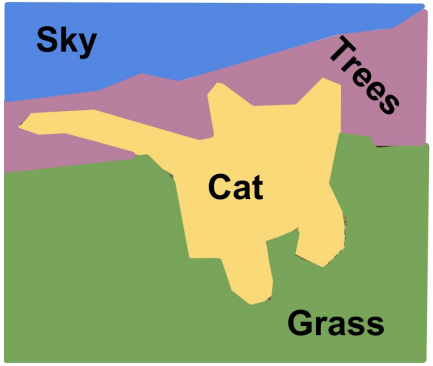
Segmentation

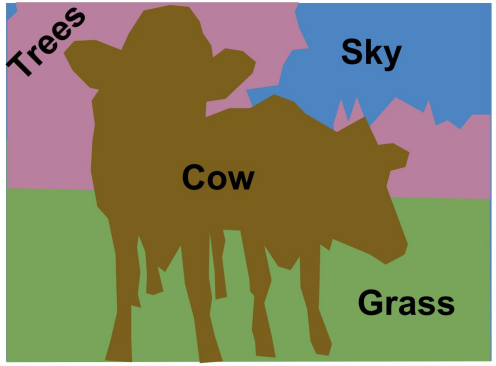
Vladislav Goncharenko
Spring 2024
MIPT, Moscow

Outline

- Segmentation task
- Simple solutions
- Upsampling methods
  - Unpooling
  - Transposed convolution
- FCN
- DeconvNet
- SegNet
- U-Net
- Mask R-CNN

Semantic Segmentation





Input image (C x H x W)

Output

mask of classes (1 x H x W)

Assumption

just class label for each pixel

Instance Segmentation

Input image (C x H x W)

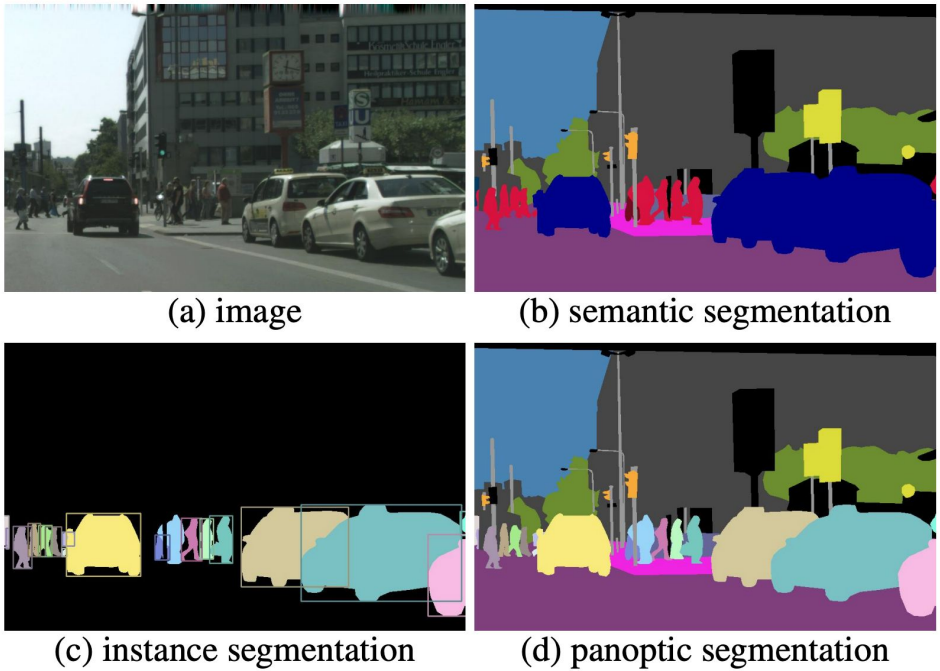
Output

mask of class instances (detection + 1 x H x W)

Assumption

separated masks for each distinct object

Panoptic segmentation



Input image (C x H x W)

Output

mask of class instances (detection + 1 x H x W)

Assumption

separated masks for each distinct object

https://arxiv.org/pdf/1801.00868.pdf

Panoptic links

- http://presentations.cocodataset.org/ECCV18/COCO18-Panoptic-PKU_360.pdf
- https://viso.ai/deep-learning/panoptic-segmentation-a-basic-to-advanced-guide-2024/

Metrics

- IoU
- Pixel accuracy
- Average Precision (AP)

