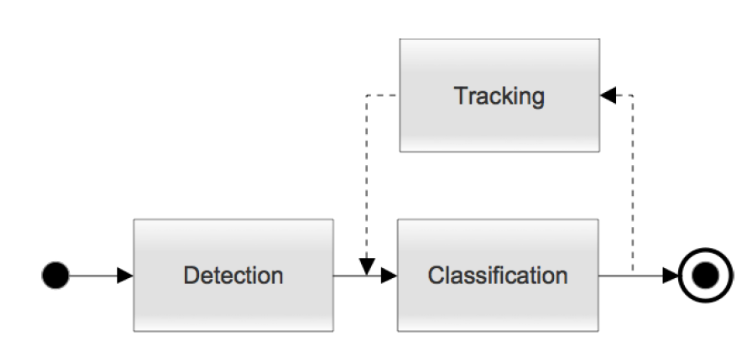
1. **Đề bài:**

Tìm hiểu bài toán nhận dạng đèn giao thông.

1. **Giới thiệu cách tiếp cận:**

Bài toán nhận diện đèn giao thông là bài toán có khả năng phân tích hình ảnh, nhận diện được đèn giao thông và màu sắc của đèn. Nguồn hình ảnh nhóm chúng em tự thu thập ngoài thực tế. Có nhiều cách khác nhau để thu thập dữ liệu hình ảnh. Chúng em lựa chọn cách thu dữ liệu giới hạn vùng nhìn và chọn địa điểm thu là tại Hà Nội. Đó là các dữ liệu thu thập giới hạn trong vùng có đèn giao thông (lấy đèn giao thông là đối tượng chính của ảnh).

Input đầu vào sẽ là một video có xuất hiện đèn giao thông sau đó thực hiện các bước xử lý ảnh để phát hiện đèn rồi đưa vào học máy để phân loại(đèn xanh, đỏ, vàng).



Bài toán sẽ được giải quyết theo 4 bước:

Bước 1: Thu dữ liệu.

Bước 2: Xác định/phát hiện đối tượng.

Bước 3: Đào tạo phân loại.

Bước 4: Demo với tập video thu được.

* 1. ***. Thu dữ liệu:***

Nhóm em tiến hành thu dữ liệu theo 2 hướng: để đào tạo phân loại sử dụng học sâu và để demo đối tượng.

Để đào tạo phân loại sử dụng học sâu, ta cần 3 nhãn (labels) là các ảnh của 3 màu đèn giao thông (xanh, đỏ và vàng). Số lượng thu dự tính 300 ảnh mỗi nhãn.

Quay một video dài ít nhất 1 phút có đèn giao thông để tiến hành phát hiện đối tượng.

* 1. ***. Detect object ( Phát hiện đối tượng, cụ thể là đèn giao thông):***

Với đầu vào là một video ta tiến hành cắt video thành các ảnh (frame) để tiến hành detect. Frame được cắt từ video là cảnh đô thị có rất nhiều đối tượng đang diễn ra: có xe hơi, người đi bộ, biển báo giao thông, tòa nhà và một loạt thông tin cần được xử lý với tốc độ nhanh chóng. Nên việc phát hiện giới hạn lượng thông tin cần được phân tích(đèn giao thông) là rất quan trọng .

Dựa trên nghiên cứu của trường Đại học Kettering là một trong tám trường đại học trên thế giới được chọn tham gia [thử thách](https://www.kettering.edu/autodrive) tự động của Society of Automotive Engineers’ (SAE) - một cuộc thi xe tự trị. Mục tiêu của cuộc thi là điều hướng một khóa học lái xe đô thị tự chủ trong vòng ba năm. Sự ra đời của cuộc thi đã mang đến một loạt các nỗ lực nghiên cứu trong việc lái xe tự trị.

*Phương pháp phát hiện:*

*2.2.1. Cắt bỏ phần dưới của hình ảnh*

Vì đèn giao thông đều ở cao, đa số ở phần trên của ảnh nên ta chỉ cần xét phần nửa trên của ảnh để tìm đèn*.* Điều này sẽ giới hạn phạm vi tìm và đồng thời cũng giảm nhiễu. *2.2.2. Chuyển đổi hình ảnh sang màu xám*

Hình ảnh gốc được thể hiện trong không gian màu RGB. Tuy nhiên, RGB trộn lẫn thông tin màu sắc và cường độ trên khắp các kênh của nó. Điều này làm cho định dạng RGB nhạy cảm với những thay đổi trong ánh sáng. Mục tiêu phần này là phát hiện đèn giao thông ( chưa quan tâm đến màu sắc đèn, chỉ quan tâm đến độ sáng của đối tượng trong ảnh ) nên việc chuyển ảnh màu sang ảnh xám sẽ cho hiệu quả hơn. *2.2.3. Áp dụng hình thái mũ trắng trên đầu (a white top hat)*

Là sự khác biệt giữa ảnh đầu vào và phép mở ảnh ( co ảnh rồi giãn ảnh)

Phép co ảnh và giãn ảnh đều là phép toán cơ bản trong hình thái học. Phép giãn có tác dụng làm cho đối tượng ban đầu trong ảnh tăng lên về kích thước. Ngược lại phép co ảnh làm cho đối tượng ban đầu giảm đi về kích thước.

