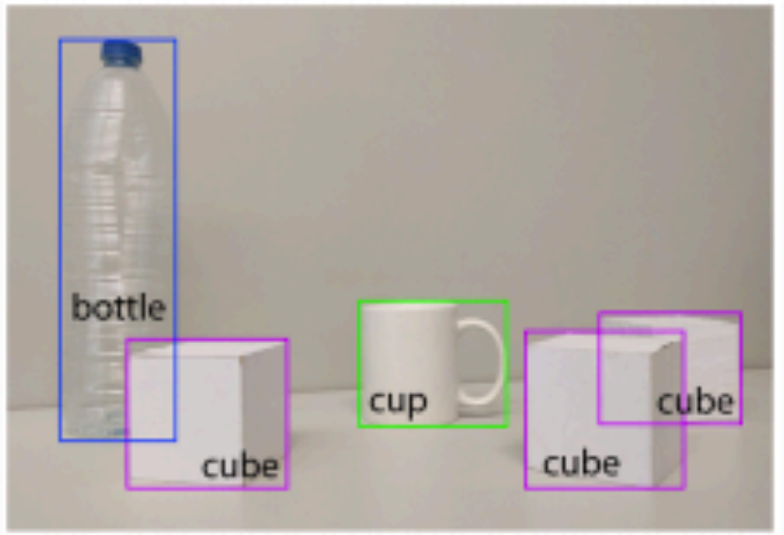
딥러닝과 머신러닝

#3 Semantic Segmentation

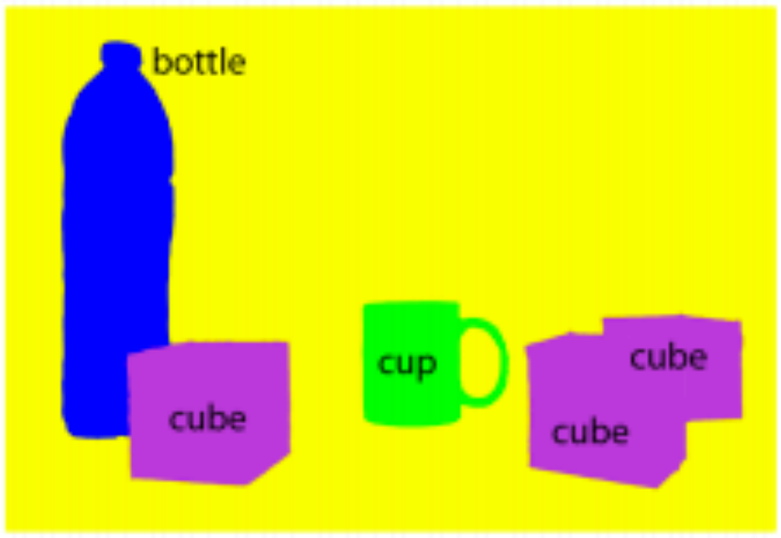
이근영(메디웨일)

Semantic Segmentation

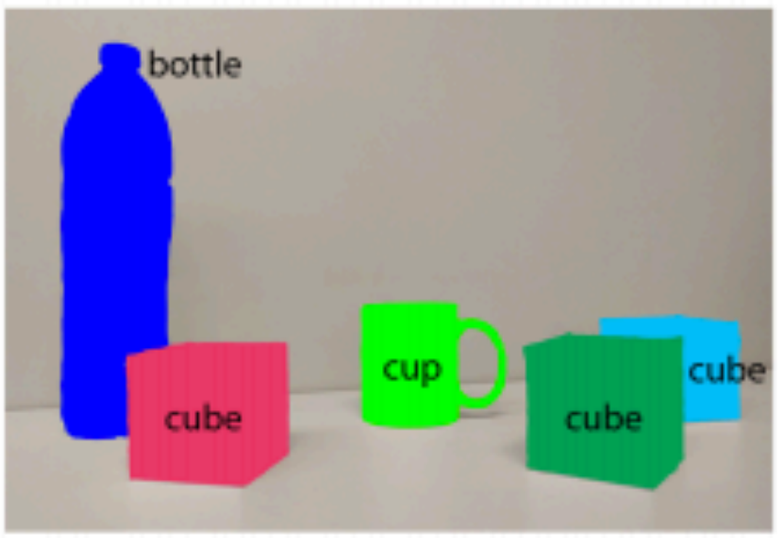
(a) Image classification



(b) Object localization



(c) Semantic segmentation



(d) Instance segmentation

Fully Convolution Network(FCN)

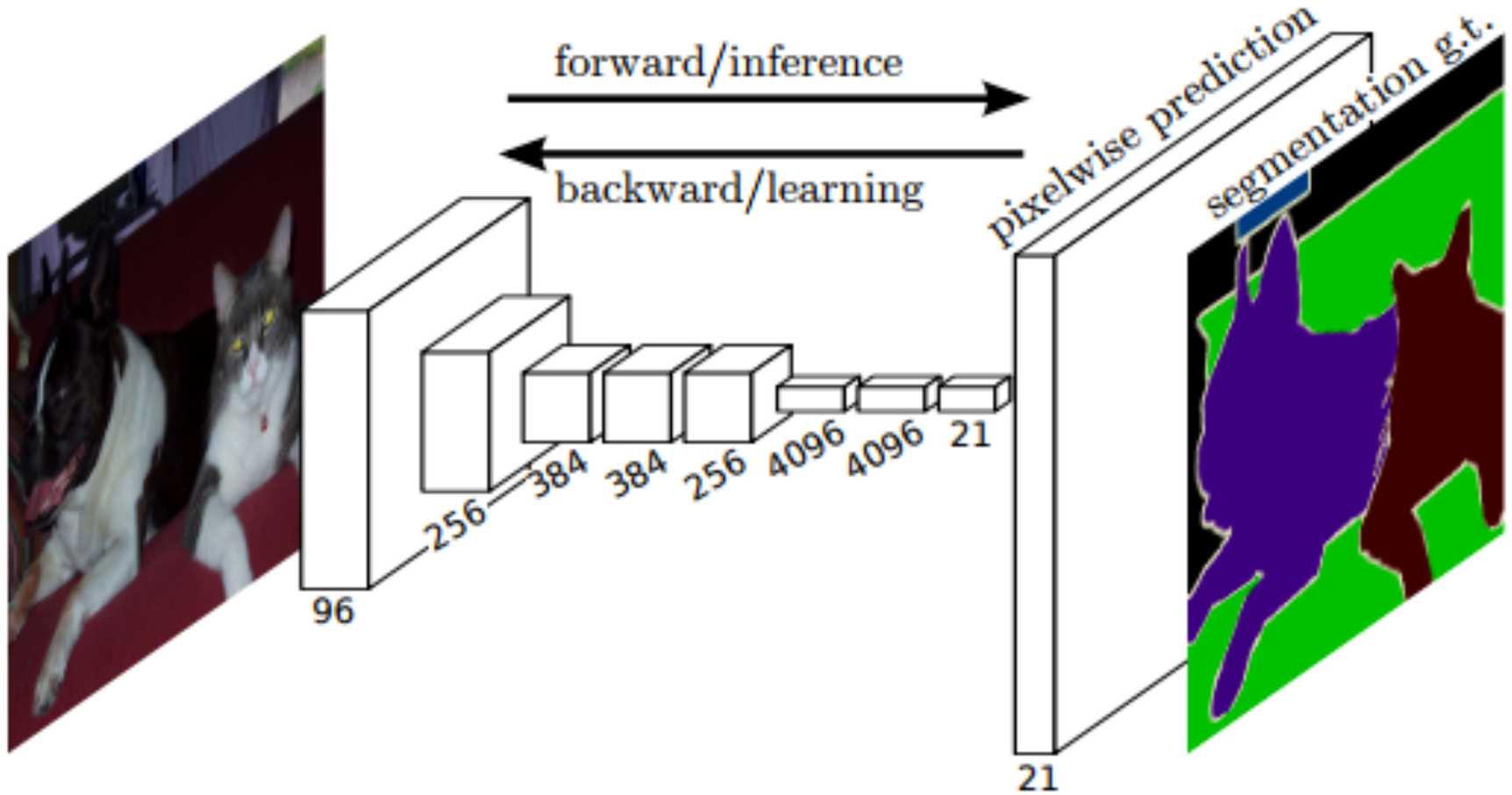


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Fully Convolution Network(FCN)

