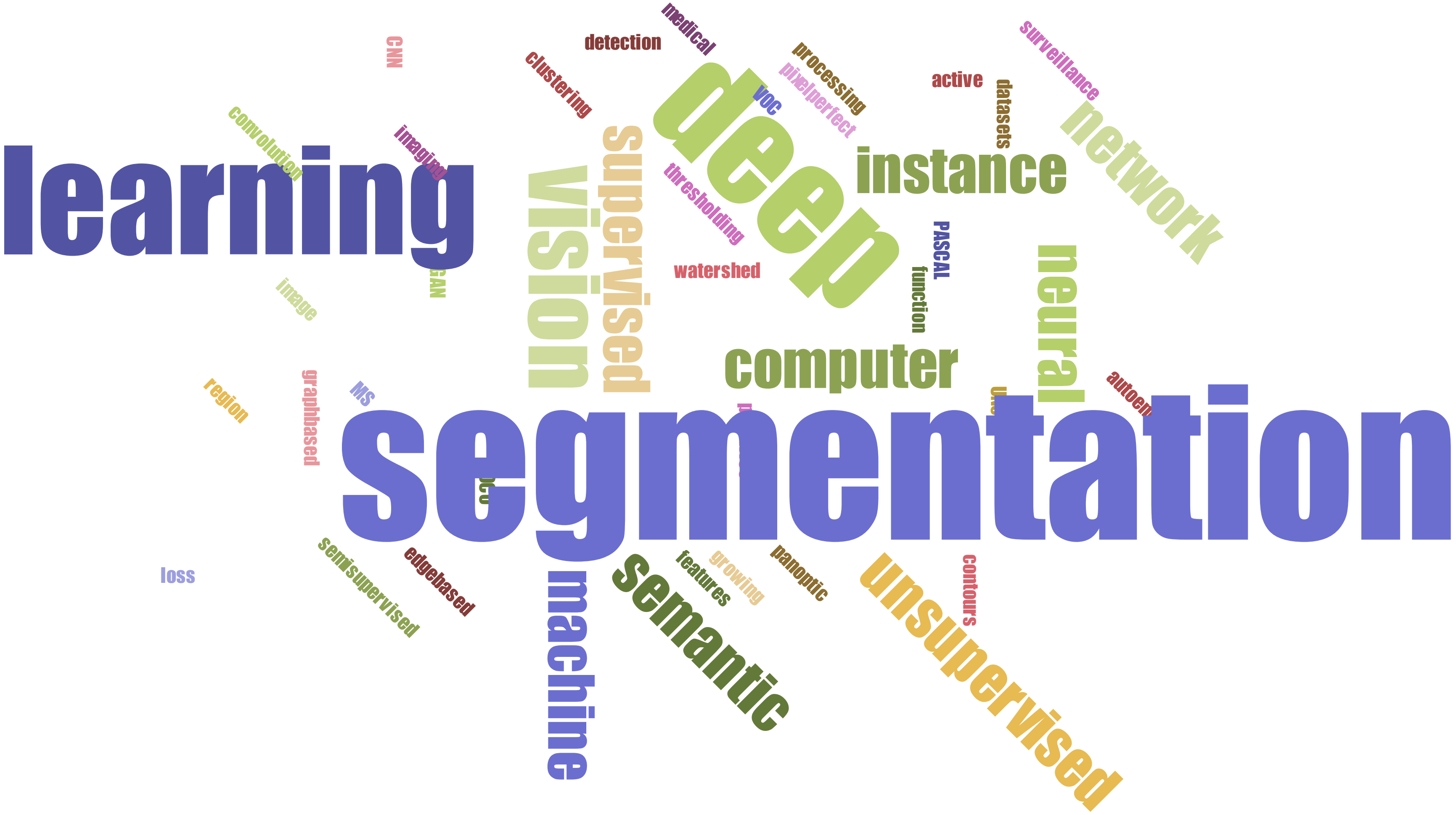
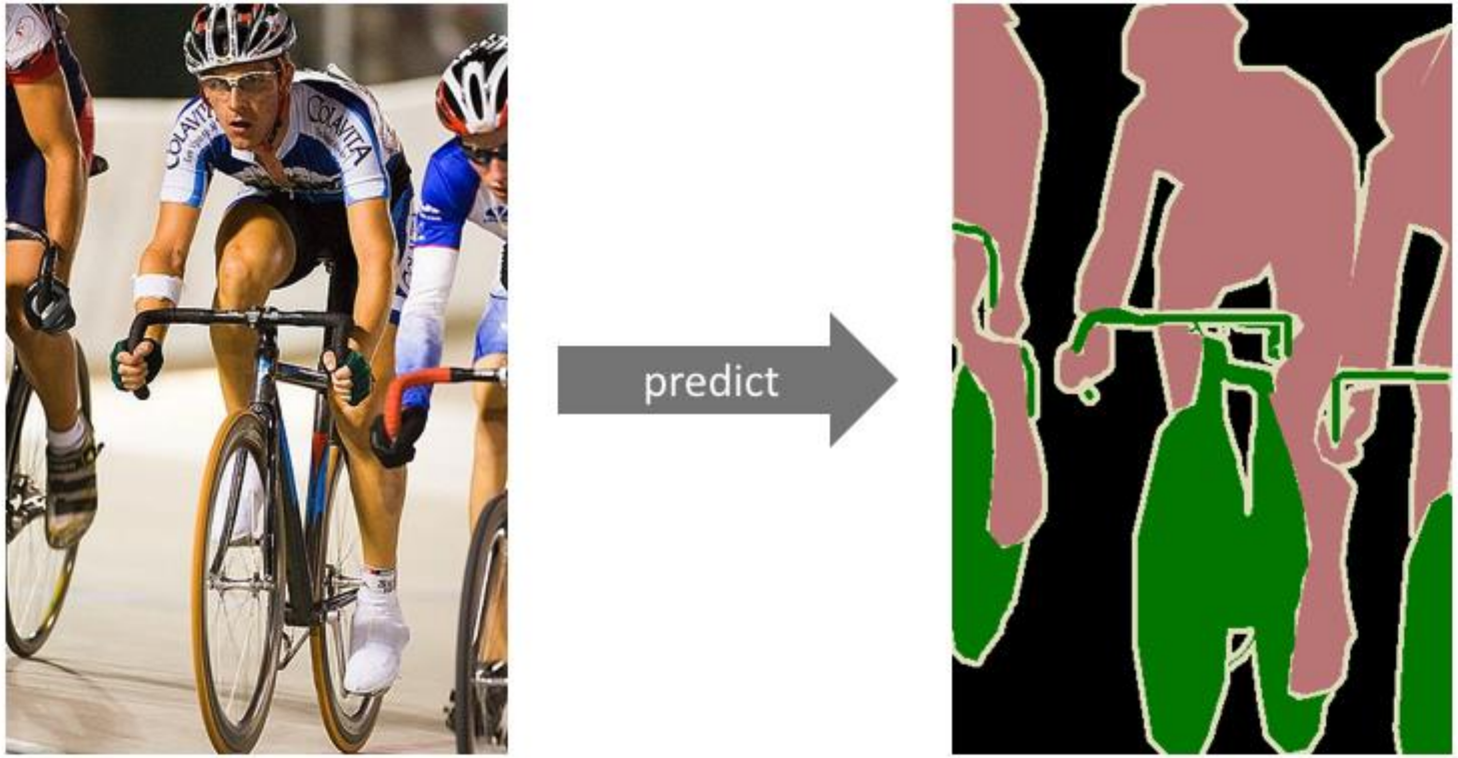

segmentation
learning
deep
vision
supervised
instance
network
neural
computer
machine
semantic
unsupervised
convolution
imaging
CNN
clustering
detection
medical
processing
pixelperfect
VOC
active
datasets
surveillance
thresholding
watershed
PASCAL
function
image
GAN
region
graphbased
MS
loss
semisupervised
edgebased
features
growing
panoptic
contours

Workshop Info

https://ikosaeder.cosy.sbg.ac.at/dihworkshop/#segmentation/

What is segmentation?

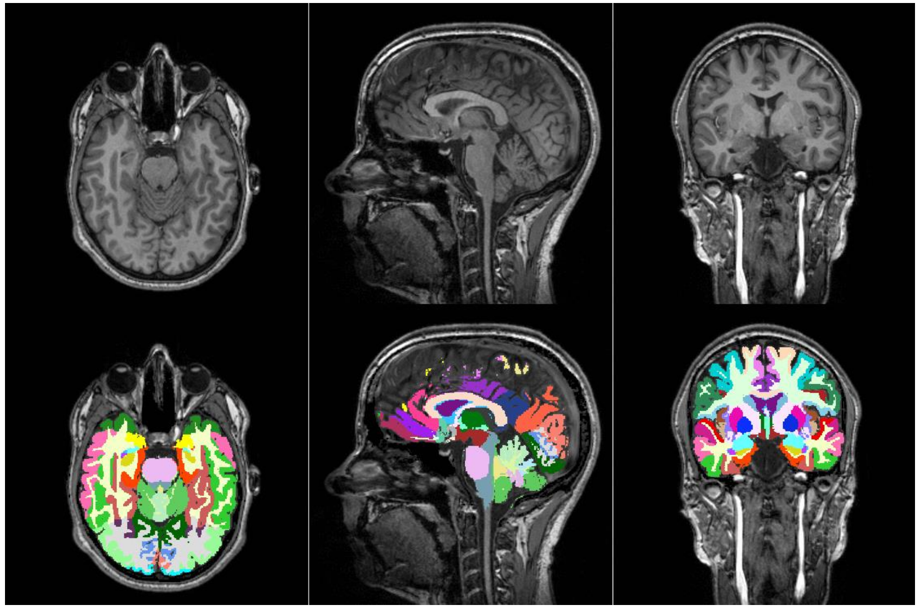


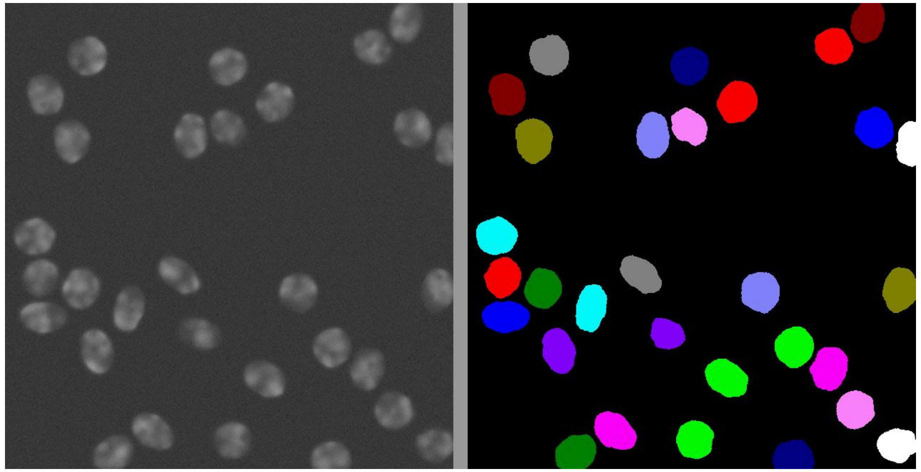
Person
Bicycle
Background

Image from [1]

Applications

- **Medical & Biomedical Imaging**
  - Locating tumors and other pathologies
  - Diagnosis and study of anatomical structure
  - Cell segmentation





Images from [11, 12]

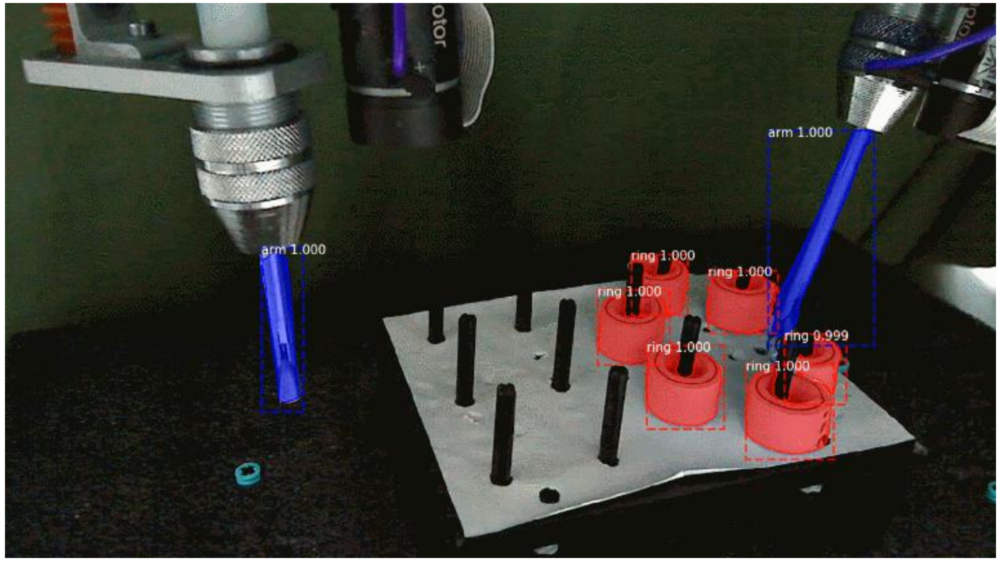
Applications

- **Autonomous vehicle and transportation**
  - Street scene segmentation

Image from [13]

Applications

- **Remote Satellite Sensing**
- **Medical Health Care / Industrial**

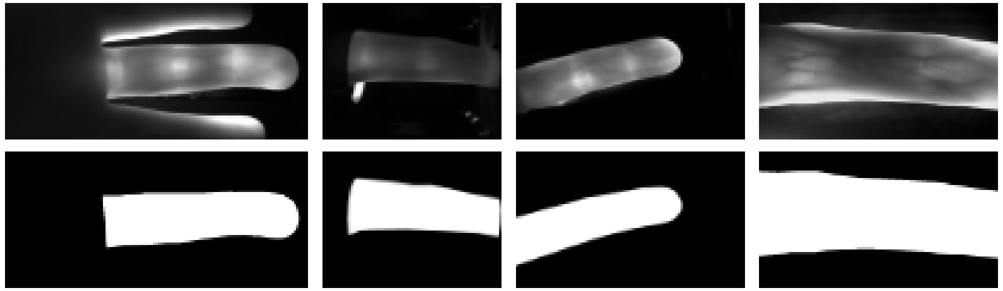


Images from [14, 15]

Applications

- **Biometrics**
  - Iris Off-angle Segmentation
  - Finger segmentation for vein recognition

(a) (b) (c)



Images from [16, 17]

Types of Segmentation

Semantic segmentation
- Assigning category label to each pixel
- Grass, sky, road

