ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

NHẬP MÔN THỊ GIÁC MÁY TÍNH LỚP: CS231.P11

PHÂN LOẠI BIỂN BÁO GIAO THÔNG

Giáo viên hướng dẫn: TS. Mai Tiến Dũng

Sinh viên thực hiện:

STT	Họ và tên	MSSV
1	Phạm Đông Hưng	22520521
2	Phan Công Minh	22520884

LỜI CẢM ƠN

Chúng em xin chân thành gửi lời cảm ơn đến TS. Mai Tiến Dũng – Giảng viên khoa Khoa học máy tính, Trường Đại học Công nghệ thông tin, Đại học Quốc gia thành phố Hồ Chí Minh, đồng thời là giảng viên giảng dạy lớp CS231.P11 – Môn Nhập môn Thị giác máy tính, trong thời gian qua đã tận tình hướng dẫn và định hướng cho chúng em trong suốt quá trình thực hiện và hoàn thành đồ án.

Trong quá trình thực hiện đồ án chúng em đã cố gắng rất nhiều để hoàn thành đồ án một cách tốt nhất và hoàn thiện nhất, song cũng sẽ không tránh khỏi được những sai sót ngoài ý muốn. Chúng em mong rằng sẽ nhận được những lời nhận xét và những lời góp ý chân thành từ quý thầy và các bạn trong quá trình thực hiện chương tình của chúng em để chương trình ngày càng hoàn thiện hơn. Mỗi ý kiến đóng góp sẽ là một nguồn động lực to lớn đối với chúng em để chúng em có thể cố gắng cải tiến chương trình ngày càng hoàn thiện và phát triển đồ án lên một mức cao hơn, chúng em cũng sẽ dựa vào đó để phát triển hơn những ưu điểm và cải thiện được phần nào đó những nhược điểm của chương trình. Hy vọng đề tài "Phân loại biển báo giao thông" do chúng em thực hiện sẽ trở thành một chủ đề hữu ích và có thể ứng dụng được trong Khoa học máy tính và cũng như ở đời thực.

Thành phố Hồ Chí Minh, ngày 06 tháng 12 năm 2024

MỤC LỤC

Table of Contents

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Công việc	Minh	Hưng	Tiến độ
Thu thập thêm ảnh cho nhãn còn thiếu	X	X	Hoàn thành
Gán lại nhãn cho dữ liệu	X		Hoàn thành
Chọn các phương pháp xử lý ảnh	X	X	Hoàn thành
Sử dụng Random Forest		X	Hoàn thành
Sử dụng SVM	X		Hoàn thành
Tìm hiểu, cải thiện hiệu suất mô hình	X	X	Hoàn thành
Tìm hiểu phương pháp đánh giá		X	Hoàn thành
So sánh kết quả của các mô hình	X	X	Hoàn thành
Làm slide và viết báo cáo	X	X	Hoàn thành
Quản lý các file đồ án và nộp bài	X		Hoàn thành

1. Giới thiệu bài toán:

- Giao thông đường bộ hầu như là loại hình giao thông chủ yếu ở mọi nơi trên thế giới. Với sự phát triển không ngừng của xã hội thì lưu lượng người và phương tiện tham gia giao thông đang không ngừng tăng lên, dẫn đến yếu tố về tham gia giao thông đúng luật lệ và an toàn được đặt lên hàng đầu.
- Lý do chính lựa chọn đề tài: Các hệ thống giúp tự động phát hiện và nhận diện biển báo giao thông đem lại lợi ích to lớn, quan trọng không chỉ cho người điều khiển phương tiện mà còn trong các ứng dụng như hệ thống hỗ trợ lái xe tự động và xe tự hành, từ đó góp phần giúp cho giao thông đường bộ được diễn ra thuận lợi và an toàn.

2. Chuẩn bị dữ liệu:

2.1. Thu thập dữ liệu:

Tổng cộng: 10873 ảnh (gồm 9471 ảnh train, 785 ảnh valid, 617 ảnh test)

- Biển báo nguy hiểm (W - Warning): 3125 ảnh train, 235 ảnh valid, 176 ảnh test.
- Biển báo chỉ dẫn (I - Instruction) : 90 ảnh train, 10 ảnh valid, 7 ảnh test.
- Biển báo hiệu lệnh (R - Regulatory) : 1796 ảnh train, 114 ảnh valid, 109 ảnh test.
- Biển báo cấm (P - Prohibition): 6496 ảnh train, 416 ảnh valid, 327 ảnh test.
- Biển báo phụ (S - Supplementary): 90 ảnh train, 10 ảnh valid, 11 ảnh test.
- Một ảnh có thể chứa nhiều loại biển báo.
- Biển báo phụ và biển báo chỉ dẫn được thu thập, sau đó xử lý qua web roboflow.

