Intro to DL4MSc: Segmentation

Alexander Binder

August 18, 2025

Links

Slides adapted from Dhananjay Tomar

know where to look for

- https://d2l.ai/chapter_computer-vision/semantic-segmentation-and-dataset.html
- https://d2l.ai/chapter_computer-vision/transposed-conv.html
- https://d2l.ai/chapter_computer-vision/fcn.html

know what

- types of segmentation
- performance measures for semantic segmentation
- architectural design patterns for segmentation

Types of Segmentation Problems

Semantic Segmentation:

- for each pixel: Object Class labels, Background class labels

Instance Segmentation:

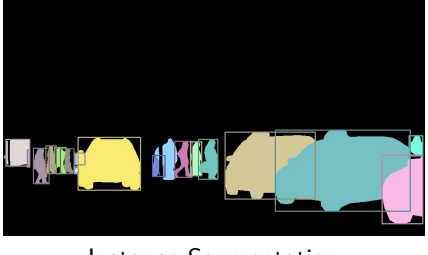
- Object Instance predictions / labels
- for each pixel: label of instance or background

Panoptic Segmentation:

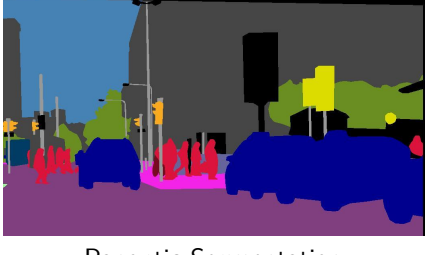
- Semantic Segmentation *AND*
- Instance Segmentation

Orig Image

Semantic Segmentation



Instance Segmentation



Panoptic Segmentation

- Classify every pixel in an image
- Differentiate between classes. Also: Background class
- Do not differentiate between multiple instances of the same class

img credit: Zhao et al 2017 https://arxiv.org/pdf/1704.08545.pdf

demo video: https://www.youtube.com/watch?v=qWl9idsCuLQ

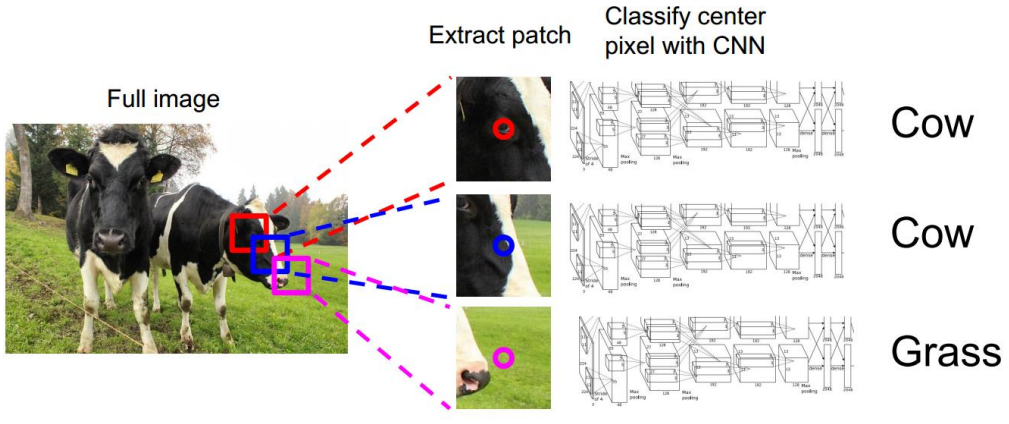
too simple approaches:

- sliding window classification of center pixel
- neural net with full resolution feature maps

better approaches:

- ensure large size receptive fields:
- Use information from early high resolution layers to capture finer details (boundary of the segmentation mask).

too simple approach 1: sliding window

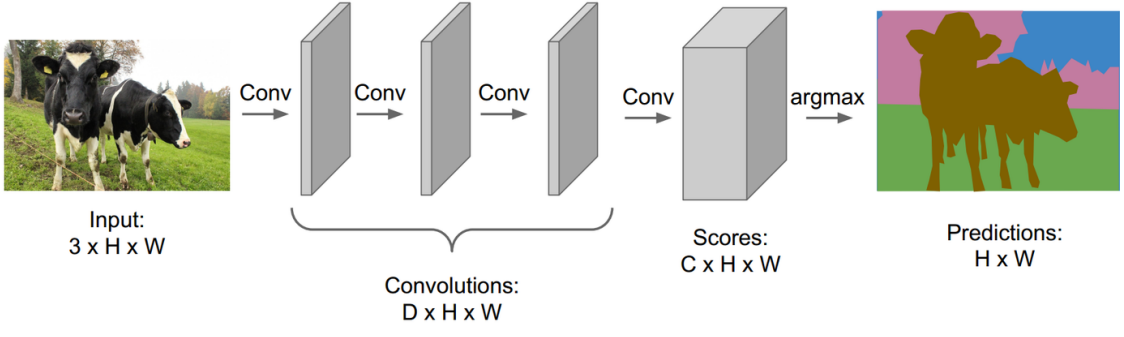


img credit: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture16.pdf

example for this approach: Ciresan et al., 2012: https://proceedings.neurips.cc/paper/2012/file/459a4ddcb586f24efd9395aa7662bc7c-Paper.pdf