*2.2.4. Chọn các điểm sáng làm điểm đánh dấu*

Đánh dấu tất cả các nhãn ( điểm sáng) theo số nguyên khác nhau.  
*2.2.5. Áp dụng thuật toán tăng trưởng vùng (watershed)*

là một thuật toán phát triển vùng phân đoạn một hình ảnh dựa trên một tập hợp các điểm đánh dấu ban đầu . (Tìm các đối tượng rồi khoanh vùng cho nó). *2.2.6. Chọn các điểm sáng không phải là một phần của một vật thể lớn hơn*

* 1. ***Classification (Phân loại, sử dụng học máy):***

*2.3.1. Resize ảnh đầu vào*

Các ảnh lúc đầu vào có kích thước khác nhau do đó các ảnh cần được resize trước khi training. Tất cả các ảnh dữ liệu đầu vào được resize về kích thước 90x90.

*2.3.2. Chuẩn hóa giá trị điểm ảnh:*

Ảnh dữ liệu đầu vào có phạm vi giá trị điểm ảnh là các số nguyên từ 0 đến 255. Một cách tốt để thực hiện bài toán này là chuẩn hóa giá trị điểm ảnh, cụ thể ở đây sẽ đưa các giá trị về trong đoạn [0,1]. Trước tiên sẽ chuyển giá trị các điểm ảnh từ kiểu integer về float, sau đó chia các giá trị điểm ảnh cho giá trị max là 255.0.

Theo các biểu diễn về mức xám, giá trị mức xám thấp cho cảm giác là điểm ảnh tối, giá trị mức xám cao đem lại cảm giác sáng.

*2.3.3. Hướng xây dựng mô hình nhận diện và phân loại:*

Với tập dữ liệu ảnh đầu vào, sau các bước chuẩn hóa như trên, ảnh sẽ chia ra làm 2 bộ dữ liệu: train và test, kích thước bộ test là 20% tập dữ liệu tổng, phần còn lại là bộ train. Cả hai bộ đều có tập nhãn đi kèm tương ứng.

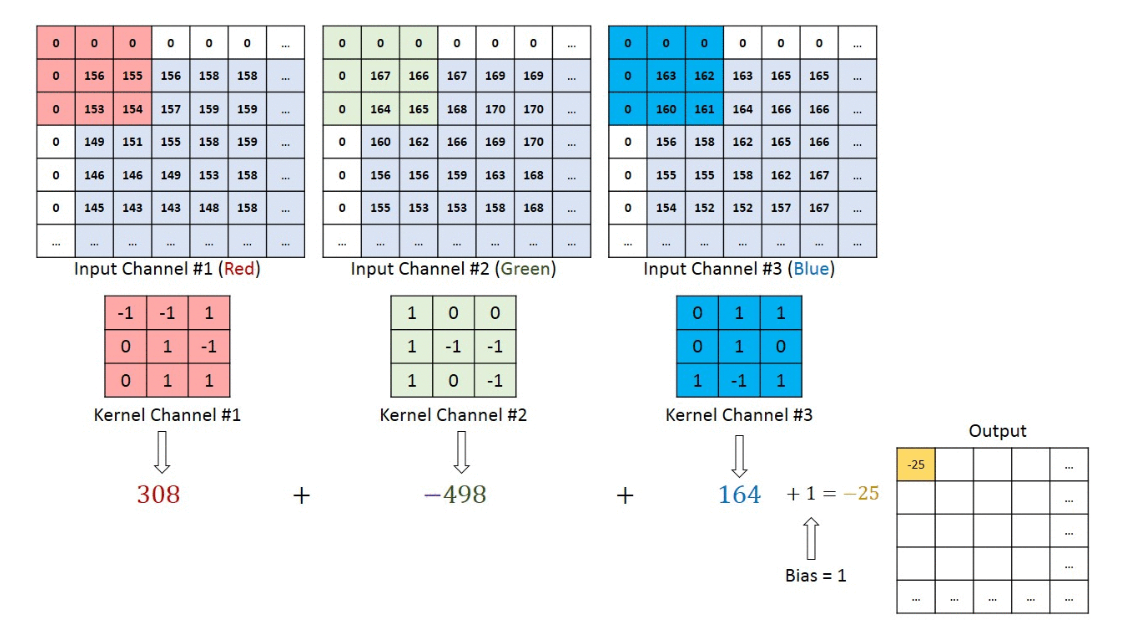
Để thực hiện bài toán nhận diện được đèn giao thông và màu sắc của nó, nhóm chúng em sử dụng mạng nơ ron tích chập (Convolutional Neural Network – CNN).

