

Computer Vision

Naeemullah Khan

naeemullah.khan@kaust.edu.sa



جامعة الملك عبد الله
للغعلوم والتكنولوجيا
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

January 28, 2026

Table of Contents

1. Introduction
2. Image Segmentation
3. Adapting CNNs to Segmentation Tasks
4. Upsampling Operations
5. Residual Connections and U-Net
6. Instance and Panoptic Segmentation

Learning Outcomes

- ▶ Understand the fundamentals of image segmentation and its importance.
- ▶ Understand how Convolutional Neural Networks (CNNs) are adapted for segmentation tasks.
- ▶ Understand different upsampling techniques used in segmentation models.
- ▶ Understand the role of residual connections and the U-Net architecture in segmentation.
- ▶ Differentiate between instance segmentation and panoptic segmentation.

Image Classification

- ▶ Previously, we discussed Image Classification
- ▶ A core task in Computer Vision



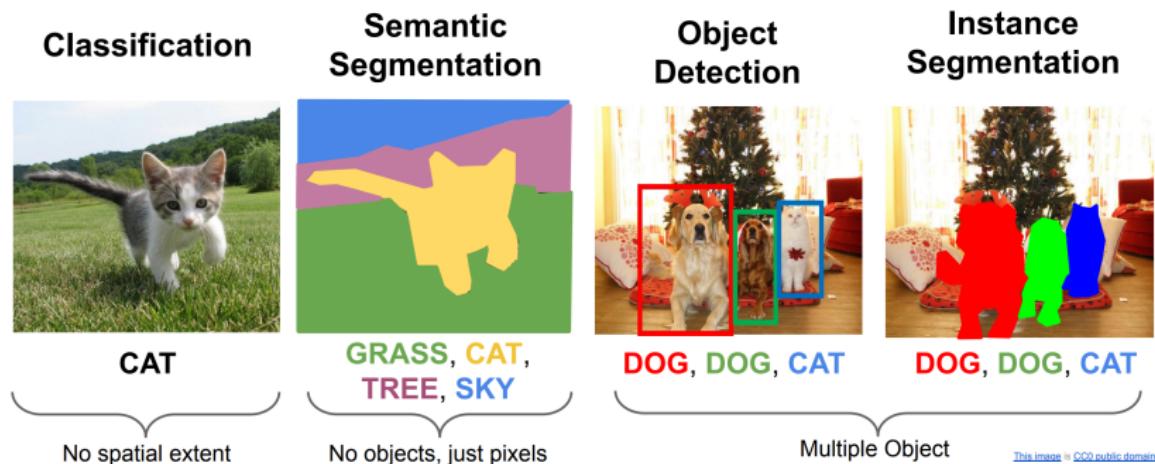
This image by [Nikita](#) is
licensed under [CC-BY 2.0](#).

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}



cat

Computer Vision Tasks



Semantic Segmentation



GRASS, CAT,
TREE, SKY, ...

Paired training data: for each training image,
each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Semantic Segmentation

Full image



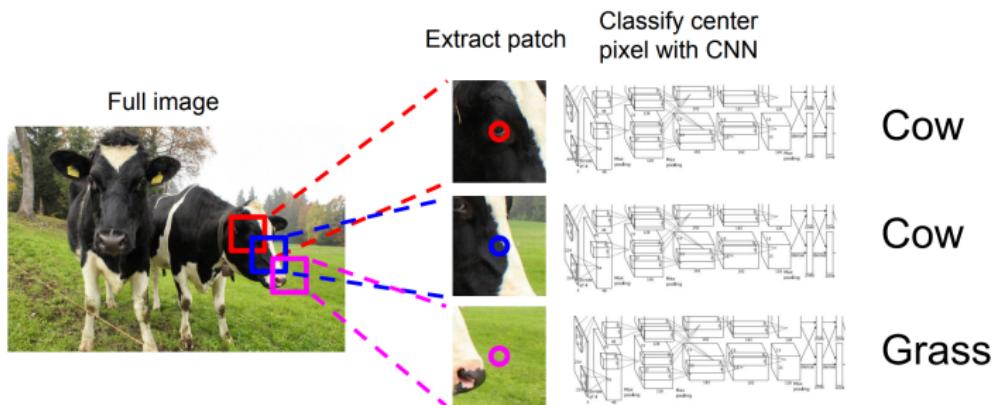
Semantic Segmentation

Full image



- ▶ Impossible to classify without context
- ▶ How do we include context?

