

07MIAR – Redes neuronales y deep learning



Universidad
Internacional
de Valencia

Tareas avanzadas de *computer vision*
empleando aprendizaje profundo

01

Introducción

Tareas avanzadas de *computer vision* empleando aprendizaje profundo

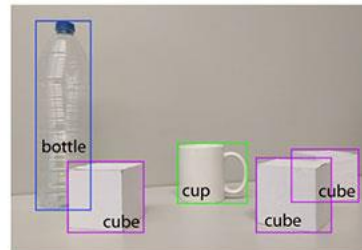
25/10/2021

Tareas avanzadas CV

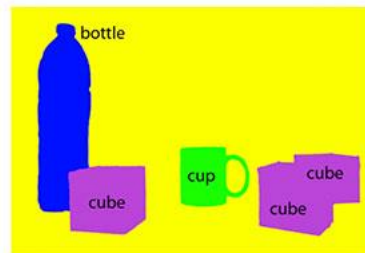
- Dentro del campo de la **visión por computador** existen diversas **tareas de interés** que tienen como denominador común el **tratamiento con los objetos** de una **escena** tanto **estática** como en **movimiento**.



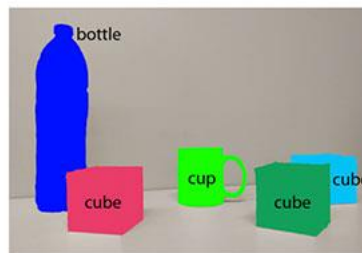
Clasificación de imagen



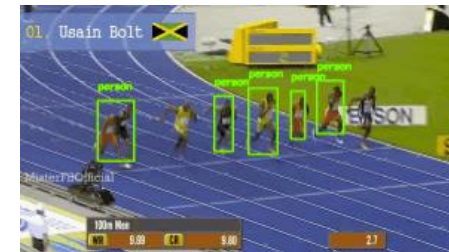
Localización de objetos



Segmentación
semántica



Segmentación de
instancia



Tracking o localización dinámica



Generación sintética de imágenes, *denoising*,
compresión, detección de anomalías, etc.



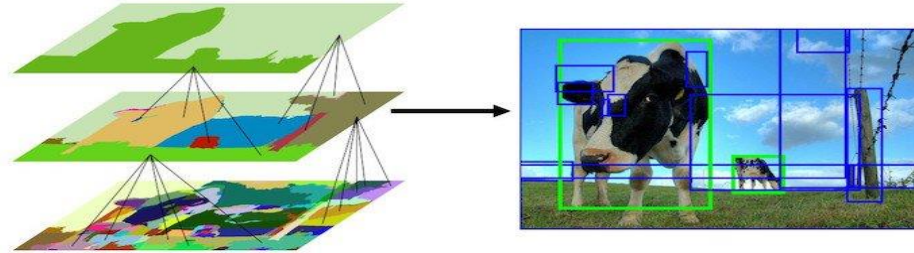
02

Detección de objetos

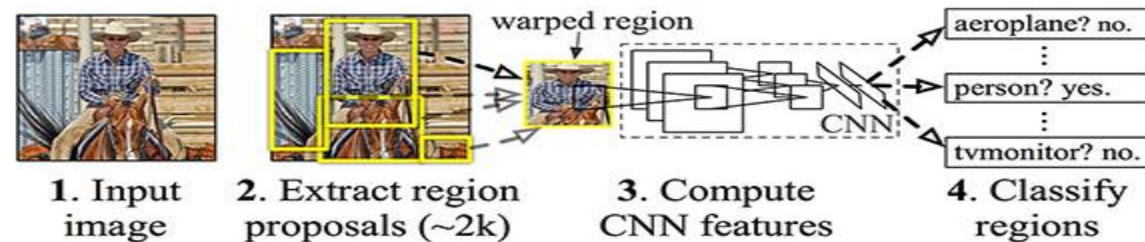
Tareas avanzadas de *computer vision* empleando aprendizaje profundo

R-CNN: Regiones con características CNN

- Método propuesto por **Ross Girshick** en **2013**. Se compone de los siguientes pasos:
 1. **Extracción de regiones candidatas** mediante el **algoritmo** de búsqueda selectiva. Agrupación jerárquica de regiones similares basada en color, textura, tamaño y forma.

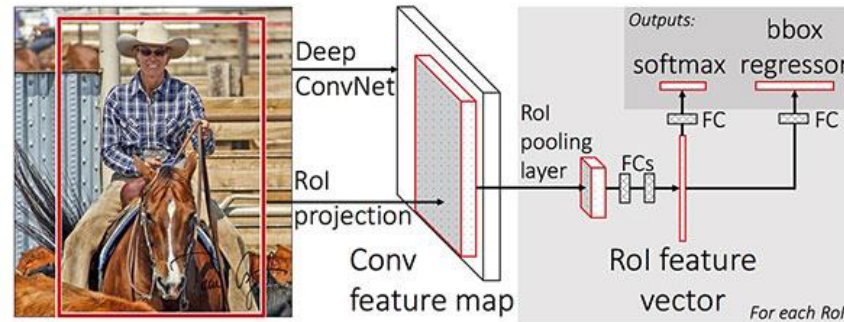
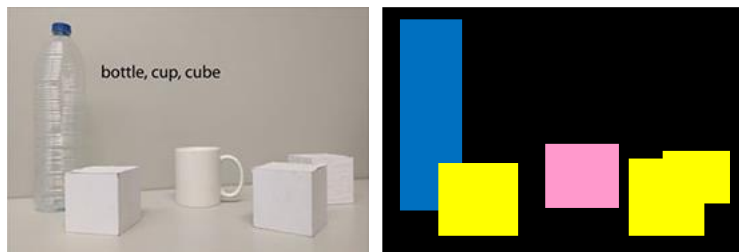


2. **Uso** de técnicas de **transfer learning** (extracción de características) sobre las regiones candidatas empleando una **arquitectura pre-entrenada (AlexNet)**.
3. **Clasificación** de cada región candidata empleando las **características extraídas** y Support Vector Machine (**SVM**).



Fast R-CNN: Regiones con características CNN

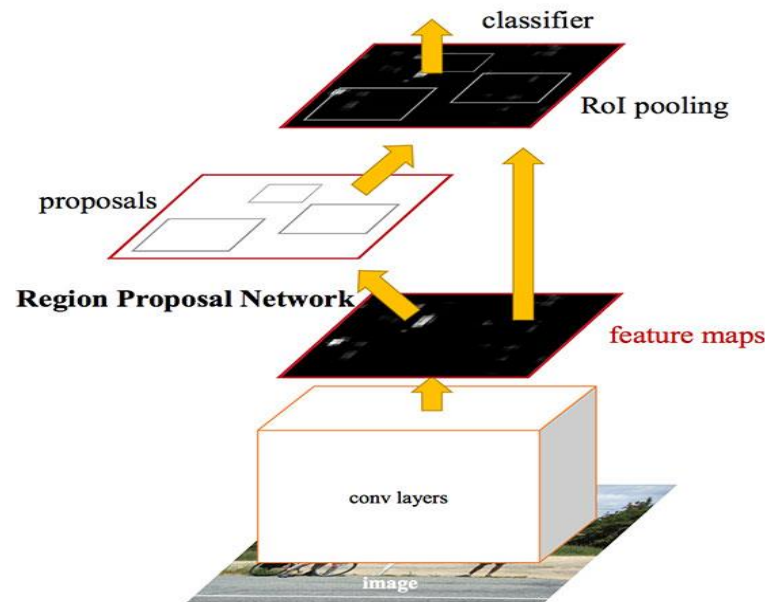
- Gran **desventaja R-CNN**: Tiene que **clasificar todas las regiones candidatas** que se extraen del método de búsqueda selectiva (i.e. **cuello de botella**).
- Girshick et al. (2015) proponen una **mejora de R-CNN** que trata de disminuir el alto coste computacional de dicho método. Proponen un **módulo de pooling sobre el mapa de activación** que obtienen al pasar la imagen por la red pre-entrenada (**proyecta ROIs**).
- El proceso de **clasificación** pasan de hacerlo con SVM a emplear un **perceptrón multicapa** que **predice la clase y la bounding box (offset)**. Consiguen un **aproximación entrenable end to end**.



- El **performance** del proceso de **inferencia** (i.e. fase de predicción) sigue sufriendo dramáticamente debido a la **dependencia** en el **algoritmo** de **búsqueda selectiva**.

