Visión por Computadora II - CEAI - FIUBA

Profesores:

- Cavalieri Juan Ignacio - juanignaciocavalieri@gmail.com
- Cornet Juan Ignacio - juanignaciocornet@gmail.com
- Seyed Pakdaman - khodadad.pakdaman@gmail.com

Cuarta clase:

- Detección de objetos:
  - Bounding boxes intermitentes.
  - CornerNet
- Segmentación
  - Aplicaciones
  - Medidas de error
  - Acercamiento al problema de segmentación
  - Arquitecturas clásicas
    - U-Net
    - DeepLab
    - Mask R-CNN
  - Dataset clásicos
  - Explicando las redes convolucionales

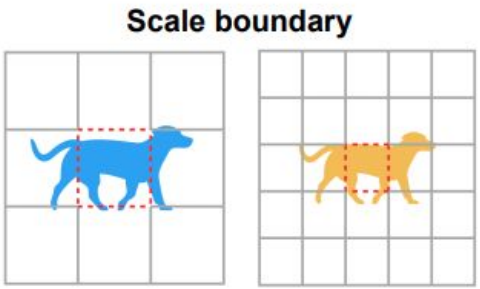
Bounding Boxes intermitentes

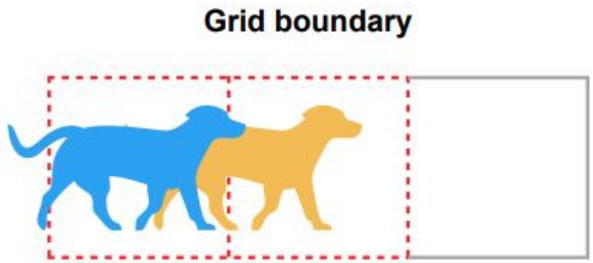
Cuando se procesa un video con una red de objetos basadas en anchor boxes se suele observar un comportamiento indeseado, en el cual, en algunos frames, los objetos dejan de ser detectados.

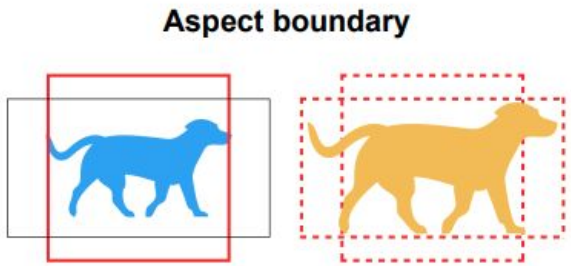
Hosoya, et al., 2019. Analysis and a Solution of Momentarily Missed Detection for Anchor-based Object Detectors. Link

Bounding Boxes intermitentes

Los orígenes de este problema se pueden clasificar en efectos externos, como oclusión u objetos borrosos, o efectos internos debido a un comportamiento inapropiado de la red sobre los bordes de los anchor boxes. Es decir, en el paper, se propone que cuando un objeto pasa de ser detectado por un anchor box a otro, la confianza de detección disminuye considerablemente. Puntualmente, se analizan tres posibles situaciones que generen un cambio en el achor box: desplazamiento, cambio de escala y cambio de relación de aspecto.



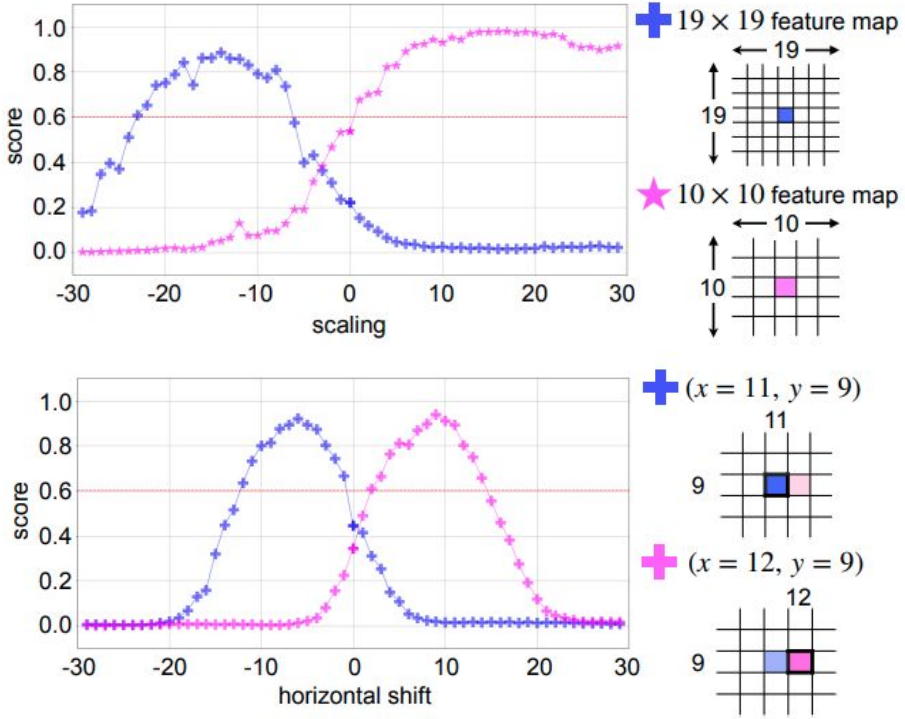




Hosoya, et al., 2019. Analysis and a Solution of Momentarily Missed Detection for Anchor-based Object Detectors. Link

Bounding Boxes intermitentes

Para comprobar este comportamiento, se procesan imágenes utilizando una SSD. Sobre dichas imágenes se realiza alguna modificación (cambio de escala en la gráfica de arriba y desplazamiento horizontal en la de abajo) y se grafica la evolución del valor de confianza arrojado por un determinado par de anchor boxes.



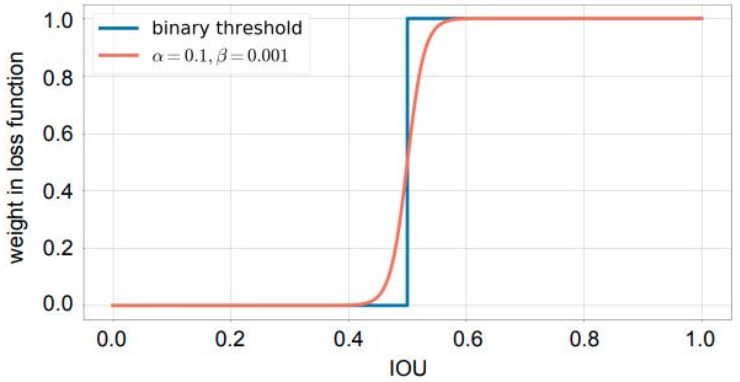
Hosoya, et al., 2019. Analysis and a Solution of Momentarily Missed Detection for Anchor-based Object Detectors. Link

Bounding Boxes intermitentes

El origen de este comportamiento está en la forma en la que se seleccionan las detecciones positivas al momento de entrenar el modelo. Originalmente, se seleccionan múltiples anchor boxes como ejemplos positivos si su IoU con la etiqueta sobrepasa un valor umbral fijo (generalmente 0.5). Se considera a este, un umbral “duro”.
En cambio, ellos proponen utilizar un umbral más “suave”, dado por la siguiente función:

$$f(r) = \frac{1}{1 + \exp\{-a(r - 0.5)\}}.$$

Esta función modifica los valores en el rango [0.5 - α, 0.5 + α]

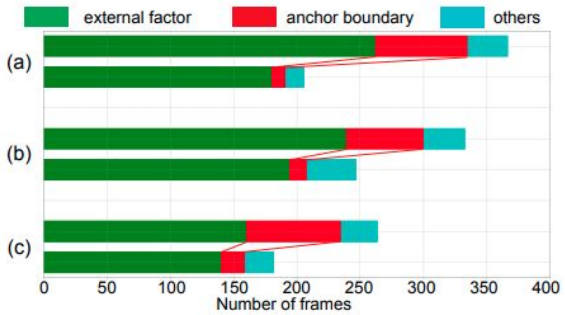


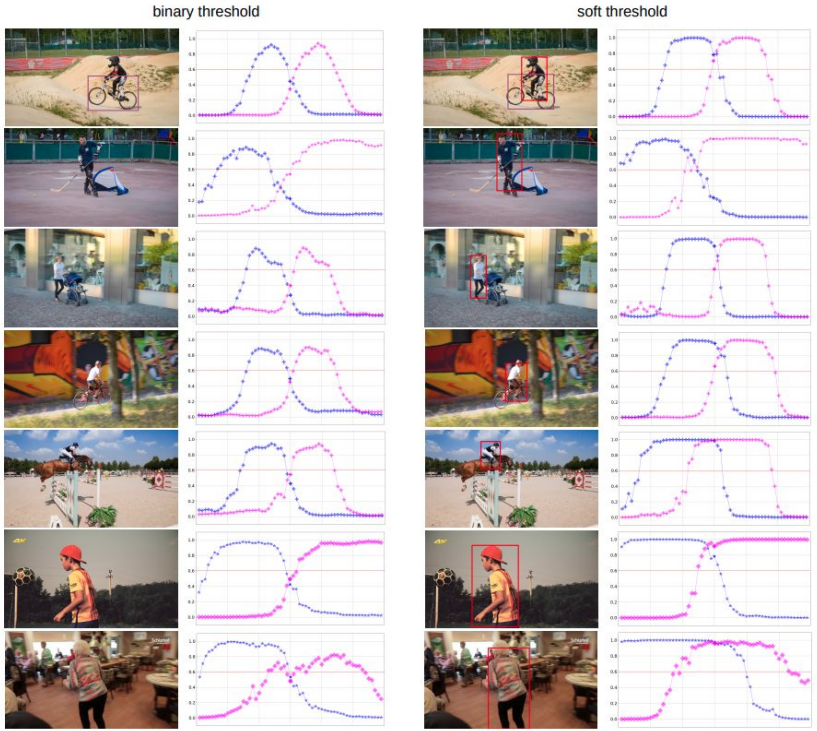
Hosoya, et al., 2019. Analysis and a Solution of Momentarily Missed Detection for Anchor-based Object Detectors. Link

Bounding Boxes intermitentes

Resultados:

