



**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
ĐẠI HỌC QUỐC GIA TP HCM**



# **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

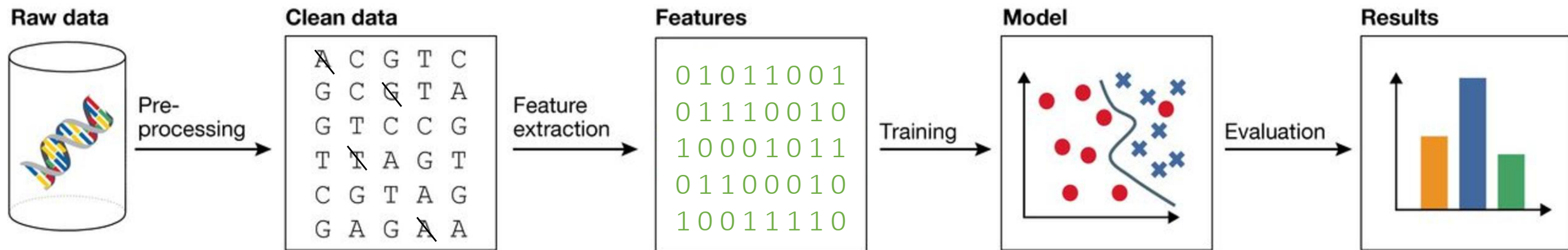
3. Phạm Nguyễn Công Danh - 19521324

<> **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

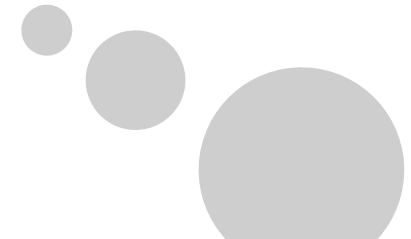


# MÔ TẢ ĐỀ TÀI



\* Đề tài “Nhận dạng kí 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.



- 

- Tổng cộng sẽ có **89 kí tự** cần thu thập.

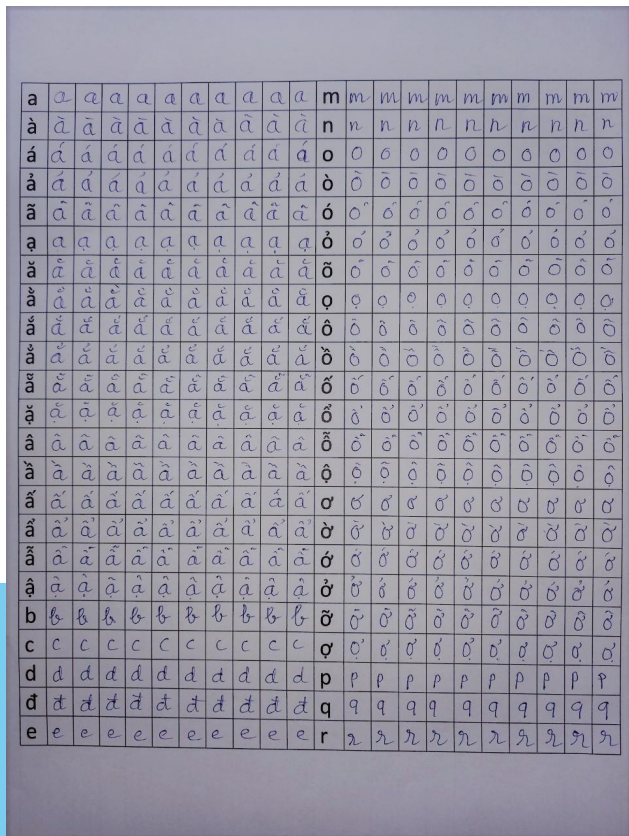
Sau đó, các mẫu sẽ được gom lại và chụp hình để tiến hành các bước xử lý dữ liệu tiếp theo.

