BÁO CÁO ĐỒ ÁN CUỐI KỲ

Lớp: CS114.L11.KHCL
Môn: MÁY HỌC

GV: PGS.TS Lê Đình Duy - THS. Phạm Nguyễn Trường An
Trường ĐH Công Nghệ Thông Tin, ĐHQG-HCM

PHÂN LOẠI KÝ TỰ VIẾT TAY TIẾNG VIỆT

Phan Nguyễn Thành Nhân - 19521943 - CS114.L11.KHCL

Lê Quang Huy - 19521617 - CS114.L11.KHCL

Nguyễn Thành Nghĩa - 19521899 - CS114.L11.KHCL

Link Github: https://github.com/thanhnhan311201/CS114.L11.KHCL

Ảnh các thành viên trong nhóm

Bài toán

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.

Bài toán

- Tên đề tài: Phân loại ký tự viết tay tiếng Việt
- Input: Một hình ảnh chứa duy nhất một ký tự tiếng Việt (có thể bao gồm cả các dấu sắc, huyền, hỏi, ngã, nặng)
- Output: Xuất ra chữ cái tương ứng với hình ảnh

Input

Output

ê

Mô tả dữ liệu

- Dataset gồm tổng cộng hơn 17.000 mẫu cho 89 class. Mỗi mẫu chứa một ký tự tiếng Việt viết tay, thu thập từ các sinh viên Đại học Công nghệ Thông tin và Đại học Bách Khoa
- Với 89 class, mỗi class có khoảng hơn 150 mẫu đã được nhóm gán nhãn
- Dataset được chia ngẫu nhiên dùng cho tập training và tập validation với tỷ lệ 8:2
- Dữ liệu dùng cho tập test bao gồm hơn 2700 mẫu được thu thập sau, hoàn toàn riêng biệt với Dataset

Xử lý dữ liệu

- **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.

- **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

Chọn mô hình cho bài toán

Các model được sử dụng:

+ Logistic Regression

+ SVM (linear and non-linear)

Đánh giá hiệu suất

	accuracy	precision	recall	f1-score
Logisitc regression	0.1	0.1	0.2	0.1
Naive Bayes	0.05	0.13	0.05	0.06
Linear SVM	0.09	0.18	0.08	0.1
SVM	0.10	0.25	0.1	0.11

Bảng so sánh model với dữ liệu tập test được xử lý grayscale + flatten vector

	accuracy	precision	recall	f1-score
Logisitc regression	0.14	0.15	0.13	0.13
Naive Bayes	0.05	0.07	0.05	0.05
Linear SVM	0.14	0.15	0.14	0.13
SVM	0.19	0.15	0.14	0.13

Bảng so sánh model với dữ liệu tập test được xử lý grayscale + HOG

Kết quả cho thấy sử dụng dữ liệu được xử lý bằng phương pháp HOG cho kết quả cao hơn, tuy nhiên độ chính xác của model còn thấp (Accuracy tốt nhất là 19% với Non-Linear SVM) .

Cải tiến lần 1: Sử dụng model MLPClassifier

Kết quả của model MLPClassifier khi dự đoán cho dữ liệu được xử lý grayscale + flatten vector.

```
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
```

```
print(accuracy_score(y_test, MLP2.predict(x_test)))
```

```
0.22008701957940538
```

Độ chính xác đã tăng lên nhưng vẫn còn một số class chưa nhận diện được, đa phần ở các chữ cái có dấu của ‘u’ và ‘e’

70	0.00	0.00	0.00	30
71	0.00	0.00	0.00	30
72	0.00	0.00	0.00	30
73	0.00	0.00	0.00	30
74	0.00	0.00	0.00	30
75	0.00	0.00	0.00	30
76	0.00	0.00	0.00	30

Cải tiến lần 1: Sử dụng model MLPClassifier

Độ chính xác khi MLPClassifier dự đoán trên tập test của dữ liệu được xử lý grayscale + HOG

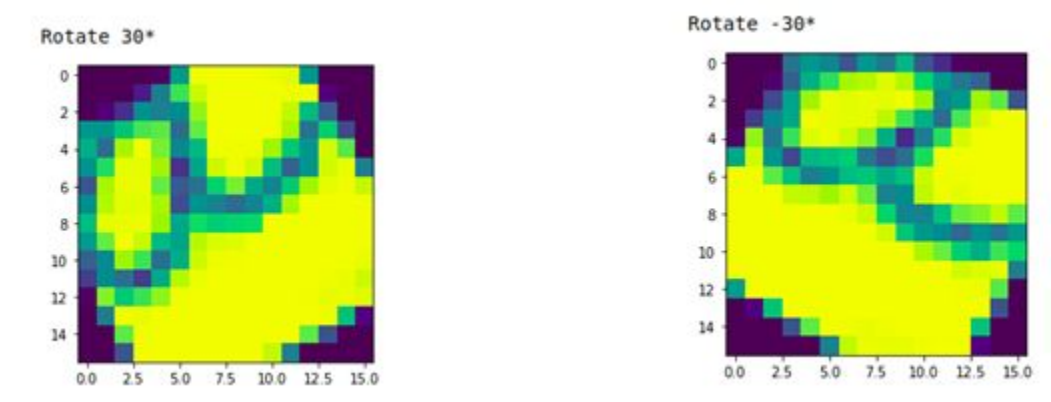
accuracy			0.24	2758
macro avg	0.27	0.24	0.24	2758
weighted avg	0.27	0.24	0.24	2758

Các class không nhận diện được đã giảm xuống. Độ chính xác tăng lên tuy nhiên không nhiều

u	0.25	0.03	0.06	30
ù	0.00	0.00	0.00	30
ú	0.22	0.07	0.10	30
ũ	0.00	0.00	0.00	30
ủ	0.25	0.17	0.20	30
ư	0.11	0.03	0.05	30

Cải tiến lần 2: Augment data

- Bởi vì dữ liệu trong tập train vẫn còn ít, nhóm quyết định tăng thêm dữ liệu bằng cách xoay con chữ theo một góc 30 độ và xử lý grayscale + hog



Cải tiến lần 2: Augment data

```
MLP2_hog = MLPClassifier(hidden_layer_sizes=(1000, 1000, 1000), max_iter=500)
MLP2_hog.fit(X_train, Y_train)
print(accuracy_score(Y_vali, MLP2_hog.predict(X_vali)))
```

```
0.49631367292225204
```

Kết quả của model MLPClassifier khi dự đoán cho dữ liệu được agument + grayscale + hog.

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

Độ chính xác khi MLPClassifier dự đoán trên tập test của dữ liệu agument + grayscale + HOG

=>Kết quả tốt nhất của là accuracy 26% với bộ dữ liệu trainning gồm 47460 ảnh được tăng thêm bằng augment data.

Kết quả và đánh giá

Các model sử dụng đã bị underfitting. Nguyên nhân do một số hạn chế sau:

- Quá ít dữ liệu, trung bình mỗi class trước khi augment data là 130 ảnh cho máy học, con số này khi 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 model máy học đơn giản không phù hợp.
- Hiểu biết về model là chưa đủ nhiều, chưa biết chọn các layer phù hợp cho model MLP classifier .
- Bộ dữ liệu train bị nhiễu rất nhiều do scan ảnh không tốt.

Phương hướng cải thiện

- 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 model tốt hơn, cách tuning hyper-paramerter cho các model.
- Tìm kiếm thêm các phương pháp xử lý, rút trích đặc trưng cho ảnh.
- Tiền xử lý dữ liệu tốt hơn, thay đổi cách thu thập dữ liệu.