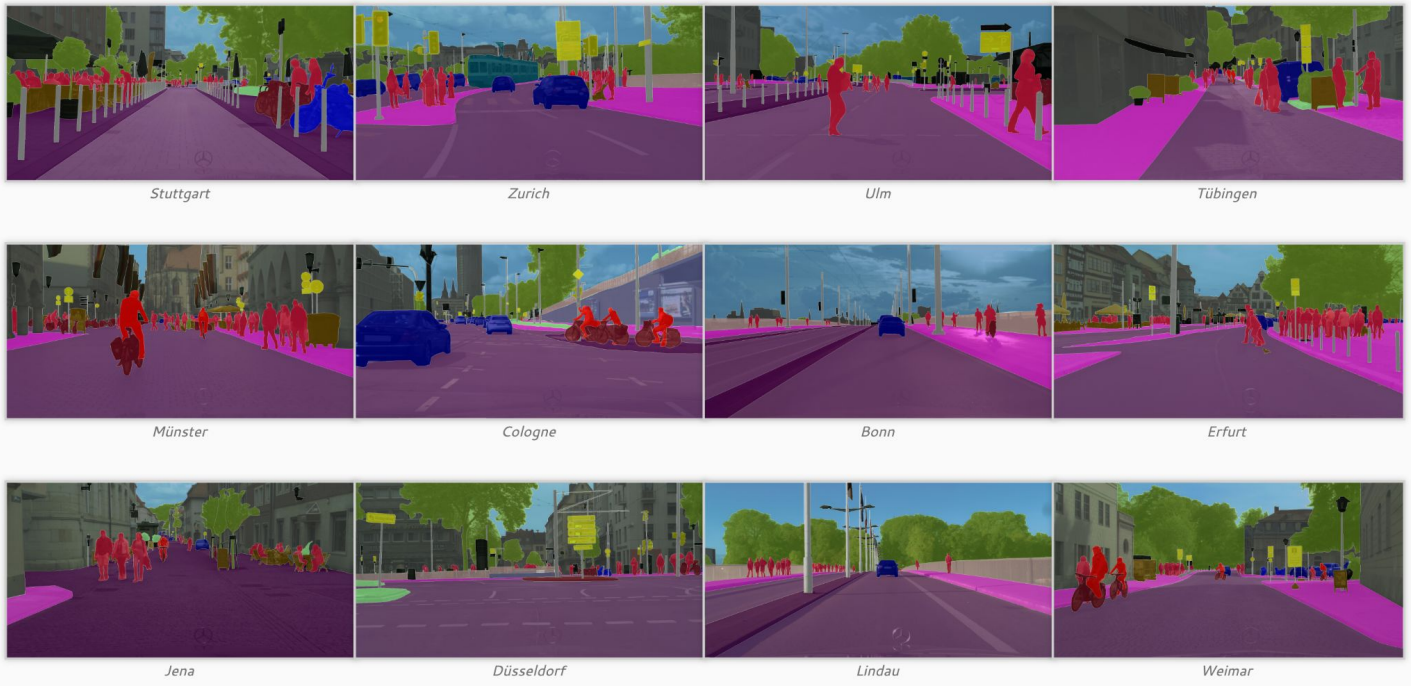
Datasets

COCO Panoptic Segmentation

https://cocodataset.org/#panoptic-2018

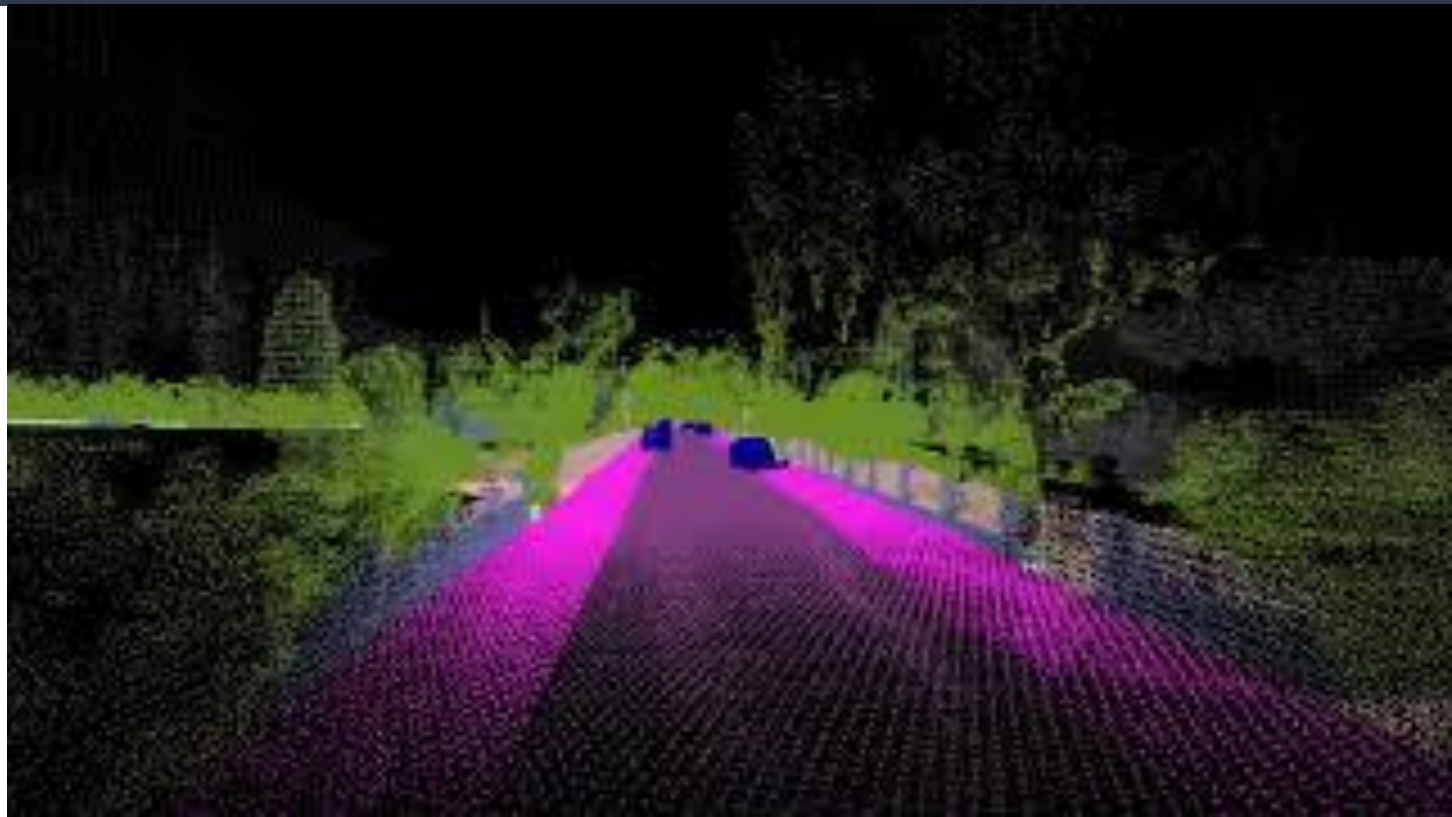
Cityscapes

https://www.cityscapes-dataset.com/



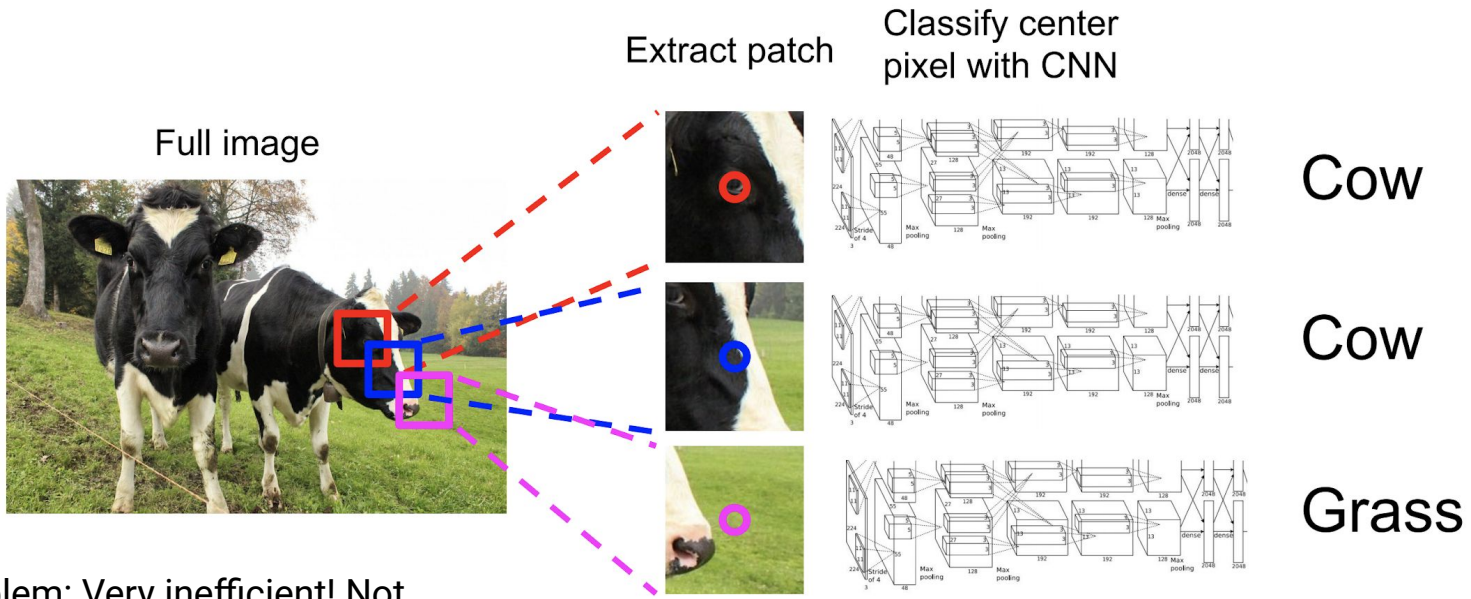
KITTI

https://www.cvlibs.net/datasets/kitti-360/



Models

Naive solution

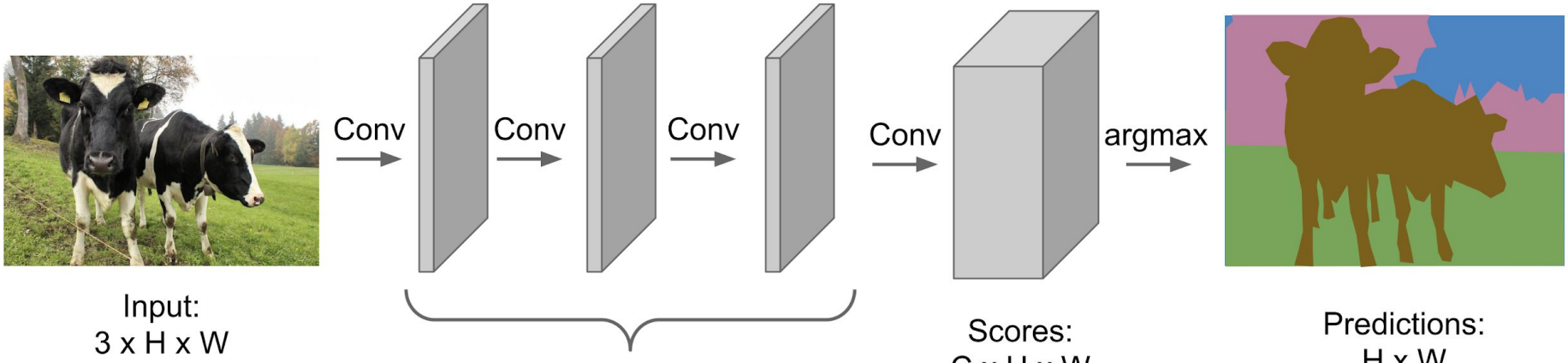


Problem: Very inefficient! Not reusing shared features between overlapping patches (same as R-CNN)

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

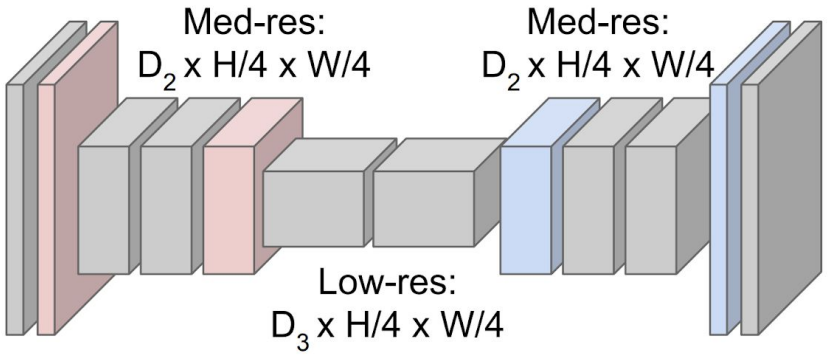


Problem: convolutions at original image resolution will be very expensive (remember VGG architecture)

Downsampling and Upsampling

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W

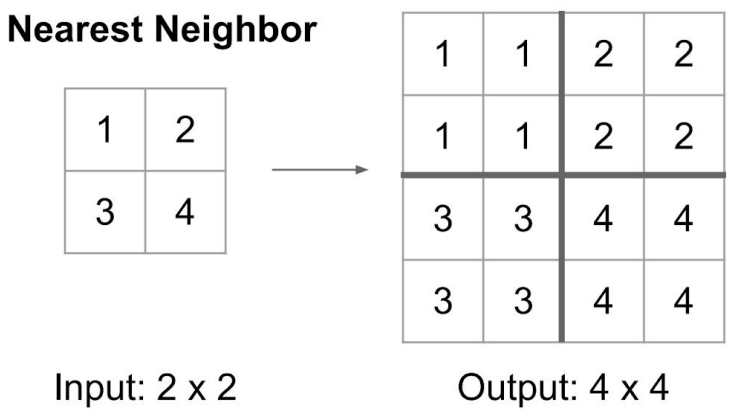


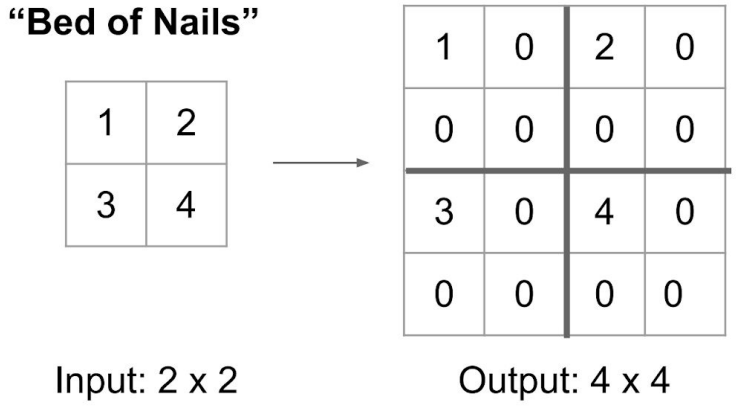
High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Simple unpooling





Max Unpooling

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

...

Rest of the network

Max Unpooling
Use positions from pooling layer

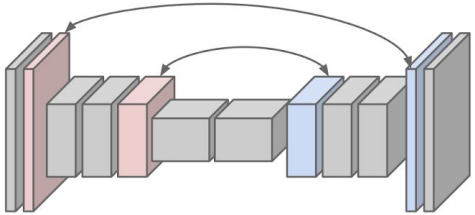
1	2
3	4

Input: 2 x 2

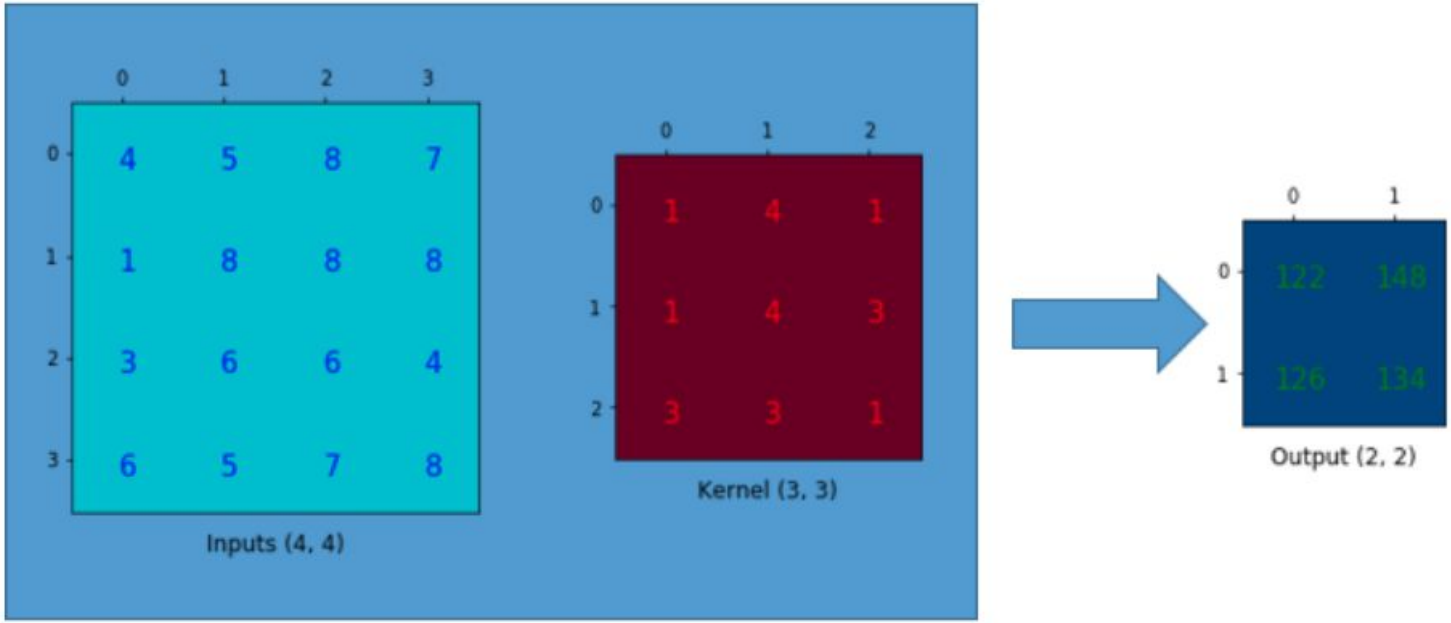
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