Instance segmentation
- Detect each object and delineate it
- Car, person, chair

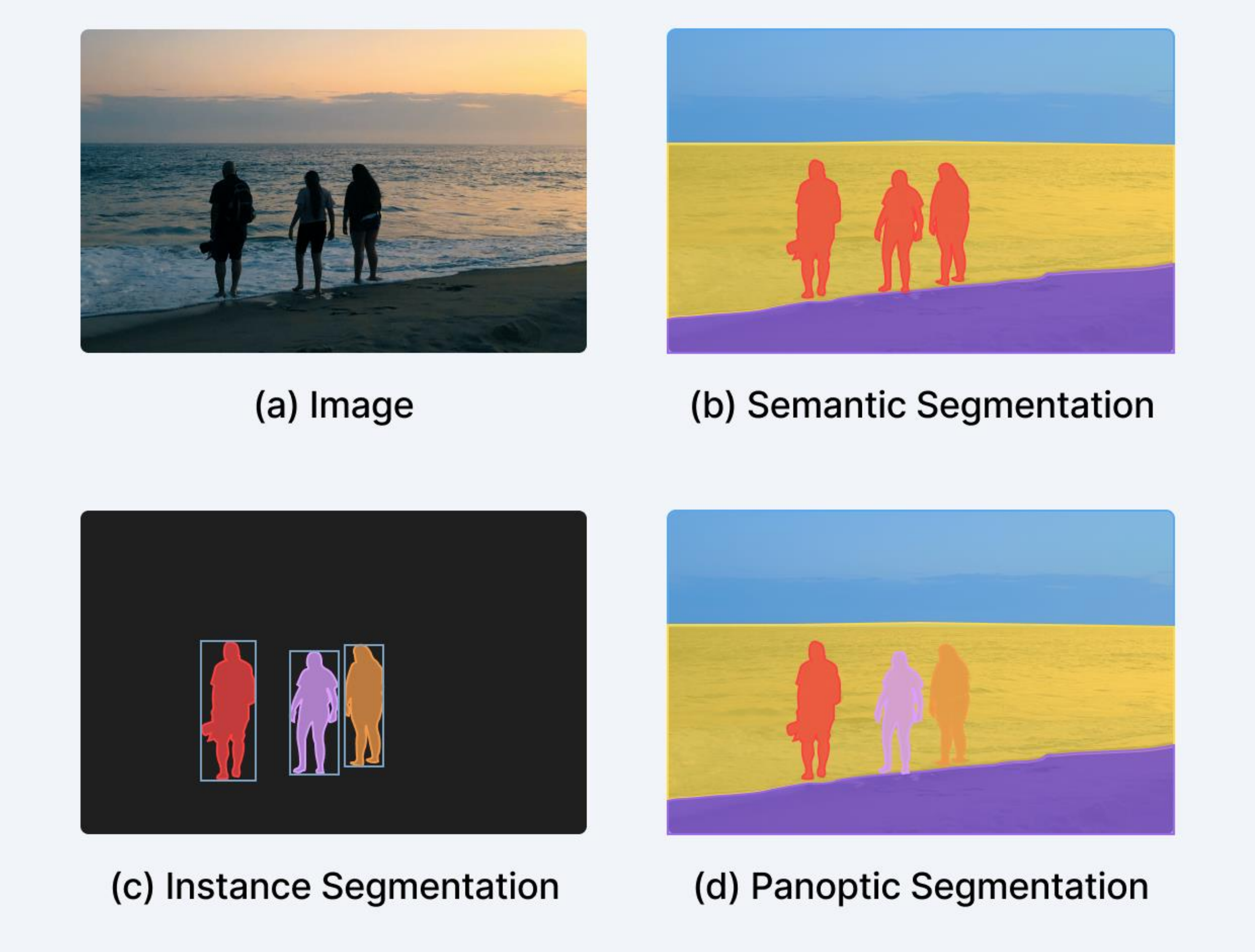


Image from [18]

„Handcrafted“ Methods for Segmentation

- Threshold-Based segmentation
- Edge-Based segmentation
- Region-Based segmentation
- Graph-Based techniques
- Clustering-Based techniques
- Watershed techniques
- Active Contour techniques

- *Deep-Learning-Based (Convolutional Neural Network, CNN) Segmentation*

We remember: CNN Basics

- Input Image
- Convolution
- Pooling
- Ground Truth
- Loss Function

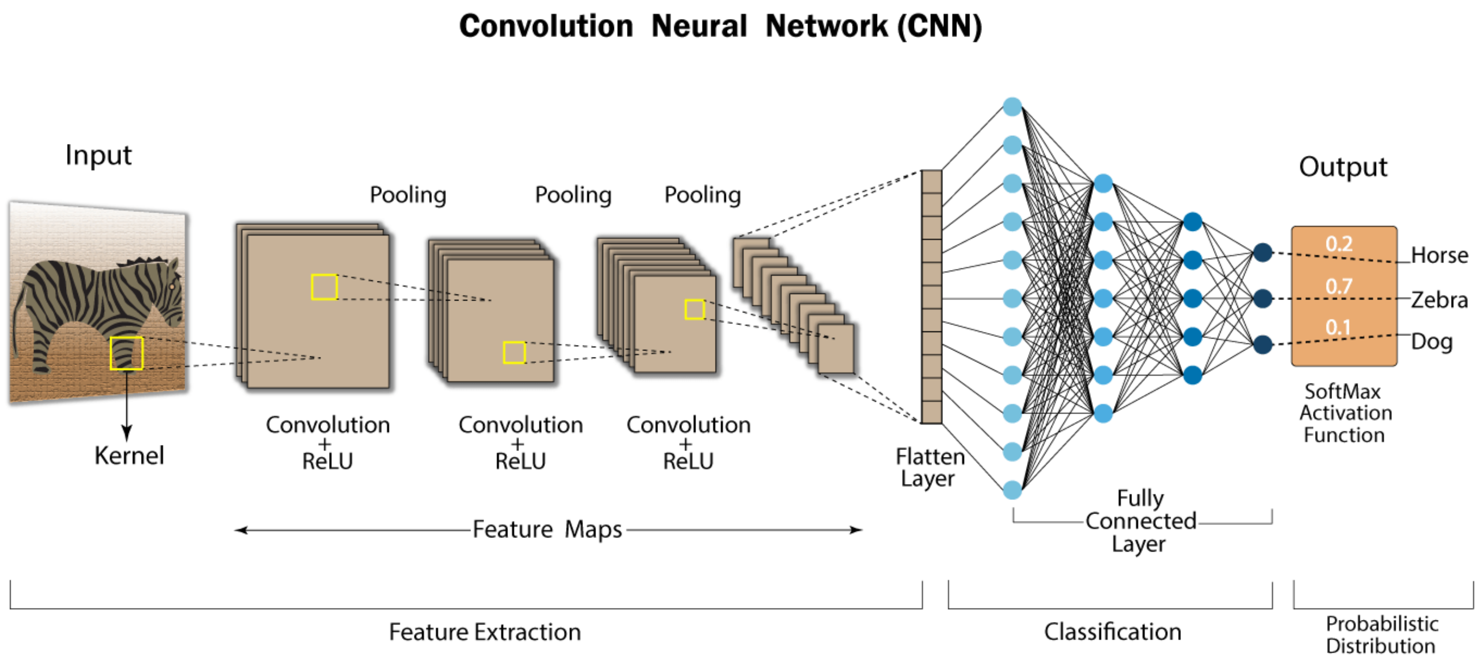


Image from [4]

Representing Input & Segmentation Map

RGB = height x width x 3

Grey = height x width x 1

Seg. Map = height x width x 1

or height x width x #classes

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

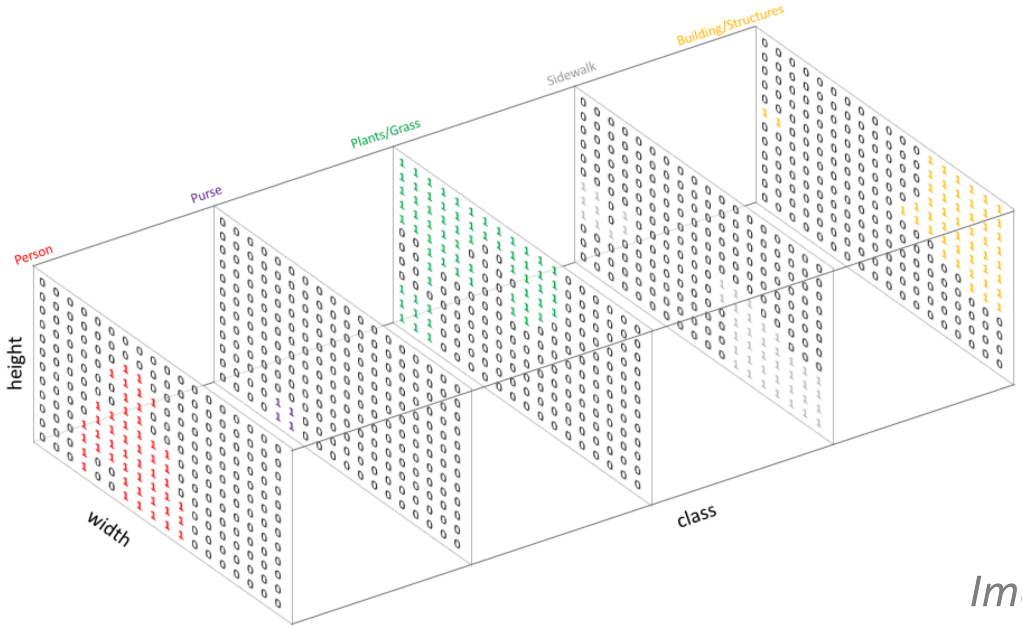
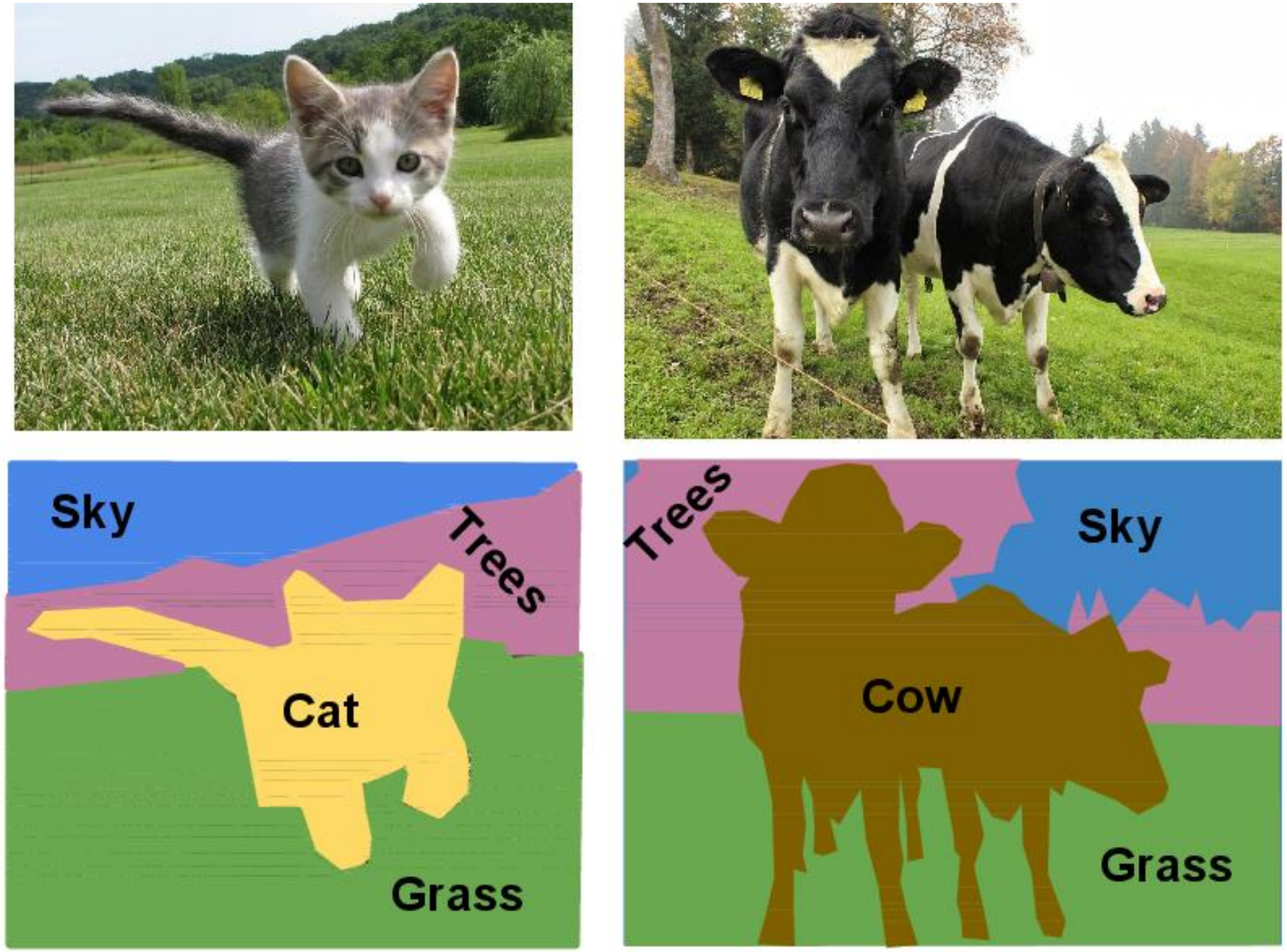


Image from [1]

Semantic Segmentation: The Task

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



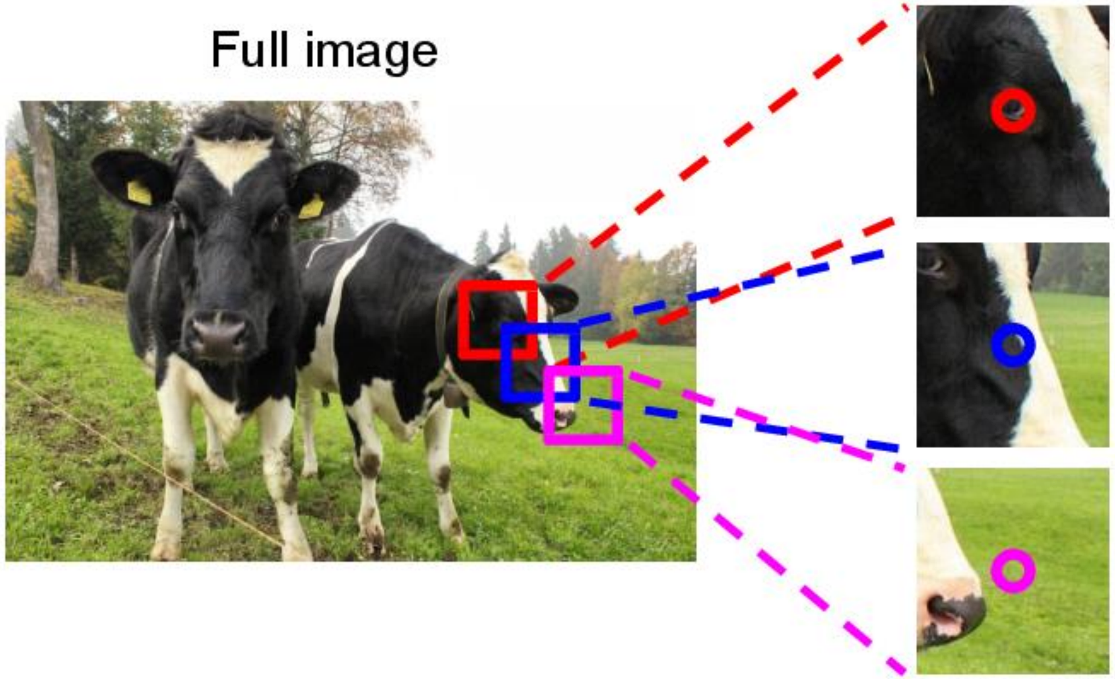
Slide from [2]

