Image Segmentation:
Part 2 - Deep Learning

Matthew Vue and Maryam Vazirabad

Semantic Segmentation

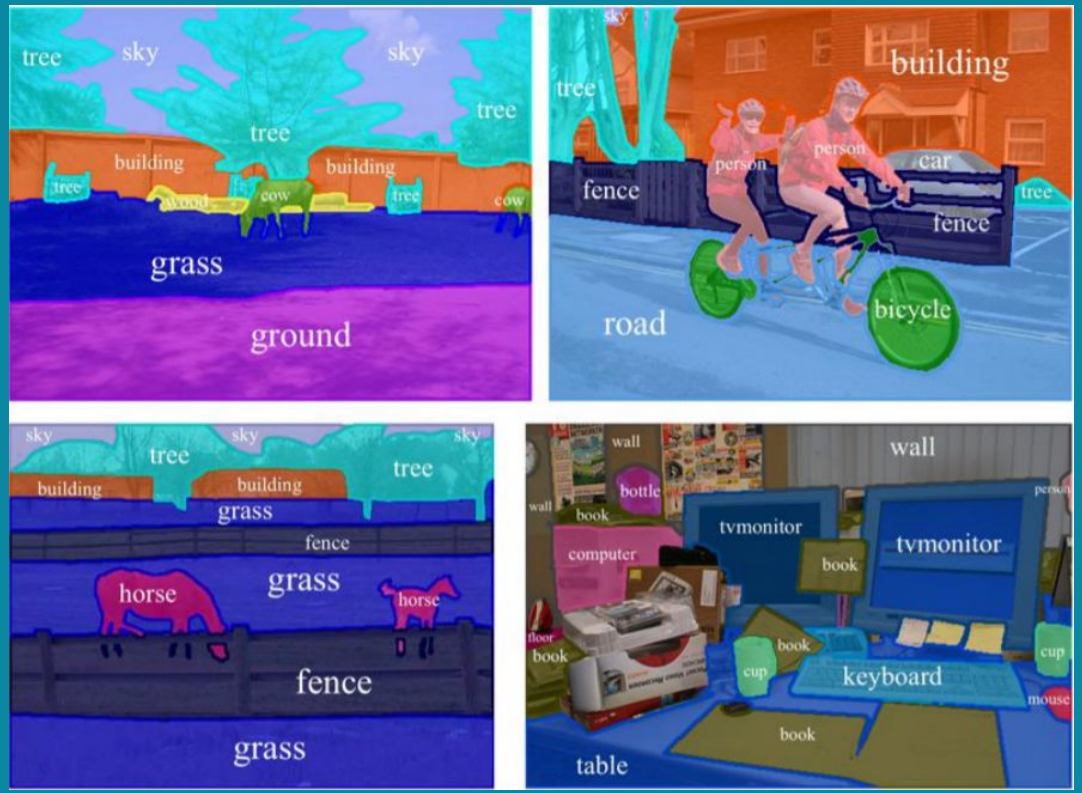
What is semantic segmentation?

Idea: Recognizing, understanding what's in the image in pixel level.

Goal: Label each pixel of an image with a corresponding class of what is being represented.

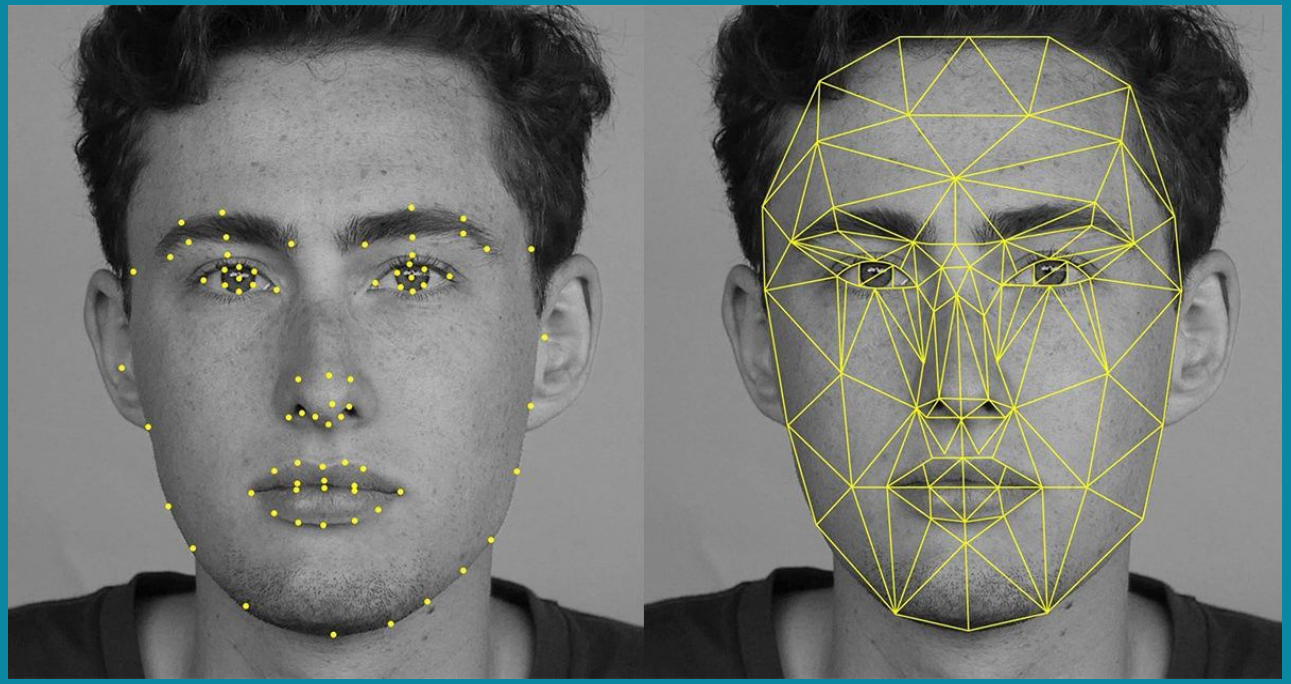
- A lot more difficult (Most of the traditional methods cannot tell different objects.)

