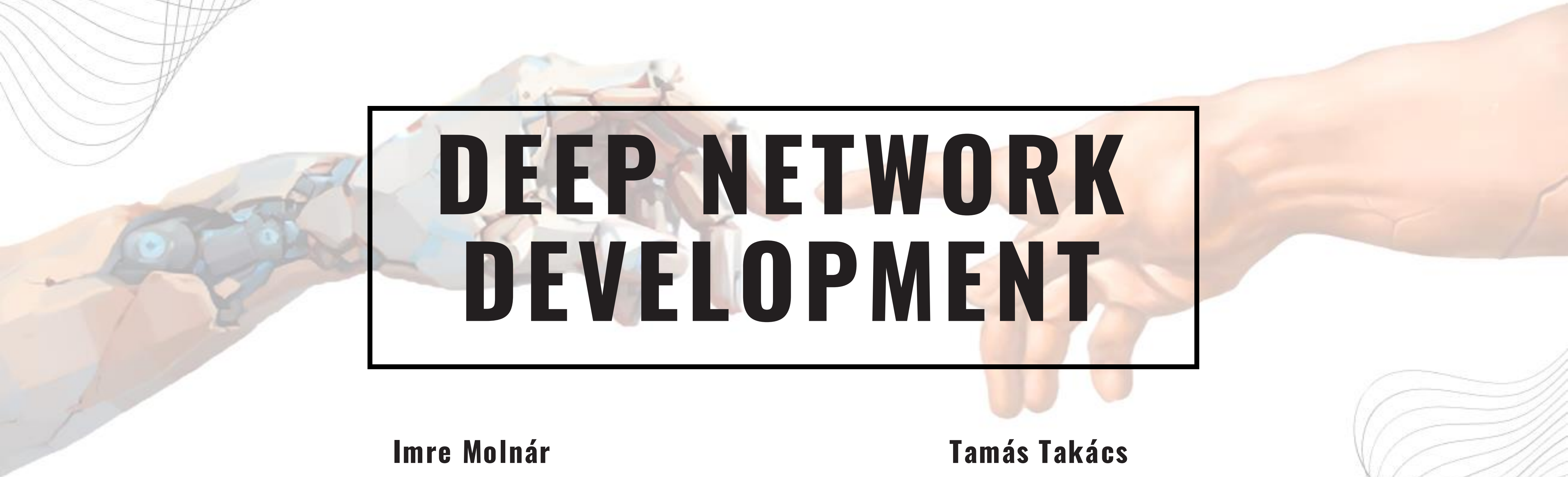


DEEP NETWORK DEVELOPMENT

Imre Molnár
PhD student, ELTE, AI Department
✉ imremolnar@inf.elte.hu
🌐 curiouspercibal.github.io

Tamás Takács
PhD student, ELTE, AI Department
✉ tamastheactual@inf.elte.hu
🌐 tamastheactual.github.io

Deep Network Development

Lecture 6.

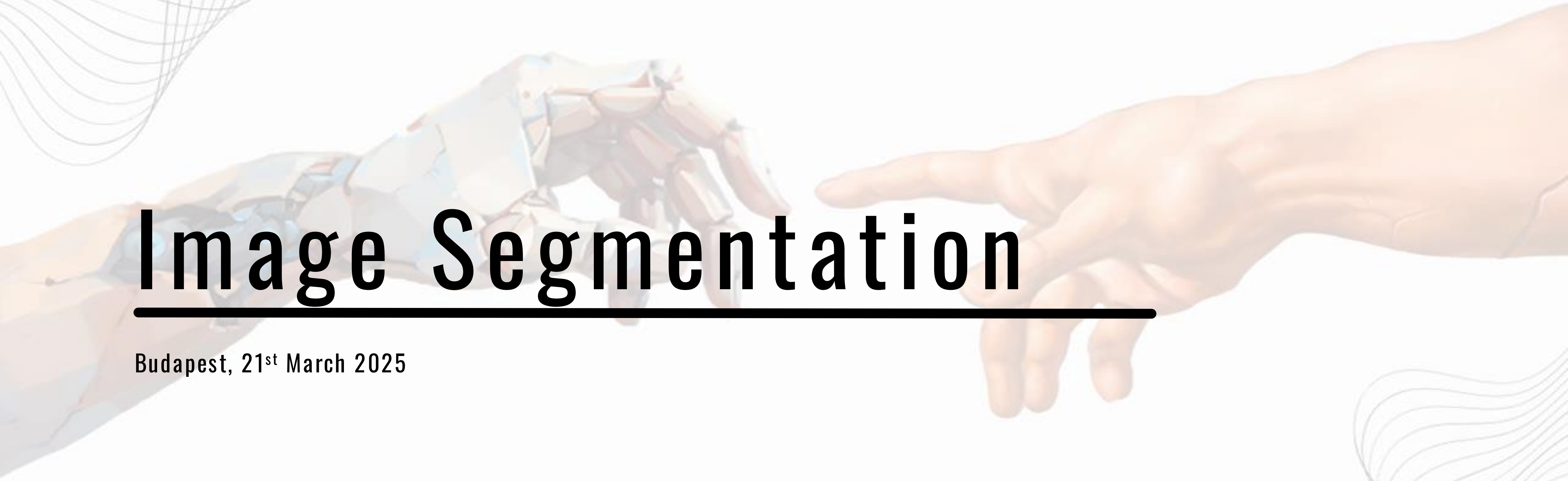


Image Segmentation

Budapest, 21st March 2025

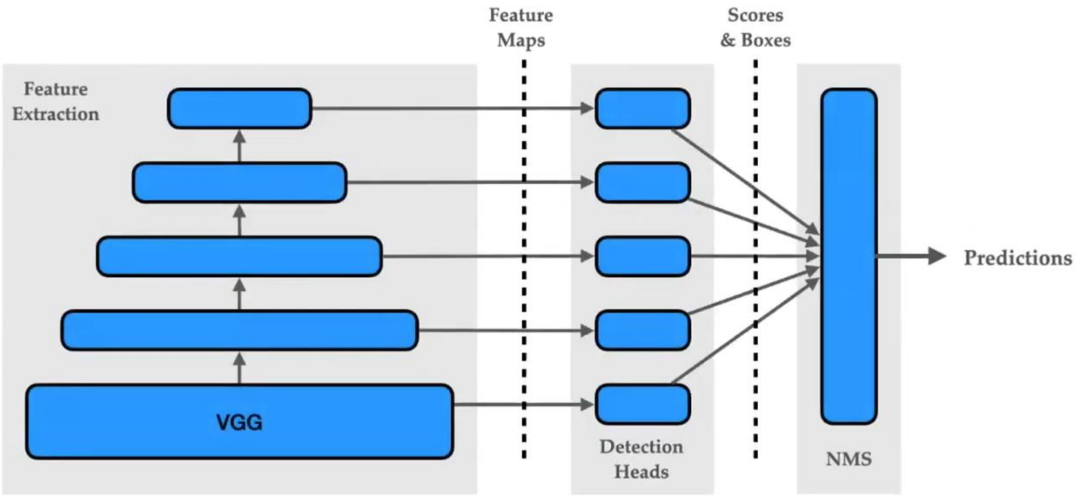
1 Upsampling

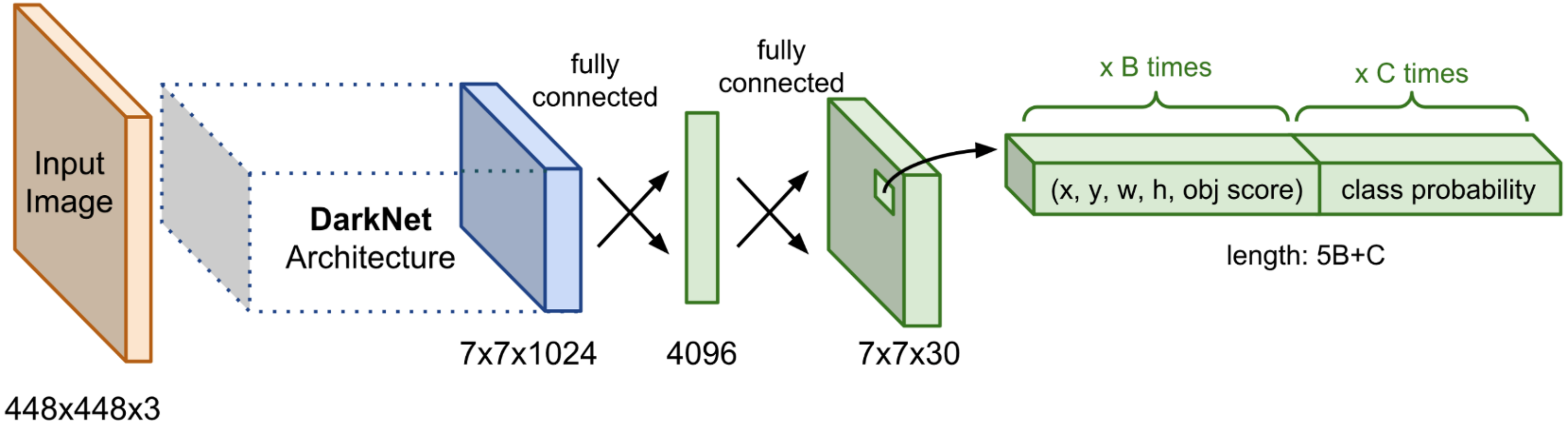
2 Semantic Segmentation

3 Instance Segmentation

Previously on Lecture 5

- Single-Shot Detectors: Naïve SSD, YOLO
- Multi-box Detection
- Non-Max Suppression (NMS)

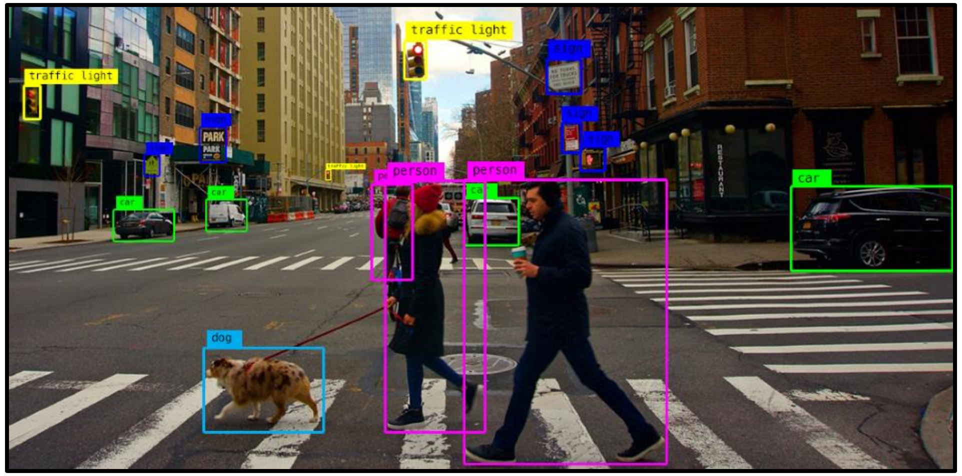


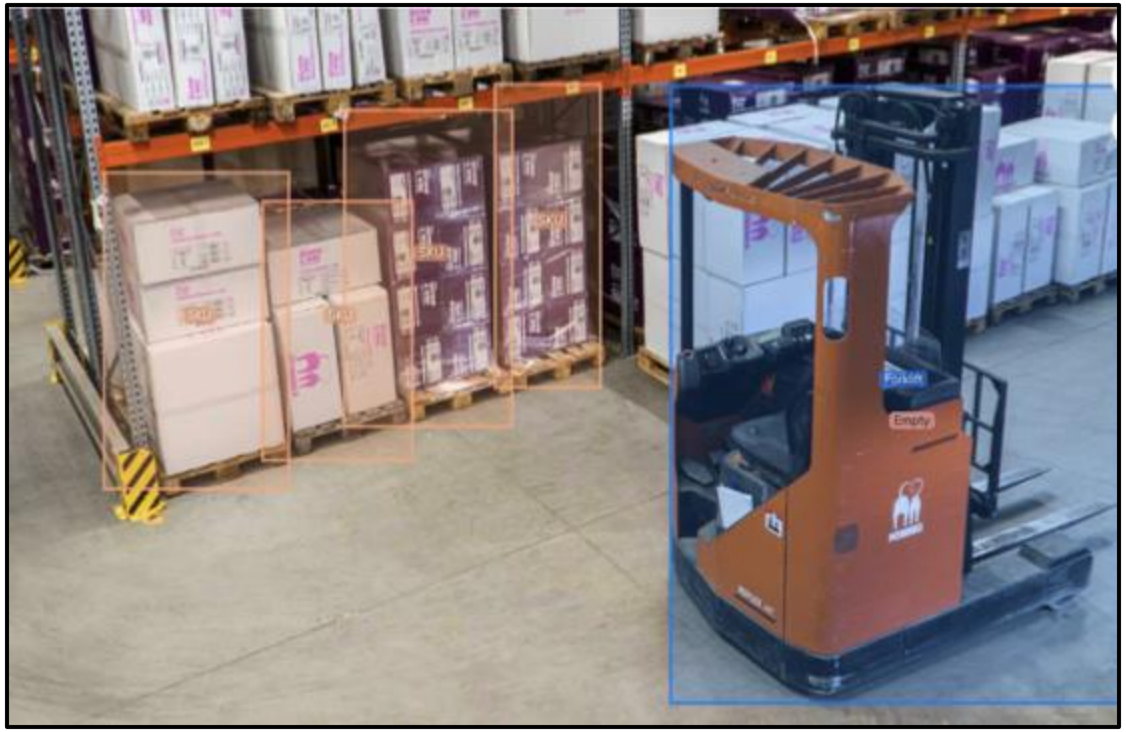


Recap

Previously on Lecture 5

- Object Detection Metrics: IoU, mAP, etc.
- Applications of Object Detection

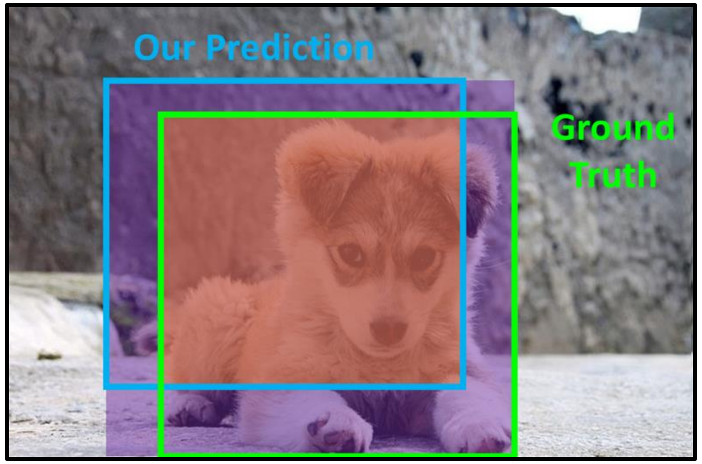




How can we compare our prediction to the ground-truth box?

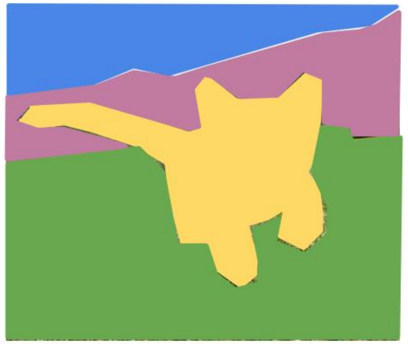
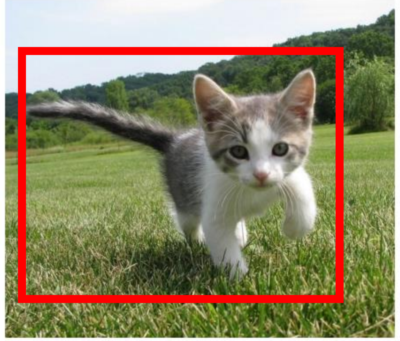
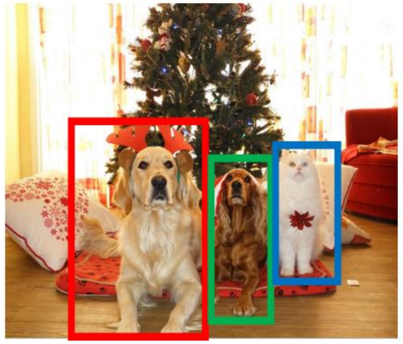
Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



Recap

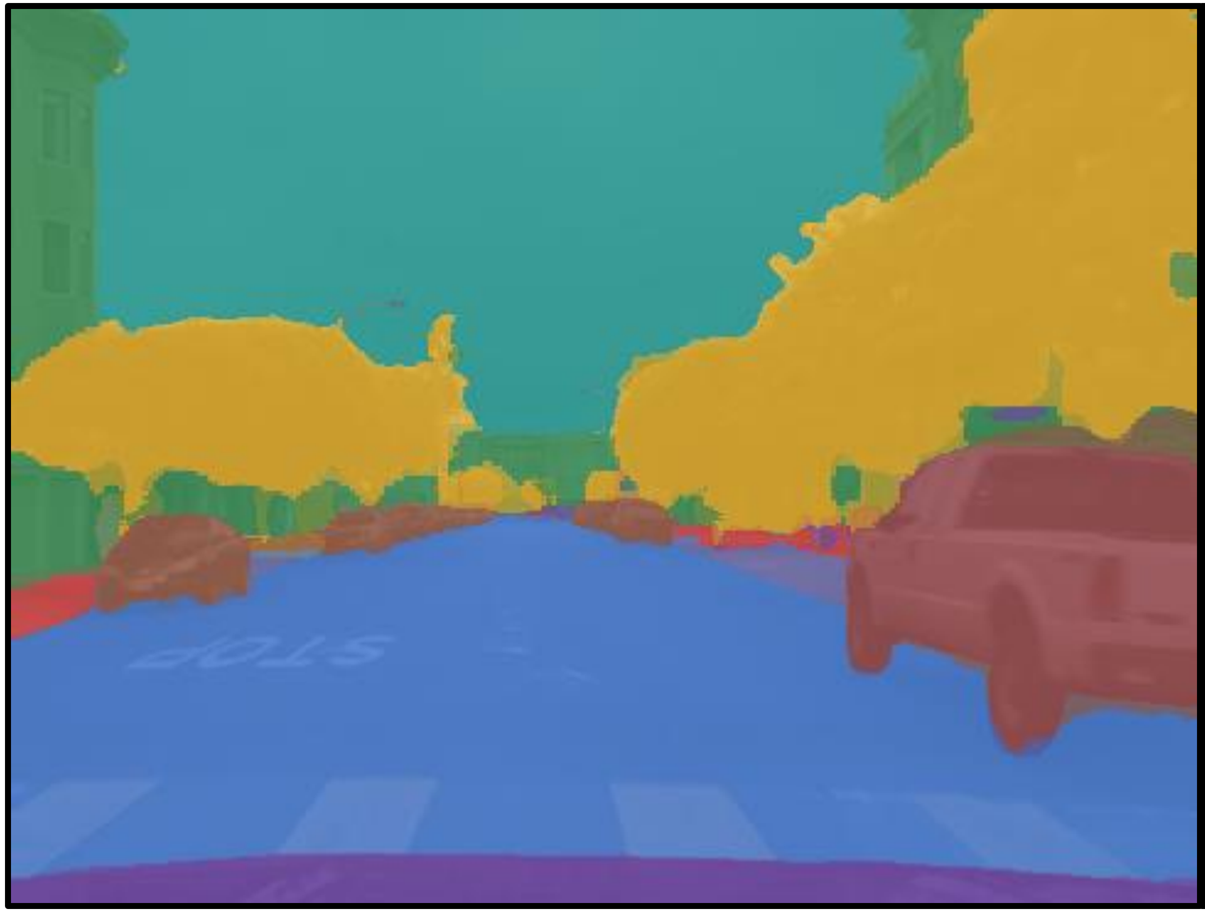
Supervised Learning tasks

Classification	Semantic Segmentation	Classification + Localization	Object Detection	Instance Segmentation
				
CAT	GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT
Single Object	No objects, just pixels	Single Object	Multiple Objects	Multiple Objects
✓	⟳	✓	✓	⟳

