

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



BÁO CÁO CUỐI KỲ
NHẬP MÔN THI GIÁC MÁY TÍNH - CS231

ĐỀ TÀI: NHẬN DẠNG BIỂN SỐ XE HƠI
CAR LICENSE PLATE RECOGNITION

Giảng viên hướng dẫn: TS. Mai Tiến Dũng

Nhóm sinh viên thực hiện:

<i>Họ và tên</i>	<i>MSSV</i>
Nguyễn Tân Khang	21520281
Hồ Yến Nhi	21520380
Huỳnh Võ Ngọc Thanh	21520449

TP.HCM, tháng 06 năm 2023

Mục lục

1 CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI	3
1.1 Giới thiệu bài toán	3
1.2 Lý do chọn đề tài	3
1.3 Quy trình giải quyết bài toán	4
1.3.1 Phát hiện biển số xe	4
1.3.2 Nhận diện ký tự	4
2 CHƯƠNG 2. TÌM HIỂU TỔNG QUAN	5
2.1 Mạng YOLO	5
2.2 Mạng WPOD-NET	5
2.3 Thuật toán kNN	6
2.4 Thuật toán SVM	7
2.5 Công cụ Tesseract OCR	8
3 CHƯƠNG 3. THỰC NGHIỆM	9
3.1 Thông tin dataset	9
3.1.1 YOLOv8	9
3.1.2 SVM và kNN	10
3.2 Train mô hình với dataset	10
3.2.1 Train YOLOv8	10
3.2.2 Train SVM và kNN	11
3.3 Phát hiện biển số xe với YOLO và WPOD-NET	12
3.3.1 YOLOv8	12
3.3.2 WPOD-NET	12
3.4 Xử lý biển số xe trước khi nhận diện ký tự	13
3.4.1 Xử lý ảnh biển số	13
3.4.2 Tách từng ký tự	14
3.5 Nhận diện biển số xe với kNN, SVM và Tesseract OCR	15
3.5.1 SVM và kNN	15
3.5.2 Tesseract OCR	15
4 CHƯƠNG 4. ĐÁNH GIÁ KẾT QUẢ THỰC NGHIỆM	16
4.1 Độ đo và cách đánh giá	16

4.2	Kết quả	16
4.2.1	Kết quả YOLO + Tesseract OCR	16
4.2.2	Kết quả YOLO + SVM	19
4.2.3	Kết quả YOLO + kNN	21
4.2.4	Kết quả WPOD + Tesseract OCR	23
4.2.5	Kết quả WPOD + SVM	25
4.2.6	Kết quả WPOD + kNN	26
4.3	Dánh giá mô hình với trường hợp thực tế	28
4.3.1	YOLO + Tesseract OCR	28
4.3.2	YOLO + SVM	29
4.3.3	YOLO + kNN	30
4.3.4	WPOD + Tesseract OCR	31
4.3.5	WPOD + SVM	32
4.3.6	WPOD + kNN	33
5	CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	34
5.1	Kết luận	34
5.2	Hướng phát triển	34
6	TÀI LIỆU THAM KHẢO	35

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Giới thiệu bài toán

Bài toán nhận dạng biển số xe là một trong những bài toán quan trọng trong lĩnh vực trí tuệ nhân tạo và xử lý ảnh. Với sự phát triển nhanh chóng của công nghệ, việc nhận diện và phân tích thông tin trên biển số xe đã trở nên cần thiết và hữu ích trong nhiều ứng dụng thực tế.

Bài toán nhận dạng biển số xe có thể được định nghĩa là quá trình tự động xác định và trích xuất các ký tự trên biển số xe từ hình ảnh hoặc video chứa thông tin này. Điều này đòi hỏi một hệ thống có khả năng nhận diện và phân tích các đặc trưng như ký tự, màu sắc và hình dạng để tìm ra vị trí và nội dung của biển số xe.

Input của bài toán là hình ảnh có chứa biển số xe ô tô dạng dài (1 dòng). Output là các ký tự trong biển số xe.



Hình minh họa

1.2 Lý do chọn đề tài

Nhóm chúng em chọn đề tài này là vì việc nhận dạng biển số xe có nhiều ứng dụng thực tiễn, bao gồm giám sát giao thông, quản lý bãi đỗ xe, tự động thu phí, an ninh và giám sát. Hệ thống nhận diện biển số xe có thể được tích hợp vào các thiết bị di động, camera giám sát, hệ thống đèn giao thông thông minh và nhiều ứng dụng khác, giúp tăng cường hiệu quả và sự tự động hóa trong quản lý giao thông và an ninh.

1.3 Quy trình giải quyết bài toán

Để giải quyết bài toán nhận dạng biển số xe, nhóm chúng em chia thành 2 bước chính:

1.3.1 Phát hiện biển số xe

Đây là bài toán phát hiện đối tượng vật thể có trong bức ảnh. Bài toán phát hiện đối tượng (object detection) nhằm xác định vị trí và phân loại các đối tượng trong hình ảnh. Đối tượng có thể là bất kỳ thứ gì từ con người, xe hơi, động vật, đến các vật thể khác. Các thuật toán phát hiện đối tượng thường sử dụng các kỹ thuật xử lý hình ảnh và học máy để tìm kiếm và phân loại các vùng quan tâm trong hình ảnh.

Ở đây đối tượng cụ thể là biển số xe hơi 1 dòng. Để giải quyết bài toán này, có nhiều phương pháp và thuật toán được sử dụng, trong đó những phương pháp dựa trên mạng nơ-ron tích chập (CNN) như Faster R-CNN, YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) là phổ biến nhất. Sau bước này thì ta sẽ thu được tấm hình chỉ chứa biển số xe và trở thành input của bước tiếp theo.