Examples

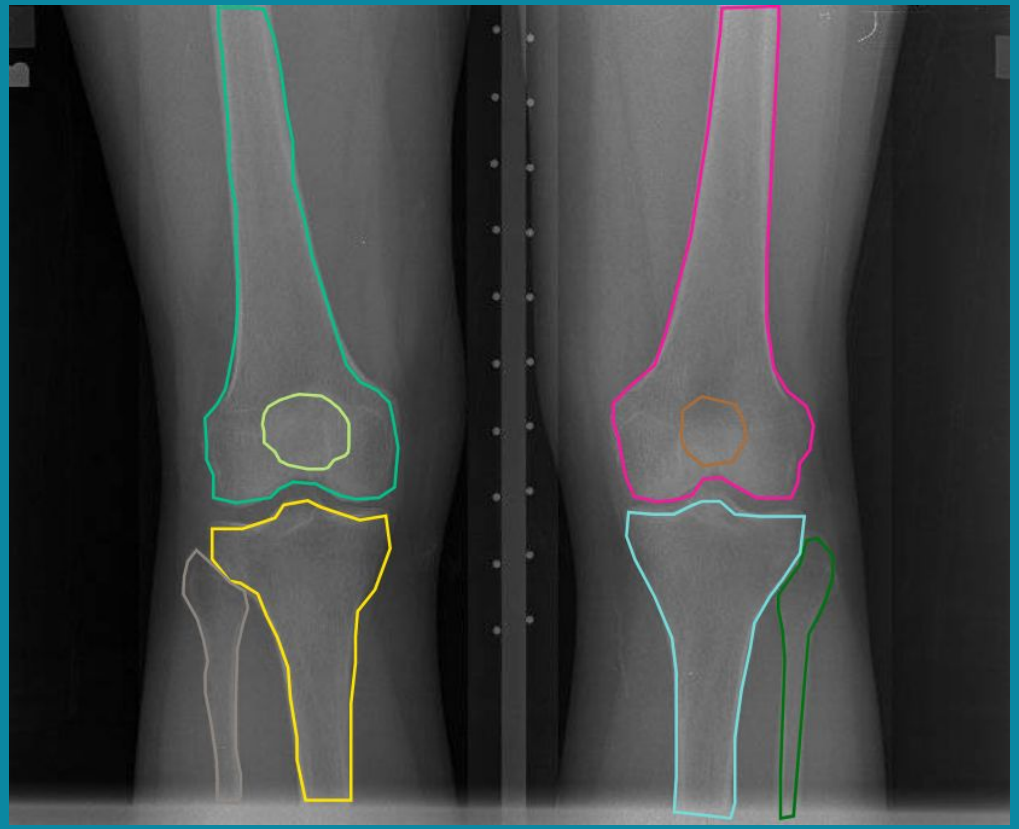

tree
sky
sky
tree
tree
building
building
tree
wood
cow
tree
cow
grass
ground
sky
tree
building
person
person
car
fence
tree
fence
road
bicycle
sky
tree
sky
sky
building
building
tree
grass
fence
horse
grass
horse
fence
grass
wall
wall
bottle
wall
person
book
tvmonitor
tvmonitor
computer
book
book
floor
book
cup
keyboard
cup
mouse
book
table

Applications: Autonomous vehicles

Applications: Facial Recognition



Applications: Healthcare



Semantic Segmentation Methods

- Sliding-Window Approach
- Fully Convolutional Network (FCN)
- Fully Convolutional Network with Downsampling and Upsampling
- U-Net

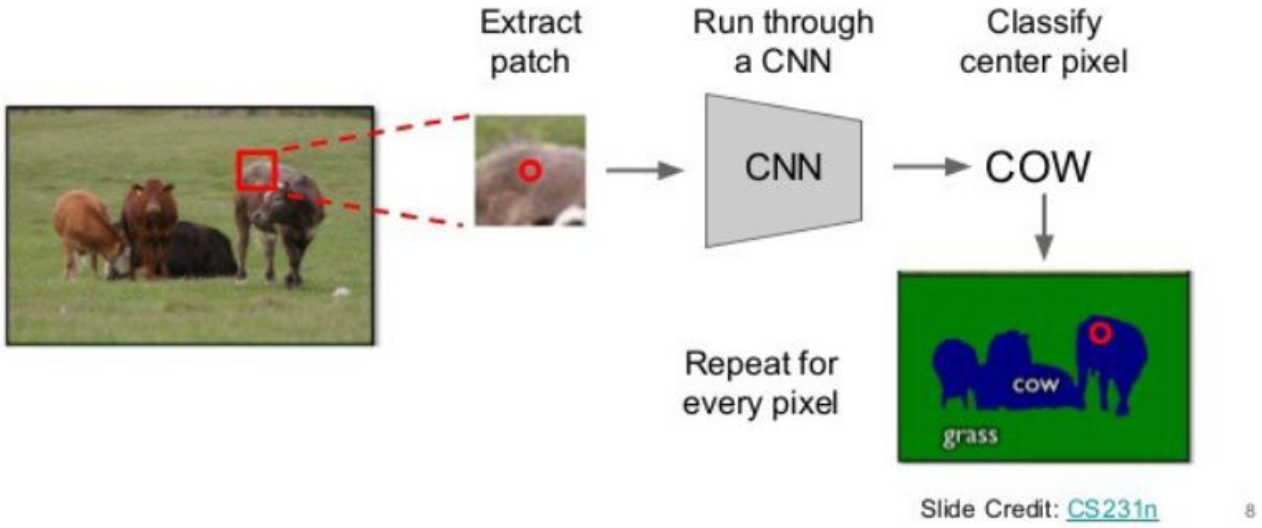
Semantic Segmentation Methods

- Sliding-Window Approach
- Fully Convolutional Network (FCN)
- Fully Convolutional Network with Downsampling and Upsampling
- U-Net

Sliding-Window Approach

Simple concept: Takes a patch from the input image and feeds it into a CNN to classify the center pixel as output.

- Does this for each of the N pixels in the image



Semantic segmentation using sliding window

Sliding-Window Approach Problem

Computationally Inefficient:

- Image patches will overlap and reprocessing them leads to unwanted computation
- We need to find a way to reduce the number of passes for an image
- How can we make predictions for pixels in an image all at once?

Semantic Segmentation Methods

- Sliding-Window Approach
- Fully Convolutional Network (FCN)
- Fully Convolutional Network with Downsampling and Upsampling
- U-Net

Fully Convolutional Network (FCN)

- No fully connected layer at end of network
- Height and width of the image remain the same
- Predicts all pixel categories at once
- Output is C feature maps
  - C is the number of categories/classes, including the background, a pixel can belong to
  - Each segmented map corresponds to a class
- Input activation or score of a pixel at the (i,j) coordinate for the kth class is denoted by $S_k(i,j)$

- Probability of the kth class at the (i,j) coordinate is $$P_k(i,j) = \frac{e^{s_k^{(i,j)}}}{\sum_{k'=1}^{C} e^{s_{k'}^{(i,j)}}}$$
- Ground truth labels at the (i,j) coordinate denoted as $y_k(i,j)$
- Cross entropy loss of the pixel at the (i,j) location is $$L(i,j) = -\sum_{k=1}^{C} y_k(i,j) \log P_k(i,j)$$
- If the height and width of the images are M x N respectively, the the total image loss is $$L = -\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=1}^{C} y_k(i,j) \log P_k(i,j)$$
- The output class for a pixel at the (i,j) coordinate is determined by taking the class k for which $P_k(i,j)$ is maximum
- Repeat for all pixels

