

Mục lục

1	Mở đầu	1
2	Inference	2
2.1	Data Preparation	2
2.1.1	Dữ liệu đầu vào	2
2.1.2	Tiền xử lý	2
2.2	Các bước Inference	3
3	Mạng nơ-ron Tích chập (CNN)	4
3.1	Khái niệm	4
3.2	Các bước đi qua	4
3.3	Cách CNN hoạt động	5
3.4	Phân biệt model và backbone	5
4	Mô hình YOLOv11 Nano pretrain	6
4.1	Kiến trúc mô hình	6
4.1.1	Nhận xét	7
4.2	Đánh giá mô hình (Evaluate)	8
4.2.1	Định nghĩa các metrics	8
4.2.2	Kết quả evaluation trên YOLOv11 Nano Pretrain của jahongir7174	9
4.2.3	Đánh giá	11

jahongir7174 YOLOv11-Pretrain

Trương Đỗ Vương

Ngày 8 tháng 10 năm 2025

Chương 1

Mở đầu

Bài báo cáo này nhằm phân tích và đánh giá mô hình [YOLOv11 Nano Pretrain](#) của jahongir7174, với các tiêu chí sau:

1. Data Preparation (Data type - continuous, discrete) – Data này ở dạng format gì: ma trận, float, int,....
2. Khái niệm: CNN - RNN
 - Các bước đi qua như thế nào.
 - Phân biệt model và backbone.
3. Mô hình
 - Phân tích các lớp của mô hình với đầu vào đơn giản.
 - Đánh giá mô hình pretrain trên.

Chương 2

Inference

2.1 Data Preparation

2.1.1 Dữ liệu đầu vào

- Một hoặc nhiều ảnh bất kỳ với kiểu biểu diễn Raster (pixel based) bao gồm:
 1. PNG, TIFF, JPEG, BMP, WebP, EXR, HDR
 2. HEIC/HEIF (Thường dùng cho Iphone/Ipad/Mac)
- Không hỗ trợ kiểu biểu diễn vector như: SVG hoặc có ảnh động (animation) như GIF.
- Hỗ trợ ảnh đầu vào ở nhiều kích thước.

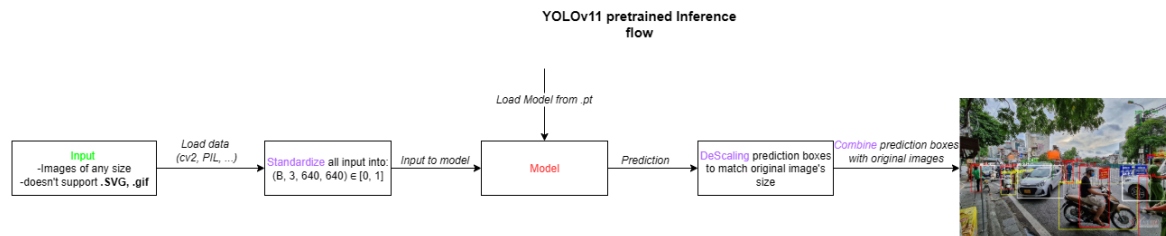
2.1.2 Tiền xử lý

Các ảnh Raster thường là một ma trận số nguyên (uint8), mỗi pixel $\in [0, 255]$

- Chỉ lấy 3 kênh màu: sử dụng *OpenCV* (BGR) hoặc *PIL* (RGB), với $\text{Shape} = (H, W, 3)$.
- Chuẩn hóa kích thước: [YOLOv11 Nano Pretrain](#) yêu cầu Tensor đầu vào có $\text{Shape} = (3, 640, 640)$. Ảnh gốc được *resize* (giữ nguyên tỉ lệ khung hình) và *padding* bằng hàm *letterbox* để vừa khung 640×640 .
- Chuẩn hóa Tensor:
 1. $\text{uint8} \rightarrow \text{float16}$ vì mô hình chạy ở chế độ half precision.
 2. $\text{pixel} \in [0, 255] \rightarrow [0, 1]$

2.2 Các bước Inference

Dưới đây là minh họa súc tích cho YOLOv11 pretrain inference flow



Hình 2.1: Minh họa YOLOv11 pretrain inference

Chương 3

Mạng nơ-ron Tích chập (CNN)

3.1 Khái niệm

Là một loại kiến trúc nơ-ron, thường bao gồm một chuỗi các lớp, dùng để biến đổi hình ảnh đầu vào (với các giá trị pixel) thành các đặc trưng đã được trích xuất.

