

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN  
BÁO CÁO MÔN HỌC

Đề tài:

PHÂN LOẠI BIỂN BÁO GIAO THÔNG  
VIỆT NAM

Môn học: Nhập môn thị giác máy tính - CS231.O22  
Sinh viên thực hiện: Phạm Đức Huy Hoàng - 22520474  
Nguyễn Hoàng Hiệp - 22520452  
Văn Tiến Hoàng - 22520480  
Giảng viên hướng dẫn: TS. Mai Tiến Dũng

TP Hồ Chí Minh, 6-2024

# Mục lục

<b>LỜI MỞ ĐẦU</b>	i
<b>THÔNG TIN THÀNH VIÊN</b>	ii
<b>CHƯƠNG 1. TỔNG QUAN</b>	1
1.1    Mô tả bài toán . . . . .	1
1.2    Lý do chọn đề tài . . . . .	1
1.3    Phát biểu bài toán . . . . .	1
<b>CHƯƠNG 2. BỘ DỮ LIỆU</b>	2
<b>CHƯƠNG 3. ĐỘ ĐO</b>	3
<b>CHƯƠNG 4 CÁC PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN</b>	4
4.1    Đặc trưng HOG . . . . .	4
4.1.1    Nguyên lý hoạt động . . . . .	4
4.1.2    Áp dụng HOG trong bài toán phân loại biển báo giao thông . .	5
4.1.3    Ưu điểm và nhược điểm của HOG . . . . .	6
4.2    Color Histogram . . . . .	7
4.2.1    Tổng quan về Color Histogram . . . . .	7
4.2.2    Nguyên lý hoạt động . . . . .	7
4.2.3    Áp dụng Color Histogram trong bài toán phân loại biển báo giao thông . . . . .	8
4.2.4    Ưu điểm và nhược điểm của Color Histogram . . . . .	9
4.3    Kết hợp giữa HOG và Color Histogram . . . . .	10
4.3.1    Áp dụng HOG và Color Histogram trong bài toán phân loại biển báo giao thông . . . . .	10
4.4    Thuật toán K-Nearest Neighbors . . . . .	11
4.4.1    Tổng quan thuật toán . . . . .	11

4.4.2	Nguyên lý hoạt động . . . . .	11
4.4.3	Tham số . . . . .	12
4.4.4	Kết quả . . . . .	12
4.5	Thuật toán Decision Tree . . . . .	14
4.5.1	Tổng quan thuật toán . . . . .	14
4.5.2	Nguyên lý hoạt động . . . . .	14
4.5.3	Tham số . . . . .	14
4.5.4	Kết quả . . . . .	15
4.6	Thuật toán Support Vector Machine . . . . .	17
4.6.1	Tổng quan thuật toán . . . . .	17
4.6.2	Nguyên lý hoạt động . . . . .	17
4.6.3	Tham số . . . . .	18
4.6.4	Kết quả . . . . .	20
<b>CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM CỦA 3 PHƯƠNG PHÁP</b>		<b>22</b>
5.1	Độ chính xác giữa 3 mô hình . . . . .	22
5.2	Confusion Matrix . . . . .	22
5.3	Demo . . . . .	23
5.4	Thử nghiệm đặc biệt . . . . .	23
<b>KẾT LUẬN</b>		<b>26</b>
<b>Kết luận chung</b>		<b>26</b>
<b>Hướng phát triển</b>		<b>27</b>
<b>TÀI LIỆU THAM KHẢO</b>		<b>28</b>

## LỜI MỞ ĐẦU

. Trong bối cảnh toàn cầu hóa và sự phát triển nhanh chóng của các đô thị, vấn đề an toàn giao thông đang trở nên ngày càng cấp thiết. Sự gia tăng về số lượng phương tiện và người tham gia giao thông đã tạo ra áp lực lớn đối với hệ thống quản lý giao thông. Hệ thống biển báo giao thông, với vai trò điều tiết và hướng dẫn giao thông, là một yếu tố quan trọng giúp giảm thiểu tai nạn và duy trì trật tự trên đường. Tại Việt Nam, biển báo giao thông được phân loại thành năm nhóm chính: biển báo cấm, biển báo nguy hiểm, biển báo chỉ dẫn, biển báo hiệu lệnh, và biển báo phụ.

Hiện nay, việc phân loại và nhận diện biển báo giao thông chủ yếu dựa vào sự quan sát và nhận biết của con người, dễ dẫn đến những sai sót do yếu tố con người. Để khắc phục vấn đề này, nhiều phương pháp tự động đã được phát triển, trong đó việc ứng dụng các công nghệ học máy và trí tuệ nhân tạo đang ngày càng được quan tâm. Các phương pháp hiện có như sử dụng mạng nơ-ron tích chập (CNN), thuật toán K-Nearest Neighbors (KNN), Support Vector Machine (SVM) và Decision Tree đã cho thấy tiềm năng trong việc phân loại biển báo giao thông với độ chính xác cao.

Tuy nhiên, việc áp dụng các phương pháp này vào thực tế vẫn đối mặt với nhiều thách thức, bao gồm độ phức tạp của biển báo, điều kiện thời tiết và ánh sáng thay đổi, cũng như sự đa dạng về hình dạng và màu sắc của các biển báo.

Mục đích của nghiên cứu này là phát triển và đánh giá các mô hình học máy hiệu quả để phân loại tự động biển báo giao thông, tập trung vào năm nhóm biển báo chính tại Việt Nam. Nghiên cứu này sẽ giới hạn phạm vi trong việc phân tích, lựa chọn và triển khai các mô hình học máy như KNN, SVM và Decision Tree để phân loại chính xác các biển báo giao thông từ hình ảnh.

## THÔNG TIN THÀNH VIÊN

STT	HỌ TÊN	MSSV	PHÂN CÔNG	MỨC ĐỘ HOÀN THÀNH
1	Phạm Đức Huy Hoàng	22520474	Tiền xử lý dữ liệu, đặc trưng HOG, đặc trưng Color Histogram, thuật toán KNN, slide, viết báo cáo.	100%
2	Nguyễn Hoàng Hiệp	22520452	Chọn lọc các ảnh từ dataset, tiền xử lý dữ liệu, thuật toán SVM, thử nghiệm đặc biệt, slide, viết báo cáo.	100%
3	Văn Tiến Hoàng	22520480	Chọn lọc các ảnh từ dataset, tiền xử lý dữ liệu, thử nghiệm đặc biệt thuật toán Decision Tree, slide, viết báo cáo.	100%

# CHƯƠNG 1. TỔNG QUAN

## 1.1 Mô tả bài toán

Trong bối cảnh giao thông đường bộ ở thành phố Hồ Chí Minh, nơi mà số lượng xe càng ngày càng tăng và các đoạn đường thường xuyên gặp biến động, việc phân loại các biển báo giao thông là một phần quan trọng trong việc quản lý an toàn giao thông.

Mỗi loại biển báo, bao gồm cấm, nguy hiểm, chỉ dẫn, hiệu lệnh và phụ, có một ý nghĩa và mục đích sử dụng riêng biệt. Tuy nhiên, sự đa dạng về hình dạng, màu sắc và kí hiệu của các biển báo khiến cho việc phân loại biển báo trở nên vô cùng khó khăn, đặc biệt đối với người lái xe. Do đó, cần phát triển một hệ thống tự động phân loại biển báo giao thông để giúp người lái xe có thể nắm bắt các biển báo một cách chính xác và phản ứng kịp thời trong các tình huống giao thông khác nhau.

## 1.2 Lý do chọn đề tài

Việc lựa chọn bài toán phân loại biển báo giao thông không chỉ đơn giản là để giải quyết vấn đề nhầm lẫn cho người lái xe, mà còn đóng vai trò quan trọng trong bài toán nhận diện biển báo thông, một phần không thể thiếu của hệ thống thông minh giao thông. Thông qua việc phân loại và nhận diện chính xác các loại biển báo, chúng ta có thể tạo ra các ứng dụng hỗ trợ người tham gia giao thông như bản đồ thông minh, cảnh báo trực tiếp trên xe và hệ thống định vị động. Đặc biệt, đề tài còn có ý nghĩa quan trọng trong việc phát triển hệ thống lái tự động, giúp xe tự lái có khả năng nhận diện và tuân thủ các biển báo giao thông một cách chính xác, tối ưu hóa hoạt động của hệ thống và tăng cường an toàn cho giao thông đường bộ.

