

# 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

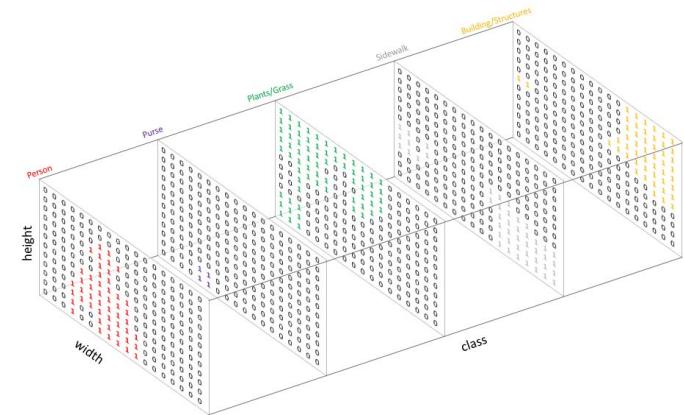
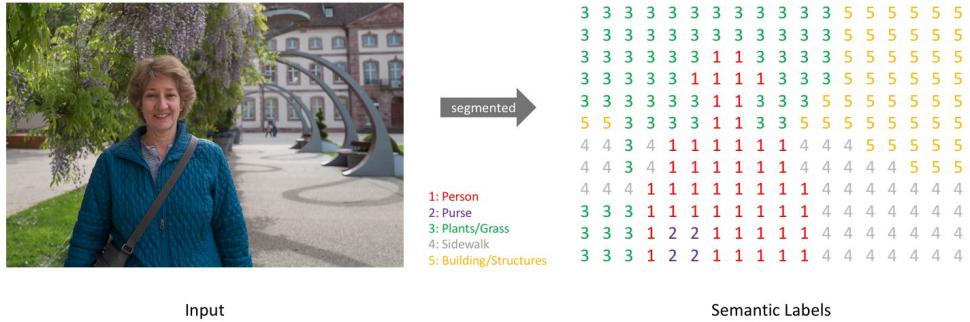
**What is the solution?**

# 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

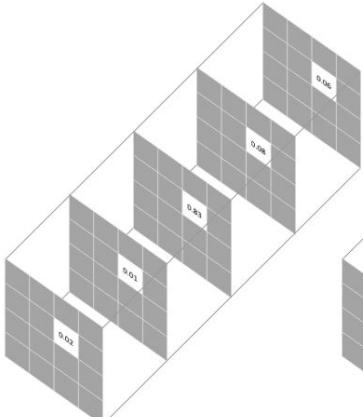


So, the output is map where we assign label to every pixel -> Output  $H \times W \times 1$ .

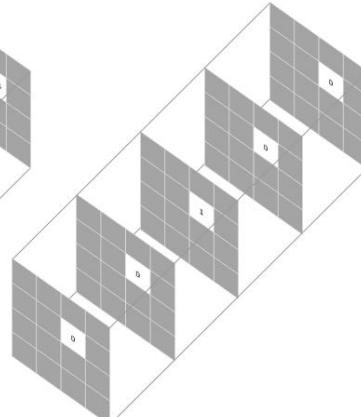
But neural network work poorly with discrete values? How to parametrize predictions?

# Loss functions

How to train semantic segmentation model?



Prediction for a selected pixel

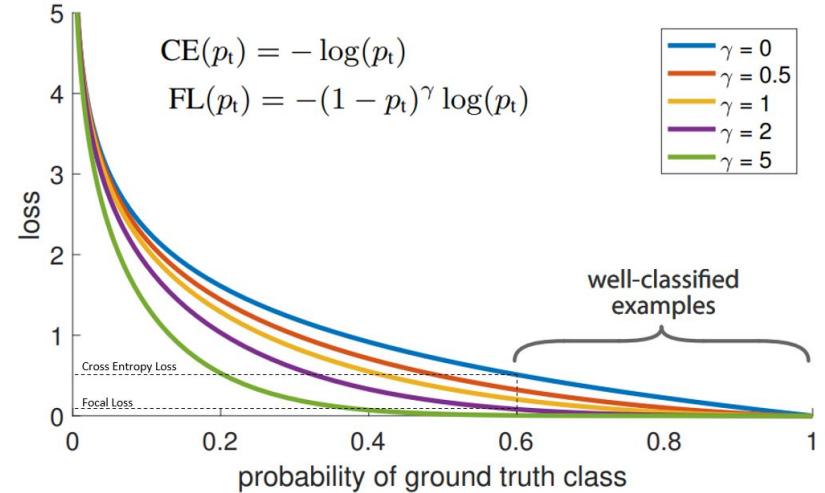


Target for the corresponding pixel

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

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

This scoring is repeated over all pixels and averaged



$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t)$$

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?



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

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

Averaging over classes

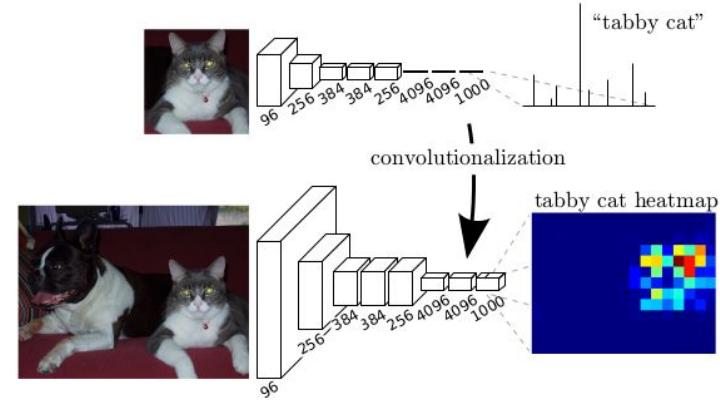
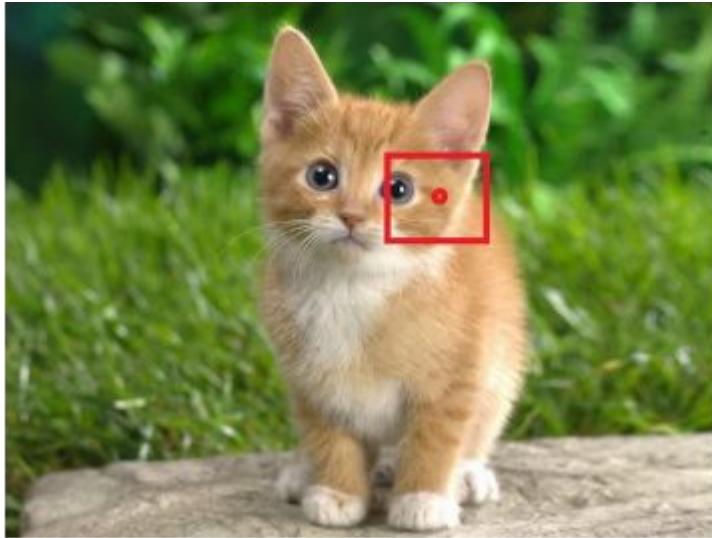
$mIoU$

Averaging over classes

$mAcc$

# Simple solution

How to transform classification model into segmentation model?

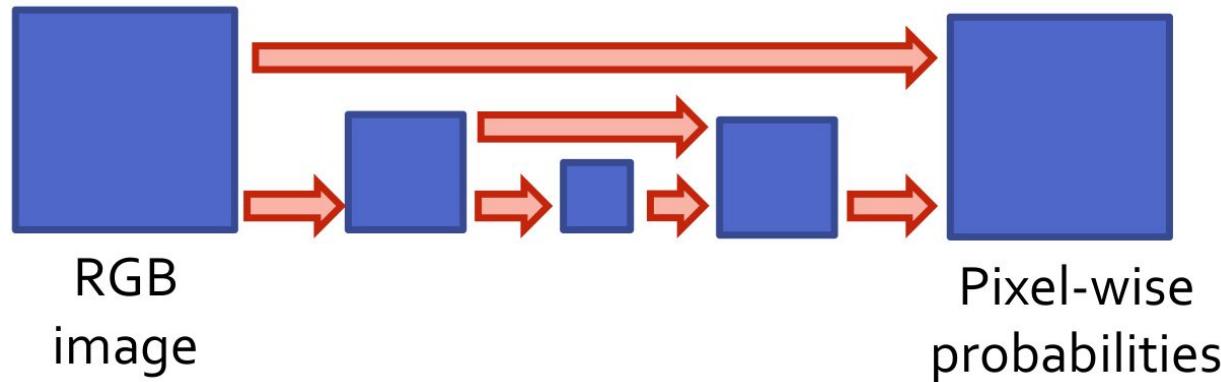


Lets, get prediction by using sliding window and constructing semantic map

- Works too long
- Resolution is very bad

# Better solution

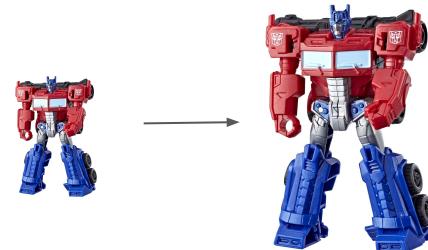
If we have a problem with low resolution so let's resolution as high as possible. But how we can do it?



