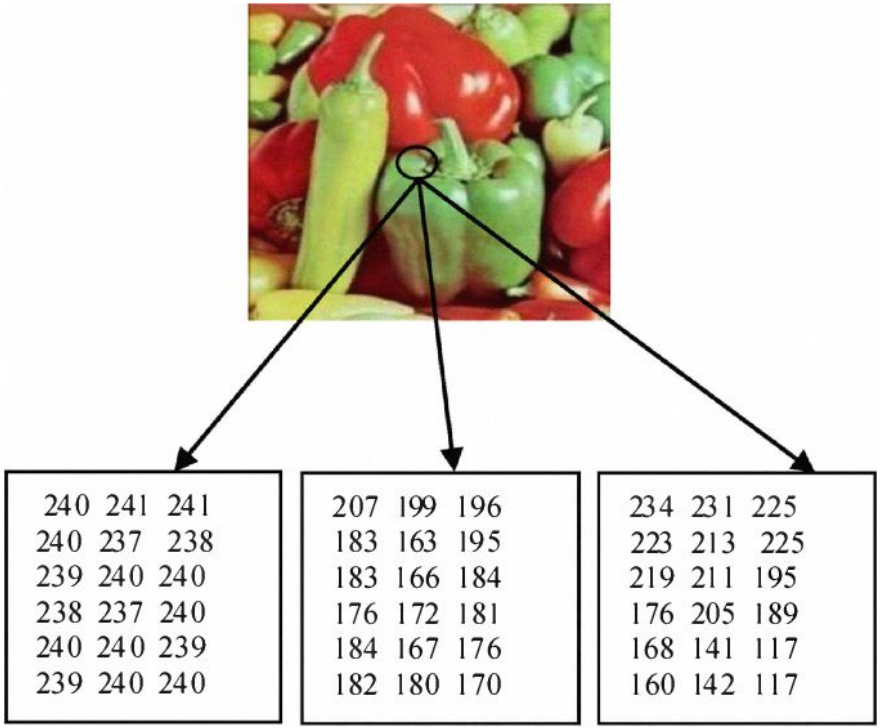
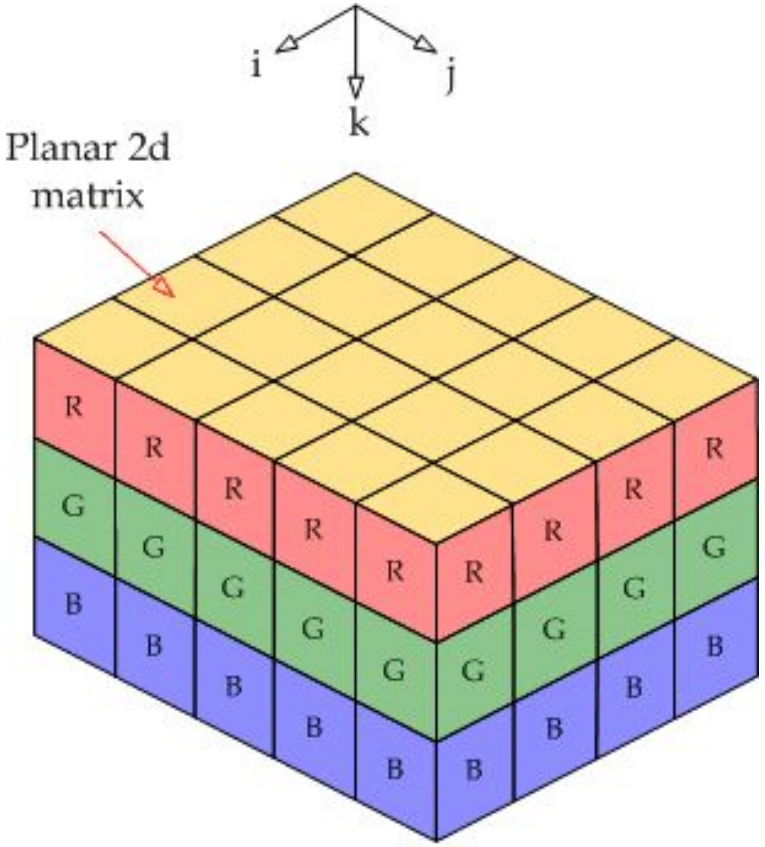
Deep Learning for Computer Vision

Whats an Image?





Graphical presentation of RGB 3d matrix

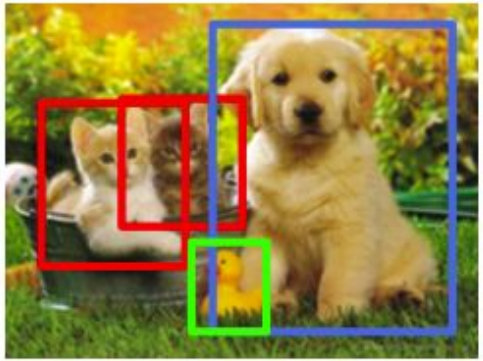
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Image Segmentation

Image segmentation is a computer vision task in which we label specific regions of an image according to what's being shown.

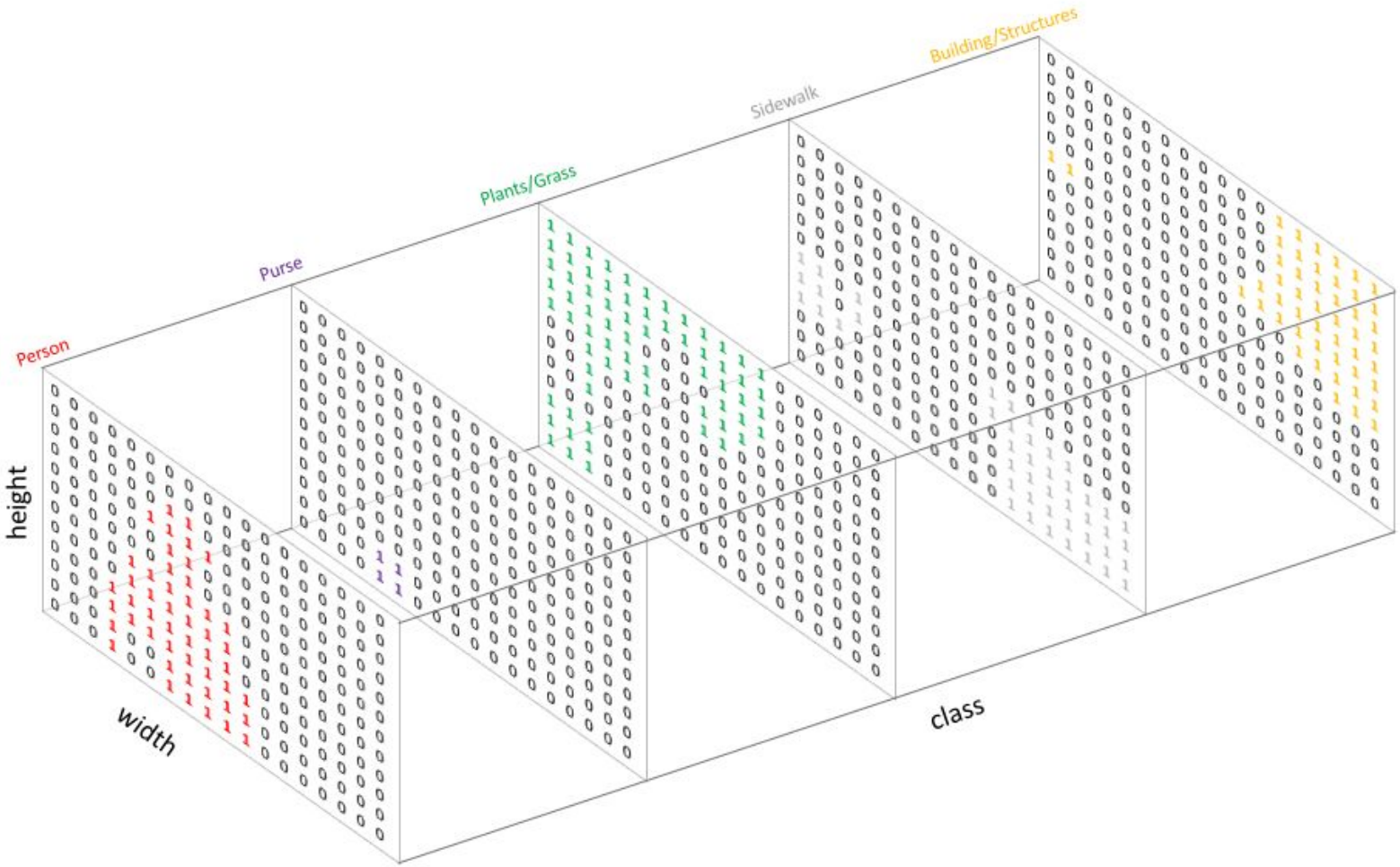
Input

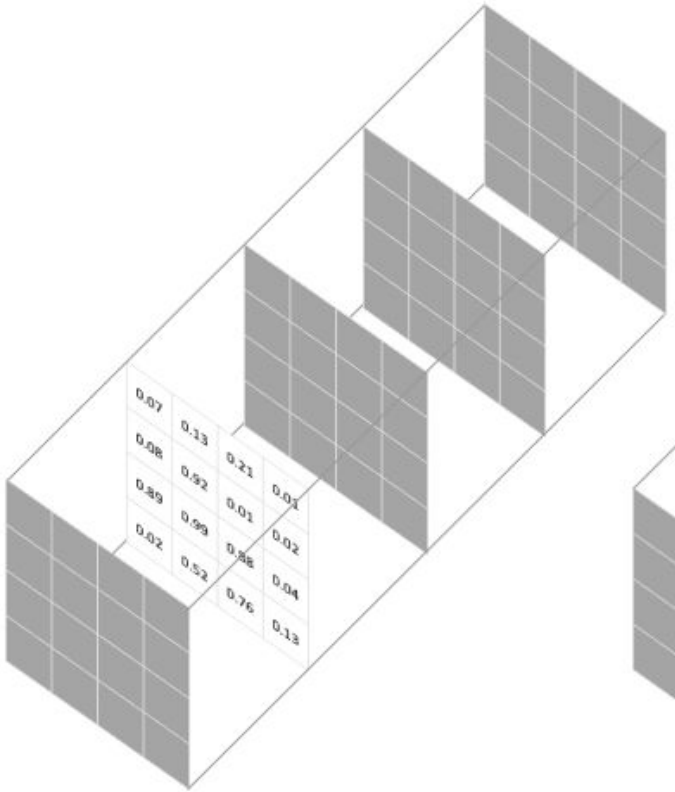
segmented

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

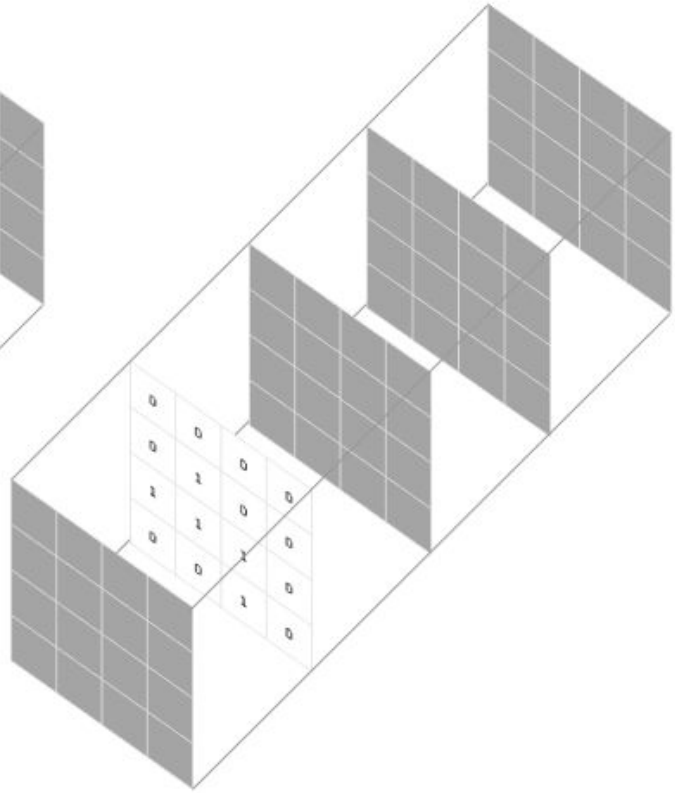
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1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