Semantic Segmentation

Full image

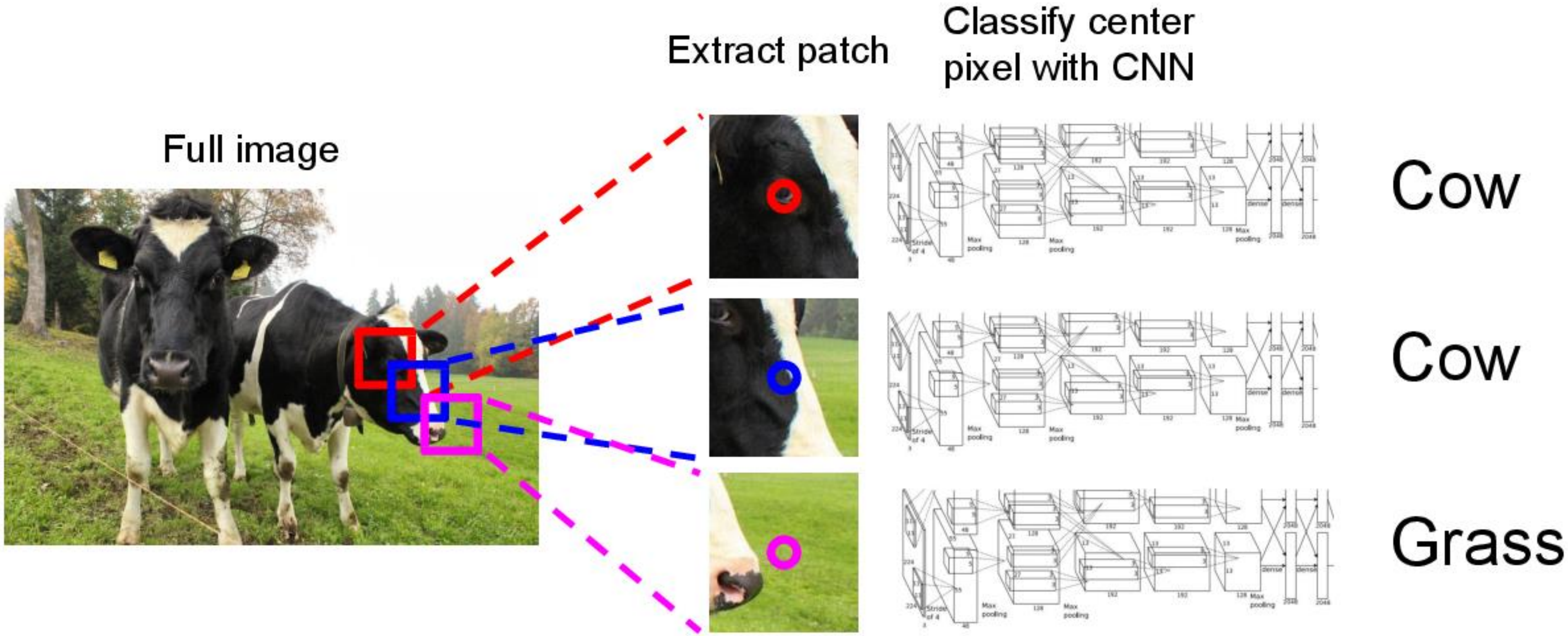
Impossible to classify without context

Semantic Segmentation Idea: Sliding Window



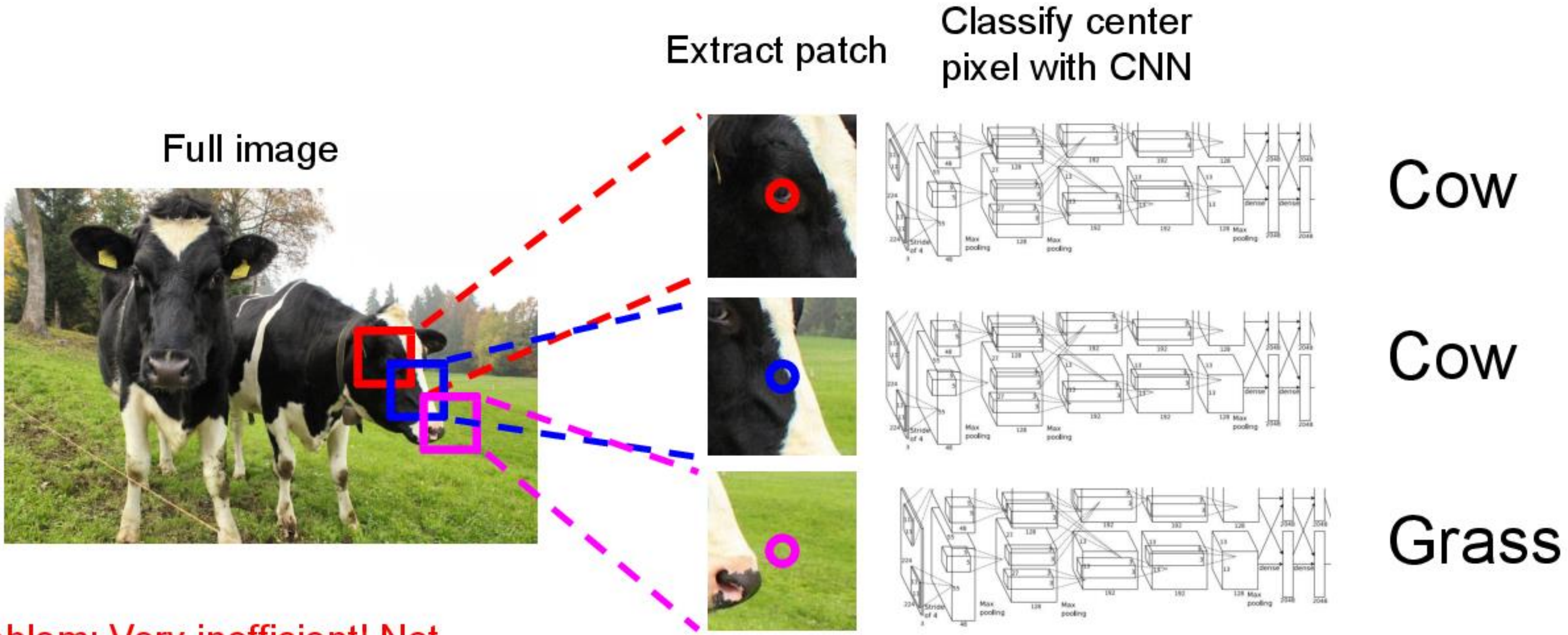
Slide from [2]

Semantic Segmentation Idea: Sliding Window



Slide from [2]

Semantic Segmentation Idea: Sliding Window

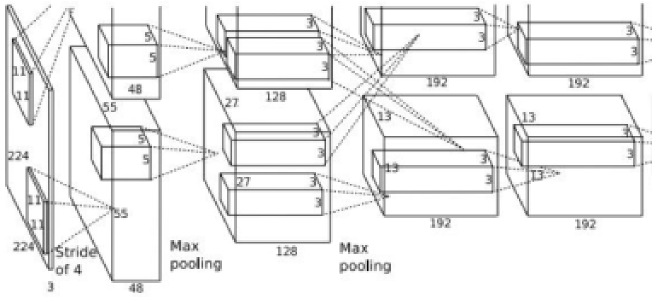


Problem: Very inefficient! Not reusing shared features between overlapping patches

Slide from [2]

Semantic Segmentation Idea: Convolution

Full image

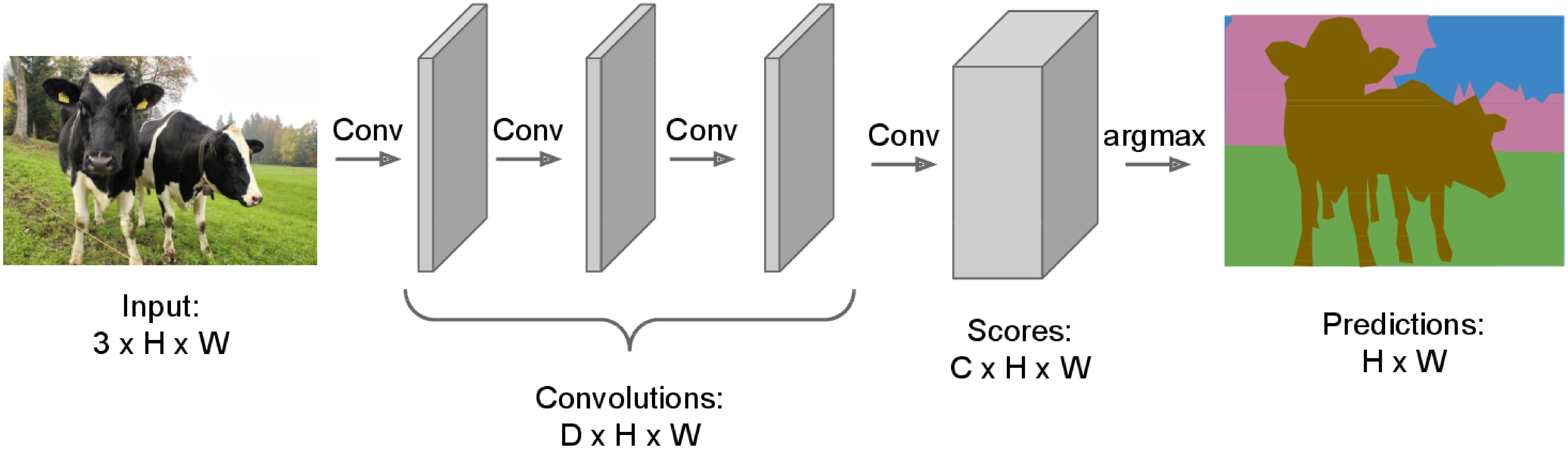


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

Semantic Segmentation Idea: Fully Convolutional

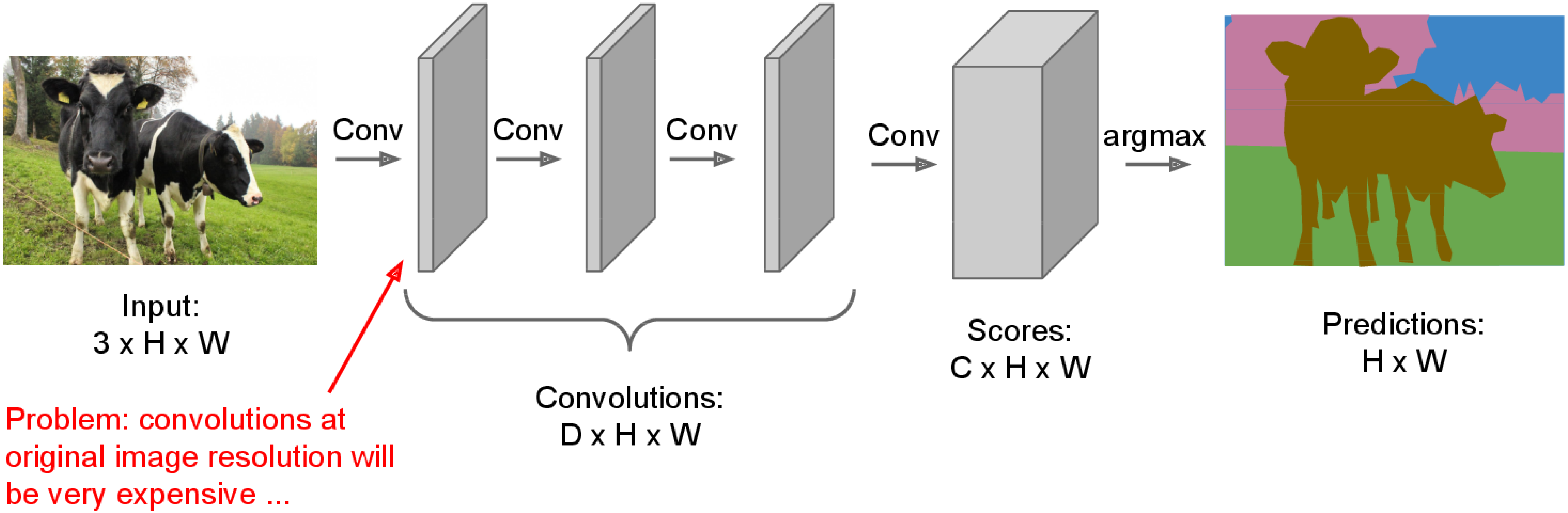
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Slide from [2]

Semantic Segmentation Idea: Fully Convolutional

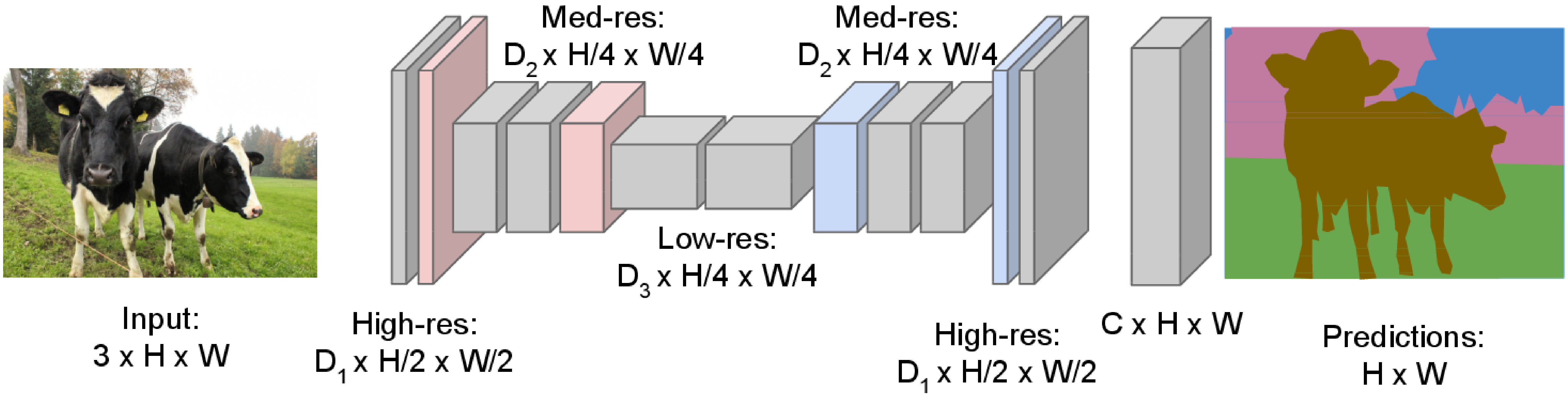
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Slide from [2]

