Semantic Segmentation

Rohan Doshi (Princeton ‘18 COS)

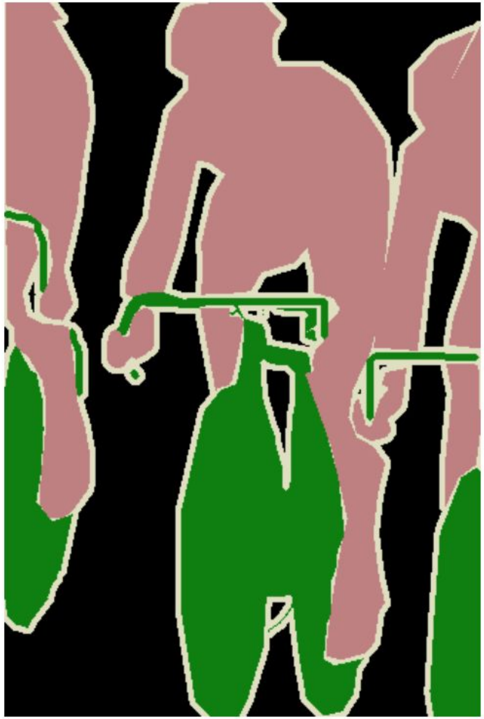
Fully Convolutional Networks for Semantic Segmentation

Long, Shelhamer and Darrell CVPR’15

[Link To Paper]

Task: Semantic Segmentation

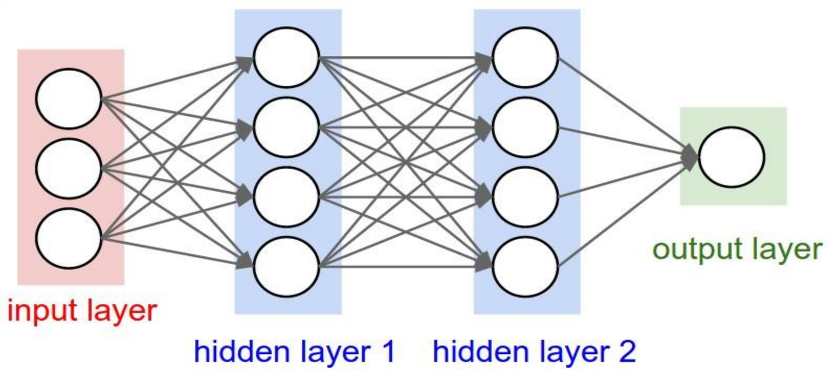
Mapping pixels to labels



Ref: Stanford CS231n

Review: Fully Connected Layer

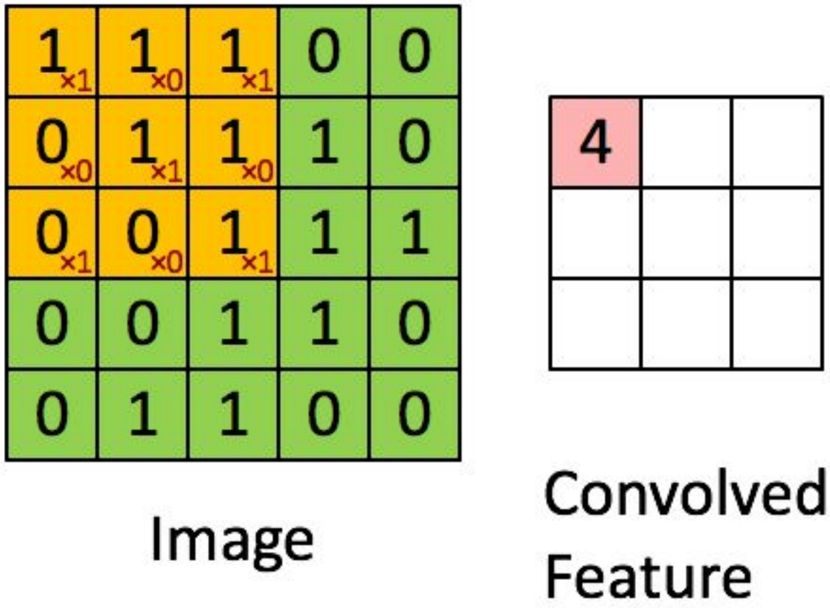
- **W:** Weight matrix: mxn
  - m: # input neurons
  - n: # output neurons
- For input vector **x**, we can calculate the layer's activation **a**:
  - **a** = **W** * **x**



Ref: Stanford CS231n

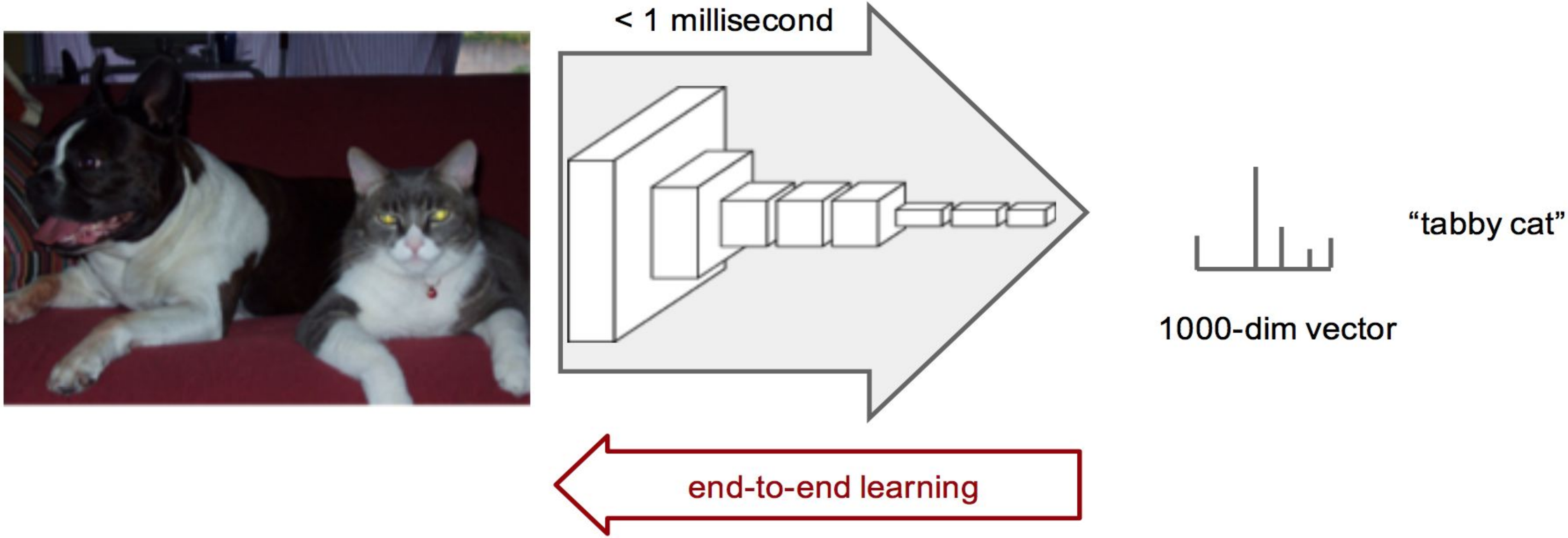
Review: Convolutional Layer

- **Input**: volume of size W1×H1×D1
- **Hyperparameters**
  - **K:** Number of filters
  - **F**: filter (square) side length
  - **S**: stride
  - **P**: amount zero padding
- **Output**: volume of size W2×H2×D2 where:
  - W2=(W1−F+2P)/S+1
  - H2=(H1−F+2P)/S+1
  - D2=K