*2.3.4: Thuật toán xây dựng mô hình – CNN:*

Sau khi chuẩn bị xong dữ liệu ảnh và nhãn tương ứng, chúng em xây dựng mô hình CNN.

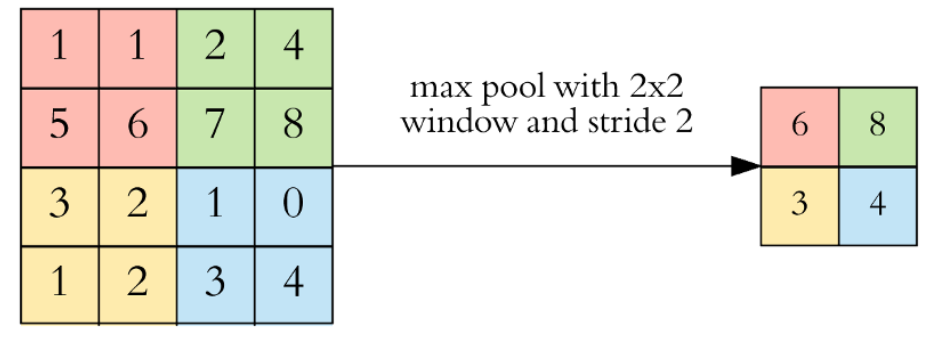
Mô hình CNN dùng để tạo model sẽ có nhiều lớp ẩn chồng lên nhau: Lớp tích chập (convolution layer) với các bộ lọc, lớp gộp pooling, lớp kết nối đầy đủ fully connected. Và sử dụng các hàm kích hoạt các trọng số: ReLU, Softmax để phân lớp đối tượng bằng cách tính xác suất thuộc class nào của đối tượng đó. Cụ thể:

***Lớp tích chập*** (convolution layer): sẽ dùng để trích xuất các đặc trưng (feature) từ ảnh đầu vào bằng cách sử dụng các bộ lọc. Các bộ lọc này được nhân tích chập với ảnh đầu vào để xác định được đặc trưng của ảnh. Các tham số trong bộ lọc sẽ được học dần dần để tự điều chỉnh lấy ra những thông tin chính xác nhất từ ảnh đầu vào. Các trọng số sẽ được chia sẻ trong toàn bộ ảnh đầu vào. Tức là với 1 filter, nó sẽ được dịch chuyển trong toàn bộ bức ảnh ở các vị trí khác nhau. Số lượng tham số cho mỗi filter kích thước **H** x **W** bằng **H** x **W** + **1**. (1 là hệ số bias). Một lớp tích chập sẽ có nhiều filter, số lượng feature đầu ra của lớp này bằng số filter sử dụng của lớp.



*Hình ví dụ về lớp Conv 3D*

***Lớp pooling***: Dùng để chắt lọc các thông tin hữu ích hơn (loại bỏ thông tin thừa), giúp đơn giản hóa thông tin đầu ra, giảm bớt số lượng neural. Cụ thể trong bài này chúng em sử dụng lớp poopling phổ biến là max-pooling kích thước 2x2. Nó chọn giá trị lớn nhất trong từng vùng kích thước 2x2 của ảnh đầu vào của lớp tức lấy ra đặc trưng tốt nhất của vùng ảnh đó. Input của pooling layer có kích thước **H**x**W**x**D**, ta tách ra làm D ma trận kích thước **H**x**W**. Với mỗi ma trận, trên vùng kích thước 2x2 trên ma trận ta tìm giá trị lớn nhất để lấy ra. Như vậy qua lớp max-pooling này, số lượng neural giảm đi một nửa. Và lớp này thường đặt ngay sau lớp tích chập.

