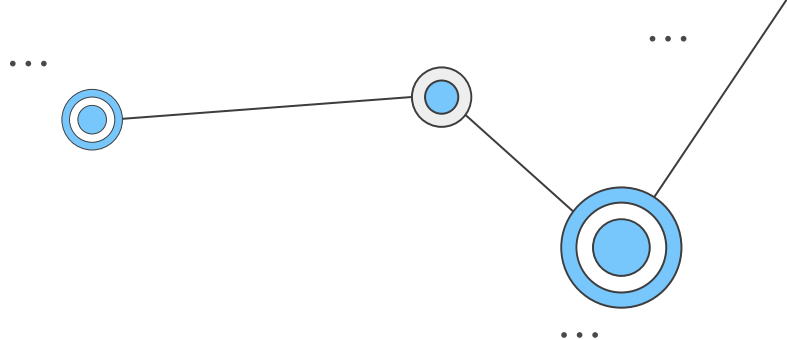
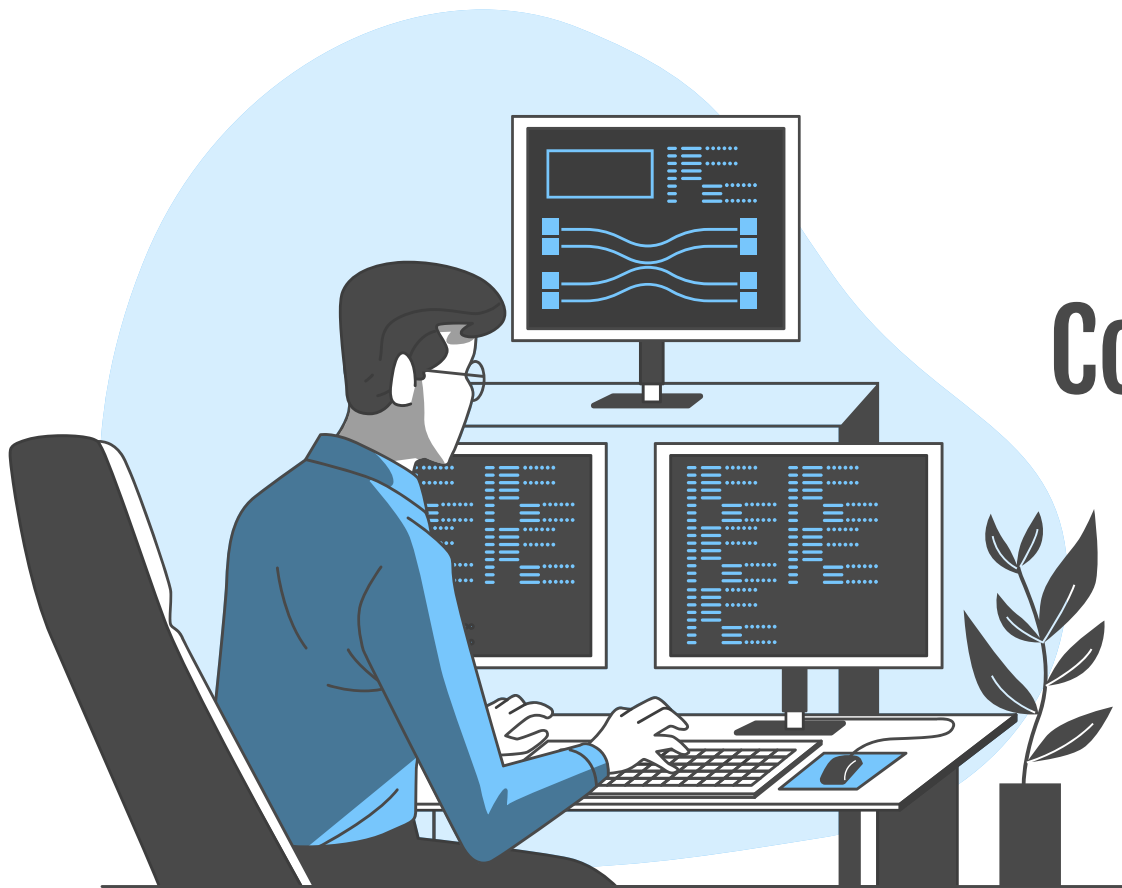
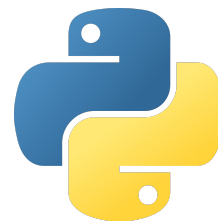


Redes Convolucionales



Contenido

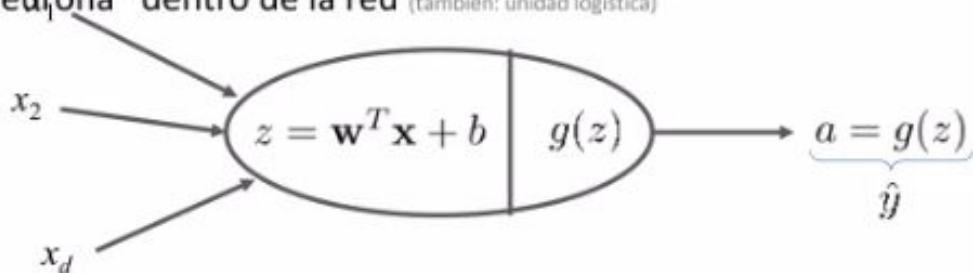
1. Redes Neuronales

- 2. Redes Neuronales Convolucionales
- 3. Aspectos Prácticos de CNNs
- 4. Detección de Objetos
- 5. Ejemplos de Detectores de Objetos

Redes Neuronales

- **Unidad neuronal**

- Representa una “neurona” dentro de la red (también: unidad logística)



- Partes importantes:
 - *Función de propagación* (combinación lineal de las entradas): $z \in \mathbb{R}$
 - Contiene pesos ($\mathbf{w} \in \mathbb{R}^d$) y bias ($b \in \mathbb{R}$)
 - *Función de activación*: $g \in \mathbb{R}$ y activación $a \in \mathbb{R}$
- Entrada: $\mathbf{x} \in \mathbb{R}^d$ (d atributos)
- Salida: $\hat{y} \in \mathbb{R}$

Redes Neuronales

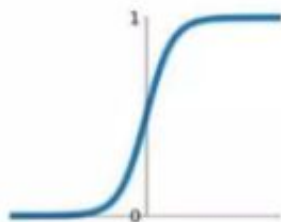
Funciones de Activación

- Tienen como entrada la combinación lineal $z = w^T x + b$

- **Función sigmoidea**

- Llamada función logística
- Mapea a (0,1)
- Se usa para la salida de toda la red

