Fully Convolutional Networks for Semantic Segmentation

Jonathan Long, Evan Shelhamer, Trevor Darrell

Presenter: Hannah Li

Convolutional Networks

Fully Convolutional Networks

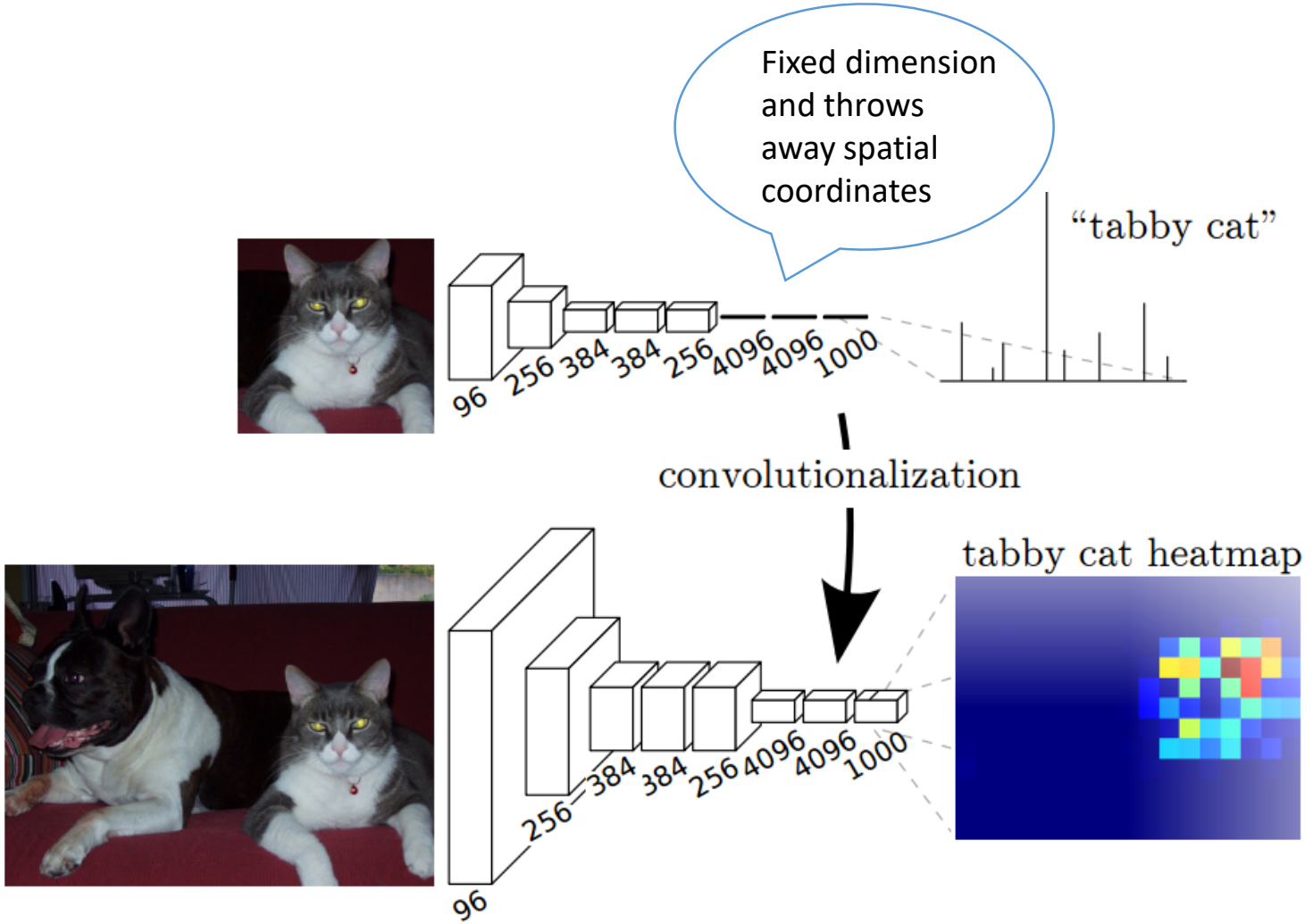
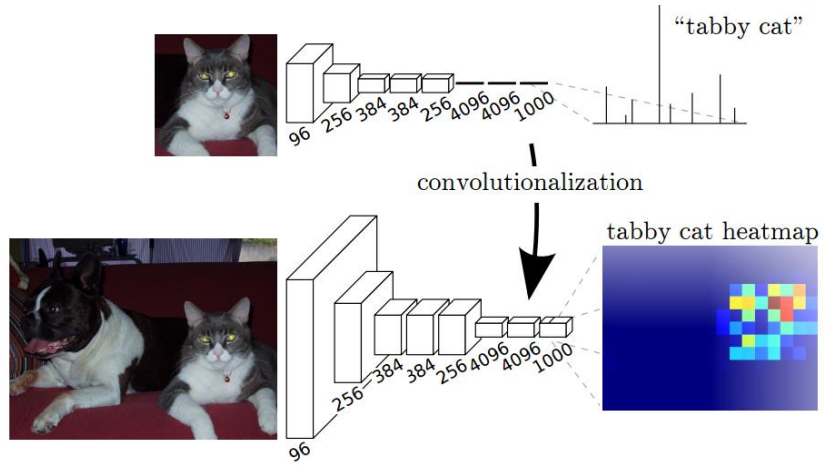