[illegible]

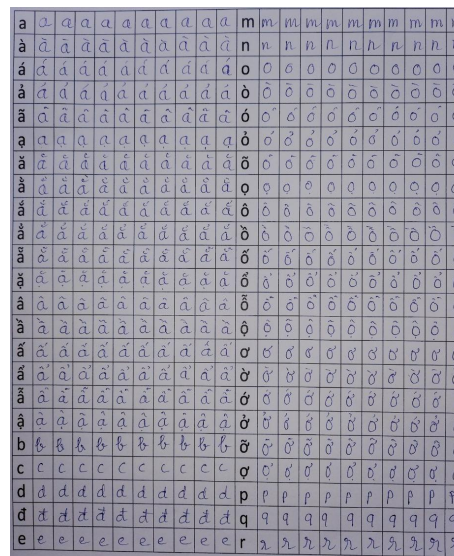




# 1. THU THẬP DỮ LIỆU



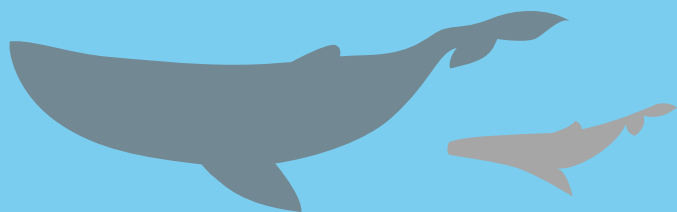
Đầu tiên ảnh sẽ được code xử lí bằng cách cắt bỏ vùng trắng dư thừa ở 4 bên



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

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

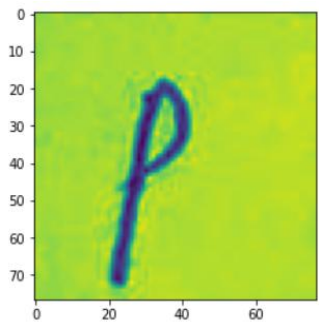
	226.jpg	tôi
	225.jpg	tôi
	224.jpg	tôi
	223.jpg	tôi
	222.jpg	tôi
	221.jpg	tôi
	220.jpg	tôi
	219.jpg	tôi
	218.jpg	tôi
	217.jpg	tôi
	216.jpg	tôi
	215.jpg	tôi



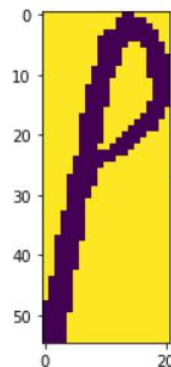


## 2. QUÁ TRÌNH LỌC VÀ XỬ LÝ DỮ LIỆU

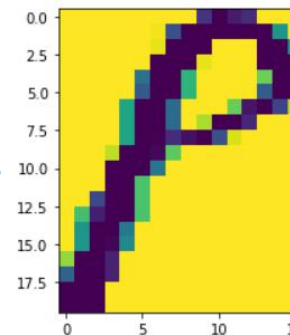
### 2.1. Xử lý dữ liệu



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



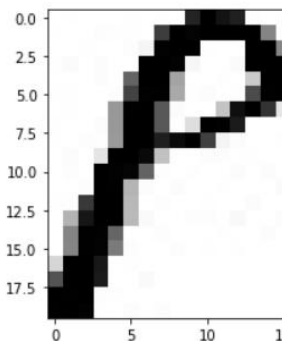
Để đồng nhất bộ data, nhóm đã resize tất cả về size (16x20)



```
255 254 255 255 255 254 255 254 255 36 0 20 32 253 255 254
254 255 255 253 254 254 255 61 0 4 0 0 0 0 133 254
255 255 254 255 255 253 245 2 0 0 0 254 255 255 0 135
254 251 255 252 255 255 0 0 3 247 255 251 0 0 0
255 255 255 252 255 100 0 0 164 255 255 254 255 203 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
254 255 253 254 154 0 0 31 11 0 64 245 252 255 255 254
255 252 255 229 0 0 9 193 255 255 252 253 255 254 255 255
255 253 255 0 2 5 99 253 255 254 255 255 252 255 252 255
255 255 253 0 1 180 252 255 252 255 255 254 255 255 251
254 254 57 0 0 184 255 253 255 255 254 255 255 252 255 255
255 180 22 0 0 185 250 255 255 253 253 255 249 255 252 255
255 132 1 2 102 255 255 253 255 255 255 254 255 255 252 255
255 134 1 3 123 253 255 255 253 255 254 255 254 255 255 255
224 0 0 24 255 254 255 255 255 255 255 255 255 255 255
83 0 0 27 255 250 255 255 255 255 255 255 255 255 255
0 1 0 253 254 255 255 255 255 255 255 255 255 255 255
1 0 3 254 255 252 255 253 255 255 255 255 255 255 255
```

