Deep Learning

Lecture 7

Recap

Architectures for classification

- ResNet
- VGG
- MobileNet
- EfficientNet
- ViT

Object detection

- RCNN
- Fast-RCNN
- Faster-RCNN

Semantic segmentation

Motivation

We have cars, buildings on both images, but scenes are different!

- So, the image classification is not enough to represent the scene.
- Object detection is not enough. We can have object on different background

What is the solution?

Semantic segmentation

Solution is semantic segmentation. Through semantic segmentation we really understand the scene not only an image but also a video. We label every pixel of image with some class (e.g. car or pedestrian). In the end, we get the image where pixel have some class label.

How to make semantic segmentation

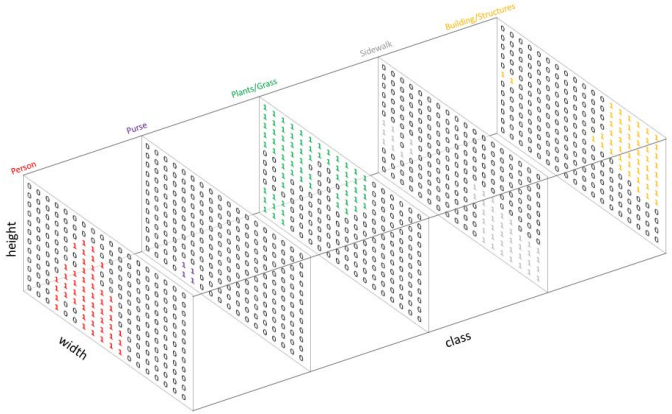
Input

segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Semantic Labels

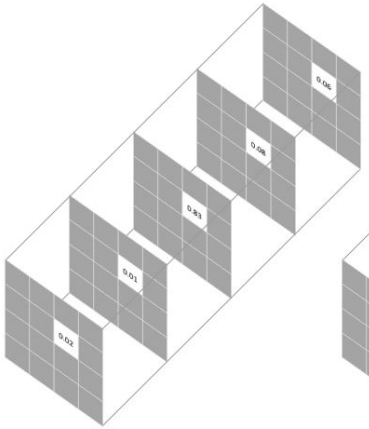


So, the output is map where we assign label to every pixel -> Output HxWx1.

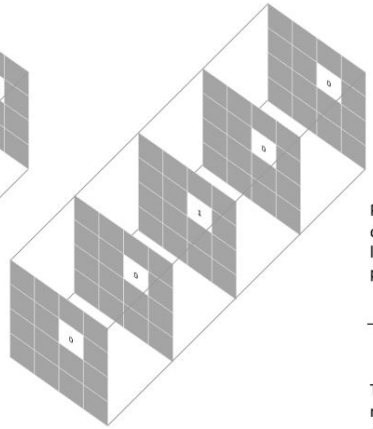
But neural network work poorly with discrete values? How to parametrize predictions?

Loss functions

How to train semantic segmentation model?



Prediction for a selected pixel

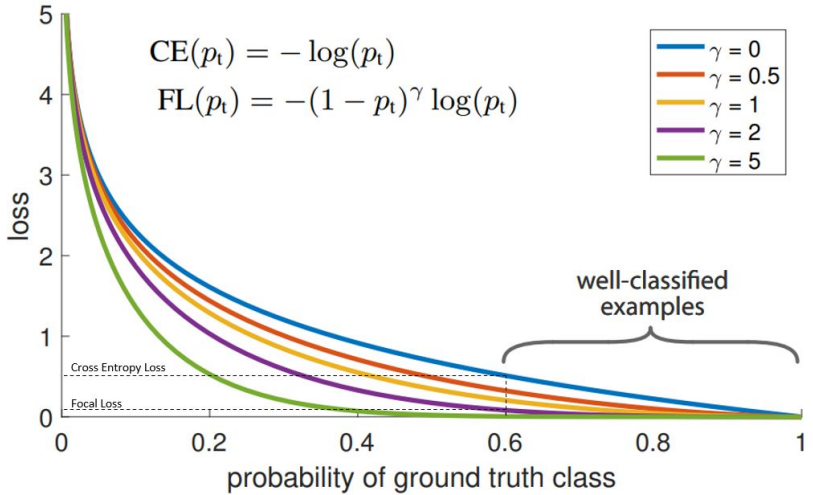


Target for the corresponding pixel

Pixel-wise loss is calculated as the log loss, summed over all possible classes

$$-\sum_{classes} y_{true} \log\left(y_{pred}\right)$$

This scoring is repeated over all **pixels** and averaged



$$\mathrm{FL}\left(p_t\right) = -\alpha_t \left(1-p_t\right)^{\gamma} \log\left(p_t\right)$$

Why do we need alphas?

Metrics

We measure quality of semantic segmentation in two ways as on detection task:

- How accurately we detect object?
- How precise our prediction?

Intersection

Union

$$IoU = \frac{\text{Intersection}}{\text{Union}}$$

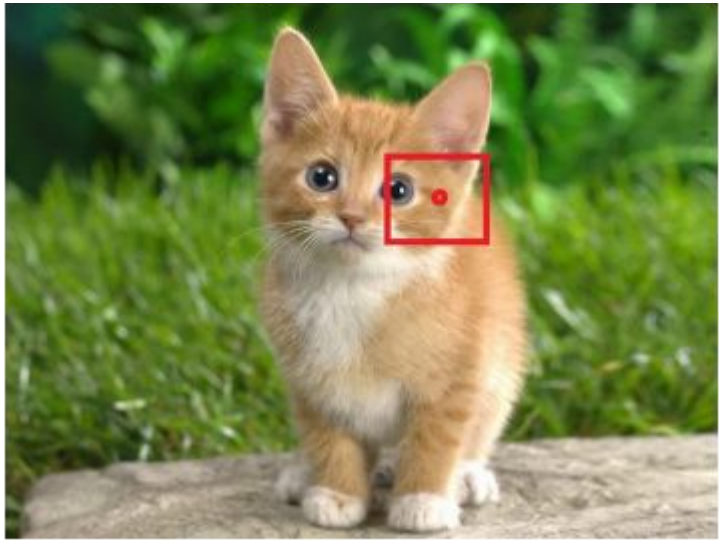
Averaging over classes → $mIoU$

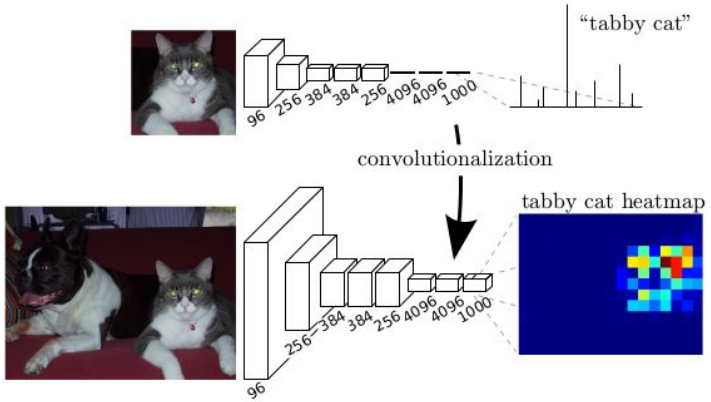
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Averaging over classes → $mAcc$

Simple solution

How to transform classification model into segmentation model?



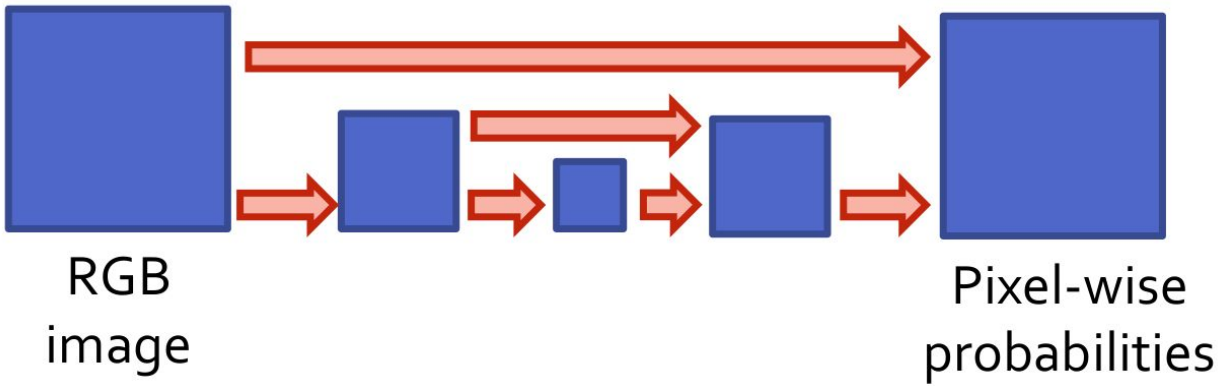


Lets, get prediction by using sliding window and constructing semantic map

- Works too long
- Resolution is very bad

Better solution

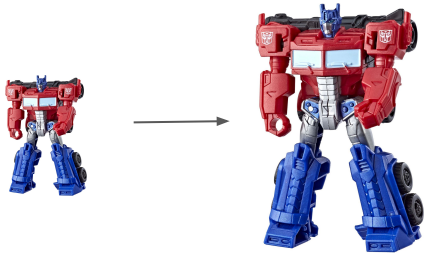
If we have a problem with low resolution so let's resolution as high as possible. But how we can do it?



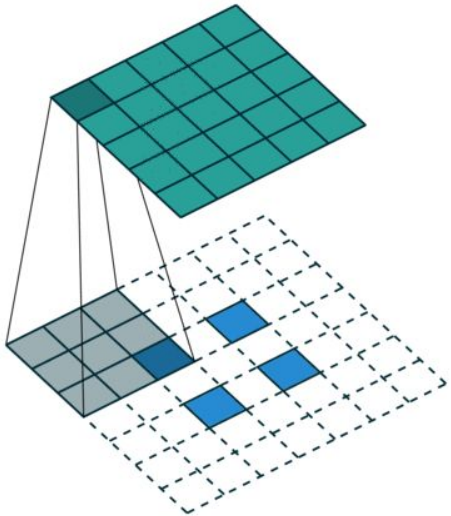
Encoder-Decoder approach (like autoencoder) solves both problems:

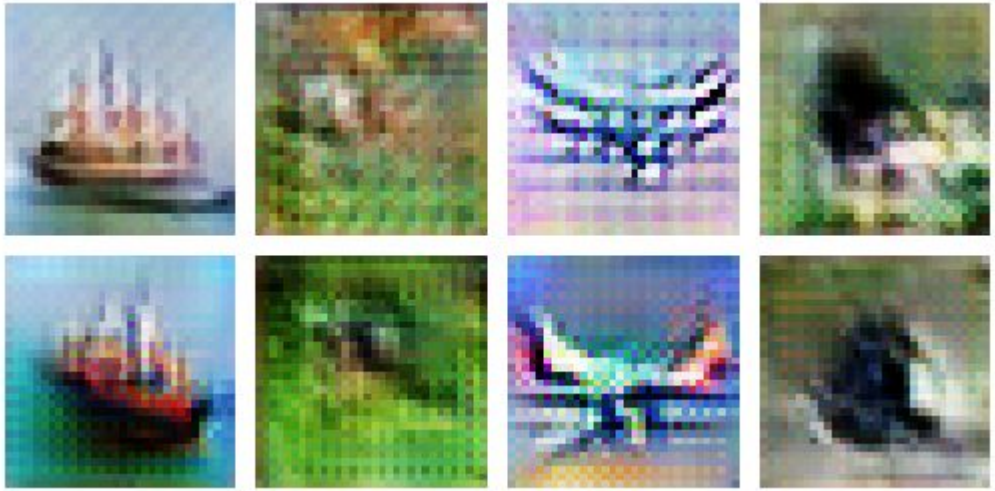
- Big receptive field
- Fine details due to top information to get precise semantic map

Upsampling



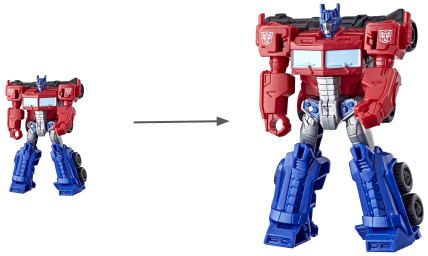
In decoder, we somehow have to upsample the image to recover its initial resolution. How can we do it?

