Lời cảm ơn !

Lời mở đầu

1.Lý do chọn đề tài

2.Mục tiêu nghiên cứu

3.Phương pháp nghiên cứu

4.Đối tượng nghiên cứu

5.Kết cấu :gồm 4 chương

- Chương 1: Giới thiệu về lĩnh vực nghiên cứu

- Chương 2: Cơ sở nhận dạng cháy nổ

- Chương 3: Xây dựng chương trình nhận dạng

- Chương 4: Kết quả và đánh giá

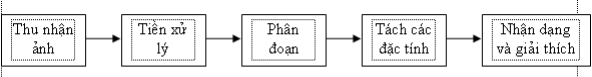
CHƯƠNG I : GIỚI THIỆU SƠ BỘ VỀ LĨNH VỰC NGHIÊN CỨU

1.Lĩnh vực xử lý ảnh

* 1. Tổng quan về lĩnh vực

Xử lý ảnh (XLA) là đối tượng nghiên cứu của lĩnh vực thị giác máy, là quá trình biến đổi từ một ảnh ban đầu sang một ảnh mới với các đặc tính và tuân theo ý muốn của ngƣời sử dụng. Xử lý ảnh có thể gồm quá trình phân tích, phân lớp các đối tượng, làm tăng chất lượng, phân đoạn và tách cạnh, gán nhãn cho vùng hay quá trình biên dịch các thông tin hình ảnh của ảnh. Cũng như xử lý dữ liệu bằng đồ hoạ, xử lý ảnh số là một lĩnh vực của tin học ứng dụng. Xử lý dữ liệu bằng đồ họa đề cập đến những ảnh nhân tạo, các ảnh này được xem xét như là một cấu trúc dữ liệu và được tạo bởi các chương trình. Xử lý ảnh số bao gồm các phương pháp và kỹ thuật biến đổi, để truyền tải hoặc mã hoá các ảnh tự nhiên. Mục đích của xử lý ảnh gồm: Biến đổi ảnh làm tăng chất lượng ảnh. Tự động nhận dạng ảnh, đoán nhận ảnh, đánh giá các nội dung của ảnh. Nhận biết và đánh giá các nội dung của ảnh là sự phân tích một hình ảnh thành những phần có ý nghĩa để phân biệt đối tượng này với đối tượng khác, dựa vào đó ta có thể mô tả cấu trúc của hình ảnh ban đầu. Có thể liệt kê một số phương pháp nhận dạng cơ bản như nhận dạng ảnh của các đối tượng trên ảnh, tách cạnh, phân đoạn hình ảnh,… Kỹ thuật này được dùng nhiều trong y học (xử lý tế bào, nhiễm sắc thể), nhận dạng chữ trong văn bản.

* 1. Các quá trình xử lý ảnh



Thu nhận ảnh: Đây là công đoạn đầu tiên mang tính quyết định đối với quá trình XLA. Ảnh đầu vào sẽ được thu nhận qua các thiết bị như camera, sensor, máy scanner,v.v… và sau đó các tín hiệu này sẽ được số hóa. Việc lựa chọn các thiết bị thu nhận ảnh sẽ phụ thuộc vào đặc tính của các đối tượng cần xử lý. Các thông số quan trọng ở bước này là độ phân giải, chất lượng màu, dung lượng bộ nhớ và tốc độ thu nhận ảnh của các thiết bị.

Tiền xử lý: Ở bước này, ảnh sẽ được cải thiện về độ tương phản, khử nhiễu, khử bóng, khử độ lệch,v.v… với mục đích làm cho chất lượng ảnh trở lên tốt hơn nữa, chuẩn bị cho các bước xử lý phức tạp hơn về sau trong quá trình XLA. Quá trình này thường được thực hiện bởi các bộ lọc.

Phân đoạn ảnh: phân đoạn ảnh là bước then chốt trong XLA. Giai đoạn này phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là cùng màu, cùng mức xám v.v… Mục đích của phân đoạn ảnh là để có một miêu tả tổng hợp về nhiều phần tử khác nhau cấu tạo lên ảnh thô. Vì lượng thông tin chứa trong ảnh rất lớn, trong khi đa số các ứng dụng chúng ta chỉ cần trích một vài đặc trưng nào đó, do vậy cần có một quá trình để giảm lượng thông tin khổng lồ đó. Quá trình này bao gồm phân vùng ảnh và trích chọn đặc tính chủ yếu.

Tách các đặc tính: Kết quả của bước phân đoạn ảnh thường được cho dưới dạng dữ liệu điểm ảnh thô, trong đó hàm chứa biên của một vùng ảnh, hoặc tập hợp tất cả các điểm ảnh thuộc về chính vùng ảnh đó. Trong cả hai trường hợp, sự chuyển đổi dữ liệu thô này thành một dạng thích hợp hơn cho việc xử lý trong máy tính là rất cần thiết. Để chuyển đổi chúng, câu hỏi đầu tiên cần phải trả lời là nên biểu diễn một vùng ảnh dưới dạng biên hay dưới dạng một vùng hoàn chỉnh gồm tất cả những điểm ảnh thuộc về nó. Biểu diễn dạng biên cho một vùng phù hợp với những ứng dụng chỉ quan tâm chủ yếu đến các đặc trưng hình dạng bên ngoài của đối tượng, ví dụ như các góc cạnh và điểm uốn trên biên chẳng hạn. Biểu diễn dạng vùng lại thích hợp cho những ứng dụng khai thác các tính chất bên trong của đối tượng, ví dụ như vân ảnh hoặc cấu trúc xương của nó. Sự chọn lựa cách biểu diễn thích hợp cho một vùng ảnh chỉ mới là một phần trong việc chuyển đổi dữ liệu ảnh thô sang một dạng thích hợp hơn cho các xử lý về sau. Chúng ta còn phải đưa ra một phương pháp mô tả dữ liệu đã được chuyển đổi đó sao cho những tính chất cần quan tâm đến sẽ được làm nổi bật lên, thuận tiện cho việc xử lý chúng.

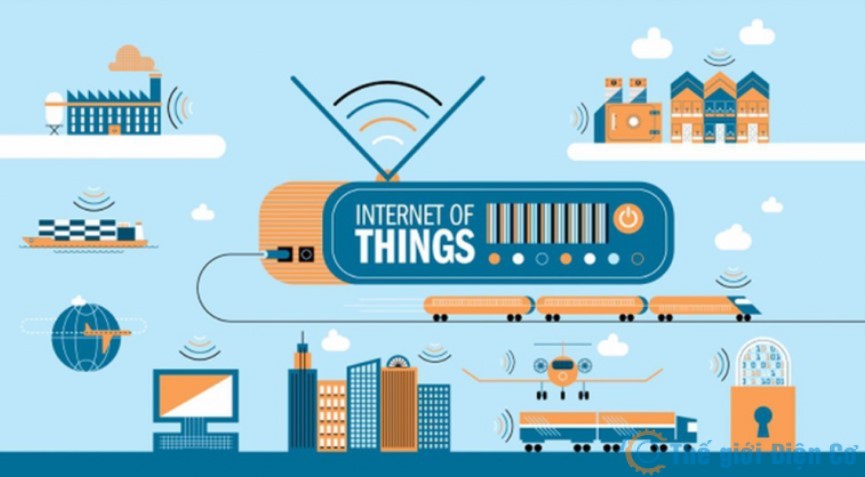
Nhận dạng và giải thích: Đây là bước cuối cùng trong quá trình XLA. Nhận dạng ảnh có thể được nhìn nhận một cách đơn giản là việc gán nhãn cho các đối tượng trong ảnh. Ví dụ đối với nhận dạng chữ viết, các đối tượng trong ảnh cần nhận dạng là các mẫu chữ, ta cần tách riêng các mẫu chữ đó ra và tìm cách gán đúng các ký tự của bảng chữ cái tương ứng cho các mẫu chữ thu được trong ảnh. Giải thích là công đoạn gán nghĩa cho một tập các đối tượng đã được nhận biết. Chúng ta cũng có thể thấy rằng, không phải bất kỳ một ứng dụng XLA nào cũng bắt buộc phải tuân theo tất cả các bước xử lý đã nêu ở trên, ví dụ như các ứng dụng chỉnh sửa ảnh nghệ thuật chỉ dừng lại ở bước tiền xử lý. Một cách tổng quát thì những chức năng xử lý bao gồm cả nhận dạng và giải thích thường chỉ có mặt trong hệ thống phân tích ảnh tự động hoặc bán tự động, được dùng để rút trích ra những thông tin quan trọng từ ảnh, ví dụ như các ứng dụng nhận dạng ký tự quang học, nhận dạng chữ viết tay v.v…

* 1. Phạm vi ứng dụng của xử lý ảnh

Trong mấy thập kỷ gần đây, xử lý ảnh đã được nghiên cứu mạnh mẽ và đã có rất nhiều ứng dụng trong thực tế.Như trong y học, xử lý ảnh số đã được dùng đẻ phát hiện và nhận dạng khối u ,cải thiện ảnh X quang ,các hình ảnh trong siêu âm.Còn trong lĩnh vực khoa hoc kỹ thuật,xử lý ảnh đã và đang có những đóng góp quan trọng, đặc biệt là trong lĩnh vực camera giám sát. Từ việc nhận dạng có thể giải quyết nhiều bài toán như chống trộm, tìm vật, nhận dạng các đối tượng …Từ những cơ sở trên ,việc nghiên cứu bài toán xử lý ảnh trong thời đại ngày nay là rất quan trọng và không thể thiếu trong hầu hết các lĩnh vực

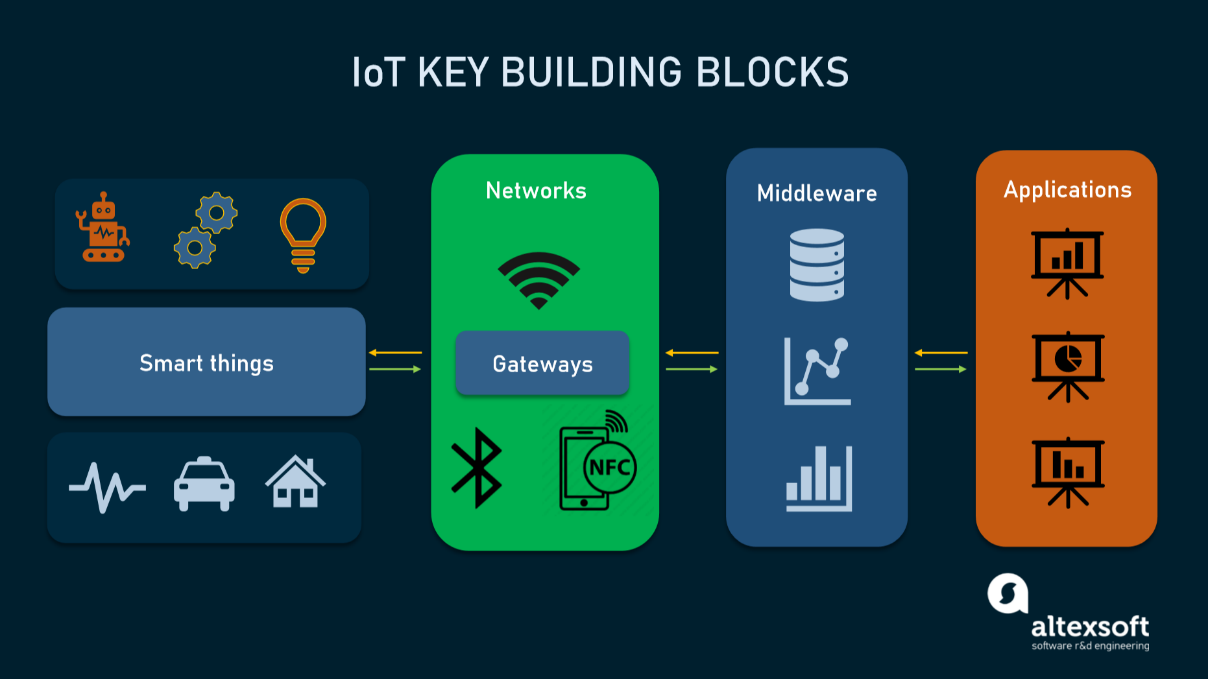
1. Lĩnh vực IOT
   1. Giới thiệu về IOT

Thuật ngữ Internet of things (viết tắt là IoT) được hiểu một cách đơn giản là một mạng lưới vạn vật kết nối với nhau thông qua Internet. Chúng bao gồm các đồ vật, con người được cung cấp một định danh của riêng mình và tất cả có khả năng truyền tải hay trao đổi thông tin hay dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và quan trọng hơn là sự có mặt của Internet. Nói đơn giản IoT là một tập hợp các thiết bị có khả năng kết nối mọi thứ lại với nhau với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó



Internet of things là gì ?

2.2 Kiến trúc hệ thống IOT



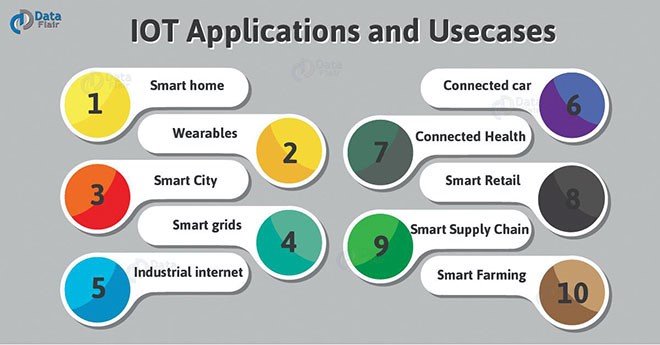
-Các thiết bị : cảm biến ,những thiết bị máy móc …dùng để thu thập thông tin ,gửi thông tin hoặc cả hai

-Trạm kết nối :Bao gồm các cơ sở hạ tầng mạng ,các giao thức truyền thông. Các trạm kết nối sẽ đóng vai trò là một trung gian trực tiếp, cho phép các vật dụng có sẵn này kết nối với điện toán đám mây một cách bảo mật và dễ dàng quản lý.