Disadvantages: lacks global context, very expensive (pixels shared among patches processed repeatedly)

too simple approach 2: full resolution feature maps



img credit: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture16.pdf

Disadvantages: scales poorly
resource-eating for large images (or low receptive field if shallow), cannot use pretrained nets for init

Semantic Segmentation:

- IoU (aka Jaccard index)
- F1 score
- Pixel Accuracy

Semantic Segmentation:

- IoU (aka Jaccard index) per class
- Shape differs from boxes, but IoU can be calculated for any shape
- mostly reported
- binary problem and 1 single IoU measure: when the positive class matters alone
- mIoU: IoU averaged over all classes in a multi-class setting (multi-label?) - when all classes matter

Semantic Segmentation:

- F1 score
  - harmonic mean of precision and recall $F1 = \frac{1}{\frac{1}{2}Prec^{-1}+\frac{1}{2}Rec^{-1}}$
  - special case of p-means for $p = -1$, see:
  - general harmonic mean: $M^{(-1)}(s_0, \ldots, s_{n-1}) = (\frac{1}{n}\sum_{i=0}^{n-1} |s_i|^{-1})^{-1}$
  - general p-mean: $M^{(p)}(s_0, \ldots, s_{n-1}) = (\frac{1}{n}\sum_{i=0}^{n-1} |s_i|^{p})^{1/p}$
- properties ?
  - TN has no influence on its value! useful if only performance on class 1 is important, and if TN rates do not matter.
  - example: when one wants to find cancer tissue and do not care about the prediction quality on the other tissue.
- multiclass extension as average over classes like in mIoU possible

Semantic Segmentation:

- F1 score
  - harmonic mean of precision and recall $F1 = \frac{1}{\frac{1}{2}Prec^{-1}+\frac{1}{2}Rec^{-1}}$
- properties (2) ?
  - very low performance in one of the measures leads to strong penalization of the score:

$$\lim_{Prec \to 0} \frac{1}{\frac{1}{2}Prec^{-1} + \frac{1}{2}Rec^{-1}} = 0 \; (\to \frac{2}{\infty + Rec^{-1}})$$

$$\lim_{Rec \to 0} \frac{1}{\frac{1}{2}Prec^{-1} + \frac{1}{2}Rec^{-1}} = 0 \; (\to \frac{2}{Prec^{-1} + \infty})$$

$$F1 = \frac{1}{\frac{1}{2}Prec^{-1} + \frac{1}{2}Rec^{-1}} \leq \min(\frac{2}{Prec^{-1}+1}, \frac{2}{Rec^{-1}+1})$$

e.g. if the precision is never larger than 0.2, then F1 is upper bounded by $\frac{2}{6}$ no matter what the recall would be (compare for an arithmetic mean).

Semantic Segmentation:

- Pixel Accuracy

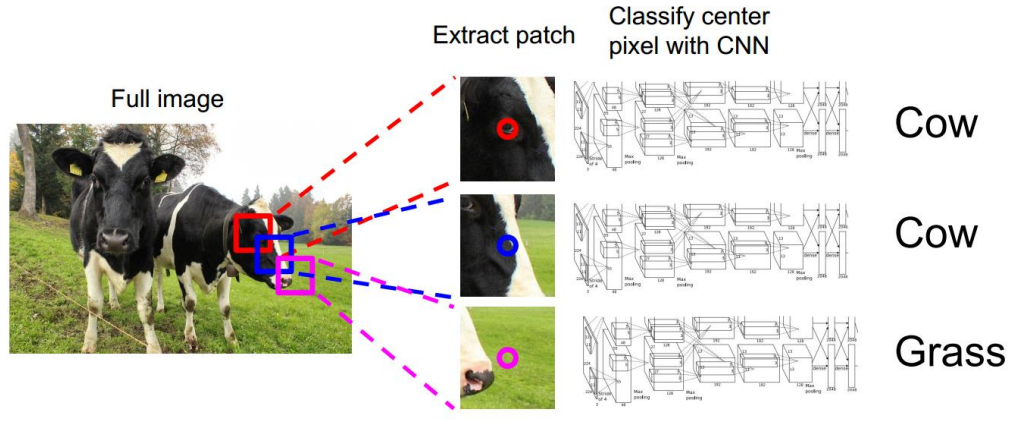
$$\frac{1}{HW}\sum_{i,j=1}^{H,W} 1[f(x_{ij}) == y_{ij}]$$

- average of accuracy over all pixels
- useful if performance on class 0 or negative class matters, too: high TN improves Pixel accuracy.
- example: if class 0 matters, too. More symmetric importance of positives and negatives like pixels of two colors in a product on a conveyor belt, or red and green chillis in grape sorting or a mix of gold and silver ore.

