**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI**

**You Only Look Once:**

**Unified, Real-Time Object Detection**

**Môn học : Toán cho khoa học máy tính**

**Giảng viên lý thuyết : Lương Ngọc Hoàng**

**Nhóm thực hiện : Trương Đăng Nghĩa – 20521658**

**Lê Phước Trung – 20522069**

***TP. Hồ Chí Minh, tháng 12 năm 2021***

## LỜI NÓI ĐẦU

Google Colaboratory (gọi tắt là Google Colab hay Colab) là một sản phẩm của Google Research. Colab dựa trên Jupyter Notebook, người dùng có thể viết và thực thi đoạn mã python thông qua trình duyệt và đặc biệt rất phù hợp với data analysis, machine learning và giáo dục.

Nhằm cũng cố kiến thức và nắm rõ hơn kiến thức của môn học, đồng thời phù hợp với điều kiện đáp ứng cấu hình của các bạn, nhóm chúng em quyết định chọn Google Colab để mô phỏng.

## LỜI CẢM ƠN

Trân trọng gửi lời cảm ơn thầy Lương Ngọc Hoàng đã tạo điều kiện cho chúng em có cơ hội được làm đồ án.

Chỉ trong vòng 15 tuần, nhưng nhờ sự chỉ dẫn nhiệt tình của thầy, chúng em đã tiếp thu được những kiến thức quan trọng để có thể làm được một bài training model hoàn chỉnh.

Cũng xin cảm ơn thầy cô trong khoa Khoa học máy tính đã nhiệt tình hỗ trợ, tạo điều kiện chúng em làm bài báo cáo này.

Mục lục

[LỜI NÓI ĐẦU 2](#_Toc91167615)

[LỜI CẢM ƠN 2](#_Toc91167616)

[I. TỔNG QUAN 3](#_Toc91167617)

[1. Lý do chọn đề tài 3](#_Toc91167618)

[2. Mục đích nghiên cứu đề tài 3](#_Toc91167619)

[3. Phạm vi nghiên cứu đề tài 3](#_Toc91167620)

[4. Công nghệ sử dụng 3](#_Toc91167621)

[II. GIỚI THIỆU VỀ YOU ONLY LOOK ONCE (YOLO): 3](#_Toc91167622)

[1. Object Detection 3](#_Toc91167623)

[2. Cấu tạo và thuật toán Yolo: 6](#_Toc91167624)

[III. KẾT QUẢ ĐẠT ĐƯỢC’ 9](#_Toc91167625)

[IV. KHÁC 9](#_Toc91167626)

[1.Thuận lợi và khó khăn 9](#_Toc91167627)

[2. Phân công công việc 10](#_Toc91167628)

[3. Tài liệu tham khảo 10](#_Toc91167629)

## I. TỔNG QUAN

### 1. Lý do chọn đề tài

Nhằm hiểu rõ cách xử lý với database, phương thức hoạt động cũng như có thể sử dụng lại Google colab để train những model khác.

### 2. Mục đích nghiên cứu đề tài

Tìm hiểu cách training model, tác dụng và các bước thực hiện của YOLO: Unified, Real-Time Object Detection.

### 3. Phạm vi nghiên cứu đề tài

Giới hạn trong những xử lý ở mức chạy được, chưa cần phải thông minh. Ngôn ngữ được hỗ trợ là python.

### 4. Công nghệ sử dụng

Training bằng Google colab, dựa trên Jupyter Notebook và sử dụng ngôn ngữ python để code.