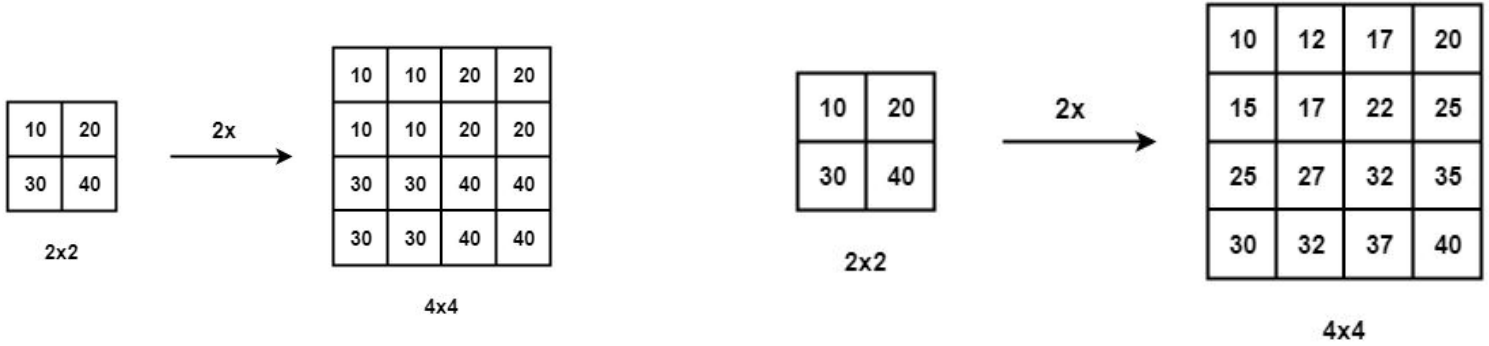




Transposed convolution can be used as upsampling function. But it can produce checkerboard artifacts

Upsampling



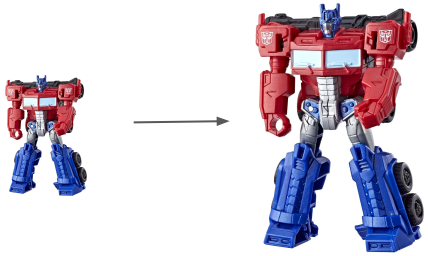


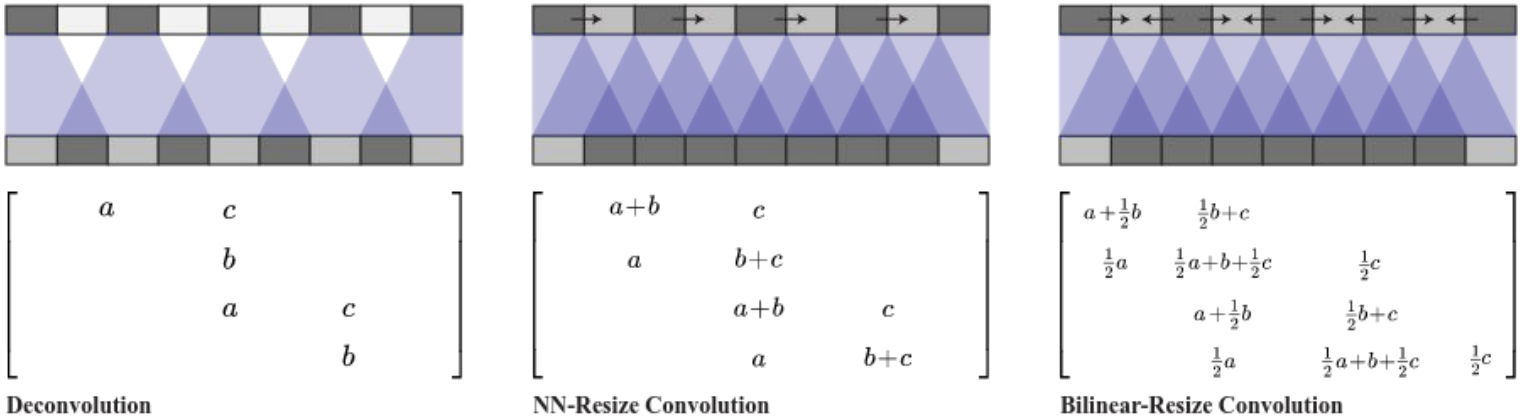
Classical upsampling methods

Nearest neighbor upsampling

Bilinear interpolation upsampling

Upsampling

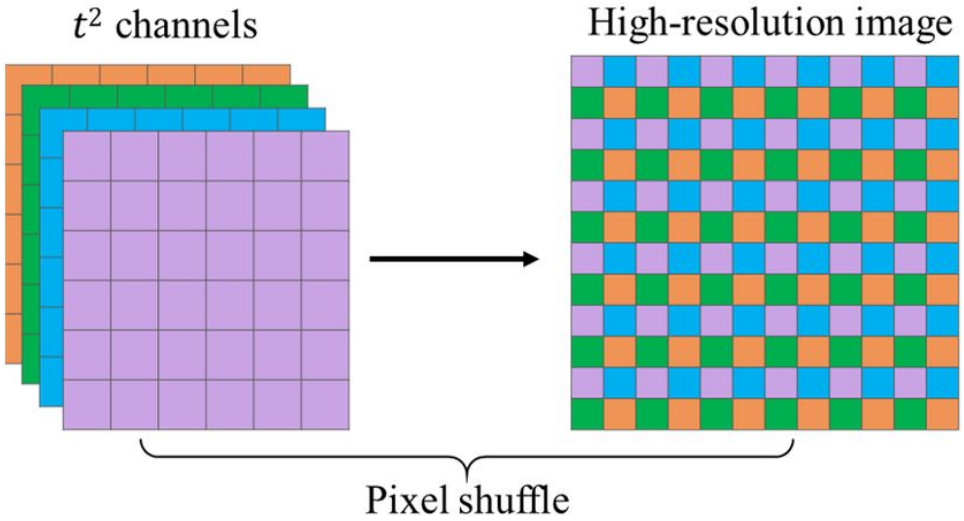




Voilà!

Pixel Shuffle

The interesting idea of pixel shuffle is to **reverse conception of less image resolution more channels** and use it for upsampling.

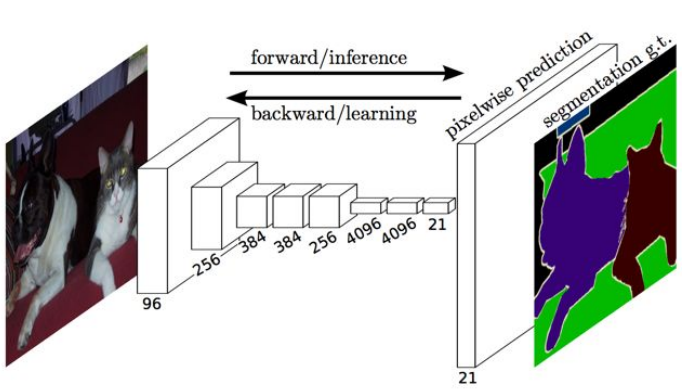


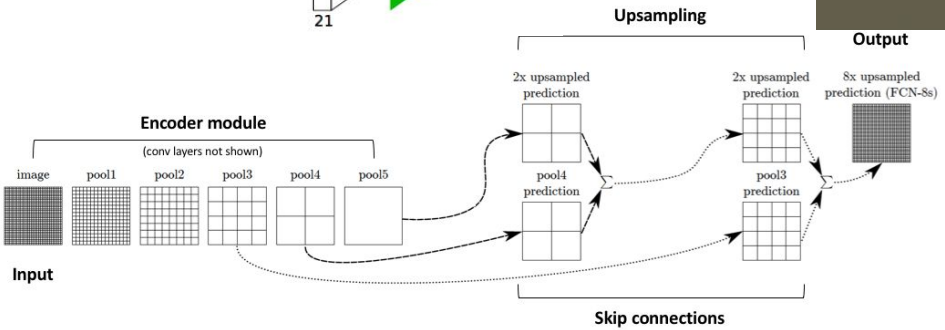
Transforms additional channels to spatial resolution (usually we do reverse)!

Architectures

FCN (Fully convolutional nets)

The idea is to transform classical classification network to convolutional. We replace FCN (fully connected layers) with FCN (**fully convolutional)**





FCN

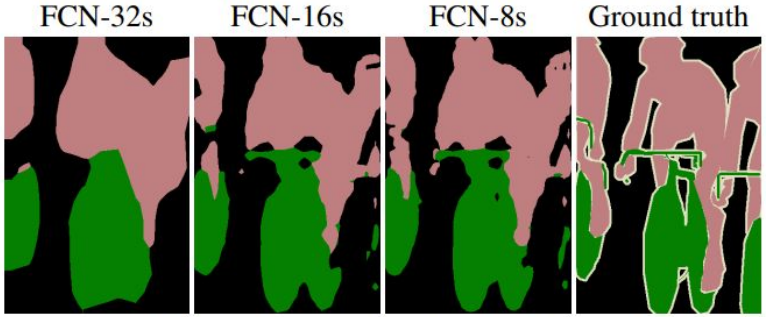
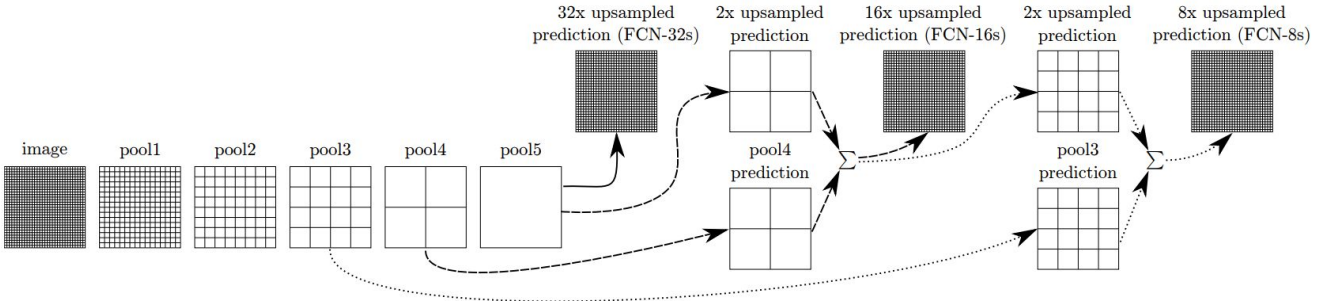
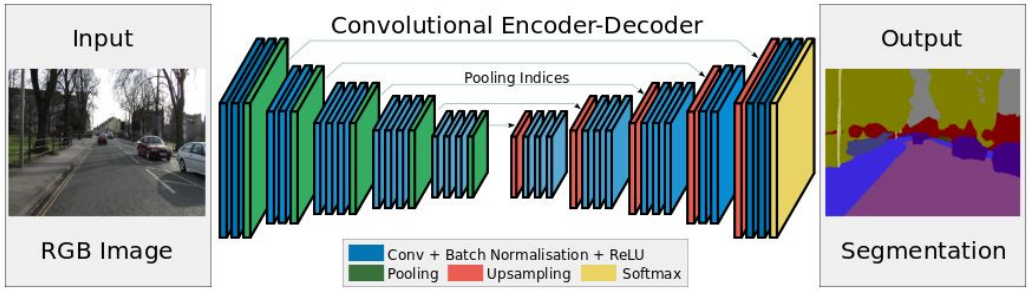