## 1.3 Phát biểu bài toán

- Input: Một ảnh biển báo mới chưa được gán nhãn cần phân loại.
- Output: Nhãn phân loại của biển báo trong ảnh ("Cấm", "Chỉ dẫn", "Hiệu lệnh", "Nguy hiểm", "Phụ").
- Mục tiêu của bài toán là xây dựng một hệ thống học máy có khả năng phân loại chính xác 5 loại biển báo giao thông thông qua việc huấn luyện trên tập dữ liệu ảnh và nhãn tương ứng. Độ chính xác của việc phân loại các loại biển báo Cấm, Chỉ dẫn, Hiệu lệnh, Nguy hiểm, và Phụ là tiêu chí quan trọng để đánh giá hiệu suất của mô hình. Hệ thống này sẽ giúp người tham gia giao thông nhận diện nhanh chóng và chính xác các biển báo, từ đó nâng cao an toàn và hiệu quả giao thông.

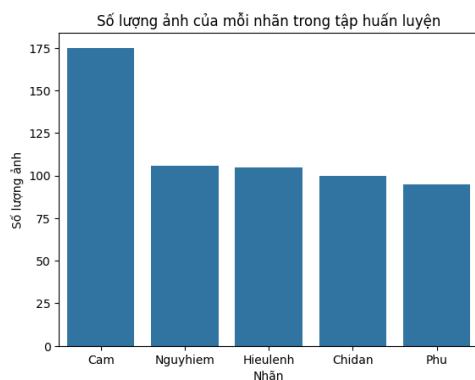
## CHƯƠNG 2. BỘ DỮ LIỆU

Dữ liệu được sử dụng trong bài toán này được chúng em tự chụp xung quanh địa điểm trường Đại học Công Nghệ Thông Tin. Sau đó, chúng em cắt và đảm bảo ảnh biển báo giao thông chiếm 70% - 80% diện tích của ảnh. Tổng cộng, sau khi cắt và kiểm tra tất cả các ảnh, chúng em có 830 ảnh, được phân thành 5 nhãn: Cấm, Chỉ dẫn, Hiệu lệnh, Nguy hiểm, và Phụ.

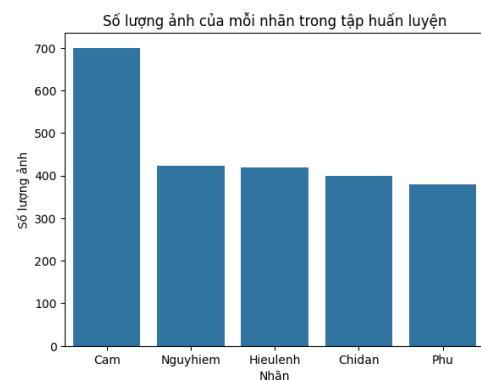
Bộ dữ liệu này được chia thành 70% cho tập huấn luyện và 30% cho tập kiểm tra thông qua **train\_test\_split** của thư viện **sklearn.model\_selection**.

Trước khi đưa vào mô hình học máy, tất cả các ảnh sẽ được điều chỉnh về cùng kích thước **224x224 pixels**. Điều này giúp đảm bảo tính đồng nhất và thuận tiện trong việc xử lý ảnh cho mô hình. Để tăng cường dữ liệu và cải thiện hiệu suất của mô hình, chúng em đã áp dụng kỹ thuật tăng cường dữ liệu thông qua TensorFlow lên tập huấn luyện. Các biến đổi như xoay, phóng to, thu nhỏ, và thêm nhiễu (noise) được áp dụng để tạo ra các phiên bản mới của ảnh gốc, từ đó cải thiện độ chính xác và khả năng tổng quát hóa của mô hình.

Sau khi tăng cường dữ liệu, tập train có 2324 ảnh và tập test có 249 ảnh.



**Hình 1.1:** Dataset trước



**Hình 1.2:** Dataset sau

## CHƯƠNG 3. ĐỘ ĐO

Trong bài báo cáo này, chúng em sẽ sử dụng độ đo accuracy để đánh giá hiệu quả của mô hình phân loại. Độ đo này giúp chúng em hiểu rõ hơn về khả năng phân loại chính xác của mô hình đối với dữ liệu.

**Accuracy** là tỷ lệ giữa số lượng dự đoán đúng (bao gồm cả True Positive và True Negative) trên tổng số dự đoán. Đây là một trong những độ đo phổ biến nhất để đánh giá hiệu suất của mô hình phân loại.

Công thức tính accuracy như sau:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3. 1)$$

Trong đó:

- TP (True Positive): Số lượng mẫu được dự đoán đúng là thuộc lớp đó.
- TN (True Negative): Số lượng mẫu được dự đoán đúng là không thuộc lớp đó.
- FP (False Positive): Số lượng mẫu được dự đoán là thuộc lớp đó nhưng thực tế không thuộc lớp đó.
- FN (False Negative): Số lượng mẫu thực tế thuộc lớp đó nhưng được dự đoán không thuộc lớp đó.

# CHƯƠNG 4 CÁC PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN

## 4.1 Đặc trưng HOG

Histogram of Oriented Gradients (HOG) [1][5] là một phương pháp phổ biến được sử dụng trong nhận dạng và phát hiện đối tượng trong lĩnh vực thị giác máy tính. HOG dựa trên việc phân tích sự phân bố của cường độ gradient hoặc hướng cạnh để mô tả hình dạng và cấu trúc của đối tượng trong ảnh.

### 4.1.1 Nguyên lý hoạt động

HOG hoạt động bằng cách tính gradient của cường độ ánh sáng để xác định các cạnh và biên của đối tượng trong ảnh. Quá trình này bao gồm các bước chính sau:

- Tính Gradient:
  - Sử dụng bộ lọc Sobel để tính toán gradient theo hai hướng x và y.
  - Tính toán magnitude và direction của gradient từ gradient x và y.
- Tạo ô (cell) và khối (block):
  - Chia ảnh thành các ô nhỏ (thường là 8x8 pixel).
  - Nhóm các ô lại thành các khối (thường là 2x2 ô).
- Tạo histogram hướng trong mỗi ô:
  - Đối với mỗi ô, tạo histogram của hướng gradient.
  - Hướng gradient được chia thành các bin (thường là 9 bin, mỗi bin ứng với khoảng góc 20 độ).
- Chuẩn hóa khối:
  - Chuẩn hóa histogram của các ô trong mỗi khối để giảm thiểu ảnh hưởng của sự thay đổi ánh sáng và độ tương phản.
  - Thông thường, sử dụng L2-norm để chuẩn hóa.
- Kết hợp đặc trưng:
  - Tạo vector đặc trưng HOG cuối cùng bằng cách kết hợp các histogram chuẩn hóa từ tất cả các khối.

