

## **Lecture12**

**Redes Neuronales Convolucionales para  
detección de objetos y segmentación  
semántica**



# Más allá de la clasificación de imágenes

## CNNs

- Conferencia anterior: clasificación de imágenes.

## Limitaciones

- Principalmente en imágenes centradas
- Solo un objeto por imagen
- No es suficiente para muchas tareas de visión del mundo real

# Más allá de la clasificación de imágenes

Classification

single  
object



# Más allá de la clasificación de imágenes

Classification

Classif + Localisation

single  
object



# Más allá de la clasificación de imágenes

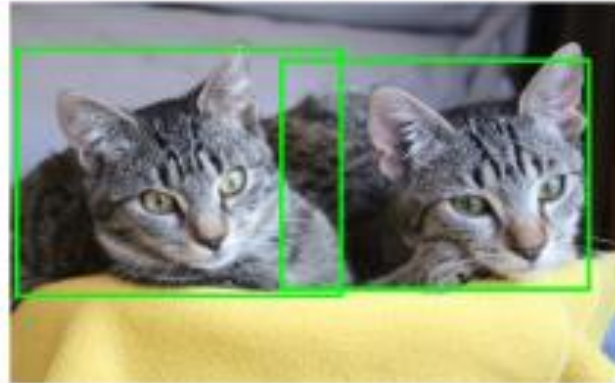
Classification

Classif + Localisation

single  
object



multiple  
objects



Object Detection



# Más allá de la clasificación de imágenes

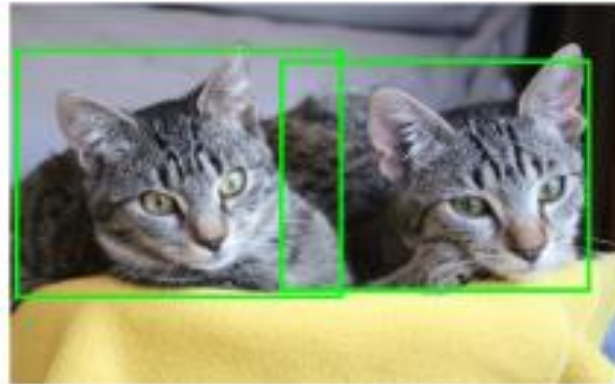
