Semantic Segmentation

Dr. Alireza Aghamohammadi

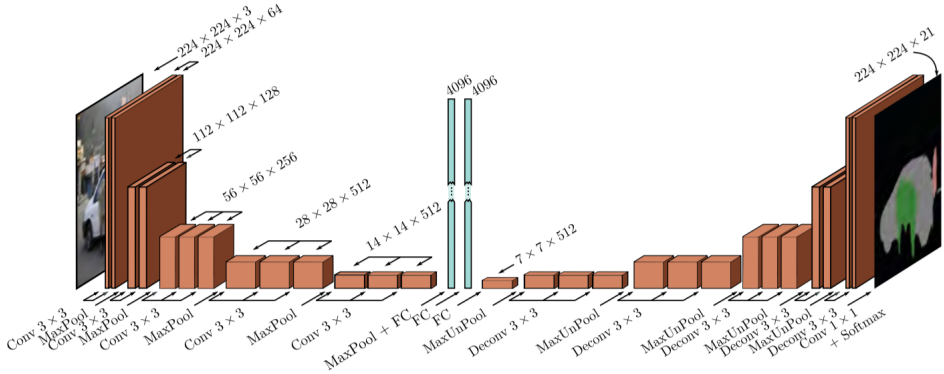
Motivation

- Semantic segmentation is a process where each pixel in an image is assigned a label.
- These labels correspond to the object that the pixel is part of. If a pixel doesn't match any object in the training database, it is left unlabeled[1].

[1]Reference: "Learning Deconvolution Network for Semantic Segmentation"

Problem Statement

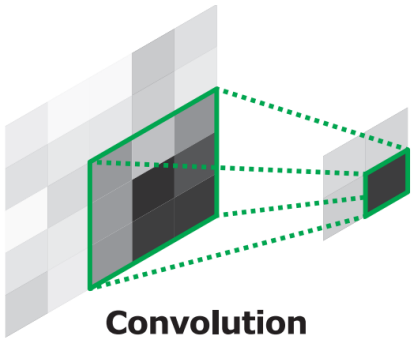
- Objective: Our goal is to ensure that the output maintains the same resolution as the input.
- Challenge: The task is to devise a network architecture that renders this per-pixel classification problem computationally feasible[2].

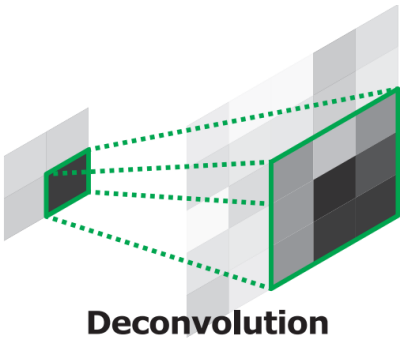


[2]Reference: "Understanding Deep Learning"

Transposed Convolution

- The requirement for transposed convolutions typically stems from the need to apply a transformation that operates in the reverse direction of a standard convolution[3].





[3]Reference: "Learning Deconvolution Network for Semantic Segmentation"

Matrix Multiplication Representation of Convolution

- The convolution operation can be represented as a matrix multiplication.
- Let's consider a kernel $k = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 1 & 2 \end{bmatrix}$ and an input $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix}$.
- We can compute the convoluted layer with a stride of 1 using the following operation:

$$\begin{bmatrix} 1 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{13} \\ x_{14} \\ x_{15} \\ x_{16} \end{bmatrix}$$

- This linear operation transforms the input matrix into a 16-dimensional vector and generates a 4-dimensional vector. This vector is subsequently reshaped into a 2×2 output matrix.

Matrix Multiplication Representation of Transposed Convolution

- Let's denote C as the sparse matrix representation of the kernel weights.
- Instead of using C in matrix multiplication, we use its transpose, denoted as C^T.
- This operation takes a 4-dimensional vector as input and generates a 16-dimensional vector as output.
- This process allows us to increase the size of the output feature map compared to the input feature map.

$$\text{output} = s \cdot (n-1) + k - 2p$$

- Here, s represents the stride, n represents the height/width of the input, k represents the kernel size, and p represents the padding (e.g., same or valid).