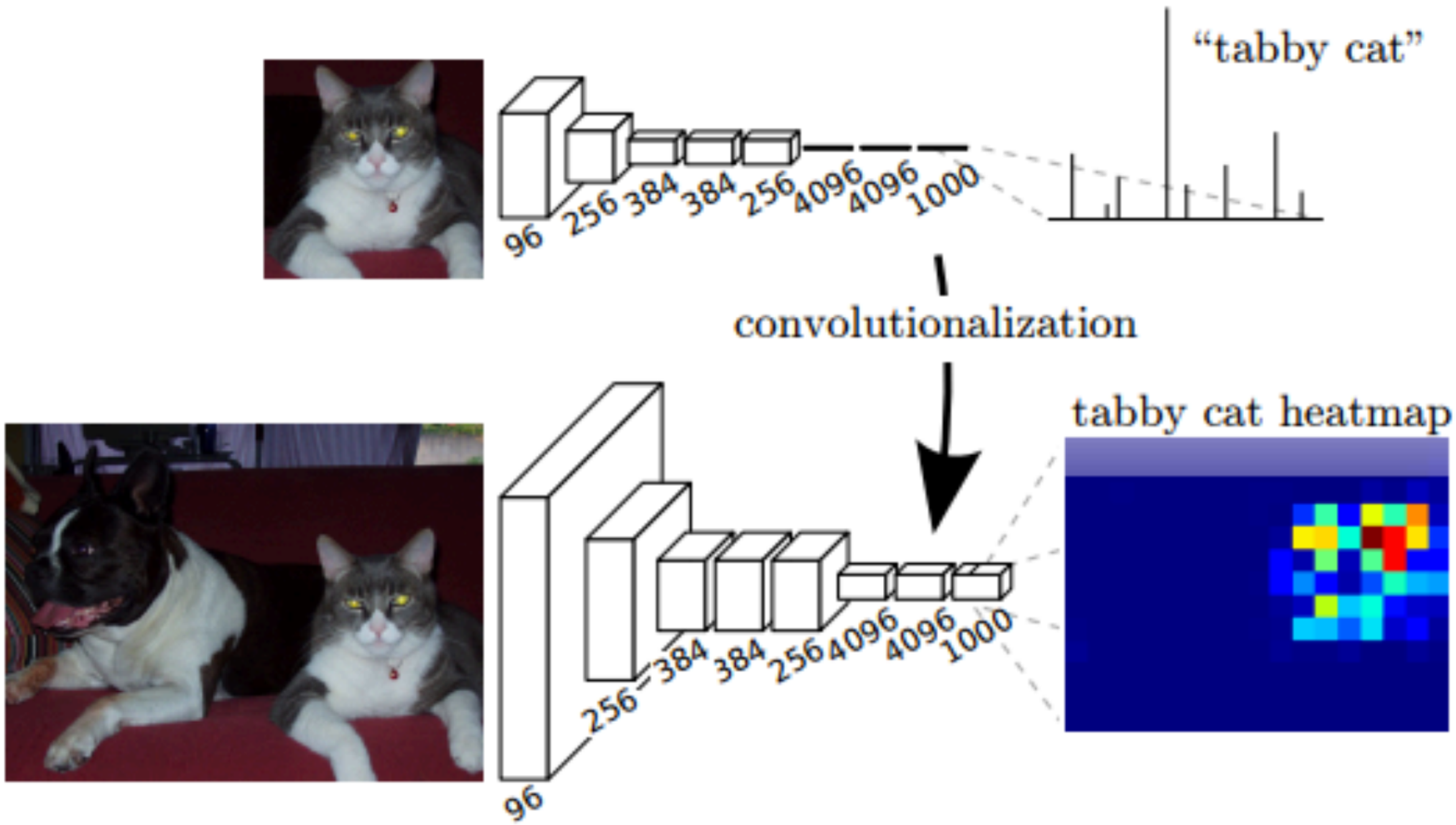
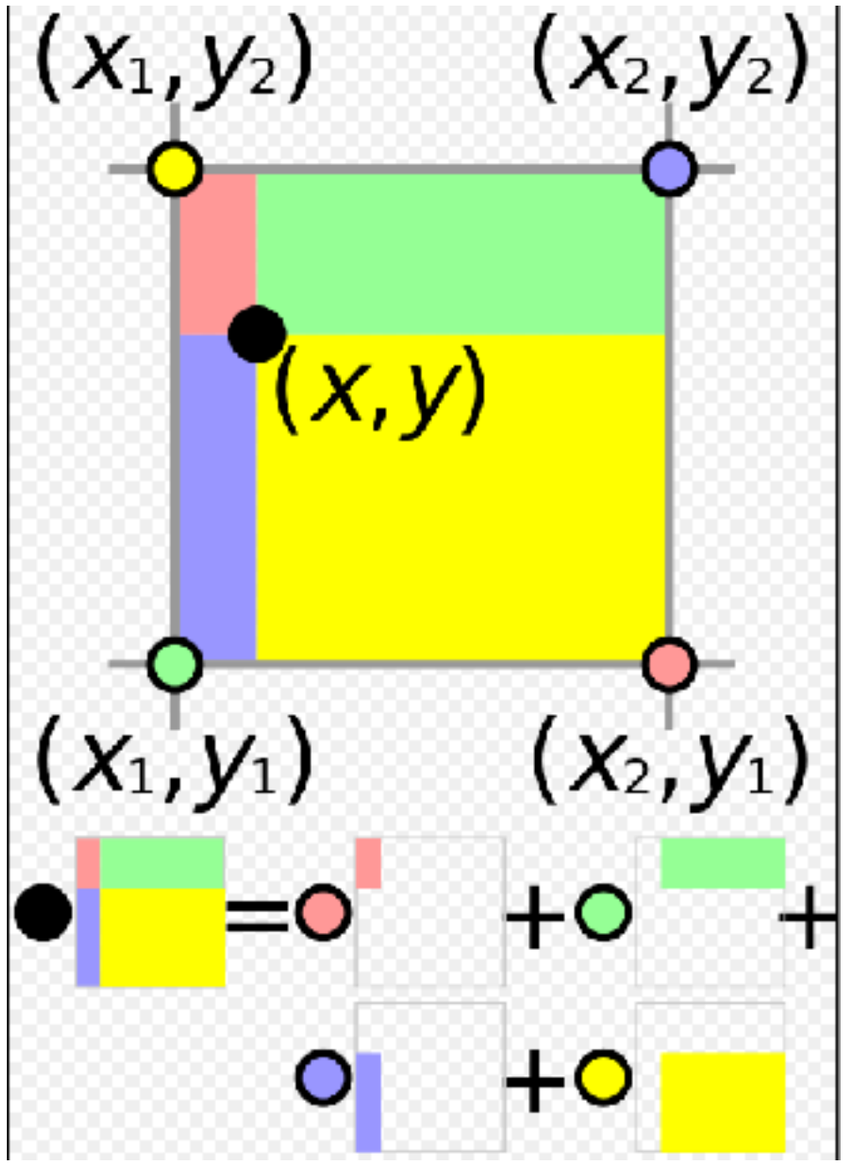


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Fully Convolution Network(FCN)



Bilinear Interpolation

- 줄어든 feature map 을 upsampling

- 학습 되지 않는 filter 로 interpolation

Fully Convolution Network(FCN)

