

**ĐẠI HỌC QUY NHƠN
KHOA CÔNG NGHỆ THÔNG TIN**

-----o0o-----



BÁO CÁO CUỐI KỲ

MÔN XỬ LÝ ẢNH

ĐỀ TÀI:

**NGHIÊN CỨU KỸ THUẬT PHÁT HIỆN, PHÂN LOẠI
VÀ ĐẾM XE DƯỚI SỰ HỖ TRỢ CAMERA**

Sinh viên thực hiện: Lê Thị Minh Tâm
 Nguyễn Thị Triều
 Huỳnh Tín Trọng

Quy Nhon, ngày 10, tháng 6 năm 2021

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT	3
DANH MỤC CÁC HÌNH.....	4
I. Mở đầu	5
1. Lý do chọn đề tài.....	5
2. Mục tiêu và nhiệm vụ	5
3. Ý nghĩa khoa học và thực tiễn	5
II. Thuật toán	6
1. DNN (Deep Neural Network)	6
2. SSD(Single Shot Detector)	7
III. Thực hiện.....	18
1. Code demo	18
2. Kết quả.....	20
IV. Đánh giá.....	20
1. Kết quả đạt được	20
2. Hạn chế	20
3. Hướng phát triển	20
V. Tài liệu tham khảo	21

DANH MỤC TỪ VIẾT TẮT

DNN	Deep Neural Network
SSD	Single Shot Detector
IoU	Intersection of Union

DANH MỤC CÁC HÌNH

Hình 1: Cấu trúc DNN	6
Hình 2: Cách thức phân chia feature map để nhận diện các hình ảnh với những kích thước khác nhau	8
Hình 3: Sơ đồ kiến trúc của mạng SSD	9
Hình 4: Vị trí của các default bounding box trên bức ảnh gốc khi áp dụng trên feature map có kích thước 4 x 4.....	12
Hình 5: Kết quả phát hiện xe.	20

I. Mở đầu

1. Lý do chọn đề tài

Giao thông luôn là vấn đề được quan tâm nhiều nhất đối với các đô thị lớn ở các nước nói chung và ở Việt Nam nói riêng. Trong những năm gần đây tình trạng này vẫn luôn là vấn đề nghiêm trọng và bức thiết. Đây là vấn đề bức xúc của toàn xã hội, ảnh hưởng đến sự phát triển về kinh tế, văn hóa, xã hội và hình ảnh của đất nước Việt Nam với bạn bè quốc tế. Hiện nay, mặc dù có những nghiên cứu, nhiều giải pháp gỡ bỏ tình trạng tắc nghẽn giao thông nhưng hiệu quả không cao và tình trạng kẹt xe vẫn thường xuyên trên các đường giao thông, đặc biệt là ở các khu đô thị. Xuất phát từ yêu cầu thực hiện nhóm em đã chọn đề tài “Nghiên cứu kỹ thuật phát hiện phân loại và đếm xe dưới sự hỗ trợ của camera”.

2. Mục tiêu và nhiệm vụ

➤ Mục tiêu:

Từ thực trạng trên đòi hỏi mục tiêu đặt ra trước mắt là :

- Giảm thiểu ùn tắc , tai nạn giao thông.
- Nâng cao nhận thức và ý thức tự giác chấp hành giao thông.
- Nghiên cứu và nhân rộng các giải pháp đột phá về khắc phục tình trạng ùn tắc giao thông , đồng thời và nâng cao hiệu quả quản lý trật tự đô thị.

➤ Nhiệm vụ:

Từ mục tiêu của đề tài , đề tài tập trung nghiên cứu các nhiệm vụ sau:

