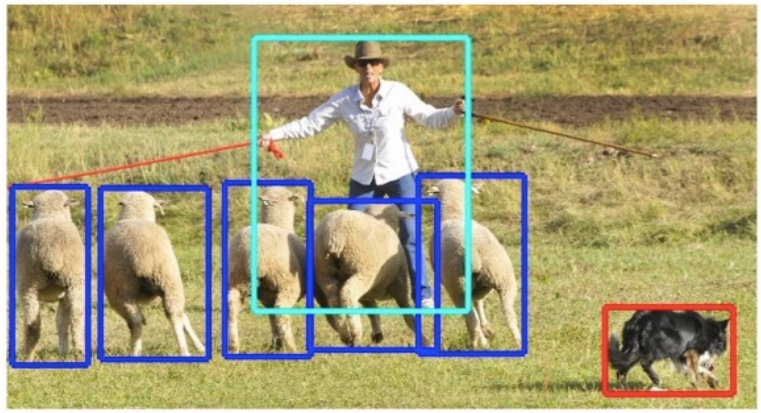
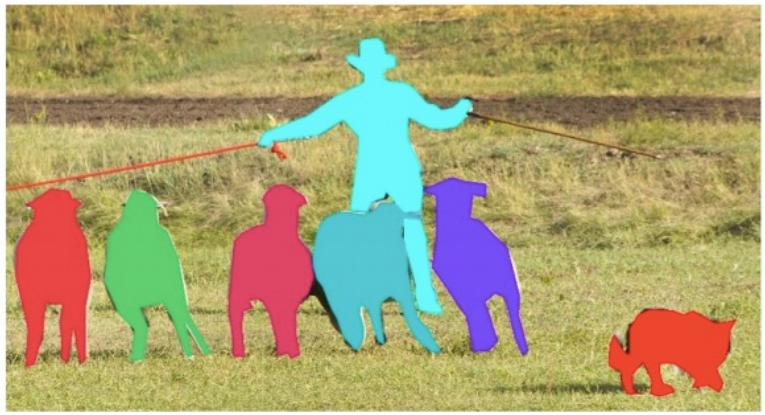
CNNs for dense image labeling

image classification



object detection

semantic segmentation



instance segmentation

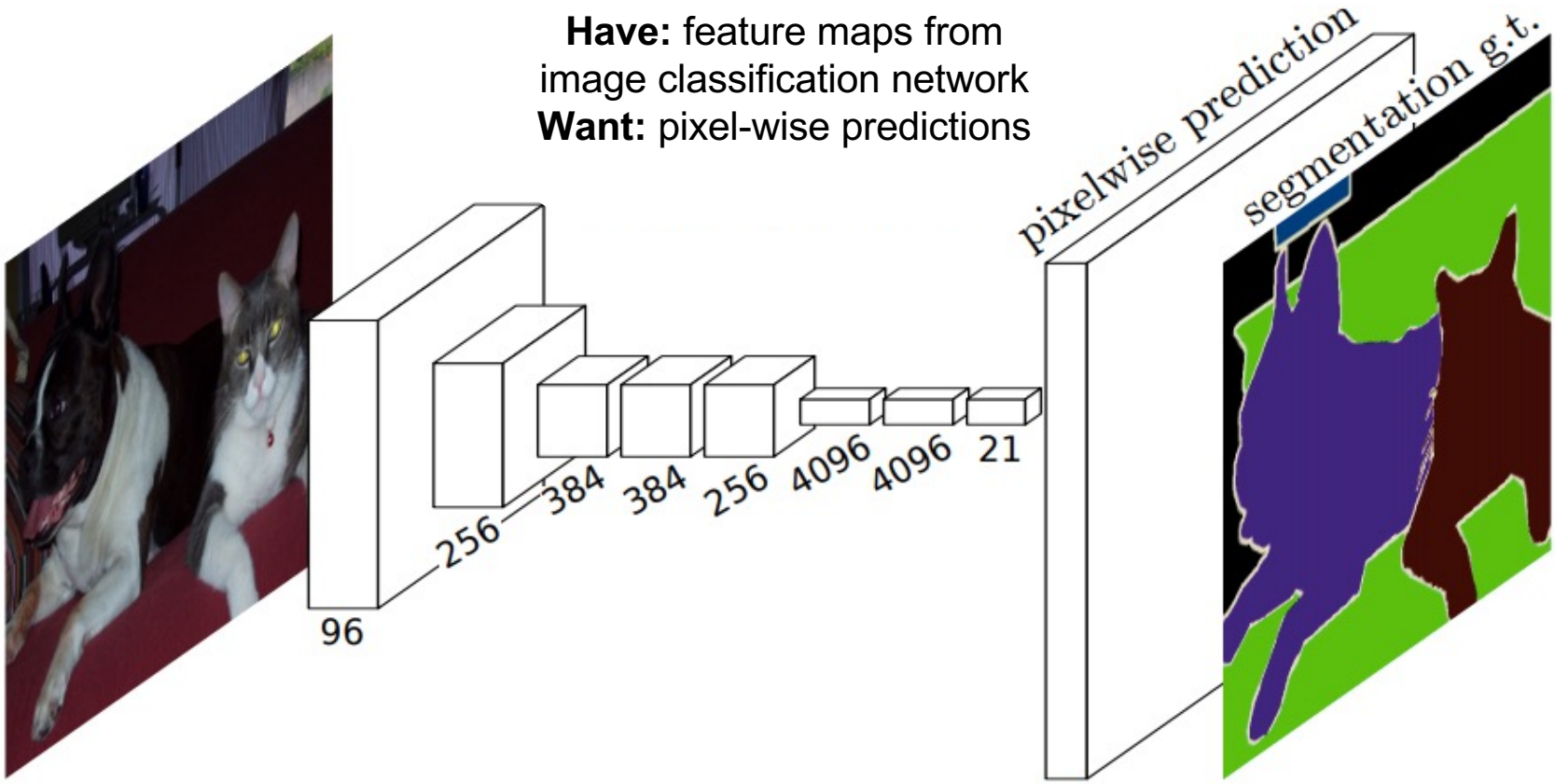
Outline

- Early “hacks”
  - Hypercolumns
- Fully Convolutional Networks (FCN)
  - Upsampling architectures
  - Dilated convolutions
- Instance segmentation
  - Mask R-CNN
  - Metric learning

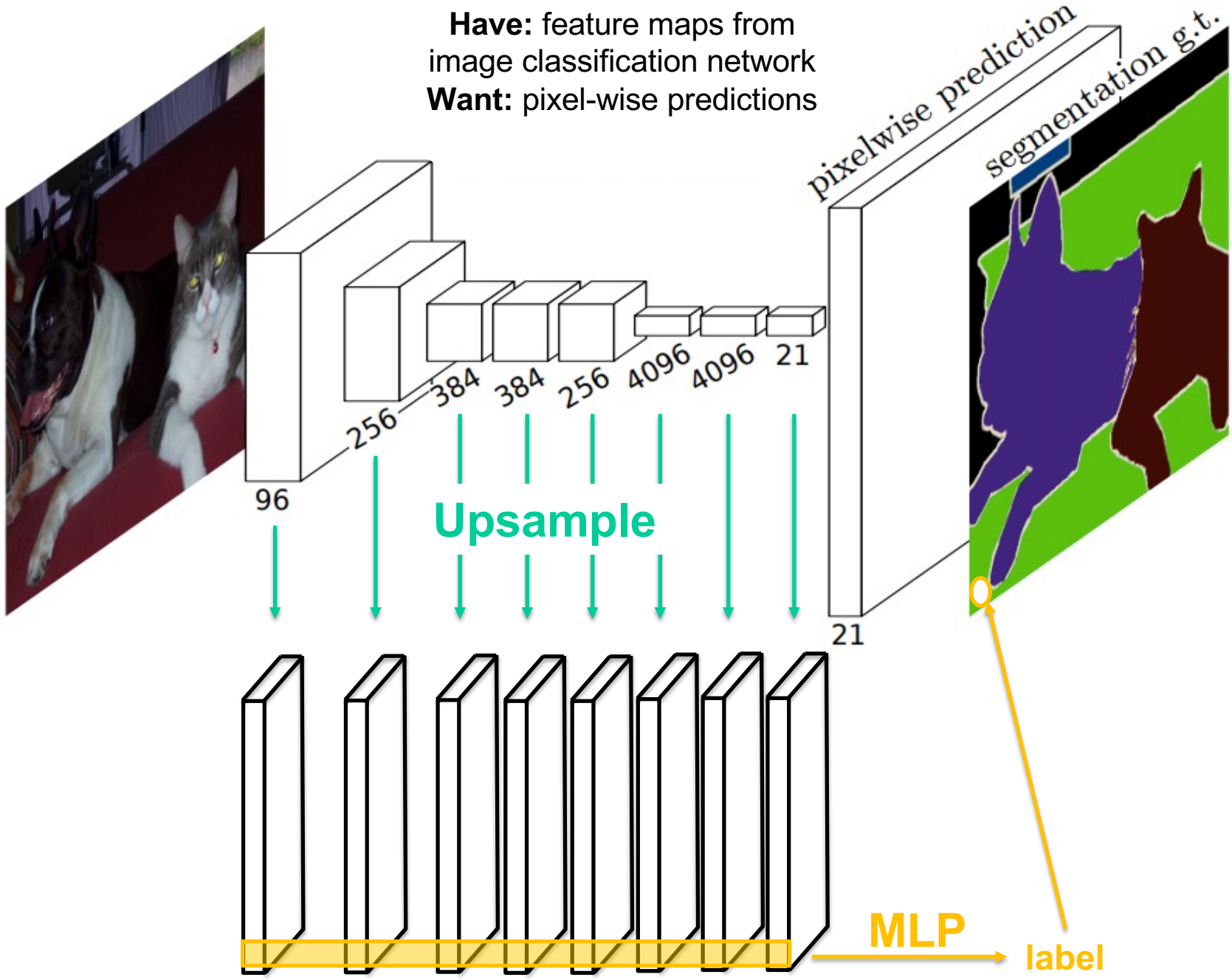
Mask R-CNN, Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.

Hypercolumns

- Do dense prediction as a post-process on top of an image classification CNN

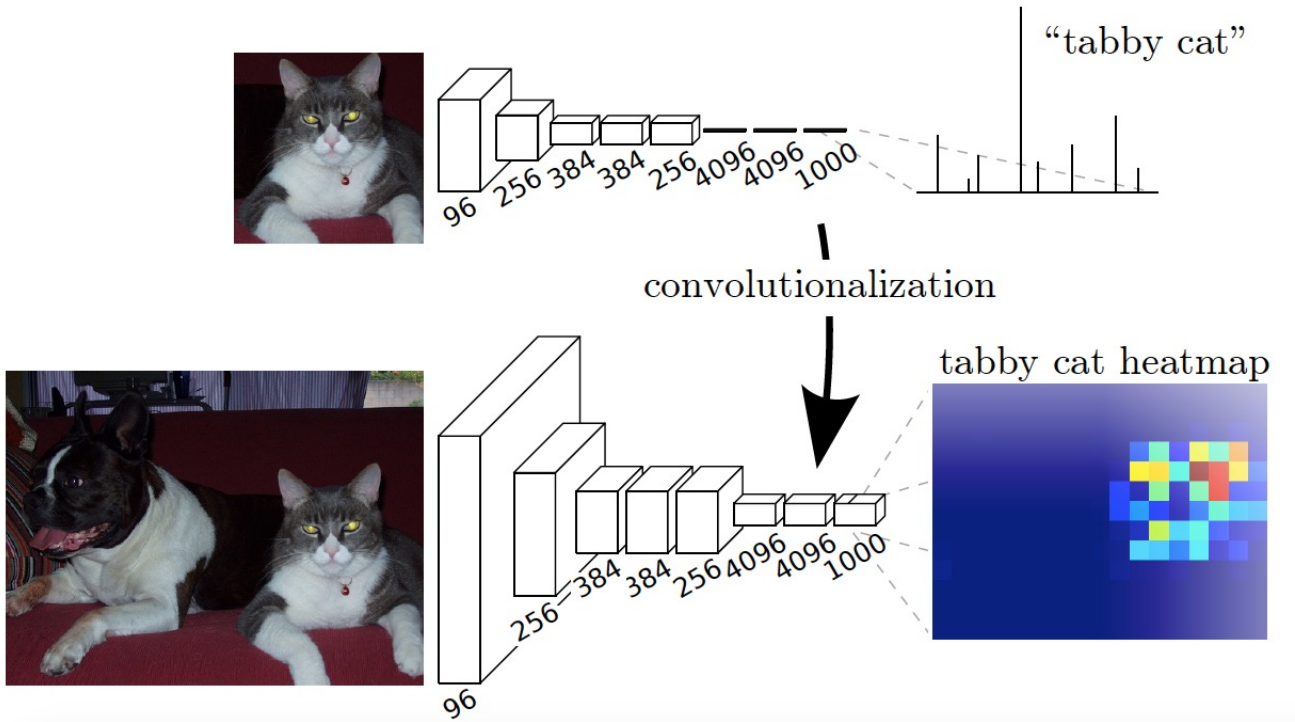


B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik
Hypercolumns for Object Segmentation and Fine-grained Localization, CVPR 2015


Have: feature maps from
image classification network
Want: pixel-wise predictions
pixelwise prediction
segmentation g.t.
96
256
384
384
256
4096
4096
21
21
Upsample
MLP
label

Fully Convolutional Networks

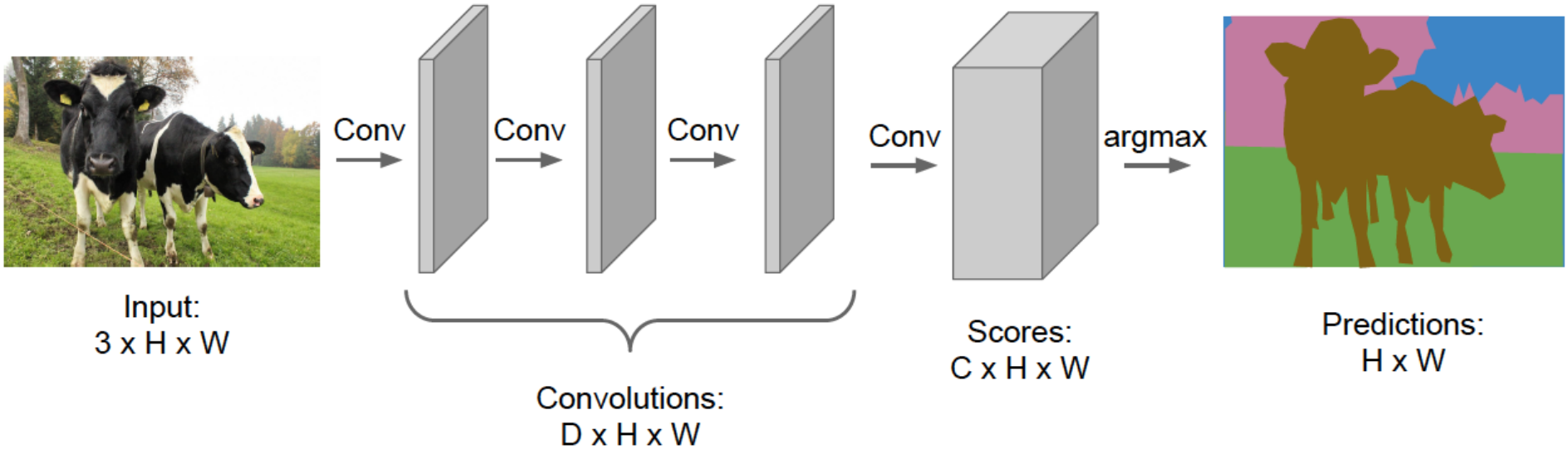
- Design a network with only convolutional layers, make predictions for all pixels at once