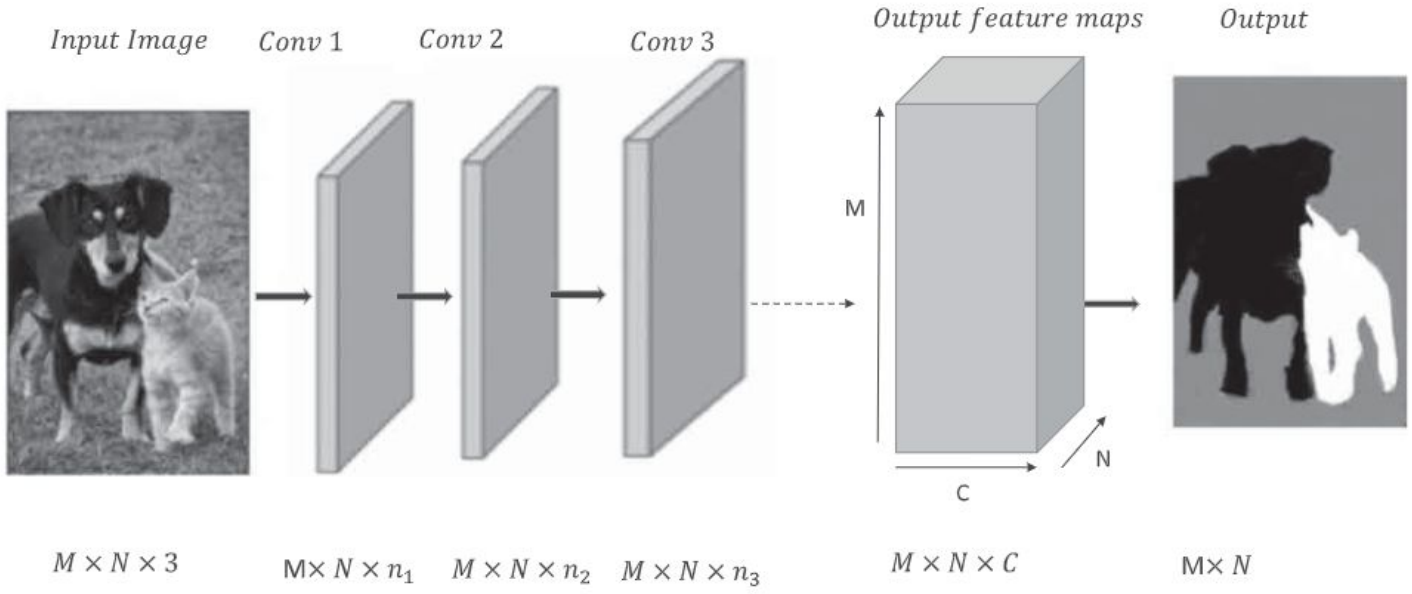


Figure 6-7. *Fully convolutional network architecture*

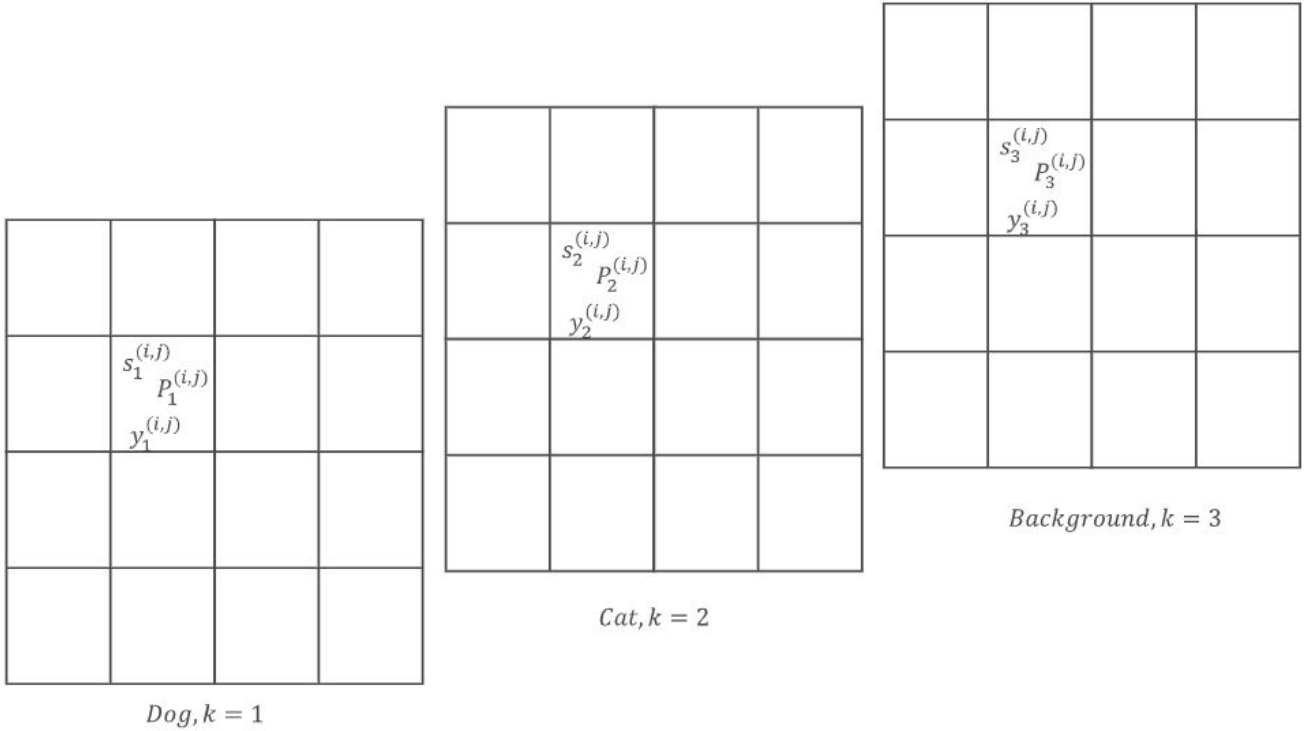


Figure 6-8. *Output feature maps corresponding to each of the three classes for dog, cat, and background*

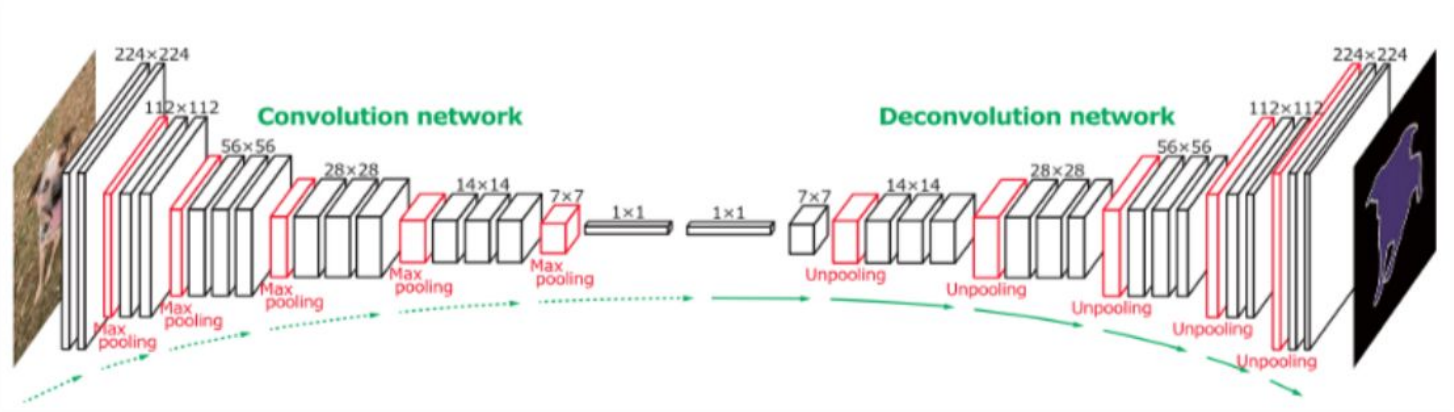
Benefits of using Fully Convolutional Networks

- More efficient than sliding-window approach: Less computational cost (from predicting pixel class labels all at once)
- Can apply the network to images of virtually any size (a fully connected layer only expects a certain size input)

Semantic Segmentation Methods

- Sliding-Window Approach
- Fully Convolutional Network (FCN)
- Fully Convolutional Network with Downsampling and Upsampling
- U-Net

Fully Convolutional Network with Downsampling and Upsampling



Fully convolutional layer for semantic segmentation

Fully Convolutional Network with Downsampling and Upsampling

In regular fully Convolutional Networks, all the convolutions retain the spatial dimensions of the input image.

