

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

------------------------------------

**BÁO CÁO ĐỒ ÁN 3**

**ĐỀ TÀI:**

**TÌM HIỂU THUẬT TOÁN NHẬN DẠNG BIỂN SỐ XE**

**GVHD:** Ph.D Nguyễn Thiên Bảo

**SVTH:**

17110153 – Võ Trần Gia Huy

17110166 – Tôn Long Hoàng Lãm

TP. HỒ CHÍ MINH – 09/01/2021

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

------------------------------------

**BÁO CÁO ĐỒ ÁN 3**

**ĐỀ TÀI:**

**TÌM HIỂU THUẬT TOÁN NHẬN DẠNG BIỂN SỐ XE**

**GVHD:** Ph.D Nguyễn Thiên Bảo

**SVTH:**

17110153 – Võ Trần Gia Huy

17110166 – Tôn Long Hoàng Lãm

TP. HỒ CHÍ MINH – 09/01/2021

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

----\*\*\*----

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên: Võ Trần Gia Huy MSSV: 17110153

Tôn Long Hoàng Lãm MSSV: 17110166

Ngành: Công nghệ thông tin

Tên đề tài: Tìm hiểu thuật toán nhận dạng biển số xe

Họ và tên Giáo viên hướng dẫn: Ph.D Nguyễn Thiên Bảo

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

.....................................................................................................................................................................................................................................................................................................................................................................................................................................................................

2. Ưu điểm:

...................................................................................................................................................................................................................................................................................................................................................................................................................................................

3. Khuyết điểm:

.............................................................................................................................................................................................................................................................................................................. ....................................................................................................................................

5. Đánh giá loại:

.................................................................................................................................................

.................................................................................................................................................

6.Điểm:……………….(Bằngchữ: .......................................................................................)

Tp*. Hồ Chí Minh, ngày tháng năm 2021*

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên)*

**MỤC LỤC**

[LỜI CẢM ƠN i](#_Toc61270563)

[DANH MỤC HÌNH ẢNH ii](#_Toc61270564)

[PHẦN MỞ ĐẦU 1](#_Toc61270565)

[1.1 Lý do chọn đề tài 1](#_Toc61270566)

[1.2 Mục tiêu nghiên cứu 1](#_Toc61270567)

[1.3 Phạm vi nghiên cứu 1](#_Toc61270568)

[1.4 Đối tượng nghiên cứu 2](#_Toc61270569)

[1.5 Phương pháp nghiên cứu 2](#_Toc61270570)

[PHẦN NỘI DUNG 2](#_Toc61270571)

[2.1 Tìm hiểu chung về OpenCV [1] 2](#_Toc61270572)

[2.1.1 Giới thiệu tổng quan về OpenCV 2](#_Toc61270573)

[2.1.2 Ứng dụng của OpenCv 2](#_Toc61270574)

[2.1.3 Pyramid 3](#_Toc61270575)

[2.1.4 Gaussian Pyramid 3](#_Toc61270576)

[2.1.5 Laplacian Pyramid 4](#_Toc61270577)

[2.2 Thuật toán SVM [2] 4](#_Toc61270578)

[2.2.1 Giới thiệu tổng quan về SVM 4](#_Toc61270579)

[2.2.2 Đặt vấn đề 5](#_Toc61270580)

[2.2.3 SVM tuyến tính 5](#_Toc61270581)

[2.2.4 Ứng dụng của SVM trong thực tế 8](#_Toc61270582)

[2.3 Dataset Car 8](#_Toc61270583)

[2.4 Cài đặt thuật toán 8](#_Toc61270584)

[2.4.1 Các thư viện được sử dụng 8](#_Toc61270585)

[2.4.2 Cài đặt và kiểm thử 10](#_Toc61270586)

[KẾT LUẬN 15](#_Toc61270587)

[TÀI LIỆU THAM KHẢO 16](#_Toc61270588)

# LỜI CẢM ƠN

*TPHCM, 09 tháng 01 năm 2021*

Trong thời gian thực hiện đồ án 3, nhóm xin chân thành cảm ơn thầy Nguyễn Thiên Bảo đã tận tình hỗ trợ cho chúng em về vật chất cũng như tinh thần, để chúng em có thể hoàn thành đồ án một cách tốt nhất

Với mỗi sinh viên chúng em nói chung, việc tích lũy kiến thức qua giáo trình, bài giảng trên lớp rất quan trọng và cần thiết, tuy nhiên, việc áp dụng các kiến thức vào thực tế chưa thực sự nhiều. Cho nên việc thực hiện đồ án 3 là một cơ hội để nhóm có thể áp dụng các kiến thức mà mình đã học vào giải quyết các vấn đề thực tiễn.

Vì kiến thức và thời gian thực hiện còn hạn chế, nên trong quá trình thực hiện không tránh xảy ra sai sót. Nhóm rất mong thầy đóng góp ý kiến để hoàn thiện đồ án tốt hơn.

Chúng em xin chân thành cảm ơn thầy!

# DANH MỤC HÌNH **ẢNH**

[Hình 2.4‑1 - TensorFlow 9](#_Toc61270542)

[Hình 2.4‑2 - Dataset Ký tự 10](#_Toc61270543)

[Hình 2.4‑3 - Train Model SVM 11](#_Toc61270544)

[Hình 2.4‑4 - Train Charset 11](file:///D:\DA3\Report2.4_NOT_DONE.docx#_Toc61270545)

[Hình 2.4‑5 - Hình ảnh gốc 12](#_Toc61270546)

[Hình 2.4‑6 - Ảnh biển số sau threshold 13](#_Toc61270547)

[Hình 2.4‑7 - Các contour tìm được 13](#_Toc61270548)

[Hình 2.4‑8 - Kết quả nhận dạng biển số 14](#_Toc61270549)

# PHẦN MỞ ĐẦU

## Lý do chọn đề tài

* Cùng với sự phát triển khoa học kỹ thuật, nhu cầu đi lại của con người ngày càng tăng, lưu lượng giao thông ngày càng lớn. Với số lượng phương tiện giao thông ngày càng lớn và còn không ngừng tăng thì việc quản lý các phương tiện giao thông gặp rất nhiều khó khăn do đó cần có một hệ thống tự động. Một trong những hệ thống đó là hệ thống nhận dạng biển số xe. Đó là một hệ thống có khả năng “đọc” và “hiểu”các biển số xe một cách tự động.
* Nhận thấy thực tế đó, nhóm quyết định lựa chọn đề tài dự đoán giá nhà dựa trên thuật toán của Machine Learning.

