MỤC LỤC

[LỜI MỞ ĐẦU 2](#_Toc42265607)

[CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VÀ CÔNG NGHỆ SỬ DỤNG 3](#_Toc42265608)

[1. Tổng quan về nhận diện biển số xe 3](#_Toc42265609)

[1.1. Tổng quan 3](#_Toc42265610)

[1.2. Lịch sử phát triển 3](#_Toc42265611)

[1.3. Ứng dụng thực tiễn tại Việt Nam 4](#_Toc42265612)

[2. Công nghệ sử dụng 5](#_Toc42265613)

[2.1. CNN (Convolution Neural Network) 5](#_Toc42265614)

[2.2. SDD (Single Shot Detector) 10](#_Toc42265615)

[CHƯƠNG 2: TÌM KIẾM DỮ LIỆU VÀ TRIỂN KHAI MÔ HÌNH 15](#_Toc42265616)

[1. Tìm kiếm và xử lý dữ liệu 15](#_Toc42265617)

[1.1. Tìm kiếm 15](#_Toc42265618)

[1.2. Xử lý dữ liệu 15](#_Toc42265619)

[2. Xây dụng mô hình huấn luyện 16](#_Toc42265620)

[2.1. VGG16 17](#_Toc42265621)

[2.2. EXTRAS LAYER 17](#_Toc42265622)

[2.3. LOCATION LAYER VÀ CONFIDENCE LAYER 18](#_Toc42265623)

[2.4. DEFAULT BOX 18](#_Toc42265624)

[2.5. NON-MAXIMUN SUPRESSION(NMS) 20](#_Toc42265625)

[2.6. LOSS FUNCTION 21](#_Toc42265626)

[2.7. HUẤN LUYỆN MÔ HÌNH 23](#_Toc42265627)

[CHƯƠNG 3: KẾT QUẢ CỦA TỪNG BƯỚC 24](#_Toc42265628)

[1. Tách biển số 24](#_Toc42265629)

[2. Tách kí tự 24](#_Toc42265630)

[3. Nhận diện kí tự: 25](#_Toc42265631)

[4. Tổng hợp các kết quả 26](#_Toc42265632)

[CHƯƠNG 4: HẠN CHẾ VÀ HƯỜNG PHÁT TRIỂN 27](#_Toc42265633)

[1. Hạn chế 27](#_Toc42265634)

[2. Hướng phát triển 27](#_Toc42265635)

[DANH MỤC TÀI LIỆU THAM KHẢO 27](#_Toc42265636)

# 

# LỜI MỞ ĐẦU

Thị giác máy tính là một lĩnh vực đang rất được quan tâm hiện nay. Ứng dụng về thị giác máy tính rất rộng rãi có thể áp dụng được cho rất nhiều lĩnh vực như y tế, giáo dục, quốc phòng, an ninh,.. Tuy nhiên, thị giác máy tính đang gặp rất nhiều thách thức để đạt tới mức độ thị giác tự nhiên của con người. Một trong những thách thức đối với thị giác máy tính đó là làm sao có thể xác định chính xác vị trí của đối tượng để đưa ra dự đoán về đối tượng nhằm đưa ra những mô tả về đối tượng ví dụ: xác định con người/máy, biển số xe, v..v.

Những năm gần đây, ta đã chứng kiến được nhiều thành tựu vượt bậc trong ngành Thị giác máy tính (Computer Vision). Các kiến trúc Object Detection xử lý về Computer Vision ngày càng hoàn thiện và đạt tới độ chính xác cao, thậm trí vượt trội so với khả năng quan sát của con người nếu nguồn dữ liệu chính xác và được xử lý trọn vẹn. Một vài thuật toán đáp ứng được việc xử lý nhanh áp dụng cho real-time trong các hệ thống camera như : SSD, Yolo(v4), R-CNN(2014), Fast R-CNN(2015), v..v.

Convolutional Neural Network (CNNs – Mạng neural tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

Các kiến trúc object detection đều dựa trên một deep CNN network chẳng hạn như VGG16 hoặc Alexnet ở giai đoạn đầu giúp trích lọc features và nhận diện các region proposal. Sau đó phát triển thuật toán nhằm tìm ra bounding box và confidence của đối tượng chứa trong bounding box. Tùy vào thiết kế mà các mô hình có thể theo dạng pipeline hoặc trong một single model. Tốc độ xử lý của mô hình phụ thuộc vào số lượng bounding box mà nó tạo ra.

Nội dung bài báo cáo bao gồm 4 chương:

* + Chương 1: Giới thiệu bài toán và công nghệ sử dụng
  + Chương 2: Tìm kiếm dữ liệu và triển khai mô hình
  + Chương 3: Kết quả của từng bước huẩn luyện
  + Chương 4: Hạn chế và hướng phát triển

# CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VÀ CÔNG NGHỆ SỬ DỤNG

## 1. Tổng quan về nhận diện biển số xe

### 1.1. Tổng quan

Hệ thống nhận dạng biển số xe là hệ thống có khả năng phân tích hình ảnh và xác định vùng chứa biển số trên xe, thông qua video, thiết bị ghi hình hoặc hình ảnh.

Đầu vào của hệ thống sẽ là ảnh chụp/video từ camera với kích thước bất kì. Yêu cầu hệ thống nhanh chóng xác định được vị trí của biển số từ đó xác định được giá trị các con số trên biển.



### 1.2. Lịch sử phát triển

Năm 1992, công nghệ ALPR (Automatic License Plate Number) hay còn gọi là tự động nhận dạng biển số xe, được phát triển tại Đại học Cambridge ở Vương quốc Anh để ứng phó với chủ nghĩa khủng bố.

Đến năm 1996, công nghệ ALPR đã được hoàn thiện tại mỗi cổng phía Tây Vương quốc Anh để đọc tất cả các biển đăng ký xe từ Ireland. Công nghệ ALPR tiếp tục được nghiên cứu và phát triển tại Anh. Kể từ tháng ba năm 2006, hầu hết các con đường, các trung tâm thị trấn, cảng, trạm xăng của London đã được lắp đặt camera chạy phần mềm ALPR.

Trên thế giới hiện nay, bài toán nhận dạng biển số xe được nghiên cứu và phát triển một cách sâu rộng. Nhiều tác giả với các công trình nghiên cứu được công bố với tỉ lệ nhận dạng ngày càng chính xác. Một số bài báo cáo nghiên cứu của các tác giả tiêu biểu trong vài năm trở lại đây như:

* Chirag N. Paunwala, 2010 với nội dung: rút trích vùng số xe trong ảnh. Ảnh đầu vào được tiền xử lý bằng cách phương pháp nâng cao chất lượng ảnh, sau đó tìm biên bằng Vertical Edge và xử lý một lần nữa bằng Opening và Closing. Các vùng ứng viên sau đó được kiểm tra bằng thuật toán scan theo dòng để tìm được vùng chứa biển số xe chính xác. Kết quả nhận dạng 750 ảnh trong các điều kiện khác nhau cho tỉ lệ 742/750 = 99.2.
* Choo Kar Soon, 2012 với nội dung: nhận dạng biển số xe tại Malaysia, sử dụng giải thuật Adaboots để training tập dữ liệu gồm gần 100 ảnh biển số. Các ký tự được nhận dạng bằng phương pháp KNN. Kết quả nhận dạng biển số 98% và nhận dạng ký tự 95% trên ảnh tĩnh. Báo cáo này nghiên cứu cách nhận dạng biển số xe với sự kết hợp của phép biến đổi Hough và giải thuật tìm Contour để cải thiện kết quả phát hiện vùng các ứng viên sau đó tiếp tục được scan theo dòng để đếm số đối tượng bị cắt và so sánh với ngưỡng, nhằm tìm ra vùng ứng viên thõa mãn. Kết quả nhận dạng đạt 98-99%.