Person
Purse
Plants/Grass
Sidewalk
Building/Structures
height
width
class



Prediction for a selected class



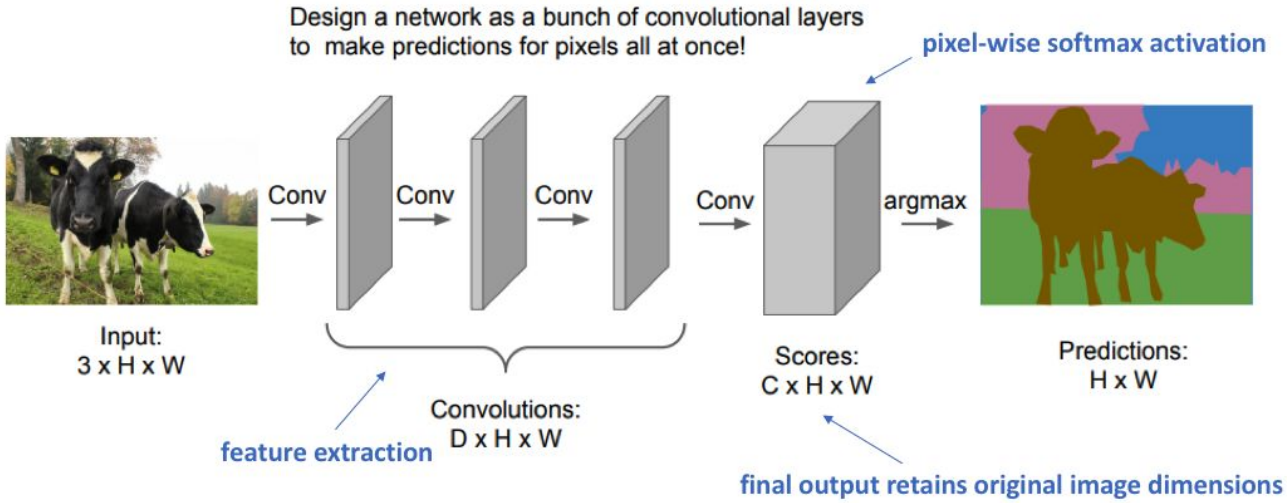
Target for the corresponding class

Soft Dice coefficient is calculated for each class mask

$$1 - \frac{2\sum_{pixels} y_{true} y_{pred}}{\sum_{pixels} y_{true}^2 + \sum_{pixels} y_{pred}^2}$$

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

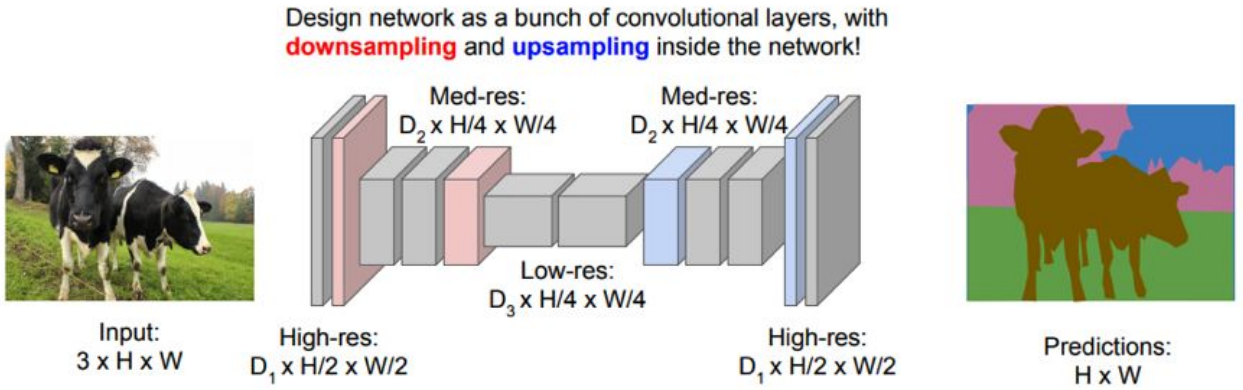
Inefficient Segmentation Architecture



Downside: Preserving image dimensions throughout entire network will be computationally expensive.

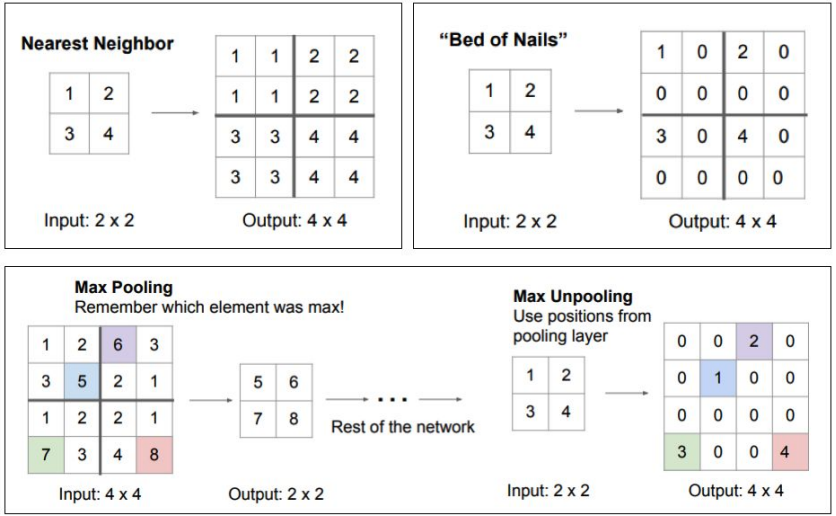
it's quite computationally expensive
to preserve the full resolution
throughout the network.

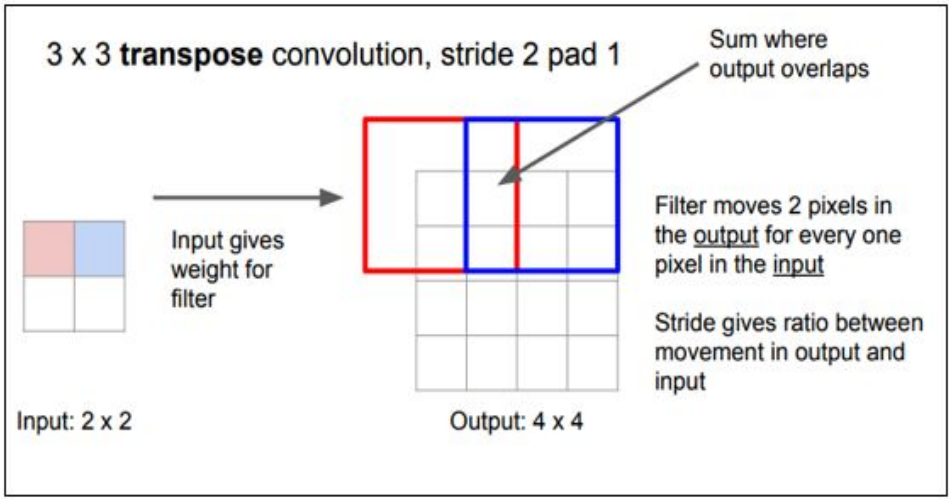
- In order to maintain expressiveness, we typically need to increase the number of feature maps (channels) as we get deeper in the network.
- Encoder->Decoder architecture could help solving this issue
- We loose fine grained informations yielding low resolution predictions



Solution: Make network deep and *work at a lower spatial resolution* for many of the layers.

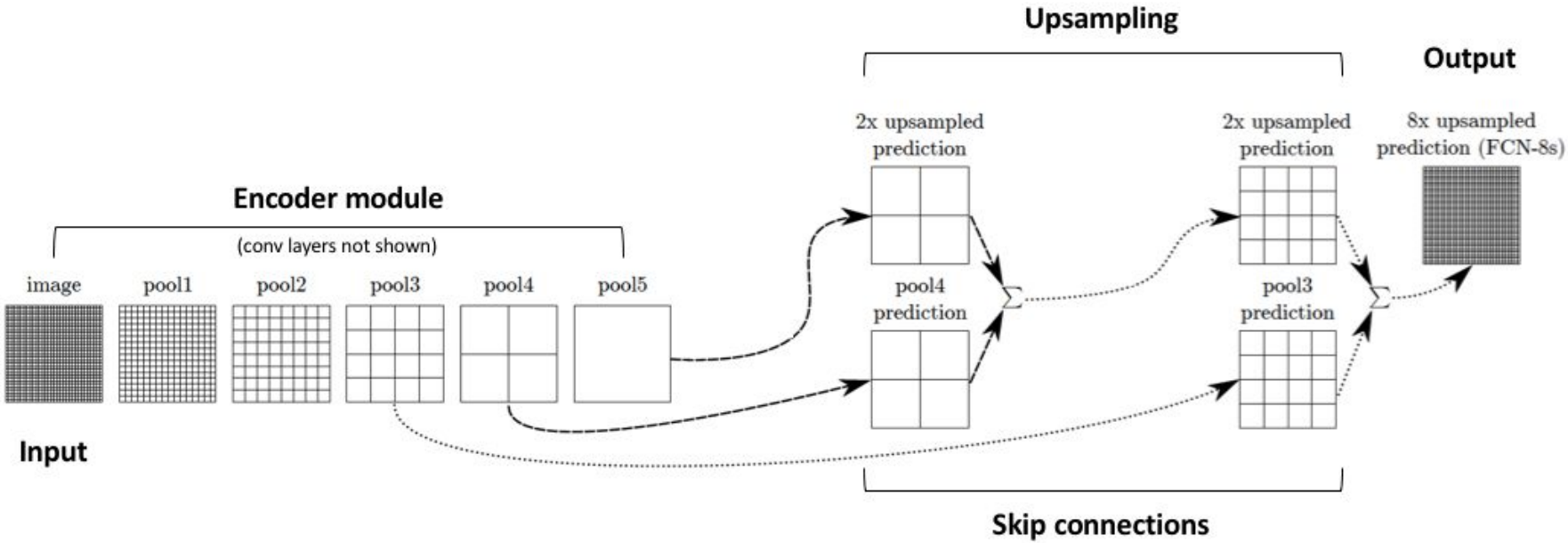
Upsampling / Upconv



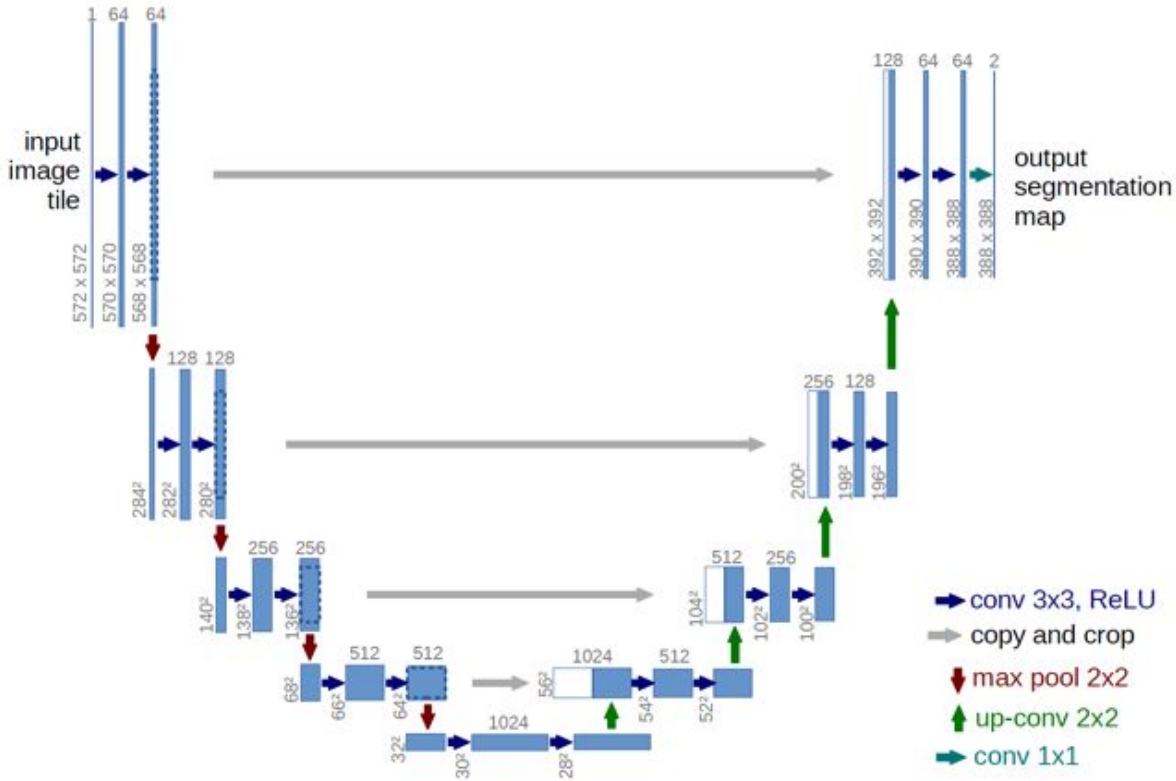


Skip connections:

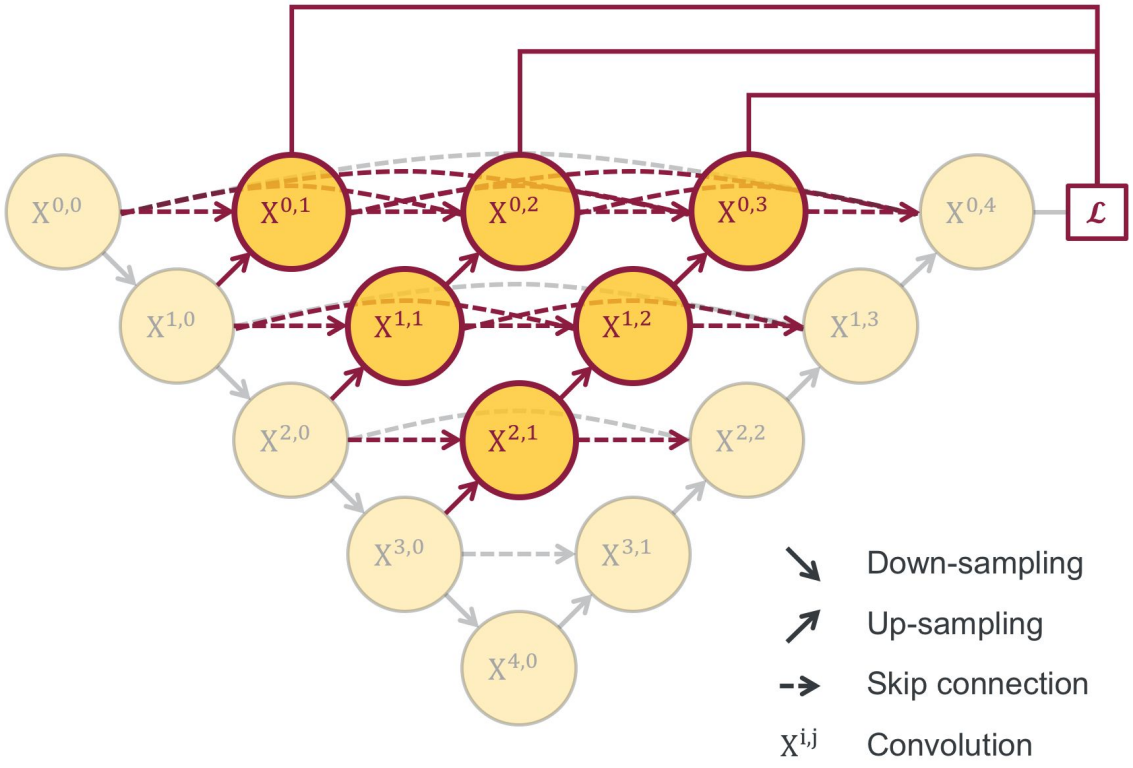
- Adding skip connections from earlier layers
- Layers prior downsampling should provide necessary detail to reconstruct shapes
- More fine-grain detailed output.



UNET architecture



Newer iterations of UNET add more skip connections



PSPNet

Max pooling with 4 different window sizes and strides to capture different scales

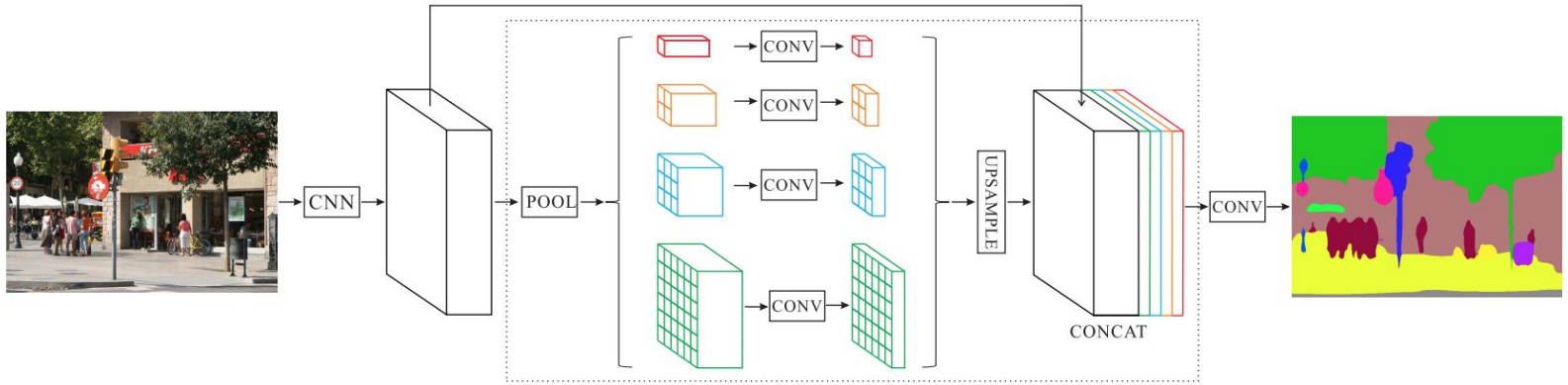
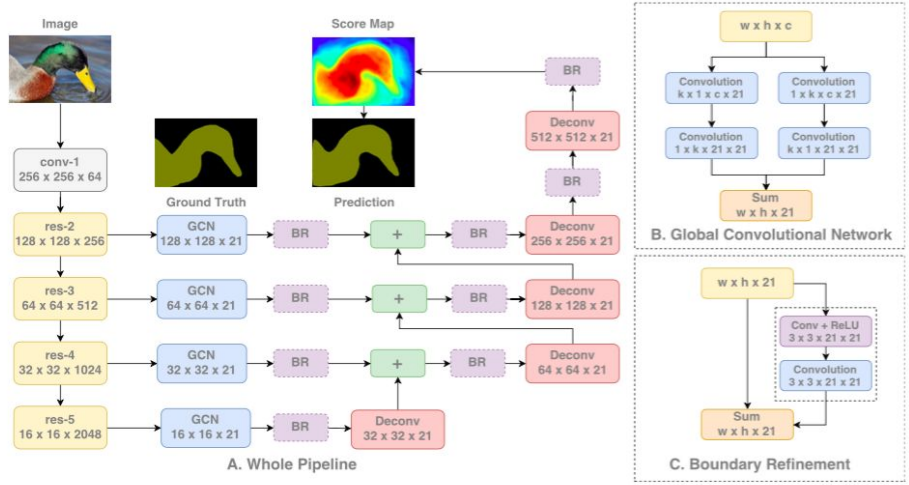


Figure 3. Overview of our proposed PSPNet. Given an input image (a), we first use CNN to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d).

Large Kernel Matters

- Larger kernels are computationally expensive and have a lot of parameters. Therefore, k x k convolution is approximated with sum of 1 x k + k x 1 and k x 1 and 1 x k convolutions

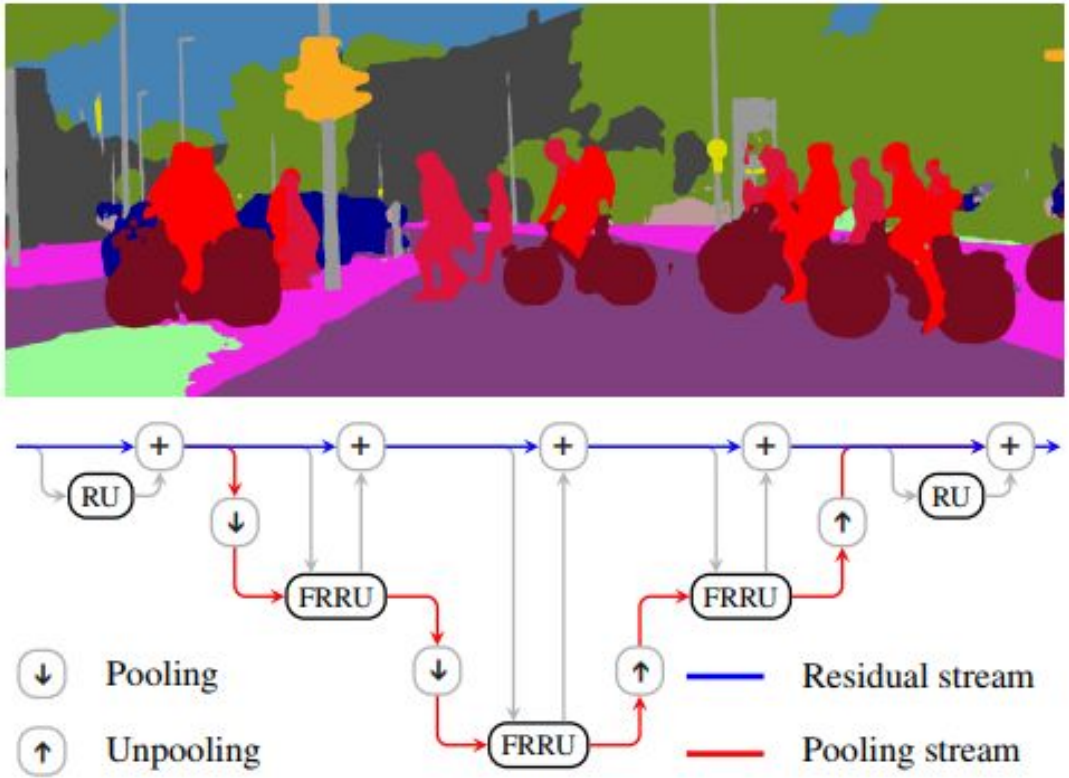
- Since fully connected layers cannot be present in a segmentation architecture, convolutions with very large kernels are adopted instead.



