THE UNIVERSITY OF MELBOURNE

Image Segmentation II

Semester 2, 2021

Kris Ehinger

Demo

- https://removal.ai

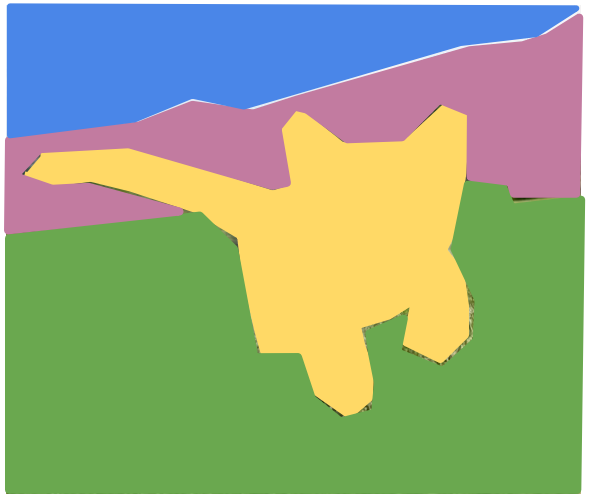
Outline

- Segmentation as classification
- U-Net
- Instance segmentation

Learning outcomes

- Explain how semantic segmentation is performed by fully-convolutional networks
- Implement max unpooling and transposed convolution
- Explain two common architectures for segmentation problems (U-Net and Mask R-CNN)

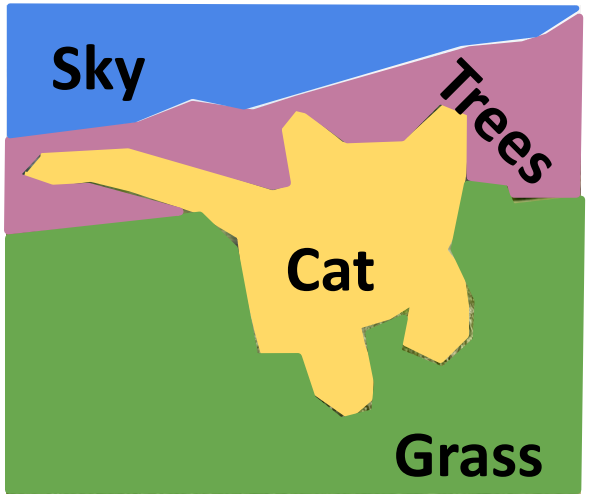
Segmentation

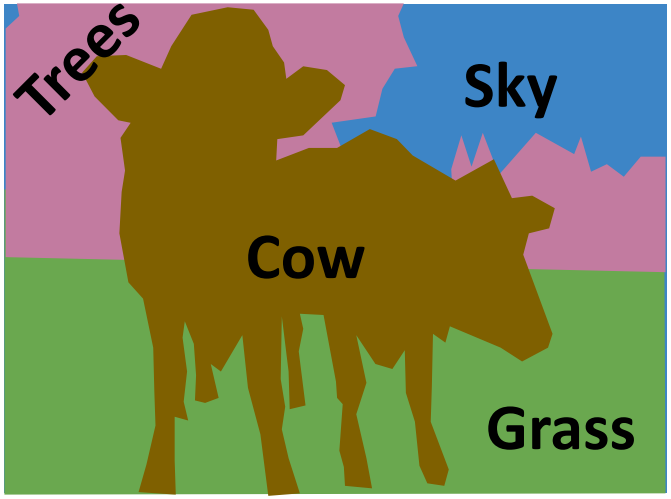


Separate image into different regions (objects, textures)

Image: J. Johnson

Semantic segmentation





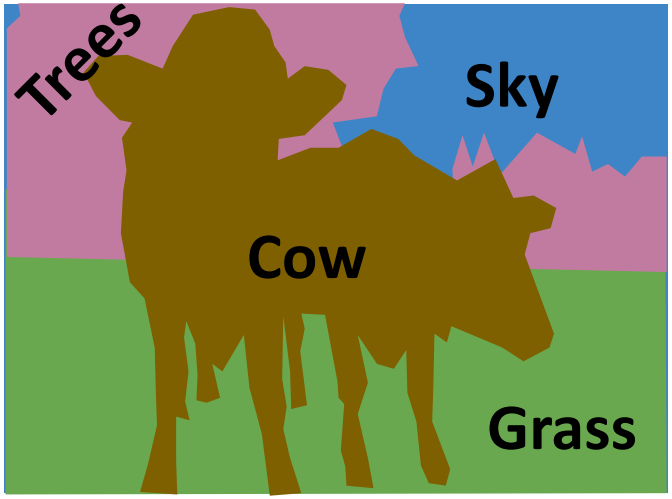
Separate image into different *labelled* regions

Image: J. Johnson

Instance segmentation

Semantic segmentation

Instance segmentation



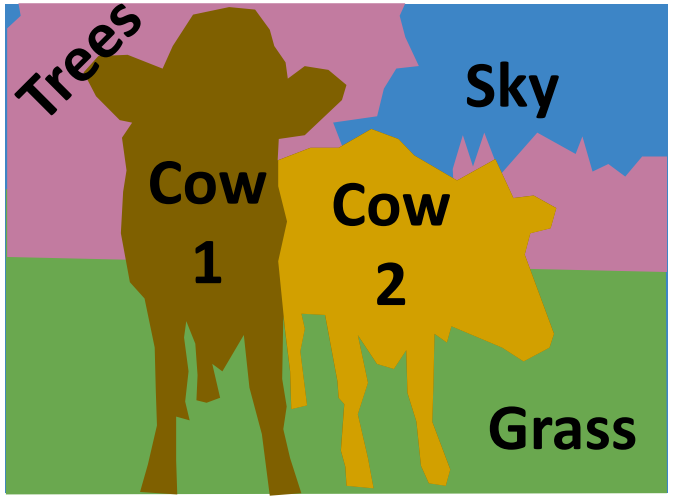
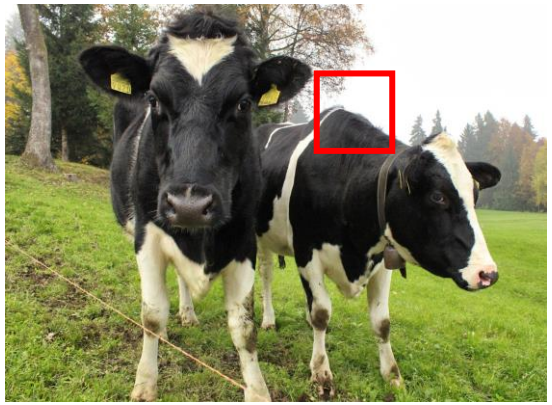
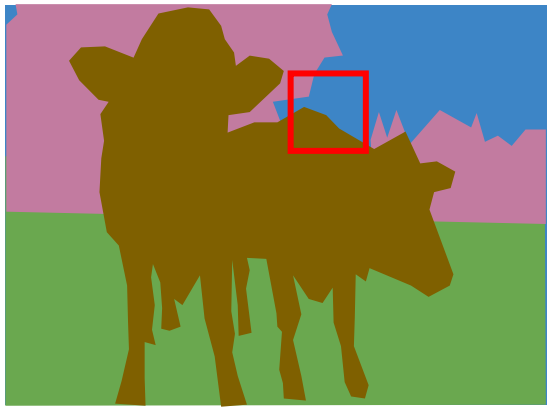


Image segmentation



Input: Image

Clustering?
Graph cuts?
Classification?

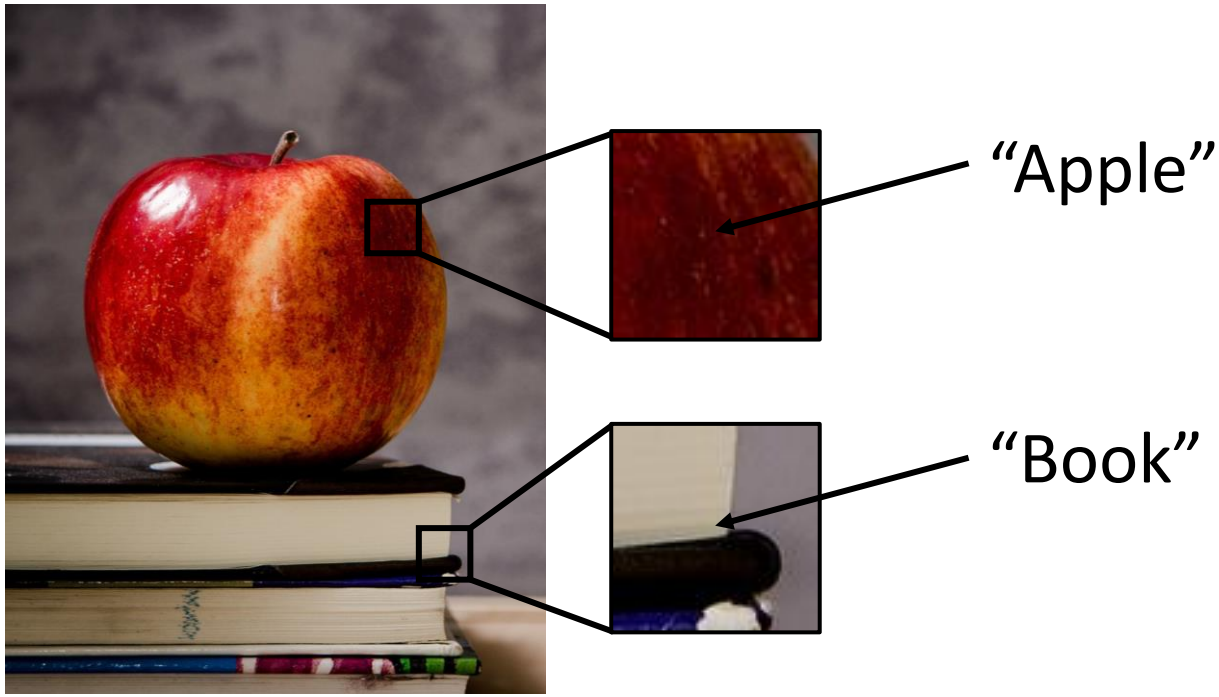


Output: Pixel classification
(and, optionally, labels)