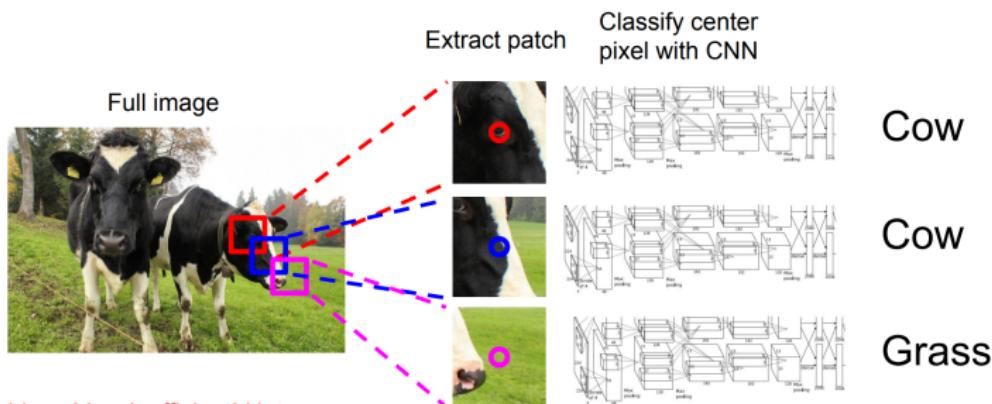
Semantic Segmentation Idea: Sliding Window



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 Idea: Sliding Window (cont.)



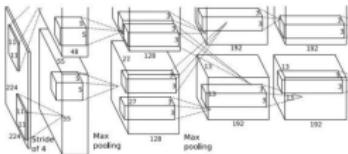
Problem: Very inefficient! Not reusing shared features between overlapping patches

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 Idea: Convolution

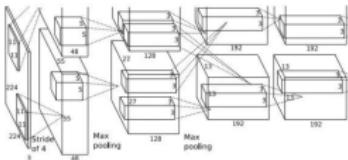
Full image



An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Semantic Segmentation Idea: Convolution (cont.)

Full image

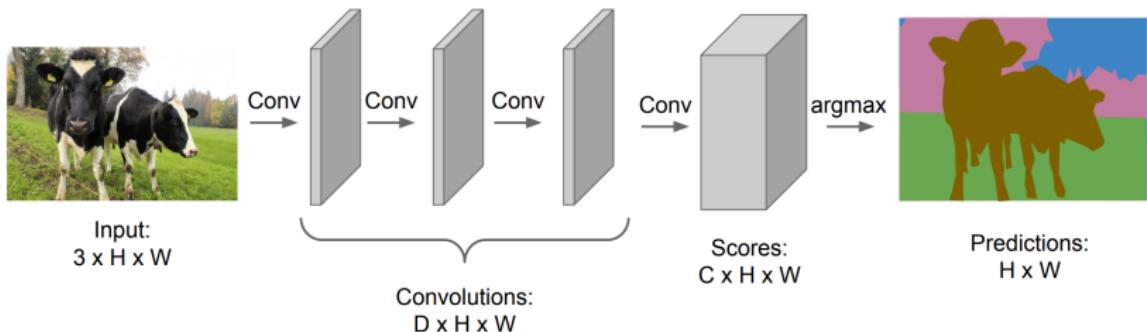


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

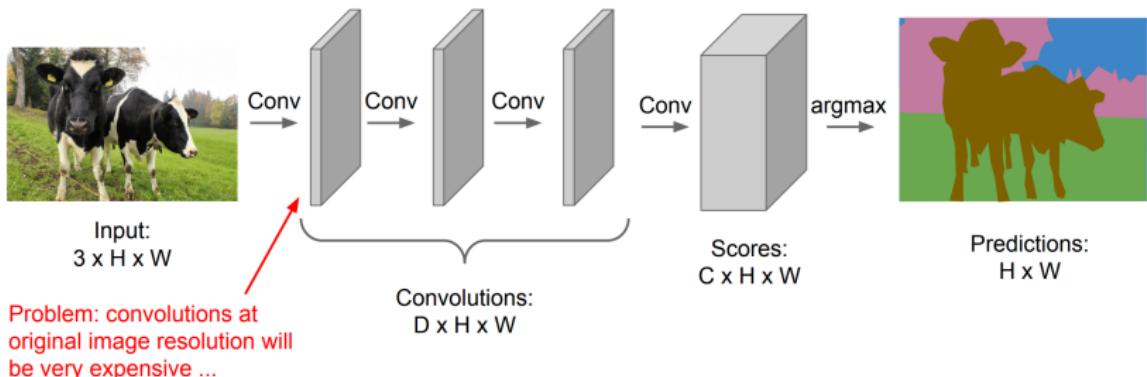
Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



