**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**Nhập môn thị giác máy tính**

**BÁO CÁO ĐỒ ÁN**

**BÀI TOÁN: NHẬN DIỆN CHỮ SỐ TỪ 0-9 TRONG NGÔN NGỮ KÝ HIỆU ASL**

**Lớp:** CS231.P11

**GVHD:** Mai Tiến Dũng

**Nhóm sinh viên thực hiện:**

1. Trương Huỳnh Thúy An MSSV: 22520033

2. Trịnh Thị Lan Anh MSSV: 22520083

3. Vương Dương Thái Hà MSSV: 22520375

Tp. Hồ Chí Minh, 12/2024

Mục lục

[I. Lý do chọn đề tài 1](#_Toc186194919)

[II. Phát biểu bài toán 1](#_Toc186194920)

[1. Input: 1](#_Toc186194921)

[2. Output: 1](#_Toc186194922)

[III. Phương pháp thực hiện 2](#_Toc186194923)

[1. Mô hình Support Vector Machine (SVM) 2](#_Toc186194924)

[a. Định nghĩa thuật toán SVM 2](#_Toc186194925)

[b. Áp dụng thuật toán SVM 3](#_Toc186194926)

[2. Bộ lọc Prewitt 4](#_Toc186194927)

[a. Định nghĩa bộ lọc Prewitt 4](#_Toc186194928)

[b. Áp dụng bộ lọc Prewitt 4](#_Toc186194929)

[3. Bộ lọc Sobel 5](#_Toc186194930)

[a. Định nghĩa bộ lọc Sobel 5](#_Toc186194931)

[b. Áp dụng bộ lọc Sobel 7](#_Toc186194932)

[4. Bộ lọc HOG 7](#_Toc186194933)

[a. Định nghĩa bộ lọc HOG 7](#_Toc186194934)

[b. Áp dụng bộ lọc HOG 11](#_Toc186194935)

[IV. Thực nghiệm 12](#_Toc186194936)

[1. Dataset 12](#_Toc186194937)

[2. Độ đo đánh giá 14](#_Toc186194938)

[a. Accuracy 14](#_Toc186194939)

[b. Precision 14](#_Toc186194940)

[c. Recall 14](#_Toc186194941)

[d. F1 score 15](#_Toc186194942)

[3. Kết quả 15](#_Toc186194943)

[a. Classification Report: 15](#_Toc186194944)

[b. Confusion Matrix và Accuracy 19](#_Toc186194945)

[c. Tăng cường dữ liệu với đặc trưng có kết quả thực nghiệm cao nhất 20](#_Toc186194946)

[V. Kết luận 21](#_Toc186194947)

[VI. Hướng phát triển đề tài 21](#_Toc186194948)

[VII. Tài liệu tham khảo 22](#_Toc186194949)

# **I. Lý do chọn đề tài**

Ngôn ngữ ký hiệu đóng vai trò quan trọng trong việc xây dựng cầu nối giao tiếp và hỗ trợ người khiếm thính hòa nhập tốt hơn vào cộng đồng. Ngôn ngữ ký hiệu Mỹ (American Sign Language - ASL) không chỉ là ngôn ngữ chính của hàng triệu người khiếm thính tại Hoa Kỳ và Canada mà còn ngày càng phổ biến trên các nền tảng mạng xã hội và được học rộng rãi như một ngôn ngữ thứ hai. Cùng với sự bùng nổ của Trí tuệ nhân tạo (Artificial Intelligence - AI) và các công nghệ xử lý ảnh đã mở ra những cơ hội mới để cải thiện khả năng giao tiếp cho người khiếm thính. Điều này đặt ra nhu cầu cấp thiết về việc nghiên cứu và phát triển các ứng dụng hỗ trợ nhận dạng và sử dụng ngôn ngữ ký hiệu một cách hiệu quả.

# **II. Phát biểu bài toán**

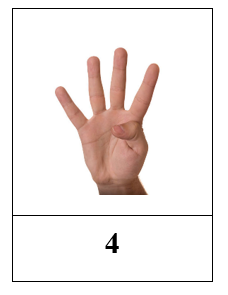
## **1. Input:**

Gồm 2 phần:

*- Bộ dữ liệu (dataset):* bao gồm các hình ảnh chụp bàn tay thực hiện các ký hiệu số từ 0 đến 9 theo ngôn ngữ ký hiệu ASL. Gồm hình ảnh và nhãn:

Hình ảnh: Chứa duy nhất một bàn tay ở chính giữa khung hình, bàn tay đầy đủ 5 ngón bình thường, không dị tật (không thừa ngón, thiếu ngón, các ngón tay dính lại vào nhau, không bị cụt mất bất kỳ ngón tay nào). Ký hiệu số từ 0 đến 5: lòng bàn tay hướng vào trong, đầu ngón tay hướng lên trên; ký hiệu số từ 6 đến 9: lòng bàn tay hướng ra ngoài, đầu ngón tay hướng lên trên. Kích thước có thể khác nhau nhưng phải giữ tỷ lệ hợp lý.

Nhãn (Label): Mỗi ảnh được gán nhãn tương ứng với số nguyên từ 0 đến 9, đại diện cho ký hiệu ngôn ngữ ký hiệu ASL trong ảnh. Nhãn được tổ chức dưới dạng cấu trúc thư mục: mỗi thư mục đại diện cho một số từ 0 đến 9, chứa các ảnh tương ứng (ví dụ: thư mục 0 chứa ảnh biểu diễn số 0).

*- Ảnh:* hình ảnh tương tự về yêu cầu với bộ dữ liệu ở trên nhưng không được gán nhãn.

## **2. Output:**

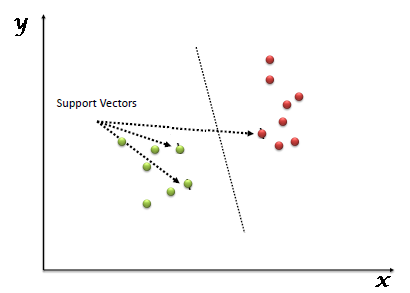
Một số nguyên trong khoảng từ 0 đến 9, biểu thị số mà ký hiệu bàn tay trong ảnh đại diện theo ngôn ngữ ký hiệu ASL.

# **III. Phương pháp thực hiện**

## **1. Mô hình Support Vector Machine (SVM)**

### **a. Định nghĩa thuật toán SVM**

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều ( ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" (*hyper-plane*) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



Ưu điểm:

- Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn.

- Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định.

- Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

Nhược điểm:

- Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tồi.

- Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm margin từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã bàn luận ở trên.

### **b. Áp dụng thuật toán SVM**

Trong bài toán nhận diện chữ số trong ngôn ngữ ký hiệu ASL, chúng tôi đã sử dụng phương pháp GridSearch để tìm ra bộ tham số tối ưu cho mô hình SVM, nhằm cải thiện độ chính xác. GridSearch giúp thử nghiệm và so sánh các tổ hợp tham số khác nhau để xác định bộ tham số tốt nhất phù hợp với dữ liệu và bài toán.

