Fully Convolutional Networks for Semantic Segmentation

Jonathan Long*   Evan Shelhamer*   Trevor Darrell

UC Berkeley

Chaim Ginzburg for Deep Learning seminar

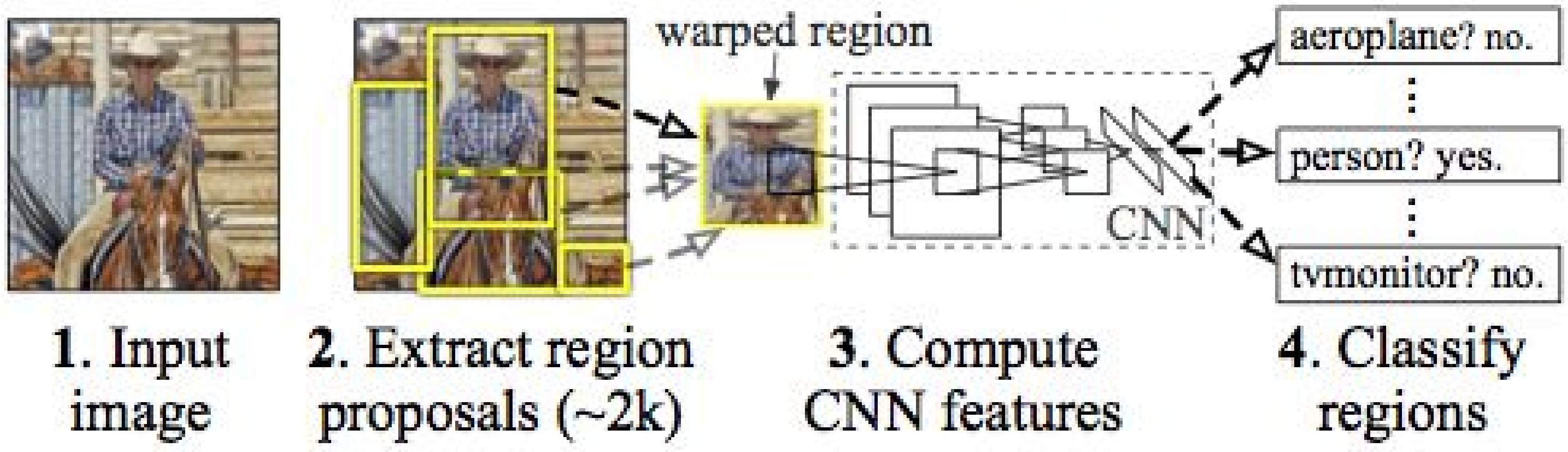
Semantic Segmentation

- Define a pixel-wise labeling for an image I as a set of random variables X = {x0, . . . , xn} n = #pixles. xi $\in$ L =labels {1, . . . ,m}.
- Use CNN to model a probability distribution Q(X|θ, I) over those random variables,


wall
book
vmonitor
table
cabine
floor
sofa

Problem - DCNN are great for WHAT but loose the WHERE

Naive Approach : Region-CNN

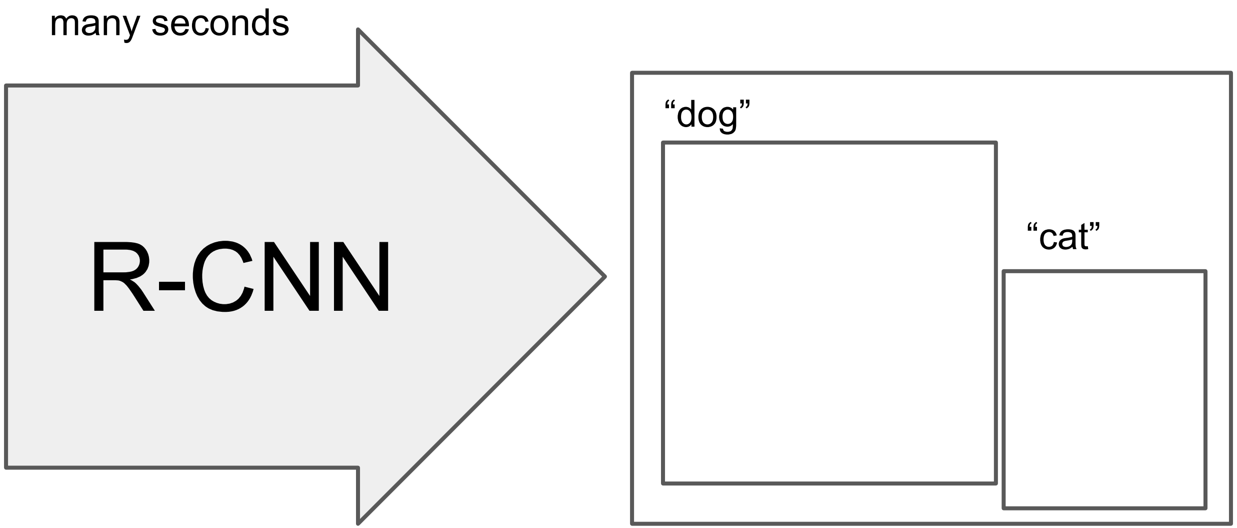


“Selective Search”

SVM trained for specific class

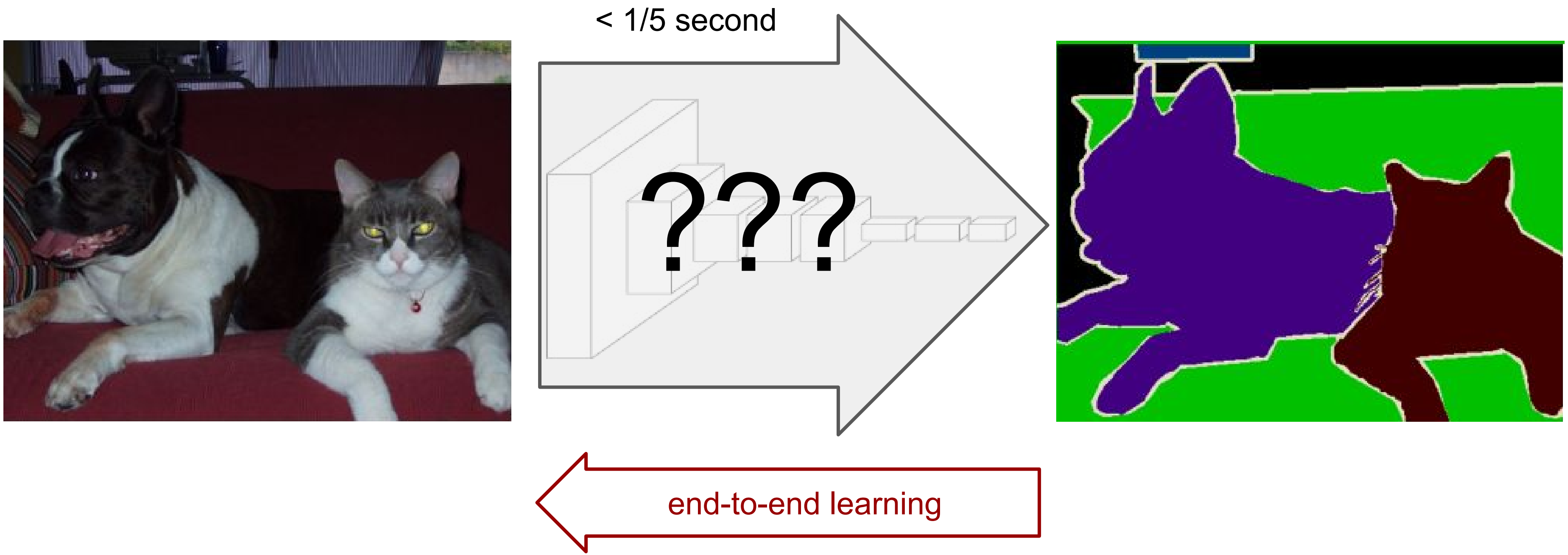
figure: Girshick et al.

