# YOLO Metrics – Complete Explanation with Numerical Examples

## Introduction

Every time YOLO trains or makes a prediction, it produces numbers that tell you **how well the model is doing**. These numbers are called **metrics**. Without understanding them, training is a black box – you won't know whether to stop early, collect more data, or tune a threshold.

This guide explains every metric that appears in the YOLO training table and validation report:

1. **IoU** – Intersection over Union (the foundation of everything else)
2. **Confidence Score** – How sure the model is about a detection
3. **box_loss** – How far predicted boxes are from ground truth
4. **cls_loss** – How wrong class predictions are
5. **dfl_loss** – How precisely boxes are localized
6. **Precision** – Of all detections made, how many were correct?
7. **Recall** – Of all real objects, how many were found?
8. **F1 Score** – Harmonic balance between Precision and Recall
9. **mAP50** – Mean Average Precision at IoU threshold 0.50
10. **mAP50-95** – Stricter mAP averaged across multiple IoU thresholds

---

## Metric 1 – IoU (Intersection over Union)

### What It Is

**IoU** measures how much a predicted bounding box overlaps with the ground truth (correct) bounding box. It is the **single most foundational metric in object detection** – every other metric in YOLO depends on it.

**Real-world analogy**: Imagine you ask someone to draw a rectangle around a cat in a photo. You already have the "correct" rectangle drawn by an expert. IoU answers: *how much do the two rectangles overlap, as a fraction of their total combined area?* If the two rectangles are identical, IoU = 1.0. If they don't touch at all, IoU = 0.0.

### Mathematical Foundation

**Intuition**: Take two rectangles – the predicted box and the ground truth box. The overlap area (intersection) divided by the total area covered by either box (union) gives a number between 0 and 1.

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

$$\text{Area of Union} = \text{Area}_{pred} + \text{Area}_{gt} - \text{Area of Intersection}$$

**Symbol definitions:**

- $\text{Area}_{pred}$ = area of the predicted bounding box
- $\text{Area}_{gt}$ = area of the ground truth bounding box
- Area of Intersection = area where both boxes overlap
- Area of Union = total area covered by either box (avoiding double-counting the overlap)

### Numerical Example

**Problem**: The model predicts a box around a dog. The ground truth and predicted boxes are:
Ground truth box: top-left $(10, 20)$, bottom-right $(60, 80)$
Predicted box: top-left $(20, 30)$, bottom-right $(70, 90)$

**Step 1 – Find the intersection rectangle**

The intersection is the region where both boxes overlap. We find it by taking the maximum of the left edges and minimum of the right edges:

$$x_1^{inter} = \max(10, 20) = 20 \qquad x_2^{inter} = \min(60, 70) = 60$$

$$y_1^{inter} = \max(20, 30) = 30 \qquad y_2^{inter} = \min(80, 90) = 80$$

**Step 2 – Compute individual areas**

$$\text{Width}_{pred} = 70 - 20 = 50 \qquad \text{Height}_{pred} = 90 - 30 = 60$$

$$\text{Area}_{pred} = 50 \times 60 = 3{,}000 \text{ px}^2$$

$$\text{Width}_{gt} = 60 - 10 = 50 \qquad \text{Height}_{gt} = 80 - 20 = 60$$

$$\text{Area}_{gt} = 50 \times 60 = 3{,}000 \text{ px}^2$$

**Step 3 – Compute intersection area**

$$\text{Width}_{inter} = 60 - 20 = 40 \qquad \text{Height}_{inter} = 80 - 30 = 50$$

$$\text{Area}_{inter} = 40 \times 50 = 2{,}000 \text{ px}^2$$

**Step 4 – Compute union and IoU**

$$\text{Area}_{union} = 3{,}000 + 3{,}000 - 2{,}000 = 4{,}000 \text{ px}^2$$

$$\text{IoU} = \frac{2{,}000}{4{,}000} = 0.50$$

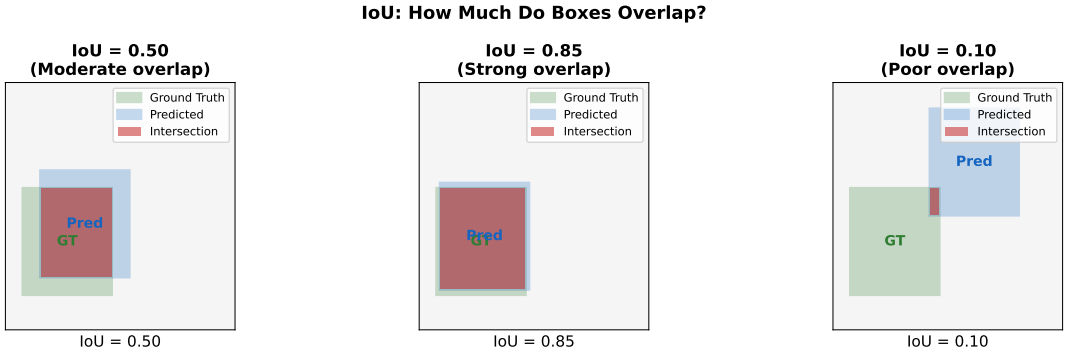
**Interpretation**: IoU = 0.50 means the predicted box overlaps exactly half the combined area. This is exactly the threshold used for **mAP50** – this prediction would be counted as correct at the mAP50 threshold but wrong at stricter thresholds like 0.75.