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Fully Convolutional Networks



$$\mathbf{y}_{ij} = f_{ks}\left(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \le \delta i, \delta j \le k}\right)$$

$\boldsymbol{x}_{i,j}$ - location (i, j) for a particular layer

$\boldsymbol{y}_{i,j}$ - location (i, j) for the *following* layer

k - kernel (filter) size

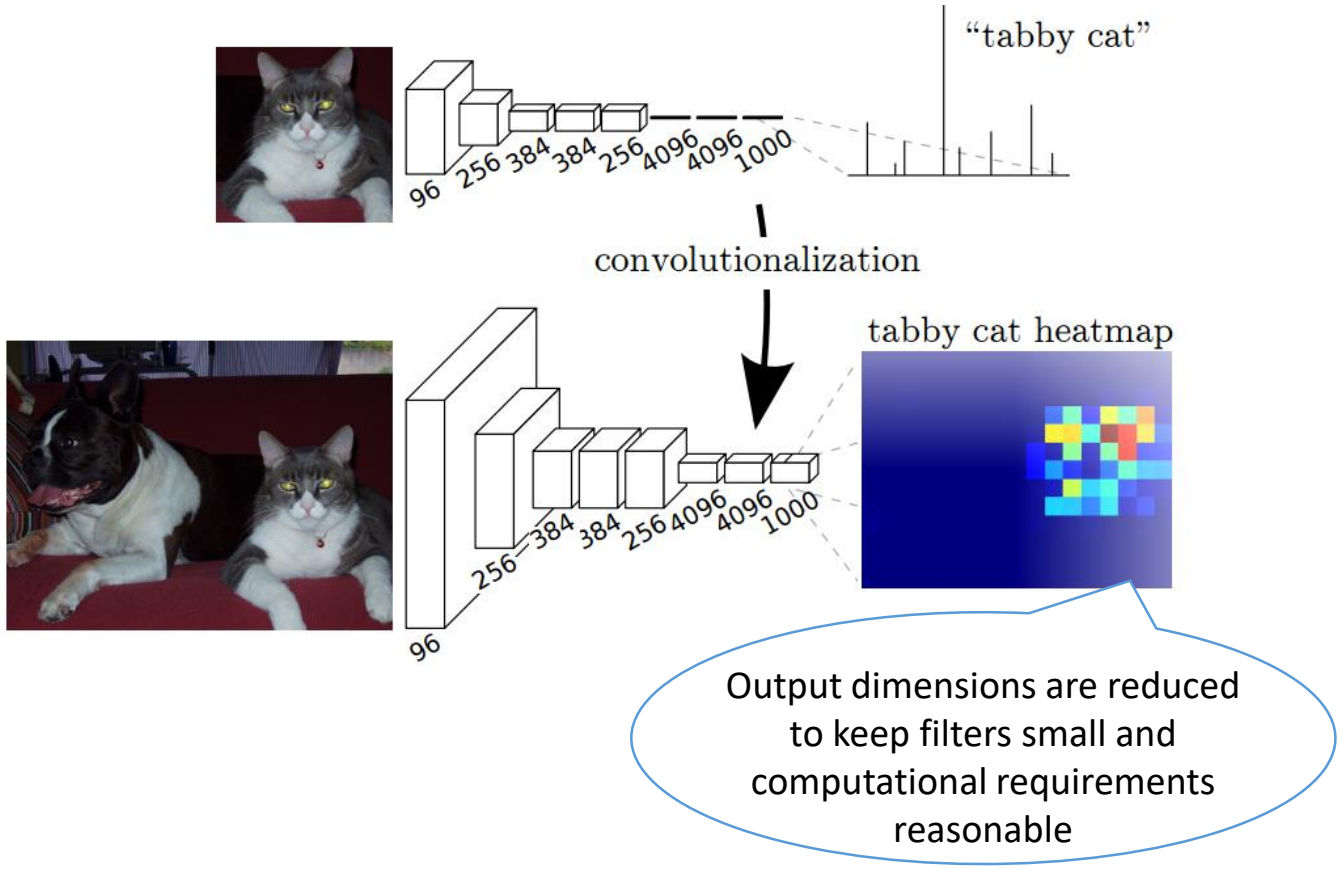
s - stride

$\boldsymbol{f}_{ks}$ - determines the layer type:

- matrix multiplication for convolution or average pooling
- spatial max for max pooling
- elementwise nonlinearity for an activation function

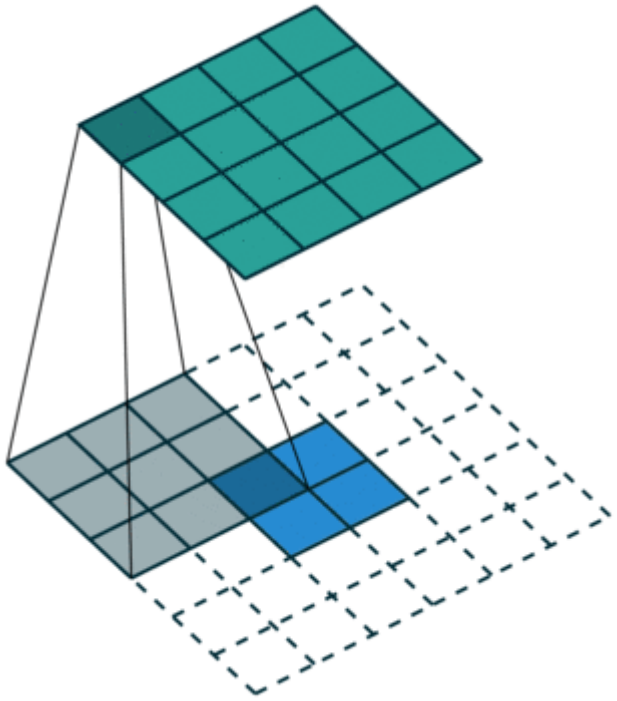
$$\ell(\mathbf{x}; \theta) = \sum_{ij} \ell'(\mathbf{x}_{ij}; \theta)$$

To find the gradient descent on the whole image, you only need to find the gradient descent of the final layer