J. Long, E. Shelhamer, and T. Darrell
Fully Convolutional Networks for Semantic Segmentation, CVPR 2015

Fully Convolutional Networks

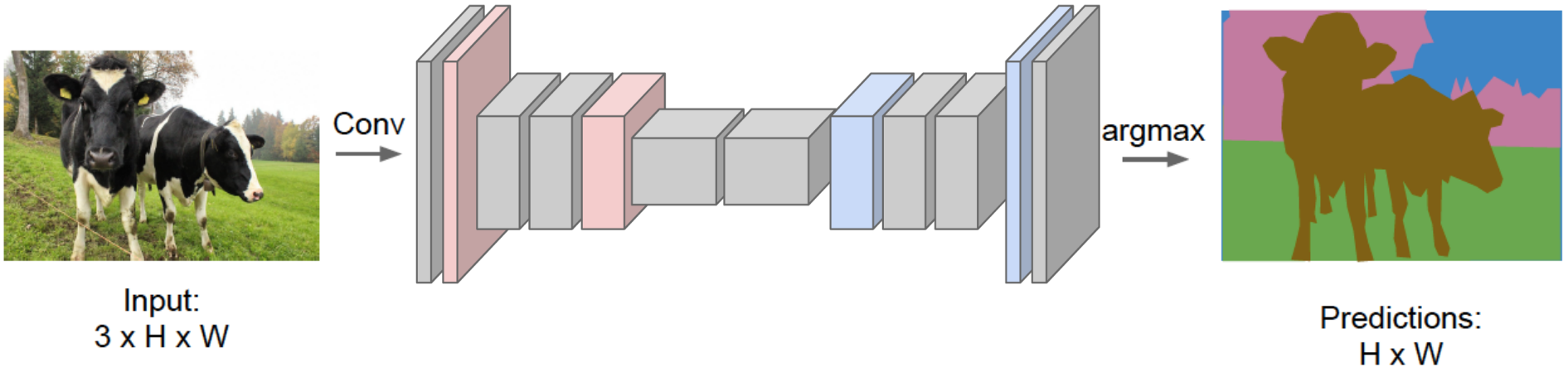
- Design a network with only convolutional layers, make predictions for all pixels at once
- Ideally: we want convolutions at full image resolution …implementing that naively is too expensive



Source: Stanford CS231n

Fully Convolutional Networks

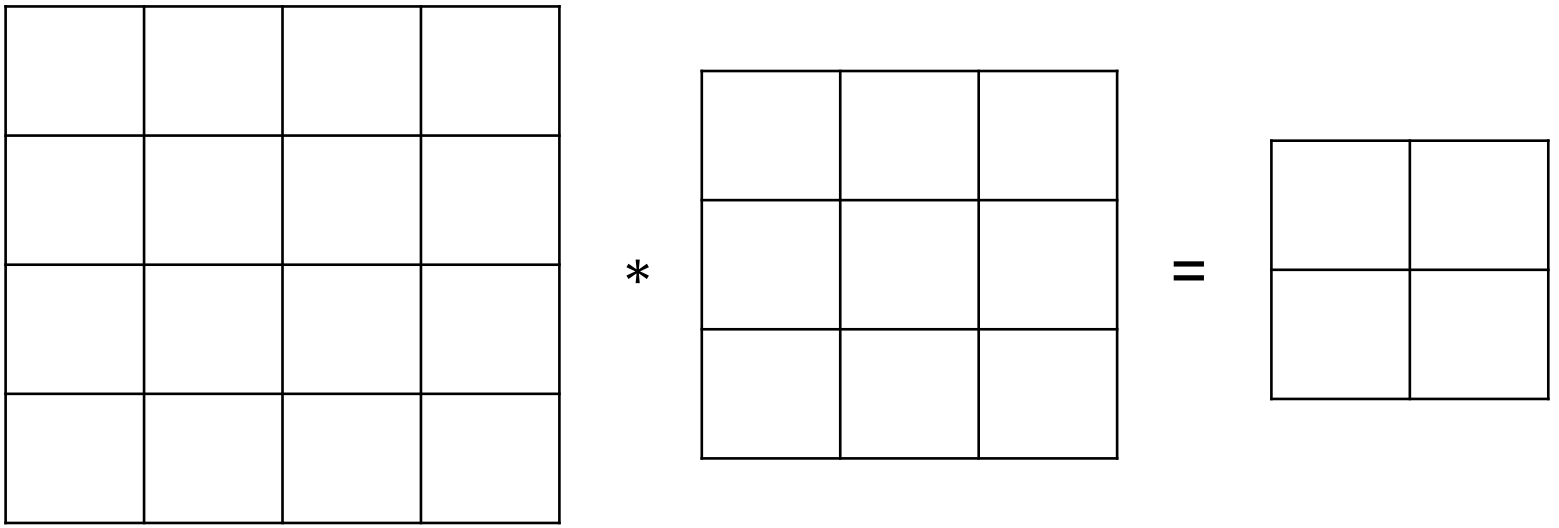
- Design a network with only convolutional layers, make predictions for all pixels at once
- Ideally: we want convolutions at full image resolution …implementing that naively is too expensive
  - Still want some form of bottleneck



Source: Stanford CS231n

Upsampling in a deep network

- Regular convolution (stride 1, pad 0)



x **w** **z**

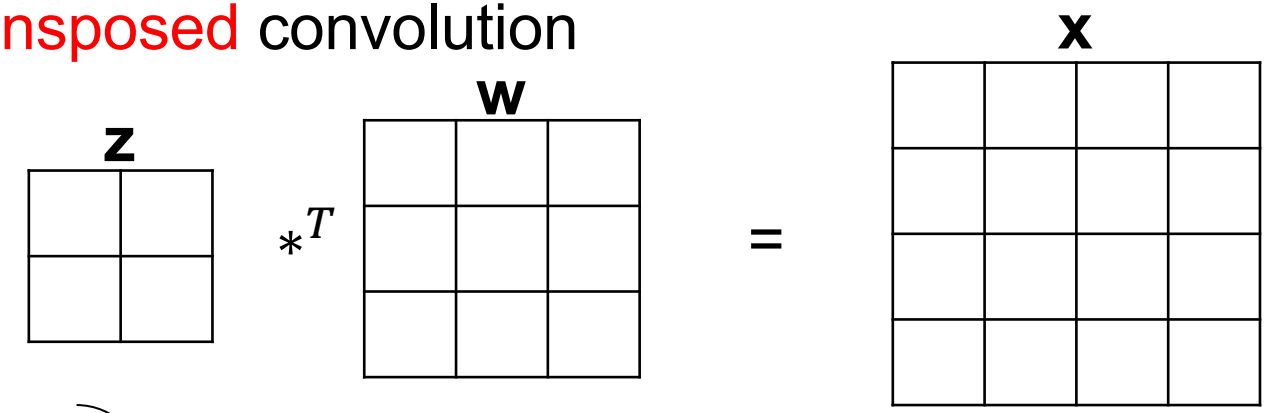
- Matrix-vector form:

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ \vdots \\ x_{44} \end{pmatrix} = \begin{pmatrix} z_{11} \\ z_{12} \\ z_{21} \\ z_{22} \end{pmatrix}$$

4x4 input, 2x2 output

Upsampling in a deep network

- Transposed convolution



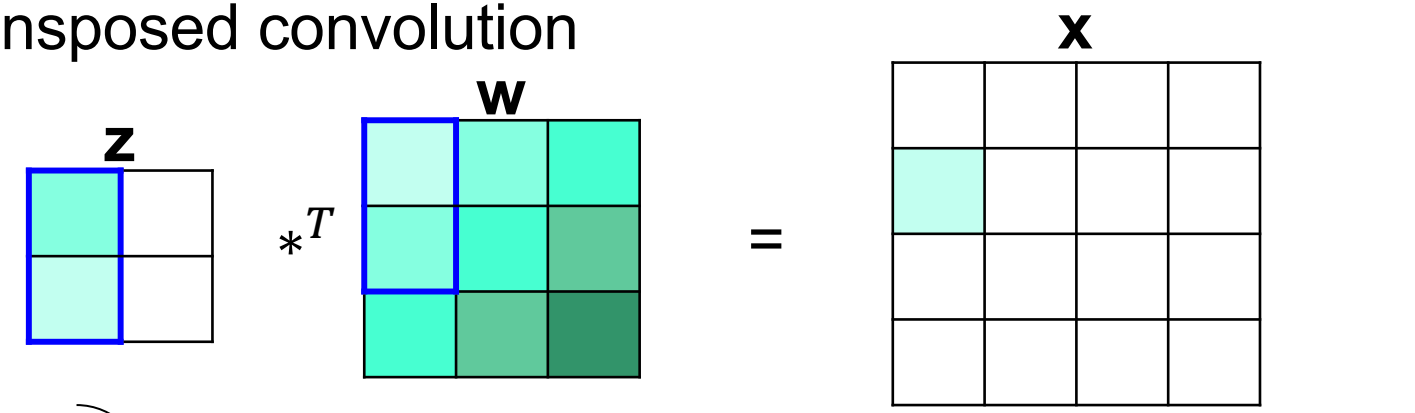
$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ z_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

2x2 input, 4x4 output

Not an inverse of the original convolution operation, simply reverses dimension change!

Upsampling in a deep network

- Transposed convolution



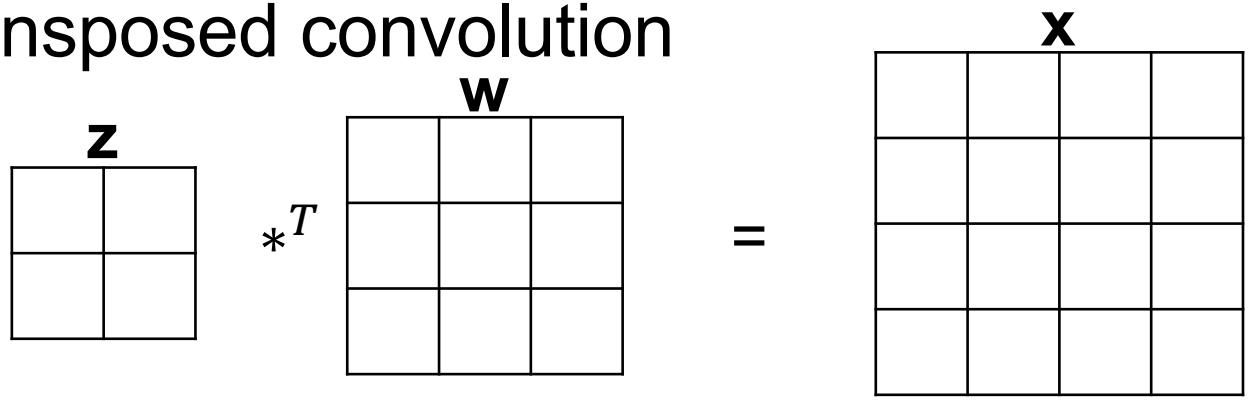
Convolve input with *flipped* filter

$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ z_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

$$x_{21} = w_{21}z_{11} + w_{11}z_{21}$$

Upsampling in a deep network

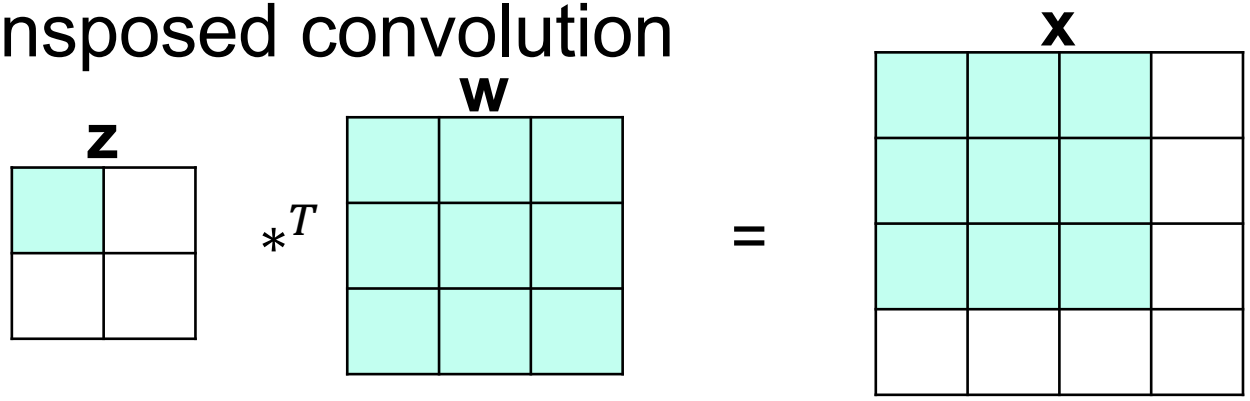
- Transposed convolution



$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ x_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

Another view

- Transposed convolution



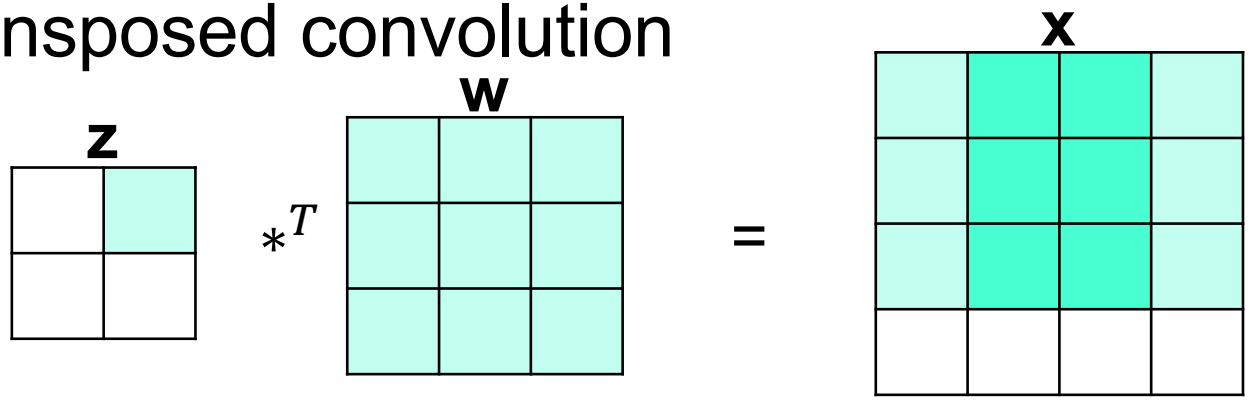
$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ x_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