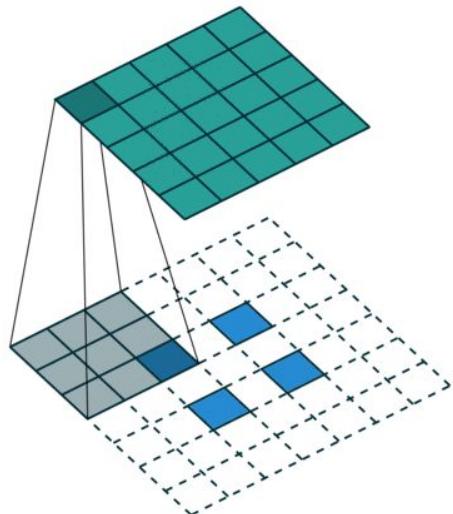
Encoder-Decoder approach (like autoencoder) solves both problems:

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

# Upsampling

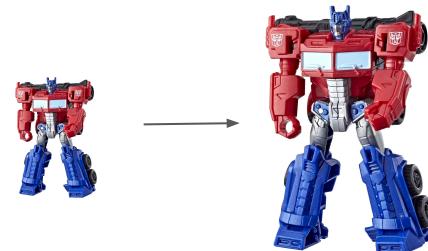


In decoder, we somehow have to upsample the image to recover its initial resolution. How can we do it?



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

# Upsampling



10	20
30	40

2x2

10	10	20	20
10	10	20	20
30	30	40	40
30	30	40	40

4x4

10	20
30	40

2x2

10	12	17	20
15	17	22	25
25	27	32	35
30	32	37	40

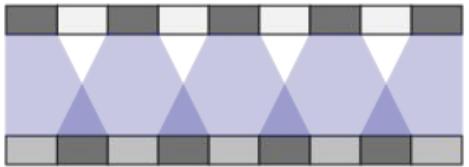
4x4

## Classical upsampling methods

Nearest neighbor upsampling

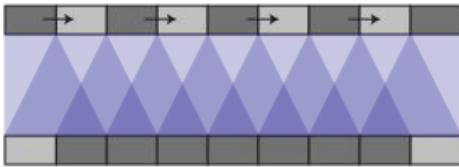
Bilinear interpolation upsampling

# Upsampling



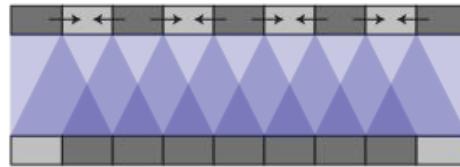
$$\begin{bmatrix} a & c \\ b & \\ a & c \\ & b \end{bmatrix}$$

Deconvolution



$$\begin{bmatrix} a+b & c \\ a & b+c \\ a+b & c \\ a & b+c \end{bmatrix}$$

NN-Resize Convolution



$$\begin{bmatrix} a + \frac{1}{2}b & \frac{1}{2}b+c & \\ \frac{1}{2}a & \frac{1}{2}a+b+\frac{1}{2}c & \frac{1}{2}c \\ a + \frac{1}{2}b & \frac{1}{2}b+c & \\ \frac{1}{2}a & \frac{1}{2}a+b+\frac{1}{2}c & \frac{1}{2}c \end{bmatrix}$$

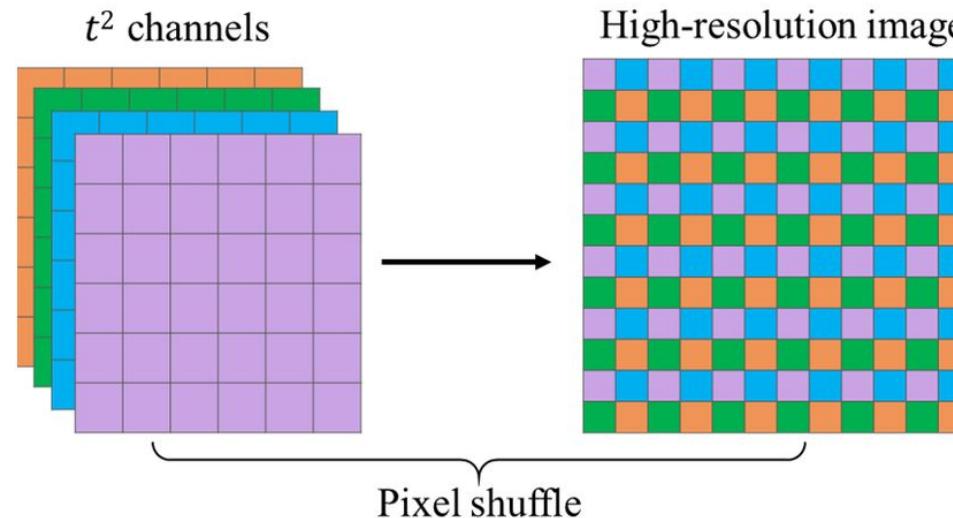
Bilinear-Resize Convolution



Voilà!

# Pixel Shuffle

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

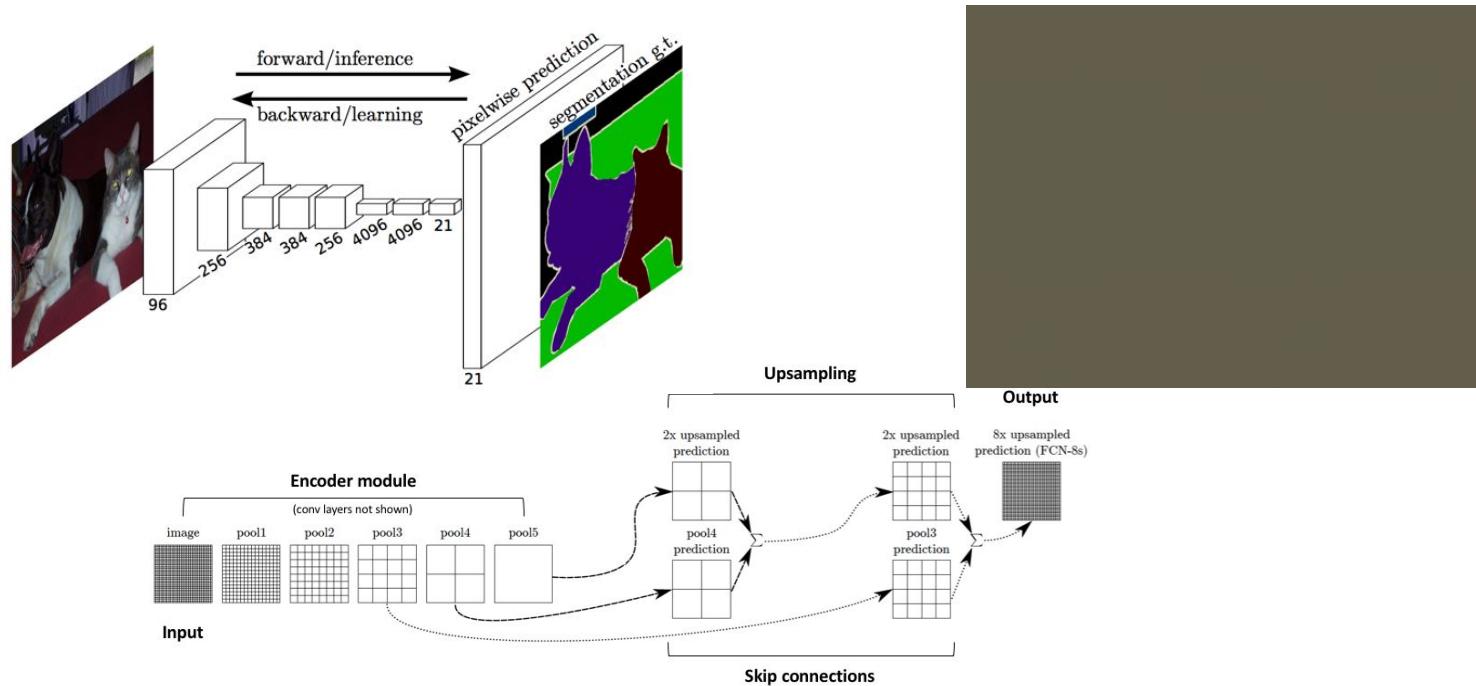


Transforms additional channels to spatial resolution (usually we do reverse)!