Bộ tham số mà chúng tôi dò tìm bao gồm các giá trị sau:

param\_grid = {

'C': [0.1, 1, 10, 100],

'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],

'gamma': ['scale', 'auto', 0.01, 0.1, 1],

'degree': [2, 3, 4],

}

- Tham số C: [0.1, 1, 10, 100] – kiểm soát mức độ phạt lỗi trong quá trình huấn luyện.

- Kernel: ['linear', 'poly', 'rbf', 'sigmoid'] – xác định loại hàm nhân (kernel function) được sử dụng để ánh xạ dữ liệu sang không gian đặc trưng.

- Gamma: ['scale', 'auto', 0.01, 0.1, 1] – ảnh hưởng đến cách mô hình học từ dữ liệu khi sử dụng các kernel phi tuyến.

- Degree: [2, 3, 4] – chỉ áp dụng cho kernel 'poly', quy định bậc của đa thức trong hàm nhân.

Cuối cùng chúng tôi tìm được tham số tối ưu cho model là:

*C=10, degree=2, gamma='scale', kernel='rbf'.*

Việc dò tìm tham số với bộ giá trị trên đảm bảo mô hình có thể tối ưu hóa độ chính xác mà không bị quá khớp hoặc quá đơn giản. Kết quả của GridSearch giúp xác định các tham số tốt nhất để mô hình hoạt động hiệu quả nhất trên tập dữ liệu của bài toán.

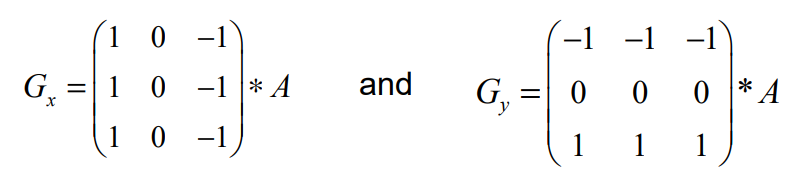
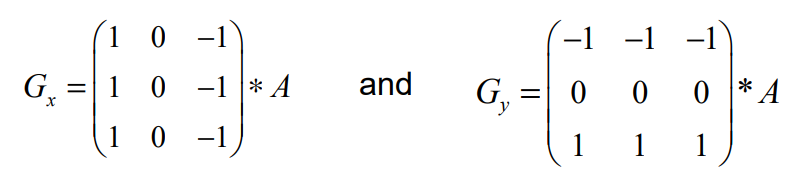
## **2. Bộ lọc Prewitt**

### **a. Định nghĩa bộ lọc Prewitt**

Bộ lọc Prewitt là một trong những phương pháp cổ điển và phổ biến trong lĩnh vực xử lý ảnh, đặc biệt được sử dụng để phát hiện cạnh trong hình ảnh kỹ thuật số. Đây là một loại bộ lọc gradient, hoạt động bằng cách tính toán sự thay đổi cường độ sáng giữa các pixel trong hình ảnh, từ đó làm nổi bật các khu vực có sự chuyển đổi đột ngột về màu sắc hoặc độ sáng, thường là những vùng biên hoặc cạnh trong ảnh.

Bộ lọc Prewitt được thiết kế để cung cấp một sự xấp xỉ đơn giản cho độ dốc của hình ảnh. Mặc dù đơn giản, nó vẫn đủ mạnh để nhận diện các đặc trưng cơ bản trong ảnh và thường được sử dụng làm bước tiền xử lý trong các ứng dụng như nhận dạng vật thể, phân đoạn ảnh, và thị giác máy tính. Với ưu điểm là dễ triển khai và hiệu quả tính toán cao, bộ lọc Prewitt phù hợp với các hệ thống xử lý ảnh thời gian thực hoặc trong các môi trường có yêu cầu không quá khắt khe về độ chính xác.

Bộ lọc Prewitt sử dụng hai toán tử tích chập (kernel) 3x3 riêng biệt để tính gradient theo hướng ngang (Gx) và dọc (Gy). Các kernel này được áp dụng lên ảnh (A) để tạo ra các ma trận đạo hàm Gx và Gy, biểu diễn sự thay đổi cường độ sáng theo các hướng x và y tương ứng.



Sau đấy kết hợp thông tin từ cả hai hướng (x và y) để đánh giá độ mạnh của cạnh tại mỗi pixel. Gradient tổng hợp cho biết mức độ thay đổi cường độ tại một điểm, đại diện cho độ mạnh của cạnh. Được tính bởi công thức sau:

### **b. Áp dụng bộ lọc Prewitt**

Trong bài toán nhận diện chữ số trong ngôn ngữ ký hiệu ASL, chúng tôi thực hiện thao tác rút trích đặc trưng với bộ lọc Prewitt như sau:

Bước 1: Tiền xử lý ảnh:

- Đọc và chuyển ảnh về ảnh xám: chuyển ảnh màu sang ảnh xám giúp giảm số kênh màu từ 3 (RGB) xuống 1, làm nổi bật sự thay đổi cường độ sáng giữa các pixel. Điều này giúp việc phát hiện các biên trở nên dễ dàng hơn, đồng thời giảm thiểu khối lượng tính toán cần thiết so với ảnh màu nhưng vẫn bảo toàn các đặc trưng quan trọng cho bài toán nhận dạng.

- Chuyển đổi kích thước ảnh về 128x128: nhằm tạo sự đồng nhất về kích thước giữa các ảnh, giảm thiểu tài nguyên tính toán mà vẫn giữ được các đặc trưng quan trọng của đối tượng. Kích thước 128x128 được chọn vì nó đủ nhỏ để tối ưu tính toán nhưng vẫn có thể giữ lại các thông tin biên đáng kể.

Bước 2: Tính gradient của ảnh trên trục x và y.

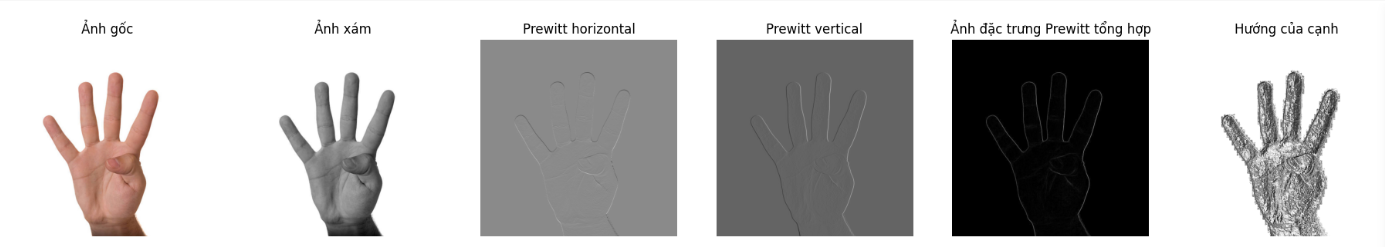
Bước 3: Tính độ lớn tổng hợp của gradient.

Bước 4: Vector hóa và chuẩn hóa đặc trưng:

- Flatten ảnh: cần đưa dữ liệu về dạng vector phẳng (1 chiều) để sử dụng cho các thuật toán học máy. Mục tiêu là biến ma trận 2D (128x128) thành một vector 1D có kích thước 16.384 (128x128 = 16.384).