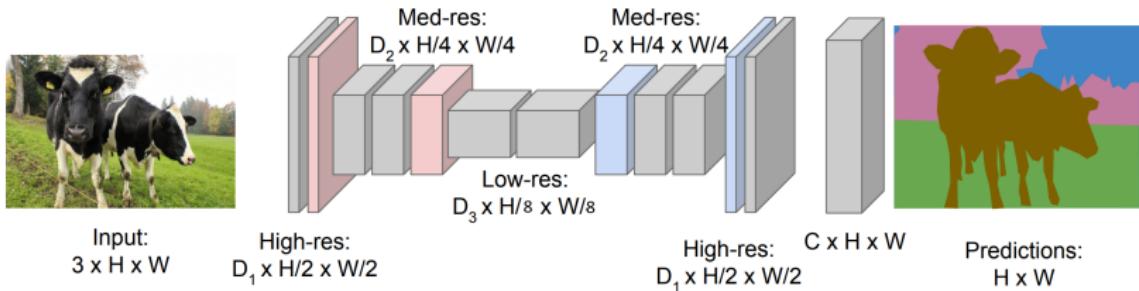
Semantic Segmentation Idea: Fully Convolutional (cont.)

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional (cont.)

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



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

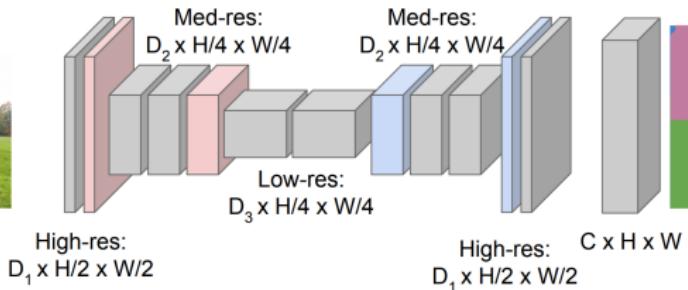
Semantic Segmentation Idea: Fully Convolutional (cont.)

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
???



Predictions:
 $H \times W$

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

In-Network Upsampling: Unpooling

Nearest Neighbor

1	2
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

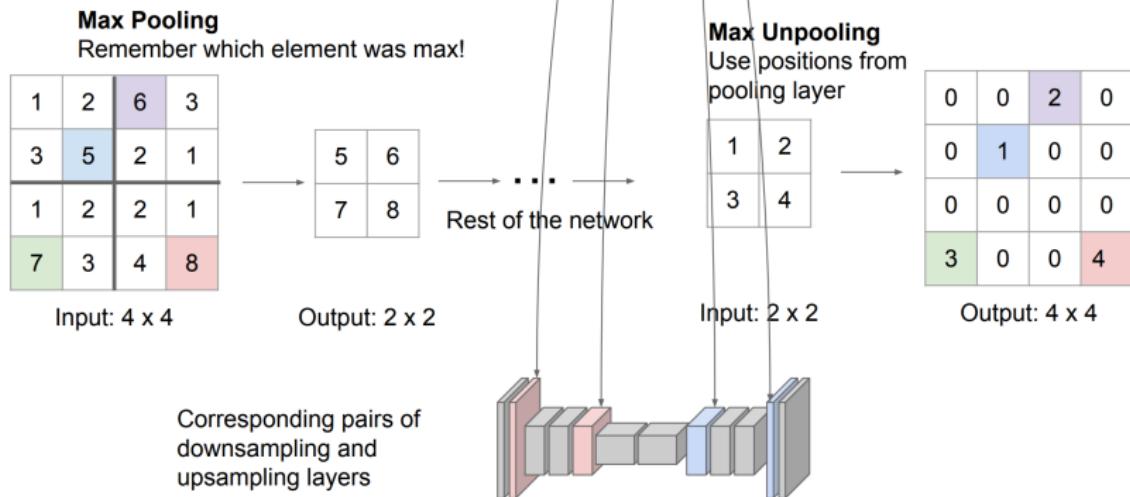


1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

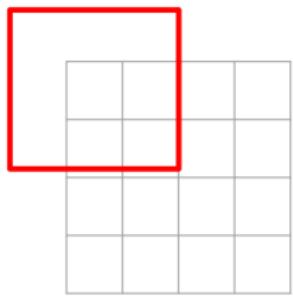
Output: 4 x 4

In-Network Upsampling: Max Unpooling



Learnable Upsampling: Transposed Convolution

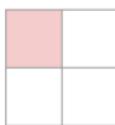
Recall: Normal 3×3 convolution, stride 2 pad 1



Input: 4×4



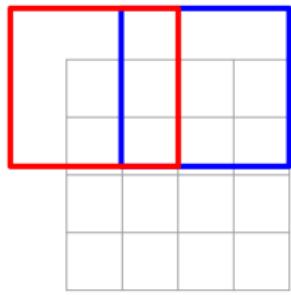
Dot product
between filter
and input



Output: 2×2

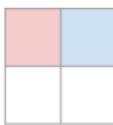
Learnable Upsampling: Transposed Convolution (cont.)

