**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐIỆN – ĐIỆN TỬ**

**🕯✡🕮🕮✡🕯**



**MÔN HỌC: ĐỒ ÁN 1**

**BÁO CÁO**

**XÂY DỰNG MÔ HÌNH CNN**

**NHẬN DIỆN VẬT THỂ**

**GVHD: TS. Đỗ Duy Tân**

**SVTH: MSSV**

**Vũ Danh Long 18119167**

**Trương Thành Lợi 18119169**

**Tp. Hồ Chí Minh, tháng 7 năm 2021**

**ĐIỂM SỐ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TIÊU CHÍ** | **NỘI DUNG** | **BỐ CỤC** | **TRÌNH BÀY** | **TỔNG** |
| **ĐIỂM** |  |  |  |  |

**NHẬN XÉT**

*Ký tên*

**TS. ĐỖ DUY TÂN**

**MỤC LỤC**

[**MỞ ĐẦU** 1](#_Toc75193929)

[1. Đặt vấn đề 1](#_Toc75193930)

[2. Tình hình nghiên cứu trong và ngoài nước 1](#_Toc75193931)

[*2.1.* *YOLOv1* 1](#_Toc75193932)

[*2.2.* *YOLOv2 & YOLO 9000* 3](#_Toc75193933)

[*2.3.* *YOLOv3* 4](#_Toc75193934)

[*2.4.* *YOLOv4* 5](#_Toc75193935)

[*2.5.* *YOLOv5* 6](#_Toc75193936)

[3. Mục đích nghiên cứu 7](#_Toc75193937)

[4. Nội dung nghiên cứu 7](#_Toc75193938)

[5. Bố cục 7](#_Toc75193939)

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 8](#_Toc75193940)

[1.1. Kiến trúc mạng YOLO 8](#_Toc75193941)

[1.2. Grid System 9](#_Toc75193942)

[1.3. Khái niệm về chỉ số IoU và thuật toán Non-max suppression 11](#_Toc75193943)

[1.4. Gán nhãn cho các mẫu 12](#_Toc75193944)

[1.5. Hàm mất mát 14](#_Toc75193945)

[1.6. Ngõ ra 15](#_Toc75193946)

[1.7. Thuật toán YOLOv4 15](#_Toc75193947)

[1.8. Cấu trúc nhận diện vật thể của YOLOv4 16](#_Toc75193948)

[1.9. Backbone 17](#_Toc75193949)

[1.10. Neck 19](#_Toc75193950)

[1.11. Head 22](#_Toc75193951)

[**CHƯƠNG 2: XÂY DỰNG MÔ HÌNH CNN PHÂN LOẠI VẬT THỂ TRÊN ẢNH** 24](#_Toc75193952)

[2.1. Sơ đồ khối tổng quát 24](#_Toc75193953)

[2.2. Datasets 24](#_Toc75193954)

[2.3. Tiền xử lý và phân chia dữ liệu 27](#_Toc75193955)

[2.4. Định nghĩa mô hình 28](#_Toc75193956)

[*2.4.1.* *Mô hình baseline* 29](#_Toc75193957)

[*2.4.2.* *Mô hình VGG16* 30](#_Toc75193958)

[*2.4.3.* *Mô hình baseline sử dụng Normalization* 31](#_Toc75193959)

[2.5. Huấn luyện mô hình 33](#_Toc75193960)

[2.6. Tạo giao diện người dùng 34](#_Toc75193961)

[**CHƯƠNG 3: XÂY DỰNG MÔ HÌNH NHẬN DIỆN VẬT THỂ TRÊN VIDEO** 38](#_Toc75193962)

[3.1. Sơ đồ khối 38](#_Toc75193963)

[3.2. Datasets 38](#_Toc75193964)

[3.3. Gán nhãn và tạo anchor box. 39](#_Toc75193965)

[3.4. Huấn luyện mô hình 40](#_Toc75193966)

[*3.4.1.* *Pycharm* 41](#_Toc75193967)

[*3.4.2.* *Huấn luyện mô hình* 41](#_Toc75193968)

[**CHƯƠNG 4: KẾT LUẬN** 44](#_Toc75193969)

[4.1. Kết luận 44](#_Toc75193970)

[*4.1.1.* *Mô hình CNN phân loại vật thể bằng ảnh.* 44](#_Toc75193971)

[*4.1.2.* *Mô hình CNN phát hiện vật thể bằng video.* 44](#_Toc75193972)

[4.2. Hướng phát triển 44](#_Toc75193973)

# 

# MỞ ĐẦU

1. **Đặt vấn đề**

Ngày nay, với sự phát triển hết sức mạnh mẽ của công nghệ, Artificial Intelligence (trí tuệ nhân tạo) hay cụ thể là Machine Learning (máy học) cũng đang từng bước phát triển không ngừng. Các ứng dụng về lĩnh vực này ngày càng được áp dụng nhiều vào đời sống con người như: Robot giúp việc, hệ thống dịch thuật, chatbox, phân loại sản phẩm,…

Mạng Nơron học sâu (Deep learning Network) là lĩnh vực nghiên cứu các thuật toán, chương trình máy tính để máy tính có thể học tập và đưa ra những dự đoán như con người. Nó được ứng dụng vào nhiều ứng dụng khác nhau như khoa học, kỹ thuật, các lĩnh vực đời sống khác cũng như các ứng dụng về phân loại và phát hiện đối tượng. Một ví dụ điển hình là CNN (Convolutional Neural Network) áp dụng để nhận dạng tự động, tìm hiểu các mẫu phân biệt từ ảnh bằng cách xếp chồng liên tiếp các lớp lên nhau và trong nhiều ứng dụng, CNN hiện nay được coi là trình phân loại ảnh mạnh và thúc đẩy các công nghệ trong lĩnh vực thị giác máy tính, làm đòn bẩy cho quá trình học máy. Nhưng bên cạnh đó, để phân loại được một đối tượng thì công nghệ CNN tiêu tốn cực lớn về tài nguyên như băng thông, bộ nhớ và khả năng xử lý của phần cứng.

Để giảm thiểu những tài nguyên tiêu hao này, những thuật toán, mô hình giải thuật theo thời gian được ra đời ngày càng nhiều và mô hình YOLO cho bài toán phân loại vật thể là một trong số đó.

1. **Tình hình nghiên cứu trong và ngoài nước**

Với bài toán nhận diện và phân loại vật thể, có hai vấn đề lớn cần giải quyết là phát hiện và phân loại vật thể. Có rất nhiều cách để phân loại vật thể bao gồm R-CNN, Fast R-CNN, Faster R-CNN,… và một số thuật toán khác. Trong đó, thuật toán YOLO (You Only Look Once) được xem là vượt trội hơn về mặt thời gian khi có thể đáp ứng thời gian thực tốt hơn, mặc dù khả năng nhận diện được đánh giá là thấp hơn R-CNN series nhưng trade-off giữa khả năng nhận diện chính xác và thời gian nhận diện vật thể được đánh giá rất cao.

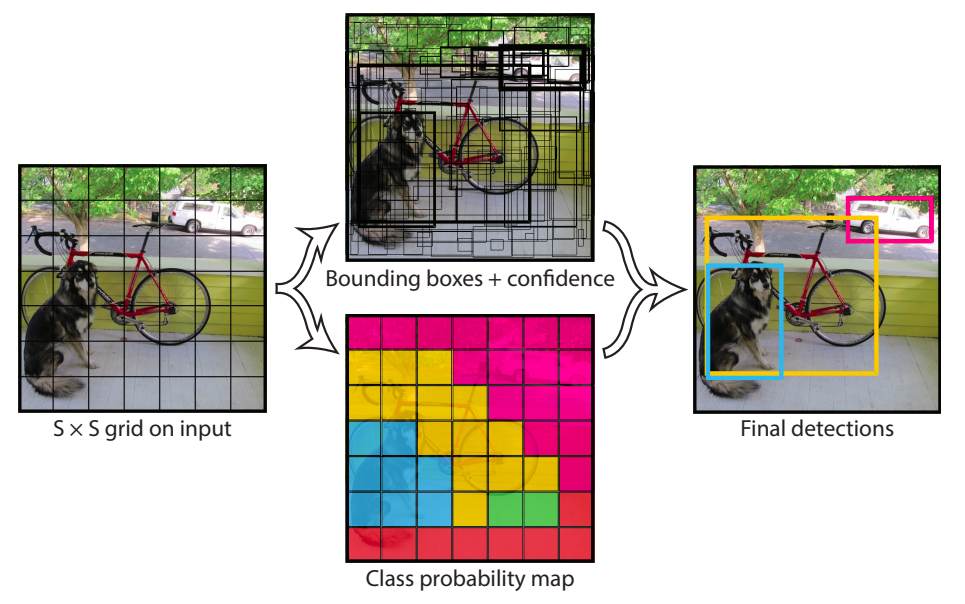
* 1. ***YOLOv1***

Đầu vào của bài toán là một bức ảnh. Bài toán giúp chúng ta phân loại được object trên bức ảnh và phải định vị được vị trí của đối tượng đó.

Khi chưa có YOLOv1, các phương pháp trước đây thường sử dụng 2 bước:

* Bước 1: sử dụng sliding window để lấy các vùng khác nhau của bức ảnh, hoặc sử dụng một thuật toán lựa chọn các vùng có khả năng chứa vật.
* Bước 2: phân loại các vị trí này xem vật đó thuộc lớp nào.

Các cách làm này có nhược điểm là yêu cầu một lượng tính toán lớn, và bị phân nhỏ thành nhiều bước, khó có thể tối ưu về mặt tốc độ. Kiến trúc YOLOv1 coi bài toán phát hiện vật như một bài toán regression. Từ input là ảnh đầu vào, qua một mạng gồm các lớp convolution, pooling và fully connected là có thể ra được output. Kiến trúc này có thể được tối ưu để chạy trên GPU với một lần forward pass, và vì thế đạt được tốc độ rất cao.



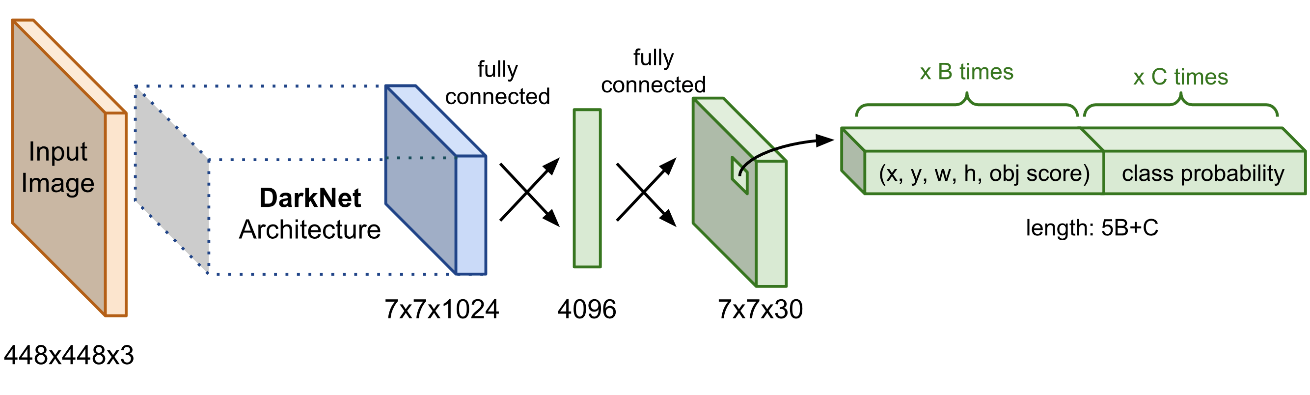
*Hình 1. Cách YOLOv1 dự đoán [1]*

YOLOv1 chia ảnh thành một lưới các ô (grid cell) với kích thước SxS (mặc định là 7x7). Với mỗi grid cell, mô hình sẽ đưa ra dự đoán cho B bounding box. Ứng với mỗi box trong B bounding box này sẽ là 5 tham số x, y, w, h, confidence, lần lượt là tọa độ tâm (x, y), chiều rộng, chiều cao và độ tự tin của dự đoán. Với grid cell trong lưới SxS kia, mô hình cũng dự đoán xác suất rơi vào mỗi class.

Độ chính xác của dự đoán ứng với mỗi bounding box được định nghĩa là  
trong đó là xác suất có vật trong cell và là Intersection over Union của vùng dự đoán và ground truth.

Xác suất rơi vào mỗi class cho một grid cell được ký hiệu . Các giá trị xác suất cho C class sẽ tạo ra C output cho mỗi grid cell. Lưu ý là B bounding box của cùng một grid cell sẽ chia sẻ chung một tập các dự đoán về class của vật, đồng nghĩa với việc tất cả các bounding box trong cùng một grid cell sẽ chỉ có chung một class.