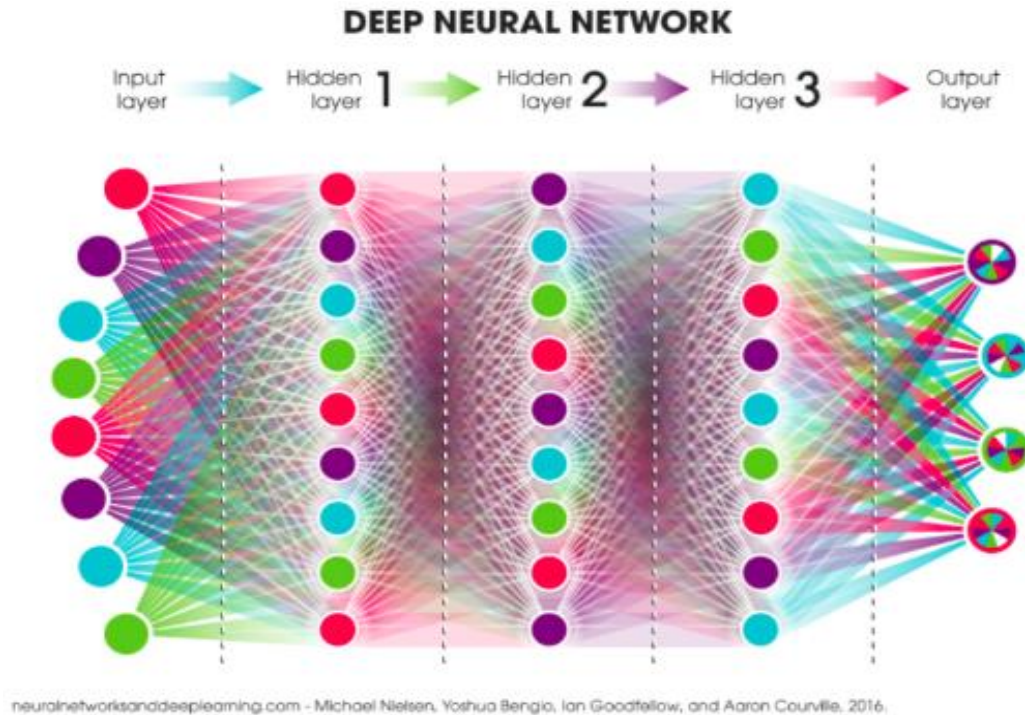
- Tìm hiểu kỹ thuật Deep Learning.
- Tìm hiểu các bộ thư viện cài đặt cho mô hình Deep Learning.
- Sử dụng công cụ nhận diện để đánh nhãn xe trong video giao thông.
- Huấn luyện và xây dựng mô hình nhận dạng , đếm xe.

3. Ý nghĩa khoa học và thực tiễn

- Về khoa học : Áp dụng phương pháp Deep Learning để phát hiện và đếm xe.
- Về thực tiễn : Hỗ trợ xác định lưu lượng xe đang tham gia giao thông trên các trục đường giao thông , giúp giải quyết các vấn đề về giao thông .

II. Thuật toán

1. DNN (Deep Neural Network)



Hình 1: Cấu trúc DNN

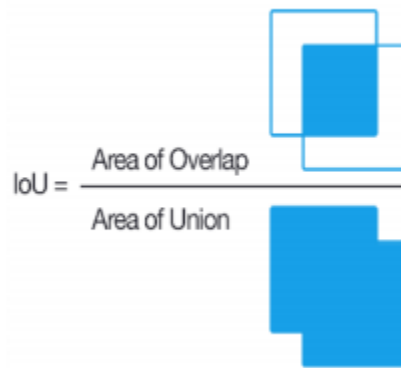
Deep neural network là hệ thống cấu trúc thần kinh phức tạp gồm nhiều đơn vị neural network mà trong đó, ngoài các lớp nguồn vào (input), nguồn ra (output) thì có hơn một lớp ẩn (hidden layer). Mỗi lớp này sẽ thực hiện một kiểu phân loại và sắp xếp riêng trong một quá trình ta gọi là “phân cấp tính năng” và mỗi lớp đảm nhiệm một trọng trách riêng, output của lớp này sẽ là input của lớp sau.

Deep Neural Network được xây dựng với mục đích mô phỏng hoạt động não bộ phức tạp của con người và được áp dụng vào nhiều lĩnh vực khác nhau, mang lại thành công và những hiệu quả đáng kinh ngạc cho con người.

2. SSD(Single Shot Detector)

Một số định nghĩa:

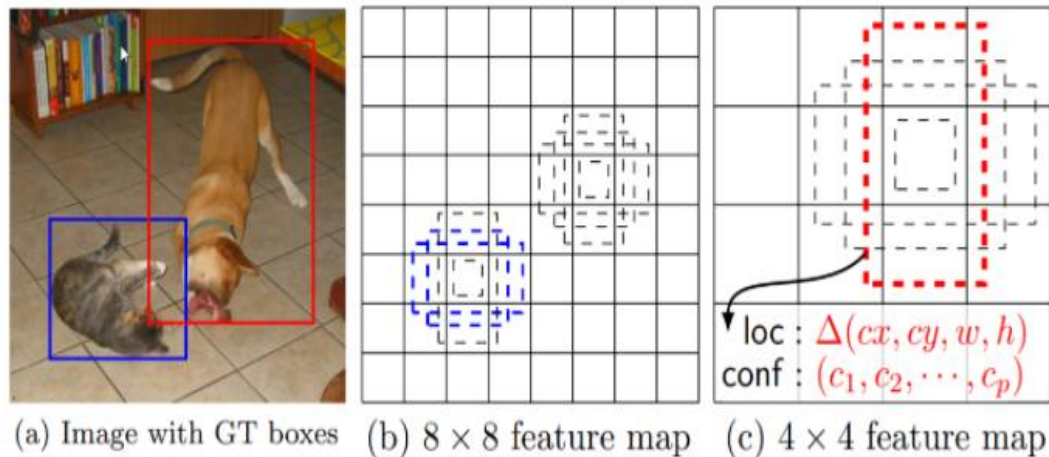
- **scale:** Tỷ lệ chiều dài và chiều rộng so với khung hình gốc. VD: Nếu khung hình gốc có giá trị là (w, h) thì sau scale khung hình mới có kích thước là (sw, sh) . Giá trị của s thường nằm trong khoảng $s \in (0,1]$. Scale sẽ kết hợp với aspect ratio để nhận được các khung hình có tỷ lệ cạnh w/h khác nhau.
- **aspect ratio:** Tỷ lệ cạnh, được đo bằng tỷ lệ giữa w/h nhằm xác định hình dạng tương đối của khung hình bao chứa vật thể. Chẳng hạn nếu vật thể là người thường có aspect ratio = 1:3 hoặc xe cộ nhìn từ phía trước là 1:1.
- **bounding box:** Khung hình bao chứa vật thể được xác định trong quá trình huấn luyện.
- **ground truth box:** Khung hình được xác định trước từ bộ dữ liệu thông qua tọa độ (c_x, c_y, w, h) giúp xác định vật thể.
- **offsets:** Các tọa độ (c_x, c_y, w, h) để xác định vật thể.
- **IoU:** **IoU:** Tỷ lệ Intersection of Union là tỷ lệ đo lường mức độ giao nhau giữa 2 khung hình (thường là khung hình dự báo và khung hình ground truth) để nhằm xác định 2 khung hình overlap không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa 2 khung hình với phần tổng diện tích giao nhau và không giao nhau giữa chúng.