Recall: Normal 3×3 convolution, stride 2 pad 1



Input: 4×4

Dot product
between filter
and input



Output: 2×2

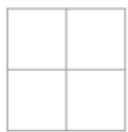
Filter moves 2 pixels in
the input for every one
pixel in the output

Stride gives ratio between
movement in input and
output

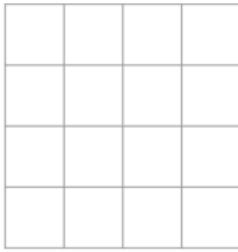
We can interpret strided
convolution as “learnable
downsampling”.

Learnable Upsampling: Transposed Convolution (cont.)

3 x 3 transposed convolution, stride 2 pad 1



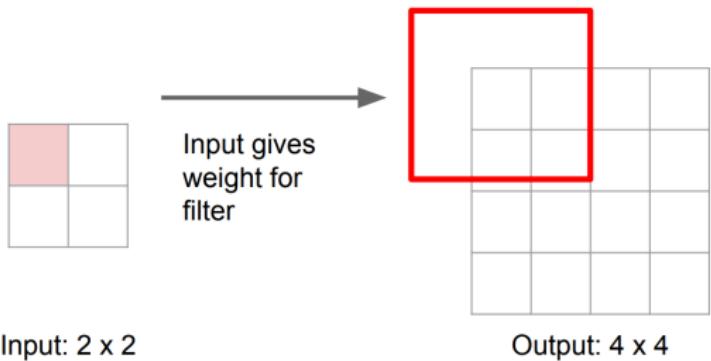
Input: 2 x 2



Output: 4 x 4

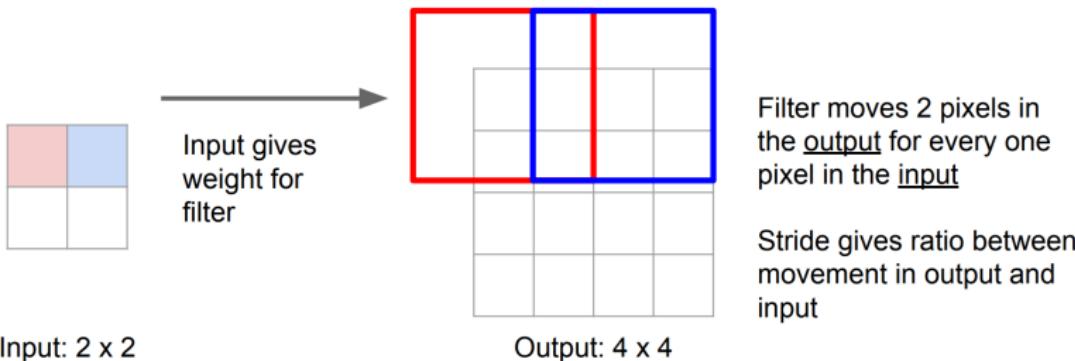
Learnable Upsampling: Transposed Convolution (cont.)

3 x 3 transposed convolution, stride 2 pad 1

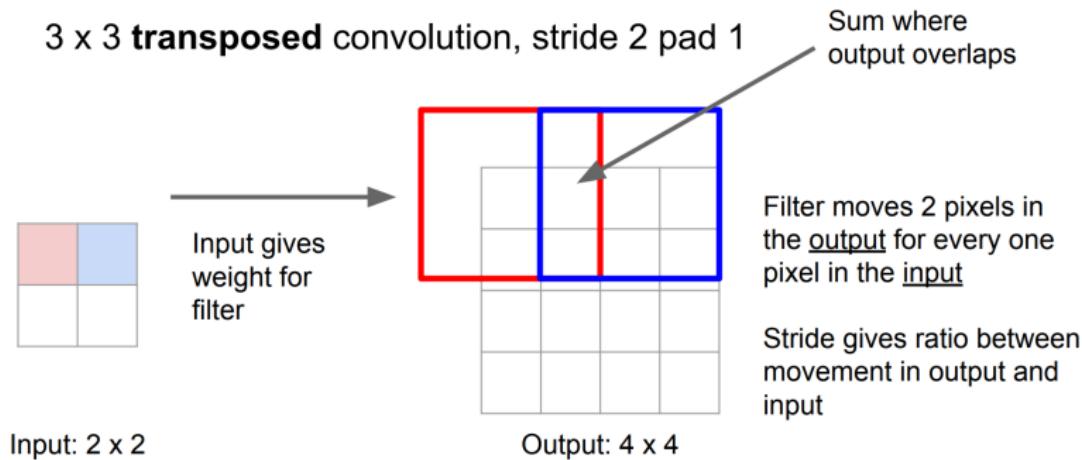


Learnable Upsampling: Transposed Convolution (cont.)