Models	SSD-VGG16		SSD-ResNet50		M2Det-VGG16	
Threshold	Binary	Soft	Binary	Soft	Binary	Soft
$mAP^{\dagger}_{0.5}$	76.5	77.2	74.8	74.7	79.3	80.4
MMD frames	367	206	333	247	264	182
External factors	262	180	239	194	160	140
Anchor boundary	73	**11**	61	**14**	75	**19**
Others	32	15	33	39	29	23

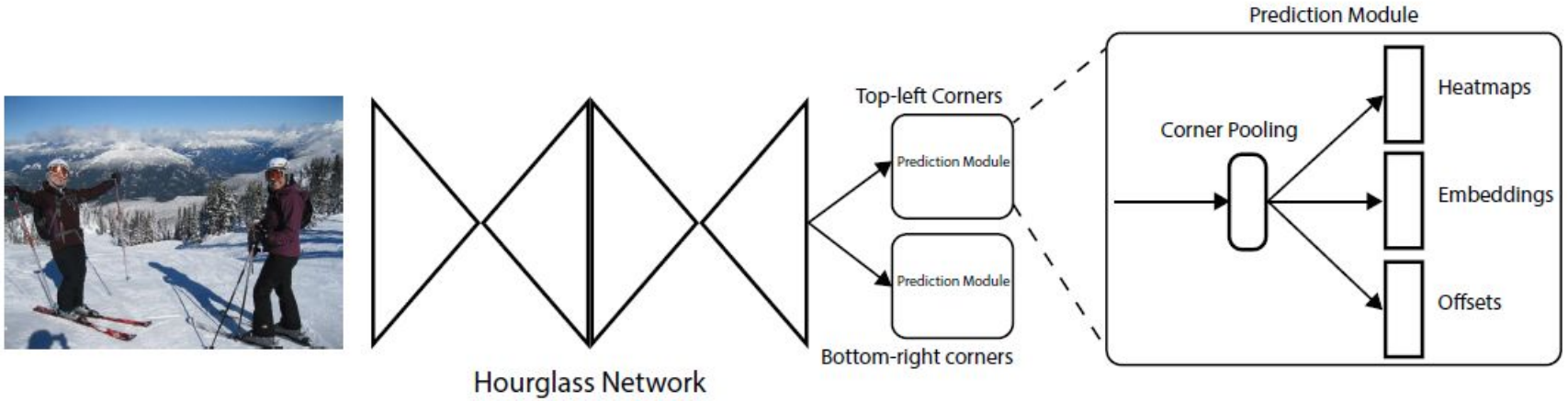




Hosoya, et al., 2019. Analysis and a Solution of Momentarily Missed Detection for Anchor-based Object Detectors. Link

CornerNet

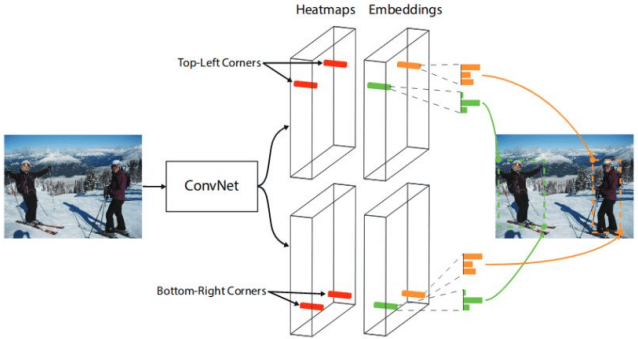
La arquitectura de esta red busca detectar a los bounding boxes de los objetos como un par de puntos, los vértices superior izquierdo y el inferior derecho, **eliminando la necesidad de utilizar anchor boxes!**



Law, et al., 2019. CornerNet: Detecting Objects as Paired Keypoints. Link

CornerNet

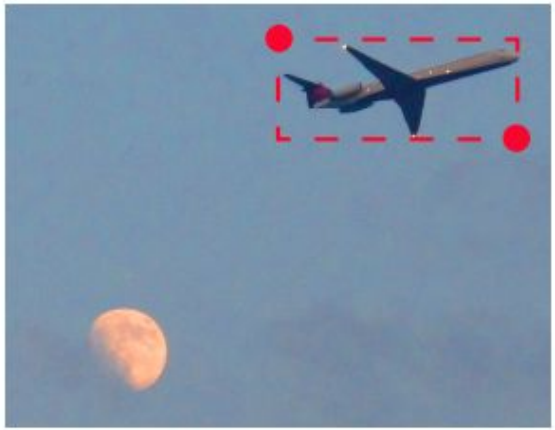
- Como red convolucional para extraer features se utiliza una arquitectura tipo hourglass.
- Hay dos módulos de predicciones, uno para las esquinas superiores izquierdas de los objetos y otro para las esquinas inferiores derechas.
- Cada módulo contiene una capa de **corner pooling** que alimenta las predicciones de heatmaps, embeddings y offsets.
- Los heatmaps indican las clases de las predicciones. La salida tiene *c* canales, donde *c* es la cantidad de clases. No hay clase de background.
- Los embeddings están entrenados para poder asociar los corners de un mismo objeto, devuelto por cada predictor.
- Los offsets sirven para ajustar finamente los bounding boxes predichos.



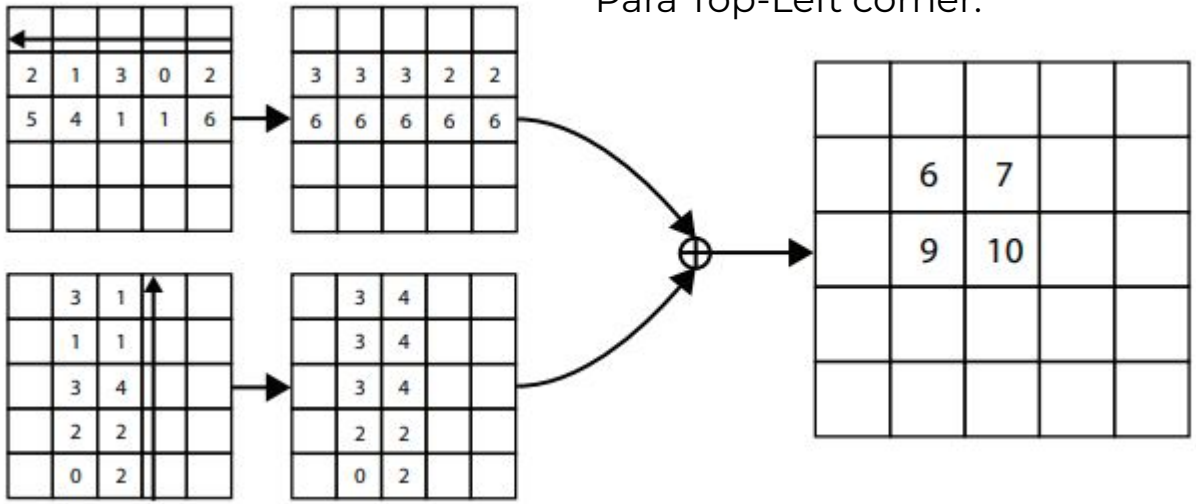
CornerNet

Corner Pooling

El problema de este tipo de redes es que las corners de los bounding boxes no suelen estar en lugares de la imagen donde haya contexto sobre el objeto en cuestión. Para solventar este problema, se introduce la capa de corner pooling.

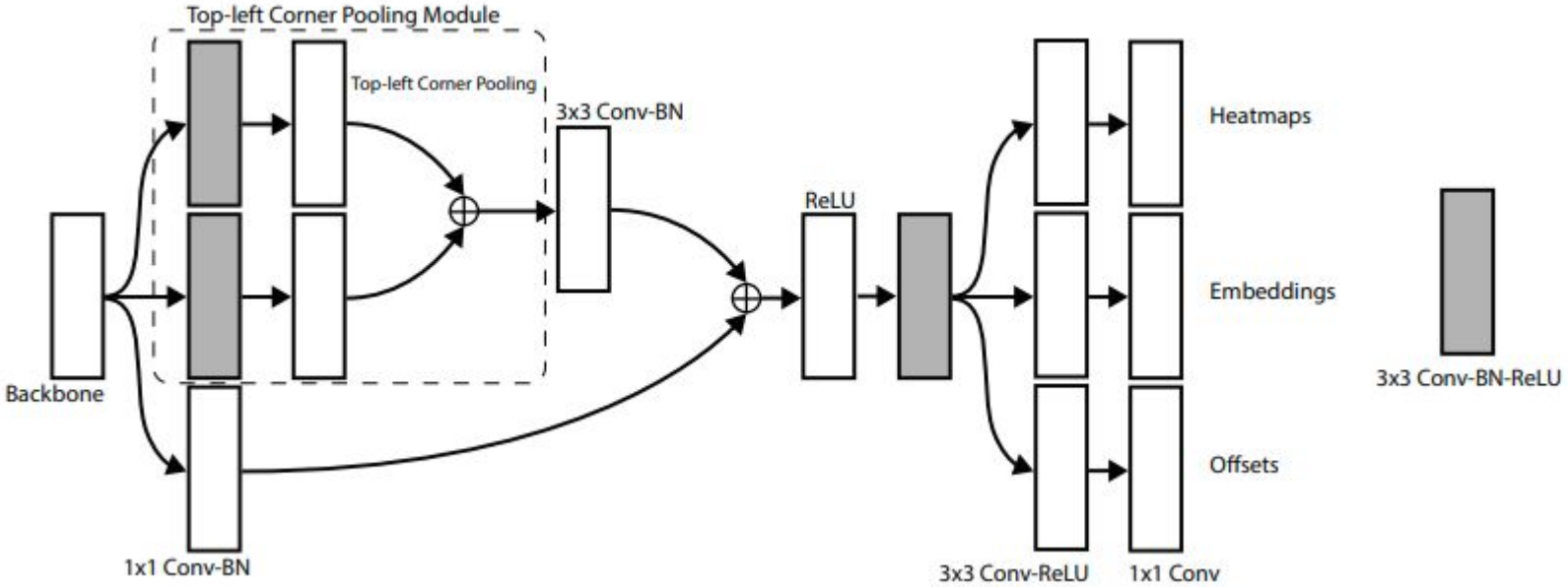


Para Top-Left corner:



CornerNet

Arquitectura de un bloque de predicciones:



CornerNet

Method	Backbone	AP	AP^{50}	AP^{75}	AP^s	AP^m	AP^l	AR^1	AR^{10}	AR^{100}	AR^s	AR^m	AR^l
Two-stage detectors													
DeNet [39]	ResNet-101	33.8	53.4	36.1	12.3	36.1	50.8	29.6	42.6	43.5	19.2	46.9	64.3
CoupleNet [46]	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8	30.0	45.0	46.4	20.7	53.1	68.5
Faster R-CNN by G-RMI [16]	Inception-ResNet-v2 [38]	34.7	55.5	36.7	13.5	38.1	52.0	-	-	-	-	-	-
Faster R-CNN+++ [15]	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9	-	-	-	-	-	-
Faster R-CNN w/ FPN [22]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Faster R-CNN w/ TDM [35]	Inception-ResNet-v2	36.8	57.7	39.2	16.2	39.8	52.1	31.6	49.3	51.9	28.1	56.6	71.1
D-FCN [7]	Aligned-Inception-ResNet	37.5	58.0	-	19.4	40.1	52.5	-	-	-	-	-	-
Regionlets [43]	ResNet-101	39.3	59.8	-	21.7	43.7	50.9	-	-	-	-	-	-
Mask R-CNN [13]	ResNeXt-101	39.8	62.3	43.4	22.1	43.2	51.2	-	-	-	-	-	-
Soft-NMS [2]	Aligned-Inception-ResNet	40.9	62.8	-	23.3	43.6	53.3	-	-	-	-	-	-
LH R-CNN [21]	ResNet-101	41.5	-	-	25.2	45.3	53.1	-	-	-	-	-	-
Fitness-NMS [40]	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5	-	-	-	-	-	-
Cascade R-CNN [4]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2	-	-	-	-	-	-
D-RFCN + SNIP [37]	DPN-98 [5]	45.7	67.3	51.1	29.3	48.8	57.1	-	-	-	-	-	-
One-stage detectors													
YOLOv2 [31]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
DSOD300 [33]	DS/64-192-48-1	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0
GRP-DSOD320 [34]	DS/64-192-48-1	30.0	47.9	31.8	10.9	33.6	46.3	28.0	42.1	44.5	18.8	49.1	65.0
SSD513 [25]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8	28.3	42.1	44.4	17.6	49.2	65.8
DSSD513 [10]	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1	28.9	43.5	46.2	21.8	49.1	66.4
RefineDet512 (single scale) [45]	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4	-	-	-	-	-	-
RetinaNet800 [23]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2	-	-	-	-	-	-
RefineDet512 (multi scale) [45]	ResNet-101	41.8	62.9	45.7	25.6	45.1	54.1	-	-	-	-	-	-
CornerNet511 (single scale)	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9	35.3	54.3	59.1	37.4	61.9	76.9
CornerNet511 (multi scale)	Hourglass-104	42.1	57.8	45.3	20.8	44.8	56.7	36.4	55.7	60.0	38.5	62.7	77.4

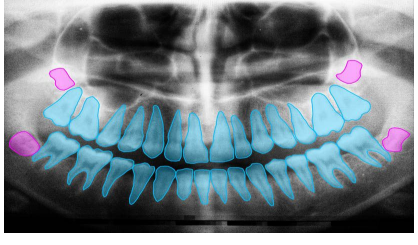
Aplicaciones de image segmentation

- Fotografía computacional
- Medición de área con satélites
- Identificación de tejidos (e.g. tumores vs. tejido sano)
- Cirugía guiada por computadora
- Realidad aumentada
- Encontrar límites geográficos con satélites
- Apps de celular para pintar regiones o aplicar filtros selectivamente
- Segregación de residuos

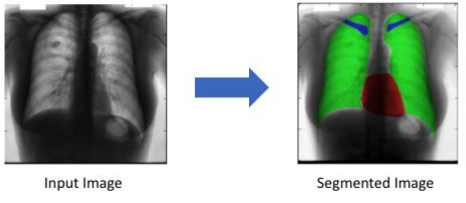
Ejemplos

- Segmentación semántica de campos: https://www.youtube.com/watch?v=wfObVKKKJkE&t=2s
- Reemplazo del cielo en realidad aumentada: https://www.youtube.com/watch?v=B6X0q7YGUDQ
- Segmentación de imágenes urbanas en tiempo real: https://www.youtube.com/watch?v=qWl9idsCuLQ

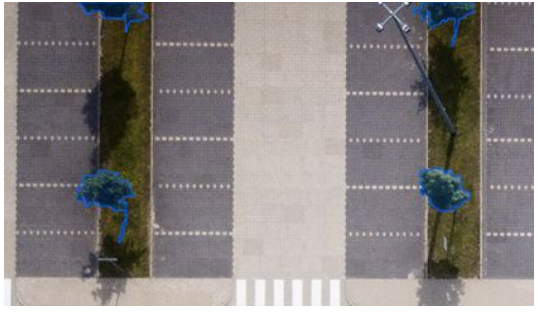
- Imágenes dentales:



- Imágenes médicas:



- Agricultura de precisión:



- Segmentación de residuos:

Medidas de error

- Pixel Accuracy: porcentaje de píxeles en la imagen clasificados correctamente.
  Problemas: si hay mucho fondo el resultado de la métrica puede no ser significativo (en general un problema si las clases están muy desbalanceadas)

$$\text{PA} = \frac{\sum_{i=0}^{K} p_{ii}}{\sum_{i=0}^{K} \sum_{j=0}^{K} p_{ij}},$$

- Mean Pixel Accuracy: para salvar el problema anterior, una alternativa muy utilizada es calcular el Pixel Accuracy por clase, y luego promediarlos.

$$\text{MPA} = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij}}.$$

Medidas de error

- IoU promedio: se toma la IoU para cada clase en la imagen y se saca el promedio

$$\text{IoU} = J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

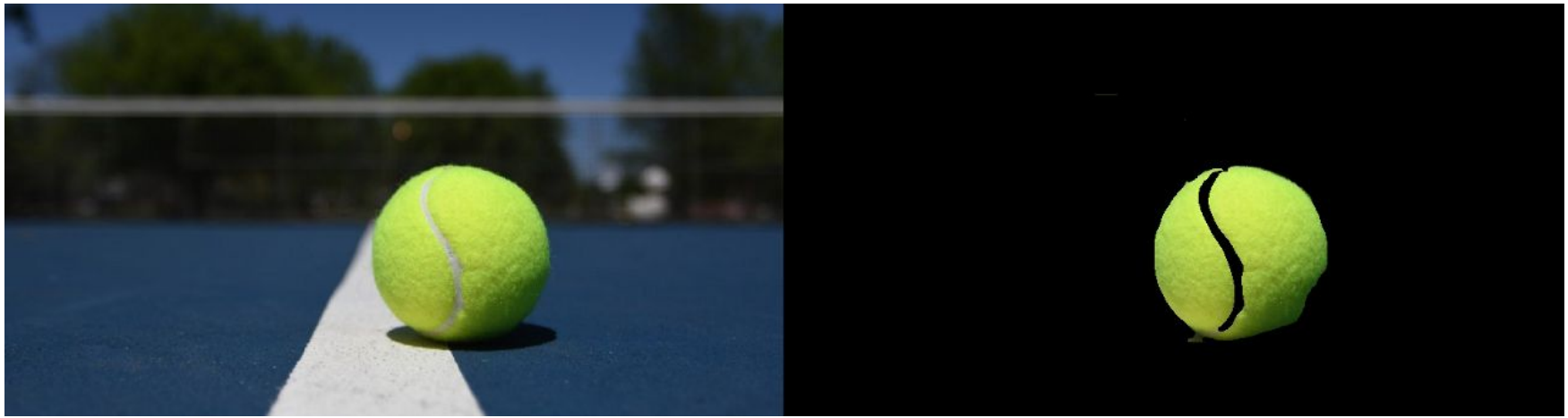
- Coeficiente Dice o F1 score: 2 veces el área de la intersección dividida la suma de las áreas

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}.$$

Otras medidas de error:

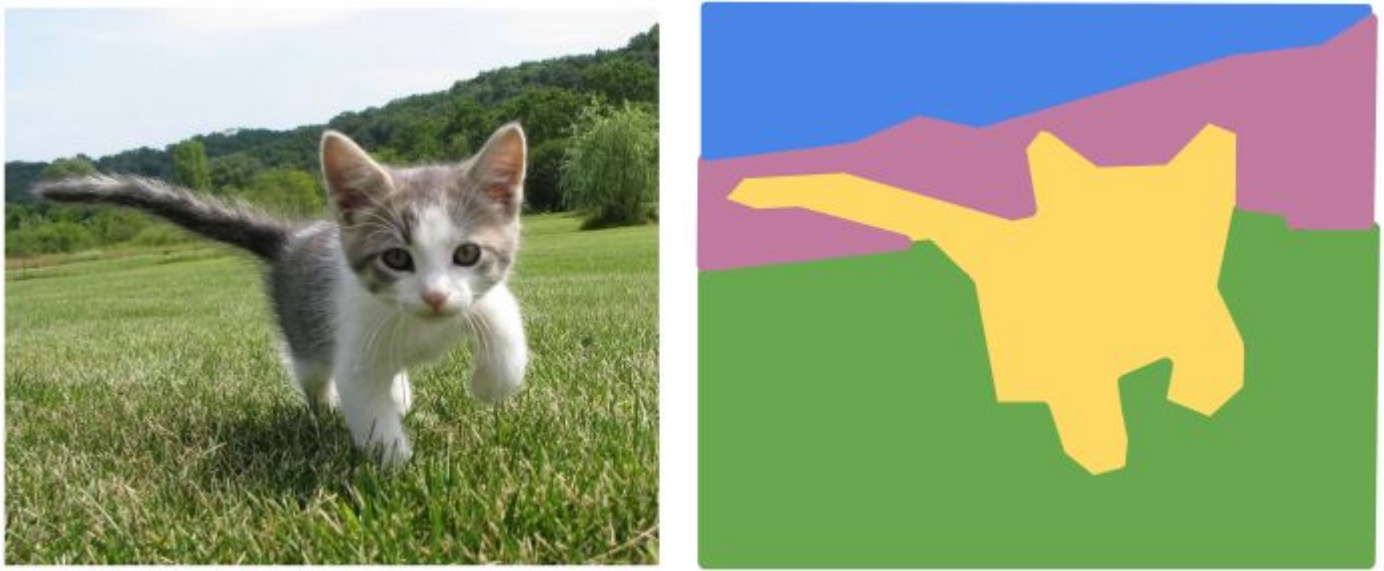
Shruti Jadon, A survey of loss functions for semantic segmentation Link