Phần mềm nhận dạng biển số xe đã được ứng dụng thực tế tại các trạm cân, trạm gửi xe, các trụ đèn giao thông để phát hiện xe vi phạm.

### 1.3. Ứng dụng thực tiễn tại Việt Nam

Hệ thống nhận dạng biển số xe được xây dựng nhằm mục đích giám sát, kiểm soát các phương tiện. Dưới đây chúng ta đề cập đến một số ứng dụng phổ biến đối với hệ thống nhận dạng biển số xe:

* Thu phí giao thông: lắp đặt hệ thống “Nhận dạng biển số xe” tại các trạm thu phí nhằm hỗ trợ hoặc tự động hóa công tác thu phí.
* Kiểm soát xe tại các đường biên giới: mỗi quốc gia đều có những quy định riêng về biển số xe, để phục vụ cho công tác quản lý và phát hiện những phương tiện giao thông (xe) vượt biên giới bất hợp pháp. Việc lắp hệ thống “Nhận dạng biển số xe” tại các trạm kiểm soát sẽ góp phần hỗ trợ công tác kiểm tra và an ninh quốc gia.
* Các trạm gác cổng: việc lắp đặt hệ thống “Nhận dạng biển số xe” sẽ hỗ trợ hoặc tự động hóa công tác mở cổng cho xe vào. Ngoài ra, hệ thống còn được ứng dụng vào công tác chống trộm xe, các bãi giữ xe tự động, điều tiết giao thông (chẳng hạn như Thành phố Dublin đã ứng dụng công nghệ “Nhận dạng biển số xe tự động” trong việc điều tiết giao thông theo dạng biển số chẵn/lẻ).

## 

## 2. Công nghệ sử dụng

### 2.1. CNN (Convolution Neural Network)

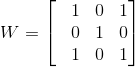
#### 2.1.1. Các Layer của mạng CNN

**Convolution layer**

Ảnh xám được biểu diễn dưới dạng ma trận A kích thước *m\*n*.

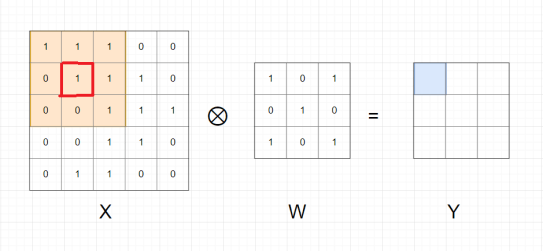
Ta định nghĩa **kernel**là một ma trận vuông kích thước *k\*k* trong đó k là số lẻ (1, 3, 5, 7, 9,…)

Ví dụ kernel kích thước *3\*3*

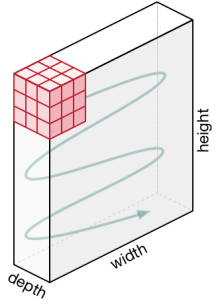


Kí hiệu phép tính convolution (⊗), kí hiệu Y = *X*⊗*W*

Với mỗi phần tử *xij*​ trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử  *xij*​ làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

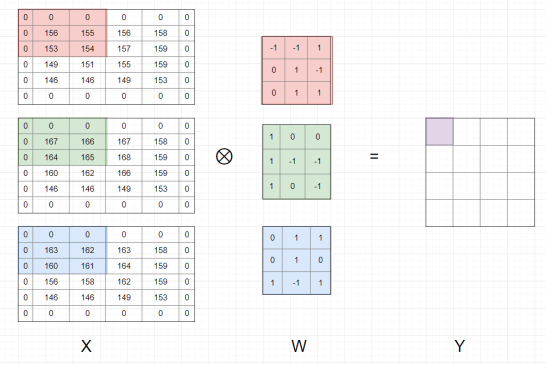


Tuy nhiên ảnh màu có tới 3 channels red, green, blue nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Nên ta cũng sẽ định nghĩa kernel là 1 tensor 3 chiều kích thước *k\*k\*3.*



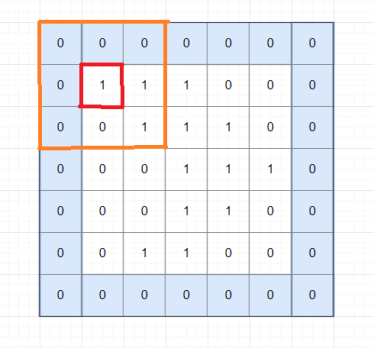
***Phép tính convolution trên ảnh màu với k=3.***

Ta định nghĩa kernel có cùng độ sâu (depth) với biểu diễn ảnh, rồi sau đó thực hiện di chuyển khối kernel tương tự như khi thực hiện trên ảnh xám.



**Padding**

Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên giờ ta muốn ma trận Y thu được có kích thước bằng ma trận X => Tìm cách giải quyết cho các phần tử ở viền => Thêm giá trị 0 ở viền ngoài ma trận X.



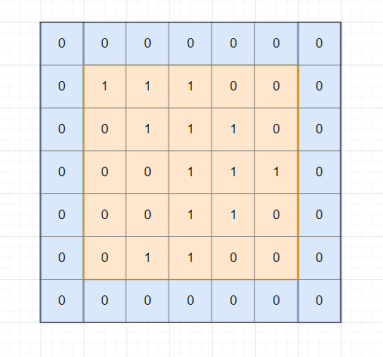
*Ma trận X khi thêm viền 0 bên ngoài*

Rõ ràng là giờ đã giải quyết được vấn đề tìm A cho phần tử *x*11​ , và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu.

Phép tính này gọi là convolution với **padding=1**. Padding=k nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

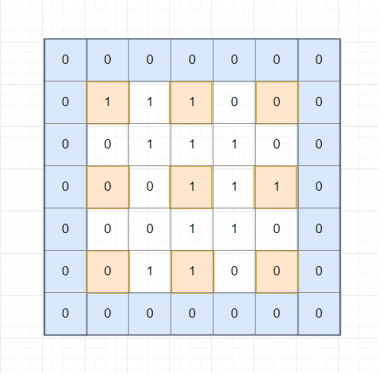
**Stride**

Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X, thu được ma trận Y cùng kích thước ma trận X, ta gọi là stride=1.



*stride=1, padding=1*

Tuy nhiên nếu **stride=k** (k > 1) thì ta chỉ thực hiện phép tính convolution trên các phần tử  *x*1+*i*∗*k*,1+*j*∗*k*​. Ví dụ k = 2.



