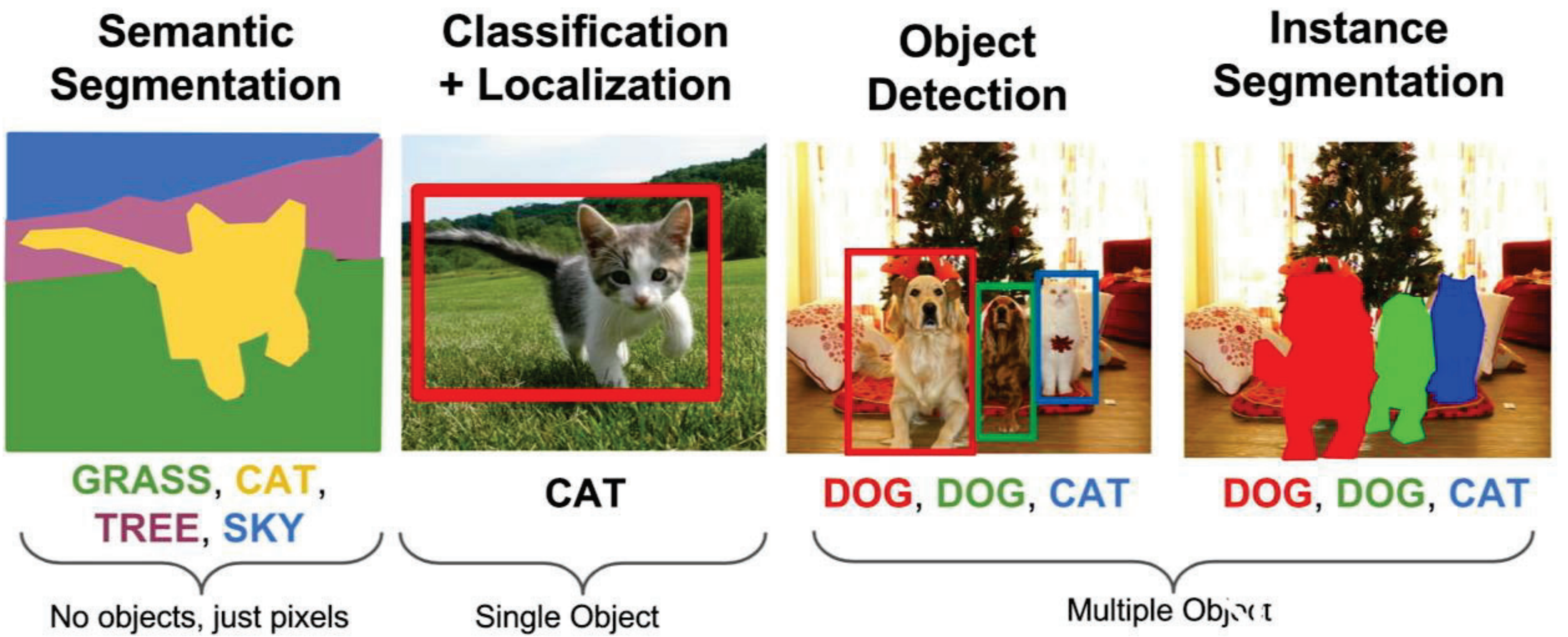
Review Typical CV tasks

- Image recognition
  - Recognize car/dog/person/building etc.
  - Recognize whether an image contains/is a certain object.
- Semantic segmentation
  - Recognize different parts in an image
  - Where is the road, where is the building, where is the sky, etc.
- Object detection
  - How may objects are there in an image and what are they.
- Instance segmentation
  - Object detection + semantic segmentation

Review Typical CV tasks

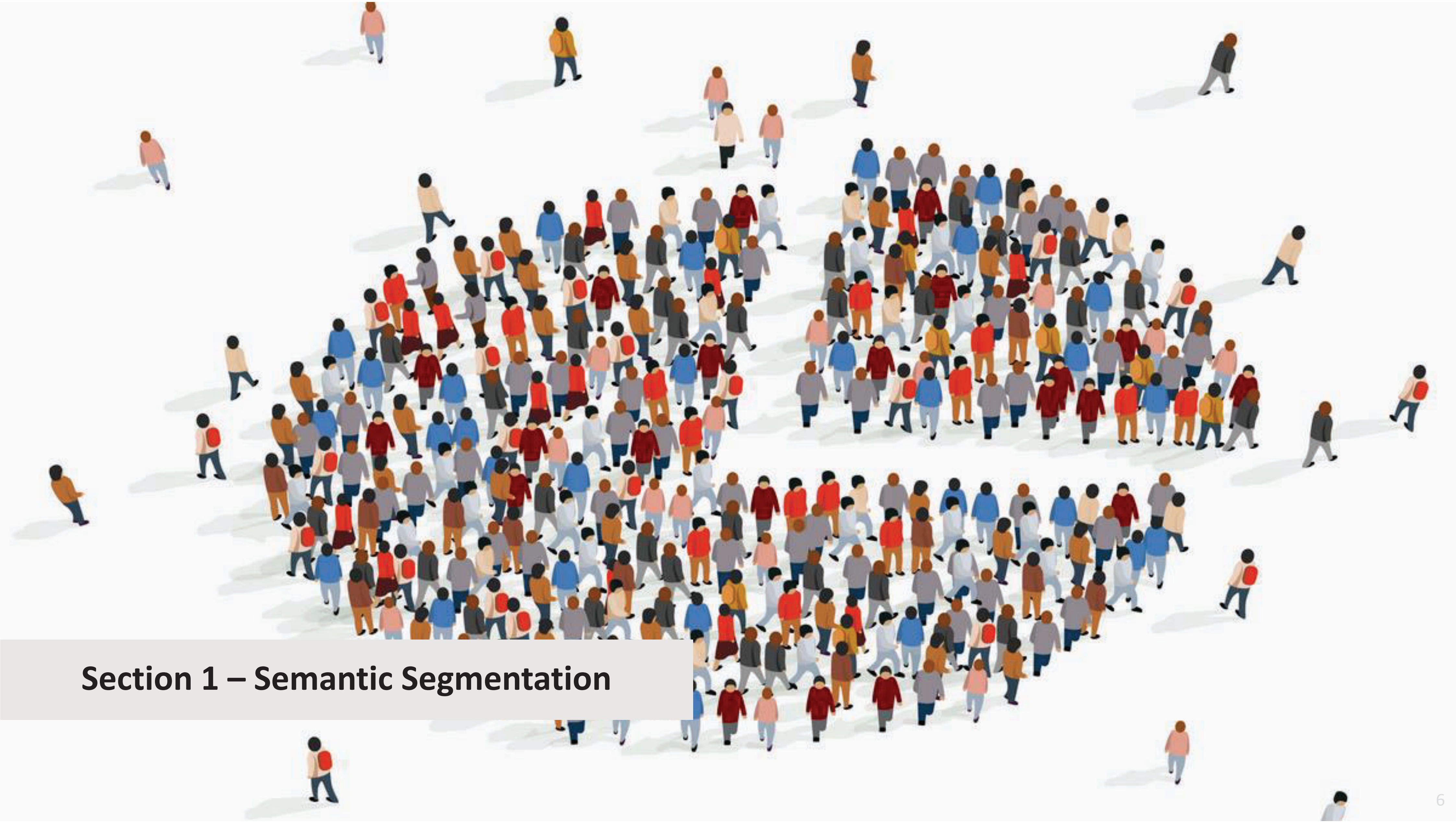


Review Object Detection

- RCNN
- Fast RCNN
- Faster RCNN
- YOLO

Quick Question

- Which method utilized the anchor box technique?
    1. RCNN
    2. Fast RCNN
    3. Faster RCNN
    4. YOLO