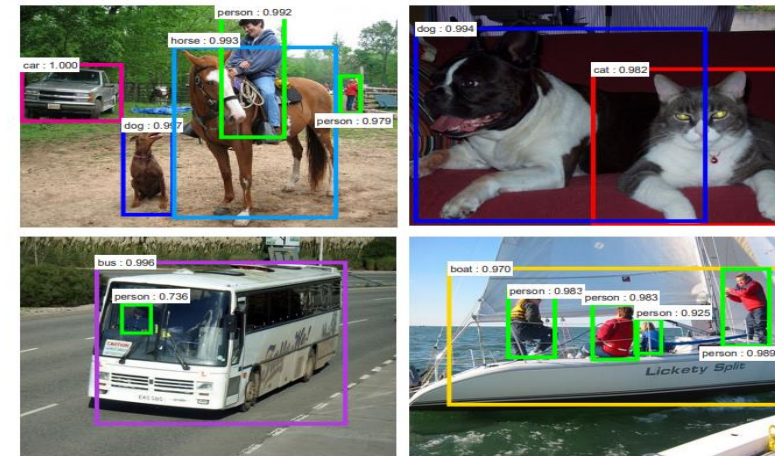
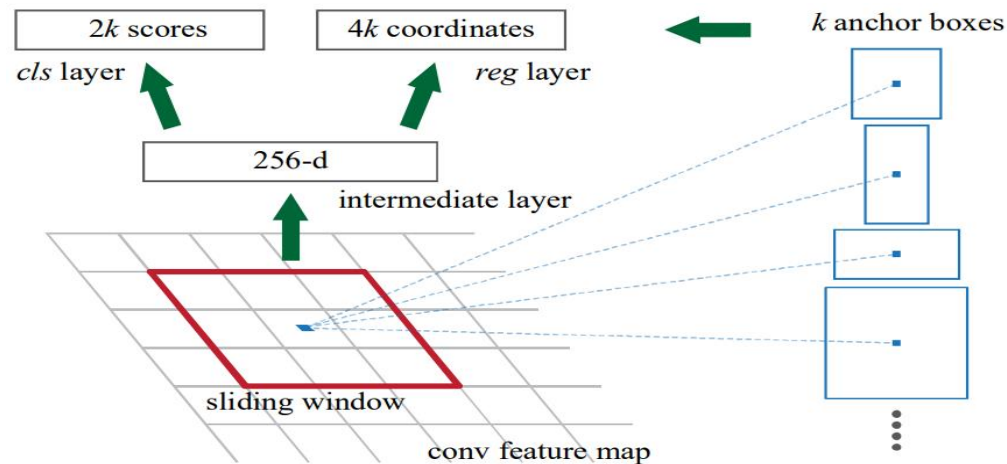
Faster R-CNN: Regiones con características CNN

- En el artículo donde Girshick et al. (2015) proponen **Faster R-CNN**, presentan la **Region Proposal Network (RPN)** que introduce la propuesta de regiones directamente en la arquitectura (sustituyendo el algoritmo de **búsqueda selectiva**).
- Supone una contribución muy importante porque la **Faster R-CNN** es capaz de **predecir 7-10 FPS** haciendo posible la detección de objetos en **tiempo real**.



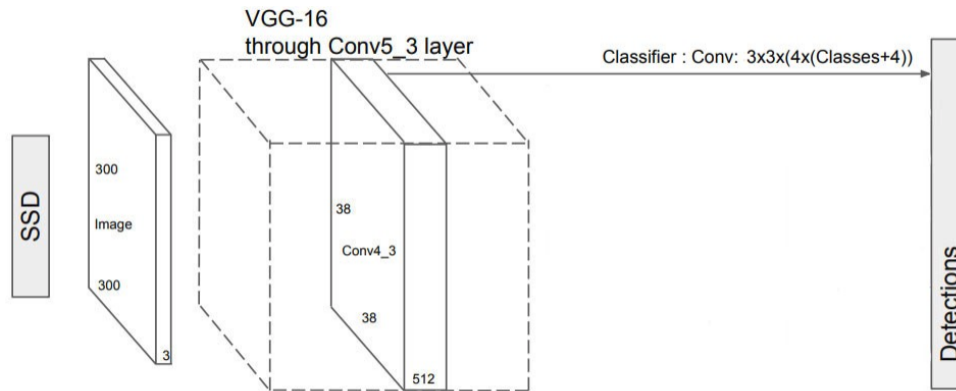
Faster R-CNN: Regiones con características CNN

- Sobre el mapa de activación se pasa una ventana deslizante y se generan k candidatos o *anchors*. Dichos candidatos se evalúan simultáneamente en la RPN.
- La RPN es un perceptrón multicapa con dos *fully-connected* de salida. Una de ellas se encarga de dar una predicción de si la región contiene o no contiene objeto mientras que la otra se encarga de predecir los bordes de la región.
- En el artículo emplean 3 escalas y 3 relaciones de aspecto para generar $k=9$ *anchors* a partir de una ventana deslizante de tamaño 3×3 .



SSD: Single Shot Multibox Detector

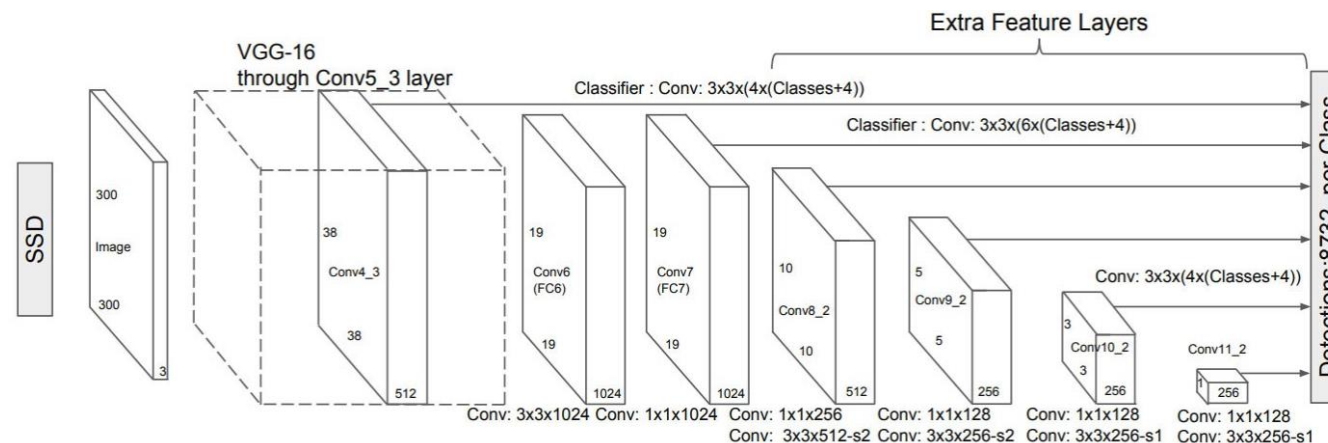
- Método propuesto por **Wei Liu** en **2016**. **Elimina** la necesidad de una **red generadora de candidatos**. La red se compone de dos niveles:
 - **Extracción de mapas de activación** (características) mediante red pre-entrenada (VGG16)
 - Aplicar **filtros convolucionales** (3x3) **sobre el mapa** de activación para detectar objetos.
- **Para cada** una de las **38 x 38 celdas** o localizaciones se obtienen **4 predicciones de objeto (4 filtros)** siguiendo una serie de ***bounding box* predefinidas (a conciencia)**.



- Cada predicción está compuesta por una *bounding box* (offset) y **21 scores** de pertenencia a clase (**20 clases + fondo**).

SSD: Single Shot Multibox Detector

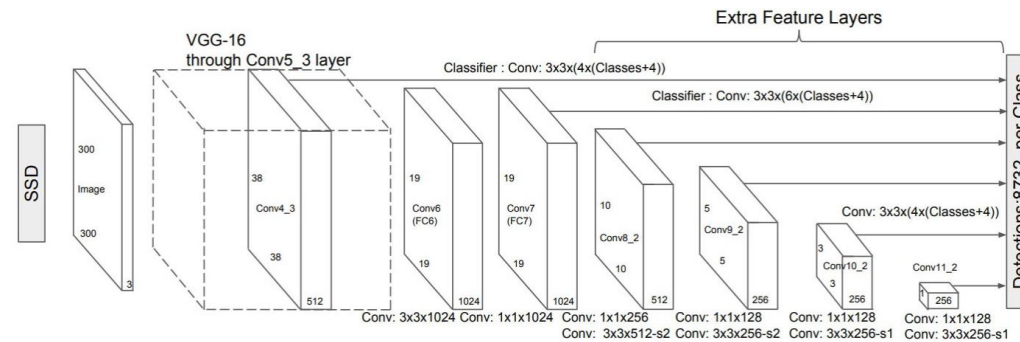
- La gran **potencia** de la red **SSD** reside en el uso de **múltiples capas** con el objetivo de realizar una **detección multiescala**.
- Concretamente **SSD añade 6 capas convolucionales** sobre el mapa de activación que obtiene de la red pre-entrenada y **sobre cinco** de ellas aplica la **detección de objetos** tal y como hemos visto anteriormente (**en 3 de ellas** hace **6 predicciones** por celda en vez de 4). Un total de **8732 predicciones**.