2.2. Phát biểu bài toán:

Input:

- Dataset: gồm các ảnh chứa biển báo giao thông và nhãn tương ứng
- Ảnh chứa biển báo giao thông

Output: Nhãn dự đoán của biển báo có trong ảnh

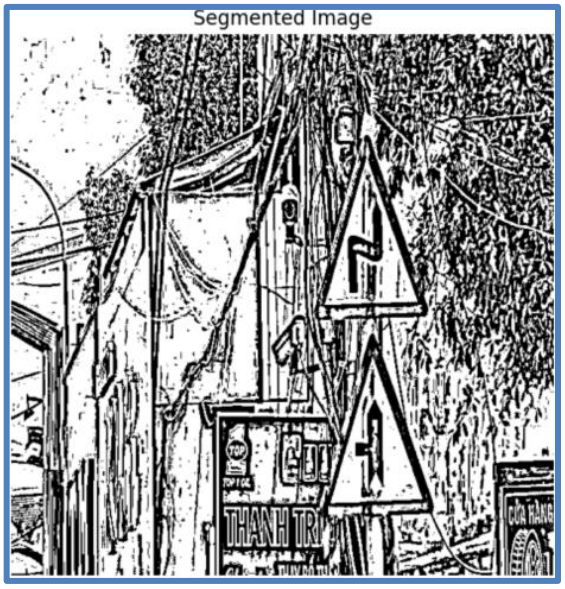
2.3. Tổng quan dữ liệu

- Dữ liệu được chia thành 3 tệp: Train, Valid. Test.
- Dữ liệu từ train được sử dụng để phân tích và huấn luyện mô hình máy học. Sau đó đánh giá trên dữ liệu valid và test.

2.4. Xử lý dữ liệu

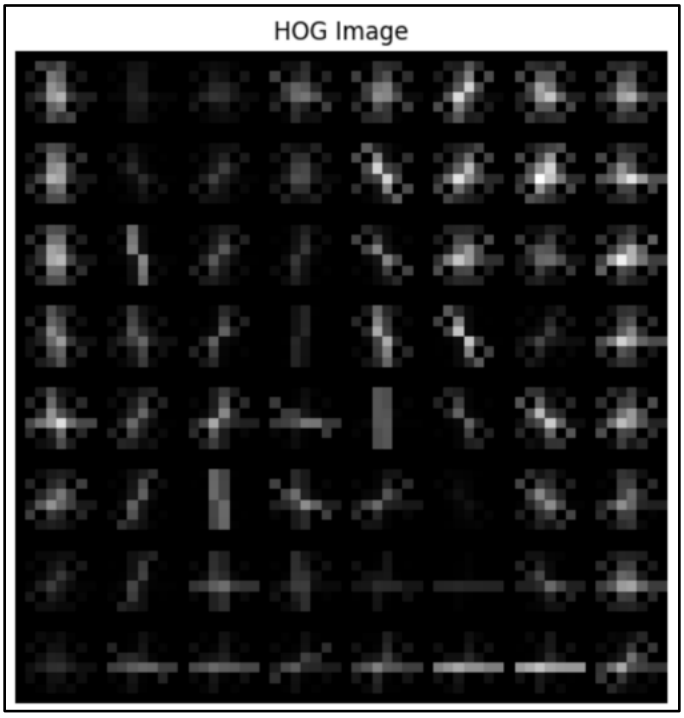
Image Conversion:

- Áp dụng Auto-Orientation cho dữ liệu pixel (loại bỏ EXIF-Orientation)
- Thay đổi kích thước hình ảnh thành 640 x 640 pixel Feature Extraction.
- Áp dụng Auto Contrast thông qua Contrast Streching.
- Áp dụng Adaptive Thresholding cho ảnh.



Feature Extraction:

- Sử dụng HOG cho việc trích xuất đặc trưng,
- Các tham số HOG:
  + Orientations: 9
  + Pixels per cell: (8, 8)
  + Cells per block: (2, 2)
  + Block norm: L2-Hys



Đặc trưng HOG

3. Các phương pháp cải thiện:

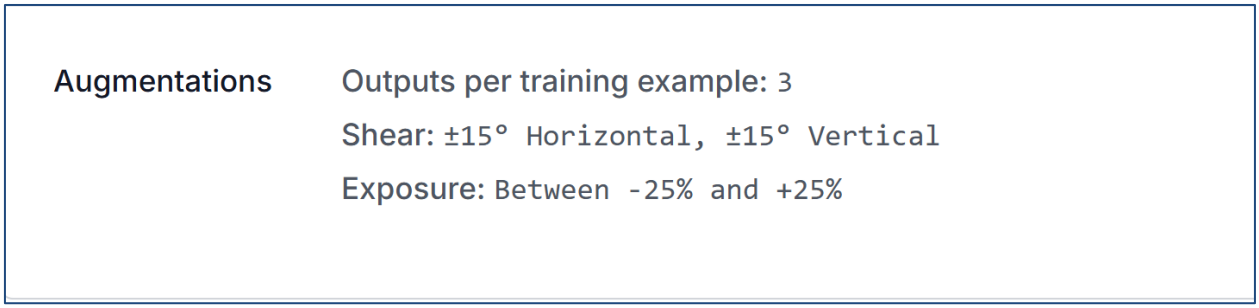
3.1. Data Augmentation

- Tăng cường dữ liệu (data augmentation) là phương pháp tạo ra các phiên bản mới từ dữ liệu hiện có, giúp mở rộng tập dữ liệu một cách nhân tạo.