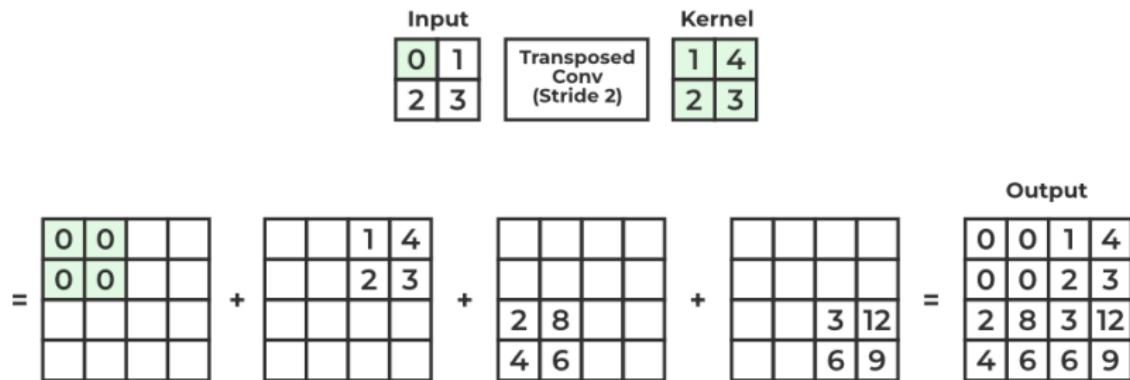
3 x 3 **transposed** convolution, stride 2 pad 1



Learnable Upsampling: Transposed Convolution (cont.)



Learnable Upsampling: Transposed Convolution (cont.)



Learnable Upsampling: Transposed Convolution (cont.)

Input	Kernel	Output								
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	<p>Transposed Conv (Stride 1)</p>	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>4</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	4	1	2	3
0	1									
2	3									
4	1									
2	3									
$= \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$	$+ \begin{matrix} 4 & 1 \\ 2 & 3 \end{matrix}$	$+ \begin{matrix} & & \\ 8 & 2 & \\ & & \\ 4 & 6 & \end{matrix} + \begin{matrix} & & \\ & 12 & 3 \\ & & \\ 6 & 9 & \end{matrix} = \begin{matrix} 0 & 4 & 1 \\ 8 & 16 & 6 \\ 4 & 12 & 9 \end{matrix}$								

Mathematical Definition: Transposed Convolution

The output $O(x, y)$ of a transposed convolution is computed as:

$$O(x, y) = \sum_{i,j} I(i, j) \cdot K(x - i \cdot s, y - j \cdot s)$$

where:

- ▶ $O(x, y)$ is the output at position (x, y) ,
- ▶ $I(i, j)$ is the input value at (i, j) ,
- ▶ $K(x', y')$ is the kernel value at (x', y') ,
- ▶ s is the stride.

Transposed Convolution Output Size Formula

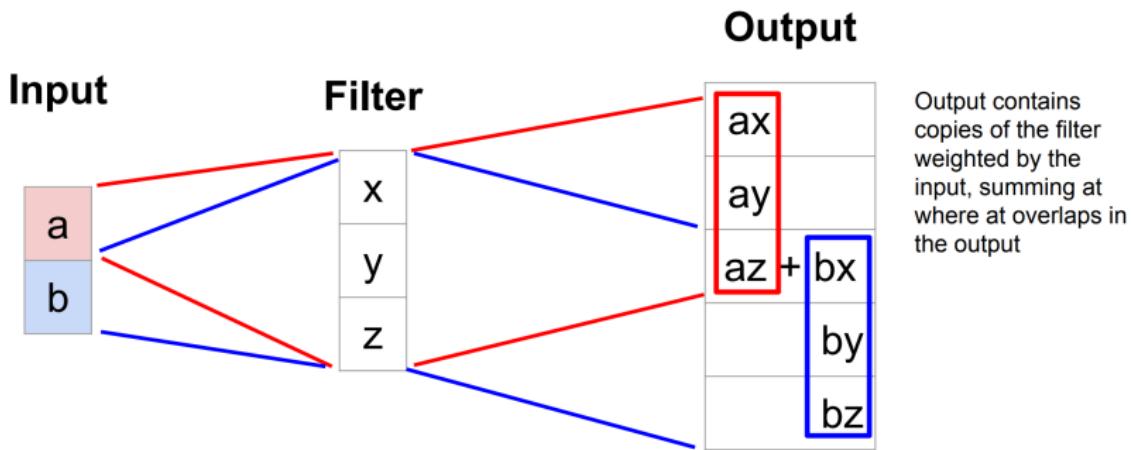
$$H_{\text{out}} = (H_{\text{in}} - 1) \times \text{stride}[0] - 2 \times \text{padding}[0] \\ + \text{dilation}[0] \times (\text{kernel_size}[0] - 1) \\ + \text{output_padding}[0] + 1 \quad (1)$$