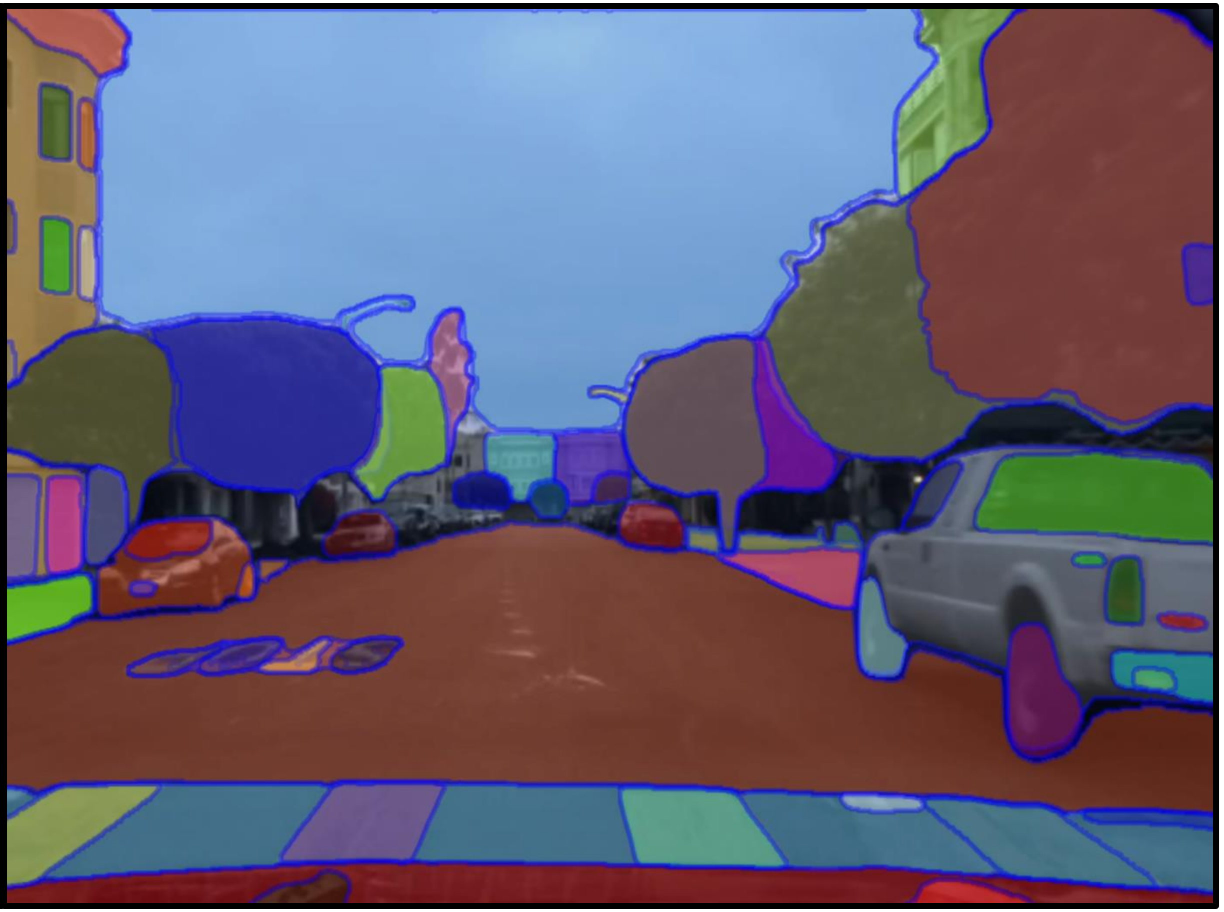
Applications of Image Segmentation

Image Segmentation

Semantic Segmentation

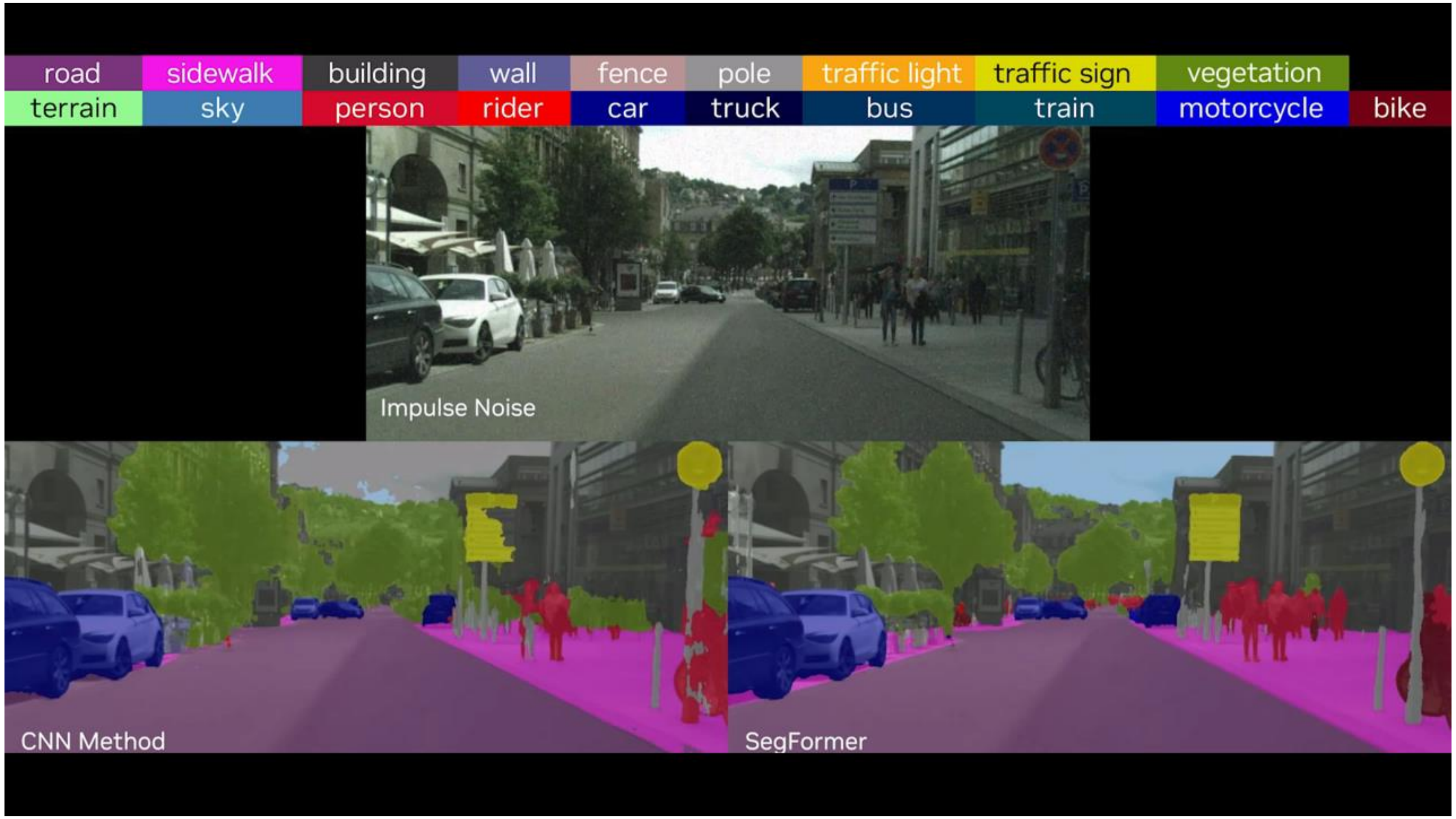


Instance Segmentation



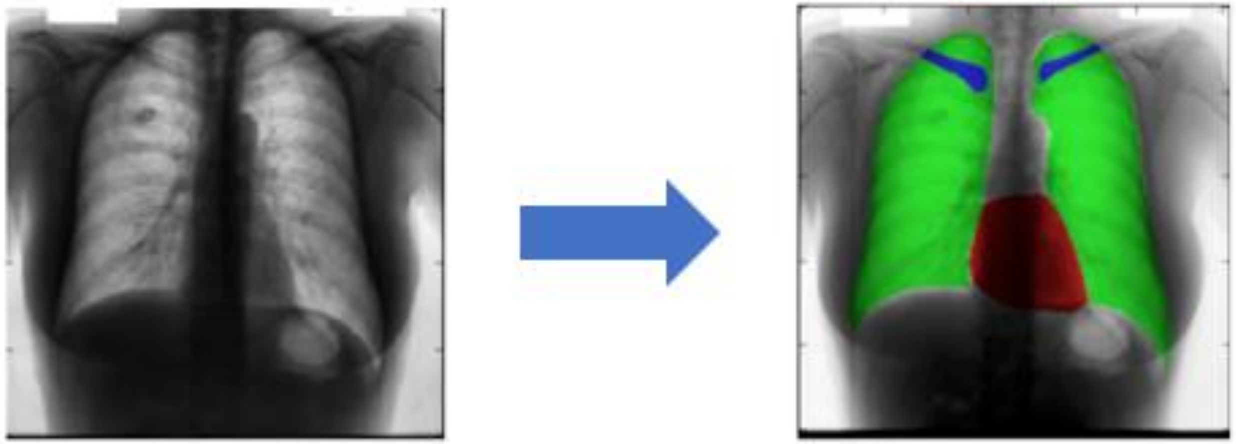
Video Segmentation

Autonomous Driving example - NVIDIA DRIVE (2024)



Applications

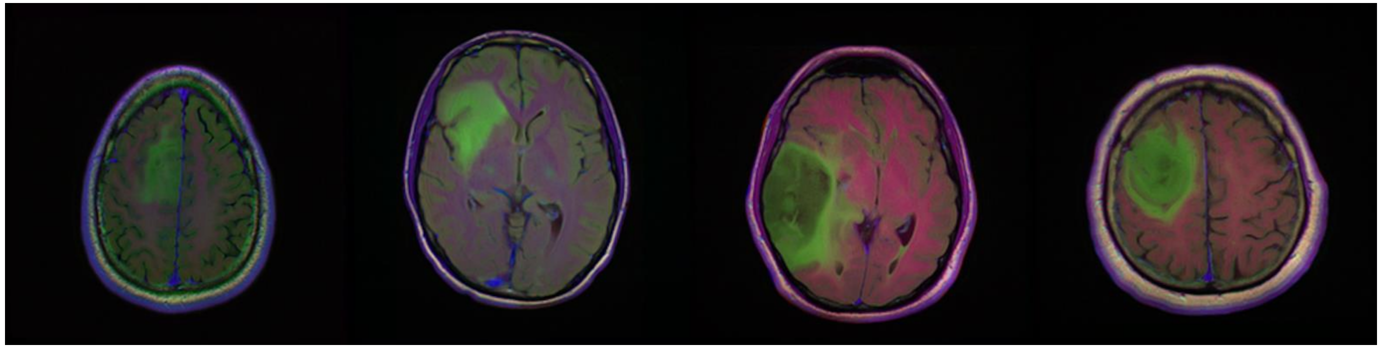
- Medical image diagnosis



Input Image

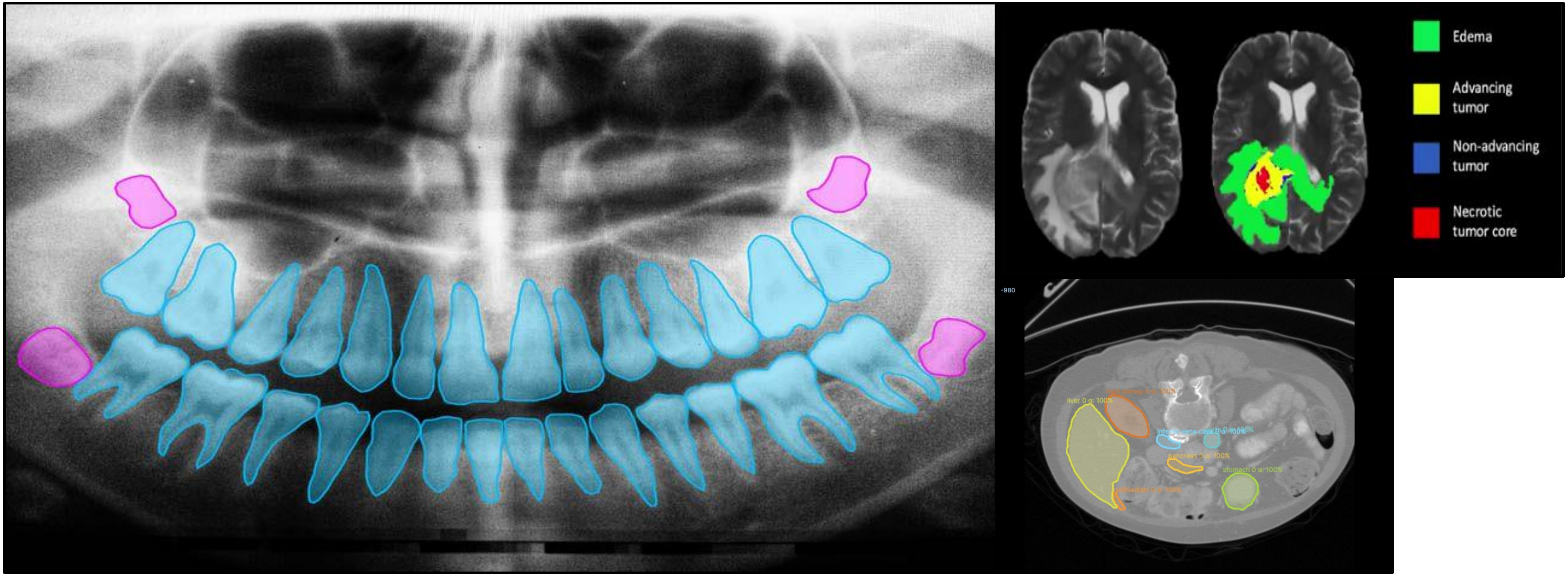
Segmented Image

https://github.com/mateuszbuda/brain-segmentation-pytorch
https://github.com/Saswatm123/3D-Brain-Tumor-Segmentation-PyTorch
https://github.com/hahnicity/pytorch-lung-segmentation



Applications

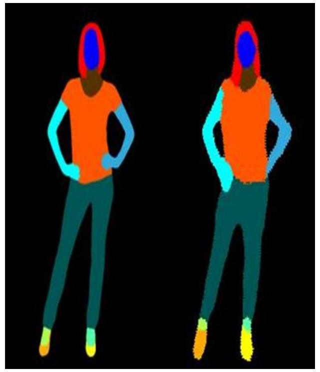
- Medical image diagnosis



Applications

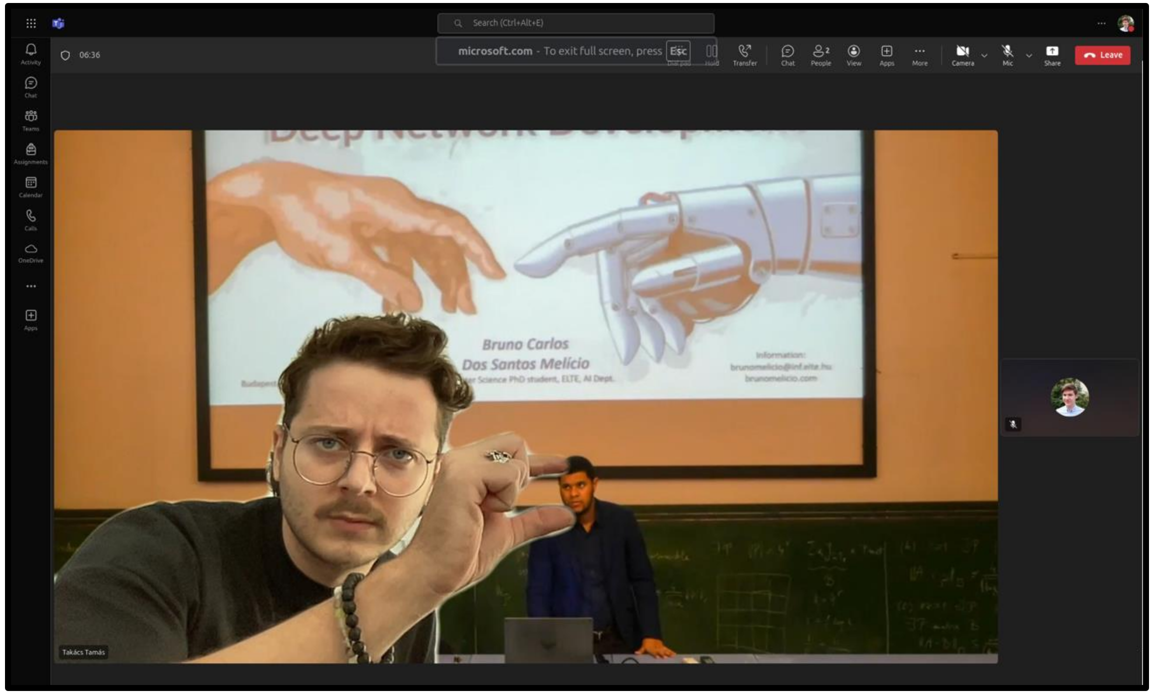
- Entertainment
    - Photo effect
    - Virtual try on

https://towardsdatascience.com/semantic-image-segmentation-with-deeplabv3-pytorch-989319a9a4fb
https://github.com/thuyngch/Human-Segmentation-PyTorch
https://github.com/kishorkuttan/Deep-Virtual-Try-On
https://github.com/shadow2496/VITON-HD
https://github.com/JDAI-CV/Down-to-the-Last-Detail-Virtual-Try-on-with-Detail-Carving



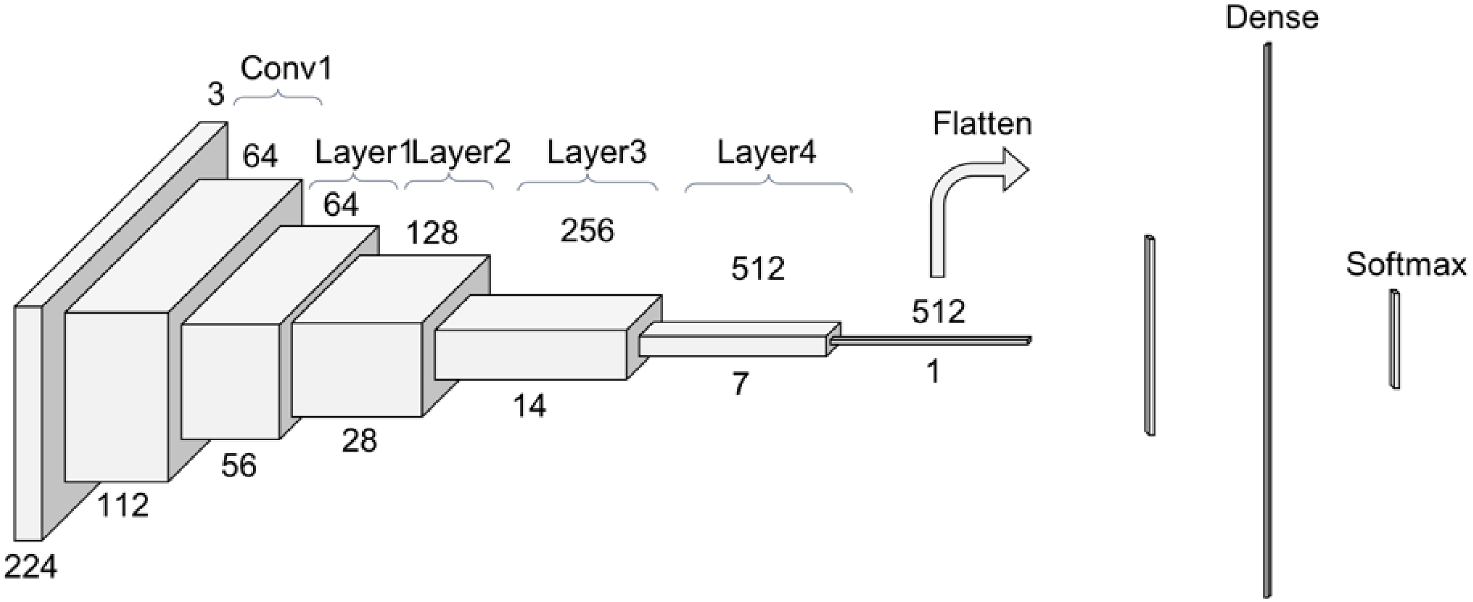
Video Segmentation

- Microsoft Teams segmentation



Classification Task

We map images (x) to labels (y)

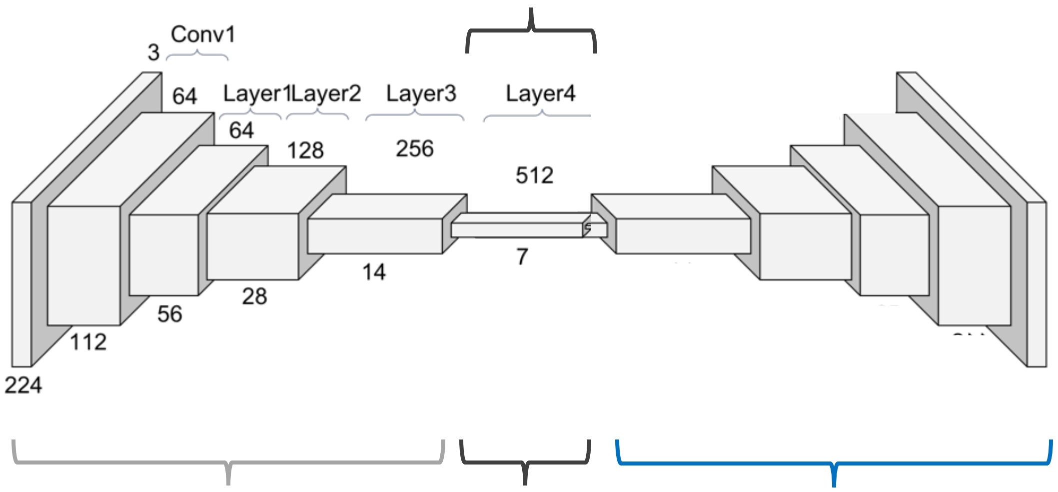


0.01	Dog
0.01	Cat
0.91	Racoon*
…	…
0.01	Flower

Reconstruction task

We try to get back the original image while constraining the network to only learn meaningful information

- Can be used for denoising images
- We lose information during the constraining
- How can we upsample from the latent representation?



Latent space representation

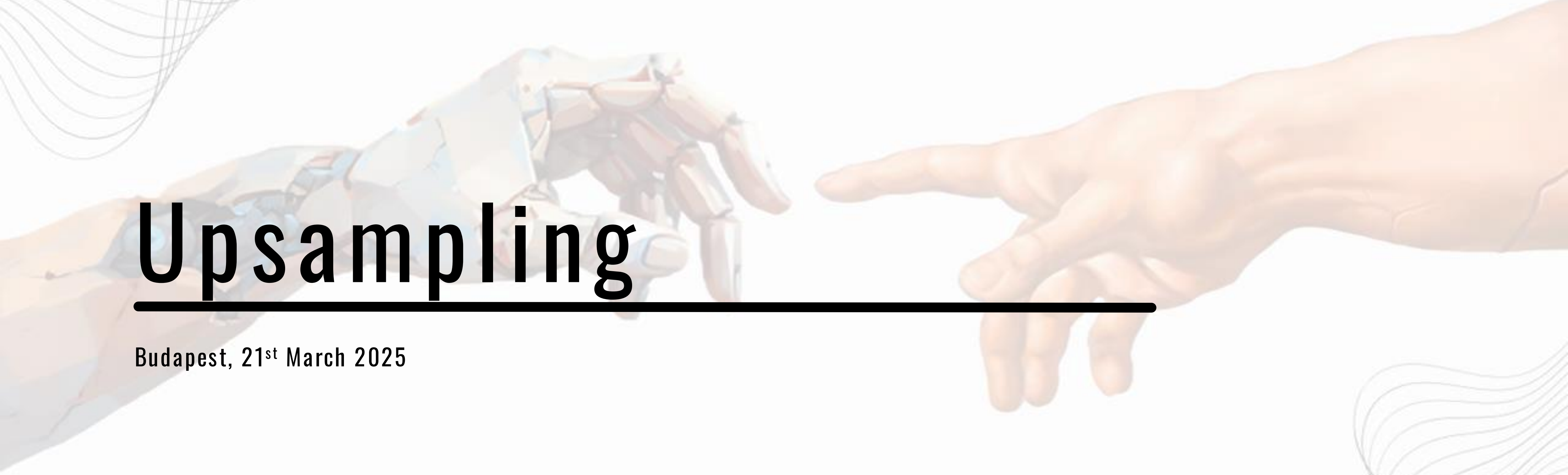
Encoder

Bottleneck

Decoder

Deep Network Development

Lecture 6.



Upsampling

Budapest, 21st March 2025

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

Upsampling

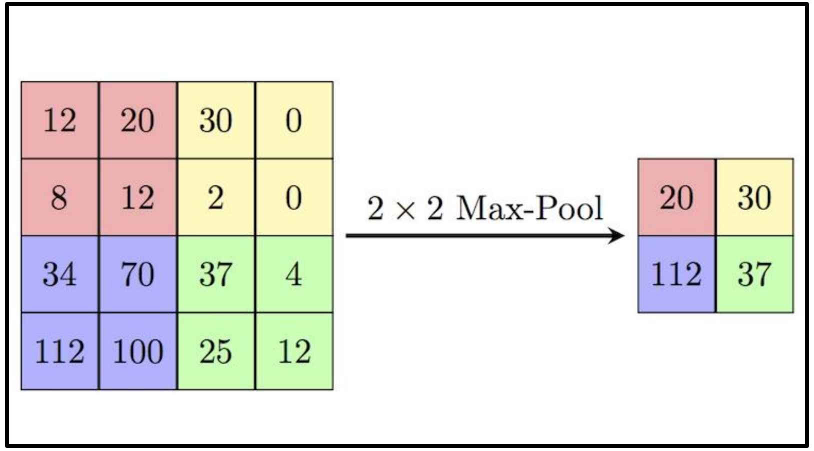
How to upsample?

1. Unpooling
2. Transposed Convolution

Unpooling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (ie. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.

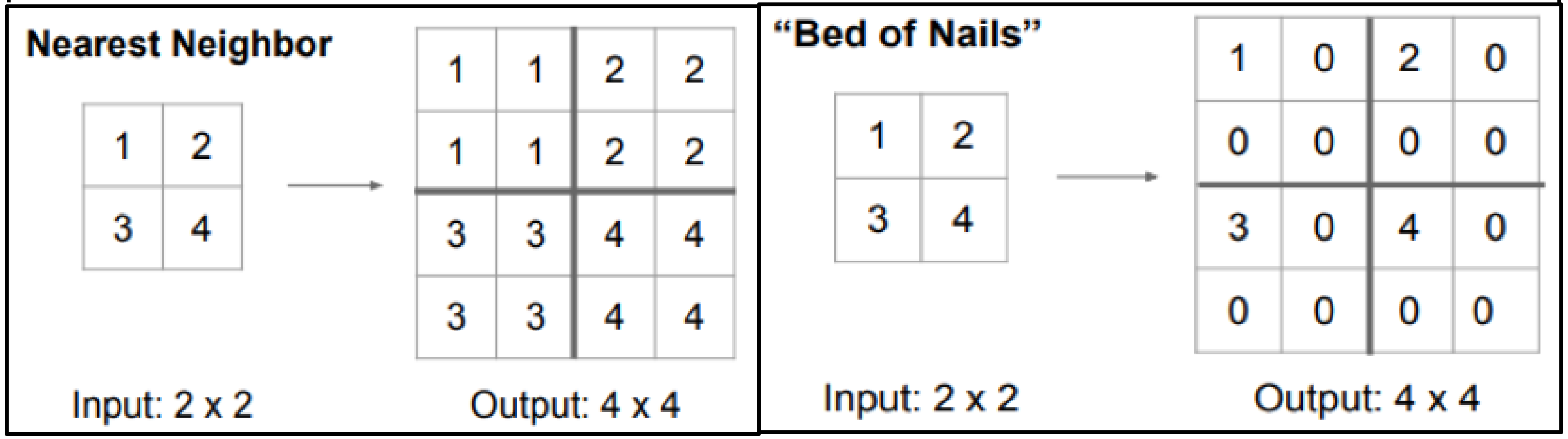
Pooling



12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

Unpooling



Nearest Neighbor

1	2
3	4

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

Input: 2 x 2

Output: 4 x 4

"Bed of Nails"

1	2
3	4

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

Input: 2 x 2

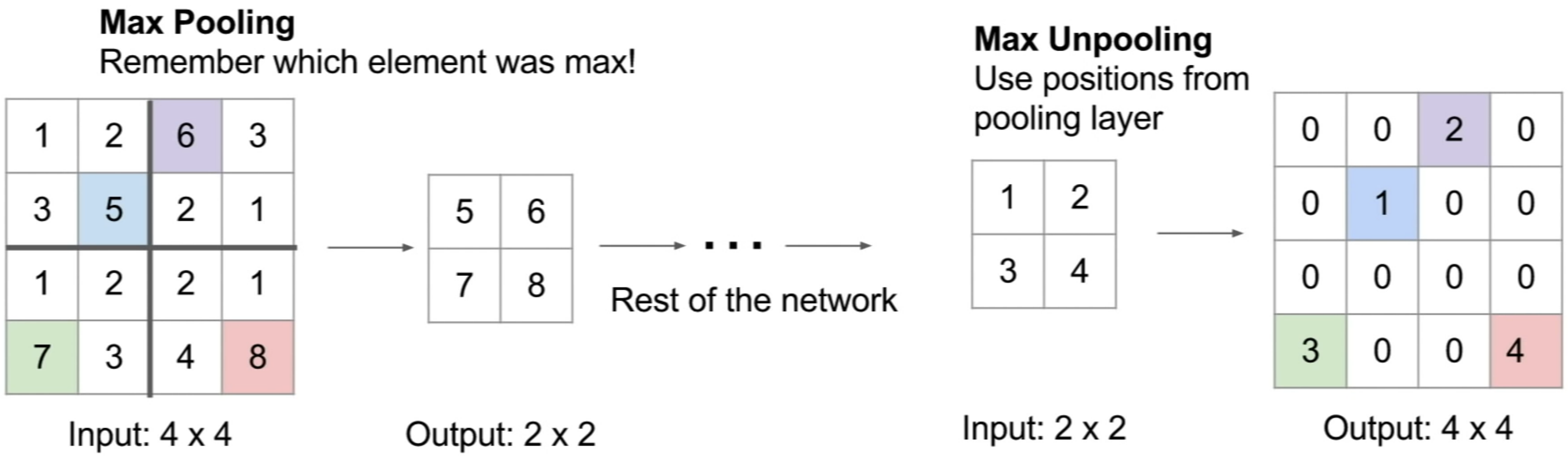
Output: 4 x 4

Bilinear
Linear Shifted

Unpooling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (I.e. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.
- **No weights, nothing to learn here!**

