

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

—o0o—



**VĨNH THỨC VÀ ỨNG DỤNG TRONG**  
**BÀI TOÁN ĐẾM TỔ HỢP**

**ĐỒ ÁN III**

*Chuyên ngành:* **TOÁN TIN**

*Chuyên sâu:* **Xác suất rời rạc**

Giảng viên hướng dẫn: **TS. LÊ HẢI HÀ**

Họ và tên sinh viên: **LAI ĐỨC THẮNG**

Số hiệu sinh viên: **20163830**

Lớp: **Toán Tin K61**

**HÀ NỘI - 2020**

# Mục lục

<b>Lời mở đầu</b>	<b>2</b>
<b>1 Lý thuyết về hệ thống thông tin địa lý và tiền xử lý ảnh viễn thám</b>	<b>4</b>
1.1 Hệ thống thông tin địa lý . . . . .	4
1.2 Ảnh viễn thám và quá trình tiền xử lý . . . . .	4
<b>2 Lý thuyết học sâu và bài toán nhận diện vật thể</b>	<b>5</b>
2.1 Neural Network - Mạng Neural . . . . .	7
2.2 Mạng Neural tích chập (Convolutional Neural Network - CNN) . . . . .	9
2.3 Bài toán nhận diện vật thể (Object Detection) . . . . .	13
2.3.1 Mô hình Faster R-CNN . . . . .	14
2.3.2 Một số phương pháp đánh giá mô hình nhận diện vật thể . . . . .	20
2.3.3 PyTorch và Faster R-CNN ResNet-50 FPN . . . . .	24
2.3.4 PyTorch . . . . .	25
2.3.5 Faster R-CNN ResNet-50 FPN . . . . .	26
<b>3 Ứng dụng học sâu vào bài toán đếm cây trên ảnh viễn thám</b>	<b>28</b>
3.1 Giới thiệu bài toán . . . . .	28
3.2 Mô hình hoá bài toán và thiết kế dữ liệu luyện . . . . .	28
3.3 Huấn luyện mạng neural . . . . .	29
3.4 Xây dựng chương trình . . . . .	29
<b>4 Cài đặt chương trình và đánh giá kết quả</b>	<b>30</b>
4.1 Môi trường cài đặt chương trình và các yêu cầu liên quan . . . . .	30

4.2	Dữ liệu đầu vào . . . . .	30
4.3	Kết quả huấn luyện . . . . .	30
4.4	Đánh giá kết quả . . . . .	30
<b>Tài liệu tham khảo</b>		<b>31</b>

# Lời mở đầu

# Chương 1

## Lý thuyết về hệ thống thông tin địa lý và tiền xử lý ảnh viễn thám

### 1.1 Hệ thống thông tin địa lý

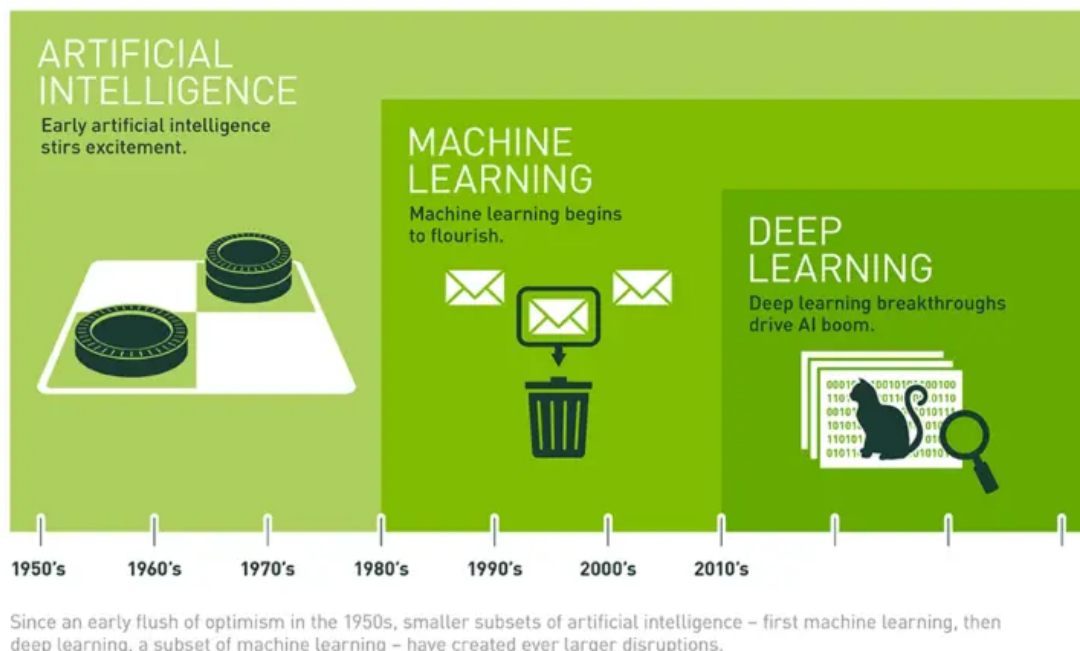
### 1.2 Ảnh viễn thám và quá trình tiền xử lý

## Chương 2

# Lý thuyết học sâu và bài toán nhận diện vật thể

Deep Learning là một kỹ thuật Machine Learning mà ở đó huấn luyện máy tính giống như cách thức tự nhiên của con người: Học qua các ví dụ. Những năm gần đây, Deep learning đã mang đến nhiều bất ngờ trên quy mô toàn cầu và dẫn đường cho những tiến triển nhanh chóng trong nhiều lĩnh vực khác nhau như thị giác máy tính, xử lý ngôn ngữ tự nhiên (natural language processing), nhận dạng giọng nói tự động (automatic speech recognition), học tăng cường (reinforcement learning), và mô hình hoá thống kê (statistical modeling). Với những tiến bộ này, chúng ta bây giờ có thể xây dựng xe tự lái với mức độ tự động ngày càng cao (nhưng chưa nhiều tới mức như vài công ty đang tuyên bố), xây dựng các hệ thống giúp trả lời thư tự động khi con người ngập trong núi email, hay lập trình phần mềm chơi cờ vây có thể thắng cả nhà vô địch thế giới, một kỳ tích từng được xem là không thể đạt được trong nhiều thập kỷ tới. Những công cụ này đã và đang gây ảnh hưởng rộng rãi tới các ngành công nghiệp và đời sống xã hội, thay đổi cách tạo ra các bộ phim, cách chẩn đoán bệnh và đóng một vài trò ngày càng tăng trong các ngành khoa học cơ bản – từ vật lý thiên văn tới sinh học. Với Deep Learning, một mô hình máy tính học cách thực hiện một công việc phân loại (classification) trực tiếp từ các hình ảnh, chữ viết (text) hoặc âm thanh. Các mô hình (models) Deep Learning có thể đạt được độ chính xác cao, đôi khi còn hơn cả con người. Các mô hình được huấn luyện bởi việc sử dụng một tập bao gồm bộ dữ liệu

được gán nhãn và các kiến trúc mạng neural gồm nhiều lớp (layer).



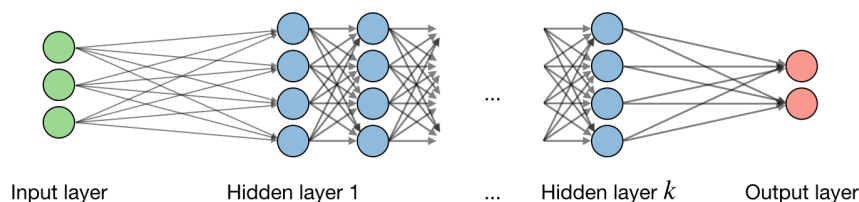
Hình 2.1: Mối quan hệ giữa AI, Machine Learning và Deep Learning. (Nguồn: *What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?*)

**Vậy điều gì mang đến sự thành công của deep learning?** Rất nhiều những ý tưởng cơ bản của deep learning được đặt nền móng từ những năm 80-90 của thế kỷ trước, tuy nhiên deep learning chỉ đột phá trong khoảng từ năm 2012. Vì sao? Có thể kể đến một vài nhân tố dẫn đến sự bùng nổ này:

- Sự ra đời của các bộ dữ liệu lớn được gán nhãn.
- Khả năng tính toán song song tốc độ cao của GPU.
- Sự cải tiến của các kiến trúc: GoogLeNet, VGG, ResNet, ... và các kỹ thuật transfer learning, fine tuning.
- Nhiều thư viện mới hỗ trợ việc huấn luyện deep network với GPU: Theano, Caffe, TensorFlow, PyTorch, Keras,...