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

- **positive matching prediction:** Khung được dự báo (predicted box) là vùng có vật thể là đúng, được xác định dựa trên tỷ lệ $\text{IoU} > 0.5$ giữa predicted box với ground truth box.

- **negative matching prediction:** Khung được dự báo (predicted box) là vùng không chứa vật thể là đúng, cũng được xác định dựa trên $\text{IoU} < 0.5$ giữa predicted box với ground truth box.

Single Shot Detector là gì?

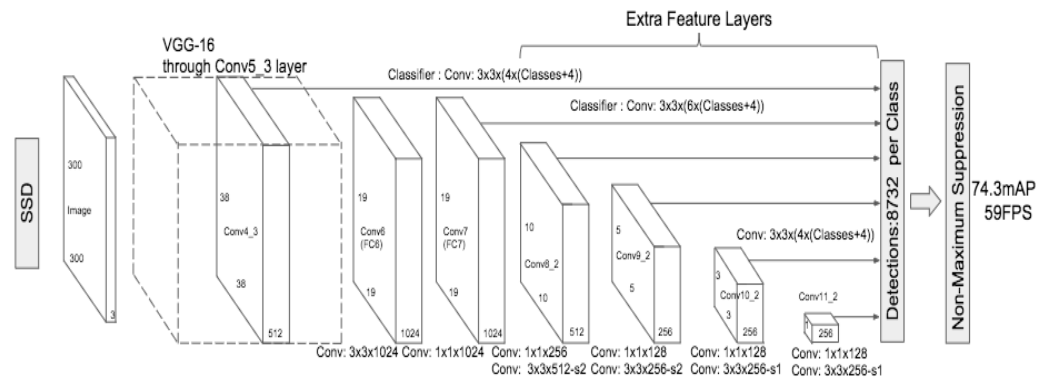


Hình 2: Cách thức phân chia feature map để nhận diện các hình ảnh với những kích thước khác nhau

SSD chỉ cần duy nhất đầu vào là 1 bức ảnh và các ground truth boxes ám chỉ vị trí bounding box các vật thể trong suốt quá trình huấn luyện. Trong quá trình phát hiện vật thể, trên mỗi một feature map, chúng ta đánh giá các một tập hợp nhỏ gồm những default boxes tương ứng với các tỷ lệ cạnh khác nhau (aspect ratio) lên các features map có kích thước (scales) khác nhau (chẳng hạn kích thước 8×8 và 4×4 trong hình (b) và (c)). Đối với mỗi default box (các boxes nét đứt trong hình) ta cần dự báo một phân phối xác suất $\mathbf{c} = (c_1, c_2, \dots, c_n)$ tương ứng với các class $\mathbf{C} = (C_1, C_2, \dots, C_n)$. Tại thời điểm huấn luyện, đầu tiên chúng ta cần match default boxes với ground truth boxes sao cho mức độ sai số được đo lường qua localization loss là nhỏ nhất (thường là hàm Smooth L1). Sau đó ta sẽ tìm cách tối thiểu hóa sai số của nhãn dự báo tương ứng với mỗi vật thể được phát hiện trong default boxes thông qua confidence loss (thường là hàm softmax).

Như vậy loss function của object detection sẽ khác với loss function của các tác vụ image classification ở chỗ có thêm localization loss về sai số vị trí của predicted boxes so với ground truth boxes.