-Tầng đám mây (cloud) : cung cấp không gian lưu trữ dữ liệu cùng với khả năng phân tích dữ liệu

-Tầng ứng dụng: Cho phép người dùng có thể giám sát các thông tin ,thông số được gửi từ thiết bị về hoặc là gửi tín hiệu của người dùng đến thiết bị

2.3 Các ứng dụng của IOT



Nhà thông minh. ...

Sản phẩm có thể đeo được. ...

Thành phố thông minh. ...

Lưới thông minh. ...

Internet công nghiệp. ...

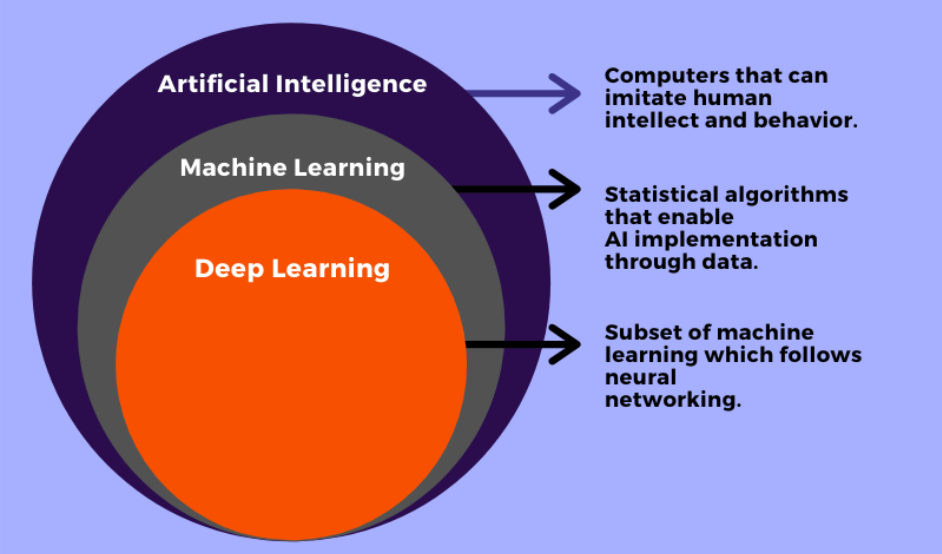
Xe được kết nối. ...

Sức khỏe được kết nối (Sức khỏe kỹ thuật số / Telehealth / Telemedicine) ...

Bán lẻ thông minh.

3. Lĩnh vực Trí tuệ nhân tạo

3.1 AI,ML,DL



- Trí tuệ nhân tạo (AI): một cỗ máy có thể bắt chước hành vi và tư duy của con người

-Học máy (machine learning): Một tính năng của AI, cho phép các chuyên gia đào tạo cho AI để nó nhận biết các mẫu dữ liệu và dự đoán

-Học sâu (deep learning): Một kỹ thuật nhỏ của machine learning, cho phép máy có thể tự đào tạo chính mình

* 1. Ứng dụng của trí tuệ nhân tạo

Trong ngành vận tải : xe tự lái

Trong sản xuất : Trí tuệ nhân tạo được ứng dụng để xây dựng những quy trình sản xuất tối ưu hơn. Công nghệ AI có khả năng phân tích cao, làm cơ sở định hướng cho việc ra quyết định trong sản xuất.

Trong y tế: Ứng dụng tiêu biểu của trí tuệ nhân tạo trong lĩnh vực y tế là máy bay thiết bị bay không người lái được sử dụng trong những trường hợp cứu hộ khẩn cấp. Thiết bị bay không người lái có tốc độ nhanh hơn xe chuyên dụng đến 40% và vô cùng thích hợp để sử dụng ở những nơi có địa hình hiểm trở. AI đưa ra quyết định lâm sàng ,các chuẩn đoán về chứng bệnh.AI dựa trên nguồn dữ liệu 129.450 hình ảnh lâm sàng để chẩn đoán bệnh ngoài da, kết quả đã chứng minh rằng hệ thống này có thể phân loại ung thư da ở mức tương đương với các bác sĩ da liễu

Trong giáo dục:Sự ra đời của trí tuệ nhân tạo giúp tạo ra những thay đổi lớn trong lĩnh vực giáo dục. Các hoạt động giáo dục như chấm điểm hay dạy kèm học sinh có thể được tự động hóa nhờ công nghệ AI. Nhiều trò chơi, phần mềm giáo dục ra đời đáp ứng nhu cầu cụ thể của từng học sinh, giúp học sinh cải thiện tình hình học tập theo tốc độ riêng của mình.

Trong truyền thông : Đối với lĩnh vực truyền thông, sự phát triển của trí tuệ nhân tạo góp phần làm thay đổi cách thức tiếp cận đối với khách hàng mục tiêu. Nhờ những ưu điểm của công nghệ AI, các công ty có thể cung cấp quảng cáo vào đúng thời điểm, đúng khách hàng tiềm năng, dựa trên việc phân tích các đặc điểm về nhân khẩu học, thói quen hoạt động trực tuyến và những nội dung mà khách hàng thường xem trên quảng cáo.

Trong ngành dịch vụ : Công nghệ AI giúp ngành dịch vụ hoạt động tối ưu hơn và góp phần mang đến những trải nghiệm mới mẻ hơn và tốt hơn cho khách hàng. Thông qua việc thu thập và phân tích dữ liệu, công nghệ AI có thể nắm bắt thông tin về hành vi sử dụng dịch vụ của khách hàng, từ đó mang lại những giải pháp phù hợp với nhu cầu của từng khách hàng.

CHƯƠNG II : CƠ SỞ NHẬN DẠNG LỬA VÀ PHÁT TÍN HIỆU BÁO CHÁY

1. Nhận dạng đối tượng trong video
   1. Tổng quan về Object Detection

Phân loại hình ảnh (image classification): liên quan đến việc gán nhãn cho hình ảnh.

Định vị vật thể (object localization): liên quan đến việc vẽ một hộp giới hạn (bounding box) xung quanh một hoặc nhiều đối tượng trong hình ảnh nhằm khoanh vùng đối tượng.

Phát hiện đối tượng (object detection): Là nhiệm vụ khó khăn hơn và là sự kết hợp của cả hai nhiệm vụ trên: Vẽ một bounding box xung quanh từng đối tượng quan tâm trong ảnh và gán cho chúng một nhãn. Kết hợp cùng nhau, tất cả các vấn đề này được gọi là object recognition hoặc object detection

Lịch sử hình thành, phát triển và đặc điểm cấu trúc của các thuật toán object detection bao gồm 2 nhóm chính:

Họ các mô hình R-CNN (Region-Based Convolutional Neural Networks) giải quyết các nhiệm vụ định vị vật thể và nhận diện vật thể.

Họ các mô hình YOLO (You Only Look Once), là một nhóm kỹ thuật thứ hai để nhận dạng đối tượng được thiết kế để nhận diện vật thể real time.

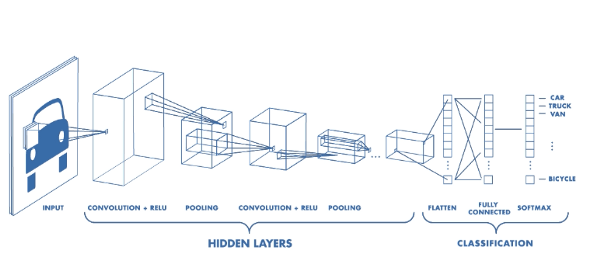
Các phương pháp nhận dạng Haar HOG ,…

1.2 Đánh giá giữa các phương pháp

1.2.1 Giữa phương pháp truyền thống và hiện đại

* + 1. Giữa các phương pháp hiện đại

1. Mạng nơron tích chập (CNN)
   1. Kiến trúc chung của mạng neural tích chập



Trong đó:

INPUT : Ảnh đầu vào

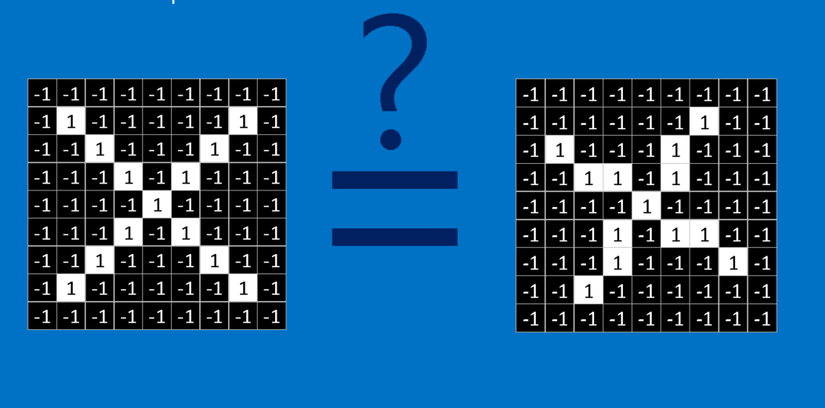
CONVOLUTION :Tầng tích chập để tìm ra các đặc trưng trong bức ảnh

RELU: Tầng biến đổi thông qua hàm kích hoạt relu để kích hoạt phi tuyến

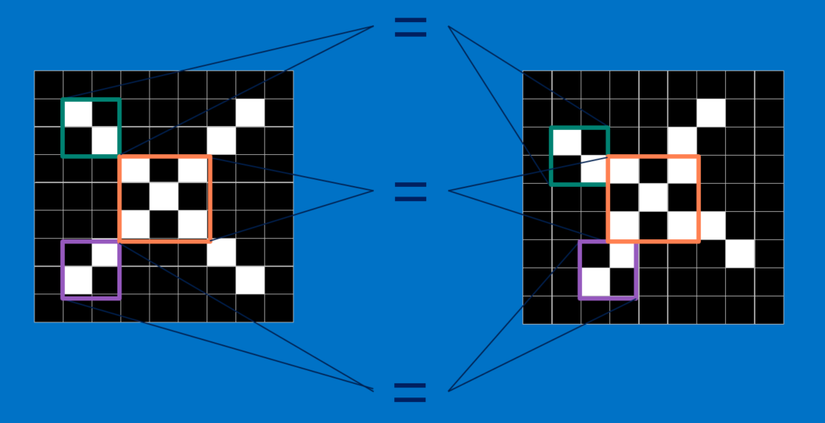
POOLING: Tầng tổng hợp, thông thường là Max pooling hoặc có thể là Average pooling dùng để giảm chiều của ma trận đầu vào.

FULLY CONECTED: Tầng kết nối hoàn toàn. Thông thường tầng này nằm ở sau cùng và kết nối với các đơn vị đại diện cho nhóm phân loại.

* 1. Nguyên lý hoạt động của từng tầng
     1. Feature



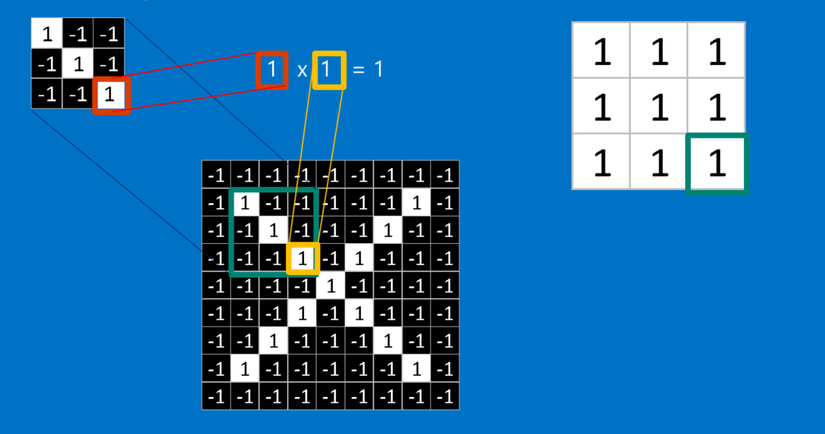
Một cách tiếp cận ngây thơ để giải quyết vấn đề này là lưu một hình ảnh của một X và của một O, và so sánh mỗi hình ảnh mới với hai hình mẫu đó để xem nó khớp với bên nào hơn. Điều khiến cho công việc này khó khăn là do máy tính chỉ hiểu các con số. Với máy tính, một hình ảnh trông giống như một mảng hai chiều các điểm ảnh (tưởng tượng nó như một bàn cờ khổng lồ) với một số trong mỗi ô. Trong ví dụ của chúng ta một điểm ảnh có giá trị 1 là trắng, và -1 là đen. Khi so sánh hai hình ảnh, nếu có giá trị điểm ảnh không trùng, thì những ảnh đó đối với máy tính là không khớp. Lý tưởng nhất là chúng ta muốn xem liệu chúng thuộc X hay O ngay cả khi chúng bị tịnh tiến, bị co, xoay hoặc biến dạng. Đây là khi ta cần CNNs.



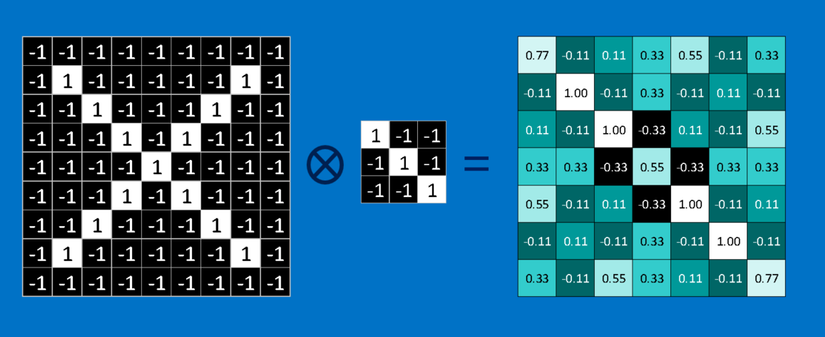
CNNs so sánh hình ảnh theo từng mảnh. Các mảnh mà nó tìm được gọi là các feature. Bằng cách tìm ở mức thô các feature khớp nhau ở cùng vị trí trong hai hình ảnh, CNNs nhìn ra sự tương đồng tốt hơn nhiều so với việc khớp toàn bộ bức ảnh.