#### 4.1.2 Áp dụng HOG trong bài toán phân loại biển báo giao thông

Trong bài toán phân loại biển báo giao thông, HOG được áp dụng để trích xuất đặc trưng hình dạng và cạnh từ ảnh xám của biển báo. Mã nguồn Python dưới đây minh họa quá trình này:

- Đoạn code sử dụng hàm **hog** có sẵn trong thư viện **sklearn.feature**.

```
def extract_hog_features(image):

    gray = cv.cvtColor(image, cv.COLOR_RGB2GRAY)

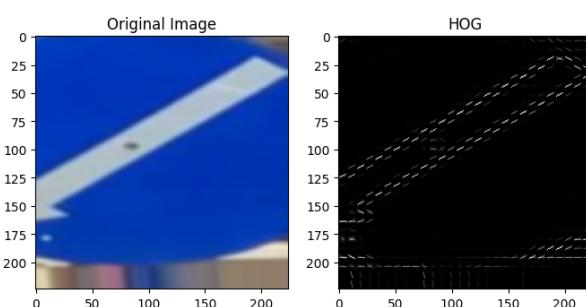
    hog_features, hog_image= hog(gray, orientations=9,
        pixels_per_cell=(8, 8), cells_per_block=(2, 2),
        visualize=True, block_norm='L2-Hys', transform_sqrt=True
    )

    hog_image_rescaled= exposure.rescale_intensity(hog_image,
        in_range=(0, 10))

    return hog_features, hog_image_rescaled
```

Mã 4.1: Đoạn mã đặc trưng HOG

- Tham số sử dụng:
  - `orientations=9`: Sử dụng 9 bins cho histogram hướng gradient.
  - `pixels_per_cell=(8, 8)`: Mỗi ô có kích thước 8x8 pixel.
  - `cells_per_block=(2, 2)`: Mỗi khối bao gồm 2x2 ô.
  - `block_norm='L2-Hys'`: Chuẩn hóa khối bằng phương pháp L2-Hys.
  - `transform_sqrt=True`: Sử dụng căn bậc hai cho các giá trị trước khi chuẩn hóa.
- Kết quả trích xuất đặc trưng HOG trên ảnh biển báo giao thông:



Hình 4.3: Ảnh biển báo giao thông sau khi trích xuất đặc trưng HOG

#### **4.1.3 Ưu điểm và nhược điểm của HOG**

- **Ưu điểm:**
  - Độ chính xác cao trong việc nhận diện các đặc trưng cục bộ của đối tượng.
  - Khả năng chống nhiễu tốt, tập trung vào cấu trúc và hình dạng thay vì màu sắc.
  - Tạo ra biểu đồ phân phối hướng gradient, hiệu quả trong việc mô tả hình dạng và cấu trúc của đối tượng.
- **Nhược điểm:**
  - Nhạy cảm với màu sắc vì chỉ sử dụng thông tin từ ảnh xám: Thông tin về màu sắc có thể bị mất và làm giảm độ chính xác của quá trình nhận diện và phân loại đối tượng.
  - Ảnh hưởng bởi sự thay đổi ánh sáng và độ tương phản trong ảnh: Sự thay đổi độ sáng giữa các pixel trong ảnh có thể ảnh hưởng đến quá trình tính toán gradient và làm giảm độ chính xác của HOG trong việc trích xuất đặc trưng.
  - Khó khăn trong việc xử lý các đối tượng có biến dạng lớn hoặc không đồng nhất.

## 4.2 Color Histogram

### 4.2.1 Tổng quan về Color Histogram

Color histogram [10][6] là một công cụ mạnh mẽ được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và thị giác máy tính. Nó là biểu đồ tần suất biểu diễn sự phân bố của các màu sắc trong một hình ảnh. Mỗi màu sắc trong hình ảnh được phân loại vào một trong các bin (ngắn) của histogram, và số lượng các điểm ảnh có màu sắc đó được đếm và hiển thị. Color histogram có thể được xây dựng cho các không gian màu khác nhau như RGB, HSV, hoặc Lab.

### 4.2.2 Nguyên lý hoạt động

Color Histogram dựa trên việc phân chia không gian màu của hình ảnh thành các khoảng màu (bins) và đếm số lượng pixel trong mỗi khoảng màu đó. Quá trình tạo ra một Color Histogram có thể được tóm tắt qua các bước sau:

- Chọn không gian màu:
  - Hình ảnh có thể được biểu diễn trong nhiều không gian màu khác nhau như RGB, HSV, hoặc Lab.
  - Mỗi không gian màu có những đặc điểm riêng và có thể ảnh hưởng đến hiệu quả của Color Histogram.
- Phân chia không gian màu:
  - Không gian màu được chia thành các khoảng màu rời rạc. Số lượng khoảng màu (bins) này có thể điều chỉnh dựa trên yêu cầu cụ thể của ứng dụng.
  - Thường thì càng nhiều bins, thông tin màu sắc được biểu diễn càng chi tiết, nhưng yêu cầu tài nguyên tính toán cũng tăng lên.
- Đếm tần suất màu sắc:
  - Mỗi pixel trong hình ảnh được ánh xạ vào một khoảng màu cụ thể
  - Histogram sau đó được tạo ra bằng cách đếm số lượng pixel thuộc về mỗi khoảng màu.
- Chuẩn hóa và làm phẳng Histogram: Sau khi tạo ra histogram, việc chuẩn hóa và làm phẳng dữ liệu là cần thiết để đảm bảo rằng giá trị histogram nằm trong một phạm vi nhất định và có thể so sánh được giữa các hình ảnh khác nhau.

#### 4.2.3 Áp dụng Color Histogram trong bài toán phân loại biển báo giao thông

Trong bài toán phân loại biển báo giao thông, Color Histogram có thể được sử dụng để nhận diện và phân loại các loại biển báo khác nhau dựa trên màu sắc đặc trưng của chúng. Mã nguồn Python dưới đây minh họa cho quá trình này:

- Đoạn code sử dụng hàm **calcHist** có sẵn trong thư viện **cv2**.

```
def extract_color_histogram(image):

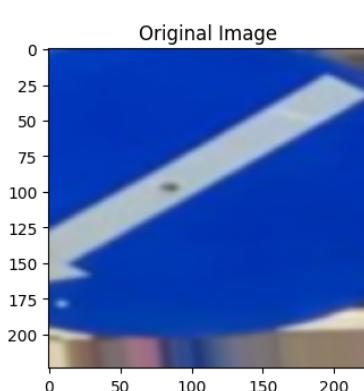
    hist = cv.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0,
        256, 0, 256, 0, 256])

    hist = cv.normalize(hist, hist).flatten()

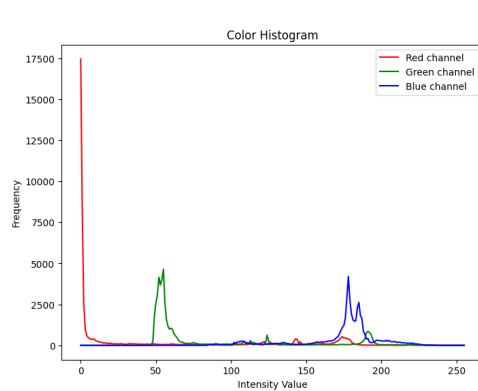
    return hist
```

Mã 4.2: Đoạn mã đặc trưng Color Histogram

- Tham số sử dụng:
  - [0, 1, 2]: Sử dụng cả ba kênh màu (RGB) để tính toán histogram.
  - [8, 8, 8]: Chia mỗi kênh màu thành 8 bins, tổng cộng có  $8 \times 8 \times 8 = 512$  bins.
  - [0, 256, 0, 256, 0, 256]: Phạm vi giá trị cho mỗi kênh màu từ 0 đến 256.
  - normalize: Chuẩn hóa histogram để các giá trị nằm trong khoảng từ 0 đến 1.
  - flatten: Làm phẳng mảng histogram thành một mảng một chiều để dễ dàng xử lý và so sánh.
- Biểu đồ của Color Histogram trên ảnh biển báo giao thông:



Hình 4.4: Ảnh gốc



Hình 4.5: Biểu đồ histogram màu

#### **4.2.4 Ưu điểm và nhược điểm của Color Histogram**

- **Ưu điểm:**
  - Đơn giản và dễ thực hiện.
  - Không phụ thuộc vào kích thước ảnh: Vì histogram chỉ phụ thuộc vào tần suất màu sắc, nó không bị ảnh hưởng bởi kích thước và tỉ lệ của hình ảnh.
  - Tính cục bộ: Color Histogram chỉ quan tâm đến sự phân bố màu sắc trên toàn bộ hình ảnh, mà không cần biết sự phân bố không gian hay vị trí cụ thể của các vùng màu.
- **Nhược điểm:**
  - Không có thông tin về cấu trúc không gian: Histogram màu không phản ánh thông tin về mối quan hệ không gian giữa các điểm ảnh.
  - Nhạy cảm với sự thay đổi đối với độ sáng và tương phản: Không phân biệt được các biến đổi nhỏ về độ sáng và tương phản của hình ảnh.
  - Không phù hợp với các vấn đề phức tạp: Trong những trường hợp mô hình màu sắc phức tạp hơn (như sự thay đổi nhiều màu sắc), histogram màu có thể không còn đủ hiệu quả.