- Phương pháp này đặc biệt hữu ích khi lượng dữ liệu đạo tạo hạn chế, vì nó có thể giúp mô hinh máy học học tốt hơn và giảm thiểu tình trạng overfitting.

- Đối với các ảnh tự thu thập (nhãn I và nhãn S):
  - Thu thập 50 ảnh mỗi nhãn từ Google Search, Bing Search.
  - Sử dụng web Roboflow để cắt bounding box và gắn nhãn cho biển báo có trong ảnh.
  - Chia theo tỷ lệ 30:10:10 cho từng tập train, test, valid.
  - Tăng thêm gấp 3 lần dữ liệu tập train từ 30 ảnh thành 90 ảnh, tổng cộng sẽ có 110 ảnh cho cả 3 tập train, test, valid.

- Các phương pháp được sử dụng:
  - Cắt ngẫu nhiên từ -15 đến +15 độ theo chiều ngang và chiều
  - Điều chỉnh phơi sáng ngẫu nhiên từ -25 đến 25%



3.2. GridSearchCV

- GridSearchCV là một công cụ mạnh mẽ dùng để tìm ra bộ hyperparameters tối ưu cho các mô hình học máy.
- Công cụ này hoạt động bằng cách thử nghiệm nhiều tổ hợp khác nhau của các giá trị hyperparameter, sau đó đánh giá hiệu suất của mô hình trên từng tổ hợp.

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Các tham số mà bạn muốn tìm kiếm
param_grid = {
    'n_estimators': [100, 120, 150],
    'max_depth': [5, 7],
    'min_samples_split': [2, 7, 10],
    'min_samples_leaf': [1, 2, 4, 7],
    'class_weight': ['balanced']
}

# Tạo mô hình Random Forest
rf_model = RandomForestClassifier(random_state=42)

# Tạo GridSearchCV
grid_search = GridSearchCV(estimator=rf_model,
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1,
                           verbose=2,
                           scoring='accuracy')

# Thực hiện tìm kiếm trên tập huấn luyện (X_train và y_train)
grid_search.fit(X_train, y_train)

# Hiển thị các tham số tối ưu
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

62m 15.5s

GridSearch Random Forest

```python
from sklearn.model_selection import GridSearchCV

def optimize_svm_hyperparameters(X_train, y_train):
    # Định nghĩa không gian tham số để tìm kiếm
    param_grid = {
        'C': [0.1, 1, 10, 100],        # Tham số điều chỉnh biên
        'kernel': ['linear', 'rbf'],   # Hàm kernel
        'gamma': ['scale', 'auto']     # Chỉ áp dụng với kernel RBF
    }

    # Khởi tạo mô hình SVM
    svm = SVC(probability=True)

    # Grid Search
    grid_search = GridSearchCV(
        svm,
        param_grid,
        cv=5,                # 5-fold cross-validation
        scoring='accuracy',  # Đo lường accuracy
        n_jobs=-1,           # Chạy song song với tất cả CPU
        verbose=1            # Hiển thị tiến trình
    )

    # Tìm kiếm tham số
    grid_search.fit(X_train, y_train)

    # Lấy tham số tốt nhất
    best_params = grid_search.best_params_
    logging.info(f"Best parameters found: {best_params}")
    print("Best Parameters:", best_params)

    # Trả về mô hình tốt nhất
    return grid_search.best_estimator_
```

GridSearch SVM

Kết quả:

- Mô hình Random Forest

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
C:\Users\Dong Hung\AppData\Roaming\Python\Python39\site-packages\numpy\ma\core.py:2846: RuntimeWarning: invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
Best parameters: {'class_weight': 'balanced', 'max_depth': 7, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
Best score: 0.8643202652251454
```

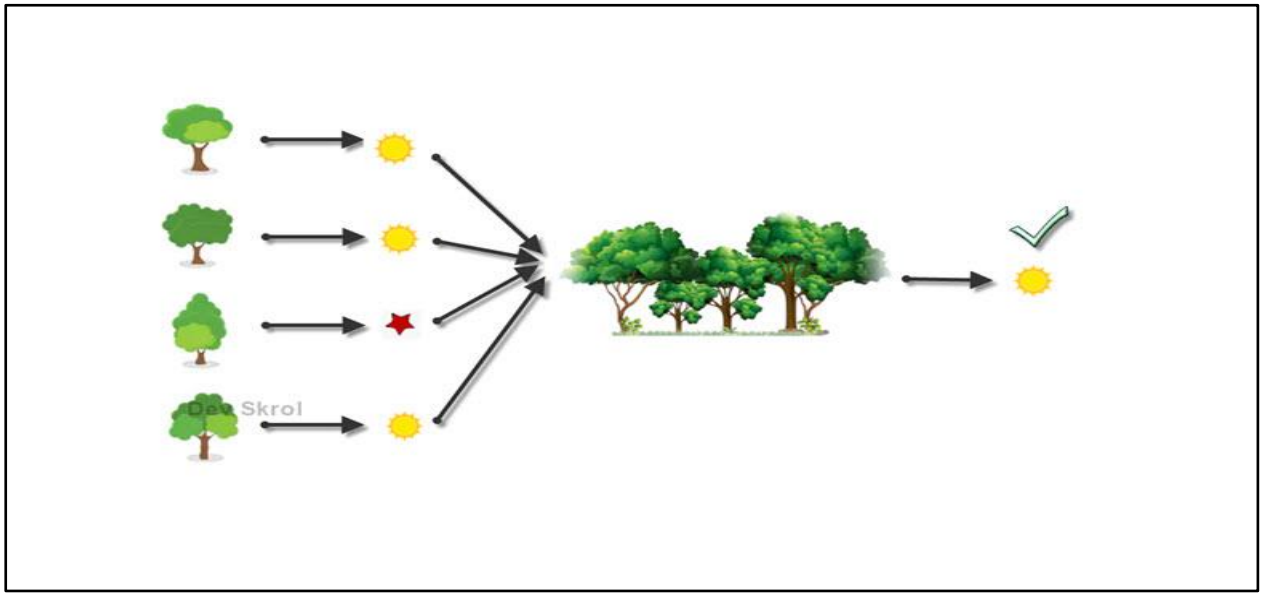
- Mô hình SVM