# 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

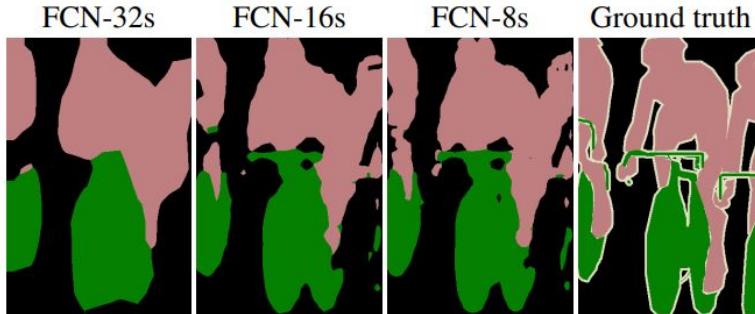
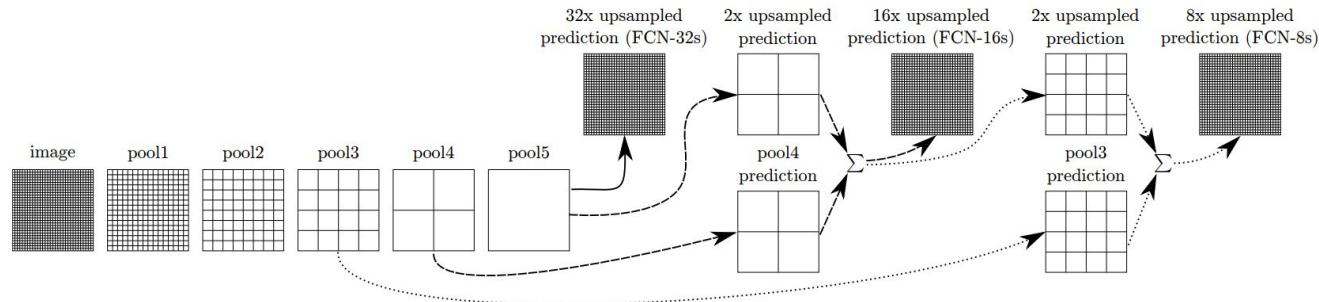
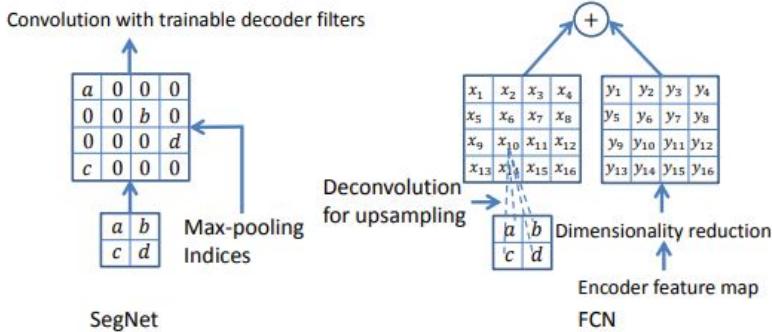
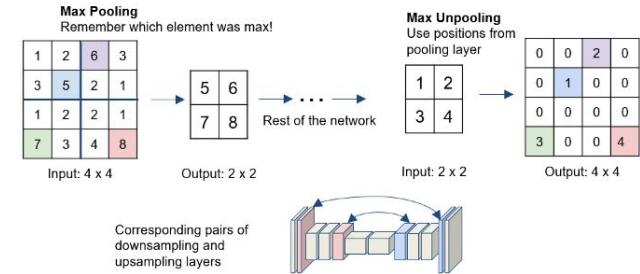
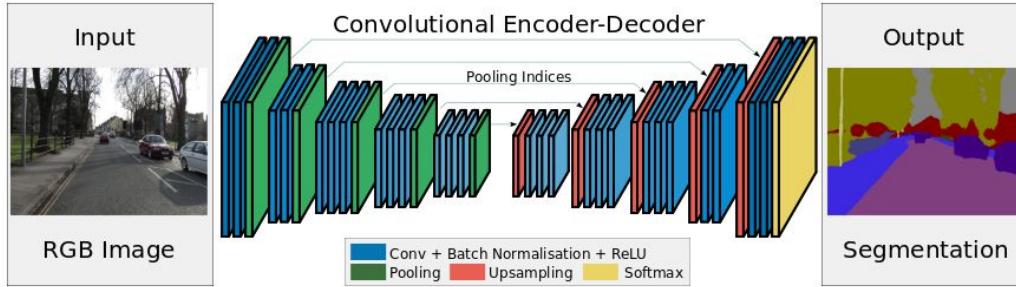


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

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

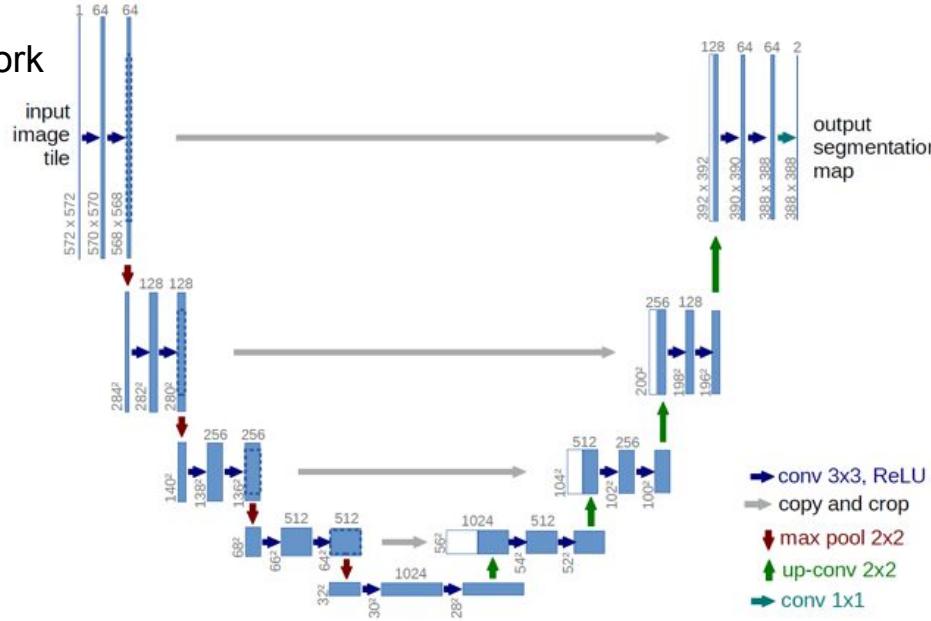


The first network with Encoder/Decoder layers.

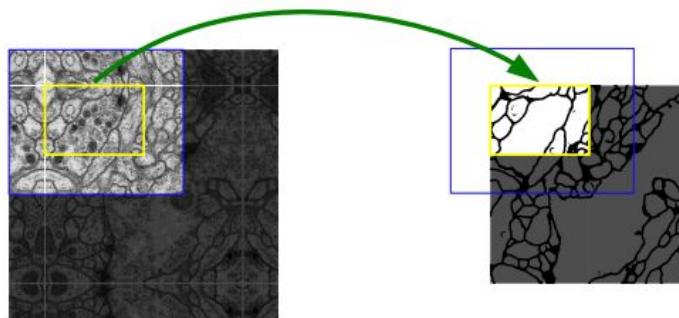
- Making encoder and decoder equal in terms of parameters
- Saving more information from encoder using max unpooling operation.

# UNet

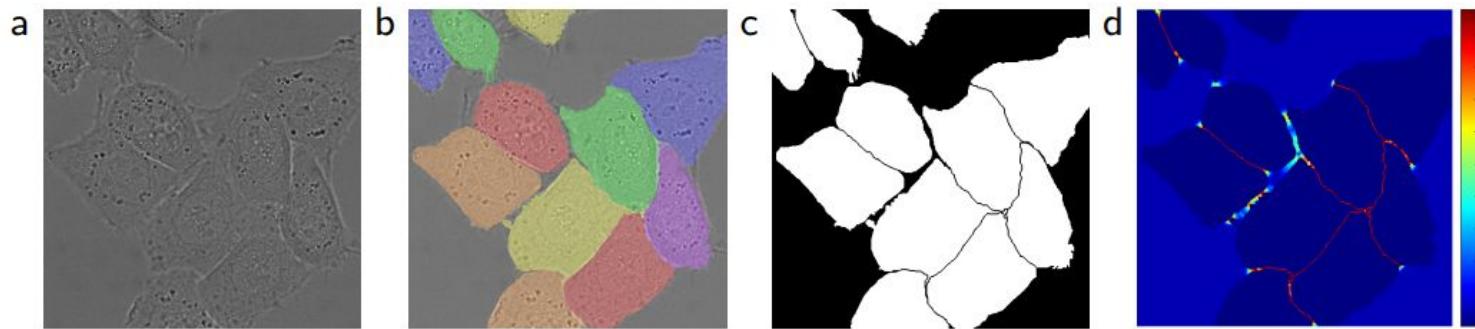
The baseline which is still in work



# 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

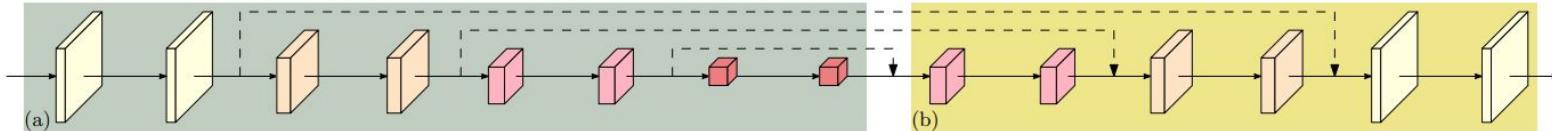
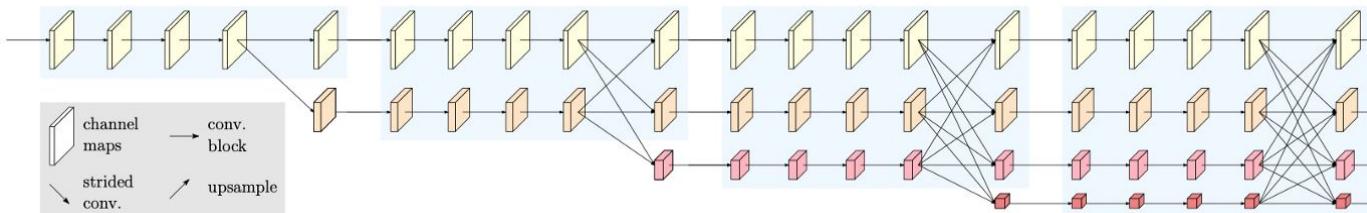


Fig. 1. The structure of recovering high resolution from low resolution. (a) A low-resolution representation learning subnetwork (such as VGGNet [126], ResNet [54]), which is formed by connecting high-to-low convolutions in series. (b) A high-resolution representation recovering subnetwork, which is 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].