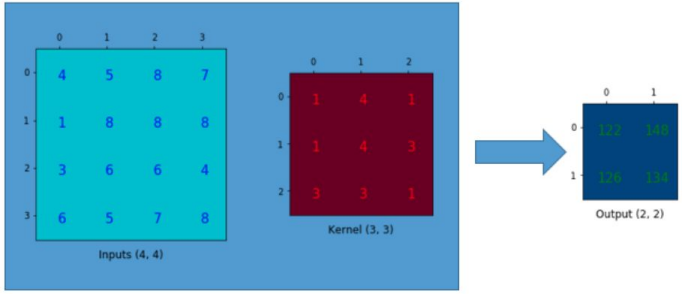


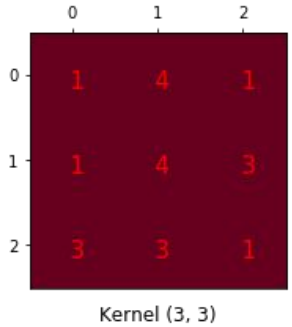
Regular convolution

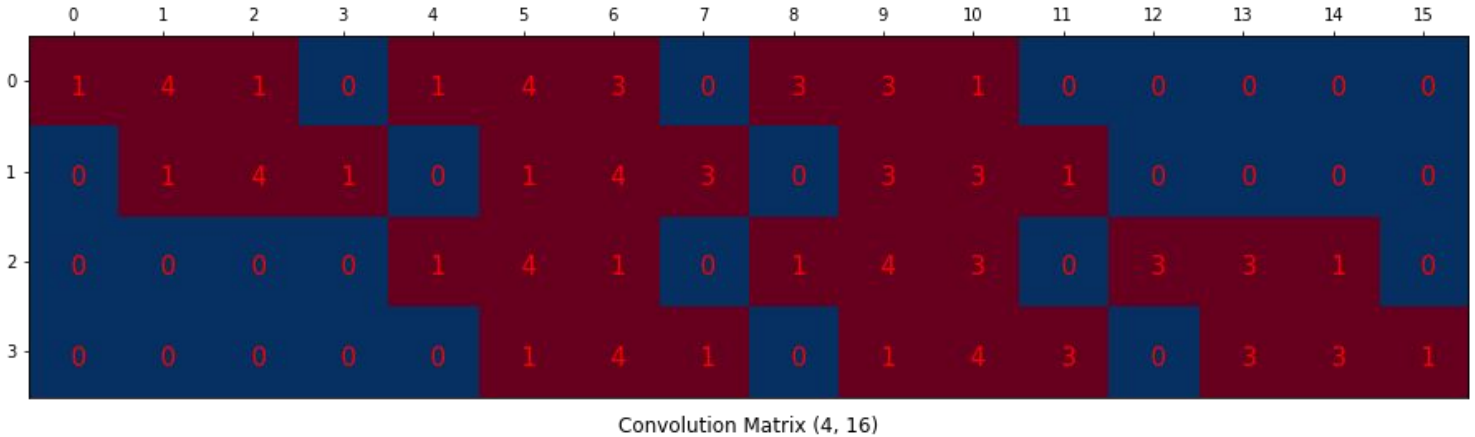


Images credentials

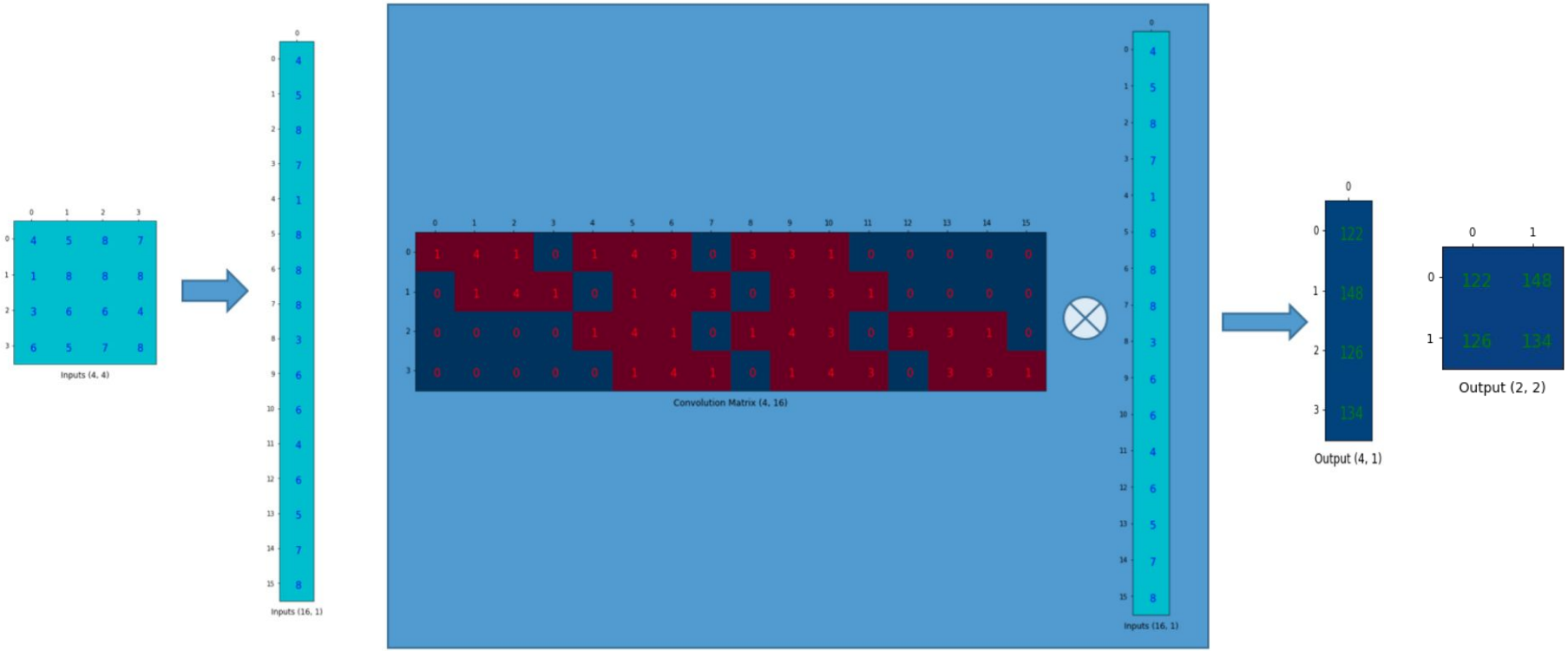
Regular convolution: matrix form



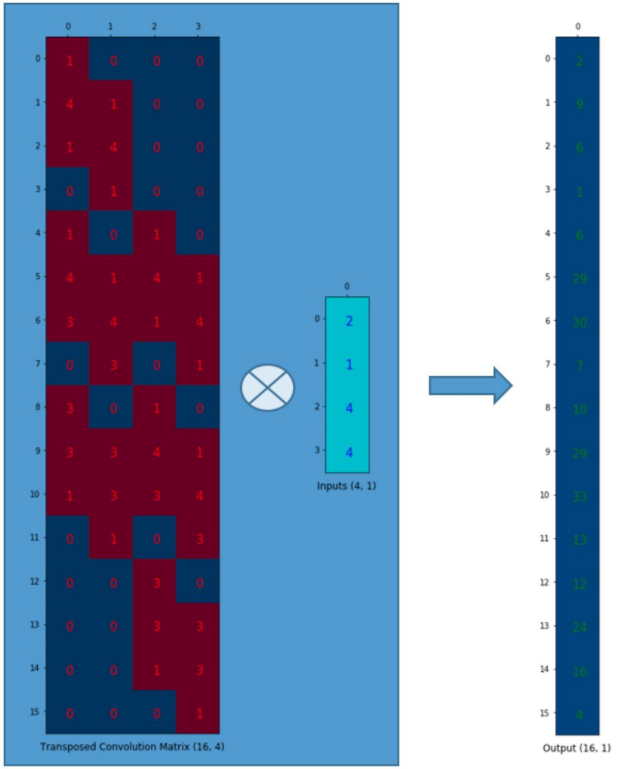


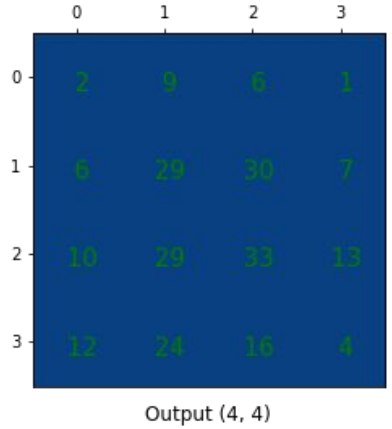


Regular convolution: matrix form



Transposed convolution matrix





Convolution as Matrix Multiplication (1D)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Terms

Some researchers denote Transposed convolution with term “Deconvolution”. However:

deconvolution is an algorithm-based process used to reverse the effects of convolution on recorded data [Wikipedia]

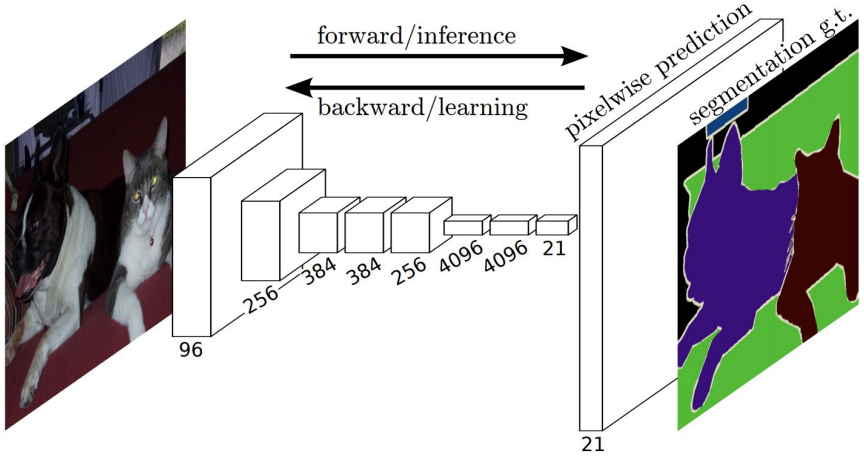
What we do is not a deconvolution.

