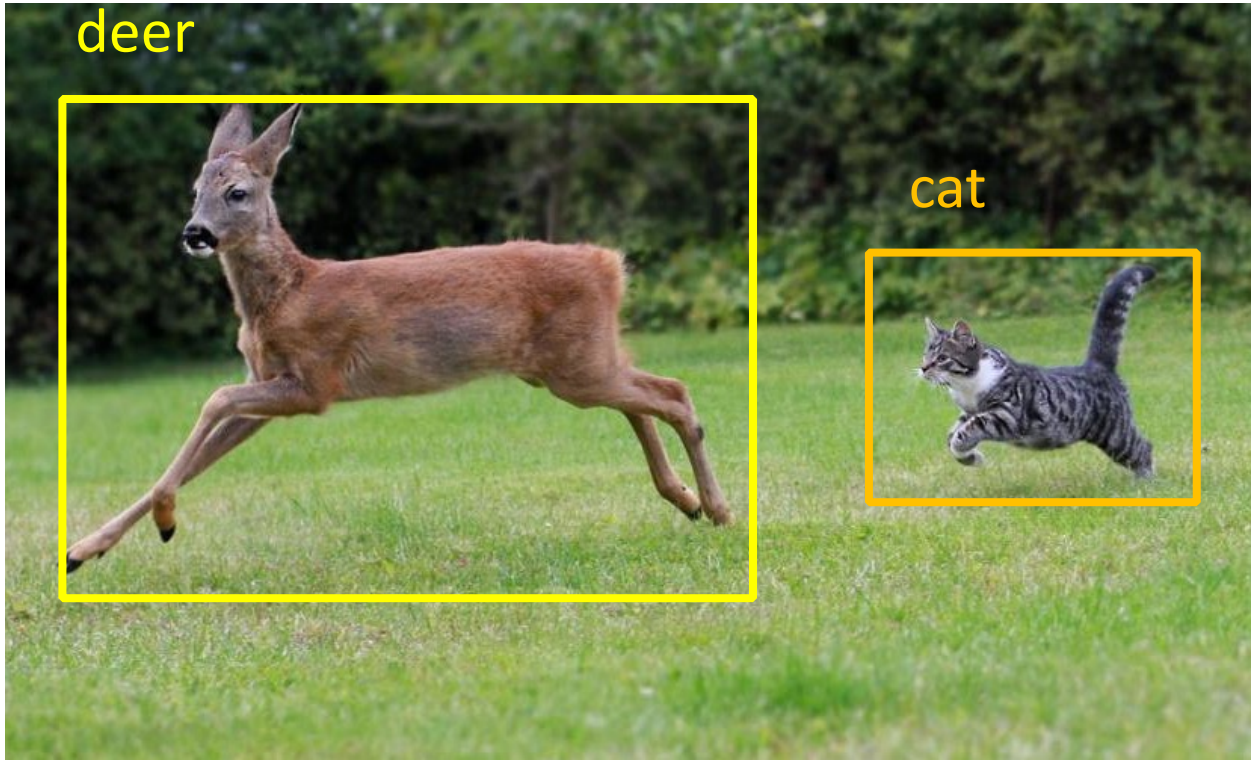
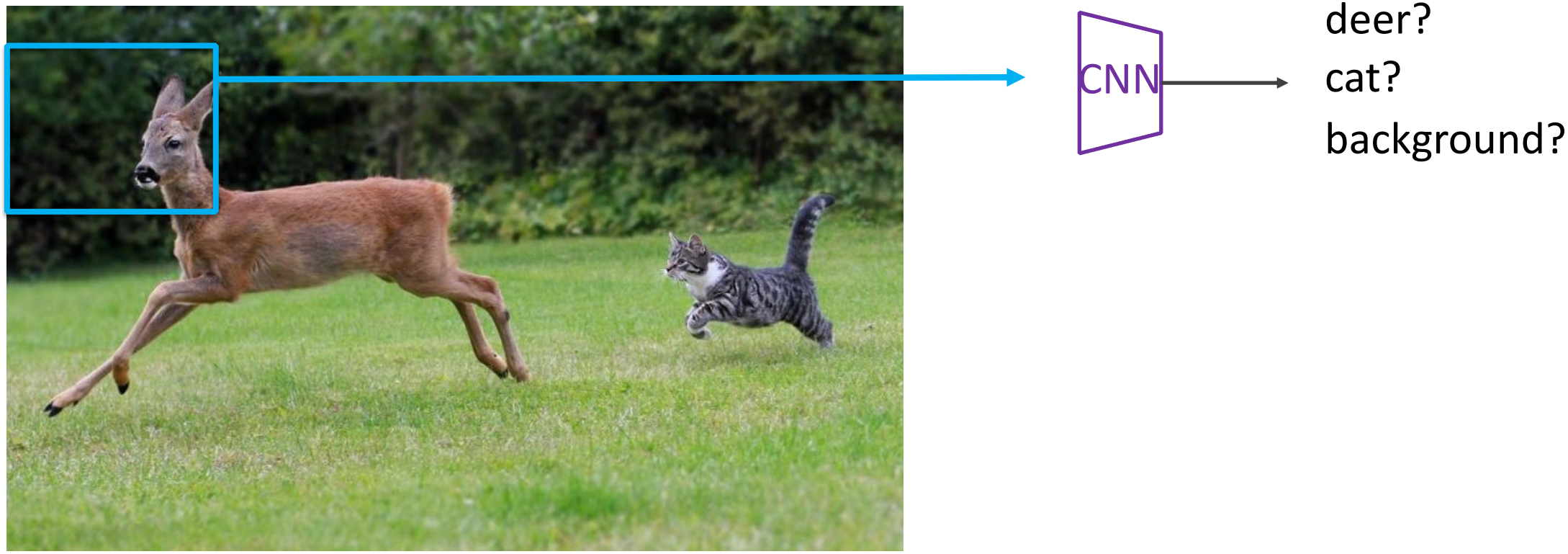
Deep Learning for Vision & Language