## Mục tiêu nghiên cứu

* Do mục đích chính của nghiên cứu này là tìm hiểu và xây dựng một hệ thống “Nhận dạng biển số xe” từ hình ảnh, phim và các thiết bị ghi hình kỹ thuật số. Nhằm trợ giúp cho công tác phát hiện xe vi phạm giao thông, chống trộm, quản lý, … được dễ dàng và nhanh chóng hơn. Sau đây là những ứng dụng của hệ thống nhận dạng biển số xe nói chung: Thu phí giao thông, Kiểm soát xe tại các đường biên giới, các trạm gác cổng, công tác chống trộm, bãi giữ xe tự động, …
* Như mọi hệ thống tự động khác, hệ thống như vậy cũng sẽ yêu cầu có cả phần cứng và phần mềm. Phần cứng là 1 camera có tác dụng thu nhận hình ảnh, còn phần mềm sẽ xử lý hình ảnh đó. Với sự phát triển của kỹ thuật điện tử, các camera sẽ dễ dàng có khả năng thu nhận được hình ảnh do đó vấn đề đặt ra luôn là vấn đề quan trọng nhất trong hệ thống, quyết định tính hiệu quả của hệ thống xử lý ảnh như vậy sẽ là phần mềm xử lý ảnh. Với vai trò như đã phân tích ở trên nghiên cứu này tập trung vào giải quyết các vấn đề đặt ra khi xử lý ảnh để đưa ra chính xác biển số xe.

## Phạm vi nghiên cứu

* Phương pháp nhận dạng biển số xe
* Thuật toán OpenCV
* Ứng dụng thuật toán vào giải quyết bài toán
* Cách đánh giá kết quả và thuật toán

## Đối tượng nghiên cứu

* Các phương pháp nhận dạng và đọc biển số xe
* Thuật toán OpenCV và Nhận dạng chữ in với Tesseract OCR

## Phương pháp nghiên cứu

* Đặt vấn đề thực tế
* Tìm hướng giải quyết
* Ứng dụng thuật toán Machine Learning

# PHẦN NỘI DUNG

## Tìm hiểu chung về OpenCV [1]

### 2.1.1 Giới thiệu tổng quan về OpenCV

* Project [OpenCV](https://opencv.org/) được bắt đầu từ Intel năm 1999 bởi Gary Bradsky. OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning, và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo real-time.
* OpenCV được phát hành theo *giấy phép BSD***(\*)**, do đó nó miễn phí cho cả học tập và sử dụng với mục đích thương mại. Nó có trên các giao diện C++, C, Python và Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để hỗ trợ hiệu quả về tính toán và chuyên dùng cho các ứng dụng real-time (thời gian thực). Nếu được viết trên C/C++ tối ưu, thư viện này có thể tận dụng được bộ xử lý đa lõi (multi-core processing).
* OpenCV có một cộng đồng người dùng khá hùng hậu hoạt động trên khắp thế giới bởi nhu cầu cần đến nó ngày càng tăng theo xu hướng chạy đua về sử dụng computer vision của các công ty công nghệ. OpenCV hiện được ứng dụng rộng rãi toàn cầu, với cộng đồng hơn 47.000 người, với nhiều mục đích và tính năng khác nhau từ interactive art, đến khai thác mỏ, khai thác web map hoặc qua robotic cao cấp.

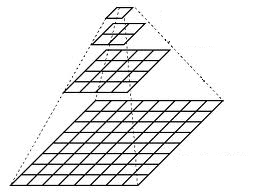
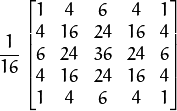
### 2.1.2 Ứng dụng của OpenCv

* OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau bao gồm:
  + Hình ảnh street view
  + Kiểm tra và giám sát tự động
  + Robot và xe hơi tự lái
  + Phân tích hình ảnh y học
  + Tìm kiếm và phục hồi hình ảnh/video
  + Phim – cấu trúc 3D từ chuyển động
  + Nghệ thuật sắp đặt tương tác

### Pyramid

* Pyramid, hoặc pyramid representation, là một dạng biểu diễn tín hiệu đa dạng được dùng trong computer vision, xử lý hình ảnh hoặc các xử lý tín hiệu. Trong đó, một tín hiệu hoặc một hình ảnh được lặp lại với các phương pháp smoothing và subsampling. Tên gọi pyramid biểu thị cho việc biểu diễn trong không gian và các phương pháp phân tích. Một image pyramid thực chất là một tập hợp các hình ảnh mà tất cả chúng đều phát sinh từ một hình ảnh ban đầu duy nhất, hình ảnh này được liên tục được lấy mẫu xuống cho tới khi đạt đến một điểm dừng mong muốn. Có hai loại image pỷamid ảnh phổ biến:
  + Gaussian pyramid: Sử dụng để downsample hình ảnh tức là để giảm độ phân giải của hình ảnh
  + Laplacian pyramid: Được sử dụng để tái tạo lại một hình ảnh được lấy mẫu từ một hình ảnh thấp hơn trong pyramid (có độ phân giải thấp hơn)