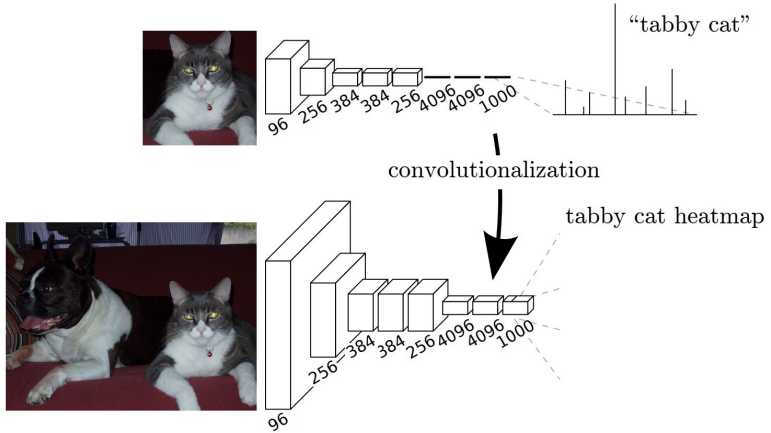
Please be precise in terms

Right terms:

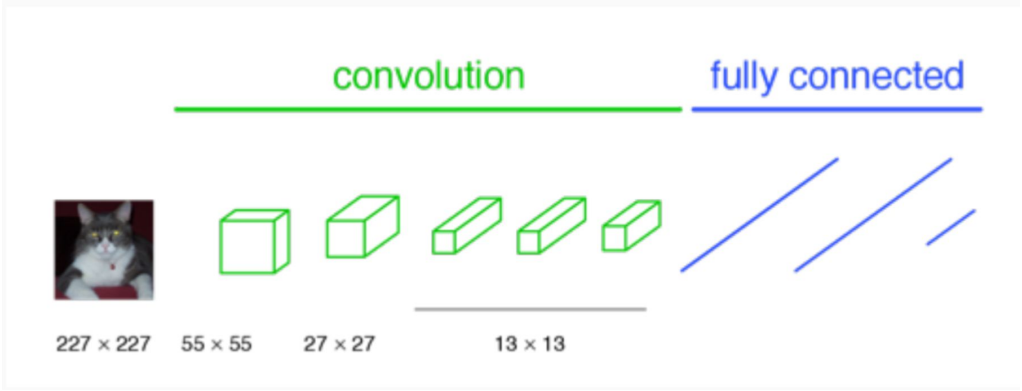
- Transposed convolution
- Upconvolution

