CSEP 576: Dense Prediction

Jonathan Huang (jonathanhuang@google.com)

University of Washington 26 May 2020

Google Research

credit: https://upload.wikimedia.org/wikipedia/commons/0/02/Intelligent_scissors_segmentation.png

Lecture Outline

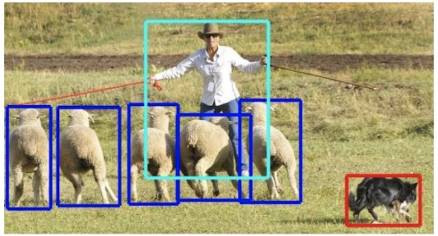
We will mainly focus on semantic segmentation as a way to introduce some of technical details behind "dense prediction"

Dense Prediction (pixel level prediction)

- Semantic Segmentation
- Instance Segmentation
- Panoptic Segmentation
- Keypoint Estimation

Problem statement

classify



classify and regress
bounding box per object

**(bounding box)
detection**

classify per pixel

**semantic
segmentation**

Segmentation Applications

Original

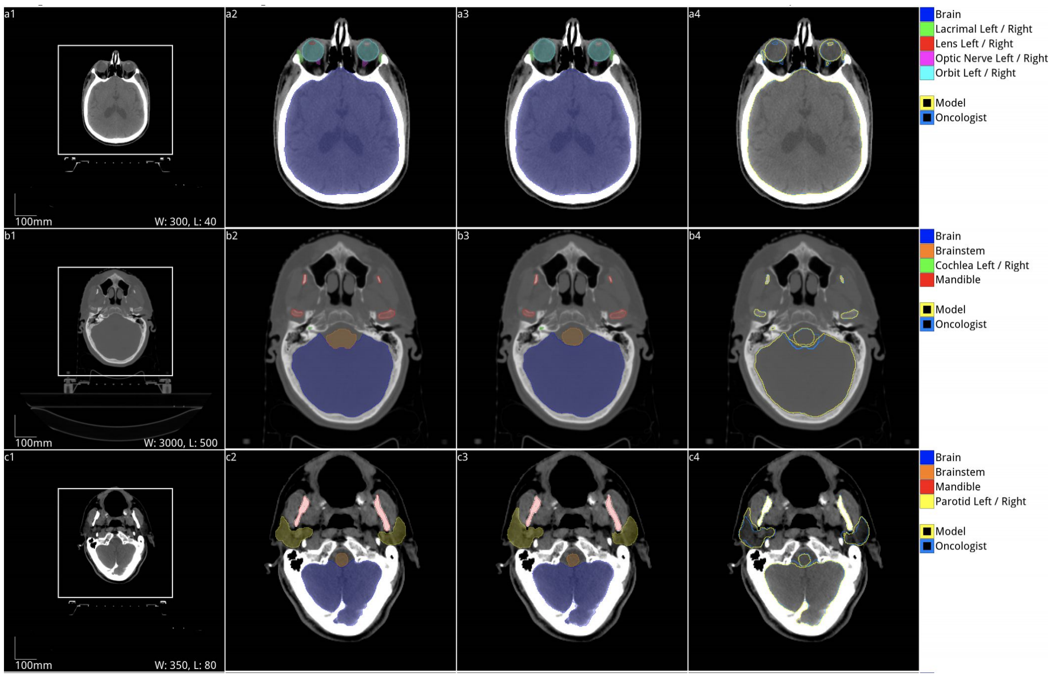
Segmentation map

Final

Segmentation Applications

[Large Scale High-Resolution Land Cover Mapping with Multi-Resolution Data] by Robinson et al

Medical Segmentation



[Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy]() by Nikolov et al

Outline of Semantic Segmentation

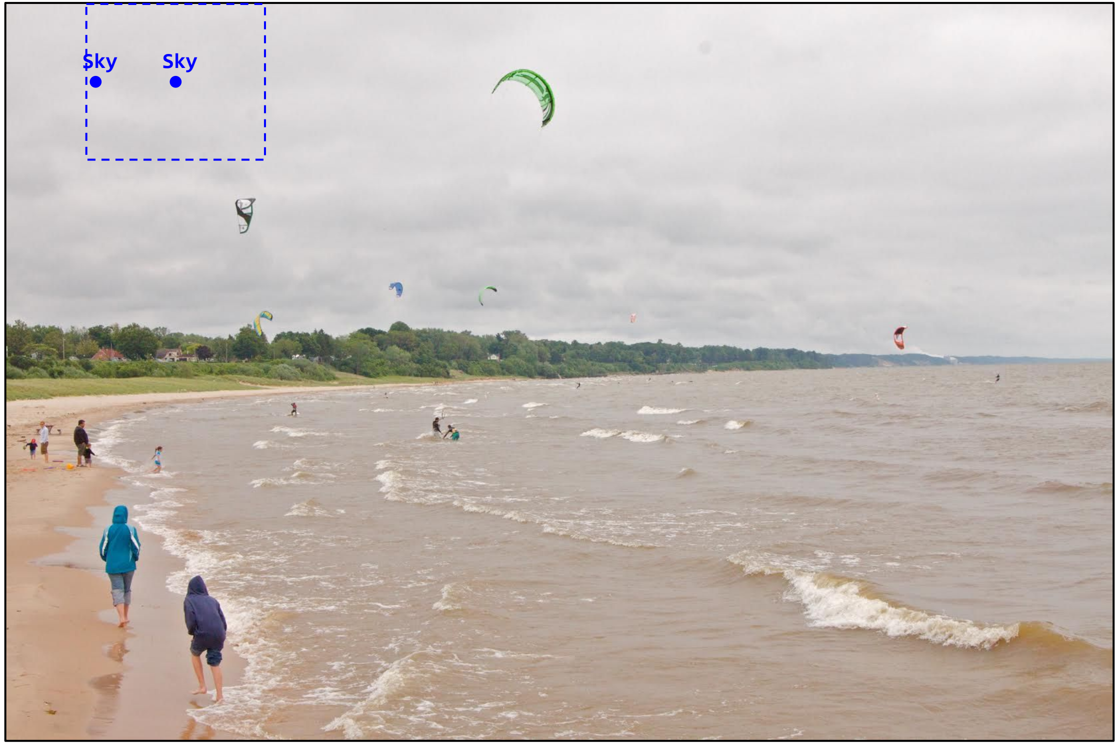
- The sliding window connection (again)
- Fully Convolutional models
- How to get high resolution outputs with
  - Atrous convolutions
  - “Upconvolutions”
- Target Assignment
- Evaluation of Semantic Segmentation

Relevant for all dense prediction tasks

“Sliding Window” Segmentation

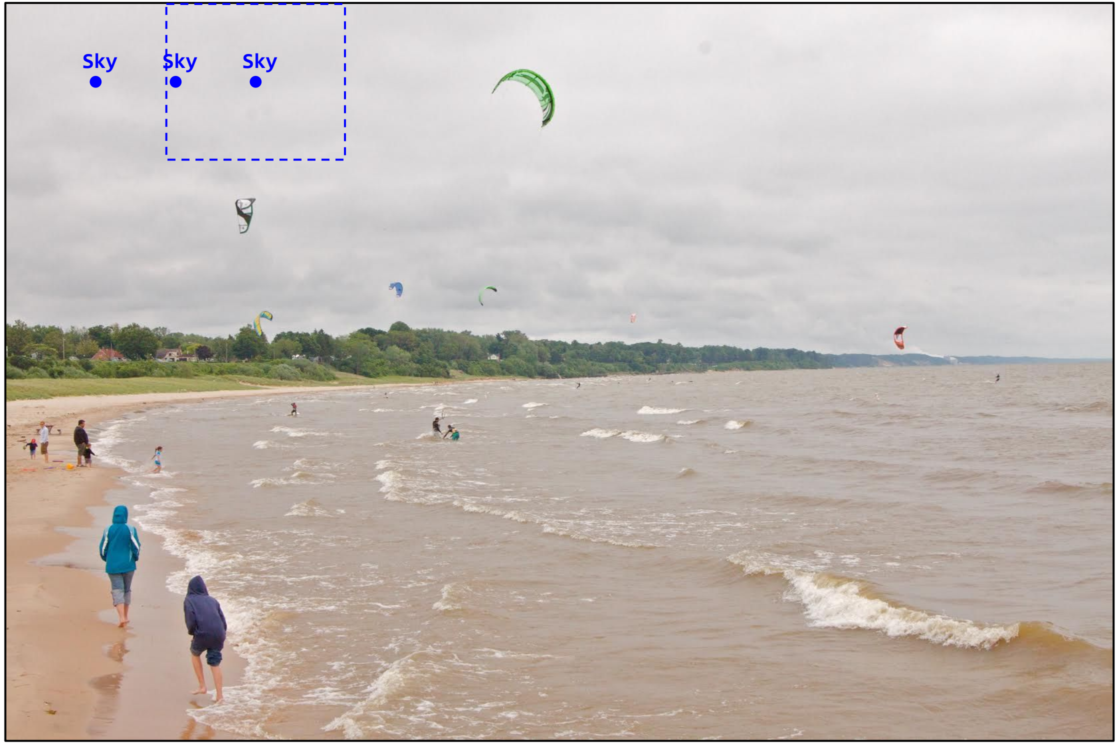
Same idea as detection:
Extract features from a window around a point;
Predict class label for point

"Sliding Window" Segmentation



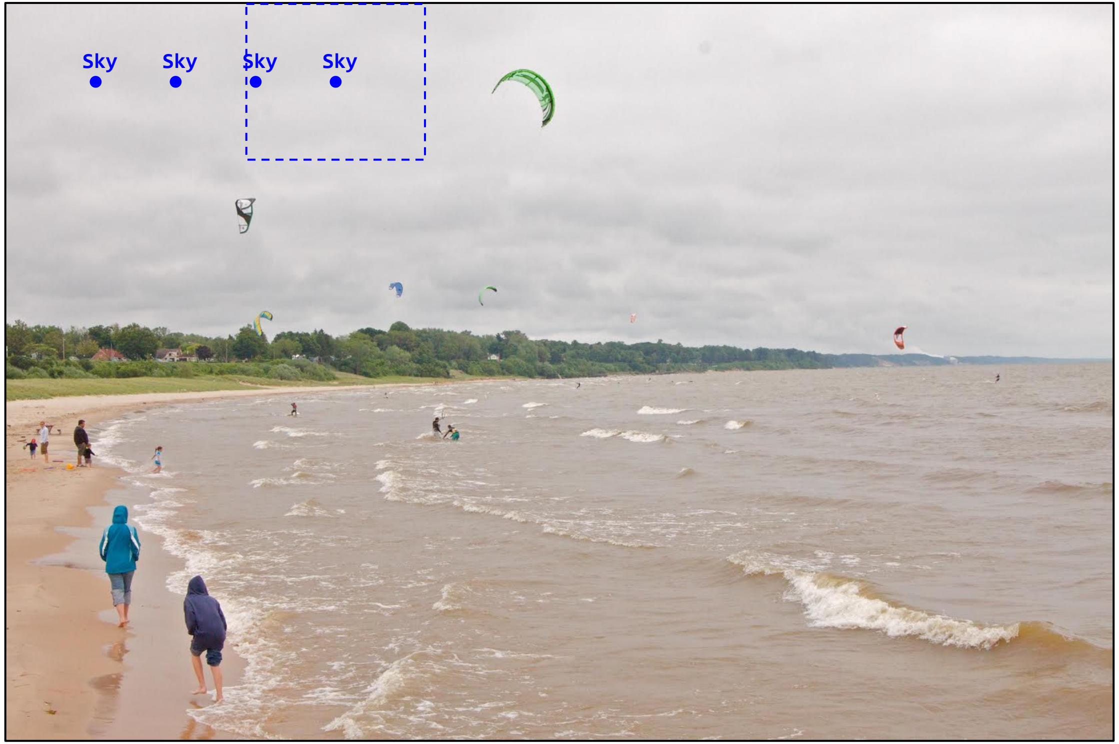
Same idea as detection: Extract features from a window around a point; Predict class label for point

"Sliding Window" Segmentation



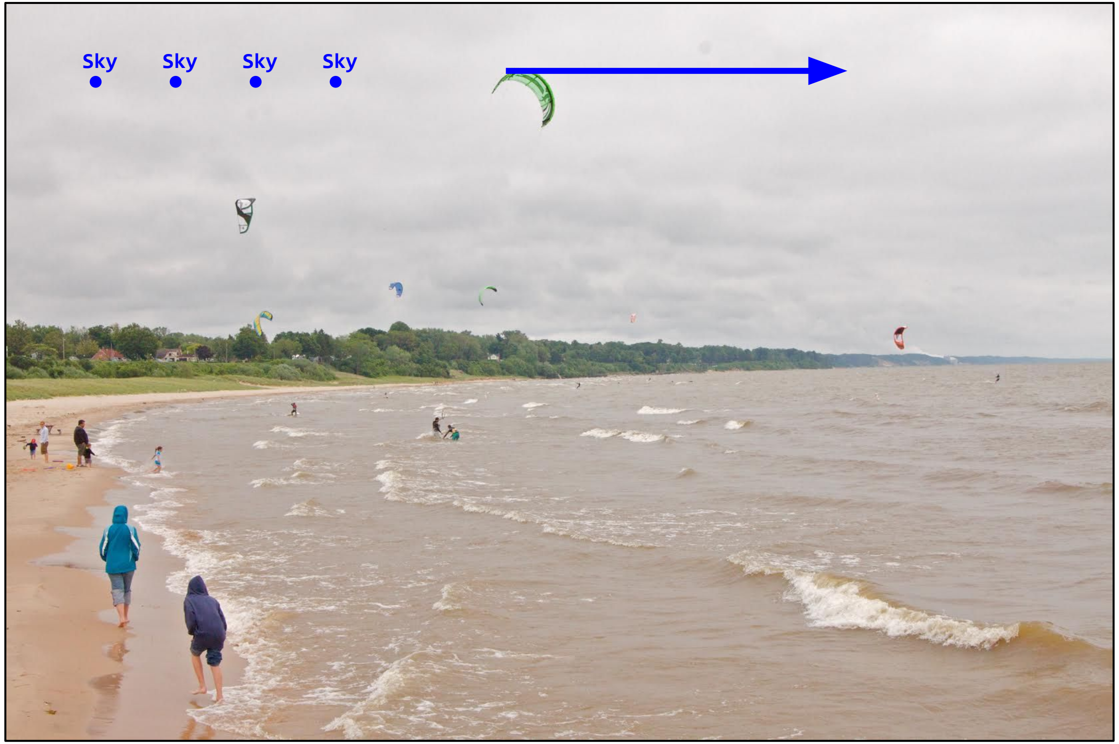
Same idea as detection:
Extract features from a window around a point;
Predict class label for point