### Gaussian Pyramid

* Gaussian pyramid là một tập các layer trong đó layer ở vị trí càng cao thì càng có kích thước nhỏ
* Mỗi layer được đánh số từ dưới lên trên, do đó, layer(i + 1) (được biểu thị là G(i + 1) nhỏ hơn layer i (G(i))
  + Gắn lại G(i) thành một Gaussian kernel:
  + Loại bỏ tất cả các hàng và cột số chẵn
* Bạn có thể dễ dàng nhận thấy rằng hình ảnh kết quả sẽ chỉ chính xác bằng một phần tư ảnh trước đó. Lặp đi lặp lại quá trình nhiều lần với hình ảnh đầu vào G(0) (hình ảnh ban đầu) ta sẽ có một tập layer image pyramid.
* Quá trình trên đã làm giảm độ chính xác của ảnh, vậy để tăng kích thước thì sao? Đầu tiên, ta tăng ma trận hình ảnh lên gấp đôi bản gốc trong mỗi chiều, với các hàng và cột mới đầy những số không (0). Sau đó, thực hiện các bước lặp với cùng một kernel ở trên.

### Laplacian Pyramid

* Laplacian pyramid được tạo ra Gaussian pyramid. Nếu bạn mở một ảnh Laplacian pyramid thì nó là một ảnh chỉ có các đường viền bao ngoài các vật thể, đã xóa hết các phần khác. Hầu hết các phần tử trong ma trận là zeros. Laplacian pyramid thường được sử dụng trong nén hình ảnh. Một level (mức độ) Laplacian pyrmaid được hình thành bởi sự khác biệt giữa level đó trong Gaussian pyramid và level trên nó (layer đã được mở rộng) trong Gaussian pyramid.

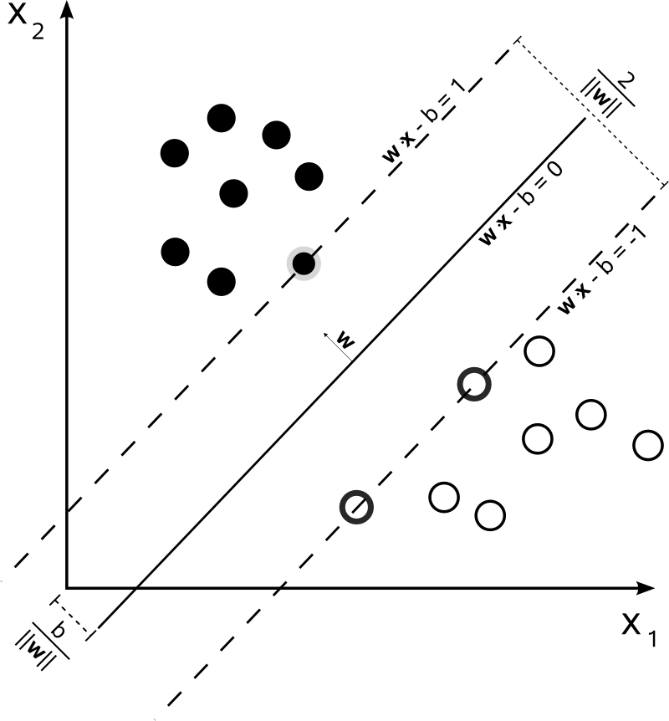
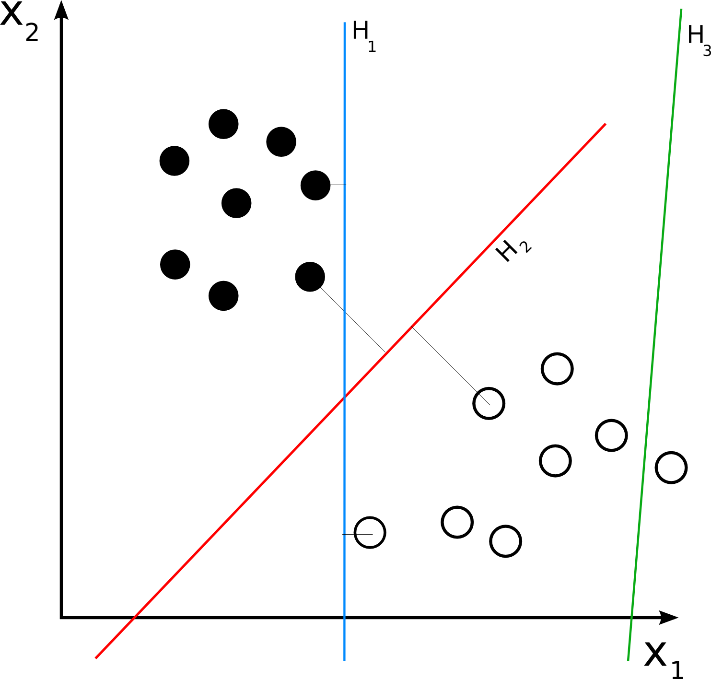
## Thuật toán SVM [2]

