



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
ĐẠI HỌC QUỐC GIA TPHCM

BÁO CÁO ĐỒ ÁN CUỐI KÌ MÔN MACHINE LEARNING

LỚP: CS114.L22.KHCL

GIẢNG VIÊN: PGS.TS. LÊ ĐÌNH DUY -
THS. PHẠM NGUYỄN TRƯỜNG AN

NHẬN DẠNG KÍ TỰ VIẾT TAY TIẾNG VIỆT

(Vietnamese Handwritten Alphabets Recognizer)



1. Trịnh Tuấn Nam - 19521874

2. Nguyễn Dương Hải - 19521464

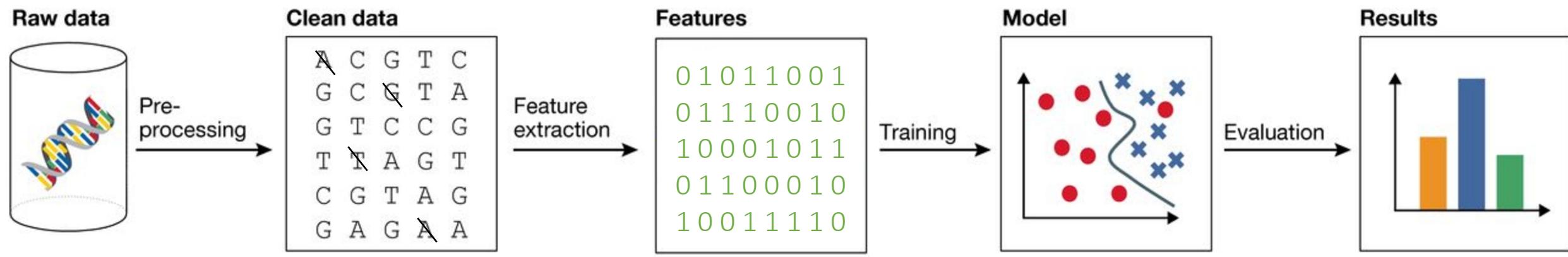
3. Phạm Nguyễn Công Danh - 19521324

<> Github: <https://github.com/namt9/CS114.L22.KHCL/tree/main/MACHINE%20LEARNING%20PROJECT>

TỔNG QUAN ĐỀ TÀI

Bài toán nhận diện chữ cái viết tay mang đến một giá trị rất lớn trong cuộc sống công nghệ thông tin hiện nay. Đã có rất nhiều nghiên cứu về đề tài này trên thế giới, thế nhưng đó là đối với bảng chữ cái la-tin tiếng Anh. Còn số lượng nghiên cứu tương tự về tiếng Việt thì còn khá hạn chế. Vì thế, nhóm quyết định thử nghiên cứu đề tài này cho chủ đề cuối kì môn học.

Quy trình tổng quan hiện thực bài toán nhận dạng chữ viết tay Tiếng Việt



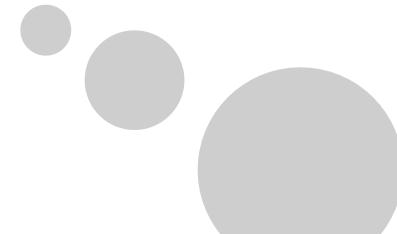
MÔ TẢ ĐỀ TÀI



* Đề tài “Nhận dạng ký tự viết tay Tiếng Việt”:



- Bài toán thuộc dạng bài toán Classification
- INPUT: ảnh bất kì một chữ cái Tiếng Việt viết tay đúng chuẩn.
- OUTPUT: kết quả chữ cái tương ứng có trong tấm ảnh đó.