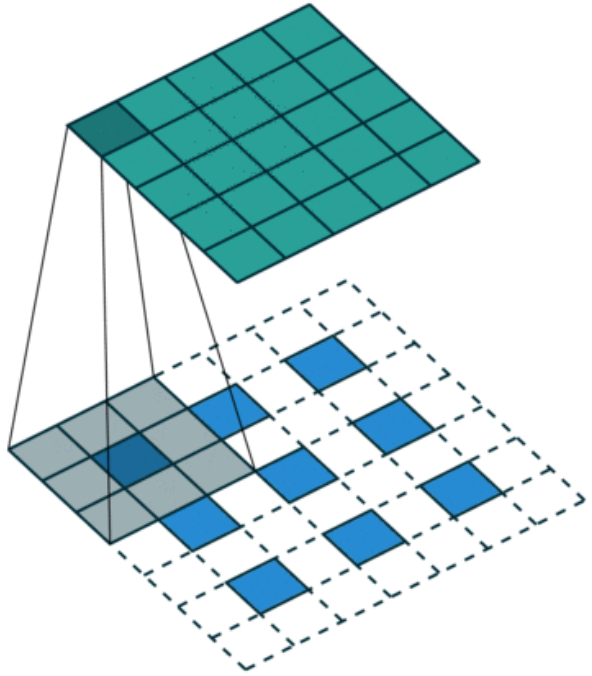


However, we need dense pixels…

Solution: Deconvolution/Upsampling

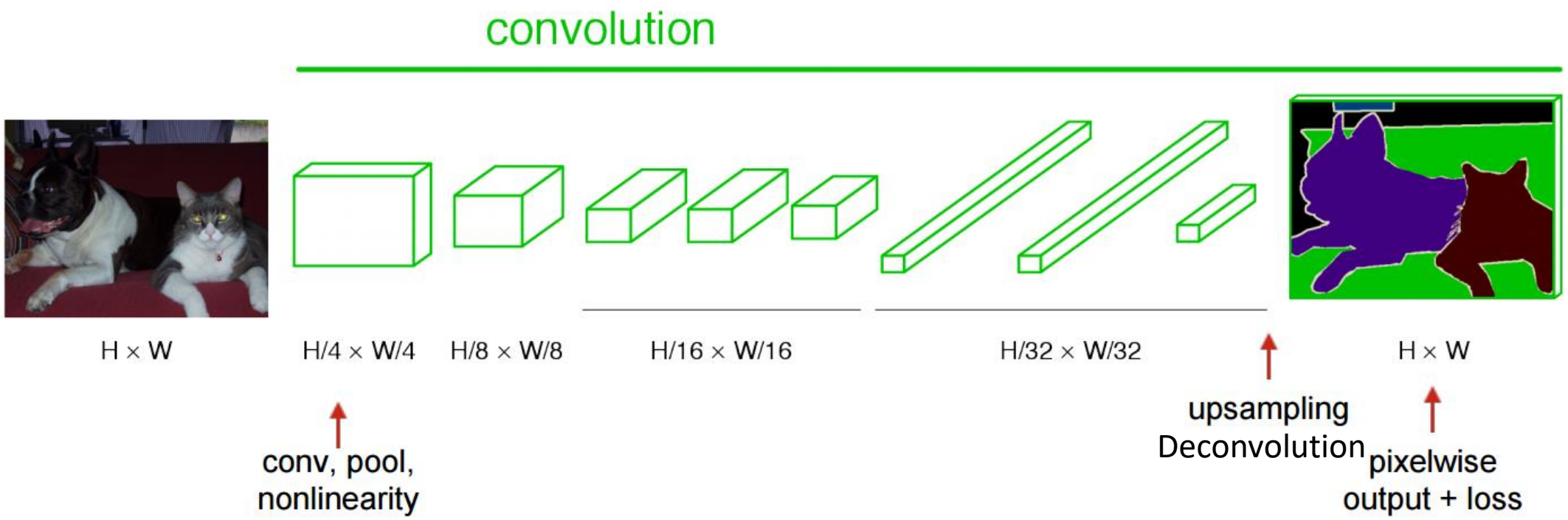


stride = 1



stride = 2

https://github.com/vdumoulin/conv_arithmetic



Bilinear interpolation

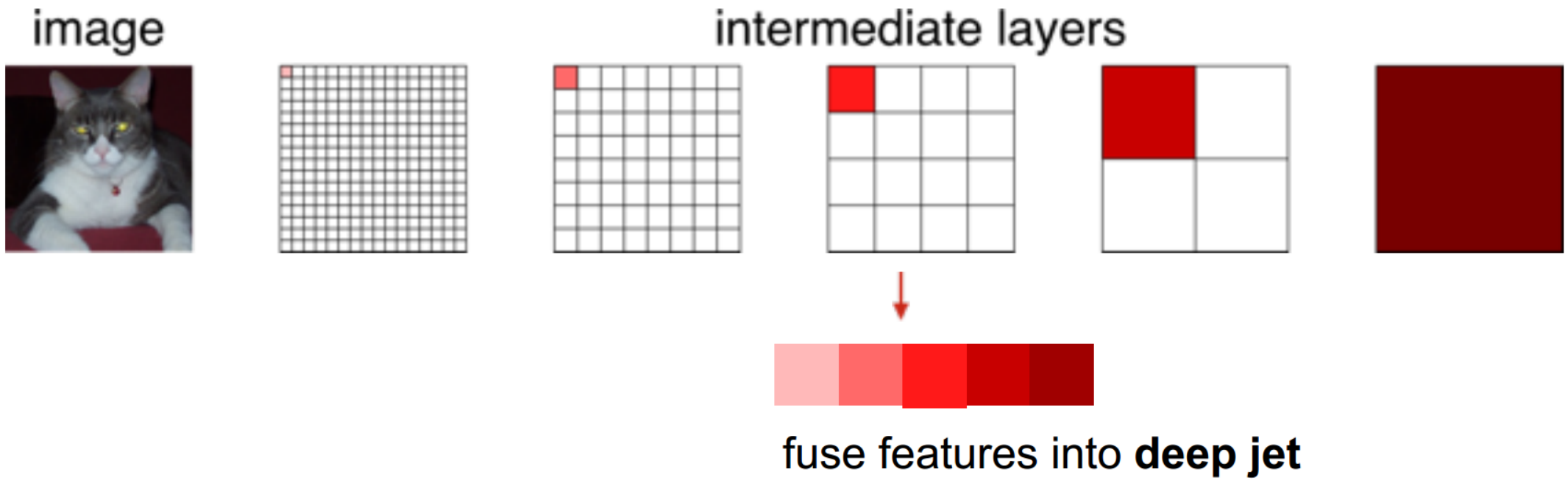
http://tutorial.caffe.berkeleyvision.org/caffe-cvpr15-pixels.pdf