In-Network upsampling: “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

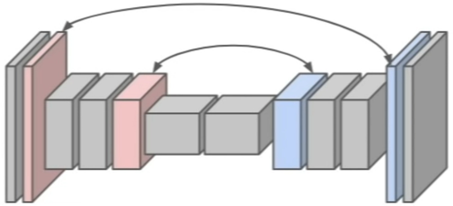
1	2
3	4

Input: 2 x 2

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

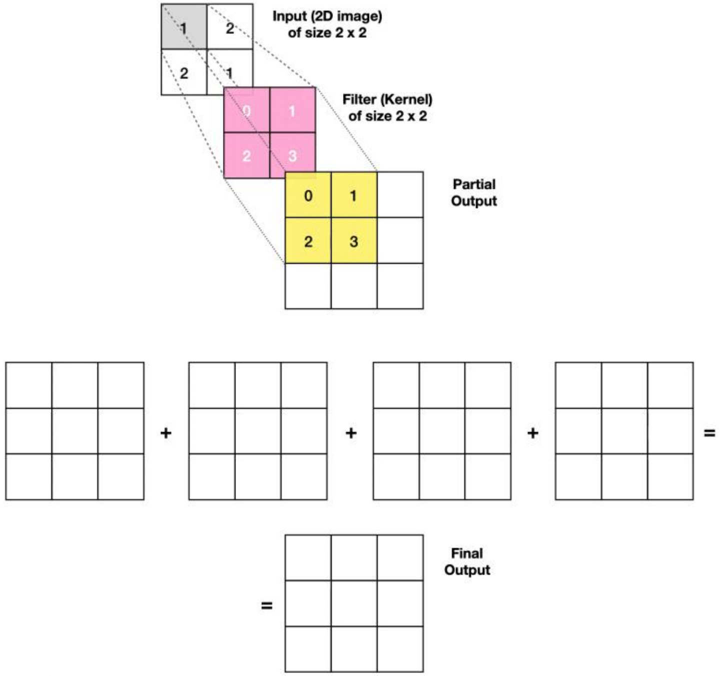
Output: 4 x 4

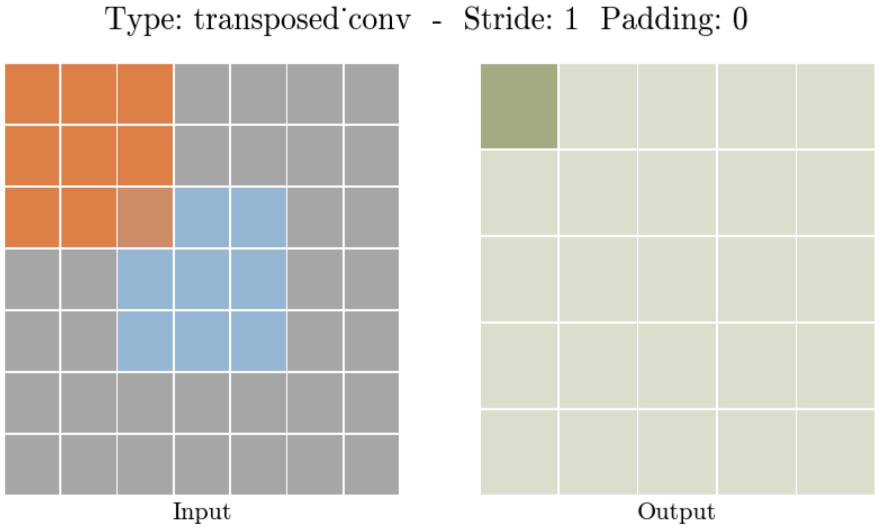
Corresponding pairs of downsampling and upsampling layers



Transposed Convolution

- Most popular approach
- Whereas a typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position, a **transpose convolution** essentially does the opposite. For a transpose convolution, we take a single value from the low-resolution feature map and multiply all the weights in our filter by this value, projecting those weighted values into the output feature map.



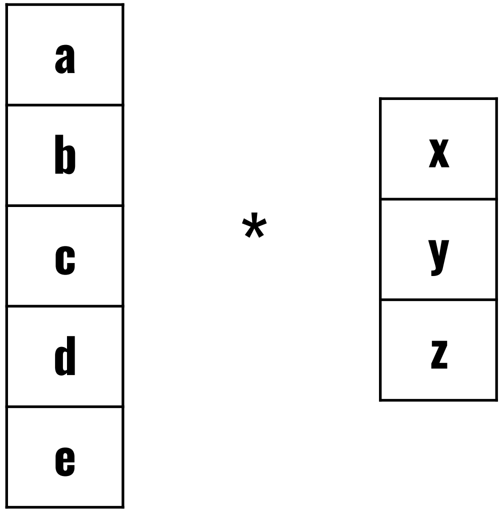


Normal Convolution

- 1D example

Input size: 5

Filter size: 3
Stride: 2



Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

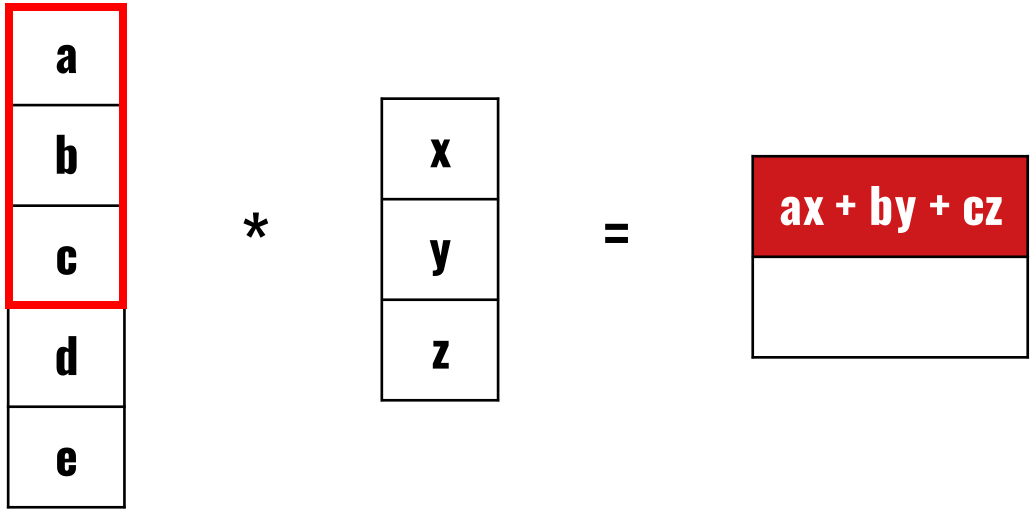
Normal Convolution

- 1D example

Input size: 5

Filter size: 3
Stride: 2

Output size: 2



Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

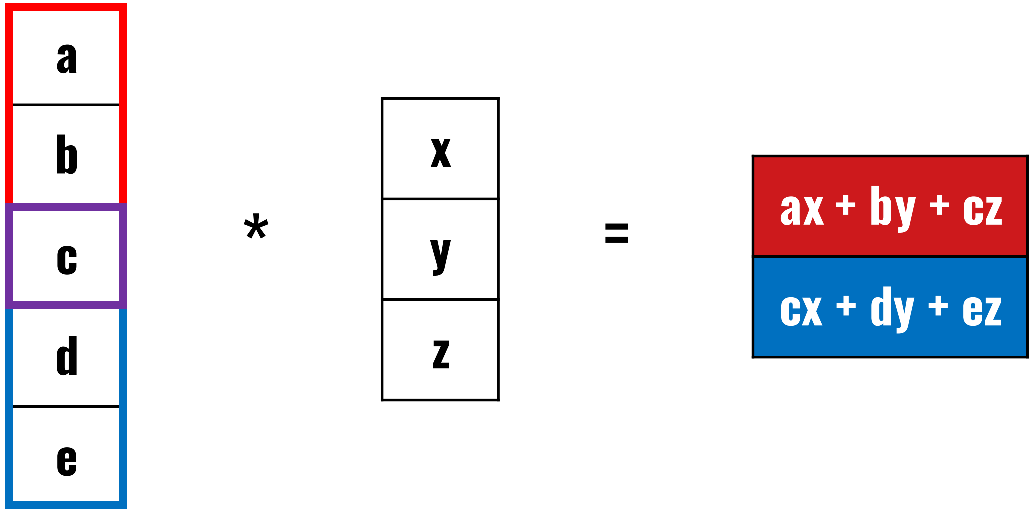
Normal Convolution

- 1D example

Input size: 5

Filter size: 3
Stride: 2

Output size: 2



$n \times n$ image $\quad f \times f$ filter

padding $p \quad$ stride s

$$\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$$

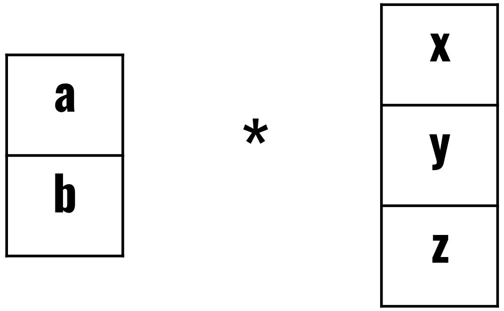
Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2

Filter size: 3
Stride: 2



$$\begin{bmatrix}a\\b\end{bmatrix} * \begin{bmatrix}x\\y\\z\end{bmatrix}$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

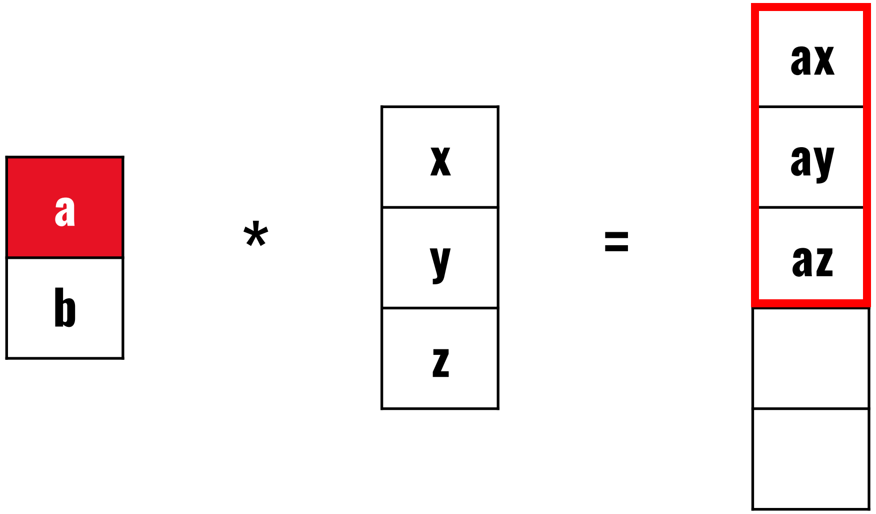
Transposed Convolution

- 1D example

Input size: 2

Filter size: 3
Stride: 2

Output size: 5



Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

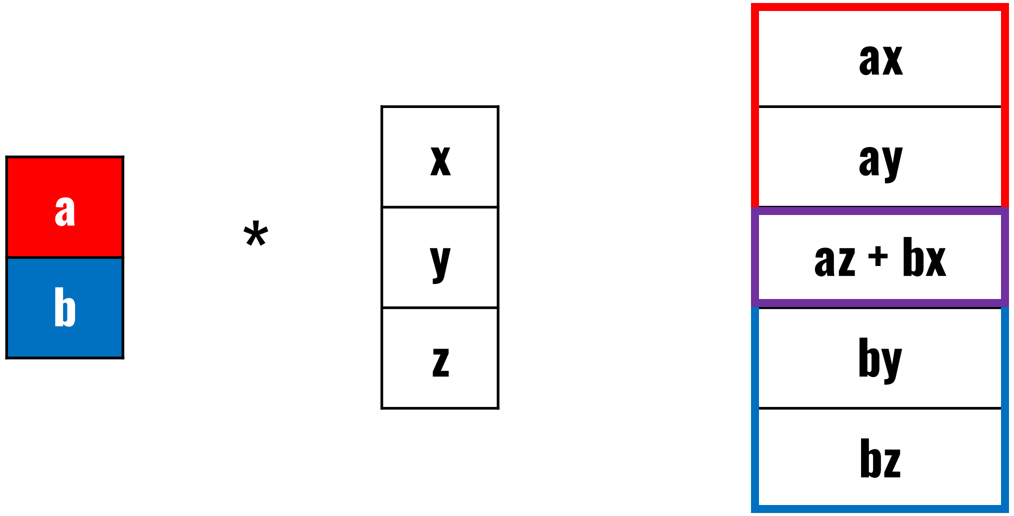
Transposed Convolution

- 1D example

Input size: 2

Filter size: 3
Stride: 2

Output size: 5



a
b

*

x
y
z

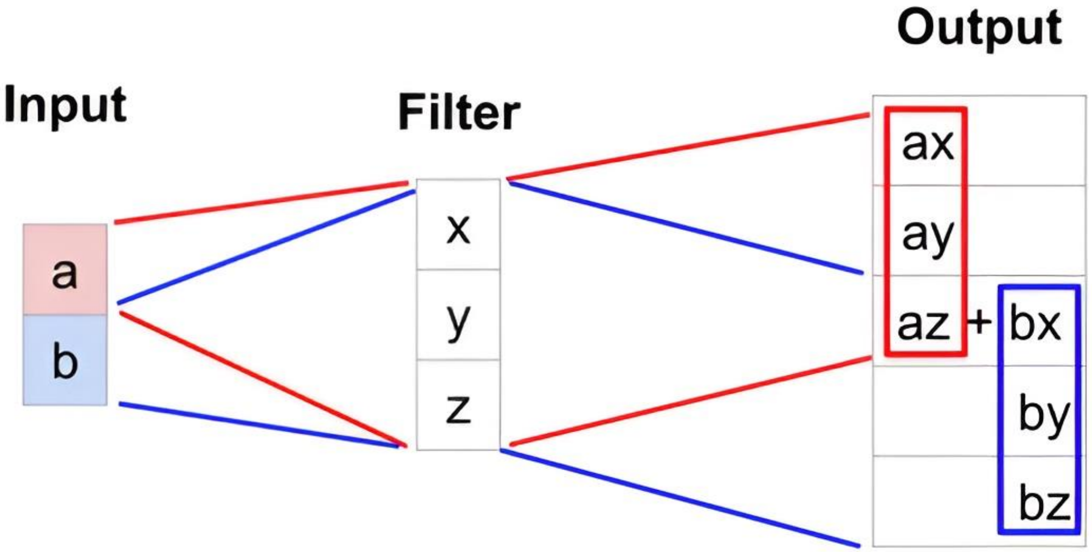
ax
ay
az + bx
by
bz

$$\boldsymbol{output\ size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example
- For filter sizes which produce an overlap in the output feature map, the overlapping values are simply added together.
- Less common names: **Deconvolution**, **Fractionally strided convolution**, **Up convolution**, …



Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Normal Convolution

$n \times n$ *image* $\quad f \times f$ *filter*

padding $p \quad$ *stride* s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

Padding = 1 →

$$\begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ e \\ 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

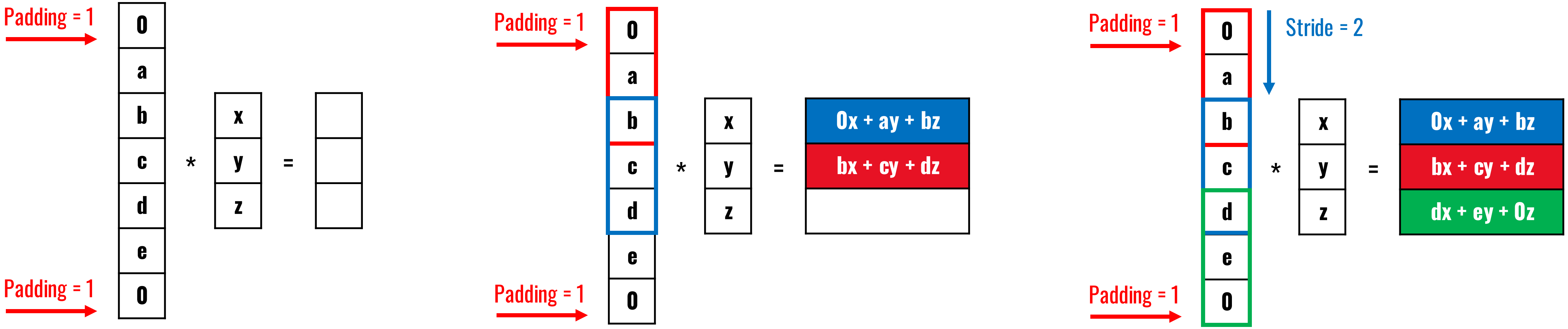
Padding = 1 →

Padding = 1 →

$$\begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ e \\ 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

Padding = 1 →

Normal Convolution



Padding = 1

Padding = 1

0
a
b
c
d
e
0

*

x
y
z

=

Padding = 1

Padding = 1

0
a
b
c
d
e
0

*

x
y
z

=

0x + ay + bz
bx + cy + dz

Padding = 1

Padding = 1

Stride = 2

0
a
b
c
d
e
0

*

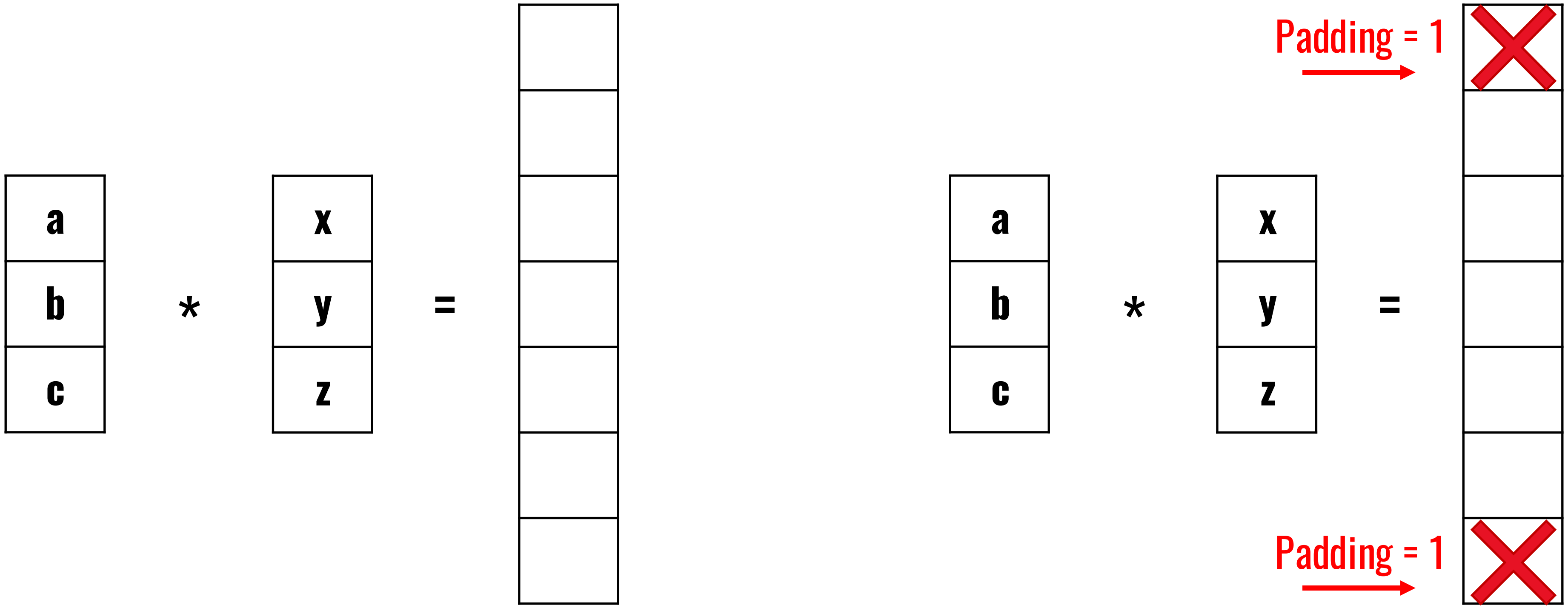
x
y
z

=

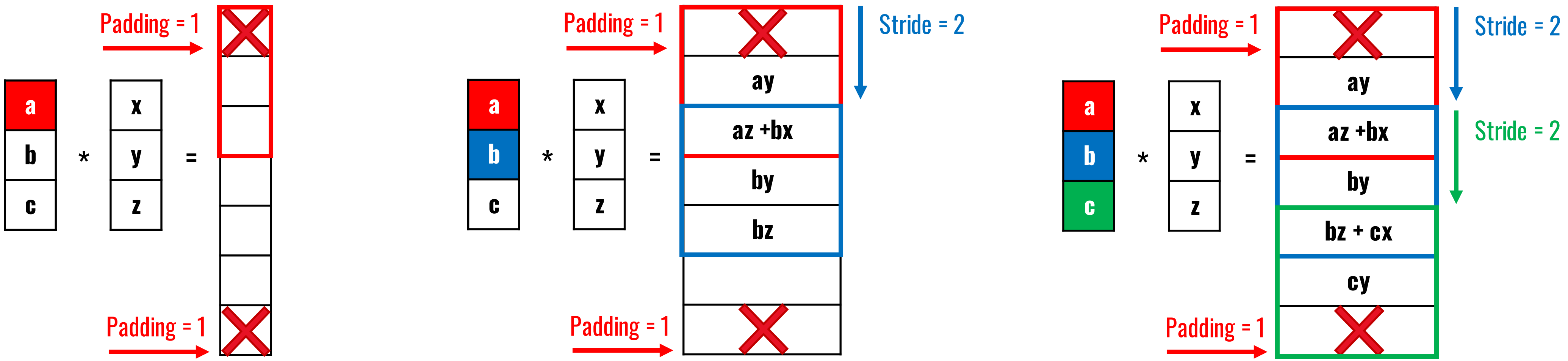
0x + ay + bz
bx + cy + dz
dx + ey + 0z

Transposed Convolution

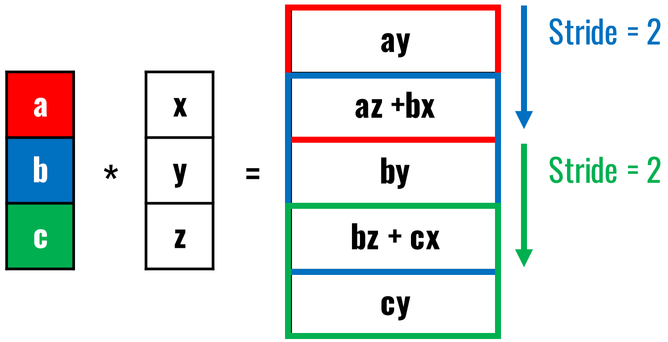
$$\boldsymbol{output\ size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

