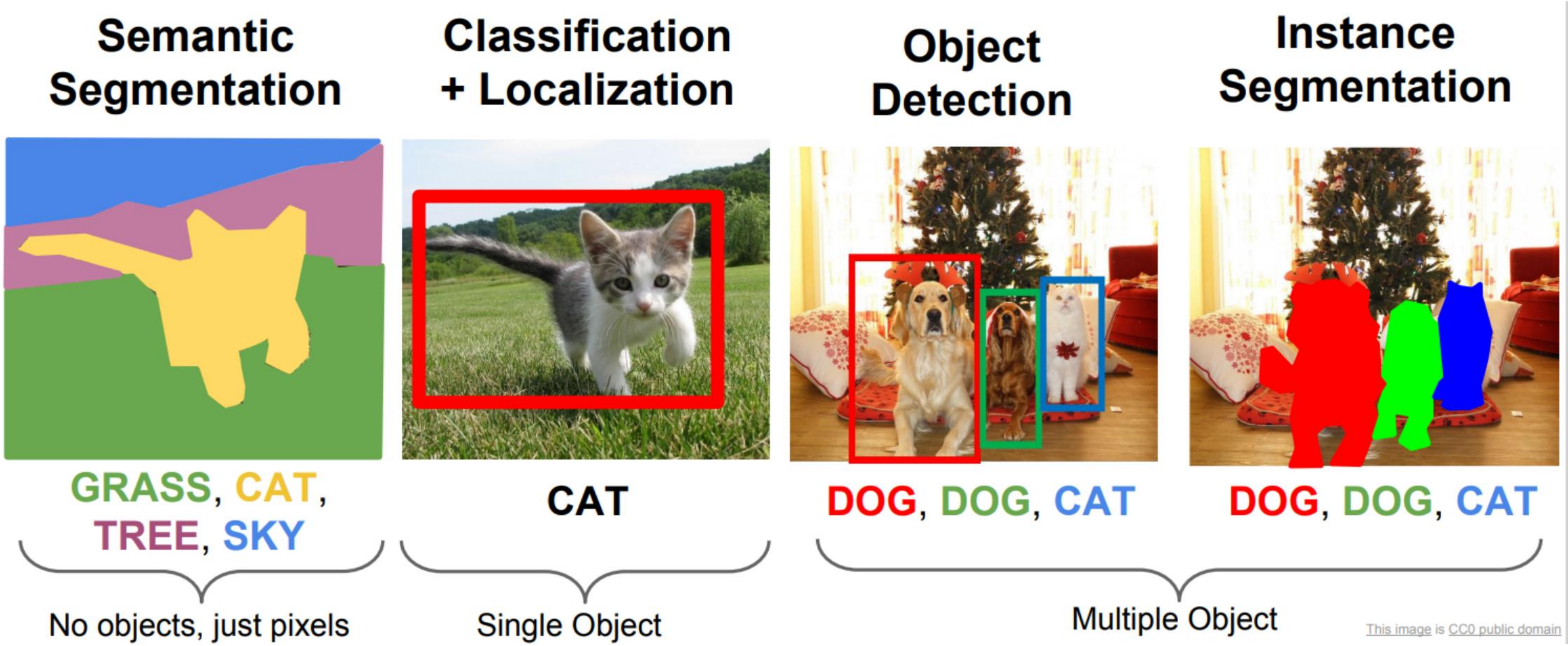
Faster R-CNN, YOLO, U-Net

KOREA Univ.
The Department of EE
Kang Taewoong

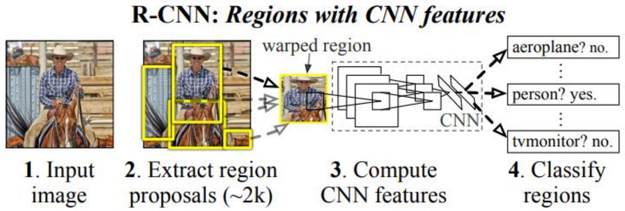
Contents

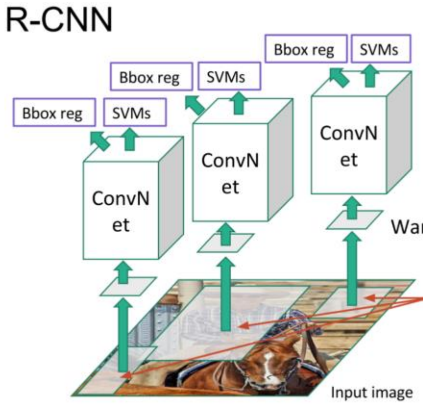
Introduction
Object Detection

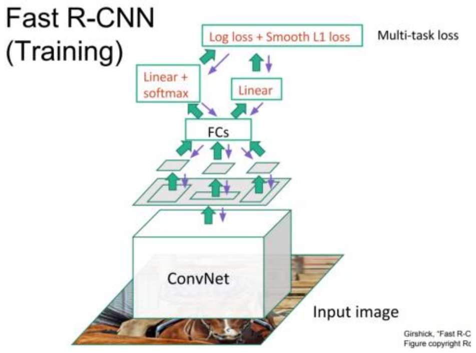


Development Process of R-CNN

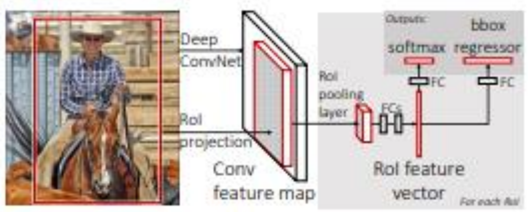