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

**Lọc dữ liệu**



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


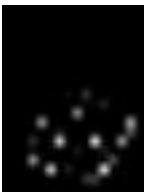





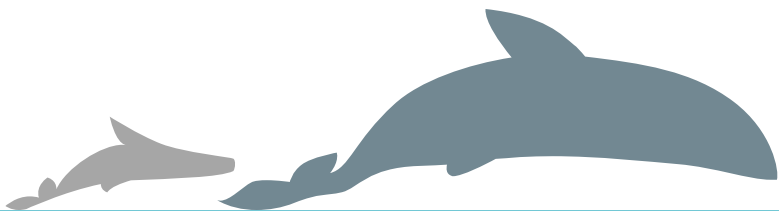
## 2. QUÁ TRÌNH LỌC VÀ XỬ LÝ DỮ LIỆU

### 2.2. Lọc dữ liệu

- Sau khi crop, resize và chuyển ảnh sang dạng “nền trắng - chữ đen”. Sẽ có những ảnh “không đạt yêu cầu”. Nhóm tiến hành lọc và xóa bỏ những ảnh này theo cách thủ công.

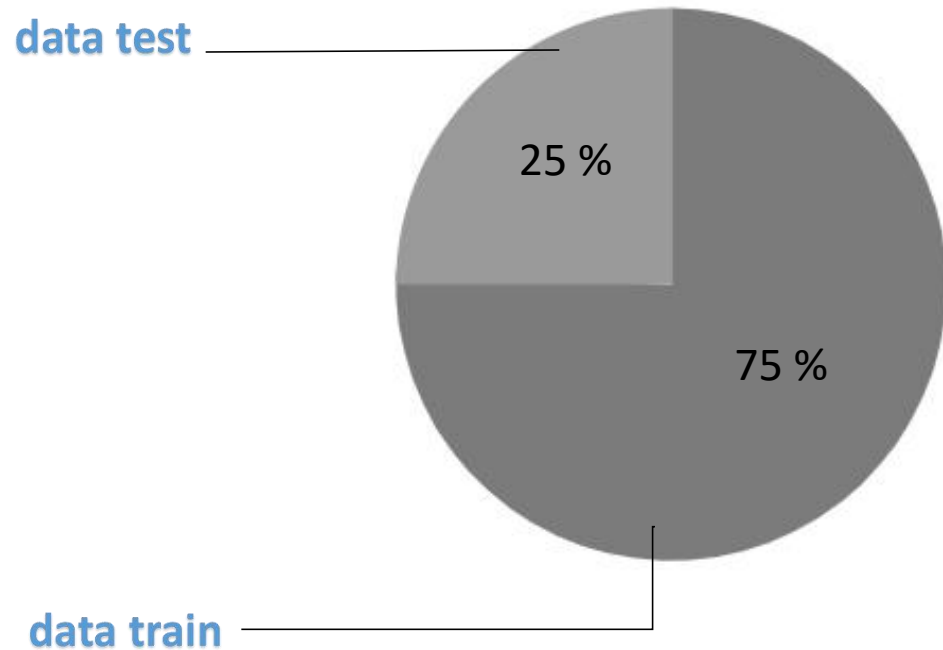
- Những ảnh bị nhiễu hạt , bị đen hình, mất nội dung  hoặc những ảnh không nhận dạng ra chữ  mà nhóm gặp phải sau quá trình xử lý sẽ bị xóa bỏ, chỉ chọn lọc những chữ rõ ràng và không bị nhiễu để xây dựng bộ dữ liệu.

- Sau đó tất cả ảnh sẽ đưa về dạng ma trận (320,) và lưu tạo thành những file datatrain và datatest .