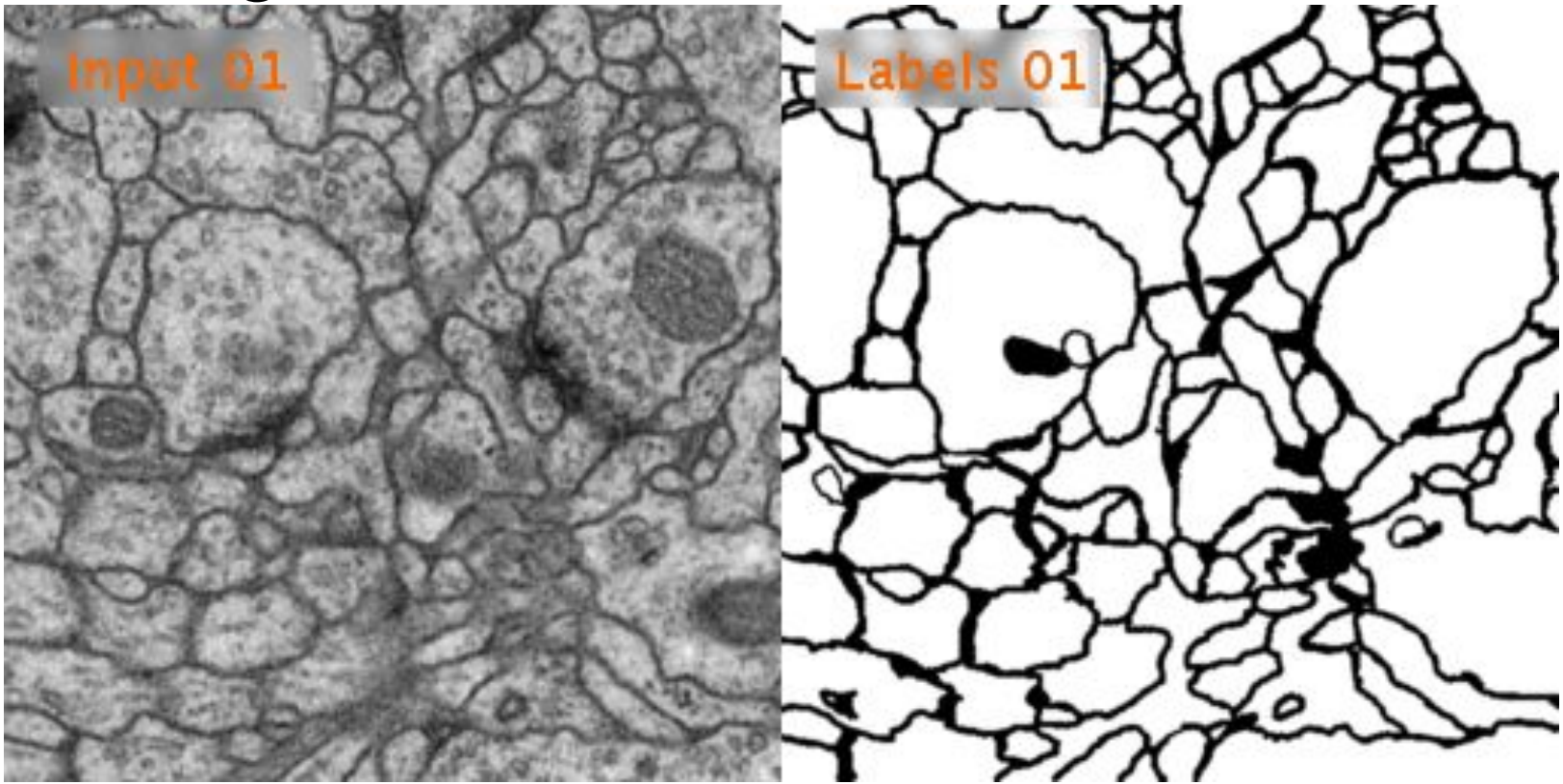

```
class GCN(nn.Module):
    def __init__(self,c,out_c,k=7): #out_Channel=21 in paper
        super(GCN, self).__init__()
        self.conv_l1 = nn.Conv2d(c, out_c, kernel_size=(k,1), padding =((k-1)/2,0))
        self.conv_l2 = nn.Conv2d(out_c, out_c, kernel_size=(1,k), padding =(0,(k-1)/2))
        self.conv_r1 = nn.Conv2d(c, out_c, kernel_size=(1,k), padding =((k-1)/2,0))
        self.conv_r2 = nn.Conv2d(out_c, out_c, kernel_size=(k,1), padding =(0,(k-1)/2))

    def forward(self, x):
        x_l = self.conv_l1(x)
        x_l = self.conv_l2(x_l)

        x_r = self.conv_r1(x)
        x_r = self.conv_r2(x_r)

        x = x_l + x_r

        return x
```

FRRN

- Different scale processing
- Residual stream (high localization accuracy)
- Pooling Stream (high classification accuracy)

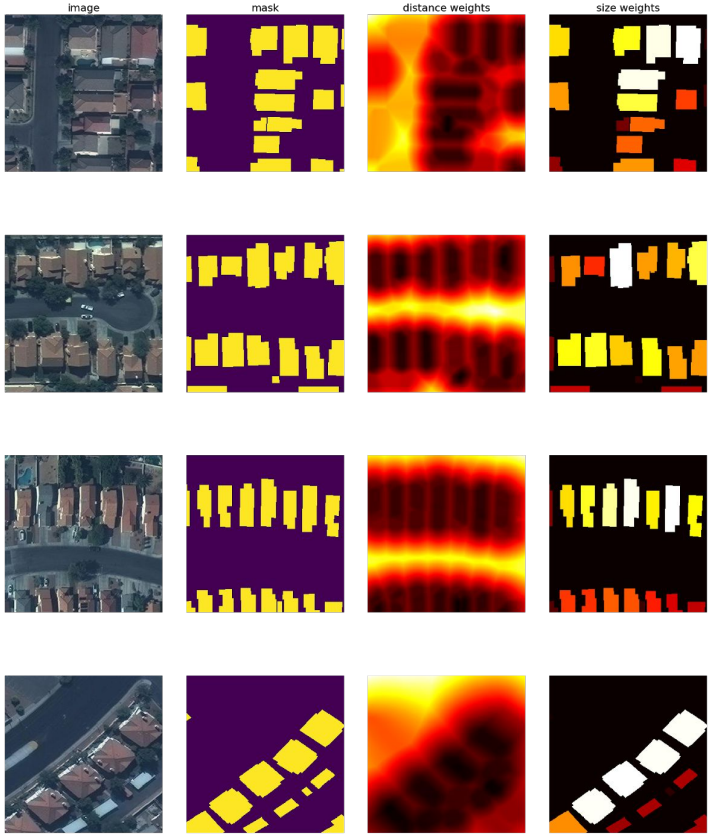


ISBI segmentation challenge



Custom loss / Loss weightening

- Depending on the application you might benefit from creating custom loss functions ,and or weightening the loss



Object Detection

- Two-Stage
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN

- One-Stage
  - YOLO (v1, v2, v3)
  - SSD
  - RetinaNet

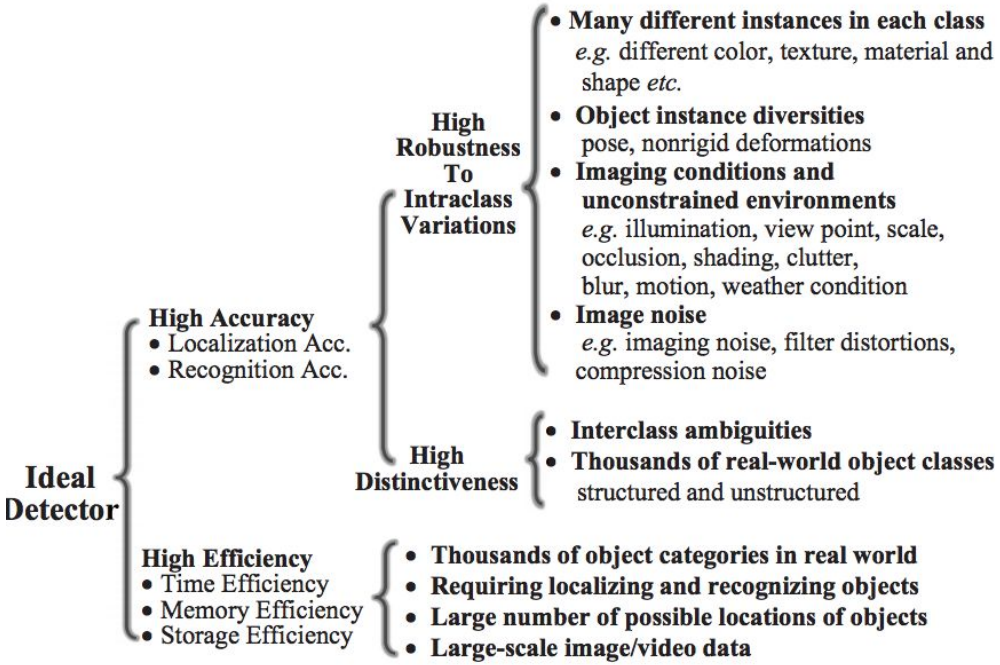
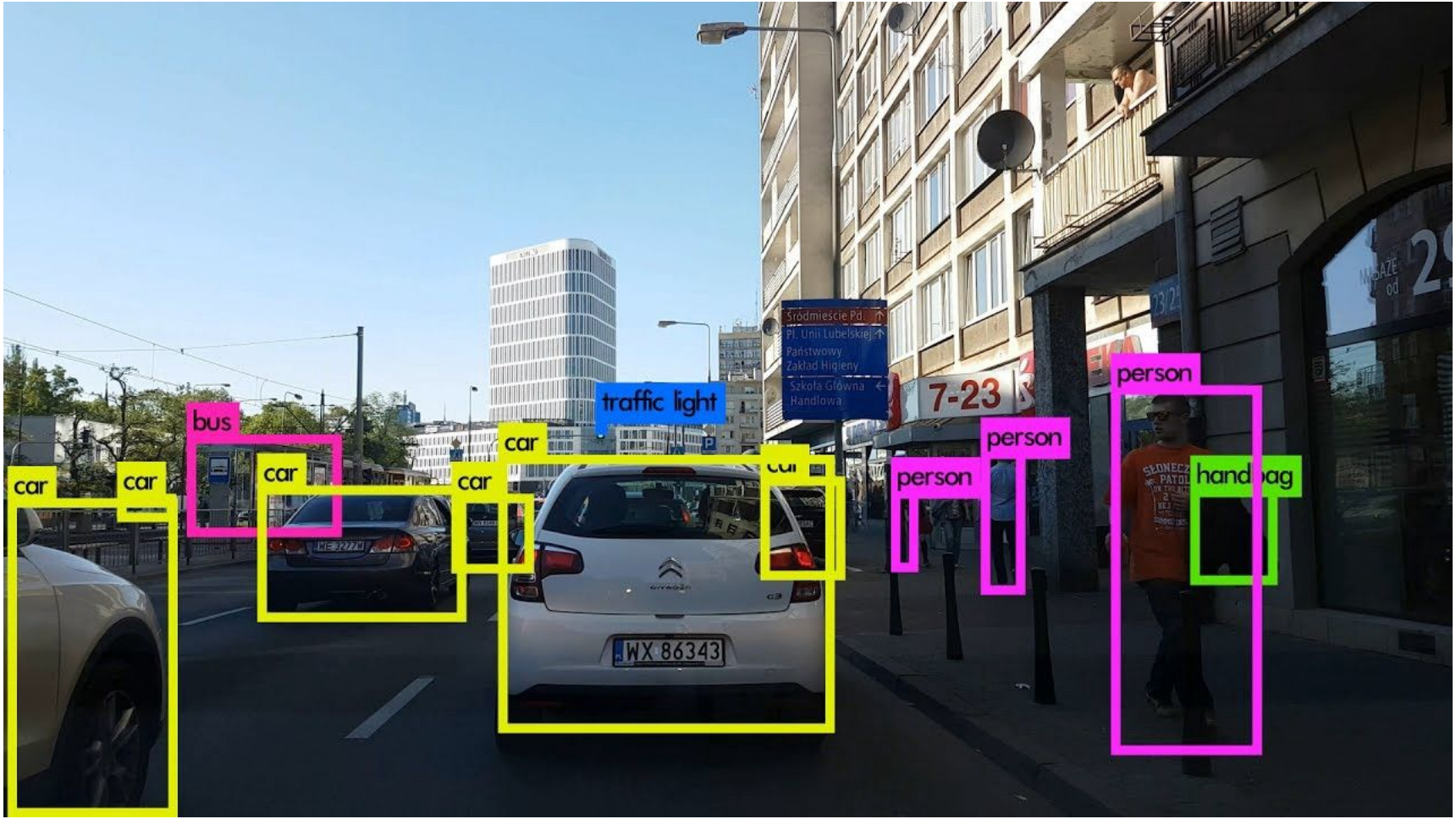
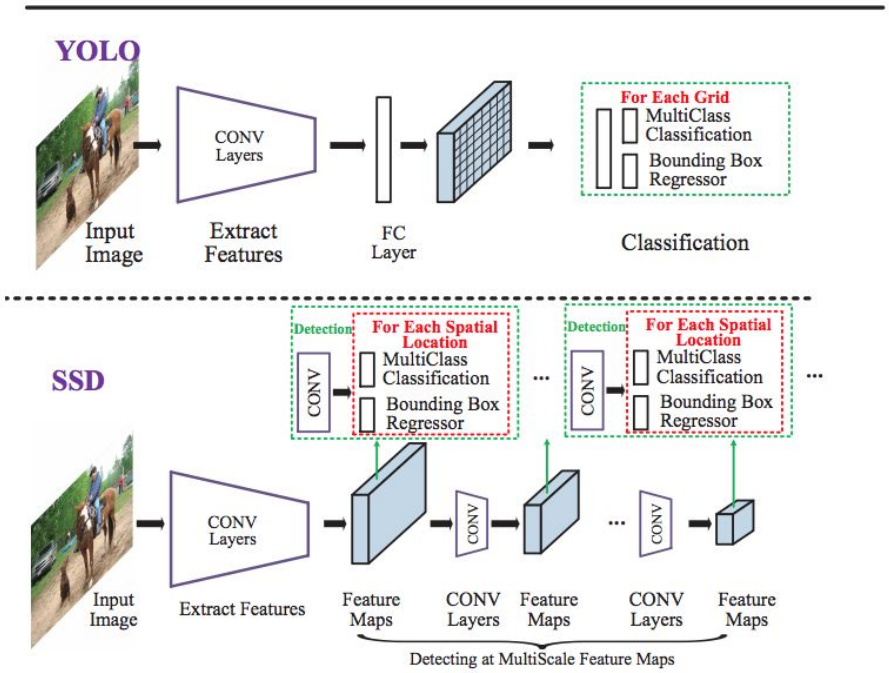
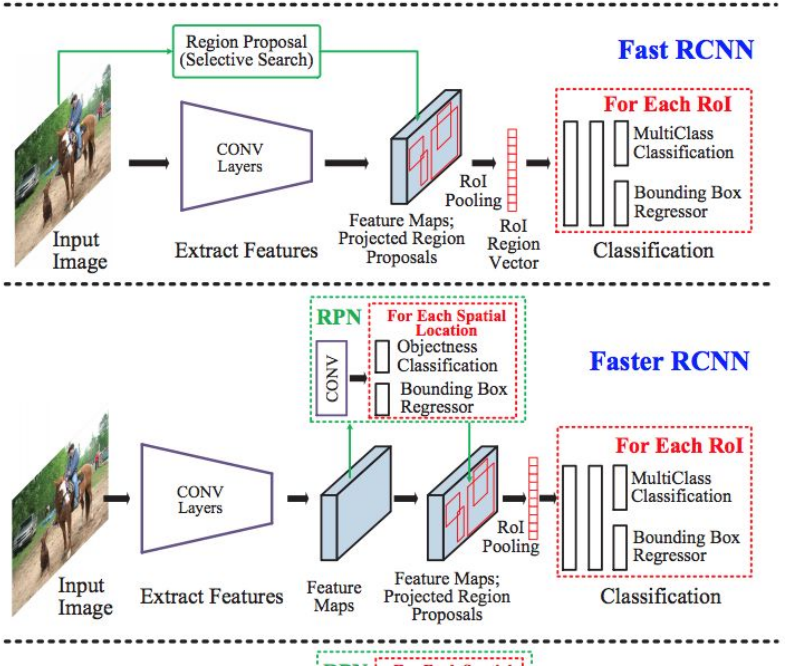