Semantic Segmentation:

- sum/average of classification losses over all pixels
- for example neg log of probability for the ground truth class in each pixel

too simple approach 1: sliding window

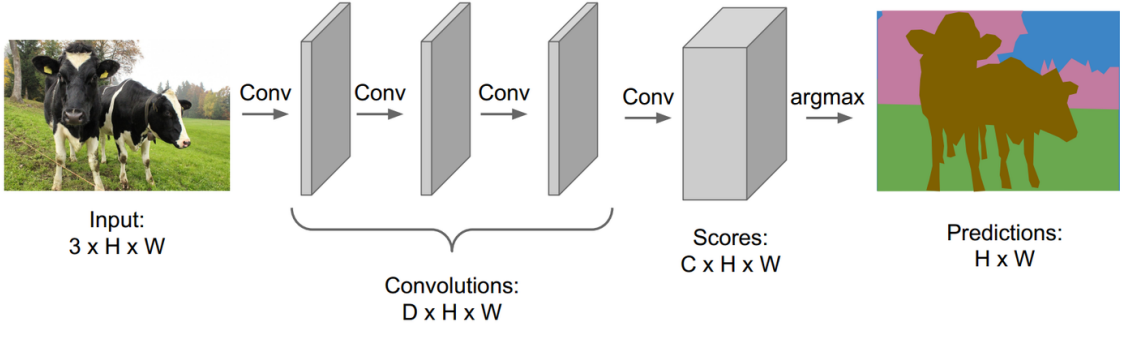


img credit: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture16.pdf

example for this approach: Ciresan et al., 2012: https://proceedings.neurips.cc/paper/2012/file/459a4ddcb586f24efd9395aa7662bc7c-Paper.pdf

Disadvantages: lacks global context, very expensive (pixels shared among patches processed repeatedly)

too simple approach 2: full resolution feature maps

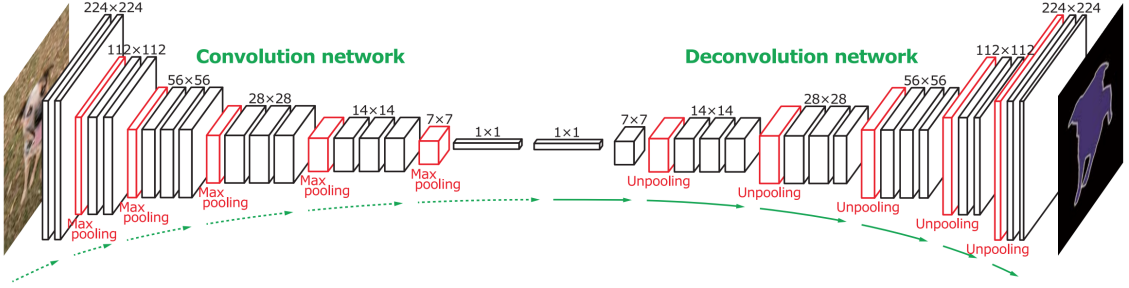


img credit: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture16.pdf

Disadvantages: scales poorly
resource-eating for large images (or low receptive field if shallow), cannot use pretrained nets for init

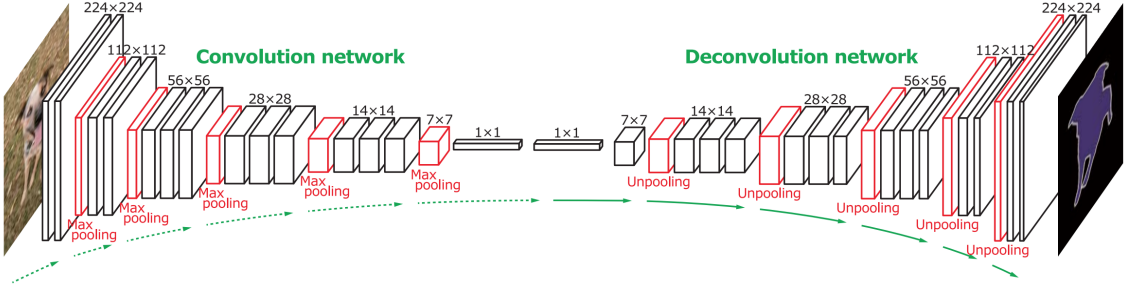
better approaches:

- ensure large size receptive fields:
  - downsample feature maps using conv/pooling with stride
  - dilated convolutions
    - problem: with large receptive field comes low res in feature maps
    - need to get back to image-level resolution
- Use information from early high resolution layers to capture finer details (boundary of the segmentation mask).
  - upsampling (NN-upsampling, bilinear upsampling, fractionally strided convolution)
  - fusion with early feature maps (U-Net, feature pyramids)