Alternate view:

- Place copies of the filter on the output, weighted by entries of the input

Upsampling in a deep network

- Transposed convolution



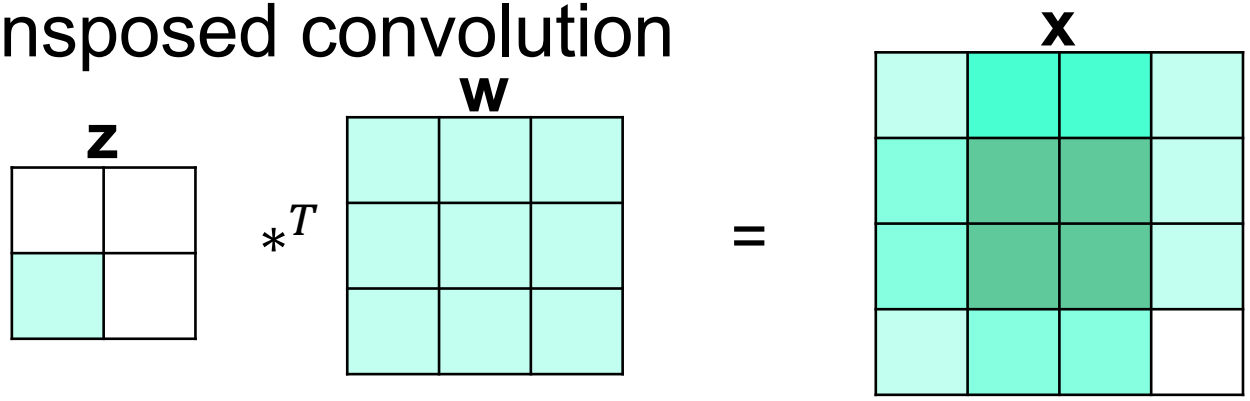
$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ x_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

Alternate view:

- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap

Upsampling in a deep network

- Transposed convolution



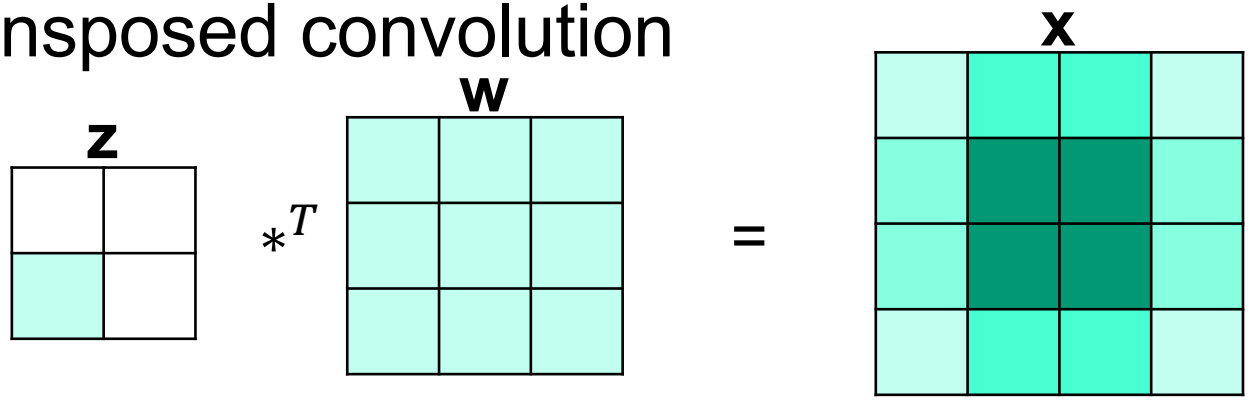
$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ x_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

Alternate view:

- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap

Upsampling in a deep network

- Transposed convolution



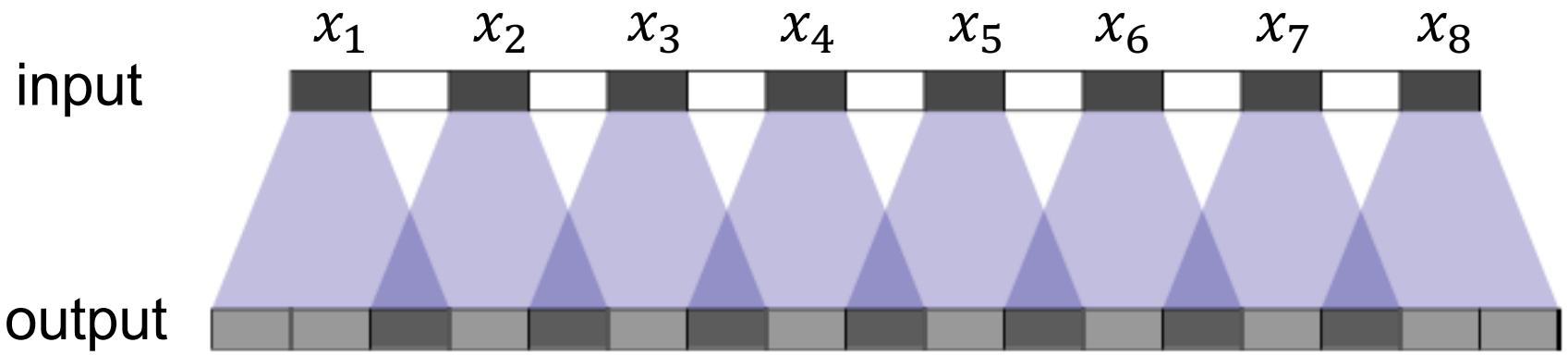
$$\begin{pmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{pmatrix} \begin{pmatrix} z_{11} \\ z_{12} \\ x_{21} \\ z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

Alternate view:

- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap

Upsampling in a deep network

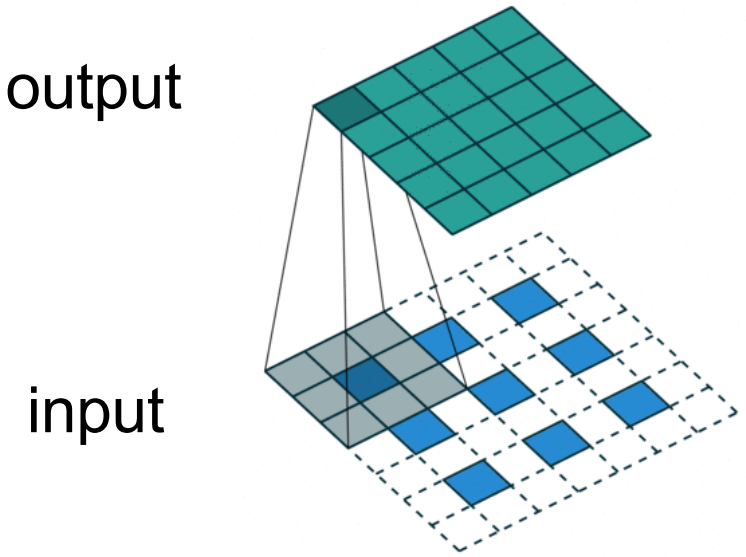
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1 (e.g. *stride 2)*



Animation: https://distill.pub/2016/deconv-checkerboard/

Upsampling in a deep network

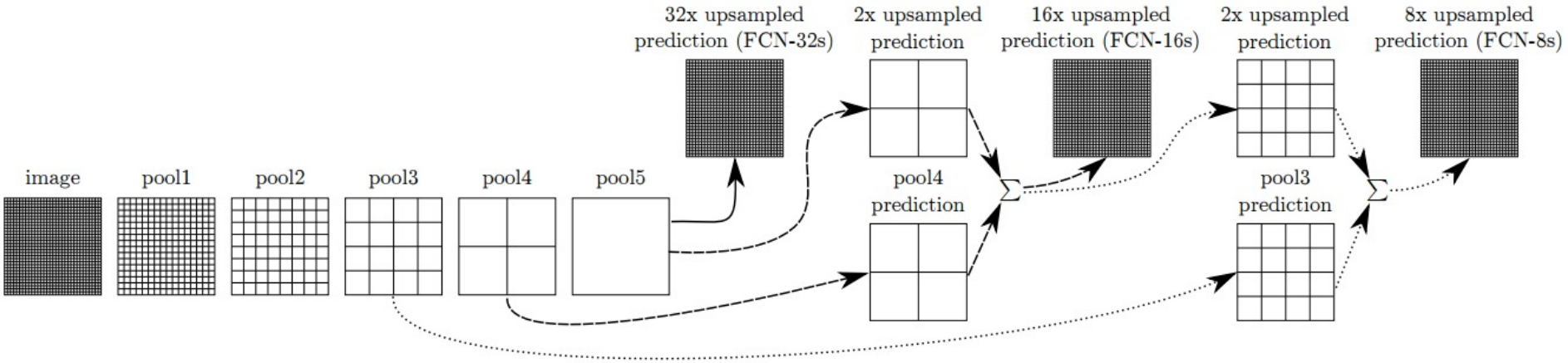
- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1
  - For stride 2, dilate the input by inserting rows and columns of zeros between adjacent entries, convolve with flipped filter
  - Sometimes called convolution with *fractional input stride* 1/2

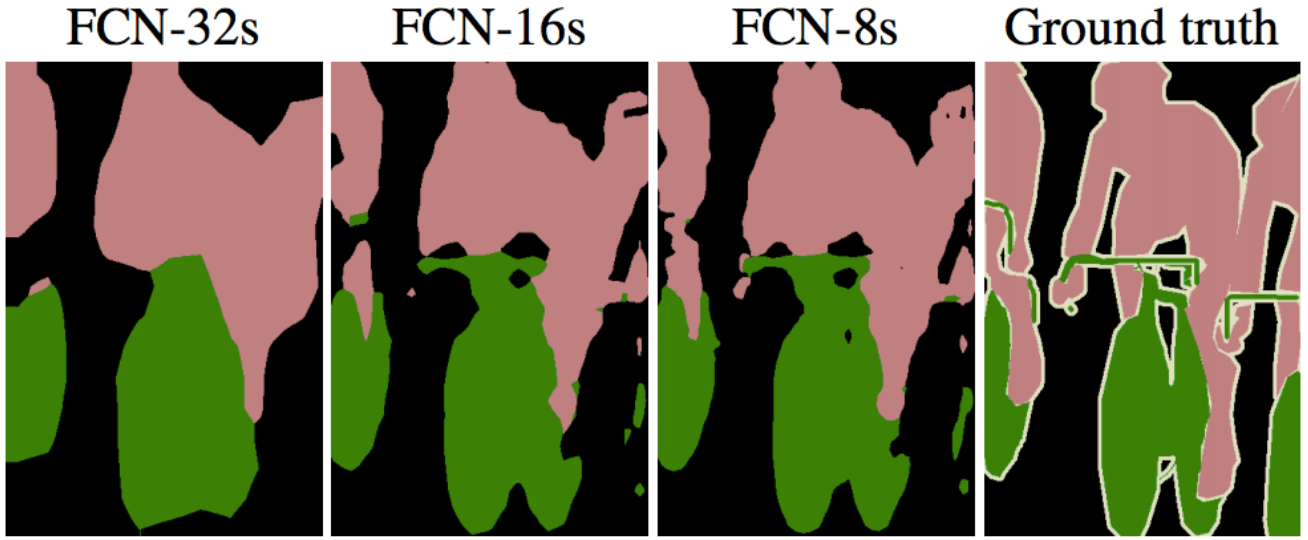


V. Dumoulin and F. Visin

A guide to convolution arithmetic for deep learning, arXiv 2018

Fully Convolutional Networks (FCN)