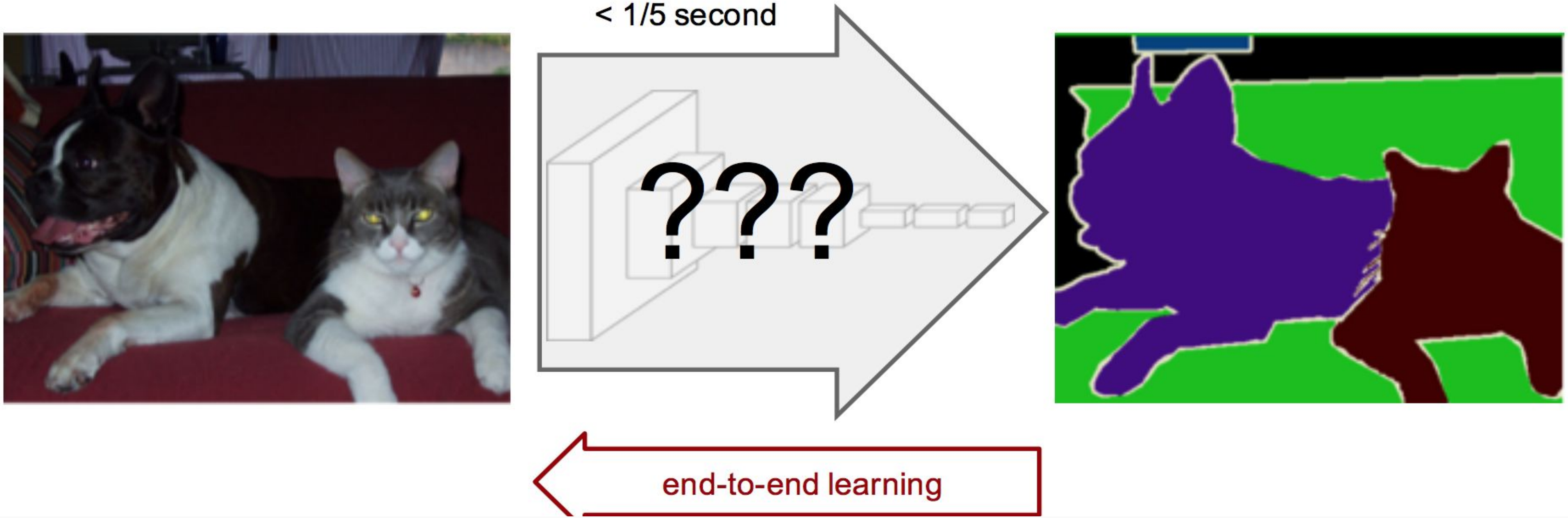
Ref: Stanford CS231n

Convolutional Neural Networks (Convnets) Perform Classification



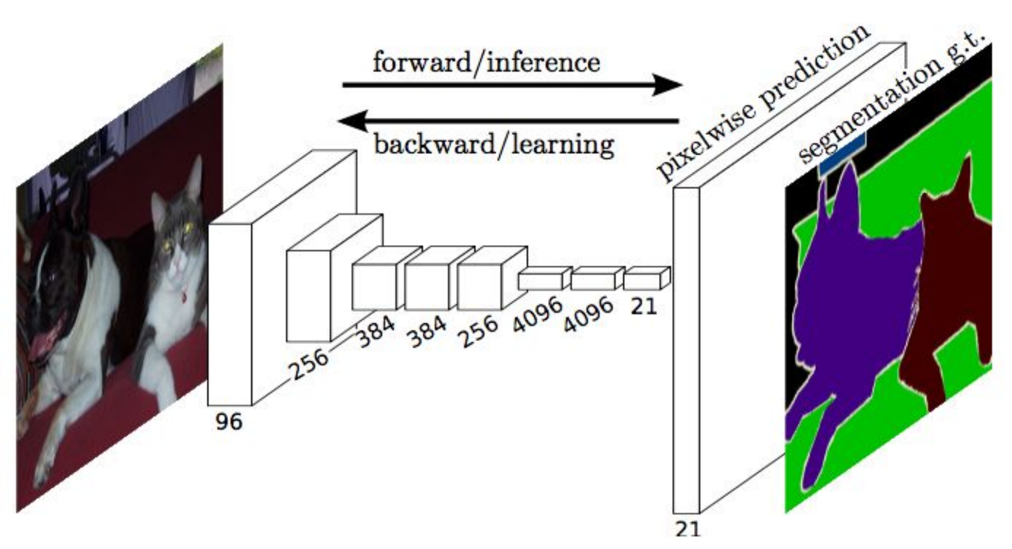
Slide Credit: Jonathon Long

How can we modify CNNs for semantic segmentation?



Slide Credit: Jonathan Long

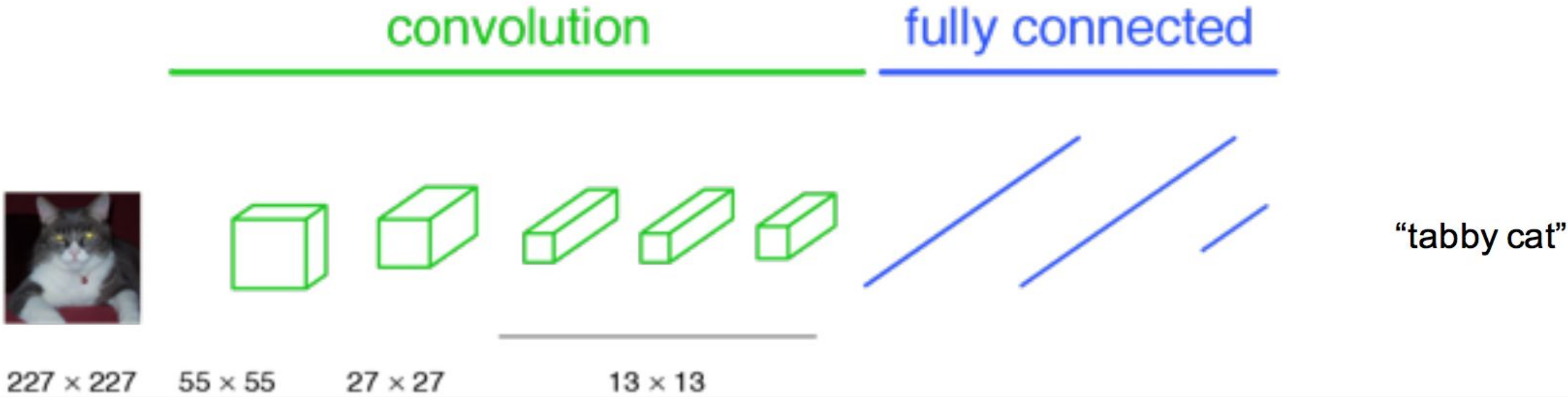
Proposal: **Fully** Convolutional Networks (FCNs)



- Reinterpret standard classification ConvNets as **fully** convolutional networks for semantic segmentation
- End-to-end pixel-to-pixel network
- Works with any sized input image (b/c convolutional layers only)
- Efficient Inference (~5ms)
- Fully Supervised Learning

