Nhận diện và theo vết các phương tiện giao thông sử dụng YOLOv4 và DeepSORT

Trần Thanh Dương
duong.jt.19@gmail.com

Source code
https://github.com/duongttr/vehicles-counting-yolov4-deepsort

TÓM TẮT

Bài báo cáo tập trung giải quyết vấn đề nhận diện và kiểm soát lưu lượng giao thông tại Việt Nam thông qua 2 bước: object detection (nhận diện vật thể), object tracking (theo vết vật thể). Tập dữ liệu được sử dụng trong bài toán này là hình ảnh từ các camera đường phố của Thành phố Hồ Chí Minh và một số hình ảnh từ các nguồn khác có góc chụp tương tự, được chụp vào các thời điểm khác nhau và chia ra thành 4 nhóm: xe máy (motorbike), xe hơi (car), xe buýt (bus), xe tải (truck). Đối với nhận diện vật thể, mô hình được sử dụng là YOLOv4 – mô hình CNN cho việc phát hiện, nhận diện và phân loại đối tượng thời gian thực. Điểm ***mAP@0.5*** *(mean-average-precision) của mô hình này sau huấn luyện trên tập dữ liệu đạt* ***0.914660*** *(91.47%). Đối với theo vết vật thể, bài toán sử dụng thuật toán DeepSORT.*

I. GIỚI THIỆU BÀI TOÁN

1. Lý do nghiên cứu

Kiểm soát lưu lượng giao thông luôn là một trong những vấn đề vô cùng phức tạp và khó giải quyết đối với nhiều quốc gia trên thế giới, đặc biệt là giao thông mang yếu tố tự phát như ở Việt Nam vì số lượng phương tiện tham gia giao thông là vô cùng lớn. Với sự phát triển nhanh và mạnh mẽ của trí tuệ nhân tạo, nhiều phương án đã được đặt ra nhằm điều phối và cải thiện tình trạng giao thông hiện nay.

2. Mục tiêu nghiên cứu

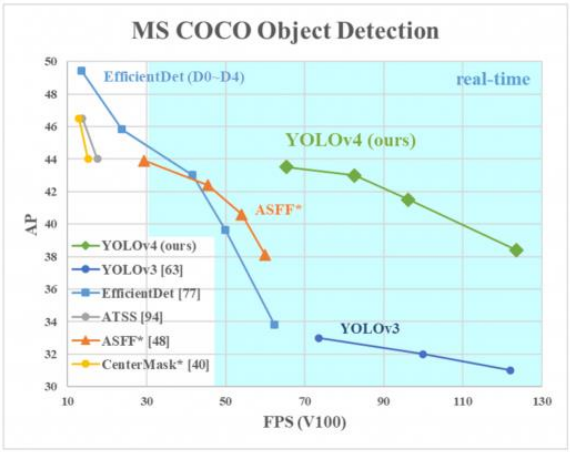
Xây dựng được mô hình có khả năng nhận diện nhanh và chính xác các vật thể, có thể ứng dụng mô hình để nhận diện theo thời gian thực trên các camera đường phố. Ngoài ra việc thiết kế thuật toán để có thể theo dấu vật thể và đếm vật thể ở mỗi camera cũng phải hiệu quả. YOLOv4 và DeepSORT là hai phương pháp có thể đáp ứng được yêu cầu và mục tiêu của bài toán.

II. LÝ THUYẾT NỀN TẢNG

1. Object Detection

Hiệu năng của các mô hình hiện đại

Bài toán đầu tiên cần giải quyết là định vị và phân biệt được các vật thể trong một bức ảnh hay còn gọi với tên **Object Detection**. Với mỗi bức ảnh đầu vào, ta sẽ tìm được các bounding box, label và phần trăm dự đoán chính xác tương ứng của từng vật thể.
Một số những mô hình có thể giải quyết bài toán này như **EfficientDet**[1] được công bố bởi nhóm tác giả **Mingxing Tan, Ruoming Pang, Quoc V. Le** (2020), phiên bản tiền nhiệm **YOLOv3**[2] được công bố bởi 2 tác giả **Joseph Redmon** và **Ali Farhadi** (2018).
YOLOv4[3] được công bố vào năm 2020 bởi **Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao** sau khi tác giả của YOLOv3 công bố sẽ không tiếp tục nghiên cứu để cập nhật những phiên bản tiếp theo của YOLO.



Hình ảnh so sánh dự đoán của YOLOv4 với các mô hình nhận diện vật thể hiện đại khác.

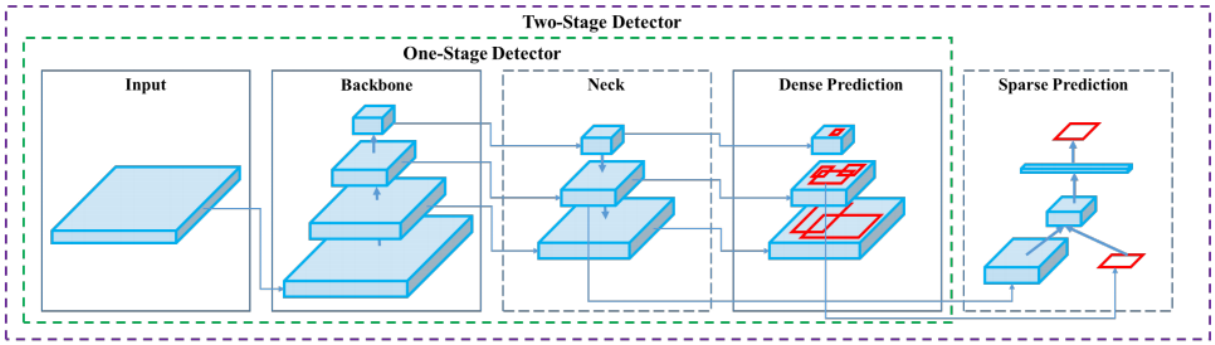
YOLOv4 chạy nhanh hơn gấp hai lần so với EfficientDet khi so sánh hiệu năng. Cải thiện điểm AP và FPS (chạy trên GPU V100) của YOLOv3 lên 10% và 12%. (trích dẫn từ paper của YOLOv4)

Cấu trúc của YOLOv4

Cấu trúc của YOLOv4 được tác giả chia làm bốn phần:

- Backbone (xương sống).
- Neck (cổ).
- Dense prediction (dự đoán dày đặc) - sử dụng các one-stage-detection như YOLO hoặc SSD.
- Sparse Prediction (dự đoán thưa thớt) – sử dụng các two-stage-detection như RCNN.