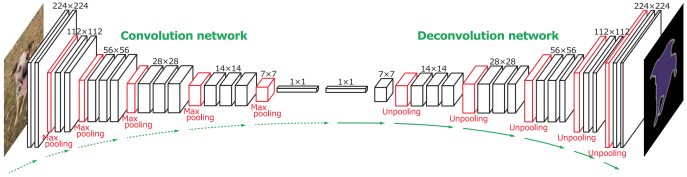
img credit: Noh et al, https://arxiv.org/pdf/1505.04366

- Global context: Information from the entire image to predict a pixel.
- Getting context has two common solutions
  - Downsample feature maps using max/sum/avg pooling or convolutions with stride > 1.
  - later in this lecture: Dilated convolutions
- for upsampling ("unpooling") see below

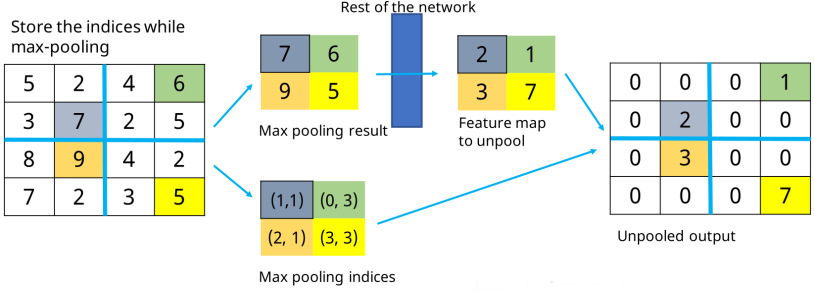


img credit: Noh et al, https://arxiv.org/pdf/1505.04366

- How to upsample? 3 common solutions
  - max unpooling
  - nearest neighbor upsampling, bilinear upsampling
  - fractionally strided convolution



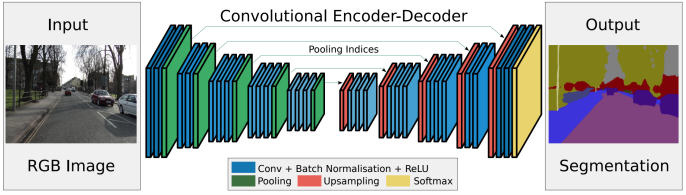
img credit: Noh et al, https://arxiv.org/pdf/1505.04366



max-unpooling Disadvantages:

- not learnable how to upsample
- 0 might be a bad fill value
- will need some convolution layers on top to make something smoother/useful from it

Architectural concepts: how to upsample?



img credit: Badrinarayanan et al, https://arxiv.org/pdf/1511.00561.pdf

1	2
3	4

3x

1	1	1	2	2	2
1	1	1	2	2	2
1	1	1	2	2	2
3	3	3	4	4	4
3	3	3	4	4	4
3	3	3	4	4	4

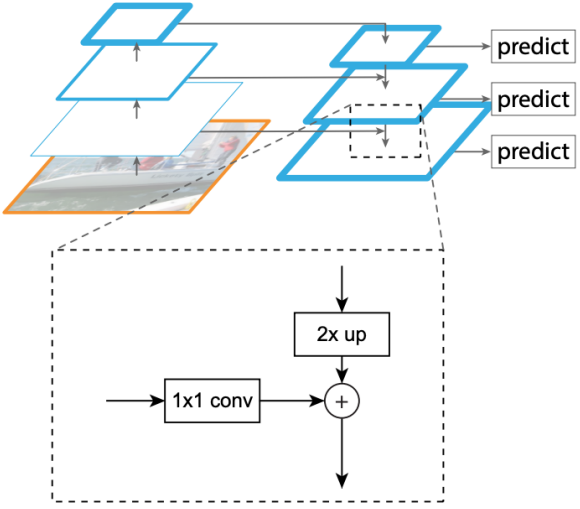
Input: C x 2 x 2

Output: C x 6 x 6

NN-upsampling Disadvantages:

- not learnable how to upsample
- will need some convolution layers on top to make something smoother/useful from it

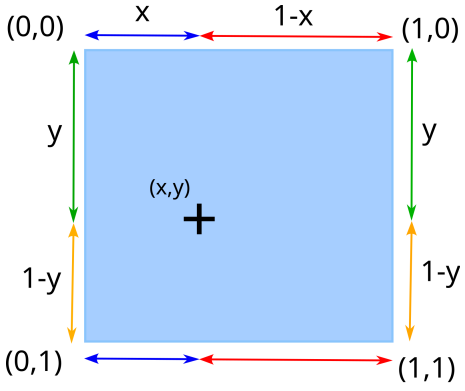
NN-upsampling also used in FPN for segmentation: https://arxiv.org/pdf/1612.03144



img credit:Lin et al, https://arxiv.org/pdf/1612.03144

bilinear interpolation:

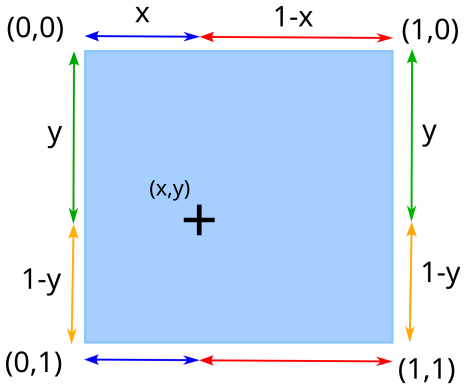
- Let us use a rectangle spanned by normalized coordinates $(0, 0)$ and $(1, 1)$.
- we have values at the corners $f(0, 0)$, $f(0, 1)$, $f(1, 0)$, $f(1, 1)$
- we have a point (x_n, y_n) in the rectangle spanned by $[0, 1] \times [0, 1]$ such that
- Then $x_n \in [0, 1]$ and $y_n \in [0, 1]$



$$\text{interpolate along } x: \quad \hat{f}[x]_{y=0} = (1-x)f(0,0) + xf(1,0)$$
$$\hat{f}[x]_{y=1} = (1-x)f(0,1) + xf(1,1)$$
$$\text{interpolate along } y: \quad \hat{f}[x,y] = (1-y)\hat{f}[x]_{y=0} + y\hat{f}[x]_{y=1}$$

bilinear interpolation:

- Let us use a rectangle spanned by normalized coordinates $(0, 0)$ and $(1, 1)$.
- we have values at the corners $f(0, 0)$, $f(0, 1)$, $f(1, 0)$, $f(1, 1)$
- we have a point (x_n, y_n) in the rectangle spanned by $[0, 1] \times [0, 1]$ such that
- Then $x_n \in [0, 1]$ and $y_n \in [0, 1]$



combine both interpolations along x and along y:

$$\hat{f}[x, y] = (1 - x)(1 - y)\, f(0, 0) + (1 - x)y\, f(0, 1) + x(1 - y)\, f(1, 0) + xy\, f(1, 1)$$

- $coord = 0$ receives $1 - coord$, $coord = 1$ receives $coord$, where $coord \in \{x, y\}$
  - bcs must have $weight = 1$ at its fringe value ;)

bilinear interpolation:

- next step: map unnormalized coordinates (x, y) in a grid between (x_{min}, y_{min}),(x_{max}, y_{max}) to normalized coordinates (x_n, y_n):

$$x_{min} \longmapsto x_n = 0, \; x_{max} \longmapsto x_n = 1, \; \text{linearly}$$
$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}}$$
$$y_{min} \longmapsto y_n = 0, \; y_{max} \longmapsto y_n = 1, \; \text{linearly}$$
$$y_n = \frac{y - y_{min}}{y_{max} - y_{min}}$$

- Then continue as above with (x_n, y_n)

Fractionally strided convolution / transposed convolution: See Fig 14.10.1

https://d2l.ai/chapter_computer-vision/transposed-conv.html

- does not compute an inner product
- computes a multiplication of a real number and a kernel. Adds the kernel-shaped result into the output
- combines upsampling and convolution-type weights, trainable

`class ConvTranspose2d` in PyTorch
https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html

We can upsample feature maps approximately

Architectural concepts: fusing different feature map resolutions and usage of global context

Two problems:

- Incorporating different feature map resolutions using Upsampling, Concatenations and Convolutions: Example U-Net
- incorporating global context: Atrous Convolutions, Example: DeepLabV3+

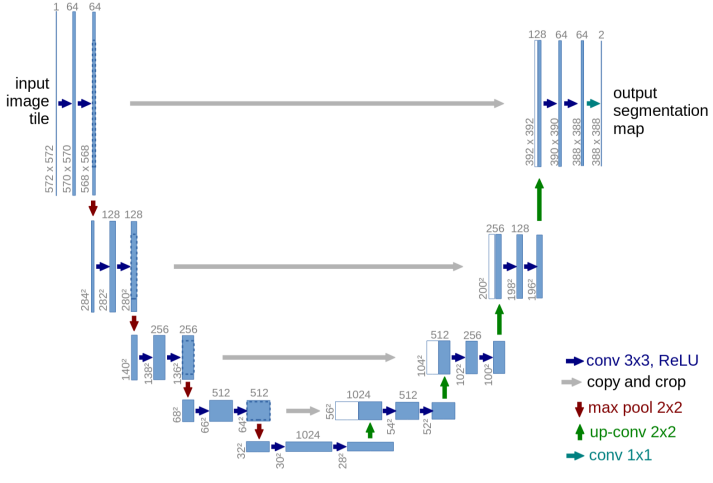
Incorporating different feature map levels