Ref: Jonathan Long et. al 2015

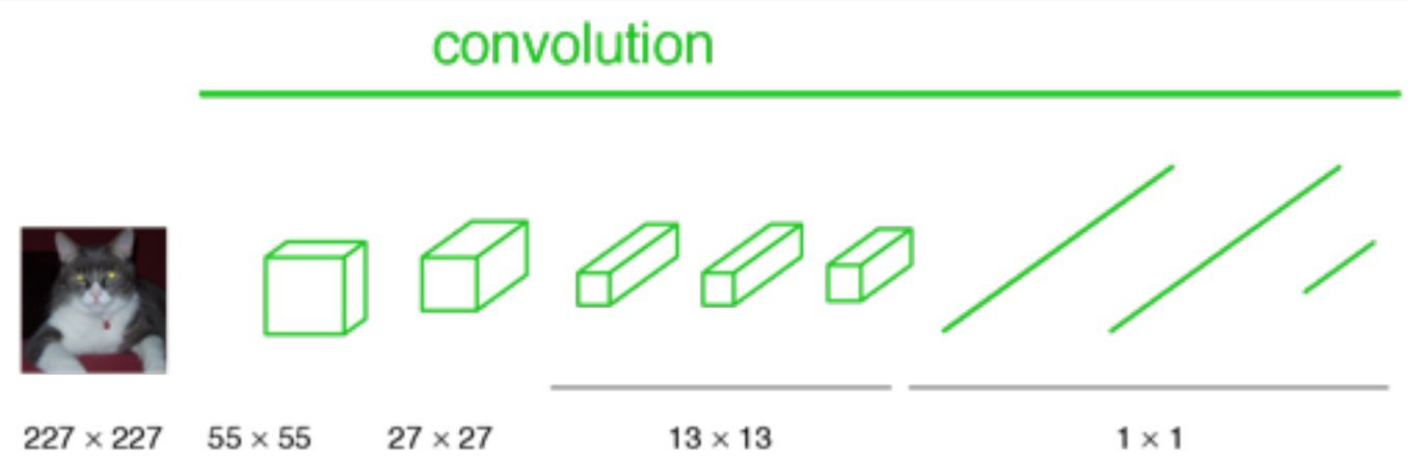
Insight 1: Adapting classifiers for dense prediction



- Problem: Fully Connected (FC) requires fixed size input and loses spatial information
- Training done patchwise (often overlapping) on a larger image→slow inference

Slide Credit: Jonathan Long

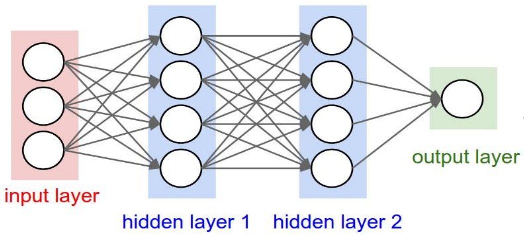
Insight 1: Adapting classifiers for dense prediction

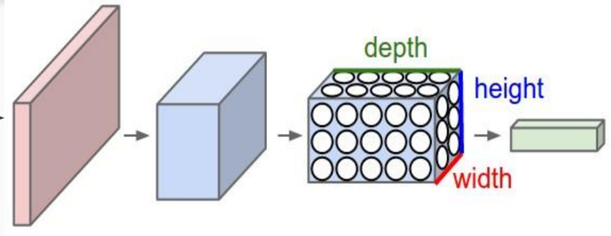


- Fully connected layers just convolutions with kernels set over the entire input region
- Replace fully connected layers with convolutional layers

Slide Credit: Jonathan Long

Fully Connected → Convolutional Layer



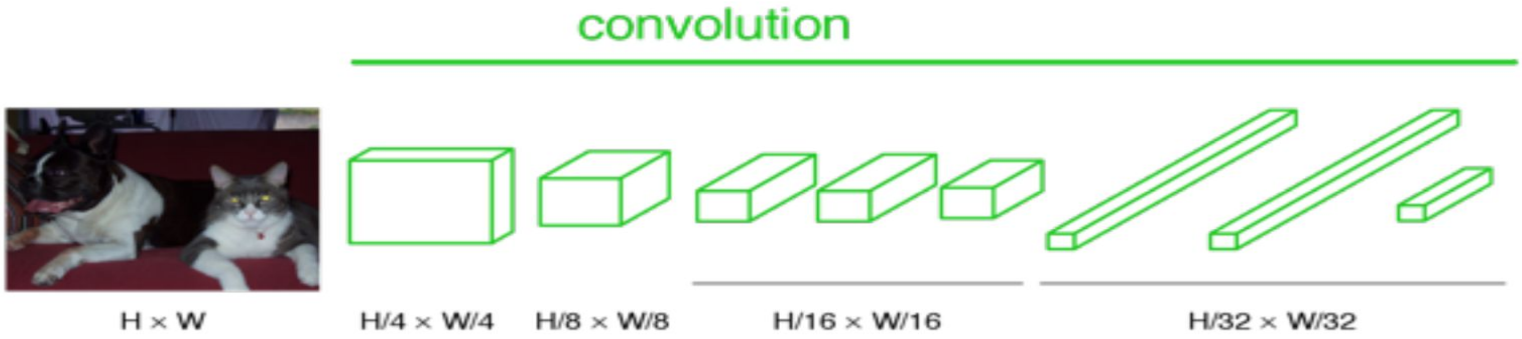


FC: 7x7x512 input layer + 4096D hidden layer 1 + 1000D layer 2. Turn into Conv:

- Replace the first FC layer that looks at [7x7x512] volume with a CONV layer that uses filter size F=7, giving output volume [1x1x4096].
- Replace the second FC layer with a CONV layer that uses filter size F=1, giving output volume [1x1x4096]
- Replace the last FC layer similarly, with F=1, giving final output [1x1x1000]

Ref: Stanford CS 231n

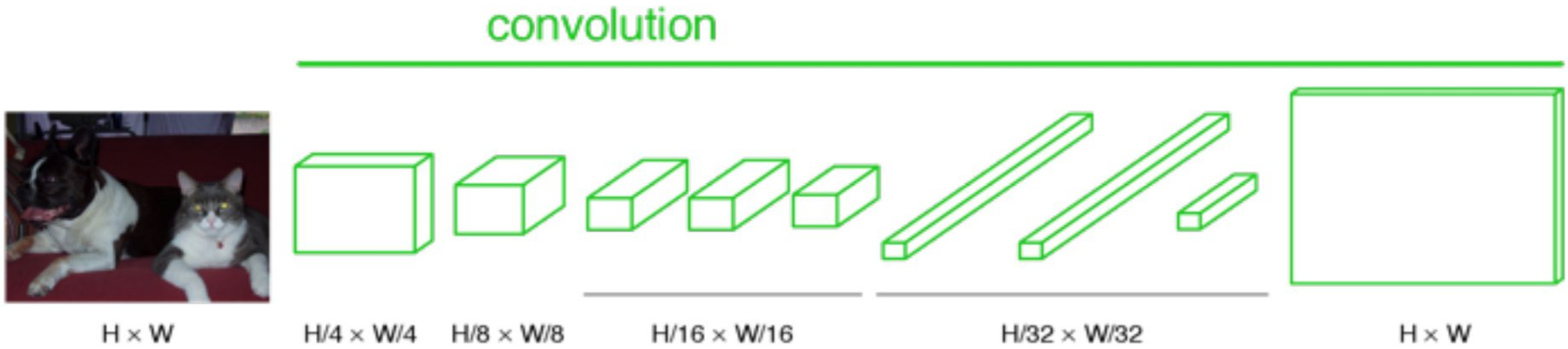
Insight 1: Adapting classifiers for dense prediction



- Faster inference: process entire image (single pass) as opposed to overlapping patches (multiple passes); convolution makes most of overlapping receptive field
- Convolution downsampled input image (32x after repeatedly downsampling by 2x)
- Generates coarse feature maps with global information

Slide Credit: Jonathan Long

Insight 1: Adapting classifiers for dense prediction



Upsample feature maps for pixel-level predictions

Fully Convolutional because no fully connected layers.