```
✓ 469m 39.3s
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best Parameters: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
```

4. Chọn mô hình huấn luyện:

4.1. Random Forest

- Random Forest là một thuật toán học máy thuộc nhóm các phương pháp học có giám sát, được sử dụng chủ yếu trong các bài toán phân loại và hồi quy. Thuật toán này là một tập hợp các **Decision Trees** (Cây quyết định), và sử dụng một cơ chế gọi là **Bagging** (Bootstrap Aggregating) để cải thiện độ chính xác và giảm thiểu overfitting so với các mô hình cây quyết định đơn lẻ.
- Bagging tạo ra nhiều tập con của dữ liệu huấn luyện bằng cách lấy mẫu ngẫu nhiên với sự thay thế (bootstrap sampling). Mỗi cây trong rừng (tương ứng với một mẫu dữ liệu) sẽ được huấn luyện độc lập. Sau đó, kết quả của tất cả các cây sẽ được kết hợp (thường là theo phương pháp bỏ phiếu đối với phân loại hoặc trung bình đối với hồi quy) để đưa ra dự đoán cuối cùng.
- Quá trình tạo thành mô hình rừng cây gồm 3 bước:
    - Lấy mẫu tái lập (Bootstrapping) một cách ngẫu nhiên từ tập huấn luyện để tạo thành một tập dữ liệu con
    - Lựa chọn ngẫu nhiên d biến và xây dựng mô hình cây quyết định dựa trên những biến này và tập dữ liệu con ở bước 1. Bước 1 và bước 2 sẽ lặp lại nhiều lần để tạo nên rừng cây
    - Thực hiện bầu cử hoặc lấy trung bình giữa các cây quyết định để đưa ra dự báo



- Mô hình Random Forest được xây dựng như sau (Đã thay các tham số từ GridSearchCV):