## 2.1 Neural Network - Mạng Neural

Tổng quan kiến trúc một mạng Neural như sau



Hình 2.2: Ví dụ một mạng Neural có  $k$  tầng ẩn (Nguồn: CS229)

Với  $i$  là lớp thứ  $i$  của mạng,  $j$  là đơn vị ẩn thứ  $j$  của lớp, ta có:

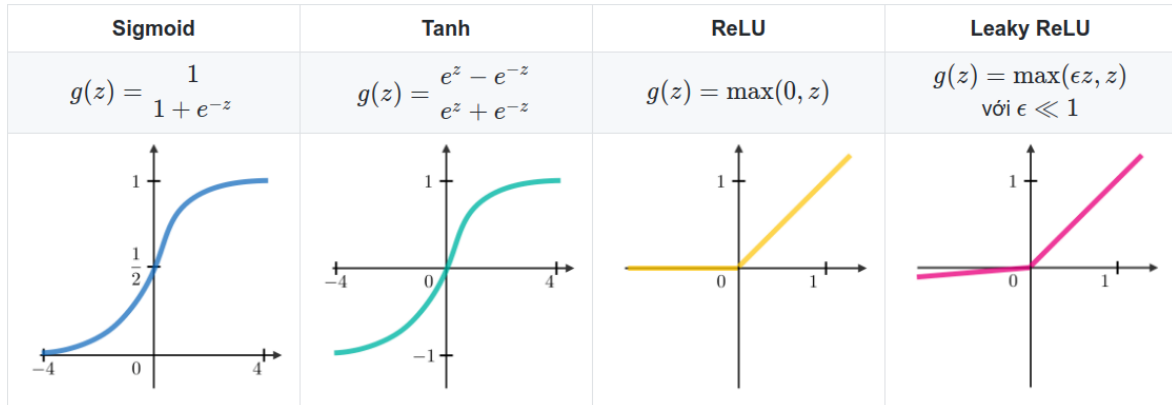
$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

trong đó:  $w$  là weight,  $b$  là bias,  $z$  là đầu ra.

**Hàm kích hoạt (Activation function):** Bản chất của công thức trên là một tổ hợp tuyến tính giữa các giá trị input  $x$  và bộ trọng số  $w$ , do đó, khi áp dụng chúng với các dữ liệu mà có dạng tuyến tính, tức là dữ liệu mà ta có thể kẻ một đường thẳng để phân cách giữa chúng thì công thức tổ hợp trên đã đủ để giúp cho mô hình máy học có thể hoạt động tốt, chúng ta không cần tới hàm kích hoạt (activation function). Nhưng với các dữ liệu không có dạng tuyến tính, ta không thể kẻ một đường thẳng tuyến tính mà phân tách 2 dữ liệu ra được, và câu hỏi đặt ra là làm thế nào với một công thức tổ hợp tuyến tính như ban đầu mà dùng để phân lớp dữ liệu phi tuyến tính được. **Hàm kích hoạt** được tạo ra để làm điều này, hàm kích hoạt đóng vai trò như một người trung gian có nhiệm vụ chuyển đổi, nén hoặc chế biến output  $z$  từ tuyến tính trở thành phi tuyến tính.

**Hàm mất mát (Loss function):** Hàm mất mát trả về một số thực không âm thể hiện sự chênh lệch giữa hai đại lượng:  $y_{pred}$  là giá trị được dự đoán và  $y_{true}$  là giá trị thực. Trong trường hợp lý tưởng,  $y_{pred} = y_{true}$ , hàm mất mát sẽ có giá trị bằng 0. Hàm loss được sử dụng phổ biến trong các mô hình Deep learning hiện nay là Cross-entropy cùng các biến thể cải tiến của nó (Weighted cross entropy, Focal loss...). Cross-entropy





Hình 2.3: Một số hàm kích hoạt thường dùng (Nguồn: CS229)

loss  $L(z, y)$  được định nghĩa như sau:

$$L(z, y) = -[y \log z + (1 - y) \log(1 - z)]$$

**Optimizer và Learning rate:** Sau khi tính giá trị hàm loss, việc cần làm là tối ưu (cực tiểu hóa) hàm loss và update bộ trọng số  $\{w\}$  mới. Learning rate, thường được ký hiệu là  $\alpha$  hoặc  $\eta$ , thể hiện cho tốc độ học hay tốc độ update trọng số. Learning rate có thể là cố định hoặc được thay đổi tùy biến trong quá trình học.

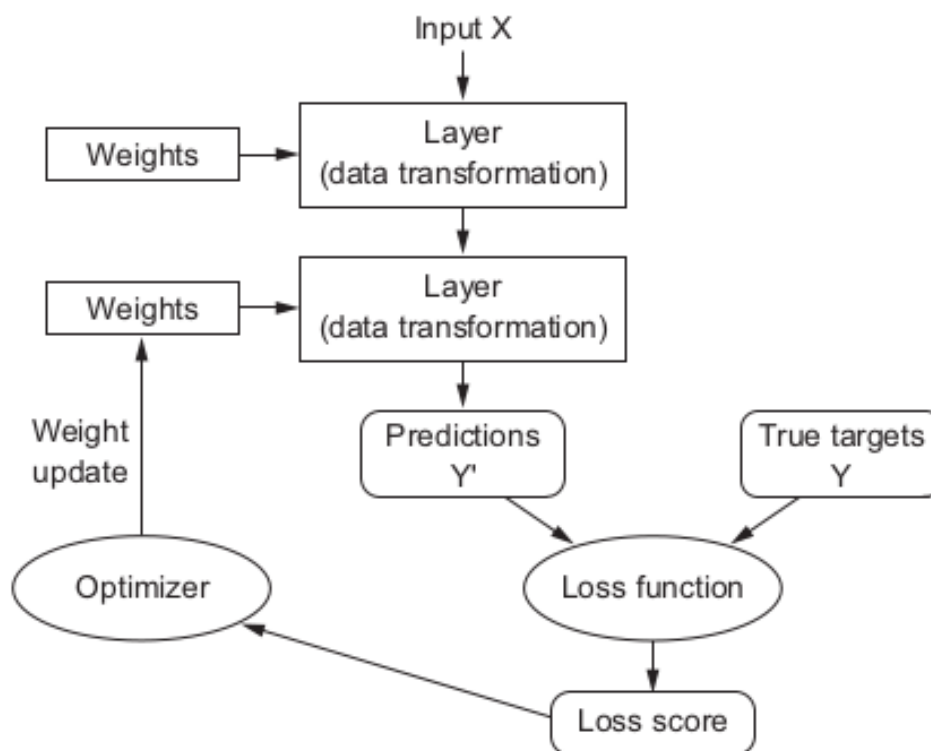
**Lan truyền ngược (Backpropagation):** Lan truyền ngược là phương thức dùng để cập nhật trọng số trong mạng neural bằng cách tính toán đầu ra thực sự và đầu ra mong muốn. Đạo hàm theo trọng số  $w$  được tính bằng cách sử dụng quy tắc chuỗi (chain rule) dưới đây:

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Như kết quả, trọng số được cập nhật như sau:

$$w := w - \eta \frac{\partial L(z, y)}{\partial w}$$

Tổng kết lại, ta có sơ đồ quá trình học của một mạng neural cơ bản như sau



Hình 2.4: Mối quan hệ giữa network, layers, loss function và optimizer (Nguồn: Deep Learning with Python - Francois Chollet)