Slide Credit: Jonathan Long

Insight 1: Adapting classifiers for dense prediction

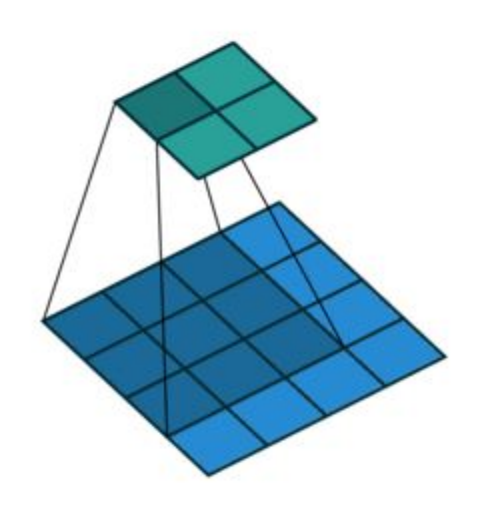
Cast ILSVRC classifiers into FCNs and compare performance on validation set of PASCAL 2011

	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

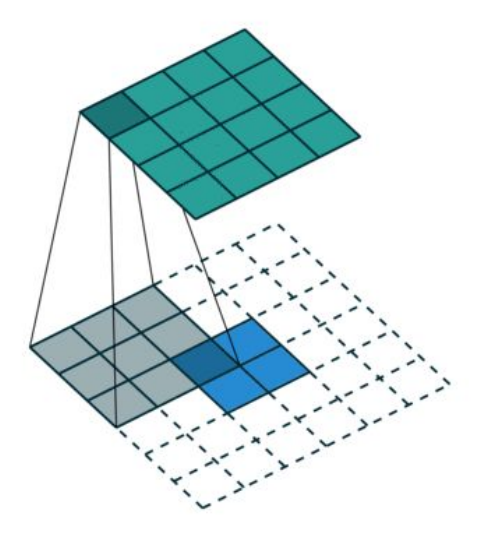
Slide Credit: Jonathan Long

Insight 2: Upsampling is transposed (aka “backwards strided”) convolution

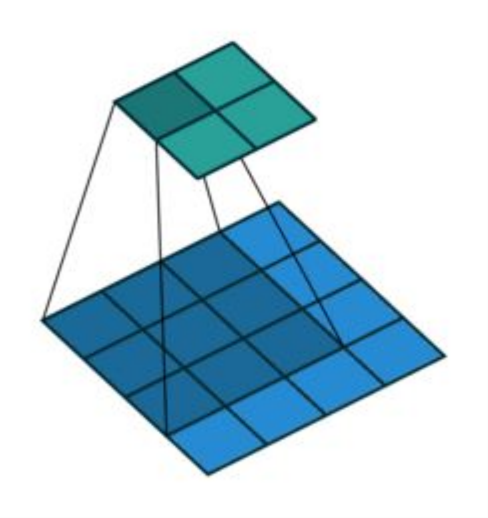
Convolution



Transposed Convolution



Why called “transpose” convolution?

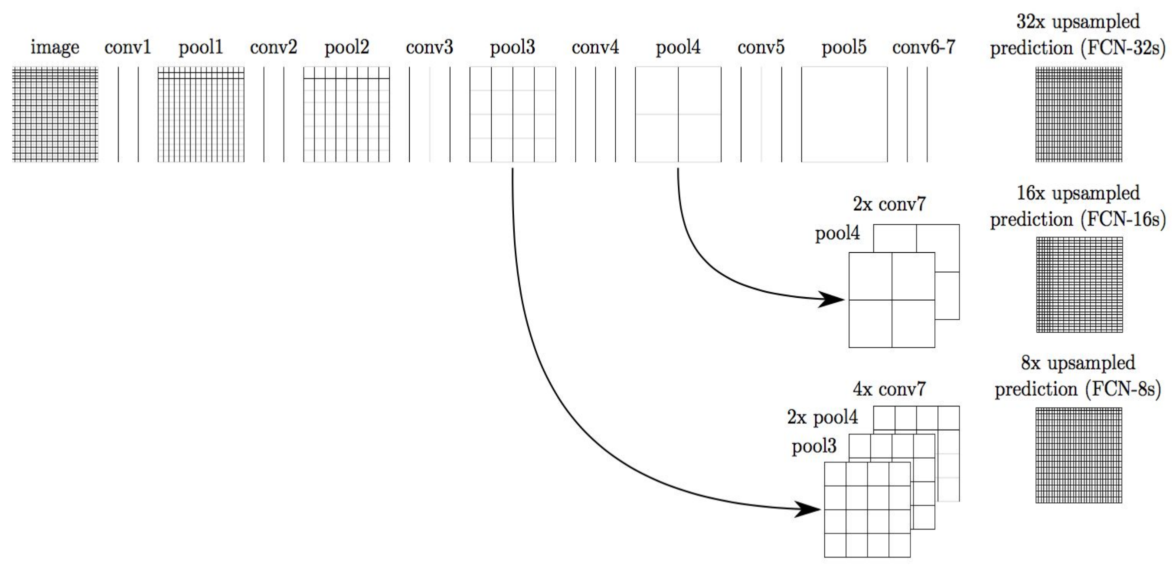


$$\begin{pmatrix}
w_{0,0} & 0 & 0 & 0 \\
w_{0,1} & w_{0,0} & 0 & 0 \\
w_{0,2} & w_{0,1} & 0 & 0 \\
0 & w_{0,2} & 0 & 0 \\
w_{1,0} & 0 & w_{0,0} & 0 \\
w_{1,1} & w_{1,0} & w_{0,1} & w_{0,0} \\
w_{1,2} & w_{1,1} & w_{0,2} & w_{0,1} \\
0 & w_{1,2} & 0 & w_{0,2} \\
w_{2,0} & 0 & w_{1,0} & 0 \\
w_{2,1} & w_{2,0} & w_{1,1} & w_{1,0} \\
w_{2,2} & w_{2,1} & w_{1,2} & w_{1,1} \\
0 & w_{2,2} & 0 & w_{1,2} \\
0 & 0 & w_{2,0} & 0 \\
0 & 0 & w_{2,1} & w_{2,0} \\
0 & 0 & w_{2,2} & w_{2,1} \\
0 & 0 & 0 & w_{2,2}
\end{pmatrix}^T$$

- Ex: Apply a 3x3 filter to 4x4 input to get 2x2 output
- Linearize the input at a vector: 4x4 input = 16D vector
- Thus, the output will be 2x2 = 4D vector
- Convert Conv → FC by rewriting the filter weights in a matrix. This will take us from 16D → 4D
- To reverse operation during upsampling (and go from 4D → 16D), just apply the transpose of the convolution weight matrix

Ref: Theano Documentation

Insight 3: Skip Layers



Semantics vs Location

When we downsample 32x, we lose spatial information.

