



Báo cáo dự án Deep Learning

Dề tài: Nhận dạng biển báo giao thông

Đặng Chí Trung, Lâm Văn Sang Em, Ngô Phương Nhi

Supervisor: Đinh Viết Sang

AI ACADEMY - VIN BIGDATA

Mục lục

1	Tổng quan bài toán	1
2	Cơ sở lý thuyết	3
2.1	Faster R-CNN	3
2.1.1	Kiến trúc mạng	3
2.1.2	Region Proposal Network	4
2.1.3	Anchor box	5
2.1.4	Hàm mất mát	6
2.2	YOLO	6
2.2.1	Cấu trúc mạng	7
2.2.2	Cách yolo hoạt động	7
2.2.3	Loss Function	8
2.3	SSD	10
2.3.1	Kiến trúc mạng	10
2.3.2	Default boxes	11
2.3.3	Box matching	11
2.3.4	Hàm mất mát	12
2.4	EfficientDet	13
3	Dữ liệu	15
3.1	Bộ dữ liệu training	16
3.2	Bộ dữ liệu validating	17
3.3	Bộ dữ liệu testing	18
4	Hướng thực hiện	20
4.1	Các mô hình sử dụng	20
4.1.1	EfficientDet-D1	20
4.1.2	SSD MobileNet v2 320x320	20
4.1.3	SSD Resnet50 FPN V1 FPN 640x640 (RetinaNet50)	21
4.1.4	Yolo v5 - small, medium, large	21
4.1.5	Faster R-CNN Resnet50 V1 640x640	21
4.2	Mean Average Precision	22
5	Kết quả	23
6	Kết luận	24
6.1	Thảo luận	24
6.2	Hướng phát triển	25
6.3	Hạn chế của dự án	25
7	Phân công công việc	26
7.1	Đặng Chí Trung - nhóm trưởng	26
7.2	Lâm Văn Sang Em	26
7.3	Ngô Phương Nhi	26
8	Tài liệu tham khảo	27

1 Tổng quan bài toán

Autopilot là công nghệ liên quan đến xe tự lái, với các chip, cảm biến cũng như là camera đặt trên xe kết hợp với các công nghệ liên quan đến trí tuệ nhân tạo đã tạo ra một công nghệ mới mang tên Autopilot. Đây đang là lĩnh vực được nhiều công ty công nghệ lớn trên thế giới cũng như là Việt Nam quan tâm đến và đầu tư. Autopilot gồm rất nhiều bài toán con, mỗi bài toán vẫn còn nhiều thách thức chưa giải quyết được. Trong số các bài toán đó thì bài toán phát hiện biển báo giao thông (traffic sign detection) đang rất được quan tâm gần đây, thể hiện qua sự xuất hiện ở các cuộc thi lớn ở Việt Nam như Zalo AI Challenge, Cuộc đua số,... để giải quyết các vấn đề còn tồn tại.

Traffic sign detection nhận ảnh đầu vào để xác định chính xác vị trí của biển báo giao thông, sau đó phân loại biển báo. Do đó, độ chính xác của việc xác định vị trí biển báo có ảnh hưởng đến độ chính xác của toàn bộ hệ thống. Rất nhiều phương pháp đã được đề xuất để giải quyết bài toán này. Các phương pháp truyền thống [12-13] dựa vào đặc trưng của biển báo như màu sắc, hình dáng, kết cấu và ứng dụng các thuật toán trong thị giác máy tính để phát hiện biển báo trong một bức ảnh. Tuy nhiên, các điều kiện ánh sáng, thời tiết trên đường, hay sự ảnh hưởng của các vật gây nhiễu xung quanh biển báo (ví dụ như các biển hiệu trên đường) khiến cho các phương pháp truyền thống khó đạt được độ chính xác cao.

Hiện nay, với sự phát triển vượt bậc của học sâu, rất nhiều mô hình học sâu đã được đề xuất [14-15] để giải quyết bài toán với độ chính xác cao hơn so với các mô hình truyền thống. Do đó, nhóm quyết định tìm hiểu các mô hình học sâu để giải quyết bài toán này. Nhóm lựa chọn sử dụng các mô hình phát hiện vật thể phổ biến như EfficientDet, , SSD và Faster R-CNN.

Bộ dữ liệu Traffic Sign Detection [9] của cuộc thi Zalo AI Challenge nhóm sử dụng cho dự án có những thách thức nhất định. Biển báo được chụp ở nhiều cự li, góc chụp khác nhau cũng như điều kiện ánh sáng khác nhau. Bên cạnh việc tìm hiểu các mô hình học sâu, nhóm cũng tìm cách xử lý và làm sạch bộ dữ liệu để tăng độ chính xác của các mô hình.

Mục tiêu của nhóm là khám phá và giải quyết các vấn đề nóng hổi của Autopilot, mà cụ thể là bài toán phát hiện biển báo giao thông. Từ đó, tích lũy được thêm những kiến thức, kinh nghiệm liên quan đến lĩnh vực Autopilot để chuẩn bị hành trang tham gia dự

án Autopilot của tập đoàn Vingroup. Bên cạnh đó, nhóm cũng muốn sử dụng cơ hội này để phát triển các kiến thức về các mô hình phát hiện đối tượng phổ biến, nâng cao kỹ năng làm việc nhóm, giao tiếp và phát triển dự án mang tính ứng dụng cao.

Qua quá trình nghiên cứu và thử nghiệm, mô hình YOLO v5 cho ra kết quả tốt nhất với $mAP = 38$ trên bộ public test của Zalo AI Challenge Traffic Sign Detection.

2 Cơ sở lý thuyết

Để xử lí bài toán phát hiện biển báo giao thông (Traffic Sign Detection), nhiều nghiên cứu đã vận dụng các hệ thống/mô hình phát hiện đối tượng để giải quyết bài toán phát hiện biển báo giao thông.

Garcia et al. (2018) [5] sử dụng mô hình Faster R-CNN Inception Resnet V2 để phát hiện biển báo trong bộ dữ liệu GTSDB với mAP là 95.77. Bên cạnh đó, nhóm nghiên cứu này cũng kiểm thử các mô hình R-FCN, SSD và YOLO v2 cùng với feature extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19) vào giải quyết bài toán.

Zaki et al. (2020) [6] xử lí bài toán phát hiện biển báo trong thời gian thực dưới các điều kiện thời tiết, ánh sáng, tầm nhìn khác nhau. Họ tập trung phát triển hai mô hình cho kết quả tốt nhất Faster R-CNN Inception v2 ($mAP = 96$) và Tiny YOLO v2 ($mAP = 73$).

Zhu et al. (2016) [7] tạo ra một bộ dữ liệu phát hiện biển báo giao thông mới với tên gọi Tsinghua-Tencent 100K. Bộ dữ liệu bao gồm ảnh thu thập dưới điều kiện thời tiết và ánh sáng khác nhau. Họ cũng phát triển một mô hình CNN mới dựa trên Overfeat nhằm phát hiện đối tượng có diện tích nhỏ hơn so với diện tích ảnh.

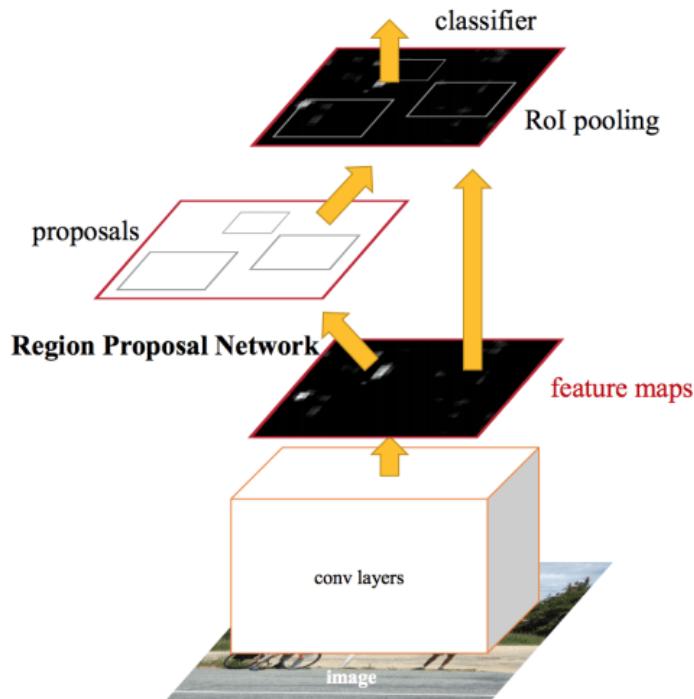
2.1 Faster R-CNN

Mô hình Faster R-CNN [3] được Ren et al. đề xuất vào năm 2016. So với mô hình RCNN và mô hình Fast R-CNN, Faster R-CNN đã được cải thiện thêm về tốc độ huấn luyện và phát hiện vật thể. Faster R-CNN sử dụng 1 mạng con gọi là RPN (Region Proposal Network) với mục đích trích xuất ra các vùng có khả năng chứa đối tượng từ ảnh (hay còn gọi là ROI - Region of Interest), khác hoàn toàn với cách xử lý của 2 mô hình anh em trước đó là RCNN và Fast-RCNN.

