TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO MÔN XỬ LÝ ẢNH VÀ ỨNG DỤNG

ĐỒ ÁN: PHÁT HIỆN BẢNG SỐ XE TRONG ẢNH SỬ DỤNG FASTER R-CNN

Lớp: CS231.I21

Sinh viên thực hiện: Hoàng Tùng Dương - 15520144

Mai Quốc Kiệt – 15520400

Lê Thiện Duy – 15520158

Vũ Mạnh Quốc - 15520703

Lời cảm ơn

Chúng em xin cảm ơn giảng viên hướng dẫn là thầy Đỗ Văn Tiến đã tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình học tập và thực hiện đồ án. Trong quá trình làm bài tập và đồ án, do thời gian và khả năng của bản thân còn hạn chế, nên chúng em không tránh khỏi sai sót. Vì vậy chúng em mong được sự bổ sung góp ý của thầy để hoàn thiện tốt hơn.

Chúng em xin chân thành cảm ơn thầy và chúc thầy gặp nhiều thành công trong cuộc sống

Mục lục

CHƯƠNG 1: GIỚI THIỆU

1. Giới thiệu về đồ án:

Áp dụng thuật toán Faster R-CNN để nhận diện bảng số xe bên trong bức ảnh được chọn với loss thấp

Cho phép người dùng chọn một hình ảnh ở trong máy, trả về hình ảnh kết quả là vùng hình chữ nhật có chứa biển số xe cùng với độ chính xác, hoặc trả về là not found khi không tìm được biển số xe trong ảnh

2. Cơ sở lý thuyết:

Faster R-CNN được coi là thuật toán neural network cho kết quả tốt nhất (nhưng không nhanh nhất) cho bài toán Object Detection.

Năm 2014, R-CNN được ra đời và sử dụng thuật toán Selective Search để tìm được những vùng được cho là có thể là object đó, sau đó áp dụng mạng CNN truyền thống để phân lớp và điều chỉnh lại.

Sau đó R-CNN được phát triển thành Fast R-CNN vào đầu năm 2015, khi đó áp dụng một kỹ thuật là Region of Interest Pooling cho phép chia sẻ những phép tính yêu cầu nhiều tài nguyên và giúp model thực hiện nhanh hơn.

Và cuối cùng là sự ra đời của Faster R-CNN

3. Công nghệ sử dụng:

IDE: Sublime Text

Ngôn ngữ lập trình: Python

Framework: Tensorflow

Thư viện cho giao diện: WxPython

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ

1. **Phân tích:**

   Tìm kiếm project trên github

   Cài đặt các thư viện cần thiết

   Tìm kiếm dữ liệu train (xin giáo viên)

   Thực hiện test trên model đã xây dựng

   Tiến hành xây dựng giao diện

2. **Cấu trúc dữ liệu:**

   Ảnh kết quả sẽ được lưu cùng folder với project với tên “result.jpg”

   Một mảng chứa tọa độ vị trí các bounding box xung quanh biển số xe nhận diện được

   Một mảng chứa giá trị là score của bounding đó.

   Một mảng chứa label là class của object đó (trong trường hợp này là biển số hoặc không là biển số

   Một mảng chứa số object là biển số xe đã detect được

CHƯƠNG 3: THỰC HIỆN

1. Lý thuyết về Faster R-CNN:

Faster R-CNN là một mạng CNN được phát triển qua nhiều giai đoạn: CNN → R-CNN → Fast CNN → Faster RCNN

a. CNN:

CNN là một mạng neural network sử dụng phép tích chập (convolution) trên các filter để rút trích đặc trưng cho quá trình phân lớp

CNN được sử dụng nhiều trong các bài toán nhận dạng object trong ảnh

CNN sẽ tập hợp các lớp tích chập 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.

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Kết quả sau quá trình học sẽ là các thông số tối ưu cho các filter tương ứng.

b. R-CNN:

CNN có một vấn đề chính đó là không biết trước có bao nhiêu đối tượng trong ảnh, nên không thiết kế được output layer hiệu quả. Điều này dẫn đến sự ra đời của R-CNN

R-CNN là viết tắt của Region-based CNN. Ý tưởng thuật toán R-CNN khá đơn giản:

Bước 1: Dùng Selective Search algorithm để lấy ra khoảng các bounding box trong input mà có khả năng chứa đối tượng.

Bước 2: Với mỗi bounding box ta xác định xem nó là đối tượng nào (người, ô tô, xe đạp,…)

Thuật toán Selective Search:

Input của thuật toán là ảnh màu, output là khoảng 2000 region proposal (bounding box) mà có khả năng chứa các đối tượng.

Đầu tiên ảnh sẽ được segmentation, các vùng sẽ được gom nhóm với nhau dựa trên độ tương đồng về màu sắc, hướng gradient, kích thước … Cuối cùng các region proposal được xác định dựa trên các nhóm vùng màu.