- Chuẩn hóa dữ liệu: Để các giá trị pixel nằm trong cùng một phạm vi và tránh sự chênh lệch quá lớn giữa các giá trị, phương pháp chuẩn hóa MinMaxScaler được áp dụng. Kỹ thuật này đưa các giá trị pixel về phạm vi [0, 1], giúp tăng tốc độ hội tụ của các thuật toán học máy và đảm bảo các đặc trưng có mức độ ảnh hưởng ngang nhau trong quá trình huấn luyện mô hình. Bên cạnh đó, việc flatten trước khi chuẩn hóa đảm bảo rằng quá trình chuẩn hóa được áp dụng cho tất cả các pixel trong ảnh một cách đồng nhất, không bị ảnh hưởng bởi cấu trúc ma trận 2D của ảnh.

Hình ảnh minh họa:



## **3. Bộ lọc Sobel**

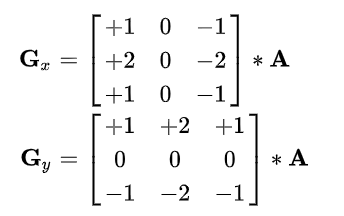
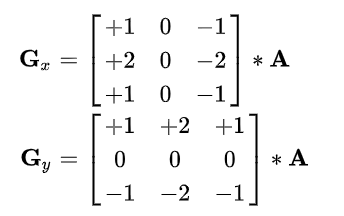
### **a. Định nghĩa bộ lọc Sobel**

Bộ lọc Sobel là một kỹ thuật xử lý ảnh phổ biến, được sử dụng chủ yếu để phát hiện biên cạnh trong ảnh. Phương pháp này giúp xác định các vùng có sự thay đổi đột ngột về cường độ sáng – đây là những khu vực mà các đối tượng trong ảnh chuyển từ màu sắc này sang màu sắc khác, thường là ranh giới của các vật thể.

Bộ lọc Sobel dựa trên việc tính toán gradient của cường độ sáng tại mỗi điểm trong ảnh. Gradient này cho biết sự thay đổi cường độ sáng theo các hướng khác nhau, từ đó giúp phát hiện các cạnh nằm theo các hướng khác nhau.

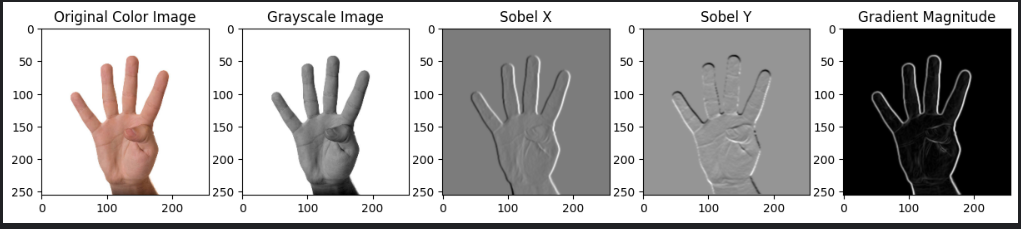
Trước khi thực hiện việc tính toán gradient, ảnh sẽ được làm mịn bằng bộ lọc Gaussian để giảm nhiễu. Việc này giúp đảm bảo quá trình phát hiện cạnh được chính xác hơn, tránh bị ảnh hưởng bởi các điểm nhiễu.

Bộ lọc Sobel sử dụng hai toán tử tích chập (kernel) 3x3 riêng biệt để tính gradient theo hướng ngang (Gx) và dọc (Gy). Các kernel này được áp dụng lên ảnh (A) để tạo ra các ma trận đạo hàm Gx và Gy, biểu diễn sự thay đổi cường độ sáng theo các hướng x và y tương ứng.



Sau đó tính độ lớn của gradient để xác định cường độ của biên cạnh tại mỗi điểm, chúng ta tính độ lớn của vector gradient từ Gx và Gy theo công thức:

Kết quả của bộ lọc Sobel là một ảnh nhị phân hoặc mức xám làm nổi bật các cạnh trong ảnh gốc, với những khu vực có giá trị gradient lớn được hiển thị rõ ràng. Đây là cơ sở để xác định hình dạng, ranh giới của các vật thể trong các ứng dụng phân đoạn ảnh và nhận dạng đối tượng.



Bộ lọc Sobel được ứng dụng rộng rãi trong các lĩnh vực liên quan đến xử lý và phân tích ảnh như nhận diện khuôn mặt, phát hiện biên cạnh trong y học, nhận diện biển báo giao thông và nhiều lĩnh vực khác trong thị giác máy tính.

### **b. Áp dụng bộ lọc Sobel**

Trong bài toán nhận diện chữ số trong ngôn ngữ ký hiệu ASL, chúng tôi thực hiện thao tác rút trích đặc trưng với bộ lọc Sobel, trong đó các bước tiền xử lý ảnh và vector hóa, chuẩn hóa đặc trưng được thực hiện với thao tác và các lý do tương tự như bộ lọc prewitt. Cụ thể quá trình được thực hiện như sau:

Bước 1: Tiền xử lý ảnh:

- Đọc và chuyển ảnh về ảnh xám.

- Chuyển đổi kích thước ảnh về 128x128.

Bước 2: Làm mịn ảnh bằng bộ lọc Gaussian: đảm bảo quá trình phát hiện cạnh được chính xác hơn, tránh bị ảnh hưởng bởi các điểm nhiễu.

Bước 3: Tính gradient của ảnh trên trục x và y.

Bước 4: Tính độ lớn tổng hợp của gradient.

Bước 5: Vector hóa và chuẩn hóa đặc trưng:

- Flatten ảnh: Duỗi ảnh từ ma trận 2 chiều 128x128 về vector có kích thước 16.384 (128x128 = 16.384).

- Chuẩn hóa dữ liệu: dùng MinMaxScaler đưa giá trị pixel về phạm vi [0,1].

## **4. Bộ lọc HOG**

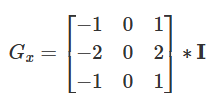
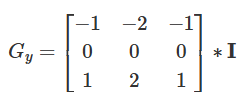
### **a. Định nghĩa bộ lọc HOG**

Trong hầu hết các thuật toán xử lý ảnh, bước đầu tiên là tiền xử lý dữ liệu ảnh (pre-processing image). Chúng ta sẽ cần chuẩn hóa màu sắc và giá trị gamma. Tuy nhiên, bước này có thể được bỏ qua trong phần tính toán bộ mô tả HOG, vì việc chuẩn hóa bộ mô tả ở bước tiếp theo đã đạt được kết quả tương tự. Thay vào đó, tại bước đầu tiên của tính toán bộ mô tả chúng ta sẽ tính các giá trị gradient. Phương pháp phổ biến nhất là áp dụng một mặt nạ đạo hàm rời rạc (discrete derivative mask) theo một hoặc cả hai chiều ngang và dọc. Cụ thể, phương pháp sẽ lọc ma trận cường độ ảnh với các bộ lọc như [Sobel mask](https://en.wikipedia.org/wiki/Sobel_operator) hoặc scharr.