From classifier to dense FCN

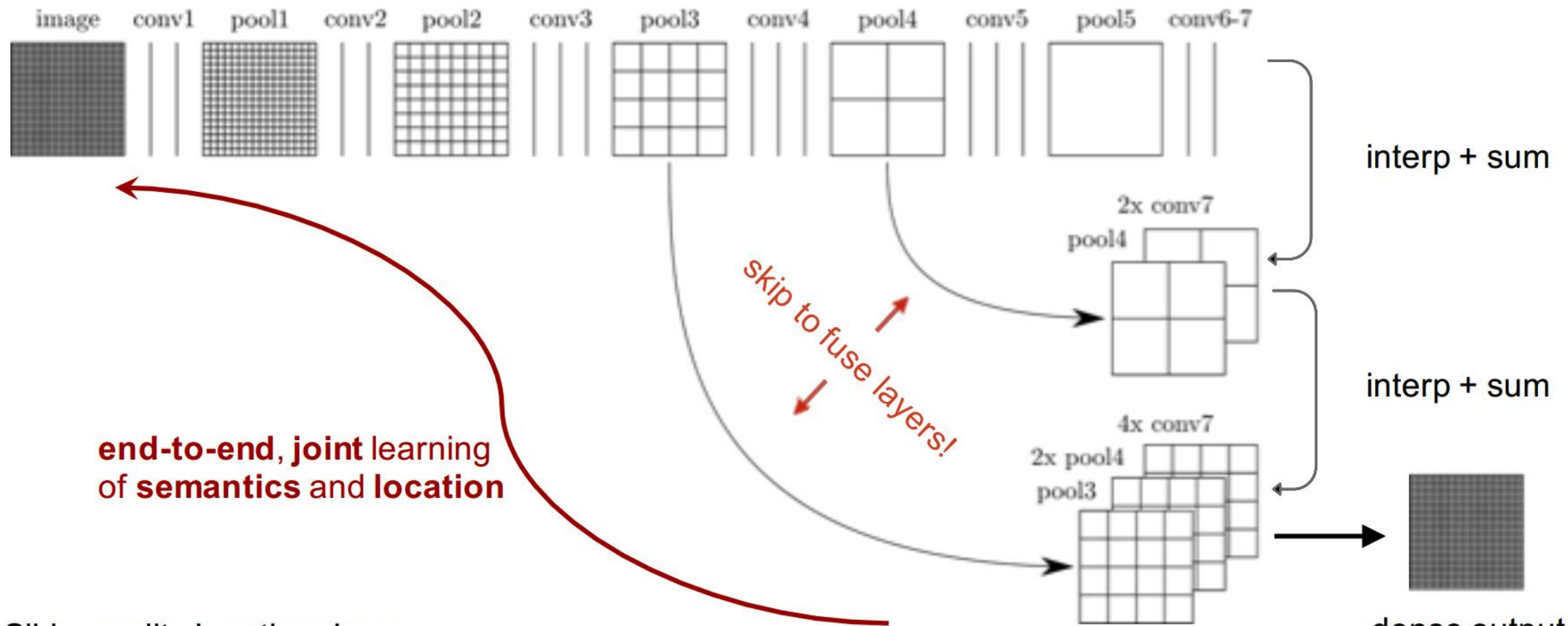
1. Discard the final layer
2. Convert all fully connected layers to convolutions
3. Append convolution with channel dimension 21 to predict score for the PASCAL classes

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

combine *where* (local, shallow) with *what* (global, deep)



(cf. Hariharan et al. CVPR15 “hypercolumn”)