Phân loại các region proposal:

Bài toán trở thành phân loại ảnh cho các region proposal. Do thuật toán selective search cho tới 2000 region proposal nên có rất nhiều region proposal không chứa đối tượng nào.

Vậy nên ta cần thêm 1 lớp background (không chứa đối tượng nào). Ví dụ như ta có 4 region proposal, ta sẽ phân loại mỗi bounding box là người, ngựa hay background.

Sau đó các region proposal được resize lại về cùng kích thước và thực hiện traning để rút trích đặc trưng, sau đó các đặc trưng đã rút trích được cho vào thuật toán SVM để phân loại ảnh.

c. Fast R-CNN:

R-CNN còn một số điểm hạn chế, đó là với mỗi ảnh ta cần phân loại các class cho 2000 region proposal nên thời gian train rất lâu và vì đó không thể áp dung real-time detection.

Vì lý do đó, Fast R-CNN được ra đời với cùng tác giả của R-CNN. Nó giải quyết được các hạn chế của R-CNN để cải thiện tốc độ

Fast R-CNN cũng sử dụng selective search để lấy ra các region proposal. Tuy nhiên là Fast R-CNN không tách 2000 region proposal ra khỏi ảnh và thực hiện bài toán image classification cho mỗi vùng. Fast R-CNN cho cả bức ảnh vào

ConvNet (một vài convolutional layer + max pooling layer) để tạo ra convolutional feature map.

Sau đó các vùng region proposal được lấy ra tương ứng từ convolutional feature map. Tiếp đó được Flatten và thêm 2 fully connected layer (FCs) để dự đoán lớp của region proposal và giá trị offset values của bounding box.

Do Fast R-CNN đã resize các region proposal về cùng kích thước trước khi dùng transfer learning nên có thể áp dụng được neural network. Tuy nhiên ở feature map ta không thể resize được, nên cần phải chuyển feature map về chung kích thước bằng Region of Interest Pooling (ROI) ra đời.

ROI pooling là một dạng của pooling layer. Điểm khác so với max pooling hay average pooling là bất kể kích thước của tensor input, ROI pooling luôn cho ra output có kích thước cố định được định nghĩa trước.

Fast R-CNN khác với R-CNN là nó thực hiện feature map với cả ảnh sau đó với lấy các region proposal ra từ feature map, còn R-CNN thực hiện tách các region proposal ra rồi mới thực hiện CNN trên từng region proposal. Do đó Fast R-CNN nhanh hơn đáng kể nhờ tối ưu việc tính toán bằng việc đưa bài toán về dạng vector

d. Faster R-CNN:

Thời gian tính region proposal ở Fasr R-CNN rất lâu và làm chậm thuật toán, nên người ta nghĩ đến việc thay thế selective search bằng deep learning để tạo ra region proposal. Đó là tiền đề để Faster R-CNN ra đời

Với ảnh input, sau khi thực hiện Faster R-CNN, ta có thể lấy ra được một list các bounding box, một nhãn label ứng với mỗi bounding box và một xác xuất cho mỗi label và bounding box.

Bước đầu, ta sẽ sử dụng một model CNN train sẵn để thực hiện phân lớp (ví dụ ImageNet với mạng ZF hoặc VGG-16). Network này sẽ được gọi là base

network. Đây là bước dùng để rút trích đặc trưng để áp dụng cho bước tiếp theo với kết quả là một feature map.