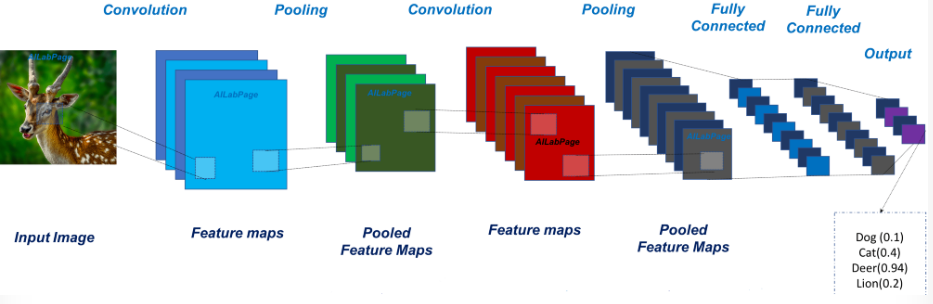


*Ví dụ về max pooling*

***Lớp Dropout***: Lớp này có ý nghĩa là bỏ qua một vài node trong mạng kết nối. Với mỗi lớp ẩn, mỗi mẫu train, chúng ta sẽ bỏ bớt đi một số node của lớp và bỏ qua nó (cũng không áp dụng hàm kích hoạt cho nó). Lớp này giúp giảm overfitting.

***Lớp Flatten***: Từ ảnh đầu vào, qua các lớp ẩn của mô hình, mô hình đã học được tương đối các đặc trưng của ảnh, thì output của layer cuối cùng với kích thước h x w x d sẽ được chuyển về một vector kích thước h x w x d, phục vụ cho việc thực hiện trong lớp fully connected phía sau.

***Lớp kết nối đầy đủ*** (fully connected): Dùng để kết hợp các đặc điểm của ảnh để ra được output của model. Các node của layer cuối sẽ lưu trữ giá trị xác suất khả năng thuộc class nào của ảnh đầu vào. Từ đó phân loại được ảnh đầu vào.



*Ví dụ về phân loại hình ảnh bằng CNN model*

***2.4. Tiến hành phân loại từ video:***

Đọc vào video đã quay được, tiến hành cắt video được ảnh (frame) đưa qua Object dection để tìm, phát hiện đối tượng có khả năng cao là đèn giao thông. Sau đó đưa vào model đã train được bên trên để dự đoán ( predict) kết quả phân loại (classification).

1. **Thực hiện:**

Với bài toán này, nhóm chúng em sử dụng ngôn ngữ lập Python để thực hiện, việc xây dựng mô hình sử dụng chủ yếu bằng công cụ có sẵn trong Keras và xử lý ảnh với open cv.

***3.1. Thu dữ liệu***

Tự thu thập dữ liệu thực tế bằng điện thoại di động. Sau đó các ảnh được chia ra làm 3 nhãn ( labels) với 3 class cần phân loại: Xanh (0), Đỏ (1), Vàng (2).

Các ảnh thu được sẽ chia ra 2 phần: train và test trong quá trình train. Phần này mô hình được nhìn nhãn của chúng.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 2 | 0 |

Sử dụng video làm ảnh dự đoán sau khi hoàn thiện mô hình với tập dữ liệu ở trên. Phần này mô hình sẽ không được nhìn thấy nhãn. Nó sẽ phải dự đoán phân lớp cho các đèn giao thông có trong video này.