Vậy tổng số output của mô hình sẽ là . Hình dưới đây là kiến trúc của YOLOv1. Mạng backbone của YOLOv1 lấy ý tưởng từ kiến trúc GoogleNet.



*Hình 2. Kiến trúc của YOLOv1 [2]*

Nhược điểm của YOLOv1:

* YOLOv1 áp đặt các ràng buộc về không gian trên những bounding box, mỗi grid cell chỉ có thể predict rất ít bounding box (B) 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.
* Ngoài ra, trong quá trình training, loss function không có sự đánh giá riêng biệt giữa error của bounding box kích thước nhỏ so với error của bounding box kích thước lớn. Việc coi chúng như cùng loại và tổng hợp lại làm ảnh hưởng đến độ chính xác toàn cục của mạng. Error nhỏ trên box lớn nhìn chung ít tác hại, nhưng error nhỏ với box rất nhỏ sẽ đặc biệt ảnh hưởng đến giá trị IOU.
  1. ***YOLOv2 & YOLO 9000***

YOLOv2 đặt tên là YOLO9000 đã được Joseph Redmon và Ali Farhadi công bố vào cuối năm 2016 và có mặt trong 2017 CVPR. Cải tiến chính của phiên bản này tốt hơn, nhanh hơn, tiên tiến hơn để bắt kịp faster R-CNN (phương pháp sử dụng Region Proposal Network), xử lý được những vấn đề gặp phải của YOLOv1.

Một số điểm cải tiến của YOLOv2 có thể nói qua như:

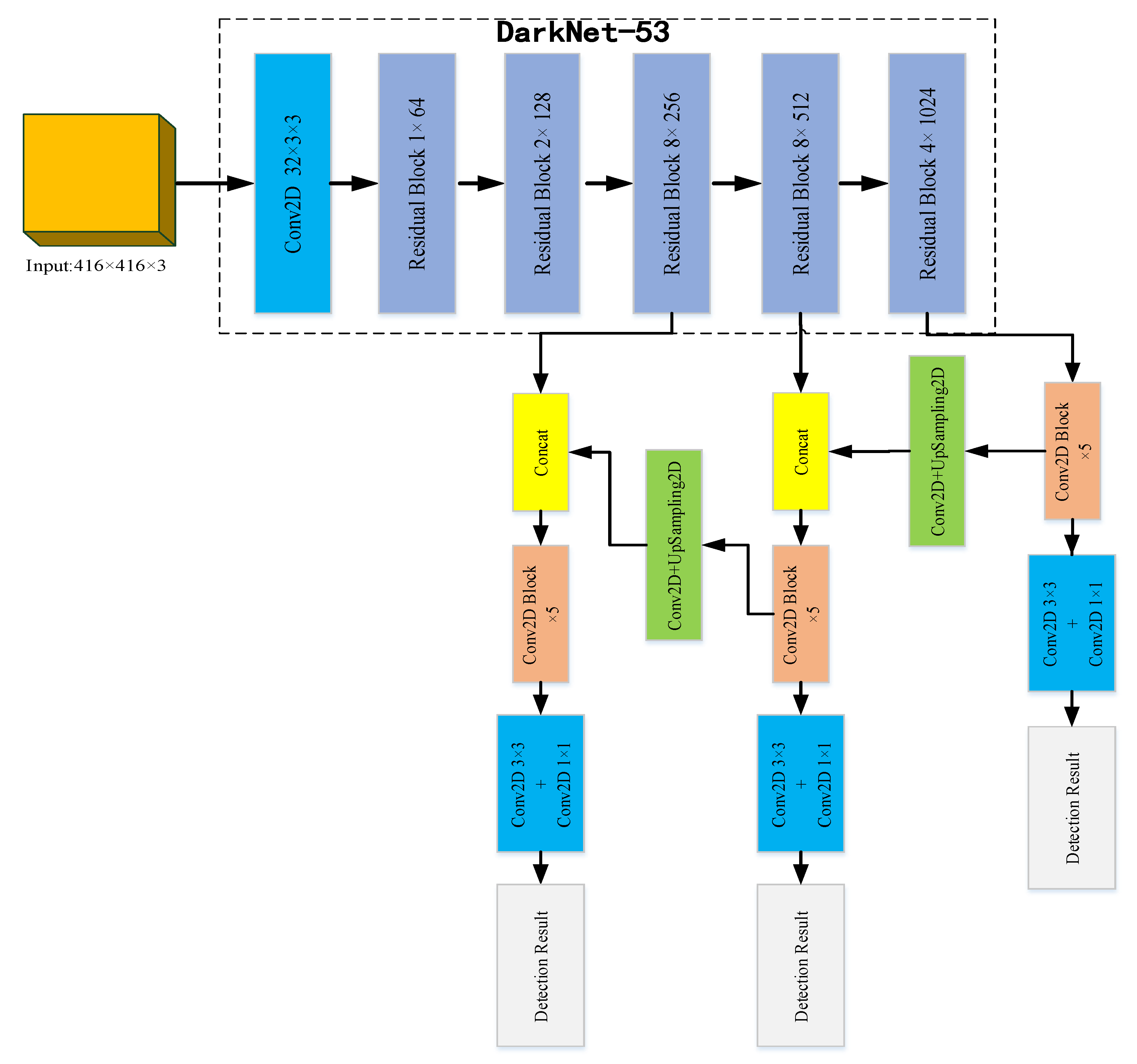
* Thêm Batch Normalization
* High resolution classifier
* Sử dụng kiến trúc anchorbox để đưa ra dự đoán
* K-mean clustering cho lựa chọn anchor
* Direct location prediction
* Thêm fine-grained features
* Multi-Scale Training
* Light-weight backbone

YOLO9000 đưa ra cách kết hợp các dataset khác với ImageNet để có thể phát hiện nhiều class hơn. Tác giả tạo một directed graph gọi là WordTree. Để có thể merge được các label từ tập ImageNet (1000 class) với COCO/PASCAL (100 class), tác giả dựa vào WordNet để xây dựng quan hệ giữa các class, từ đó có thể huấn luyện mạng nhận dạng các class có quan hệ với nhau.

* 1. ***YOLOv3***

YOLOv3 có kiến trúc khá giống YOLOv2. Tác giả đã thêm các cải tiến mới trong các nghiên cứu gần đây vào YOLOv2 để tạo ra YOLOv3. Các cải tiến đó bao gồm:

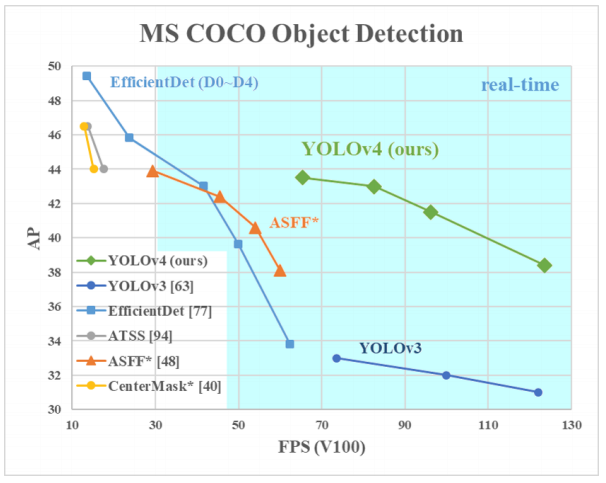
* Logistic regression cho confidence score: YOLOv3 predict độ chính xác của bounding box (có chứa vật hay không) sử dụng logistic regression.
* Thay softmax bằng các logistic classifier rời rạc: YOLOv3 sử dụng các logistic classifier thay vì softmax cho việc classify đối tượng. Việc này cho hiệu quả tốt hơn nếu các label không "mutually exclusive", tức là có thể có đối tượng cùng thuộc 2 hay nhiều class khác nhau. Ví dụ với bài toán cần phát hiện 2 đối tượng là "chó" và "chó Phú Quốc". Rõ ràng nếu đối tượng là "chó Phú Quốc" thì nó cũng thuộc class "chó", và việc sử dụng softmax là không phù hợp trong trường hợp này.
* Backbone mới - Darknet-53: Backbone được thiết kế lại với việc thêm các residual blocks (kiến trúc sử dụng trong ResNet).
* Multi-scale prediction: YOLOv3 sử dụng kiến trúc Feature Pyramid Networks (FPN) để đưa ra các dự đoán từ nhiều scale khác nhau của feature map. Việc này giúp YOLOv3 tận dụng các feature map với độ thô - tinh khác nhau cho việc dự đoán.
* Skip-layer concatenation: YOLOv3 cũng thêm các liên kết giữa các lớp dự đoán. Mô hình upsample các lớp dự đoán ở các tầng sau và sau đó concatenate với các lớp dự đoán ở các tầng trước đó. Phương pháp này giúp tăng độ chính xác khi predict các object nhỏ.



*Hình 3. Kiến trúc YOLOv3 [3]*

* 1. ***YOLOv4***

YOLOv4 là một loạt các cải tiến về tốc độ so với YOLOv3 và được cài đặt từ một bản fork của Darknet. Kiến trúc của YOLOv4 đã đưa bài toán object detection dễ tiếp cận hơn với những người không có tài nguyên tính toán mạnh. Chúng ta hoàn toàn có thể huấn luyện một mạng phát hiện vật với độ chính xác rất cao bằng YOLOv4 chỉ với GPU 1080ti hoặc 2080ti. Trong tương lai, việc tối ưu lại các mạng hiện tại để phù hợp với tài nguyên tính toán yếu hoặc tạo ra sự song song hóa cao ở các server chắc chắn phải được thực hiện để có thể đưa các ứng dụng computer vision vào thực tế.



*Hình 4. Kết quả so sánh YOLOv4 với các SOTA ở thời điểm hiện tại. YOLOv4 chạy nhanh gấp đôi EfficientDet, tăng AP và FPS so với YOLOv3 lần lượt là 10% và 12%.*

Trên đây là kết qủa so sánh của YOLOv4 với các thuật toán state-of-the-art tại thời điểm hiện tại. Nhìn vào biểu đồ, ta dễ dàng thấy được sự hiệu quả của YOLOv4 so với các mạng tốt nhất hiện nay. Cụ thể hơn YOLOv4 đạt 43.5% AP trên tập dữ liệu MS COCO ở tốc độ 65 FPS, trên GPU Tesla V100. Để tạo ra kết quả này, tác giả sử dụng một loạt các kĩ thuật mới: Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation, Mosaic data augmentation, DropBlock regularization, và CIoU loss.

* 1. ***YOLOv5***

Hiện tại có một phiên bản YOLOv5 đang được phát triển bởi Ultralytics LLC. Phiên bản này hiện khá triển vọng theo các số liệu được cung cấp bởi công ty phát triển. Tuy nhiên phiên bản YOLOv5 này chưa có paper chính thức được chấp nhận và cũng đang có nhiều tranh cãi xung quanh tính hiệu quả của mô hình đang được phát triển này.

1. Mục đích nghiên cứu

Tìm hiểu về Deep Learning và các ứng dụng.

Hiểu rõ được cơ sở lý thuyết, kiến trúc của mô hình CNN, YOLO cho bài toán nhận diện vật thể.

Sử dụng các thư viện hỗ trợ, môi trường ảo để thực thi mô hình.

Thực thi được mô hình nhận diện biển báo cụ thể.

1. Nội dung nghiên cứu

Ở báo cáo này, nhóm sẽ nghiên cứu tìm hiểu về Deep Learning, các mô hình CNN, YOLOv4 và tiến hành xây dựng mô hình mạng CNN phân loại biển báo giao thông cụ thể, sau đó đánh giá nghiên cứu và tùy chỉnh để phù hợp với yêu cầu của báo cáo.

1. Bố cục

Bài báo cáo được chia làm 3 chương:

Chương 1: Cơ sở lý thuyết

Chương 2: Xây dựng mô hình CNN phân loại biển báo giao thông trên ảnh

Chương 3: Xây dựng mô hình nhận diện biển báo giao trông trên video

Chương 4: Kết luận và hướng phát triển.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

* 1. **Kiến trúc mạng YOLO**

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

Diagram

Description automatically generated

*Hình 5. Sơ đồ kiến trúc mạng YOLO [2].*

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.

A picture containing text, receipt

Description automatically generated

*Hình 6. Các layer trong mạng Darknet-53 [5].*

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.