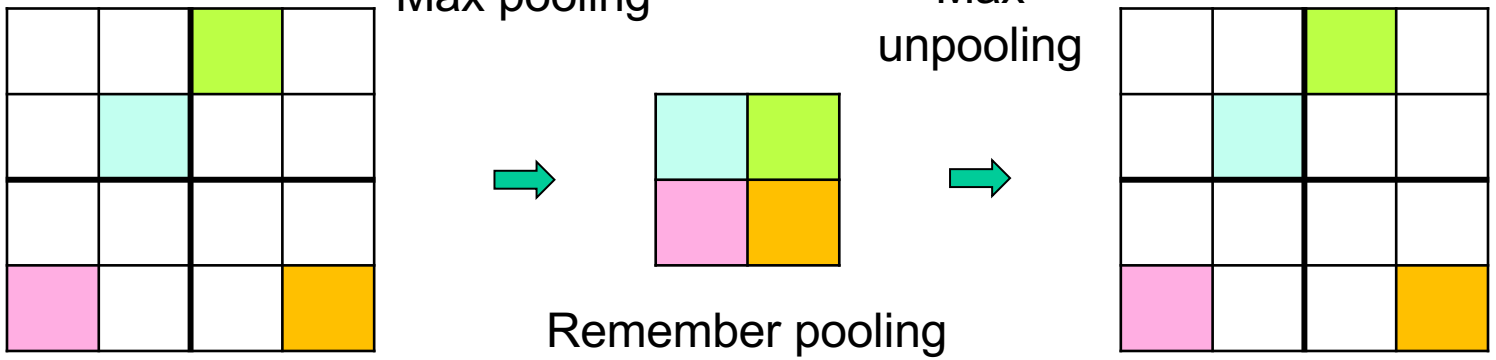




J. Long, E. Shelhamer, and T. Darrell
Fully Convolutional Networks for Semantic Segmentation, CVPR 2015

Upsampling in a deep network

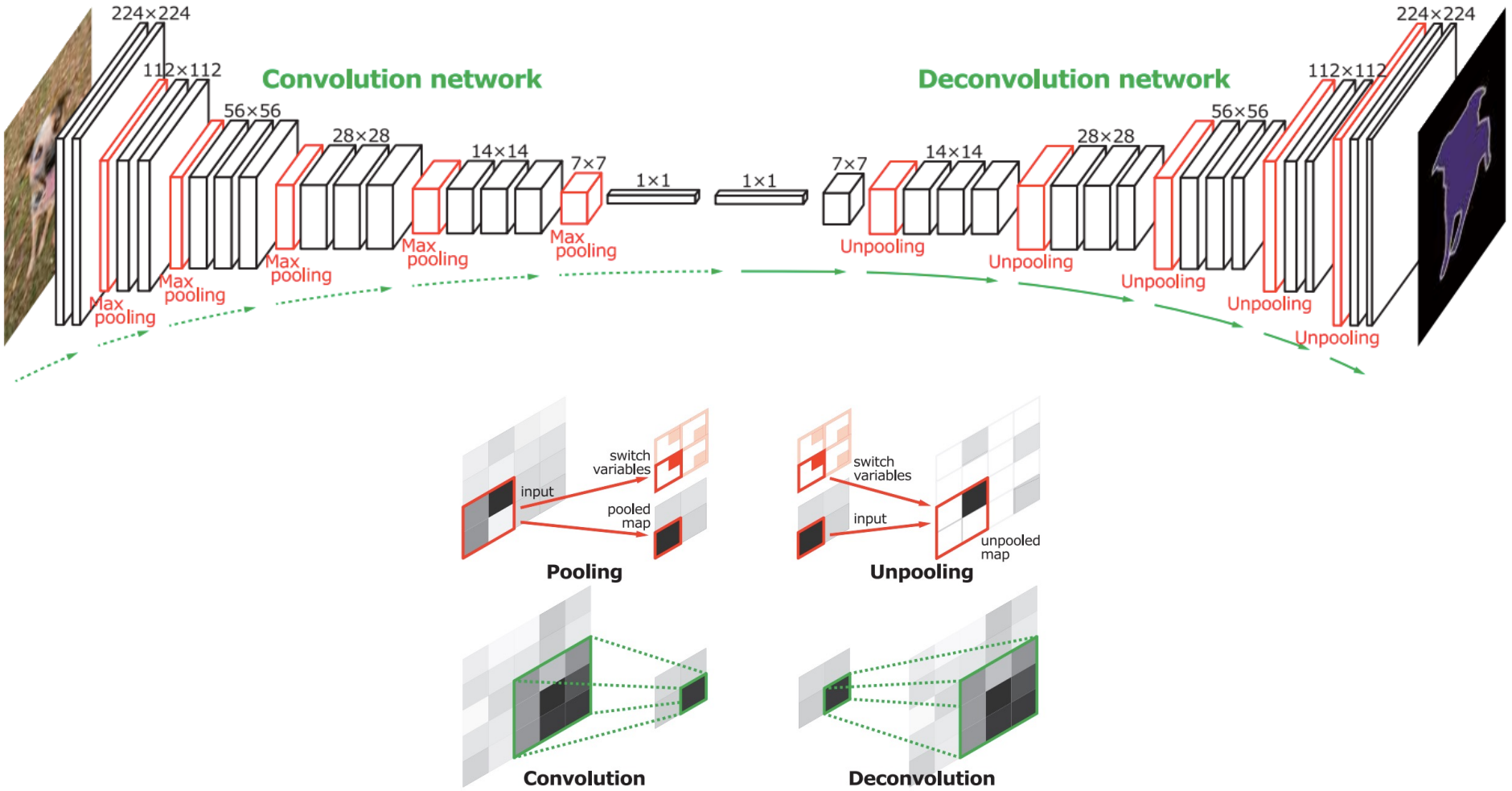
- Alternative to transposed convolution: max unpooling



Remember pooling indices (which element was max)

Output is sparse, so need to follow this with a transposed convolution layer

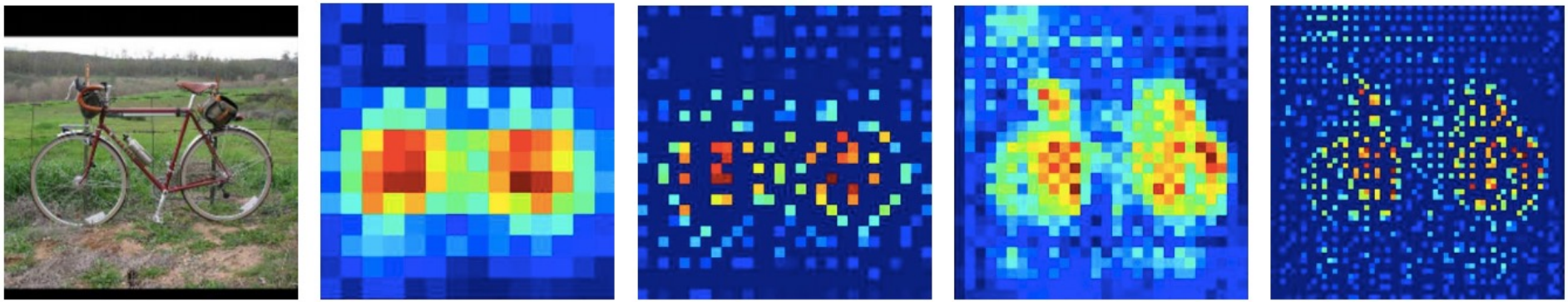
DeconvNet



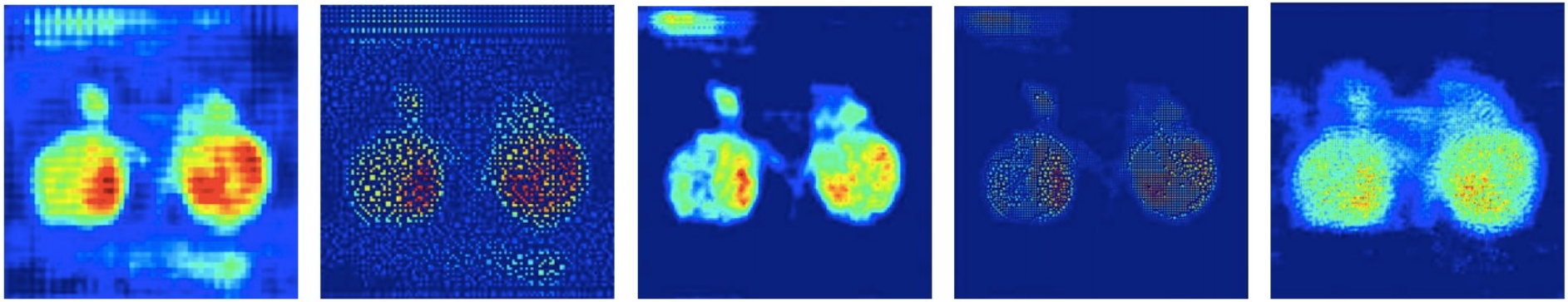
H. Noh, S. Hong, and B. Han

Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet



Original image | 14x14 deconv | 28x28 unpooling | 28x28 deconv | 54x54 unpooling



54x54 deconv | 112x112 unpooling | 112x112 deconv | 224x224 unpooling | 224x224 deconv

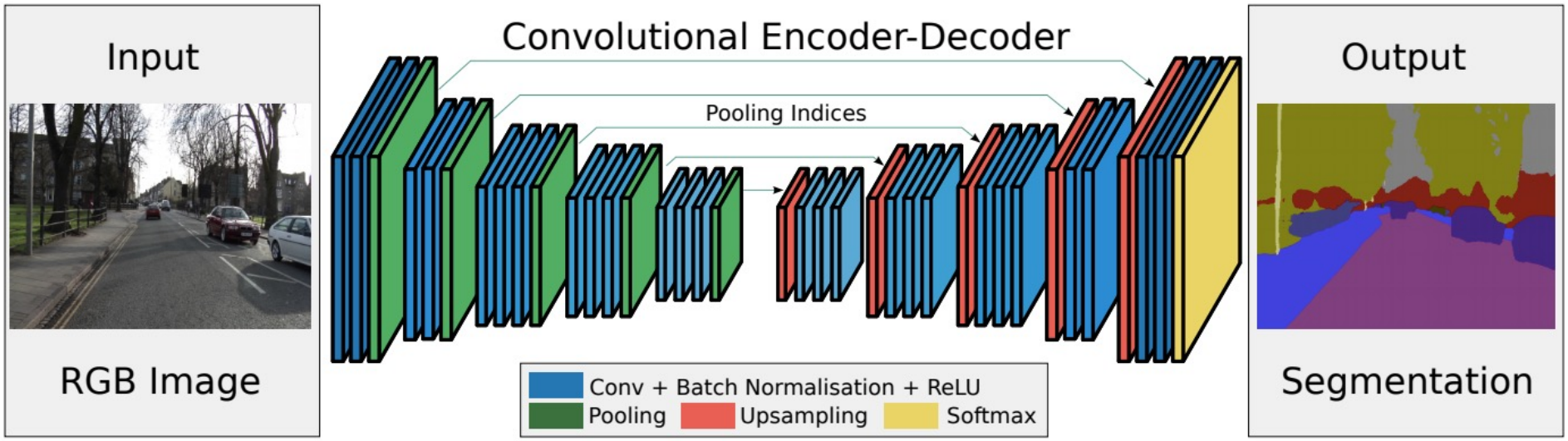
H. Noh, S. Hong, and B. Han

Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet results

PASCAL VOC 2012	mIoU
Hypercolumns	59.2
FCN-8	62.2
DeconvNet	69.6
Ensemble of DeconvNet and FCN	71.7

Similar architecture: SegNet

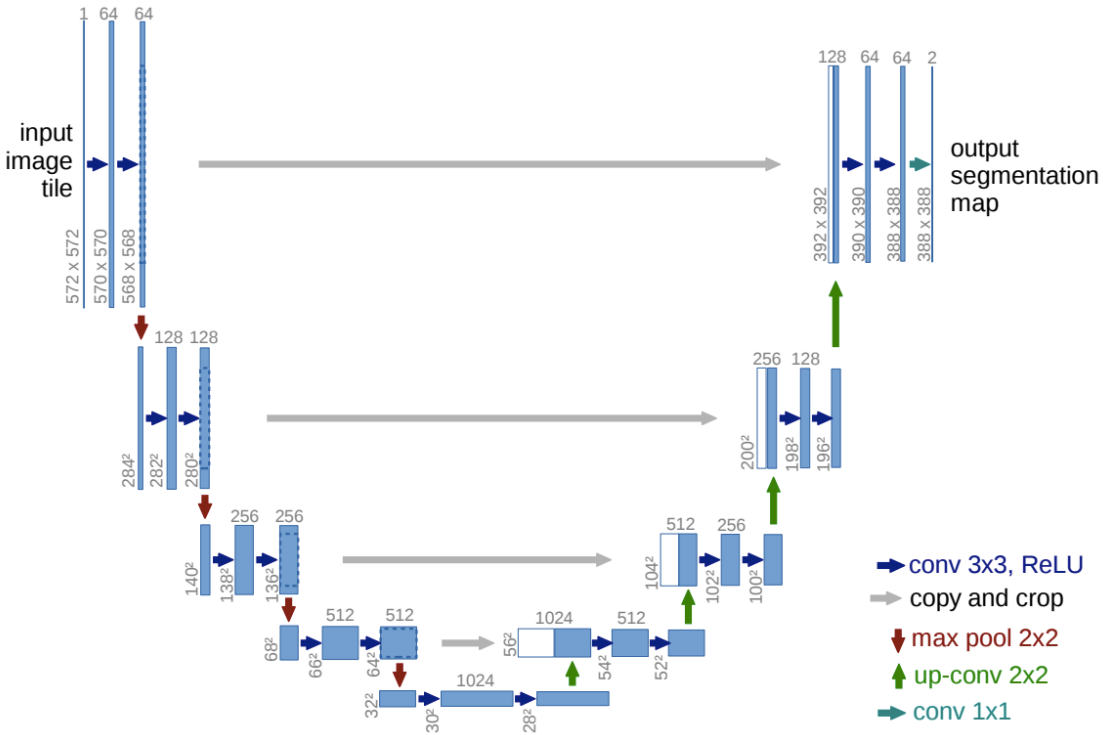


Drop the FC layers,
get better results

V. Badrinarayanan, A. Kendall and R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

U-Net

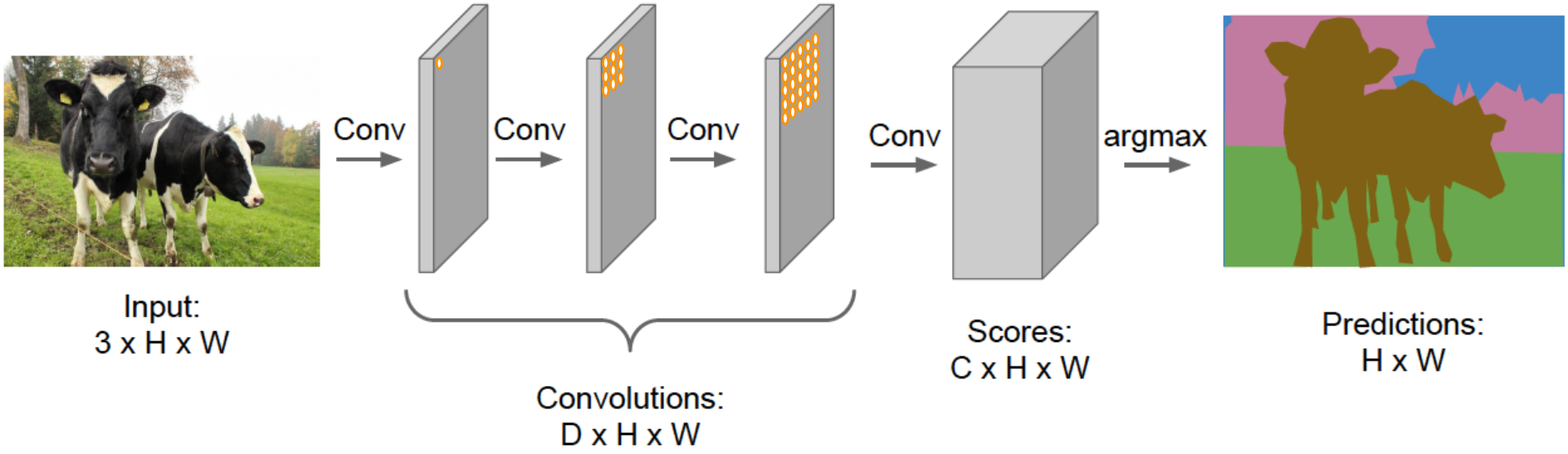
- Like FCN, fuse upsampled higher-level feature maps with higher-res, lower-level feature maps
- Unlike FCN, fuse by concatenation, predict at the end



O. Ronneberger, P. Fischer, T. Brox

U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

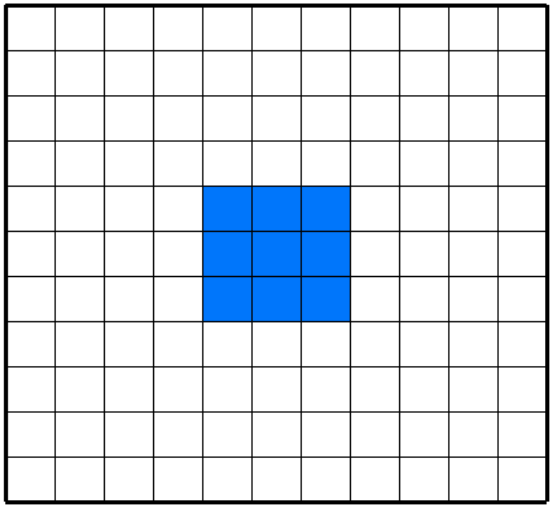
Receptive field



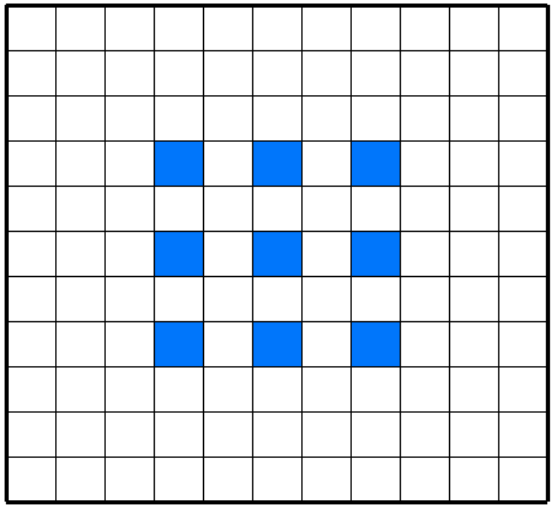
Dilated convolutions

- Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter
  - Also known as atrous convolution