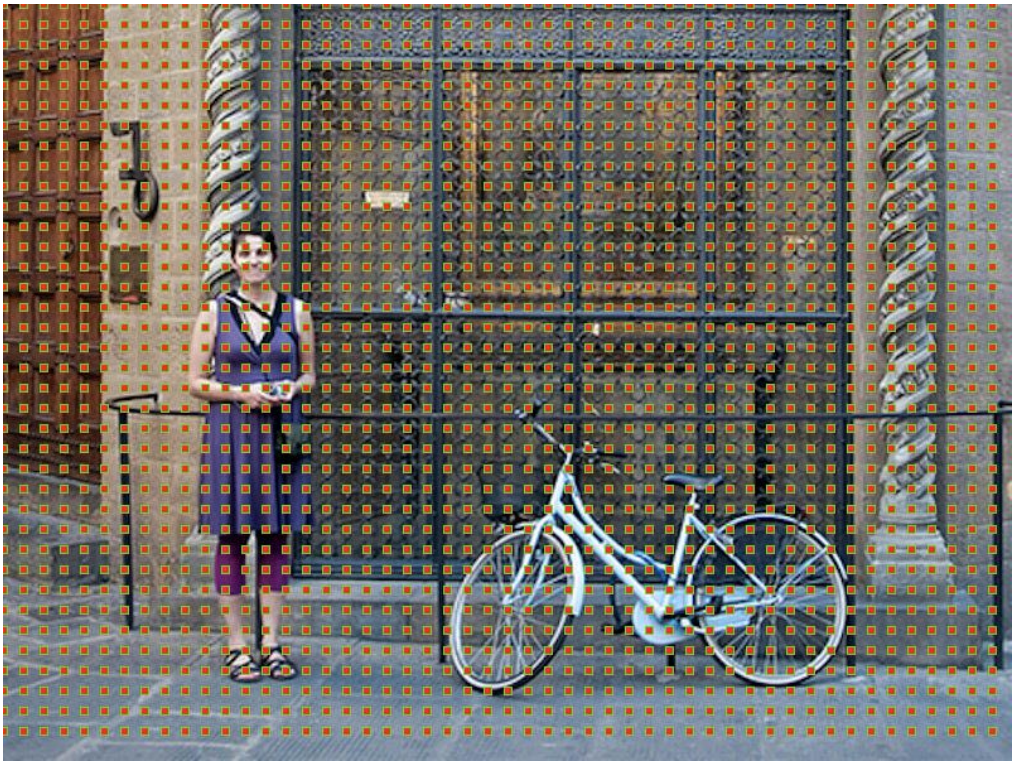
Với VGG, input sẽ là tensor 224x224x3 (ảnh RGB có kích thước 224x244). Mỗi lớp tích chập sẽ tạo ra thông tin dựa trên các thông tin trước đó. Các layer đầu tiên thường dùng để tìm cạnh, các lớp tiếp theo tìm các mẫu trong cạnh để có thể học được nhiều cạnh phức tạp hơn. Cuối cùng ta sẽ có kết quả là một feature map có kich thước nhỏ hơn ảnh input do áp dụng pooling nhưng có chiều sâu tăng nhiều hơn dựa trên số lượng filter. Nói tóm lại, feature map đã encode tất cả thông tin trong ảnh mà vẫn giữ được vị trí của những vật mà nó đã mã hóa dựa vào ảnh gốc.

Khi đã có feature map, ta cần tìm những vùng có thể chứa object (region proposal). Mục tiêu của chúng ta là tìm những bounding box bao quanh các region proposal. Chúng được biểu diễn dưới dạng các hình chữ nhật với kích thước và tỉ lệ khác nhau. Ý tưởng ban đầu là sẽ train một mạng neural mà trả về các giá trị x_{min}, x_{max}, y_{min}, y_{max} là tọa độ của bounding box.

Nhưng có một cách dễ dàng hơn là ta sẽ dự đoán bounding box đó bằng cách dự đoán các chỉ số. Chúng ta sẽ nhận vào x_{center}, y_{center} và width, height để dự đoán x_{min}, x_{max}, y_{min}, y_{max}, khi đó ta sẽ dể dàng thay đổi giá trị bounding box theo ý muốn.

Ở đây xuất hiện định nghĩa mới, đó là anchor. Anchor là một bounding box có vị trí không thay đổi mà được đặt khắp ảnh với các kích thước và tỉ lệ khác nhau mà sẽ được sử dụng cho việc tham khảo khi thực hiện dự đoán vị trí object.

Mặc dù anchor được định nghĩa dựa trên convolution feature map, những anchor cuối cùng sẽ dựa trên ảnh gốc. Nếu ảnh gốc có kích thước w, h thì feature map sẽ có kích thước w/r, h/r với r sẽ dựa vào mạng CNN áp dụng (trong trường hợp VGG-16 thì r = 16)

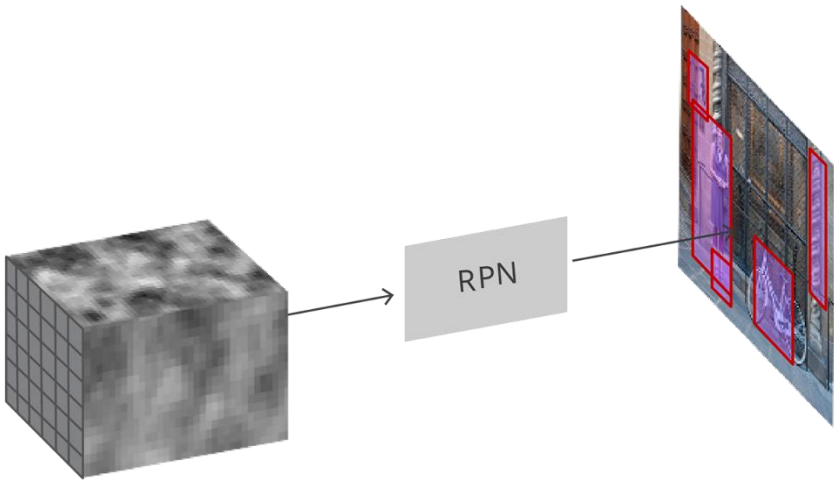


Các anchor center trên ảnh gốc

Region Proposal Network (RPN):

RPN có tác dụng sẽ lấy tất cả các anchor và trả về một tập hợp các proposal (proposal) có thể là object. Vì thế mỗi anchor sẽ có một trong hai kết quả đó là object hoặc background.