*padding=1, stride=2*

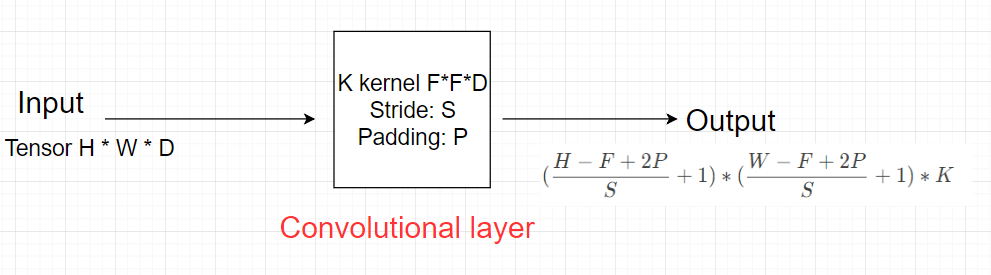
Hiểu đơn giản là bắt đầu từ vị trí *x*11​ sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X.

***Convolutional layer tổng quát***

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H \* W \* D.

Kernel có kích thước F \* F \* D (kernel luôn có depth bằng depth của input và F là số lẻ), stride: S, padding: P. Convolutional layer áp dụng K kernel.

=> Output của layer là tensor 3 chiều có kích thước:

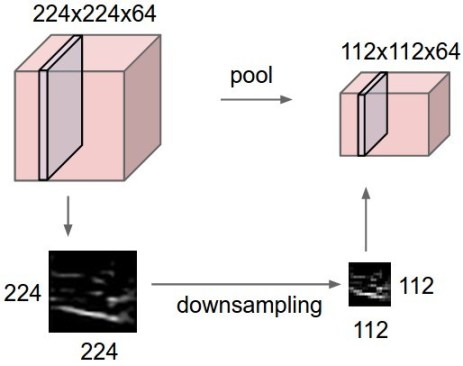


**Pooling layer**

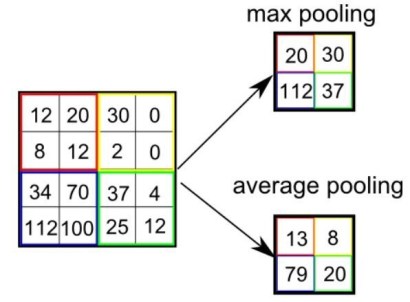
Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

Gọi pooling size kích thước K\*K. Input của pooling layer có kích thước H\*W\*D, ta tách ra làm D ma trận kích thước H\*W. Với mỗi ma trận, trên vùng kích thước K\*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

Nhưng hầu hết khi dùng pooling layer thì sẽ dùng size=(2,2), stride=2, padding=0. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên .



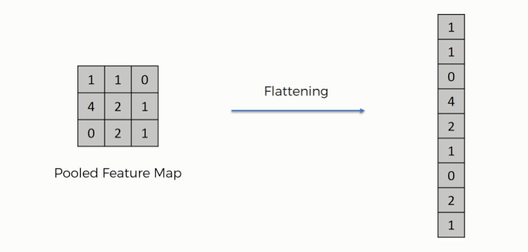
Có 2 loại pooling layer phổ biến là: max pooling và average pooling.



Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

**Fully connected layer**

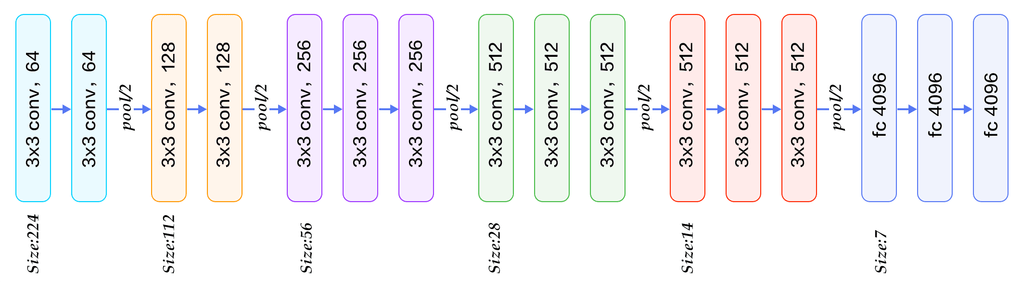
Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,…) thì tensor của output của layer cuối cùng, kích thước *H\*W\*D*, sẽ được chuyển về 1 vector kích thước (*H\*W\*D)*



Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.

#### 2.1.1. Cấu trúc mạng VGG16

VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu [***ImageNet***](http://www.image-net.org/)gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ áp dụng kiến thức ở trên để phân tích mạng VGG16.



***Kiến trúc vgg16***

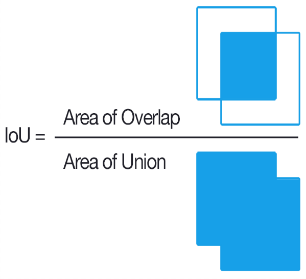
Phân tích:

* Convolutional layer: kích thước *3\*3*, padding=1, stride=1. Mặc định sẽ là stride=1 và padding để cho output cùng width và height với input.
* Pool/2 : max pooling layer với size *2\*2*
* *3\*3* conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
* Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
* Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

### 2.2 SDD (Single Shot Detector)

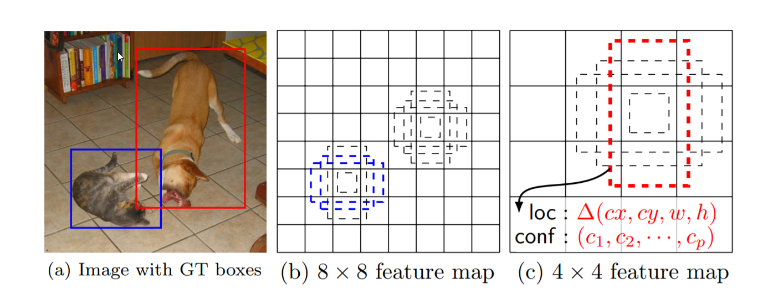
**Một số định nghĩa:**

* **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∈(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 (,,w,h) giúp xác định vật thể.
* **offsets**: Các tọa độ (,,w,h) để xác định vật thể.
* **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 gữa 2 khung hình với phần tổng diện tích giao nhau và không giao nhau giữa chúng

.

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

**Single Shot Detector là gì?**



*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 tợp 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 8x8 và 4x4 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 **c = (,,…,)** tương ứng với các class **C = ,,…,**. 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.

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.

**Quá trình huấn luyện**

**Chiến lược mapping default box** Trong suốt quá trình huấn luyện ta cần mapping các default boxes 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**](https://en.wikipedia.org/wiki/Jaccard_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 qui 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 = 0 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 ∑ ≥1. Hàm loss function là tổng có trọng số của localization loss (loc) và confidence loss (conf):

L(x,c,p,g) =( (x,c)+ α(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:

1. **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*.

(x,p,g) =

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

Thành phần  chính là tổng khoảng cách giữa predicted box (p) và ground truth box (gg) 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.

2. **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:

(x,c) = −

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 :

−

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  =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  = 0.9 theo công thức:

= + (k - 1), k∈[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 \*

h =

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

=

**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 *with* và *height*), và *aspect ratio* ta xác định kích thước các cạnh của các *bounding box* theo công thức:
* **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:
* **Bước 3**: Tính tọa độ các điểm (,,w,h) trên hình ảnh gốc dựa trên phép *linear interpolation* qua hàm **[np.linspace()](https://docs.scipy.org/doc/numpy/reference/generated/numpy.linspace.html)**.