Segmentación Pixel a Pixel



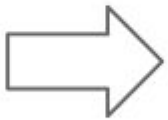
Segmentación Pixel a Pixel

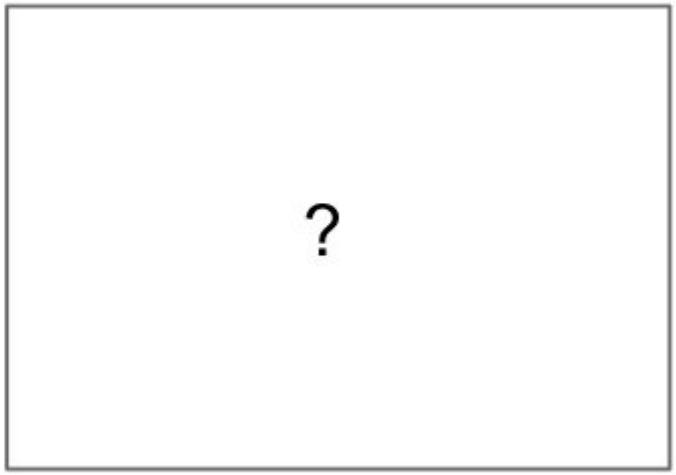
Segmentación semántica



Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Segmentación semántica



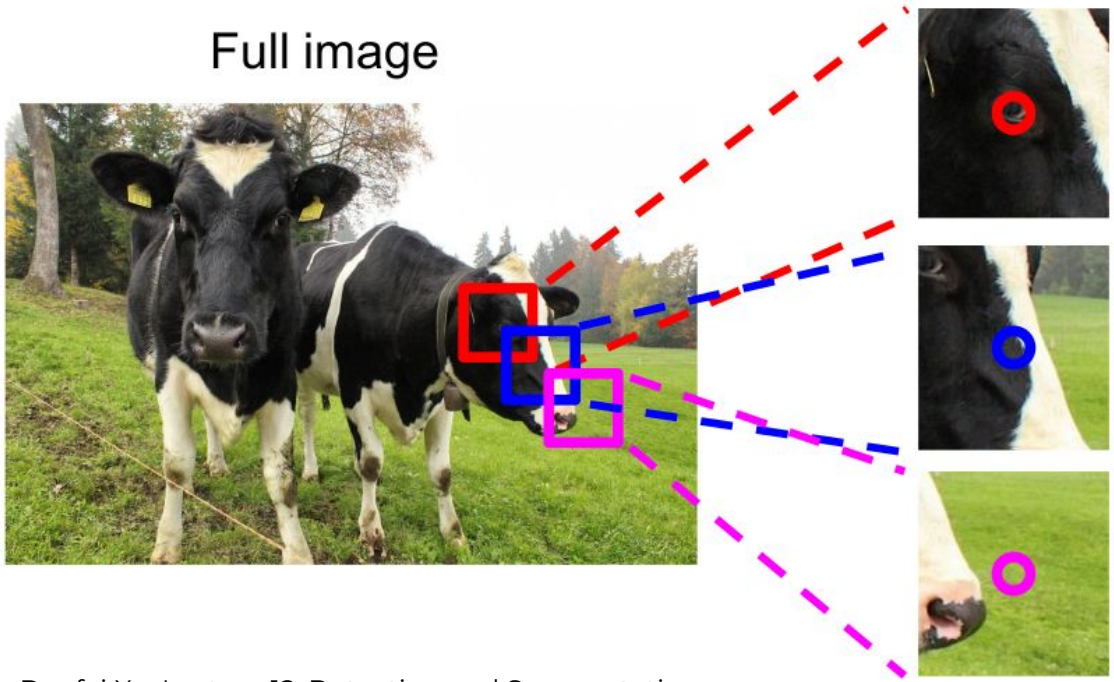


Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Enfoque con ventana deslizante

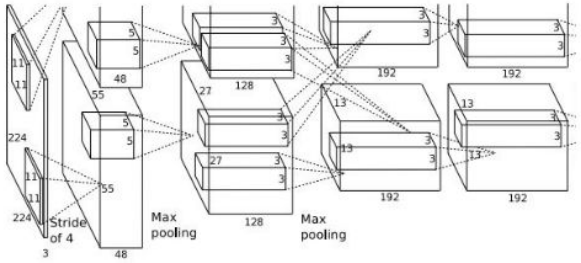
Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Enfoque con ventana deslizante



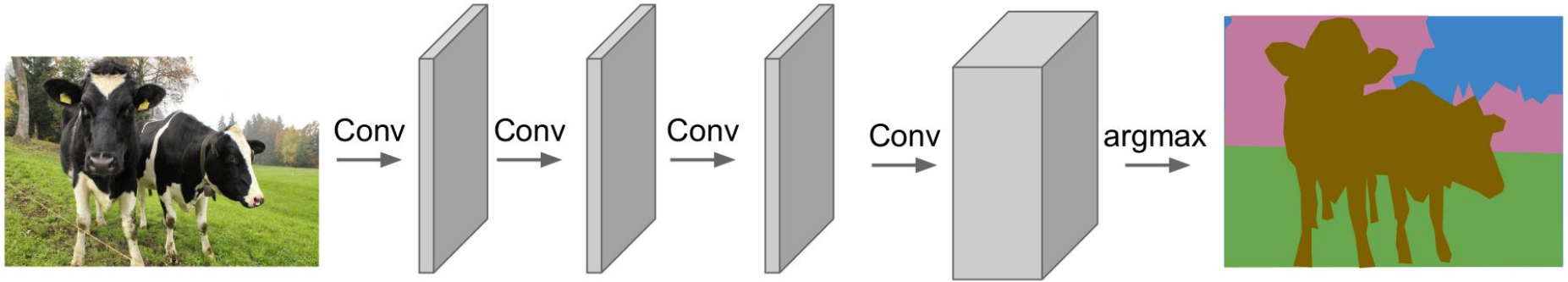
Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Enfoque con ventana deslizante

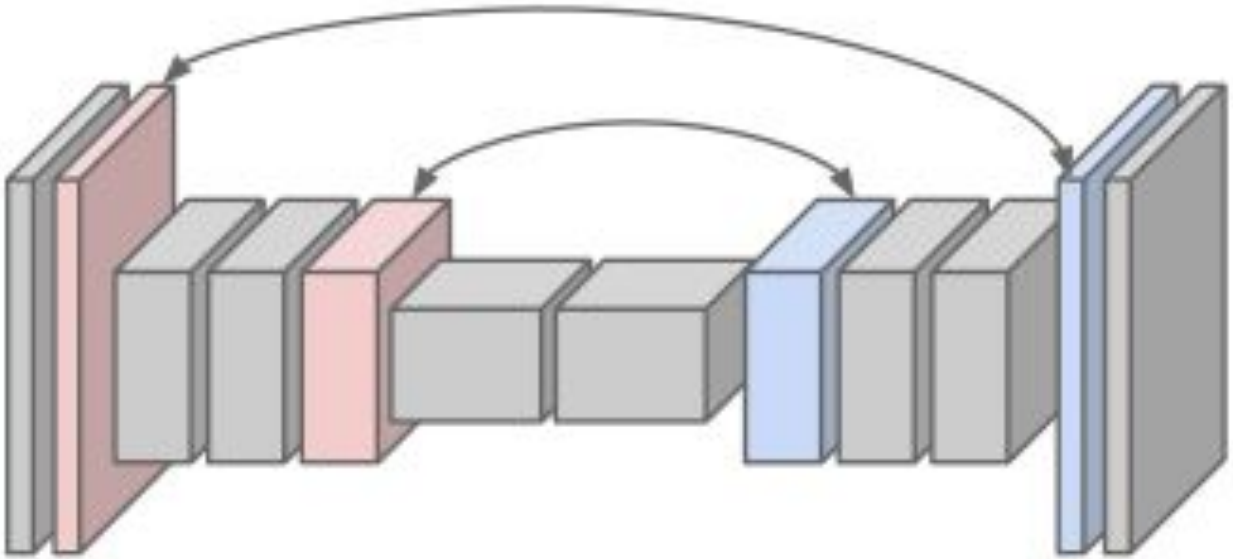


Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Fully Convolutional Network

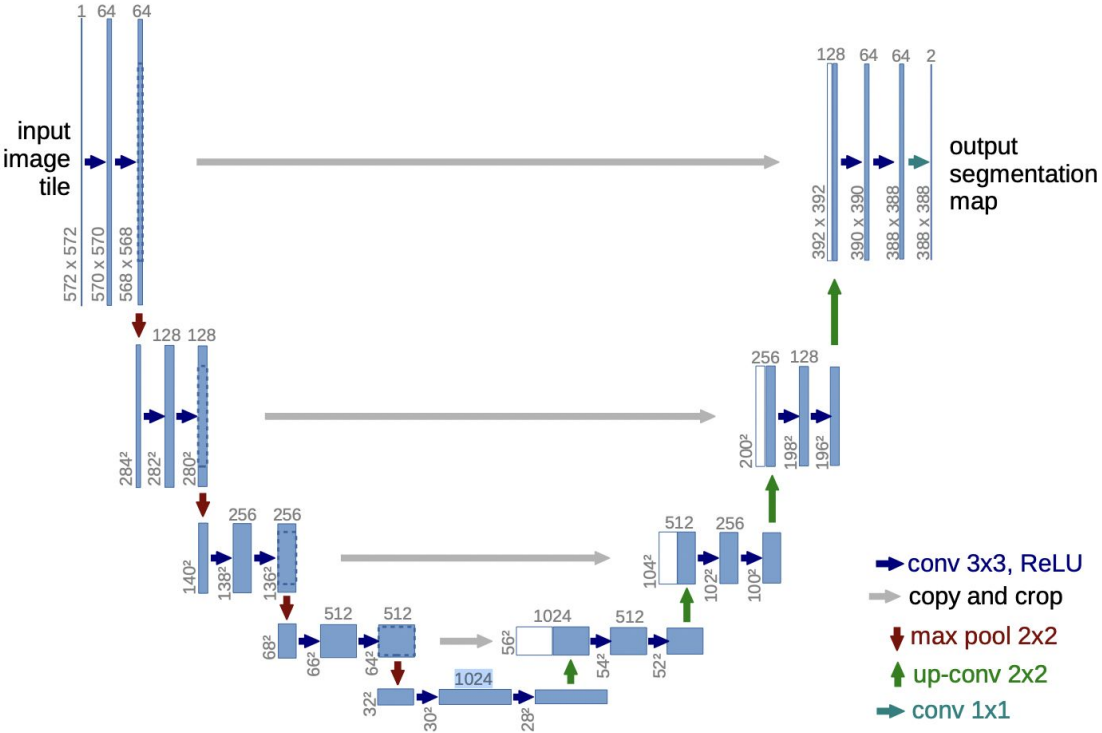


Enfoque Codificador Decodificador



Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

U-Net: Arquitectura



Ronnemberg et al, U-Net: Convolutional Networks for Biomedical Image Segmentation

U-Net

- Es una red fully convolutional, esto quiere decir que convierte capas fully connected en convolucionales. De esta manera permite aplicar la red a imágenes de diverso tamaño.