Kết quả đầu tiên, anchor đó là object sẽ được dựa vào “objectness score” (giá trị khả năng đó là object). RPN không quan tâm đó là object thuộc class nào, mà chỉ quan tâm là nó nhìn giống một object mà không phải là background.



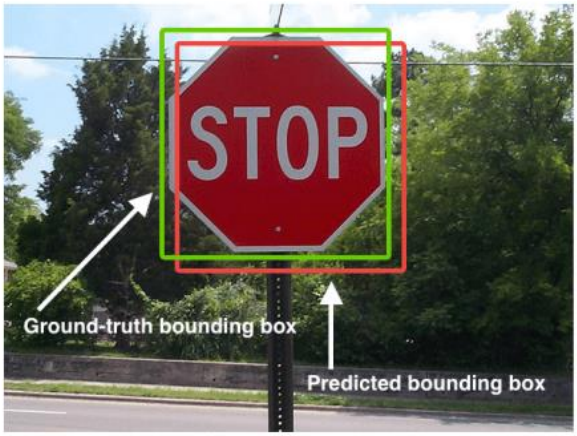
RPN nhận vào feature map và tạo ra các proposal là object trong ảnh

Với classification layer, ta sẽ đưa ra 2 output cho mỗi anchor: giá trị nó là background và giá trị nó là foreground (object)

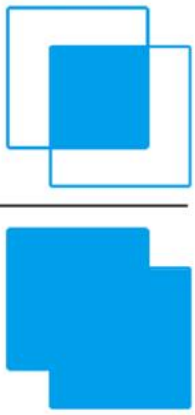
Với bounding box adjustment layer, chúng ta sẽ trả về 4 output Δxcenter, Δycenter, Δwidth, Δheight mà chúng ta sẽ sử dụng cho những anchor trong proposal cuối cùng. Sau bước này ta sẽ có một tập các proposal là object.

Training, target và hàm loss:

Ở bước này xuất hiện một định nghĩa là Intersection over Union (IoU). IoU được xử dụng trong bài toán object detection, để đánh giá xem bounding box dự đoán đối tượng khớp với ground truth của đối tượng.



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



RPN thực hiện hai loại dự đoán: phân lớp nhị phân và điều chỉnh lại bounding box.

Để training, ta lấy tất cả anchor và chia chúng thành hai loại khác nhau. Những anchor mà đè lên ground-truth của object mà có kết quả IoU lớn hơn 0.5 được cho là foreground và những anchor mà không đè lên hoặc có kết quả IoU nhỏ hơn 0.1 được cho là background.

Sau đó ta thực hiện tính loss function để dựa trên các foreground anchor và ground-truth của object gần nhất và tính toán giá trị Δ cần để chuyển đổi anchor vào object.

Trong trường hợp không tìm thấy bất kì anchor nào là foreground, ta sử dụng anchor có giá trị IoU gần ground truth box nhất.

Trong trường hợp các anchor đè lên nhau, ta áp dụng thuật toán Non-Maximum Suppression (NMS). Ta thực hiện như sau:

Bước 1: Chọn ra anchor box (A) có xác xuất là foreground lớn nhất trong tập Input
Bước 2: Thêm A vào tập Ouput.
Bước 3: Loại bỏ A và các anchor box trong tập Input mà có hệ số IoU với A lớn hơn 0.5 ra khỏi tập Input.
Bước 4: Kiểm tra nếu tập Input rỗng hoặc tập Output đủ 100 anchor thì dừng lại, nếu không quay lại bước 1.

Region of Interest Pooling:

Sau bước áp dụng RPN, chúng ta có nhóm những proposal mà không được gắn nhãn của bất cứ class nào. Vấn đề cần phải thực hiện đó là lấy những bounding box này và phân lớp chúng thành những loại mà chúng ta muốn

Cách tiếp cận dễ nhất là lấy từng proposal, cắt ra và sau đó đưa vào mạng đã được train trước. Faster R-CNN sẽ sử dụng convolution feature map có sẵn ở bước đầu. Việc này được thực hiện bằng cách rút trích các feature map có kích thước không thay đổi cho mỗi proposal bằng region of interest

pooling. Các feature map có kích thước không đổi là cần thiết cho R-CNN để phân lớp các proposal thành một số lượng class không đổi

Region-based Convolution Neural Network (R-CNN):

R-CNN là bước cuối cùng của Faster R-CNN. Sau khi có được một convolution feature map từ ảnh và sử dụng nó để lấy ra object proposal with RPN và cuối cùng rút trích đặc trưng cho từng proposal (bằng RoI Pooling), chúng ta sẽ áp dụng những đặc trưng này để phân lớp. R-CNN áp dụng tương tự những bước cuối cùng của CNN, lúc đó sẽ sử dụng một tầng liên kết đầy đủ để đưa ra một score cho mỗi class của object có thể có.