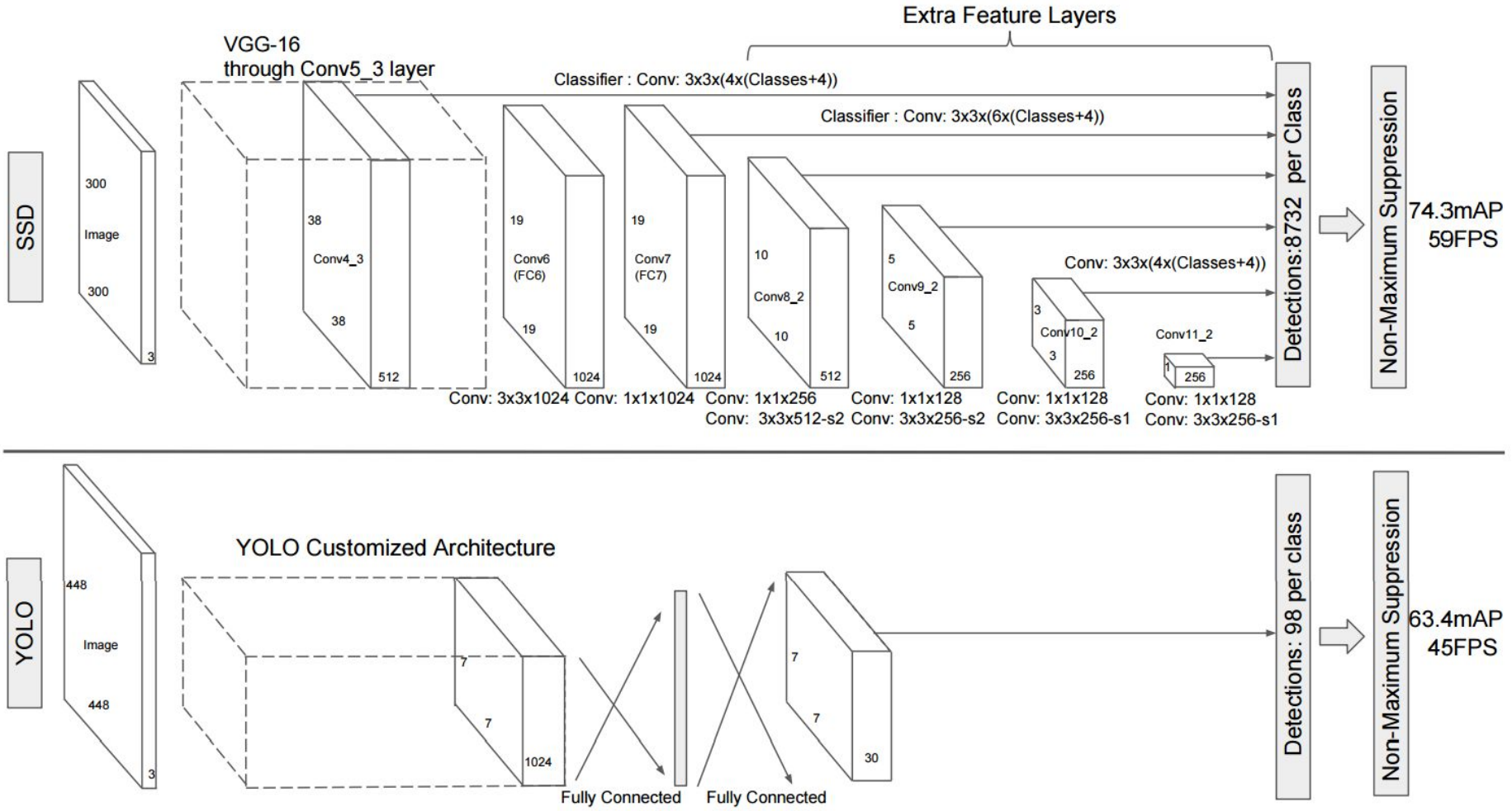
Fig. 4 Summary of challenges in generic object detection.


car
car
bus
car
car
car
traffic light
car
person
person
person
handbag
Śródmieście Pd.
Pl. Unii Lubelskiej
Państwowy
Zakład Higieny
Szkoła Główna
Handlowa
7-23
WX 86343

One Stage vs Two Stage Detection



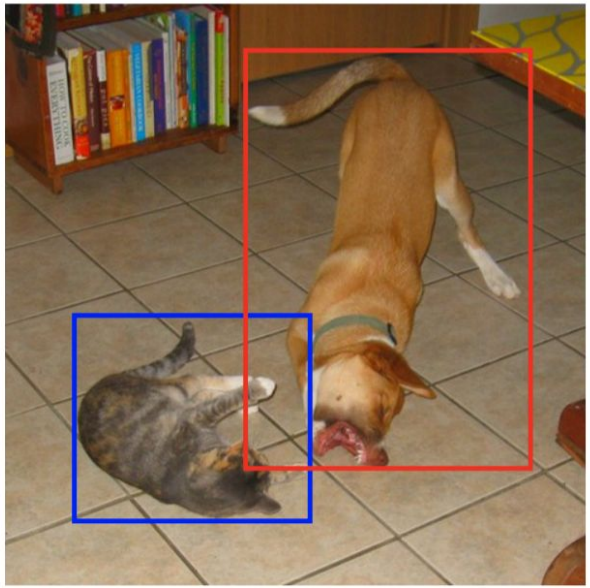



SSD
300
Image
300
3
VGG-16
through Conv5_3 layer
38
Conv4_3
38
512
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
1
Conv11_2
256
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
YOLO
448
Image
448
3
YOLO Customized Architecture
7
7
1024
Fully Connected
Fully Connected
7
7
30
Detections: 98 per class
Non-Maximum Suppression
63.4mAP
45FPS

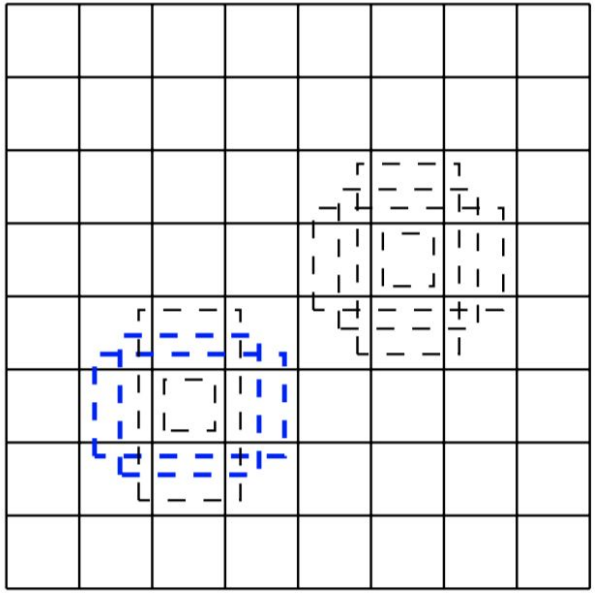
Object detection steps (SSD/Retina like setting)

- Map labels (bounding boxes) to anchors to unify the target
- Calculate a target as a offset (how to move anchor so its closer to our target)
- For each anchor box predict the class it belongs to.
- Calculate loss for each of the problem
    - F1 Loss for offset regression
    - Focal/CE for class prediction

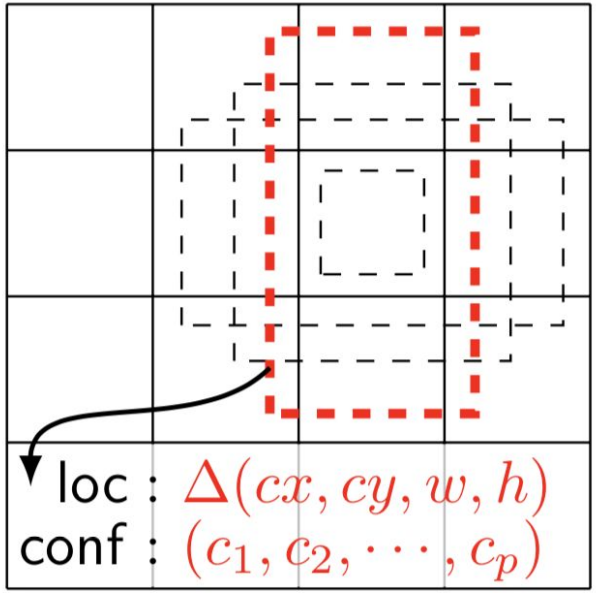
- Create the predictions by applying the offset to the anchor boxes