1.3.2 Nhận diện ký tự

Bài toán nhận diện ký tự (character recognition) là quá trình nhận dạng và phân loại các ký tự trong hình ảnh hoặc video. Bài toán này thường được áp dụng trong nhiều lĩnh vực, bao gồm nhận dạng biển số xe, OCR (Optical Character Recognition) để chuyển đổi tài liệu giấy thành văn bản số, nhận diện chữ viết tay, và nhiều ứng dụng khác liên quan đến xử lý ngôn ngữ tự nhiên. Các phương pháp phân loại phổ biến bao gồm sử dụng các mô hình học máy như SVM (Support Vector Machine), kNN (k-Nearest Neighbors), Random Forest, hay sử dụng các mạng nơ-ron như MLP (Multilayer Perceptron) hoặc CNN.

CHƯƠNG 2. TÌM HIỂU TỔNG QUAN

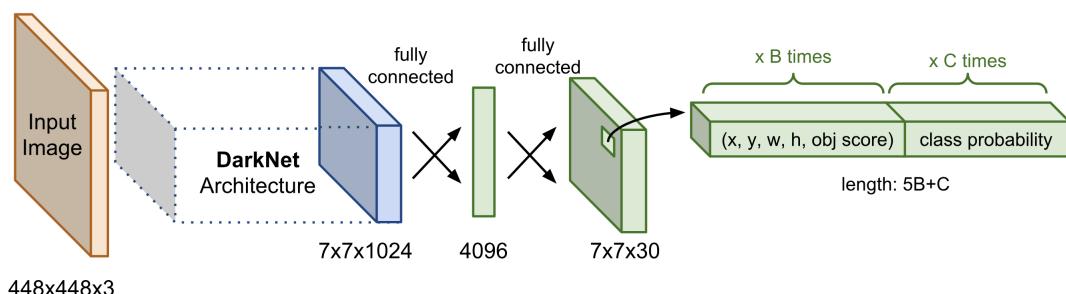
2.1 Mạng YOLO

YOLO (You Only Look Once) là một phương pháp nhận dạng và phân loại đối tượng trong hình ảnh được đề xuất bởi Joseph Redmon và nhóm nghiên cứu của ông. Ban đầu, thuật toán YOLO đã được giới thiệu trong bài báo "You Only Look Once: Unified, Real-Time Object Detection" tại hội nghị "Conference on Computer Vision and Pattern Recognition (CVPR) 2016" vào năm 2016. YOLO có nhiều phiên bản cải tiến từ YOLOv1 cho đến hiện tại là YOLOv8 và YOLO-NAS.

YOLO là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn fully-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

Thành phần Darknet Architecture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước $7 \times 7 \times 1024$ sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

Sau quá trình tìm hiểu, nhóm em quyết định chọn mô hình YOLOv8.



Sơ đồ kiến trúc mạng YOLO

Tham khảo:

<https://arxiv.org/pdf/2304.00501.pdf>

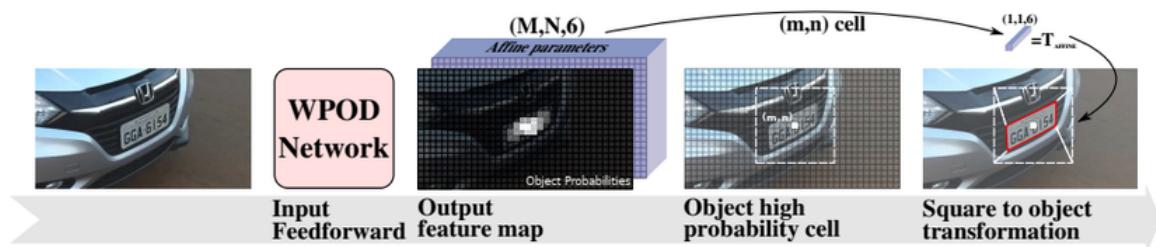
<https://github.com/ultralytics/ultralytics>

2.2 Mạng WPOD-NET

WPOD được đề xuất bởi Sergio Montazzolli Silva và Claudio Rosito Jung

trong bài báo "License Plate Detection and Recognition in Unconstrained Scenarios" năm 2018. Bài báo này được trình bày tại "European Conference on Computer Vision (ECCV)" năm 2018.

WPOD - Wraped Planer Object Detection Network được thiết kế dựa trên ý tưởng của YOLO, SSD và STN (Spatial Transformer Network). Trong khi các mạng YOLO và SSD chỉ trả về một hình chữ nhật bao quanh biển số xe mà không quan tâm đến không gian xung quanh biển số xe là như thế nào thì WPOD có thể trả về một vùng tứ giác bao quanh biển số xe và đưa biển số về hướng nhìn chính diện.



Hoạt động của mạng WPOD

Hình bên thể hiện cách thức hoạt động của mạng WPOD. Ta có thể thấy từ 1 ảnh đầu vào thông qua quá trình lan truyền tiến qua mạng ta thu được output features map gồm 8 channels trong đó 2 channels đầu tiên là xác suất có/không có biển số xe và 6 channels còn lại là những thông số để tính toán ma trận transform (là ma trận được sử dụng để chuyển đổi góc nhìn của biển số xe).

Để trích xuất ra được vùng biển số xe thì nhóm tác giả đã xét một hình vuông với kích thước cố định (phần hình vuông viền trắng trên hình) xung quanh từng cell trên output features map. Nếu xác suất có đối tượng của cell đó lớn hơn ngưỡng quy định thì những giá trị của 6 channels còn lại của cell đó sẽ được sử dụng để tính toán ma trận transform từ vùng hình vuông về vùng biển số xe (đa giác viền đỏ trên hình). Và ta cũng sẽ sử dụng ma trận này để đưa biển số xe về hướng nhìn chính diện.

