Computer vision and imaging

From *classical* to *deep learning-based* methods in Computer Vision

Week 6, Lecture 1

Aleš Leonardis

Office: UG 34

Last week

- From *classical* to machine ***(deep) learning-based*** **computer vision** approaches
- A step-by-step progress towards an end-to-end deep neural network
  - Design principles
  - Design choices
- We now know the building blocks of convolutional neural networks

Plan for today

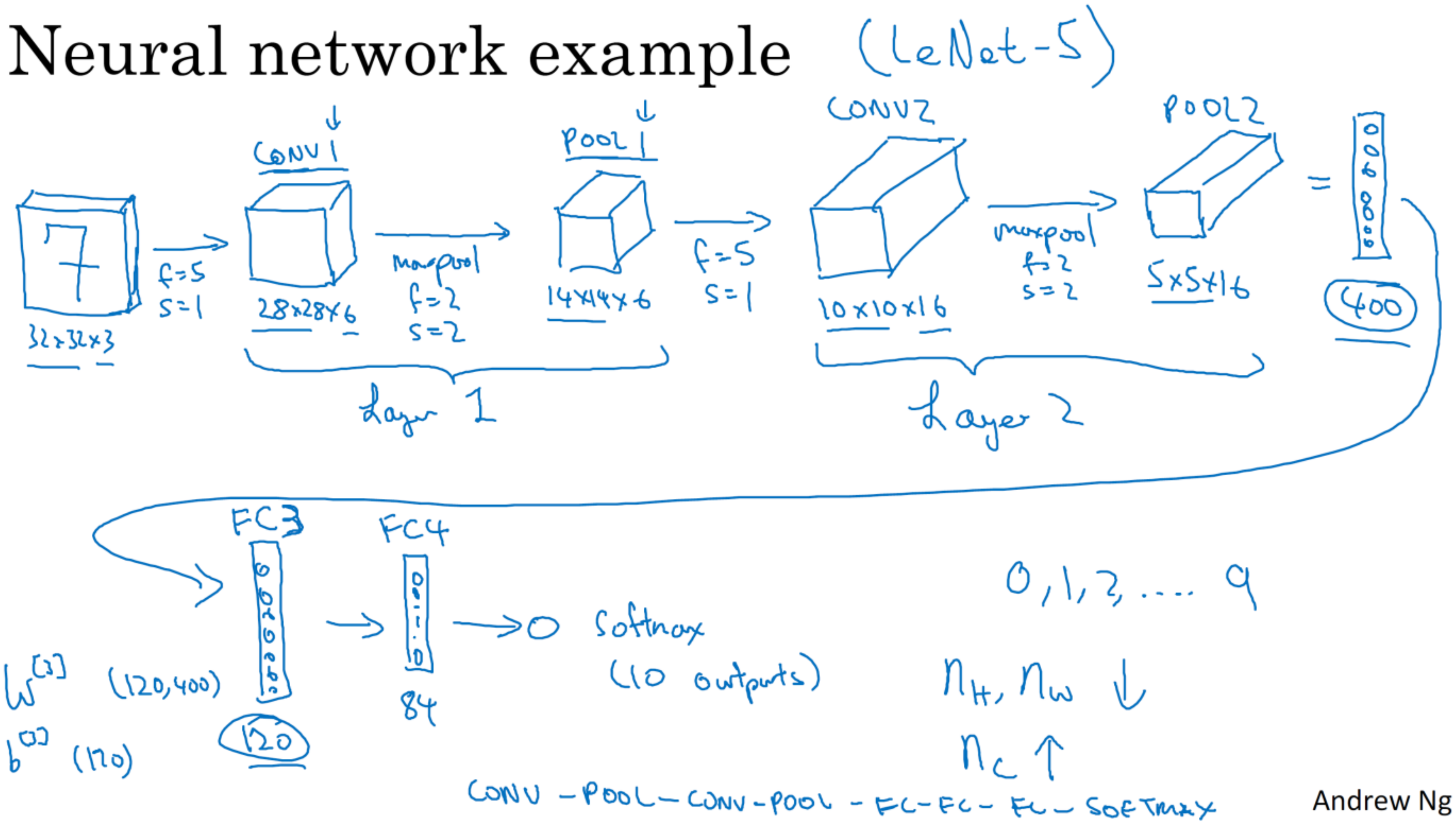
- We will build an example of a convolutional neural network (activation shape, activation size, parameters)
- We will look at the main properties of the convolutional neural networks and what makes them distinctive
  - Parameter sharing, sparsity of connections, receptive field, visualisation (interpretability)

- Then we will briefly revisit the classical computer vision models for segmentation and show how segmentation can be achieved with deep neural networks

Plan for the Week 6, lecture 2

- We will revisit the PCA method and show how it can be related to deep learning model of autoencoders
- We will summarise the overall insights and contrast the two methodological approaches (domain of applicability)

Convolutional neural network example



Andrew Ng

Convolutional neural network example

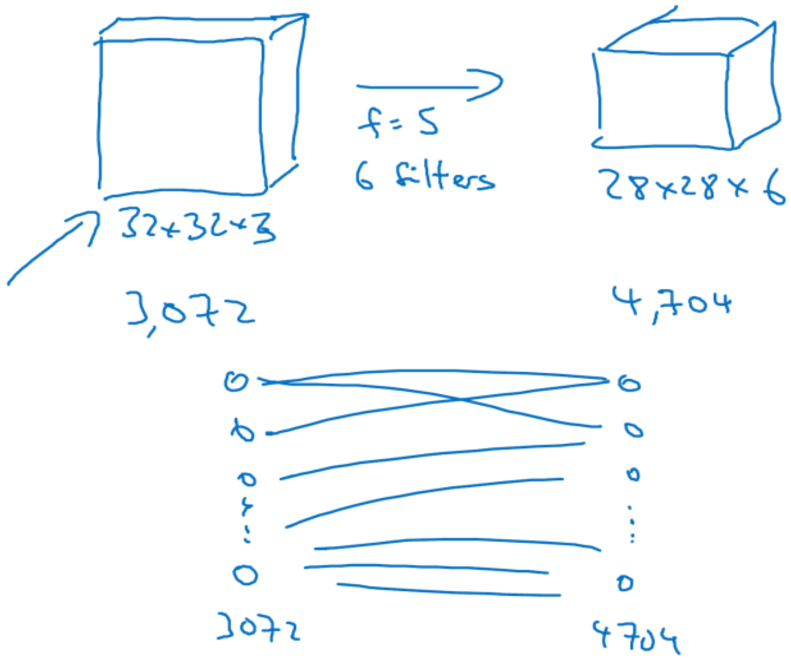
Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	608
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	3216
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48120
FC4	(84,1)	84	10164
Softmax	(10,1)	10	850