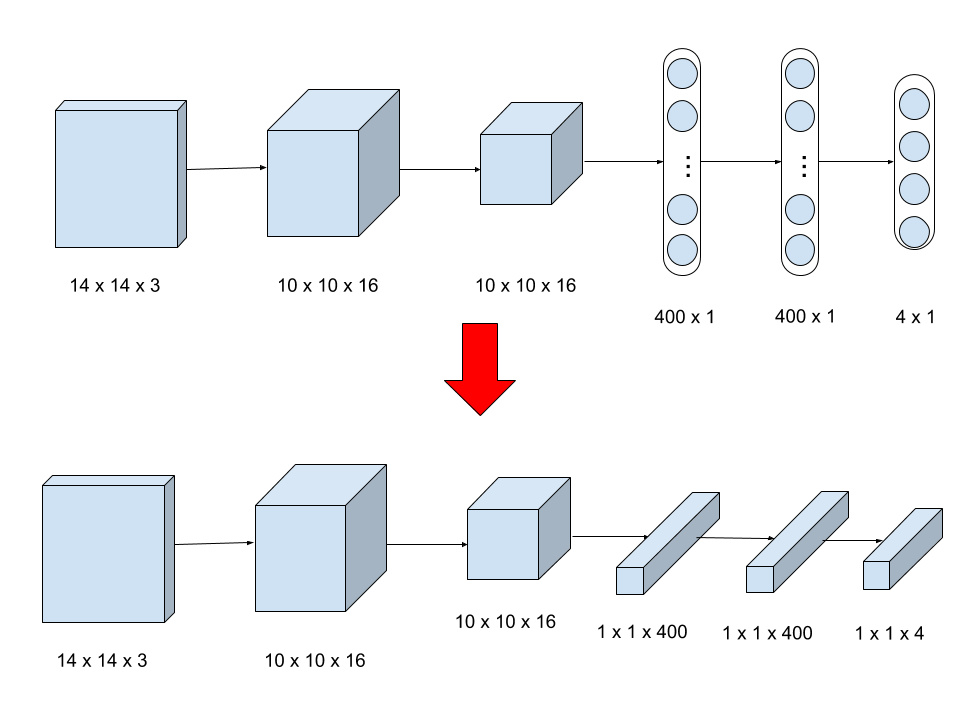
Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là 416x416 và 608x608. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo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

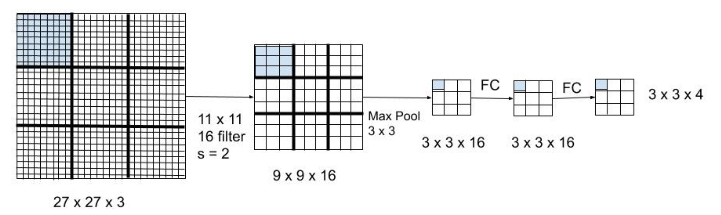
* 1. **Grid System**

Ý tưởng của việc thực hiện Grid System xuất phát từ việc thay vì thực hiện các lớp Fully Connected ở cuối mô hình thì các lớp này sẽ được chuyển thành các lớp tích chập như các lớp phía trước trong bài toán phân loại vật thể. Khi đó ở lớp cuối cùng thay cho lớp fully connected có kích thước (số classes, 1)  thành một lớp tích chập có kích thước 3 chiều (1,1,số classes)

Việc chuyển đổi này không làm ảnh hưởng đến kết quả dự đoán mà còn thể hiện được chúng ta hoàn toàn có thể phân loại vật thể bằng các phép toán tích chập với kết quả dự đoán nằm ở lớp tích chập cuối cùng và còn ưu việt hơn khi vẫn giữ được vị trí của vật thể. Việc chuyển đổi diễn ra như sau, ví dụ, hình ảnh chứa vật cần phân loại có kích thước 14 x 14 x 3, sau khi thực hiện các bước tích chập nhận được lớp tích chập cuối cùng có kích thước 1 x 1 x 4, chính lớp tích chập mang kết quả phân loại vật thể.

  
  
*Hình 7: Chuyển các lớp Fully Connected cuối thành các lớp tích chập.*

Để chứng minh vị trí vật thể không thay đổi và vẫn có thể xác định được qua lớp cuối ta giả sử hình ảnh đang xét có kích thước 27 x 27 x 3 được chia thành 3 x 3 grid như hình dưới, vị trí của vật thể nằm ở ô có đánh màu, sau khi thực hiện các bước tích chập, ta thu được lớp tích chập cuối cùng có kích thước 3 x 3 x 3 .



*Hình 8: Vị trí và thông tin của vật thể được duy trì đến lúc cuối cùng*

*(Để đơn giản, trong hình trên không vẽ số chiều của lớp tích chập).*

Ta có thể thấy sau khi thực hiện tích chập dữ liệu và vị trí của vật thể được duy trì cho đến lớp cuối cùng, và ở ô màu tương ứng mang kết quả phân loại của vật thể. Như vậy ta vừa có thể phân loại vật thể vừa xác định được vị trí của vật thể.

Qua đó, grid system sẽ chia hình gốc thành số grid tương đương với kích thước của lớp cuối (không đề cập đến số chiều), như ví dụ trên lớp cuối có kích thước 3 x 3 vậy ta sẽ chia hình gốc thành 3 x 3 grid (đường kẻ đậm). Tại đó mỗi grid sẽ mang 3 thông tin chính: Grid có đang chứa vật thể hay không, tọa độ của các bounding box (gồm tọa độ x,y của góc trên bên trái và chiều dài, chiều rộng của bounding box), xác suất phân loại vật thể. Xét hình 2.15, giả sử chúng ta cần nhận diện biển báo cấm vượt quá 30km/h, mặc dù biển báo không nằm trọn trong một grid nhưng thuật toán sẽ chỉ xác định tâm vật thể (hình tròn trong hình dưới) và tâm vật thể nằm ở grid nào thì grid đó sẽ được xác định tồn tại vật thể. Ma trận của 1 grid như sau:

[ pc , bx , by , bh , bw , c1 , c2 , c3 , ... ]

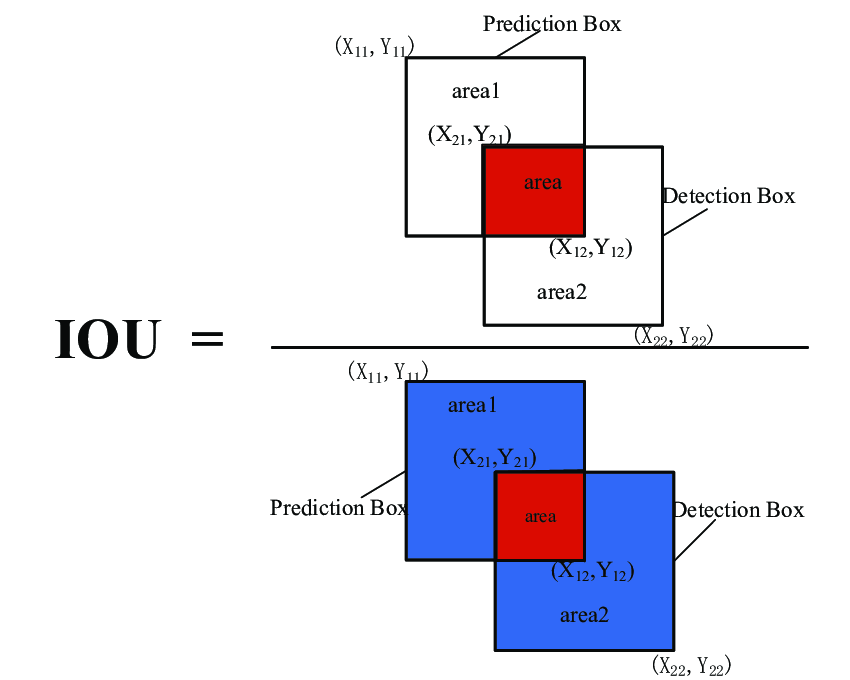


*Hình 9: Mô tả grid system trong bài toán nhận diện biển báo.*

* 1. **Khái niệm về chỉ số IoU và thuật toán Non-max suppression**

Chỉ số IoU (Intersection over Union) cho ta biết tỉ lệ trùng vào nhau của 2 box.

 Trong đó  A B là phần giao nhau (Intersection) của box A và box B, A B là phần chung của 2 box (Union) bằng tổng diện tích của 2 box trừ đi phần giao nhau. Việc xác định IoU giúp tính toán khả năng phát hiện chính xác vật thể, trong đó box A thường là các anchor box ( hay groundtruth bounding box) được gán nhãn ở pha huấn luyện và box B là bounding box của hệ thống xác định ở pha kiểm tra. Tính toán IoU để đánh giá mô hình đã phát hiện vật thể đúng hay chưa.



*Hình 10: Phần giao nhau giữa 2 box A và B.*

Trong pha kiểm tra, hệ thống sẽ đưa ra nhiều bounding box khác nhau với các xác suất dự đoán khác nhau và tỉ số IoU khác nhau, vì vậy thuật toán Non-max suppression giúp loại bỏ các bounding box có tỷ lệ dự đoán thấp và chỉ giữ lại 1 bounding box cuối cùng có tỷ lệ dự đoán cao nhất. Thuật toán Non-max suppression diễn ra như sau:

Bước 1: Loại bỏ tất cả các bounding box có xác suất xuất hiện của vật thể pc thấp hơn ngưỡng. Việc loại bỏ như vậy để các grid không chứa vật thể có xác suất xuất hiện của vật thể thấp sẽ không hiển thị bounding box.

Bước 2: Chọn các bounding box có xác suất xuất hiện vật thể cao nhất.

Bước 3: Nếu có nhiều bounding box có cùng xác suất xuất hiện vật thể cao nhất thì ta sẽ loại bỏ bằng IoU, bounding box nào có chỉ số IoU thấp hơn ngưỡng sẽ bị loại bỏ. Kết thúc bước ba, ta sẽ nhận được bounding box có tỉ lệ nhận diện vật thể tốt nhất.

* 1. **Gán nhãn cho các mẫu**

Tương tự các bài toán "Máy học"/"Học sâu", bài toán nhận diện vật thể cũng gồm có 2 pha, pha kiểm tra và pha huấn luyện. Trong pha học của thuật toán YOLO, chúng ta sẽ thực hiện label vị trí và phân loại cho vật thể, khi đó y có dạng như sau:

 y =[pc , bx , by , bh , bw , c1 , c2 , c3,...]

Ví dụ đối với hình 9, giả sử bài toán cần tìm vị trí vật thể và phân loại với 3 nhãn, trong đó biển báo của hình thuộc nhãn thứ nhất. Khi đó, ở grid phát hiện được biển báo giao thông, y sẽ được gán như sau y = [1 , 230 , 120 , 20 , 30 , 1 , 0 , 0], tức là grid đó có vật thể, tọa độ (x,y) ở phía trên bên trái của anchor box là (230,120), anchor box có chiều dài là 20 chiều rộng là 30, biển thuộc nhãn 1 và không thuộc hai nhãn còn lại. Nếu xét grid không chứa vật thể y sẽ được gán y = [0 , x , x , x , x , x , x , x ] tức là không có vật thể ở grid đó và các giá trị còn lại không cần quan tâm.

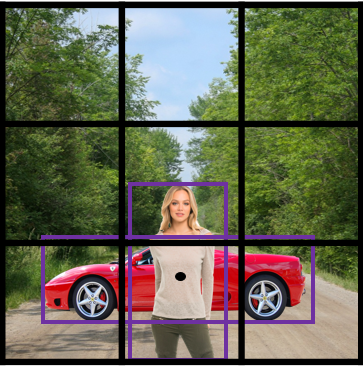
Tuy nhiên, nếu có cùng 2 vật thể cùng xuất hiện trong 1 grid thì không thể thực hiện gán y như trên. Với vấn đề đó, chúng ta có thể xử lý theo 2 cách:

Cách 1: Chia nhỏ grid ra đến khi nào 2 vật thể nằm ở 2 grid khác nhau. Tuy nhiên nếu chia càng nhỏ grid, việc học không thể diễn ra sâu khiến việc phân loại vật thể khó chính xác do các lớp sau không học được các đặc trưng cao.Và nếu tâm của 2 vật thể gần như trùng nhau cũng không thể giải quyết được như trên. Khi đó, ta phải thực hiện theo cách thứ 2.

Cách 2: Thay vì y chỉ được gán cho 1 vật thể, y sẽ được mở rộng ra với nhiều vật thể như sau:

y =[pc1 , bx1 , by1 , bh1 , bw1 , c1 , c2 , c3,...,pc2 , bx2 , by2 , bh2 , bw2 , c1 , c2 , c3,...]