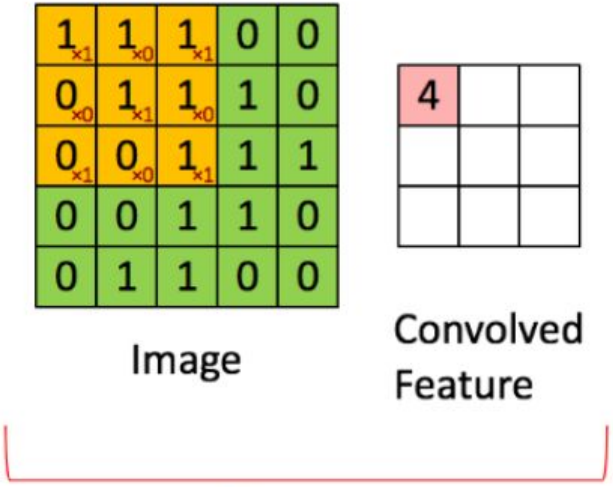
- For high-resolution images the network would be computationally expensive

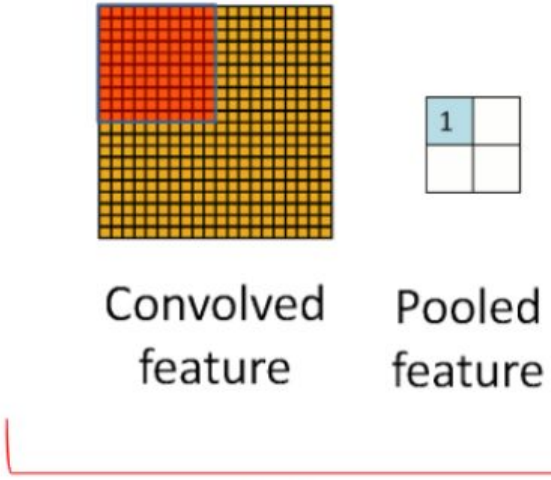
What if we downsample in the first half of the network and upsample in the second half to restore the spatial dimensions to the original size?

Review: Downsampling (pooling)

Main function: to reduce the size of the input images to make computation faster and retain imp

- Done by using a specified filter and stride


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
4
Image
Convolved Feature
Convolution


1
Convolved feature
Pooled feature
Pooling

Max pooling

Filter Size: 2 * 2
Stride: 2

Types of pooling

Average Pooling: Calculate the average value for each patch on the feature map.

Sum pooling: Calculate the sum for each patch on the feature map.

Min pooling: Calculate the minimum value for each patch of the feature map.

Max pooling: Calculate the maximum value for each patch of the feature map.

Upsampling

Main idea: After downsampling, we need to restore the spatial dimensions of the image so we can classify each pixel

Upsampling: Different techniques

1. Unpooling
2. Max unpooling
3. Transpose convolution

Unpooling

In unpooling, we increase the spatial dimensions of the image by repeating a pixel value in a neighborhood

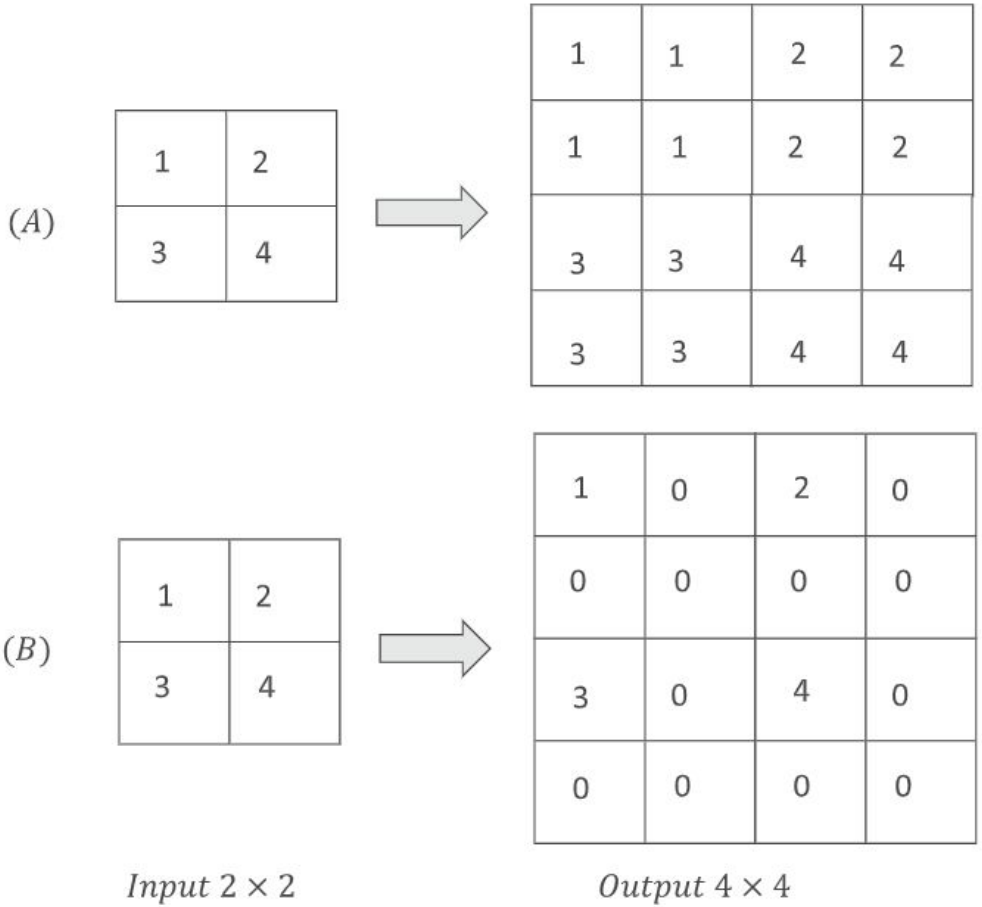


Figure 6-10. *Unpooling operation*

Max unpooling

Place the value of the input pixel in the output location corresponding to the one where the max pooling output got its input from

Therefore, we need to remember the position of each maximum activation value when doing max pooling

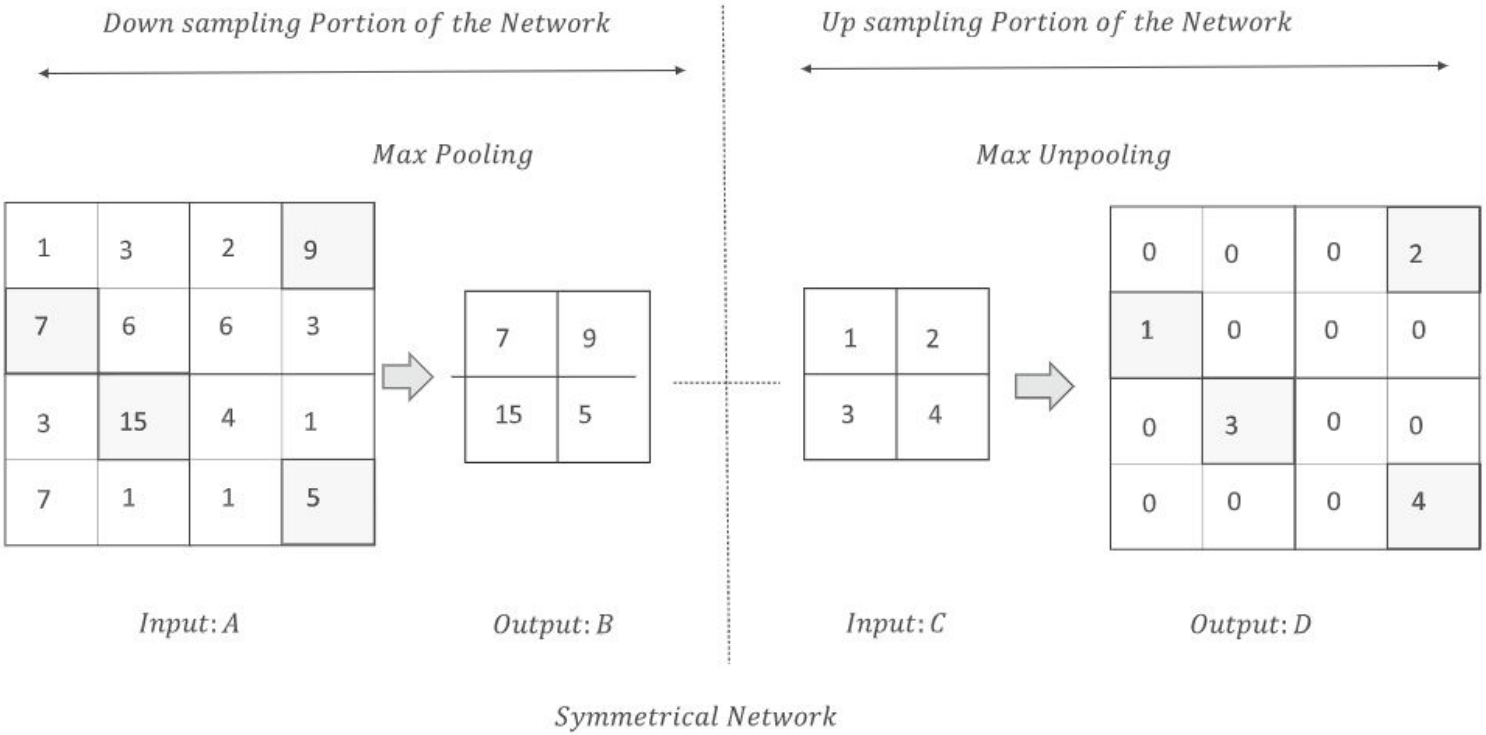


Figure 6-11. *Max unpooling illustration for a symmetric fully connected segmentation network*