“Sliding Window” Segmentation



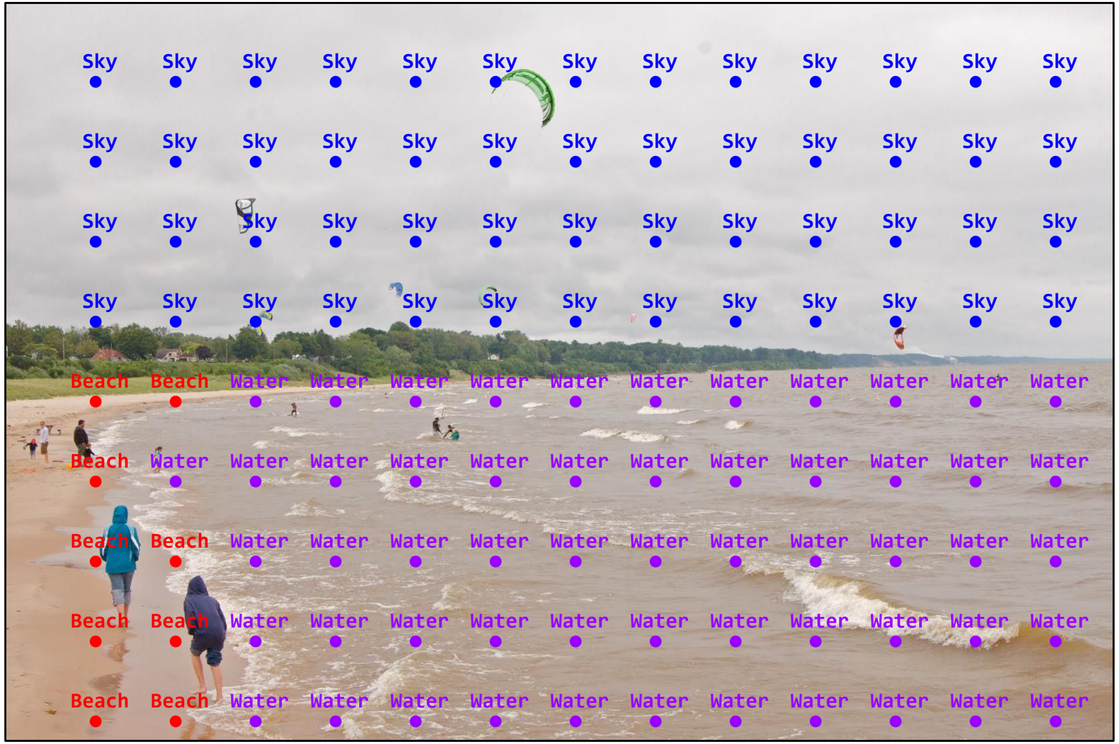
Same idea as detection:
Extract features from a window around a point;
Predict class label for point

“Sliding Window” Segmentation



Same idea as detection:
Extract features from a window around a point;
Predict class label for point

"Sliding Window" Segmentation



Same idea as detection:
Extract features from a window around a point;
Predict class label for point

Fully Convolutional Networks - Standard for detection / segmentation / keypoint prediction

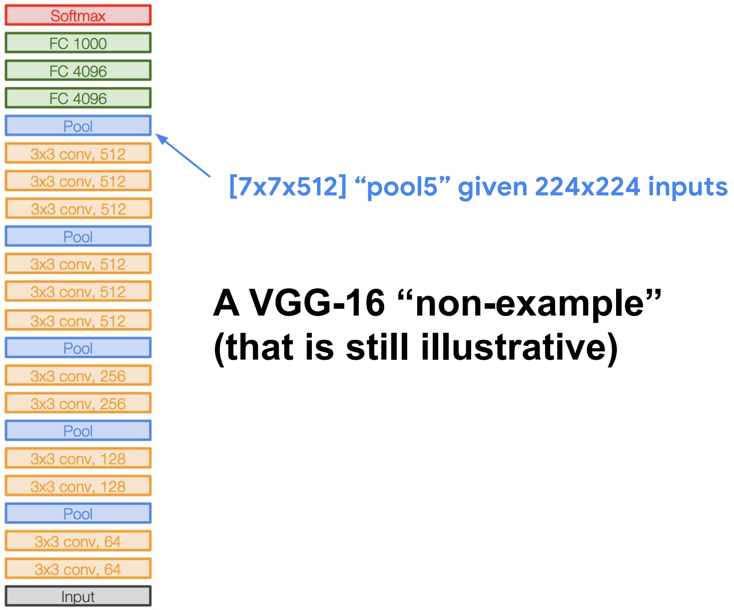
“Fully Convolutional”: All layers operate on local inputs (e.g. Conv, Pool, ReLU); E.g. no FC layers allowed.

Properties of FCNs:

- Operate on input of any size
- Output tensors scale with input size
- Can train with heterogenous resolutions
- Can train and test at different resolutions

Fully Convolutional Networks - Standard for detection / segmentation / keypoint prediction

"Fully Convolutional": All layers operate on local inputs (e.g. Conv, Pool, ReLU); E.g. no FC layers allowed.



VGG trained on 224x224 images

A VGG-16 "non-example" (that is still illustrative)

Properties of FCNs:

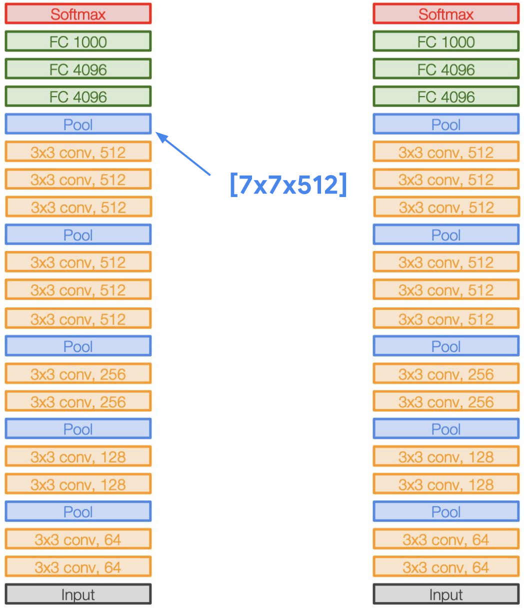
- Operate on input of any size
- Output tensors scale with input size
- Can train with heterogenous resolutions
- Can train and test at different resolutions

Fully Convolutional Networks - Standard for detection / segmentation / keypoint prediction

"Fully Convolutional": All layers operate on local inputs (e.g. Conv, Pool, ReLU); E.g. no FC layers allowed.

Properties of FCNs:

- Operate on input of any size
- Output tensors scale with input size
- Can train with heterogenous resolutions
- Can train and test at different resolutions



VGG trained on 224x224 images

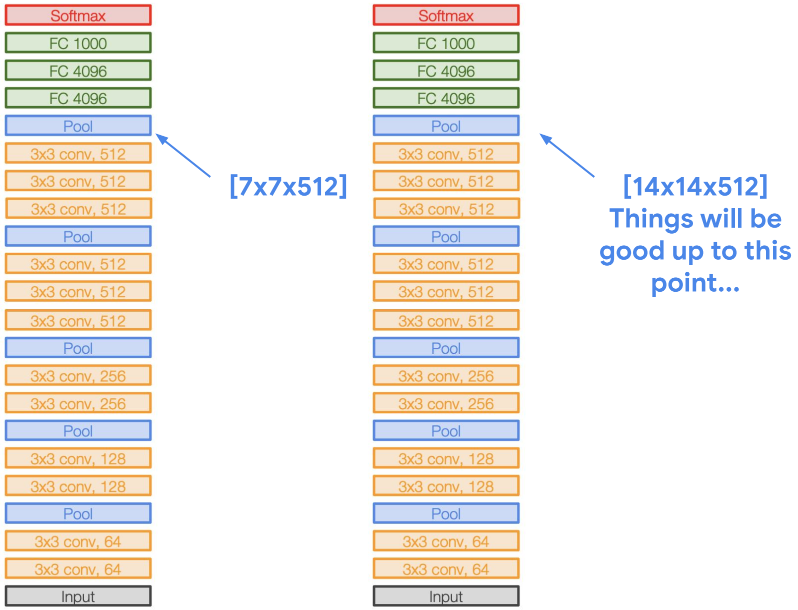
What if we try running inference 448x448 image?

Fully Convolutional Networks - Standard for detection / segmentation / keypoint prediction

“Fully Convolutional”: All layers operate on local inputs (e.g. Conv, Pool, ReLU); E.g. no FC layers allowed.

Properties of FCNs:

- Operate on input of any size
- Output tensors scale with input size
- Can train with heterogenous resolutions
- Can train and test at different resolutions



VGG trained on 224x224 images

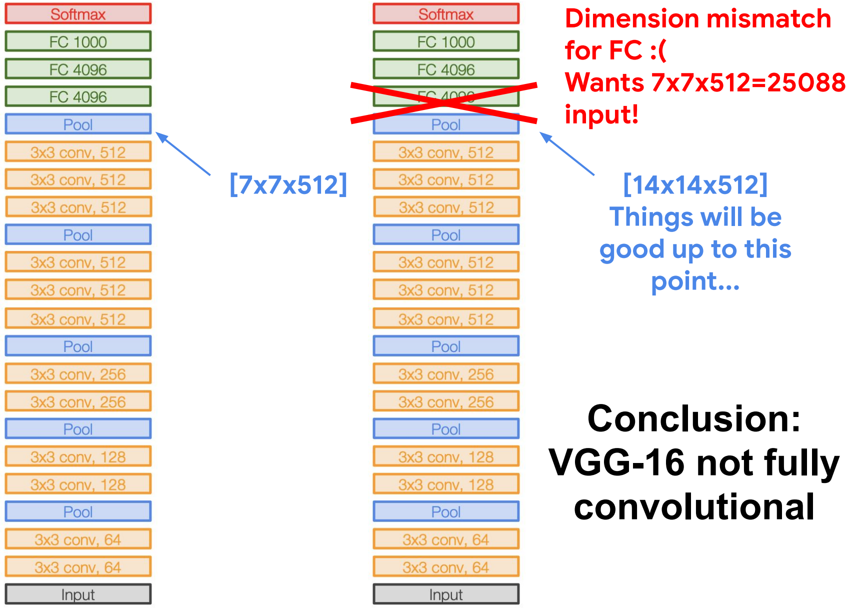
What if we try running inference 448x448 image?

Fully Convolutional Networks - Standard for detection / segmentation / keypoint prediction

"Fully Convolutional": All layers operate on local inputs (e.g. Conv, Pool, ReLU); E.g. no FC layers allowed.

Properties of FCNs:

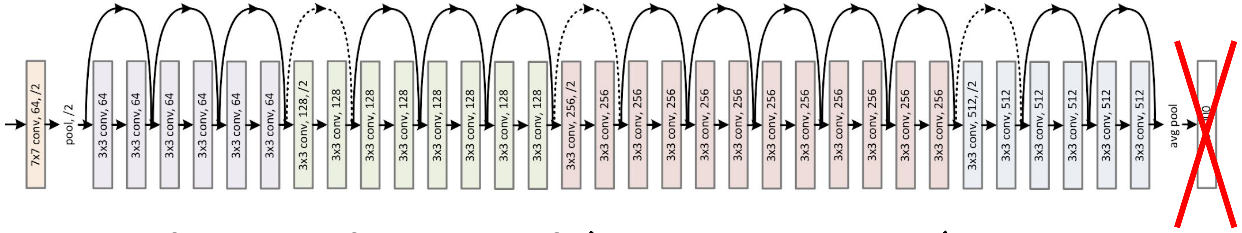
- Operate on input of any size
- Output tensors scale with input size
- Can train with heterogenous resolutions
- Can train and test at different resolutions



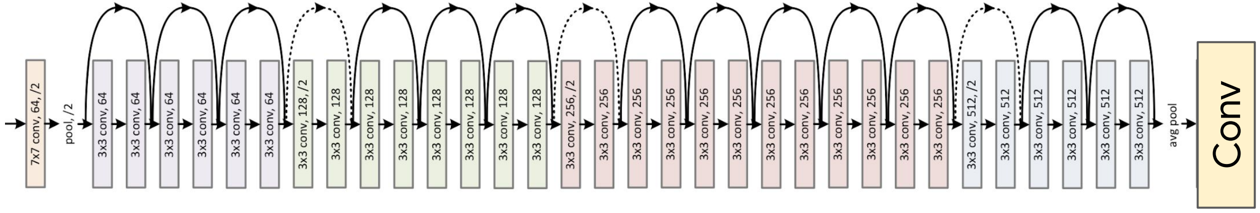
VGG trained on 224x224 images

What if we try running inference 448x448 image?

Ways to get an FCN (from an existing non-FCN)

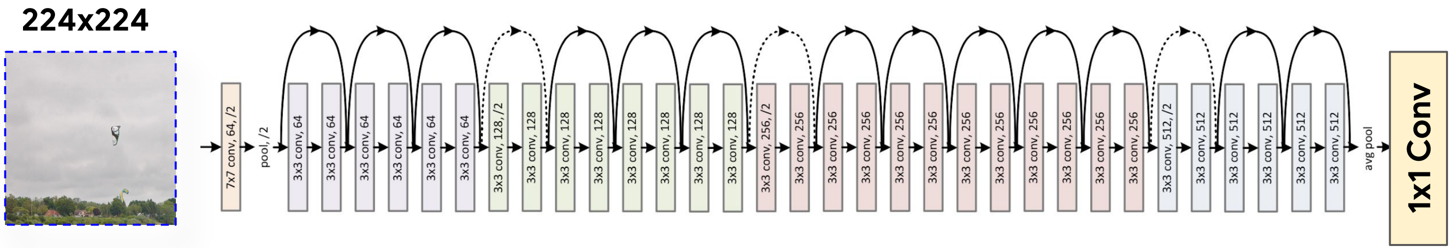


Option 1: Chop off FC (and pooling layers) at top
(and possibly add new convs)

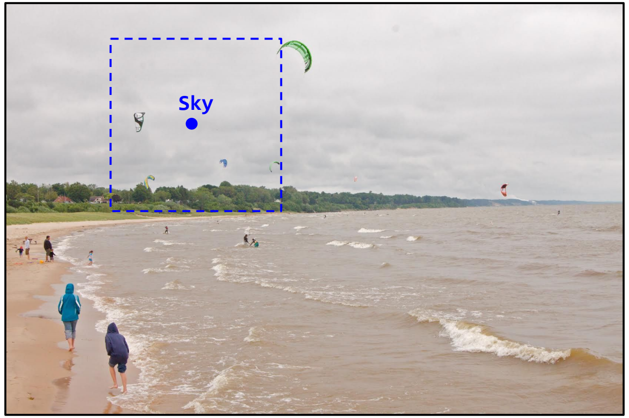


Option 2: Convert FC layers to “Equivalent” Convs

Ways to get an FCN (from an existing non-FCN)



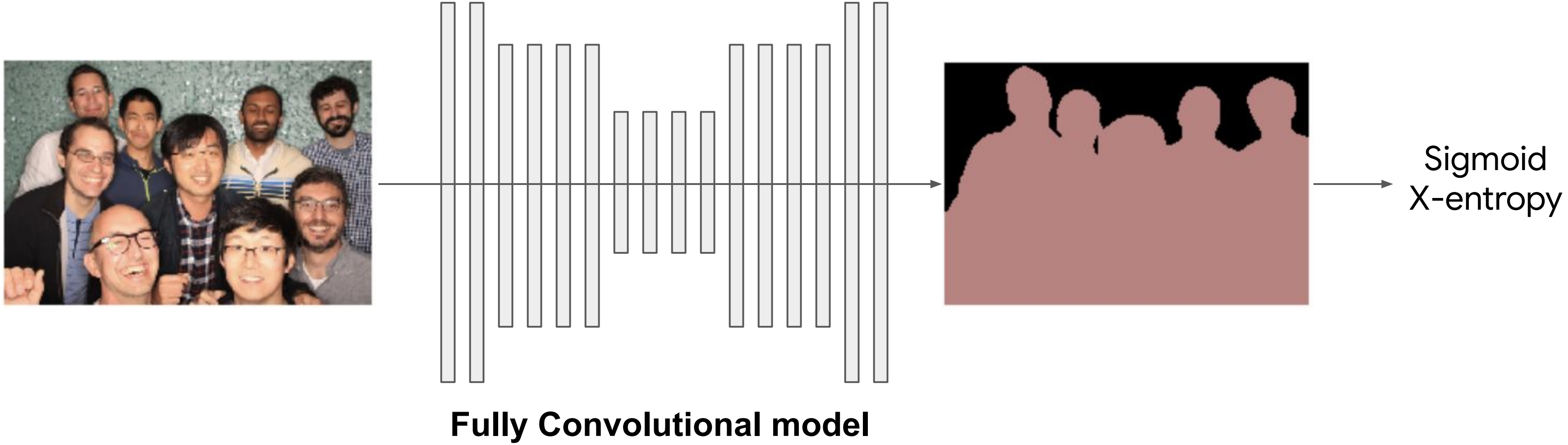
Convert top FC layer to Conv layer that takes full extent of input (in this case, FC is 1x1 with 1000 output channels)



Now can run network on much larger image (even after training!)

