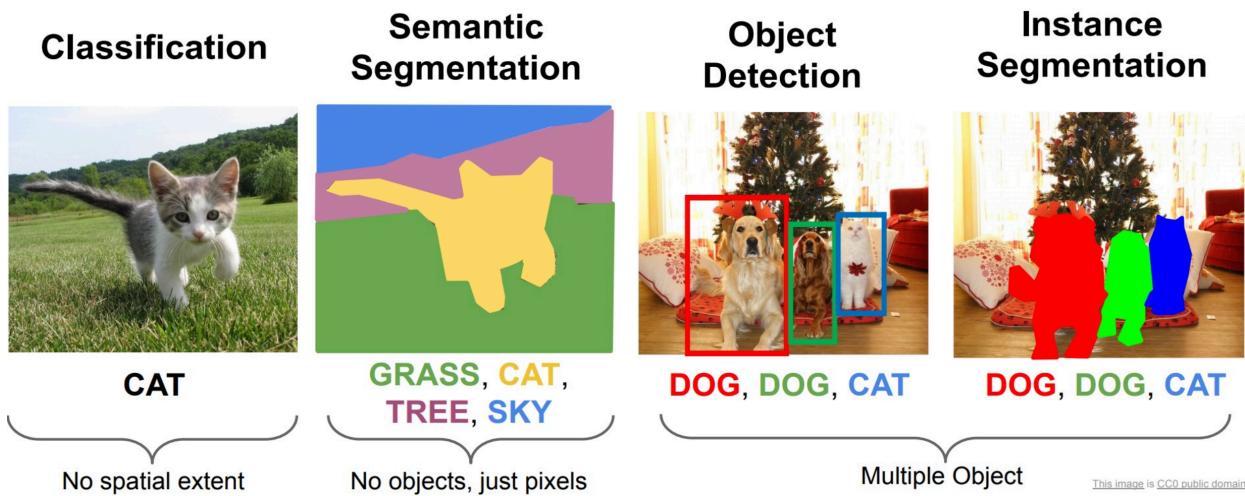


Clase 5

Problemas en computer vision

- **Clasificación.** A una imagen le queremos asignar una etiqueta.
- **Segmentación semántica.** Asignamos una etiqueta a cada pixel.
- **Detección de objetos.** Queremos indicar la posición de un objeto en una imagen.
- **Segmentación de instancias.** Segmentamos a nivel pixel, pero cada instancia del objeto tiene una etiqueta distinta.

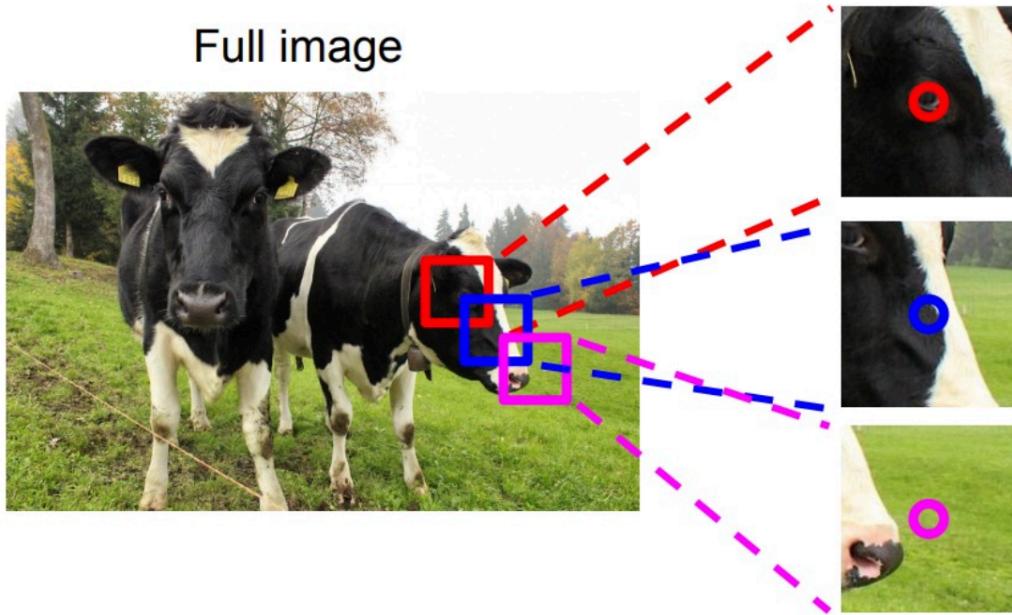


Segmentación de imágenes

- A diferencia de las redes convolucionales usadas para clasificación, que nos devuelven una probabilidad de pertenencia a una clase, estas redes neuronales devuelven una representación de la imagen.