- Trong CNN, có nhiều thành phần khác nhau:
 1. Convolution layer: lớp trích xuất đặc trưng từ ảnh.
 2. Pooling layer: dùng để giảm kích thước không gian (DownSampling).
 3. Upsampling/ Transposed Convolution: dùng để tăng kích thước (phóng to) đặc trưng.

3.2 Các bước đi qua

Một mô hình CNN đi qua các bước chính:

- **Input:** ảnh dạng tensor:
 - Pytorch: $(3, H, W)$
 - TensorFlow, Keras: $(H, W, 3)$
- **Convolution:** quét ảnh bằng các kernel/filter để trích xuất đặc trưng cục bộ.
- **Activation:** thêm hàm phi tuyến tính (ReLU, SiLU, ...).
- **Pooling/Stride:** giảm kích thước không gian (H, W) , giữ lại đặc trưng quan trọng.
- **Head:** lớp cuối biến đặc trưng thành dự đoán
 - **Phân loại:** xác suất $\in [0, 1]$ (thường sử dụng softmax hoặc sigmoid).
 - **Bounding box:** tập các tọa độ $[(x_a, y_a, x_b, y_b), \dots]$, trong đó (x_a, y_a) là góc trên-trái, (x_b, y_b) là góc dưới-phải.

3.3 Cách CNN hoạt động

Các lớp tích chập sử dụng nhiều bộ lọc có thể học được (kernels) lướt trên đầu vào để tính tích vô hướng (dot product) và sinh các đặc trưng (feature maps); mỗi bộ lọc tạo ra một kênh trong tensor đầu ra.

Chú ý rằng: nhiều hướng dẫn trình bày cơ chế hoạt động của lớp tích chập, lớp pooling hay upsampling như các “sliding windows”. Tuy nhiên, hiện thực trực tiếp việc “lướt” bằng nhiều vòng lặp sẽ rất chậm, thậm chí không khả thi do số vòng lặp quá lớn. Thay vào đó, người ta thường chuyển đổi đầu vào và các kernel thành một ma trận lớn bằng các thuật toán như `im2col-col2im`, `im2row-row2im`, `kn2row-row2kn` hay `kn2col-col2kn` và tính toán như một lớp tuyến tính (phép nhân ma trận/GEMM), tận dụng tối đa tối ưu của thư viện BLAS để tăng tốc.

3.4 Phân biệt model và backbone

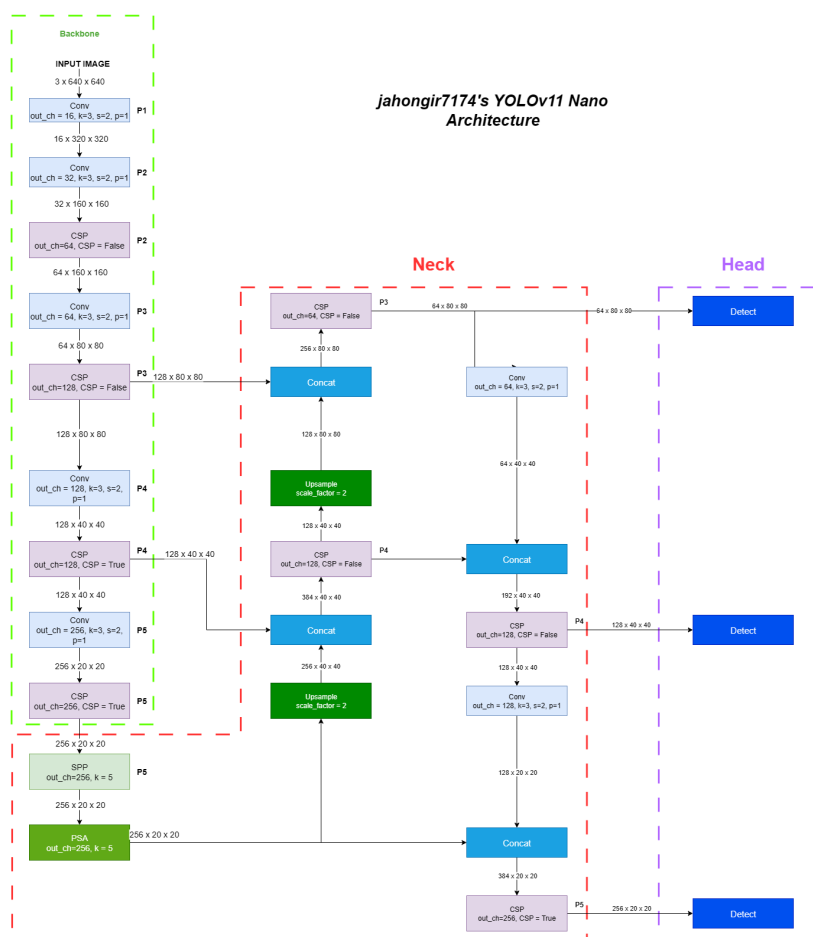
- **Model:** toàn bộ kiến trúc cho một tác vụ cụ thể (classification, detection, segmentation). Bao gồm **backbone + neck + head**.
- **Backbone:** phần *xương sống*, chuyên để trích xuất đặc trưng.
- **Neck:** Tổng hợp các đặc trưng khác nhau từ nhiều phần của backbone. trong YOLO, backbone gồm các lớp Conv, C2f, CSP, ...
- **Head:** phần cuối đưa ra dự đoán (class scores, bounding box, mask).