Andrew Ng

Why convolutions?

Why convolutions



5×5 — 25

26

6 × 26 = 156 parameters

3,072 × 4,704 ≈ 14M

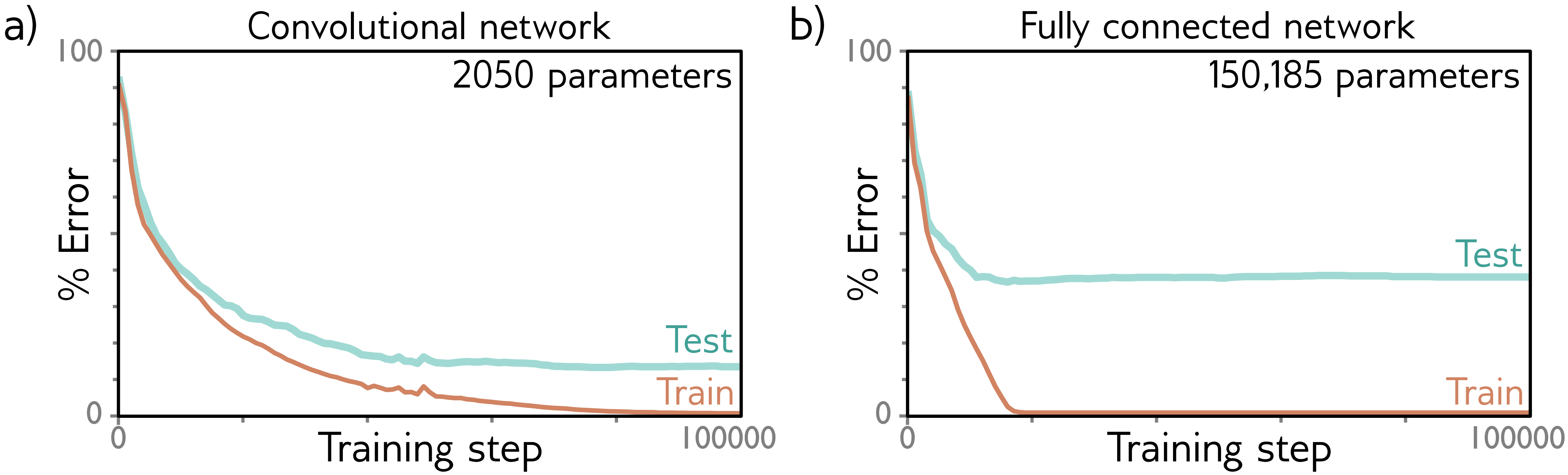
Andrew Ng

Networks for images

- Problems with fully-connected networks

1. Size
   - 224x224 RGB image = 150,528 dimensions
   - Hidden layers generally larger than inputs
   - One hidden layer = 150,520x150,528 weights -- 22 billion
2. Nearby pixels statistically related
   - But could permute pixels and relearn and get same results with FC
3. Should be stable under transformations
   - Don’t want to re-learn appearance at different parts of image

Performance



Convolutional networks

- Parameters only look at local image patches
- Share parameters across image

Why convolutions?

Why convolutions

translation invariance

$$
\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3\times 3} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}
$$

Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Receptive fields

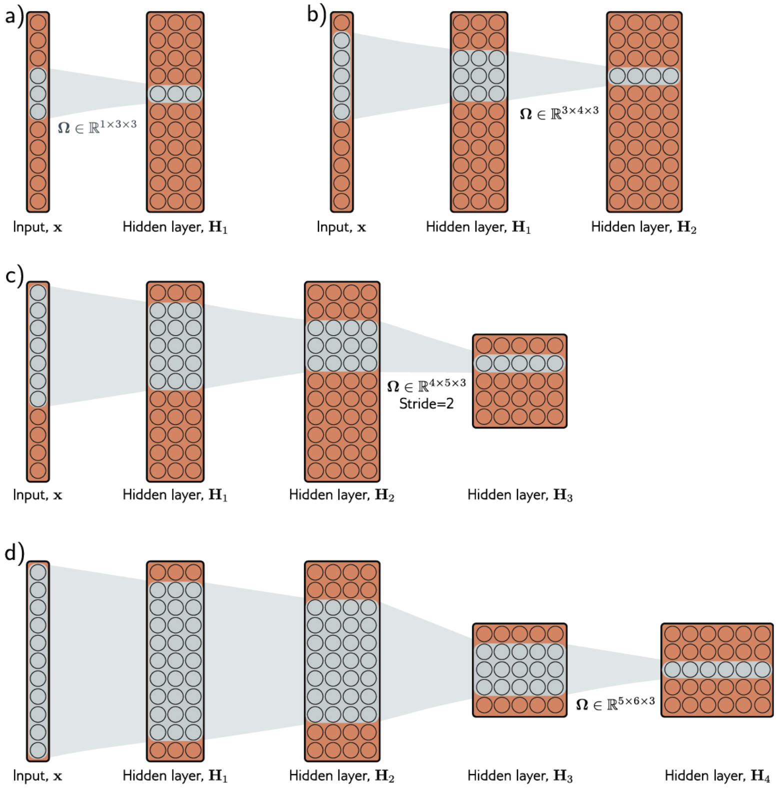
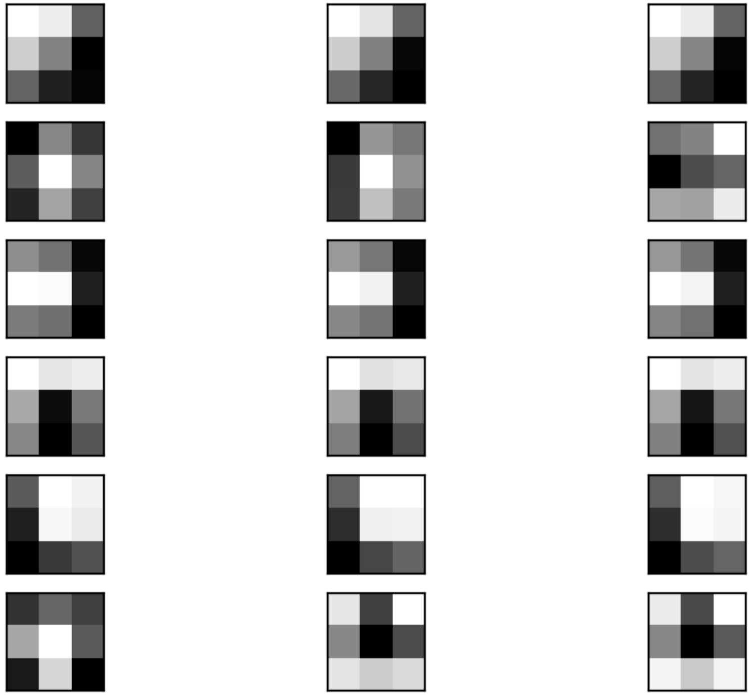
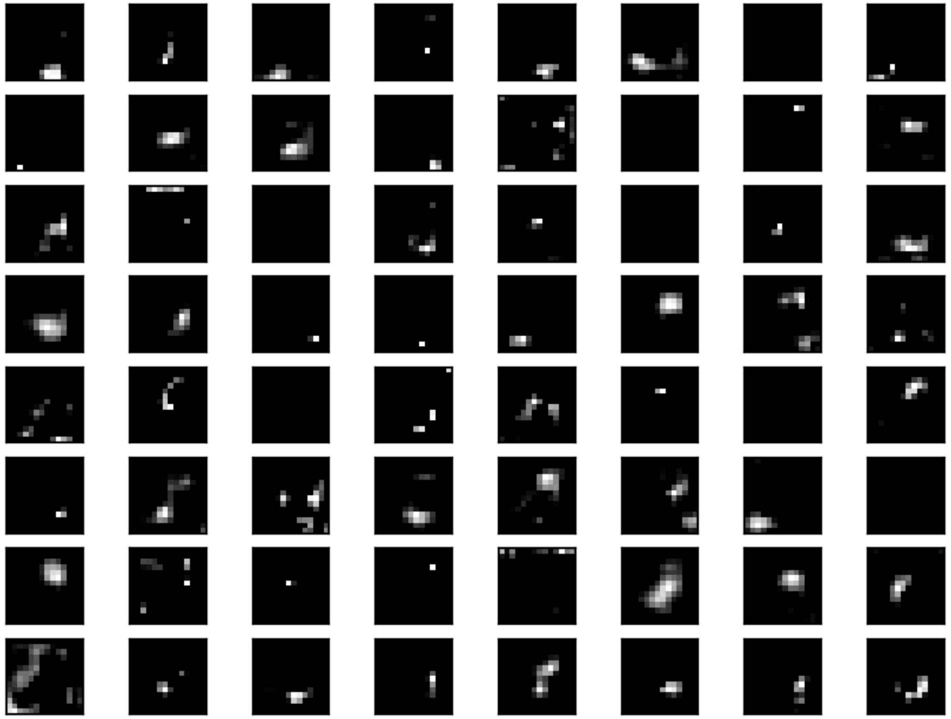


Figure 10.6 Receptive fields for network with kernel width of three. a) An input with eleven dimensions feeds into a hidden layer with three channels and convolution kernel of size three. The pre-activations of the three highlighted hidden units in the first hidden layer $\mathbf{H}_1$ are different weighted sums of the nearest three inputs, so the receptive field in $\mathbf{H}_1$ has size three. b) The pre-activations of the four highlighted hidden units in layer $\mathbf{H}_2$ each take a weighted sum of the three channels in layer $\mathbf{H}_1$ at each of the three nearest positions. Each hidden unit in layer $\mathbf{H}_1$ weights the nearest three input positions. Hence, hidden units in $\mathbf{H}_2$ have a receptive field size of five. c) The hidden units in the third layer (kernel size three, stride two) increases the receptive field size to seven. d) By the time we add a fourth layer, the receptive field of the hidden units at position three have a receptive field that covers the entire input.

Visualisation (filters)



Visualisation (features)



Recap

- From simple manually designed edge detection kernels to deep neural networks
- From a few filters to a bank of filters
- From requirements for (almost) no-data and low compute to data-driven models and significant computational demands
- From easily interpretable models to much more opaque models
- Compare design principles!

