

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

XE TỰ HÀNH KHÔNG NGƯỜI LÁI

HỘI ĐỒNG: Kỹ thuật máy tính

GVHD: ThS. Trần Thanh Bình

GVPB: ThS. Nguyễn Cao Trí

—o0o—

SVTH 1: Nguyễn Quốc Mạnh (1813043)

SVTH 2: Huỳnh Trương Quốc Khánh
(1810990)

TP. HỒ CHÍ MINH, 12/2021

Lời cam đoan

Nhóm sinh viên thực hiện xin cam đoan rằng đề cương luận văn: "**XE TỰ HÀNH KHÔNG NGƯỜI LÁI**" là kết quả nghiên cứu do nhóm thực hiện dưới sự hướng dẫn của thầy ThS. Trần Thanh Bình.

Những tài liệu tham khảo, nội dung trích dẫn được trình bày chi tiết, cụ thể. Còn lại những nội dung khác chưa từng được công bố hoặc sử dụng để nhận bằng cấp ở những nơi khác.

Nếu phát hiện có bất kỳ sự gian lận nào, nhóm xin hoàn toàn chịu trách nhiệm về nội dung đề cương luận văn của mình. Trường đại học Bách Khoa TP. Hồ Chí Minh không liên quan đến những vi phạm (nếu có) về tác quyền, bản quyền do nhóm gây ra trong quá trình thực hiện.

TP.Hồ Chí Minh, ngày 06 tháng 12 năm 2021

Nhóm sinh viên thực hiện

Nguyễn Quốc Mạnh
Huỳnh Trương Quốc Khánh

Lời cảm ơn

Để hoàn thành đề cương luận văn này, nhóm sinh viên thực hiện xin gửi lời cảm ơn tới:

Quý thầy, cô trong Khoa Khoa học và Kỹ thuật máy tính, bộ môn Kỹ thuật máy tính, trường Đại học Bách Khoa TP. Hồ Chí Minh, những người đã hết lòng truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Đặc biệt, nhóm xin gửi lời cảm ơn chân thành và trân trọng nhất đến thầy **ThS. Trần Thanh Bình** đã tận tình hướng dẫn, giúp đỡ nhóm trong suốt quá trình nghiên cứu và thực hiện đề tài. Ngoài ra, nhóm xin được gửi lời cảm ơn chân thành đến thầy *TS. Nguyễn Trần Hữu Nguyên* - GVCN lớp Kỹ thuật máy tính khoá 2018 đã giúp đỡ, định hướng cho chúng em trong quá trình học tập, những kiến thức được thầy truyền đạt là hành trang quý báu cho chúng em trên con đường học tập, công tác và nghiên cứu sau này.

Xin gửi lời cảm ơn chân thành đến tất cả các bạn lớp Kỹ thuật máy tính khoá 2018 đã luôn giúp đỡ và hỗ trợ trong suốt quá trình học tập lý thuyết tại trường.

Xin chân thành cảm ơn!

Tóm tắt nội dung

Đề cương luận văn tập trung vào việc nghiên cứu và mô phỏng xe tự hành có khả năng di chuyển theo làn đường và biển báo giao thông sử dụng Hệ điều hành robot (ROS) và công cụ xử lý ảnh OpenCV. Hệ thống được đánh giá thử nghiệm trên công cụ mô phỏng Gazebo với robot Turtlebot3.

Mục lục

1	Giới thiệu	1
1.1	Mở đầu	1
1.2	Yêu cầu và mục tiêu của đề tài	1
1.2.1	Yêu cầu	1
1.2.2	Mục tiêu	1
1.3	Bố cục của luận văn	2
1.4	Giới thiệu các công cụ	2
1.4.1	Robot Operating System (ROS)	2
1.4.2	OpenCV cho Python	3
1.4.3	Gazebo	3
2	Xây dựng và mô phỏng hệ thống	4
2.1	Giải thuật phát hiện làn đường	4
2.1.1	Thresholding	4
2.1.2	Warping	4
2.1.3	Pixel Summation	4
2.2	Xây dựng bộ điều khiển cho robot	5
2.2.1	Lấy dữ liệu từ camera	5
2.2.2	Di chuyển theo độ cong của làn đường	6
2.3	Mô phỏng	6
3	Nhận diện biển báo	8
3.1	Xây dựng model CNN hỗ trợ nhận diện biển báo	8
3.1.1	CNN model	8
3.1.2	Keras	10
3.1.3	Xây dựng model CNN	10
3.1.4	Nhận diện bằng openCV	17
4	Kết luận và hướng phát triển	21
4.1	Đánh giá kết quả thực hiện	21
4.2	Những mặt hạn chế	21
4.3	Hướng phát triển	21

Danh sách hình vẽ

2.1	<i>Region of Interest</i>	5
2.2	<i>Ảnh qua bước Warping</i>	5
2.3	<i>Ảnh qua bước mapping</i>	5
2.4	<i>Giá trị độ cong</i>	5
2.5	<i>Thế giới dùng trong mô phỏng</i>	7

Chương 1

Giới thiệu

1.1 Mở đầu

Xe tự hành không người lái là mục tiêu phát triển trong tương lai của ngành giao thông vận tải. Một chiếc xe để có thể tự lái cần phải hiểu rõ môi trường xung quanh và tìm được đường đi với sự trợ giúp tối thiểu của con người. Giữ an toàn cho người lái và hành khách là yêu cầu quan trọng trong việc phát triển xe tự hành. Bám theo làn đường và hỗ trợ rẽ là một số hệ thống cho phép các phương tiện phát hiện làn đường và giúp giữ xe ở vị trí lý tưởng so với làn đường và hỗ trợ trong việc rẽ của xe.

1.2 Yêu cầu và mục tiêu của đề tài

1.2.1 Yêu cầu

- Nghiên cứu ROS (Robot Operating System).
- Nghiên cứu kỹ thuật xử lý hình ảnh bằng OpenCV và Machine learning.
- Làm quen với mô phỏng trên Gazebo.
- Xây dựng 1 hệ thống nhận diện hình ảnh và điều khiển robot tự động di chuyển.
- Thử nghiệm hệ thống trên môi trường mô phỏng Gazebo tích hợp ROS sử dụng robot Turtlebot3.