Chương 4

Mô hình YOLOv11 Nano pretrain

4.1 Kiến trúc mô hình

Dưới đây là minh họa kiến trúc mô hình với *một* ảnh đầu vào input ($B = 1$).



Hình 4.1: Chi tiết tại [net.py](#)

4.1.1 Nhận xét

Mô hình YOLOv11 Pretrain của [jahongir7174 yolov11](#) có những điểm khác biệt nổi bật so với mô hình gốc của [ultralytics yolov11](#) về kiến trúc.

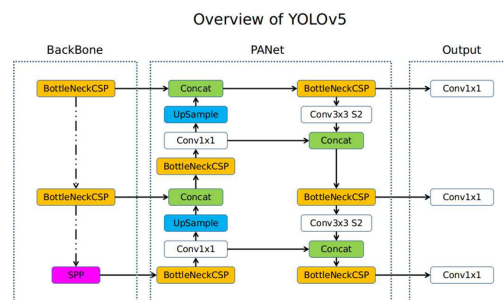
Theo như [source code](#) và bài review của Nidhal Jegham và các cộng sự [1](trang 8). Có 3 bộ phận khác nhau, gồm:

ultralytics	jahongir7174
C2K3	CSP
SPPF	SPP
C2PSA	SPA

Bảng 4.1: Các thành phần khác nhau giữa 2 mô hình

Theo đó:

- CSP (Cross Stage Partial) vs C2K3/C3k2:
 - CSP (Cross Stage Partial) (được dùng trong YOLOv5)



Hình 4.2: Kiến trúc mô hình YOLOv5, [Nguồn](#)

- YOLOv11 thay thế CSP bằng C3k2 nhằm cải thiện độ hiệu quả.
- SPP vs SPPF (Spatial Pyramid Pooling). Phiên bản được dùng trong mô hình của ultralytics thực chất là SPP-Fast. Và SPP cũng có nguồn gốc từ YOLOv5.
 - SPPF cũng được áp dụng cho YOLOv5 sau này thay thế SPP, với tốc độ gấp 2 lần và đầu ra được bảo toàn so với SPP:

"The SPP structure is replaced with SPPF. This alteration more than doubles the speed of processing while maintaining the same output."

- SPA vs C2PSA (Spatial Attention): SPA có thể được đặt với nghĩa là "1 PSA" khi kiến trúc gốc cho thấy, nó có 2-PSA.
 - Điều đó cho biết tác giả jahongir7174 đã cố ý "đơn giản hóa" mô hình của mình so với ultralytics.

Kết luận

Dù là có cùng flow, nhưng mô hình YOLOv11 pretrain của [jahongir7174 yolov11](#) là phiên bản được đơn giản hóa so với của [ultralytics yolov11](#)

Lý do cho điều này có thể là vì tác giả chỉ muốn tự tìm hiểu YOLOv11 qua huấn luyện và đánh giá mô hình mới tới một mức độ nhất định.

4.2 Đánh giá mô hình (Evaluate)

Để đánh giá một mô hình YOLO, người ta thường dùng các metric như: Average Precision (AP), Average Recall (AR).

4.2.1 Định nghĩa các metrics

IoU (Intersection over Union)

- Phần diện tích mà gt(ground-truth) và dự đoán trùng nhau, chia (\div) cho tổng diện tích của gt + predicted
- IoU cao \rightarrow box dự đoán càng chính xác so với box label (gt)

AP (Average Precision)

- AP đo lường chất lượng tổng thể của mô hình object detection, kết hợp cả **precision** và **recall**.
- **Precision** = tỉ lệ các box dự đoán đúng so với tổng số box dự đoán:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** = tỉ lệ các object thực sự được phát hiện:

$$\text{Recall} = \frac{\text{True Positives}}{\text{Tổng số GT objects}}$$

- AP được tính bằng cách lấy diện tích dưới đường cong **Precision–Recall**:

$$\text{AP} = \int_0^1 \text{Precision}(\text{Recall}) d \text{ Recall}$$

- Trong COCO, AP thường được tính ở nhiều mức IoU:
 - **AP@[0.50:0.95]**: trung bình AP qua IoU từ 0.50 đến 0.95 (strict, đánh giá chính xác box)