But

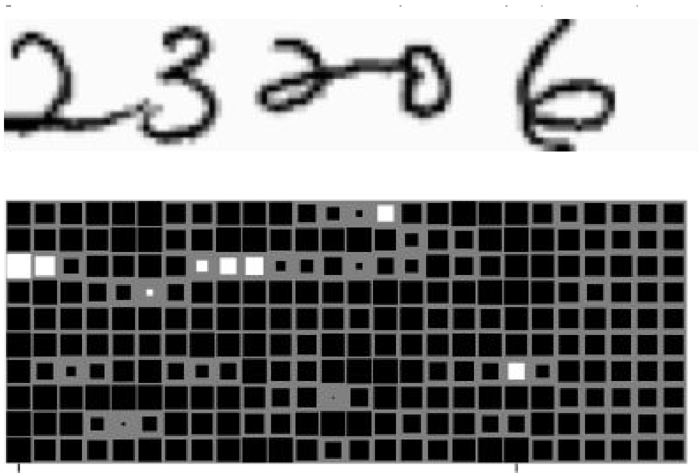


\+ Not end-to-end

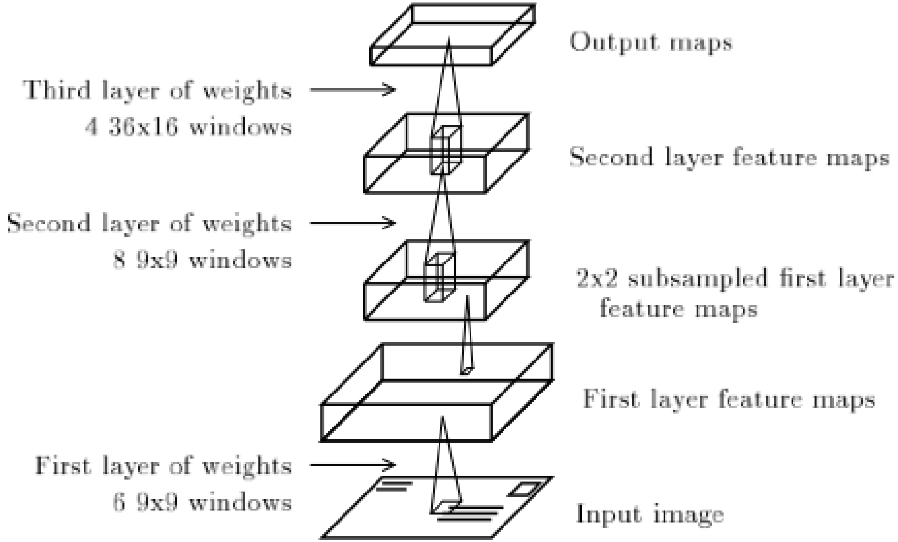
FCN



History of FCN

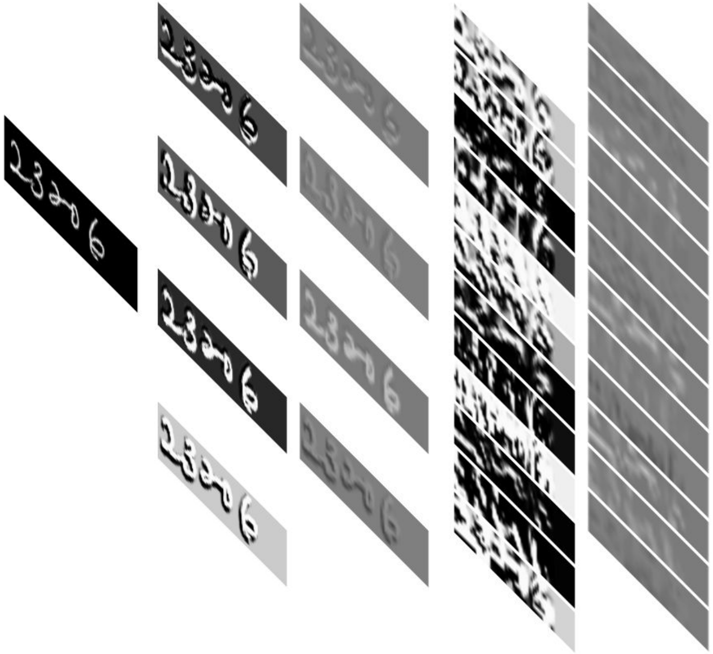


Shape Displacement Network
Matan & LeCun 1992

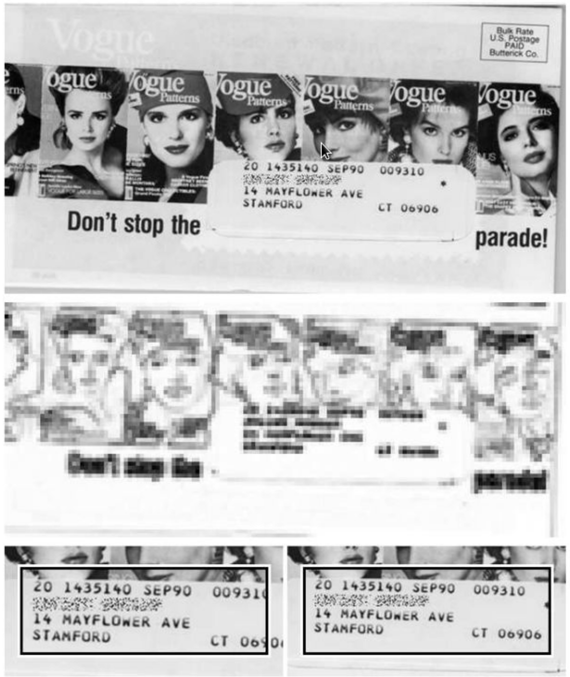


Convolutional Locator Network
Wolf & Platt 1994

Predict numbers in a row



Locate the “data” square



A Classic Classification Network

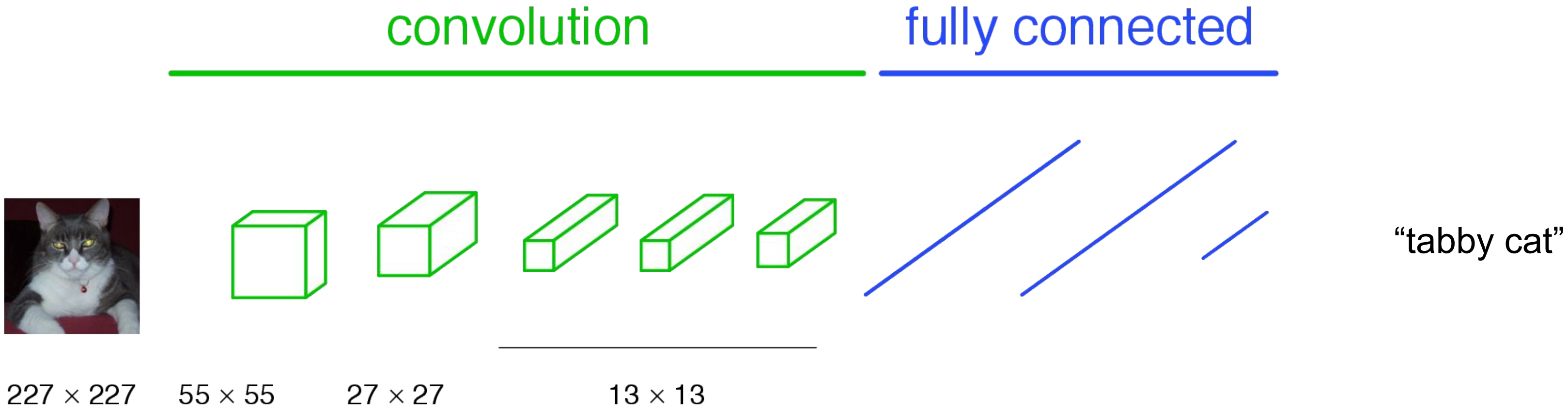
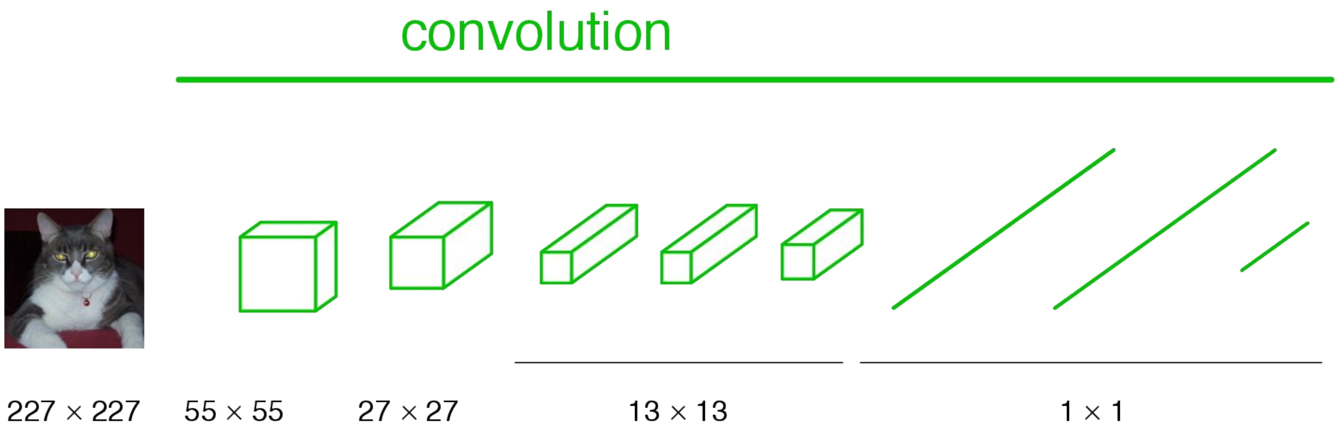


Diagram of activations