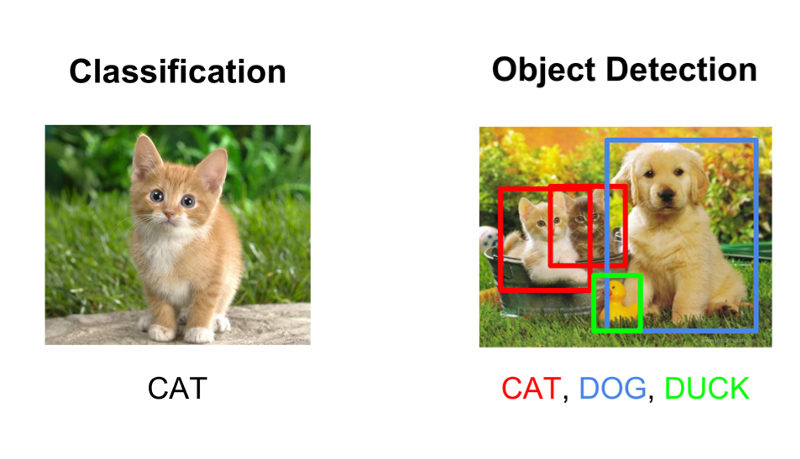
## II. GIỚI THIỆU VỀ YOU ONLY LOOK ONCE (YOLO):

### Object Detection

Object detection (OD), dịch nôm na là nhận diện đối tượng trong ảnh, đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Tuy vậy, một số người vẫn hay bị nhầm lẫn với Image classification (IC).

- Image classification là đưa một tấm ảnh vào và đầu ra là kết quả tấm ảnh đó là cái gì. Vd: chó, mèo, gà, lợn, hay gái, trai….

-Còn Object detection thì khác, input sẽ là một tấm ảnh hoặc video, trả về đầu ra sẽ là các vật thể, đối tượng có trong ảnh đó kèm theo tọa độ (bounding box) của nó. Vd: trong ảnh có con mèo tại tọa độ x1,y1,x2,y2 và có chó tại tọa độ x3,y3,x4,y4…



* 1. **. Khởi đầu cho Object Detection**

Để tìm giải pháp cho OD trên một bức ảnh thì suy nghĩ rất tự nhiên là ta sẽ sử dụng một cái kéo và một mạng CNN. Cách thực hiện vô cùng đơn giản là cắt riêng các vùng ảnh sau đó đưa từng vùng ảnh đó vào mạng CNN để xem trong vùng đó có cái giống gì. Nếu phát hiện vật thể trong vùng ảnh nào đó thì lưu lại vị trí đã cắt để làm bounding box.

Các làm đơn giản này gặp khá nhiều khó khăn bởi các vật thể có kích thước khác nhau giữa các ảnh, góc quay, hướng quay khác nhau giữa các ảnh nên số lượng vùng ảnh phải rất lớn và “cực phong phú đa dạng” về kích thước, tỷ lệ thì may ra mới phát hiện được hết các vật thể trong ảnh. Điều này làm cho thuật toán chạy chậm và tốn nhiều tài nguyên tính toán.

* 1. **. R-CNN**

Đến năm 22/10/2014, [Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik](https://arxiv.org/pdf/1311.2524.pdf" \t "_blank) mới nghĩ ra cách cải tiến mới.

Cách làm là thay vì chọn tất cả vùng ảnh sau đó dùng CNN vào thì sẽ rất tốn thời gian và dung lượng máy tính, họ áp dụng một thuật toán [**Selective Search**](https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf) để chọn ra 2000 vùng ảnh từ những vùng ảnh được sinh ra ban đầu. Sau đó mới áp dụng CNN vào 2000 vùng ảnh đó. Bớt đi được ối việc cho máy tính đó.

Diagram

Description automatically generated with medium confidence

Chi tiết hơn về cái việc áp dụng CNN vào các vùng ảnh một chút thì các bạn để ý hình trên sẽ thấy:

* Input: là vùng ảnh được trích ra trên ảnh gốc.
* Output: Thông qua đặc trưng và predict sẽ cho ra được class của vùng ảnh cũng như 4 giá trị offset để điều chỉnh bounding box quanh vật thể trong vùng ảnh.Ví dụ có một vùng ảnh nhận là chó, nhưng không phải cả vùng ảnh đó là con chó mà chỉ có nửa người con chó thôi thì bounding box không thể là cả vùng ảnh đó được, ta dùng offset để chỉnh cho bounding box vào đúng phần thân con chó.