Dilation factor 1



Dilation factor 2



Dilation factor 3

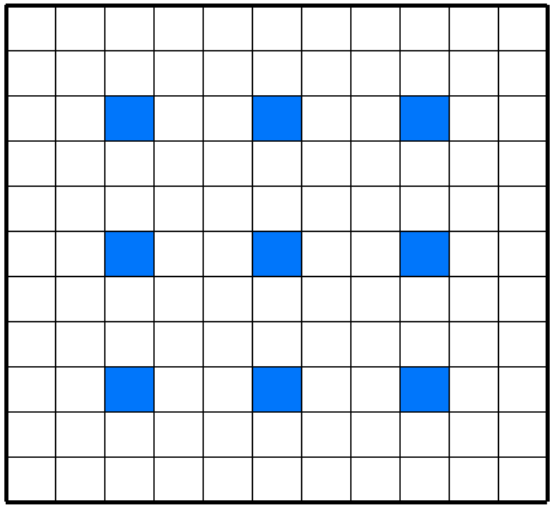
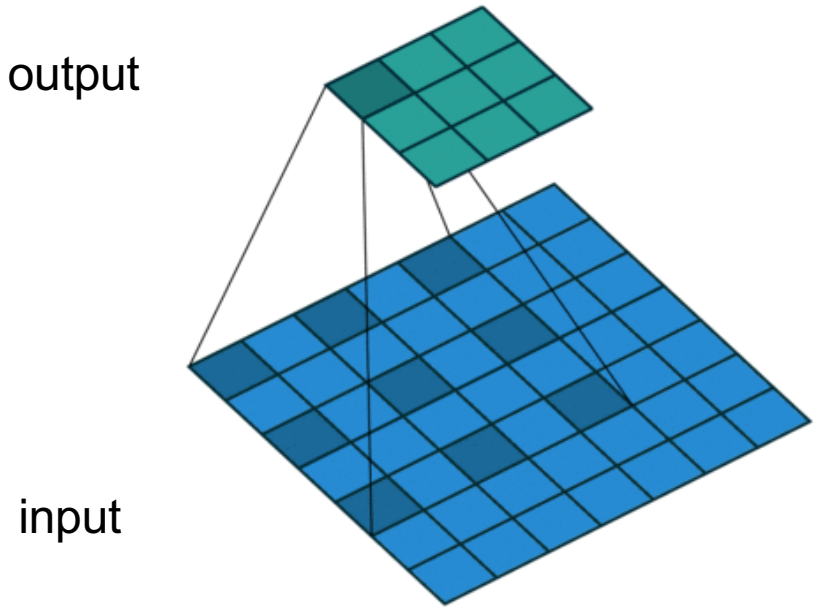


Image source

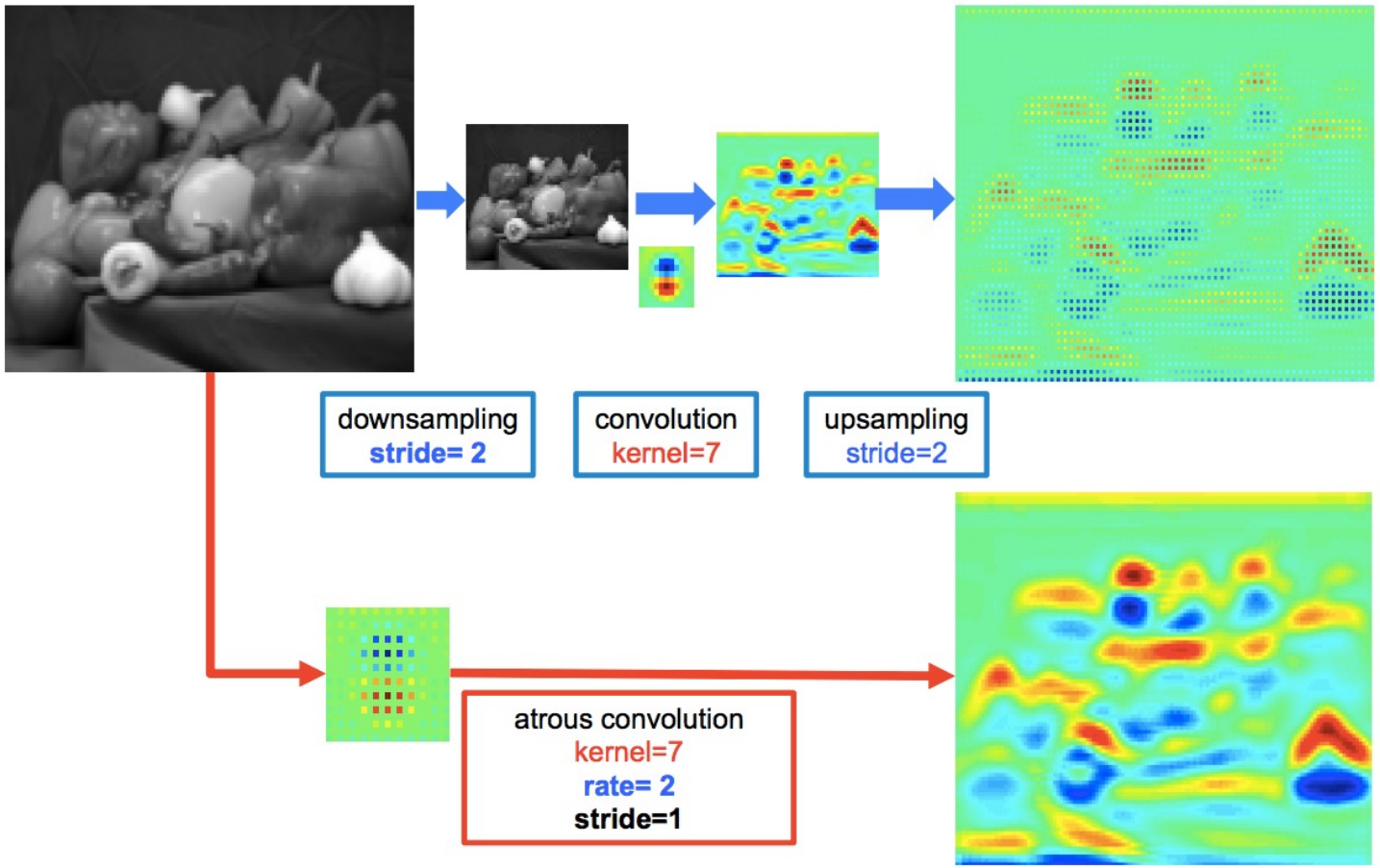
Dilated convolutions

- Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter



V. Dumoulin and F. Visin
A guide to convolution arithmetic for deep learning, arXiv 2018

Dilated convolutions



L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. Yuille, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, PAMI 2017

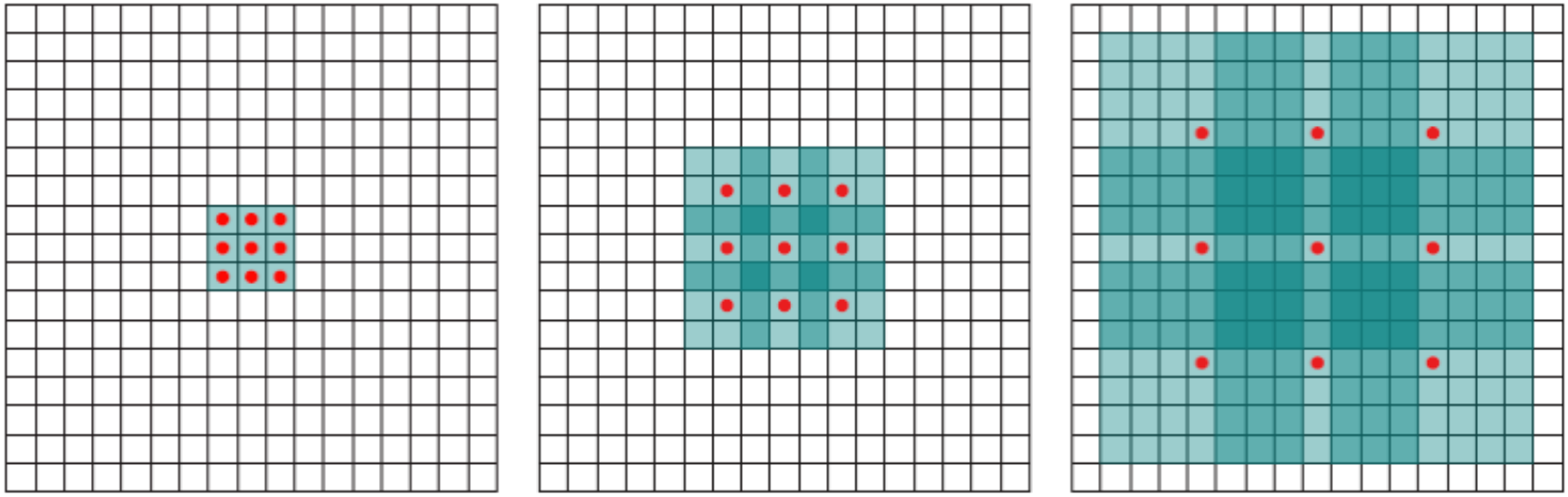
Dilated convolutions

- Can increase receptive field size exponentially with a linear growth in the number of parameters

Feature map 1 (F1) produced from F0 by 1-dilated convolution

F2 produced from F1 by 2-dilated convolution

F3 produced from F2 by 4-dilated convolution



Receptive field: 3x3

Receptive field: 7x7

Receptive field: 15x15

F. Yu and V. Koltun

Multi-scale context aggregation by dilated convolutions, ICLR 2016

Dilated convolutions

- Context module with dilation
  - Returns same number of feature maps at the same resolution as the input, so can be plugged in to replace components of existing dense prediction architectures

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67
Output channels								
Basic	C	C	C	C	C	C	C	C
Large	$2C$	$2C$	$4C$	$8C$	$16C$	$32C$	$32C$	C

F. Yu and V. Koltun

Multi-scale context aggregation by dilated convolutions, ICLR 2016

Dilated convolutions: Evaluation

Results on VOC 2012

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean IoU
Front end	86.3	38.2	76.8	**66.8**	63.2	87.3	78.7	82	33.7	76.7	53.5	73.7	76	76.6	83	**51.9**	77.8	44	79.9	**66.3**	69.8
Front + Basic	86.4	37.6	78.5	66.3	64.1	89.9	79.9	84.9	**36.1**	79.4	**55.8**	77.6	81.6	79	83.1	51.2	81.3	43.7	82.3	65.7	71.3
Front + Large	**87.3**	**39.2**	**80.3**	65.6	**66.4**	**90.2**	**82.6**	**85.8**	34.8	**81.9**	51.7	**79**	**84.1**	**80.9**	**83.2**	51.2	**83.2**	**44.7**	**83.4**	65.6	**72.1**

*Front end: re-implementation of **FCN-8** with last two pooling layers dropped (5% better than original FCN-8)

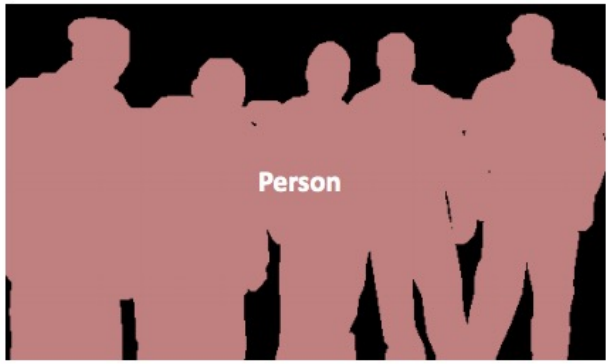
Dilated convolutions: Evaluation

(a) Image (b) FCN-8s (c) DeepLab (d) Our front end

Instance segmentation

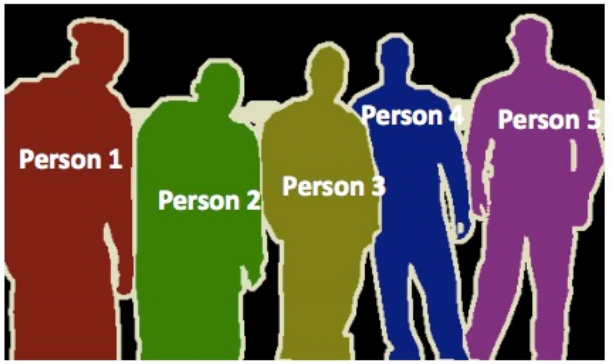
Object Detection

✓



Semantic Segmentation

✓



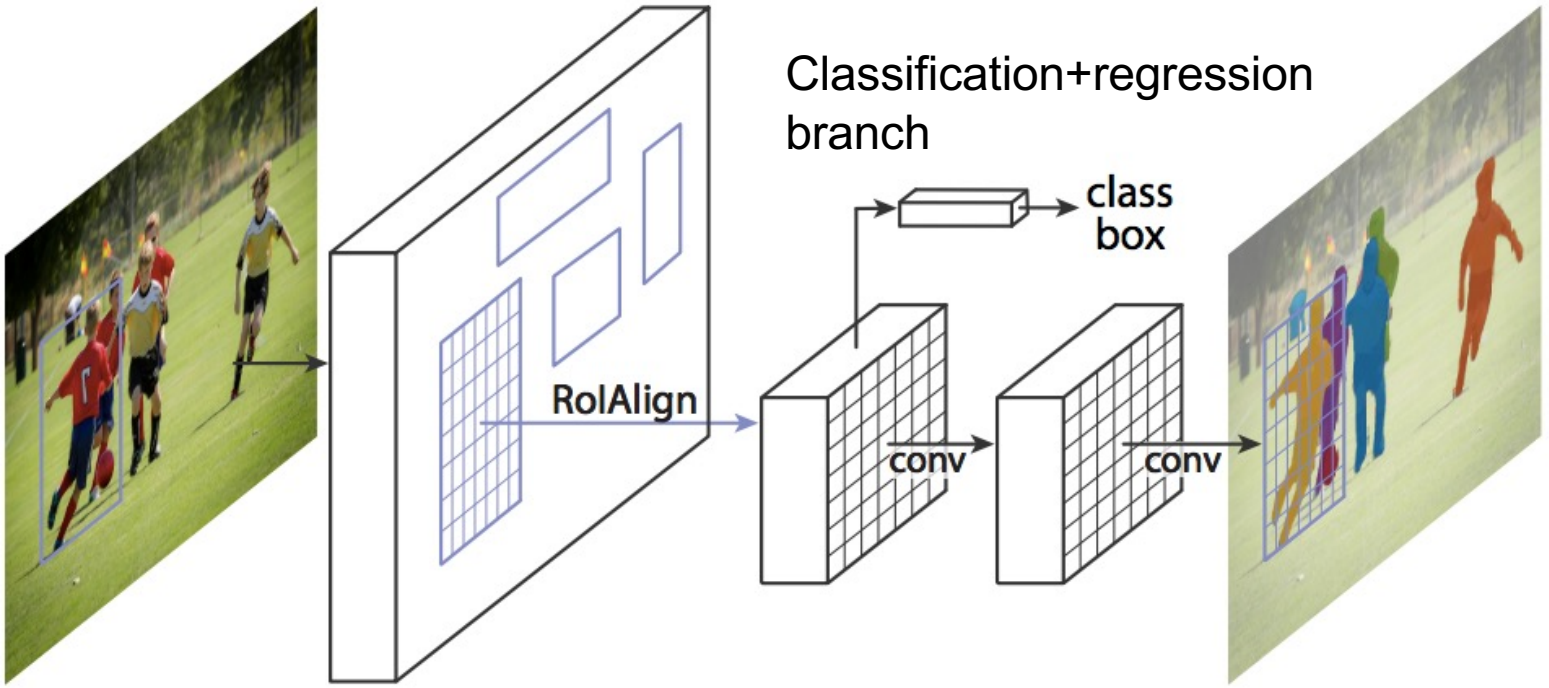
Instance Segmentation

?