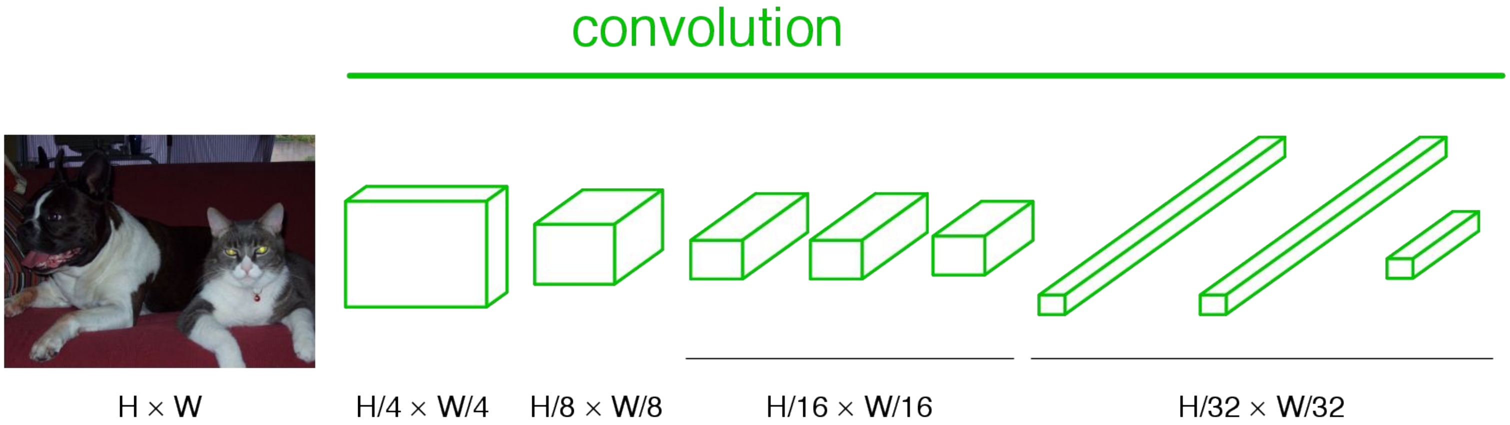
Becoming Fully Convolutional

- Fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions



Becoming Fully Convolutional

- In order to get an “heatmap”, final layers need width and height so we don’t want that big kernel...
- Add a final 1X1 conv with channel for each class

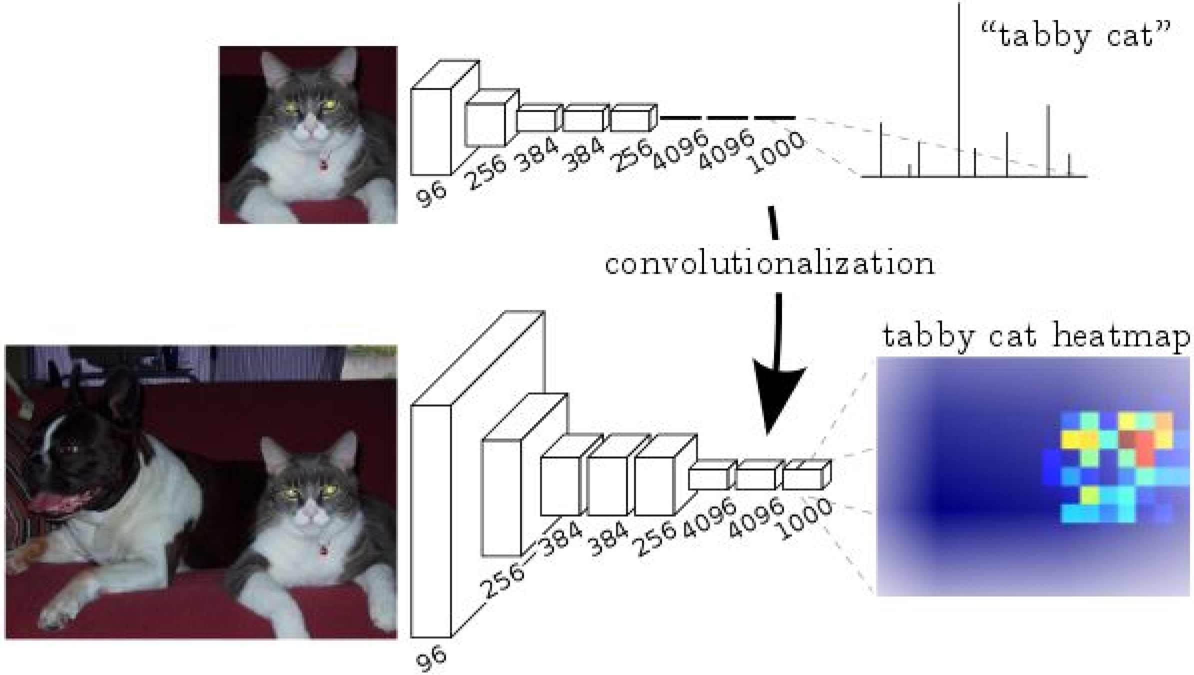


Patchwise vs Whole Image

1.2 ms with AlexNet
on 227X227

22ms to produce
10X10 from
500X500

100 results of
classification



Convolution is fast on GPU!