Note: w/o the avg pool, we'd convert the FC to a 7x7 conv with 1000 output channels

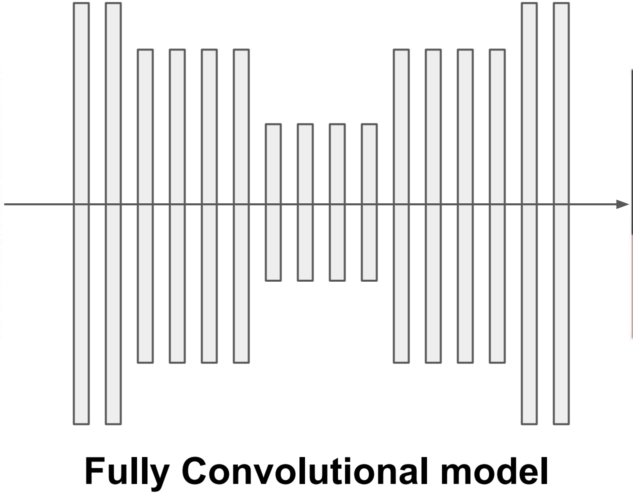
Typical Semantic Segmentation model

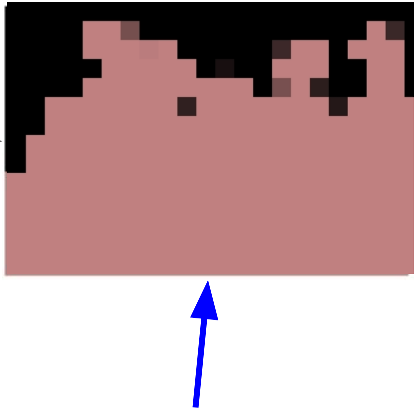


- Run image through FCN
- Train with per-pixel sigmoid X-entropy

Figure from Chen et al

Typical Semantic Segmentation model



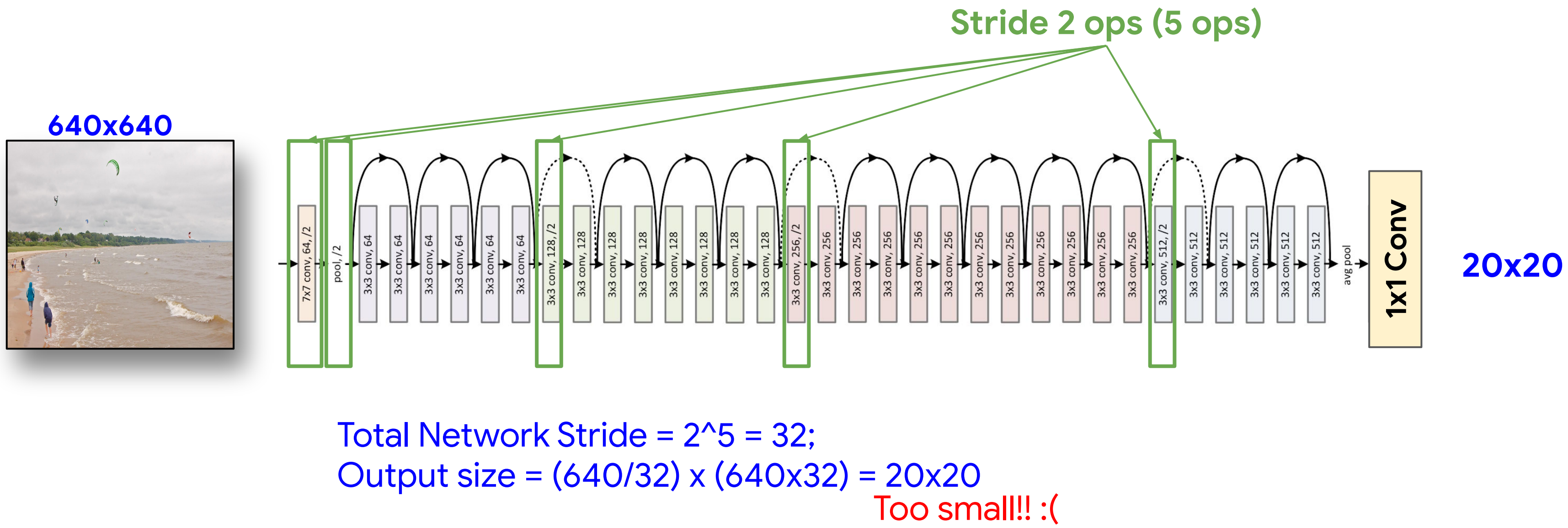


But: if we directly convert typical classification model (e.g. VGG) to FCN, we'll get something like this :(

- Run image through FCN
- Train with per-pixel sigmoid X-entropy

Figure from Chen et al

Typical CNN Output sizes are too small

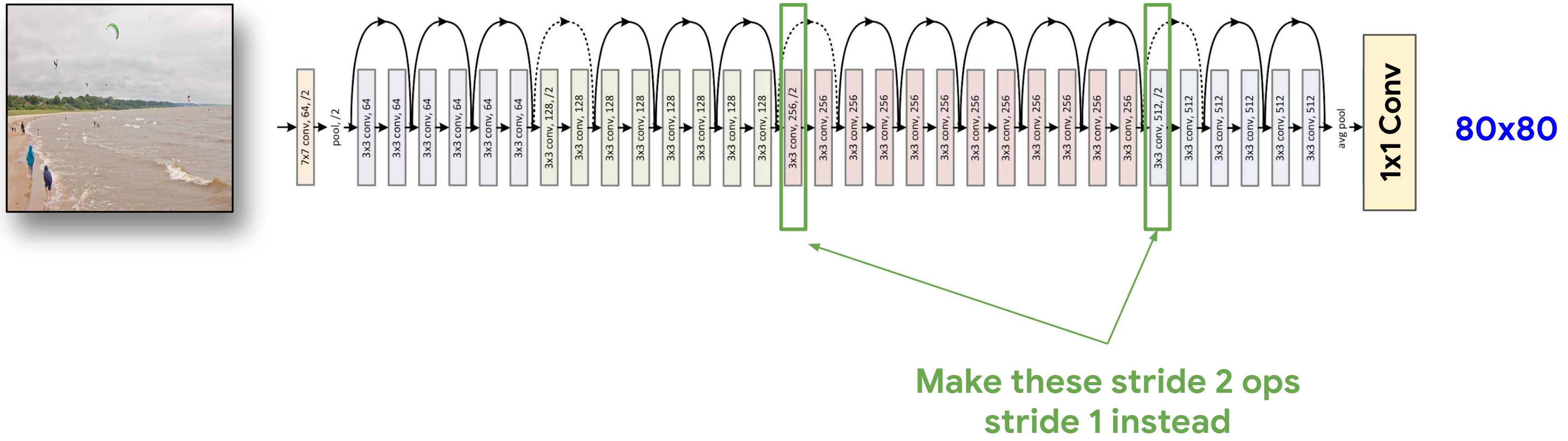


- Network stride = product of layer strides (for single path network)
  - For typical ImageNet networks (e.g. AlexNet, VGG, Resnet) stride prior to FC layers is 32
- For segmentation we typically want smaller network stride (e.g. 2, 4 or 8)

How to get high resolution outputs (e.g. w/stride < 32)

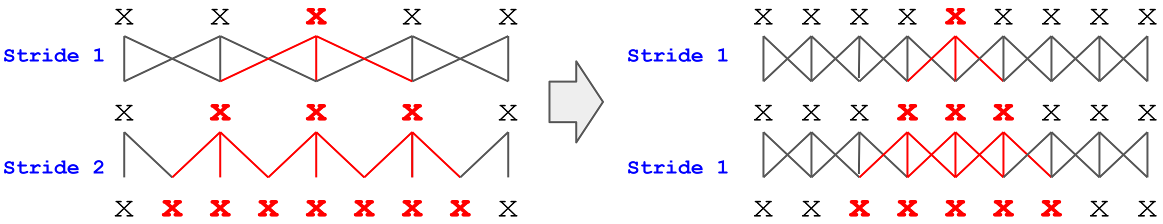
- **Use fewer stride 2 convolutions**
- Use “upconvolution” operators

Approach 1: Just don't downsample that many times



Resulting network stride: 8

Replace stride 2 convolutions with stride 1



Problem: Doing this directly can significantly reduce receptive field size...

Chen, Papandreou, et al, 2015

Some Receptive Field arithmetic

How big is our receptive field?

Receptive field size

Kernel size at layer l

Product of strides up to layer l

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

Sum over network layers

Resnet-{34,50}

# layers	stride @ layer
1	1
1	2
3	4
4	8
6	16
3	32

RO = 1 + (3-1) * (1 * 1
+ 1 * 2
+ 3 * 4
+ 4 * 8
+ 6 * 16
+ 3 * 32)

= **479**

https://distill.pub/2019/computing-receptive-fields/

Resnet-{34,50}

after converting last 2 stride 2 layers to stride 1

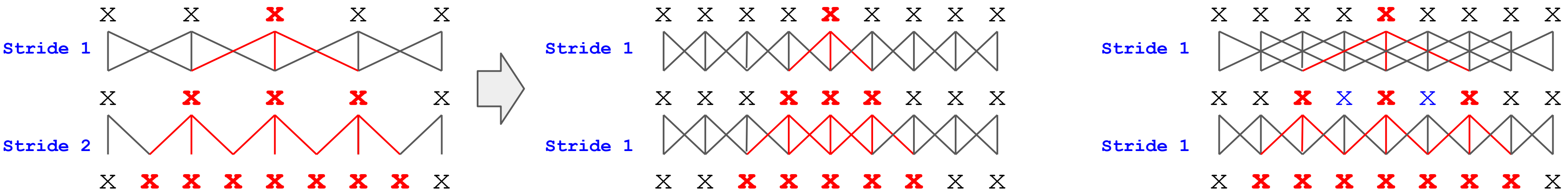
# layers	stride @ layer
1	1
1	2
3	4
4	8
6	8
3	8

RO = 1 + (3-1) * (1 * 1
+ 1 * 2
+ 3 * 4
+ 4 * 8
+ 6 * 8
+ 3 * 8)

= **239**

Receptive field area reduced 4x :(

Replace stride 2 convolutions with stride 1



Convolution with atrous rate=1,
(i.e., ordinary convolution)

Convolution with atrous rate=2

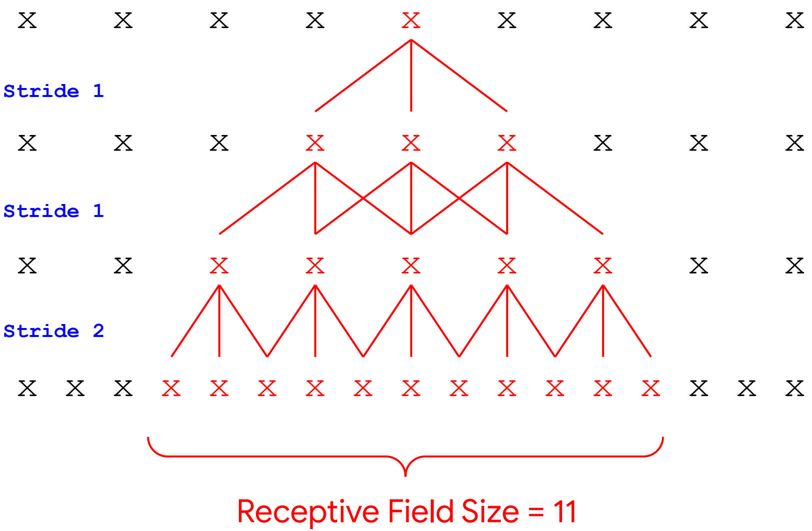
Problem: Doing this directly can reduce receptive field size...

Solution: Use dilated/*atrous* convolution (convolution with holes, *en français*) to compensate at the second layer.

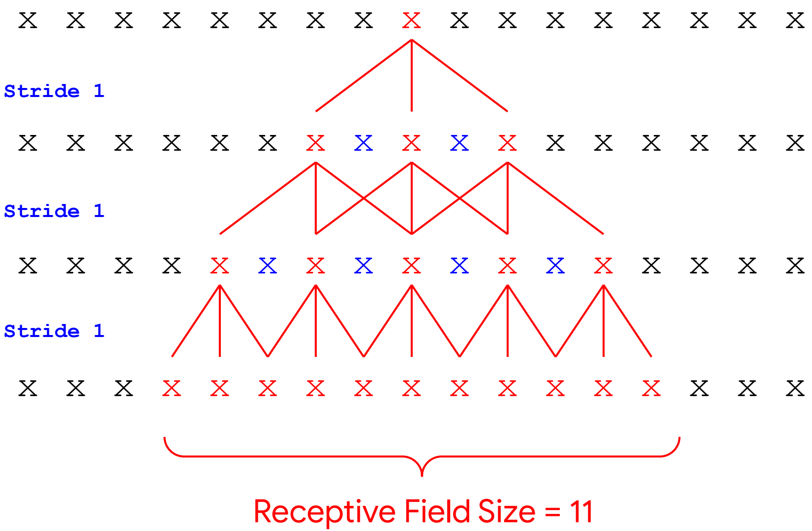
Chen, Papandreou, et al, 2015

Stringing atrous through multiple layers

Compensation needs to happen at all higher layers



Use convolution with atrous rate=2 at both layers above to maintain receptive field size



Atrous Cost/Benefit

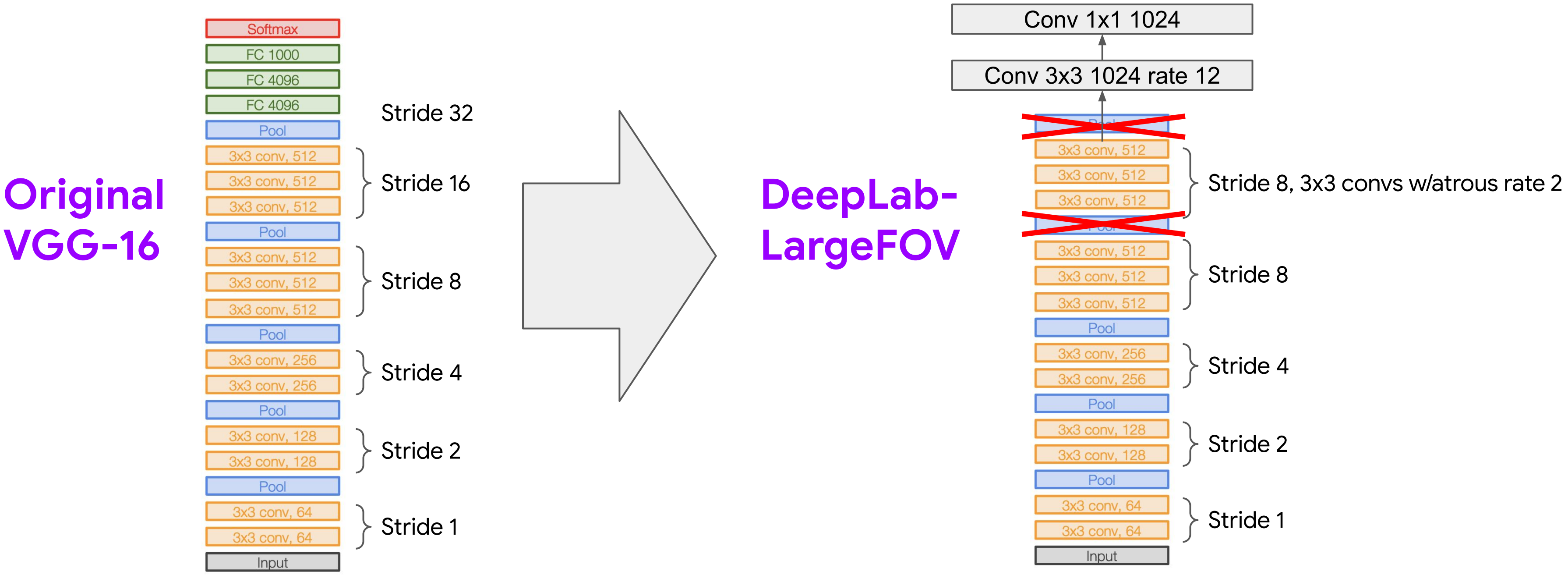
- Quadrupled memory
- Quadrupled theoretical FLOPS
- Same # parameters