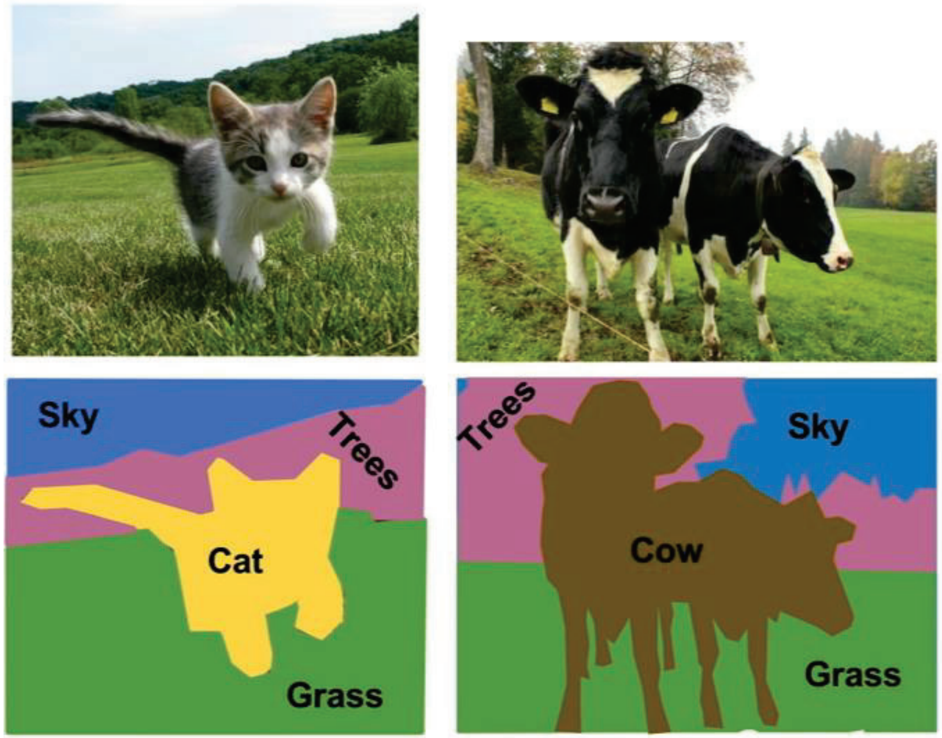


Section 1 – Semantic Segmentation

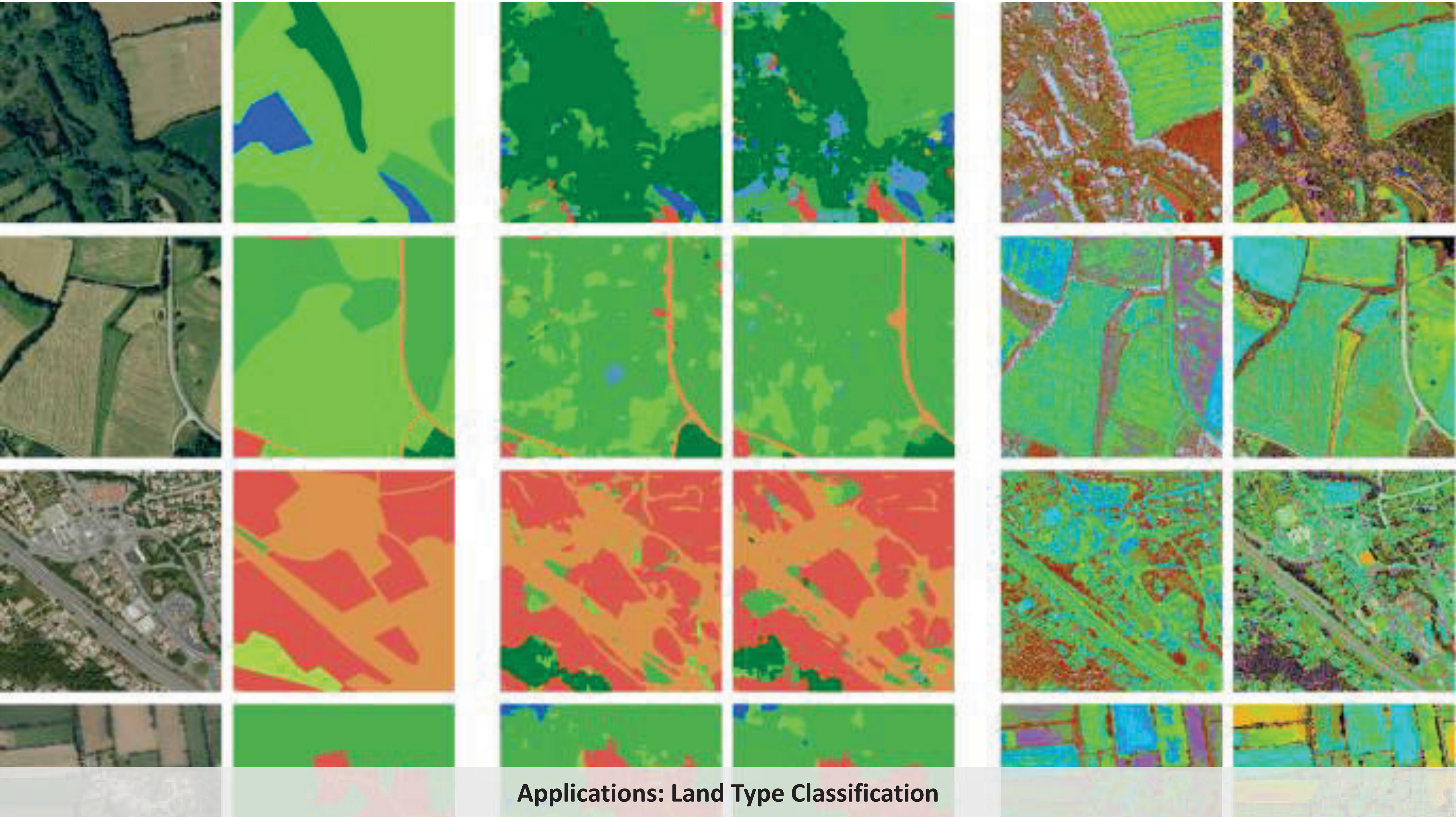
Semantic Segmentation

Label each pixel in the image with a category label

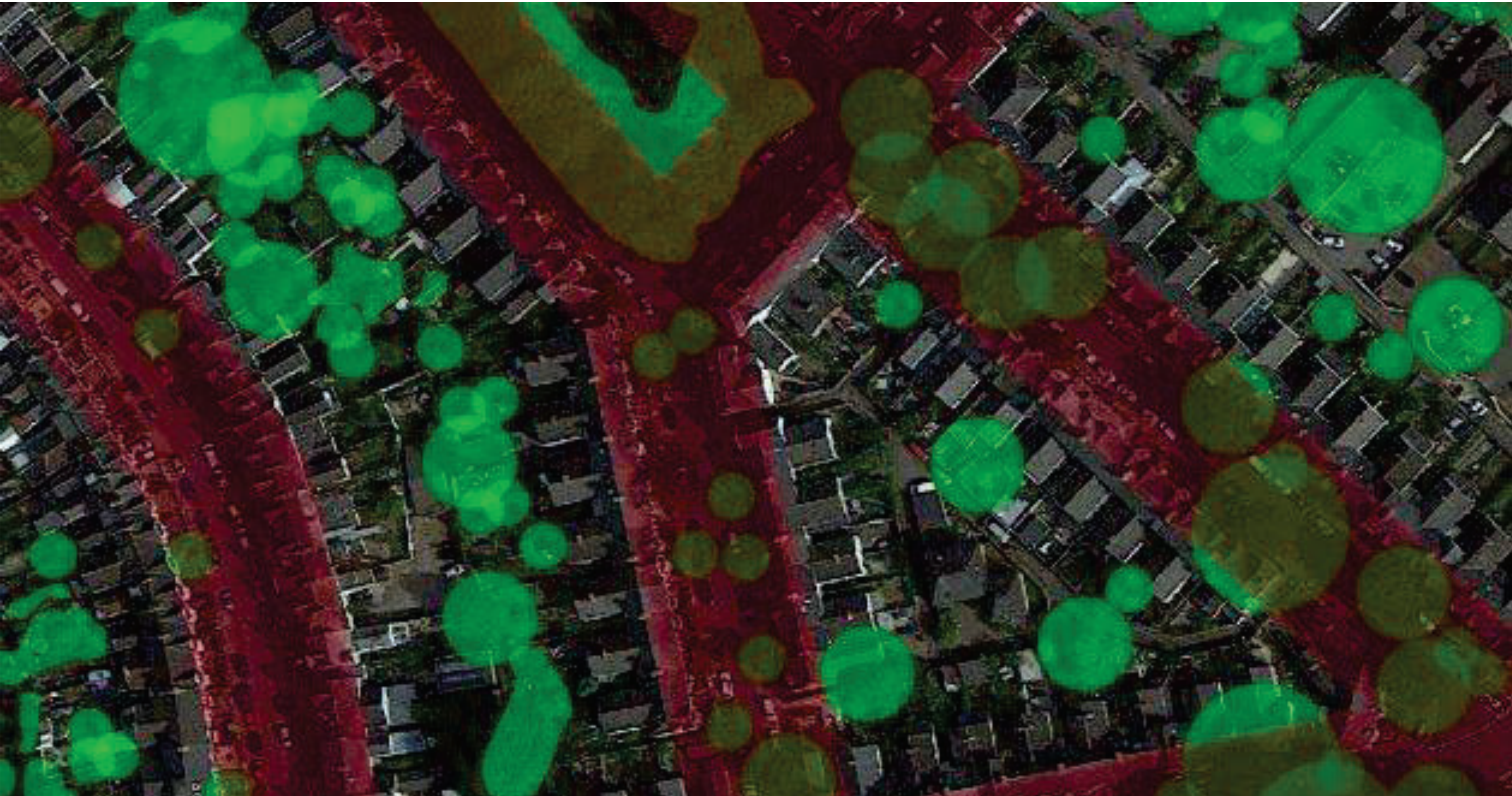
Don't differentiate instances, only care about pixels