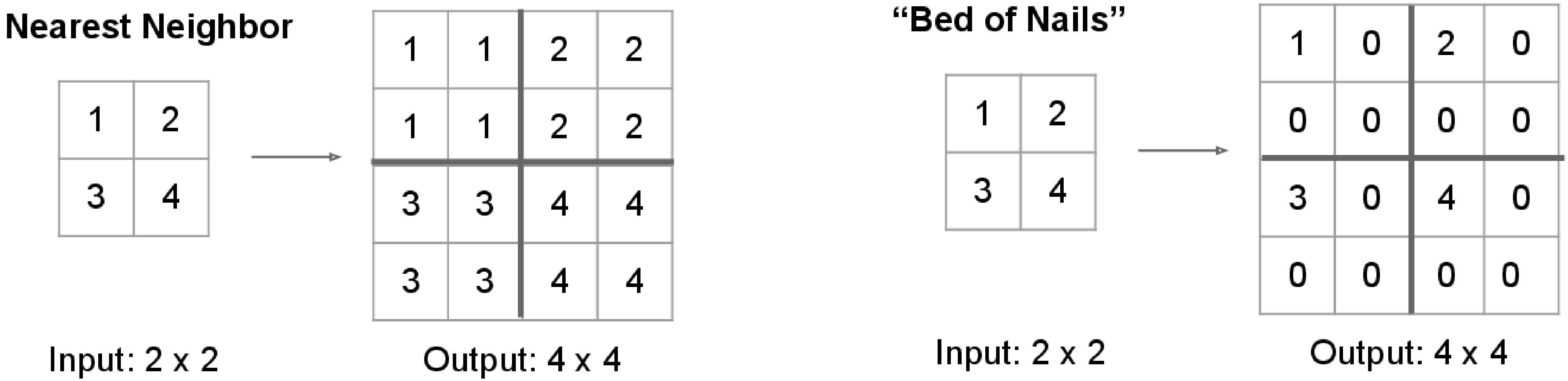
Semantic Segmentation Idea: Fully Convolutional

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



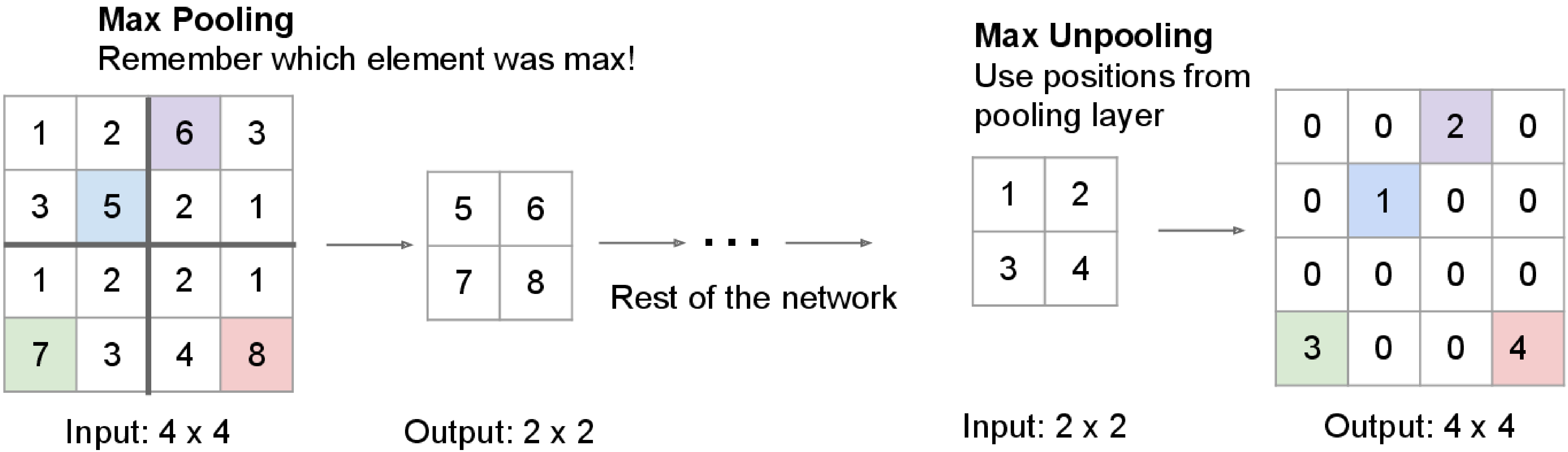
Slide from [2]

In-Network upsampling: “Max Unpooling”

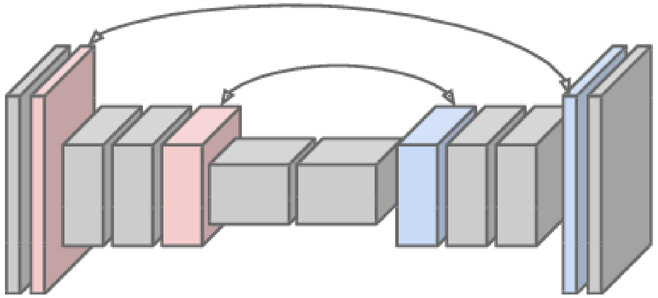


Slide from [2]

In-Network upsampling: “Max Unpooling”

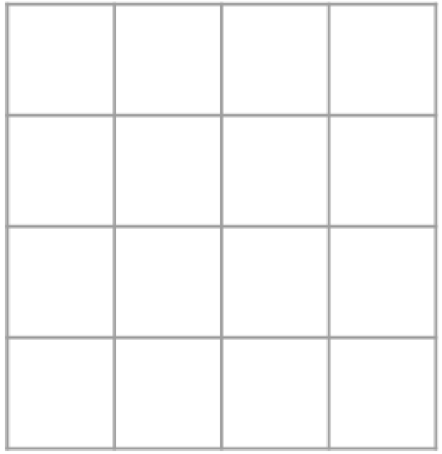


Corresponding pairs of downsampling and upsampling layers

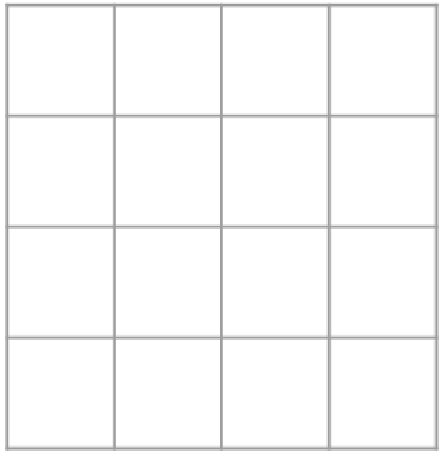


Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



Input: 4 x 4

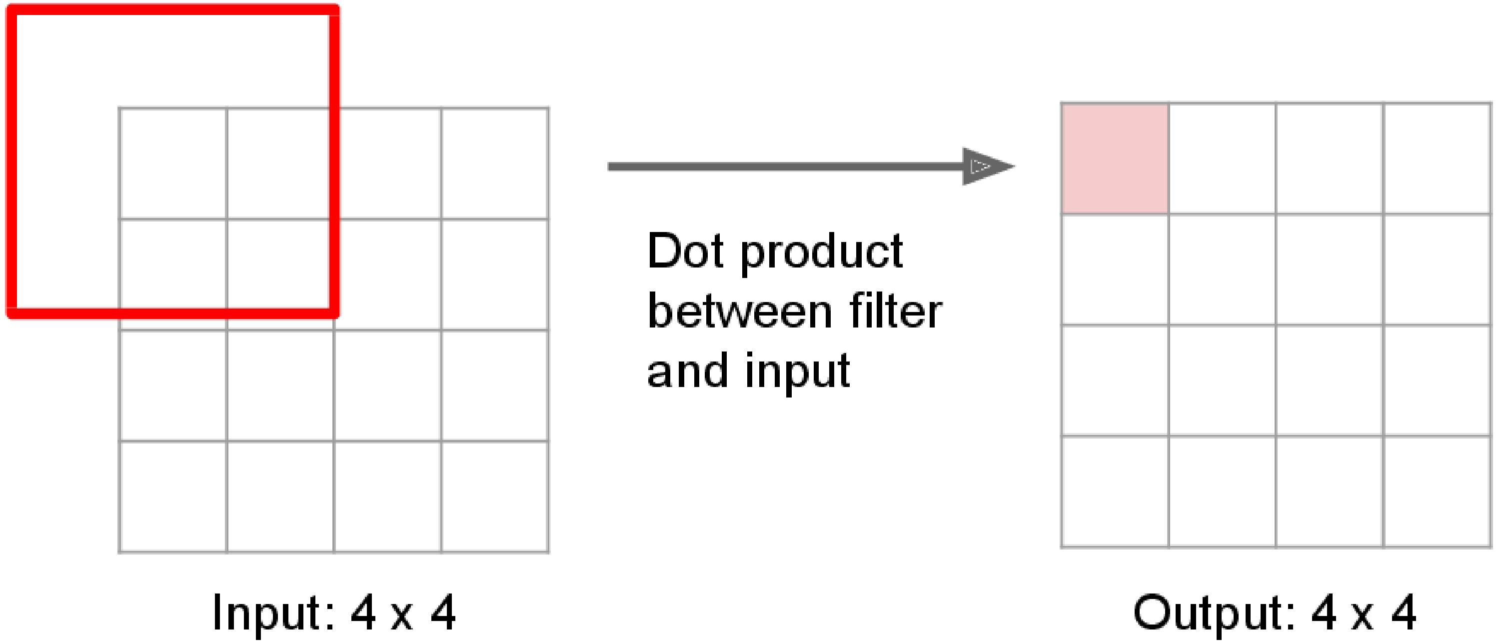


Output: 4 x 4

Slide from [2]

Learnable Upsampling: Transposed Convolution

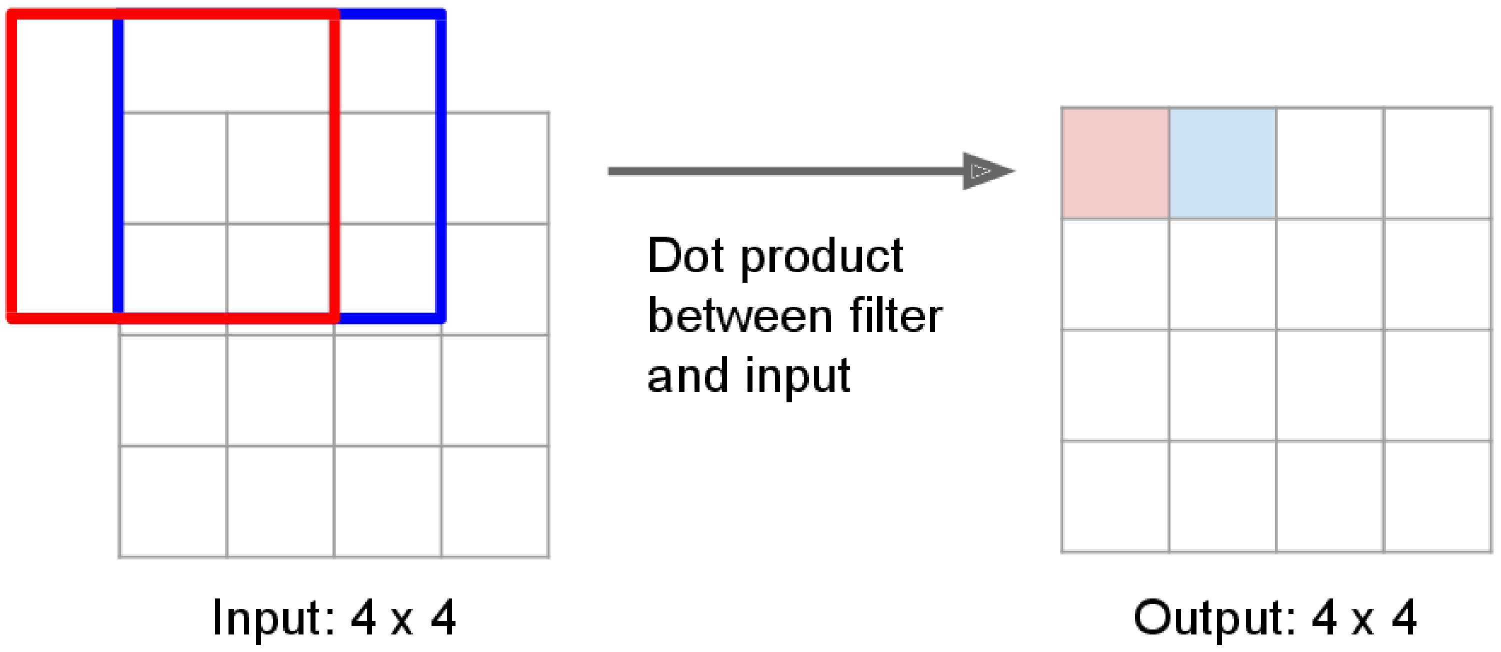
Recall: Normal 3 x 3 convolution, stride 1 pad 1



Slide from [2]

Learnable Upsampling: Transposed Convolution

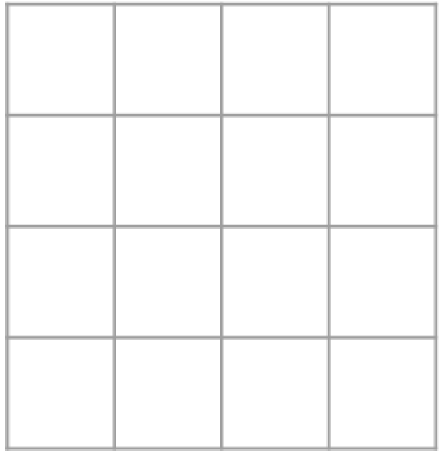
Recall: Normal 3 x 3 convolution, stride 1 pad 1



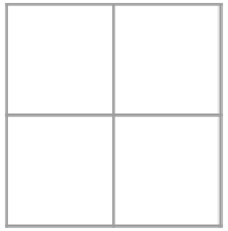
Slide from [2]

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

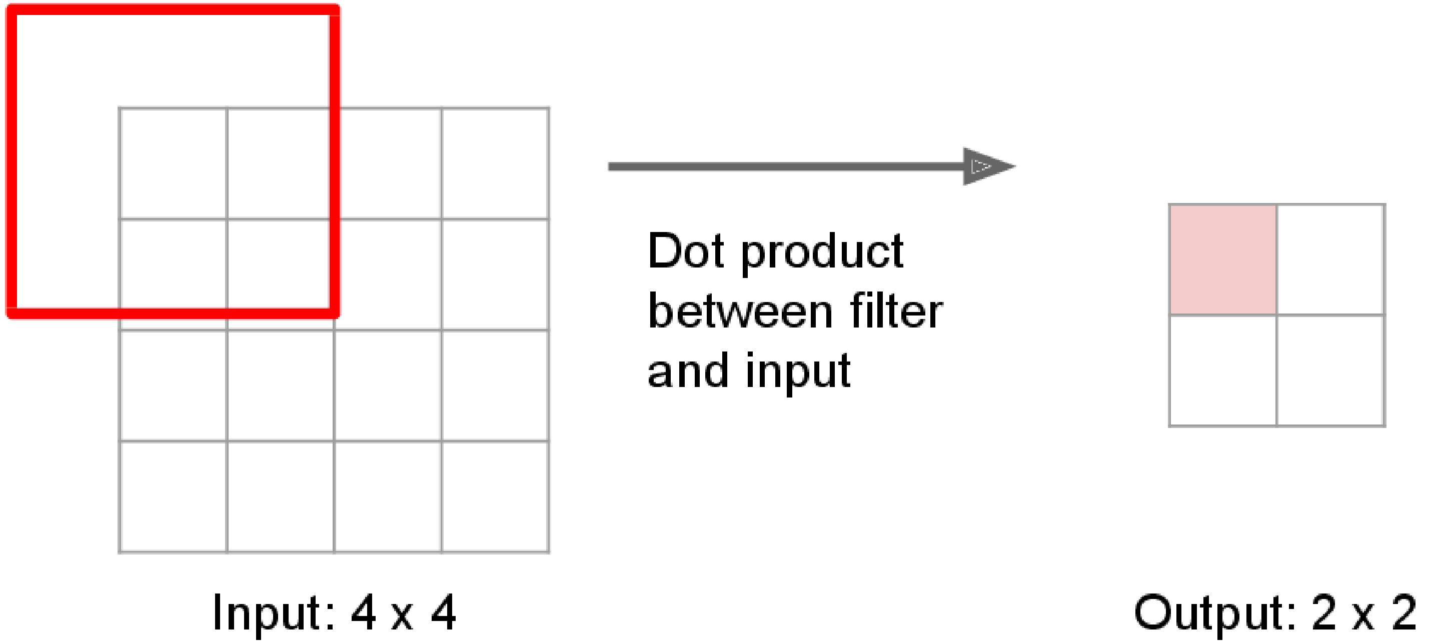


Output: 2 x 2

Slide from [2]