## 4.3 Kết hợp giữa HOG và Color Histogram

HOG không biểu diễn thông tin về màu sắc trong khi Color Histogram không biểu diễn thông tin về hình dạng và cấu trúc của đối tượng. Để tận dụng các ưu điểm của cả HOG và Color Histogram và khắc phục nhược điểm của từng phương pháp, chúng ta có thể kết hợp chúng lại để trích xuất đặc trưng cho hình ảnh biển báo giao thông.

### 4.3.1 Áp dụng HOG và Color Histogram trong bài toán phân loại biển báo giao thông

Để kết hợp hai đặc trưng này, chúng em sử dụng một hàm như sau để trích xuất và kết hợp các đặc trưng từ hình ảnh:

```
def combined_features(image):
    hog_features, hog_image_rescaled = extract_hog_features(image)

    hist = extract_color_histogram(image)

    combined_features = np.hstack((hog_features, hist))

    return combined_features
```

Mã 4.3: Kết hợp HOG và Color Histogram

Trong đó:

- `hog_feature`: Là vector đặc trưng được trích xuất từ HOG của hình ảnh.
- `hist`: Là vector đặc trưng được trích xuất từ Color Histogram của hình ảnh.
- `np.hstack`: Dùng để ghép hai đặc trưng trên theo chiều ngang, tạo thành một vector `combined_features` chứa thông tin từ cả HOG và Color Histogram.

## 4.4 Thuật toán K-Nearest Neighbors

### 4.4.1 Tổng quan thuật toán

Thuật toán K-Nearest Neighbors (KNN) [3] là một trong những thuật toán đơn giản và phổ biến nhất trong lĩnh vực học máy và thị giác máy tính. Được giới thiệu bởi Cover và Hart vào năm 1967, KNN là một phương pháp phân loại và hồi quy dựa trên việc tìm kiếm các điểm dữ liệu gần nhất trong không gian đặc trưng. Khi cần dự đoán nhãn cho một mẫu mới, KNN sẽ tìm ra k điểm dữ liệu gần nhất (k-nearest neighbors) trong tập train và sử dụng thông tin từ các điểm này để đưa ra dự đoán.

### 4.4.2 Nguyên lý hoạt động

- Nguyên lý hoạt động của KNN bao gồm các bước sau:

- **Tính khoảng cách:** KNN tính toán khoảng cách giữa điểm dữ liệu cần dự đoán và tất cả các điểm trong tập huấn luyện. Các loại khoảng cách phổ biến được sử dụng bao gồm khoảng cách Euclidean, Manhattan và Minkowski.
  - \* **Khoảng cách Euclidean:** Đo khoảng cách đường thẳng giữa hai điểm trong không gian.

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (4.2)$$

- **Khoảng cách Manhattan:** Đo khoảng cách tuyệt đối giữa hai điểm trong không gian.

$$d(p, q) = \sum_{i=1}^n |p_i - q_i| \quad (4.3)$$

- **Khoảng cách Minkowski:** Là dạng tổng quát của khoảng cách Euclidean và Manhattan.

$$d(p, q) = \left( \sum_{i=1}^n |p_i - q_i|^p \right)^{\frac{1}{p}} \quad (4.4)$$

- **Tìm k điểm gần nhất:** Chọn K điểm dữ liệu có khoảng cách ngắn nhất tới điểm dữ liệu mới.

- **Dự đoán nhãn:**

- \* **Phân loại (Classification):** KNN sẽ dựa vào nhãn của k điểm gần nhất để quyết định nhãn của mẫu mới. Phương pháp phổ biến là sử dụng biểu quyết đa số (majority voting).
- \* **Hồi quy (Regression):** KNN sẽ tính giá trị trung bình của các nhãn từ k điểm gần nhất để đưa ra giá trị dự đoán.

#### 4.4.3 Tham số

- **k\_neighbors:** :
  - Đây là tham số chính của KNN, quyết định số lượng điểm gần nhất được xem xét.
  - Giá trị k nhỏ có thể dẫn đến mô hình bị quá khớp (overfitting), trong khi giá trị k lớn có thể dẫn đến mô hình bị thiếu khớp (underfitting).
- **metric:** Lựa chọn loại khoảng cách tính toán (Euclidean, Manhattan, Minkowski, v.v.) cũng ảnh hưởng đến hiệu quả của mô hình.
- **weights:** Một số biến thể của KNN sử dụng trọng số cho các điểm láng giềng, ví dụ như các điểm gần hơn có trọng số cao hơn trong quá trình dự đoán

#### 4.4.4 Kết quả

- Với tham số mặc định:
  - Báo cáo chi tiết:
- Accuracy: 0.647

Cam	0.81	0.81	0.81	67
Chidan	0.71	0.31	0.43	49
Hieulenh	0.38	0.75	0.51	52
Nguyhiem	1.00	0.80	0.89	51
Phu	0.67	0.40	0.50	30
accuracy			0.65	249
macro avg	0.71	0.61	0.63	249
weighted avg	0.72	0.65	0.65	249

Bảng 4. 1: Bảng báo phân loại với tham số KNN mặc định

- Với tham số tối ưu: metric = 'euclidean', n\_neighbors = 3, weights = 'distance'.
  - Accuracy: 0.707
  - Báo cáo chi tiết:

	precision	recall	f1-score	support
Cam	0.84	0.87	0.85	67
Chidan	0.76	0.39	0.51	49
Hieulenh	0.44	0.83	0.58	52
Nguyhiem	1.00	0.82	0.90	51
Phu	0.88	0.47	0.61	30
accuracy			0.71	249
macro avg		0.78	0.67	249
weighted avg		0.78	0.71	249

**Bảng 4. 2:** Bảng báo phân loại với tham số KNN tối ưu

- Nhận xét:

- So sánh giữa mô hình KNN mặc định và mô hình tối ưu, ta thấy rằng mô hình tối ưu có độ chính xác cao hơn (0.707 so với 0.647)
- Việc tối ưu hóa mô hình KNN bằng cách sử dụng GridSearchCV[7] với các tham số như metric = 'euclidean', n\_neighbors = 3, weights = 'distance' đã mang lại hiệu quả rõ rệt trong việc cải thiện khả năng phân loại của mô hình so với tham số mặc định.

## 4.5 Thuật toán Decision Tree

### 4.5.1 Tổng quan thuật toán

- Decision Tree (Cây quyết định)[4] [8] là một trong những thuật toán quan trọng và phổ biến nhất trong lĩnh vực machine learning. Nó được sử dụng rộng rãi cho cả bài toán phân loại và hồi quy trong các ứng dụng thực tế. Với tính linh hoạt và khả năng diễn giải cao, cây quyết định thường được ưa chuộng cho việc áp dụng trong các lĩnh vực từ y tế, tài chính, đến marketing và hơn thế nữa.
- Một cây quyết định được xem như một cấu trúc dữ liệu cây, gồm các nút và các cạnh. Mỗi nút trong cây biểu diễn một "kiểm tra" trên một thuộc tính dữ liệu, dẫn đến việc chia dữ liệu thành các nhánh con. Các nút lá của cây đại diện cho các nhãn hoặc giá trị dự đoán cuối cùng.

### 4.5.2 Nguyên lý hoạt động

Quá trình xây dựng cây bắt đầu từ nút gốc và tiếp tục đệ quy xuống từng nút con cho đến khi một điều kiện dừng được đạt đến. Trong quá trình này, thuật toán sẽ lựa chọn thuộc tính tốt nhất để chia dữ liệu tại mỗi bước, thường dựa trên một tiêu chí như entropy hoặc độ thuần khiết Gini. Mục tiêu là tạo ra các nhánh cây sao cho mỗi nhánh chứa các mẫu càng thuần khiết hoặc ít không chắc chắn càng tốt.