2.2.2 Tích chập

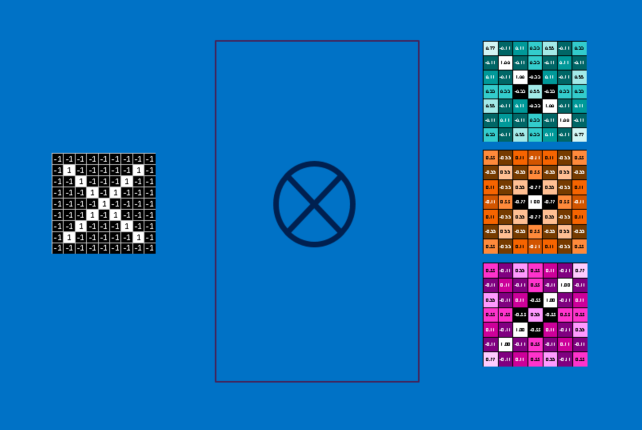
Khi xem một hình ảnh mới, CNN không biết chính xác nơi các feature này sẽ khớp nên nó sẽ thử chúng khắp mọi nơi, ở mọi vị trí có thể. Khi tính toán sự khớp của một feature trên toàn bộ ảnh, chúng ta làm thành một filter (bộ lọc). Phần toán ta sử dụng để làm điều này được gọi là tích chập, từ đó mà Mạng Nơ-ron Tích chập (Convolutional Neural Networks) có tên là như vậy.



Để tính toán sự khớp của một feature đối với một mảnh của hình ảnh, ta chỉ cần nhân mỗi điểm ảnh trong feature với giá trị của điểm ảnh tương ứng trong mảnh hình ảnh. Sau đó cộng tổng lại và chia cho số lượng điểm ảnh trong feature. Nếu cả hai điểm ảnh màu trắng (giá trị 1) thì 1 x 1 = 1. Nếu cả hai đều là màu đen, thì ( -1 ) x ( -1 ) = 1. Dù bằng cách nào, mỗi điểm ảnh mà khớp thì đều cho ra kết quả 1. Tương tự như vậy, bất kỳ cái nào không khớp đều ra -1. Nếu tất cả các điểm ảnh trong feature đều khớp, thì cộng tổng lại rồi chia cho số điểm ảnh thì sẽ ra là 1. Tương tự, nếu không có điểm ảnh nào trong feature khớp với mảnh hình ảnh, thì kết quả là -1.

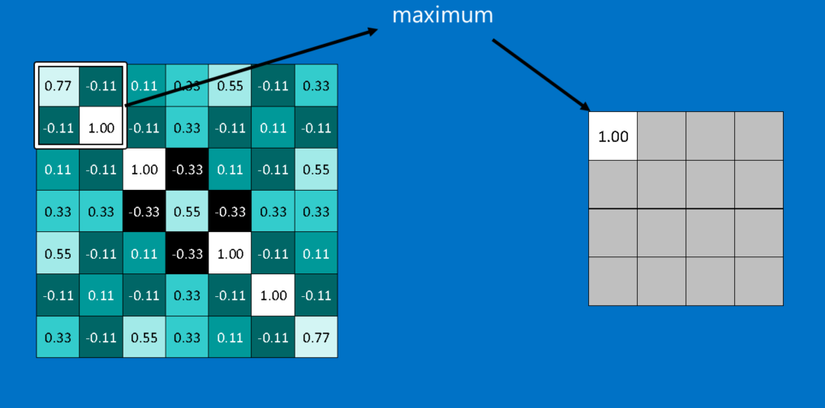


Để hoàn tất tích chập, chúng ta lặp lại quá trình này, xếp các feature với mọi mảnh hình ảnh có thể. Ta có thể lấy kết quả từ mỗi tích chập và tạo một mảng hai chiều mới từ đó, dựa vào vị trí của mỗi mảnh nằm trong hình ảnh. Bản đồ các phần khớp nhau này cũng là một phiên bản đã được lọc từ hình ảnh gốc. Nó là một bản đồ thể hiện nơi tìm thấy feature trong hình ảnh. Các giá trị gần 1 cho thấy sự khớp mạnh, các giá trị gần -1 cho thấy sự khớp mạnh với âm bản của feature, và các giá trị gần bằng 0 cho thấy không khớp với bất kỳ loại nào.



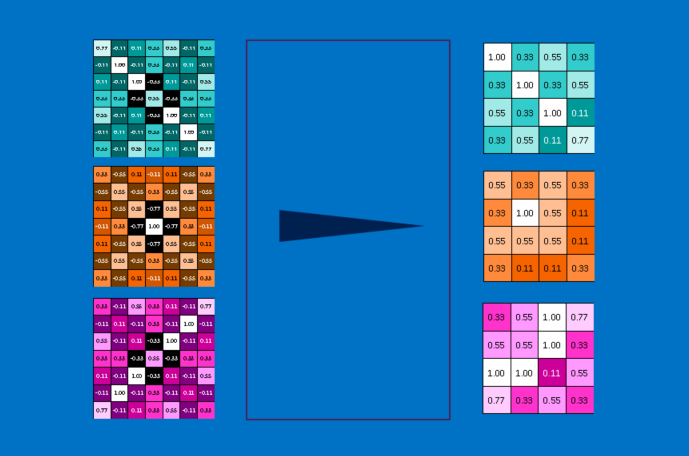
Bước tiếp theo là lặp lại toàn bộ quá trình chập cho từng feature khác. Kết quả là một tập hợp các hình ảnh đã được lọc, mỗi cái ứng với mỗi filter của chúng ta. Sẽ thuận tiện khi nhìn toàn bộ các thao tác tích chập như là một bước xử lý duy nhất.

* + 1. Pooling



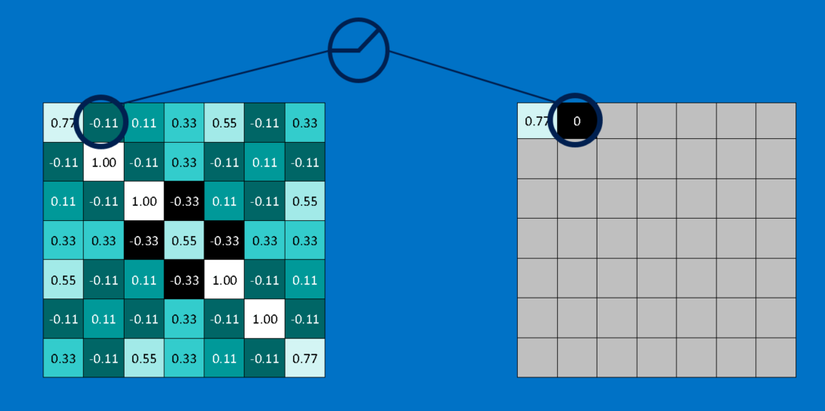
Một công cụ mạnh mẽ khác mà CNNs sử dụng được gọi là pooling. Pooling là một cách lấy những hình ảnh lớn và làm co chúng lại trong khi vẫn giữ các thông tin quan trọng nhất trong đó. Pooling chỉ dùng kiến thức toán của lớp hai. Nó bao gồm việc duyệt bước một ô vuông cửa sổ nhỏ dọc trên một hình ảnh và lấy giá trị lớn nhất từ cửa sổ ở mỗi bước. Trong thực tế, một cửa sổ có cạnh 2 hoặc 3 điểm ảnh và duyệt bước mỗi 2 điểm ảnh là được.

Sau khi pooling, một hình ảnh sẽ có khoảng một phần tư số điểm ảnh so với lúc bắt đầu. Vì nó giữ các giá trị lớn nhất từ mỗi cửa sổ, nó sẽ bảo toàn tính khớp của mỗi feature bên trong cửa sổ. Nghĩa là nó không quan tâm quá nhiều về vị trí chính xác nơi feature khớp, miễn là nó khớp ở chỗ nào đó trong cửa sổ. Kết quả là CNNs có thể tìm xem liệu một feature có nằm trong hình ảnh mà không cần lo nó nằm ở đâu. Điều này giúp giải quyết vấn đề của máy tính là quá trực nghĩa.

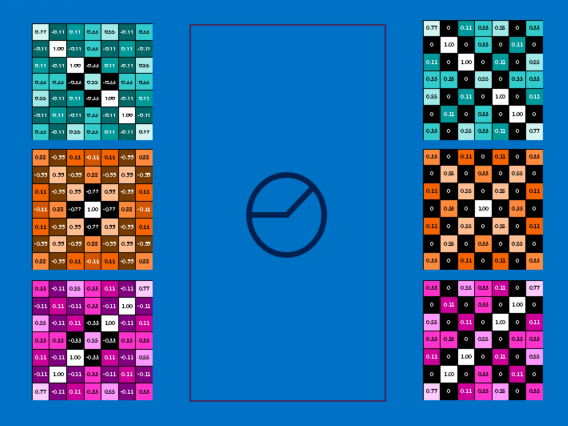


Một layer pooling là hoạt động thực hiện pooling trên một hình ảnh hoặc một tập các hình ảnh. Đầu ra sẽ có cùng số lượng hình ảnh, nhưng mỗi cái sẽ có điểm ảnh ít hơn. Điều này cũng rất hữu ích trong việc quản lý tải trọng tính toán. Hạ một tấm ảnh 8 megapixel xuống còn 2 megapixel sẽ giúp mọi xử lý tải về trở nên dễ dàng.

* + 1. **Rectified Linear Units (**ReLU)

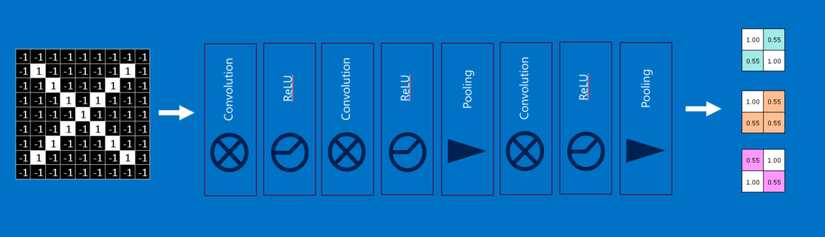


Đóng vai trò nhỏ nhưng quan trọng trong quá trình này là Rectified Linear Unit hoặc ReLU. Toán của cái này cũng rất đơn giản -- bất cứ nơi nào có số âm, hoán đổi nó với 0. Điều này giúp CNN giữ vững sự tin cậy toán học bằng cách giữ các giá trị đã được học khỏi bị mắc kẹt gần 0 hoặc bị thổi bay về vô tận. Đó là thứ dầu mỡ bôi trơn CNNs- không hấp dẫn mấy, nhưng nếu không có nó, chúng sẽ không đi xa hơn được.

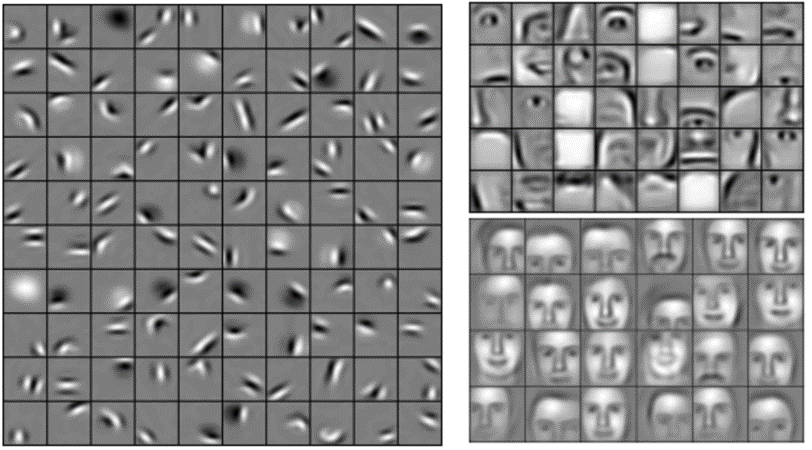


Đầu ra của một layer ReLU có kích thước giống với đầu vào, chỉ là tất cả các giá trị âm được loại bỏ.

2.2.4 Học sâu (Deep learning)



Bạn có thể thấy rằng các đầu vào cho mỗi layer (mảng hai chiều) trông rất giống đầu ra (cũng mảng hai chiều). Vì vậy, chúng ta có thể xếp chồng chúng như những mẩu Lego. Những hình ảnh gốc được filtered (lọc), rectified (tinh chỉnh) và pooled (gộp lại) để tạo ra một tập các hình ảnh đã được lọc và thu gọn. Chúng có thể được lọc và thu gọn lại liên tục. Mỗi lần như vậy, các feature trở nên lớn hơn và phức tạp hơn, và hình ảnh trở nên nhỏ gọn hơn. Điều này giúp các layer thấp hơn đại diện cho các khía cạnh đơn giản của hình ảnh, chẳng hạn như các cạnh và các điểm sáng. Các layer cao hơn có thể đại diện cho các khía cạnh tinh vi hơn của hình ảnh, chẳng hạn như các hình khối (shape) và các hình mẫu (pattern). Dễ thấy được điều này. Ví dụ, trong một CNN được đào tạo nhận diện mặt người, các lớp cao nhất đại diện cho các mẫu giống khuôn mặt người rất rõ ràng.