1.2.2 Mục tiêu

Về kiến thức

- Nắm vững kiến thức về ROS, sử dụng ROS để điều khiển robot.
- Nắm vững các kiến thức lập trình python, ROS
- Nắm vững các kỹ thuật xử lý hình ảnh: nhận diện màu sắc, machine learning,...
- Phân tích, giải quyết yêu cầu bài toán nhận diện hình ảnh bằng OpenCV.

Về sản phẩm

- Robot có thể nhận diện được làn đường và các biển báo điều hướng từ đó di chuyển đúng làn đường được thiết kế sẵn.
- Nắm được quy trình phát triển sản phẩm: phân tích - thiết kế - hiện thực - kiểm tra.

1.3 Bố cục của luận văn

- **Chương 1: Giới thiệu**
 - Giới thiệu tổng quan về đề tài.
 - Trình bày yêu cầu, mục tiêu, đối tượng và phạm vi nghiên cứu.
- **Chương 2: Xây dựng mô hình và bộ điều khiển cho robot**
 - Giải thuật tìm lân đường
 - Xây dựng mô hình hỗ trợ nhận diện biển báo
 - Xây dựng bộ điều khiển cho robot
 - Xây dựng thế giới dùng trong mô phỏng
- **Chương 3: Nhận diện biển báo**
 - Xây dựng model CNN hỗ trợ nhận diện biển báo
- **Chương 4: Kết luận và hướng phát triển**
 - Đánh giá về ưu nhược điểm đề tài đã thực hiện.
 - Hướng hoàn thiện và phát triển đề tài.

1.4 Giới thiệu các công cụ

1.4.1 Robot Operating System (ROS)

Robot operating system là một hệ thống phần mềm chuyên dụng để lập trình và điều khiển robot, bao gồm các công cụ để lập trình, hiển thị, tương tác trực tiếp với phần cứng, và kết nối cộng đồng robot trên toàn thế giới.

Nó không chỉ cung cấp các dịch vụ hệ điều hành tiêu chuẩn (trừu tượng hóa phần cứng, quản lý tranh chấp, quản lý quy trình) mà còn cung cấp các chức năng cấp cao (cơ sở dữ liệu tập trung, hệ thống cấu hình robot, v.v.). Nó cũng cung cấp các công cụ và thư viện để xây dựng, viết và chạy mã nguồn trên nhiều máy tính.

Một số khái niệm trong ROS:

- **Package:** Package là đơn vị tổ chức phần mềm của mã ROS. Mỗi package có thể chứa thư viện, tệp thực thi, tập lệnh.
- **Node:** Một node là một process thực hiện tính toán. Các node được kết hợp với nhau thành một mạng lưới và giao tiếp với nhau bằng cách sử dụng các luồng topic, dịch vụ RPC và Máy chủ tham số (Parameter Server). Các node này hoạt động ở quy mô chi tiết. Một hệ thống điều khiển robot thường sẽ bao gồm nhiều node. Ví dụ: một node điều khiển công cụ laser xác định khoảng cách, một node điều khiển động cơ bánh xe của robot, một node thực hiện bản địa hóa (localization), một node thực hiện lập kế hoạch đường đi, một node cung cấp chế độ xem đồ họa của hệ thống.
- **Message:** Các node giao tiếp với nhau bằng cách publish các message đến các topic. Một message là một cấu trúc dữ liệu đơn giản, bao gồm các trường đã nhập. Các kiểu nguyên thủy tiêu chuẩn (số nguyên, dấu phẩy động, boolean, v.v) được hỗ trợ, cũng như các mảng của kiểu nguyên thủy. Message có thể bao gồm các cấu trúc và mảng được lồng nhau tùy ý (giống như cấu trúc C).
- **Topic:** Topic là các "tuyến đường" được đặt tên mà các nút trao đổi message. Các topic có cơ chế publish/subscribe ẩn danh, giúp tách rời việc truyền thông tin khỏi việc nhận nó.

Các node thường không nhận thức được chúng đang giao tiếp với ai. Thay vào đó, các node quan tâm đến dữ liệu cần thiết sẽ subscribe vào topic có liên quan; các node tạo dữ liệu sẽ publish đến topic có liên quan. Có thể có nhiều publisher và subscriber trên cùng một topic.

1.4.2 OpenCV cho Python

Project OpenCV 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 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.

Python là ngôn ngữ được dùng nhiều để demo/test OpenCV do tính ngắn gọn, ít phải thiết lập. Bên cạnh đó, nếu dùng Python thì cũng có thể code được trên nhiều hệ điều hành.

1.4.3 Gazebo

Gazebo là một công cụ mô phỏng robot 3D mã nguồn mở, Gazebo có thể sử dụng nhiều công cụ vật lý (physics engines) hiệu suất cao, chẳng hạn như ODE, Bullet, v.v. (mặc định là ODE). Nó cho phép kết xuất (render) môi trường rất chân thực với ánh sáng, bóng và kết cấu. Nó có thể mô hình hóa các cảm biến cảm nhận môi trường mô phỏng như laser xác định khoảng cách, máy ảnh (bao gồm cả góc rộng), cảm biến kiểu Kinect.

Chương 2

Xây dựng và mô phỏng hệ thống

2.1 Giải thuật phát hiện làn đường

2.1.1 Thresholding

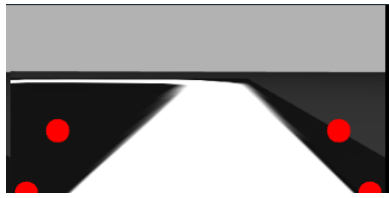
Kỹ thuật Thresholding (Phân ngưỡng ảnh) theo màu sắc được thực hiện để phát hiện màu sắc của làn đường. Kỹ thuật Thresholding không được áp dụng trực tiếp trên hình ảnh từ dữ liệu camera của robot mà phải qua một số chuyển đổi. Hình ảnh có định dạng BGR (Blue Green Red) được chuyển sang định dạng HSV (Hue Saturation Value) để phát hiện màu sắc dễ dàng hơn. Không gian màu HSV thể hiện cách màu sắc đáp ứng với ánh sáng. Biểu diễn dưới dạng toán học, HSV bao gồm ba ma trận cho "Hue", "Saturation" và "Value" với giá trị dao động trong khoảng lần lượt là 0-179, 0-255 và 0-255. Ứng dụng chính của HSV là phân vùng ảnh dựa trên màu sắc để phân tách làn đường ra khỏi môi trường xung quanh. Kỹ thuật Thresholding sẽ chuyển hình ảnh thành một ảnh nhị phân, trong đó màu sắc của làn đường được phát hiện sẽ có màu trắng và phần còn lại của hình ảnh sẽ có màu đen.