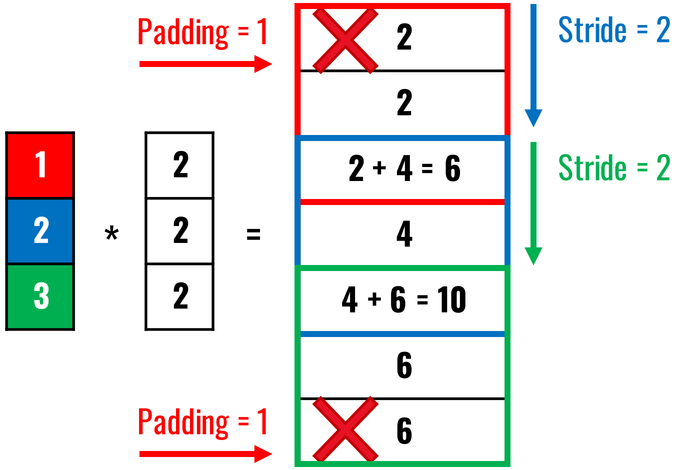


Transposed Convolution



Transposed Convolution





Summary on 1D

Convolution

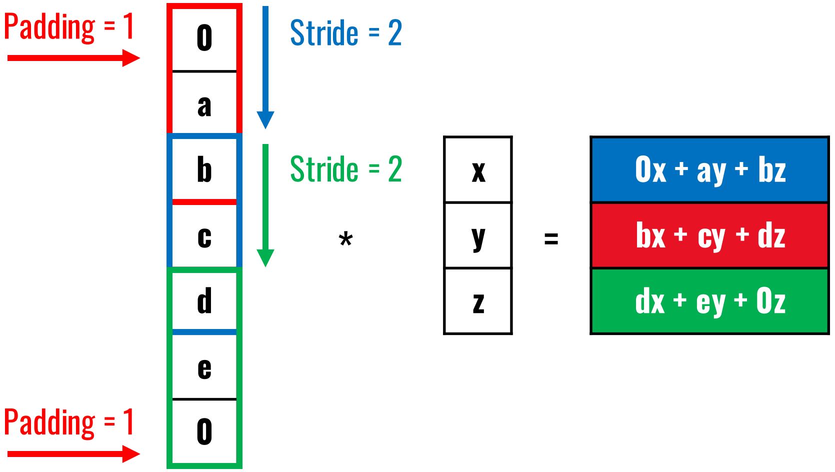
VS

Transposed Convolution

Input size: 5

Filter size: 3
Stride: 2
Padding: 1

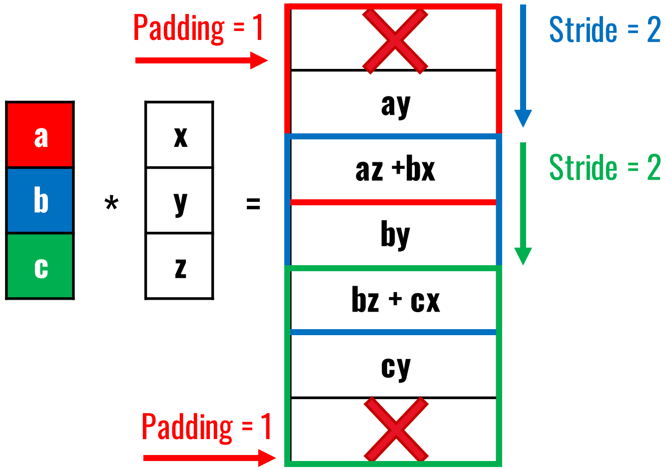
Output size: 3



Input size: 3

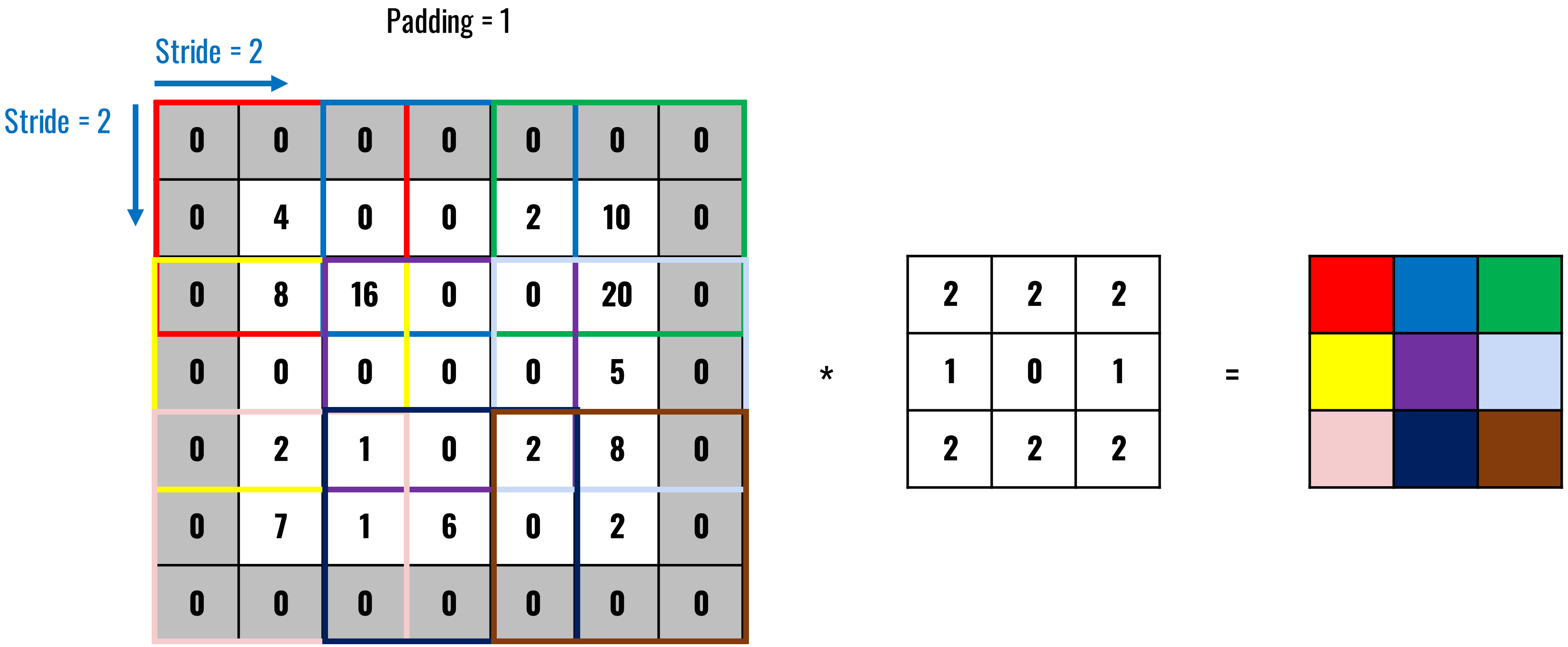
Filter size: 3
Stride: 2
Padding: 1

Output size: 5



Convolution

- 2D example



Padding = 1

Stride = 2

Stride = 2

0	0	0	0	0	0	0
0	4	0	0	2	10	0
0	8	16	0	0	20	0
0	0	0	0	0	5	0
0	2	1	0	2	8	0
0	7	1	6	0	2	0
0	0	0	0	0	0	0

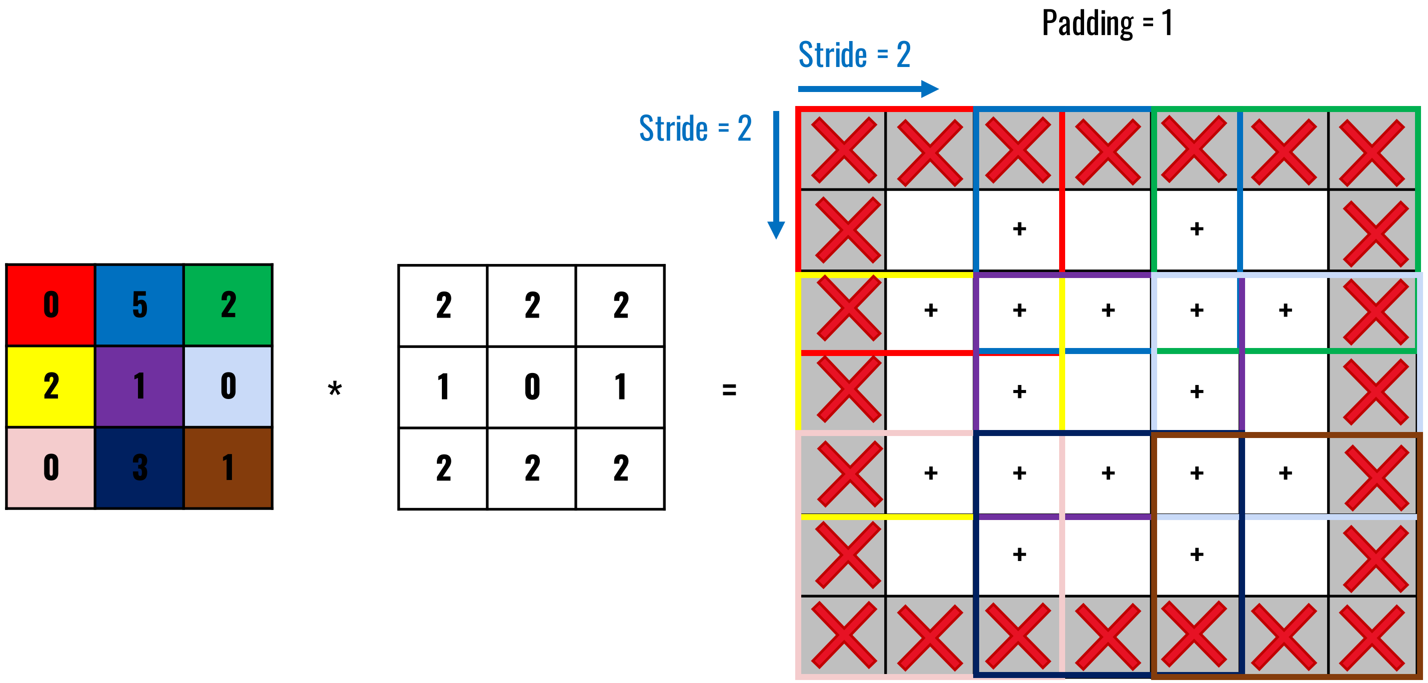
*

2	2	2
1	0	1
2	2	2

=

Transposed Convolution

- 2D example



0	5	2
2	1	0
0	3	1

*

2	2	2
1	0	1
2	2	2

=

✗	✗	✗	✗	✗	✗	✗
✗		+		+		✗
✗	+	+	+	+	+	✗
✗		+		+		✗
✗	+	+	+	+	+	✗
✗		+		+		✗
✗	✗	✗	✗	✗	✗	✗

Interactive Jupyter Notebook available on Canvas

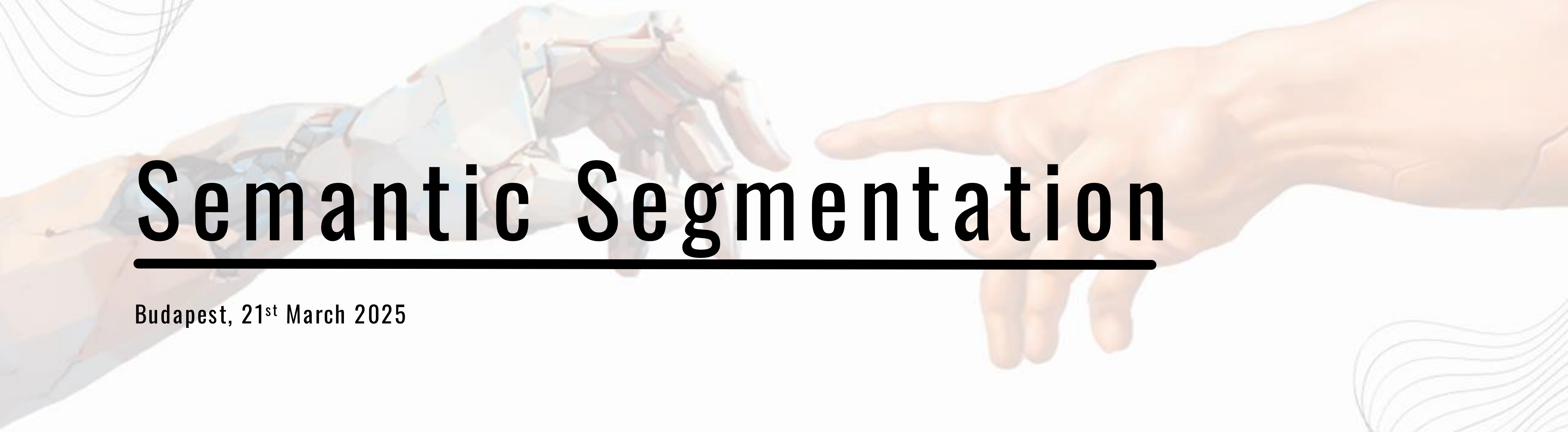
```
$jupyter nbconvert <notebook_name>.ipynb --to slides --post serve
```

Interactive Explanation on HuggingFace

https://huggingface.co/spaces/PercibalBuxus/TransposedConvolutions

Deep Network Development

Lecture 6.



Semantic Segmentation

Budapest, 21st March 2025

1 Upsampling

2 Semantic Segmentation

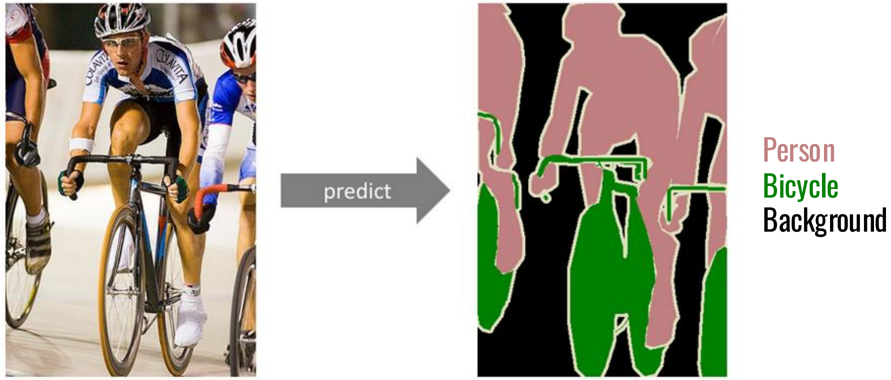
3 Instance Segmentation

Semantic Segmentation

Semantic image segmentation is the task of **labelling each pixel** of an image with a corresponding **class** of what is being represented.

Input: RGB image (height × width × 3) or a grayscale (height × width × 1)

Output: a segmentation map (height × width × 1), where each pixel contains a class label represented as an integer.



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

Input

Semantic Labels

Multiclass Classification (Recap)

Output of the neural network is a K long vector

How should we encode the ground-truth values?

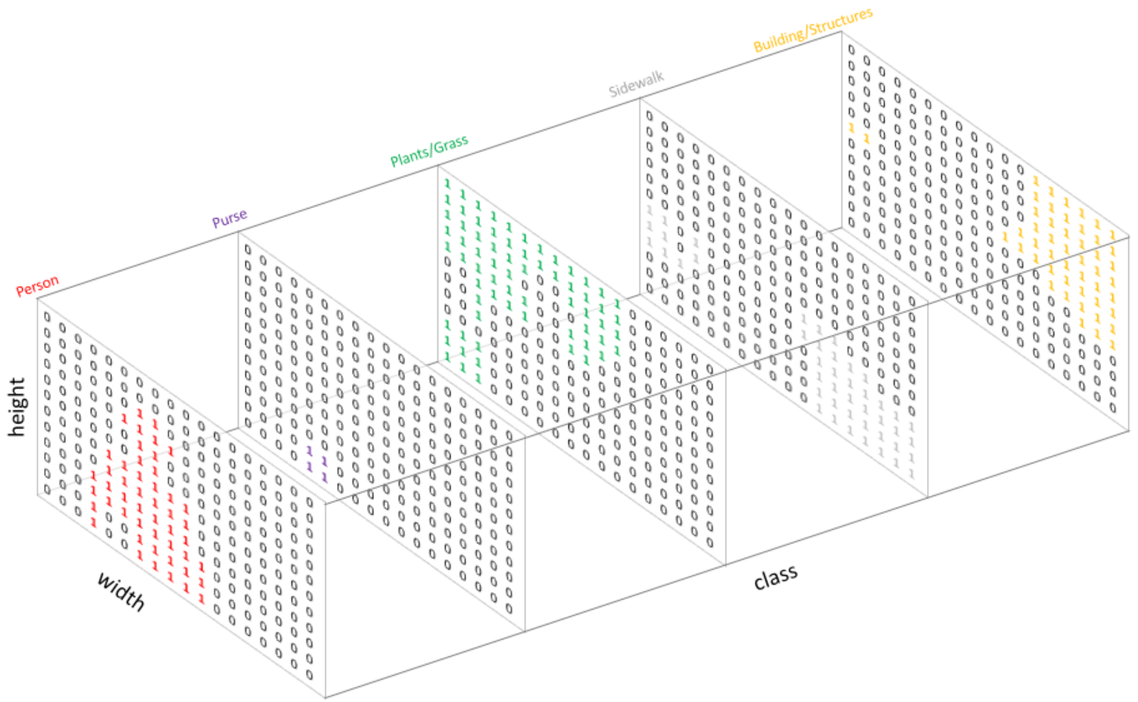
One-hot encoding (K=3):

(cat) 1 -> [1, 0, 0]
(dog) 2 -> [0, 1, 0]
(horse) 3 -> [0, 0, 1,]

Just as h(x): values between 0 and 1, sum up to 1

Semantic Segmentation

Similar to how we treat standard categorical values, we'll create our **target** by one-hot encoding the class labels - essentially creating **an output channel for each of the possible classes**.



Depth-wise argmax

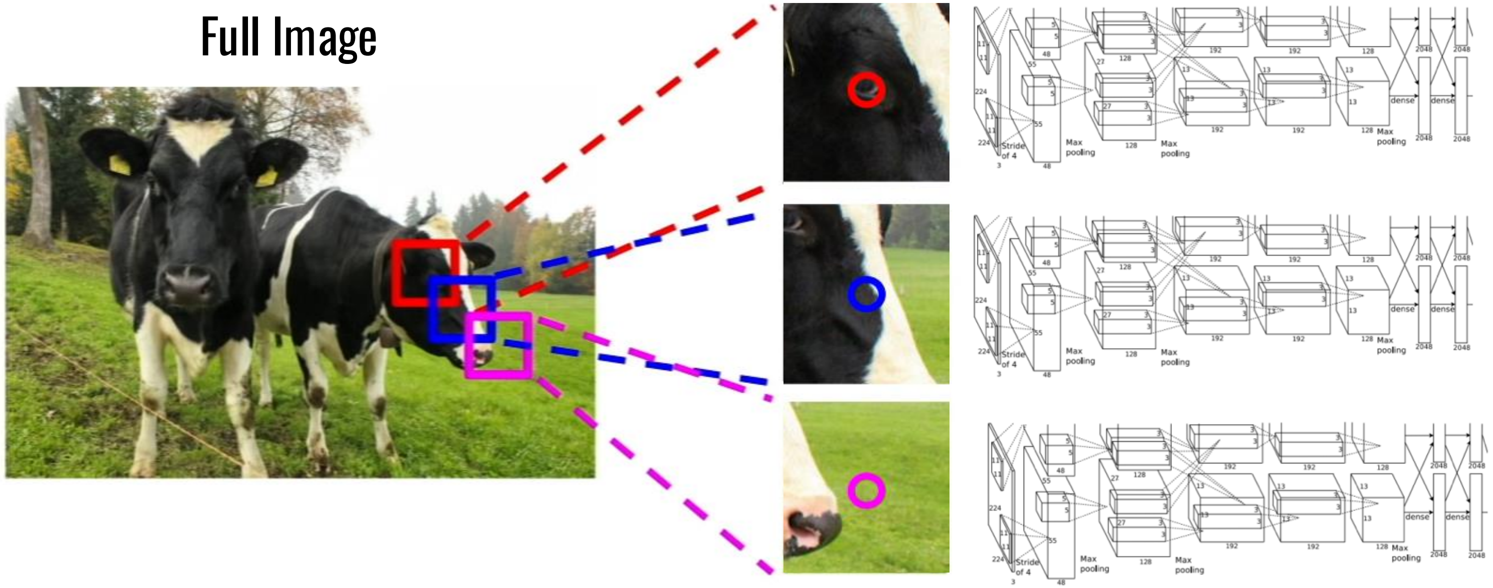
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1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

A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.

How to solve it? The naïve approach:

- Sliding Windows + Classification
  - Computationally expensive
  - Not reusing shared features



Full Image

Extract patch

Classify center pixel with CNN

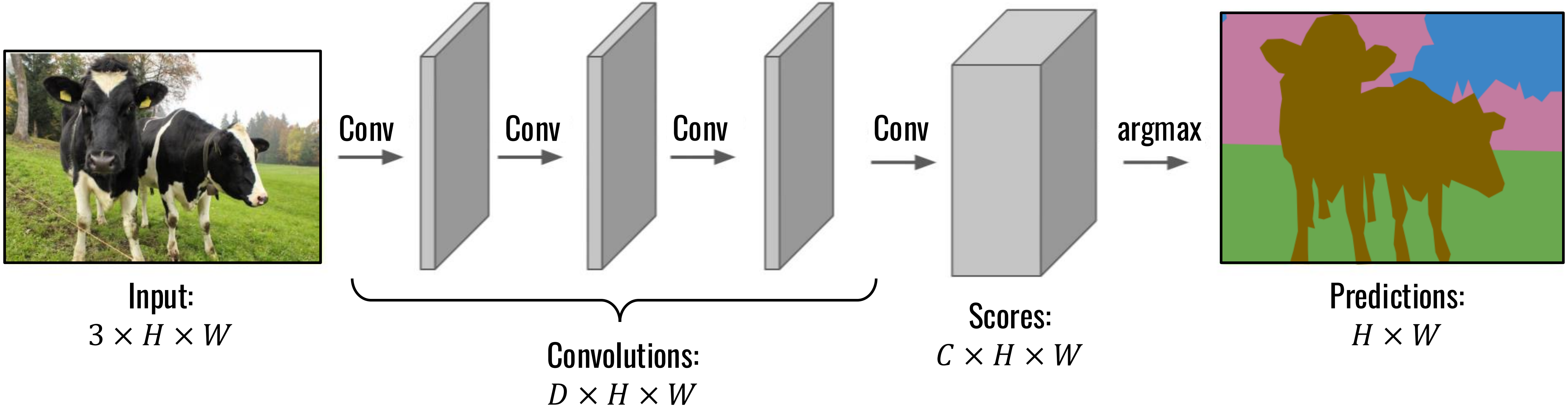
Cow

Cow

Grass

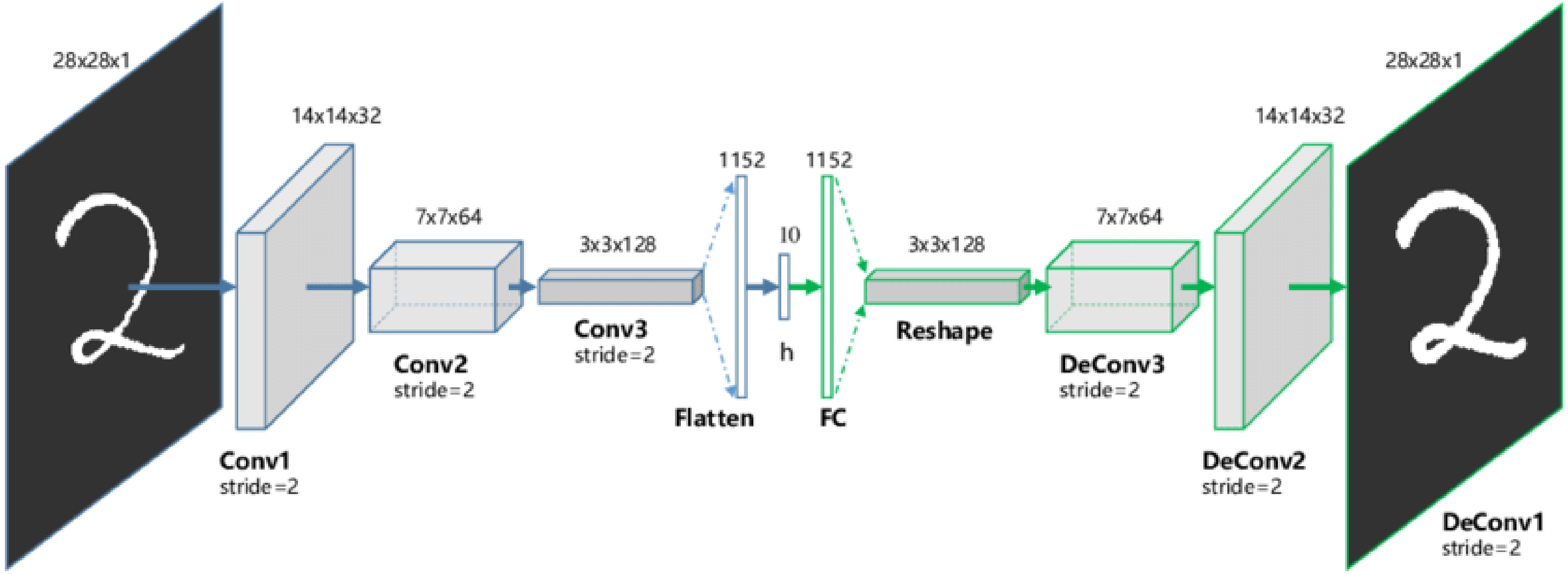
How to solve it? The naïve approach:

- Apply convolutions for all pixels at once, keeping original resolution
    - Computationally expensive
    - Does not enforce network to learn key features. It only learns a direct mapping from input pixels to the segmentation pixels.



