg	0.27	0.24	0.24	2758
weighted avg	0.27	0.24	0.24	2758

Nhận xét:

- Accuracy của các mô hình có sự thay đổi:
  - + Logistic Regression:
    - Trên bộ validation: Accuracy tăng từ 26% lên 31%.
    - Trên bộ test: Accuracy tăng từ 10% lên 14%.
  - + SVM:
    - Trên bộ validation: Accuracy tăng từ 39% lên 43%.
    - Trên bộ test: Accuracy tăng từ 10% lên 19%.
  - + MLPClassifier:
    - Trên bộ validation: Accuracy giảm từ 63% xuống 49%.
    - Trên bộ test: Accuracy tăng từ 20% lên 24%.
- Bộ dữ liệu được xử lý bằng phương pháp HOG cho kết quả accuracy ở tập test cao hơn, vì thế với các ảnh sau này sẽ dùng phương pháp hog để lấy feature thay vì dùng grayscale + flatten vector.
- Thông qua thử nghiệm với các mô hình thì có thể thấy được MLPClassifier cho kết quả tốt hơn ở cả 2 tập dữ liệu nên nhóm sẽ sử dụng MLPClassifier làm mô hình chính cho các tập dữ liệu sau này.

- Tuy nhiên kết quả vẫn chưa được cải thiện rõ rệt, nguyên nhân là do kích thước bộ dữ liệu chưa đủ nhiều để máy có thể học tốt hơn. Nên nhóm đã quyết định sử dụng phương pháp augment data để tăng bộ dữ liệu.

3. Với bộ dữ liệu tăng cường bằng phương pháp augment data (blur + rotate)

- **Thuật toán MLPClassifier:**

```
MLP2_hog = MLPClassifier(hidden_layer_sizes=(1000, 1000, 1000), max_iter=500)
MLP2_hog.fit(X_train, Y_train)
```

- Kết quả đối với tập validation: Accuracy 50%

accuracy			0.50	2984
macro avg	0.53	0.50	0.50	2984
weighted avg	0.53	0.50	0.51	2984

- Kết quả đối với tập test: Accuracy 26%

accuracy			0.26	2758
macro avg	0.31	0.26	0.26	2758
weighted avg	0.31	0.26	0.26	2758

Nhận xét:

- Độ chính xác tăng từ 24% lên 26%, điều đó cho thấy phương pháp Augment data có hiệu quả.
- Mô hình vẫn bị underfit, nguyên nhân rõ ràng nhất là do dữ liệu vẫn quá ít để có thể huấn luyện máy học, muốn tăng Accuracy thì cách tốt nhất là tăng kích thước tập dữ liệu để huấn luyện cho máy để đạt kết quả tốt hơn.

VII. Kết luận

1. Kết luận khái quát

- Mô hình sau khi huấn luyện có độ chính xác rất thấp, điểm accuracy cao nhất là 26%.
- Trong đó các class ‘è’, ‘’ị’, ‘ư’ có độ chính xác thấp nhất là 0%. Cụ thể class ‘è’ hay bị dự đoán sang class ‘ỳ’, class ‘ị’ hay dự đoán sang class ‘t’ và ‘l’, class ‘ư’ hay dự đoán sang class ‘ứ’, ‘ự’ và ‘v’.

- Nguyên nhân chủ yếu khiến cho mô hình chưa tốt:
    + Quá ít dữ liệu, trung bình mỗi class trước khi augment data là 130 ảnh cho máy học, con số này tăng lên khi augment data là không nhiều dẫn đến sự cải thiện accuracy là không đáng kể.
    + Số class quá nhiều dẫn đến các mô hình máy học đơn giản không phù hợp.
    + Hiểu biết về mô hình là chưa đủ nhiều, chưa biết chọn các layer phù hợp cho mô hình MLPclassifier .
    + Chưa có kinh nghiệm trong khâu thu thập và gán dữ liệu nên dẫn đến dữ liệu nhiễu khá nhiều.

2. Hướng cải thiện và phát triển

- Tăng thêm kích thước dữ liệu, kỳ vọng là với 10000 ảnh cho mỗi class thì accuracy sẽ đạt trên 60%.
- Học thêm về cách sử dụng mô hình tốt hơn, cách tuning hyperparameter cho các mô hình.
- Tìm kiếm thêm các phương pháp xử lý, rút trích đặc trưng cho ảnh.
- Phát triển bài toán rộng ra, không chỉ là phân loại chữ viết nữa, mà sẽ vừa nhận diện chữ viết trong ảnh, vừa lấy những con chữ đó ra thành các đoạn text trên máy để xử lý các bài toán khác (ví dụ: nhận diện cảm xúc của nhân vật qua lời thoại trong truyện tranh, phân loại khu vực giao thư thông qua địa chỉ trên thư,…)

VIII. Tài liệu tham khảo

Các nguồn tài liệu tham khảo dữ liệu:

1. **https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel**
2. **https://www.quora.com/What-is-the-C-parameter-in-logistic-regression**
3. **https://learnopencv.com/histogram-of-oriented-gradients/**
4. **https://stackoverflow.com/questions/38640109/logistic-regression-python-solvers-defintions**
5. **https://machinelearningcoban.com/2017/04/22/kernelsmv/**

6. **https://machinelearningcoban.com/2017/01/27/logisticregression/**
7. **https://machinelearningcoban.com/2017/02/24/mlp/**
8. **https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html**
9. **https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html**
10. **https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html**
11. **https://www.geeksforgeeks.org/python-opencv-cv2-blur-method/**
12. **https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html**