Kiến trúc của mô hình



Hình 3: Sơ đồ kiến trúc của mạng SSD

SSD dựa trên một tiến trình lan truyền thuận của một kiến trúc chuẩn (chẳng hạn VGG16) để tạo ra một khối feature map output gồm 3 chiều ở giai đoạn sớm. Gọi kiến trúc mạng này là base network (tính từ input Image đến Conv7). Sau đó sẽ thêm những kiến trúc phía sau base network để tiến hành nhận diện vật thể như phần Extra Feature Layers trong sơ đồ.

Các layer của mô hình SSD:

- **Input Layer:** Nhận input đầu vào là các bức ảnh có kích thước (width x height x channels) = 300 x 300 x 3 đối với kiến trúc SSD300 hoặc 500 x 500 x 3 đối với kiến trúc SSD500.
- **Conv5_3 Layer:** Chính là base network sử dụng kiến trúc của VGG16 nhưng loại bỏ một số layers fully connected ở cuối cùng. Output của layer này chính là Conv4_3 Layer và là một feature map có kích thước 38 x 38 x 512.
- **Conv4_3 Layer:** Ta có thể coi Conv4_3 là một feature map có kích thước 38 x 38 x 512. Trên feature map này ta sẽ áp dụng 2 biến đổi chính đó là:

Áp dụng một convolutional layer như một mạng CNN thông thường để thu được output layer tiếp theo. Cụ thể convolutional layer có convolutional kernel kích thước 3 x 3 x 1024, đầu ra thu được Conv6 có kích thước là 19 x 19 x 1024.

Đồng thời ở bước này ta cũng áp dụng một classifier (như trong sơ đồ) và cũng dựa trên convolutional filter kích thước 3×3 để nhằm nhận diện vật thể trên feature map. Đây là một quá trình khá phức tạp vì nó phải đảm bảo phát hiện vật thể (thông qua phát hiện bounding box) và phân loại vật thể. Quá trình này thực hiện tương tự như mô tả ở hình 2. Đầu tiên ta sẽ phân chia feature map kích thước $38 \times 38 \times 512$ thành một grid cell kích thước 38×38 (bỏ qua độ sâu vì ta sẽ thực hiện tích chập trên toàn bộ độ sâu). Sau đó mỗi một cell trên grid cell sẽ tạo ra 4 default bounding boxes với các tỷ lệ khung hình khác nhau (aspect ratio), mỗi một default bounding box ta cần tìm các tham số sau: phân phối xác suất của nhãn là một véc tơ có $n_classes + 1$ chiều (Lưu ý số lượng classes luôn cộng thêm 1 cho background). Đồng thời chúng ta cần thêm 4 tham số là offsets để xác định bounding box của vật thể trong khung hình. Do đó trên một default bounding box sẽ có $n_classes + 4$ tham số và trên 1 cell sẽ có $4 \times (n_classes + 4)$ output cần dự báo. Nhân với số cells của Conv4_3 để thu được số lượng output là một tensor kích thước $38 \times 38 \times 4 \times (n_classes + 5)$, trong trường hợp coi background cũng là 1 nhãn thì tensor có kích thước $38 \times 38 \times 4 \times (n_classes + 4)$. Và số lượng các bounding box được sản sinh ra là $38 \times 38 \times 4$.

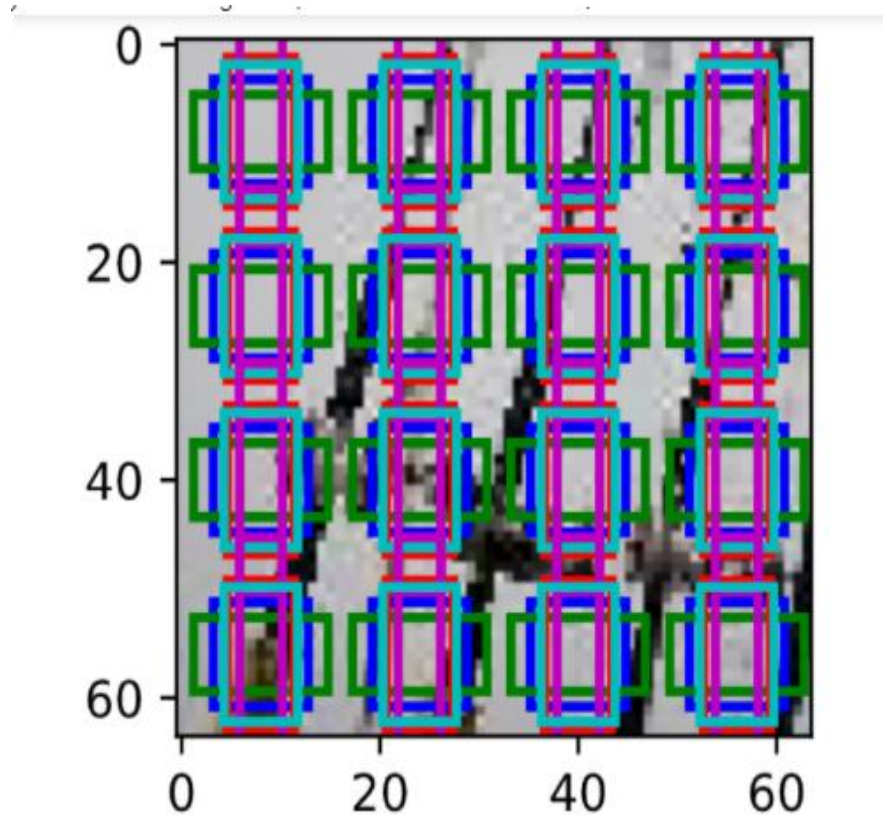
Quá trình áp dụng classifier lên feature map cũng tương tự với các layer Conv7, Conv8_2, Conv9, Conv10_2, Conv11_2. Shape của các layer sau sẽ phụ thuộc vào cách thức áp dụng tích chập (convolutional process) ở layer trước, kích thước kernel filter (như trong sơ đồ trên thì kernel_size luôn là 3×3) và stride (độ lớn bước nhảy) của tích chập. Trên mỗi cell thuộc feature map ta xác định một lượng 4 hoặc 6 các default bounding boxes. Do đó, số lượng các default boxes sản sinh ra ở các layers tiếp theo lần lượt như sau:

- **Conv7:** $19 \times 19 \times 6 = 2166$ boxes (6 boxes/cell)
- **Conv8_2:** $10 \times 10 \times 6 = 600$ boxes (6 boxes/cell)
- **Conv9_2:** $5 \times 5 \times 6 = 150$ boxes (6 boxes/cell)
- **Conv10_2:** $3 \times 3 \times 4 = 36$ boxes (4 boxes/cell)
- **Conv11_2:** $1 \times 1 \times 4 = 4$ boxes (4 boxes/cell) Tổng số lượng các boxes ở output sẽ là: $5766 + 2166 + 600 + 150$

+ 36 + 4 = 8732. Tức là chúng ta cần phải dự đoán class cho khoảng 8732 khung hình ở output. Số lượng này lớn hơn rất nhiều so với YOLO khi chỉ phải dự đoán chỉ 98 khung hình ở output. Đó là lý do tại sao thuật toán có tốc độ chậm hơn so với YOLO. Các véc tơ output tương ứng cho mỗi default bounding box dạng:

$$y^T = [\underbrace{x, y, w, h}_{\text{bounding box}}, \underbrace{c_1, c_2, \dots, c_C}_{\text{scores of C classes}}]$$

- **Áp dụng các feature map với các kích thước khác nhau:** Sau khi thu được feature map ở base network. Chúng ta sẽ tiếp tục thêm các convolutional layers. Những layers này sẽ nhằm giảm kích thước của feature map từ đó giảm số lượng khung hình cần dự báo và 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. Cụ thể hơn về kích thước kernel filters sẽ xem ở sơ đồ kiến trúc trong phần Extra Features Layers.
- **Dự báo thông qua mạng tích chập đối với object:** Mỗi một feature layer thêm vào ở Extra Features Layers sẽ tạo ra một tập hợp cố định các output y giúp nhận diện vật thể trong ảnh thông qua áp dụng các convolutional filters. Kích thước ở đầu ra (with x height x chanel) ở mỗi loại kích thước feature layer sẽ phụ thuộc vào kernel filters và được tính toán hoàn toàn tương tự như đối với mạng neural tích chập thông thường.
- **Default box và tỷ lệ cạnh (aspect ratio):** Cần liên kết một tập hợp default bounding boxes với mỗi một cell trên feature map. Các default boxes sẽ phân bố lát gạch trên feature map theo thứ tự từ trên xuống dưới và từ trái qua phải để tính tích chập, do đó vị trí của mỗi default box tương ứng với cell mà nó liên kết là cố định tương ứng với một vùng ảnh trên bức ảnh gốc. Cụ thể như hình ảnh minh họa bên dưới:



Hình 4: Vị trí của các default bounding box trên bức ảnh gốc khi áp dụng trên feature map có kích thước 4 x 4.

Như vậy grid cells sẽ có kích thước là 4 x 4 và trên mỗi cell ta sẽ xác định 4 defaults bounding boxes khác nhau như hình vẽ. Tâm của các bounding boxes này là trùng nhau và chính là tọa độ tâm của các cell mà nó liên kết.

Tại mỗi một default bounding box của feature map chúng ta dự báo 4 offsets tương ứng với một tọa độ và kích thước của nó. 4 offsets ở đây được hiểu là một tọa độ gồm 4 tham số (c_x , c_y , w , h). Trong đó (c_x , c_y) giúp xác định tâm và (w , h) là kích thước dài rộng của bounding box. Thành phần thứ 2 được dự báo là điểm số của bounding box tương ứng với mỗi class. Lưu ý ta sẽ có thêm một class thứ $C+1$ để đánh dấu trường hợp default bounding box không có vật thể (hoặc rơi vào background).