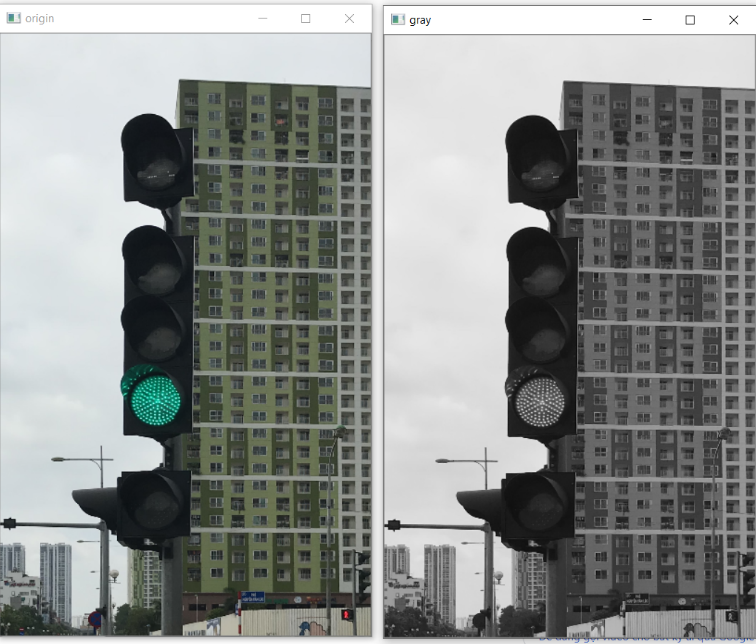
***3.2. Detect object:***

*3.2.1. Cắt bỏ phần dưới của hình ảnh*

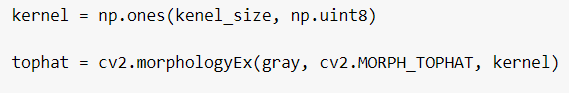
**

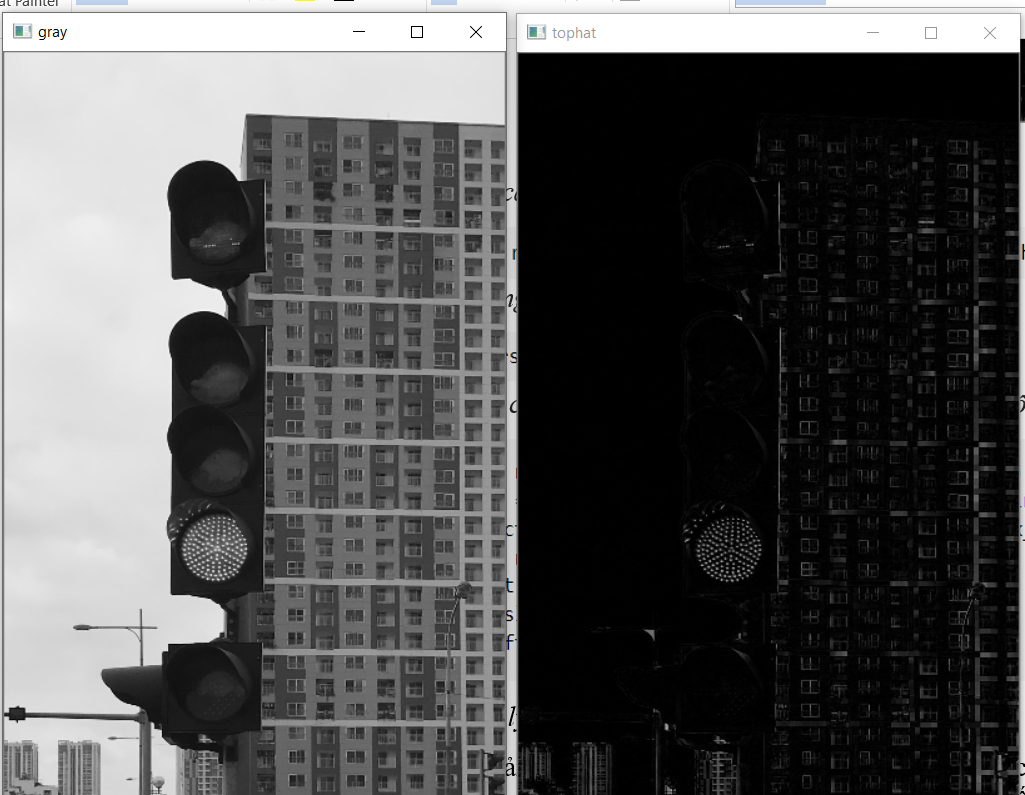
*3.2.2. Chuyển đổi hình ảnh sang màu xám*