- La arquitectura tiene forma de U y tiene un enfoque de codificador, decodificador. Hay una parte inicial que contrae la imagen y otra parte posterior que la expande nuevamente. Para ello se usa upsampling a través de transposed convolution (“convoluciones transpuestas”)

- Se propaga la información desde capas de la parte contractiva a capas de la parte expansiva de la misma “granularidad con “skip connections”

- Data augmentation: para el entrenamiento se realizan traslaciones, rotaciones, modificaciones de la escala de grises y **deformaciones elásticas aleatorias**

- Se usó para segmentación de estructuras neuronales y de células

U-Net: Convolutional Networks for Biomedical Image Segmentation, Link

Código para construcción de UNet

https://colab.research.google.com/drive/1_-v-2n9SfkDe6vITX9lK4akGba88s8kL?usp=sharing

Fuentes: https://www.kaggle.com/phoenigs/u-net-dropout-augmentation-stratification, https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5

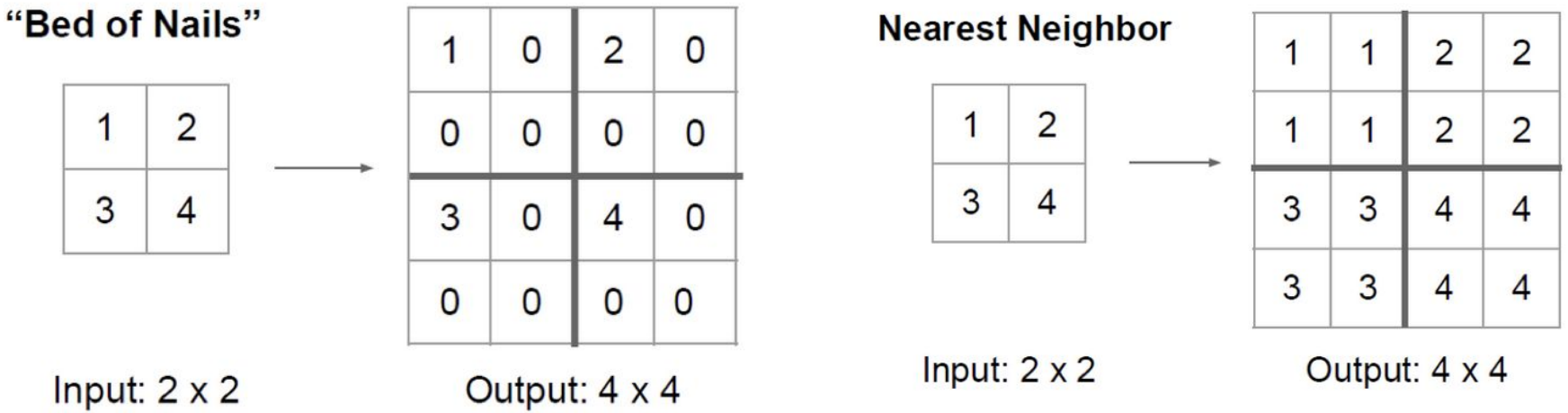
Función de costo:

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

- La función de costo se calcula sumando la contribución de cada pixel:
- **p_l(x)** es una softmax que usa las activaciones en cada canal para estimar una probabilidad que le asigna la red a que ese píxel esté en la clase **k**, y **l** es el índice de la verdadera clase del píxel
- La función **w(x)** tiene dos propósitos.
  - **w_c** compensa el desbalance de clases
  - el segundo término le da más peso a píxeles que están cerca de bordes, **d_1** es la distancia a la celda más cercana y **d_2** a la segunda celda más cercana, da más peso a píxeles cerca de bordes

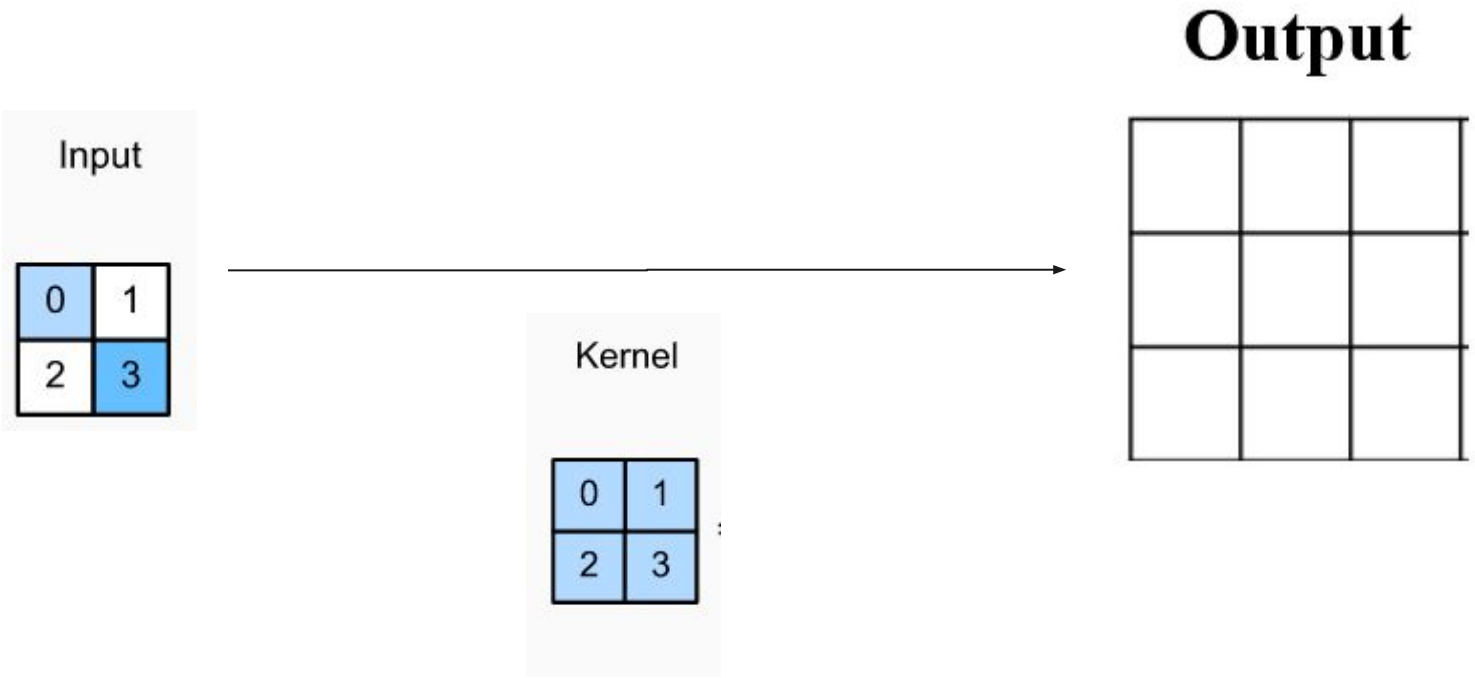
$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