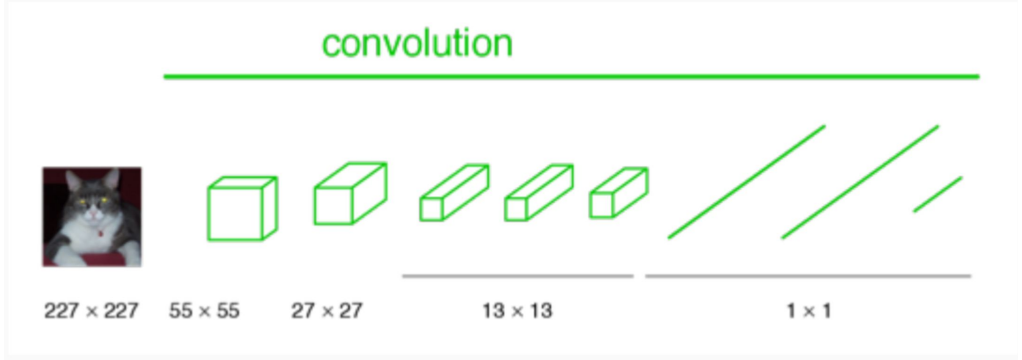
FCN – Fully Convolutional Network



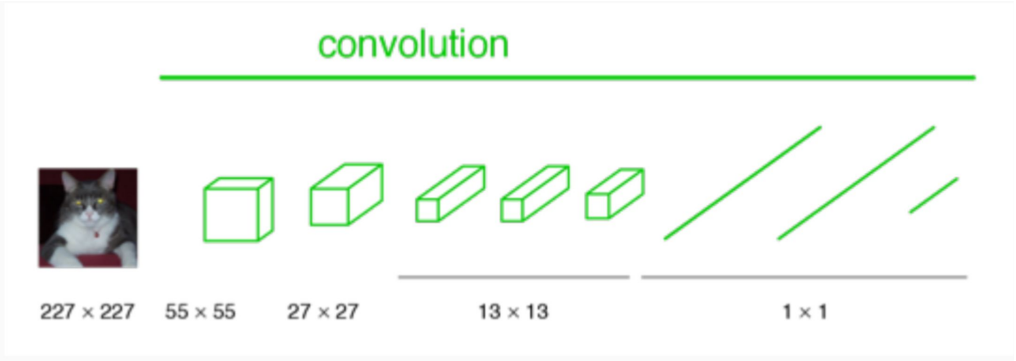


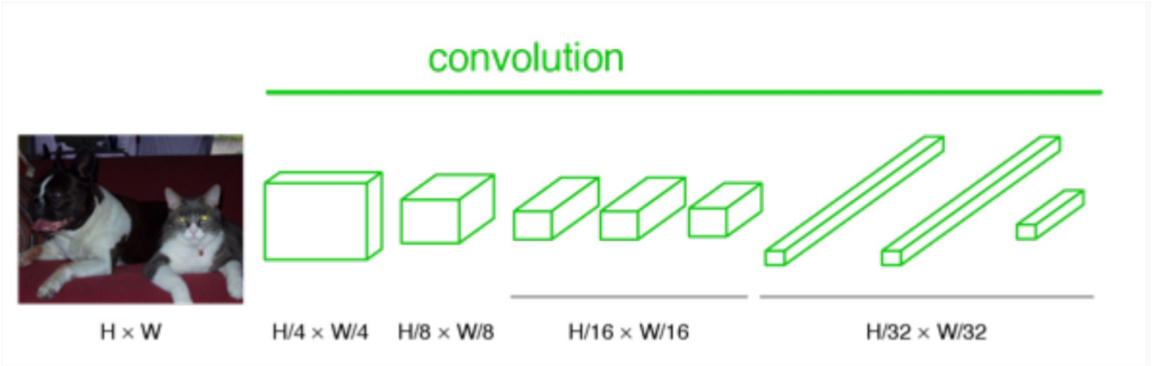
Regular CNN to FCN



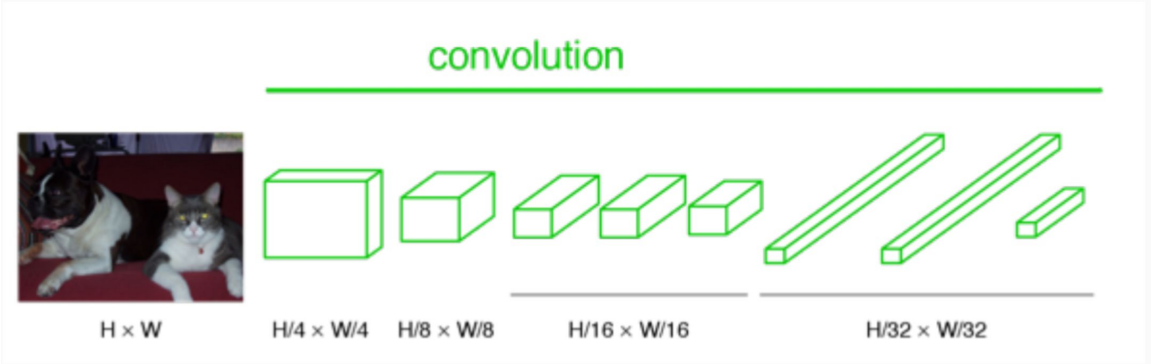


Regular CNN to FCN



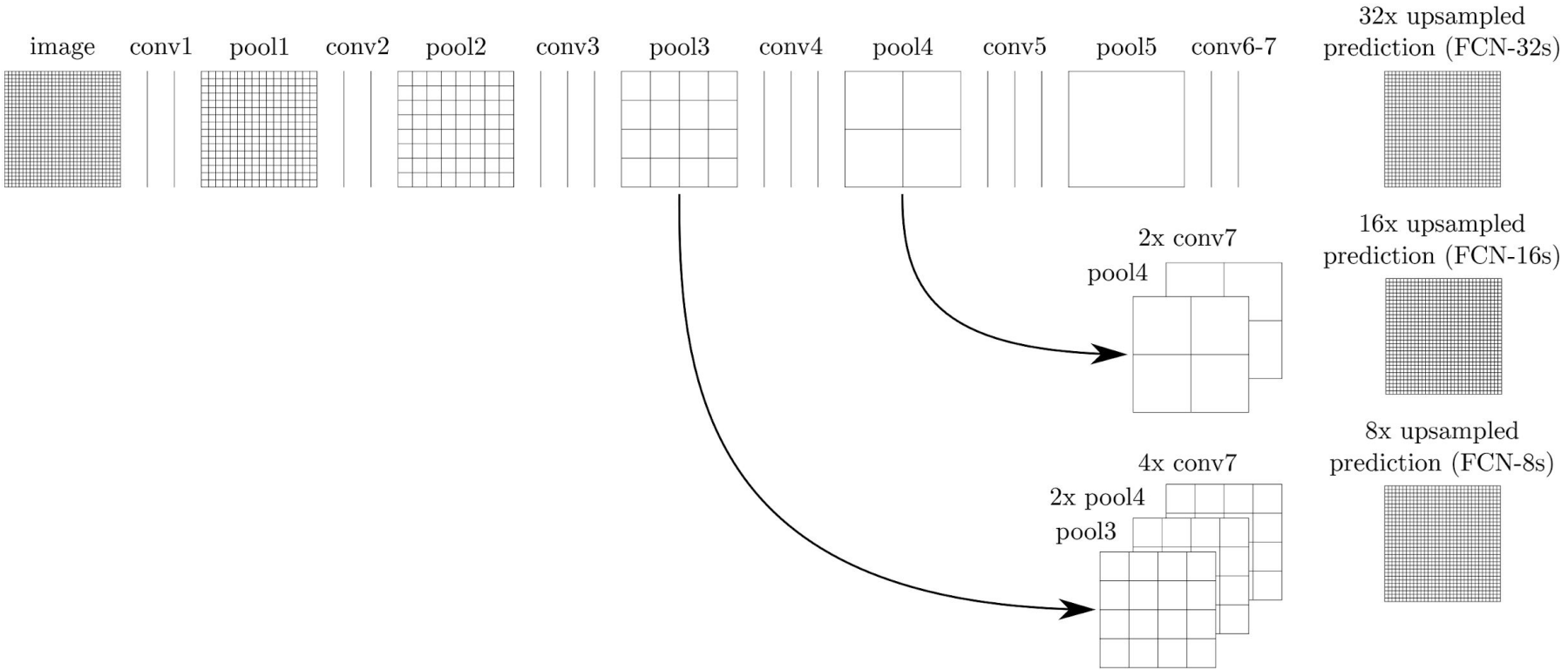


Regular CNN to FCN

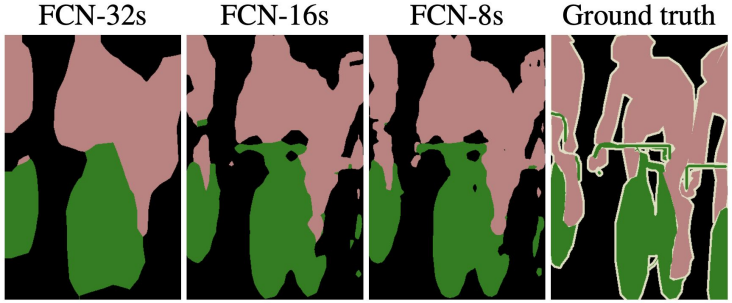


Problem: we lose resolution

FCN: Introduce skip connections

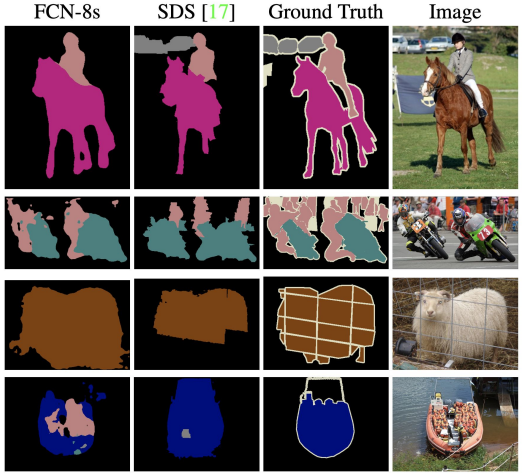


FCN: Results

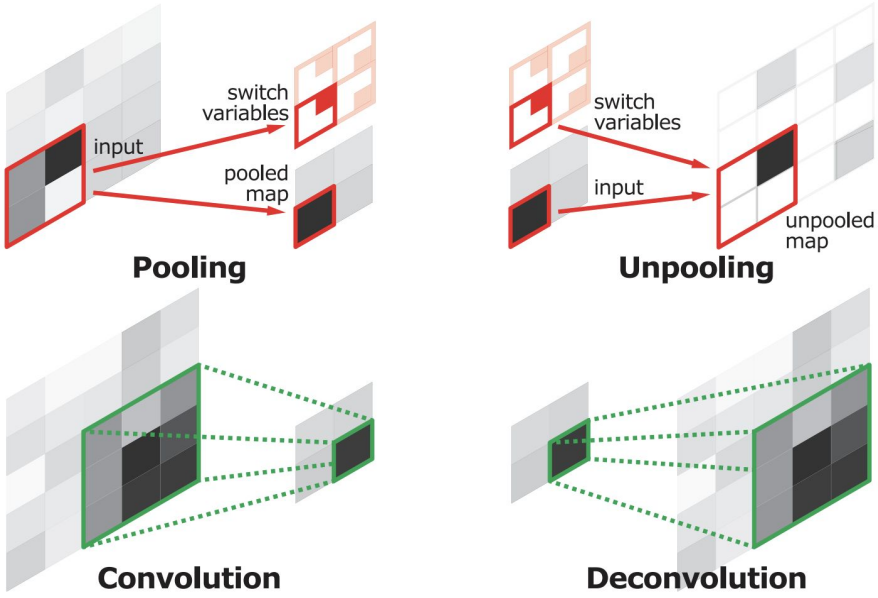


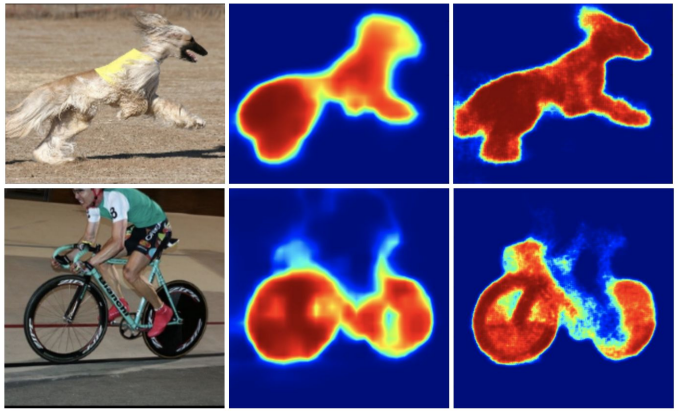
	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32



DeconvNet: use Max Unpooling





(a) Input image (b) FCN-8s (c) Ours

Figure 5. Comparison of class conditional probability maps from FCN and our network (top: dog, bottom: bicycle).

DeconvNet: use Max Unpooling

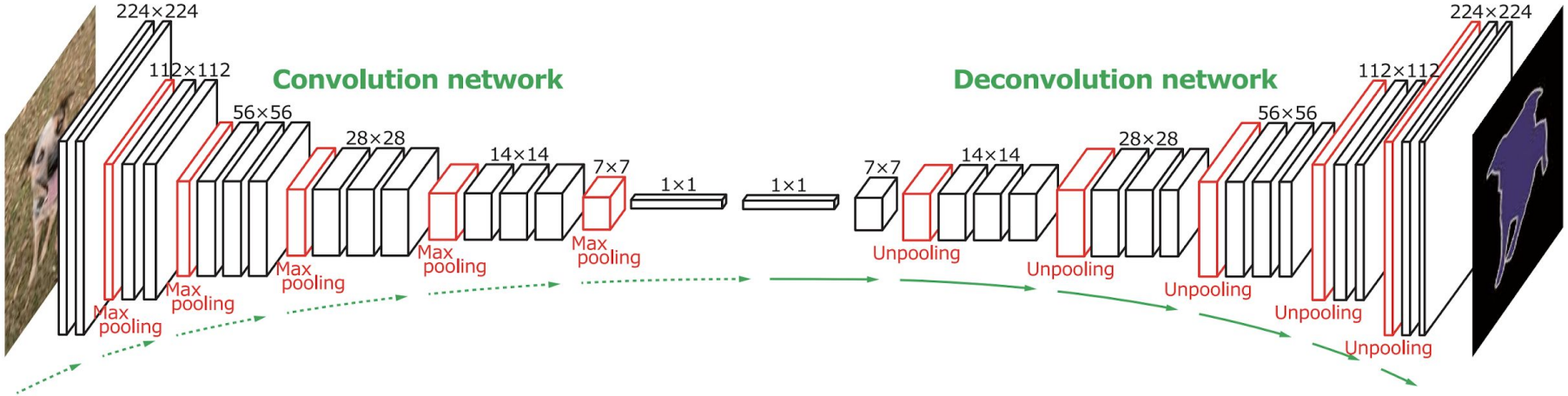
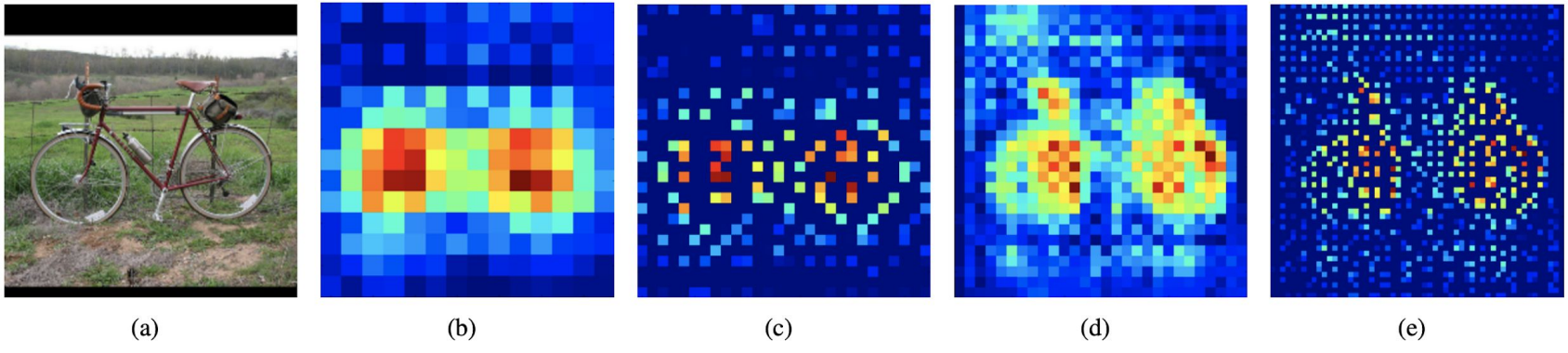