Quy trình thực hiện phát hiện biên cạnh với bộ lọc HOG như sau:

Đầu tiên là tính gradient, chúng ta có thể tính toán với hai kernel như sau:

- Đạo hàm theo chiều dọc (Gx) và đạo hàm theo chiều ngang (Gy):



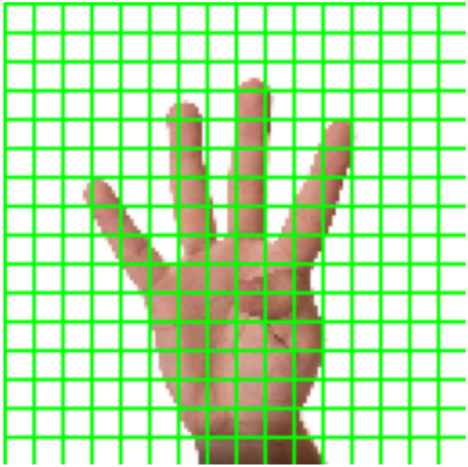
Giá trị độ lớn gradient (gradient magnitude) và phương gradient (gradient direction) có thể được tạo ra từ 2 đạo hàm Gx và Gy theo công thức bên dưới:

Gradient Direction:

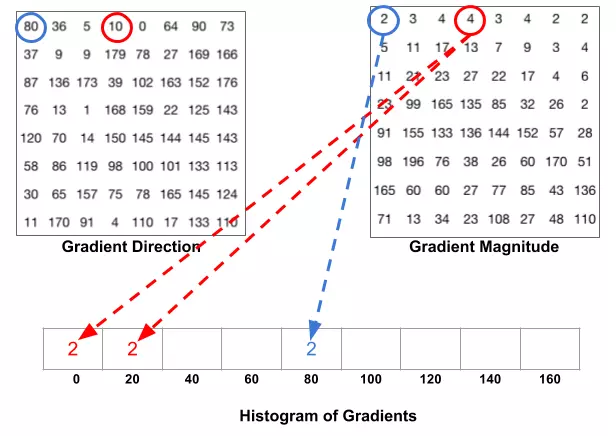
Gradient Magnitude:

Tiếp theo đó là thực hiện tính vector đặc trưng cho từng ô. Nhận thấy đặc trưng của mỗi bức ảnh được biểu diễn thông qua 2 thông số đó là mức độ thay đổi cường độ màu sắc (ma trận gradient magnitude) và hướng thay đổi cường độ màu sắc (ma trận gradient direction). Do đó chúng ta cần tạo ra được một *bộ mô tả* (feature descriptor) sao cho biến đổi bức ảnh thành một véc tơ mà thể hiện được cả 2 thông tin này.

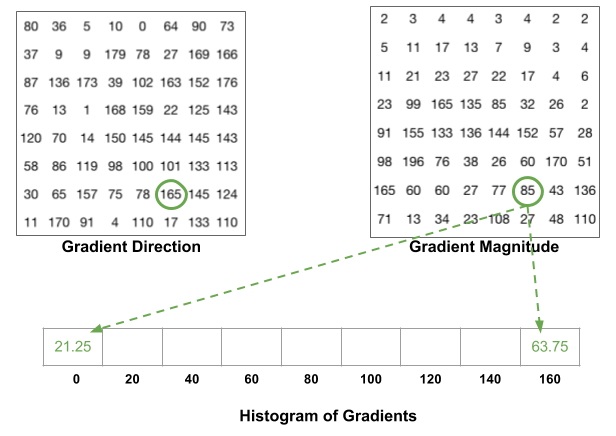
Hình ảnh được chia thành một lưới ô vuông mà mỗi một ô có kích thước 8x8 pixels. Như vậy chúng ta có tổng cộng 64 ô pixels tương ứng với mỗi ô. Trên mỗi một ô trong 64 pixels ta sẽ cần tính ra 2 tham số đó là độ lớn gradient (gradient magnitute) và phương gradient (gradient direction). Như vậy tổng cộng 8x8x2 = 128 giá trị cần tính bao gồm 64 giá trị gradient magnitute và 64 giá trị gradient direction như ma trận hình bên dưới:



Mapping độ lớn gradient vào các bins tương ứng của phương gradient. Sắp xếp các giá trị phương gradient theo thứ tự từ nhỏ đến lớn và chia chúng vào 9 bins. Độ lớn của phương gradient sẽ nằm trong khoảng [0, 180] nên mỗi bins sẽ có độ dài là 20 như hình bên dưới.

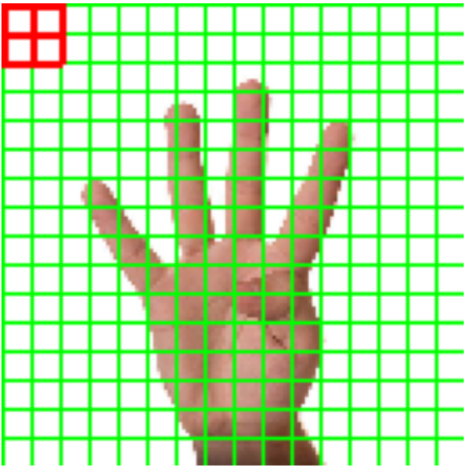
Đầu mút là các giá trị chia hết cho độ rộng của một bin (chẳng hạn 0, 20, 40,… là những đầu mút bin). Trong trường hợp độ lớn phương gradients không rơi vào các đầu mút, ta sẽ sử dụng linear interpolation để phân chia độ lớn gradient về 2 bins liền kề mà giá trị phương gradient rơi vào. Ví dụ: giá trị phương gradient bằng x ghép cặp với độ lớn gradient bằng y. tức là phương gradients rơi vào khoảng giữa bin thứ (l−1) và bin thứ l: . Khi đó tại 2 bins (l−1) và l được điền vào giá trị cường độ theo công thức interpolation:

Giá trị tại bin l - 1:

Giá trị tại bin l:

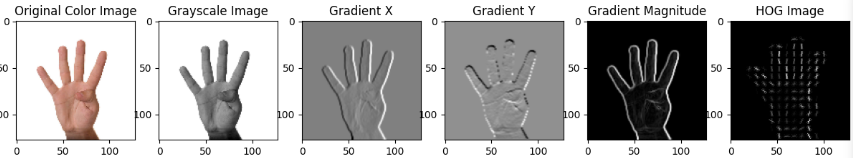
Sau khi xong bước này, với mỗi cell ta sẽ thu được vector với độ dài là 9.

Kế tiếp là chuẩn hóa vector histogram theo block 16x16. Histogram của ảnh phụ thuộc vào cường độ pixel, nên với hai ảnh cùng nội dung nhưng khác độ sáng (ví dụ, một ảnh tối hơn là ảnh gốc nhân 1/2), histogram của ảnh gốc sẽ gấp đôi histogram của ảnh tối. Vì vậy, cần chuẩn hóa histogram để cả hai ảnh có cùng một vector biểu diễn.

Quá trình chuẩn hóa sẽ thực hiện trên một block kích thước 2x2 trên lưới ô vuông ban đầu (mỗi ô kích thước 8x8 pixel). Như vậy chúng ta sẽ có 4 vector histogram kích thước 1x9, concatenate các vector sẽ thu được vector histogram tổng hợp kích thước là 1x36 và sau đó chuẩn hóa theo norm chuẩn bậc 2 trên vector này.