Unpooling

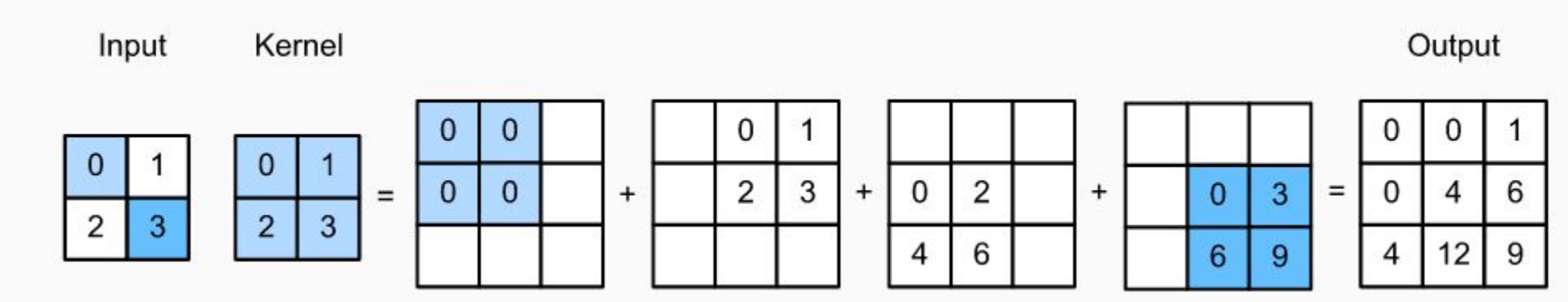


Unsampling: Unpooling and Transpose Convolution, Link

Convolución transpuesta



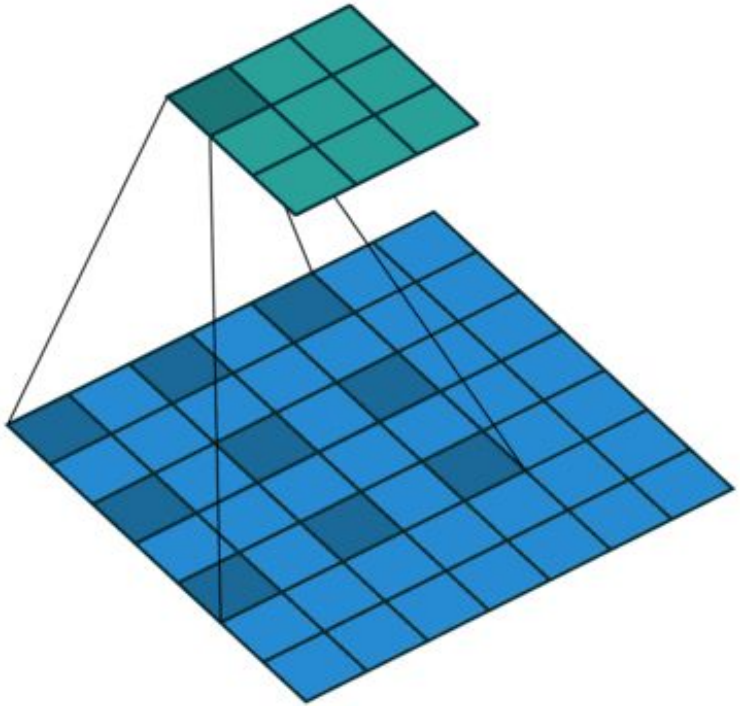
Convolución transpuesta



DeepLab

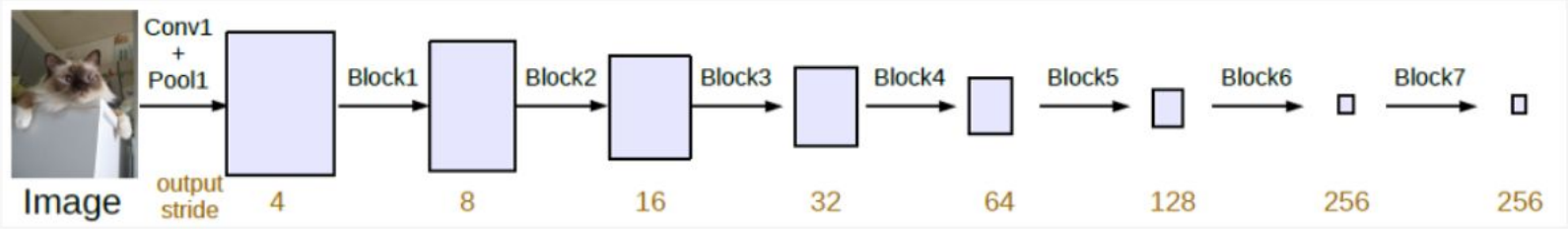
- Modelo moderno de segmentación semántica: https://github.com/tensorflow/models/tree/master/research/deeplab
- De backend usa Resnet-101 y luego Aligned XCeption (V3+)
- Introduce varias novedades:
  - Atrous convolution: permite controlar la resolución a la que las diferentes mapas de caraterísticas son computados en DCNN
  - Atrous Spatial Pyramid Pooling: para segmentar objetos en múltiples escalas con filtros de diferentes sampling rates y field-of-views (campos de visión) efectivos.
  - En versiones posteriores además incluyen mejoras adicionales, como batch normalization, mejoras en la detección de bordes, y en el control del trade-off entre precisión y velocidad de procesamiento
- La versión más nueva es Deeplab V3+
- Paper de Deeplab, Paper Deeplab V2, Paper Deeplab V3, Paper Deeplab V3+

Atrous convolution

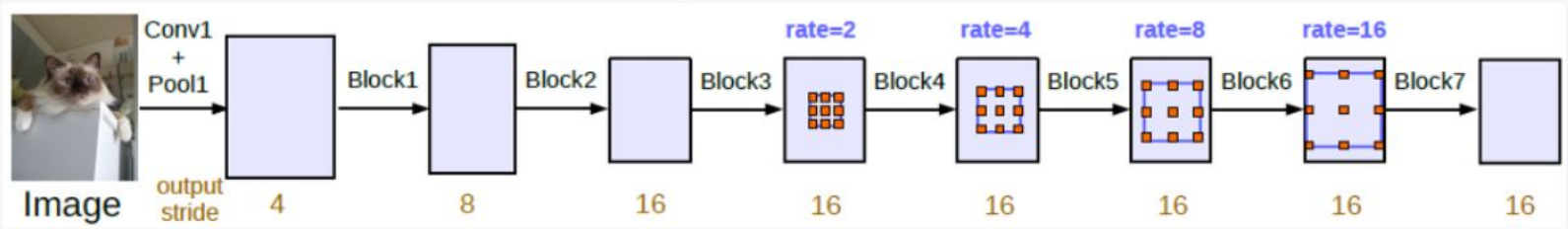


$$y[i] = \sum_{k} x[i + r \cdot k] w[k]$$

Atrous convolution

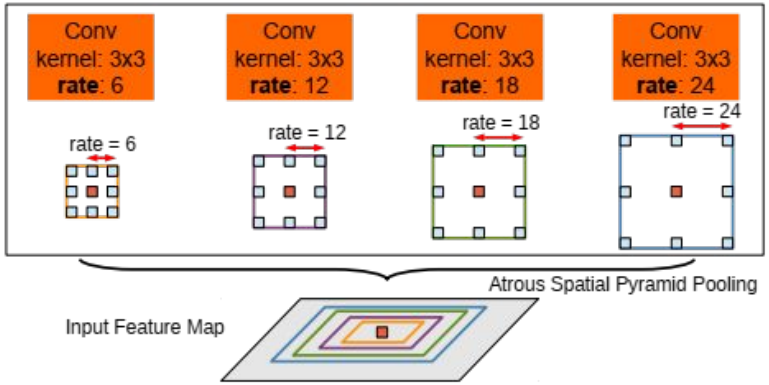


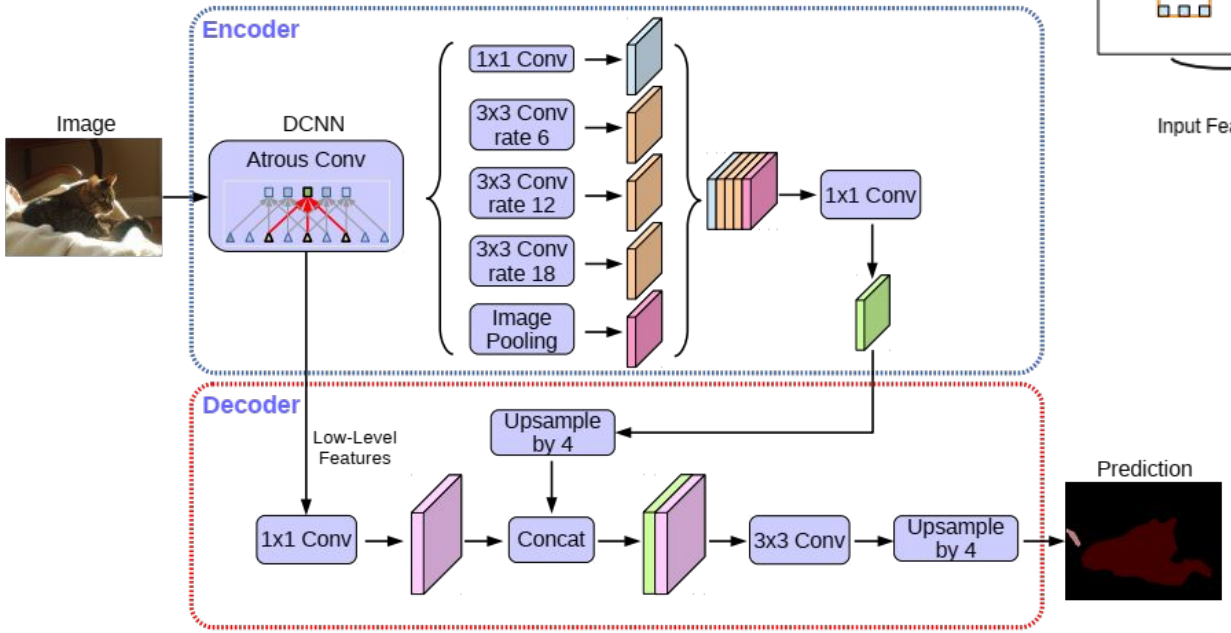
(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Atrous Spatial Pyramid Pooling





- El encoder toma features a diferentes escalas con varios rates de atrous convolutions
- El encoder hace downsample x16 y el decoder x4 y luego x4 para obtener mismo tamaño

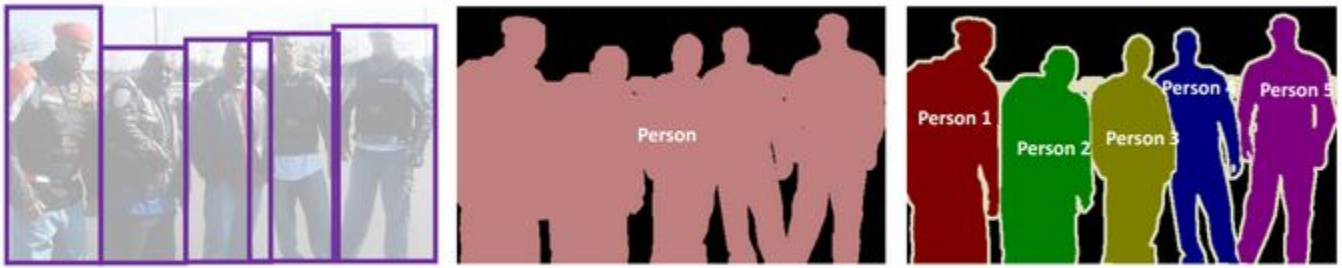
Performance Pascal VOC 2012

Method	mIOU
Deep Layer Cascade (LC) [82]	82.7
TuSimple [77]	83.1
Large_Kernel_Matters [60]	83.6
Multipath-RefineNet [58]	84.2
ResNet-38_MS_COCO [83]	84.9
PSPNet [24]	85.4
IDW-CNN [84]	86.3
CASIA_IVA_SDN [63]	86.6
DIS [85]	86.8
DeepLabv3 [23]	85.7
DeepLabv3-JFT [23]	86.9
DeepLabv3+ (Xception)	87.8
DeepLabv3+ (Xception-JFT)	89.0

