Object Detection Metrics

what all are the metrics in object detection

In **object detection**, the primary goal is to **detect and localize objects** in an image. To evaluate performance, several metrics are used—some measuring **localization accuracy**, others **classification accuracy**, and some **overall performance**.

Here's a structured breakdown of the **key object detection metrics**:

🔹 1. Intersection over Union (IoU)

- **Definition**: Measures the overlap between predicted bounding box and ground truth box.
- **Formula**:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

- **Usage**: Used to determine if a prediction is a **True Positive (TP)**. Common thresholds: `IoU > 0.5` or stricter (e.g., `0.75`).

🔹 2. Precision

- **Definition**: Fraction of correctly predicted boxes over all predicted boxes.
- $$\text{Precision} = \frac{TP}{TP + FP}$$

🔹 3. Recall

- **Definition**: Fraction of correctly predicted boxes over all ground truth boxes.
- $$\text{Recall} = \frac{TP}{TP + FN}$$

🔹 4. Average Precision (AP)

- **Definition**: Area under the Precision-Recall curve.
- Calculated for each class separately.
- Depends on IoU threshold (e.g., `AP@0.5`, `AP@0.75`).
- **Interpolated AP**: Often smoothed over recall levels (e.g., Pascal VOC).

🔹 5. mean Average Precision (mAP)

- **Definition**: Mean of Average Precision across all classes.
- **Variants**:
    - **mAP@0.5 (Pascal VOC)**: AP at IoU threshold of 0.5.
    - **mAP@[.5:.95] (COCO)**: Average AP at multiple IoU thresholds (0.5 to 0.95 with 0.05 step).

🔹 6. F1 Score

- Harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

🔹 7. AP Small / Medium / Large (COCO specific)

- Measures AP on different object sizes:
    - `AP_small`: Area < 32^2
    - `AP_medium`: Area in [32^2, 96^2]
    - `AP_large`: Area > 96^2

🔹 8. Number of Detections per Image

- Helps evaluate model behavior—too few: under-detecting; too many: over-detecting.

🔹 9. Inference Time / FPS

- **Latency** and **Throughput** are critical in real-time systems.

🔹 10. False Positive (FP) and False Negative (FN) Rates

- Important for analyzing errors and tradeoffs between over-detection and under-detection.

🟩 Example (COCO Evaluation Output):

```txt
 AP: 0.367
AP50: 0.582
AP75: 0.391
AP_small: 0.219
AP_medium: 0.400
AP_large: 0.494
```

If you're building or evaluating a model, let me know your use case (real-time? class imbalance? small objects?), and I can recommend the most relevant metrics and thresholds.

what is focal loss and why use it. explain math

Focal Loss is a loss function designed to address **class imbalance** in tasks like **object detection**, particularly in **dense detectors** like RetinaNet. It is a modified version of the **Cross Entropy Loss**, tailored to **focus more on hard, misclassified examples** and **down-weight the easy ones**.

🔸 Why Focal Loss?

In dense object detection:

- Most anchor boxes are **background** (negative class).
- Standard **Cross-Entropy Loss** treats all examples equally → model is overwhelmed by easy negatives → poor performance on rare/hard positives.

👉 Focal Loss dynamically **scales down the loss from easy examples**, focusing learning on hard misclassified ones.

🔸 Standard Binary Cross Entropy Loss

$$\text{CE}(p, y) = -y \cdot \log(p) - (1 - y) \cdot \log(1 - p)$$

where:

- $p \in [0, 1]$ is the model's estimated probability for the class with label $y \in \{0, 1\}$

🔸 Focal Loss: Formula

Let:

- $p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$

  → The probability of the **true class**

Then **Focal Loss** is:

$$\boxed{FL(p_t) = -\alpha_t \cdot (1 - p_t)^\gamma \cdot \log(p_t)}$$

🔸 Explanation of Terms

Term	Meaning
p_t	Probability of the correct class
$\log(p_t)$	Log loss as in standard cross-entropy
$(1 - p_t)^\gamma$	Modulating factor that reduces the loss for well-classified examples
γ	Focusing parameter $\gamma \geqslant 0$, typically 2.0
α_t	Class balancing factor (optional), helps balance positive/negative samples

---

🔸 Behavior of Focal Loss

Case	p_t	$(1 - p_t)^\gamma$	Effect on Loss
Easy example (correct, confident)	~1.0	~0	Loss → small (ignored)
Hard example (wrong or low confidence)	~0.0	~1	Loss → large (focused on)

🔸 Graphical Intuition

With Cross-Entropy:

- Loss is always high for incorrect predictions.
- Easy negatives dominate total loss.