Tham khảo:

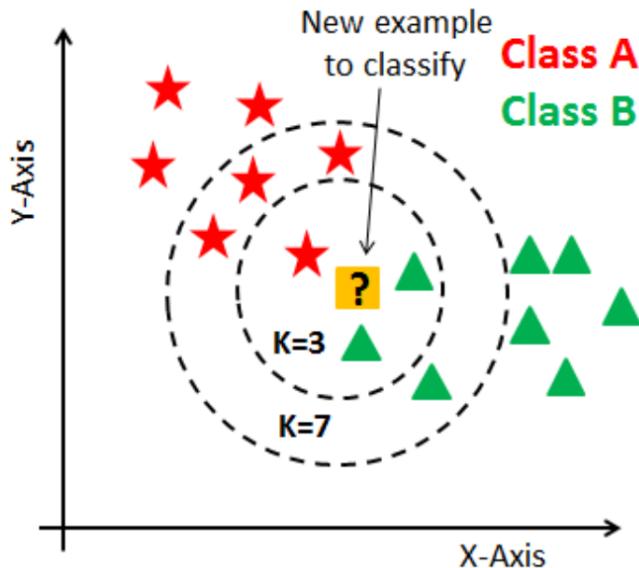
https://openaccess.thecvf.com/content_ECCV_2018/papers/Sergio_Silva_License_Plate_Detection_ECCV_2018_paper.pdf
<http://sergiomsilva.com/pubs/alpr-unconstrained/>

2.3 Thuật toán kNN

Thuật toán k-Nearest Neighbor (kNN) là một phương pháp học có giám sát

do Evelyn Fix và Joseph Hodges phát triển lần đầu vào năm 1951, và sau đó được mở rộng bởi Thomas Cover. Nó cũng được sử dụng cho bài toán phân loại và hồi quy.

Nhiệm vụ của phương pháp k-Nearest Neighbours (kNN) là phân loại các ký tự trong biển số xe vào các lớp ký tự đã biết trước.

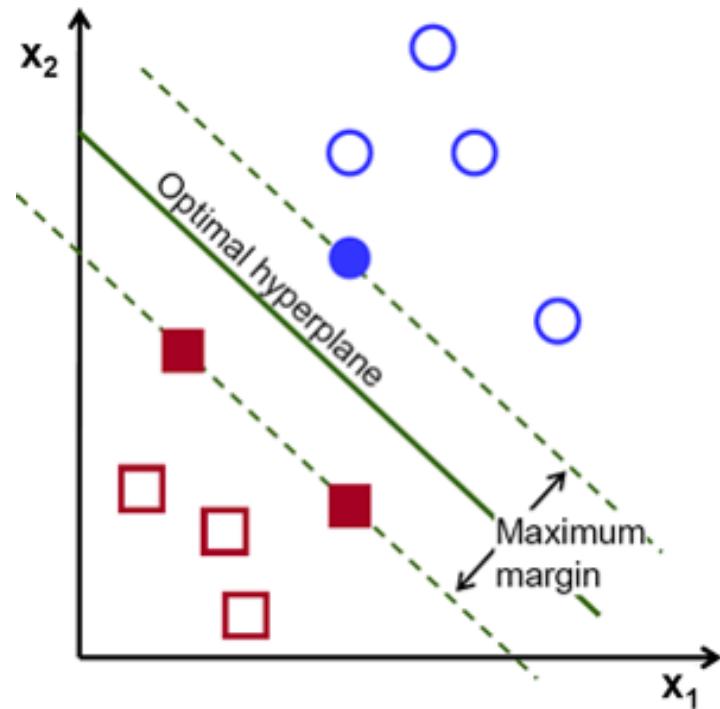


Trong giai đoạn phân loại, k là hằng số do người dùng xác định. Nhiệm vụ của thuật toán kNN là phân loại một đối tượng theo đa số phiếu bầu của các ‘hàng xóm’ xung quanh nó, tức là đối tượng sẽ được gán vào lớp phổ biến nhất trong số k ‘hàng xóm’ gần nhất với nó. Nếu $k = 1$, thì đối tượng được gán đơn giản vào lớp gần nhất.

Các ‘hàng xóm’ được lấy từ một tập hợp các đối tượng đã biết lớp. Đây có thể coi là tập huấn luyện cho thuật toán, mặc dù không yêu cầu bước huấn luyện rõ ràng. Một điểm đặc biệt của thuật toán kNN là nó nhạy cảm với cấu trúc cục bộ của dữ liệu. Các ví dụ huấn luyện là các vector trong một không gian đặc trưng nhiều chiều, mỗi vector có một nhãn lớp. Giai đoạn đào tạo của thuật toán chỉ bao gồm lưu trữ các vector đặc trưng và nhãn lớp của các mẫu đào tạo.

2.4 Thuật toán SVM

Support Vector Machine (SVM) được Vapnik giới thiệu vào năm 1979. SVM là một thuật toán học có giám sát, có thể sử dụng cho cả việc phân loại và hồi quy.



Mục tiêu của SVM là xây dựng một siêu mặt phẳng trong không gian đa chiều (N chiều), hiệu quả trong phân chia các điểm dữ liệu thuộc các lớp tương ứng khác nhau. Siêu phẳng này được tối ưu dựa trên một số điểm dữ liệu gọi là vector hỗ trợ (support vectors), nằm gần nhất với ranh giới giữa các lớp. SVM cố gắng tối đa hóa khoảng cách giữa siêu phẳng và các vector hỗ trợ, được gọi là biên (margin) nhằm đảm bảo tính tổng quát và khả năng phân loại tốt trên dữ liệu mới.

2.5 Công cụ Tesseract OCR

Tesseract OCR là một công cụ nhận dạng ký tự quang học (OCR) cho các hệ điều hành khác nhau. Ban đầu được phát triển bởi Hewlett-Packard là một phần mềm độc quyền vào những năm 1980, nó được phát hành dưới dạng mã nguồn mở vào năm 2005 và được Google tài trợ phát triển từ năm 2006 trở đi.

Tesseract OCR sử dụng các thuật toán tiên tiến để xác định và trích xuất ký tự từ hình ảnh. Nó có khả năng nhận dạng các ngôn ngữ khác nhau và hỗ trợ đa ngôn ngữ, bao gồm cả các ngôn ngữ không phải là chữ Latinh.