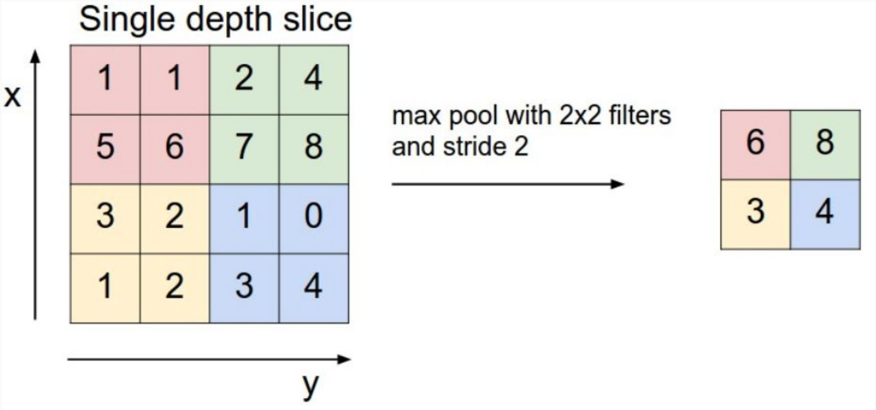
Idea: Combine **What + Where**

- **What:** coarse, downsampled high-layer predictions
- **Where**: fine, low-layer predictions

Ref: Long et. al 2015

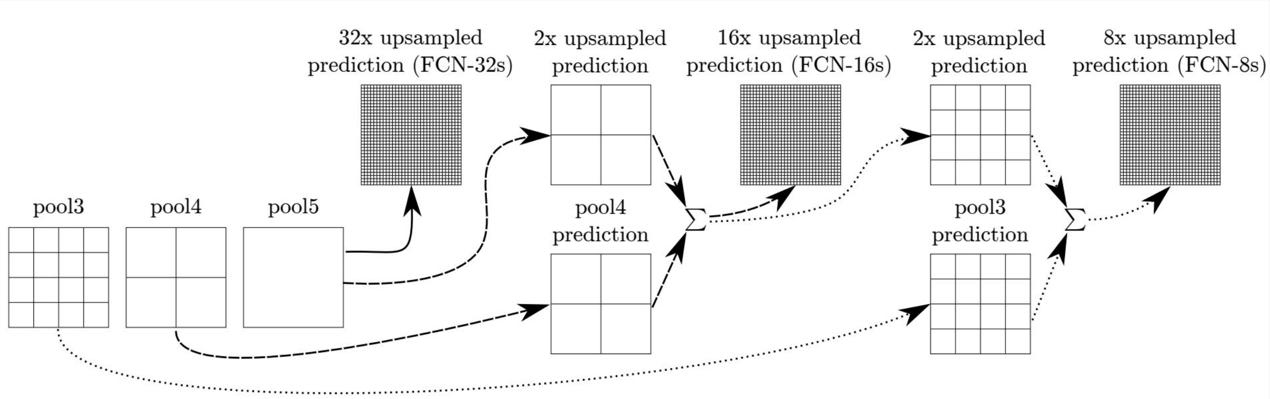
Review: Pooling Layer

- **Input**: volume of size W1×H1×D1
- **Hyperparameters**
  - **F**: filter length
  - **S**: stride
- **Output**: volume of size W2×H2×D2 where:
  - W2=(W1−F)/S+1
  - H2=(H1−F)/S+1
  - D2=D1
  - Pooling → ***downsampling***



Ref: Stanford CS231n

Insight 3: Skip Layers



Ex: FCN-16s:

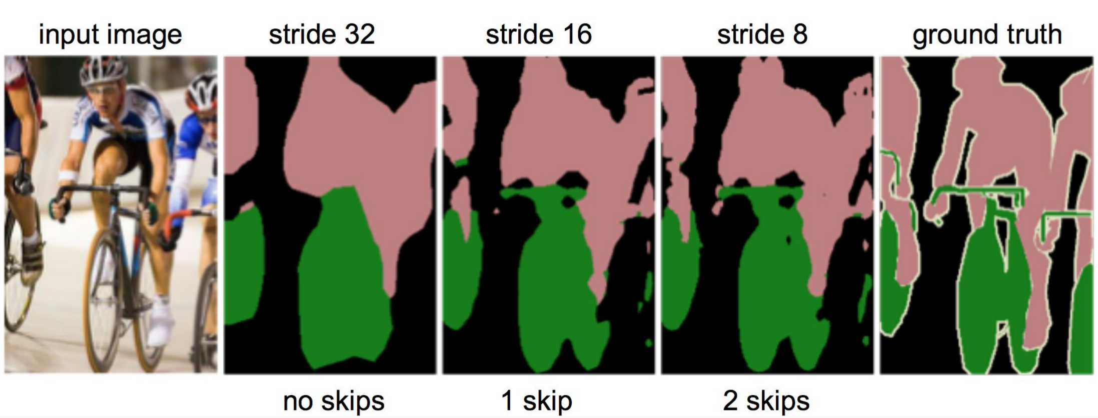
1. Take Conv 7 (from stride 32 layer), apply 1x1 Conv (K=21), upsample by 2x using transpose convolution.
2. Take Pool 4 (from stride 16 layer), apply 1x1 Conv (K=21)
3. Add the two tensors, upsample 16x using transpose convolution to get dimensions of original image, with 21 predictions at each pixel.

Ref: Long et. al 2014

Comparison of Skip FCNs

Results on subset of validation set for PASCAL VOC 2011

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**



Slide Credit: Jonathan Long

Results: PASCAL VOC 2011

VOC 2011: 8498 training images (from additional labeled data)

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

Slide Credit: Jonathan Long

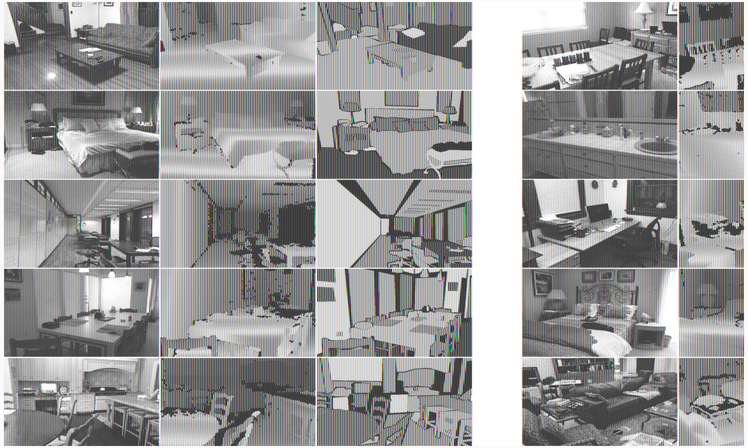
Results: NYUDv2

1449 RGB-D images collected with Microsoft Kinect

Pixelwise labels → 40 categories

Literature suggests alternative encoding: HHA

	pixel acc.	mean acc.	mean IU	f.w. IU
Gupta *et al.* [14]	60.3	-	28.6	47.0
FCN-32s RGB	60.0	42.2	29.2	43.9
FCN-32s RGBD	61.5	42.4	30.5	45.5
FCN-32s HHA	57.1	35.2	24.2	40.4
FCN-32s RGB-HHA	64.3	44.9	32.8	48.0
FCN-16s RGB-HHA	**65.4**	**46.1**	**34.0**	**49.5**