Convolutional Neural Networks for Detection and Segmentation

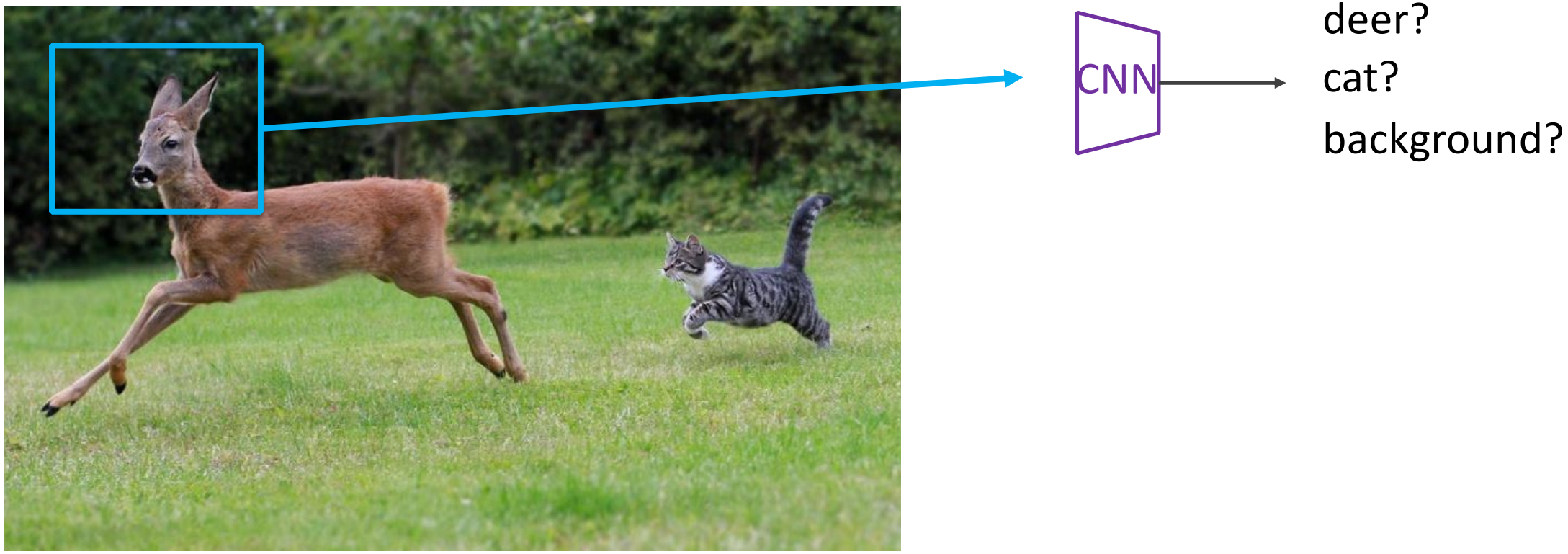
Object Detection



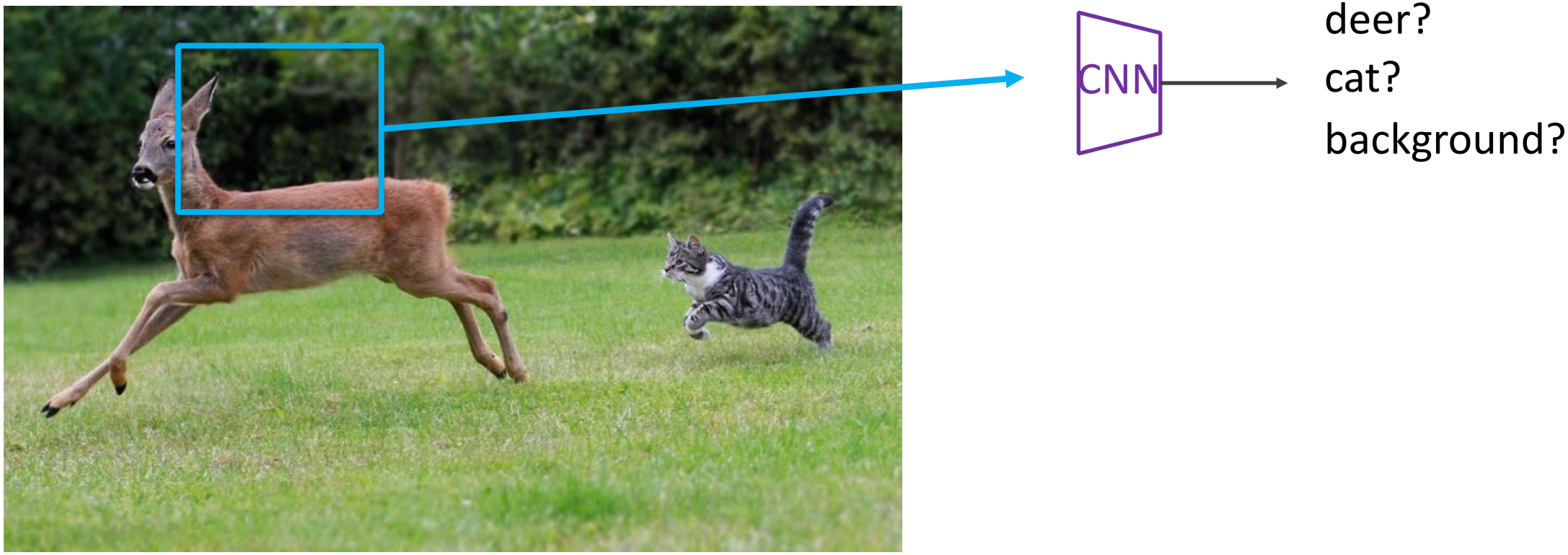
Object Detection as Classification



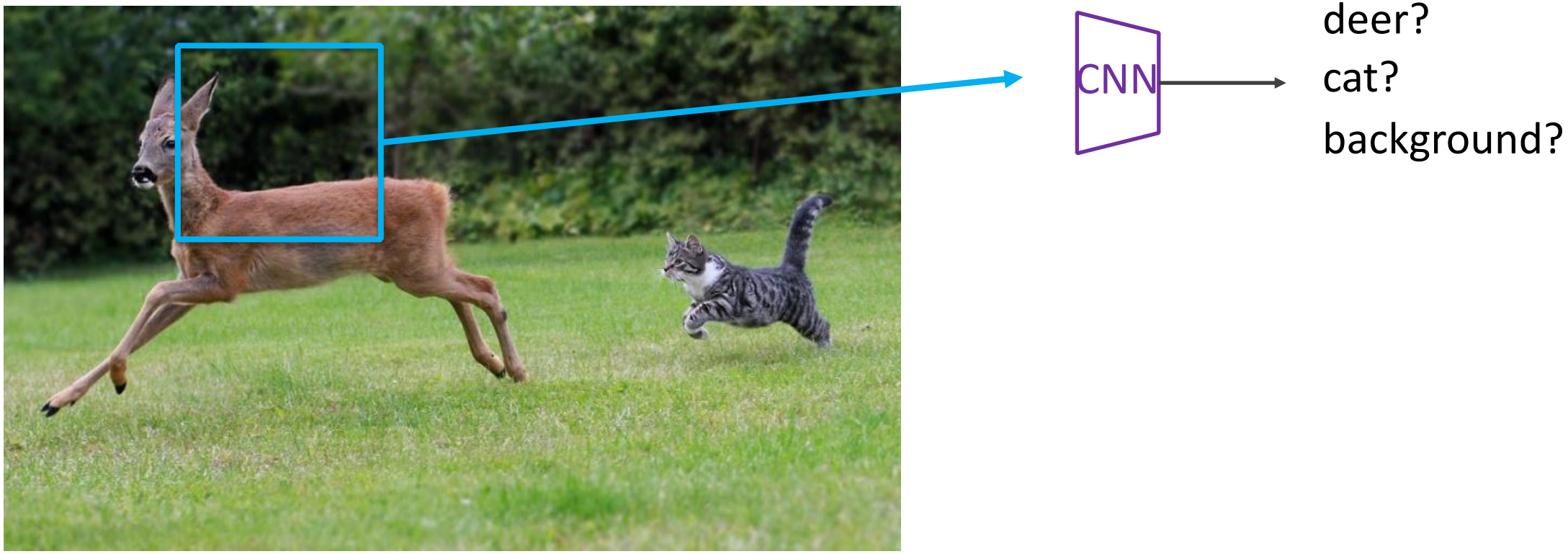
Object Detection as Classification



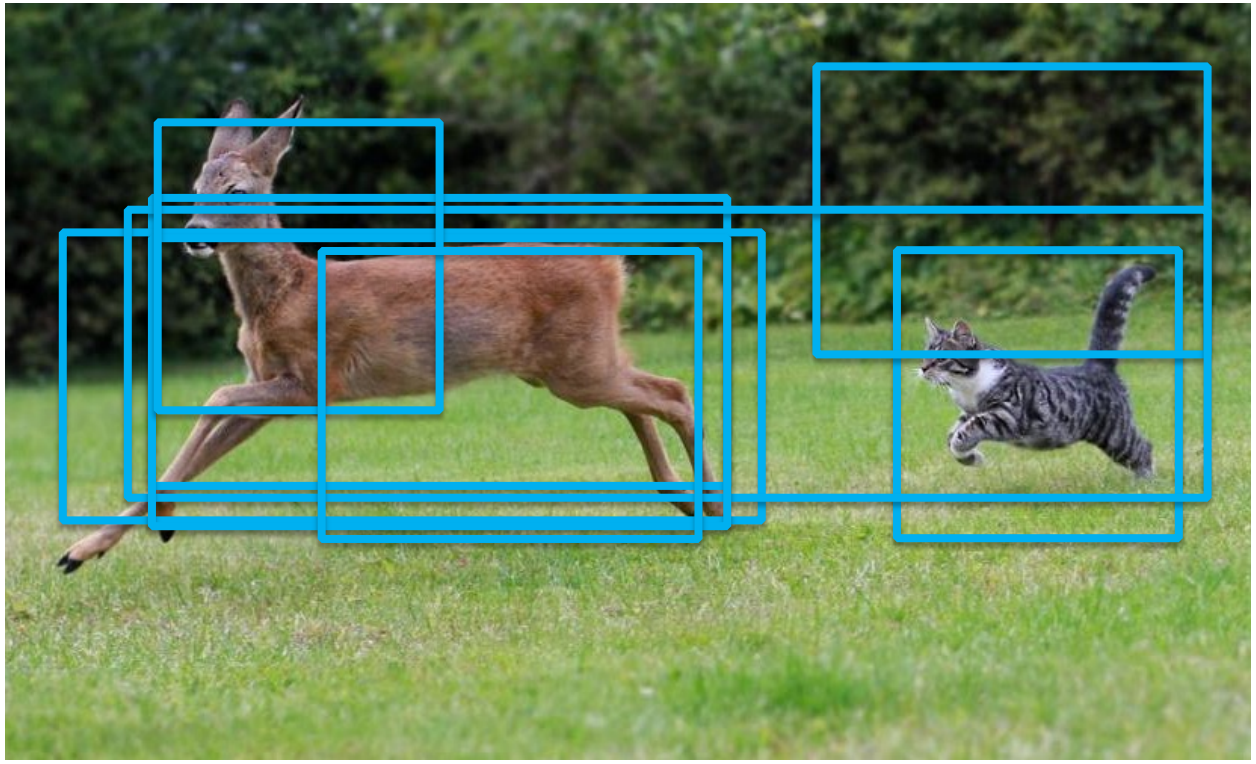
Object Detection as Classification



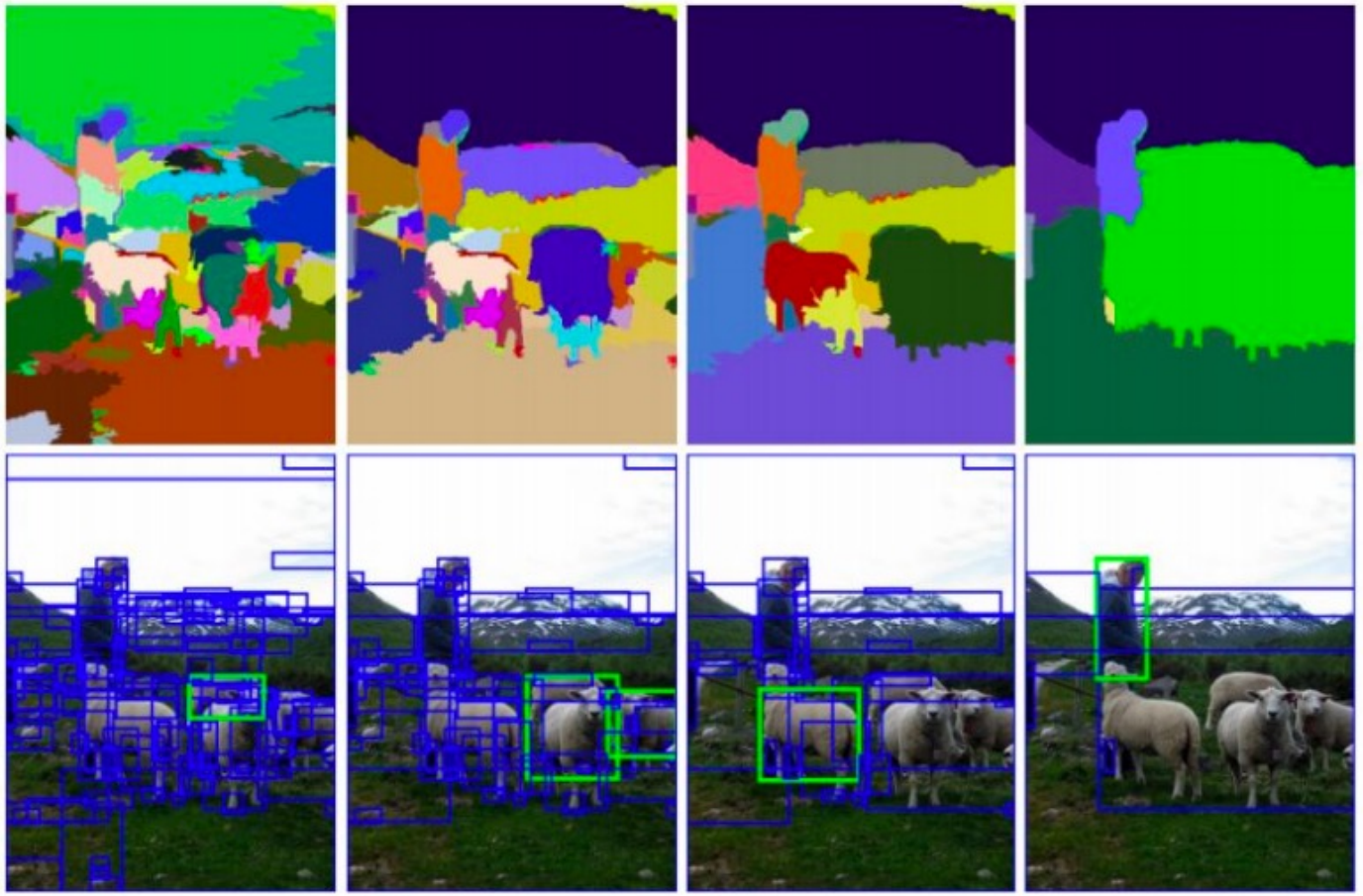
Object Detection as Classification with Sliding Window