2.1.2 Warping

Hình ảnh có được từ dữ liệu camera của robot bị ảnh hưởng bởi góc nhìn. Vì vậy, ranh giới của làn đường không thể được nhận diện chính xác sử dụng hình ảnh này. Kỹ thuật Warping (Nắn ảnh) được sử dụng để điều chỉnh hình ảnh. Kỹ thuật Warping được áp dụng nhằm lấy được góc nhìn làn đường theo hướng từ trên xuống bằng cách thay đổi góc nhìn của hình ảnh. Để sử dụng Warping, một bộ bốn điểm được đặt lên hình ảnh. Giá trị của các điểm này sẽ quyết định Region of Interest (Vùng quan tâm - ROI). Việc này được thực hiện bằng cách truyền tọa độ các điểm vào các hàm của thư viện OpenCV. Kết quả nhận được sẽ là một hình ảnh được cắt theo Region of Interest với góc nhìn từ trên xuống. Warping là một trong những kỹ thuật tiền xử lý ảnh phổ biến nhất được sử dụng trong các hệ thống phát hiện làn đường vì nó cung cấp một hình ảnh mà các tính toán tiếp theo có thể được áp dụng lên một cách dễ dàng hơn so với hình ảnh nguyên bản lấy được từ dữ liệu camera. Ý tưởng cơ bản là có được một hình chữ nhật khi đoạn đường đang thẳng. [1]

2.1.3 Pixel Summation

Để tìm được độ cong của đường, việc cộng các pixel trên ảnh được thực hiện thông qua các bước:



Hình 2.1: *Region of Interest*

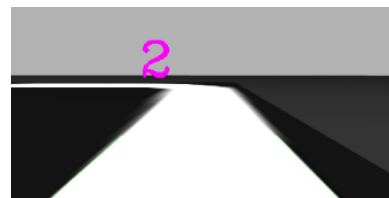


Hình 2.2: *Ảnh qua bước Warping*

- Hình ảnh đã qua các bước Thresholding và Warping là một ảnh nhị phân, tức là ảnh chỉ có các pixel đen và trắng.
- Bước tiếp theo bao gồm định vị và lập bản đồ (mapping) làn đường trên hình ảnh đã xử lý bằng cách sử dụng biểu đồ tần suất (histogram). Trong lĩnh vực xử lý ảnh, biểu đồ tần suất được dùng để thống kê số lần xuất hiện các mức sáng trong ảnh.
- Trong trường hợp này, hình ảnh được phân ngưỡng dưới dạng nhị phân nên biểu đồ tần suất sẽ biểu diễn các điểm đen (điểm không) và trắng (điểm khác không). Các pixel khác không thể hiện vị trí của làn đường, vị trí của pixel bắt đầu từ góc dưới trái của ảnh theo trục x thể hiện vị trí bắt đầu của làn đường.
- Có khả năng một số pixel trong hình ảnh bị ảnh hưởng bởi nhiễu. Để tránh được chúng trong quá trình tính toán, một giá trị ngưỡng được thiết lập để quyết định một cột bất kỳ có phải là một phần của làn đường hay là nhiễu.
- Giá trị trung bình của biểu đồ tần suất của một phần tư dưới cùng của hình ảnh sẽ được xem là điểm gốc (base point) hay trung tâm của làn đường. Giá trị trung bình của biểu đồ tần suất của cả hình ảnh sẽ được xem là điểm giữa (middle point). Vậy lấy điểm giữa trừ đi điểm gốc sẽ cho ta một giá trị độ cong có thể được hiệu chỉnh để sử dụng cho các tính toán tiếp theo như tìm ra góc rẽ cho xe/robot.[2]



Hình 2.3: *Ảnh qua bước mapping*



Hình 2.4: *Giá trị độ cong*

2.2 Xây dựng bộ điều khiển cho robot

2.2.1 Lấy dữ liệu từ camera

Subscribe vào topic `/camera/image` để lấy dữ liệu trực tiếp từ camera của robot

```
1 class Test_Img:
2     def __init__(self):
3         self.cv_bridge = CvBridge()
4         self.image_sub = rospy.Subscriber("/camera/image", Image
, self.callback)
```

Mỗi khi nhận dữ liệu từ camera, gọi hàm *callback* để xử lý ảnh nhận được, cần chuyển ảnh từ định dạng message của ROS thành hình ảnh có thể xử lý với OpenCV và thay đổi kích thước thành 480x240 pixel trước khi tiến hành xử lý để tìm được giá trị độ cong của làn đường

```
1 def callback(self, data):
2     try:
3         cv_image = self.cv_bridge.imgmsg_to_cv2(data, "bgr8")
4     except CvBridgeError as e:
5         print(e)
6     cv_image = cv2.resize(cv_image, (480, 240))
```

