# CHƯƠNG I. CƠ SỞ LÝ THUYẾT

## Phát hiện và theo dõi đối tượnng

### Phát hiện đối tượng: OBJECT DETECTION

Phát hiện đối tượng là làm nhiệm vụ xác định một hoặc nhiều đối tượng hiện diện trong cùng một hình ảnh. Việc phát hiện đối tượng bao gồm hai phần cụ thể là phân loại và xác định vị trí. Đây là một thách thức trong bài toán này vì phải phân loại đối tượng mà còn phải xác định chính xác vị trí tất cả các đối tượng có trong hình ảnh cùng với việc phát hiện sự hiện diện của chúng. Phát hiện đối tượng là một trong số ứng dụng quan trọng, bởi nhiều nhiệm vụ cần sự giám sát của con người có thể tự động hóa bằng cách phát hiện các đối tượng trong hình ảnh. Phát hiện đối tượng về cơ bản là phân loại và định vị các đối tượng, gắn nhãn cho chúng bằng các hộp hình chữ nhật để thể hiện điểm tin cậy của dự đoán. Mạng nơ-ron tích chập (CNN) kết hợp với sliding windows cửa sổ trượt để phân loại từng hình ảnh này có đối tượng hoặc không đối tượng, sau đó kết hợp các kết quả lại với nhau để thực hiện dự đoán.

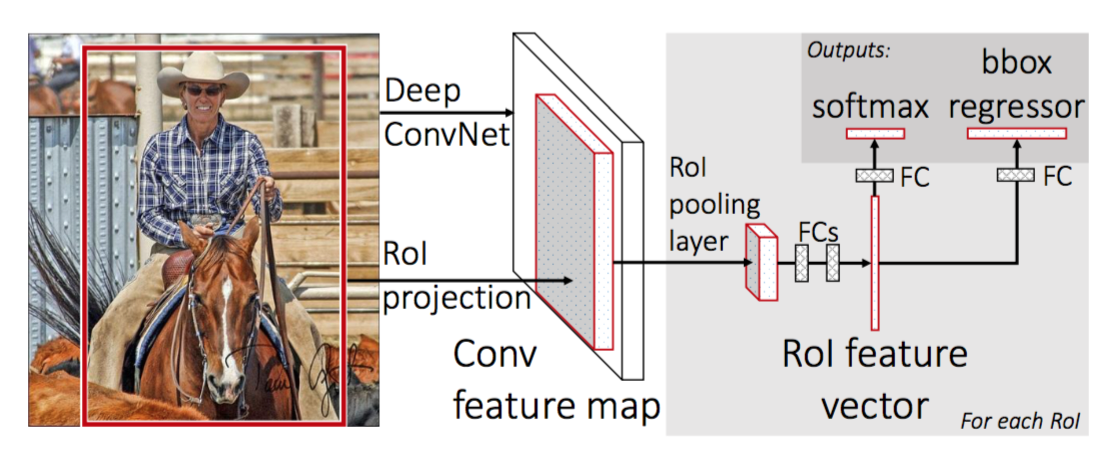
Trong những năm gần đây, rất nhiều mạng hiện đại phát hiện đối tượng đã được đề xuất bởi cộng đồng học sâu DL, như Faster R-CNN , YOLO , R-FCN và SSD . Mục tiêu chính của các thiết kế này là cải thiện độ chính xác phát hiện theo mAP và độ phức tạp trong tính toán của các mô hình giúp cải thiện thời gian xử lý, để có thể đạt được hiệu suất thời gian thực cho các nền tảng nhúng và di động. Các mô hình mạng phát hiện vật thể này có thể được chia thành hai loại dựa trên kiến trúc của chúng: 1) phương pháp tiếp cận một bước và 2) phương pháp hai bước (dựa trên khu vực – region based). Cách tiếp cận một bước có tốc độ chạy nhanh hơn và cho thấy hiệu quả bộ nhớ cao hơn trong khi cách tiếp cận hai bước tuy đạt được độ chính xác mAP tốt hơn nhưng lại tốn bộ nhớ hơn để chạy.

#### Mạng phát hiện đối tượng theo khu vực (mạng hai bước)

Mạng phát hiện đối tượng theo khu vực được chia làm hai bước. Bước thứ nhất là tạo các phân vùng trong ảnh mà khả năng đối tượng cần phát hiện sẽ nằm trong đó. Bước thứ hai sẽ thực hiện việc phát hiện đối tượng và phân loại đối tượng đó là vật gì dựa theo đầu vào phân vùng được tạo ở bước thứ nhất. Có thể kể đến một số mô hình nổi tiếng như R-CNN, Fast R-CNN, FPN và R-FCN.

* **Mạng Fast R-CNN**

Để tăng tốc cho R-CNN, Girshick đã cải thiện quá trình huấn luyện bằng các hợp nhất 3 mô hình độc lập thành một và được đào tạo chung và tăng khả năng tính toán. Mô hình này gọi là Fast-RCNN. Thay vì trích xuất đặc trưng bằng CNN cho mỗi vùng ảnh, mô hình tổng hợp chúng thành một CNN forward trên toàn bộ hình ảnh. Chính vì điều này giúp cho mạng Fast R-CNN giảm được cả thời gian chạy và khối lượng tính toán. Ngoài ra Fast R-CNN còn đưa ra phương pháp cho phép các đầu ra phân loại và hồi quy được huấn luyện đồng thời như trong hình 1.8.



Hình 1.7. Cấu trúc mạng Fast R-CNN

Mạng Fast-RCNN tối ưu tốc độ và tài nguyên hơn R-CNN. Hàm mất mát dễ huấn luyện hơn và tránh được vấn đề “Nổ” Gradient (Exploding Gradient). Nổ gradient là hiện tượng khi sai số gradient dần bị tích lũy và ngày càng tăng lên giá trị rất lớn khiến cho mạng học sâu trở nên bất ổn định. Không những thế giá trị của trọng số trở nên lớn hơn bất thường còn làm gia tăng các tham số tính toán, khiến cho tài nguyên bộ nhớ bị tràn.

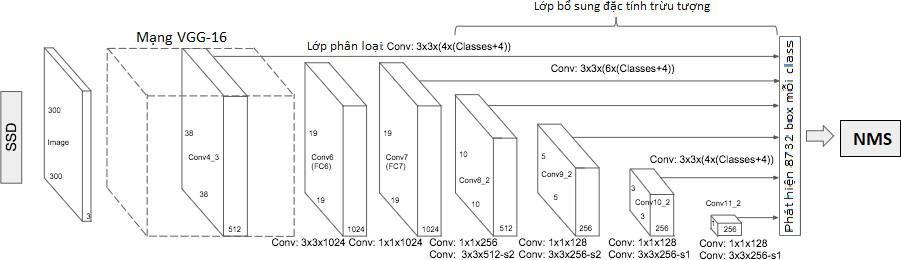
#### 1.1.1.2 Mạng phát hiện, nhận dạng một bước

Mạng phát hiện đối tượng một bước có kiến trúc giản đơn và thời gian xử lý của mạng nhanh hơn, với độ chính xác có thể chấp nhận được khi so sánh với các loại mạng phát hiện đối tượng hai bước dựa theo vùng. Các mạng phát hiện đối tượng một bước gần giống với mạng phát hiện đối tượng hai bước RPN, nhưng thay vì dự đoán vật hay không phải là vật, mạng phát hiện đối tượng một bước thì dự đoán phân loại vật thuộc lớp nào và cũng xác định vị trí tọa độ của vật trong ảnh. Một số mạng phát hiện đối tượng một bước hiện đại như **SSD** và **YOLO.**

* **Single Shot Multibox Detection (SSD)**

SSD [4] là mạng nhận dạng đối tượng có đầu ra là một tập hợp nhận dạng cùng kích thước mà kết quả có giá trị lớn nhất sẽ được lựa chọn, và giá trị đó được coi là giá trị tượng trưng cho độ tin tưởng đối tượng được nhận diện đó thuộc lớp vật nào. Cấu trúc của mạng SSD ở những lớp đầu sử dụng cấu trúc từ các mạng bài toán phân loại như CNN, VGG-16, ResNet để làm mạng cơ sở, hay còn được biết đến như phần xương sống cho mạng học sâu nhận dạng đối tượng. Ở lớp cuối của phần mạng cơ sở đó, lớp kết nối đầy đủ sẽ được bỏ đi, cuối cùng kết quả ở phần xương sống sẽ là đặc tính trừu tượng của dữ liệu ảnh đầu vào.

Trong tài liệu [4] tác giả đã sử dụng mạng VGG-16 với một số chỉnh sửa để cấu trúc hoạt động nhanh hơn mà không làm giảm đáng kể động chính xác sau đó. Sau đó họ thêm một tập hợp các lớp mạng tích chập ở cuối mạng cơ sở để giảm dần kích thước ảnh, đồng thời cho phép trích xuất thêm đặc điểm đặc trưng ở các tỷ lệ kích thước khác như như trong hình 1.9. Mỗi bản đồ trích xuất đặc trưng (feature map) có kích thước m \* n \* p.



Hình 1.8. Cấu trúc mạng SSD sử dụng mạng cơ sở VGG-16

Trong đó m \* n là số lượng ô đặc trưng trong bản đồ và p là độ sâu của véc-tơ đặc trưng cho mỗi ô. Đặc tính trừu tượng của mỗi lớp được sử dụng vào hai việc: thứ nhất là đưa vào lớp tiếp theo, hai là được trực tiếp sử dụng để dự đoán vật thể. Đối với mỗi ô, người ta có thể xuất số lượng đầu ra dự đoán cố định (ví dụ 4 đầu ra) liên quan đến vị trí của ô trong ảnh, cho phép các hộp giới hạn có kích thước và vị trí khác nhau. Ma trận điểm ảnh ở các lớp ban đầu có kích thước m\*n rất lớn sẽ được chia ra thành nhiều phần nhỏ để dự đoán vật thể nhỏ hơn. Những cải tiến chính của SSD đã vượt qua YOLO là nó có khả năng tích chập hoàn toàn, có phát hiện ở nhiều kích thước tỷ lệ khác nhau. Với mạng CNN, các lớp kết nối đầy đủ là cực kỳ tốn kém vì chúng đòi hỏi một số lượng lớn các kết nối như vậy làm chậm đáng kể việc tính toán.

