



# Visión por Computador II

**CEAI, FIUBA**

Profesores:

- Javier A. Kreiner, [javkrei@gmail.com](mailto:javkrei@gmail.com)
- Andrés F. Brumovsky, [abrumov@gmail.com](mailto:abrumov@gmail.com)



## Quinta clase:

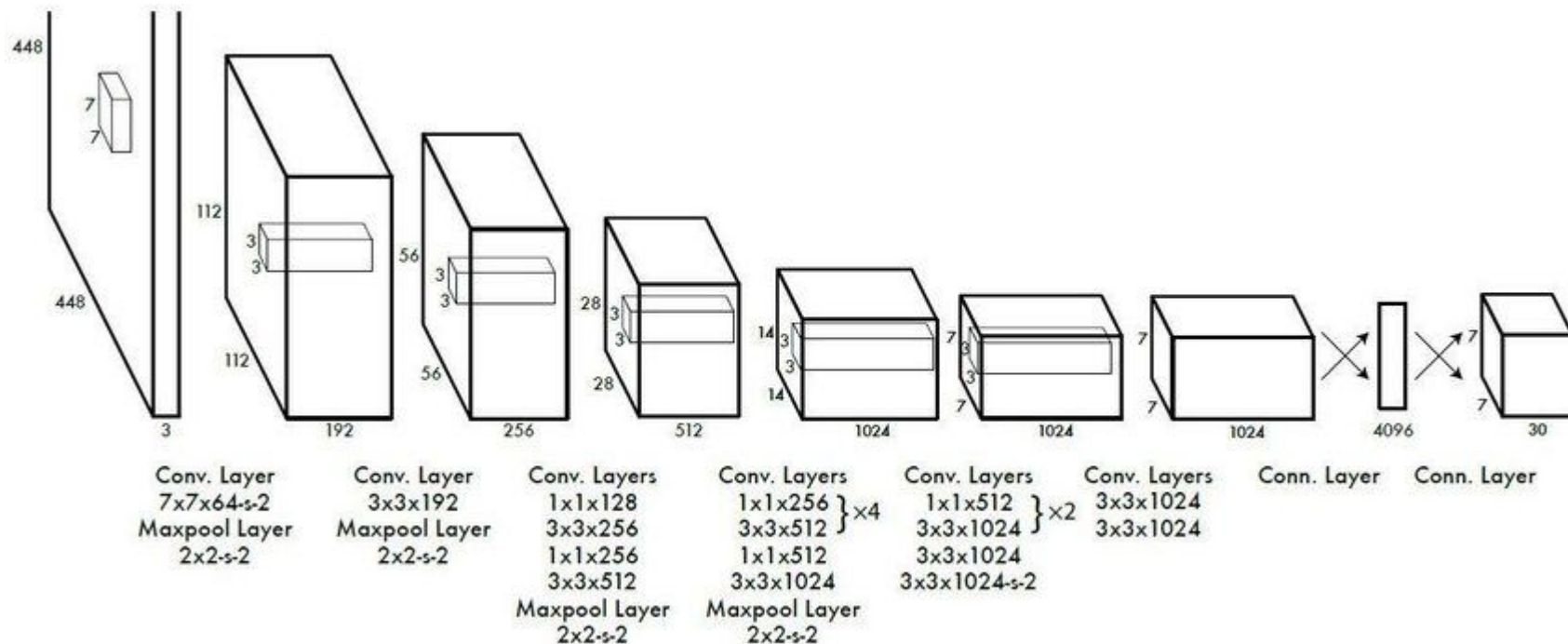
- Más algoritmo YOLO
  - Arquitectura
  - Ejemplo simple de programación
  - YOLO v2, v3, v4 y v5
  - Ejemplo de programación de YOLO v5
- Segmentacion de imagenes
- UNet
  - Arquitectura
  - Ideas
- Deeplab
  - Atrous Convolution
  - Atrous spatial pyramid pooling
  - Ejemplo de programación