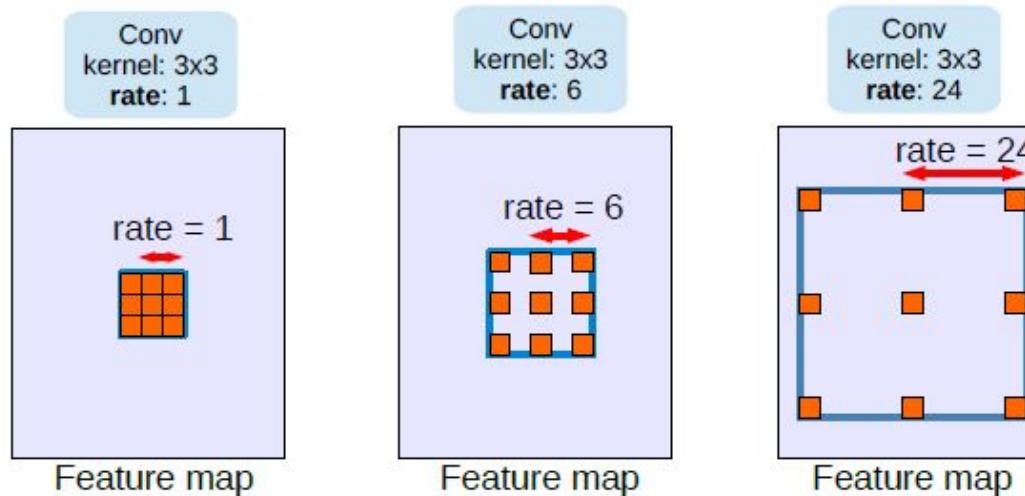
## Base template for segmentation architecture



## HRNet (UNet on anabolics)

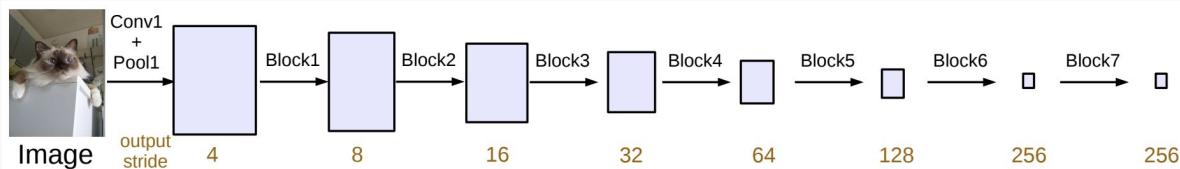
# Atrous convolution

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?**

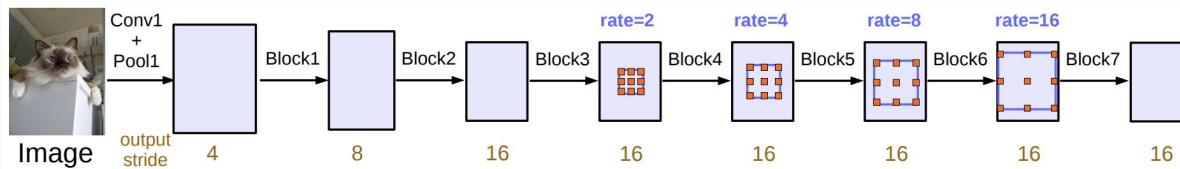


Let's do convolution not with the neighbours but with pixels far from current location

# Atrous convolution



(a) Going deeper without atrous convolution.

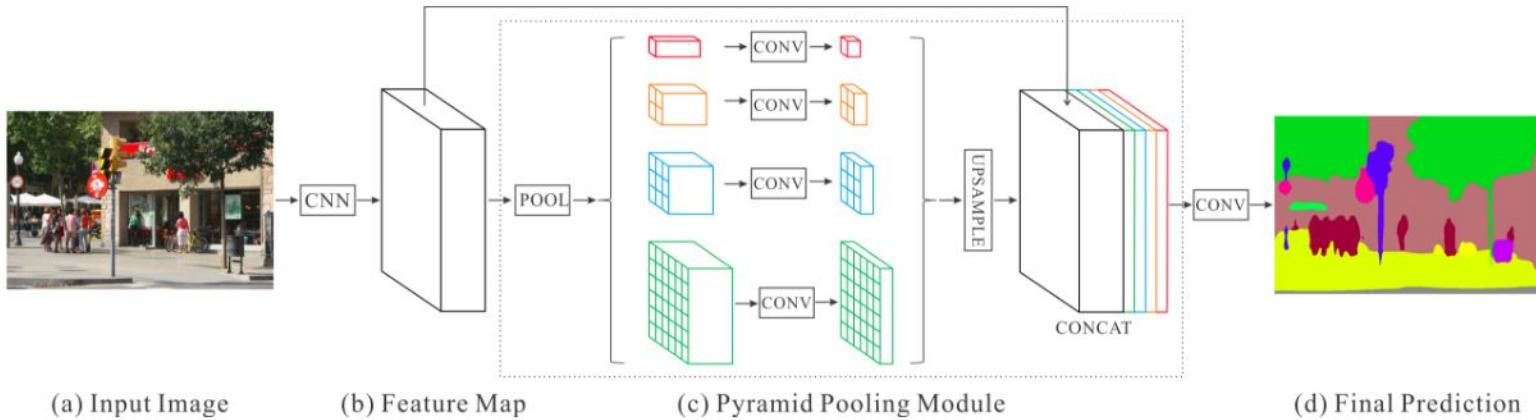


(b) Going deeper with atrous convolution. Atrous convolution with  $rate > 1$  is applied after block3 when  $output\_stride = 16$ .

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

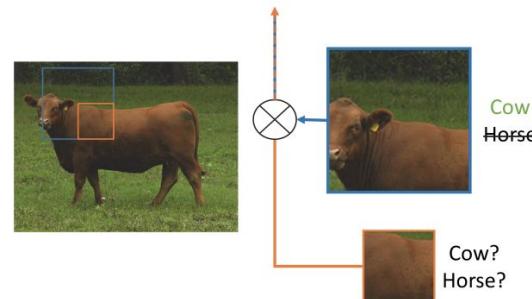


(a) Input Image

(b) Feature Map

(c) Pyramid Pooling Module

(d) Final Prediction



# DeepLab

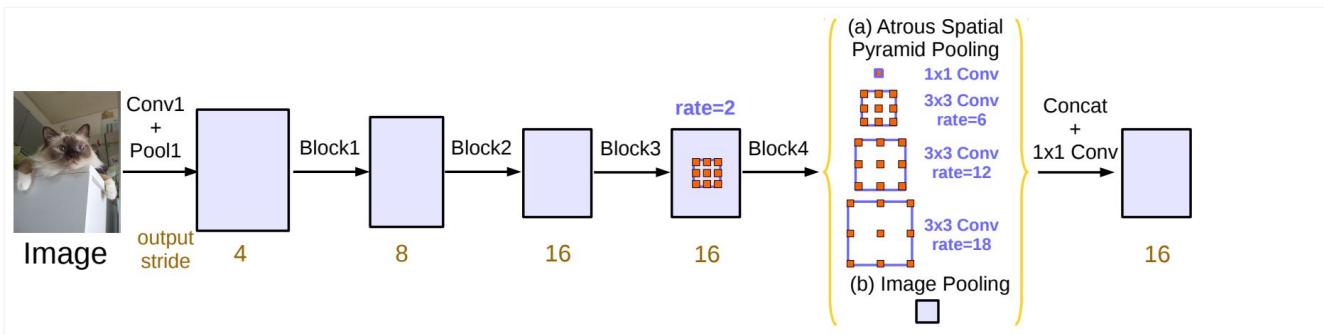
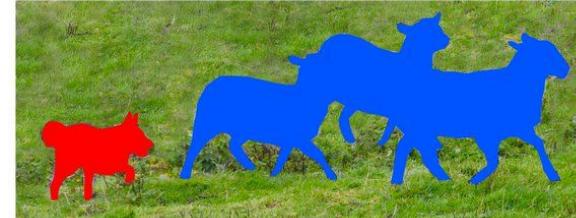