Learnable Upsampling: Transposed Convolution

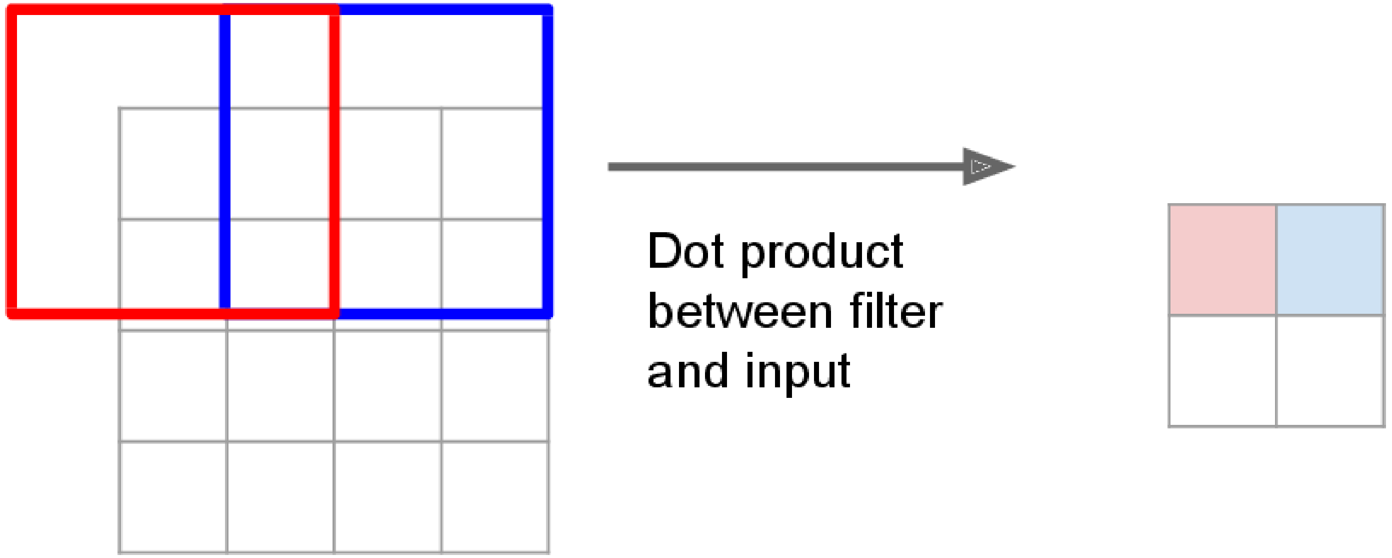
Recall: Normal 3 x 3 convolution, stride 2 pad 1



Slide from [2]

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, <u>stride 2</u> pad 1



Dot product between filter and input

Input: 4 x 4

Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

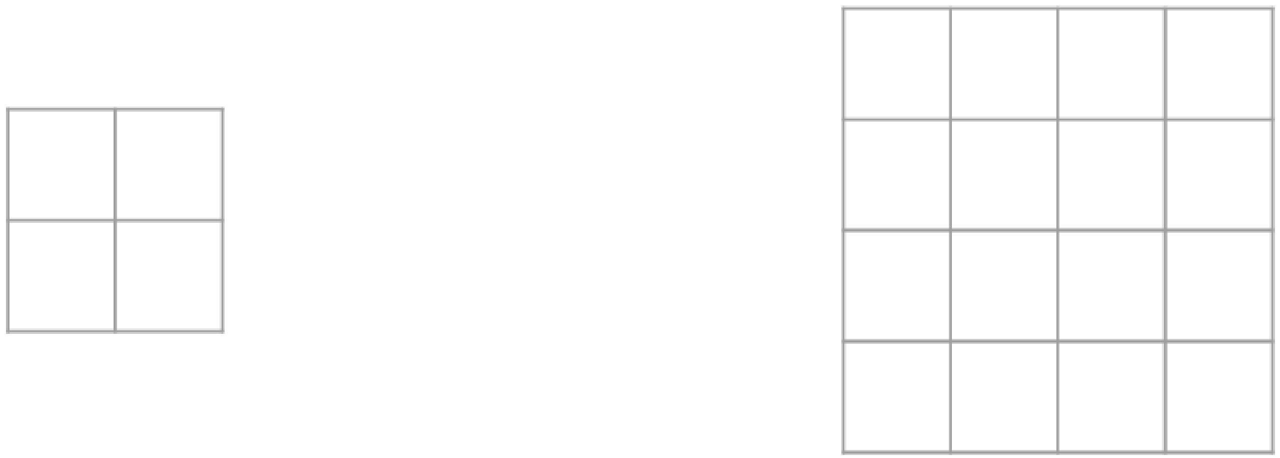
Stride gives ratio between movement in input and output

We can interpret strided convolution as “learnable downsampling”.

Slide from [2]

Learnable Upsampling: Transposed Convolution

3 x 3 **transposed** convolution, stride 2 pad 1



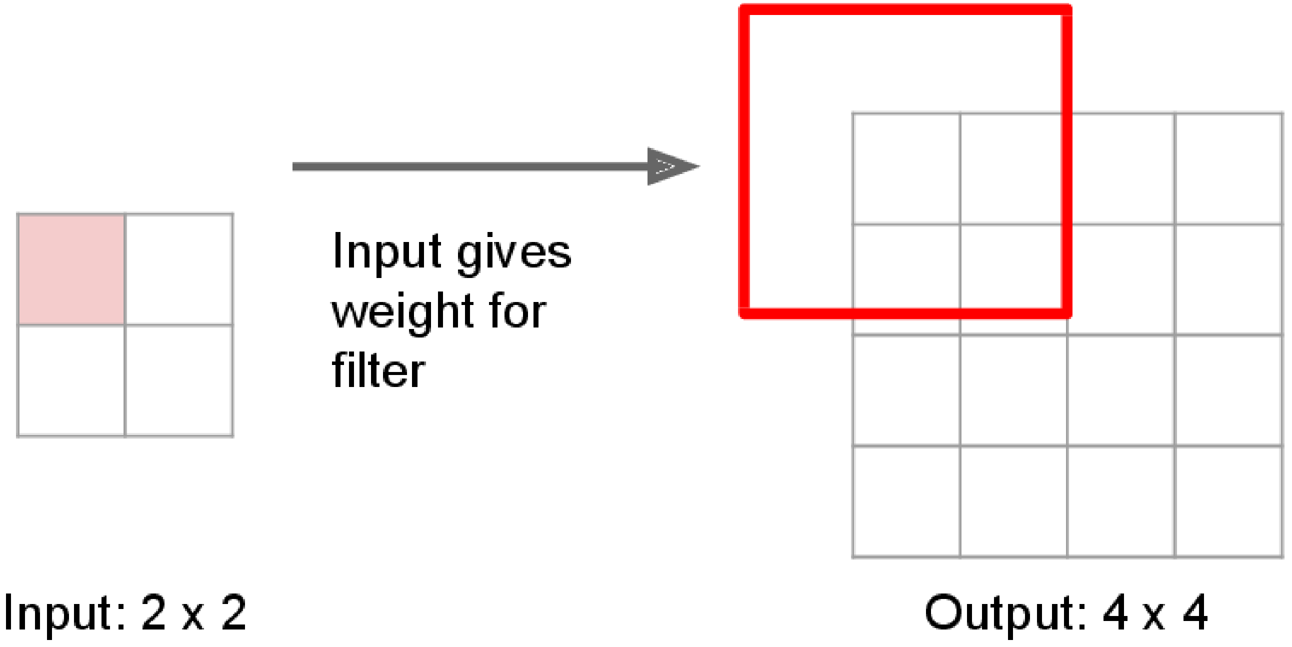
Input: 2 x 2

Output: 4 x 4

Slide from [2]

Learnable Upsampling: Transposed Convolution

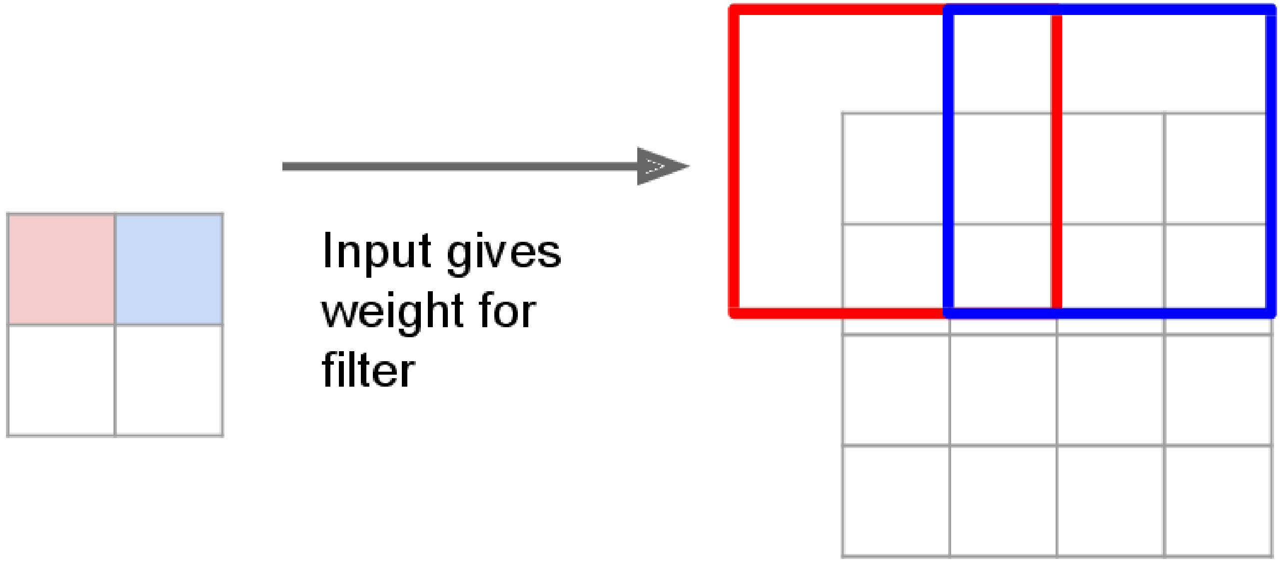
3 x 3 **transposed** convolution, stride 2 pad 1



Slide from [2]

Learnable Upsampling: Transposed Convolution

3 x 3 **transposed** convolution, stride 2 pad 1



Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

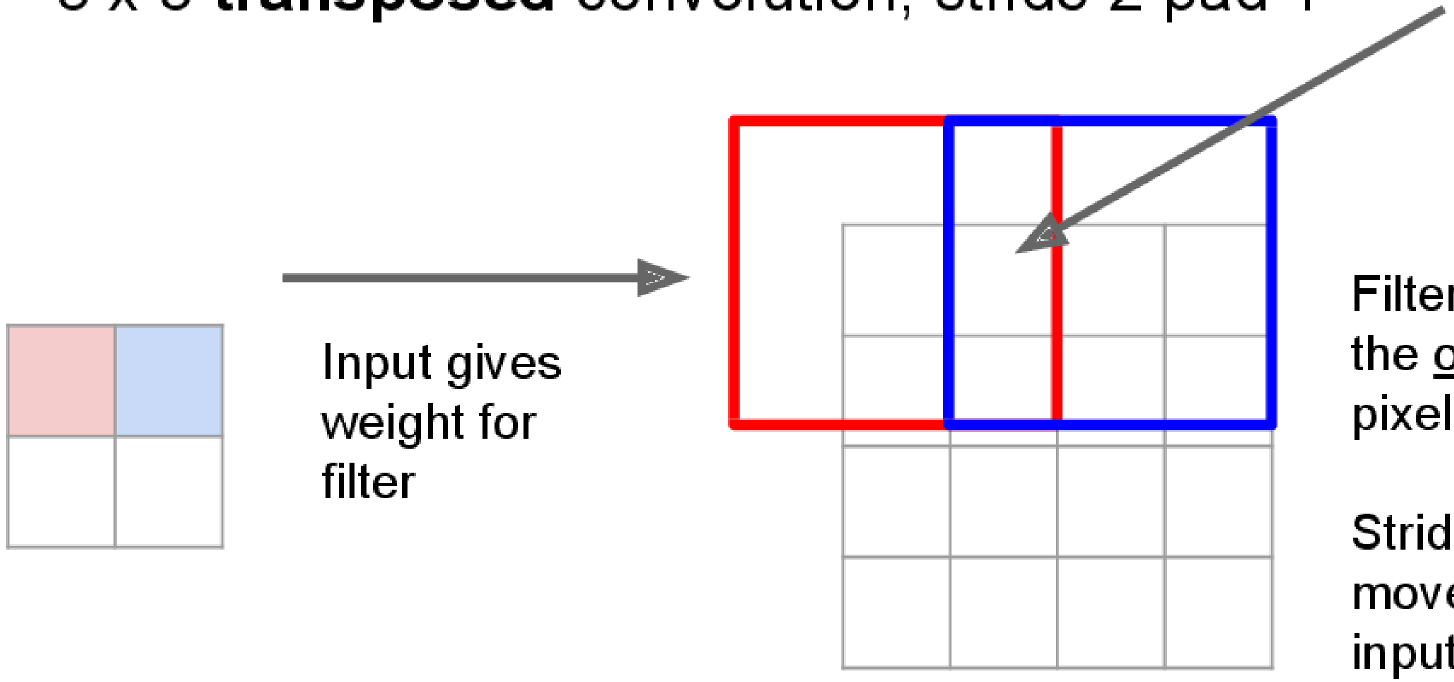
Stride gives ratio between movement in output and input

Slide from [2]