### 3. PHÂN CHIA BỘ DATASET



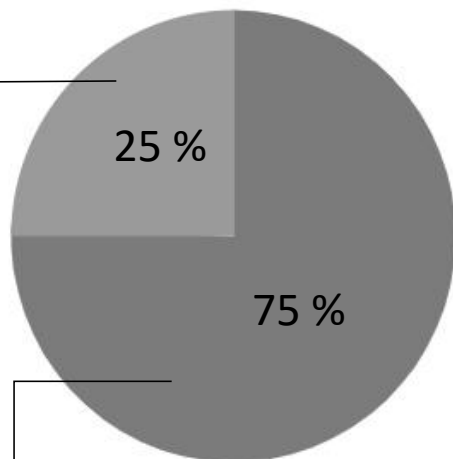
Vì kích thước bộ Dataset nhóm tự thu thập chưa được lớn, nên việc phân chia thành 3 tập Train, Test và Valid là không hợp lí.

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.



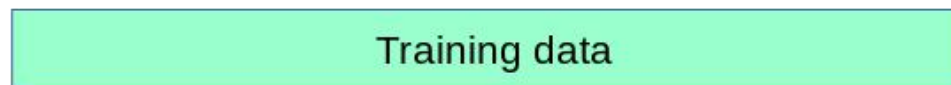
### 3. PHÂN CHIA BỘ DATASET

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.



	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

...



Test Fold

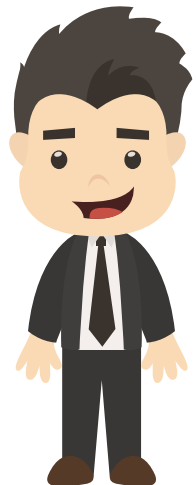


Train Fold





## 4. CÁC MÔ HÌNH NHÓM SỬ DỤNG



1

SVM (kernel = 'linear')

2

SVM (kernel = 'rbf')

3

SVM (kernel = 'poly')

4

Logistics Regression

5

Multinomial Naive Bayes

6

Gaussian Naive Bayes

7

Bernoulli Naive Bayes

8

MLP Classification

9

K-Neighbors

10

Random Forest

11

Convolutional neural network (CNN)



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

	SVM			LOGISTIC REGRESSION	NAIVE BAYES			MLP CLASSIFI CATION	K- NEIGHBORS	RANDOM FOREST
MODEL DATA	SVM (‘linear’)	SVM (‘rbf’)	SVM (‘poly’)	Logistics Regression	Multi Naive Bayes	Gaussian Naive Bayes	Bernoulli Naive Bayes	MPL Classifica tion	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**

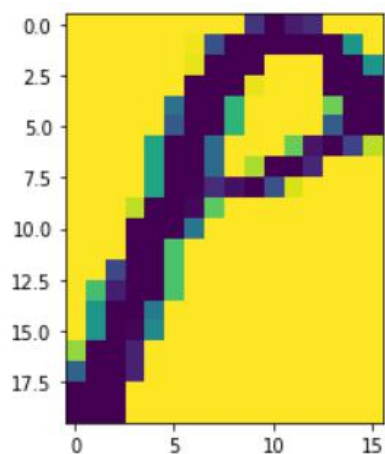
+ MPLClassification(): nhóm đã thử và chọn setup parameter **hidden\_layer\_sizes=(320, 320, 320)** và

**max\_iter=500**

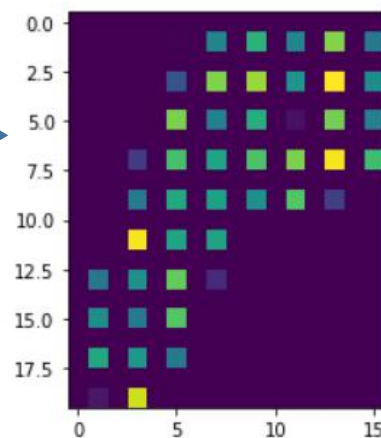




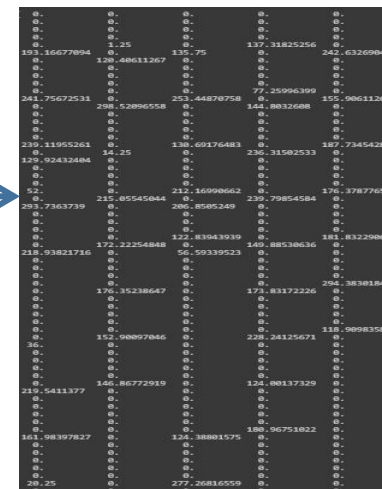
## 4.2. CẢI TIẾN DATA: HISTOGRAM of ORIENTED GRADIENTS (HOG) -> FLATTEN VECTOR