Summary – classical CV vs deep learning

- Edge/contour detection – First-principle CV
  - Design principles (mathematical models)
  - Design choices (edge models, noise models, first, second order derivatives, etc.; parameter settings)
  - Advantages (compact, robust, stable, interpretable and quick to both train and evaluate; low on data, compute requirements)
  - Disadvantages (performance is lacking)
  - Conditions under which can be applied (clear understanding of the problem)

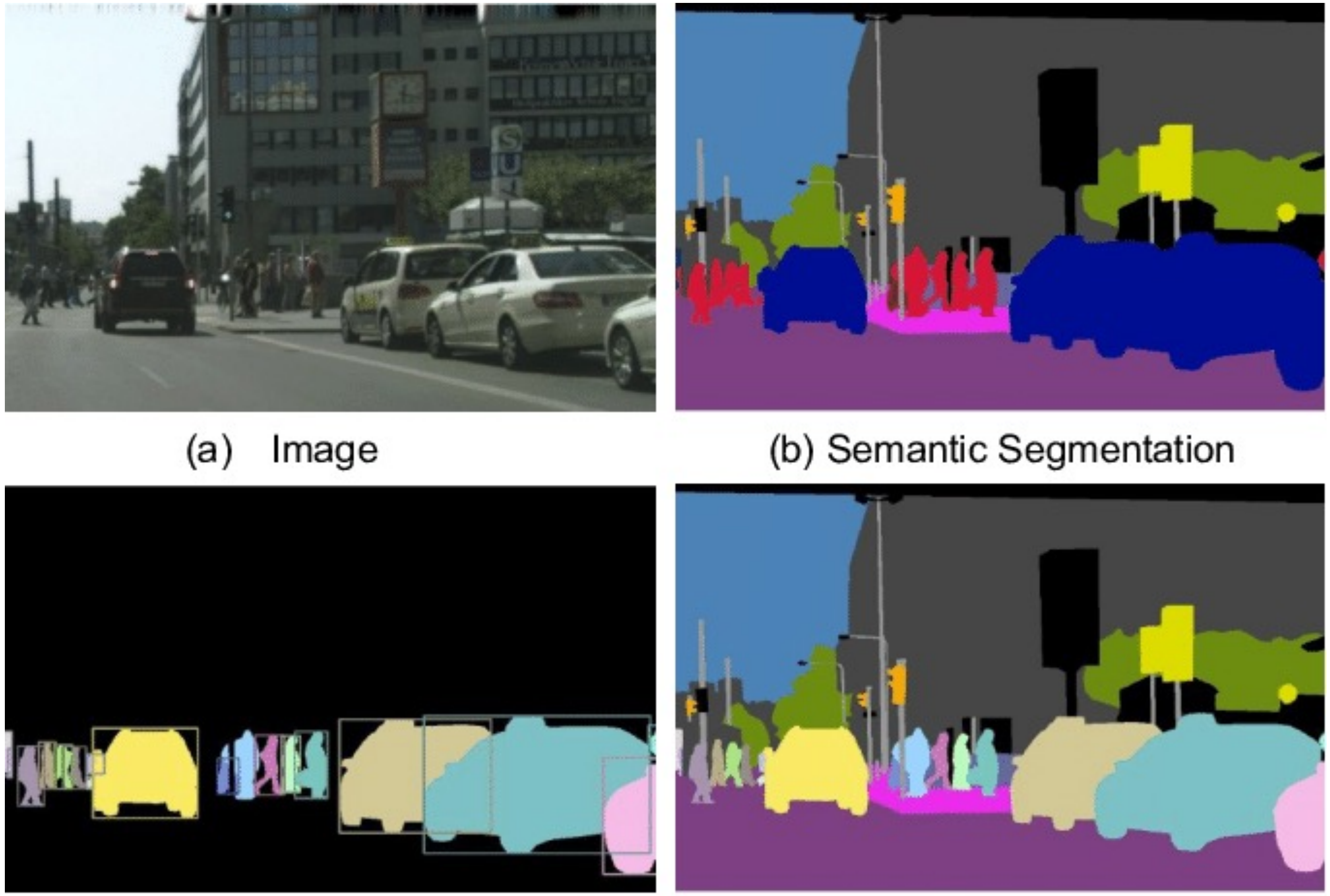
- Edge/contour detection – DL
  - Design principles (data, network model)
  - Design choices (network meta parameters: no of layers, filters, pooling, etc., loss functions)
  - Advantages (performance)
  - Disadvantages (not compact, not interpretable and high on data, compute requirements)
  - Conditions under which can be applied (data, compute, seek answers within the distribution of the training set)

Traditional Segmentation methods vs Segmentation with Deep Learning

- Before moving on to the Case study II on PCA and Autoencoders, let us quickly think about *segmentation problems*
- Segmentation in the classical CV
  - Very different approaches (thresholds, histograms, morphology, region growing, curve fitting)
- Deep (convolutional) networks (streamlined methodology)
  - Fully convolutional network (FCN)
  - U-Net (un-pooling; upsampling)