Classification

Classif + Localisation

single  
object



multiple  
objects



Object Detection

Semantic Segmentation

# Más allá de la clasificación de imágenes

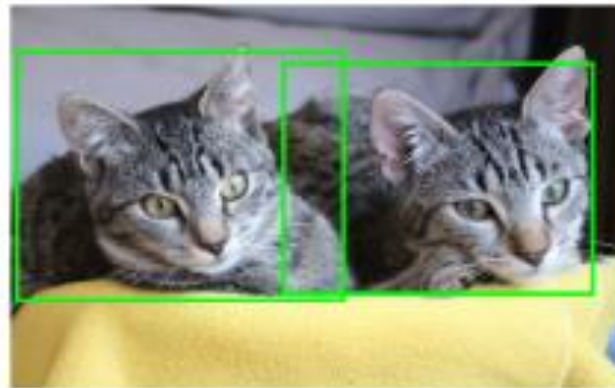
Classification

Classif + Localisation

single  
object



multiple  
objects



Object Detection

Instance Segmentation



Esquema

Localización simple como regresión

Algoritmos de detección

Redes completamente convolucionales

Segmentación semántica y de instancias

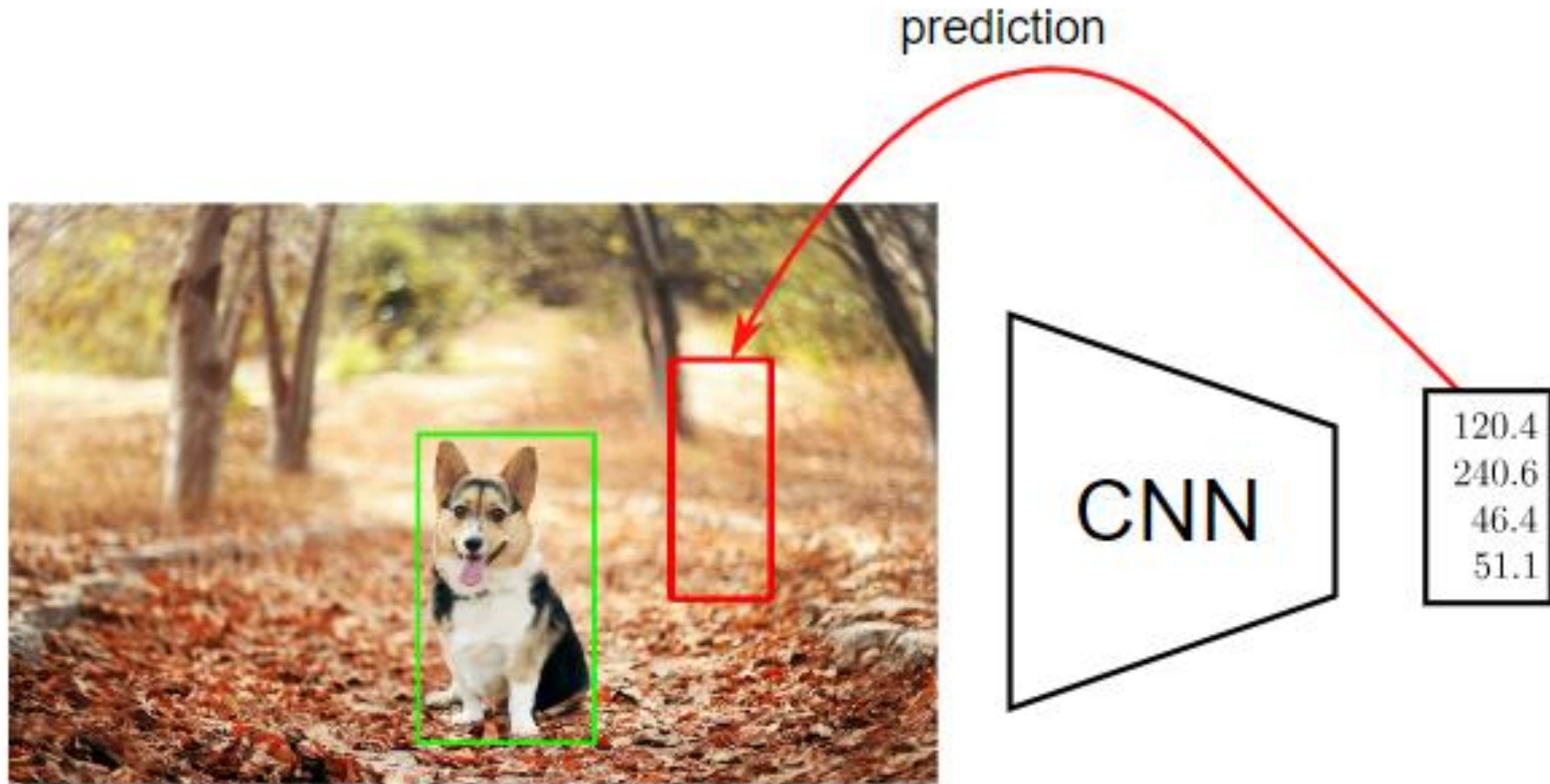
Localización

# Localización

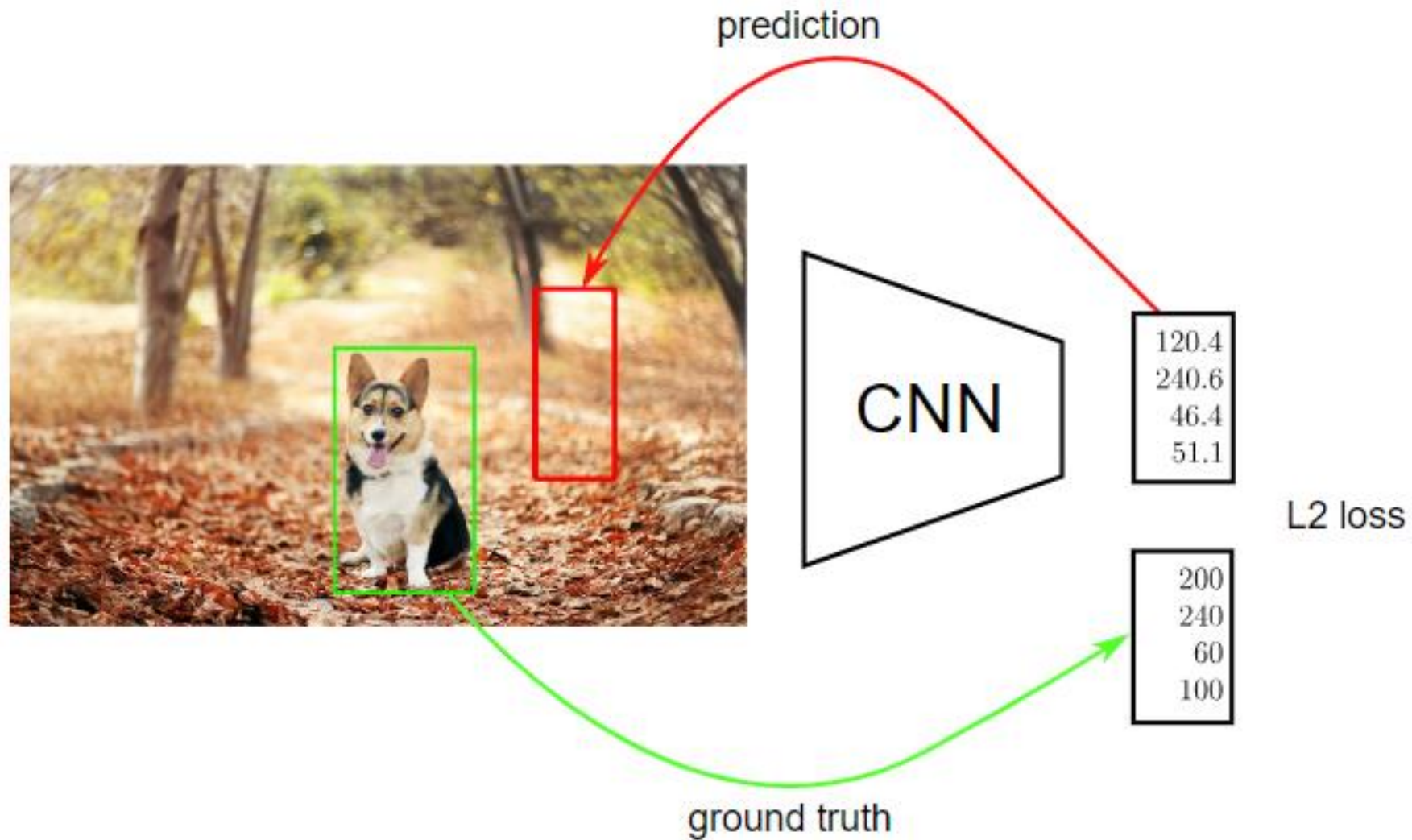


- Un solo objeto por imagen
- Predecir las coordenadas de una caja delimitadora (x, y, w, h)
- Evaluar mediante la intersección sobre unión (IoU)