Cuối cùng là tính toán vector HOG. Sau khi chuẩn hóa các véc tơ histogram, chúng ta sẽ concatenate các vector 1x36 này thành một vector lớn. Đây chính là véc tơ HOG đại diện cho toàn bộ hình ảnh. Ta sẽ có 15\*15 = 225 block, với mỗi block sẽ có 1 vector 36\*1. Do đó độ lớn vector của chúng ta là 225\*36=8100.

Dưới đây là biểu diễn các vector đặc trưng thành một bức ảnh gọi là HOG Image:



Output của HOG (Histogram of Oriented Gradients) là một vector đặc trưng (feature vector) mô tả các đặc điểm hình học của ảnh, đặc biệt là các cạnh và hướng gradient trong ảnh. Từ vector này có thể được sử dụng trong nhiều ứng dụng nhận dạng đối tượng

### **b. Áp dụng bộ lọc HOG**

Quá trình rút trích đặc trưng được thực hiện theo các bước sau:

Bước 1: Tiền xử lý ảnh:

- Đọc và chuyển ảnh về ảnh xám.

- Chuyển đổi kích thước ảnh về 128x128.

Bước 2: Dùng bộ lọc Sobel tính độ lớn và phương gradient.

Bước 3 Tính Histogram of Gradients trên cell.

Bước 4: Chuẩn hóa vector histogram trên các block.

Bước 5: Tính toán vector HOG.

Trong bộ lọc HOG chúng tôi đã thử nghiệm trên nhiều bộ tham số khác nhau để tìm ra được mô hình tốt nhất cho việc dự đoán. Các tham số cụ thể như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Orientation** | **Cell size** | **Block size** | **Block norm** |
| 9 | (10,10) | (2,2) | L2-HYS |
| 9 | (10,10) | (2,2) | L2 |
| 9 | (11,11) | (2,2) | L2 |
| 9 | (12,12) | (2,2) | L2 |
| 9 | (13,13) | (2,2) | L2 |
| 9 | (14,14) | (2,2) | L2 |
| 9 | (15,15) | (2,2) | L2 |
| 9 | (16,16) | (2,2) | L2 |
| 9 | (8,8) | (2,2) | L2 |
| 9 | (8,8) | (3,3) | L2 |
| 9 | (12,12) | (3,3) | L2 |
| 9 | (16,16) | (3,3) | L2 |

Sau khi thử nghiệm chúng tôi đã tìm được bộ tham số tối ưu đó là:

*Orientation = 9, cell size = (10,10), block size = (2,2), block norm = L2.*

# **IV. Thực nghiệm**

## **1. Dataset**

Bộ dữ liệu sử dụng trong bài toán gồm **14.194** ảnh tay thực hiện các ký hiệu số từ 0 đến 9 trong ngôn ngữ ký hiệu ASL. Trong đó **10.000** ảnh được thu thập từ Kaggle (Synthetic ASL) và **4.194** ảnh tự chụp.

Đối với ảnh tự chụp, để bổ sung tính đa dạng và phù hợp với bối cảnh bài toán, chúng tôi tiến hành thu thập dữ liệu như sau:

- Bối cảnh chụp: Vì bộ dataset trên Kaggle chủ yếu được chụp trong bối cảnh ngoài trời, nên chúng tôi quyết định chụp thêm ảnh trong phòng để tăng khả năng tổng quát hóa của mô hình.

- Ảnh được chụp từ bàn tay của cả nam và nữ để đảm bảo mô hình có thể nhận diện ký hiệu từ nhiều kiểu dáng bàn tay khác nhau. Người thực hiện không đeo bao tay nhằm giữ lại các đặc điểm tự nhiên của bàn tay.

- Nền chụp: Các bức ảnh được chụp tại các góc trong phòng học của trường Đại học Công nghệ Thông tin (UIT), phần lớn có nền trơn, giúp giảm nhiễu và tập trung vào ký hiệu tay.

- Tư thế và hướng tay: Các ký hiệu tay được thực hiện tuân theo ràng buộc hình ảnh của bộ dữ liệu đã trình bày ở mục Input, bao gồm các hướng tay khác nhau để đảm bảo tính đa dạng.

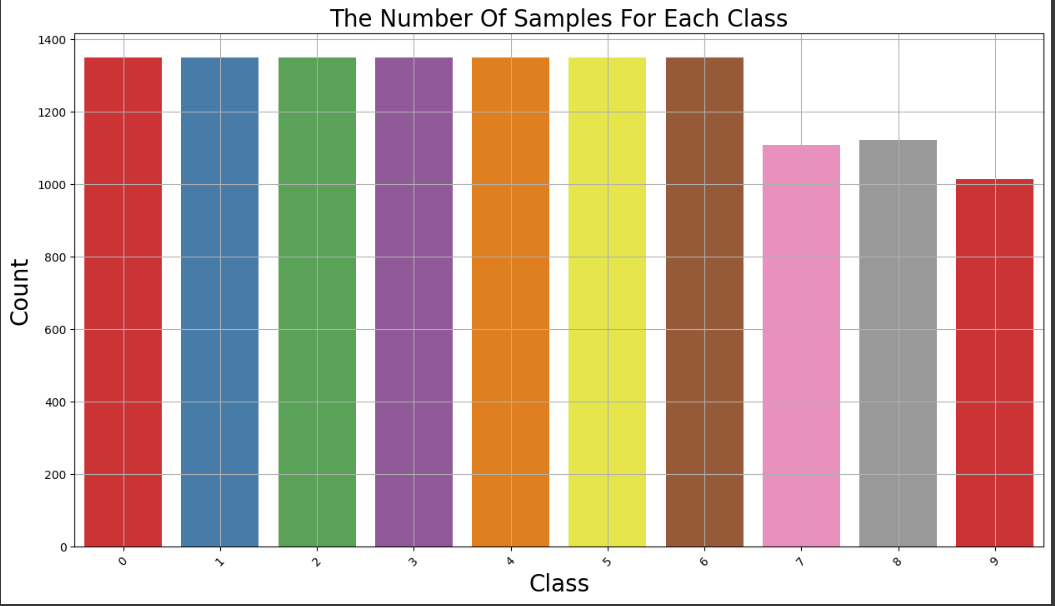
Tổng cộng, chúng tôi đã chụp **5.327** ảnh. Sau đó, tiến hành chọn lọc, loại bỏ các bức ảnh bị mờ, ảnh có ký hiệu không rõ ràng, hoặc ảnh bị thiếu sáng. Kết quả cuối cùng thu được là **4.194** ảnh tự chụp chất lượng cao, bổ sung vào bộ dữ liệu sử dụng trong bài toán.