*HOG*



**Flatten  
Vector**



DataTrain\_HOG.csv  
DataTest\_HOG.csv





### 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 % <i>(-~2%)</i>	84.1178 % <i>(-~2%)</i>	79.8586 % <i>(+~1%)</i>	60.0810 % <i>(+~3%)</i>	44.6202 % <i>(-~1%)</i>	41.3541 % <i>(-~0.5%)</i>	7.0882 % <i>(-~21%)</i>	82.9736 % <i>(+~2%)</i>	74.0496 % <i>(+~14%)</i>	78.8600 % <i>(+~3%)</i>
test	48.2805 % <i>(+~6%)</i>	55.6530 % <i>(+~6%)</i>	53.4727 % <i>(+~6%)</i>	42.6837 % <i>(+~3%)</i>	31.0856 % <i>(-~1%)</i>	30.1191 % <i>(-~1%)</i>	4.9225 % <i>(-~14%)</i>	57.3163 % <i>(+~3%)</i>	48.6177 % <i>(+~18%)</i>	49.7415 % <i>(+~5%)</i>

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**



## 4.4. 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 trước 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 gia tăng hiệu quả training và độ chính xác cho đầu ra.



## 4.5. CẢI TIẾN MODEL: CONVOLUTIONAL NEURAL NETWORK (CNN) IN KERAS

*Số lượng class trong bài toán là quá lớn (89 classes) dẫn đến những mô hình máy học đơn giản trước áp dụng không có hiệu quả cao.*

*Chọn model có độ phức tạp cao hơn và chuyên dụng hơn dành cho bài toán xử lý hình ảnh đa class.*

```
model = Sequential()

model.add(Conv2D(32, (3, 2), input_shape=(20, 16, 1), activation='relu', data_format="channels_last", padding="same"))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 2), input_shape=(20, 16, 1), activation='relu', data_format="channels_last", padding="same"))
model.add(BatchNormalization())
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

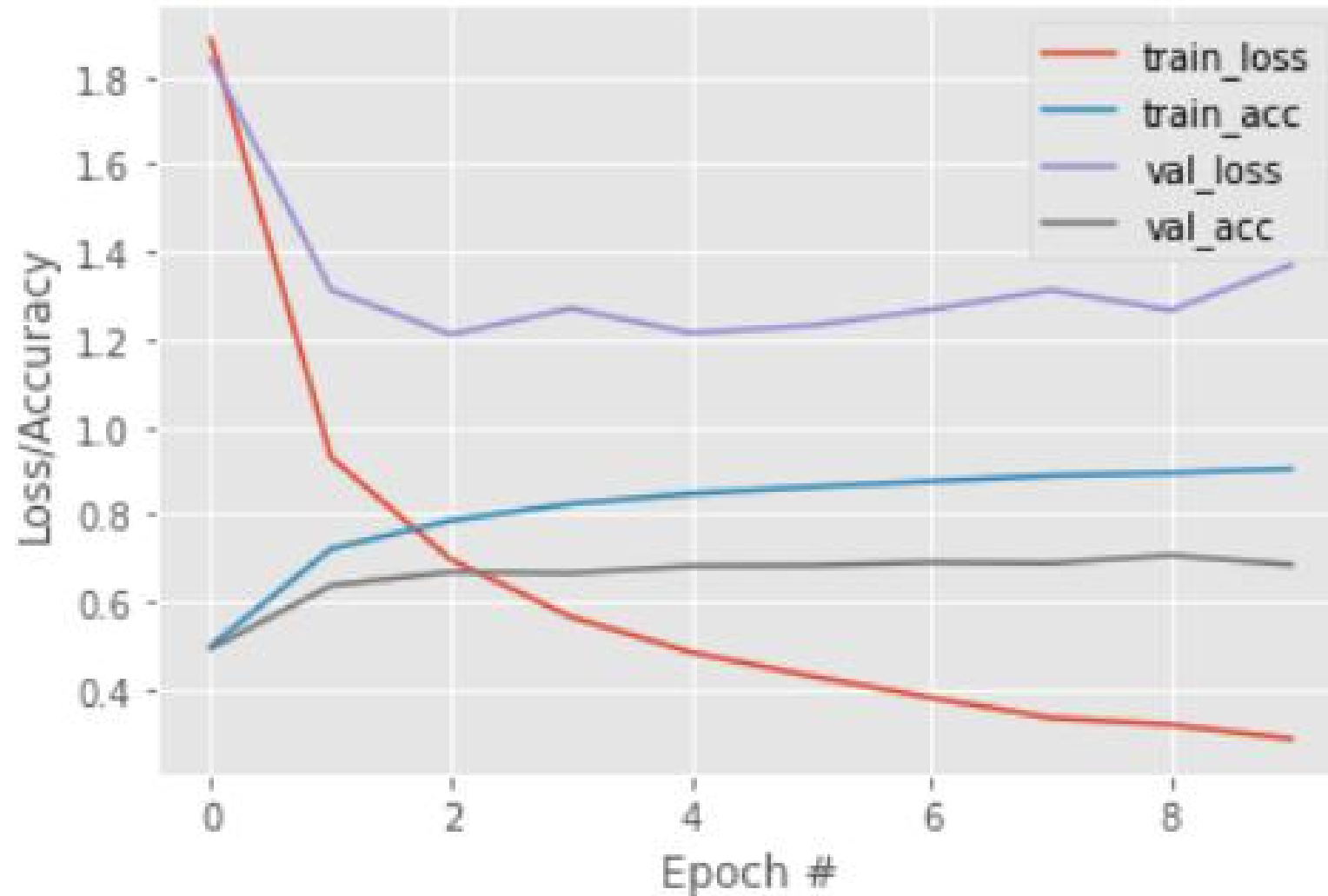
model.add(Conv2D(64, (3, 2), activation='relu', data_format="channels_last", padding="same"))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 2), activation='relu', data_format="channels_last", padding="same"))
model.add(BatchNormalization())
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax'))
```