- R-CNN







- Fast R-CNN



- Faster R-CNN

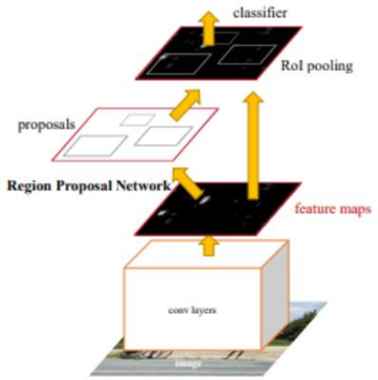


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Faster R-CNN

RPN(+Anchor)

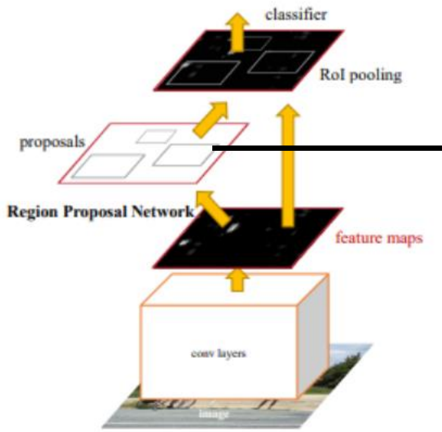
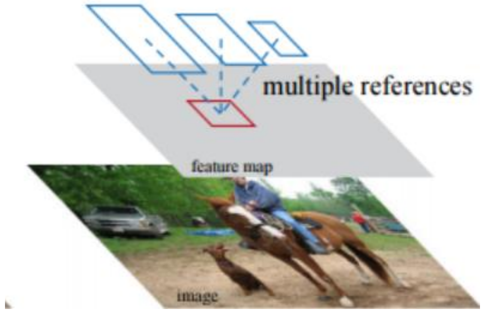
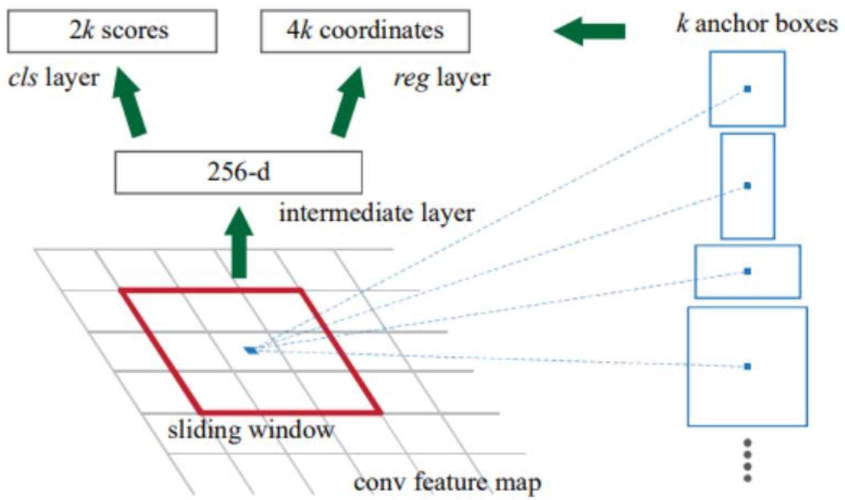


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.