* **YOLO – You Only Look Once**

YOLO là một phương pháp của Đại học Washington, giới thiệu một mạng lưới đơn sắc dự đoán các hộp giới hạn và phân loại của chúng trực tiếp từ một hình ảnh có kích thước đầy đủ. Mục tiêu của mạng là chỉ nhìn vào mọi phần của hình ảnh một lần, do đó có tên You Only Look Once – tạm dịch là **“Bạn chỉ nhìn một lần”**. Để nhận ra điều này, nhóm tác giả đã bắt đầu bằng việc chia hình ảnh thành một lưới 7x7 trong đó mỗi ô có trách nhiệm dự đoán. Đầu tiên, họ dự đoán một nhóm nhỏ các hộp giới hạn mà mỗi hộp có ô là trung tâm của chúng. Số lượng dự đoán cho mỗi ô là một hằng số nhỏ, trong bài báo được đặt giá trị bằng 2. Thứ hai, là nó sẽ dự đoán các giá trị độ tin cậy cho mỗi hộp giới hạn. Điều này phản ánh mức độ chắc chắn của mạng YOLO là hộp giới hạn đó bao phủ một số đối tượng.



Hình 1.9. Cách thức nhận dạng của YOLO

Nếu ô không chứa bất kỳ đối tượng nào, nó vẫn sẽ đề xuất cùng một bộ các hộp giới hạn, nhưng giá trị độ tin cậy sẽ thấp hơn so với các hộp giới hạn ở ô có chứa đối tượng. Bằng cách kết hợp các kết quả từ mỗi ô 7x7, chúng ta sẽ có được thứ gì đó trông rất giống với dự đoán của khung giới hạn được mô tả trong các phương thức R-CNN, nhưng ít trùng lặp hộp giới hạn với nhau hơn. So với R-CNN có khoảng 2000 hộp giới hạn có thể phân loại riêng lẻ, YOLO hiện có 7x7x2 = 98 (2 hộp dự đoán cho mỗi ô trong lưới) với các giá trị độ tin cậy tương ứng. Tiếp theo, mạng YOLO sẽ cố gắng phân loại từng hộp giới hạn (anchor box). Điều này không được thực hiện bằng cách chạy riêng cho từng hộp, mà chạy tính trên cho từng ô. Bởi vì mọi hộp giới hạn được đặt ở giữa một trong các ô, chúng ta có thể giả định rằng đối tượng sẽ nằm trong hộp giới hạn giống như đối tượng được xác định cho ô. Đầu ra cuối cùng sau đó là tất cả các hộp giới hạn với các giá trị độ tin cậy và xác suất cho từng lớp tương ứng. Để tìm các hộp chính xác và các lớp của chúng, YOLO đặt ngưỡng cho các giá trị độ tin cậy để loại bỏ tất cả các hộp xác suất hay độ tin cậy thấp hơn.

### Theo dõi đối tượng: Object Tracking

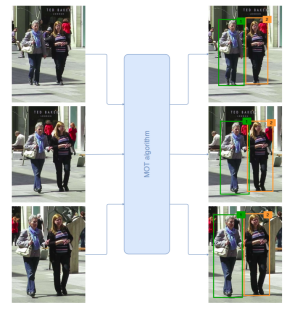
Object Tracking là bài toán theo dõi một hoặc nhiều đối tượng chuyển động theo thời gian trong một video. Hiểu một cách đơn giản nhất, nó là bài toán ở mức độ cao hơn so với object detection, khi đối tượng được xử lí không đơn giản là một hình ảnh mà là một chuỗi các hình ảnh hay video.

Object tracking và object detection có một mối quan hệ rất chặt chẽ. Object detection là việc mô hình xác định một hoặc nhiều đối tượng trong một khung hình nhất định còn object traking là theo dõi đối tượng trong toàn bộ video. Để theo dõi được một đối tượng trước hết mô hình cần phải phát hiện được đối tượng đó trong từng frame sau đó sẽ cho qua một thuật toán về tracking nên việc theo dõi đối tượng có chính xác hay không thì cần phải xem việc phát hiện đối tượng tốt đến đâu. Object detection sẽ phát hiện sự xuất hiện của những đối tượng trong hình ảnh tuy nhiên để xem trong video đó khi các đối tượng chuyển động thì cùng 1 đối tượng mô hình sẽ hiểu đó là cùng 1 đối tượng chứ không phải là 2 đối tượng để giữ nguyên màu sắc trên đối tượng thì đó chính là bài toán tracking.

#### 1.1.2.1 Phân loại

Tùy theo các tiêu chí khác nhau, có nhiều cách phân loại object tracking. Cách phân chia phổ biến nhất là dựa trên đối tượng mục tiêu của object tracking.

* Single Object Tracking (SOT): Single object tracking nhắm đến khả năng theo dõi chuyến động của một hoặc một số đối tượng được định nghĩa trước trong những khung hình đầu tiên. Mục tiêu của trình theo dõi sau đó là xác định cùng một mục tiêu trong tất cả các khung hình còn lại của video. Việc định nghĩa các đối tượng theo dõi ban đầu được thực hiện một cách thủ công bởi người dùng. Điều này có nghĩa là trình theo dõi có thể theo dõi bất kỳ đối tượng nào mà chúng được cung cấp, mà không cần quan tâm đến vấn đề phân loại đối tượng.
* Multiple Object Tracking (MOT): Multiple object tracking là bài toán khó hơn và mang tính ứng dụng rộng rãi hơn (hình 1.1). Thay vì việc người dùng cần định nghĩa trước đối tượng như SOT, trình theo dõi trong multiple object tracking cần tự phân loại và định danh tất cả các đối tượng xuất hiện trong video. Điều này thực sự là vấn đề khó xử lý khi số lượng đối tượng trong video là không biết trước, đặc biệt là trường hợp số lượng dày đặc. Một vấn đề khác đáng quan tâm là đảm bảo được liên kết và duy trì định danh của tất cả đối tượng qua từng khung hình.



Hình 1.1: Ví dụ về multiple object tracking.

Một cách phân chia object tracking phổ biến khác là dựa trên cách xử lý chuỗi khung hình của thuật toán.

* Offline Tracking: Offline tracking sử dụng toàn bộ các khung hình của video để xử lí và theo dõi. Cách xử lý này giúp trình theo dõi đạt được sự tối ưu toàn cục và đem lại độ chính xác vượt trội. Tuy nhiên, quy trình xử lý này lại khó để ứng dụng vào thực tế, khi mà các nguồn video thường đến từ các camera giám sát và luôn cập nhật liên tục.
* Online Tracking: Online tracking có cách xử lý đơn giản hơn, chỉ sử dụng khung hình hiện tại và một số khung hình trước đó để ước lượng vị trí của đối tượng. Dù độ chính xác thấp hơn các thuật toán offline tracking, nhưng online tracking lại là lựa chọn phù hợp nhất để ứng dụng vào thực tế nhằm phát triển các hệ thống theo dõi thời gian thực.

## Một số kỹ thuật Object Tracking phổ biến

### ****1.2.1 Centroid based Object Tracking****

Centroid-based object tracking tận dụng Euclidean distance (khoảng cách euclid) giữa các Tâm của đối tượng được phát hiện giữa 2 frames liên tiếp trong một video.

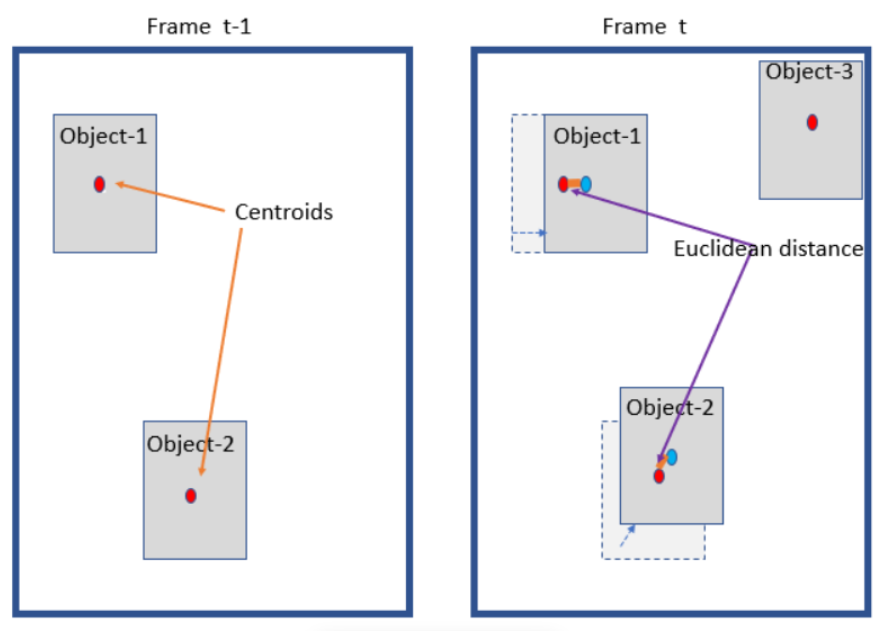
**Bước 1:**Tại thời gian t-1, Các Objects được phát hiện đóng khung trong các bounding boxes.

**Bước 2**: Tính toán các centroids đối với các đối tượng được phát hiện tại thời gian t-1.

**Bước 3**: Tại thời gian t, Các Objects được phát hiện đóng khung trong các bounding box . Gán các unique ID tới các objects.

**Bước 4**: Tính toán các centroids của các objects được phát hiện tại thời gian t.

**Bước 5**: Tính toán Euclidean distance giữa các centroids của các objects được phát hiện tại thời gian t-1 và t.



Hình 1.11. Kỹ thuật **Centroid based Object Tracking**

**Bước 6:** Nếu khoảng cách giữa centroids tại thời gian t-1 và t nhỏ hơn threshold 🡪 nó là cùng của một đối tượng chuyển động. Do đó sử dụng lại object ID hiện tại và cập nhật lại bounding box mới cho đối tượng.

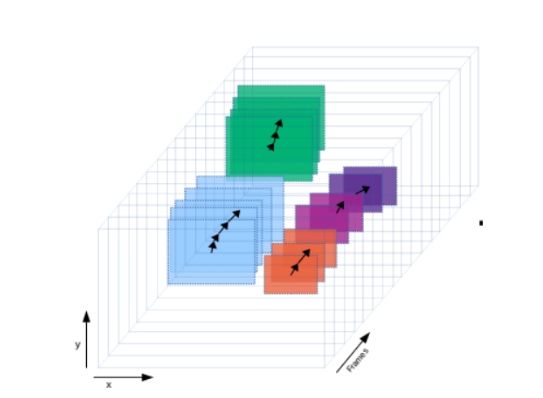
**Bước 7**: Nếu khoảng cách lớn hơn threshlod, thêm một object ID mới.