### Giới thiệu tổng quan về SVM

* SVM (Support Vector Machine) 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 qui (Regression).
* SVM được đề xuất bởi Vladimir N. Vapnik và các đồng nhiệp của ông vào năm 1963 tại Nga và sau đó trở nên phổ biến trong những năm 90 nhờ ứng dụng giải quyết các bài toán phi tuyến tính (nonlinear) bằng phương pháp Kernel Trick.
* Một máy vectơ hỗ trợ xây dựng một [siêu phẳng](https://vi.wikipedia.org/wiki/Si%C3%AAu_ph%E1%BA%B3ng" \o "Siêu phẳng) hoặc một tập hợp các siêu phẳng trong một không gian nhiều chiều hoặc vô hạn chiều, có thể được sử dụng cho phân loại, hồi quy, hoặc các nhiệm vụ khác. Một cách trực giác, để phân loại tốt nhất thì các siêu phẳng nằm ở càng xa các điểm dữ liệu của tất cả các lớp (gọi là hàm lề) càng tốt, vì nói chung lề càng lớn thì [sai số tổng quát hóa](https://vi.wikipedia.org/w/index.php?title=Sai_s%E1%BB%91_t%E1%BB%95ng_qu%C3%A1t_h%C3%B3a&action=edit&redlink=1" \o "Sai số tổng quát hóa (trang chưa được viết)) của thuật toán phân loại càng bé.
* Trong nhiều trường hợp, không thể phân chia các lớp dữ liệu một cách tuyến tính trong một không gian ban đầu được dùng để mô tả một vấn đề. Vì vậy, nhiều khi cần phải [ánh xạ](https://vi.wikipedia.org/wiki/%C3%81nh_x%E1%BA%A1" \o "Ánh xạ) các điểm dữ liệu trong không gian ban đầu vào một không gian mới nhiều chiều hơn, để việc phân tách chúng trở nên dễ dàng hơn trong không gian mới. Để việc tính toán được hiệu quả, ánh xạ sử dụng trong thuật toán SVM chỉ đòi hỏi [tích vô hướng](https://vi.wikipedia.org/wiki/T%C3%ADch_v%C3%B4_h%C6%B0%E1%BB%9Bng" \o "Tích vô hướng) của các vectơ dữ liệu trong không gian mới có thể được tính dễ dàng từ các tọa độ trong không gian cũ. Tích vô hướng này được xác định bằng một hàm hạt nhân *K*(*x*,*y*) phù hợp.[[1]](https://vi.wikipedia.org/wiki/M%C3%A1y_vect%C6%A1_h%E1%BB%97_tr%E1%BB%A3#cite_note-1) Một siêu phẳng trong không gian mới được định nghĩa là tập hợp các điểm có tích vô hướng với một vectơ cố định trong không gian đó là một hằng số. Vectơ xác định một siêu phẳng sử dụng trong SVM là một tổ hợp tuyến tính của các vectơ dữ liệu luyện tập trong không gian mới với các hệ số *αi*. Với siêu phẳng lựa chọn như trên, các điểm *x* trong không gian đặc trưng được ánh xạ vào một siêu mặt phẳng là các điểm thỏa mãn:
  + - * Σ*i* *αi* *K*(*xi*,*x*) = hằng số.
* Ghi chú rằng nếu *K*(*x*,*y*) nhận giá trị ngày càng nhỏ khi *y* xa dần khỏi *x* thì mỗi số hạng của tổng trên được dùng để đo độ tương tự giữa *x* với điểm *xi* tương ứng trong dữ liệu luyện tập. Như vậy, tác dụng của tổng trên chính là so sánh khoảng cách giữa điểm cần dự đoán với các điểm dữ liệu đã biết. Lưu ý là tập hợp các điểm *x* được ánh xạ vào một siêu phẳng có thể có độ phức tạp tùy ý trong không gian ban đầu, nên có thể phân tách các tập hợp thậm chí không lồi trong không gian ban đầu.

### Đặt vấn đề

* [Phân loại thống kê](https://vi.wikipedia.org/wiki/Ph%C3%A2n_lo%E1%BA%A1i_th%E1%BB%91ng_k%C3%AA) là một nhiệm vụ phổ biến trong [học máy](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_m%C3%A1y). Trong mô hình học có giám sát, thuật toán được cho trước một số điểm dữ liệu cùng với nhãn của chúng thuộc một trong hai lớp cho trước. Mục tiêu của thuật toán là xác định xem một điểm dữ liệu *mới* sẽ được thuộc về lớp nào. Mỗi điểm dữ liệu được biểu diễn dưới dạng một vector p-chiều, và ta muốn biết liệu có thể chia tách hai lớp dữ liệu bằng một [siêu phẳng](https://vi.wikipedia.org/wiki/Si%C3%AAu_ph%E1%BA%B3ng) *p* − 1 chiều. Đây gọi là [phân loại tuyến tính](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_lo%E1%BA%A1i_tuy%E1%BA%BFn_t%C3%ADnh&action=edit&redlink=1). Có nhiều siêu phẳng có thể phân loại được dữ liệu. Một lựa chọn hợp lý trong chúng là siêu phẳng có lề lớn nhất giữa hai lớp.

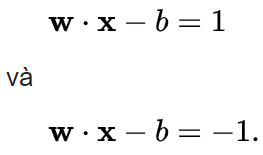


### SVM tuyến tính

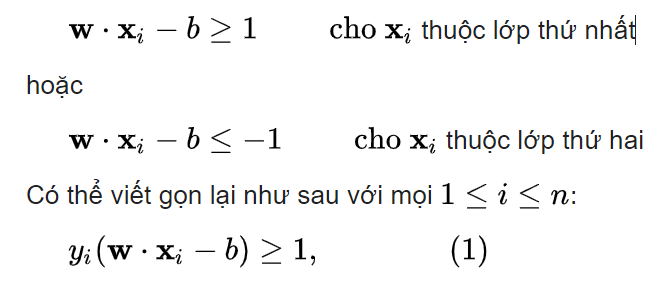
* Ta có một tập huấn luyện {\displaystyle {\mathcal {D}}}  gồm *n* điểm có dạng