```
def train_model(X_train, y_train):
    model = RandomForestClassifier(n_estimators=150, max_depth=7, min_samples_split=2, min_samples_leaf=1, random_state=42, class_weight='balanced')
    model.fit(X_train, y_train)
    return model
```

  - Các tham số của Random Forest
    - n_estimators: 150 (Số lượng cây quyết định)
    - max_depth: 7 (Độ sâu tối đa của cây)
    - min_samples_leaf: 1 (Số lượng mẫu tối thiểu cần có ở mỗi lá)
    - min_sample_split: 2 (Số lượng mẫu tối thiểu của 1 node để phân nhánh)
    - class_weight: "balance" (Tham số xử lý các vấn đề liên quan đến mất cân bằng lớp)

  - Kết quả:

Báo cáo đánh giá (Valid):

	precision	recall	f1-score	support
I	0.43	0.55	0.48	11
P	0.97	0.88	0.92	740
R	0.49	0.84	0.62	115
S	0.89	0.57	0.70	14
W	0.97	0.94	0.95	322
accuracy			0.89	1202
macro avg	0.75	0.76	0.74	1202
weighted avg	0.92	0.89	0.90	1202

Độ chính xác tổng thể: 0.8868552412645591

Báo cáo đánh giá (Test):

	precision	recall	f1-score	support
I	0.50	0.57	0.53	7
P	0.96	0.88	0.92	593
R	0.53	0.81	0.64	110
S	0.80	0.33	0.47	12
W	0.95	0.94	0.94	220
accuracy			0.88	942
macro avg	0.75	0.71	0.70	942
weighted avg	0.90	0.88	0.88	942

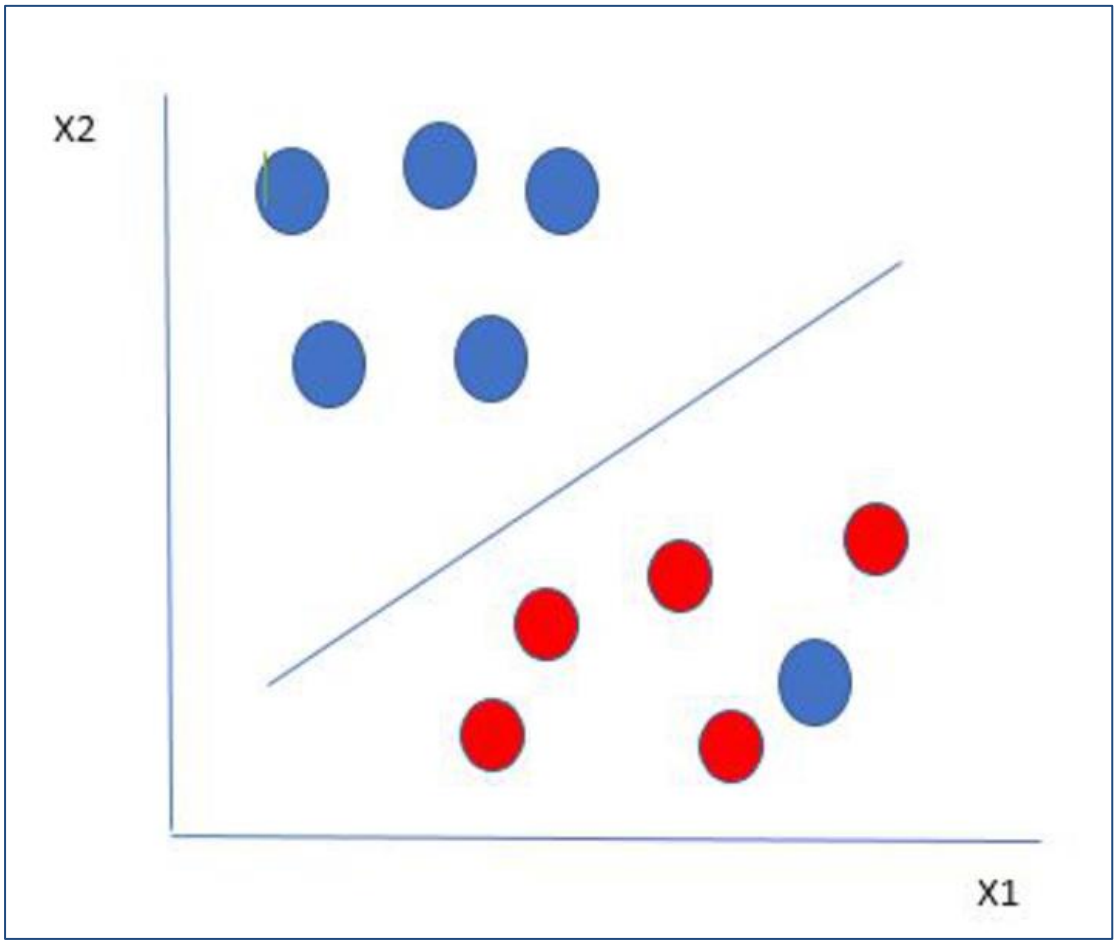
Độ chính xác tổng thể: 0.8779193205944799

4.2. SVM

Khái niệm:

- Support Vector Machine (SVM) là một thuật toán học máy có giám sát mạnh mẽ, được sử dụng rộng rãi cho cả phân loại tuyến tính và phi tuyến, cũng như các tác vụ hồi quy và phát hiện ngoại lệ.