**Bước 8**: Khi các objects được phát hiện từ các frame phía trước không thể match với bất kỳ các objects đang tồn tại nào–> xóa object id ra khỏi quá trình truy vết

### 1.2.2 IOU Object Tracker

Intersection-over-Union là một kỹ thuật khác trong tracking liên kết các detections của các frames liên tiếp sử dụng sự chồng lấn không gian của nó để có thể truy vết.

Mỗi track sẽ lấy object được phát hiện với IOU cao nhất với một object đã được định danh gán ID trước đó. Sự liên kết này được giải quyết bằng một bài toán linear assignment – thuật toán Hungarian để tối đa hóa tổng của tất cả các IOUs trong từng frame.



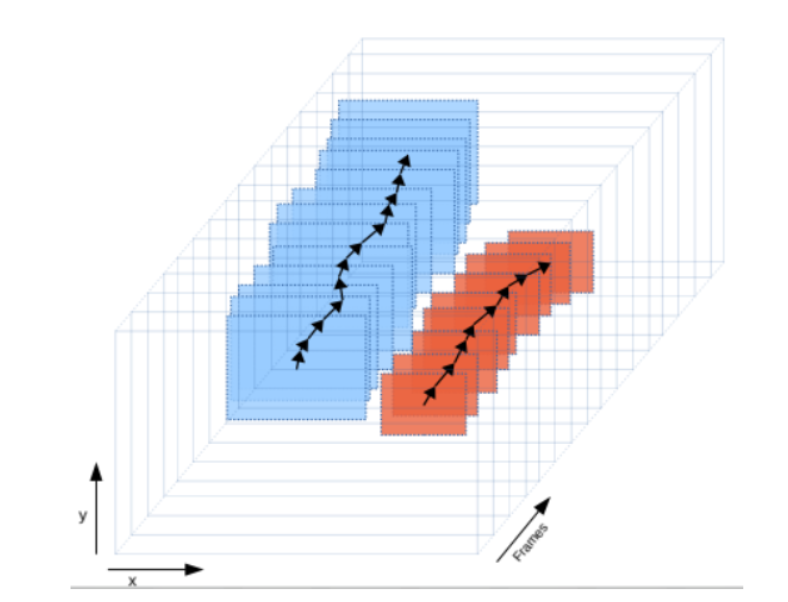
Hình 1.12. Kỹ thuật IOU Object Tracker

### 1.2.3 Visual IOU Object Tracker

Đây là một sự cải tiến đối với IOU-based Object Tracker bằng việc kết hợp với visual tracking. Visual IOU Object Tracker được thực hiện trong 2 hướng: visual forward và backward tracking nhằm giúp hợp nhất các tracks bị gián đoạn không liên tục.

* Khi không có object detection nào thỏa mãn IOU threshold đối với một object track, visual tracker sẽ được khởi tạo tại vị trí được biết đến ở frame trước đó của object và được sử dụng để truy vết object trong một vài frames nhất định.
* Nếu một object mới thỏa mãn IOU threshold được phát hiện chỉ trong một vài frames chỉ định kế tiếp, visual tracking sẽ dừng lại và IOT tracker sẽ tiếp tục. Nếu không có thì việc truy vết sẽ kết thúc.

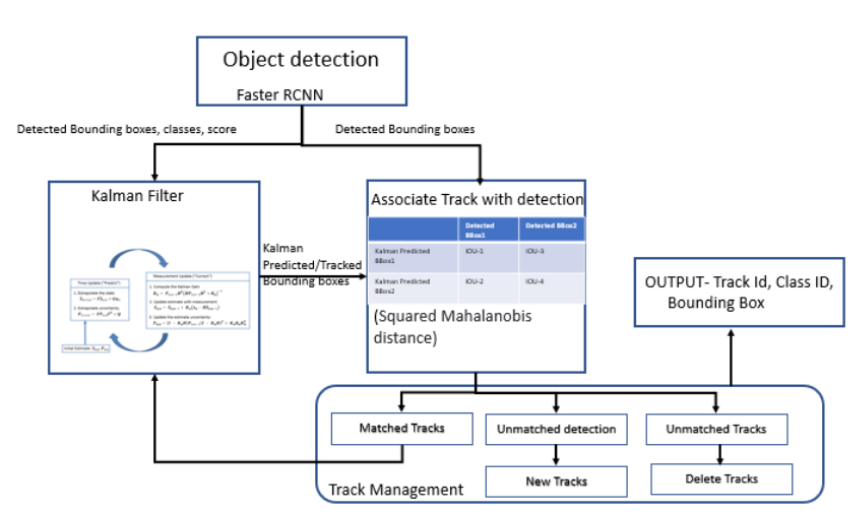
Hướng tiếp cận trên giúp giải quyết phần nào tình trạng mất dấu đối tượng.

****

Hình 1.13. Kỹ thuật Visual IOU Object Tracker

### 1.2.4 SORT - Simple Online Realtime Object Tracking

Thuật toán SORT dựa trên một giả định rằng chất lượng của tracking sẽ phụ thuộc vào performance của thuật toán object detection sử dụng.



Hình 1.14. Kỹ thuật SORT

Đầu tiện, SORT sử dụng mạng Faster Region-CNN(FrRCNN) dùng để detect các objects.

* Ở stage thứ nhất, FrRCNN sẽ trích xuất đặc trưng và đề xuất những vùng có khả năng chứa các objects cho stage thứ hai.
* Ở stage thứ hai, FrRCNN sẽ phân loại các objects trong ở các vùng được đề xuất.

FrRCNN chia sẻ các parameters giữa 2 stages, tạo ra một model hiệu quả trong việc detection.

Những track mới được khởi tạo khi các detections không thể liên kết với các tracks đã tồn tại trước đó. Khi một object detection với overlap nhỏ hơn IOU, báo hiệu rằng có sự tồn tại của một untracked object.

Những tracks mà vượt quá maximun age đã được định ra từ trước thì xem như đã ra khỏi ngữ cảnh và track sẽ bị xóa. Các tracks được chấm dứt nếu như chúng không được detected trong T(lost) frames, và T(Lost) frames thường được thiết lập bằng 1.

**Thuật toán SORT giúp giảm tình trạng mất dấu khi các mục tiêu bị che khuất, ID switches để hoạt động tốt khi chuyển động của các đối tượng tương đối chậm. SORT có thể gặp khó khăn trong trường hợp ngữ cảnh động đúc hoặc các đối tượng chuyển động nhanh.**

### 1.2.5 DeepSort

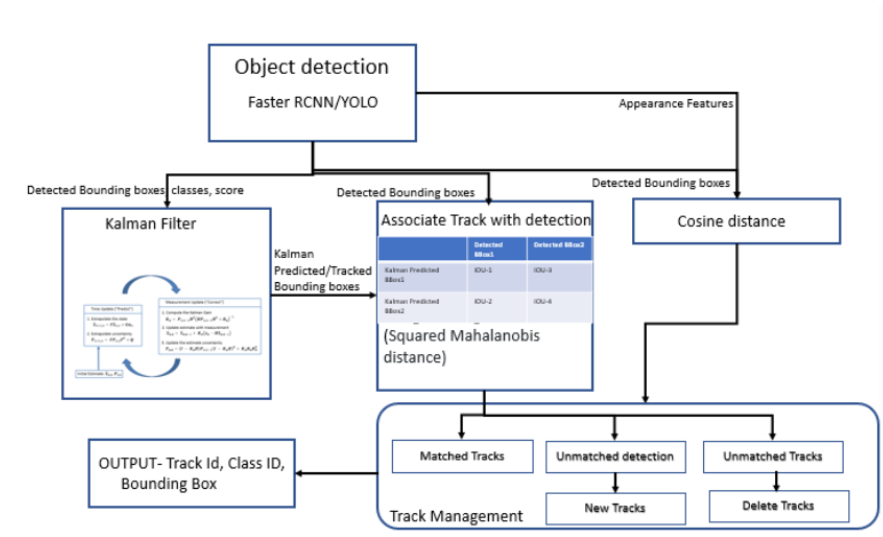
**Deep SORT và SORT là những phương pháp tracking theo cascade-style (kiểu thác nước).** Đầu tiên chúng thực hiện dự đoán những bounding boxes đối với những đối tượng được phát hiện và sau đó trích xuất các features từ chúng để ước tính các re-ID features tương ứng.

**Deep SORT cho phép thực hiện truy vết đối với những đối tượng bị chồng lấn trong khoảng thời gian dài hơn, đơn giản để implement và chạy trong real-time.**

Deep SORT cũng tuân theo một phương pháp tracking sử dụng**recursive Kalman filering và liên kết dữ liệu frame-by-frame sử dụng thuật toán Hungarian.**

Kalman filter dự đoán quỹ đạo có thể có của đối tượng trong tương lai dựa trên vị trí hiện tại. Nó lấy tọa độ bounding box tại vị trí quan sát hiện tại của đối tượng để cung cấp một ước tính sơ lược về vị trí di chuyển đến kế tiếp của đối tượng.

Để mỗi bounding boxes được đựa vào danh sách tracked của Kalman Filter, chúng ta phải liên kết những detections ở frame mới nhất với những predictions mới nhất của Kalman. **Quá trình liên kết đó được giải quyết sử dụng thuât toán Hungarian.**



Hình 1.15. Kỹ thuật DEEPSORT

**Deep SORT giúp giảm tình trạng ID Switches và occlusions (che lấp), từ đó giảm tỷ lệ False Positives.**

## Ứng dụng của Object Tracking

Object tracking là một trong những tác vụ của thị giác máy tính có nhiều ứng dụng thực tế nhất. Những ứng dụng này trải rộng trên nhiều lĩnh vực khác nhau như an ninh, quân sự, giao thông vận tải, ... Khả năng theo dõi một đối tượng trong video phụ thuộc vào nhiều yếu tố, như kiến thức về đối tượng đích, loại thông số được theo dõi và loại video hiển thị đối tượng.

### An ninh

Các hệ thống an ninh hiện nay yêu cầu tính tự động và bảo mật rất cao với sự góp mặt của một lượng lớn camera giám sát, điều mà khiến object tracking thật sự cần thiết. Các ứng dụng rõ ràng nhất của object tracking có thể kể đến như:

* Hệ thống nhận dạng con người dựa trên chuyển động cơ thể Giúp ích cho quá trình xác thực thông tin, nhận dạng đối tượng, đảm bảo an ninh trong khu vực giám sát
* Hệ thống giám sát tự động

Hỗ trợ người vận hành quan sát hình ảnh từ camera giám sát trên các màn hình hoặc lưới màn hành hình khổ lớn. Ngoài ra có thể tích hợp thêm các mô đun cảnh báo khi nhận thấy những hành động bất thường, khả nghi.

