
# COSE474 Deep Learning

# Lecture 10: Encoder-Decoder Architecture

Seungryong Kim

Computer Vision Lab. (CVLAB)

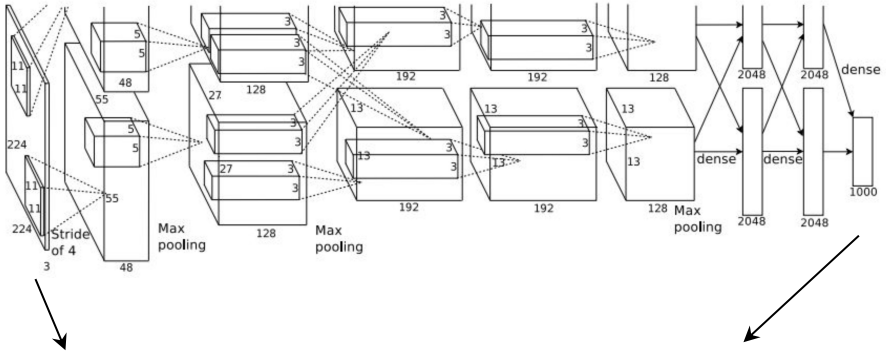
Department of Computer Science and Engineering

Korea University

# Image-level vs. Pixel-level Prediction

## Image-level prediction

Ex) Image classification, etc.

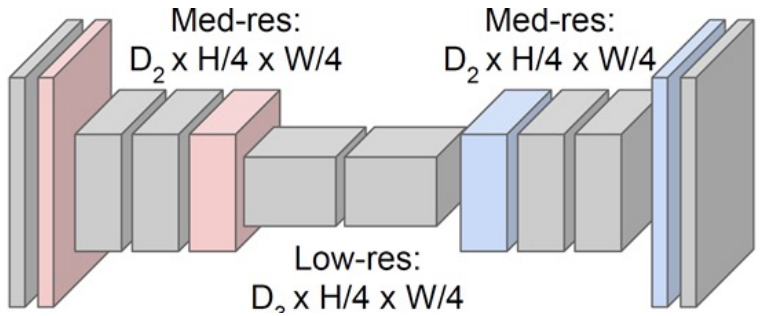



**Input**
Color image
$224 \times 224 \times 3$

**Image classification output**
$1000 \times 1$ probability vector that indicates what the input image contains in a probabilistic fashion

## Pixel-level prediction

Ex) Segmentation, depth estimation, etc.

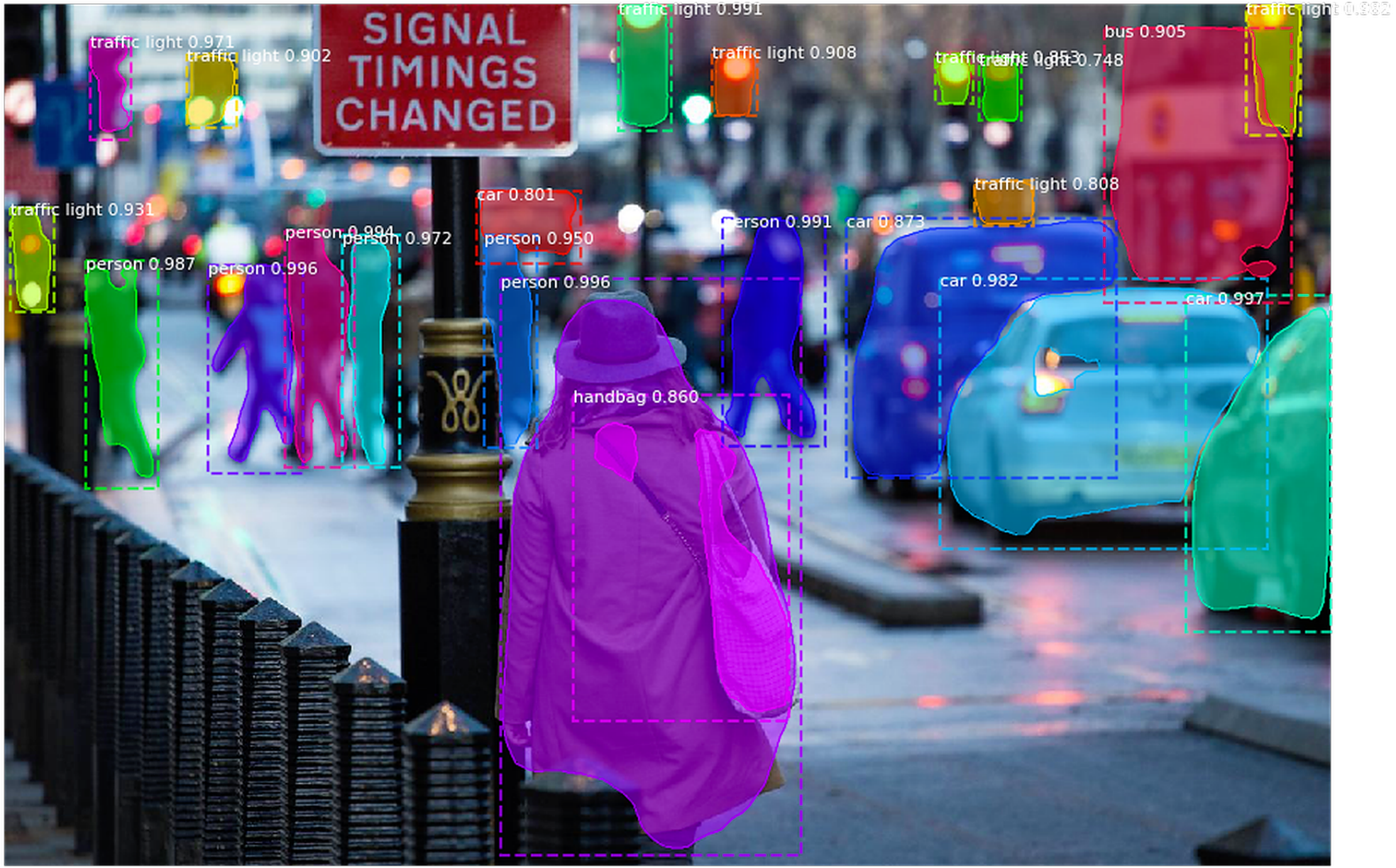



**Input**
Color image
$224 \times 224 \times 3$

**Segmentation output**
$224 \times 224 \times 1000$ probability volume that indicates the probability vector for each pixel