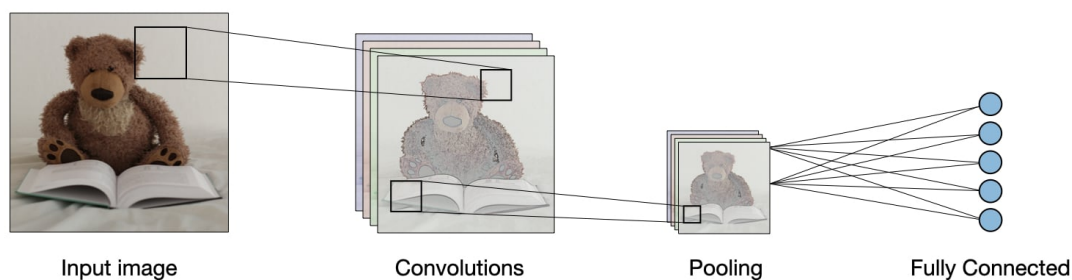
## 2.2 Mạng Neural tích chập (Convolutional Neural Network - CNN)

Mạng Neural tích chập là một trong những mô hình Deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong lĩnh vực Computer Vision. CNNs được dùng trong nhiều bài toán như nhận dạng ảnh, phân tích video, ảnh MRI, hoặc có thể cho cả các bài của lĩnh vực xử lý ngôn ngữ tự nhiên, và hầu hết đều giải quyết tốt các bài toán này.

Mạng neural tích chập, còn được biết đến với tên CNNs, là một dạng mạng neural được cấu thành bởi các layer sau:

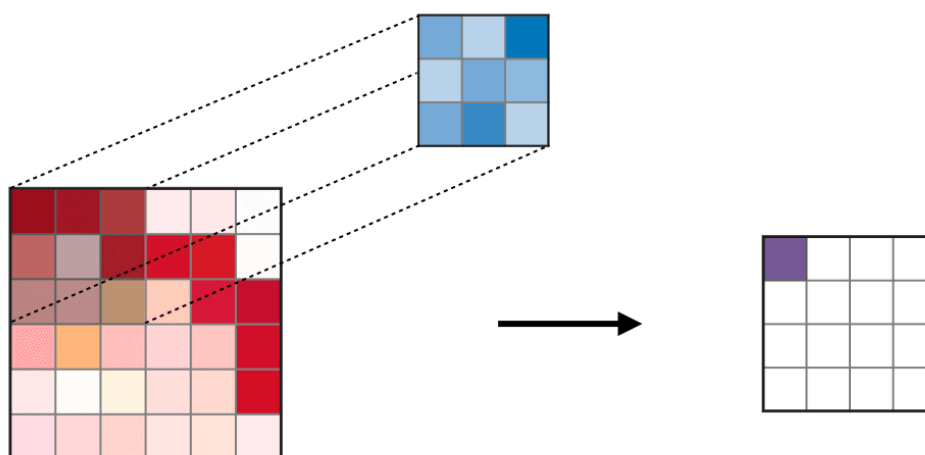
**Lớp tích chập (Convolution layer):** Tầng tích chập (CONV) sử dụng các bộ lọc (filters) để thực hiện phép tích chập khi đưa chúng đi qua input  $I$  theo các chiều của

## 2.2. MẠNG NEURAL TÍCH CHẬP (CONVOLUTIONAL NEURAL NETWORK - CNN) CHƯƠNG 2. LÝ THUYẾT HỌC SÂU VÀ BÀI TOÁN NHẬN DIỆN VẬT THỂ



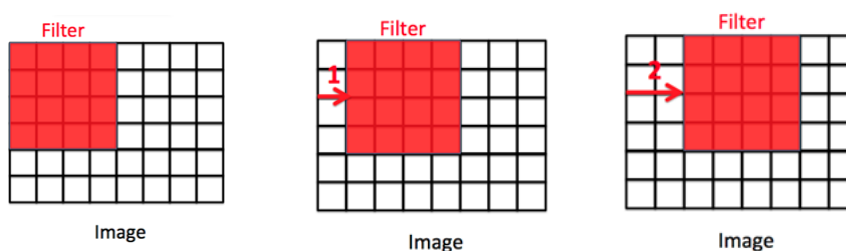
Hình 2.5: Ví dụ về một CNN (Nguồn: CS230)

nó. Các *hyperparameters* của filter bao gồm kích thước  $F$ , độ trượt (stride)  $S$ . Kết quả đầu ra của lớp này được gọi là feature map.



Hình 2.6: Mô tả hoạt động của CONV (Nguồn: CS230)

**Stride** là số lượng pixel dịch chuyển trên ma trận đầu vào hay Stride dùng để dịch chuyển filter theo mỗi bước xác định.



Ví dụ về stride = 1 và stride bằng 2

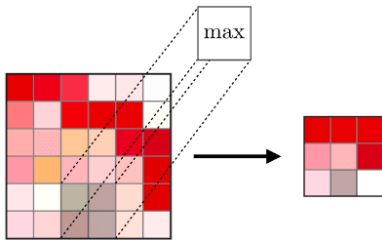
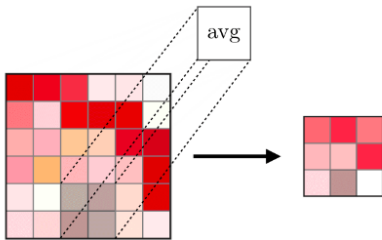
**Padding:** Khi áp dụng phép CONV thì ma trận đầu vào sẽ có nhỏ dần đi, do đó số layer của mô hình CNN sẽ bị giới hạn, và không thể xây dựng mô hình mong muốn.

## 2.2. MẠNG NEURAL TÍCH CHẬP (CONVOLUTIONAL NEURAL NETWORK - CNN) HƯỚNG 2. LÝ THUYẾT HỌC SÂU VÀ BÀI TOÁN NHẬN DIỆN VẬT THỂ

Để giải quyết tình trạng này, ta cần "bọc" bên ngoài ma trận đầu vào để đảm bảo kích thước đầu ra sau mỗi tầng convolution là không đổi. Do đó có thể xây dựng được mô hình với số tầng convolution lớn tùy ý. Một cách đơn giản và phổ biến nhất để padding là sử dụng hàng số 0, ngoài ra có một số phương pháp khác như reflection padding hay là symmetric padding.

**Lớp Pooling:** 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.

**Spatial pooling** được gọi là lấy mẫu con làm giảm chiều của mỗi map nhưng vẫn giữ được thông tin quan trọng. Spatial pooling có thể có nhiều loại khác nhau như Max Pooling và Average Pooling.

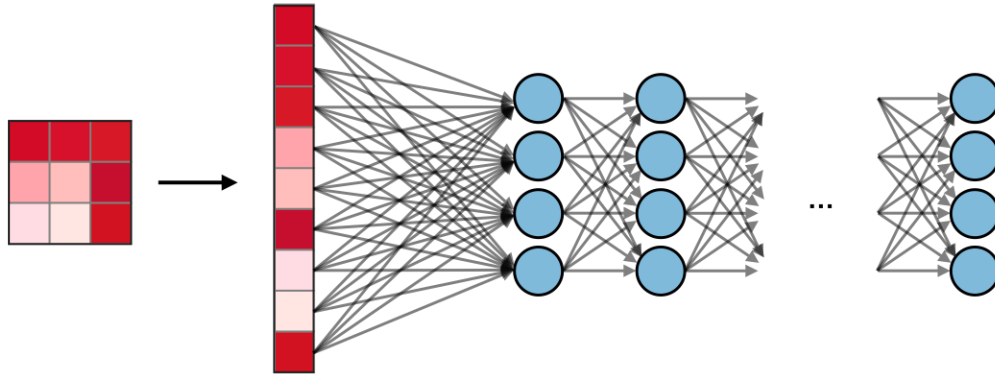
Kiểu	Max pooling	Average pooling
Chức năng	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng
Minh họa		
Nhận xét	<ul style="list-style-type: none"> <li>• Bảo toàn các đặc trưng đã phát hiện</li> <li>• Được sử dụng thường xuyên</li> </ul>	<ul style="list-style-type: none"> <li>• Giảm kích thước feature map</li> <li>• Được sử dụng trong mạng LeNet</li> </ul>