* Hệ thống nhận dạng hành vi

Khá tương tự hệ thống tự động giám sát, hệ thống hỗ trợ phân tích, đánh giá và phân loại hành vi của đối tượng. Tùy thuộc vào các hành vi này, hệ thống có thể đưa ra cảnh báo hoặc nhận định phù hợp.

### Thương mại

Trong lĩnh vực thương mại, object tracking tỏ ra vô cùng hữu ích, khi góp phần tạo ra nhiều ứng dụng, hệ thống giúp tăng chất lượng công việc, giảm chi phí lao động của các công ty và doanh nghiệp,

* Hệ thống giám sát hành vi người dùng

Hỗ trợ người quản lý đưa ra các chiến lược kinh doanh hợp lí nhờ quá trình phân tích hành vi, thói quen và sở thích của khách hàng. Ví dụ, bằng việc phân loại hoạt động của khách hàng trong siêu thị, hệ thống giúp xác định số lượng khách hàng quan tâm đến gian hàng và đánh giá hiệu quả trưng bày.

* Hệ thống theo dõi và giám định sản phẩm

Hỗ trợ thủ tục nhập xuất hàng hóa tại các siêu thị, bến cảng, ...

Với mục tiêu nhanh hơn, minh bạch hơn, ít giấy tờ hơn, hệ thống tuân thủ chính xác việc đưa hàng hóa ra/vào kho, tiếp nhận và trả hàng nhanh chóng cho khách hàng, đồng thời góp phần cắt giảm chi phí nhân công, thời gian đi lại giải quyết.

### 1.3.3 Giao thông vận tải

Một lĩnh vực tiềm năng không kém của object tracking là giao thông vận tải, khi mà hiện nay một lượng lớn camera giám sát đã được lắp đặt ở rất nhiều đoạn đường, ngã tư trọng điểm. Object tracking sẽ là vấn đề được quan tâm hàng đầu khi mong muốn tự động hóa quá trình vận hành toàn bộ lượng camera này. Một số hệ thống có thể kể đến như:

* Hệ thống giám sát giao thông

Hỗ trợ người vận hành quan sát hình ảnh giao thông từ xa trên các màn hình hoặc lưới màn hành hình khổ lớn tại trung tâm quản lý điều hành giao thông.

* Hệ thống điều hướng phương tiện

Tự động ước lượng số lượng phương tiện và hướng di chuyển, điều hướng giao thông nhằm giảm thiểu tắc nghẽn.

* Hệ thống đo đếm lưu lượng phương tiện

Thu thập, phân tích, báo cáo và lưu trữ thống kê các dữ liệu mô tả tình trạng và hiệu suất khai thác của hệ thống đường bộ.

### Các lĩnh vực khác

Ngoài các lĩnh vực kể trên Object Tracking còn rất nhiều ứng dụng trong các lĩnh vực khác của đời sống như:

* Phát hiện sản phẩm lỗi trong dây chuyền sản xuất, theo dõi vật nuôi
* Theo dõi hành vi sức khoẻ người cao tuổi
* Monitoring social distancing (theo dõi giãn cách xã hội): sử dụng trong đại dịch COVID
* …

# CHƯƠNG II: YOLO VÀ DEEPSORT TRONG BÀI TOÁN OBJECT TRACKING

## YOLO - You only look once

### 2.1.1 Tổng quan về YOLO

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

YOLO có thể không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ gần như real time mà độ chính xác không quá giảm so với các model thuộc top đầu.

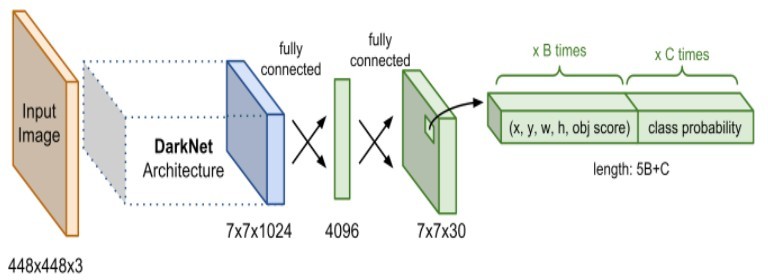
YOLO là thuật toán object detection nên mục tiêu của mô hình không chỉ là dự báo nhãn cho vật thể như các bài toán classification mà nó còn xác định location của vật thể. Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất một nhãn cho một bức ảnh.

Một trong nhưng ưu điểm mà YOLO đem lại đó là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ object box chứa các đối tượng, mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descent.

#### 2.1.1.1 Kiến trúc mạng YOLO

Kiến trúc YOLO bao gồm: Base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.

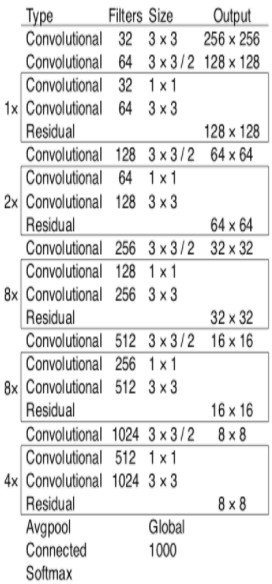
Base network của YOLO sử dụng chủ yếu là các convolutional layer và các fully conntected layer. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.[1]



*Hình 2.1: Kiến trúc mạng YOLO*

Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

Ở phiên bản thứ 3 của YOLO tức là YOLOv3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mẹo này có tác dụng giảm thiểu số lượng tham số cho mô hình.



*Hình 2.2: Các layer trong mạng darknet-5*

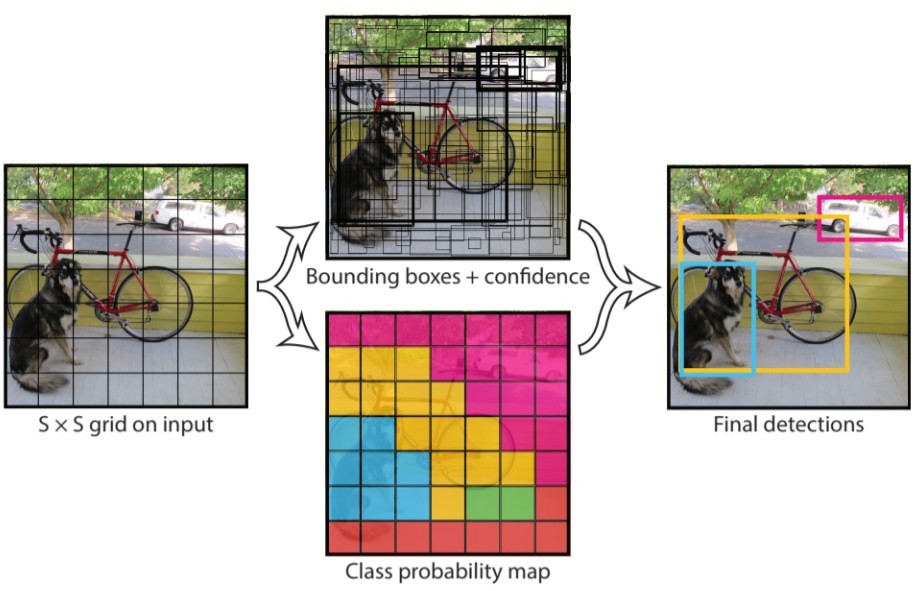
Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện

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

Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input 416x416 thì feature map có các kích thước là 13x13, 26x26 và 52x52. Và khi input là 608x608 sẽ tạo ra feature map 19x19, 38x38, 72x72.

#### 2.1.1.2 Nguyên lý hoạt đông của mạng YOLO

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không sau đó sẽ xác định toạ độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành SxS ô thường thì sẽ là 3x3, 7x7, 9x9,… Việc chia ô có ảnh hưởng đến việc phát hiện đối tượng của mô hình. [1]



*Hình 2.3: Cách hoạt đông của mạng YOLO*

Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước SxSx(5 x N + M) với số lượng tham số mỗi ô là (5 x N + M) với N và M lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán. Xét ví dụ ở hình trên chia thành 7x7 ô, mỗi ô cần dự đoán 2 bounding box và 3 object: con chó, ô tô, xe đạp thì output sẽ là 7x7x13, mỗi ô sẽ có 13 tham số, cho kết quả trả về (7x7x2 =98) bounding box.

### 2.1.2 Output của yolo

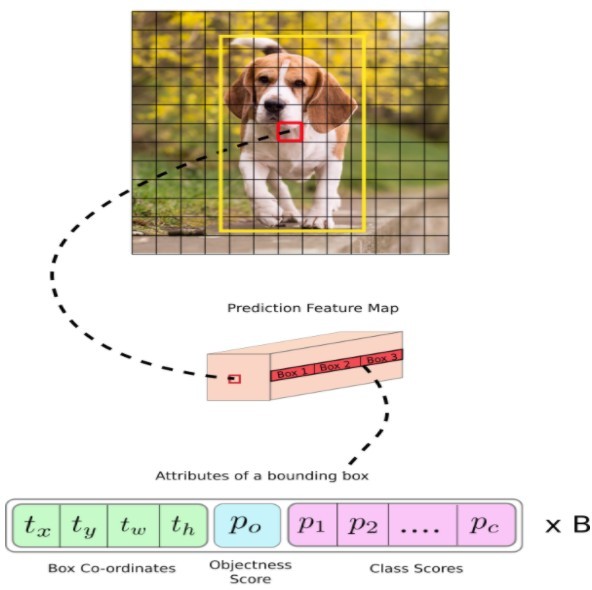
Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

yT  = [p0, (tx, ty, tw, th), (p1, p2, p3, p4) ]

trong đó:

* p­0  là xác suất dự báo vật thể xuất hiện trong bounding box.
* **(tx, ty, tw, th)** giúp xác định bounding box. Trong đó **tx, ty**, là tọa độ tâm và **tw, th** là kích thước rộng, dài của bounding box.
* **(p1, p2, p3, p4)** là véc tơ phân phối xác suất dự báo của các classes.

Như vậy output sẽ được xác định theo số lượng classes theo công thức (n\_class+5). Nếu huấn luyện 80 classes thì bạn sẽ có output là 85. Trường hợp áp dụng 3 anchors/cell thì số lượng tham số output sẽ là: (n\_class + 5) x 3= 85 x 3 = 255

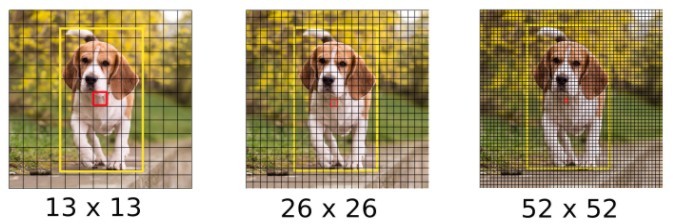


*Hình 2.4: Kiến trúc môt output của model YOLO*

Hình ảnh gốc là một feature map kích thước 13x13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate của 3 bounding boxes. Các attributes của một bounding box được mô tả như dòng cuối cùng trong hình.