Xét ví dụ với hình 2.16 bên dưới, cô gái và chiếc xe đều nằm cùng chung 1 grid. Khi đó, y sẽ được gán như sau y = [1, 120, 20, 20 , 120 , 1 , 0 , 1 , 90, 50 ,  90 , 20 , 0 , 1] . Tức  là ở grid này có xuất hiện ô tô (y[0] = 1), (120,20,20,120) là 4 thông số anchor box của xe ô tô, (y[5]=1,y[6]=0) để phân loại cho anchor box này là xe ô tô chứ không phải cô gái, tương tự y[7]=1 có nghĩa grid này cũng có cô gái, 4 thông số tiếp theo để xác định tọa độ anchor box cho cô gái cà (y[12]=0,y[13]=1) để phân loại anchor box này là cô gái chứ không phải xe ô tô. Như vậy, với cách 2 ta sẽ ghép y của 2 hay nhiều vật thể nằm trong cùng 1 grid lại thành một, y[0:6] để xác định cho việc phát hiện chiếc xe, và y[7:13] để xác định cho cô gái. Tuy nhiên với cách thứ 2, nếu ta ghép càng nhiều tốc độ xử lý càng lâu do càng có nhiều phép toán phép thực hiện, vì vậy không nên quá lạm dụng cách này mà nên phối hợp hài hòa với cách thứ 1, tăng số grid phải chia lên.



*Hình 11: Tâm của 2 vật thể trùng nhau và cùng nằm trong 1 grid.*

*Nguồn : reddit.com*

* 1. **Hàm mất mát**

Sau khi gán nhãn cho toàn bộ tập dữ liệu, các mẫu dữ liệu sẽ được đưa qua mạng CNN (toàn bộ ảnh sẽ đưa vào mạng CNN) để thực hiện việc học các tham số. Trong quá trình này, các hàm mất mát sẽ được tính toán. Đối với bài toán nhận diện vật thể sẽ cần tính toán 3 hàm mất mát.

* Hàm mất mát phân loại (Classification Loss Function):
* Hàm mất mát vị trí (Localization Loss Function): Được sử dụng để tính toán độ sai số giữa các bounding box dự đoán với các anchor box, cải thiện Localization Loss sẽ giúp việc phát hiện vật thể trở nên chính xác hơn:

- Hàm mất mát dự đoán (Confidence Loss Function): thể hiện sai số giữa dự đoán của bounding box với nhãn thực tế:

Hàm mất mát tổng quát sẽ bằng tổng 3 hàm mất mát phía trên:

* 1. **Ngõ ra**

Ở ngõ ra, mỗi grid sẽ thực hiện dự đoán 2 bounding box có pc cao nhất. Loại bỏ tất cả bounding box có pc thấp trên toàn bộ bức ảnh. Và cuối cùng thuật toán Non-max suppression sẽ được thực hiện để giữ lại bounding box chính xác nhất cho từng vật thể.



1. (b) (c)

*Hình 12: Thuật toán được áp dụng ở ngõ ra*

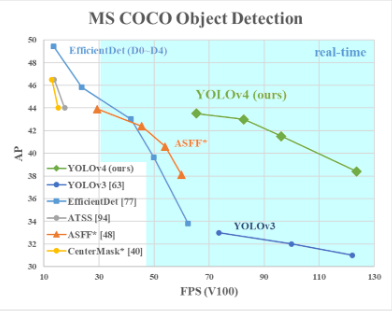
*(a): Với mỗi grid, lấy 2 bounding box có pc cao nhất.*

*(b): Loại bỏ tất cả các bounding box có pc thấp trên toàn bộ bức ảnh, bước này giúp loại bỏ các grid không chứa vật thể.*

*(c): Áp dụng non-max suppression để chỉ chọn ra bounding-box cuối cùng có xác suất dự đoán cao nhất.*

* 1. **Thuật toán YOLOv4**

Thuật toán YOLOv4 về cơ bản cũng thừa kế các phương pháp cơ bản của các YOLO, tuy nhiên YOLOv4 áp dụng một số thuật toán phát hiện vật thể nhanh, tối ưu hóa các phép toán thực hiện song song giúp tăng tốc độ nhận diện và tăng độ chính xác.

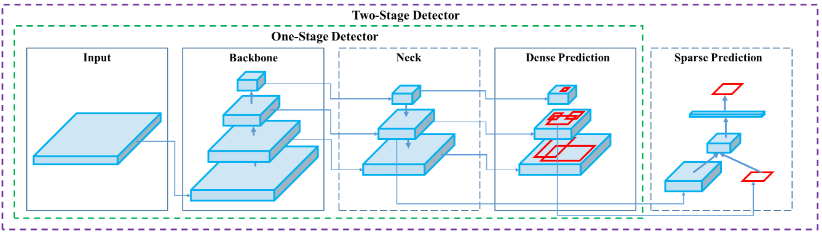


*Hình 13: So sánh tốc độ xử lý và độ chính xác của YOLOv4 với các thuật toán khác trong cùng tập dữ liệu MS COCO[6]. Có thể thấy so với YOLOv3 với cùng một FPS (Frame per Second) YOLOv4 cho độ chính xác (AP) cao hơn hẳn.*

* 1. **Cấu trúc nhận diện vật thể của YOLOv4**

Cấu trúc nhận diện vật thể của YOLOv4 thường có 2 phần:

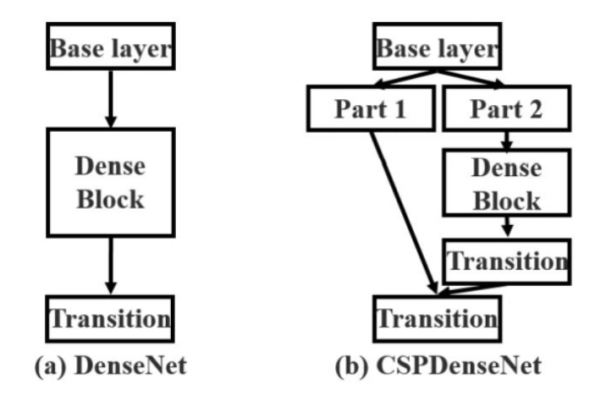
* Backbone: Backbone là 1 mô hình pre-train của 1 mô hình học chuyển (transfer learning) khác để học các đặc trưng và vị trí của vật thể. Các mô hình học chuyển thường là VGG16, ResNet-50,... Mô hình học chuyển được áp dụng trong YOLOv4 là CSP Darknet53
* Head: Phần head được sử dụng để tăng khả năng phân biệt đặc trưng để dự đoán class và bounding-box. Ở phần head có thể áp dụng 1 tầng hoặc 2 tầng:
  + Tầng 1: Dense Prediction, dự đoán trên toàn bộ hình với các mô hình RPN, YOLO, SSD,...
  + Tầng 2: Sparse Prediction dự đoán với từng mảng được dự đoán có vật thể với các mô hình R-CNN series,..
* Neck: Ở phần giữa Backbone và Head, thường có thêm một phần Neck. Neck thường được dùng để làm giàu thông tin bằng cách kết hợp thông tin giữa quá trình bottom-up và quá trình top-down (do có một số thông tin quá nhỏ khi đi qua quá trình bottom-up bị mất mát nên quá trình top-down không tái tạo lại được). Các mô hình được dùng trong quá trình Neck của YOLOv4 là SPP, PAN



*Hình 14: Cấu trúc nhận diện vật thể của YOLOv4[7]*

* 1. **Backbone**

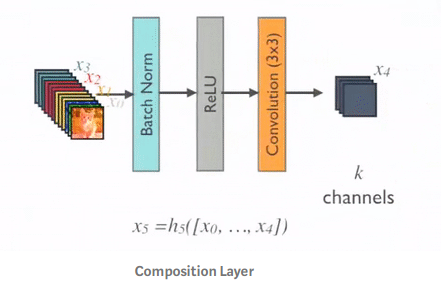
Với backbone, YOLOv4 áp dụng mô hình CSPDarkNet53 vì theo tác giả mô hình có độ chính xác trong việc phát hiện vật thể cao hơn so với các mô hình ResNet thông thường, và tăng khả năng phân loại nhờ vào hàm Mish. CSP-Net về cơ bản cũng tương tự như mô hình DenseNet nhưng có một số khác biệt. CSP(Cross-stage-partial-connection) chia thông tin thành 2 phần bằng nhau, một phần được đưa thẳng vào lớp chuyển kế tiếp (Transition block), phần còn lại được đưa vào dense block (được nhắc đến ở phần sau) trước khi đưa vào lớp chuyển kế tiếp. Chính nhờ phần đưa vào dense block giúp giữ thông tin của các lớp phía trước, và phần đưa thẳng vào Transition block giúp giảm số tham số cần tính toán.



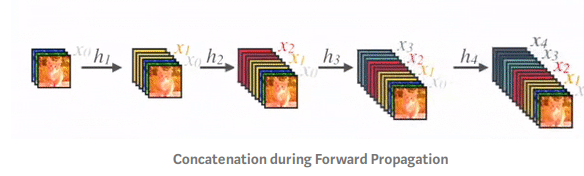
*Hình 15: Sự khác biệt giữa mạng DenseNet và mạng CSPDenseNet.*

DenseNet là một trong những mô hình mạng mới nhất cho kỹ thuật nhận diện vật thể được công bố vào năm 2016. DenseNet gồm 2 khối, trong đó khối chính là Dense Block và 1 khối chuyển. Dense Block là một kĩ thuật mở rộng của skip-connection. Dense Block giúp mô hình trở nên phức tạp hơn làm tăng khả năng học của mô hình, từ đó tăng độ chính xác mà lại giảm được 1 nửa số tham số so với mô hình ResNet trước đó. Tương tự như ResNet, DenseNet cũng giúp giảm overfitting hay vanishing gradient.

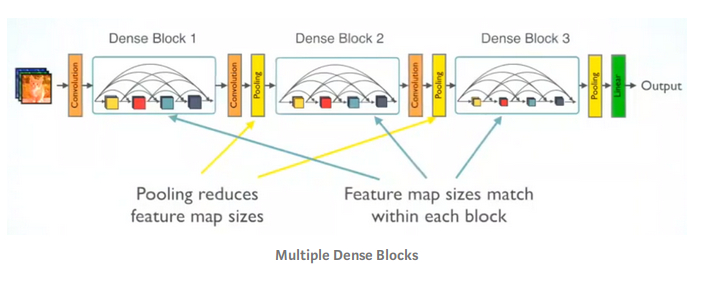
Lớp tích chập đầu tiên được đưa vào DenseNet sẽ thực hiện Batch-Normalization, ReLU và lấy tích chập với 1 filter có kích thước 3x3 để lấy được lớp tích chập thứ 2 (Hình 2.21). Lớp tích chập thứ 2 này sẽ tiếp tục thực hiện ghép chồng (concatenate) với lớp tích chập thứ 1, đây cũng là điểm mới của DenseNet so với ResNet khi ResNet sẽ chỉ lấy element-wise, sẽ làm giảm đáng kể số tham số phải học so với ResNet. Quá trình trên tiếp tục lặp lại với lớp tích chập thứ 2, thứ 3,...(Hình 2.22) Kết thúc quá trình, lớp tích chập cuối cùng sẽ được đưa vào lớp chuyển để thực hiện lấy tích chập hoặc max-pooling và lại đưa vào một Dense Block mới.(Hình 2.23)



*Hình 16: Quá trình lấy Batch-Norm, ReLU, tích chập với filter 3x3 của 1 lớp[8]*

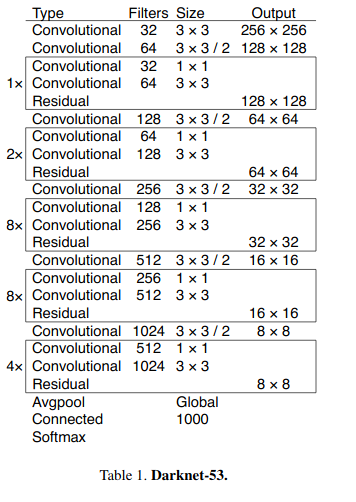
**

*Hình 17: Quá trình ghép chồng lớp tích chập phía trước vào lớp tích chập phía sau[8]*

**

*Hình 18: Mô hình DenseNet với 3 khối Dense Block và các lớp chuyển chèn giữa[8]*

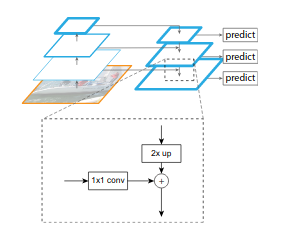
Mô hình Darknet-53 gồm 53 tầng tích chập đã được giới thiệu trong YOLOv3



*Hình 19: Mô hình tích chập Darknet-53 gồm 53 tầng tích chập[5]*

* 1. **Neck**

Quá trình backbone, khi đi đáy lên đỉnh (bottom-up stream) giúp việc xác định vị trí của vật thể nhanh và chính xác, tuy nhiên khi đi qua quá trình trên việc các feature map càng ngày càng thu nhỏ, độ phân giải giảm xuống làm cho một số các vật thể nhỏ qua đó cũng bị biến mất làm mất mát thông tin. Để khắc phục điều đó trước khi đẩy về head để nhận dạng, Neck thường thực hiện một quá trình ngược lại (top-down stream) tạo ra các reconstruction map để khôi phục lại một số thông tin bị mất trong quá trình bottom-up stream. Nhưng một số thông tin đã bị mất khi khôi phục bằng quá trình top-down cũng không còn hiển thị lại nữa, do đó mạng FPN (Feature Pyramid Network), sẽ tái tạo lại các thông tin bị mất nhờ các kết nối skip-connection giữa các feature mác được lấy tích chập 1x1 và reconstruction map tạo ra các feature map mới giàu thông tin giúp việc phát hiện và phân loại vật thể đạt độ chính xác cao hơn.