$$W_{\text{out}} = (W_{\text{in}} - 1) \times \text{stride}[1] - 2 \times \text{padding}[1] \\ + \text{dilation}[1] \times (\text{kernel_size}[1] - 1) \\ + \text{output_padding}[1] + 1 \quad (2)$$

where:

- ▶ $H_{\text{out}}, W_{\text{out}}$ - Output height and width.
- ▶ $H_{\text{in}}, W_{\text{in}}$ - Input height and width.
- ▶ Stride - Step size of the filter movement.
- ▶ Padding - Number of pixels added around the input.
- ▶ Dilation - Spacing between kernel elements.
- ▶ Kernel size - Size of the convolution filter.
- ▶ Output padding - Additional padding applied to the output.

Transposed Convolution: 1D Example



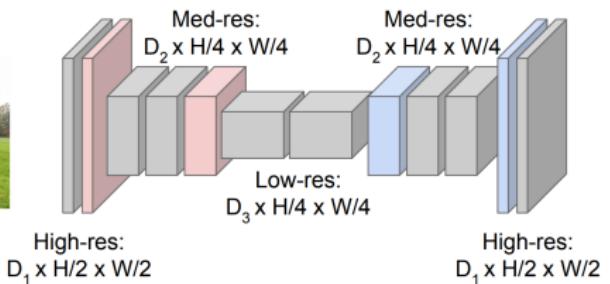
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
Unpooling or strided transposed convolution



Predictions:
 $H \times W$

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

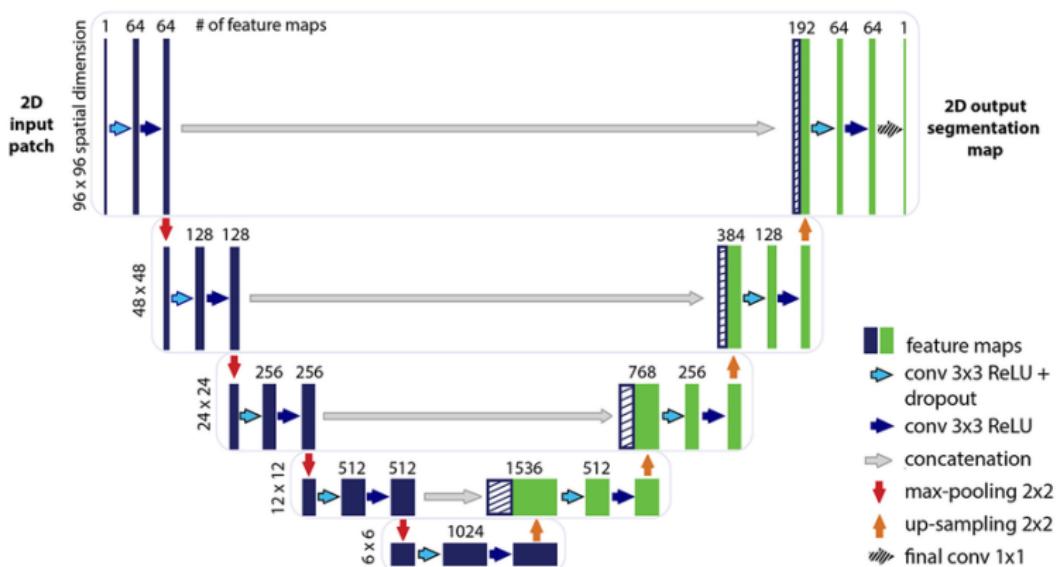
Can We Do Better?

- ▶ The downsampling-then-upsampling approach works well for semantic segmentation.
- ▶ **But... can we do better?**
- ▶ **Problem:** Important details and spatial information may be lost during downsampling.

- ▶ The downsampling-then-upsampling approach works well for semantic segmentation.
- ▶ **But... can we do better?**
- ▶ **Problem:** Important details and spatial information may be lost during downsampling.
- ▶ **Solution:** Introduce **residual connections** to preserve spatial information.