**

**

*3.2.3. Áp dụng hình thái mũ trắng trên đầu (a white top hat)*

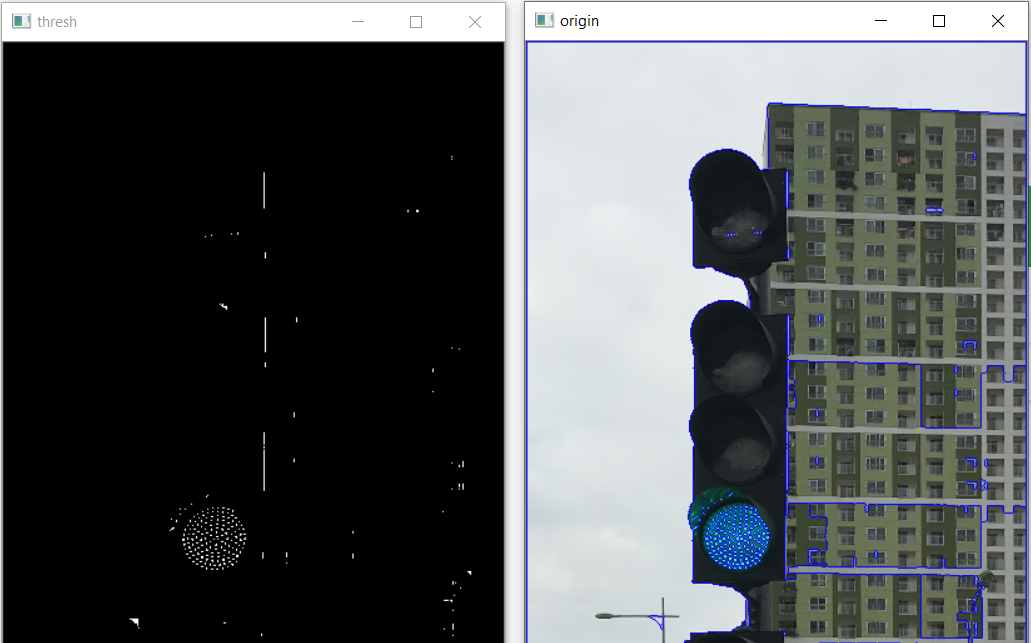
**

**

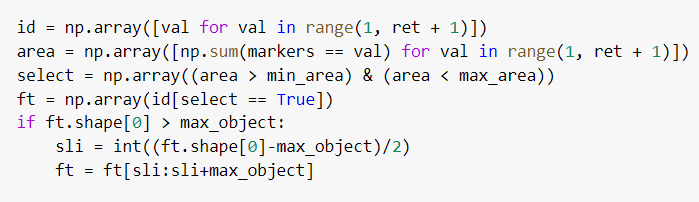
*3.2.4. Chọn các điểm sáng làm điểm đánh dấu*

*  
3.2.5. Áp dụng thuật toán tăng trưởng vùng (watershed)*

**

**

*3.2.6.*  *Chọn các điểm sáng không phải là một phần của một vật thể lớn hơn*

******

* 1. ***. Classification (Phân loại, sử dụng học máy):***

*3.3.1: Tiền xử lý dữ liệu ảnh*

Các ảnh được resize về kích thước 90x90. Sau đó chuẩn hóa các điểm ảnh để đổi giá trị điểm ảnh từ 0 đến 255 về giá trị từ 0 đến 1.

Hàm để resize và chuẩn hóa trong OpenCV:

|  |
| --- |
|  |

Thực hiện các hàm chuyển đổi trên lần lượt các ảnh trong file dữ liệu đầu vào. Đọc ảnh từ đường dẫn đến nơi chứa dữ liệu, tạo 2 list là data[] để lưu dữ liệu và label[] để lưu nhãn dữ liệu. Đưa class biểu diễn dưới dạng nhị phân.

Với mỗi file ảnh trong folder dữ liệu sẽ đều thực hiện resize và chuẩn hóa điểm ảnh. Mỗi ảnh lần lượt được cho vào list data[], mỗi nhãn tương ứng của ảnh được cho vào list label[].

Phân chia tập dữ liệu thành 2 phần: train và test. Với phần train là 80% và phần test 20% tập dữ liệu.

|  |
| --- |
|  |

*3.3.2: Xây dựng model:*

Như đã nêu ở mục 2.3.5, bài thoán sử dụng CNN để xây dựng mô hình. Mô hình được xây dựng với các lớp đã nêu, sử dụng thuật tối ưu SGD, hàm loss 'categorical\_crossentropy’ để đo sự sai khác của nhãn ảnh đầu ra dự đoán và nhãn thực tế. Để theo dõi tính chính xác của mô hình, khi training sẽ hiển thị thêm về số liệu đo độ chính xác “accuracy”

|  |
| --- |
|  |

Mô tả:

Lớp 1: Là lớp tích chập Conv2D sử dụng 32 bộ lọc với kích thước 7x7 sử dụng trích xuất đặc trưng của ảnh đầu vào. Ở lớp này sử dụng padding = “SAME” để cho tất cả các điểm ảnh trong ảnh ban đầu đều được đi qua bộ lọc, thu được feature map có kích thước không đổi so với ban đầu. Sau khi thực hiện lớp convolutional layer, hàm kích hoạt là ReLU với công thức f(x) = max(0, x) được sử dụng.



Lớp 2: Là lớp pooling MaxPooling2D được áp dụng vào output của lớp 1. Lớp này giúp lấy ra các điểm ảnh Max (tiêu biểu) trong một phạm vi, giúp ảnh có kích thước bé đi một nửa. Ở đây cũng sử dụng padding = “SAME”



Lớp 3: Lại là một lớp tích chập. Nhưng lần này sử dụng 64 bộ lọc có kích thước 5x5, padding = “SAME” và vẫn sử dụng hàm kích hoạt ReLU.



Lớp 4: Là lớp pooling giống lớp 2, giúp cho ảnh từ output của lớp trước đó có kích thước giảm đi một nửa.

Lớp 5: Là lớp Dropout. Lớp này có ý nghĩa là bỏ qua một vài node trong mạng kết nối. Với mỗi hidden layer, mỗi mẫu train, chúng ta sẽ bỏ bớt đi 25% số node của lớp và bỏ qua nó (cũng không áp dụng hàm kích hoạt cho nó). Lớp này giúp giảm overfitting.



Lớp 6: Là một lớp tích chập như các lớp trước, chỉ khác kích thước lần này sử dụng 128 bộ lọc kích thước 3x3.



Lớp 7: Là lớp pooling tương tự như trên, giúp giảm kích thước ảnh output vừa thu được đi một nửa.