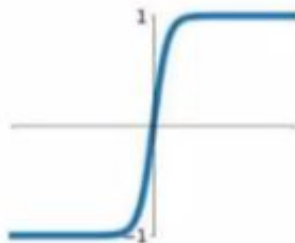
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



- **Función tangente hiperbólica**

- No tiene "bias" (centrado en 0)
- Mapea a (-1, 1)
- Se usa para capas ocultas

$$a(z) = \tanh(z)$$



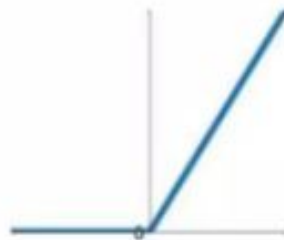
Redes Neuronales

Funciones de Activación

- **Función ReLU**

- Significa: "Rectified Linear Unit"
- Ventaja: facilidad de cálculo
- Se suele usar en capas ocultas

$$a(z) = \max(0, z) = \begin{cases} 0, & \text{si } z < 0 \\ z, & \text{si } z \geq 0 \end{cases}$$



- **Función Leaky ReLU**

- Variación de ReLU
- El valor no se fija a cero: pequeña pendiente
- Puede facilitar el "aprendizaje" (optimización)

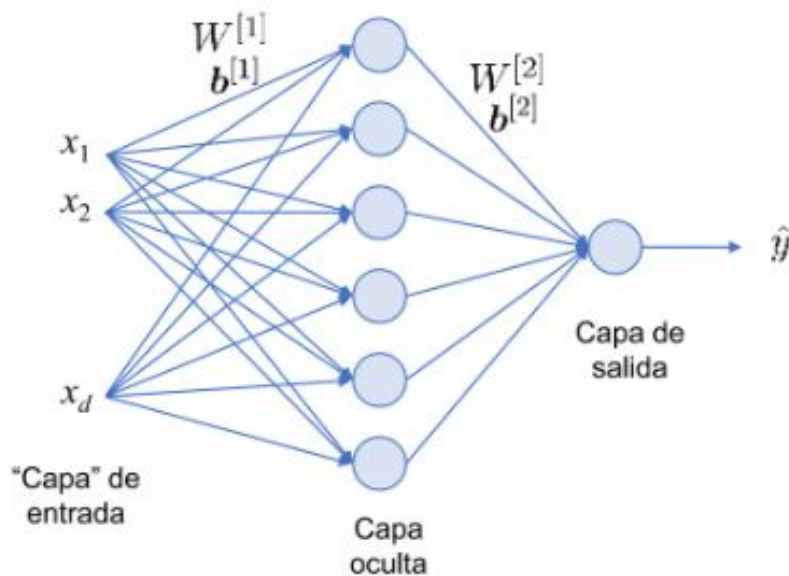
$$a(z) = \max(0.01z, z) = \begin{cases} 0.01z, & \text{si } z < 0 \\ z, & \text{si } z \geq 0 \end{cases}$$



Redes Neuronales

Redes Neuronales Simples (pocas capas)

- Ejemplo: red neuronal con una capa oculta y una capa de salida (el corchete indica el número de capa)

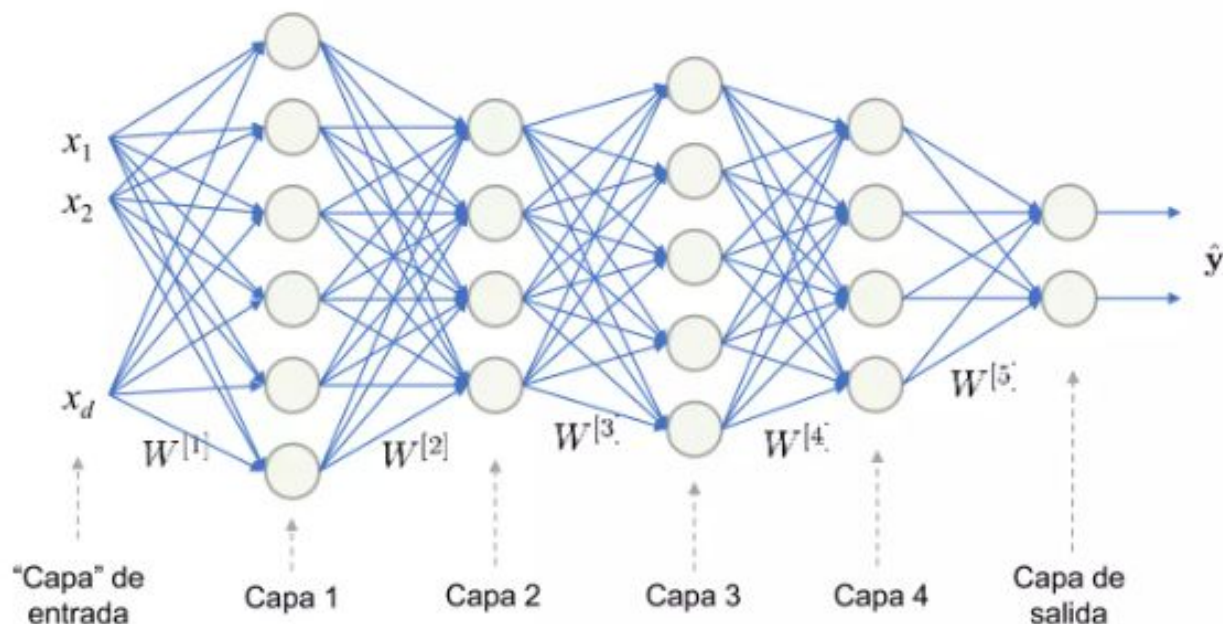


- En general puede haber más de 1 salida (para clasificar varias clases)

Redes Neuronales

Redes Neuronales Profundas

- Llamadas: *Deep Neural Networks (DNN)*
- Son redes neuronales con varias capas



Redes Neuronales

Aplicación en Visión Computacional

Visión computacional “tradicional”

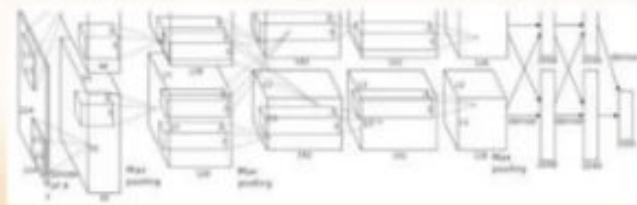
- Se basa en extracción manual de atributos (features)
- Los atributos se usan para el entrenamiento



**Entrenamiento
basado en atributos**
(SVM, redes neuronales,
random forest,
AdaBoost, etc.)

Visión computacional basada en CNN (“convolutional neural networks”)