Learnable Upsampling: Transposed Convolution

3 x 3 **transposed** convolution, stride 2 pad 1

Sum where output overlaps



Input: 2 x 2

Output: 4 x 4

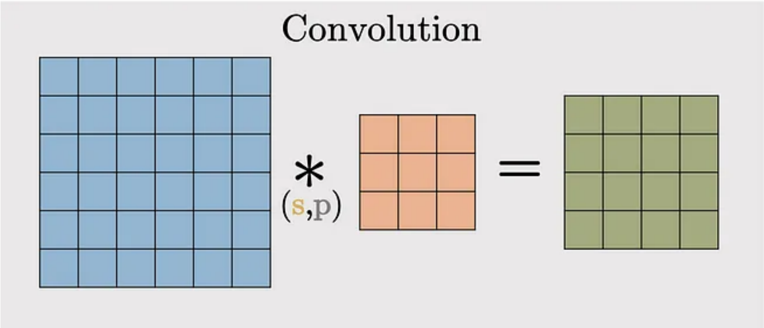
Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

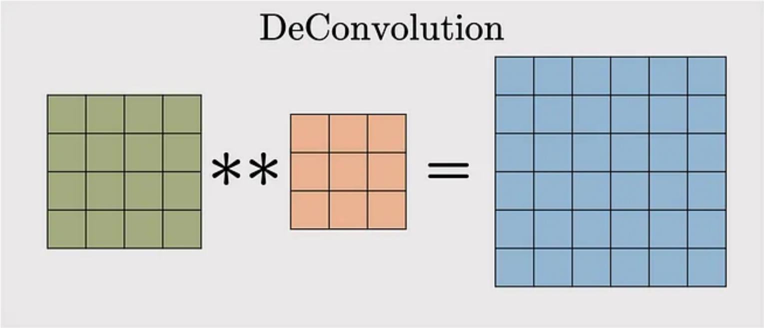
Stride gives ratio between movement in output and input

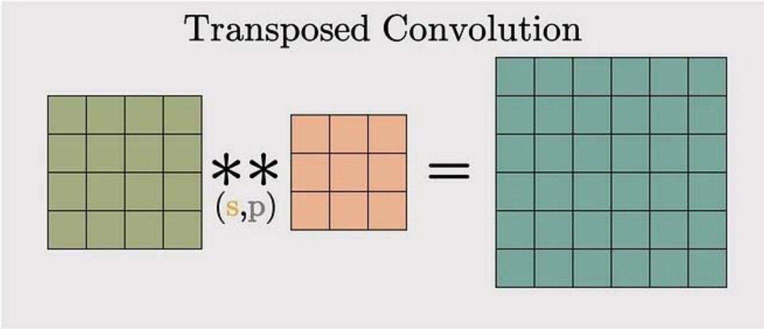
Slide from [2]

Learnable Upsampling: Transposed Convolution

- Often denoted „deconvolution“ (bad)





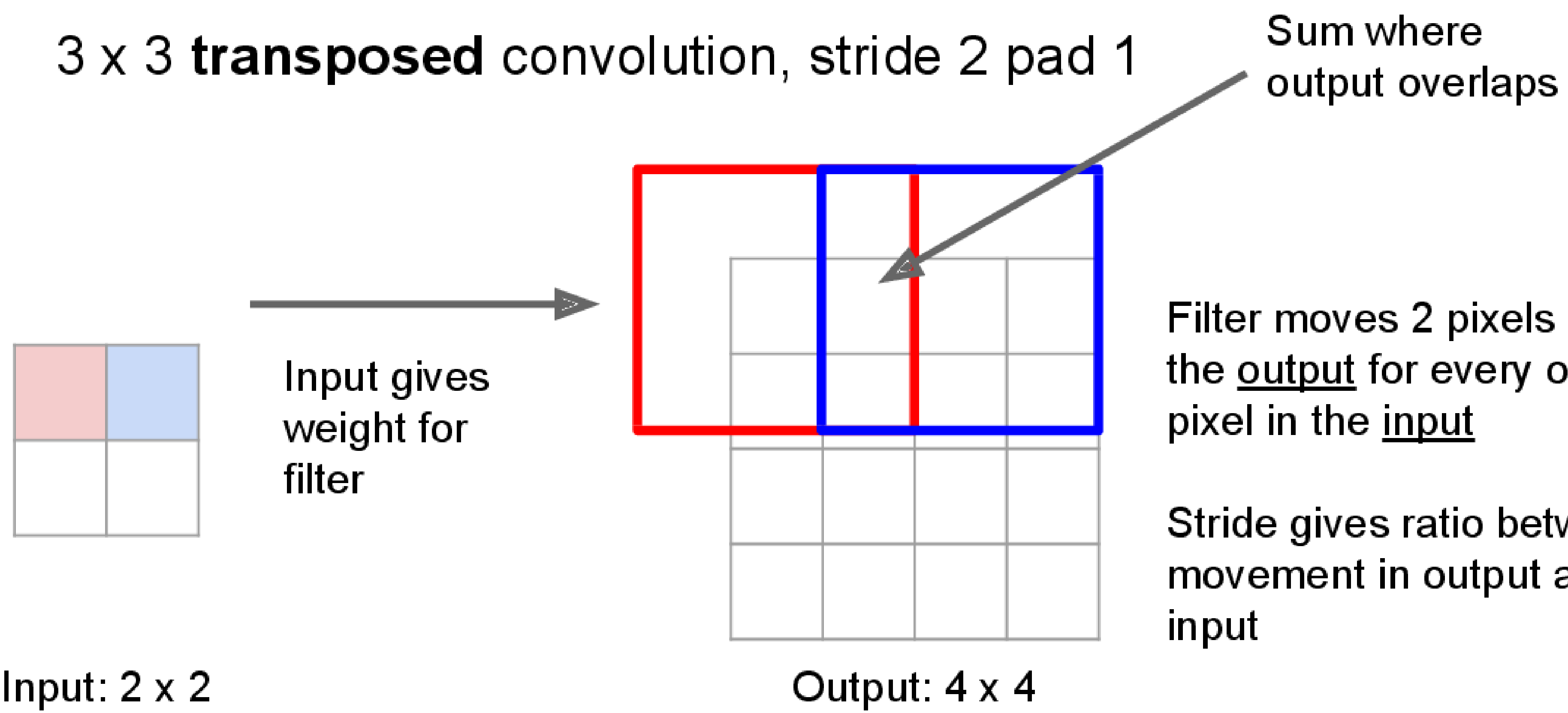


Images from [3]

Learnable Upsampling: Transposed Convolution

Q: Why is it called transposed convolution?

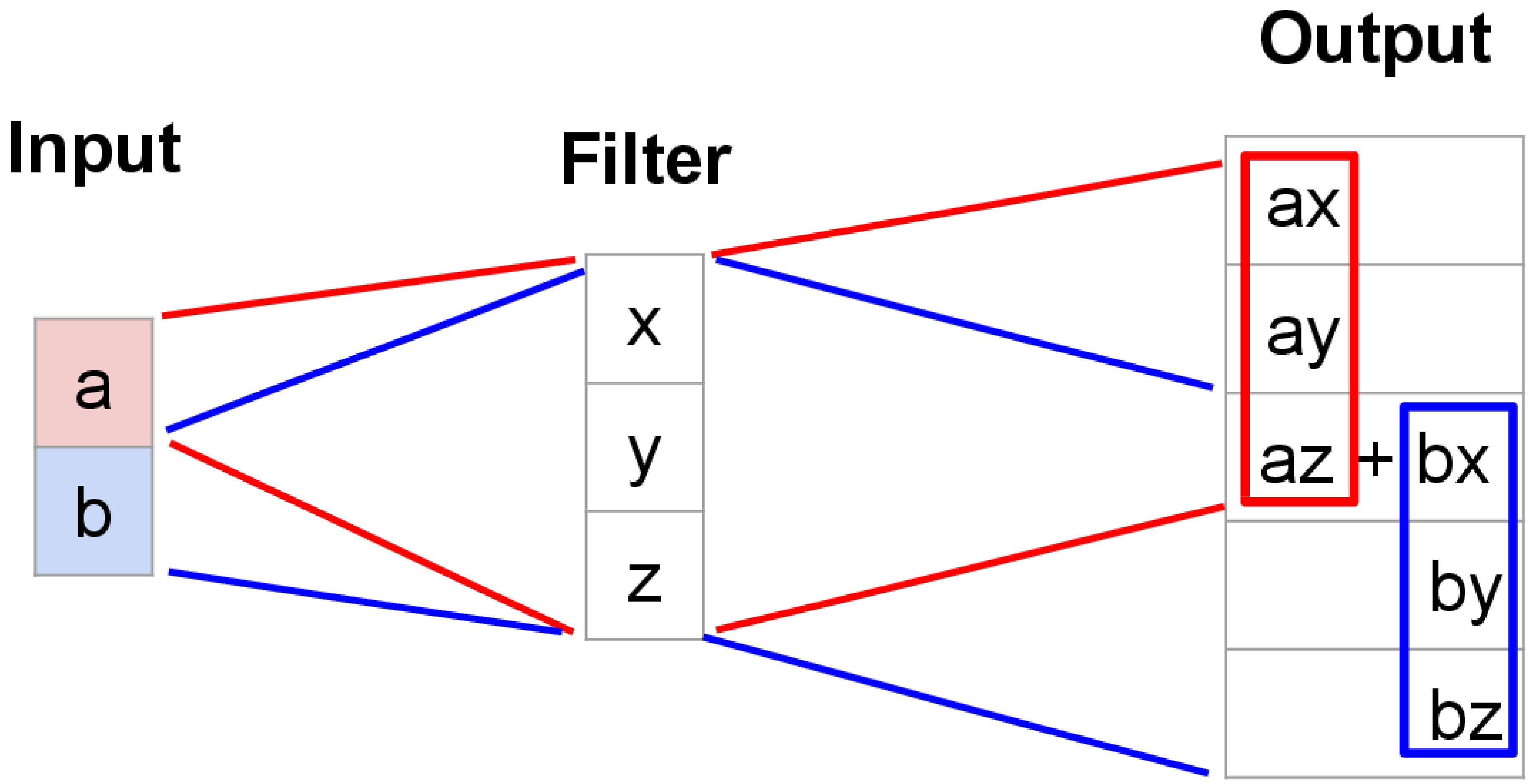
3 x 3 **transposed** convolution, stride 2 pad 1



Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Stride = 2

Slide from [2]

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Slide from [2]

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, <u>stride=2</u>, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

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

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Example: 1D transposed conv, kernel size=3, <u>stride=2</u>, padding=0

Slide from [2]

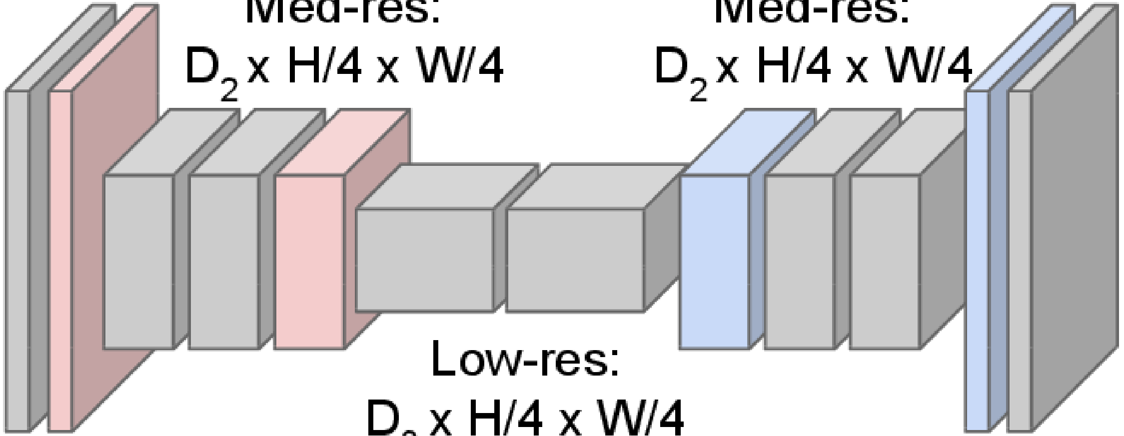
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: Unpooling or strided transposed convolution

Input:
$3 \times H \times W$

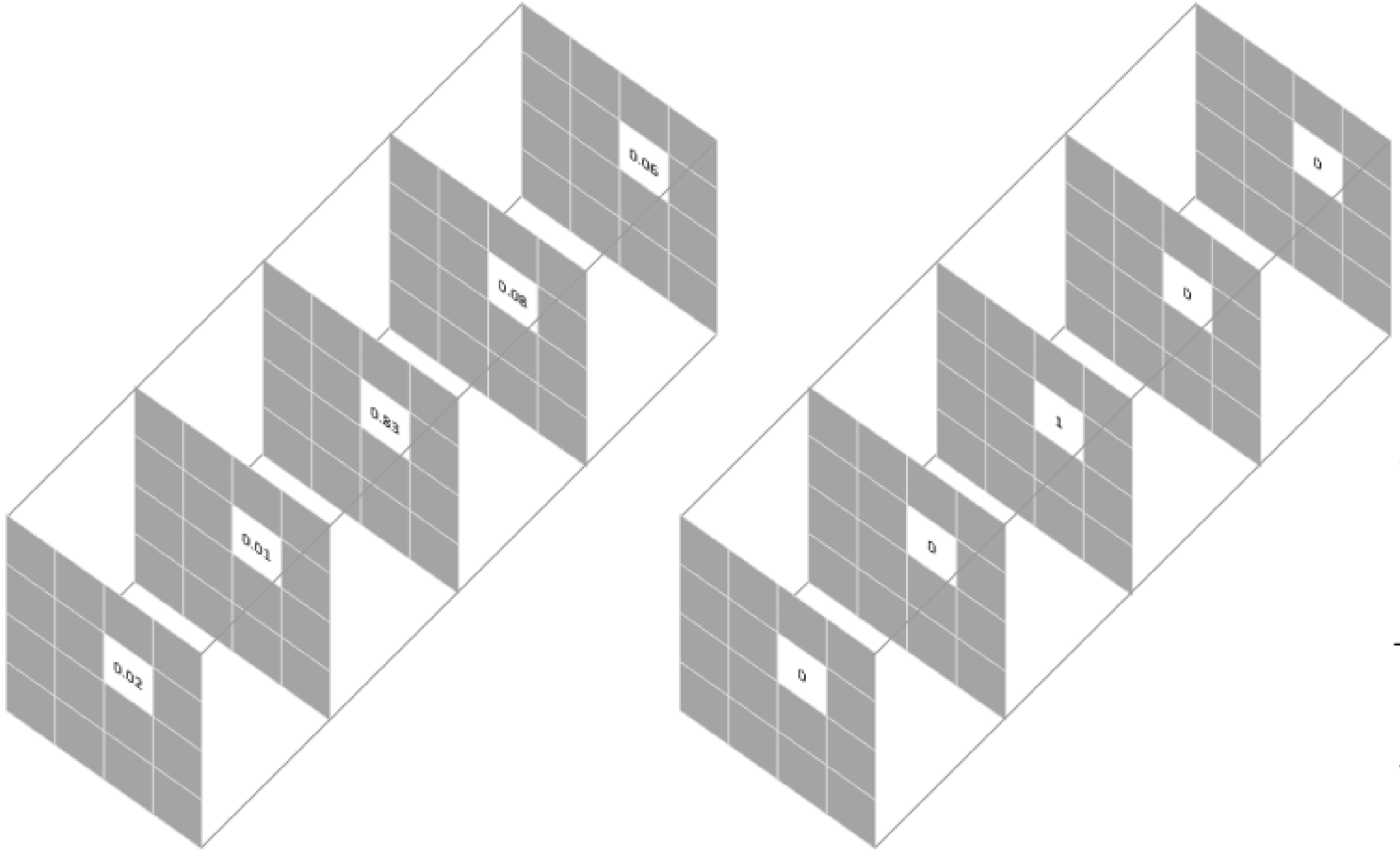


Predictions:
$H \times W$

Slide from [2]

What to do at the end?

