Computer Vision – Tipos de problemas

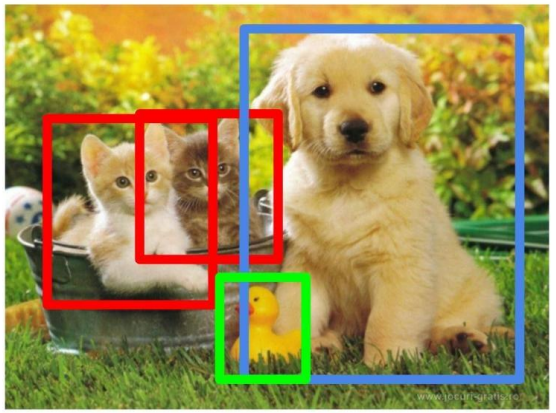
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

Clasificación vs Detección

Image Classification:

- **Input**: Imagen completa
- **Output**: Una clase
- **Ejemplo**: "Esta imagen contiene un perro"

Object Detection:

- **Input**: Imagen completa
- **Output**: Múltiples boxes + clases + confidencias
- **Ejemplo**: "Perro en (100,50,200,150) con 95% confianza"

Componentes Clave

Localización: ¿Dónde está el objeto?

Bounding box: [x1, y1, x2, y2]

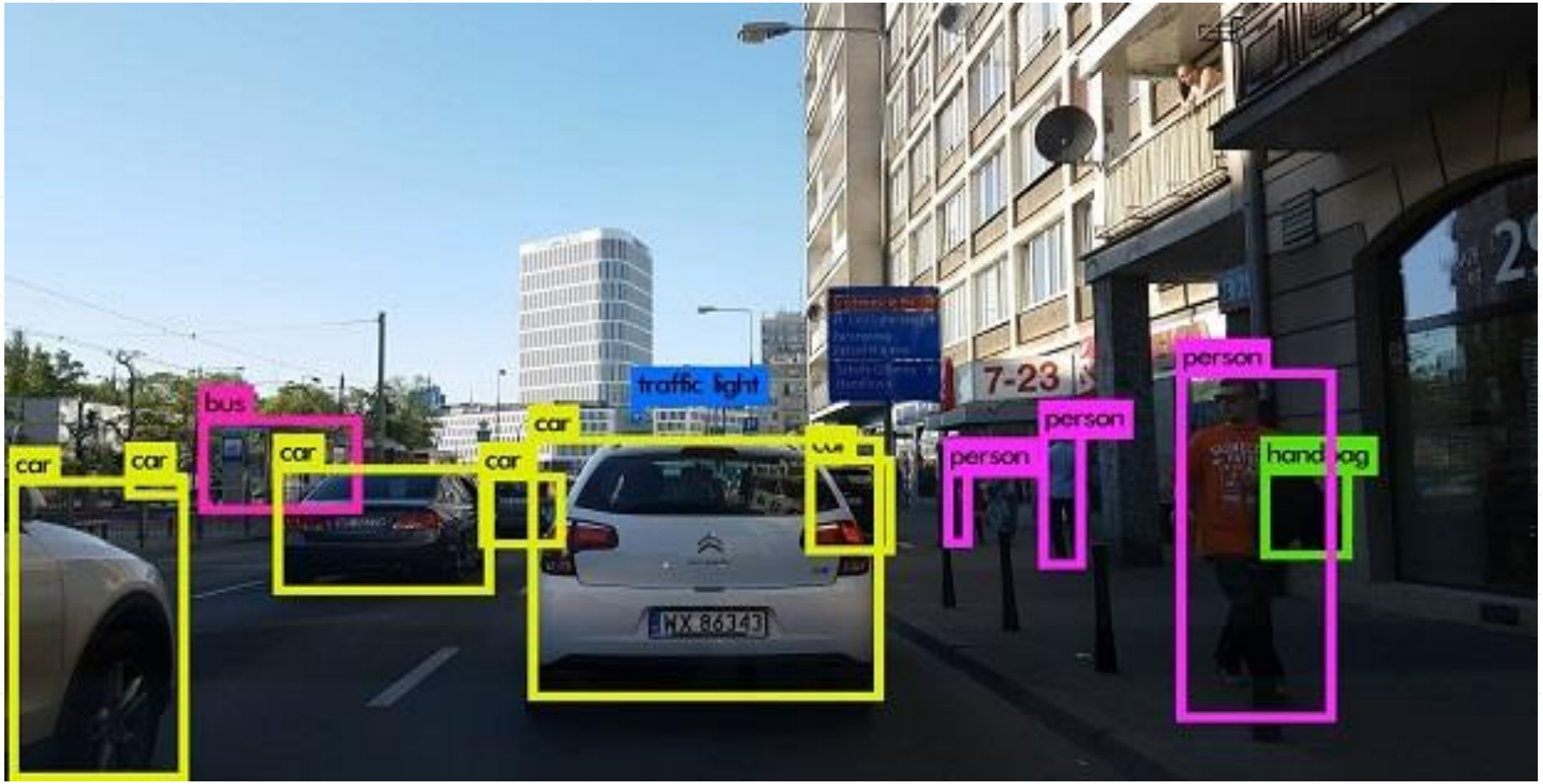
Clasificación: ¿Qué es el objeto?