Convolutional Autoencoders

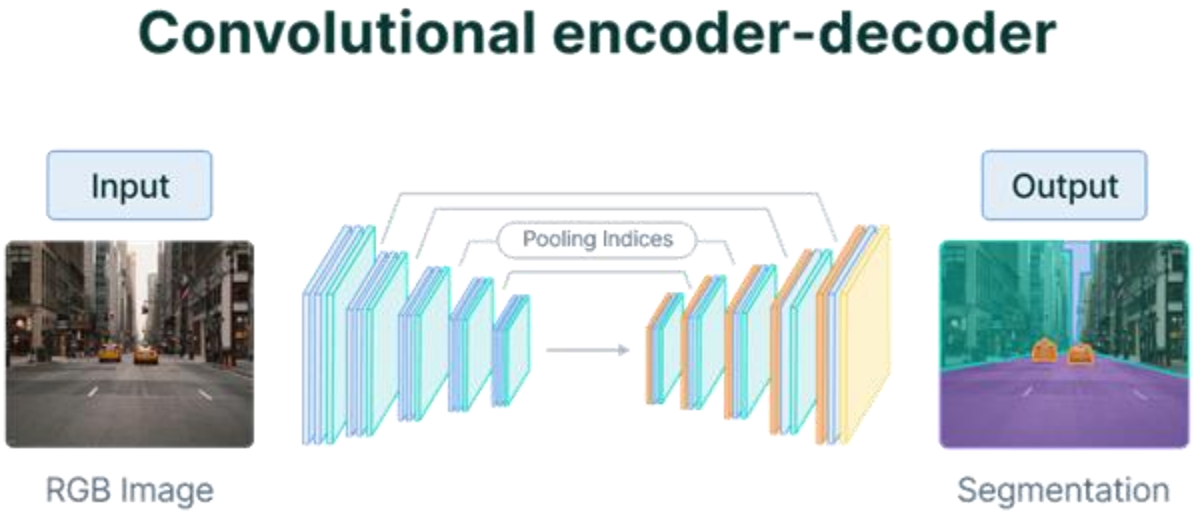
- Specifically designed for **image data**.
- They employ convolutional layers in both the encoder and decoder parts of the network.
- This architecture allows them to **capture spatial dependencies and hierarchical features** effectively.
- The reconstruction of the input image is often blurry and of lower quality due to compression during which information is lost.



Convolutional Autoencoders

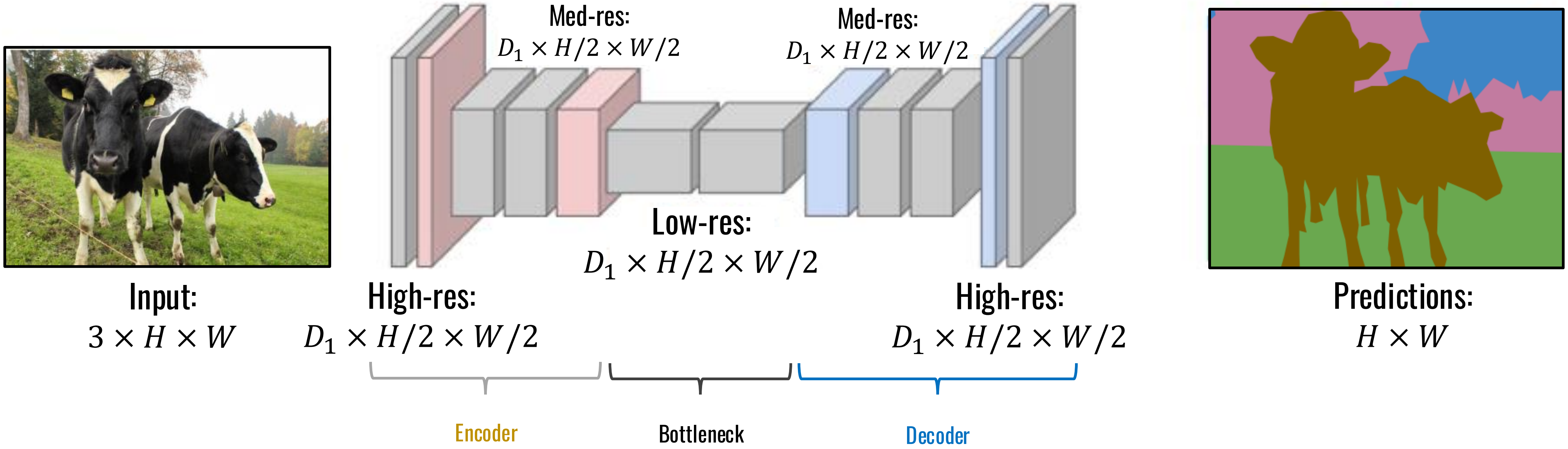
Image Segmentation

- Image segmentation is the process of **partitioning** an image **into multiple segments** each belonging to a class.
- The goal is to simplify and/or change the representation of an image by **grouping pixel values according to the class they belong to**.



Encoder-Decoder Structure

- One popular approach for image segmentation models is to follow an **encoder**/**decoder** **structure** where we **downsample** the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and then **upsample** the feature representations into a full-resolution segmentation map.



Input: $3 \times H \times W$

High-res: $D_1 \times H/2 \times W/2$

Med-res: $D_1 \times H/2 \times W/2$

Low-res: $D_1 \times H/2 \times W/2$

Med-res: $D_1 \times H/2 \times W/2$

High-res: $D_1 \times H/2 \times W/2$

Predictions: $H \times W$

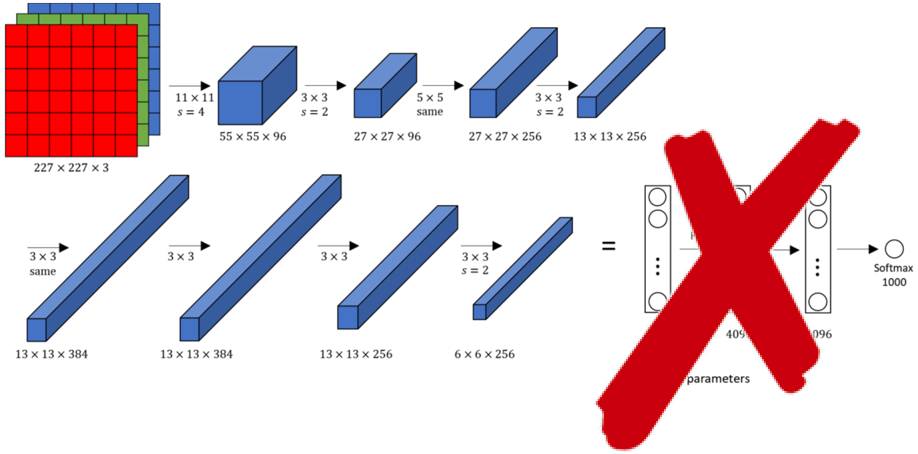
Encoder

Bottleneck

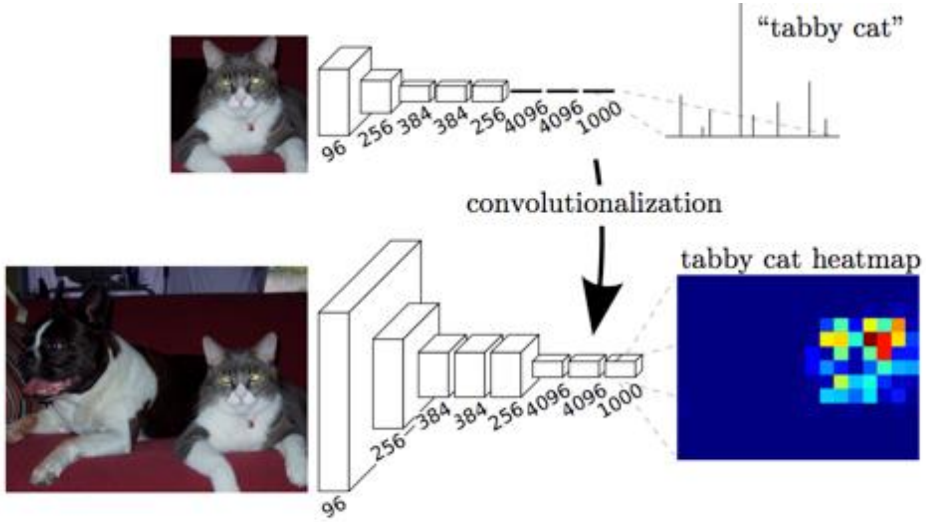
Decoder

Fully Convolutional Network (FCN) [1]

The approach of using a *"Fully Convolutional"* network trained end-to-end, pixels-to-pixels for the task of image segmentation was introduced by Long et al. in late 2014. The paper's authors propose adapting existing, well-studied **image classification** networks (e.g. **AlexNet**) to serve as the encoder module of the network, appending a decoder module with transpose convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.



Original AlexNet model

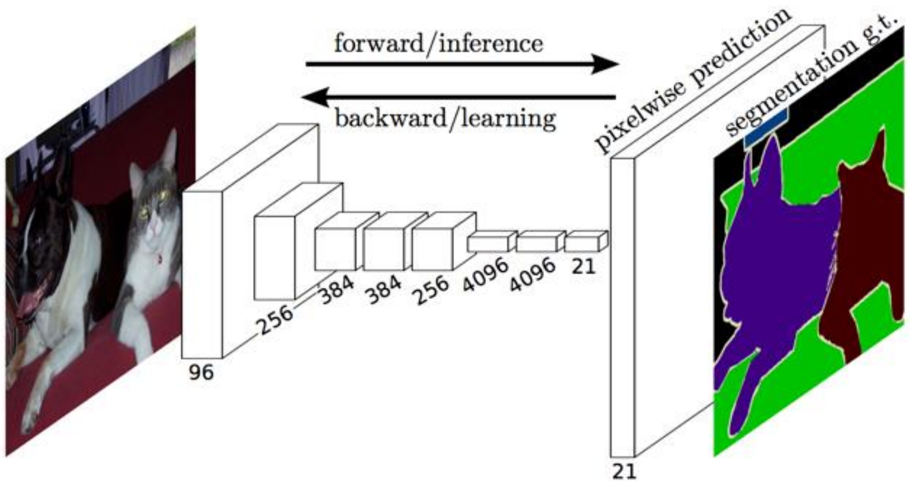


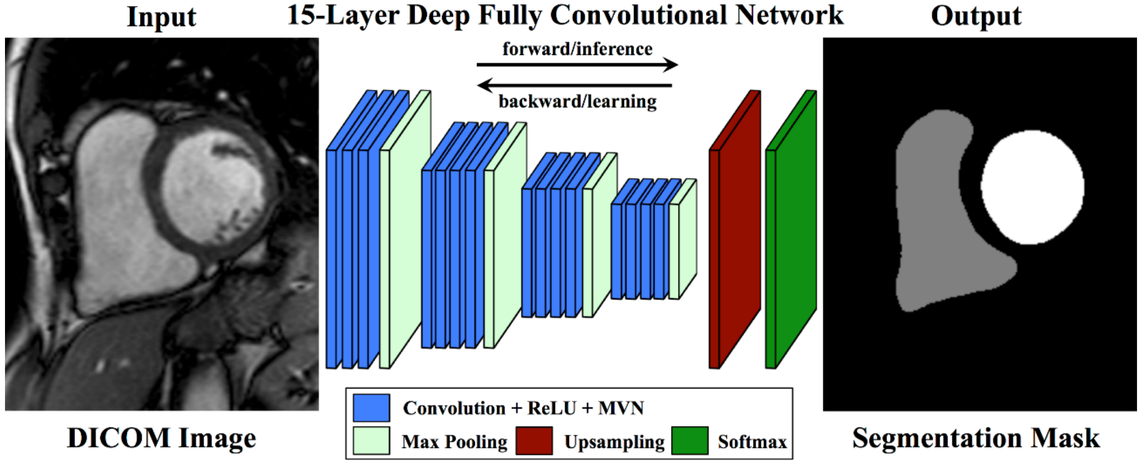
Repurposed AlexNet model

The encoder produces a coarse feature map which is then refined by the decoder module.

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.

Fully Convolutional Network (FCN) [1]





However, because the encoder module reduces the resolution of the input by a factor of 32, the decoder module **struggles to produce fine-grained segmentations**.

Ground truth target

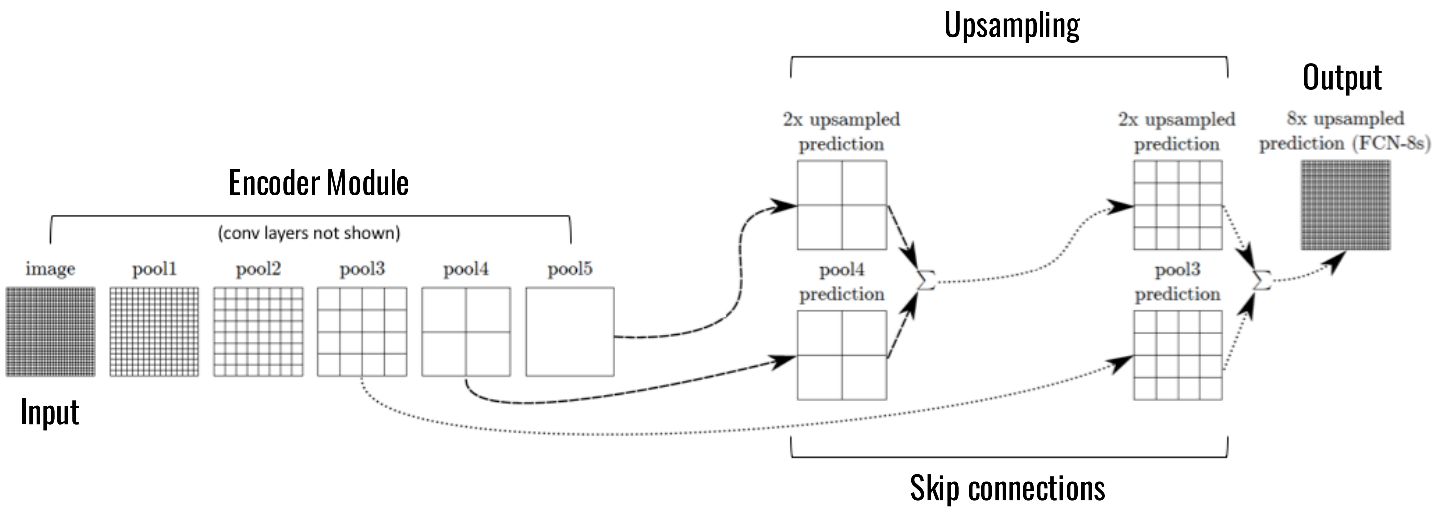
Predicted segmentation

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Fully Convolutional Network (FCN) [1]

- **Adding skip connections** - The authors address this tension by slowly upsampling (in stages) the encoded representation, adding "skip connections" from earlier layers, and summing these two feature maps.

- These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail to reconstruct accurate shapes for segmentation boundaries. Indeed, we can recover more fine-grain detail with the addition of these skip connections.



Upsampling

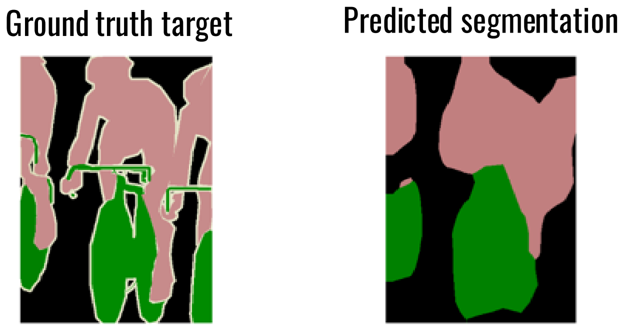
Output

Encoder Module

Input

Skip connections

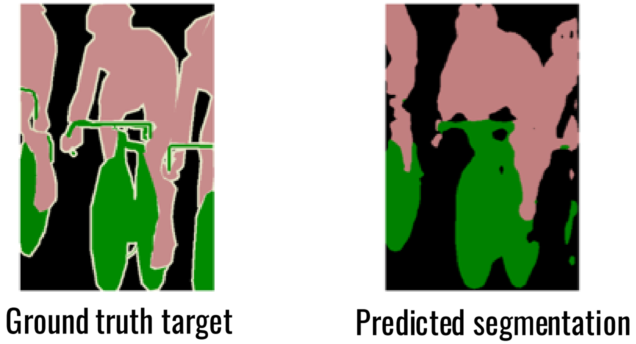
Before



Ground truth target

Predicted segmentation

with skip connections



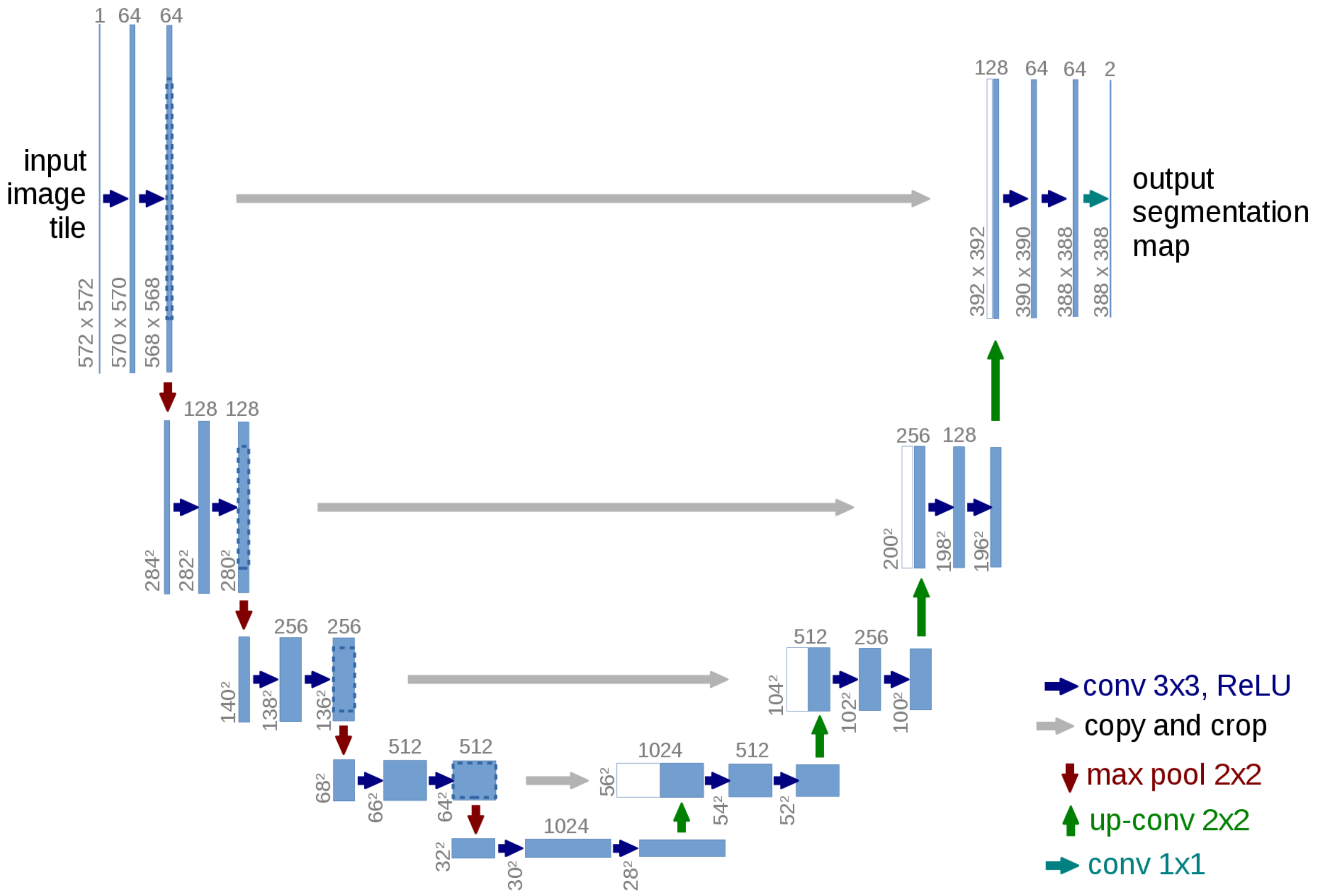
Ground truth target

Predicted segmentation

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.

U-Net: Convolutional Networks for Biomedical Image Segmentation [2]

- Ronneberger et al. improve upon the "fully convolutional" architecture primarily through ***expanding the capacity of the decoder*** module of the network. More concretely, they propose the **U-Net architecture** which "consists of a contracting path to capture context and a ***symmetric*** expanding path that enables precise localization."

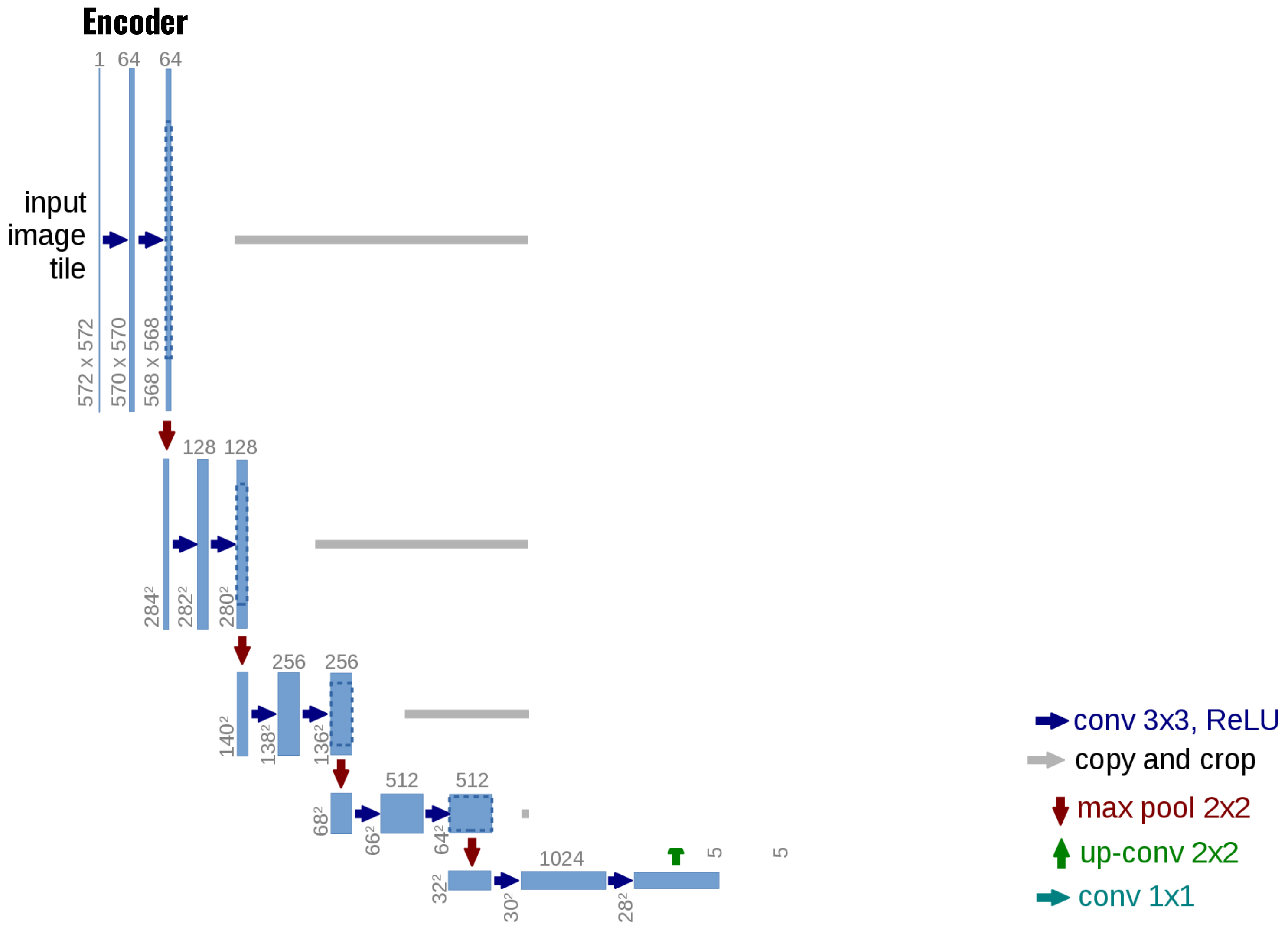


---

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer International Publishing, 2015.