Segmentation as classification

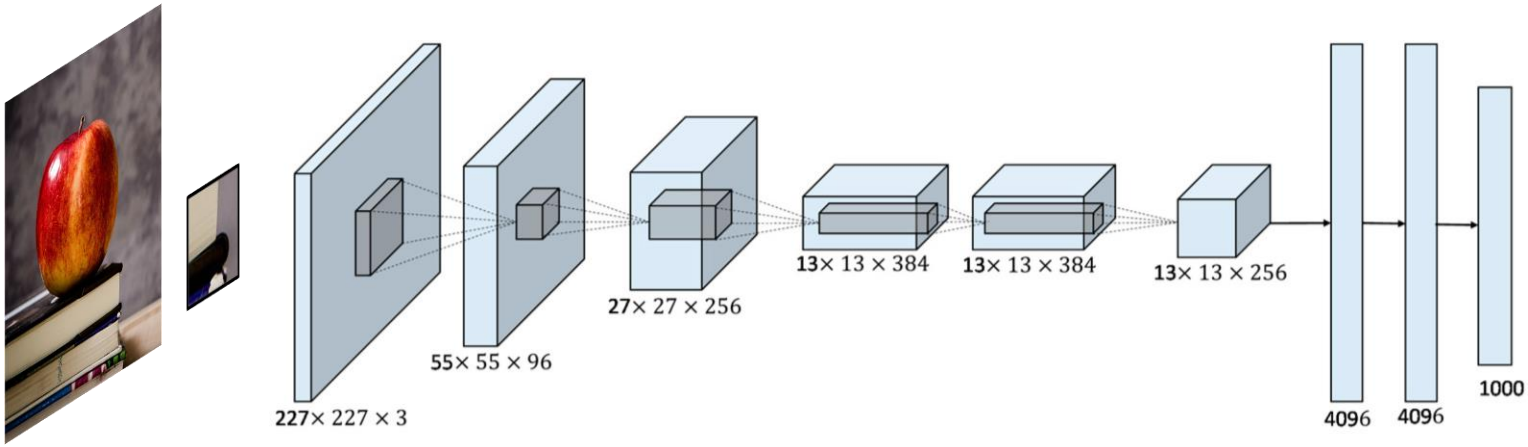
Pixel classification

- Image segmentation as a classification problem
- Given a window (NxN pixels), classify central pixel

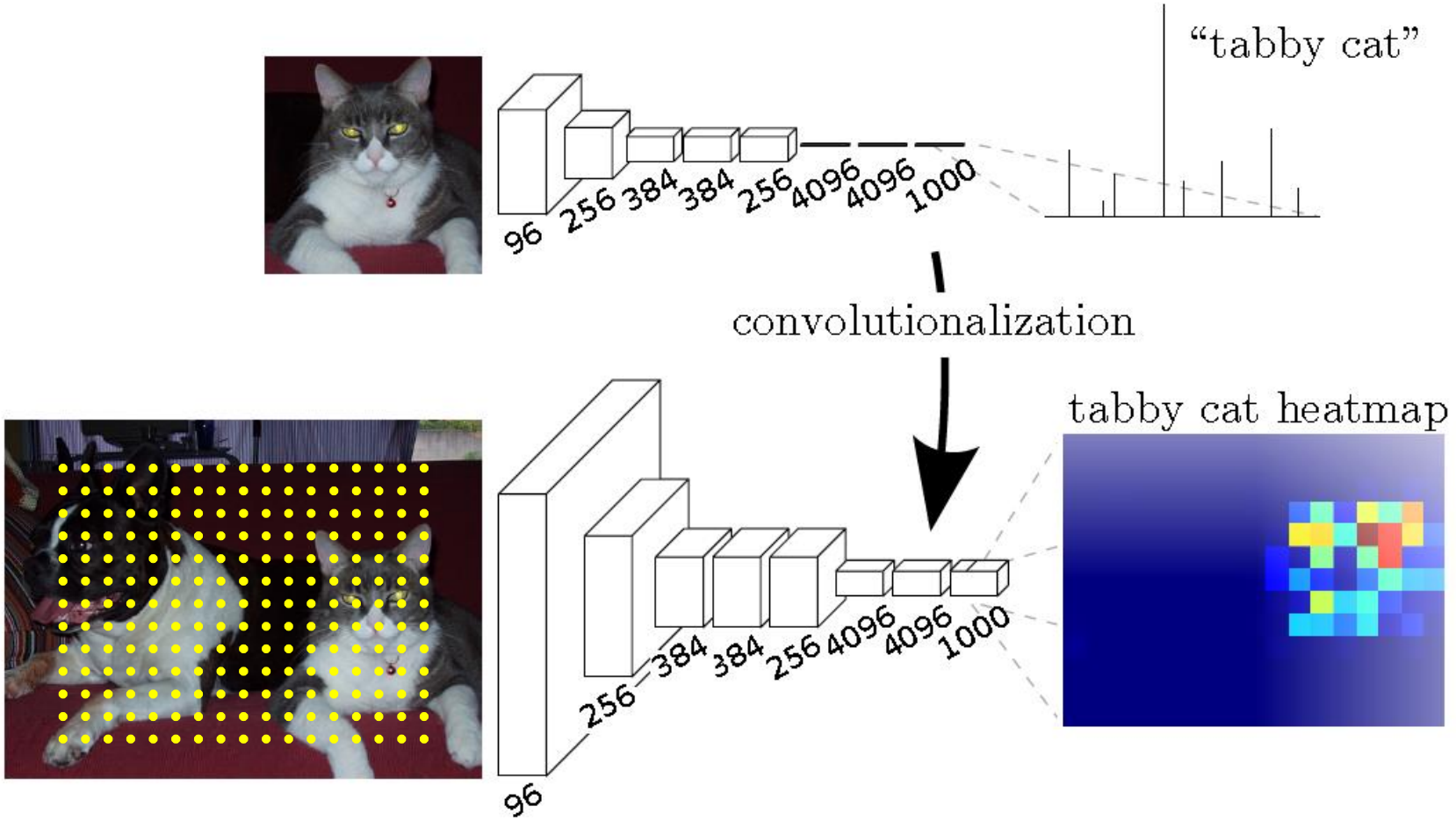


Pixel classification

- Classifying individual pixels is potentially very slow (e.g., a small image = 600x800 = 480,000 pixels)
- But a CNN can classify multiple pixels in parallel



Parallel patch classification



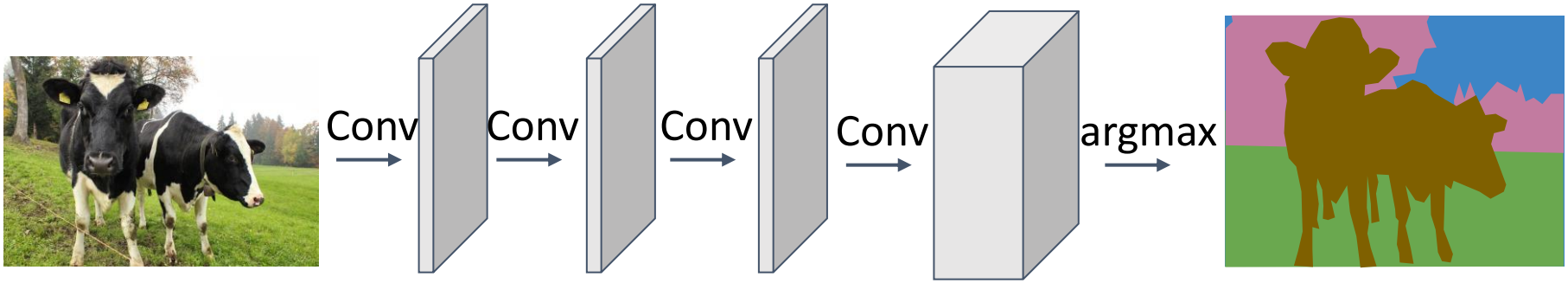
Long, Shelhamer, & Darrell (2015)

Fully-convolutional network

- Fully-connected network (FCN) has only convolutional layers, no fully-connected layers
- Last layer is a spatial map
- Can accept any size image as input, output map size depends on input size

Parallel patch classification

- Standard CNN architecture poses two problems:
  - Receptive field size is linear with the number of convolutional layers
  - Most methods downsample (e.g., with maxpooling) to reduce computation



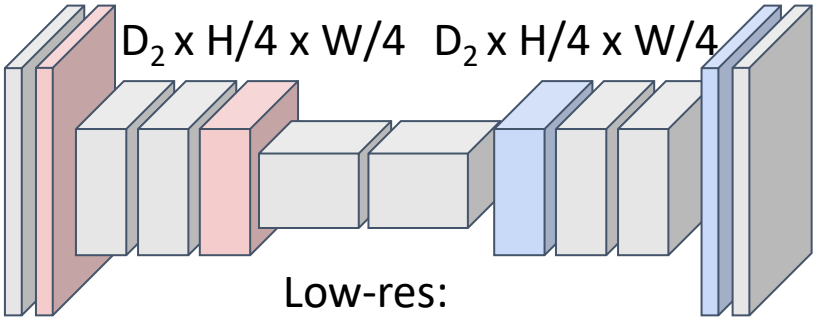
We could avoid maxpooling, but this will make the network slow and doesn't solve the first problem