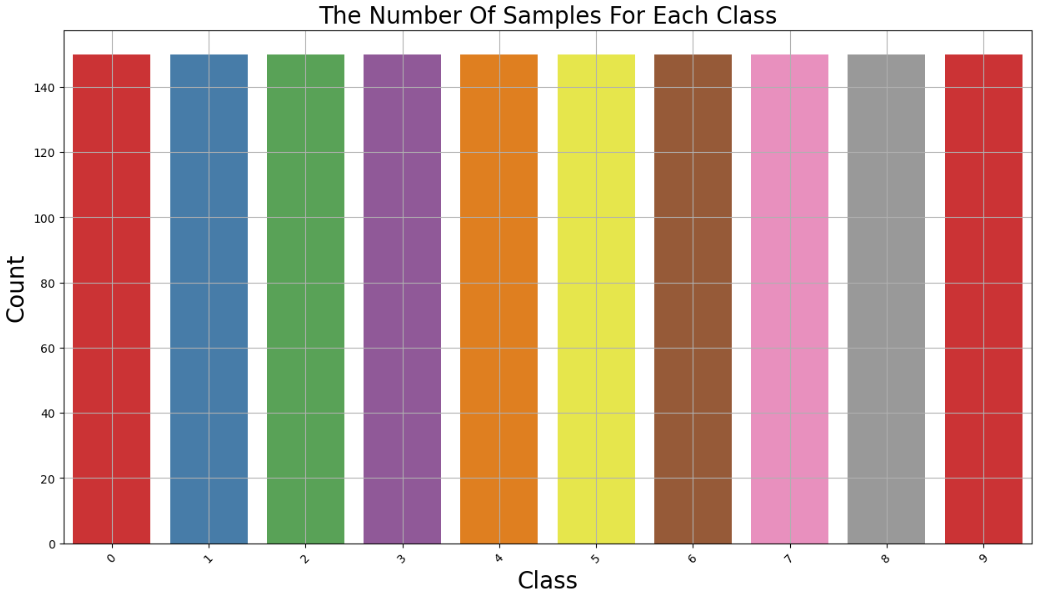
Với tổng số lượng ảnh khá lớn, bao gồm 14.194 ảnh nên dữ liệu được chia ở tập train là **12.694** ảnh **~90%** và dữ liệu trong tập test là **1.500** ảnh **~ 10%.** Với mỗi tập, các ảnh được lưu trong 10 thư mục con, mỗi thư mục đại diện cho một lớp (từ 0 đến 9), với tên thư mục là nhãn của lớp.

Trong tập train, từ class 0 đến 6, mỗi class chứa 1.350 ảnh, class 7 chứa 1.109 ảnh, class 8 chứa 1.121 ảnh và class 9 chứa 1.014 ảnh.

Biểu đồ phân bố dữ liệu trong tập train:

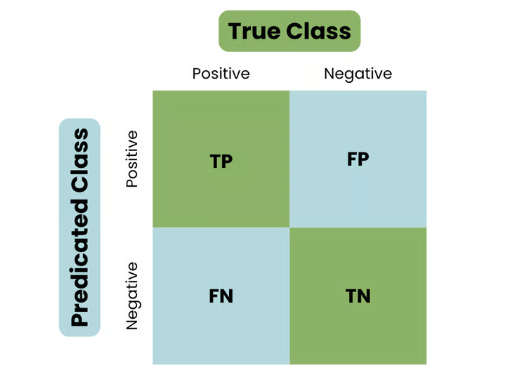
Trong tập test, mỗi class chứa 150 ảnh.

Biểu đồ phân bố dữ liệu trong tập test:

****

## **2. Độ đo đánh giá**

Cho các định nghĩa TP, FP, FN, TN như hình bên dưới. Chúng tôi sử dụng các độ đo Accuracy, Precision, Recall và F1 score để đánh giá mô hình của mình.



### **a. Accuracy**

Accuracy đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán, thể hiện mức độ chính xác tổng quát của mô hình. Tuy nhiên, độ đo này không phản ánh chi tiết khả năng phân loại đúng của từng lớp hoặc mức độ nhầm lẫn giữa các lớp, đặc biệt trong các trường hợp dữ liệu không cân bằng (class imbalance).

### **b. Precision**

Precision được định nghĩa là tỷ lệ mẫu True Positive (TP) trong số các mẫu được dự đoán là Positive. Nó đánh giá khả năng của mô hình trong việc không gán nhầm nhãn Positive cho các mẫu không thuộc lớp đó. Precision phù hợp khi lỗi dương tính giả (False Positive) cần được hạn chế.

### **c. Recall**

Recall đo lường tỷ lệ mẫu True Positive (TP) trong số các mẫu thực sự thuộc lớp Positive. Nó phản ánh khả năng phát hiện đầy đủ các mẫu thuộc lớp mục tiêu của mô hình. Recall hữu ích trong các trường hợp cần hạn chế bỏ sót các mẫu dương tính (False Negative).

### **d. F1 score**

F1-score là trung bình điều hòa của precision và recall, giúp cân bằng giữa hai độ đo này. F1-score hữu ích khi cần một chỉ số duy nhất để đánh giá mô hình, đặc biệt trong trường hợp dữ liệu không cân bằng.

## **3. Kết quả**

### **a. Classification Report:**

##### *- Mô hình svm\_with\_prewitt:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **class** | **Precision** | **Recall** | **F1-score** | **Support** |
| 0 | 0.69 | 0.72 | 0.70 | 150 |
| 1 | 0.76 | 0.77 | 0.77 | 150 |
| 2 | 0.79 | 0.83 | 0.81 | 150 |
| 3 | 0.82 | 0.85 | 0.83 | 150 |
| 4 | 0.51 | 0.92 | 0.65 | 150 |
| 5 | 0.72 | 0.73 | 0.72 | 150 |
| 6 | 0.87 | 0.83 | 0.85 | 150 |
| 7 | 0.78 | 0.60 | 0.68 | 150 |
| 8 | 0.82 | 0.60 | 0.69 | 150 |
| 9 | 0.93 | 0.53 | 0.68 | 150 |
| Average | 0.77 | 0.74 | 0.74 | 1500 |

Mô hình sử dụng đặc trưng Prewitt đạt Average Precision là **0.77**, Average Recall và Average F1-score đều là **0.74**. Đây là kết quả ở mức tương đối khả quan, nhưng vẫn chưa thực sự cao. Điều này có thể được lý giải bởi Prewitt là một phương pháp phát hiện cạnh đơn giản, không có khả năng loại bỏ thông tin nhiễu một cách hiệu quả. Do đó, đặc trưng mà phương pháp này tạo ra chưa đủ chi tiết và giàu thông tin để mô hình đạt hiệu suất cao hơn.

Phân tích cụ thể cho từng lớp:

- Chữ số dễ nhận diện nhất: Chữ số 9 có Precision = 0.93, cao nhất trong tất cả các lớp. Điều này cho thấy mô hình nhận diện chữ số này rất tốt, với rất ít nhầm lẫn xảy ra.

- Chữ số khó nhận diện nhất: Chữ số 4 có Precision = 0.51 và F1-score = 0.65, thấp nhất trong tất cả các lớp. Kết quả này phản ánh rằng mô hình gặp nhiều khó khăn trong việc phân biệt chữ số 4 với các chữ số khác, có thể do đặc trưng của Prewitt không đủ khả năng tách biệt rõ ràng các đặc điểm của chữ số này.

Nhìn chung, kết quả cho thấy việc sử dụng Prewitt cần được cải thiện bằng cách kết hợp thêm các phương pháp phát hiện đặc trưng mạnh hơn hoặc kết hợp với các kỹ thuật tiền xử lý nhằm giảm nhiễu trong dữ liệu đầu vào.