Sliding Window:

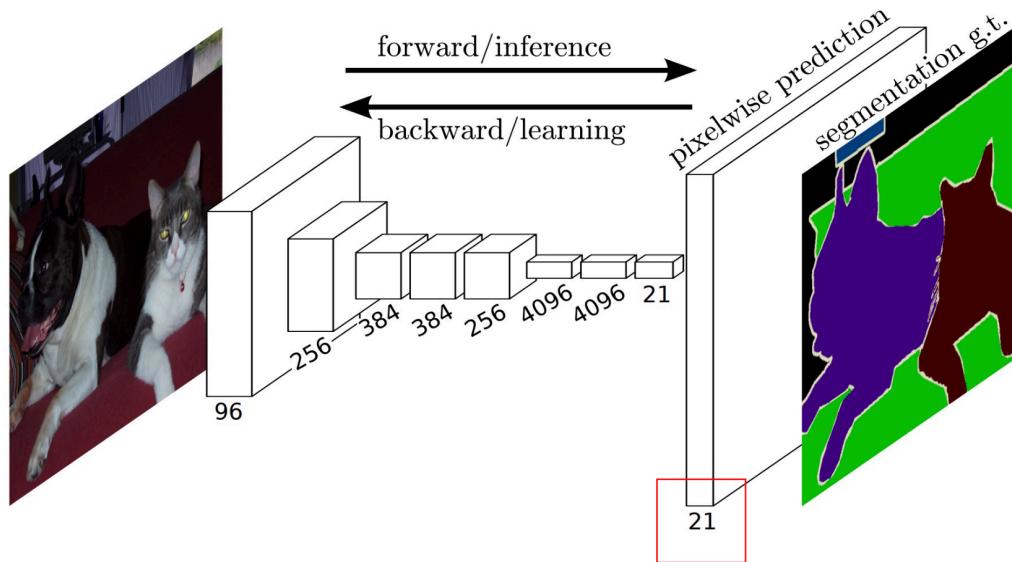
- Armamos un algoritmo que va eligiendo ventanas de una imagen, y a cada una le asigna la etiqueta a la que pertenece el pixel del medio. Por ejemplo, en la imagen siguiente vemos que en las primeras dos ventanas asignamos la etiqueta "vaca" y en la tercera ventana asignamos la etiqueta "pasto".



- Después de asignarle la etiqueta, pasamos cada ventana por una CNN para segmentarla. Sin embargo, no es muy eficiente porque no hace uso de las features compartidas.

Fully Convolutional Networks

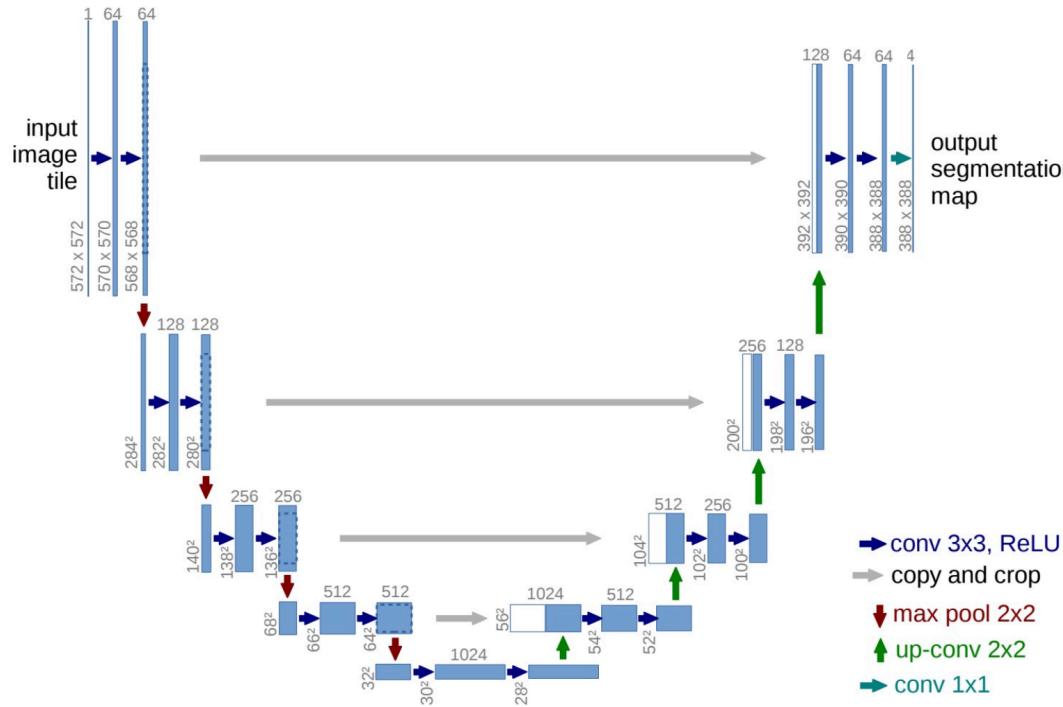
- Cualquier red que solo tenga capas convolucionales es una FCN.
- La primer parte es igual a las CNN: recibe una imagen de entrada a la que le aplica capas convolucionales y capas de pooling hasta llegar a una representación de muy baja resolución (pero que mantiene una estructura de imagen).
- Al final agregamos una capa adicional con tantos feature maps como clases tiene nuestro problema (la dimensión es igual a la dimensión de la última capa convolucional). Por ejemplo, en la siguiente imagen, esta capa está representada por la capa con 21 feature maps (porque hay 21 clases).
 - Los pixels en el primer feature map representan la probabilidad de pertenecer a la primera clase. Los pixels en el segundo feature map representan la probabilidad de pertenecer a la segunda clase. Y así sucesivamente.
 - Para que cada pixel represente la probabilidad de cada clase, aplicamos una función *Softmax*.
- Para volver a la dimensión original aplicamos una operación de *up-pooling* o *up-convolution*.
- La función de pérdida la computamos para cada pixel y promediamos.