- Toda la imagen se usa para el entrenamiento



Entrenamiento usando CNN

Convolución

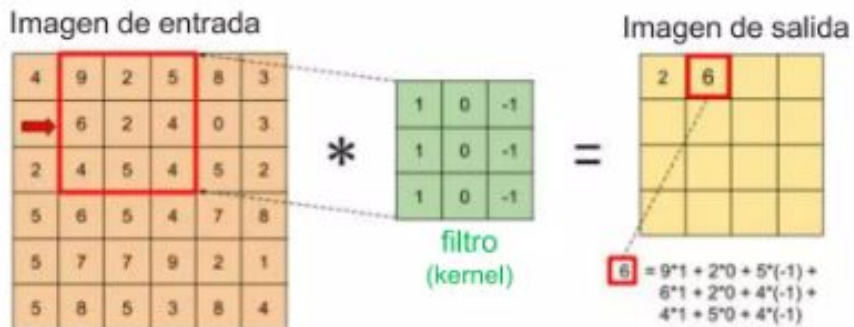
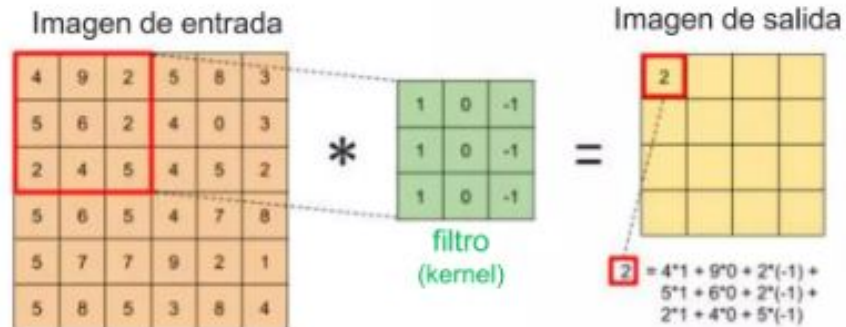
La convolución es una operación matemática que combina dos funciones para describir la superposición entre ambas. La convolución toma dos funciones, "desliza" una sobre la otra, multiplica los valores de las funciones en todos los puntos de superposición, y suma los productos para crear una nueva función.

...

Redes Neuronales Convolucionales

Convoluciones en 2 Dimensiones

- Ejemplo de convolución:



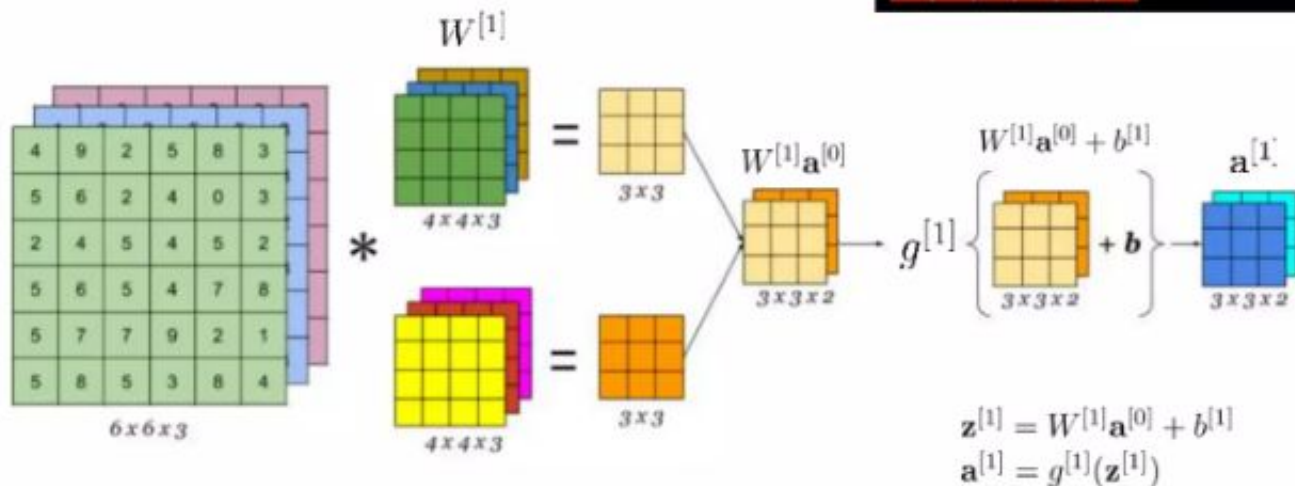
Redes Neuronales Convolucionales

Capa de Convolución con imágenes a color

- Componentes:

- Mapa de activación z (convolución de la entrada con los filtros + bias)
- Funciones de activación g (aplicadas a los mapas de activación)
 - Usualmente ReLU (eventualmente tanh)

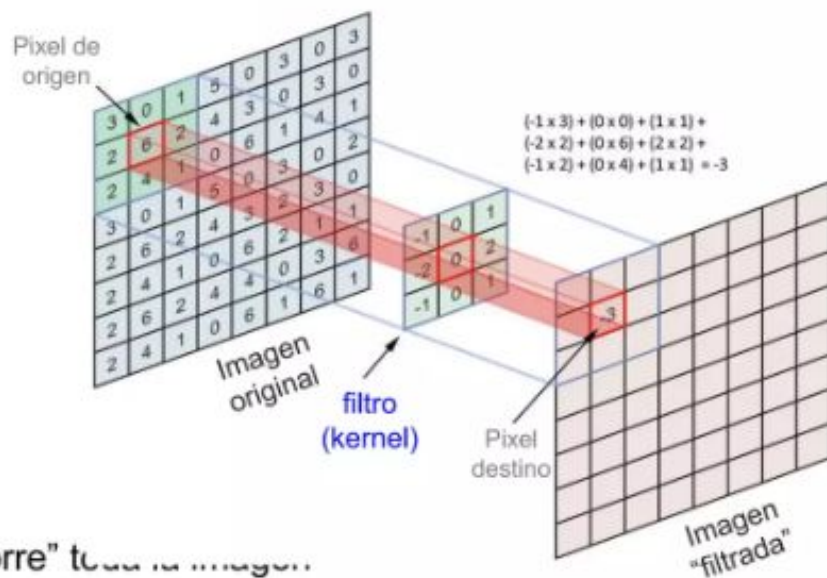
- Ejemplo



Redes Neuronales Convolucionales

Convoluciones en 2 Dimensiones

- Convolución consiste en:
 - Sobreponer un filtro a la imagen, realizar las multiplicaciones término a término, y sumar el resultado

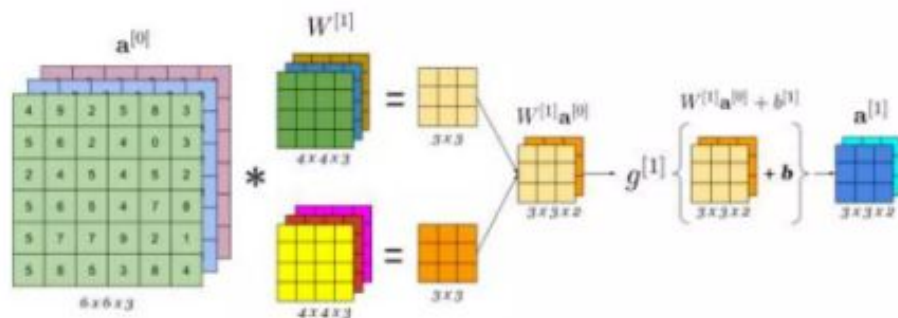


- El filtro "recorre" toda la imagen.