Clase: person, car, dog, etc.

Confianza: ¿Qué tan seguro estamos?

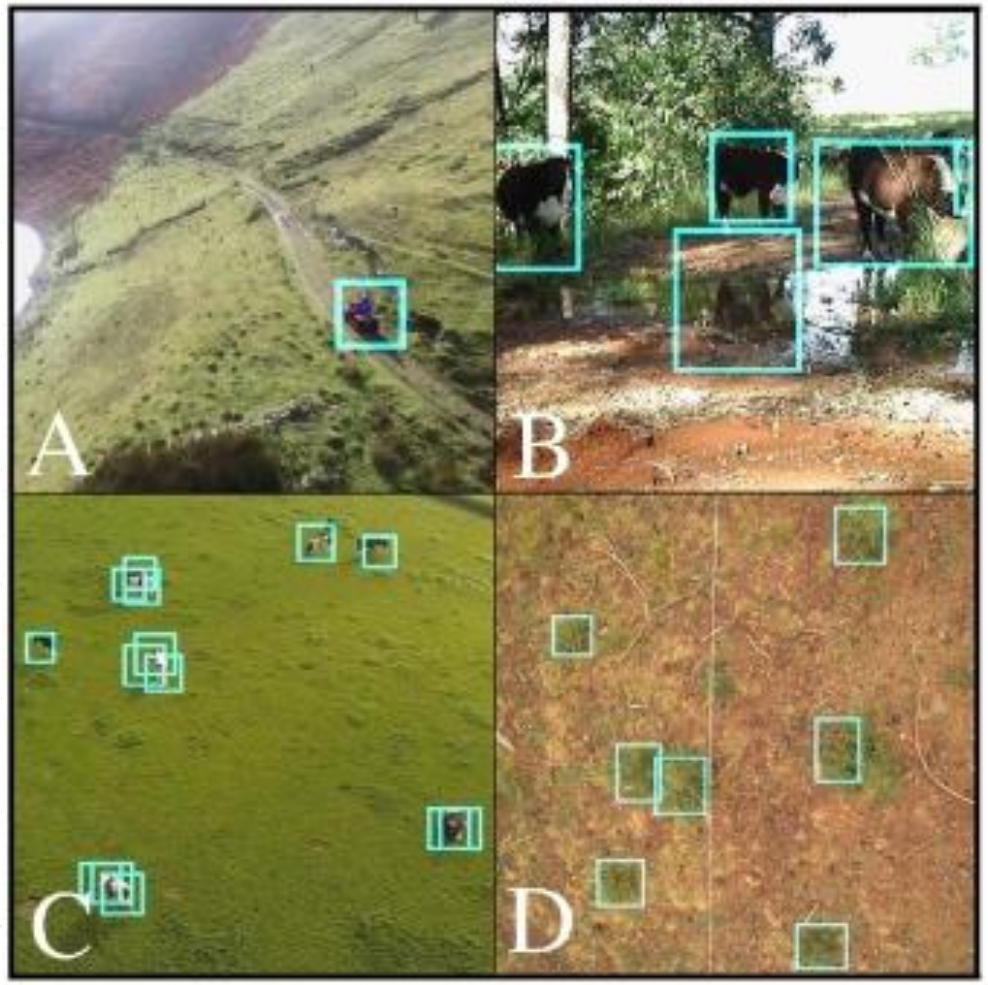
Score: 0.0 – 1.0