**Depth output**
$224 \times 224 \times 1$ depth image

# Applications: Segmentation



# Applications: Depth Estimation

Ranftl et al, “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer”, TPAMI 2020

# Applications: Super-Resolution

bicubic
(21.59dB/0.6423)

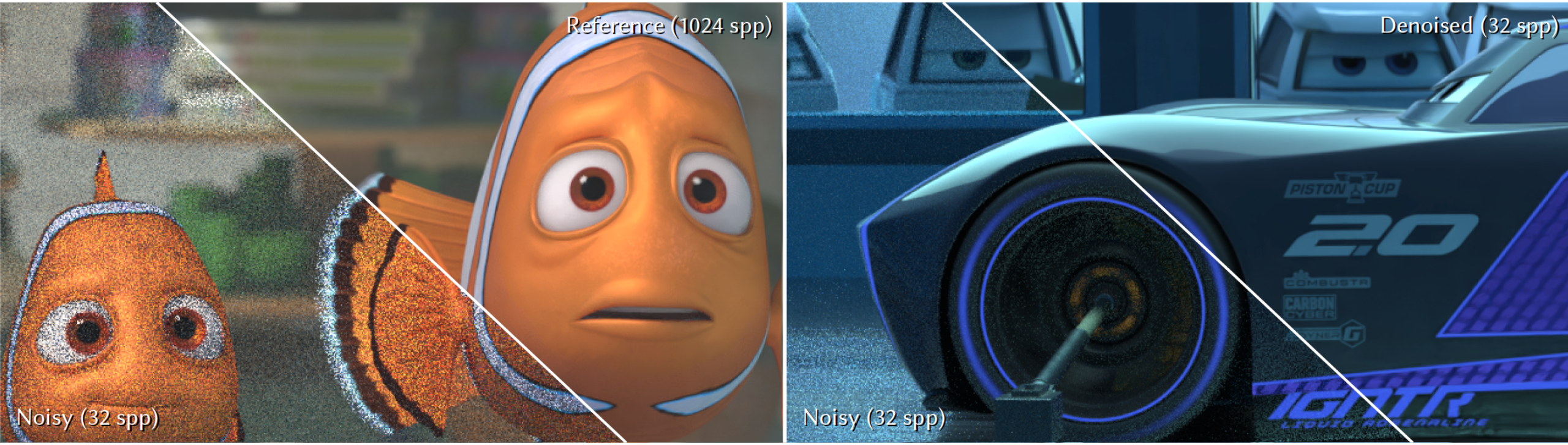
SRResNet
(23.53dB/0.7832)

SRGAN
(21.15dB/0.6868)