- Pixel-wise Softmax for class prediction + pixel-wise cross entropy for loss



Prediction for a selected pixel

Target for the corresponding pixel

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

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

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

Image from [1]

Evaluation Metrics for Semantic Segmentation

- Pixel Accuracy

PA = $\frac{Correctly\ Classified\ Pixels}{All\ Pixels}$

- Intersection over Union: IoU (Jaccard Index)

IoU = $\frac{\|A \cap B\|}{\|A \cup B\|}$

- Dice Coefficient

Dice = $\frac{2\ \|A \cap B\|}{\|A\| + \|B\|}$

Evaluation Metrics for Semantic Segmentation

- Pixel Accuracy

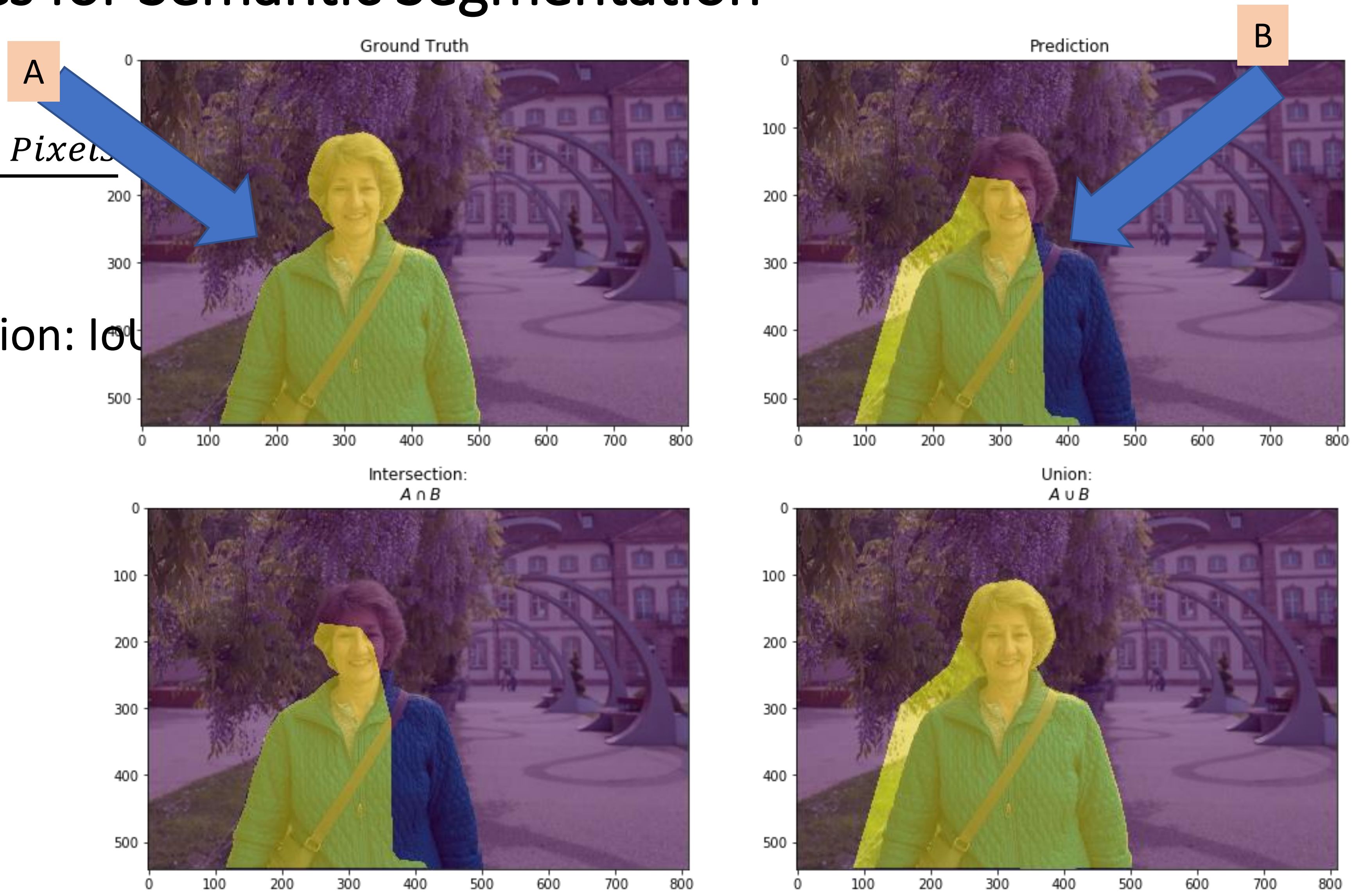
$$PA = \frac{Correctly\ Classified\ Pixels}{All\ Pixels}$$

- Intersection over Union: IoU

$$IoU = \frac{\|A \cap B\|}{\|A \cup B\|}$$

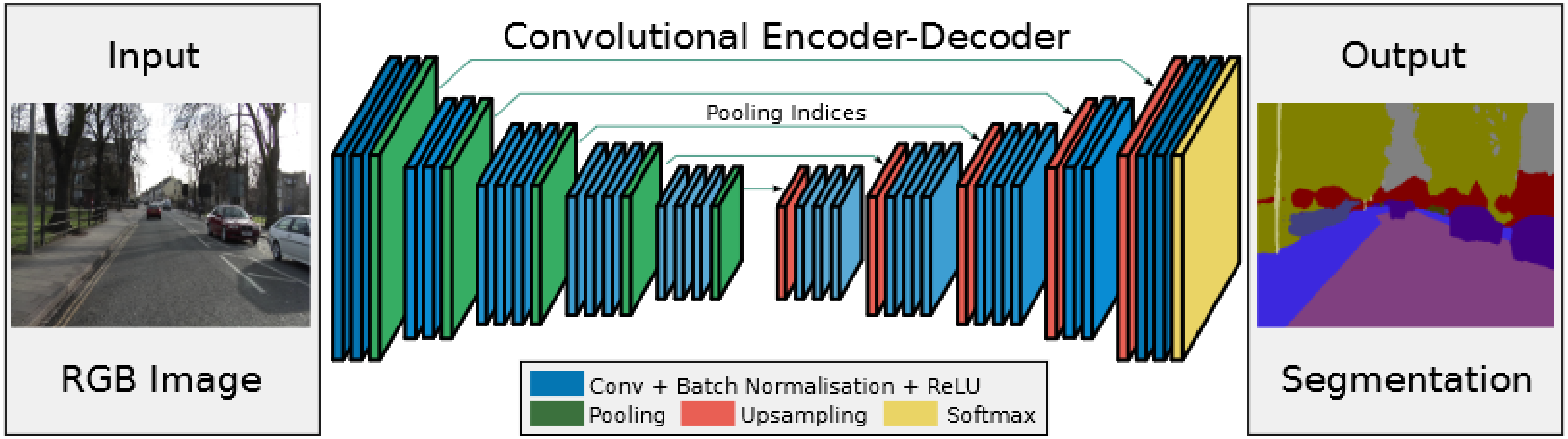
- Dice Coefficient

$$Dice = \frac{2\ \|A \cap B\|}{\|A\| + \|B\|}$$



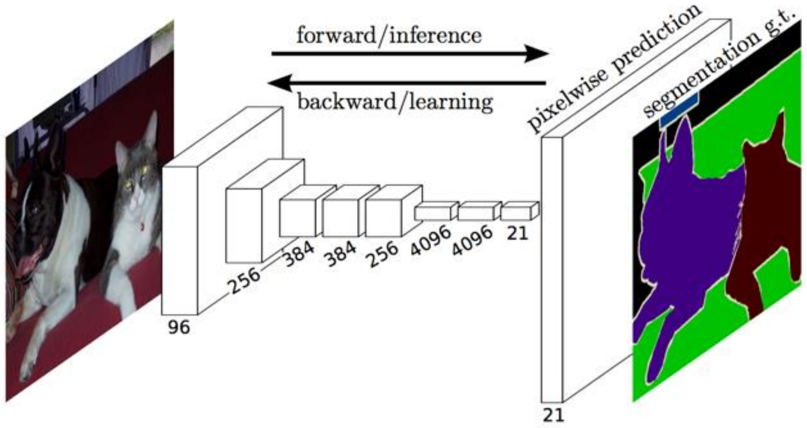
Images from [9]

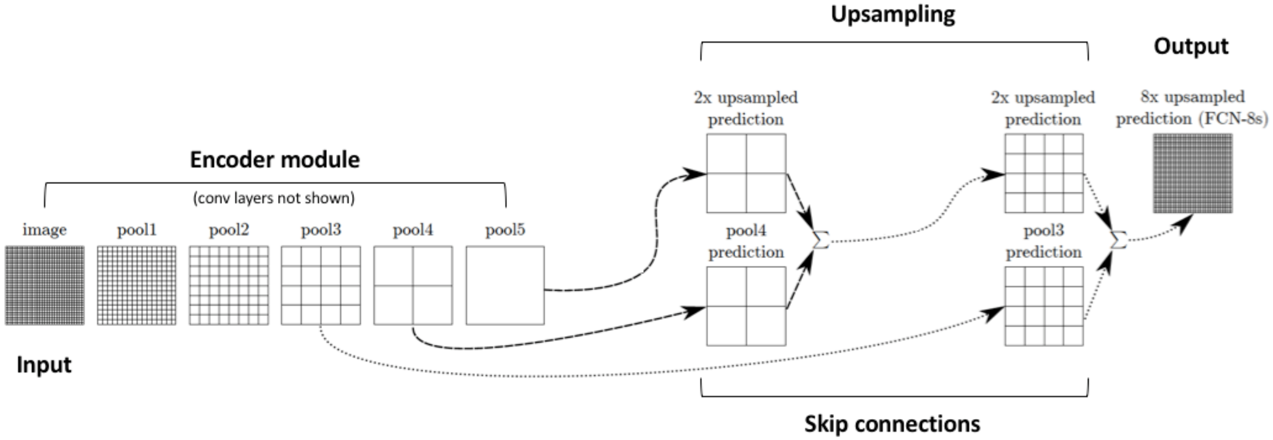
„SegNet“, Badrinarayanan et al. 2015 [6]



- Encoder-Decoder Architecture
- 13 Conv Layers from VGG16 Architecture
- Max unpooling

Fully Connected Network „FCN“, Long et al. 2014 [7]



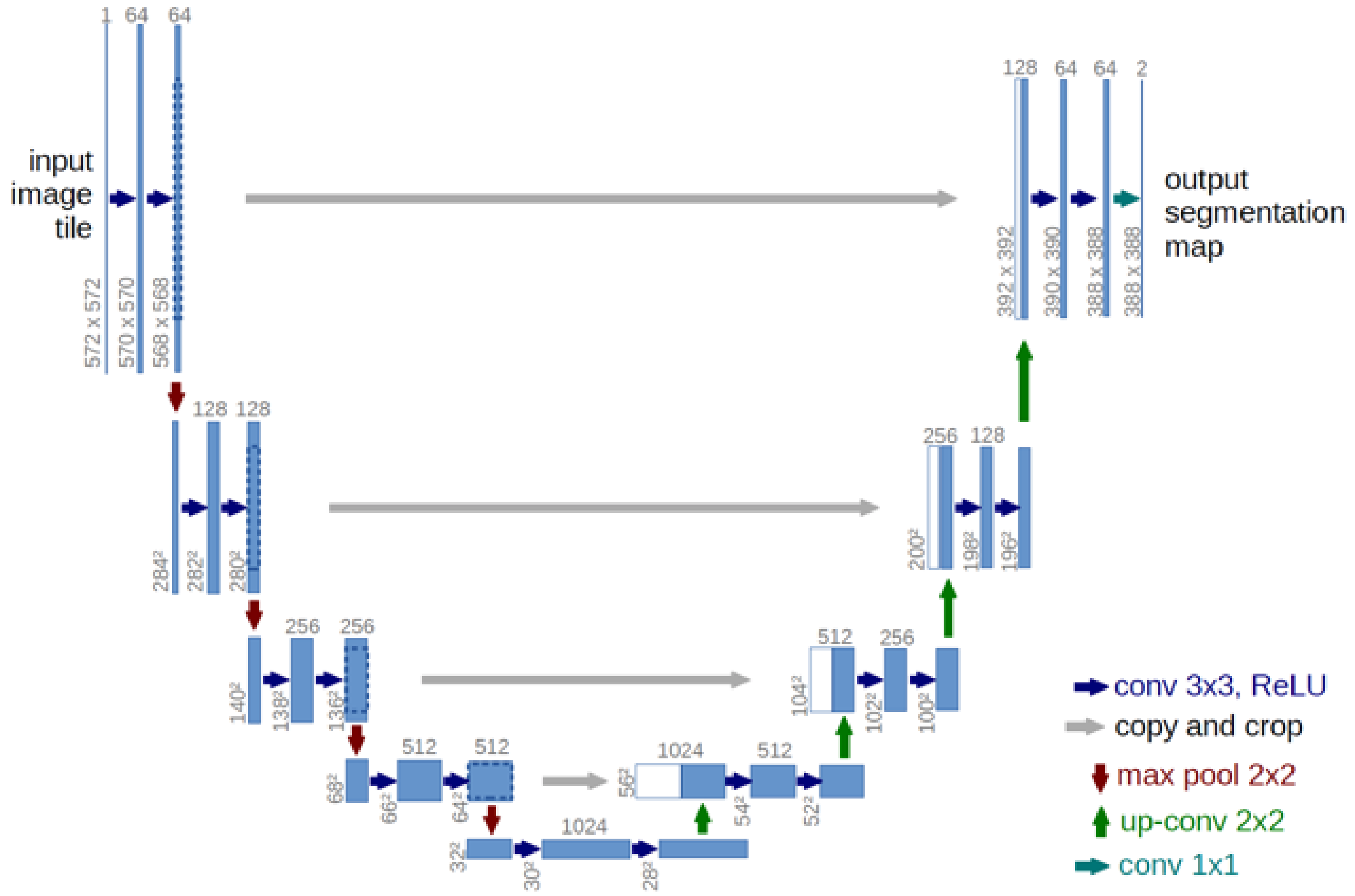


Ground truth target　　Predicted segmentation

Ground truth target　　Predicted segmentation

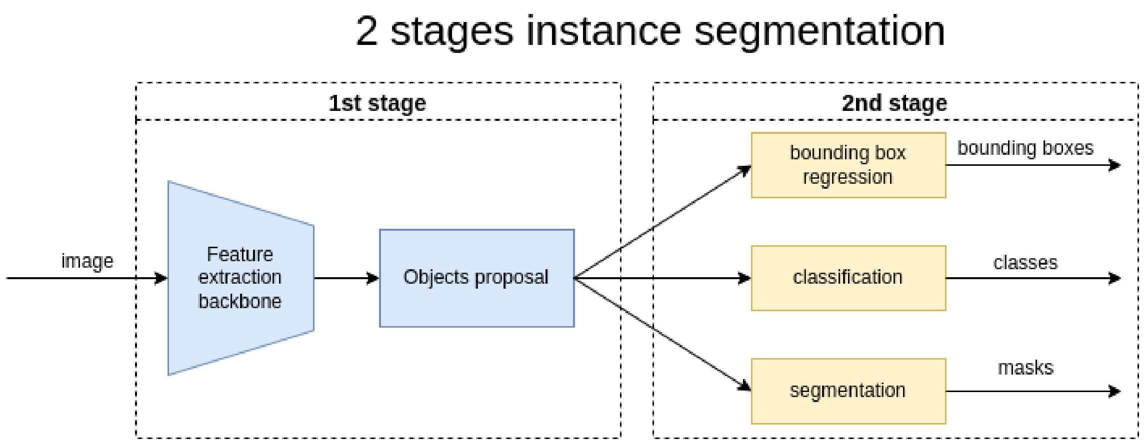
„U-Net“, Ronneberger et al. 2015 [8]