Source: Kaiming He

Mask R-CNN

- Mask R-CNN = Faster R-CNN + FCN on RoIs

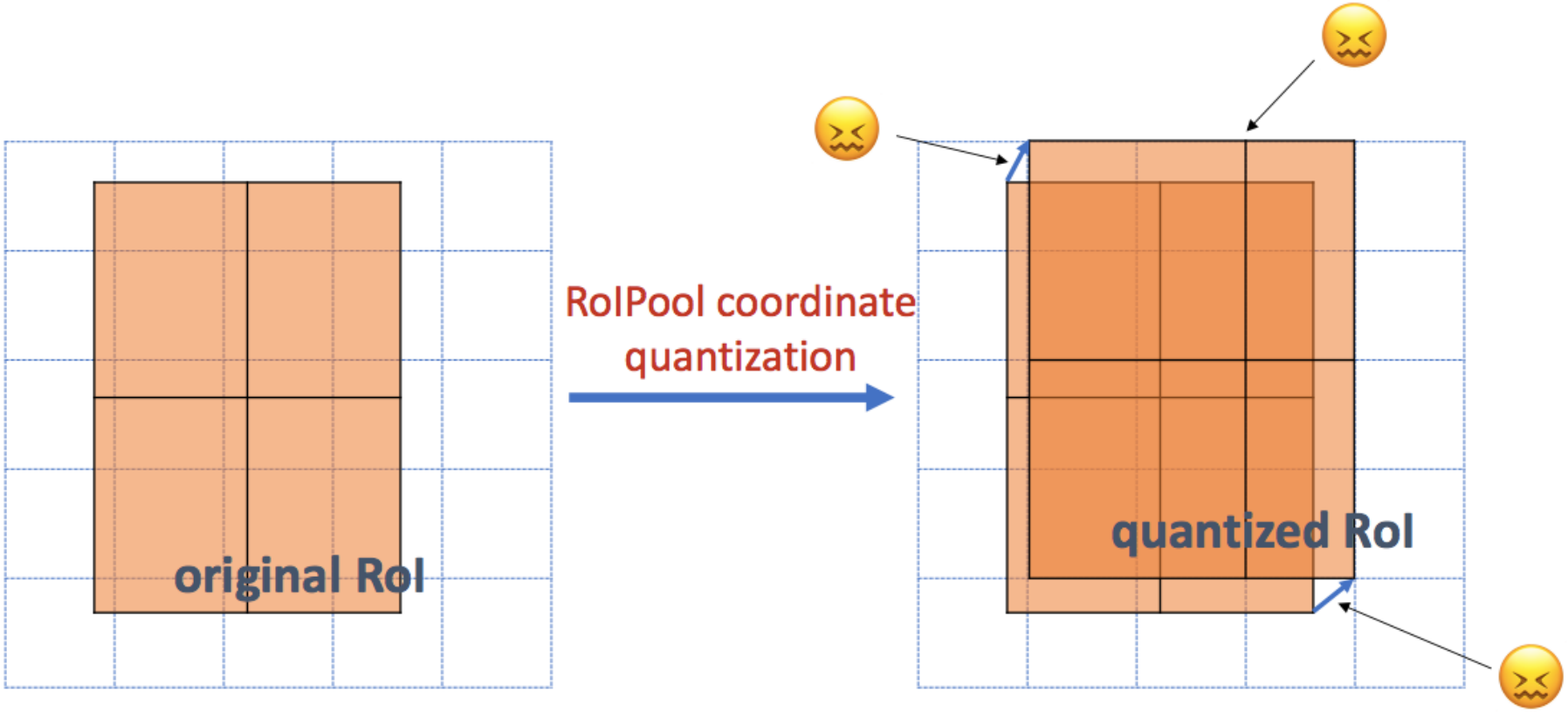


Mask branch: separately predict segmentation for each possible class

K. He, G. Gkioxari, P. Dollar, and R. Girshick
Mask R-CNN, ICCV 2017 (Best Paper Award)

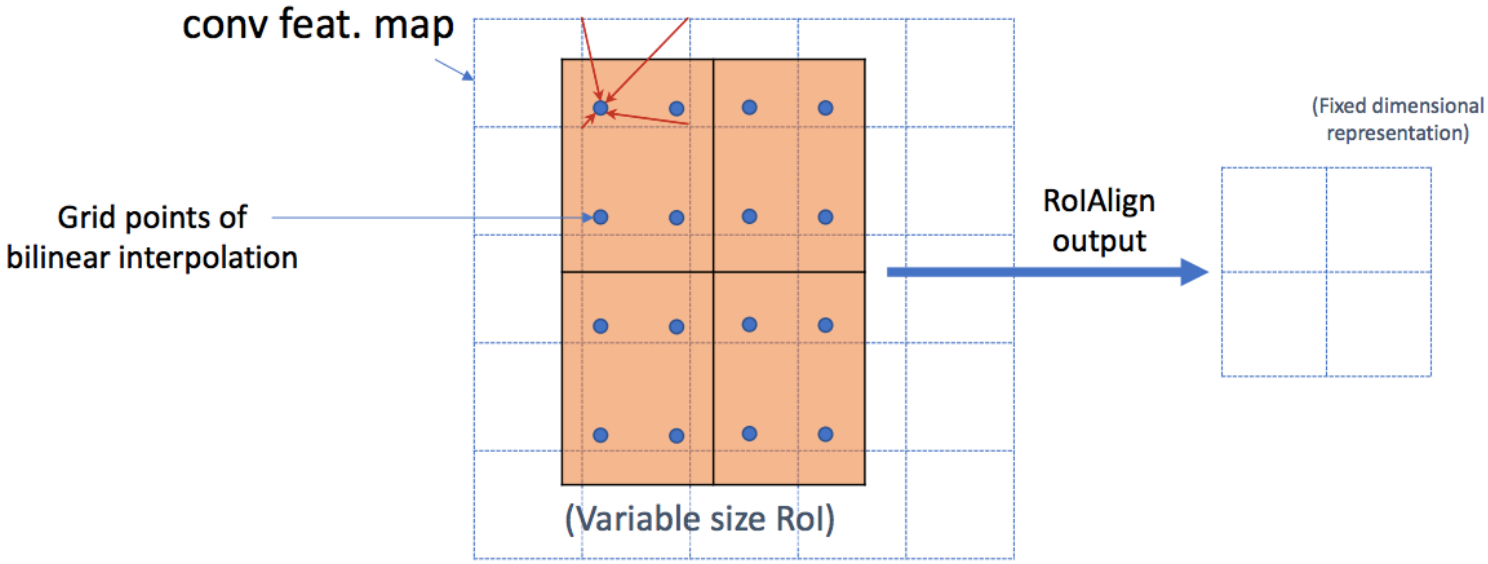
(remember) Fast R-CNN?

- RoIPool: nearest neighbor quantization



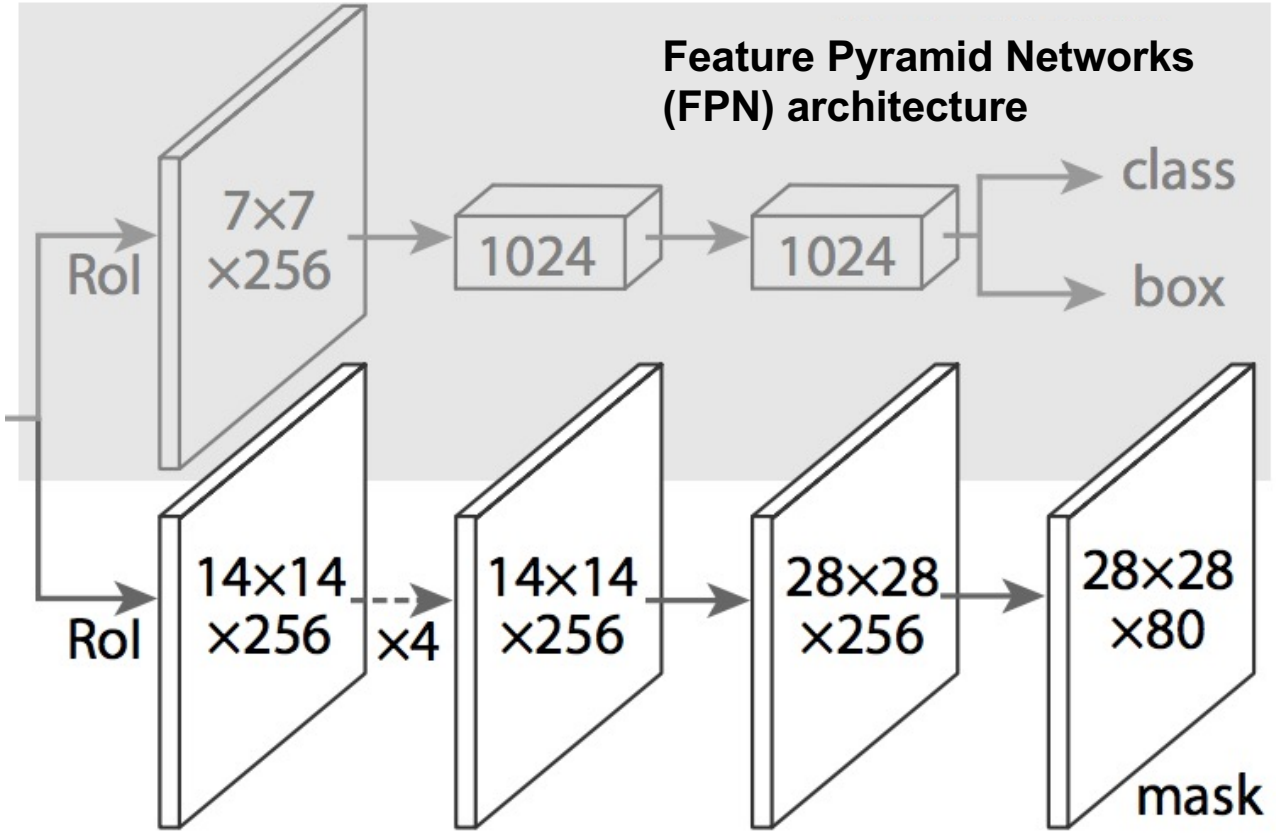
Mask R-CNN

- RoIAlign: bilinear interpolation



Mask R-CNN

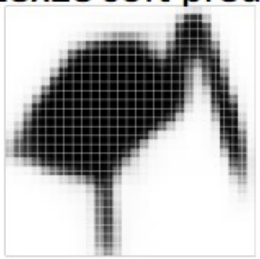
- From RoIAlign features, predict class label, bounding box, and segmentation mask