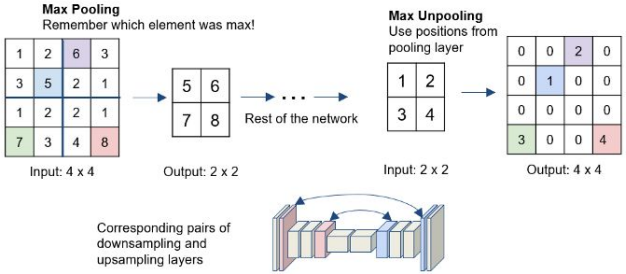
Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

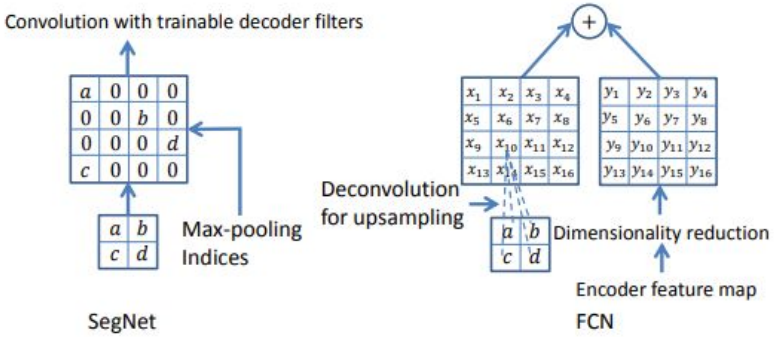
If we use more top information we get the better quality, remember the simple solution (using only classification network). The model understand semantics but it loses fine-grained details



SegNet





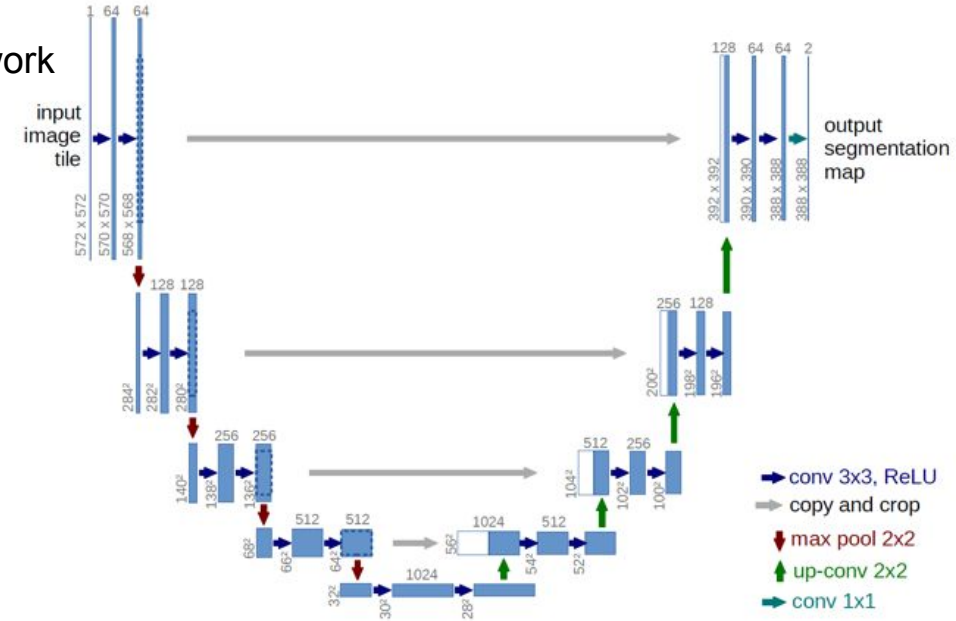


The first network with Encoder/Decoder layers.

- Making encoder and decoder equal in terms of parameters
- Saving more information from encoder using max unpooling operation.

UNet

The baseline which is still in work



UNet

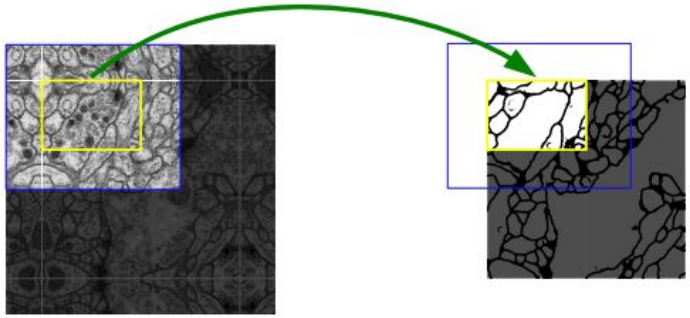
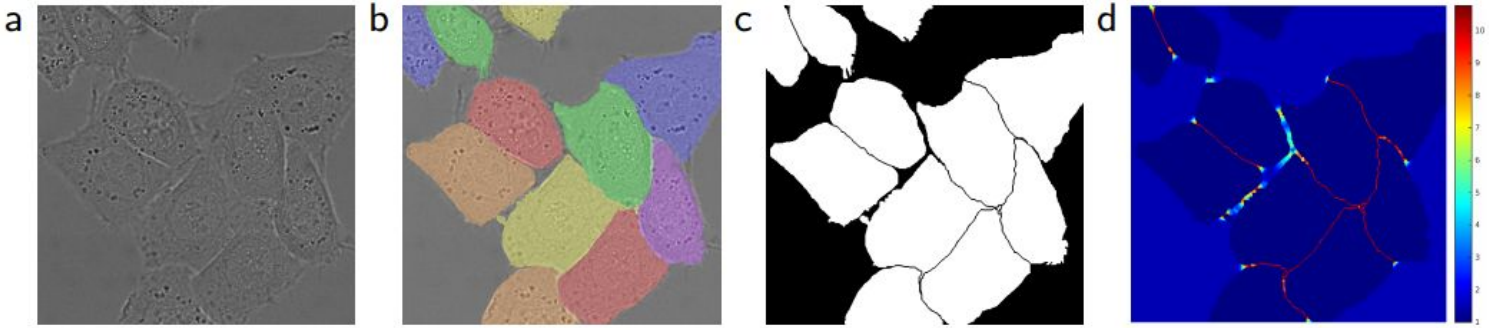


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring



HRNet

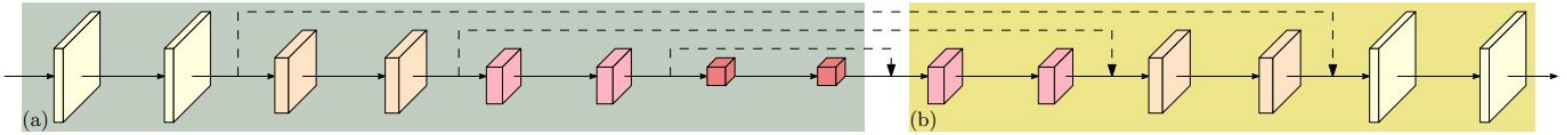
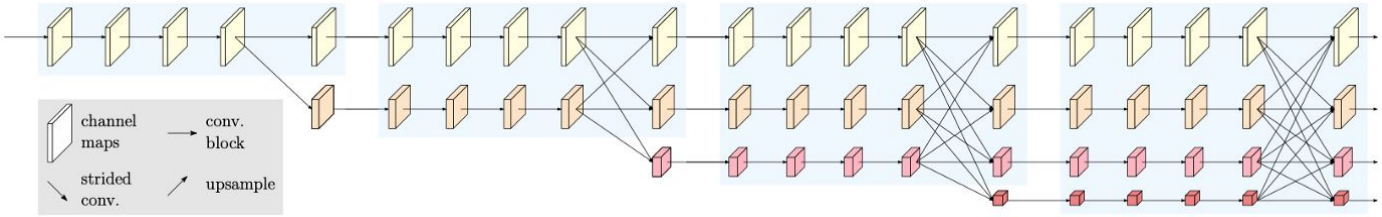


Fig. 1. The structure of recovering high resolution from low resolution. (a) A low-resolution representation learning subnetwork (such as VGGNet [126], ResNet [54]), which is formed by connecting high-to-low convolutions in series. (b) A high-resolution representation recovering subnetwork, which is formed by connecting low-to-high convolutions in series. Representative examples include SegNet [3], DeconvNet [107], U-Net [119] and Hourglass [105], encoder-decoder [112], and SimpleBaseline [152].