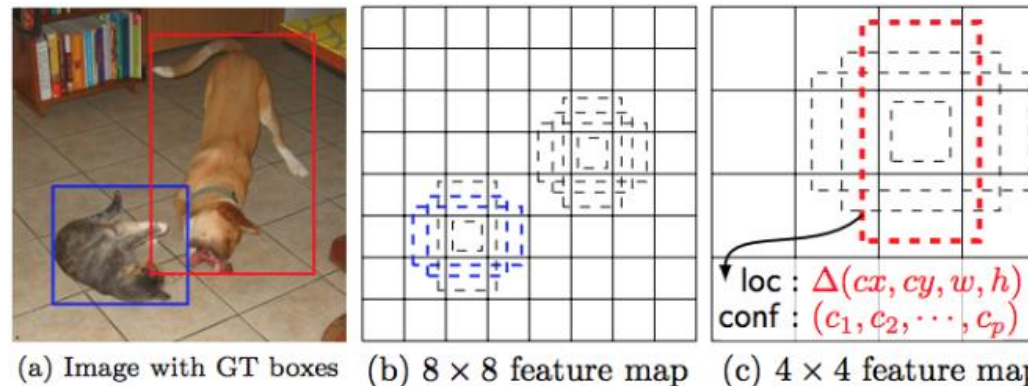
- Estas **capas convolucionales** van disminuyendo las dimensiones del mapa de características gradualmente (*stride* \neq 1).



SSD: Single Shot Multibox Detector

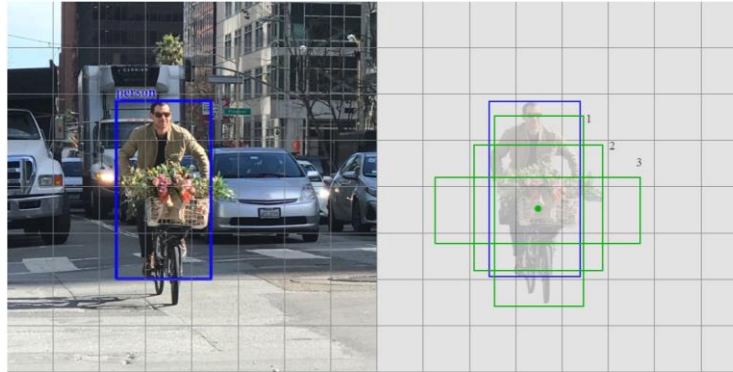


- SSD usa las **resoluciones bajas** para detectar **objetos grandes** mientras que los **objetos pequeños** son detectados en los **primeros mapas** de activación.



SSD: Single Shot Multibox Detector

- En la **fase de entrenamiento**, el coste para **optimizar la localización** solo se calcula sobre las coincidencias positivas. Una coincidencia positiva se define por tener una **IoU > 0.5** entre la **BB por defecto** y la **del GT**.



- Una vez identificadas las **coincidencias positivas**, estas se emplean para **calcular el error** con respecto a la **BB predecida**.
- En la **fase de test**, SSD utiliza la técnica **non-máximum supression** (algoritmo basado en el **nivel de confianza** y en la **IoU entre predicciones**) para **eliminar predicciones duplicadas** sobre un mismo objeto.



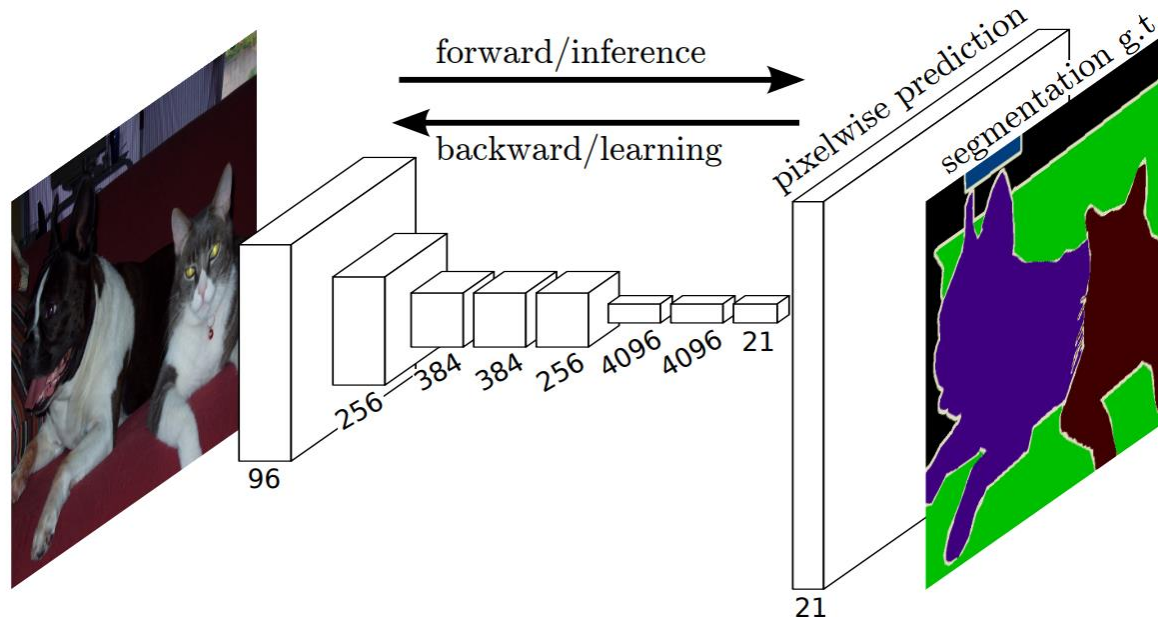
03

Segmentación semántica y de instancia

Tareas avanzadas de *computer vision* empleando aprendizaje profundo

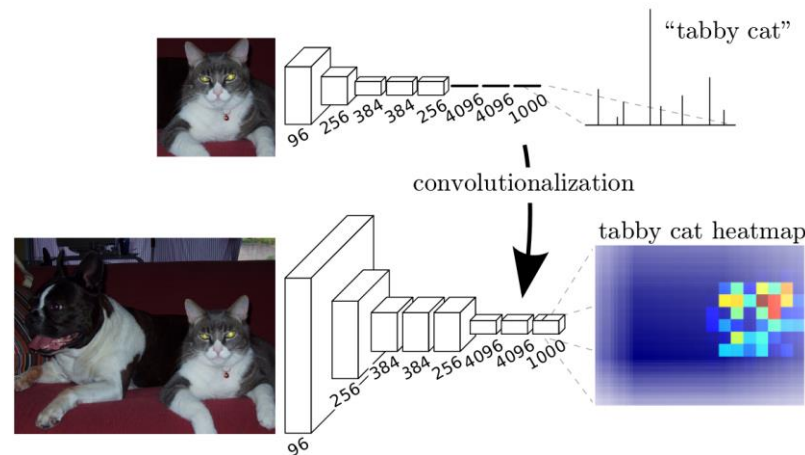
FCNN para la segmentación semántica

- J. Long et al. (2015) fueron los primeros en utilizar una **arquitectura** basada exclusivamente en **capas convolucionales** y de **pooling** para realizar tareas de **segmentación semántica**.
- Este tipo de red recibe como **entrada** una **imagen** de un **determinado tamaño** y como **salida** devuelve la **imagen segmentada** del **mismo tamaño**.