Applications: Street Scene Understanding



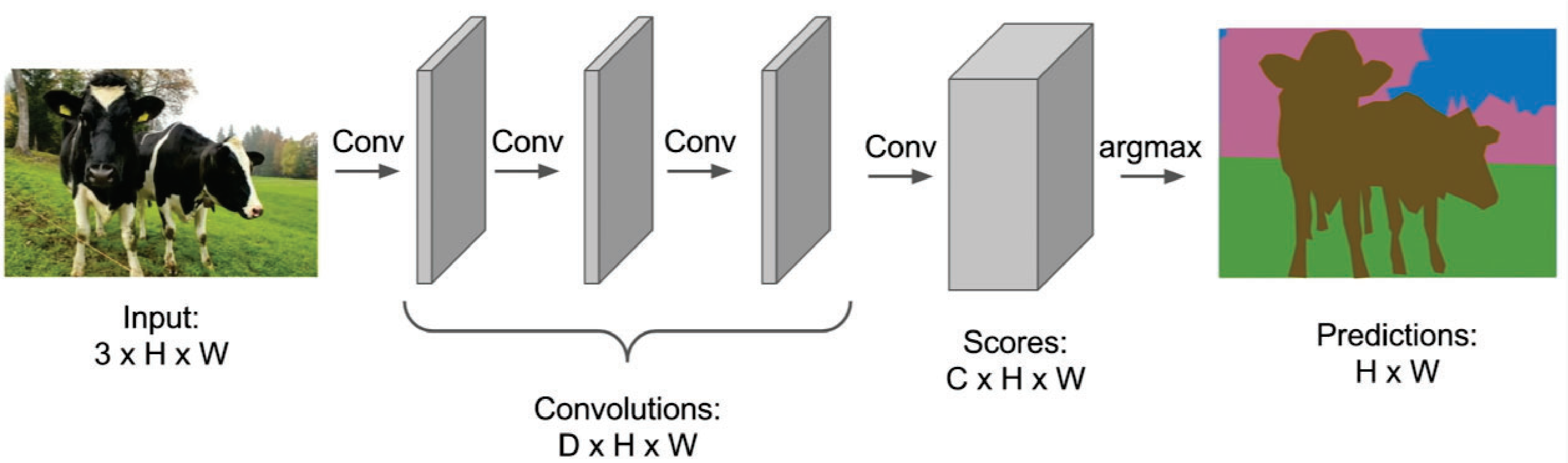
Applications: Land Type Classification



Application: Urban Vegetation/Road Analysis

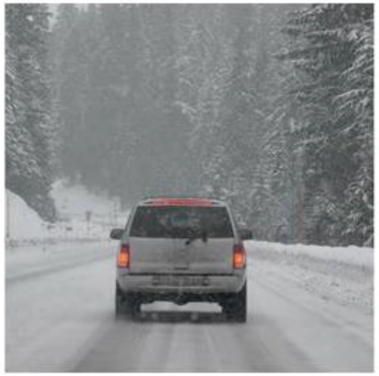
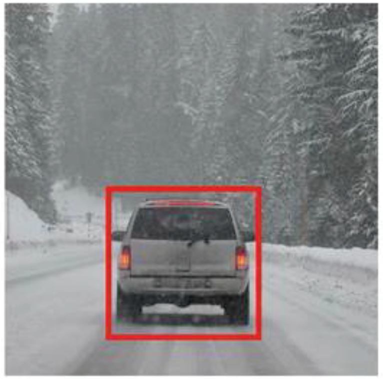
Application: Roof/Building Analysis

The basic idea is the easy



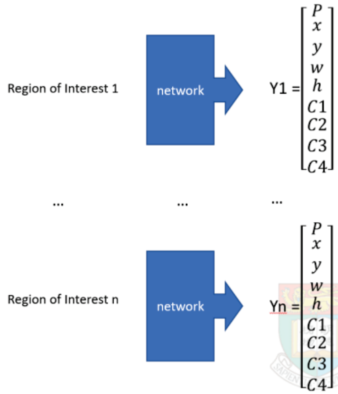
Just change the outputs into a matrix (CxHxW, where C is the category, H and W is the original image size)

Try to feel the difference (x are the same, feel y)

Image classification	Classification with localization	Detection	Segmentation
			

$$Y = \begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$$

$$Y = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$$

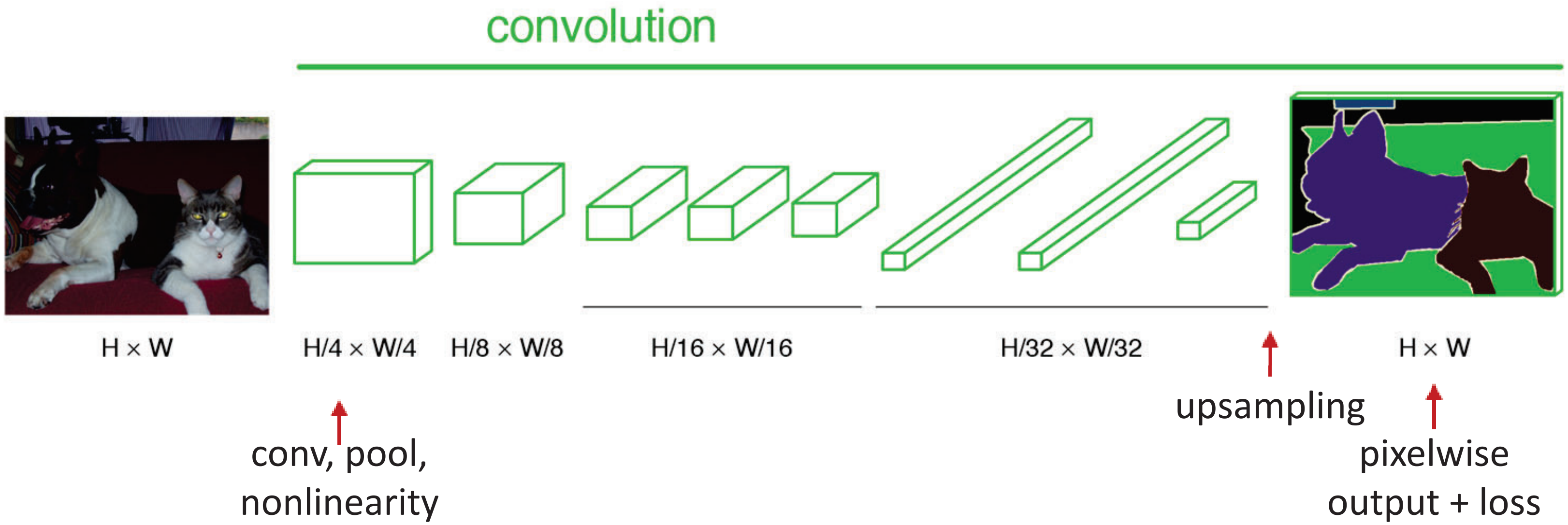


$$Y = \begin{bmatrix} C1C1C1 & & C1C1C1 \\ C1C1C1 & & C1C1C1 \\ C1C1C1 & & C1C1C1 \\ C2C2C2 & & C2C2C2 \\ C2C2C2 & \cdots & C2C2C2 \\ C2C2C2 & & C2C3C3 \\ \cdots\ \cdots\ \cdots & & \cdots\ \cdots\ \cdots \\ C2C2C2 & & C2C3C3 \\ C4C4C4 & & C4C4C4 \end{bmatrix}$$

Quick Question

- Which method introduce regional proposal network?
  1. RCNN
  2. Fast RCNN
  3. Faster RCNN
  4. YOLO

Major Technical Difference



1. Pixelwise Output
2. Loss
3. Up sampling

1. Pixelwise Output

- The process is easy to understand
- But the data could be time consuming to prepare.

- Precise localization is hard to annotate
- Common solution: annotate few classes (often things), mark rest as "Other"
- Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)

2. Loss

- Cross entropy
  - Cross entropy of every pixels and then average or sum.
  - Just understand them a classification task for HxWx3 number of elements.

$$L(y, \hat{y}) = -y\log(\hat{y}) - (1-y)log(1-\hat{y})$$

- Others
  - Dice Coefficient

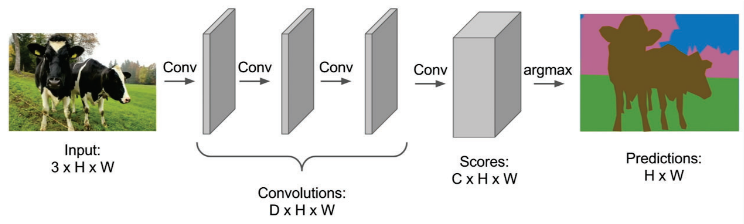
$$DL(y, \hat{y}) = 1 - \frac{2y\hat{y} + 1}{y + \hat{y} + 1}$$

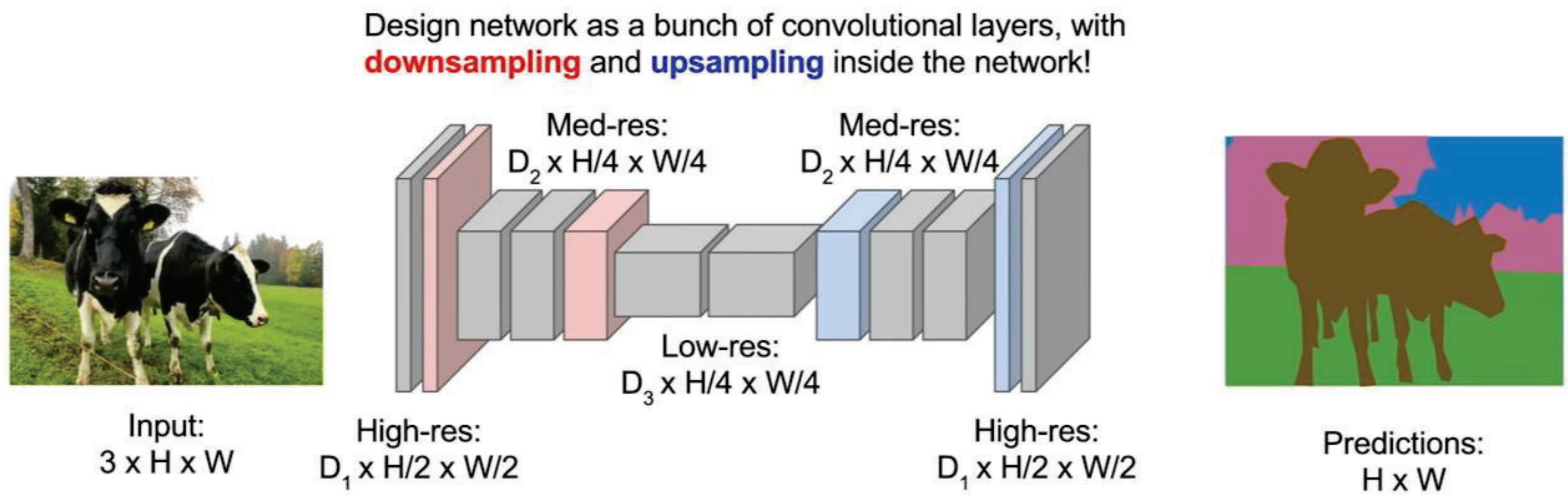
  - Shape-aware loss

$$L(y, \hat{y}) = -w(\mathbf{x}) \times [y\log(\hat{y}) + (1-y)\log(1-\hat{p})]$$