(a) Image with GT boxes



(b) 8×8 feature map

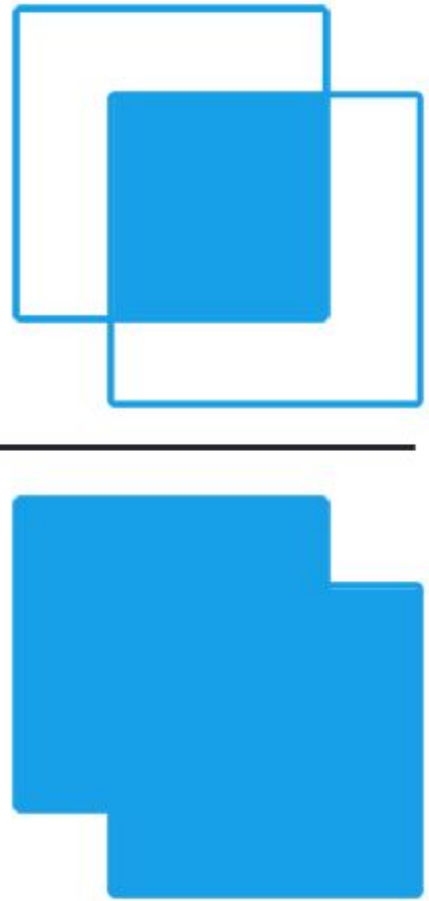


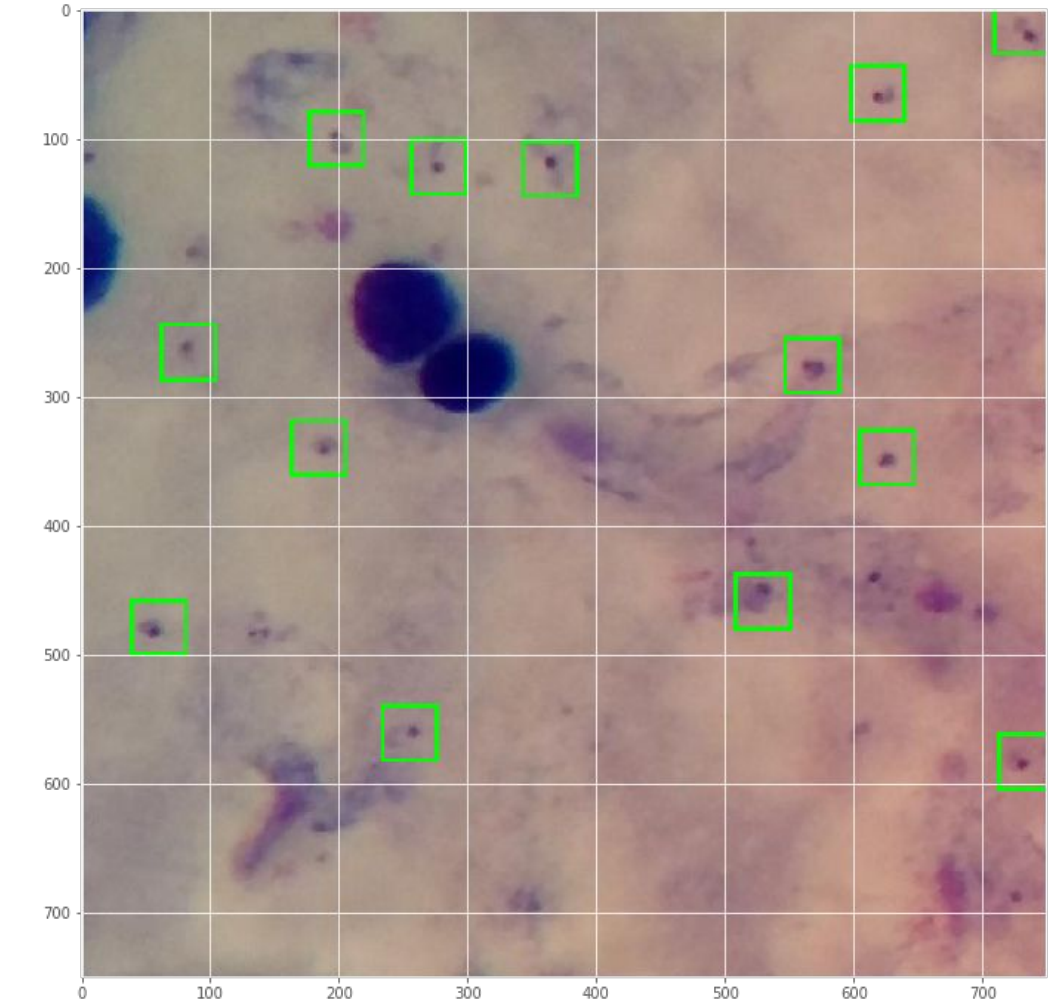
(c) 4×4 feature map

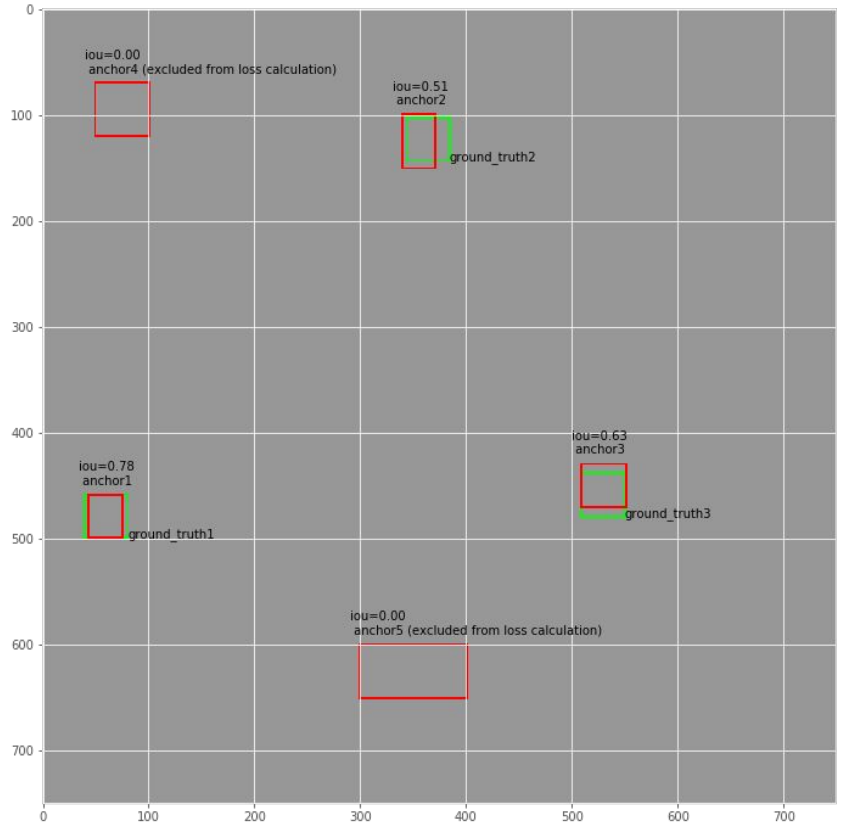
IoU is used to assess the quality of the anchor box (how good it matches the target bounding box)

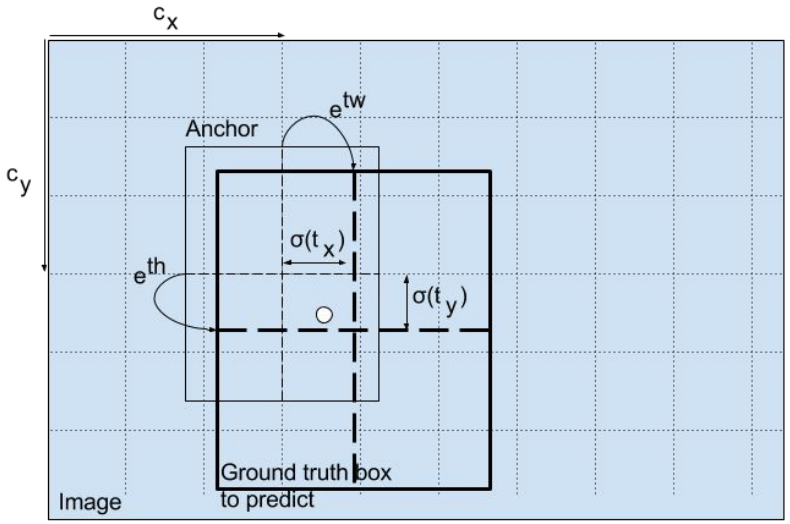
Usually IoU between 0.5-0.6 Is used as a “good match”

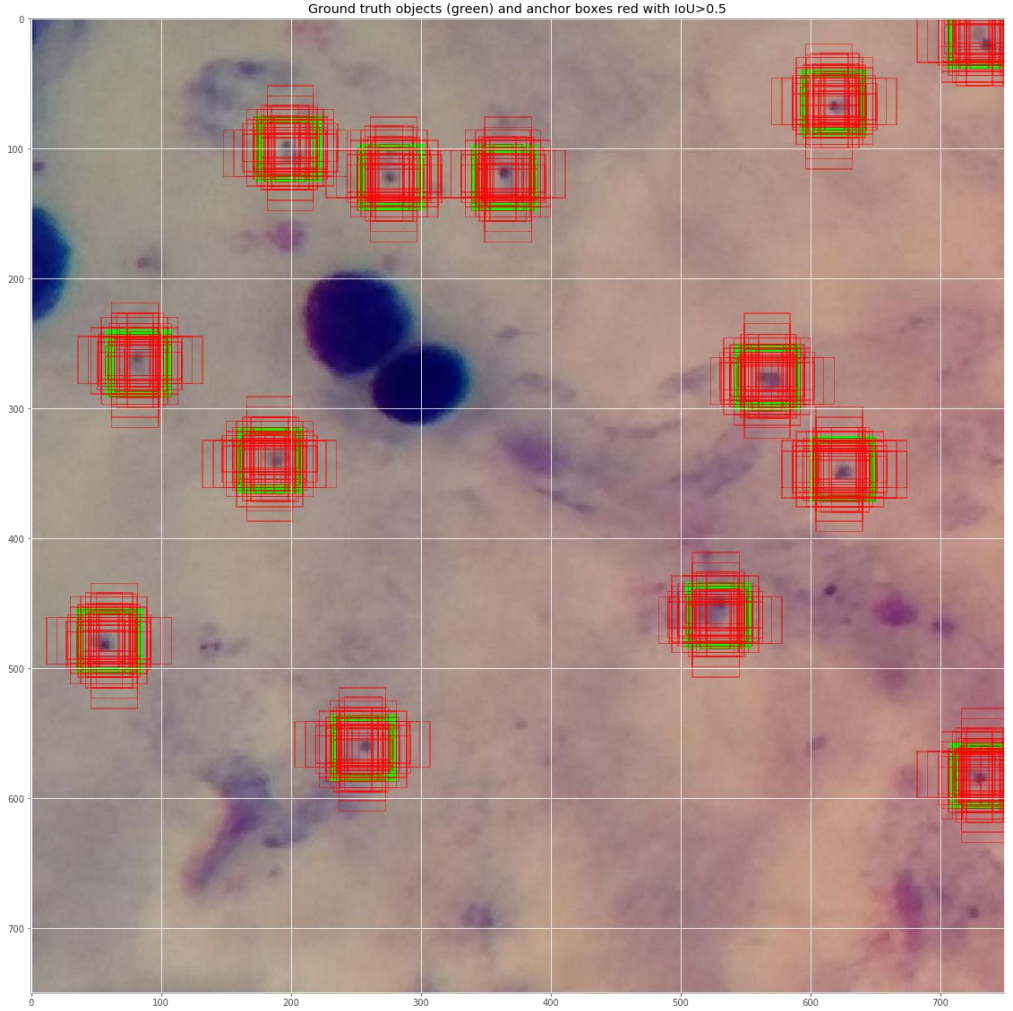
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



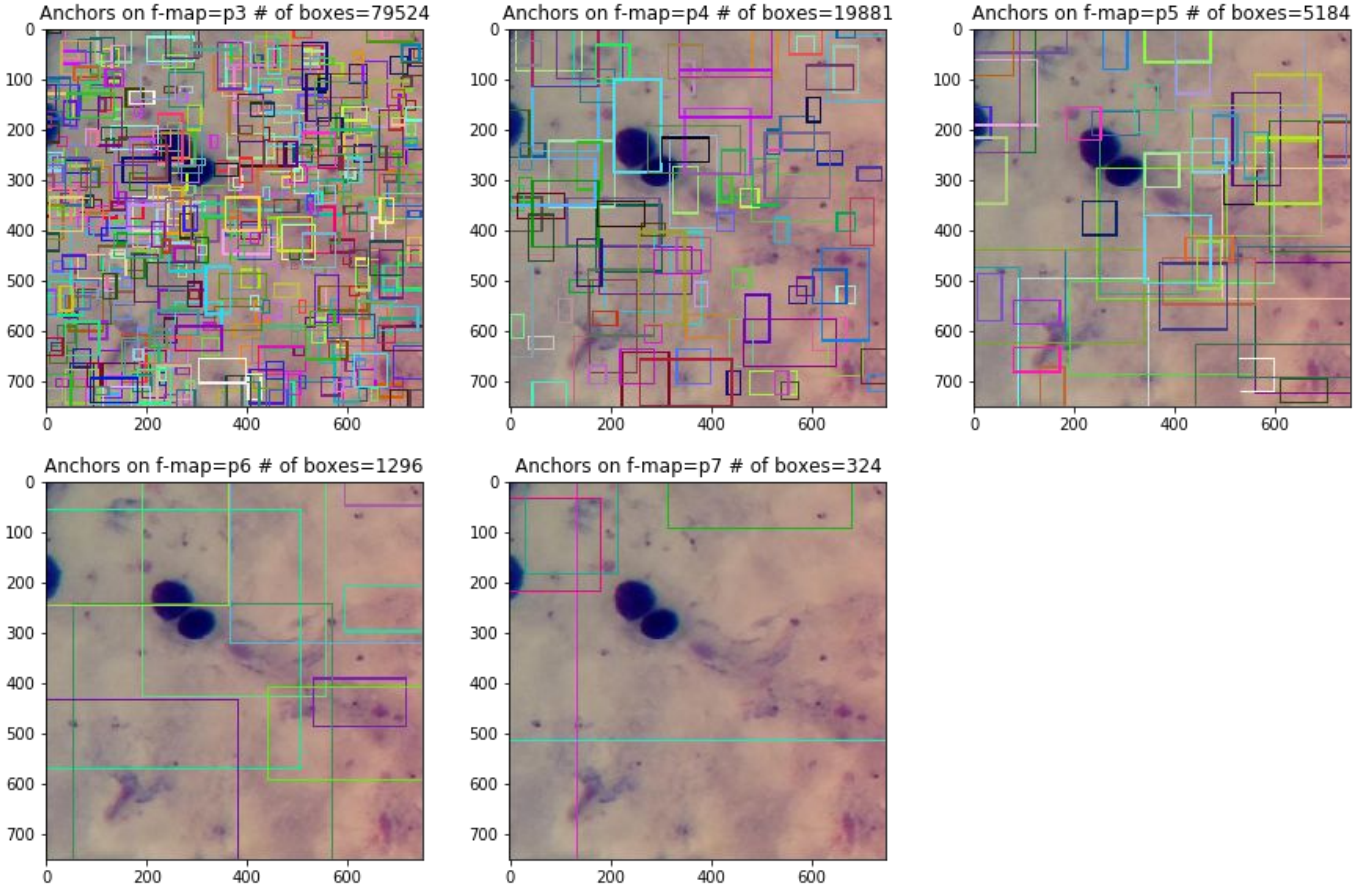



iou=0.00
anchor4 (excluded from loss calculation)
iou=0.51
anchor2
ground_truth2
iou=0.63
anchor3
ground_truth3
iou=0.78
anchor1
ground_truth1
iou=0.00
anchor5 (excluded from loss calculation)