### IoU Thresholds in Practice

| IoU Value | Quality | Counted as Correct? |
|---|---|---|
| 1.00 | Perfect overlap | ✓ Yes (all thresholds) |
| 0.75 | Strong overlap | ✓ Yes (IoU $\geq$ 0.75) |
| 0.50 | Moderate overlap | ✓ Yes (mAP50 threshold) |
| 0.30 | Rough overlap | × No (below standard) |

| IoU Value | Quality | Counted as Correct? |
|---|---|---|
| 0.00 | No overlap | × No |

### Visualization




---

## Metric 2 – Confidence Score

### What It Is

The **confidence score** is a number between 0 and 1 that represents how certain YOLO is that a detection is a real object of the predicted class. It combines two things:

$$\text{Confidence Score} = P(\text{object exists}) \times P(\text{correct class} \mid \text{object})$$

**Real-world analogy**: Imagine a security guard at a concert. Their confidence score combines two questions: (1) *Is there actually someone at this door?* and (2) *Is that person holding a valid ticket for this concert?* Only if both answers are "yes" with high certainty does the guard let them in.

### Symbol Definitions

- $P(\text{object exists})$ = probability that any object (not a specific class) is present in that grid cell
- $P(\text{correct class} \mid \text{object})$ = conditional probability that the object belongs to the predicted class, given that an object exists
- Confidence Score = the final per-class detection score shown in YOLO output

### Numerical Example

The model examines a grid cell containing what appears to be a dog.

$$P(\text{object exists}) = 0.90$$

$$P(\text{dog} \mid \text{object}) = 0.85$$

$$\text{Confidence Score}_{dog} = 0.90 \times 0.85 = 0.765$$

For the same cell, for the "cat" class:

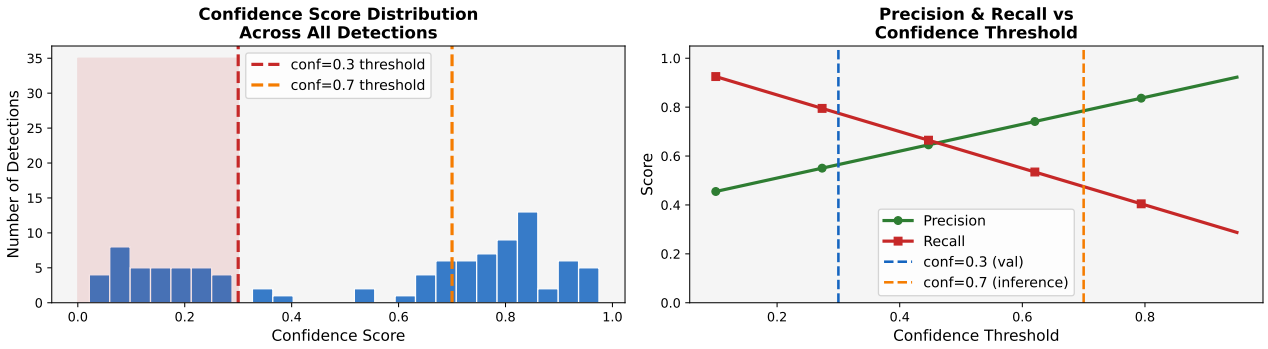
$$P(\text{cat} \mid \text{object}) = 0.10$$

$$\text{Confidence Score}_{cat} = 0.90 \times 0.10 = 0.090$$

With `conf=0.3` (our threshold), only the dog detection ($0.765 \geq 0.30$) is kept. The cat detection ($0.090 < 0.30$) is discarded.

**Interpretation**: Even though the model is 90% sure something exists, it only reports the dog class because that's the only class scoring above the threshold. This is why lowering `conf` reveals more detections (including weaker ones), and raising it produces fewer but more reliable detections.

### Visualization




---

## Metric 3 – box_loss (Bounding Box Regression Loss)

### What It Is

**box_loss** measures how far the predicted bounding box coordinates are from the ground truth box. It is computed during training for every detected object and averaged across the batch. A decreasing `box_loss` means the model is getting better at predicting *where* objects are.

**Real-world analogy**: Think of it like archery. The bullseye is the ground truth box. Each arrow (predicted box) lands somewhere on the target. `box_loss` is the average distance all your arrows are from the bullseye – it should shrink as you practice (train).

### Mathematical Foundation

YOLOv8 uses **CIoU loss** (Complete IoU Loss) for bounding box regression. It is an improved version of plain IoU loss that also penalizes:

- Distance between box centers
- Aspect ratio differences

$$\mathcal{L}_{box} = 1 - \text{IoU} + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha \cdot v$$

**Symbol definitions:**

- IoU = standard Intersection over Union between predicted and GT boxes

- $\rho^2(b, b^{gt})$ = squared Euclidean distance between the centers of the predicted box $b$ and ground truth box $b^{gt}$
- $c^2$ = squared length of the diagonal of the smallest enclosing box (normalizes the center distance)
- $v$ = aspect ratio consistency term: $v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2$
- $\alpha$ = weighting factor for the aspect ratio term: $\alpha = \frac{v}{(1-\text{IoU})+v}$

**Why CIoU over plain IoU?** Plain IoU loss becomes zero when boxes don't overlap at all – giving zero gradient and no learning signal. CIoU still provides useful gradient through the center distance term, even for non-overlapping boxes.