3. The up-sampling in practice

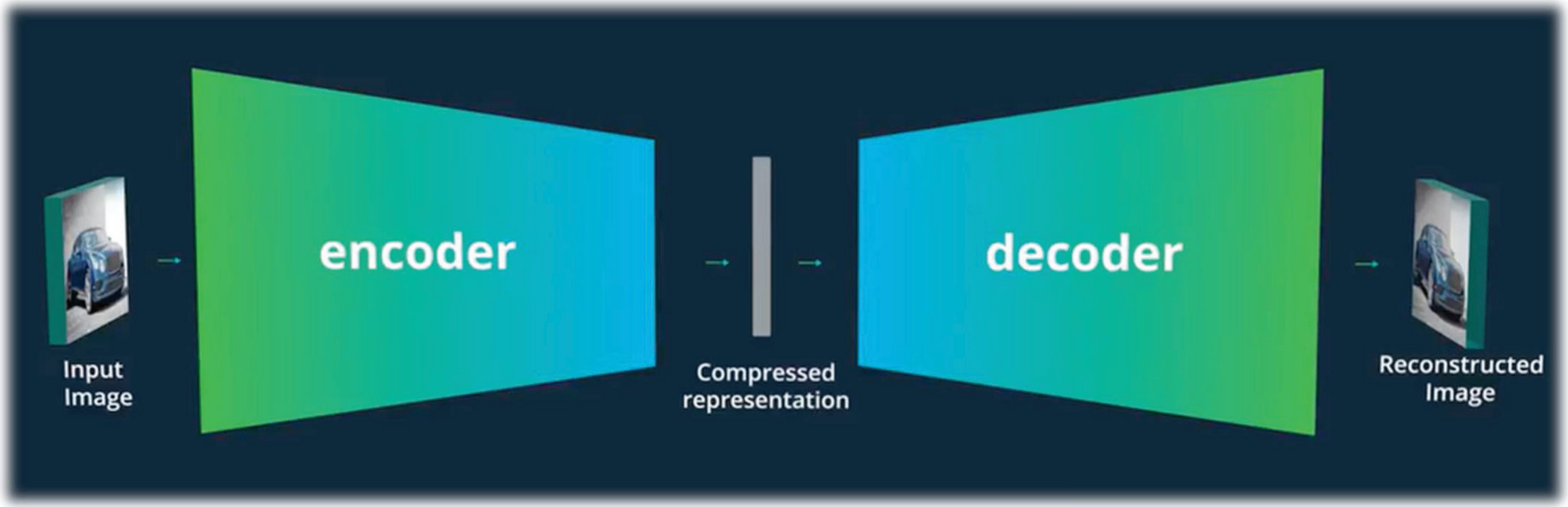
- Directly learn the segmentation from images are difficult
- We trying to utilize the power from image recognition
- The encoding and decoding idea





The idea is like you teach a kid how to draw an image. The process is let him know he is drawing a cow first and the other details.

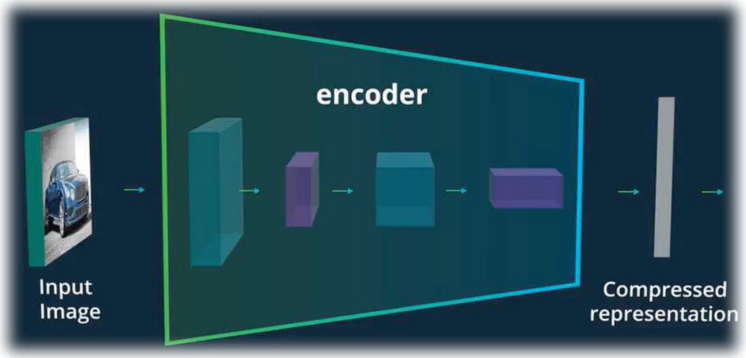
The key idea is called autoencoder

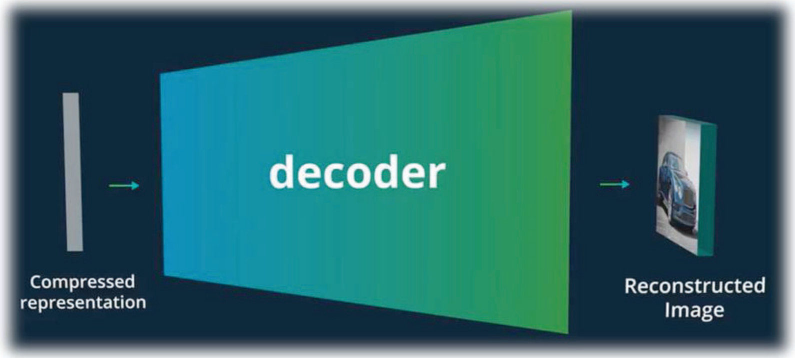


For applications involving image reconstruction:

1) Semantic segmentation, 2) noise removal, 3) image reconstruction, 4) style transfer etc.

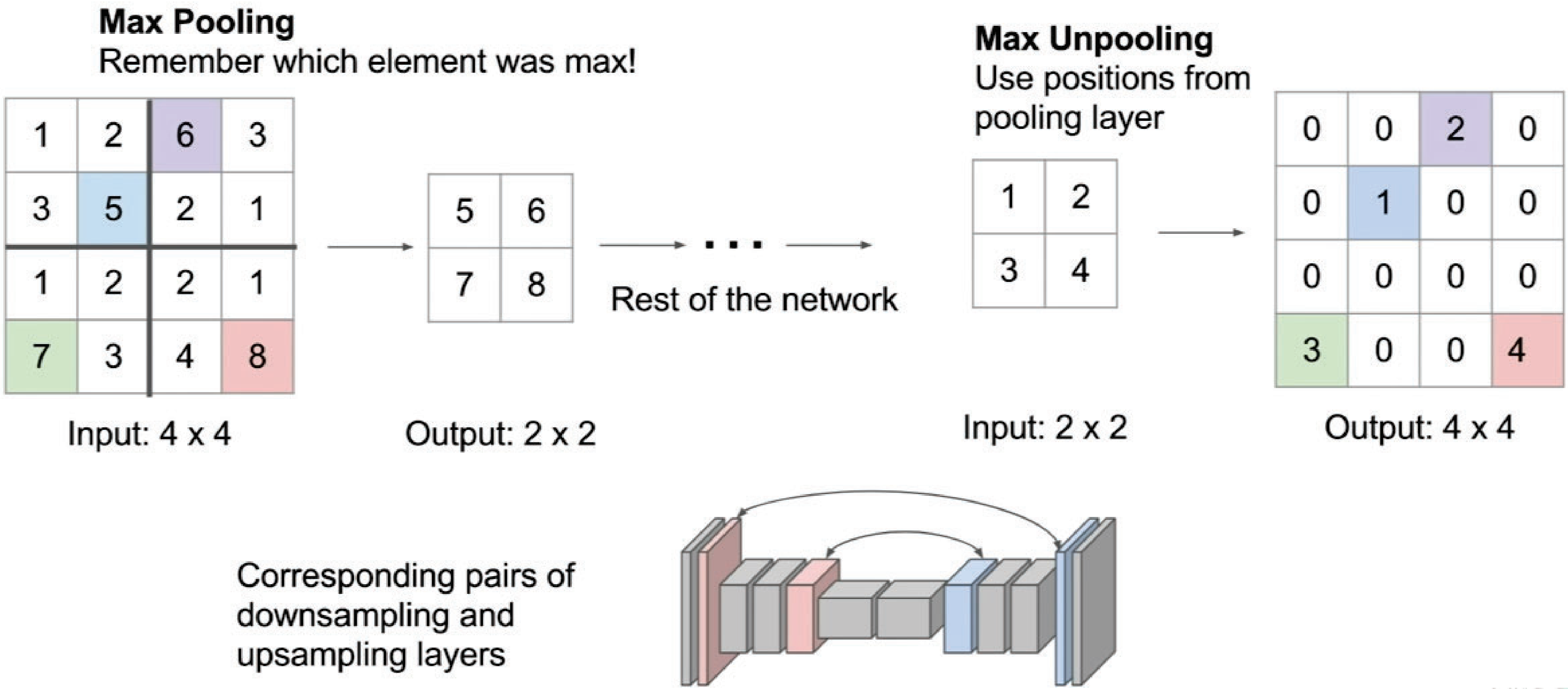
The decoder process is trying to "reverse"





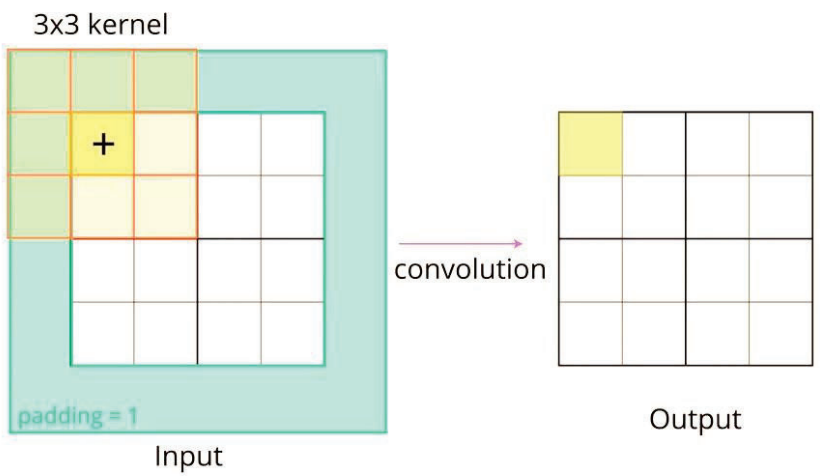
- In the encoding CNN part, we have convolutional layer and max polling layer
- In decoder, we somehow reverse the process:
  - "reverse" fully connected layer: still fully connected.
  - "reverse" max pooling: max unpooling
  - "reverse" convolution: transpose convolution