Segmentation

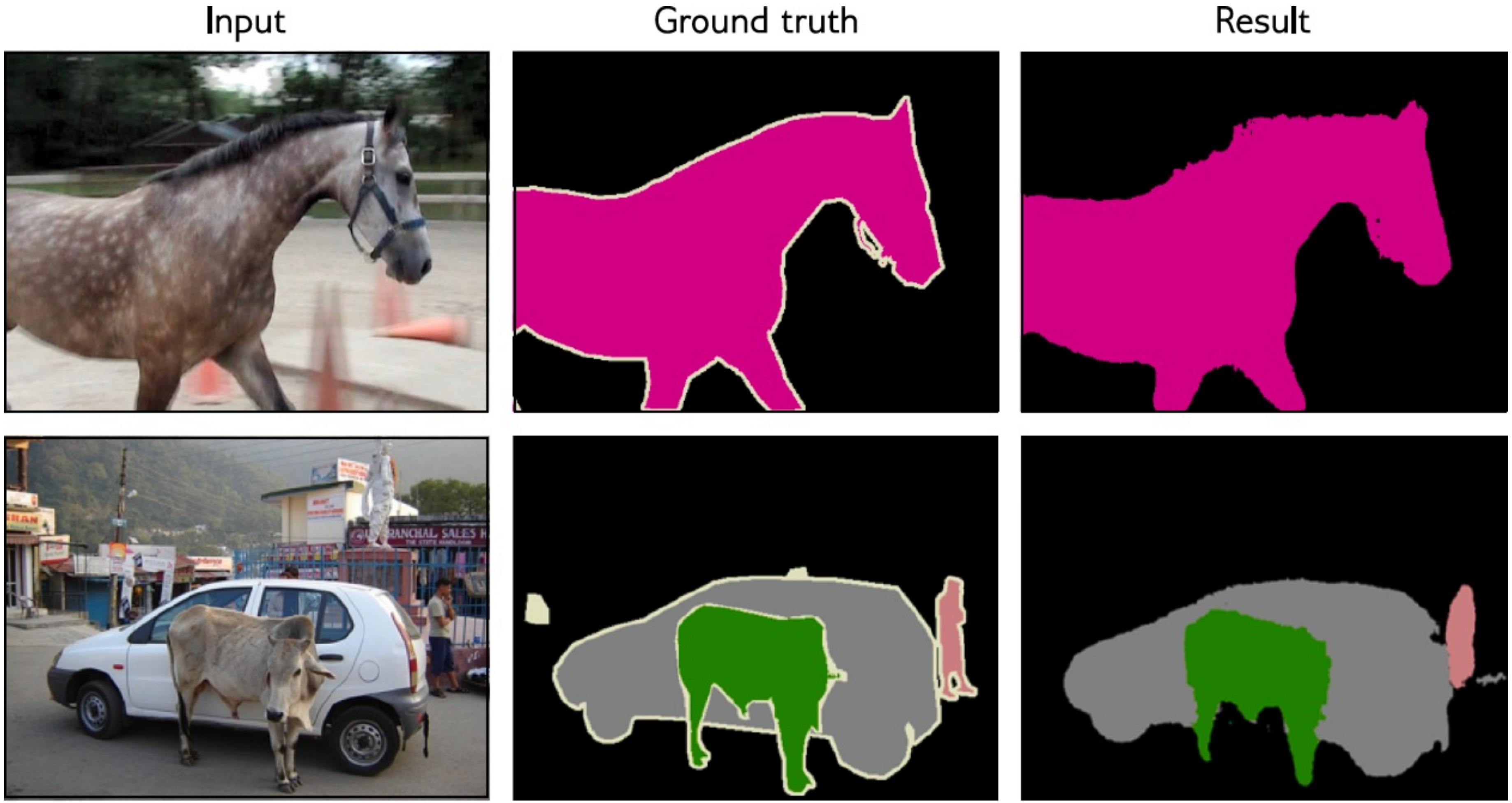
- Semantic segmentation
- Instance segmentation
- Panoptic segmentation



