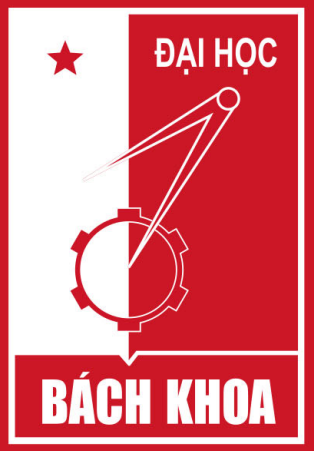


ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Lesson 10:
Image semantic segmentation

Outline

- Introduction to image segmentation problem
- Up-sampling layers
- Objective functions
- Some typical image segmentation networks

Introduction to image segmentation problem

Computer vision problems

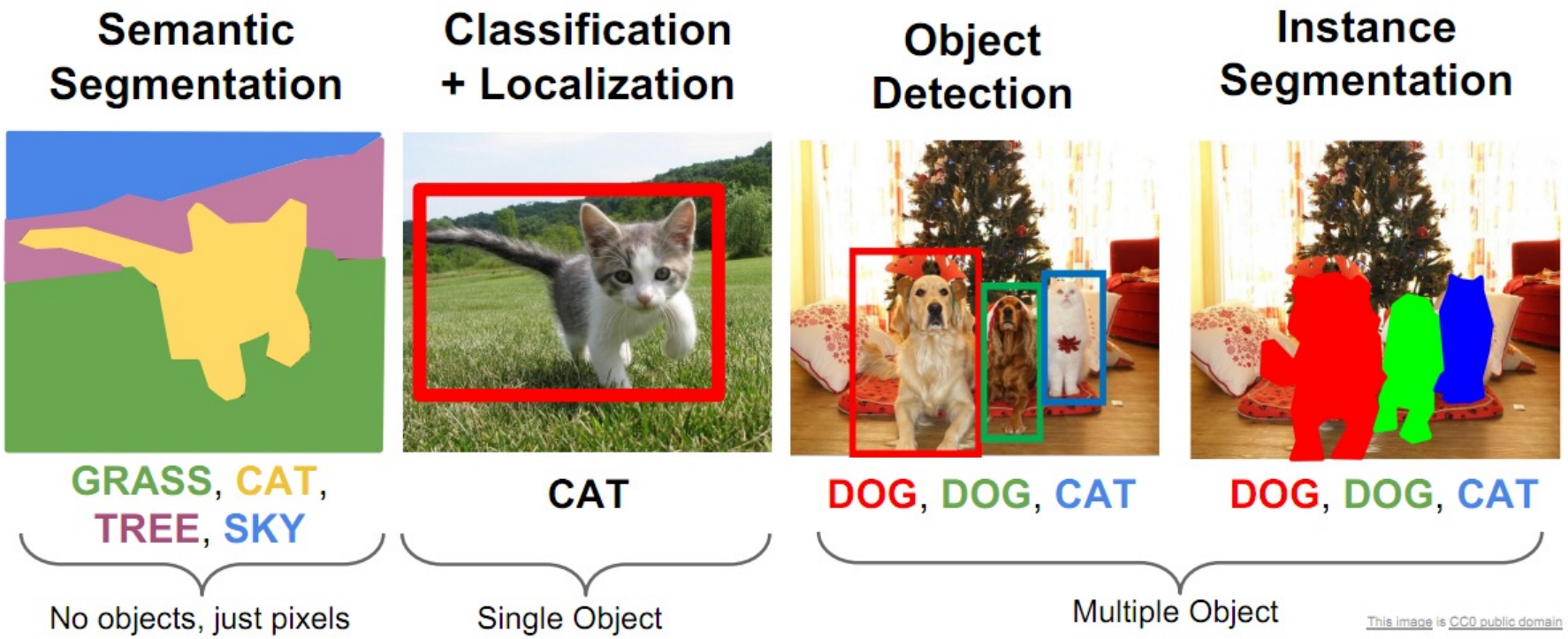
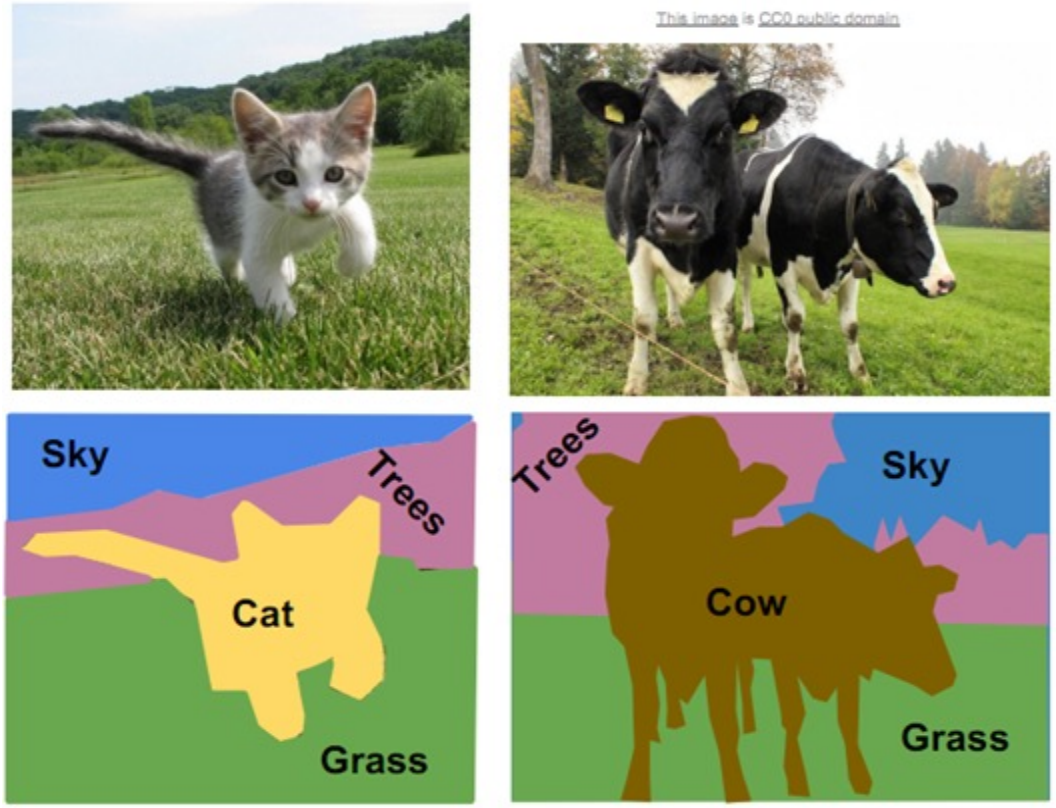


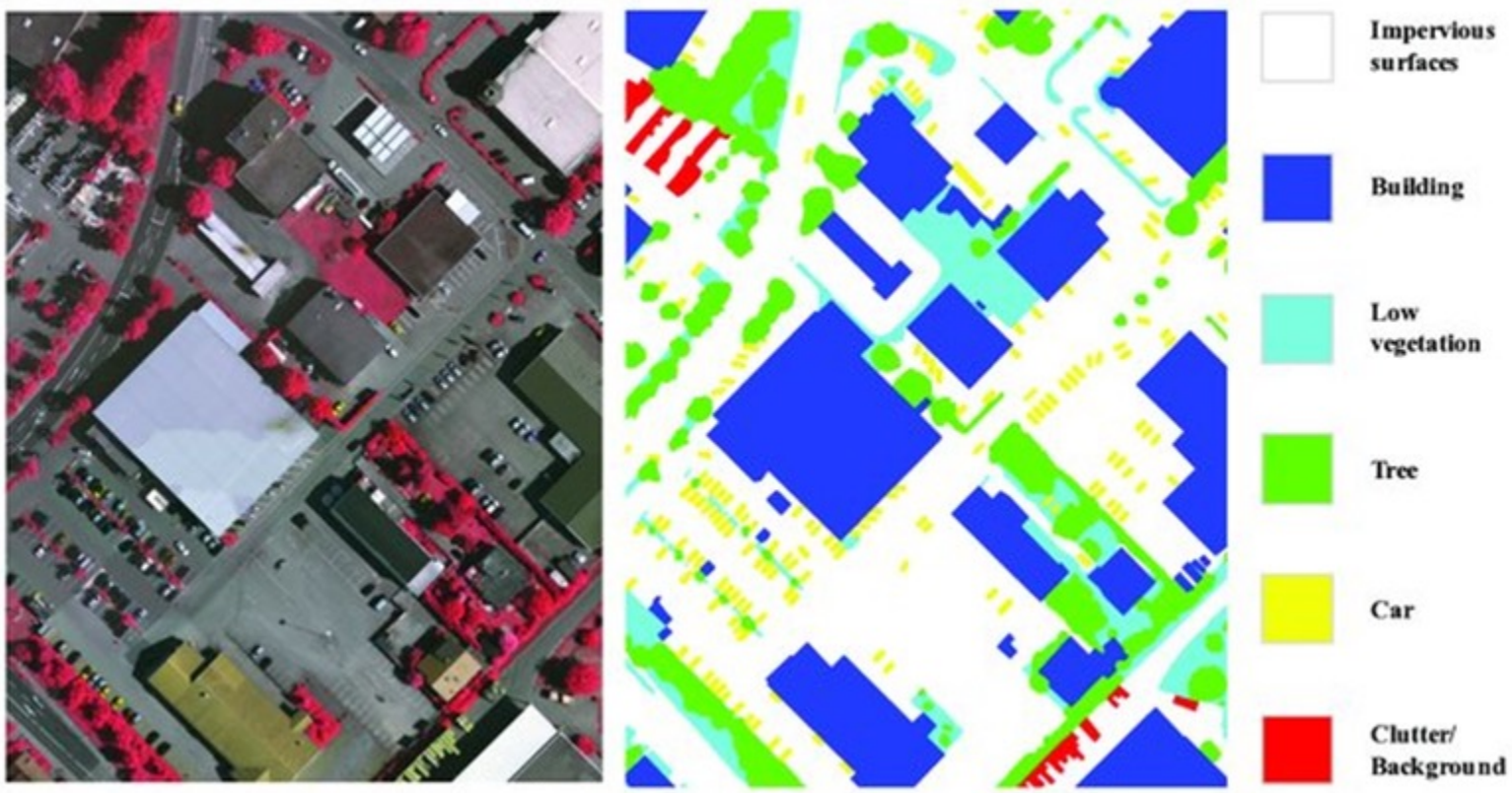
Image segmentation

- Classify each pixel in an image
- Does not distinguish objects of the same class in the image