- Una ventaja de estas redes es que, a diferencia de las CNN, pueden segmentar imágenes de cualquier tamaño.

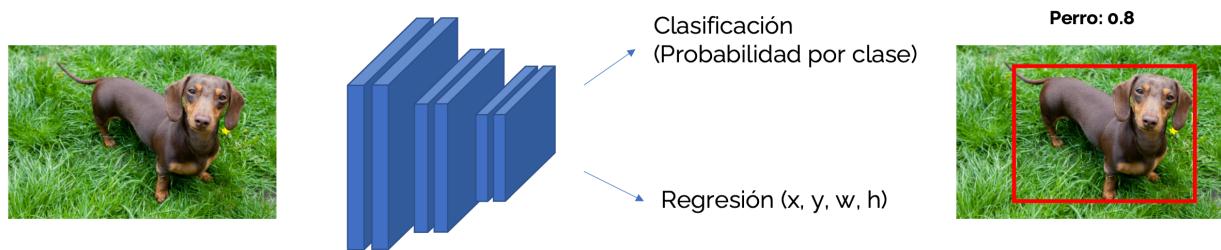
UNet

- Esta red neuronal es una arquitectura de Encoder-Decoder.
- La primera parte es muy similar a las CNN: aplica capas de convolución, para generar nuevas representaciones, y capas de pooling, para reducir la resolución (dimensión). La segunda parte se aplica de forma espejada y aplica capas de convolución y capas de *up-convolution*, para poder aumentar nuevamente la resolución. Esta es la primer diferencia que encontramos con las CNN.
- Una característica importante de esta red neuronal es que tiene **skip-connections**.
 - Al reducir la resolución de las representaciones, estamos perdiendo información. Las skip-connections nos permite volver a incorporar información que, de otra manera, perderíamos.
 - Esto es posible porque al hacer up-convolution recuperamos la misma dimensión que antes.
 - Otra cosa buena es que, al ir agregando información, no corremos el riesgo de que el gradiente se desvanezca.



Arquitectura de localización

- Pasamos la imagen por una serie de capas convolucionales y de pooling. Luego, aplicamos dos rede neuronales:
 - Una red neuronal va a ser para clasificación. Nos va a devolver la probabilidad de pertenecer a cada clase.
 - Una red neuronal va a ser para regresión. Nos va a devolver la posición del objeto en la imagen.



Evaluación de la calidad de segmentación

Dice

$$Dice = \frac{2 | Solución \cap Predicción |}{|Solución| + |Predicción|} = \frac{2 TP}{FN + 2 TP + FP}$$

Jaccard (intersection over union)

- Suele penalizar más las malas clasificaciones en términos cuantitativos.

$$Jaccard = \frac{|Solución \cap Predicción|}{|Solución \cup Predicción|} = \frac{TP}{FN + TP + FP}$$

- Hay una relación entre Jaccard y Dice:

$$Jaccard = \frac{Dice}{2 - Dice}$$

Redes Neuronales Recurrentes

- Son redes neuronales diseñadas para procesar datos secuenciales (series de tiempo, texto, etc.).
- Pueden procesar secuencias de cualquier longitud.
- Una neurona recurrente toma dos inputs: el dato de entrada en el período actual y el estado oculto (i.e., la memoria de la neurona - estado anterior). A esos valores les aplicamos una función parametrizada por los pesos (los cuales se comparten para todos los períodos) para obtener un nuevo estado:

Función parametrizada por W (la misma para todos los instantes t)

$$h_t = f_W(h_{t-1}, x_t)$$

↓

Nuevo Estado Antiguo Estado Entrada

Un ejemplo de esta función:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$