[1] *https://arxiv.org/abs/1911.09070*
[2] *https://arxiv.org/abs/1804.02767*
[3] *https://arxiv.org/abs/2004.10934v1*



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

Cấu trúc YOLOv4

Backbone – trích xuất đặc trưng hình ảnh

Backbone cho mô hình nhận dạng vật thể thường sử dụng các mô hình được đào tạo trước (pre-trained model). Tác giả đã xem xét sử dụng các loại Backbone sau: **CSPResNext50, CSPDarknet53, EfficientNetB3**

Mặc dù **EfficientNet**[4] (Mingxing Tan, Quoc V. Le - 2019) được thiết kế để chủ yếu nghiên cứu vấn đề mở rộng mô hình của mạng nơ-ron tích chập và vượt trội hơn các mạng khác có cùng kích thước về phân loại hình ảnh. Tuy nhiên tác giả của YOLOv4 cho rằng các mạng khác có khả năng hoạt động tốt hơn, cụ thể là tác giả đã chọn CSPDarknet53 làm Backbone cho mô hình.

Theo tác giả, CSPDarknet53 có độ chính xác cao hơn ResNet trong tác vụ nhận diện vật thể, mặc dù ResNet có độ chính xác cao hơn trong tác vụ phân loại vật thể, hạn chế này có thể được cải thiện nhờ hàm activation Mish[5].

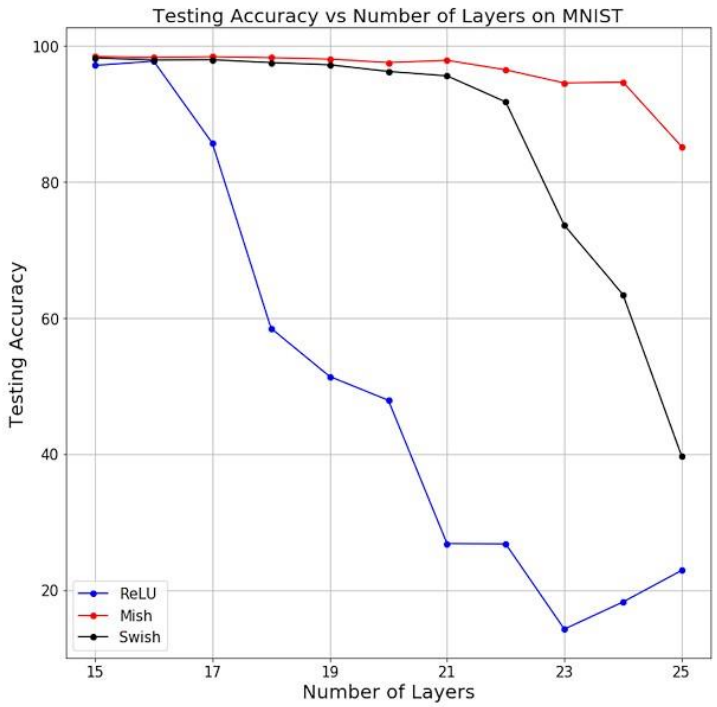
$$Mish(x) = x \tanh(\ln(1 + e^x))$$

Một số thuộc tính của Mish:

- Không có cận trên
- Có cận dưới
- Không đơn điệu, giữ lại một phần nhỏ negative gradient cho phép model học tốt hơn.
- Liên tục: Mish có đạo hàm bậc 1 tại mọi điểm thuộc miền giá trị.

Vì Mish trông "smooth" hơn các hàm kích hoạt khác, cũng như trả về negative gradient nhỏ nên giúp truyền tải thông tin xuống lớp sâu hơn trong mạng dễ dàng hơn.

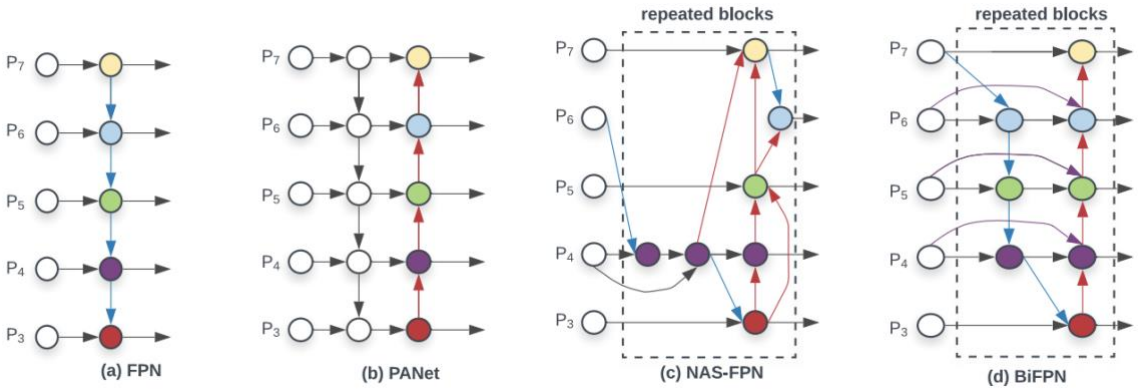
[4] https://arxiv.org/abs/1905.11946v5
[5] https://arxiv.org/abs/1908.08681



Neck – tổng hợp đặc trưng hình ảnh

Phần Neck có vai trò kết hợp các đặc trưng đã được huấn luyện ở quá trình trích xuất đặc trưng (backbone) và quá trình nhận dạng (dense prediction).
Với mỗi lần thực hiện detect với các kích thước ảnh rescale khác nhau tác giả đã thêm các luồng đi từ dưới lên và các luồng đi từ trên xuống vào cùng nhau theo từng hoặc được nối với nhau trước khi đưa vào head (phần đầu), từ đó lớp nhận dạng sẽ chứa thông tin phong phú hơn.

Tác giả của YOLOv4 đã cho phép tùy biến cấu trúc phần Neck mà chúng ta muốn sử dụng như là: FPN, PAN, NAS-FPN, BiFPN, ASFF, SFAM, SSP.