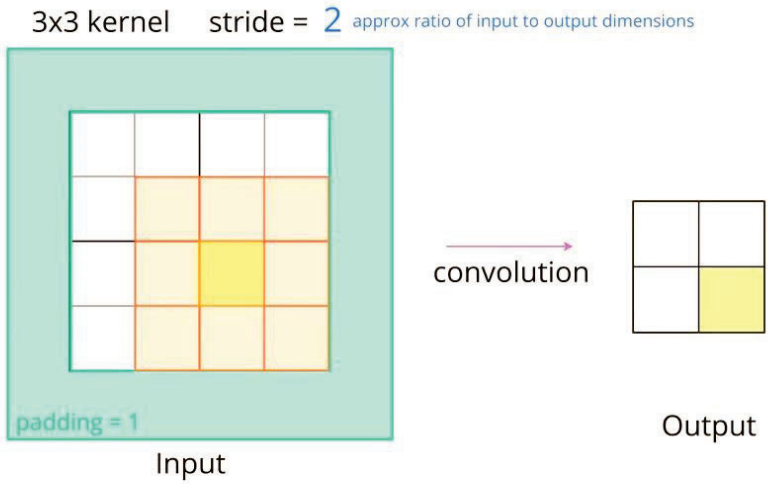
The max pooling and max unpooling



Learnable Up-sampling: Transpose Convolution

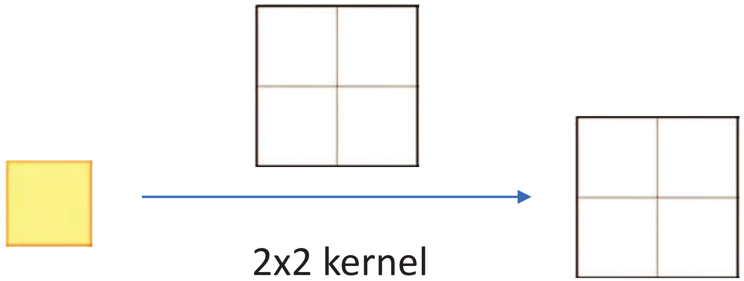
- How about the convolutional layers? How to reverse? --Transpose convolution
- Before transpose convolution, let's go over forward/normal convolution one more time.

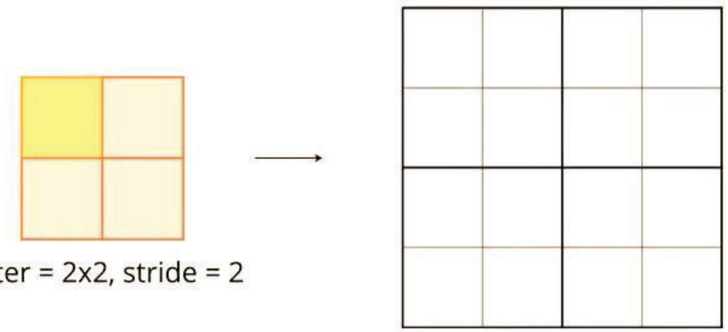




Forward convolution layer

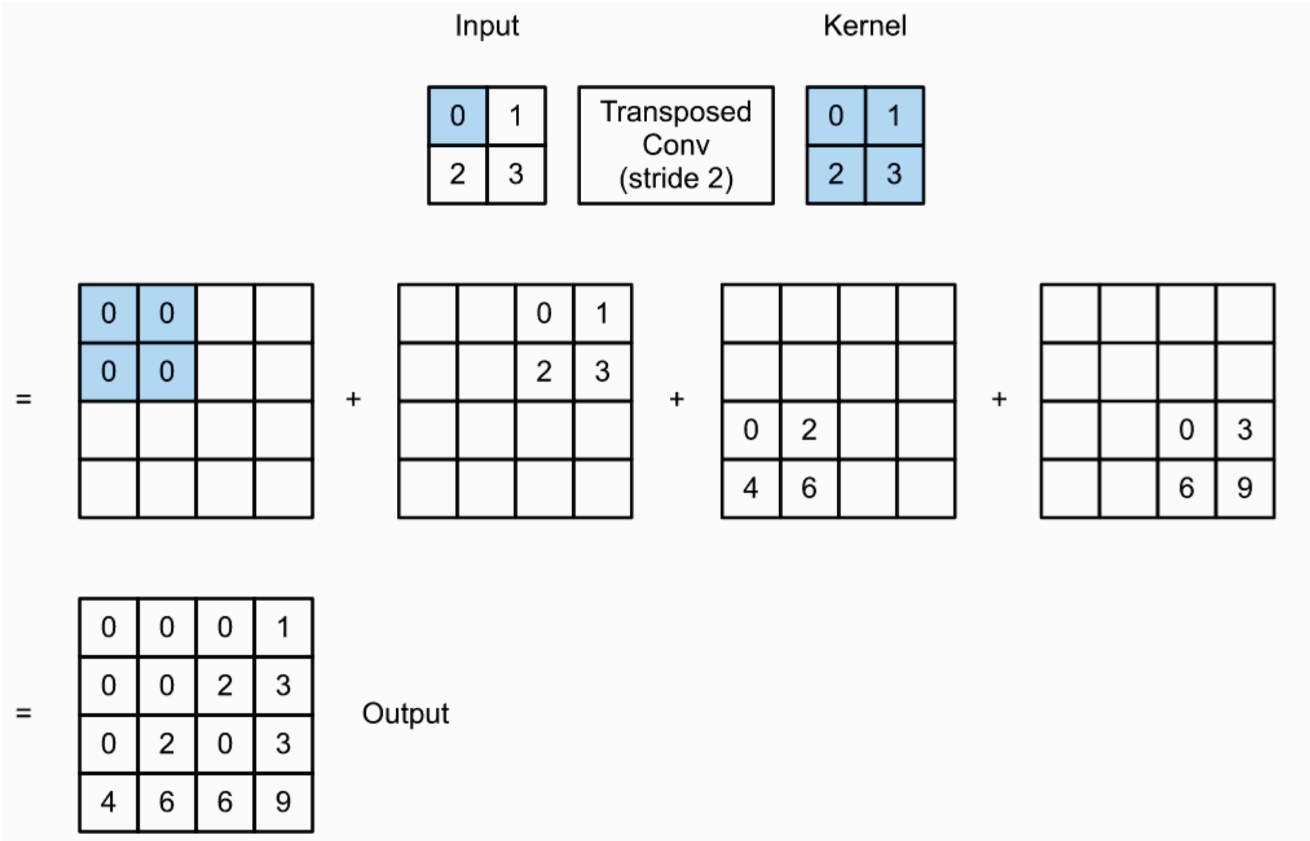
Learnable Up-sampling: Transpose Convolution





Usually we use filter = 2x2, stride = 2.

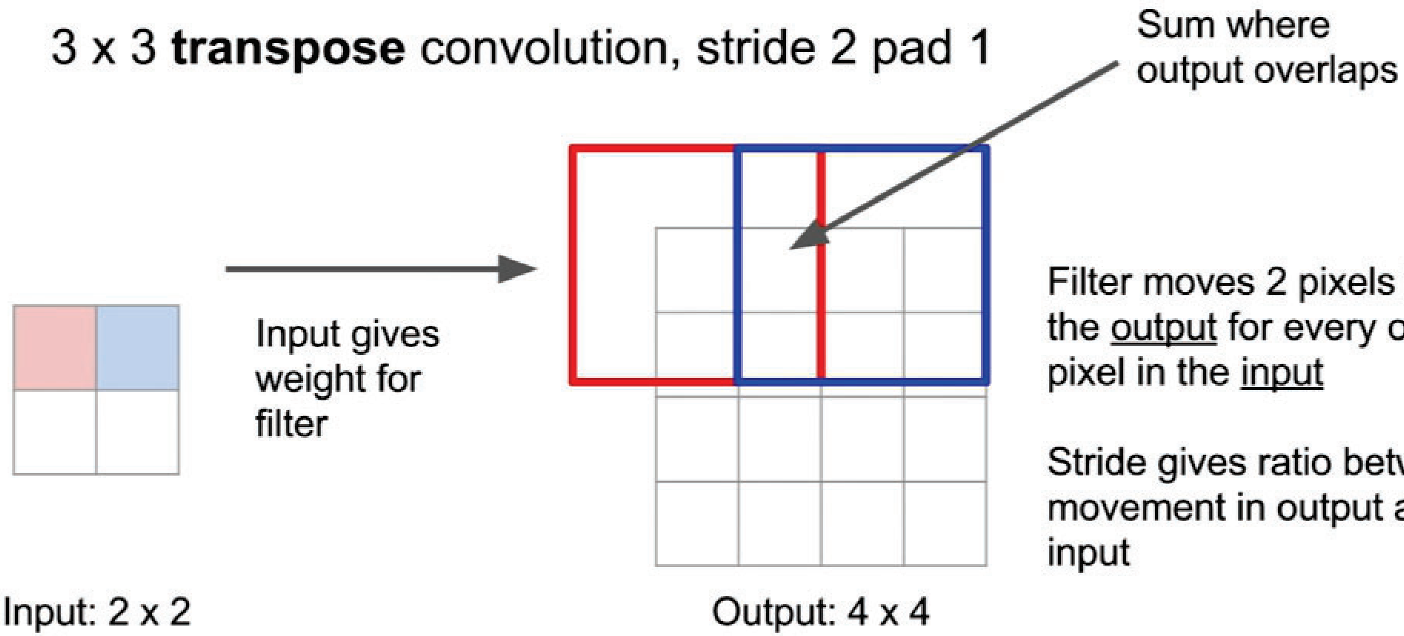
Learnable Up-sampling: Transpose Convolution