Ví dụ đối với một feature map có kích thước $m \times n$ tương ứng với p channels (chẳng hạn như kích thước 8×8 hoặc 4×4), một kernel filter kích thước $3 \times 3 \times p$ sẽ được áp dụng trên toàn bộ feature layer.

Các giá trị trong kernel này chính là các tham số của mô hình và được tinh chỉnh trong quá trình training.

Các kernel filter sẽ dự đoán đồng thời Xác suất nhãn và kích thước offset tương ứng với tọa độ của default box.

Với mỗi location (hoặc cell) nằm trên feature map ta sẽ liên kết nó với bounding boxes. Các boxes này có kích thước khác nhau và tỷ lệ cạnh khác nhau.

Với mỗi một bounding box, chúng ta tính được phân phối điểm của C classes là $c = (c_1, c_2, \dots, c_n)$ và 4 offsets tương ứng với kích thước ban đầu của default bounding box.

Kết quả cuối cùng ta thu được $(C+4) * mnk$ outputs.

Các default box của chúng ta tương tự như anchor boxes trong mạng faster R-CNN nhưng được áp dụng trên một vài feature maps với những độ phân giải khác nhau. Điều này cho phép các default bounding box phân biệt hiệu quả kích thước vật thể khác nhau.

Quá trình huấn luyện

Chiến lược mapping default box: Trong suốt quá trình huấn luyện ta cần mapping các defaultboxes có tỷ lệ aspect ratio khác nhau với ground truth box. Để mapping được chúng với nhau ta cần đo lường chỉ số IoU (Intersection of Union) hoặc chỉ số Jaccard overlap index được dùng để đo lường tỷ lệ diện tích giao nhau giữa 2 vùng hình ảnh so với tổng diện tích (không tính phần giao nhau) của chúng. Chúng ta sẽ match các default boxes với bất kì ground truth nào có threshold > 0.5 .

Mỗi cell chỉ quy định một số lượng nhất định (4 hoặc 6, tùy từng feature map) các default bounding box.

Huấn luyện để tìm ra object: Việc dự báo các object sẽ được thực hiện trên tập hợp các khung hình output của mạng SSD. Đặt $x_{ij}^k = 0, 1$ là chỉ số đánh giá cho việc matching giữa default bounding box

thứ i với ground truth box thứ j đối với nhãn thứ k . Trong quá trình mapping chúng ta có thể có nhiều bounding box được map vào cùng 1 ground truth box với cùng 1 nhãn dự báo nên tổng $\sum_i x_{ij}^k \geq 1$. Hàm loss function là tổng có trọng số của localization loss (loc) và confidence loss (conf):

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

Trong đó N là số lượng các default boxes matching với ground truth boxes. Ta nhận thấy giá trị của hàm loss function của SSD bao gồm 2 thành phần:

- **localization loss:** là một hàm Smooth L1 đo lường sai số giữa tham số của box dự báo (predicted box) (p) và ground truth box (g) như bên dưới: Chúng ta sẽ cần hồi qui các offsets cho tâm (x, y) và của default bounding box (d) và các chiều dài h và chiều rộng w .

$$L_{loc}(x, p, g) = \sum_{i \in \text{Pos}} \sum_{m \in \{x, y, w, h\}} x_{ij}^k L_1^{\text{smooth}}(P_i^m - g_i^m)$$

Như vậy Localization loss chỉ xét trên các positive matching ($i \in \text{Pos}$) giữa predicted bounding box với ground truth bounding box. Nếu $\text{IoU} > 0.5$ thì được coi là positive matching (tức predicted bounding box chứa vật thể). Trái lại, nếu $\text{IoU} \leq 0.5$ ta không cần quan tâm và coi như xóa các predicted bounding box này khỏi hình ảnh.

Thành phần $\sum_{m \in \{x, y, w, h\}} x_{ij}^k L_1^{\text{smooth}}(P_i^m - g_i^m)$ chính là tổng khoảng cách giữa predicted box (p) và ground truth box (g) trên 4 offsets (x, y, w, h).

Nếu để nguyên các giá trị tọa độ tâm và kích thước của khung hình sẽ rất khó để xác định sai số một cách chuẩn xác. Ta hãy so sánh sai số trong trường hợp khung hình lớn và khung hình bé. Trong trường hợp khung hình lớn có predicted box và ground truth box rất khớp nhau. Tuy nhiên do khung hình quá to nên khoảng cách tâm của chúng sẽ lớn một chút, giả định là aspect ratio của chúng bằng nhau. Còn trường hợp khung hình bé, sai số của tâm giữa predicted box và ground truth box có thể bé hơn

trường hợp khung hình lớn về số tuyệt đối. Nhưng điều đó không có nghĩa rằng predicted box và ground truth box của khung hình bé là rất khớp nhau. Chúng có thể cách nhau rất xa. Do đó chúng ta cần phải chuẩn hóa kích thước width, height và tâm sao cho không có sự khác biệt trong trường hợp khung hình bé và lớn.