- **AP@0.50**: $\text{IoU} \geq 0.50$
- **AP@0.75**: strict, yêu cầu $\text{IoU} \geq 0.75$
- **AP_small / AP_medium / AP_large**: AP tính riêng theo kích thước vật thể.
- Ý nghĩa của AP:
 - AP cao \rightarrow mô hình vừa phát hiện đúng với nhiều đối tượng, vừa có các box chính xác.
 - AP thấp \rightarrow mô hình bỏ sót nhiều đối tượng hoặc dự đoán nhiều box sai (false positives), hoặc box kém chất lượng.

AR (Average Recall)

- **AR (Average Recall)** đo khả năng mô hình tìm ra tất cả các object trong ảnh (sensitivity). Lấy trung bình recall trên nhiều ngưỡng IoU.
- AR được tính giống như AP nhưng tập trung vào **recall**, không quan tâm nhiều đến precision:

$$\text{AR} = \frac{1}{N_{\text{IoU}}} \sum_{i=1}^{N_{\text{IoU}}} \text{Recall@IoU}_i$$

- **maxDets** = số lượng box dự đoán tối đa được sử dụng khi tính AR:
 - **maxDets=1**: chỉ dùng box dự đoán cao điểm nhất mỗi ảnh.
 - **maxDets=10**: sử dụng 10 box dự đoán cao điểm nhất.
 - **maxDets=100**: sử dụng 100 box dự đoán cao điểm nhất.
- Ý nghĩa:
 - AR cao \rightarrow mô hình tìm được hầu hết các vật thể trong ảnh.
 - Nếu AR@maxDets=1 thấp nhưng AR@maxDets=100 cao \rightarrow mô hình cần nhiều dự đoán để tìm hết vật thể.
 - AR theo kích thước (small / medium / large) giúp đánh giá khả năng phát hiện các vật thể theo từng kích thước: nhỏ, trung bình, lớn.

4.2.2 Kết quả evaluation trên YOLOv11 Nano Pretrain của jahongir7174

Sử dụng COCO API và [COCO 2017 val dataset](#) cho quá trình evaluation, dưới đây là bảng kết quả đánh giá mô hình YOLOv11 với tập tham số pretrain

Metric	IoU	Area	MaxDets	Value
Mean Average Precision (mAP)	0.50:0.95	all	100	0.325
Average Precision (AP)	0.50	all	100	0.443
Average Precision (AP)	0.75	all	100	0.355
Mean Average Precision (mAP)	0.50:0.95	small	100	0.119
Mean Average Precision (mAP)	0.50:0.95	medium	100	0.352
Mean Average Precision (mAP)	0.50:0.95	large	100	0.512
Average Recall (AR)	0.50:0.95	all	1	0.269
Average Recall (AR)	0.50:0.95	all	10	0.371
Average Recall (AR)	0.50:0.95	all	100	0.373
Average Recall (AR)	0.50:0.95	small	100	0.131
Average Recall (AR)	0.50:0.95	medium	100	0.400
Average Recall (AR)	0.50:0.95	large	100	0.593

Bảng 4.2: YOLOv11 Nano Pretrain với COCO evaluation metrics (chi tiết tại: [evaluation.ipynb](https://github.com/ultralytics/ultralytics/blob/main/docs/evaluation.ipynb))

COCO báo cáo nhiều metric để đánh giá hiệu năng mô hình dưới các góc nhìn khác nhau: ngưỡng IoU, kích thước đối tượng, và giới hạn số detections.

- **AP (0.50:0.95, all, 100) = 0.325**

Đây là *mAP chính* của COCO: mô hình đạt $\approx 32.5\%$ khi lấy trung bình trên IoU từ 0.50 đến 0.95. Dùng để so sánh tổng quát giữa các mô hình trên COCO.

- **AP (0.50, all, 100) = 0.443 và AP (0.75, all, 100) = 0.355**

AP@0.5 lớn hơn AP@[0.5:0.95], cho thấy mô hình thường phát hiện đối tượng ở mức IoU thấp tốt, nhưng khi yêu cầu độ chính xác hơn (IoU cao), thì hiệu năng giảm.

- **AP (0.50:0.95, small, 100) = 0.119**

Rất thấp cho đối tượng nhỏ — mô hình gặp khó khăn với small objects.