### Numerical Example

Two boxes during training:

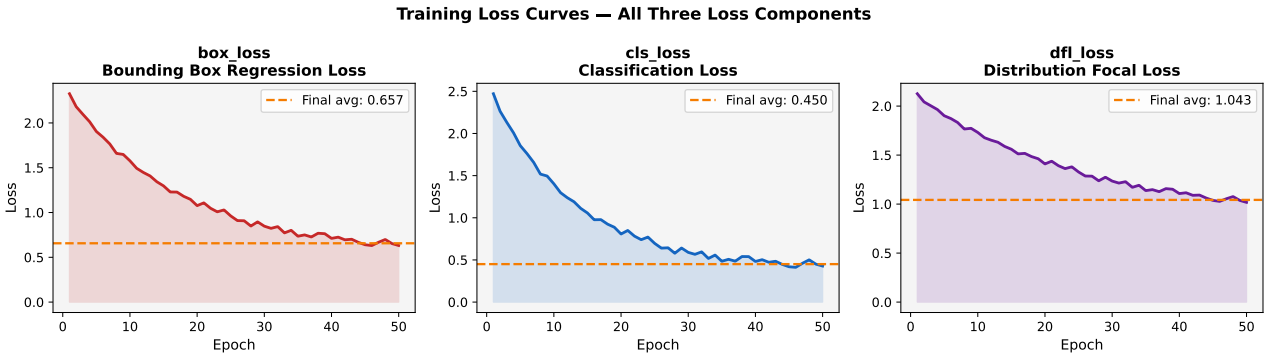
$$\text{Predicted center: } (35, 50) \qquad \text{GT center: } (40, 55)$$

$$\text{Predicted size: } w = 48, h = 60 \qquad \text{GT size: } w = 50, h = 60$$

| Step | Operation | Calculation | Result |
|---|---|---|---|
| 1 | Compute IoU | (from box areas) | 0.82 |
| 2 | Center distance $\rho^2$ | $(35-40)^2 + (50-55)^2$ | 50 |
| 3 | Enclosing diagonal $c^2$ | $(80)^2 + (90)^2$ | 14,500 |
| 4 | Center penalty | 50/14,500 | 0.00345 |
| 5 | Aspect ratio $v$ | $\frac{4}{\pi^2}(\arctan(50/60) - \arctan(48/60))^2$ | $\approx 0.0008$ |
| 6 | CIoU Loss | $1 - 0.82 + 0.00345 + 0.0008$ | $\approx 0.184$ |

**Interpretation**: A `box_loss` of 0.184 means the boxes are reasonably close but not perfect. Early in training you might see values like 1.8. By epoch 50, a well-trained model should reach 0.4-0.9 depending on dataset complexity.

### Training Curve Interpretation




---

## Metric 4 – cls_loss (Classification Loss)

### What It Is

**cls_loss** measures how wrong the model's class predictions are. Even if YOLO perfectly locates an object (low `box_loss`), it might still call a cat a dog – `cls_loss` captures that mistake.

**Real-world analogy**: Imagine a game show where contestants must identify animals. `box_loss` is how accurately they point to the animal on screen. `cls_loss` is how often they name the wrong animal. Both need to be low to win.

### Mathematical Foundation

YOLOv8 uses **Binary Cross-Entropy (BCE)** applied per class (not softmax), which allows multi-label detection (an object can belong to multiple classes):

$$\mathcal{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\left[y_{ic}\log(\hat{p}_{ic}) + (1-y_{ic})\log(1-\hat{p}_{ic})\right]$$

**Symbol definitions:**

- $N$ = number of detected objects in this batch
- $C$ = number of classes (e.g., 3 for cat/dog/bird)
- $y_{ic}$ = ground truth label: 1 if object $i$ belongs to class $c$, else 0
- $\hat{p}_{ic}$ = model's predicted probability that object $i$ belongs to class $c$
- log = natural logarithm

**Why use BCE per class instead of softmax?** Softmax forces class probabilities to sum to 1, which assumes an object belongs to exactly one class. BCE treats each class independently, which is more flexible and works better in practice for YOLO.

### Numerical Example

One detected object (a cat) with 3 classes: cat=0, dog=1, bird=2.