Lớp 8: Là lớp Dropout như lớp 5:



Lớp 9: Là lớp Flatten. Output từ layer trước khi đi qua lớp này sẽ được duỗi ra thành một vector.



Lớp 10: Là lớp Dense là một lớp Fully connected. Tức là các note ở output thu được từ layer phía trước sẽ kết nối đầy đủ với 128 node ở lớp Dense này. Hàm kích hoạt được sử dụng là ReLU.



Lớp 11: Là một lớp Dropout với ý nghĩa như các lớp Dropout trên nhưng lần này sẽ bỏ qu 50% node.

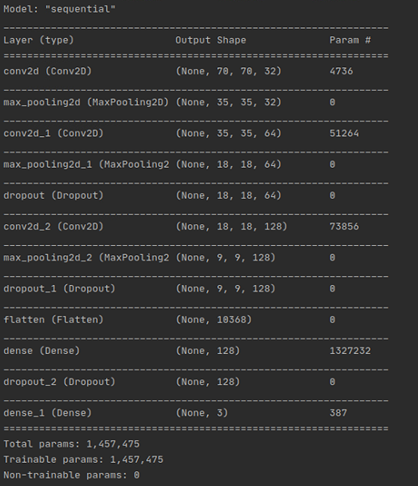


Lớp 12: Là lớp Fully connected cuối cùng. Các node ở output thu được từ layer phía trước sẽ kết nối đầy đủ với 3 node ở lớp Dense này. 3 lớp ứng với 3 class cần phân loại. Với hàm kích hoạt softmax được sử dụng để tính được xác suất ứng với các node. Node nào có giá trị xác suất cao hơn thì input sẽ thuộc class phân loại đó.



Thuật toán tối ưu được lựa chọn cho bài toán này là Stochastic Gradient Descent.

Mô tả tổng quan về mô hình: hình dạng đầu ra sau mỗi lớp ẩn và số lượng tham số cần tính toán và lưu trữ trong mô hình:



*3.3.3: Thực hiện train với bộ dữ liệu đã chuẩn bị và lưu trữ:*

Với bộ dữ liệu đã chuẩn bị, sẽ train dữ liệu qua 150 epochs.

Sau đó sẽ lưu trữ quá trình train dữ liệu lại

|  |
| --- |
|  |

Sử dụng các trọng số mô hình sau khi train đã lưu trữ:

|  |
| --- |
|  |

Dự đoán với tập dữ liệu mới được đưa vào, chưa từng được mô hình biết trước. Dữ liệu dùng để dự đoán này cũng phải đi qua quá trình xử lý resize và chuẩn hóa điểm ảnh như dữ liệu dùng để train.

|  |
| --- |
|  |

Mã nguồn đầy đủ của chương trình sẽ được trình bày trong phần phụ lục.

*3.3.4: Tiến hành trên video:*

Đọc video:

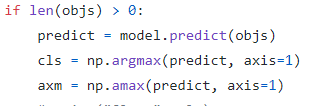


Đưa từng frame qua ObjectDetection():

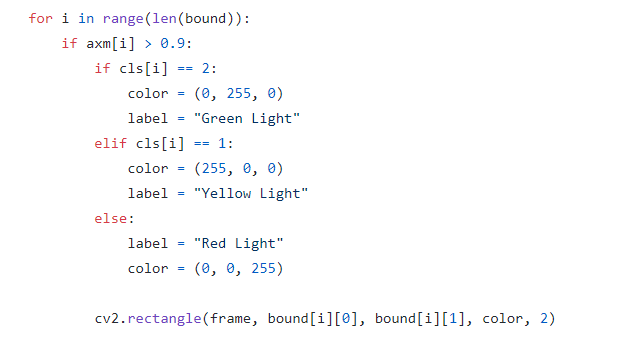




Đưa vào model dự đoán phân loại:



Gán nhãn đối tượng tìm được:



3.3.5. Thực nghiệm và kết quả thu được:

Kết quả train model:

|  |  |
| --- | --- |
|  |  |

Kết quả demo với video:

|  |  |
| --- | --- |
|  |  |
|  |  |

1. **Thảo luận và bài học thu được:**

Bài toán xây dựng phù hợp về mặt lý thuyết logic với những gì đã học trong môn xử lý ảnh.

Bên cạnh đó giúp hiểu được tầm quan trọng của xử lý ảnh trong học sâu nói riêng và trong đời sống nói chung.