Hình 2.7: Hai kiểu pooling phổ biến (Nguồn: CS230)

**Fully Connected (FC):** Lớp Fully Connected nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neural. Trong mô hình mạng CNNs, các lớp FC thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

**Các hàm kích hoạt thường gặp:**

- **Rectified Linear Unit (ReLU):** ReLU là một hàm kích hoạt  $g$  được sử dụng trên tất cả các thành phần. Mục đích của nó là tăng tính phi tuyến tính cho



Hình 2.8: Fully Connected layer (Nguồn: CS230)

mạng. Những biến thể khác của ReLU được tổng hợp ở bảng dưới

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ với $\alpha \ll 1$

Hình 2.9: ReLU và biến thể (Nguồn: CS230)

- **Softmax:** Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị  $x \in \mathbb{R}^n$  và cho ra là một vector gồm các xác suất  $p \in \mathbb{R}^n$  thông qua một hàm softmax ở layer cuối.

$$p = (p_1, \dots, p_n)^T \text{ trong đó } p_j = \frac{e^{x_j}}{\sum_{j=1}^n e^{x_j}}$$

## 2.3 Bài toán nhận diện vật thể (Object Detection)

Các hình ảnh trong cuộc sống bình thường thì không chỉ chứa 1 đối tượng mà thường bao gồm rất nhiều các đối tượng. Ta quan tâm đến vị trí của từng đối tượng trong ảnh. Bài toán như vậy được gọi là: object detection.



Hình 2.10: Ví dụ về đầu ra của bài toán nhận diện vật thể

Bài toán object detection có input là ảnh màu và output là vị trí của các đối tượng trong ảnh. Ta thấy nó bao gồm 2 bài toán nhỏ:

- Xác định các bounding box (hình chữ nhật) quanh đối tượng.
- Với mỗi bounding box thì cần phân loại xem đây là đối tượng gì (người, mèo, ô tô,...) với bao nhiêu phần trăm chắc chắn.

Object Detection là 1 bài toán đã đạt được rất nhiều các thành tựu trong những năm gần đây, cả phần ứng dụng và mô hình thuật toán. Điển hình là các phương pháp Object Detection sử dụng Deep Learning đã đạt được các bước cải thiện vượt trội so với các phương pháp xử lý ảnh thông thường khác.

Có hai loại bài toán Object detection: two-stage object detection và one-stage object detection.

**Two-stage object detection:** Diễn hình họ các thuật toán R-CNN. Việc gọi là two-stage là do cách model xử lý để lấy ra được các vùng có khả năng chứa vật thể từ bức ảnh. Ví dụ, với Faster-RCNN thì trong stage-1, ảnh sẽ được đưa ra 1 sub-network gọi là RPN (Region Proposal Network) với nhiệm vụ extract các vùng trên ảnh có khả năng chứa đối tượng dựa vào các anchor. Sau khi đã thu được các vùng đặc trưng từ RPN, model Faster-RCNN sẽ thực hiện tiếp việc phân loại đối tượng và xác định vị trí nhờ vào việc chia làm 2 nhánh tại phần cuối của mô hình (Object classification & Bounding box regression).

**One-stage Object Detection:** Các thuật toán diễn hình như: SSD, YOLO, RetinaNet. Gọi là one-stage vì trong việc thiết kế model hoàn toàn không có phần trích chọn các vùng đặc trưng (các vùng có khả năng chứa đối tượng) như RPN của Faster-RCNN. Các mô hình one-stage object detection coi phần việc phát hiện đối tượng (object localization) như một bài toán regression (với 4 tọa độ offset, ví dụ x, y, w, h) và cũng dựa trên các box được định nghĩa sẵn gọi là anchor để làm việc đó. Các mô hình dạng này thường nhanh hơn tuy nhiên "độ chính xác" của model thường kém hơn so với two-stage object detection. Tuy nhiên, một số mô hình one-stage vẫn tỏ ra vượt trội hơn một chút so với two-stage như Retina-Net với việc thiết kế mạng theo FPN (Feature Pyramid Network) và Focal Loss.

## 2.3.1 Mô hình Faster R-CNN

### Regions with CNN features (R-CNN)

R-CNN được giới thiệu lần đầu vào 2014 bởi Ross Girshick và các cộng sự ở UC Berkeley một trong những trung tâm nghiên cứu AI hàng đầu thế giới trong bài báo *Rich feature hierarchies for accurate object detection and semantic segmentation*. RCNN có thể là xem một trong những ứng dụng nền móng đầu tiên của CNN đối với bài toán định vị, phát hiện và phân đoạn đối tượng.

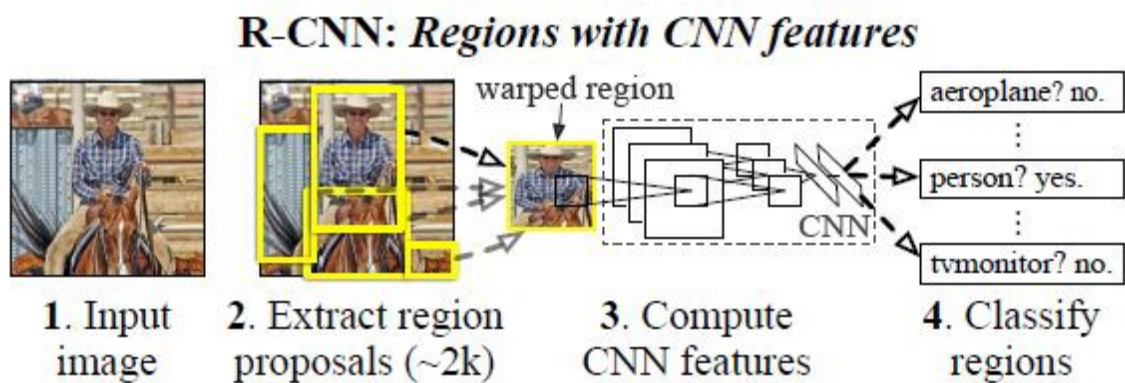
Kiến trúc của R-CNN gồm 3 thành phần đó là:

- Vùng đề xuất hình ảnh (Region proposal): Có tác dụng tạo và trích xuất các vùng đề xuất chứa vật thể được bao bởi các bounding box.

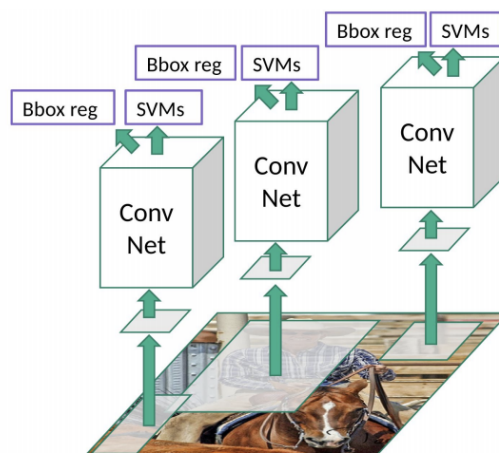


### 2.3. CHUỖ ĐÓNG NHẬN THUẬN TẠO CHỈ SỐ (OBB) ĐẾN NHẬN DIỆN VẬT THỂ

- Trích lọc đặc trưng (Feature Extractor): Trích xuất các đặc trưng giúp nhận diện hình ảnh từ các region proposal thông qua các CNN.
- Phân loại (classifier): Dựa vào input là các features ở phần trước để phân loại hình ảnh chứa trong region proposal về đúng nhãn.