# YOLO v1



- Red con 24 capas convolucionales seguidas de 2 capas fully connected. La red está inspirada en GoogLeNet
- Las primeras 20 capas son pre-entrenadas con Imagenet por una semana hasta lograr top-5 accuracy de 88% en el dataset de validación de ImageNet 2012
- Luego se modifica la red para realizar detección, se agregan cuatro capas convolucionales y dos capas fully connected.
- Al entrenar cada celda puede predecir varios objetos, se asigna la predicción al objeto con el cual tiene mayor IOU (usando el ground truth), esto lleva a la especialización de predictores, cada uno se vuelve mejor en predecir ciertos tamanhos, aspect ratios y clases de objetos, lo que mejora el recall
- Para data augmentation se introduce scaling aleatorio y translaciones de hasta 20% del tamaño de la imagen original. También se ajusta aleatoriamente la exposición y saturación de la imagen hasta un factor de 1.5 en el espacio de colores HSV
- También hacen una red llamada Fast YOLO que usa 9 en vez de 24 capas convolucionales, y menos filtros en cada capa

# Arquitectura de YOLO v1



# Limitaciones de YOLO v1



# Limitaciones de YOLO v1



- El algoritmo original utiliza una grilla de 7x7 y cada celda puede reconocer sólo dos objetos, entonces el máximo número de objetos es 98
- Esta limitación espacial limita el número de objetos cercanos que YOLO puede detectar, al modelo le cuesta detectar objetos pequeños que están en grupos, como una banda de pájaros, como aprende a predecir bounding boxes de los datos, le cuesta generalizar a configuraciones y aspect ratios inusuales
- Las características son bastante gruesas para predecir bounding boxes, hay mucho downsampling.
- La función de pérdida trata errores en una bounding box grande igual que en una bounding box chica
- La mayor fuente de error es por localización

[Paper original de YOLO](#)



## Ejemplo de construcción de yolo v1 simple:

- colab: [https://colab.research.google.com/drive/1PCqIip7wt3fZzm4vUtLUXGAqp\\_xD82hg](https://colab.research.google.com/drive/1PCqIip7wt3fZzm4vUtLUXGAqp_xD82hg)

(obtenido de: <https://www.maskaravivek.com/post/yolov1/>)

# YOLO v2

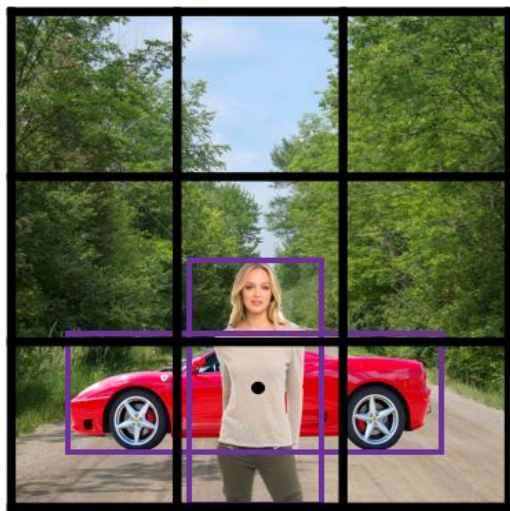


Mejoras:

- Batch Normalization en vez de dropout
- Clasificador de mayor resolución: el tamaño de la entrada fue aumentado de 224x224 a 448x448 (en un paso de tuning del entrenamiento)
- Anchor boxes: permiten detectar varios objetos en la misma celda. Se eligen utilizando un algoritmo de k-means clustering en el training set,  $k = 5$ , o sea que cada celda reconoce 5 objetos
- se aumenta la grilla a 13x13, lo que permite identificar y localizar objetos más pequeños
- entrenamiento multi-escala: si YOLO es entrenado con imágenes pequeñas de un objeto tiene dificultad de detectarlo cuando el objeto aparece más grande en la imagen. Para eso se entrena con imágenes aleatorias con diferentes dimensiones. Como el modelo sólo tiene capas convolucionales y de pooling la entrada puede ser redimensionada 'on the fly'. Como resultado la red es más robusta para detectar objetos en diferentes resoluciones
- Darknet 19: Utiliza la arquitectura Darknet 19 con 19 capas convolucionales, 5 max pool y softmax para clasificación. La arquitectura es muy veloz. (Darknet es una denominación paraguas para varias arquitecturas)