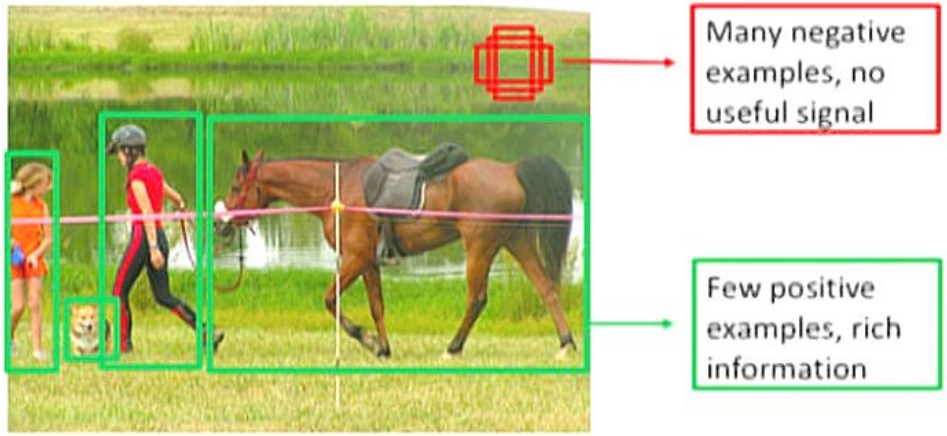

c_x
c_y
Anchor
e^tw
e^th
σ(t_x)
σ(t_y)
Ground truth box
to predict
Image

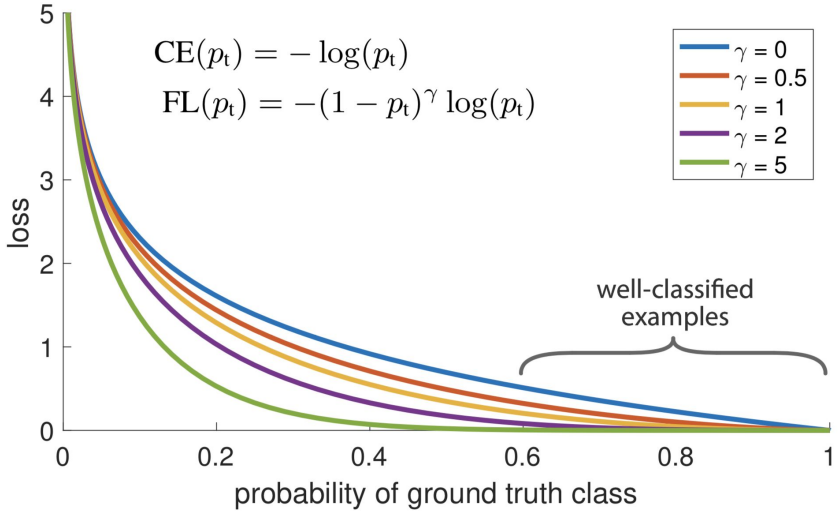

Ground truth objects (green) and anchor boxes red with IoU>0.5
0
100
200
300
400
500
600
700

Anchor boxes on different feature maps.

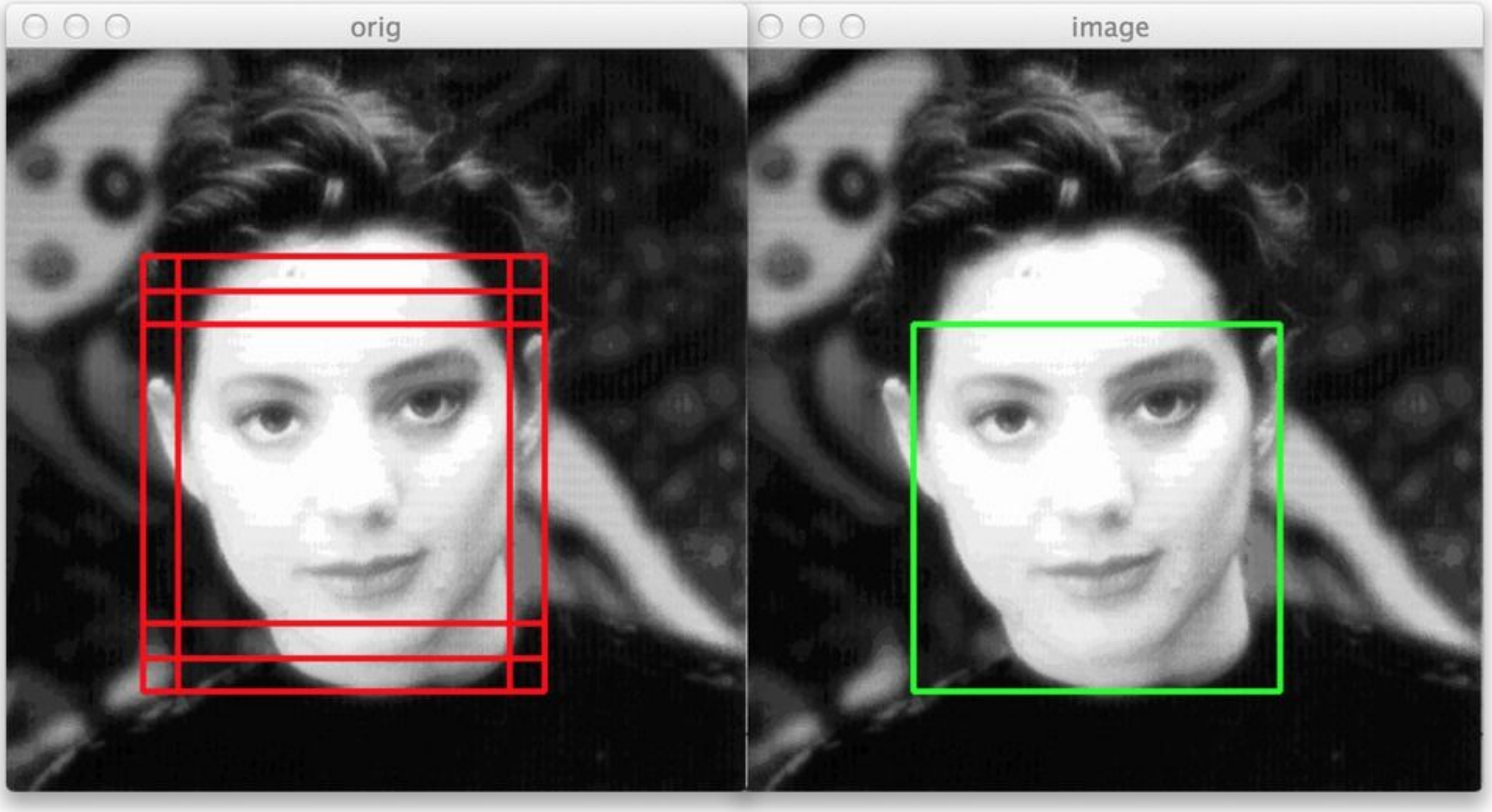


Focal Loss
Helps fighting imbalances (ratio of positive to negative bounding boxes)