Only in affected layers, and due to larger inputs (Atrous Conv itself is not more expensive than ordinary Conv)

- High resolution outputs
- Large receptive field
- *Can initialize model from ImageNet w/o retraining*

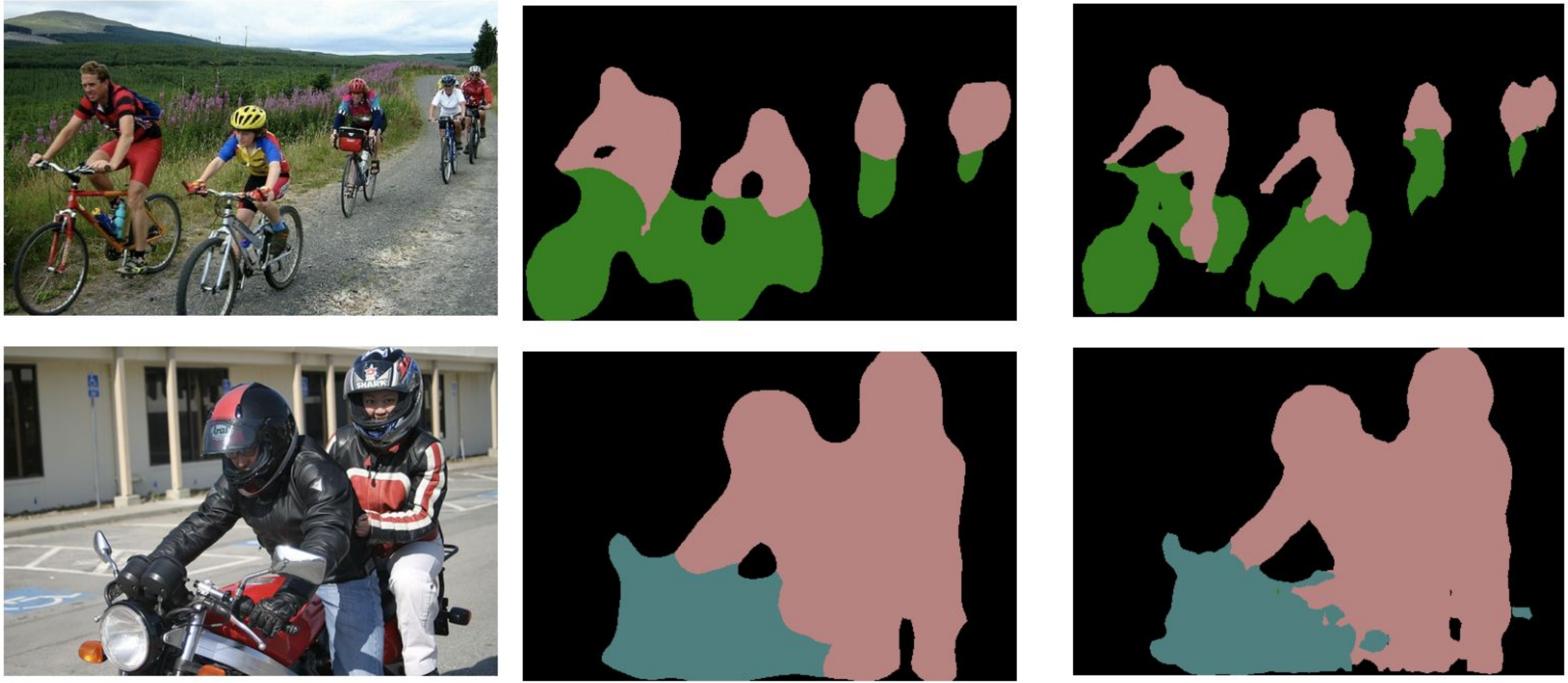
Case Study (2015): DeepLab-LargeFOV Architecture

Start with VGG; Remove last two pools; Use Atrous Convs in higher layers



[Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs]() by Chen et al
[DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs]() by Chen et al

DeepLab results (Pascal VOC dataset)



VGG based
DeepLab

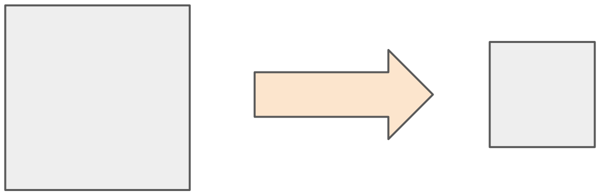
Resnet-101 based
DeepLab

How to get high resolution outputs (e.g. w/stride < 32)

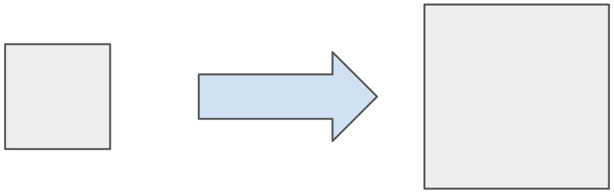
- Use fewer stride 2 convolutions
- **Use "upconvolution" operators**

“Upconvolution” operators

- Resize + Conv
- Fractional / Sub-pixel Convolution
- Transpose Convolution
- Convolution + “Periodic Reshuffling”
- Unpool (not super common)

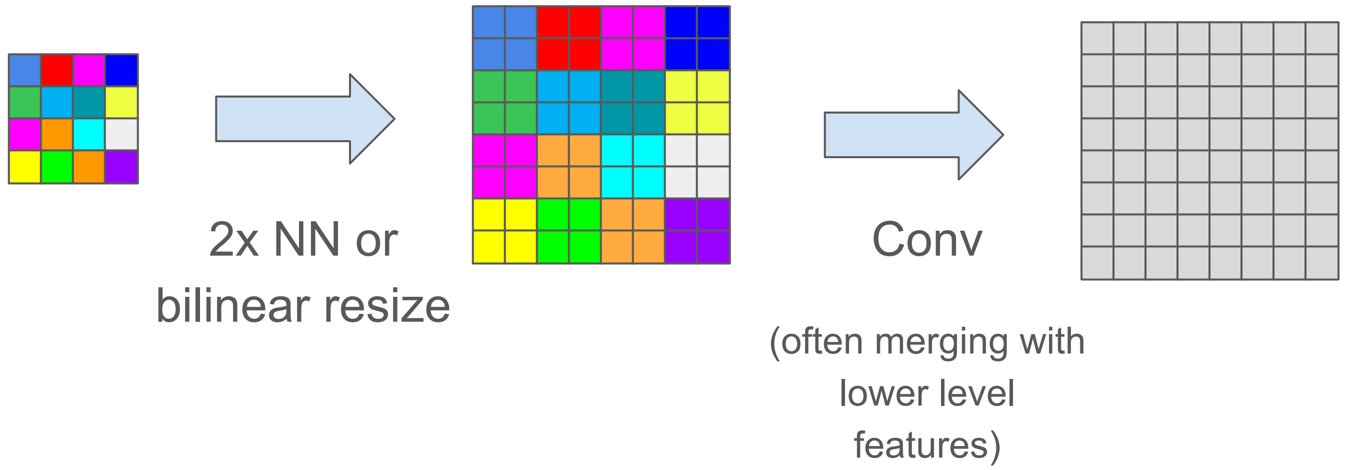


To reduce spatial resolution, use
Convolution w/stride 2

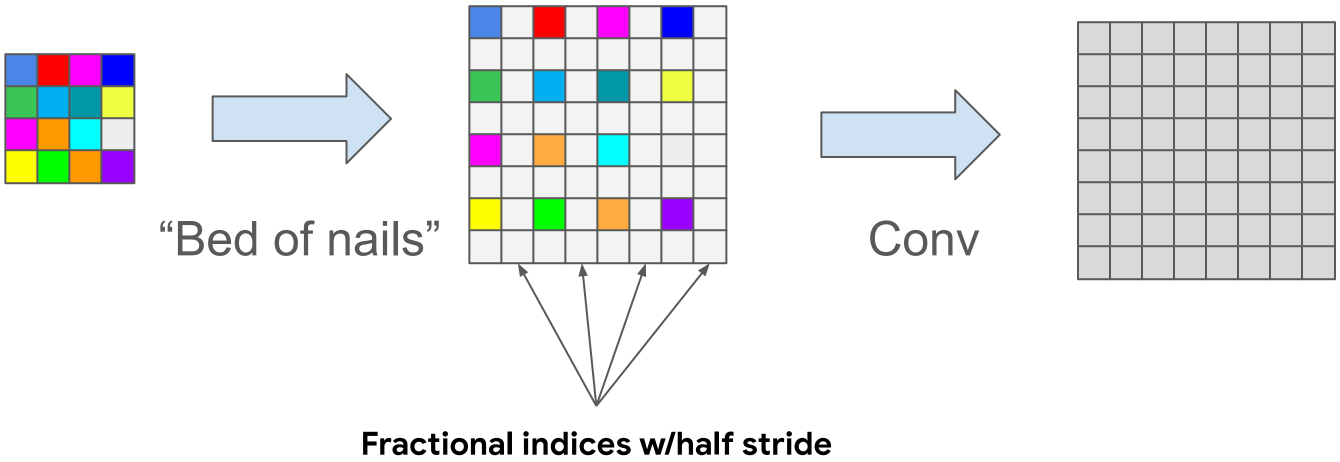


To *increase* spatial resolution, use **???**

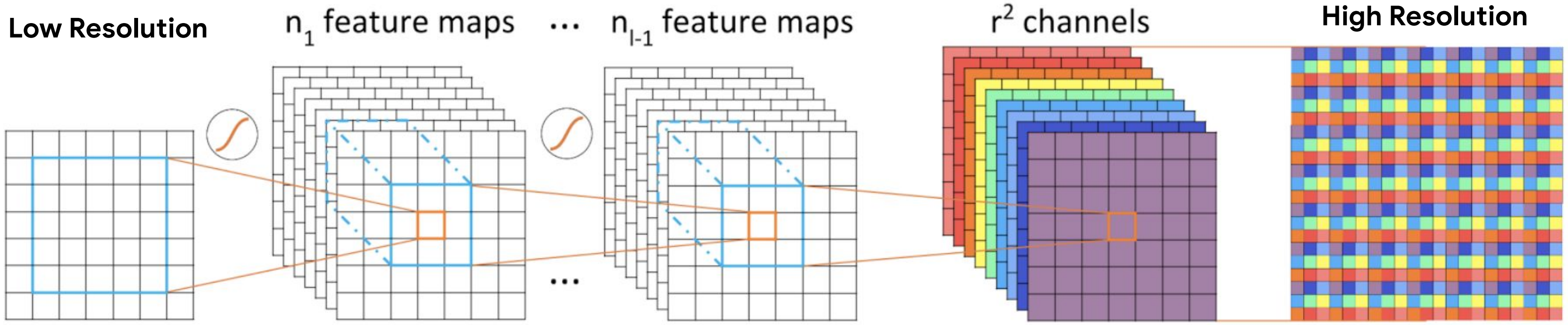
Resize + Conv


2x NN or
bilinear resize
Conv
(often merging with
lower level
features)

Fractionally Strided / Subpixel Convolution

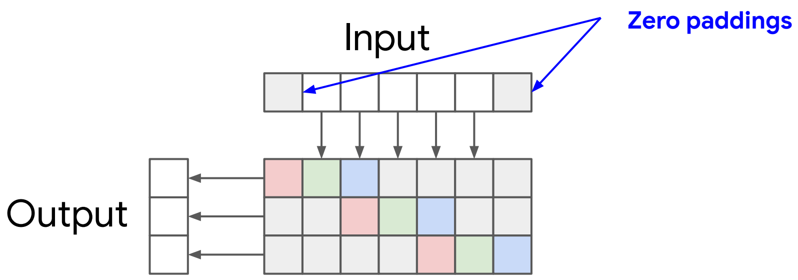


Convolution + “Periodic reshuffling”



[Is the deconvolution layer the same as a convolutional layer?](#) by Shi et al

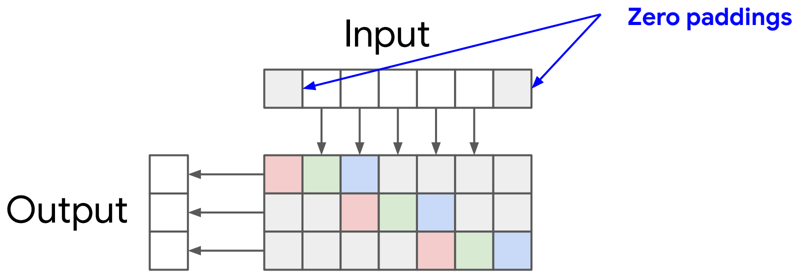
Transpose Conv



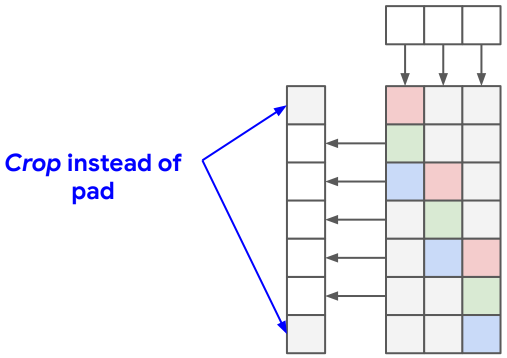
1-D Convolution (stride 2)

We can always write (ordinary) convolution as a matrix multiplication

Transpose Conv



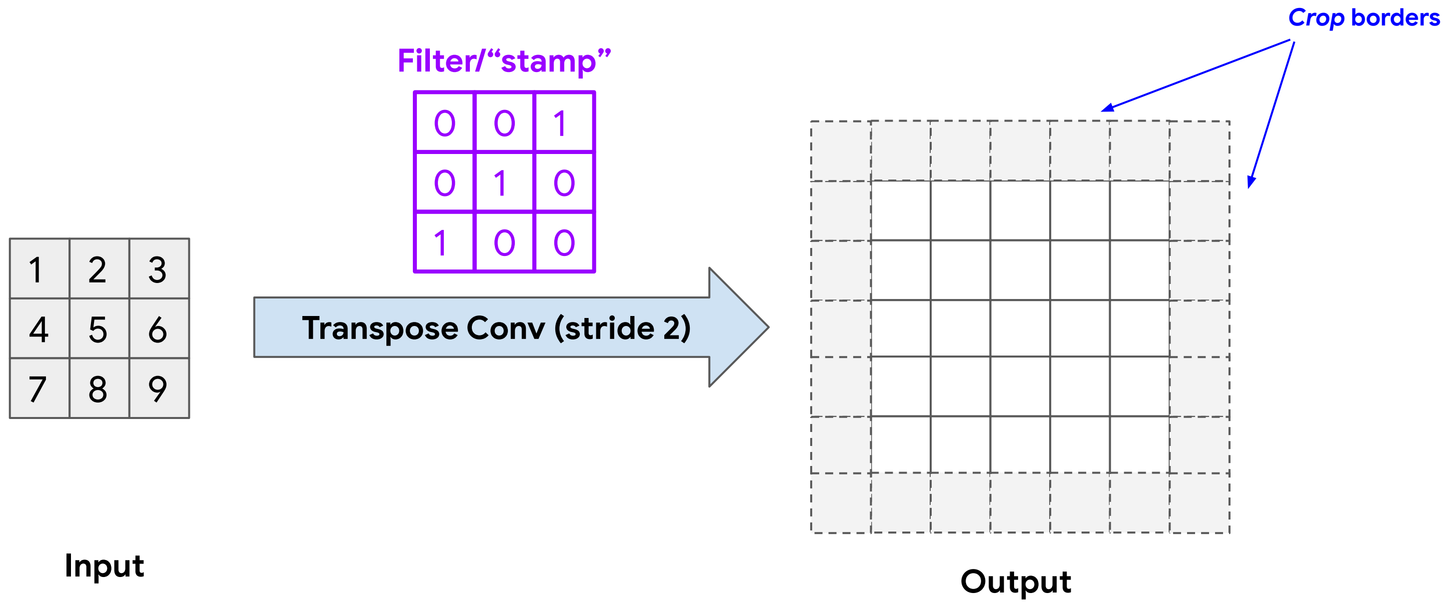
1-D Convolution (stride 2)