- Sliding - window
- Takes advantage of GPU
- End-to-end

Training Stage

- Loss Function

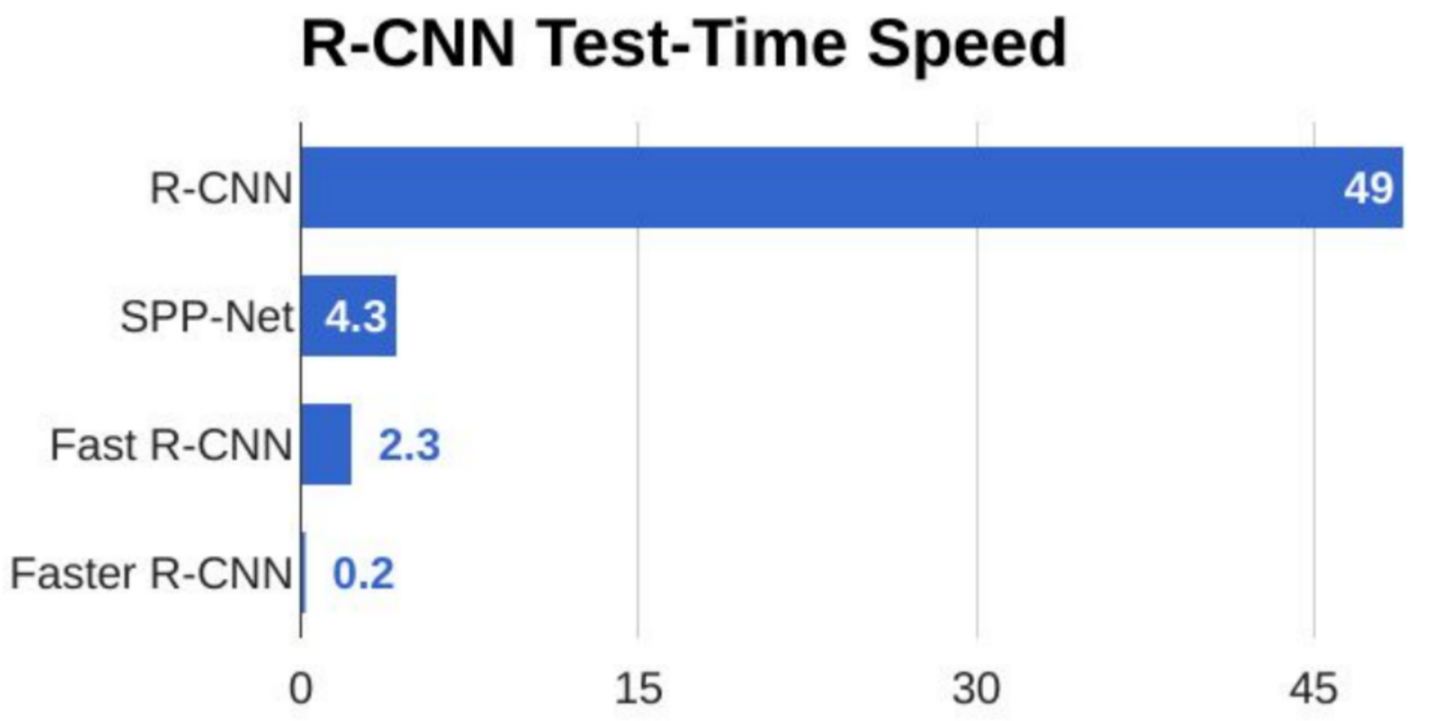
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

- 4-Stage Alternative training
  - 1st, training the RPN
  - 2nd, training a fast R-CNN
  - 3rd, fix conv layer train RPN
  - 4th, fix conv layer train fast R-CNN

- + IoU, NMS

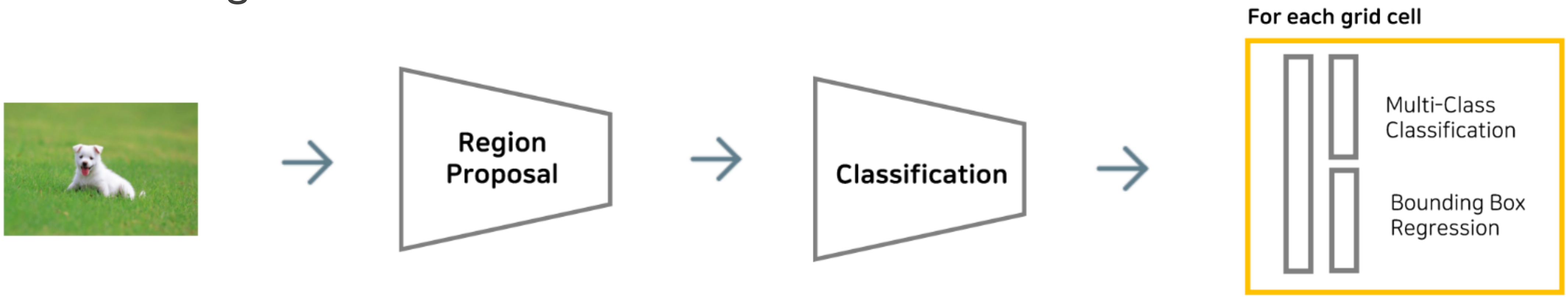
Experiments



One-Stage Detector VS. Two-Stage Detector

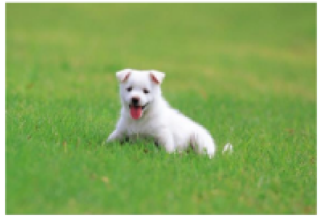
◆ One-Stage Detector

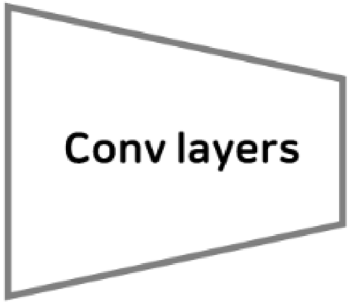
◆ Two-Stage Detector



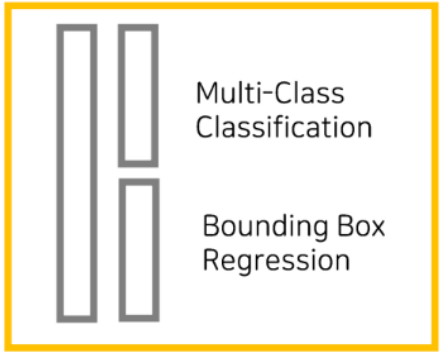
One-Stage Detector VS. Two-Stage Detector