MÔ TẢ BỘ DATASET



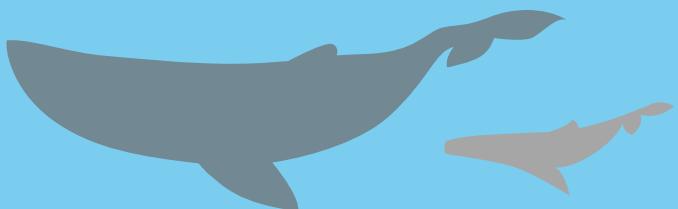
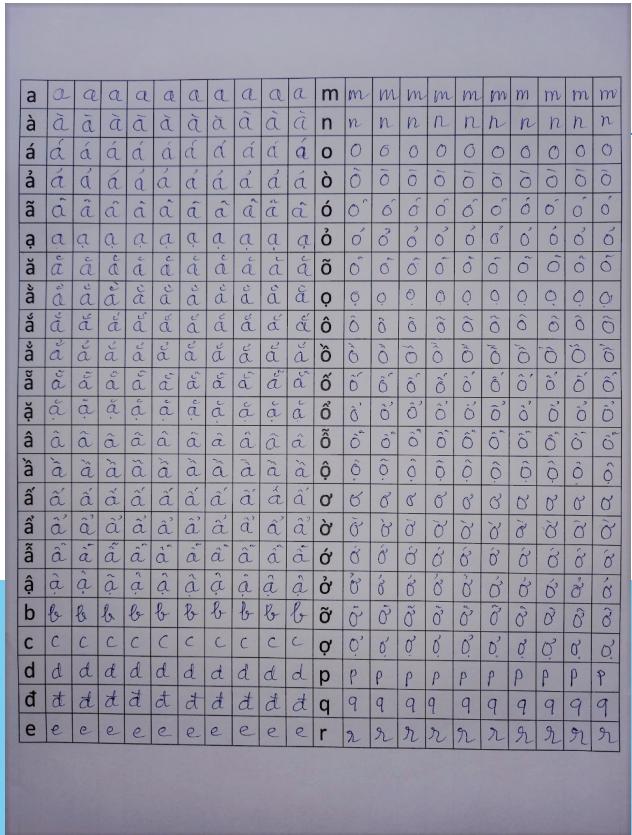
- Bảng chữ cái Tiếng Việt bao gồm 29 chữ cái:
 - + 22 chữ cái la-tin (a, b, c, d, e, g, h, i, k, l, m, n, o, p, q, r, s, t, u, v, x, y)
 - + 7 chữ cái biến thể bằng cách thêm “dấu” (ă, â, đ, ê, ô, ơ, ư)
 - Ngoài ra, còn có các “dấu”: **ngang, sắc, huyền, hỏi, ngã, nặng** sẽ kết hợp được cùng với a, ă, â, đ, e, ê, i, o, ô, ơ, u và ư
 - Tổng cộng sẽ có **89 kí tự** cần thu thập.

Nhóm quyết định sẽ tiến hành thu thập data từ người quen và bạn bè theo mẫu sau, mỗi người sẽ chỉ viết 1 bản để đảm bảo tính chất của bộ dữ liệu. Bên cạnh đó, để làm giàu thêm bộ data, nhóm quyết định sẽ gộp chung bộ data huấn luyện cùng với nhóm của bạn *Trần Vĩ Hào*.

Sau đó, các mẫu sẽ được gom lại và chụp hình để tiến hành các bước xử lý dữ liệu tiếp theo.



1. THU THẬP DỮ LIỆU



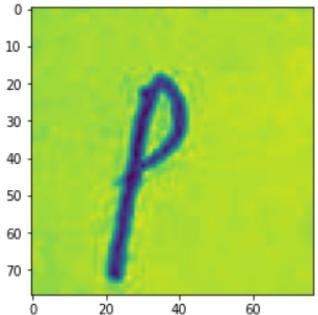
**Đầu tiên ảnh sẽ
được code xử lí
bằng cách cắt bỏ
vùng trắng dư
thừa ở 4 bên**

Sau cùng, nhóm sẽ tiến hành việc chọn lọc và di chuyển ảnh sang từng thư mục tương ứng với từng chữ cái trong ảnh

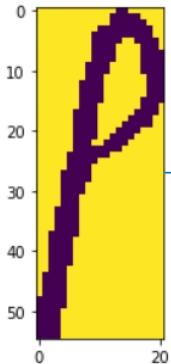
- Sau đó ảnh sẽ
được cắt thành
từng ô một chứa
các chữ và lưu trữ lại

P	226.jpg	tōi
P	225.jpg	tōi
P	224.jpg	tōi
P	223.jpg	tōi
P	222.jpg	tōi
P	221.jpg	tōi
P	220.jpg	tōi
P	219.jpg	tōi
P	218.jpg	tōi
P	217.jpg	tōi
P	216.jpg	tōi
P	215.jpg	tōi