1-D Transpose Convolution (stride 2)

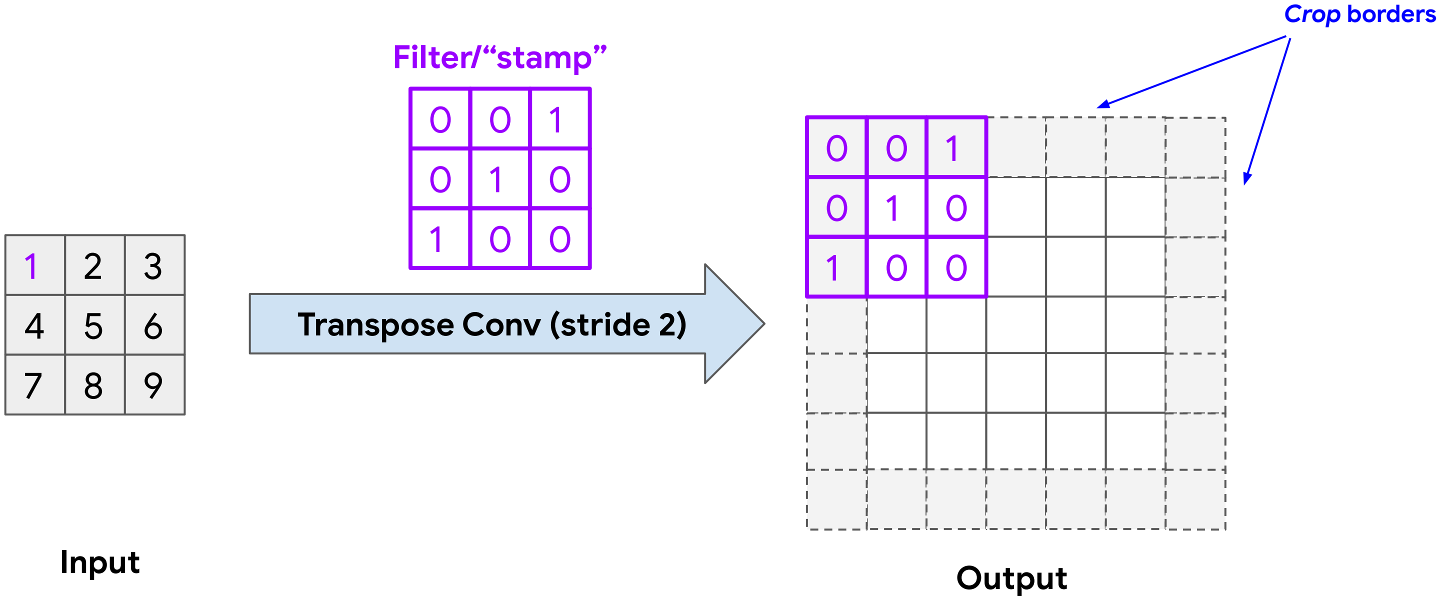
Interesting fact: Swapping forwards and backwards passes of Conv op will give Transpose Conv op

Transpose Conv (2-d example)



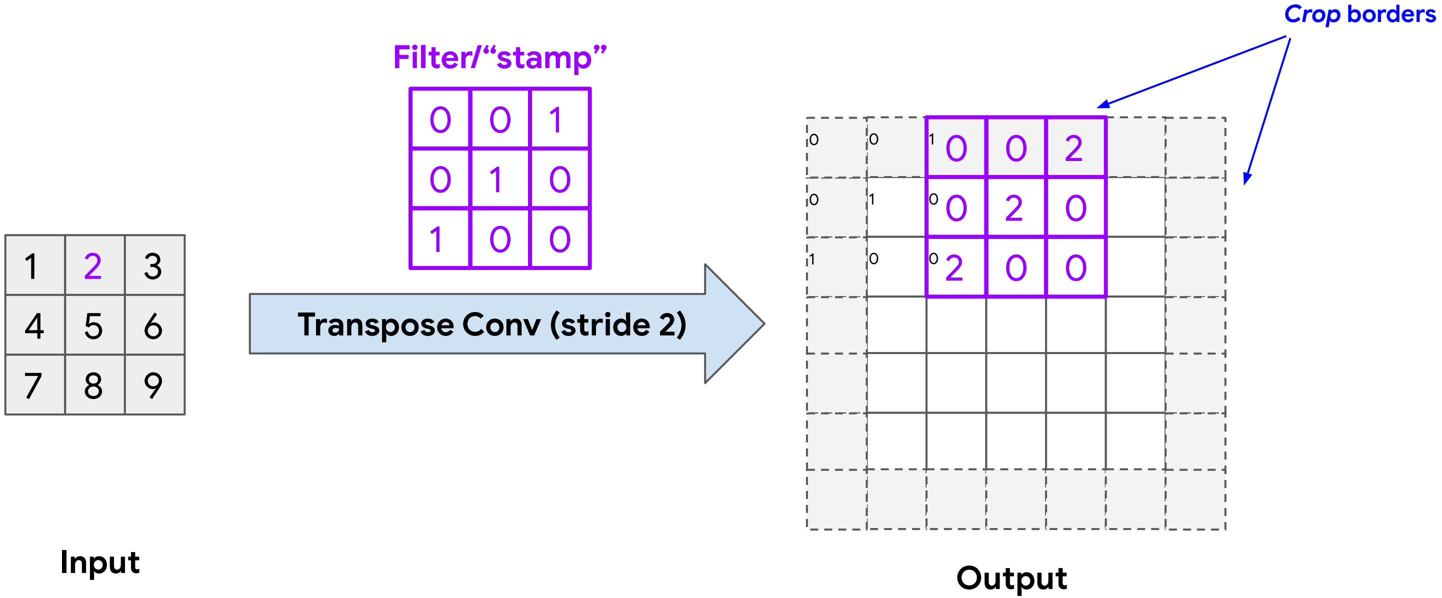
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



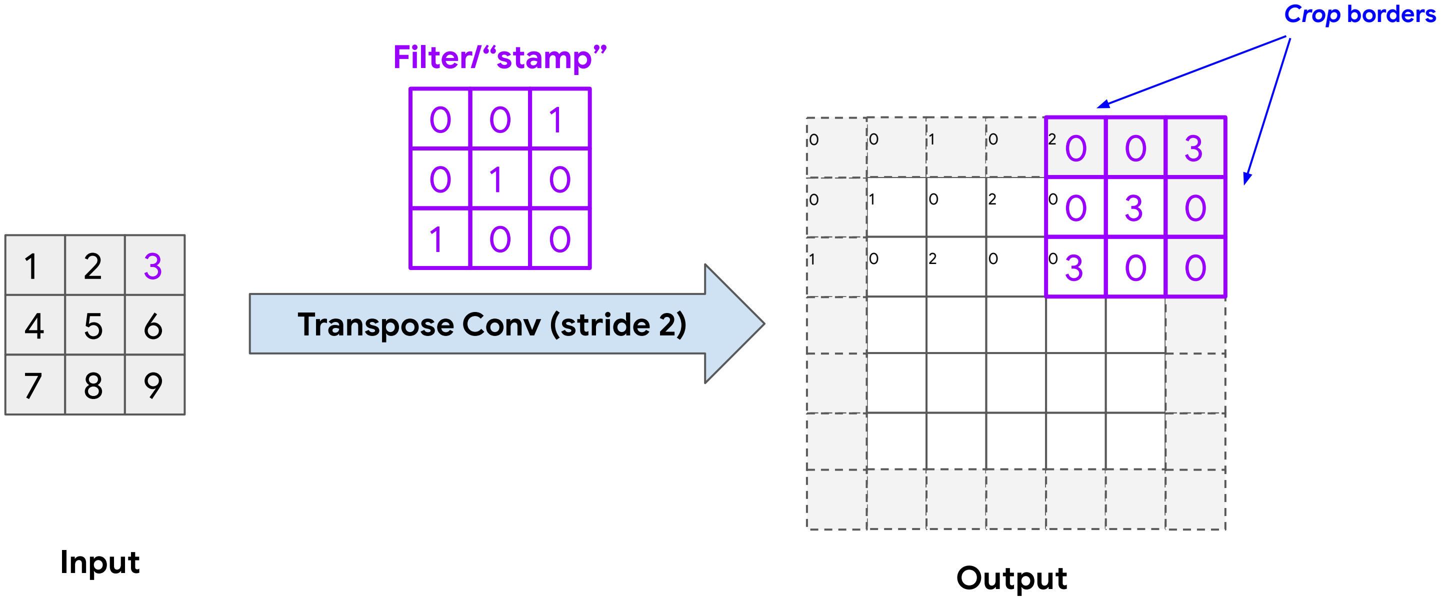
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



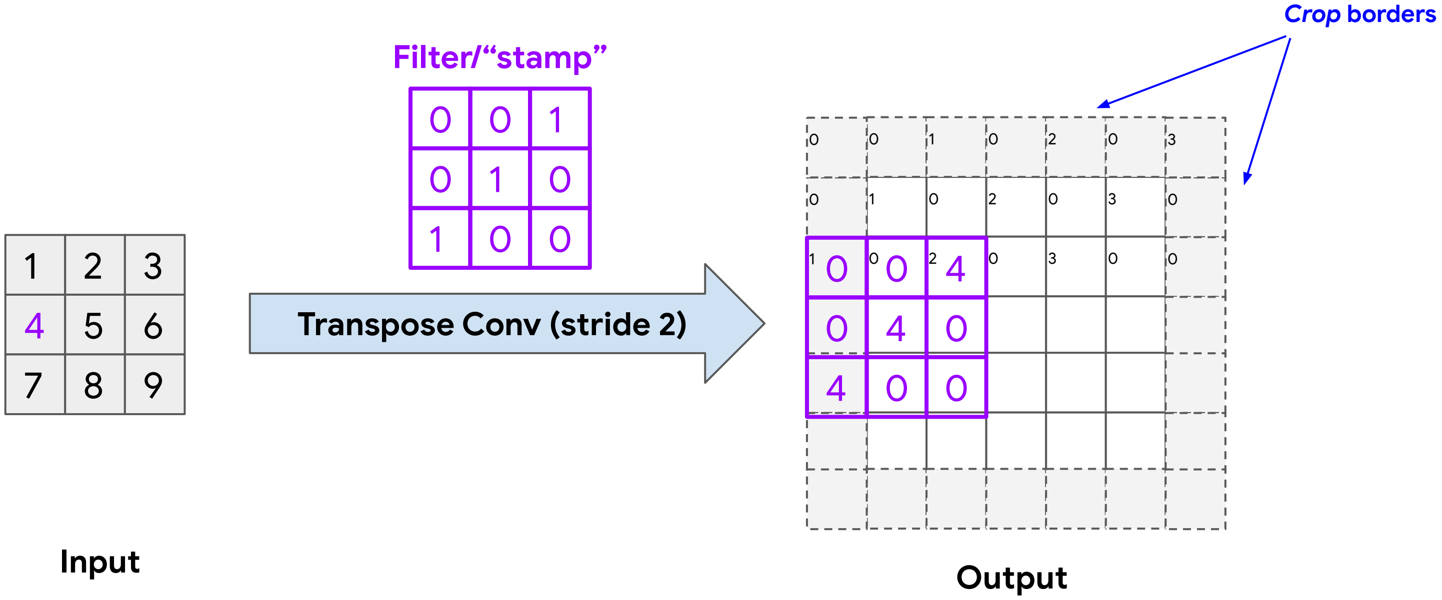
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



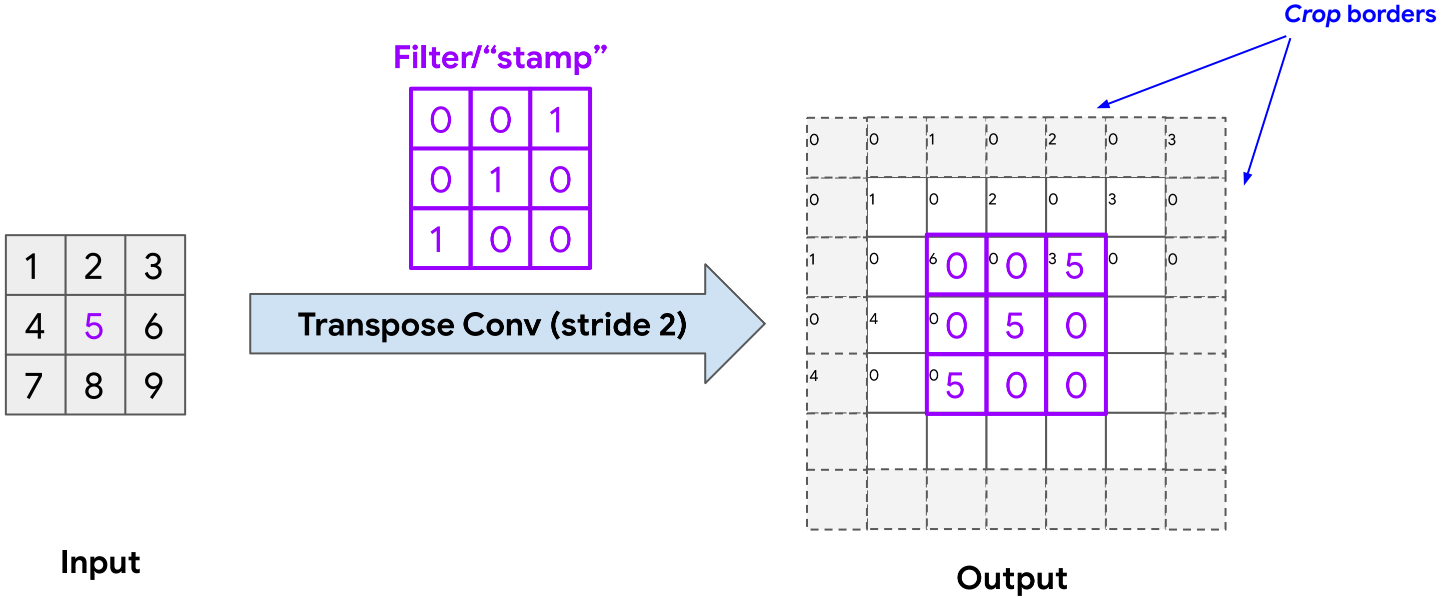
Think of "stamping" filter across the output image

Transpose Conv (2-d example)