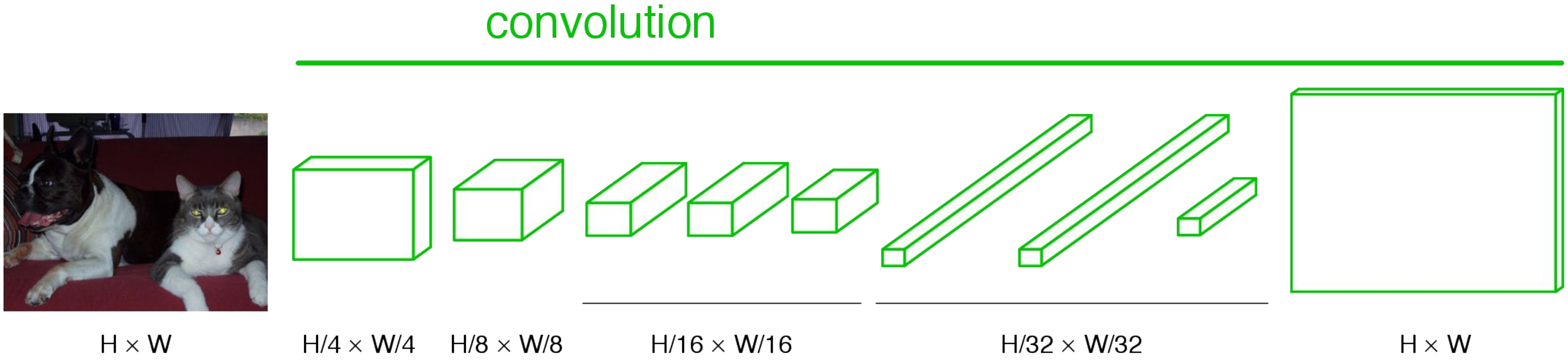
Loss Function

simply the sum of chosen loss
function on each pixel at the heatmap

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

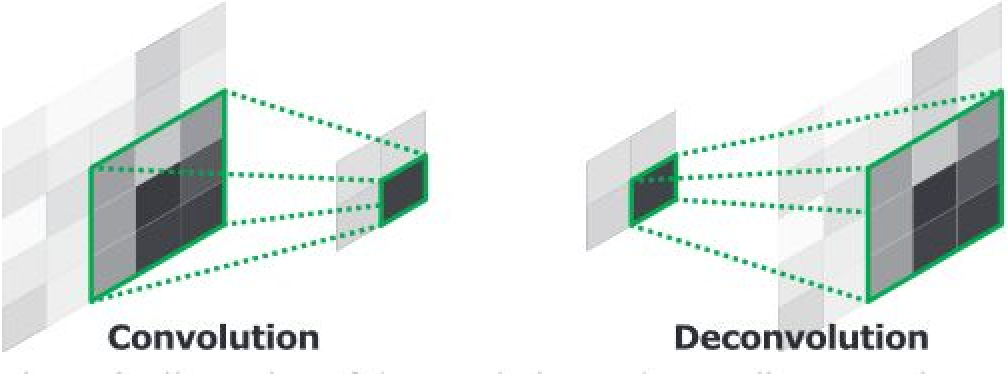
Upsampling Output

- In order to produce full image segmentation, we need to upsample the output
- Method chose: Deconvolution



Deconvolution

- Convolutional layers connect multiple input activations within a filter window to a single activation
- Deconvolutional layers associate a single input activation with multiple outputs

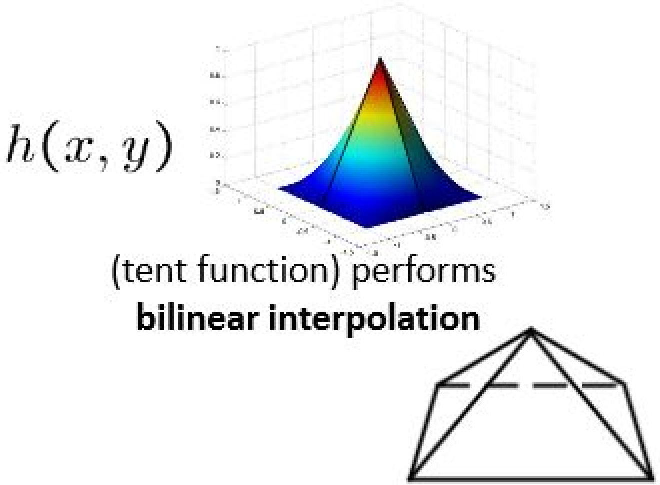


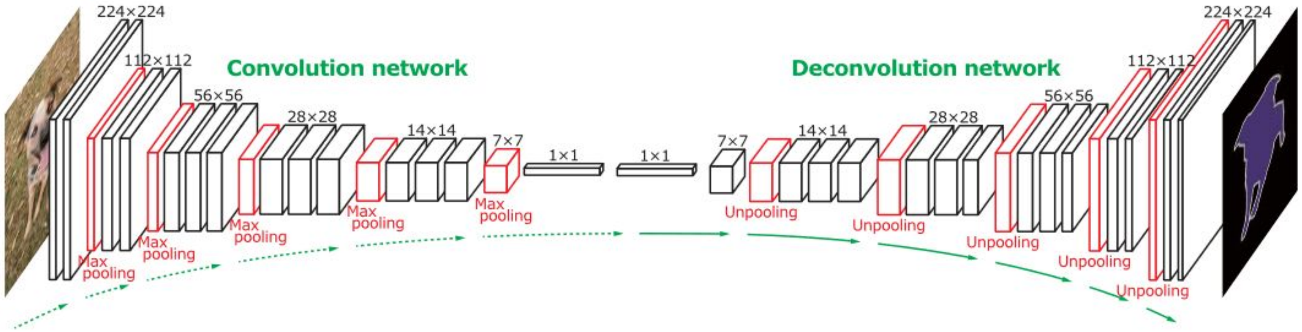
Upsampling

- Upsampling with factor f is convolution with input stride of 1/f
- Equivalent to backward convolution (aka Deconvolution) with output stride f which is already implemented in the existing code...
- Thus upsampling is performed in-network for end-to-end learning by backpropagation from the pixelwise loss

Upsampling

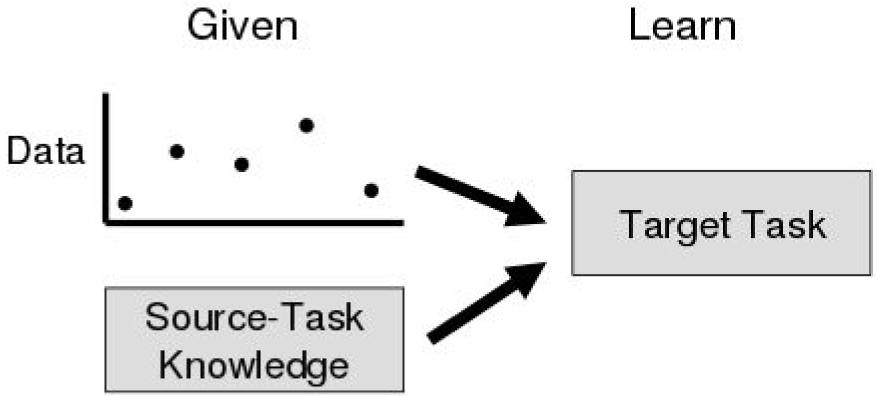
- Upsample X32 in single pass by convolving with “tent kernels” - not learned!
- It has already been proven in other work that learning the kernels and upsampling gradually can achieve slightly better results