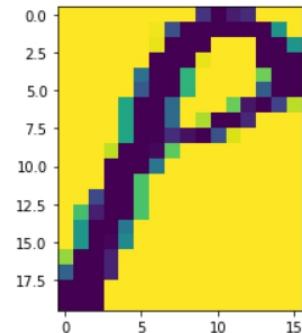
2. QUÁ TRÌNH XỬ LÍ DỮ LIỆU



Hình ban đầu
sẽ chứa khoảng
trắng khá lớn
xung quanh chữ,
nhóm đã thực hiện
crop vào sát chữ

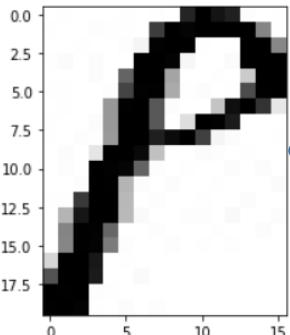


Để đồng nhất
bộ data, nhóm
đã resize tất cả
về size (16x20)



```
255 254 255 255 255 254 255 254 255 36 0 28 32 253 255 254  
254 255 255 253 254 254 255 255 255 61 0 4 0 0 0 0 0 133 254  
255 254 255 255 255 255 255 245 2 0 0 254 255 255 0 0 135  
254 255 255 255 255 255 255 0 0 0 3 247 255 255 255 0 0 0  
255 255 255 255 255 100 0 0 164 255 255 254 255 283 0 0  
254 254 255 255 253 68 2 1 170 254 252 255 254 76 1 0  
255 254 255 255 152 1 0 88 252 255 255 196 15 0 51 231  
254 255 255 252 152 0 0 88 255 222 0 2 27 255 255 255 255  
254 255 253 154 0 0 31 11 0 64 245 252 255 255 255 255  
255 252 255 229 0 0 9 193 255 255 252 253 253 254 255 255  
255 253 255 0 2 5 99 253 255 254 255 255 255 255 255 255  
255 250 253 0 1 180 252 255 252 255 255 255 255 255 255 255  
254 254 57 0 0 184 255 253 255 255 254 255 255 255 255 255  
255 180 22 0 0 185 258 255 255 253 253 255 249 255 255 255  
255 132 1 2 182 255 255 255 255 255 254 255 255 255 255  
255 134 1 3 123 253 255 255 253 255 254 255 255 255 255  
214 0 0 24 253 254 255 255 255 255 255 255 255 255 255  
83 0 0 27 255 255 256 255 255 255 255 255 255 255 255  
0 1 0 253 254 255 255 255 255 255 255 255 255 255  
1 0 3 254 255 252 253 253 255 255 255 255 255 255 255
```

Chuyển tất cả hình
binary (16x20) về
ma trận (320,) và
lưu tạo thành 1
file .csv



Ảnh sau cùng trong
bộ dataset sau khi
đã bitwise để đảo
ngược màu
(chữ đen, nền trắng)

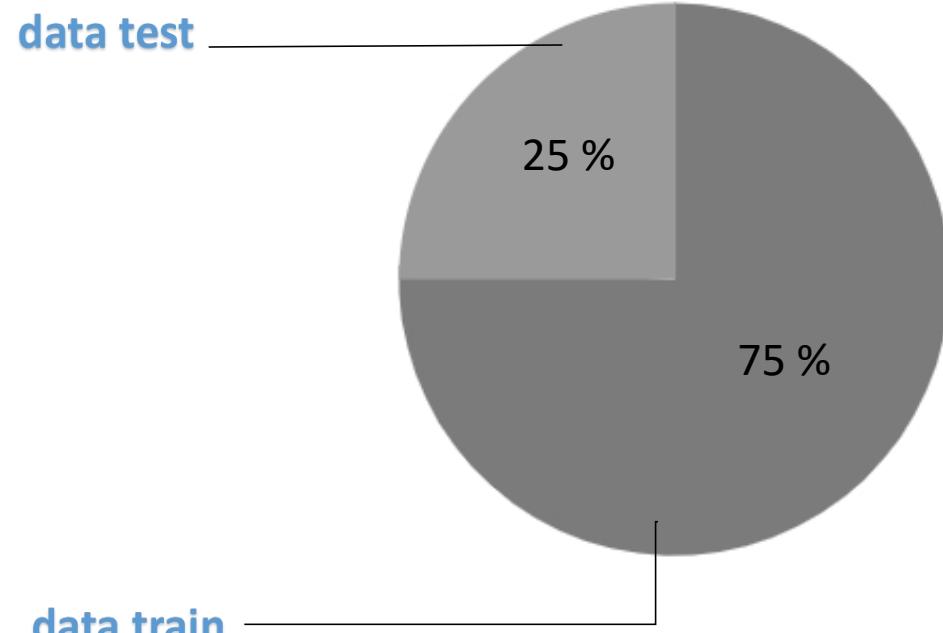
Tất cả gồm có **24.229** arrays đại
diện cho **24.229** hình ảnh được
tổng hợp lại và tách làm 2 file .csv