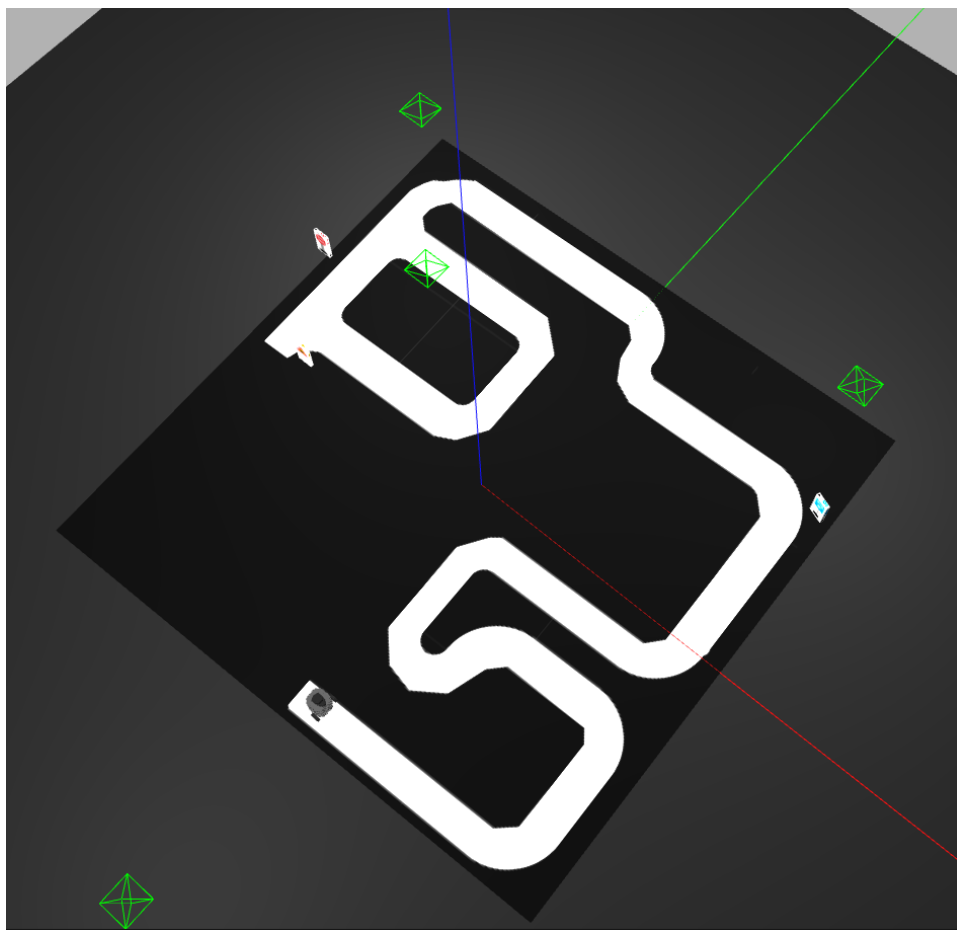
2.2.2 Di chuyển theo độ cong của làn đường

Xây dựng hàm điều khiển việc di chuyển của robot theo giá trị độ cong của làn đường. Góc lệch khi di chuyển của robot cho phù hợp với giá trị độ cong của làn đường được tìm bằng cách thử nghiệm các giá trị góc lệch khác nhau cho đến khi tối ưu.

```
1 def move(self, curveVal):
2     cmd_vel = Twist()
3     cmd_vel.linear.x = 0.1
4     cmd_vel.angular.z = 0.0
5
6     NegativeCurve = False
7     if curveVal > 0: NegativeCurve = True
8
9     if curveVal < 0.2 and curveVal > -0.2: curveVal = 0.0
10    elif curveVal < 0.4 and curveVal > -0.4: curveVal = 0.123
11    elif curveVal < 0.5 and curveVal > -0.5: curveVal = 0.123
12    elif curveVal < 0.6 and curveVal > -0.6: curveVal = 0.123
13    elif curveVal < 0.7 and curveVal > -0.7: curveVal = 0.263
14    elif curveVal < 0.8 and curveVal > -0.8: curveVal = 0.263
15    elif curveVal < 0.9 and curveVal > -0.9: curveVal = 0.263
16    elif curveVal < 1.0 and curveVal > -1.0: curveVal = 0.263
17    elif curveVal < 1.2 and curveVal > -1.2: curveVal = 0.525
18    elif curveVal < 1.5 and curveVal > -1.5: curveVal = 0.613
19    else: curveVal = 0.349
20
21    if NegativeCurve == True:
22        cmd_vel.angular.z = -curveVal
23    else:
24        cmd_vel.angular.z = curveVal
25    self.cmd_vel_pub.publish(cmd_vel)
```

2.3 Mô phỏng

Thế giới dùng trong mô phỏng với làn đường và các biển báo



Hình 2.5: *Thế giới dùng trong mô phỏng*

Chương 3

Nhận diện biến báo

3.1 Xây dựng model CNN hỗ trợ nhận diện biến báo

3.1.1 CNN model

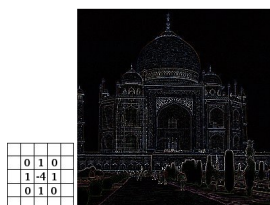
CNN là gì ?

CNN - Convolutional Neural Network là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh.

Convolutional là gì?

Convolutional là một cửa sổ trượt (Sliding Windows) trên một ma trận.

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.



Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5x5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3 .

Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhân ma trận Filter với ma trận ảnh 5×5 bên trái.

Cấu trúc mạng của CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

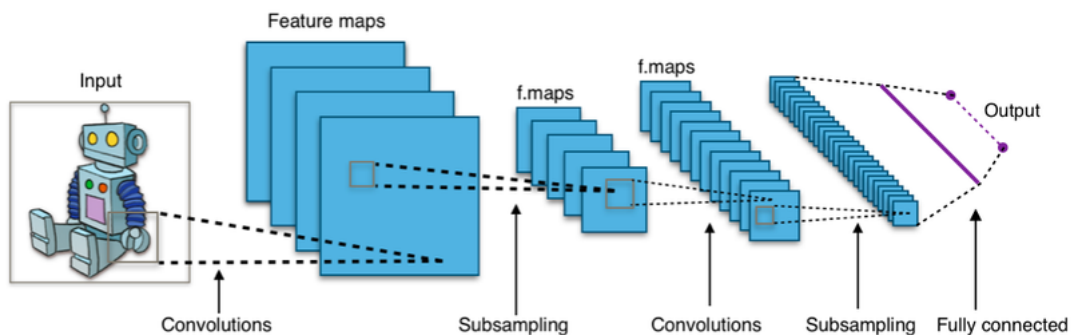
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

3.1.2 Keras

Keras là một open source cho Neural Network được viết bởi ngôn ngữ Python. Nó là một library được phát triển vào năm 2015 bởi Francois Chollet, là một kỹ sư nghiên cứu Deep Learning. Keras có thể sử dụng chung với các thư viện nổi tiếng như Tensorflow, CNTK, Theano. Một số ưu điểm của Keras như:

- Dễ sử dụng, dùng đơn giản hơn Tensor, xây dựng model nhanh.
- Run được trên cả CPU và GPU.
- Hỗ trợ xây dựng CNN, RNN hoặc cả hai. Với những người mới tiếp cận đến Deep như mình thì mình chọn sử dụng Keras để build model vì nó đơn giản, dễ nắm bắt hơn các thư viện khác.

