# 18/09/25

# **Informe técnico — Rediseño de *unpacker* (detección)**

## **1) Resumen**

Se reescribió el *unpacker* de salidas para soportar, de forma vectorizada y con bajo uso de CPU, los tres formatos más comunes en detección. El *unpacker* devuelve siempre filas con layout mínimo:

**[b0, b1, b2, b3, score, class\_id]** *(floats)*

Donde b0..b3 conservan el formato nativo de la IA (p.ej. YOLO: cx,cy,w,h; EfficientDet/SSD: ymin,xmin,ymax,xmax). La conversión final a **[x1,y1,x2,y2, score, class]** la hace el output\_adapter usando tensor\_structure del JSON.

Además, si la IA entrega cajas normalizadas [0..1] y se pasa input\_tensor\_size=(W,H) al callback, el *unpacker* escala a píxeles del tensor de entrada (coherente con undo\_transform del postprocesador).

## **2) Desempaquetadores actualmente soportados**

### **2.1) yolo\_flat**

**Entrada** (típica ONNX/TorchScript):

* Tensor 2D (N, 5+C) con columnas [cx, cy, w, h, obj, p0..pC] (logits o probs de clase).

**Operación (vectorizada):**

* best\_class = argmax(p, axis=1)
* best\_class\_score = p[range(N), best\_class]
* score = obj \* best\_class\_score
* Filtrado por umbral (confidence\_threshold)
* Si parece normalizado y se provee (W,H): escala cx,cy,w,h a píxeles del tensor de entrada.

**Salida:**

* Filas [cx, cy, w, h, score, class\_id] *(floats)*.

**Notas de rendimiento:**

* No se aplica softmax (innecesario para argmax); se evitan bucles por ancla.
* Sólo operaciones numpy vectorizadas y un column\_stack.

**JSON (ejemplo):**

"output": {

"pack\_format": "yolo\_flat",

"tensor\_structure": {

"box\_format": "cxcywh",

"coordinates": {"cx":0, "cy":1, "w":2, "h":3},

"confidence\_index": 4,

"class\_index": 5

}

}

### **2.2) boxes\_scores**

**Entrada** (típica EfficientDet/SSD *pre-NMS*):

* Tupla/lista con **dos tensores** (orden indistinto):  
  + boxes: (1,N,4) o (N,4) en **[ymin, xmin, ymax, xmax]**
  + class\_scores: (1,N,C) o (N,C) (logits o probs)

**Operación (vectorizada):**

* Detecta cuál tensor es boxes por shape[-1] == 4.
* best\_class = argmax(class\_scores, axis=1)
* score = class\_scores[range(N), best\_class]
* Filtro por umbral.
* Si parece normalizado y hay (W,H): escala ymin,xmin,ymax,xmax a píxeles del tensor de entrada.

**Salida:**

* Filas [ymin, xmin, ymax, xmax, score, class\_id] *(floats)*.

**Notas:**

* Sin NMS aquí (es *pre-NMS*). El NMS se aplica en el postprocesador si apply\_nms=true.

**JSON (ejemplo):**

"output": {

"pack\_format": "boxes\_scores",

"tensor\_structure": {

"box\_format": "yxyx",

"coordinates": {"y1":0, "x1":1, "y2":2, "x2":3},

"confidence\_index": 4,

"class\_index": 5

}

}

### **2.3) tflite\_detpost**

**Entrada** (TFLite con DetectionPostProcess, NMS interno):

* Tupla (boxes, scores, classes[, count])
  + boxes: (1,N,4) o (N,4) en **[ymin, xmin, ymax, xmax]** (normalizado)
  + scores: (1,N) o (N,)
  + classes: (1,N) o (N,)
  + count: (1,) opcional

**Operación:**

* Selecciona primeros N=count si existe.
* Si parece normalizado y hay (W,H): escala a píxeles del tensor de entrada.
* No aplica filtrado extra (ya viene filtrado por la op interna). Se puede añadir un umbral ligero en el postproceso.

**Salida:**

* Filas [ymin, xmin, ymax, xmax, score, class\_id] *(floats)*.

**JSON (ejemplo):**

"output": {

"pack\_format": "tflite\_detpost",

"tensor\_structure": {

"box\_format": "yxyx",

"coordinates": {"y1":0, "x1":1, "y2":2, "x2":3},

"confidence\_index": 4,

"class\_index": 5

}

}

### **2.4) raw**

**Entrada/Salida:**

* Retorna el raw\_output sin modificar. Útil para pruebas o modelos que ya emiten el contrato final.

**Advertencia:**

* Debe usarse sólo si el tensor ya viene como [[x1,y1,x2,y2, score, class]] ó equivalente que tu *adapter* pueda entender directamente.

## **3) Flujo de integración**

1. **ModelController**: pasar el tamaño del tensor de entrada al callback (para escalar si viene normalizado):  
     
    unpacked = self.unpack\_fn(raw\_output, input\_tensor\_size=(RuntimeSession)
2. **Output Adapter**: con tensor\_structure convierte de formato nativo (p.ej. cxcywh o yxyx) a **xyxy** y ordena campos a [x1,y1,x2,y2, score, class].
3. **Output Transformer**: aplica filtro por confianza (excepto en tflite\_detpost si no se desea), NMS opcional y undo\_transform para deshacer letterbox/escala al espacio original.

## **4) Pruebas unitarias (pendiente inmediato)**

**Objetivo:** garantizar contrato, formas y semántica mínima.

**Casos por desempaquetador:**

* yolo\_flat:  
  + (N,5+C) con valores controlados → chequear score=obj\*max\_cls, máscara por umbral, tipos float.
* boxes\_scores:  
  + Tuplas en ambos órdenes (boxes, scores) y (scores, boxes).
  + Distintos C y N. Filtro por umbral. Escalado condicional.
* tflite\_detpost:  
  + Con y sin count. Normalizado vs. píxeles. Tipos float. Sin refiltrado (a menos que se decida lo contrario).
* raw:  
  + Identidad.