### 4.5.3 Tham số

- **Max Depth (Độ sâu tối đa):**

- Tham số này xác định độ sâu tối đa mà một cây quyết định có thể phát triển. Nếu độ sâu của cây vượt qua giới hạn này, việc phát triển cây sẽ dừng lại ngay cả khi vẫn còn thuộc tính có thể chia.
  - Một giá trị max depth thấp có thể dẫn đến một cây quyết định đơn giản hóa, dễ hiểu, nhưng cũng có nguy cơ mất đi tính tổng quát và dẫn đến underfitting. Trong khi đó, max depth cao hơn có thể tạo ra một mô hình phức tạp hơn, có khả năng mô tả dữ liệu huấn luyện tốt hơn, nhưng cũng dễ bị overfitting với dữ liệu mới.

- **Min Sample Split (Số lượng mẫu tối thiểu để chia nút):**

- Tham số này xác định số lượng mẫu tối thiểu mà một nút phải có trước khi có thể chia thành các nút con.

- Một giá trị min sample split thấp có thể dẫn đến việc tạo ra các nhánh cây chứa ít dữ liệu hơn, làm tăng nguy cơ overfitting. Trái lại, giá trị cao hơn sẽ tạo ra các nhánh cây lớn hơn, giảm nguy cơ overfitting nhưng cũng có thể làm giảm khả năng mô hình hóa dữ liệu huấn luyện.
- Min Sample Leaf (Số lượng mẫu tối thiểu trong nút lá):**
  - Tham số này xác định số lượng mẫu tối thiểu mà một nút lá phải có.
  - Một giá trị min sample leaf thấp có thể dẫn đến việc tạo ra các nút lá chứa ít dữ liệu hơn, tăng cơ hội overfitting. Ngược lại, giá trị cao hơn có thể dẫn đến các nút lá lớn hơn và một cây quyết định đơn giản hơn.
- Criterion (Tiêu chí chia nút):**
  - Tiêu chí này xác định cách đo lường "chất lượng" của việc chia nút. Các tiêu chí phổ biến bao gồm entropy và độ thuần khiết Gini.
  - Entropy đo lường mức độ không chắc chắn trong tập dữ liệu, trong khi Gini impurity đo lường mức độ không thuần khiết của các nhãn trong nút. Cả hai đều được sử dụng để tối ưu hóa việc chọn thuộc tính tốt nhất để chia nút.

#### 4.5.4 Kết quả

- Với tham số mặc định:
- Báo cáo chi tiết:
  - Accuracy: 0.719

	precision	recall	f1-score	support
Cam	0.76	0.79	0.77	67
Chidan	0.58	0.63	0.61	49
Hieulenh	0.62	0.58	0.60	52
Nguyhiem	0.92	0.88	0.90	51
Phu	0.69	0.67	0.68	30
accuracy			0.72	249
macro avg	0.72	0.71	0.71	249
weighted avg	0.72	0.72	0.72	249

**Bảng 4. 3:** Bảng báo phân loại với tham số Decison Tree mặc định

- Với tham số tối ưu: max\_depth = 10, criterion = 'entropy', min\_samples\_split = 2, min\_samples\_leaf = 4.
  - Accuracy: 0.771
  - Báo cáo chi tiết:

	precision	recall	f1-score	support
Cam	0.81	0.93	0.86	67
Chidan	0.60	0.80	0.68	49
Hieulenh	0.79	0.50	0.61	52
Nguyhiem	0.94	0.86	0.90	51
Phu	0.78	0.70	0.74	30
accuracy			0.77	249
macro avg	0.78	0.76	0.76	249
weighted avg	0.78	0.77	0.77	249

**Bảng 4. 4:** Bảng báo phân loại với tham số Decision Tree tối ưu

- Nhập xét:
  - So sánh giữa mô hình Decision Tree mặc định và mô hình tối ưu, ta thấy rằng mô hình tối ưu có độ chính xác cao hơn (0.771 so với 0.719)
  - Việc tối ưu hóa mô hình Decision Tree bằng cách sử dụng GridSearchCV[7] với các tham số như max\_depth = 10, criterion = 'entropy', min\_samples\_split = 2, min\_samples\_leaf = 4 đã mang lại hiệu quả rõ rệt trong việc cải thiện khả năng phân loại của mô hình so với tham số mặc định.

## 4.6 Thuật toán Support Vector Machine

### 4.6.1 Tổng quan thuật toán

- SVM (Support Vector Machine)[2] [9] là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification) hay hồi quy (Regression).
- Mục tiêu của SVM là tìm ra một siêu phẳng trong không gian N chiều (ứng với N đặc trưng). Ví dụ, trong không gian 2 chiều, thì tìm ra 1 đường thẳng sao cho chia dữ liệu thành 2 phần, trong không gian 3 chiều thì tìm 1 mặt phẳng chia dữ liệu thành hai phần tương ứng với lớp của chúng.
- Để phân chia hai lớp dữ liệu, rõ ràng là có rất nhiều siêu phẳng có thể làm được điều này. Mặc dù vậy, mục tiêu của SVM là tìm ra siêu phẳng có lề rộng nhất tức là có khoảng cách tới các điểm của hai lớp là lớn nhất. Điều này được thực hiện bằng cách tối ưu hóa các hệ số của siêu phẳng, thông qua việc truyền vào các tham số cho hàm svm.

### 4.6.2 Nguyên lý hoạt động

- **Tìm siêu phẳng phân chia dữ liệu.**

SVM hoạt động bằng cách tìm kiếm một siêu phẳng (hyperplane) trong không gian nhiều chiều để phân chia dữ liệu thành các lớp khác nhau. Trong không gian hai chiều, siêu phẳng này là một đường thẳng, trong không gian ba chiều, nó là một mặt phẳng, và trong không gian nhiều chiều hơn, nó là một siêu phẳng.

- **Tối đa hóa khoảng cách biên.**

Mục tiêu của SVM là tìm ra siêu phẳng mà tối đa hóa khoảng cách giữa các điểm dữ liệu của hai lớp gần nhất, được gọi là các điểm hỗ trợ (support vectors). Khoảng cách giữa siêu phẳng và các điểm hỗ trợ này được gọi là khoảng cách biên (margin). Một siêu phẳng có khoảng cách biên lớn hơn sẽ có khả năng phân loại dữ liệu mới chính xác hơn.

- **Hàm mục tiêu.**

SVM tìm kiếm siêu phẳng phân chia bằng cách giải bài toán tối ưu hóa. Hàm mục tiêu của bài toán này là:

$$\min \frac{1}{2} \|w\|^2 \quad (4.5)$$

với ràng buộc:

$$y_i(w \cdot x_i + b) \geq 1, \forall i \quad (4.6)$$

Trong đó:

- $w$  là vector trọng số.
- $x_i$  là các điểm dữ liệu.
- $y_i$  là nhãn của các điểm dữ liệu (+1+1+1 hoặc -1-1-1).
- $b$  là độ lệch.

#### 4.6.3 Tham số

- **Kernel:** là một hàm ánh xạ dữ liệu từ không gian ít chiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Kernel gồm nhiều loại :

**- Tuyến tính (Linear):**

- \* Kernel tuyến tính sử dụng phép tích tích vô hướng giữa hai điểm dữ liệu để đo độ tương đồng.
- \* Phù hợp cho dữ liệu có biên phân loại rõ ràng khi các lớp là tách biệt tuyến tính.
- \* Dễ huấn luyện và có thể hiệu quả trên dữ liệu lớn.

**- Đa thức (Polynomial):**

- \* Kernel đa thức tính toán tương đồng dựa trên một đa thức của các tích vô hướng.
- \* Có thể mở rộng không gian đặc trưng của dữ liệu, giúp phân loại các điểm dữ liệu không tách biệt tuyến tính.
- \* Độ phức tạp của kernel đa thức tăng theo bậc của đa thức.