2.1.1 Kiến trúc mạng

Faster R-CNN bao gồm mạng CNN feature extraction, hay là pretrained model. Sau đó, là hai subnetwork, Region Proposal Network (RPN) và ROI pooling. Ảnh được đưa qua pretrained model để lấy feature map. Feature map được dùng cho RPN để trích xuất các

region proposal. Sau khi lấy được vị trí các region proposal thì đưa qua ROI pooling để dự đoán nhãn của vật.

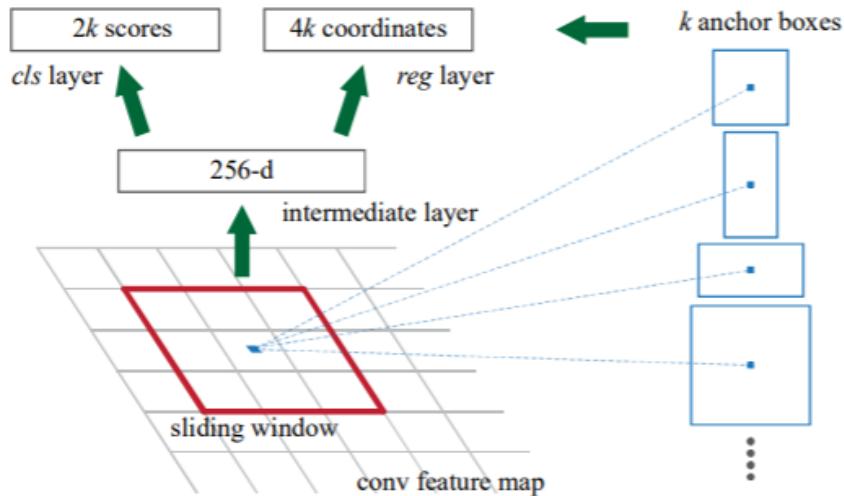


Hình 2.1: Mô hình Faster R-CNN

2.1.2 Region Proposal Network

Để tạo ra region proposal, chúng ta sử dụng sliding window kích thước $n \times n$ trượt trên feature map (Hình 2.3). Mỗi sliding window sẽ được ảnh xạ đến một feature với số chiều thấp hơn. Feature này là đầu vào của 2 lớp fully connected: box regression layer (*reg*) để dự đoán vị trí của region và box classification layer (*cls*) để dự đoán xem vùng này chứa vật hay không.

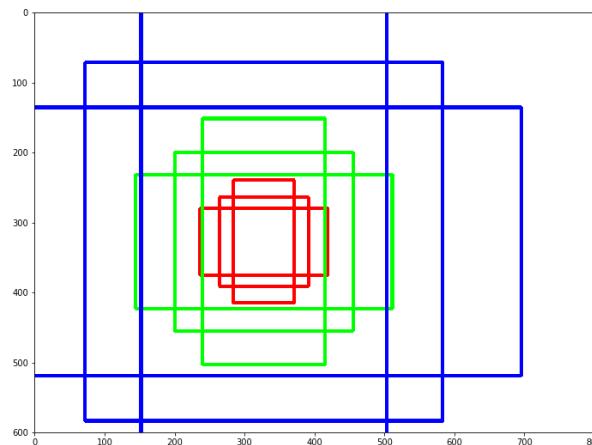
Tại mỗi vị trí của sliding window, mô hình dự đoán đồng thời *k* region proposal. Vây lớp *reg* có $4k$ đầu ra dự đoán vị trí của k proposal, *cls* layer có $2k$ đầu ra dự đoán xác suất chứa vật thể của proposal.



Hình 2.2: Region Proposal Network

2.1.3 Anchor box

Anchor box là một khái niệm quan trọng trong Faster R-CNN. Anchor box tạo ra một tổ hợp bounding box được định nghĩa trước với kích thước và tỉ lệ khác nhau để dễ dàng nhận dạng các vật khác nhau. Anchor box được đặt ở tâm sliding window. Dưới đây (Hình ??) là ví dụ về anchor box với 3 kích thước $[128 \times 128, 256 \times 256, 512 \times 512]$ và 3 tỉ lệ $[1 : 1, 1 : 2, 2 : 1]$ tại tâm $(320, 320)$ của ảnh kích thước 680×680 .



Hình 2.3: Ví dụ về 9 anchor box của ảnh kích thước 680×680

Tác giả sử dụng 3 kích thước và 3 tỉ lệ, nên $k = 9$. Với feature map kích thước $m \times n$, tổng số anchor box được sinh ra là $m \times n \times 9$. Các anchors này sẽ được gán mác là positive hoặc negative dựa vào diện tích overlap với ground truth box theo luật như sau.

Anchor được đánh giá là positive nếu: (i) anchor có IoU cao nhất với một groundtruth box, hoặc (ii) anchor có IoU cao hơn 0.7 với groundtruth box bất kỳ. Các anchor không thỏa mãn 2 điều kiện nêu trên thì được gán nhãn negative, không được đánh giá trong quá trình training object.

2.1.4 Hàm mất mát

Hàm mất mát của Faster R-CNN được định nghĩa theo công thức sau

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Với i là index của anchor trong mini-batch và p_i là xác suất dự đoán anchor i là một đối tượng. $p_i^* = 1$ nếu anchor là positive, và là không khi anchor là negative.

t_i là một vector 4 chiều biểu diễn giá trị tọa độ của bounding box đã được dự đoán.

t_i^* là vector 4 chiều biểu diễn giá trị tọa độ của groundtruth box tương ứng với positive anchor.

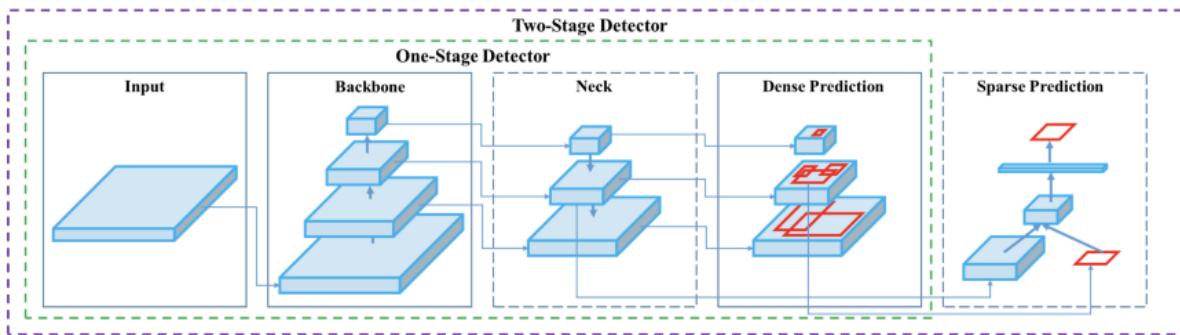
L_{cls} là log loss của 2 class (object và non-object).

L_{reg} dùng Smooth L_1 Loss.