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

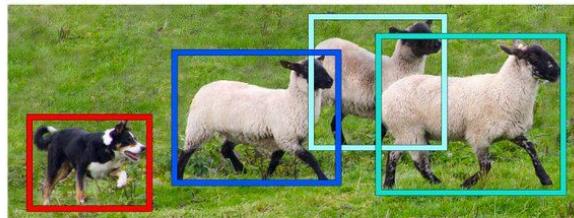
# Instance Segmentation



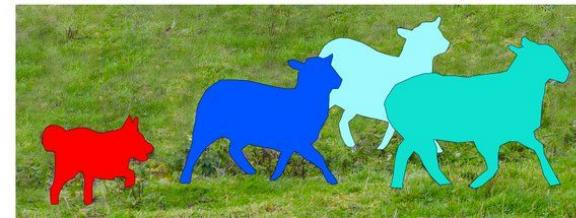
Image Recognition



Semantic Segmentation



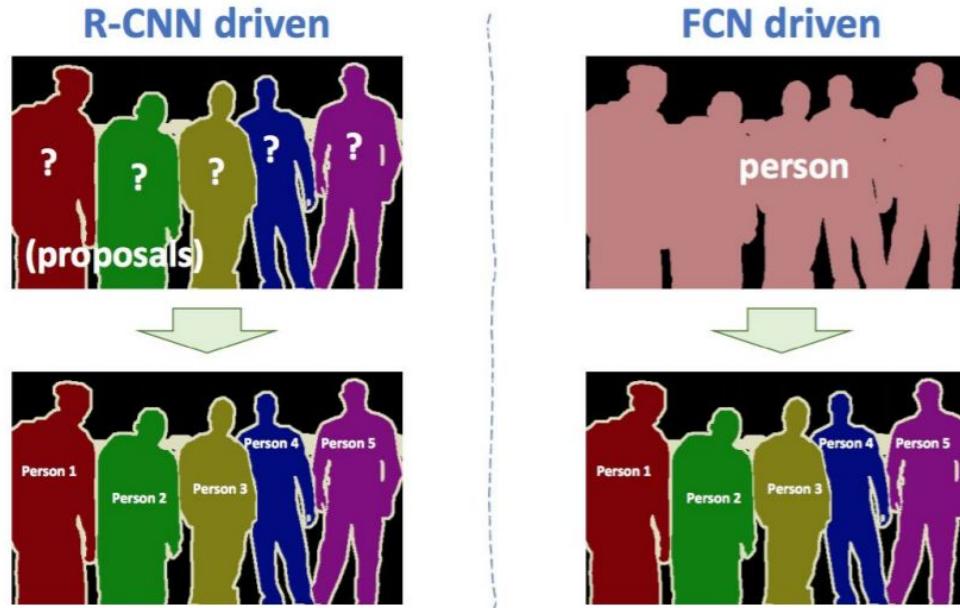
Object Detection



Instance Segmentation

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

# Instance Segmentation

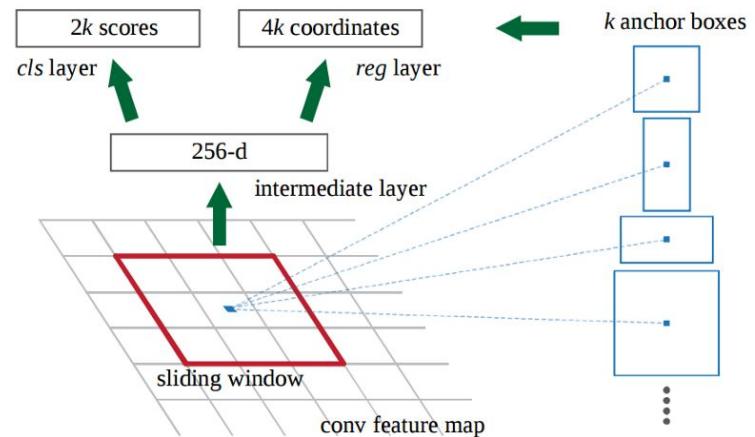
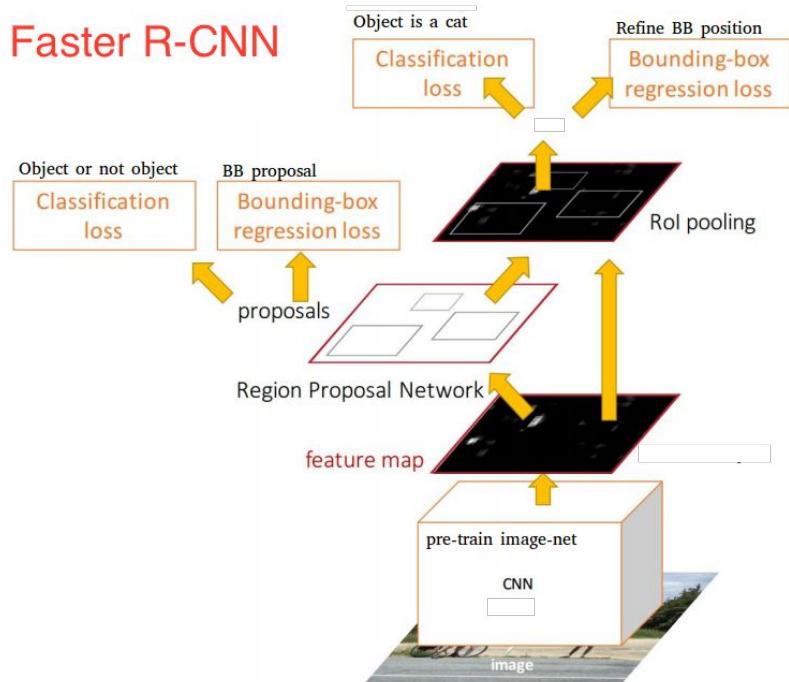


Instance segmentation has two approaches:

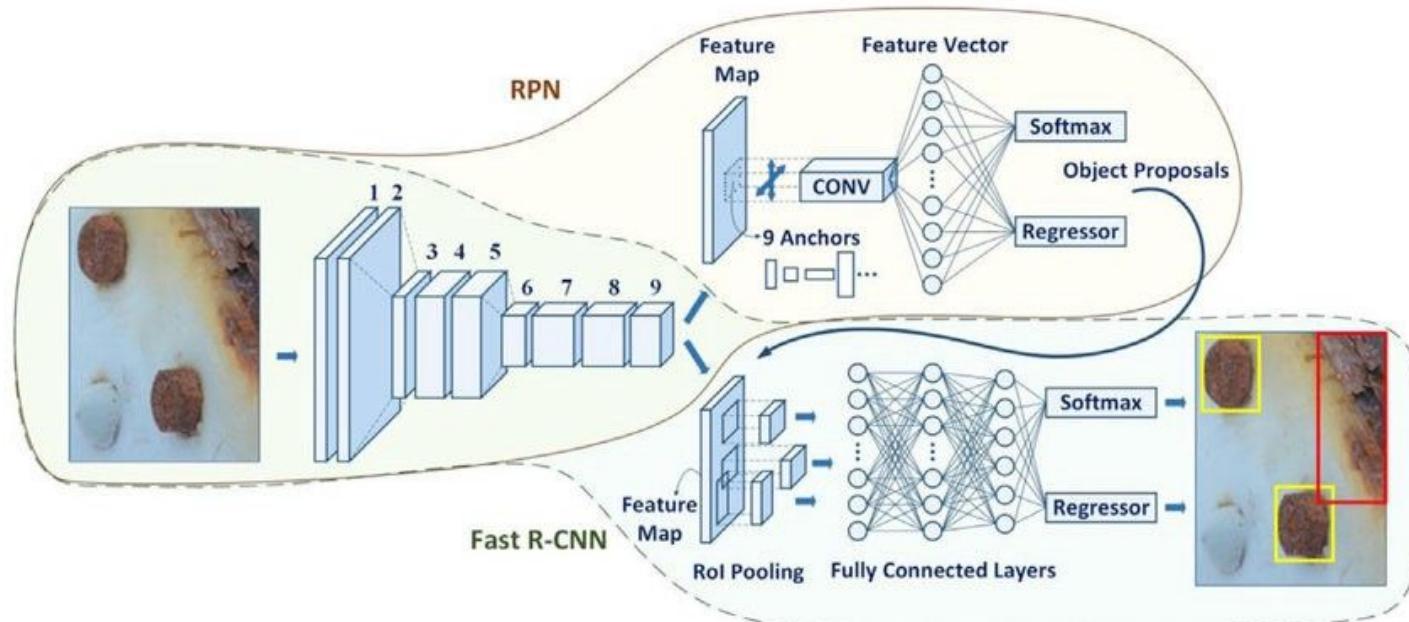
- Inspired by object detection (R-CNN driven).
- Inspired by segmentation (FCN driven)

# Reminder:

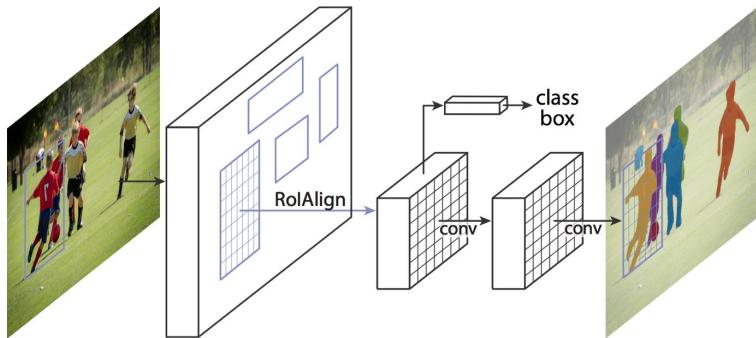
## Faster R-CNN



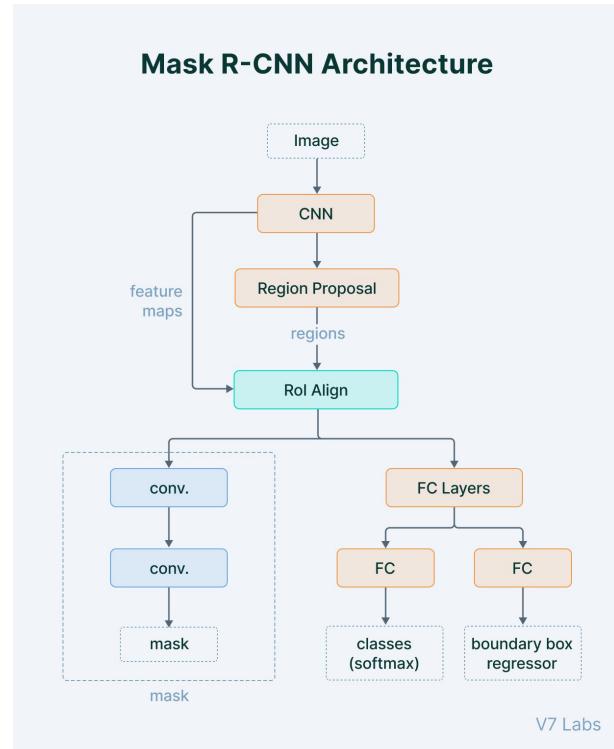
# Faster RCNN



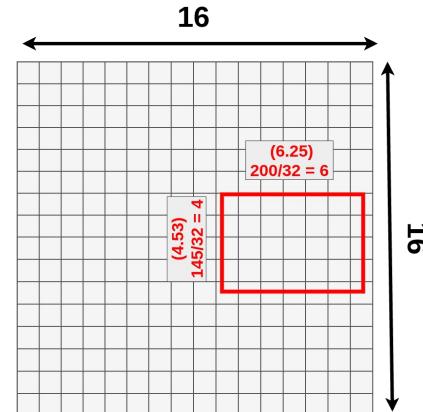
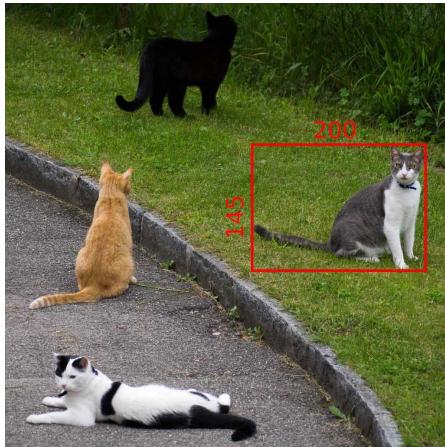
# Mask R-CNN