Learnable Up-sampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

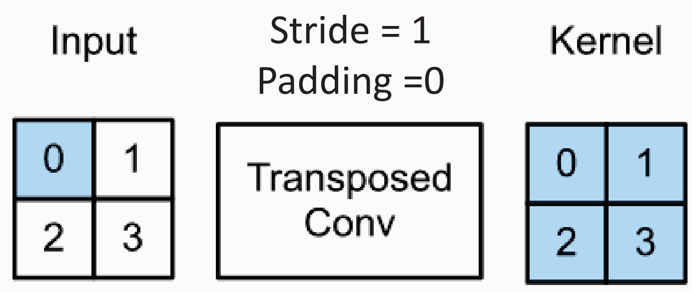
3 x 3 **transpose** convolution, stride 2 pad 1



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

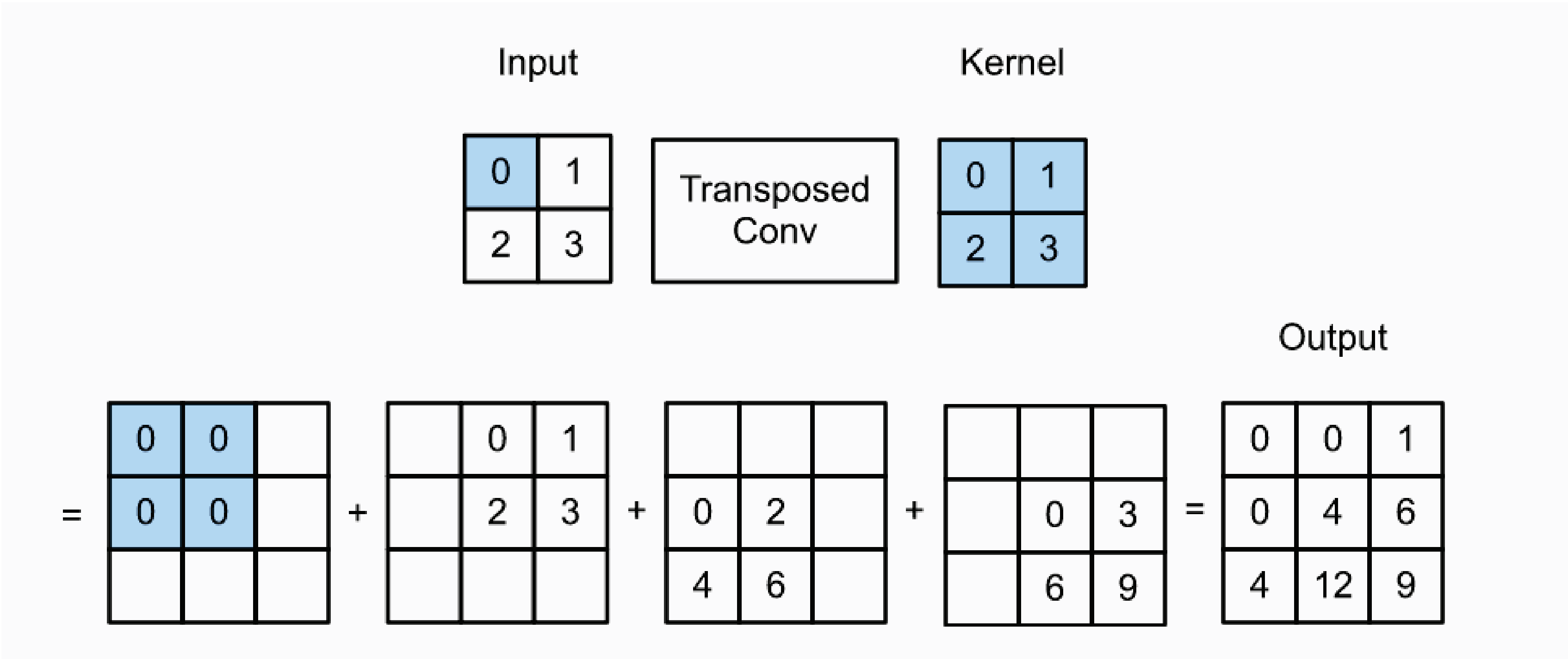
Stride gives ratio between movement in output and input

Quick question



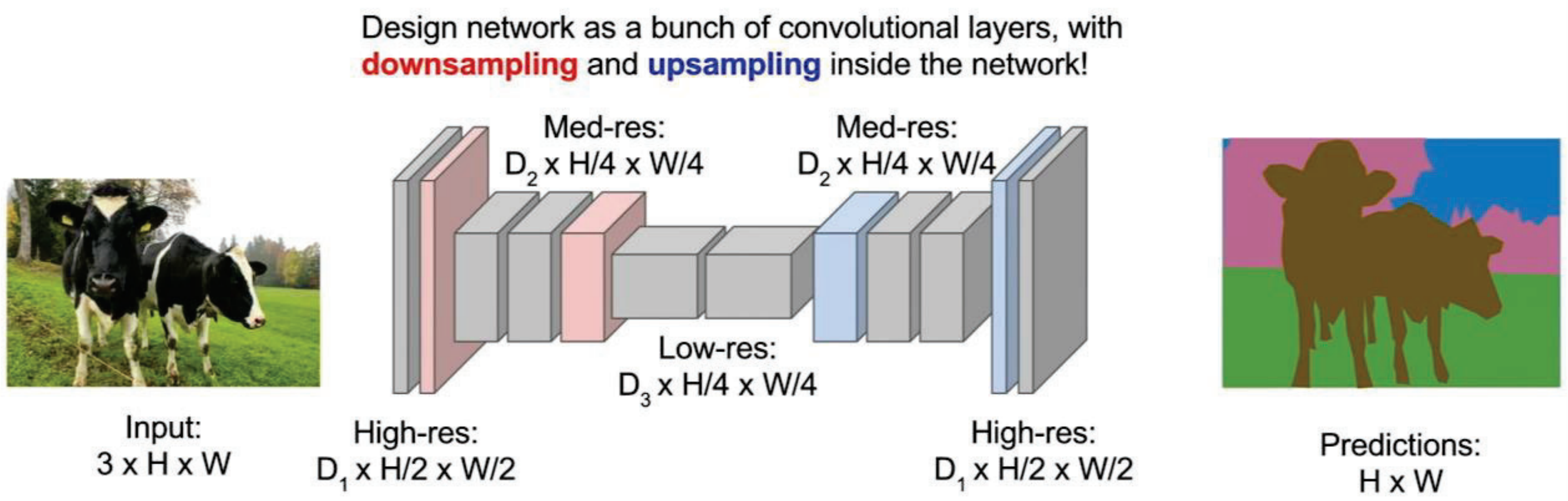
What is the output shape?

Learnable Up-sampling: Transpose Convolution



If overlapped, then sum

Semantic Segmentation



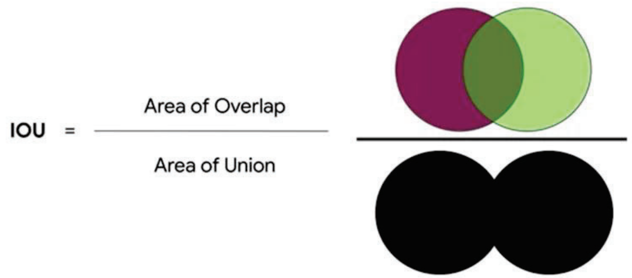
Famous Networks for Semantic Segmentation

- For demonstration:
  - Typically autoencoder CNN types of structures
  - CNN, AlexNet, VGG, ResNet, etc.
- Unet (Unet ++)
- DeepLab (DeepLab v3+)
- For 3D:
  - Point cloud: PointNet, ACNN
  - Video: Spatio-Temporal FCN

- Data labeling tools: https://github.com/heartexlabs/awesome-data-labeling

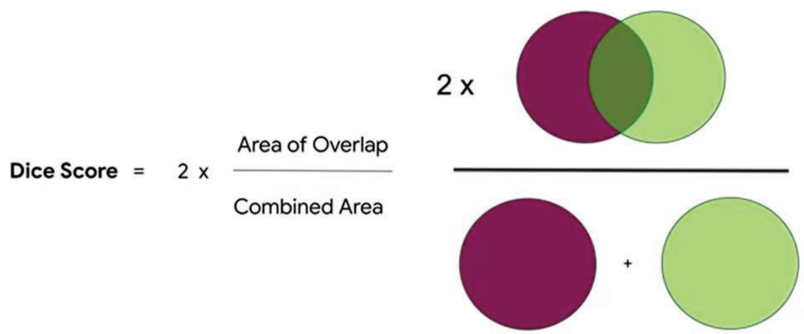
Evaluation Metric

- IoU



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

- Dice score (different from Dice coefficient in the loss)



