Hands-On Object Detection Workshop with YOLO & Transformers

From Fundamentals to Fine-Tuning State-of-the-Art Models

Indranil Bhattacharyya

Data Scientist,Renault Group

Workshop of **Shaastra'26**, IIT Madras

January 4, 2026

Workshop Roadmap

Session 1: The Core (Theory)

- Fundamentals: IoU, NMS
- YOLO Architecture (CNNs)
- Transformers & DETR
- Comparative Analysis

Session 2: The Practice (Hands-On)

- Custom Dataset Workflow
- Annotation (Roboflow)
- Training in Google Colab
- Evaluation & Deployment

Defining the Problem: Beyond Classification

Classification vs. Detection

- **Classification:** "What is in this image?" (Single label)
- **Object Detection:** "What is where?"
  - Localization + Classification
  - Output: $[x, y, w, h, class, confidence]$

Goal: Predict a bounding box and class for every object of interest.

Core Metrics

1. Intersection over Union (IoU)

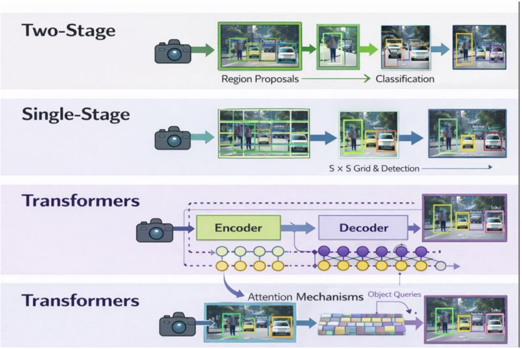
- Fundamental localization metric.
- $IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$
- Measures how well predicted boxes overlap with ground truth.

2. Mean Average Precision (mAP)

- Primary evaluation metric for object detection.
- Combines **precision** and **recall** across classes.
- Computed over multiple IoU thresholds (e.g., mAP@0.5, mAP@0.5:0.95).

Evolution of Architectures

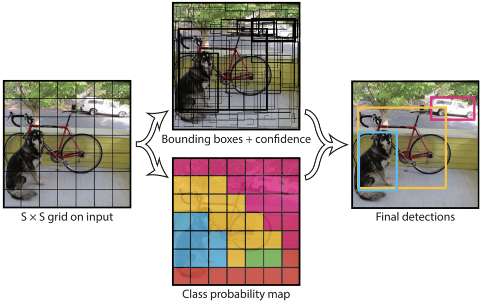
- **Two-Stage Detectors**
  - Region proposals → Classification
  - Accurate but slow
- **Single-Stage Detectors**
  - Pixels → Boxes (Direct regression)
  - Real-time inference
- **Transformer-Based Detectors**
  - Attention + Object Queries
  - End-to-end set prediction



YOLO: You Only Look Once

Core Concept: Reframe detection as a single regression problem.

- **Grid System:** Image split into $S \times S$ grid.
- **Responsibility:** Cell predicts object if center falls inside.
- **Speed:** 45–155 FPS (Real-time).



YOLO Architecture

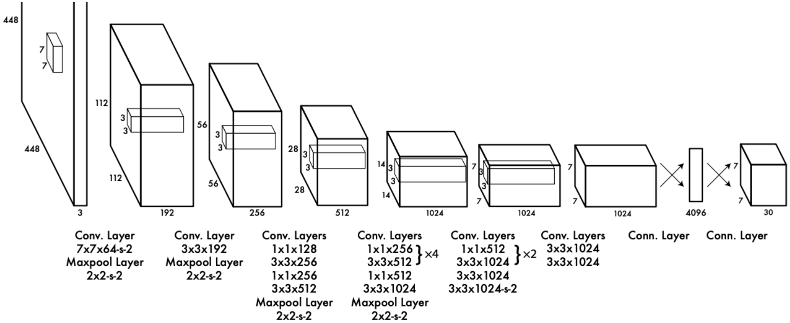


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Deep Dive: The YOLO Output Tensor

Each grid cell predicts B bounding boxes.