Transfer learning

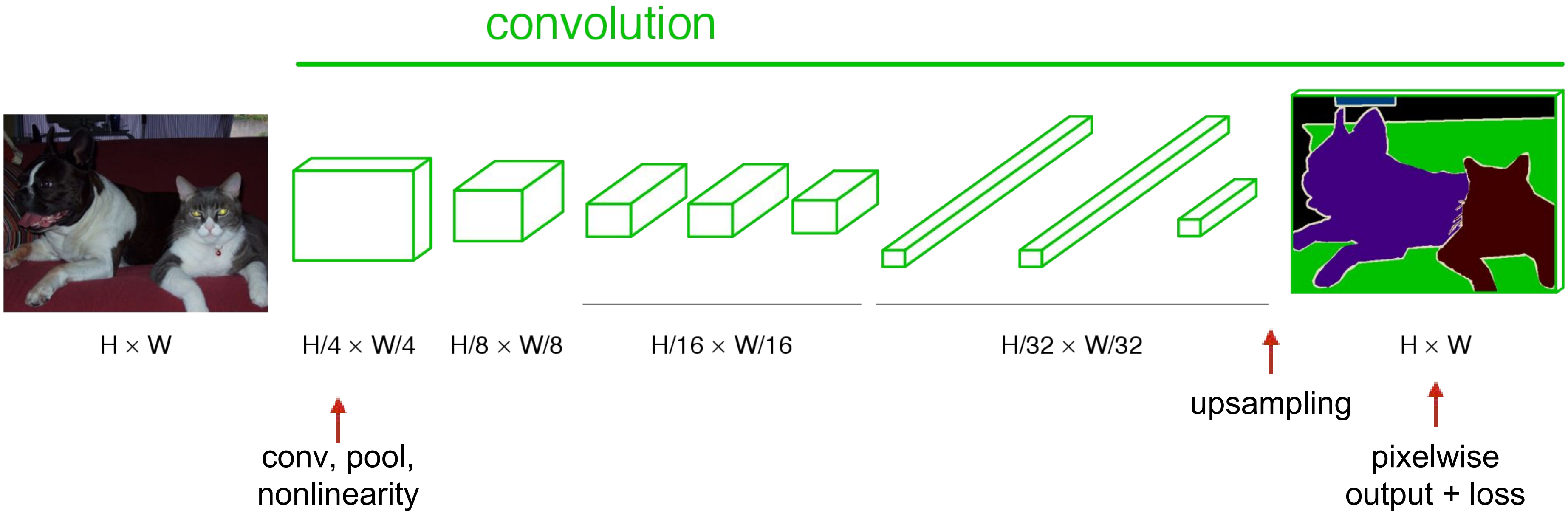
"Transfer learning is the improvement of learning in a new
task through the transfer of knowledge from a related task that has already been learnt"
(Lisa Torrey and Jude Shavlik)



Transfer learning

- Cast ILSVRC classifiers into FCNs and augment them for dense prediction: discard classifier layer, transform FC to CONV, add 1X1 CONV with 21 channel dimension for score at each output location.
- Then use in-network upsampling.
- Train for segmentation by fine-tuning all layers with PASCAL VOC 2011 with a pixelwise loss.

End-to-End, Pixels-to-Pixels Network



Evaluation Metrics

Let n_{ij} be the number of pixels of class i predicted to belong to class j
let n_{cl} the number of different classes
let $t_i = \sum_j n_{ij}$ be the total number of pixels of class i
Then we compute:

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuraccy: $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$
- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU:
  $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

#pixels that really belong to class i

#pixel that both belong to class i and got the label of class i

#pixels that got the label of class i

Results - Single Stream Created From Different Classifiers

	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

But output is coarse

Upgrade: Multi-Resolution Fusing

0 1 2

Scale Pyramid, *Burt & Adelson ‘83*

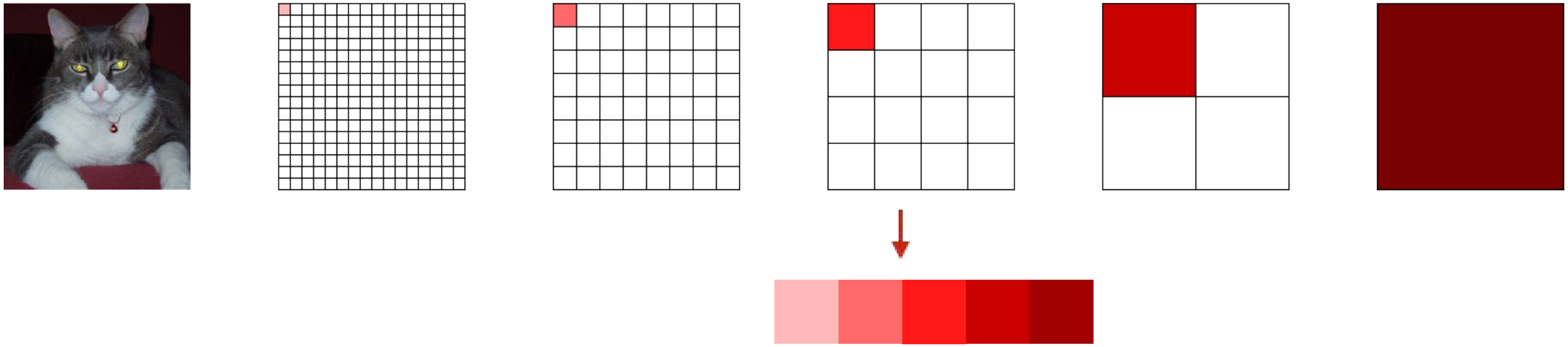
The scale pyramid is a classic multi-resolution representation.