Tuy vậy, R-CNN vẫn còn nhiều nhược điểm cần cải tiến như:

* Số lượng vùng ảnh vẫn lớn và vẫn chọn một cách ngẫu nhiên, thủ công mỹ nghệ và do đó nhiều vùng ảnh chả có cái gì cũng mất công đưa vào CNN để predict.
* Do các điểm trên nên tốc độ predict khá chậm, tốn đến 47s/ ảnh (vào thởi điểm tác giả thử nghiệm).

Từ đó, họ cải thiện và nâng cấp object detection R-CNN thành Fast R-CNN trong một nghiên cứu tiếp theo với mong muốn tăng tốc độ predict bằng cách giảm chi phí tính toán cho máy tính.

Cơ bản về cách làm cũng gần giống với R-CNN nhưng cải tiến quan trọng nhất giúp tăng tốc độ tính toán là thay vì input 2000 vùng ảnh trực tiếp vào trong mạng CNN thì các họ đã thực hiện đưa cả ảnh vào mạng CNN 1 lần duy nhất.

Sau khi đưa vào CNN thì ta sẽ có Feature Map với kích thước nhỏ hơn ảnh gốc rất nhiều. Và bây giờ ta sử dụng một ROI Pooling layer để đưa các vùng ảnh về kích thước vuông và reshape về cùng một kích thước đầu ra. Với kết quả có được, ta lại đưa tiếp qua các lớp FCs và có được một output gọi là ROI feature vector.

Rồi, có được ROI feature vector rồi thì đơn giản, ta lại predict class và offset của bounding box cho vùng mảnh đó để lấy kết quả đầu ra như đã làm với R-CNN.

Như vậy, nhìn lại một chút thì Fast R-CNN nhanh hơn R-CNN ở chỗ thay vì mất 2000 lần đưa các vùng ảnh vào CNN thì chỉ cần 1 lần đưa ảnh gốc vào CNN. Tuy nhiên, việc lựa chọn các vùng ảnh bằng seletive search trên Feature Maps vẫn là nút thắt và làm chậm cho thuật toán.

**1.4. Faster R-CNN**

Và một lần nữa thuật toán Object Detection mới lại ra đời để xử lý cái nút thắt nói trên. Nhưng lần này không phải là các tác giả cũ nữa mà là [Shaoqing Ren](https://arxiv.org/pdf/1506.01497.pdf) với thuật toán Faster R-CNN.

Về cơ bản thuật toán này tập trung xử lý cái nút thắt – Selective Search để tăng tốc. Họ đã cải tiến bằng cách thay vì thủ công mỹ nghệ lựa chọn thì build luôn một mạng cho nó học luôn đâu là vùng ảnh nên chọn để tìm kiếm object.

Diagram

Description automatically generated

Thế cụ thể họ làm thế nào? Họ cũng áp dụng giống Fast R-CNN đó là đưa ảnh vào CNN để lấy feature map (FM) cái đó. Nhưng cái khác là sau khi có FM thì thay vì dùng selective search thì đưa cái FM vào một cái mạng riêng gọi là Regional Proposal Network (RPN) – mạng tìm kiếm các vùng ảnh. Thế là sau khi qua cái mạng đó thì ta có các vùng ảnh có khả năng chứa object.

Cuối cùng thì ta đưa các vùng ảnh tìm được qua ROI Pooling layer để reshape, rồi sau đó sử dụng một số lớp FCs tiếp theo để predict class, offset của bounding box như các thuật toán trên.

Nhanh hơn ở đây là khử cái selective search và thay vì phải predict trên các vùng ảnh hú hoạ thì ta chỉ cần xử lý trên các vùng ảnh có khả năng chứa vật cao nhất mà thôi.

**1.5 YOLO**

Version mới nhất của YOLO này là phiên bản số 4 (bản số 5 còn nhiều tranh cãi và chưa có chính thức).