Hình 2.11: Sơ đồ xử lý trong mô hình mạng R-CNN (Nguồn: *Medium*)



Hình 2.12: R-CNN (Nguồn: *Medium*)

Để vượt qua vấn đề chọn một số lượng lớn các region, Ross Girshick đề xuất một phương pháp gọi là selective search để chỉ trích xuất 2000 regions từ hình ảnh và gọi chúng là vùng đề xuất (region proposals), tức vùng có khả năng chứa đối tượng. Do đó, thay vì cố gắng phân loại một số lượng lớn các region, ta chỉ cần làm việc với 2000 regions. 2000 proposal regions này được uốn cong thành một hình vuông và được đưa



### 2.3. THUẬT TOÁN NHẬN THỨC TÀI VẬT (OBJECT DETECTION) DIỆN VẬT THỂ

vào một mạng CNN tạo ra một vector đặc trưng 4096 chiều làm đầu ra. CNN hoạt động như một công cụ trích xuất đặc trưng và dense layer đầu ra bao gồm các đặc trưng được trích xuất từ các ảnh và các đặc trưng được đưa vào một SVM để phân loại sự hiện diện của đối tượng trong proposal region được lấy ra đó. Ngoài việc dự đoán sự hiện diện của một đối tượng trong các proposal region, thuật toán cũng dự đoán bốn giá trị là độ lệch (offset values) để tăng độ chính xác của bounding box, Ví dụ với một region proposal cho trước, thuật toán sẽ dự đoán sự hiện diện của người nhưng khuôn mặt người đó trong region proposal đó có thể bị cắt đi một nửa. Do đó, các giá trị bù giúp điều chỉnh bounding box của region proposal.

#### **Nhược điểm của R-CNN:**

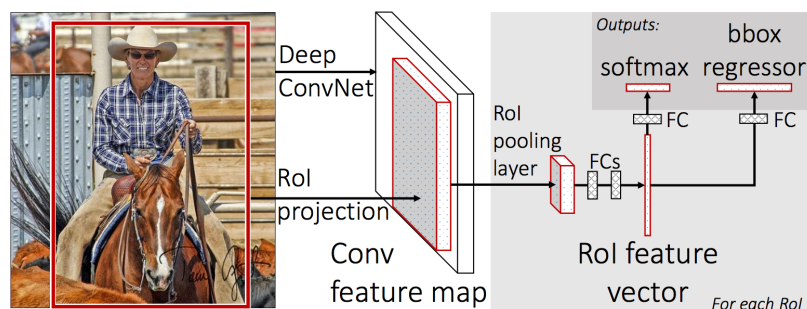
- Vẫn tốn lượng lớn thời gian để luyện mạng vì phải phân loại 2000 region proposals mỗi ảnh.
- Không thể chạy với thời gian thực do nó tốn thời gian 47 giây cho mỗi ảnh
- Thuật toán selective search là một thuật toán cố định. Do đó, không có việc học nào đang diễn ra ở giai đoạn đó, Điều này có thể dẫn đến việc tạo ra các region proposal tồi.

#### **Fast R-CNN**

Dựa trên thành công của R-CNN, Ross Girshick (lúc này đã chuyển sang Microsoft Research) đề xuất một mở rộng để giải quyết vấn đề của R-CNN trong một bài báo vào năm 2015 với tiêu đề rất ngắn gọn Fast R-CNN.

Tương tự như R-CNN thì Fast R-CNN vẫn dùng selective search để lấy ra các region proposal. Tuy nhiên là nó không tách 2000 region proposal ra khỏi ảnh và thực hiện bài toán image classification cho mỗi ảnh. Fast R-CNN cho cả bức ảnh vào ConvNet (một vài convolutional layer + max pooling layer) để tạo ra convolutional feature map. Sau đó các vùng region proposal được lấy ra tương ứng từ convolutional feature map. Tiếp đó được Flatten và thêm 2 Fully connected layer (FCs) để dự đoán lớp của region proposal và offset values của bounding box.

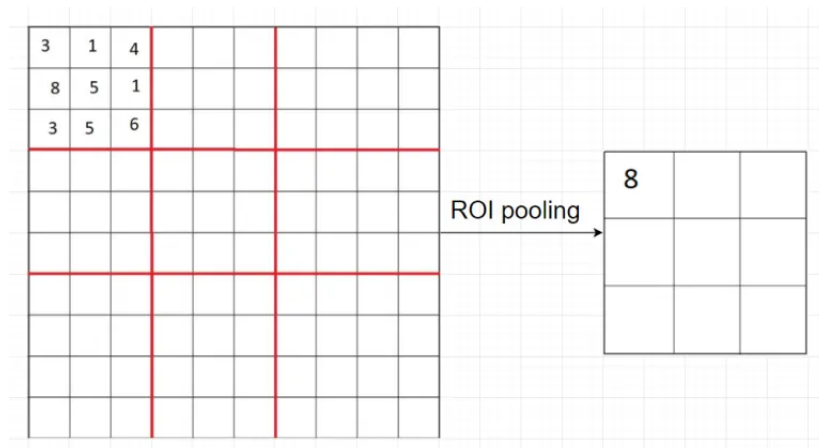
### 2.3. CHƯƠNG 2. NHẬN THỨC VẬT THỂ SẴN (OBJECT DETECTION) DIỆN VẬT THỂ



Hình 2.13: Fast R-CNN (Nguồn: *Medium*)

Tuy nhiên là kích thước của các region proposal khác nhau nên khi Flatten sẽ ra các vector có kích thước khác nhau nên không thể áp dụng neural network được. Với R-CNN, nó đã resize các region proposal về cùng kích thước trước khi dùng transfer learning. Tuy nhiên ở feature map ta không thể resize được, nên ta phải có cách gì đây để chuyển các region proposal trong feature map về cùng kích thước nên **Region of Interest (RoI) pooling** ra đời.

**Region of Interest (RoI) pooling:** RoI pooling là một dạng của pooling layer. Điểm khác so với max pooling hay average pooling là bất kể kích thước của tensor input, RoI pooling luôn cho ra output có kích thước cố định được định nghĩa trước.

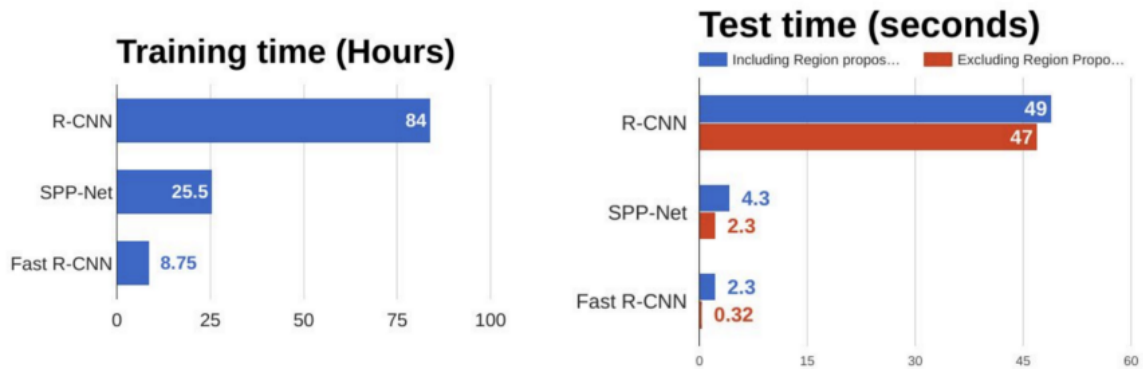


Hình 2.14: RoI pooling (Nguồn: *nttuan8.com*)

Fast R-CNN khác với R-CNN là nó thực hiện feature map với cả ảnh sau đó với lấy các region proposal ra từ feature map, còn R-CNN thực hiện tách các region proposal ra rồi mới thực hiện CNN trên từng region proposal. Do đó Fast R-CNN nhanh hơn

### 2.3. CHẤU ĐOÁN NHẬN THUYẾT VẬT THỂ (OBJECT DETECTION) DIỆN VẬT THỂ

đáng kể nhờ tối ưu việc tính toán bằng Vectorization.



Hình 2.15: So sánh một số thuật toán object detection (Nguồn: *towardsdatascience.com*)

Tuy nhiên nhìn hình trên ở phần test time với mục Fast R-CNN thì thời gian tính region proposal rất lâu và làm chậm thuật toán. Do đó cần thay thế thuật toán selective search và việc dùng Deep learning để tạo ra region proposal được thực hiện với mô hình Faster R-CNN.