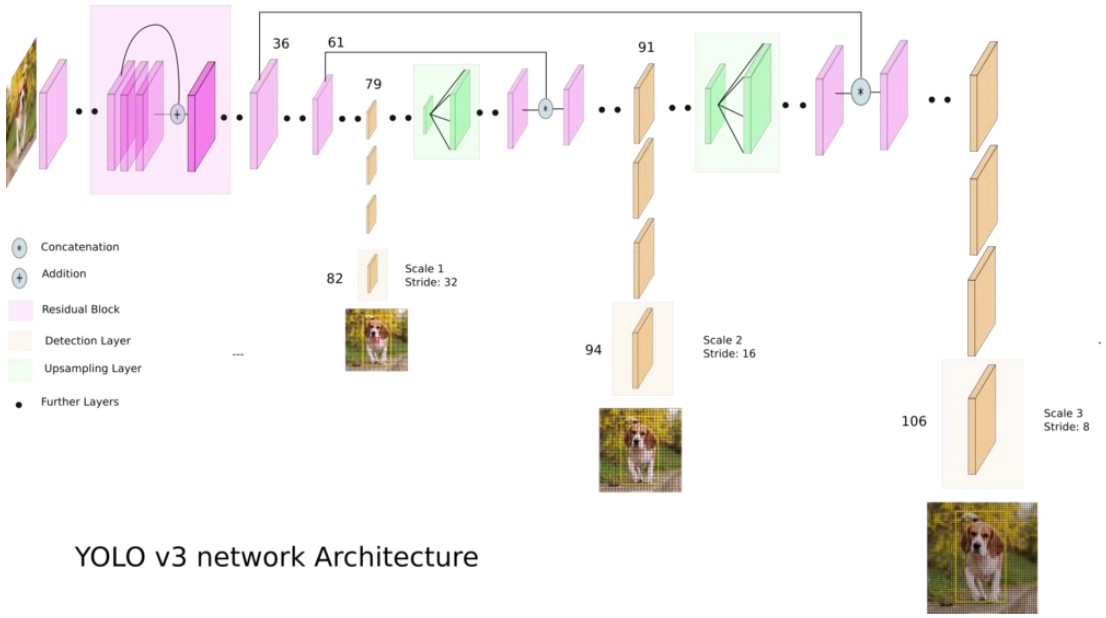


Cấu trúc các phần Neck được sử dụng

Mỗi nút P_i ở trên đại diện cho một lớp thuộc tính của CSPDarknet53 ở phần Backbone.

Head – phần dự đoán

Kể từ phần này trở đi, cấu trúc phần Head của YOLOv4 được kế thừa lại từ cấu trúc phần Head của YOLOv3.



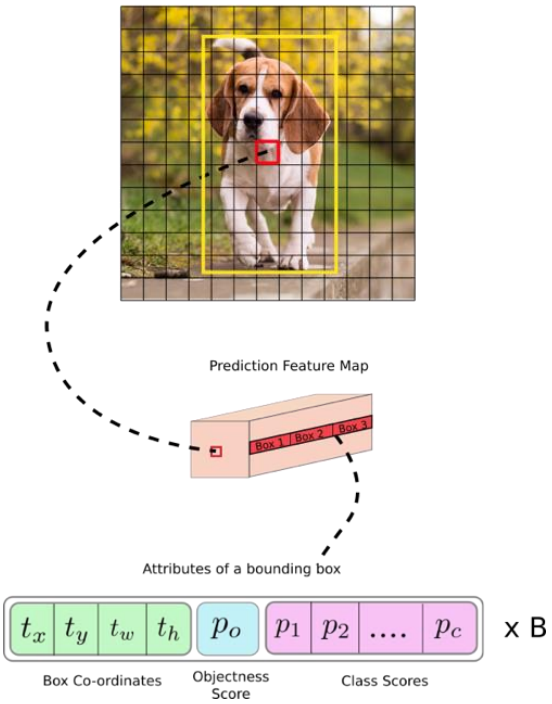
Kiến trúc YOLOv3

Phần Head xử lý phần nhận diện vật thể bằng cách "**downsampling**" hình ảnh thành 3 kích thước khác nhau:

- Kích thước 13x13, lớp chập thứ 82: Tìm các vật thể có kích thước lớn trong hình.
- Kích thước 26x26, lớp chập thứ 94: Tìm các vật thể có kích thước trung bình trong hình .
- Kích thước 52x52, lớp chập thứ 106: Tìm các vật thể có kích thước bé trong hình.

Kết quả đầu ra sẽ là một Tensor có kích thước $S \times S \times K$, trong đó:

- S lần lượt được gán với các giá trị: $13, 26, 52$.
- $K = B \times (5 + C)$, B là số lượng bounding box, C là số lớp vật thể.
- Mỗi bounding box sẽ chứa 5 giá trị: toạ độ (x, y), kích thước chiều dài và rộng của bounding box (w, h) và xác suất ô lưới có chứa vật thể p_0
- C giá trị tiếp theo là xác suất mà ô lưới đó chứa các vật thể cụ thể $p_1, p_2, p_3 \dots, p_C$



Trong quá trình huấn luyện trên mỗi kích thước, mô hình sẽ tạo ra các anchor box trong mỗi ô, bản chất là bounding box nhưng được tạo sẵn và tính độ sai sót giữa ground-truth box và anchor box, sau đó điều chỉnh các giá trị x, y, w, h để học được các đặc điểm của vật thể.

YOLOv4 sử dụng hàm IoU-loss (Intersect Over Union) để tính độ sai sót và làm cơ sở để cập nhật hệ số của mô hình:

$$IoU = \frac{|B \cap B^{GT}|}{|B \cup B^{GT}|}$$

$$\mathcal{L}_{IoU} = 1 - IoU = 1 - \frac{|B \cap B^{GT}|}{|B \cup B^{GT}|}$$

Trong đó, B và B^{GT} lần lượt là diện tích của anchor box và ground-truth box.

Tuy nhiên nếu xét trường hợp 2 box trên không chồng lên nhau (overlapping) thì sẽ gây khó khăn cho việc tính toán, do đó GIoU-loss (Generalized IoU loss) có thể giải quyết được điều này bằng cách thêm vào một thành phần C, box có diện tích nhỏ nhất mà chứa cả prediction box và ground-truth box:

$$\mathcal{L}_{GIoU} = 1 - IoU + \frac{|C - B \cup B^{GT}|}{|C|}$$

GIoU sẽ có khuynh hướng mở rộng prediction box cho tới khi nó overlapping với ground-truth box, sau đó mới co lại để giảm IoU. Để giải quyết vấn đề này, DIoU ràng buộc khoảng cách giữa tâm của prediction box và ground-truth box:

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{||B - B^{GT}||_2^2}{C^2}$$

Cuối cùng CIoU-loss (Complete IoU loss) thêm vào tham số duy trì tỷ lệ các box.