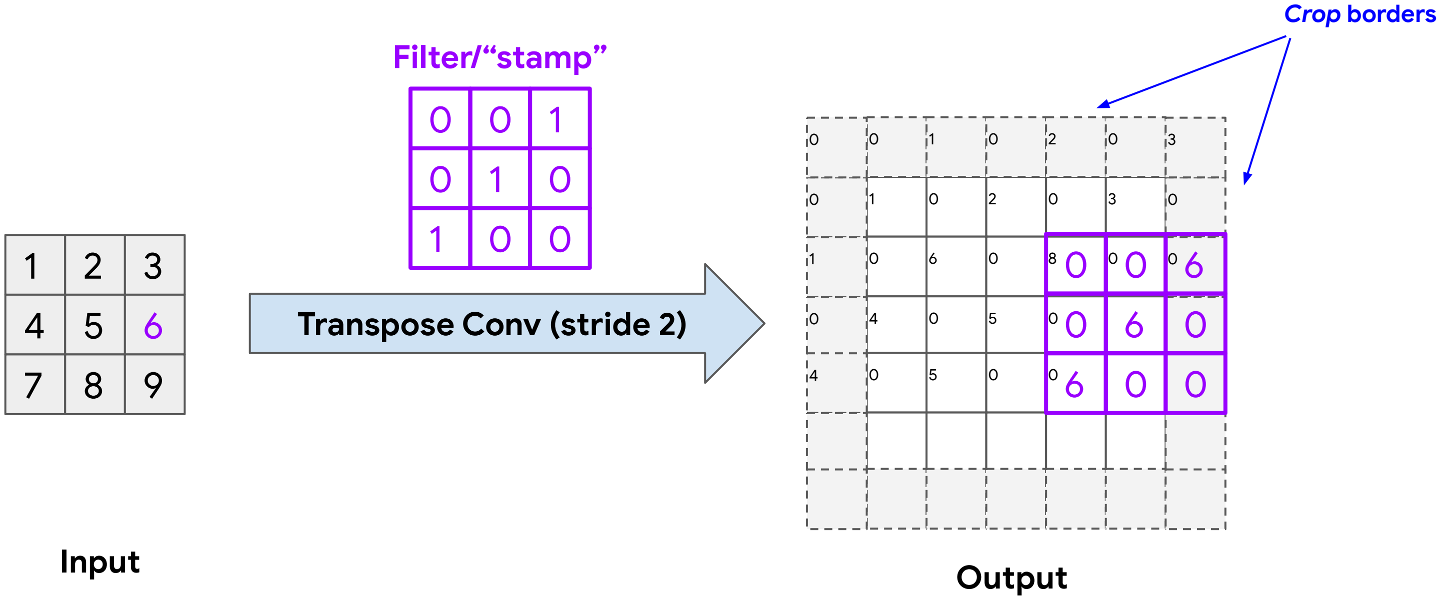
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



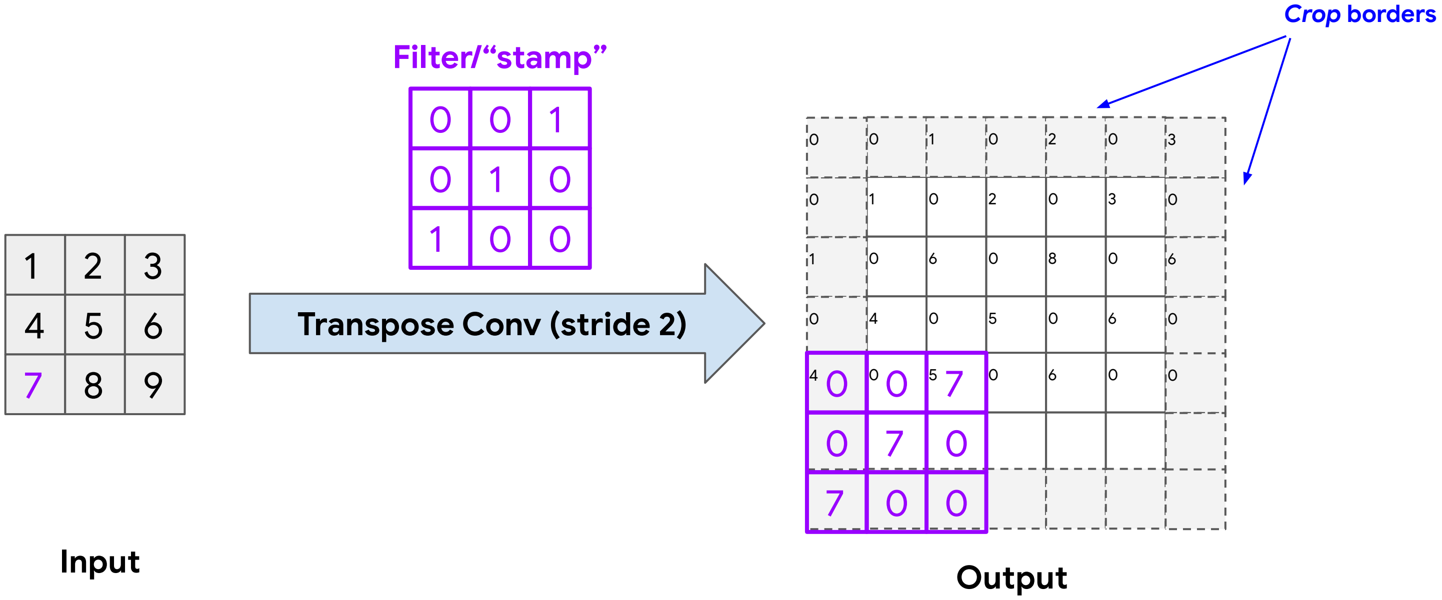
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



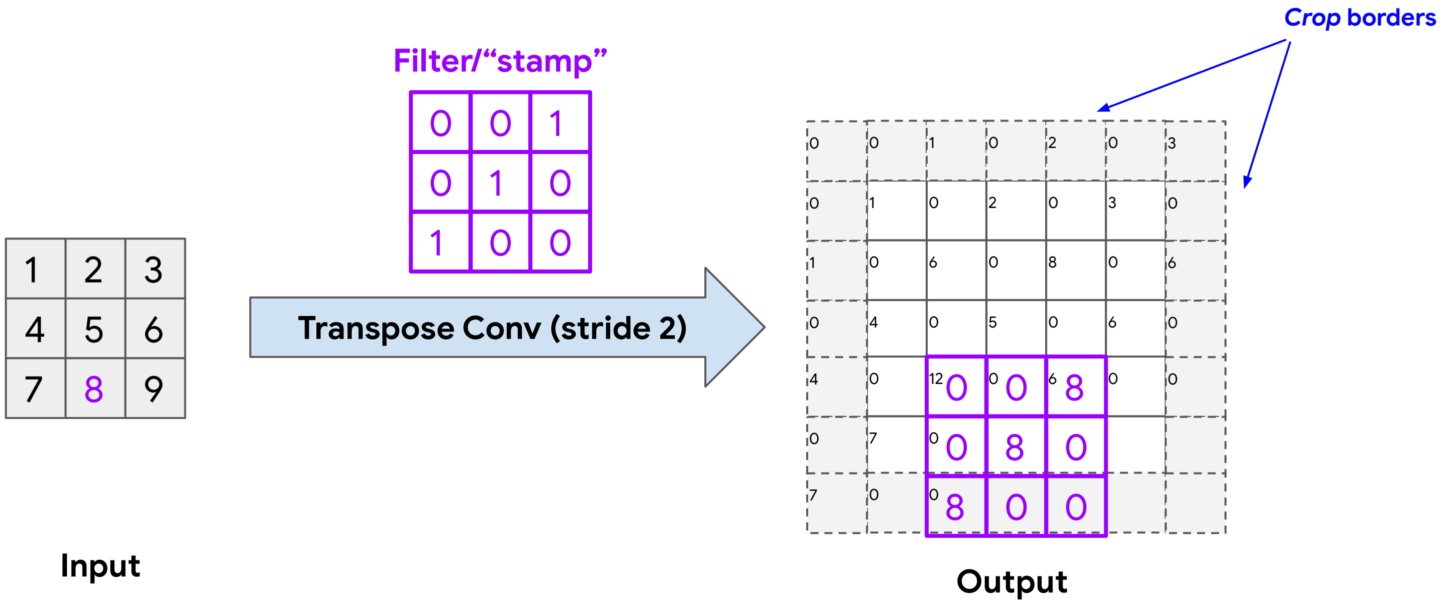
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



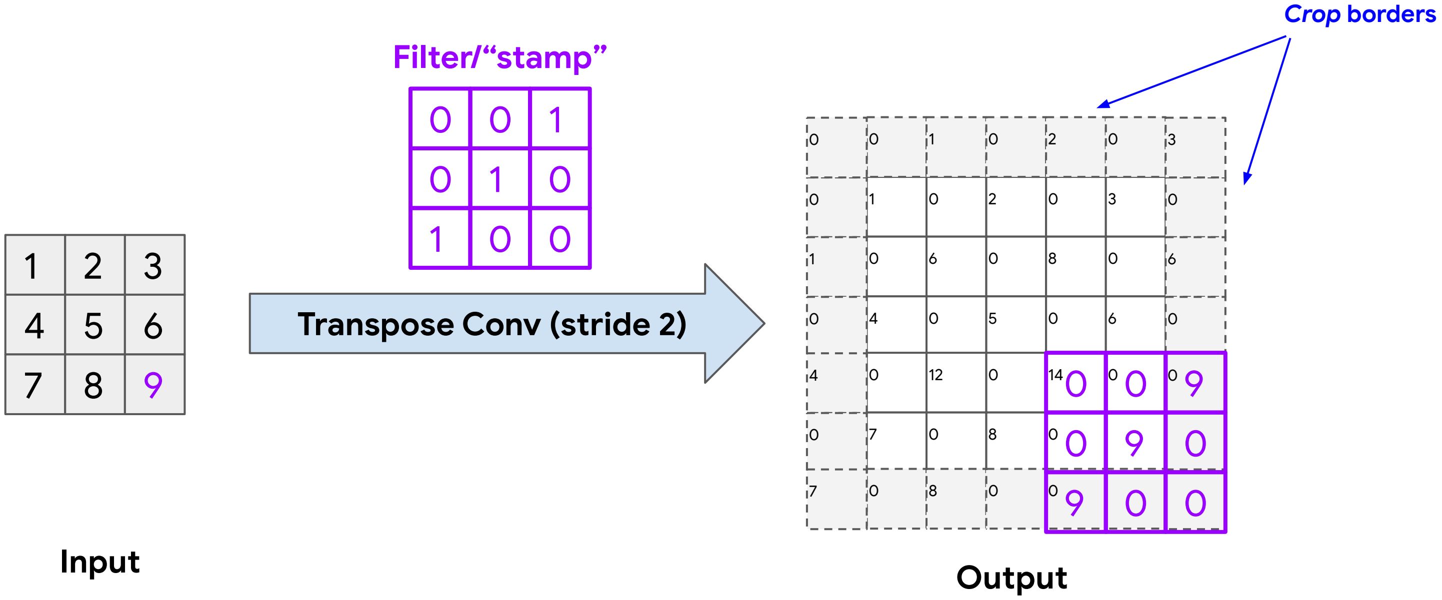
Think of “stamping” filter across the output image

Transpose Conv (2-d example)



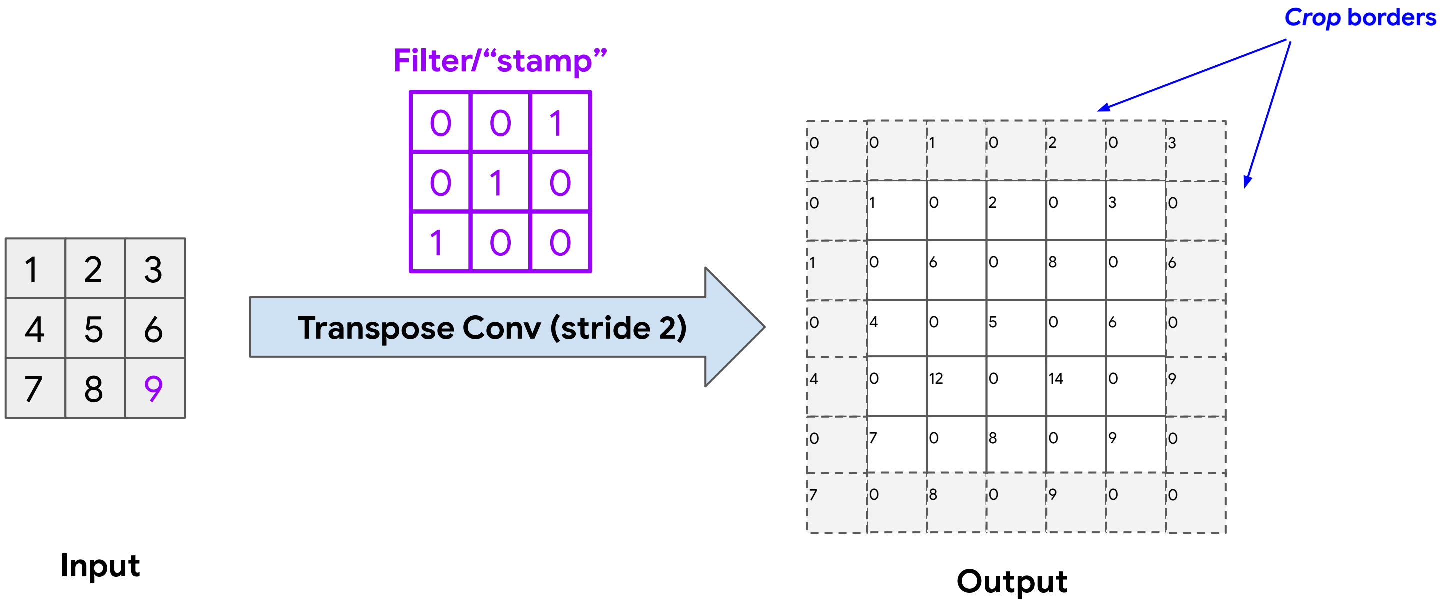
Think of "stamping" filter across the output image

Transpose Conv (2-d example)



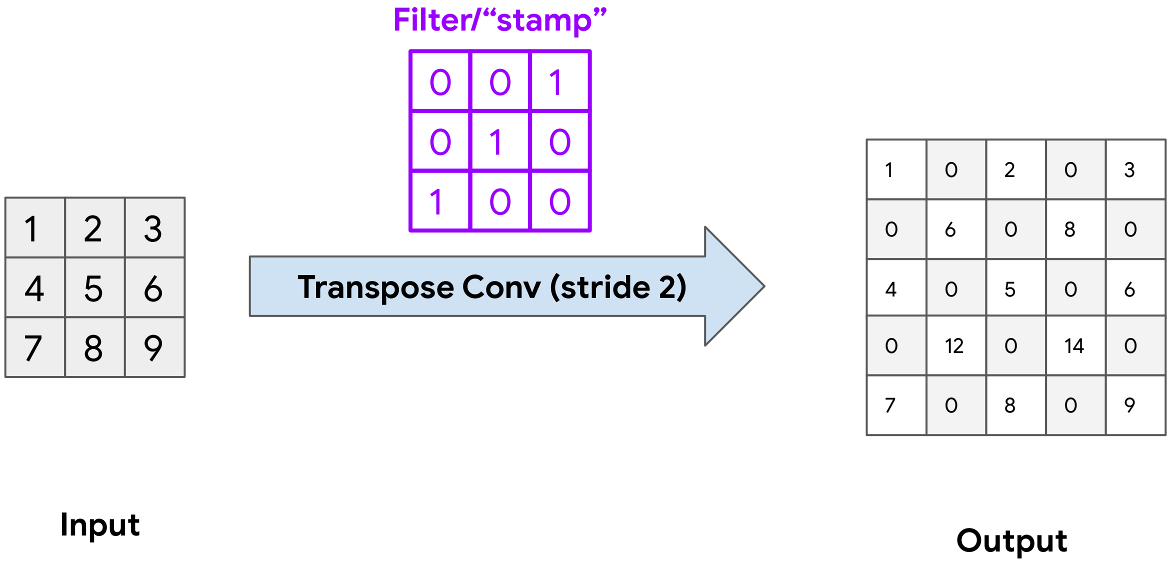
Think of "stamping" filter across the output image

Transpose Conv (2-d example)



Think of “stamping” filter across the output image

Transpose Conv (2-d example)



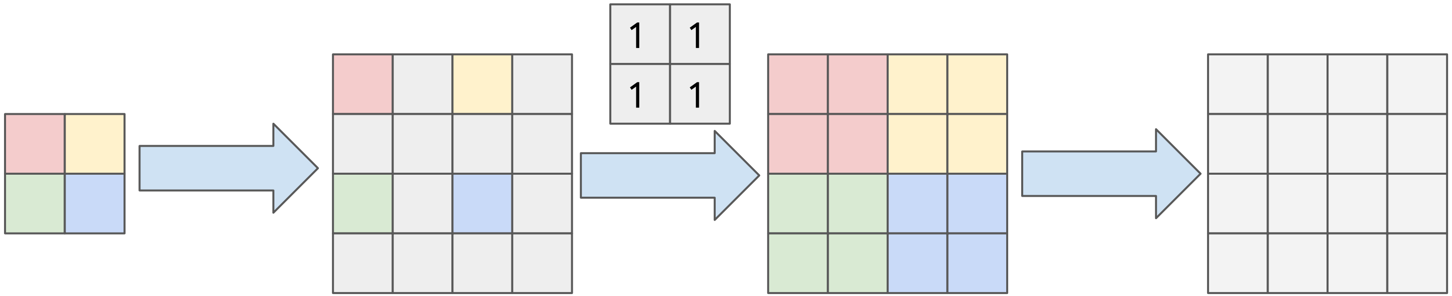
Think of “stamping” filter across the output image

Which one should I use??