- SVM hoạt động bằng cách tìm siêu phẳng có biên độ lớn nhất phân tách tốt nhất các điểm dữ liệu của các lớp khác nhau. Nó sử dụng các vectơ hỗ trợ, là các điểm dữ liệu gần siêu phẳng nhất, để xác định ranh giới này.



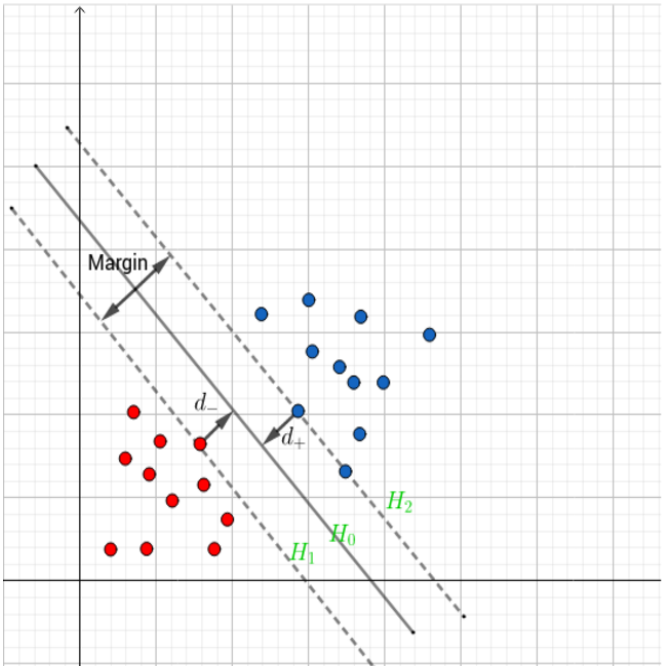
Cách thực thi:

- Siêu phẳng phân cách được biểu diễn bằng hàm số $< W.X >= b$ (W và X là các vector) hay $W^T = b$ (W^T là ma trận chuyển vị).

Cách chọn siêu phẳng tối ưu:

- Giả sử chúng ta phải phân loại tập dữ liệu các lớp dương (màu xanh) nhãn là 1 và các dữ liệu lớp âm (màu đỏ) nhãn là -1 (tập dữ liệu có thể phân tách tuyến tính

- Siêu phẳng phân tách hai lớp dữ liệu H_0 thỏa mãn $<W.X>+b=0$. Siêu phẳng này tạo ra hai nửa không gian (half space) dữ liệu: không gian các dữ liệu lớp âm X_i thỏa mãn $<W.X_i>+b\leq -1$ và không gian dữ liệu lớp dương X_j thỏa mãn $<W.X_j>+b\geq 1$



- Tiếp theo ta chọn hai siêu phẳng lề H_1 đi qua điểm thuộc lớp âm và H_2 đi qua điểm thuộc lớp dương đều song song với H_0:
  - H_1: $<W.X>+b=-1$
  - H_2: $<W.X>+b=1$
- Khoảng cách từ H_1 đến H_0 là d_-
- Khoảng cách từ H_2 đến H_0 là d_+
- $m=d_-+d_+$ được gọi là mức lề
- Siêu phẳng tối ưu mà chúng ta cần chọn là siêu phẳng phân tách có lề lớn nhất. Lý thuyết học máy đã chỉ ra rằng một siêu phẳng như vậy sẽ cực tiểu hóa giới hạn lỗi mắc phải.

Xây dựng mô hình SVM:

- Sử dụng lớp SVC từ module sklearn.svm (Đã thay các tham số tối ưu tìm được từ hàm GridSearchCV)