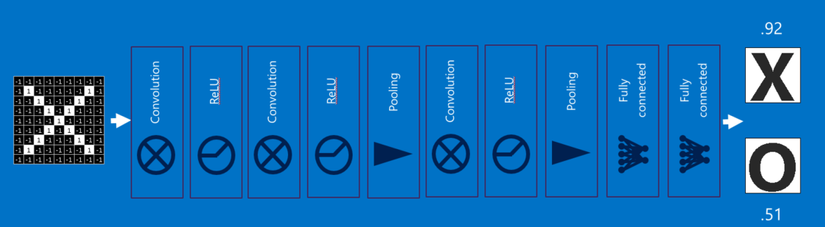


2.2.5 Các layer được kết nối đầy đủ (Fully connected layers)



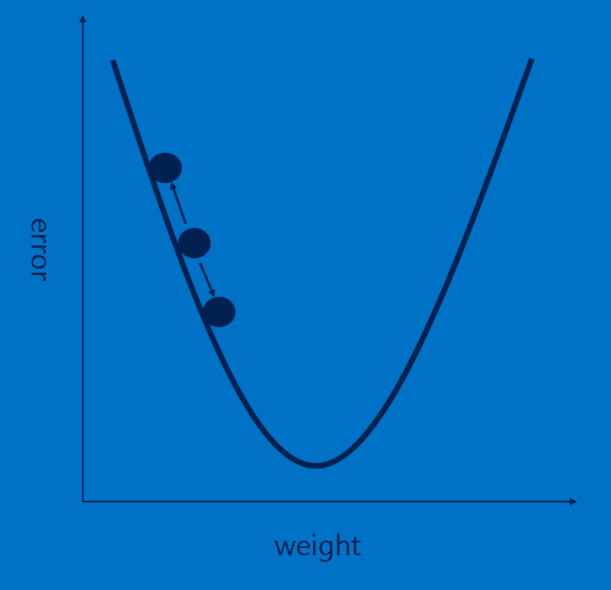
Các CNN còn có một thứ vũ khí nữa. Các layer được kết nối đầy đủ lấy các hình ảnh đã lọc ở cấp cao và chuyển chúng thành các phiếu bầu (vote). Trường hợp của chúng ta, chỉ phải quyết định giữa hai loại, X và O. Các layer được kết nối đầy đủ là một khối chính của mạng nơ-ron truyền thống. Thay vì coi đầu vào như một mảng hai chiều, chúng được coi như một list đơn và tất cả đều được xử lý giống nhau. Mỗi giá trị bỏ phiếu riêng bầu cho hình ảnh hiện tại là X hay O. Tuy nhiên, quá trình này không hoàn toàn dân chủ. Một số giá trị cho biết hình ảnh là một X tốt hơn nhiều so với những giá trị khác, và một số lại đặc biệt tốt khi cho biết hình ảnh là một O. Chúng có giá trị bỏ phiếu lớn hơn so với những cái khác. Chúng được thể hiện như trọng số (weight), hoặc là mức độ kết nối, giữa mỗi giá trị (trong list) và mỗi loại (X hay O).

Khi một hình ảnh mới được đưa vào CNN, nó sẽ thấm qua các lớp thấp hơn cho đến khi cuối cùng nó đạt đến các layer được kết nối đầy đủ. Sau đó, một cuộc bầu chọn được tổ chức. Câu trả lời có nhiều phiếu nhất sẽ thắng và được tuyên bố là thuộc loại nào (X hay O).



Các layer được kết nối đầy đủ, giống như các layer khác, có thể được xếp chồng lên nhau vì đầu ra của chúng (một list các vote) trông giống với đầu vào (một list các giá trị). Trong thực tế, một vài các layer được kết nối đầy đủ thường được xếp chồng lên cùng với nhau, với mỗi layer trung gian bỏ phiếu cho các loại "ẩn". Mỗi layer bổ sung sẽ cho phép mạng học các tổ hợp tinh vi hơn của các feature và sẽ giúp nó đưa ra quyết định tốt hơn.

2.2.6 Lan truyền ngược (Backpropagation)



Câu chuyện của chúng ta đang được lấp đầy, nhưng nó vẫn có một lỗ hổng lớn - Các feature đến từ đâu? và Làm thế nào để tìm trọng số trong các layer được kết nối đầy đủ? Nếu chúng phải chọn bằng tay, CNN hẳn sẽ kém phổ biến hơn giờ. May mắn thay, một chút ma thuật của học máy (machine learning) gọi là lan truyền ngược (backpropagation) làm việc này giúp chúng ta.

Để sử dụng lan truyền ngược, chúng ta cần một tập các hình ảnh mà chúng ta đã biết câu trả lời. Điều này có nghĩa rằng một số thanh niên kiên nhẫn sẽ lật qua hàng ngàn hình ảnh và gán cho chúng một nhãn X hoặc O. Chúng ta sử dụng chúng với một CNN chưa được huấn luyện, nghĩa là mỗi điểm ảnh của mỗi feature và mỗi trọng số trong mỗi các layer được kết nối đầy đủ được đặt một giá trị ngẫu nhiên. Sau đó ta bắt đầu đưa các hình ảnh đi qua nó, từng cái một.

Mỗi hình ảnh CNN xử lý cho kết quả là một phiếu bầu (vote). Số lượng nhầm lẫn trong phiếu bầu, gọi là sai số (error), cho ta biết mức độ tốt của các feature và trọng số chúng ta đang có. Các feature và trọng số sau đó có thể được điều chỉnh để làm cho các sai số ít hơn. Mỗi giá trị được điều chỉnh cao hơn một chút và thấp hơn một chút, và tính toán các sai số mới mỗi lần. Điều chỉnh nào làm cho các sai số ít đi sẽ được giữ lại. Sau khi làm điều này cho mỗi điểm ảnh feature trong mỗi layer tích chập và mỗi trọng số trong mỗi các layer được kết nối đầy đủ, các trọng số mới sẽ đưa ra câu trả lời tốt hơn một chút cho hình ảnh đó. Điều này sau đó được lặp đi lặp lại với mỗi hình ảnh tiếp theo trong tập các hình ảnh đã được gắn nhãn. Các khuyết tật xảy ra trong một hình ảnh nhanh chóng bị lãng quên, nhưng hình mẫu (pattern) xuất hiện trong rất nhiều hình ảnh được chế biến thành các feature và các trọng số kết nối. Nếu bạn có đủ các hình ảnh đã được gắn nhãn, chúng sẽ hình thành một tập ổn định hoạt động khá tốt trên một loạt các trường hợp.

1. YOLO
   1. Giới thiệu về YOLO và Darknet

YOLO trong object detection có nghĩa là “You only look once”. Tức là chúng ta chỉ cần nhìn 1 lần là có thể phát hiện ra vật thể.

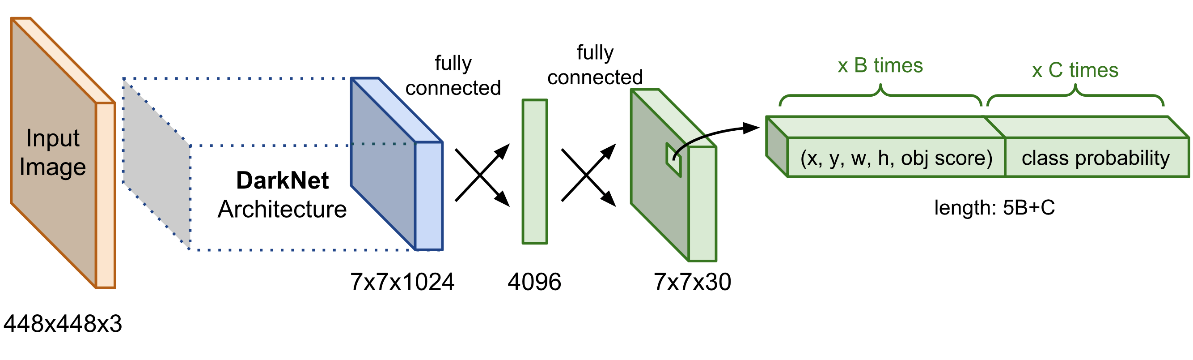
Thật vậy, về độ chính xác thì YOLO có thể không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ gần như real time mà độ chính xác không quá giảm so với các model thuộc top đầu.

YOLO là thuật toán object detection nên mục tiêu của mô hình không chỉ là dự báo nhãn cho vật thể như các bài toán classification mà nó còn xác định location của vật thể. Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất một nhãn cho một bức ảnh.

[Darknet](https://pjreddie.com/darknet/) là một framework open source chuyên về Object Detection được viết bằng ngôn ngữ C và CUDA. Darknet dùng để huấn luyện các mô hình YOLO một cách nhanh chóng, dễ sử dụng.

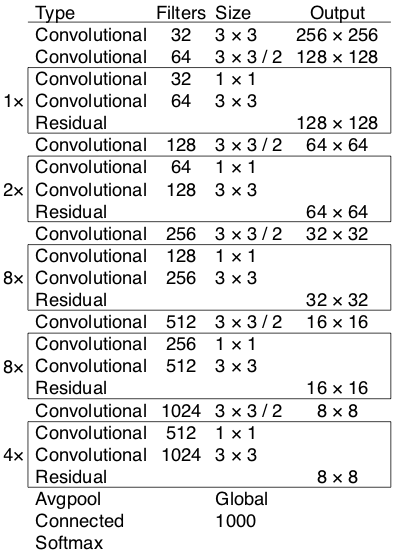
* 1. Kiến trúc mạng YOLO

Kiến trúc YOLO bao gồm: base network là các convolution layer làm nhiệm vụ trích xuất đặc trưng của ảnh. Phần phía sau là những Extra Layers (fully connected layer) được áp dụng để phát hiện vật thể trên feature map của base network. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau



**Hình 1:** Sơ đồ kiến trúc mạng YOLO. Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

Trong YOLO version 3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mẹo này có tác dụng giảm thiểu số lượng tham số cho mô hình.



Các layer trong mạng darknet-53.

Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện.

Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là 416x416 và 608x608. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map.

Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input 416x416 thì feature map có các kích thước là 13x13, 26x26 và 52x52. Và khi input là 608x608 sẽ tạo ra feature map 19x19, 38x38, 72x72.

* 1. Output của YOLO

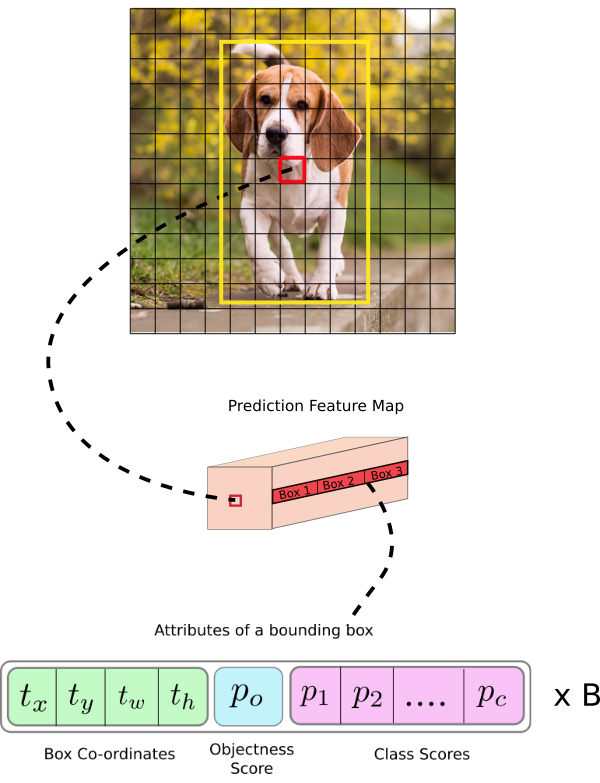
Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

Trong đó

* là xác suất dự báo vật thể xuất hiện trong bounding box.
* ⟨tx,ty,tw,th⏟bounding box⟩ giúp xác định bounding box. Trong đó tx,ty là tọa độ tâm và tw,th là kích thước rộng, dài của bounding box.
* ⟨p1,p2,…,pc⏟scores of c classes⟩ là véc tơ phân phối xác suất dự báo của các classes.

Việc hiểu output khá là quan trọng để chúng ta cấu hình tham số chuẩn xác khi huấn luyện model qua các open source như darknet. Như vậy output sẽ được xác định theo số lượng classes theo công thức (n\_class+5). Nếu huấn luyện 80 classes thì bạn sẽ có output là 85. Trường hợp bạn áp dụng 3 anchors/cell thì số lượng tham số output sẽ là:

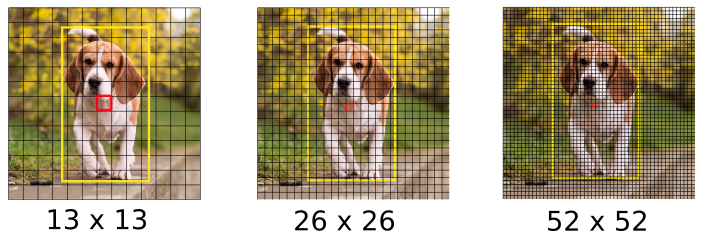
(n\_class+5)×3=85×3=255



**Hình 3:** Kiến trúc một output của model YOLO. Hình ảnh gốc là một feature map kích thước 13x13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình.

3.4 Dự báo trên nhiều feature map

Cũng tương tự như SSD, YOLOv3 dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ.



**Hình 4**: Các feature maps của mạng YOLOv3 với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52.

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 featue map x 3 anchor box).