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{||B - B^{GT}||_2^2}{C^2} + \alpha v$$

Hàm loss binary cross-entropy được sử dụng để đánh giá tham số p_0:

$$\mathcal{L}_{BCE}(Y, P_0) = -\frac{1}{N}\sum_{i}^{N} y_{i0}\log(p_{i0}) + (1 - y_{i0})\log(1 - p_{i0})$$

Hàm loss categorical cross-entropy được sử dụng để đánh giá các tham số $p_1,\ p_2, \dots,\ p_C$:

$$\mathcal{L}_{CCE}(Y, P_{1..C}) = -\frac{1}{N}\sum_{i}^{N}\sum_{j=1}^{C} y_{ij}\log\left(\frac{e^{p_{ij}}}{\sum_{k=1}^{C} e^{p_{ik}}}\right)$$

2. Object Tracking

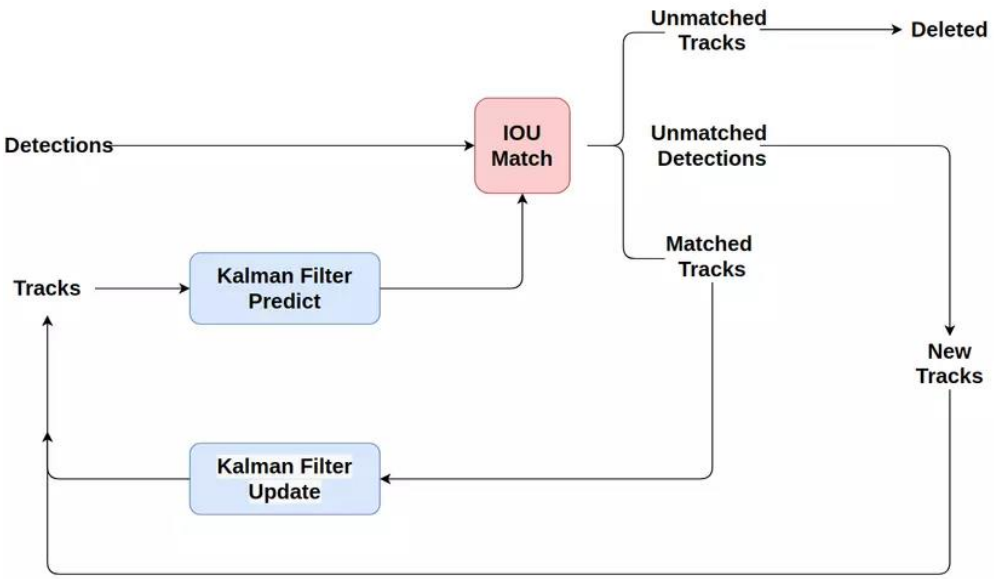
Trong real-time object detection, các vật thể được nhận diện trong mỗi khung hình của video là độc lập với nhau, nghĩa là mô hình nhận diện vật thể không thể xác định được sự liên quan của vật thể giữa các khung hình. Có môt thuật toán vô cùng hiệu quả để giải quyết bài toán này là **DeepSORT**[6] (Deep Simple Online Realtime Object Tracking), được công bố bởi nhóm tác giả **Nicolai Wojke, Alex Bewley, Dietrich Paulus** (2017)**.**

SORT (Simple Online Realtime Object Tracking)

Trước khi DeepSORT ra đời, SORT (2016) được đề xuất làm giải pháp cho object tracking, đồng thời giải quyết cả 2 vấn đề mutiple object tracking và realtime object tracking.

SORT tập trung vào vấn đề liên kết giữa các detection và track sau khi đã detect được từ frame, do đó, phần object detection có thể là bất cứ mô hình detector nào hiện nay. Hai thuật toán cốt lõi của SORT là Kalman Filter và giải thuật Hungary.

[6] https://arxiv.org/abs/1703.07402



Nhưng SORT vẫn tồn tại những vấn đề chưa thể giải quyết:

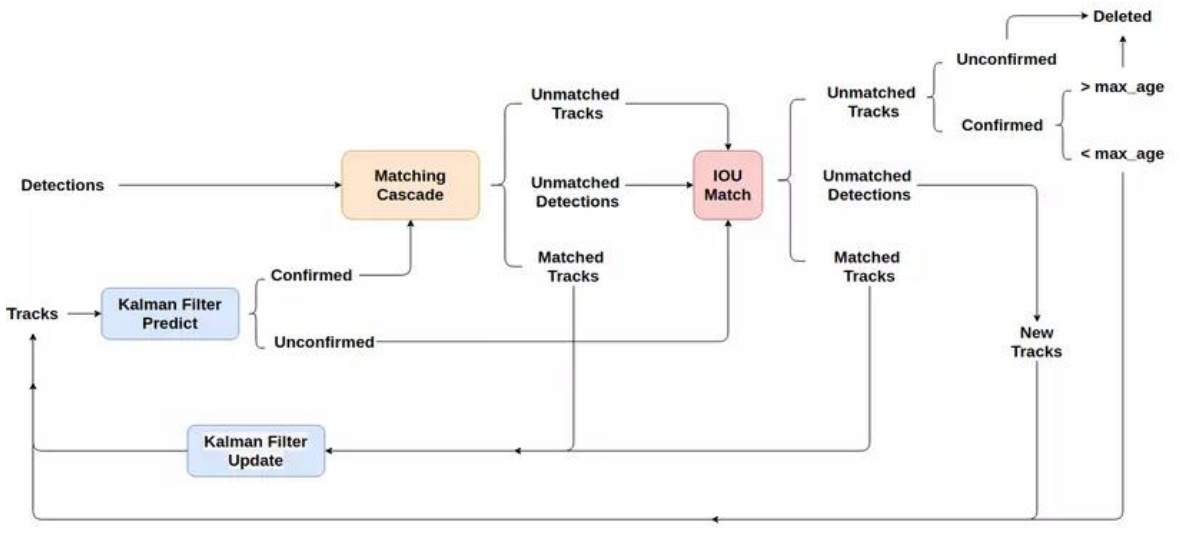
- Giả định tuyến tính : SORT đang sử dụng Linear Kalman Filter trong thuật toán cốt lõi, điều này trong thực tế là chưa phù hợp. Để cải thiện vấn đề này, chúng ta cần quan tâm đến các Kalman Filter phức tạp hơn, như Extended Kalman filter, Unscented Kalman filter, ...
- ID Switches : Đây là vấn đề lớn nhất của SORT hiện tại. Do việc liên kết giữa detection và track trong SORT chỉ đơn giản dựa trên độ đo IOU (tức SORT chỉ quan tâm đến hình dạng của đối tượng), điều này gây ra hiện tượng số lượng ID Switches của 1 đối tượng là vô cùng lớn khi đối tượng bị che khuất, khi quỹ đạo trùng lặp,...