- **confidence loss:** là một hàm mất mát được tính toán dựa trên sai số dự báo nhãn. Đối với mỗi một positive match prediction, chúng ta phạt loss function theo confidence score của các nhãn tương ứng. Đối với mỗi một negative match prediction, chúng ta phạt loss function theo confidence score của nhãn '0' là nhãn đại diện cho background không chứa vật thể. Cụ thể hàm confidence loss như bên dưới:

$$L_{conf}(x, c) = - \sum_{i \in \text{Pos}} x_{ij}^k \log(c_i^0)$$

Trong trường hợp positive match prediction thì vùng được dự báo có vật thể chính xác là chứa vật thể. Do đó việc dự báo nhãn cho nó sẽ tương tự như một bài toán classification với hàm softmax thông thường có dạng:

$$- \sum_{i \in \text{Pos}} x_{ij}^k \log(c_i^0)$$

Hàm loss function cuối cùng được tính là tổng của 2 confidence loss và localization loss.

Lựa chọn kích cỡ (scales) và tỷ lệ cạnh (aspect ratio):

Các default boundary box được lựa chọn thông qua aspect ratio và scales. SSD sẽ xác định một tỷ lệ scale tương ứng với mỗi một features map trong Extra Feature Layers. Bắt đầu từ bên trái, Conv4_3 phát hiện các object tại các scale nhỏ nhất là $s_{min} = 0.2$ (đôi khi là 0.1) và sau đó gia tăng tuyến tính để layer cuối cùng ở phía bên phải có scale là $s_{max} = 0.9$ theo công thức:

$$sk = s_{min} + \frac{s_{max} - s_{min}}{m - 1} (k - 1), k \in [1, m]$$

Với k là số thứ tự của layers. Kết hợp giữa giá trị scale với aspect ratio chúng ta sẽ tính được width và height của default boxes. Với các

layers có 6 dự báo, SSD sẽ tạo ra 5 default boxes với các aspect ratios lần lượt là: 1, 2, 3, 1/2, 1/3. Sau đó width và height của default boxes được tính theo công thức:

$$w = scale * \sqrt{aspect\ ratio}$$

$$h = \frac{scale}{\sqrt{aspect\ ratio}}$$

Trong trường hợp $\sqrt{aspect\ ratio} = 1$ thì ta sẽ thêm một default bounding box thứ 6 với scale được tính theo công thức:

$$s_k' = \sqrt{s_k s_{k+1}}$$

Anchor Box

Phần tinh túy nhất của SSD có lẽ là việc xác định các layers output của anchor box (hoặc default bounding box) ở các feature map. anchor box layer sẽ nhận đầu vào ra một feature map có kích thước (feature_width, feature_height, n_channels) và các scales, aspect ratios, trả ra đầu ra là một tensor kích thước (feature_width, feature_height, n_boxes, 4).

- **Bước 1:** Từ scale, size (giá trị lớn nhất của width và height), và aspect ratio ta xác định kích thước các cạnh của các bounding box theo công thức:

$$\begin{cases} box_h = scale * size / \sqrt{aspect\ ratio} \\ box_w = scale * size * \sqrt{aspect\ ratio} \end{cases}$$

- **Bước 2:** Từ các cell trên feature map chiếu lại trên ảnh input image để thu được step khoảng cách giữa các center point của mỗi cell theo công thức:

$$\begin{cases} step_h = img_h / feature_map_h \\ step_w = img_w / feature_map_w \end{cases}$$

- **Bước 3:** Tính tọa độ các điểm (c_x, c_y, w, h) trên hình ảnh gốc dựa trên phép linear interpolation qua hàm np.Linspace ().


```
[ cx = np. Linspace (start_w, end_w, feature_map_w)
  cy = np. Linspace (start_h, end_h, feature_map_h)
```

Kết quả trả về là một tensor có shape là (feature_width, feature_height, n_boxes, 8), trong đó chiều cuối cùng = 8 tương ứng với 4 offsets của default bounding box và 4 variances đại diện cho các scales của default bounding box.