# Localización como regresión

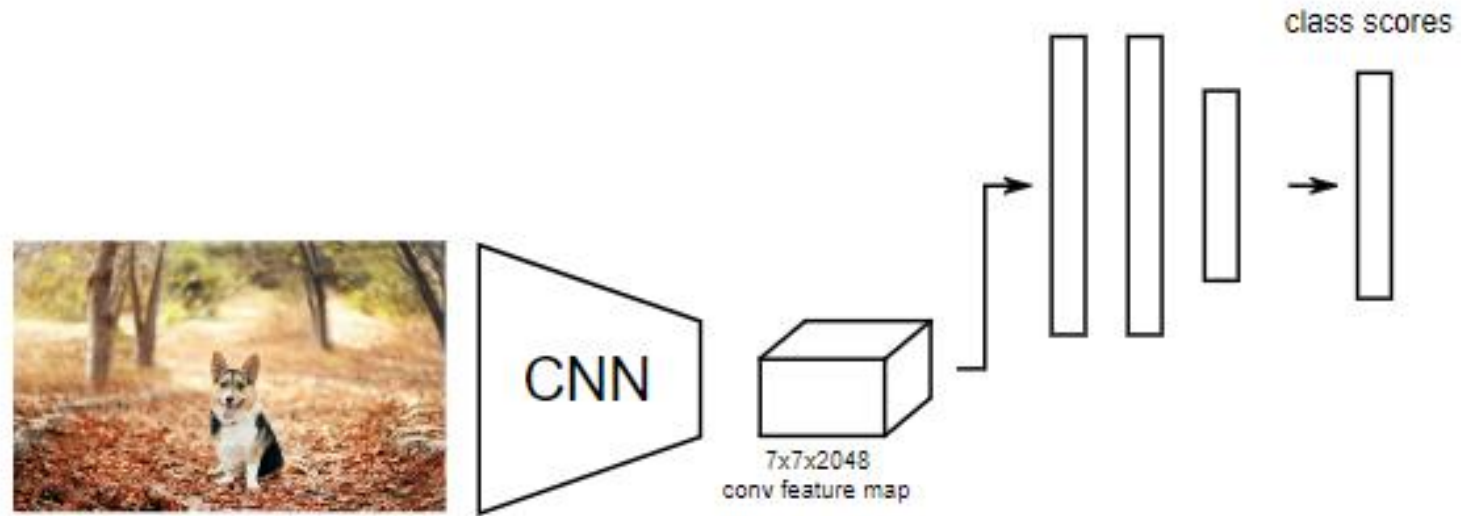


# Localización como regresión

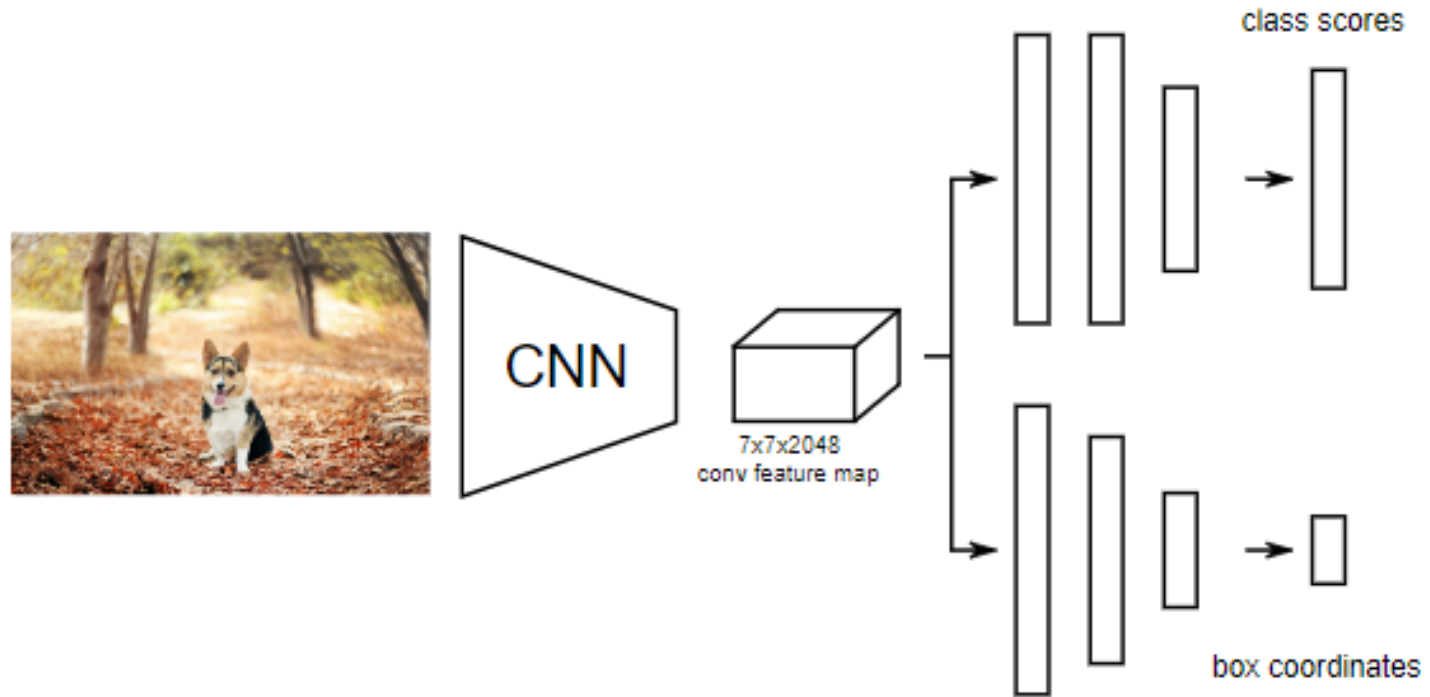




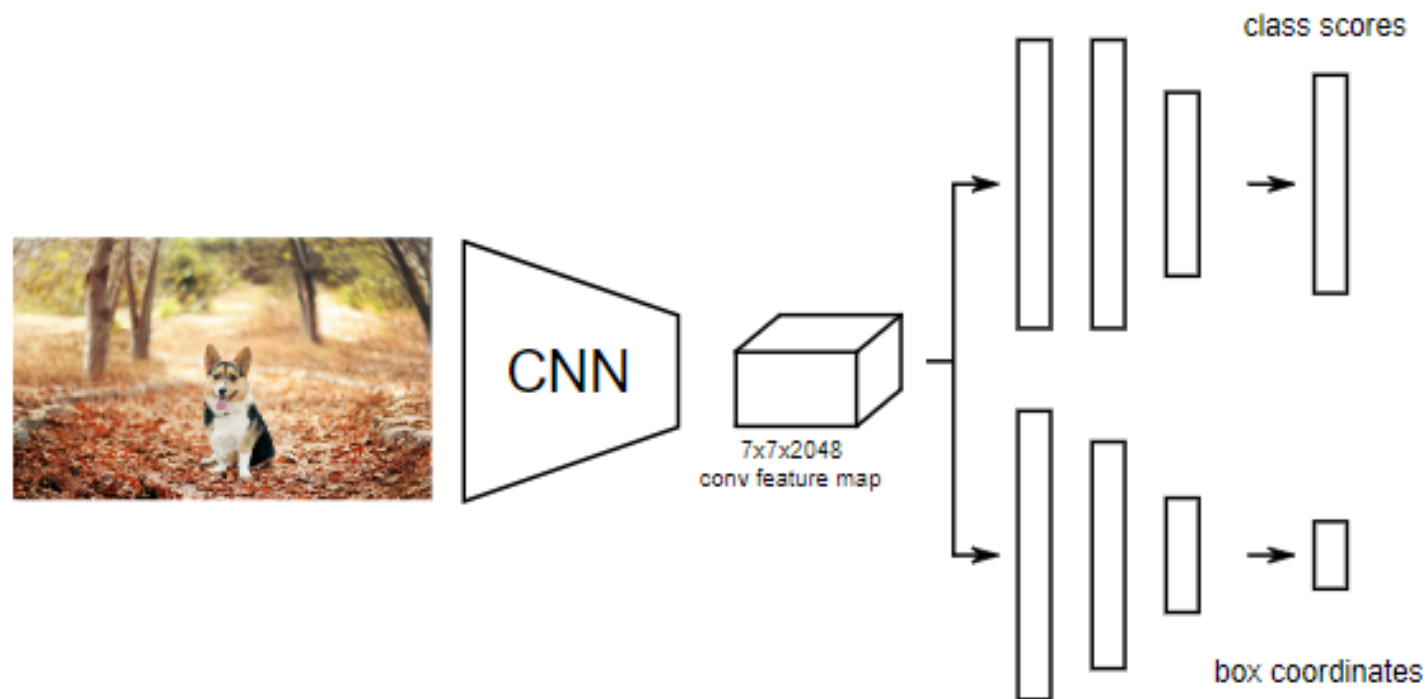
# Clasificación + Localización



# Clasificación + Localización

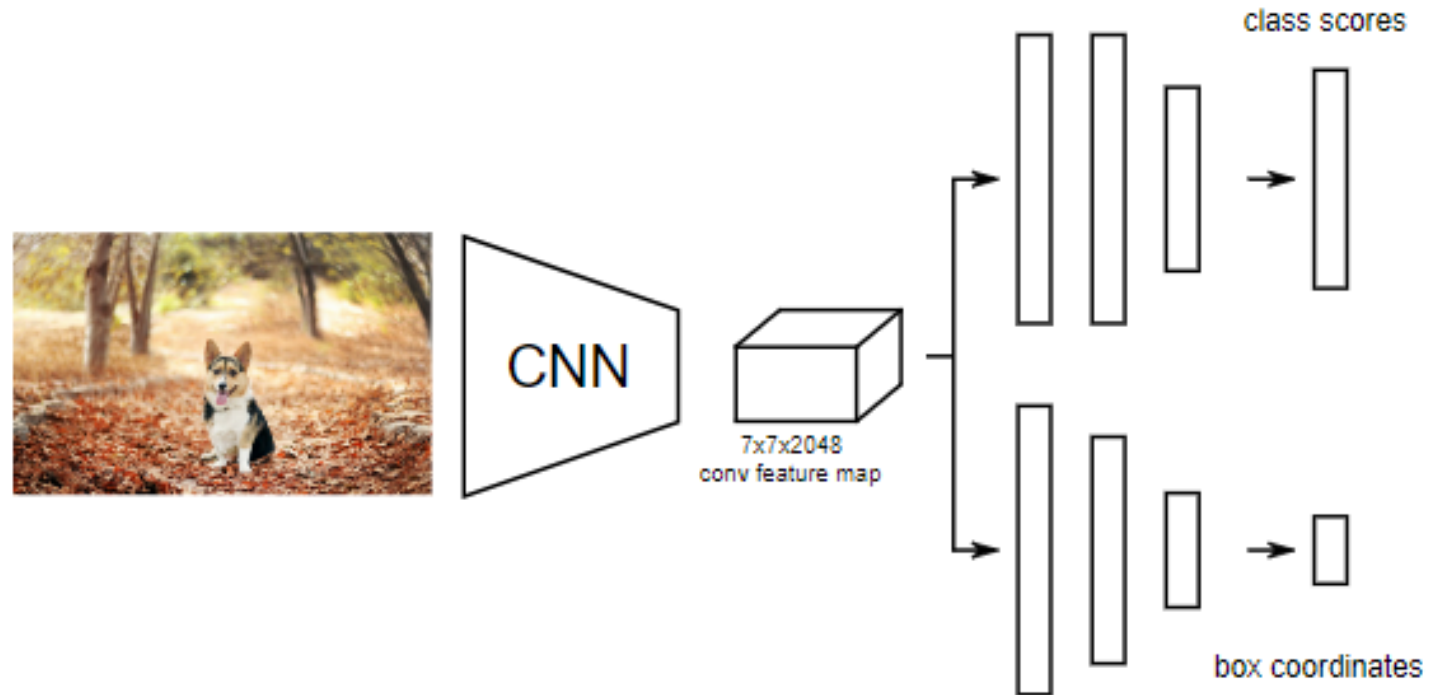


# Clasificación + Localización



- Utiliza una CNN preentrenada en ImageNet (ej. ResNet)
- La "cabeza de localización" se entrena por separado con regresión
- Posible ajuste fino de extremo a extremo para ambas tareas
- En la fase de prueba, se usan ambas cabezas

# Clasificación + Localización



$C$  clases, 4 dimensiones de salida (1 caja)

**Predecir exactamente  $N$  objetos:** predecir las coordenadas ( $N \times 4$ ) y las puntuaciones de clase ( $N \times K$ )