Some image segmentation applications

- Satellite and aerial image segmentation

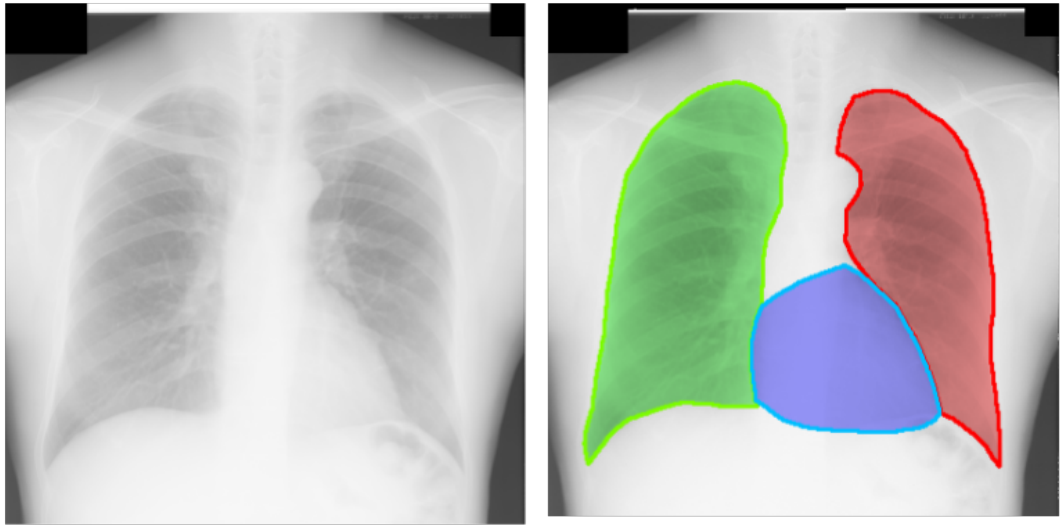


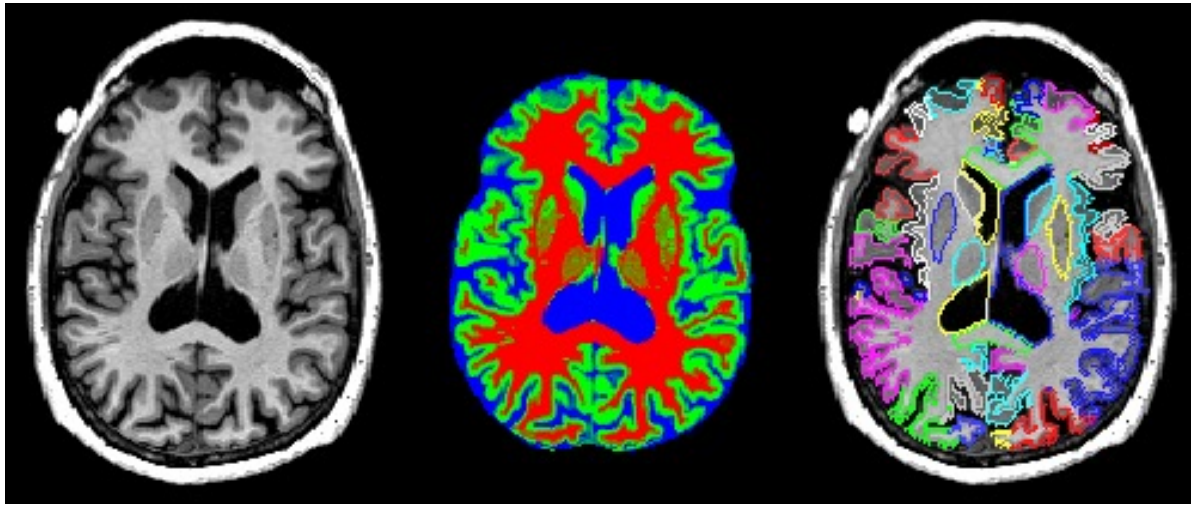
Some image segmentation applications

- Autonomous vehicle

Some image segmentation applications

- Healthcare

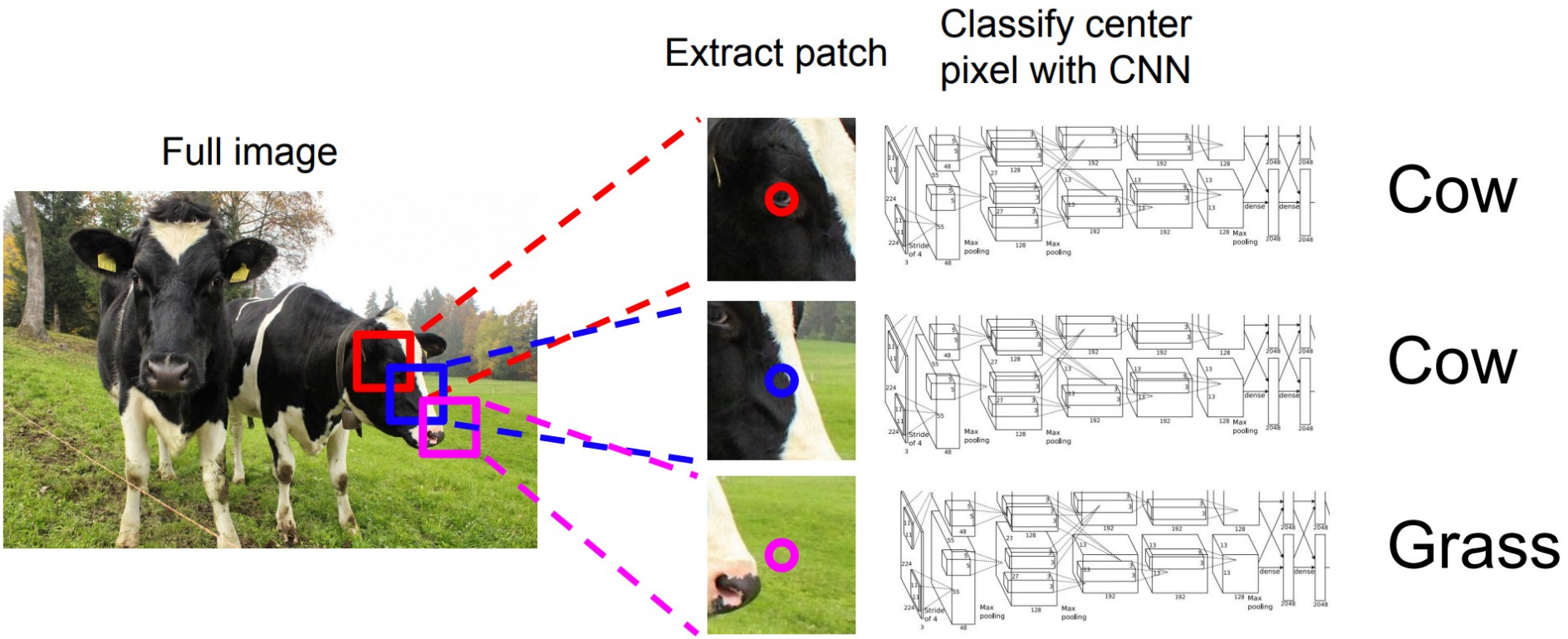




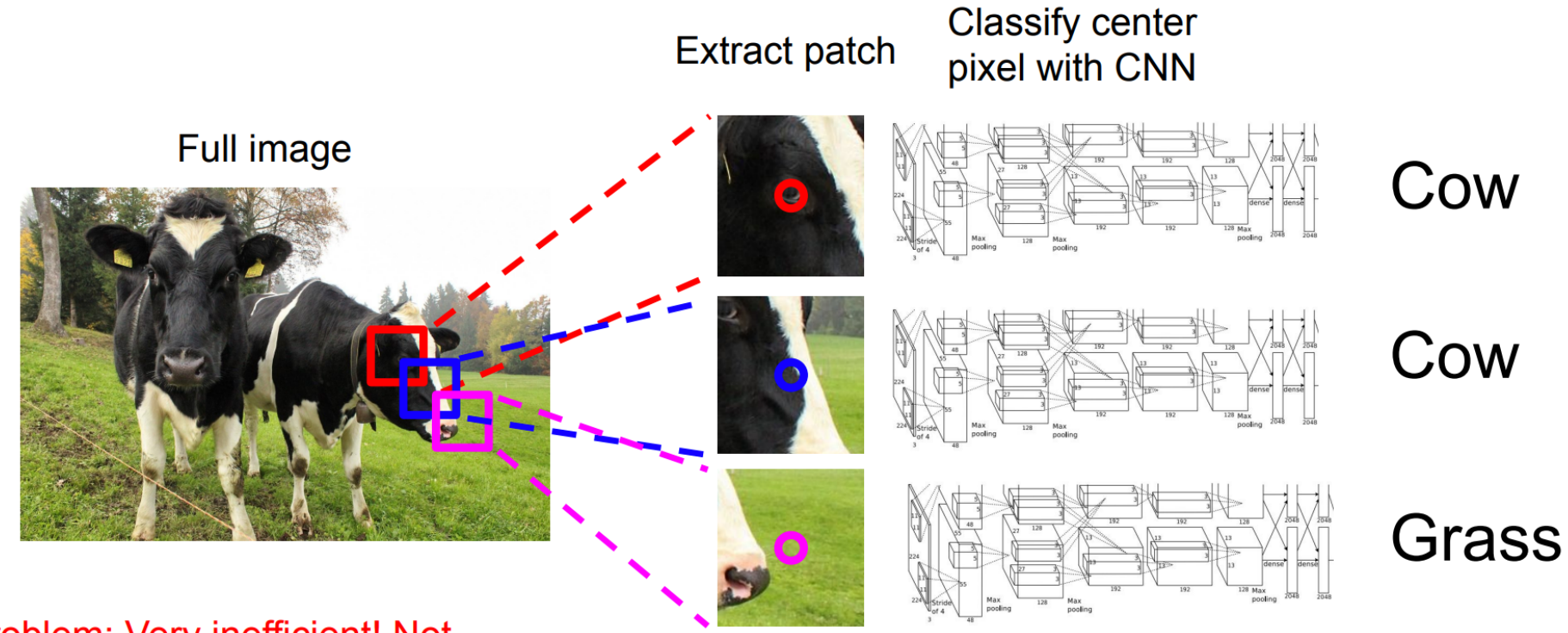
Some image segmentation applications

- OCR

Sliding window approach



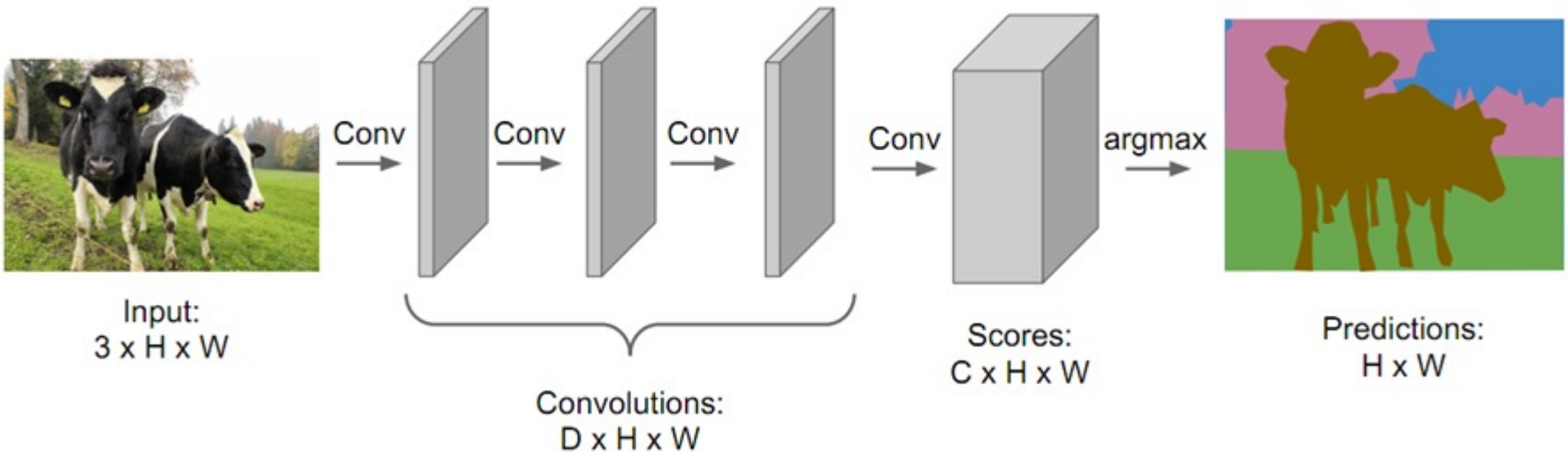
Sliding window approach [2]