Residual Connections in Segmentation

- ▶ Directly connect features from downsampling layers to upsampling layers.
- ▶ Help recover lost spatial details and improve segmentation accuracy.



Residual Connections in Segmentation

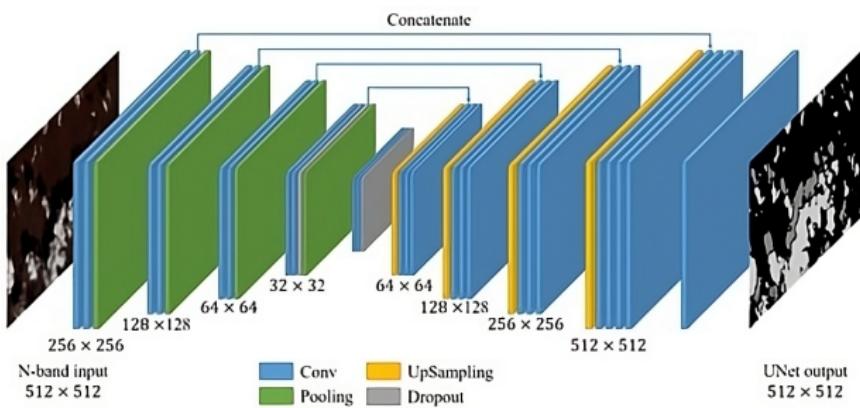
► There are two Types of Residuals:

- **Addition:** Adds features from the encoder to the decoder element-wise.
- **Concatenation:** Concatenates features from the encoder to the decoder along the **channel dimension**.

► Which Is Better?

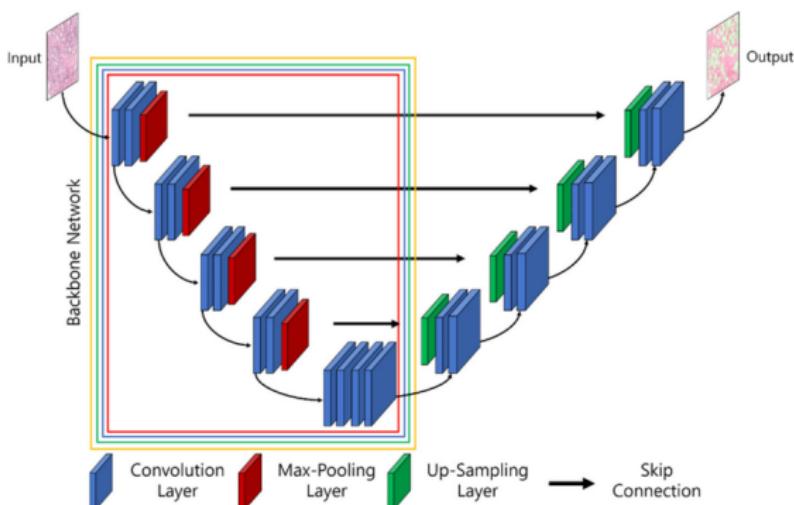
- Concatenation is often better because it retains more feature information from the encoder.
- **Note:** technically, concatenation might be harder to implement because it requires aligning input and output shapes.

- ▶ This architecture, with residual connections, is called **U-Net**.
- ▶ **Why the name?**
 - The architecture resembles the shape of the letter "U".
 - Features are downsampled in the encoder and upsampled in the decoder, with skip connections in between.
- ▶ U-Net is widely used for segmentation tasks, especially in biomedical imaging.



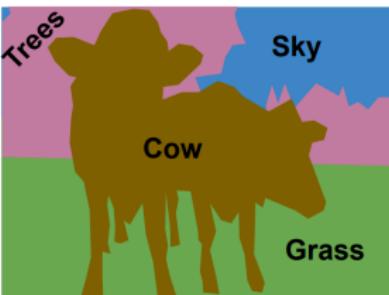
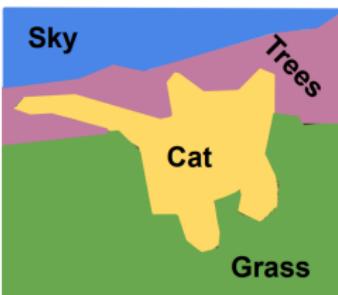
► Pretrained Encoder:

- The encoder can use a pretrained backbone (e.g., ResNet, EfficientNet).
- This helps utilize features learned on large datasets (e.g., ImageNet).
- Only the decoder is trained from scratch for segmentation-specific tasks.