DataTrain.csv (19229 arrays) trung bình 216 ảnh / chữ
DataTest.csv (5000 arrays) trung bình 55 ảnh / chữ



3. PHÂN CHIA BỘ DATASET



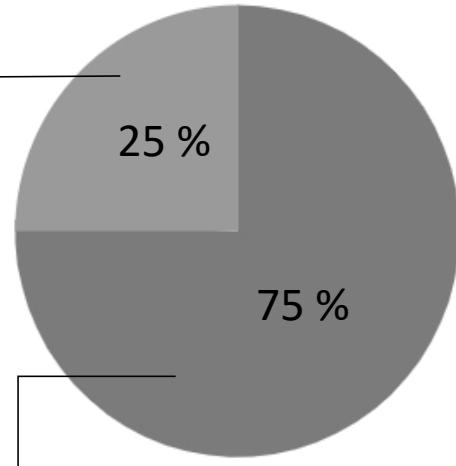
Vì kích thước bộ Dataset nhóm tự thu thập chưa được lớn, nên việc phân chia thành 3 tập Train, Test và Valid là không hợp lý.

Nhóm quyết định chỉ chia thành 2 tập cơ bản là Train và Test. Sau đó, áp dụng phương pháp "**K-Fold cross validation**" để đánh giá model hiệu quả hơn khi có ít dữ liệu.



3. PHÂN CHIA BỘ DATASET

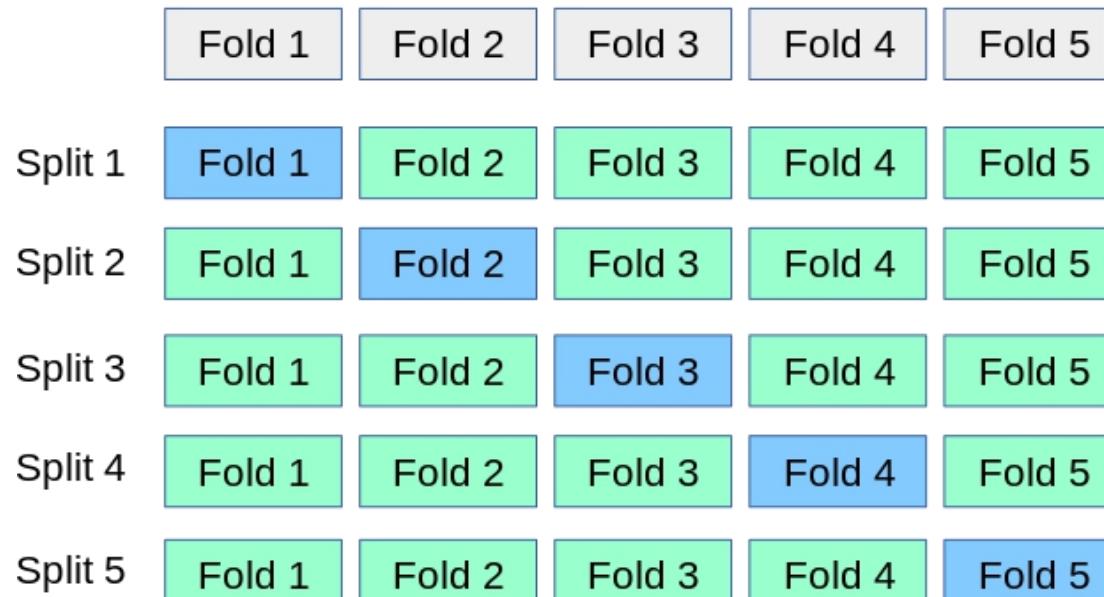
data test



data train

Với phương pháp này, mô hình sẽ học được nhiều hơn nhưng vẫn có dữ liệu để đánh giá độ hiệu quả của mô hình.