### Faster R-CNN

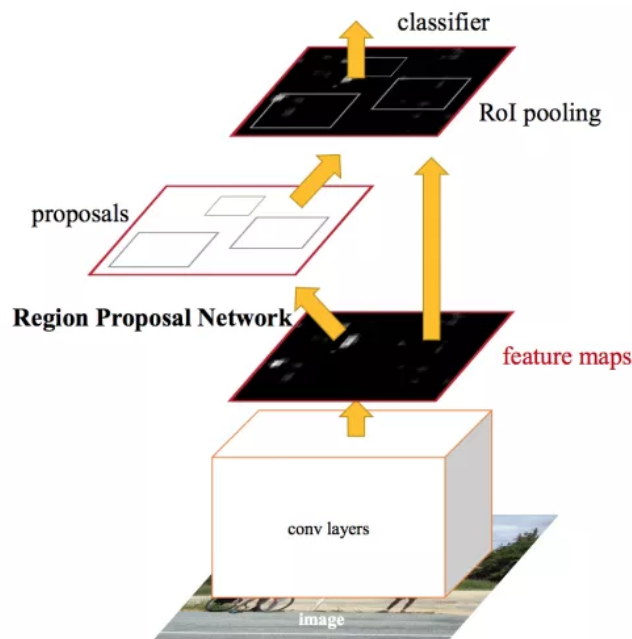
Faster R-CNN không dùng thuật toán selective search để lấy ra các region proposal, mà nó thêm một mạng CNN mới gọi là Region Proposal Network (RPN) để tìm các region proposal.

Một Region Proposal Network nhận đầu vào là ảnh với kích thước bất kì và cho đầu ra là region proposal (tập vị trí của các hình chữ nhật có thể chứa vật thể), cùng với xác suất chứa vật thể của hình chữ nhật tương ứng.

**Region Proposal Network (RPN):** Quy trình tính toán của RPN được mô tả chi tiết dưới đây:

1. Dùng một lớp tích chập  $3 \times 3$  với padding bằng 1 để biến đổi đầu ra của CNN và đặt số kênh đầu ra bằng  $c$ . Bằng cách này phần tử trong feature map mà CNN trích xuất ra từ bức ảnh là một đặc trưng mới có độ dài bằng  $c$ .

### 2.3. CHUỖ ĐOÁN NHẬN THỨC VẬT THỂ (OBJECT DETECTION) DIỆN VẬT THỂ

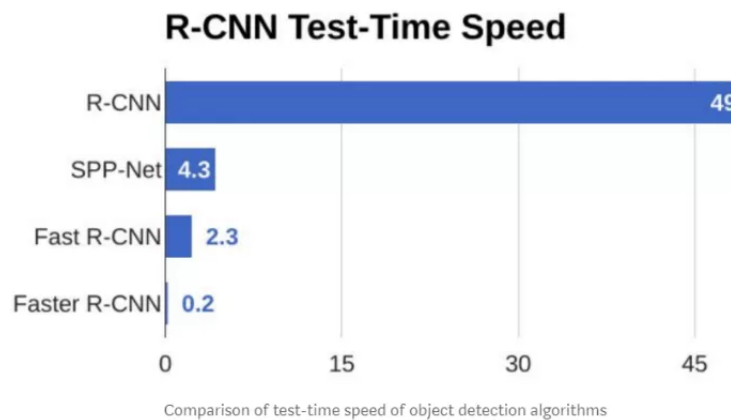


Hình 2.16: Mô hình Faster R-CNN (Nguồn: *arXiv:1506.01497*)

2. Lấy mỗi phần tử trong feature map làm tâm để tạo ra nhiều anchor box có kích thước và tỷ lệ khác nhau, sau đó gán nhãn cho chúng.
3. Lấy những đặc trưng của các phần tử có độ dài  $c$  ở tâm của anchor box để phân loại nhị phân (là vật thể hay là nền) và dự đoán bounding box tương ứng cho các anchor box.
4. Sau đó, sử dụng non-maximum suppression để loại bỏ các bounding box có kết quả giống nhau của hạng mục “vật thể”. Cuối cùng, ta xuất ra các bounding box dự đoán là các proposal region rồi đưa vào lớp RoI pooling.

Vì là một phần của mô hình Faster R-CNN, nên RPN được huấn luyện cùng với phần còn lại trong mô hình. Ngoài ra, trong đối tượng Faster R-CNN còn chứa các hàm dự đoán hạng mục và bounding box trong bài toán phát hiện vật thể, cũng như các hàm dự đoán hạng mục nhị phân và bounding box cho các anchor box trong RPN. Sau cùng, RPN có thể học được cách sinh ra những proposal region có chất lượng cao, giảm đi số lượng proposal region trong khi vẫn giữ được độ chính xác khi phát hiện vật thể.

### 2.3. CHƯƠNG TRÌNH NHẬN THỨC TÀI VẬT CHẤT (OBJECT DETECTION) DIỆN VẬT THỂ



Hình 2.17: Faster R-CNN nhanh hơn hẳn các dòng R-CNN trước đó, vì vậy có thể dùng cho real-time object detection (Nguồn: *nttuan8.com*)

#### 2.3.2 Một số phương pháp đánh giá mô hình nhận diện vật thể

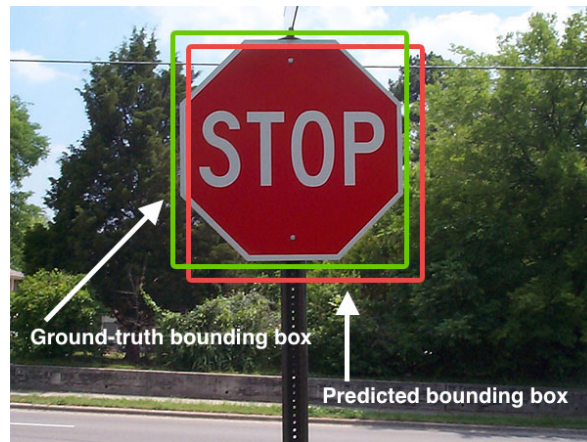
##### Intersection over Union (IoU)

Intersection over Union là chỉ số đánh giá được sử dụng để đo độ chính xác của Object detector trên tập dữ liệu cụ thể. IoU đơn giản chỉ là một chỉ số đánh giá. Mọi thuật toán có khả năng predict ra các bounding box làm output đều có thể được đánh giá thông qua IoU.

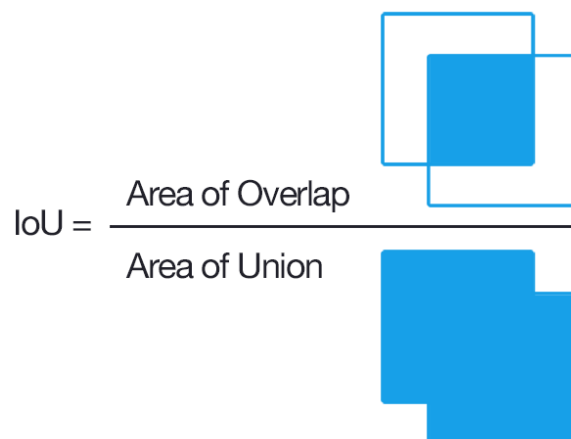
Để áp dụng được IoU để đánh giá một mô hình nhận diện vật thể bất kì ta cần:

- Những ground-truth bounding box (bounding box đúng của đối tượng, ví dụ như bounding box của đối tượng được khoanh vùng và gán nhãn bằng tay sử dụng trong tập test.)
- Những bounding box dự đoán được model sinh ra.