# Detección de objetos

No sabemos de antemano la cantidad de objetos en la imagen. La detección de objetos se basa en la *propuesta de objetos* y la *clasificación de objetos*

**Propuesta de objetos:** encontrar regiones de interés (RoIs) en la imagen.

**Clasificación de objetos:** clasificar el objeto en esas regiones.

**Dos familias principales:**

- Single-Stage: Una cuadrícula en la imagen donde cada celda es una propuesta (SSD, YOLO, RetinaNet).
- Two-Stage: Propuesta de región y luego clasificación (Faster-RCNN).



# YOLO

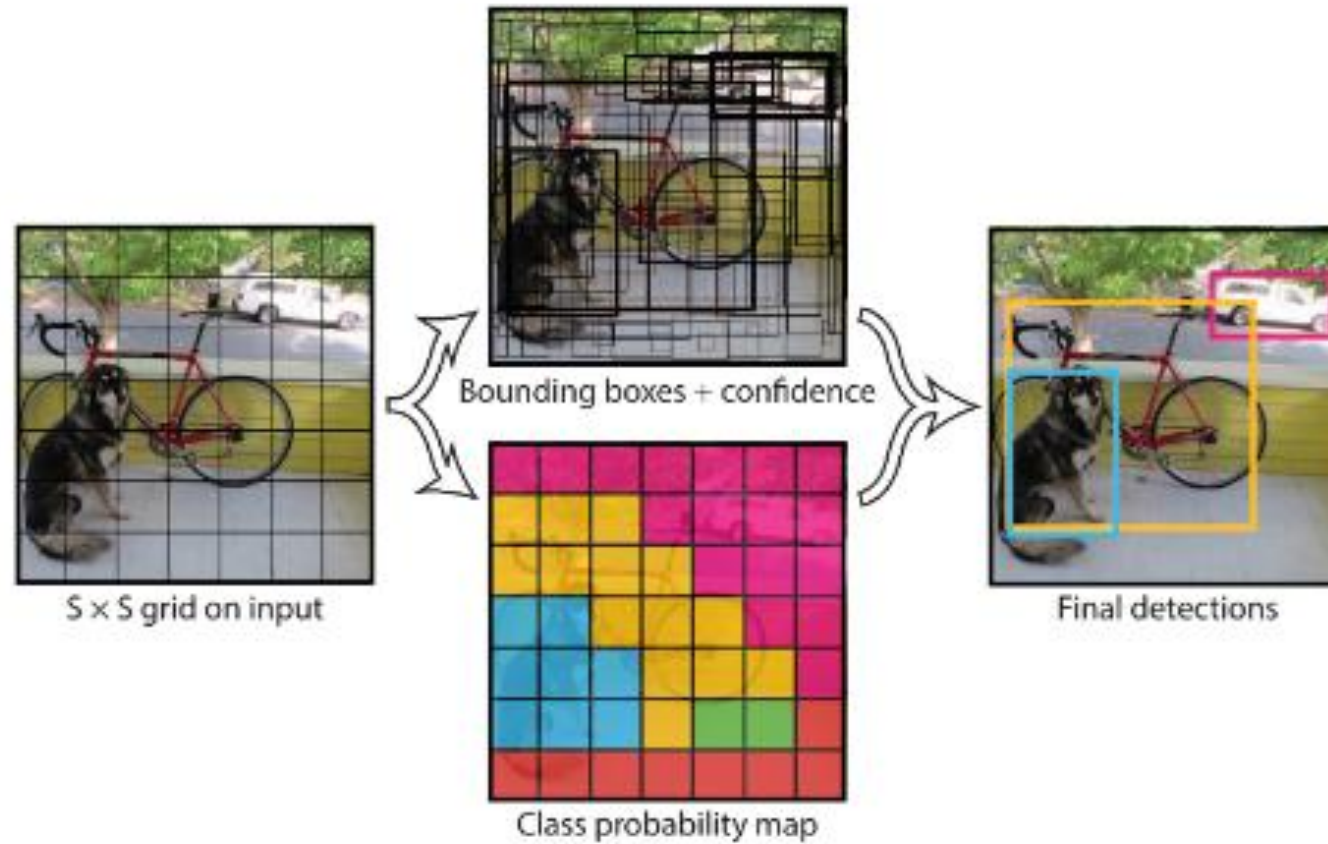


$S \times S$  grid on input

Para cada celda de la cuadrícula  $S \times S$  predecir:

- **$B$**  cajas delimitadoras y puntuaciones de confianza  $C$  ( $5 \times B$  valores) + **clases  $C$**