Object Detection as Classification with Box Proposals

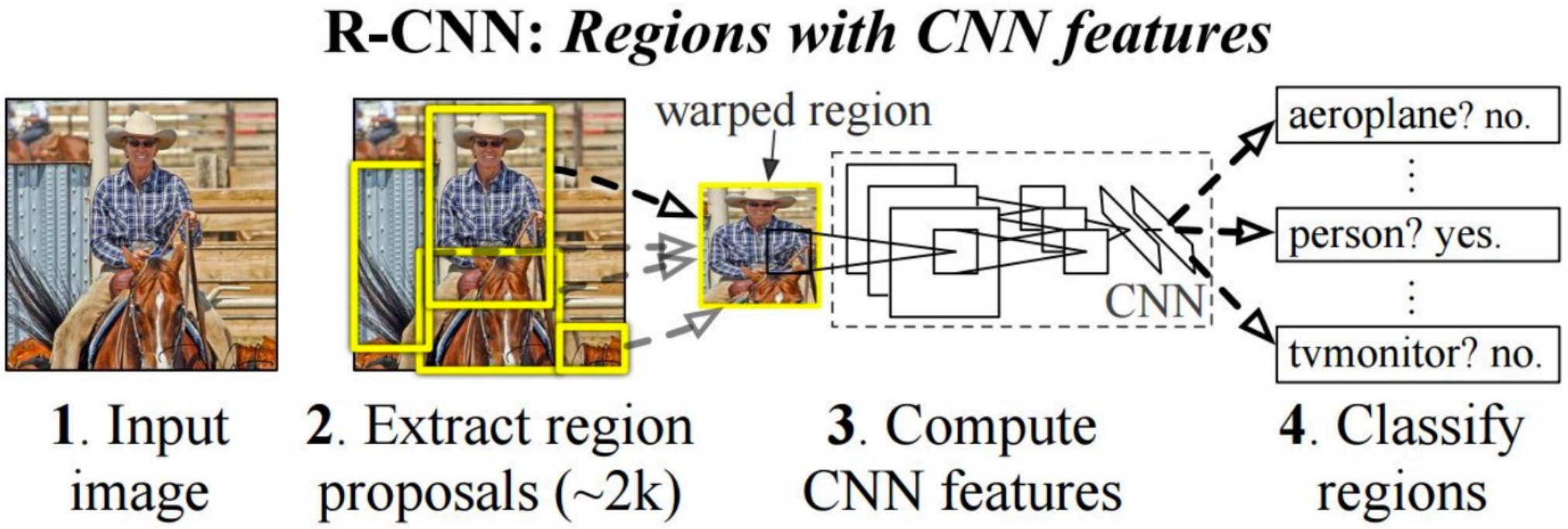


Box Proposal Method – SS: Selective Search



Segmentation As Selective Search for Object Recognition. van de Sande et al. ICCV 2011

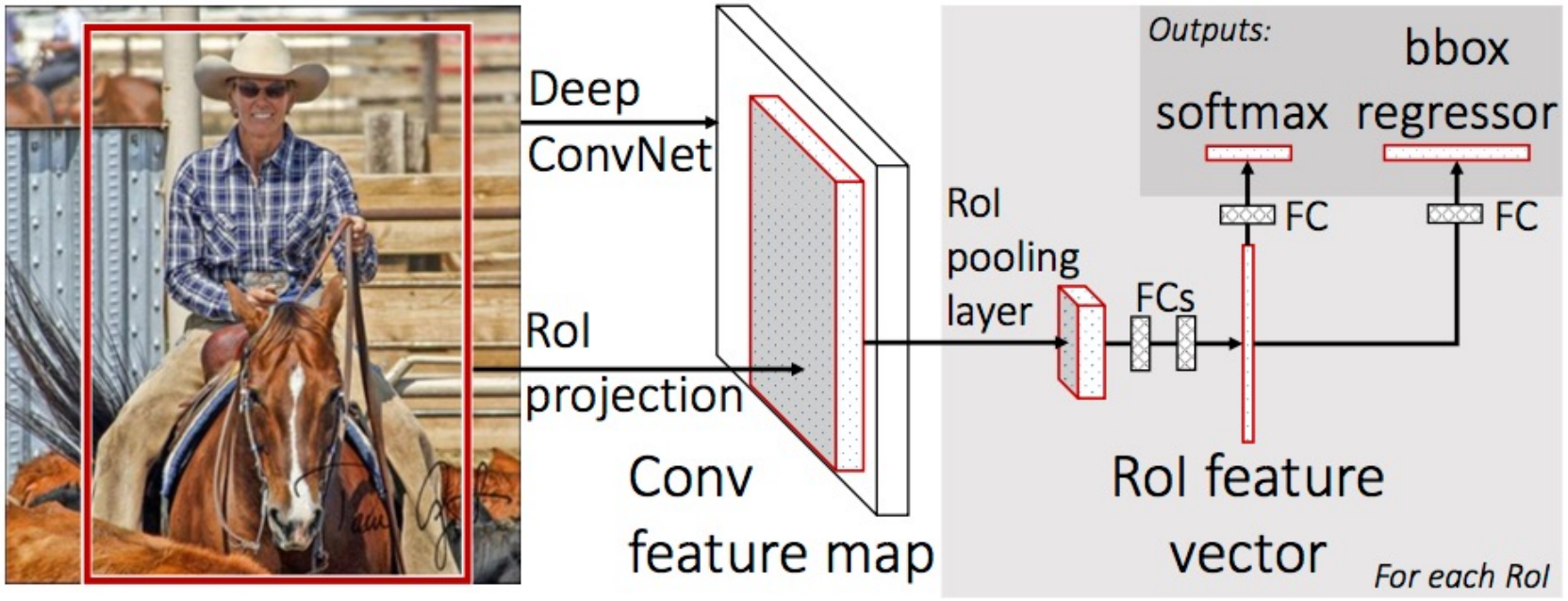
RCNN



https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf

Rich feature hierarchies for accurate object detection and semantic segmentation. Girshick et al. CVPR 2014.

Fast-RCNN



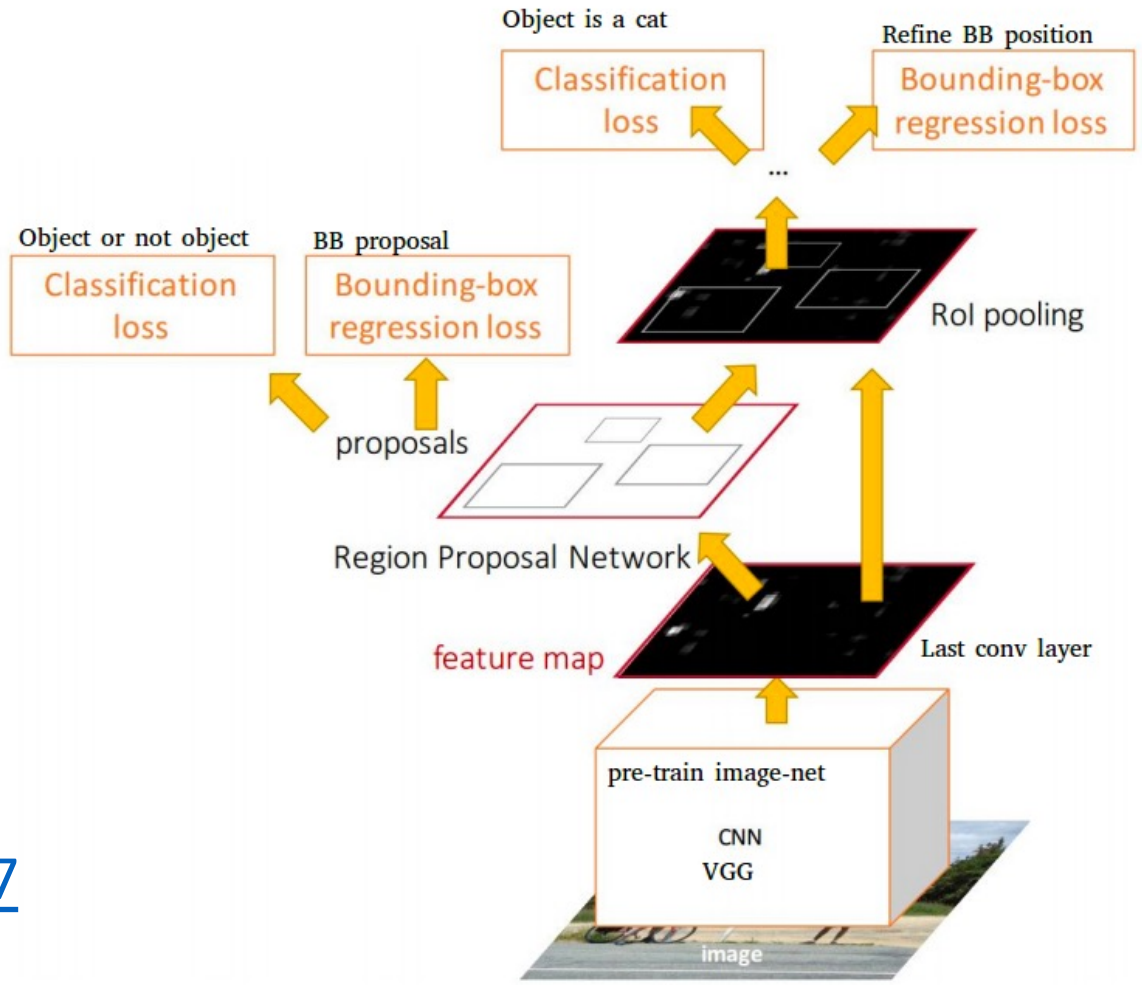
Idea: No need to recompute features for every box independently,
Regress refined bounding box coordinates.

https://arxiv.org/abs/1504.08083

Fast R-CNN. Girshick. ICCV 2015.

https://github.com/sunshineatnoon/Paper-Collection/blob/master/Fast-RCNN.md

Faster-RCNN



Idea: Integrate the Bounding Box Proposals as part of the CNN predictions

https://arxiv.org/abs/1506.01497

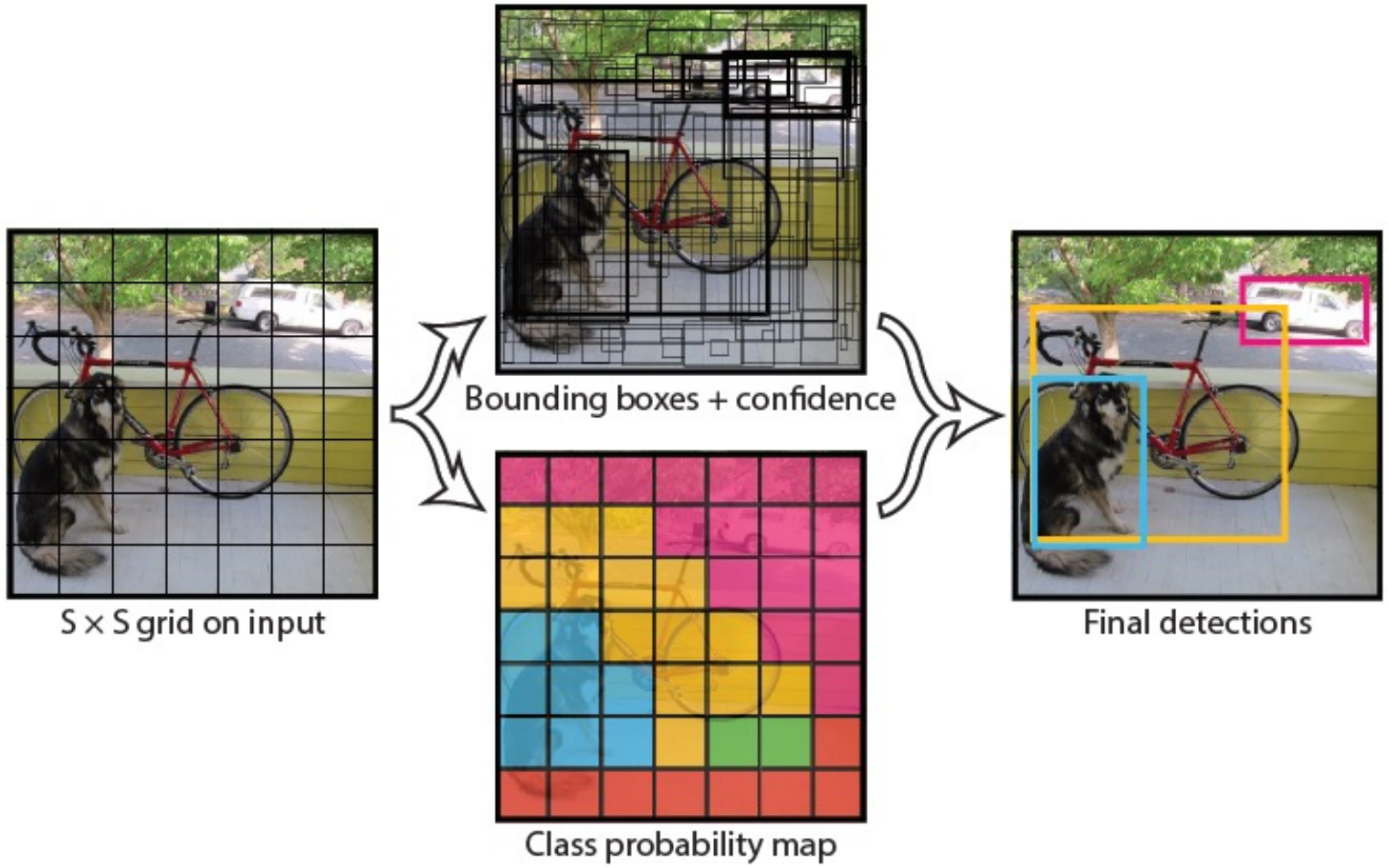
Ren et al. NIPS 2015.

