ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ VIỄN THÔNG
BỘ MÔN MÁY TÍNH – HỆ THỐNG NHÚNG

NGUYỄN TRẦN QUỐC KHÔI

Đề tài đồ án tốt nghiệp:

PHÁT TRIỂN HỆ THỐNG NHẬN DIỆN BIỂN BÁO GIAO THÔNG TRÊN THIẾT BỊ NHÚNG

Chuyên ngành Máy Tính - Hệ Thống Nhúng

TP. Hồ Chí Minh, tháng 7 năm 2025

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG
BỘ MÔN MÁY TÍNH - HỆ THỐNG NHÚNG

NGUYỄN TRẦN QUỐC KHÔI
20200237

Đề tài:

PHÁT TRIỂN HỆ THỐNG NHẬN DIỆN BIỂN BÁO GIAO THÔNG TRÊN THIẾT BỊ NHÚNG

DEVELOPMENT OF TRAFFIC SIGN RECOGNITION SYSTEM ON EMBEDDED DEVICE

ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN
NGÀNH KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG
CHUYÊN NGÀNH MÁY TÍNH - HỆ THỐNG NHÚNG

NGƯỜI HƯỚNG DẪN KHOA HỌC
ThS. TRẦN TUẤN KIỆT

TP. Hồ Chí Minh, tháng 7 năm 2025

Lời Cảm ơn

Em xin gửi lời cảm ơn sâu sắc đến Ths.Trần Tuấn Kiệt – người hướng dẫn khoa học tận tâm và nhiệt huyết trong suốt quá trình thực hiện dự án. Thầy đã dành rất nhiều thời gian chỉ dẫn, định hướng phương pháp nghiên cứu cũng như hỗ trợ kỹ thuật để em hoàn thiện tốt nhất dự án này. Những ý kiến đóng góp quý báu của thầy đã giúp em nhận thức rõ hơn về các vấn đề chuyên môn và cải thiện chất lượng nghiên cứu một cách đáng kể. Sự tận tình và tinh thần trách nhiệm của thầy đã truyền cảm hứng và động viên em vượt qua nhiều khó khăn trong quá trình làm việc. Em rất biết ơn vì được thầy tận tâm hướng dẫn, chỉ dạy không chỉ kiến thức chuyên ngành mà còn cách làm việc khoa học nghiêm túc. Em xin chân thành cảm ơn thầy đã đồng hành, giúp đỡ và tạo điều kiện thuận lợi để dự án được hoàn thành tốt đẹp.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến các anh chị trong phòng thí nghiệm đã hỗ trợ nhiệt tình về thiết bị và môi trường làm việc trong suốt quá trình thực hiện dự án. Sự giúp đỡ kịp thời và tạo điều kiện thuận lợi từ các anh chị đã góp phần quan trọng giúp dự án được tiến hành suôn sẻ và hiệu quả. Đồng thời, em cũng xin trân trọng cảm ơn Ths.Đặng Tấn Phát đã tận tình giải đáp những thắc mắc chuyên môn phát sinh trong quá trình thực nghiệm. Những ý kiến và hướng dẫn quý báu của thầy đã giúp em khắc phục nhiều khó khăn và nâng cao chất lượng công trình. Em rất trân trọng và biết ơn sự hỗ trợ đáng giá này.

MỤC LỤC

DANH MỤC CÁC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu , chữ viết tắt	Chữ viết đầy đủ
ADAS	Advanced Driver Assistance Systems
ADS	Autonomous Driving Systems
AP	Average Precision
CNN	Convolution Neural Netwoks
CSPNet	Cross Stage Partial Network
DFL	Distribution Focal Loss
DNN	Deep Neural Networks
EDA	Exploratory Data Analysis
FCL	Fully Connected Layers
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Networks
FPS	Frame Per Second
IOU	Intersection Over Union
mAP	Mean Average Precision
PAN	Path Aggregation Networks
QSP	Quantize Stage Partial
SOTA	State Of The Art
SPP	Spatial Pyramid Pooling
SSD	Single Shot Detector
TP	True Positive
YOLO	You Only Look Once
YOLO-NAS	Yolo-Neural Architecture Search

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BẢNG

TÓM TẮT

Dự án phát hiện biển báo giao thông tại Việt Nam sử dụng mô hình YOLO NAS SMALL, một kiến trúc mạng nơ-ron tích chập hiện đại tối ưu cho các thiết bị nhúng. Mục tiêu là xây dựng hệ thống nhận dạng biển báo giao thông chính xác và nhanh chóng trong thời gian thực, nhằm hỗ trợ người lái xe và nâng cao an toàn giao thông. Dữ liệu huấn luyện gồm hơn 3.900 ảnh biển báo với 41 loại biển báo khác nhau, được thu thập và xử lý đa dạng trong các điều kiện giao thông và thời tiết thực tế. Mô hình được huấn luyện và tinh chỉnh trên nền tảng Kaggle, sau đó tối ưu hóa bằng TensorRT để triển khai trên máy tính nhúng NVIDIA Jetson Nano B01 với tài nguyên giới hạn. Kết quả mô hình đạt độ chính xác cao (mAP trên 90%) với tốc độ xử lý khoảng 8.6 FPS trên Jetson Nano. Hệ thống có tiềm năng ứng dụng rộng rãi trong việc giám sát và cảnh báo giao thông, đồng thời là bước đệm quan trọng cho việc phát triển các hệ thống lái xe tự động tại Việt Nam.

CHƯƠNG 1 : GIỚI THIỆU

Trong những năm gần đây, sự gia tăng dân số kéo theo lượng phương tiện tham gia giao thông ngày càng cao. Điều này dẫn đến áp lực lớn lên hệ thống giao thông và làm gia tăng nguy cơ xảy ra tai nạn. Các nguyên nhân phổ biến bao gồm mệt mỏi, buồn ngủ, sử dụng rượu bia, mất tập trung và không tuân thủ luật lệ giao thông. Việc lựa chọn tốc độ không phù hợp, không nhường đường ưu tiên và không kịp thời phát hiện hoặc tuân thủ biển báo giao thông cũng là những yếu tố gây ra các tai nạn nghiêm trọng. Giao thông hiện đại đòi hỏi người điều khiển phương tiện có phản xạ nhanh, khả năng phán đoán chính xác và duy trì sự tập trung liên tục.

Để giảm thiểu tai nạn và nâng cao chất lượng cuộc sống, việc nghiên cứu các hệ thống hỗ trợ lái xe đã trở thành xu hướng phát triển lâu dài. Phương tiện giao thông không còn chỉ đơn thuần là công cụ di chuyển, mà còn là nền tảng để tích hợp các công nghệ thông minh. Hệ thống hỗ trợ lái xe tiên tiến (advanced driver assistance systems - ADAS) ra đời nhằm hỗ trợ người lái nhận diện và xử lý kịp thời các tình huống nguy hiểm. Những công nghệ này giúp cải thiện đáng kể độ an toàn và sự thoải mái khi điều khiển phương tiện. ADAS được xem là bước đệm quan trọng cho sự phát triển của xe tự hành trong tương lai.

Việc ứng dụng các công nghệ như thị giác máy tính,trí tuệ nhân tạo , cảm biến radar, LIDAR và camera đã mở ra nhiều giải pháp giám sát giao thông hiệu quả. Các hệ thống này có khả năng thu thập và xử lý dữ liệu môi trường xung quanh xe trong thời gian thực. Những công nghệ này cho phép thực hiện hiệu quả các tác vụ như phát hiện, định vị, theo dõi và nhận dạng vật thể trong điều kiện môi trường đa dạng. Với sự tiến bộ không ngừng trong khả năng tính toán và cảm biến, các giải pháp hỗ trợ lái xe ngày càng trở nên chính xác, thông minh và dễ tiếp cận hơn.

CHƯƠNG 2 : ỨNG DỤNG MẠNG NƠ-RON TÍCH CHẬP (CNN) TRONG NGÀNH CÔNG NGHIỆP Ô TÔ

Khi công nghệ ngày càng phát triển, kỳ vọng của người mua xe cũng thay đổi, với nhu cầu ngày càng cao về những phương tiện có khả năng tích hợp liền mạch với cuộc sống số, đồng thời ưu tiên yếu tố an toàn. Các hãng sản xuất ô tô đang trải qua một sự chuyển dịch mô hình – từ việc tập trung vào công suất động cơ sang ưu tiên các công nghệ kỹ thuật số, đặc biệt là trong việc phát triển xe tự lái, hay còn gọi là hệ thống lái xe tự động (ADS – Autonomous Driving Systems). Trong những năm tới, thế hệ ADS tiếp theo được kỳ vọng sẽ mang lại trải nghiệm lái xe tự động ở mức độ cao hơn. Để đạt được điều này, cần phải tích hợp nhiều công nghệ khác nhau vào một hệ thống thống nhất, có khả năng giao tiếp, phối hợp và tái hiện các chức năng của con người trong hầu hết các tình huống lái xe. Theo đề xuất của Intel , để đạt đến mức độ tinh vi này, hệ thống cần tích hợp nhiều loại cảm biến như radar, LIDAR, GPS (Hệ thống Định vị Toàn cầu) và camera tiên tiến. Các cảm biến này có nhiệm vụ thu thập dữ liệu quan trọng với tốc độ khoảng 1 GB/giây và xử lý theo thời gian thực nhằm đưa ra các quyết định lái xe chính xác. Mặc dù LIDAR có những lợi thế như độ chính xác cao và khả năng tạo bản đồ 3D, nhưng chi phí quá cao khiến nó khó có thể được ứng dụng rộng rãi trong thương mại. Do đó, một số công ty như Tesla và Mobileye đã lựa chọn camera quang học và radar làm cảm biến chính trong các hệ thống nhận diện dựa trên hình ảnh của ADS. [1]

Trong bối cảnh này, mạng nơ-ron tích chập (CNN), một kỹ thuật xử lý hình ảnh, đã chứng minh hiệu quả vượt trội và được các hãng ô tô ứng dụng rộng rãi trong các mô-đun phát hiện vật thể của xe. [1]

Tuy nhiên, mặc dù đã đạt được nhiều tiến bộ đáng kể trong việc phát hiện vật thể bằng CNN cho nhiều ứng dụng khác nhau, việc triển khai mạng này trong hệ thống lái xe tự động vẫn là một thách thức lớn. Nguyên nhân chính là do ADS đòi hỏi phải xử lý và ra quyết định an toàn trong thời gian thực

với độ trễ cực kỳ thấp. Do đó, việc phát triển một thiết kế ADS hoàn chỉnh từ đầu đến cuối vẫn chưa được giải quyết triệt để, và các hệ thống này vẫn đang trong quá trình thử nghiệm và cải tiến liên tục [1]