**- RBF (Radial Basis Function):**

- \* RBF kernel đo độ tương đồng dựa trên khoảng cách Euclidean giữa các điểm dữ liệu.
- \* Kernel này tạo ra một không gian đặc trưng phi tuyến tính, giúp phân loại dữ liệu không tách biệt tuyến tính.
- \* Có thể điều chỉnh thông qua tham số gamma để kiểm soát độ phức tạp của mô hình.

**- Sigmoid:**

- \* Kernel sigmoid dựa trên hàm sigmoid của tích vô hướng giữa hai điểm dữ liệu.
- \* Thường được sử dụng trong mô hình mạng nơ-ron nhân tạo, nhưng không phổ biến trong SVM.
- \* Có thể được sử dụng để ánh xạ dữ liệu vào không gian đặc trưng mới.

- **C:** Mức độ chấp nhận lỗi - C càng lớn có nghĩa là SVM càng bị phạt nặng khi thực hiện phân loại sai. Do đó, lề càng hẹp và càng ít vectơ hỗ trợ được sử dụng.
  - **C lớn:** Khi C có giá trị lớn, SVM sẽ cố gắng phân loại chính xác tất cả các điểm dữ liệu trong tập huấn luyện, kể cả khi điều này dẫn đến một biên phân cách phức tạp và nhỏ hẹp. Mô hình sẽ bị phạt nặng nếu có bất kỳ lỗi phân loại nào. Điều này có thể dẫn đến overfitting (quá khớp), tức là mô hình hoạt động rất tốt trên tập huấn luyện nhưng kém hiệu quả trên dữ liệu mới.
  - **C nhỏ:** Khi C có giá trị nhỏ, SVM sẽ cho phép một số lỗi phân loại trên tập huấn luyện nếu điều này giúp tạo ra một biên phân cách đơn giản và mượt mà hơn. Mô hình sẽ có biên phân cách rộng hơn, và có thể dẫn đến underfitting (chưa khớp), tức là mô hình có thể không nắm bắt được hết các đặc trưng của dữ liệu.
- **Gamma:** tham số gamma được sử dụng trong kernel RBF (Radial Basis Function). Tham số này ảnh hưởng đến độ phức tạp của mô hình SVM, cụ thể là đường ranh giới quyết định.
  - **Ảnh hưởng của gamma cao:** Nếu gamma quá cao, mô hình sẽ học rất chi tiết từ tập huấn luyện, dẫn đến việc quá khớp (overfitting). Biên quyết định sẽ rất cong và phức tạp, làm cho mô hình không tổng hóa tốt trên dữ liệu mới. Biên quyết định sẽ cố gắng chia tách từng điểm dữ liệu một cách rõ ràng, dẫn đến một đường phân tách rất chi tiết và phức tạp.
  - **Ảnh hưởng của gamma thấp:** Nếu gamma quá thấp, mô hình sẽ không đủ linh hoạt để nắm bắt cấu trúc dữ liệu phức tạp, dẫn đến việc mô hình quá đơn giản (underfitting). Biên quyết định sẽ rất mượt và có thể không phân loại tốt các điểm dữ liệu. Biên quyết định sẽ trở nên rất mượt và ít nhạy cảm với các điểm dữ liệu cá biệt, dẫn đến một đường phân tách đơn giản hơn.

#### 4.6.4 Kết quả

- Với tham số mặc định:
- Báo cáo chi tiết:
  - Accuracy: 0.767

	precision	recall	f1-score	support
Cam	0.75	0.90	0.82	67
Chidan	0.68	0.78	0.72	49
Hieulenh	0.74	0.50	0.60	52
Nguyhiem	1.00	0.90	0.95	51
Phu	0.66	0.70	0.68	30
accuracy			0.77	249
macro avg	0.77	0.75	0.75	249
weighted avg	0.77	0.77	0.76	249

Bảng 4. 5: Bảng báo phân loại với tham số SVM mặc định

- Với tham số tối ưu: Kernel = 'rbf, C = 100, gamma = 0.001.
- Accuracy: 0.815
- Báo cáo chi tiết:

	precision	recall	f1-score	support
Cam	0.82	0.96	0.88	67
Chidan	0.74	0.80	0.76	49
Hieulenh	0.77	0.65	0.71	52
Nguyhiem	1.00	0.90	0.95	51
Phu	0.71	0.67	0.69	30
accuracy			0.82	249
macro avg	0.81	0.79	0.80	249
weighted avg	0.82	0.82	0.81	249

Bảng 4. 6: Bảng báo phân loại với tham số SVM tối ưu

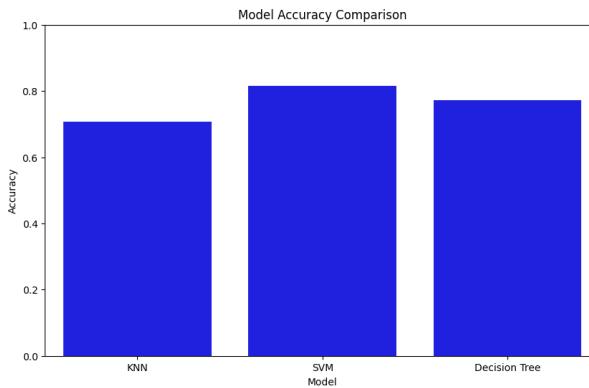
- Nhận xét:

- So sánh giữa mô hình mặc định và mô hình tối ưu, ta thấy rằng mô hình tối ưu đạt được độ chính xác (accuracy) cao hơn (0.815 so với 0.767).
- Việc tối ưu hóa mô hình SVM bằng cách sử dụng GridSearchCV[7] với các tham số như Kernel = 'rbf', C = 100 và gamma = 0.001 đã mang lại hiệu quả rõ rệt trong việc cải thiện khả năng phân loại của mô hình.

# CHƯƠNG 5. THÍ NGHIỆM VÀ KẾT QUẢ

## 5.1 Độ chính xác giữa 3 mô hình

- Biểu đồ so sánh:



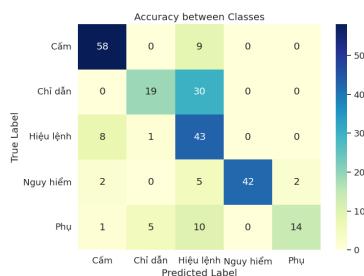
**Hình 5.6:** Biểu đồ thể hiện accuracy của 3 mô hình với tham số tối ưu

- Giữa 3 phương pháp là KNN, SVM và Decision Tree thì KNN là phương pháp có độ chính xác thấp nhất (0.707) và 2 phương pháp SVM (0.815) và Random Forest (0.771) thì phương pháp SVM có độ chính xác cao hơn phương pháp Random Forest khoảng 4.4%.

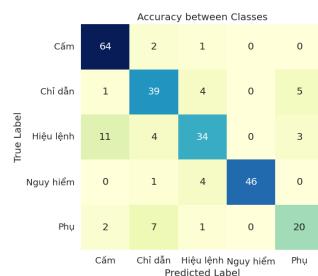
⇒ Phương pháp SVM có độ chính xác cao nhất.

## 5.2 Confusion Matrix

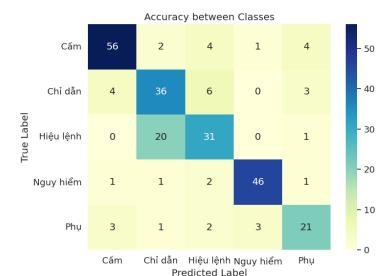
- Confusion Matrix của 3 mô hình:



**Hình 5.7:** KNN



**Hình 5.8:** SVM



**Hình 5.9:** Decision Tree

- Nhận xét:

- KNN:

\* Mô hình KNN có tỷ lệ dự đoán chính xác cao cho lớp "Cam" với 58/67 dự đoán đúng

- \* Tuy nhiên có sự nhầm lẫn cao giữa các lớp "Chidan" và "Hieulenh", với 30/49 mẫu "Chidan" bị nhầm lẫn thành "Hieulenh".