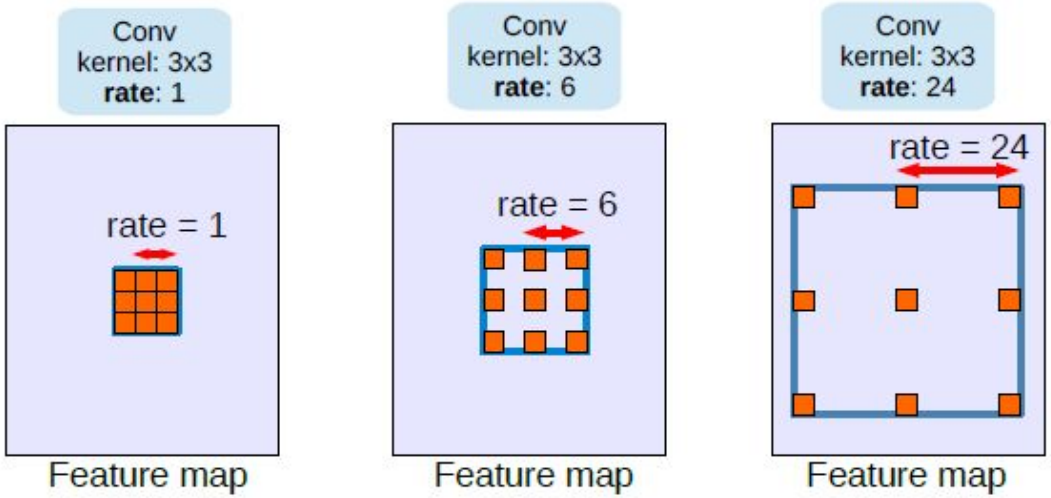
Base template for segmentation architecture



HRNet (UNet on anabolics)

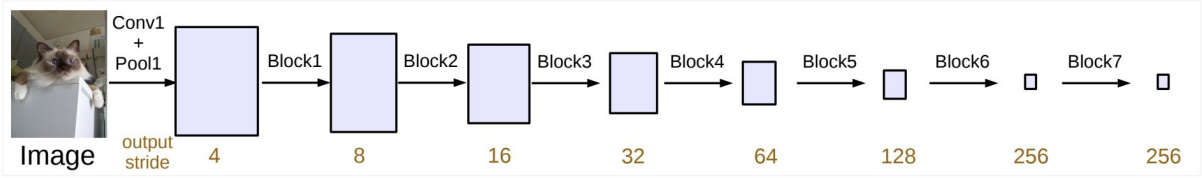
Atrous convolution

Pooling/Downsampling is essential to get good reception field. Otherwise, we have to do lots of convolutions. **How to increase receptive field w/o lowering image resolution?**

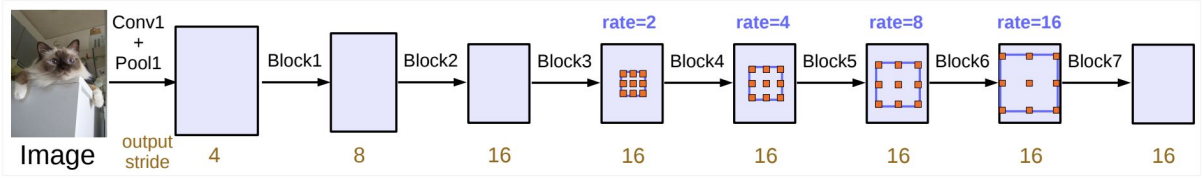


Let’s do convolution not with the neighbours but with pixels far from current location

Atrous convolution



(a) Going deeper without atrous convolution.

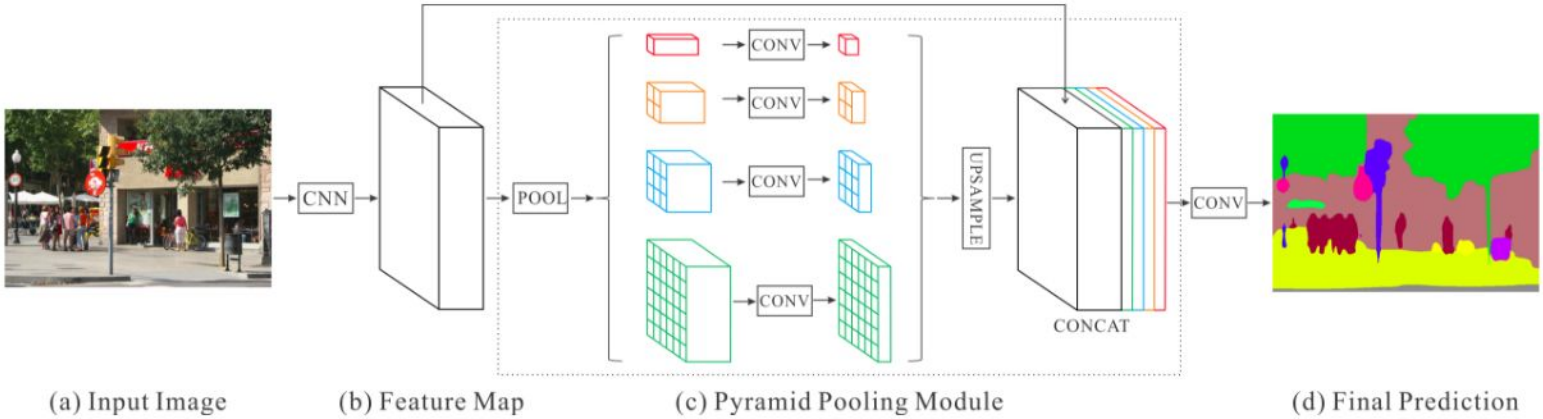


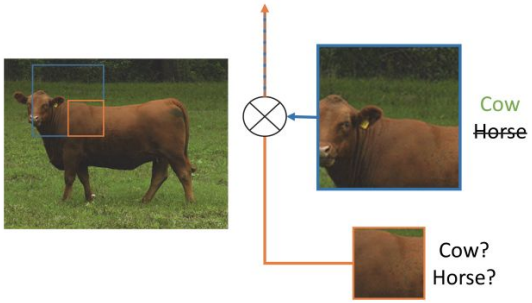
(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Figure 3. Cascaded modules without and with atrous convolution.

How standard architecture is change using this operation?. Atrous convolution allows to keep resolution but keeping the growth of receptive field

PSPNet


CNN
POOL
CONV
CONV
CONV
CONV
UPSAMPLE
CONCAT
CONV
(a) Input Image
(b) Feature Map
(c) Pyramid Pooling Module
(d) Final Prediction


Cow
Horse
Cow?
Horse?

DeepLab

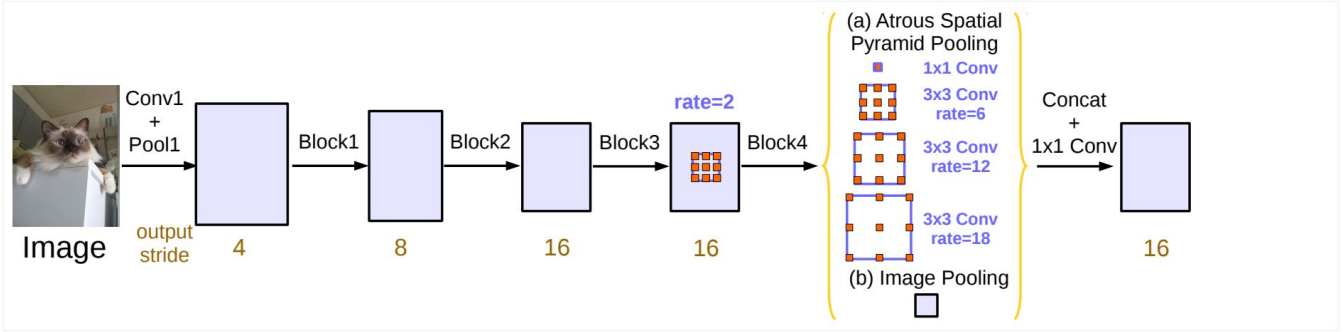
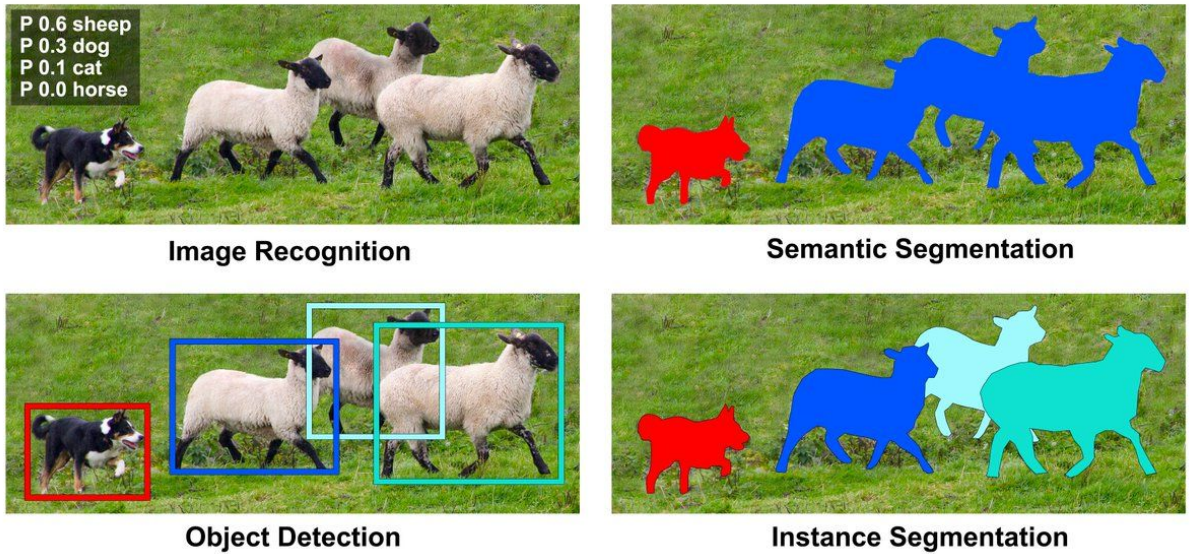


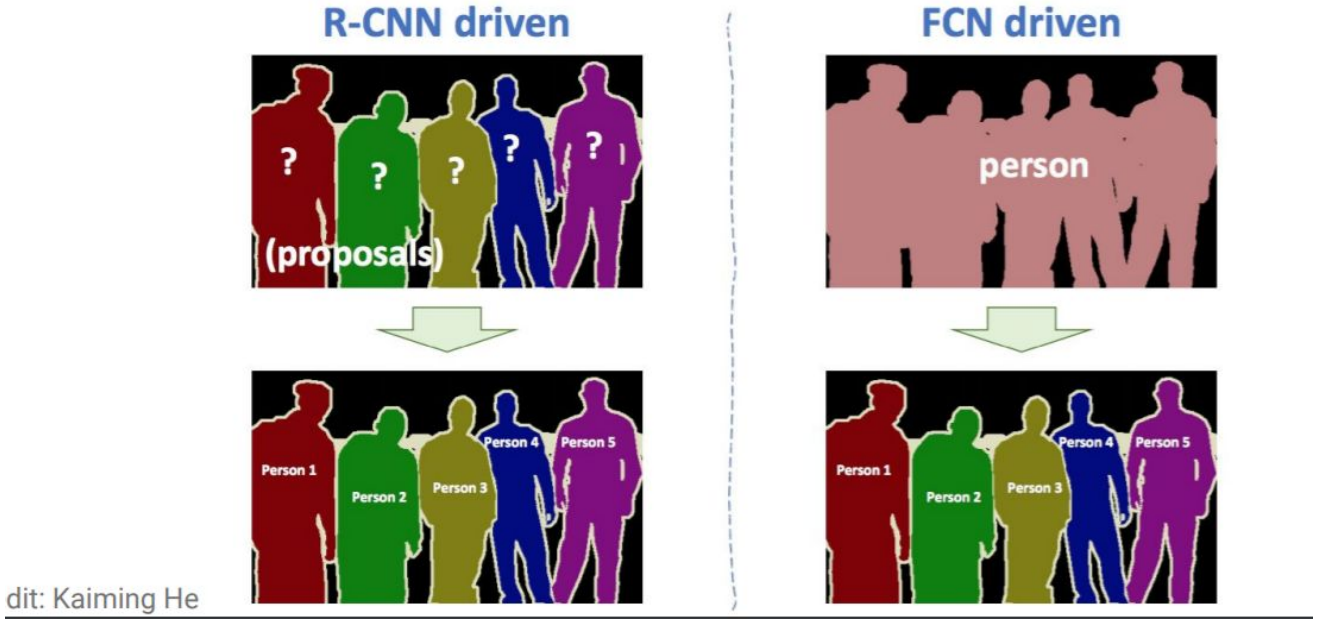
Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