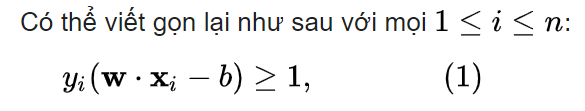


* với *yi* mang giá trị 1 hoặc −1, xác định lớp của điểm {\displaystyle \mathbf {x} \_{i}} . Mỗi {\displaystyle \mathbf {x} \_{i}}  là một vectơ thực *p*-chiều. Ta cần tìm siêu phẳng có lề lớn nhất chia tách các điểm có {\displaystyle y\_{i}=1}  và các điểm có {\displaystyle y\_{i}=-1} . Mỗi siêu phẳng đều có thể được viết dưới dạng một tập hợp các điểm {\displaystyle \mathbf {x} } **x** thỏa mãn với {\displaystyle \cdot } ký hiệu cho [tích vô hướng](https://vi.wikipedia.org/wiki/T%C3%ADch_v%C3%B4_h%C6%B0%E1%BB%9Bng) và {\displaystyle {\mathbf {w} }}**w** là một [vectơ pháp tuyến](https://vi.wikipedia.org/w/index.php?title=Vect%C6%A1_ph%C3%A1p_tuy%E1%BA%BFn&action=edit&redlink=1) của siêu phẳng. Tham số {\displaystyle {\tfrac {b}{\|\mathbf {w} \|}}}  xác định khoảng cách giữa gốc tọa độ và siêu phẳng theo hướng vectơ pháp tuyến {\displaystyle {\mathbf {w} }}**w**.
* Chúng ta cần chọn {\displaystyle {\mathbf {w} }}w và {\displaystyle b}b để cực đại hóa lề, hay khoảng cách giữa hai siêu mặt song song ở xa nhau nhất có thể trong khi vẫn phân chia được dữ liệu. Các siêu mặt ấy được xác định bằng

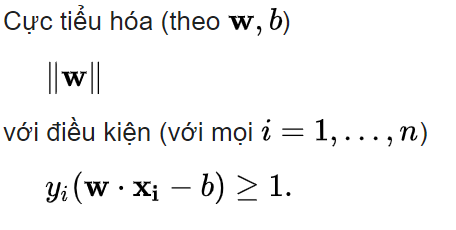


* Để ý rằng nếu dữ liệu huấn luyện có thể được chia tách một cách tuyến tính, thì ta có thể chọn hai siêu phẳng của lề sao cho không có điểm nào ở giữa chúng và sau đó tăng khoảng cách giữa chúng đến tối đa có thể. Bằng hình học, ta tìm được khoảng cách giữa hai siêu phẳng là {\displaystyle {\tfrac {2}{\|\mathbf {w} \|}}} . Vì vậy ta muốn cực tiểu hóa giá trị {\displaystyle \|\mathbf {w} \|} . Để đảm bảo không có điểm dữ liệu nào trong lề, ta thêm vào các điều kiện sau:
* Với mỗi ta có





* Tóm lại ta tối ưu hóa như sau:



* **Dạng ban đầu**
  + Bài toán tối ưu ở mục trên tương đối khó giải vì hàm mục tiêu phụ thuộc vào ||**w**||, là một hàm có khai căn. Tuy nhiên có thể thay ||**w**|| bằng hàm mục tiêu {\displaystyle {\tfrac {1}{2}}\|\mathbf {w} \|^{2}}  (hệ số 1/2 để tiện cho các biến đổi toán học sau này) mà không làm thay đổi lời giải (lời giải của bài toán mới và bài toán ban đầu có cùng **w** và b). Đây là một bài toán [quy hoạch toàn phương](https://vi.wikipedia.org/wiki/Quy_ho%E1%BA%A1ch_to%C3%A0n_ph%C6%B0%C6%A1ng" \o "Quy hoạch toàn phương). Cụ thể hơn:

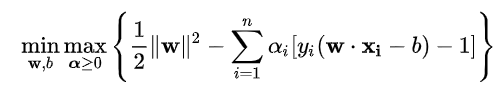
Cực tiểu hóa (theo {\displaystyle {\mathbf {w} ,b}}**w, b**)



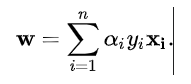
* + với điều kiện (với mọi {\displaystyle i=1,\dots ,n}i=1,…,n)



* + Bằng cách thêm các nhân tử Lagrange {\displaystyle {\boldsymbol {\alpha }}}  bài toán trên trở thành



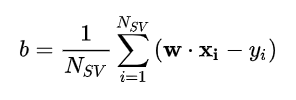
* + nghĩa là ta cần tìm một [điểm yên ngựa](https://vi.wikipedia.org/w/index.php?title=%C4%90i%E1%BB%83m_y%C3%AAn_ng%E1%BB%B1a&action=edit&redlink=1). Khi đó, tất cả các điểm không nằm trên lề, nghĩa là {\displaystyle y\_{i}(\mathbf {w} \cdot \mathbf {x\_{i}} -b)-1>0}  đều không ảnh hưởng đến giá trị hàm mục tiêu vì ta có thể chọn {\displaystyle \alpha \_{i}}  bằng không.



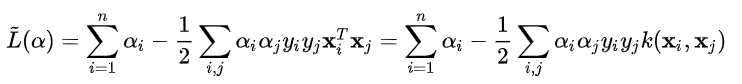
* + Chỉ có một vài {\displaystyle \alpha \_{i}}  nhận giá trị lớn hơn 0. Các điểm {\displaystyle \mathbf {x\_{i}} }  tương ứng là các *vectơ hỗ trợ* nằm trên lề và thỏa mãn {\displaystyle y\_{i}(\mathbf {w} \cdot \mathbf {x\_{i}} -b)=1} . Từ điều kiện này, ta nhận thấy



từ đó ta suy ra được giá trị {\displaystyle b}b. Trên thực tế, một cách thức tốt hơn để tính {\displaystyle b}b là tính giá trị trung bình từ tất cả {\displaystyle N\_{SV}}  vectơ hỗ trợ:



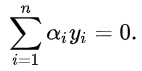
* **Dạng đối ngẫu**
  + Nếu viết điều kiện phân loại dưới [dạng đối ngẫu](https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_%C4%91%E1%BB%91i_ng%E1%BA%ABu) không điều kiện thì sẽ dễ dàng nhận thấy siêu phẳng với lề lớn nhất, và do đó nhiệm vụ phân loại, chỉ phụ thuộc vào các điểm luyện tập nằm trên lề, còn gọi là các *vectơ hỗ trợ*.
  +  ta nhận thấy bài toán đối ngẫu của SVM là chính là bài toán tối ưu hóa sau:
  + Cực đại hóa (theo {\displaystyle \alpha \_{i}}ai)



* + Với điều kiện (với mọi i = 1,…,n)

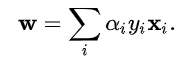


và điều kiện sau ứng với việc cực tiểu hóa theo {\displaystyle b}b.