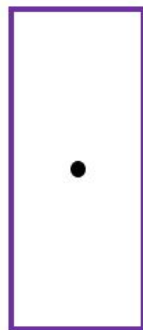
[Paper de YOLOv2](#)



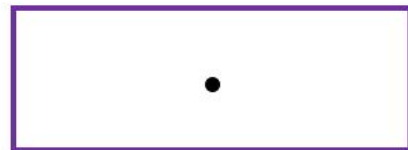



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



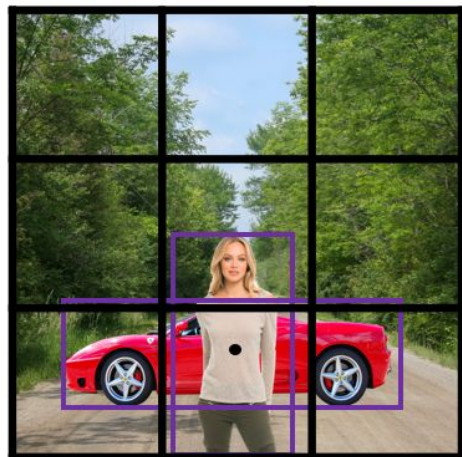
Anchor box 2:





Antes: cada objeto en la imagen de entrenamiento es asignado a una celda de la grilla que contiene el punto medio del objeto.

Ahora: Cada objeto en la imagen de entrenamiento es asignado a la celda de la grilla que contiene el centro del objeto y a la anchor box con la cual tiene mayor IoU.



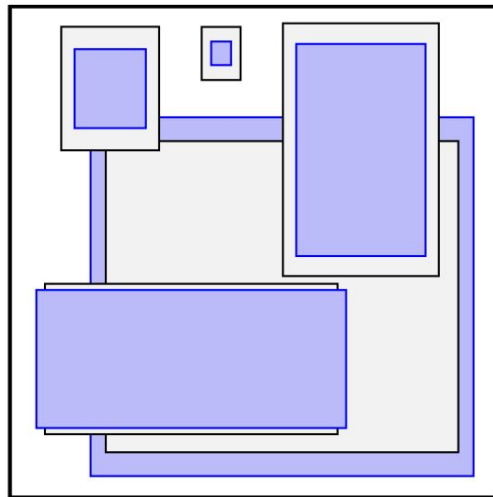
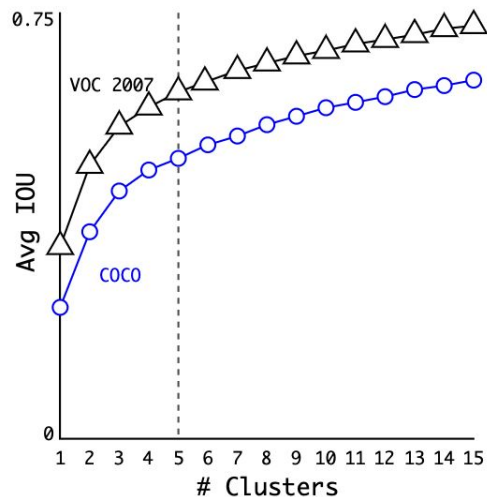
Anchor box 1:    Anchor box 2:



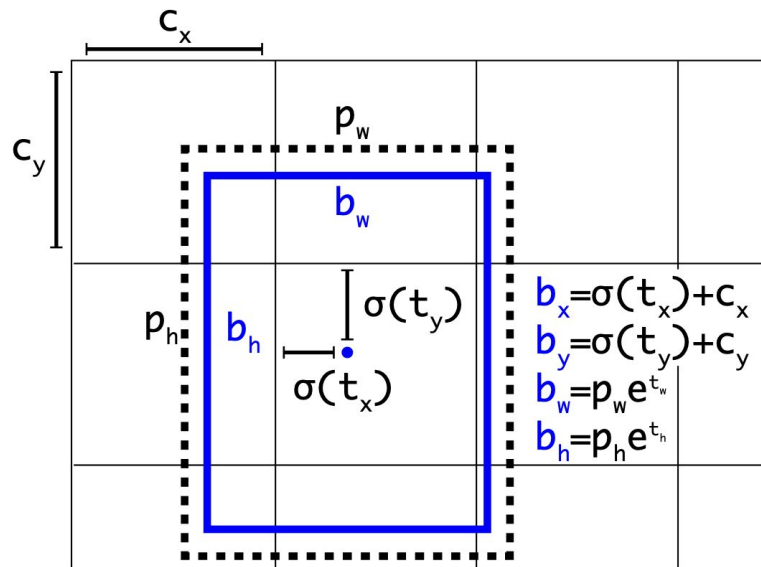
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