Instance Segmentation



What if object detection is not enough and we want to have

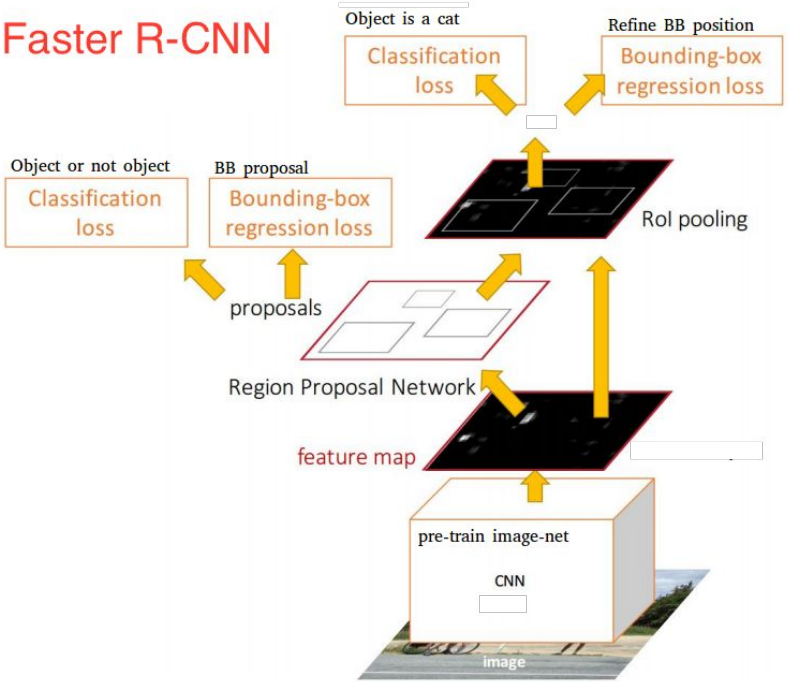
Instance Segmentation

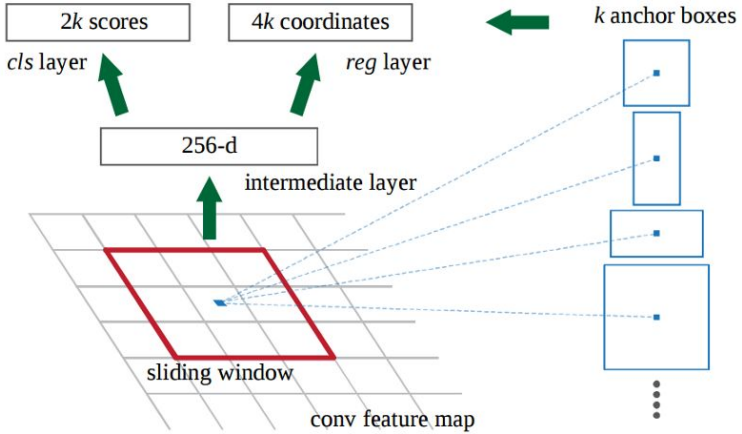


Instance segmentation has two approaches:

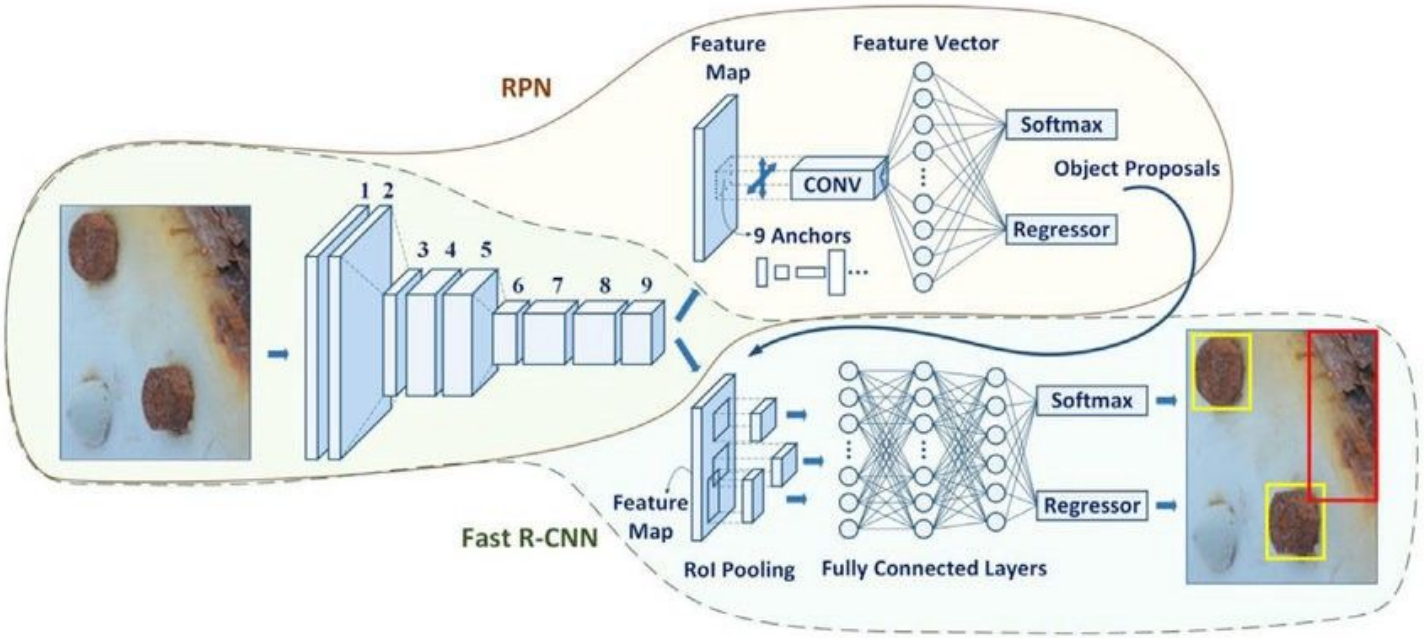
- Inspired by object detection (R-CNN driven).
- Inspired by segmentation (FCN driven)

Reminder:

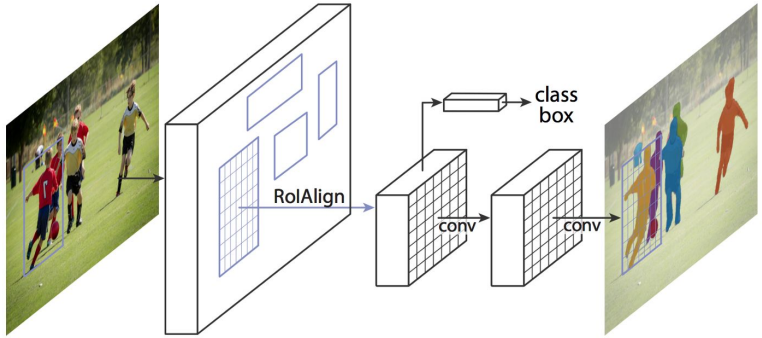

Faster R-CNN
Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image


2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

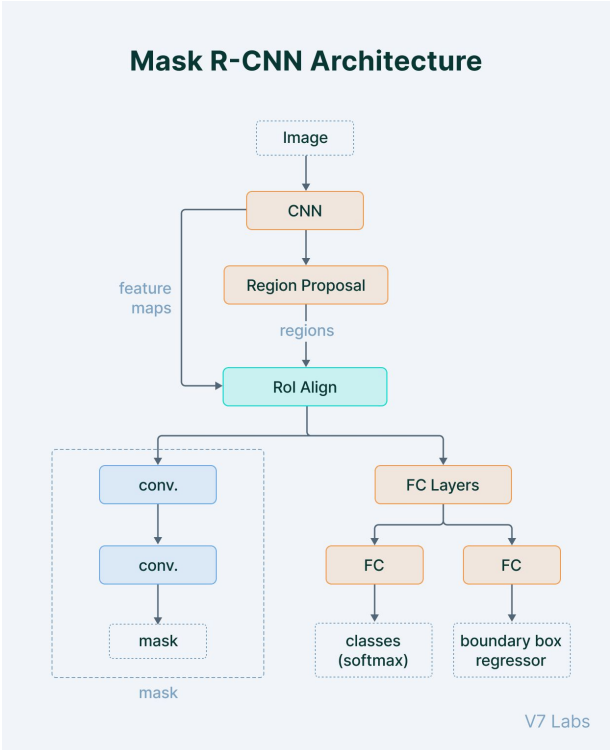
Faster RCNN


RPN
Feature Map
Feature Vector
Softmax
Object Proposals
CONV
Regressor
9 Anchors
1 2
3 4 5
6 7 8 9
Feature Map
Softmax
Regressor
Fast R-CNN
RoI Pooling
Fully Connected Layers