Slide Credit: Jonathan Long

Results: SIFT Flow

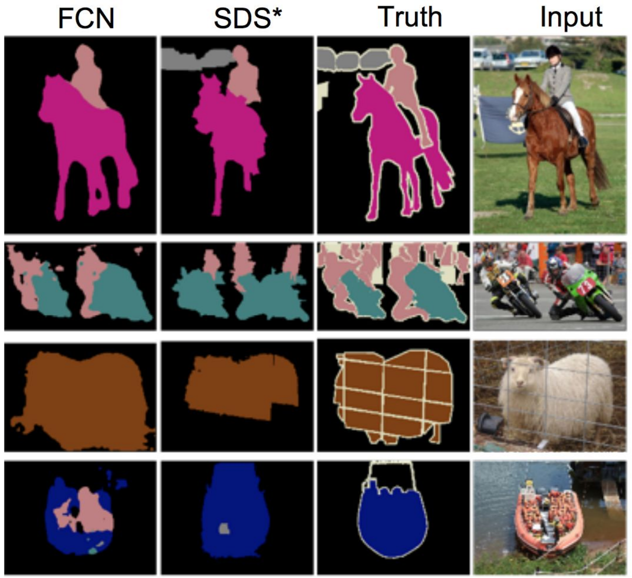
- 2688 images with pixel labels
  - 33 semantic categories, 3 geometric categories
- Learn both label spaces jointly
  - learning and inference have similar performance and computation as independent models

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al.* [23]	76.7	-	-	-	-
Tighe *et al.* [33]	-	-	-	-	90.8
Tighe *et al.* [34] 1	75.6	41.1	-	-	-
Tighe *et al.* [34] 2	78.6	39.2	-	-	-
Farabet *et al.* [8] 1	72.3	50.8	-	-	-
Farabet *et al.* [8] 2	78.5	29.6	-	-	-
Pinheiro *et al.* [28]	77.7	29.8	-	-	-
FCN-16s	**85.2**	**51.7**	39.5	76.1	**94.3**

Slide Credit: Jonathan Long

Baseline against previous state-of-art: SDS

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



Slide Credit: Jonathan Long

Recap

- Reinterpret standard classification convnets as “Fully convolutional” networks (FCN) for semantic segmentation
- Used AlexNet, VGG, and GoogleNet in experiments
- Novel architecture: combine information from different layers for segmentation
- State-of-the-art segmentation for PASCAL VOC 2011, NYUDv2, and SIFT Flow at the time
- Inference less than one fifth of a second for a typical image

Fully Convolutional Multi-Class Multiple Instance Learning

Pathak et al. ICLR workshop ’15

Link To Paper

Collecting Semantic Segmentation Datasets for Full Supervision is Expensive

- Full supervision training for semantic segmentation requires large, annotated datasets → bottleneck
- It is time consuming and expensive to annotate large datasets
    - "79s per label per image" [Russakovsky et al. Arxiv 2015]

Slide Credit: Deepak Pathak

Idea: Weak Supervision

- Image-level labels (presence or absence of class)
- Cheap to obtain
- Labeling procedure:
  - For each image, label class as *present* (at least one pixel has this class) or *absent*
- ***Objective: Learn a semantic segmentation model from just weak image-level labels***

person
horse
background

Slide Credit: Deepak Pathak

Proposal

- Train FCN end-to-end on weak image-level labels to output heatmap for each class; generate semantic segmentation by taking argmax of heatmaps at each pixel and bilinearly interpolates to image resolution.
  - FCN works with images of any size
  - Don’t require object proposal regions (e.g. bounding boxes)
- Multiclass MIL (Multiple Instance Learning) Loss
  - Inspired by binary MIL Scenario
- Challenge: Localization
  - Classes may not be centered in image; may be multiple objects

Fully Convolutional MIL (MIL-FCN)

- Weights initialized using the16-layer VGG ILSVRC 2014 classifier weights (used for image-level labels)
  - Transferred output parameters from the classes common to both ILSVRC and PASCAL
  - Pre-training prevents model from converging to all background and other degenerate solutions
- Replaced the FC layers with Conv layers for semantic segmentation
- Fine-tuned with MIL loss (next slide)

Multi-class MIL Loss

$$(x_l, y_l) = \arg\max_{\forall(x,y)} \hat{p}_l(x, y) \qquad \forall l \in \mathcal{L}_I$$

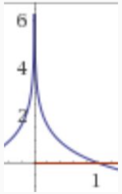
$$\implies \text{MIL LOSS} = \frac{-1}{|\mathcal{L}_I|} \sum_{l \in \mathcal{L}_I} \log \hat{p}_l(x_l, y_l)$$

- Maximize classification score based on each pixel-instance
- Takes advantage of inter-class competition to narrow down instance hypotheses

L_I: Label set of present classes

(x_l, y_l): max scoring pixel in coarse heat-maps of a class I

$\hat{p}(xl, yl)$: output heat-map for the l^{th} label at location (x, y)



y = -log(x)

Results

- PASCAL VOC 2011 with Hariharan et al. (2011) train augmentations
- Test on held out PASCAL VOC 2012 test set
- Inference *fast* (~ ⅕ second)

Approach	mean IU (VOC2011 val)	mean IU (VOC2012 test)
Baseline (no classifier)	3.52%	-
Baseline (with classifier)	13.11%	13.09%
MIL-FCN	**25.05%**	**25.66%**
Oracle (supervised)	59.43%	63.80%