Adding mask prediction head



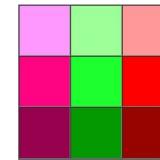
# RoI Pooling



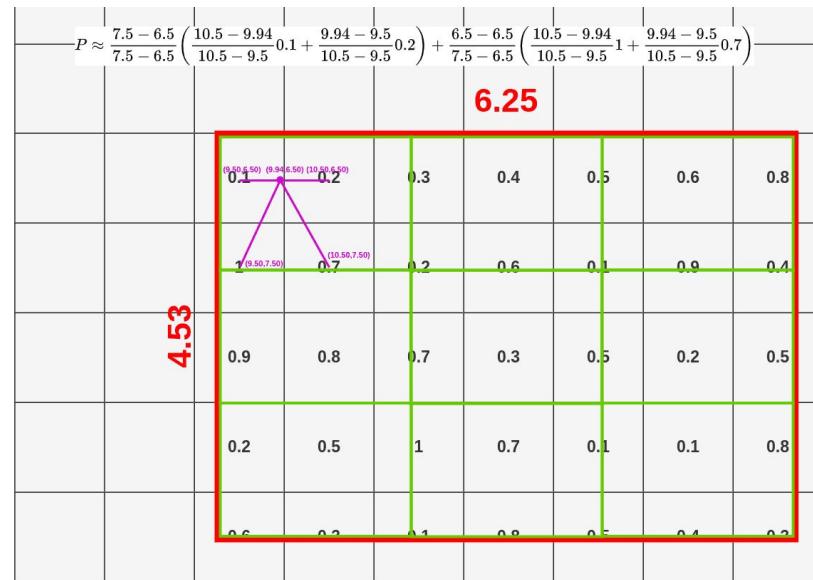
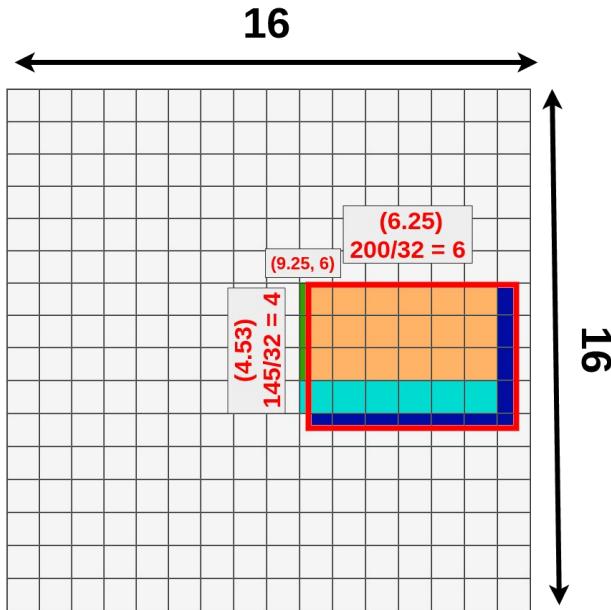
4x6 RoI

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

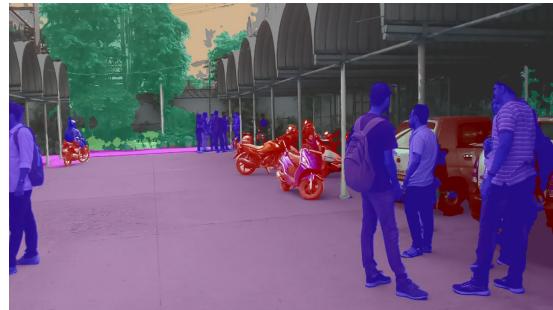
3x3 RoI Pooling



# RoI Align



# Panoptic Segmentation



Semantic



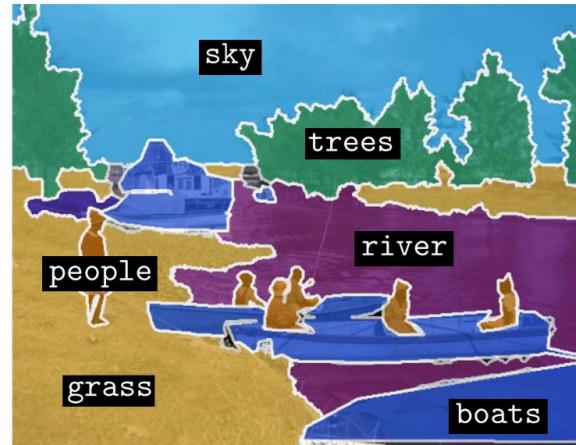
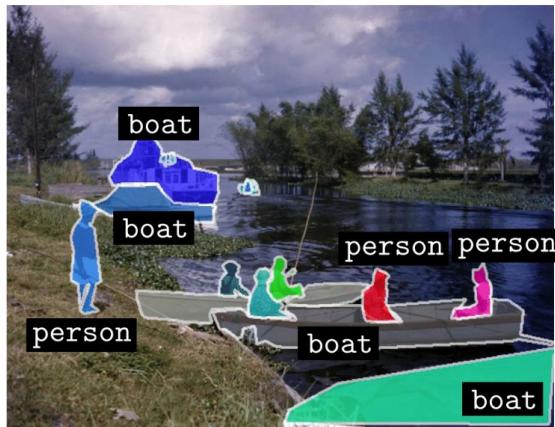
Instance



Panoptic

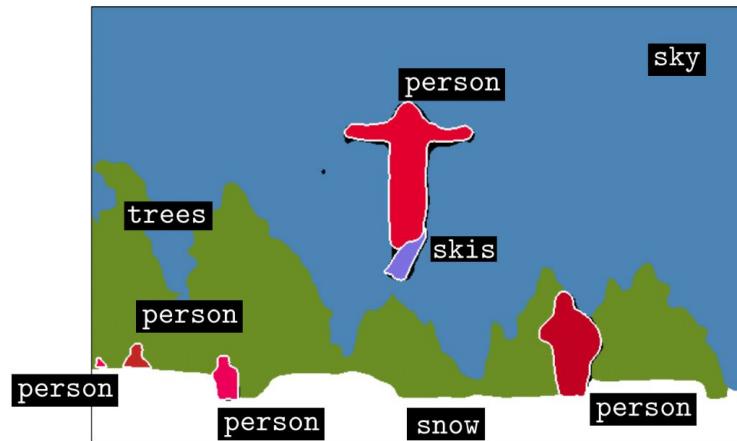
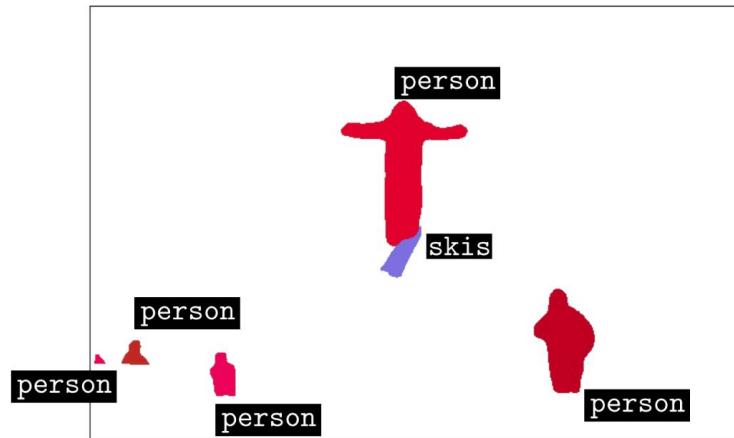
# Panoptic Segmentation

What do every segmentation do?



# Panoptic Segmentation

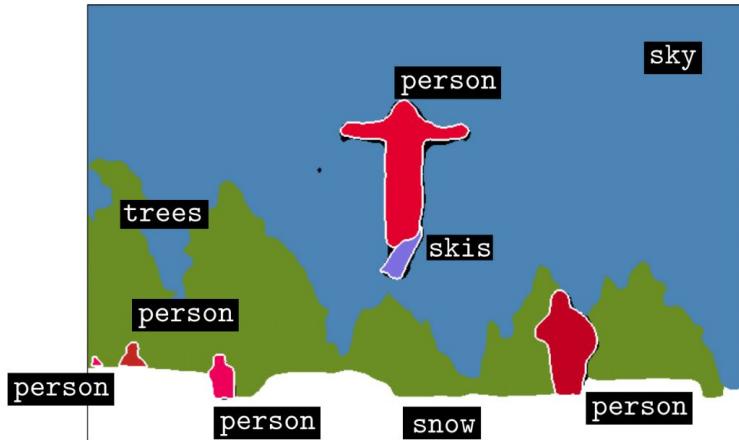
What do instance segmentation see?