original

Ledig et al, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, CVPR 2017

# Applications: Image Denoising



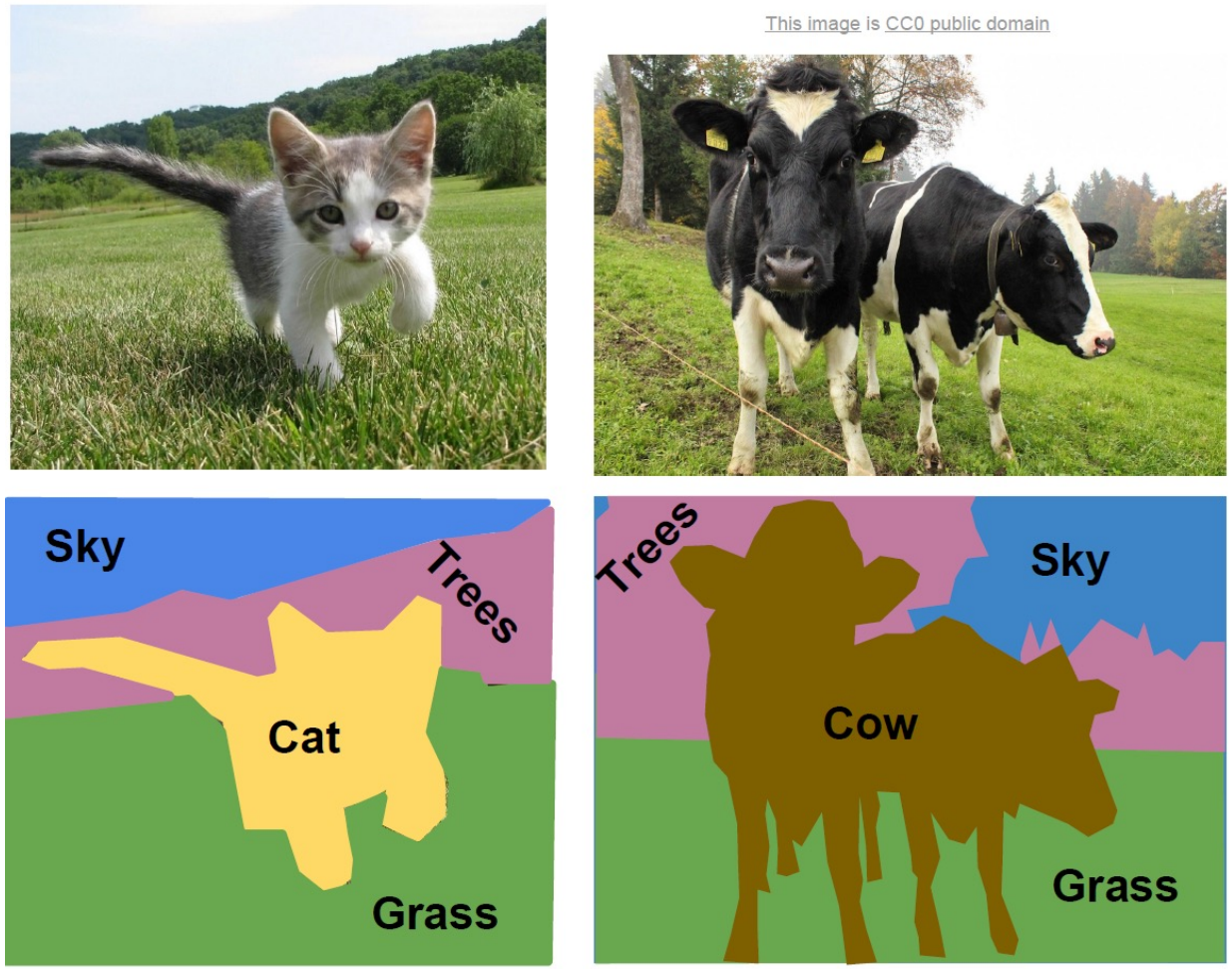

TRAINING TEST

Ledig et al, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, CVPR 2017

# Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels




No objects, just pixels

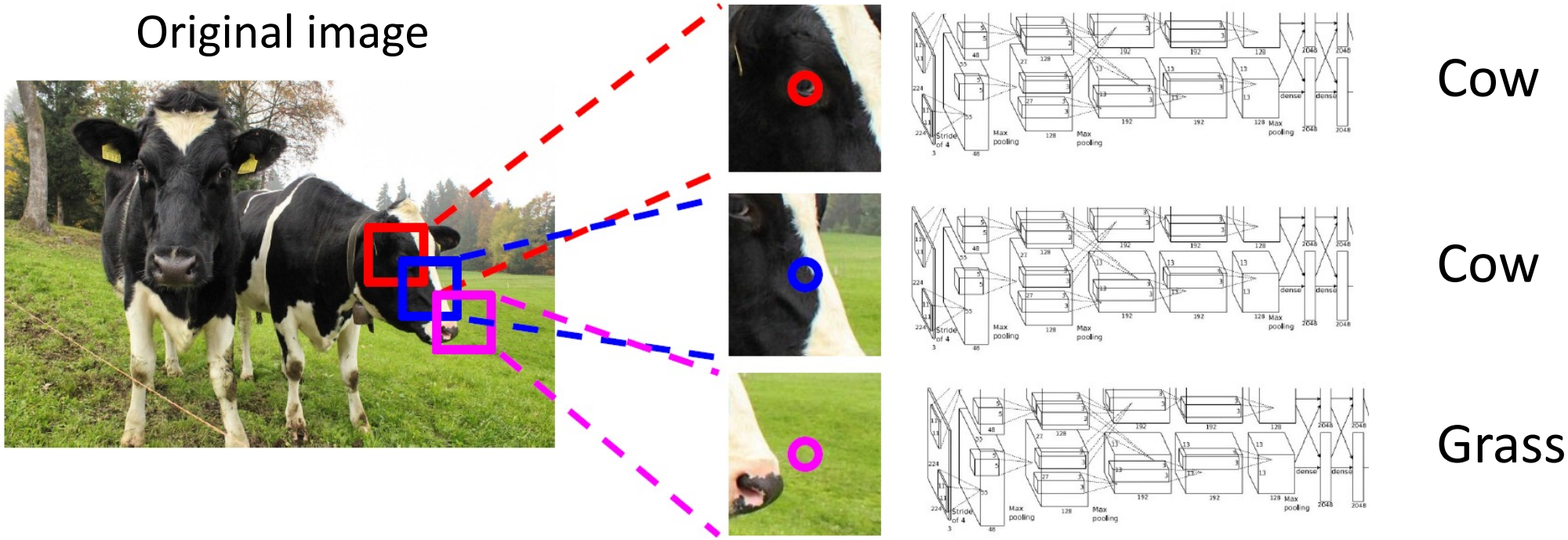
# Early Idea of Semantic Segmentation

## Sliding Window

Extract patch

Classify center pixel with CNN

Original image




Problem:

Very inefficient! Not reusing shared features between overlapping patches

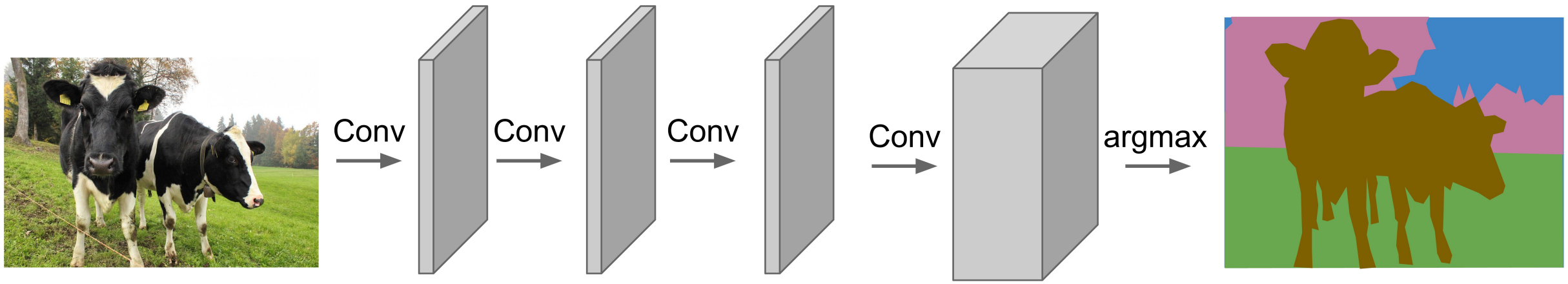
Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation

## Fully Convolutional Networks

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!




Input:
3 x H x W

Convolutional activations:
D x H x W

Scores:
C x H x W

Predictions:
H x W

Problem:
Convolutions at original image
resolution will be very expensive ...