Image: J. Johnson

Parallel patch classification

- Solution: use encoder-decoder structure
- Encoder downsamples image, decoder upsamples

Input:
3 x H x W



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

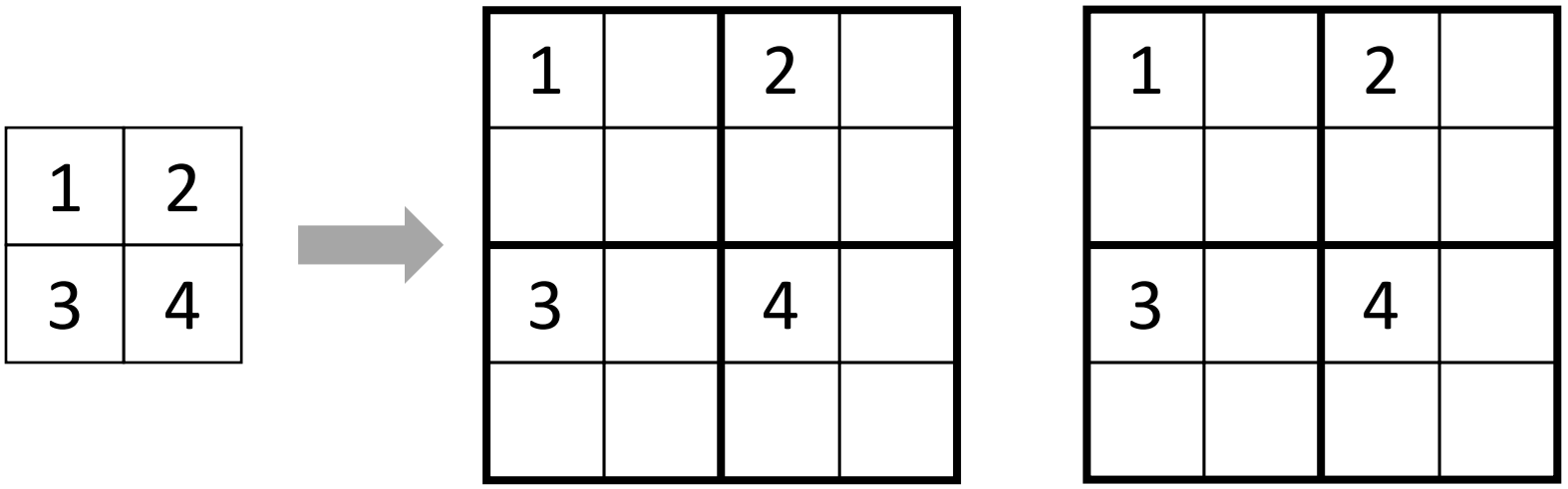
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

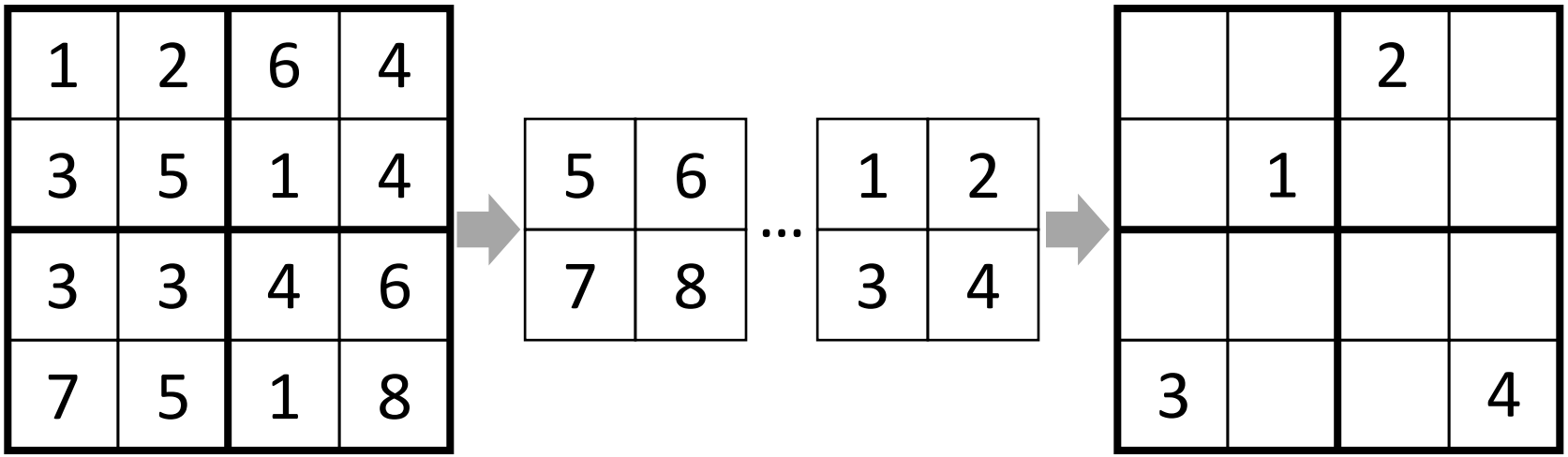
Image: J. Johnson

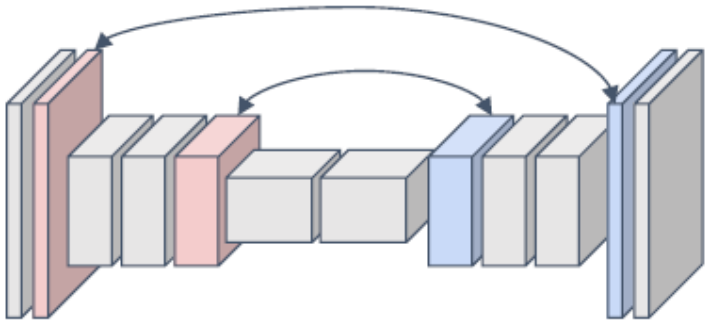
Upsampling

- How to upsample a feature layer in a CNN?



Max Unpooling



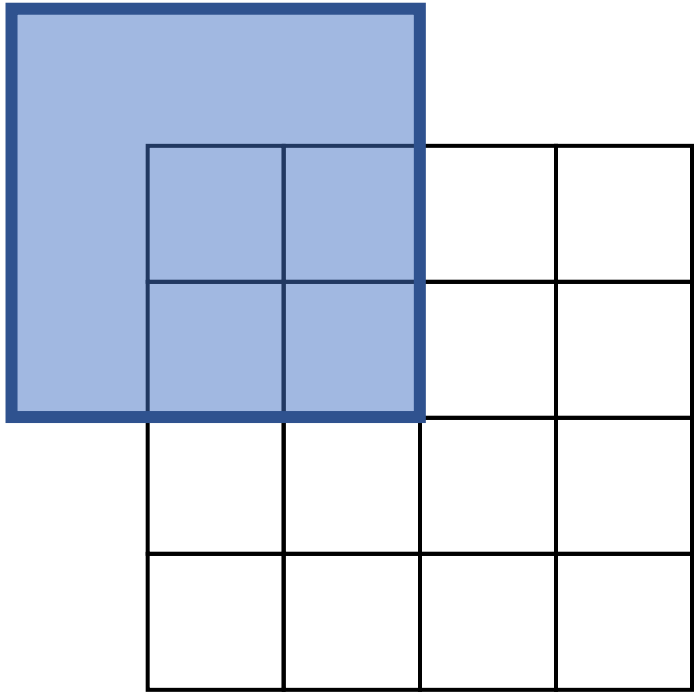


Each upsampling layer is paired with a downsampling layer
The locations of the max items are saved and passed to upsampler