Kết quả trả về là một tensor có shape là (feature\_width, feature\_height, n\_boxes, 8), trong đó chiếu cuối cùng = 8 tương ứng với 4 *offsets* của *default bounding box* và 4 *variances* đại diện cho các *scales* của *default bounding box*.

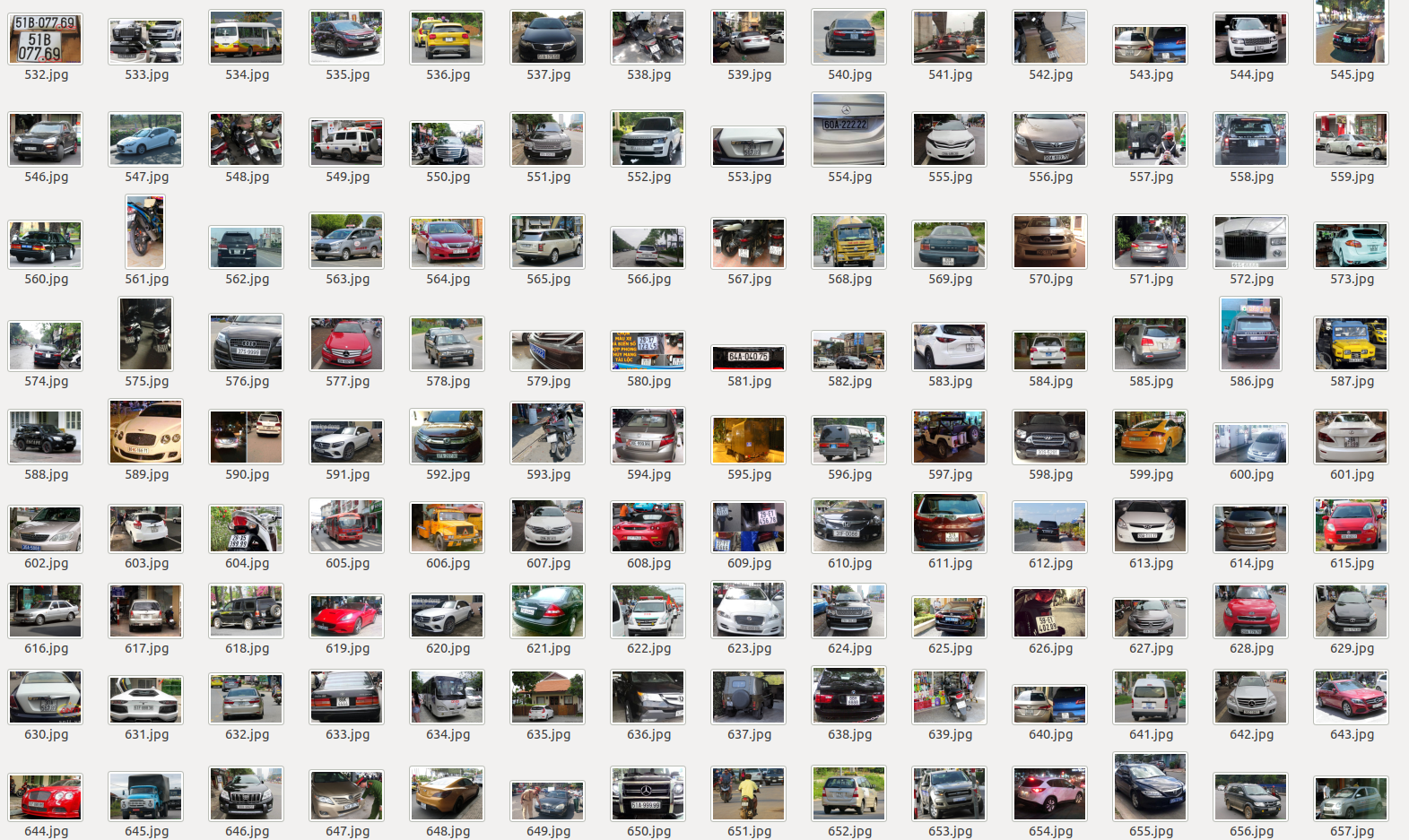
# CHƯƠNG 2: TÌM KIẾM DỮ LIỆU VÀ TRIỂN KHAI MÔ HÌNH

## 1. Tìm kiếm và xử lý dữ liệu

### 1.1. Tìm kiếm

Bài toàn nhận diện biển số xe cần rất nhiều ảnh gồm các loại phương tiện có chứa biển số. Đây là mô hình xác định đối tượng nên lượng dữ liệu tương đối nhiều và từ nhiều loại xe khác nhau nhằm có tính tổng quát và tính chính xác với tất cả cảc loại ảnh.

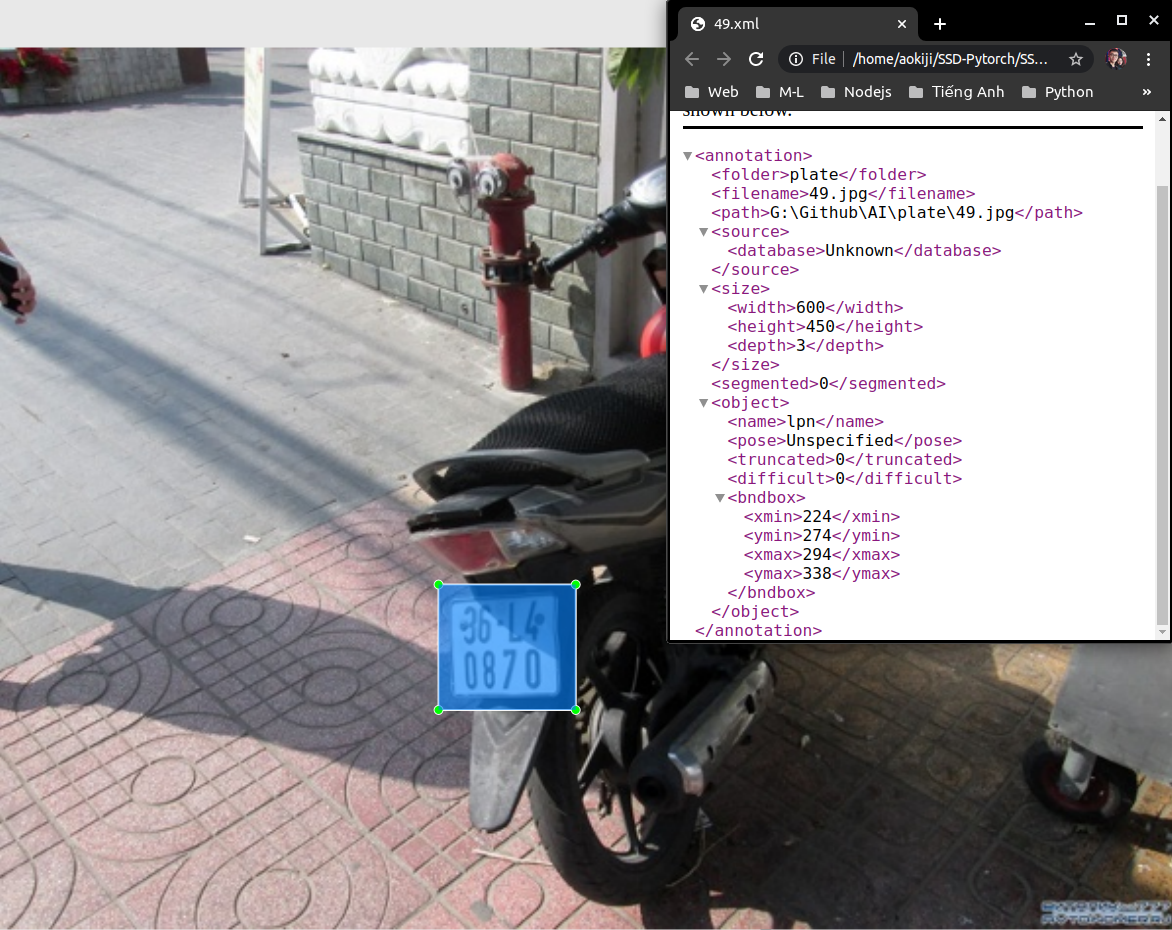
Nguồn dữ liệu được tìm kiếm từ Google và tự chụp gần 3000 bức ảnh bao gồm tất cả các loại ảnh và các phương tiện.