##### *- Mô hình svm\_with\_sobel:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **class** | **Precision** | **Recall** | **F1-score** | **Support** |
| 0 | 0.68 | 0.74 | 0.71 | 150 |
| 1 | 0.74 | 0.75 | 0.74 | 150 |
| 2 | 0.84 | 0.82 | 0.83 | 150 |
| 3 | 0.84 | 0.89 | 0.87 | 150 |
| 4 | 0.54 | 0.93 | 0.68 | 150 |
| 5 | 0.72 | 0.75 | 0.74 | 150 |
| 6 | 0.85 | 0.85 | 0.85 | 150 |
| 7 | 0.76 | 0.57 | 0.65 | 150 |
| 8 | 0.85 | 0.60 | 0.70 | 150 |
| 9 | 0.90 | 0.58 | 0.70 | 150 |
| Average | 0.77 | 0.75 | 0.75 | 1500 |

Nhìn chung, mô hình với đặc trưng Sobel đạt kết quả gần giống với Prewitt, trong đó Average Precision là **0.77**, Average Recall và Average F1-score cùng là **0.75**. Đây là kết quả đáng khích lệ nhưng chưa quá cao, do Sobel là một phương pháp phát hiện cạnh đơn giản, không có khả năng loại bỏ nhiễu hiệu quả, nên các đặc trưng thu được chưa thực sự trực quan và giàu thông tin.

Phân tích chi tiết:

- Các chữ số dễ nhận diện nhất: Các chữ số 0, 1, 2, 3, 5, 6 có Precision, Recall, và F1-score xấp xỉ ngang nhau, cho thấy chúng được nhận diện tốt. Các chữ số này dễ phân biệt vì ký hiệu của chúng có đặc trưng khác biệt rõ ràng so với các lớp còn lại. Điều này giúp mô hình ít nhầm lẫn khi phân loại các chữ số này.

- Các chữ số có Precision cao nhưng Recall thấp: Các chữ số 7, 8, 9 có Precision cao nhưng Recall thấp. Precision cao cho thấy mô hình dự đoán chính xác khi khẳng định một mẫu thuộc các lớp này. Tuy nhiên, Recall thấp phản ánh rằng mô hình bỏ sót nhiều mẫu thực sự thuộc các lớp này, có thể do ký hiệu của các chữ số này dễ bị nhầm lẫn với các lớp khác.

- Chữ số khó nhận diện nhất: Chữ số 4 có Precision thấp (0.54) nhưng Recall cao (0.93). Điều này cho thấy mô hình nhận diện được hầu hết các mẫu thực sự thuộc lớp 4 (Recall cao), nhưng lại dự đoán sai nhiều mẫu từ các lớp khác vào lớp 4 (Precision thấp). Nguyên nhân có thể là ký hiệu của số 4 có nét tương đồng với một số chữ số khác, khiến mô hình khó phân biệt.

Các kết quả trên phản ánh rằng mô hình hoạt động tốt hơn với các chữ số có đặc trưng khác biệt rõ ràng, trong khi gặp khó khăn với các chữ số có ký hiệu dễ nhầm lẫn. Việc cải thiện hiệu suất mô hình có thể tập trung vào việc tăng khả năng loại bỏ nhiễu và trích xuất đặc trưng mạnh mẽ hơn, từ đó giúp phân biệt tốt hơn giữa các chữ số dễ nhầm lẫn như 4, 7, 8, và 9.

##### *- Mô hình svm\_with\_hog:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **class** | **Precision** | **Recall** | **F1-score** | **Support** |
| 0 | 0.83 | 0.97 | 0.90 | 150 |
| 1 | 0.72 | 0.95 | 0.82 | 150 |
| 2 | 0.91 | 0.97 | 0.94 | 150 |
| 3 | 0.99 | 0.99 | 0.99 | 150 |
| 4 | 0.98 | 0.99 | 0.99 | 150 |
| 5 | 0.76 | 0.99 | 0.86 | 150 |
| 6 | 0.97 | 1.00 | 0.98 | 150 |
| 7 | 0.97 | 0.66 | 0.79 | 150 |
| 8 | 0.90 | 0.69 | 0.78 | 150 |
| 9 | 0.98 | 0.65 | 0.78 | 150 |
| Average | 0.90 | 0.89 | 0.89 | 1500 |

Nhìn chung, mô hình với đặc trưng HOG đạt kết quả cao nhất và tốt nhất trong cả ba phương pháp. Cụ thể, Average Precision đạt **0.90**, Average Recall và Average F1-score cùng đạt **0.89**. Điều này cho thấy HOG, với khả năng trích xuất đặc trưng hình dạng mạnh mẽ, đã cung cấp các đặc trưng trực quan, giàu thông tin, giúp mô hình phân loại tốt hơn.

Phân tích chi tiết:

- Các chữ số dễ nhận diện nhất: Các chữ số 3, 4, 6 có Precision, Recall, và F1-score gần như hoàn hảo và xấp xỉ ngang nhau. Các chữ số này dễ nhận diện nhờ ký hiệu của chúng có đặc trưng rõ ràng, giúp mô hình phân biệt tốt với các lớp còn lại.

- Các chữ số có Precision cao nhưng Recall thấp: Các chữ số 7, 8, 9 có Precision cao nhưng Recall thấp. Điều này chỉ ra rằng mô hình nhận diện chính xác các mẫu dự đoán thuộc các lớp này (Precision cao), nhưng lại bỏ sót nhiều mẫu thực sự thuộc các lớp đó (Recall thấp). Nguyên nhân có thể do các ký hiệu của số 7, 8, và 9 có nét tương đồng với một số chữ số khác, khiến mô hình dễ nhầm lẫn.

- Chữ số khó nhận diện nhất: Chữ số 1 có Precision thấp nhất, cho thấy mô hình gặp khó khăn trong việc nhận diện chính xác các mẫu thuộc lớp này. Nguyên nhân có thể là do ký hiệu của số 1 đơn giản và dễ bị nhầm lẫn với các nét khác trong dữ liệu.

So sánh với các phương pháp khác:

- So với Prewitt và Sobel, mô hình với HOG đã cải thiện rõ rệt về hiệu suất. Precision, Recall, và F1-score đều cao hơn, đặc biệt ở các lớp như 3, 4, và 6.

- HOG giúp nhận diện tốt hơn các ký hiệu có hình dạng rõ ràng, nhưng vẫn gặp khó khăn với các ký hiệu dễ nhầm lẫn như 7, 8, 9, và đặc biệt là 1.

Mô hình sử dụng đặc trưng HOG đã đạt kết quả tốt nhất, chứng minh rằng việc trích xuất đặc trưng hình dạng mạnh mẽ là yếu tố quan trọng để cải thiện hiệu suất phân loại. Tuy nhiên, để cải thiện hơn nữa, cần tập trung xử lý các lớp có ký hiệu dễ nhầm lẫn và tinh chỉnh thêm dữ liệu hoặc các bước tiền xử lý.