Ground truth: $y = [1, 0, 0]$ (it's a cat)

Model predictions: $\hat{p} = [0.85, 0.10, 0.05]$

**BCE for each class:**

| Class | $y_{ic}$ | $\hat{p}_{ic}$ | Formula | Loss |
|---|---|---|---|---|
| cat | 1 | 0.85 | $-[1\cdot\log(0.85) + 0\cdot\log(0.15)]$ | 0.163 |
| dog | 0 | 0.10 | $-[0\cdot\log(0.10) + 1\cdot\log(0.90)]$ | 0.105 |
| bird | 0 | 0.05 | $-[0\cdot\log(0.05) + 1\cdot\log(0.95)]$ | 0.051 |

$$\mathcal{L}_{cls} = \frac{0.163 + 0.105 + 0.051}{3} = \frac{0.319}{3} = 0.106$$

**Now if the model is confused and predicts $\hat{p} = [0.40, 0.40, 0.20]$:**

| Class | $y_{ic}$ | $\hat{p}_{ic}$ | Loss |
|---|---|---|---|
| cat | 1 | 0.40 | $-\log(0.40) = 0.916$ |
| dog | 0 | 0.40 | $-\log(0.60) = 0.511$ |
| bird | 0 | 0.20 | $-\log(0.80) = 0.223$ |

$$\mathcal{L}_{cls} = \frac{0.916 + 0.511 + 0.223}{3} = 0.550$$

**Interpretation**: A confused prediction gives 5× higher `cls_loss`. Early in training (epoch 1-5), `cls_loss` values of 2.0-3.0 are normal. By epoch 50, a well-trained model should reach 0.3-0.8. If `cls_loss` remains high while `box_loss` drops, the model is finding objects but misidentifying them – a signal to check label quality or add more diverse training samples per class.

---

## Metric 5 – dfl_loss (Distribution Focal Loss)

### What It Is

**dfl_loss** measures the precision of bounding box edge localization. While `box_loss` (CIoU) ensures the predicted box roughly overlaps the ground truth, `dfl_loss` ensures the **exact edges** of the box are in the right place.

**Real-world analogy**: Imagine wrapping a gift. `box_loss` ensures you use roughly the right amount of wrapping paper. `dfl_loss` ensures the edges are folded precisely – no wrinkles or gaps at the corners.

### Mathematical Foundation

YOLOv8 does not predict box edge coordinates directly. Instead, it models each edge position as a **probability distribution** over a set of discrete bins (positions). The DFL loss measures how concentrated this distribution is around the correct edge position.

$$\mathcal{L}_{dfl} = -\left[(y_{i+1} - y)\log(p_i) + (y - y_i)\log(p_{i+1})\right]$$

**Symbol definitions:**

- $y$ = true edge position (e.g., true left edge = 23.7 pixels)
- $y_i, y_{i+1}$ = the two integer bin positions surrounding $y$ (23 and 24)
- $p_i, p_{i+1}$ = predicted probabilities for bins $y_i$ and $y_{i+1}$
- The loss encourages the two surrounding bins to have the highest probability

### Numerical Example

True left edge of a bounding box: $y = 23.7$ pixels. The model distributes probability across bins 20-27:

| Bin | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|
| $p$ | 0.01 | 0.02 | 0.05 | 0.50 | 0.30 | 0.07 | 0.03 | 0.02 |

The two surrounding bins are 23 and 24, so $y_i = 23$, $y_{i+1} = 24$, $p_i = 0.50$, $p_{i+1} = 0.30$.

$$\mathcal{L}_{dfl} = -\left[(24 - 23.7)\log(0.50) + (23.7 - 23)\log(0.30)\right]$$

$$= -\left[0.3 \times (-0.693) + 0.7 \times (-1.204)\right]$$

$$= -[-0.208 - 0.843] = 1.051$$

**If the distribution is sharper** (model more certain): $p_{23} = 0.70$, $p_{24} = 0.28$:

$$\mathcal{L}_{dfl} = -[0.3 \times \log(0.70) + 0.7 \times \log(0.28)] = -[0.3(-0.357) + 0.7(-1.273)] = 0.998$$

**Interpretation**: Lower `dfl_loss` means the model produces crisper, more precisely-edged bounding boxes. This directly improves mAP50-95 (which requires accurate boxes at strict IoU thresholds) more than mAP50.

---

## Metric 6 – Precision

### What It Is

**Precision** answers: *Of all the detections the model made, what fraction were actually correct?*
**Real-world analogy**: A doctor ordering medical tests. Precision asks: *Of all the patients the doctor said "you have disease X," what fraction actually had it?* Low precision = many false alarms (healthy people told they're sick).

### Mathematical Foundation

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

**Symbol definitions:**

- $TP$ (True Positive) = model detected an object, and it was correct (right class, IoU $\geq$ threshold)
- $FP$ (False Positive) = model detected an object, but it was wrong (no real object there, wrong class, or IoU too low)
- $TP + FP$ = total number of detections made by the model

### Numerical Example

The model processes 10 images and makes 15 total detections.

| Detection | Ground Truth | IoU | Confidence | Verdict |
|---|---|---|---|---|
| cat box #1 | cat ✓ | 0.82 | 0.91 | TP |
| cat box #2 | cat ✓ | 0.73 | 0.85 | TP |
| dog box #1 | dog ✓ | 0.91 | 0.88 | TP |
| dog box #2 | dog ✓ | 0.55 | 0.72 | TP |
| bird box #1 | bird ✓ | 0.61 | 0.67 | TP |
| cat box #3 | nothing × | 0.00 | 0.35 | FP (hallucinated) |
| dog box #3 | cat × | 0.71 | 0.42 | FP (wrong class) |

| Detection | Ground Truth | IoU | Confidence | Verdict |
|---|---|---|---|---|
| bird box #2 | bird × | 0.38 | 0.31 | FP (IoU too low) |
| cat box #4 | cat ✓ | 0.79 | 0.88 | TP |
| dog box #4 | dog ✓ | 0.66 | 0.75 | TP |

$TP = 7$, $FP = 3$, Total detections = 10 (ignoring the other 5 for this example)

$$\text{Precision} = \frac{7}{7+3} = \frac{7}{10} = 0.70$$

**Interpretation**: 70% of the model's detections were correct. 30% were false alarms. In applications like medical imaging or security, you want precision as high as possible – false positives have real costs.