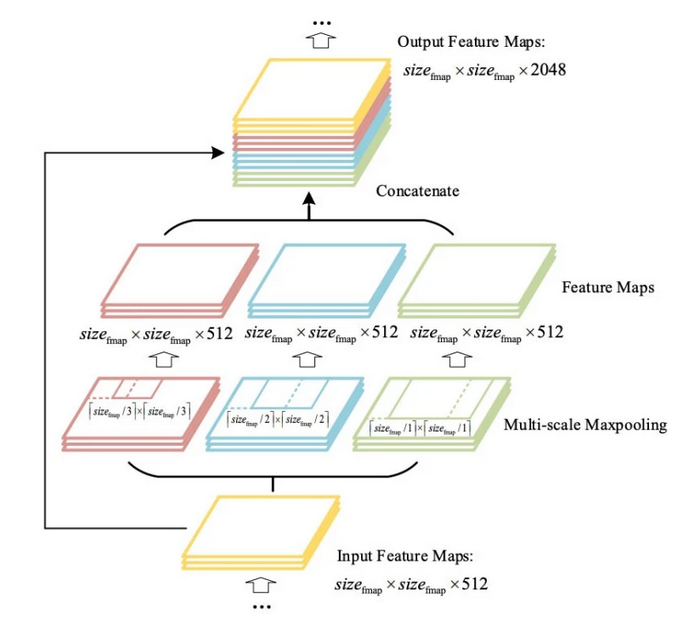


*Hình 20: Mô hình FPN.[9]*

*Trong đó các ô trong khung bottom-up là các feature map trải qua trong quá trình backbone. Các ô trong khung top-down là các reconstruction map. Các ô có viền nét đứt là các feature map sau khi thực hiện skip-connection.*

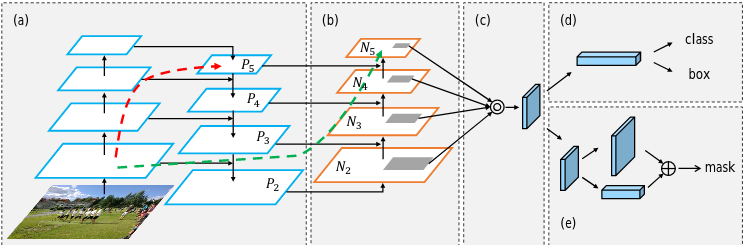
Trong YOLOv4, tác giả có 2 cải tiến đối với mô hình FPN gồm:

* Đối với khối skip-connection giữa feature map và reconstruction map, tác giả sử dụng mô hình YOLO-SPP (Spatial Pyramid Pooling) thay cho các feature map được tích chập 1x1. YOLO-SPP được giới thiệu trong YOLOv3 và tiếp tục được sử dụng trong YOLOv4. YOLO-SPP sử dụng các mạng tích chập max-pooling với các kích thước filter khác nhau, sau khi được lấy tích chập các lớp ngõ ra sẽ được xếp chồng lên nhau. Việc thực hiện max-pooling như vậy giúp giữ được các đặc trưng quan trọng của feature map mà gần như không làm giảm tốc độ xử lý.



*Hình 21: Mô hình YOLO-SPP[10]*

* Đối với mô hình đường tổng hợp (Path Aggregation) trước khi đẩy ra Head, tác giả sử dụng mạng PAN (Path Aggregation Network).



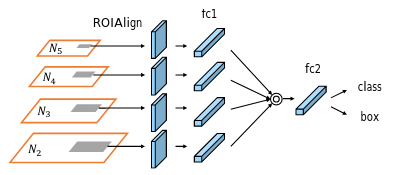
*Hình 22: Mô hình mạng PAN[11]*

PAN được cải tiến từ FPN. Do các mạng Deep Learning càng sâu thì càng dễ mất mát thông tin. Vì vậy, PAN thêm một đường bottom-up được gắn sau đường top-down của FPN. Ở đường bottom-up thứ hai này, feature map của tầng trước đó sẽ được lấy tích chập qua filters 3x3 rồi add với feature của stage tương ứng ở đường top-down. Tuy nhiên trong YOLOv4, tác giả thay vì add tác giả gộp (concat) với stage của đường top-down.



*Hình 23: Thay vì thực hiện cộng như mô hình PAN, YOLOv4 thực hiện phép gộp.[7]*

Sau đó, các vật thể sẽ được nhận diện ở từng tầng. Chính vì từng tầng, các vật thể lại được nhận diện lại dẫn đến việc dư thừa dữ liệu. Do đó, với các vùng ROI (Region of Interest - Vùng quan tâm) sẽ được đi qua 1 mạng SPP để lấy tích chập rồi đưa về mạng Fully Connected, rồi thực hiện lấy max đối với các phần tử (element-wise max) để thu được prediction.



*Hình 24: ROI sẽ đi qua mạng SPP để thu được prediction[11]*

* 1. **Head**

Ở phần Head để thực hiện dự đoán bounding box và class, YOLOv4 được thực hiện giống như YOLOv3.

* Dự đoán cho bounding box: Sử dụng hồi quy logistic để dự đoán. Kết quả dự đoán bằng 1 nếu bounding box có sự trùng lặp so với anchor box tốt nhất trong các bounding box. Các bounding box khác dù có vượt qua mức ngưỡng nhưng không phải là bounding box tốt nhất cũng sẽ bị gán là 0.
* Phân lớp: Để tăng kết quả dự đoán với bài toán phân loại đa nhãn (multi-label classification), tác giả sử dụng phân lớp logistic độc lập (independent logistic classifier) thay cho hàm softmax. Ví dụ với bài toán phân loại 2 lớp, hàm softmax cho kết quả predict chỉ là 0,4 và 0,48 trong khi với hàm phân lớp logistic độc lập kết quả này có thể tăng lên 0.85 và 0.8 cho 2 lớp trên.

# CHƯƠNG 2: XÂY DỰNG MÔ HÌNH CNN PHÂN LOẠI VẬT THỂ TRÊN ẢNH

* 1. **Sơ đồ khối tổng quát**

Bước 1: Tìm kiếm và thu thập datasets về các loại biển báo giao thông, tiền xử lý và gán nhãn cho từng bức ảnh.

Bước 2: Phân chia dữ liệu thành tập train và tập test.

Bước 3: Huấn luyện mô hình.

Bước 4: Đánh giá mô hình.

Sơ đồ khối hệ thống:

A screenshot of a cell phone

Description automatically generated

*Hinh 2.1. Sơ đồ khối tổng quát*

Các giai đoạn xây dựng mô hình sẽ được đề cập trong các phần tiếp theo.

* 1. **Datasets**

Để huấn luyện một mô hình Deep Learning trước tiên ta cần chuẩn bị một tập dữ liệu. tập dữ liệu là yếu tố vô cùng quan trọng trong việc huấn luyện mô hình. Vì dữ liệu ở các môi trường khác nhau (cường độ ánh sáng, vật thể bị méo dạng,..), thì pixel của các ảnh khác nhau. Do đó, tập dữ liệu các lớn giúp mô hình học được vật thể ở nhiều trạng thái môi trường khác nhau, từ đó tăng khả năng chính xác khi dự đoán ngoài thực tế.

Tập dữ liệu thường được chia làm 3 tập:

* Tập training: tập training thường có kích thước lớn nhất. Tập training thường được dán nhãn và cho trước nhãn để tính toán loss function và cập nhập các tham số để mô hình tăng độ chính xác.
* Tập validation (hay tập development): Tập validation thường được chia ra từ tập training. Tập validation là không được dùng để cập nhập các tham số của mô hình, mà nó chỉ được dùng để đánh giá xem mô hình có cần điều chỉnh parameter lại hay không, có bị high bias hay high variance hay không.
* Tập test: tập test được dùng để đánh giá lại mô hình có hiệu quả hay không. Dữ liệu của tập test phải chưa từng xuất hiện trong tập training để có thể đánh giá mô hình ngoài thực tế.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sai số tập training | 1% | 15% | 15% | 0.5% |
| Sai số tập validation | 22% | 16% | 30% | 1% |
| Đánh giá mô hình | Mô hình bị high variance (overfitting) | Mô hình bị high bias  (underfitting) | Mô hình bị cả high bias và high variance | Mô hình hoạt động ổn định |

Các tập training, validation, test thường được chia với tỉ lệ 60/20/20 với tập dữ liệu không quá lớn và được chia tỉ lệ 98/1/1 hoặc 99.5/0.1/0.4 với tập dữ liệu rất lớn (khoảng 1 triệu mẫu).

Trong việc phân loại biển báo giao thông trong ảnh, tập datasets được sử dụng là các biển báo giao thông của Đức được công khai nguồn dữ liệu trên trang web Kaggle [12].

Tại tập datasets này, chúng tôi sử dụng file Train.zip bao gồm 9030 mẫu biển báo của 43 class để tiến hành training và file Test.zip để tiến hành kiểm tra lại.



*Hình 2.2: Tập dữ liệu được sử dụng để xây dựng mô hình CNN*

Trong đó, tập dữ liệu training mẫu biển báo giao thông của Đức với 43 class được gán nhãn như sau:

**BẢNG 3.1: NHÃN VÀ TÊN CÁC BIỂN BÁO TRONG BÀI TOÁN PHÂN LOẠI**

|  |  |  |  |
| --- | --- | --- | --- |
| Nhãn | Tên biển báo | Nhãn | Tên biển báo |
| 0 | Tốc độ giới hạn 20km/h | 22 | Cảnh báo đường nhấp nhô |
| 1 | Tốc độ giới hạn 30km/h | 23 | Cảnh báo đường trơn trượt |
| 2 | Tốc độ giới hạn 50km/h | 24 | Đường bị hẹp bên phải |
| 3 | Tốc độ giới hạn 60km/h | 25 | Đường đang thi công |
| 4 | Tốc độ giới hạn 70km/h | 26 | Cảnh báo có đèn tín hiệu |
| 5 | Tốc độ giới hạn 80km/h | 27 | Cảnh báo có người đi bộ |
| 6 | Hết đoạn đường giới hạn 80km/h | 28 | Cảnh báo có học sinh đi qua |
| 7 | Tốc độ giới hạn 100km/h | 29 | Cảnh báo có xe đạp đi qua |
| 8 | Tốc độ giới hạn 120km/h | 30 | Cảnh báo tuyết lở |
| 9 | Cấm vượt | 31 | Cảnh báo thú hoang đi qua |
| 10 | Cấm xe trên 3,5 tấn vượt | 32 | Hết tất cả lệnh cấm |
| 11 | Giao nhau với đường không ưu tiên | 33 | Rẽ phải phía trước |
| 12 | Giao nhau với đường ưu tiên | 34 | Rẽ trái phía trước |
| 13 | Biển cảnh báo nguy hiểm | 35 | Đi thẳng |
| 14 | Biển dừng lại | 36 | Đi thẳng hoặc rẽ phải |
| 15 | Biển đường cấm | 37 | Đi thẳng hoặc rẽ trái |
| 16 | Cấm xe trên 3.5 tấn | 38 | Hướng phải đi vòng sang phải |
| 17 | Cấm đi ngược chiều | 39 | Hướng trái đi vòng sang trái |
| 18 | Biển nguy hiểm khác | 40 | Nơi giao nhau chạy theo vòng xuyến |
| 19 | Nguy hiểm khi rẽ trái | 41 | Hết đoạn đường cấm vượt |
| 20 | Nguy hiểm khi rẽ phải | 42 | Hết đoạn đường cấm vượt với xe trên 3.5 tấn |
| 21 | Chỗ ngoặt nguy hiểm |  |  |