Đồng thời trên một feature map hình vuông S x S, mô hình YOLOv3 sinh ra một số lượng anchor box là: S x S x 3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

(13×13+26×26+52×52)×3=10647(anchor boxes)

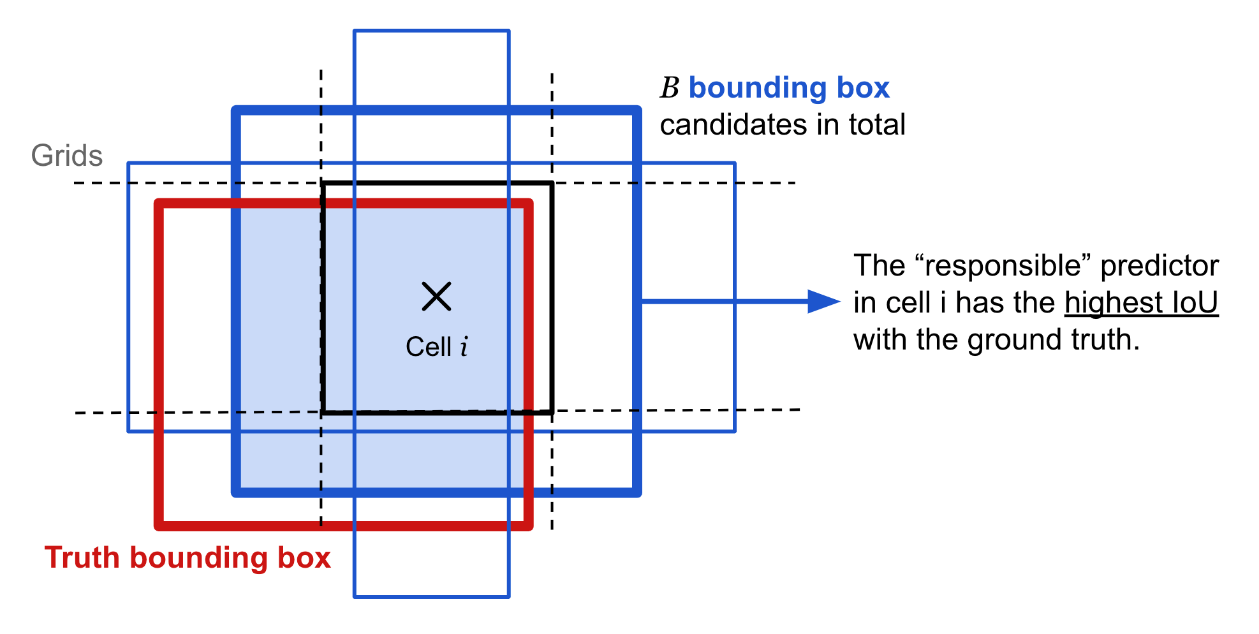
Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

Một số lưu ý khi huấn luyện YOLO:

* Khi huấn luyện YOLO sẽ cần phải có RAM dung lượng lớn hơn để save được 10647 bounding boxes như trong kiến trúc này.
* Không thể thiết lập các batch\_size quá lớn như trong các mô hình classification vì rất dễ Out of memory. Package darknet của YOLO đã chia nhỏ một batch thành các subdivisions cho vừa với RAM.
* Thời gian xử lý của một step trên YOLO lâu hơn rất rất nhiều lần so với các mô hình classification. Do đó nên thiết lập steps giới hạn huấn luyện cho YOLO nhỏ. Đối với các tác vụ nhận diện dưới 5 classes, dưới 5000 steps là có thể thu được nghiệm tạm chấp nhận được. Các mô hình có nhiều classes hơn có thể tăng số lượng steps theo cấp số nhân tùy bạn.
  1. Anchor box

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Sau này thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO:

* Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth bounding box là cao nhất.

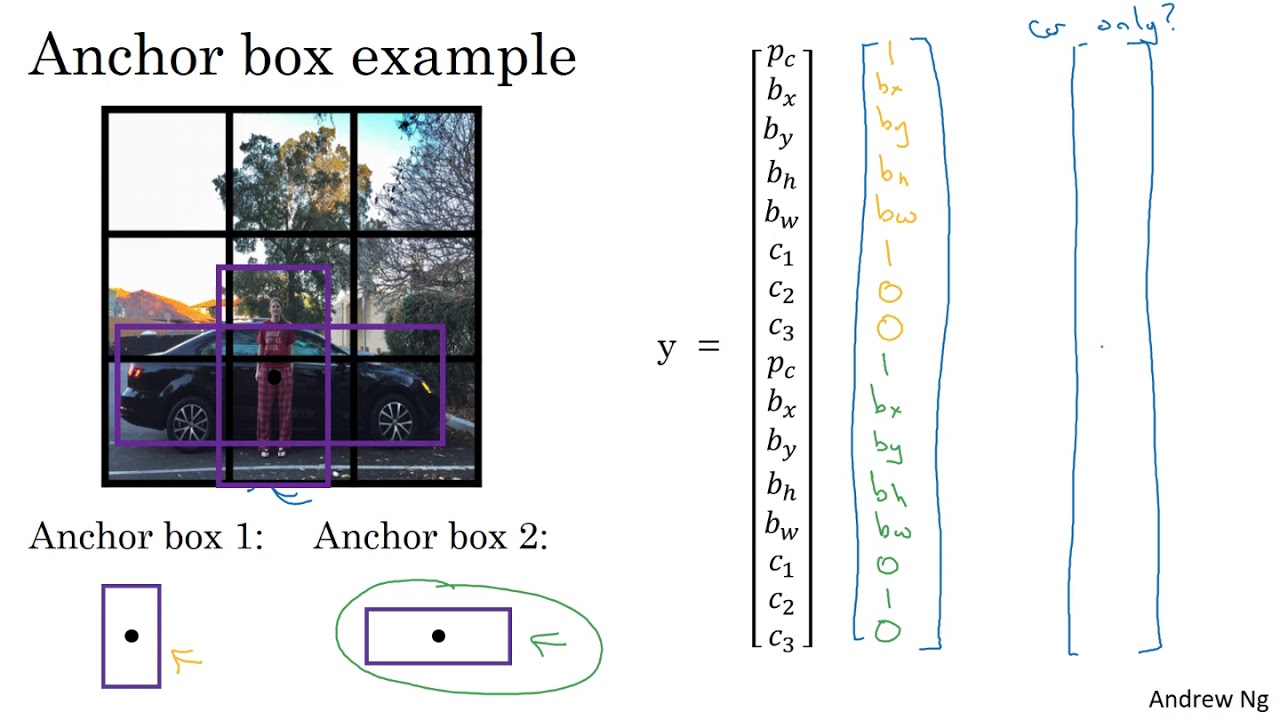


**Hình 5**: Xác định anchor box cho một vật thể. Từ Cell i ta xác định được 3 anchor boxes viền xanh như trong hình. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất.

* Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map mà chứa điểm mid point của vật thể. Chẳng hạn như hình chú chó trong hình 3 sẽ được phân về cho cell màu đỏ vì điểm mid point của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là (cell, anchor box). Không chỉ riêng mình cell hoặc chỉ mình anchor box.

Một số trường hợp 2 vật thể bị trùng mid point, mặc dù rất hiếm khi xảy ra, thuật toán sẽ rất khó xác định được class cho chúng.



**Hình 6**: Khi 2 vật thể người và xe trùng mid point và cùng thuộc một cell. Thuật toán sẽ cần thêm những lượt tiebreak để quyết định đâu là class cho cell.

* 1. Hàm loss function

Hàm loss function của YOLO chia thành 2 phần:  (localization loss) đo lường sai số của bounding box và  (confidence loss) đo lường sai số của phân phối xác suất các classes.

* : Hàm indicator có giá trị 0,1 nhằm xác định xem cell i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và 0 nếu không chứa.
* : Cho biết bounding box thứ j của cell i có phải là bouding box của vật thể được dự đoán hay không? (xem hình 5).
* : Điểm tin cậy của ô i, P(contain object) \* IoU (predict bbox, ground truth bbox).
* : Điểm tự tin dự đoán.
* : Tập hợp tất cả các lớp.
* : Xác suất có điều kiện, có hay không ô i có chứa một đối tượng của lớp c∈C.
* : Xác suất có điều kiện dự đoán.

Có thể ban đầu công thức trên khá khó hiểu với người bắt đầu. Chúng ta hãy hiểu đơn giản hóa mục đích của chúng:

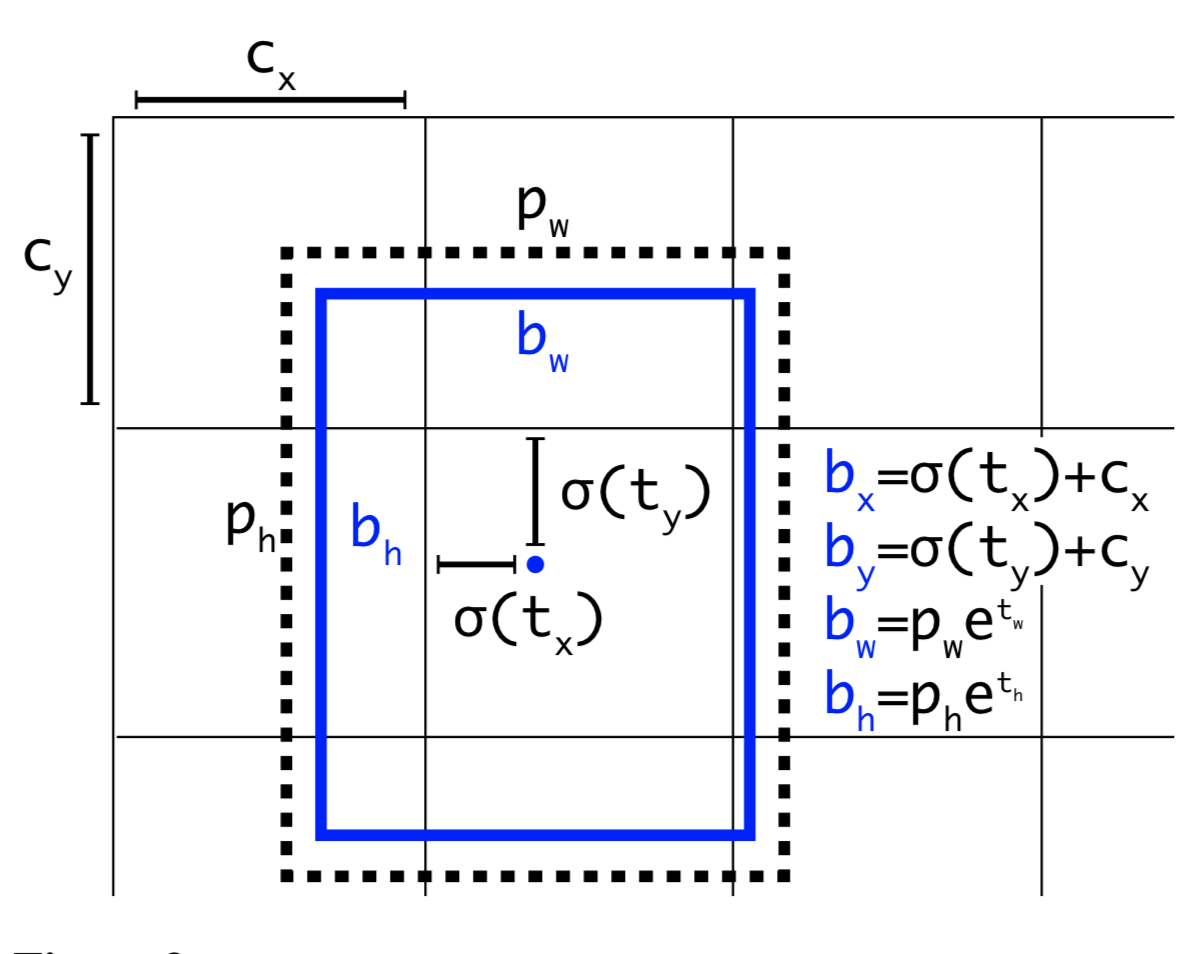
* là hàm mất mát của bounding box dự báo so với thực tế.
* là hàm mất mát của phân phối xác suất. Trong đó tổng đầu tiên là mất mát của dự đoán có vật thể trong cell hay không? Và tổng thứ 2 là mất mát của phân phối xác suất nếu có vật thể trong cell
  1. Dự báo bounding box

Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell.