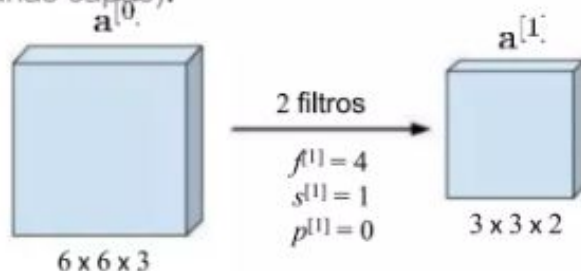
Redes Neuronales Convolucionales

Capa de Convolución

- Simplificación de la representación:



- De manera simplificada (cuando hay varias capas):

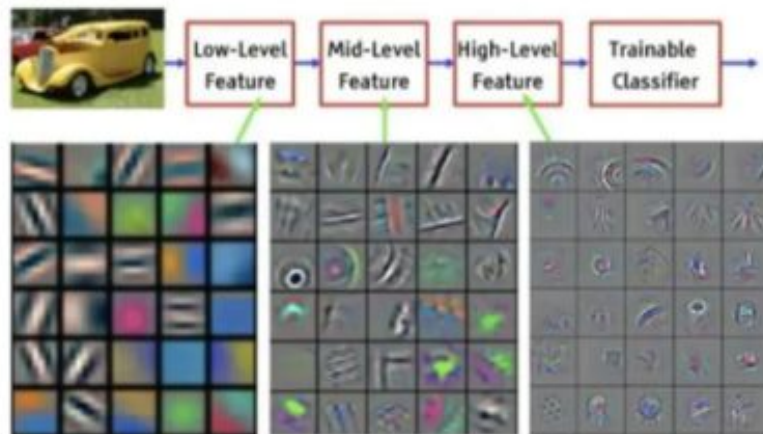




La operación convolución en una CNN

La convolución aplicada en una imagen para una CNN, tiene la función de “aprende” a extraer los atributos o características de los objetos que constituyen la imagen.

En la fase de entrenamiento de una CNN, la CNN “aprende” los valores óptimos para cada filtro o kernel, esto valores óptimos son los que permiten extraer atributos significativos (texturas, bordes, formas), y estos atributos conforman el mapa de características de la imagen o patrones, patrones que luego se utilizaran para clasificar los objetos en otras imágenes.

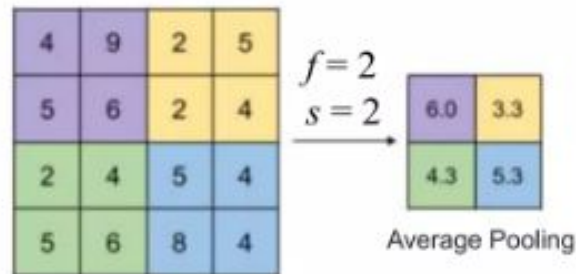
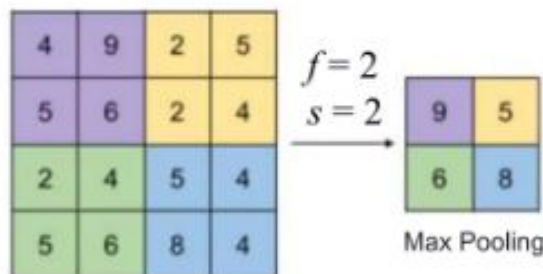


Visualizing and understanding convolutional networks (2014), M. Zeiler and R. Fergus [pdf]

Redes Neuronales Convolucionales

Capa de Pooling

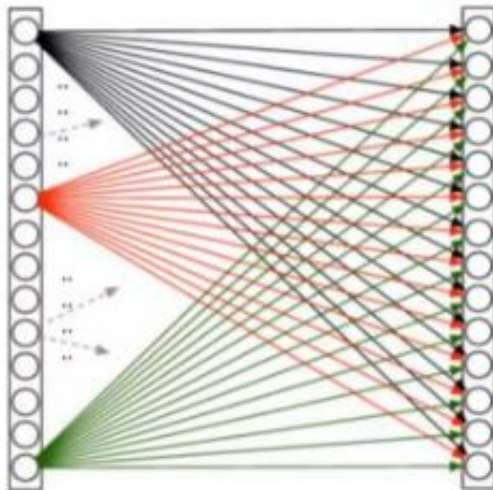
- Reduce el tamaño de las representaciones ("downsampling")
 - Opera en cada canal de manera independiente
 - No introduce ningún nuevo parámetro (no hay que "aprender" nada)
- Tipos más utilizados
 - **Max Pooling**: utiliza el valor máximo de cada conjunto de píxeles
 - **Average Pooling**: utiliza el valor promedio de cada conjunto de píxeles



Redes Neuronales Convolucionales

Capa "Fully Connected"

- Es una capa que se encuentra completamente conectada (como en redes neuronales ordinarias)