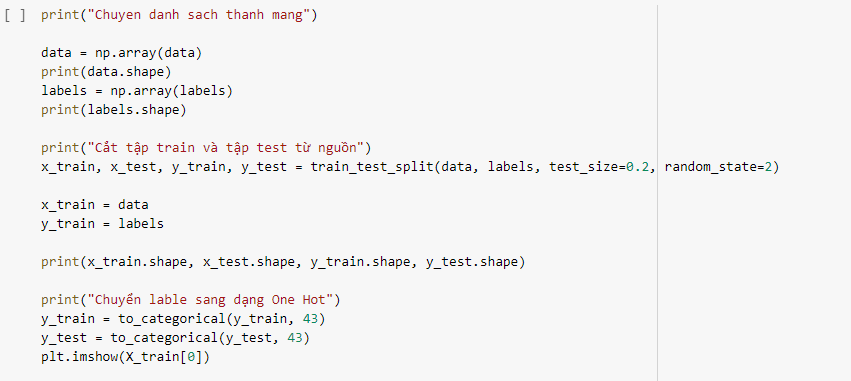
* 1. **Tiền xử lý và phân chia dữ liệu**

Đầu tiên, chúng ta phải truy suất đến tập tin lưu dữ liệu. Sau đó, chúng ta sẽ lưu lại dữ liệu, nhãn của từng bức ảnh và reshape lại tập datasets cho phù hợp với đầu vào của mạng nơtron tích chặp. Quá trình trên được viết như sau:



*Hình 2.3: Tiền xử lý tập datasets*

Tiếp theo, để dễ dàng cho việc tính toán của mạng nơtron tích chặp, chúng ta phải chuyển dữ liệu của mẫu biển báo này từ dạng danh sách sang mảng, sau đó tiến hành cắt tập train và tập test với tỷ lệ 8:2 và chuyển nhãn sang dạng one-hot cho phù hợp với mạng nơtron tích chặp. Quá trình trên được viết như sau:



*Hình 2.4: Xử lý và phân chia tập train, tập test*

* 1. **Định nghĩa mô hình**
     1. ***Mô hình baseline***

Mô hình baseline gồm có 4 block trong đó 3 block đầu thực hiện chức năng trích chọn đặc trưng và block cuối cùng để phân loại dữ liệu. Hàm kích hoạt được sử dụng là hàm ReLU activation ở tất cả các lớp trừ lớp cuối cùng (fc6) hàm kích hoạt được sử dụng là hàm softmax activation.

Ở 3 block tích chập gồm có :

* 2 lớp thực hiện chức năng tích chập.
* 1 lớp thực hiện max pooling.
* lớp thực hiện drop out với tỉ lệ drop là 0.2

Ở block cuối cùng để phân loại dữ liệu gồm có:

* 1 lớp flatten (lớp fc4).
* 2 lớp fully connected (lớp fc5).
* 1 lớp drop out ở giữa 2 lớp fully connected fc5 và fc6 với tỉ lệ drop là 0.5
* 1 lớp fully connected (lớp fc6).

**BẢNG 3.2 : KIẾN TRÚC MÔ HÌNH BASELINE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại | Tên | Số filter | Kích thước filter | stride | Kích thước ngõ ra |
| Input | Input | 3 |  |  | 32 x 32 x 3 |
| Convolution | Conv1-1 | 32 | 3 x 3 |  | 32 x 32 x 32 |
| Convolution | Conv1-2 | 32 | 3 x 3 |  | 32 x 32 x 32 |
| Pooling | Pool1 |  | 2 x 2 | 2 x 2 | 16 x 16 x 32 |
| Drop out | Drop1 |  |  |  | 16 x 16 x 32 |
| Convolution | Conv2-1 | 64 | 3 x 3 |  | 16 x 16 x 64 |
| Convolution | Conv2-2 | 64 | 3 x 3 |  | 16 x 16 x 64 |
| Pooling | Pool2 |  | 3 x 3 |  | 8 x 8 x 64 |
| Dropout | Drop2 |  |  |  | 8 x 8 x 64 |
| Convolution | Conv3-1 | 128 | 3 x 3 |  | 8 x 8 x 128 |
| Convolution | Conv3-2 | 128 | 3 x 3 |  | 8 x 8 x 128 |
| Pooling | Pool3 |  | 2 x 2 | 2 x 2 | 4 x 4 x 128 |
| Drop out | Drop3 |  |  |  | 4 x 4 x 128 |
| Fully Connected | Fc4 |  | 2048 |  | 2048 x 1 |
| Fully Connected | Fc5 |  | 512 |  | 512 x 1 |
| Drop out | Drop4 |  |  |  | 512 x 1 |
| Fully Connected | Fc6 |  | 43 |  | 43 x 1 |

* + 1. ***Mô hình VGG16***

Mô hình VGG16-Net gồm có 6 block trong đó 5 block đầu thực hiện chức năng trích chọn đặc trưng và block cuối cùng để phân loại dữ liệu. Hàm kích hoạt được sử dụng là hàm ReLU activation ở tất cả các lớp trừ lớp cuối cùng (fc6) hàm kích hoạt được sử dụng là hàm softmax activation.

Ở 2 block tích chập đầu gồm có :

* 2 lớp thực hiện chức năng tích chập.
* 1 lớp thực hiện max pooling.

Ở 3 block tích chập tiếp theo gồm có:

* 3 lớp thực hiện chức năng tích chập
* 1 lớp thực hiện max pooling

Ở block cuối cùng để phân loại dữ liệu gồm có:

* 1 lớp flatten (lớp fc6).
* 3 lớp fully connected (lớp fc7, fc8, fc9).
* 2 lớp drop out ở giữa 2 lớp fully connected fc7 và fc8 và giữa 2 lớp fc8 và fc9 với tỉ lệ drop là 0.5

**BẢNG 3.3 : KIẾN TRÚC MÔ HÌNH VGG-16**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại | Tên | Số filter | Kích thước filter | stride | Kích thước ngõ ra |
| Input | Input | 3 |  |  | 32 x 32 x 3 |
| Convolution | Conv1-1 | 64 | 3 x 3 |  | 32 x 32 x 64 |
| Convolution | Conv1-2 | 64 | 3 x 3 |  | 32 x 32 x 64 |
| Pooling | Pool1 |  | 2 x 2 | 2 x 2 | 16 x 16 x 64 |
| Convolution | Conv2-1 | 128 | 3 x 3 |  | 16 x 16 x 128 |
| Convolution | Conv2-2 | 128 | 3 x 3 |  | 16 x 16 x 128 |
| Pooling | Pool2 |  | 3 x 3 |  | 8 x 8 x 128 |
| Convolution | Conv3-1 | 256 | 3 x 3 |  | 8 x 8 x 256 |
| Convolution | Conv3-2 | 256 | 3 x 3 |  | 8 x 8 x 256 |
| Convolution | Conv3-3 | 256 | 3 x 3 |  | 8 x 8 x 256 |
| Pooling | Pool3 |  | 2 x 2 | 2 x 2 | 4 x 4 x 256 |
| Convolution | Conv4-1 | 512 | 3 x 3 |  | 4 x 4 x 256 |
| Convolution | Conv4-2 | 512 | 3 x 3 |  | 4 x 4 x 256 |
| Convolution | Conv4-3 | 512 | 3 x 3 |  | 4 x 4 x 256 |
| Pooling | Pool4 |  | 2 x 2 | 2 x 2 | 2 x 2 x 512 |
| Convolution | Conv5-1 | 512 | 3 x 3 |  | 2 x 2 x 512 |
| Convolution | Conv5-2 | 512 | 3 x 3 |  | 2 x 2 x 512 |
| Convolution | Conv5-3 | 512 | 3 x 3 |  | 2 x 2 x 512 |
| Pooling | Final\_pool |  | 2 x 2 | 2 x 2 | 1 x 1 x 512 |
| Fully Connected | Fc6 |  | 512 |  | 512 x 1 |
| Fully Connected | Fc7 |  | 4096 |  | 4096 x 1 |
| Drop out | Drop7 |  |  |  | 4096 x 1 |
| Fully Connected | Fc8 |  | 4096 |  | 4096 x 1 |
| Drop out | Drop8 |  |  |  | 4096 x 1 |
| Fully Connected | Fc9 |  | 43 |  | 43 x 1 |

* + 1. ***Mô hình baseline sử dụng Normalization***

Mô hình huấn luyện của Baseline model Normalization tương tự với mô hình Baseline tuy nhiên với mô hình thực hiện thêm 1 lớp chức năng chuẩn hóa các dữ liệu. Việc chuẩn hóa dữ liệu này được thực hiện với epsilon là 10-6 và momentum là 0.99.

Ở 3 block tích chập gồm có :

* 2 lớp thực hiện chức năng tích chập.
* 1 lớp thực hiện chức năng chuẩn hóa dữ liệu
* 1 lớp thực hiện max pooling.
* 1 lớp thực hiện drop out với tỉ lệ drop là 0.2

Ở block cuối cùng để phân loại dữ liệu gồm có:

* 1 lớp flatten (lớp fc4).
* 1 lớp fully connected (lớp fc5)
* 1 lớp drop out ở giữa 2 lớp fully connected fc5 và fc6 với tỉ lệ drop là 0.5
* 1 lớp thực hiện chức năng chuẩn hóa dữ liệu sau lớp drop out.
* 1 lớp fully connected (lớp fc6)

**BẢNG 3.4 : KIẾN TRÚC MÔ HÌNH BASELINE MODEL NORMALIZATION**

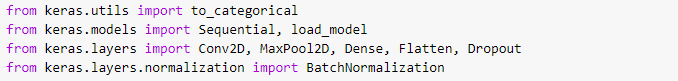
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loại | Tên | Số filter | Kích thước filter | Stride | Kích thước ngõ ra |
| Input | Input | 3 |  |  | 32 x 32 x 3 |
| Convolution | Conv1-1 | 32 | 3 x 3 |  | 32x 32 x 32 |
| Convolution | Conv1-2 | 32 | 3 x 3 |  | 32 x 32 x 32 |
| Batch-Normalization | Norm1 |  |  |  | 32 x 32 x 32 |
| Pooling | Pool1 |  | 2 x 2 | 2 x 2 | 16 x 16 x 32 |
| Drop out | Drop1 |  |  |  | 16 x 16 x 32 |
| Convolution | Conv2-1 | 64 | 3 x 3 |  | 16 x 16 x 64 |
| Convolution | Conv2-2 | 64 | 3 x 3 |  | 16 x 16 x 64 |
| Batch-Normalization | Norm2 |  |  |  | 16 x 16 x 64 |
| Pooling | Pool2 |  | 2 x 2 | 2 x 2 | 8 x 8 x 64 |
| Dropout | Drop2 |  |  |  | 8 x 8 x 64 |
| Convolution | Conv3-1 | 128 | 3 x 3 |  | 8 x 8 x 128 |
| Convolution | Conv3-2 | 128 | 3 x 3 |  | 8 x 8 x 128 |
| Pooling | Pool3 |  | 2 x 2 | 2 x 2 | 4 x 4 x 128 |
| Drop out | Drop3 |  |  |  | 4 x 4 x 128 |
| Fully Connected | Fc4 |  | 2048 |  | 2048 x 1 |
| Fully connected | Fc5 |  | 512 |  | 512 x 1 |
| Drop out | Drop5 |  |  |  | 512 x 1 |
| Batch-Normalization | Norm5 |  |  |  | 512 x 1 |
| Fully connected | Fc6 |  | 43 |  | 43 x 1 |

* 1. **Huấn luyện mô hình**

Với đề tài này, nhóm chúng tôi huấn luyện mô hình trên thư viện Keras. Đây là một mã nguồn mở dành cho Neural Network. Keras là một framework cấp cao, cung cấp một số công cụ trừu tượng, 4 module được dược sử dụng chủ yếu trong Keras bao gồm : Keras models, Keras layers, Keras losses, Keras optimizers. Có 5 bước cơ bản để huấn luyện và kiểm tra mô hình dùng các hàm đã xây dựng sẵn của Keras.

Bước 1: Khai báo các module cần thiết của thư viện Keras.

VD: Khai báo các thư viện cần thiết để thực hiện huấn luyện 3 kiến trúc bên trên



*Hình 2.5: Khai báo thư viện Keras*

Bước 2: Tạo model bằng cách sử dụng các hàm trong thư viện Keras:

VD: Tạo một Convolutional Layer 2D, Maxpooling và Dropout



*Hình 2.6: Tạo model Keras*

Bước 3: Compile model bằng lệnh sau:

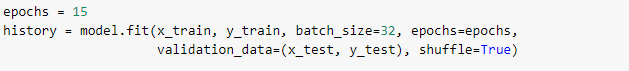


*Hình 2.7: Compile model Keras*

Trong đó:

* Hàm optimizer có thể là ‘adam’, ‘sdg’,RMSprop, …
* Hàm Loss dùng để tính toán chất lượng của model từ đó tìm cách giảm thiểu trong quá trình đào tạo.