CHƯƠNG 3 : PHÁT HIỆN VẬT THỂ DỰA TRÊN CNN

Phát hiện vật thể đóng vai trò then chốt trong việc giúp xe tự lái và hệ thống hỗ trợ lái xe tiên tiến (ADAS) nhận biết môi trường xung quanh và đưa ra các quyết định chính xác. Nhiều thuật toán phát hiện, bao gồm các phương pháp deep CNN , đã được đề xuất để phục vụ mục đích này. Tuy nhiên, các mô hình này thường có yêu cầu tính toán cao, khiến chúng khó triển khai trên các nền tảng nhúng với tài nguyên hạn chế. Do đó, việc phát triển một mô hình mạng nơ-ron nhỏ gọn hơn để giải quyết bài toán tính toán này là hết sức cần thiết. Thách thức lớn nhất là đảm bảo việc chuyển giao mô hình phát hiện vật thể sang nền tảng nhúng mà vẫn duy trì được độ chính xác và ổn định, điều vốn rất quan trọng để đảm bảo an toàn và độ tin cậy của xe tự lái cũng như các hệ thống ADAS. [1]

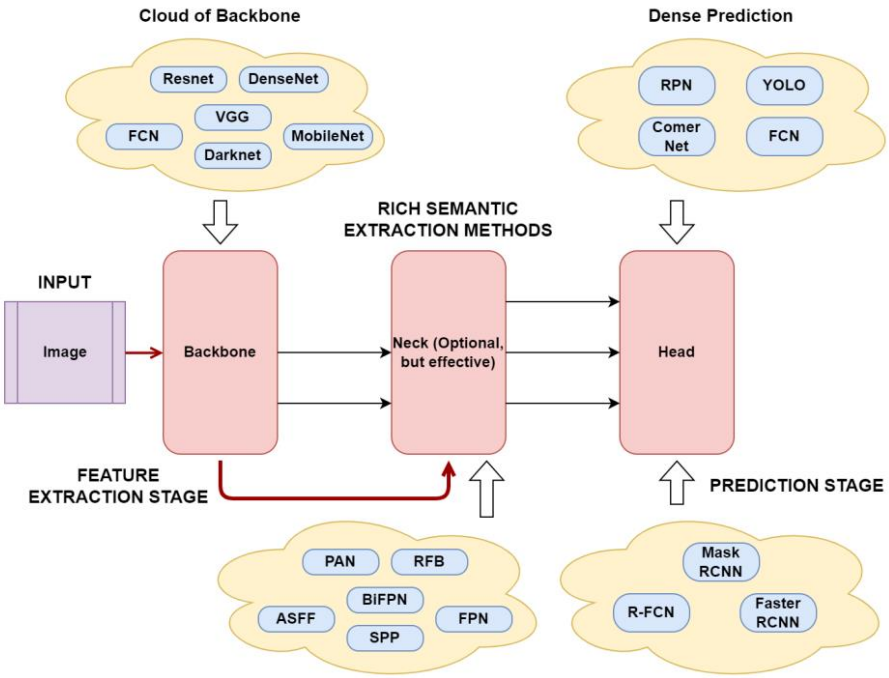
Mạng nơ-ron tích chập (CNN) thuộc họ mạng nơ-ron sâu (DNN – Deep Neural Networks), được đặc trưng bởi các nơ-ron nhân tạo được tổ chức thành các lớp. CNN có một số đặc điểm riêng biệt so với các loại mạng nơ-ron khác, bao gồm vùng tiếp nhận cục bộ (local receptive field) , lấy mẫu không gian (spatial subsampling) và trọng số được chia sẻ. CNN cho phép sự kết hợp giữa nhiều giai đoạn xử lý tuần tự, bao gồm các bước chập (convolution), đệm (padding), gộp (pooling), và đưa vào các hàm phi tuyến thông qua các hàm kích hoạt như Sigmoid, Softmax, TanH, ReLU, Leaky ReLU cùng với các lớp kết nối đầy đủ (FCL – Fully Connected Layers). Mỗi lớp chập có đầu vào và đầu ra được gọi là các bản đồ đặc trưng (feature maps), là các mảng 2 chiều đối với ảnh màu, 3 chiều đối với video và 1 chiều đối với dữ liệu âm thanh. Trong quá trình xử lý dữ liệu đầu vào, mỗi tầng CNN sẽ trích xuất các đặc trưng mới, và các tầng sau sẽ nhận diện các đặc trưng ngày càng nổi bật hơn thông qua các phép chập. T. Turay và cộng sự [2] đã cung cấp định nghĩa toàn diện về tất cả các thành phần kiến trúc của một mạng CNN, bao gồm các phương trình toán học, đồng thời sử dụng kiến trúc tham chiếu là AlexNet [3].

Các thuật toán phát hiện vật thể hiện nay được phân loại thành hai nhóm chính: thuật toán một giai đoạn (one-stage) và hai giai đoạn (two-stage). Chỉ các thuật toán một giai đoạn mới có khả năng hoạt động hiệu quả trong điều kiện hạn chế tài nguyên tính toán của hệ thống nhúng theo thời gian thực. Trong khi đó, các thuật toán hai giai đoạn, điển hình là Faster R-CNN [4], yêu cầu nhiều tài nguyên tính toán hơn, khiến chúng không phù hợp để triển khai trên các nền tảng nhúng. Hiện tại, SSD (Single Shot Detector) [5] và YOLO (You Only Look Once) [6] là hai phương pháp phát hiện vật thể một giai đoạn được sử dụng phổ biến nhất.

Biển báo giao thông thường xuất hiện dưới nhiều kích thước khác nhau trong khung hình , từ rất nhỏ khi ở xa và trở nên lớn hơn khi tiến đến càng gần biển báo . tại các phiên bản đầu tiên của YOLO (YOLOV1) và SSD , Theo W. Liu et al. [5], YOLO có thể không hiệu quả trong việc phát hiện nhiều mục tiêu dày đặc và các đối tượng lớn, trong khi máy dò SSD được coi là phù hợp hơn để phát hiện hiệu quả một số lượng lớn các đối tượng với các quy mô và loại khác nhau. Tuy nhiên , Ngay từ phiên bản đầu tiên, YOLO đã nổi tiếng với tốc độ xử lý cực nhanh, vượt xa các đối thủ cùng thời, bao gồm cả SSD. Điều này làm cho YOLO trở thành lựa chọn hàng đầu cho các ứng dụng yêu cầu thời gian thực như xe tự lái, giám sát video, robot, v.v. , nhờ có tốc độ vượt trội YOLO được chú ý trong cộng đồng và được các bên khác nhau phát triển và có nhiều cải tiến đáng kể khi trải qua nhiều phiên bản khác nhau. Mỗi phiên bản đều mang lại những cải tiến về độ chính xác, tốc độ và hiệu quả tính toán. để biết được những cải tiến trong các phiên bản YOLO , những ai quan tâm được khuyến khích tham khảo bài báo [7]

CHƯƠNG 4 : KIẾN TRÚC MÔ HÌNH PHÁT HIỆN ĐỐI TƯỢNG VÀ MÔ HÌNH YOLO-NAS

4.1 KIẾN TRÚC MÔ HÌNH PHÁT HIỆN ĐỐI TƯỢNG



Hình 4.1 : kiến trúc mô hình phát hiện đối tượng

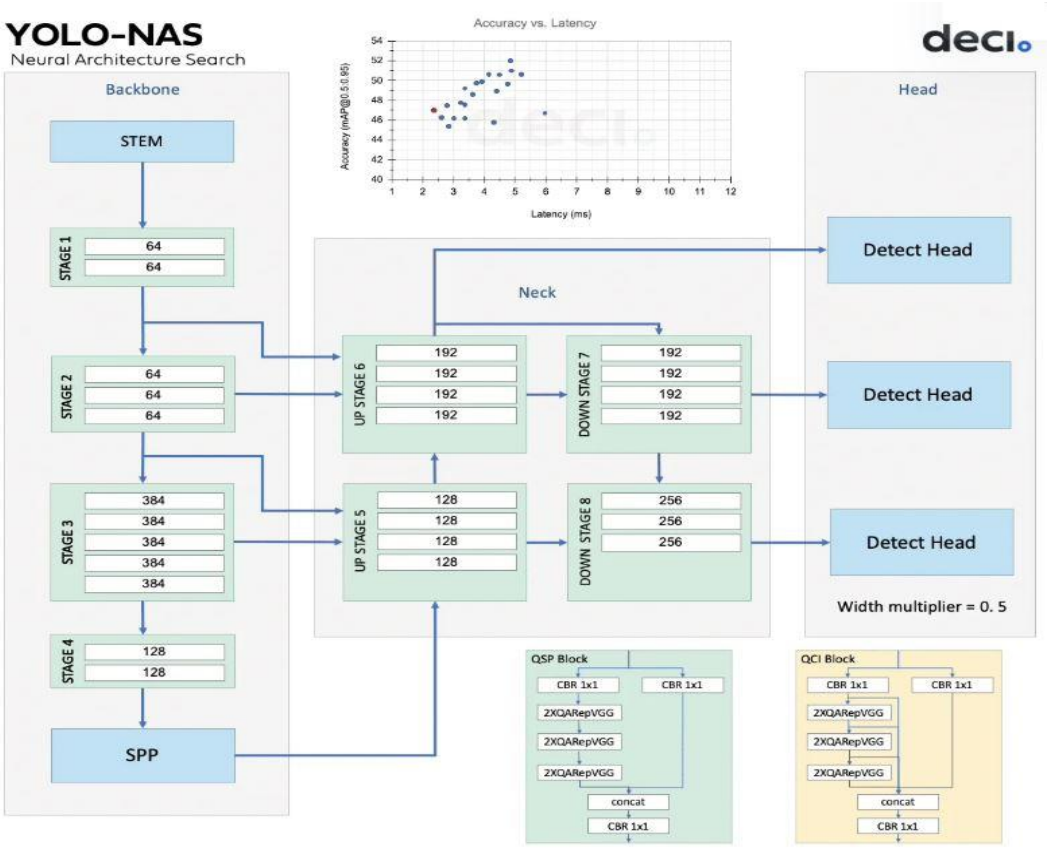
Backbone chịu trách nhiệm trích xuất các đặc trưng hữu ích từ ảnh đầu vào. Thông thường, backbone là một mạng nơ-ron tích chập (CNN) được huấn luyện trên các tác vụ phân loại hình ảnh quy mô lớn, chẳng hạn như ImageNet. Backbone sẽ nắm bắt các đặc trưng theo cấu trúc phân cấp ở nhiều mức độ khác nhau, với các đặc trưng ở cấp thấp hơn (như cạnh và kết cấu) được trích xuất từ những lớp đầu tiên, và các đặc trưng cấp cao hơn (như các bộ phận của vật thể và thông tin ngữ nghĩa) được trích xuất từ những lớp sâu hơn.

Neck là một thành phần trung gian, kết nối backbone với head. Neck tổng hợp và tinh chỉnh các đặc trưng được trích xuất bởi backbone, thường tập trung vào việc cải thiện thông tin không gian và thông tin ngữ nghĩa trên nhiều tỷ lệ

khác nhau. Neck có thể bao gồm các lớp tích chập bổ sung, Feature Pyramid Networks (FPN) [8], hoặc các cơ chế khác nhằm cải thiện chất lượng biểu diễn của các đặc trưng.