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

DeconvNet: use Max Unpooling



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

SegNet: Further development

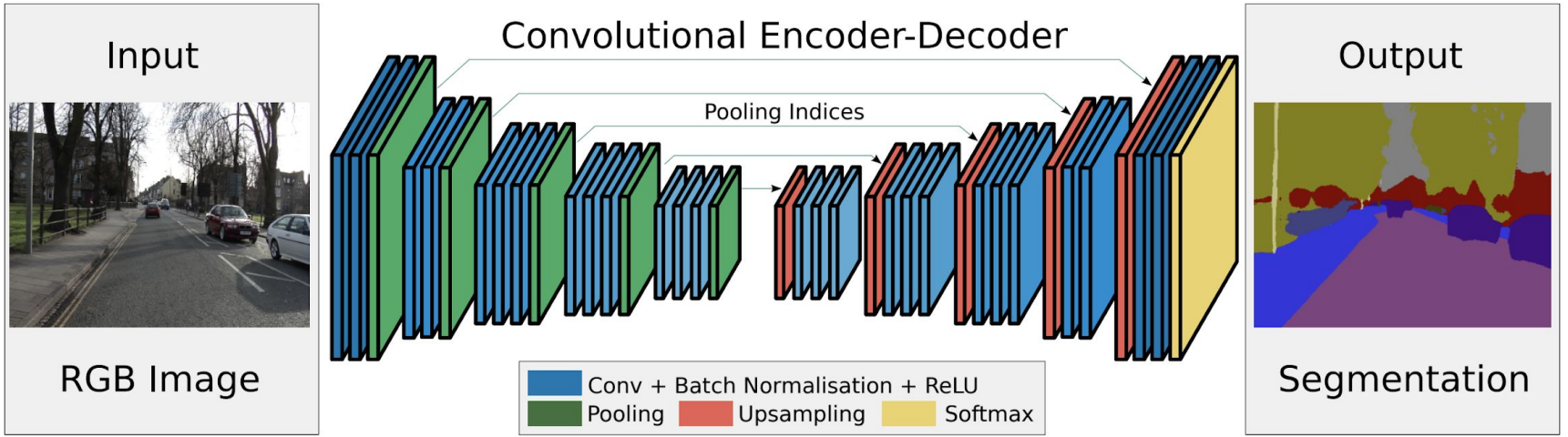
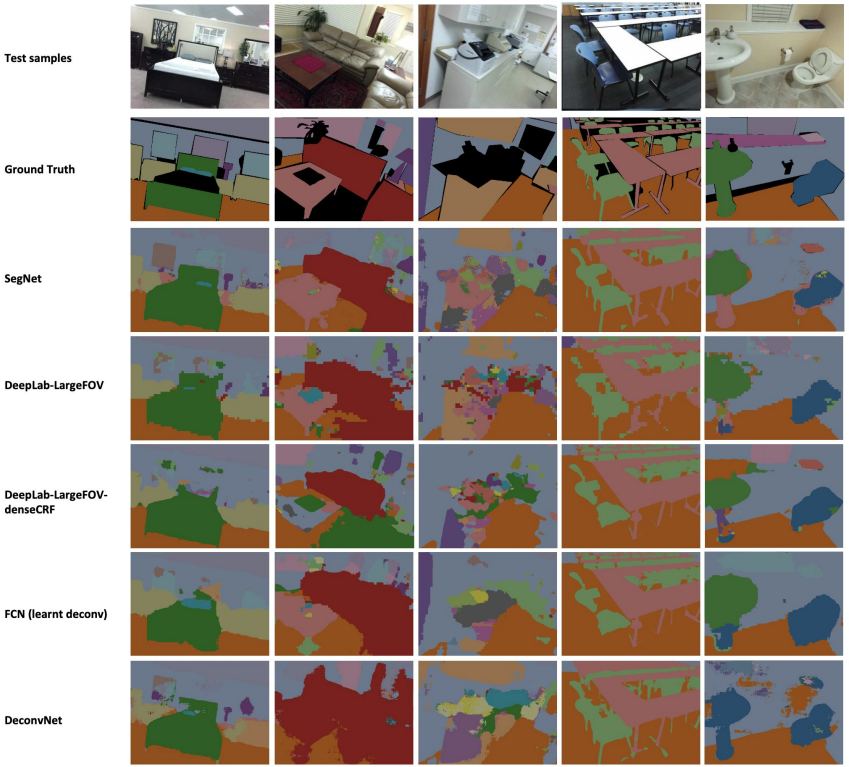


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

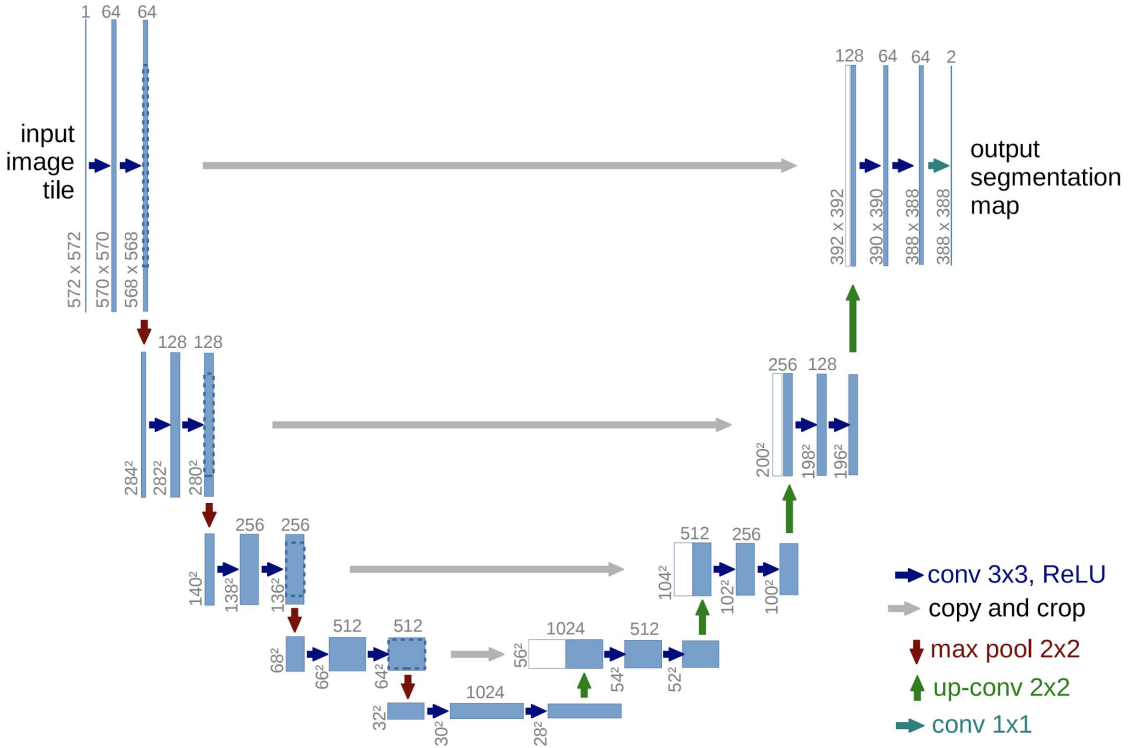
SegNet: Further development



U-Net

2015

https://arxiv.org/abs/1505.04597



U-Net

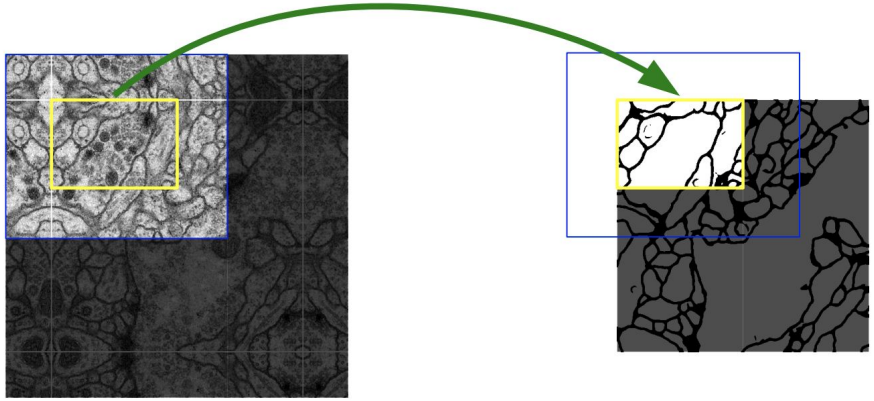
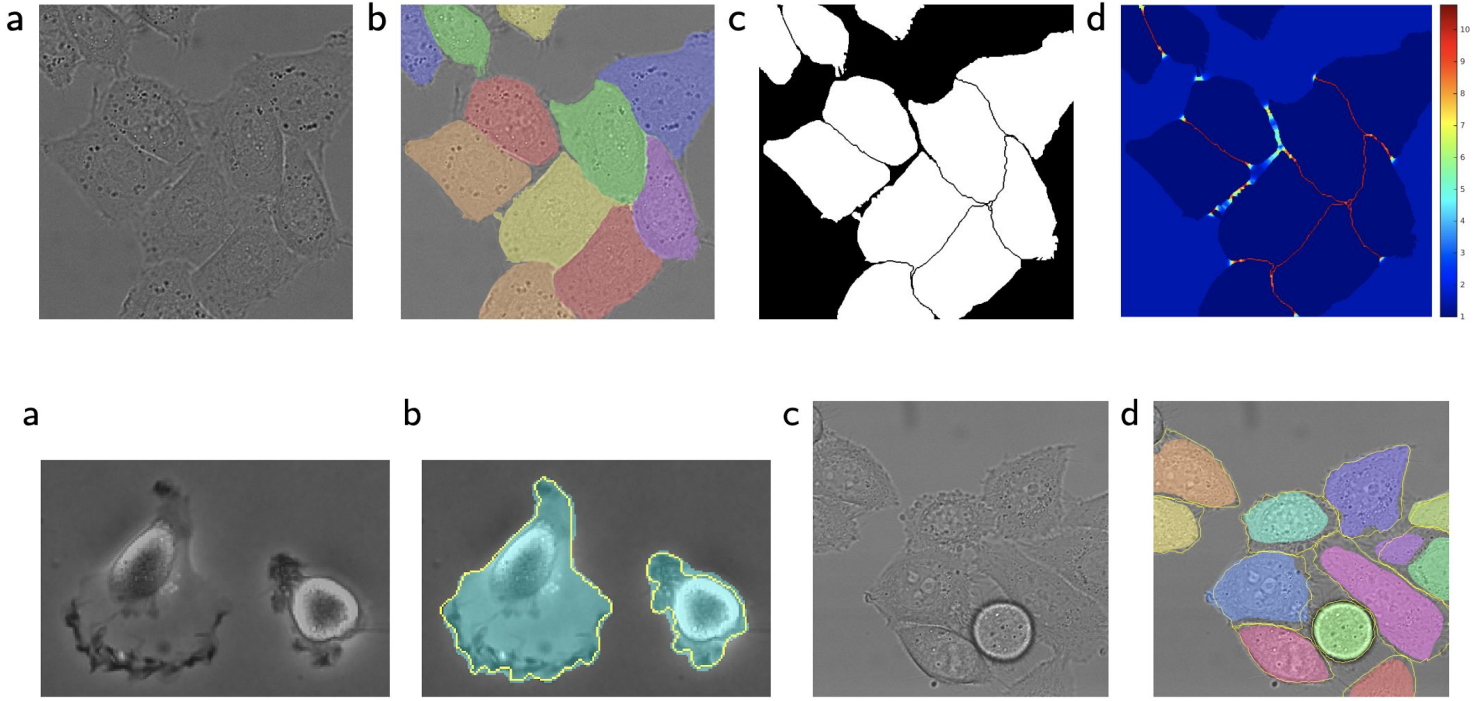