YOLOv2 vả YOLOv3 dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (pw,ph) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (cx,cy), mô hình dự đoán 4 tham số (tx,ty,tw,th) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (bx,by) và kích thước (bw,bh) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

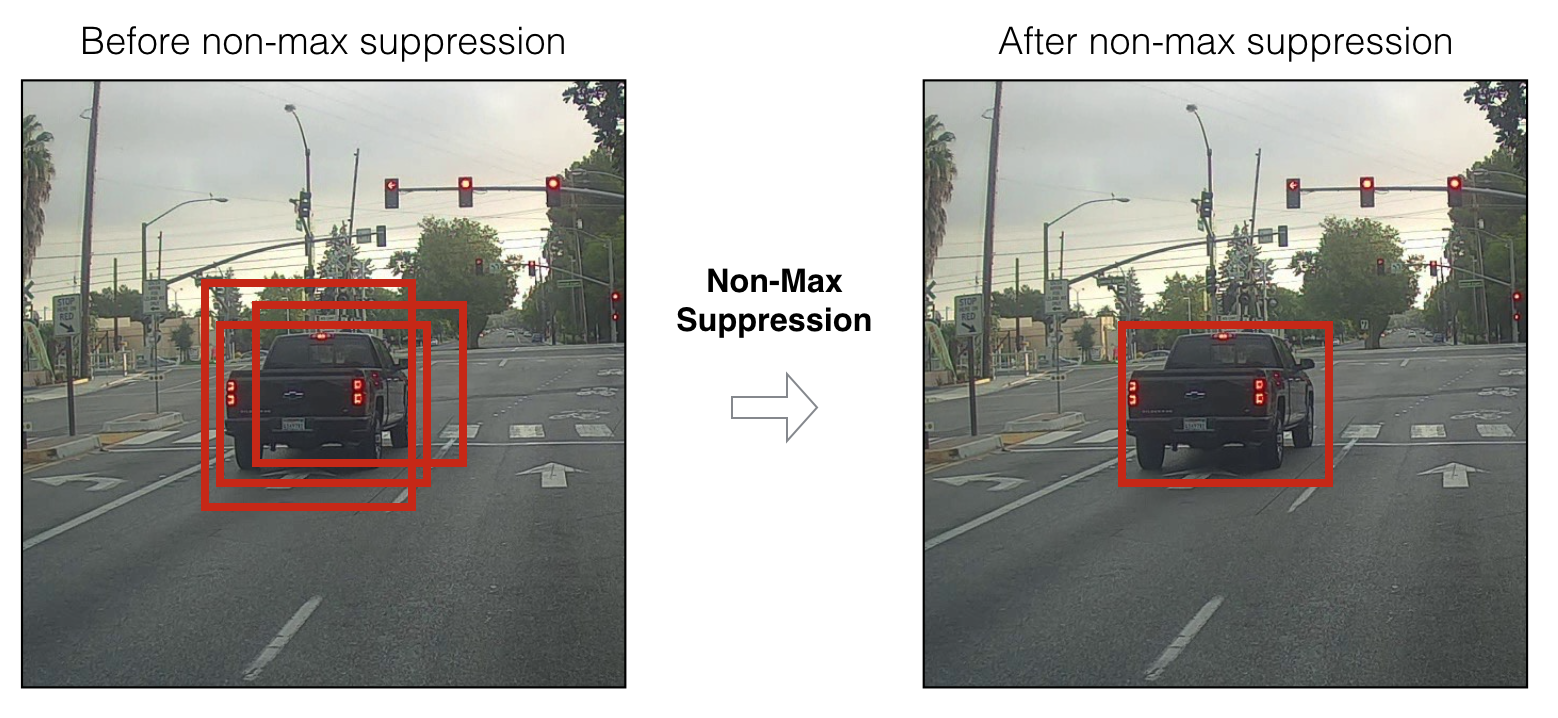
Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0, 1]. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.



**Hình 7:** Công thức ước lượng bounding box từ anchor box. Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (pw,ph). Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLO version 1.

# 3.8 Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap là rất cao. Trong trường hợp đó YOLO sẽ cần đến non-max suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể.



**Hình 8**: non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm xuống còn một bounding box cuối cùng.

Các bước của non-max suppression:

* Step 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.
* Step 2: Đối với các bouding box giao nhau, non-max suppression sẽ lựa chọn ra một bounding box có xác xuất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại.
* Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding có có xác xuất thấp hơn và giữ lại bouding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

1. Giao thức MQTT
   1. Giới thiệu MQTT

MQTT viết tắt của MQ Telemetry Transport (trước đó MQTT được định nghĩa là Message Queuing Telemetry Transport), là giao thức ở tầng ứng dụng chạy trên nền TCP thuộc tập giao thức TCP/IP.

   Để giao thức này hoạt động được, cần có một thành phần trung tâm gọi là Broker và các thiết bị/ứng dụng khách còn được gọi là các Client, các Client sẽ kết nối đến Broker. Broker và Client sẽ “nói chuyện” với nhau thông qua các gói tin MQTT được xây dựng dựa theo chuẩn OASIS. Tiêu chuẩn này còn định nghĩa các mức quality of service để đảm bảo độ tin cậy khi truyền nhận gói tin, các kịch bản always connected (luôn giữ kết nối) hoặc sometimes connected (thỉnh thoảng kết nối); khả năng mở rộng để hỗ trợ kết nối số lượng lớn thiết bị…

Kiến trúc

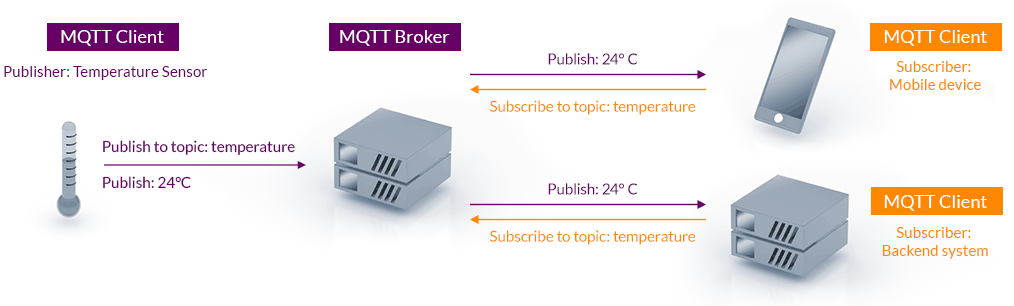
Giao thức MQTT có kiến trúc theo mô hình Publish/Subscribe giúp truyền tải dữ liệu giữa các thiết bị, ứng dụng với nhau. Dữ liệu trong giao thức này là một chuỗi nhị phân (binary) chứ không phải chuỗi văn bản (text string), được định dạng theo gói tin command hoặc gói tin command acknowledgement.

Các thành phần lõi trong kiến trúc của MQTT:

– MQTT Broker: Được cung cấp dưới dạng mã nguồn mở hoặc các phiên bản thương mại, có thể đi kèm với các dịch vụ điện toán đám mây. Công việc của Broker là lọc các tin nhắn dựa trên topic, sau đó phân phối các tin nhắn đến các thiết bị/ứng dụng đã đăng ký topic đó. Các bạn có thể tham khảo một số MQTT Broker như: HiveMQ , Mosquitto, MQTTRoute, Jmqtt, …

– MQTT Client: Là các thiết bị/ứng dụng Client kết nối đến Broker để thực hiện truyền nhận dữ liệu. Hiện nay có rất nhiều mã nguồn mở MQTT Client được viết dưới nhiều ngôn ngữ khác nhau như HiveMQ MQTT Client được phát triển dựa trên ngôn ngữ Java, Eclipse Paho dựa trên C/C++, Python, …

– Topic: Mỗi MQTT Client thực hiện truyền/nhận dữ liệu với nhau thông qua các Topic được quản lý bởi Broker. Một Client đăng ký nhận dữ liệu được gọi là một Subcriber còn một Client gửi dữ liệu đi được gọi là một Publisher. Để nhận dữ liệu từ Publisher, đầu tiên Subcriber phải subscribe (đăng ký theo dõi) đến một Topic, sau đó bất cứ Client nào publish dữ liệu đến đúng Topic, thì Broker sẽ lọc và chuyển tiếp gói tin đến đúng Subscriber đó. Một Client có thể subscribe hoặc publish đến nhiều Topic khác nhau.



4.2 QoS

MQTT có 3 mức Quality of Service (QoS) được xác định đảm bảo độ tin cậy trong việc gửi gói tin.

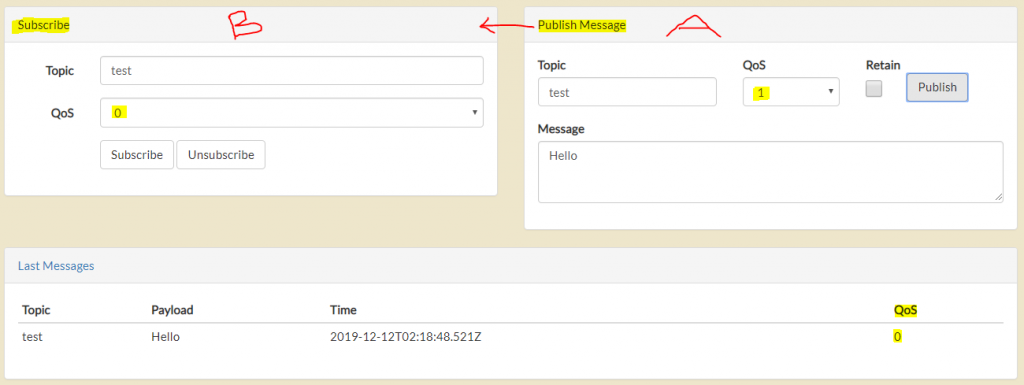
QoS = 0 (at-most-once): mỗi gói tin được gửi đến đích tối đa một lần => như vậy gói tin truyền thành công tối thiểu sẽ là “0 lần” => nghĩa là đối với trường hợp này có thể xảy ra trường hợp mất gói tin và chỉ phù hợp với những ứng dụng gửi nhiều gói tin, trong đó có thể chấp nhận được khi mất đi một vài gói.

QoS = 1 (at-least-one): mỗi gói tin được gửi đến đích tối thiểu là một lần => như vậy một gói tin truyền đến đích tối đa có thể là “vô số lần” => nghĩa là có thể xảy ra trường hợp bị trùng lặp gói tin, về cơ bản QoS = 1 sẽ giải quyết được vấn đề mất gói của QoS = 0 nhưng phải chấp nhận việc có thể bị lặp gói.

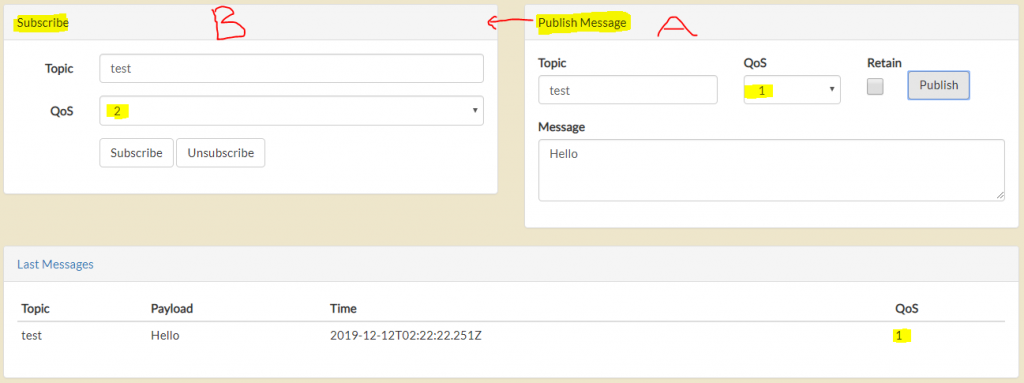
QoS = 2 (Exactly-once): mỗi gói tin sẽ được gửi đến đích duy nhất là một lần => không bị mất gói cũng như bị lặp gói. Đây là mức độ đảm bảo nhất, dùng trong các hệ thống cần đảm bảo nghiêm ngặt về mức độ đảm bảo của mỗi gói tin khi truyền nhận.

Yếu tố gây nhầm lẫn: Việc truyền nhận dữ liệu trong giao thức MQTT được thực hiện giữa publisher và subscriber nên ngữ nghĩa về “đảm bảo gói tin” trong bài viết này được xét đến khi truyền một gói tin từ publisher đến subscriber. Nhưng có một yếu tố quan trọng là 2 đối tượng client này không hề giao tiếp trực tiếp với nhau mà lại thông qua một broker trung tâm. Vì thế QoS được cấu hình bởi client chính xác phải là QoS của publisher đối với broker và QoS của subscriber đối với broker chứ không phải giữa 2 client với nhau, mỗi client đều có quyền lựa chọn QoS của mình để giao tiếp với broker, nhưng mức độ đảm bảo của gói tin để truyền thành công từ publisher->subscriber thì chỉ có 1 mà thôi.  
Xét một ví dụ cụ thể: khi một publisher A gửi một gói tin có QoS gọi là QA đến một subscriber có QoS là QB , vậy thì sự khác nhau giữa 2 giá trị QA và QB sẽ dẫn đến mức độ đảm bảo gói tin khi truyền từ A->B sẽ có QoS là bao nhiêu?

Trường hợp 1: QA>QB (vd: QA=1, QB=0): kết quả gói có QoS =QB = 0



Trường hợp 2: QA<QB (vd: QA=1, QB=2): kết quả gói tin có QoS = QA = 1



Giải thích:

Trong trường hợp 1, A khi gửi tin đã cấu hình QoS = 1 với mong muốn gói tin của mình khi gửi đến B không bị mất, tuy nhiên điều này chỉ đúng trong quá trình gửi từ A đến Broker, trong khi đó B chỉ sử dụng QoS = 0 (có thể để ưu tiên tốc độ truyền nhận) nên Broker sẽ chuyển tiếp gói tin đến B với QoS=0, nghĩa là gói tin đó hoàn toàn có thể bị mất trong quá trình chuyển tiếp này

=> Gói tin gửi từ A->B có QoS=0

Trong trường hợp 2, B khi đăng ký nhận tin đã cấu hình QoS=2 với mong muốn là tất cả gói tin khi truyền từ A đến topic chung đều có thể nhận được một lần duy nhất (không mất , không lặp). Tuy nhiên nếu xét cả quá trình từ A->B, vì A truyền  gói tin đến broker với QoS=1 thì có thể bị lỗi lặp gói tin, nên chắc chắn gói tin lặp này cũng sẽ được chuyển tiếp đến B đẫn đến B cũng sẽ bị lặp gói tin dù QoS =2.

=> Gói tin gửi từ A->B có QoS=1

Kết luận: QoS có tính chất Downgrade, nghĩa là QoS tổng hợp sẽ có giá trị bằng với QoS nhỏ nhất được cấu hình giữa A và B.

QoS (A->B) = min (QA,QB)

4.3 Retain

Retain là một cờ (flag) được gắn cho một message của giao thức MQTT. Retain chỉ nhận giá trị 0 hoặc 1. Nếu retain =1, broker sẽ lưu lại message cuối cùng của 1 topic kèm theo mức QoS tương ứng. Khi client bắt đầu subscribe topic có message được lưu lại đó, client ngay lập tức nhận được message

4.4 LWT

Gói tin LWT (last will and testament) không thực sự biết được Client có trực tuyến hay không, cái này do gói tin KeepAlive đảm nhận. Tuy nhiên gói tin LWT như là thông tin điều gì sẽ xảy đến sau khi thiết bị ngoại tuyến.