3.1.3 Xây dựng model CNN

[3] Trong Keras có hỗ trợ 2 cách dựng models là Sequential model và Function API. Ở đây chúng ta sẽ sử dụng Sequential model.

Trong dự án này chúng ta sẽ xây dựng model CNN hỗ trợ cho việc nhận diện 4 loại biển báo
Công cụ sử dụng



Import library

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from keras.models import Sequential
4 from keras.layers import Dense
5 from keras.optimizers import Adam
6 from keras.utils.np_utils import to_categorical
7 from keras.layers import Dropout, Flatten
8 from keras.layers.convolutional import Conv2D, MaxPooling2D
9 import cv2
10 from sklearn.model_selection import train_test_split
11 import pickle
12 import os
13 import pandas as pd
14 import random
15 from keras.preprocessing.image import ImageDataGenerator
```

Khai báo biến

```
1 path = 'myData' # folder with all the class folders
2 labelFile = 'labels.csv' #file with all class folders
3 batch_size_val= 50 # file with all name of classes
```

```

4 steps_per_epoch_val=2000
5 epochs_val=10
6 imageDimesions = (32,32,3)
7 testRatio = 0.2 #if 1000 image 20% of remaining 800 will be 200
  for validation
8 validationRatio = 0.2

```

```

1 count = 0
2 images = []
3 classNo = []
4 myList = os.listdir(path)
5 print("Total Classes Detected:",len(myList))
6 noOfClasses=len(myList)
7 print("Importing Classes.....")
8 for x in range (0,len(myList)):
9     myPicList = os.listdir(path+"/"+str(count))
10    for y in myPicList:
11        curImg = cv2.imread(path+"/"+str(count)+"/"+y)
12        images.append(curImg)
13        classNo.append(count)
14    print(count,end = " ")
15    count +=1
16 print(" ")
17 images = np.array(images)
18 classNo = np.array(classNo)

```

```

Total Classes Detected: 4
Importing Classes.....
0 1 2 3

```

```

1 X_train, X_test, y_train, y_test = train_test_split(images,
  classNo,test_size=testRatio)
2 X_train, X_validation, y_train, y_validation = train_test_split
  (X_train,y_train,test_size=validationRatio)

```

```

1 print("Data Shapes")
2 print("Train",end = "");print(X_train.shape,y_train.shape)
3 print("Validation",end = "");print(X_validation.shape,
  y_validation.shape)
4 print("Test",end="");print(X_test.shape,y_test.shape)
5 assert (X_train.shape[0]==y_train.shape[0]),"The number of
  image in not equal to the number of label in training set"
6 assert(X_validation.shape[0]==y_validation.shape[0]), "The
  number of images in not equal to the number of lables in
  validation set"

```

```

7 assert(X_test.shape[0]==y_test.shape[0]), "The number of images
  in not equal to the number of lables in test set"
8 assert(X_train.shape[1:]==(imageDimesions)), " The dimesions of
  the Training images are wrong "
9 assert(X_validation.shape[1:]==(imageDimesions)), " The
  dimesionas of the Validation images are wrong "
10 assert(X_test.shape[1:]==(imageDimesions)), " The dimesionas of
  the Test images are wrong"

```

Data Shapes

```

Train(1746, 32, 32, 3) (1746,)
Validation(437, 32, 32, 3) (437,)
Test(546, 32, 32, 3) (546,)

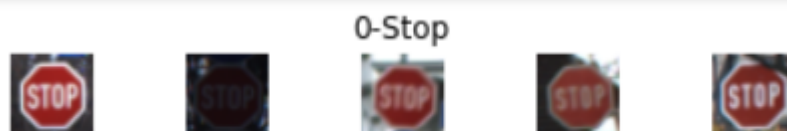
```

```

1 data=pd.read_csv(labelFile)
2 print("data shape ",data.shape,type(data))

1 num_of_samples = []
2 cols = 5
3 num_classes = noOfClasses
4 fig, axs = plt.subplots(nrows=num_classes,ncols=cols,figsize
  =(5,300))
5 fig.tight_layout()
6 for i in range(cols):
7     for j,row in data.iterrows():
8         x_selected = X_train[y_train == j]
9         axs[j][i].imshow(x_selected[random.randint(0,len(
  x_selected)-1),:,:],cmap=plt.get_cmap("gray"))
10        axs [j][i].axis("off")
11        if i == 2:
12            axs [j][i].set_title(str(j)+ "-" +row["Name"])
13            num_of_samples.append(len(x_selected))

```

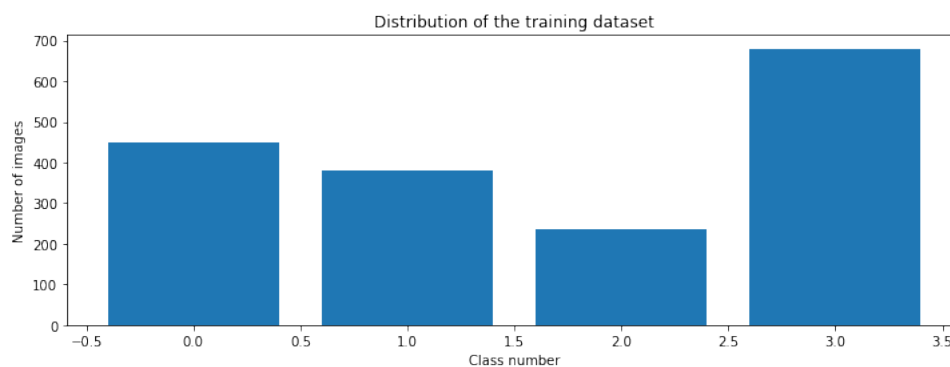




```

1 print(num_of_samples)
2 plt.figure(figsize=(12, 4))
3 plt.bar(range(0, num_classes), num_of_samples)
4 plt.title("Distribution of the training dataset")
5 plt.xlabel("Class number")
6 plt.ylabel("Number of images")
7 plt.show()

```



```

1 def grayscale(img):
2     img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
3     return img
4
5
6 def equalize(img):
7     img = cv2.equalizeHist(img)
8     return img

```