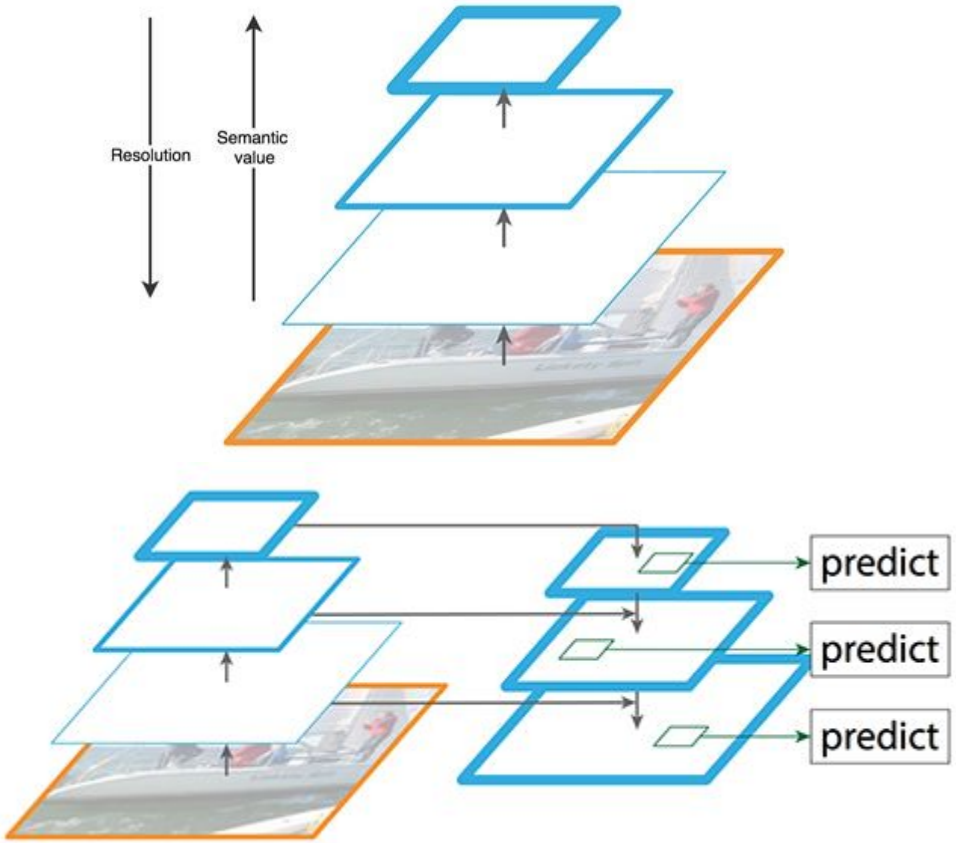
Non Maximum Suppression



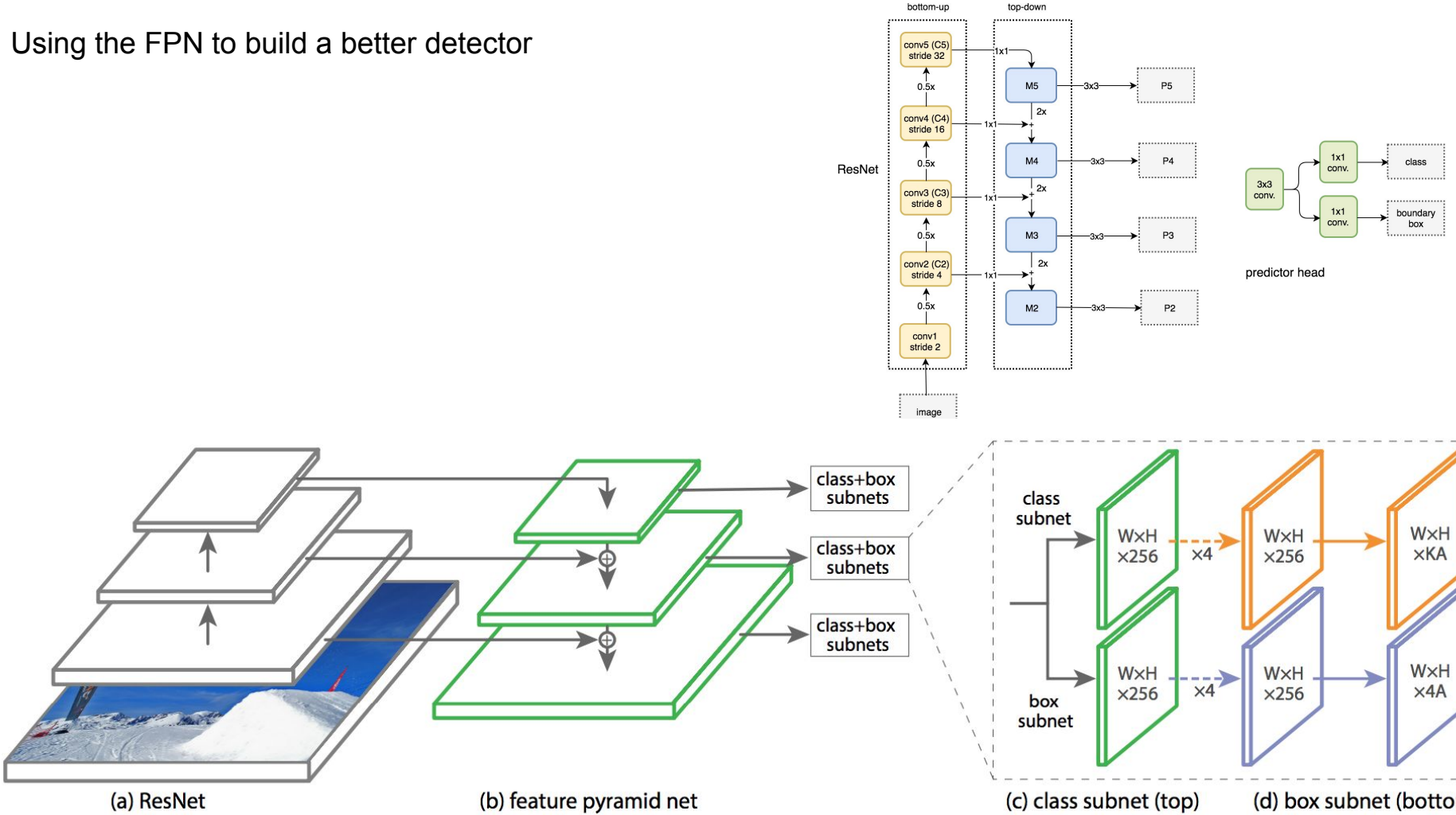
Feature Pyramid Networks - Solving the small object problem

- While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling.

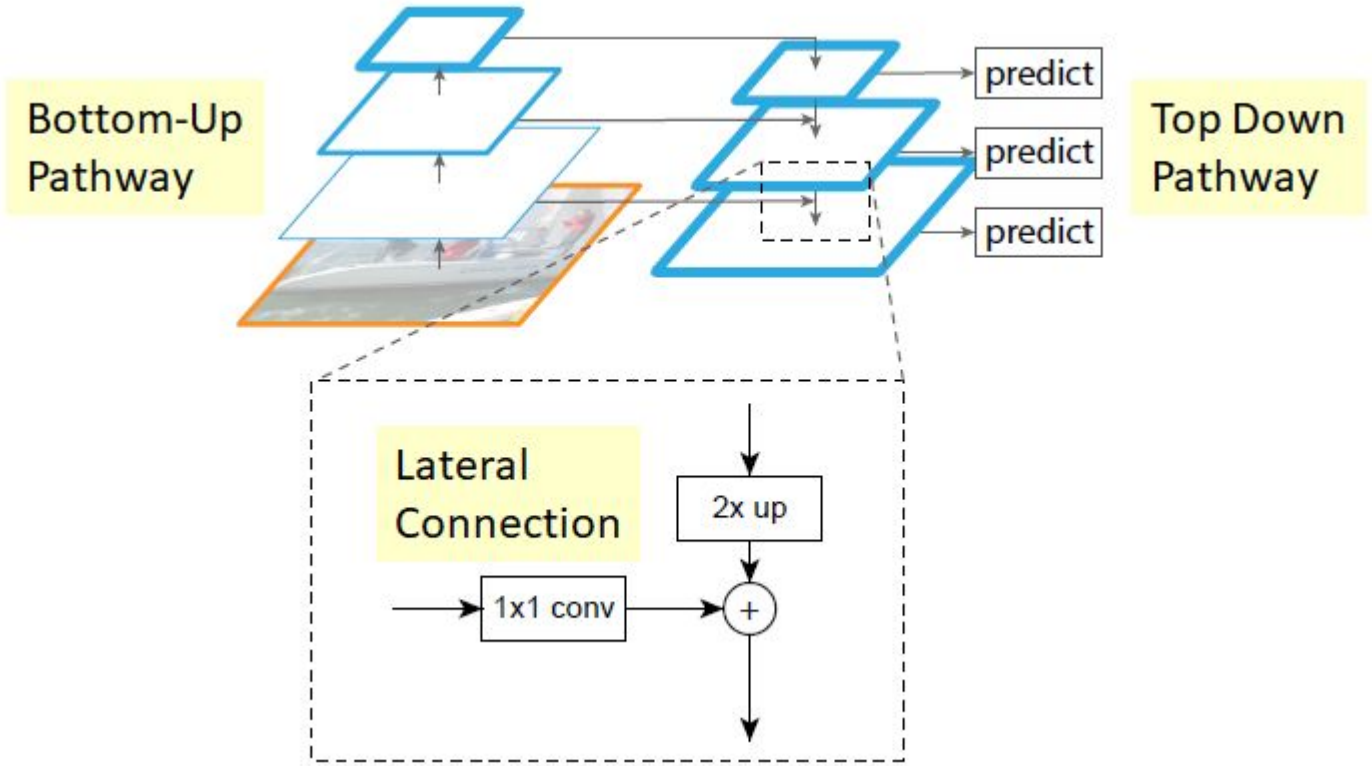
- We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the location betters.



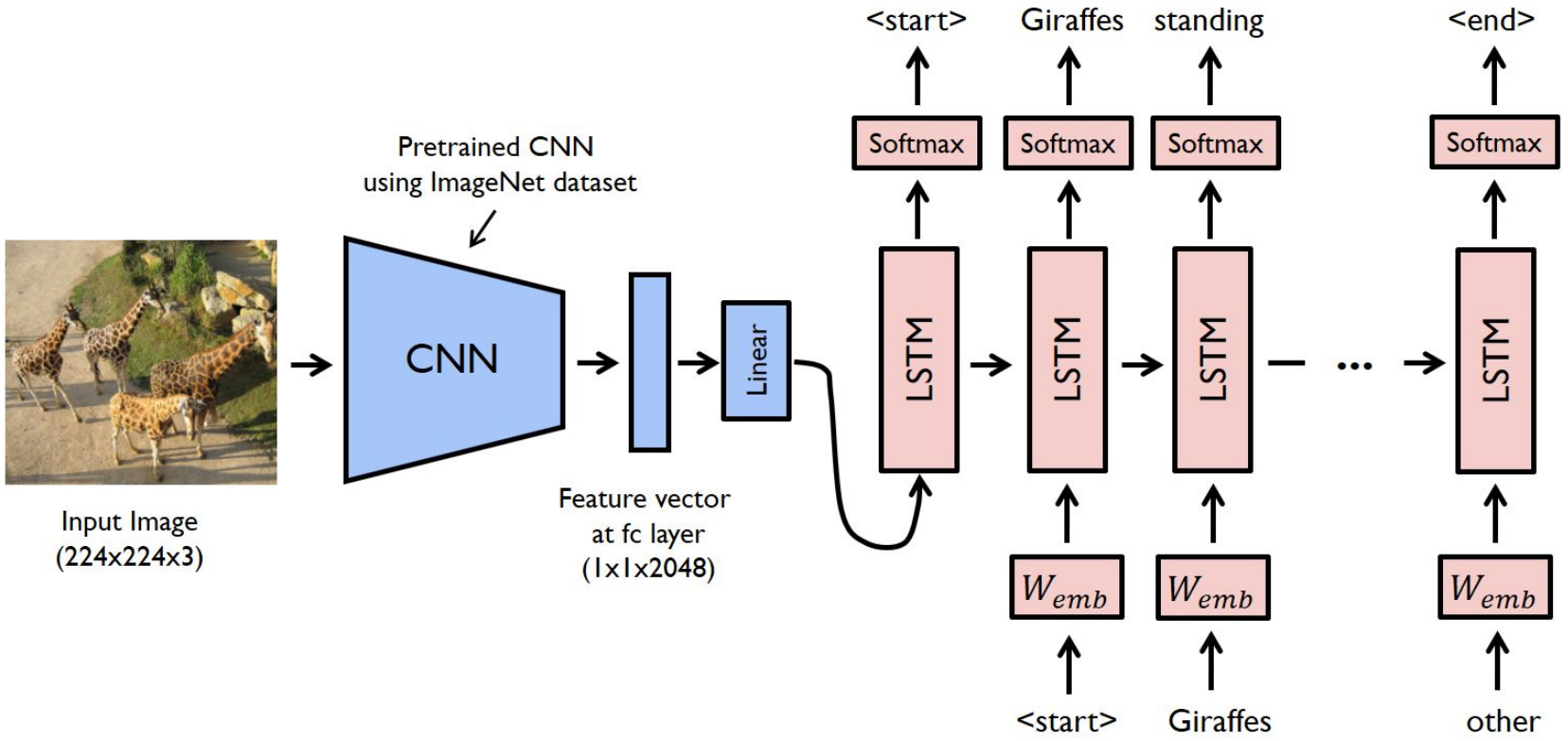
Using the FPN to build a better detector



- Feature extractors (VGG16, ResNet, Inception, MobileNet)
- Output strides for the extractor.
- Input image resolutions.
- Matching strategy and IoU threshold (how predictions are excluded in calculating loss).
- Non-max suppression IoU threshold.
- Hard example mining ratio (positive v.s. negative anchor ratio).
- The number of proposals or predictions.
- Boundary box encoding.
- Data augmentation.
- Training dataset.
- Use of multi-scale images in training or testing (with cropping).
- Which feature map layer(s) for object detection.
- Localization loss function.
- Deep learning software platform used.
- Training configurations including batch size, input image resize, learning rate, and learning rate decay.


Bottom-Up Pathway
predict
predict
predict
Top Down Pathway
Lateral Connection
2x up
1x1 conv
+

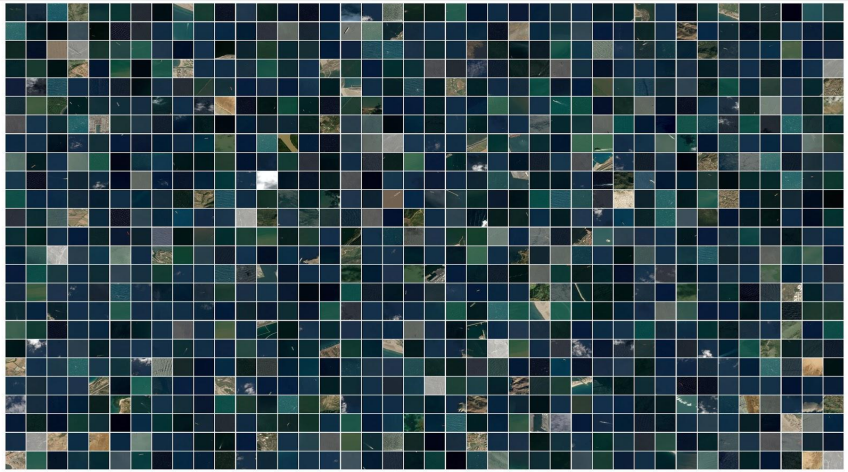
Encoder - > Decoder Network for image captioning



```
In [1]: from easyimages import EasyImageList, EasyImage
        from easyimages.utils import make_notebook_wider

In [3]: make_notebook_wider()

In [4]: EL = EasyImageList.from_folder('test_v2/')

In [5]: EL
Out[5]: <ImageList with 15606 EasyImages [filter OFF]>
```

TEST V2

```
In [13]: EL.html(size=50, sample=1000)
```



```
In [10]: EL_TRAIN  = EasyImageList.from_folder('./train')

In [12]: EL_TRAIN.html(size=50, sample=1000)
```