Ý tưởng của DeepSORT

Trong multiple object tracking, đặc biệt là đối với lớp thuật toán tracking-by-detection, có 2 yếu tố chính ảnh hưởng trực tiếp đến performance của việc theo dõi:

- **Data Association:** Quan tâm đến vấn đề liên kết dữ liệu, cụ thể là tiêu chí để xét và đánh giá nhằm liên kết một detection mới với các track đã được lưu trữ sẵn
- **Track Life Cycle Management:** Quan tâm đến việc quản lý vòng đời của một track đã được lưu trữ, bao gồm, khi nào thì khởi tạo track, khi nào thì ngưng theo dõi và xóa track ra khỏi bộ nhớ,...

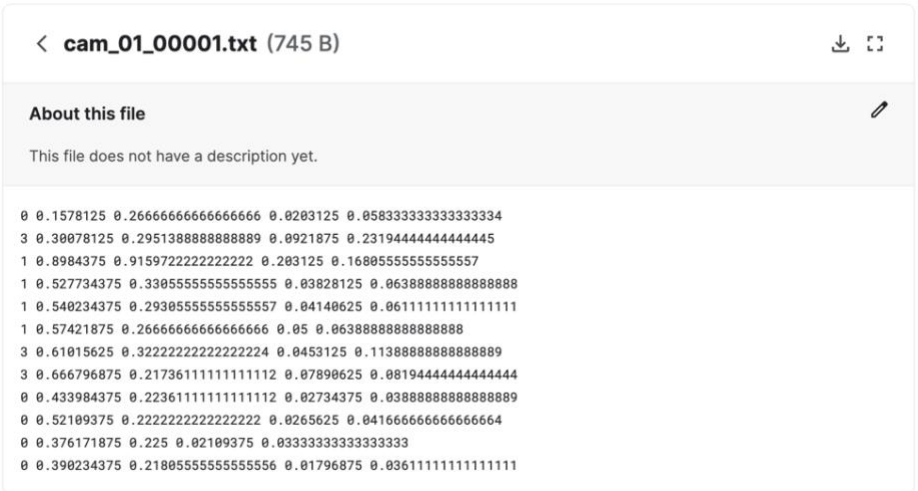
Trong DeepSORT, nhóm tác giả giải quyết vấn đề data association dựa trên thuật toán Hungary (tương tự như SORT), tuy nhiên, việc liên kết không chỉ dựa trên IOU mà còn quan tâm đến các yếu tố khác: khoảng cách của detection và track (xét tính tương quan trong không gian vector) và khoảng cách cosine giữa 2 vector đặc trưng được trích xuất từ detection và track - 2 vector đặc trưng của cùng 1 đối tượng sẽ giống nhau hơn là đặc trưng của 2 đối tượng khác nhau.



III. TẬP DỮ LIỆU

1. Cấu trúc của tập dữ liệu

Nguồn dữ liệu[7] được tổng hợp từ camera đường phố tại địa bàn thành phố Hồ Chí Minh và một số nơi khác, các vật thể trích xuất từ hình ảnh đều có đặc điểm chung là được chụp từ vị trí cao.
Nguồn dữ liệu gốc có cấu trúc không đúng với format của YOLO Darknet, nên sau khi chuyển đổi format và loại bỏ những hình ảnh không có bounding box, tập dữ liệu được chia ra thành 2 thư mục chính: daytime (trời sáng) và nighttime (trời tối). Đối với mỗi hình ảnh sẽ có một tệp text cùng tên đi kèm, trong tệp text này chứa thông tin về toạ độ của các bounding box trong hình ảnh đó:



Định dạng của các bounding box trong tệp text như sau: $[class - id]$ $[x]$ $[y]$ $[width]$ $[height]$
Trong đó:

- $[class - id]$: là số nguyên nằm trong khoảng [0; classes-1].

[7] https://www.kaggle.com/datasets/duongtran1909/vietnamese-vehicles-dataset

- $[x], [y]$: lần lượt là tạo độ **điểm chính giữa** của bounding box.
- $[width], [height]$: chiều dài và rộng của của bounding box.