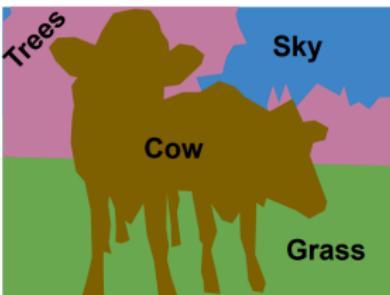
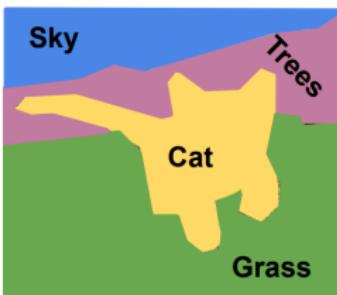
Semantic Segmentation

- ▶ Label each pixel in the image with a category label



Semantic Segmentation

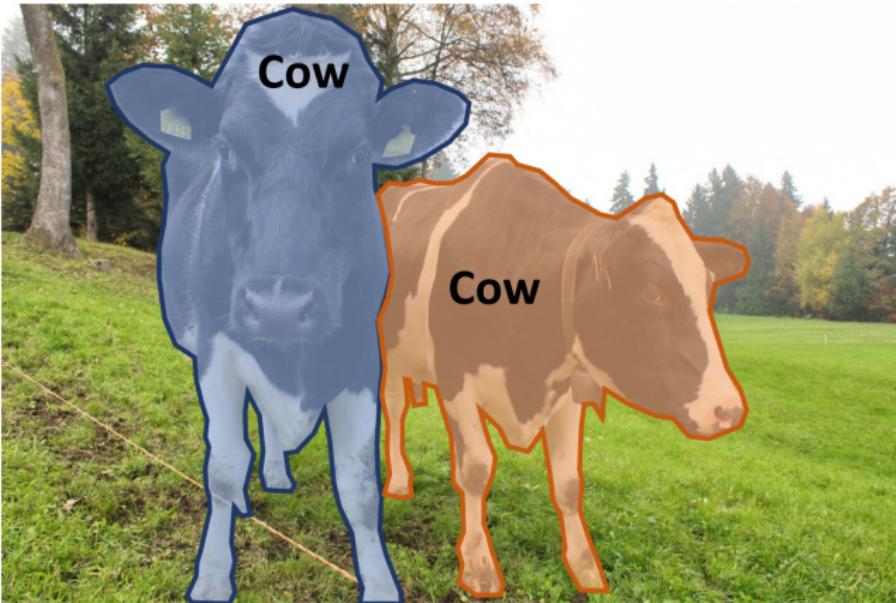
- ▶ Label each pixel in the image with a category label



- ▶ Does not differentiate instances, only care about pixels

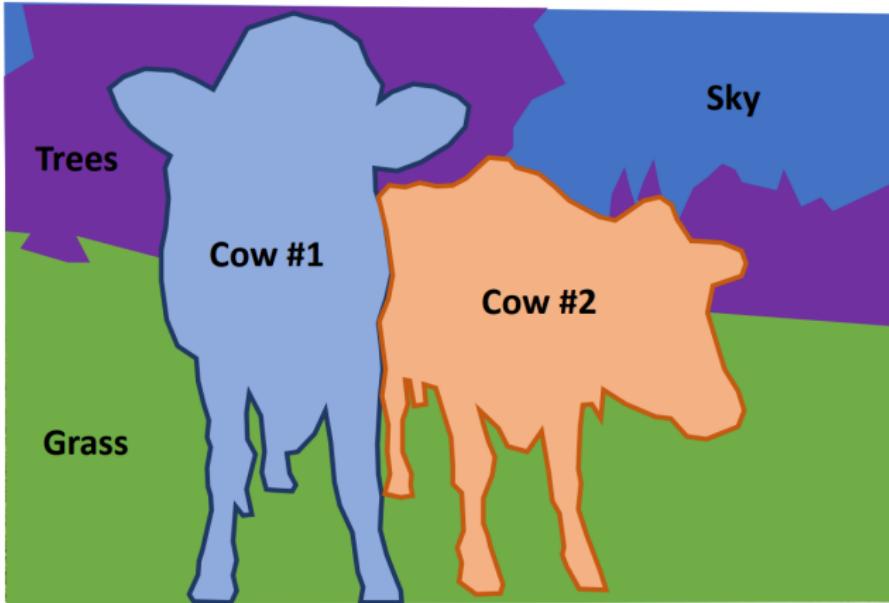
Instance Segmentation

- ▶ Separate object instances, but only things



Panoptic Segmentation

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



These slides have been adapted from

- ▶ Fei-Fei Li, Yunzhu Li & Ruohan Gao, Stanford CS231n: Deep Learning for Computer Vision
- ▶ Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
- ▶ Justin Johnson, UMich EECS 498.008/598.008: Deep Learning for Computer Vision