With Focal Loss:

- Loss is **suppressed** when p_t is high (easy).
- Loss is **amplified** when p_t is low (hard).

🔸 Class-Balanced Focal Loss (Optional)

Add class weights $\alpha_t \in [0, 1]$ to handle class imbalance:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

- In RetinaNet, $\alpha = 0.25$, $\gamma = 2.0$

🔸 Use Case in RetinaNet

In **RetinaNet**, focal loss is critical to:

- Handle **millions of background anchors**
- Focus training on **rare positive detections**

Without focal loss, Cross Entropy causes instability and underperformance.

🔸 Summary Table

Feature	Focal Loss
Based on	Cross-Entropy
Solves	Class imbalance in object detection
Key idea	Down-weight easy examples
Parameters	γ: focusing, α: class weight
Use case	RetinaNet, single-shot detectors

Would you like a PyTorch or TensorFlow implementation next?

explain the need of focal loss and the mathematics in depth

Let’s go **in-depth into the motivation and math behind Focal Loss**, including why it was created, how it works mathematically, and how it improves performance in **imbalanced classification tasks**, especially in **dense object detection** like in RetinaNet.

🔸 1. Problem: Class Imbalance in Object Detection

◾ Scenario:

In dense detectors (e.g., RetinaNet, SSD, YOLO):

- Each image may have **thousands of anchor boxes**.
- Only a **tiny fraction** (~1%) of anchors match a ground truth object (positive samples).
- The **vast majority** are **easy negatives** (background class, correctly classified with high confidence).

◾ What goes wrong with Cross-Entropy Loss:

- Each of those easy negatives contributes equally to the total loss.
- The gradient is dominated by these easy samples.
- Model becomes **biased towards background**, ignoring hard or rare positive examples.

🔸 2. Standard Binary Cross Entropy (BCE)

$$\mathrm{L}_{\mathrm{BCE}}(p, y) = -y \cdot \log(p) - (1 - y) \cdot \log(1 - p)$$

Where:

- $y \in \{0, 1\}$ is the ground truth label
- $p \in (0, 1)$ is the model’s predicted probability for class 1

This loss treats all examples **equally**, regardless of how well the model is already doing.

🔸 3. Focal Loss: Core Idea

To **focus learning on hard examples** and **down-weight easy ones**.

🔸 4. Rewriting BCE for Simplification

Let's define:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$$

So BCE becomes:

$$L_{\text{BCE}} = -\log(p_t)$$

🔸 5. Focal Loss Formulation

Focal Loss introduces a modulating factor $(1 - p_t)^{\gamma}$:

$$L_{\text{Focal}}(p_t) = -\alpha_t \cdot (1 - p_t)^{\gamma} \cdot \log(p_t)$$

◾ Terms explained:

- $\alpha_t \in [0, 1]$: balancing factor for positive/negative classes
- $\gamma \geqslant 0$: focusing parameter
- $(1 - p_t)^{\gamma}$: reduces loss contribution from **easy examples**

🔸 6. Mathematical Behavior

Let's examine how focal loss behaves:

✅ When $p_t \to 1$ (easy sample):

- $(1 - p_t)^{\gamma} \to 0$
- → **Loss is nearly zero**

✅ When $p_t \to 0$ (hard sample):

- $(1 - p_t)^\gamma \to 1$
- → **Loss is large**

🔸 7. Effect on Gradient

◾ Cross-Entropy:

$$\frac{d}{dp}\mathrm{L}_{\mathrm{BCE}} = -\frac{y}{p} + \frac{1-y}{1-p}$$

◾ Focal Loss (Derivative with respect to p):

If $y = 1$, then $p_t = p$, and:

$$\frac{d}{dp}\mathrm{L}_{\mathrm{Focal}} = \alpha(1-p)^\gamma \left[\frac{\gamma p \log(p)}{1-p} - \frac{1}{p}\right]$$

This gradient is:

- **Small for confident predictions (p → 1)**: suppresses their impact.
- **Large for poor predictions (p → 0)**: focuses optimization here.