Results

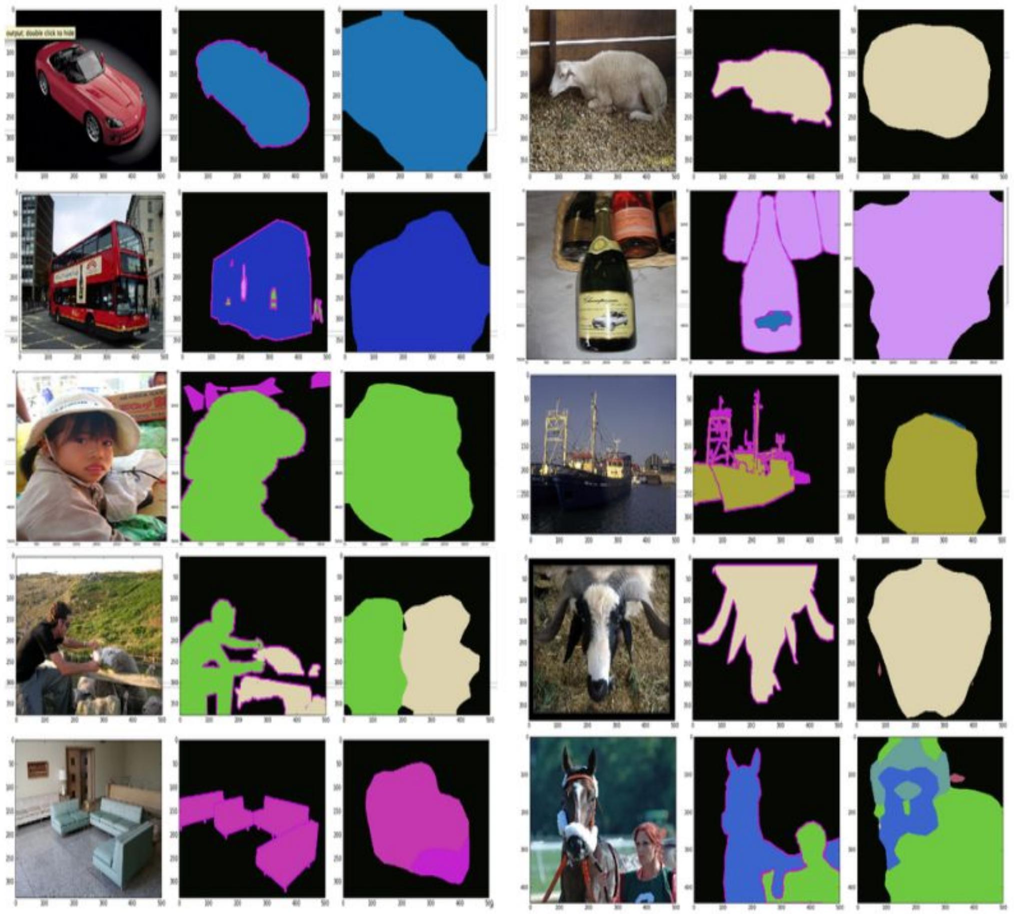


Figure 1: *Sample images from PASCAL VOC 2011 val-segmentation data.* Each row shows input (left), ground truth (center) and MIL-FCN output (right).

Future Improvements

- Coarse output is merely interpolated
- Conditional random field regularization could refine predictions (upcoming)
- Grouping methods could drive learning by selecting whole segments instead of single points for MIL training

Semantic image segmentation with deep convolutional nets and fully connected CRFs

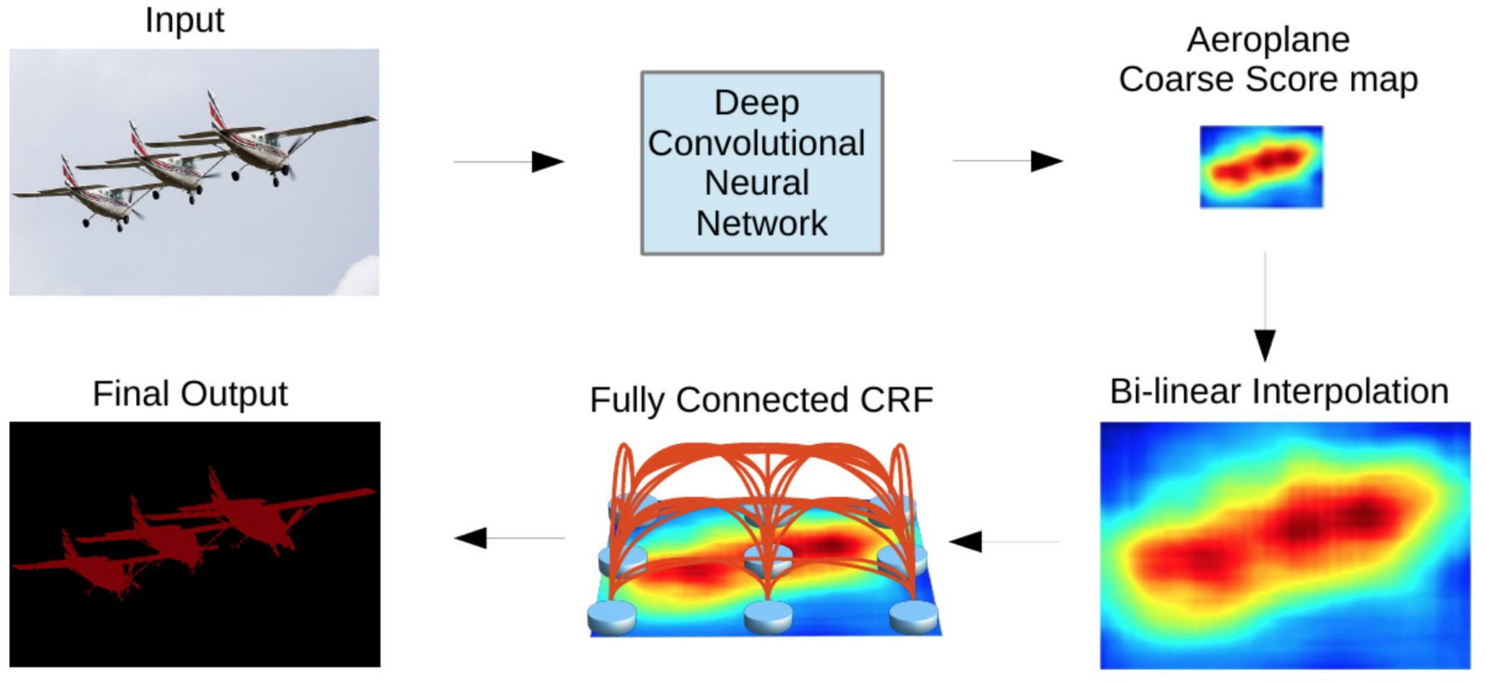
Chen et al. ICLR ’15

[Link To Paper]

Motivation: The accuracy/localization tradeoff

- Deep CNNs (DCNN) trade-off: classification accuracy and localization accuracy
  - DCNNs can reliably predict the presence and rough position of objects
  - But, coarse output (due to downsampling) not sufficiently localized for accurate object segmentation
- Fully connected CRFs excel at localization for segmentation tasks
- Idea: Bring together DCNNs (deep CNNs) and probabilistic graphical models (e.g. Conditional Random Fields) for semantic segmentation

Proposed System: “DeepLab”



Slide Credit: George Papandreou

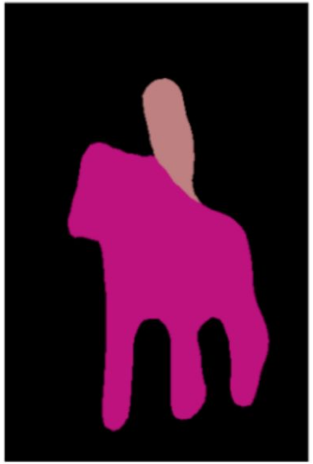
DCNN

- Weights initialized using the16-layer VGG ILSVRC 2014 classifier weights; FC layers converted into Conv layers
- Train convnet to predict label of center pixel
- Apply in sliding window fashion to generate coarse score map, and apply bilinear interpolation to return output to size of original image