Ở đây hàm hạt nhân được định nghĩa là 

* + Sau khi giải xong, có thể tính {\displaystyle \mathbf {w} }**w** từ các giá trị {\displaystyle \alpha }  tìm được như sau:



### Ứng dụng của SVM trong thực tế

* Nhận dang kí tự quang học

## Dataset Car

* Dataset Car là một dataset đa dạng các thuộc tính thuộc về đối tượng xe máy/ xe hơi. Với lượng data lớn sẽ train cho model SVM nhận dạng từng kí tự trong biển số xe và cho ra kết quả giúp quá trình nhận dạng trở nên tốt hơn.

## Cài đặt thuật toán

### Các thư viện được sử dụng

#### **Numpy**

* Numpy là gì: là một thư viện toán học phổ biến và mạnh mẽ nhất của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi sử dụng “core Python” đơn thuần
* Cài đặt thư viện Numpy: pip install numpy

#### **OpenCV**

* OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV đươc viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực. Opencv có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOs lẫn Android, iOS OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần.
* Cài đặt thư viện Numpy: pip install opencv

#### **Tensorflow**

* **Cách thức hoạt động của Tensorflow**
  + **TensorFlow** cho phép các nhà phát triển tạo các biểu đồ dataflow – các cấu trúc mô tả cách thức dữ liệu di chuyển qua biểu đồ hoặc một loạt các node xử lý. Mỗi node trong biểu đồ đại diện cho một hoạt động toán học và mỗi kết nối giữa các node là một mảng dữ liệu đa chiều hoặc tenxor.
  + TensorFlow cung cấp tất cả những điều này cho lập trình viên bằng ngôn ngữ Python. Python rất dễ học và làm việc, nó cung cấp các cách thuận tiện để diễn tả mức độ trừu tượng cao được ghép với nhau. Các node và tenxor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow chính là các ứng dụng Python.
  + Các hoạt động toán học thực tế, tuy nhiên chúng không được thực hiện trong Python. Các thư viện biến đổi có sẵn thông qua **TensorFlow** được viết dưới dạng nhị phân C ++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các tóm tắt lập trình cấp cao để nối chúng lại với nhau.
  + Các ứng dụng **TensorFlow** có thể chạy trên hầu hết mọi mục tiêu thuận tiện: một máy cục bộ, một cụm trong cloud, thiết bị iOS và Android, CPU hoặc GPU. Nếu bạn sử dụng cloud của Google, bạn có thể chạy **TensorFlow** trên silicon của đơn vị xử lý TensorFlow (TPU) tùy chỉnh của Google để tăng tốc hơn nữa. Tuy nhiên, các mô hình kết quả được tạo bởi **TensorFlow**, có thể được triển khai trên hầu hết mọi thiết bị nơi chúng được sử dụng để phục vụ những dự đoán.
  + **TensorFlow 2.0**, được phát hành vào tháng 10 năm 2019, đã cải thiện framework theo nhiều cách dựa trên phản hồi của người dùng, để làm việc dễ dàng hơn (ví dụ: bằng cách sử dụng Keras API tương đối đơn giản để đào tạo mô hình) và hiệu quả hơn. Đào tạo phân tán dễ chạy hơn nhờ API mới và hỗ trợ cho **TensorFlow Lite** cho phép triển khai các mô hình trên nhiều nền tảng khác nhau. Tuy nhiên, code được viết cho các phiên bản trước của **TensorFlow** đôi khi chỉ được viết lại một chút, đôi khi đáng kể, để tận dụng tối đa các tính năng mới của **TensorFlow 2.0**.



Hình 2.4‑1 - TensorFlow

* **Lợi ích của tensorflow**
  + Lợi ích lớn nhất mà cũng là duy nhất **TensorFlow** cung cấp cho sự phát triển của **machine learning** là không tưởng. Thay vì xử lý các chi tiết khó hiểu khi thực hiện các **thuật toán** tìm ra cách thích hợp để output của một function sang input của một function khác, nhà phát triển có thể tập trung vào logic tổng thể của ứng dụng.
  + TensorFlow cung cấp các tiện ích bổ sung cho các nhà phát triển, những người cần debug và hướng vào các ứng dụng TensorFlow. Chế độ **Eager execution**cho phép bạn đánh giá và sửa đổi từng hoạt động của biểu đồ một cách riêng biệt và minh bạch, thay vì xây dựng toàn bộ biểu đồ dưới dạng một đối tượng mờ đục và đánh giá tất cả cùng một lúc. Bộ phần mềm trực quan **TensorBoard** cho phép bạn kiểm tra và lập hồ sơ cách thức biểu đồ chạy bằng bảng điều khiển dựa trên web tương tác.
  + Một cảnh báo: Một số chi tiết về việc triển khai **TensorFlow**, dẫn đến việc khó có được kết quả hoàn toàn xác định cho một số công việc đào tạo. Đôi khi một mô hình được đào tạo trên một hệ thống sẽ thay đổi một chút so với mô hình được đào tạo trên một hệ thống khác, ngay cả khi chúng được cung cấp cùng một dữ liệu. Điều đó có nghĩa rằng, có thể giải quyết các vấn đề đó và TensorFlow đang xem xét nhiều kiểm soát hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.
  + Hiện nay [iRender](https://irender.vn/) đang cung cấp dịch vụ [GPUhub](https://irender.vn/price/), sử dụng hiệu suất máy tính vào lĩnh vực **rendering**, **AI/deep learning**. Đối với việc cài đặt phần mềm **tensorFlow** bạn không phải lo lắng vì ngay khi kết nối sử dụng đến hệ thống máy tính của chúng tôi các phần mềm này đều được cài đặt sẵn. Việc của bạn là kết nối đến máy và sử dụng nó, mọi thao tác chỉ đơn giản bằng vài cái click. Hãy liên hệ ngay với chúng tôi để được hỗ trợ nếu bạn gặp bất cứ rắc rối nào.