Mask R-CNN

Validation image with box detection shown in red

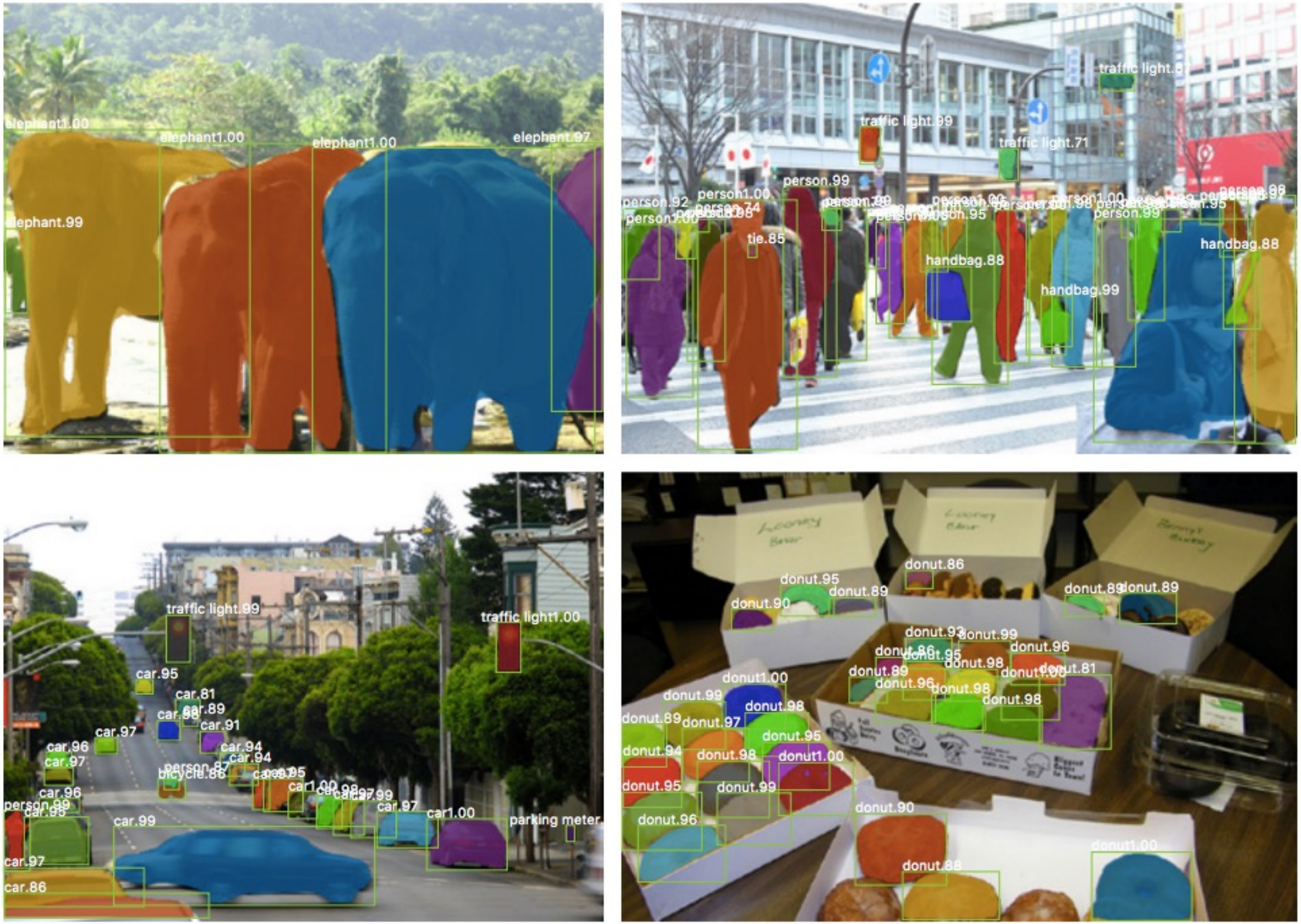
28x28 soft prediction



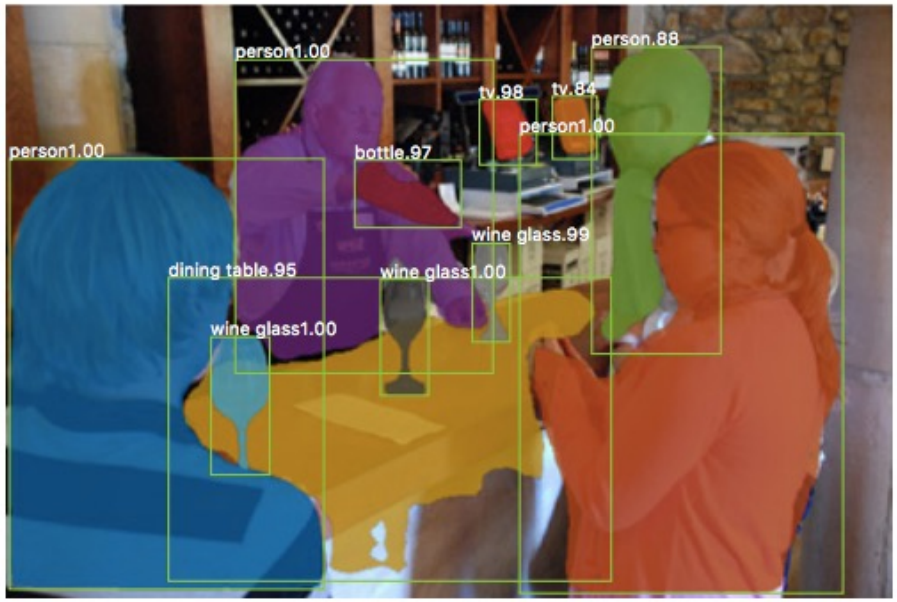
Resized Soft prediction

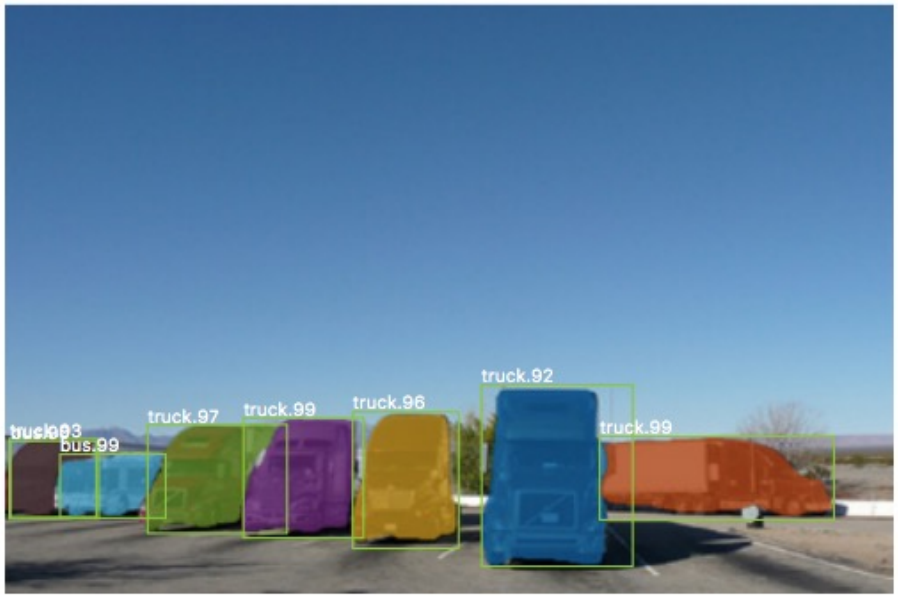
Final mask

Mask R-CNN: Example results



Mask R-CNN: Example results





Mask R-CNN: evaluation on COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

AP at different IoU thresholds

AP for different size instances

Semantic Instance Segmentation via Deep Metric Learning

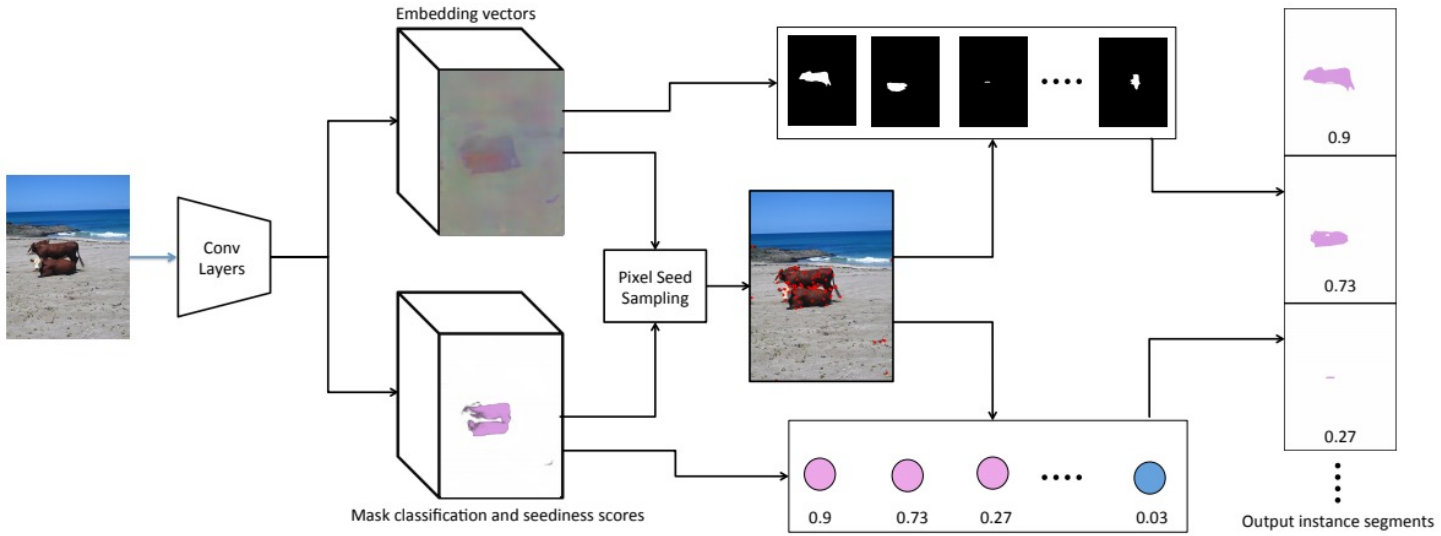
Alireza Fathi* Zbigniew Wojna* Vivek Rathod* Peng Wang† Hyun Oh Song*
Sergio Guadarrama* Kevin P. Murphy*

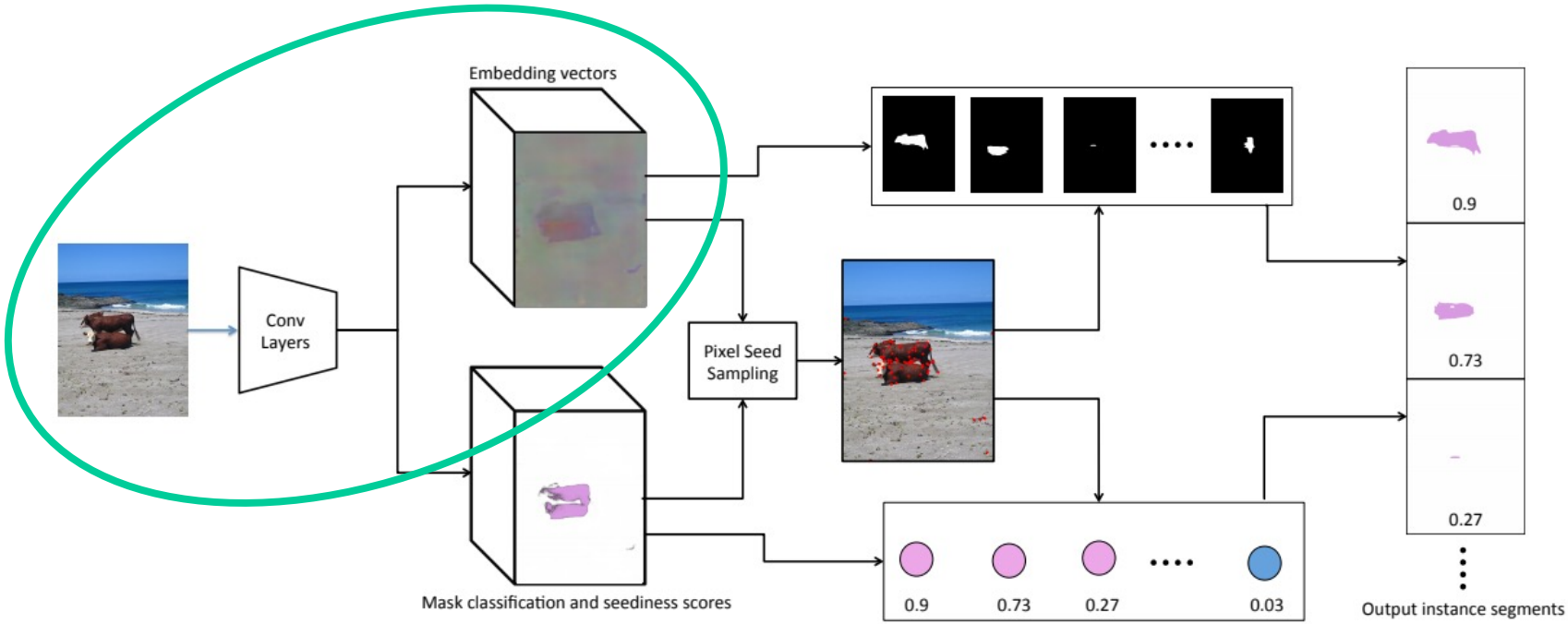
Abstract

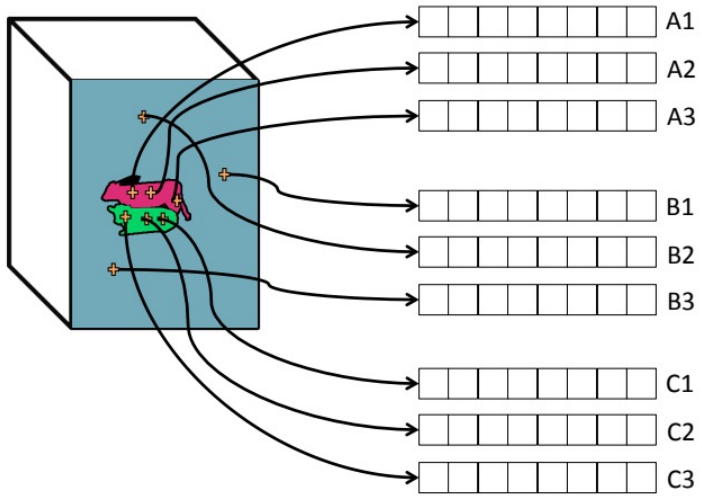
We propose a new method for semantic instance segmentation, by first computing how likely two pixels are to belong to the same object, and then by grouping similar pixels together. Our similarity metric is based on a deep, fully convolutional embedding model. Our grouping method is

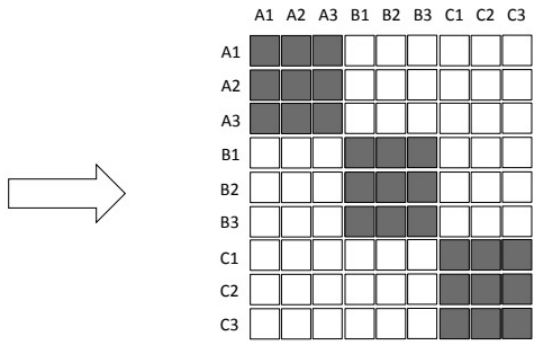
non-axis-aligned objects like ships seen from the air.)