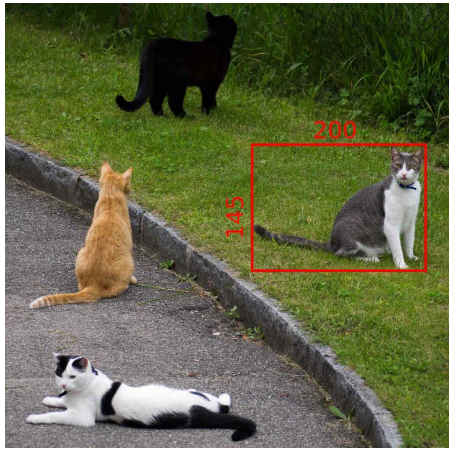
Mask R-CNN

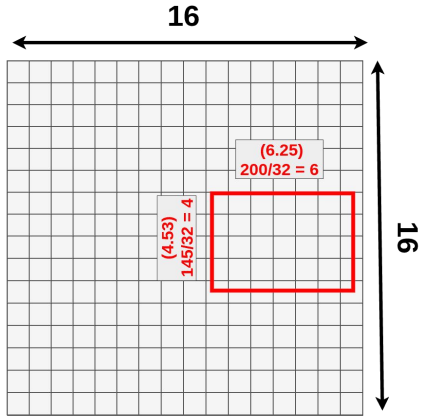


Adding mask prediction head



RoI Pooling

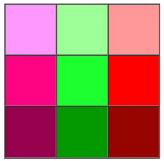




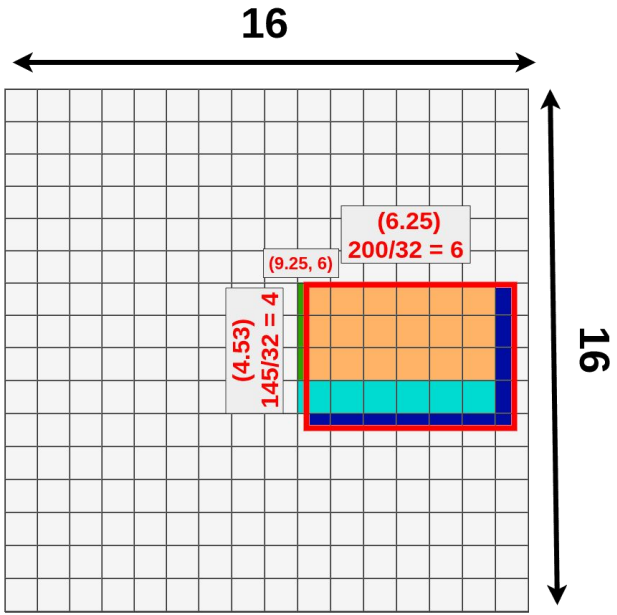
4x6 RoI

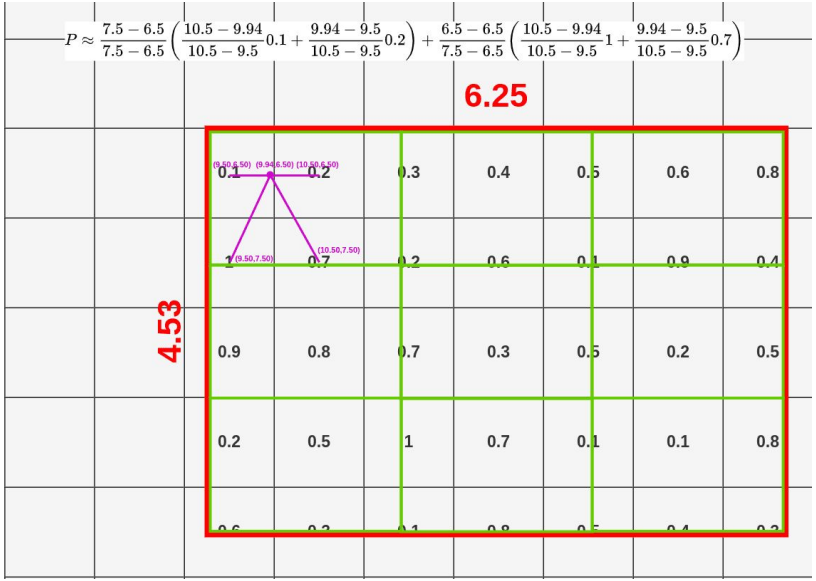
0.1	0.2	0.3	0.4	0.5	0.6
1	0.7	0.2	0.6	0.1	0.9
0.9	0.8	0.7	0.3	0.5	0.2

3x3 RoI Pooling

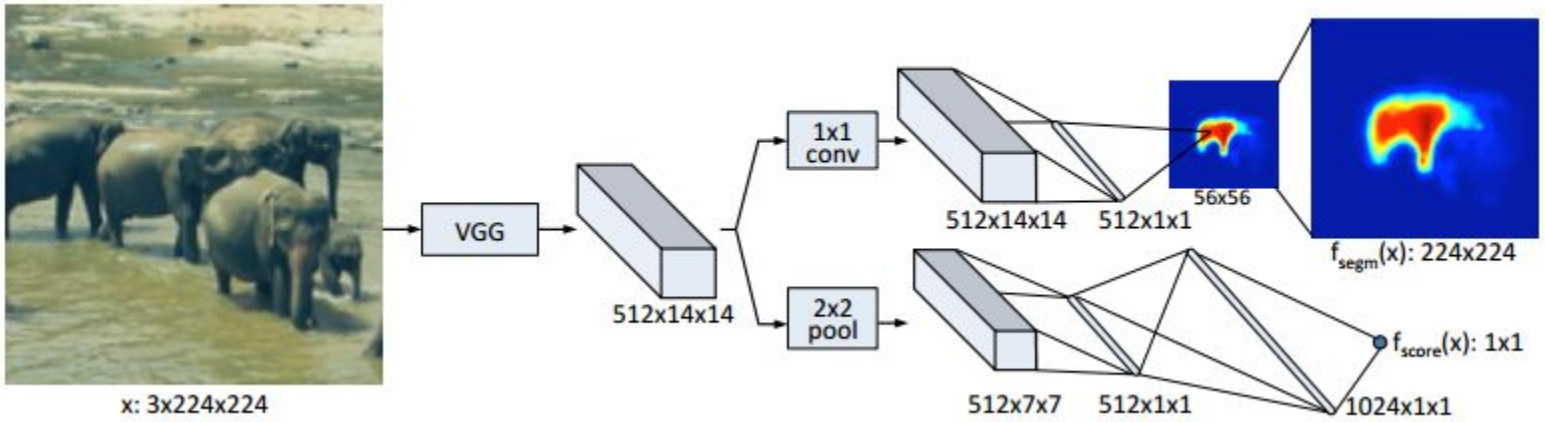


RoI Align


16
16
(6.25)
200/32 = 6
(9.25, 6)
(4.53)
145/32 = 4


$P \approx \frac{7.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}0.1+\frac{9.94-9.5}{10.5-9.5}0.2\right)+\frac{6.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}1+\frac{9.94-9.5}{10.5-9.5}0.7\right)$
6.25
4.53
0.1 0.2 0.3 0.4 0.5 0.6 0.8
1 0.7 0.2 0.6 0.1 0.9 0.4
0.9 0.8 0.7 0.3 0.5 0.2 0.5
0.2 0.5 1 0.7 0.1 0.1 0.8

DeepMask


x: 3x224x224
VGG
512x14x14
1x1 conv
2x2 pool
512x14x14
512x1x1
56x56
f_{segm}(x): 224x224
512x7x7
512x1x1
1024x1x1
f_{score}(x): 1x1

Panoptic Segmentation

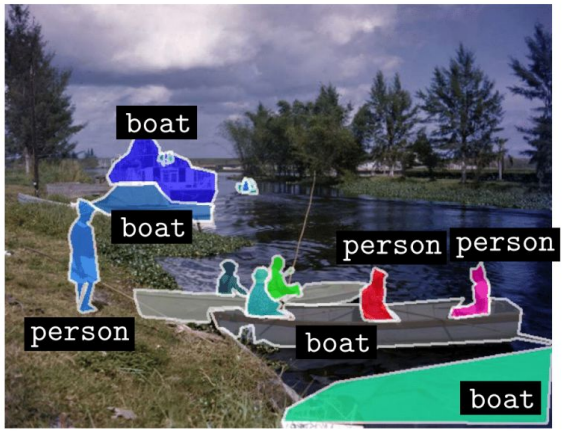
Semantic

Instance

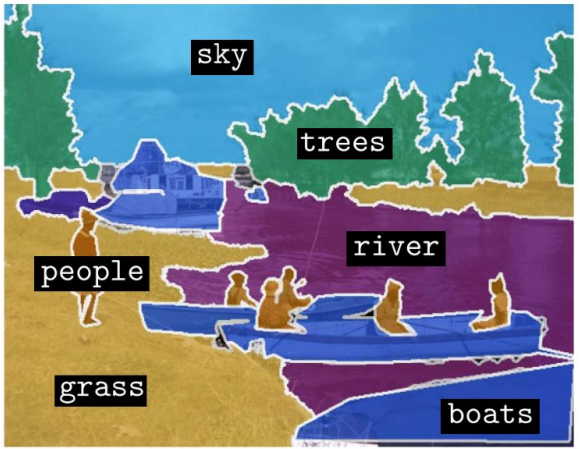
Panoptic

Panoptic Segmentation

What do every segmentation do?



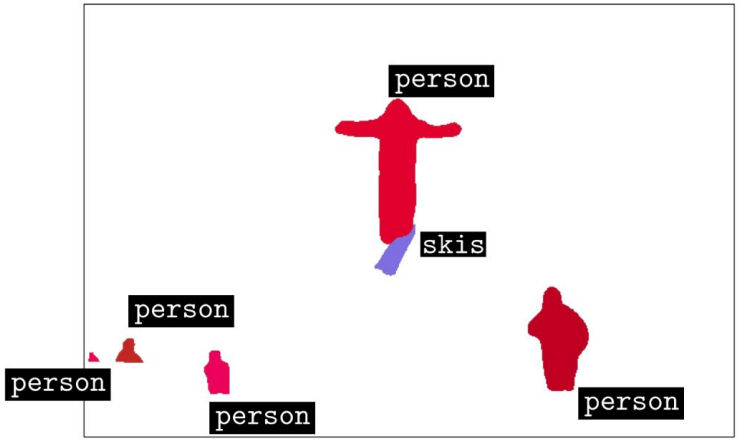
instance segmentation

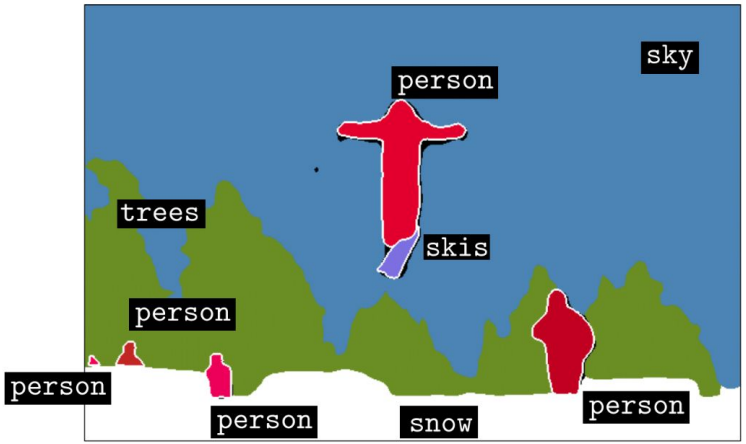


semantic segmentation

Panoptic Segmentation

What do instance segmentation see?

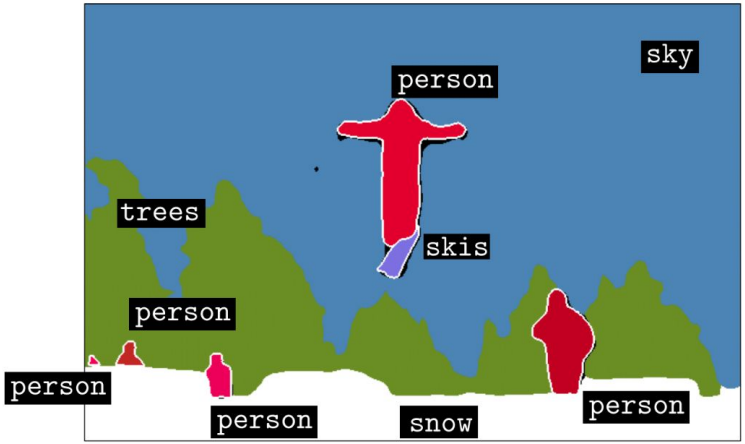


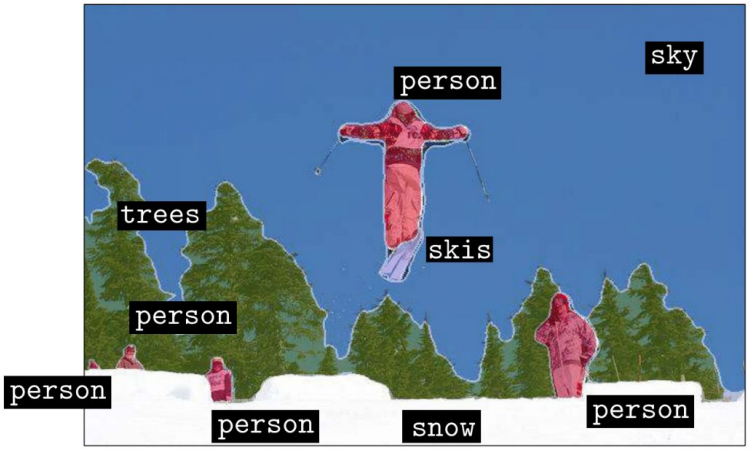


Let’s add semantic segmentation information

Panoptic Segmentation

How does the real image looks like?