FCNN para la segmentación semántica

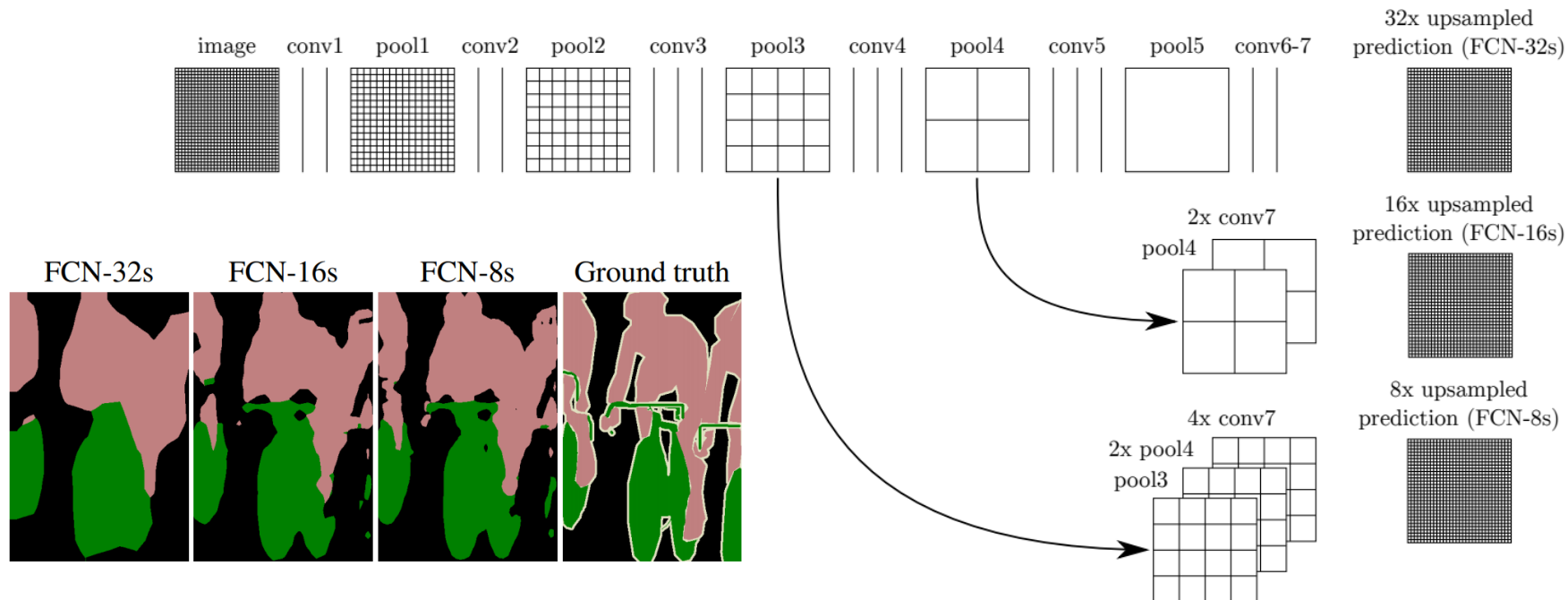
- Los autores **modifican** redes conocidas como **AlexNet**, **VGG16** o **GoogleNet** eliminando el **top model** (destinado a clasificación) y **reemplazándolo** por más **bloques convolucionales** produciendo **pequeños mapas** de características con representaciones **densas**.



- Cuando se llega al final de la red, **al último mapa** se le debe aplicar un **upsampling** para llevarlo a las **dimensiones espaciales originales** de la imagen.
- Se emplea una **capa convolucional** con un valor de **stride** = $1/f$ consiguiendo **ampliar** el mapa de activación **por un factor f** . Esta técnica es conocida como **deconvolución**. Los **filtros** aprendidos en estas capas constituyen las **bases** para **reconstruir la forma**.

FCNN para la segmentación semántica

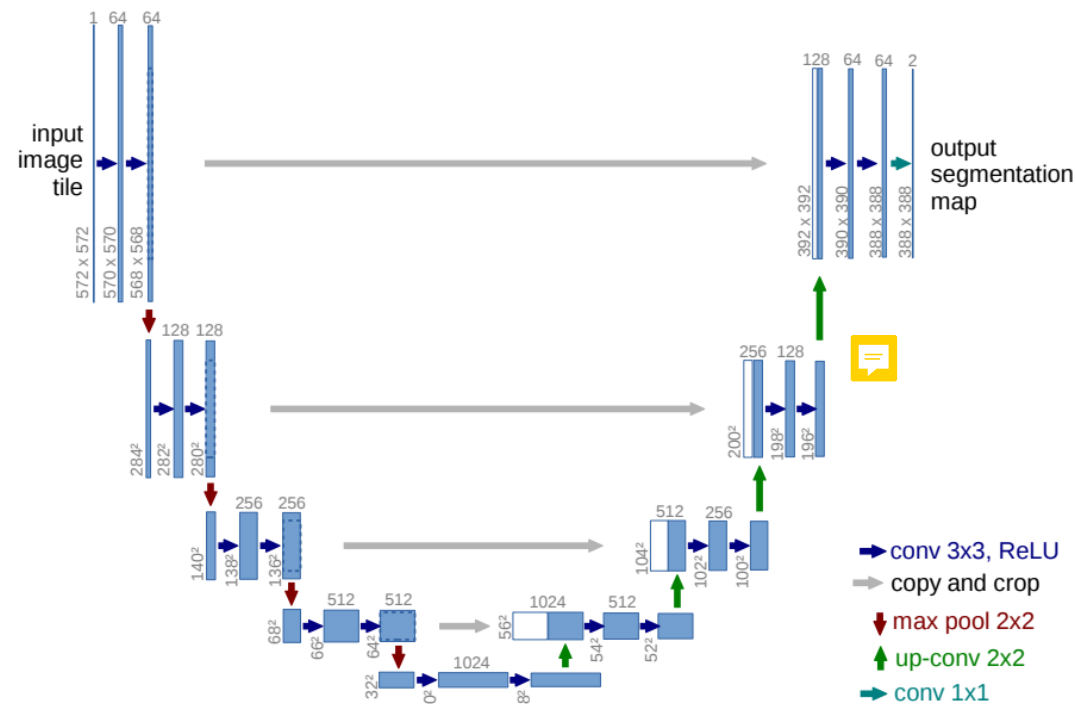
- La **red** es **entrenada** empleando una **función de pérdidas** a nivel de **pixel** (por ejemplo **Dice**).
- Además, los autores introducen ***skip connections*** en la red con el objetivo de **combinar mapas de características** que contienen **representaciones de alto y bajo nivel** que posteriormente fusionan dando lugar a la segmentación final.




Segmentación semántica: U-net

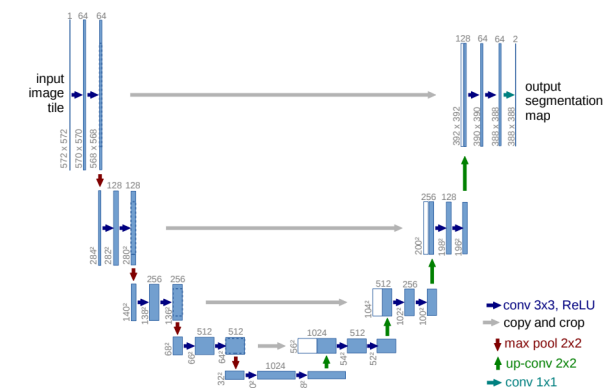


- **O. Ronneberger et al. (2015)** extienden el trabajo anterior a imágenes de microscopía. Los autores crean **U-net** que se compone de **dos partes**: el **contracting path** encargado de codificar la **información relevante** de la imagen y el **expansive path** que localiza **espacialmente** los patrones relevantes en la imagen.



