

# ?

# Overview

**YOLOv1 (You Only Look Once)** is the **first version** of YOLO (2015, by Joseph Redmon). It treats **object detection as a single regression problem** — meaning it predicts **bounding boxes and class probabilities directly from the image** in one pass (instead of using multiple stages like R-CNN).

---

## □ Step-by-Step Architecture Explanation

---

### Step 1: Input Image

- Input image size: **448 × 448 × 3** (Height × Width × RGB channels)
- The image is resized to this fixed size before passing into the network.

So, if you input any image, YOLO first rescales it to 448×448.

---

### Step 2: Feature Extraction (Convolutional Layers)

YOLOv1 uses a **custom CNN architecture** inspired by **GoogLeNet (Inception network)**.

- It consists of **24 Convolutional Layers** followed by **2 Fully Connected (FC) layers**.
  - The convolutional layers act as **feature extractors**, learning edges, textures, shapes, and object patterns.
- 

### □ Breakdown of Key Layer Blocks:

Layer Type	Details	Purpose
Conv + MaxPool	7×7 conv with stride 2	Reduces image size and extracts low-level features
Conv + MaxPool	3×3 conv layers	Extracts deeper features like corners, object edges
Multiple Conv layers	1×1 and 3×3 combos	Alternate between 1×1 (to reduce depth) and 3×3 (to learn spatial info)
MaxPooling	Reduces spatial dimension	Helps make features invariant to small shifts
Final Conv layers	Deep layers	Learn object-level features

- These convolutional layers produce a **feature map** (a compact representation of the image).

---

### Step 3: Flatten + Fully Connected Layers

After the 24 Conv layers:

- The feature map is **flattened** (converted into a long vector)
- Passed through **two fully connected layers**

The final fully connected layer outputs a **vector of size  $7 \times 7 \times 30 = 1470$**

---

### Step 4: Divide the Image into Grid Cells ( $7 \times 7$ )

Now comes the core YOLO idea:

- The image is divided into a  **$7 \times 7$  grid** (total 49 cells).
- Each cell is responsible for **detecting objects whose center** lies inside it.

So the image is conceptually broken like this:

```
+---+---+---+---+---+---+  
|   |   |   |   |   |   |  
+---+---+---+---+---+---+  
|   |   |   |   |   |   |  
+---+---+---+---+---+---+  
... total 7x7 grid cells ...
```

---

### Step 5: Each Grid Cell Predicts (B=2 Boxes + C=20 Classes)

For each of the 49 grid cells, YOLOv1 predicts:

Prediction Type	Count	Meaning
Bounding Boxes	2 per cell	Each box has (x, y, w, h, confidence) = 5 values
Class Probabilities	20	One for each class in Pascal VOC dataset

So per cell →  
**2 boxes × 5 + 20 = 30 values**

Total output =  **$7 \times 7 \times 30 = 1470$  values**

---

### Step 6: Understanding the Predictions

Each grid cell predicts:

1. **Bounding Box Coordinates (x, y, w, h)**

- (x, y): center of the box **relative to the grid cell**
  - (w, h): width and height **relative to the whole image**
2. **Confidence Score**
    - Measures how confident the model is that an object exists in the box.
    - $\text{Confidence} = \Pr(\text{Object}) \times \text{IoU}(\text{pred}, \text{truth})$
  3. **Class Probabilities**
    - Probability distribution over 20 object classes.
- 

## Step 7: Combine Predictions

To get the **final detection scores**:

- Multiply **class probability** and **confidence score** for each box:  
 $P(\text{Class}_i) \times \text{Confidence}$
  - This gives the final score for each object class per bounding box.
- 

## Step 8: Non-Maximum Suppression (NMS)

Multiple boxes might predict the same object.

So:

- YOLO applies **NMS** (Non-Max Suppression)
- It **keeps the box** with the highest score for each object
- **Removes overlapping boxes** with  $\text{IoU} > \text{threshold}$  (e.g., 0.5)

This step ensures **only one bounding box per object** remains.

---

## Step 9: Output

Finally, YOLOv1 outputs:

- Detected objects with **bounding boxes**, **class names**, and **confidence scores**.

Example:

Person: 0.92  
Dog: 0.85  
Bicycle: 0.76

---

## □ Architecture Summary (Layer-Wise)

Layer	Type	Kernel	Stride	Filters	Output size
1	Conv	7×7	2	64	224×224×64
2	MaxPool	2×2	2	-	112×112×64
3–5	Conv	3×3	1	192	112×112×192
6	MaxPool	2×2	2	-	56×56×192
7–10	Conv	3×3	1	512	56×56×512
11	MaxPool	2×2	2	-	28×28×512
12–17	Conv	3×3	1	1024	28×28×1024
18	MaxPool	2×2	2	-	14×14×1024
19–24	Conv	3×3	1	1024	14×14×1024
25–26	Fully Connected	-	-	-	1470

---

## □ Key Characteristics of YOLOv1

Feature	Description
Single forward pass	Detects all objects in one go (very fast)
Grid-based detection	Each grid cell detects one object
Regression-based	Directly predicts bounding boxes + classes
Limitation	Struggles with small objects or multiple objects close together

---

## □ Example Flow (Summary)

1. Input image → 448×448
  2. CNN extracts features → feature map
  3. Divide image → 7×7 grid
  4. Each cell → predicts 2 boxes + 20 class probs
  5. Combine → 7×7×30 output
  6. Apply NMS → remove duplicates
  7. Final detections → bounding boxes + labels
- 

Would you like me to show a **diagram of YOLOv1 architecture (CNN + grid + output)** to visualize how all this connects? It'll make the whole pipeline crystal clear.