Sự kết hợp của khả năng nhận dạng đa ngôn ngữ, khả năng xử lý ảnh đa dạng và linh hoạt của nó làm cho Tesseract OCR trở thành một công cụ rất hữu ích cho việc tự động hóa xử lý tài liệu và trích xuất thông tin từ hình ảnh.

CHƯƠNG 3. THỰC NGHIỆM

Trong phần thực nghiệm của nhóm em thì tụi em sẽ train YOLOv8, SVM, kNN. Còn WPOD-NET nhóm chúng em sử dụng pretrained model, Tesseract OCR thì chỉ sử dụng phần mềm có sẵn.

3.1 Thông tin dataset

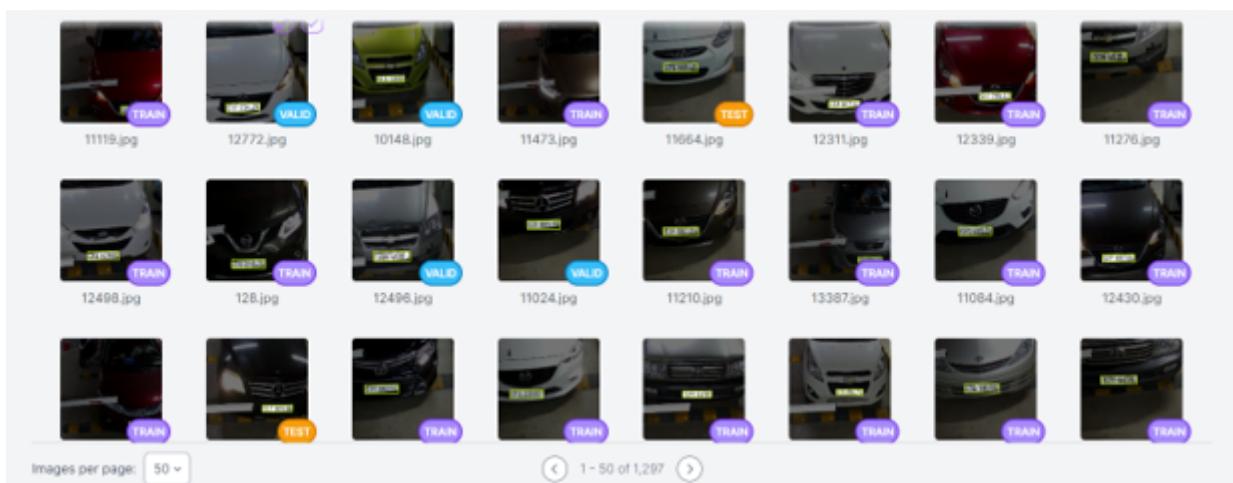
3.1.1 YOLOv8

Dataset dùng cho YOLOv8 gồm 1297 ảnh chứa nhiều loại biển số xe dạng dài, được chụp ở nhiều góc độ khác nhau. Để thu thập bộ dữ liệu, nhóm em đã search Google lấy từ nhiều nguồn khác nhau và sau đó tiến hành lọc để loại bỏ các ảnh nhiễu và ảnh trùng lặp. Nhóm em đã sử dụng công cụ label Roboflow (<https://roboflow.com/>) để gán nhãn cho bộ dataset.

Link dataset: <https://universe.roboflow.com/uit-ze0fr/cs231-ddyhg>

Dataset đã được chia thành các phần train, test và validation như sau:

- Training set: Bộ dữ liệu huấn luyện có 1039 ảnh, chiếm 80% tổng số ảnh trong dataset. Phần này được sử dụng để huấn luyện mô hình YOLOv8.
- Validation set: Bộ dữ liệu validation gồm 129 ảnh, tương ứng với 10% tổng số ảnh trong dataset. Phần này được sử dụng để đánh giá hiệu suất của mô hình YOLOv8 trong quá trình huấn luyện.
- Test set: Bộ dữ liệu kiểm tra có 129 ảnh, tương ứng với 10% tổng số ảnh trong dataset. Phần này được sử dụng để đánh giá độ chính xác và hiệu suất tổng quan của mô hình YOLOv8 sau khi đã huấn luyện.



3.1.2 SVM và kNN

Dataset dùng cho SVM và kNN gồm 1934 ảnh chứa nhiều loại ký tự trong biển số xe dạng dài, gồm có tổng cộng 31 ký tự, mỗi loại ký tự có khoảng từ 55 đến 70 ảnh.

Link digit dataset:

https://www.mediafire.com/file/3l3x7bd7rq9115r/data_digit.zip/file



3.2 Train mô hình với dataset

Nhóm em train trên Google Colab với GPU Tesla T4.

3.2.1 Train YOLOv8

Với YOLOv8, để train với dataset của nhóm em thì import thư viện và gọi hàm có sẵn để train.

```
from ultralytics import YOLO
!yolo task=detect mode=train model=yolov8n.pt
  data={'/content/drive/MyDrive/CS231_Project/CS231/CS231-2'}/data.yaml
  epochs=100 imgsz=640
```

Trong đoạn code trên thì nhóm em đã import YOLO từ thư viện ultralytics. Sau đó chỉnh các tham số cho quá trình train:

- task = detect: YOLO sẽ được dùng cho bài toán object detection, output sẽ gồm các bounding box và label tương ứng.
- mode = train: dùng để train.
- model = yolov8n.pt: sử dụng tham số từ pretrained model.
- data = {link}/data.yaml: chọn thư mục chứa data và file config data.yaml.
- epochs = 100: chọn epoch bằng 100.
- imgsz = 640: ảnh sẽ được resize thành ảnh 640 x 640 khi đưa vào train.
- Ngoài ra thì còn 1 vài thông số mặc định như batch = 16, optimizer = SGD, ...