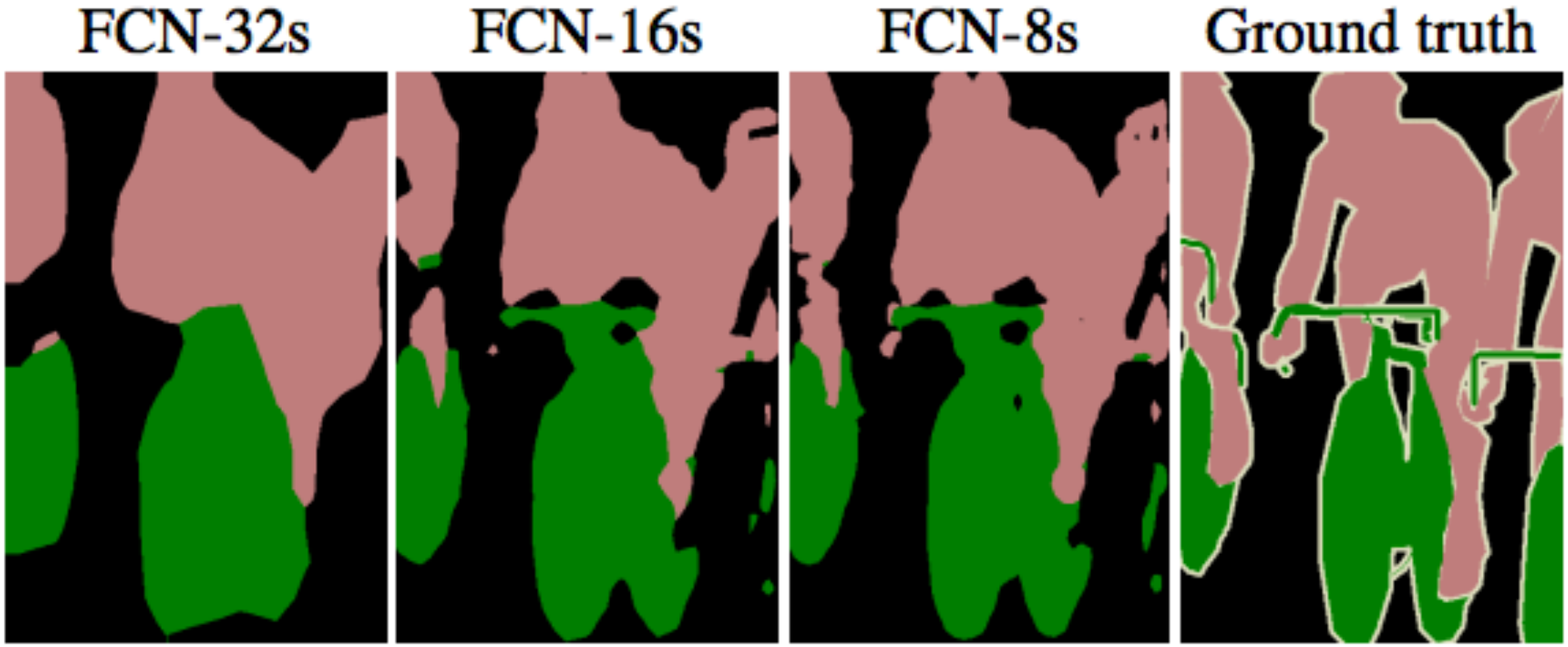


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

upsampling 이 많을 수록 부정확한 segmentation

Fully Convolution Network(FCN)

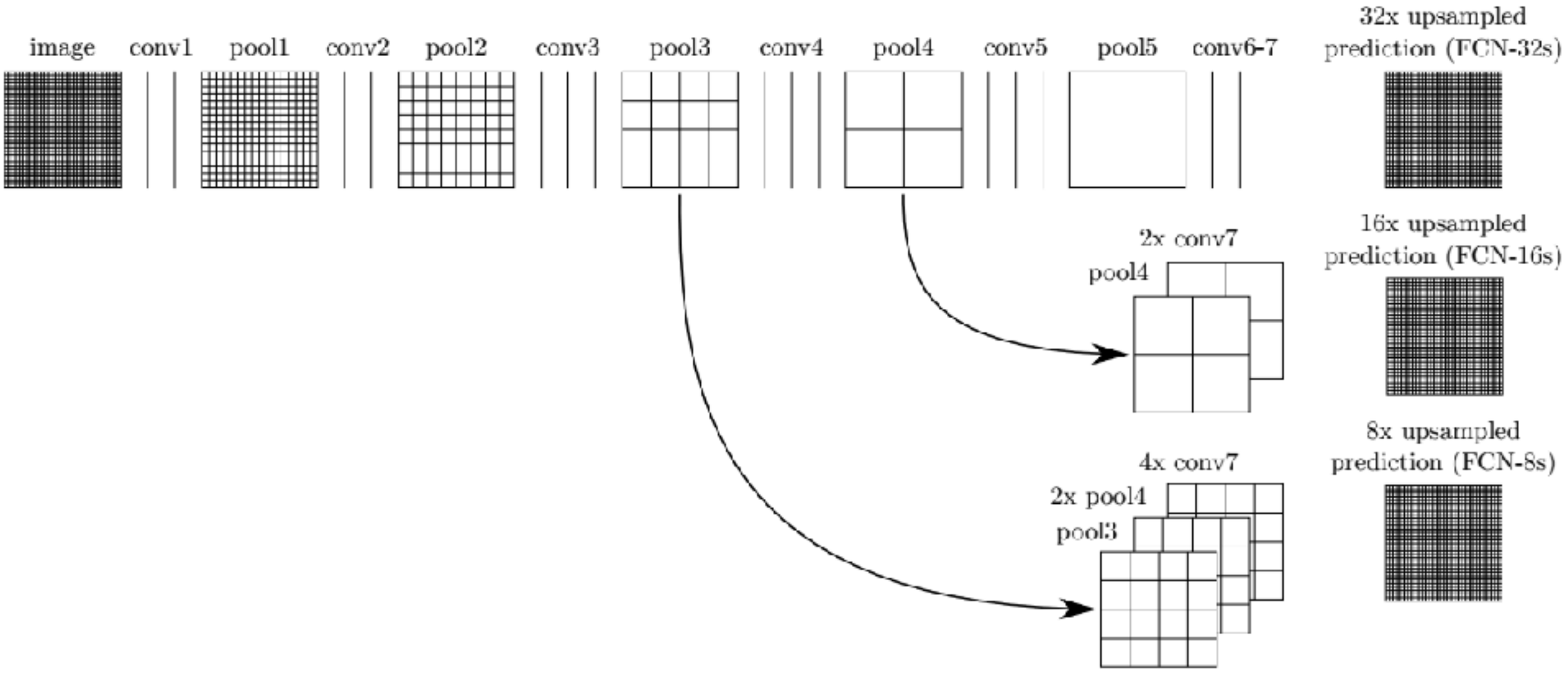
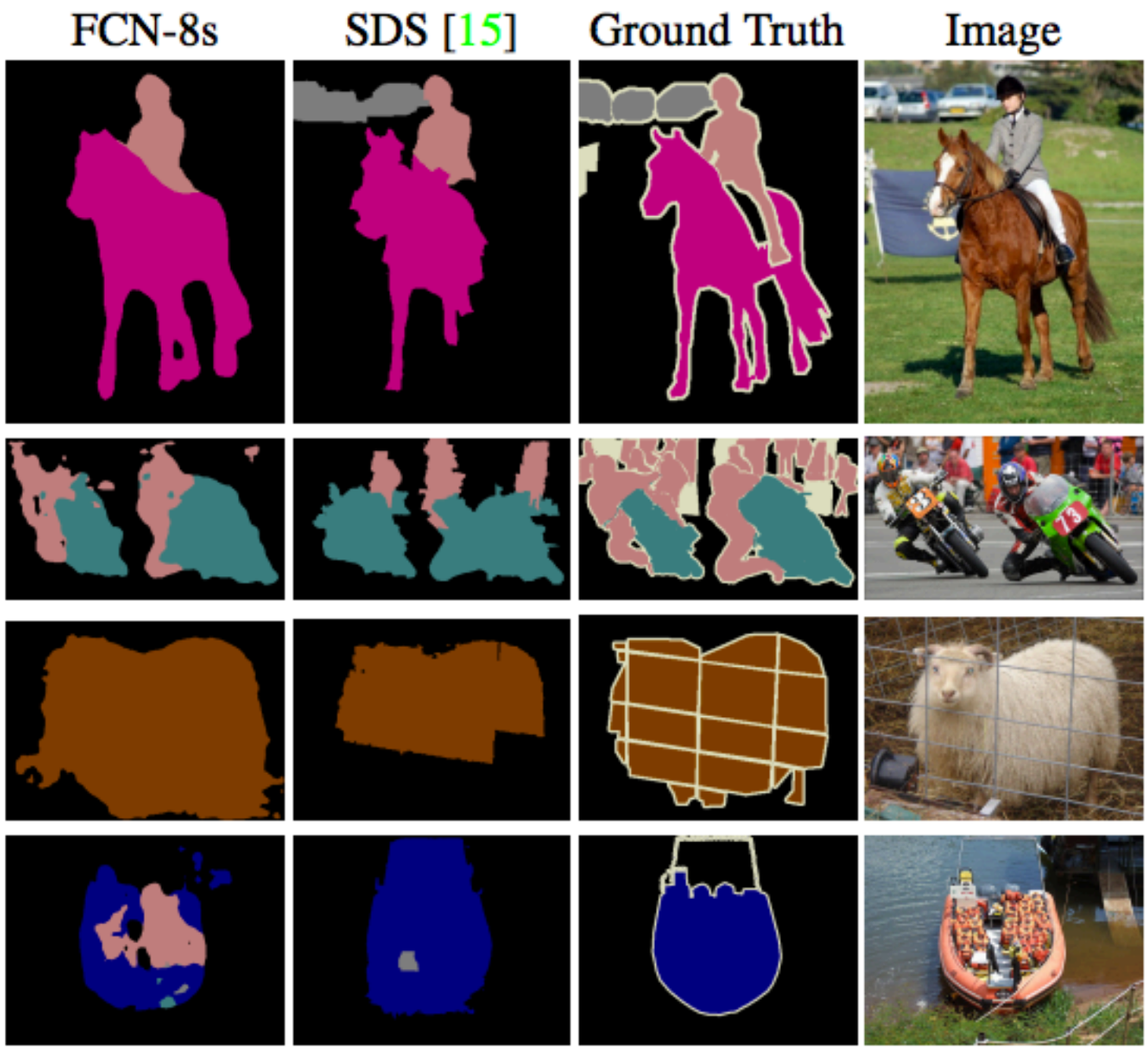