---

## Metric 7 – Recall

### What It Is

**Recall** answers: *Of all the real objects in the images, what fraction did the model find?*

**Real-world analogy**: Same doctor scenario. Recall asks: *Of all the patients who actually had disease X, what fraction did the doctor correctly identify?* Low recall = missed diagnoses (sick people told they're healthy) – often more dangerous than low precision.

### Mathematical Foundation

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

**Symbol definitions:**

- $TP$ = correctly detected objects
- $FN$ (False Negative) = real objects the model completely missed (no detection, or detection discarded by threshold)
- $TP + FN$ = total number of real objects in the dataset

### Numerical Example

Using the same 10 images from before. The ground truth contains **12 real objects** total. The model found 7 (TP) and missed 5 (FN).

$$\text{Recall} = \frac{7}{7+5} = \frac{7}{12} = 0.583$$

**Interpretation**: The model found 58.3% of all real objects. It missed 41.7%. This is low – a well-trained model on a moderate dataset should achieve recall above 0.75.

### The Precision-Recall Tradeoff

Precision and Recall are inversely related as you change the confidence threshold. This is the most important tradeoff in object detection:

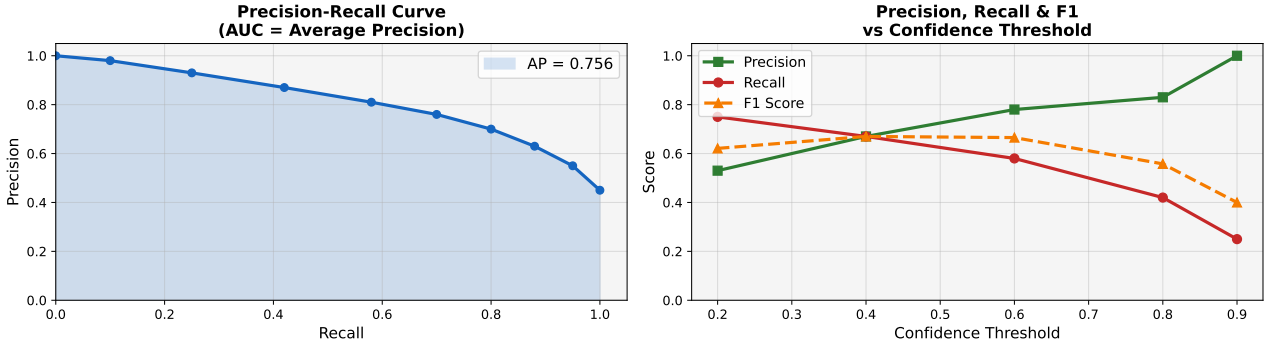
$$\uparrow \text{conf threshold} \Rightarrow \uparrow \text{Precision}, \downarrow \text{Recall}$$

$$\downarrow \text{conf threshold} \Rightarrow \downarrow \text{Precision}, \uparrow \text{Recall}$$

**Why?** Raising the threshold discards more detections. Discarded detections include both FP (good – precision goes up) and TP (bad – recall goes down).

| conf Threshold | TP | FP | FN | Precision | Recall |
|---|---|---|---|---|---|
| 0.20 | 9 | 8 | 3 | 0.53 | 0.75 |
| 0.40 | 8 | 4 | 4 | 0.67 | 0.67 |
| 0.60 | 7 | 2 | 5 | 0.78 | 0.58 |
| 0.80 | 5 | 1 | 7 | 0.83 | 0.42 |
| 0.90 | 3 | 0 | 9 | 1.00 | 0.25 |

### Visualization




---

## Metric 8 – F1 Score

### What It Is

**F1 Score** is the harmonic mean of Precision and Recall. It gives you a single number that balances both metrics. It is especially useful when you need to compare models across different confidence thresholds.

**Real-world analogy**: A search engine result page. Precision = what fraction of results are relevant. Recall = what fraction of all relevant pages on the internet were returned. F1 = one number that captures both – a search engine that returns only one perfect result has high precision but terrible recall. One that returns every page on the internet has perfect recall but terrible precision. F1 punishes both extremes.

**Why harmonic mean?** The harmonic mean is lower than the arithmetic mean when the two values differ – it penalizes imbalance. If Precision=1.0 and Recall=0.0, harmonic mean = 0 (correctly indicating failure), but arithmetic mean = 0.5 (misleadingly positive).