### **b. Confusion Matrix và Accuracy**

Confusion matrix của các mô hình:

|  |  |
| --- | --- |
| **svm\_with\_prewitt** | **svm\_with\_sobel** |
|  |  |
| **svm\_with\_hog** | |
|  | |

Độ chính xác của các mô hình:

|  |  |
| --- | --- |
| **Mô hình** | **Accuracy** |
| svm\_with\_prewitt | 0.7387 |
| svm\_with\_sobel | 0.748 |
| svm\_with\_hog | 0.886 |

Trong ba mô hình với ba loại đặc trưng thì mô hình svm\_with\_hog cho kết quả cao nhất tuy nhiên mô hình vẫn bị nhầm lẫn khi dự đoán các số 7, 8 và 9 thành các số 0, 1, 5.

### **c. Tăng cường dữ liệu với đặc trưng có kết quả thực nghiệm cao nhất**

Sau quá trình đánh giá các mô hình với ba loại đặc trưng khác nhau, mô hình cho kết quả cao nhất là svm\_with\_hog, sử dụng đặc trưng HOG, đạt độ chính xác 0.886. Với mục tiêu nâng cao hiệu quả và đảm bảo tính tổng quát của mô hình trong phần demo, chúng tôi đã thực hiện thêm thao tác tăng cường dữ liệu (Data Augmentation) cho đặc trưng HOG. Các kỹ thuật tăng cường dữ liệu được áp dụng bao gồm:

- Xoay ngẫu nhiên (-45o đến 45o): Giúp mô hình học được các đặc trưng từ nhiều góc nhìn khác nhau của ngôn ngữ ký hiệu, tăng khả năng nhận diện khi dữ liệu thực tế có sự thay đổi nhỏ về góc nhìn.

- Phóng to/Thu nhỏ (70%-120%): Giúp mô hình trở nên linh hoạt với các thay đổi về kích thước hình ảnh do khoảng cách hoặc cách chụp ảnh.

- Thay đổi độ sáng (70%-120%): Tăng khả năng chịu nhiễu của mô hình trước các điều kiện ánh sáng khác nhau.

- Lật ngang ngẫu nhiên (xác suất 50%): Giúp mô hình học được các biến thể đối xứng, đặc biệt hiệu quả với các ký hiệu có thể bị đảo ngược khi sử dụng hai tay.

Sau khi áp dụng các kỹ thuật trên, mô hình SVM với đặc trưng HOG được huấn luyện lại và dùng tập dữ liệu đã được tăng cường.

Kết quả thu được như sau:

|  |  |
| --- | --- |
| **Mô hình** | **Accuracy** |
| svm\_with\_hog | 0.886 |
| svm\_with\_hog\_arg | 0.890 |

Việc áp dụng tăng cường dữ liệu đã giúp cải thiện độ chính xác từ 0.886 lên 0.890. Mặc dù mức cải thiện không lớn, nhưng điều này cho thấy tính hiệu quả của việc tăng cường dữ liệu, giúp mô hình trở nên linh hoạt hơn khi xử lý các biến thể trong tập dữ liệu thực tế.

# **V. Kết luận**

Trong bài toán nhận diện chữ số ngôn ngữ ký hiệu ASL, các đặc trưng trích xuất từ ảnh đóng vai trò quan trọng trong hiệu suất của mô hình. Trong số các đặc trưng được thử nghiệm HOG đạt độ chính xác cao nhất với accuracy là 0.886 và sau khi được tăng cường dữ liệu thì accuracy tăng lên 0.890. Trong khi đó, Prewitt và Sobel (accuracy tương ứng là 0.7387 và 0.748) cho kết quả gần bằng nhau và thấp hơn hẳn so với HOG.

Prewitt và Sobel cho kết quả chưa tốt do 2 bộ lọc này phát hiện cạnh bằng phương pháp đơn giản, chưa có tính tổng quát và dễ bị nhiễu bởi ngoại cảnh, cho nên không tốt khi dùng cho phân biệt các ký hiệu bàn tay có phần phức tạp, giống nhau.

HOG thể hiện kết quả vượt trội hơn bởi vì nó nắm bắt các đặc trưng cấu trúc của hình ảnh, tạo được đặc trưng tổng quát và không bị ảnh hưởng bởi nhiễu từ ngoại cảnh.Sự khác biệt này khẳng định rằng việc lựa chọn đúng đặc trưng trích xuất là yếu tố then chốt trong bài toán xử lý ảnh và nhận diện ngôn ngữ ký hiệu. Trong tương lai, các phương pháp tiên tiến hơn như kết hợp HOG với các kỹ thuật học sâu có thể được áp dụng để tiếp tục cải thiện độ chính xác và khả năng tổng quát hóa của hệ thống.

# **VI. Hướng phát triển đề tài**

Với những kết quả đã đạt được, chúng tôi đề xuất một số hướng phát triển như sau:

- Cải thiện hiệu suất mô hình: Tích hợp các kỹ thuật rút trích đặc trưng tiên tiến như CNN (Convolutional Neural Networks) để tự động học đặc trưng từ dữ liệu, thay thế cho các phương pháp truyền thống như Prewitt, Sobel hay HOG. Ngoài ra, có thể thử nghiệm các mô hình hiện đại như Random Forest, Gradient Boosting, hoặc Deep Learning để cải thiện độ chính xác và khả năng khái quát hóa của hệ thống.

- Ứng dụng trên thiết bị di động: Triển khai hệ thống trên các nền tảng di động hoặc nhúng (embedded systems), cho phép người dùng nhận diện chữ số trực tiếp thông qua camera trên điện thoại hoặc các thiết bị hỗ trợ đặc biệt, phục vụ mục đích như nhận diện số điện thoại, ngày tháng năm sinh, hoặc tuổi tác từ ngôn ngữ ký hiệu.

- Tích hợp công nghệ thời gian thực: Phát triển mô hình có khả năng nhận diện chữ số từ ngôn ngữ ký hiệu trong video hoặc camera thời gian thực, nhằm tăng cường hiệu quả và tính tương tác của hệ thống trong thực tế.

- Đào tạo mô hình với dữ liệu đa ngôn ngữ ký hiệu: Để phục vụ cộng đồng người dùng rộng rãi hơn, hệ thống có thể được mở rộng để nhận diện chữ số từ các ngôn ngữ ký hiệu khác nhau như ASL (American Sign Language), BSL (British Sign Language), hoặc VSL (Vietnamese Sign Language).

Với các hướng phát triển trên, đề tài không chỉ giới hạn ở việc nhận diện chữ số đơn thuần mà còn có tiềm năng trở thành một công cụ giao tiếp hiệu quả, hỗ trợ mạnh mẽ cho cộng đồng người khuyết tật và góp phần nâng cao chất lượng cuộc sống trong xã hội.

# **VII. Tài liệu tham khảo**

[1]]<https://viblo.asia/p/gioi-thieu-ve-support-vector-machine-svm-6J3ZgPVElmB>

[2]]<https://machinelearningcoban.com/2017/04/09/smv/>

[3]]<https://en.wikipedia.org/wiki/Prewitt_operator>

[4]]<https://viblo.asia/p/part1-edge-detection-voi-opencv-L4x5xLVB5BM>

[5]]<https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/>

[6]]<https://viblo.asia/p/tim-hieu-ve-phuong-phap-mo-ta-dac-trung-hog-histogram-of-oriente-gradients-V3m5WAwxZO7>