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Fully Convolution Network(FCN)



Deconvolution Network

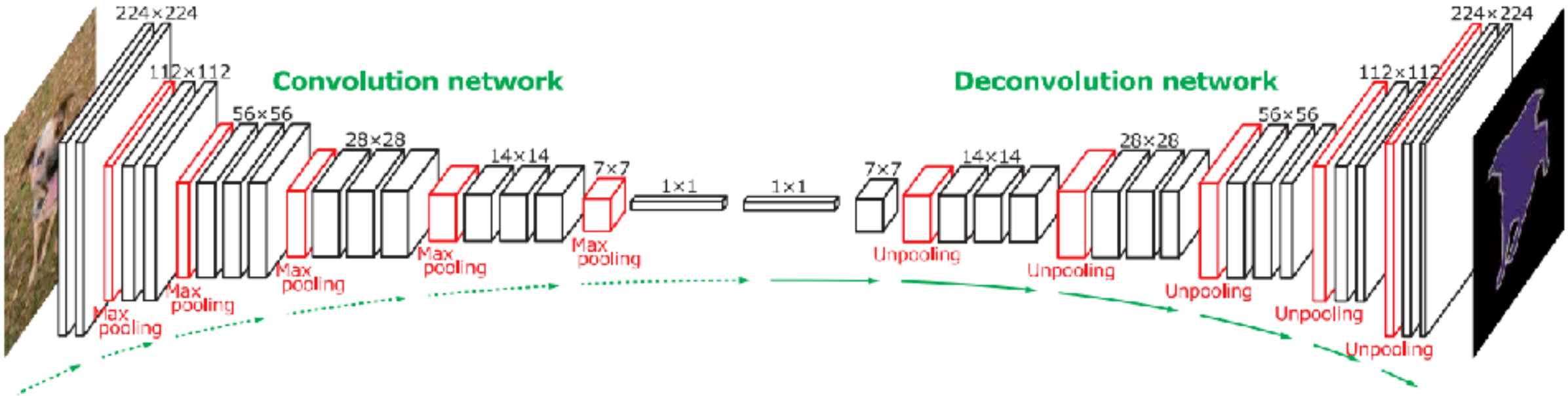


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

Deconvolution Network

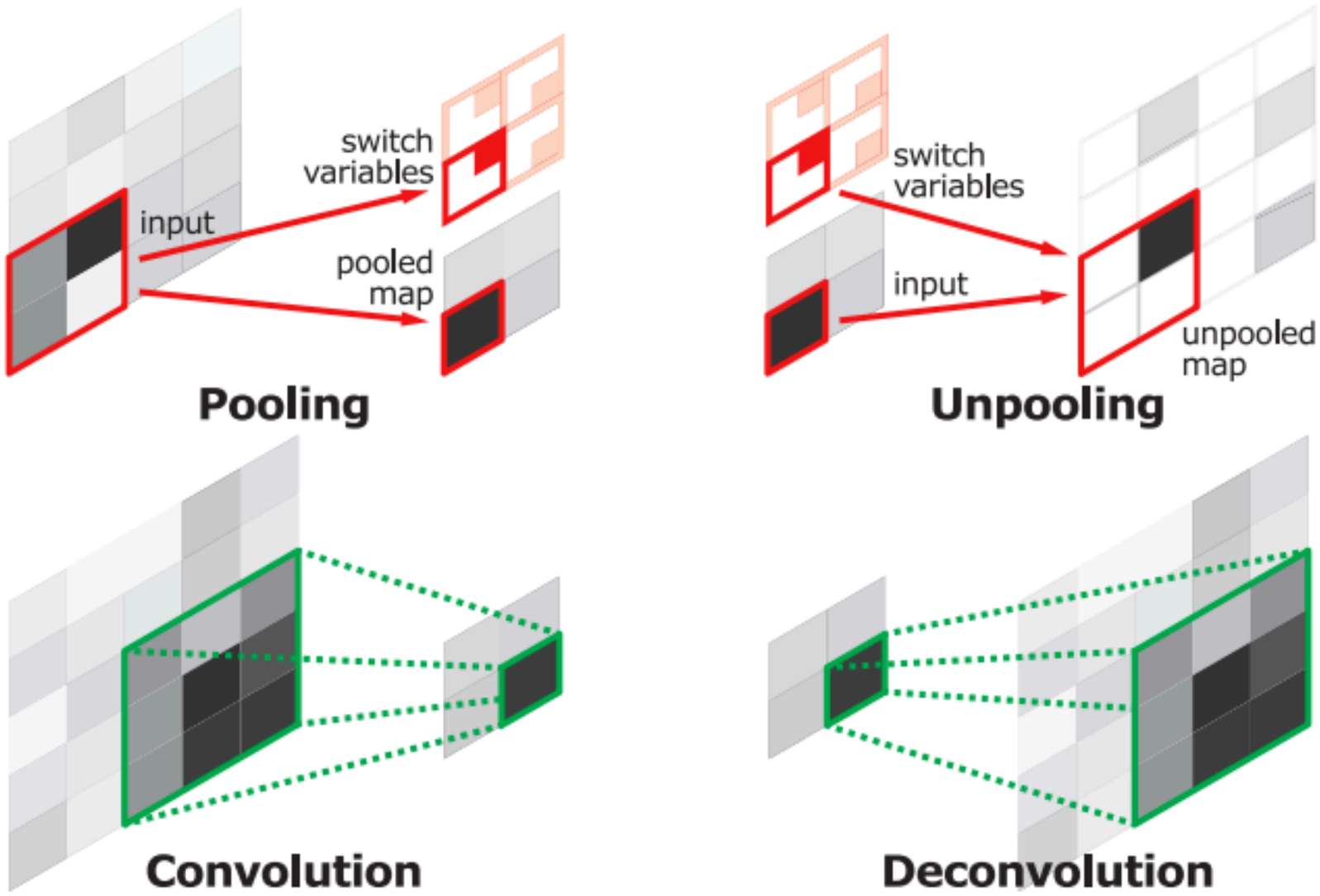


Figure 3. Illustration of deconvolution and unpooling operations.