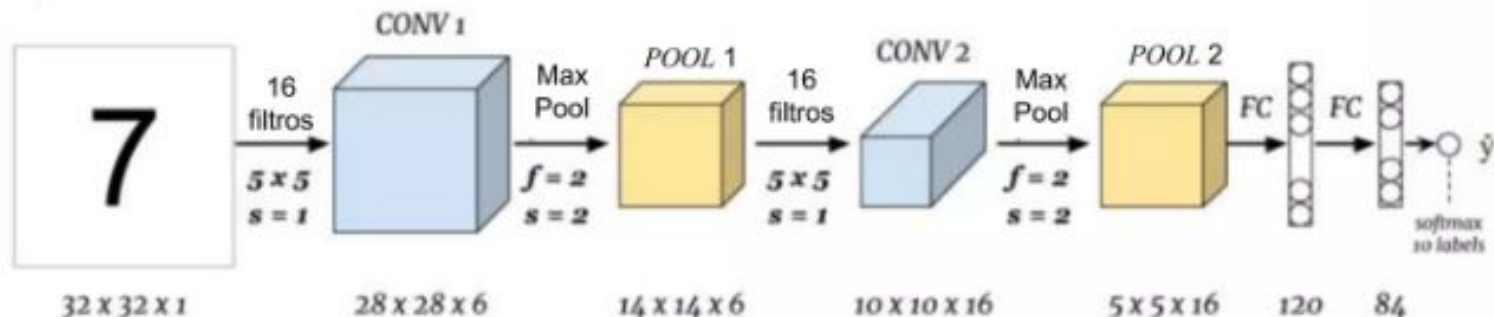


- Se le suele abreviar como FC

Redes Neuronales Convolucionales

Ejemplos de C

- Red “similar” a LeNet – 5

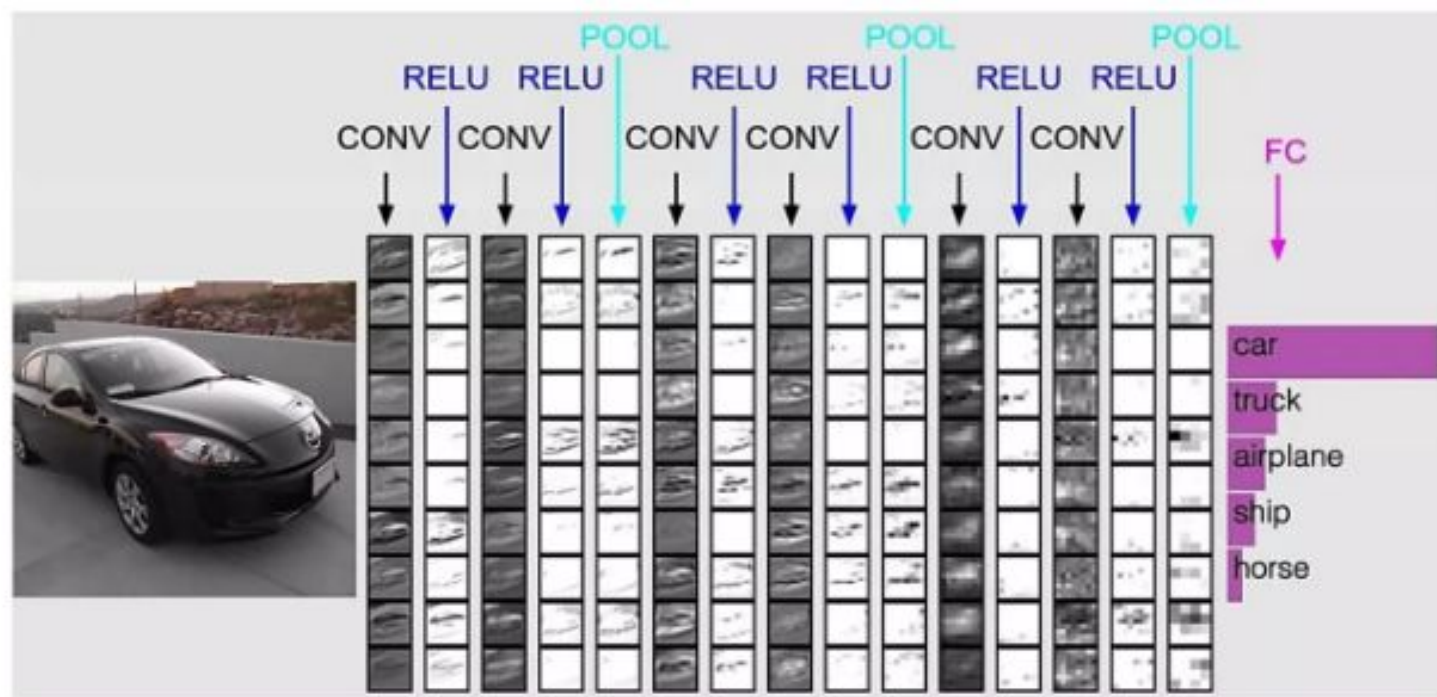


- A medida que se va más adentro en la red:
 - El tamaño tiende a decrecer
 - La profundidad tiende a incrementarse
- Patrón común: Conv-Pool-Conv-Pool-FC-FC-FC-Softmax

Redes Neuronales Convolucionales

Ejemplos de CNN

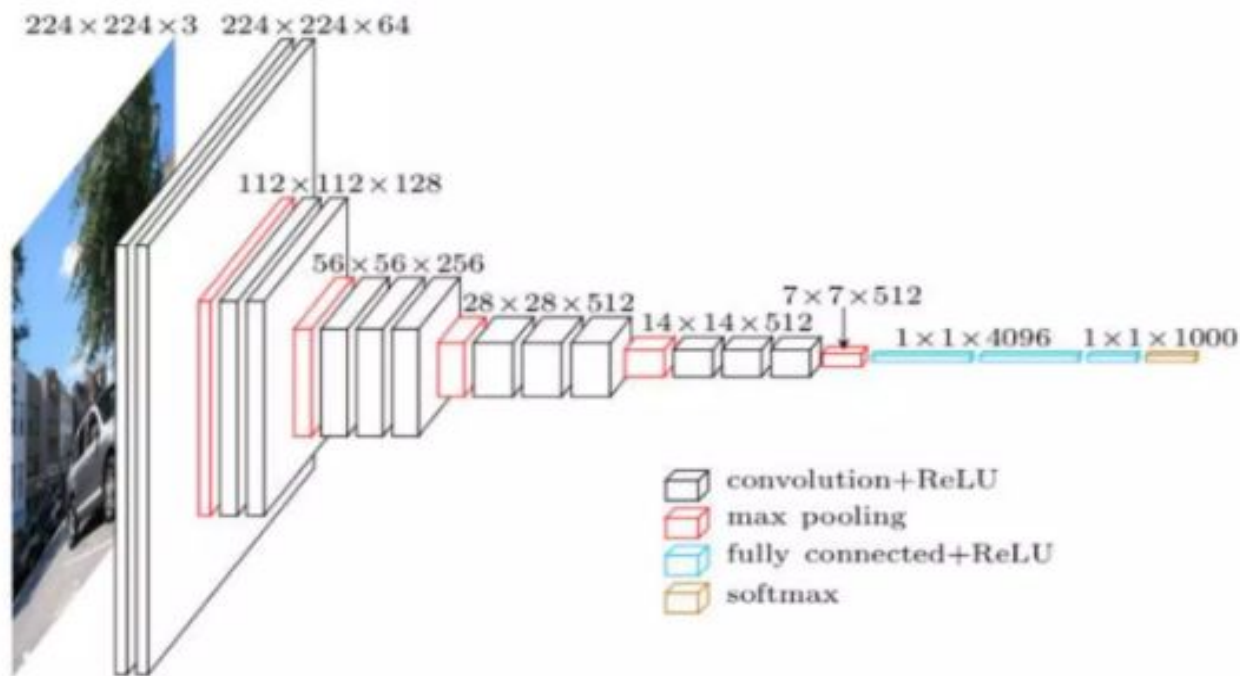
- Efectos de la arquitectura de la red



Redes Neuronales Convolucionales

Ejemplos de CNN

- VGG-16



Contenido

1. Convoluciones
2. Capas de una Red Neuronal Convolutiva
- 3. Aspectos prácticos de CNNs**
4. Detección de Objetos
5. Ejemplos de Detectores de Objetos