🔸 8. Why It Works: Intuition

Term	Role
$\log(p_t)$	Standard classification loss term
$(1 - p_t)^\gamma$	Suppresses well-classified examples
α_t	Class-balancing to correct for skewed distributions
Combined effect	Forces model to focus on **misclassified**, **rare**, and **hard** samples

---

🔸 9. Hyperparameters

Parameter	Effect	Common Value (RetinaNet)
γ	Focusing strength (↑γ = ↑focus on hard)	2.0
α	Class balance (for positives/negatives)	0.25 for positive class

🔸 10. Use Case in RetinaNet

RetinaNet proposes **Focal Loss** as key innovation:

- Without it, one-stage detectors (no proposal mechanism like in Faster R-CNN) underperform.
- With it, RetinaNet becomes SOTA on COCO dataset.

🔸 11. Comparison with Cross-Entropy

Property	Cross-Entropy	Focal Loss
Equal weighting	✅	❌
Focus on hard examples	❌	✅
Handles imbalance	Poorly	Very well (especially with α)

🔸 12. Summary

- **Need**: Handle class imbalance and hard/easy example imbalance in detection tasks.
- **Solution**: Focal loss modifies CE to reduce loss from easy examples.
- **Math**: Introduces $(1 - p_t)^\gamma$ modulating factor and optional α balancing.
- **Impact**: Boosts performance in one-stage dense detectors like RetinaNet.

Would you like a plot showing how focal loss changes shape compared to cross-entropy, or code in PyTorch or TensorFlow?

why alpha is needed. and also explain with examples of different values how focal loss varies against crossentropy

Great questions. Let's dive deep into both:

🔶 Why Is **Alpha** Needed in Focal Loss?

🔸 Problem 1: Imbalanced Classes

In **binary classification** for object detection:

- Class 0 (background): 95%
- Class 1 (object): 5%

Even with focal loss suppressing easy negatives, the **total loss** is still skewed because **negatives outnumber positives** by a huge margin.

🔸 Solution: Weight the positive class higher using α (alpha)

Focal Loss (with alpha):

$$FL(p_t) = -\alpha_t \cdot (1 - p_t)^{\gamma} \cdot \log(p_t)$$

Where:

- $\alpha_t = \alpha$ if class is positive (y = 1)
- $\alpha_t = 1 - \alpha$ if class is negative (y = 0)

🔸 Why?

It ensures the **positive class is not drowned out** by the volume of easy negatives.

🔶 Visual and Numeric Intuition: How Focal Loss Varies with γ and α

Let's compare **Cross-Entropy** and **Focal Loss** under different configurations.

🔷 Scenario: Ground truth $y = 1$

Let’s fix:

- Ground truth: $y = 1$
- p: model’s prediction for class 1

Then:

- $p_t = p$
- Cross Entropy: $-\log(p)$
- Focal Loss: $-\alpha(1-p)^\gamma \log(p)$

We’ll try different values of p, and compare:

p	**CE Loss**	**FL (γ=2, α=1.0)**	**FL (γ=2, α=0.25)**	**FL (γ=1, α=0.25)**
0.1	2.3026	1.865	0.466	0.863
0.3	1.2040	0.588	0.147	0.264
0.5	0.6931	0.173	0.043	0.087
0.7	0.3567	0.051	0.013	0.027
0.9	0.1053	0.001	0.00026	0.00087

🔎 Interpretation:

1. **Cross-Entropy** (no α, no γ):
    - Doesn’t care how confident the prediction is — it always punishes based on how wrong it is.

   - Even if p = 0.9 (a good prediction), still has ~0.1 loss.

2. **Focal Loss with γ = 2, α = 1.0**:
   - Focuses more on **low p** (hard examples).
   - **Suppresses high-confidence (easy) examples**.
3. **Focal Loss with α = 0.25**:
   - Same as above, but down-weights the class itself.
   - Used when **positives are rare**, to **reduce dominance of negative class**.
4. **Lower γ (e.g., γ=1)**:
   - Less aggressive focusing; still emphasizes hard examples but not as sharply.