Head là thành phần cuối cùng của một mô hình phát hiện vật thể; chịu trách nhiệm đưa ra các dự đoán dựa trên các đặc trưng được cung cấp bởi backbone và neck. Thông thường, head bao gồm một hoặc nhiều mạng con chuyên biệt thực hiện các nhiệm vụ như phân loại (classification), định vị (localization), và gần đây hơn là phân vùng thực thể (instance segmentation) hay ước lượng tư thế (pose estimation). Head xử lý các đặc trưng do neck cung cấp và tạo ra các dự đoán cho từng ứng viên vật thể. Cuối cùng, một bước hậu xử lý, chẳng hạn như Non-Maximum Suppression (NMS), sẽ loại bỏ các dự đoán chồng lấn và chỉ giữ lại những phát hiện có độ tin cậy cao nhất.4.2 kiến trúc mô hình YOLO NAS

4.2 KIẾN TRÚC MÔ HÌNH YOLO NAS



Hình 4.2 : kiến trúc mô hình YOLO NAS

4.2.1 Backbone

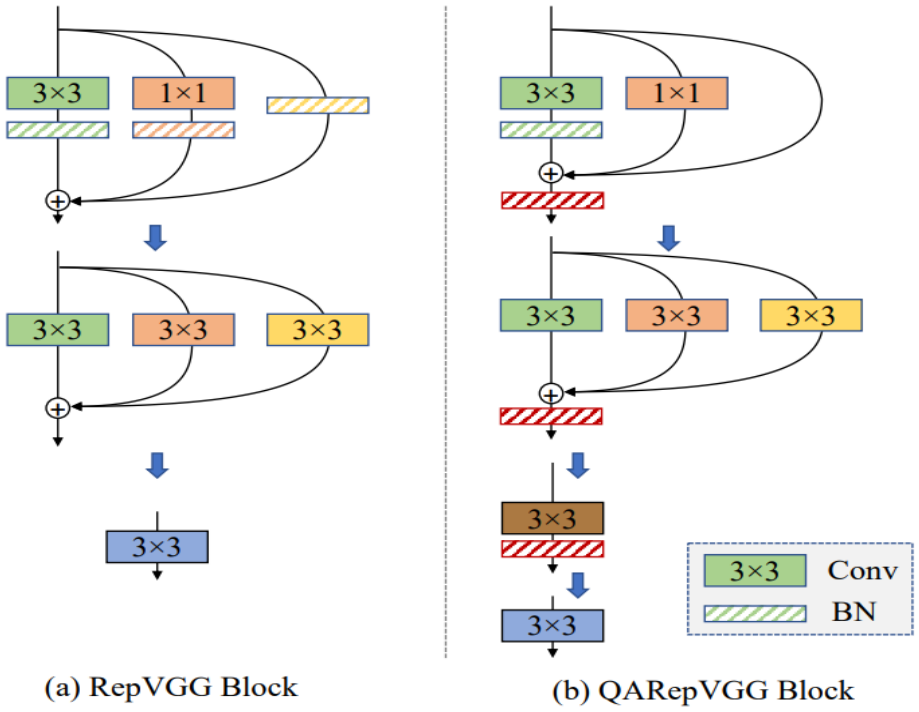
```
-----------------------------------------------
YoloNAS_S (YoloNAS_S)
├─NStageBackbone (backbone)
│    └─YoloNASStem (stem)
│    │    └─QARepVGGBlock (conv)
│    └─YoloNASStage (stage1)
│    │    └─QARepVGGBlock (downsample)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASStage (stage2)
│    │    └─QARepVGGBlock (downsample)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASStage (stage3)
│    │    └─QARepVGGBlock (downsample)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASStage (stage4)
│    │    └─QARepVGGBlock (downsample)
│    │    └─YoloNASCSPLayer (blocks)
│    └─SPP (context_module)
│    │    └─Conv (cv1)
│    │    └─ModuleList (m)
│    │    └─Conv (cv2)
```

Hình 4.3 : phần backbone của YOLO NAS SMALL

Backbone bao gồm khối stem ,bốn khối stage1 đến stage4 và module SPP(spatial pyramid pooling) [9]

Khối stem chỉ bao gồm 1 QARepVGGBlock [10]

Khối stage1 đến stage4 đều bao gồm khối downsample và các block

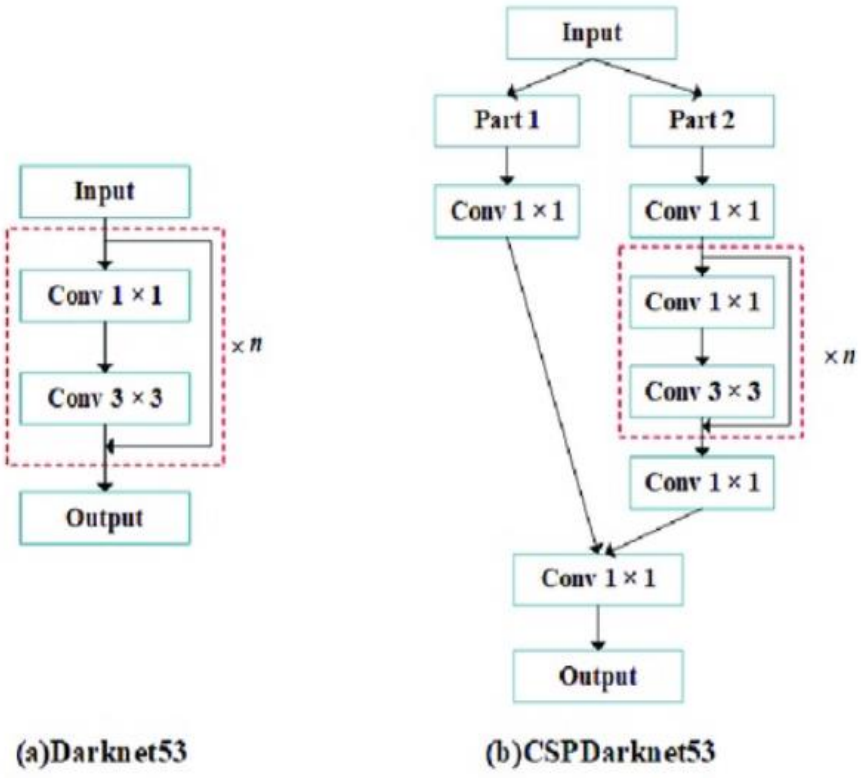


Hình 4.4 : kiến trúc khối RepVGG Block và QARepVGG Block

YoloNASCSPLayser với số lượng khác nhau cho mỗi stage .

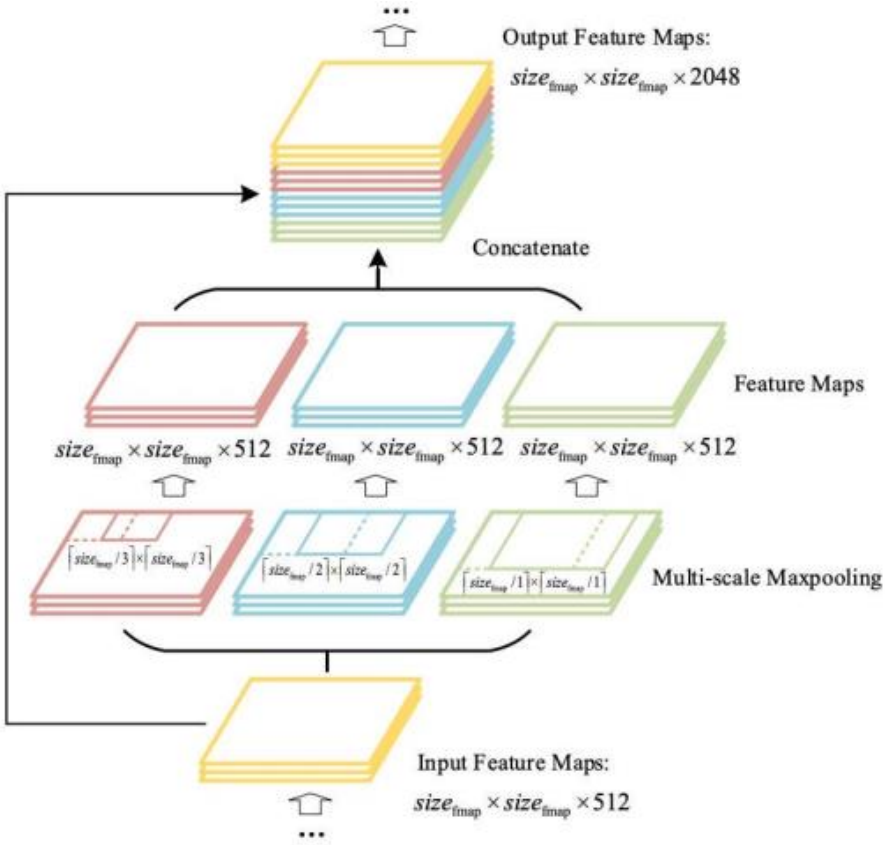
YoloNASCSPLayer tuân theo kiến trúc của QSP Block.

QSP Block được tinh chỉnh từ CSPNet (Cross Stage Partial Network) [11], thay thế khối Conv 1x1- Conv3x3 bằng 2xRepVGGBlock đồng thời các khối Conv1x1 được thay thế bằng khối CBR1x1(Convolution-Batchnorm-Relu)



Hình 4.5 : kiến trúc khối vi mô của Darknet53 và CSPDarknet53

Module SSP bao gồm 2 khối CBR1x1 và một Module list bao gồm 3 maximum pooling (MaxPool2d có kernel size lần lượt là 5, 9,13)



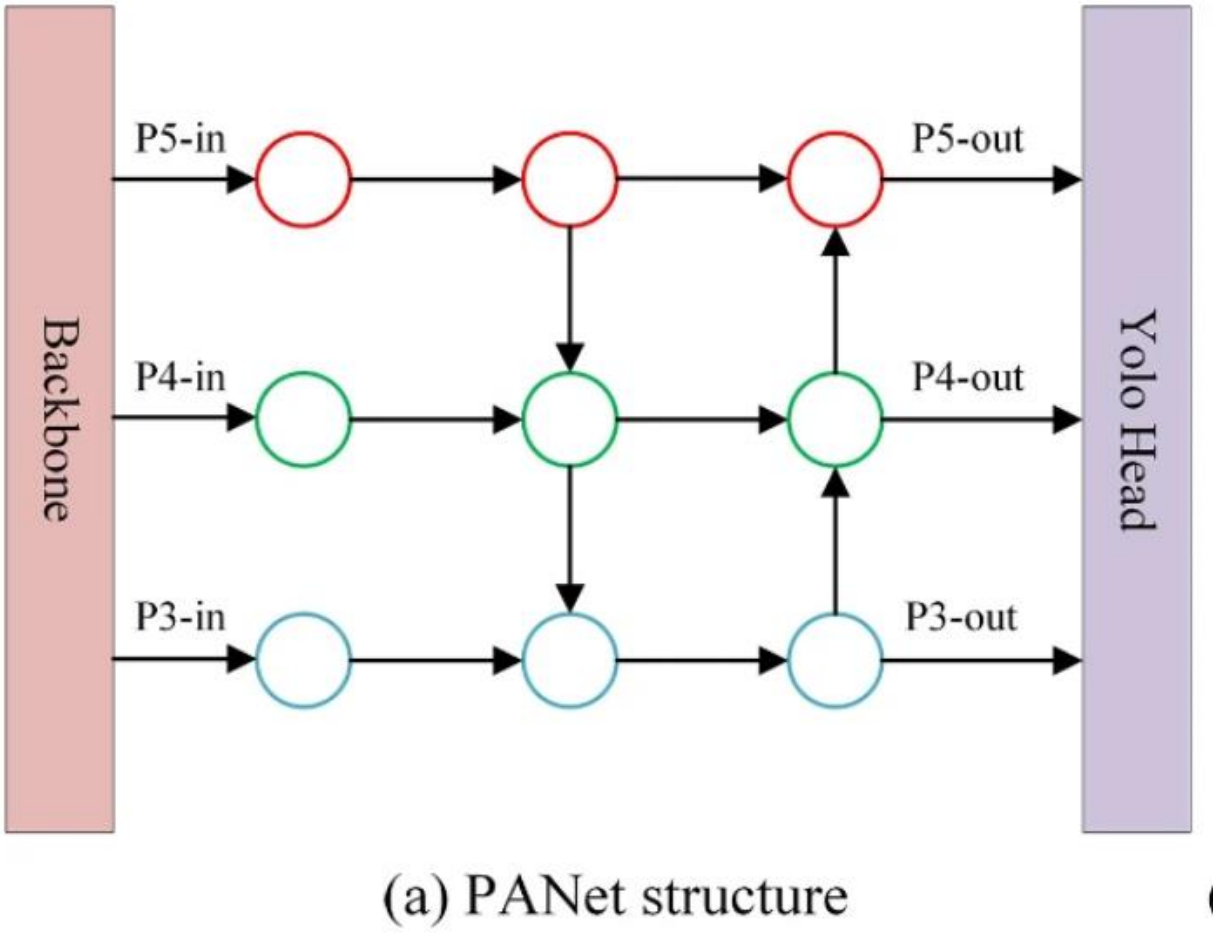
Hình 4.6 : kiến trúc SPP được sử dụng trong YOLO