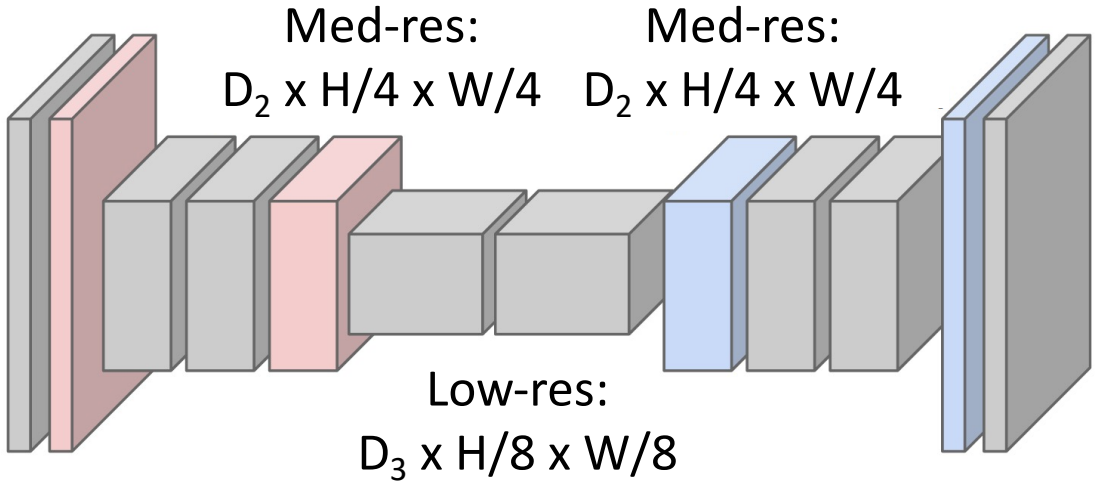
# Semantic Segmentation

## Fully Convolutional Networks

**Encoder-Decoder Network**

Design network as a bunch of convolutional layers, with *down-sampling* and *up-sampling* inside the network!

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

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

# Semantic Segmentation

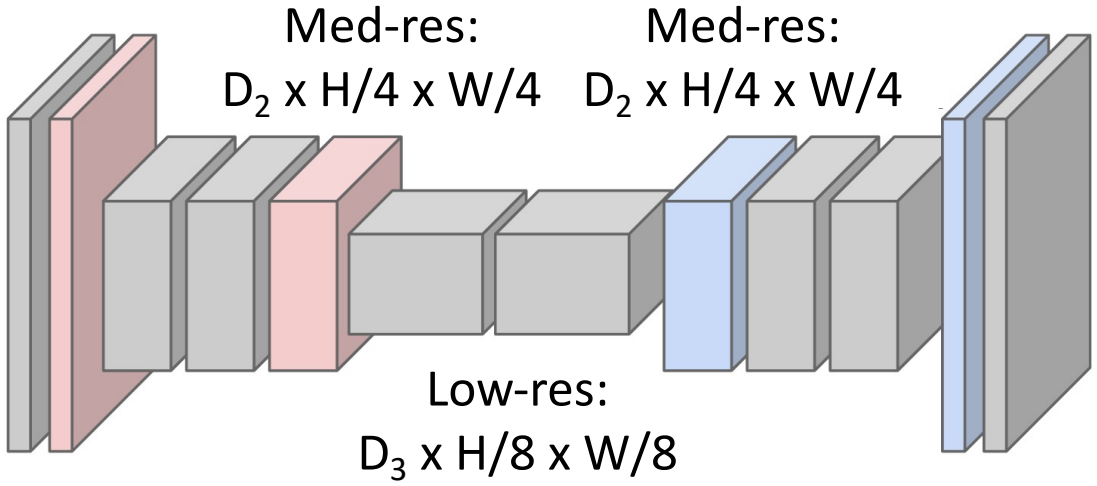
## Fully Convolutional Networks

**Encoder-Decoder Network**

Design network as a bunch of convolutional layers, with *down-sampling* and *up-sampling* inside the network!

**Down-sampling:**
Pooling, strided convolution

**Up-sampling:**
???




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

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

# Up-sampling of Activation Maps

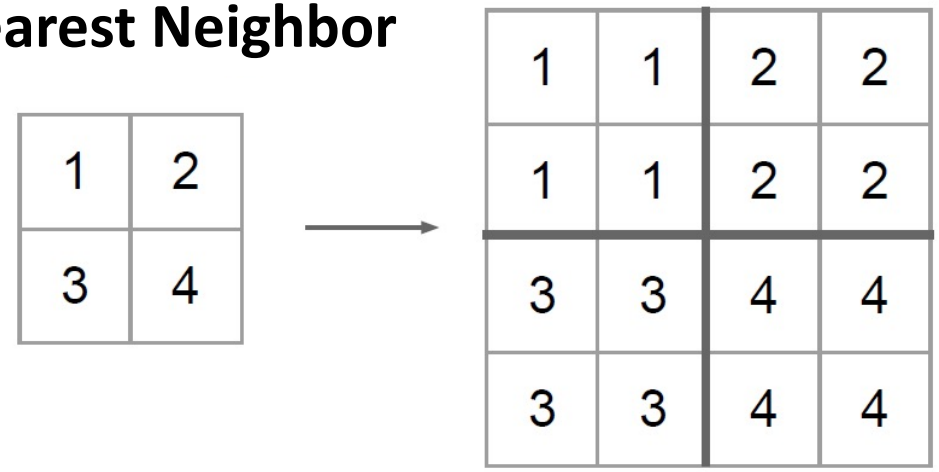
**Up-sampling Methods**

- Nearest neighbor interpolation
- Bilinear interpolation
- Bed of nails
- Max Unpooling
- Transpose convolution (a.k.a. fractionally strided convolution)