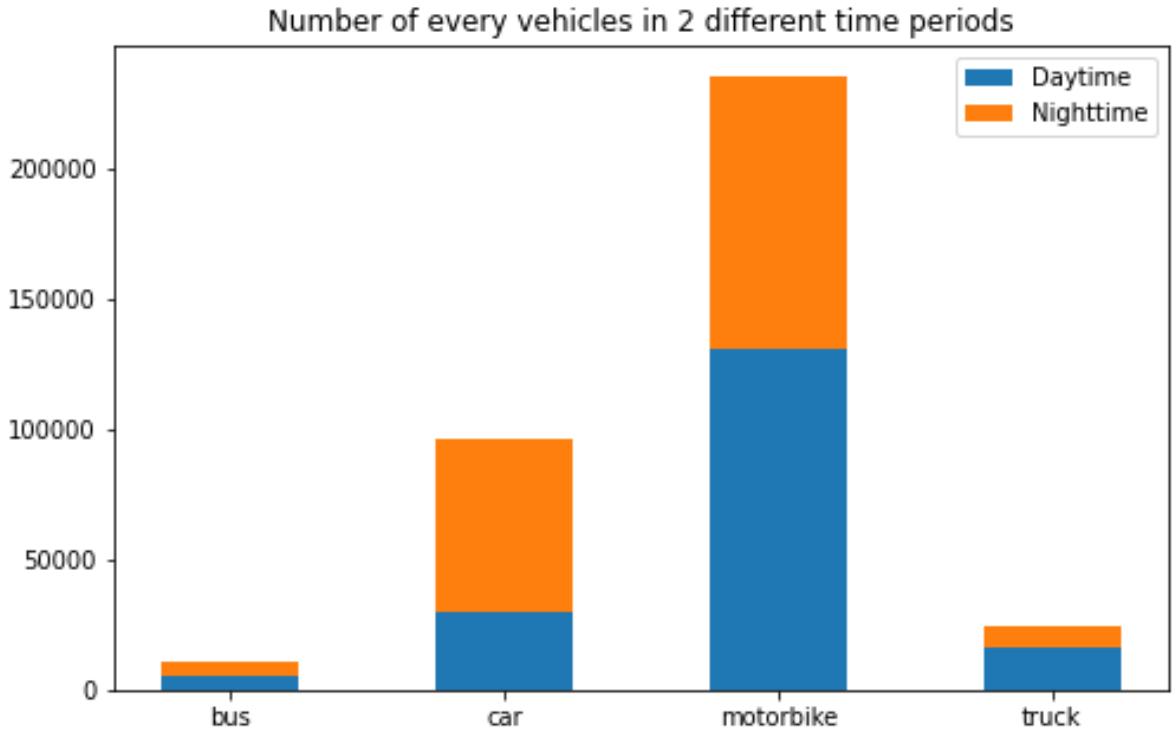
Cần lưu ý rằng $[x], [y], [width], [height]$ là những giá trị sau khi đã được chia tỷ lệ với chiều dài thực và chiều rộng thực của hình ảnh, hay nói cách khác:

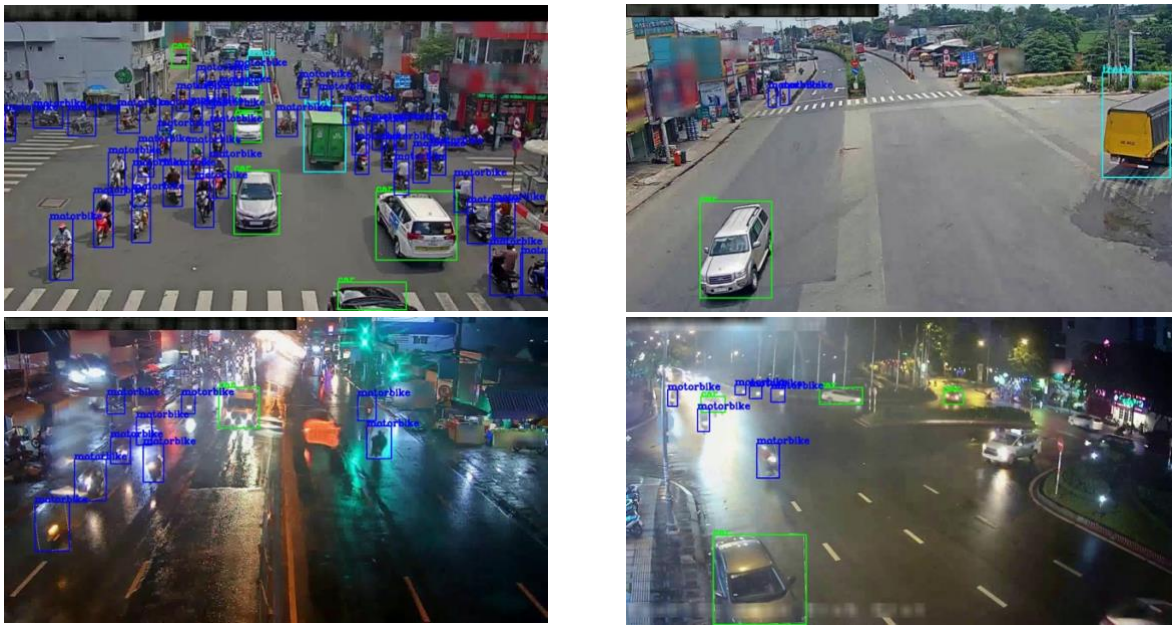
$$[x] = \frac{absolute\ x}{image_{width}}, [y] = \frac{absolute\ y}{image_{height}}$$

$$[width] = \frac{bbox_{width}}{image_{width}}, [height] = \frac{bbox_{height}}{image_{height}}$$

2. Phân tích dữ liệu

	Motorbike	Car	Bus	Truck	Total
Day time	131058	30114	5381	16162	182715
Night time	104435	65692	5806	8273	184206
Total	235493	95806	11187	24435	366921





IV. PHƯƠNG PHÁP NGHIÊN CỨU

1. Phương pháp tăng sinh dữ liệu (data augmentation)

CutMix data augmentation – Cắt và trộn

Dựa trên ý tưởng của CutOut: loại bỏ một vùng trên mỗi ảnh để model không thể overfit một feature đặc biệt nào đó trên tập training.Tuy nhiên, vùng ảnh bị loại bỏ được điền vào các giá trị 0. CutMix thay vùng ảnh bị loại bỏ bằng một phần ảnh của ảnh khác trong dataset. Vùng bị thay thế này sẽ bắt buộc bộ object detector học với nhiều loại đặc điểm (features) hơn.

Mosaic data augmentation – "Khảm" dữ liệu

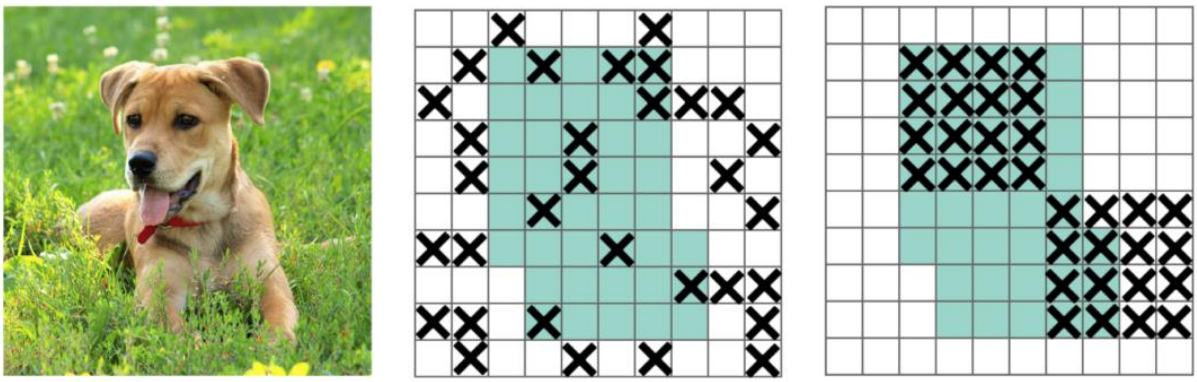
Thay vì mỗi ảnh đầu vào là sự kết hợp của 2 ảnh thì Mosaic sử dụng kết hợp 4 ảnh. Việc này giúp cho bối cảnh của ảnh phong phú hơn.