Như vậy đến phần Faster R-CNN chúng ta đã đạt được thành quả khá ổn với việc chỉ xử lý trên các vùng ảnh có khả năng chứa object cao nhất thay vì xử lý toàn ảnh như lúc ban đầu.

YOLO là You Only Look Once và đúng như vậy, nó không xem xét từng vùng mà nhìn toàn bộ bức ảnh một lần sau đó predict ra class, bounding box cho từng object trong hình. Thay vì xét vùng này vùng kia thì YOLO chia ảnh thành một lưới vuông với kích thước giả sử là NxN chẳng hạn. Sau đó nó duyệt từng ô trong ảnh, với mỗi ô thì lại thử m cái anchor box. Với mỗi anchor box đó thì sẽ dự đoán class và offset của bounding box của vật thể.

Cuối cùng thì các bouding box có xác suất lớn nhất trên ngưỡng sẽ được giữ lại áp dụng một thuật toán có tên Non Max Supression (NMS) để chỉ giữ lại các bounding box tốt nhất và output ra kết quả cho người dùng.

Xét về tốc độ thì YOLO hiện đang khá nhanh so với các phương pháp nói trên (tốc độ thử nghiệm max đạt được là 45 FPS). Tuy vậy điểm yếu của YOLO lại là hay bỏ sót các object nhỏ trên ảnh như: phát hiện xe trên ảnh vệ tinh, ảnh chụp flycam….

Trong thực tế mạng YOLOv3 đã sử dụng 3 lưới khác nhau (3 feature map size khác nhau) để tiến hành predict trên các ô của lưới. Điều này làm tăng khả năng detection của mạng, tránh bỏ sót.

### Cấu tạo và thuật toán Yolo:

* 1. **Cấu trúc:**

YOLO ver 1 sử dụng 24 lớp chập: bao gồm lớp (1×1) gọi là Reduction layers (dùng để giảm kích thước ảnh),lớp chập (3×3) gọi là Convolution layers.Xen kẽ giữa 24 lớp chập là các lớp max pooling và kết thúc bằng 2 lớp fully connected. Kết quả là một ma trận 3 chiều có dạng 7x7x30.

Với mỗi 1x1x30 trong 7x7x30 được gọi là một tensor, vậy đối với  **7×7**x30 thì ta sẽ có **7×7**=49 tensors.

Diagram, engineering drawing

Description automatically generated

YOLO  chia một bức ảnh thành **SxS** ô lưới (grid cell), thường là 7×7 . Trọng tâm của vật thể được tìm trong các ô lưới đó ( được học thông qua việc gán nhãn và đào tạo).Nếu trọng tâm của đối tượng nằm trong ô lưới bất kỳ, thì nó (ô lưới chứa trọng tâm của đối tượng) sẽ chịu trách nhiệm tìm vật thể đó.

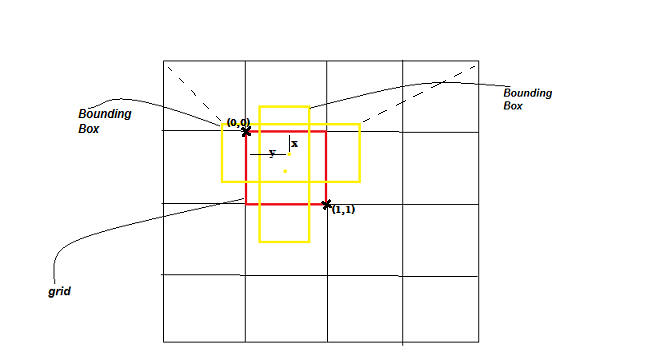
A dog looking at a bicycle

Description automatically generated with low confidence

Vậy làm sao để những ô lưới đó tìm vật thể?