2.2 YOLO

YOLO [1] là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

2.2.1 Cấu trúc mạng



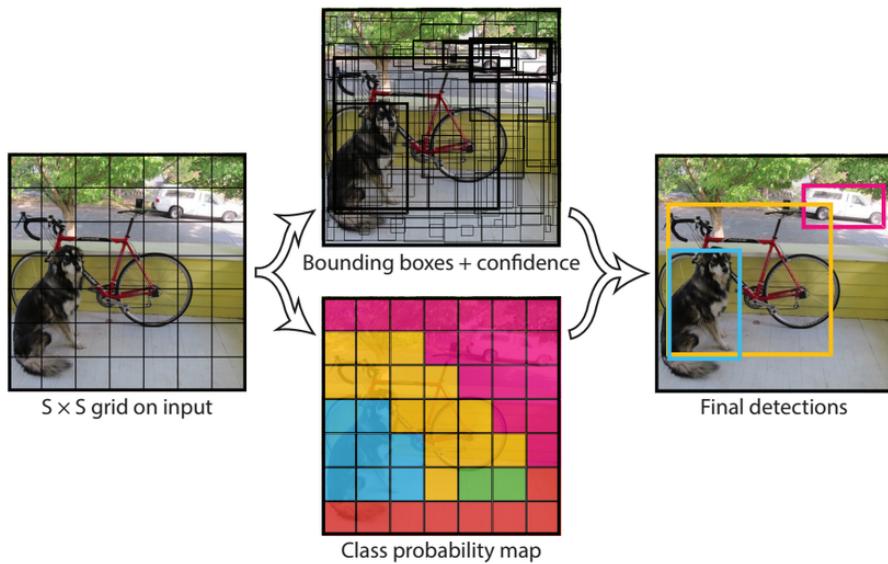
Hình 2.4: Mô hình You Only Look Once

Cấu trúc mạng của YOLO gồm 4 phần:

- Mạng Backbone: Trong YOLOv5 [8], mạng CSP được dùng để trích xuất đặc trưng từ ảnh đầu vào. Sự thay đổi đã giúp tăng tốc thời gian xử lý đáng kể đối với các cấu trúc mạng nhiều tham số.
- Mạng Neck: Mạng Neck hoạt động dựa trên khái niệm pyramid (FPN) giúp cải thiện độ chính xác của mô hình đối với cái vật có kích thước nhỏ
- Mạng Head: Nhiệm vụ chính của mạng này là phần nhận diện vật thể. Đầu ra của mạng sẽ là xác suất của lớp, độ tự tin của box và 4 kích thước chính của bounding box.

2.2.2 Cách yolo hoạt động

Ý tưởng chính của YOLOv1 là chia ảnh thành một lưới các ô (grid cell) với kích thước $S \times S$ (mặc định là 7×7). 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 $x, y, w, h, \text{confidence}$, lần lượt là tọa độ tâm (x, y), chiều rộng, chiều cao và độ tự tin của dự đoán. Với grid cell trong lưới $S \times S$ kia, mô hình cũng dự đoán xác suất rơi vào mỗi class.



Hình 2.5: Cách hoạt động của YOLO

Độ tự tin của dự đoán ứng với mỗi bounding box được định nghĩa là $p(\text{Object}) \times IoU$ trong đó $p(\text{Object})$ là xác suất có vật trong cell và IoU là intersection over union của vùng dự đoán và ground truth.

Xác suất rơi vào mỗi class cho một grid cell được ký hiệu $p(\text{Class}_i | \text{Object})$. 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 $S \times S \times (5 \times B + C)$.

2.2.3 Loss Function

Hàm lỗi trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính. Cụ thể hơn nó là tổng độ lỗi của 3 thành phần sau :

- Độ lỗi của việc dự đoán loại nhãn của object - Classification loss Classification loss - độ lỗi của việc dự đoán loại nhãn của object, hàm lỗi này chỉ tính trên những ô vuông có xuất hiện object, còn những ô vuông khác ta không quan tâm. Classification loss được tính bằng công thức sau:

$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{I}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}_i(c))^2$$

Trong đó:

\mathbb{I}_i^{obj} : bằng 1 nếu ô vuông đang xét có object ngược lại bằng 0

$\hat{p}_i(c)$: là xác xuất có điều của lớp c tại ô vuông tương ứng mà mô hình dự đoán

- Độ lỗi của dự đoán tọa độ tâm, chiều dài, rộng của boundary box (x, y, w, h) (Localization loss) là hàm lỗi dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của so với vị trí thực tế từ dữ liệu huấn luyện của mô hình. Lưu ý rằng chúng ta không nên tính giá trị hàm lỗi này trực tiếp từ kích thước ảnh thực tế mà cần phải chuẩn hóa về $[0, 1]$ so với tâm của bounding box. Việc chuẩn hóa này kích thước này giúp cho mô hình dự đoán nhanh hơn và chính xác hơn so với để giá trị mặc định của ảnh. Giá trị hàm Localization loss được tính trên tổng giá trị lỗi dự đoán tọa độ tâm (x, y) và (w, h) của predicted bounding box với ground-truth bounding box. Tại mỗi ô có chưa object, ta chọn 1 boundary box có IOU (Intersect over union) tốt nhất, rồi sau đó tính độ lỗi theo các boundary box này. Giá trị hàm lỗi dự đoán tọa độ tâm (x, y) của predicted bounding box và (\hat{x}, \hat{y}) là tọa độ tâm của truth bounding box được tính như sau :

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

Giá trị hàm lỗi dự đoán (w, h) của predicted bounding box so với truth bounding box được tính như sau :

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

- Độ lỗi của việc dự đoán bounding box đó chứa object so với nhãn thực tế tại ô vuông đó - Confidence loss Là độ lỗi giữa dự đoán boundary box đó chứa object so với nhãn thực tế tại ô vuông đó. Độ lỗi này tính trên cả những ô vuông chứa object và không chứa object.

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobject} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

- Total loss Tổng lại chúng ta có hàm lỗi là tổng của 3 hàm lỗi trên hay

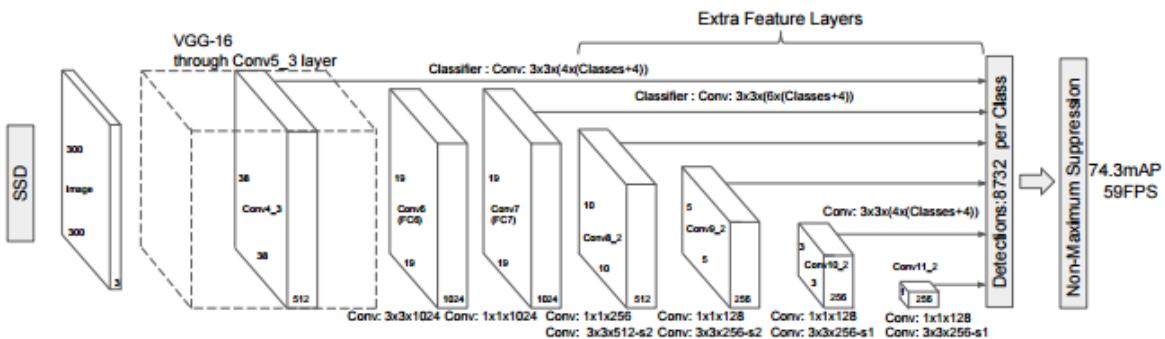
$$L_{total} = L_{classification} + L_{localization} + L_{confidence}$$

2.3 SSD

Single Shot Multibox Detector (SSD) [2] được Liu et al. (2016) đề xuất. Mô hình xây dựng một tổ hợp default boxes với tỉ lệ và kích thước khác nhau trên từng vị trí của feature map. Bên cạnh đó, mô hình cũng kết hợp feature map ở các độ phân giải khác nhau làm kết quả dự đoán để giải quyết vấn đề phát hiện vật với các kích thước khác nhau.