◆ One-Stage Detector

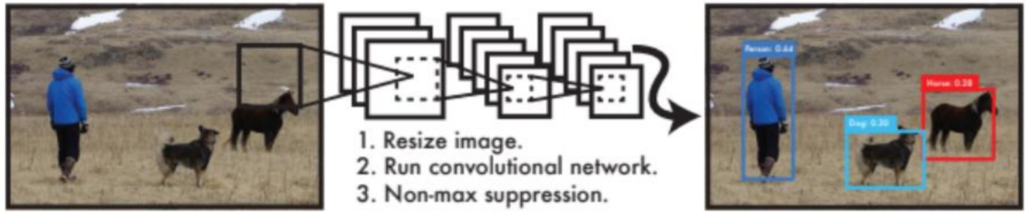




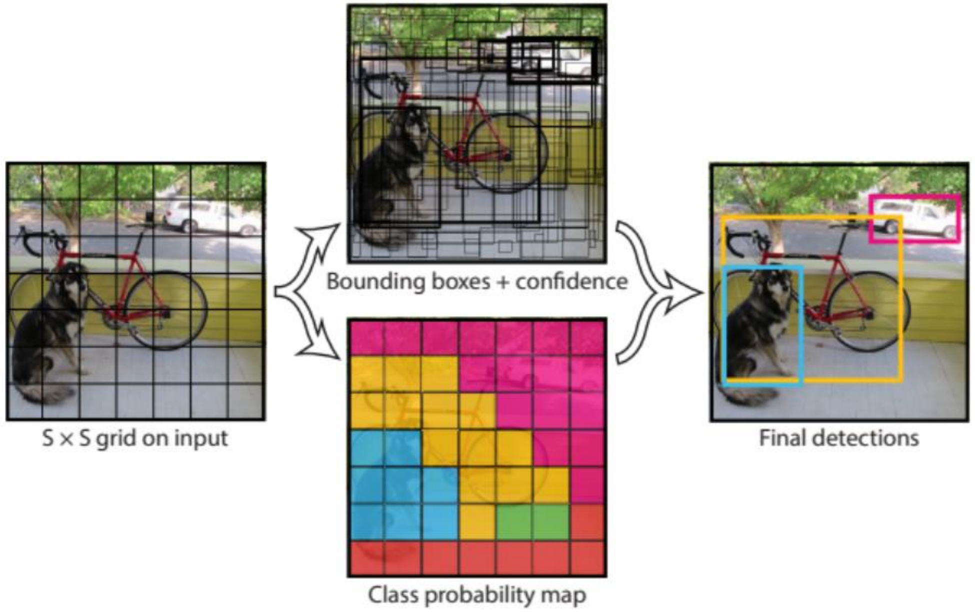
For each grid cell



◆ YOLO



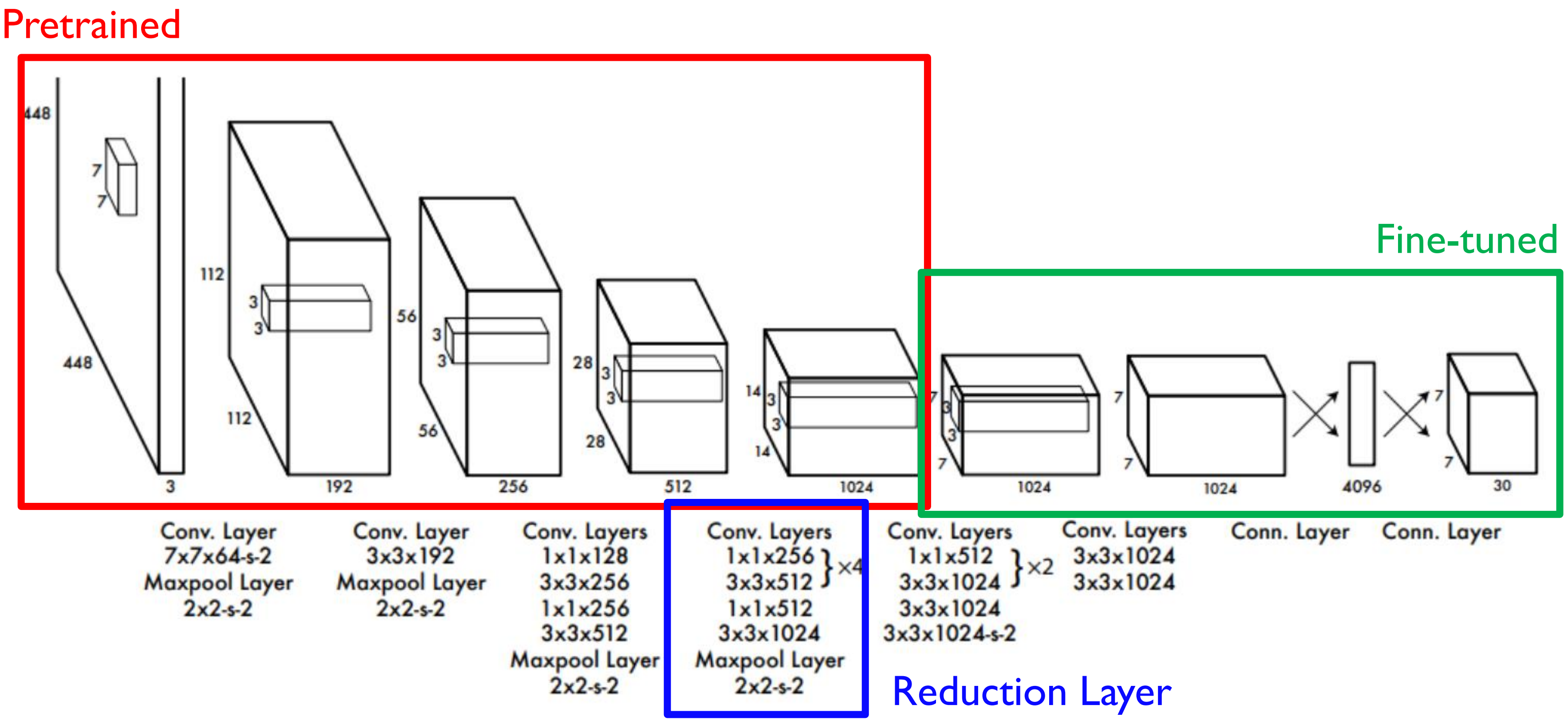
Unified Detection



- S x S grid cell => each grid cell, B bbox prediction + confidence & class probabilities
  => SxSx(B*5 + C)

+ B => [x, y, w, h, pc]   (pc = Pr(object) * IOU truth/pred)