#### 2.1.2.1 Dự báo trên nhiều feature map

Cũng tương tự như SSD, YOLO (cụ thể hơn là YOLOv3) dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ.[1]



*Hình 2.5: Các feature maps của mạng YOLOv3 với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52.*

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 featue map x 3 anchor box).

Đồng thời trên một feature map hình vuông SxS, mô hình YOLOv3 sinh ra một số lượng anchor box là: SxSx3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

(13x13+26+52x52)x3 = 10647( anchor box)

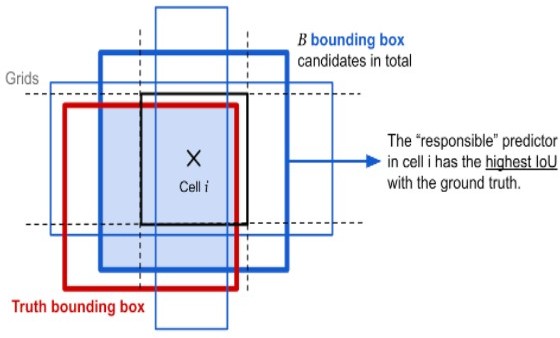
Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes

Một số lưu ý khi huấn luyện YOLO:

* Khi huấn luyện YOLO sẽ cần phải có RAM dung lượng lớn hơn để save được 10647 bounding boxes như trong kiến trúc này.
* Không thể thiết lập các batch\_size quá lớn như trong các mô hình classification vì rất dễ Out of memory. Package darknet của YOLO đã chia nhỏ một batch thành các subdivisions cho vừa với RAM.
* Thời gian xử lý của một step trên YOLO lâu hơn rất rất nhiều lần so với các mô hình classification. Do đó nên thiết lập steps giới hạn huấn luyện cho YOLO nhỏ. Đối với các tác vụ nhận diện dưới 5 classes, dưới 5000 steps là có thể thu được nghiệm tạm chấp nhận được. Các mô hình có nhiều classes hơn có thể tăng số lượng steps theo cấp số nhân tùy người dùng

#### 2.1.2.2 Anchor box

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Sau này thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO:

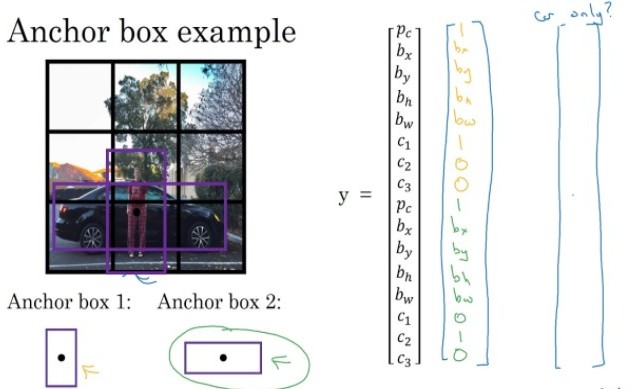
* + - * Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth bounding box là cao nhất.

*Hình 2.6: Xác định anchor box cho một vật thể*

Từ Cell i ta xác định được 3 anchor boxes viền xanh như trong hình. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất.

Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map mà chứa điểm mid point của vật thể. Chẳng hạn như hình chú chó trong hình 3 sẽ được phân về cho cell màu đỏ vì điểm mid point của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là (cell, anchor box). Không chỉ riêng mình cell hoặc chỉ mình anchor box.

Một số trường hợp 2 vật thể bị trùng mid point, mặc dù rất hiếm khi xảy ra, thuật toán sẽ rất khó xác định được class cho chúng.

*Hình 2.7: Khi 2 vật thể người và xe trùng mid point và cùng thuộc một cell. Thuật toán sẽ cần thêm những lượt tiebreak để quyết định đâu là class cho cell*

### 2.1.3 Dự báo bounding box

Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell.

YOLO dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

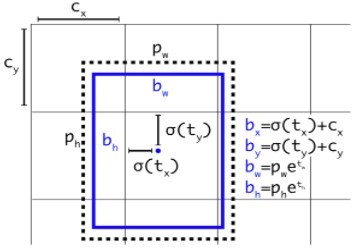
Cho một anchor box có kích thước (pw ,ph) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (Cx , Cy) mô hình dự đoán 4 tham số ( tx,ty,tw,th) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (bx,by) à kích thước (bw,bh) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

bx=(tx) + cx

by=(ty) + cy bw= w*etw*

bh= h*eth*

Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0, 1]. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.



*Hình 2.9: Công thức ước lượng bounding box từ anchor box*

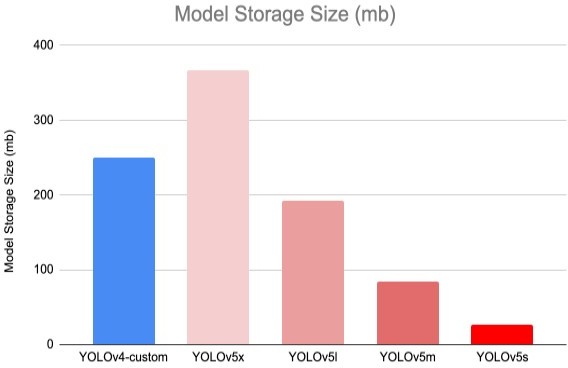
Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (pw,ph). Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này.

### 2.1.4 YOLOv5

YOLOv5 là môt bản cải tiến mang tính mở rông theo môt cách tự nhiên của YOLOv3 PyTorch bởi tác giả Glenn Jocher. Kho lưu trữ YOLOv3 PyTorch là điểm đến phổ biến cho các nhà phát triển để chuyển các trọng số YOLOv3 Darknet sang PyTorch và sau đó chuyển sang sản xuất. Những cải tiến này ban đầu được gọi là YOLOv4 nhưng do việc phát hành gần đây của YOLOv4 trong khuôn khổ Darknet, để tránh xung đột phiên bản, nó đã được đổi tên thành YOLOv5. Thuật toán YOLOv5 về cơ bản cũng thừa kế các phương pháp cơ bản của các YOLO, tuy nhiên YOLOv5 áp dụng một số thuật toán phát hiện vật thể nhanh, tối ưu hóa các phép toán thực hiện song song giúp tăng tốc độ nhận diện và giảm thời gian huấn luyện một cách tối ưu.

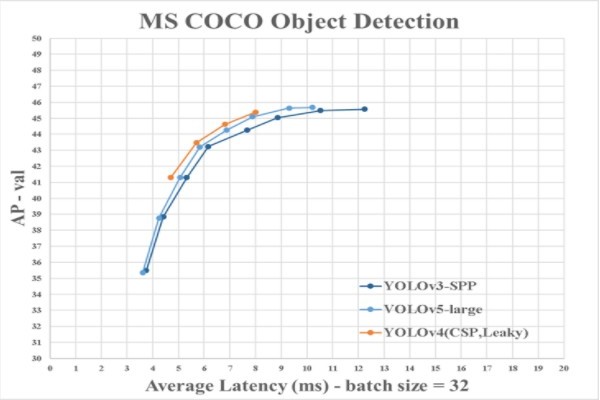
#### 2.1.4.1 Phân loại YOLOv5

Có 4 mô hình khác nhau: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. Đầu tiên là nhỏ nhất và kém chính xác nhất, cuối cùng là lớn nhất với độ chính xác lớn nhất. Tất cả các mô hình đều chạy trên PyTorch. Về hiêu năng: YOLOv5 nhỏ hơn và thường dễ sử dụng hơn trong sản xuất. Do nó được triển khai nguyên bản trong PyTorch (chứ không phải Darknet), việc sửa đổi kiến trúc và rất đơn giản trong viêc khác nhau.



*Hình 2.11: So sánh kích thước lưu trữ Model của các mẫu mã YOLOv5*

Về tốc đô ̣ : YOLOv5 thực hiện suy luận hàng loạt ở khoảng 140 FPS theo mặc định. Về đô ̣ chính xác: YOLOv5 gần như chính xác như YOLOv4 trong các tác vụ nhỏ (0,895 mAP so với 0,892 mAP trên BCCD). Trên các tác vụ lớn hơn như COCO, YOLOv4 hoạt động hiệu quả hơn.

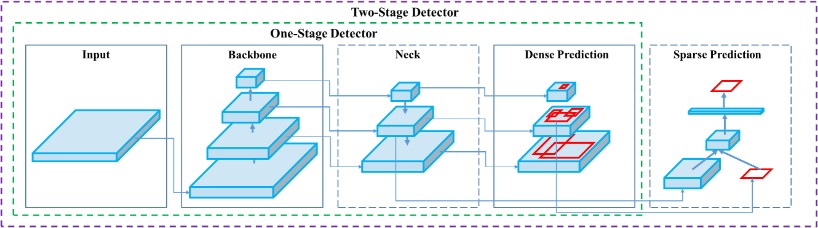


*Hình 2.12: So sánh độ trễ trung bình giữa các phiên bản YOLO(v3,v4,v5)*

#### 2.1.4.2 Cấu trúc của Yolov5

Bao gồm 3 phần chính: nhân diên vât thể (Object Detection)

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



*Hình 2.13: Cấu trúc nhân diên vât thể của YOLOv5*

#### 2.1.4.3 Những cải tiến của YOLOv5 so với các phiên bản trước

YOLOv5 được triển khai trong PyTorch ban đầu nên nó được hưởng lợi từ hệ sinh thái PyTorch đã được thiết lập: hỗ trợ đơn giản hơn và triển khai dễ dàng hơn. Hơn nữa, là một khung nghiên cứu được biết đến rộng rãi hơn, việc lặp lại trên YOLOv5 có thể dễ dàng hơn cho cộng đồng nghiên cứu rộng lớn hơn. Điều này cũng làm cho việc triển khai đến các thiết bị di động đơn giản hơn vì mô hình có thể được biên dịch sang ONNX và CoreML một cách dễ dàng.[6] Khả năng đào tạo cũng như khả năng suy luận rất là nhanh, độ chính xác cao. Cuối cùng YOLOv5 có dung lượng nhỏ. YOLOv5 rất nhỏ. Cụ thể, một tệp trọng số cho YOLOv5 là 27 megabyte. Trong khi đó môt tệp trọng số của cho YOLOv4 (với kiến trúc Darknet) là 244 megabyte. YOLOv5 nhỏ hơn gần 90% so với YOLOv4. Điều này có nghĩa là YOLOv5 có thể được triển khai cho các thiết bị nhúng dễ dàng hơn nhiều.[6]

## 2.2 SORT - Simple Online Realtime Object Tracking và DEEP SORT

### 2.2.1 Giải thuật Hungary

Giải thuật Hungary được phát triển và công bố vào năm 1955, đề xuất để giải bài toán phân công công việc (assignment problem).