Problem: How to fetch precise boundary locations ? Will be difficult if one uses only low-resolution feature maps close to the output of a classification network

- feature maps close to the output of a classification network encode information about high-level concepts, at low spatial resolution
- feature maps close to the input of a classification network encode information about low-level concepts (like textures, small parts), at high spatial resolution
  - one solution is to use FPN. Below show another common one: U-Net type architectures

Incorporating different feature map levels

Ronneberger et al. https://arxiv.org/pdf/1505.04597

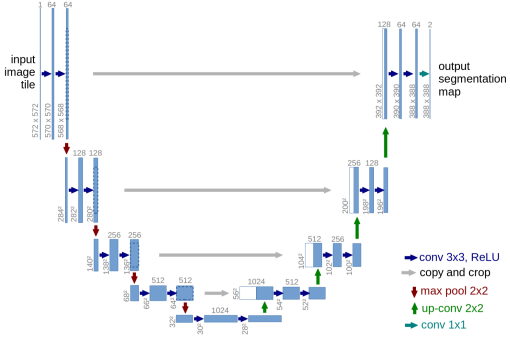


img source https://arxiv.org/pdf/1505.04597

many variants like attention-U-Net, residual U-net, etc https://code.likeagirl.io/u-net-vs-residual-u-net-vs-attention-u-net-vs-attention-residual-u-net-899b57c5698

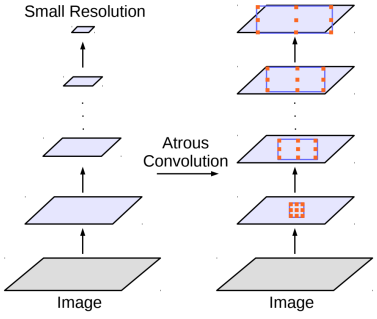
Incorporating different feature map levels

Ronneberger et al. https://arxiv.org/pdf/1505.04597

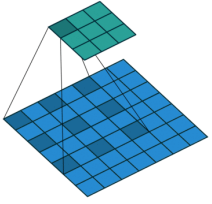


img source https://arxiv.org/pdf/1505.04597

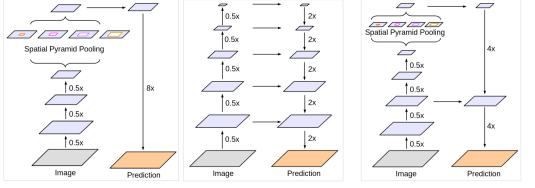
- (Concat+ convolution) learns to weight addition of
  - the copied channels from close to the input
  - the upsampled channels



img credit: Chen et al. https://arxiv.org/pdf/1802.02611



- Standard convolution is a special case of atrous convolution with dilation rate $r = 1$
- $r > 1$: read every r-th pixel in the input for computing the inner product with a given kernel
- Atrous convolution with $r > 1$ allows to capture larger spatial context
- DeepLabv3+ (next slide) applies atrous convolutions with different rates in parallel on the last feature map and concatenates them before upsampling and making pixel-level predictions.



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

Fig. 1. We improve DeepLabv3, which employs the spatial pyramid pooling module (a), with the encoder-decoder structure (b). The proposed model, DeepLabv3+, contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows us to extract features at an arbitrary resolution by applying atrous convolution.

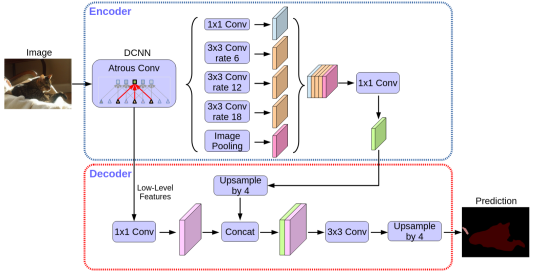
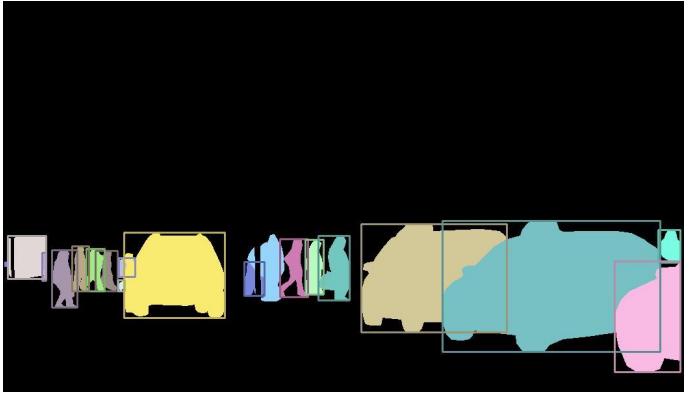