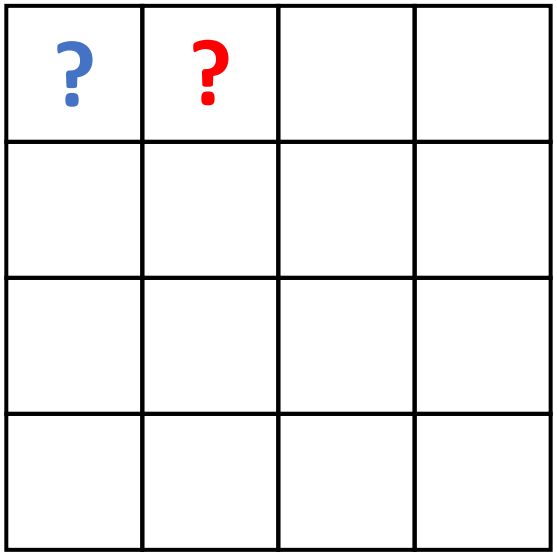
Image: J. Johnson

Convolution review

kernel = 3x3, stride = 1, pad = 1



Input: 4 x 4

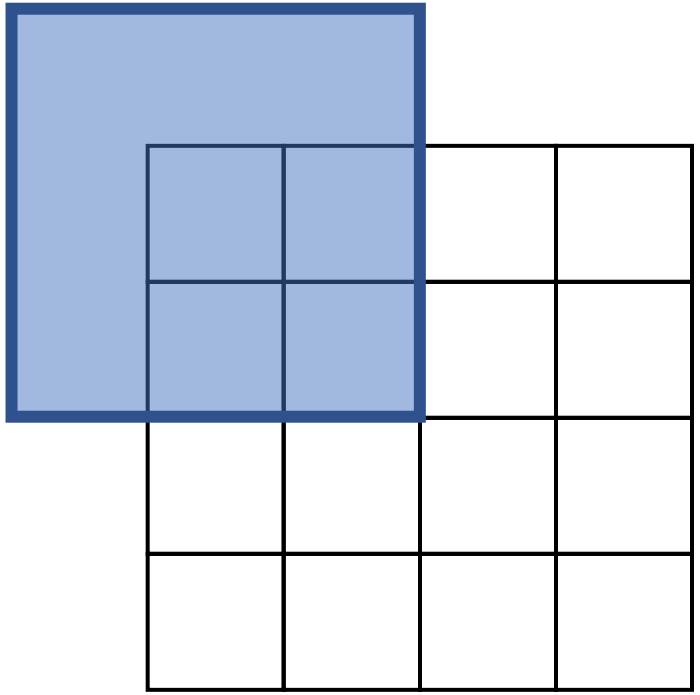


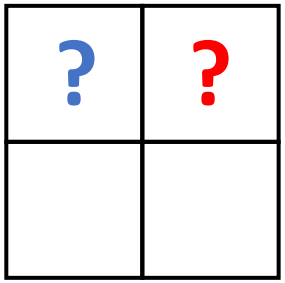
Output: 4 x 4

Based on slide by: J. Johnson

Convolution review

kernel = 3x3, stride = 2, pad = 1





Input: 4 x 4

Output: 2 x 2

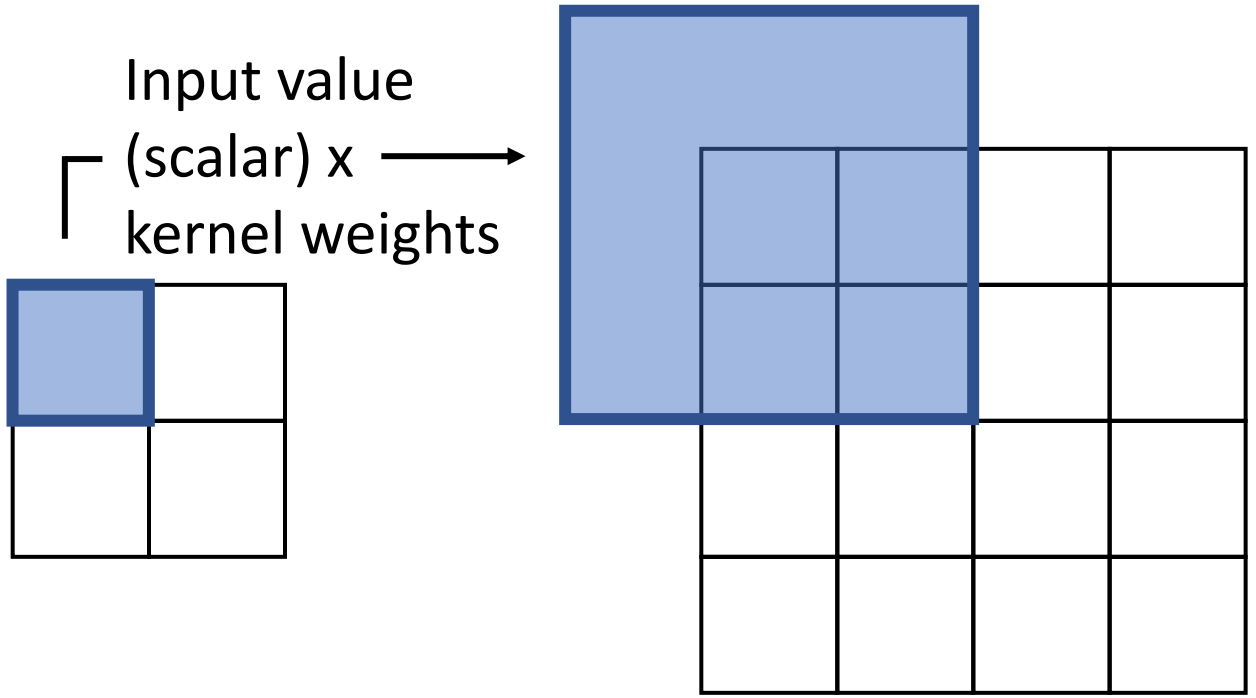
Based on slide by: J. Johnson

Transposed convolution

- Convolution with stride > 1 does a form of downsampling
  - E.g., stride = 2 means filter moves 2 pixels in input for every 1 pixel in output
  - “Learnable downsampling”
- Can we reverse this to do upsampling?
  - E.g., filter moves 2 pixels in *output* for every 1 pixel in *input*
- **Transposed convolution**: convolution with a stride < 1
  - In some papers, may also be called deconvolution, upconvolution, fractionally strided convolution