Single-shot Object Detectors

- No two-steps of box proposals + Classification
- Anchor Points for predicting boxes

YOLO- You Only Look Once

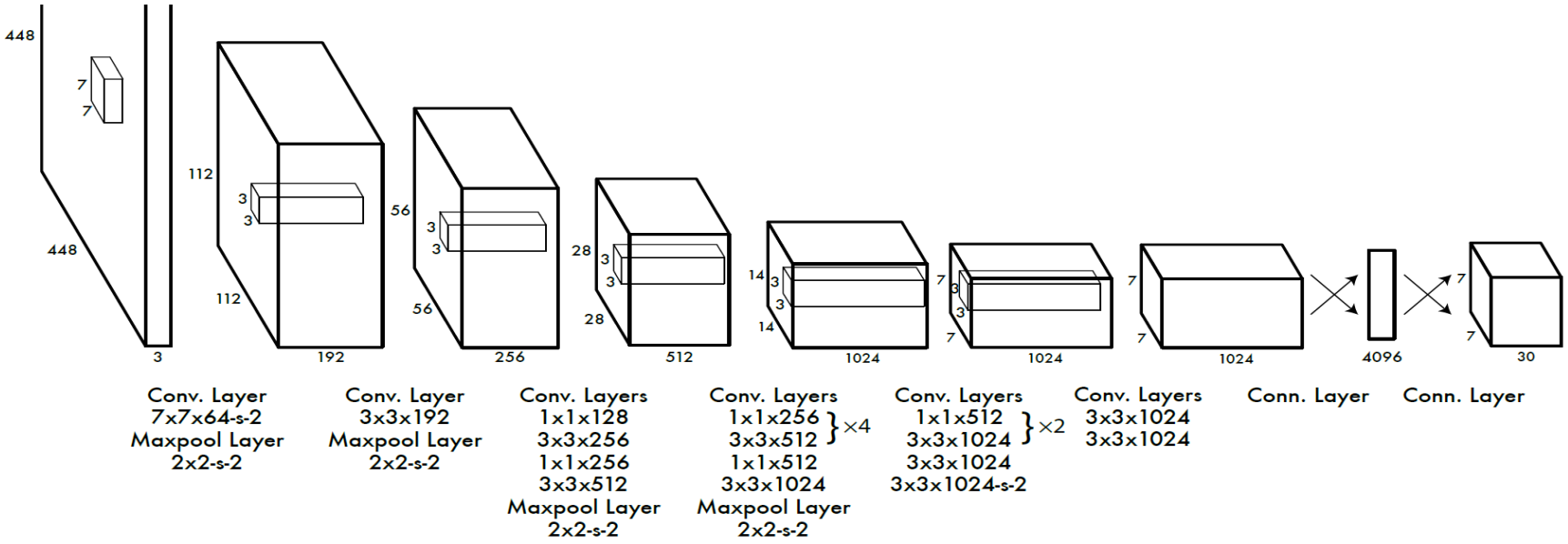
Idea: No bounding box proposals. Predict a class and a box for every location in a grid.



https://arxiv.org/abs/1506.02640

Redmon et al. CVPR 2016.

YOLO- You Only Look Once



Divide the image into 7x7 cells.
Each cell trains a detector.
The detector needs to predict the object's class distributions.
The detector has 2 bounding-box predictors to predict bounding-boxes and confidence scores.

https://arxiv.org/abs/1506.02640

Redmon et al. CVPR 2016.

YOLO - Loss Function

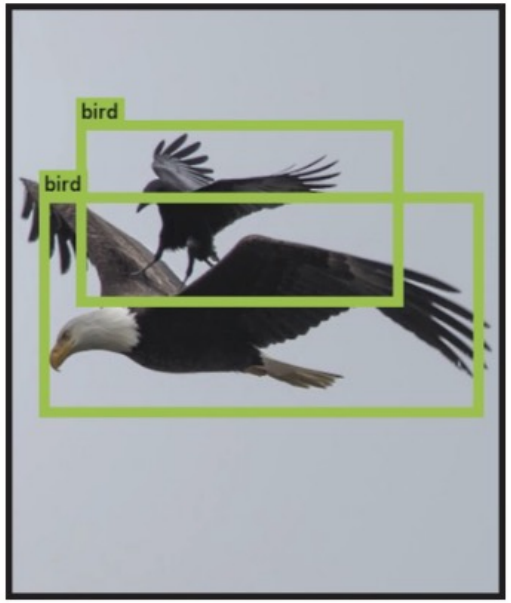
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

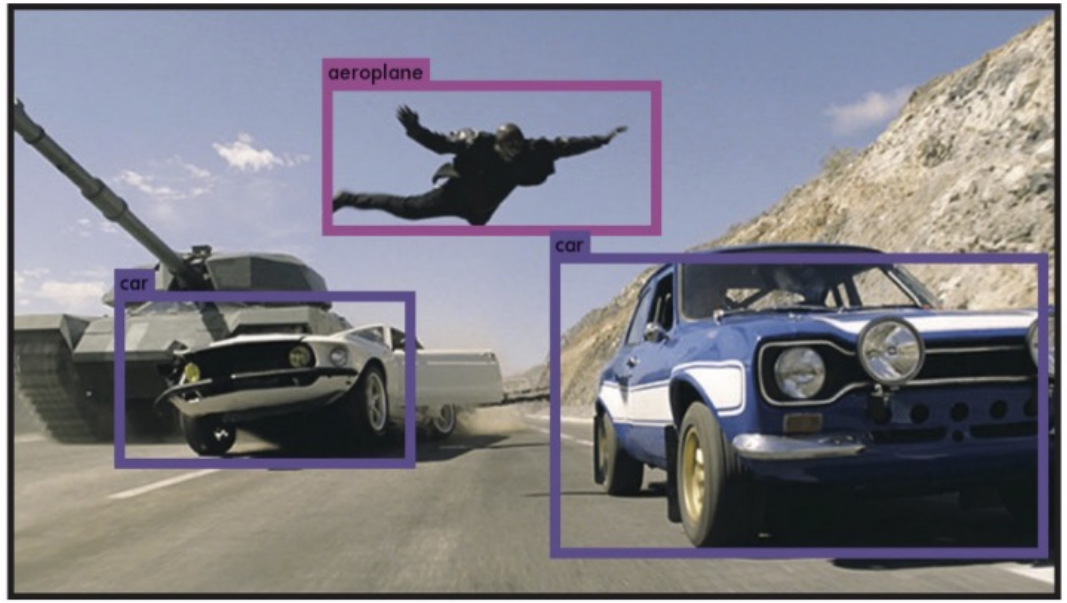
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

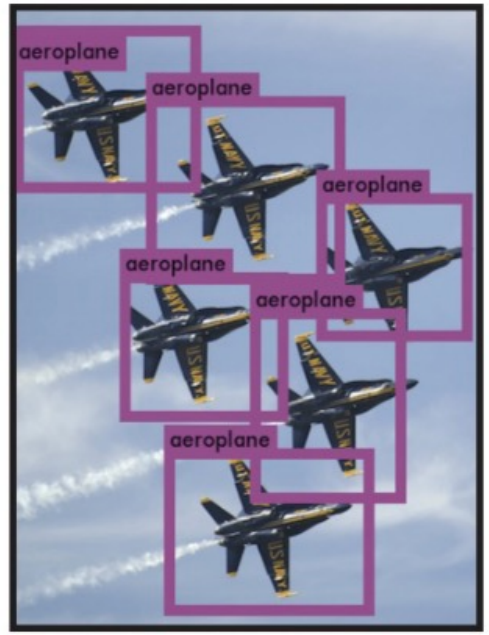
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