## 4.5. CẢI TIẾN MODEL: CONVOLUTIONAL NEURAL NETWORK (CNN) IN KERAS







## 4.6. ĐÁNH GIÁ HIỆU SUẤT VỚI MODEL CNN IN KERAS

MODEL DATA	CONVOLUTIONAL NEURAL NETWORK (CNN)
train	90.36 %
test	68.44 %

Khi sử dụng model CNN trong xử lý hình ảnh cùng với việc lựa chọn các tham số phù hợp, cả kết quả thể hiện trên cả tập train và test đều **tăng** so với các model đơn giản khi trước.

Nhưng thực tế với bộ dataset chưa đủ lớn như hiện tại thì loss trên cả tập train và test vẫn còn cao.

	precision	recall	f1-score	support
a	0.94	1.00	0.97	46
à	0.78	0.91	0.84	43
á	0.68	0.60	0.64	57
â	0.50	0.81	0.62	31
ã	0.62	0.82	0.70	38
ä	0.42	0.58	0.49	36
å	0.42	0.55	0.48	38
ä	0.42	0.54	0.47	39
ă	0.52	0.58	0.55	45
ā	0.58	0.72	0.64	40
ā	0.72	0.69	0.71	52
ā	0.50	0.57	0.53	44
ā	0.48	0.73	0.58	33
ā	0.62	0.53	0.57	59
ā	0.82	0.77	0.80	53
ą	0.88	0.86	0.87	51
ā	0.44	0.81	0.57	27
ā	0.72	0.77	0.74	47
b	0.84	0.88	0.86	48
c	0.94	0.92	0.93	51
d	0.72	0.84	0.77	43
e	0.82	0.95	0.88	43
è	0.66	0.69	0.67	48
é	0.82	0.64	0.72	64
è	0.86	0.43	0.58	99
è	0.56	0.51	0.53	55
ë	0.44	0.30	0.36	73
ë	0.36	0.58	0.44	31
ë	0.50	0.57	0.53	44
ë	0.54	0.44	0.49	61
è	0.66	0.60	0.63	55
ę	0.94	0.76	0.84	62
ê	0.74	0.93	0.82	40