4.2.2 Neck

```
├─YoloNASPANNeckWithC2 (neck)
│    └─YoloNASUpStage (neck1)
│    │    └─Conv (reduce_skip1)
│    │    └─Conv (reduce_skip2)
│    │    └─Conv (downsample)
│    │    └─Conv (conv)
│    │    └─ConvTranspose2d (upsample)
│    │    └─Conv (reduce_after_concat)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASUpStage (neck2)
│    │    └─Conv (reduce_skip1)
│    │    └─Conv (reduce_skip2)
│    │    └─Conv (downsample)
│    │    └─Conv (conv)
│    │    └─ConvTranspose2d (upsample)
│    │    └─Conv (reduce_after_concat)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASDownStage (neck3)
│    │    └─Conv (conv)
│    │    └─YoloNASCSPLayer (blocks)
│    └─YoloNASDownStage (neck4)
│    │    └─Conv (conv)
│    │    └─YoloNASCSPLayer (blocks)
```

Hình 4.7 : phần neck của YOLO NAS SMALL

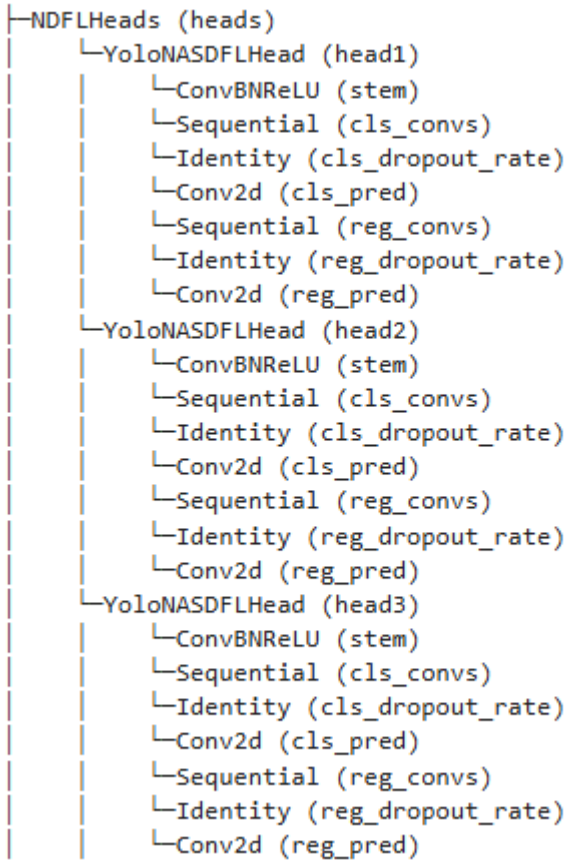
Kiến trúc PAN + cross scale connection [12] giúp tổng hợp đặc trưng ở ba độ phân giải khác nhau để phát hiện vật thể đa tỉ lệ.



Hình 4.8 : kiến trúc PAN

Cross scale connection giúp đưa các đặc trưng từ các layer nông đến các layer sâu hơn để giữ được các chi tiết như các cấu trúc biên , góc , đỉnh, …, của đối tượng .

4.2.3 Head



Hình 4.9 : phần head của YOLO NAS SMALL

Ba head song song xử lý ba bản đồ đặc trưng P3–P5 sinh ra từ Neck, bảo đảm nhận dạng vật thể nhỏ → trung → lớn.

Khối stem (CBR) chịu trách nhiệm nhận bản đồ đặc trưng từ neck và chuẩn hóa độ rộng kênh.

Khối cls_convs và cls_pred kết hợp với nhau tạo nên một CNN phân lớp (FC layer được thực thi dưới dạng Conv 1x1) tạo ra kết quả cho nhánh classsification.

Khối reg_convs và reg_pred kết hợp với nhau tạo nên một CNN hồi quy (FC layer được thực thi dưới dạng Conv 1x1) tạo ra kết quả cho nhánh regression .

DFL – Distribution Focal Loss [13] cho phép hồi quy 4 tọa độ bouding box dưới dạng 4 phân phối xác xuất và để suy ra tọa độ ta áp dụng công thức kỳ vọng cho mỗi phân phối.

CHƯƠNG 5 : HÀM MẤT MÁT (LOSS FUNCTION)

5.1 PPYoloELoss

PPYoloELoss là một hàm loss toàn diện được sử dụng trong các mô hình YOLO (You Only Look Once), Hàm loss này kết hợp nhiều thành phần đo lường các khía cạnh khác nhau về hiệu suất của mô hình.

5.1.1 PPYoloELoss/loss_cls

Thành phần này đo lường mức độ mô hình phân loại chính xác mỗi đối tượng vào đúng phân lớp của nó. Nó so sánh xác suất lớp dự đoán với nhãn lớp thực sự. Phương trình tính toán PPYoloELoss/loss_cls được đưa ra bên dưới:

$$Loss_{cls} = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} -y_{i,c}\log \hat{y}_{i,c} \qquad (5.1)$$

trong đó N là số lượng mẫu , C là số lượng class , $y_{i,c}$ là nhãn (xác suất) thực tế (ground truth) của class c thuộc mẫu thứ i , $\hat{y}_{i,c}$ là nhãn (xác suất) dự đoán của class c thuộc mẫu thứ i

5.1.2 PPYoloELoss/loss_iou

Hàm mất mát Intersection over Union (IoU), đo lường mức độ chồng lấn (overlap) giữa các hộp giới hạn dự đoán và các hộp giới hạn thực tế (ground truth). Mục tiêu của hàm này là tối đa hóa sự chồng lấn (overlap) đó, qua đó cải thiện độ chính xác của việc dự đoán các hộp giới hạn. Phương trình tính toán được đưa ra bên dưới :

$$\text{Loss}_{iou} = 1 - \frac{|B \cap B_{gt}|}{|B \cup B_{gt}|} \qquad (5.2)$$

trong đó B là hộp giới hạn dự đoán và B_{gt} là hộp giới hạn thực tế.

5.1.3 PPYoloELoss/loss_dfl

Hàm mất mát Distribution Focal Loss (DFL), giúp cải thiện độ chính xác của hồi quy đối với các tọa độ hộp giới hạn bằng cách tập trung vào những khoảng phân bố quan trọng nhất cho mỗi tọa độ. Phương trình tính toán được đưa ra bên dưới:

$$Loss_{dfl} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{4}\sum_{k=0}^{k-1} -y_{i,j,k}\log\left(\hat{y}_{i,j,k}\right) \qquad (5.3)$$

trong đó N là số lượng mẫu, j là chỉ số bốn tọa độ của hộp giới hạn (x_{min} , y_{min} , x_{center} , y_{center}), k là số lượng các khoảng (bins) rời rạc trong phân bố , $y_{i,j,k}$ là xác suất thực tế (ground truth) của bin thứ k cho tọa độ j của mẫu i, $\hat{y}_{i,j,k}$ là xác suất dự đoán tương ứng .

5.1.4 PPYoloELoss/loss

PPYoloELoss/loss là hàm mất mát tổng thể, được tính bằng tổng các thành phần mất mát riêng lẻ, thường được nhân trọng số bởi các hệ số cụ thể nhằm cân bằng đóng góp của từng thành phần vào giá trị mất mát tổng thể. Phương trình tính toán PPYoloELoss/loss được trình bày dưới đây:

$$Loss = Loss_{cls} + \lambda_{iou}Loss_{iou} + \lambda_{dfl}Loss_{dfl} \qquad (5.4)$$

CHƯƠNG 6 : CÁC CHỈ SỐ ĐÁNH GIÁ (METRICS EVALUATION)

6.1 PRECISION (ĐỘ CHÍNH XÁC)

Precision (độ chính xác) là tỷ lệ giữa số lượng các phát hiện đúng (true positives) trên tổng số các phát hiện được mô hình dự đoán là dương tính (positive detections). Ở ngưỡng Intersection over Union (IoU) bằng 0.50, một phát hiện được xem là true positive nếu giá trị IoU giữa hộp giới hạn được dự đoán và hộp giới hạn thực tế đạt ít nhất là 0.50. Chỉ số này đo lường mức độ chính xác của các dự đoán dương tính được tạo ra bởi mô hình. Độ chuẩn xác càng cao cho thấy càng ít trường hợp dự đoán sai dương tính (false positives). Phương trình tính toán precision được trình bày như sau:

$$\text{Precision@0.50} = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)} \qquad (6.1)$$

6.2 RECALL (ĐỘ BAO PHỦ)

Recall (độ bao phủ) là tỷ lệ giữa số lượng các phát hiện đúng (true positives) trên tổng số các đối tượng thực tế có trong tập dữ liệu. Ở ngưỡng Intersection over Union (IoU) bằng 0.50, một phát hiện được coi là true positive nếu giá trị IoU giữa hộp giới hạn dự đoán và hộp giới hạn thực tế đạt ít nhất là 0.50. Chỉ số này đo lường khả năng của mô hình trong việc phát hiện tất cả các đối tượng liên quan. Recall càng cao cho thấy số lượng bỏ sót (false negatives) càng ít. Phương trình tính toán Recall được trình bày như sau :

$$\text{Recall@0.50} = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative (FN)} \qquad (6.2)$$

6.3 MEAN AVERAGE PRECISION (MAP)

Mean Average Precision (mAP) tại ngưỡng IoU 0.50 là giá trị trung bình của các điểm Average Precision (AP) cho tất cả các lớp đối tượng tại ngưỡng Intersection over Union (IoU) lớn hơn hoặc bằng 0.50. Trong đó, AP là diện tích bên dưới đường

cong precision-recall của từng lớp. Công thức tính toán mAP được trình bày như sau:

$$\text{mAP@0.50} = \frac{1}{C}\sum_{c=1}^{C} AP_c \qquad (6.3)$$

trong đó C là số lượng các phân lớp , $AP_c = \int_0^1 p_c(r)dr$ là Average Precision của phân lớp thứ c tại ngưỡng IoU lớn hơn hoặc bằng 0.50 , $p_c(r)$ là độ chính xác (precision) tại giá trị recall R được tính cho phân lớp c và dr là kí hiệu vi phân biểu diễn một sự thay đổi vô cùng nhỏ của độ bao phủ (recall).