Recently there has been a move towards "box-free" methods, that try to directly predict the mask for each object (e.g., [20, 21, 13, 17]). The most common approach to this is to modify the Faster RCNN architecture [23] so that at each point, it predicts a "centeredness" score (the probability the current pixel is the center of an object instance),

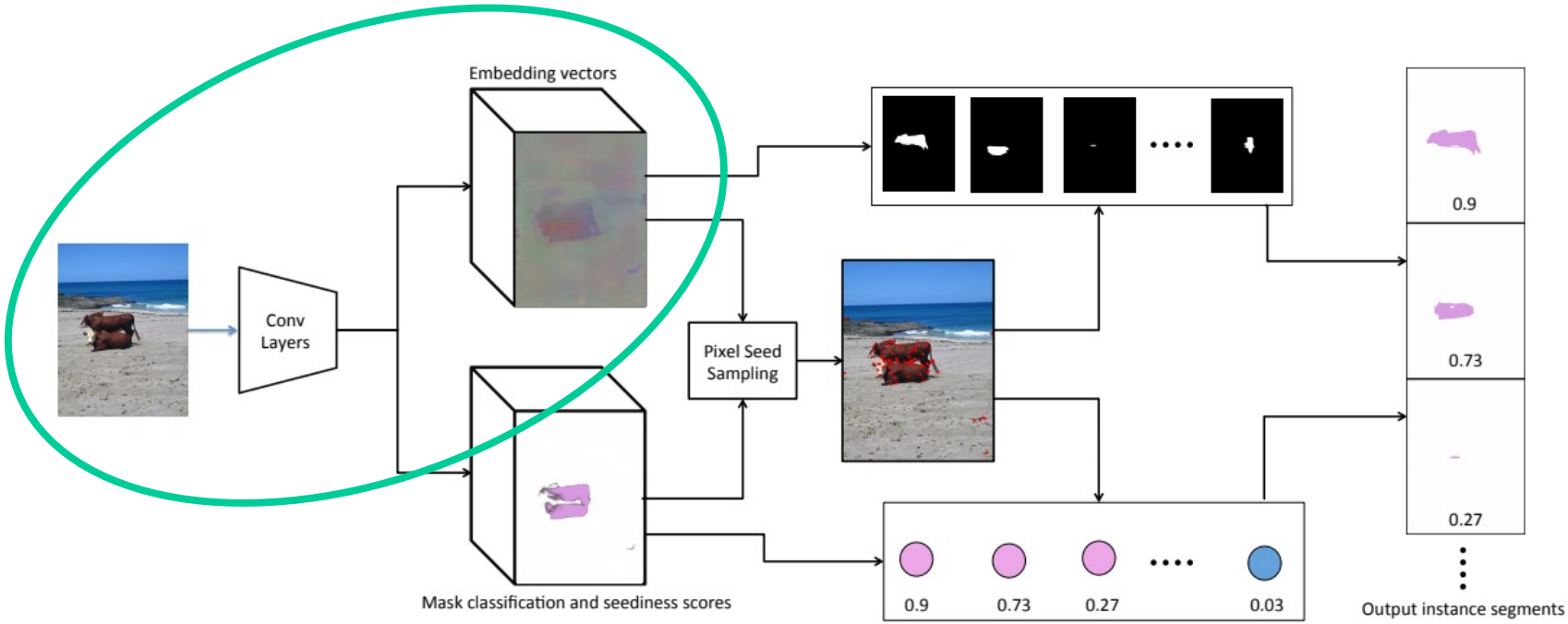


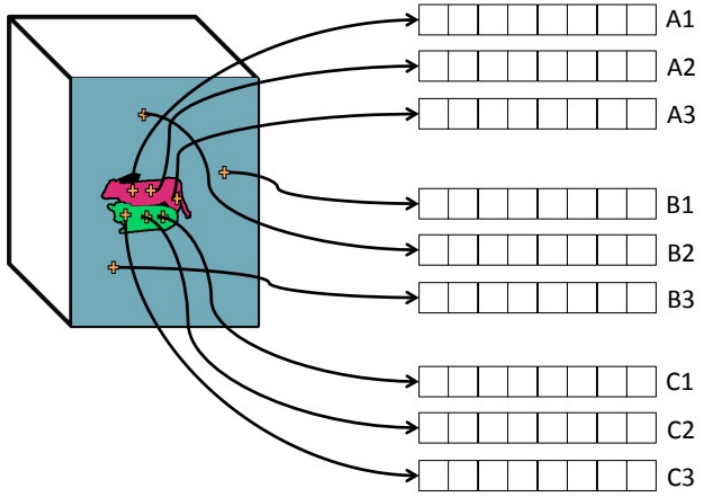






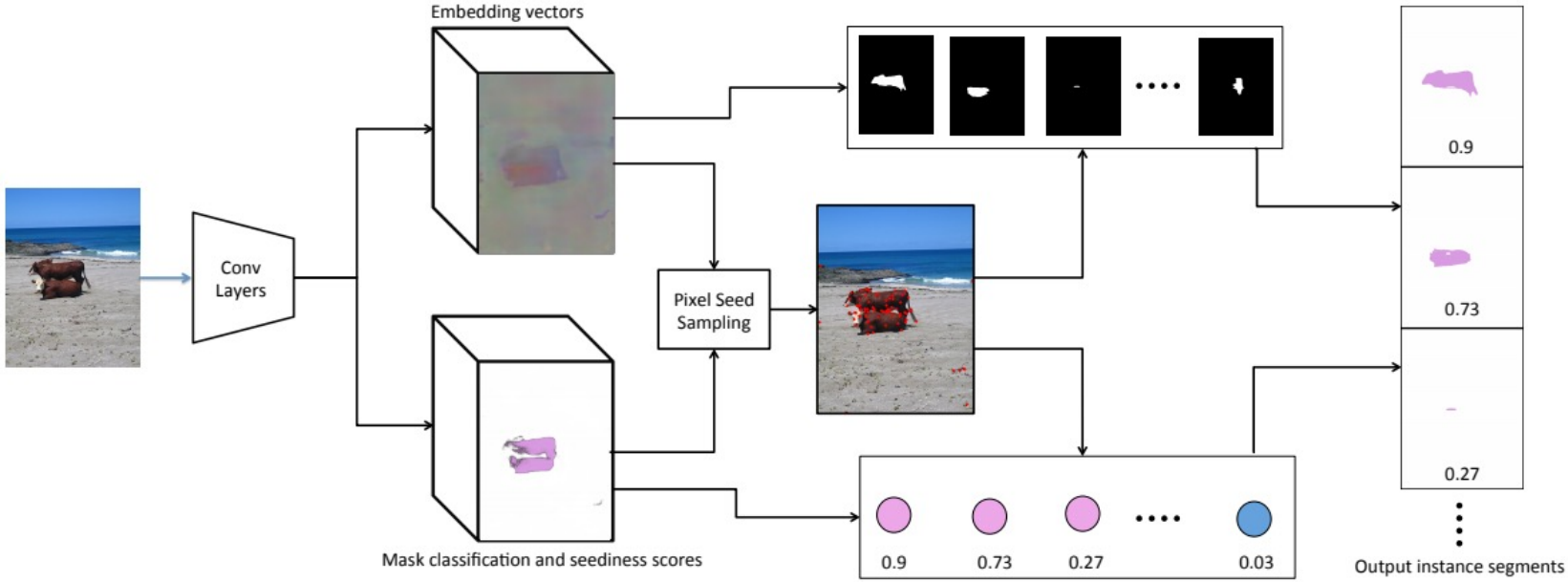
Sigmoid cross entropy loss on the similarity between pairs of embedding vectors

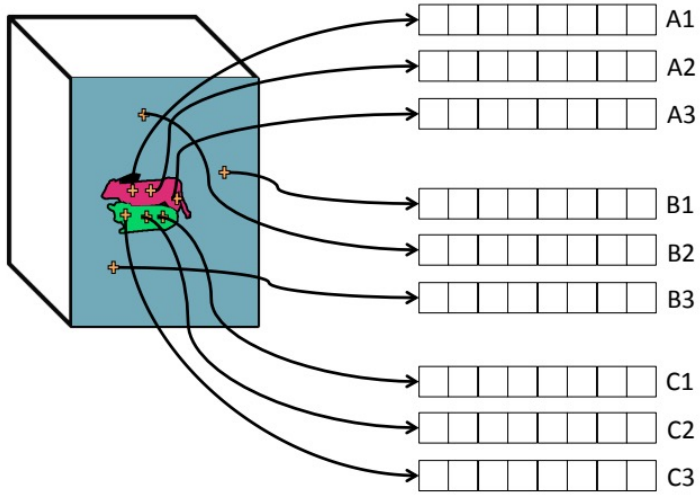




$$\mathcal{L}_e = -\frac{1}{|S|} \sum_{p,q \in S} w_{pq} \left[1_{\{y_p = y_q\}} \log(\sigma(p,q)) + 1_{\{y_p \neq y_q\}} \log(1 - \sigma(p,q)) \right]$$

$$\sigma(p,q) = \frac{2}{1 + \exp(||e_p - e_q||_2^2)} \qquad (1)$$





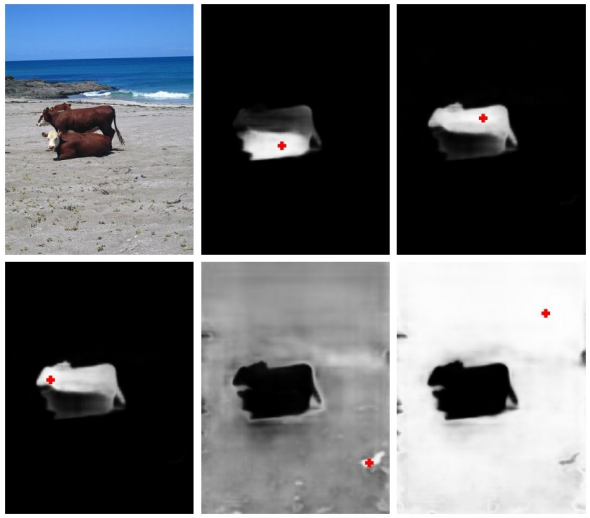
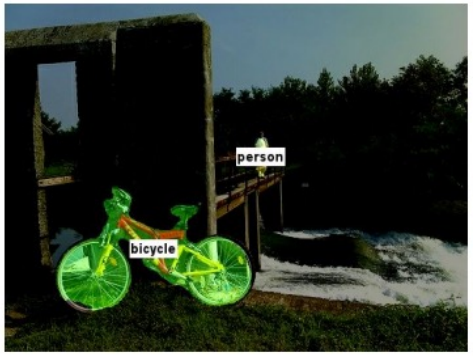


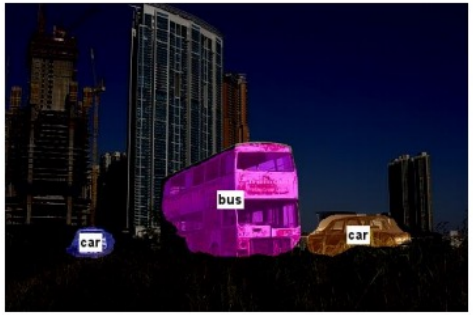
Figure 4. We visualize the similarity of each pixel and a randomly chosen seed pixel in each image. The randomly chosen seed pixel is shown by a red mark in the image. The brighter the pixels the higher the similarity.


bird
bird

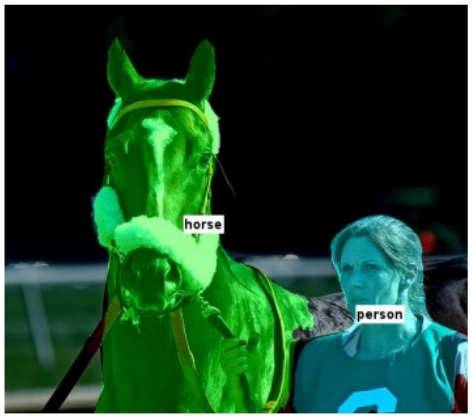

car
car
train

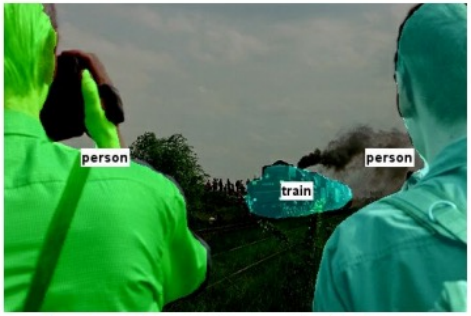

person
bicycle

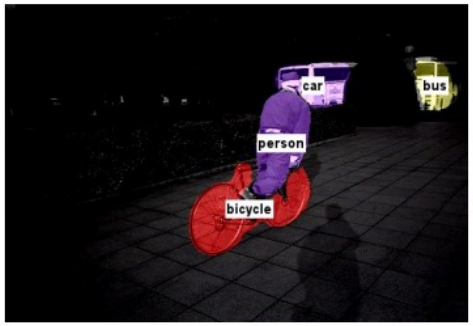

train
person

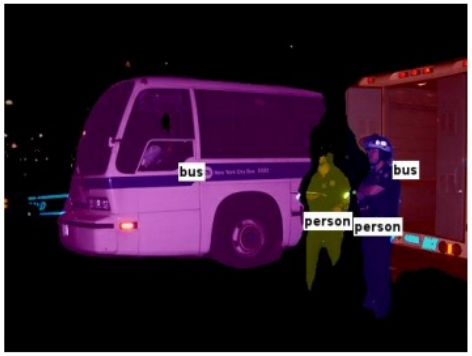

bus
car
car

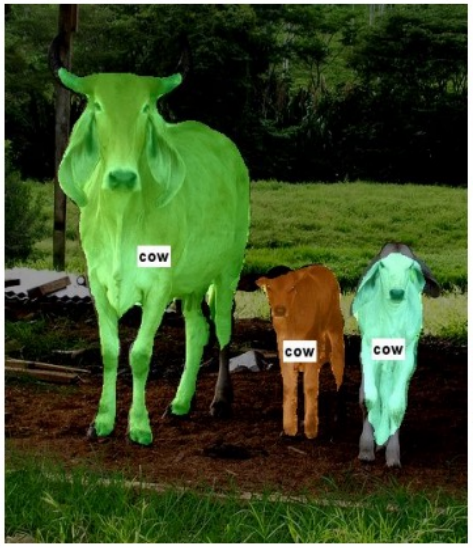

TV/Monitor
bottle


horse
person


person
train
person

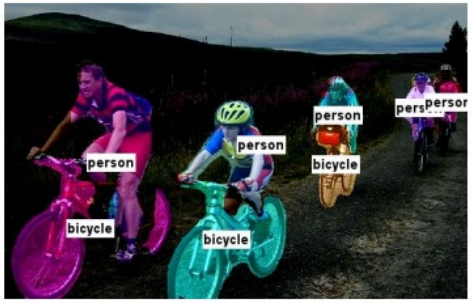

car
bus
person
bicycle


bus
bus
person
person


cow
cow
cow

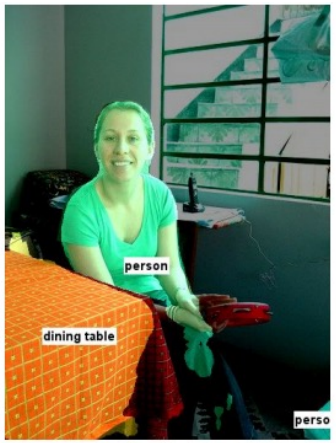

bus
bus
car
bus


person
person
person


person
person
person
bicycle
bicycle
bicycle


potted plant
potted plant
potted plant
potted plant
motorbike


cow
cow


person
dining table


chair
dining table
chair
chair
chair


bird


bus
person
train


motorbike
person
motorbike


bus
car
person
car
car


person
person


person
sheep
sheep
sheep

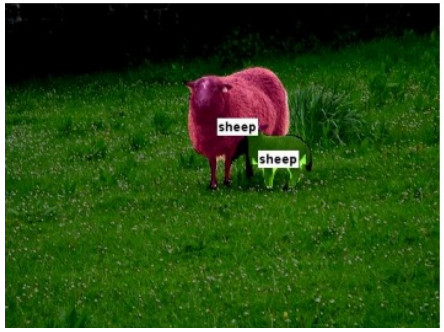

person
sofa
dog

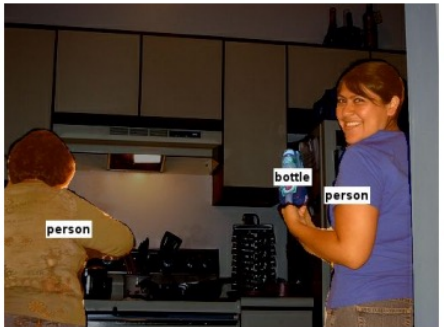

person
motorbike
motorbike
motorbike
motorbike


dog
dog


person
person
car


IDEE KAFFEE
person
horse


sheep
sheep


bottle
person
person


ed plant
potted p
car
person
motorbike


person
motorbike