Aspectos Prácticos de CNNs

- Inicialización de parámetros (pesos)
 - Depende de la función de activación y puede ayudar en la convergencia
 - Algunas formas:
 - Inicialización aleatoria
 - Inicialización de Xavier
 - Inicialización de Kaiming/MSRA
- Regularización
 - Evita el “overfitting” y permite un mejor entrenamiento
 - Algunas formas:
 - Normalización de batch (“batch normalization”)
 - Aumento de datos (“Data augmentation”)
 - Drop connect
 - Fractional pooling
 - Profundidad estocástica
 - Cutout

Aspectos Prácticos de CNNs

Transferencia del Aprendizaje

- Problema en CNNs:
 - Se requiere muchos datos para entrenar/usar CNNs
 - Requiere mucho tiempo de entrenamiento
- Alternativa
 - Aprovechar parte de alguna red ya entrenada (es común en la práctica)
 - Se conoce como "Transfer Learning"

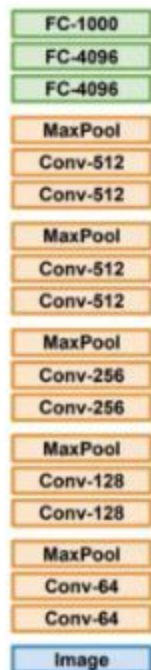
- Idea



Aspectos Prácticos de CNNs

Transferencia del Aprendizaje

Red entrenada en
ImageNet



Conjunto de
entrenamiento pequeño



Conjunto de
entrenamiento más grande



Aspectos Prácticos de CNNs

Cómo Aplicar Transfer Learning

- Ideas para utilizar transfer learning
 - Según la cantidad de datos que se tenga
 - Según la similaridad o diferencia con el sistema pre-entrenado

	Dataset muy similar	Dataset muy diferente
Muy pocos datos	Hacer un "finetune" del clasificador de la capa final	Intentar "data augmentation" o recolectar más datos
Bastantes datos	Hacer un "finetune" de algunas capas	Hacer un "finetune" de un número más grande de capas

Contenido

1. Convoluciones
2. Capas de una Red Neuronal Convolutiva
3. Aspectos prácticos de CNNs
- 4. Detección de Objetos**
5. Ejemplos de Detectores de Objetos

Detección de Objetos

- Detección de objetos: localización de múltiples objetos en una imagen
- Ejemplos
 - Detección de personas para vigilancia con robots aéreos



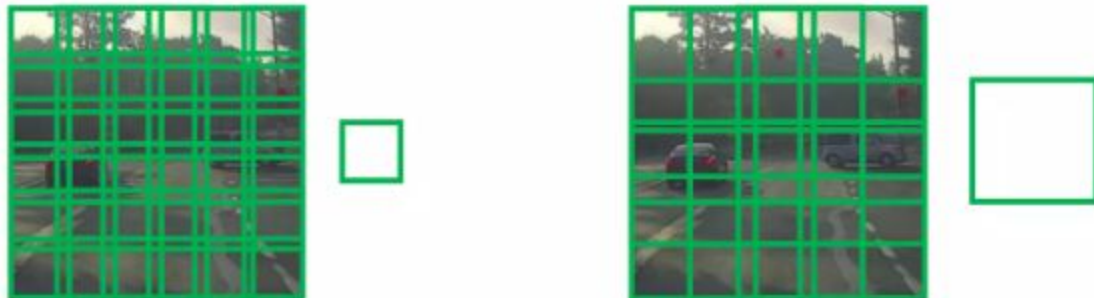
- Vehículos autónomos



Detección de Objetos

Ventanas Deslizantes

- **Idea básica:**
 - Usar ventanas deslizantes por toda la imagen
 - Aplicar CNN (clasificación) a cada ventana

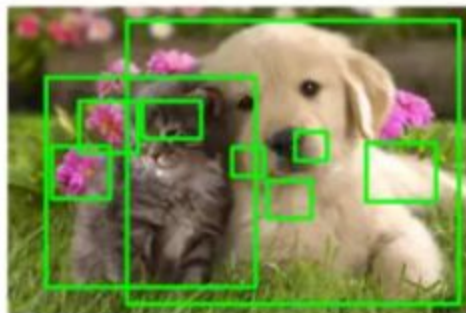


- Requiere recorrer todas las posibles localizaciones
- Las ventanas deslizantes deben ser de diferentes tamaños (todas las escalas posibles)
 - Alternativa: pirámide de imágenes

Detección de Objetos

Regiones candidatas


- En inglés “region proposals”
- Son un conjunto de regiones (cuadros delimitadores) que tienen alta probabilidad de contener objetos
- Se basan en:
 - Búsqueda selectiva (“selective search”)
 - Cuadros de borde (“edge boxes”)

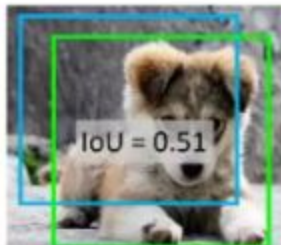


Detección de Objetos

“Intersection over Union” (IoU)

- Es una medida de la superposición entre dos cuadros delimitadores
- También se denomina índice de (similaridad de) “Jaccard”
- Dados dos cuadros delimitadores:

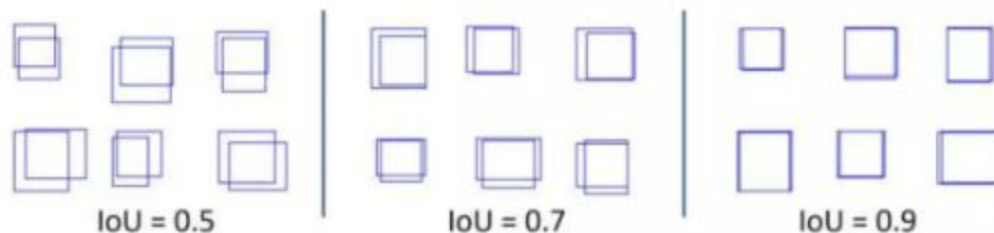
$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de Unión}}$$