Network Design



- Pretrained 20 conv layer + fine-tuned 4 conv layer+2 fc layer
- Fast Yolo => 9 conv layer + 2 fc layer

Training Stage

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

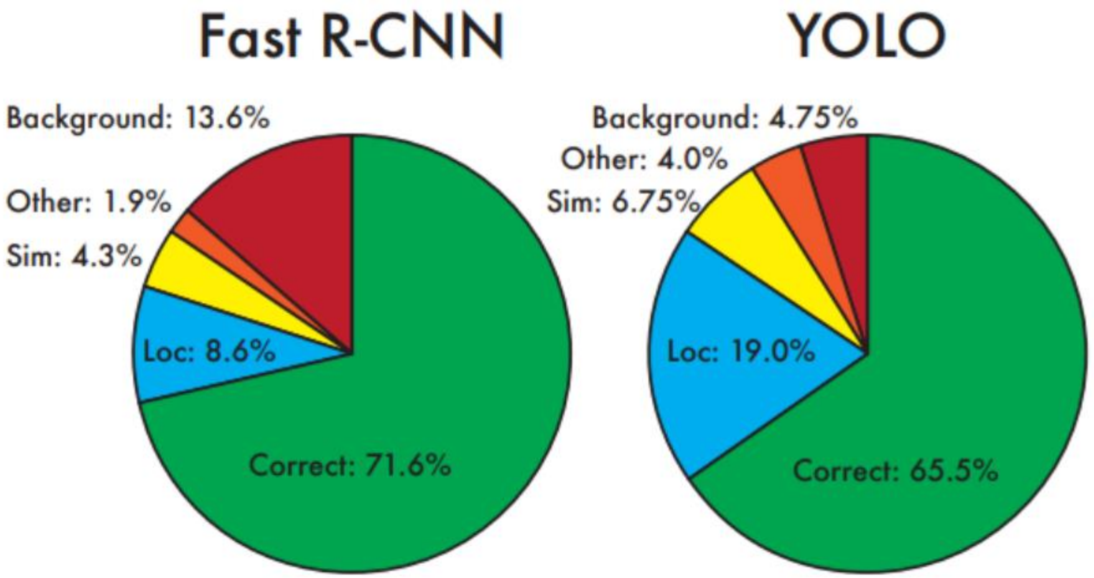
- Bbox 좌표와 GT box 좌표
- Pr(class|object)와 GT 값
- Pr(Object)*IOU truth/pred 예측값과 GT box 값