🔶 Visual Intuition (you can plot this):

- Plot curves of CE and FL over $p \in (0, 1)$
- For CE: curve is steep at low p, flat near $p = 1$
- For FL:
  - With γ = 2, the curve **peaks** when p is small and **drops rapidly** as p increases
  - With α < 1, the entire FL curve is **scaled down** (less importance)

🔶 Summary Table

Term	Role in Focal Loss
γ	**Focuses on hard examples** by down-weighting high-confidence (easy) predictions
α	**Balances positive vs. negative class** contributions in imbalanced datasets
When α = 1	All classes are treated equally
When α < 1	Positive class is **given more importance** (used for rare classes)

✅ Example: When Is α Needed?

Case	Solution
Balanced binary classes	α can be omitted
Imbalanced (90% background)	α = 0.25 for positive class
Multi-class imbalance	Use **per-class α**

Would you like a code snippet to **visualize these curves** or compare **loss curves interactively** using matplotlib?

explain mean average precision in extreme detail rigth down from basic to advance including all math formula and calculation.

Absolutely. Let's **build a complete, step-by-step explanation of Mean Average Precision (mAP)** used in **object detection**, from fundamentals to full mathematical depth.

🔶 PART 1: FOUNDATION

🔹 What is mAP?

- **Mean Average Precision (mAP)** is a single number that summarizes how well your object detection model performs across all classes.
- It is the **mean** of the **Average Precision (AP)** for each class.
- Each AP is computed from the **Precision-Recall curve** for that class.

🔶 PART 2: CORE CONCEPTS

🔹 1. Precision and Recall

Let’s say the model makes N predictions. For each prediction, we compare with ground truth:

- **True Positive (TP)**: Correct class and IoU $\geqslant$ threshold (e.g., 0.5)
- **False Positive (FP)**: Wrong class or IoU < threshold
- **False Negative (FN):** Missed ground truth

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

🔹 2. IoU: Intersection over Union

To decide if a prediction is correct:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \in [0, 1]$$

- If IoU $\geqslant$ threshold (say 0.5), the prediction is a **match** (TP)

🔹 3. Precision-Recall Curve

For a class:

- Sort all detections **by confidence score (descending)**
- For each threshold (e.g., top-k predictions), calculate **Precision** and **Recall**
- Plot **Precision vs Recall**

🔶 PART 3: CALCULATING AVERAGE PRECISION (AP)

There are **two major versions** of AP calculation:

✅ Method A: VOC-style AP (11-point Interpolated)

From Pascal VOC 2007:

1. For 11 recall levels $R \in \{0.0, 0.1, ..., 1.0\}$
2. Find the maximum precision at **each recall $\geqslant$ R**:

$$P_{\text{interp}}(R) = \max_{\tilde{R} \geqslant R} P(\tilde{R})$$

3. Then:

$$AP = \frac{1}{11} \sum_{R \in \{0.0,...,1.0\}} P_{\text{interp}}(R)$$

✅ Method B: COCO-style AP (Integral over PR curve)

Used in modern benchmarks (e.g., COCO):

1. Compute precision-recall pairs at **every prediction**
2. Interpolate precision:

$$P_{\text{interp}}(r) = \max_{\tilde{r} \geqslant r} P(\tilde{r})$$

3. Approximate the area under the curve using discrete summation:

$$AP = \sum_{n=1}^{N} (r_n - r_{n-1}) \cdot P_{\text{interp}}(r_n)$$

> This is essentially **trapezoidal integration** of the PR curve.

🔶 PART 4: FROM AP TO mAP

Once you have **AP for each class**, compute:

$$\boxed{mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c}$$

Where:

- C = total number of classes
- AP_c = Average Precision for class c

🔶 PART 5: AP AT DIFFERENT IoU Thresholds

COCO Evaluation computes AP at **10 different IoU thresholds**:

$$\text{IoU thresholds} = \{0.50,\ 0.55,\ 0.60,\ \ldots,\ 0.95\}$$

Final score:

$$AP_{COCO} = \frac{1}{10} \sum_{t=0.5}^{0.95} AP_t$$

This captures **both precision and localization accuracy** (higher IoU is stricter).