You said:

yes

ChatGPT said:

Share

You said:

i thought each grid cell predicts pc bx by bw bh c then what is it

ChatGPT said:

Excellent question, Salman ☐ — you've spotted one of the **most confusing points** in understanding YOLOv1's output structure.

Let's unpack it carefully step by step.

---

## ☐ Step 1 — What each grid cell actually predicts

In YOLOv1, the **image is divided into  $7 \times 7$  grid cells** (49 total).

Each **grid cell** predicts information about:

1. **Objects inside it** (if any)
2. **Bounding boxes**
3. **Class probabilities**

Now let's write exactly what each cell outputs.

---

### ☐ Each grid cell predicts:

$[B \times 5] + C[B \times 5] + C[B \times 5]$

where:

- **B = 2** → number of bounding boxes per cell
- **5** → represents ( $pc, bx, by, bw, bh$ )
- **C = 20** → number of classes (Pascal VOC dataset)

So each grid cell predicts:

$(2 \times 5) + 20 = 30$  values  $(2 \times 5) + 20 = 30$  values  $(2 \times 5) + 20 = 30$  values

That's why total output =  $7 \times 7 \times 30 = 1470$  numbers.

---

## □ Step 2 — Understanding Each Term

### 1. $P_c \rightarrow$ Objectness (Confidence score)

- Tells whether an object exists in the cell.
  - $P_c = \text{Probability}(\text{object exists}) \times \text{IoU}(\text{predicted box, ground truth})$
  - Range: 0 to 1
    - 0 → no object
    - 1 → definitely object
- 

### 2. $bx, by \rightarrow$ Center coordinates of the bounding box

- Represent **offsets relative to the grid cell**.
  - Both values are **normalized between 0 and 1**.
  - For example, if a box center is at the middle of a grid cell  $\rightarrow bx = 0.5, by = 0.5$ .
- 

### 3. $bw, bh \rightarrow$ Width and height of the bounding box

- Represent **size of the predicted box**, relative to the **entire image**.
  - These are also normalized (e.g., 0.2 means 20% of image width/height).
- 

### 4. $C$ (20 class probabilities) $\rightarrow$ One for each class

- Example: person, dog, car, bottle, etc.
- These probabilities are **conditional**:

$$\Pr(\text{Class}_i \mid \text{Object}) \Pr(\text{Class}_i \mid \mid \text{Object}) \Pr(\text{Class}_i \mid \text{Object})$$

meaning: if an object is present, how likely it is to belong to class  $i$ .

---

## □ Step 3 — What happens during detection

When YOLO predicts:

- Each cell gives **2 bounding boxes**, each with its **own ( $P_c, bx, by, bw, bh$ )**
- But **class probabilities ( $C$ )** are shared by the cell (not per box)

So, YOLO combines these two sets of information as follows:

$$\Pr(\text{Class}_i) \times P_c \times \text{IoU} \Pr(\text{Class}_i) \times P_c \times \text{IoU} \Pr(\text{Class}_i) \times P_c \times \text{IoU}$$

This gives a **final score** per box per class — how confident YOLO is that:

"This bounding box contains a particular class."

---

## □ Step 4 — Example Output from One Cell

### Parameter Box 1 Box 2

Pc	0.85	0.30
bx	0.45	0.60
by	0.52	0.47
bw	0.25	0.40
bh	0.18	0.50

**Class probabilities (same for both boxes):**

### Class Prob

Person	0.90
Dog	0.07
Car	0.03

---

Then final class score for Box 1:

$$\text{Score(Person)} = \text{Pc} \times \text{Pr(Person)} = 0.85 \times 0.90 = 0.765$$

If this is the highest among all boxes → YOLO declares “Person” detected at that position.

---

## □ Summary Table

Symbol	Meaning	Predicted by
Pc	Object confidence	Each box
bx, by	Box center offset (within grid cell)	Each box
bw, bh	Box width & height (relative to image)	Each box
C (20 values)	Conditional class probabilities	Shared per cell

So yes — you were **correct** that each grid predicts (Pc, bx, by, bw, bh, c) but since there are **2 boxes** per cell and **20 shared classes**, it becomes:

$$2 \times (\text{Pc}, \text{bx}, \text{by}, \text{bw}, \text{bh}) + 20 \times (\text{Pc}, \text{bx}, \text{by}, \text{bw}, \text{bh}) + 20 \times (\text{Pc}, \text{bx}, \text{by}, \text{bw}, \text{bh}) + 20$$