Slide credit: Jonathan Long

FCN for segmentation

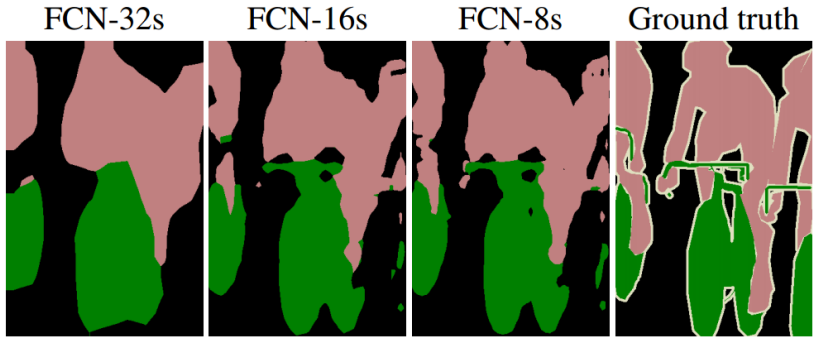
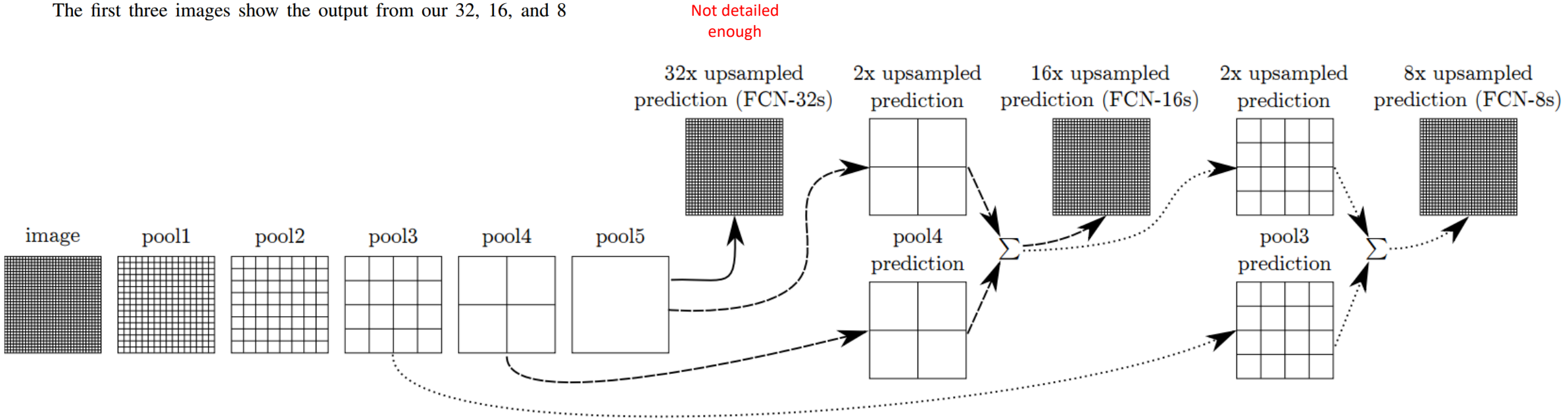
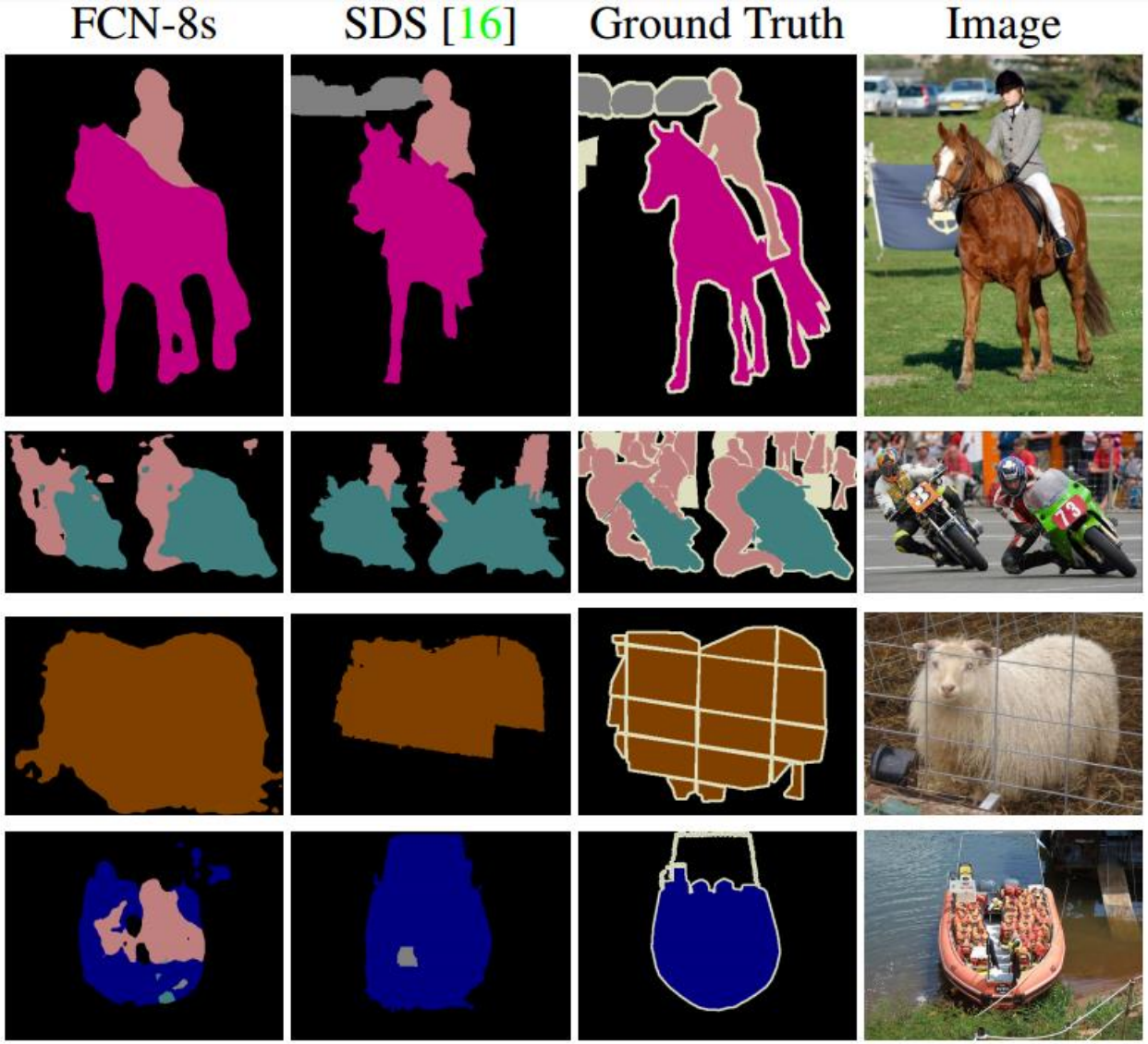


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8

Technique: combining final prediction layer with lower layers with finer strides

Combining fine layers and coarse layers lets the model make local predictions that respect global structure.