Metric

Semantic segmentation

- IoU, per-pixel metric

Instance segmentation

- mAP, object-size agnostic

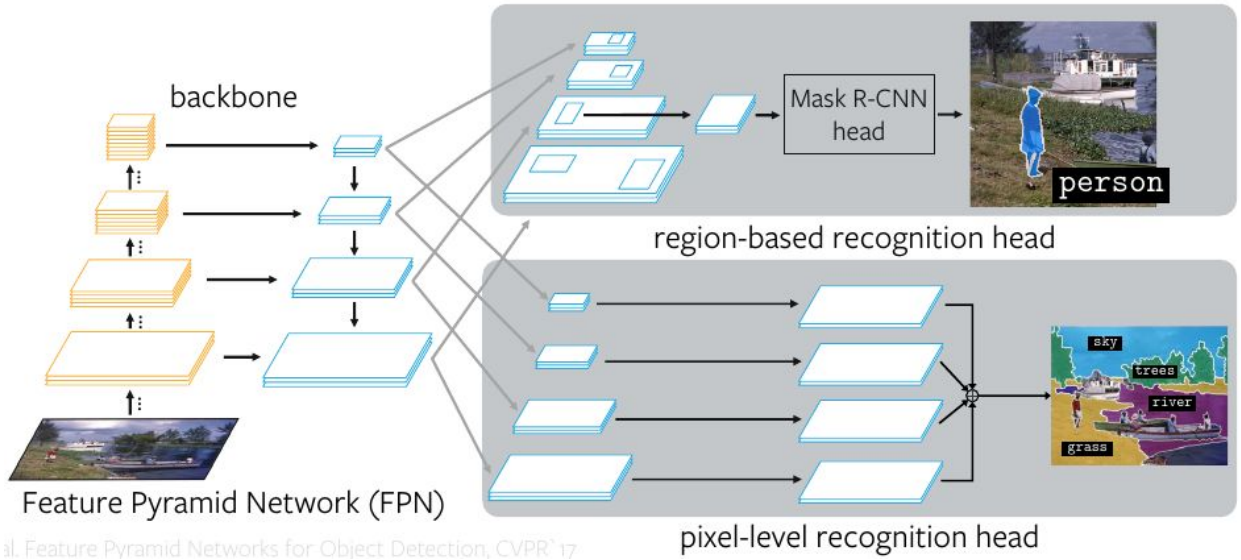
Panoptic segmentation should include both metrics simultaneously

Panoptic Segmentation

How to evaluate performance of panoptic segmentation model?

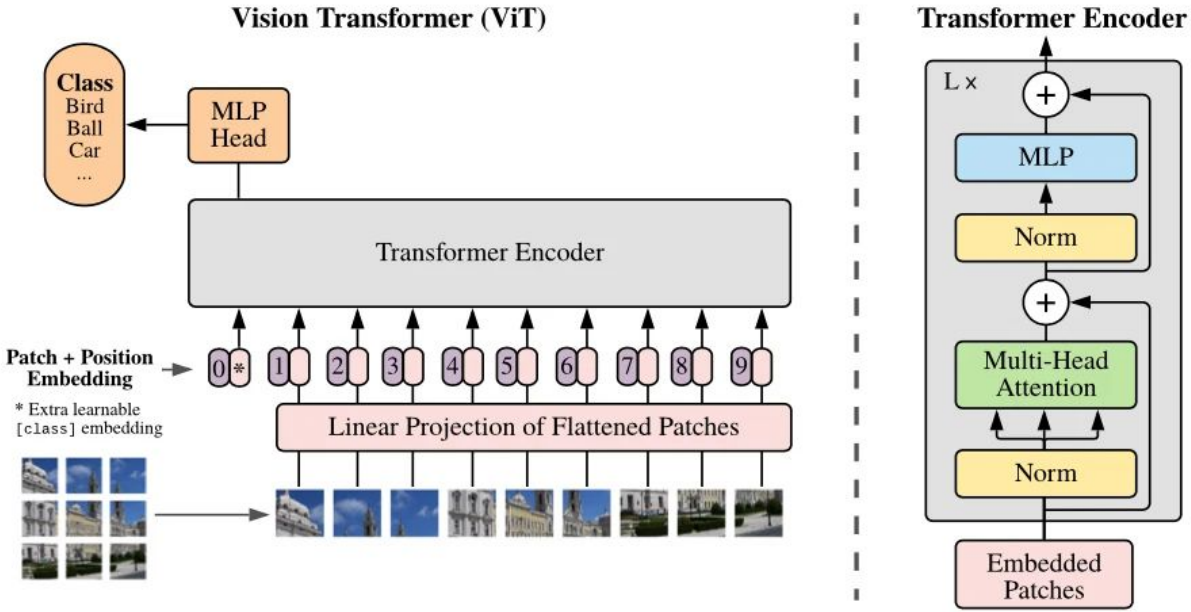
$$\begin{aligned}\mathrm{PQ} &= \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \\ &= \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}\end{aligned}$$

Panoptic architecture

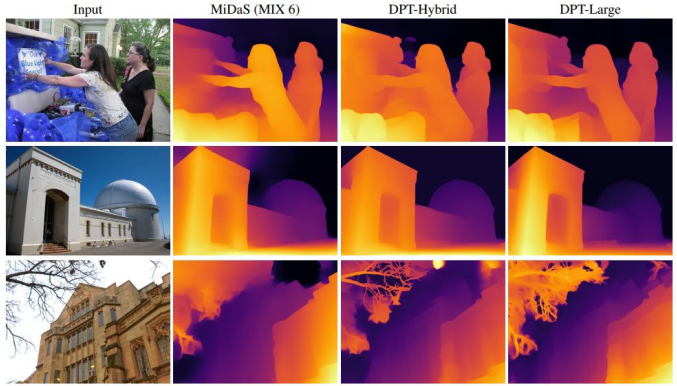


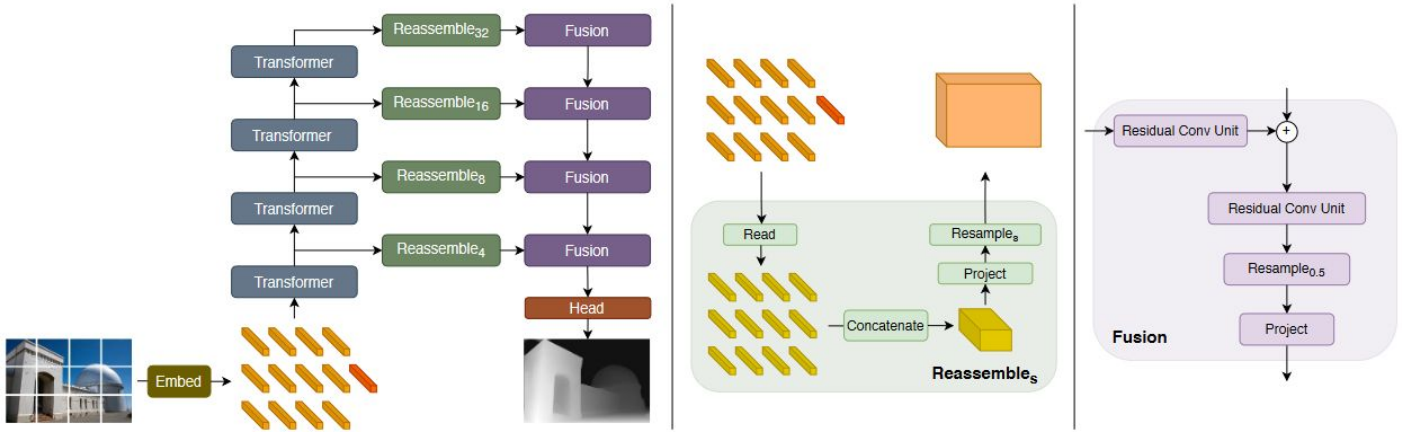
al. Feature Pyramid Networks for Object Detection, CVPR`17

Reminder: ViT

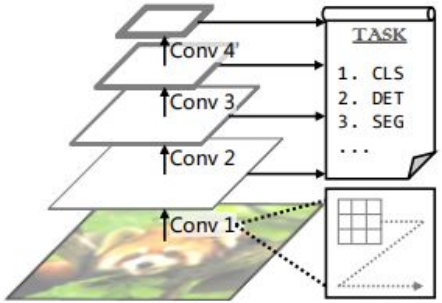


DPT (Vision Transformers for Dense Prediction)

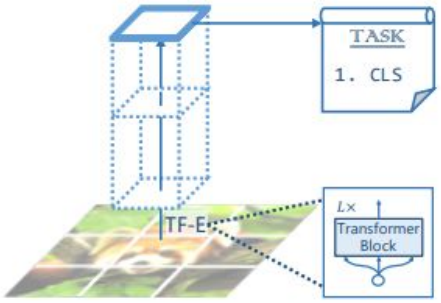




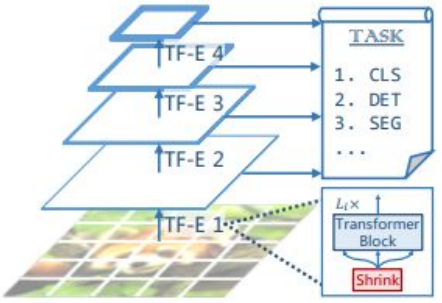
Pyramid Vision Transformer



(a) CNNs: VGG [54], ResNet [22], *etc.*

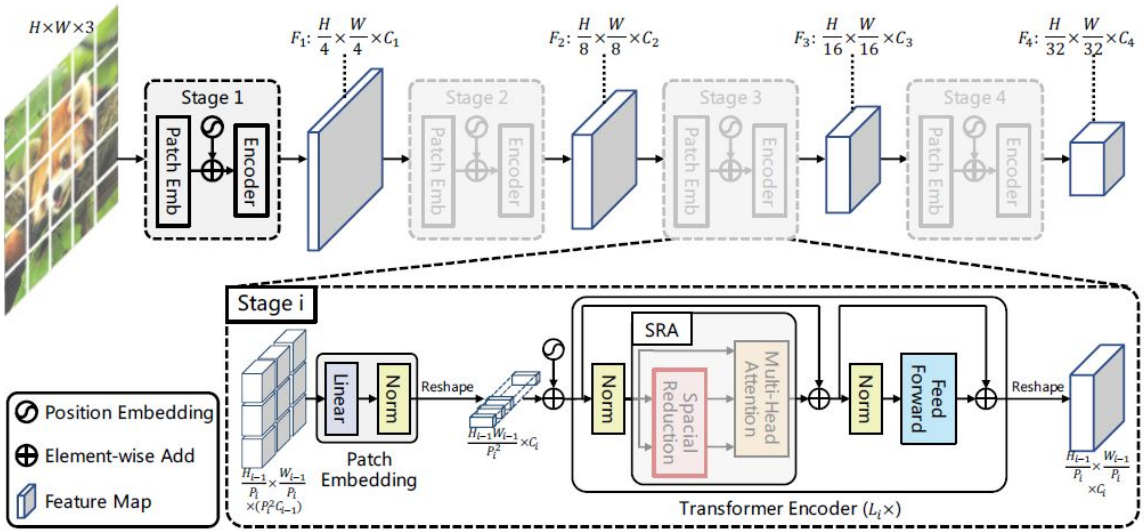


(b) Vision Transformer [13]