```
# --- 5. Huấn luyện mô hình ---
def train_svm(X_train, y_train):
    svm = SVC(kernel='rbf', probability=True, gamma='scale', C=100)
    svm.fit(X_train, y_train)
    return svm
```

- **Accuracy score**:

```
✓ 469m 39.3s

Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best Parameters: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
              precision    recall  f1-score   support

           P       0.98      0.99      0.99       740
           R       0.97      0.92      0.95       115
           S       0.82      0.64      0.72        14
           W       0.98      0.98      0.98       322
           I       0.67      0.55      0.60        11

    accuracy                           0.98      1202
   macro avg       0.88      0.82      0.85      1202
weighted avg       0.97      0.98      0.98      1202

              precision    recall  f1-score   support

           P       0.98      0.99      0.98       593
           R       0.94      0.88      0.91       110
           S       0.67      0.33      0.44        12
           W       0.97      0.98      0.98       220
           I       0.50      0.57      0.53         7

    accuracy                           0.96       942
   macro avg       0.81      0.75      0.77       942
weighted avg       0.96      0.96      0.96       942
```

5. Kết quả thực nghiệm:

5.1. Các phương pháp đánh giá hiệu suất:

- Thang đo đánh giá được sử dụng trong đề tài là các độ đo như Accuracy, Precision, Recall, F1-score và Confusion Matrix.

Confusion Matrix: là một bảng dữ liệu được sử dụng để đánh giá hiệu suất của một mô hình phân loại. Nó bao gồm các ô sau:

- True Positives (TP): Số lượng các điểm dữ liệu được dự đoán đúng là positive.
- False Positives (FP): Số lượng các điểm dữ liệu được dự đoán sai là positive.
- True Negatives (TN): Số lượng các điểm dữ liệu được dự đoán đúng là negative.
- False Negatives (FN): Số lượng các điểm dữ liệu được dự đoán sai là negative.

Recall (Còn được gọi là sensitivity hoặc true positive rate): là tỷ lệ giữa số lượng true positives và tổng số positive examples trong dữ liệu.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision: Là tỷ lệ giữa số lượng true positives và tổng số positive predictions (bao gồm cả true positives và false positives).

$$\text{Precision} = \frac{TP}{TP + FP}$$

F1-score: Là một số đo tổng hợp của recall và precision, được tính bằng trung bình điều hòa của hai giá trị này. Nó là một chỉ số hữu ích khi cần cân nhắc cả recall và precision. F1-score được tính bằng công thức:

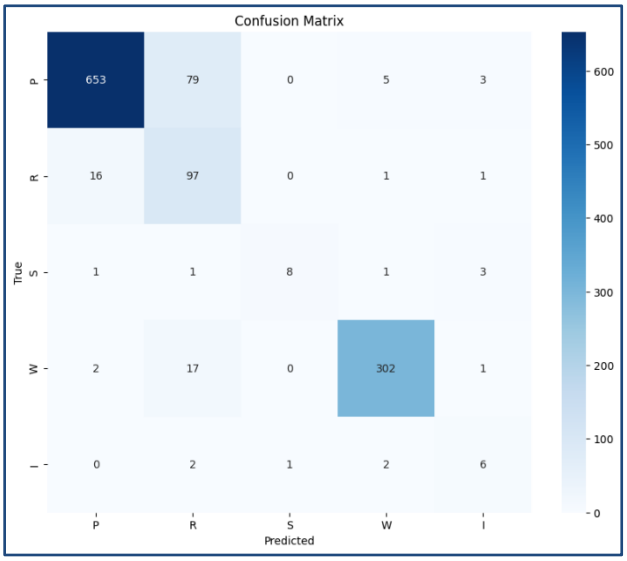
$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy: Accuracy là tỷ lệ giữa số lượng dự đoán đúng và tổng số dự đoán, bao gồm cả positive và negative predictions. Được tính bằng công thức:

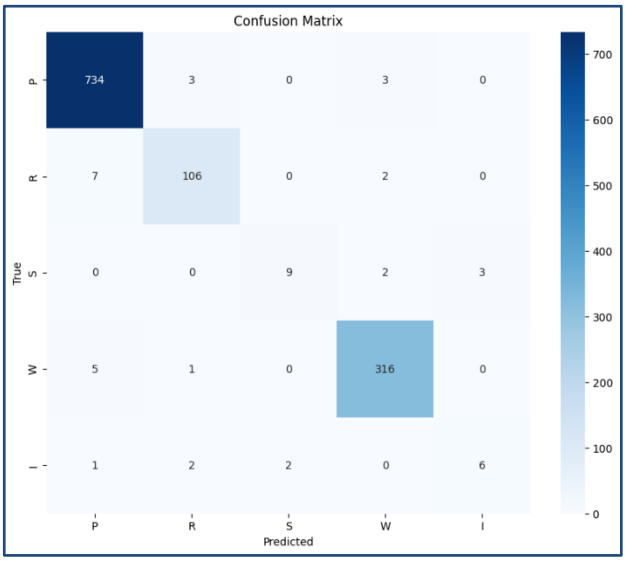
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