Performance Cityscapes

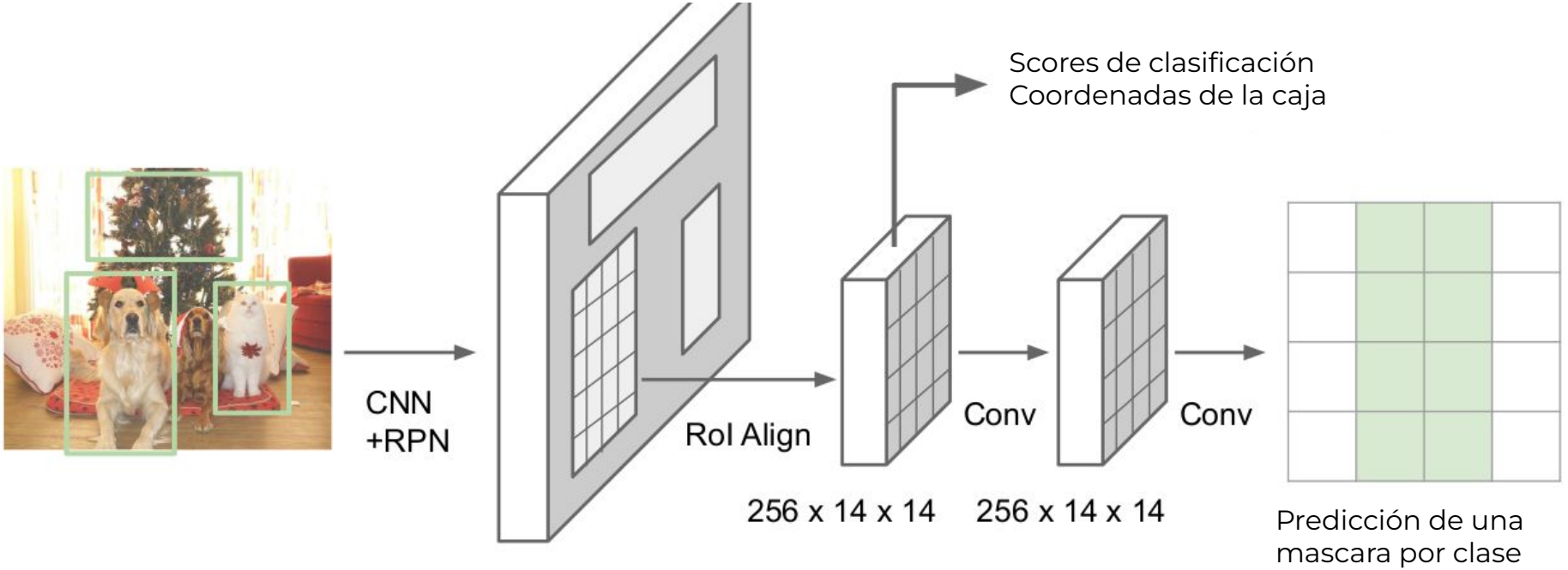
Method	Coarse	mIOU
ResNet-38 [83]	✓	80.6
PSPNet [24]	✓	81.2
Mapillary [86]	✓	82.0
DeepLabv3	✓	81.3
DeepLabv3+	✓	82.1

Segmentación de instancias



Object Detection | Semantic Segmentation | **Instance Segmentation**

Mask R-CNN



Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 12: Detection and Segmentation

Mask R-CNN

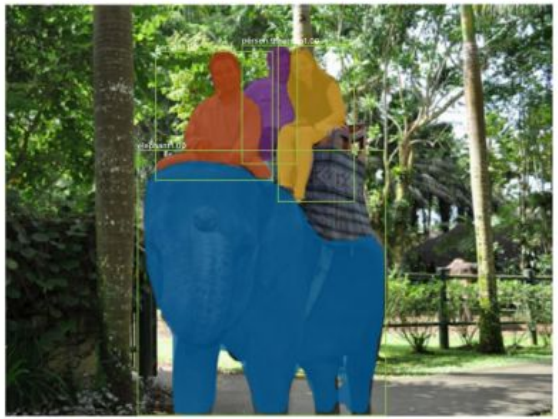
Proviene de la familia R-CNN, Fast R-CNN y Faster R-CNN.

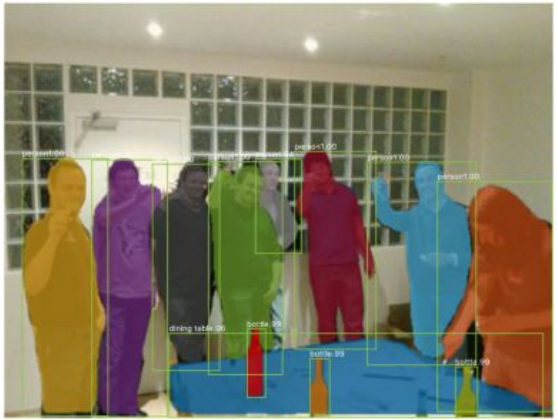
Como ellas es una red de dos etapas.

En la segunda etapa se predice una mascara binaria para cada clase en cada una de las RoI, la cual es procesada en paralelo con la predicción de clase y bounding box.
Se define una función de costo multitarea, contemplando un término para la clase, un término para el bounding box y otro para la máscara. Los primeros son idénticos a los de Fast R-CNN.
L mask es la binary cross-entropy entre la máscara predicha y la ground truth, pixel a pixel (en la resolución de la máscara).
Solo la máscara de la clase relacionada a la RoI en cuestión contribuye a la función de costo.

He et al. Mask R-CNN. Link

Mask R-CNN





He et al. Mask R-CNN. Link

Open Source Frameworks

Hay muchas buenas implementaciones de código abierta, en varios lenguajes.

TensorFlow Detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection
Faster RCNN, SSD, RFCN, Mask R-CNN, ...

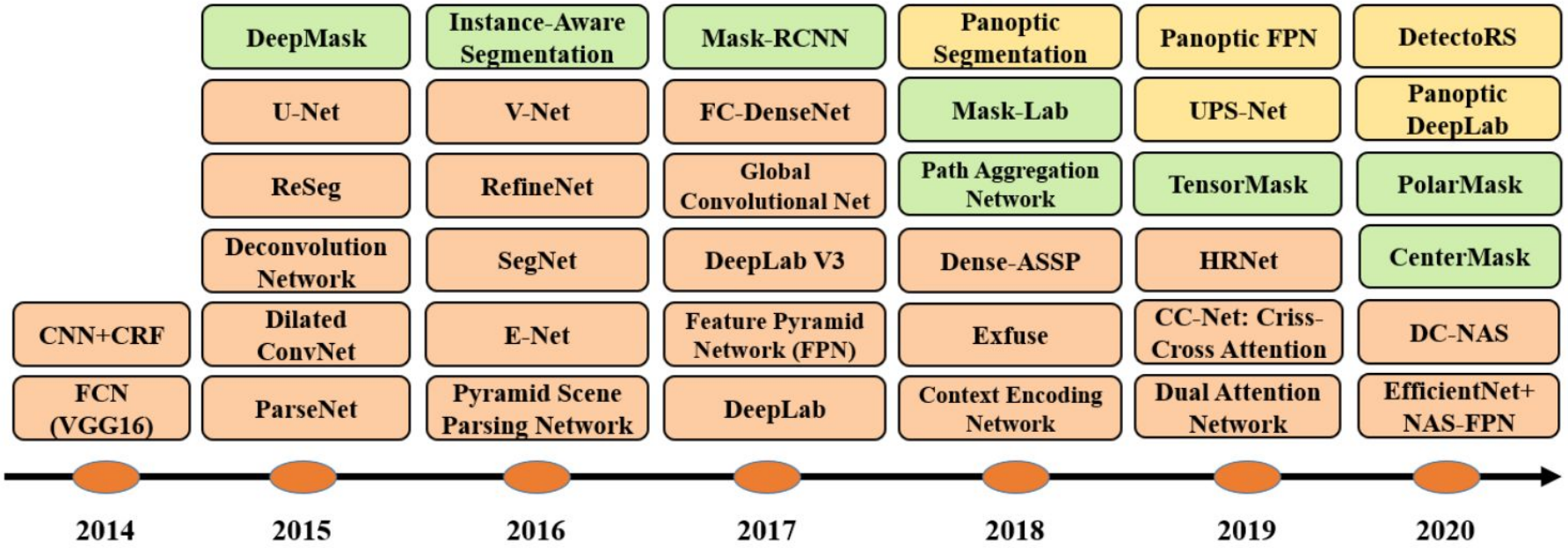
Detectron2 (PyTorch)
https://github.com/facebookresearch/detectron2
Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN, ...

Pueden ser usados para reentrenar modelos haciendo Fine Tuning o Transfer Learning.