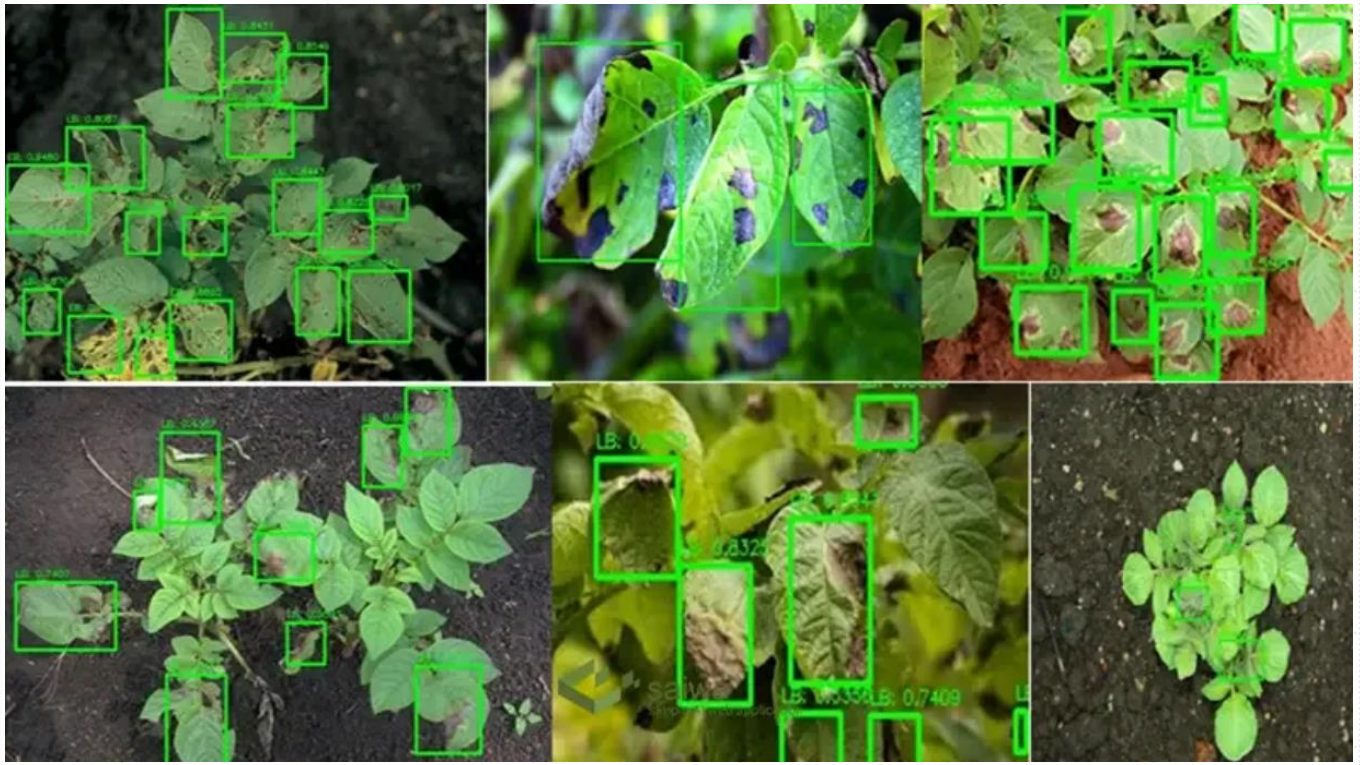
Aplicaciones Reales



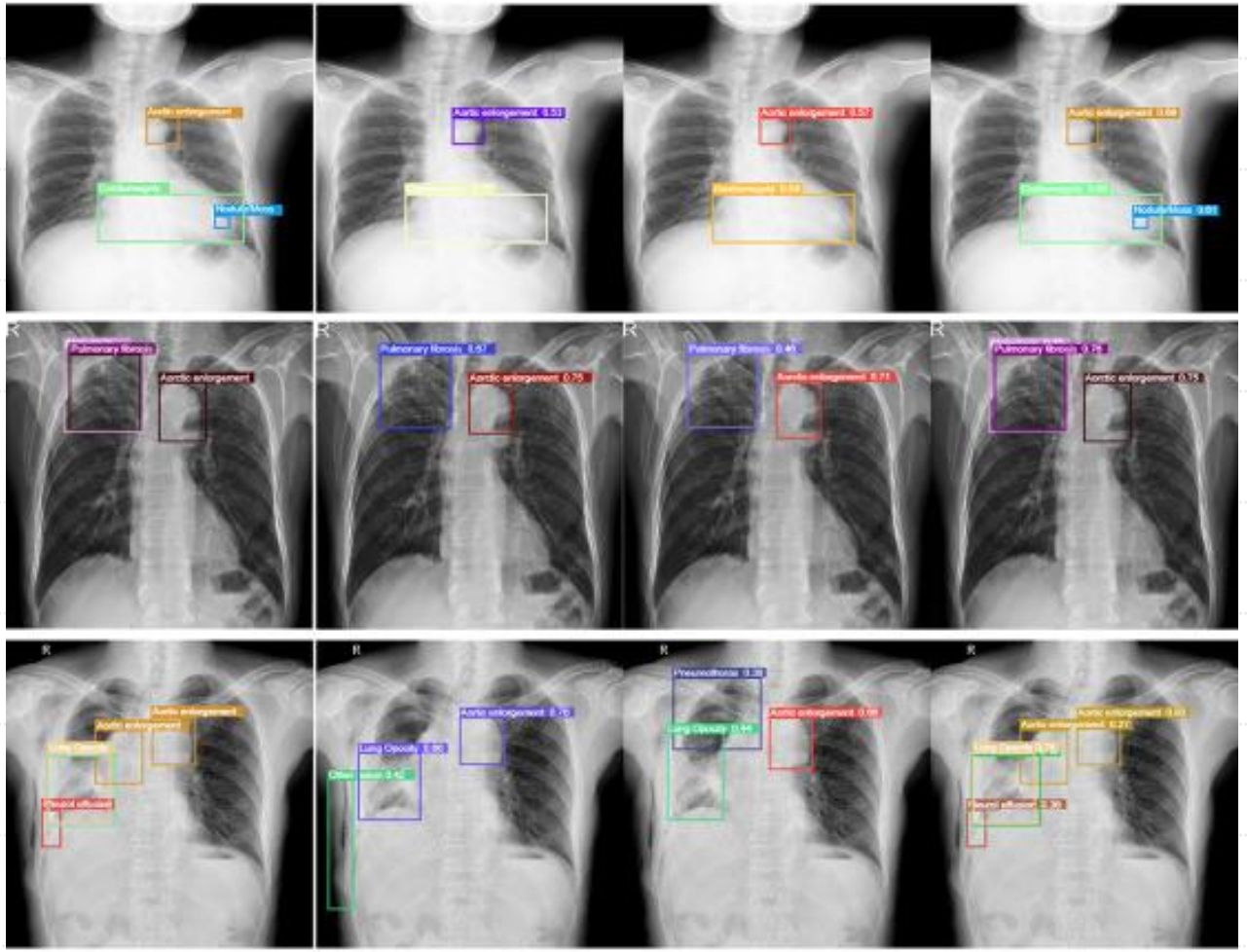
Aplicaciones Reales

Aplicaciones Reales