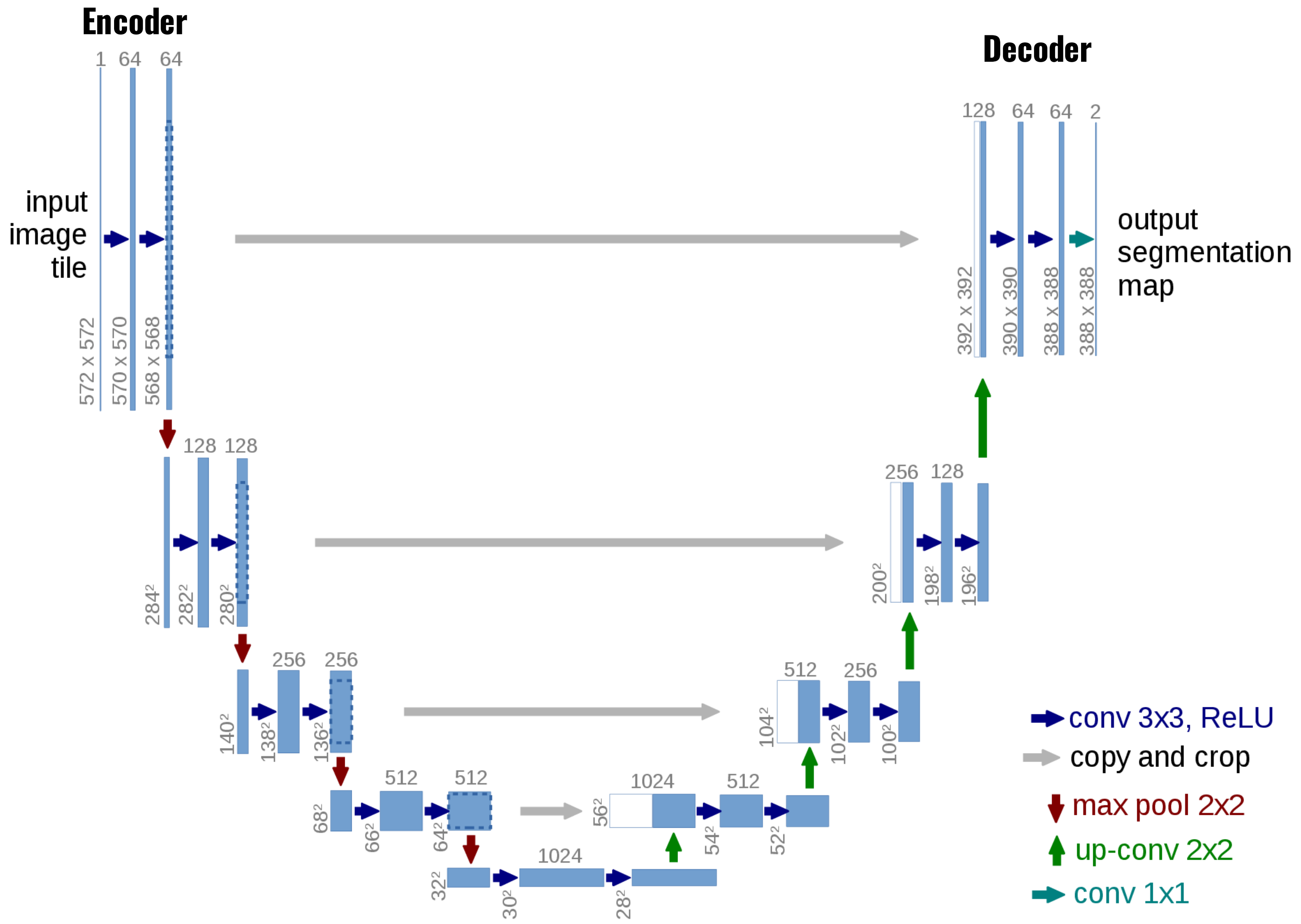
2. Semantic Segmentation

U-Net



2. Semantic Segmentation

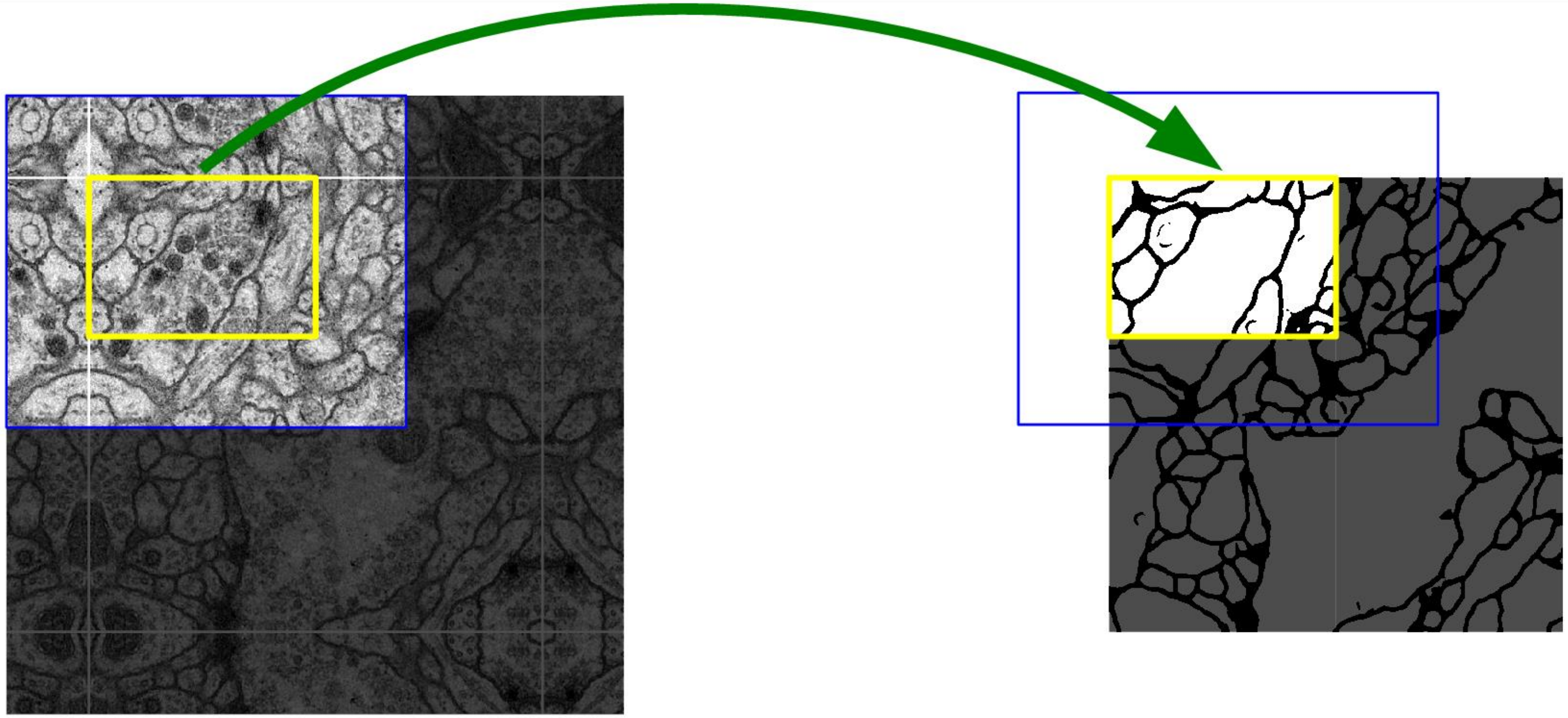
U-Net



U-Net

How can we use it for images with arbitrary size?

- Do the segmentation for smaller regions of the image
- On the edges mirror the image



Architectures

Advanced U-Net variants

- The standard U-Net model consists of a series of convolution operations for each "block" in the architecture. These convolutional blocks can be replaced for more advanced ones such as:
    - ResNet blocks
    - Inception modules
    - Dense blocks
    - Etc.

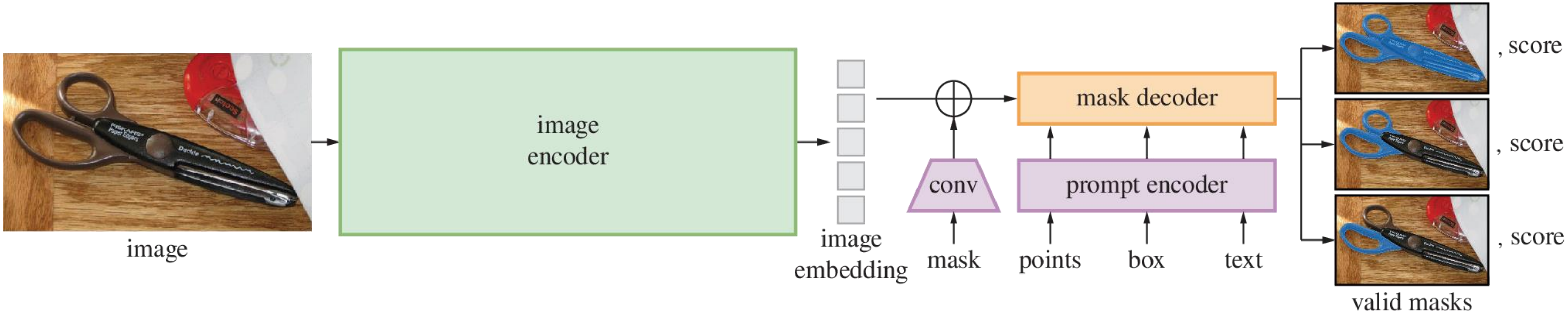
DeepLab

- Deeplab from a group of researchers from Google have proposed a multitude of techniques to improve the existing results and get finer output at lower computational costs. The 3 main improvements suggested as part of the research are:
    - Atrous convolutions
    - Atrous Spatial Pyramidal Pooling
    - Conditional Random Fields usage for improving final output

DeepLab v3: https://arxiv.org/abs/1706.05587v3

Segment Anything Model (SAM)

- Encoder-Decoder architecture
- The focus is on interactive segmentation based on generalized mask prediction
- Incorporates the user selected point/box/text to the prediction
- https://segment-anything.com/demo#



[6] Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., … Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2408.00714

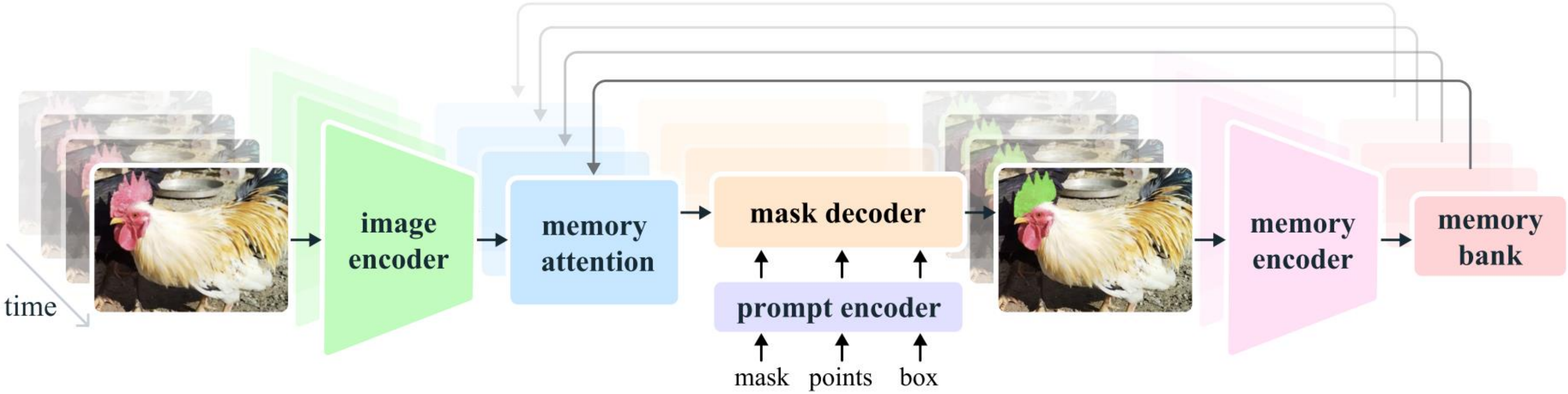
Segment Anything Model (SAM)

- The generated masks does not contain label prediction
- https://segment-anything.com/demo#

[6] Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., … Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2408.00714

Segment Anything Model (SAM 2)

- Extending SAM with memory to keep the masks over the whole video
- Images are considered as a single frame video
- iVOS/VOS
- https://sam2.metademolab.com/demo

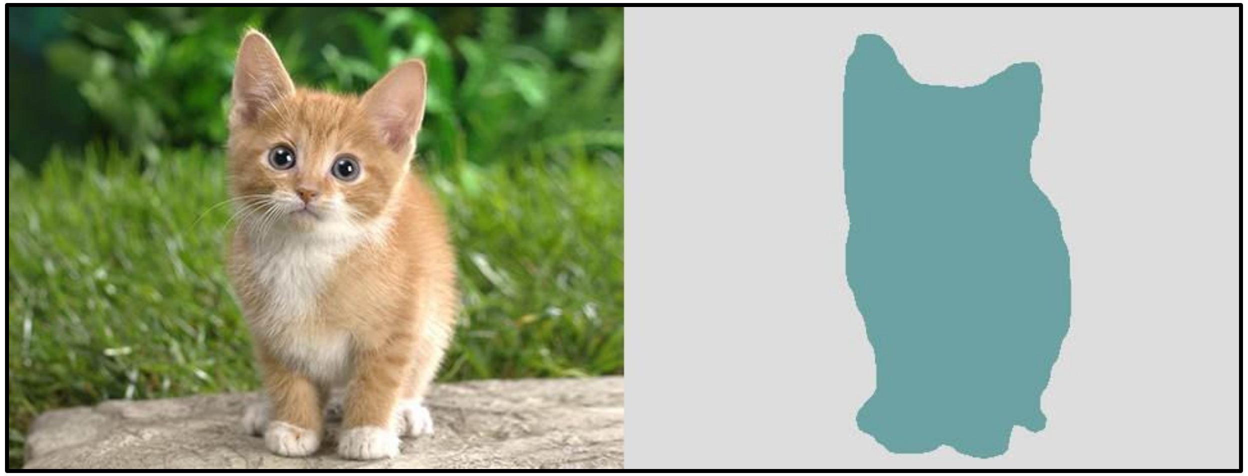


[7] Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., … Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2408.00714

Training

How to train such networks?

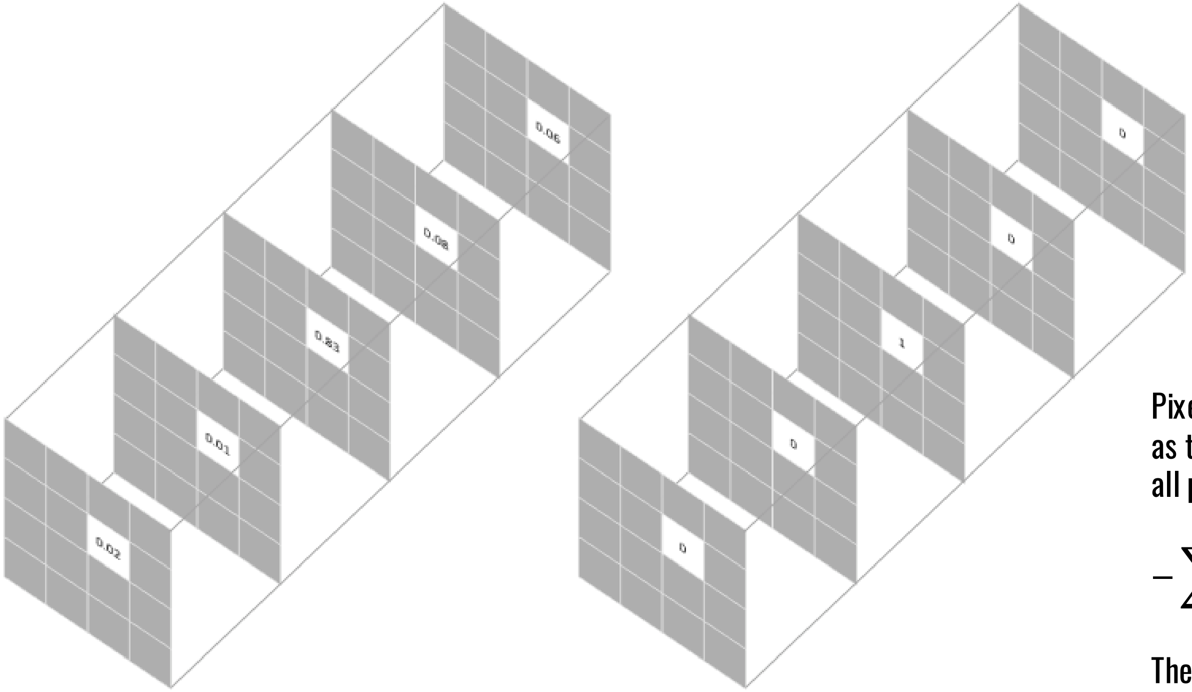
- Have input X (images) and labels Y (masks).
- Define architecture
- Set hyperparameters
- Define loss function and metrics



Losses

Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.
- Problematic for unbalanced classes



Prediction for a selected pixel

Target for the corresponding pixel

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

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

The scoring is repeated over all pixels and averaged

Losses

Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.
- Problematic for unbalanced classes

Losses

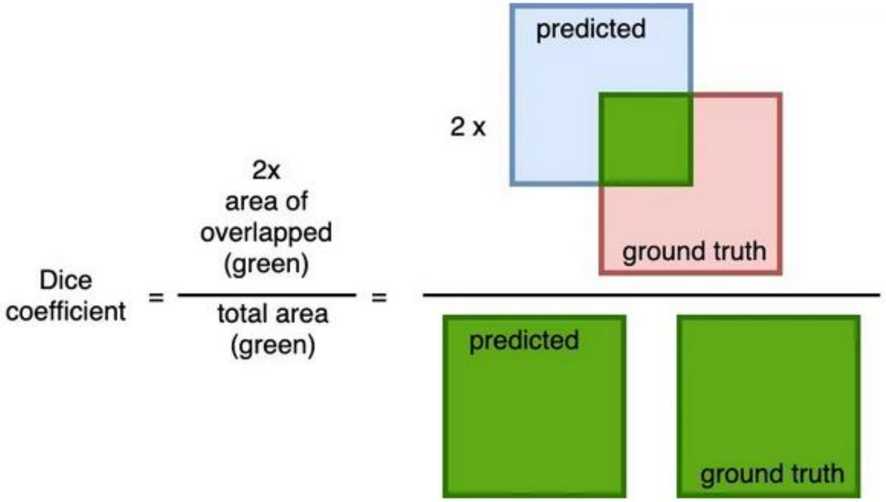
Dice Loss

- Another popular loss function for image segmentation tasks is based on the Dice coefficient, which is essentially a measure of overlap between two samples. This measure ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap. The Dice coefficient was originally developed for binary data, and can be calculated as:

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

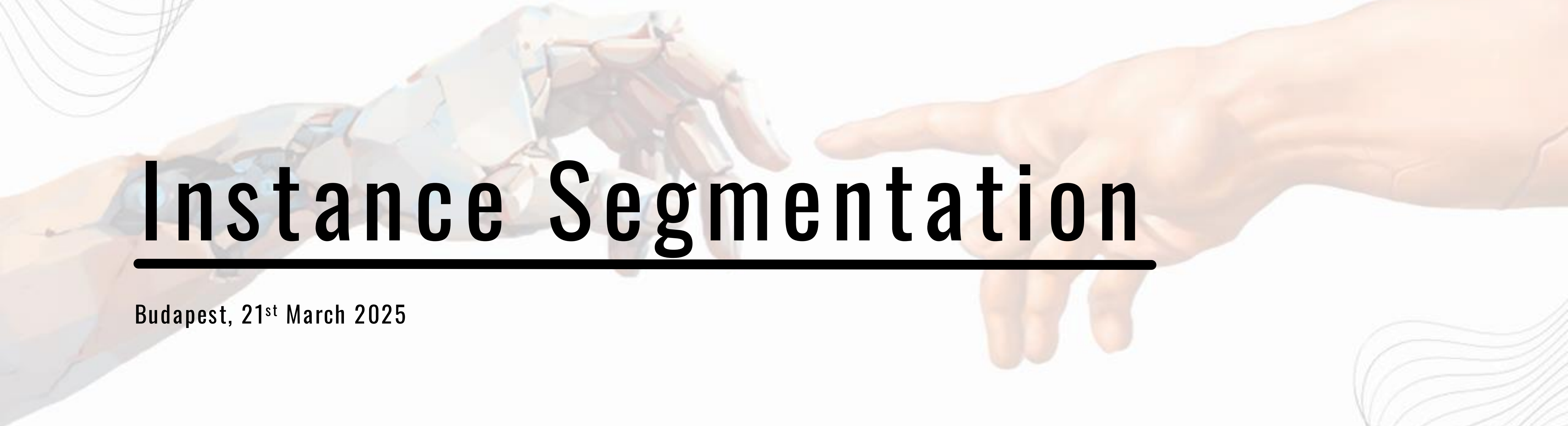
where:

- $|A \cap B|$ represents the common elements between sets A and B, and
- $|A|$ represents the number of elements in set A
- $|B|$ represents the number of elements in set B



Deep Network Development

Lecture 6.



Instance Segmentation

Budapest, 21st March 2025

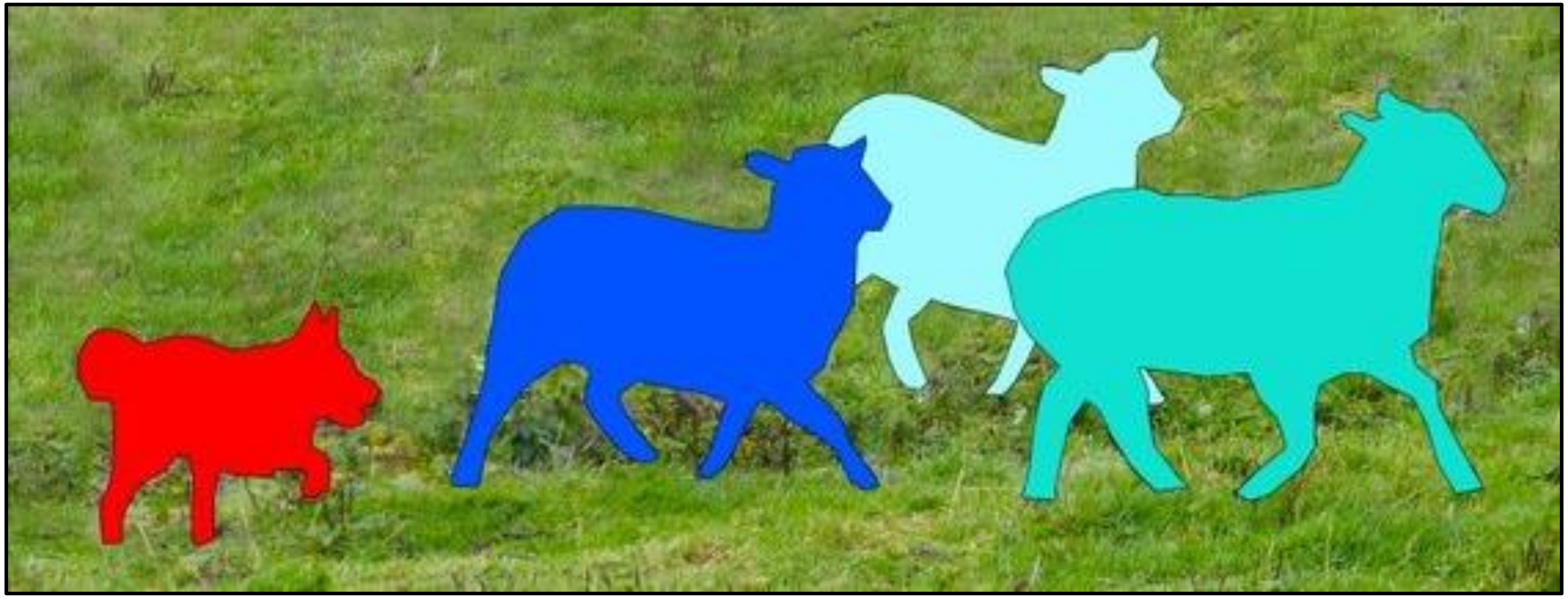
1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

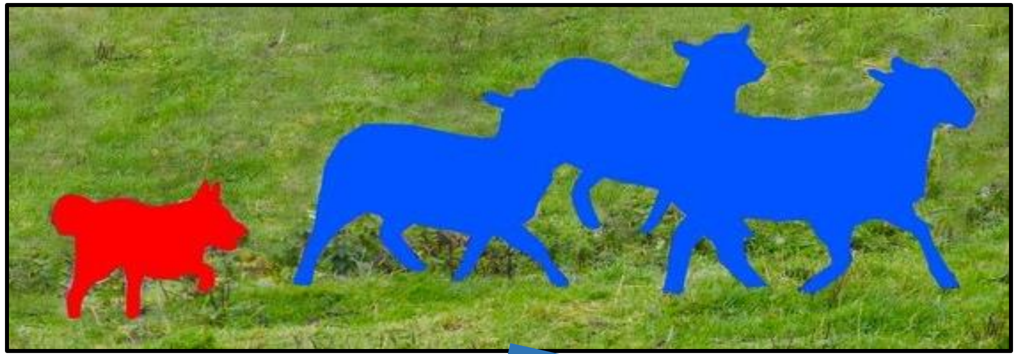
What is Instance Segmentation?

- Instance Segmentation is identifying each object instance for every known object within an image. It assigns a label to each pixel of the image.

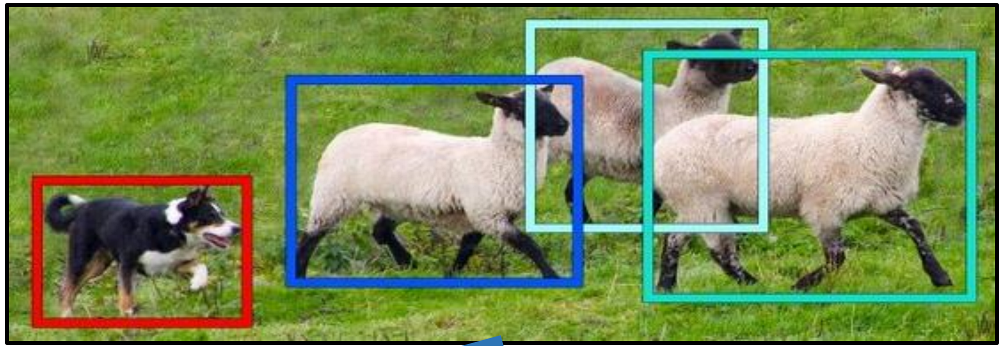


What is Instance Segmentation?