5.2. Confusion Matrix cho các thuật toán

Tập Valid:

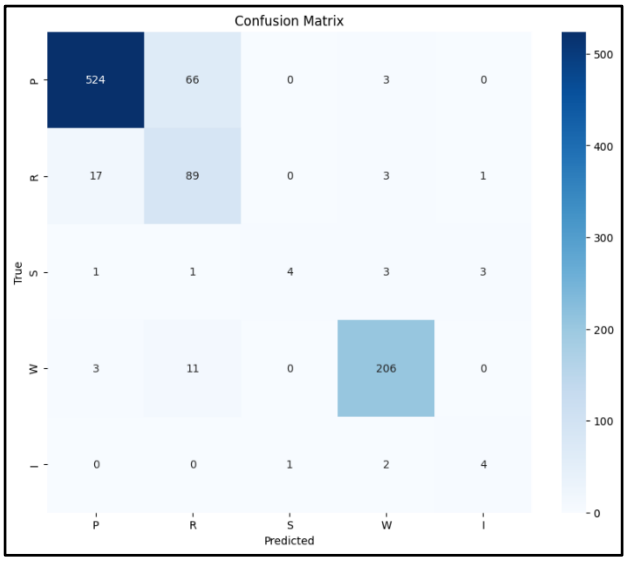


Random Forest

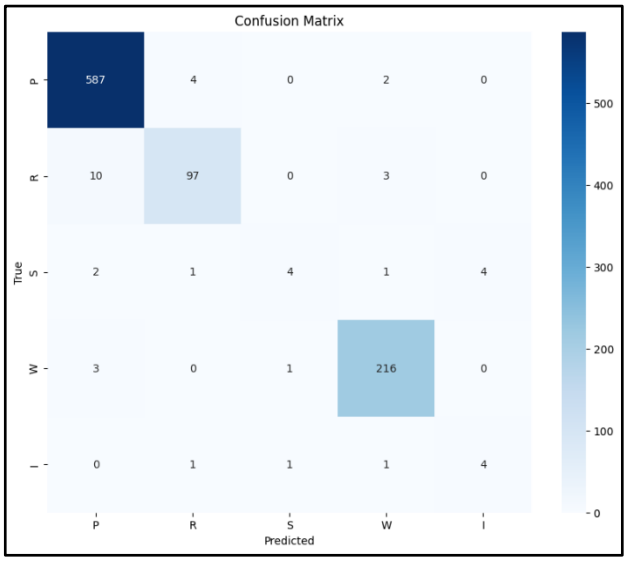


SVM

Tập Test:

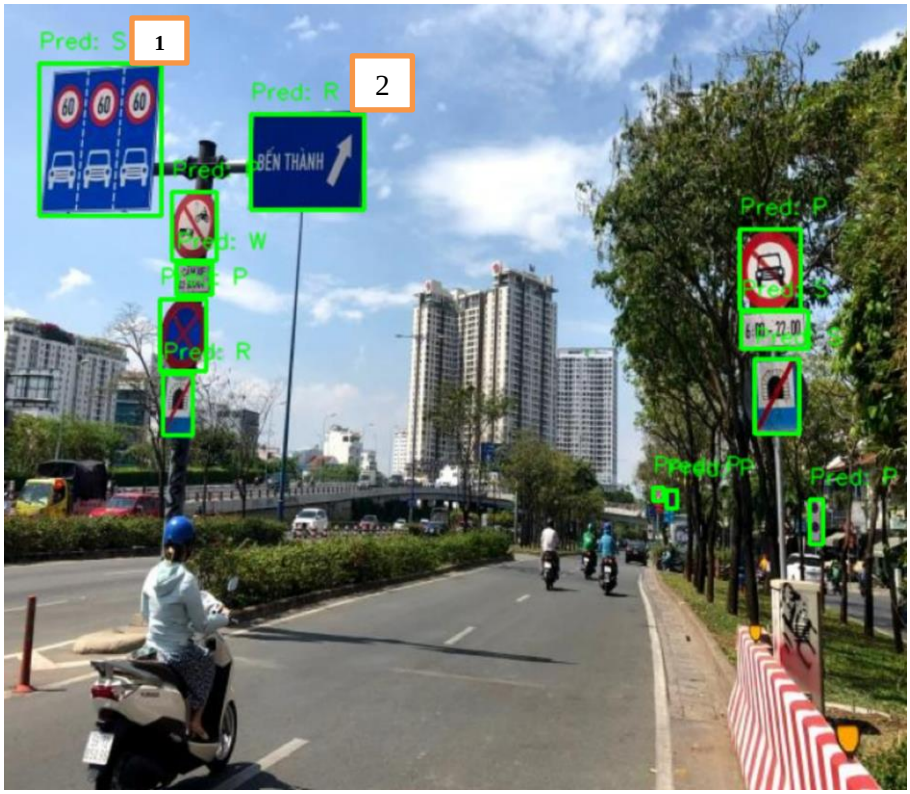


Random Forest



SVM

- Kết quả của ma trận nhầm lẫn cho ta thấy 2 mô hình cho ra Accuracy khá đồng đều. Tuy nhiên, mô hình SVM cho thấy sự nhỉnh hơn đôi chút. Nên sử dụng mô hình SVM để dự đoán biển báo hiện lệnh (R) vì mô hình Random Forest thường nhầm lẫn biển R so với biển báo P.



Nhầm lẫn biển P (1) là biển R và biển I (2) là biển R

Nhầm lẫn biển W (3) là biển P

5.3. Tổng kết

Mô hình	Validation Accuracy	Test Accuracy
Random Forest	0.89	0.88
Super Vector Machine	0.98	0.96

- Kết quả huấn luyện trả về từ hai mô hình máy học được áp dụng trên qua hai lần chạy kiểm thử trên valid và test nhìn chung là khá tốt. SVM có độ chính xác tổng thể và độ ổn định của các thông số có phần nhỉnh hơn RF một chút.
- Hai phương pháp đều thể hiện hiệu suất đáng khích lệ dành cho bài toán phân loại biển báo giao thông và còn có thể hứa hẹn cải thiện hiệu suất hơn nhờ vào bước phân tích đặc trưng và xử lý ảnh tốt hơn (như thêm bước trích xuất đặc trưng màu sắc biển báo).
- Hướng phát triển phương pháp là tích hợp thêm bước nhận diện được biển báo ở vị trí cụ thể trong ảnh, để dữ liệu đầu vào không còn phụ thuộc vào thông tin bbox sẵn có.
- Hướng phát triển về dữ liệu là tìm kiếm thêm dữ liệu cho nhãn ít ảnh để tăng tính đa dạng, hoặc áp dụng thêm các phương pháp tăng cường dữ liệu khác với phương pháp đã áp dụng trong đề tài.

6. Tài liệu tham khảo

Tiếng Việt

- TS. Mai Tiến Dũng, *Slides bài giảng môn Nhập môn Thị giác máy tính*, Trường ĐH Công nghệ thông tin – ĐHQGHCM.
- phamdinhkhanh.github.io, *Thuật toán HOG (Histrogram of oriented gradient)*, Phạm Đình Khánh, https://phamdinhkhanh.github.io/2019/11/22/HOG.html.
- phamdinhkhanh.github.io, Giới thiệu về mô hình rừng cây (Random Forest), Phạm Đình Khánh,
  9. Giới thiệu về mô hình rừng cây (Random Forest) — Deep AI KhanhBlog
- SVM – Machine Learning cơ bản: Machine Learning cơ bản

Tiếng Anh

- Traffic sign detection and recognition based on random forests - ScienceDirect
- Support Vector Machine (SVM) Algorithm - GeeksforGeeks
- Adaptive_Thresholding OpenCV: Image Thresholding
- Yao, Chang, et al. "Traffic sign recognition using HOG-SVM and grid search." 2014 12th International Conference on Signal Processing (ICSP). IEEE, 2014.

Chat Bot

- Chat GPT

Dữ liệu và Benchmark: Street Traffic Signs in VietNam - Coco trên Kaggle kết hợp thêm 110 ảnh biển báo chỉ dẫn (I) và 110 ảnh biển báo phụ (S).

- Đây là bộ dữ liệu quan trọng và là chuẩn mực (benchmark) cho bài toán nhận diện biển báo giao thông. Bộ dữ liệu này chứa các hình ảnh biển báo giao thông đã được gán nhãn, rất phù hợp để huấn luyện và đánh giá các mô hình nhận diện. Street Traffic Signs in Vietnam-Coco