Deconvolution Network

3.2.2 Deconvolution

The output of an unpooling layer is an enlarged, yet sparse activation map. The deconvolution layers densify the sparse activations obtained by unpooling through convolution-like operations with multiple learned filters. However, contrary to convolutional layers, which connect multiple input activations within a filter window to a single activation, deconvolutional layers associate a single input activation with multiple outputs, as illustrated in Figure 3. The output of the deconvolutional layer is an enlarged *and* dense activation map. We crop the boundary of the enlarged activation map to keep the size of the output map identical to the one from the preceding unpooling layer.

Deconvolution Network

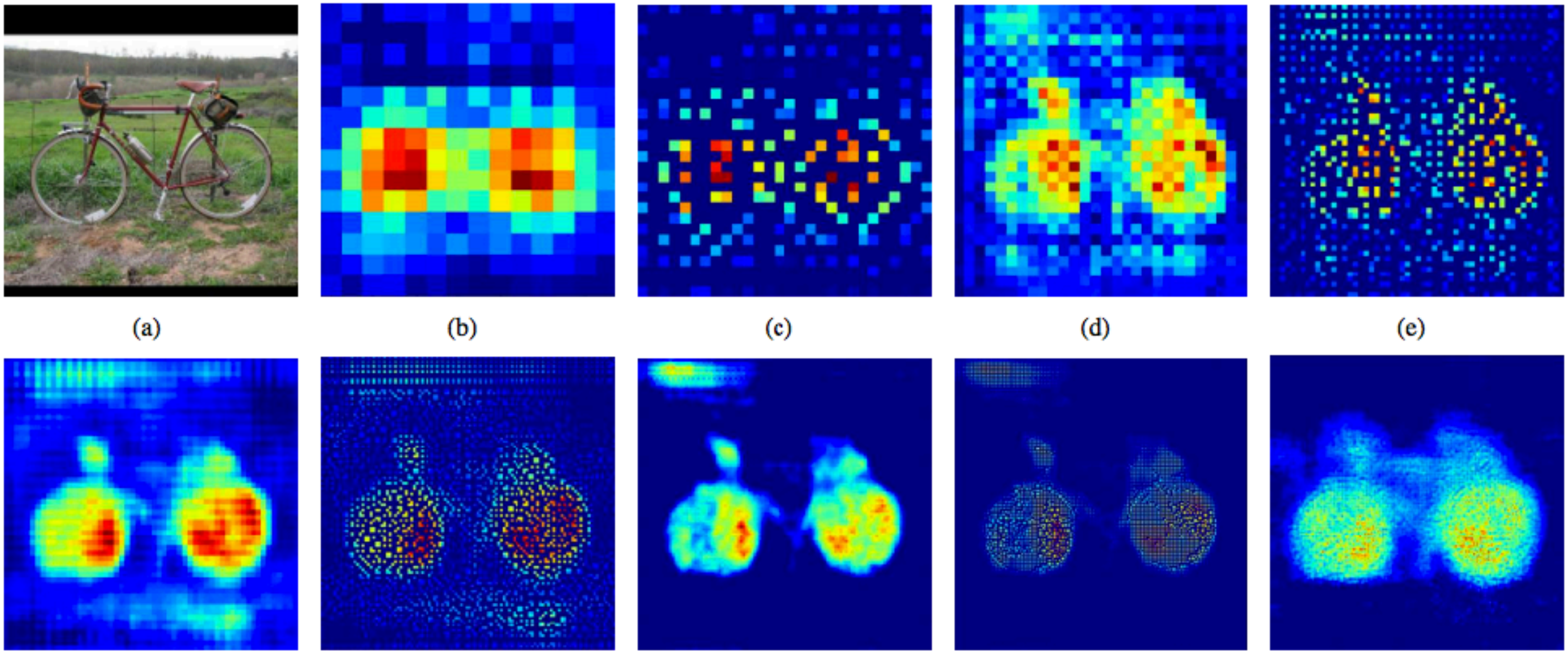
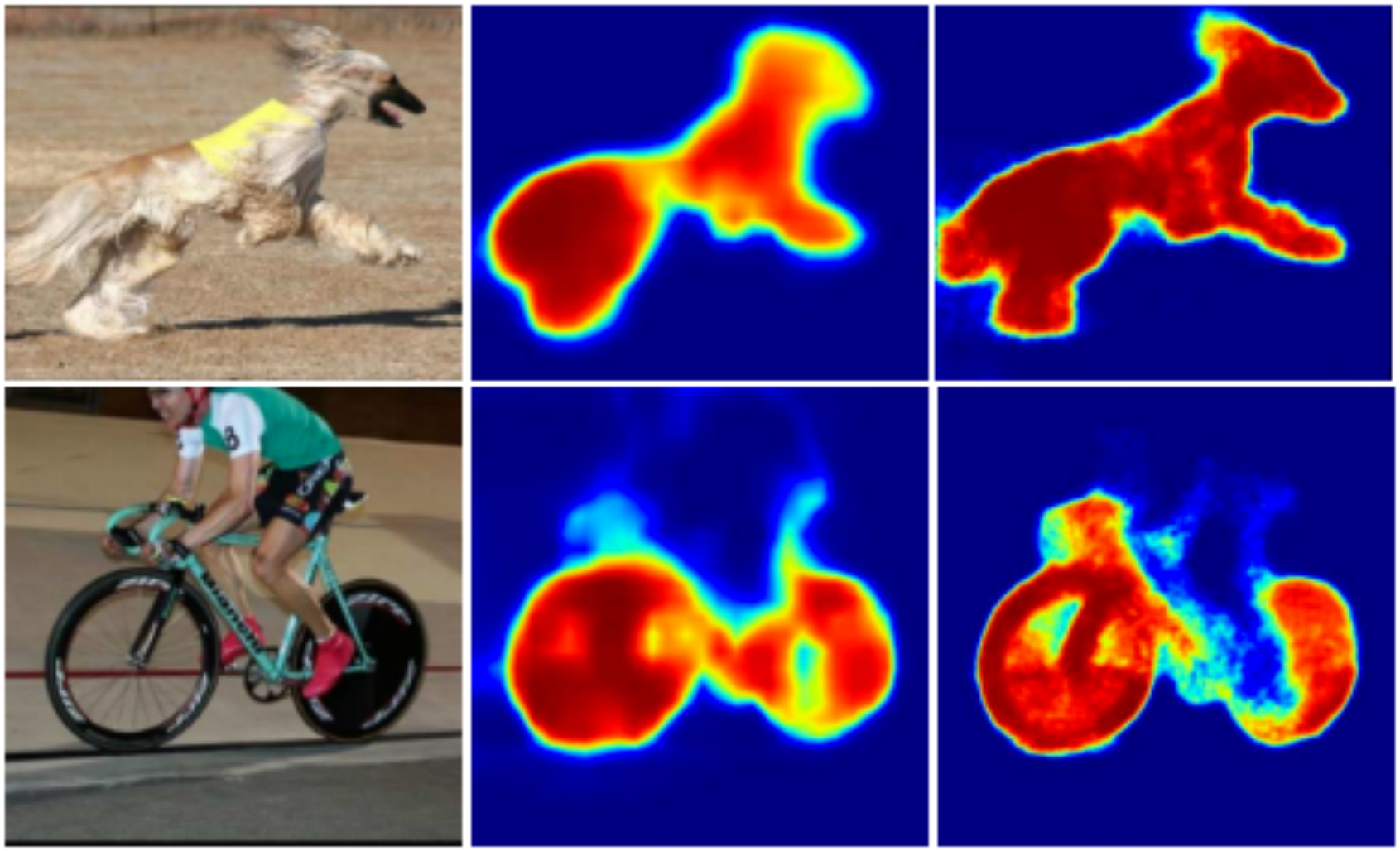


Figure 4. Visualization of activations in our deconvolution network. The activation maps from (b) to (j) correspond to the output maps from lower to higher layers in the deconvolution network. We select the most representative activation in each layer for effective visualization. The image in (a) is an input, and the rest are the outputs from (b) the last 14×14 deconvolutional layer, (c) the 28×28 unpooling layer, (d) the last 28×28 deconvolutional layer, (e) the 56×56 unpooling layer, (f) the last 56×56 deconvolutional layer, (g) the 112×112 unpooling layer, (h) the last 112×112 deconvolutional layer, (i) the 224×224 unpooling layer and (j) the last 224×224 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.