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

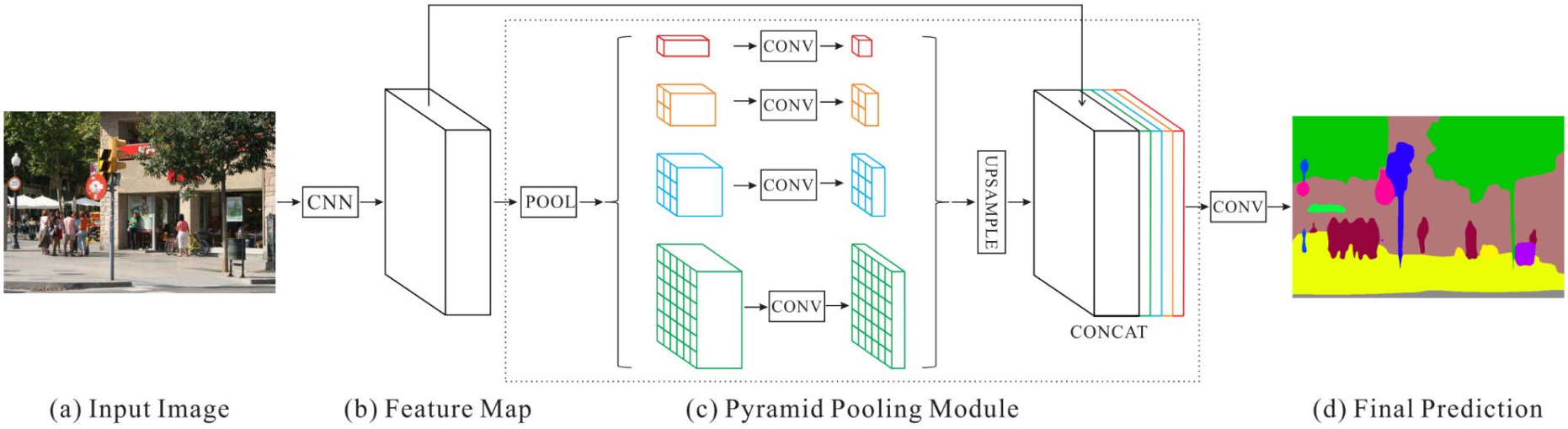
U-Net



PSPNet

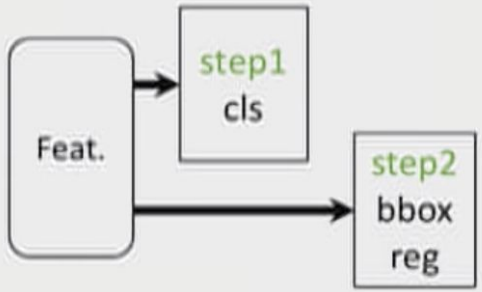
2017

https://arxiv.org/pdf/1612.01105.pdf

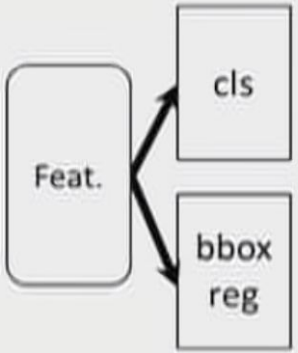


Mask R-CNN: Instance segmentation

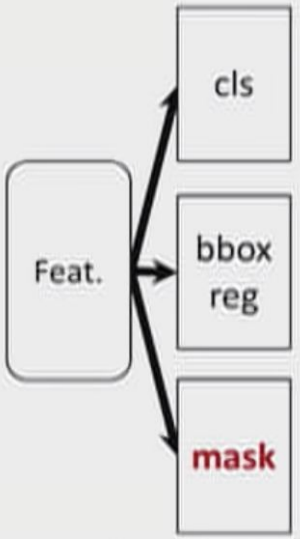
- Easy, fast to implement and use



(slow) R-CNN

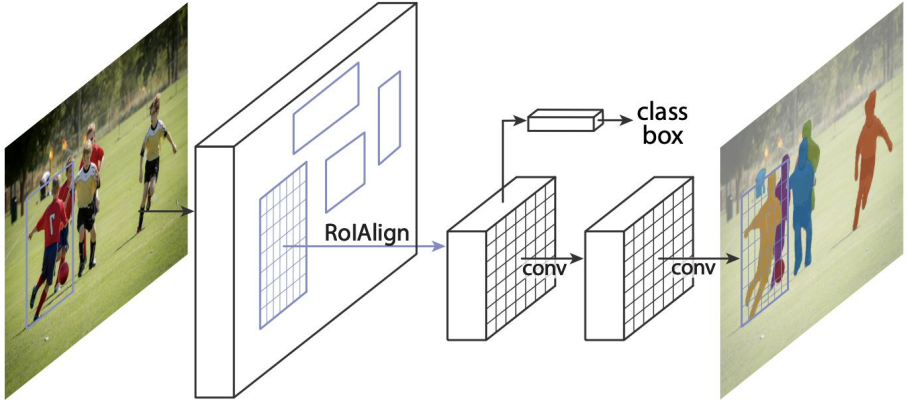


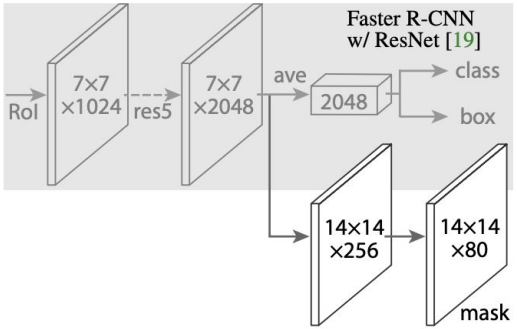
Fast/er R-CNN

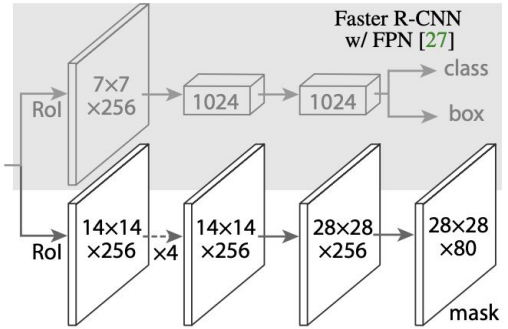


Mask R-CNN

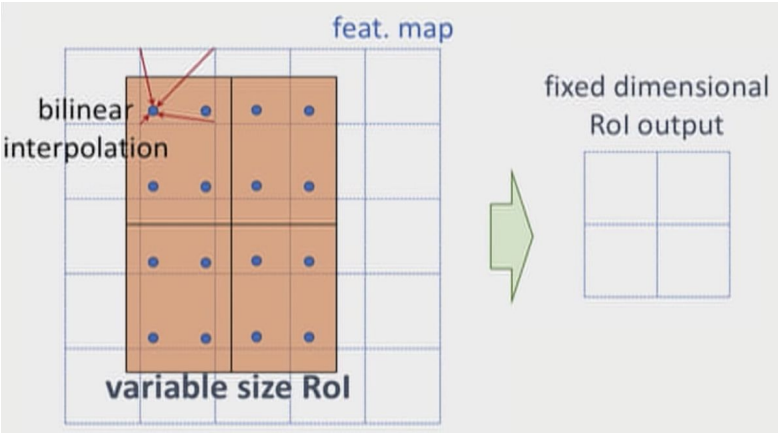
Mask R-CNN

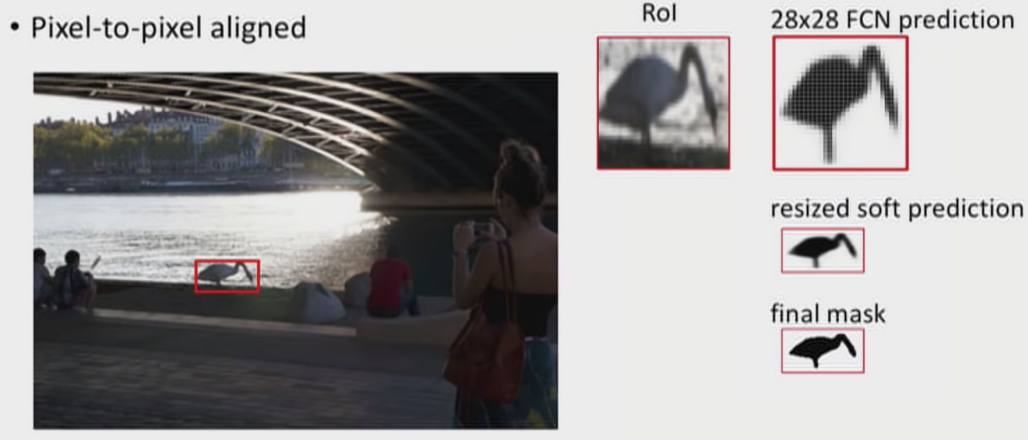

RoIAlign
class
box
conv
conv


Faster R-CNN
w/ ResNet [19]
RoI
7×7
×1024
res5
7×7
×2048
ave
2048
class
box
14×14
×256
14×14
×80
mask


Faster R-CNN
w/ FPN [27]
RoI
7×7
×256
1024
1024
class
box
RoI
14×14
×256
×4
14×14
×256
28×28
×256
28×28
×80
mask