R-CNN có hai mục tiêu khác nhau:

Phân lớp các proposal thành một trong những class, cộng với một background class (để loại bỏ những proposal không tốt)

Điều chỉnh các bounding box sao cho tốt hơn dựa vào class đã dự đoán

Sau đó, R-CNN sử dụng hai tầng liên kết đầy đủ khác nhau cho từng object riêng biệt:

Một tầng liên kết đầy đủ với N + 1 đơn vị, trong đó N là tổng số class và một class được thêm vào là background class

Một tầng liên kết đầy đủ với 4N đơn vị. Chúng ta muốn có một dự đoán hồi quy, nên chúng ta sẽ cần Δcenterx, Δcentery, Δwidth, Δ height cho từng class thuộc N class có thể có.

Training và mục tiêu:

Mục tiêu của R-CNN là tính toán tương tự như RPN, nhưng xem xét đến những class có thể có. Chúng ta sử dụng những proposal và ground truth box, sau đó tính toán IoU giữa chúng.

Những proposal mà cho giá trị IoU lớn hơn 0.5 với bất cứ ground truth box nào được gán cho ground truth đó. Những proposal có giá trị từ 0.1 đến 0.5 được gán nhãn là background.Trái ngược với những gì chúng ta đã làm khi tập hợp các mục tiêu cho RPN, chúng ta bỏ qua các proposal mà không giao với ground truth box. Đó là bởi vì ở giai đoạn này chúng ta giả định rằng chúng ta đã có những proposal đủ tốt và chúng ta mong muốn giải quyết vấn đề khó hơn. Đương nhiên, các giá trị này là những hyperparameter mà có thể thay đổi để phù hợp hơn với kiểu của object mà chúng ta mong muốn tìm kiếm.

Những đối tượng cho việc hồi quy bounding box được tính toán dựa vào tọa độ giữa proposal của ground truth box tương ứng của nó, chỉ áp dụng cho những proposal được gán nhãn dựa theo IoU threshold. Tương tự như RPN, tiếp theo chúng ta sẽ tính loss cho việc phân lớp.

Quá trình hậu xử lý:

Tương tự như RPN, chúng ta có một số lượng các object với class của chúng và cần xử lý thêm trước khi trả về.

Để có thể áp dụng điều chỉnh bounding box, ta cần xét đến class nào cho khả năng cao nhất cho proposal đó. Chúng ta cũng cần bỏ qua những proposal mà có background class cho khả năng cao nhất.

Sau khi lấy được những object cuối cùng và bỏ qua những proposal được dự đoán là background, chúng ta áp dụng NMS. Việc này được thực hiện bằng cách nhóm các object theo class và sắp xếp chúng theo khả năng và sau đó áp dụng NMS cho từng nhóm độc lập trước khi kết hợp chúng lại với nhau.

Với danh sách cuối cùng các object, chúng ta có thể đặt cho chúng một threshold cho giá trị khả năng và giới hạn số lượng object của mỗi class

Training:

Faster R-CNN dược train sử dụng một cách tiếp cận nhiều bước, training từng phần độc lập với nhau và ghép những weight đã train được trước khi thực hiện tiếp cận đầy đủ cuối cùng.

Sau khi kết hợp thành model đầy đủ, chúng ta sẽ có bốn giá trị loss khác nhau, hai cho RPN và hai cho R-CNN. Chúng ta có tầng có thể train trong RPN và R-CNN, và chúng ta cũng có mạng neural cơ bản mà chúng ta có thể train.

Quyết định có train mạng cơ bản hay không phụ thuộc vào bản chất của object ta muốn train và khả năng tính toán có sẵn. Nếu ta muốn nhận diện các object mà tương tự như các object có trong dataset để train, chúng ta không cần thiết phải train mạng neural cơ bản đó. Ngược lại, nếu chúng ta muốn nhận diện những ảnh mà khác so với dataset ở một khía cạnh nào đó, chúng ta sẽ cần phải train mạng neural cơ bản. Tuy nhiên, chúng ta cần phải cân nhắc bởi vì training có thể tốn nhiều thời gian và phần cứng cần thiết.

Tốc độ học bắt đầu từ 0.001 và sau đó giảm xuống 0.0001 sau 50000 bước. Đây là một trong những hyperparameter quan trọng nhất. Khi train, ta thường bắt đầu với giá trị cơ bản và thay đổi sau đó.

Đánh giá:

Việc đánh giá sẽ dựa trên Mean Average Precision (mAP) ở một vài IoU threshold cố định. mAP là một đơn vị đo xuất hiện từ truy vấn thông tin và thường được dùng để tính toán lỗi trong những vấn đề liên quan đến xếp hạng và đánh giá những vấn đề liên quan đến việc nhận diện object.

mAP sẽ giảm khi bạn không nhận diện ra một box mà đáng lẽ nên được nhận diện, hoặc khi bạn nhận diện một thứ không có hoặc nhận diện một thứ nhiều lần.