### 2.3. CHẤU ĐOÁN NHẬN THUYẾT VÀO CẢ (OBJECT DETECTION) DIỆN VẬT THỂ



Hình 2.18: Một ví dụ về phát hiện biển báo Stop từ hình ảnh. (Nguồn: *pyimagesearch.com*)



Hình 2.19: Tính toán Intersection over Union. (Nguồn: *pyimagesearch.com*)



Hình 2.20: Một ví dụ về tính toán IoU cho những bounding box khác nhau. (Nguồn: *pyimagesearch.com*)

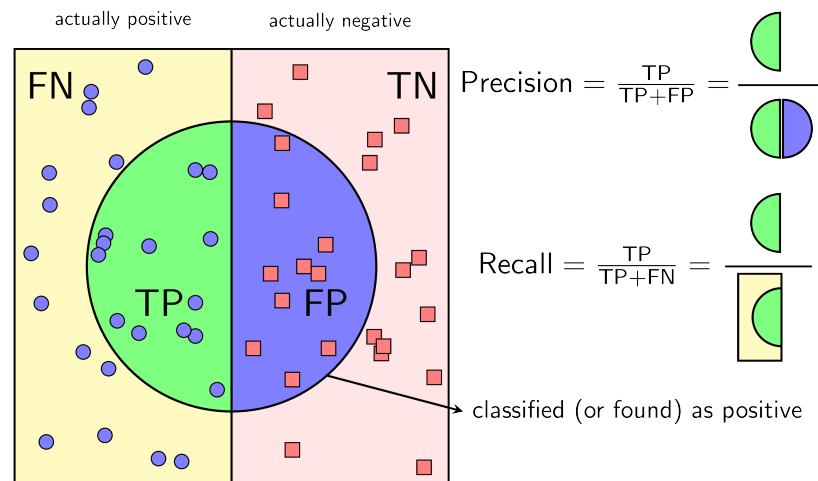
## Average Precision

AP (Average Precision) là một metric phổ biến trong việc đánh giá độ chính xác của các mô hình nhận diện vật thể như aster R-CNN, SSD,...

### Precision – Recall

Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đo hiệu quả thường được sử dụng là Precision-Recall.

Trước hết xét bài toán phân loại nhị phân. Ta cũng coi một trong hai lớp là positive, lớp còn lại là negative.



Hình 2.21: Cách tính Precision và Recall. (Nguồn: *Machine Learning cơ bản*)

**Precision** cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. **Recall** cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các điểm thực sự positive là thấp.

### Precision-Recall curve và Average precision

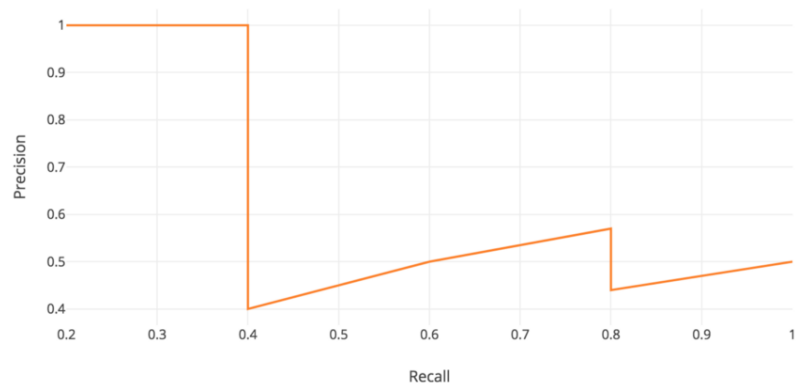
Ta có thể đánh giá mô hình dựa trên việc thay đổi một ngưỡng và quan sát giá trị của Precision và Recall.

**Average Precision:** Ta xét một ví dụ, với bộ dữ liệu có 5 quả táo, mô hình lần lượt đưa ra 10 dự đoán, ta chọn ngưỡng cho dự đoán đúng là  $\text{IoU} > 0.5$ , dự đoán được xếp giảm dần theo sự 'tự tin' của dự đoán. Số liệu được thể hiện trong bảng sau:

### 2.3. CHUỖ ĐÓNG NHẬN THUẬN TẠO CHỈ SỐ (OBB) ĐẾN NHẬN DIỆN VẬT THỂ

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Biểu diễn các điểm precision-recall ta được một đường zig-zag sau



Hình 2.22: Precision-recall curve (Nguồn: *Medium*)

Định nghĩa tổng quát cho Average Precision (AP) là diện tích phía dưới precision-recall curve

$$AP = \int_0^1 p(r)dr$$

Precision và recall luôn nằm trong đoạn  $[0; 1]$  do đó AP cũng nằm trong đoạn  $[0; 1]$ . Mean average precision (mAP) là trung bình của AP. Trong một số trường hợp, ta tính AP cho mỗi class và lấy trung bình của chúng, một số khác thì lại giống nhau. Ví dụ theo COCO, không có sự khác biệt giữa AP và mAP.



## 2.3. CHẤU ĐOÁN NHẬN THUYẾT VÀO CẢ SẴ (CÓ BÀI ĐỀ ÁN NHẬN) DIỆN VẬT THỂ

<b>Average Precision (AP):</b>	
AP	% AP at IoU=.50:.95 (primary challenge metric)
AP <sup>IoU=.50</sup>	% AP at IoU=.50 (PASCAL VOC metric)
AP <sup>IoU=.75</sup>	% AP at IoU=.75 (strict metric)
<b>AP Across Scales:</b>	
AP <sup>small</sup>	% AP for small objects: area < 32 <sup>2</sup>
AP <sup>medium</sup>	% AP for medium objects: 32 <sup>2</sup> < area < 96 <sup>2</sup>
AP <sup>large</sup>	% AP for large objects: area > 96 <sup>2</sup>
<b>Average Recall (AR):</b>	
AR <sup>max=1</sup>	% AR given 1 detection per image
AR <sup>max=10</sup>	% AR given 10 detections per image
AR <sup>max=100</sup>	% AR given 100 detections per image
<b>AR Across Scales:</b>	
AR <sup>small</sup>	% AR for small objects: area < 32 <sup>2</sup>
AR <sup>medium</sup>	% AR for medium objects: 32 <sup>2</sup> < area < 96 <sup>2</sup>
AR <sup>large</sup>	% AR for large objects: area > 96 <sup>2</sup>

Hình 2.23: Một số metric được sử dụng để đánh giá kết quả trên bộ dữ liệu COCO (Nguồn: *cocodataset.org*)

### 2.3.3 PyTorch và Faster R-CNN ResNet-50 FPN

#### Sơ lược về Transfer Learning

Những năm gần đây, Deep Learning phát triển cực nhanh dựa trên lượng dữ liệu training khổng lồ và khả năng tính toán ngày càng được cải tiến của các máy tính. Các kết quả cho bài toán phân loại ảnh ngày càng được nâng cao. Bộ cơ sở dữ liệu thường được dùng nhất là ImageNet với 1.2M ảnh cho 1000 classes khác nhau. Rất nhiều các mô hình Deep Learning đã giành chiến thắng trong các cuộc thi ILSVRC (ImageNet Large Scale Visual Recognition Challenge). Có thể kể ra một vài: AlexNet, ZFNet, GoogLeNet, ResNet, VGG.

Nhìn chung, các mô hình này đều bao gồm rất nhiều layers. Các layers phía trước thường là các Convolutional layers kết hợp với các nonlinear activation functions và pooling layers (và được gọi chung là ConvNet). Layer cuối cùng là một Fully Connected Layer và thường là một Softmax Regression (Xem Hình 1). Số lượng units ở layer cuối cùng bằng với số lượng classes (với ImageNet là 1000). Vì vậy output ở layer gần cuối cùng (second to last layer) có thể được coi là feature vectors và Softmax Regression chính là Classifier được sử dụng.

Chính nhờ việc features và classifier được trained cùng nhau qua deep networks khiến

## 2.3. CHẤU ĐOÁN NHẬN THỨC TÀI VẬT (CƠ BÀN ĐỂ CÁC BÀI TOÁN NHẬN DIỆN VẬT THỂ)

cho các mô hình này đạt kết quả tốt. Tuy nhiên, những mô hình này đều là các Deep Networks với rất nhiều layers. Việc training dựa trên 1.2M bức ảnh của ImageNet cũng tốn rất nhiều thời gian (2-3 tuần).

Với các bài toán dựa trên tập dữ liệu khác, rất ít khi người ta xây dựng và train lại toàn bộ Network từ đầu, bởi vì có rất ít các cơ sở dữ liệu có kích thước lớn. Thay vào đó, phương pháp thường được dùng là sử dụng các mô hình (nêu phía trên) đã được trained từ trước, và sử dụng một vài kỹ thuật khác để giải quyết bài toán. Phương pháp sử dụng các mô hình có sẵn như thế này được gọi là Transfer Learning.