- Fractional / Sub-pixel Convolution
- Transpose Convolution
- Convolution + “Periodic Reshuffling”
- Resize + Conv

Representationally Equivalent!

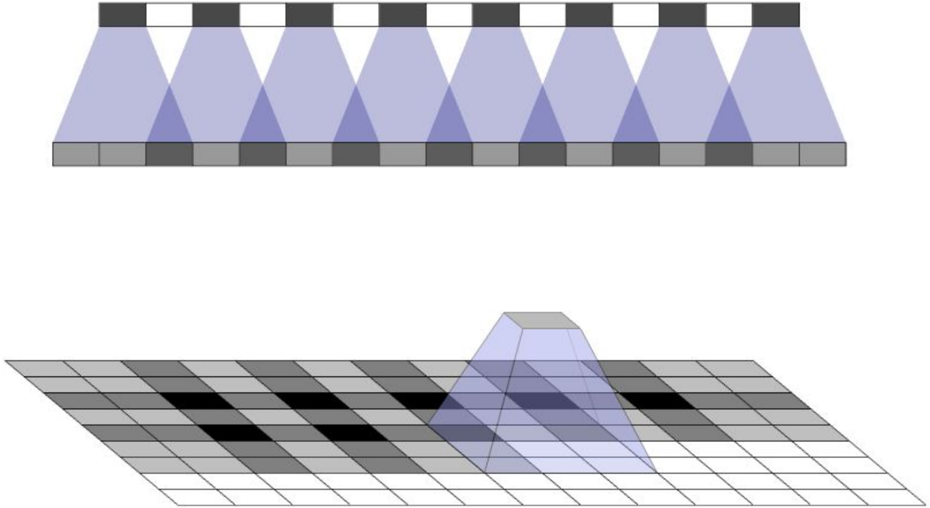
Slightly less expressive



Resize + Conv equivalent to Bed-of-Nails followed by an “all ones” 2x2 Conv then ordinary Conv

Checkerboard artifacts

Transpose Convolutions "want" to generate checkerboards



Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

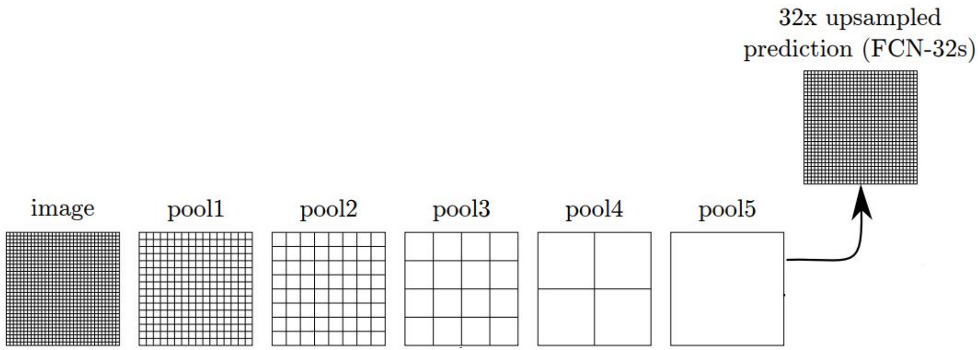
Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

All layers use resize-convolution.
No artifacts.

Resize + Conv less expressive than Transpose Conv, but less susceptible to checkerboard artifacts

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

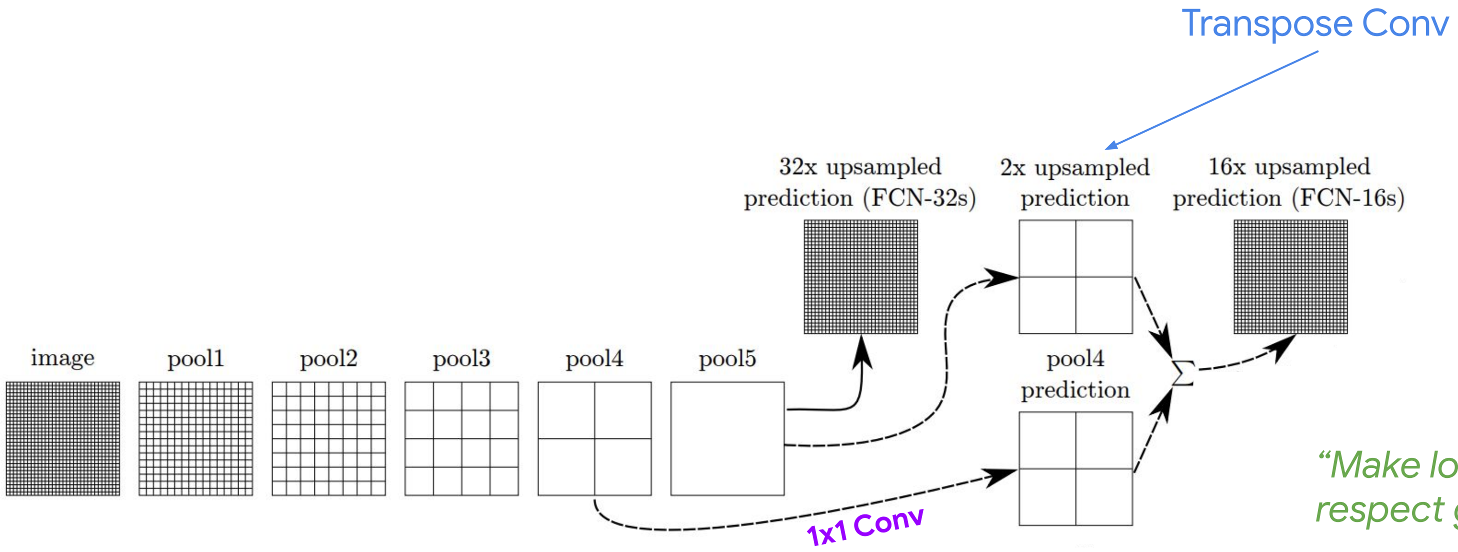
Case Study (2015): FCN



VGG-based FCN (stride 32)

Fully Convolutional Networks for Semantic Segmentation by Long et al.

Case Study (2015): FCN

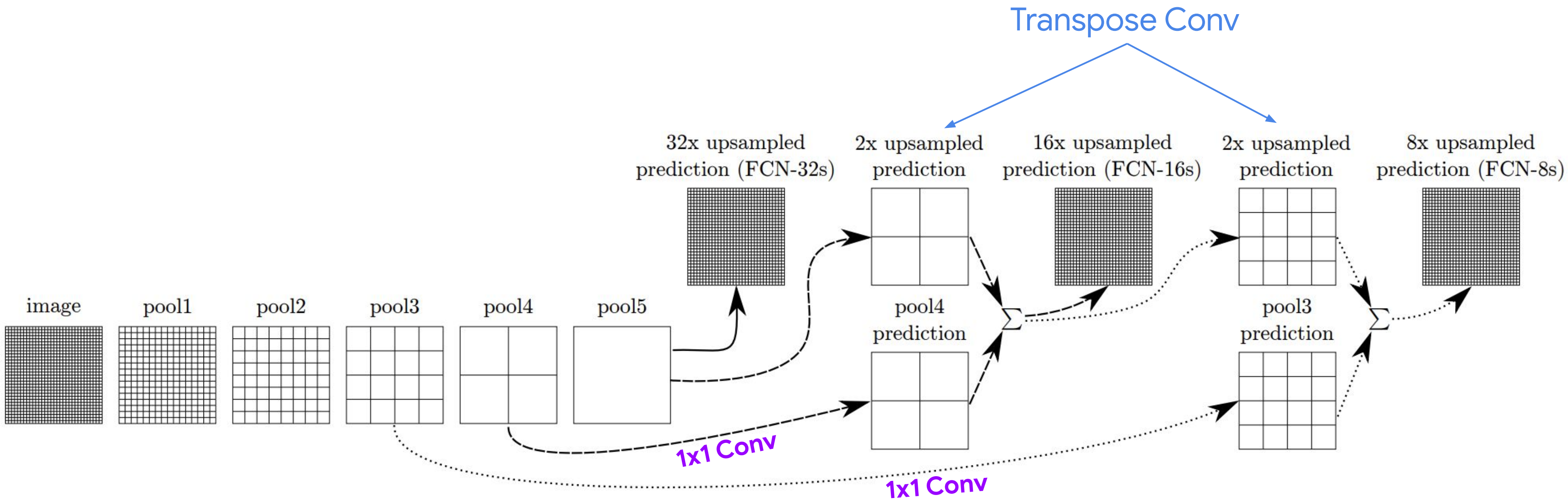


"Make local predictions that respect global structure"

VGG-based FCN (stride 16)

Fully Convolutional Networks for Semantic Segmentation by Long et al.

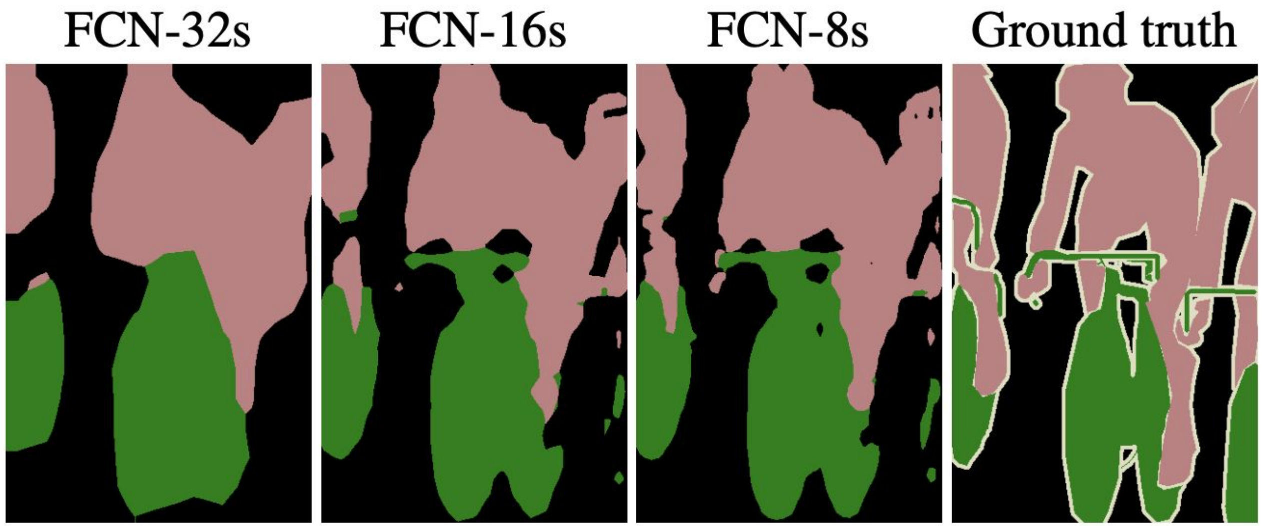
Case Study (2015): FCN



VGG-based FCN (stride 8)

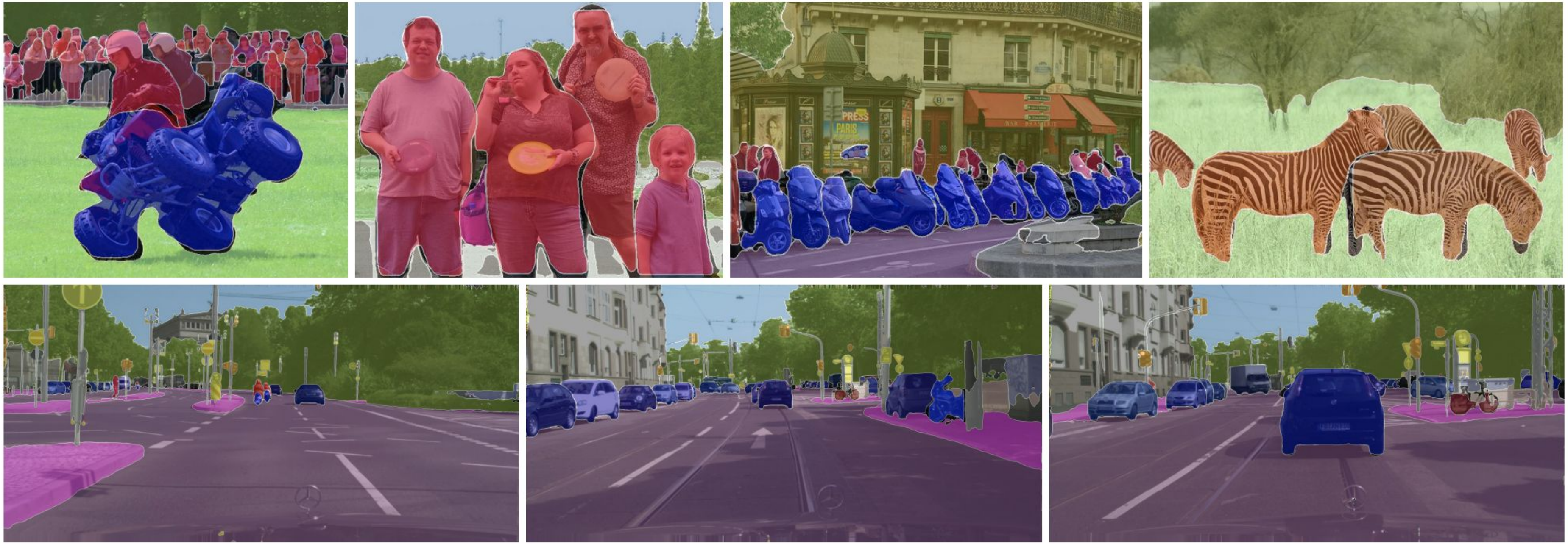
Fully Convolutional Networks for Semantic Segmentation by Long et al.

Case Study (2015): FCN



Fully Convolutional Networks for Semantic Segmentation by Long et al.