4.5 Ưu điểm

MQTT là giao thức 2 chiều, các node không chỉ truyền mà còn có thể dễ dàng đăng ký nhận tin một cách thụ động theo cơ chế publish-subscribe (với HTTP thì node không thể nhận tin thụ động được mà luôn luôn phải khởi tạo kết nối trước với server)

MQTT hỗ trợ cơ chế 1-n, nghĩa là một node có khả năng gửi tin đến rất nhiều node khác chỉ trong một lần gửi hoặc ngược lại (HTTP chỉ hỗ trợ 1-1 giữa client và server theo cơ chế request-response)

MQTT là một “lightweight protocol”, với ưu điểm gói tin nhẹ, không chứa nhiều dữ liệu dư thừa nên có thể truyền nhận dữ liệu một cách mượt mà trong điều kiện bị giới hạn về băng thông đường truyền.

MQTT hỗ trợ cấu hình QoS còn HTTP thì không.

CHƯƠNG III: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DẠNG VÀ TÍN HIỆU BÁO CHÁY

* 1. Lưu đồ thuật toán của hệ thống

Thu ảnh đầu vào từ camera

Tiền xử lý ảnh

Trích chọn đặc trưng

Dự đoán nhãn và tọa độ bouding box

Label= fire

Vẽ bouding box và phát cảnh báo

S

Đ

2. Training

2.1 Chuẩn bị dữ liệu

2.1.1 Đánh giá bộ dữ liệu

- Bộ dữ liệu gồm 2190 tấm ảnh về lửa

-Ảnh lửa được chụp từ nhiều vị trí khác nhau nên bắt được các hình dạng cũng như kích thước lửa

-Ảnh được chụp trong các mốc thời gian trong ngày nên độ sáng tương đối đa dạng

- Một số hình ảnh được thêm vật gây nhiễu để quá trình nhận dạng hạn chế đưa cảnh báo sai

- Ảnh có độ rõ nét cao nên cần phải làm mờ để tăng độ đa dạng

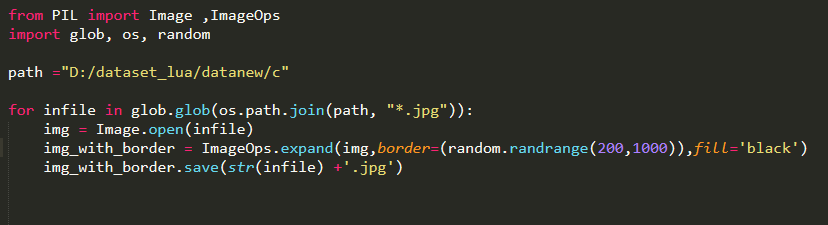
- Bộ dữ liệu tương đối tốt nhưng cần áp dụng các phương pháp tăng độ đa dạng của bộ dữ liệu lên để đưa ra kết quả chính xác nhất

2.1.2 Các phương pháp tăng độ đa dạng của dữ liệu

-Đa dạng về kích thước của lửa

Ngoài hình thức chụp ảnh ở các vị trí khác nhau (xa ,gần,góc độ,…) ta còn có thể dùng các giải thuật để thay đổi kích thước của lửa:

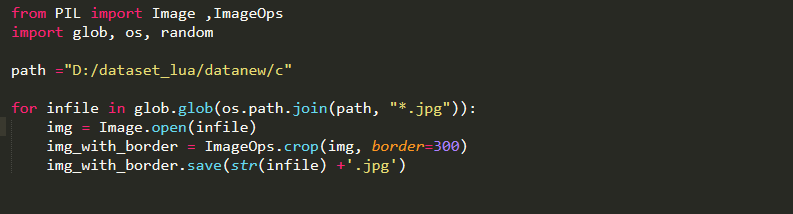
* Cách 1: Thu nhỏ kích thước biển bằng cách thêm biên kích thước ngẫu nhiên vào ảnh gốc, sau đó resize ảnh bằng kích thước ảnh ban đầu.





ảnh minh họa trước và sau khi thêm biên

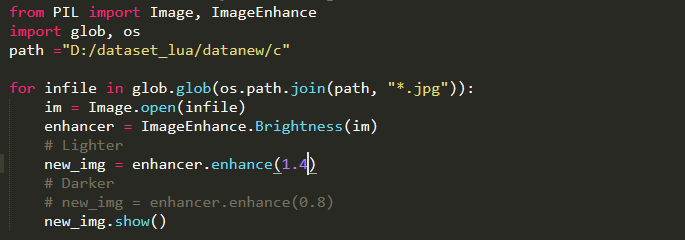
* Cách 2: Crop ảnh chứa biển số với kích thước ngẫu nhiên, sau đó resize ảnh bằng kích thước ảnh ban đầu.





Ảnh trước và sau khi crop

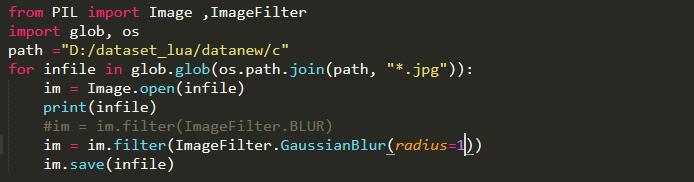
-Thay đổi độ sáng của ảnh





Ảnh trước và sau khi thay đổi độ sáng

- Làm mờ ảnh

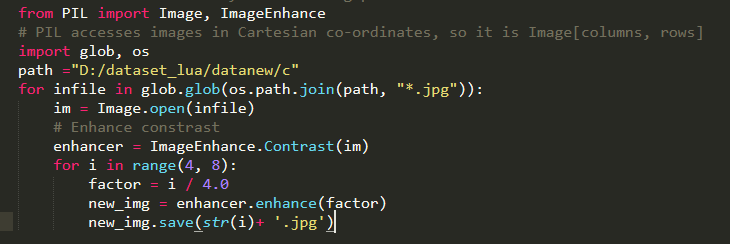


Đoạn chương trình duyệt qua các ảnh và làm mờ



Ảnh càng mờ khi tăng tham số radius

-Độ tương phản



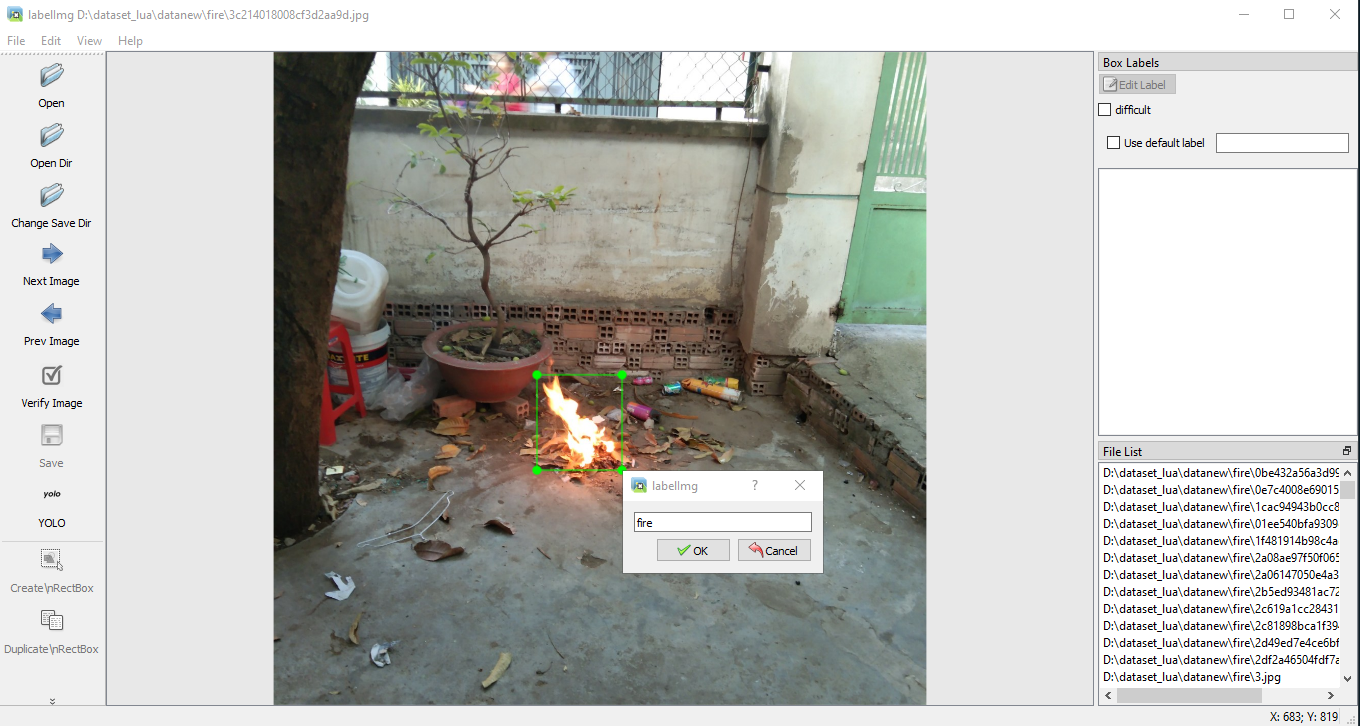
Đoạn chương trình làm tăng giảm độ tương phản của ảnh



Ảnh lửa trước và sau khi tăng độ tương phản

2.1.3 Gán nhãn dữ liệu

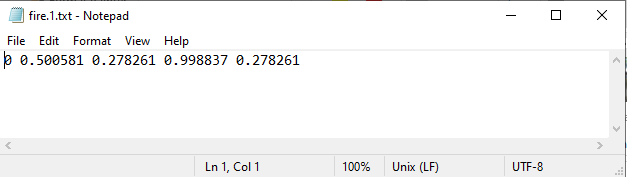
Dùng tool gán nhãn labelImage để gán nhãn cho đối tượng



Xác định vùng ảnh chứa lửa và gán nhãn

**LabelImg** hỗ trợ gán nhãn trên cả 2 định dạng PASCAL VOC và YOLO với phần mở rộng file annotation tương ứng là .xml và .txt.

Trong bài toán sử dụng mô hình YOLO, đầu ra là file annotation dưới dạng .txt.



Nội dung file annotation

Mỗi dòng trong một file annotation bao gồm: <object-class> <x> <y> <width> <height>.

Trong đó: <x> <y> <width> <height> tương ứng là tọa độ trung tâm và kích thước của đối tượng. Các giá trị này đã được chuẩn hóa lại, do vậy giá trị luôn nằm trong đoạn [0,1]. object-class là chỉ số đánh dấu các classes.

Lưu ý: Với bài toán có nhiều nhãn, nhiều người cùng gán nhãn thì cần thống nhất với nhau trước về thứ tự nhãn. Nguyên nhân do trong file annotation chỉ lưu chỉ số (0,1,3,4,…) của nhãn chứ không lưu tên nhãn.

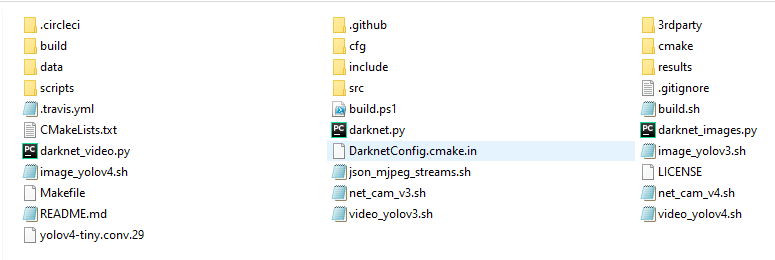
Sau khi gán nhãn xong các bạn để file annotation và ảnh tương ứng vào cùng một thư mục.

* 1. Cấu hình darknet

Việc cấu hình trực tiếp trên Google Colab tương đối khó khăn với những bạn chưa quen làm việc với linux, bên cạnh đó vấn đề delay kết nối giữa Google Colab với drive trong một số trường hợp. Vì vậy nhóm em sẽ thực hiện cấu hình trên máy cá nhân, xong nén thành file .zip, sau đó đẩy lên Google Drive.

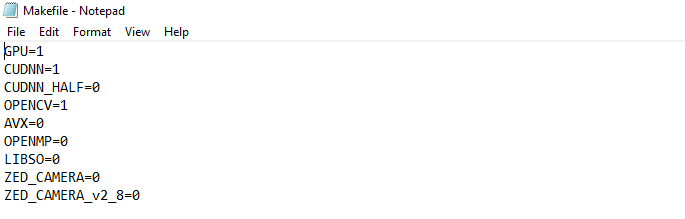
Việc cấu hình gồm các bước sau:

Bước 1 :Tải darknet về máy



Nội dụng file Darknet

Bước 2: Sửa file Makefile trong thư mục Darknet



Cài đặt GPU =1 ,CUDNN =1 để tăng tốc độ trên GPU

Cài đặt CUDNN\_HALF = 1 khi huấn luyện mô hình trên GPU có TensorCores để tăng tốc độ huấn luyện. Trên Google Colab có hỗ trợ GPU Tesla T4, là GPU có kiến trúc mới nhất trên Colab và có TensorCores.Nên ta có thể thay giá trị lên 1.

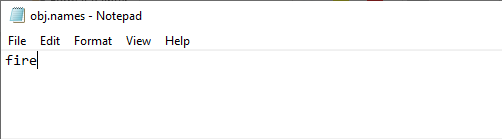
Cài đặt AVX=1 ,OPENMP =1 để tăng tốc độ trên CPU

Bước 3: Tạo file yolo-tinyv4-fire.cfg.