🔶 PART 6: COCO Metrics Overview

Metric	Meaning
AP@[0.5:0.95] (mAP)	Mean over 10 IoU thresholds (main metric)
AP@0.5	IoU = 0.5 (lenient match)
AP@0.75	IoU = 0.75 (strict match)
AP_small	Objects < 32^2 pixels
AP_medium	Objects between 32^2 and 96^2
AP_large	Objects > 96^2 pixels

🔶 PART 7: Step-by-Step Example (Single Class)

Ground Truth Boxes:

- GT = 3 boxes

Model Predictions (sorted by confidence):

Conf	IoU w/ GT	TP/FP
0.95	0.8	TP
0.90	0.6	TP
0.85	0.3	FP
0.75	0.55	TP
0.60	0.2	FP

Compute cumulative TP/FP → Precision & Recall:

Rank	TP	FP	Precision	Recall
1	1	0	1.00	0.33
2	2	0	1.00	0.66
3	2	1	0.67	0.66
4	3	1	0.75	1.00
5	3	2	0.60	1.00

Now compute **area under PR curve (AP)** using interpolation or trapezoid approximation.

🔶 PART 8: Summary Formulae

Component	Formula
IoU	$\frac{\text{Intersection}}{\text{Union}}$
Precision	$\frac{TP}{TP+FP}$

Component	Formula
Recall	$\frac{TP}{TP+FN}$
Interpolated Precision	$\max_{r' \geq r} P(r')$
AP (COCO)	$\int_0^1 P_{\text{interp}}(r)\,dr \approx \sum \Delta r \cdot P(r)$
mAP	$\frac{1}{C} \sum_{c=1}^{C} AP_c$
mAP@[.5:.95]	$\frac{1}{10C} \sum_{t=0.5}^{0.95} \sum_{c=1}^{C} AP_{c,t}$

Would you like me to:

- Show you Python code to compute AP from raw predictions?
- Plot a precision-recall curve and shaded AP area for better visual intuition?

i still did not understand anything. what is map and how is it calulcate and the math. you did not explain any of them

You're right — let me now **start from absolute scratch**, and walk you step-by-step through:

> ✅ What is **mAP**
> ✅ What is **AP**
> ✅ What is **Precision**, **Recall**, and how **IoU** is used
> ✅ **Mathematical formulas and step-by-step calculation**

I'll build from **1 prediction** to **all predictions**, and finally **mAP**.

🧠 1. What Is the Goal in Object Detection?

For each image:

- The model **predicts bounding boxes and class labels**, with **confidence scores** (e.g., “dog, 0.93”).

- We want to **measure how good** these predictions are.

That’s where **mAP (mean Average Precision)** comes in.

🔶 2. What is **IoU** (Intersection over Union)?

Before any metric, we must know:

> Is a predicted box close enough to a ground truth box?

We use:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

If IoU $\geqslant 0.5$ (or 0.75, etc.), then the predicted box is **"correct"** (i.e., a **True Positive**).

🔶 3. Definitions of TP, FP, FN

Let’s say for a class like "car":

Term	Meaning
TP	Detected a real car correctly (IoU $\geqslant$ 0.5 and correct class)
FP	Predicted a car, but either no car there or wrong class or low IoU
FN	There was a car, but you didn’t predict it

🔶 4. What is **Precision and Recall**?

Term	Formula
Precision	$\frac{TP}{TP+FP}$ — How many predicted cars were correct
Recall	$\frac{TP}{TP+FN}$ — How many real cars you found

🔶 5. How to Build a **Precision-Recall Curve**

For a class like **"car"**:

Step-by-step:

1. Predict N boxes, each with confidence score (say: 0.95, 0.9, 0.7, ...)
2. Sort predictions by score **descending**.
3. Go down the list, for each box:
    - Check: Is it a TP or FP (using IoU with GT)?
    - Keep track: `#TP`, `#FP`
    - Calculate precision and recall

You now have a set of points: (recall, precision)

> Plot this to get the **Precision–Recall Curve**

🔶 6. What Is **Average Precision (AP)**?

It's the **area under the Precision–Recall Curve** (AUC).

Approximate as:

$$AP = \sum_{i=1}^{N} (R_i - R_{i-1}) \cdot P_i$$

Where:

- R_i = recall at step i
- P_i = precision at step i

It’s like: sum of **rectangle areas** under the curve.

🧮 Mini Example:

Suppose you have 3 ground truth cars.