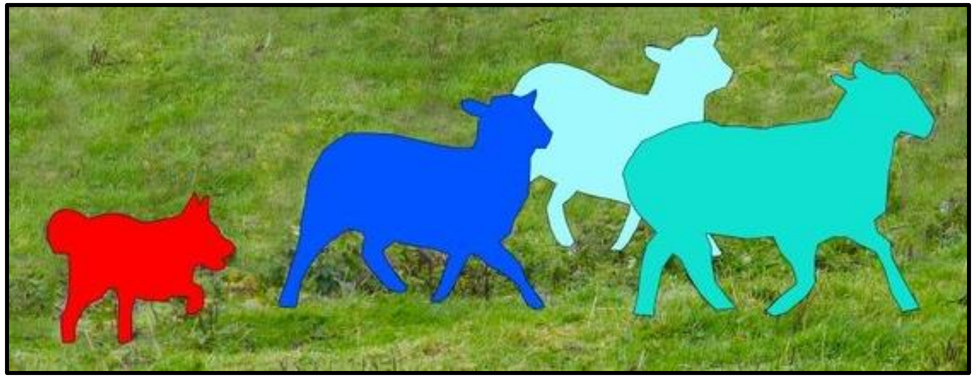
Semantic Segmentation: gives per-pixel labels, but merges instances.



Object Detection: detects individual object instances, but only gives boxes.



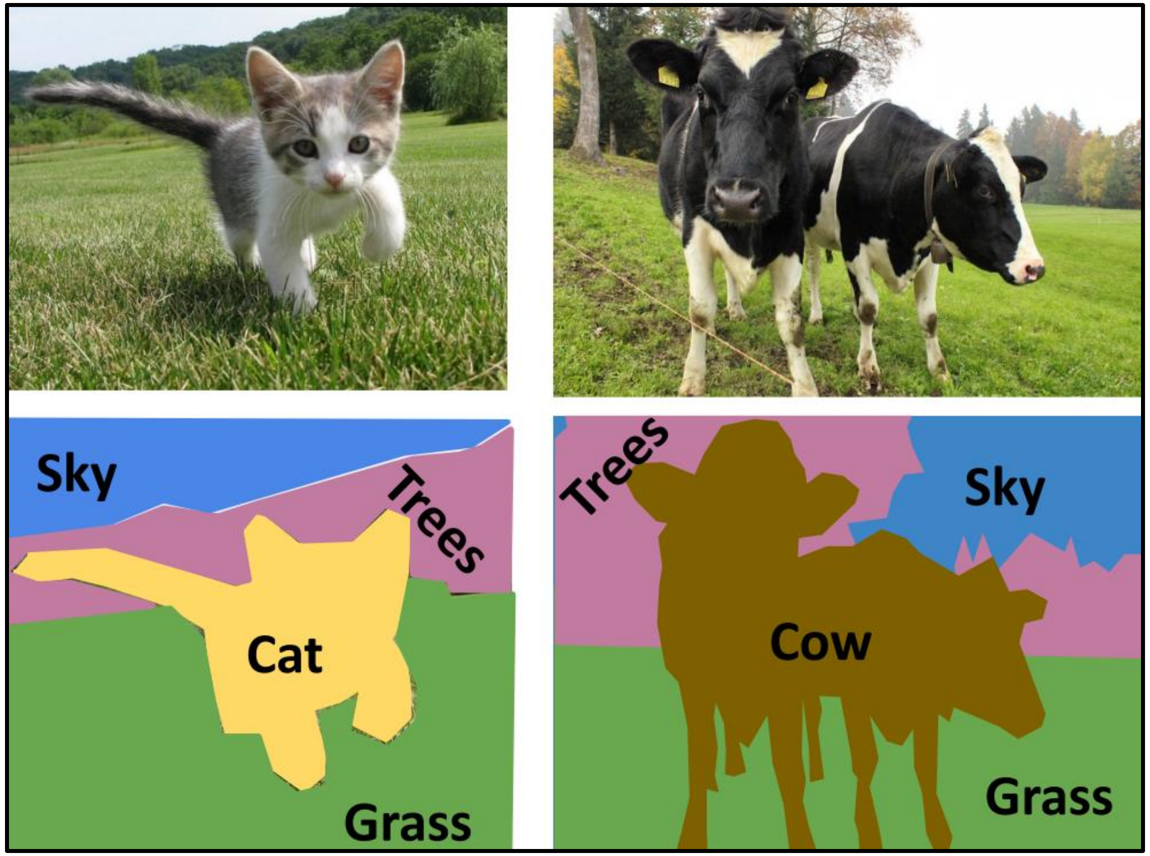
Instance Segmentation



What is Instance Segmentation?

Things and Stuff

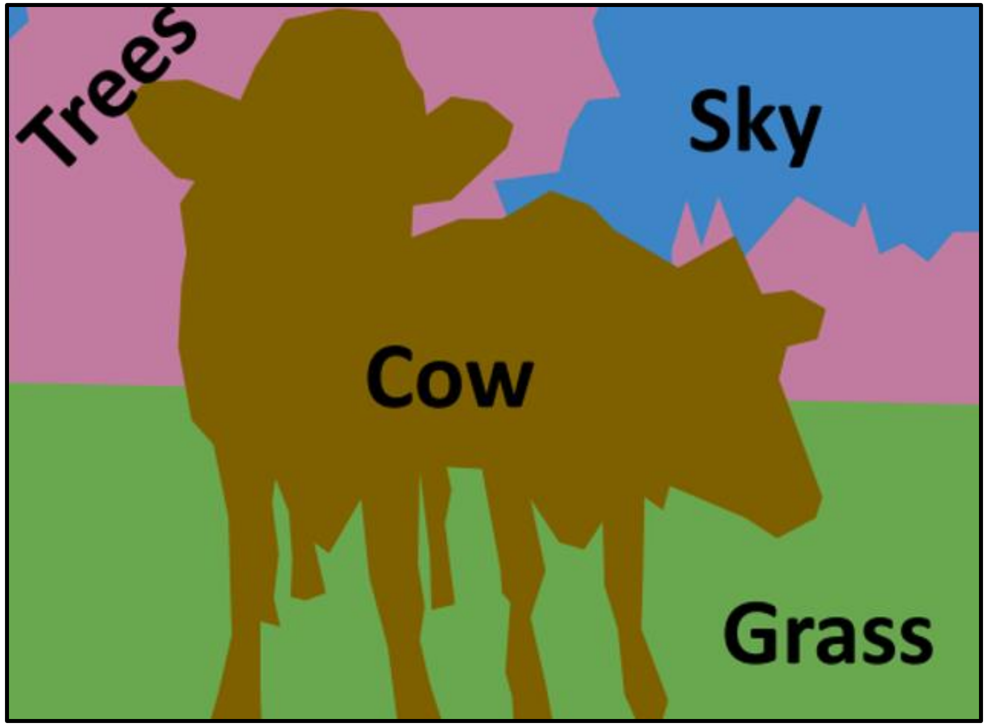
- **Things:** Object categories that can be separated into object instances (e.g. cats, cars, person)
- **Stuff:** Object categories that cannot be separated into instances (sky, grass, water, trees)

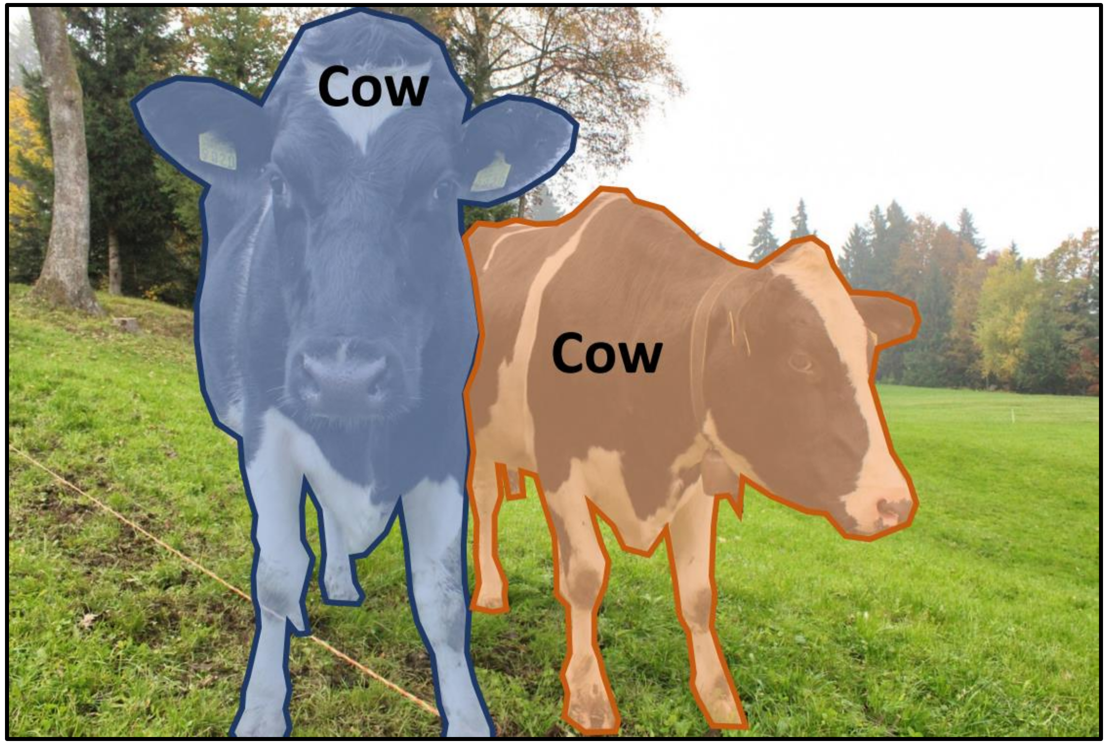


What is Instance Segmentation?

Semantic Segmentation: detects both objects and regions but doesn't distinguish individual instances.

Instance Segmentation: distinguishes individual object instances, but only for countable objects.



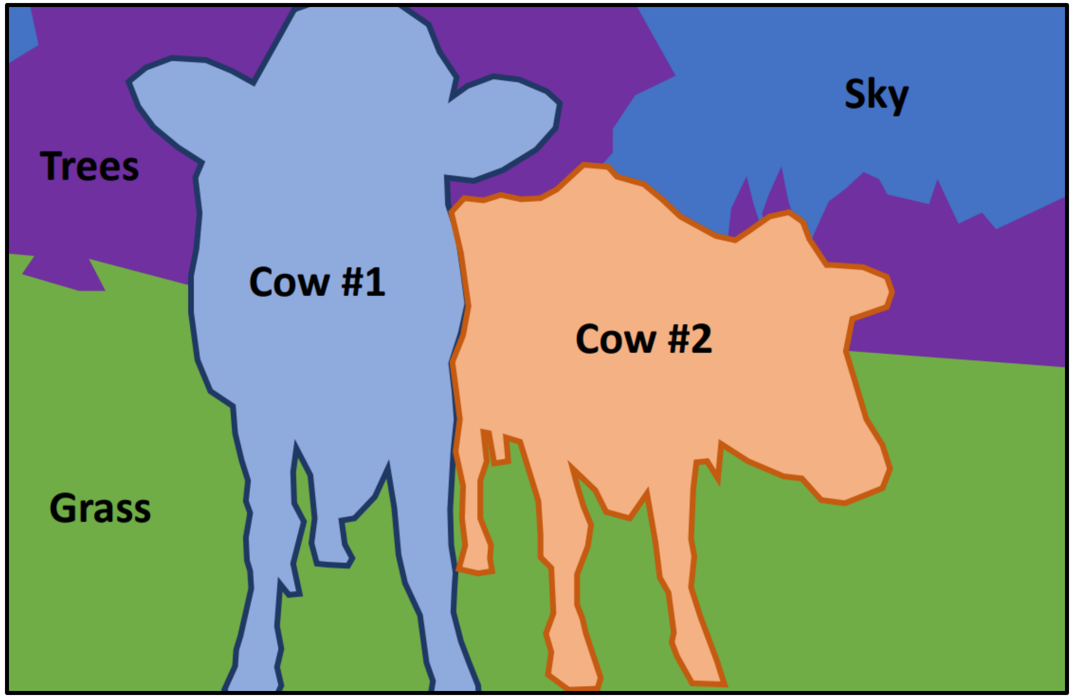


What is Instance Segmentation?

Beyond Instance Segmentation: Panoptic Segmentation [3]

Label all pixels in the image (both things and stuff).

For “thing” categories also separate into instances.



[3] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.00868

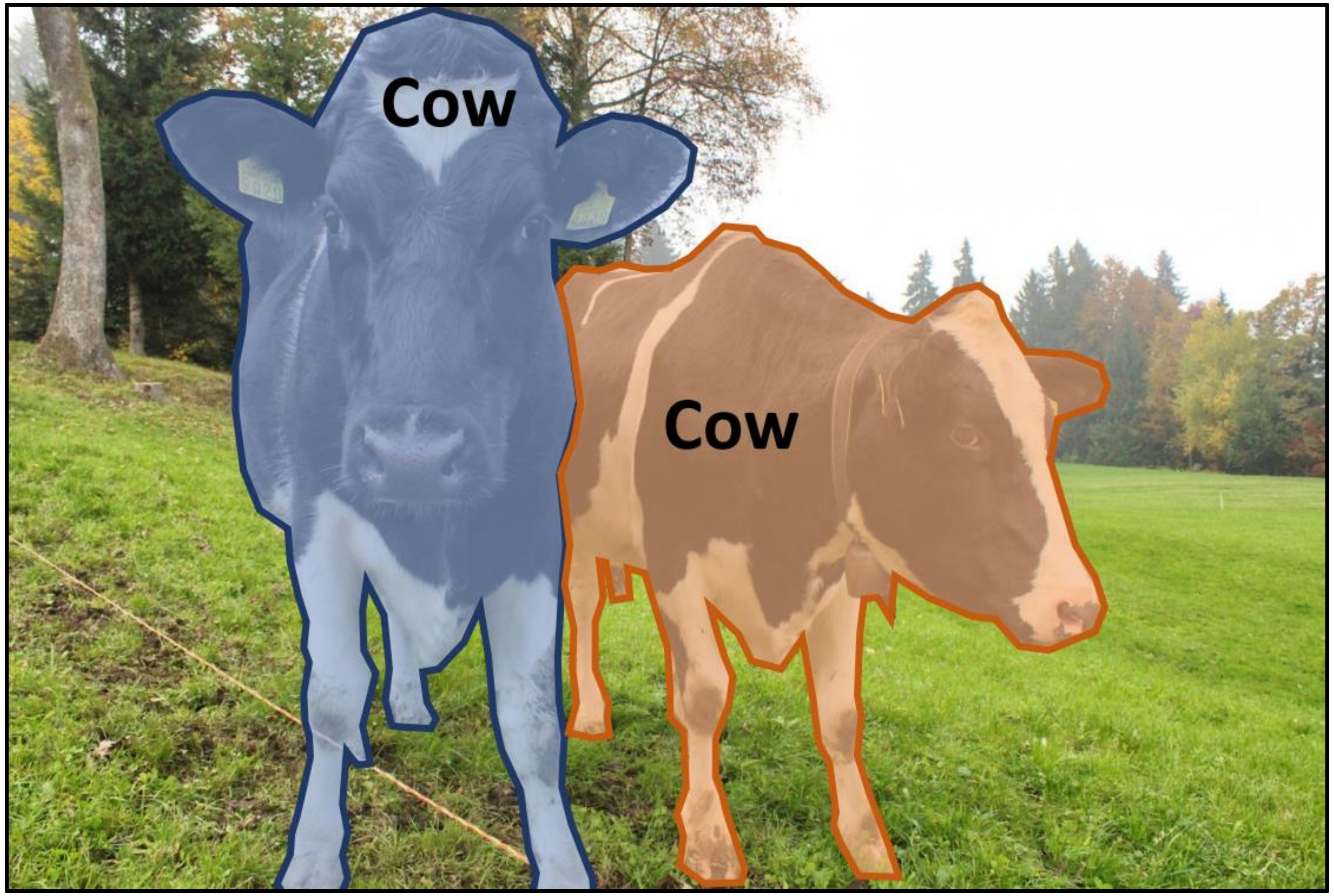
How does Instance Segmentation work?

Instance Segmentation:

- Detect all objects in the image and identify the pixels that belong to each object (Only things!)

Approach:

- Perform object detection, then predict a segmentation mask for each object!

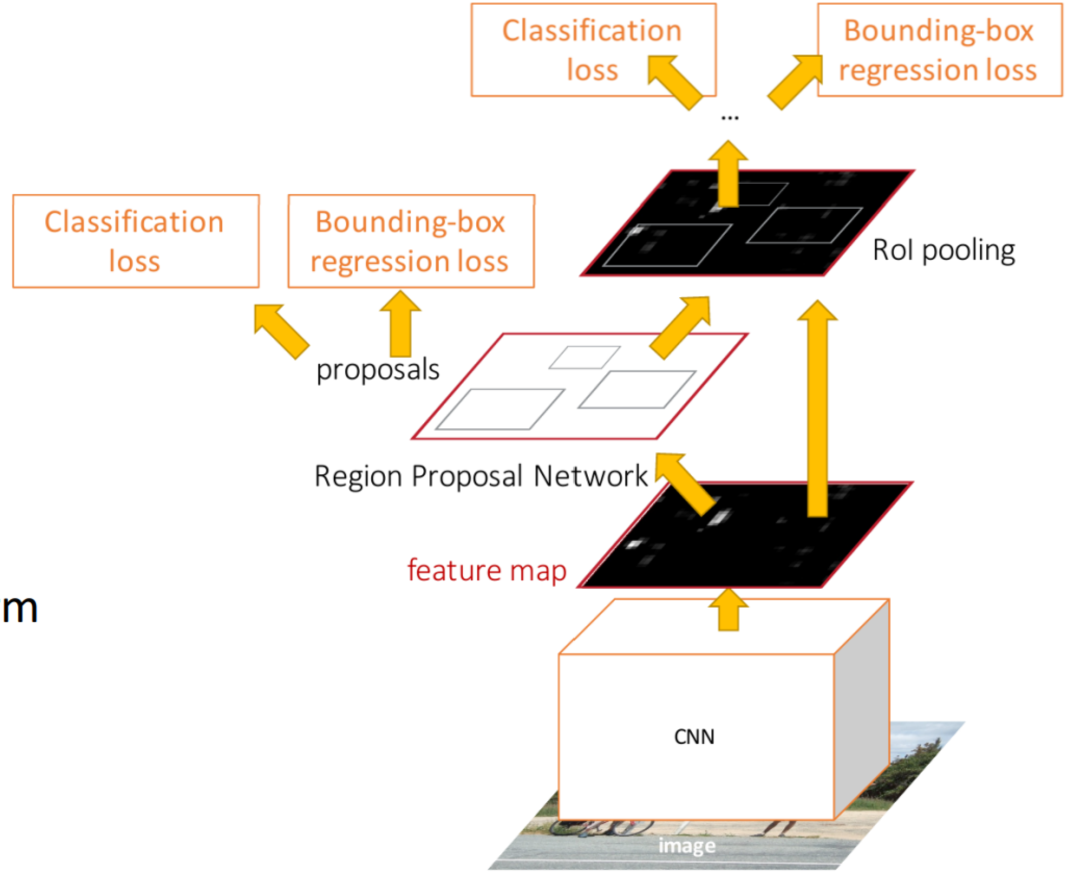


How does Instance Segmentation work?

Faster R-CNN: Learnable Region Proposals [4]

Jointly train with 4 losses:

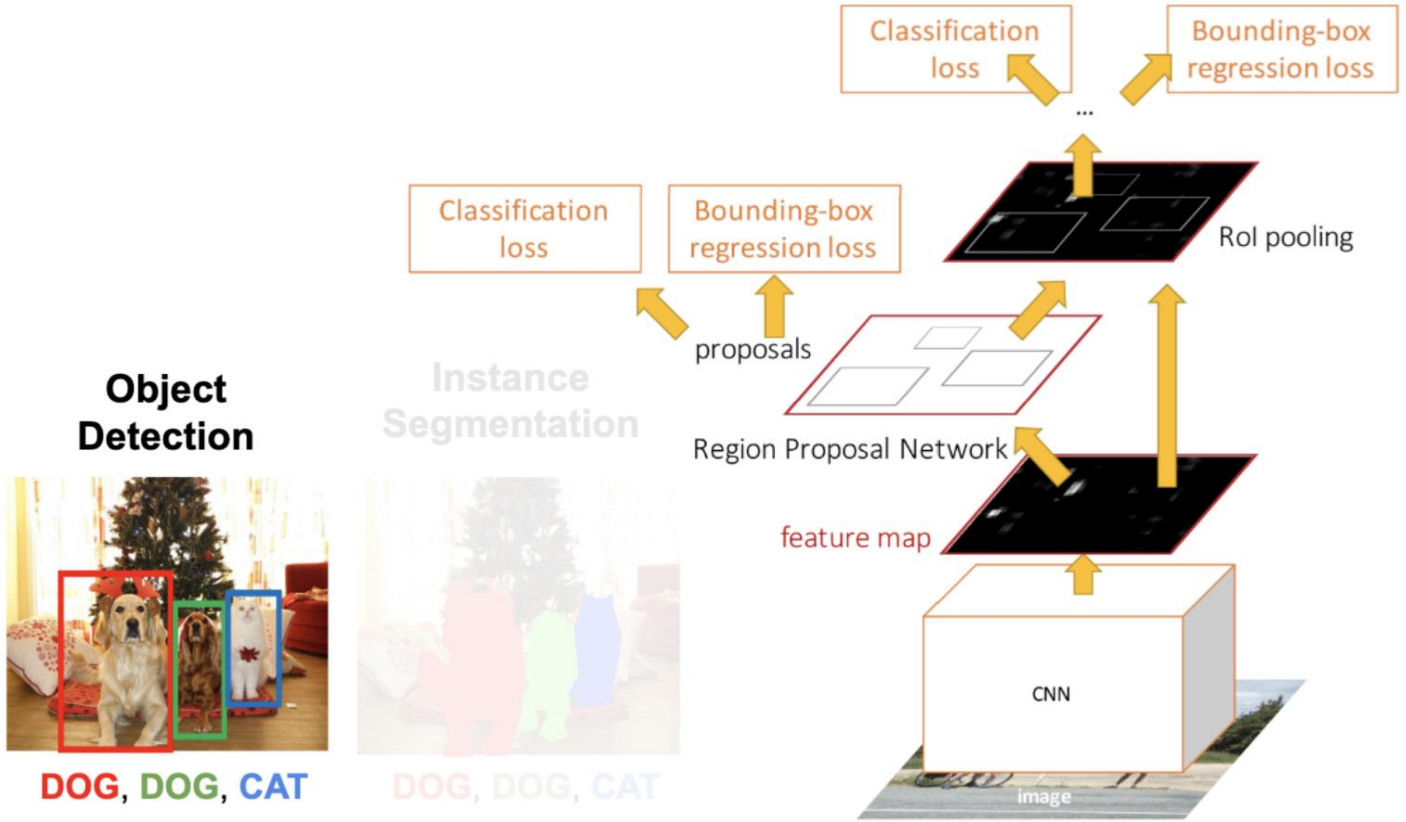
1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



[4] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

How does Instance Segmentation work?