```

9
10
11 def preprocessing(img):
12     img = grayscale(img) # CONVERT TO GRAYSCALE
13     img = equalize(img) # STANDARDIZE THE LIGHTING IN AN IMAGE
14     img = img / 255 # TO NORMALIZE VALUES BETWEEN 0 AND 1
15     # INSTEAD OF 0 TO 255
16     return img
17
18 X_train = np.array(list(map(preprocessing, X_train))) # TO
19 # IRETATE AND PREPROCESS ALL IMAGES
20 X_validation = np.array(list(map(preprocessing, X_validation)))
21 X_test = np.array(list(map(preprocessing, X_test)))
22 cv2.imshow("GrayScale Images",X_train[random.randint(0,len(
23     X_train)-1)]) # TO CHECK IF THE TRAINING IS DONE PROPERLY
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
26
```

```

9 y_train = to_categorical(y_train, noOfClasses)
10 y_validation = to_categorical(y_validation, noOfClasses)
11 y_test = to_categorical(y_test, noOfClasses)

```

```

1 def myModel():
2     no_Of_Filters = 60
3     size_of_Filter = (5, 5) # THIS IS THE KERNEL THAT MOVE
    AROUND THE IMAGE TO GET THE FEATURES.
4     # THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN USING 32
    32 IMAGE
5     size_of_Filter2 = (3, 3)
6     size_of_pool = (2, 2) # SCALE DOWN ALL FEATURE MAP TO
    GERNALIZE MORE, TO REDUCE OVERFITTING
7     no_Of_Nodes = 500 # NO. OF NODES IN HIDDEN LAYERS
8     model = Sequential()
9     model.add((Conv2D(no_Of_Filters, size_of_Filter,
    input_shape=(imageDimesions[0],imageDimesions[1],1),
    activation='relu')) # ADDING MORE CONVOLUTION LAYERS =
    LESS FEATURES BUT CAN CAUSE ACCURACY TO INCREASE
10    model.add((Conv2D(no_Of_Filters, size_of_Filter, activation
    ='relu'))))
11    model.add(MaxPooling2D(pool_size=size_of_pool)) # DOES NOT
    EFFECT THE DEPTH/NO OF FILTERS
12
13    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2,
    activation='relu'))))
14    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2,
    activation='relu'))))
15    model.add(MaxPooling2D(pool_size=size_of_pool))
16    model.add(Dropout(0.5))
17
18    model.add(Flatten())
19    model.add(Dense(no_Of_Nodes, activation='relu'))
20    model.add(Dropout(0.5)) # INPUTS NODES TO DROP WITH EACH
    UPDATE 1 ALL 0 NONE
21    model.add(Dense(noOfClasses, activation='softmax')) #
    OUTPUT LAYER
22    # COMPILE MODEL
23    model.compile(Adam(lr=0.001), loss='
    categorical_crossentropy', metrics=['accuracy'])
24    return model

```

```

1 model = myModel()
2 print(model.summary())
3 history = model.fit_generator(dataGen.flow(X_train, y_train,
    batch_size=batch_size_val),steps_per_epoch=
    steps_per_epoch_val, epochs=epochs_val,validation_data=(
    X_validation, y_validation), shuffle=1)

```

4 ##### PLOT

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 28, 28, 60)	1560
conv2d_6 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_7 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_8 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 30)	0
dropout_3 (Dropout)	(None, 4, 4, 30)	0
flatten_2 (Flatten)	(None, 480)	0
dense_3 (Dense)	(None, 500)	240500
dropout_4 (Dropout)	(None, 500)	0
dense_4 (Dense)	(None, 4)	2004
Total params: 358,484		
Trainable params: 358,484		
Non-trainable params: 0		

```
None
Epoch 1/10
2000/2000 [=====] - 395s 198ms/step - loss: 0.0778 - accuracy: 0.9717 - val_loss: 2.7207e-05 - val_acc
uracy: 1.0000
Epoch 2/10
2000/2000 [=====] - 381s 190ms/step - loss: 0.0100 - accuracy: 0.9970 - val_loss: 9.0834e-06 - val_acc
uracy: 1.0000
Epoch 3/10
2000/2000 [=====] - 372s 186ms/step - loss: 0.0067 - accuracy: 0.9980 - val_loss: 0.0021 - val_accu
racy: 1.0000
Epoch 4/10
2000/2000 [=====] - 373s 187ms/step - loss: 0.0058 - accuracy: 0.9984 - val_loss: 2.9897e-07 - val_acc
uracy: 1.0000
Epoch 5/10
2000/2000 [=====] - 372s 186ms/step - loss: 0.0051 - accuracy: 0.9986 - val_loss: 1.3367e-08 - val_acc
uracy: 1.0000
Epoch 6/10
2000/2000 [=====] - 379s 190ms/step - loss: 0.0044 - accuracy: 0.9988 - val_loss: 2.7279e-10 - val_acc
uracy: 1.0000
Epoch 7/10
2000/2000 [=====] - 374s 187ms/step - loss: 0.0044 - accuracy: 0.9988 - val_loss: 0.0000e+00 - val_acc
uracy: 1.0000
Epoch 8/10
2000/2000 [=====] - 371s 186ms/step - loss: 0.0055 - accuracy: 0.9987 - val_loss: 0.0000e+00 - val_acc
uracy: 1.0000
Epoch 9/10
2000/2000 [=====] - 376s 188ms/step - loss: 0.0040 - accuracy: 0.9992 - val_loss: 0.0000e+00 - val_acc
uracy: 1.0000
Epoch 10/10
2000/2000 [=====] - 380s 190ms/step - loss: 0.0050 - accuracy: 0.9988 - val_loss: 2.7279e-10 - val_acc
uracy: 1.0000
```

```
1 plt.figure(1)
2 plt.plot(history.history['loss'])
3 plt.plot(history.history['val_loss'])
4 plt.legend(['training', 'validation'])
5 plt.title('loss')
6 plt.xlabel('epoch')
7 plt.figure(2)
8 plt.plot(history.history['accuracy'])
9 plt.plot(history.history['val_accuracy'])
10 plt.legend(['training', 'validation'])
11 plt.title('Accuracy')
12 plt.xlabel('epoch')
13 plt.show()
14 score = model.evaluate(X_test, y_test, verbose=0)
15 print('Test Score:', score[0])
16 print('Test Accuracy:', score[1])
```

Lưu model dưới dạng file.

```
1 model.save("my_model")
2 model.save_weights("weights.h5")
```

3.1.4 Nhận diện bằng openCV

```
1 import cv2
2 import numpy as np
3 from scipy.stats import itemfreq
4 import pickle
```

Load model.