# Cómo se eligen las anchor boxes

- Se hace k-means clustering en el tamaño de las bounding boxes del dataset:



# Se predice la posición y ancho/altura del objeto



$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

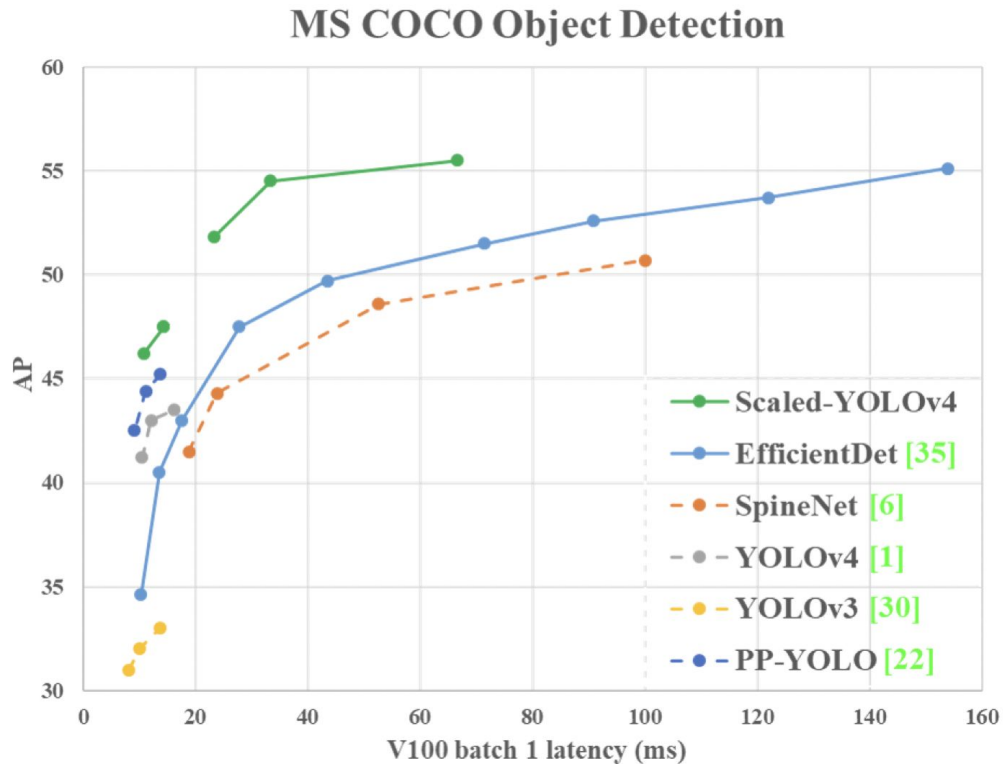
# YOLOv4 - Discusión de una arquitectura moderna



- Una arquitectura moderna tiene los siguientes componentes
  - Un backbone pre entrenado en ImageNet (VGG, ResNet, etc. para GPU, SqueezeNet, MobileNet, etc. para CPU)
  - Una cabeza usada para predecir clases y bounding boxes, puede ser de:
    - una etapa: YOLO, SSD, RetinaNet
    - dos etapas: R-CNN, fast R-CNN, faster R-CNN, etc.
  - Capas opcionales entre el backbone y la cabeza que se puede llamar el “cuello”. Usados para recolectar mapas de características (feature maps) de diversas etapas.