Deconvolution Network



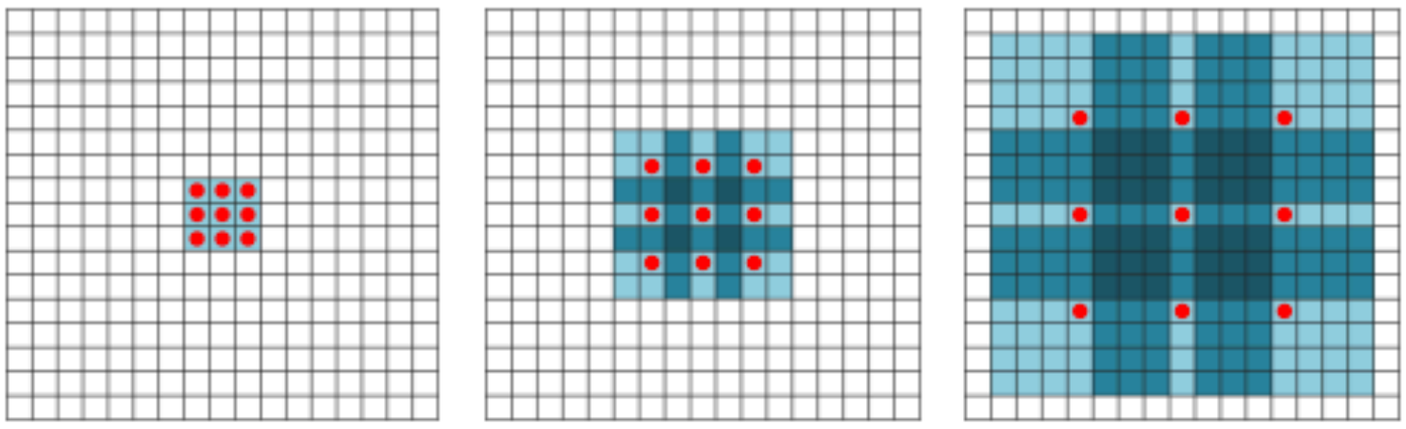
(a) Input image (b) FCN-8s (c) Ours

Figure 5. Comparison of class conditional probability maps from FCN and our network (top: dog, bottom: bicycle).

Dilated Convolution(Atrous Convolution)

4.2.2 *Dilated Convolutions*

Dilated convolutions, also named *à-trous* convolutions, are a generalization of Kronecker-factored convolutional filters [96] which support exponentially expanding receptive fields without losing resolution. In other words, dilated convolutions are regular ones that make use of upsampled filters. The dilation rate l controls that upsampling factor. As shown in Figure 12, stacking l-dilated convolution makes the receptive fields grow exponentially while the number of parameters for the filters keeps a linear growth. This means that dilated convolutions allow efficient dense feature extraction on any arbitrary resolution. As a side note, it is important to remark that typical convolutions are just 1-dilated convolutions.



(a) 1-dilated (b) 2-dilated (c) 3-dilated

Dilated Convolution(Atrous Convolution)

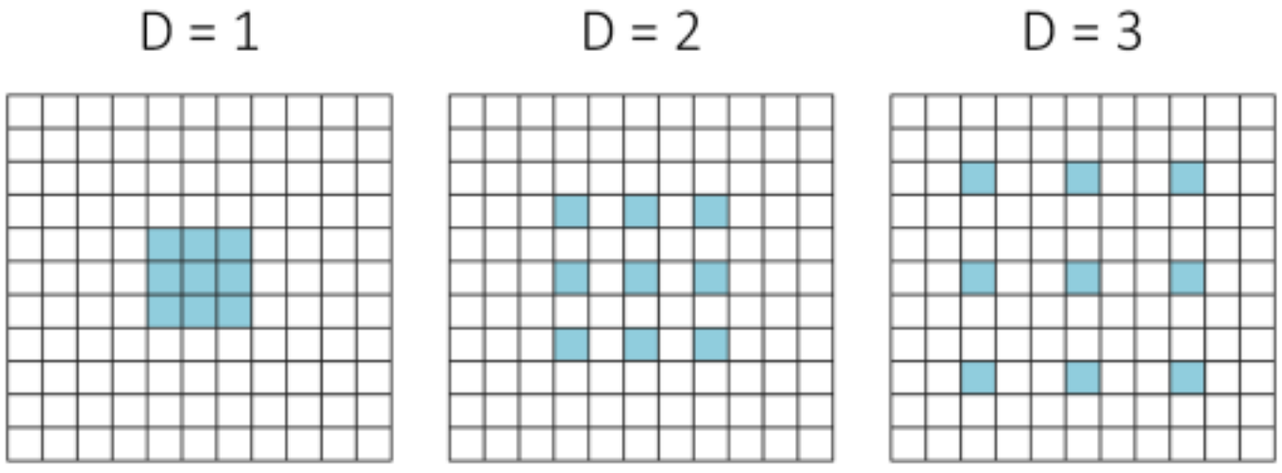
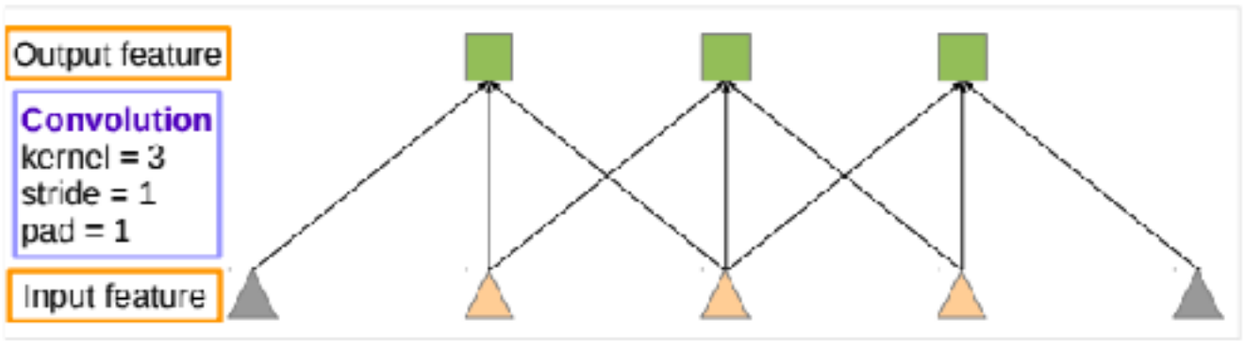
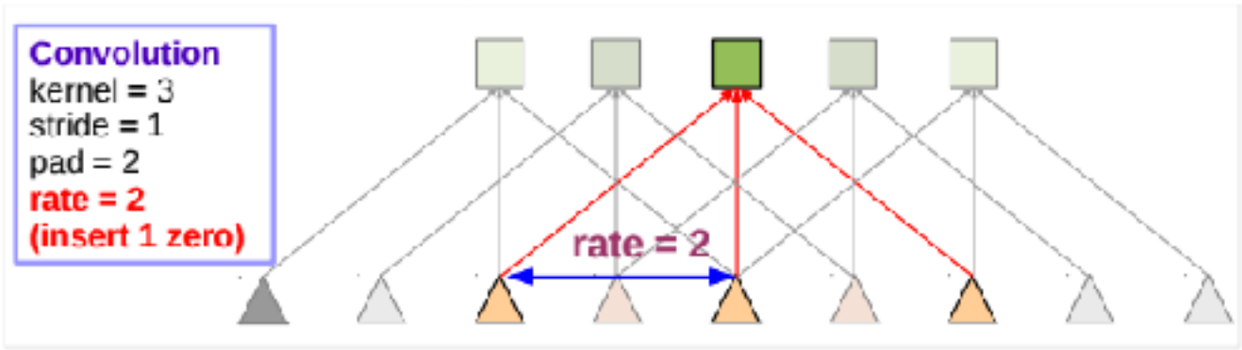


Fig. 13: Filter elements (green) matched to input elements when using 3×3 dilated convolutions with various dilation rates. From left to right: 1, 2, and 3.