Chỉ số này cung cấp một phép đo toàn diện về độ chính xác (precision) và độ bao phủ (recall) của mô hình trên tất cả các lớp đối tượng. Giá trị mAP càng cao thể hiện hiệu suất tổng thể của mô hình càng tốt.

CHƯƠNG 7 : PHƯƠNG PHÁP TỐI ƯU

7.1 ADAM

Giải thuật Adam [14]

Algorithm 1: Adam, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are α = 0.001, $\beta_1 = 0.99$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and to β_2 the power *t*.

Require: α: Stepsize

Require: $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates

Require: f(θ): Stochastic objective function with parameters θ

Require: θ_0: Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)

$v_0 \leftarrow 0$ (Initialize 2nd moment vector)

$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t stochastic objective at timestep t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)

$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$ (Compute bias-corrected first moment estimate)

$\widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$ (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}+\epsilon}$ (Update parameters)

end while

return θ_t (Resulting parameters)

7.2 ADAMW

AdamW là một biến thể của Adam. Ý tưởng của AdamW khá đơn giản: khi thực hiện thuật toán Adam với L2 regularization (chuẩn hóa L2), tác giả loại bỏ phần suy giảm của trọng số (weight decay) $\lambda\theta_t$ khỏi công thức tính gradient hàm mất mát tại thời điểm t :

$$g_t = \nabla f(\theta_{t-1}) + \lambda\theta_{t-1} \tag{7.1}$$

và thay vào đó, đưa phần giá trị đã được phân tách này vào quá trình cập nhật trọng số:

$$\theta_t = \theta_{t-1} - \alpha\frac{1}{\sqrt{\widehat{v_t}}+\epsilon}\widehat{m_t} + \lambda\,\theta_{t-1} \tag{7.2}$$

CHƯƠNG 8 : DỮ LIỆU

8.1 THU THẬP DỮ LIỆU

Tổng cộng có 4489 tấm ảnh được thu thập từ roboflow và các nguồn khác nhau trên internet. Các tấm ảnh được thu thập ở các điều kiện giao thông khác nhau như đường núi dốc , đường nội thành , đường ngoại ô , đường cao tốc, v.v. Ngoài ra điều kiện thời tiết và ngoại cảnh xung quanh cũng rất đa dạng , bao gồm : tầm nhìn tốt khi đầy đủ ánh sáng và vị trí biển báo rõ ràng , tầm nhìn tương đối hạn chế khi trời mưa , sương mù , ánh sáng yếu , biển báo bị che khuất một phần, v.v.

Có 41 loại biển báo thuộc về ba phân lớp chính là biển giới hạn tốc độ , biển cấm và biển cảnh báo .

Việc thu thập dữ liệu ở những điều kiện khác nhau như vậy giúp mô hình có thể học khái quát các trường hợp khác nhau trong bối cảnh giao thông và thời tiết rất phức tạp diễn ra trong thực tế .

8.2 TIỀN XỬ LÝ DỮ LIỆU

Các tấm ảnh được thu thập ban đầu chưa được chuẩn hóa về chất lượng nên có thể bị trùng , không thể nhận diện được biển báo trong khung hình , v.v. Sẽ được loại bỏ . Sau quá trình này sẽ thu được nguồn dữ liệu chất lượng cho giai đoạn đào tạo mô hình .

Từ 4489 bức ảnh sau khi tiền xử lý thu được 3917 bức ảnh.

8.3 GÁN NHÃN DỮ LIỆU

Để gán nhãn nhãn cho ảnh , công cụ chú thích hình ảnh LabelImg được sử dụng . ảnh sau khi được chú thích được lưu dưới dạng VOC sau đó được chuyển sang định dạng COCO để phù hợp với yêu cầu của mô hình .

Hình bên dưới trực quan hóa một số ảnh sau khi được tạo chú thích hộp tọa độ.

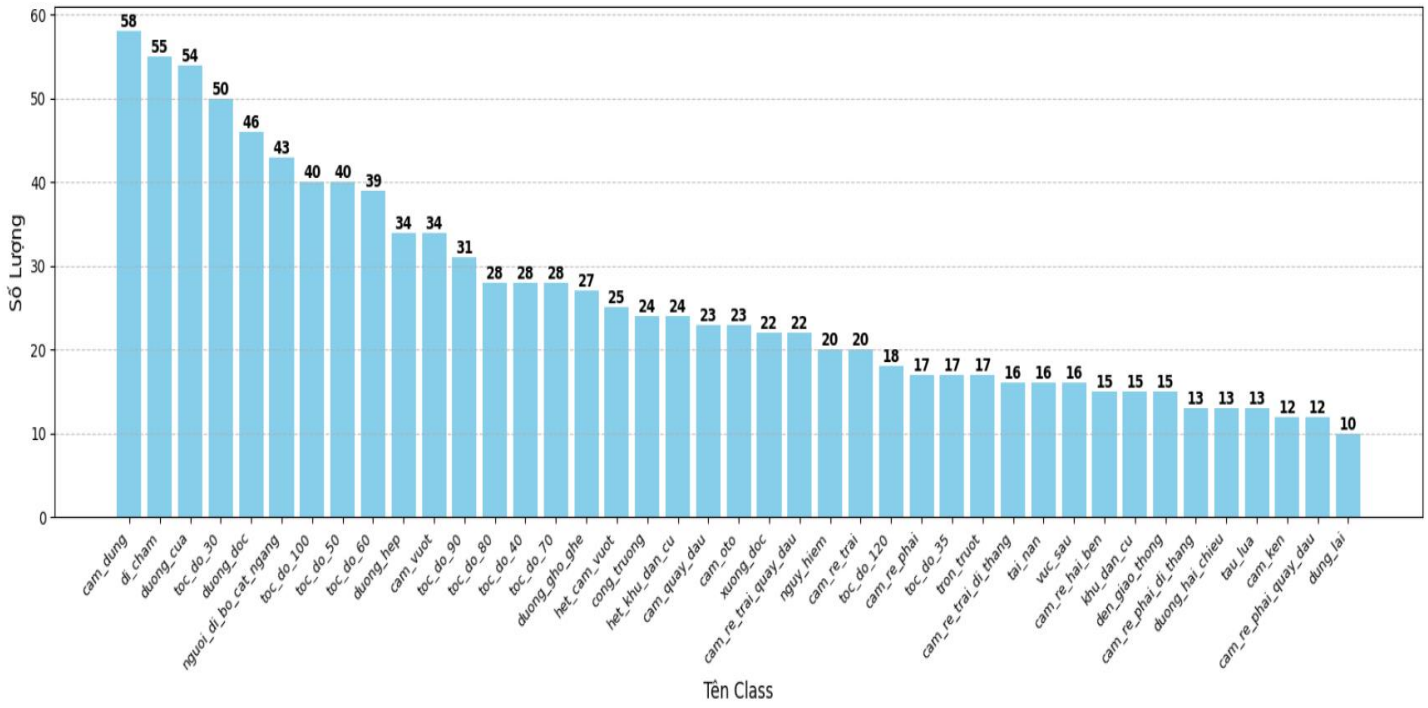
Hình 8.1 : ảnh sau khi được tạo chú thích hộp tọa độ

8.3 PHÂN TÍCH KHAI PHÁ DỮ LIỆU (EXPLORATORY DATA ANALYSIS – EDA)

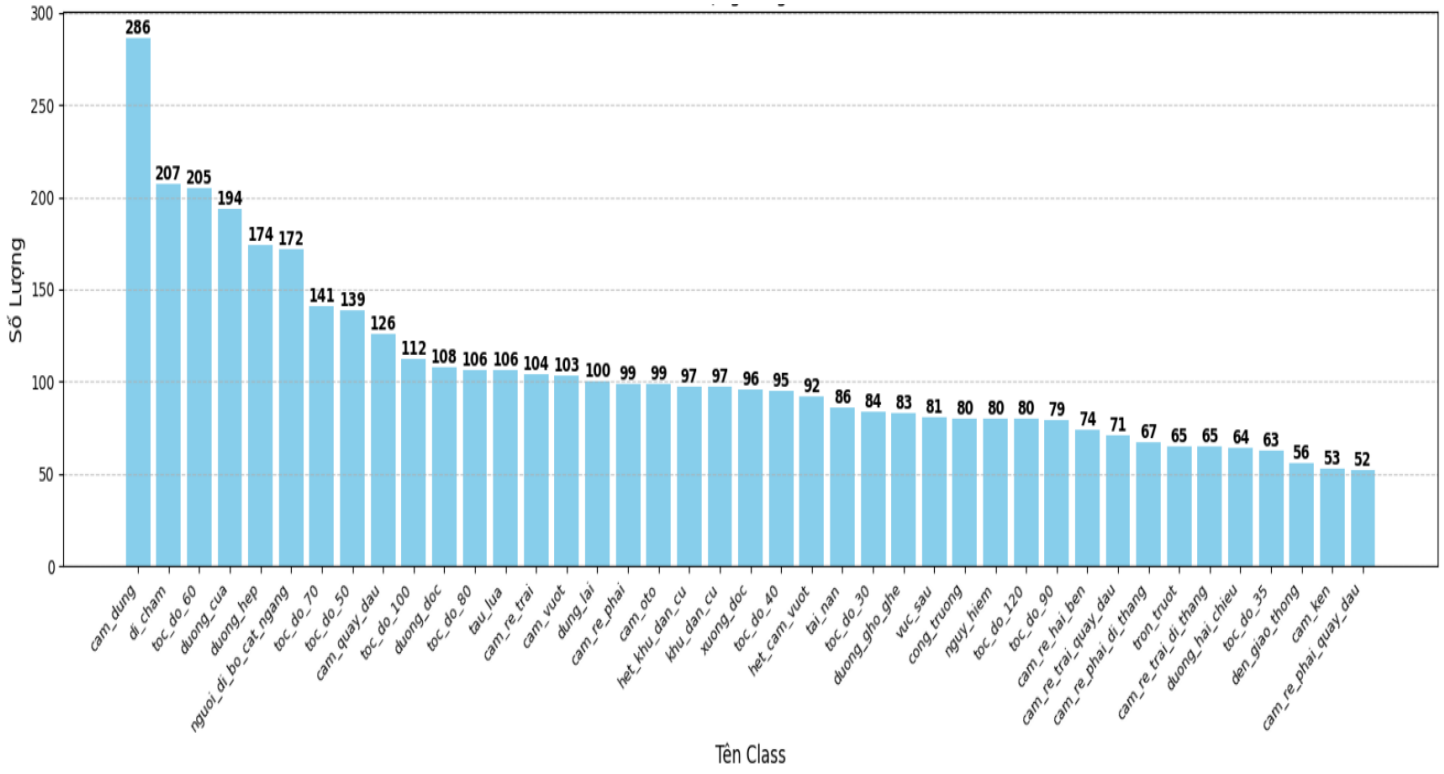
EDA là một kỹ thuật phân tích dữ liệu sử dụng các chỉ số thống kê và biểu đồ nhằm mục tiêu thấy được những " insight" của dữ liệu trước khi tiến hành đào tạo mô hình.