### 1.2. Xử lý dữ liệu

*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. Do đó cần xử lý 1 bức ảnh bằng cách xác định được vị trí 4 điểm xung quang một đối tượng của bức ảnh.

m sử dụng *labelImg* để xác định đối tượng. Các thông số sau khi được xác định sẽ được lưu vào file *annotation.xml*.

****

***height, width*** *: chiều cao và chiều rộng bức ảnh,* ***name*** *: nhãn được ghi là “lpn”*

***xmin, ymin, xmax, ymax*** *: tọa độ 4 điếm xung quanh biển số*

Sau đó sẽ viết hàm để chuẩn hóa các thông số theo đúng chuẩn của mô hình để huấn luyện : xmin /= width, ymin /= height, xmax /= width, ymax /=height

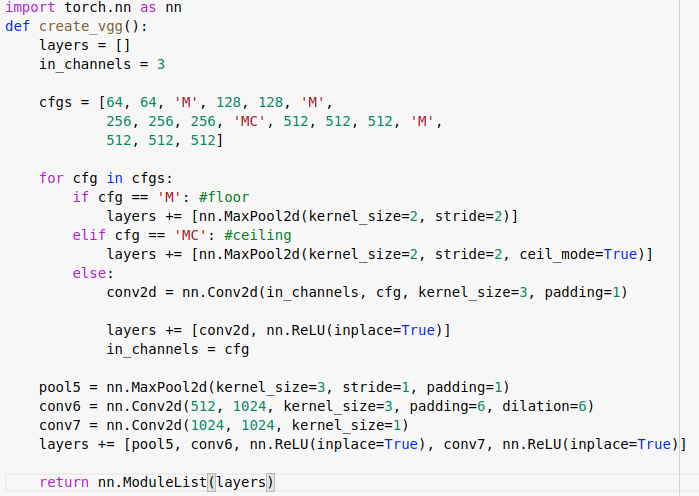
## 2. Xây dụng mô hình huấn luyện

**A screenshot of a cell phone

Description automatically generated**

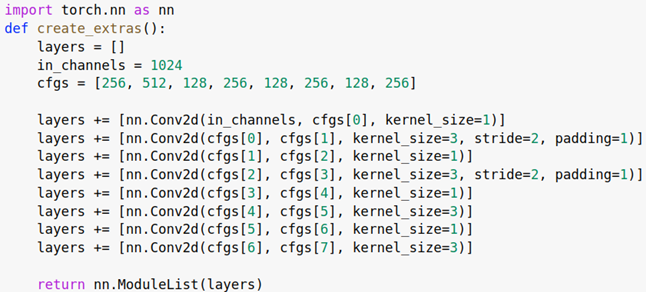
### 2.1. VGG16

***SSD*** sử dụng *base network* là ***VGG 16*** để trích xuất các feature map *38x38, 19x19*. Mô hình sử dụng nhiều *Convolution layer* với kích thước khác nhau xen kẽ là các *Maxpooling layer, Relu layer* nhằm trích xuất các đặc trưng quan trọng để xác định được các đối tượng kích thước nhỏ.



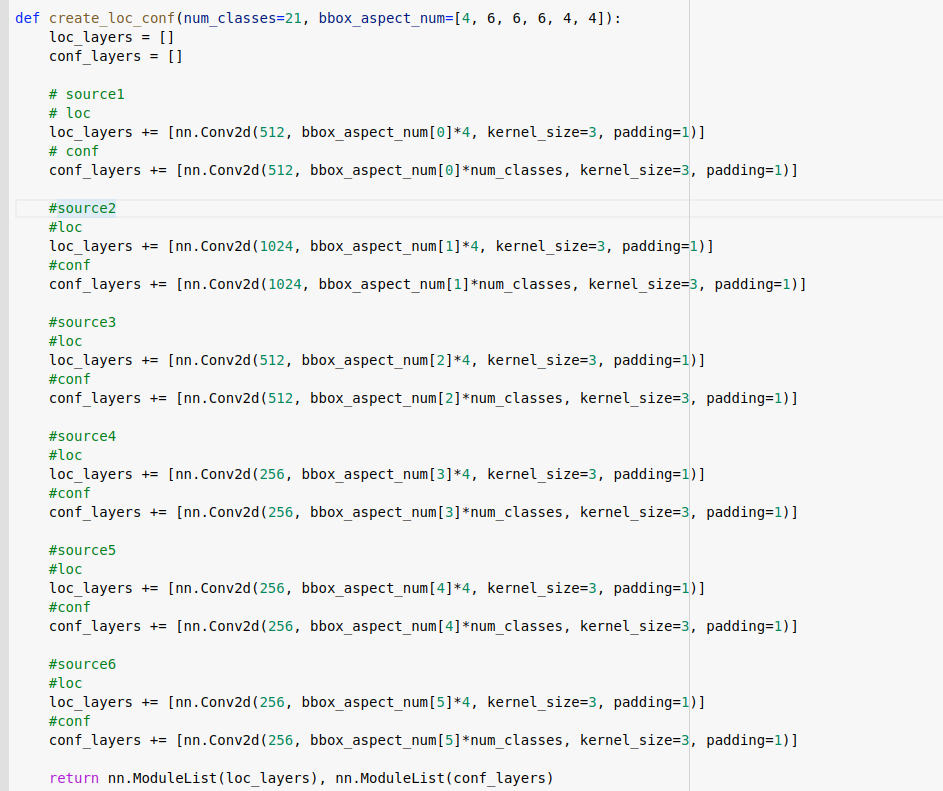
### 2.2. EXTRAS LAYER

Sau khi ảnh được lan truyền qua mạng **VGG 16** và trích xuất được các *feature map 38x38, 19x19* cần xây dựng thêm các lớp **Convolution2D** trích xuất được các *feature map* nhỏ hơn là *10x10, 5x5, 3x3, 1x1* để có thêm nhiều đặc trưng xác định các đối tượng lớn.