[Paper de YOLO v4](#)

# Scaled YOLO-v4 Benchmark



- [Paper Scaled YOLOv4](#)
- comparación de modelos:  
<https://paperswithcode.com/sota>



## YOLOv5 ejemplo de uso:

- Entrenamos un modelo de YOLOv5 con datos propios
- Roboflow (tips para labeling: <https://blog.roboflow.com/tips-for-how-to-label-images>)
- Extensiones de chrome útiles: Download All Images, Colab Alive
- Proceso de rotulación
- Exportación de datos
- Entrenamiento de un modelo customizado:  
<https://colab.research.google.com/drive/1ZnkOM8ymAWtCKj8AkQuLUaHAQPToLhKg?usp=sharing>



# De clasificación a detección a segmentación(semántica y de instancias)



Image Classification

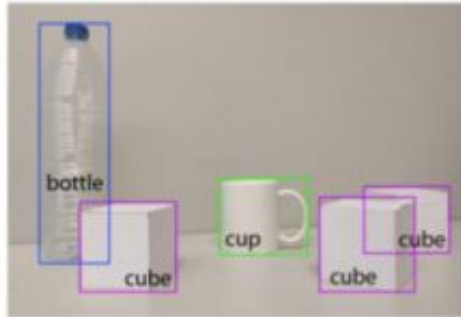


Image Detection

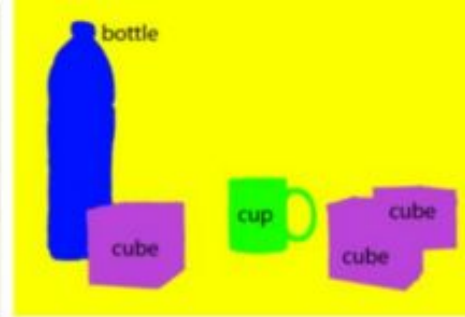
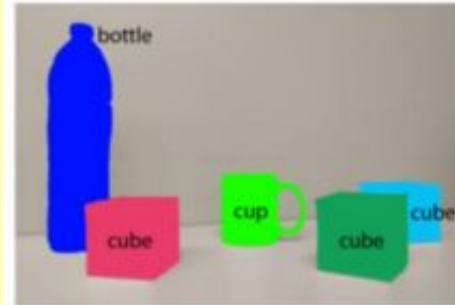


Image Segmentation



Instance Segmentation



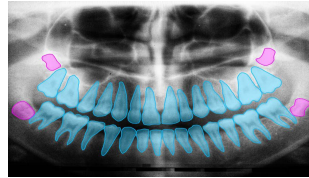
# Aplicaciones de image segmentation

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

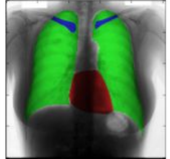
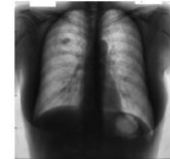
# Ejemplos

- 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=qWI9idsCuLQ>

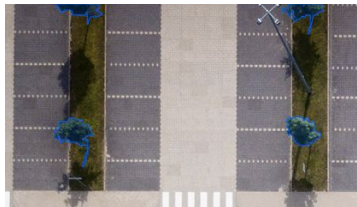
- Imágenes dentales:



- Imágenes médicas:



- Agricultura de precisión:

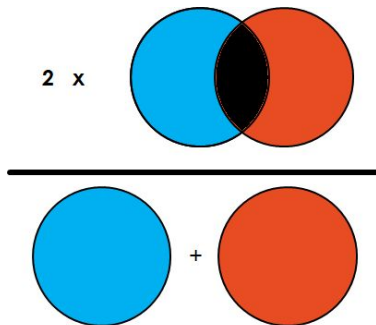


- Segmentación de residuos:



# Medidas de error

- pixel accuracy: porcentaje de píxeles en la imagen clasificados correctamente. Problemas: si hay mucho fondo por ejemplo puede ser muy mala (en general un problema si las clases están muy desbalanceadas)
- IoU promedio: se toma la IoU para cada clase en la imagen y se saca el promedio
- Coeficiente Dice o F1 score: 2 veces el área de la intersección dividida la suma de las áreas (parecida a la anterior)