Bước 4: Train model và test model bằng lệnh sau:



*Hình 2.8: Training model Keras*

Bước 5: Lưu model để sử dụng sau này:

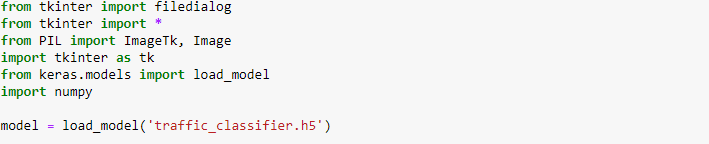


*Hình 2.9: Lưu model Keras*

* 1. **Tạo giao diện người dùng**

Sau quá trình training, chúng ta sẽ nhận được file lưu trọng số của model, lúc này chúng ta sẽ tiến hành xây dựng giao diện người dùng sử dụng thư viện Tkinter của Python và load model để nhận diện biển báo.

Bước 1: Khai báo thư viện của Tkinter và Keras



*Hình 2.10: Khai báo thư viện Tkinter và Keras để tạo giao diện người dùng*

Bước 2: Tạo mảng lưu lại nhãn của 43 lớp mẫu biển báo

VD: Tạo cho 17 biển báo



*Hình 2.11: Tạo mảng gồm các nhãn của biển báo*

Bước 3: Thiết kế giao diện



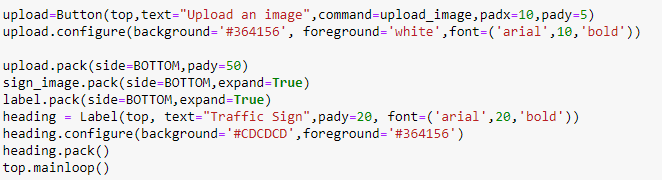
*Hình 2.12: Tạo giao diện*

Bước 4: Tạo các hàm để xử lý và đưa ra kết quả



*Hình 2.13: Tạo hàm xử lý ảnh*

Bước 5: Gọi hàm vừa khai báo và tiến hành classify ảnh



*Hình 2.14: Tiến hành xử lý ảnh*

Kết quả thu được sau khi tạo mô hình và thiết kế giao diện:



*Hình 2.15: Kết quả*

# CHƯƠNG 3: XÂY DỰNG MÔ HÌNH NHẬN DIỆN VẬT THỂ TRÊN VIDEO

* 1. **Sơ đồ khối**

Để thực hiện mô hình nhận diện biển báo trên video, nhóm chúng tôi sử dụng thuật toán YOLO.

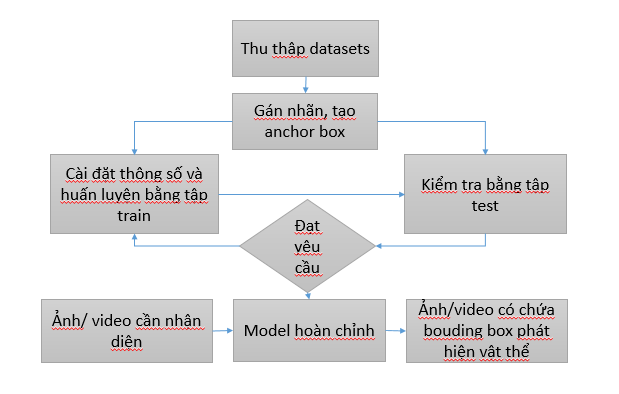
Bước 1: Tìm kiếm và thu thập datasets, tiến hành gán nhãn.

Bước 2: Phân chia dữ liệu thành tập training và tập testing.

Bước 3: Huấn luyện mô hình sử dụng dữ liệu trong tập huấn luyện.

Bước 4: Kiểm tra và đánh giá mô.

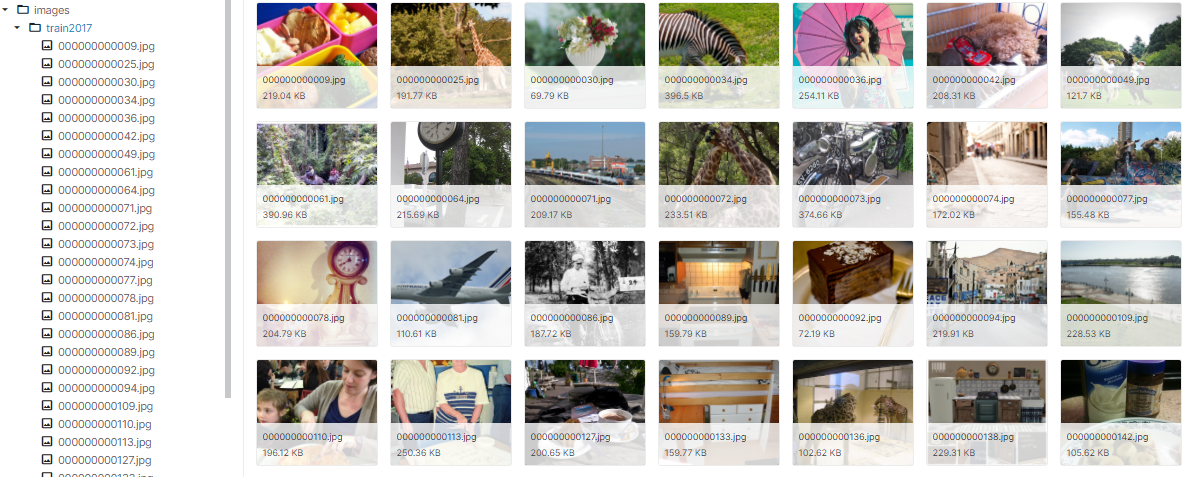
Dưới đây là sơ đồ khối tổng quát của hệ thống:



*Hình 3.1. Sơ đồ khối quá trình thực hiện mô hình nhận diện vật thể*

* 1. **Datasets**

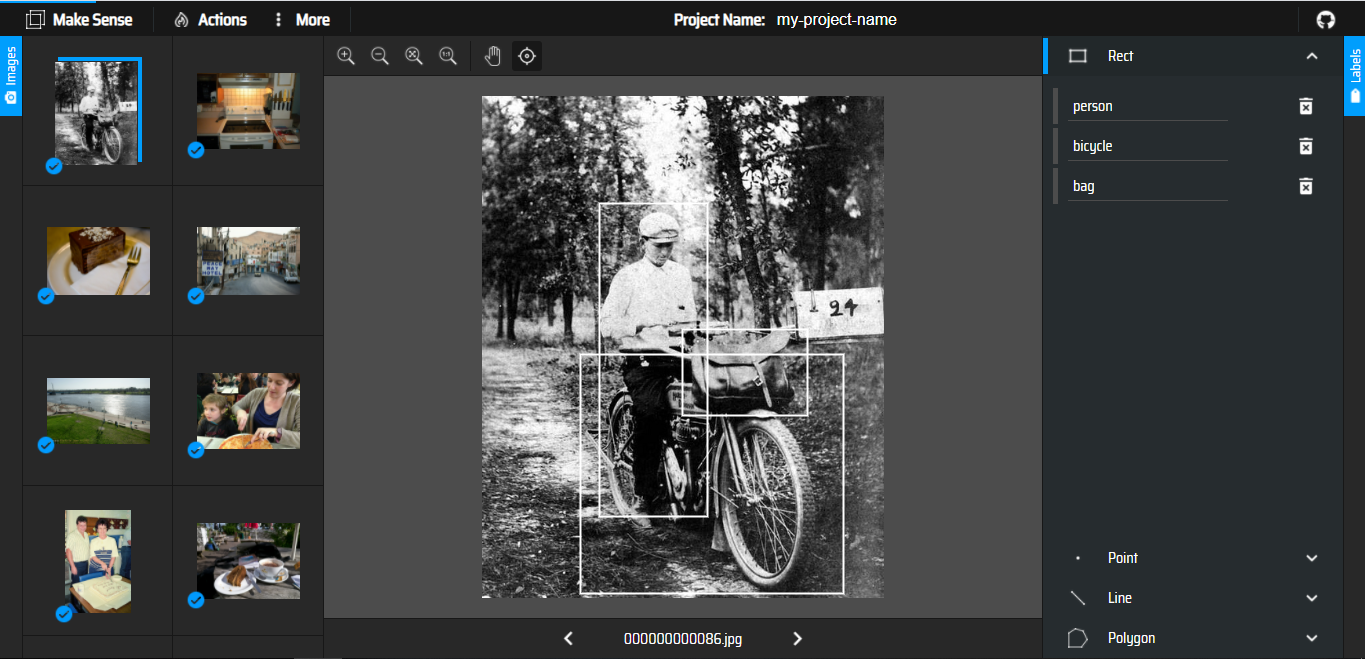
Nhóm sử dụng tập dữ liệu Coco128 được tải xuống từ Kaggle, tập datasets này bao gồm 128 hình ảnh vật thể của 80 labels khác nhau.



*Hình 3.2. Một vài hình ảnh của tập dữ liệu.*

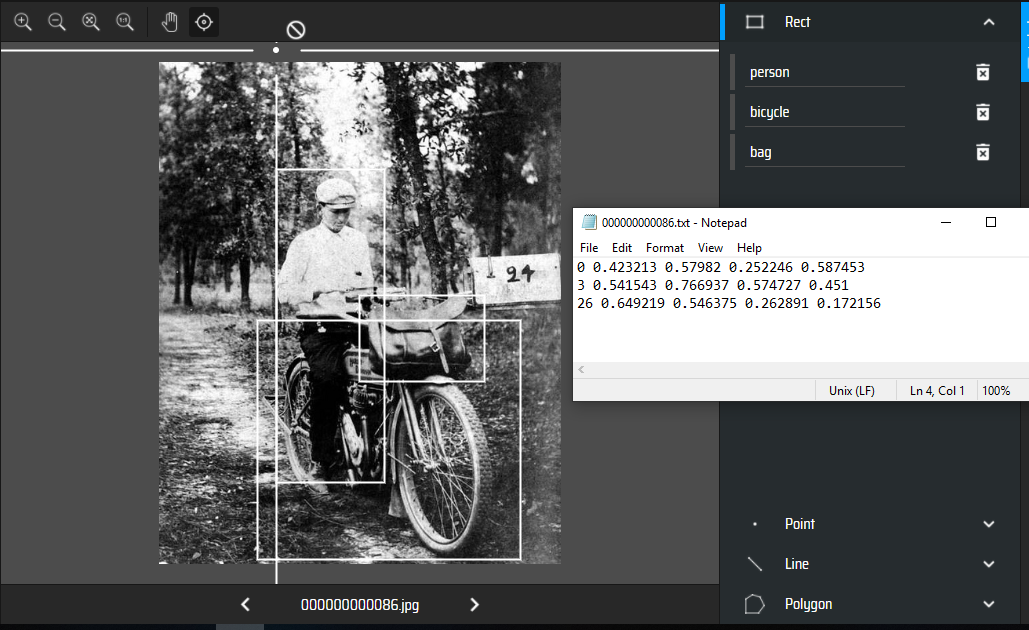
* 1. **Gán nhãn và tạo anchor box.**

Sau khi có bộ datasets, các ảnh sẽ được xác định tọa độ của anchor box và gán nhãn bằng tools tại website: *makesense.ai.* Kết quả trả về sau khi gán nhãn là một file txt chứa nhãn và các tọa độ cần thiết của một anchor box.



*Hình 3.3. Tiến hành tạo anchor box và gán nhãn cho vật thể.*

Kết quả trả về sẽ là 1 file txt chứa class và tọa độ của anchor box

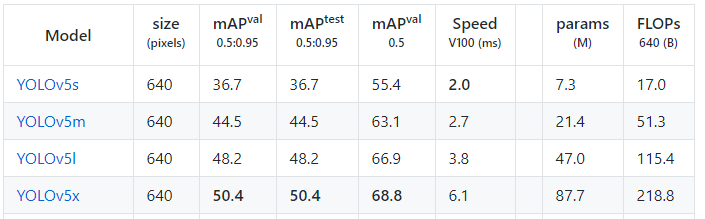


*Hình 3.4. Kết quả sau khi tiến hành gán nhãn và xác định tọa độ anchor box.*

* 1. **Huấn luyện mô hình**

Sau khi đã có tập datasets hoàn chỉnh, chúng ta sẽ tiến hành huấn luyện mô hình. Ở báo cáo này, nhóm sử dụng mô hình YOLOv5 để nhận diện vật thể.