Transposed convolution

3x3 transposed convolution, stride = 2



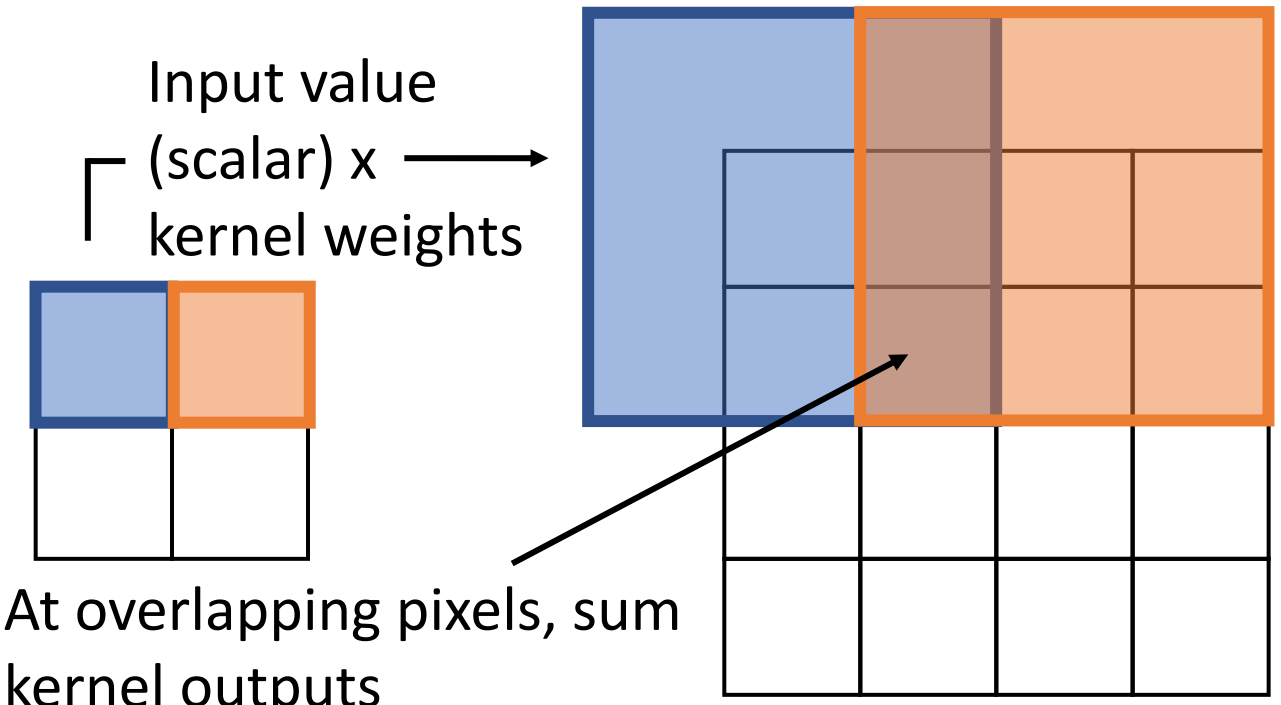
Input: 2 x 2

Output: 4 x 4

Based on slide by: J. Johnson

Transposed convolution

3x3 transposed convolution, stride = 2

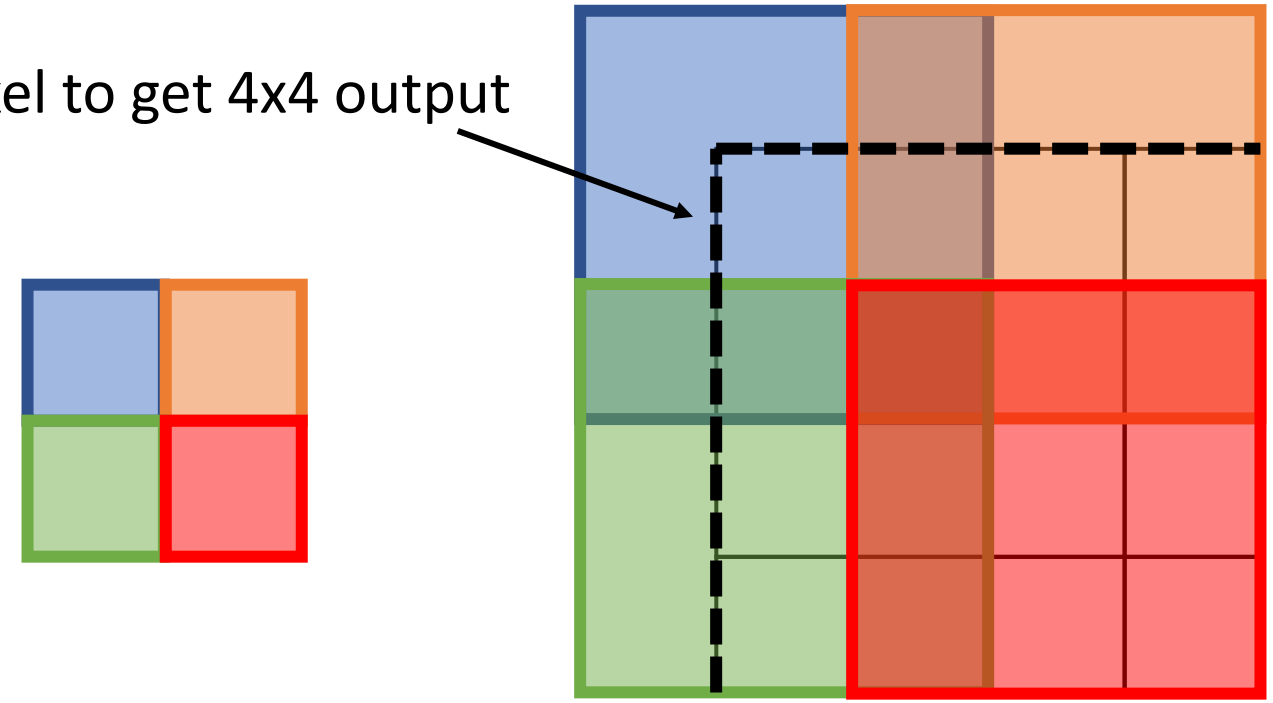


Based on slide by: J. Johnson

Transposed convolution

3x3 transposed convolution, stride = 2

Trim 1 pixel to get 4x4 output

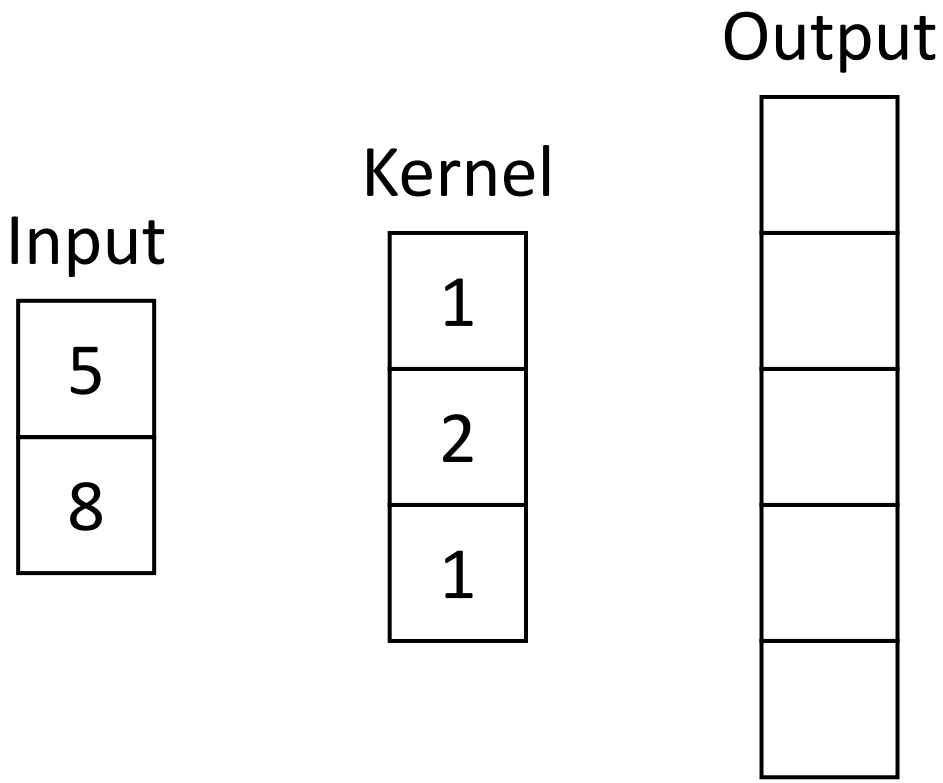


Input: 2 x 2

Output: 4 x 4

Based on slide by: J. Johnson

Practice: 1D example



PollEv.com/
krisehinger432

Drag the
correct values
into spaces 1-5

Why is it transposed convolution?

We can express convolution as a matrix multiplication:

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

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

kernel=3, stride=1, pad=1

Transposed convolution multiplies by the transpose of the same matrix:

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

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

kernel=3, stride=1, pad=1

Based on slide by: J. Johnson

Why is it transposed convolution?

We can express convolution as a matrix multiplication:

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

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

kernel=3, stride=2, pad=1

Transposed convolution multiplies by the transpose of the same matrix:

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

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

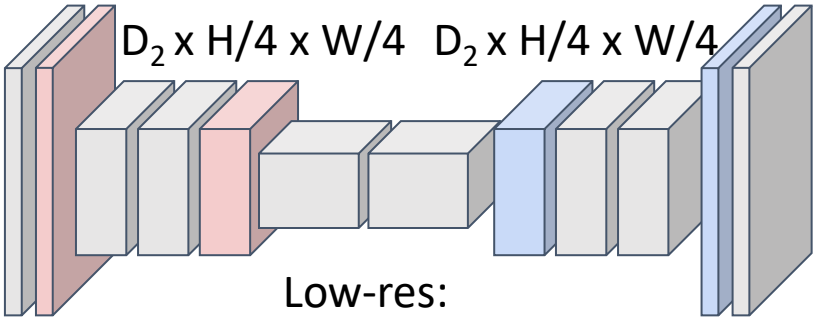
kernel=3, stride=2, pad=1

Based on slide by: J. Johnson

Parallel patch classification

- Solution: use encoder-decoder structure
- Encoder downsamples image, decoder upsamples

Input:
3 x H x W



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

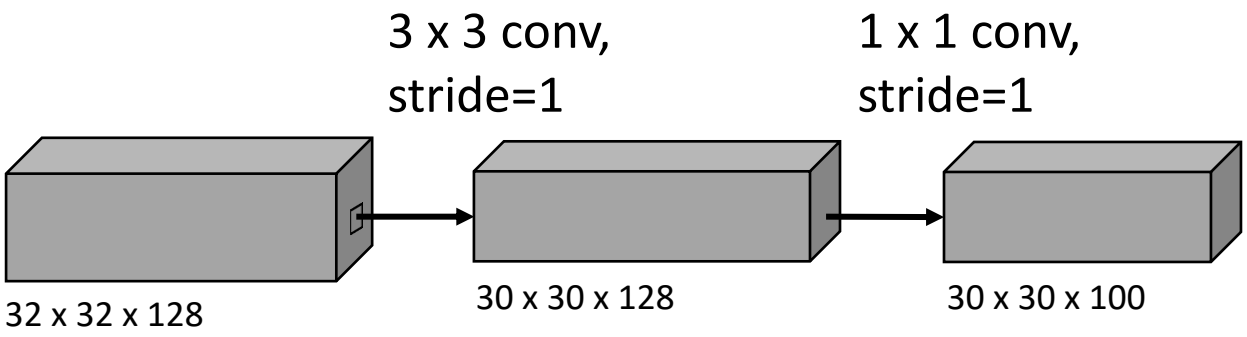
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Image: J. Johnson

1x1 convolution layer

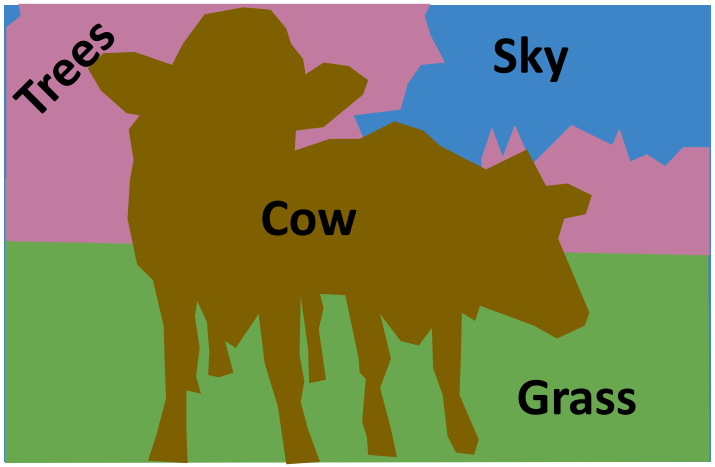
- Convolutional layer with a 1x1 kernel
- Commonly found as last layer(s) of a fully-connected network
- What do these kernels learn?