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

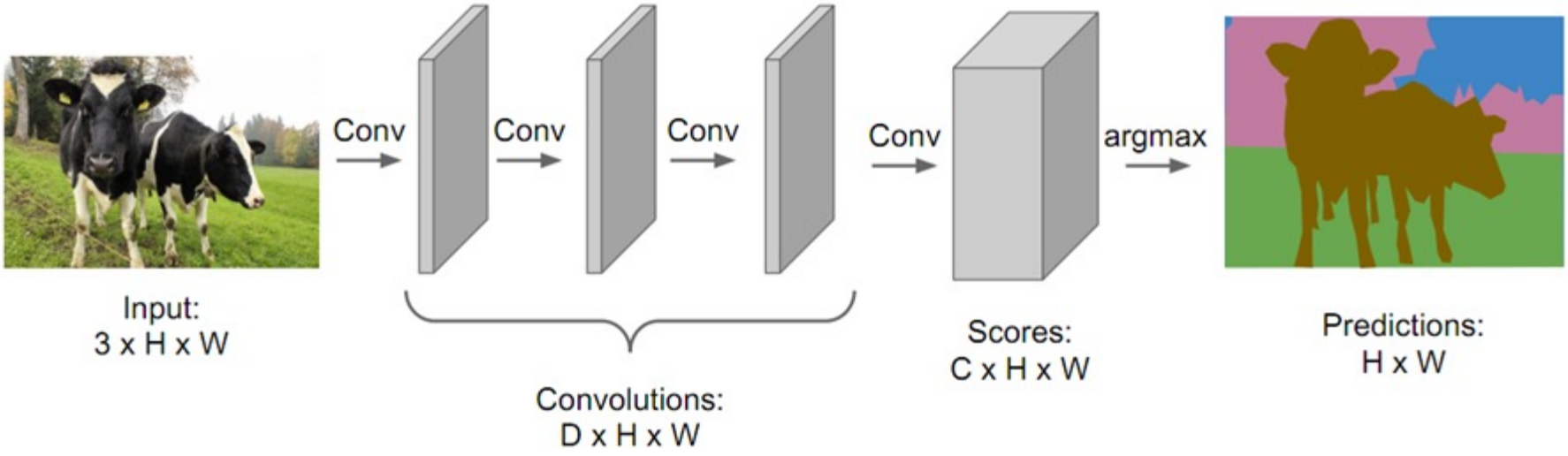
Semantic Segmentation Idea: Convolution

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



Semantic Segmentation Idea: Convolution

- 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: Convolution

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

Input:
3 x H x W

Conv → Conv → Conv → Conv → argmax

Convolutions:
D x H x W

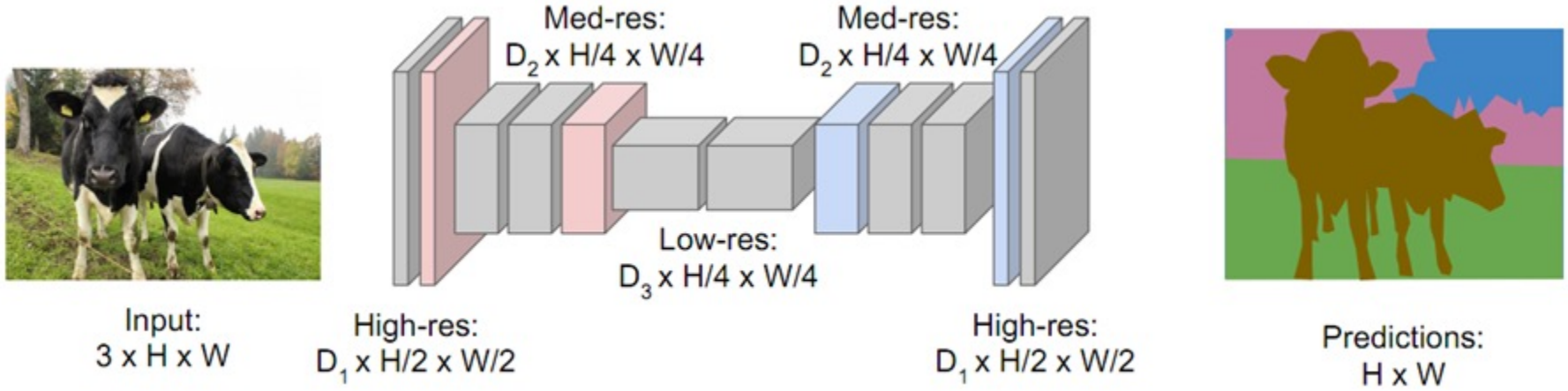
Scores:
C x H x W

Predictions:
H x W

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

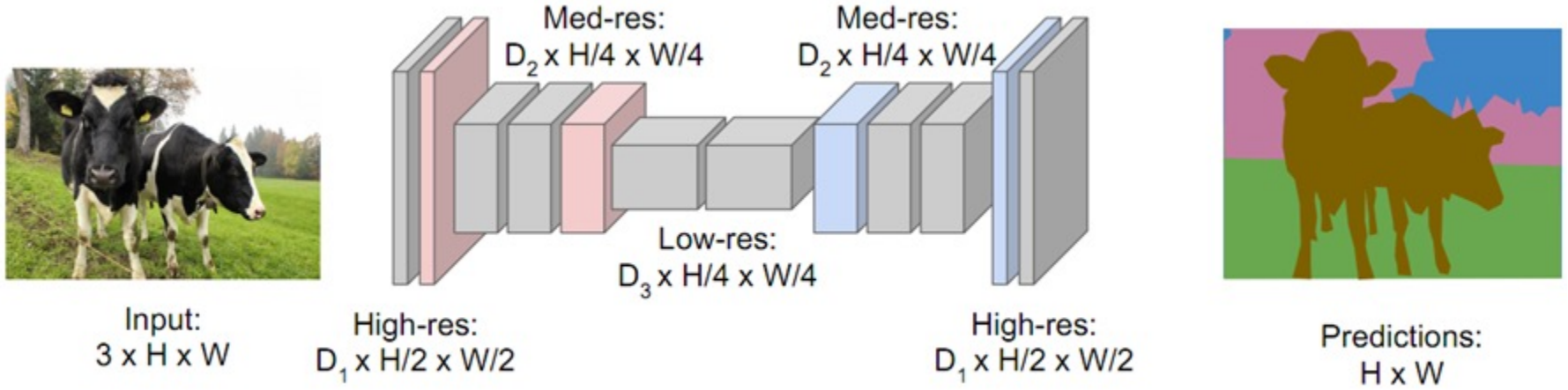
Semantic Segmentation Idea: Convolution

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



Semantic Segmentation Idea: Convolution

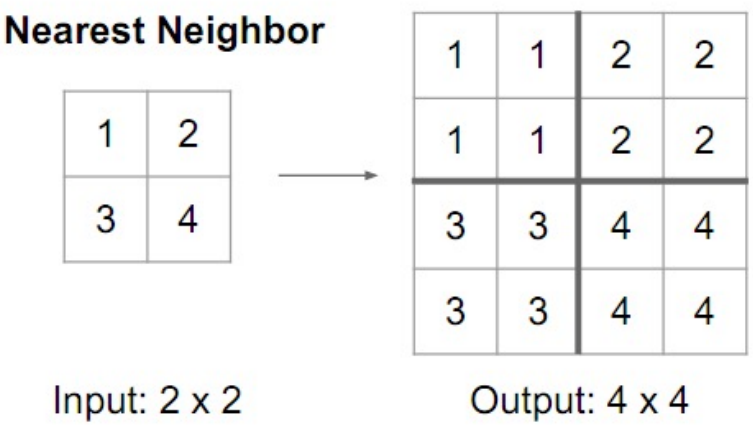
- Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
- Downsampling: Pooling, strided convolution
- Upsampling: ???