Slide Credit: George Papandreou

Challenge: Controlling Receptive Field

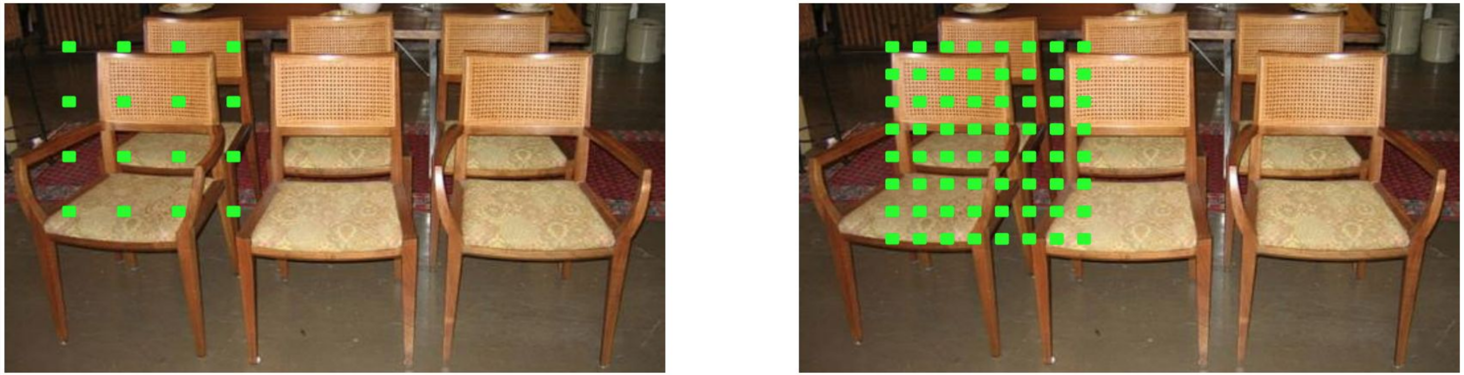
- Large CNN receptive field → poor performance near boundaries



Slide Credit: George Papandreou

Idea 1: Explicit control of response density

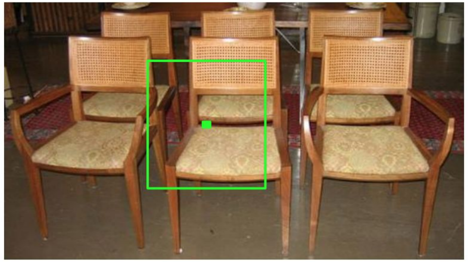
- Decrease score map stride: 32 → 8
- Efficient implementation with “atrous” algorithm
- Enables “dense” feature extraction



Slide Credit: George Papandreou

Idea 2: Controlling receptive field (RF) size to accelerate dense computation

- Reduce RF size by conv layer manipulation
  - We convert VGG's first FC layer (4096 neurons) to 7x7x4096 Conv filter → computational bottleneck. Subsample first FC layer 7x7 → 3x3, making computation about 2 - 3 times faster.
  - Reducing channels from 4096 to 1024 didn't sacrifice performance while significantly reducing computational load.



Slide Credit: George Papandreou

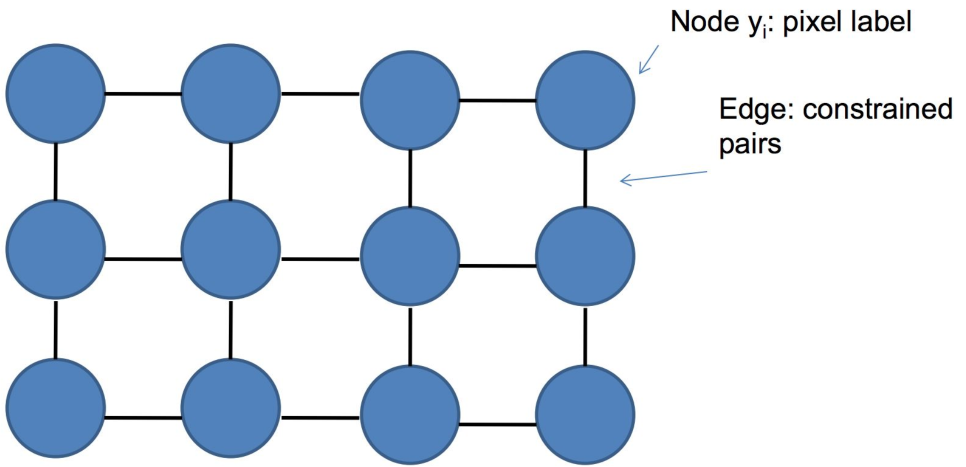
Accurate Boundary Recovery with Conditional Random Fields (CRFs)

- CRFs excel at localization and can improve ***segmentation*** boundaries

Slide Credit: George Papandreou

Review: Markov Random Fields

- It break graph into segments, treats each pixel as a node, and deletes edges that cross segments
- Used for GrabCut algorithm
- Uses graph cuts, unary potential, and energy function to generate segments



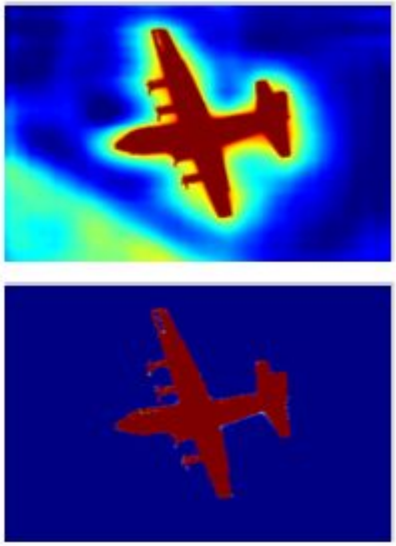
Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) \; \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

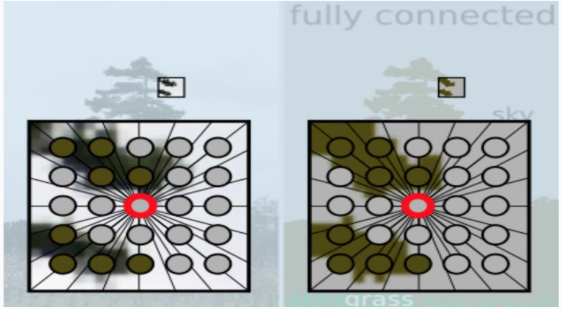
Challenges: Conditional Random Fields (CRFs)

- CRFs traditionally used to smooth noisy segmentation maps
  - Models contain energy terms that couple neighboring nodes, favoring same-label assignment
- DCNN coarse outputs already smooth with homogenous classifications
- We want to recover ***detailed local structure*** and ***thin structures***



Fully Connected Conditional Random Fields (CRFs)

- Idea: treat every pixel as a ***fully-connected*** CRF node in order to exploit long-range dependencies
  - Every node is connected to every other node
  - Use CRF inference to directly optimize a DCNN-driven cost function



Fully Connected CRF: Energy Function

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

Unary Term

$$\theta_i(x_i) = -\log P(x_i).$$

$P(x_i)$: label assignment prob. of pixel i

Pairwise Term

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

where $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, else 0

Each $\boldsymbol{k^m}$ is the Gaussian kernel depends on features (denoted as $\boldsymbol{f}$) extracted for pixel $\boldsymbol{i}$ and $\boldsymbol{j}$ and is weighted by parameter $\boldsymbol{w_m}$:

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

The first kernel depends on both pixel positions (denoted as $\boldsymbol{p}$) and pixel color intensities (denoted as $\boldsymbol{I}$), and the second kernel only depends on pixel positions. The hyper parameters $\boldsymbol{\sigma\alpha}$, $\boldsymbol{\sigma\beta}$ and $\boldsymbol{\sigma\gamma}$ control the "scale" of the Gaussian kernels.

Slide Credit: George Papandreou

Results

- Achieved state-of-the-art for PASCAL VOC-2012 semantic image segmentation task (71.6 IoU), using Hariharan (2011) annotation augmentation
- Input, DCNN, CRF-DCNN:

Slide Credit: George Papandreou