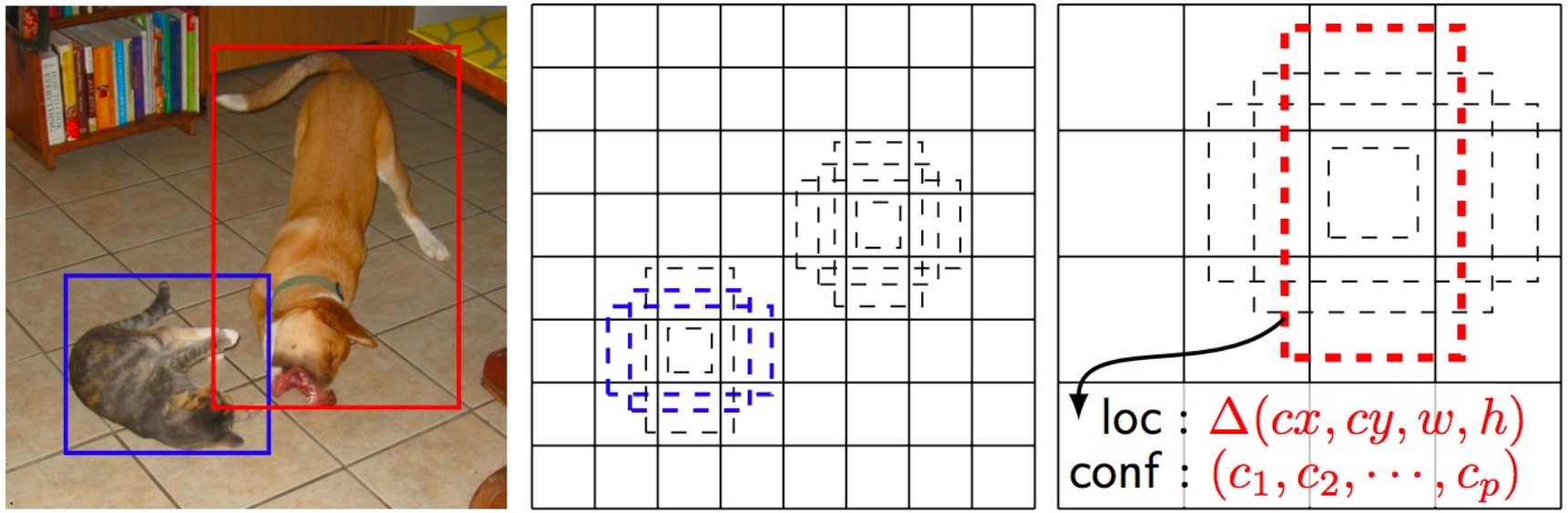

bird
bird


aeroplane
car
car


person
boat


aeroplane
aeroplane
aeroplane
aeroplane
aeroplane
aeroplane

SSD: Single Shot Detector

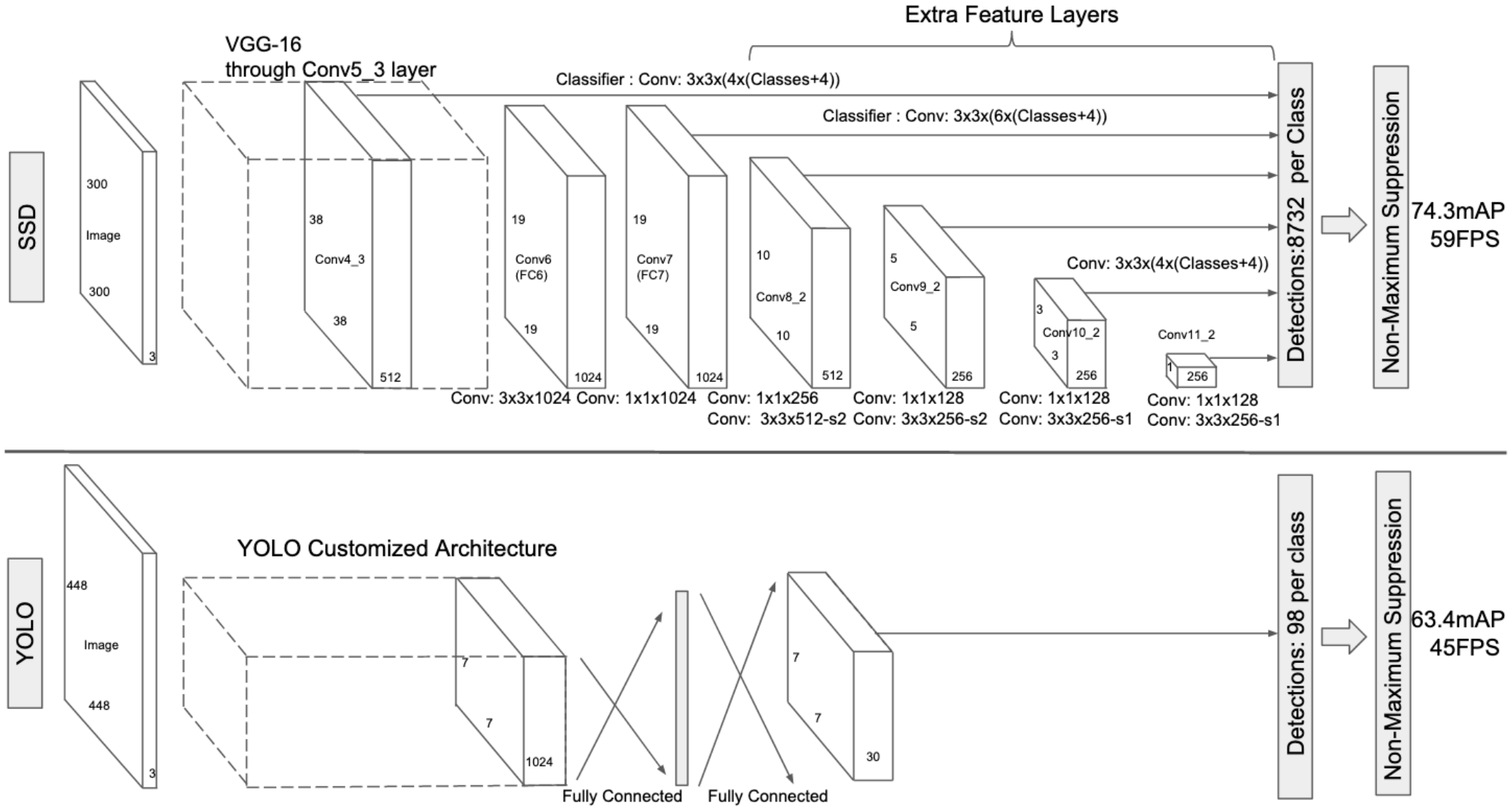


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

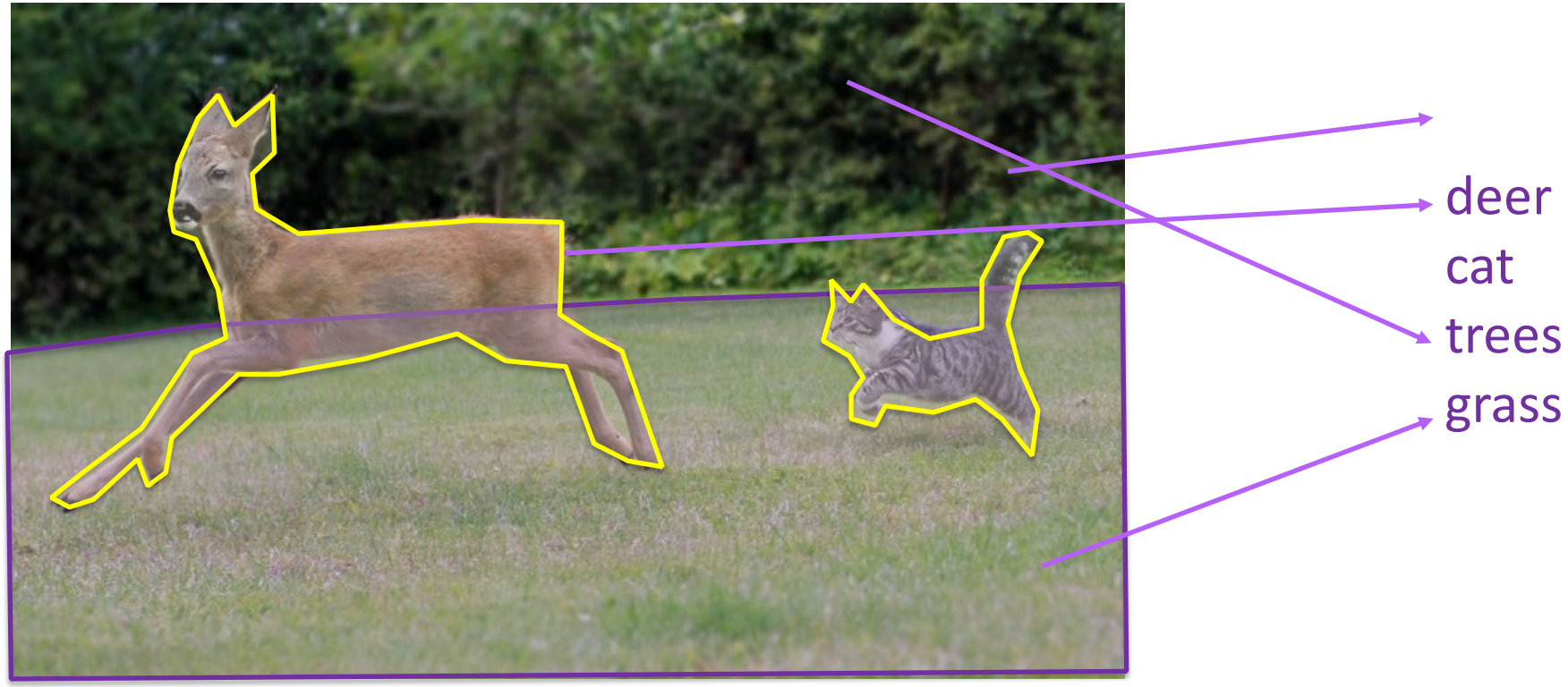
Idea: Similar to YOLO, but denser grid map, multiscale grid maps. + Data augmentation + Hard negative mining + Other design choices i n the network.

Liu et al. ECCV 2016.

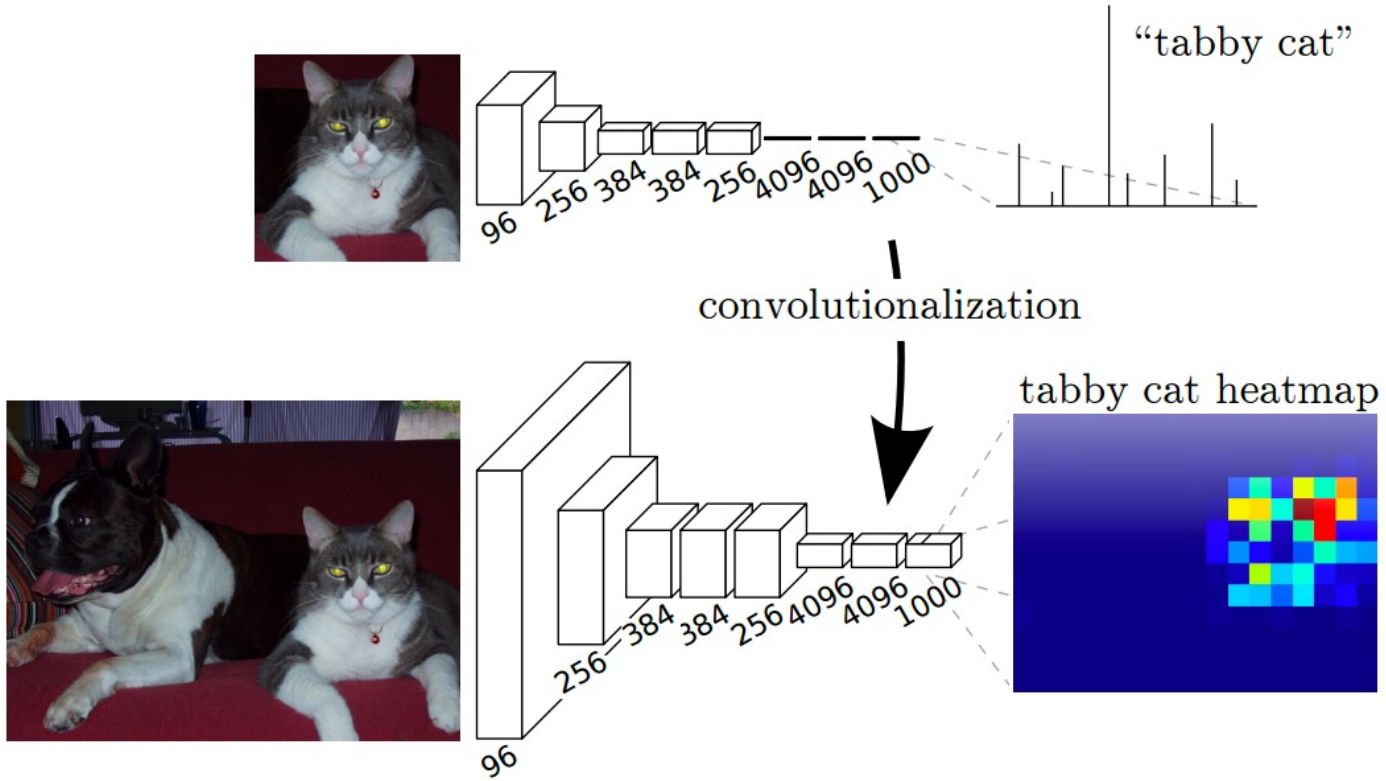
SSD vs YOLO


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
Conv11_2
1
256
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
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