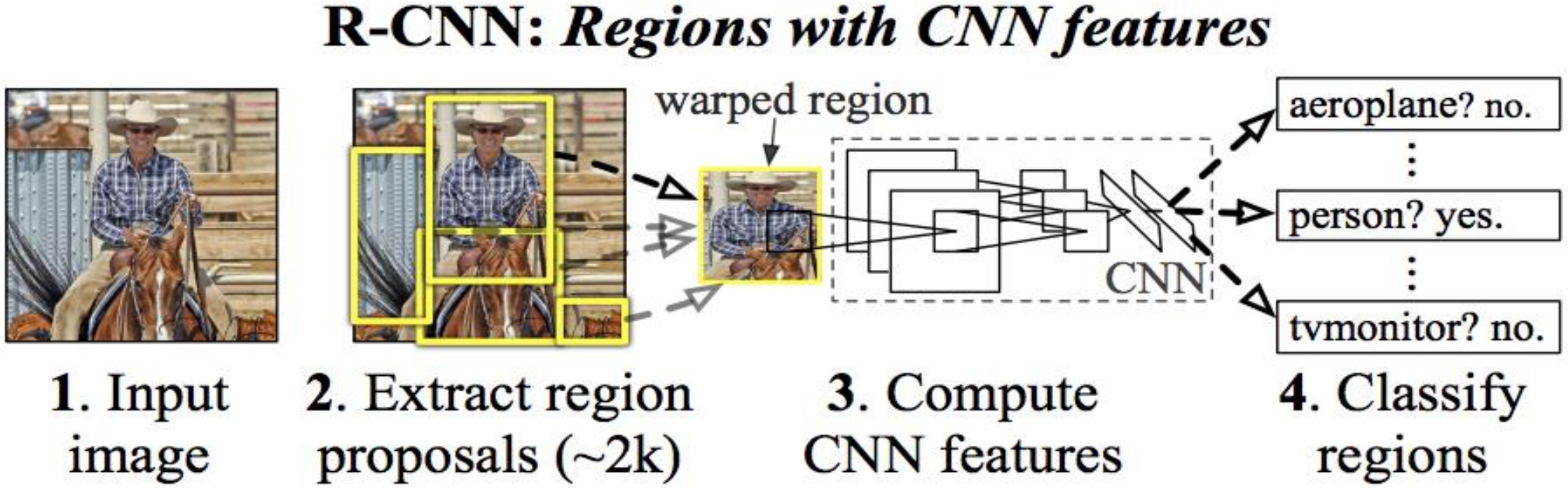
Aplicaciones Reales



(a) Ground Truth (b) yolov7-tiny (c) yolov9-c (d) our

Arquitecturas: Two-Stage