3.2.2 Train SVM và kNN

Với SVM và kNN nhóm chúng em sử dụng chung bộ dữ liệu và sử dụng 2 model có sẵn trong thư viện OpenCV nên cách đọc và train dữ liệu cũng khá giống nhau, theo từng bước:

- Chia các hình ký tự vào thư mục với tên tương ứng (theo mã ASCII, ví dụ chữ A là 65, ...).
- Tạo 2 mảng lưu dữ liệu ký tự và label của ký tự đó.
- Đọc ảnh ký tự, resize lại thành vector và lưu vào mảng ký tự. Bên cạnh đó, đọc tên thư mục và lưu vào mảng label theo thứ tự tương ứng.

```
import cv2
svm_model = cv2.ml.SVM_create()
svm_model.setType(cv2.ml.SVM_C_SVC)
svm_model.setKernel(cv2.ml.SVM_INTER)
svm_model.setTermCriteria((cv2.TERM_CRITERIA_MAX_ITER, 100, 1e-6))
svm_model.trainAuto(digit_list, cv2.ml.ROW_SAMPLE, label_list)
svm_model.save('svm.xml')
```

- Tạo model SVM từ thư viện OpenCV.

- Đặt loại SVM thành C-Support Vector Classification - C-SVC. Điều này cho phép SVM xây dựng một mô hình phân loại đa lớp.
 - Đặt kernel của SVM thành SVM Inter.
 - Đặt tiêu chí dừng cho quá trình huấn luyện SVM. Ở đây, sử dụng tiêu chí dừng với số lượng tối đa vòng lặp là 100 và sai số tối đa 1e-6.
 - Huấn luyện mô hình SVM sử dụng dữ liệu huấn luyện và nhãn tương ứng, mỗi hàng của ma trận dữ liệu đại diện cho một mẫu.
 - Lưu mô hình SVM đã được huấn luyện vào tệp "svm.xml". Tệp này có thể được sử dụng sau này để tải lại mô hình và thực hiện dự đoán trên dữ liệu mới.
-

```
kNN_model = cv2.ml.KNearest_create()
kNN_model.setDefaultK(1)
kNN_model.train(digit_list, cv2.ml.ROW_SAMPLE, label_list)
kNN_model.save('kNN_1.xml')
```

- Đối với kNN nhóm em chọn k = 1 và làm tương tự như SVM.
- Validation:

SVM	kNN
0.98	0.97

3.3 Phát hiện biển số xe với YOLO và WPOD-NET

3.3.1 YOLOv8

```
model = YOLO('/content/drive/MyDrive/CS231_Project/CS231/runs/detect/
train5/weights/best.pt')
res = model(img)
res_plotted = res[0].plot()
cv2_imshow(res_plotted)
```

Đầu tiên load model từ đường dẫn mà YOLO sau khi train lưu vào, nhóm em sử dụng bộ trọng số (weight) có loss thấp nhất. Sau đó thì truyền ảnh vào hàm `model()`. Ảnh sau khi được detect sẽ lưu vào `res` và dùng `plot()` để truy xuất.

3.3.2 WPOD-NET

```
wpod_net_path = "wpod-net_update1.json"
wpod_net = load_model(wpod_net_path)
predict = model.predict(img)
```

WPOD nhóm chúng em sử dụng pretrained model nên chỉ cần load model từ file .json và sử dụng hàm `load_model` để lấy bộ trọng số từ file .h5 có cùng tên. Sau đó thì gọi hàm `predict` để nhận kết quả trả về là ảnh biển số xe.

3.4 Xử lý biển số xe trước khi nhận diện ký tự

Trước khi nhận diện ký tự thì nhóm em có thực hiện các giai đoạn tiền xử lý để tăng cường thông tin trong việc nhận diện biển số, ở đây cụ thể là nhóm em sẽ làm 2 giai đoạn như sau:

3.4.1 Xử lý ảnh biển số

Đầu tiên, chuyển ảnh biển số thành ảnh xám (Grayscale) bằng `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`, loại bỏ thông tin về màu sắc và chỉ giữ lại thông tin về độ sáng. Quá trình này giúp giảm chi phí tính toán và tập trung vào đặc trưng cần thiết cho việc nhận diện biển số.



Tiếp theo, làm mờ ảnh (Blur) bằng `cv2.medianBlur(gray, 3)` để giảm nhiễu và loại bỏ chi tiết không cần thiết, làm mờ các chi tiết không liên quan và tạo ra một ảnh mượt hơn, dễ dàng để tách biệt các ký tự trên biển số.



Tiếp tục áp dụng ngưỡng động (Thresholding) bằng `cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 31, 2)` để chuyển đổi ảnh thành ảnh nhị phân, trong đó các pixel được gán một trong hai giá trị (trắng hoặc đen) dựa trên độ sáng của chúng từ đó có thể tách biệt các ký tự từ nền biển số.



Cuối cùng, tăng kích thước vùng ký tự (Character Region Resizing) bằng `cv2.dilate(binary, kernel3, iterations = 1)` để điều chỉnh kích thước của vùng ký tự từ đó giúp cải thiện khả năng nhận diện và định vị các ký tự.



3.4.2 Tách từng ký tự

Sau khi có ảnh sau các bước tiền xử lý, tiếp theo là tách từng ký tự trong biển số. Quá trình này nhằm tách riêng các ký tự thành các phần riêng biệt để dễ dàng nhận diện và định vị mỗi ký tự.

Đầu tiên, tìm và xác định các contours bằng `cv2.findContours(thre_mor, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)`. Các contours sẽ tương ứng với các đối tượng có đặc trưng khác biệt với nền. Trong trường hợp này, các contours tương ứng với các ký tự trong biển số.



Tiếp đến, vẽ các bounding box bằng `cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)`: Với mỗi contour tìm được, chúng ta có thể vẽ một hình chữ nhật xung quanh (bounding box) nó. Bước này giúp xác định vùng chứa ký tự.



Lọc bounding box theo 2 điều kiện, 1 là aspect ratio trong đoạn từ 1 đến 3.5 $1 \leq h/w \leq 3.5$, 2 là chiều cao từ 55% biến số trở lên $h/roi.shape[0] \geq 0.55$

Cuối cùng, tách ký tự và đưa vào mô hình nhận dạng (SVM, kNN) để thực hiện quá trình nhận diện và gán nhãn cho từng ký tự.