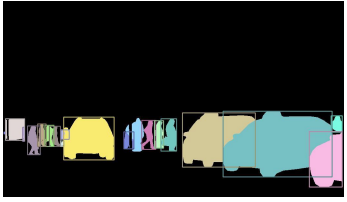
Fig. 2. Our proposed DeepLabv3+ extends DeepLabv3 by employing a encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.

img credit: Chen et al. https://arxiv.org/pdf/1802.02611



Instance Segmentation:

- Object Instance predictions / labels
- for each pixel: label of instance $\in [1, \ldots, O_{max}]$ or background label (e.g. 0)

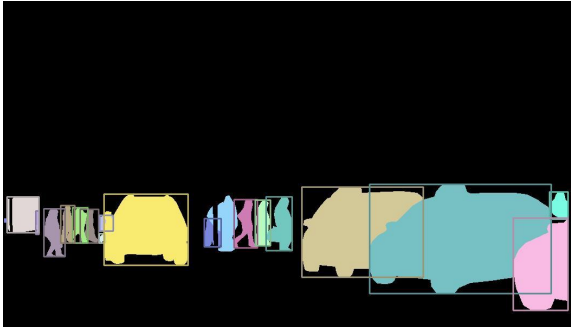


How to measure performance ?

- panoptic segmentation: mask AP / mask mAP
- difference to AP/ mAP used in object detection?
  https://github.com/rafaelpadilla/Object-Detection-Metrics

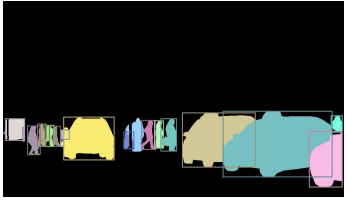
Instance Segmentation:

- have multiple segmented instances, similar to multiple boxes in object detection! Shape differs from boxes, but IoU can be calculated for any shape.
- Average Precision for a single class
- mAP as mean of Average Precisions over all classes of interest
- for segmentation: adjust IoU threshold if the object is very thin / has low ratio of volume to minimal enclosing convex shape like https://en.wikipedia.org/wiki/Box_jellyfish or https://en.wikipedia.org/wiki/Salp#/media/File:Sea_Salp_Chain.jpg or https://www.mineralienatlas.de/lexikon/index.php/MineralData?mineral=Millerit



How to measure performance ?

- pure instance segmentation without class labels: mean IoU over all instance labels
- caveat! instance labels are orderless, undetermined up to permutation of label values.
- Have to evaluate best match / or check all permutations



4. Experiments: Instance Segmentation

We perform a thorough comparison of Mask R-CNN to the state of the art along with comprehensive ablations on the COCO dataset [28]. We report the standard COCO metrics including AP (averaged over IoU thresholds), AP_{50}, AP_{75}, and AP_S, AP_M, AP_L (AP at different scales). Unless noted, AP is evaluating using *mask* IoU. As in previous work [5, 27], we train using the union of 80k train images and a 35k subset of val images (`trainval35k`), and report ablations on the remaining 5k val images (`minival`). We also report results on `test-dev` [28].

5

credit: https://arxiv.org/pdf/1703.06870

How to measure performance ?

- panoptic segmentation: mask AP / mask mAP
- difference to AP/ mAP used in object detection? https://github.com/rafaelpadilla/Object-Detection-Metrics
  - step 1 – use IoU threshold to determine TP and FP: compute IoU not between bounding boxes (gt vs predicted) but between ground truth instance segmentation mask and predicted instance mask
  - step 1 – IoU between ground truth instance segmentation mask and predicted instance masks
  - step 2 – varying class confidence threshold, as before

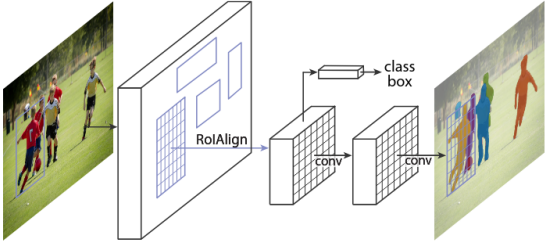


Figure 1. The **Mask R-CNN** framework for instance segmentation.

credit: https://arxiv.org/pdf/1703.06870

How to predict ?

- Example paper: mask R-CNN
  https://arxiv.org/pdf/1703.06870

How to predict ? Example paper: mask R-CNN
https://arxiv.org/pdf/1703.06870

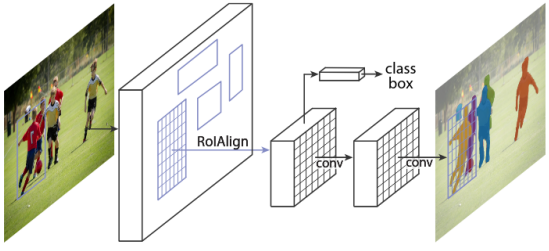


Figure 1. The **Mask R-CNN** framework for instance segmentation.

credit: https://arxiv.org/pdf/1703.06870

- start with a 2-stage object detection system
- Fig 1: stage 1 predict ROIs
- for each ROI: predict object bounding boxes as before
- for each ROI: predict something on a "synthetic" $m \times m$ grid
- predict for each $(i, j) \in grid(m \times m)$ a class label: background (0) ?
  ... or class label of the instance ?