Region-based Convolutional Neural Network



Arquitecturas: Two-Stage

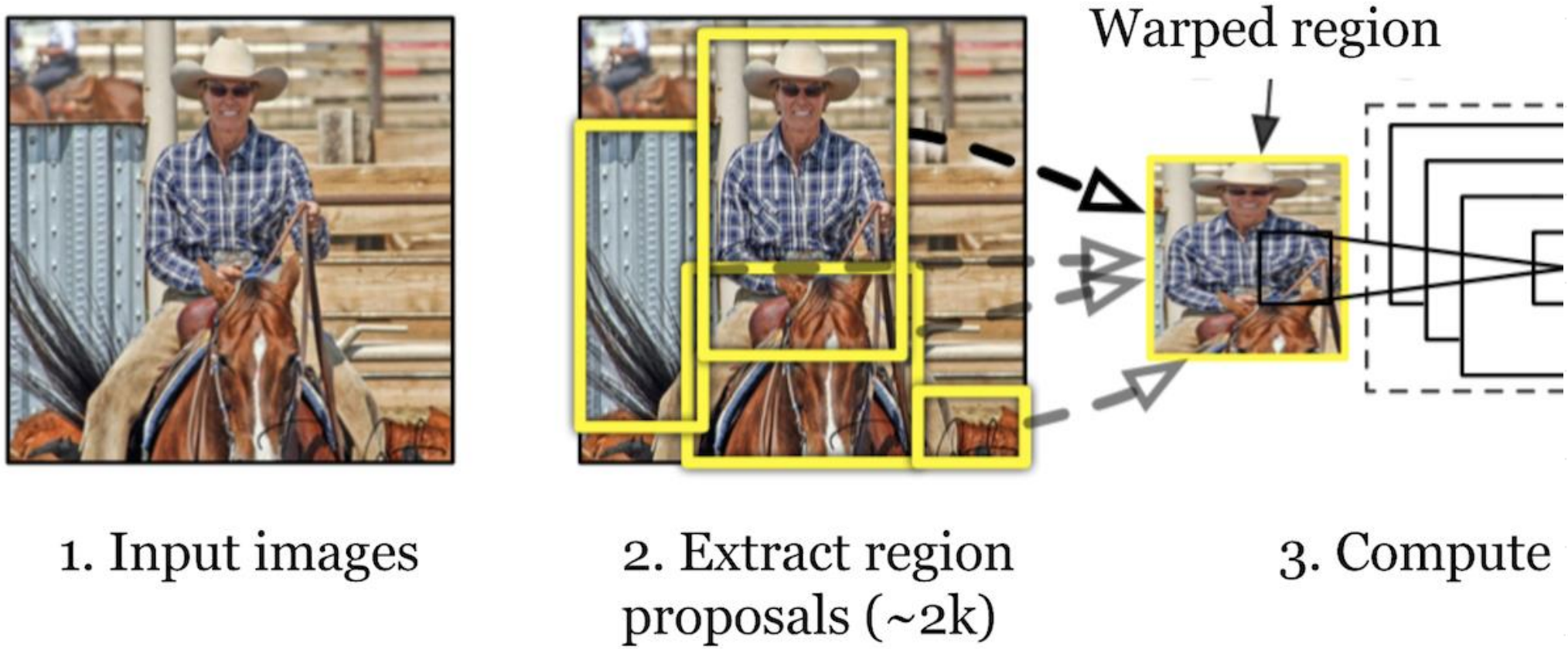
Region-based Convolutional Neural Network