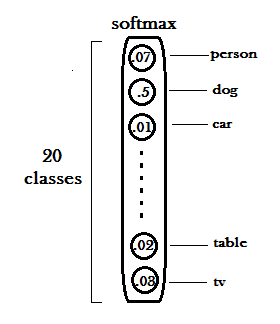
Các ô lưới này sẽ phải đưa ra (2xB +C) giá trị dự đoán, trong đó:

* + B bounding box( tức là hộp chứa đối tượng)- đối với YOLOv1 thì B=2. Với mỗi B sẽ có 5 giá trị nhỏ nữa là : **x** (trọng tâm đối tượng theo phương x- đối chiếu với ô lưới chứa nó), **y** ( trọng tâm đối tượng theo phương y – đối chiếu với ô lưới chứa nó), **w**(độ rộng của vật thể- đối chiếu với kích thước của toàn bộ bức ảnh), **h**( độ cao của vật thể – đối chiếu với kích thước của toàn bộ bức ảnh) và **confidence**( độ tự tin- xác suất đó là vật thể).
  + Từ 2 bounding box B, ta dựa vào **confidence**của bounding box nào lớn hơn và IOU so với  ground truth box ( nhãn được gán bằng tay) lớn hơn thì được xuất ra là vật thể.



* + C sẽ bằng số classes và có giá trị là xác suất của trọng tâm vật thể rơi vào ô lưới đó( đối với YOLOv1 thì số classes là 20, bởi vì model YOLOv1 là một model đã được train sẵn và chúng ta không thể đào tạo hay sửa chữa được nên chỉ có thể lấy ra dùng).

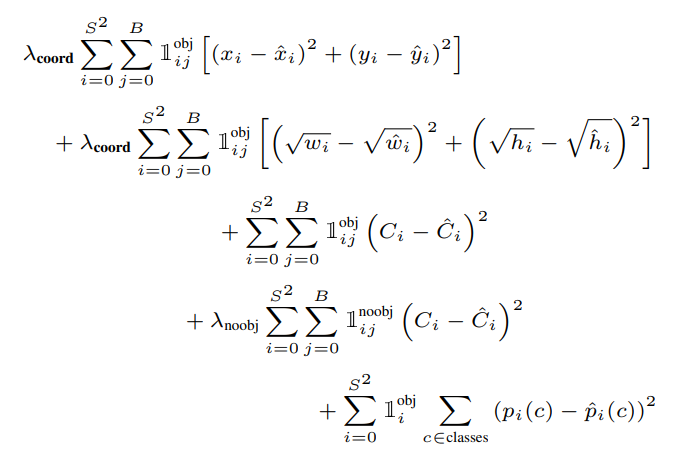
**LƯU Ý: confidence**và**C là**khác nhau.



Với mỗi ô lưới thì tối đa sẽ chỉ phát hiện được 1 đối tượng, vậy nên đối với YOLOv1 thì sẽ xảy ra những vấn đề sau:

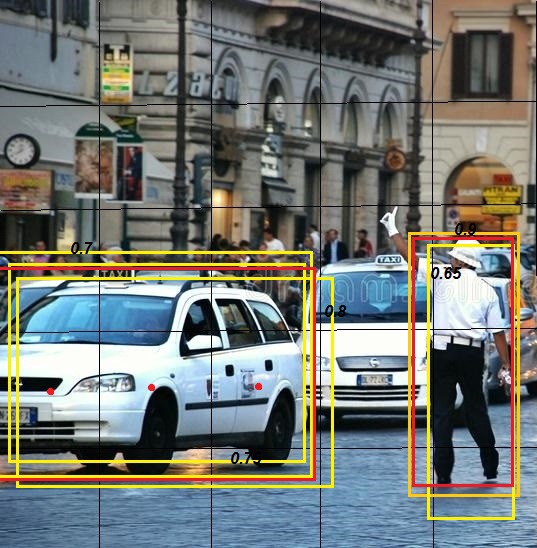
* Phát hiện được tối đa 49 đối tượng( nếu thuật toán chia ảnh thành 7×7).
* Nếu một ô lưới chứa trọng tâm của nhiều đối tượng, thì model cũng chỉ phát hiện được một đối tượng có độ chính xác cao nhất.
* Nếu vật thể nào đó quá lớn so với bức ảnh, đồng nghĩa với việc vật thể đó sẽ có nhiều trọng tâm ở nhiều ô lưới khác nhau. Vậy nên vật thể đó sẽ được phát hiện nhiều lần.
  1. **Loss function.**