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

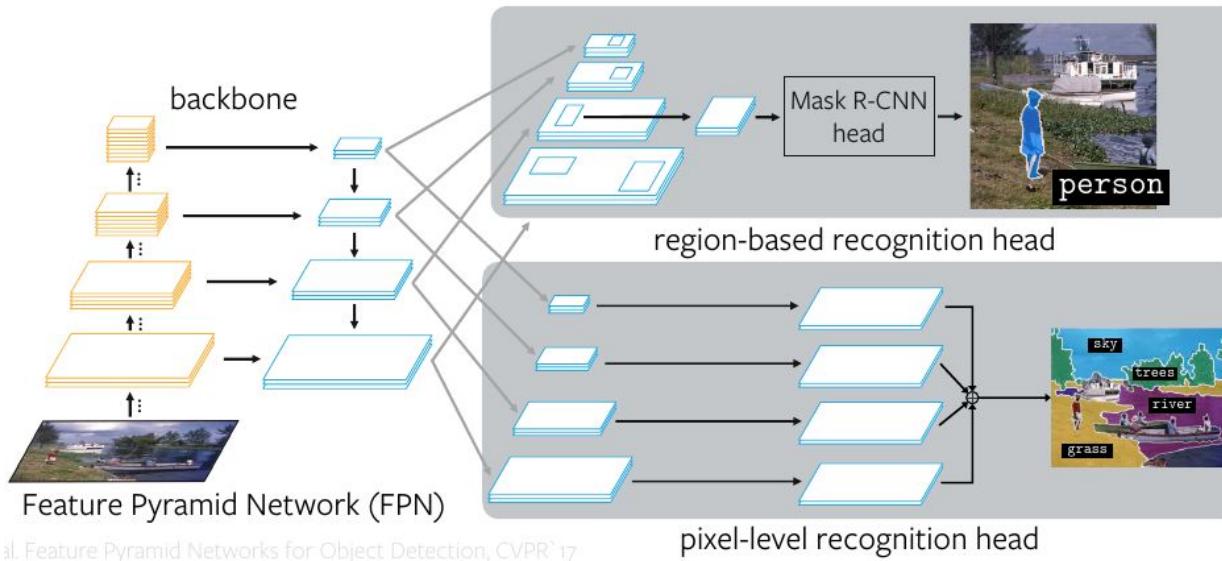
Panoptic segmentation should include both metrics simultaneously

# Panoptic Segmentation

How to evaluate performance of panoptic segmentation model?

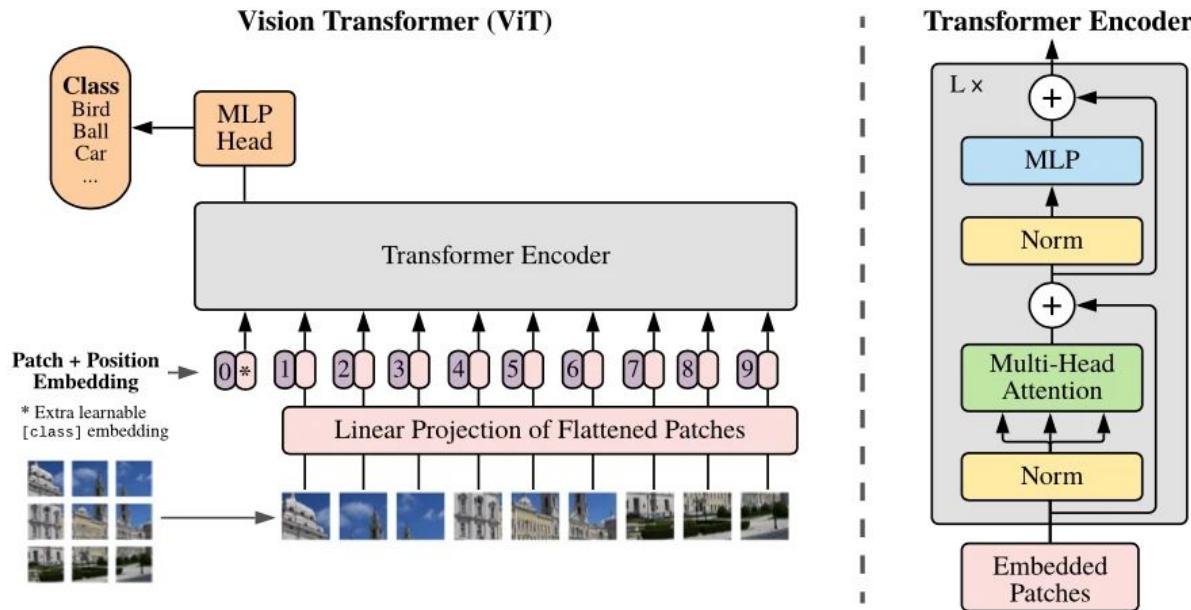
$$\begin{aligned} \text{PQ} &= \frac{\sum_{(p,g) \in TP} \text{IoU}(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \\ &= \underbrace{\frac{\sum_{(p,g) \in TP} \text{IoU}(p, g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition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

# Panoptic architecture

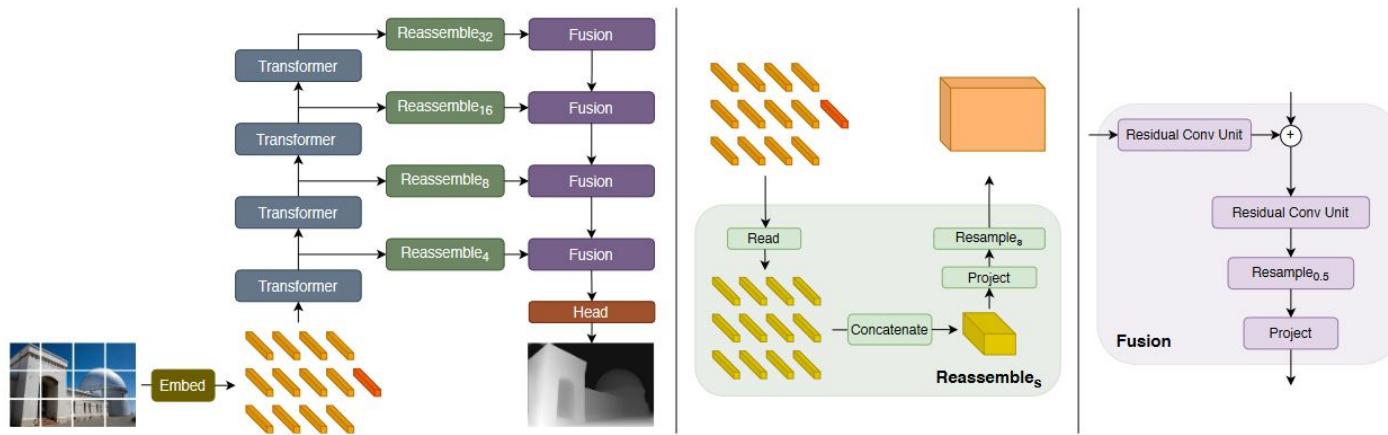
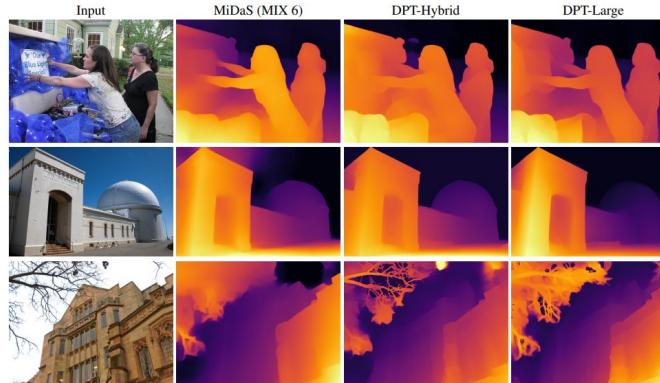


§. Feature Pyramid Networks for Object Detection, CVPR'17

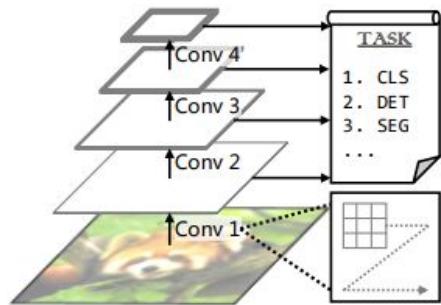
# Reminder: ViT



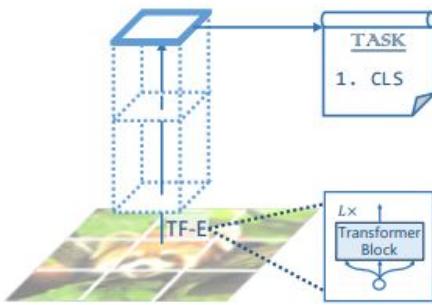
# DPT (Vision Transformers for Dense Prediction)



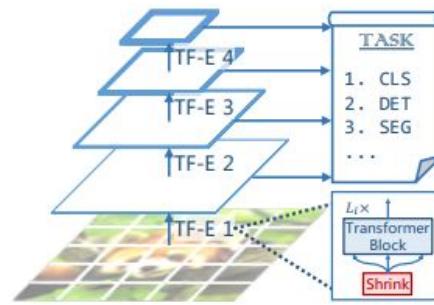
# Pyramid Vision Transformer



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

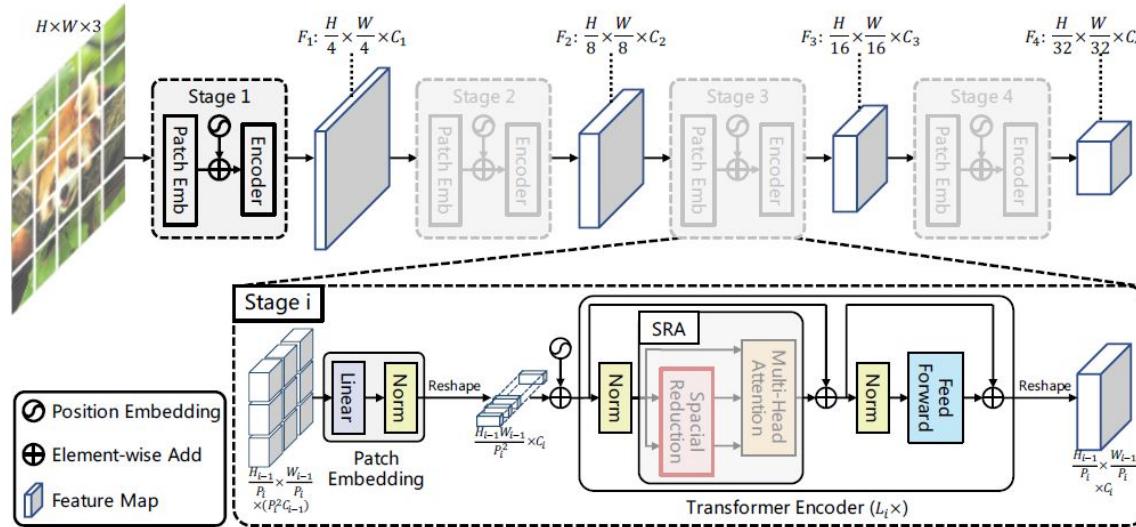


(b) Vision Transformer [13]