Segmentación semántica: U-net

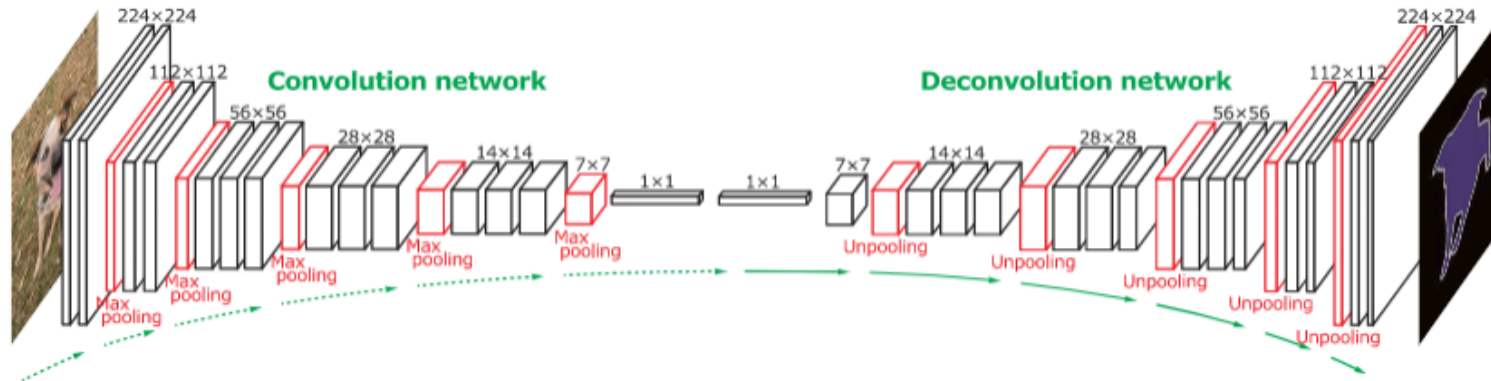
- El **downsampling** o **contracting path** esta **basado** en una **FCN** con **tamaño de filtros constante** (i.e. **3x3**).
- El **upsampling** o **expansive path** emplea **up-convolutions** (i.e. deconvoluciones, convoluciones transpuestas) reduciendo el número de mapas de características a la vez que aumenta el tamaño espacial. 
- Una **importante contribución** es que para **evitar pérdida de información**, **copian** los **mapas** de cada **bloque** convolucional del **encoder** en su correspondiente **mapa en el decoder**. Esta **información se concatena** en la **tercera dimensión** gracias a las **skip connections**.
- **Por último, una capa convolucional 1x1 mapea las dimensiones del último mapa de activación a tantos mapas 2D como clases se tengan.** Posteriormente, aplicando la función **softmax** se **categoriza** cada uno de los **pixels**, obteniendo la segmentación semántica.



Segmentación de instancia: Redes Conv-Deconv



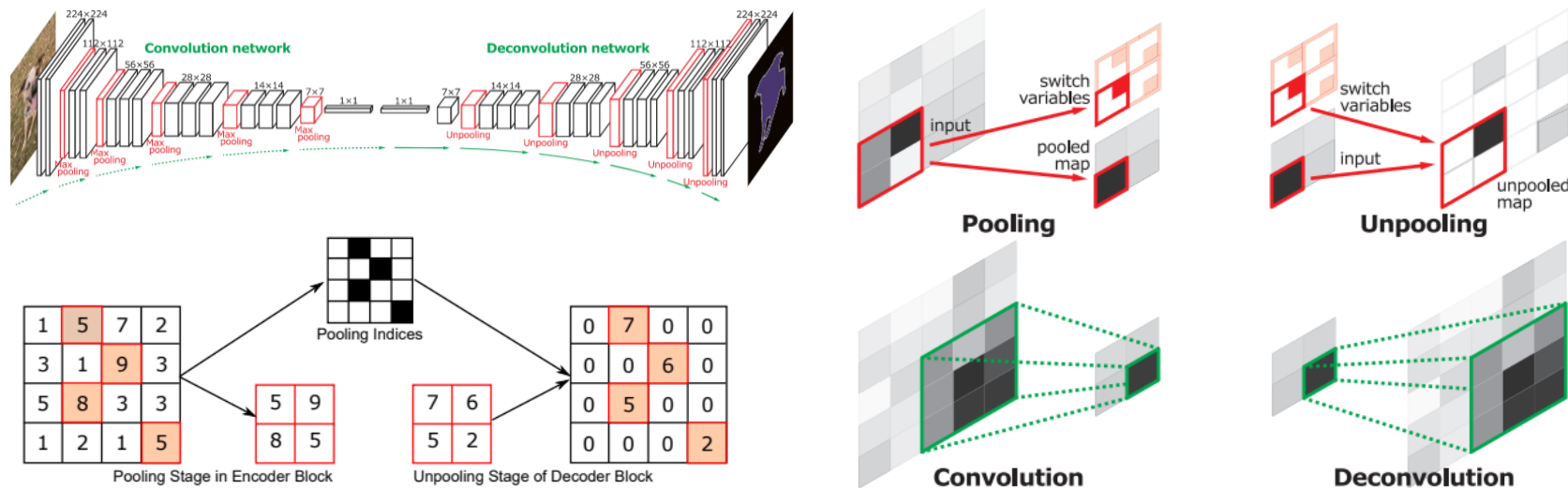
- **H. Noh et al. (2015)** diseñan una **red *end to end*** compuesta por dos niveles para realizar segmentación de instancia. La primera de ellas es un módulo detector de objetos mediante VGG16.
- La **red** que proponen para la **segmentación** se compone de **dos fases**, la de **convolución** y la **deconvolución**.



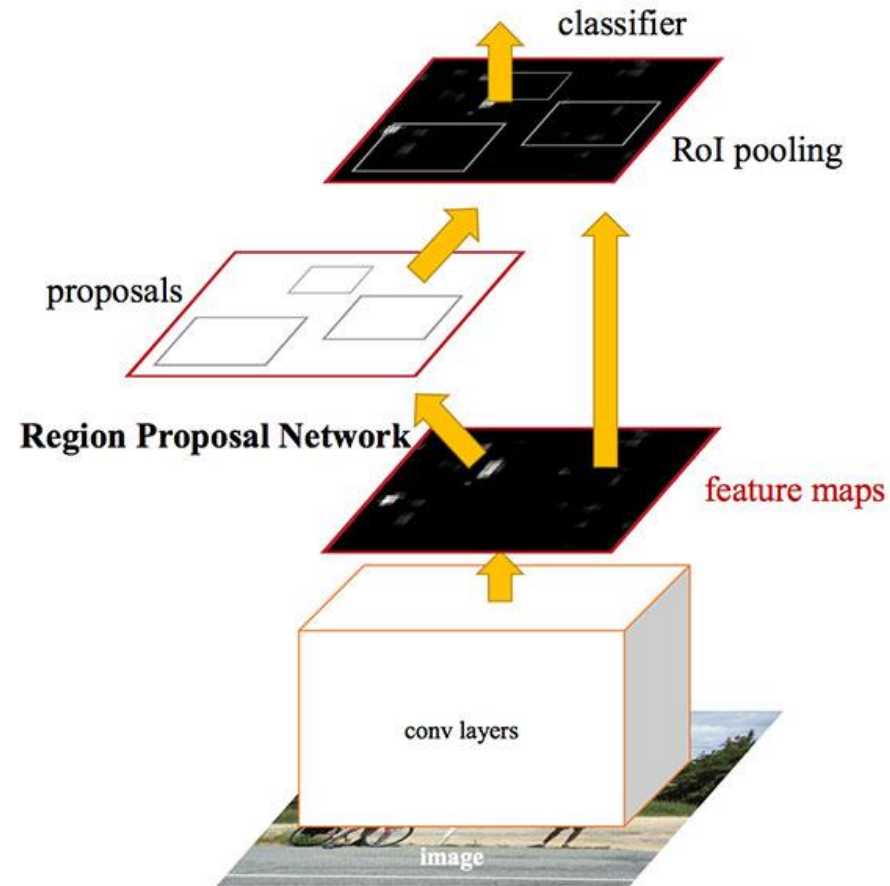
- Una **región candidata** es **procesada por el *encoder*** generando un **vector de características**.

Segmentación de instancia: Redes Conv-Deconv

- El **decoder** toma dicho vector de características y genera un mapa a nivel de pixel con la probabilidad de pertenencia a clase.
- La **subred de deconvolución** emplea la operación **unpooling** localizando las activaciones máximas para mantener su posición original en el mapa de características que va ampliando. Tras la **capa unpooling** se insertan varias **capas deconvolucionales** para mantener la densidad de la información.



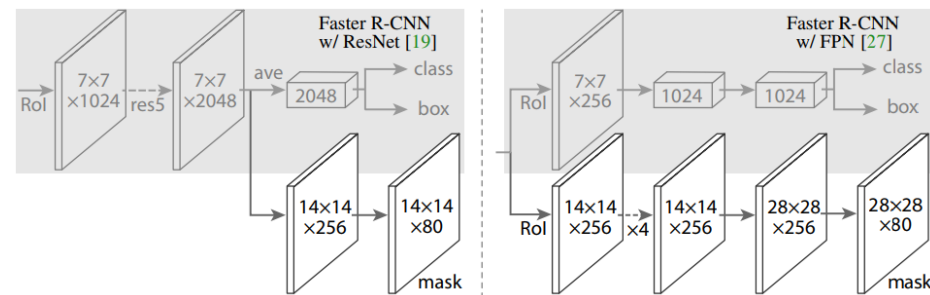
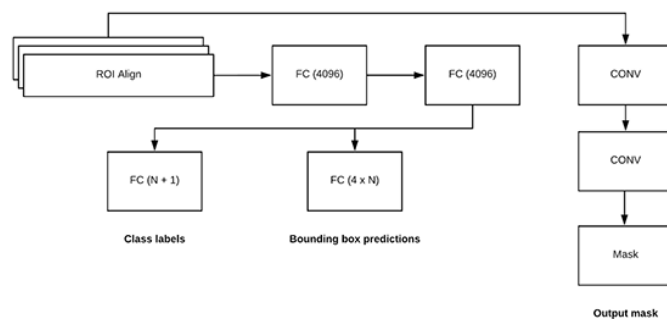
Faster R-CNN: Regiones con características CNN



Mask R-CNN para la segmentación de instancia

- La **Mask R-CNN** es una arquitectura para la **segmentación de instancia** fue propuesta por **He et al. (2018)** y nace a partir de la Faster R-CNN:
 - Se reemplaza el módulo **ROI pooling** por el módulo **ROI align** mucho más preciso para el propósito de **segmentación**.
 - Se inserta una **rama adicional** a la salida del nuevo módulo que realiza la **segmentación**.