(a) Sparse feature extraction



(b) Dense feature extraction

Dilated Convolution(Atrous Convolution)

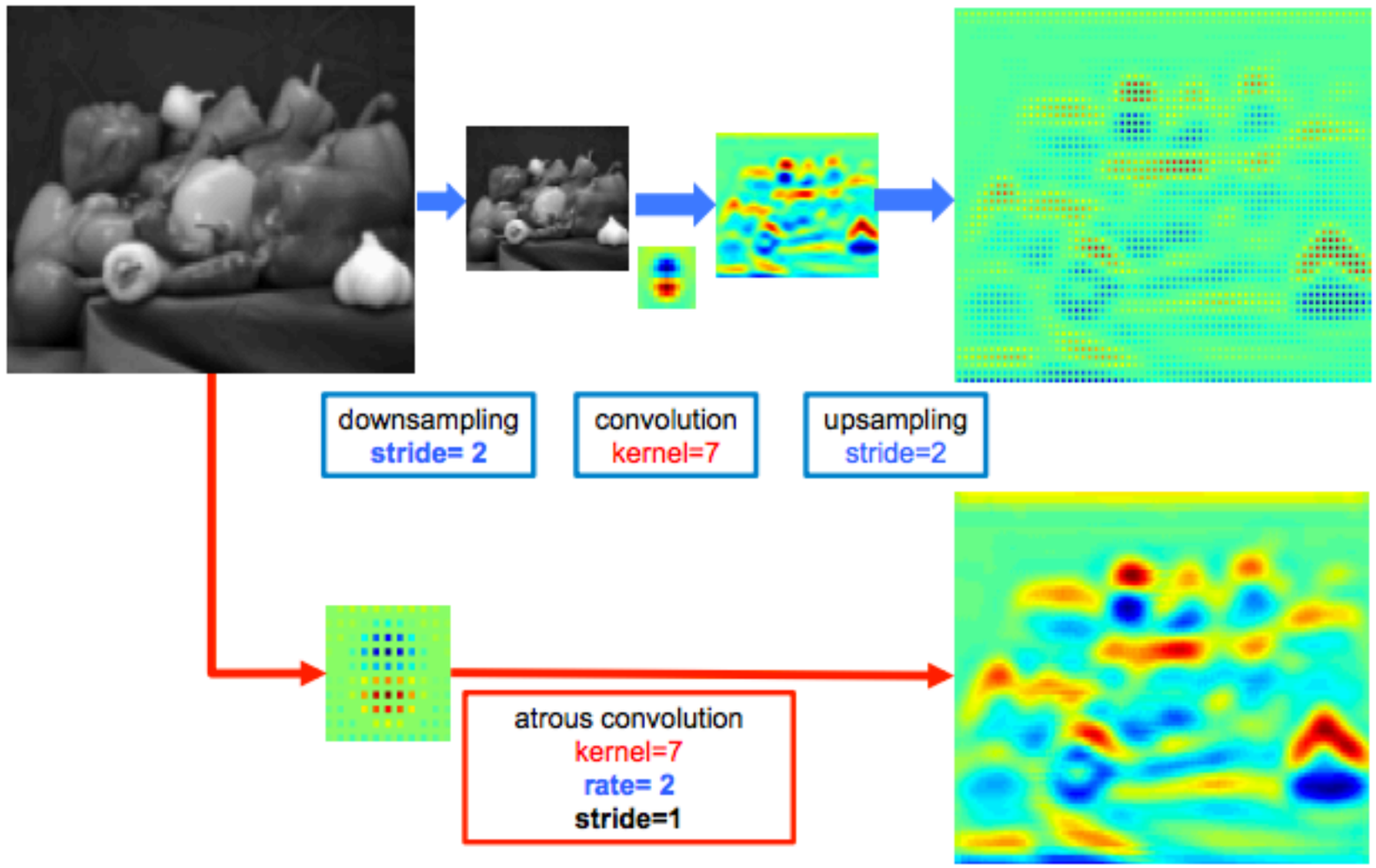


Fig. 3: Illustration of atrous convolution in 2-D. Top row: sparse feature extraction with standard convolution on a low resolution input feature map. Bottom row: Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

U-Net

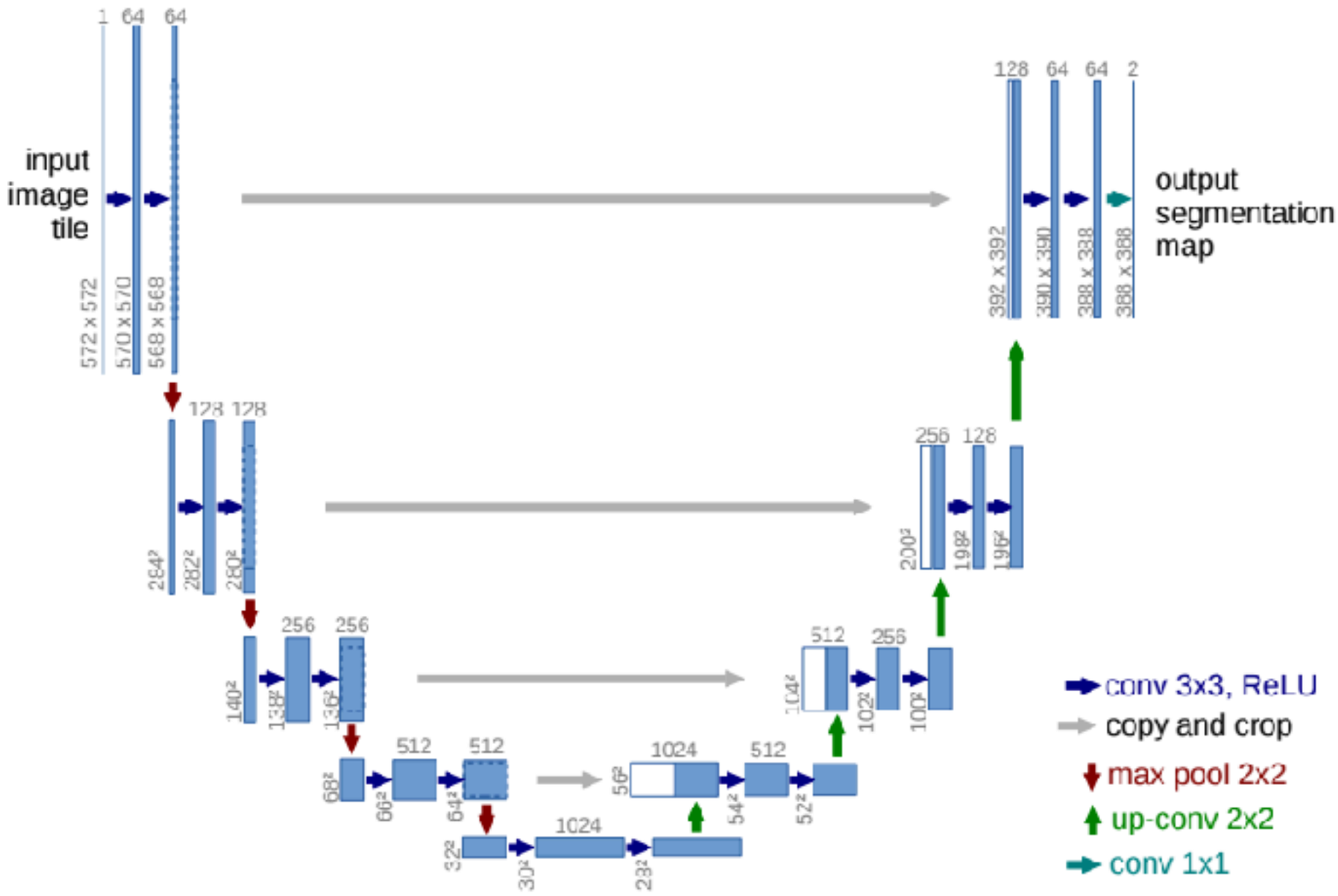
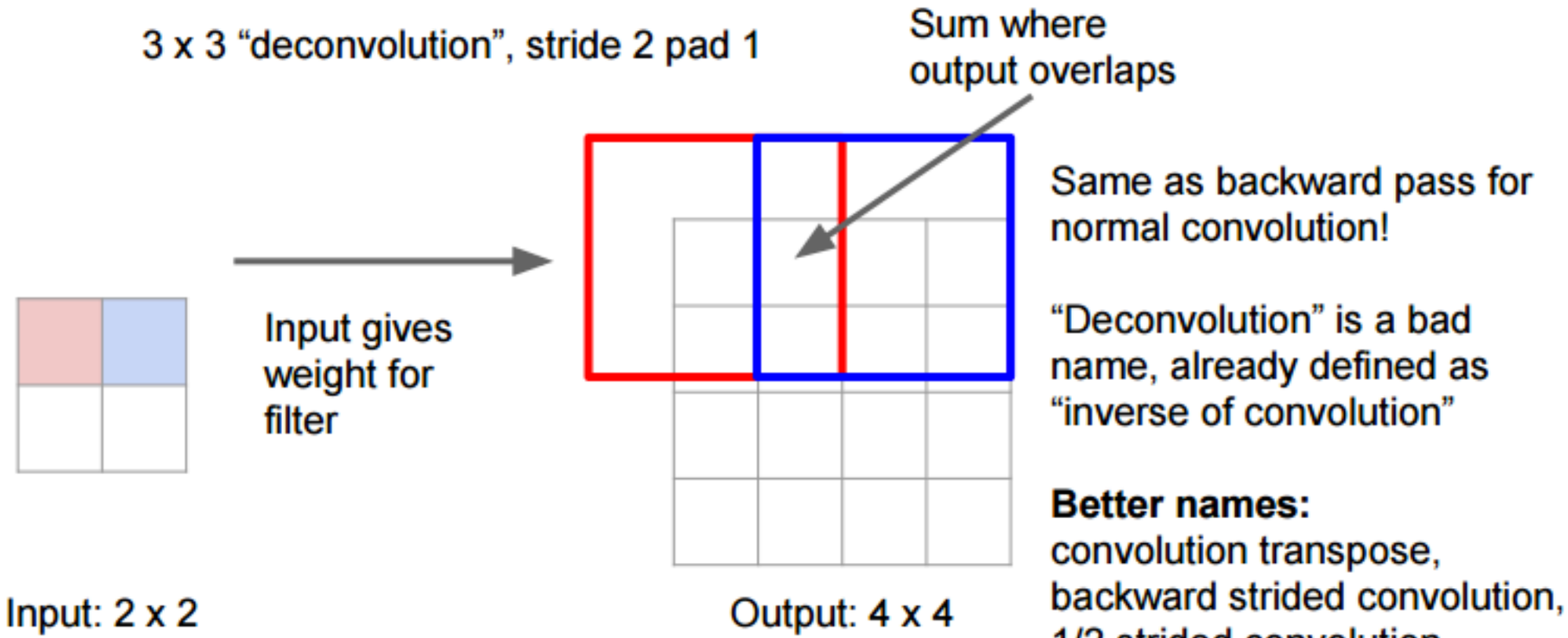