Results

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
* SDS [16]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

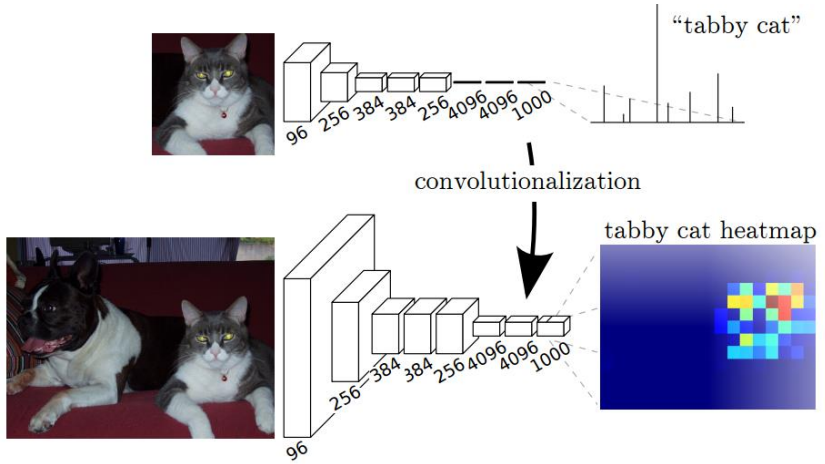
20% improvement for mean IoU

286 times faster

*Simultaneous detection and segmentation. ECCV 2014

Conclusions

- “Fully convolutional” networks take inputs of any size
- Adapt AlexNet, VGG net, GoogLeNet into fully convolutional networks
- Fine-tune them to the segmentation task
- New architecture: combines semantic info from coarse layer with info from shallow, fine layer
- 20% relative improvement to 62.2% mean IU (PASCAL VOC 2012)

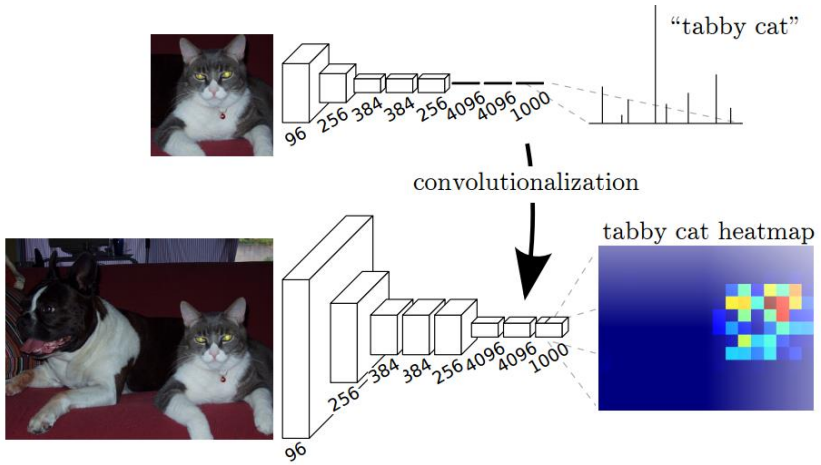


Extras

Fully Convolutional Networks

$$\mathbf{y}_{ij} = f_{ks}\left(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \le \delta i, \delta j \le k}\right)$$

- $\boldsymbol{x}_{i,j}$ - location (i, j) for a particular layer
- $\boldsymbol{y}_{i,j}$ - location (i, j) for the *following* layer
- k - kernel (filter) size
- s - stride
- $\boldsymbol{f}_{ks}$ - determines the layer type:
  - matrix multiplication for convolution or average pooling
  - spatial max for max pooling
  - elementwise nonlinearity for an activation function



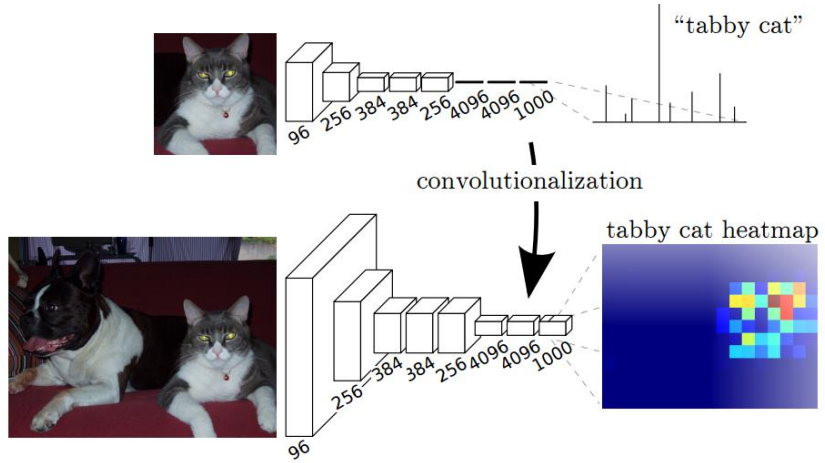
$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s', ss'}.$$

FCN naturally operates on an input of any size!