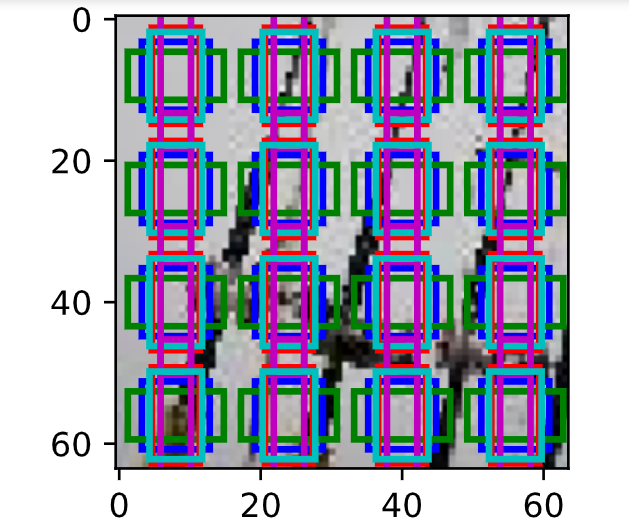
### 2.3. LOCATION LAYER VÀ CONFIDENCE LAYER

Ảnh sau khi qua các lớp **VGG 16** và **Extras** sẽ thu được output là các **source feature map** *38x38, 19x19, 10x10, 5x5, 3x3, 1x1*. Mỗi pixel trên các *feature map* sẽ xác định được 4 hoặc 6 các box bao quanh pixel với tỉ lệ *width/height* khác nhau (*aspect ratio*) và độ tự tin (*confidence* ) của pixel với từng đối tượng cần xác định. Qua lớp này, với 1 ảnh đầu vào *300x300* **SSD** sẽ đưa ra output gồm 1 ma trận *1x8732x4* mô tả 8732 các box xác định được đồng thời độ tự tin khi xác định đối tượng của từng box trong ma trận *1x8732x2* (2 gồm biển số xe/kí tự và *background*)



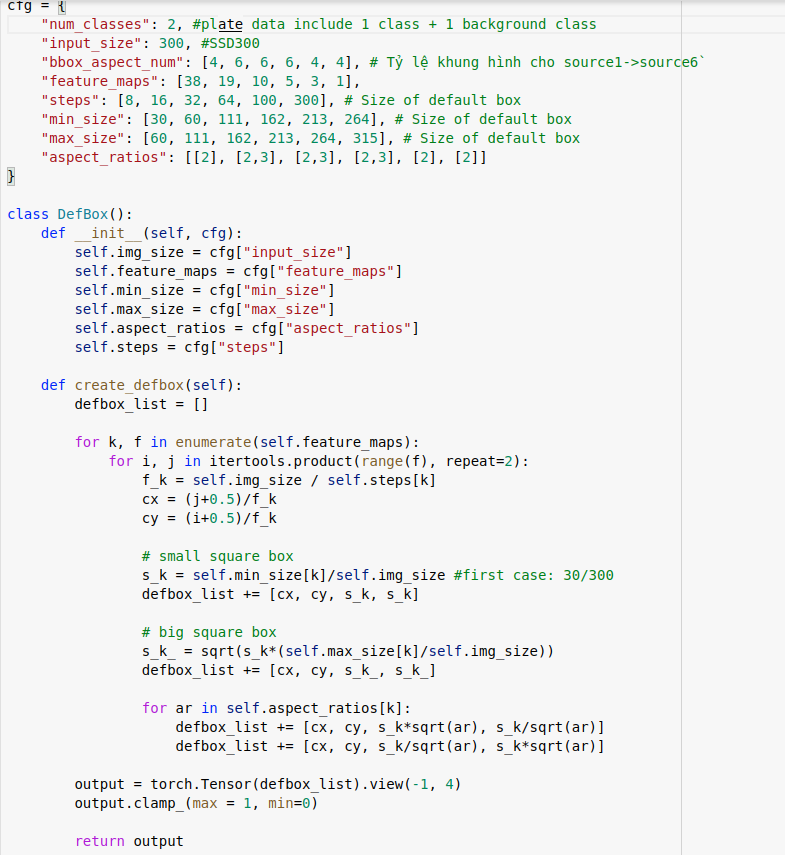
### 2.4. DEFAULT BOX

Chúng ta cần liên kết một tợp 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.



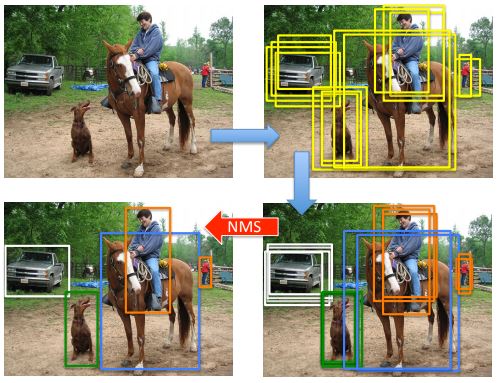
Tại mỗi một *default bounding box* của *feature map* chúng ta dự báo *4 offsets* tương ứng với một tọa độ và kích thước của nó. *4 offsets* ở đây được hiểu là một tọa độ gồm 4 tham số (,,w,h) . Trong đó ,) giúp xác định tâm và *(w, h*) là kích thước dài rộng của *bounding box.* Thành phần thứ 2 được 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 x* *n* tương ứng với *p channels* (chẳng hạn như kích thước 8 x 8 hoặc 4 x 4), một kernel filter kích thước *3 x 3 x 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 k 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=() và 4 offsets tương ứng với kích thước ban đầu của *default bounding box*.
* Kết quả cuối cùng ta thu được *(C+4) x mnk* outputs.