g	0.82	0.93	0.87	44
h	0.44	0.96	0.60	23
i	0.54	0.71	0.61	38
ì	0.82	0.57	0.67	72
í	0.68	0.62	0.65	55
ï	0.80	0.89	0.84	45
ï	0.68	0.63	0.65	54
ï	0.58	0.48	0.53	60
k	0.78	0.57	0.66	68
l	0.72	0.51	0.60	71
m	0.82	0.95	0.88	43
n	0.80	0.78	0.79	51
o	0.98	0.89	0.93	55
ò	0.88	0.75	0.81	59
ó	0.46	0.70	0.55	33
ô	0.74	0.67	0.70	55
ð	0.62	0.65	0.63	48
õ	0.76	0.40	0.52	95
õ	0.58	0.63	0.60	46
õ	0.86	0.61	0.72	70
õ	0.66	0.72	0.69	46
ó	0.76	0.50	0.60	76
σ	0.56	0.72	0.63	39
ờ	0.70	0.61	0.65	57
ớ	0.68	0.55	0.61	62
õ	0.66	0.65	0.65	51
ở	0.46	0.57	0.51	40
ợ	0.68	0.74	0.71	46
ọ	0.92	0.88	0.90	52
ộ	0.76	0.59	0.67	64
p	0.98	0.88	0.92	56
q	0.90	0.92	0.91	49
r	0.76	0.78	0.77	49
s	0.64	0.57	0.60	56
t	0.80	0.85	0.82	47
u	0.84	0.93	0.88	45
ù	0.58	1.00	0.73	29
ú	0.46	0.74	0.57	31

ũ	0.66	0.80	0.73	41
ű	0.62	0.62	0.62	50
ur	0.72	0.78	0.75	46
űr	0.70	0.73	0.71	48
úr	0.46	0.47	0.46	49
űr	0.66	0.56	0.61	59
űr	0.66	0.51	0.57	65
ұ	0.84	0.95	0.89	44
ұ	0.80	0.80	0.80	50
v	0.78	0.87	0.82	45
x	0.92	0.82	0.87	56
y	0.76	0.58	0.66	65
ÿ	0.76	0.69	0.72	55
ý	0.74	0.73	0.73	51
ÿ	0.56	0.61	0.58	46
ÿ	0.54	0.79	0.64	34
ÿ	0.40	0.87	0.55	23
đ	0.72	0.78	0.75	46