### Mathematical Foundation

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

### Numerical Example

Using the values from our threshold table:

**At conf=0.40:** Precision = 0.67, Recall = 0.67

$$F1 = \frac{2 \times 0.67 \times 0.67}{0.67 + 0.67} = \frac{0.898}{1.34} = 0.670$$

**At conf=0.60:** Precision = 0.78, Recall = 0.58

$$F1 = \frac{2 \times 0.78 \times 0.58}{0.78 + 0.58} = \frac{0.905}{1.36} = 0.666$$

**At conf=0.80:** Precision = 0.83, Recall = 0.42

$$F1 = \frac{2 \times 0.83 \times 0.42}{0.83 + 0.42} = \frac{0.697}{1.25} = 0.558$$

| Threshold | Precision | Recall | F1 Score | Best? |
|---|---|---|---|---|
| 0.20 | 0.53 | 0.75 | 0.621 | |
| 0.40 | 0.67 | 0.67 | **0.670** | ✓ |
| 0.60 | 0.78 | 0.58 | 0.666 | |
| 0.80 | 0.83 | 0.42 | 0.558 | |

**Interpretation**: The F1 score peaks at `conf=0.40` for this example. This is why YOLO generates an **F1 curve** – it shows you the optimal confidence threshold. The YOLO output file `F1_curve.png` saves this curve so you can read off the best threshold for your use case.

---

## Metric 9 – mAP50 (Mean Average Precision at IoU=0.50)

### What It Is

**mAP50** is the primary accuracy metric for object detection. It answers: *On average, across all classes, how well does the model detect objects when we require at least 50% box overlap?*

It is computed in three steps:

1. Compute a **Precision-Recall (PR) curve** for each class
2. Compute the **Average Precision (AP)** = area under the PR curve for each class
3. **Average the AP** values across all classes

**Real-world analogy**: Imagine a model being graded on three subjects (cat, dog, bird). AP is the grade for each subject – it captures performance at all possible confidence thresholds, not just one. mAP50 is the GPA – the average grade across all subjects.

### Step 1 – Building the PR Curve for One Class

Sort all detections by confidence (highest first). For each detection, compute cumulative Precision and Recall:

**Example**: 8 detections for "cat" class (GT has 5 real cats):

| Rank | Confidence | Correct? | Cumulative TP | Cumulative FP | Precision | Recall |
|---|---|---|---|---|---|---|
| 1 | 0.95 | ✓ TP | 1 | 0 | 1.000 | 0.200 |
| 2 | 0.88 | ✓ TP | 2 | 0 | 1.000 | 0.400 |
| 3 | 0.81 | × FP | 2 | 1 | 0.667 | 0.400 |
| 4 | 0.73 | ✓ TP | 3 | 1 | 0.750 | 0.600 |
| 5 | 0.65 | ✓ TP | 4 | 1 | 0.800 | 0.800 |
| 6 | 0.54 | × FP | 4 | 2 | 0.667 | 0.800 |
| 7 | 0.42 | ✓ TP | 5 | 2 | 0.714 | 1.000 |
| 8 | 0.31 | × FP | 5 | 3 | 0.625 | 1.000 |

This table generates the PR curve points: $(0.20, 1.000), (0.40, 1.000), (0.60, 0.750), (0.80, 0.800), (1.00, 0.71$

**Step 2 – Average Precision (AP) = Area Under PR Curve**

AP is computed as the area under the Precision-Recall curve using interpolation:

$$AP = \sum_{k=1}^{n} (R_k - R_{k-1}) \times P_k^{interp}$$

where $P_k^{interp} = \max_{j \geq k} P_j$ (take the maximum precision at each recall level).
Using our table with interpolated precision:

$$AP_{cat} \approx (0.20-0)\times 1.000 + (0.40-0.20)\times 1.000 + (0.60-0.40)\times 0.750 + (0.80-0.60)\times 0.800 + (1.00-0.80)\times 0.$$

$$= 0.200 + 0.200 + 0.150 + 0.160 + 0.143 = 0.853$$

So $AP_{cat} = 0.853$ (85.3% – strong performance on the cat class).

**Step 3 – Mean AP Across All Classes**

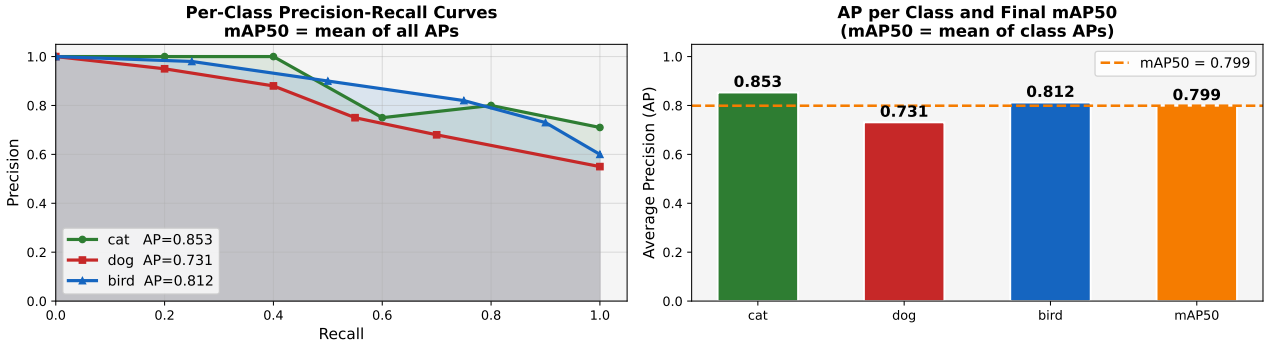
Suppose we have 3 classes with these AP values:

$$AP_{cat} = 0.853 \qquad AP_{dog} = 0.731 \qquad AP_{bird} = 0.812$$

$$\text{mAP50} = \frac{0.853 + 0.731 + 0.812}{3} = \frac{2.396}{3} = 0.799$$

**Interpretation**: mAP50 = 0.799 (79.9%) means the model is performing well overall. The dog class (0.731) is dragging the average down – this class needs more training data or improved annotations. A mAP50 above 0.90 is considered excellent for a custom dataset.

### Visualization




---

## Metric 10 – mAP50-95

### What It Is

**mAP50-95** is a stricter version of mAP50. Instead of requiring IoU $\geq 0.50$, it averages performance across **10 different IoU thresholds**: 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95.

**Real-world analogy**: Instead of just passing a driving test by roughly staying in your lane (IoU=0.50), mAP50-95 also checks how well you stay in the lane at increasingly tight tolerances – slight drift acceptable at first, but nearly perfect lane-keeping required for the highest thresholds.

### Mathematical Foundation

$$\text{mAP50-95} = \frac{1}{10} \sum_{t \in \{0.50, 0.55, \ldots, 0.95\}} \text{mAP}_t$$

where each $\text{mAP}_t$ is computed exactly like mAP50 but using IoU threshold $t$ to determine if a detection counts as TP or FP.

### Numerical Example

Using our 3-class dataset, computing mAP at each threshold:

| IoU Threshold | $AP_{cat}$ | $AP_{dog}$ | $AP_{bird}$ | mAP at this threshold |
|---|---|---|---|---|
| 0.50 | 0.853 | 0.731 | 0.812 | 0.799 |
| 0.55 | 0.840 | 0.712 | 0.795 | 0.782 |
| 0.60 | 0.821 | 0.688 | 0.770 | 0.760 |
| 0.65 | 0.791 | 0.651 | 0.738 | 0.727 |
| 0.70 | 0.748 | 0.601 | 0.693 | 0.681 |
| 0.75 | 0.690 | 0.540 | 0.635 | 0.622 |
| 0.80 | 0.615 | 0.462 | 0.560 | 0.546 |
| 0.85 | 0.520 | 0.371 | 0.462 | 0.451 |
| 0.90 | 0.390 | 0.260 | 0.340 | 0.330 |
| 0.95 | 0.190 | 0.110 | 0.155 | 0.152 |

$$\text{mAP50-95} = \frac{0.799 + 0.782 + 0.760 + 0.727 + 0.681 + 0.622 + 0.546 + 0.451 + 0.330 + 0.152}{10}$$

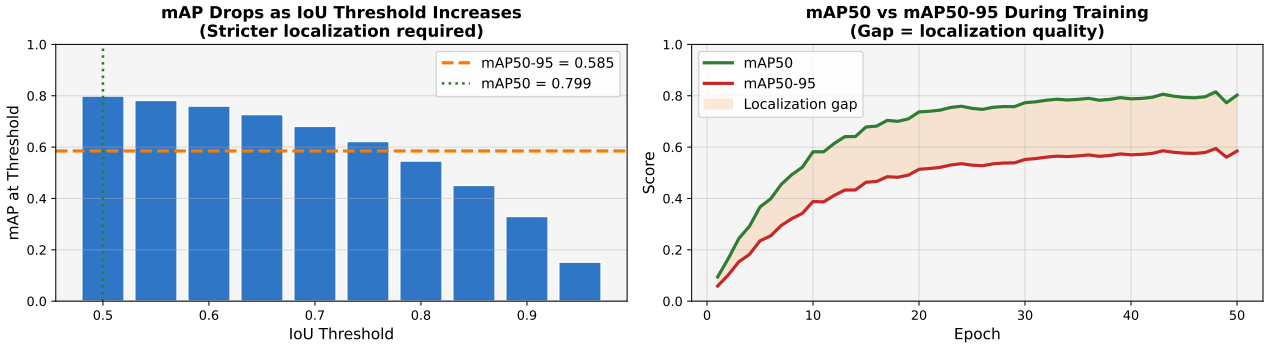
$$= \frac{5.850}{10} = 0.585$$

**Interpretation**: mAP50-95 = 0.585 is the standard COCO-style evaluation metric. Notice it is **significantly lower** than mAP50 (0.799). The gap reveals how much the model struggles to produce tightly localized boxes – the difference (0.214) is the "localization quality gap." A gap larger than 0.30 suggests the model finds objects (good recall) but draws imprecise boxes (poor localization) – addressed by tuning `dfl_loss` or using a larger model variant.