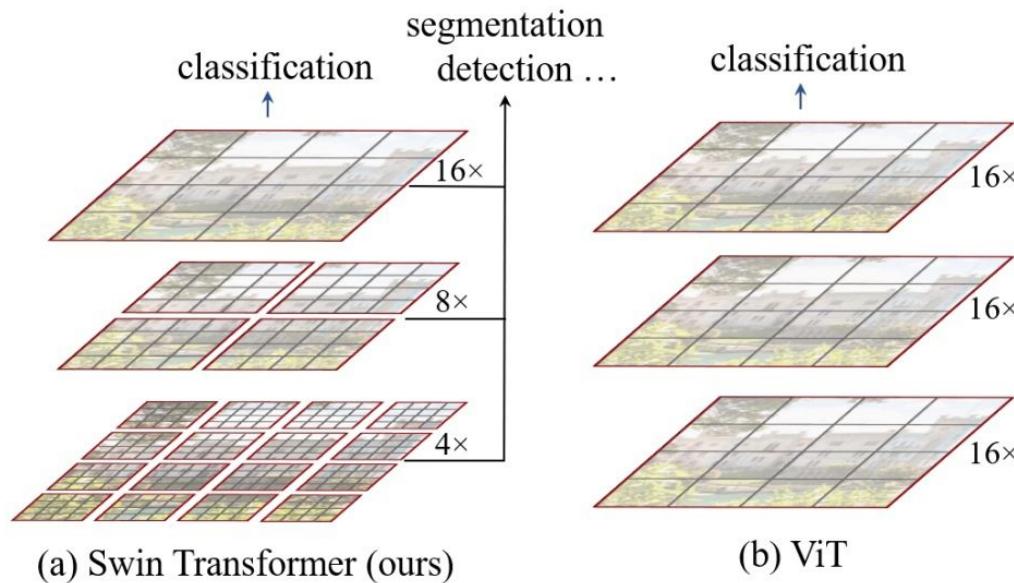


(c) Pyramid Vision Transformer (ours)

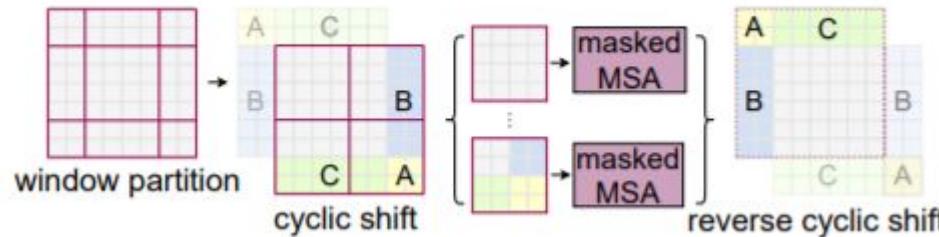
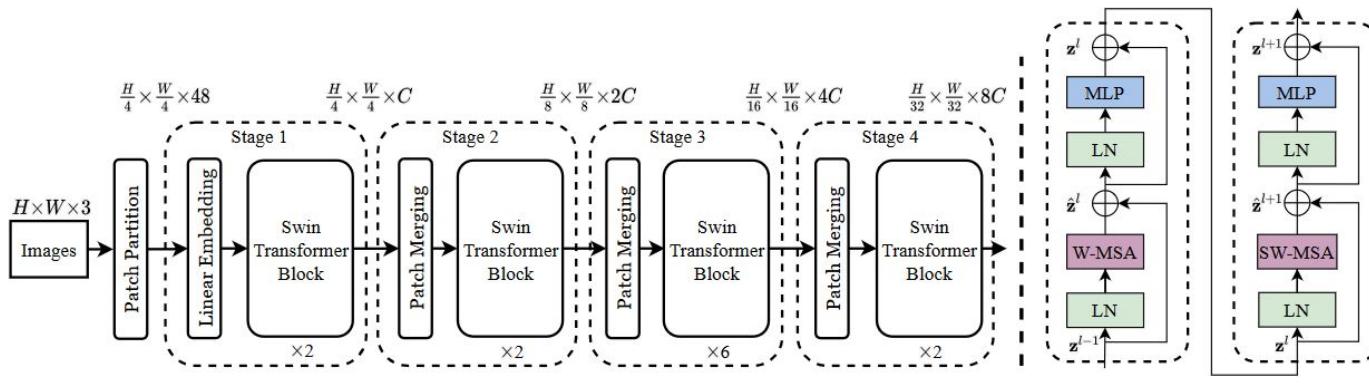
# Pyramid Vision Transformer



# Swin



# Swin



# HRFormer

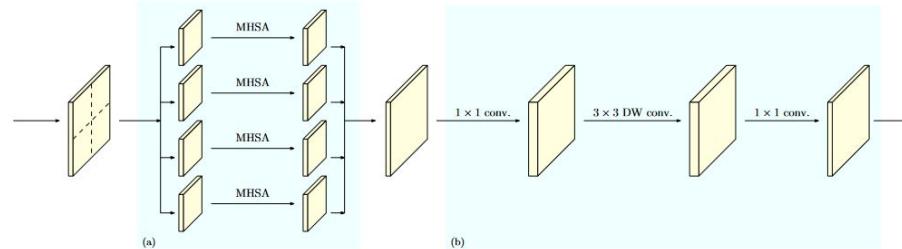
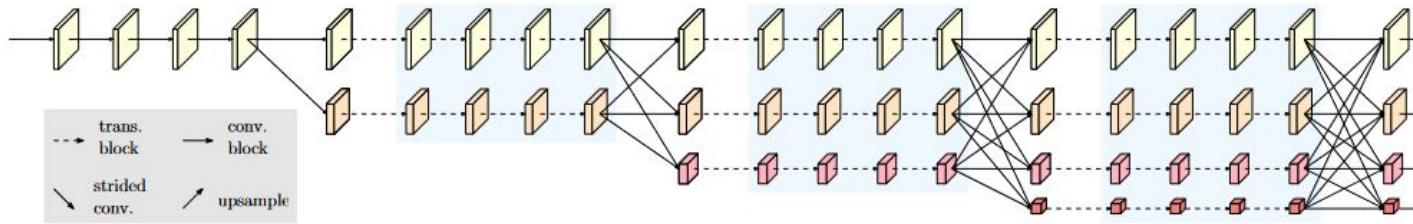


Figure 1: **Illustrating the HRFormer block.** The HRFormer block is composed of (a) local-window self-attention and (b) feed-forward network (FFN) with depth-wise convolution. The local-window self-attention scheme is inspired by the interlaced sparse self-attention [56, 21].

# HRFormer

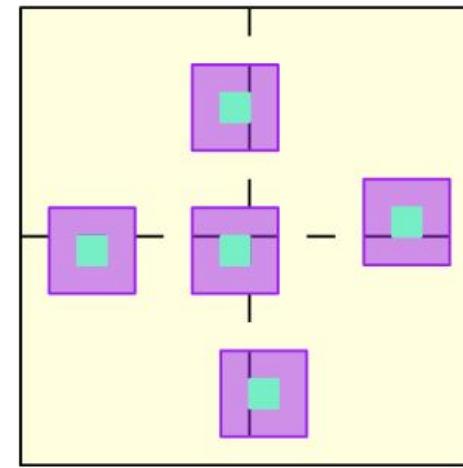
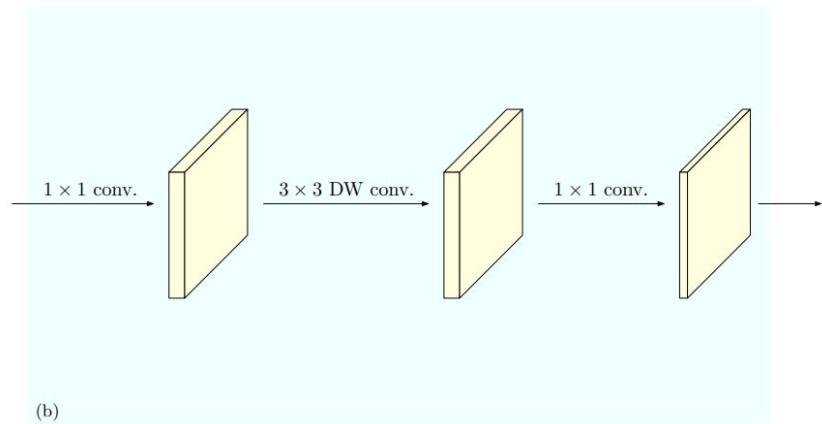


Figure 3: Illustrating that FFN with  $3 \times 3$  depth-wise convolution connects the non-overlapping windows.

# Recap

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