III. Thực hiện

1. Code demo

```
1  #Import the necessary libraries
2  import numpy as np
3  import argparse
4  import cv2
5
6  # construct the argument parse
7  parser = argparse.ArgumentParser(
8      description='Script to run MobileNet-SSD object detection network ')
9  parser.add_argument("--video", help="path to video file. If empty, camera's stream will be used")
10  parser.add_argument("--prototxt", default="MobileNetSSD_deploy.prototxt",
11                      help='Path to text network file: '
12                          'MobileNetSSD_deploy.prototxt for Caffe model or '
13                          ')
14  parser.add_argument("--weights", default="MobileNetSSD_deploy.caffemodel",
15                      help='Path to weights: '
16                          'MobileNetSSD_deploy.caffemodel for Caffe model or '
17                          ')
18  parser.add_argument("--thr", default=0.2, type=float, help="confidence threshold to filter out weak detections")
19  args = parser.parse_args()
20  font = cv2.FONT_HERSHEY_SIMPLEX
21  count = 0
22
23  # Labels of Network.
24  classNames = {_0: 'background',
25                1: 'bicycle', 6: 'bus', 7: 'car',
26                14: 'motorbike'}
27
28  # Open video file or capture device.
29  if args.video:
30      cap = cv2.VideoCapture(args.video)
31  else:
32      cap = cv2.VideoCapture(0)
33
34  #Load the Caffe model
35  net = cv2.dnn.readNetFromCaffe(args.prototxt, args.weights)
36
37  while True:
38      # Capture frame-by-frame
39      ret, frame = cap.read()
40      frame_resized = cv2.resize(frame, (300, 300)) # resize frame for prediction
41
42      # MobileNet requires fixed dimensions for input image(s)
43      # so we have to ensure that it is resized to 300x300 pixels.
44      # set a scale factor to image because network the objects has different size.
45      # We perform a mean subtraction (127.5, 127.5, 127.5) to normalize the input;
46      # after executing this command our "blob" now has the shape:
47      # (1, 3, 300, 300)
48      blob = cv2.dnn.blobFromImage(frame_resized, 0.007843, (300, 300), (127.5, 127.5, 127.5), False)
49      #Set to network the input blob
50      net.setInput(blob)
```

```

54     #Prediction of network
55     detections = net.forward()
56
57     #Size of frame resize (300x300)
58     cols = frame_resized.shape[1]
59     rows = frame_resized.shape[0]
60
61     #For get the class and location of object detected,
62     # There is a fix index for class, location and confidence
63     # value in @detections array .
64     for i in range(detections.shape[2]):
65         confidence = detections[0, 0, i, 2] #Confidence of prediction
66         if confidence > args.thr: # Filter prediction
67             class_id = int(detections[0, 0, i, 1]) # Class label
68
69             # Object location
70             xLeftBottom = int(detections[0, 0, i, 3] * cols)
71             yLeftBottom = int(detections[0, 0, i, 4] * rows)
72             xRightTop = int(detections[0, 0, i, 5] * cols)
73             yRightTop = int(detections[0, 0, i, 6] * rows)
74
75             # Factor for scale to original size of frame
76             heightFactor = frame.shape[0]/300.0
77             widthFactor = frame.shape[1]/300.0
78             # Scale object detection to frame
79             xLeftBottom = int(widthFactor * xLeftBottom)
80             yLeftBottom = int(heightFactor * yLeftBottom)
81             xRightTop = int(widthFactor * xRightTop)
82             yRightTop = int(heightFactor * yRightTop)
83             # Draw location of object
84             cv2.rectangle(frame, (xLeftBottom, yLeftBottom), (xRightTop, yRightTop),
85                           (0, 255, 0))
86
87             # Draw label and confidence of prediction in frame resized
88             if class_id in classNames:
89
90                 label = classNames[class_id] + ": " + str(confidence)
91                 labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
92
93                 yLeftBottom = max(yLeftBottom, labelSize[1])
94                 cv2.rectangle(frame, (xLeftBottom, yLeftBottom - labelSize[1]),
95                               (xLeftBottom + labelSize[0], yLeftBottom + baseLine),
96                               (255, 255, 255), cv2.FILLED)
97                 cv2.putText(frame, label, (xLeftBottom, yLeftBottom),
98                             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
99
100             #print(label) #print class and confidence
101     str_up = 'count: '+str(count)
102     cv2.putText(frame, str_up, (10, 40), font, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
103     cv2.namedWindow("frame", cv2.WINDOW_NORMAL)
104     cv2.imshow("frame", frame)
105     if cv2.waitKey(1) >= 0: # Break with ESC
106         break
107 #python mobilenet_ssd_python.py --video tes1.MP4 --prototxt MobileNetSSD_deploy.prototxt --weights
108 # MobileNetSSD_deploy.caffemodel

```

2. Kết quả



Hình 5: Kết quả phát hiện xe.

IV. Đánh giá

1. Kết quả đạt được

- Nghiên cứu và cài đặt thành công thuật toán SSD để phát hiện đối tượng và phân loại xe.

2. Hạn chế

- Chưa đếm được số lượng xe.

3. Hướng phát triển

- Giải quyết vấn đề đếm xe.
- Hoàn thiện bài toán tốt hơn để áp dụng vào thực tế giải quyết vấn đề ùn tắc giao thông.

V. Tài liệu tham khảo

<https://thigiacmaytinh.com/phat-hien-doi-tuong-p1-ly-thuyet/>

<https://iq.opengenus.org/single-shot-detection-ssd-algorithm/>

SSD: Single Shot MultiBox Detector Research Paper Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg <https://arxiv.org/abs/1512.02325>

Original Implementation

(CAFFE) <https://github.com/weiliu89/caffe/tree/ssd>