Training data



Test Fold



Train Fold





4. CÁC MÔ HÌNH NHÓM SỬ DỤNG



- | | |
|---|--|
| <p>1 SVM (kernel = 'linear')</p> <p>2 SVM (kernel = 'rbf')</p> <p>3 SVM (kernel = 'poly')</p> <p>4 Logistics Regression</p> | <p>5 Multinomial Naive Bayes</p> <p>6 Gaussian Naive Bayes</p> <p>7 Bernoulli Naive Bayes</p> <p>8 MPL Classification</p> <p>9 K-Neighbors</p> <p>10 Random Forest</p> |
|---|--|



4.1. ĐÁNH GIÁ HIỆU SUẤT VỚI BỘ DATASET THÔNG THƯỜNG (grayscale + flatten vector)

	SVM			LOGISTIC REGRESSION	NAIVE BAYES			MLP CLASSIFICATION	K-NEIGHBORS	RANDOM FOREST
MODEL \ DATA	SVM ('linear')	SVM ('rbf')	SVM ('poly')	Logistics Regression	Multi Naive Bayes	Gaussian Naive Bayes	Bernoulli Naive Bayes	MPL Classification	K-NEIGHBORS	RANDOM FOREST
train	73.6023 %	83.5301 %	78.8861 %	63.3055 %	43.9545 %	41.6298 %	28.8418 %	81.2575 %	60.8716 %	75.4070 %
test	42.7287 %	51.9218 %	46.6622 %	39.6269 %	32.7040 %	29.0178 %	19.0380 %	54.5291 %	30.5012 %	44.3470 %

Mô hình cho kết quả “học” và test tốt so với phần còn lại là SVM (kernel = ‘rbf’) và MPLClassification():

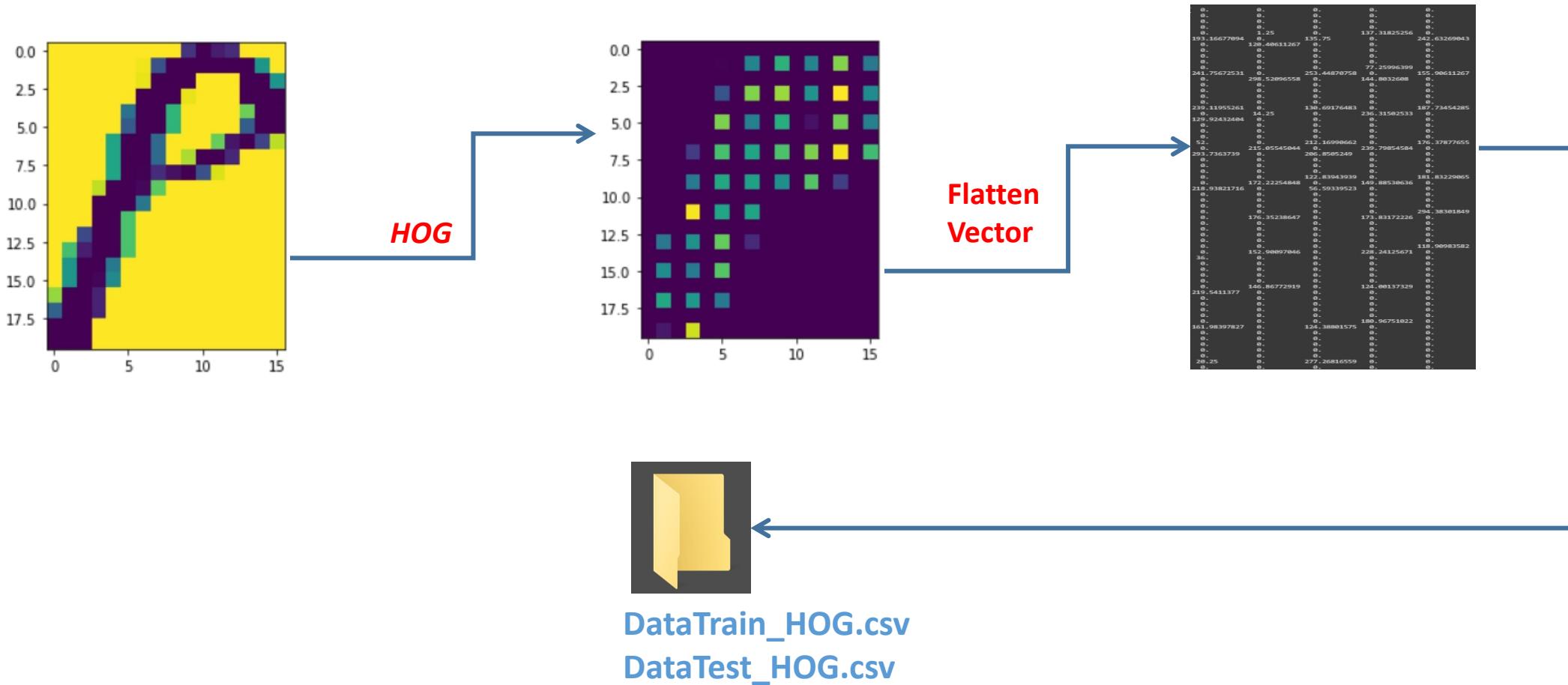
+ SVM (kernel = ‘rbf’): nhóm đã thử và chọn setup parameter **C = 7**