Tiến hành thống kê số lượng hộp tọa độ của tất cả các phân lớp và trực quan hóa dưới dạng biểu đồ dạng cột (bar chart)

Nhìn vào hai biểu đồ có thể thấy dữ liệu bị mất cân bằng giữa các phân lớp . điều này có thể dẫn đến khả năng phát hiện của mô hình với các phân lớp ít xuất hiện trong tập dữ liệu sẽ bị giảm đi.



Hình 8.2 : biểu đồ số lượng hộp tọa độ cho mỗi phân lớp trong tập kiểm thử



Hình 8.3 : biểu đồ số lượng hộp tọa độ cho mỗi phân lớp trong tập đào tạo

CHƯƠNG 9 : HUẤN LUYỆN MÔ HÌNH

9.1 KAGGLE

Kaggle là một nền tảng trực tuyến nổi tiếng dành cho khoa học dữ liệu và machine learning. hỗ trợ môi trường lập trình notebook (dựa trên Jupyter notebook) cho phép người dùng trực tiếp lập trình, chạy code Python/R trong trình duyệt, với tài nguyên máy chủ miễn phí (GPU, TPU) giúp tăng tốc các tác vụ huấn luyện mô hình AI.

9.2 SUPERGRADIENT

SuperGradients là thư viện mã nguồn mở dựa trên PyTorch, do Deci AI phát triển, với đặc điểm là có thể truy cập , huấn luyện và tinh chỉnh các mô hình computer vision hiện đại (SOTA – State Of The Art) như YOLO-NAS một cách nhanh chóng và hiệu quả.

Thư viện cung cấp các "recipe" – tập hợp hyper-parameters đã được kiểm thử, giúp đạt được kết quả chất lượng cao mà không tốn quá nhiều thời gian thử nghiệm thủ công.

Thư viện hỗ trợ nhiều tác vụ như phân loại, phát hiện đối tượng, phân đoạn ngữ nghĩa và ước lượng tư thế, đồng thời tương thích với các công cụ triển khai như ONNX, TensorRT và OpenVINO, đảm bảo sẵn sàng đưa vào môi trường sản xuất.

9.3 THUẬT TOÁN HUẤN LUYỆN MÔ HÌNH

Đầu vào : I ← *ảnh* đầu *vào trong tập dữ liệu* đào *tạo*

M ← *mô hình YOLO NAS SMALL*

α ← *tốc* độ *học* (*learning rate or step size*)

λ ← *w*eight decay L2

λ_{iou} ← *trọng số cho hàm* $Loss_{iou}$

λ_{dfl} ← *trọng số cho hàm* $Loss_{dfl}$

T ← tập dữ liệu đào tạo

Epoch ← số lần lặp qua toàn bộ dữ liệu

Chuẩn hóa ảnh đầu vào I

$I_{norm} = \frac{I-\mu}{\sigma}$

trong đó μ là $trung\ bình$ và σ là độ $lệch\ chuẩn\ của\ giá\ trị\ điểm\ ảnh$

Tăng cường bộ dữ liệu T với các kỹ thuật như xoay , lật , cắt bỏ , cắt ghép , tinh chỉnh độ bão hòa ,.. của ảnh để tạo ra bộ dữ liệu T_{aug}

Khởi tạo mô hình YOLO NAS SMALL M với bộ trọng số θ được đào tạo trên bộ dữ liệu COCO 2017

Định nghĩa hàm mất mát PPYoloELoss L

Phương thức tối ưu hóa AdamW

For each epoch:

For each batch B in T_{aug}:

Forward pass : tính toán đầu ra P = M(B)

Tính toán hàm mất mát L sử dụng đầu ra dự đoán

P và giá trị thực tế G

L=$\text{Loss}_{cls}(P,G) + \lambda_{iou}\text{Loss}_{iou}(P,G) + \lambda_{dfl}\text{Loss}_{dfl}(P,G)$

Backward pass: tính toán gradient của L đối với các trọng số θ của mô hình

$\nabla L = \nabla \text{Loss}_{cls} + \lambda_{iou}\nabla \text{Loss}_{iou} + \lambda_{dfl}\nabla \text{Loss}_{dfl}$

Cập nhật trọng số θ của mô hình:

$\theta = \theta - \alpha\nabla L + \lambda\theta$

Đầu ra : mô hình YOLO NAS SMALL M với bộ trọng số đã được hiệu chỉnh tối ưu

9.4 THÔNG SỐ THAM SỐ HUẤN LUYỆN MÔ HÌNH

Mô hình YOLO-NAS SMALL có tổng cộng 19,02 triệu tham số. Tất cả các tham số này đều đã được tối ưu hóa. Việc thiết lập siêu tham số một cách tối ưu đóng vai trò quan trọng trong việc đạt được hiệu suất hiệu quả và chính xác. Bằng cách sử dụng kích thước lô (batch size) là 16, mỗi lần lặp có thể xử lý một lượng hình ảnh phù hợp, từ đó đạt được sự cân bằng hài hòa giữa khối lượng tính toán và độ chính

xác của mô hình. Giá trị tích lũy lô (batch accumulation) là 1 cho thấy các gradient được cập nhật sau mỗi lô, nhờ đó duy trì quá trình học liên tục..

Mô hình được huấn luyện theo chu kỳ thường xuyên với 195 lần lặp (iteration) trên mỗi epoch và tương ứng với số lần cập nhật gradient trong mỗi epoch, tạo ra quá trình học tập ổn định và tiến bộ dần dần. Tốc độ học (learning rate) là 0.0002 được lựa chọn nhằm tối ưu sự cân bằng giữa tốc độ hội tụ của mô hình và sự ổn định trong quá trình huấn luyện, qua đó giúp mô hình không vượt quá nghiệm tối ưu. Các tham số trọng số suy giảm (weight decay) được gán giá trị là 0.001 nhằm thực hiện quy chuẩn hóa (regularization) cho mô hình. Chiến lược quy chuẩn hóa này áp dụng hình phạt lên các trọng số lớn, từ đó giảm nguy cơ quá khớp (overfitting) và tăng khả năng tổng quát hóa của mô hình với dữ liệu mới chưa từng thấy.

Bảng 9.1 trình bày các thiết lập tham số huấn luyện dựa trên giá trị đầu vào được cung cấp và phần mô tả tương ứng.

Parameter	Value/Description
silent_mode	True - Reduces logging during training
average_best_models	True - Enables averaging of the best models found during training
warmup_mode	"linear_epoch_step" - Warmup strategy for learning rate
warmup_initial_lr	1e-6 - Initial learning rate during warmup phase
lr_warmup_epochs	3 - Number of epochs for warmup phase
initial_lr	2e-4 - Initial learning rate for main training phase
lr_mode	"cosine" - Learning rate follows a cosine annealing schedule
cosine_final_lr_ratio	0.5 - Final learning rate is 10% of the initial learning rate
optimizer	"Adamw" - Optimization algorithm
optimizer_params	{"weight_decay": 0.001} - Weight decay (L2 regularization) parameter
zero_weight_decay_on_bias_and_bn	True - Weight decay not applied to bias and batch normalization parameters
ema	True - Enables Exponential Moving Average (EMA) of model parameters
ema_params	{"decay": 0.99, "decay_type": "threshold"} - Parameters for EMA
max_epochs	300 - Maximum number of training epochs
mixed_precision	True - Enables mixed precision training
loss	PPYoloELoss(...) - Loss function configuration

use_static_assigner	False - Specifies not to use a static assigner
num_classes	len(dataset_params['classes']) - Number of classes based on the dataset
reg_max	16 - Maximum regression value
valid_metrics_list	[DetectionMetrics_050(...)] - List of validation metrics
score_thres	0.1 - Score threshold for considering a prediction
top_k_predictions	300 - Top-k predictions to consider
num_cls	len(dataset_params['classes']) - Number of classes based on the dataset
normalize_targets	True - Specifies whether to normalize targets
post_prediction_callback	PPYoloEPostPredictionCallback(...) - Post-prediction processing settings
score_threshold	0.01 - Score threshold for post-prediction
nms_top_k	1000 - Top-k predictions for non-maximum suppression (NMS)
max_predictions	300 - Maximum number of predictions to consider
nms_threshold	0.7 - NMS threshold
metric_to_watch	'mAP@0.50' - Main metric to monitor during training (mean Average Precision at IoU threshold of 0.50)

Bảng 9.1 : tham số huấn luyện

Ngưỡng điểm số (score threshold) được đặt ở mức 0.01 để loại bỏ các dự đoán có độ tin cậy thấp hơn 1%, đảm bảo rằng chỉ những phát hiện có khả năng cao nhất mới được xem xét xử lý tiếp theo. Thứ hai, 1000 dự đoán hàng đầu dựa trên điểm số độ tin cậy sẽ được giữ lại trước khi áp dụng thuật toán NMS (Non-Maximum Suppression). Việc này giúp giảm tải tính toán và tập trung vào các phát hiện tiềm năng nhất.

Tham số *Max predictions* được đặt là 300, giới hạn số lượng dự đoán cuối cùng trên mỗi hình ảnh hoặc khung hình video là 300, nhằm cân bằng giữa độ bao phủ phát hiện và khả năng quản lý kết quả. Ngưỡng NMS là 0.7 được áp dụng trong quá trình NMS để loại bỏ các hộp giới hạn (bounding boxes) trùng lặp đáng kể (với IoU – lớn hơn hoặc bằng 0.7), chỉ giữ lại hộp có điểm số cao nhất trong mỗi cụm.

Tổng thể, các thiết lập này giúp nâng cao độ chính xác và mức độ phù hợp của các phát hiện của mô hình bằng cách giảm thiểu các phát hiện sai (false positives) và

đảm bảo số lượng dự đoán có độ tin cậy cao ở mức có thể xử lý được. Bảng 9.2 trình bày kết quả của các chỉ số đánh giá mô hình thu được.

9.4 KIỂM THỬ VÀ ĐÁNH GIÁ MÔ HÌNH

Evaluation Metrics	Score
PPYoloELoss/loss_cls	0.53
PPYoloELoss/loss_iou	0.19
PPYoloELoss/loss_dfl	0.29
PPYoloELoss/loss	1.02
Precision@0.50	0.48
Recall@0.50	0.98
mAP@0.50	0.96

Bảng 9.2 : các chỉ số đánh giá mô hình

Hiệu suất của mô hình YOLO-NAS SMALL đã được đánh giá dựa trên một số chỉ số quan trọng. Tổng mất mát (overall loss), được biểu thị bằng chỉ số PPYoloELoss, có giá trị là 1.02, phản ánh sai số tổng hợp trên nhiều khía cạnh khác nhau trong các dự đoán của mô hình.