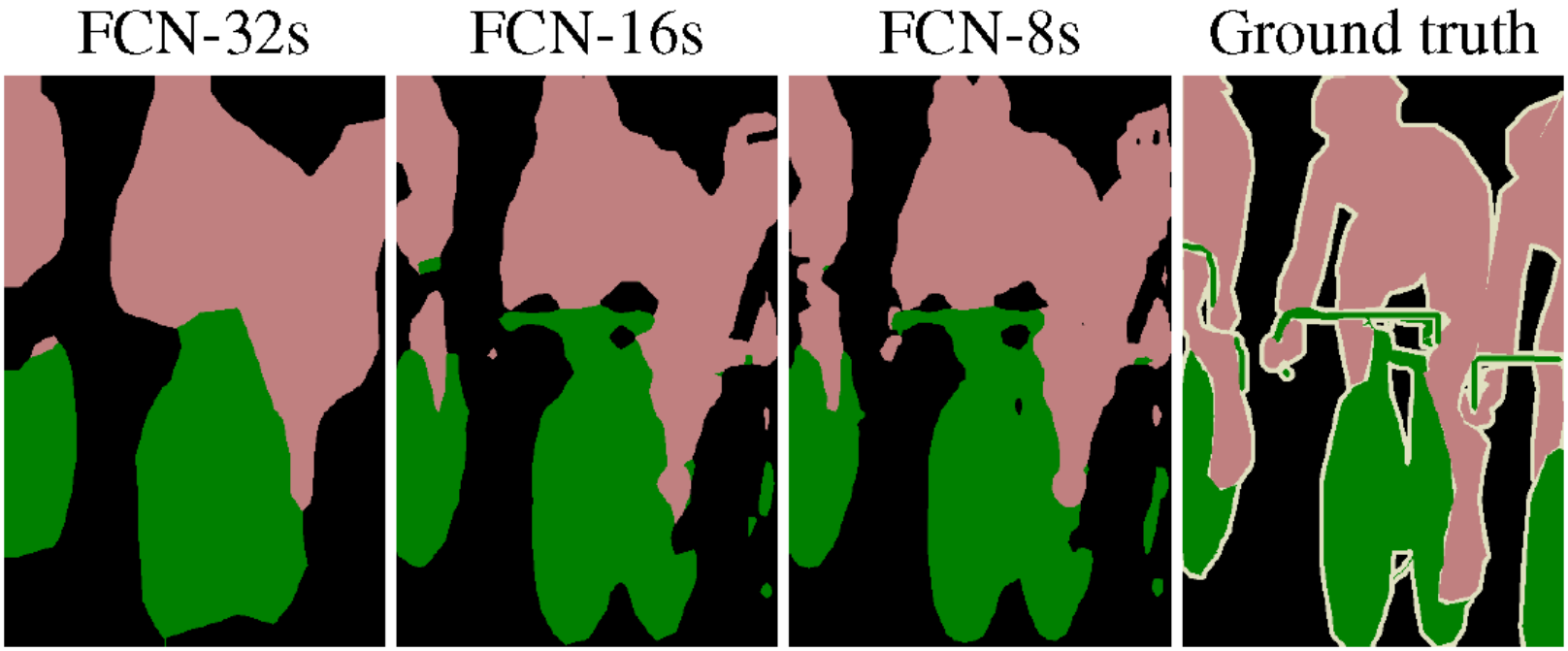
Loss function

- At each pixel, compute cross-entropy loss between predicted class and known classes

$$E = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$



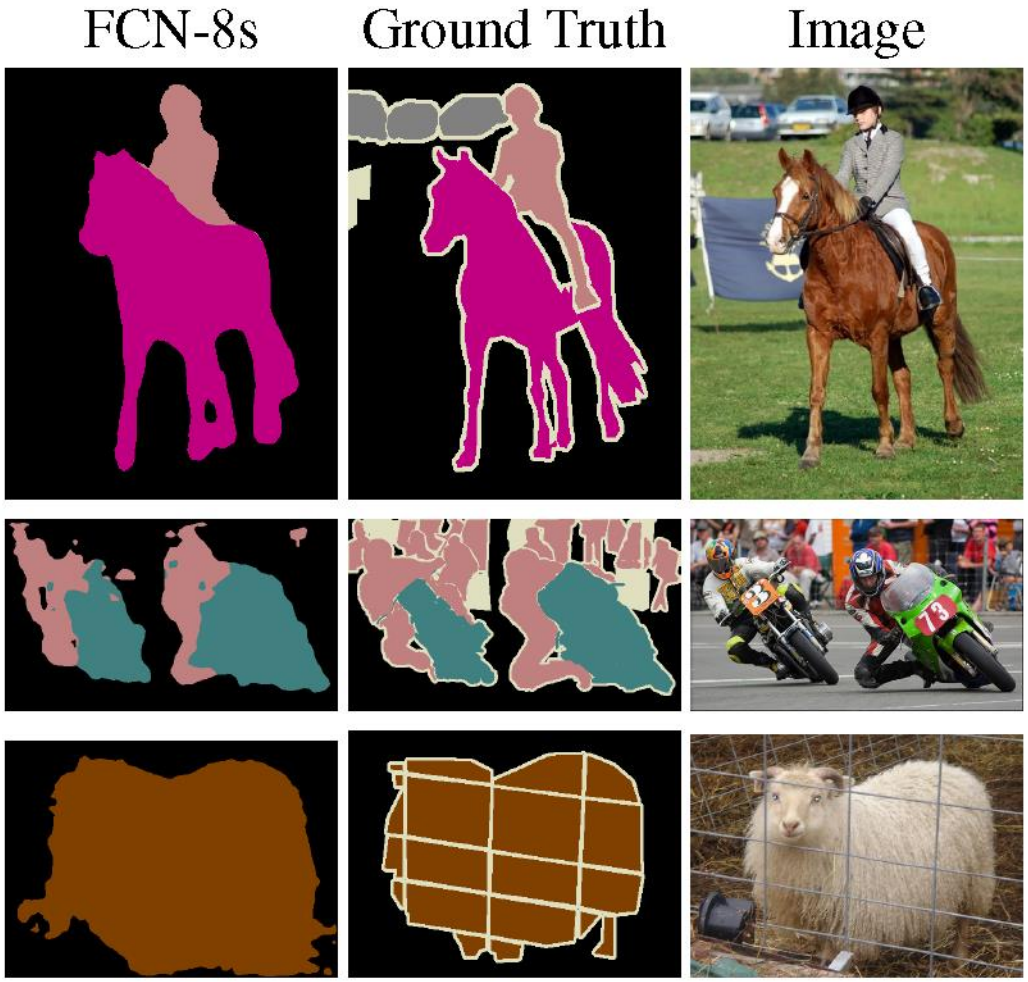
Segmentation with FCN results



Decreasing stride of convolution→

Long, Shelhamer, & Darrell (2015)

Segmentation with FCN results



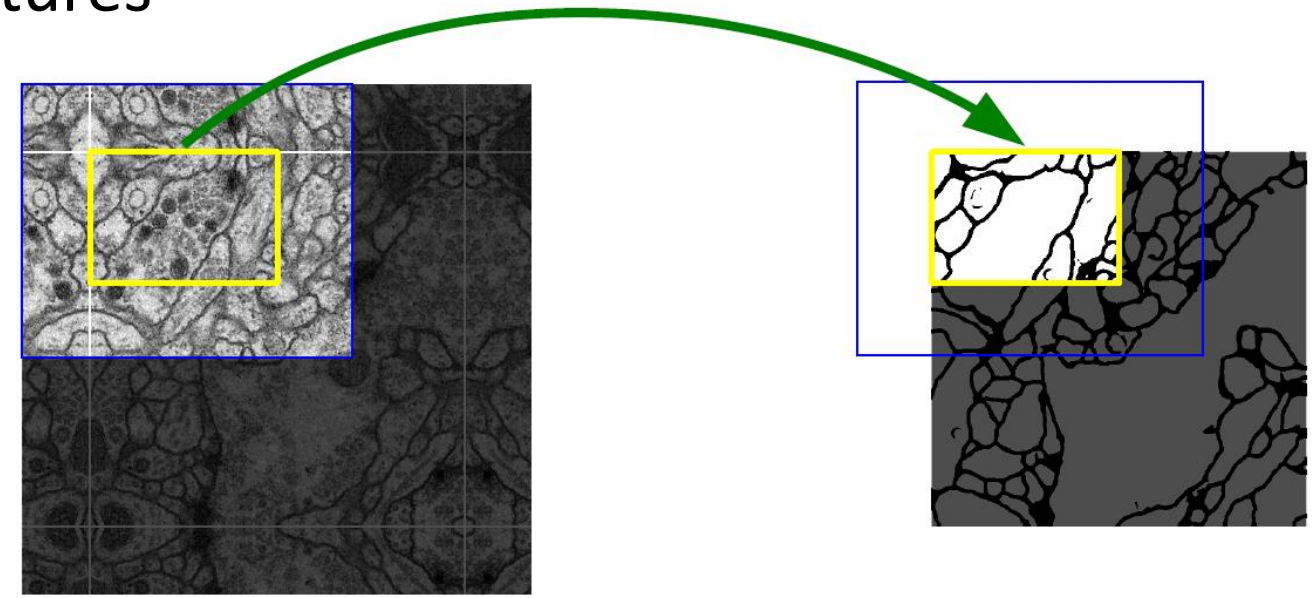
Summary

- Semantic segmentation can be treated as a pixel classification problem
- This can be done efficiently with a fully-convolutional network
- Encoder-decoder architecture:
  - Downsample with max pooling, strided convolution
  - Upsample with max unpooling, transposed convolution
- Output is a label for each pixel