Transpose convolution

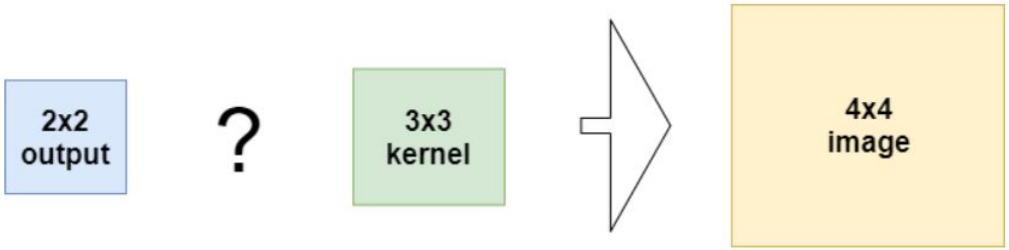
Unlike unpooling or max unpooling, transpose convolution involves parameters that would be learned by the network.

- Using filters: similar to how convolution works

Transpose convolution

Strides greater than 1 provide upsampling. So, if we use a stride of 2, then the input size is doubled in each spatial dimension.

Operation of transpose convolution for an input of dimension 2x2 by a filter or kernel size of 3x3 to produce a 4x4 output.



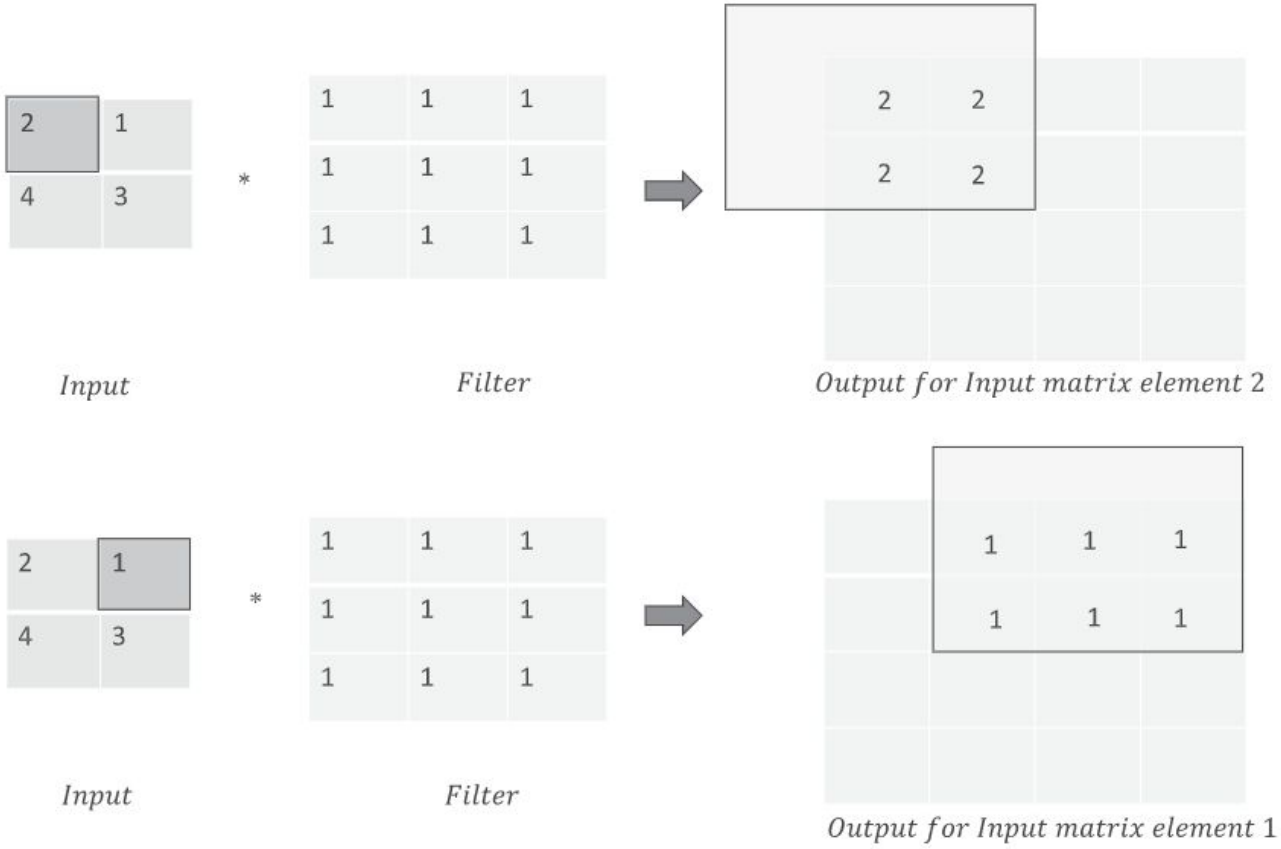


Figure 6-13a. *Transpose convolution for upsampling*

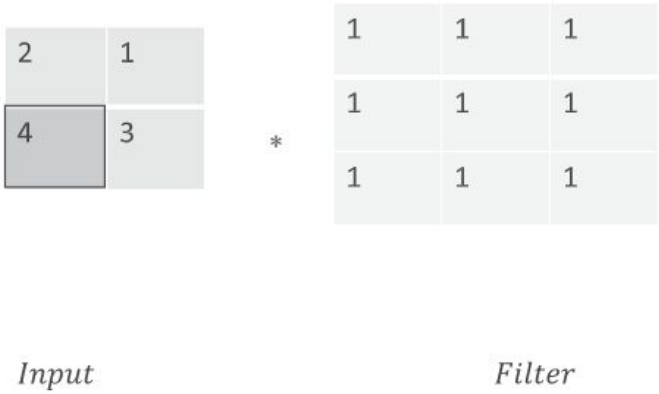


Figure 6-13b. *Transpose convolution for upsampling*

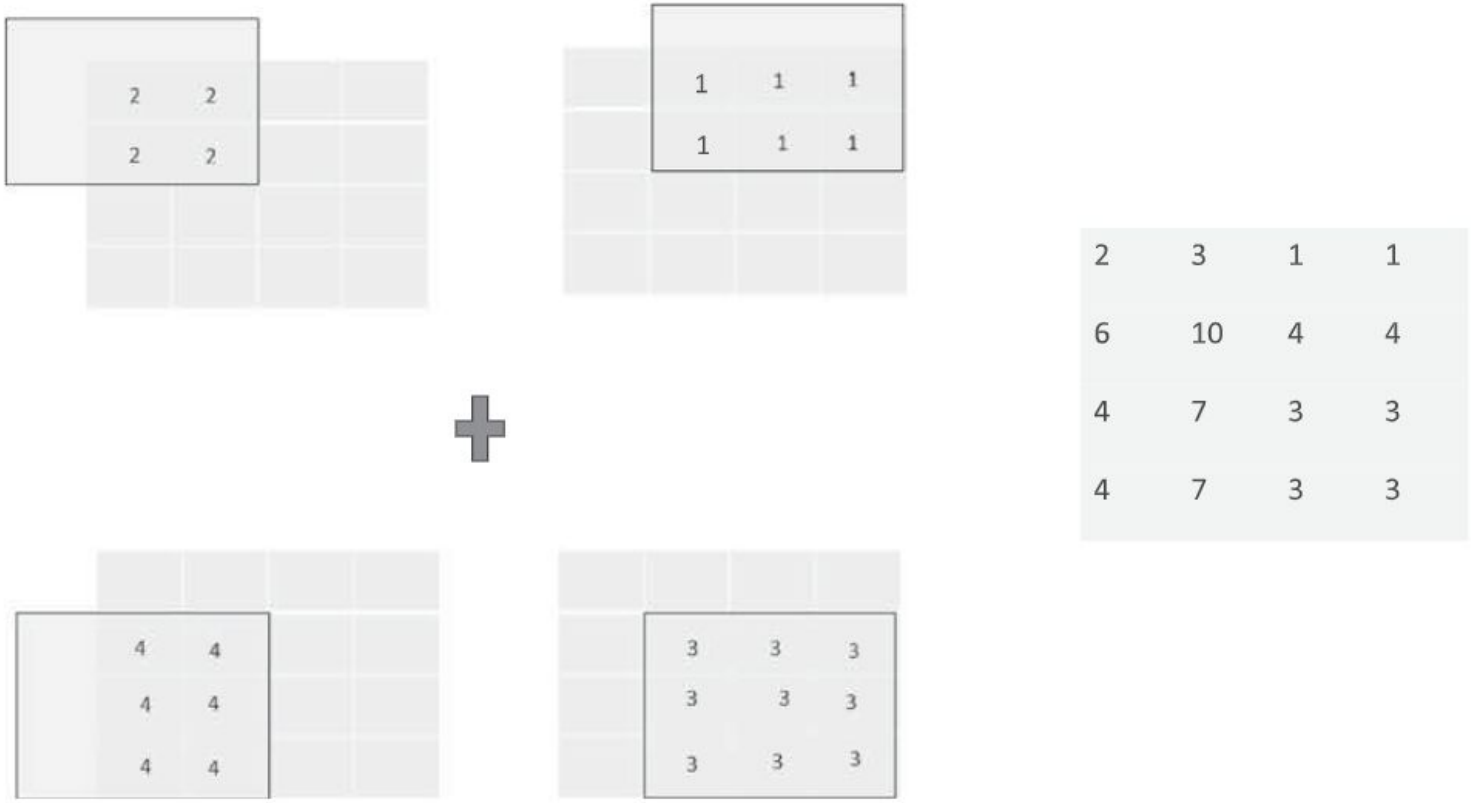


Figure 6-13c. *Transpose convolution for upsampling*

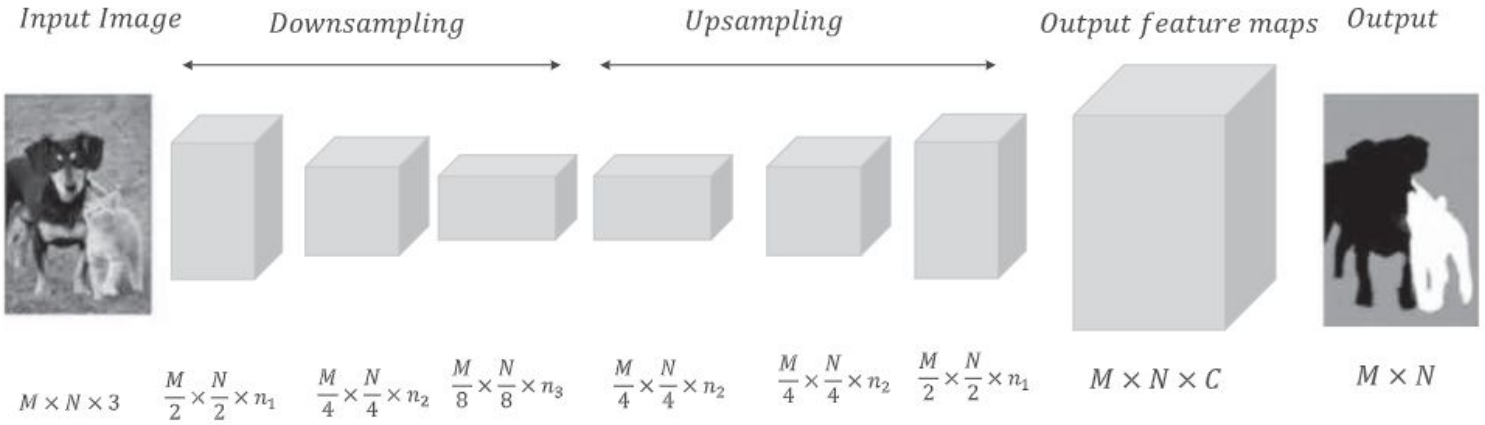
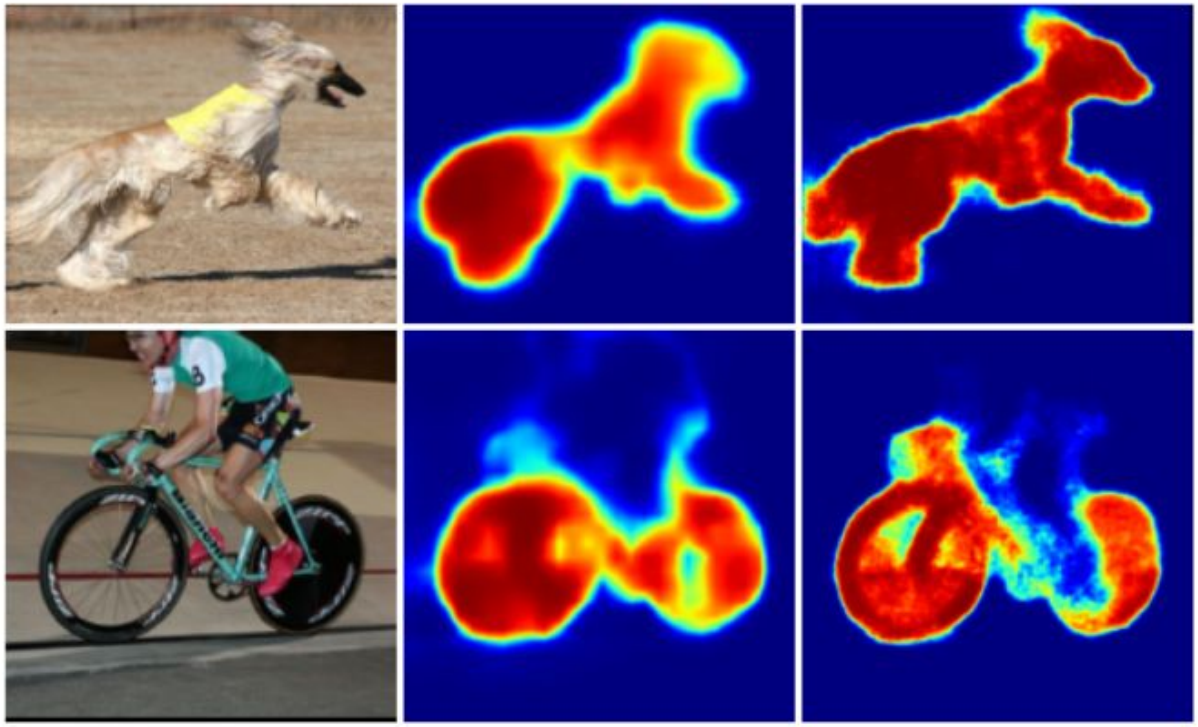