- Inference Stage

=> NMS를 통해 Object당 한 개의 Bbox만 남김

Experiments

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

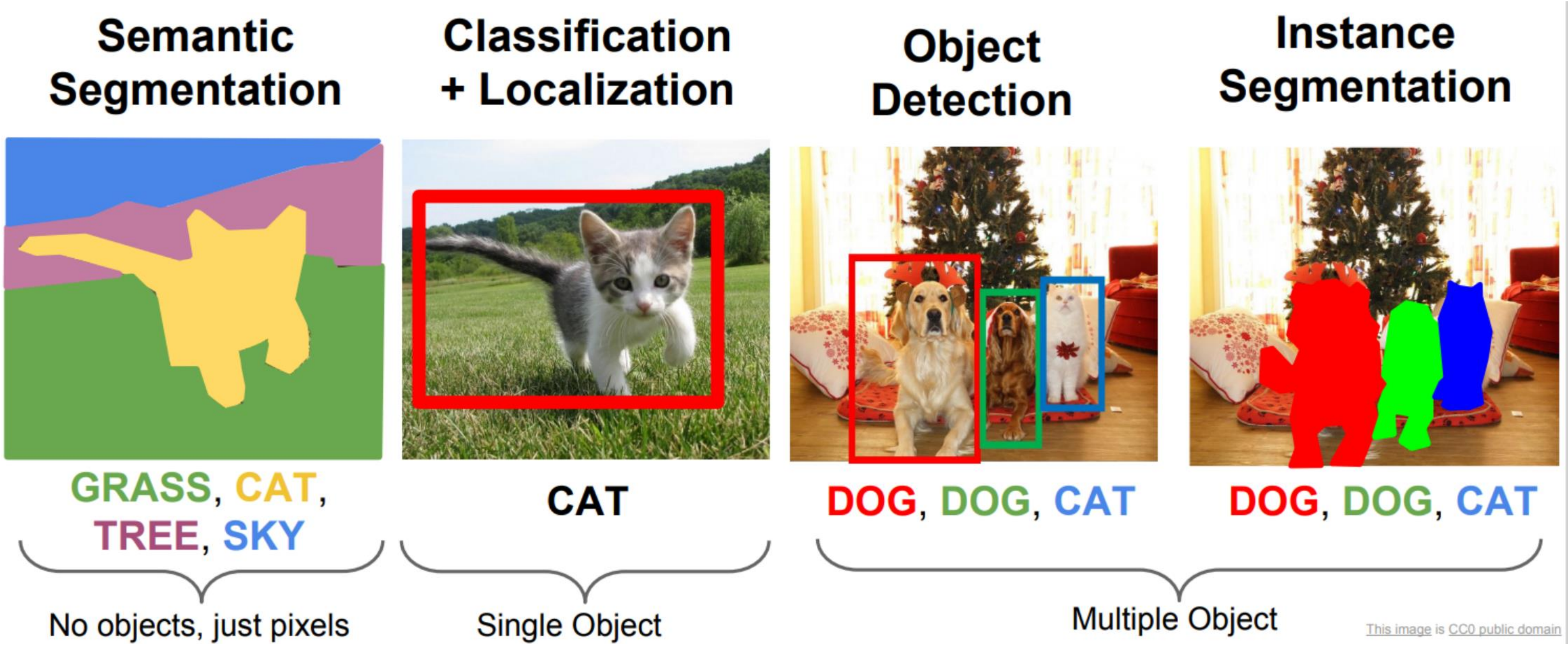


- Speed

  Fast YOLO > YOLO > Faster R-CNN
- mAP

  Faster R-CNN > Fast R-CNN > YOLO

Semantic Segmentation



Architecture (Fully Convolutional Network)