# Datasets

- Pascal VOC: <http://host.robots.ox.ac.uk/pascal/VOC/>,  
<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/segexamples/index.html>
- Pascal Context: <https://cs.stanford.edu/~roozbeh/pascal-context/>
- MS COCO: <https://cocodataset.org/#panoptic-2020>
- Cityscapes: <https://www.cityscapes-dataset.com/dataset-overview/>

# UNet



- Es una red fully convolutional -> convierte capas fully connected en convolucionales (lo vimos en una de las clases anteriores). Permite aplicar la red a imágenes de diverso tamaño.
- La arquitectura tiene forma de U: 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: se realizan traslaciones, rotaciones, modificaciones de la escala de grises y **deformaciones elásticas aleatorias**
- Se usó para segmentación de estructuras neuronales y de células
- [Paper de UNet](#)



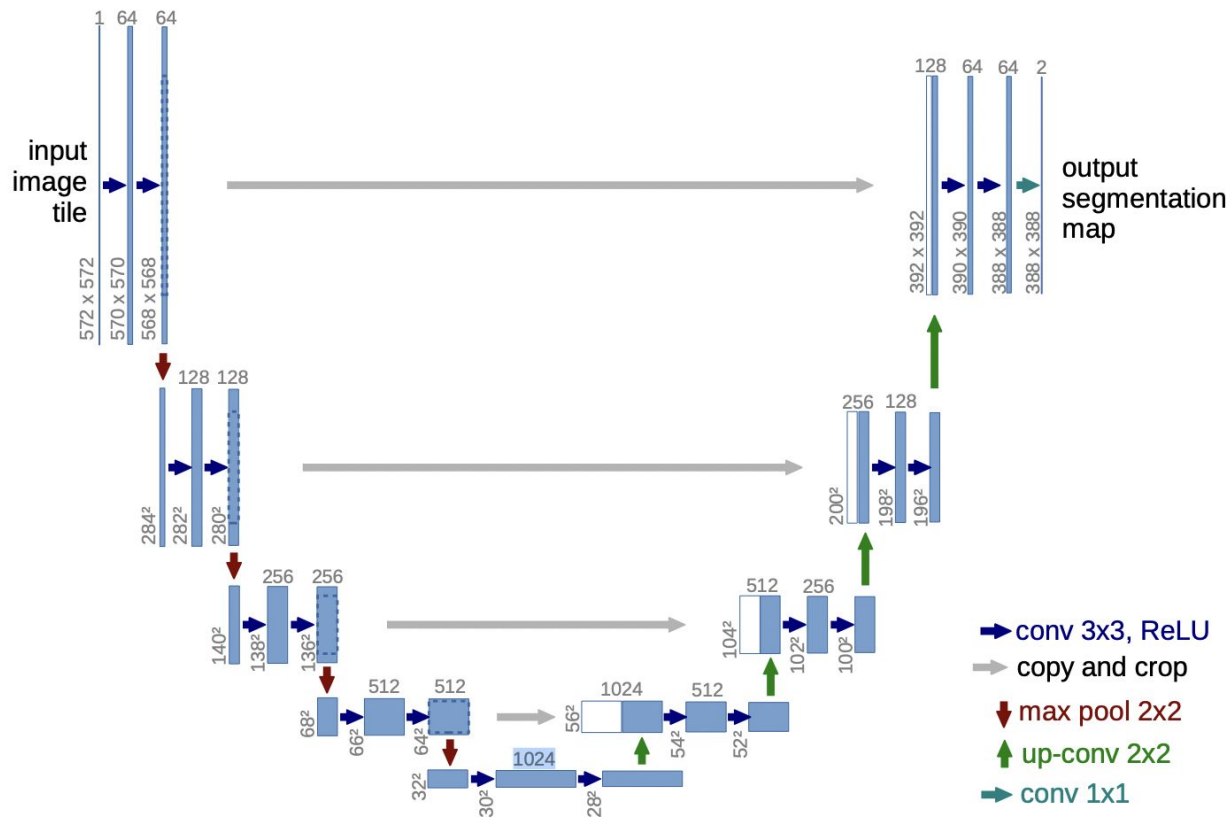
## Función de costo:

- La función de costo se calcula sumando la contribución de cada pixel:  $E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$
- $p_k(\mathbf{x})$  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  $\ell$  es el índice de la verdadera clase del píxel
- La función  $w(\mathbf{x})$  tiene dos propósitos.