- **AP (0.50:0.95, medium, 100) = 0.352, AP (0.50:0.95, large, 100) = 0.512**

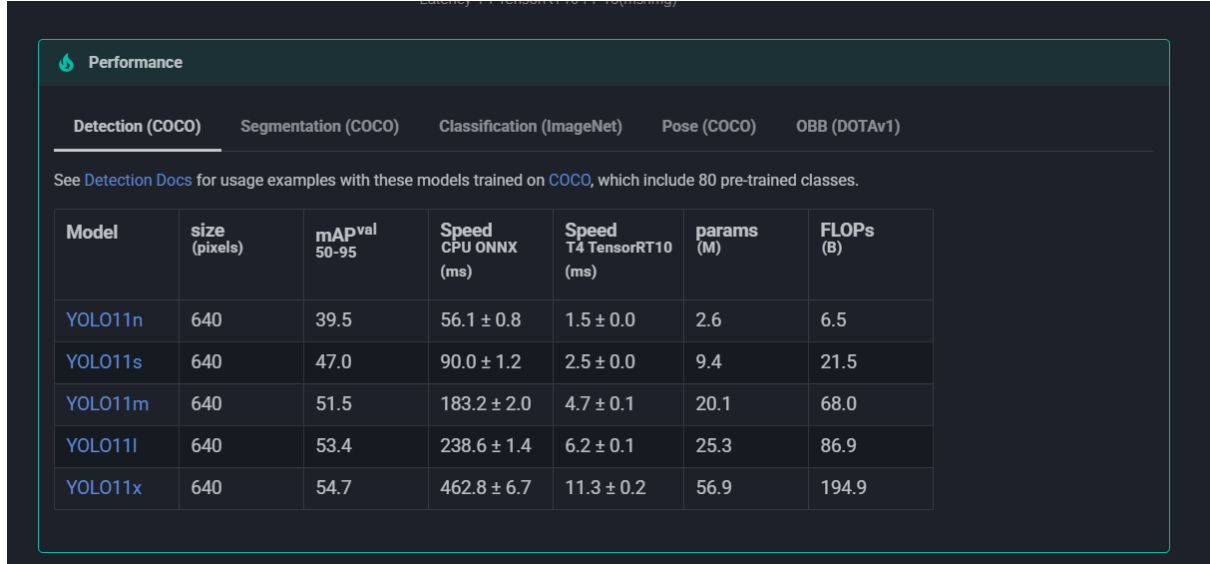
Mô hình hoạt động tốt với medium và đặc biệt tốt với large objects.

- **AR (0.50:0.95, all, MaxDets)** với MaxDets = 1, 10, 100 là 0.269, 0.371, 0.373

Tăng MaxDets từ 10 lên 100 gần như không cải thiện recall (0.371 \rightarrow 0.373), nghĩa là hầu hết các object đúng đã có trong top-10 predictions.

4.2.3 Đánh giá

Mô hình trên có hiệu năng thấp hơn so với phiên bản YOLOv11 cùng kích cỡ, và trên cùng 1 tập dataset (COCO 2017 val) của ultralytics.



The screenshot shows the 'Performance' tab in the Ultralytics interface. It features a table with columns for Model, size (pixels), mAP^{val}₅₀₋₉₅, Speed CPU ONNX (ms), Speed T4 TensorRT10 (ms), params (M), and FLOPs (B). The table lists five models: YOLO11n, YOLO11s, YOLO11m, YOLO11l, and YOLO11x, with their respective metrics. A note above the table states: 'See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.'

Model	size (pixels)	mAP ^{val} ₅₀₋₉₅	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Hình 4.3: Evaluation của YOLOv11 từ ultralytics

Dù là chỉ có metric tổng thể: mAP_{50-95}^{val} , nhưng so ánh với nhau, ta có thể thấy được độ chênh lệch tổng thể giữa 2 mô hình.

$$\begin{cases} \text{jahongir yolov11 nano : } mAP_{50-95}^{val} = 0.325 \\ \text{ultralytics yolov11 nano } mAP_{50-95}^{val} = 0.395 \end{cases}$$

Điều đó cho thấy rằng dù giống nhau về flow nhưng việc sử dụng các kiến trúc cũ hơn (CSP, SPP, SPA) đã có tác động rất lớn đến hiệu năng của mô hình nói chung.

Tài liệu tham khảo

- [1] Nidhal Jegham **and others**. *YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions*. 2025. arXiv: [2411.00201](https://arxiv.org/abs/2411.00201) [cs.CV]. URL: <https://arxiv.org/abs/2411.00201>.