Semantic Segmentation / Image Parsing



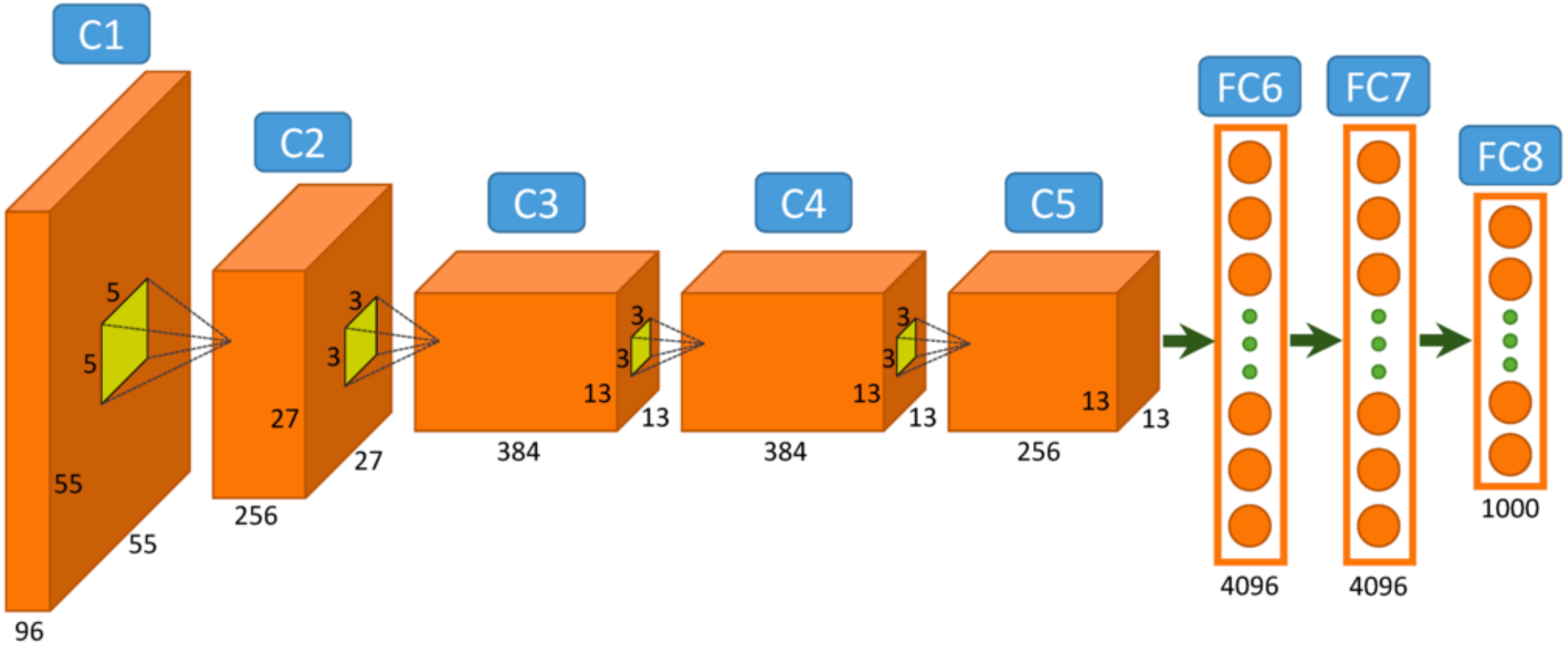
Idea 1: Convolutionalization



However resolution of the segmentation map is low.

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

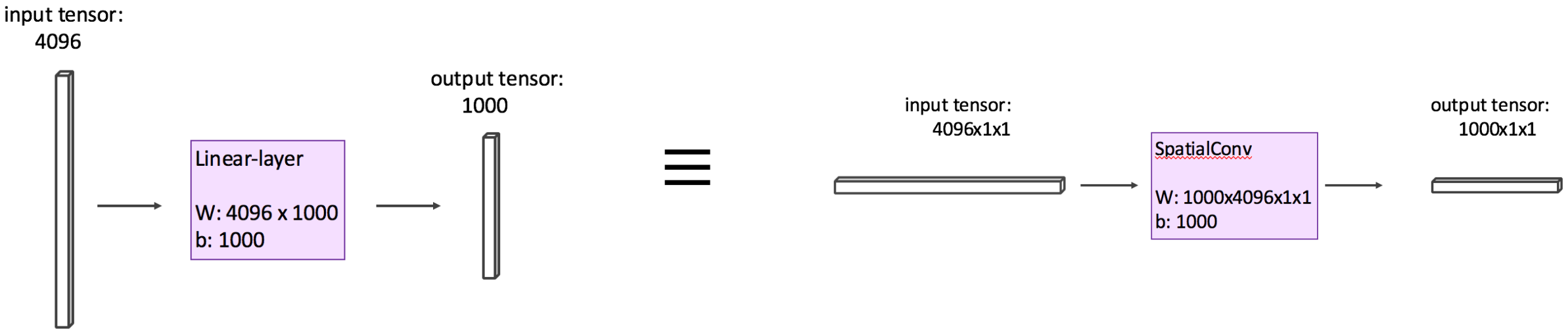
Alexnet



https://www.saagie.com/fr/blog/object-detection-part1

Idea 1: Convolutionalization

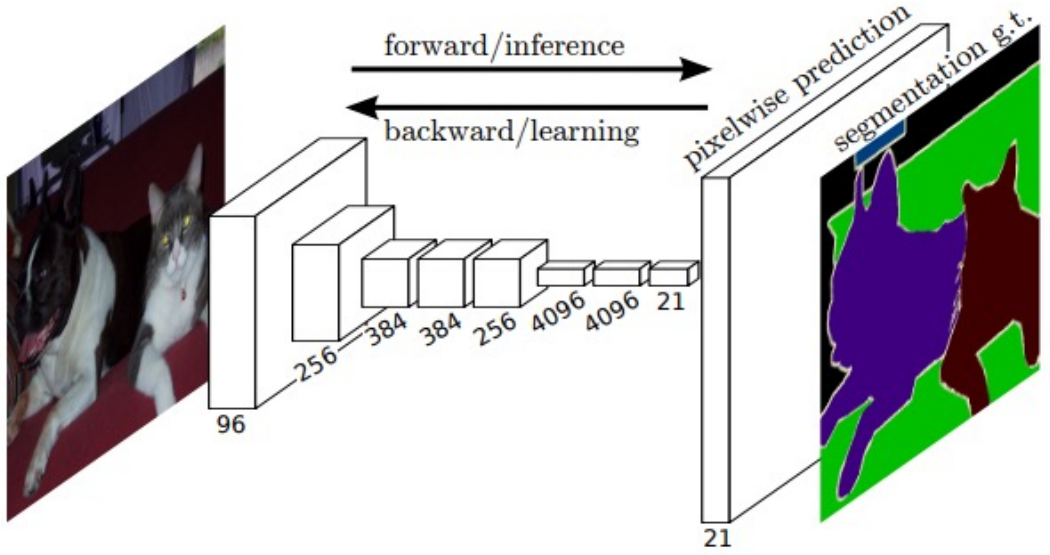
```
nn.Linear(n_inputs, n_outputs) == nn.SpatialConvolution(n_inputs, n_outputs, 1, 1, 1, 1)
```



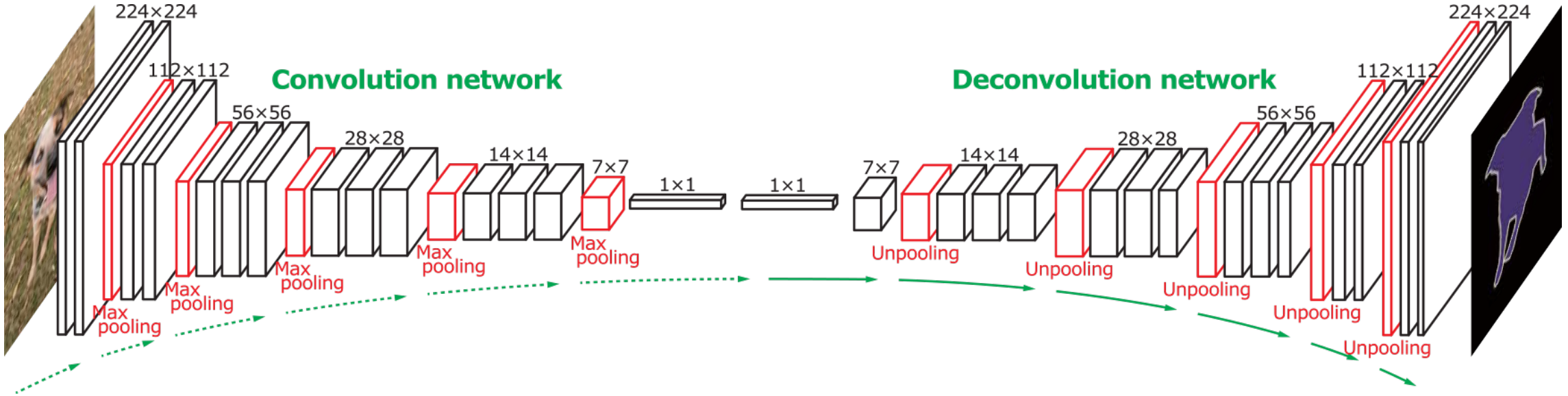
Fully Convolutional Networks (CVPR 2015)

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu



Idea 2: Up-sampling Convolutions or ”Deconvolutions”