2.3.1 Kiến trúc mạng

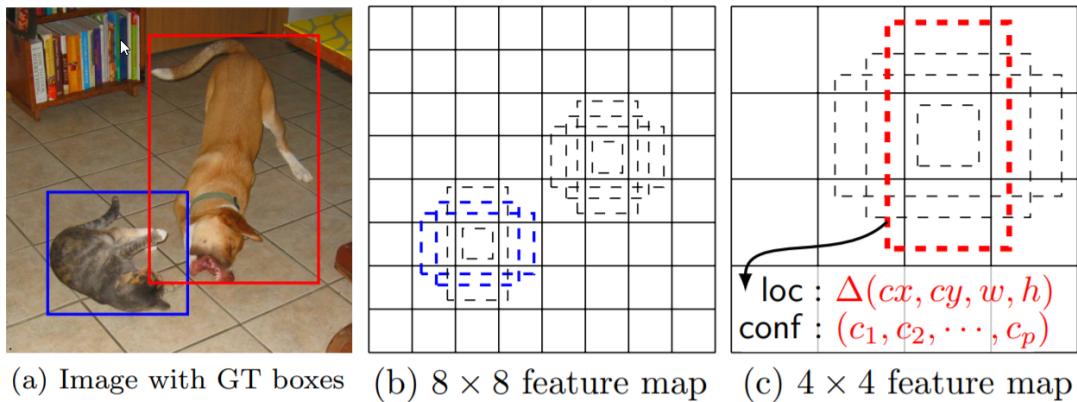
SSD được xây dựng dựa trên base network là VGG16. Sau khi thu được feature map ở base network, những convolution layer được thêm vào để tạo thêm các feature map với kích thước giảm dần. Kết hợp feature map ở các kích thước khác nhau cho phép dự báo và nhận diện vật thể ở nhiều hình dạng kích thước khác nhau. Những feature map có kích thước lớn sẽ phát hiện tốt các vật thể nhỏ và các feature map kích thước nhỏ giúp phát hiện tốt hơn các vật thể lớn. Chi tiết mô hình được biểu diễn ở hình bên dưới



Hình 2.6: Mô hình Single Shot MultiBox Detector

2.3.2 Default boxes

Tương tự như anchor box trong Faster R-CNN, SSD xây dựng một tổ hợp default box cố định với kích thước và tỉ lệ khác nhau trên các cell của feature map. Mỗi cell có từ 4 đến 6 box với kích thước và tỉ lệ khác nhau (Hình 2). Giả sử trên một feature map kích thước $m \times n$, tại mỗi cell khởi tạo k box, SSD sẽ tính toán để phân loại C lớp. Với mỗi default box, SSD sẽ dự báo 4 offset (cx, cy, w, h) để tìm ra vị trí và kích thước hộp đúng với ground truth nhất. Vậy tổng số tham số để huấn luyện trên mỗi feature map là $(C + 4) \times k \times m \times n$.



Hình 2.7: Mô hình Single Shot MultiBox Detector

2.3.3 Box matching

Trong quá trình huấn luyện, các default box với kích thước và tỉ lệ khác nhau được map với ground truth box. Đầu tiên, với mỗi ground truth box, chọn một default box với tỉ lệ IoU cao nhất. Điều này đảm bảo là mỗi một ground truth sẽ có ít nhất một default box được huấn luyện. Vì số lượng ground truth box nhỏ hơn số lượng default box rất nhiều, ta tiếp tục map các default box còn lại với ground truth box có $\text{IoU} > 0.5$. Tác giả giải thích rằng, việc này sẽ đơn giản hóa quá trình học, cho phép mô hình dự đoán nhiều kết quả cho một ground truth.

2.3.4 Hàm mất mát

Tác giả xây dựng hai hàm mất mát riêng biệt cho hai công việc phân loại (L_{conf}) và xác định (L_{loc}). Hàm mất mát của SSD là tổng hợp của hai hàm mất mát trên theo công thức

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

N ở đây là tổng số default boxes được gán với ground truth. Nếu $N = 0$ thì giá trị của hàm mất mát là 0. c là vector điểm phân loại của các lớp, l là box dự đoán và g là box ground truth.

Hàm mất mát xác định (localization loss) được tính theo smooth L1 giữa box dự đoán (l) và box ground truth (g). Chỉ những hộp được gán với ground truth mới bị phạt. Default boxes không được gán với ground truth thì giá trị mất mát là 0.

The localization loss between the predicted box l and the ground truth box g is defined as the smooth L1 loss with cx, cy as the offset to the default bounding box d of width w and height h .

$$\begin{aligned} L_{loc}(x, l, g) &= \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m) \\ \hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^w & \hat{g}_j^{cy} &= (g_j^{cy} - d_i^{cy})/d_i^h \\ \hat{g}_j^w &= \log\left(\frac{g_j^w}{d_i^w}\right) & \hat{g}_j^h &= \log\left(\frac{g_j^h}{d_i^h}\right) \end{aligned}$$

$$x_{ij}^p = \begin{cases} 1 & \text{if } IoU > 0.5 \text{ between default box } i \text{ and ground true box } j \text{ on class } p \\ 0 & \text{otherwise} \end{cases}$$

Hình 2.8: Localization Loss của SSD ?

Đối với hàm mất mát phân loại (classification loss), tác giả sử dụng softmax trên tất cả các lớp

$$L_{conf}(x, c) = - \sum_{i \in Pos}^N N_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg}^n \log(\hat{c}_j^0)$$

với

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

2.4 EfficientDet

EfficientDet [4], đề xuất bởi Tan et al. (2020), được thiết kế để cải thiện độ chính xác và phù hợp với nguồn tài nguyên tính toán hạn chế với nhiều đặc điểm mới được cải thiện. Thứ nhất, mô hình được các tác giả đề xuất thêm weighted bi-directional feature pyramid network (BiFPN) để làm cho việc multi-scale feature fusion dễ dàng và nhanh chóng. Thứ hai, kết hợp các phương pháp scaling đồng đều như resolution, depth, và width cho tất cả backbone, feature network, và box/class prediction networks trong cùng một thời gian. Tác giả cải tiến mô hình Simplified PANet để tạo ra Bidirectional Feature Pyramid Network

Kiến trúc của mô hình như hình bên dưới:

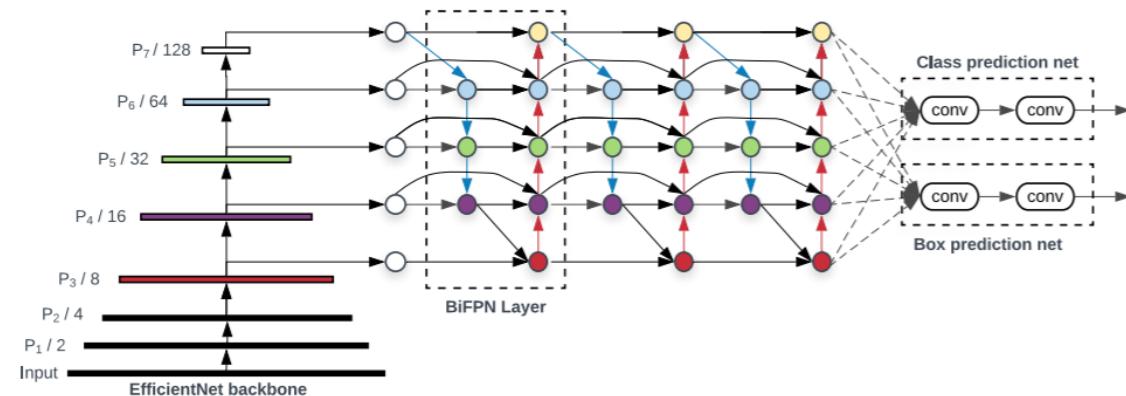


Figure 3: **EfficientDet architecture** – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Hình 2.9: Kiến trúc mạng EfficientDet

Backbone network: sử dụng lại giống các hệ số scaling width/depth của EfficientNet-B0 đến B6

Tăng BiFPN Width (số kênh) theo cấp số nhân, tăng depth (chiều sâu của mạng) theo tuyến tính bởi vì cần làm tròn depth đến số nguyên nhỏ.

$$W_{bifpn} = 64 \cdot (1.35^\phi), \quad D_{bifpn} = 3 + \phi \quad (1)$$

Box/Class prediction network: Width được giữ giống như BiFPN nhưng depth (chiều sâu của mạng) thì được tăng tuyến tính

$$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor \quad (2)$$

Input image resolution: cần phải chia hết cho 128

$$R_{input} = 512 + \phi \cdot 128 \quad (3)$$

Theo các công thức (1) (2) (3) thì ta thu được các mạng EfficientDet-D0 ($\phi = 0$) đến D7 ($\phi = 7$).

	Input size R_{input}	Backbone Network	BiFPN		Box/class
			#channels W_{bifpn}	#layers D_{bifpn}	#layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D7 ($\phi = 7$)	1536	B6	384	8	5
D7x	1536	B7	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D6 –** ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN*, *box/class net*, and *input size* are scaled up using equation 1, 2, 3 respectively.