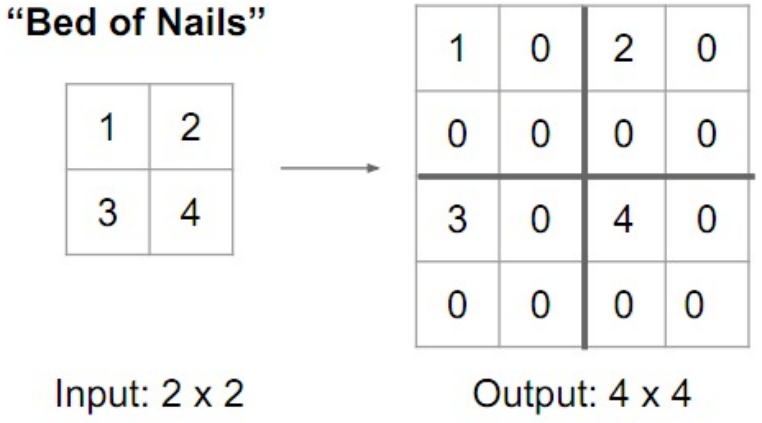


Upsampling layers

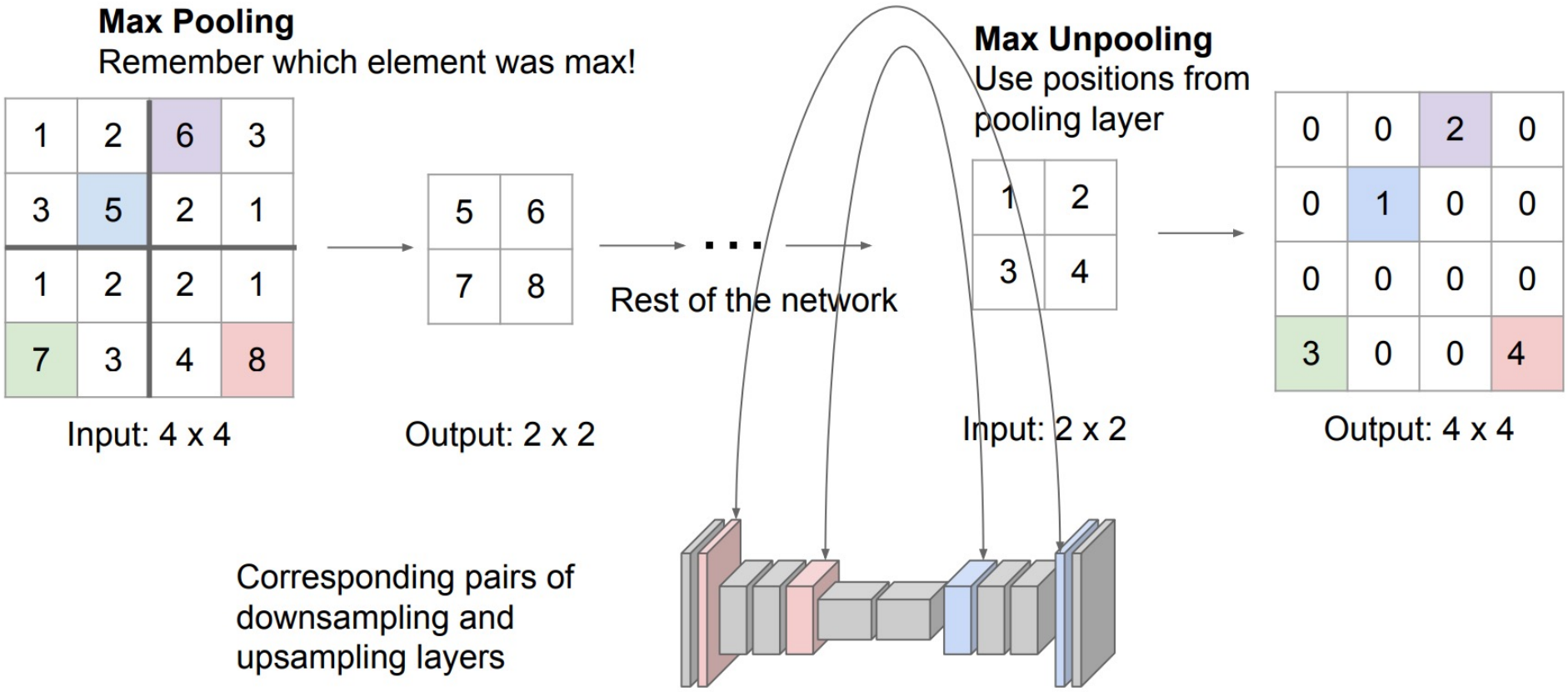
Unpooling

- These layers have no parameters



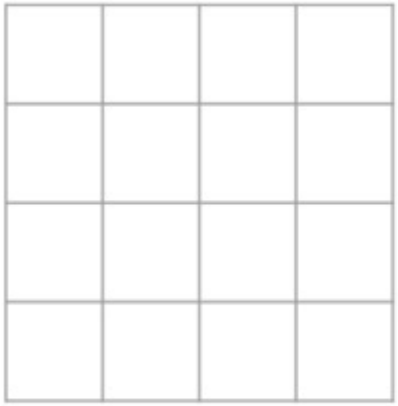


Max Unpooling

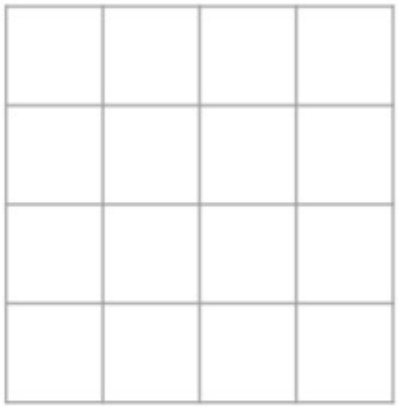


Learnable Upsampling: Transposed Convolution

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

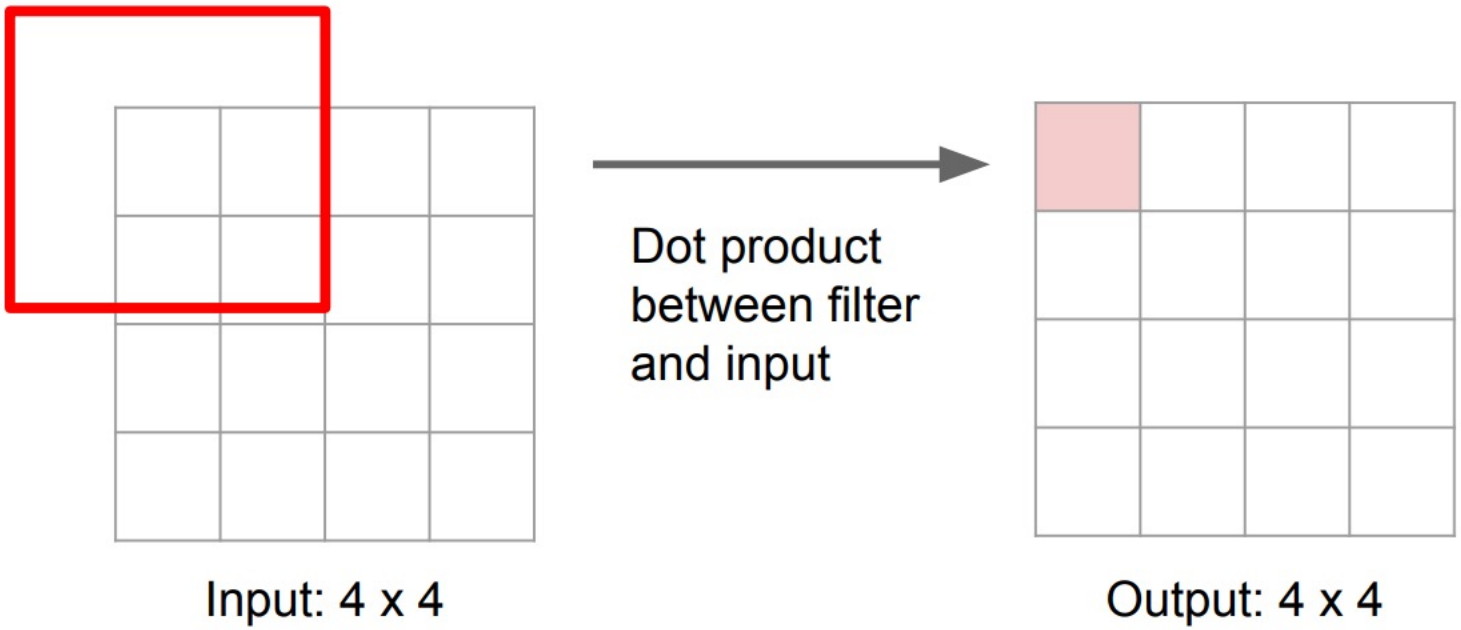


Input: 4 x 4

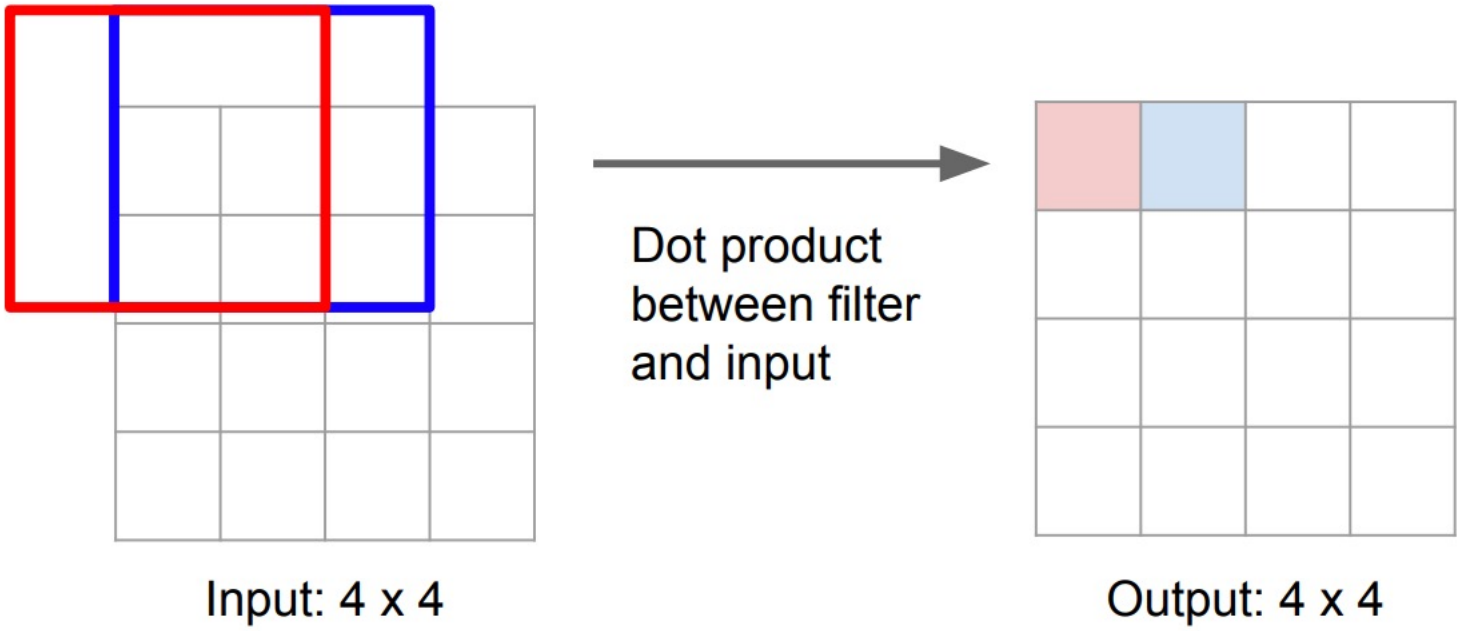


Output: 4 x 4

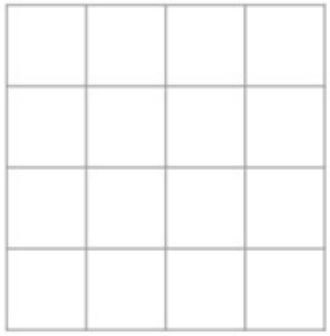
Recall: Normal 3 x 3 convolution, stride 1 pad 1



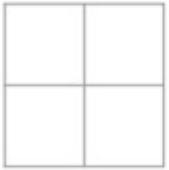
Recall: Normal 3 x 3 convolution, stride 1 pad 1



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

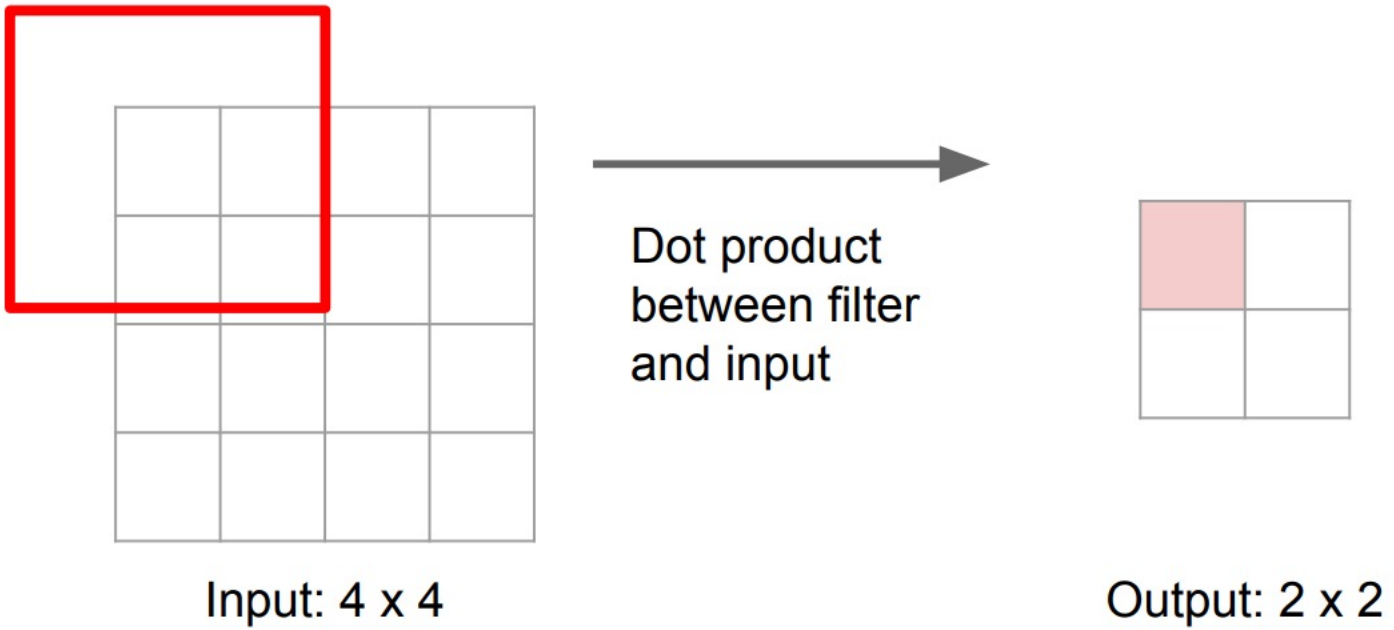


Input: 4 x 4

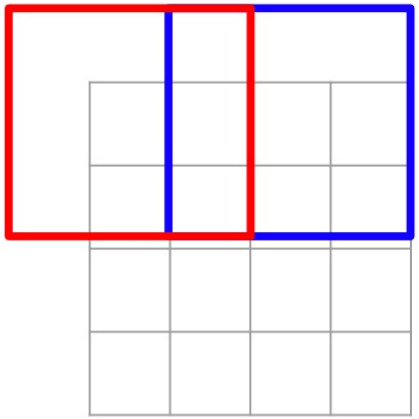


Output: 2 x 2

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

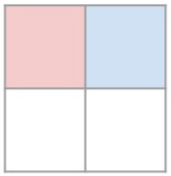


Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

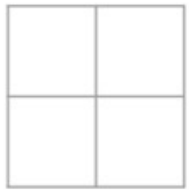
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

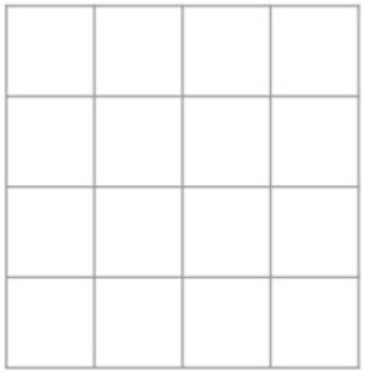
We can interpret strided convolution as “learnable downsampling”.

Learnable Upsampling: Transposed Convolution

- It is an upsampling that contains trainable parameters
- 3 x 3 transpose convolution, stride 2 pad 1



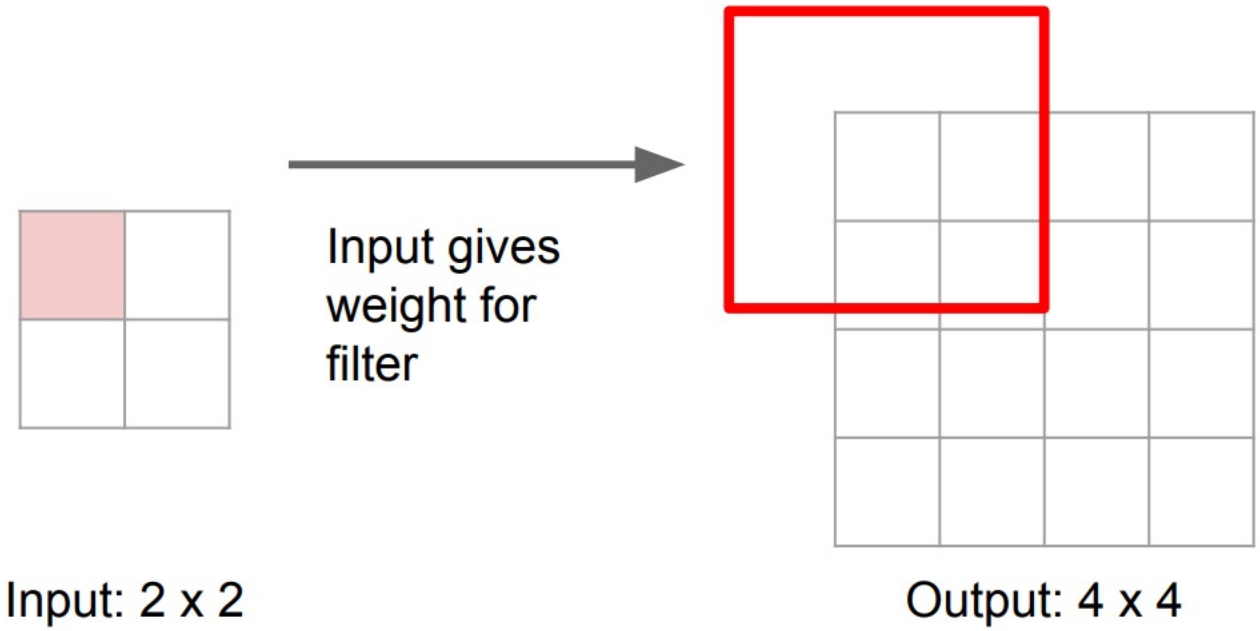
Input: 2 x 2



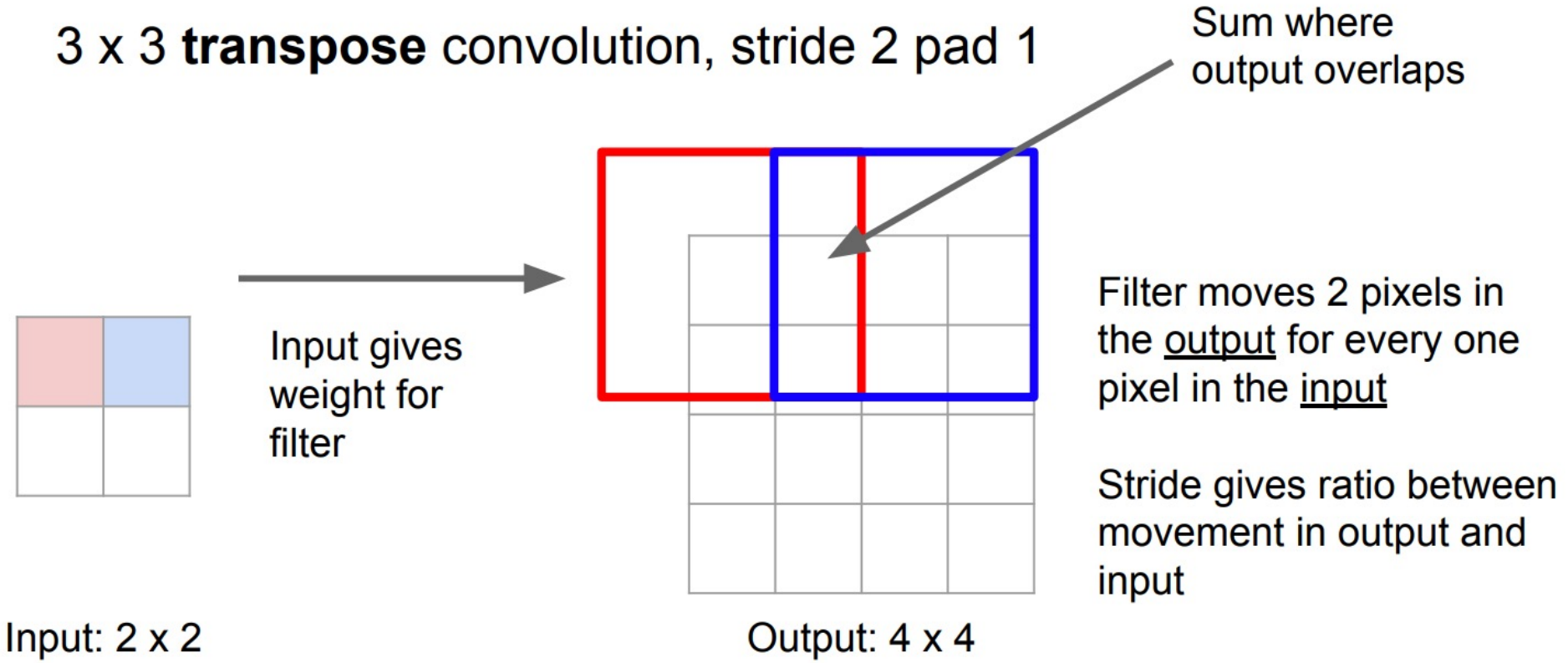
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

3 x 3 transpose convolution, stride 2 pad 1



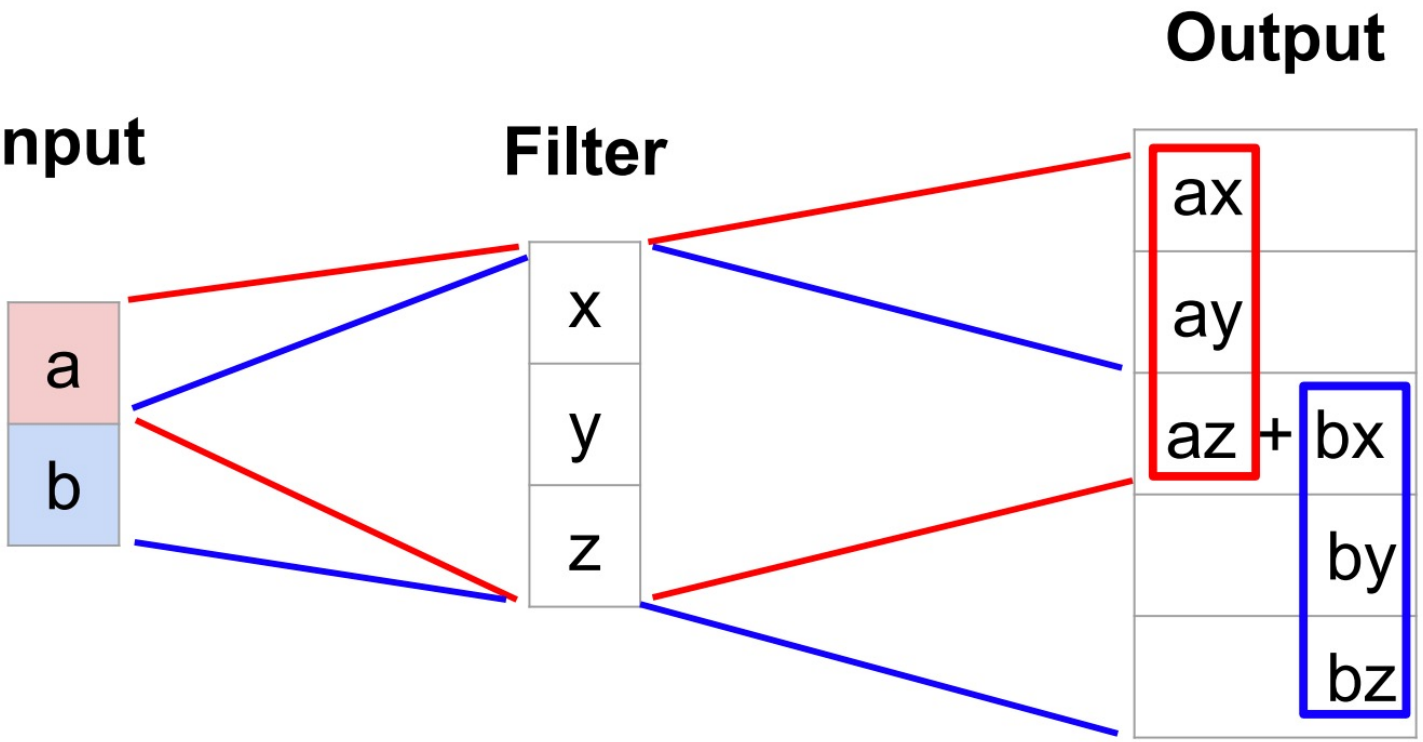
Learnable Upsampling: Transposed Convolution



Learnable Upsampling: Transposed Convolution

- Other names?:
  - Deconvolution (should not, easy to misunderstand)
  - Upconvolution
  - Fractionally strided convolution
  - Backward strided convolution

Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

Example: 1D transpose conv, kernel size=3, stride=2, padding=0

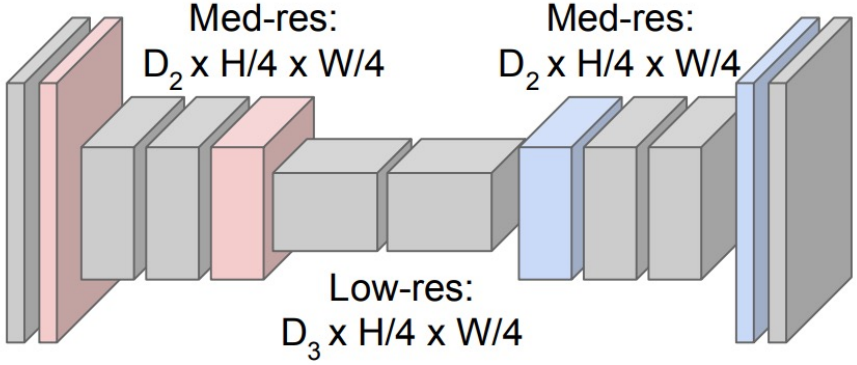
Recap: Semantic Segmentation Idea

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transpose convolution

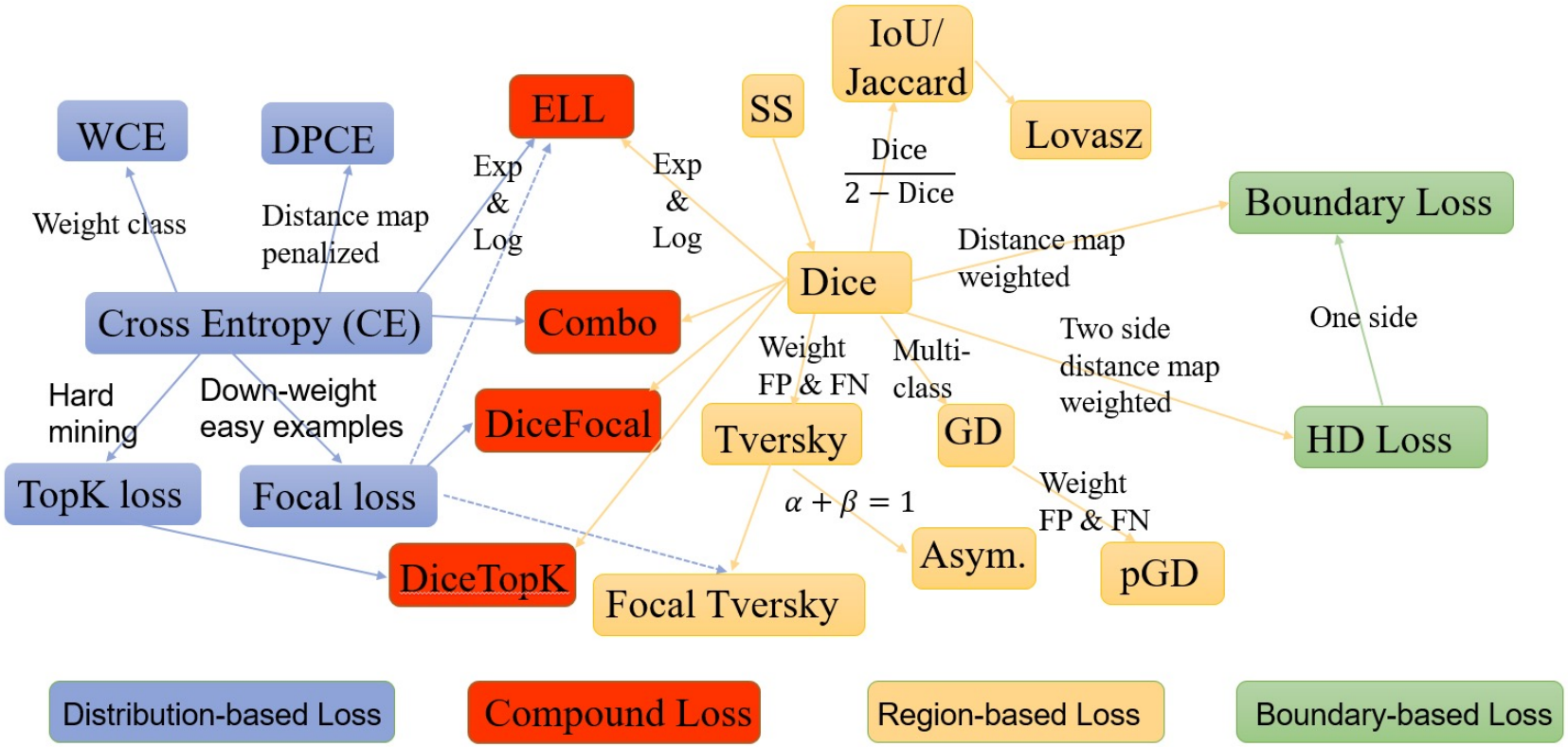
Input:
3 x H x W



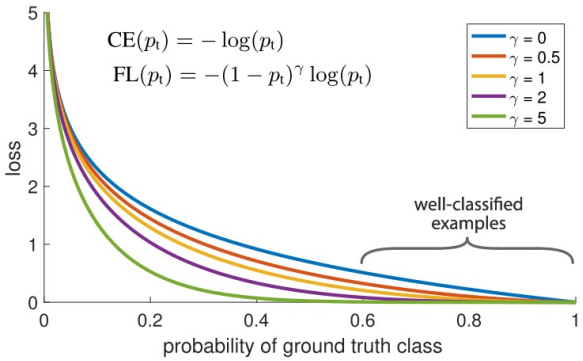
Predictions:
H x W

Objective functions for image segmentation

Objective functions



Distribution-based objective functions



- Cross Entropy (CE):

$$\mathrm{CE}\left(p, \hat{p}\right) = -\left(p\log(\hat{p}) + (1-p)\log(1-\hat{p})\right)$$

- Weighted CE: Each class has a different weight

$$\mathrm{WCE}\left(p, \hat{p}\right) = -\left(\beta p\log(\hat{p}) + (1-p)\log(1-\hat{p})\right)$$

- Focal loss: solve the problem of large imbalance between the background layer and the object layer of interest. The objective function value for the easy to classify objects is reduced so that the network focuses more on difficult objects.

$$\mathrm{FL}\left(p, \hat{p}\right) = -\left(\alpha(1-\hat{p})^{\gamma}p\log(\hat{p}) + (1-\alpha)\hat{p}^{\gamma}(1-p)\log(1-\hat{p})\right)$$

Area-based objective functions

- Dice coefficient and IoU:

$$\mathrm{DC} = \frac{2TP}{2TP + FP + FN} = \frac{2|X \cap Y|}{|X| + |Y|}$$

$$\mathrm{IoU} = \frac{TP}{TP + FP + FN} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

- Dice loss: $$\mathrm{DL}\left(p, \hat{p}\right) = 1 - \frac{2p\hat{p} + 1}{p + \hat{p} + 1}$$
- Tversky loss:

$$\mathrm{TI}\left(p, \hat{p}\right) = \frac{p\hat{p}}{p\hat{p} + \beta(1-p)\hat{p} + (1-\beta)p(1-\hat{p})}$$

Combined objective functions

- Dice loss + CE:

$$\mathrm{CE}\left(p, \hat{p}\right) + \mathrm{DL}\left(p, \hat{p}\right)$$

- Dice loss + Focal loss

$$\mathrm{CE}\left(p, \hat{p}\right) + \mathrm{FL}\left(p, \hat{p}\right)$$

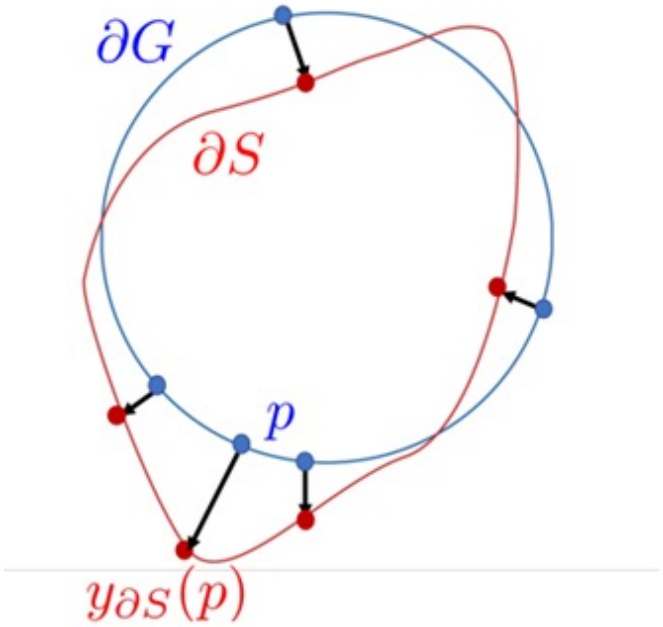
- …

Boundary loss

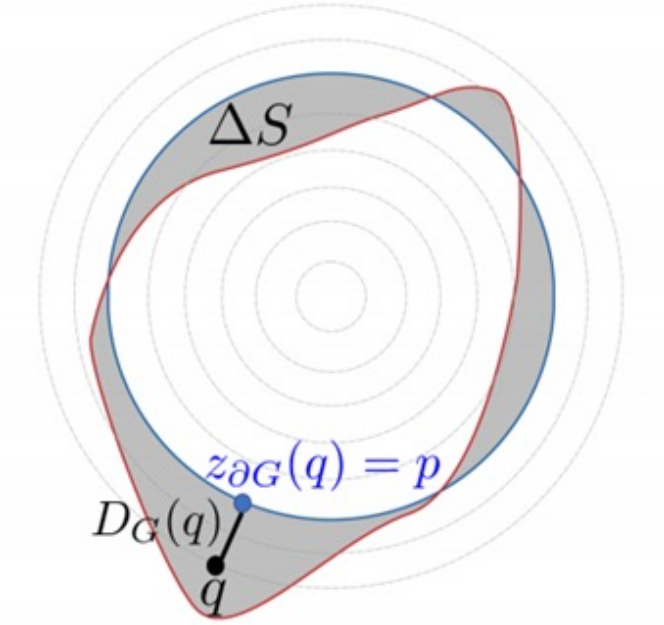
distance metric on the space of contours, not regions

$$\text{Dist}(\partial G, \partial S) = \int_{\partial G} \|y_{\partial S}(p) - p\|^2 dp$$

$$\text{Dist}(\partial G, \partial S) = 2 \int_{\Delta S} D_G(q) dq$$



(a) Differential

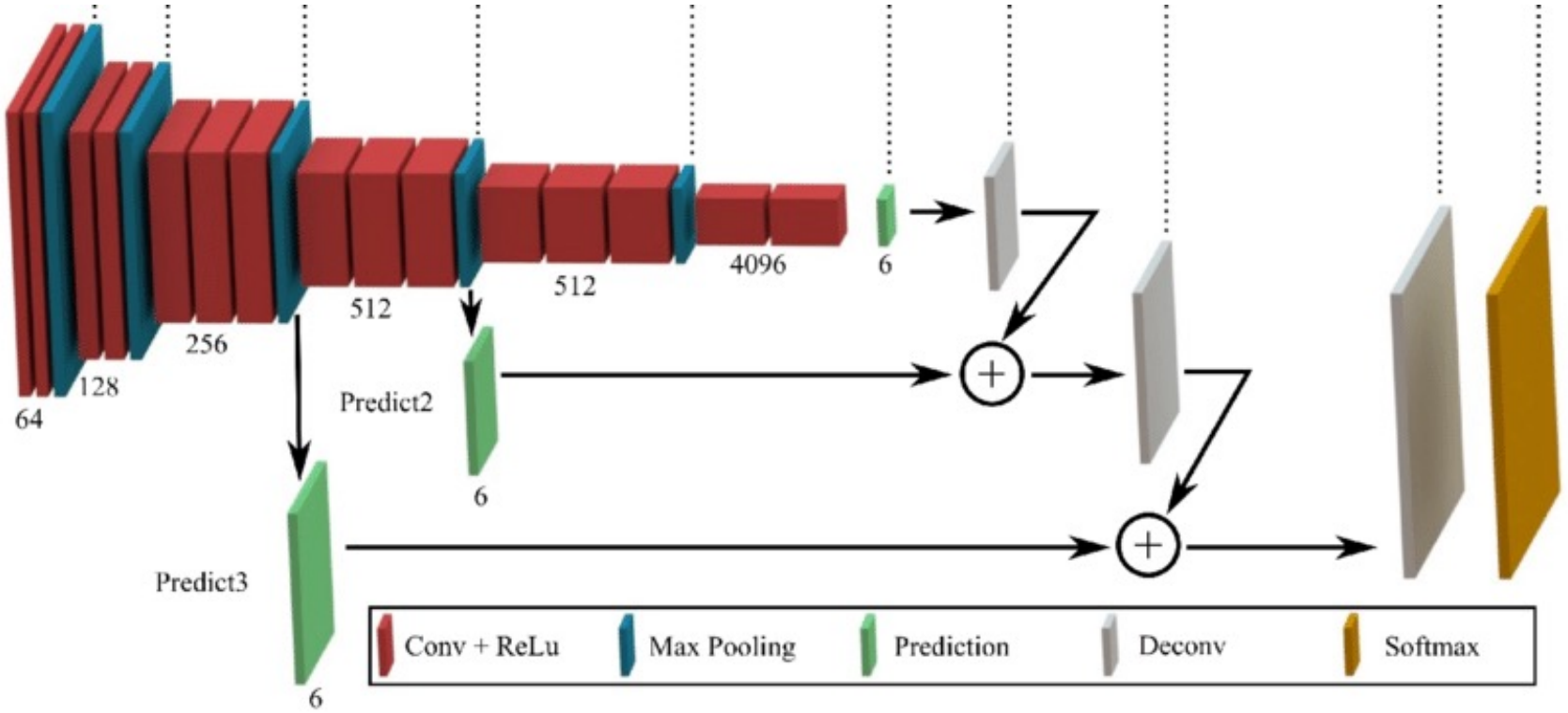


(b) Integral

$$\frac{1}{2}\text{Dist}(\partial G, \partial S) = \int_\Omega \phi_G(q) s(q) dq - \int_\Omega \phi_G(q) g(q) dq$$

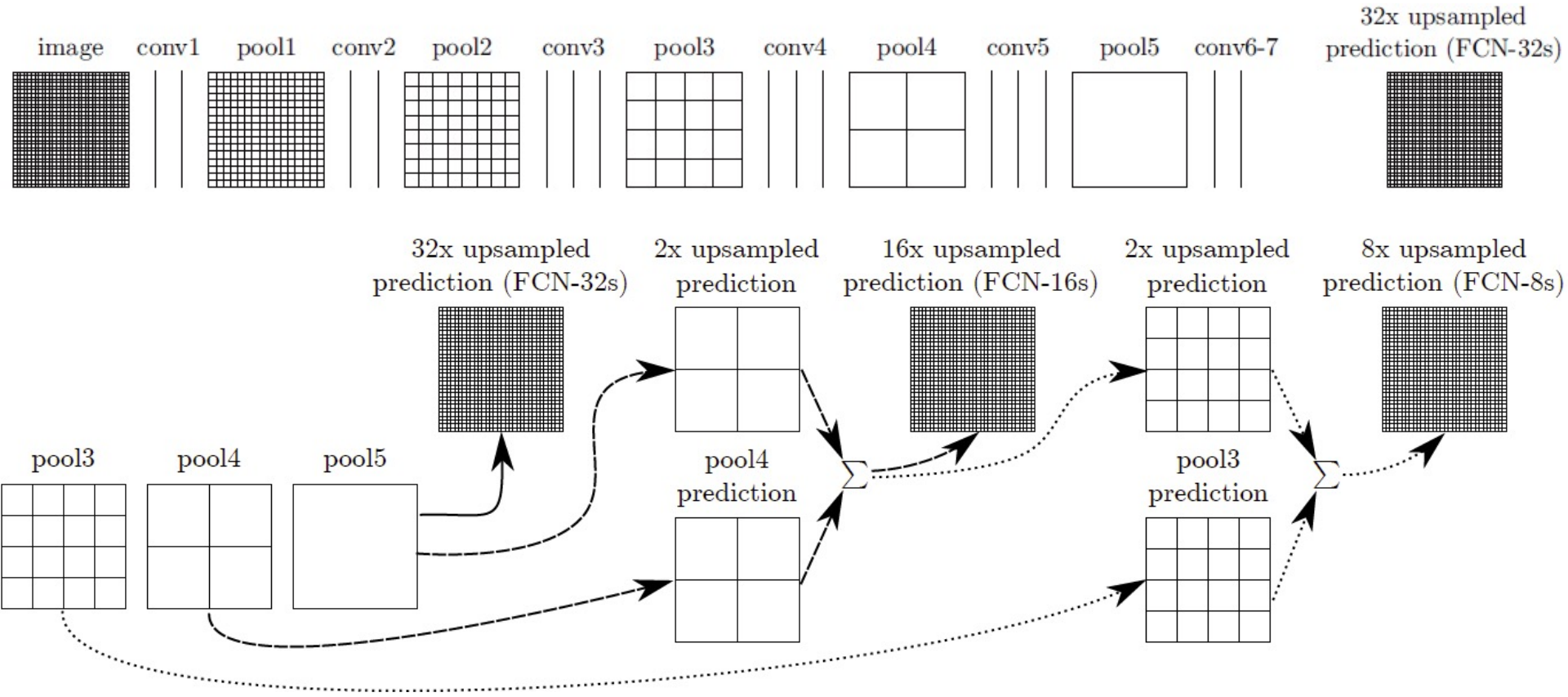
Some typical image segmentation networks

FCN with 2 shortcuts

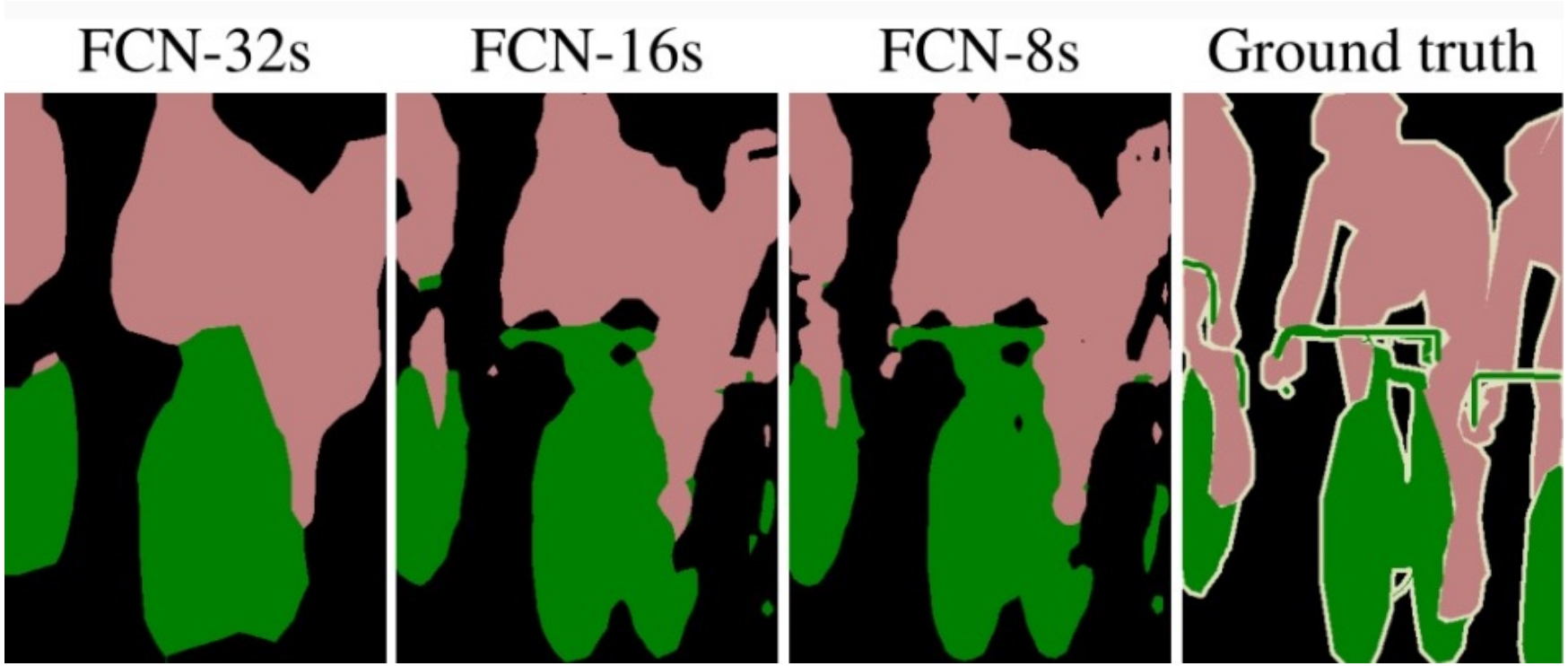


FCN with 2 shortcuts (2)

- Illustrate FCN results with different levels of resolution

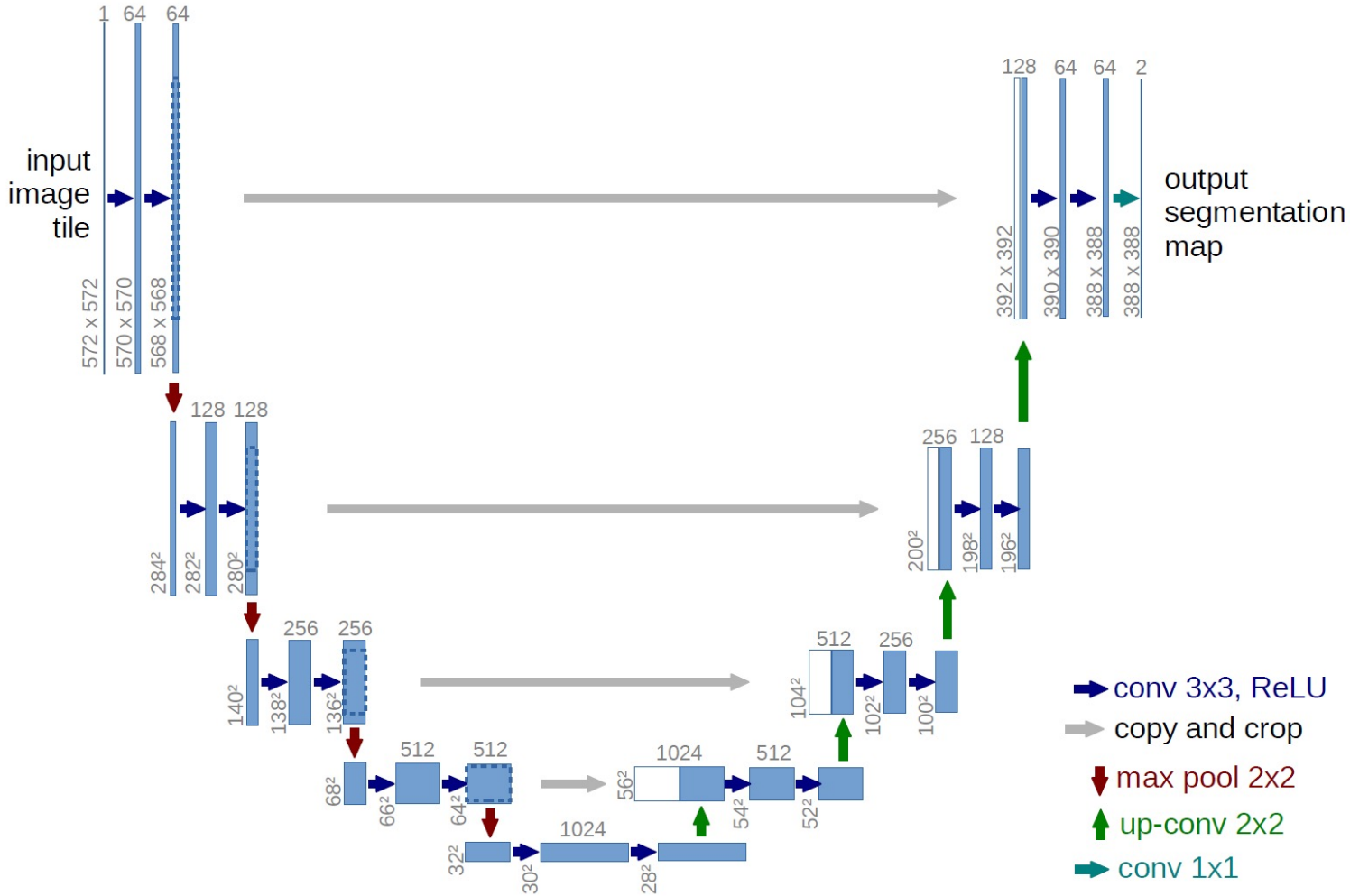


Results

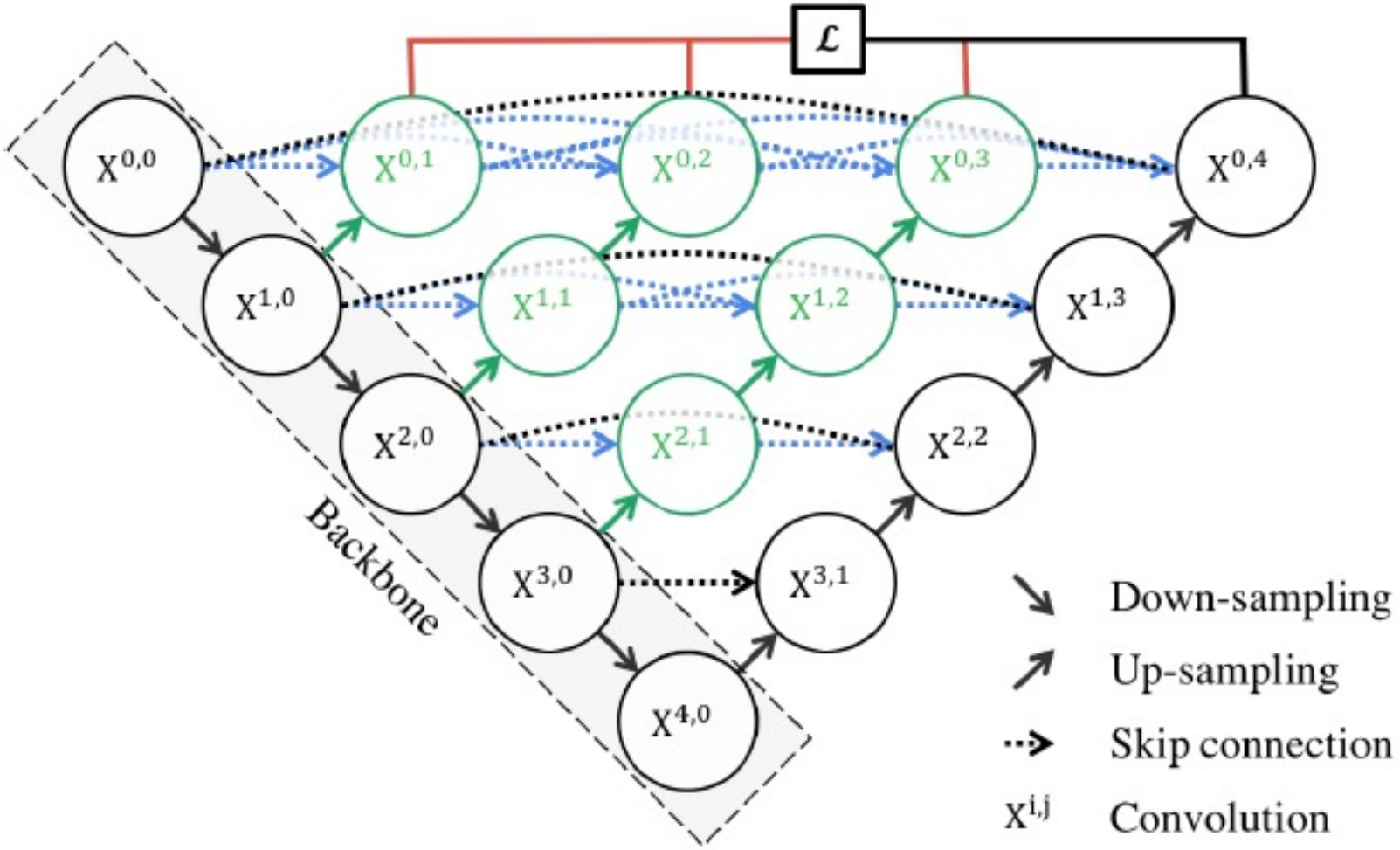


U-Net

- Widely used in medical images

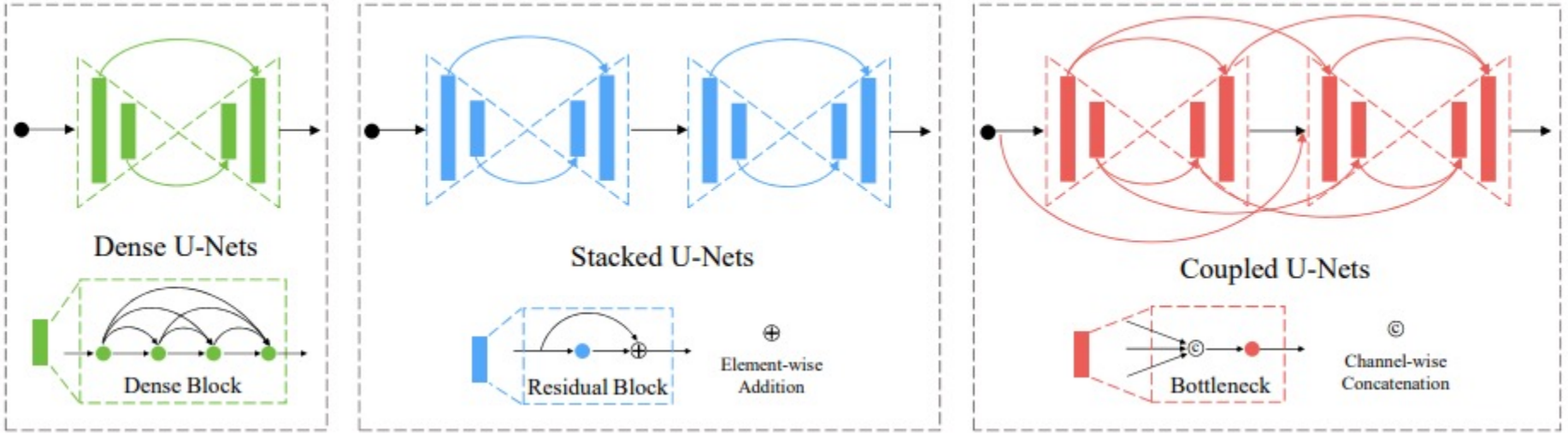


U-Net++


$\mathcal{L}$
$X^{0,0}$
$X^{0,1}$
$X^{0,2}$
$X^{0,3}$
$X^{0,4}$
$X^{1,0}$
$X^{1,1}$
$X^{1,2}$
$X^{1,3}$
$X^{2,0}$
$X^{2,1}$
$X^{2,2}$
$X^{3,0}$
$X^{3,1}$
$X^{4,0}$
Backbone
Down-sampling
Up-sampling
Skip connection
$X^{i,j}$ Convolution

Stacked UNets and CUNets

- Stacked UNets: stack many UNets in series
- CUNets: also stack many UNets in series, but there are additional short connections between UNets



References

1. cs231n:

http://cs231n.stanford.edu

2. Losses for segmentation

https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!