U-Net

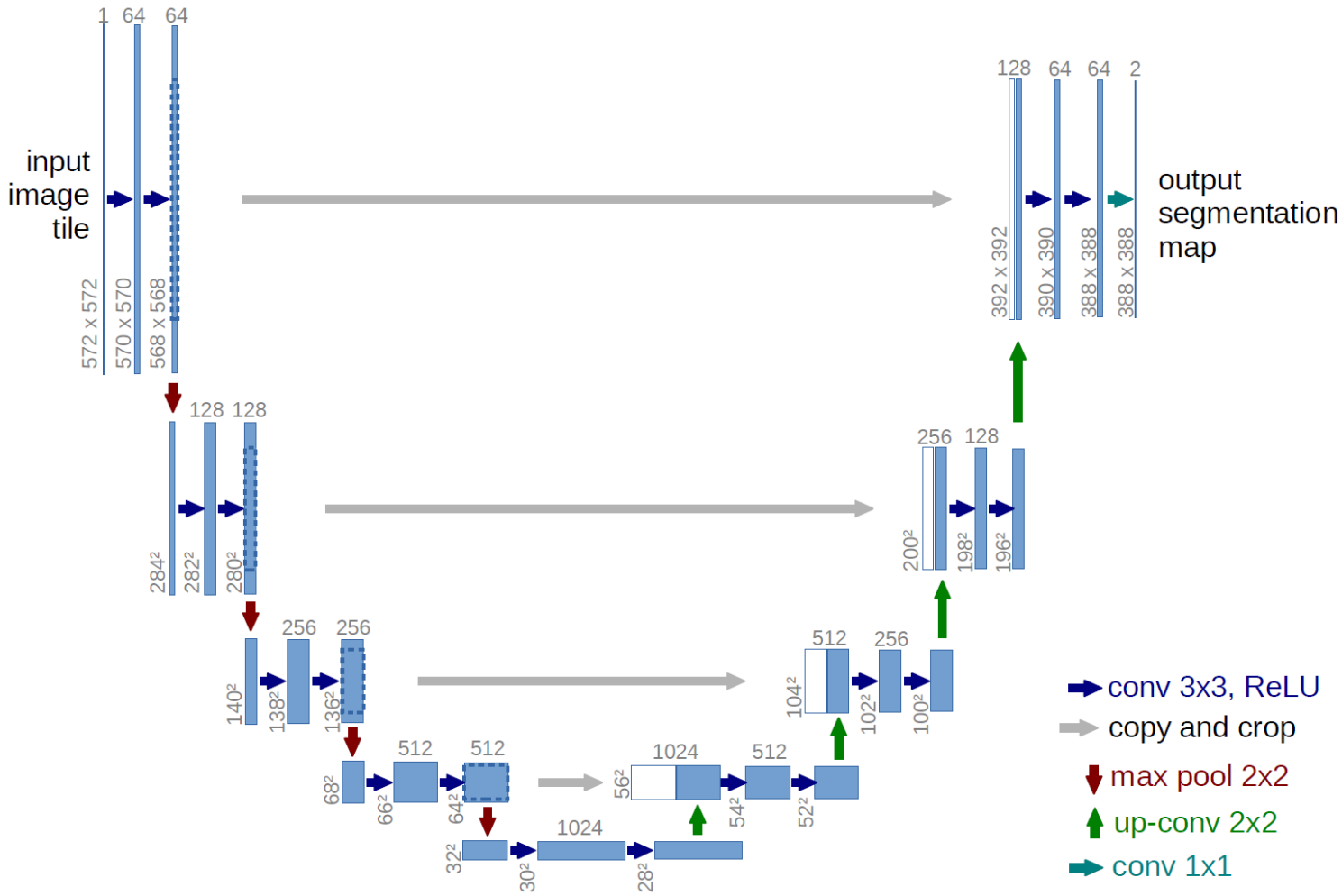
U-Net

- Originally proposed for medical image segmentation
- Encoder-decoder structure with some additional features



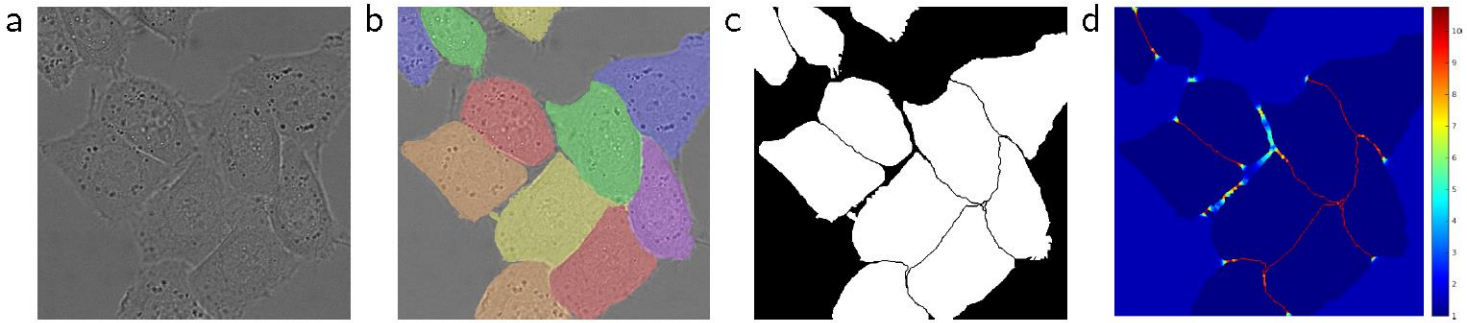
Ronneberger, Fischer, & Brox (2015)

U-Net architecture

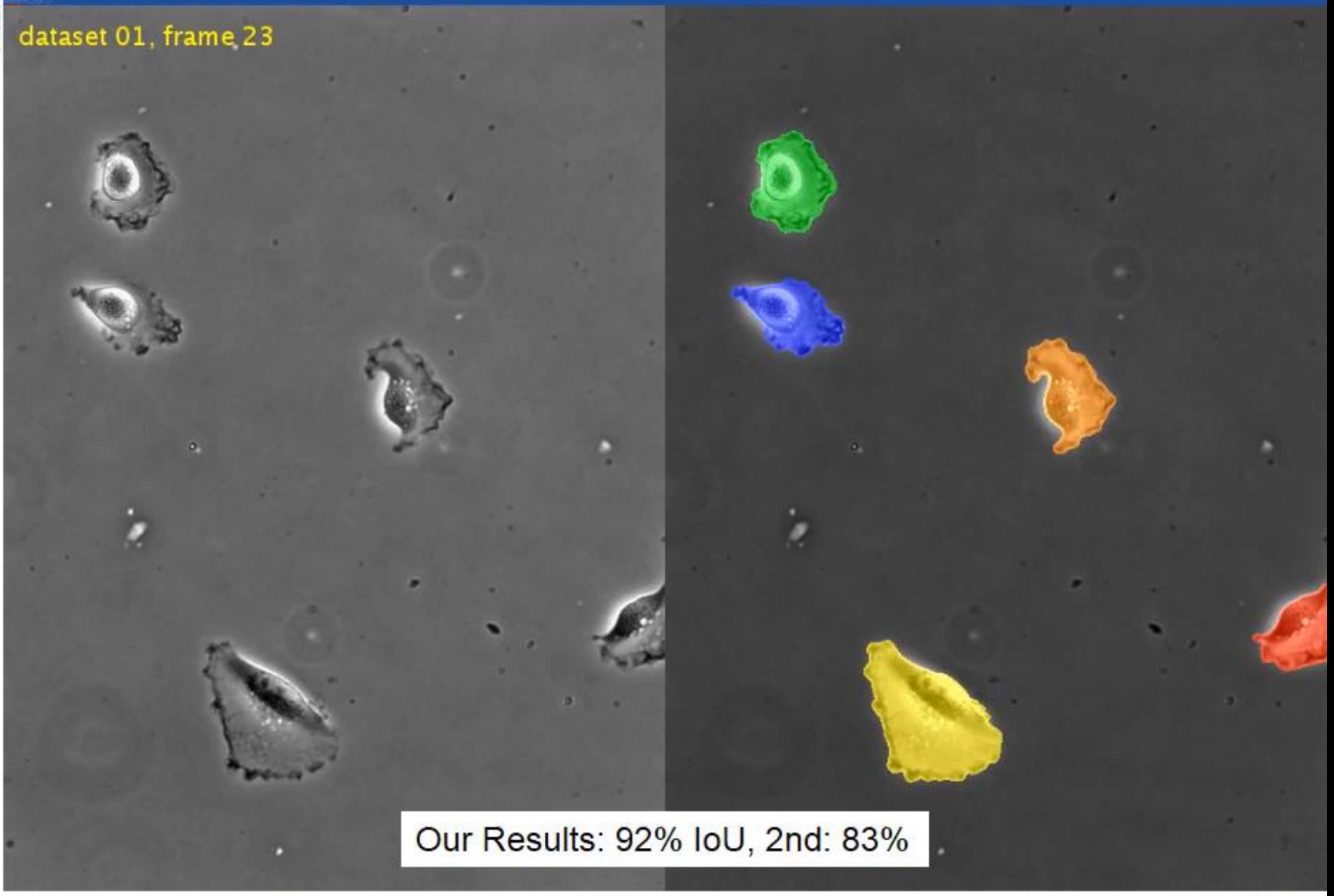


U-Net architecture

- In decoder, the upsampled feature map is concatenated with the original features from the corresponding encoder layer
- For cell segmentation, U-Net uses only 2 classes and weighted cross-entropy loss: edges between cells have higher weight



ISBI cell tracking challenge: **PhC-U373**



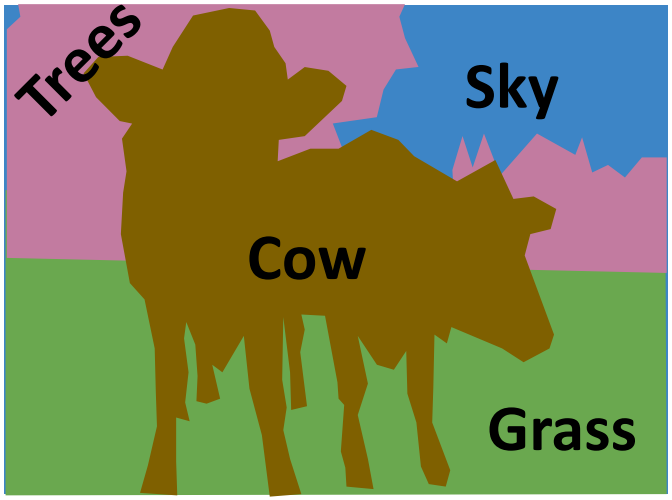
Summary

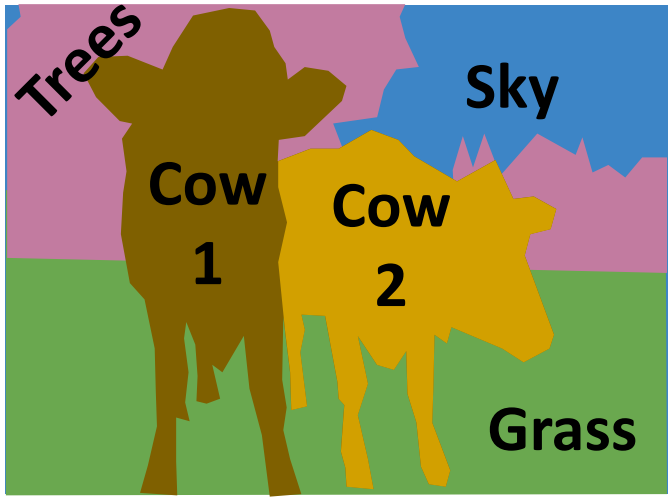
- U-Net: fully-convolutional network for segmentation, with some modifications
- Originally for medical image analysis, but commonly used for other segmentation tasks
- Also used as an encoding-decoding network for tasks like:
  - Image denoising
  - Image inpainting

Instance segmentation

Instance segmentation

- Semantic segmentation classifies pixels, doesn't distinguish between instances
- How to separate instances?