# Semantic Segmentation

## In-Network Upsampling: “Unpooling”

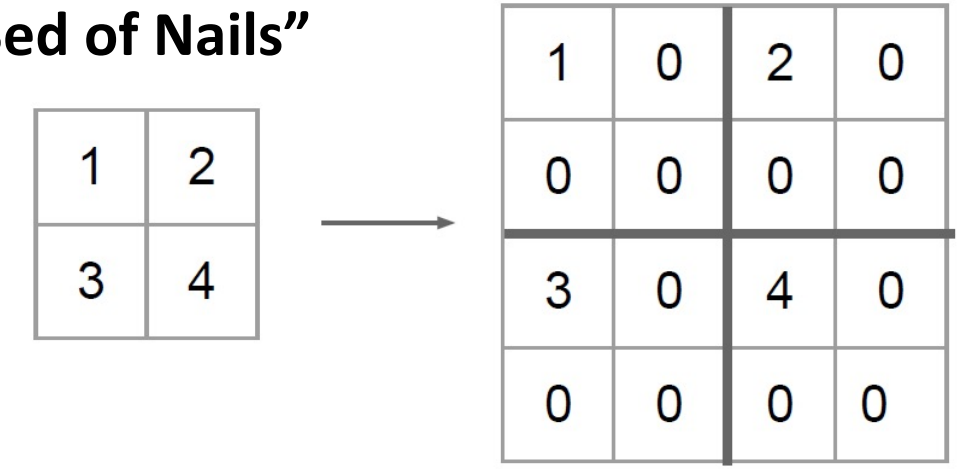
**Nearest Neighbor**




Input: 2 x 2

Output: 4 x 4

**“Bed of Nails”**




Input: 2 x 2

Output: 4 x 4

# Semantic Segmentation

## 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

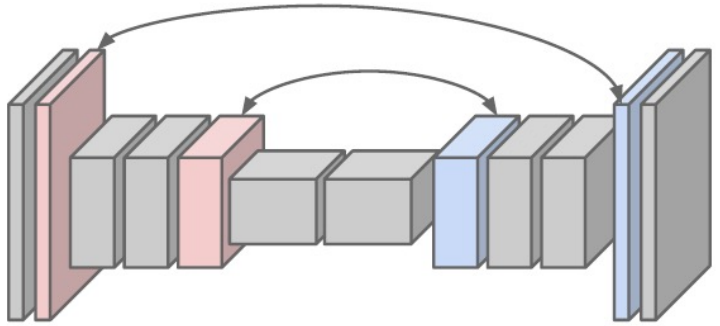
| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 0 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 |

Output: 4 x 4

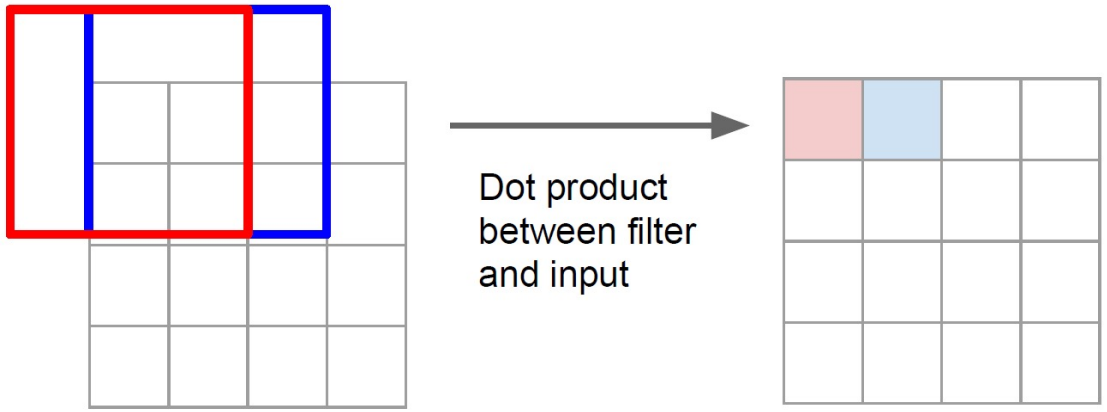
Corresponding pairs of downsampling and upsampling layers



# Semantic Segmentation

## Learnable Upsampling: Transpose Convolution

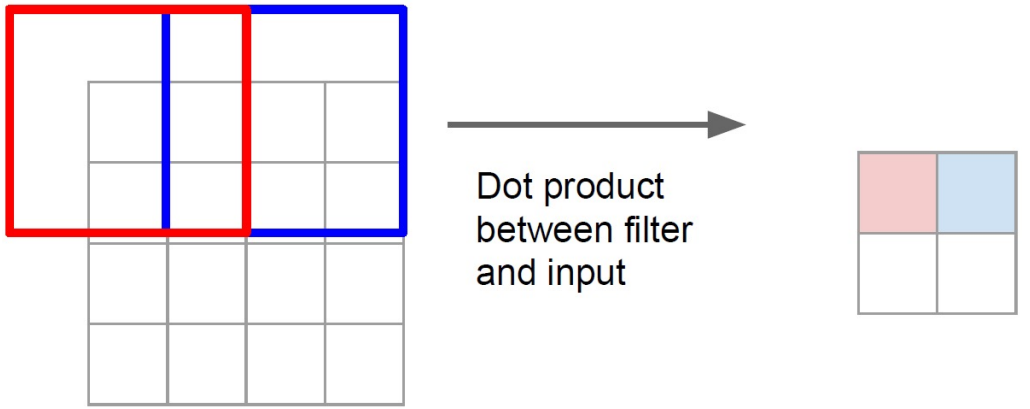
**Recall:** Normal 3 x 3 convolution, stride 1, pad 1