Cụ thể:

Mất mát phân loại (PPYoloELoss/loss_cls**)** đạt 0.53

Mất mát IoU (PPYoloELoss/loss_iou) đạt 0.19

Mất mát phân phối tiêu điểm (PPYoloELoss/loss_dfl) đạt 0.29

Các giá trị này cho thấy mô hình đang hoạt động hiệu quả trong việc phân loại đúng các đối tượng, dự đoán chính xác hộp giới hạn (bounding boxes), và tập trung vào việc định vị chính xác.

Về độ chính xác và khả năng thu hồi:

Precision@0.50 đạt 0.48, nghĩa là 48% trong số các đối tượng được phát hiện là đúng (true positives), phản ánh độ chính xác của mô hình trong việc phát hiện đúng mục tiêu.

Recall@0.50 đạt 0.98, cho thấy mô hình có khả năng phát hiện gần như tất cả các đối tượng có liên quan trong tập dữ liệu, giảm thiểu số lượng bỏ sót.

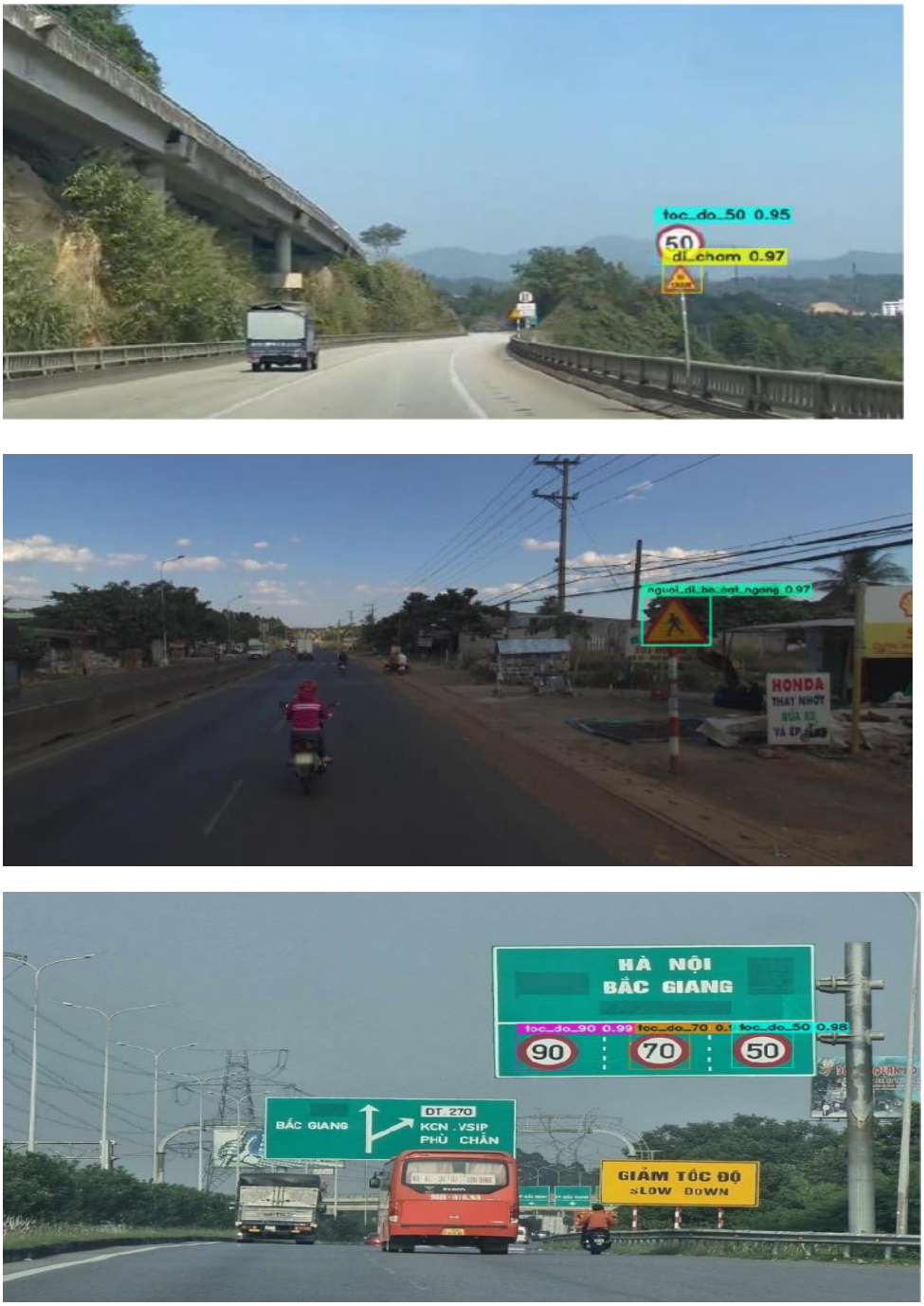
Cuối cùng, chỉ số mAP@0.50 đạt 0.96, nhấn mạnh hiệu suất mạnh mẽ của mô hình. Đây là chỉ số trung bình của độ chính xác (precision) trên tất cả các lớp và ngưỡng IoU, cho thấy mức độ chính xác cao trong các dự đoán của mô hình.

Phân lớp	**AP@0.50**
cam_re_trai	0.88
cam_re_phai	0.88
toc_do_60	0.74
toc_do_30	0.98
toc_do_40	0.97
toc_do_35	0.99
toc_do_100	0.99
toc_do_80	0.88
cam_dung	0.93
toc_do_50	0.85
cam_oto	0.98
nguoi_di_bo_cat_ngang	0.87
cam_quay_dau	0.98
duong_cua	0.91
tron_truot	0.99
den_giao_thong	0.99
duong_hep	0.94
nguy_hiem	0.99
cam_vuot	0.85
het_khu_dan_cu	0.99
het_cam_vuot	0.99
cam_re_phai_quay_dau	0.99
duong_gho_ghe	0.82
di_cham	0.99
cong_truong	0.93
duong_hai_chieu	0.99
cam_re_trai_quay_dau	0.99
toc_do_70	0.96
cam_re_phai_di_thang	0.99
cam_re_hai_ben	0.99
cam_re_trai_di_thang	0.99
xuong_doc	0.99
duong_doc	0.99
tau_lua	0.99
khu_dan_cu	0.99
vuc_sau	0.99

toc_do_120	0.96
toc_do_90	0.99
dung_lai	0.99
cam_ken	0.95
tai_nan	0.93

Bảng 9.3 : chỉ số AP@0.50 của tất cả các phân lớp

9.5 TRỰC QUAN HÓA MỘT VÀI DỰ ĐOÁN CỦA MÔ HÌNH



Hình 9.1 : dự đoán của mô hình

CHƯƠNG 10 : TỐI ƯU HÓA MÔ HÌNH

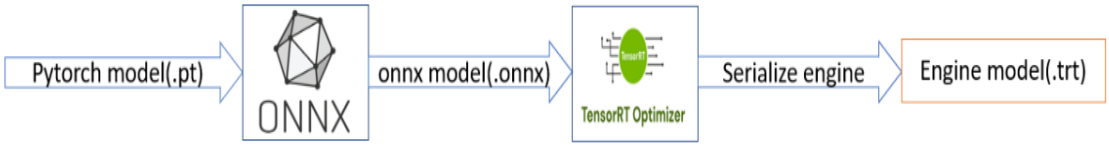
Việc thực thi mạng nơ-ron học sâu theo thời gian thực trên các thiết bị biên (edge devices) là yếu tố then chốt để hiện thực hóa nhiều lĩnh vực ứng dụng. Tuy nhiên, do bộ nhớ, tài nguyên tính toán và điện năng của các thiết bị này bị giới hạn, nên cần tối ưu hóa các mạng nơ-ron này cho phù hợp với ứng dụng nhúng.

10.1 TENSORRT

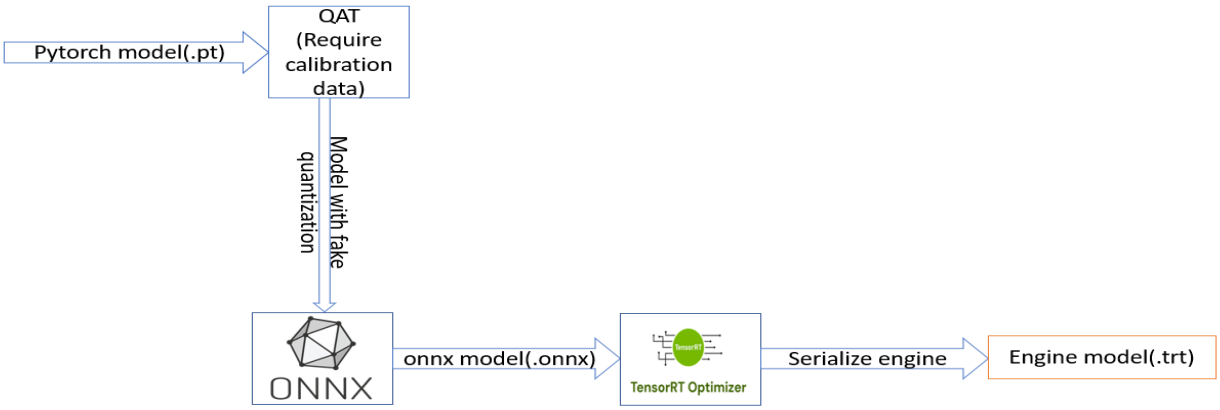
Giai đoạn này của dự án tập trung vào việc chuyển đổi mô hình sang định dạng tương thích với TensorRT. Việc này được thực hiện bằng cách xuất tệp Pytorch với phần mở rộng ".pt" sang định dạng ".ONNX", sau đó tạo ra tệp ".trt" từ định dạng này để có thể thực thi trên thiết bị Jetson Nano thông qua nền tảng TensorRT.

TensorRT hỗ trợ ba loại độ chính xác tính toán: độ chính xác đơn (FP32), độ chính xác nửa (FP16), và số nguyên có dấu 8-bit (INT8).

Người đọc quan tâm đến quy trình và cách mà TensorRT tối ưu được khuyến nghị xem tại [15]



Hình 10.1 : Quy trình tối ưu mô hình về độ chính xác FP16



Hình 10.1 : Quy trình tối ưu mô hình về độ chính xác INT8

Bảng 10.1 trình bày kích thước ,tốc độ xử lý khung hình FPS và mAP@0.50của mô hình trước và sau khi tối ưu, sử dụng GPU NVIDIA-T4.

Định dạng mô hình / Thông số	Pytorch(.pt) – FP32	ONNX(.onnx) – FP32	TensorRT – FP16	TensorRT – INT8
kích thước(Mb)	250	46	26.8	14.6
FPS	15	16	102	116
mAP@0.50	0.96	0.96	0.96	0.94

Bảng 10.1 : kích thước ,tốc độ xử lý khung hình FPS và mAP@0.50của mô hình trước và sau khi tối ưu, sử dụng GPU NVIDIA-T4

10.2 OPENVINO

Giai đoạn này của dự án tập trung vào việc chuyển đổi mô hình sang định dạng tương thích với OpenVINO. Việc này được thực hiện bằng cách xuất tệp Pytorch với phần mở rộng “.pt” sang định dạng “.ONNX”, sau đó tạo ra tệp “.bin” từ định dạng này để có thể thực thi trên thiết bị có CPU của Intel thông qua nền tảng OpenVINO.