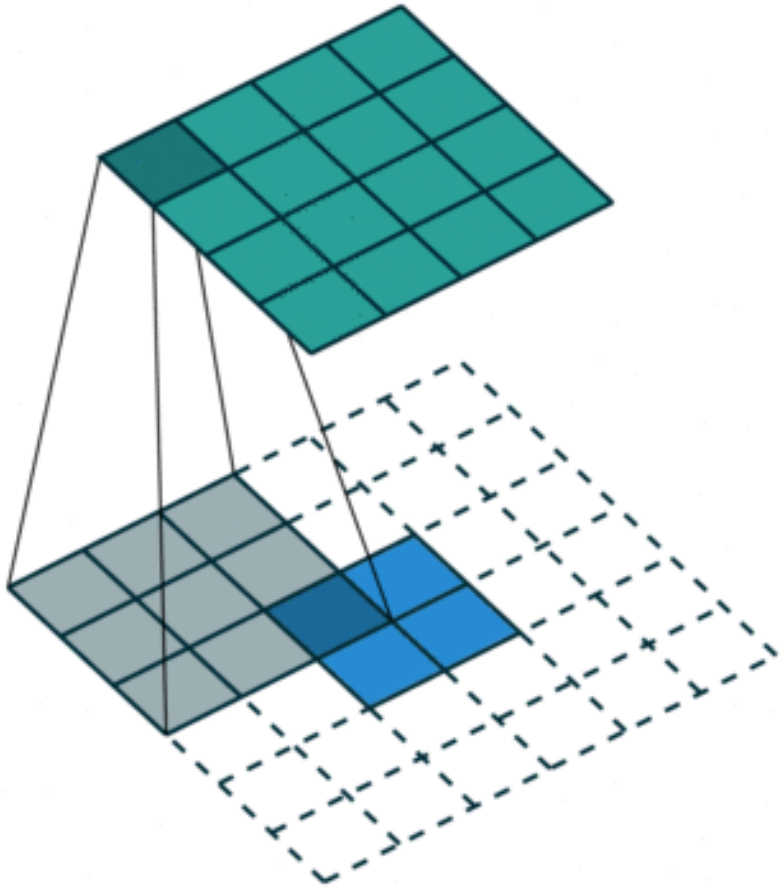


Learning Deconvolution Network for Semantic Segmentation

Hyeonwoo Noh Seunghoon Hong Bohyung Han
Department of Computer Science and Engineering, POSTECH, Korea
{hyeonwoonoh_,maga33,bhhan}@postech.ac.kr

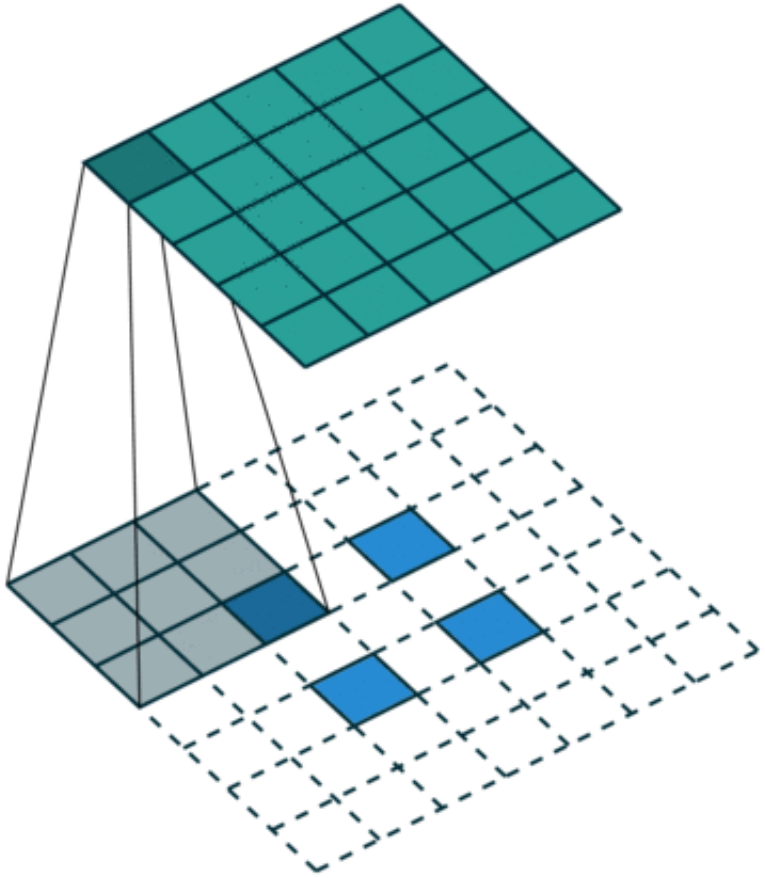
http://cvlab.postech.ac.kr/research/deconvnet/

Idea 2: Up-sampling Convolutions or ”Deconvolutions”



https://github.com/vdumoulin/conv_arithmetic

Idea 2: Up-sampling Convolutions or ”Deconvolutions”



https://github.com/vdumoulin/conv_arithmetic

Idea 2: Up-sampling Convolutions or ”Deconvolutions”

Deconvolutional Layers

Upconvolutional Layers

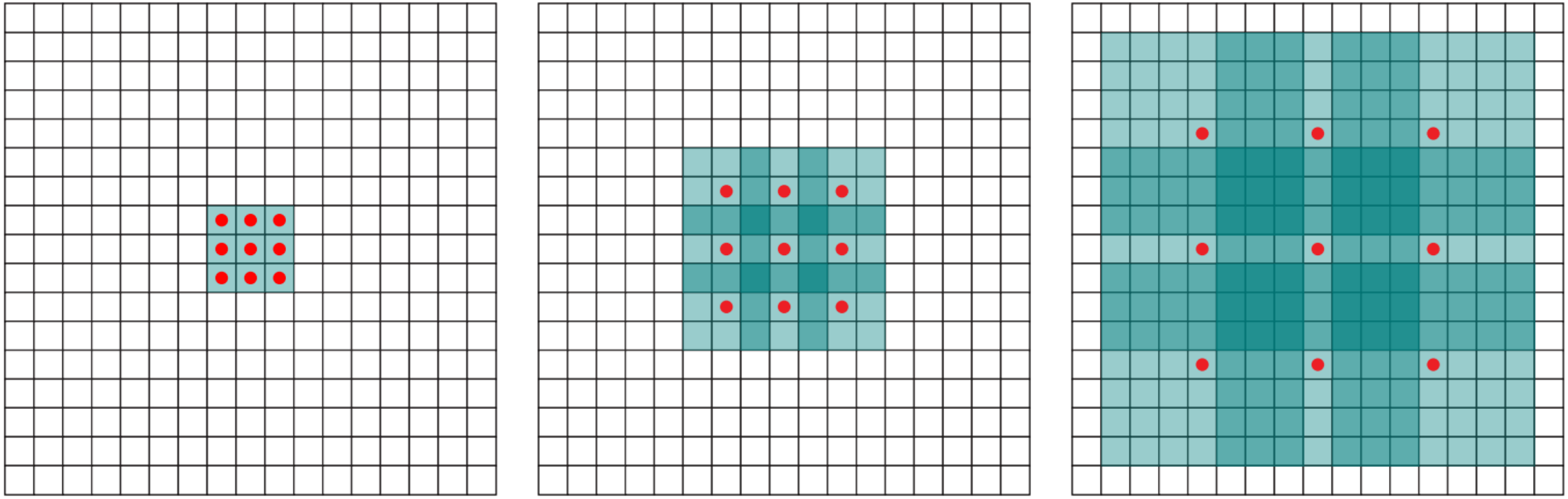
Backwards Strided
Convolutional Layers

Fractionally Strided
Convolutional Layers

Transposed
Convolutional Layers

Spatial Full
Convolutional Layers

Idea 3: Dilated Convolutions



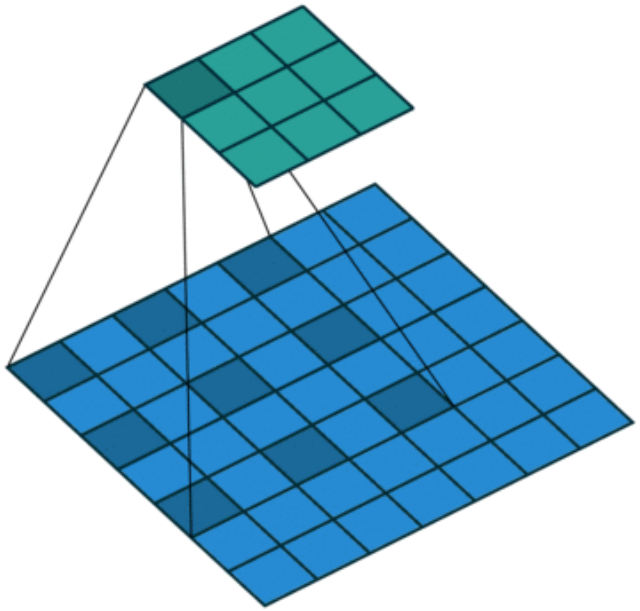
MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

ICLR 2016

Idea 3: Dilated Convolutions



MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

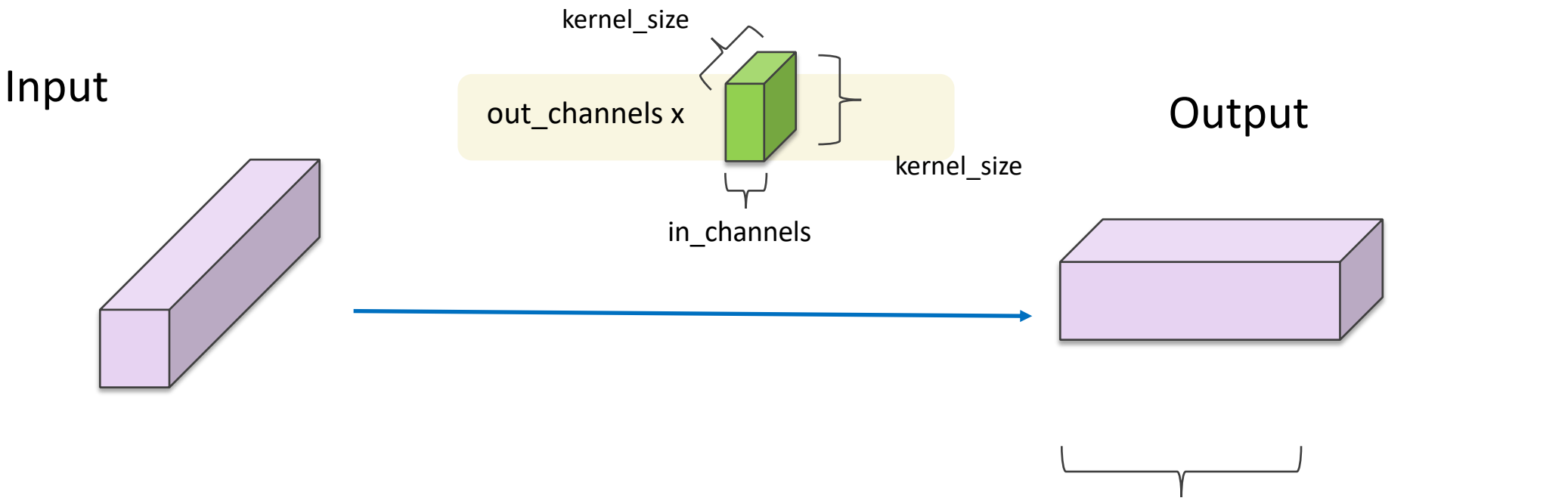
Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