- Biomedical Area = Few Annotated Data
- Encoder-Decoder Architecture
- Skip Connections (cropped)

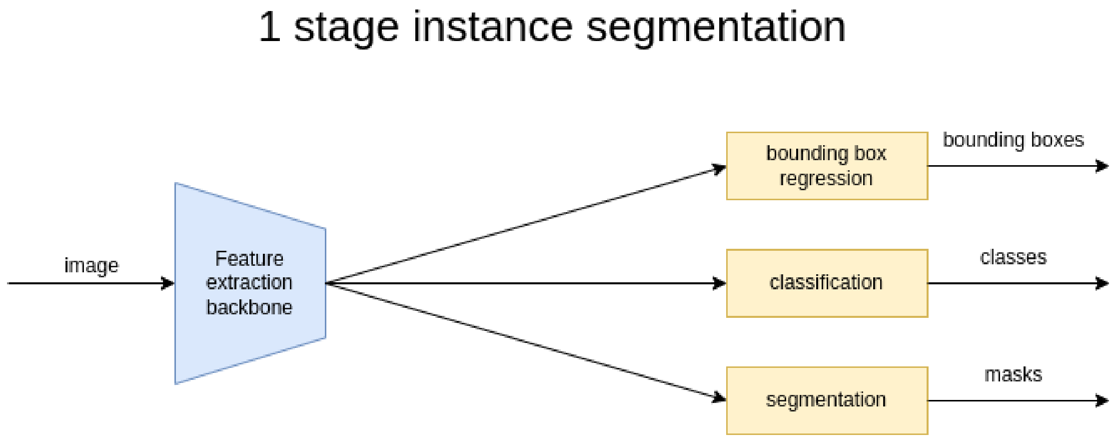


Instance Segmentation

- Detection Based Instance Segmentation (Mask R-CNN)



- Single Shot Instance Segmentation (YOLACT)



Images from [19]

Mask R-CNN

- Backbone
- Region Proposal Network (RPN)
- RoI Align = Interpolate Region Proposals to fixed size
- Nets with 3 outputs (mask, bounding box, image class)

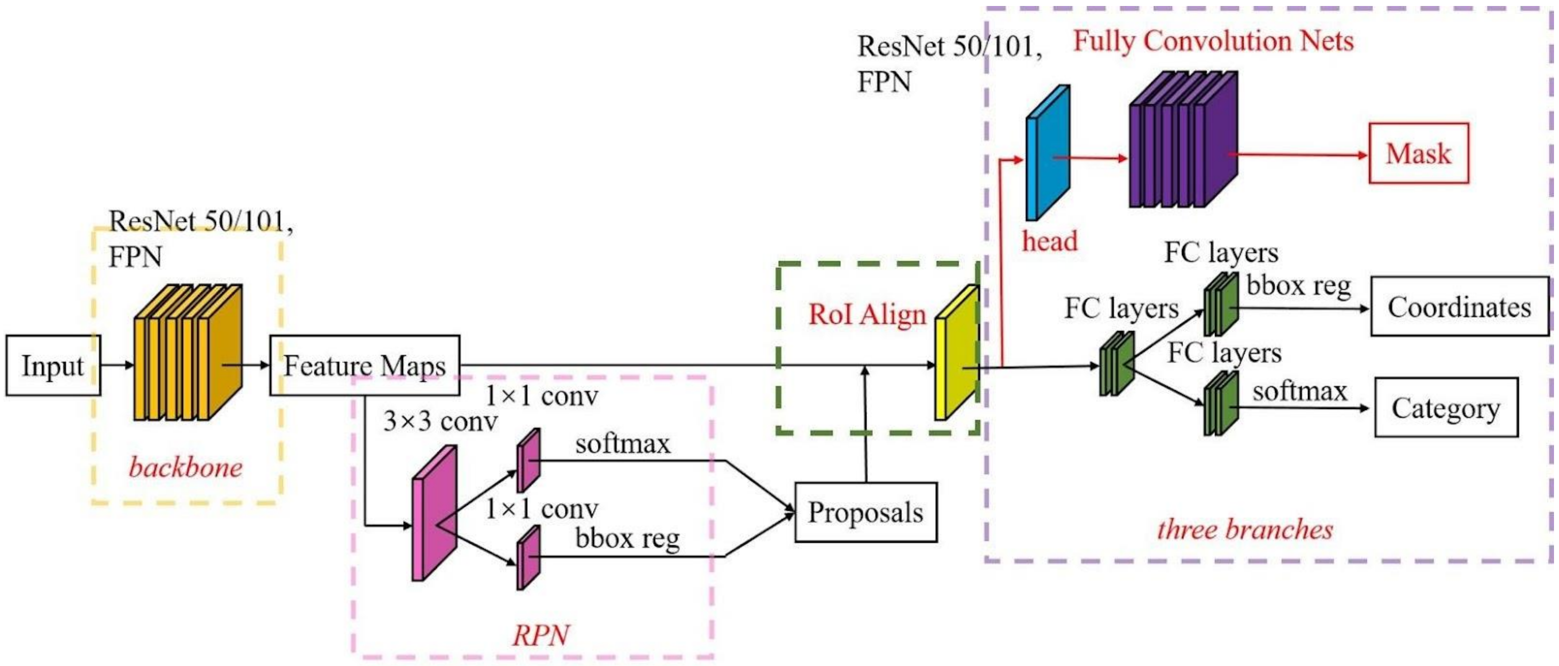


Image from [20]

YOLACT (You only look at coefficients)

- Backbone (FPN + ResNet)
- „Protonet“ yields k „prototypes“
- Mask Coefficients (prediction head)
- Mask Assembly

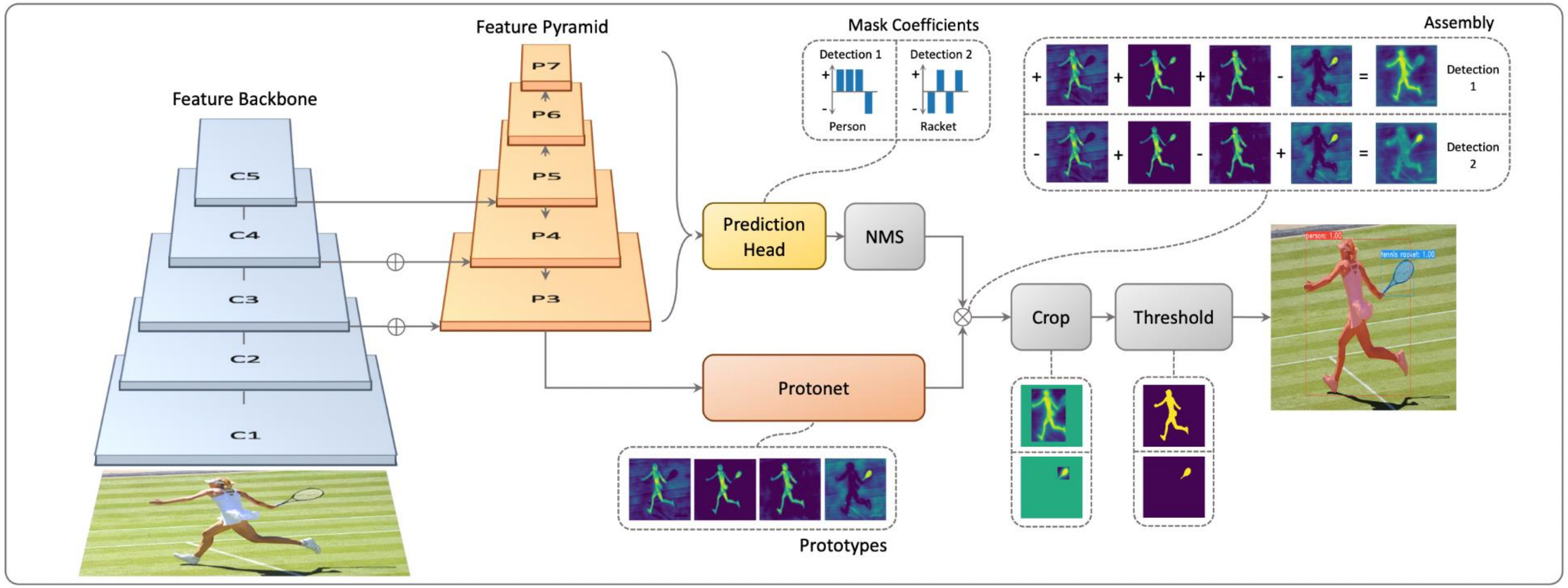


Image from [21]

State of the Art: COCO Segmentation Challenge

98	**PolarMask** (ResNeXt-101-FPN)	32.9%	55.4%	33.8%	15.5%	35.1%	46.3%	×	PolarMask: Single Shot Instance Segmentation with Polar Representation	2019	FPN ResNeXt
99	**PolarMask** (ResNet-101-FPN)	30.4%	51.9%	31%	13.4%	32.4%	42.8%	×	PolarMask: Single Shot Instance Segmentation with Polar Representation	2019	FPN ResNet
100	**YOLACT** (ResNet-50-FPN)	29.8%						×	YOLACT: Real-time Instance Segmentation	2019	FPN ResNet
101	**FCIS +OHEM**	29.2%	49.5%		7.1%	31.3%	50.0%	×	Fully Convolutional Instance-aware Semantic Segmentation	2016	
102	**MultiPath Network**	25.0%						×	A MultiPath Network for Object Detection	2016	

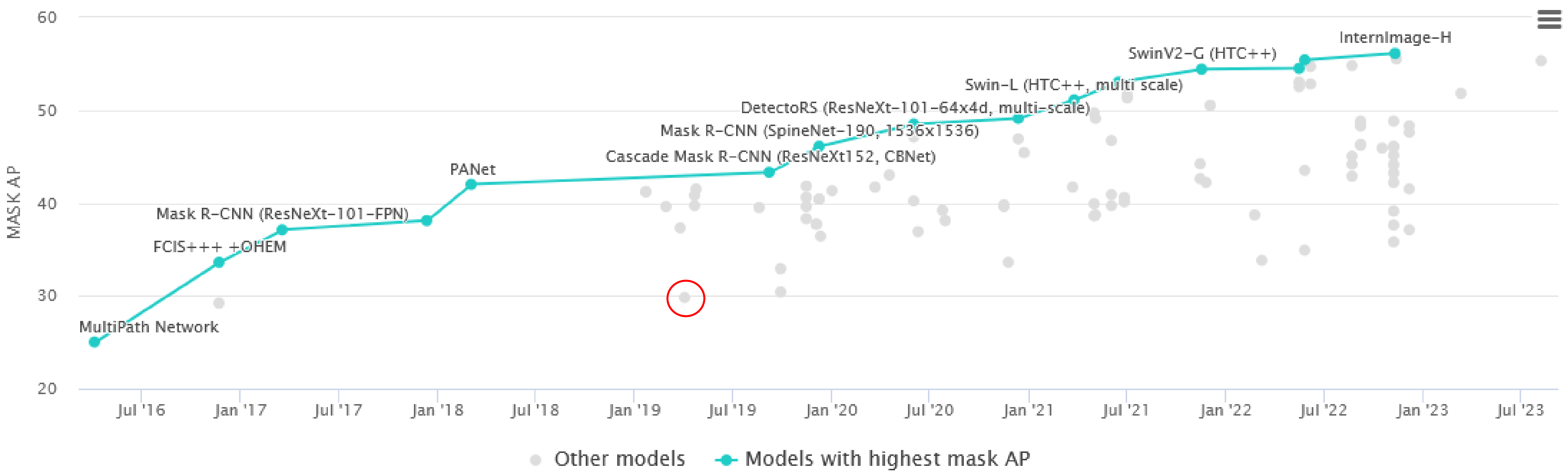
Screenshot from [22]

Instance Segmentation

Instance Segmentation on COCO test-dev

Leaderboard Dataset

View mask AP by Date for All models



Filter: FPN ResNet ResNeXt Cascade single scale multiscale DCN Transformer Swin-Transformer Giant GCN swin-transformer Vision Transformer Focal-Transformer EfficientNet VoVNet untagged

Edit Leaderboard

Screenshot from [22]

General Problems

- Data Annotation cumbersome

- Architectures are kind of a blackbox
  - often unclear what ist doing
  - Explainable AI (XAI)

- Bigger models need lot of memory (even for inference)
  - Smartphones therefore often need „lighter“ models

Thank You

Sources:

- [1] https://www.jeremyjordan.me/semantic-segmentation/
- [2] http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf
- [3] https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11
- [4] https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/
- [5] https://arxiv.org/pdf/1505.04366.pdf
- [6] https://arxiv.org/pdf/1511.00561.pdf
- [7] https://arxiv.org/pdf/1411.4038.pdf
- [8] https://arxiv.org/pdf/1505.04597.pdf
- [9] https://www.jeremyjordan.me/evaluating-image-segmentation-models/
- [10] https://blog.roboflow.com/mask-rcnn/

Sources:

- [11] https://torchio.readthedocs.io/transforms/transforms.html
- [12] http://celltrackingchallenge.net/annotations/
- [13] https://towardsdatascience.com/review-pspnet-winner-in-ilsvrc-2016-semantic-segmentation-scene-parsing-e089e5df177d
- [14] https://medium.com/@ilias_mansouri/part-1-introduction-to-computer-vision-9a02a393d86d
- [15] http://dx.doi.org/10.25080/Majora-4af1f417-015
- [16] https://wavelab.at/papers/Jalilian21b.pdf
- [17] https://wavelab.at/papers/Prommegger22a.pdf
- [18] https://www.v7labs.com/blog/image-segmentation-guide#panoptic-segmentation
- [19] https://www.reasonfieldlab.com/post/instance-segmentation-algorithms-overview

Sources:

- [20] https://blog.roboflow.com/mask-rcnn/
- [21] https://arxiv.org/pdf/1904.02689.pdf
- [22] https://paperswithcode.com/sota/instance-segmentation-on-coco