### mAP50 vs mAP50-95: When to Use Each

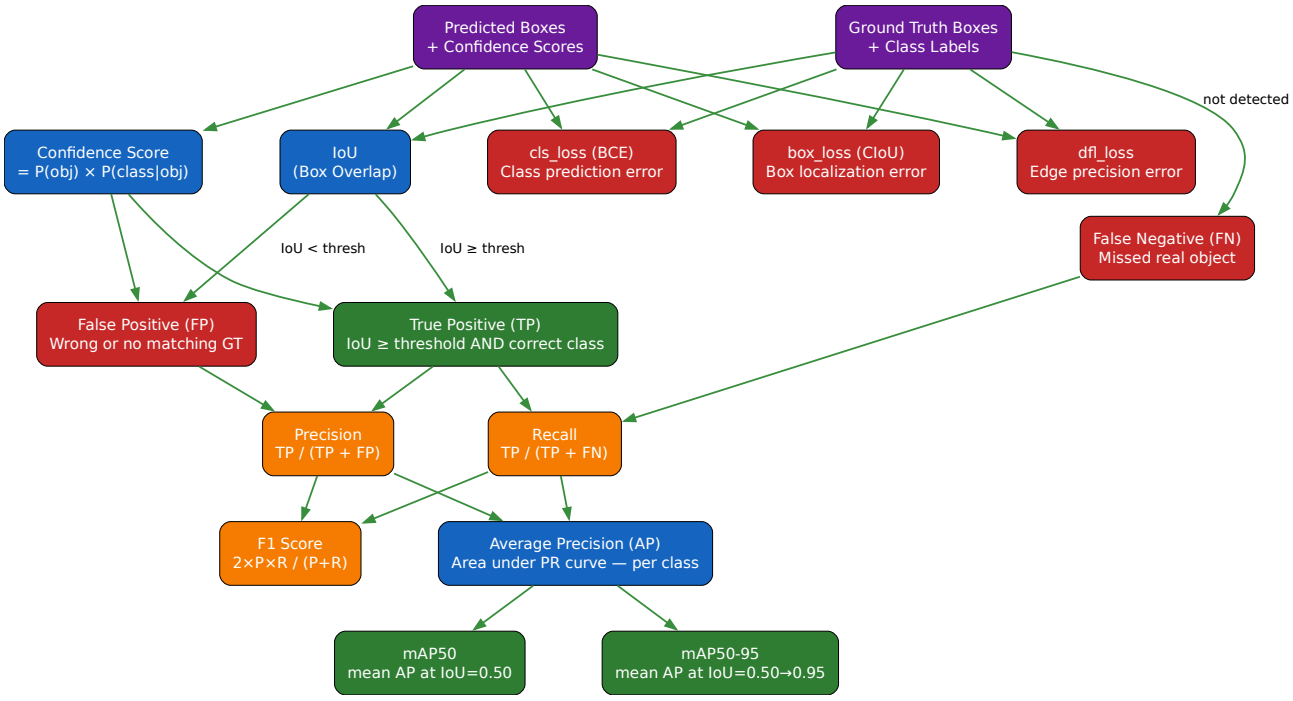
| Metric | Best For | Insight It Gives |
|---|---|---|
| mAP50 | Quick evaluation, small datasets | "Can the model find objects at all?" |
| mAP50-95 | Production systems, COCO benchmarks | "How precisely does the model localize?" |

### Visualization



# Complete Metrics Reference

## How All Metrics Connect




## Quick Reference – All Metrics

| Metric | Formula | Range | Good Value | Appears In |
|---|---|---|---|---|
| IoU | $\frac{\text{Intersection}}{\text{Union}}$ | 0-1 | $\geq 0.50$ | Threshold for TP/FP |
| Confidence | $P(\text{obj}) \times P(\text{class} \mid \text{obj})$ | 0-1 | Tunable | Prediction output |
| box_loss | CIoU loss | 0-$\infty$ | $\downarrow$ to ~0.5 | Training table |
| cls_loss | BCE per class | 0-$\infty$ | $\downarrow$ to ~0.4 | Training table |
| dfl_loss | Distribution focal loss | 0-$\infty$ | $\downarrow$ to ~0.9 | Training table |
| Precision | $TP/(TP+FP)$ | 0-1 | $\geq 0.80$ | Val report |
| Recall | $TP/(TP+FN)$ | 0-1 | $\geq 0.75$ | Val report |
| F1 | $2PR/(P+R)$ | 0-1 | $\geq 0.75$ | F1 curve |
| mAP50 | mean AP at IoU=0.50 | 0-1 | $\geq 0.80$ | Val report |
| mAP50-95 | mean AP at IoU=0.50-0.95 | 0-1 | $\geq 0.55$ | Val report |

## Diagnosing Your Model from Metrics

| Symptom | Most Likely Cause | Action |
|---|---|---|
| High `cls_loss`, low `box_loss` | Model finds objects but misclassifies | Check label consistency, add class-specific data |

| Symptom | Most Likely Cause | Action |
|---|---|---|
| High box_loss, low cls_loss | Model knows class but draws bad boxes | Check annotation quality, increase training epochs |
| Low Precision, high Recall | Too many false positives | Raise conf threshold |
| High Precision, low Recall | Missing many real objects | Lower conf threshold |
| Large mAP50 vs mAP50-95 gap | Boxes are rough, not tight | Use larger model or higher imgsz |
| Both mAP metrics low | General poor performance | More data, more epochs, better annotations |