Hình 2.10: Các mô hình EfficientDet từ D0 đến D7

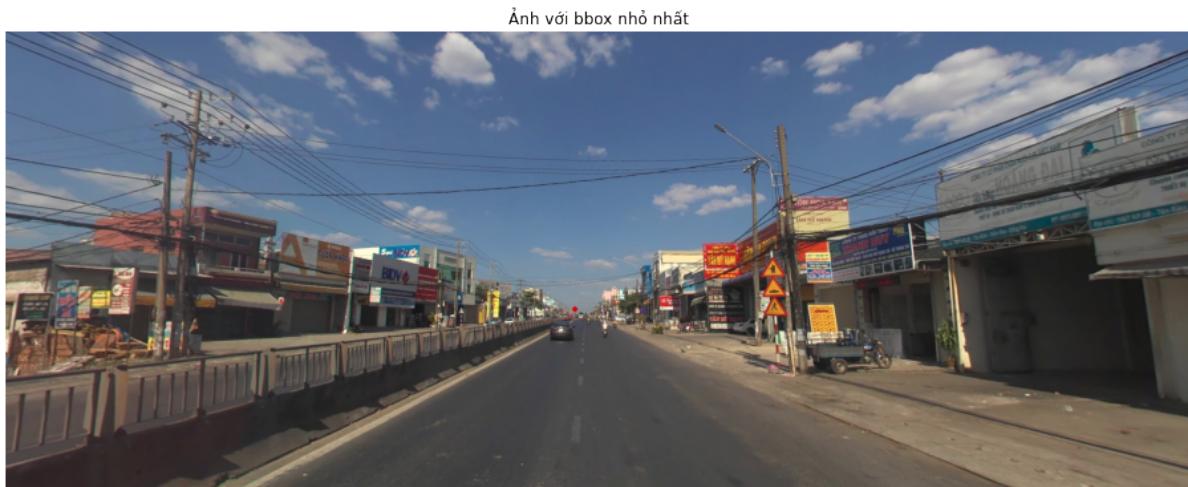
3 Dữ liệu

Bộ dữ liệu nhóm sử dụng được lấy từ Zalo AI Challenge 2020 về Traffic Sign Detection [9]. Những hình ảnh trong bộ dữ liệu được cào từ Google Street View Map.

Bộ dữ liệu bao gồm 4500 ảnh training có gán nhãn và 3521 ảnh testing không có nhãn. Tất cả các ảnh có cùng kích thước là 1622x626.

Có tất cả 11000 bounding box trong bộ dữ liệu training, trung bình có 2.44 bbox trong mỗi ảnh. Bộ dữ liệu được chia thành 7 nhóm: cấm ngược chiều, cấm dừng và đỗ, cấm rẽ, giới hạn tốc độ, cấm còn lại, nguy hiểm và hiệu lệnh.

Trong bộ dữ liệu tồn tại những groundtruth box chỉ chiếm 0.0004% diện tích ảnh (Hình 3.1). Trên thực tế, hệ thống xe tự lái không cần phát hiện biển báo ở một cự li quá xa như vậy. Hơn nữa, với các mô hình nhóm có thể huấn luyện dưới tài nguyên và thời gian cho phép, nhóm quyết định loại bỏ những ground truth box có kích thước dưới 40 pixel vuông, chiếm 0.004% diện tích ảnh.



Hình 3.1: Ảnh có groundtruth box với diện tích bằng 4. Box là chấm đỏ ở trên ô tô.

Bên cạnh đó, trong quá trình khai phá bộ dữ liệu, nhóm cũng tìm thấy nhiều trường hợp mà một biển báo được gán nhiều ground truth box cùng nhãn (Hình 3.2). Điều này khiến cho việc huấn luyện mô hình thêm phần khó khăn. Nhóm đã loại bỏ những box định nghĩa cùng một biển báo với IoU trên 0.7.



Hình 3.2: Ví dụ ảnh có nhiều ground truth box cho cùng một biển báo. Ảnh này có 20 groundtruth box trong khi chỉ có 5 biển báo.

Sau khi loại bỏ những groundtruth box với chiến lược nêu ở trên, bộ dữ liệu huấn luyện được trích ra để làm bộ dữ liệu đánh giá.

Bên cạnh đó, nhóm cũng gán nhãn bộ dữ liệu public testing gồm 579 ảnh.

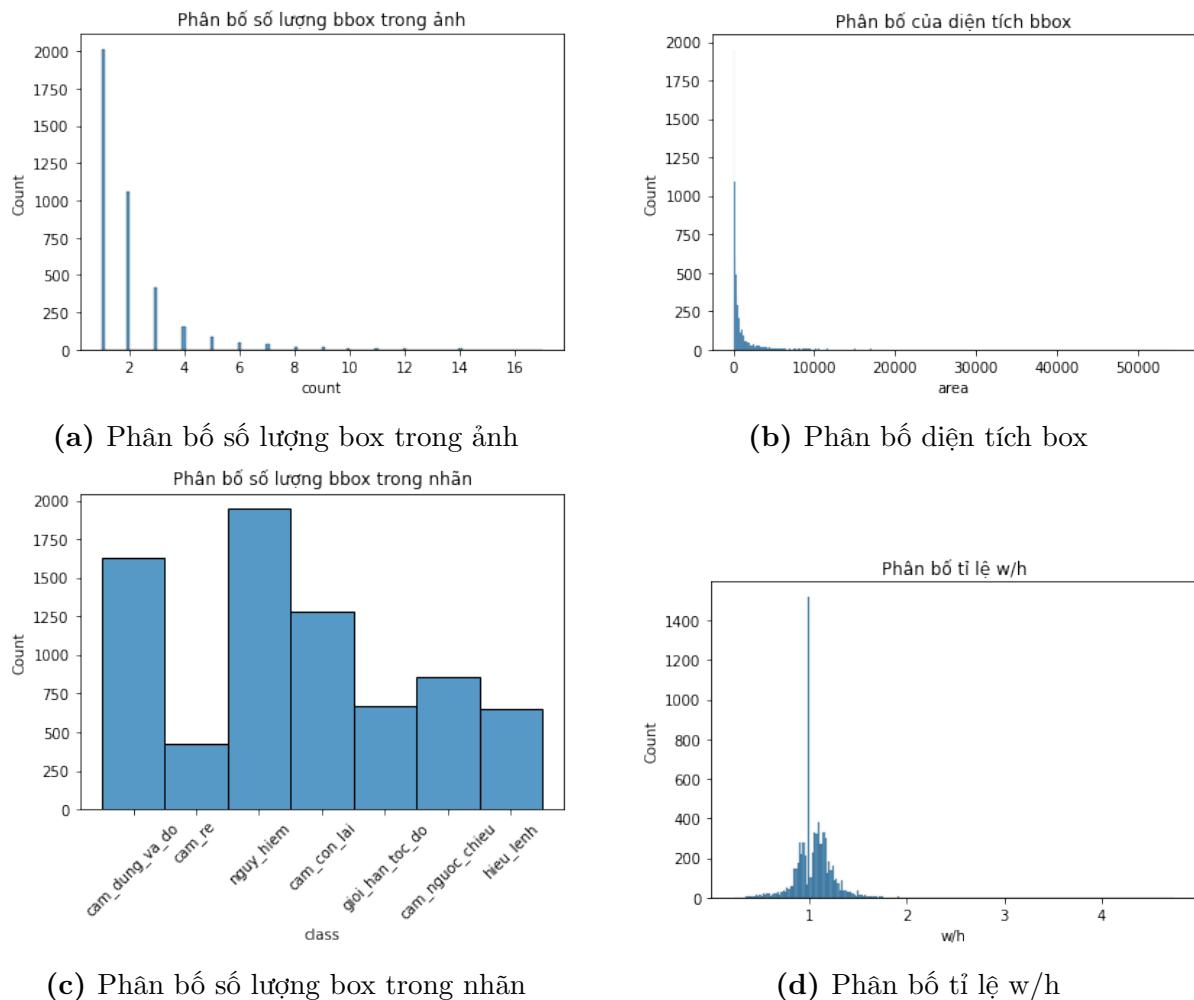
3.1 Bộ dữ liệu training

Tập training bao gồm 3861 ảnh với 7464 groundtruth. Trung bình mỗi ảnh có 1.933 box. Hầu hết các ảnh có từ 1-2 box. Có 2015 ảnh có 1 box, và 26 ảnh có 10 box trở lên. Ảnh có nhiều box nhất bao gồm 17 box (Hình 3.3a).