ICLR 2016

Convolutional Layer in pytorch

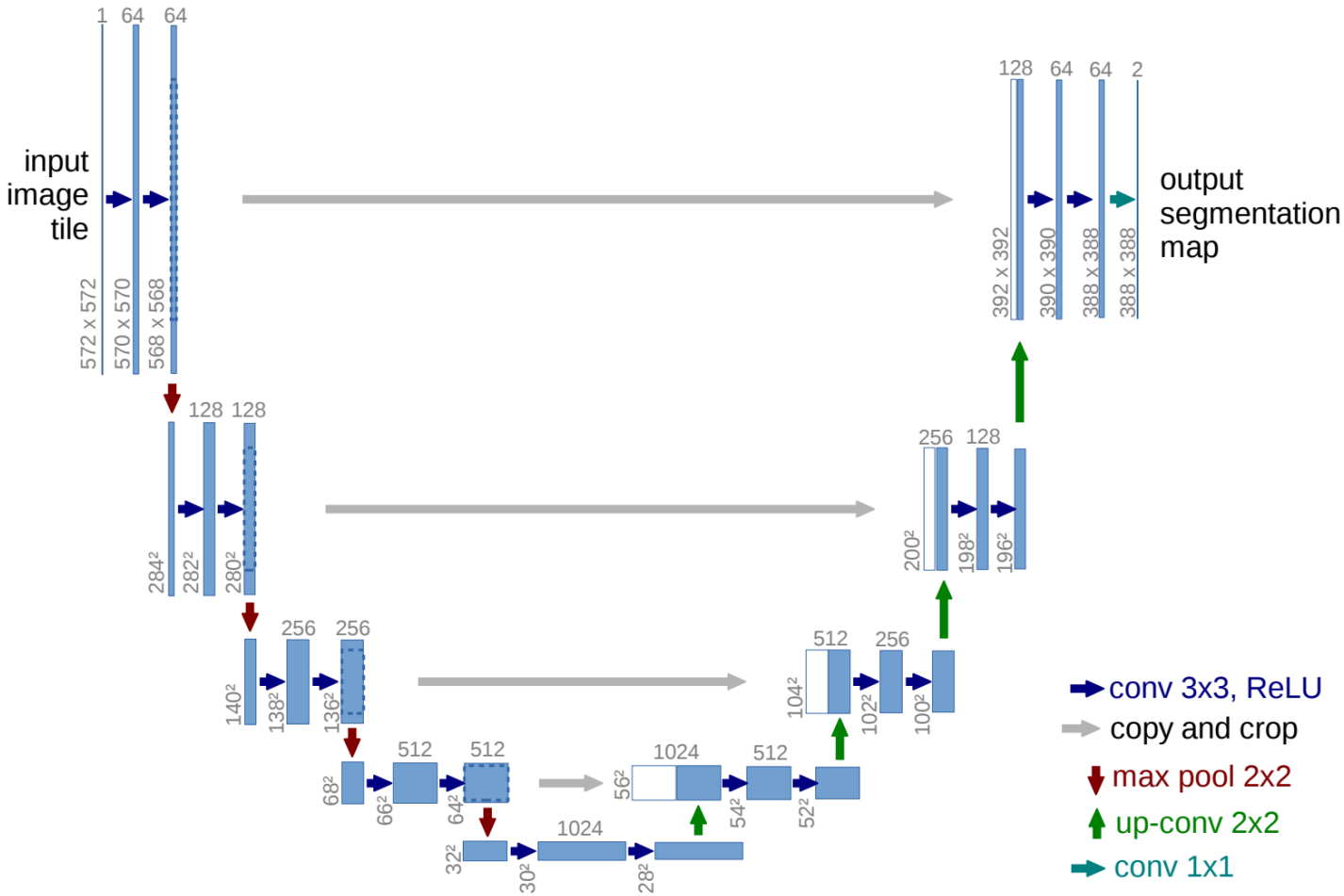
```
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
groups=1, bias=True)  [source]
```



U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany



https://arxiv.org/abs/1505.04597

https://github.com/milesial/Pytorch-UNet

https://github.com/usuyama/pytorch-unet

UNet in Pytorch

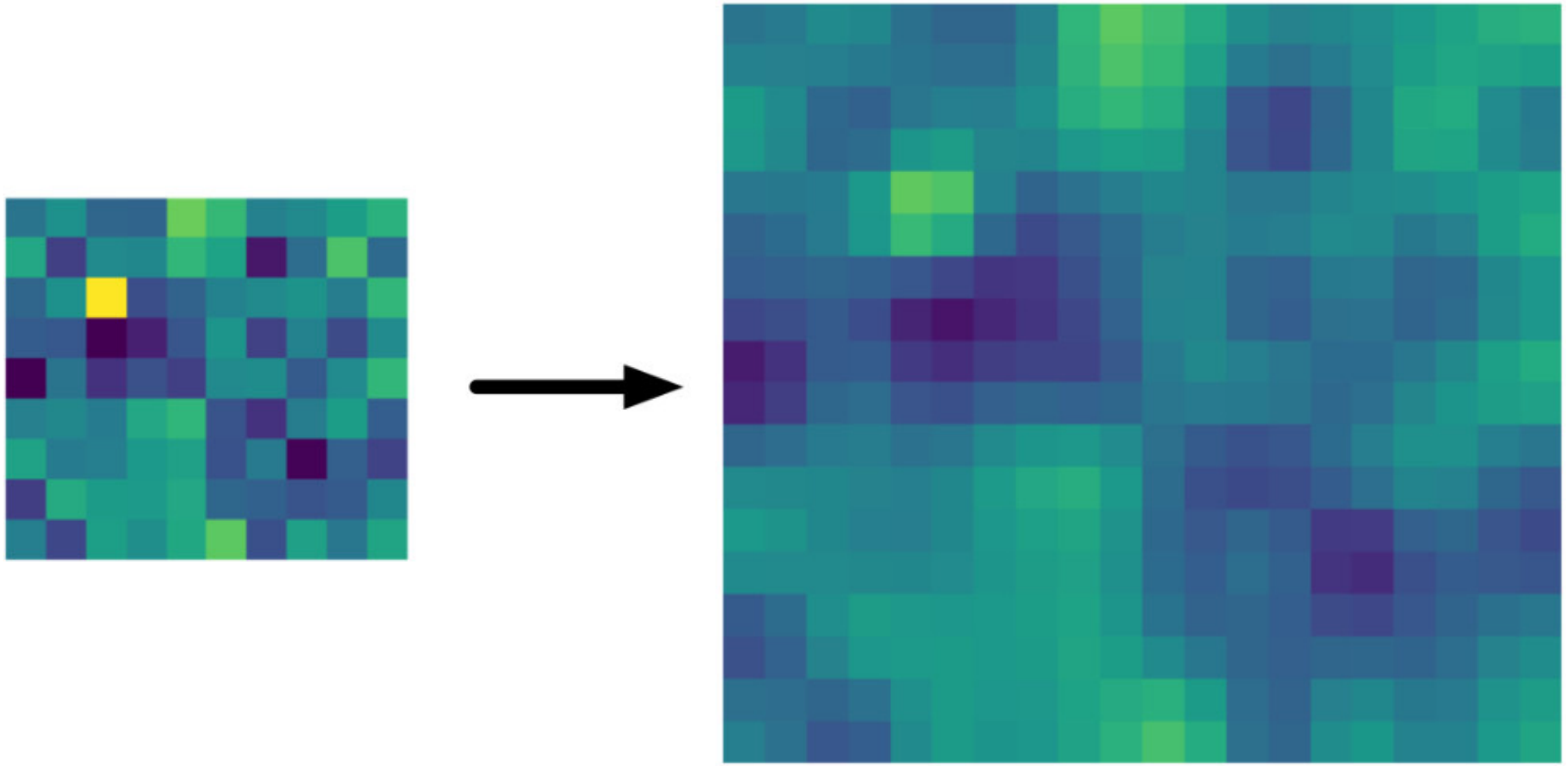
```
from .unet_parts import *


class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = (DoubleConv(n_channels, 64))
        self.down1 = (Down(64, 128))
        self.down2 = (Down(128, 256))
        self.down3 = (Down(256, 512))
        factor = 2 if bilinear else 1
        self.down4 = (Down(512, 1024 // factor))
        self.up1 = (Up(1024, 512 // factor, bilinear))
        self.up2 = (Up(512, 256 // factor, bilinear))
        self.up3 = (Up(256, 128 // factor, bilinear))
        self.up4 = (Up(128, 64, bilinear))
        self.outc = (OutConv(64, n_classes))

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits
```

Bilinear Upsampling Layer

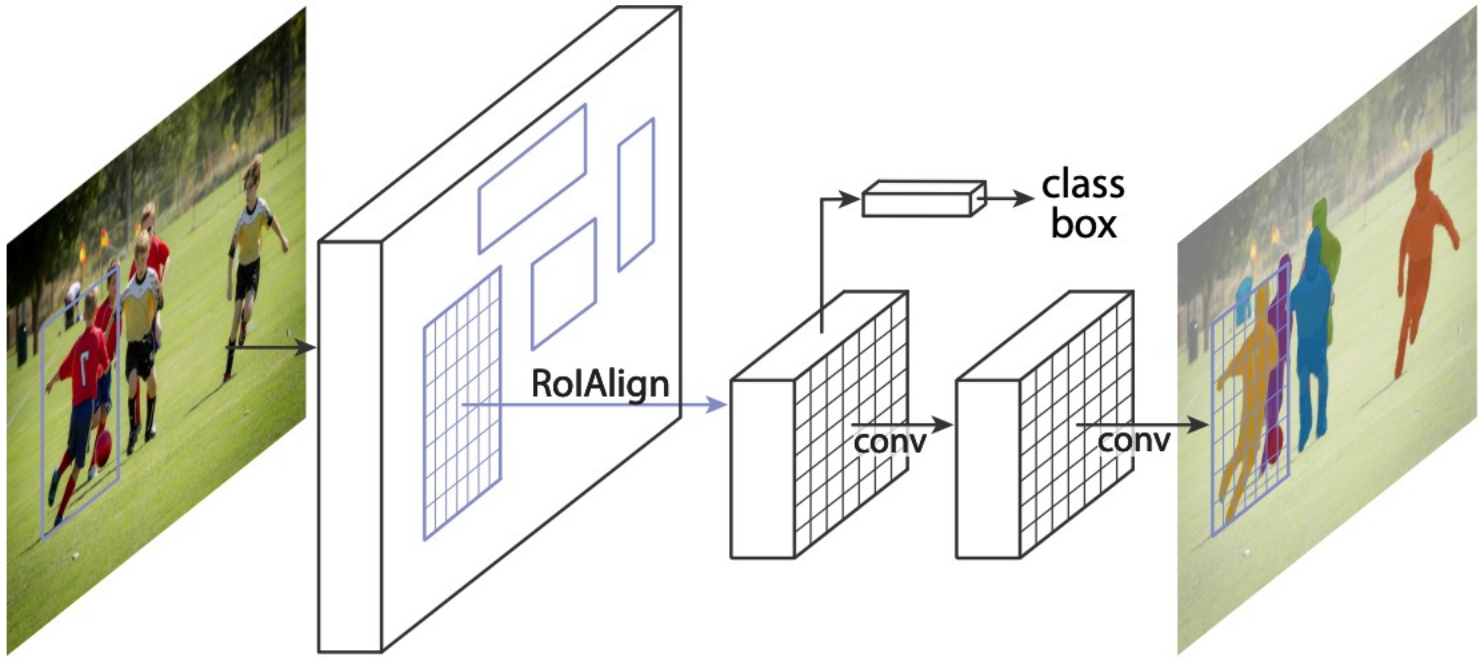


https://machinethink.net/blog/coreml-upsampling/

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick

Facebook AI Research (FAIR)



https://github.com/facebookresearch/detectron2

https://arxiv.org/abs/1703.06870

Questions