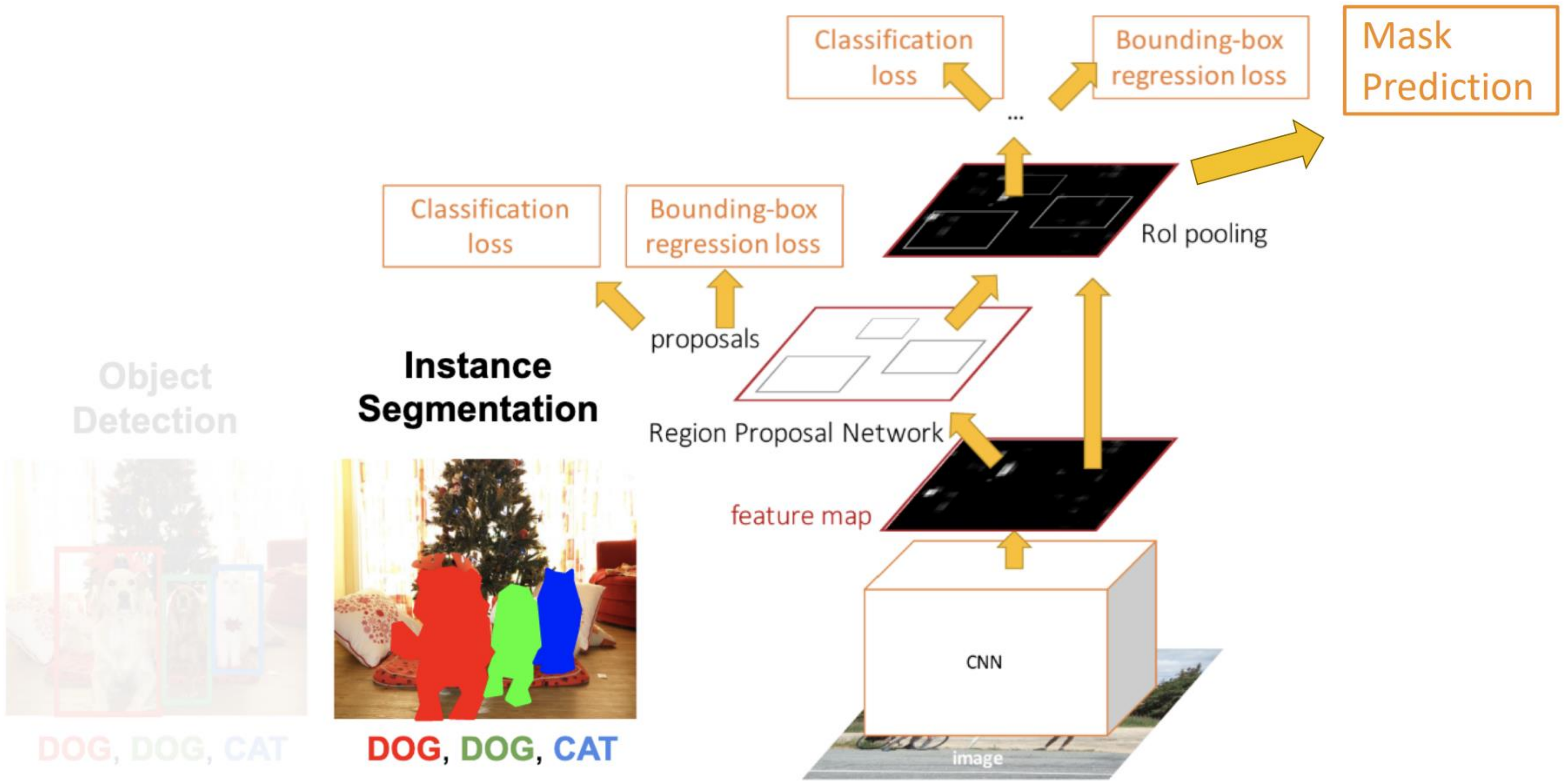
Object Detection: Faster R-CNN [4]



[4] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

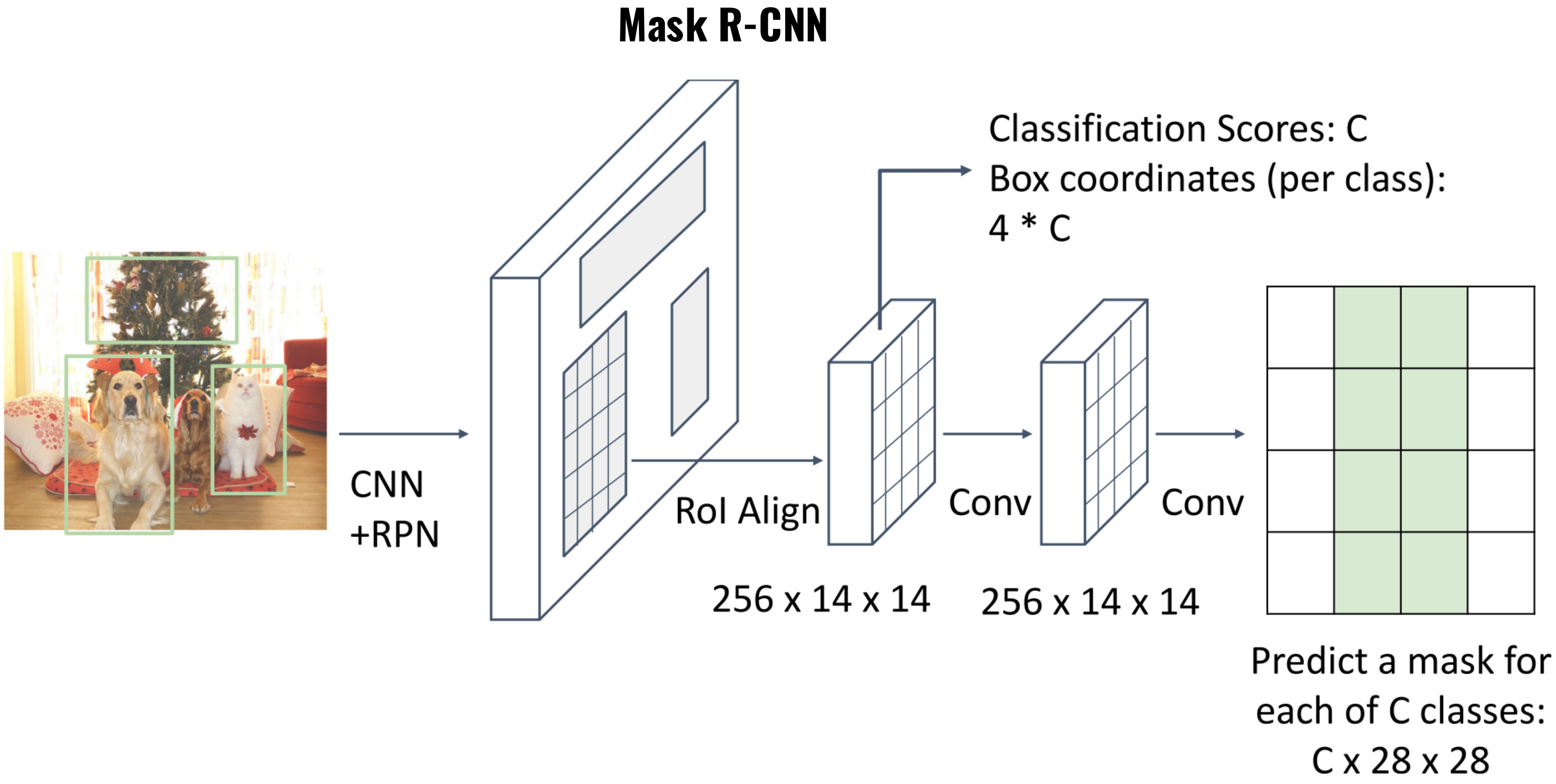
How does Instance Segmentation work?

Instance Segmentation: Mask R-CNN [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

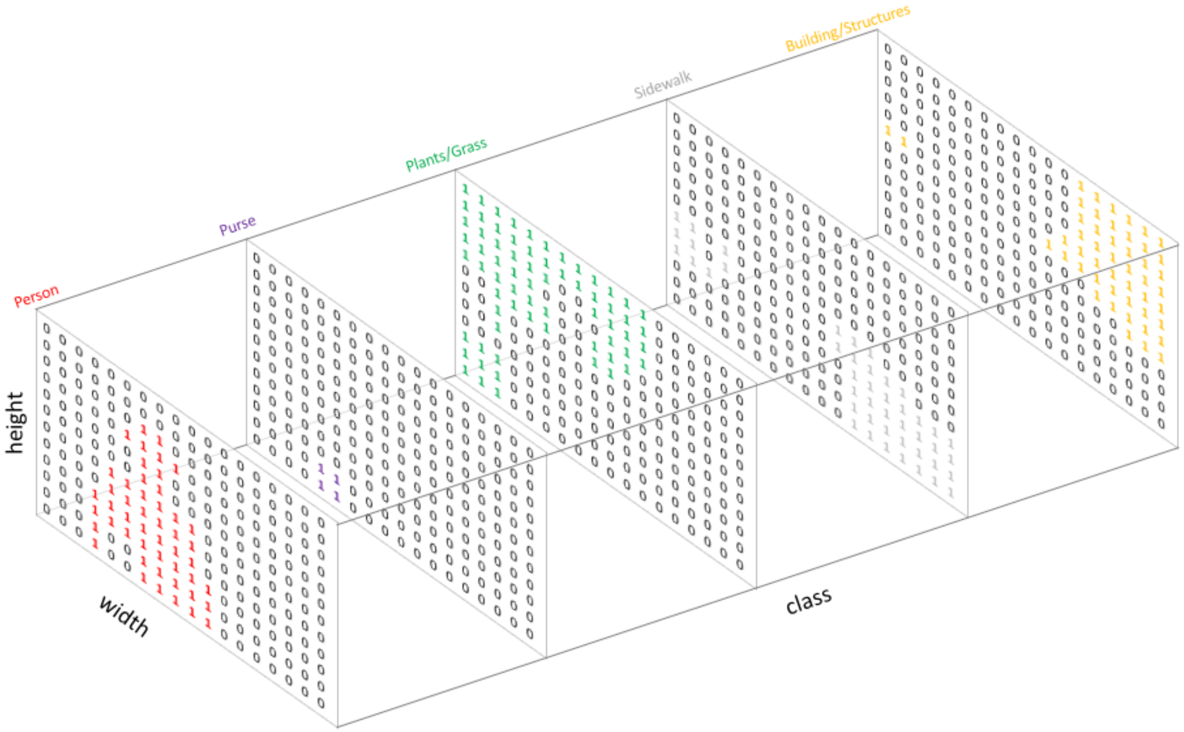
Mask R-CNN architecture (2017) [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Segmentation labels (recap)

- Similar to how we treat standard categorical values, we'll create our target by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.
- A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.



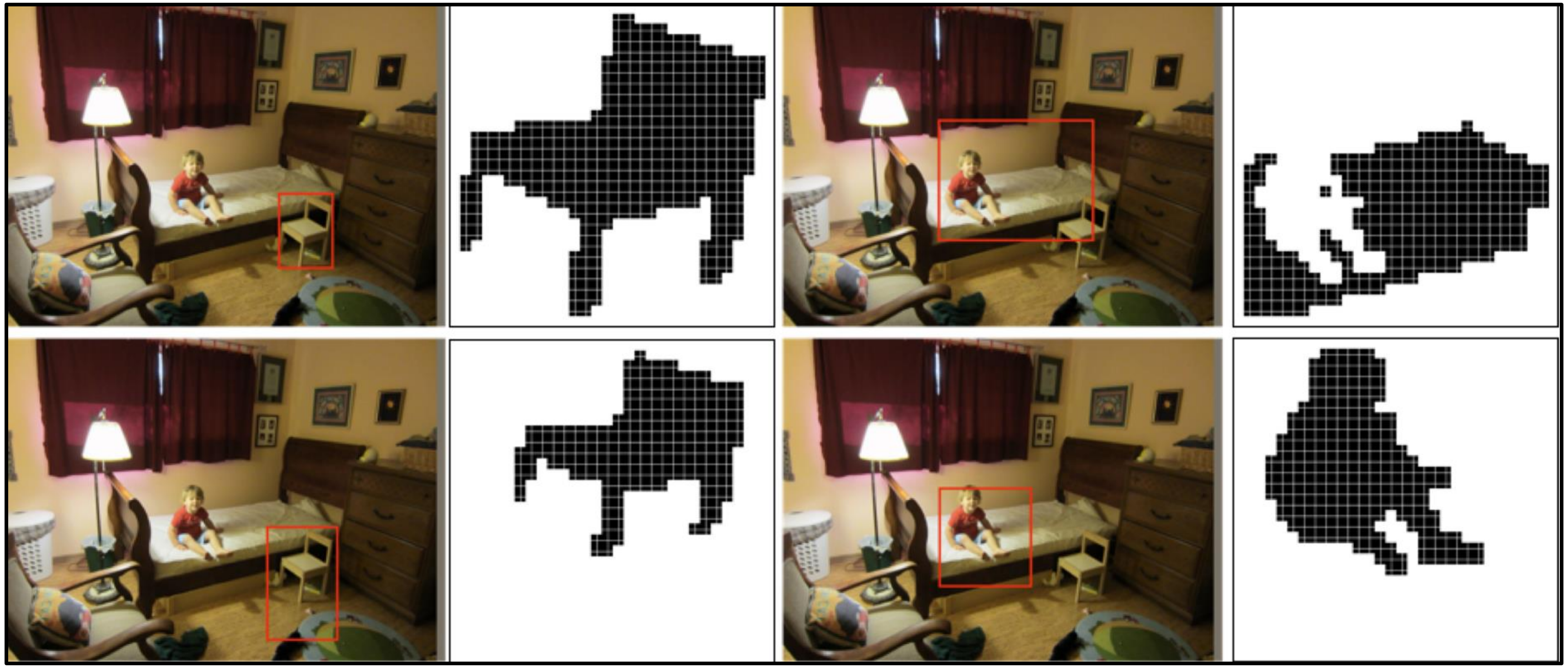
Depth-wise argmax

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

Instance Segmentation labels

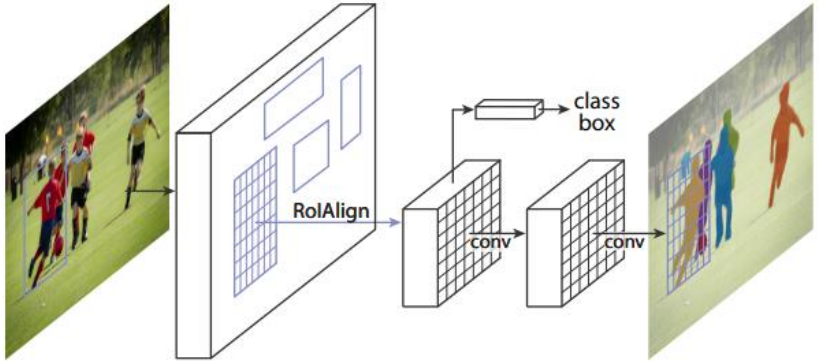
Mask R-CNN: Example Training Targets [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN architecture

- From [5] (More details on the paper)



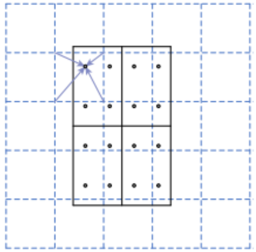


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

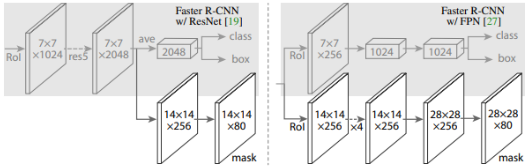
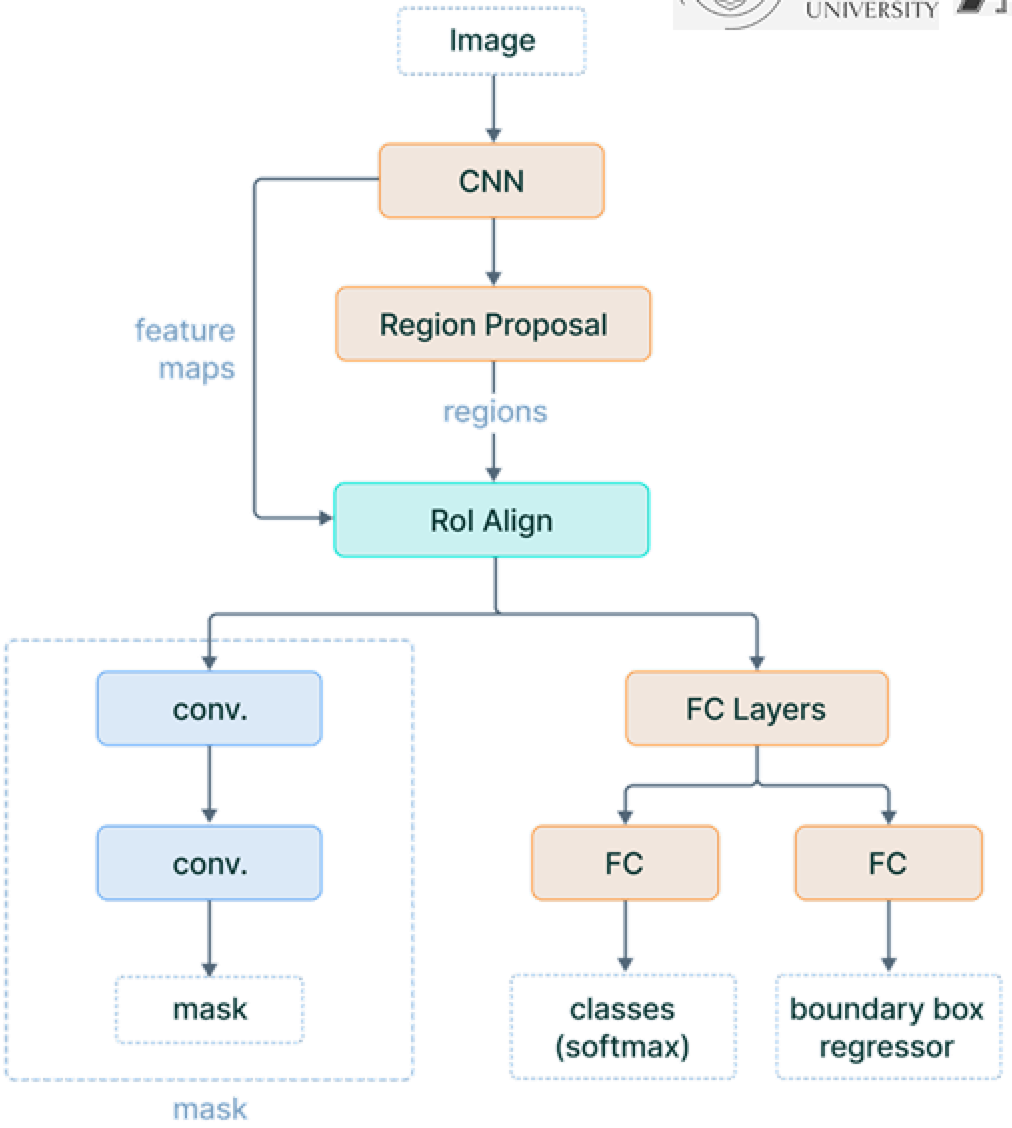
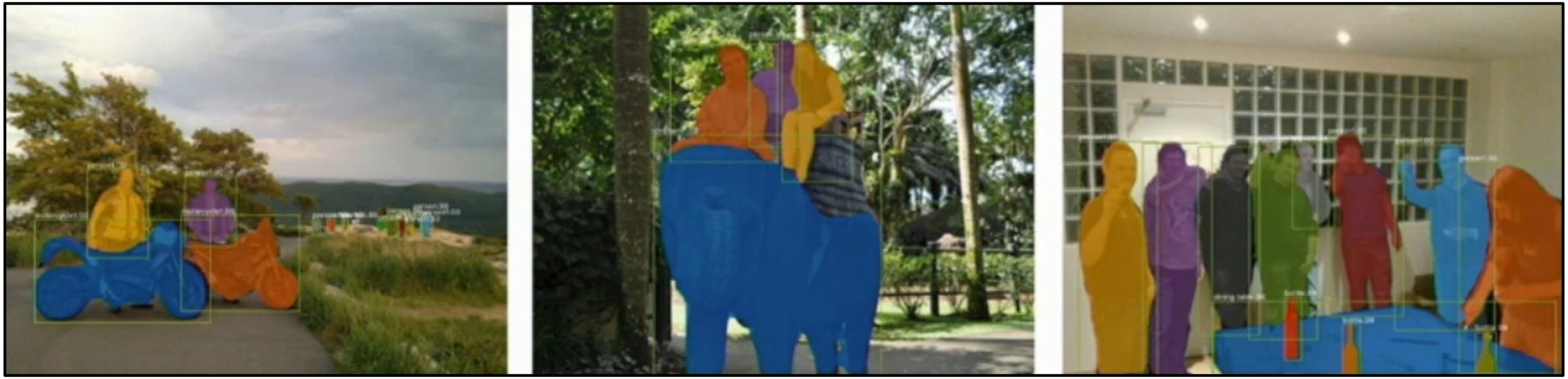


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN results [5]

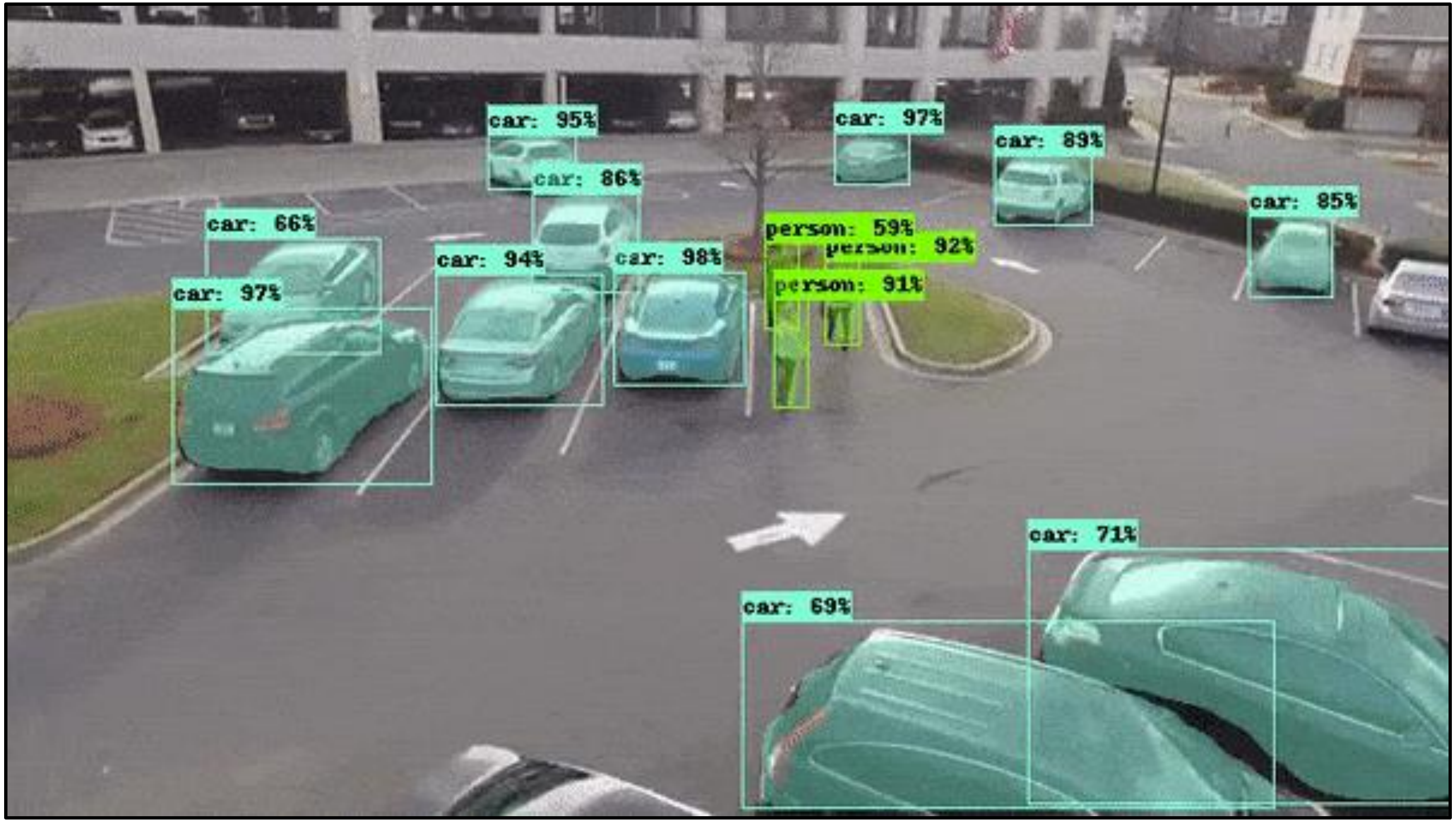


	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Mask R-CNN results [5]



[5] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1703.06870

Summary

- **Upsampling** is essential to reconstruct the original image from lower-resolution feature maps.
- By increasing the resolution, **upsampling** enlarges images with the following methods:
  - **Unpooling** upsamples by distributing a single value over higher resolution.
  - **Transpose Convolution** reverses the operation of convolution.

- Object masks are predicted within an image through **Image Segmentation.**
- **Fully Convolutional Networks (FCNs)** serve as encoders for coarse feature maps but struggle with detailed segmentations.
- **U-Net** improves localization by expanding the decoder's capacity for segmentation tasks.
- With **Mask R-CNN**, adding a mask prediction head allows for extended segmentation capabilities.

- **SAM:** Interactive image segmentation based on user prompt. The predicted mask does not contain a label.
- **SAM2:** Extending SAM with a memory to keep track of the segmented object for a video.

- **Semantic Segmentation**: Treats all objects of the same class as one, using one-hot encoded class labels.
- **Instance Segmentation**: Identifies individual instances of the same object.

Further Links + Resources

- A survey of loss functions for semantic segmentation - https://arxiv.org/pdf/2006.14822
- R-CNN - https://medium.com/@selfouly/r-cnn-3a9beddfd55a
- Review: Fully Convolutional Network (Semantic Segmentation) - https://medium.com/towards-data-science/review-fcn-semantic-segmentation-eb8c9b50d2d1

Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

That’s all for today!