OpenVINO hỗ trợ ba loại độ chính xác tính toán: độ chính xác đơn (FP32), độ chính xác nửa (FP16), và số nguyên có dấu 8-bit (INT8).

Bảng 10.2 trình bày kích thước , tốc độ xử lý khung hình FPS và mAP@0.50 của mô hình trước và sau khi tối ưu , sử dụng CPU Intel Xeon@2.3Ghz.

Định dạng mô hình / Thông số	Pytorch(.pt) – FP32	ONNX(.onnx) – FP32	OpenVINO(.bin) – FP16	OpenVINO(.bin) – INT8
kích thước(Mb)	250	46	26.7	14.6
FPS	5.2	5.2	7	9.2
mAP@0.50	0.96	0.96	0.96	0.94

Bảng 10.2: kích thước , tốc độ xử lý khung hình FPS và mAP@0.50 của mô hình trước và sau khi tối ưu , sử dụng CPU Intel Xeon@2.3Ghz

10.3 NHẬN XÉT

Có thể thấy việc tối ưu hóa mang lại lợi ích to lớn về mặt lưu trữ và tốc độ xử lý khung hình và chỉ cần đánh đổi một lượng rất nhỏ về phương diện độ chính xác.

CHƯƠNG 11 : TRIỂN KHAI MÔ HÌNH TRÊN HỆ THỐNG NHÚNG

11.1 HỆ THỐNG NHÚNG ĐƯỢC ĐỀ XUẤT

Tốc độ khung hình chính thức trong ADAS dao động từ 5 đến 60 FPS (khung hình/giây).

Trong các ứng dụng thực tế, hệ thống thường yêu cầu tốc độ khung hình cao hơn để theo dõi hiệu quả những thay đổi nhanh chóng về khoảng cách, nhưng để phát hiện biển báo giao thông, tốc độ khung hình thấp hơn 10 FPS được coi là đủ.

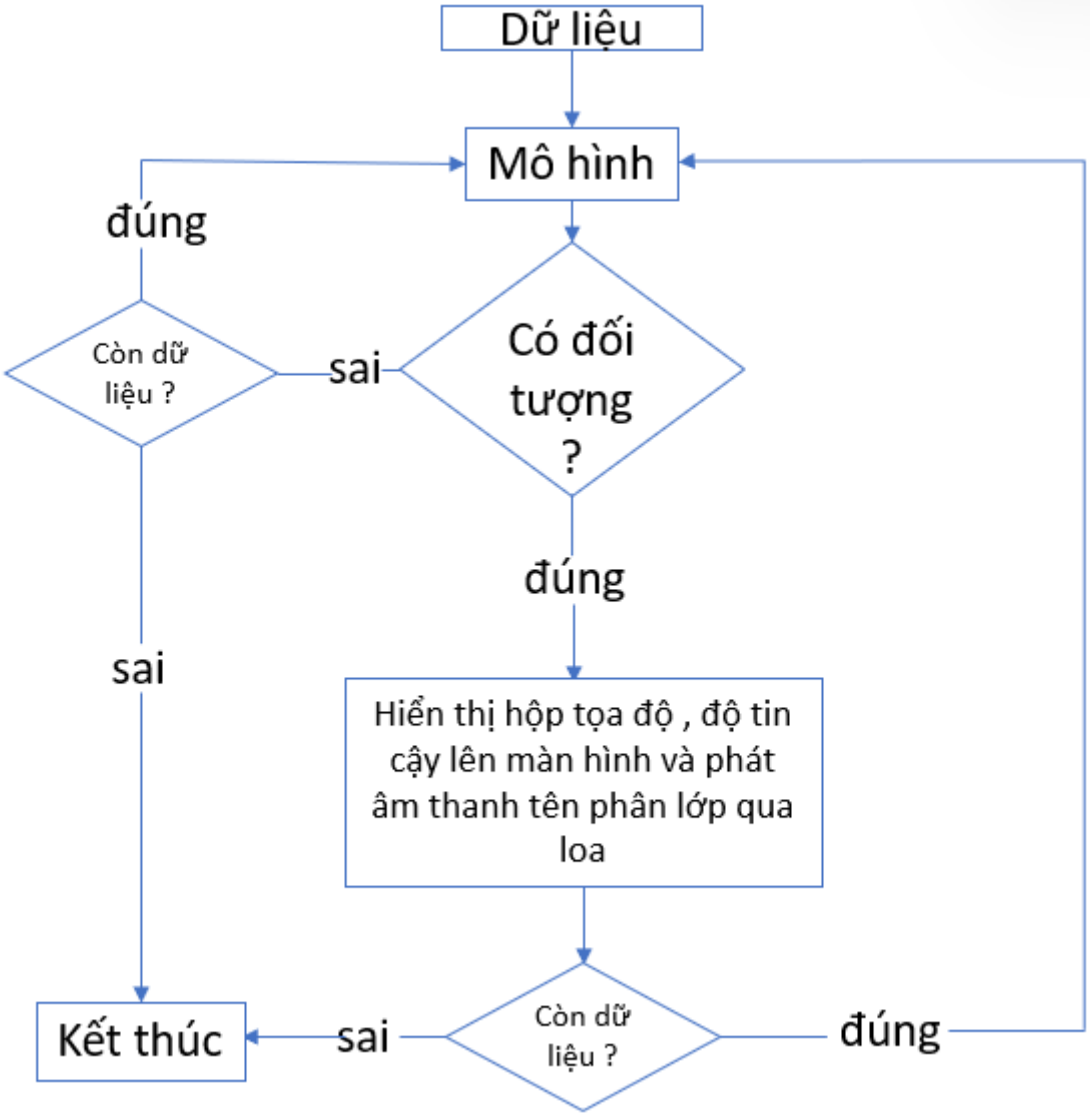
Máy tính nhúng NVIDIA Jetson Nano B01 4 GB, một nền tảng chuyên dụng được thiết kế riêng cho các ứng dụng AI và các giải pháp lấy hình ảnh làm trung tâm . Thiết bị này có CPU ARM A57 4 nhân @ 1,43 GHz, RAM LPDDR4 4 GB 64 bit và GPU Maxwell 128 nhân (Nhà sản xuất: NVIDIA; Quốc gia xuất xứ: Santa Clara, CA, Hoa Kỳ), những thông số này cùng với giá cả phải hợp lý khiến nó rất phù hợp để làm thử nghiệm cho dự án nhỏ , mang tính thử nghiệm khi mới bắt đầu triển khai

11.2 THIẾT LẬP MÔI TRƯỜNG THỬ NGHIỆM

Tiến hành cài đặt hệ điều hành jetpack và các thư viện openCV, tensorRT ,cuda và các thư viện liên quan.

Sau khi cài đặt đầy đủ các thư viện và thiết lập đường dẫn , tiến hành thực hiện tối ưu hóa mô hình YOLO NAS SMALL được huấn luyện trước đó về dạng tensorRT – FP16 (.trt)

11.3 NGUYÊN LÝ HOẠT ĐỘNG CỦA HỆ THỐNG



Hình 11.1: Nguyên lý hoạt động của hệ thống

11.4 KẾT QUẢ ĐẠT ĐƯỢC

Trên máy tính nhúng NVIDIA Jetson Nano B01 4 GB , mô hìnhYOLO NAS SMALL được tối ưu về dạng tensorRT – FP16 đạt được tốc độ xử lý khung hình 8.6 FPS.

CHƯƠNG 12 : KẾT LUẬN

Tóm lại, đồ án này giới thiệu một hệ thống nhúng được thiết kế để phát hiện biển báo giao thông tại Việt Nam theo thời gian thực . Giải pháp được đề xuất, máy tính nhúng NVIDIA Jetson Nano B01 , camera, loa , màn hình hiển thị cung cấp một giải pháp thay thế giá cả phải chăng, phù hợp với nhiều loại xe.

Mặc dù hệ thống đã chứng minh được độ chính xác khả quan, trên 90% , cùng với tốc độ suy luận 8.6 FPS, thấp hơn một chút so với mục tiêu đã đề ra, nhưng việc đưa ra kết luận chắc chắn một cách thận trọng là rất quan trọng do những hạn chế của bộ dữ liệu.

Bộ dữ liệu chỉ có 3917 mẫu và có tới 41 phân lớp biển báo khác nhau , với số lượng mẫu rất hạn chế so với lượng dữ liệu để huấn luyện một mô hình học sâu cần để có thể huấn luyện mô hình có khả năng khái quát cao.

Hệ thống chưa có khả năng hoạt động vào ban đêm hoặc điều kiện thiếu sáng trầm trọng do không có camera nhìn đêm để thu thập dữ liệu cũng như thực hiện thử nghiệm thực tế.

Các nỗ lực phát triển trong tương lai sẽ tập trung vào việc mở rộng cả bộ dữ liệu đào tạo và thử nghiệm, tiến hành thử nghiệm rộng rãi hơn để xác thực và tinh chỉnh thêm khả năng của hệ thống, bên cạnh đó việc đề xuất nâng cấp phần cứng để tăng cường khả năng xử lý của hệ thống nhằm hướng tới mục tiêu thương mại hóa là một điều rất đáng cân nhắc .

TÀI LIỆU THAM KHẢO

[1] L. A. ,. J. M. Kornel Sarvajcz, *AI on the Road: NVIDIA Jetson Nano-Powered Computer Vision-Based System for Real-Time Pedestrian and Priority Sign Detection,* no. Computing and Artificial Intelligence, 2024.

[2] T. V. T. Turay, *Toward Performing Image Classification and Object Detection with Convolutional Neural Networks in Autonomous Driving Systems.*

[3] A. Krizhevsky, I. Sutskever and G. Hinton, *Imagenet classification with deep convolutional neural networks. Commun.*

[4] S. Ren, K. He, R. Girshick and J. Sun, *Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015).*

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and C. Berg, *SSD: Single Shot MultiBox Detector,* 2016.

[6] S. D. R. G. A. F. Joseph Redmon, *You Only Look Once: Unified, Real-Time Object Detection.*

[7] D.-M. C.-E.-A. R.-G. Juan Terven, *A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,* 2023.

[8] P. D. R. G. K. H. B. H. S. B. Tsung-Yi Lin, *Feature Pyramid Networks for Object Detection,* 2017.

[9] J. W. X. F. ,. T. Y. G. W. Zhanchao Huang, *DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection.*

[10] L. L. B. Z. Xiangxiang Chu, *Make RepVGG Greater Again: A Quantization-aware Approach,* 2023.

[11] H.-Y. M. L.-H. Y.-H. W.-Y. C.-W. H. Chien-Yao Wang, *CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING,* 2019.

[12] J. Z. Z. X. B. &. N. H. Qunpo Liu, *A YOLOX Object Detection Algorithm Based on Bidirectional Cross-scale Path Aggregation,* 2024.

[13] W. W. W. S. C. X. H. L. T. Y. Xiang Li, *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection.*

[14] J. L. B. Diederik P. Kingma, *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,* 2015.

[15] [Online]. Available: https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9431-tensorrt-inference-with-tensorflow.pdf.