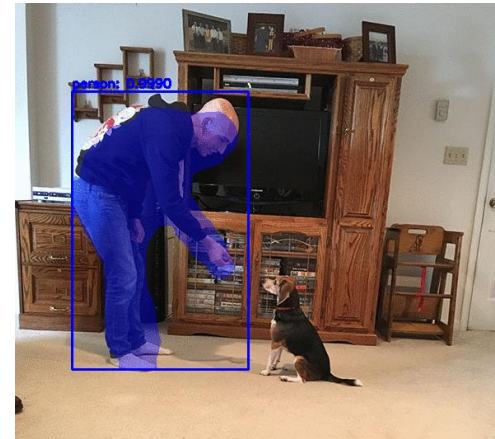
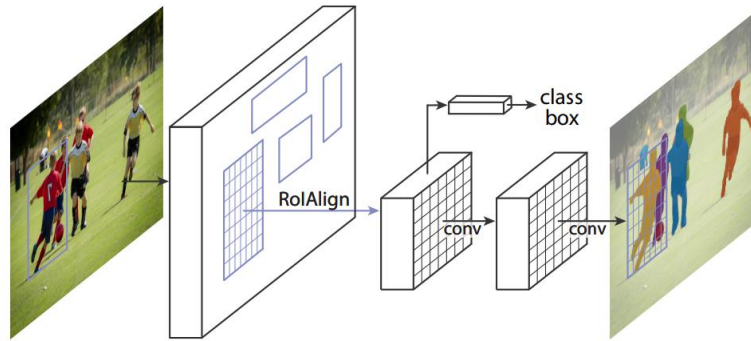
Mask R-CNN



- La **salida** de la **rama convolucional** es la **máscara** de la segmentación del **objeto** previamente detectado (objeto vs fondo), se obtiene aplicando la función **sigmoide** a **cada pixel** del **último mapa** de activación.

Mask R-CNN para la segmentación de instancia

- La **Mask R-CNN** es una arquitectura para la **segmentación de instancia** fue propuesta por **He et al. (2018)** y nace a partir de la Faster R-CNN:
 - Se reemplaza el módulo **ROI pooling** por el módulo **ROI align** mucho más preciso para el propósito de **segmentación**.
 - Se inserta una **rama adicional** a la salida del nuevo módulo que realiza la **segmentación**.



- La **salida** de la **rama convolucional** es la **máscara** de la segmentación del **objeto** previamente detectado (objeto vs fondo), se obtiene aplicando la función **sigmoide** a **cada pixel** del **último mapa** de activación.



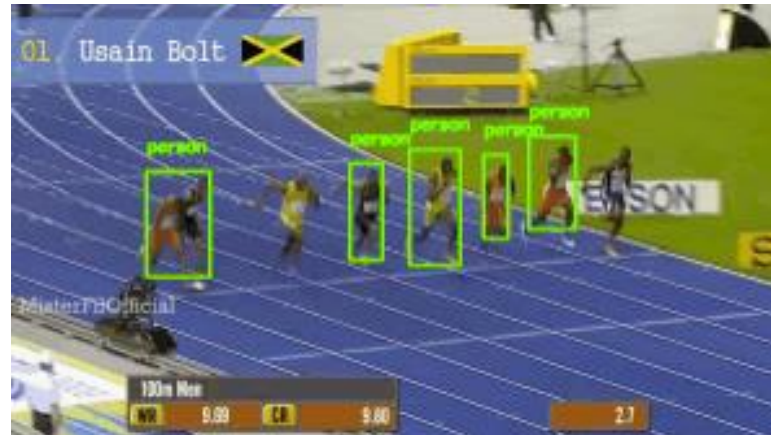
04

Tracking de objetos

Tareas avanzadas de *computer vision* empleando aprendizaje profundo

Introducción

- La **tarea** de ***tracking*** se basa en aplicar la técnica de **detección de objetos** (DO) durante una serie de ***frames*** consecutivos. La problemática reside en el **tiempo de procesamiento** del algoritmo de **DO** para realizar dicha tarea en **tiempo real**.



Faster R-CNN vs SSD

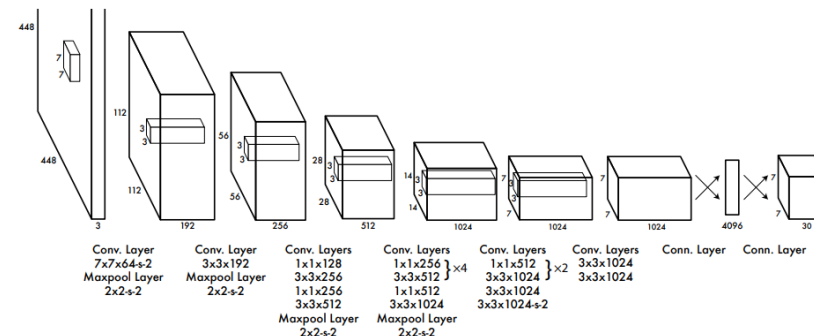
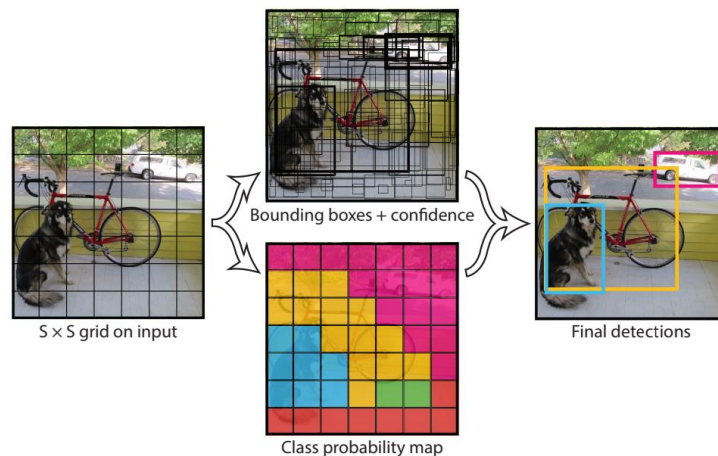
- Si rescatamos las dos metodologías basadas en aprendizaje profundo más comunes de la literatura (i.e. **Faster R-CNN** y **SSD** en sus múltiples versiones) y las **analizamos** desde el punto de vista tanto de **precisión** como de *frames* por segundo (**FPS**) procesados se puede observar que **Faster R-CNN** es un **método** de **detección** de **objetos estático** o que únicamente se podría aplicar a tracking en **videos time-lapse**.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

- Compromiso** entre **precisión** en la detección y **FPS** procesados. Por este motivo nace YOLO (You Only Look Once).

YOLO: You Only Look Once

- Introducida por **Redmon et al. (2015)** se basa en una **estrategia *single-stage*** (i.e. una única red para todo el proceso, al igual que SSD).
- **División** de la **imagen** en un **grid $S \times S$** . Para cada celda **se predicen B bounding boxes** y un **nivel de confianza** (i.e. si hay o no objeto y que objeto de todas las clases es).
- **CNN para extraer las características**, las predicciones se realizan mediante **dos FC layers** después del último bloque convolucional.



YOLO: You Only Look Once

- Los autores afirman que **YOLO** realiza una detección de objetos *super real-time* obteniendo **45 FPS** en una GPU. Desarrollan también una **variante** más **ligera** que corre a **155 FPS**.
- Existen YOLOv2 (2016) y YOLOv3 (2018) que **mejoran la precisión** en la detección de la primera versión ya que se entrenan en **datasets más grandes** (i.e. **COCO**).