2. Phương pháp xử lý dữ liệu

DropBlock regularization

Trong Dropout, giả thiết là các điểm gần nhau thường có đặc điểm giống nhau, do đó ta có thể loại bỏ các điểm này bằng cách set weight=0 tại một số vị trí trên feature map.

Trong DropBlock, các vị trí được chọn không phân bố ngẫu nhiên nữa mà tập trung thành các block.



Class label smoothing

Thay giá trị 1.0 thành 0.9 trong one-hot coding. Điều này giúp ta ngay cả khi đoán đúng class của một bức ảnh thì vẫn có loss. Do đó model sẽ phải điều chỉnh trọng số, giúp tránh việc overconfident vào kết quả dự đoán của mình, tránh bị overfitting.

3. Framework và mô hình object detection được sử dụng

Phương án xây dựng mô hình trong bài báo này sử dụng neural network framework **Darknet**[8], đây là framework phù hợp cho xây dựng mô hình real-time object detection. Ngoài ra, Darknet có thể xây dựng mô hình trên cả CPU và GPU, hỗ trợ tăng tốc độ tính toán huấn luyện.

Mô hình CNN trong framework Darknet được định nghĩa trong tệp config[9] (.cfg). Sau đây là phần giải thích của một vài hyperparameter quan trọng trong mỗi layer:

Phần [net] – định nghĩa các hyperparameter

Hyperparameter cho training:

- $batch = 64$: số lượng hình ảnh được truyền qua neural network ở mỗi batch.
- $subdivisions = 16$: mỗi batch sẽ được chia nhỏ ra thành nhiều "block", những block này sẽ được chạy song song trên GPU.
- $width = 416, height = 416, channels = 3$: hình ảnh đầu vào sẽ được điều chỉnh về các kích thước cố định.

Hyperparameter cho optimizer:

- $momentum$: Ở mỗi lần cập nhật, các trọng số sẽ được cộng thêm một đại lượng $momentum * previous\_gradient$ (ý tưởng giống với GD with Momentum).
- $Decay$: giúp loại bỏ sự mất cân bằng trong tập dữ liệu.
- $learning\_rate$: tốc độ học, huấn luyện.
- $max\_batches$: số lần iteration tối đa của training.

[8] https://github.com/AlexeyAB/darknet
[9] https://github.com/duongttr/vehicles-counting-yolov4-deepsort/blob/main/utils_obj_detection/yolo_samples/yolov4-custom.cfg

- $Policy = steps, steps = 48000,54000, scales = .1, .1$: ở bước training thứ 48000 và 54000 learning_rate sẽ được nhận với các hệ số scales (0.1 và 0.1)

Hyperparameter cho data augmentation:

- $angle$: xoay hình ảnh trong quá trình training
- $saturation$: thay đổi độ rực màu của hình ảnh trong quá trình training.
- $exposure$: thay đổi độ sáng trong quá trình training.
- hue: thay đổi sắc thái của hình ảnh trong quá trình training.
- $mosaic$: sử dụng Mosaic data augmentation (đã được trình bày ở trên)

Lớp [convolutional]:

- $batch\_normalize$: sử dụng Batch Normalization
- $filters$: số lượng filters
- $size$: kích thước của filter (sizexsize)
- $stride$: bước nhảy của filter trên hình ảnh
- pad: thêm padding vào hình ảnh
- $activation$: hàm kích hoạt

Lớp [route]: Concatenation layer

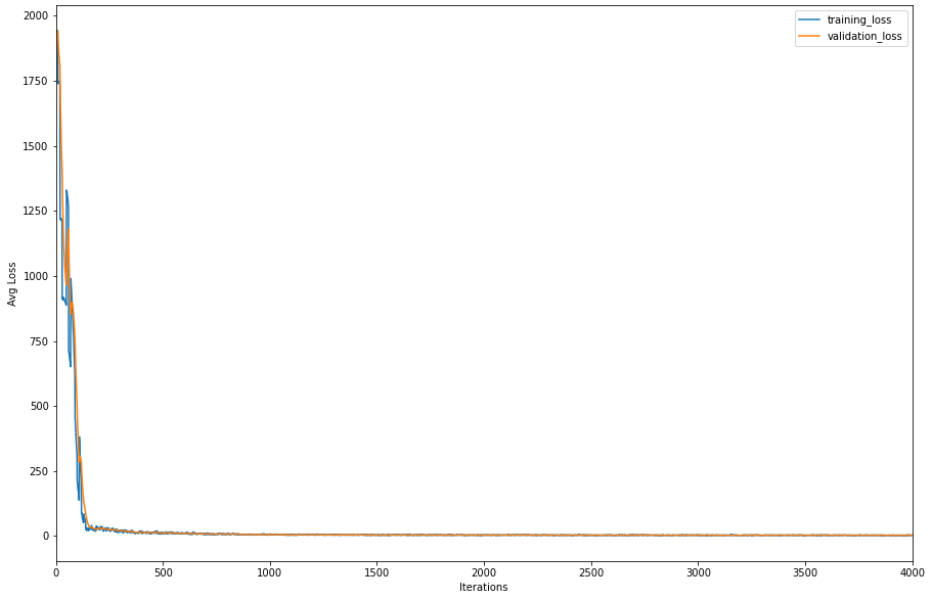
Lớp [shortcut]: Residual layer (ResNet)

Lớp [yolo]: Detection layer trong YOLOv4

- $mask$: index của anchor được sử dụng trong lớp [yolo] này
- anchors: khởi tạo các kích thước các anchor box.
- $classes$: số lượng class
- num: số lượng anchor
- $ignore\_thresh$: loại bỏ các detection bounding box trùng lập có IoU(detect, truth) < ignore_thresh
- $truth\_thresh$: điều chỉnh các detection bounding box trung lập có IoU(detect, truth) < truth_threshold
- iou_loss: sử dụng hàm CIoU-loss