How to predict ? Example paper: mask R-CNN
https://arxiv.org/pdf/1703.06870

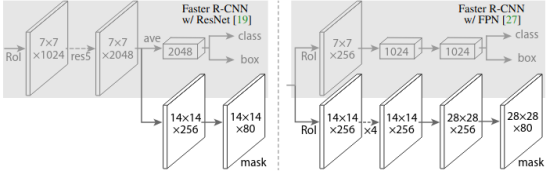


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively,

credit: https://arxiv.org/pdf/1703.06870

- start with a 2-stage object detection system
- Fig 1: stage 1 predict ROIs
- for each ROI: predict object bounding boxes as before
- for each ROI: predict something on a "synthetic" $m \times m$ grid
- predict for each $(i, j) \in grid(m \times m)$ a class label: background (0) ?
  ... or class label of the instance ?

How to predict ? Example paper: mask R-CNN
https://arxiv.org/pdf/1703.06870

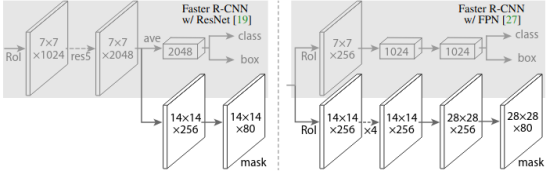


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively,

credit: https://arxiv.org/pdf/1703.06870

- start with a 2-stage object detection system
- Fig 1: stage 1 predict ROIs
- for each ROI: predict object bounding boxes as before
- for each ROI: predict something on a "synthetic" $m \times m$ grid
- "synthetic" $m \times m$ grid: each $(i, j) \in grid(m \times m)$ is not 1 pixel in the image.
  - Depends on size of the bounding box $(center_{box,h}, center_{box,w}, h_{box}, w_{box})$:
  - each grid point (i, j) has size: $h_{box}/m, w_{box}/m$

How to predict ? Example paper: mask R-CNN
https://arxiv.org/pdf/1703.06870

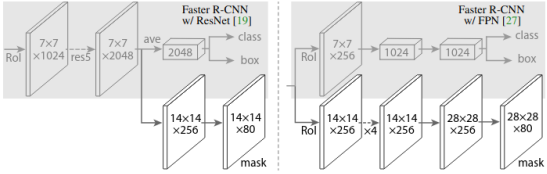


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively,

credit: https://arxiv.org/pdf/1703.06870

- start with a 2-stage object detection system
- for each ROI: predict object bounding boxes as before
- for each ROI: predict on a "synthetic" $m \times m$ grid
- "synthetic" $m \times m$ grid: each $(i, j) \in grid(m \times m)$ is not 1 pixel in the image.
  - each grid point (i, j) has size: $h_{box}/m, w_{box}/m$
  - ROIAlign computes the feature for a synthetic pixel using bilinear interpolation over a feature map: https://erdem.pl/2020/02/understanding-region-of-interest-part-2-ro-i-a

How to train ? Example paper: mask R-CNN
https://arxiv.org/pdf/1703.06870

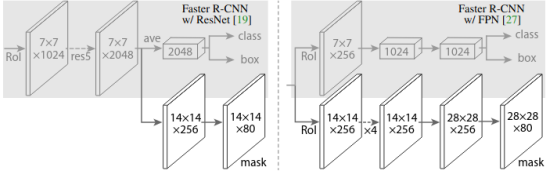


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively,

credit: https://arxiv.org/pdf/1703.06870

$$Loss = L_{cls} + L_{box} + L_{mask}$$

- usual object detection loss $L_{cls} + L_{box}$
- if L_{box} is added, then add L_{mask}
- L_{mask} is a binary cross-entropy loss, summed for each of the $(i, j) \in grid(m \times m)$ synthetic pixels

Formally, during training, we define a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$. The classification loss L_{cls} and bounding-box loss L_{box} are identical as those defined in [12]. The mask branch has a Km^2-dimensional output for each RoI, which encodes K binary masks of resolution $m \times m$, one for each of the K classes. To this we apply a per-pixel sigmoid, and define L_{mask} as the average binary cross-entropy loss. For an RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask (other mask outputs do not contribute to the loss).

credit: https://arxiv.org/pdf/1703.06870

1h lecture for watching at home:

https://www.youtube.com/watch?v=9AyMR4IhSWQ