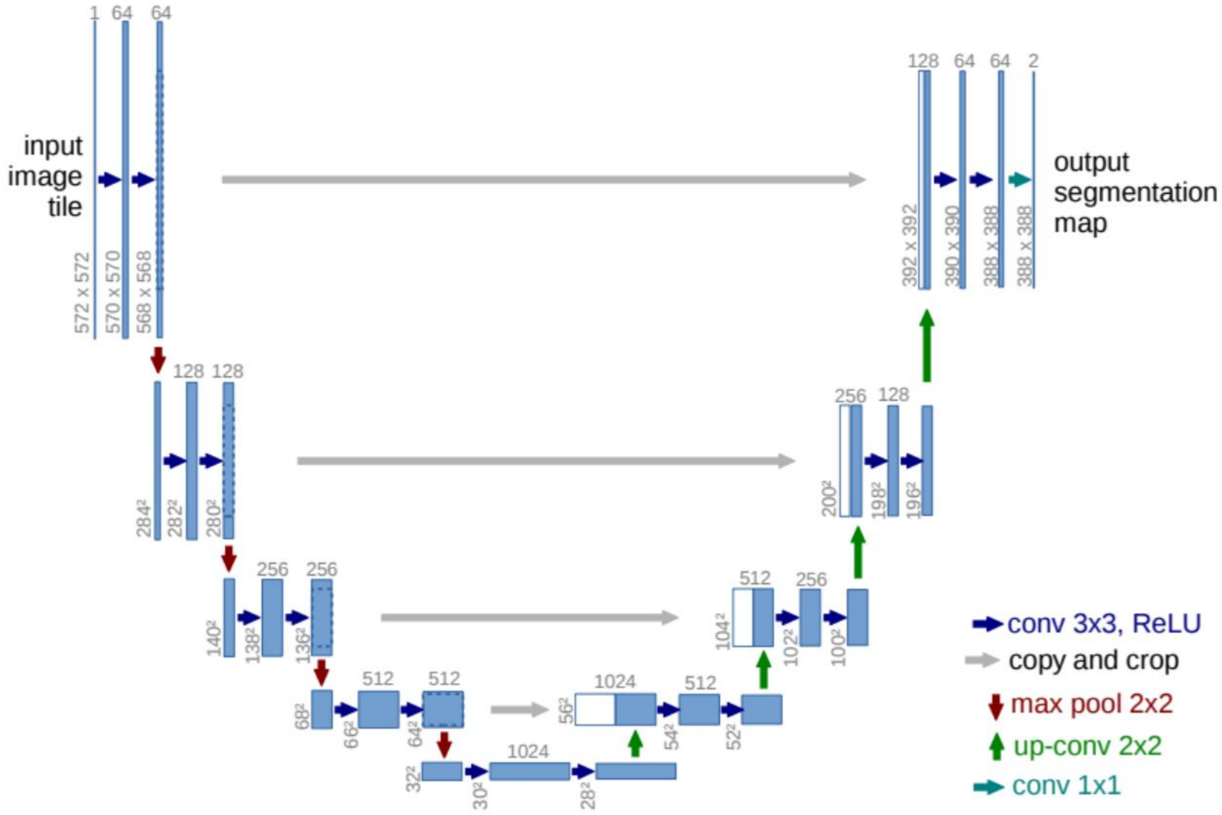
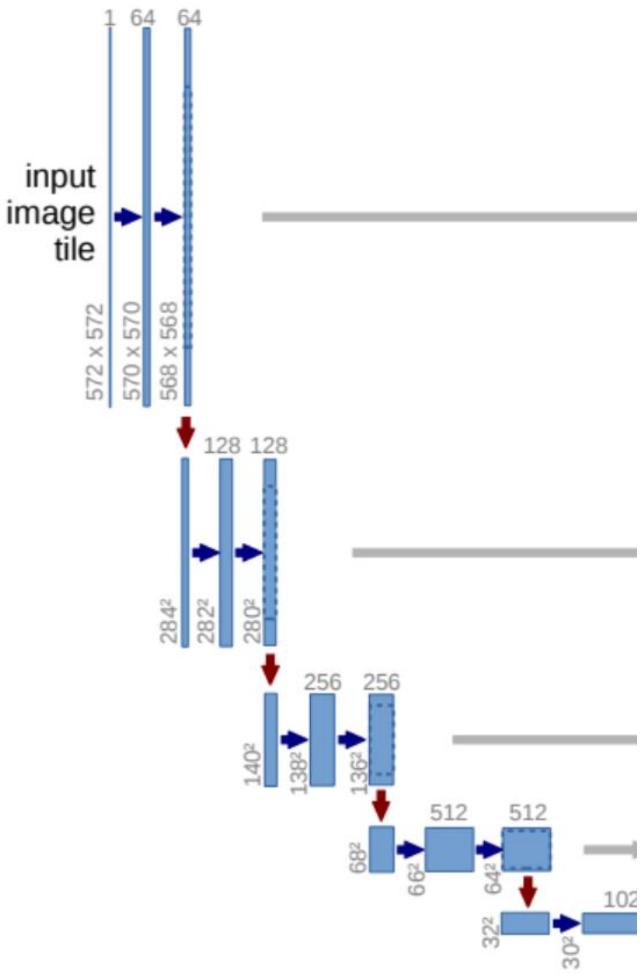


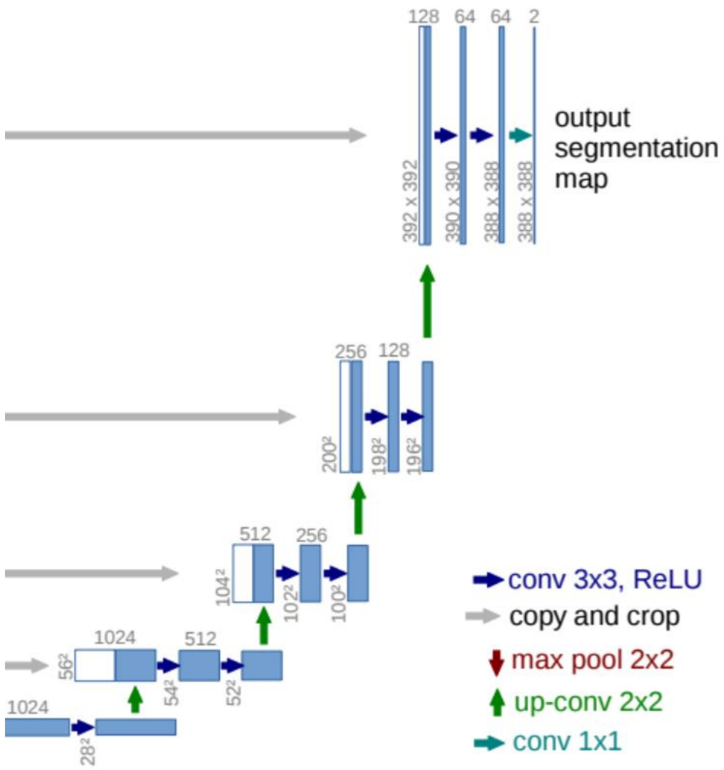
Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Architecture (Contracting Path)