Side note: Stuff and things

- Visual scenes include both “things” and “stuff”
  - “Things” = objects, “countable nouns” (cow, person, car)
  - “Stuff” = regions/materials, “mass nouns” (road, grass)
  - Some classes can be either (trees)
- Instance segmentation (and computer vision in general) mainly focusses on “things”

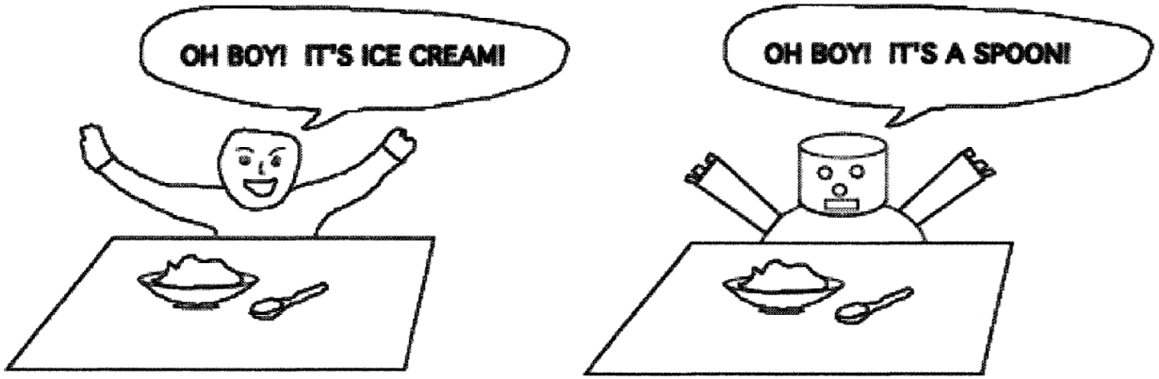


Image: Adelson (2001) “On seeing stuff: The perception of materials by humans and machines”

Instance segmentation

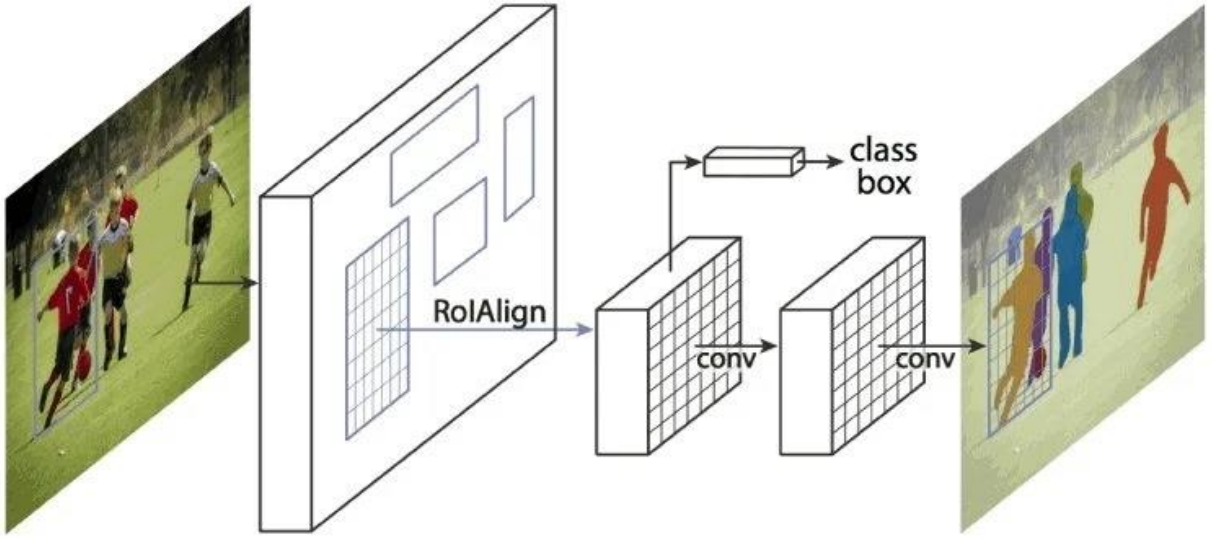
- Instead of giving each pixel an instance label, extract patches of image that are likely to be separate instances
- Do segmentation within each patch
- Commonly-used architecture: Mask R-CNN

R-CNN

- R-CNN = Region-based convolutional neural network
- Efficiently extracts “regions of interest” (image patches likely to contain objects) for further processing
- (We’ll cover these networks in more detail next week)

Mask R-CNN

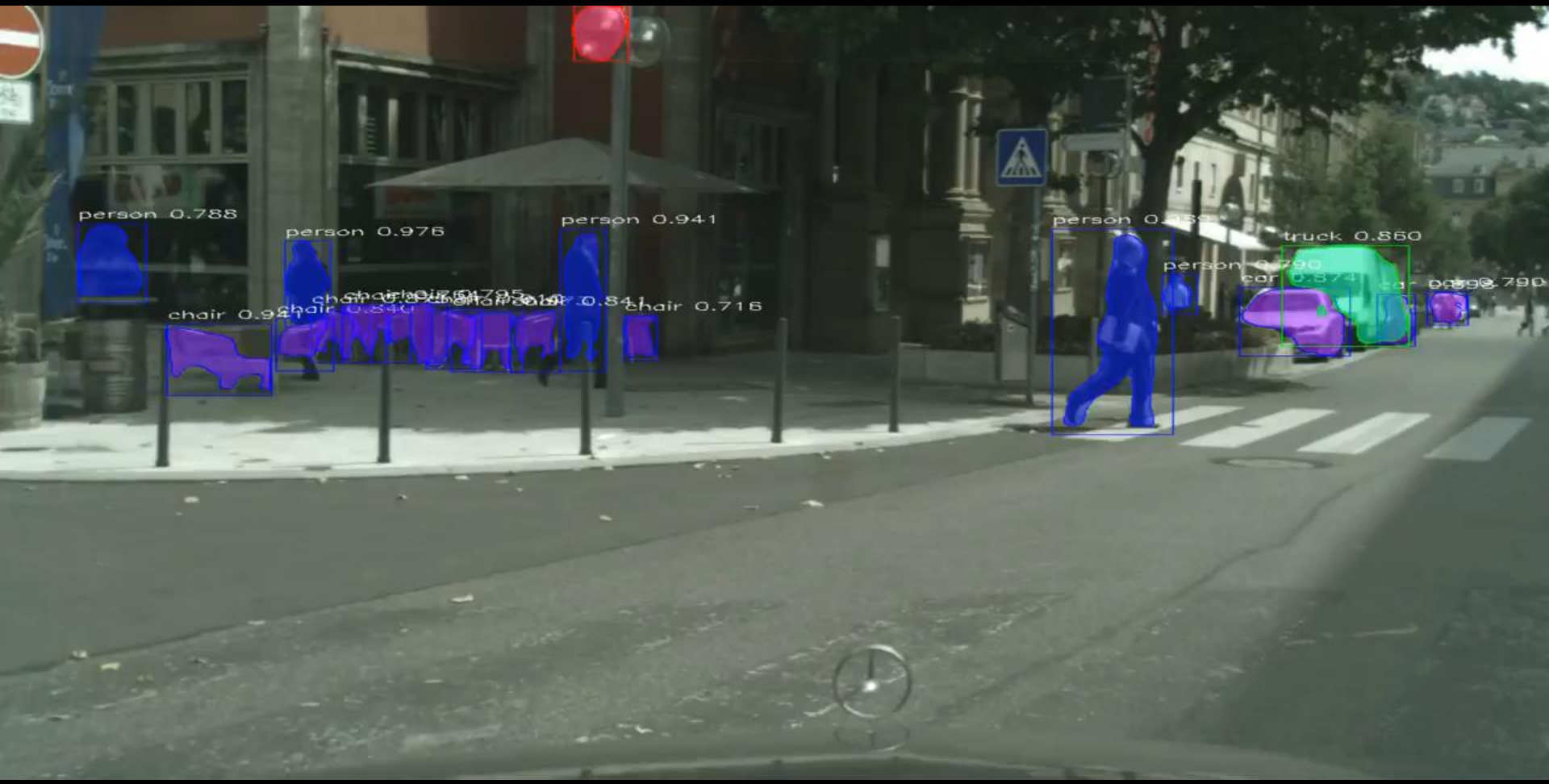
- Mask R-CNN takes patches extracted by R-CNN and runs them through a fully-convolutional network
- FCN predicts a binary segmentation mask ("object" or "background")



He, Gkioxari, Dollár, & Girshick (2017)

Mask R-CNN results

He, Gkioxari, Dollár, & Girshick (2017)



https://www.youtube.com/watch?v=EfaCtOEEmLA

Summary

- Instance segmentation gives separate labels to each instance of a class, not just class labels
- Generally only applies to countable “thing” classes
- Typical methods detect object patches, then do binary segmentation within the patch

Summary

- Segmentation can be approached in various ways – clustering/graph-based methods vs. pixel classification with FCNs
- Advantages of FCNs
  - Better able to handle very complex objects/backgrounds
  - Likely to give better results for the classes on which they are trained
- Disadvantages of FCNs
  - Tend to be worse at capturing precise boundary details
  - May not generalise to classes outside their training set