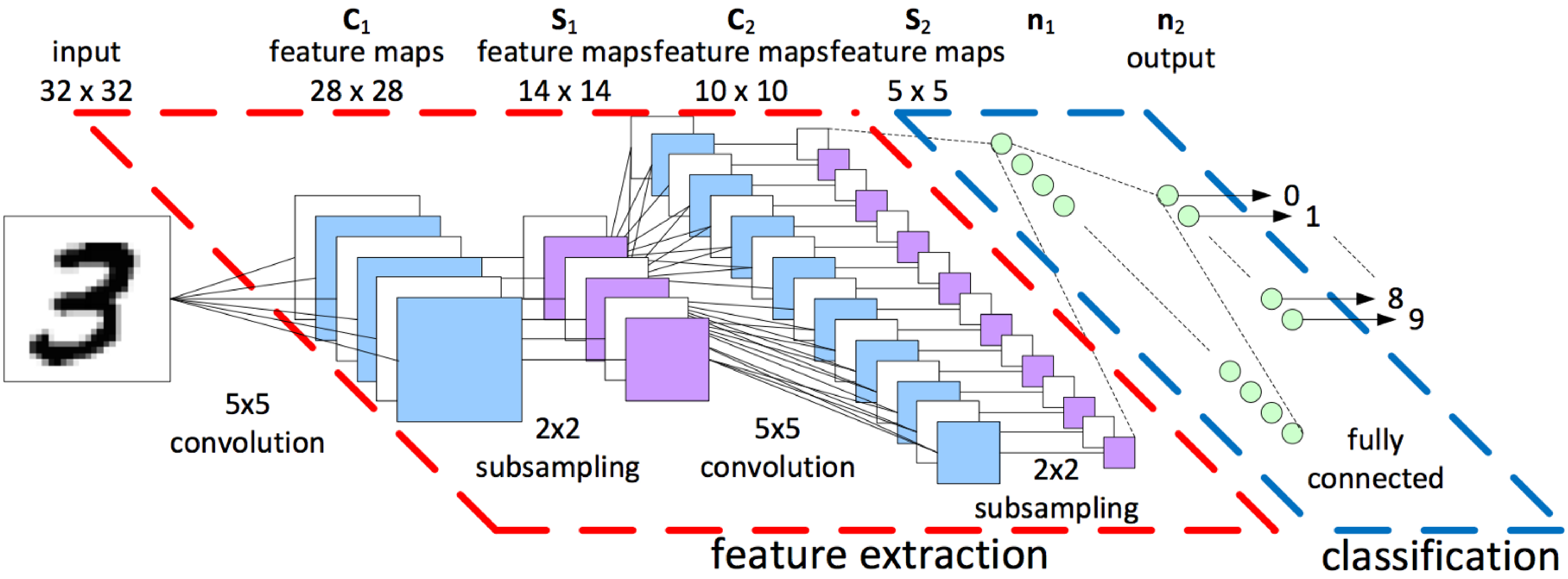
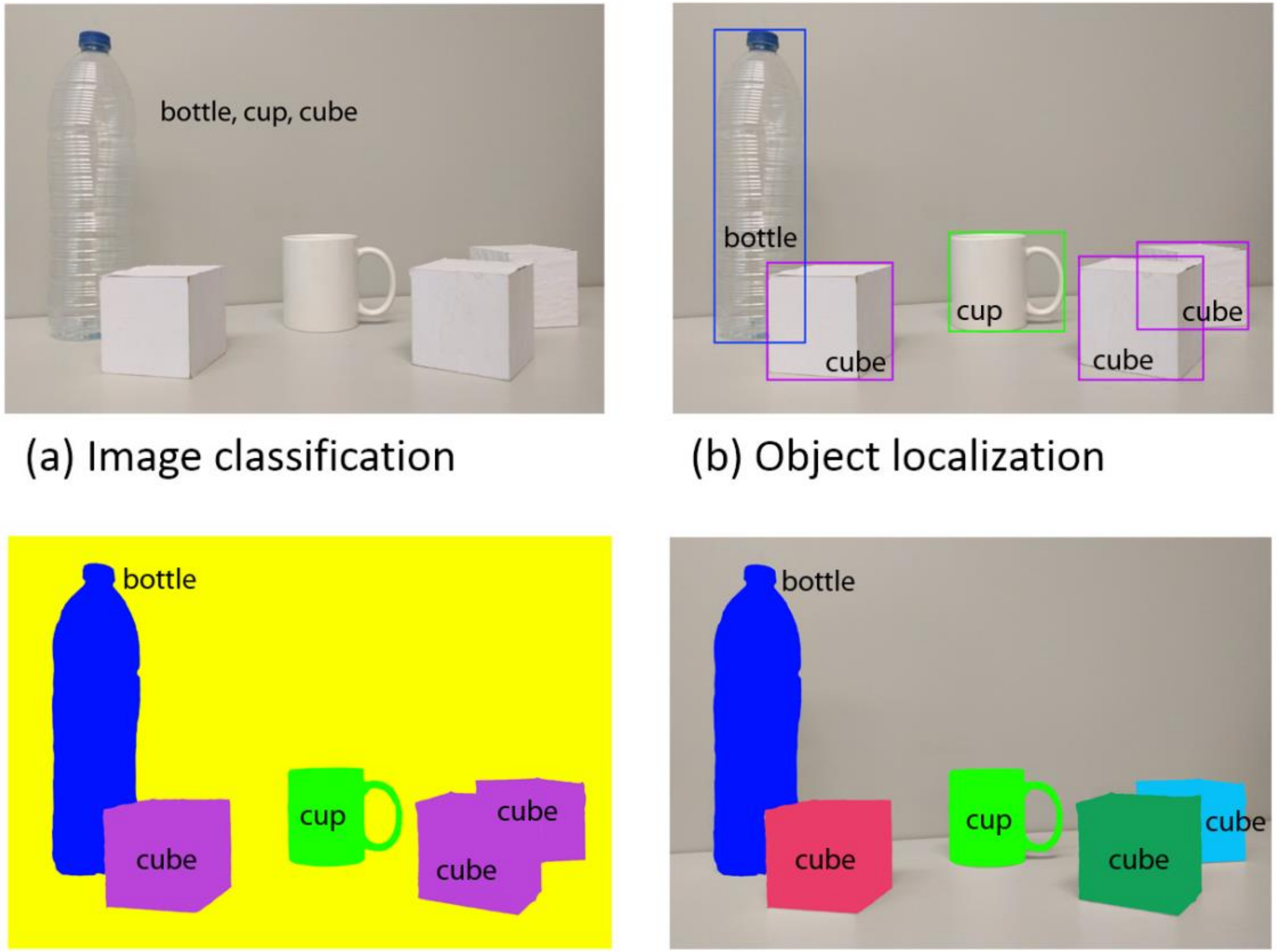
Introduction to Image Segmentation using Deep Learning

So far image classification



Other Computer Vision Tasks



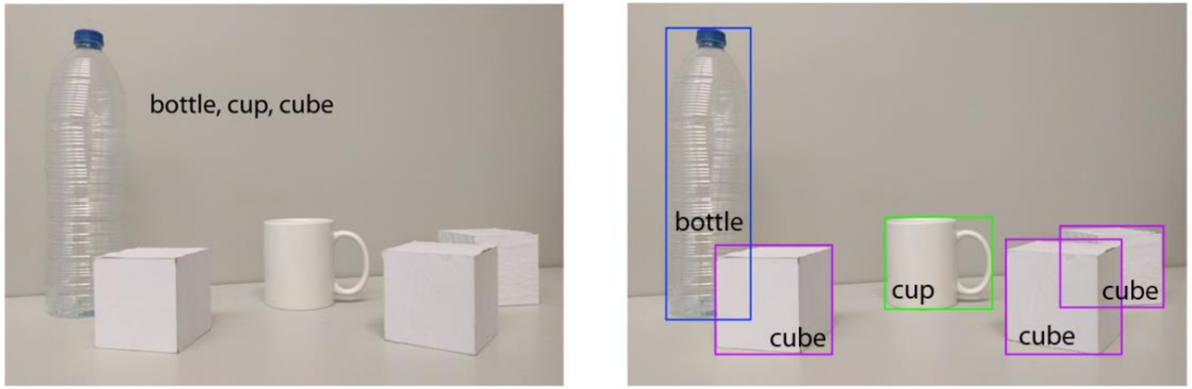
(a) Image classification (b) Object localization

(c) Semantic segmentation (d) Instance segmentation

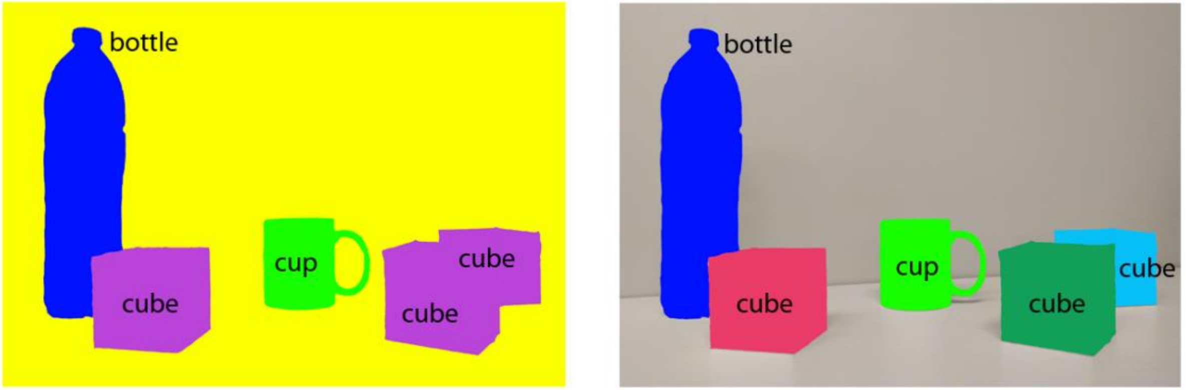
A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, *"A review on deep learning techniques applied to semantic segmentation,"*

Other Computer Vision Tasks

- **Object Localization or object detection** is to identify all the class relevant objects in an image
- **Segmentation** is a partition of an image into several parts by giving pixel label, but without any attempt at understanding what these parts represent.
- **Semantic segmentation** attempts to partition the image into semantically meaningful parts, and to classify each part into one of the pre-determined classes. But, still can't differ among objects/instances in the same class
- **Instance segmentation** can differ objects/instances in the same class while performing the semantic segmentation



(a) Image classification (b) Object localization

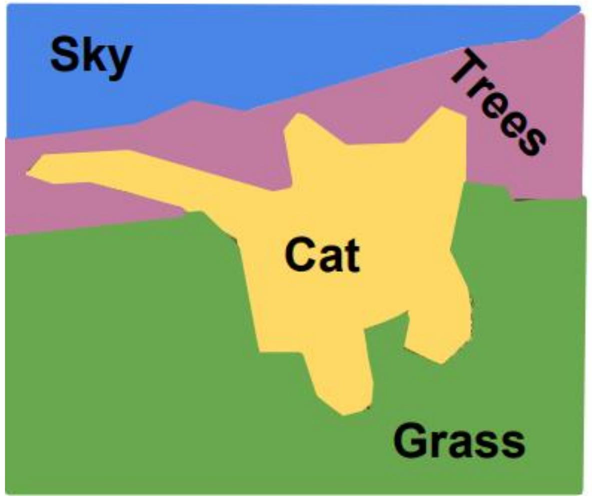


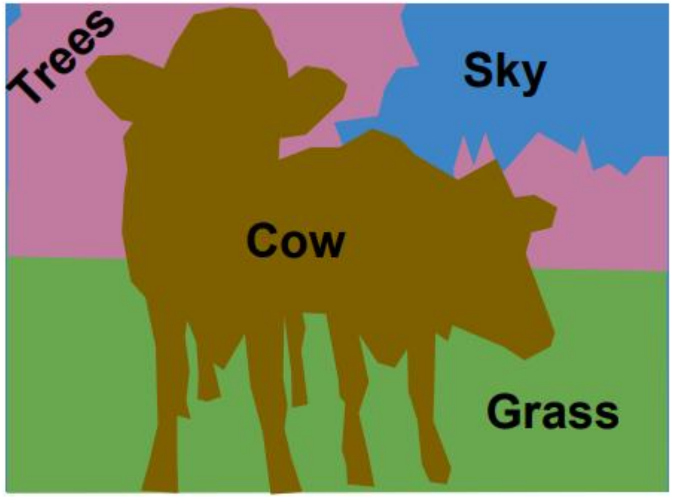
(c) Semantic segmentation (d) Instance segmentation

A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, *"A review on deep learning techniques applied to semantic segmentation,"*

Semantic Segmentation

- Purely supervise learning
- Label each pixel in the image with a category label
- But do not differentiate instances, only care about pixels
- Dataset preparation is very expensive
- Online-tools are there for manual labeling
  https://en.wikipedia.org/wiki/List_of_manual_image_annotation_tools
- Training could be slower than image classification

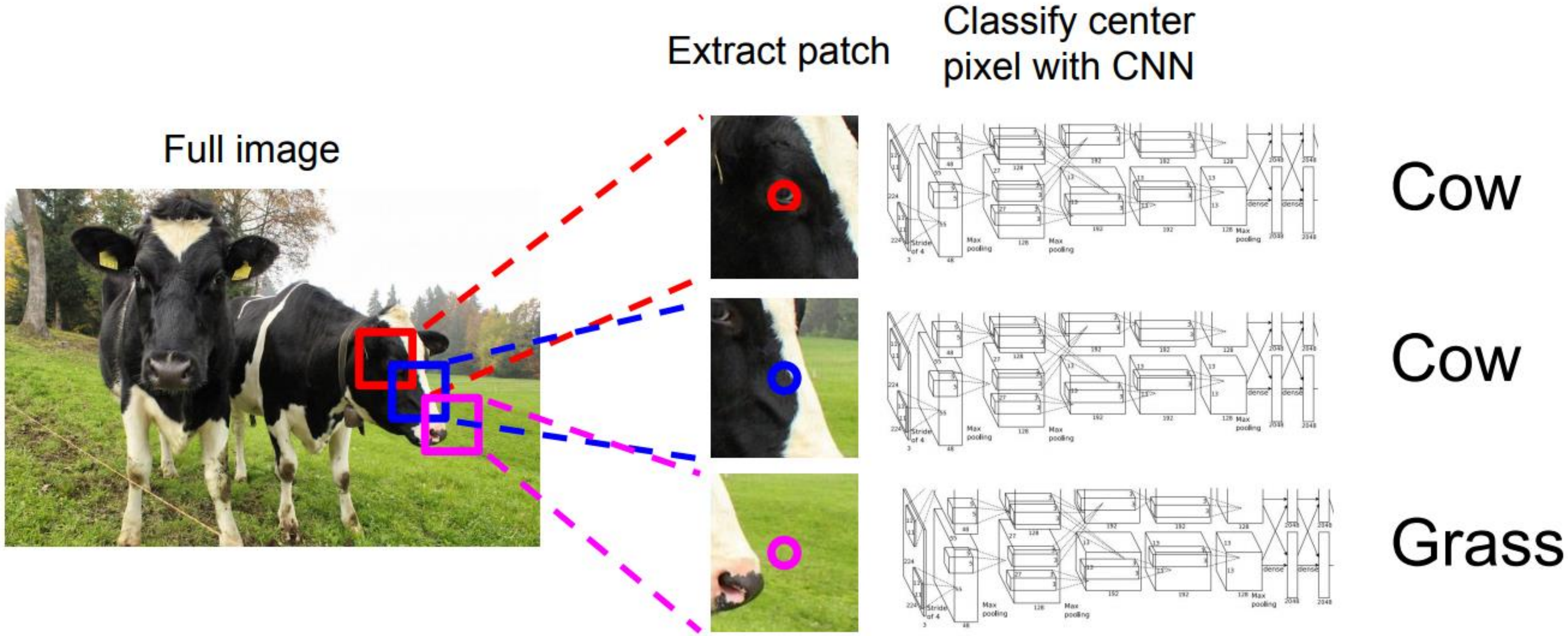




Semantic Segmentation methods

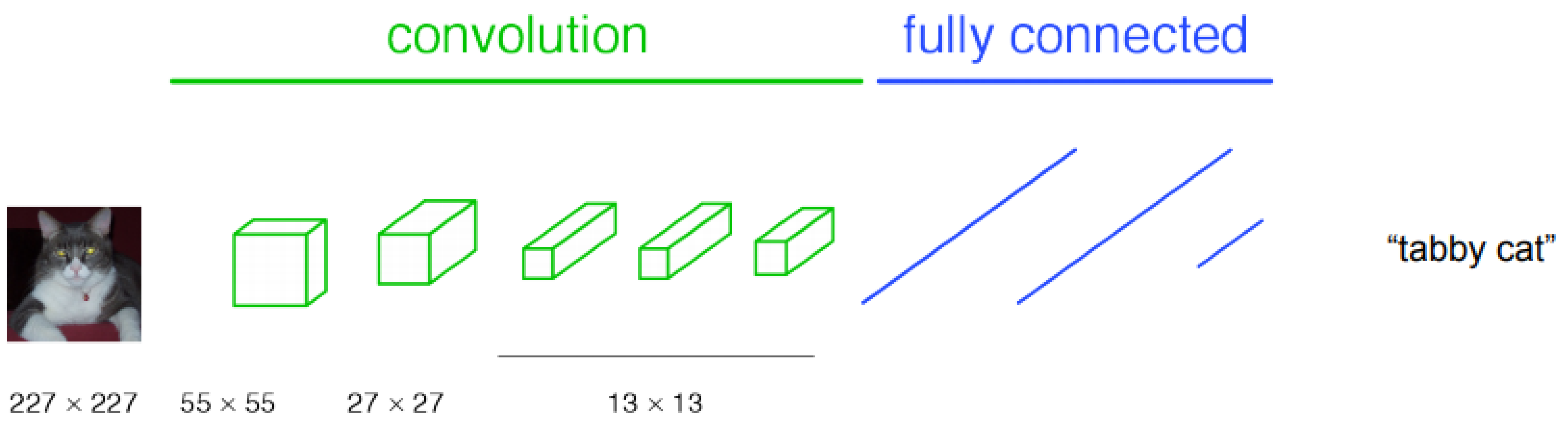
- Sliding Window approach
  - Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
  - Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014
- Fully convolutional network approach
  - Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
  - Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Semantic Segmentation: sliding window

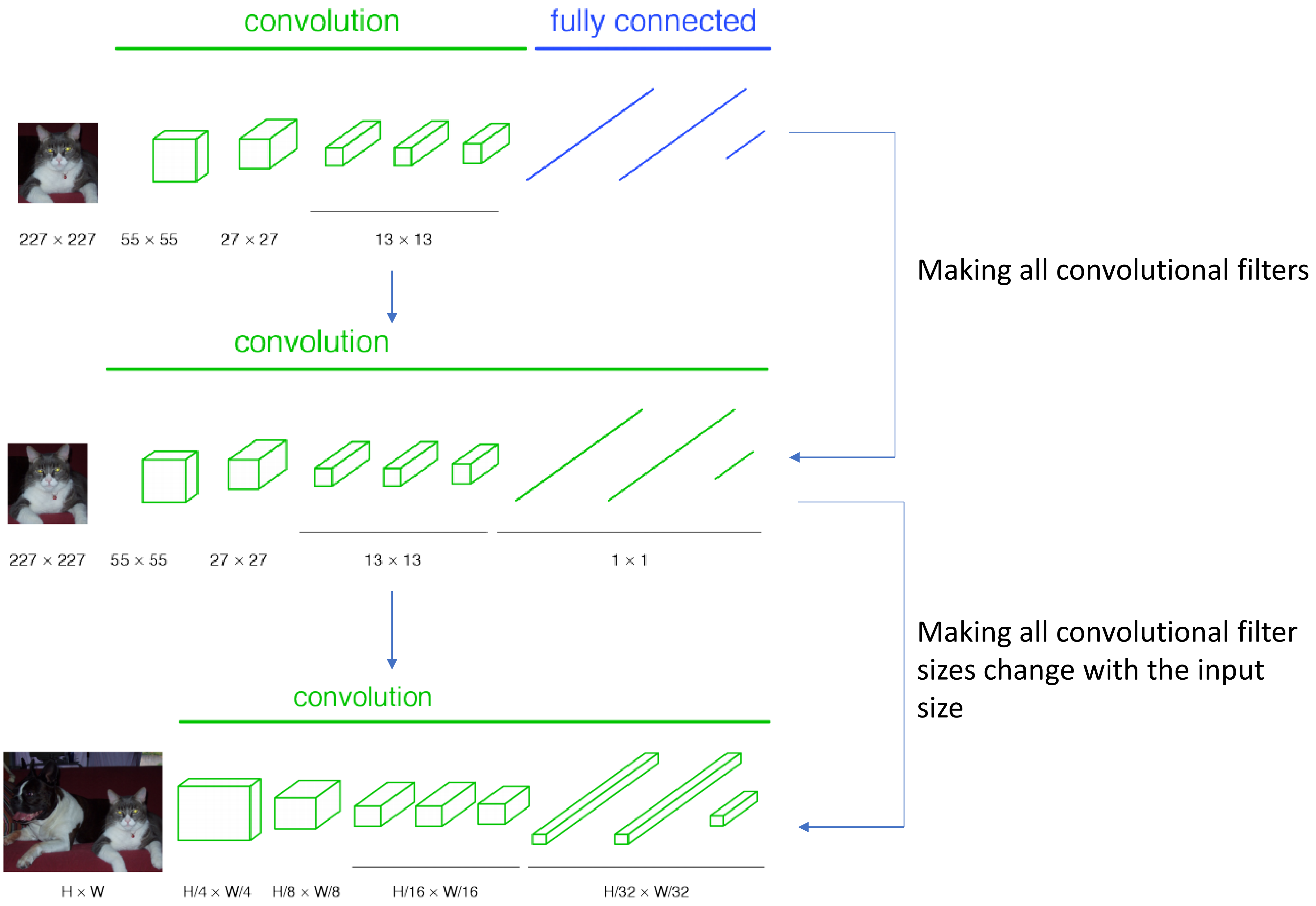


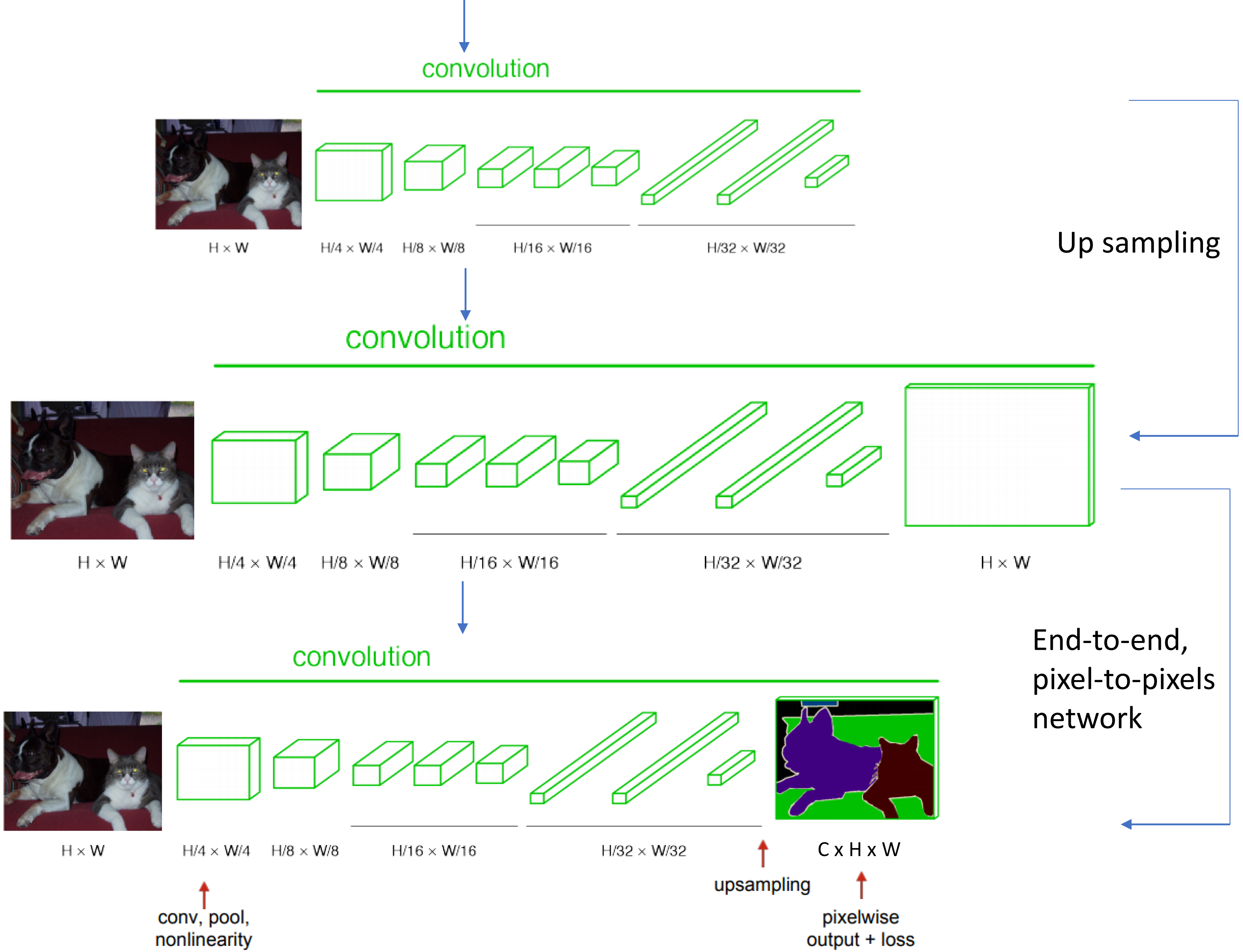
- Predict the middle pixel of the window while sliding it throughout the image
- But this method is very inefficient and less accurate
- Since method using a sliding window, for a single image there will be huge number of forward passes and backward passes.
- Another reason is that its not reusing shared features between overlapping patches

Semantic Segmentation: Fully Convolutional Networks



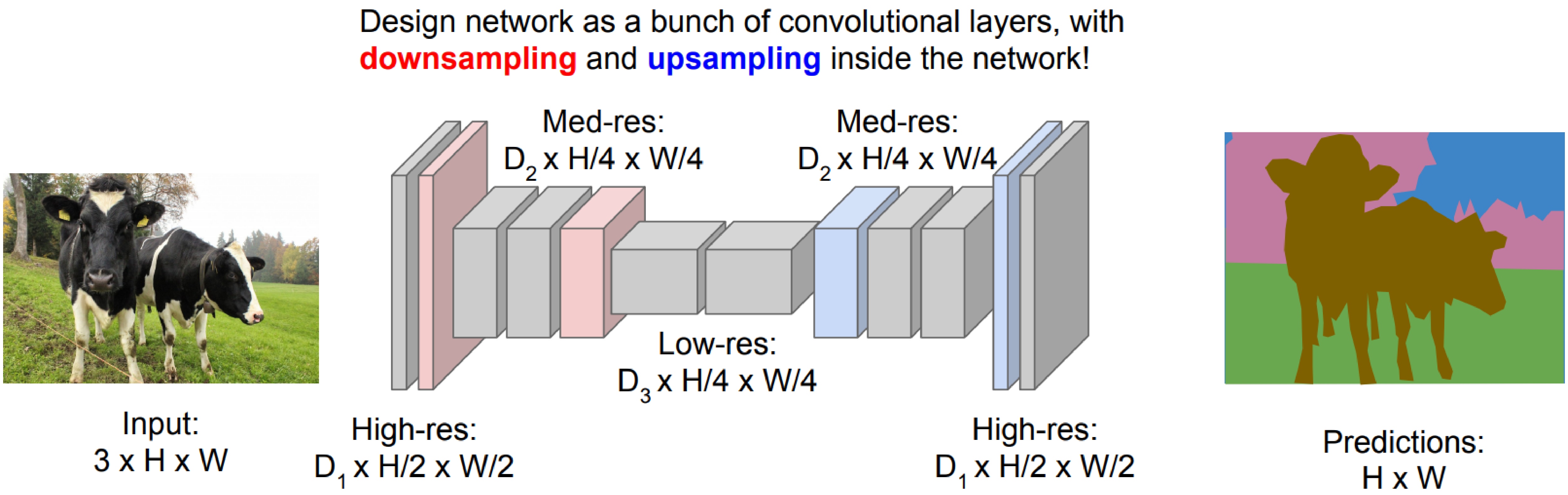
- Normal classification convolutional networks have a fully connected network for classification
- Since the fully connected networks have a fixed size, input image need to be resize into a fix size input
- But when the image is resize, we loss some pixels or generate new pixels which can cause accuracy problems in the segmentation
- There for the solution is to remove the fully connected part


convolution
fully connected
227 × 227
55 × 55
27 × 27
13 × 13
convolution
227 × 227
55 × 55
27 × 27
13 × 13
1 × 1
convolution
H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
Making all convolutional filters
Making all convolutional filter sizes change with the input size



convolution
H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
Up sampling
convolution
H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
H × W
convolution
H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
C x H x W
upsampling
conv, pool,
nonlinearity
pixelwise
output + loss
End-to-end,
pixel-to-pixels
network

Designing a Fully Convolutional Networks

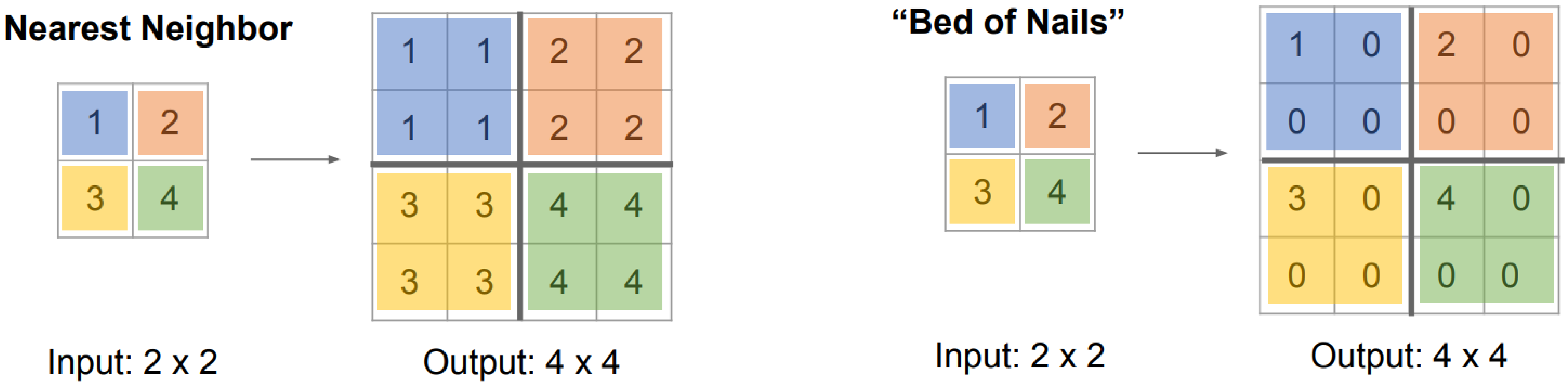


- Down sample the high resolution image to increase the computational efficiency
  - Strided convolution
  - Polling (max, average)
- Up sample the convolutional filters up to image resolution to compare it with the image size segmentation label.

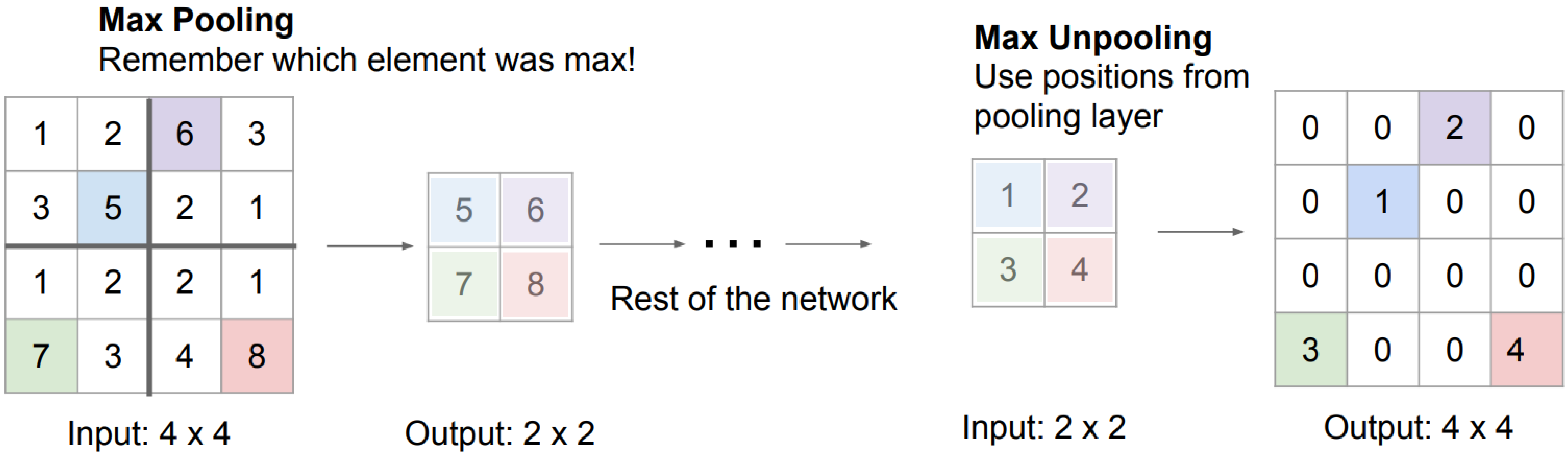
Upsampling

- In-Network upsampling: “Unpooling”
  - Nearest Neighbor
  - Bed of Nails
  - Max Unpooling
- Learnable Upsampling: Transpose Convolution

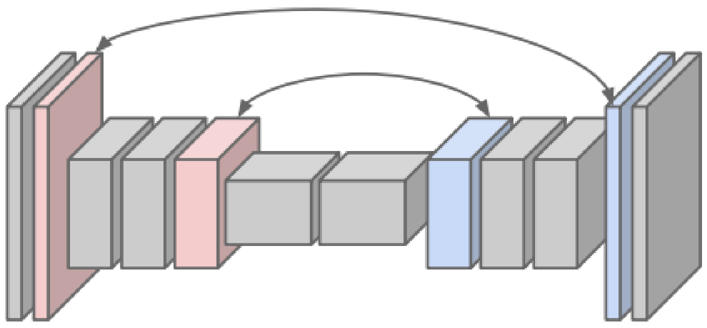
In-Network upsampling: “Unpooling”



In-Network upsampling: “Max Unpooling”



Corresponding pairs of downsampling and upsampling layers



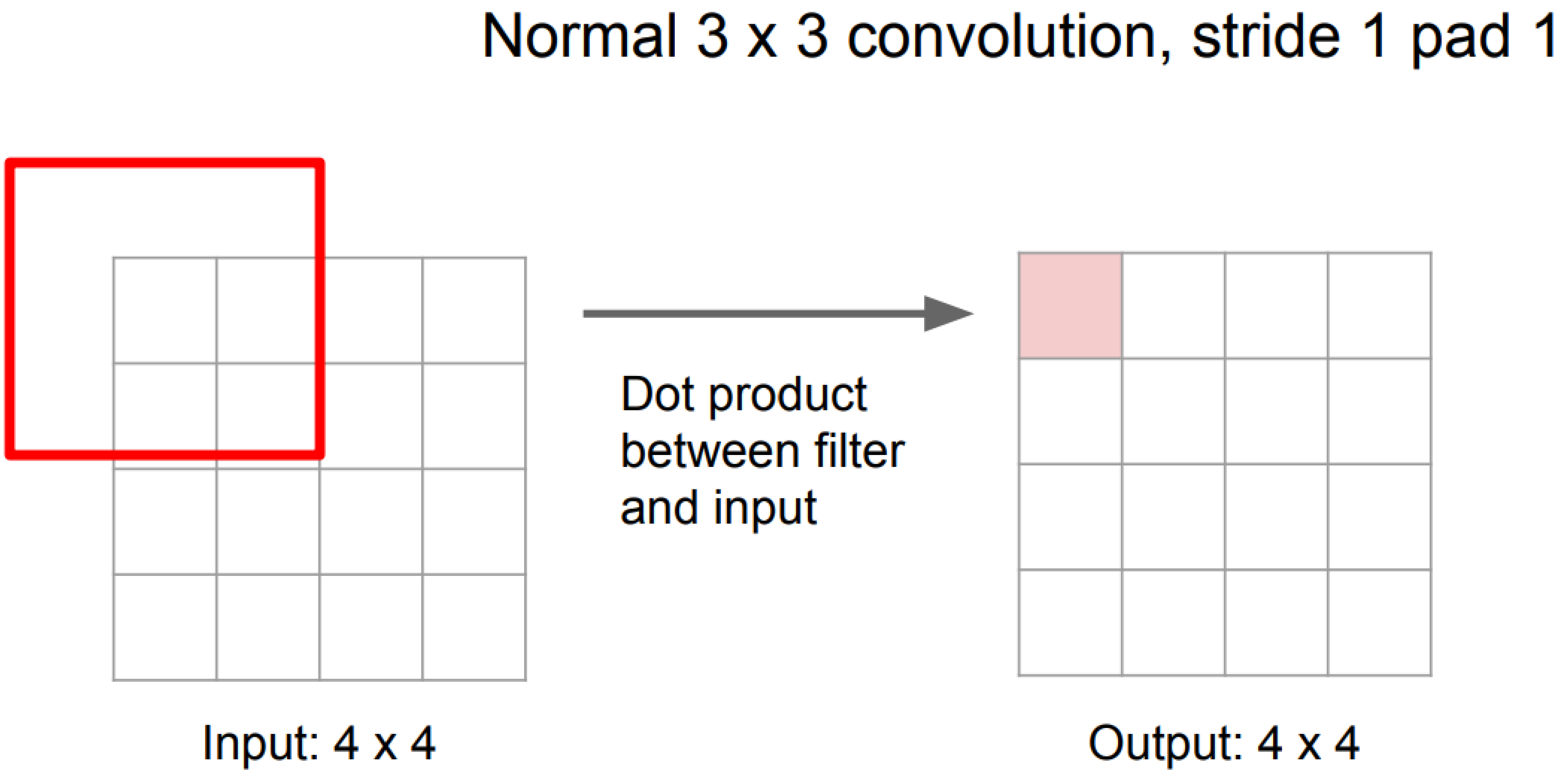
Learnable Upsampling: Transpose Convolution

Typical 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

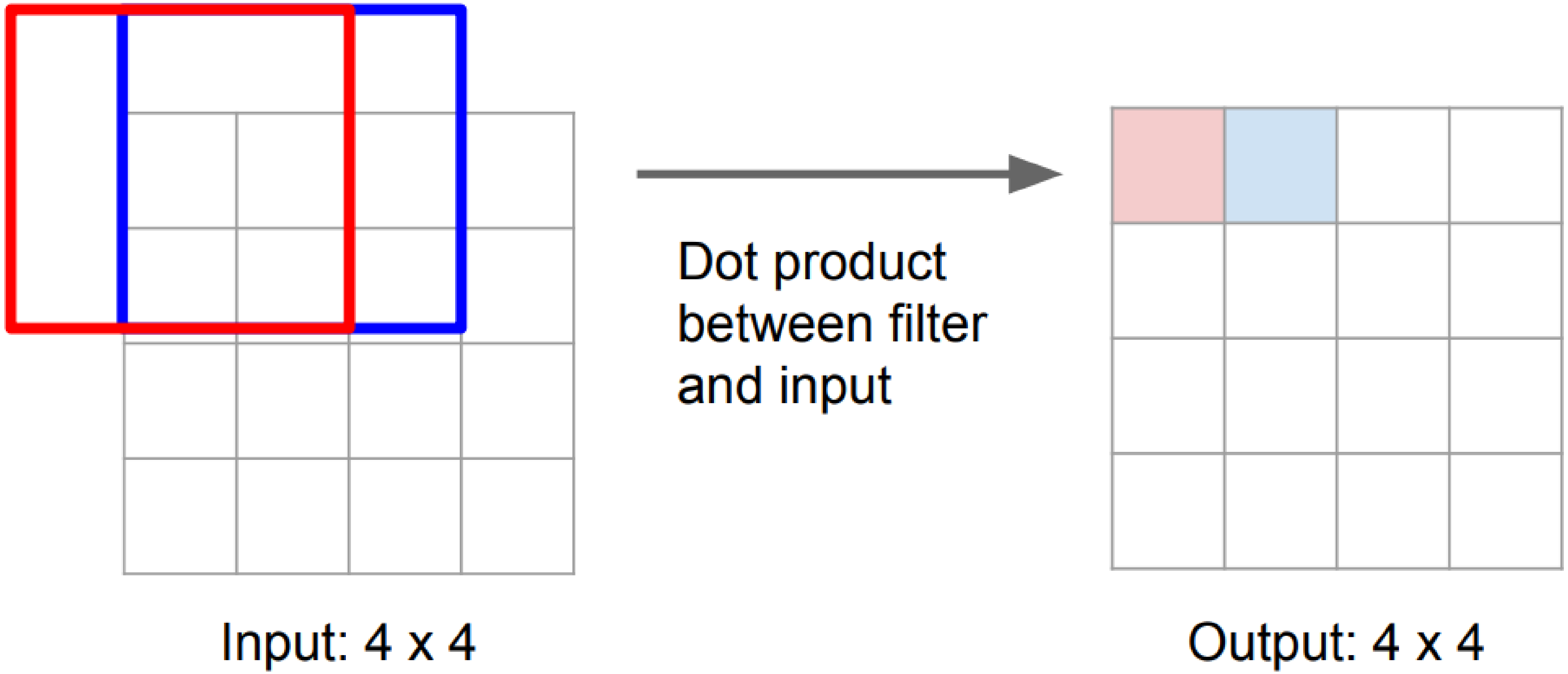
Output: 4 x 4

Learnable Upsampling: Transpose Convolution



Learnable Upsampling: Transpose Convolution

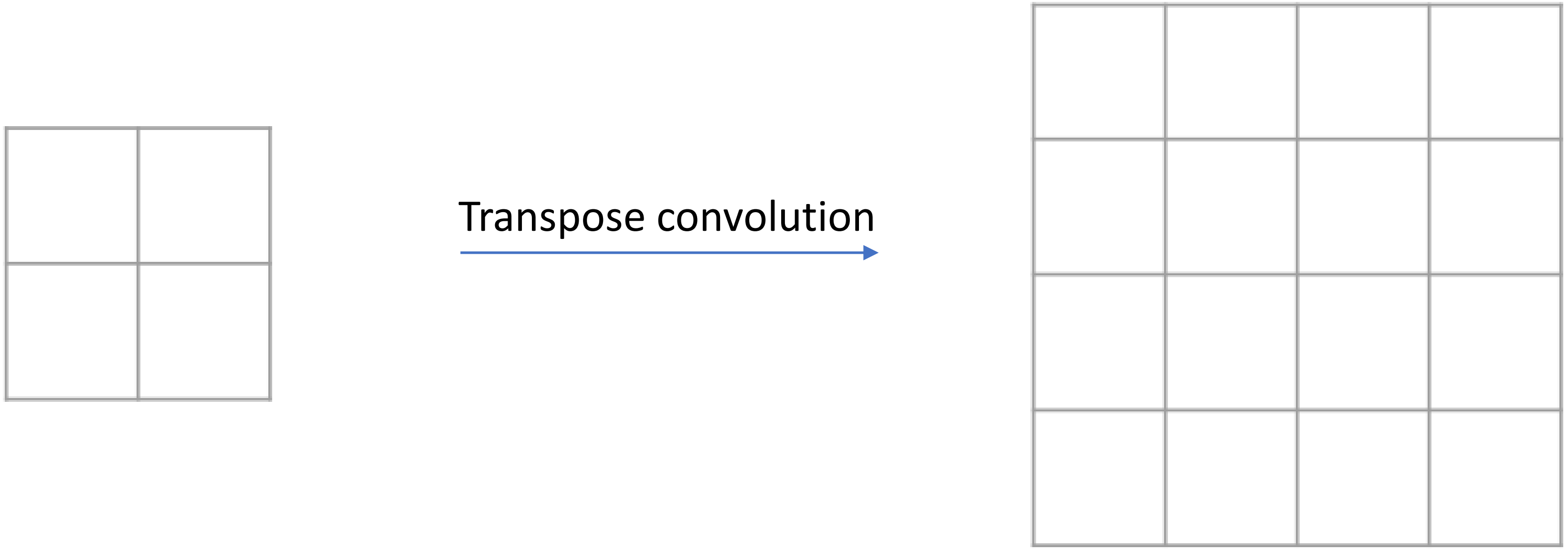
Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling: Transpose Convolution

Now the objective is increase the filter size for the decoder to match with the label

3 x 3 transpose convolution, stride 2 pad 1



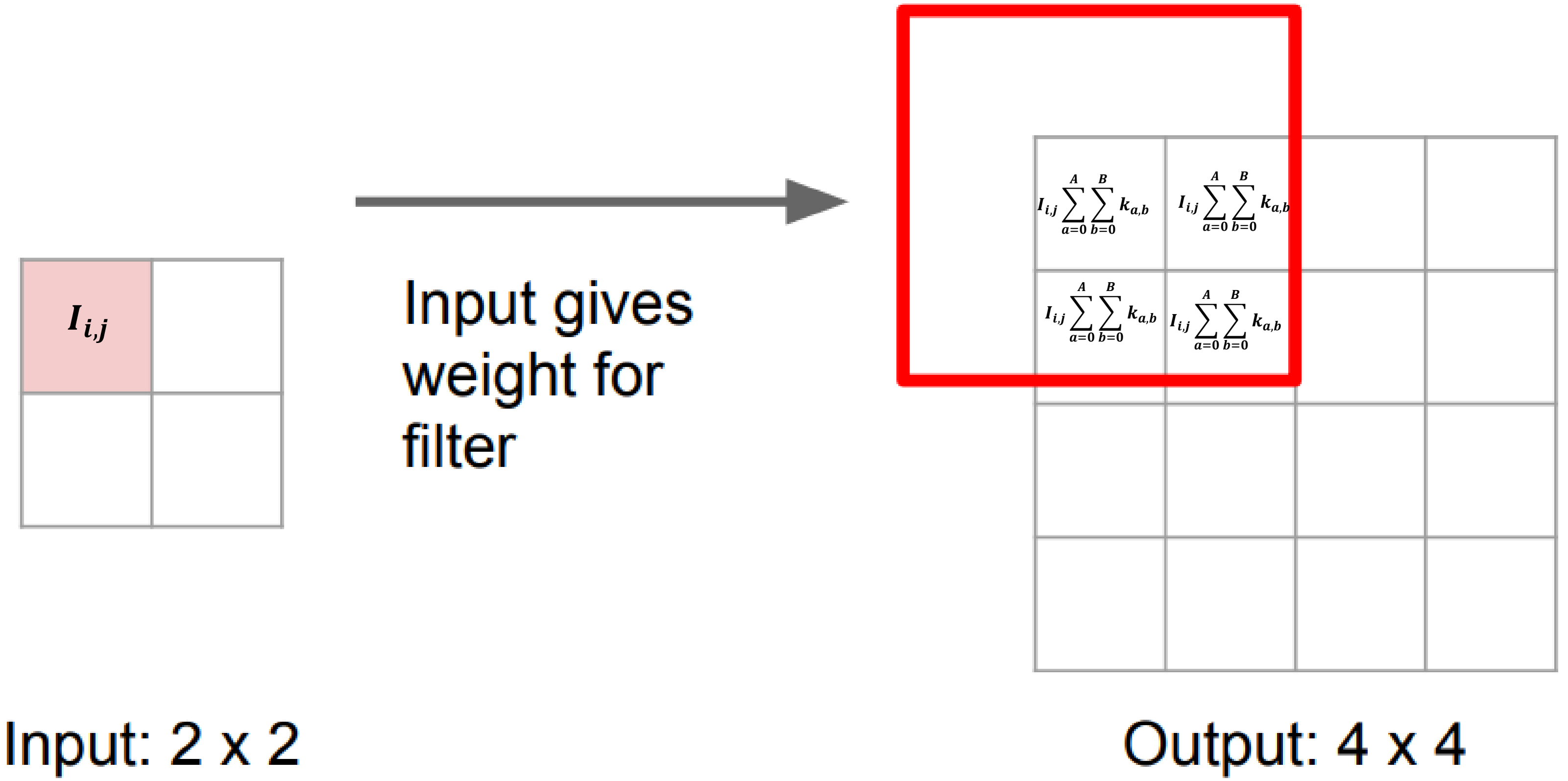
Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Transpose Convolution

Now the objective is increase the filter size for the decoder to match with the label

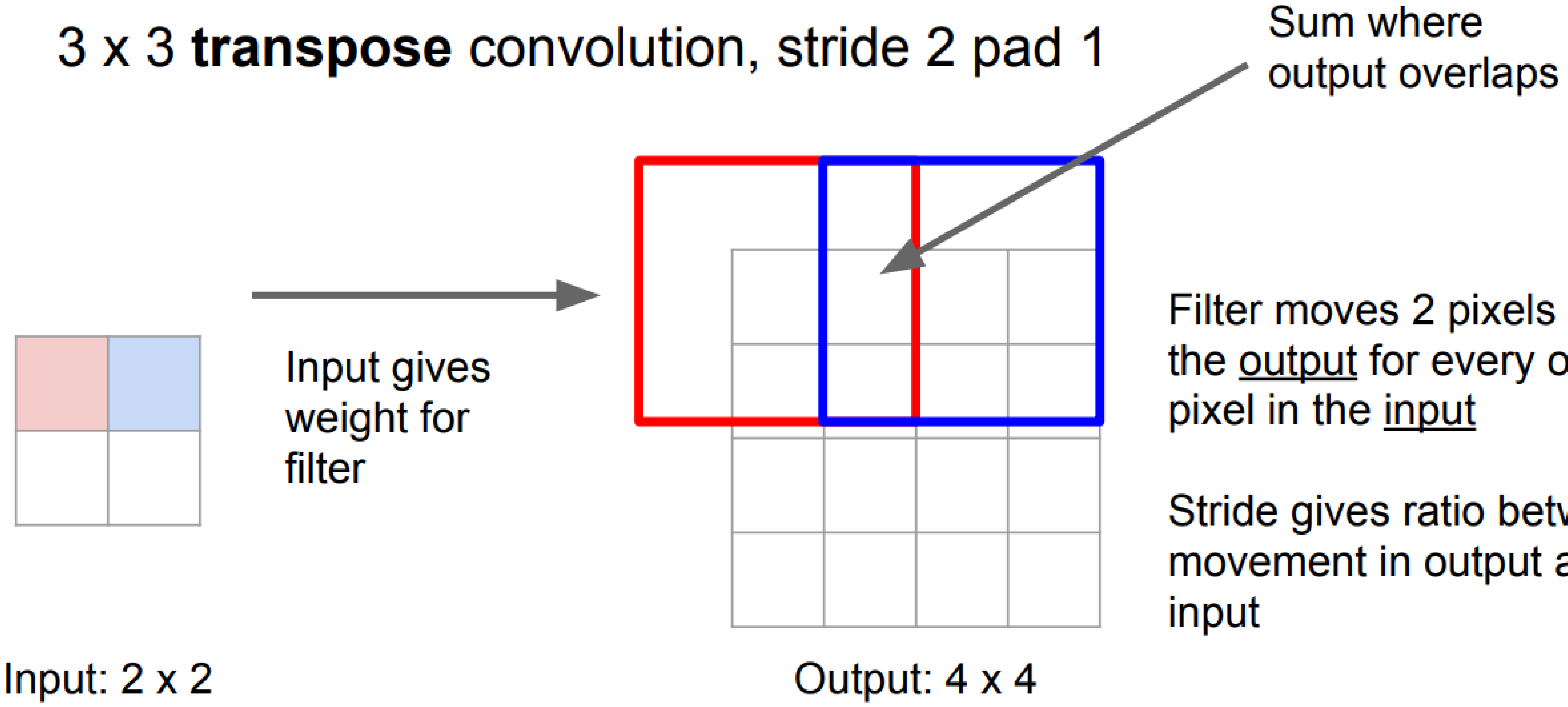
3 x 3 transpose convolution, stride 2 pad 1



$$I_{i,j}\sum_{a=0}^{A}\sum_{b=0}^{B}k_{a,b}$$

Learnable Upsampling: Transpose Convolution

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



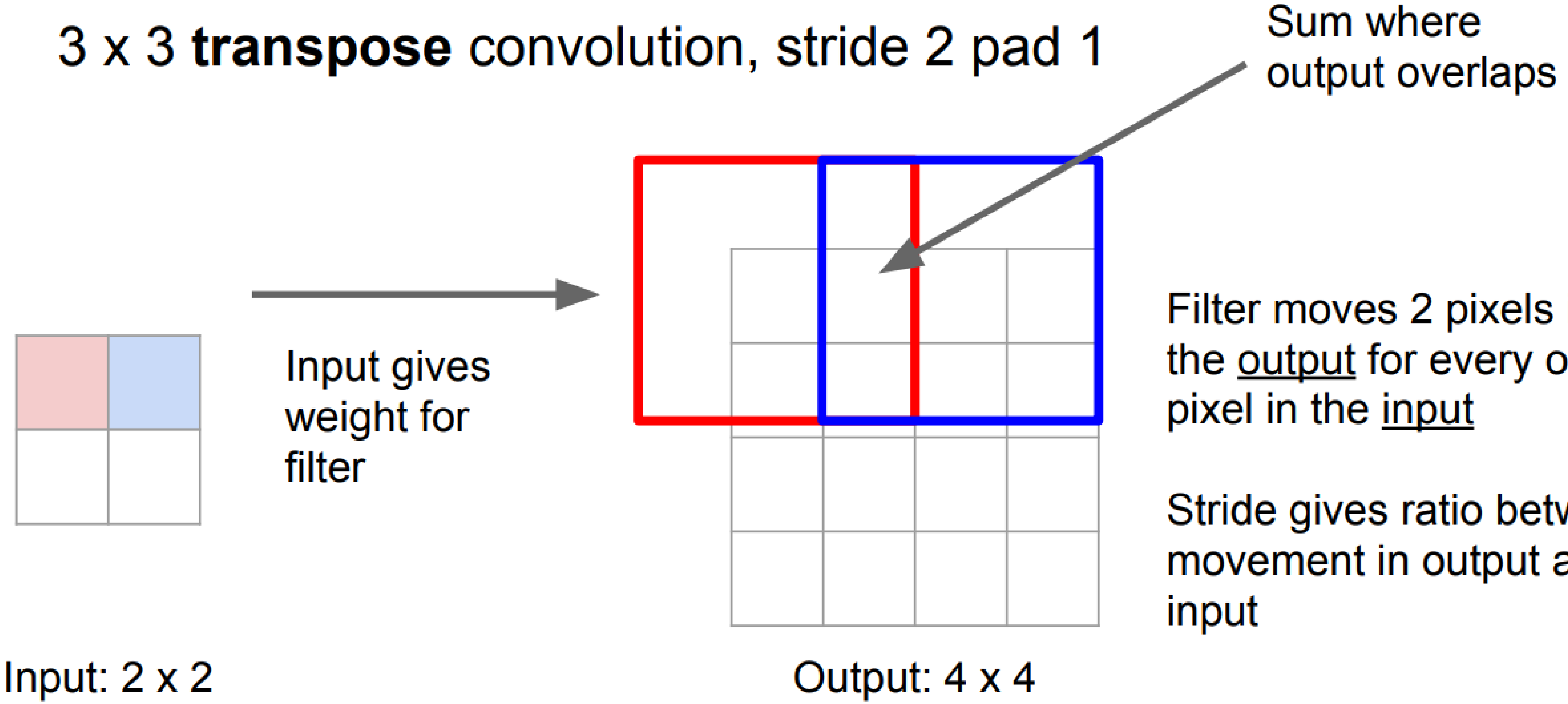
Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

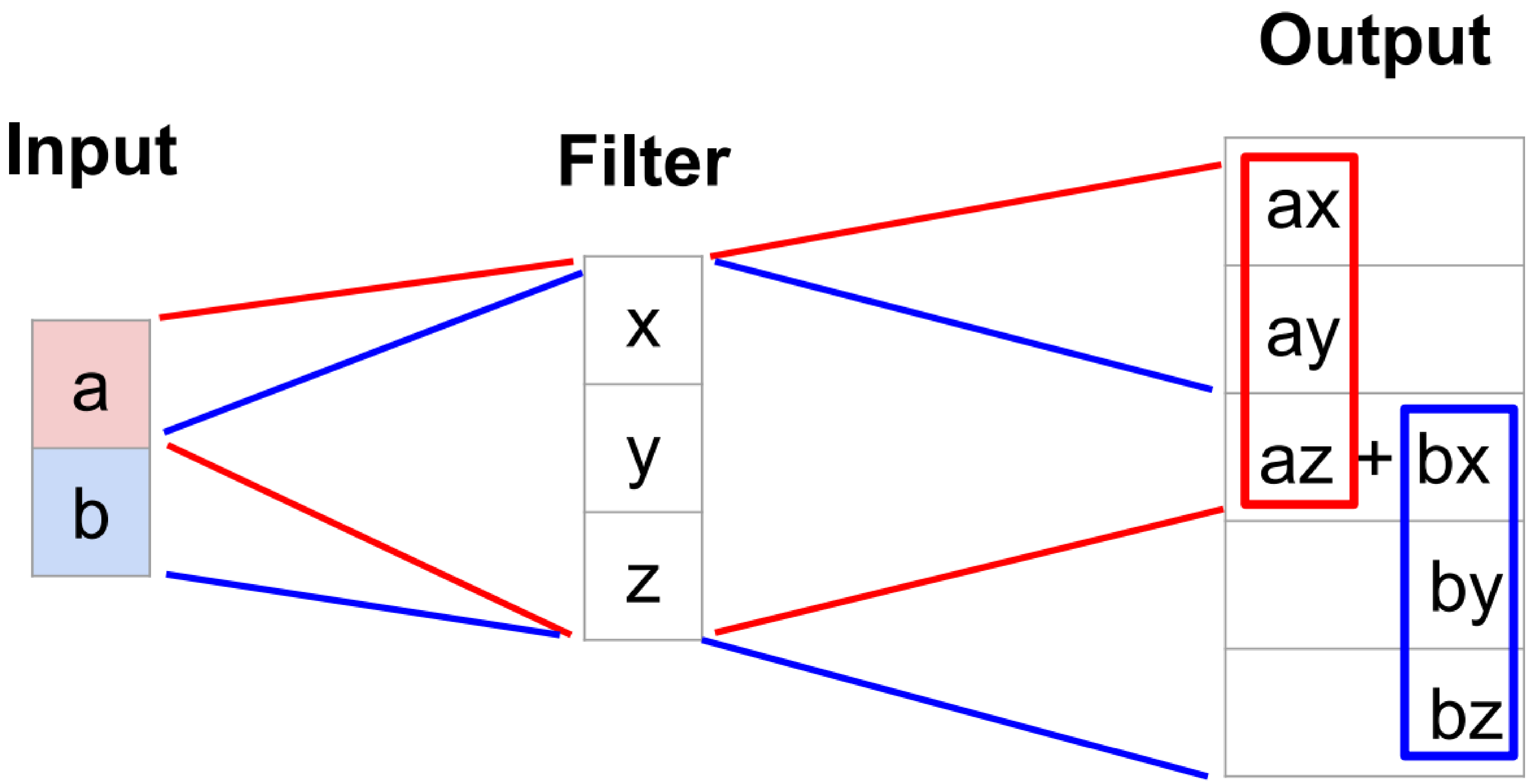
3 x 3 **transpose** convolution, stride 2 pad 1



Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

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)

We can express convolution in
terms of a matrix multiplication

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

kernel size = 3

Padding=1

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

stride=1

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Transpose Convolution

Convolution transpose multiplies by the transpose of the same matrix:

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

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

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

Transpose Convolution 1D Example in Segmentation

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

$$\overbrace{\begin{bmatrix} x & y & z \end{bmatrix}}^{\text{kernel size = 3}}$$

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

stride=2

Padding=1

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

Example: 1D conv, kernel size=3, <u>stride=2</u>, padding=1

Encoder

Convolution transpose multiplies by the transpose of the same matrix:

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

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} Ax \\ Ay \\ Az + Ax \\ By \\ Bz \\ 0 \end{bmatrix}$$

Decoder

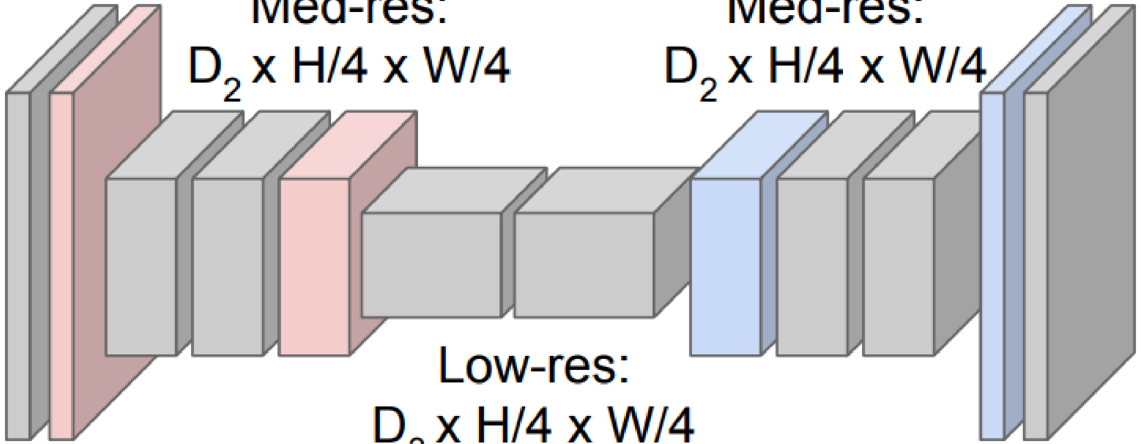
Semantic Segmentation Idea: Fully Convolutional

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

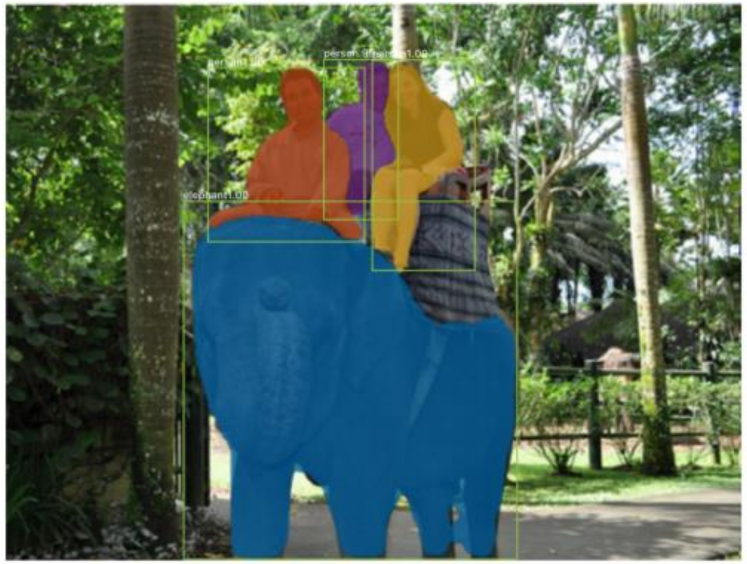
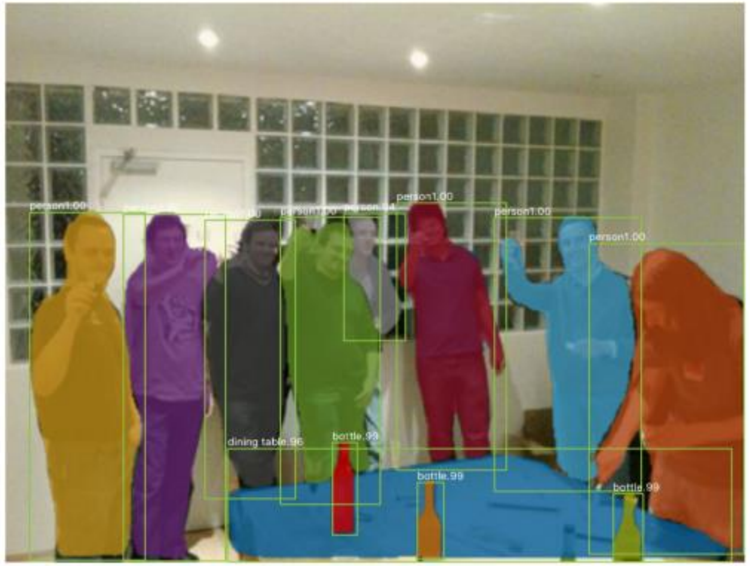
Input:
3 x H x W



Predictions:
H x W

Instance Segmentation

- Instance segmentation techniques needs semantic segmentation theories as well as object detection theories
- (object localization) R-CNN → Fast-RCNN → Faster-RCNN → Mask R-CNN (instance segmentation)
- This can be a topic to a next meeting

References

- Stanford Vision : http://cs231n.stanford.edu/
- A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "*A review on deep learning techniques applied to semantic segmentation,*"
- Farabet et al, "*Learning Hierarchical Features for Scene Labeling,*" TPAMI 2013
- Pinheiro and Collobert, "*Recurrent Convolutional Neural Networks for Scene Labeling*", ICML 2014
- Long, Shelhamer, and Darrell, "*Fully Convolutional Networks for Semantic Segmentation*", CVPR 2015
- Noh et al, "*Learning Deconvolution Network for Semantic Segmentation*", ICCV 2015
- K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R- ´ CNN. arXiv:1703.06870, 2017

Thank You