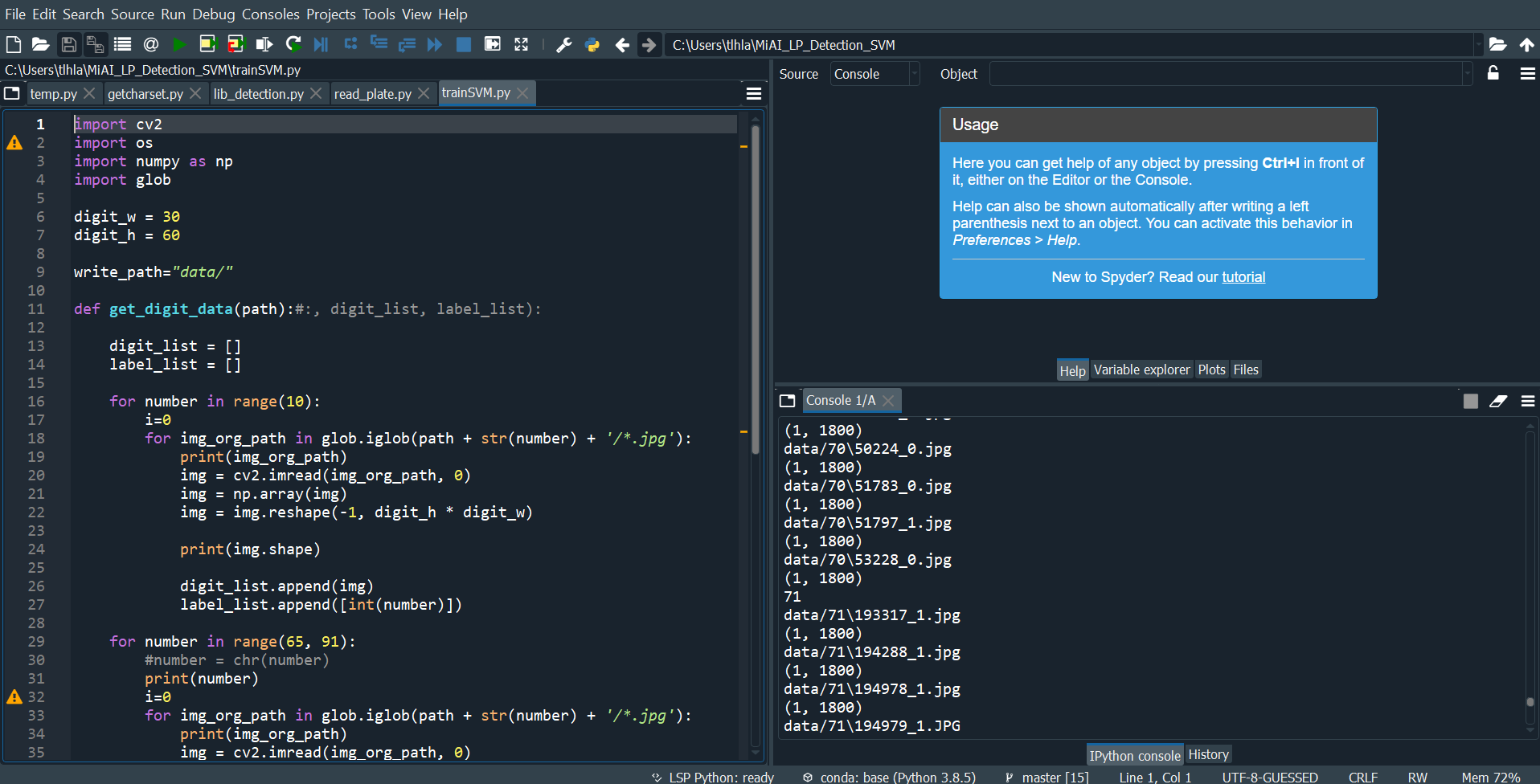
### Cài đặt và kiểm thử

* Chuẩn bị data: tìm kiếm nguồn dữ liệu train cho model.