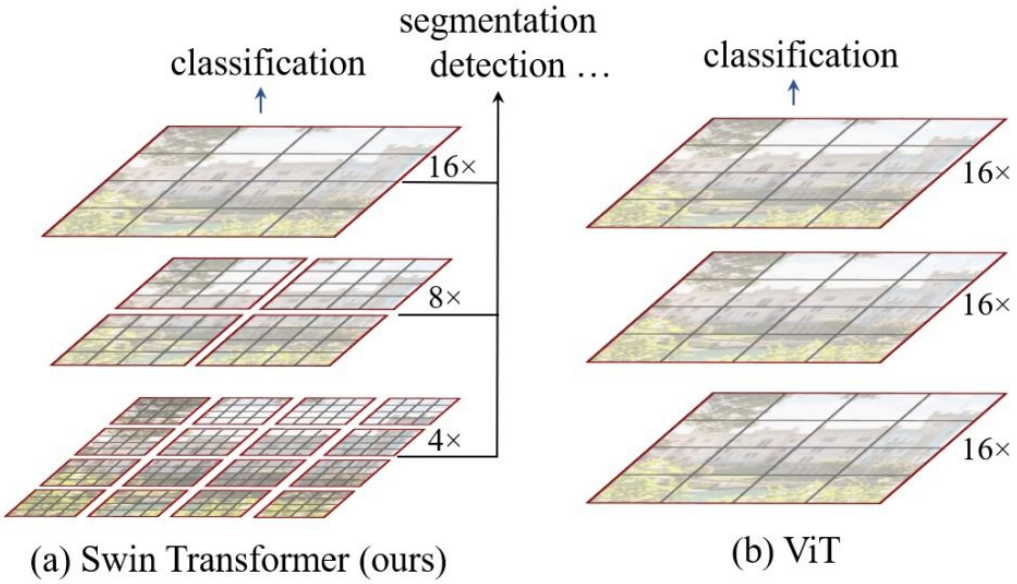


(c) Pyramid Vision Transformer (ours)

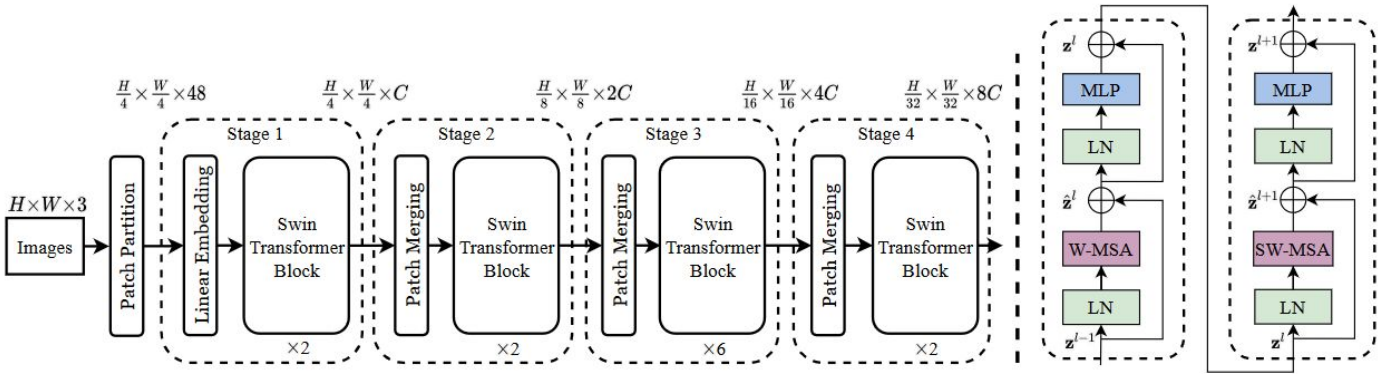
Pyramid Vision Transformer

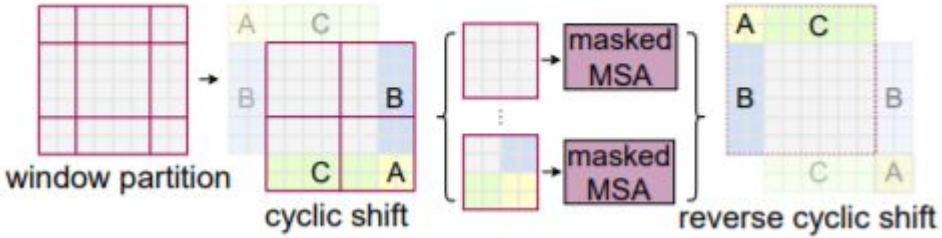


Swin


classification
segmentation
detection …
16×
8×
4×
(a) Swin Transformer (ours)
classification
16×
16×
16×
(b) ViT

Swin


$H \times W \times 3$
Images
Patch Partition
$\frac{H}{4} \times \frac{W}{4} \times 48$
Linear Embedding
Stage 1
Swin Transformer Block
×2
$\frac{H}{4} \times \frac{W}{4} \times C$
Patch Merging
Stage 2
Swin Transformer Block
×2
$\frac{H}{8} \times \frac{W}{8} \times 2C$
Patch Merging
Stage 3
Swin Transformer Block
×6
$\frac{H}{16} \times \frac{W}{16} \times 4C$
Patch Merging
Stage 4
Swin Transformer Block
×2
$\frac{H}{32} \times \frac{W}{32} \times 8C$
$\mathbf{z}^{l}$
MLP
LN
$\hat{\mathbf{z}}^{l}$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^{l}$


window partition
A
C
B
B
C
A
cyclic shift
masked MSA
masked MSA
A
C
B
B
C
A
reverse cyclic shift

HRFormer

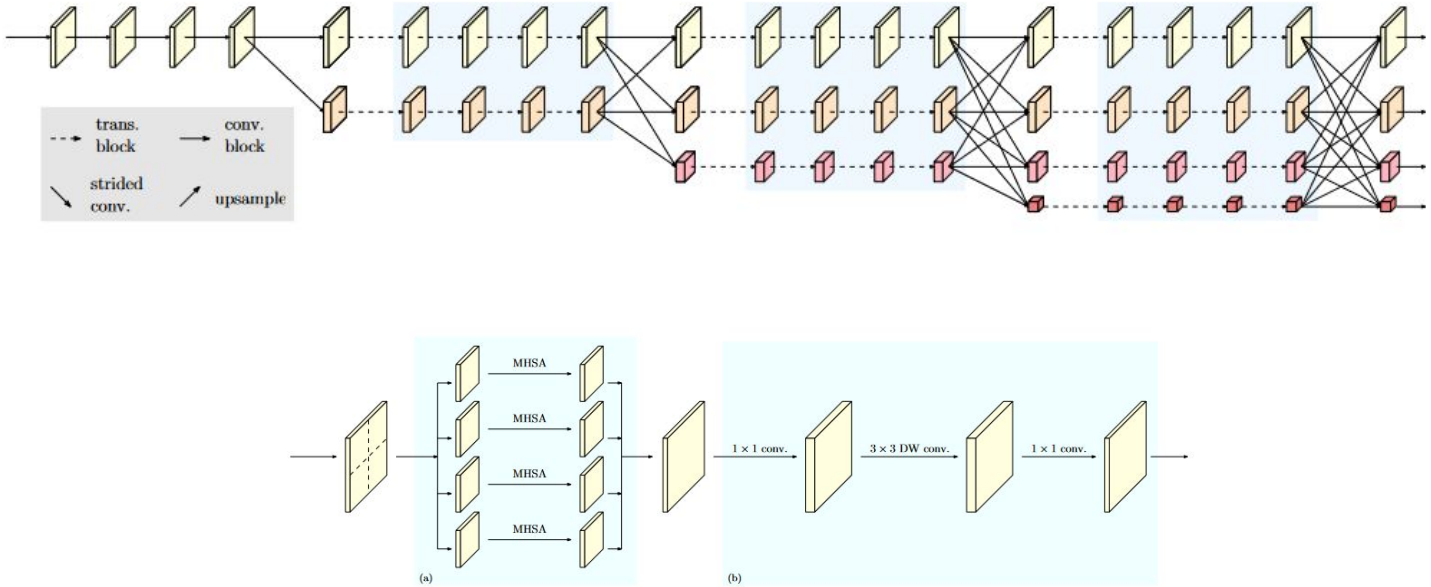
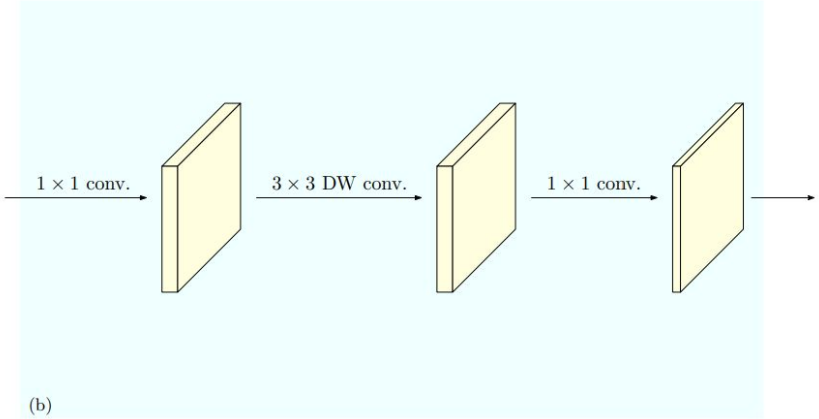


Figure 1: **Illustrating the HRFormer block.** The HRFormer block is composed of (a) local-window self-attentionm and (b) feed-forward network (FFN) with depth-wise convolution. The local-window self-attention scheme is inspired by the interlaced sparse self-attention [56, 21].

HRFormer



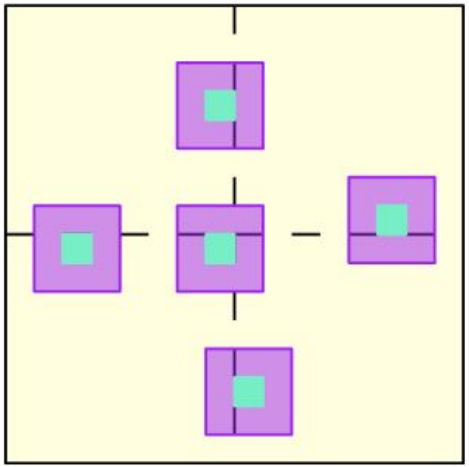


Figure 3: Illustrating that FFN with 3×3 depth-wise convolution connects the non-overlapping windows.

Recap

- Semantic segmentation problem
- Upsampling
- Architectures
- Panoptic / Instance segmentation