```
1 #####33
2
3 # #####3
4 font = cv2.FONT_HERSHEY_SIMPLEX
5 from tensorflow import keras
6 my_model = keras.models.load_model("my_model")
7 my_model.load_weights("weights.h5")
```

Chuyển ảnh về dạng COLOR_BGRGRAY để tăng khả năng xử lý

```
1 def grayscale(img):
2     img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
3     return img
```

Cân bằng sáng. Tham khảo giải thuật cân bằng sáng của opencv. // link

```
1 def equalize(img):
2     img = cv2.equalizeHist(img)
3     return img
```

Chuyển các giá trị pixel của ảnh nằm trong khoảng từ 0 đến 1 thay vì 0 đến 255.

Đối với hầu hết dữ liệu hình ảnh, giá trị pixel là số nguyên có giá trị từ 0 đến 255.

Mạng nơ-ron xử lý đầu vào bằng cách sử dụng các giá trị trọng số nhỏ và đầu vào có giá trị số nguyên lớn có thể làm gián đoạn hoặc làm chậm quá trình học. Do đó, bạn nên chuẩn hóa các giá trị pixel để mỗi giá trị pixel có giá trị từ 0 đến 1.

Nó hợp lệ cho hình ảnh có giá trị pixel trong phạm vi 0-1 và hình ảnh có thể được xem bình thường.

```
1 def preprocessing(img):
2     img = grayscale(img)
3     img = equalize(img)
4     img = img/255
5     return img
```

Tạo hệ thống class name

```
1 def getClassName(classNo):
2     if classNo == 0: return 'Stop'
3     elif classNo == 1: return 'Right'
4     elif classNo == 2: return 'Left'
5     elif classNo == 3: return 'ahead only'
6
7 #####33
```

```

1 clicked = False
2 def onMouse(event, x, y, flags, param):
3     global clicked
4     if event == cv2.EVENT_LBUTTONDOWN:
5         clicked = True

```

Lấy ảnh từ camera và xử lý.

Để tăng độ chính xác chung ta có sử dụng thêm 1 số chức năng của opencv: Sử dụng cv2.HoughCircles để lọc hình tròn trong ảnh. Và chỉ lọc lấy hình tròn lớn nhất.

Rồi sử dụng model.predict(image) để nhận diện.