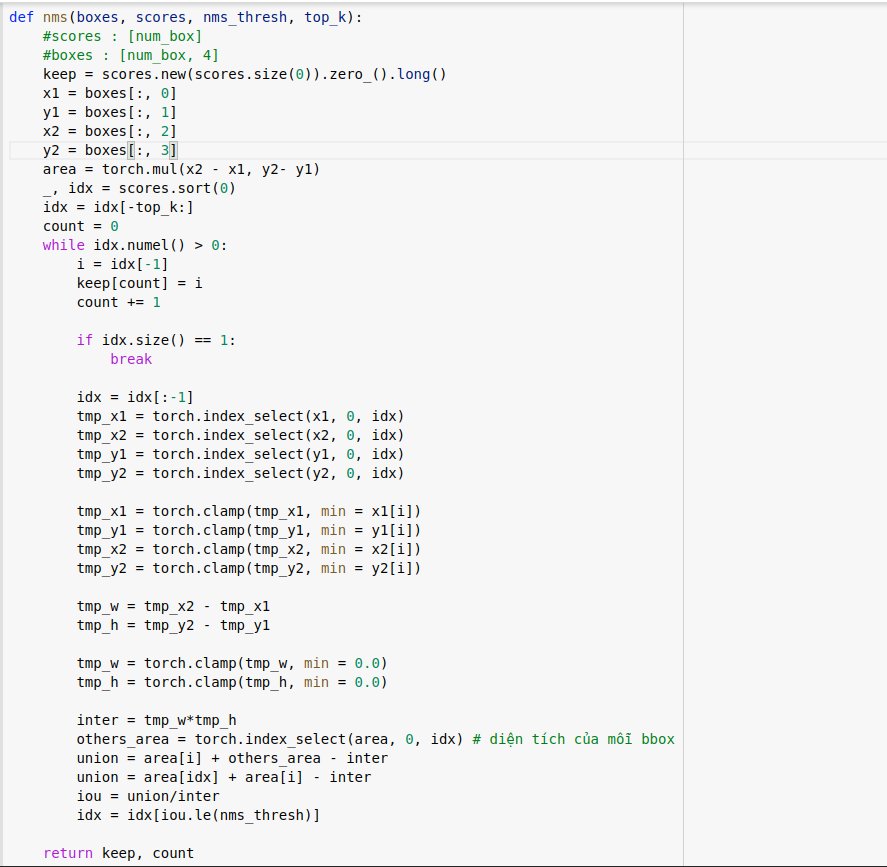


### 2.5. NON-MAXIMUN SUPRESSION(NMS)

Sau khi thực hiện *object detection feed* một ảnh qua mạng neural, chúng ta sẽ thu được rất nhiều *proposals* (như hình ở dưới).



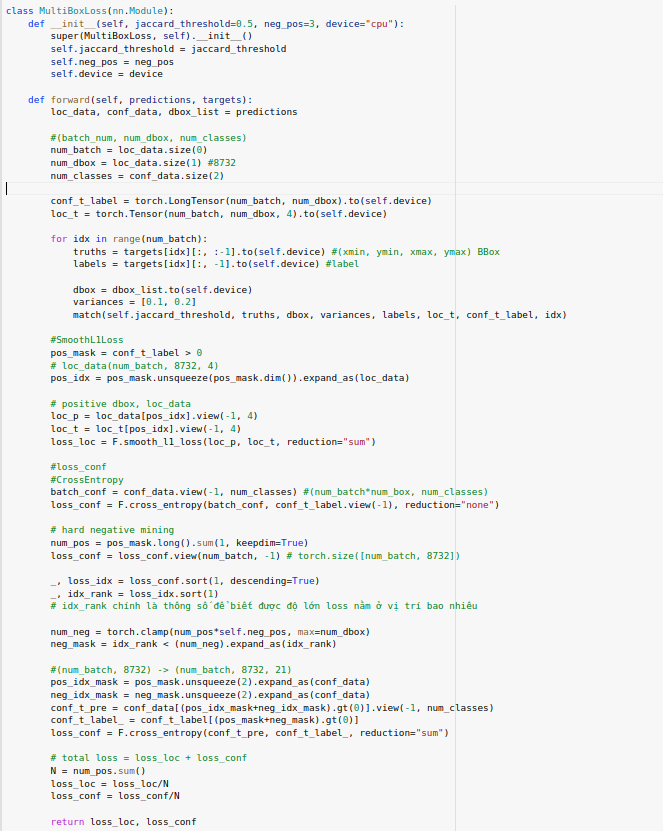
Ở trạng thái này, có rất nhiều *proposals* là *boding box* cho một object duy nhất, điều này dẫn tới việc dư thừa. Chúng ta sử dụng thuật toán **Non-maximum suppression (NMS)** để giải quyết bài toán này.



### 2.6. LOSS FUNCTION

Việc xây dựng Loss Function cho **SSD** tương đối phức tạp. Trong phạm vi của báo cáo, em chỉ đưa ra hàm mà không đi vào xây dựng chi tiết.

Hàm Loss Funtion sẽ nhận đầu vào là output của ảnh khi lan truyền qua mạng SSD, kết quả lan truyền qua mạng là *predicted box* ,, w, h) và độ tự tin (confidence) đối với đối tượng cụ thể và danh sách 8732 các default box. Hàm Loss Function sẽ tính toán sai lệch so với **ground truth box của đối tượng và cross-entropy so với nhãn của đối tượng trong C classes.**

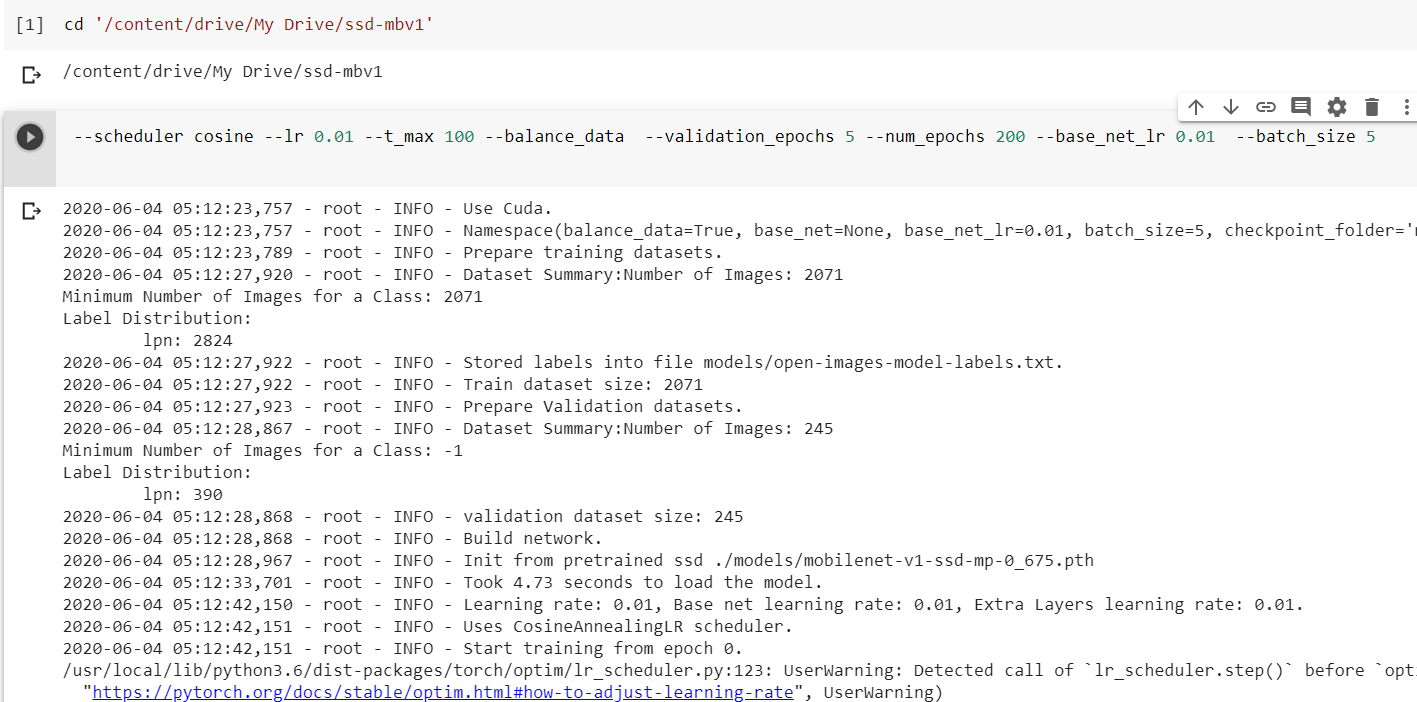


### 2.7. HUẤN LUYỆN MÔ HÌNH

Sau khi tìm kiểm, xử lý và chuẩn hóa dữ liệu, xây dựng các lớp cần thiết để trích xuất các *feature map*, các *layer* để dự đoán độ tự tin của từng pixel với các đối tương, hàm mất mát chúng ta sẽ dùng *Colab* để huấn luyện mô hình tách biển số ra khỏi ảnh.