V. KẾT QUẢ

Training loss và validation loss:



Đánh giá mô hình trên tập validation:

Label	Name	TP	FP	**AP@0.5**
d_motorbike	Daytime motorbike	12091	2328	92.09%
d_car	Daytime car	2673	570	91.65%
d_bus	Daytime bus	412	75	85.90%
d_truck	Daytime truck	1341	296	89.68%
n_motorbike	Nighttime motorbike	9241	1082	93.93%
n_car	Nighttime car	6047	582	98.22%
n_bus	Nighttime bus	420	78	90.14%
n_truck	Nighttime truck	743	164	90.12%
mAP@0.50				91.47 %

Total TP = 32968, Total FP = 5175, Total FN = 3303, average IoU = 70.21%

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 0.86$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 0.91$$

$$\text{F}_1 - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 0.89$$

VI. THẢO LUẬN NGHIÊN CỨU VÀ KHUYẾN NGHỊ

Để đánh giá về mức độ hiệu quả của mô hình thông qua bảng biểu đã thu được ở phía trên, cần hiểu được những metric được sử dụng để đánh giá, cụ thể là Precision, Recall, F1-Score, AP và mAP.

Precision được định nghĩa là tỉ lệ số điểm true positive (TP) trong những điểm được phân loại là positive (TP+FP). **Recall** là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN)
Như vậy, Precision cao đồng nghĩa với việc độ chính xác của các vật thể tìm được là cao. Recall cao đồng nghĩa với việc tỉ lệ bỏ sót các điểm thực sự positive thấp.
Có thể thấy Precision và Recall tương đối cao, lần lượt là 0.86 và 0.91. Nhưng cả 2 giá trị này vẫn chưa thể đánh giá được gì về một mô hình đã được huấn luyện bên trên.
Giả sử Precision là rất cao, nghĩa là có rất ít điểm negative bị lẫn vào kết quả, nhưng Precision vẫn chưa thể trả lời được câu hỏi liệu mô hình đã tìm được tất cả các điểm positive hay chưa. Nếu Recall là rất cao, nghĩa là mọi điểm positive đều được tìm thấy. Tuy nhiên đại lượng này lại không đo được liệu có bao nhiêu điểm negative bị lẫn vào trong đó. Vì thế, F1-Score được xem như là cách để đánh giá chất lượng của mô hình dựa vào Precision và Recall.
Có thể thấy **F1-Score = 0.89**, có thể đánh giá rằng mô hình hoạt động tốt.

Ngoài ra, việc đánh giá tính hiệu quả của mô hình trên từng class cụ thể cũng rất quan trọng, có thể làm được điều này thông qua metric **AP (Average Precision)** – là trung bình của các Precision tại từng hình ảnh mà kết quả đúng được trả về. Trong mô hình ở trên, metric AP được đánh giá với IoU threshold bằng 0.5, nghĩa là nếu IoU của vật thể lớn hơn hoặc bằng 0.5 thì vật thể đó được xếp vào TP, ngược lại là FP, khí hiệu là **AP@0.5**.Để tổng quát hoá độ hiệu quả của mô hình cho toàn bộ class, có thể tính trung bình các AP của tất cả các class, giá trị cuối cùng được gọi là **mAP (mean Average Precision)**.

Mô hình trên được đánh giá với **mAP@0.5** bằng **0.9147** hay **91.47%.** Từ đây hoàn toàn có thể kết luận được mức độ ổn định và phù hợp của mô hình trên là khá tốt.

Mặc dù các thảo luận cho ta những kết quả về mô hình là tương đối tốt, nhưng song song đó vẫn còn nhiều bất cập trong quá trình thu thập dữ liệu để huấn luyện mô hình, cụ thể là sự mất cân bằng tập dữ liệu (imblanced dataset). Nhìn vào tập dữ liệu có thể dễ dàng thấy được việc thu thập dữ liệu về xe máy là dễ dàng hơn rất nhiều (dữ liệu đã thu thập tận hơn 25000 dữ liệu) so với việc thu thập dữ liệu về các loại xe khác, dẫn đến việc dự đoán đôi khi bị kém hiệu quả trên tập thiểu số và bị thiên về nhóm đa số.
Có thể giải quyết vấn đề này thông qua một số cách như:

- Thu thập thêm dữ liệu của các tập thiểu số hoặc tăng sinh dữ liệu (data augmentation)
- Sử dụng metric phù hợp (Precision, Recall, F1-Score,…)
- Giảm số lượng của các tập đa số.
- “Phạt mô hình” nặng hơn trên các tập thiểu số.

VII. KẾT LUẬN

Bài báo cung cấp giải pháp quản lý giao thông thông qua bài toán Object Detection và Object Tracking. Xây dựng được mô hình thông qua việc sử dụng YOLOv4, một mô hình CNN vô cùng hiệu quả để giải quyết bài toán real-time object detection và ứng dụng thuật toán DeepSORT để giải quyết bài toán theo vết các phương tiện giao thông dựa trên hai thuật toán cốt lõi của SORT là Kalman Filter và giải thuật Hungary.

Nhìn chung phương pháp được đề ra vẫn còn gặp vấn đề khó áp dụng đại trà ngoài thực tế khi mô hình YOLOv4 tương đối nặng, do đó chi phí để tích hợp công nghệ này vào các camera đường phố là không hề rẻ. Do đó cần tìm kiếm phương pháp tối giản hoá độ phức tạp mô hình bằng cách sử dụng những mô hình đơn giản hơn, nhưng sẽ đánh đổi về mặt chính xác của kết quả.

TÀI LIỆU THAM KHẢO

EfficientDet - Mingxing Tan, Ruoming Pang, Quoc V. Le (2020)

YOLOv3 - Joseph Redmon và Ali Farhadi (2018)

YOLOv4 - Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao (2020)

EfficientNet - Mingxing Tan, Quoc V. Le (2019)

DeepSORT - Nicolai Wojke, Alex Bewley, Dietrich Paulus (2017)

Darknet - Alexey Bochkovskiy (2020)

You Only Look Once - Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2015)

Mish - Diganta Misra (2019)