Predictions:

Confidence	Correct (IoU ⩾ 0.5)?
0.95	✅ TP1
0.9	✅ TP2
0.85	❌ FP1
0.8	✅ TP3

Let's compute:

Step	TP	FP	Precision	Recall
1	1	0	1.00	1/3 = 0.33
2	2	0	1.00	2/3 = 0.66
3	2	1	2/3	2/3 = 0.66
4	3	1	3/4 = 0.75	1.00

Now compute **AP** as:

$$AP = (0.66 - 0.33) \cdot 1.00 + (1.00 - 0.66) \cdot 0.75 = 0.33 + 0.255 = 0.585$$

🔶 7. What Is mean Average Precision (mAP)?

If your dataset has **C classes**, then:

$$\boxed{mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c}$$

That is: average the AP over all classes.

🔶 8. Variants of mAP

Metric	Meaning
mAP@0.5	IoU threshold = 0.5
mAP@0.75	IoU threshold = 0.75 (stricter)
mAP@[.5:.95]	COCO-style average of 10 IoU thresholds

✅ Final Summary:

Step	Description
1	Predict bounding boxes with confidence
2	Match with ground truth using IoU ⩾ 0.5
3	Sort predictions by confidence
4	Label each as TP or FP
5	Calculate Precision and Recall
6	Build PR curve
7	Compute AP = area under PR curve
8	Compute mAP = average AP over all classes

Would you like a **code implementation in Python** that walks through this with real values and plots?

what is 1-point Interpolated or 10 different IoU thresholds ?

Great question! Let's clarify **two different things** you mentioned:

🔷 A. What is **11-point Interpolated AP** (Pascal VOC)

> This is a way to **approximate Average Precision (AP)** from the Precision–Recall curve using **11 fixed recall levels**:

$$\{0.0, 0.1, 0.2, \dots, 1.0\}$$

📌 Definition (11-point interpolated AP):

For each recall level $r \in \{0.0, 0.1, \dots, 1.0\}$:

$$P_{\text{interp}}(r) = \max_{\tilde{r} \geqslant r} P(\tilde{r})$$

That is: For each recall threshold, take the **maximum precision at or above that recall**.

Then, average over all 11 values:

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \dots, 1.0\}} P_{\text{interp}}(r)$$

> ✅ This was used in **Pascal VOC 2007**.

🔸 Why interpolation?

Because the Precision–Recall curve can be **jagged**, and interpolation **smooths** it for a fairer estimate.

🧮 Example:

Suppose your PR curve gives you precision at some recall values:

Recall	Precision
0.2	0.8
0.4	0.7

Recall	Precision
0.7	0.6

We interpolate these to compute:

- $P(0.0) = \max_{r \geqslant 0.0} P(r) = 0.8$
- $P(0.1) = 0.8$
- …
- $P(0.4) = 0.7$
- …
- $P(0.7) = 0.6$
- $P(0.8) = 0$
- …
- Average these 11 values = **AP**

🔷 B. What is mAP@[.5:.95] with 10 IoU thresholds (COCO metric)

> Instead of computing AP at **just IoU ⩾ 0.5**, modern benchmarks compute it at **multiple IoU thresholds**.

📌 COCO uses:

$$\text{IoU thresholds} = \{0.50,\ 0.55,\ 0.60,\ \ldots,\ 0.95\} \quad \text{(step size: 0.05)}$$

📌 Final mAP is:

$$\boxed{mAP@[0.5:0.95] = \frac{1}{10} \sum_{\text{IoU}=0.5}^{0.95} AP_{\text{IoU}}}$$

> This forces the model to be good at **both finding objects** and **localizing them precisely**.

🔸 mAP@0.5 vs mAP@0.75

- **mAP@0.5**: Match if box has IoU ⩾ 0.5 — **lenient**
- **mAP@0.75**: Match only if IoU ⩾ 0.75 — **strict**

✅ Final Summary

Metric	Description
11-point Interpolated AP	Smooth PR curve by max precision at ⩾ recall. Used in VOC 2007
mAP@[.5:.95]	Mean of APs computed at 10 IoU thresholds. Used in COCO
AP@0.5	AP at IoU ⩾ 0.5 only
AP@0.75	AP at IoU ⩾ 0.75 only

Would you like plots or code to visualize **interpolated PR curves** and IoU thresholds?