Tạo file yolo-tinyv4-fire.cfg với nội dung tương tự file yolov4-tiny.cfg trong thư mục darknet/cfg, sau đó chỉnh sử một số dòng:

* Dòng 6: Thay đổi batch=64. Nghĩa là: batch = số ảnh (cả file annotation) được đưa vào huấn luyện trong một batch.
* Dòng 7: Thay đổi subdivisions=16. Trong một batch được chia thành nhiều block, mỗi block chứa batch/subdivisions ảnh được đưa vào GPU xử lý tại một thời điểm. Weights của mô hình được update sau mỗi batch.
* Dòng 20: Thay đổi max\_batches=classes2000, không nhỏ hơn số ảnh trong tập huấn luyện, và không nhỏ hơn 6000 (theo [đây](https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects)}). VD: max\_batches=6000.
* Dòng 22: Thay đổi steps= 80%, 90% max\_batches. VD: steps=4800,5400. Sau khi huấn luyện được 80%, 90% max\_batches, learning\_rate sẽ được nhân với một tỷ lệ (dòng 23 trong file), mặc định là 0.1.
* Thay đổi classes=1 trong mỗi layer [yolo], dòng 217, 266.
* Thay đổi filters trong mỗi layer [convolutional] trước layer [yolo] theo công thức filters=(số class+5)\*3. Trong bài toán này filters=18.

Bước 4: Tạo file obj.names chứa tên của các class, sau đó lưu trong thư mục darknet/data.



Bước 5: Tạo file obj.data, sau đó lưu trong thư mục darknet/data.



Bước 6: Đưa toàn bộ thư mục chứa ảnh và file annotation ở trên vào thư mục darknet/data.

Bước 7: Download pre-trained weights của YOLO-Tinyv4, lưu trong thư mục darknet.

Bước 8: Nén thư mục darknet thành file darknet.zip, sau đó đưa lên Google Drive.

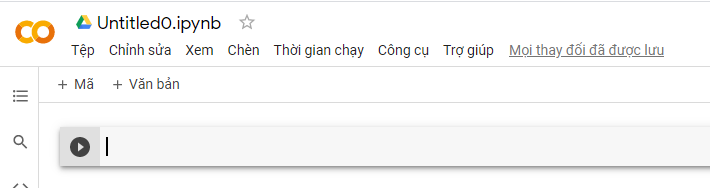
* 1. Huấn luyện model trên Colab

2.3.1 Tại sao lại huấn luyện trên Colab ?

* Huấn luyện model deep learning cần tài nguyên khá lớn. Đối với một quốc gia nghèo như Việt Nam thì tôi không cho rằng việc một sinh viên bỏ ra vài nghìn $ mua GPU để lắp vào Laptop là hợp lý. Trái lại, rất lãng phí và không hiệu quả.
* Google colab là một virtual cloud machine được google cung cấp miễn phí cho các nhà nghiên cứu. Đây là môi trường lý tưởng để phát triển các mô hình vừa và nhỏ. Điểm tuyệt vời ở google colab đó là môi trường của nó đã cài sẵn các packages machine learning và frame works deep learning thông dụng nhất.
* Việc cài các frame work deep learning trên máy cá nhân đôi khi khá tốn thời gian vì các lỗi xung đột package, xung đột hệ điều hành. Các bạn có thể mất vài ngày để sửa các lỗi cấu hình trên máy. Trong khi sử dụng google colab là dùng được ngay.
* Cấu hình RAM và GPU của các bạn chưa chắc đã tốt như google. Theo ước tính của tôi, bạn cần 100 triệu để xây một cấu hình máy tương đương với google colab.
* Việc sử dụng GPU trên cấu hình RAM, chip yếu có thể khiến laptop của bạn nhanh bị hỏng.
* Với google colab, bạn có thể dễ dàng làm việc với data được chia sẻ trên google drive từ người khác

2.3.2 Các bước huấn luyện mô hình trên Colab

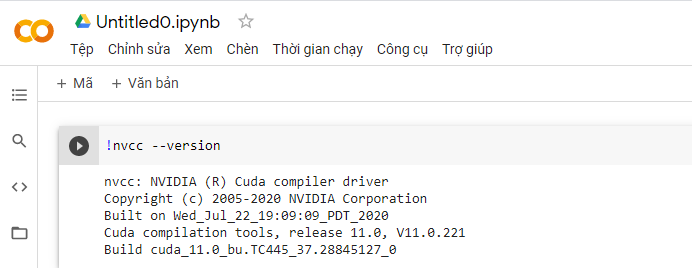
Bước 1 : Thiết lập cài đặt ban đầu



Giao diện GG COLAB

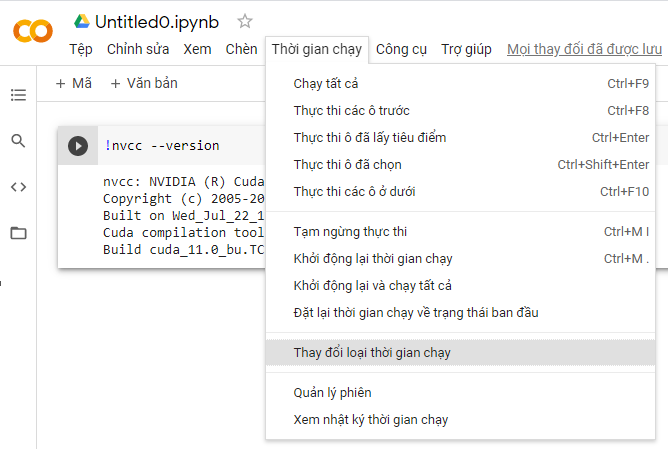
* Chúng ta vừa có thể trình bày văn bản kết hợp với xử lý câu lệnh. Các phần trình bày văn bản được tạo ra từ các block text và xử lý code được tạo ra từ các block codes.
* Ngôn ngữ thực thi mặc định của Google colab là python. Ngoài ra bạn có thể thực thi các lệnh command line bằng cách thêm một markup là dấu ! ở đầu câu lệnh.

Ví dụ bên dưới đang chạy một vài lệnh trên commandline của ubuntu để kiểm tra version cuda được google cài đặt.

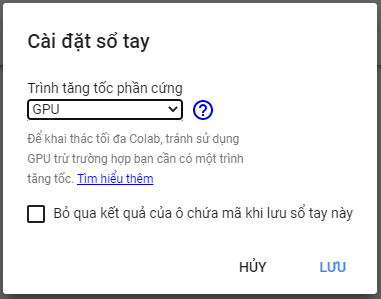


Enable GPU trên Google Colab

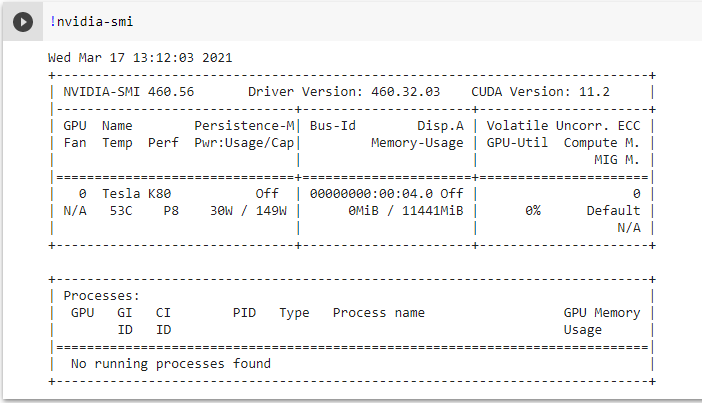
Mặc định google colab sẽ disable GPU để tiết kiệm tài nguyên. Do đó chúng ta enable bằng cách: Trên thanh công cụ của google colab click vào Runtime(Thời gian chạy) > change runtime type.



Tại cửa sổ mới mục Trình tăng tốc phần cứng ta lựa chọn GPU và lưu.



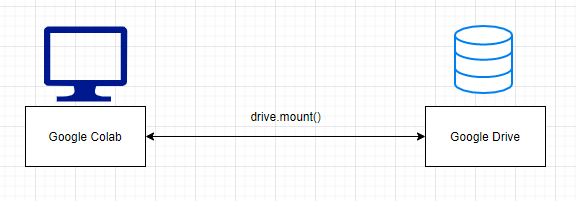
Muốn biết GPU đã enable thành công chưa, ta sử dụng:



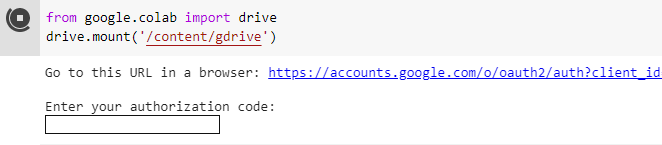
Như vậy google colab cung cấp 1 GPU Tesla K80 với bộ nhớ 11441MiB. Và phiên bản CUDA là 11.2 .Các thiết lập đơn giản không cần cài đặt phức tạp tránh được các xung đột lỗi khi cài đặt.

Bước 2 : Mount Google Driver

Google colab có tác dụng như là một VM (virtual machine computing) làm nhiệm vụ tính toán, xử lý dữ liệu. Google Drive là nơi lưu trữ dữ liệu. Do đó để VM truy cập được tới dữ liệu tại Google drive thì ta cần mount drive.

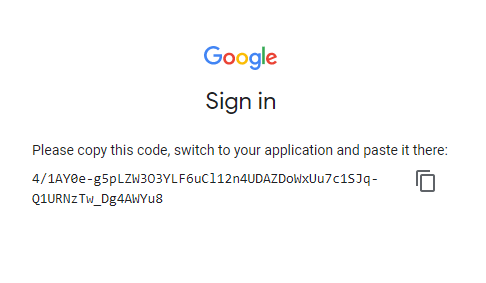


Ảnh minh họa mount giữa Google Colab và Google Drive



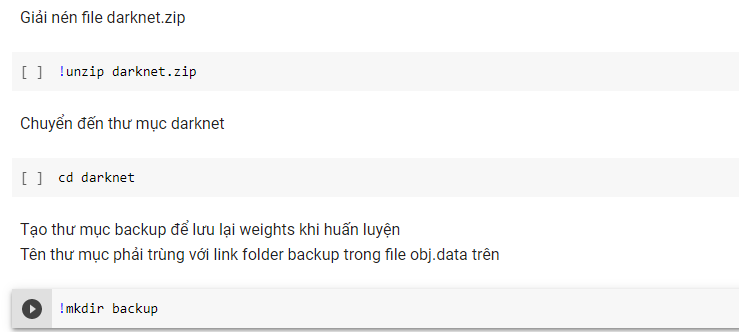
Dòng Enter your authorization code:: Yêu cầu chúng ta nhập token để đăng nhập vào VM.

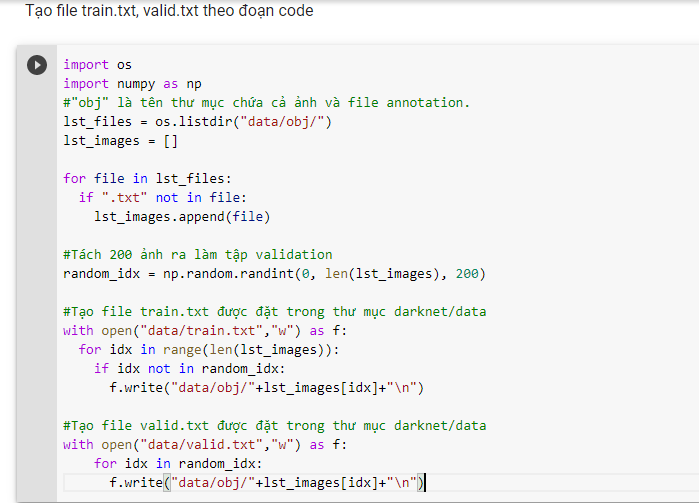
Click vào link Go to this URL in a browser để xác thực email. Sau khi accept các bước ta thu được đoạn mã token bên dưới:

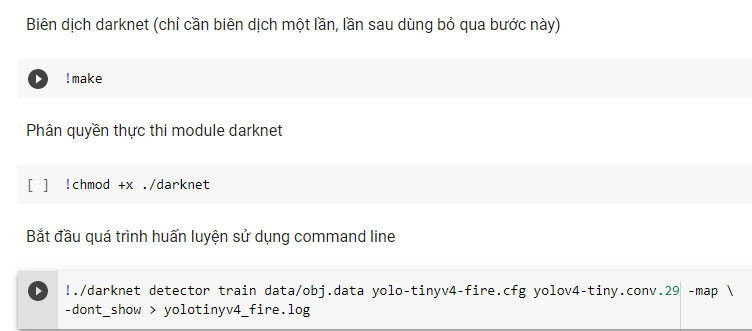


Copy token và paste vào dòng Enter your authorization code:. Sau bước này chúng ta đã establish một connection từ VM đến Google drive.

Bước 3 : Huấn luyện model bằng các dòng lệnh commandline







Cú pháp tổng quát để huấn luyện:

!./darknet detector train [data config file] [model config file] [pre-trained weights] -map > [file log saved]

-map: Dùng để hiển thị mAP được tính trên tập validation.

Nếu bạn gặp lỗi:  
CUDA Error: out of memory: File exists thì hãy quay lại sửa subdivisions=32 trong file yolo-tinyv4-obj.cfg

Quá trình huấn luyện sẽ được lưu vào file yolotinv4\_fire.log, ngoài ra darknet tự động tạo ra ảnh chart.png lưu trong thư mục darknet và được cập nhật liên tục để theo dõi trực tiếp thông số của quá trình huấn luyện.



3. Cảnh báo

3.1 Các cách thức cảnh báo

3.2 Xây dựng chương trình cảnh báo

CHƯƠNG IV : KẾT QUẢ VÀ ĐÁNH GIÁ

1. Kết quả
2. Đánh giá