Real-Time Detectors		Train	mAP	FPS
100Hz DPM [31]		2007	16.0	100
30Hz DPM [31]		2007	26.1	30
Fast YOLO		2007+2012	52.7	155
YOLO		2007+2012	63.4	45
Less Than Real-Time				
Fastest DPM [38]		2007	30.4	15
R-CNN Minus R [20]		2007	53.5	6
Fast R-CNN [14]		2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]		2007+2012	73.2	7
Faster R-CNN ZF [28]		2007+2012	62.1	18
YOLO VGG-16		2007+2012	66.4	21

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	171
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2 / 2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2 / 2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2 / 2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2 / 2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2 / 2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

DarkNet-19 en YOLOv2

Type	Filters	Size	Output
Convolutional	32	3 × 3	256 × 256
Convolutional	64	3 × 3 / 2	128 × 128
Convolutional	32	1 × 1	128 × 128
Convolutional	64	3 × 3	
Residual			128 × 128
Convolutional	128	3 × 3 / 2	64 × 64
Convolutional	64	1 × 1	64 × 64
Convolutional	128	3 × 3	
Residual			64 × 64
Convolutional	256	3 × 3 / 2	32 × 32
Convolutional	128	1 × 1	32 × 32
Convolutional	256	3 × 3	
Residual			32 × 32
Convolutional	512	3 × 3 / 2	16 × 16
Convolutional	256	1 × 1	16 × 16
Convolutional	512	3 × 3	
Residual			16 × 16
Convolutional	1024	3 × 3 / 2	8 × 8
Convolutional	512	1 × 1	8 × 8
Convolutional	1024	3 × 3	
Residual			8 × 8
Avgpool		Global	1000
Connected		1000	
Softmax			

DarkNet-53 en YOLOv3



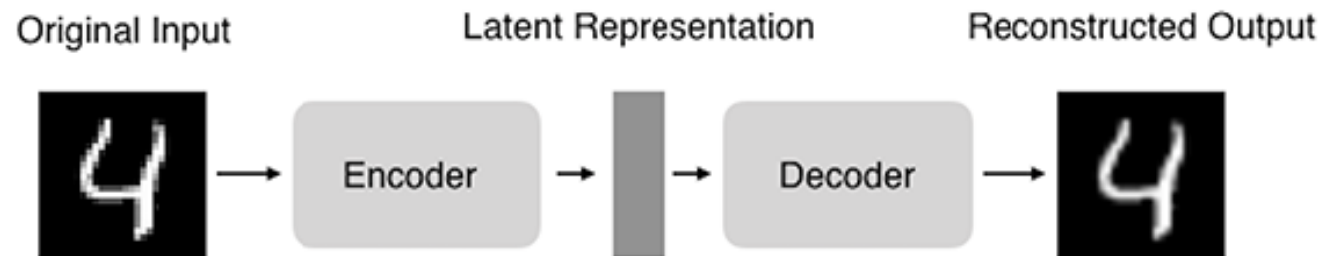
05

Otras tareas

Tareas avanzadas de *computer vision* empleando aprendizaje profundo

Autoencoders

- Los **autoencoders** son un tipo de red neuronal no supervisada que tienen como objetivo comprimir los datos de entrada en una representación denominada **espacio latente**.
- El **espacio latente** se caracteriza por una **dimensionalidad mucho menor** que la dimensionalidad de los **datos de entrada**.
- La idea de un autoencoder es **reconstruir los datos** a partir del espacio latente entrenado minimizando cierta función de error (MSE, MAE, etc.).



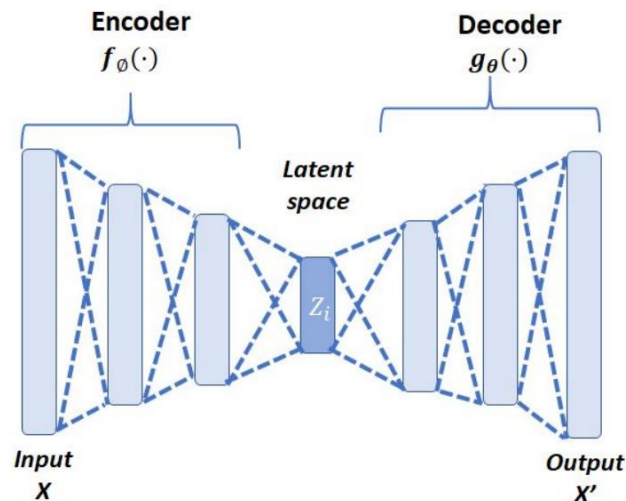
Aplicaciones autoencoders

- **Reducción de dimensionalidad:** Mapeo **no lineal** de los datos de entrada a un nuevo espacio vectorial que describe gran parte de la variabilidad del set de datos de entrada.
- **Denoising:** Eliminar ruido de un set de imágenes a la entrada. El **encoder** caracteriza la **distribución del ruido** y el **decoder** es capaz de generar **muestras sin dicho** ruido gracias al espacio latente generado.
- **Compresión de datos:** Generar nuevas representaciones de datos reducidas a partir del espacio latente.
- **Detección de anomalías/outliers:** Detectar **datos clasificados erróneamente** o detectar cuando un **dato** a la entrada **no sigue** la **distribución** típica de la población.
- **Sistemas Content Based Image Retrieval (CBIR):** Creación de sistemas para la recuperación automática de información. Basado en similitudes entre una **Query** y un **diccionario de representaciones**.
- **Natural Language Processing:** Comprensión de texto, construcción de Word embeddings, o resumen de textos.

Autoencoder convencional

Una red **autoencoder** esta caracterizada por **dos** niveles o **subredes** (al igual que las arquitecturas para segmentación de imagen):

- **Encoder** - $f_{\theta}(\cdot)$: Comprime los datos de entrada en un espacio latente (Z_i) mediante $Z_i = f_{\theta}(X)$.
- **Decoder** - $g_{\theta}(\cdot)$: Tiene como entrada el espacio latente (Z_i) y se encarga de reconstruir una imagen de salida a partir de este según $g_{\theta}(Z_i)$.
- El **proceso completo** de un **autoencoder** queda definido por $g_{\theta}(f_{\theta}(X))$ y el proceso de optimización se basa en **minimizar** el **error** entre los datos reconstruidos y los originales.

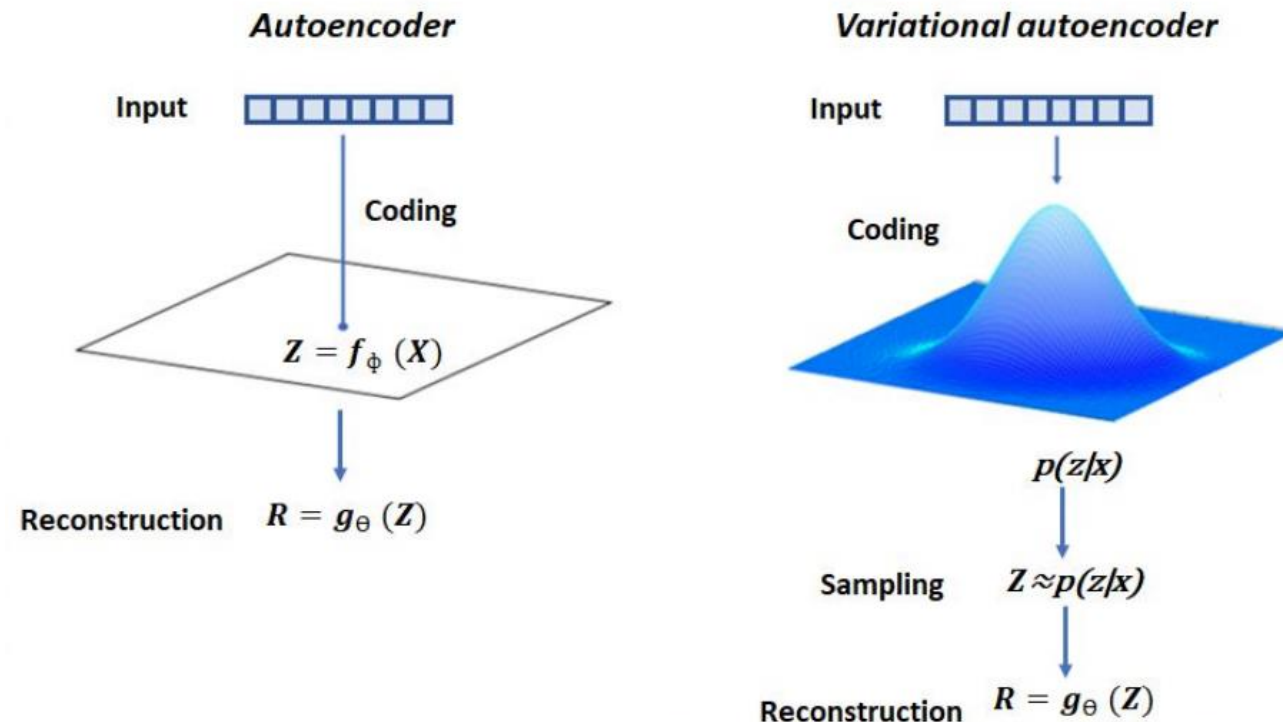


$$\min_{\theta, \phi} L_{rec} = \min \frac{1}{n} \sum_{i=1}^n ||x_i - g_{\theta}(f_{\phi}(x_i))||^2$$

El grand true es la misma dato que se tiene como entrada, ya que el decoder trata de reconstruirlo con el mínimo error posible

Autoencoder variacional

- La versión variacional de un autoencoder (**VAE** del inglés) introduce una **regularización** en el espacio latente para mejorar sus propiedades. VAE codifica los datos de entrada como una **distribución normal multivariante** alrededor de un punto en el espacio latente.