```

1 cameraCapture = cv2.VideoCapture(0)
2 cv2.namedWindow('camera')
3 cv2.setMouseCallback('camera', onMouse)
4
5 success, frame = cameraCapture.read()
6
7 while success and not clicked:
8     cv2.waitKey(1)
9     success, frame = cameraCapture.read()
10
11     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
12     img = cv2.medianBlur(gray, 37)
13     circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT,
14                               1, 50, param1=120, param2=40)
15
16     if not circles is None:
17         circles = np.uint16(np.around(circles))
18         max_r, max_i = 0, 0
19         for i in range(len(circles[:, :, 2][0])):
20             if circles[:, :, 2][0][i] > 50 and circles[:, :,
21             2][0][i] > max_r:
22                 max_i = i
23                 max_r = circles[:, :, 2][0][i]
24
25         x, y, r = circles[:, :, :][0][max_i]
26         if y > r and x > r:
27             square = frame[y-r:y+r, x-r:x+r]
28             img = np.asarray(square)
29             img = cv2.resize(img, (32,32))
30             img = preprocessing(img)
31             cv2.imshow("Processed Image",img)
32             img = img.reshape(1,32,32,1)
33             cv2.putText(square, "TRAFFIC SIGN: ",(20,35),font
34             ,0.75, (0,0,255),2,cv2.LINE_AA)
35             cv2.putText(square, "PROBABILITY: ",(20,75),font
36             ,0.75, (0,0,255),2,cv2.LINE_AA)
37             # PREDICT IMAGE
38             predictions = my_model.predict(img)

```

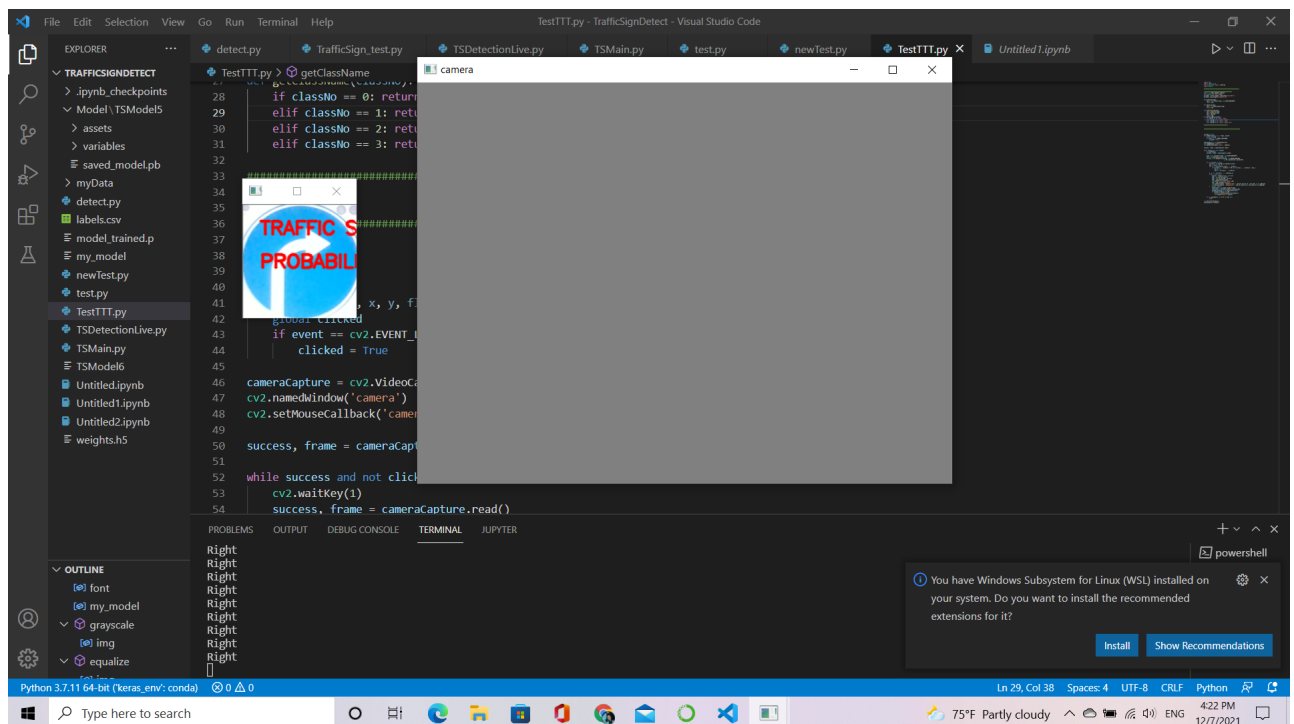


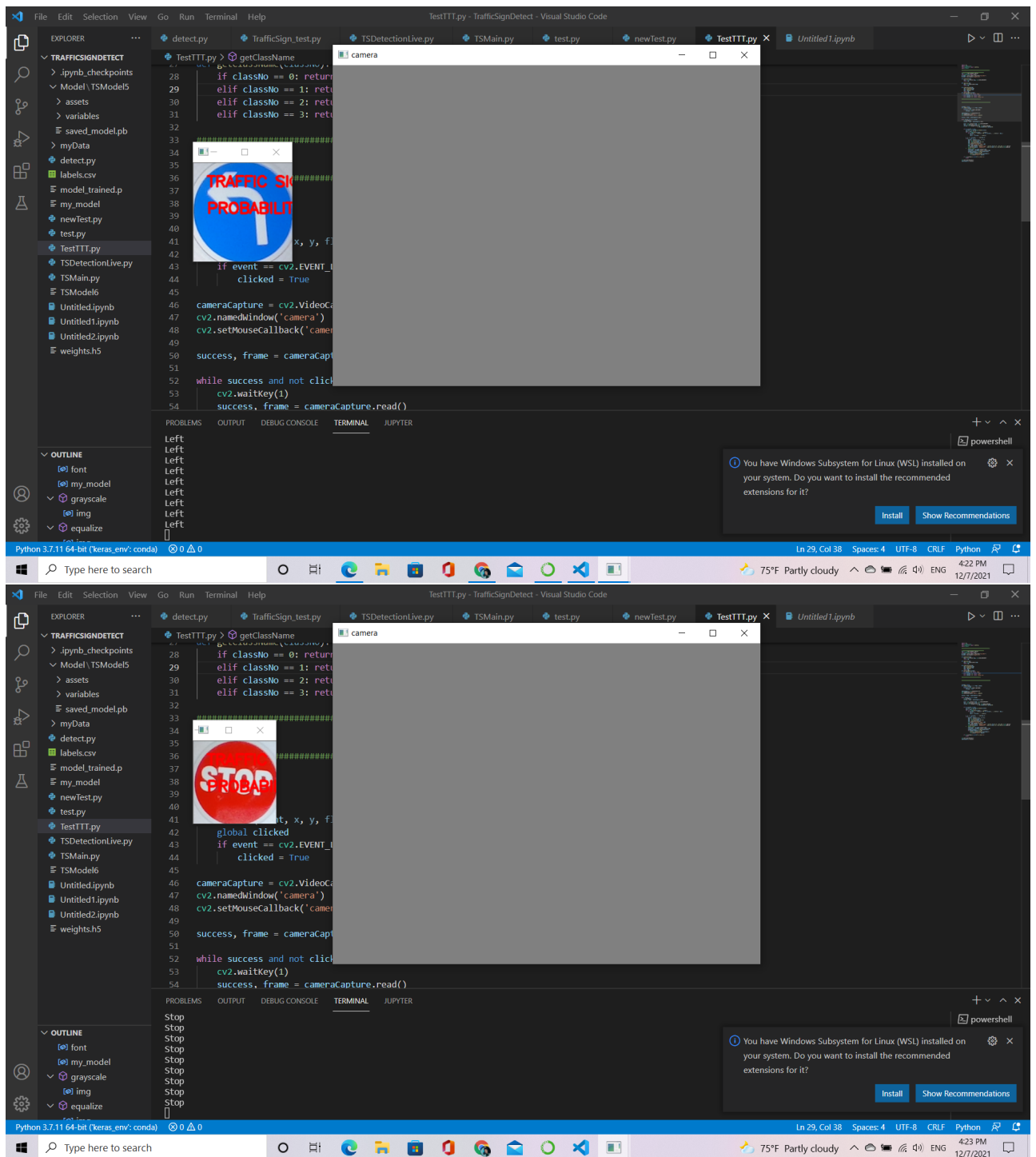
```

36         classIndex = my_model.predict_classes(img)
37         probabilityValue = np.amax(predictions)
38         if probabilityValue > 0.99:
39             print(str(getClassName(classIndex)))
40             cv2.imshow("Result", square)
41
42         if cv2.waitKey(1) and 0xFF == ord('q'):
43             break
44
45 cv2.destroyAllWindows()
46 cameraCapture.release()

```

Demo trên webcam laptop.





Chương 4

Kết luận và hướng phát triển

4.1 Đánh giá kết quả thực hiện

- Đã tìm hiểu, phân tích và xây dựng mô hình của robot; đánh giá được kết quả xây dựng thông qua mô phỏng trên Gazebo.
- Cơ bản hiểu lý thuyết và ứng dụng được thư viện xử lý ảnh OpenCV cho Python.

4.2 Những mặt hạn chế

- Robot di chuyển chưa thật sự tối ưu ở các góc rẽ
- Việc tích hợp phần nhận diện biển báo vào bộ điều khiển robot còn gặp nhiều khó khăn

4.3 Hướng phát triển

- Tối ưu hóa việc di chuyển của robot
- Hoàn thiện phần nhận diện biển báo và đưa ra quyết định hướng di chuyển của robot
- Kiểm tra kết quả mô phỏng trên robot thật, đáp ứng yêu cầu đề ra của đề tài

Tài liệu tham khảo

- [1] Vighnesh Devane, Ganesh Sahane, Hritish Khairmode, and Gaurav Datkhile. Lane detection techniques using image processing. In *ITM Web of Conferences*, volume 40, pages 1–4, Jan. 2021.
- [2] M.Hassan. Self-driving car using raspberry pi. In *computer-vision.zone*, 2020.
- [3] M.Hassan. Traffic signs classification using convolution neural networks cnn. In *computer-vision.zone*, February 2020.