2. Thực hiện:

Tạo data training:

Nhóm thực hiện train trên dataset có được (dataset được giáo viên cung cấp)

Chuyển file .xml của database thành file .csv

Thay đổi số lượng class và tên class cho phù hợp với dữ liệu training

Tạo labelmap và cấu hình configure training:

Tạo file labelmap.pbtxt có nội dung: item {

id: 1

name: 'plate'

}

Thay đổi path trong file "faster_rcnn_inception_v2_pets.config" cho phù hợp với ý muốn.

Training: chạy file train.py để train.py, train đến khi loss trong khoảng 0.003 đến 0.2 thì dừng.

Export Inference Graph: tạo một inference graph không thay đổi được và export ra file .pb. File .pb này sẽ chứa bộ phân lớp để nhận diện object

Kết quả sau quá trình train là: file model.ckpt chứa model đã được train

Sau đó nhóm thực hiện detect trên model đã train:

Load detection graph .pb file từ thư mục inference_graph

Load model đã train từ thư mục training

Load label map chứa các label của các class (ví dụ kết quả dự đoán là 5 → car)

Tạo một tensorflow graph bằng hàm tensorflow.Graph(). Tensorflow sử dụng một dataflow graph để thể hiện việc tính toán liên quan tới sự phụ

thuộc giữa các phép tính riêng lẻ, sau đó tạo ra một tensorflow session để chạy các phần của graph trên một tập hợp các thiết bị ở local và remote.

Dataflow mà một model lập trình thông dụng cho việc tính toán song song. Trong một dataflow graph, node sẽ thể thiện các đơn vị tính toán, cạnh thể hiện dữ liệu tiêu thụ hoặc tạo ra dựa vào tính toán.

tf.Graph() chứa hai loại thông tin:

Graph structure: node và cạnh của graph, thể hiện những phép toán đơn lẻ được liên kết với nhau như thế nào.

Graph collections: tensorflow cung cấp một cơ chế chung để lưu trữ các collection chứa metadata trong tf.Graph

Sau đó ta sử dụng hàm as_default() để tạo thêm graph trong cùng một quá trình thực hiện. Hàm này sẽ tạo ra một global default graph cùng với tất cả các toán tử đi chung với graph đó.

Tiếp theo ta sử dụng hàm graphDef() để tạo ra một graph có thể sử dụng được ở bất kỳ tensorflow front-end nào (Python, R, C++, Java, …) và được lưu trữ dưới dạng file .pb.

Sau đó ta import graphDef đó vào buffer, rút trích các object riêng lẻ như tf.Tensor và tf.Operation. Sau khi rút trích, những object này được đặt vào default graph đã được tạo từ các bước trước bằng hàm tf.import_graph_def().

Sau khi đã import graphDef, ta cần định nghĩa input và output tensor. Input tensor sẽ là bức ảnh được đưa vào, output tensor sẽ là bounding box, score các class và số lượng object nhận diện được. Score sẽ được thể hiện ở trên ảnh kết quả, cùng với label của class. Cuối cùng ta tạo một session dựa trên default graph đã tạo.

```
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
```

Input và Output Tensor

```
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
    sess = tf.Session(graph=detection_graph)
```

Tạo graphDef và session

Sau đó ta tạo tiến hành chạy tensorflow session bằng hàm run(). Lúc này máy tính sẽ thực hiện phát hiện object trong ảnh dựa trên model đã được train:

```
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})
```

Kết quả sau khi thực hiện detect trên model, ta nhận được bốn tham số: boxes chứa bounding box bao quanh các object, scores chứa score của object, classes chứa các lớp của object và num chứa số lượng các object đã detect được.

Khi đã có các kết quả trên, nhóm tiến hành vẽ kết quả lên bức ảnh ban đầu và lưu cùng thư mục với file .py với tên “result.jpg”

CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

1. Giao diện của chương trình:

Thư viện cần để chạy chương trình:

Python: 3.6.8

Tensorflow GPU 1.12

Cuda 9

Cuda NN 7.3

WxPython

OpenCV

Và một số thư viện nhỏ

Link cài đặt các thư viện được lưu trong file readme.md trong repository github của nhóm

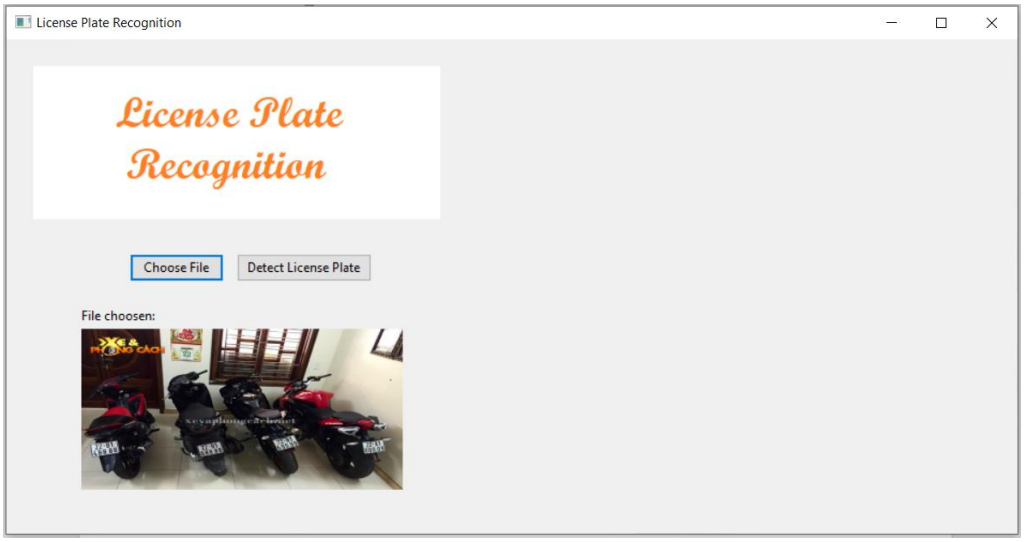
Để chạy chương trình, ta vào file UI.py và bấm tools → build hoặc ctrl + B