- Kernel size and stride for this functional form obeys the transformation rule above
- Computes a nonlinear filter (as opposed to a general nonlinear function)

Loss Function

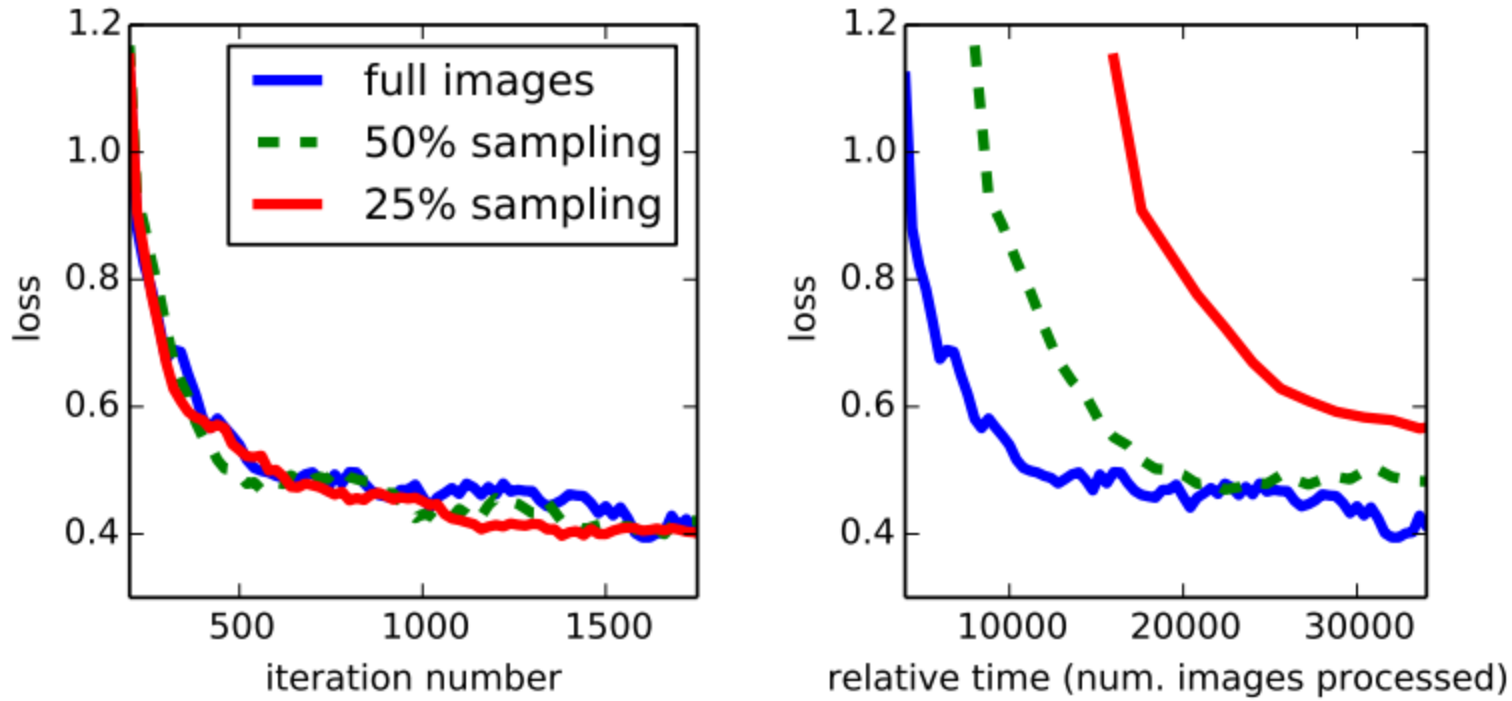
$$\ell(\mathbf{x};\theta) = \sum_{ij} \ell'(\mathbf{x}_{ij};\theta)$$



- Loss function is a sum over the spatial dimension of the final layer
- Gradient of loss function is a sum over the gradients of each of its spatial components
- Takes all of the final layer receptive fields as a minibatch

The receptive fields overlap significantly ->
More efficient!
5 times faster than AlexNet

Spatial output maps are suitable dense problems like semantic segmentation


1.2
1.0
0.8
0.6
0.4
loss
full images
50% sampling
25% sampling
500
1000
1500
iteration number
1.2
1.0
0.8
0.6
0.4
loss
10000
20000
30000
relative time (num. images processed)