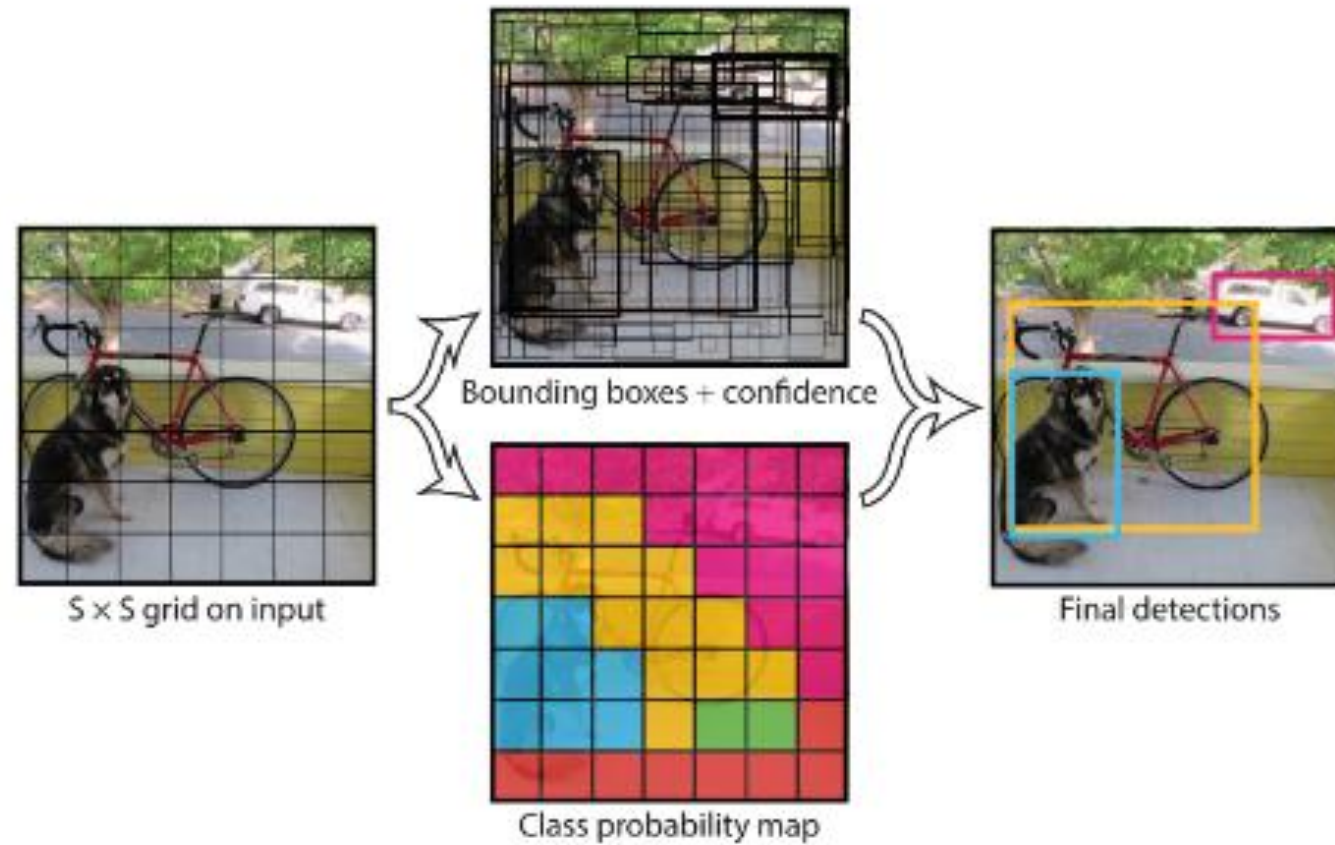
# YOLO



Para cada celda de la cuadrícula  $S \times S$  predecir:

- **$B$**  cajas delimitadoras y puntuaciones de confianza  $C$  ( $5 \times B$  valores) + clases  $C$

# YOLO



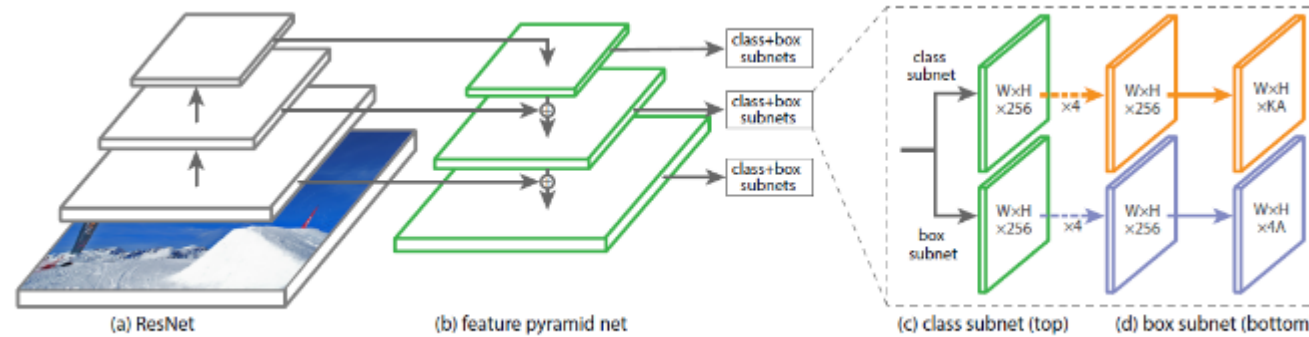
Detecciones finales:  $C_j * \text{prob}(c) > \text{umbral}$

# YOLO

- Después del preentrenamiento en ImageNet, toda la red se entrena de extremo a extremo.
- La pérdida es una suma ponderada de diferentes regresiones.

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} \mathbf{1}_{ij}^{\text{obj}} \sum_{c \in \text{Classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

# RetinaNet



- Detector de una sola etapa con:
- Múltiples escalas a través de una *Red Piramidal de Características (Feature Pyramid Network)*
- Pérdida focal (Focal Loss)\*\* para gestionar el desequilibrio entre el fondo y los objetos reales
- Consulta este [link](#) para más información.



# Propuestas de Cajas

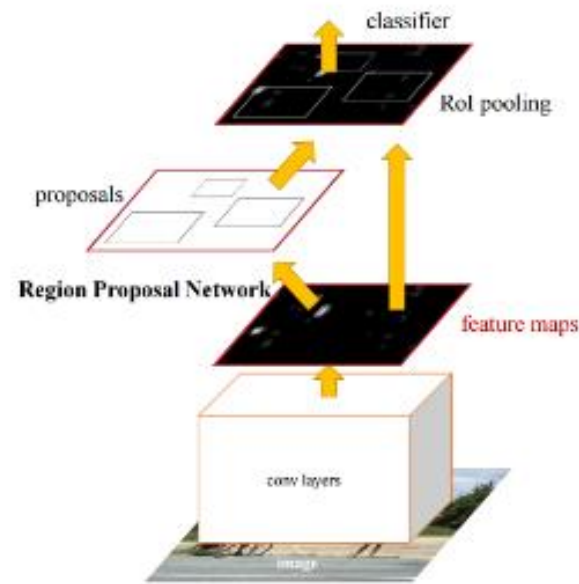
En lugar de tener un conjunto predefinido de propuestas de cajas, encuéntralas en la imagen:

- **Selective Search:** a partir de píxeles (no aprendido, ya no se usa).
- **Faster-RCNN:** Red de Propuestas de Regiones (RPN).