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

Evaluation Metric

IOU Results

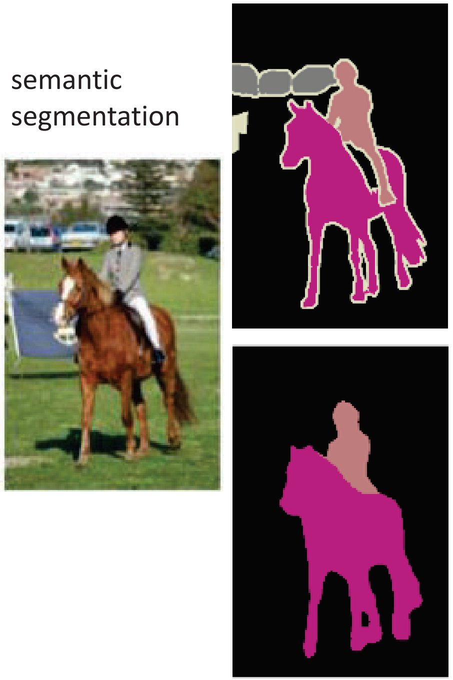
sky	0.8779669959482955
building	0.7570989578412737
column/pole	4.57875457665808e-10
road	0.915543155822588
side walk	0.7235628237658467
vegetation	0.7664541807647628
traffic light	3.0202657798187055e-05
fence	0.006380242448568188
vehicle	0.2950299461448835
pedestrian	0.00012643332766080866
byciclist	0.023621930993270864
void	0.16456276759816527

Dice Score Results

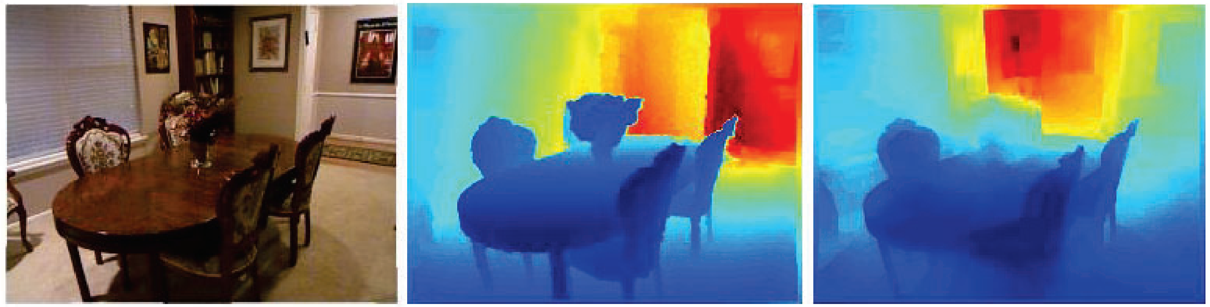
sky	0.9350185576821789
building	0.861760180856767
column/pole	9.15750915331616e-10
road	0.9559097147402678
side walk	0.8396129387346395
vegetation	0.8677883515092748
traffic light	6.040349126864078e-05
fence	0.012679586065167121
vehicle	0.45563416820437186
pedestrian	0.0002528346886971326
byciclist	0.0461536242658659
void	0.2826172572009191

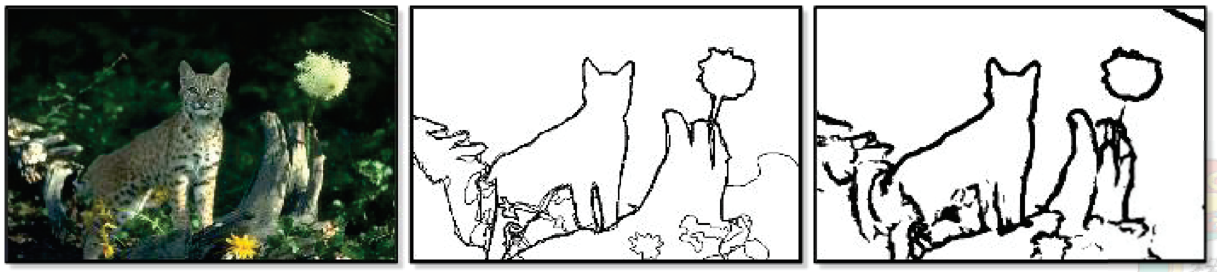
Other Applications of Autoencoder type network

semantic
segmentation



monocular depth estimation (Liu et al. 2015)





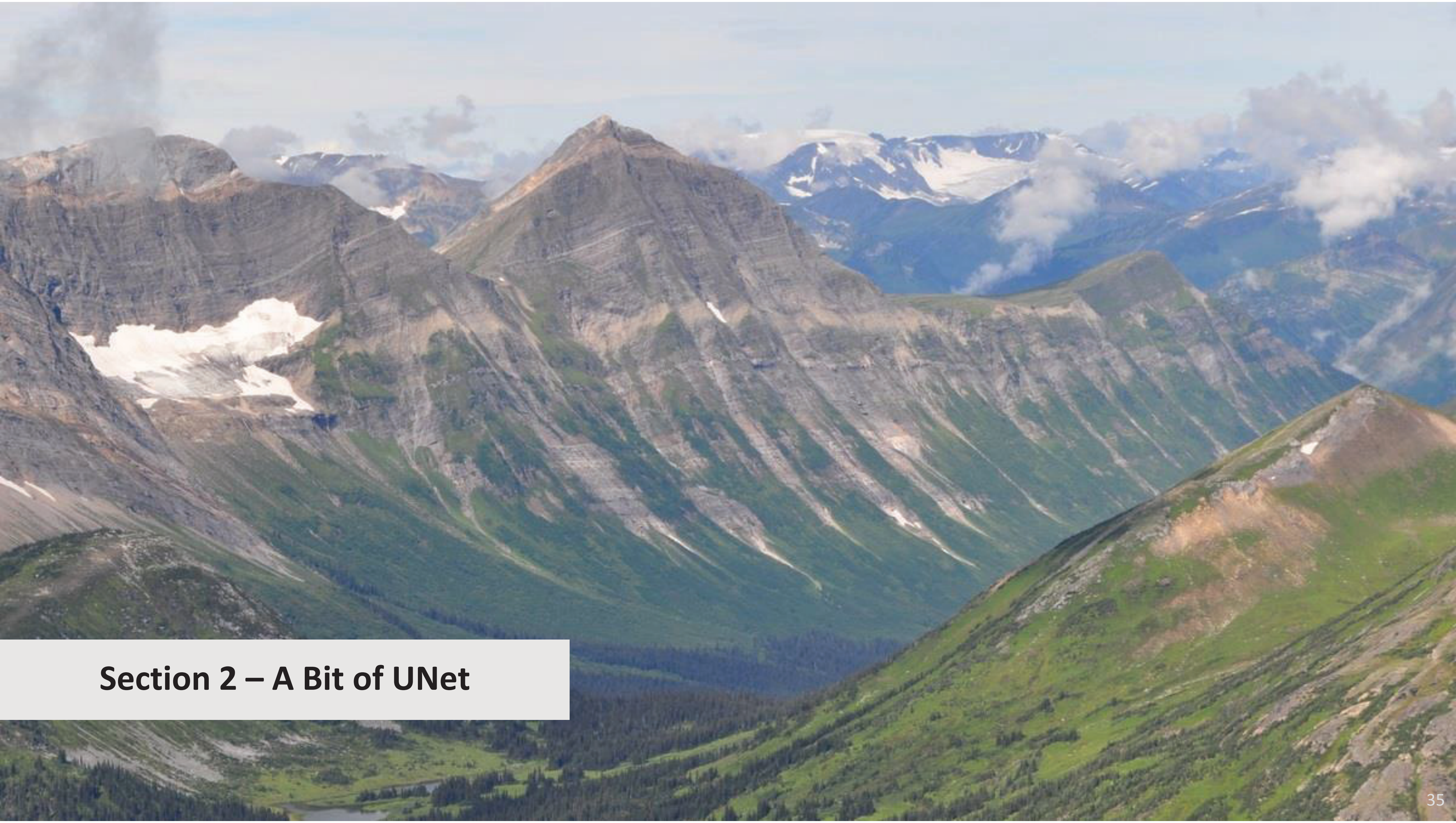
boundary prediction (Xie & Tu 2015)

Others: change background, change clothes/dress, …

Quick Question

- Which is not a commonly seen technique/concept in semantic segmentation?
  1. Maxpooling
  2. Pixel wise output
  3. Upsampling
  4. Regional proposals

Q



Section 2 – A Bit of UNet

Why UNet

- The structure contains useful innovations in network design, which is helpful to take a look.