Tương tự, với bài toán phân công công việc, object tracking cần giải quyết như sau Có n detection (i = 1, 2, …, n) và n track predicted (j = 1, 2, … n). Để liên kết một detection i với một track j giả sử dựa vào 1 độ đo D - D là khoảng cách giữa i và j trong không gian vector . Bài toán đặt ra là cần liên kết mỗi detection với mỗi track tương ứng sao cho sai số của việc liên kết là nhỏ nhất.

Với: -

-

-

Các thỏa mãn các điều kiện trên gọi là một phương án phân công, một phương án đạt cực tiểu của z được gọi là một phương án tối ưu hay lời giải của bài toán.

#### Các bước giải thuật toán Hungary

* **Bước 1:** (Bước chuẩn bị). Trừ các phần tử trên mỗi hàng của C cho phần tử nhỏ nhất trên hàng đó, tiếp theo trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó. Kết quả ta nhận được ma trận C' có tính chất: trên mỗi hàng, cột có ít nhất một phần tử 0 và bài toán giao việc với ma trận C' có cùng lời giải như bài toán với ma trận C.
* **Bước 2**: Vẽ một số tối thiểu các đường thẳng trên dòng và cột để đảm bảo mọi phần tử 0 đều được đi qua.
* **Bước 3:** Nếu có n đường thẳng được vẽ, kết thúc thuật toán và tiến hành phân công công việc. Nếu số đường thẳng được vẽ nhỏ hơn n, vẫn chưa tìm được phương án phân công tối ưu, tiến hành bước tiếp theo.
* **Bước 4:** Mỗi hàng (hoặc cột) có đường thẳng vẽ qua, ta gọi các hàng (cột) đó là các hàng (cột) thiết yếu. Các hàng (cột) còn lại là các hàng (cột) không thiết yếu. Tìm phần tử nhỏ nhất không nằm trong các hàng (cột) thiết yếu, tiến hành trừ mỗi hàng không thiết yếu cho phần từ nhỏ nhất ấy và cộng giá trị nhỏ nhất ấy cho cột thiết yếu. Ta được ma trận C’’ có cùng lời giải với ma trận C’. Sau đó quay lại Bước 2

#### 2.2.2.2 Ví dụ

Bài toán: Phân công cho 5 nhân viên đi địa điểm nào với thời gian là thấp nhất

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nhân Viên  / Địa điểm | I | II | III | IV | V |
| Nhân | 54 | 40 | 25 | 42 | 52 |
| Lễ | 50 | 59 | 18 | 40 | 37 |
| Nghĩa | 61 | 45 | 32 | 40 | 45 |
| Trí | 33 | 35 | 29 | 45 | 15 |
| Tín | 35 | 30 | 20 | 22 | 40 |

Bước 1: Trừ các phần tử trên mỗi hàng của C cho phần tử nhỏ nhất trên hàng đó, tiếp theo trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 29 | 15 | 0 | 17 | 27 |
| 32 | 41 | 0 | 22 | 19 |
| 29 | 13 | 0 | 8 | 13 |
| 18 | 20 | 14 | 30 | 0 |
| 15 | 10 | 0 | 2 | 20 |
| 14 | 5 | 0 | 15 | 27 |
| 17 | 31 | 0 | 20 | 19 |
| 14 | 3 | 0 | 6 | 13 |
| 3 | 10 | 14 | 28 | 0 |
| 0 | 0 | 0 | 0 | 20 |

Bước 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 5 | 0 | 15 | 27 |
| 17 | 31 | 0 | 20 | 19 |
| 14 | 3 | 0 | 6 | 13 |
| 3 | 10 | 14 | 28 | 0 |
| 0 | 0 | 0 | 0 | 20 |

Bước 3+4:

3 đường thẳng đã được vẽ < n=5

Chọn số 3 là phần tử nhỏ nhất của cột(hàng) chưa tối ưu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 2 | 0 | 12 | 27 |
| 14 | 28 | 0 | 17 | 19 |
| 11 | 0 | 0 | 3 | 13 |
| 0 | 7 | 14 | 25 | 0 |
| 0 | 0 | 3 | 0 | 23 |

4 đường đã được vẽ < n = 5 tiếp tục thực hiện (chọn 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 2 | 0 | 9 | 24 |
| 11 | 28 | 0 | 14 | 16 |
| 8 | 0 | 0 | 0 | 10 |
| 0 | 10 | 17 | 25 | 0 |
| 0 | 3 | 6 | 0 | 23 |

4 đường đã được vẽ < n = 5 vẫn tiếp tục thực hiện (chọn 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 0 | 0 | 7 | 22 |
| 9 | 26 | 0 | 12 | 14 |
| 8 | 0 | 2 | 0 | 10 |
| 0 | 10 | 19 | 25 | 0 |
| 0 | 3 | 8 | 0 | 23 |

5 đường thẳng đã được vẽ = n=5 kết thúc thuật toán

Kết luận: - Nhân đi địa điểm II với thời gian là 40phut

- Lễ đi địa điểm III với thời gian là 18p

- Nghĩa đi địa điểm IV với thời gian là 40p

- Trí đi địa điểm V với thời gian là 15p

- Tín đi địa điểm I với thời gian là 35p

- Tổng thời gian thấp nhất là 148p

### 2.2.2 Bộ lọc Kalman

Bộ lọc Kalman (Kalman Filter) là một mô hình Linear-Gaussian State Space Model, được giới thiệu lần đầu năm 1960 bởi Rudolf (Rudy) E. Kálmán, là thuật toán sử dụng chuỗi các giá trị đo lường, bị ảnh hưởng bởi nhiễu hoặc sai số, để ước đoán biến số nhằm tăng độ chính xác so với việc sử dụng duy nhất một giái trị đo lường. Bộ lọc Kalman thực hiện phương pháp truy hồi đối với chuỗi các giá trị đầu vào bị nhiễu, nhằm tối ưu hoá giái trị ước đoán trạng thái của hệ thống. Bộ lọc Kalman được ứng dụng trong rất nhiều lĩnh vực khác nhau: Xe tự lái, thực tế ảo, kinh tế lượng, tracking, điều khiển tối ưu, ...

Trong object tracking, Kalman filter được biết đến nhiều nhất với vai trò dự đoán các trạng thái của đối tượng hiện tại dựa vào các track trong quá khứ và update lại các detection sau khi đã được liên kết với các track trước đó.

Quá trình cần xử lí là 1 quá trình ngẫu nhiên với các mô hình đã được định nghĩa từ trước:

Với:

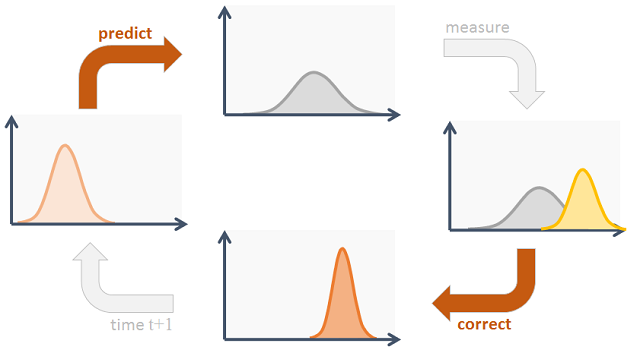
* là giá trị biến trạng thái của quá trình, thường là các giá trị ẩn, không thể quan sát được.
* là giá trị đo được, quan sát được của quá trình.
* là các mô hình định nghĩa từ trước.
* lần lượt là nhiễu của quá trình và nhiễu trong lúc đo đạc.

Áp dụng thuật toán Linear Kalman Filter (ngoài ra còn có Extended Kalman Filter, Unscented Kalman Filter, …) vào bài toán.

Linear Kalman Filter giả định các mô hình của quá trình đều là các mô hình tuyến tính. Khi đó:

Với:

* ~ N (: p(=
* là các ma trận n\*n (state transistion matrix) và m\*n (measurement function)
* ~ N(0,Q), ~ N(0,R)



Các bước xử lý tiếp theo của Kalman Filter có thể chia làm 2 phần chính (cách tiếp cận dựa trên xác xuất):

**Dự đoạn (Prediction):**

Để dự đoán các giá trị trạng thái của quá trình ngẫu nhiên, ta dự đoán các giá trị mean và covariance Σ. Theo tính chất kỳ vọng và ma trận phương sai với vector ngẫu nhiên, ta có:

**Hiệu chỉnh (Update):**

Áp dụng định lý Bayes cho Linear Gaussian System:

Tiếp tục áp dụng đồng nhất thức ma trận Woodbury:

“Đồng nhất thức ma trận Woodbury (Matrix Inversion Lemma)

Có 4 ma trận A(n\*n), U(n\*k), C(k\*k), V(k\*n).

Khi đó (A+UCV)-1 = A-1 – A-1U(C-1+VA-1U)-1VA-1”

Đặt với được gọi là hệ số Kalman tại thời điểm k, khi đó:

)

|  |  |
| --- | --- |
| Dự đoán (Prediction) | Hiệu chỉnh (Update) |
|  | * ) |

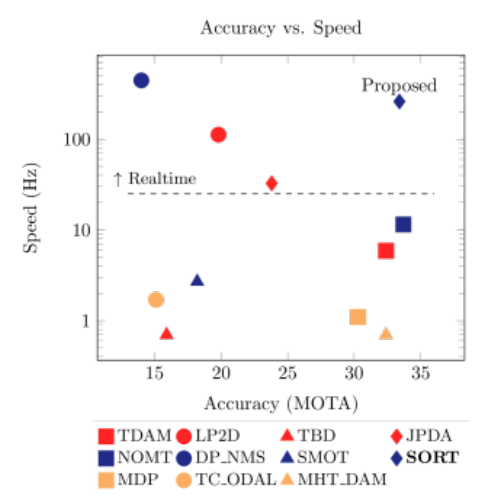
### Sort

SORT - Simple Online and Realtime Tracking, được nhóm tác giả từ đại học Sydney và đại học Queensland giới thiệu lần đầu vào ngày 2/2/2016, nhằm đề xuất giải pháp cho bài toán theo dõi đối tượng theo thời gian thực.