Prediction Vector per Box

$$y = [P_c, b_x, b_y, b_w, b_h, c_1, c_2, \dots, c_n]$$

- P_c: Objectness Score (Is there an object?)
- b_x, b_y: Center coordinates (relative to cell)
- b_w, b_h: Dimensions (relative to anchors)
- $c_1 \dots c_n$: Class probabilities

Final Tensor Shape: $S \times S \times B \times (5 + C)$

Deep Dive: Anchors & Constraints

Anchor Boxes

- Pre-defined shapes (tall, wide).
- Model predicts *offsets*, not raw sizes.
- It learns to adjust the closest "template" box.

Logistic Constraint

$$b_x = \sigma(t_x) + c_x$$

- The sigmoid function σ forces predictions to stay within the grid cell $(0 - 1)$.
- This prevents unstable gradients during training.

Transformers Enter Vision

"Attention is All You Need"

- **CNN Limitation:** Local receptive fields (sliding windows). CNNs struggle to see relationships between distant pixels.
- **Transformer Strength:** Self-Attention captures **global context**.
- Every pixel can attend to every other pixel effectively.
- Removes need for hand-crafted NMS or Anchor Engineering.

DETR: DEtection TRansformer

Core Idea: End-to-end object detection using attention.

- **Backbone (CNN):** ResNet-50 extracts feature maps.
- **Encoder:** Self-attention builds global context.
- **Decoder:** **Object Queries** attend to features.
- **Prediction Heads:** FFN outputs box + class.

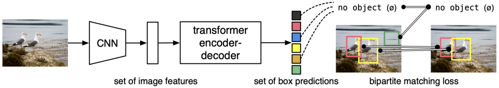


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.

DETR Architecture

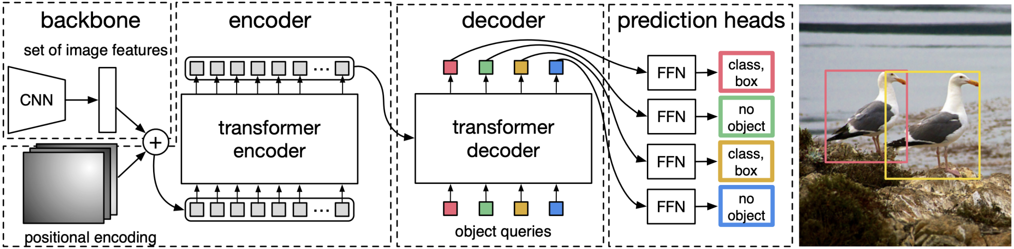


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Deep Dive: Object Queries

What are they?

- A fixed set of N (e.g., 100) learnable embeddings.
- They act as "slots" that the model tries to fill.

The Detective Analogy:

- Imagine 100 detectives sent into a room.
- Detective 1 asks: "Is there something large in the center?"
- Detective 2 asks: "Is there a small object in the corner?"
- They learn these questions during training.

Deep Dive: Bipartite Matching (Hungarian Algorithm)

The Problem

Model outputs 100 predictions. Image has only 3 objects.

The Solution

Set Prediction Loss:

- Find the optimal **One-to-One** matching between Predictions and Ground Truth.
- Minimizes cost (Class Prob + Box Distance).
- Result: No duplicates $\rightarrow$ **No NMS needed**.

YOLO vs. DETR

Feature	YOLO (CNN)	DETR (Transformer)
Mechanism	Grid, Anchors	Attention, Queries
Speed	Real-time (Fast)	Slower
Training	Fast convergence	Long training needed
Post-Process	NMS required	None (End-to-End)

Session 2: Custom Workflow

1. **Data Collection:** Diversity is key.
2. **Annotation:** Draw boxes.
    - Format: `class x_center y_center width height`
    - Normalized (0 to 1).
3. **Training:** Transfer Learning on GPU.
4. **Validation:** mAP metrics.

Hands-On Lab Plan

- **Platform:** Google Colab (Free GPU).
- **Dataset:** Roboflow Universe (Custom Data).
- **Model:** Ultralytics YOLOv8.
- **Task:**
  - Load Data.
  - Train for 20 epochs.
  - Run Inference on video.

Questions?

Let's build some detectors.

Submit Your Feedback