- 2 x 2 Max Pooling
  - Width, height => /2
- Channel size =>*2
- Conv => ReLU => Max Pooling



Architecture (Expansive Path)



- 2 x 2 Up-Convolution
  - Width, height => *2
- Channel size => /2
- Contraction path Feature crop and concatenate
- Conv => ReLU => Max Pooling

Training Stage (Overlap-tile Strategy)

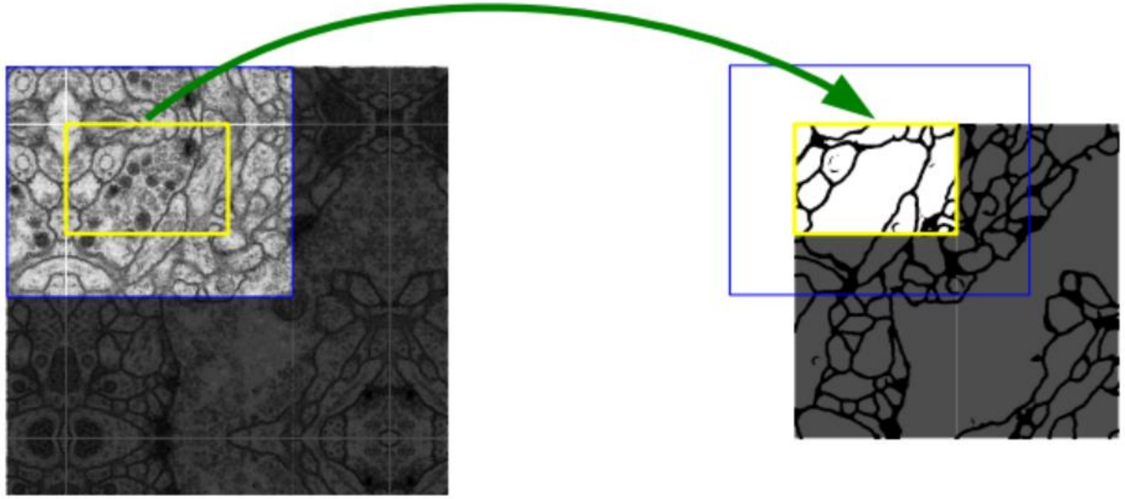


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

- +) Data Augmentation
  - Shift, Rotation and Random-elastic Deformation

Training Stage (Objective Function)

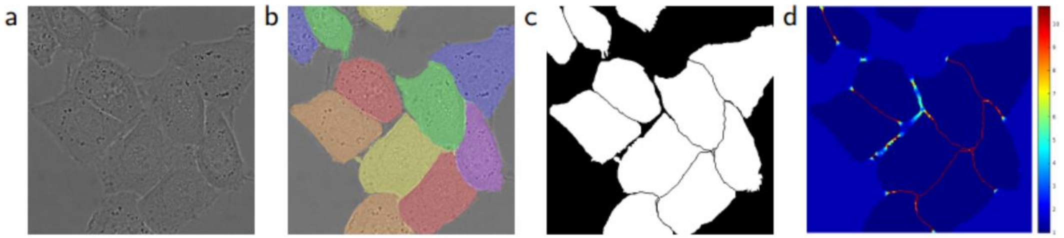
- Pixel-wise Softmax

$$p_k(\mathbf{x}) = \exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K} \exp(a_{k'}(\mathbf{x}))\right)$$

- Cross Entropy Loss

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

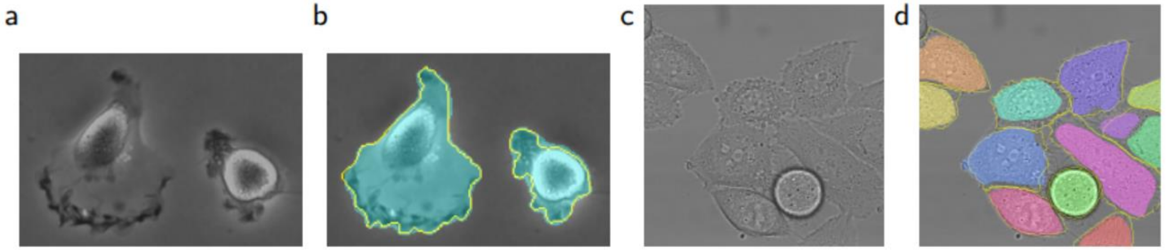
$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$



Experiments

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

Segmentation results (IOU)



Reference

- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- Fast R-CNN
- Rich feature hierarchies for accurate object detection and semantic segmentation
- You Only Look Once: Unified, Real-Time Object Detection
- U-Net: Convolutional Networks for Biomedical Image Segmentation
- CS231n-lecture note 11

Thank you