Theo paper, bài toán theo dõi đa đối tượng (Multiple Object Tracking) trong một video có thể được xem xét như bài toán liên kết dữ liệu (data association) mà mục đích là liên kết được các đối tượng phát hiện (detection) qua từng frame. Dựa trên cơ sở này, nhóm tác giả bỏ qua phần detector, mà tập trung sâu hơn vào các thuật toán liên kết các đối tượng nhằm đảm bảo được tính liên tục (online) và sử dụng được trong thời gian thực (realtime) khi xử lí video.

Vào thời điểm ra mắt paper, các thuật toán truyền thống cho việc multiple object tracking, có thể kể đến như Multiple Hypothesis Tracking (MHT) và Joint Probabilistic Data Asociation (JPDA), mặc dù đạt được những kết quả khá tốt, nhưng những thuật toán này có thời gian xử lý tương đối lớn khiến nó khó có thể phù hợp cho việc online tracking.

SORT sử dụng một bộ detector từ Faster Region CNN detection frame work đồng thời tập trung vào vấn đề data association bằng việc kết hợp bộ lọc Kalman và thuật toán Hungary dựa trên chỉ số IOU nhằm tìm các liên kết tương ứng cho đối tượng được phát hiện và các đối tượng đang được theo dõi. Thuật toán này thật sự tỏ ra hiệu quả khi tốc độ tracking nhanh hơn gấp 20 lần so với những phương pháp hiệu quả nhất lúc bấy giờ



Hình 3.9: So sánh SORT và các thuật toán tracking khác.

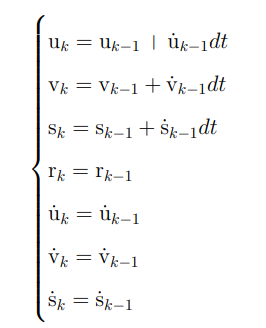
#### **2.1.3.1** Mô hình hóa bài toán và bộ lọc Kalman

Để mô hình hóa lại chuyển động của đối tượng cần theo dõi, SORT giả định việc chuyển động của các đối tượng là chuyển động đều (vận tốc không đổi), và các đối tượng có chuyển động độc lập với nhau. Véc tơ trạng thái được xác định với 7 thành phần:

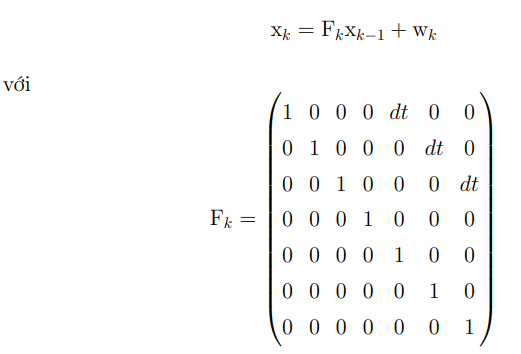


Với (u*,* v) ở đây là tọa độ tâm của đối tượng (ở đây là tâm của bounding box), s là diện tích, r là tỉ lệ giữa chiều dài và chiều rộng và các vận tốc tương ứng ˙u*,* ˙v*,* ˙s (với tỉ lệ cạnh r là không đổi).

Do giả định các đối tượng chuyển động đều , ta có:



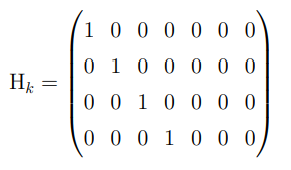
Khi đó, với w*k* được khởi tạo tuân theo phân phối chuẩn, có kì vọng = 0 và phương sai không đổi, ta có:



Mặt khác, ma trận mô hình quan sát của bài toán được xác định:

z*k* = H*k*x*k* + v*k*

với v*k* được khởi tạo tuân theo phân phối chuẩn, có kì vọng = 0 và phương sai không đổi và

**

Từ đây, SORT áp dụng bộ lọc Kalman lần lượt thực hiện 2 giai đoạn: Dự đoán và hiệu chỉnh để ước lượng các trạng thái về quỹ đạo của đối tượng.

#### 2.1.3.2 Vấn đề liên kết và thuật toán Hungary

Để bộ lọc Kalman có thể ước lượng chính xác trạng thái cho từng đối tượng, cần đảm bảo tính nhất quán cho các detection được đưa vào bộ lọc Kalman. Đây là bài toán chính mà vấn đề liên kết dữ liệu cần giải quyết, nhằm đảm bảo bộ lọc Kalman không bị ước lượng nhầm giữa các đối tượng trong khung hình.

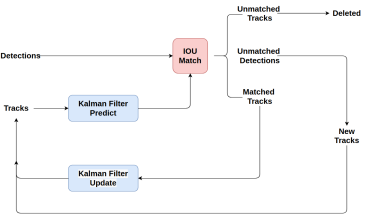
Bài toán có thể viết lại dưới góc nhìn của bài toán phân công như sau: *Có n detection - các bounding box của đối tượng (i = 1, 2, . . . , n) và n track - các trạng thái đã được ước lượng trước đó (j = 1, 2, . . . n). Để liên kết một detection i với một track j giả sử dựa vào 1 độ đo dij - dij là khoảng cách giữa i và j trong không gian vector . Bài toán đặt ra là cần liên kết mỗi detection với mỗi track tương ứng sao cho sai số của việc liên kết là nhỏ nhất.*

Việc xác định *dij* là giai đoạn Liên hệ (Affinity) trong các thuật toán Tracking by detection. SORT sử dụng dụng độ đo IOU để xác định đại lượng này. IOU được tính toán dựa trên các tọa độ của bounding box, do đó, SORT quan tâm nhiều các đặc điểm về hình dạng của đối tượng, và lấy các đặc điểm này để liên kết giữa detection và track với nhau.

Ngoài ra, một ngưỡng IOU đủ nhỏ (IOU min) cũng được sử dụng để loại đi những mối liên kết quá thấp, nhằm giúp giảm thiểu khối lượng tính toán cho mô hình.

#### 2.1.3.3 Tổng kết thuật toán

Luồng xử lí thuật toán có thể được mô tả gắn gọn như sau (hình 3.3):



Hình 3.10: Luồng xử lí trong SORT.

* Bước 1: Sử dụng Faster Region CNN (với backbone là VGG16) để phát hiện các đối tượng trong khung hình hiện tại.
* Bước 2: SORT tiến hành sử dụng Kalman Filter để dự đoán các trạng thái track mới dựa trên các track trong quá khứ.
* Bước 3: Sử dụng những track vừa dự đoán được, kết hợp với các detection thu được từ trình phát hiện (detector), xây dựng ma trận chi phí cho vấn đề liên kết. Chi phí được sử dụng để đánh giá ở đây là giá trị IOU giữa các bounding box của track và detection.
* Bước 4: Sử dụng giải thuật Hungary giải bài toán phân công với ma trận chi phí vừa lập.
* Bước 5: Xử lí, phân loại các detection.
* Bước 6: Sử dụng Kalman filter để hiệu chỉnh lại giá trị của track từ những detection đã được liên kết với track.

Bên cạnh đó, còn một số điểm khác đáng chú ý trong xử lí của SORT:

* Với những detection có giá trị assignment nhỏ hơn IOU min, các đối tượng này được nhận định như một đối tượng mới xuất hiện trong video và sẽ được gán cho 1 id mới. Giá trị vận tốc cho những đối tượng này cũng được khởi tạo bằng 0 và phương sai được để giá trị lớn để thể hiện sự không chắc chắn về giá trị vận tốc.
* Để xóa đi những track đã ra khỏi tầm quan sát của video, SORT đưa ra một ngưỡng là *TLoss*. Cụ thể hơn, với tất cả các những track mà không được phát hiện sau *TLoss* frame sau đó, những track này được nhận định là đã ra khỏi màn hình và không còn được theo dõi nữa.
* Trong SORT, giá trị *TLoss* được chọn bằng 1 vì việc giả định vận tốc đối tượng là không đổi vốn đã là 1 giả định kém, không phù hợp với thực tế, việc xử lí frame-to-frame để sớm xóa đi các mục tiêu đã mất sẽ khiến việc theo dõi hiệu quả hơn.

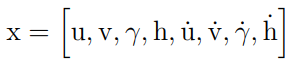
### Deep Sort

Mặc dù đạt được tốc độ xử lí cũng như performance khá cao, SORT gặp một vấn đề là số lượng switches id (switches id là tổng số lượng được gán cho cùng 1 đối tượng trong suốt quá trình theo dõi) khá lớn. Điều này xảy ra do thuật toán của SORT chỉ đảm bảo độ chính xác khi việc ước lượng trạng thái (cập nhật và điều chỉnh) là đủ tốt, điều mà khó đạt được trong trường hợp các đối tượng che lấp nhau hoặc các bounding box trùng khớp nhau nhiều.

Deep SORT - Simple Online and Realtime Tracking with a Deep As sociation Metric[24], được Nicolai Wojke và Alex Bewley phát triển ngay sau đó nhằm giải quyết các vấn đề thiếu sót của SORT. Hướng giải quyết mà deep SORT đề xuất dựa trên việc sử dụng deep learning để trích xuất các đặc trưng của đối tượng nhằm tăng độ chính xác trong quá trình liên kết dữ liệu. Ngoài ra, một chiến lược liên kết cũng được xây dựng mang tên Matching Cascade giúp việc liên kết các đối tượng sau khi đã biến mất 1 thời gian được hiểu quả hơn.

#### **2.1.4.1** Mô hình hóa bài toán và bộ lọc Kalman

Do SORT đã mô hình hóa bài toán khá ổn, bên deep SORT, các thành phần trong trạng thái track không thay đổi quá nhiều. Mỗi track bao gồm 8 thành phần:

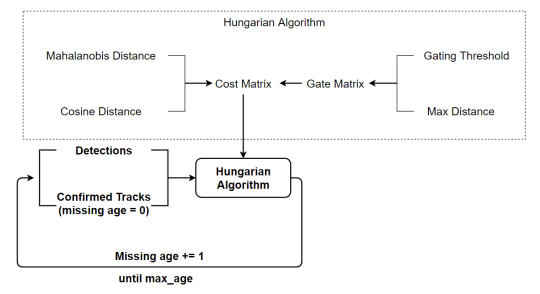


Với (u*,* v) ở đây là tọa độ tâm của đối tượng (ở đây là tâm của bounding box), *γ* là tỉ lệ khung hình, h là chiều cao của bounding box, và các vận tốc tương ứng ˙u*,* ˙v*,* ˙y, ˙h.