Có 2 loại transfer learning:

- **Feature extractor:** Sau khi lấy ra các đặc điểm của ảnh bằng việc sử dụng ConvNet của pre-trained model, thì ta sẽ dùng linear classifier (linear SVM, softmax classifier,...) để phân loại ảnh.
- **Fine tuning:** Sau khi lấy ra các đặc điểm của ảnh bằng việc sử dụng ConvNet của pre-trained model, thì ta sẽ coi đây là input của 1 CNN mới bằng cách thêm các ConvNet và Fully Connected layer.

### 2.3.4 PyTorch

PyTorch là một thư viện machine learning mã nguồn mở dựa trên Torch, được sử dụng cho lĩnh vực Thị giác máy tính (Computer Vision) và xử lý ngôn ngữ tự nhiên (Natural language processing), được phát triển bởi Phòng nghiên cứu AI của Facebook (FAIR).

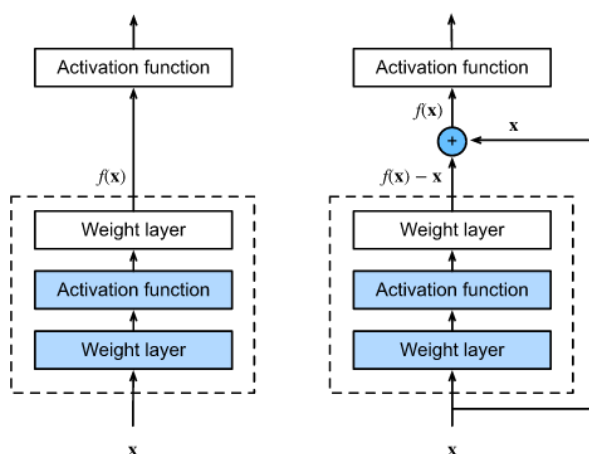
Pytorch tập trung vào 2 khả năng chính:

- Một sự thay thế cho bộ thư viện numpy để tận dụng sức mạnh tính toán của GPU.
- Một platform Deep learning phục vụ trong nghiên cứu, mang lại sự linh hoạt và tốc độ.

Cấu trúc dữ liệu cốt lõi được sử dụng trong PyTorch là **Tensor**.

### 2.3.5 Faster R-CNN ResNet-50 FPN

**Residual Network (ResNet):** Một vấn đề phổ biến của Deep learning là Vanishing Gradients, tức là theo công thức tính đạo hàm bằng chain rule (trong lan truyền ngược, back propagation), nếu gradient của các lớp sau nhỏ thì gradient ở các lớp đầu sẽ gần bằng 0. Vì thế mà parameter của các lớp trước sẽ không được cập nhật nên các parameter này không đóng góp được gì trong việc đưa ra output. Vậy nên dù ta dùng nhiều lớp, thực chất số lớp hữu ích chỉ là một vài lớp cuối, điều đó làm giảm sự hiệu quả của mạng neural. ResNet ra đời để giải quyết vấn đề đó, giải pháp mà ResNet đưa ra là sử dụng kết nối "tắt" đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block.



Hình 2.24: Khối thường (trái) và khối residual (phải)

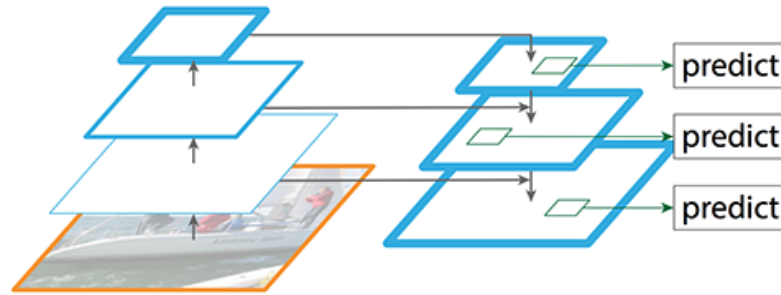
ResNet-50 là một mạng bao gồm 50 lớp residual.

**Feature Pyramid Networks:** Dò tìm các đối tượng có kích thước nhỏ là một vấn đề đáng được giải quyết để nâng cao độ chính xác. Và FPN là mô hình mạng được thiết kế ra dựa trên khái niệm pyramid để giải quyết vấn đề này.

Mô hình FPN kết hợp thông tin của mô hình theo hướng bottom-up kết hợp với top-down để dò tìm đối tượng (trong khi đó, các thuật toán khác chỉ thường sử dụng bottom-up). Khi chúng ta ở bottom và đi lên (up), độ phân giải sẽ giảm, nhưng giá trị ngữ nghĩa sẽ tăng lên.

Faster R-CNN ResNet-50 FPN là một mô hình Faster R-CNN với mạng backbone là

### 2.3. CHƯƠNG TRÌNH NHẬN THỨC VẬT THỂ (OBJECT DETECTION) DIỆN VẬT THỂ



Hình 2.25: FPN (Nguồn: *arXiv: 1612.03144*)

mạng ResNet-50 kết hợp với Feature Pyramid Networks. Trong Faster R-CNN, FPN sẽ tạo ra các feature map, sau đó, chúng ta sẽ rút trích các ROIs trên các feature map đó. Dựa trên kích thước của các ROI, chúng ta sẽ chọn feature map nào tốt nhất để tạo các feature patches (các hình chữ nhật nhỏ).

## Chương 3

# Ứng dụng học sâu vào bài toán đếm cây trên ảnh viễn thám

### 3.1 Giới thiệu bài toán

### 3.2 Mô hình hoá bài toán và thiết kế dữ liệu luyện

a) *Mô hình hóa bài toán*

**Input:**

Ảnh viễn thám được lưu ở định dạng .tif, có kích thước lớn (từ vài chục MB đến GB)

**Tiền xử lý:**

Ảnh đầu vào sẽ được chia nhỏ thành các ảnh có kích thước  $256 \times 256$  (bằng với kích thước trong tập ảnh training), với tỉ lệ *overlap* giữa các tấm ảnh là 0.5

**Output:**

Các file định dạng txt chứa các bounding box mà mô hình đoán đó là vật thể trong bức ảnh đó. Vì mục tiêu của bài toán là nhận dạng và đếm số cây trên một bức ảnh lớn, nên ta sẽ ghép các tấm ảnh nhỏ ở trên lại để đưa về hình ảnh gốc của chúng. Vì trong quá trình tiền xử lý, ta đã thiết lập  $overlap = 0.5$  nên việc

các bounding box của các bức ảnh gần nhau sẽ đè lên nhau, gây ra hiện tượng trùng lặp. Để giải quyết vấn đề này, các bounding box thu được trên toàn bộ các ảnh đầu tiên sẽ được đưa về tọa độ địa lý, sau đó sử dụng thuật toán NMS để loại bỏ các bounding box trùng nhau (nếu như giá trị IOU của hai bounding box này vượt quá một ngưỡng (trong mô hình này em lấy ngưỡng = 0.3))

b) a durian

### 3.3 Huấn luyện mạng neural

### 3.4 Xây dựng chương trình

## Chương 4

### Cài đặt chương trình và đánh giá kết quả

- 4.1 Môi trường cài đặt chương trình và các yêu cầu liên quan
- 4.2 Dữ liệu đầu vào
- 4.3 Kết quả huấn luyện
- 4.4 Đánh giá kết quả

# Tài liệu tham khảo

## Tài liệu tham khảo tiếng Việt

- [1.] Nguyễn Đức Nghĩa, Nguyễn Tô Thành, "Toán rời rạc", Nhà xuất bản Đại Học Quốc Gia Hà Nội, 1997, tr. 147-155.
- [2.] Tống Đình Quỳ, "Giáo trình xác suất thống kê", Nhà xuất bản Bách Khoa-Hà Nội, 2016.

## Tài liệu tham khảo tiếng Anh

- [1.] Martin Aigner, Günter M. Ziegler, "Proofs from THE BOOK", 6<sup>th</sup> Edition.
- [2.] Valle Martinez, Vicente, "Notes on the proof of the van der Waerden permanent conjecture"(2018).