Figure 6-9. *Fully convolutional network with downsampling and upsampling*



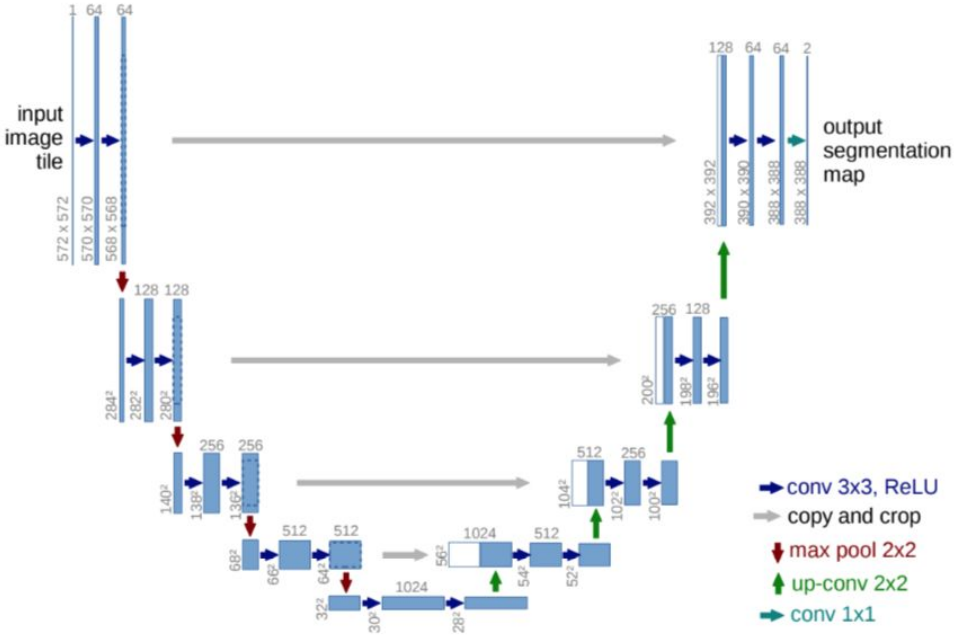
Input Image (Left), FCN-8s (Middle), DeconvNet (Right)

Semantic Segmentation Methods

- Fully Convolutional Network (FCN)
- Fully Convolutional Network with Downsampling and Upsampling
  U-Net

U-Net

The u-net comprises of two parts a contraction path (left side) and an expansion path (right side).

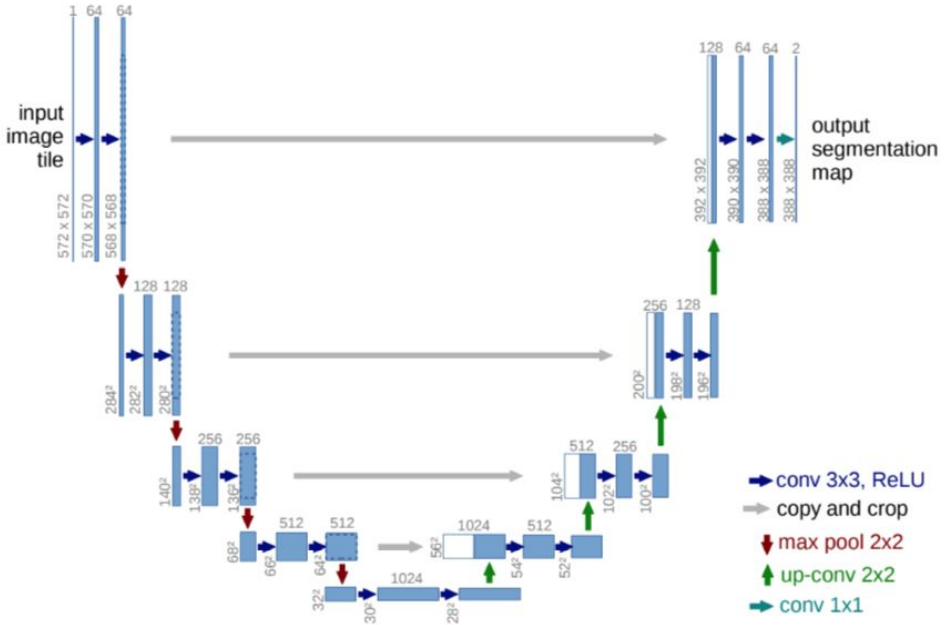


U-Net

The contracting path is composed of 4 blocks. Each block is composed of

3x3 Convolution Layer + activation function (ReLu)
3x3 Convolution Layer + activation function (ReLu)
2x2 Max Pooling and doubling number of feature channels (64 -> 128, etc.)

- Skip connections: contextual information transferred to the upsampling path

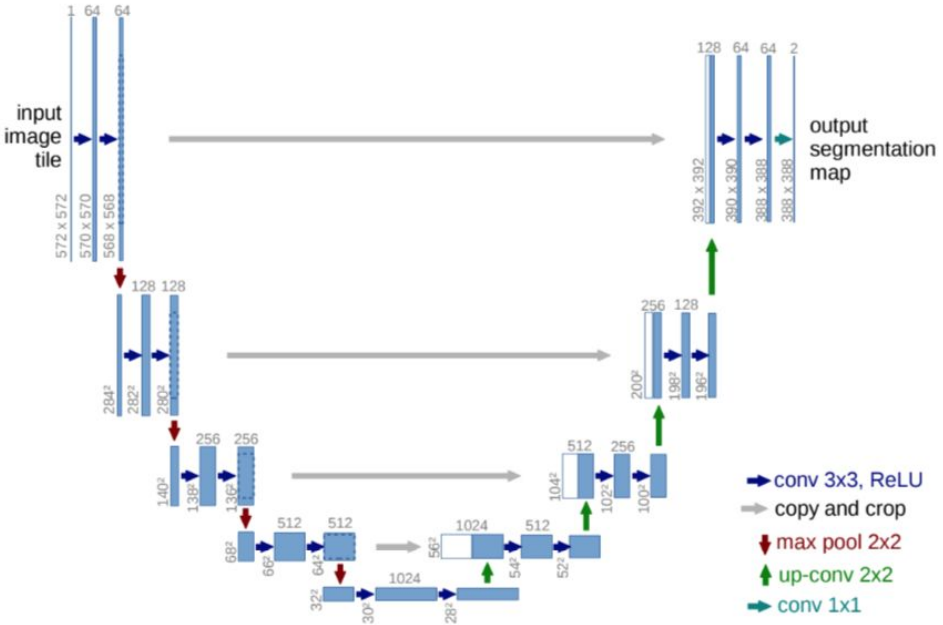


U-Net

Bottleneck

This part of the network is between the contracting and expanding paths.

The bottleneck is built from simply 2 convolutional layers with dropout.



U-Net

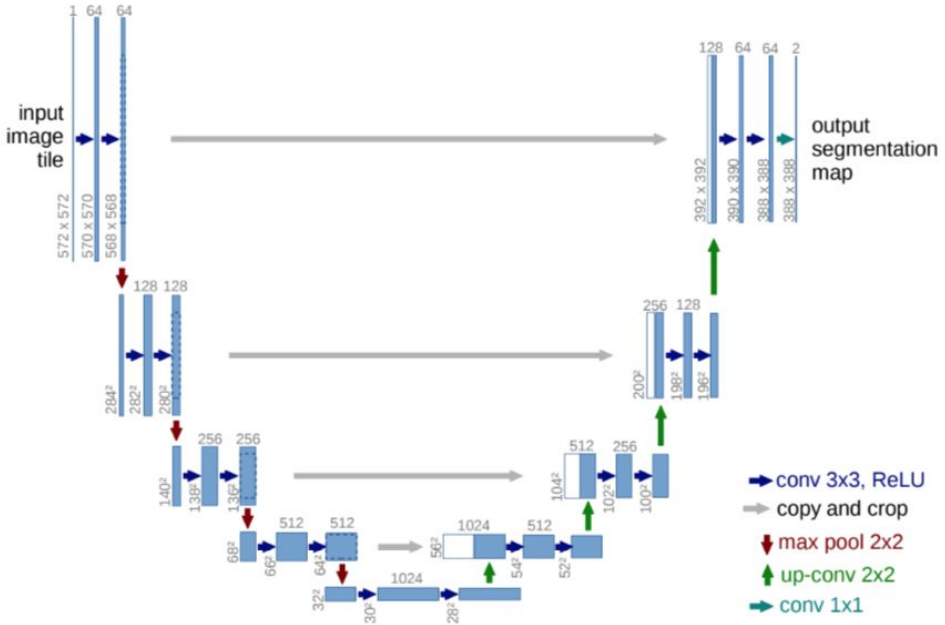
The expanding path is also composed of 4 blocks. Each of these blocks is composed of

Deconvolution layer with stride 2

Concatenation with the corresponding cropped feature map from the contracting path

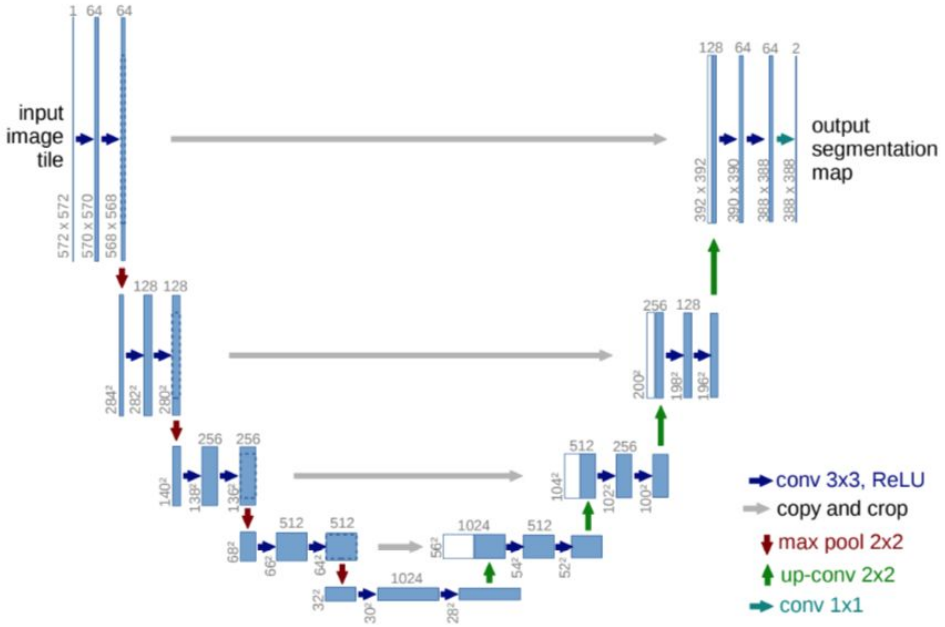
3x3 Convolution layer + activation function
3x3 Convolution layer + activation function

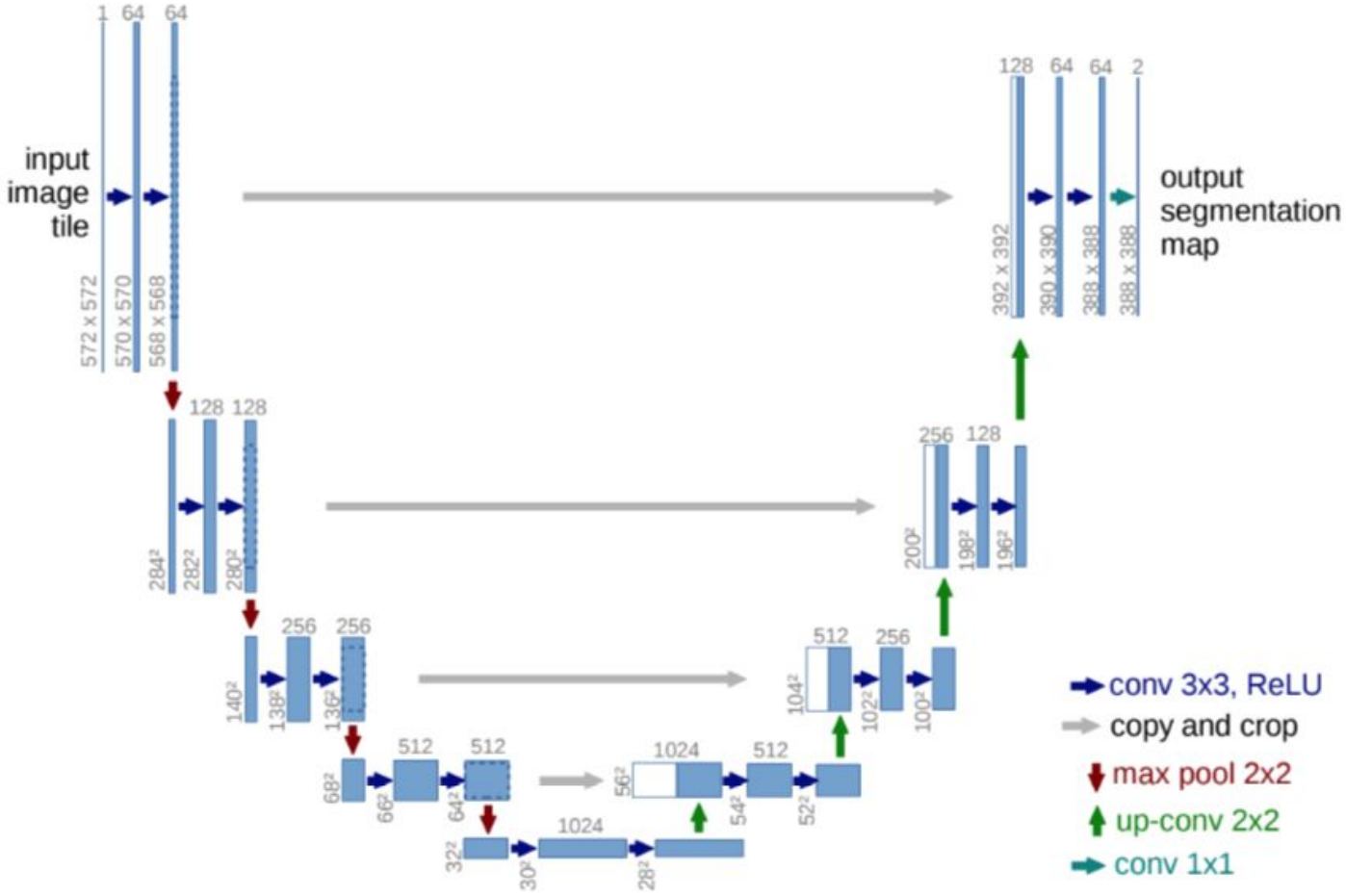
The purpose of this expanding path is to enable precise localization and information from skip connections



U-Net

At the final layer, a 1x1 convolution is used to map each 64-components feature vector to the desired number of classes.




input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Improvements with U-Net

- A significant amount of training data can be generated with only a few annotated or hand-labeled segmented images.

- The U-Net does good segmentation even when there are touching objects of the same class that need to be separated. Methods such as the Watershed algorithm require a lot of input to come up with reasonable segmentation. The U-Net does good separation between touching segments of the same class by introducing high weights for misclassification of pixels around the borders of touching segments