Các mô hình trạng thái và mô hình quan sát của deep SORT hầu như giống với SORT.

#### 2.1.4.2 Vấn đề liên kết và thuật toán Hungary

Các thiếu sót của SORT về lượng switches ID được nhóm tác giả nhận định là do vấn đề liên kết dữ liệu chưa đủ tốt. Do đó, bên cạnh việc chỉ quan tâm đến hình dạng của đối tượng (sử dụng IOU), deep SORT tiến hành sử dụng thêm 2 độ đo nữa, độ đo về khoảng cách và đặc trưng của đối tượng (hình 3.11).



Hình 3.11: Vấn đề liên kết dữ liệu trong deep SORT.

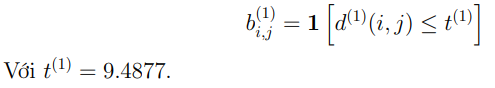
Để trích xuất thông tin về chuyển động, khoảng cách Mahalanobis được sử dụng:



Trong đó (***y****i,****S****i*) là giá trị kì vọng và ma trận hiệp phương sai của track thứ i, và ***d****j* là giá trị của detection thứ j.

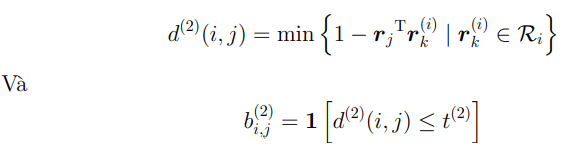
Khoảng cách Mahalanobis là thước đo khoảng cách giữa điểm P và phân phối D, do P. C. Mahalanobis đưa ra vào năm 1936. Nó là sự tổng quát hóa đa chiều về ý tưởng đo lường độ lệch chuẩn giữa P so với giá trị kì vọng của D. Khoảng cách này bằng 0 nếu P ở giá trị kì vọng của D và tăng lên khi P di chuyển ra xa giá trị kì vọng này.

Ngoài việc đo lường khoảng cách giữa track và detection, khoảng cách Mahalanobis còn được dùng để loại trừ các liên kết không chắc chắn bằng cách lập ngưỡng khoảng cách Mahalanobis ở khoảng tin cậy 95% được tính từ phân phối *χ*



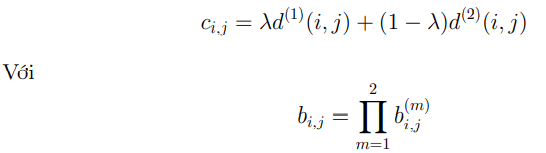
Deep SORT đồng thời sử dụng một độ đo khác về đặc trưng của đối tượng, nhằm đảm bảo việc liên kết chuẩn xác dù đối tượng đã biến mất và sau đó xuất hiện trở lại trong khung hình. Để thực hiện việc này, một bộ trích xuất đặc trưng được sử dụng. Với mỗi detection, đặc trưng *rj* được trích xuất với *||****r****j||* = 1. Với mỗi track, một danh sách với độ dài khoảng 100 được sử dụng để lưu trữ đặc trưng của 100 track gần nhất:



Khi đó, độ đo mới giữa track và detection được tính bằng khoảng cách cosine: 

với *t*(2) được lựa chọn tùy vào tập dữ liệu sử dụng.

Khoảng cách Mahalanobis cung cấp các thông tin về vị trí đối tượng dựa trên chuyển động tức thời, đặc biệt hữu ích cho các dự đoán ngắn hạn. Mặt khác, khoảng cách cosine xem xét thông tin về đặc trưng của đối tượng, đặc biệt hữu ích cho các dự đoán dài hạn hoặc các đối tượng khó phân biệt. Bằng việc kết hợp 2 độ đo với trọng số phù hợp, deep SORT tạo ra một độ đo mới:



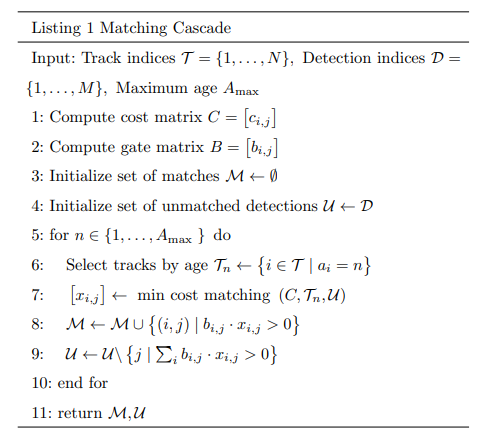
Giá trị *bi,j* sau đó được sử dụng trong chiến lược liên kết matching cascade sẽ được trình bày ở phần sau. Giá trị *λ* có thể được điều chỉnh tùy theo trường hợp sử dụng để cân bằng giữa 2 độ đo. Trong bài báo, tác giả để *λ* = 0, tức là không sử dụng đến khoảng cách Mahalanobis. Dù vậy, khoảng cách Mahalanobis vẫn hữu ích, do có thể lọc bỏ những detection có độ liên kết không đảm bảo ngưỡng.

Sau cùng, khi được lọc qua các độ đo này, giá trị IOU được sử dụng để đưa ra những liên kết cuối cùng.

#### 2.1.4.3 Chiến lược đối sánh theo tầng (Matching Cascade)

Deep SORT đề xuất một chiến lược đối sánh theo tầng nhằm cải thiện độ chính xác của liên kết, chủ yếu là vì khi đối tượng biến mất trong thời gian dài, độ không chắc chắn của bộ lọc Kalman sẽ tăng lên rất nhiều và sẽ dẫn đến phân tán xác suất dự đoán liên tục. Vì vậy, nếu dự đoán liên tục không được cập nhật, phương sai của phân phối chuẩn sẽ ngày càng lớn. Khi đó, giá trị của khoảng cách Mahalanobis giữa các điểm xa giá trị kì vọng và các điểm gần giá trị kì vọng là như nhau.

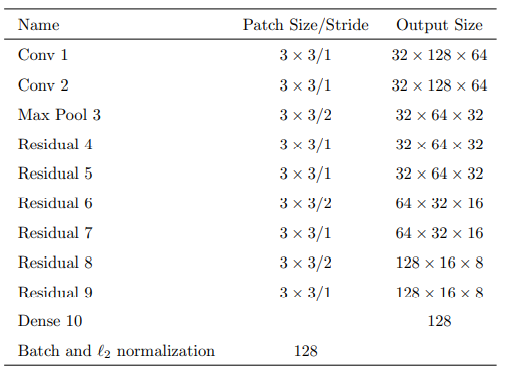
Chiến lược đối sánh theo tầng tiến hành lấy lần lượt từng track ở các frame trước đó, để tiến hành xây dựng ma trận chi phí và giải bài toán phân công theo từng tầng. Chi tiết thuật toán được trình bày cụ thể hơn ở mã giả dưới đây:



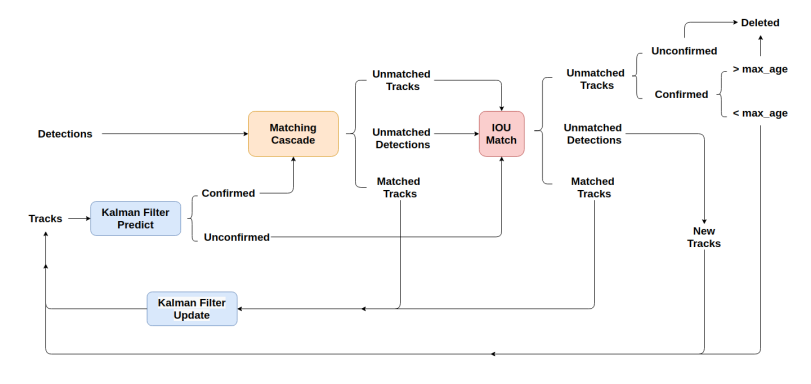
#### 2.1.4.4 Bộ trích xuất đặc trưng đối tượng

Để trích xuất được đặc trưng của đối tượng từ bounding box thu được, Deep SORT sử dụng một mạng WRN (Wide Residual Network). Mạng WRN này được huấn luyện riêng trên một bộ dữ liệu quy mô lớn về định danh người (large-scale persion re-id dataset) với khoảng 1.100.000 hình ảnh về 1.261 người đi bộ.

Cấu trúc mạng WRN được mô tả dưới đây:



**3.2.4.5 Tổng kết thuật toán**

****

Hình 3.12: Luồng xử lí trong deep SORT.

Luồng xử lí của Deep SORT được thực hiện tuần tự qua các bước dưới đây (hình 3.11):

* Bước 1: Sử dụng Faster Region CNN (với backbone là VGG16) để phát hiện các đối tượng trong khung hình hiện tại.
* Bước 2: Deep SORT sử dụng Kalman Filter để dự đoán các trạng thái track mới dựa trên các track trong quá khứ. Các trạng thái này lúc mới khởi tạo sẽ được gán 1 giá trị mang tính thăm dò (tentative). Giá trị này nếu vẫn đảm bảo duy trì được trong 3 frame tiếp theo, trạng thái sẽ chuyển từ thăm dò sang xác nhận (confirmed), và sẽ cố gắng được duy trì theo dõi trong 30 frame tiếp theo. Ngược lại, nếu mất dấu khi chưa đủ 3 frame, trạng thái sẽ bị xóa khỏi trình theo dõi.
* Bước 3: Sử dụng những track đã được xác nhận, tiến hành đưa vào chiến lược đối sánh phân tầng (matching cascade) nhằm liên kết với các detection phát hiện được dựa trên độ đo về khoảng cách và đặc trưng.
* Bước 4: Các track và các detection chưa được liên kết sẽ được đưa đến 1 lớp lọc tiếp theo. Sử dụng giải thuật Hungary giải bài toán phân công với ma trận chi phí IOU để liên kết lần 2
* Bước 5: Xử lý, phân loại các detection và các track
* Bước 6: Sử dụng Kalman filter để hiệu chỉnh lại giá trị của track từ những detection đã được liên kết với track và khởi tạo các track mới.

Với chiến lược liên kết cũng như sử dụng những độ đo phù hợp, Deep SORT đã cải thiện được vấn đề của SORT. Lượng switches ID giảm từ 1423 xuống còn 781, tức giảm 45% đồng thời giảm các lỗi liên quan đến đối tượng bị che khuất hoặc biến mất 1 thời gian. Dù tốc độ xử lỉ có giảm nhẹ, Deep SORT vẫn đảm bảo tốc độ xấp xỉ thời gian thực (realtime) nếu sử dụng GPU.

## 