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

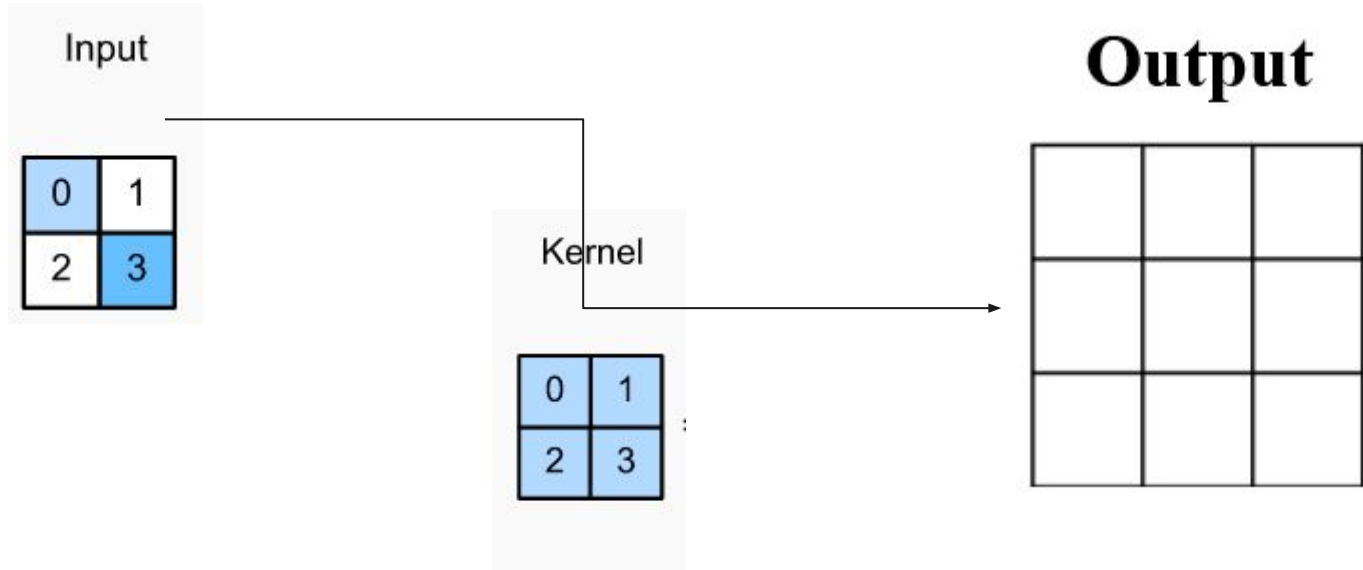
  - $w_c$  compensa el desbalance de clases
  - el segundo término le da más peso a píxeles que están cerca de bordes,  $d_1$  es la distancia a la celda más cercana y  $d_2$  a la segunda celda más cercana, da más peso a píxeles cerca de bordes