Detección de Objetos

“Intersection over Union” (IoU)

- Siempre varía entre 0 y 1
 - IoU > 0.5: “decente”
 - IoU > 0.7: “muy bueno”
 - IoU > 0.9: “casi perfecto”



- Modificaciones de IoU
 - GIoU (generalized IoU)
 - DIoU (distance IoU)
 - CIoU (complete IoU)

Detección de Objetos

Supresión de no máximos (NMS)

- NMS = “Non-Maxima Suppression”
- Usualmente hay muchas detecciones de un mismo objeto



- NMS: Reduce las múltiples detecciones a 1 sola detección

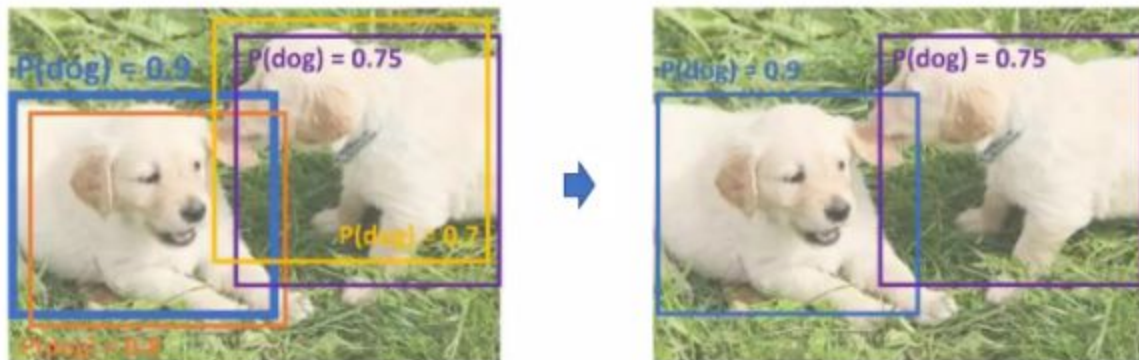
Detección de Objetos

Supresión de no máximos (NMS)

- Procedimiento

1. Seleccionar el cuadro con mayor puntaje (probabilidad de clase)
2. Eliminar cuadros con más bajo puntaje si $\text{IoU} > T$ (ejemplo: $T=0.7$)
3. Si queda algún cuadro, repetir desde 1

- Ejemplo



Detección de Objetos

Tipos de Detectores de Objetos

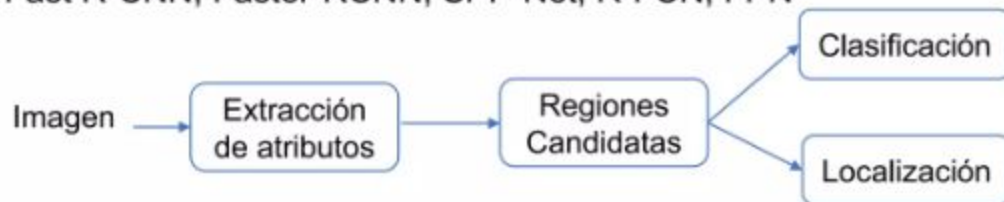
- Detectores de una etapa

- Ejemplos: SSD, YOLO, CornerNet, RetinaNet, CenterNet, ExtremeNet



- Detectores de dos etapas

- Ejemplos: R-CNN, Fast R-CNN, Faster-RCNN, SPP-Net, R-FCN, FPN



Contenido

1. Convoluciones
2. Capas de una Red Neuronal Convolutacional
3. Aspectos prácticos de CNNs
4. Detección de Objetos
- 5. Ejemplos de Detectores de Objetos**

Detectores de Dos Etapas

- Realizan la detección procesando más de una vez cada imagen (en dos etapas)
- Etapas
 - Etapa 1
 - Determinar “regiones candidatas” (*region proposals*) o regiones de interés (ROI) en la imagen
 - Alternativas de selección de regiones:
 - Manualmente: “selective search”
 - Automáticamente: entrenando una CNN
 - Etapa 2
 - Determinar si hay una clase en cada región candidata (clasificación)
- Ejemplos:
 - R-CNN (*Region-based CNN*), SPP-Net, Fast R-CNN, Faster R-CNN, Mask R-CNN, FPN (*Feature Pyramid Network*)

Detectores de Dos Etapas

Regiones Candidatas

- Segmentación y unión de segmentos



- Los segmentos se convierten en "bounding boxes"

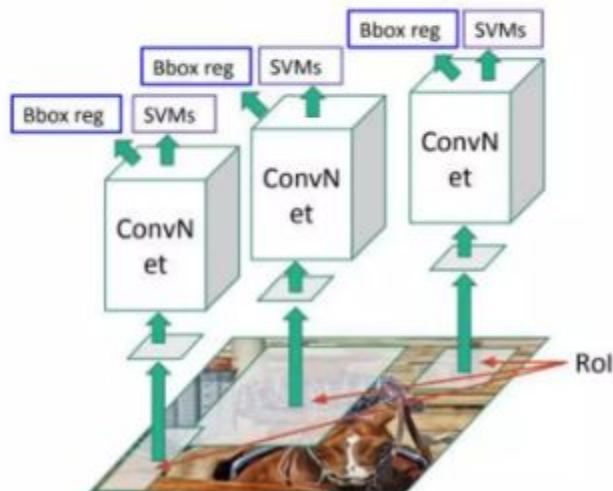


Detectores de Dos Etapas

R-CNN

- Procedimiento

- Se extrae las regiones de interés (aproximadamente 2000)
- Se re-escala ("warp") el tamaño de las regiones (224 x 224)
- Se extrae atributos de cada región por separado (usando Capas Convolucionales)
- Se clasifica cada región usando SVMs y se predice los bordes ("boxes")