Box có diện tích lớn nhất là 54526 chiếm 5.37%. Box có diện tích nhỏ nhất là 40 chiếm 0.004%. Diện tích trung bình của box là 1181.38 chiếm 0.12% (Hình 3.3b).

Lớp nguy hiểm là lớp có nhiều box nhất, gồm 1950 box. Lớp cấm rẽ có ít box nhất, gồm 429 box. Trung bình mỗi lớp có 1066 box (Hình 3.3c).

Trong tập training, tỉ lệ w/h lớn nhất là 4.74, tỉ lệ nhỏ nhất 0.23. Trung bình, tỉ lệ w/h là 1.05 (Hình 3.3d).



Hình 3.3: Biểu đồ phân tích bộ dữ liệu training

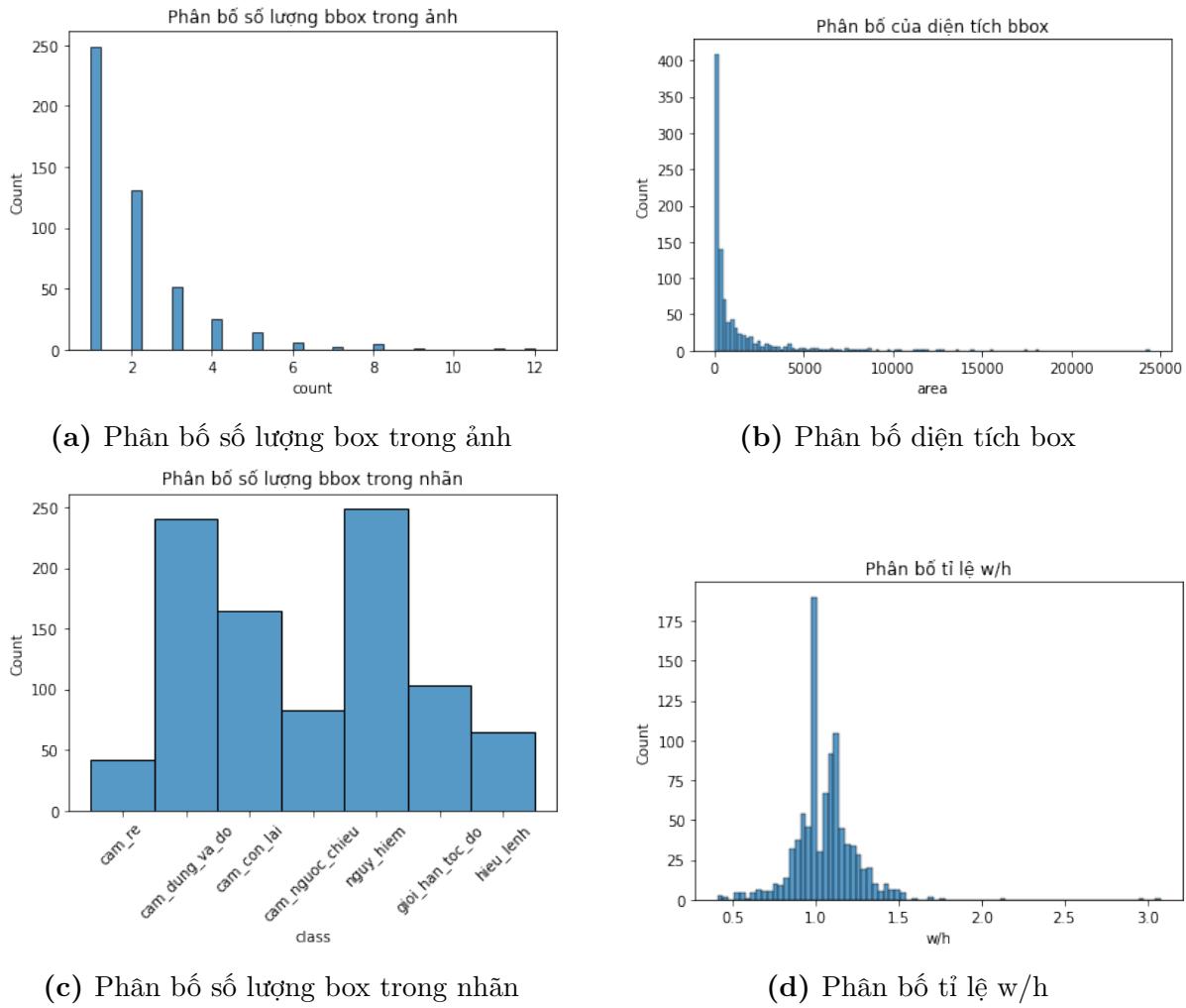
3.2 Bộ dữ liệu validating

Tập validating bao gồm 485 ảnh với 948 groundtruth. Trung bình mỗi ảnh có 1.95 box. Hầu hết các ảnh có từ 1-2 box. Có 249 ảnh có 1 box, và 2 ảnh có 10 box trở lên. Ảnh có nhiều box nhất bao gồm 12 box (Hình 3.4a).

Box có diện tích lớn nhất là 24400 chiếm 2.4%. Box có diện tích nhỏ nhất là 42 chiếm 0.004%. Diện tích trung bình của box là 1216.23 chiếm 0.12% (Hình 3.4b).

Lớp nguy hiểm là lớp có nhiều box nhất, gồm 249 box. Lớp cẩm rẽ có ít box nhất, gồm 42 box. Trung bình mỗi lớp có 135 box (Hình 3.4c).

Trong tập validating, tỉ lệ w/h lớn nhất là 3.08, tỉ lệ nhỏ nhất 0.41. Trung bình, tỉ lệ w/h là 1.06 (Hình 3.4d).