1. Buscar zonas interesantes (Regiones)
    - Diferentes versiones
        - R-CNN: propone zonas de interes con un algoritmo de vision "Selective Search"
        - Fast R-CNN: En base a los features maps de salida de la CNN (mapa de activacion) extaer regiones de interes
        - Faster R-CNN: reemplaza Selective Search por una Region Proposal Network (RPN), basicamente una red que aprende a sugerir zonas de interes

Arquitecturas: Two-Stage

Region-based Convolutional Neural Network

2. Cortar cada cajita (zona de interes)



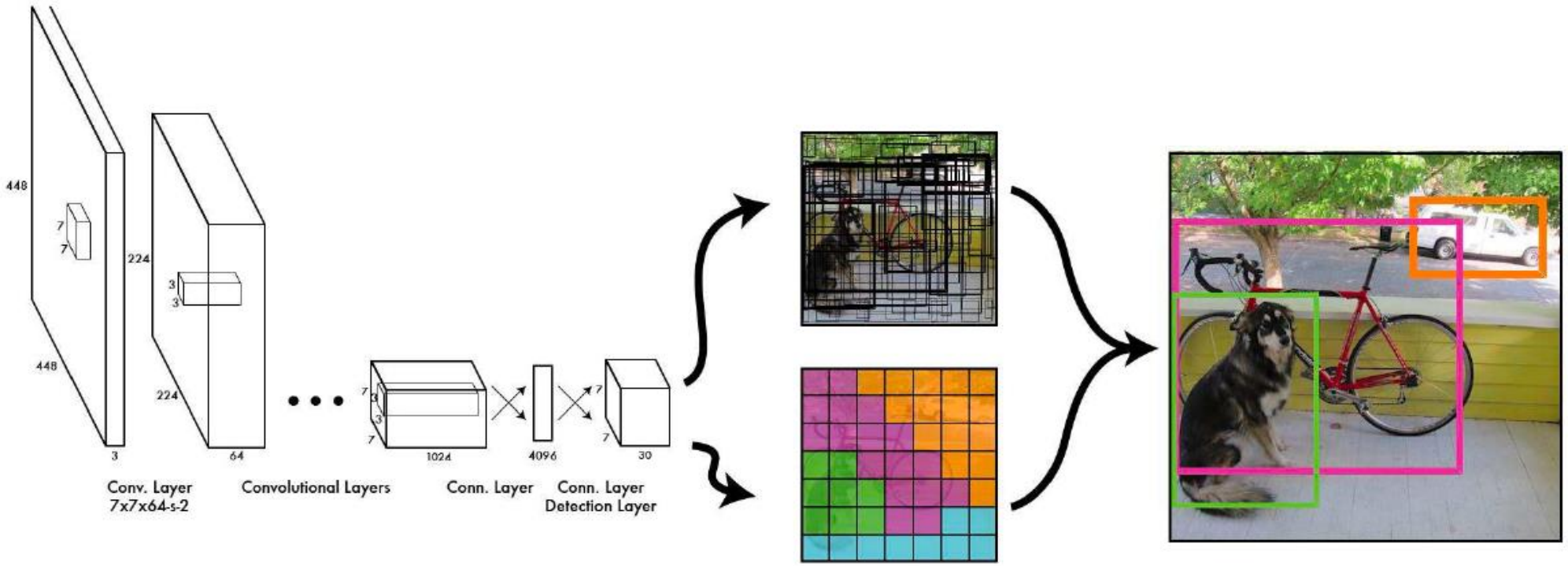
Arquitecturas: Two-Stage

Region-based Convolutional Neural Network

3. Pasar cada cajita por una red (CNN), con esto clasifica y ajusta
    - Con esas características, R-CNN usa:
        - Un **clasificador** para decir *qué objeto es* (“esto parece un gato”).
        - Un **regresor lineal** para ajustar mejor la **posición de la caja** (moverla un poquito si no está perfecta).