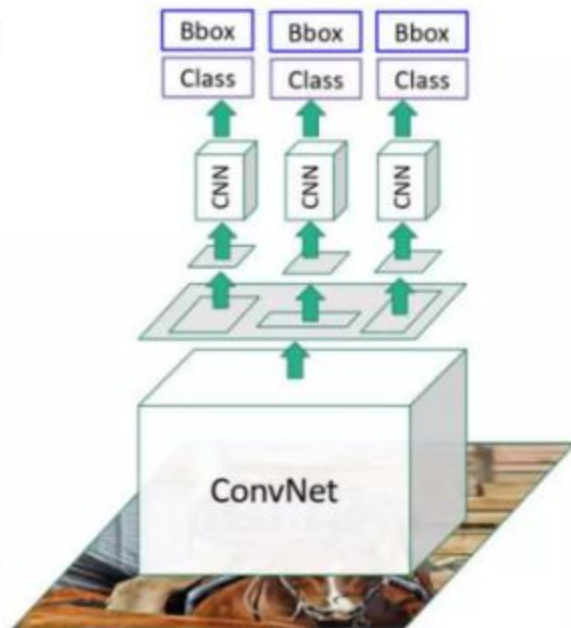


Girshick et al. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

Detectores de Dos Etapas

Fast R-CNN

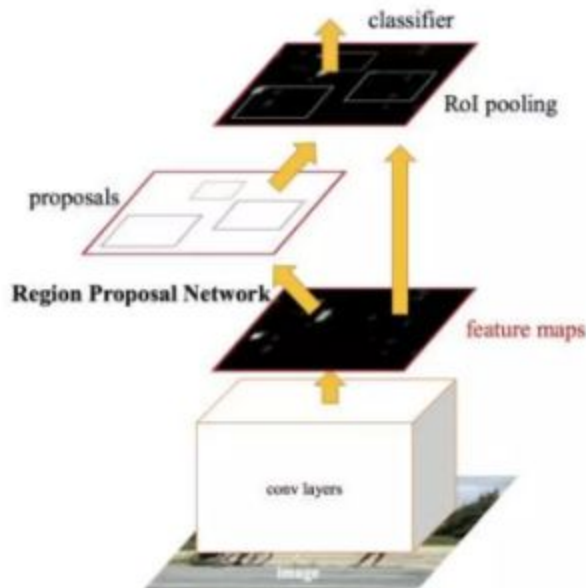
- Es más rápida que RCNN: extrae atributos (capa convolucional) para toda la imagen (antes de crear regiones)
- Etapas
 - Generación de atributos para toda la imagen (Capa convolucional ConvNet): "backbone" AlexNet, VGG, ResNet, etc.
 - Generar regiones de interés (usando método de regiones candidatas)
 - Re dimensionar Rol
 - Aplicar CNN a cada región para generar la clase y el cuadro delimitador



Detectores de Dos Etapas

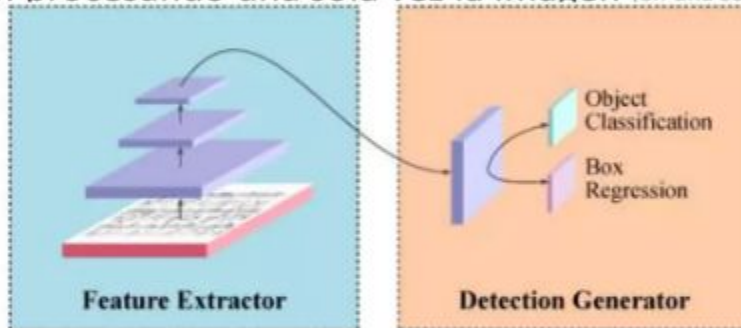
Faster R-CNN

- Inserta una red neuronal (llamada RPN = Región Proposal Network) para predecir regiones de interés
 - Es más rápido que "selective search"



Detectores de una Etapa

- Realizan la detección procesando una sola vez la imagen (en una sola etapa)

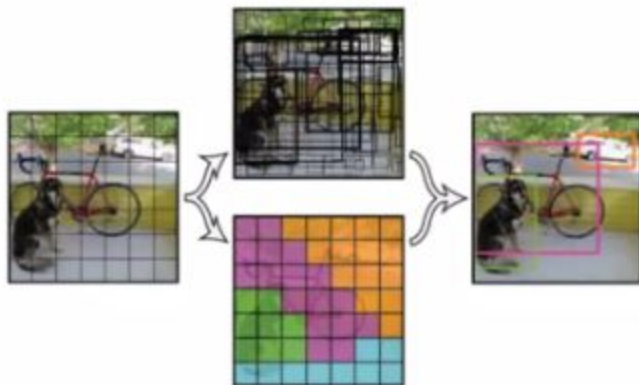


- Predicen la clase y la región ("bounding box") en toda la imagen
 - Funcionan en "tiempo real" (varios frames por segundo)
 - Ejemplos: YOLO (You Only Look Once), RetinaNet, SSD (Single Shot Multibox Detector), CornerNet, ExtremeNet

Detectores de una Etapa

YOLO

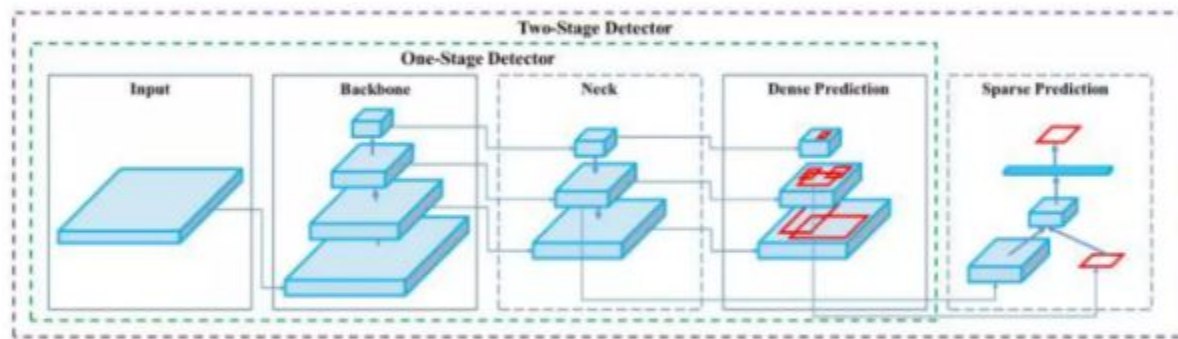
- YOLO: "You Only Look Once" (una sola pasada por la imagen: "only once")
 - Es mucho más rápido que los modelos R-CNN (útil para tiempo real)
 - Puede ser menos exacto que modelos R-CNN (sacrifica exactitud por velocidad)
- **Funcionamiento**
 - Divide la imagen en celdas: cada una predice el cuadro delimitador (si contiene el centro)
 - El mapa de probabilidades y "bounding boxes" luego se combina



Detectores de una Etapa

YOLO v4

- Arquitectura



Input: { Image, Patches, Image Pyramid, ... }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [88], Darknet53 [63], ... }

Neck: { FPN [44], PANet [49], Bi-FPN [77], ... }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], ... }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], ... }