- SVM:

- \* Độ chính xác tốt hơn với 64/67 mẫu "Cam" được dự đoán đúng.
- \* Vẫn còn nhầm lẫn giữa "Hieulenh" và các lớp khác, với 11/52 mẫu "Hieulenh" bị nhầm lẫn thành "Cam".

- Decision Tree:

- \* Có sự nhầm lẫn cao giữa các lớp "Hieulenh" và "Chidan", với 23/52 mẫu "Hieulenh" bị nhầm lẫn thành "Chidan".
- \* Tuy nhiên, mô hình dự đoán đúng 39/49 mẫu "Chidan".

### 5.3 Demo

- Chúng em sẽ tiến hành dự đoán nhãn với cả 3 mô hình, kết quả dự đoán thu được như sau:



**Hình 5.10:** Dự đoán nhãn của ảnh mới với 3 mô hình KNN, SVM và Decision Tree

- Nhận xét:

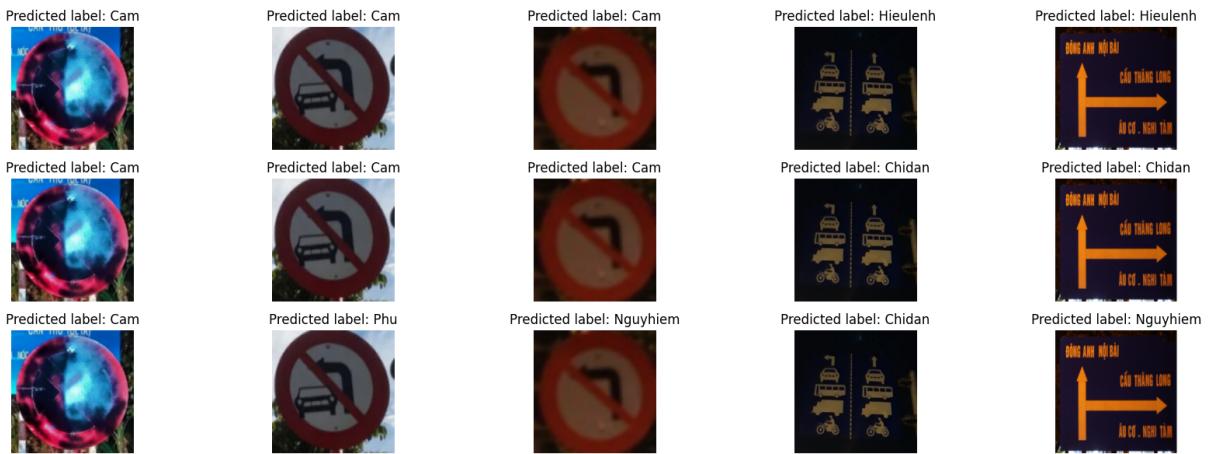
- KNN đã dự đoán đúng nhãn của biển báo trong ảnh là biển "Hieulenh".
- SVM và Decision Tree dự đoán sai biển báo trong ảnh là biển "Cam".

### 5.4 Thủ nghiệm đặc biệt

• **Ảnh điều kiện sáng không tốt:**

- Trong điều kiện tối, độ sáng tổng thể của ảnh giảm, làm cho các màu sắc trở nên mờ nhạt hoặc bị biến đổi. Điều này làm thay đổi phân bố màu sắc trong Color Histogram, khiến cho các đặc trưng màu sắc không còn phản ánh đúng màu sắc thực tế của biển báo.

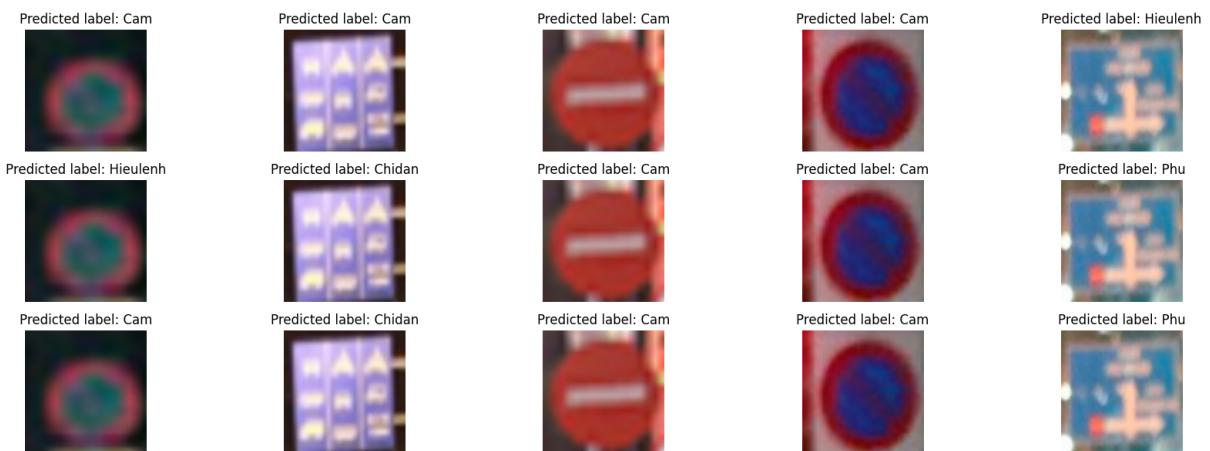
- Các chi tiết quan trọng của biển báo có thể bị mất đi hoặc không rõ ràng. Điều này ảnh hưởng đến các đặc trưng HOG vì các gradient có thể không còn rõ ràng như trong điều kiện ánh sáng tốt. Ảnh chụp trong điều kiện tối thường có nhiều nhiễu hơn. Điều này có thể làm biến dạng các đặc trưng mà mô hình dựa vào để phân loại, dẫn đến dự đoán sai.



**Hình 5.11:** Ảnh điều kiện sáng không tốt: KNN - SVM - Decision Tree

#### • Ảnh mờ:

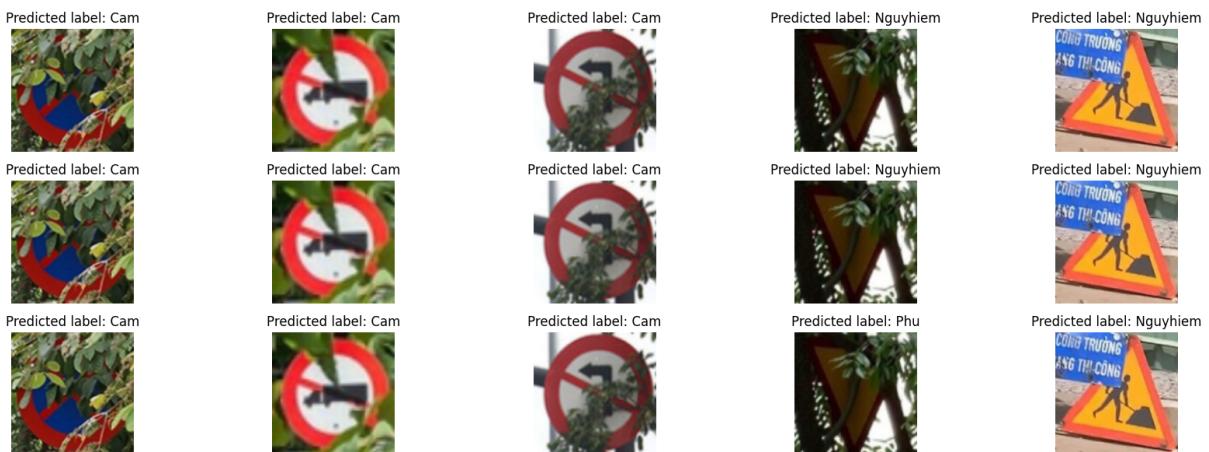
- Khi ảnh bị mờ, các màu sắc có thể bị pha trộn và trở nên ít rõ ràng hơn. Color Histogram, vốn dựa vào sự phân bố màu sắc trong ảnh, sẽ bị ảnh hưởng do sự pha trộn này.
- Khi ảnh bị mờ, các biên trở nên mờ nhạt hoặc bị làm mờ hoàn toàn, làm cho các gradient ít rõ ràng hơn. Điều này khiến cho HOG không thể trích xuất đúng các đặc trưng về hình dạng.