3.5 Nhận diện biển số xe với kNN, SVM và Tesseract OCR

3.5.1 SVM và kNN

Load model được lưu trong file xml bằng hàm `cv2.ml.SVM_load('svm.xml')` và `cv2.ml.KNearest.load("knn_1.xml")`.

Chuyển từng vùng ký tự sau khi được tách thành vector và đưa vào model bằng hàm `model_svm.predict()` và `model_knn.findNearest()`. Kết quả trả về sẽ là từng ký tự và tiền hành cộng chuỗi.

3.5.2 Tesseract OCR

Vì đây là công cụ có sẵn nên chỉ cần truyền ảnh biển số xe vào hàm `pytesseract.image_to_string(cropped_img, lang="eng", config="--psm 7")`. Kết quả trả về sẽ là 1 chuỗi ký tự.

CHƯƠNG 4. ĐÁNH GIÁ KẾT QUẢ THỰC NGHIỆM

4.1 Độ đo và cách đánh giá

Dộ chính xác (accuracy) giúp đánh giá hiệu quả việc dự đoán của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình càng chuẩn xác.

Dộ chính xác là tỉ lệ giữa số điểm dữ liệu được dự đoán đúng và tổng số điểm dữ liệu.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total data points}}$$

Nhưng vì trong bài toán này, nhóm em nhận thấy có những trường hợp biến số phát hiện ra chỉ sai 1 hoặc 2 ký tự, hay là bị dư bị thiếu ký tự nên nhóm chúng em quyết định tính theo cách so dãy con chung dài nhất của chuỗi dự đoán và chuỗi label. Cuối cùng thì đem đếm đếm chia cho tổng số hình đưa vào dự đoán để tính trung bình.

4.2 Kết quả

YOLO + Tesseract OCR	YOLO + SVM	YOLO + kNN
65.27%	33.01%	33.53%

WPOD + Tesseract OCR	WPOD + SVM	WPOD + kNN
75.17%	60.87%	62.72%

Với mỗi phương pháp, nhóm em có chọn ra 4 tấm ảnh khi test để thể hiện độ chính xác của từng phương pháp.

4.2.1 Kết quả YOLO + Tesseract OCR

Có thể thấy, với những tấm ảnh rõ nét và được chụp chính diện thì phương pháp này nhận dạng ký tự rất tốt (ảnh 1 và 2).



Ảnh test 1 và Biển số sau khi detect

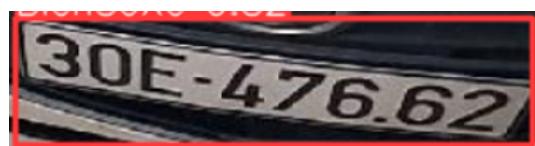


Ảnh test 2 và Biển số sau khi detect

Còn khi chụp những góc nhìn không chính diện thì phương pháp này lại khá khó khăn trong việc nhận dạng được ký tự (ảnh 3 và 4).

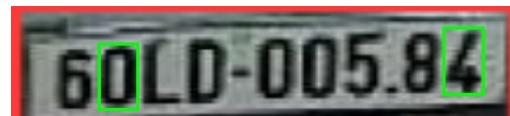
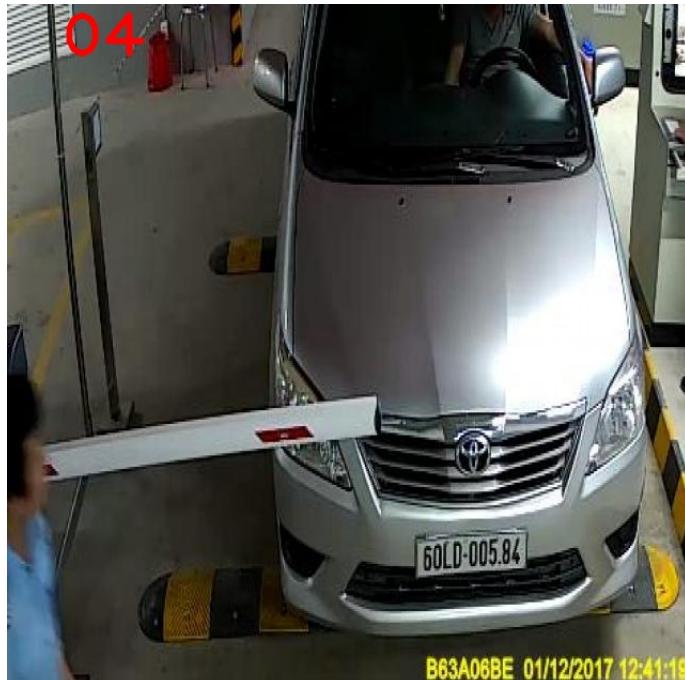