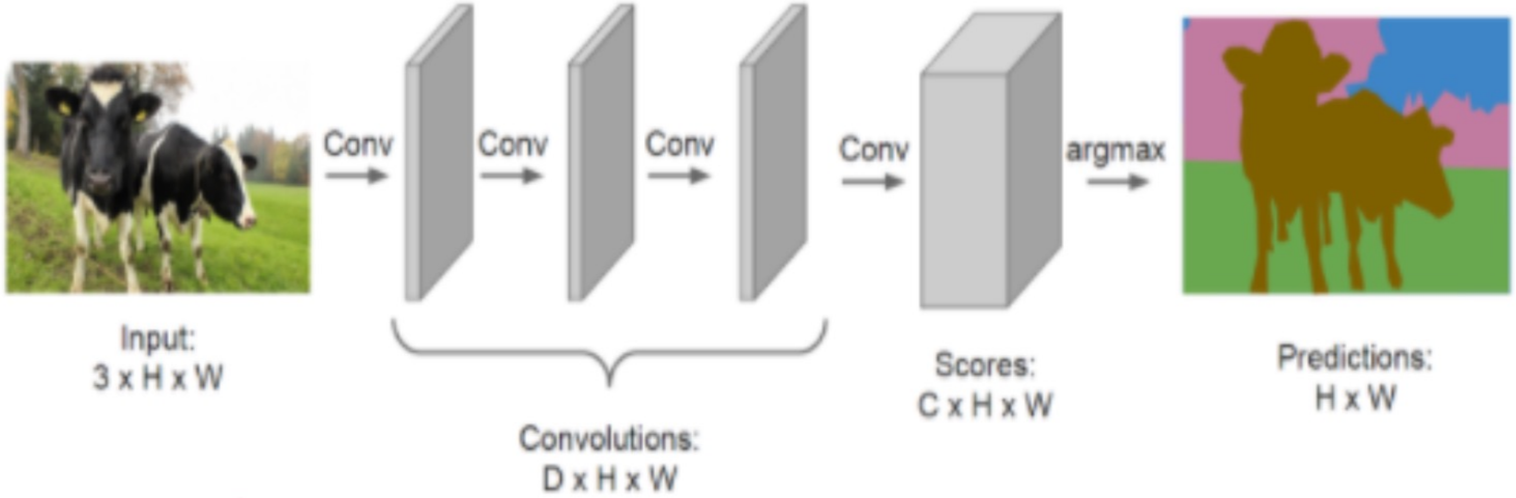
(a) Image (b) Semantic Segmentation

(c) Instance Segmentation (d) Panoptic Segmentation

Semantic segmentation results



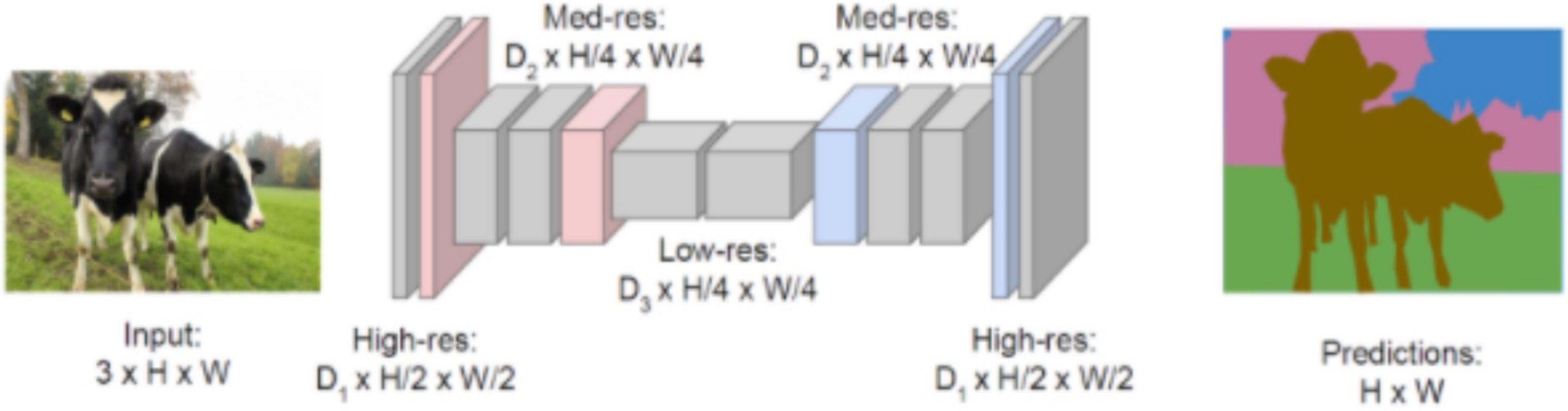
Semantic Segmentation



Sample

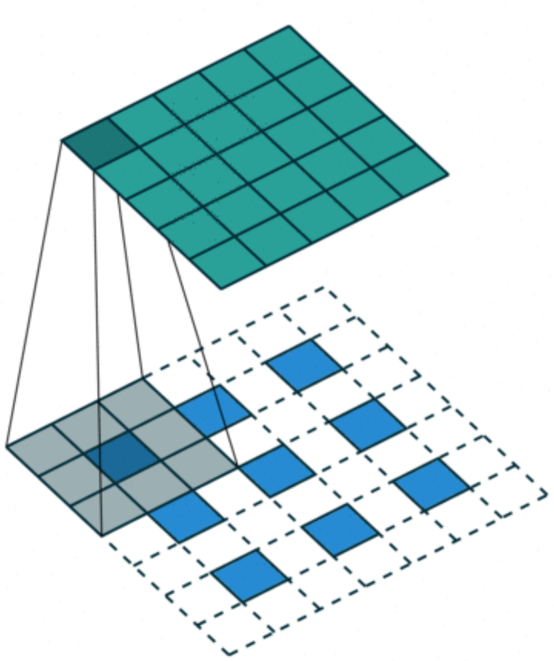
Problem: Preservation of full-resolution becomes quite computationally costly
Solution: FCN with downsampling and upsampling

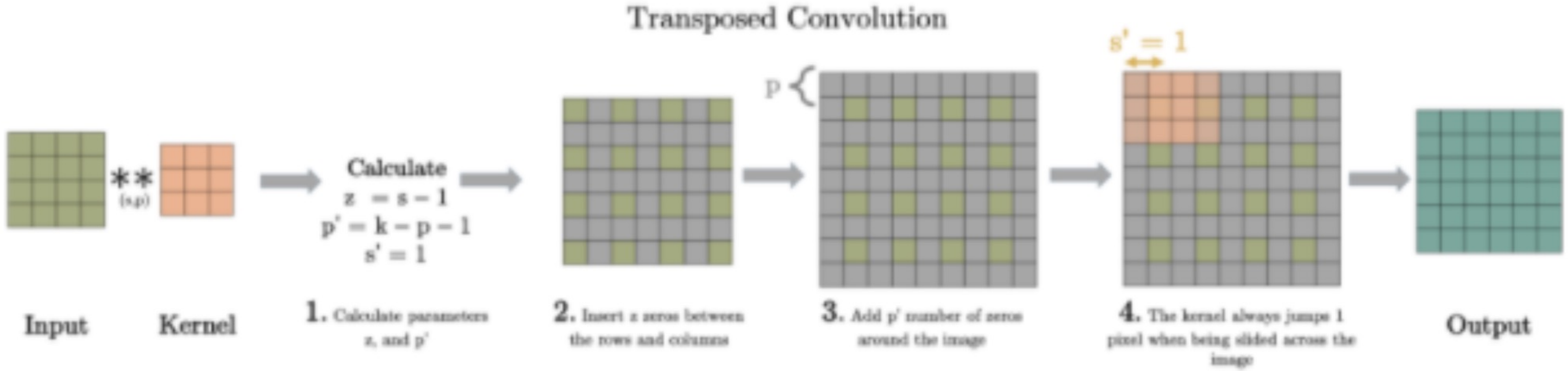
Semantic Segmentation



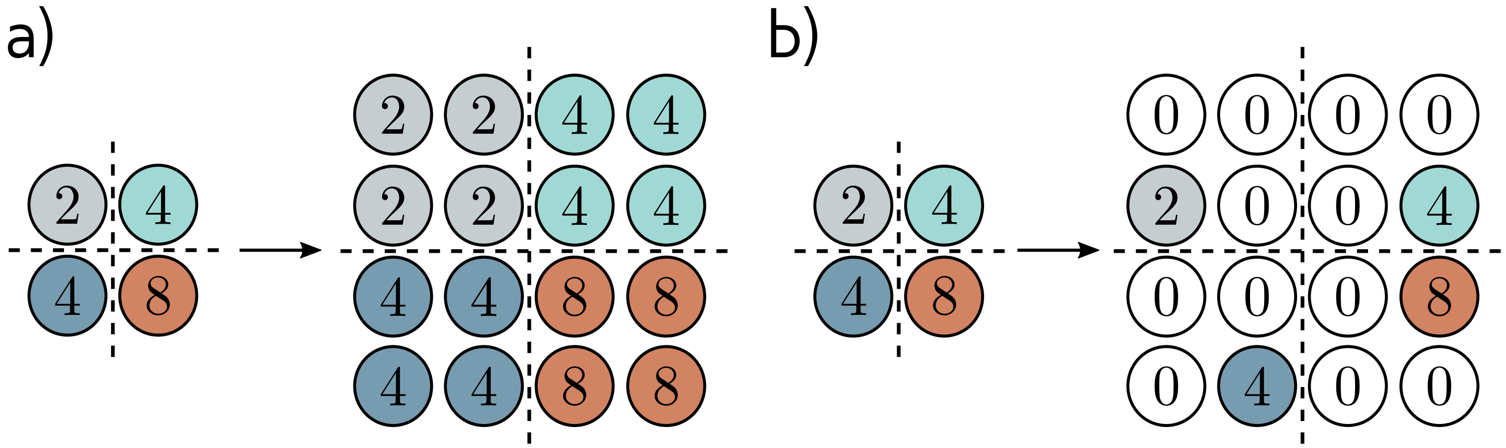
Solution: FCN with downsampling and upsampling