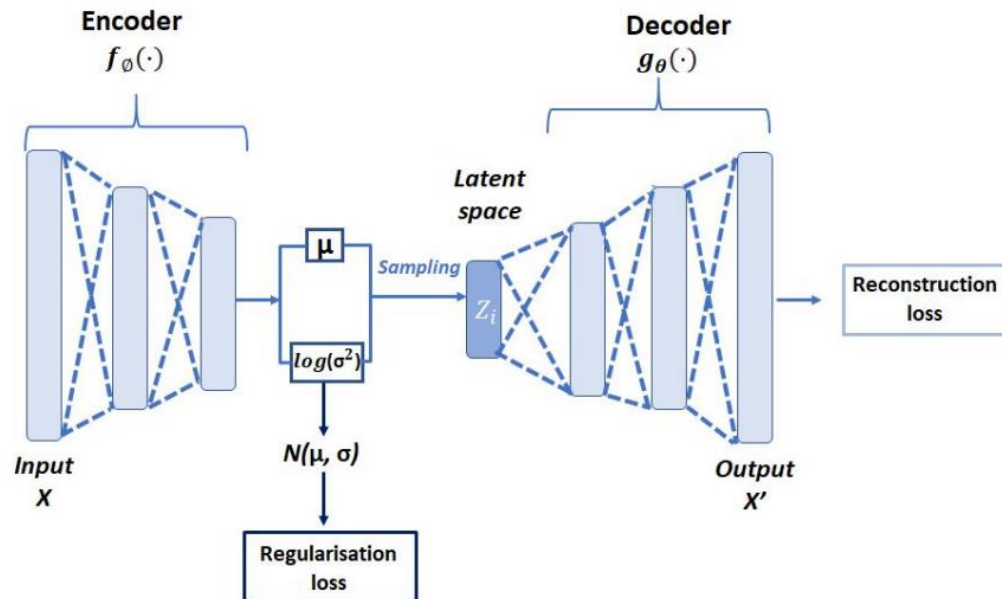


Autoencoder variacional

DKL (Divergencia de Kullback-Leibler), también conocida como entropía cruzada relativa, es una medida utilizada para cuantificar la diferencia o disimilitud entre dos distribuciones de probabilidad.

La DKL se utiliza en diversas aplicaciones, como la clasificación de texto, la compresión de datos, el aprendizaje automático y la teoría de la información. En el aprendizaje automático, la DKL se utiliza para medir la diferencia entre la distribución de salida predicha por un modelo y la distribución de salida real. Minimizar la DKL puede ser un objetivo común en la optimización de modelos de aprendizaje automático.

- El **encoder** asigna cada muestra de entrada a un vector de medias y otro de varianzas.
- Necesidad de **regularizar** tanto el logaritmo de la varianza como la media de la distribución que devuelve el *encoder* → **Match** entre **distribución** que saca el **encoder** y una **distribución normal estándar** (media cero y desviación unidad).
- **Sampling** de la **distribución multivariante** para reconstruir los datos originales.



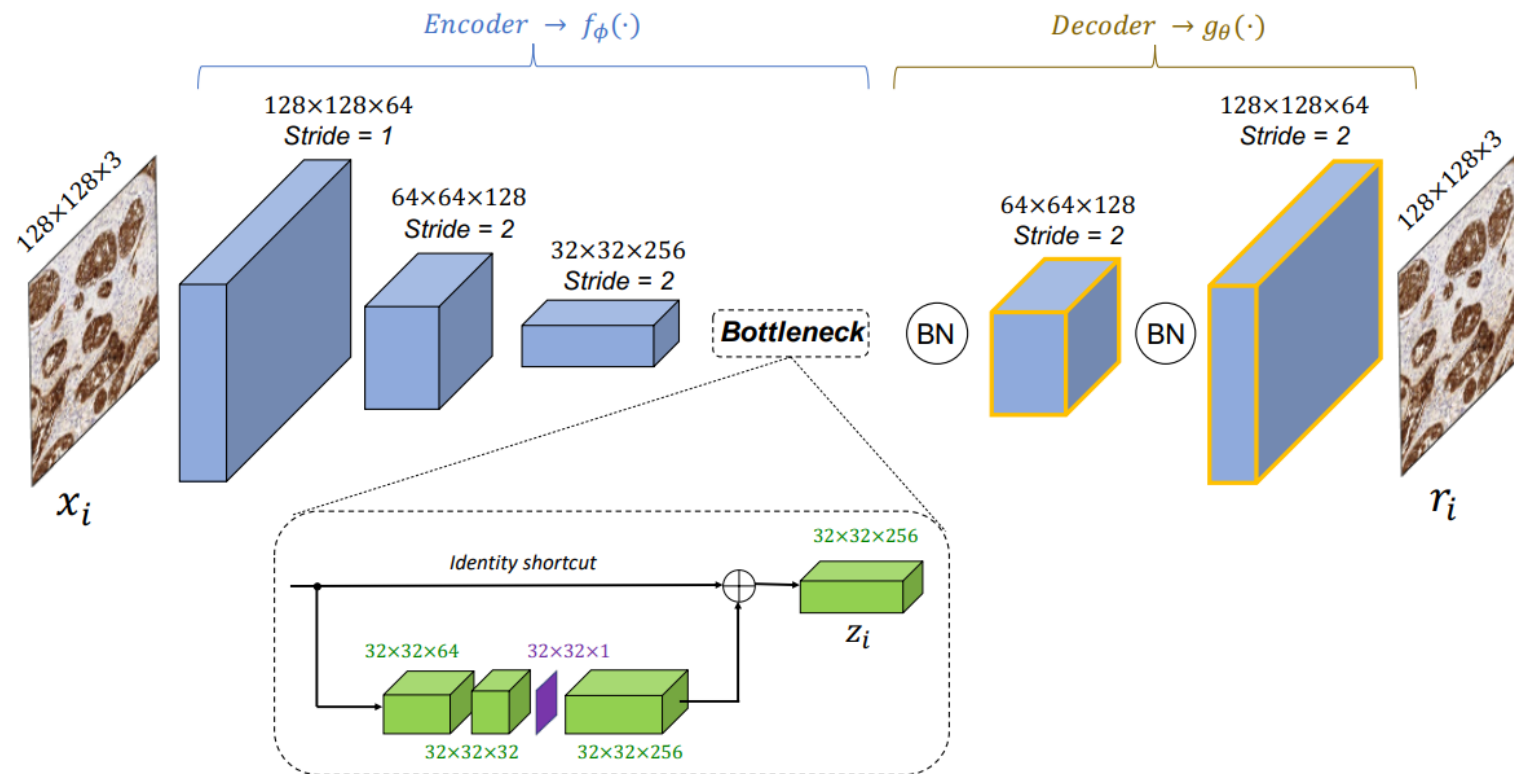
$$Z \approx p(z|x) = \mu + \sigma \cdot \epsilon$$

$$\min_{\theta, \phi} L_{rec} = \min \frac{1}{n} \sum_{i=1}^n \|x_i - g_{\theta}(f_{\phi}(x_i))\|^2$$

$$D_{KL}[N(\mu, \sigma) || N(0, 1)] = \frac{1}{2} \sum (1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

Autoencoder convolucional

- La arquitectura de los **autoencoders** varían según el caso de uso, más concretamente según el tipo de datos a la entrada.





viu

Universidad
Internacional
de Valencia

universidadviu.com

De:
 Planeta Formación y Universidades