Mask R-CNN: RoI Align



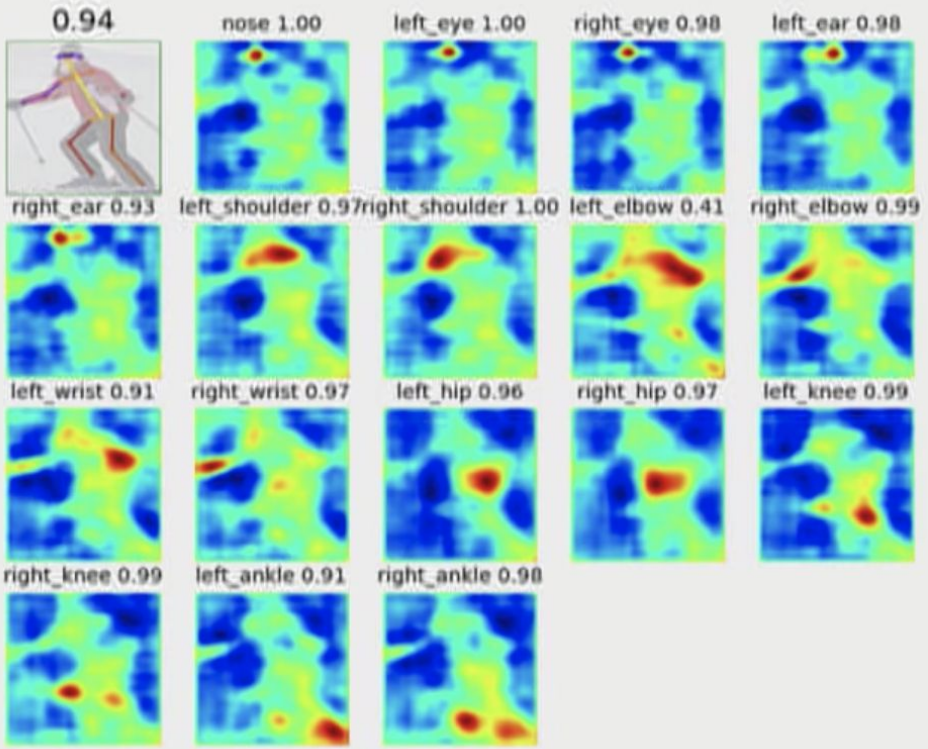


Mask R-CNN: Results

Bonus: Pose estimation

- keypoint = 1-hot mask
- human pose = 17 masks

- One framework for
  - ✓bbox
  - ✓mask
  - ✓keypoint

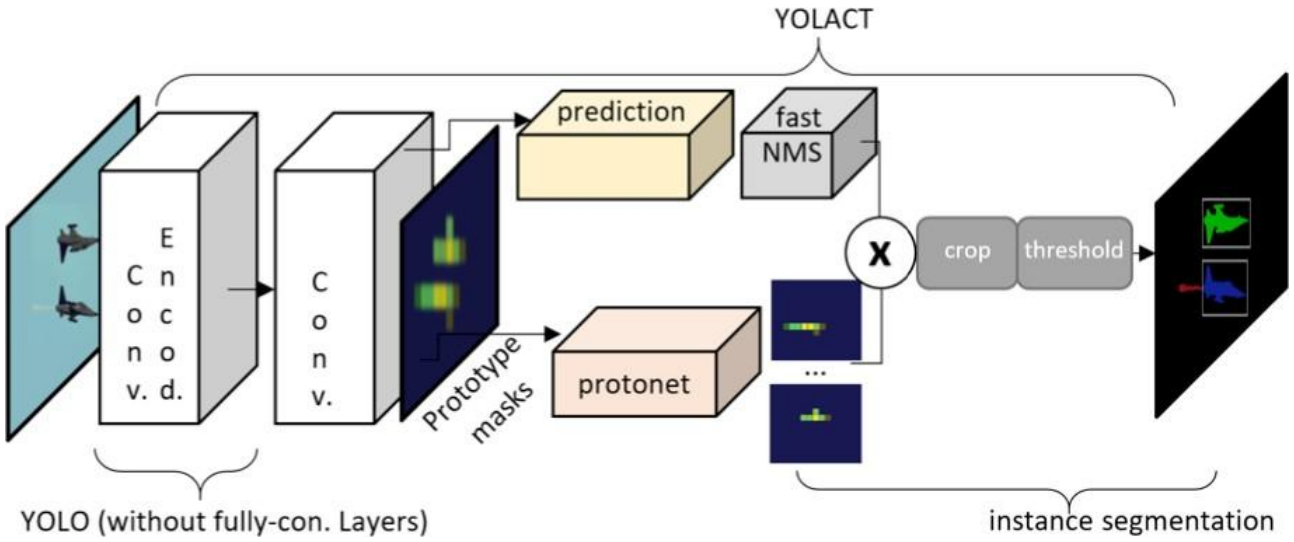


Bonus: Pose estimation

YOLACT

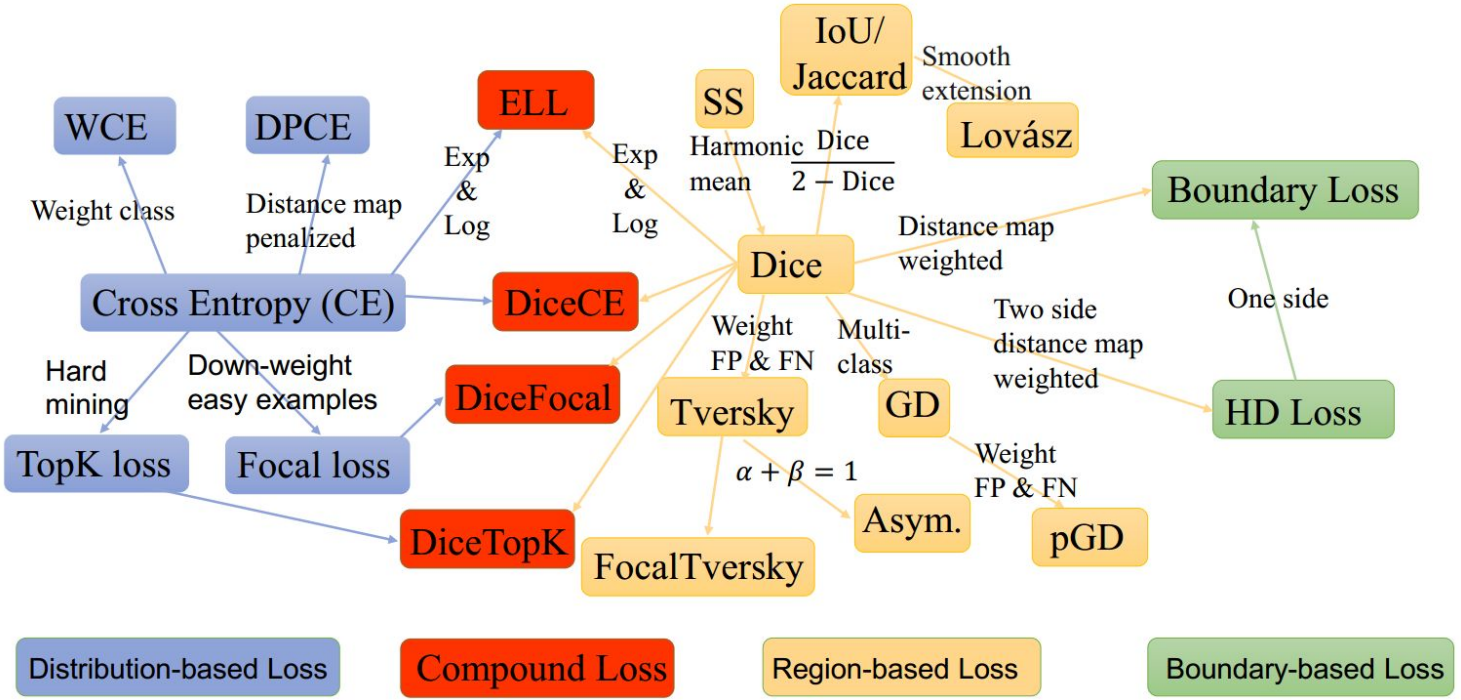
31.2% mAP on COCO dataset

~ 30 fps



Loss functions

Segmentation Loss Odyssey



https://arxiv.org/pdf/2005.13449.pdf

Distribution-based Losses

- Crossentropy
- Focal loss
- Top-k loss
- etc

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c log s_i^c$$

$$L_{FL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c}^{C}(1-s_i^c)^{\gamma} g_i^c log s_i^c$$

Region-based Losses

- Dice
- IoU
- Tversky loss
- Asymmetric similarity loss
- etc

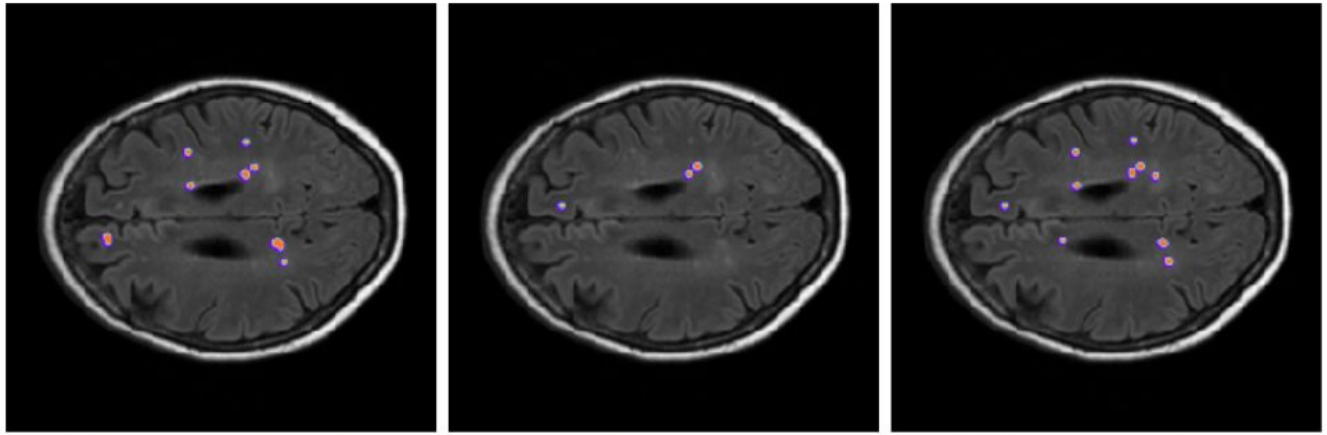
$$L_{Dice} = 1 - \frac{2\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c s_i^c}{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^{c2} + \sum_{i=1}^{N}\sum_{c=1}^{C} s_i^{c2}}$$

$$L_{IoU} = 1 - \frac{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c s_i^c}{\sum_{i=1}^{N}\sum_{c=1}^{C} (g_i^c + s_i^c - g_i^c s_i^c)}$$

Boundary-based Losses

- Boundary (BD) loss
- Hausdorff Distance (HD) loss

Used for pixelwise quality segmentation



(a) Ground truth (b) GDL (c) GDL w/ boundary loss

- https://arxiv.org/pdf/2006.14822.pdf
- https://github.com/JunMa11/SegLossOdyssey
  - https://github.com/shruti-jadon/Semantic-Segmentation-Loss-Functions
- https://www.sciencedirect.com/science/article/abs/pii/S1361841521000815
- https://github.com/LIVIAETS/boundary-loss
- https://arxiv.org/pdf/1812.07032.pdf
- Transposed convolution visualization
- https://www.cvat.ai/
- https://github.com/hustvl/YOLOP/tree/main
-

Revise

- Segmentation task
- Simple solutions
- Upsampling methods
  - Unpooling
  - Transposed convolution
- FCN
- DeconvNet
- SegNet
- U-Net
- Mask R-CNN

Next time

- Variational AutoEncoders
- GANs

Interactive playground for image generation