Input: 4 x 4

Output: 4 x 4

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




Input: 4 x 4

Output: 2 x 2

Output size $W_2$

$W_2 = (W_1 - F + 2P)/S + 1$

where $W_1$: input size

F: filter size

P: zero padding

S: stride

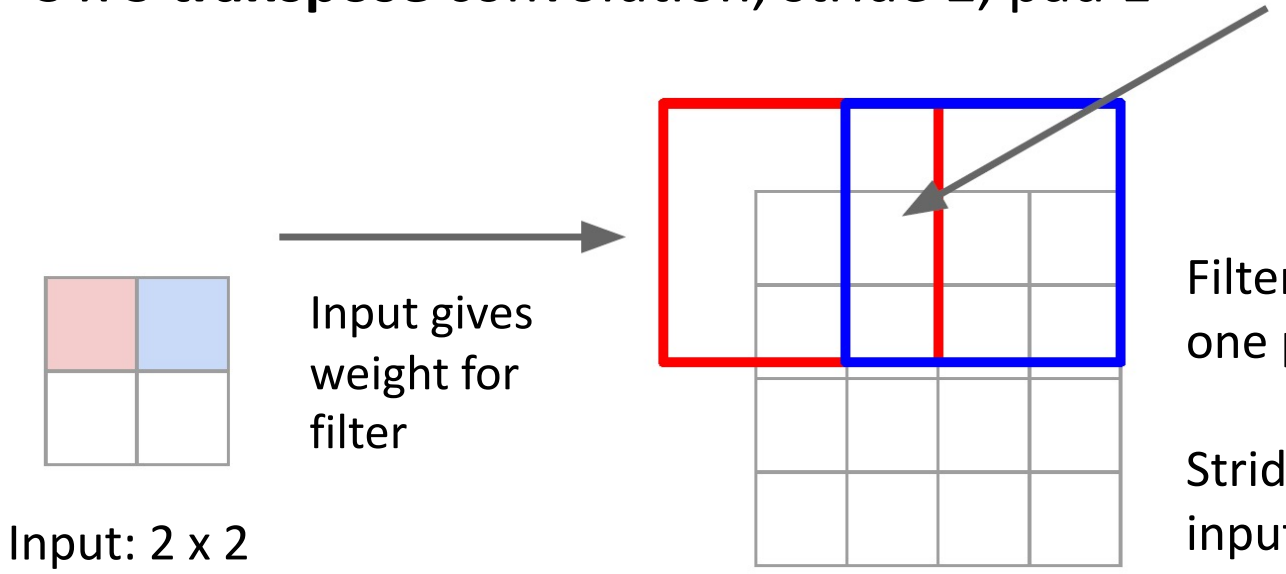
Filter moves 2 pixels in the input for every one pixel in the output.

Stride gives ratio between movement in input and output.

# Semantic Segmentation

## Learnable Upsampling: Transpose Convolution

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




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

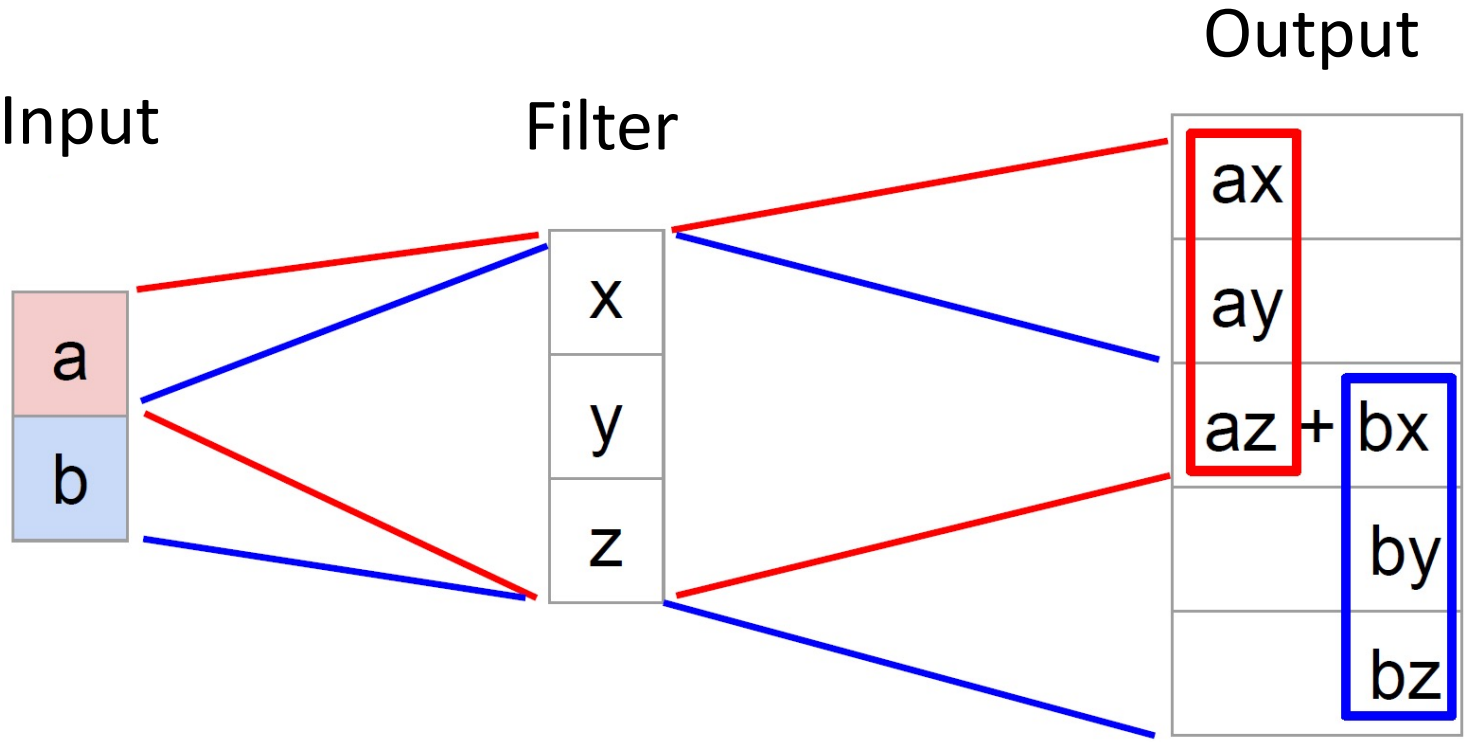
Stride gives ratio between movement in input and output.

Other names:
- Deconvolution
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution

# Semantic Segmentation

## Transpose Convolution: 1D Example

- Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.
- Need to crop one pixel from output to make output exactly 2x input.



# Convolution as Matrix Multiplication (1D)

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

Convolution as a matrix multiplication

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

# Convolution as Matrix Multiplication (1D)

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

Convolution as a matrix multiplication

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

When stride>1, convolution transpose is no longer a normal convolution!