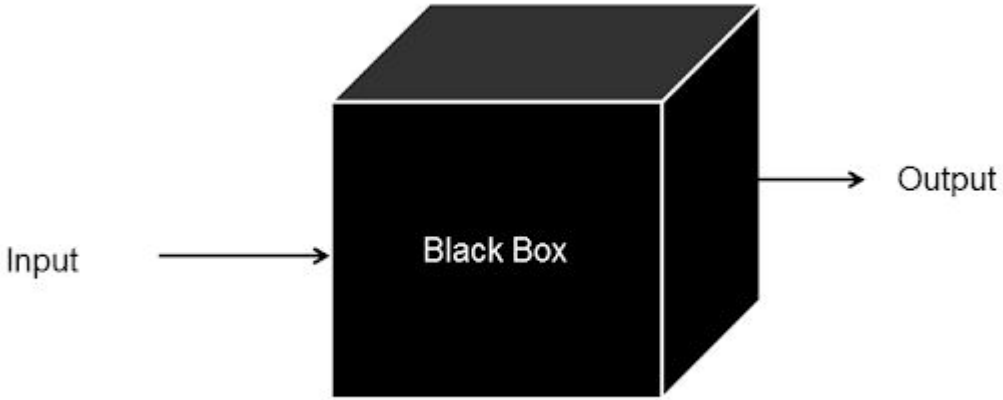
Datasets

- Pascal VOC: http://host.robots.ox.ac.uk/pascal/VOC/,
- Pascal Context: https://cs.stanford.edu/~roozbeh/pascal-context/
- MS COCO: https://cocodataset.org/#panoptic-2020
- Cityscapes: https://www.cityscapes-dataset.com/dataset-overview/

Timeline Algoritmos de segmentación

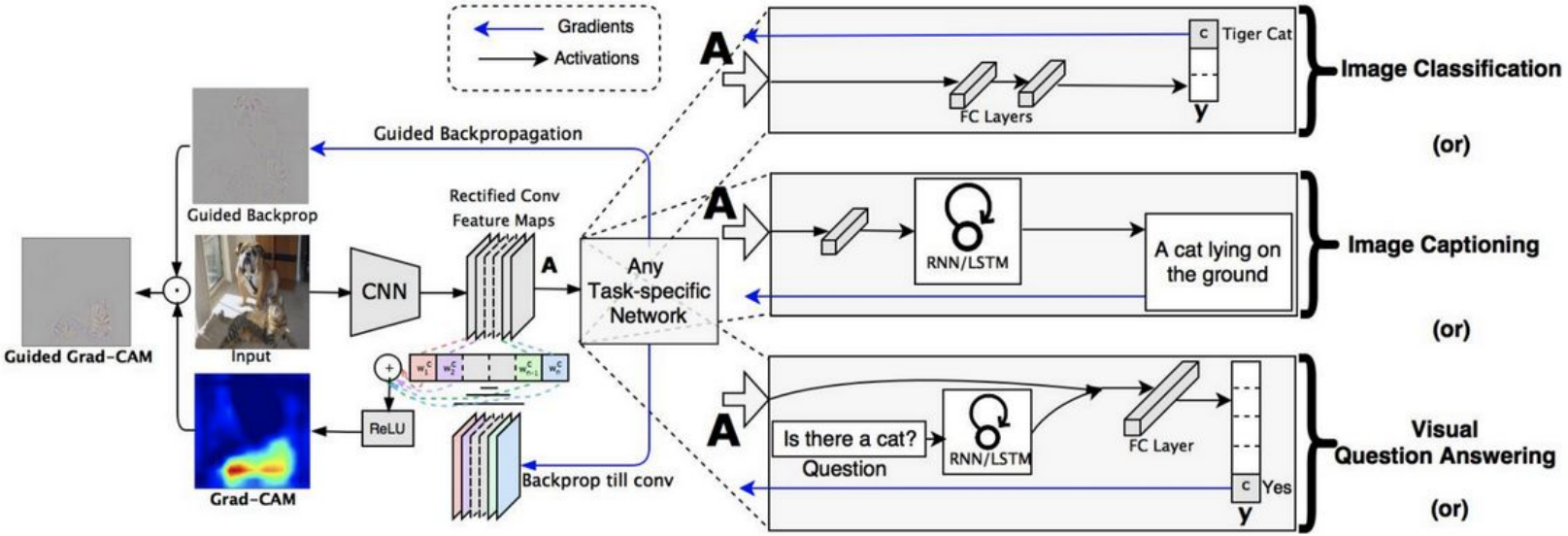


Sobre la explicabilidad de las redes convolucionales



Internal behavior of the code is unknown

Grad CAM



Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Link

Grad CAM

Método para analizar el funcionamiento de casi cualquier red convolucional.
Analiza qué activaciones son más relevantes para la activación de la salida (o alguna neurona intermedia)
Dada las activaciones de interés A(k) se calcula la importancia de cada canal del mapa de características dado para la salida de interés, y luego se calcula la combinación lineal de todos los mapas y sus importancias.
Generalmente se hace sobre las últimas capas que tienen un buen compromiso entre conservación espacial y contenido semántico.

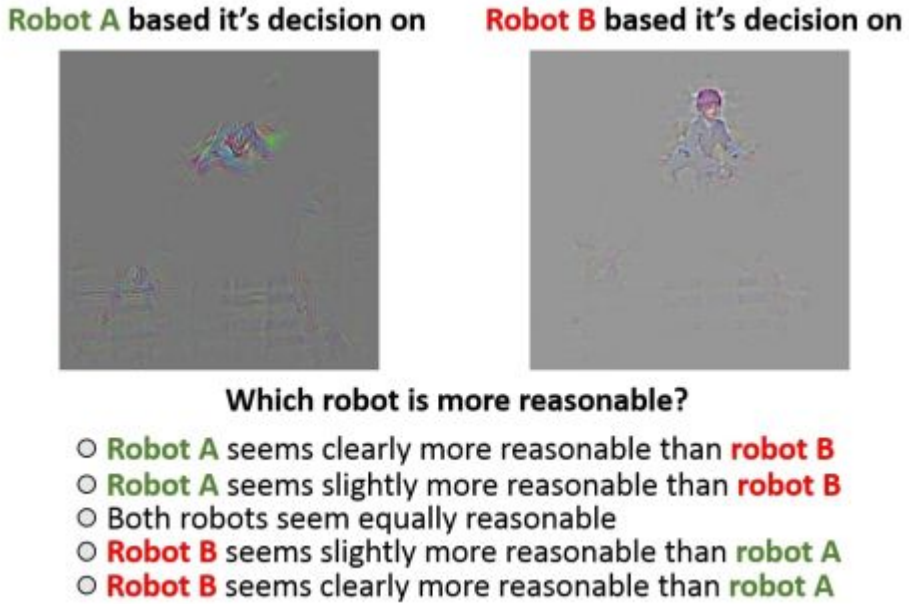
$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Link

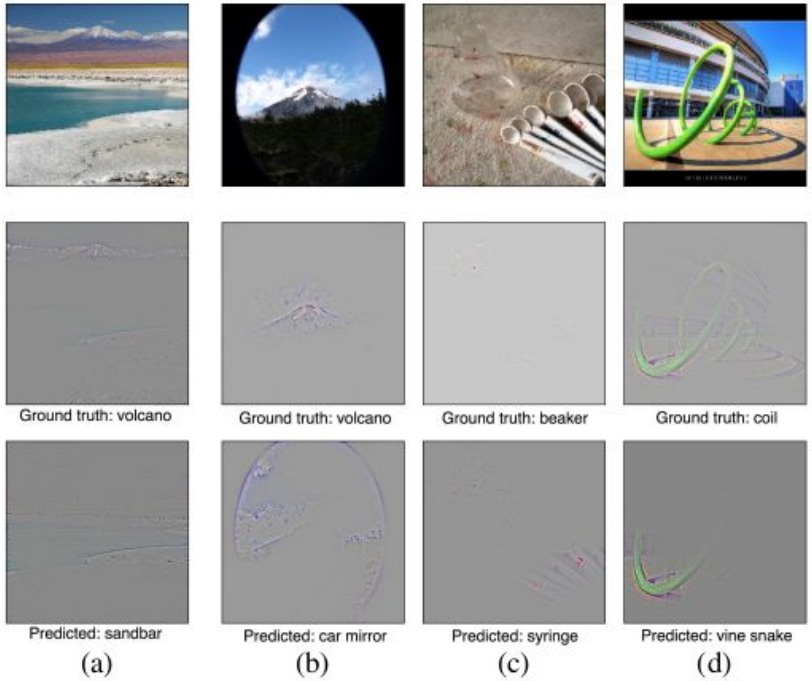
Grad CAM

Grad CAM: Comparación de modelos



Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Link

Grad CAM: Análisis de modo de fallas



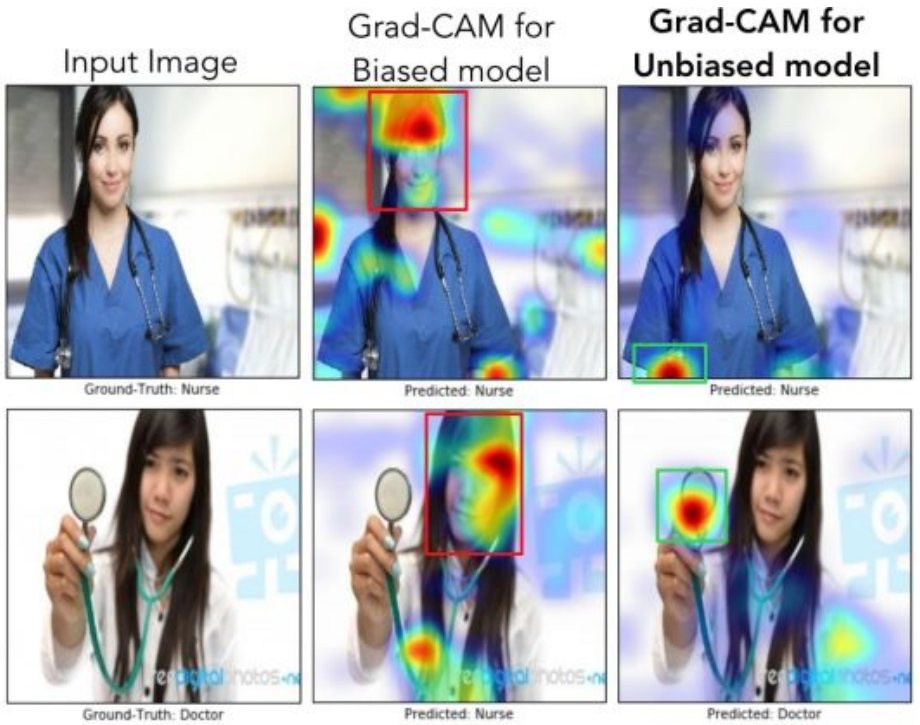
Se comparan los mapas de activación dados por Grad CAM de las clases correspondientes a los Ground Truth y a las salidas predicha, para los casos en los que el modelos dio falsas categorizaciones para las imágenes dadas.

Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Link

Grad CAM: Análisis de modo de fallas

Un problema muy frecuente en Deep Learning, y uno de los que más se le critica al área es el de los sesgos.

Con Grad CAM podrían analizarse los sesgos en los que incurre determinado modelo, para tomar acciones correctivas.



Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Link