YOLO sử dụng hàm “tổng bình phương lỗi” (sum-squared error. Viết tắt là SSE). “Lỗi” ở đây tức là lấy giá trị của ground truth box( nhãn gán bằng tay) trừ đi bounding box prediction ( BB mà model dự đoán), sau đó bình phương giá trị này lên. Các giá trị x,y,w,h,C là giá trị của ground truth box, còn các giá trị có dấu mũ là của bb mà model dự đoán.



* Vế đầu tiên và vế thứ hai tính toán **Localization loss**.
* Vế thứ ba và bốn tính toán **Confidence loss**.
* Vế cuối cùng tính **Classification loss**.
  1. **Non-maximal supperession (NMS)**

Như đã trình bày ở trên, sẽ có nhiều vật thể lớn và trọng tâm của nó có thể rơi vào nhiều ô lưới(grid cell) cùng một lúc, sẽ dẫn tới việc một vật thể sẽ được dự đoán nhiều lần…Cho nên tác giả đã sử dụng NMS để loại bỏ những trường hợp như thế này. Cách thực hiện:



1. Loại bỏ những đối tượng có confidence C < C-threshold ( ngưỡng này chúng ta có thể re setup).
2. Sắp xếp những đối tượng có confidence C theo thứ tự giảm dần.
3. Chọn bounding box của những vật thể có confidence C cao nhất và xuất ra kết quả.
4. Loại bỏ những bounding box có IOU < IOU-threshold (ngưỡng này chúng ta có thể re setup)
5. Quay lại bước 3 cho tới khi mọi kết quả được kiểm tra.

## IV. KẾT QUẢ ĐẠT ĐƯỢC’

* Thao tác, làm việc bằng google colab.
* Sử dụng ngôn ngữ Python.
* Nắm được lí thuyết Object Detection.
* Có thể tự tạo dataset và làm việc với darknet, sử dụng YOLO để train và detect object.

## V. KHÁC

### Thuận lợi và khó khăn

* 1. **Thuận lợi**
     + Hầu hết dữ liệu cần import đều có trên google colab, không cần phải cài đặt thêm.
     + Google colab có liên kết google drive, tiện lợi cho việc tương tác với dữ liệu và lưu trữ.
     + Có thể sử dụng CPU/GPU của google để chạy, vì vậy không cần thiết bị tốt vẫn có thể chạy được.
     + Model có sẵn để dựa theo, không cần phải tự thiết kế từ đầu.
  2. **Khó khăn**
     + Vì google colab chỉ có thể sử dụng trong vòng 10 tiếng, vì vậy nên bị giới hạn thời gian train.
     + Vì YOLO áp đặt ràng buộc về không gian trên nhữn bounding box, mỗi grid cell chỉ có thể predict rất ít bounding box và duy nhất một class. Các ràng buộc này hạn chế khả năng nhận biết số object nằm gần nhau, cũng như đối với các object có kích thước nhỏ.

### 2. Phân công công việc

|  |  |
| --- | --- |
| Chức năng | Người phụ trách |
| Bản báo cáo | Nghĩa, Trung |
| Thiết kế slide cho PP để thuyêt trình | Trung |
| Sử dụng model để code, kiểm tra kết quả thực nghiệm | Trung, Nghĩa |

### 3. Tài liệu tham khảo

Redmon You\_Only\_Look\_CVPR (2016): <https://openaccess.thecvf.com/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf>

YOLO9000: Better, Faster, Stronger:

<https://arxiv.org/pdf/1612.08242.pdf>

YOLOv3: An Incremental Improvement: <https://pjreddie.com/media/files/papers/YOLOv3.pdf>

YOLOv4: Optimal Speed and Accuracy of Object Detection: <https://arxiv.org/pdf/2004.10934v1.pdf>

Và một vài nguồn tài liệu khác.