# Arquitectura

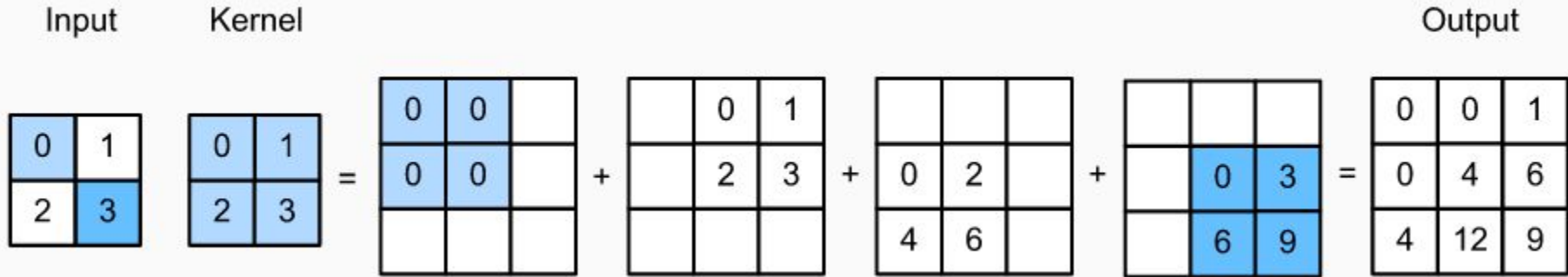




# Transposed convolution



# Transposed convolution





# Código para construcción de UNet

[https://colab.research.google.com/drive/1\\_-v-2n9SfkDe6vITX9IK4akGba88s8kL?usp=sharing](https://colab.research.google.com/drive/1_-v-2n9SfkDe6vITX9IK4akGba88s8kL?usp=sharing)

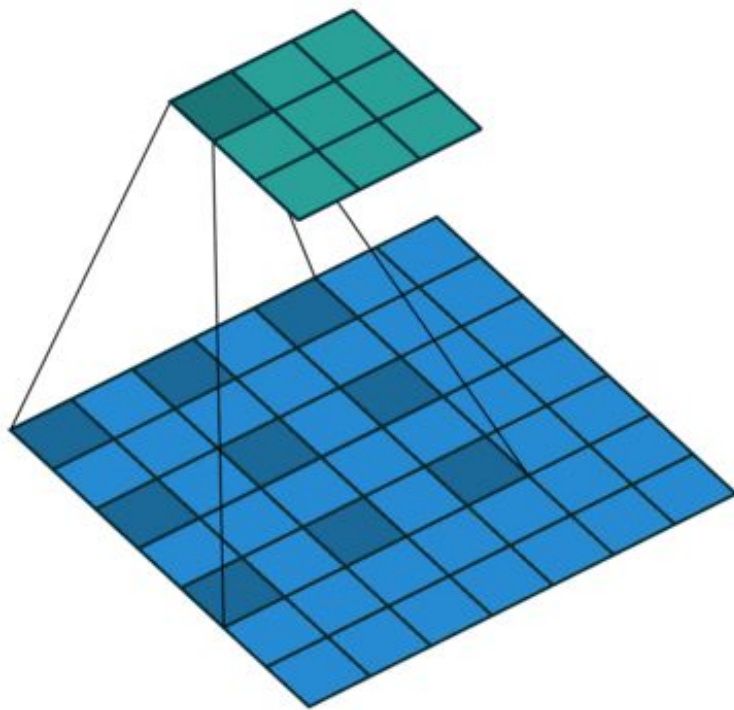
Fuentes: <https://www.kaggle.com/phoenigs/u-net-dropout-augmentation-stratification>,  
<https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5>



# 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 características 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





# 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





# Ejemplo de Deeplab para segmentación semántica

- colab:

<https://colab.research.google.com/drive/1dzhZ2V8fiDu-insxTXPxNF0xlfmnWAXg?usp=sharing>



# Discusión del trabajo final

- Competencias en kaggle:
  - <https://www.kaggle.com/c/siim-covid19-detection/overview>
  - <https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7/overview>
  - <https://www.kaggle.com/c/flower-classification-with-tpus>
  - <https://www.kaggle.com/c/bengaliai-cv19/leaderboard>
  - <https://www.kaggle.com/c/Kannada-MNIST>
  - [https://www.kaggle.com/c/understanding\\_cloud\\_organization/data](https://www.kaggle.com/c/understanding_cloud_organization/data)
  - <https://www.kaggle.com/c/severstal-steel-defect-detection/overview>
- Objetivo:
  - elegir una de estos datasets, considerar para eso el tamaño del dataset, cantidad de ejemplos de training, etc. y según la capacidad de cómputo que cada uno tenga disponible y el tiempo, abordar el que consideren adecuado
  - hacer una exploración de datos inicial breve
  - entrenar hasta 3 modelos como máximo, explicar las decisiones de diseño que se realizaron, documentar en un Jupyter Notebook lo realizado, evaluar desempeño.
- Grupos de entre 2 y 4 personas.