Spectrum of Deep Features

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

image

intermediate layers

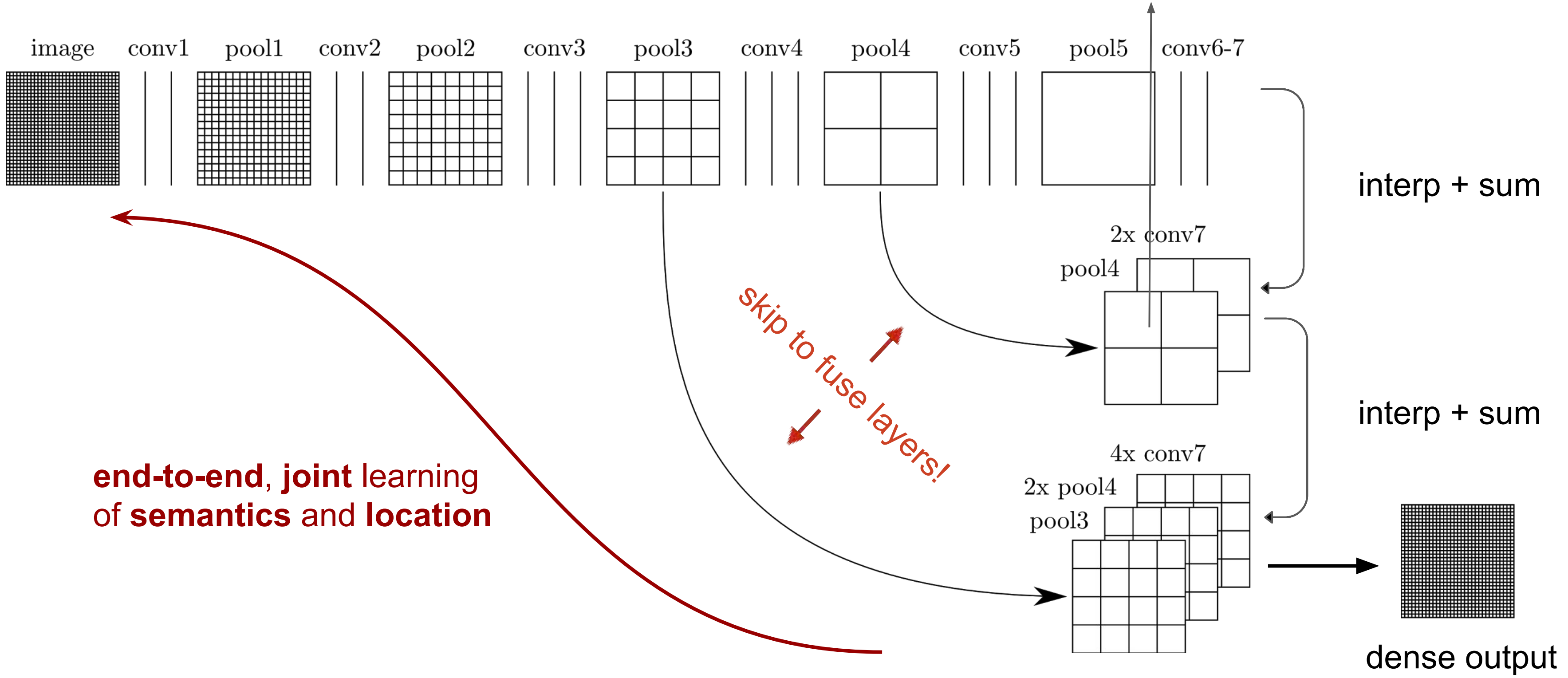


fuse features into **deep jet**

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

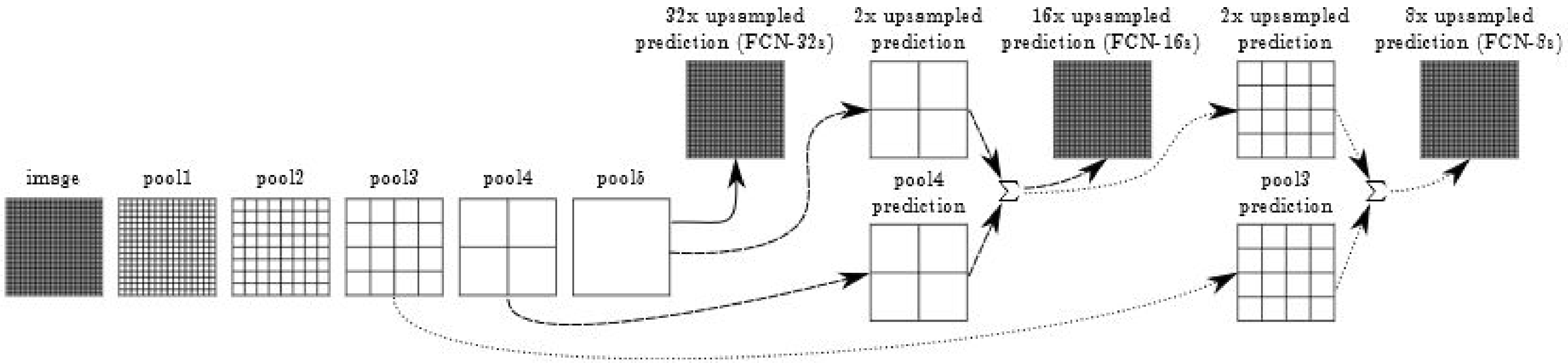
Skip Layers

Adding 1X1 conv classifying layer on top of pool4,then upsample X2 (init to bilinear and then learned) conv7 prediction, sum both, and upsample X16 for output



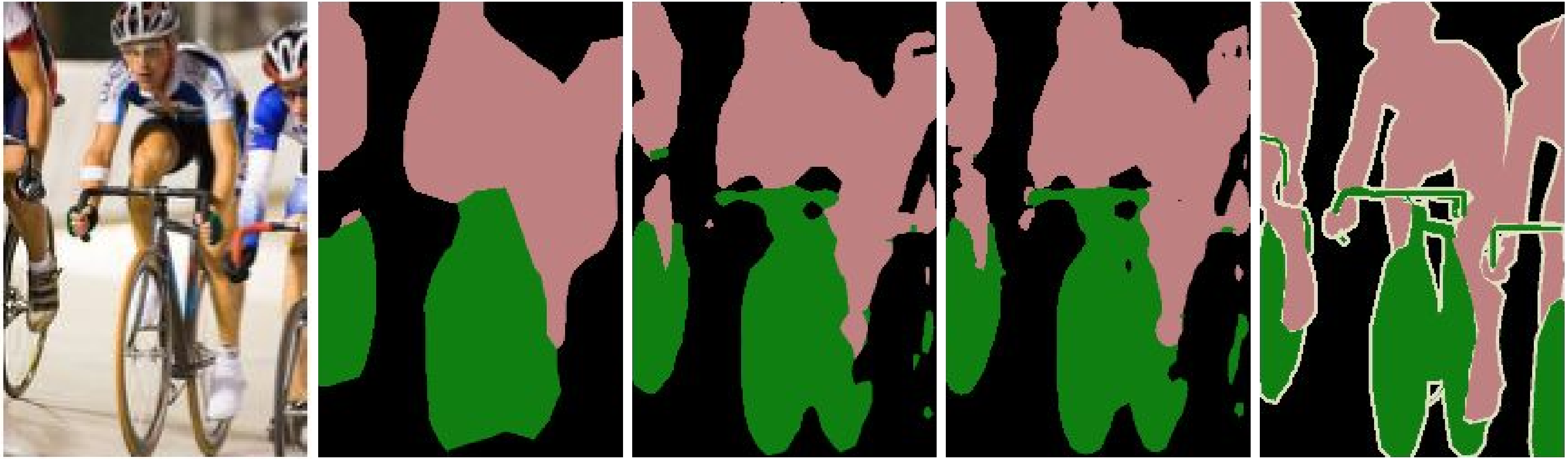
FCN-32s , 16s, 8s

- 32s is the single stream net, the final layer is downsampled X32 (2^5 pooling layers)
- 16s has skip layer from pool4 (initialized with the parameters of 32s, additional params initialized to zero)
- 8s has skip from pool3
- Each net is learned end-to-end but initialize with the “coarser” nets’ params