**Hình 5.12:** Ảnh mờ: KNN - SVM - Decision Tree

#### • Ảnh bị che một phần:

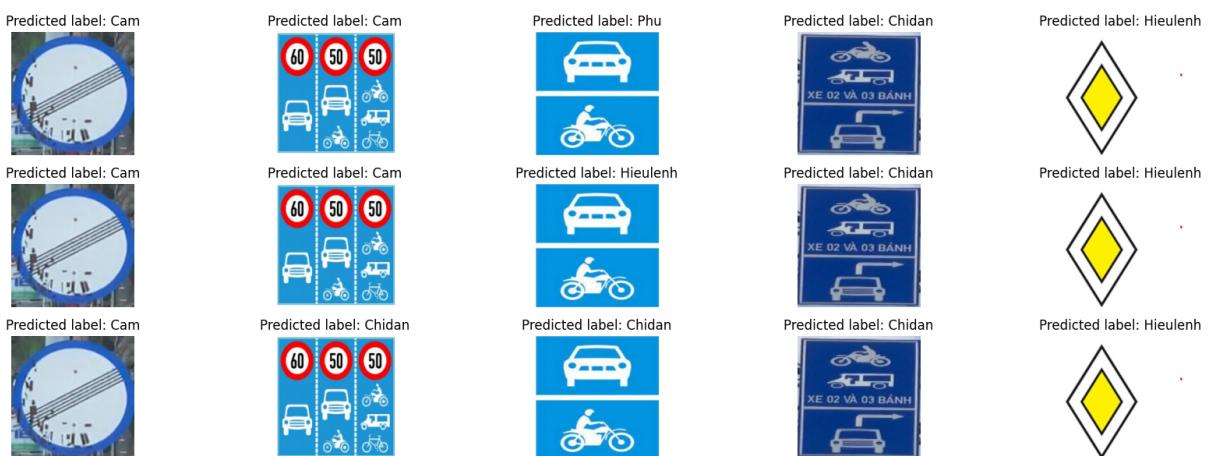
- Color Histogram dựa vào sự phân bố màu sắc trong toàn bộ ảnh. Khi một phần của biển báo bị che khuất, các màu sắc trong khu vực bị che sẽ không được tính vào histogram. Điều này dẫn đến việc phân bố màu sắc bị thay đổi, làm sai lệch đặc trưng màu sắc của biển báo.
- Khi một phần của biển báo bị che khuất, các biên và gradient trong khu vực bị che sẽ không còn xuất hiện trong ảnh. Điều này dẫn đến việc HOG không thể trích xuất đầy đủ các đặc trưng hình dạng cần thiết, làm giảm độ chính xác của mô hình trong việc nhận diện biển báo.



**Hình 5.13:** Ảnh bị che một phần: KNN - SVM - Decision Tree

#### • Biển dễ gây nhầm:

- Một số biển báo tuy khác nhau về nhãn nhưng lại có hình dạng khá tương đồng nhau, làm cho việc phân loại trở nên không chính xác.



**Hình 5.14:** Biển dễ gây nhầm: KNN - SVM - Decision Tree

# KẾT LUẬN

## Kết luận chung

- Với tập dataset đã được chuẩn bị thì trong 3 mô hình nhóm chọn thì mô hình SVM đạt độ chính xác cao nhất (80%).
- Nhãn cấm được phân loại tốt nhất trên cả 3 mô hình - có thể lí do:
  - Số lượng ảnh "Cam" có phần cao hơn các nhãn khác.
  - Biển báo "Cam" thường có viền và nền màu đỏ, dễ nhận diện do sự khác biệt màu sắc so với môi trường xung quanh. Màu đỏ này là đặc trưng mạnh mẽ, dễ nhận diện và ít bị nhầm lẫn với các biển báo khác.
- Biển "Hieulenh" và biển "Chidan" thường bị phân loại nhầm lẫn vì chúng có màu sắc và hình dạng khá tương đồng.
- Trong thực tế, với nhiều trường hợp hình ảnh biển báo có một số điều kiện chưa xuất hiện trong dataset, các mô hình vẫn chưa nắm bắt được hết và phân loại sai

## Hướng phát triển

- Mở rộng và cải thiện dataset:
  - Thu thập thêm dữ liệu: Tăng cường số lượng và đa dạng của các biển báo giao thông trong dataset, bao gồm các biến thể về điều kiện ánh sáng, thời tiết, và góc chụp khác nhau.
  - Dữ liệu từ nhiều nguồn: Sử dụng dữ liệu từ nhiều nguồn khác nhau, bao gồm cả hình ảnh từ các camera khác nhau, để tăng tính đa dạng và khả năng tổng quát hóa của mô hình.
- Cải thiện tiền xử lý dữ liệu:
  - Xử lý điều kiện ánh sáng: Áp dụng các kỹ thuật tiền xử lý như cân bằng sáng, lọc nhiễu, và tăng cường ảnh để cải thiện chất lượng hình ảnh đầu vào.
  - Phát hiện và loại bỏ nhiễu: Sử dụng các phương pháp phát hiện và loại bỏ nhiễu trong hình ảnh để cải thiện độ chính xác của mô hình.
- Nâng cấp mô hình học máy:
  - Sử dụng các mô hình học sâu: Triển khai các mô hình học sâu như Convolutional Neural Networks (CNNs) để cải thiện khả năng nhận diện và phân loại biển báo giao thông.

- Cải thiện đặc trưng trích xuất:
  - Kết hợp các phương pháp trích xuất đặc trưng: Kết hợp nhiều phương pháp trích xuất đặc trưng khác nhau để tận dụng ưu điểm của từng phương pháp.
  - Sử dụng các kỹ thuật trích xuất đặc trưng tiên tiến: Áp dụng các kỹ thuật trích xuất đặc trưng tiên tiến như SIFT, SURF, hoặc các kỹ thuật học sâu để cải thiện độ chính xác của mô hình.

## TÀI LIỆU THAM KHẢO

- [1] TS.Mai Tiến Dũng. “Histogram of Oriented Gradient HOG”. CS231. ĐHQG TPHCM: Nhập môn Thị giác máy tính, trường Đại học Công nghệ thông tin. URL: [https://courses.uit.edu.vn/pluginfile.php/548889/mod\\_resource/content/1/Bai05\\_Hog.p%20df](https://courses.uit.edu.vn/pluginfile.php/548889/mod_resource/content/1/Bai05_Hog.p%20df).
- [2] Tiep Vu Huu. *Bài 19: Support Vector Machine*. URL: <https://machinelearningcoban.com/2017/04/09/smv/>.
- [3] IBM. *What is the k-nearest neighbors (KNN) algorithm?* URL: <https://www.ibm.com/topics/knn>.
- [4] Tuấn Nguyễn. *Decision Tree algorithm*. URL: [https://machinelearningcoban.com/tabml\\_book/ch\\_model/decision\\_tree.html](https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html).
- [5] phamdinhkhanh. “Thuật toán HOG (Histogram of oriented gradient)”. In: (2019). URL: <https://phamdinhkhanh.github.io/2019/11/22/HOG.html>.
- [6] Inc Pinecone Systems. *Color Histograms in Image Retrieval*. URL: <https://www.pinecone.io/learn/series/image-search/color-histograms/>.
- [7] Rahul Shah. *Tune Hyperparameters with GridSearchCV*. URL: <https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/?>.
- [8] trituenhantao.io/. *Cây Quyết Định (Decision Tree)*. URL: <https://trituenhantao.io/kien-thuc/decision-tree/>.
- [9] Huynh Chi Trung. *Giới thiệu về Support Vector Machine (SVM)*. URL: <https://viblo.asia/p/gioi-thieu-ve-support-vector-machine-svm-6J3ZgPVElmB>.
- [10] Wikipedia contributors. *Color histogram*. 2023. URL: [https://en.wikipedia.org/wiki/Color\\_histogram](https://en.wikipedia.org/wiki/Color_histogram).