Chúng ta sẽ thêm xử lý và chuẩn hóa dữ liệu của các biển số sau khi được tách thành các box kí tự (em sẽ trình bày ở mục sau). Và huấn luyện mô hình tách kí tự trên *Colab*.

Cuối cùng sẽ cần huấn luyện một mô hình để phân lớp các ký tự theo đúng nhãn của chúng.



# CHƯƠNG 3: KẾT QUẢ CỦA TỪNG BƯỚC

## 1. Tách biển số

Ảnh đầu vào sau khi qua xử lý (chuyển về kích thước *300x300*) và đưa qua mạng SSD với các trọng số *w* đã được huấn luyện ở từng layer. Kết quả bước 1 thu được là hình ảnh biển số xe.

A truck is parked in front of a building

Description automatically generated 🡺 A close up of a sign

Description automatically generated

## 2. Tách kí tự

Dựa vào mô hình 1 tách tất cả các biển số xe có trong tập dữ liệu ban đầu và làm dữ liệu tương tự như xử lý dữ liệu ở bước 1

A picture containing monitor, clock, white

Description automatically generated

Đưa tập dữ liệu mới bao gồm ảnh của biển số xe và vị trí các kí tự đã được chuẩn hóa huấn luyện mô hình thứ 2 : mô hình tách kí tự.

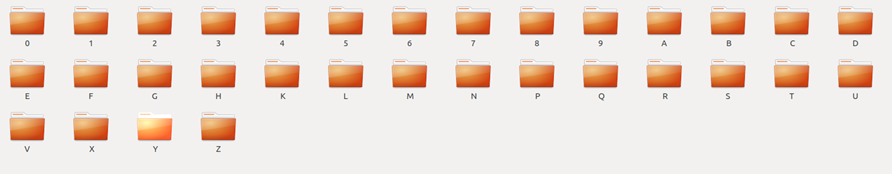
A close up of a sign

Description automatically generated

*Kết quả thu được sau bước 2 : các kí tự trong biển số xe đã được tách ra*

## 3. Nhận diện kí tự:

Sử dụng mô hình thứ 2 để tách toàn bộ các kí tự trong các ảnh biển số ở tập dữ liệu thứ 2 và chia chúng vào đúng folder để thực hiện bước 3 : phân lớp ký tự (0-9, A-Z)

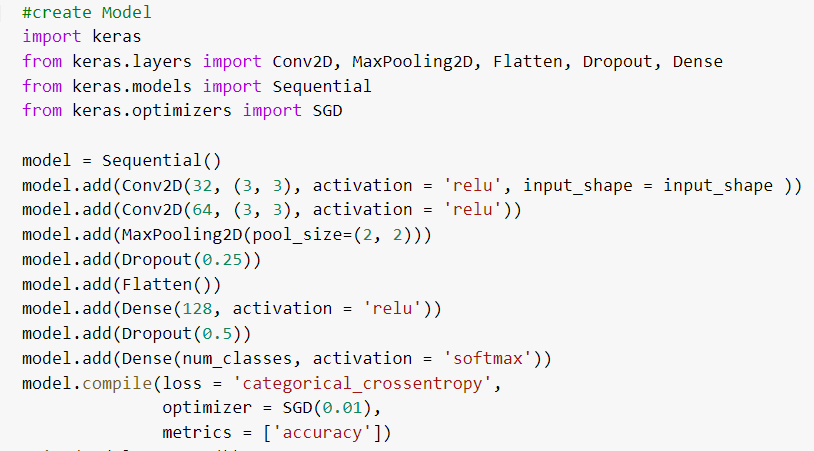


*Các folder ảnh của ký tự tách từ biển*



*Ví dụ folder 5*

Xử lý dữ diệu và làm các layer mô hình thứ 3: Classification



*Huấn luyện mô hình bằng Colab*

Kết quả ở bước 3 : Các kí tự khi đưa qua mạng sẽ được nhận diện

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| A | 4 | 6 | 4 | 0 | 3 | 5 | 1 |

## 4. Tổng hợp các kết quả

Kết hợp kết quả của các bước và sắp xếp lại vị trí các kí tự và hiển thị kết quả .

A truck is parked in front of a building

Description automatically generated

# CHƯƠNG 4: HẠN CHẾ VÀ HƯỜNG PHÁT TRIỂN

## 1. Hạn chế

Dữ liệu làm thủ công chia đều cho các thành viên nên việc vẽ box lệch quá hoặc vẽ sai sẽ dẫn đến sai lệch trong việc huấn luyện.

Mô hình đã cho kết quả khả quan và tương đối chính xác nhưng vẫn không thể đưa vào ứng dụng thực tế vì tốc độ xử lý ảnh là khá cao : trung bình 0.15s cho 1 ảnh không phù hợp với video (các frame ảnh được đọc liên tục).

Mô hình dự đoán tốt các ảnh có biển số to, rõ ràng nhưng không dự đoán thật chính xác các ảnh có biển số mờ hoặc nhỏ, dẫn đến việc tách và nhận diện kí tự gặp khó khăn.

Tập dữ liệu ở mô hình dự đoán kí tự không được tốt vì mất cân bằng dữ liệu. Do nguồn dữ liệu được lấy trực tiếp trong mô hình thứ 2 nên lượng dữ liệu chữ và số không đồng đều dẫn đến có thể nhận diện kí tự nhầm lẫn : 2 🡺 Z, 1 🡺 7, 2🡺L, 0 🡺 D,….

Mô hình **VGG-SSD** em đã chạy thử nghiệm và cho kết quả tốt tuy nhiên về tốc độ khá chậm do base-network sử dụng là mạng khá nặng chưa rất nhiều params. Vì vậy em đã sử dụng Mô hình nhẹ hơn rất nhiều có thể chạy real-time là **Mobilenet-SSD300** được mô tả ở phần “Tài liệu tham khảo” với các bước hướng dẫn cụ thể.



*Nhận diện sai biển*

*Nhận dạng sai 2 thành L*

## 2. Hướng phát triển

Sử dụng các mô hình khác có độ chính xác cao và nhanh hơn : ***Yolov4, Faster R-CNN,…***

Làm dữ liệu chuẩn hơn : vẽ box chuẩn, dùng model đã train để tự vẽ dữ liệu, cân bằng dữ liệu các kí tự,…

# DANH MỤC TÀI LIỆU THAM KHẢO

[0]. Source code Mobilenet-SSD300 : <https://github.com/sonnh173341/license-plate-ssd300>

[1]. Ebook Machine learning cơ bản của tác giả Vũ Hữu Tiệp

[2]. Trang Deep Learning cơ bản: <https://nttuan8.com/>

[3]. Blog về Khoa học dữ liệu: <https://phamdinhkhanh.github.io>

[4]. Paper gốc của SSD: <https://arxiv.org/abs/1512.02325>

[5]. Xây dựng mô hình SSD : <https://github.com/qfgaohao/pytorch-ssd>