# Semantic Segmentation

## Fully Convolutional Networks

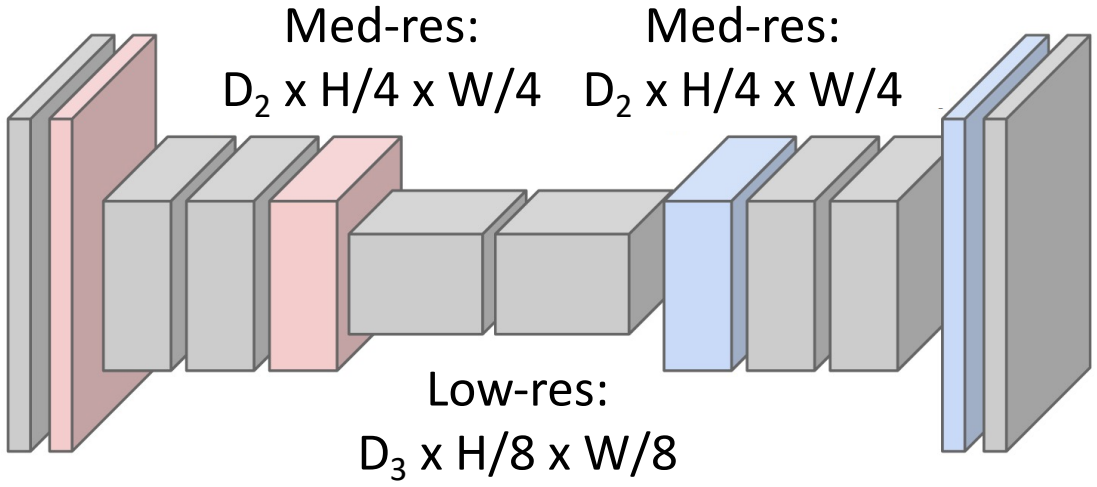
**Encoder-Decoder Network**

Design network as a bunch of convolutional layers, with *down-sampling* and *up-sampling* inside the network!