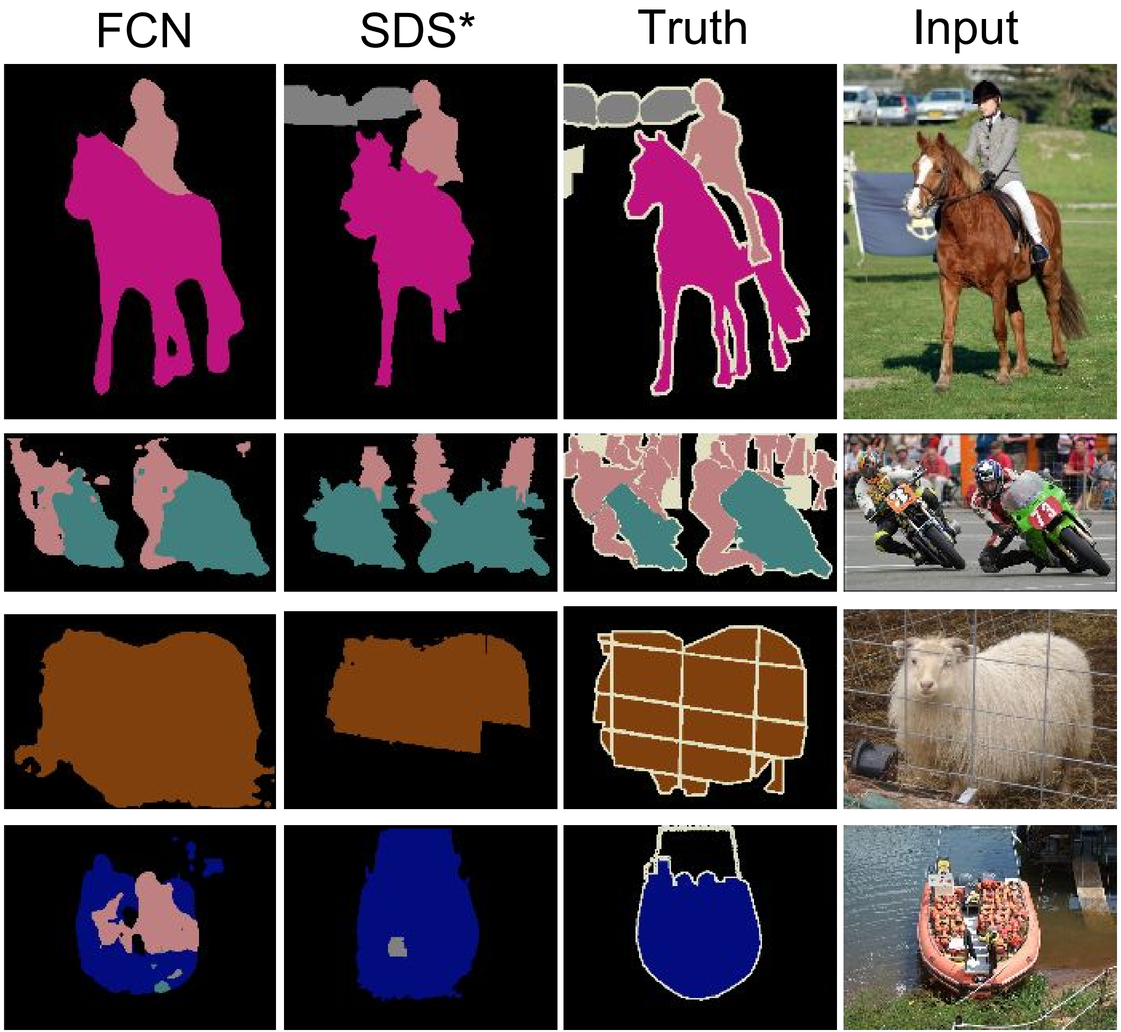


Skip Layer Refinement

input image stride 32 stride 16 stride 8 ground truth



no skips 1 skip 2 skips



Relative to prior state-of-the-art SDS:

- 20% relative improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation
Hariharan et al. ECCV14

Extensions

- Random fields
- Weak supervision

Fully Conv. Nets + Random Fields

- Apply CRF inference as a post-processing step

minimize $$E(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

unary term $$\theta_i(x_i) = -\log P(x_i)$$

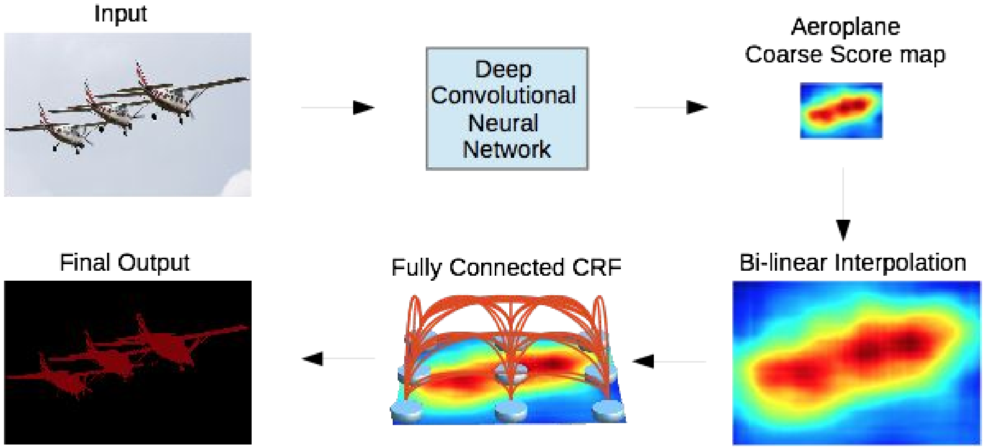
binary term $$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} w_m \cdot k^m(\boldsymbol{f}_i, \boldsymbol{f}_j)$$

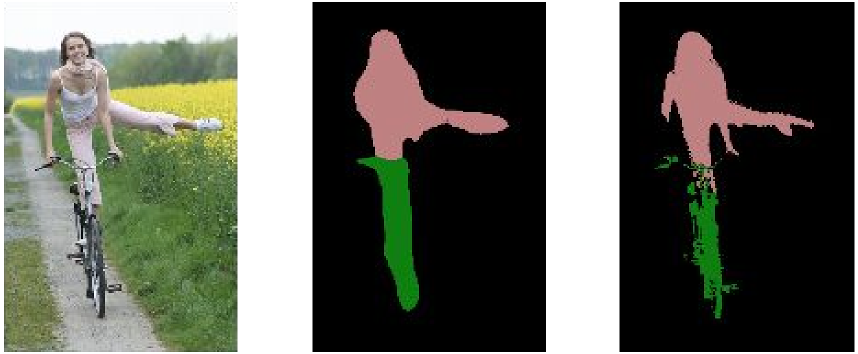
$$\mu(x_i, x_j) = 1 \text{ if } x_i \neq x_j$$ zero otherwise

position intensity

K = $$w_1 \exp\left(- \frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) + w_2 \exp\left(- \frac{||p_i - p_j||^2}{2\sigma_\gamma^2} \right)$$

Fully Conv. Nets + Random Fields

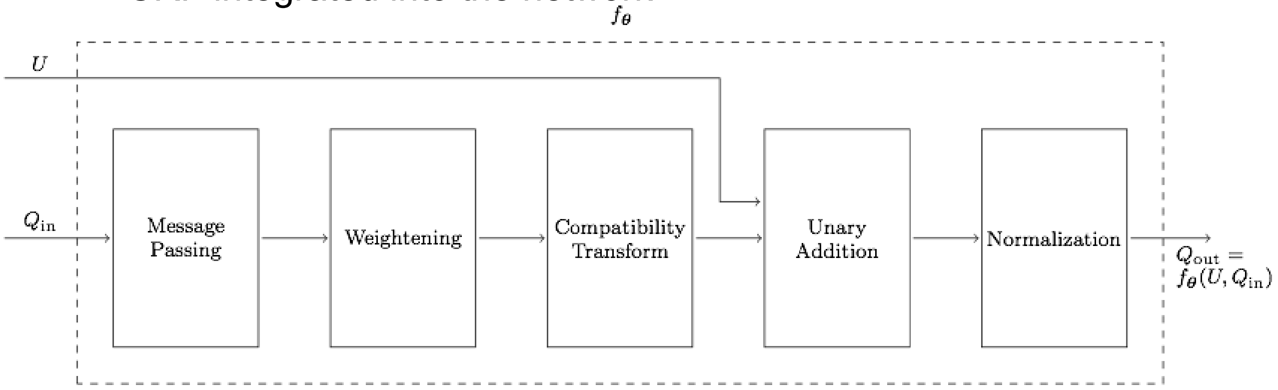




Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs.
Chen* & Papandreou* et al. ICLR 2015.