Các chức năng của chương trình:

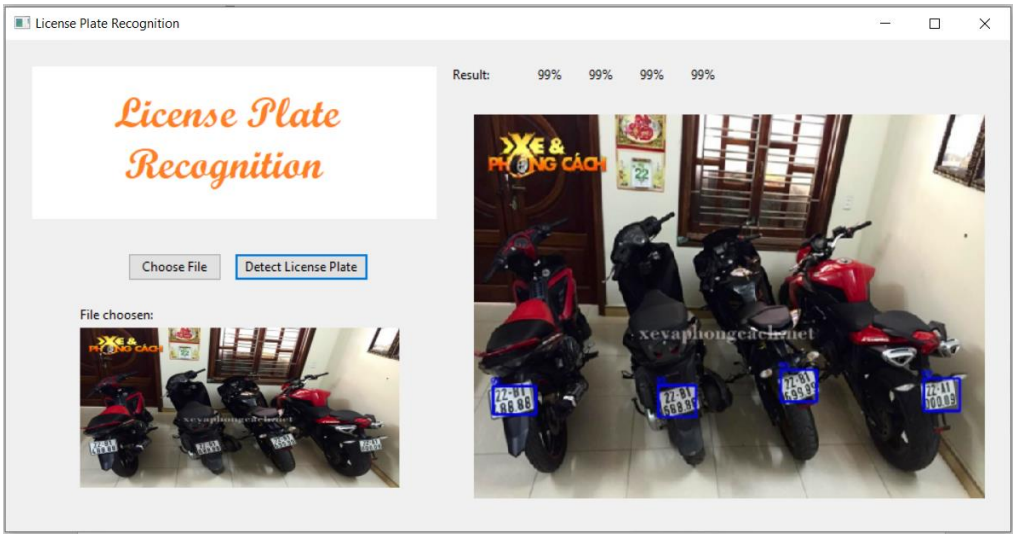
Nút Choose File để chọn ảnh

Nút Detect License Plate để tiến hành nhận diện bảng số xe

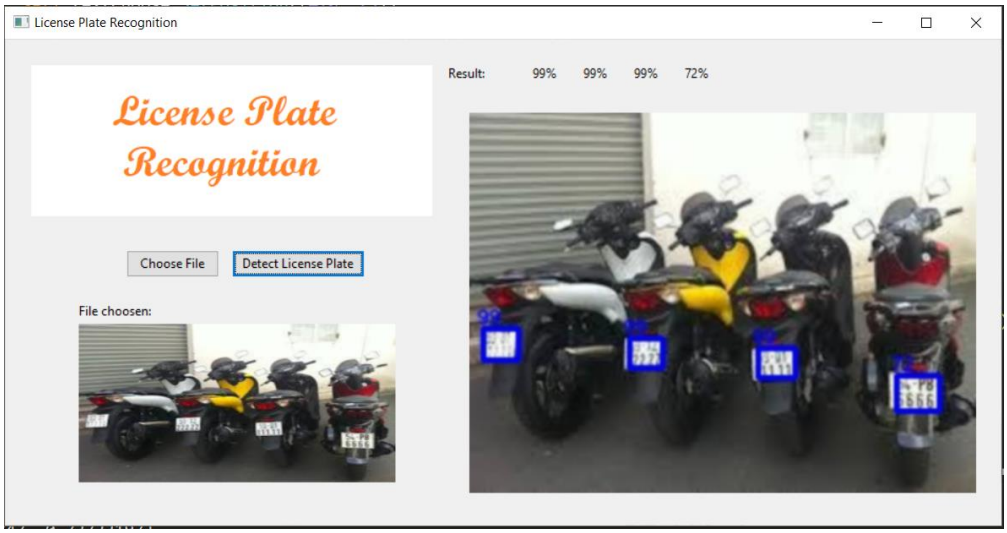
2. Kết quả chạy chương trình:



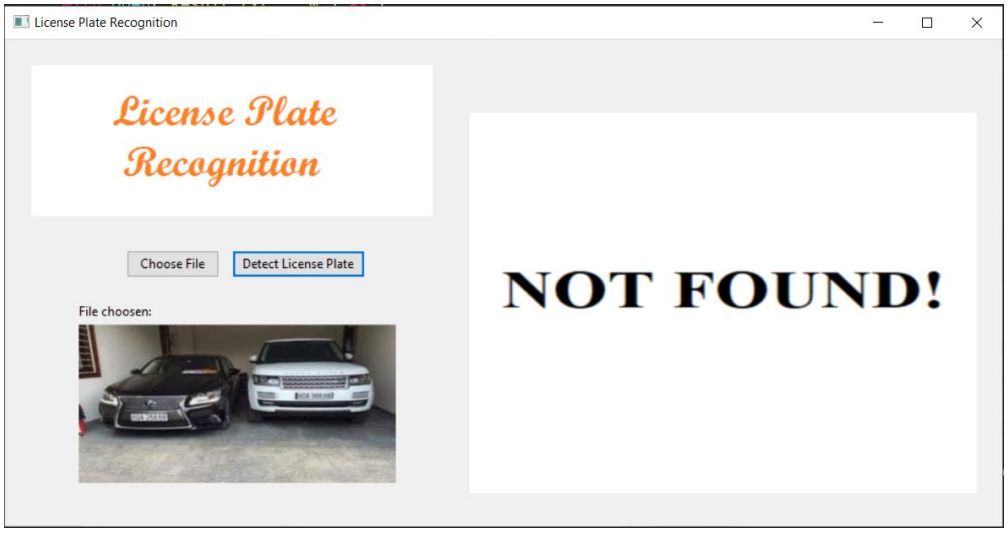
Chọn ảnh cần nhận diện



Kết quả chạy chương trình khi nhận diện được



Kết quả chạy chương trình khi nhận diện được



Kết quả chạy chương trình khi không nhận diện được

(do dataset là bảng số xe máy)

3. Kết quả trả về:

File ảnh kết quả "result.jpg"

Tọa độ các bounding box được lưu trong file "result.txt":

62	57	69	64
59	31	67	38
57	3	65	9
69	84	78	92

CHƯƠNG 5: KẾT LUẬN

1. **Kết quả đạt được:**

   Giao diện của chương trình đã hoàn thành

   Chương trình chạy đúng và cho kết quả chấp nhận được

2. **Hạn chế:**

   Cần phải có hiểu biết về Python và phải cài thư viện

   Xây dựng web service không thành công

3. **Hướng phát triển:**

   Deploy web service lên hosting

   Thực hiện thêm nhận diện chữ trong bảng số đã nhận diện được

4. **Phân công công việc:**

   Về việc lên ý tưởng: cả nhóm cùng họp nhóm và đóng góp ý tưởng, sau đó nhóm trưởng sẽ thống nhất và chia công việc cho các thành viên trong nhóm

   Nhóm sẽ có các buổi họp nhóm, các thành viên sẽ đều tham gia. Thành viên nào được giao nhiệm vụ sẽ code chính nhiệm vụ đó, còn các thành viên còn lại sẽ đóng góp ý kiến và hỗ trợ giải quyết vấn đề. Các thành viên nhóm sẽ đọc hiểu project và chia sẻ kiến thức với nhau.

   Mai Quốc Kiệt: thực hiện training

   Lê Thiện Duy: làm UI, xử lý các kết quả training

   Vũ Mạnh Quốc: làm UI, tích hợp UI với code xử lý

   Hoàng Tùng Dương: tìm hiểu Flask, làm báo cáo, slide

5. **Biên bản làm việc nhóm:**

Tiêu chuẩn đánh giá	Tỉ trọng	Vũ Mạnh Quốc	Mai Quốc Kiệt	Lê Thiện Duy	Hoàng Tùng Dương
Đóng góp về nội dung, chất lượng làm bài	**50%**	**100%**	**100%**	**100%**	**100%**
Giải quyết khó khăn khi làm bài, tư duy sáng tạo	**15%**	**100%**	**100%**	**100%**	**100%**
Hợp tác, nhiệt tình, tinh thần trách nhiệm cao	**15%**	**100%**	**100%**	**100%**	**100%**
Sự chuyên cần	**10%**	**100%**	**100%**	**100%**	**100%**
Tư duy phản biện, biết tôn trọng, lắng nghe	**10%**	**100%**	**100%**	**100%**	**100%**

6. **Tài liệu tham khảo:**

Các slide bài giảng của giảng viên

Documentation của TensorFlow: www.tensorflow.org

Tài liệu hướng dẫn giao diện: http://zetcode.com/wxpython/

Tìm hướng giải quyết các vấn đề chưa xử lý được: https://stackoverflow.com/

Link Github nhóm tham khảo: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

COCO trained model: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Và một số trang web khác

7. **Link Github thực hiện project của nhóm:**

https://github.com/Duyle2468/License-Plate-Recognition