+ MPLClassification(): nhóm đã thử và chọn setup parameter **hidden_layer_sizes=(320, 320, 320)** và

max_iter=500



4.2. CÀI TIẾN DATA: HISTOGRAM of ORIENTED GRADIENTS (HOG) -> FLATTEN VECTOR



4.3. ĐÁNH GIÁ HIỆU SUẤT VỚI BỘ DATASET CẢI TIẾN (grayscale -> HOG -> flatten vector)

	SVM			LOGISTIC REGRESSION	NAIVE BAYES			MLP CLASSIFICATION	K-NEIGHBORS	RANDOM FOREST
MODEL DATA	SVM ('linear')	SVM ('rbf')	SVM ('poly')	Logistics Regression	Multi Naive Bayes	Gaussian Naive Bayes	Bernoulli Naive Bayes	MPL Classification	K-NEIGHBORS	RANDOM FOREST
train	71.0646 % (-~2%)	84.1178 % (-~2%)	79.8586 % (+~1%)	60.0810 % (+~3%)	44.6202 % (-~1%)	41.3541 % (-~0.5%)	7.0882 % (-~21%)	82.9736 % (+~2%)	74.0496 % (+~14%)	78.8600 % (+~3%)
test	48.2805 % (+~6%)	55.6530 % (+~6%)	53.4727 % (+~6%)	42.6837 % (+~3%)	31.0856 % (-~1%)	30.1191 % (-~1%)	4.9225 % (-~14%)	57.3163 % (+~3%)	48.6177 % (+~18%)	49.7415 % (+~5%)

Sau cải tiến bộ dataset với phương pháp HOG, có sự chuyển biến tích cực đối với hầu hết các mô hình, đặc biệt hai mô hình SVM (kernel = 'rbf') và MPLClassification() vẫn có sự gia tăng Accurary với bộ Test tốt nhất so với phần còn lại:

+ SVM (kernel = 'rbf'): nhóm đã thử và chọn setup parameter **C = 7**

+ MPLClassification(): nhóm đã thử và chọn setup parameter

hidden_layer_sizes=(320, 320, 320) và **max_iter=500**



NHẬN XÉT

1

Các Model sử dụng ít nhiều đều bị overfitting hoặc underfitting

2

2 model cho kết quả học tốt nhất và accuracy cao ở bộ test là SVM (kernel = 'rbf') và MPLClassification()

NGUYÊN NHÂN:

- + Dữ liệu đào tạo quá ít, trung bình 170 hình cho mỗi class. Nguyên nhân là do quá trình scan hình ảnh và cắt hình chưa được tốt, dẫn đến mất mát dữ liệu nhiều.
- + Số lượng class trong bài toán là quá lớn (89 classes) dẫn đến những mô hình máy học đơn giản áp dụng không có hiệu quả cao.
- + Sự điều chỉnh các parameter trong các model là chưa phù hợp nhất dẫn đến hiệu quả của model chưa được tối ưu.

HƯỚNG CẢI THIỆN:

- Tìm hiểu cách scan ảnh để ảnh thu lại được nhiều hơn.
- Tăng gấp 100 lần số lượng ảnh hiện tại.
- Chọn model có độ phức tạp cao hơn và chuyên dụng hơn dành cho bài toán xử lý hình ảnh đa class.
- Tìm hiểu thêm các phương pháp để cải thiện chất lượng data, rút trích đặc trưng tốt hơn và tiền xử lý data hiệu quả hơn.
- Hy vọng sẽ tránh được underfit và overfit cho bài toán và tăng hiệu quả training và độ chính xác cho đầu ra.

CẢM ƠN THẦY ĐÃ NHẬN XÉT