Hình 3.4: Biểu đồ phân tích bộ dữ liệu validating

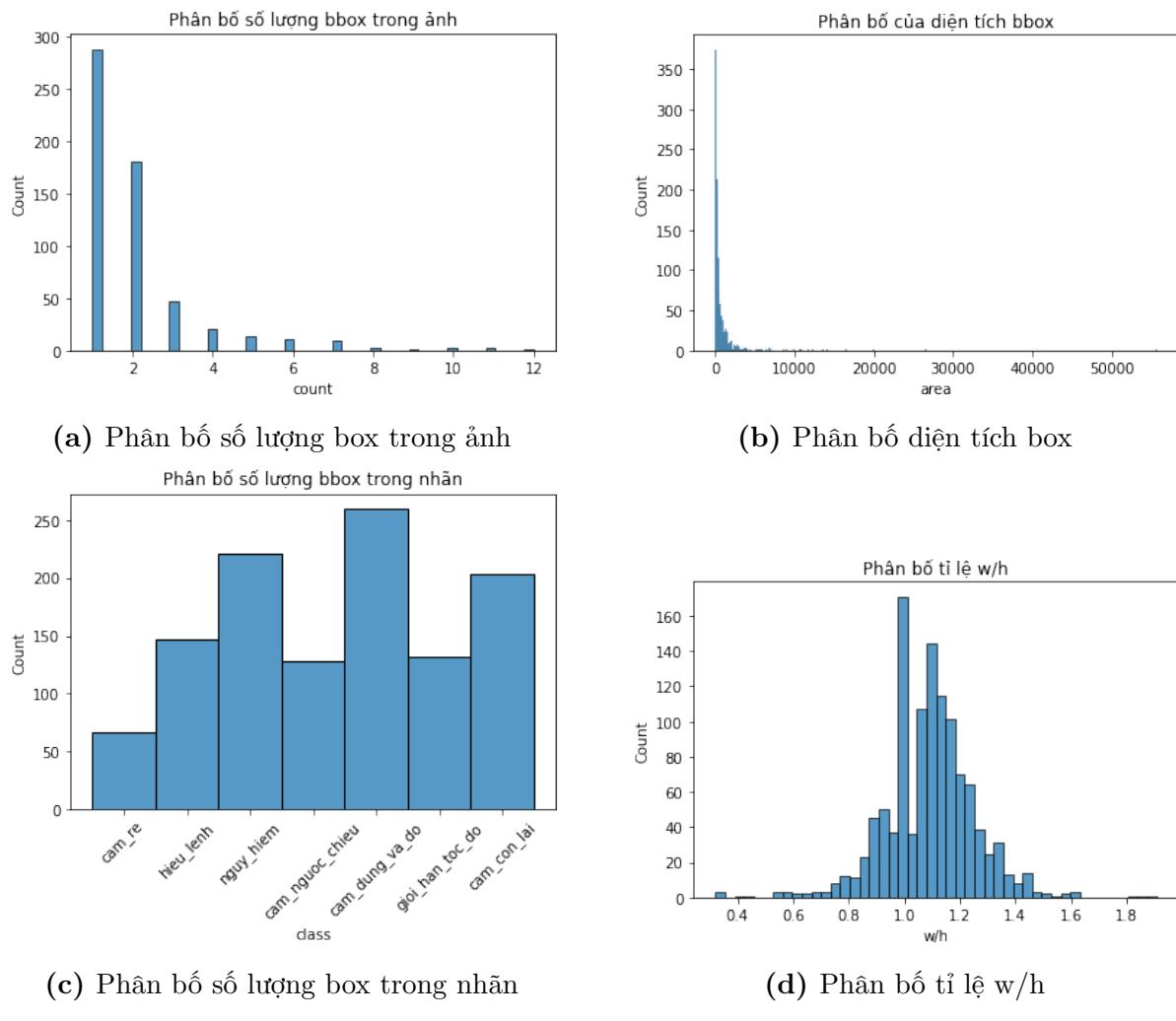
3.3 Bộ dữ liệu testing

Tập testing bao gồm 579 ảnh với 1159 groundtruth. Trung bình mỗi ảnh có 22 box. Hầu hết các ảnh có từ 1-2 box. Có 288 ảnh có 1 box, và 6 ảnh có 10 box trở lên. Ảnh có nhiều box nhất bao gồm 12 box (Hình 3.5a).

Box có diện tích lớn nhất là 55687 chiếm 5.48%. Box có diện tích nhỏ nhất là 42 chiếm 0.004%. Diện tích trung bình của box là 922.12 chiếm 0.09% (Hình 3.5b).

Lớp cấm dừng và đỗ là lớp có nhiều box nhất, gồm 260 box. Lớp cấm rẽ có ít box nhất, gồm 67 box. Trung bình mỗi lớp có 166 box (Hình 3.5c).

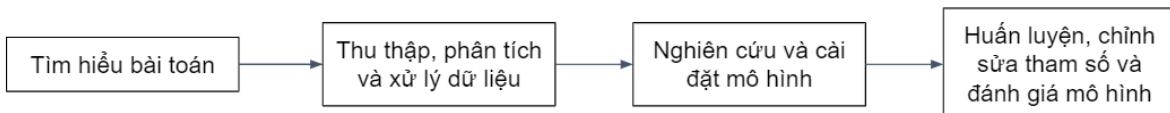
Trong tập testing, tỉ lệ w/h lớn nhất là 1.91, tỉ lệ nhỏ nhất 0.32. Trung bình, tỉ lệ w/h là 1.09 (Hình 3.5d).



Hình 3.5: Biểu đồ phân tích bộ dữ liệu testing

4 Hướng thực hiện

Quá trình thực hiện dự án có thể được chia thành 4 giai đoạn chính.



Hình 4.1: Quá trình thực hiện

4.1 Các mô hình sử dụng

4.1.1 EfficientDet-D1

Để huấn luyện mô hình này chúng tôi dùng pretrained model EfficientDet-D1 (COCO mAP = 38.4) của tensorflow object detection API với kích thước ảnh đầu vào là 640x640, Với ssd anchor generator, nhóm chọn 5 layer với min layer = 3, max layer = 7 và tỉ lệ của default box là [1 : 1, 2 : 1, 1 : 2]. Hàm weighted smooth L1 được dùng cho localization loss và weighted sigmoid focal cho classification loss (focal loss được dùng để cải thiện việc mất cân bằng dữ liệu của các đối tượng cần nhận dạng). Hàm kích hoạt là SWISH, với learning rate khởi tạo là 0.0004 kết hợp sử dụng chiến thuật giảm learning rate theo hàm mũ (cứ 2000 step sẽ giảm learning rate còn 80% so với learning rate ban đầu). Do bị hạn chế về mặt tài nguyên tính toán nên chúng tôi huấn luyện mô hình với 20000 step tương đương với 40 epoch, batch size: 8, sử dụng chuẩn L2 với chỉ số regularizer là: 0.0004. Thuật toán tối ưu được sử dụng để huấn luyện mô hình là Adam optimizer.

4.1.2 SSD MobileNet v2 320x320

Nhóm sử dụng pretrained model SSD MobileNet v2 320x320 (COCO mAP = 20.2) của tensorflow object detection API. Kích thước ảnh đầu vào là 300×300 . Với ssd anchor generator, nhóm chọn 6 layer với min layer = 2, max layer = 7 và tỉ lệ của default box là [1 : 1, 2 : 1, 1 : 2, 5 : 2, 1 : 3]. Hàm weighted smooth L1 được dùng cho localization loss và weighted sigmoid focal cho classification loss. Trong quá trình huấn luyện có thay đổi trọng số của hai loss để total loss được cân bằng giữa localization và classification loss.

Hàm kích hoạt là SIGMOID. Hàm tối ưu là Adam optimizer với learning rate giảm theo cấp số mũ, bắt đầu là 0.0004, giảm 0.2 mỗi 2000 bước. Nhóm huấn luyện với batch size là 32, 20000 bước, tương đương với 165 epoch.

4.1.3 SSD Resnet50 FPN V1 FPN 640x640 (RetinaNet50)

Nhóm huấn luyện mô hình dựa trên pretrained model SSD ResNet50 V1 FPN 640x640 (RetinaNet50) (COCO mAP = 34.3) của tensorflow object detection API. Kích thước ảnh đầu vào là 640×640 . Với anchor generator, min layer = 2, max layer = 7, anchor scale = 1 và tỉ lệ của anchor box là [1 : 1, 2 : 1, 1 : 2]. Hàm weighted smooth L1 được dùng cho localization loss và weighted sigmoid focal cho classification loss. Trong quá trình huấn luyện có thay đổi trọng số của hai loss để total loss được cân bằng giữa localization và classification loss. Hàm kích hoạt là SIGMOID. Hàm tối ưu là Adam optimizer với learning rate giảm theo cấp số mũ, bắt đầu là 0.0004, giảm 0.2 mỗi 2000 bước. Nhóm huấn luyện với batch size là 8, 70000 bước, tương đương với 145 epoch.

4.1.4 Yolo v5 - small, medium, large

Nhóm huấn luyện mô hình dựa trên pretrained mode Yolov5 của Ultralytics [8]. Kích thước ảnh đầu vào là 480×480 . Huấn luyện với batch size là 16, số epoch với 3 mô hình small, medium và large lần lượt là 400, 200 và 100. Các tham số còn lại được giữ nguyên theo Ultralytics. Khi test mô hình trên tập public của Zalo, kích thước ảnh là 2000×2000 và có sử dụng augmentation giúp tăng độ chính xác.

Model	size	AP ^{val}	AP ^{test}	AP ₅₀	Speed _{V100}	FPS _{V100}	params	GFLOPS
YOLOv5s	640	36.8	36.8	55.6	2.2ms	455	7.3M	17.0
YOLOv5m	640	44.5	44.5	63.1	2.9ms	345	21.4M	51.3
YOLOv5l	640	48.1	48.1	66.4	3.8ms	264	47.0M	115.4

Hình 4.2: So sánh các mô hình small, medium và large của yolov5

4.1.5 Faster R-CNN Resnet50 V1 640x640

Mô hình được huấn luyện dựa trên pretrained model Faster R-CNN ResNet50 V1 640x640 (COCO mAP = 29.3) của tensorflow object detection API. Kích thước ảnh đầu vào là

640×640 . Với grid anchor generator, height stride = width stride = 8, tỉ lệ anchor box là $[1 : 2, 1 : 1, 2 : 1]$ và kích thước box là $[0.25, 0.5, 1.0, 2.0]$. Trọng số của localization loss so với classification loss là 2:1. Hàm kích hoạt là SOFTMAX. Hàm tối ưu là Adam optimizer với learning rate giảm theo cấp số mũ, bắt đầu là 0.0004, giảm 0.2 mỗi 2000 bước. Nhóm huấn luyện với batch size là 8, 60000 bước, tương đương với 124 epoch.