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

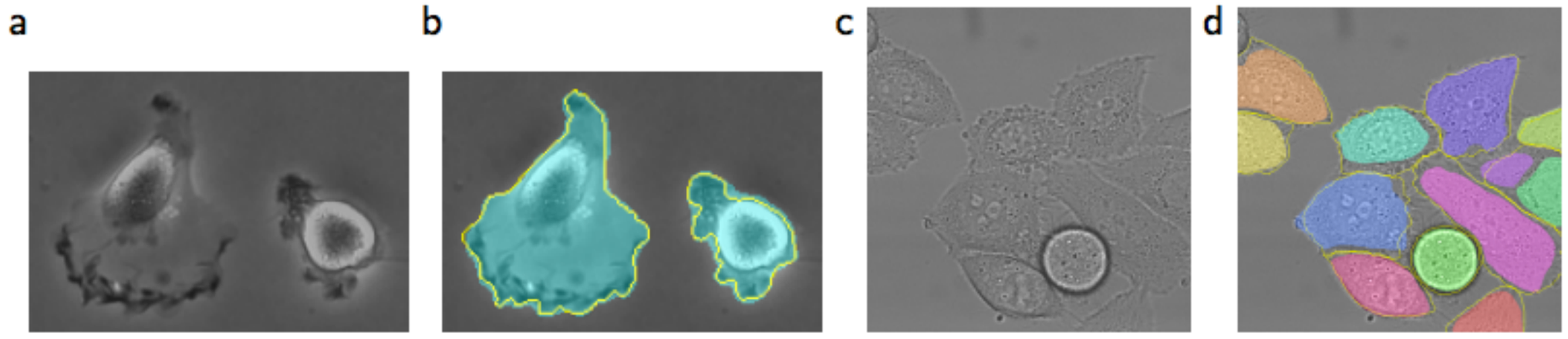
U-Net

Not Using unpooling(only up-convolution)

Learnable Upsampling: “Deconvolution”



U-Net



Conditional Random Fields(CRF)

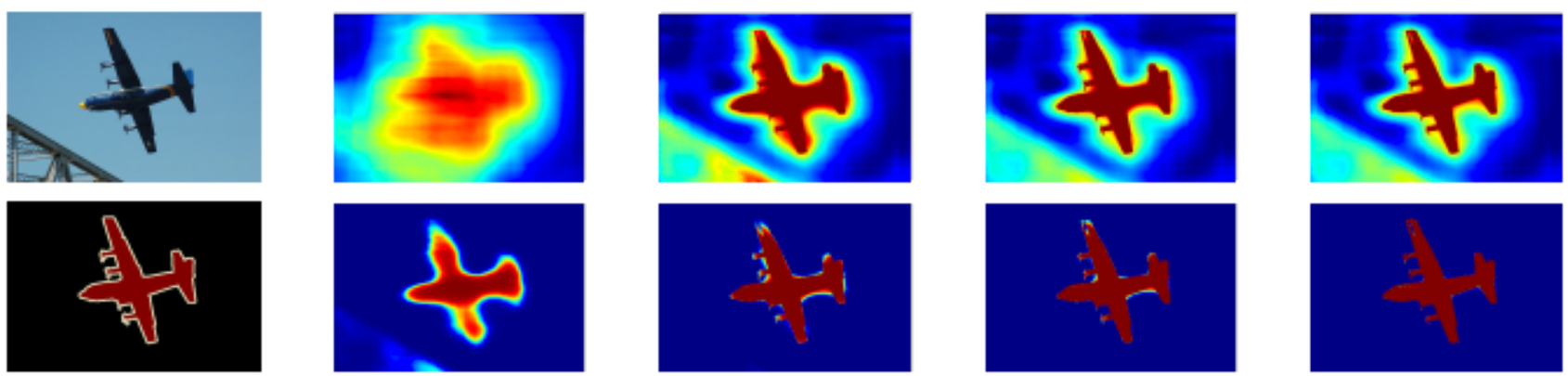
레이블의 인접성에 대한 정보를 바탕으로 레이블을 추측하는 기계학습 기법이다.

4.2.1 Conditional Random Fields

As we mentioned before, the inherent invariance to spatial transformations of CNN architectures limits the very same spatial accuracy for segmentation tasks. One possible and common approach to refine the output of a segmentation system and boost its ability to capture fine-grained details is to apply a post-processing stage using a Conditional Random Field (CRF). CRFs enable the combination of low-level image information – such as the interactions between pixels [92] [93] – with the output of multi-class inference systems that produce per-pixel class scores. That combination is especially important to capture long-range dependencies, which CNNs fail to consider, and fine local details.

Conditional Random Fields(CRF)

Traditionally, conditional random fields (CRFs) have been employed to smooth noisy segmentation maps [23], [31]. Typically these models couple neighboring nodes, favoring same-label assignments to spatially proximal pixels. Qualitatively, the primary function of these short-range CRFs is to clean up the spurious predictions of weak classifiers built on top of local hand-engineered features.



Image/G.T. DCNN output CRF Iteration 1 CRF Iteration 2 CRF Iteration 10

Conditional Random Fields(CRF)

레이블의 인접성에 대한 정보를 바탕으로 레이블을 추측하는 기계학습 기법이다.