**Down-sampling:**
Pooling, strided convolution

**Up-sampling:**
Unpooling, transpose convolution

Input:
3 x H x W



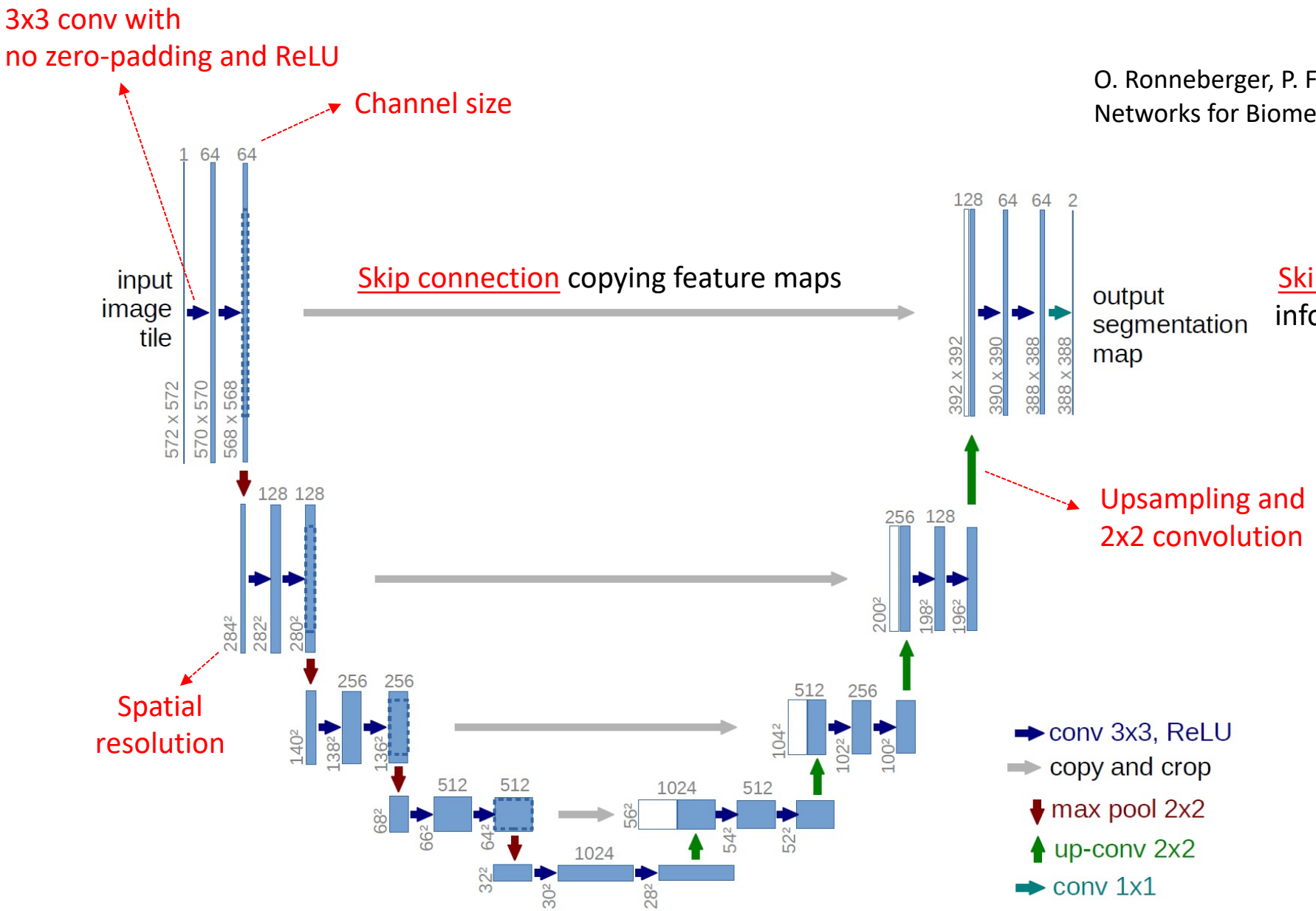

High-res:
$D_1$ x H/2 x W/2

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

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

# U-Net Architecture



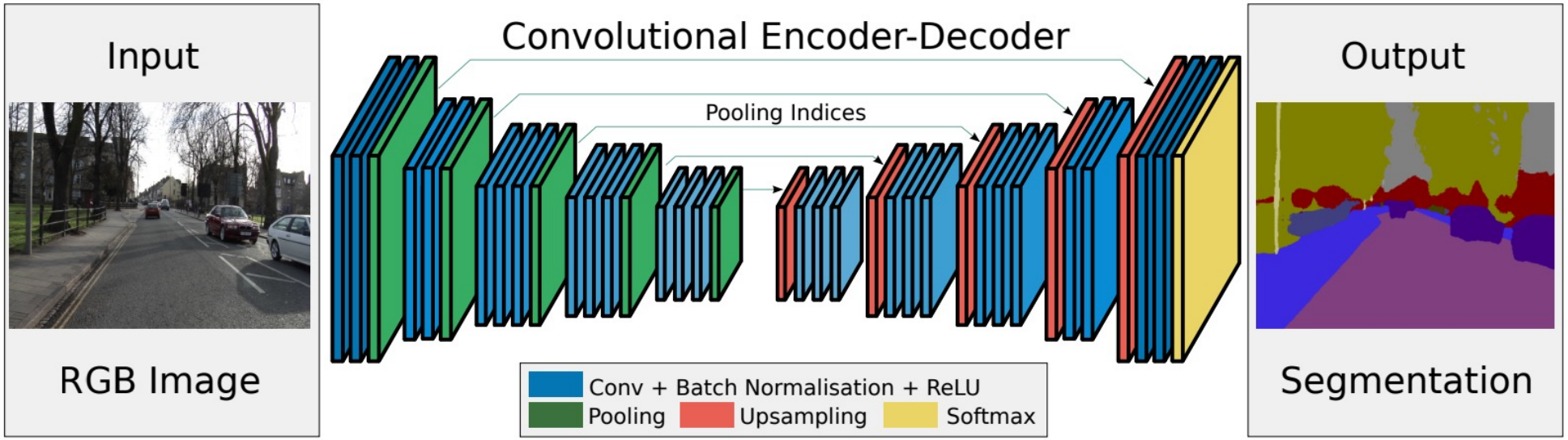

O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI 2015

Skip connection is useful in that multi-scale information is considered at the same time.

Note) This is just one of examples implementing encoder-decoder architectures. There are lots of variants depending on conv parameters and down/upsampling.

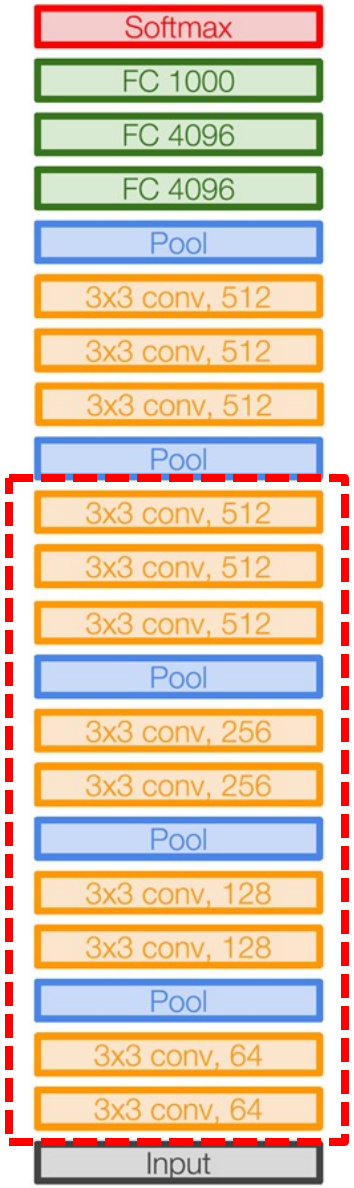
# SegNet Architecture

V. Badrinarayanan, A. Kendall, R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” IEEE Trans. on PAMI, 2017

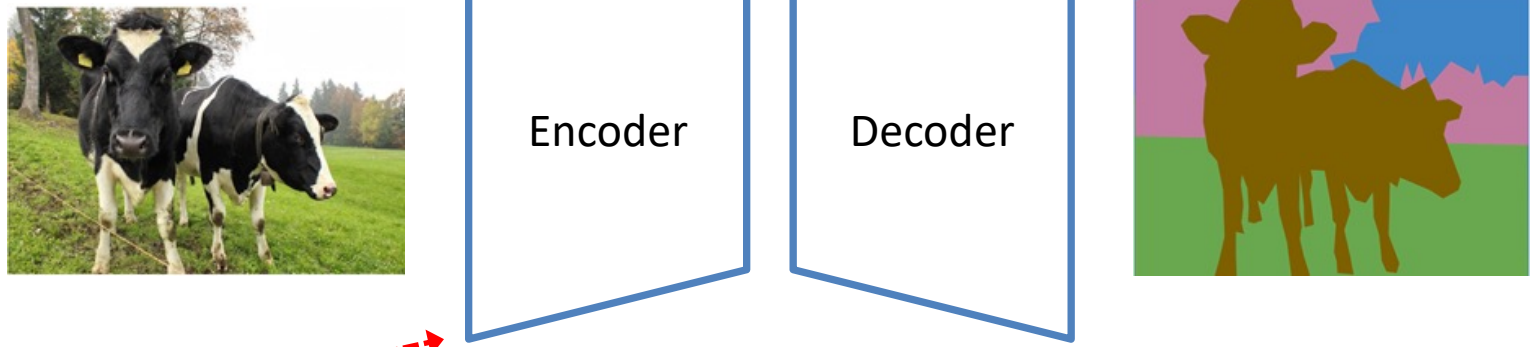



- Batch Norm is used
- Max unpooling is used for upsampling

# Variants Using Baseline CNN Architectures




1. Use the part of VGG 16 or ResNet-50/101/152 in the encoder
2. Initialize the encoder using the pre-trained parameters with ImageNet
3. Design the decoder in a symmetric manner
4. Initialize the decoder randomly (from scratch)




Use this part as the encoder

# Conclusion

- Encoder-decoder architecture is used for pixel-level prediction such as segmentation and image denoising.

- Various up-sampling methods
  - Nearest neighbor/bilinear interpolation, bed of nails, max unpooling
  - Transpose convolution (a.k.a. fractionally strided convolution)

# Thank you!
# Q & A