1. **Tài liệu tham khảo:**

1)<https://www.researchgate.net/publication/224562453_Real_Time_Visual_Trafric_Lights_Recognition_Based_on_Spot_Light_Detection_and_Adaptive_Traffic_Lights_Templates>

2)<https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html>

3) <https://www.tensorflow.org/tutorials/quickstart/advanced?hl=vi>

4) <https://en.wikipedia.org/wiki/Normalization_(image_processing)>

1. **Phụ lục:**

* Mã nguồn của thuật toán chính:

|  |
| --- |
| class Model:  def \_\_init\_\_(self, epoch=60, input\_size=(90, 90), clasf=3, datasetname="dataset", modelSaveName="model.h5", learning\_rate=0.001, configGPU=True):  self.des = "Model CNN v1 by PhungHK"  print(self.des)  self.datasetname = datasetname  self.clasf = clasf  self.input\_size = input\_size  self.epochs = epoch  self.learning\_rate = learning\_rate  self.modelSaveName = modelSaveName  if configGPU:  self.config()  def config(self):  physical\_devices = tf.config.experimental.list\_physical\_devices('GPU')  assert len(physical\_devices) > 0, "Not enough GPU hardware devices available"  return tf.config.experimental.set\_memory\_growth(physical\_devices[0], True)  def pre\_data(self, img):  image = cv2.resize(img, self.input\_size)  image = np.array(image, dtype="float") / 255.0  return image  def training(self):  print("Loading image...")  imagePaths = list(paths.list\_images(self.datasetname))  random.seed(42)  random.shuffle(imagePaths)  # clasf = 3  labels = []  data = []  eye = np.eye(self.clasf, dtype=int)  print("Eye: {0}".format(eye))  # loop over the input images  for imagePath in imagePaths:  image = cv2.imread(imagePath)  image = self.pre\_data(image)  data.append(image)  label = imagePath.split(os.path.sep)[-2]  labels.append(np.asarray(eye[int(label), :]))  data = np.array(data)  print(data.shape)  labels = np.array(labels)  print(labels.shape)  (trainX, testX, trainY, testY) = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)  lb = LabelBinarizer()  trainY = lb.fit\_transform(trainY)  print(trainX.shape)  testY = lb.transform(testY)  model = Sequential()  model.add(Conv2D(32, (7, 7), input\_shape=(self.input\_size[0], self.input\_size[1], 3), padding="SAME", activation='relu'))  model.add(MaxPooling2D(pool\_size=(2, 2), padding="SAME"))  model.add(Conv2D(64, (5, 5), padding="SAME", activation='relu'))  model.add(MaxPooling2D(pool\_size=(2, 2), padding="SAME"))  model.add(Dropout(0.25))  model.add(Conv2D(128, (3, 3), padding="SAME", activation='relu'))  model.add(MaxPooling2D(pool\_size=(2, 2), padding="SAME"))  model.add(Dropout(0.25))  model.add(Flatten())  model.add(Dense(units=128, activation='relu'))  model.add(Dropout(0.5))  model.add(Dense(units=self.clasf, activation='softmax'))  print(model.summary())  opt = SGD(learning\_rate=self.learning\_rate)  model.compile(optimizer=opt, loss='categorical\_crossentropy', metrics=['accuracy'])  # Train the network  print("Training network...")  history = model.fit(trainX, trainY, epochs=self.epochs, validation\_data=(testX, testY))  # Save the network to disk  print("Saving model....")  model.save(self.modelSaveName)  print("Model saved!")  def loadding(self):  self.model = load\_model(self.modelSaveName)  def predict(self, listimg):  imgs = []  for im in listimg:  im = self.pre\_data(im)  imgs.append(im)  imgs = np.array(imgs)  return self.model.predict(imgs) |

Thư viện chuyên biệt để sử dụng:

Các thư viện cơ bản được cài đặt sẵn từ đầu môn học Xử lý ảnh: OpenCV, Matplotlib, Numpy, Scipy. Ngoài ra có thêm các thư viện khác phục vụ cho việc xây dựng mô hình.

Tensorflow v2: Dùng tính toán số học dựa trên biểu đồ mô tả sự thay đổi dữ liệu hay còn gọi là biểu đồ luồng dữ liệu

Cách cài đặt:

* Tải Anaconda về máy và bật Anaconda Prompt để cài đặt.
* Sử dụng Anaconda để cài Tensorflow bằng câu lệnh:

conda install -c conda-forge tensorflow

Keras: Dùng để xây dựng model

Cách cài đặt:

* Tải Anaconda về máy và bật Anaconda Prompt để cài đặt.
* Sử dụng Anaconda để cài Keras bằng câu lệnh:

conda install keras

Sklearn: Thư viện dành cho các thuật toán học máy. Cung cấp các công cụ xử lý các bài toán học máy như classfication, regression,…

Cách cài đặt:

* Tải Anaconda về máy và bật Anaconda Prompt để cài đặt.
* Sử dụng Anaconda để cài sklearn bằng câu lệnh:

conda install scikit-learn