Operador de recorte y redimensionamiento (RoI-Pooling):

- Entrada: mapa convolucional +  $N$  regiones de interés.
- Salida: tensor de  $N \times 7 \times 7 \times \textit{profundidad}$  de las cajas.
- Permite propagar el gradiente solo en las regiones de interés, y facilita el cálculo eficiente.

# Faster - RCNN



- Entrenar conjuntamente la **RPN** y la otra cabeza
- 200 propuestas de cajas, el gradiente se propaga solo en las cajas positivas
- La propuesta de región es invariante a la traslación, en comparación con YOLO

# Medición del rendimiento

method	test size shorter edge/max size	feature pyramid	align	mAP@[0.5:0.95]	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
R-FCN [17]	600/1000			32.1	12.8	34.9	46.1
Faster R-CNN (2fc)	600/1000			30.3	9.9	32.2	47.4
Deformable [3]	600/1000		✓	34.5	14.0	37.7	50.3
G-RMI [13]	600/1000			35.6	-	-	-
FPN [19]	800/1200	✓		36.2	18.2	39.0	48.2
Mask R-CNN [7]	800/1200	✓	✓	38.2	20.1	41.1	50.2
RetinaNet [20]	800/1200	✓		37.8	20.2	41.1	49.2
RetinaNet ms-train [20]	800/1200	✓		39.1	21.8	42.7	50.2
Light head R-CNN	800/1200		✓	<b>39.5</b>	21.8	43.0	50.7
Light head R-CNN ms-train	800/1200		✓	<b>40.8</b>	22.7	44.3	52.8
Light head R-CNN	800/1200	✓	✓	<b>41.5</b>	25.2	45.3	53.1

Medidas: Precisión Promedio Media (mAP) con umbrales dados de IoU

- AP @0.5 para la clase "gato": precisión promedio para la clase, donde  $IoU(box^{pred}, box^{true}) > 0.5$

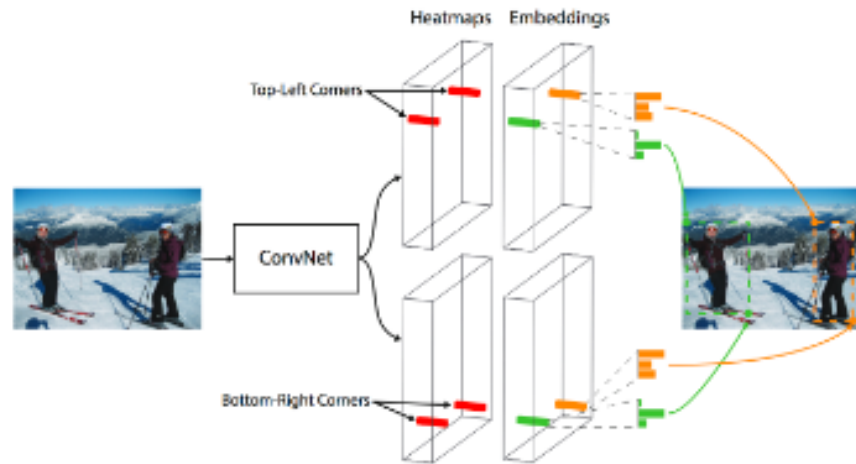
# Estado del arte

Model	FLOPs	# Params	AP <sub>val</sub>	AP <sub>test-dev</sub>
SpineNet-190 (1536) [11]	2076B	176.2M	52.2	52.5
DetectoRS ResNeXt-101-64x4d [43]	—	—	—	55.7 <sup>†</sup>
SpineNet-190 (1280) [11]	1885B	164M	52.6	52.8
SpineNet-190 (1280) w/ self-training [71]	1885B	164M	54.2	54.3
EfficientDet-D7x (1536) [56]	410B	77M	54.4	55.1
YOLOv4-P7 (1536) [60]	—	—	—	55.8 <sup>†</sup>
Cascade Eff-B7 NAS-FPN (1280)	1440B	185M	54.5	54.8
w/ Copy-Paste	1440B	185M	(+1.4) <b>55.9</b>	(+1.2) <b>56.0</b>
w/ self-training Copy-Paste	1440B	185M	(+2.5) <b>57.0</b>	(+2.5) <b>57.3</b>

- Tamaños de imagen más grandes, modelos más grandes y mejores, mejores datos aumentados.
- <https://paperswithcode.com/sota/object-detection-on-coco>

# Otros trabajos

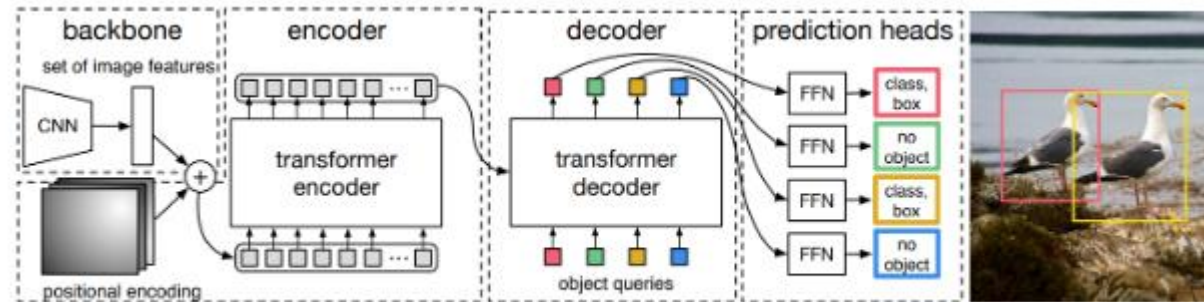
- Nuevos enfoques intentan evitar el uso de anclas.
- CornerNet solo predice los dos extremos de una caja:





# Otros trabajos

- Nuevos enfoques intentan evitar el uso de anclas.
- DeTr usa un Transformer para mapear un conjunto de características a un conjunto de cajas (con diferente cardinalidad).



La pérdida es una coincidencia por pares entre el conjunto de verdad de terreno y el conjunto de predicciones. Esta asignación óptima se calcula con el algoritmo Húngaro.