Ảnh test 3 và Biển số sau khi detect



Ảnh test 4 và Biển số sau khi detect

4.2.2 Kết quả YOLO + SVM



Ảnh test 1 và Biển số sau khi detect

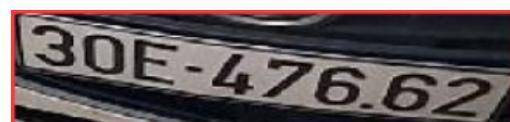


Ảnh test 2 và Biển số sau khi detect



24/10/2020 08:22

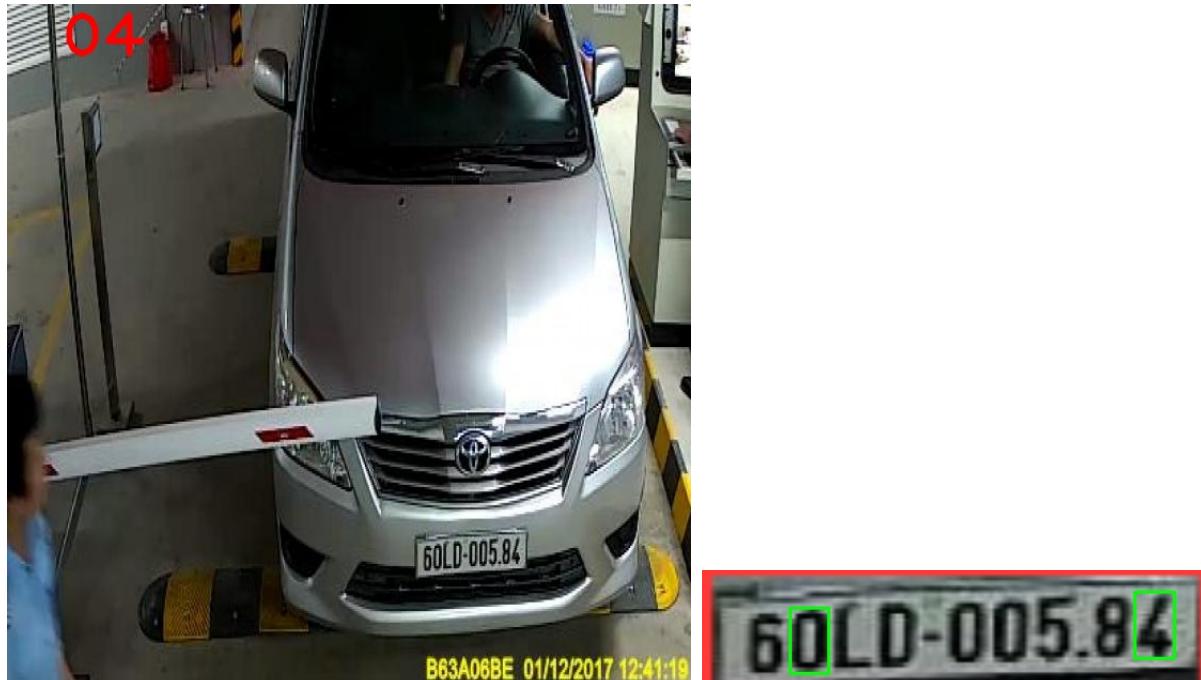
Ảnh test 3 và Biển số sau khi detect



28/10/2020 06:36

Ảnh test 4 và Biển số sau khi detect

4.2.3 Kết quả YOLO + kNN



Ảnh test 1 và Biển số sau khi detect



Ảnh test 2 và Biển số sau khi detect



24/10/2020 08:22

Ảnh test 3 và Biển số sau khi detect



28/10/2020 06:36

Ảnh test 4 và Biển số sau khi detect

4.2.4 Kết quả WPOD + Tesseract OCR



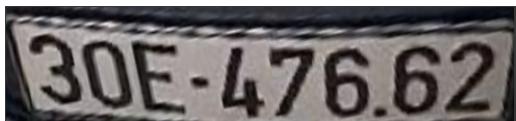
Ảnh test 1 và Biển số sau khi detect



Ảnh test 2 và Biển số sau khi detect



Ảnh test 3 và Biển số sau khi detect

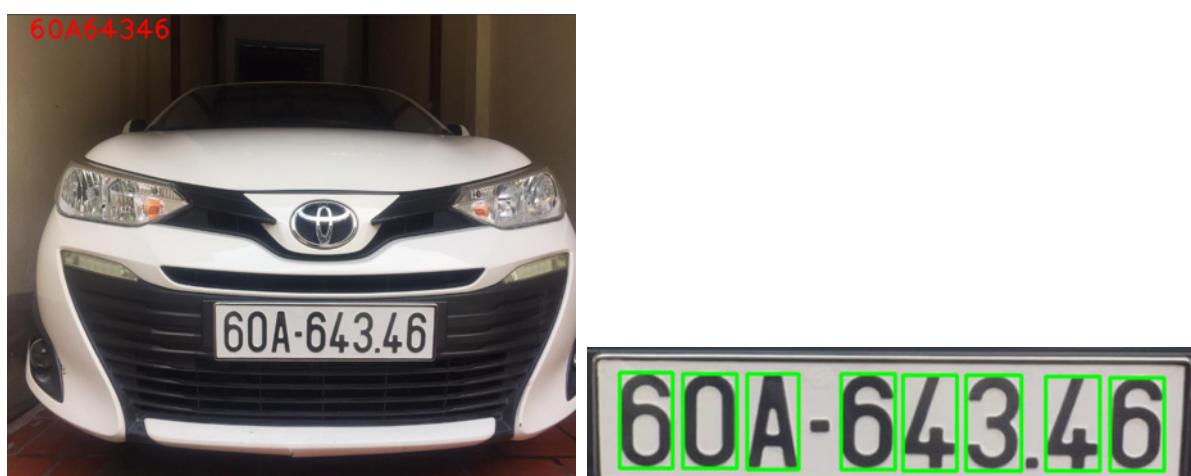


Ảnh test 4 và Biển số sau khi detect

4.2.5 Kết quả WPOD + SVM



Ảnh test 1 và Biển số sau khi detect



Ảnh test 2 và Biển số sau khi detect

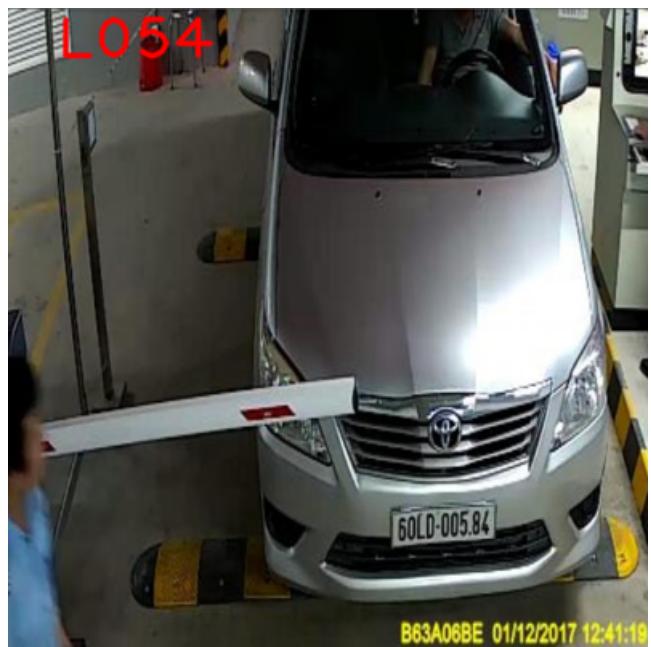


Ảnh test 3 và Biển số sau khi detect



Ảnh test 4 và Biển số sau khi detect

4.2.6 Kết quả WPOD + kNN



Ảnh test 1 và Biển số sau khi detect



Ảnh test 2 và Biển số sau khi detect



Ảnh test 3 và Biển số sau khi detect



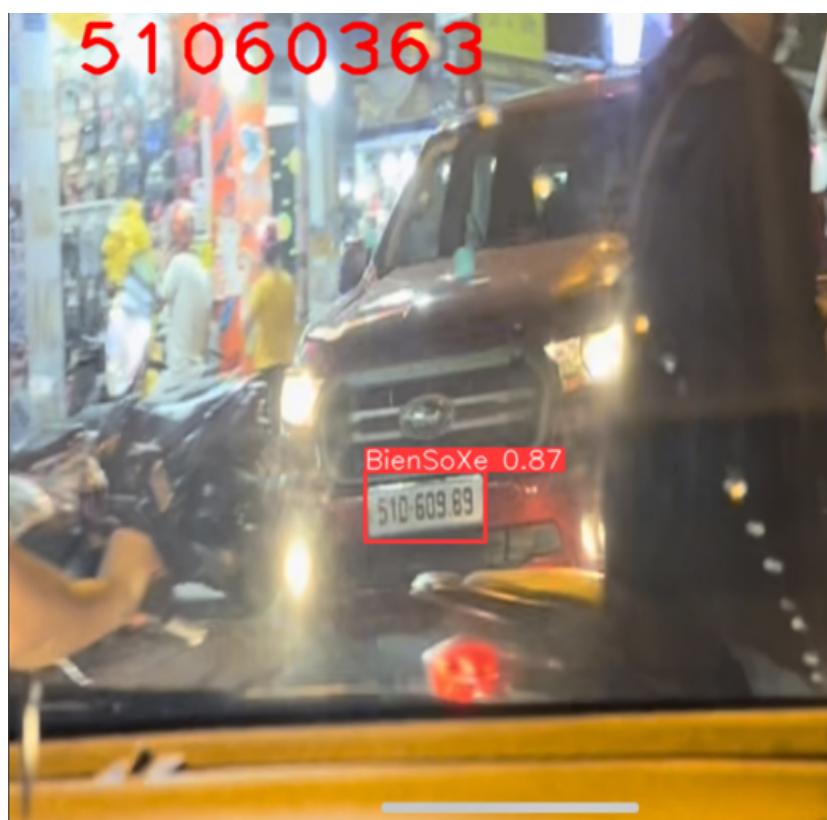
Ảnh test 4 và Biển số sau khi detect

4.3 Đánh giá mô hình với trường hợp thực tế

4.3.1 YOLO + Tesseract OCR



Ảnh 1



Ảnh 2

4.3.2 YOLO + SVM



Ảnh 1



Ảnh 2

4.3.3 YOLO + kNN



Ảnh 1



Ảnh 2

4.3.4 WPOD + Tesseract OCR



Ảnh 1



Ảnh 2

4.3.5 WPOD + SVM



Ảnh 1



Ảnh 2

4.3.6 WPOD + kNN



Ảnh 1



Ảnh 2