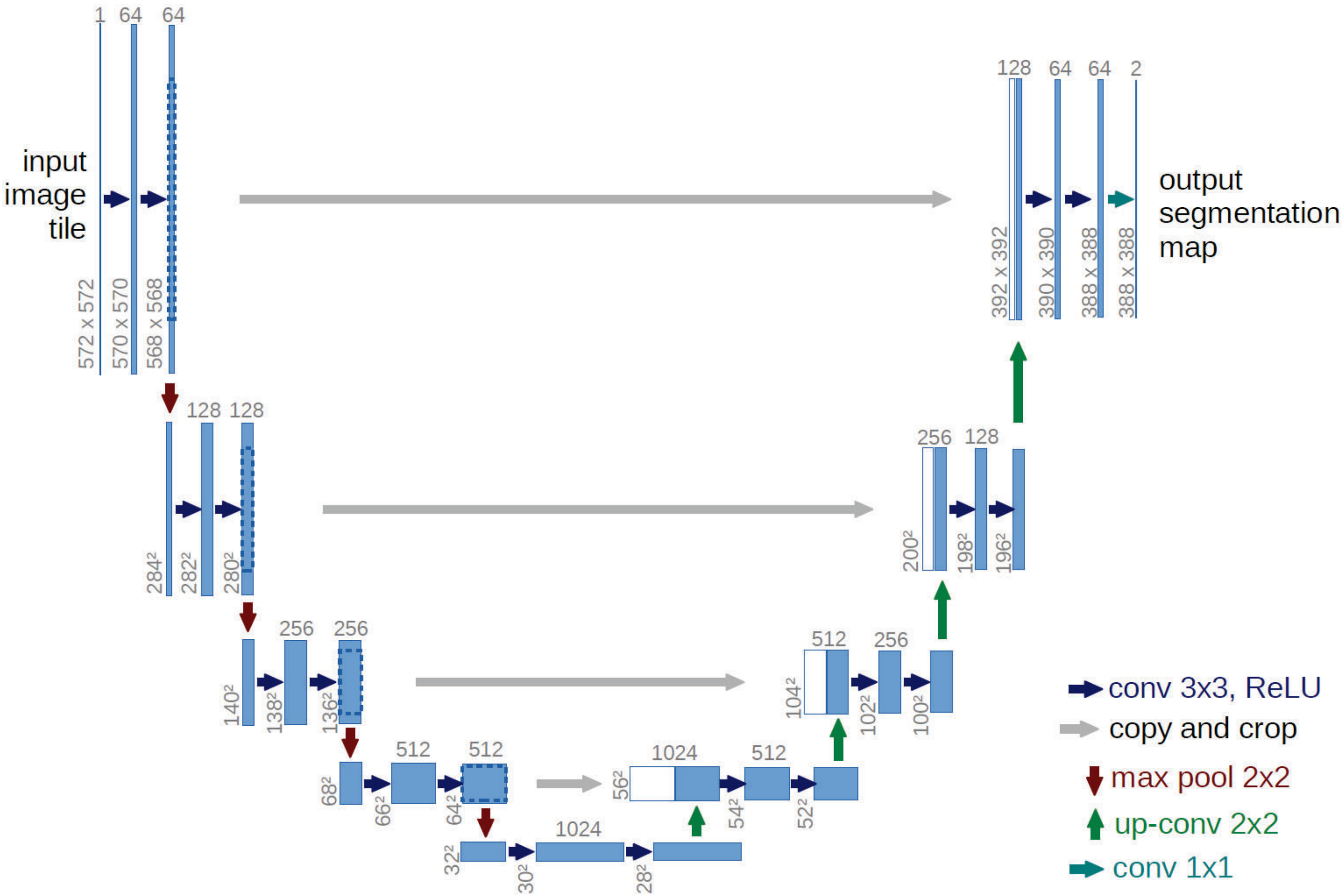
- Easy to understand

- Try to feel the flexibility in network design, and try to understand that why scholars design structures like this.

UNet

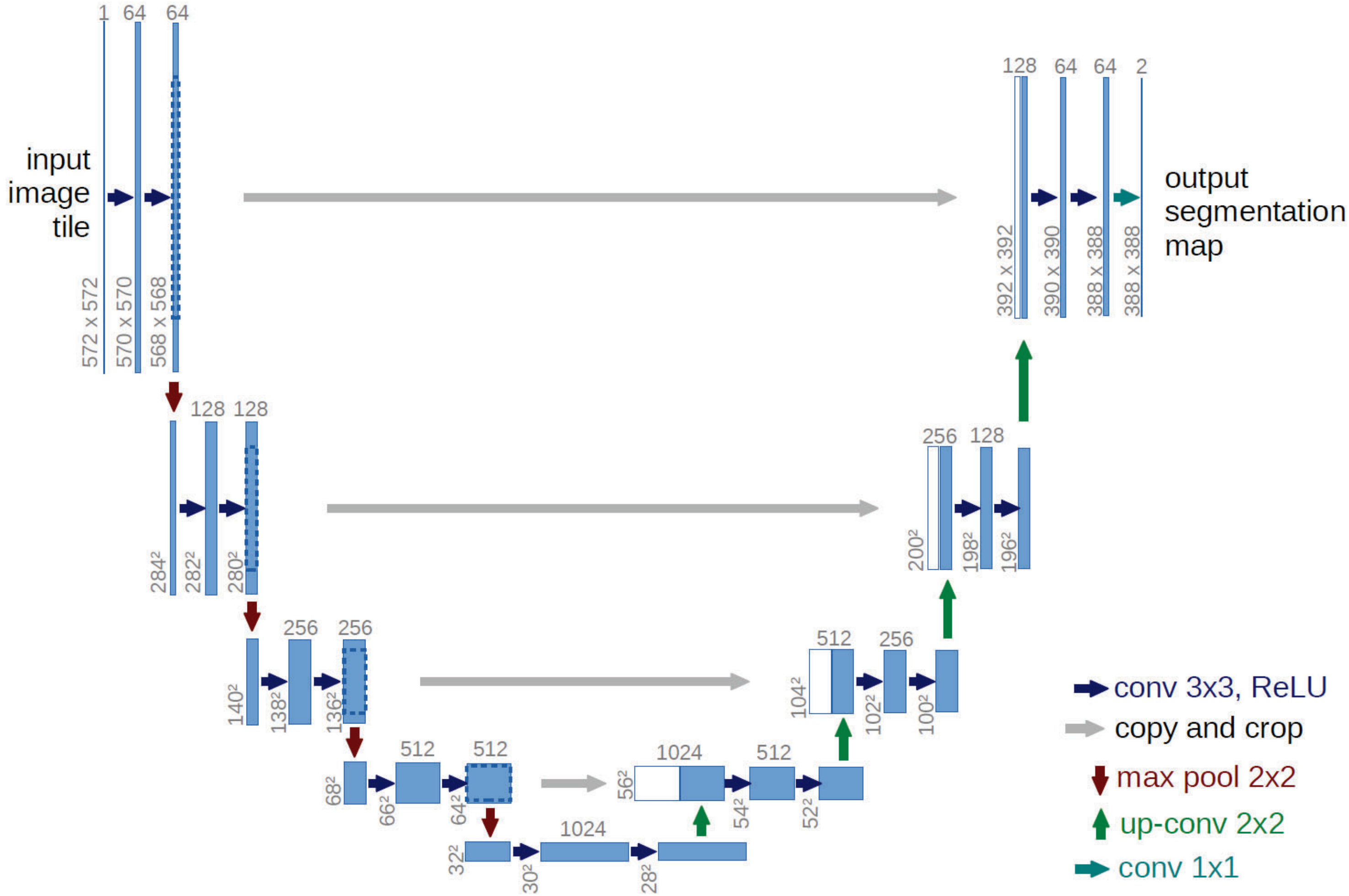
- The U-Net build upon the concept of FCN. Its architecture, similar to the above encoder-decoder architecture, can be divided into three parts:
  - The contracting or downsampling path consists of 4 blocks where each block applies two 3x3 convolution (+batch norm) followed by 2x2 max-pooling. The number of features maps are doubled at each pooling layer (after each block) as 64 -> 128 -> 256 and so on. (≈ encoding)
  - The horizontal bottleneck consists of two 3x3 convolution followed by 2x2 up-convolution.
  - The expanding or upsampling path, complimentary to the contracting path, also consists of 4 blocks, where each block consists of two 3x3 conv followed by 2x2 upsampling (transpose convolution). The number of features maps here are halved after every block. (≈ decoding)

UNet

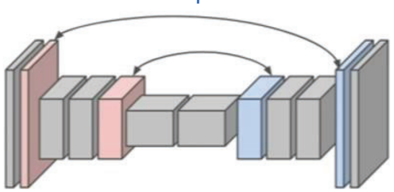


1
64
64
input
image
tile
572 x 572
570 x 570
568 x 568
128
128
284²
282²
280²
256
256
140²
138²
136²
512
512
68²
66²
64²
1024
32²
30²
28²
1024
512
56²
54²
52²
512
256
104²
102²
100²
256
128
200²
198²
196²
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

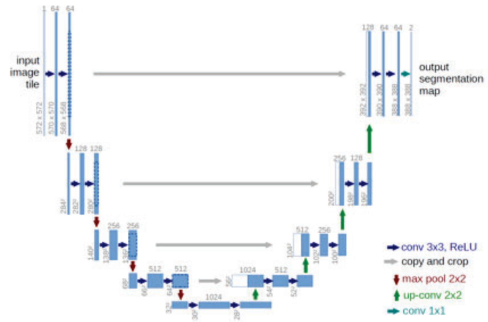
UNet



A fancy way of structure presentation



UNet



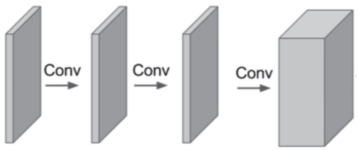
- Contraction Phase
  - Reduce spatial dimension, but increases the “what.”

- Expansion Phase
  - Recovers object details and the dimensions, which is the “where.”

- Concatenating feature maps from the Contraction phase helps the Expansion phase with recovering the “where” information.

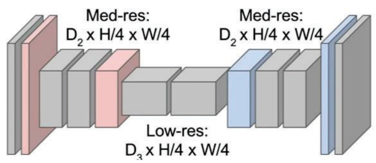
Try to feel why scholars design structures like this

- Like a kid learns to draw an outer man picture



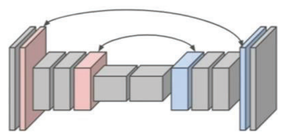
Classic model

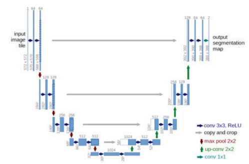
The kid knows little but randomly learn to paint the colors



Autoencoder

Based on the encoder part, the kid at least knows he was drawing an outer man





UNet

The Concatenation of feature maps is like guiding the kid using outlines

Define a UNet in PyTorch

- https://amaarora.github.io/2020/09/13/unet.html

Demonstration

- Unet, Deeplabv3+
- Data preparation for semantic segmentation
- Custom training using DeepLabv3+

- Unet from scratch (should be no problem to understand):
- https://pyimagesearch.com/2021/11/08/u-net-training-image-segmentation-models-in-pytorch/

- A handy Semantic Segmentation toolbox:
- https://github.com/qubvel/segmentation_models.pytorch

- If your project focus on street scenes (directly use deeplabv3+ or transfer learning your own):
- https://github.com/VainF/DeepLabV3Plus-Pytorch

- If you were a facebook/kaiming fan:
- https://github.com/facebookresearch/detectron2