# Segmentación

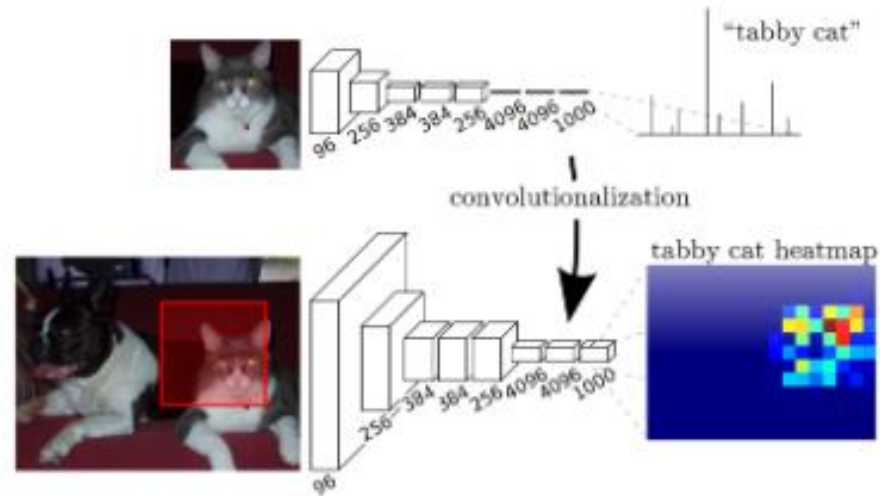
# Segmentación

- Salida de un mapa de clases para cada píxel (aquí: perro vs fondo)



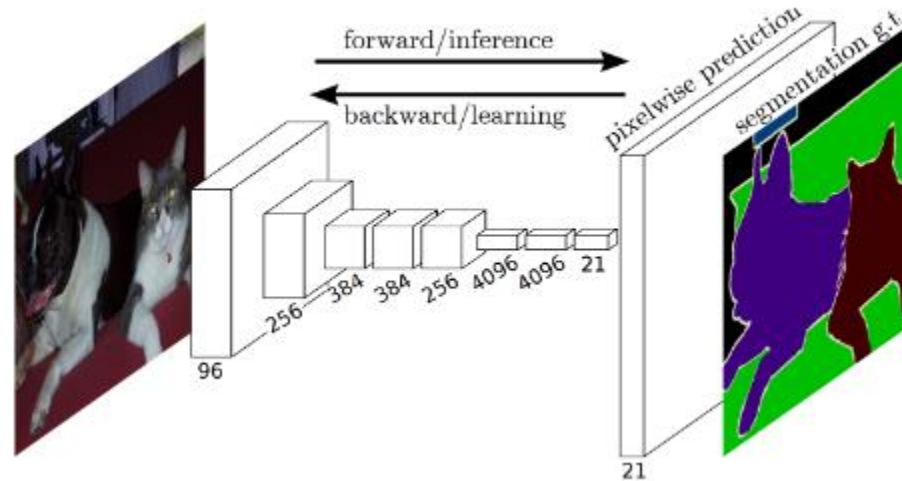
- **Segmentación de instancias:** especifica cada instancia de objeto (dos perros tienen diferentes instancias).
- Esto se puede lograr a través de **detección de objetos + segmentación.**

# Convolución



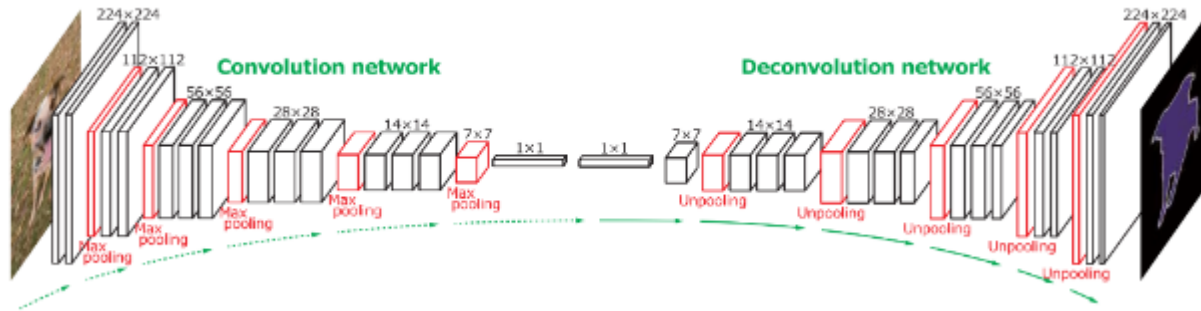
- Desliza la red con una entrada de (224, 224) sobre una imagen más grande. Salida de tamaño espacial variable.
- **Convolutionar**: cambia la capa densa (4096, 1000) a una convolución de  $1 \times 1$ , con 4096 canales de entrada y salida.
- Da una **segmentación** burda (sin supervisión adicional).

# Red Totalmente Convolutiva

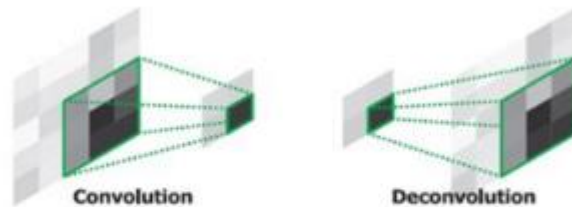


- Predecir / retropropagar por cada píxel de salida.
- Agregar mapas de varias convoluciones a diferentes escalas para obtener resultados más robustos.