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Về phần phát hiện biển số xe, WPOD-NET cho ra kết quả rất tốt vì sử dụng pretrained model, còn YOLO cũng cho kết quả tương đối tốt nhưng trong thực tế các biển số xe bị mờ thì chưa nhận ra được. Lý do là vì lượng data nhóm em thu thập chưa đủ nhiều và đa dạng.

Về phần nhận dạng ký tự, đối với OCR sẽ cho ra kết quả khá tốt nếu như vùng cần đọc có màu đen và nền là màu trắng và vật thể phải khá rõ nét. Còn về SVM và kNN thì còn bị nhầm lẫn giữa ký tự "D" và "0", "A" và "4", đây là do các hình ký tự dùng để train còn ít và không được đa dạng kiểu chữ nên dễ nhầm lẫn. Ngoài ra còn phụ thuộc vào việc xác định được bounding box và cắt ra vùng chứa ký tự, trong khi chúng em test thì có nhiều hình biển số bị dư hay thiếu bounding box, dẫn đến dự đoán sai.

⇒ Dataset còn ít, chưa đa dạng, phụ thuộc nhiều yếu tố như góc chụp, độ sáng. Còn lại thì mô hình vẫn nhận dạng khá ổn.

5.2 Hướng phát triển

- Tìm thêm nhiều nguồn ảnh, làm giàu dataset.
- Train model YOLO với nhiều epoch hơn, tham số phù hợp hơn.
- Tìm cách xác định vùng ký tự tốt hơn.
- Sử dụng mô hình phức tạp hơn cho nhận dạng ký tự, ví dụ như các mạng CNN.
- Xây dựng hệ thống web để cho người dùng dễ sử dụng.

TÀI LIỆU THAM KHẢO

<https://miae.vn/2019/11/30/nhan-dien-bien-so-xe-chuong-4-nhan-dien-bien-so-xe-bang-wpod-va-tesseract-ocr/>

<https://miae.vn/2019/12/05/nhan-dien-bien-so-xe-chuong-5-nhan-dien-bien-so-xe-bang-wpod-va-svm/>

<https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/>

https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html

https://docs.opencv.org/4.x/d5/d26/tutorial_py_knn_understanding.html

<https://docs.ultralytics.com/>