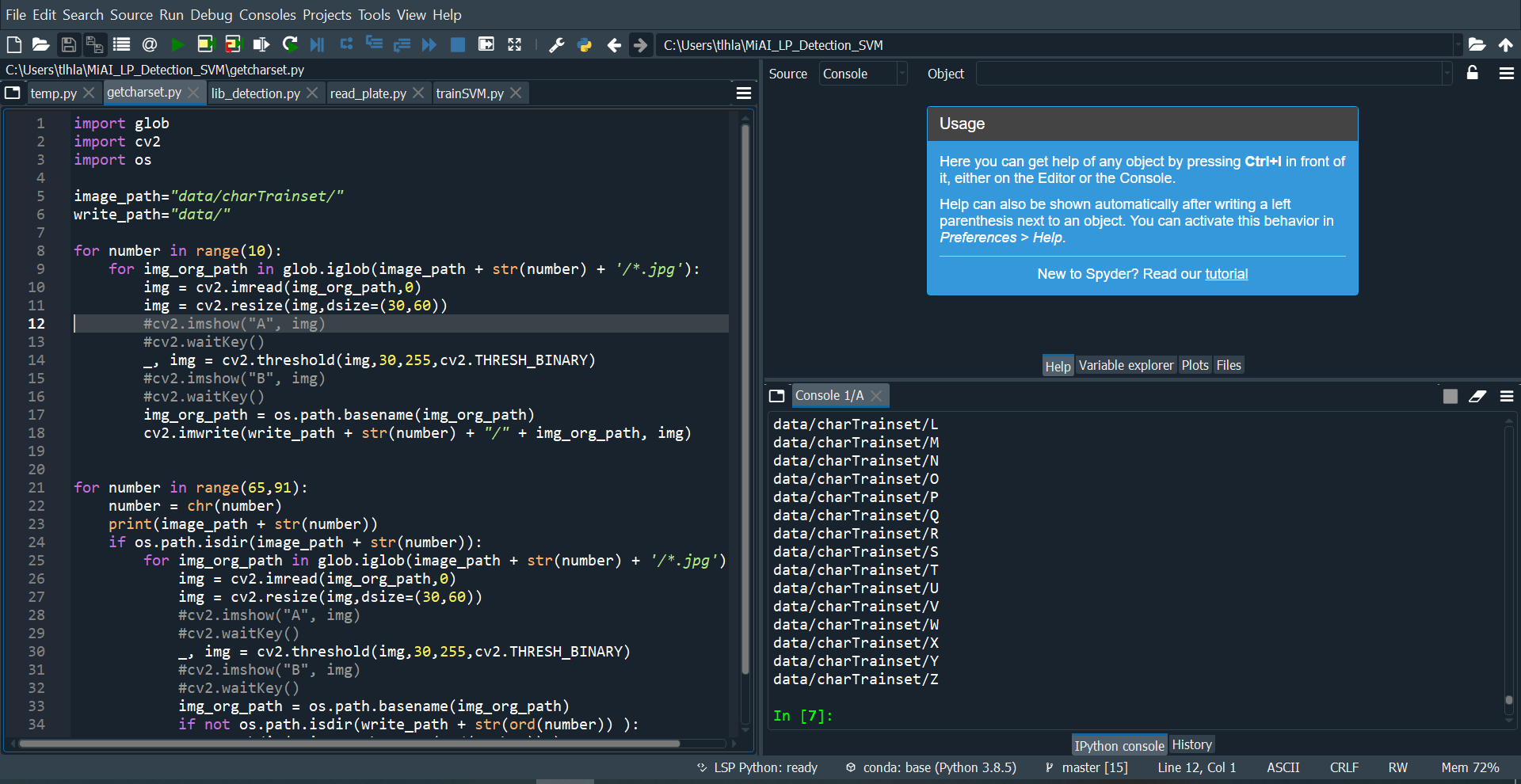


Hình 2.4‑2 - Dataset Ký tự

* Train Model SVM:



Hình 2.4‑3 - Train Model SVM

* Get Charset:

Hình 2.4‑4 - Train Charset

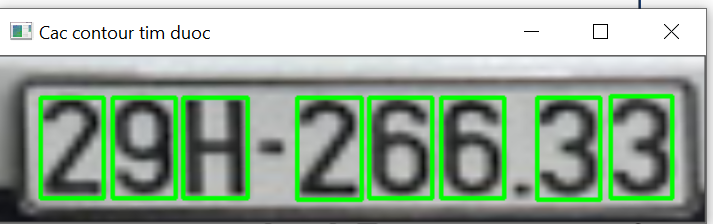
* Kiểm thử:



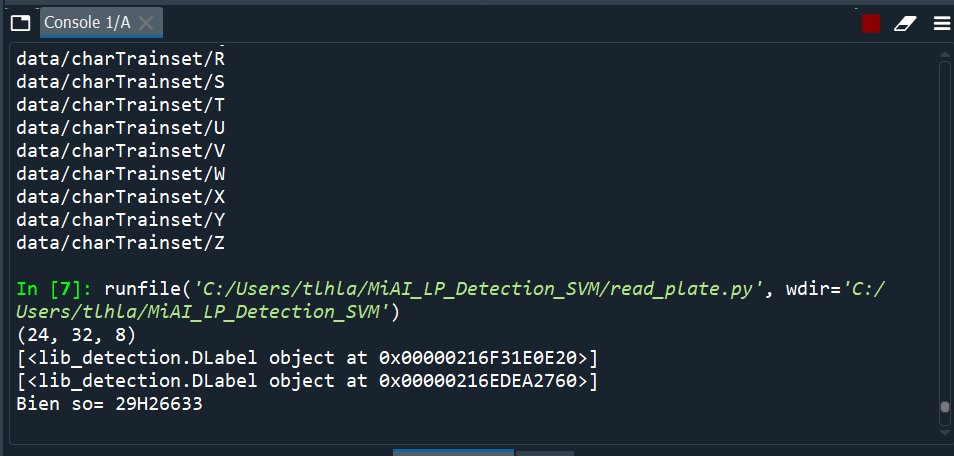
Hình 2.4‑5 - Hình ảnh gốc



Hình 2.4‑6 - Ảnh biển số sau threshold



Hình 2.4‑7 - Các contour tìm được



Hình 2.4‑8 - Kết quả nhận dạng biển số

# KẾT LUẬN

Với mục tiêu ban đầu là ứng dụng OpenCV và thuật toán SVM vào bài toán nhận dạng bảng số xe. Nhóm đã tìm hiểu và nhận thấy đã thực hiện được khoảng 90%. Đưa ra được một model hợp lý. Có thể nhận diện được các biển số xe.

Sau khi hoàn thành đồ án, nhóm nhận thấy mình đã hiểu hơn về Machine learning nói chung, và thuật toán OpenCV, SVM nói riêng. Biết cách xử lý dữ liệu cho hợp lý, áp dụng thuật toán, phân tích lỗi và tối ưu model để cho ra kết quả tốt nhất.

# TÀI LIỆU THAM KHẢO

[1] Tổng quan về OpenCV . Retrieved from: https://viblo.asia/p/tim-hieu-ve-opencv-Do754NrXZM6

[2]Tìm hiểu thuật toán SVM – machine learning. Retrieved from: https://vi.wikipedia.org/wiki/M%C3%A1y\_vect%C6%A1\_h%E1%BB%97\_tr%E1%BB%A3