# Deconvolución

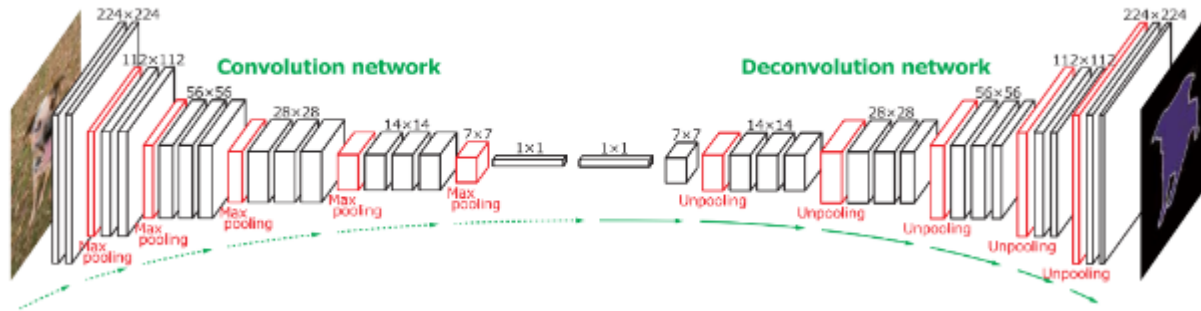


- "Deconvolución": convoluciones transpuestas



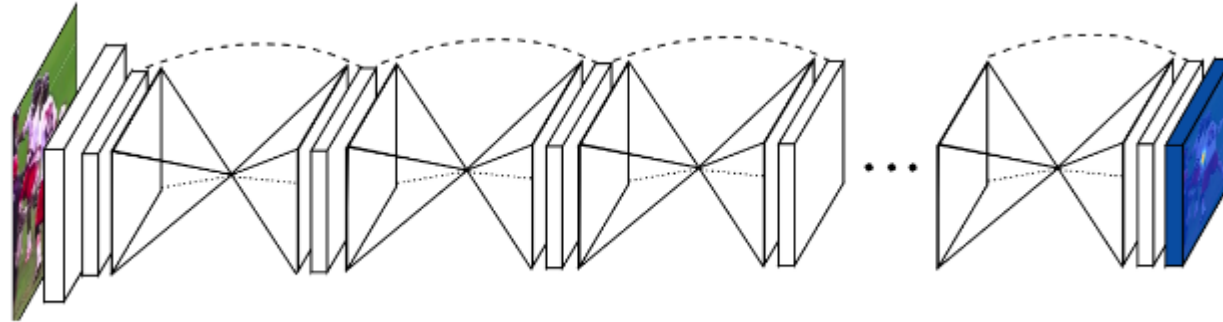


# Deconvolución



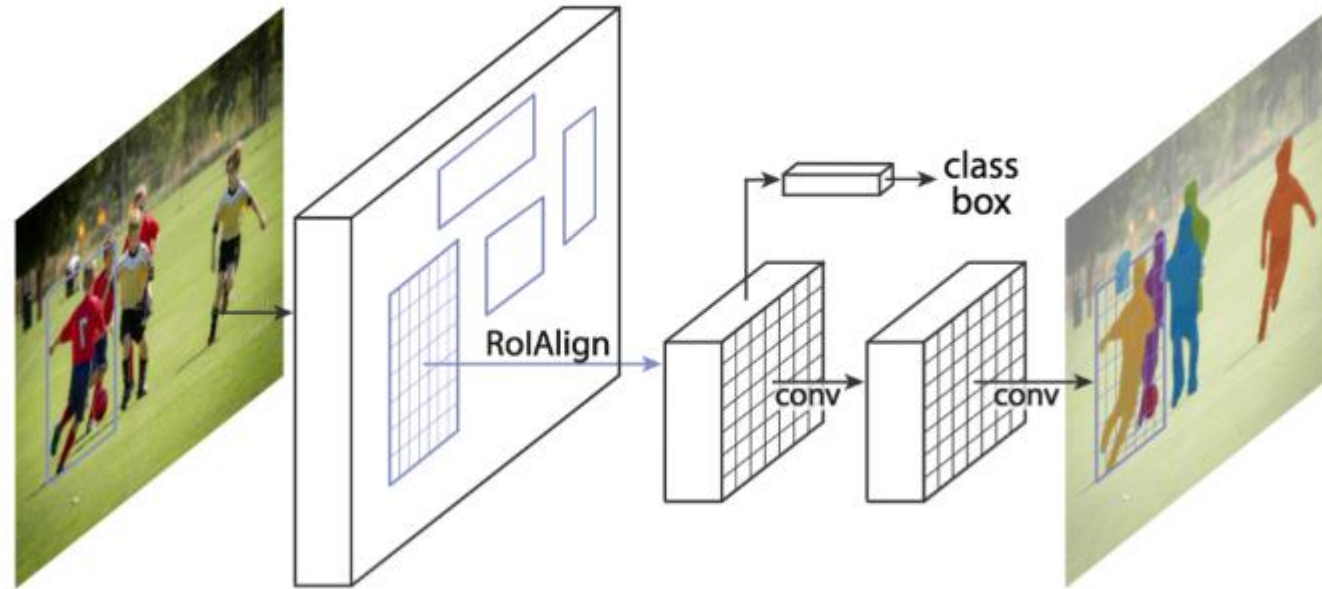
- **Conexiones de salto** entre las capas correspondientes de convolución y deconvolución.
- **Máscaras más definidas** utilizando información espacial precisa (capas tempranas).
- **Mejor detección de objetos** utilizando información semántica (capas tardías).

# Red Hourglass



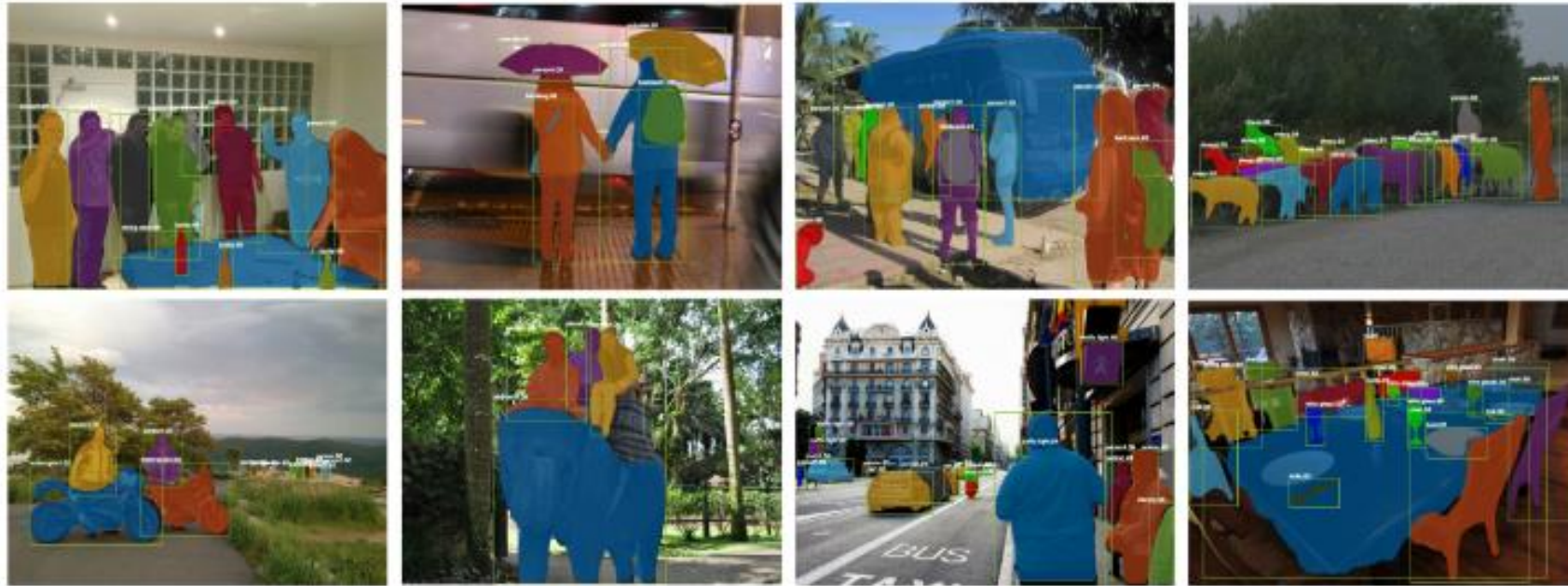
- Arquitecturas tipo U-Net repetidas secuencialmente.
- Cada bloque refina la segmentación para el siguiente.
- Cada bloque tiene una pérdida de segmentación.

# Mask-RCNN



Arquitectura Faster-RCNN con una tercera cabeza de máscara binaria.

# Resultados



- Los resultados de las máscaras aún son burdos (baja resolución de máscara).
- Excelente generalización de instancias.





# Estado del arte y enlaces

La mayoría de los benchmarks y arquitecturas recientes se reportan aquí:

<https://paperswithcode.com/area/computer-vision>

## Tensorflow

[object detection API](#)

## Pytorch

Detectron: <https://github.com/facebookresearch/Detectron>

- Mask-RCNN, RetinaNet y otras arquitecturas.
- Focal loss, Redes Piramidales de Características, etc.



# Red Neuronal para Visión: Conceptos Clave

## Características preentrenadas como base

- ImageNet: objetos centrados, dominio de imágenes muy amplio.
- Más de 1 millón de etiquetas y muchas clases diferentes que resultan en representaciones muy **generales y desenredadas**.
- Mejores redes (es decir, ResNet vs VGG) tienen un **gran impacto**.

## Ajuste fino (Fine tuning)

- Añadir nuevas capas sobre la capa convolucional o densa de las CNNs.
- **Ajuste fino** de toda la arquitectura de extremo a extremo.
- Utilizar un conjunto de datos más pequeño pero con etiquetas más ricas (cajas delimitadoras, máscaras...).