Arquitecturas: Single-Stage

YOLO "You Only Look Once"



Arquitecturas: Single-Stage

YOLO "You Only Look Once"

1. **Divide la imagen en una cuadrícula**
2. **Por cada cuadrito, YOLO predice varias cosas:**
    1. **(x, y)** → dónde está el centro de la caja del objeto dentro del cuadrito
    2. **(w, h)** → qué tan grande es la caja
    3. **confidence** → cuán seguro está de que haya un objeto
    4. **clases** → qué tipo de objeto es (perro, gato, pelota, etc.)
3. **Después se limpian los resultados**
    1. Algunas cajas se superponen.
    2. Se aplica **Non-Maximum Suppression (NMS)**:
       → se queda solo con las cajas más seguras y elimina las duplicadas.

Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS)

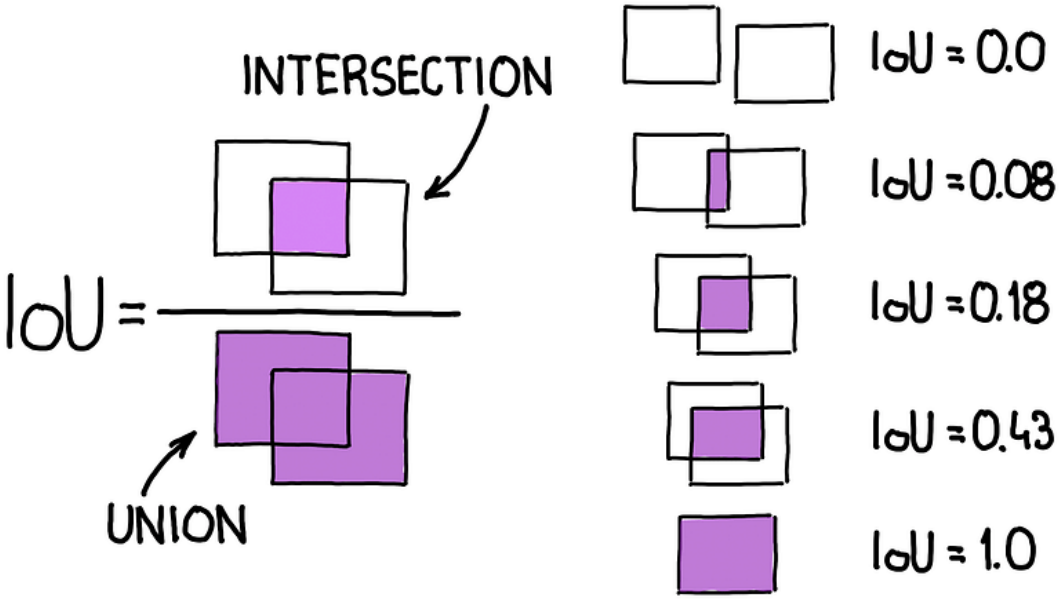
1. **Ordenar las cajas** por confianza (de mayor a menor).
2. **Tomar la mejor (la primera)**
3. **Comparar esa caja con todas las demás:**
   1. Se mide cuánto se **superponen** usando un valor llamado **IoU** (*Intersection over Union*):

$$IoU = \frac{\text{Área de superposición}}{\text{Área total combinada}}$$

   2. Si el IoU > cierto umbral (por ejemplo 0.5 o 0.6), significa que **hablan del mismo objeto** → se descarta la caja con menor score.
4. **Repetir** con la siguiente caja que no haya sido eliminada.

Métricas de Object Detection

IoU (Intersection over Union)



$$IoU = \frac{\text{Área de superposición}}{\text{Área total combinada}}$$

Métricas de Object Detection

Precision, Recall, mAP

- **Precision**: De las detecciones, ¿cuántas son correctas?
- **Recall**: De los objetos reales, ¿cuántos detectamos?
- **AP (Average Precision)**: Área bajo curva Precision-Recall
- **mAP (mean AP)**: Promedio de AP sobre todas las clases
  - **mAP@0.5**: IoU threshold = 0.5
  - **mAP@0.5:0.95**: Promedio de IoU thresholds 0.5 a 0.95
- **Interpretación**:
  - mAP = 0.5: Modelo decente
  - mAP = 0.7: Buen modelo
  - mAP = 0.9: Excelente modelo

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

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