Transpose convolution





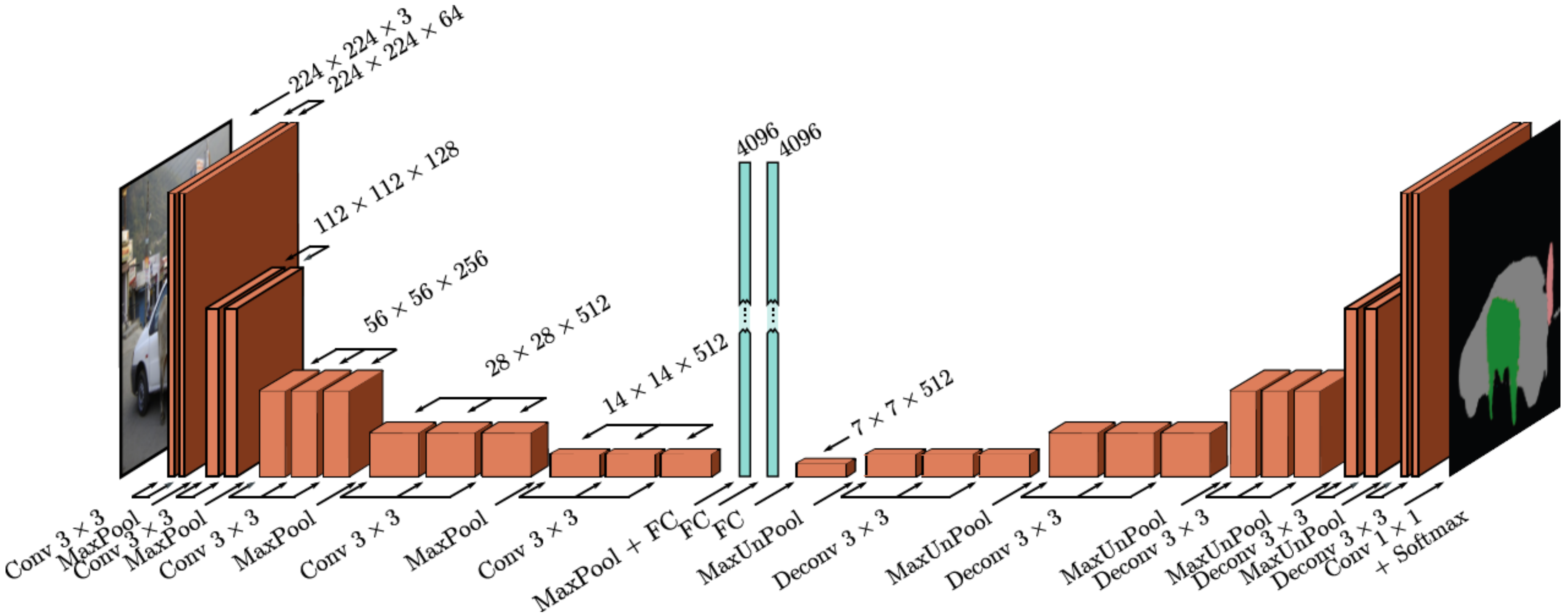
Upsampling/unpooling



Duplicate

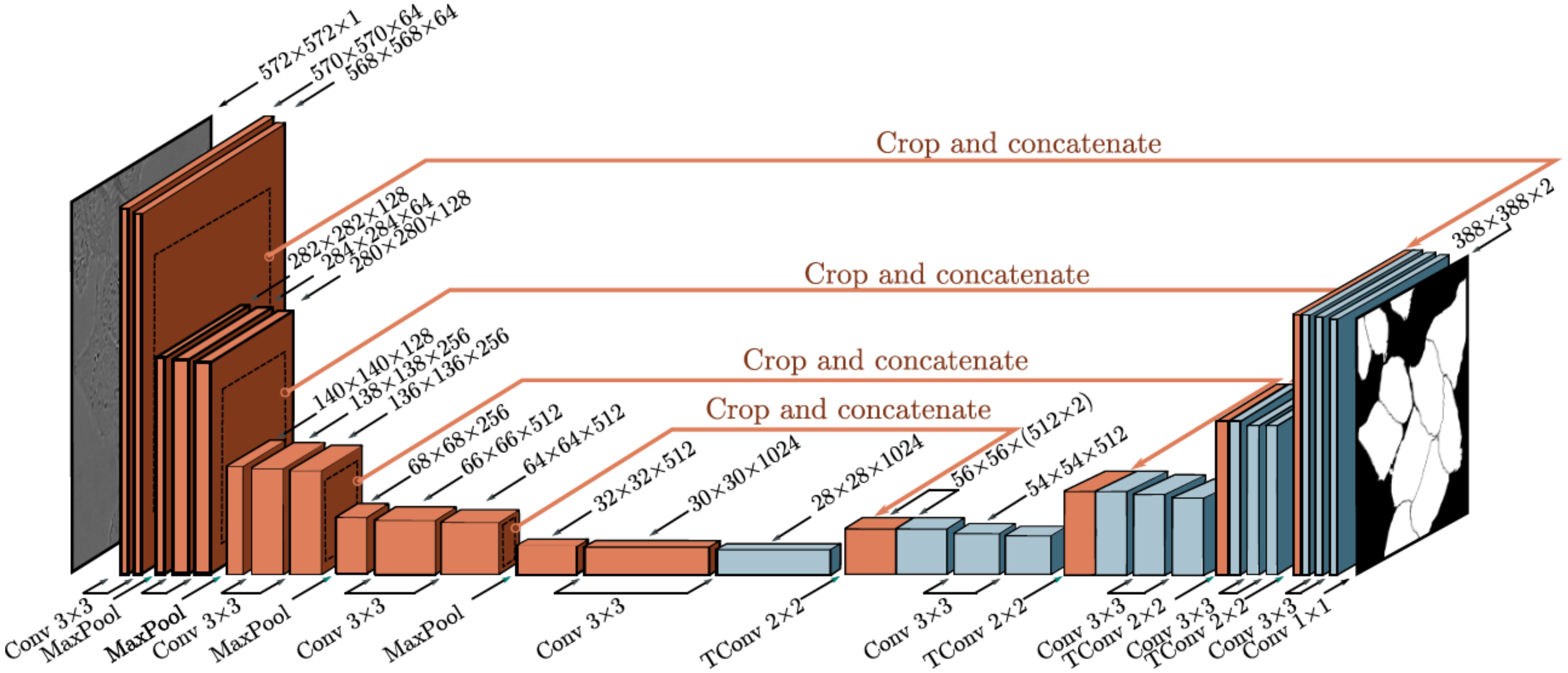
Max-upsampling

Semantic Segmentation (2015)



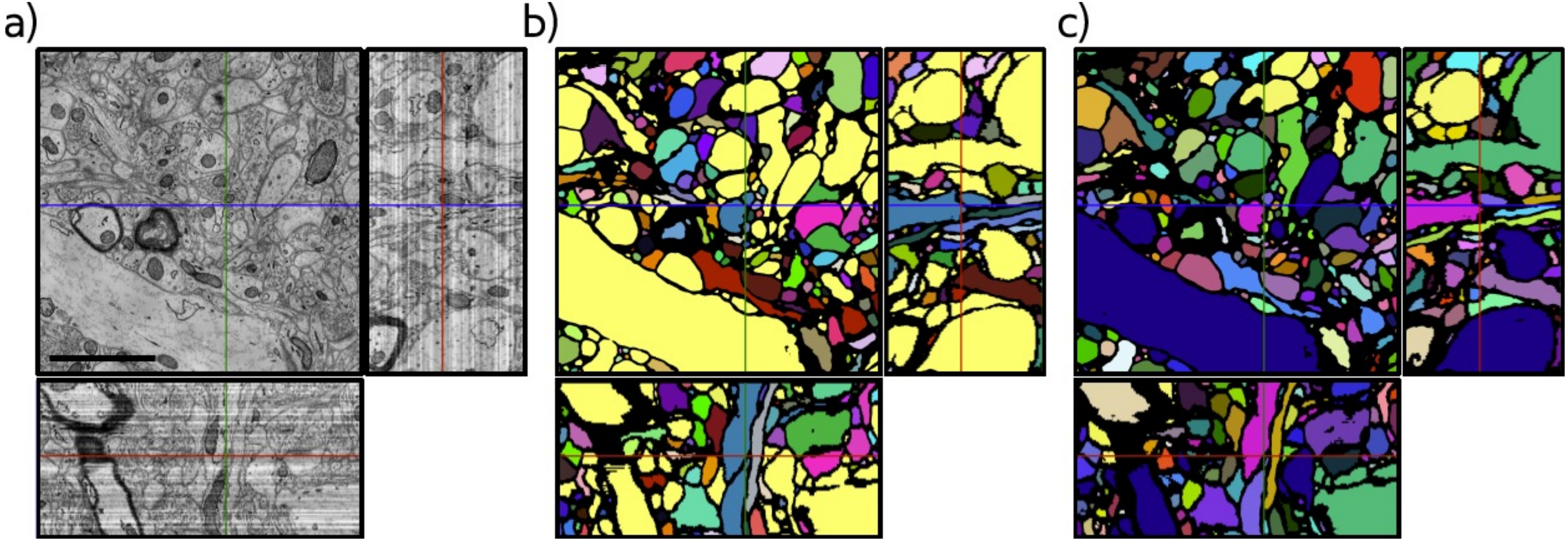
Encoder Decoder

U-Net (2016)



572×572×1
570×570×64
568×568×64
282×282×128
284×284×64
280×280×128
140×140×128
138×138×256
136×136×256
68×68×256
66×66×512
64×64×512
32×32×512
30×30×1024
28×28×1024
56×56×(512×2)
54×54×512
388×388×2
Crop and concatenate
Crop and concatenate
Crop and concatenate
Crop and concatenate
Conv 3×3
MaxPool
MaxPool
Conv 3×3
MaxPool
Conv 3×3
MaxPool
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
TConv 2×2
Conv 3×3
Conv 1×1

U-Net Results



What have we learned today

- We built an example of a convolutional neural network (shape of activations, no. of activations, no. of parameters)
- We looked at the main properties of the convolutional neural networks and what makes them distinctive (shareability, sparsity, receptive filed, visualisation)
- We briefly revisited the classical computer vision models for segmentation and showed their deep learning extensions (FCN, U-Net)

What is coming next?

- We will revisit PCA method and show how it can be related to deep learning model of autoencoders
- We will summarise the overall insights and contrast the two methodological approaches (First-principle CV vs DL CV)