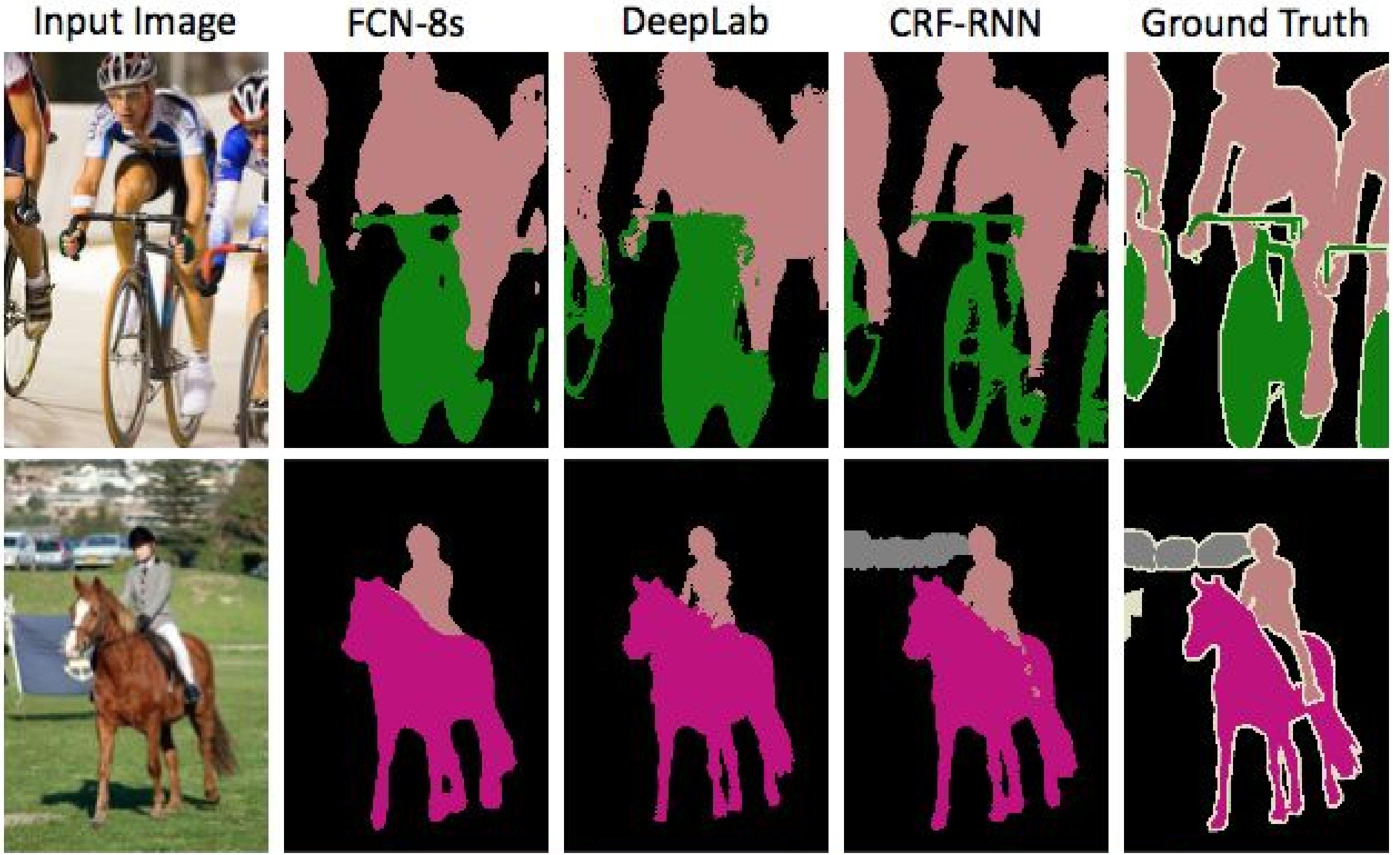
Fully Conv. Nets + Random Fields

CRF integrated into the network



Method	Without COCO	With COCO
Plain FCN-8s	61.3	68.3
FCN-8s and CRF disconnected	63.7	69.5
End-to-end training of CRF-RNN	**69.6**	**72.9**

Conditional Random Fields as Recurrent Neural Networks. *Zheng* & Jayasumana* et al.* arxiv 2015.



[comparison credit: CRF as RNN, Zheng* & Jayasumana* et al. ICCV 2015]

DeepLab: Chen* & Papandreou* et al. ICLR 2015. **CRF-RNN**: Zheng* & Jayasumana* et al. ICCV 2015

Weak Supervision

- Sometimes we cannot use the full power of supervised deep learning, due to lack of data
- Creating semantic segmentation ground truth requires a lot of work
- However, creating “weaker” ground truth is sometimes easier to create

Fully Conv. Nets + Weak Supervision

FCNs expose a spatial loss map to guide learning:
segment from tags by MIL or pixelwise constraints.

Constrained Convolutional Neural Networks for Weakly Supervised Segmentation.
Pathak et al. arXiv 2015.

Fully Conv. Nets + Weak Supervision

- Easier to express simple constraints on the output space than to craft regularizers or ad-hoc training procedures to guide the learning
- Such constraints can describe the existence and expected distribution of labels from image level tags (next slide)
- Use a loss function to optimize convolutional networks with arbitrary linear constraints on the structured output space of pixel labels

Constraints on Labels Examples

suppress any label l
that does not appear
in the image

$$\sum_{i=1}^{n} p_i(l) \leq 0 \qquad \forall\, l \notin \mathcal{L}_I\,.$$

force some
labels to appear

$$a_l \leq \sum_{i=1}^{n} p_i(l) \qquad \forall\, l \in \mathcal{L}_I\,.$$

background
constraint

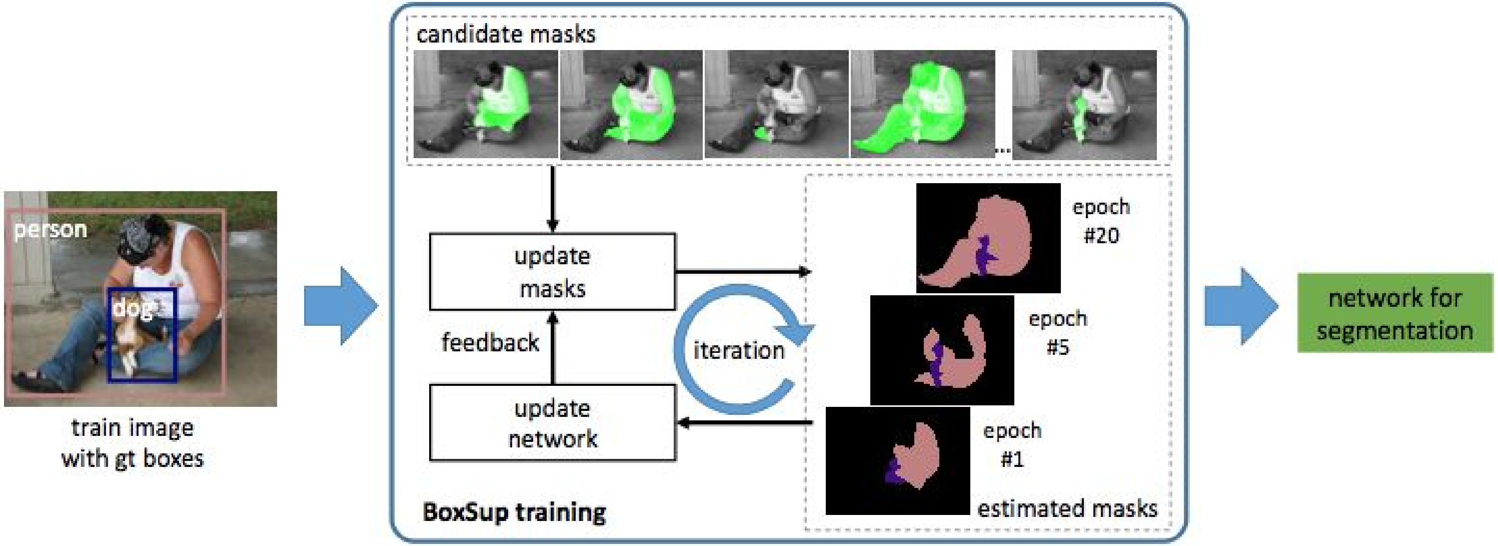
$$a_0 \leq \sum_{i=1}^{n} p_i(0) \leq b_0.$$

boost all classes larger than 10%
of the image by setting a_i= 0.1n
also put an upper bound constraint
on the classes L that are
guaranteed to be small

$$\sum_{i=1}^{n} p_i(l) \leq b_l.$$

Fully Conv. Nets + Weak Supervision

FCNs expose a spatial loss map to guide learning:
mine boxes + feedback to refine masks.



BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation.
Dai et al. 2015.

Fully Conv. Nets + Weak Supervision

- In the training, iterate between regular training with weakly supervised data (with segmentation masks created automatically from bounding box “ground truth” labeling) and iterations where we fix the network parameters and let the suggested masks slightly change
- It’s also possible to mix the training data with “fully supervised” data (with per pixel labeling)