Có nhiều tham số được sử dụng trong thuật toán YOLOv5, tuy nhiên ở đây nhóm chỉ đề cập tới một số mô hình tham số quan trọng ảnh hưởng lớn đến quá trình huấn luyện. YOLOv5 sử dụng 4 mô hình chính để huấn luyện, việc lựa chọn và sử dụng mô hình hoàn toàn phụ thuộc vào mục đích sử dụng và tài nguyên có sẵn để huấn luyện được mô hình đó. Bốn mô hình đó là YOLOv5 small, YOLOv5 medium, YOLOv5 large và YOLOv5 Xlarge.



*Hình 3.4. Bốn mô hình huấn luyện chính của YOLOv5.*

* + 1. ***Pycharm***

PyCharm là môi trường phát triển tích hợp đa nền tảng (IDE) được phát triển bởi Jet Brains và được thiết kế đặc biệt cho Python. PyCharm có mặt trên cả 3 nền tảng Windows, Linux và Mac OS.

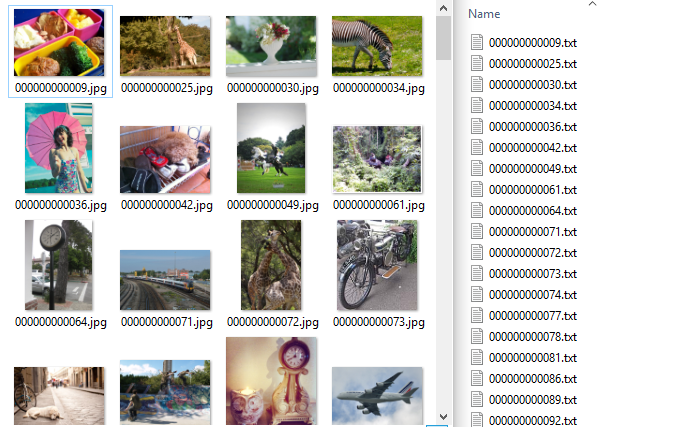
Vì sự thuận tiện cho việc lập trình python trên máy tính cá nhân nên nhóm đã sử dụng phần mềm Pycharm cho báo cáo này.

* + 1. ***Huấn luyện mô hình***

Tạo file data chứa dữ liệu các tập tin để huấn luyện, gồm:

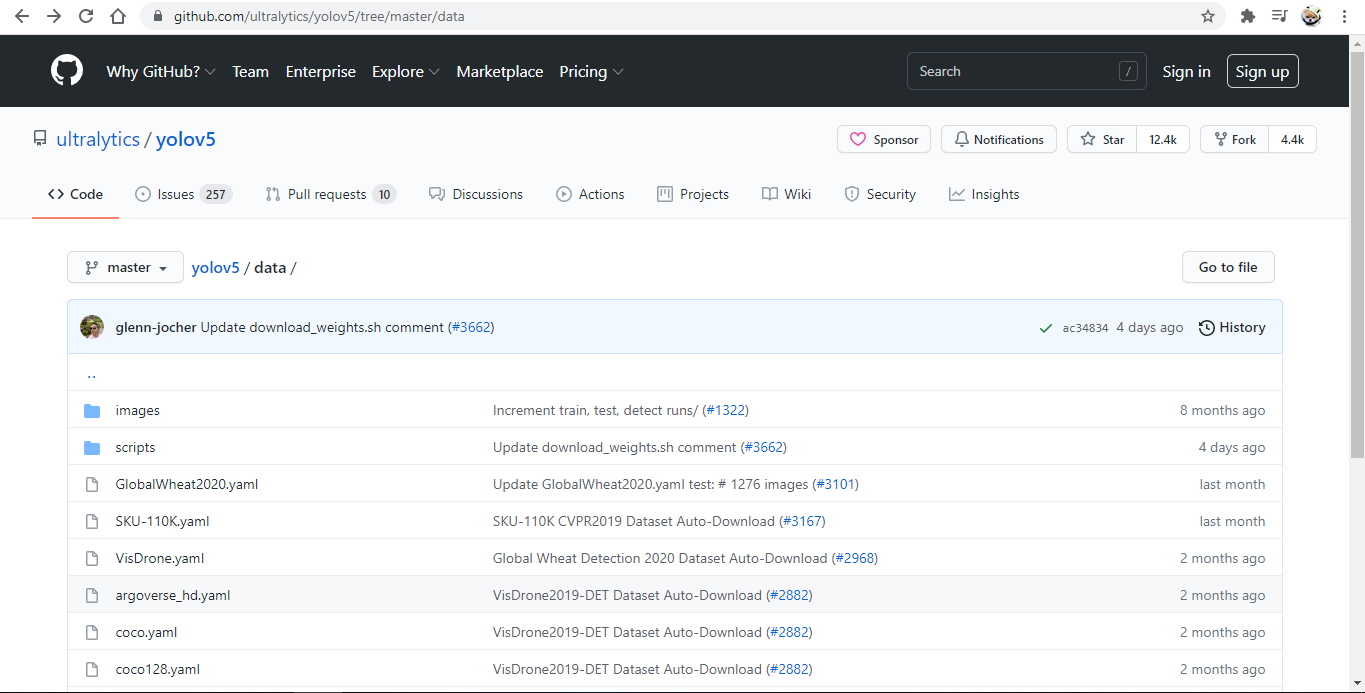
- 1 file images chứa 128 hình ảnh.

- 1 file label chứa 128 file txt chứa label và tọa độ anchor box của các vật thể trong ảnh.



*Hình 3.5: File images (bên trái) và labels (bên phải)*

Tải mô hình mẫu của tác giả tại website: github.com/ultralytics/yolov5.



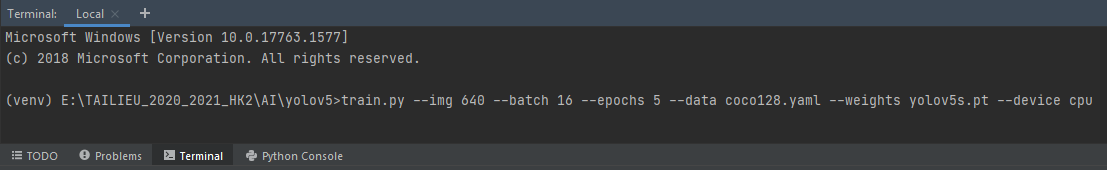
*Hình 3.6: Hình ảnh về mã nguồn data của tác giả.*

Sau khi tải về mã nguồn, YOLOv5 sẽ yêu cầu chúng ta tạo một file yaml cho bộ datasets, file này sẽ bao gồm đường dẫn của tập train, tập labels; số lượng class; tên các class.



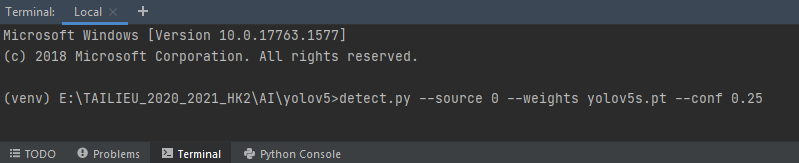
*Hình 3.7: Hình ảnh về file yaml để đưa dữ liệu training vào mô hình.*

Sau khi tạo xong bộ dữ liệu, chúng ta sẽ tiến hành training mô hình. Tại mô hình này, do cấu hình máy tính cá nhân của nhóm không được tốt nên nhóm sử dụng mô hình huấn luyện YOLOv5 Small, ảnh đầu vào là 480x640, batch size 16 và training bằng CPU.



*Hình 3.8: Tiến hành training mô hình.*

Sau quá trình training mô hình, một file có tên yolov5s.pt sẽ được tạo ra, file này chứa toàn bộ dữ liệu và parameter của quá trình huấn luyện cuối cùng. Chúng ta sẽ sử dụng file này để tiến hành bước nhận diện cuối cùng.



*Hình 3.9: Tiến hành nhận diện vật thể.*



*Hình 3.10: Kết quả mô hình nhận diện.*

# CHƯƠNG 4: KẾT LUẬN

* 1. **Kết luận**
     1. ***Mô hình CNN phân loại vật thể bằng ảnh.***

Sau khi huấn luyện và kiểm tra 3 mô hình Baseline, VGG16, Baseline với Normalization , chúng tôi đưa ra được bảng đo độ chính xác như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Độ chính xác | Tập huấn luyện | Tập kiểm định | Tập kiểm tra |
| Baseline | 97,56% | 93,55 % | 89,97% |
| VGG16 | 98,84% | 95,36% | 91,75% |
| Baseline BatchNorm | 99,5% | 99,6% | 98,76% |

Từ bảng trên có thể kết luận, mô hình baseline sử dụng batch normalization là mô hình cho hiệu quả tốt nhất.

* + 1. ***Mô hình CNN phát hiện vật thể bằng video.***

Sau khi thực hiện huấn luyện dữ liệu. Độ sai lệch của mô hình còn lại : 0.2544

Độ chính xác (AP) đối với tập kiểm tra: 79.34%. Độ chính xác không cao do dữ liệu có số lượng hình ảnh thấp, đa số các label chủ yếu là “person” nên khi phát hiện thì cho kết quả dự đoán khá chính xác. Còn những labels ít xuất hiện khác như car, bag, donuts,… ít xuất hiện trong tập train nên khi phát hiện thì khá thiếu chính xác.

* 1. **Hướng phát triển**

Mặc dù thuật toán phân loại và phát hiện vật thể của chúng tôi đã đạt được độ chính xác tương đối cao, nhưng việc áp dụng vào hệ thống thực tế chưa thực sự tốt. Việc kết hợp 2 thuật toán phân loại và phát hiện vật thể còn chưa thể kết hợp hoàn chỉnh. Trong giới hạn về mặt kiến thức và thời gian thực hiện, chúng tôi vẫn tồn tại nhiều khuyết điểm cho đề tài này.

Để phát triển đề tài này, chúng tôi có các đề xuất như sau:

* Sử dụng camera chuyên dụng để thu thập thêm dữ liệu, hỗ trợ việc huấn luyện mô hình tốt hơn.
* Sử dụng máy có cấu hình GPU tốt cho phép tăng tốc độ huấn luyện và tăng số lớp có thể nhận diện.
* Xây dựng 1 app điện thoại, cho phép người dùng tra cứu thông tin thông qua ảnh chụp trong thời gian thực.
* Bên cạnh phần mềm, có thể xây dựng phần cứng để gắn vào các phương tiện ô tô, xe gắn máy.
* Xây dựng một dữ liệu lớn hơn với cả thời điểm dữ liệu được lấy vào ban đêm khi ánh sang yếu.

Trên đây là những định hướng phát triển cho để tài nhận diện biển vật thể trong báo cáo này.

TÀI LIỆU THAM KHẢO

[1] Joseph Redmon, Santosh Divvalay, Ross Girshick, Ali Farhadiy, (2016) ,*You Only Look Once: Unified, Real-Time Object Detection*, University of Washington.

[2] Lilianweng, (2018), *Object Detection Part 4: Fast Detection Models*, lilianweng.github.io

[3] Yi-Qi Huang, Jia-Chun Zheng, Shi-Dan Sun, Cheng-Yi Chen, (2020) *,Optimized YOLOv3 Algorithm and Its Application in Traffic Flow Detections.*

[4] Alexey Bochkovskiy, (2020), *YOLOv4: Optimal Speed and Accuracy of Object Detection.*

[5] Vũ Hữu Tiệp,(2016), “Machine Learning cơ bản”, Nhà xuất bản khoa học và kỹ thuật

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, (2016) *You Only Look Once: Unified, Real-Time Object Detection*, University of Washington, Allen Institute for AI, Facebook AI Research.

[7] Alexey Bachkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao,(2020), *YOLOv4: Optimal Speed and Accuracy of Object Detection*, Institute of Information science Academia Sinica.

[8] Sik-Ho Tsang,(2018) *Review: DenseNet - Dense Convolution Network (Image Classification)*, towardsdatascience.com.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, (2017) *Feature Pyramid Networks for Object Detectio*, Facebook AI Research (FAIR), Cornell University and Cornell Tech.

[10] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia,(2018) *Path Aggregation Network for Instance Segmentation*, The Chinese University of Hong Kong, Peking University, SenseTime Research, YouTu Lab, Tencent.

[11] Joseph Redmon, Ali Farhadi,(2018) *YOLOv3: An incremental Improvement* , University of Washington.

[12] Mykola, (2018), *GTSRB - German Traffic Sign Recognition Benchmark,* https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign.