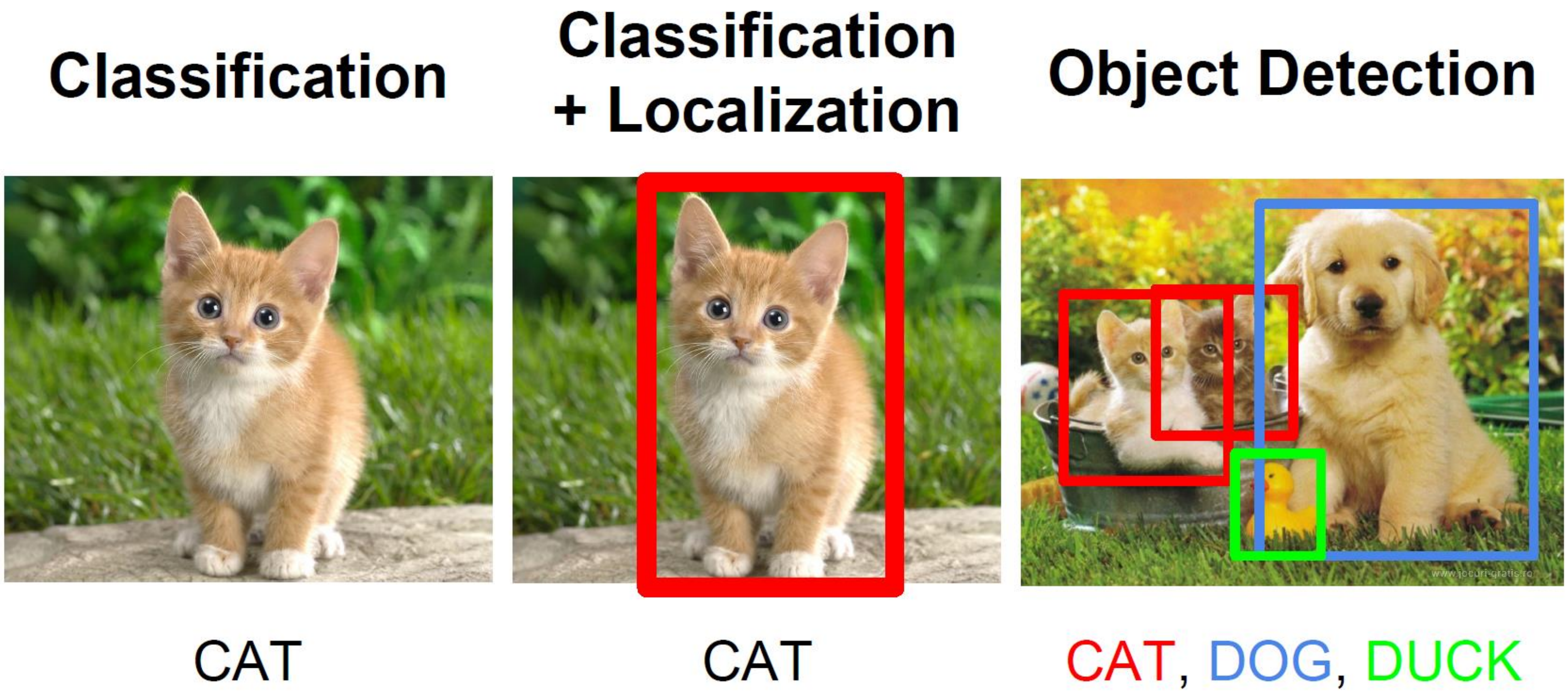
# CENG 506 Deep Learning

## Lecture 8 - Segmentation

Slides were prepared using the course material of
Stanford's CNN Course (CS231n by Fei-Fei, Johnson, Yeung)

# Previously we have seen

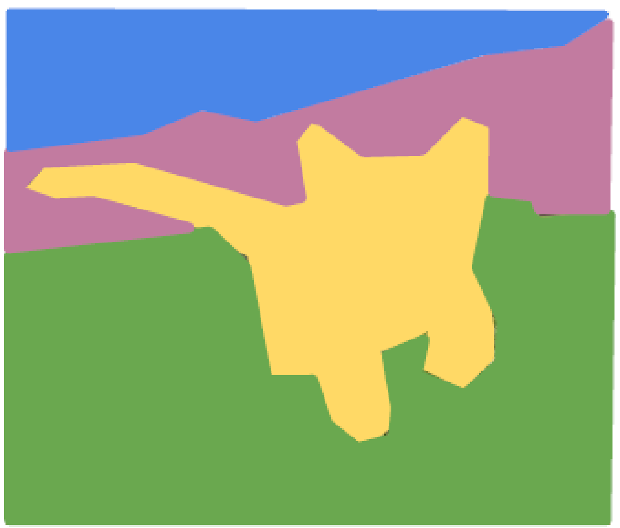



Single object

Multiple objects

# Today

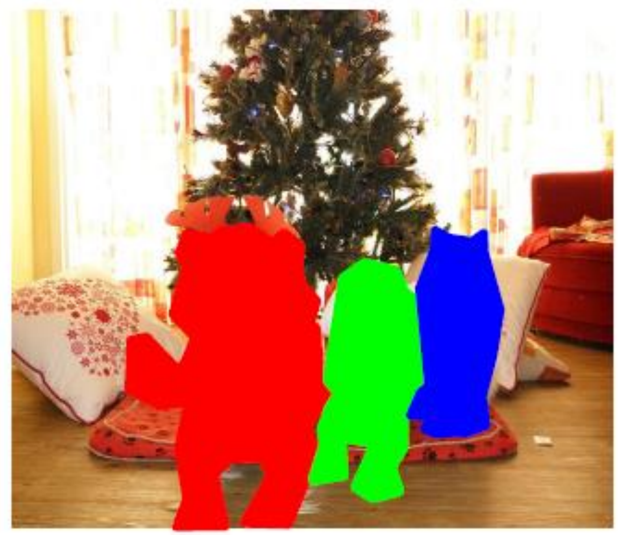
**Semantic Segmentation**



**GRASS, CAT, TREE, SKY**

No objects, just pixels

**Instance Segmentation**

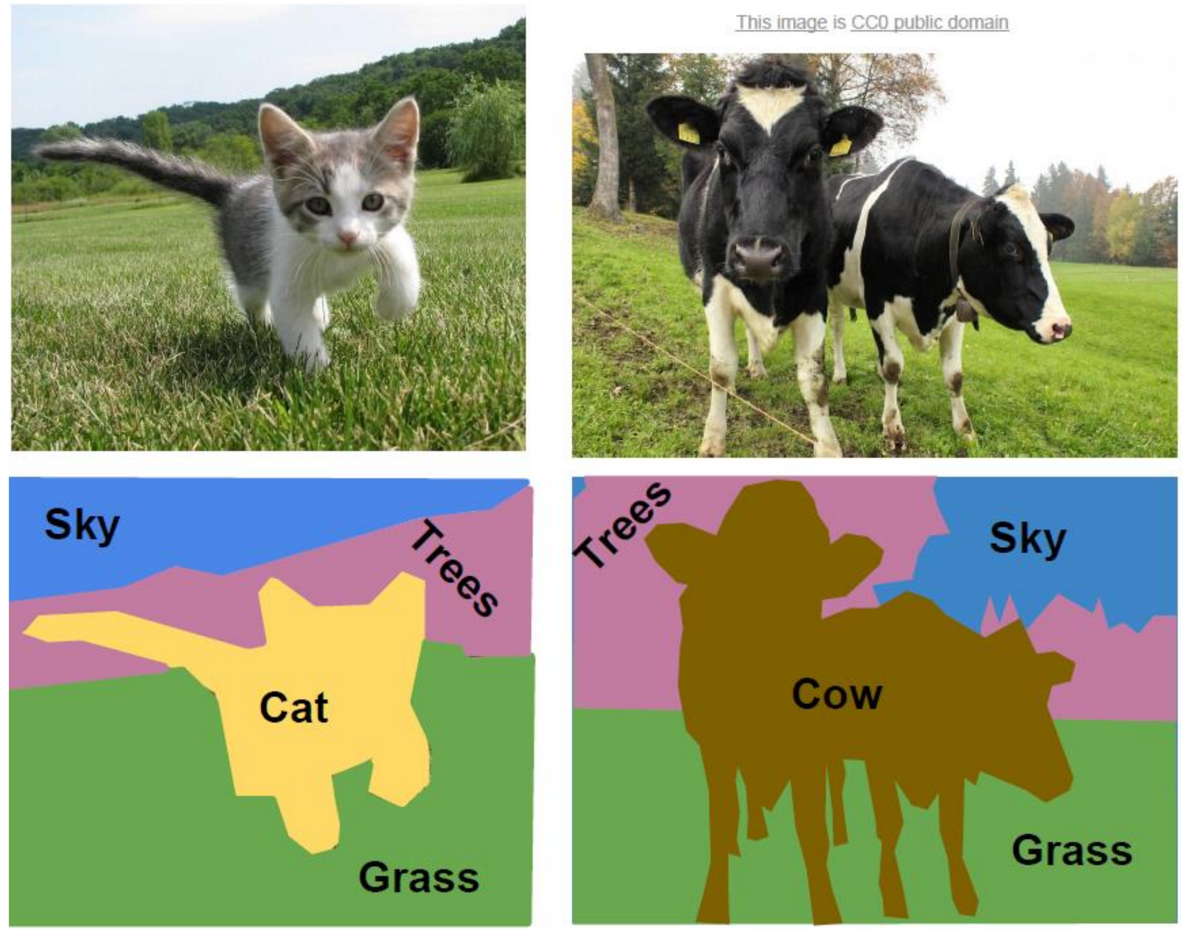


**DOG, DOG, CAT**

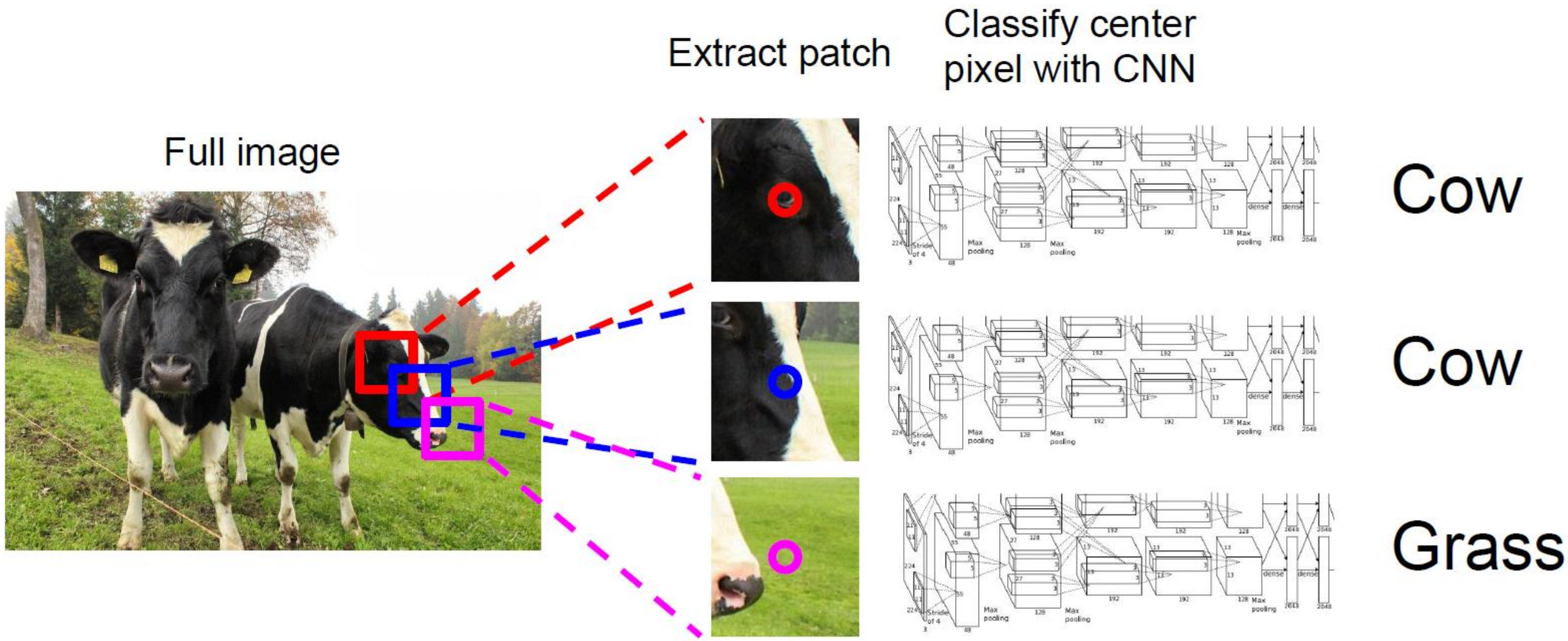
# Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



# Idea 1: Sliding window

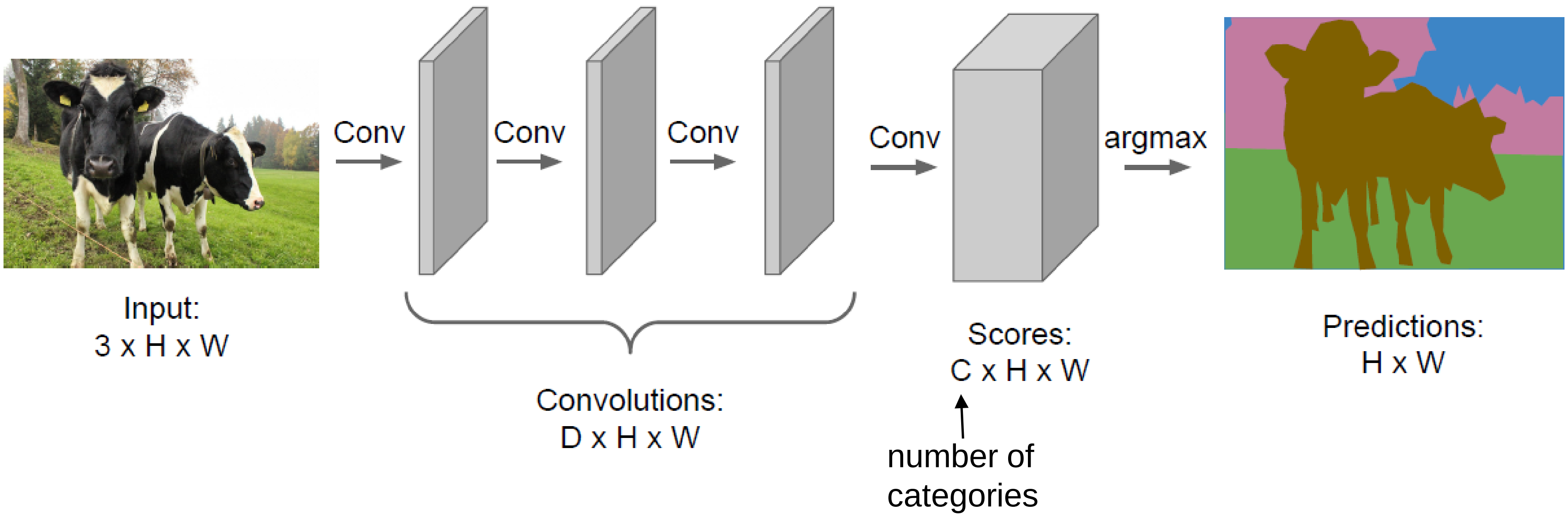



Problem: Very inefficient!
Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Idea 2: Fully convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!




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

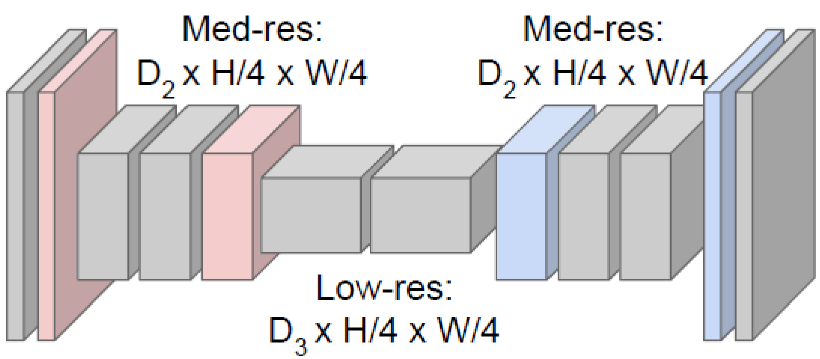
# Idea 2: Fully convolutional

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

Downsampling: pooling, convolution

Upsampling ?

Input:
3 x H x W




High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

Predictions:
H x W

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

# In-Network upsampling: “Unpooling”

**Nearest Neighbor**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 2 |
| 3 | 3 | 4 | 4 |
| 3 | 3 | 4 | 4 |

Output: 4 x 4

**“Bed of Nails”**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

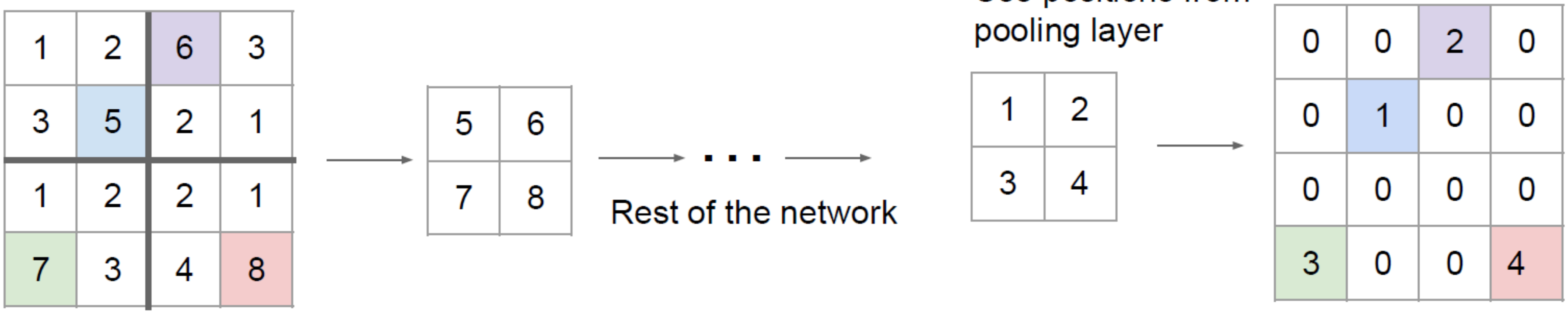
| | | | |
|---|---|---|---|
| 1 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 |

Output: 4 x 4

# In-Network upsampling: “Max Unpooling”

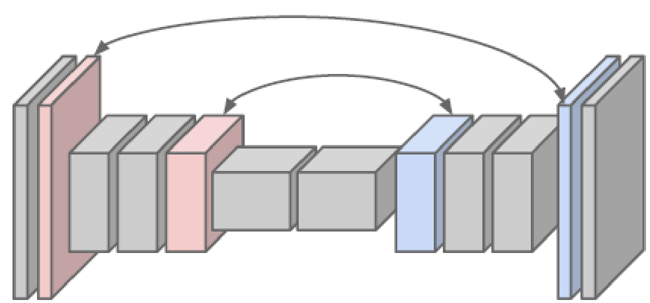
**Max Pooling**
Remember which element was max!

**Max Unpooling**
Use positions from pooling layer



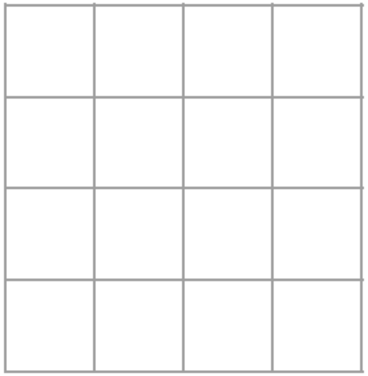

Input: 4 x 4 Output: 2 x 2 Input: 2 x 2 Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

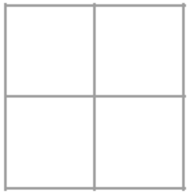


# Learnable Upsampling: Transpose Convolution

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



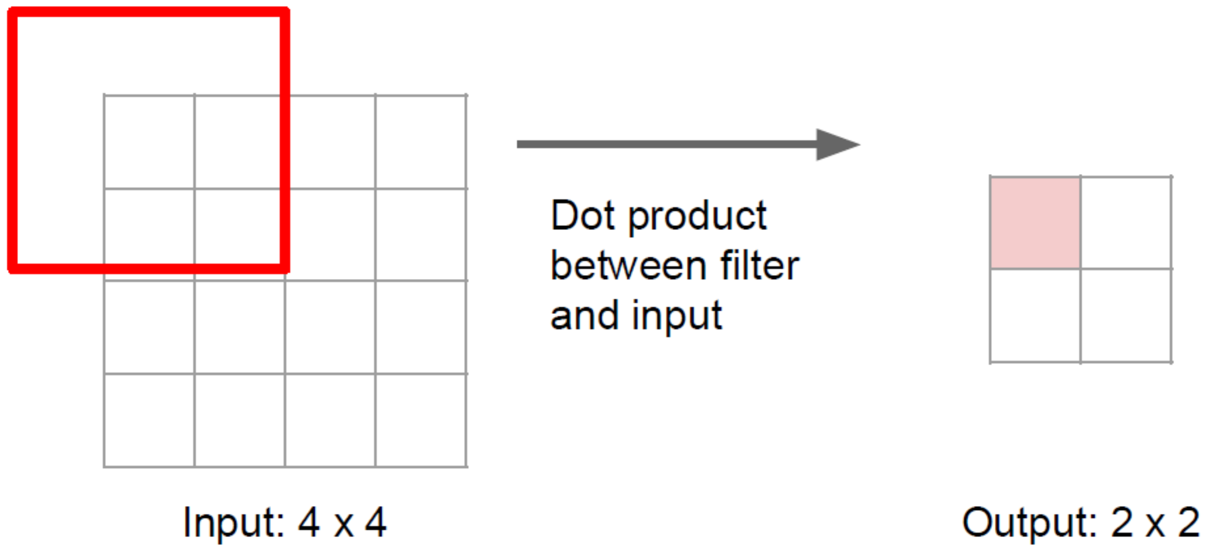
Input: 4 x 4



Output: 2 x 2

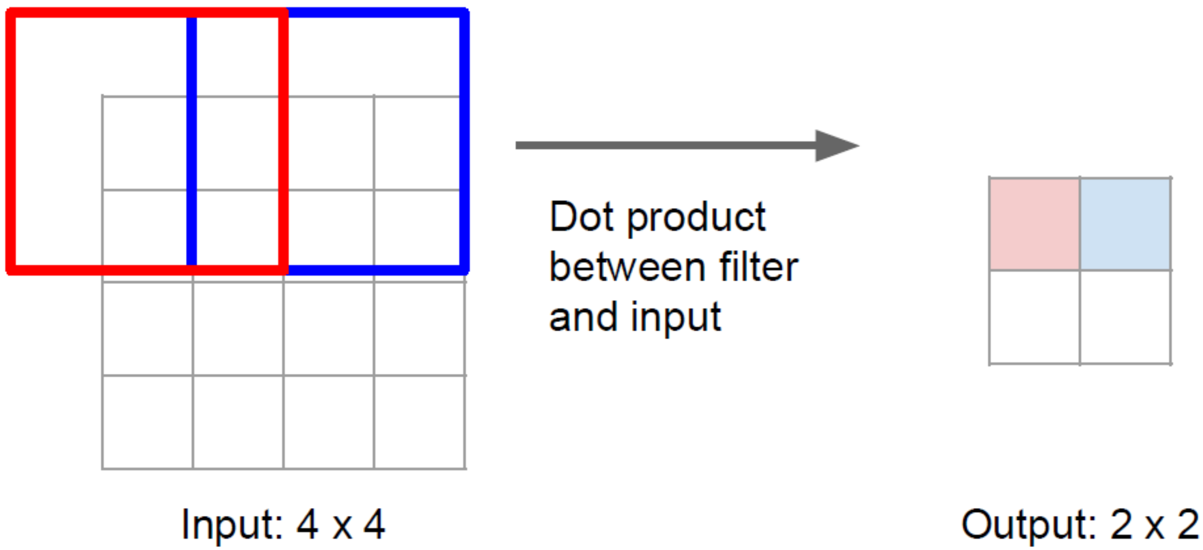
# Learnable Upsampling: Transpose Convolution

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



# Learnable Upsampling: Transpose Convolution

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

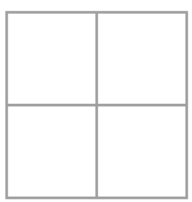



Filter moves 2 pixels in input for every one pixel in output

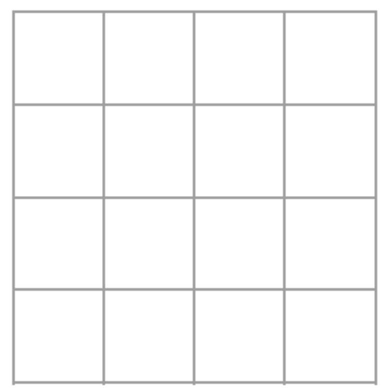
Stride gives ratio between movement in input and output

# Learnable Upsampling: Transpose Convolution

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



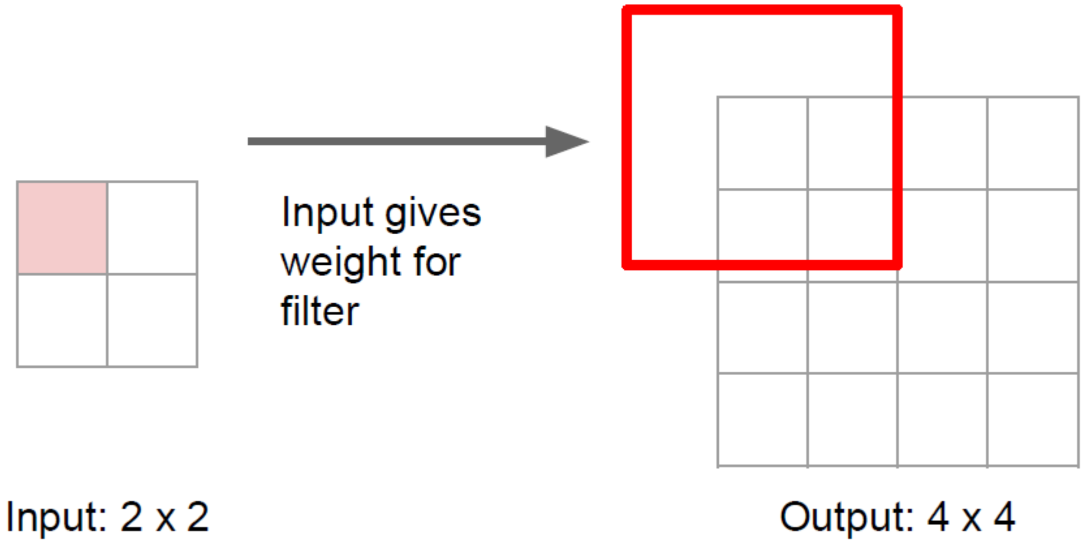
Input: 2 x 2



Output: 4 x 4

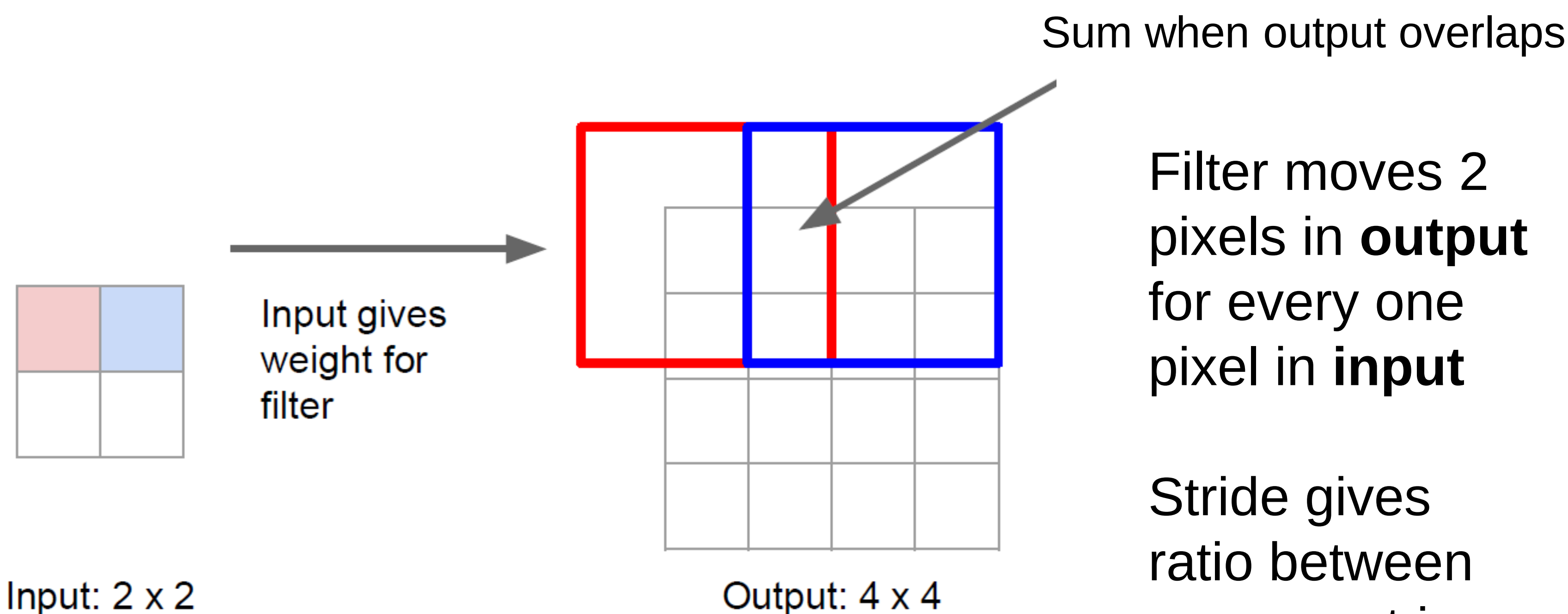
# Learnable Upsampling: Transpose Convolution

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



# Learnable Upsampling: Transpose Convolution

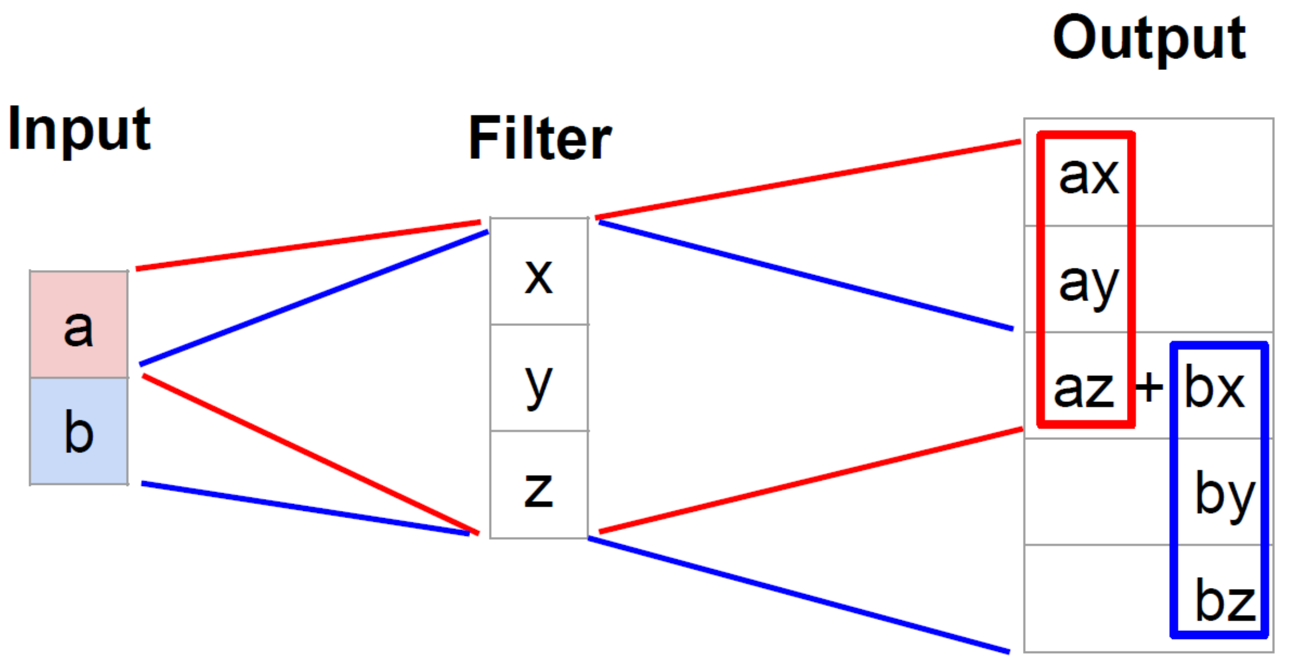
3 x 3 **transpose** convolution, stride 2 pad 1




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

Stride gives ratio between movement in output and input

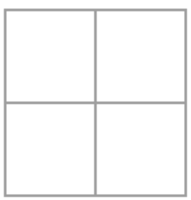
# Transpose Convolution: 1D Example



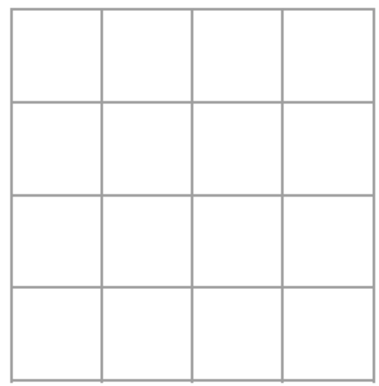

Output contains copies of the filter weighted by the input, summing where they overlap in the output

Need to crop one pixel from output to make output exactly 2x input

# Exercise: 3 x 3 **transpose** convolution, stride 2 pad 1



Input: 2 x 2



Output: 4 x 4

# Learnable Upsampling

Other Names for Transpose Convolution:
- Deconvolution
- Upconvolution

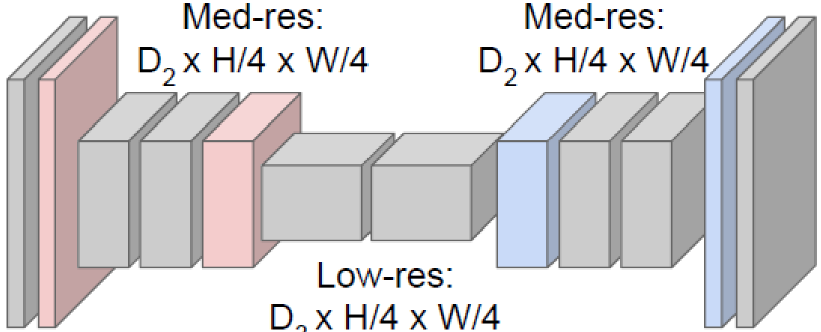
# Idea 2: Fully convolutional

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

Downsampling: pooling, convolution

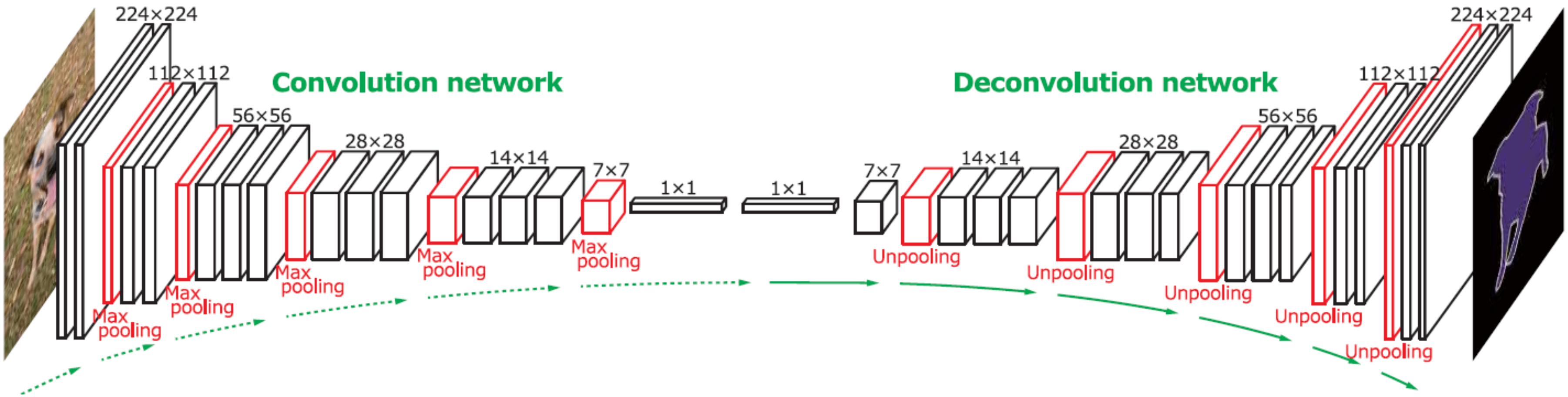
Upsampling: Unpooling and transpose convolution (deconvolution)

Input:
3 x H x W




Predictions:
H x W

# Idea 2 Example: Noh et al, 2015




- Convolution network: VGG 16-layer net.
- Given a feature vector obtained from the convolution network, a deconvolution network produces pixel-wise segmentation map.
- Deconvolution layers densify the sparse activations by unpooling and multiple learned deconvolution filters.

Figure source: Noh, Hong, and Han “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

# Idea 2 Example: Noh et al, 2015

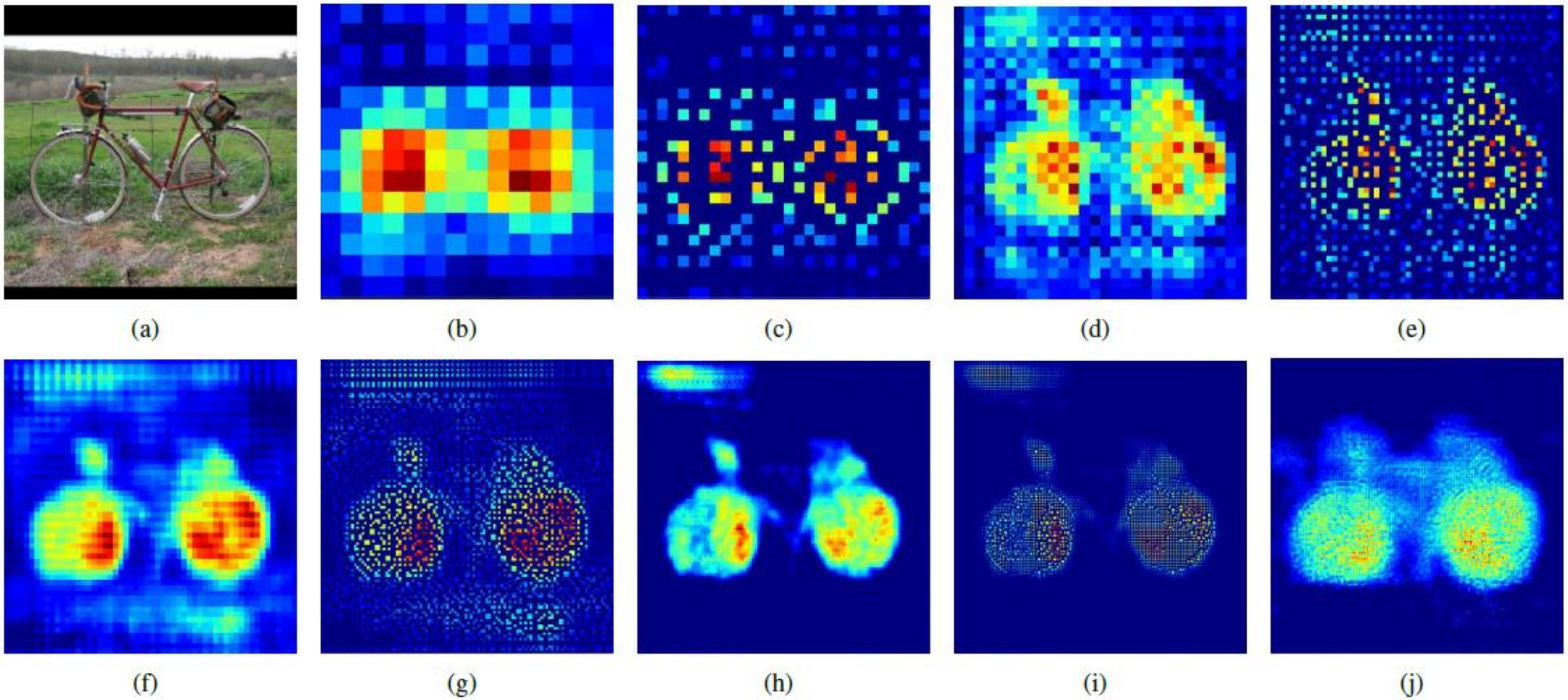
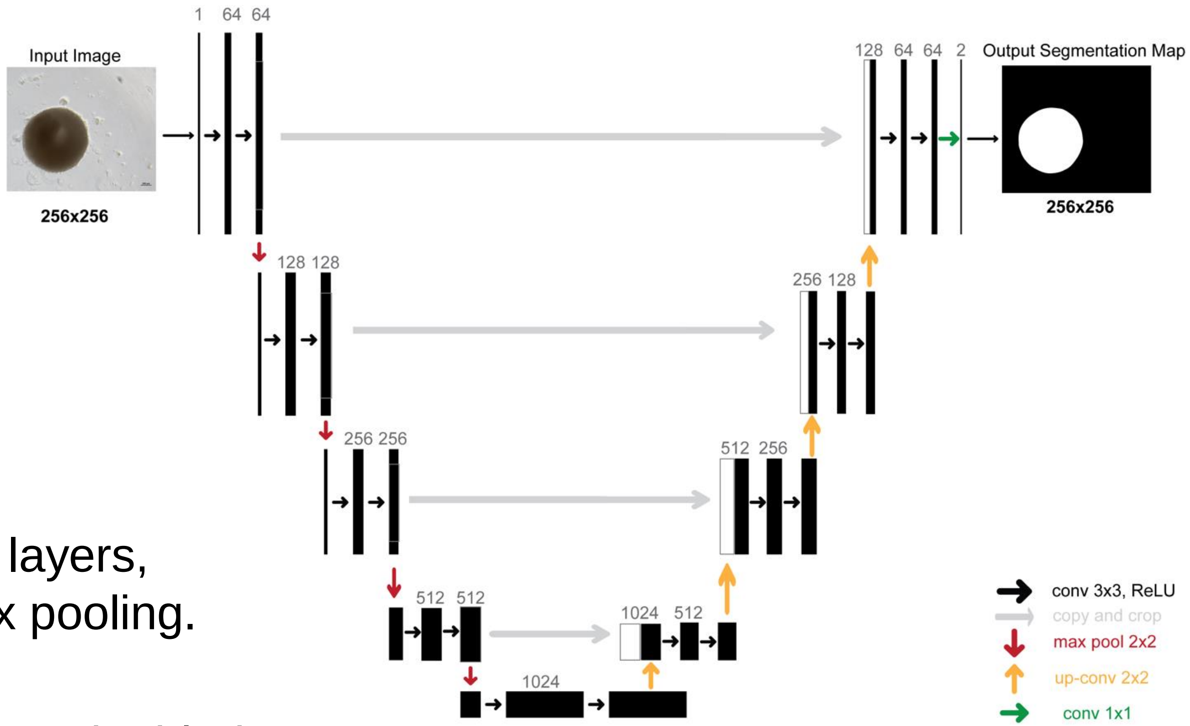


Figure 4. Visualization of activations in our deconvolution network. The activation maps from (b) to (j) correspond to the output maps from lower to higher layers in the deconvolution network. We select the most representative activation in each layer for effective visualization. The image in (a) is an input, and the rest are the outputs from (b) the last $14 \times 14$ deconvolutional layer, (c) the $28 \times 28$ unpooling layer, (d) the last $28 \times 28$ deconvolutional layer, (e) the $56 \times 56$ unpooling layer, (f) the last $56 \times 56$ deconvolutional layer, (g) the $112 \times 112$ unpooling layer, (h) the last $112 \times 112$ deconvolutional layer, (i) the $224 \times 224$ unpooling layer and (j) the last $224 \times 224$ deconvolutional layer. The finer details of the object are revealed, as the features are forward-propagated through the layers in the deconvolution network. Note that noisy activations from background are suppressed through propagation while the activations closely related to the target classes are amplified. It shows that the learned filters in higher deconvolutional layers tend to capture class-specific shape information.

Noh, Hong, and Han, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

# Another Idea 2 Example: U-Net

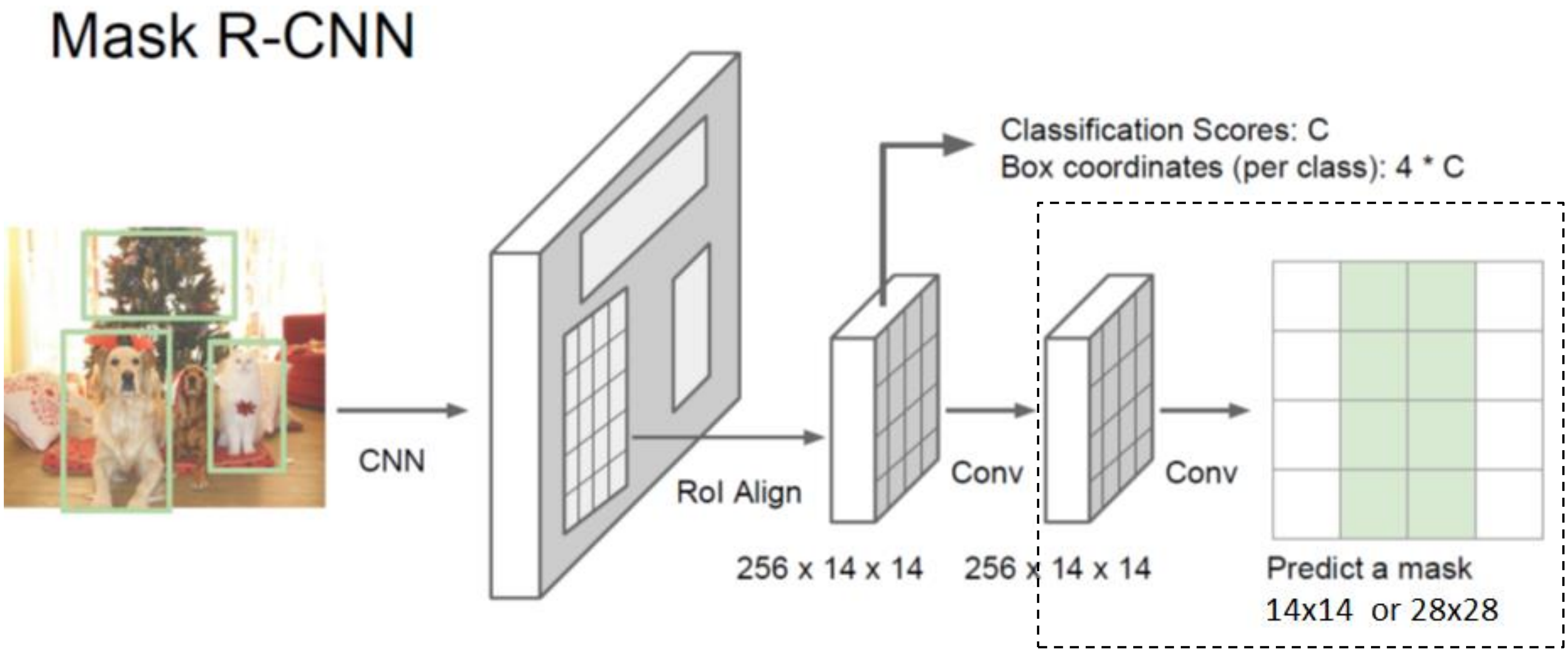



- Repeated double 3x3 conv layers, each followed by a 2x2 max pooling.
- After each downsampling, the no of feature channels are doubled.
- In the second half, each up-sampling step is followed by a 2x2 up-convolution and a concatenation with the corresponding level of the first half.
- Final layer uses a convolution to reduce the feature map depth to the desired number of classes.

Source: Ronneberger, Fischer, and Brox. U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015.

# Instance Segmentation

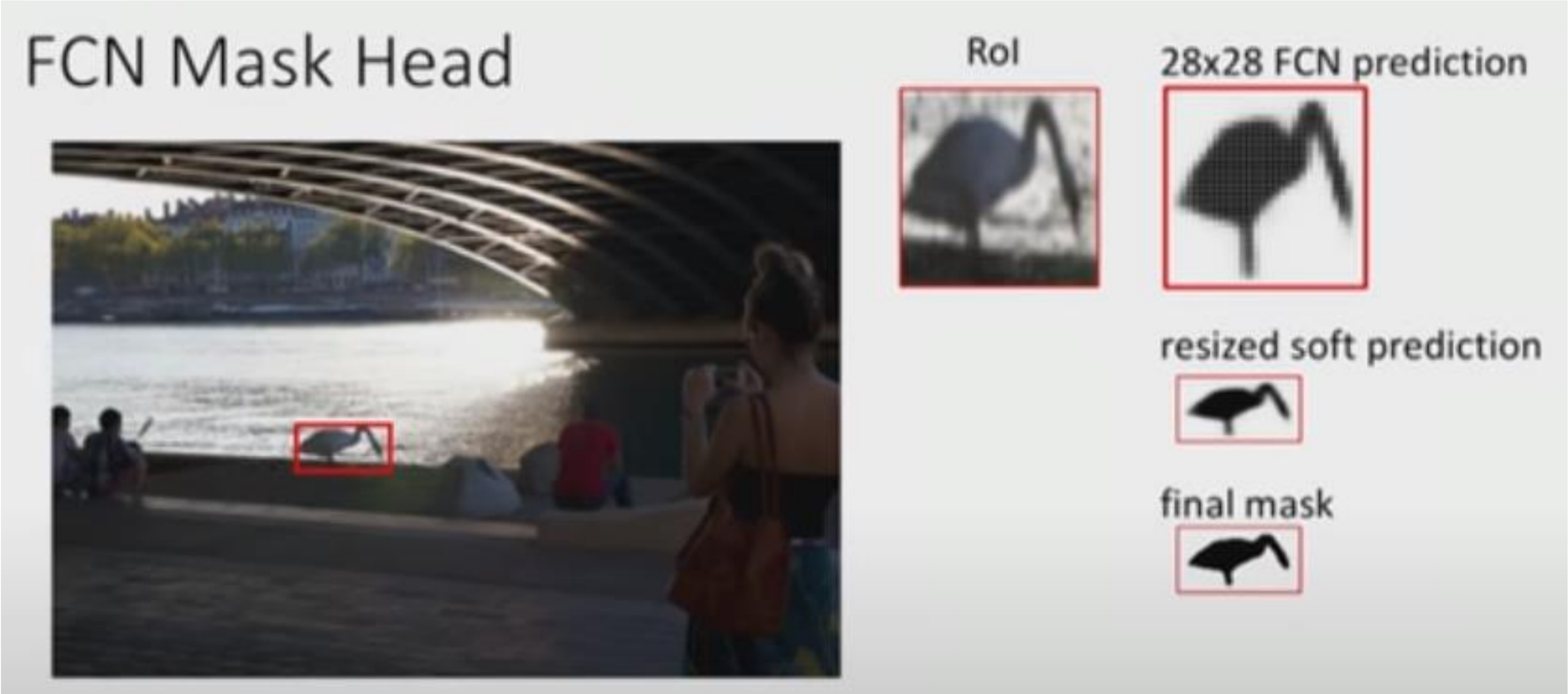
# Instance Segmentation
## (Object detection + Semantic segmentation)




We add a third 'Mask' head
to Faster R-CNN.

He et al, "Mask R-CNN", ICCV 2017.

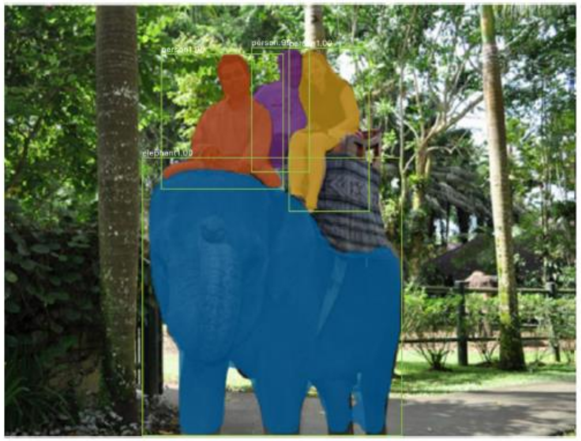
# Mask R-CNN

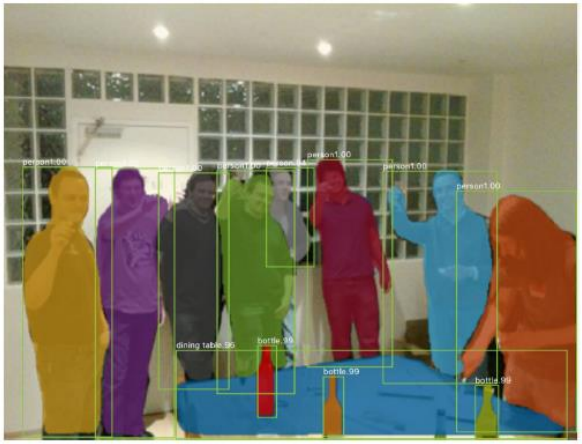



'Mask head' is actually a fully convolutional network with some downsampling and upsampling.

He et al, "Mask R-CNN", ICCV 2017.

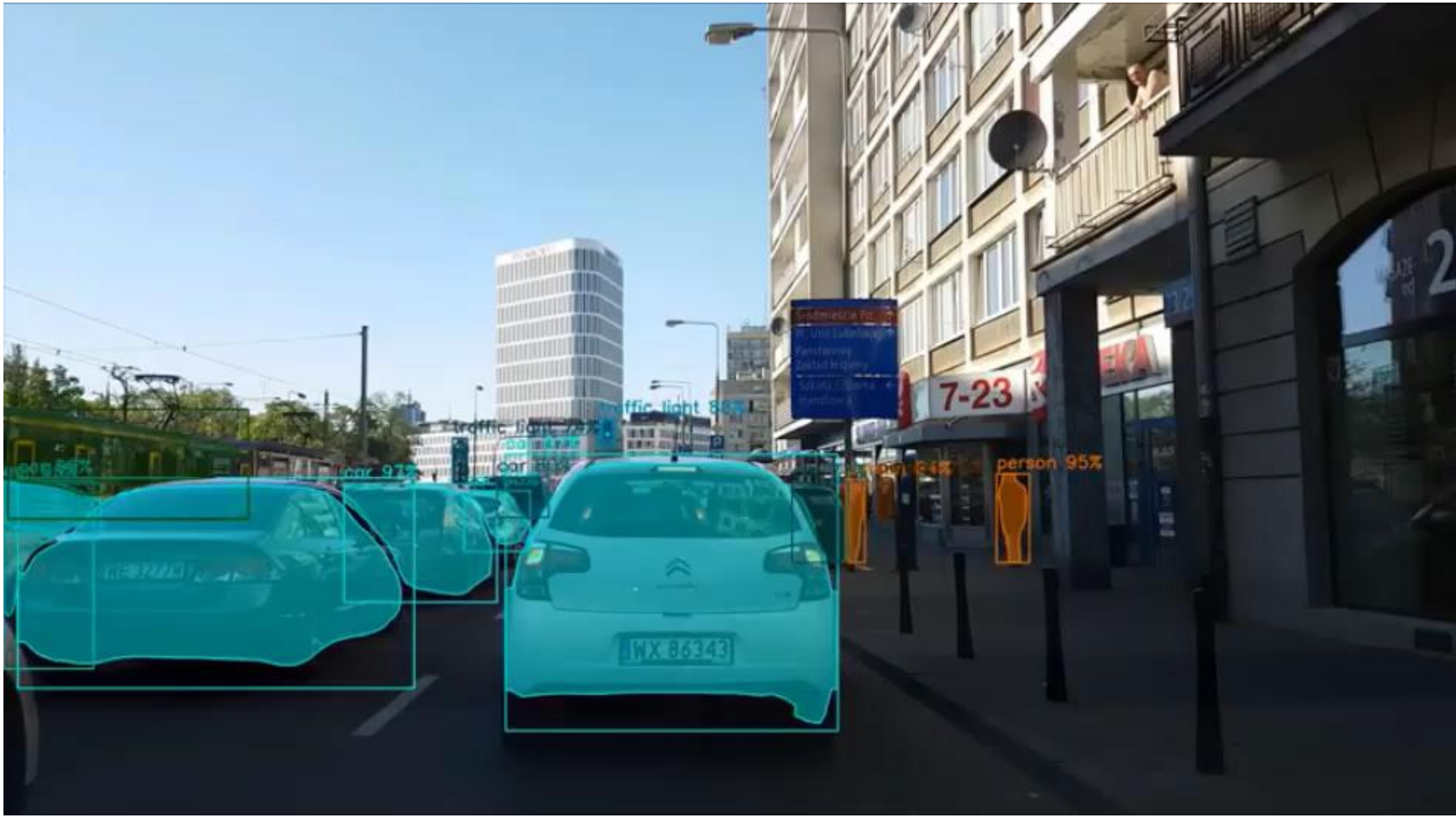
# Mask R-CNN: Very Good Results!





He et al, “Mask R-CNN”, ICCV 2017.

# Mask R-CNN: Very Good Results!




See the full video at: https://www.youtube.com/watch?v=OOT3UIXZztE

# Panoptic Segmentation

Semantic segmentation covers all pixels of the image. Instance segmentation concentrates on instances.

We can combine the results of separate semantic segmentation and instance segmentation tasks via a network.

Figure source: Efficient PS: Deep Convolutional Neural Networks for Panoptic Segmentation
Rohit Mohan and Abhinav Valada, Univ. of Freiburg, http://panoptic.cs.uni-freiburg.de/

# Panoptic Segmentation: EfficientPS

EfficientPS consists of a shared backbone, a two-way Feature Pyramid Network (FPN), instance and semantic heads, and a panoptic fusion module.

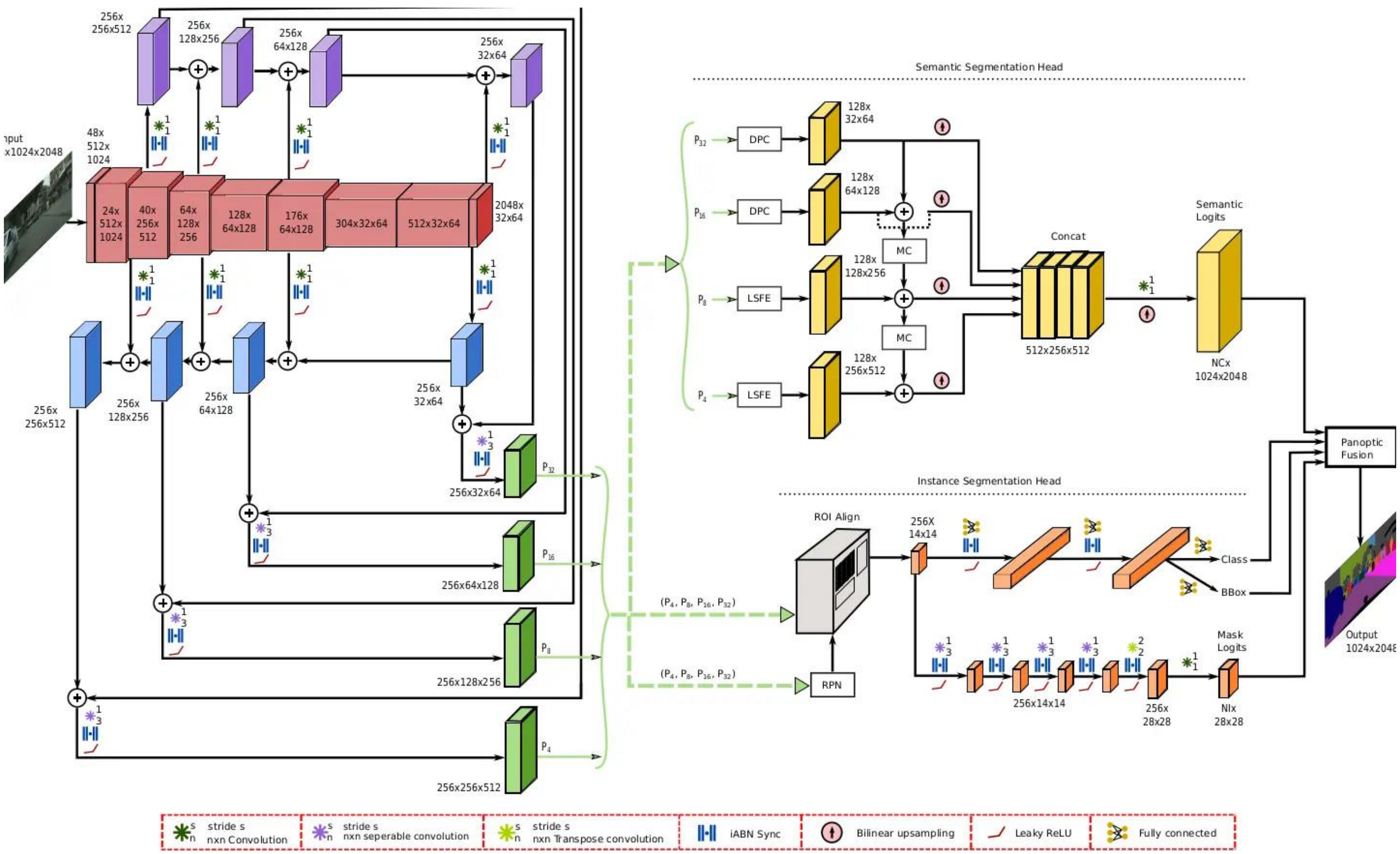



Figure source: Efficient PS: Deep Convolutional Neural Networks for Panoptic Segmentation
Rohit Mohan and Abhinav Valada, Univ. of Freiburg, http://panoptic.cs.uni-freiburg.de/