4.2 Mean Average Precision

Để đánh giá độ chính xác của mô hình, nhóm sử dụng metric mean average precision (mAP) của MS-COCO [10] trên tập validating và tập testing.

mAP được tính là trung bình AP trên 10 khoảng IoU 0.5 đến 0.95 (mỗi khoảng cách nhau 0.05). AP được tính trung bình giữa các lớp.

5 Kết quả

Models	mAP validation (%)	mAP test (%)
YOLO v5 small	39.3	35.8
YOLO v5 medium	40.6	38
YOLO v5 large	40.5	37.5
SSD MobileNet	10.7	7.77
SSD ResNet50 FPN	46.7	20.8
EfficientDet D1	13.2	8.2
Faster R-CNN Resnet50	26.9	14.5

Bảng 5.1: Kết quả huấn luyện

Mô hình đạt kết quả mAP tốt nhất trên tập validating là SSD Resnet50 FPN (mAP = 46.7). Mô hình đạt kết quả mAP tốt nhất trên tập public testing là YOLO v5 medium (mAP = 38).

Nhóm chọn ra 3 ảnh trong tập public test và dự đoán trên 3 ảnh sử dụng 5 model (chọn model YOLO có kết quả tốt nhất). Dưới đây là kết quả dự đoán.



Hình 5.1: Ảnh nguyên gốc



Hình 5.2: Kết quả dự đoán của YOLO v5



Hình 5.3: Kết quả dự đoán của SSD MobileNet



Hình 5.4: Kết quả dự đoán của SSD ResNet50 FPN



Hình 5.5: Kết quả dự đoán của EfficientDet D1



Hình 5.6: Kết quả dự đoán của Faster R-CNN ResNet50

So với kết quả public test của ứng viên tham gia cuộc thi Zalo AI Challenge, nhóm nằm trong top 60.

6 Kết luận

6.1 Thảo luận

Trong quá trình giải quyết bài toán, nhóm đã sử dụng thử nghiệm nhiều mô hình học sâu khác nhau để kiểm tra độ chính xác của các phương pháp để giải quyết bài toán một cách tối ưu. Trong quá trình đó, nhóm gặp rất nhiều hạn chế khiến cho việc giải quyết bài toán chưa thật sự tối ưu. Những hạn chế đó đến từ việc thiếu tài nguyên để thực hiện bài toán cũng như tối ưu các mô hình.Thêm vào đó, việc giải quyết bài toán được thực hiện trong thời gian ngắn dẫn đến kết quả đạt được cho bài toán này chưa thật sự cao nhưng vẫn có thể chấp nhận được. Với sự phát triển của học sâu ngày nay thì bài toán hoàn toàn có thể được giải quyết với các phương pháp và mô hình phức tạp và mang lại kết quả tốt hơn. Việc không đủ tài nguyên để thử nghiệm với các mô hình phức tạp hơn cũng là nguyên nhân chính là nguyên nhân làm cho việc giải quyết bài toán của nhóm chưa đạt độ chính xác cao.

6.2 Hướng phát triển

- Điều chỉnh tham số, áp dụng các mô hình khác để giải quyết bài toán.
- Nghiên cứu thêm các bộ dữ liệu khác mang tính thực tế hơn để triển khai mô hình nhận dạng biển báo thời gian thực.
- Nghiên cứu phương pháp ensemble learning với các mô hình nhằm đạt độ chính xác cao hơn
- Nghiên cứu thêm cách triển khai mô hình lên các thiết bị nhúng để có thể triển khai ứng dụng vào thực tế.

6.3 Hạn chế của dự án

Do hạn chế về tài nguyên và thời gian nên chỉ chọn được những mô hình với kích thước ảnh nhỏ. Tuy đã chọn ra được mô hình tốt nhất nhưng vẫn còn một số trường hợp mô hình vẫn cho kết quả không tốt. Khi thực hiện trên tập dữ liệu Traffic Sign Detection của cuộc thi Zalo AI Challenge thì kết quả không tốt bằng các đội khác đã tham gia giải quyết bài toán này từ trước. Nhóm có độ đo mAP trên tập Public test là 38 , trong khi nhiều đội khác giải quyết bài toán này với kết quả rất cao với độ đo mAP trong khoảng từ 0.49 đến 0.52.

7 Phân công công việc

7.1 Đặng Chí Trung - nhóm trưởng

1. Thu thập và tiền xử lý dữ liệu
2. Nghiên cứu các phương pháp giải quyết bài toán
3. Xây dựng, huấn luyện và đánh giá mô hình (YOLO)
4. Thuyết trình và viết báo cáo

7.2 Lâm Văn Sang Em

1. Tiền xử lý dữ liệu
2. Nghiên cứu các phương pháp giải quyết bài toán
3. Xây dựng, huấn luyện và đánh giá mô hình (EfficientDet)
4. Thuyết trình và viết báo cáo

7.3 Ngô Phương Nhi

1. Tiền xử lý và phân tích dữ liệu
2. Nghiên cứu các phương pháp giải quyết bài toán
3. Xây dựng, huấn luyện và đánh giá mô hình (SSD, Faster R-CNN)
4. Thuyết trình và viết báo cáo

8 Tài liệu tham khảo

- [1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. (2016). You Only Look Once: Unified, Real-Time Object Detection.
- [2] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A. (2016). SSD: Single Shot MultiBox DetectorLecture Notes in Computer Science, 21–37.
- [3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
- [4] Mingxing Tan, Ruoming Pang, Quoc V. Le. (2020). EfficientDet: Scalable and Efficient Object Detection.
- [5] Arcos-García, Álvaro Alvarez-Garcia, Juan Soria Morillo, Luis. (2018). Evaluation of Deep Neural Networks for traffic sign detection systems. Neurocomputing. 316. 10.1016/j.neucom.2018.08.009.
- [6] Pavly Salah Zaki, Marco Magdy William, Bolis Karam Soliman, Keroles Gamal Alexsan, Keroles Khalil, Magdy El-Moursy. (2020). Traffic Signs Detection and Recognition System using Deep Learning.
- [7] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li and S. Hu, "Traffic-Sign Detection and Classification in the Wild," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2110-2118, doi: 10.1109/CVPR.2016.232.
- [8] Glenn Jocher <https://github.com/ultralytics/yolov5/tree/v4.0>, doi: <https://doi.org/10.5281/zenodo.4418161>
- [9] Zalo AI Challenge - Traffic Sign Detection. <https://challenge.zalo.ai/portal/traffic-sign-detection>
- [10] MS-COCO Evaluation. <https://cocodataset.org/detection-eval>
- [11] Hoanh Nguyen, "Fast Traffic Sign Detection Approach Based on Lightweight Network and Multilayer Proposal Network", Journal of Sensors, vol. 2020, Article ID 8844348, 13 pages, 2020. <https://doi.org/10.1155/2020/8844348>
- [12] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic

sign detection, tracking, and recognition using color, shape, and motion information,” in IEEE Proceedings. Intelligent Vehicles Symposium, 2005, pp. 255–260, Las Vegas, NV, USA, 2005.

[13] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, “An automatic traffic sign detection and recognition system based on colour segmentation, shape matching, and SVM,” Mathematical Problems in Engineering, vol. 2015, Article ID 250461, 11 pages, 2015.

[14] Z. Liu, J. Du, F. Tian, and J. Wen, “MR-CNN: a multi-scale region-based convolutional neural network for small traffic sign recognition,” IEEE Access, vol. 7, pp. 57120–57128, 2019.

[15] T. Yang, X. Long, A. K. Sangaiah, Z. Zheng, and C. Tong, “Deep detection network for real-life traffic sign in vehicular networks,” Computer Networks, vol. 136, pp. 95–104, 2018.