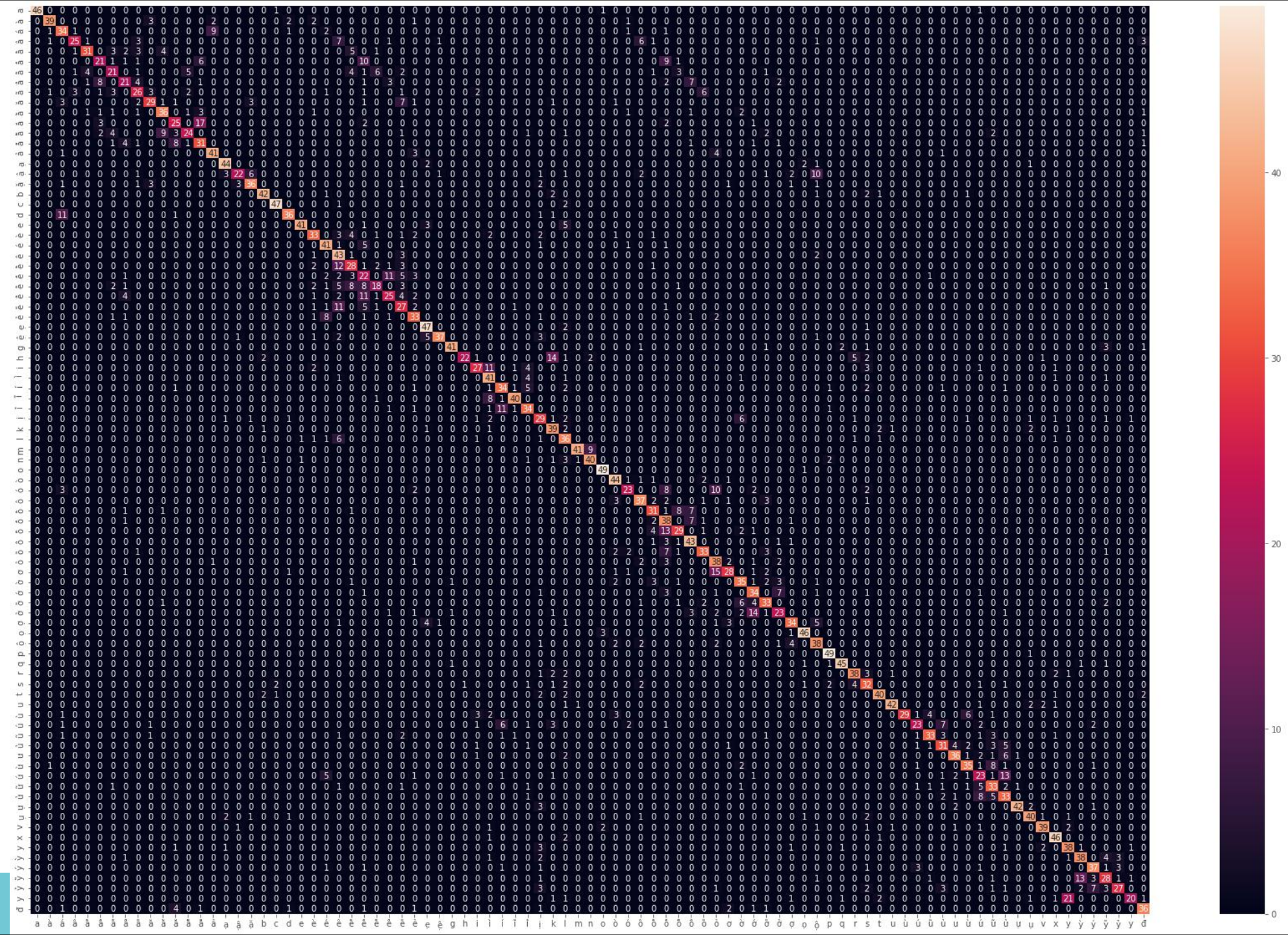


4.6.

CNN

CONFUSION

MATRIX





## 4.7. TỔNG KẾT

### \* Đối với bộ Dataset:

- Nhiệm vụ cần làm giàu thêm bộ dataset với đa dạng dữ liệu hơn để cải thiện hiệu suất của bài toán.
- Vẫn phải tìm cách thu thập và xử lí data hiệu quả hơn để thu lại được 1 bộ dataset chất lượng hơn.

### \* Đối với Model áp dụng:

- Tuy sau khi sử dụng CNN thì kết quả cải thiện tốt hơn trước rất nhiều, nhưng so với yêu cầu thực tế thì độ chính xác trên vẫn chưa thể chấp nhận được.
- Tìm hiểu thêm các model, tìm hiểu kĩ về cách thiết lập các parameter trong model đó để tăng hiệu suất bài toán.



## 5. HƯỚNG PHÁT TRIỂN

- + Nhận diện từ - câu - đoạn văn bản viết tay Tiếng Việt
- + Phát hiện nội dung dạng chữ trong bảng hiệu, trang sách

## 6. NGUỒN THAM KHẢO

Cut Image: <https://stackoverflow.com/questions/59182827/how-to-get-the-cells-of-a-sudoku-grid-with-opencv>

HOG: <https://phamdinhhkhanh.github.io/2019/11/22/HOG.html>

CNN: <https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2>

<https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5>

Crop to Image: [https://github.com/nivla0607/CS114.L22.KHCL/blob/main/Hand\\_writting\\_digit\\_recognition.ipynb](https://github.com/nivla0607/CS114.L22.KHCL/blob/main/Hand_writting_digit_recognition.ipynb)

K-Fold: <https://www.miai.vn/2021/01/18/k-fold-cross-validation-tuyet-chieu-train-khi-it-du-lieu/>





**CẢM ƠN THẦY ĐÃ NHẬN XÉT**