Case Study (2019): FPN (revisited)



Panoptic Feature Pyramid Networks by Kirillov et al

Case Study(2018) DeepLabV3+

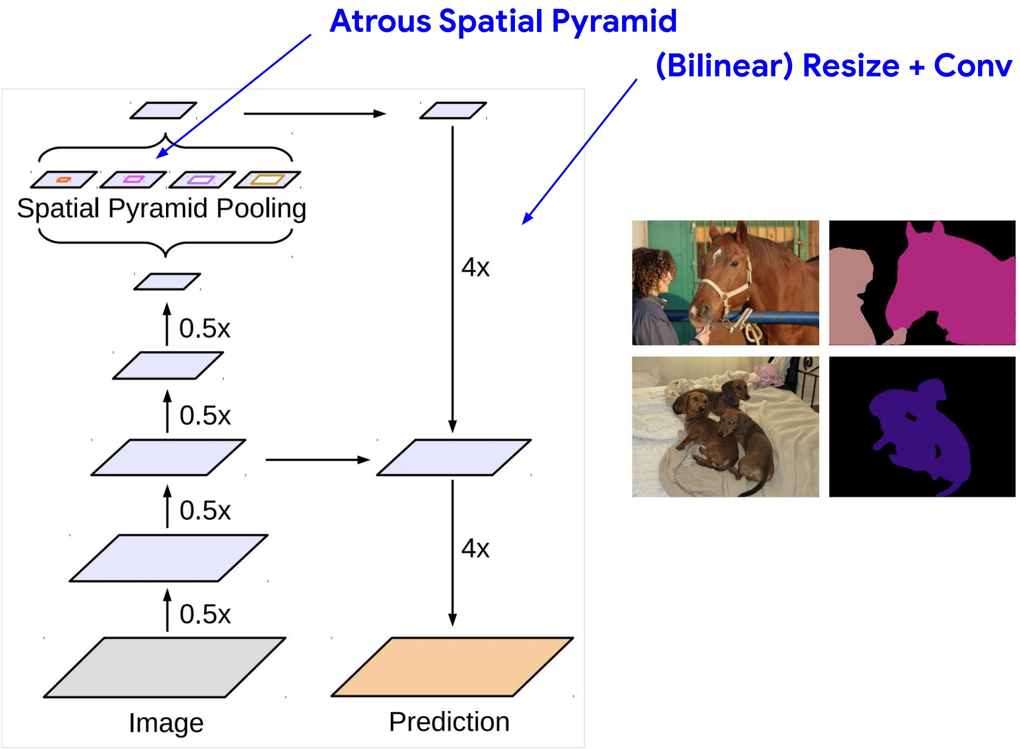


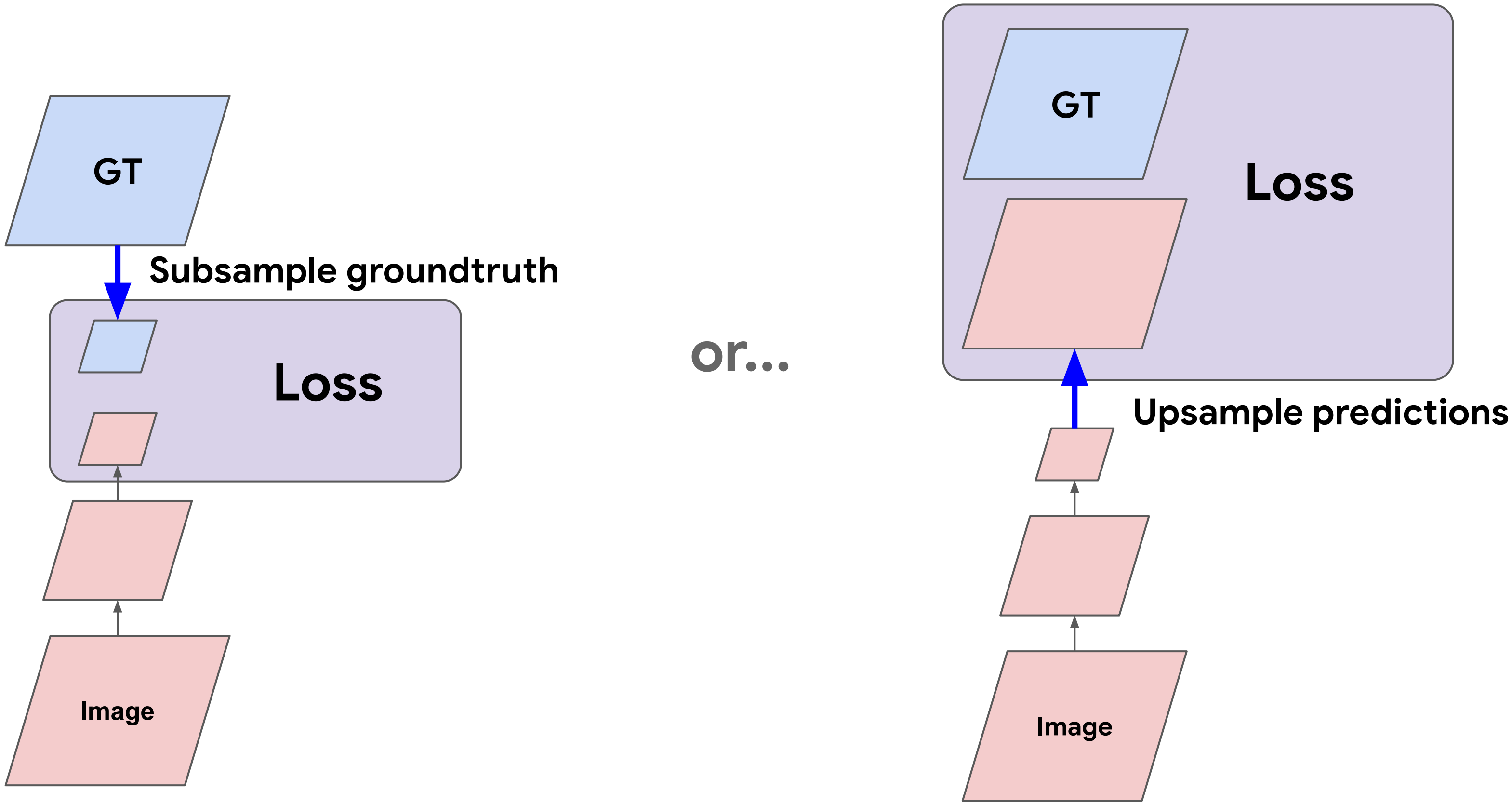
Figure from Chen et al

Outline of Semantic Segmentation

- The sliding window connection (again)
- Fully Convolutional models
- How to get high resolution outputs with
  - Atrous convolutions
  - “Upconvolutions”
- **Target Assignment**
- Evaluation of Semantic Segmentation

Relevant for all dense prediction tasks

Target Assignment / Alignment

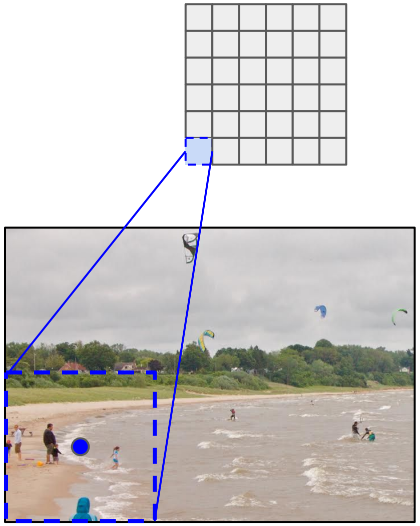


Target Assignment / Alignment

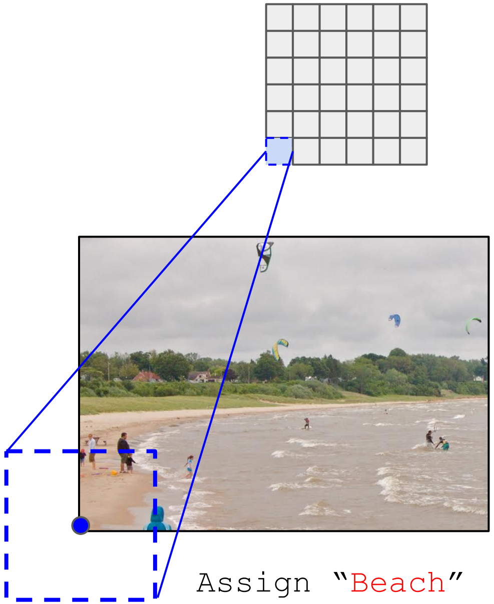
A reasonable desideratum: *groundtruth target for a particular logit should be sampled at center of that prediction's receptive field*

- *Getting this right requires thinking about padding, specific resizing algorithm*

"Valid padded" network



`Assign "Water"`



"Same padded" network

`Assign "Beach"`

Recap

We want

- High output resolution
- Large receptive fields
- “Alignment” between receptive fields and targets

Outline of Semantic Segmentation

- The sliding window connection (again)
- Fully Convolutional models
- How to get high resolution outputs with
  - Atrous convolutions
  - "Upconvolutions"
- Target Assignment
- **Evaluation of Semantic Segmentation**

Relevant for all dense prediction tasks

How to evaluate a segmentation model: Per-Pixel Accuracy

Problem with per-pixel accuracy --- not fair to small/thin classes

Categories: Water, Land

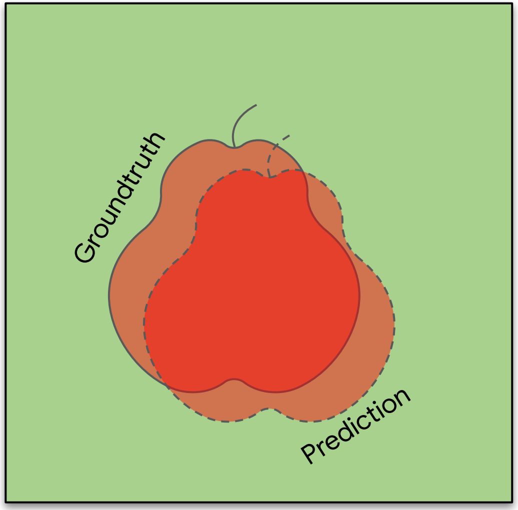
How to evaluate a segmentation model: Per-Pixel Accuracy

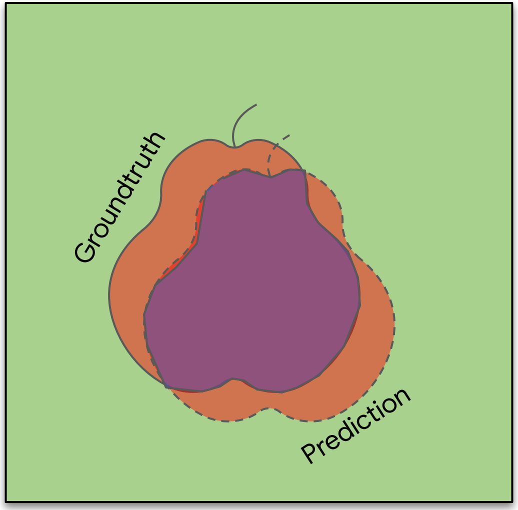
Problem with per-pixel accuracy --- not fair to small/thin classes

*Setting every pixel to “**Land**” is >90% Accuracy*

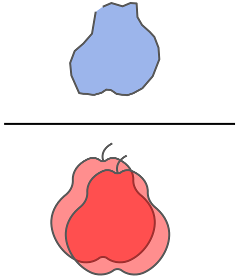
Categories: **Water**, **Land**

How to evaluate a segmentation model: "Mask IOU"





$$\mathrm{IOU} = \frac{\mathrm{Intersection}}{\mathrm{Union}}$$



- Masks are disjoint if and only if IOU=0
- Masks are identical if and only if IOU=1

How to evaluate a semantic segmentation model

Image

Groundtruth

Prediction

Mean IOU = Mean(IOU(groundtruth_c, predicted_c)
for c in {Sky, Building, Pole, ...})

[Figure](#) from Jégou et al.

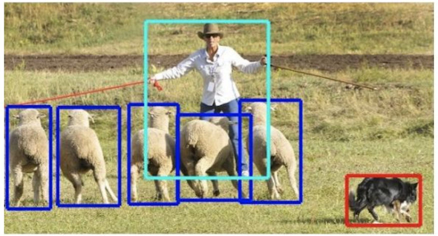
Lecture Outline

Dense Prediction (pixel level prediction)

- Semantic Segmentation
- **Instance Segmentation**
- **Panoptic Segmentation**
- **Keypoint Estimation**

Semantic vs Instance Segmentation: Don’t get confused!

classify



classify and regress
bounding box per object

**(bounding box)
detection**

classify per pixel

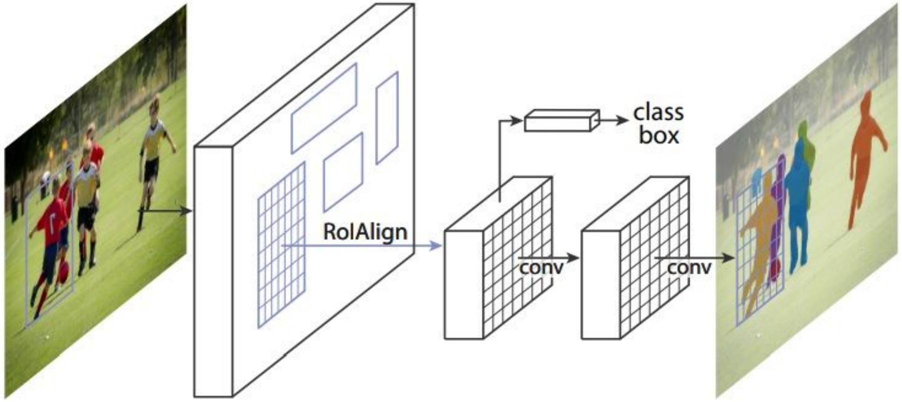
**semantic
segmentation**

classify per pixel per object

**instance
segmentation**

Figure from Lin et al 2014

Mask R-CNN



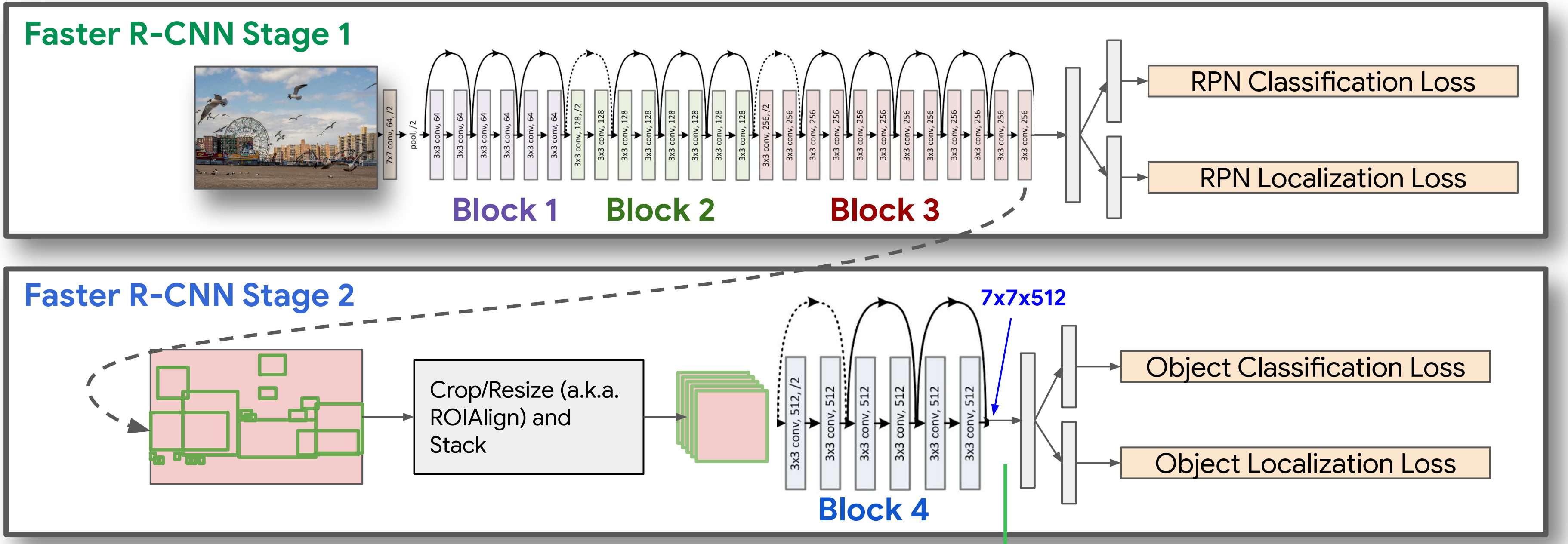
Boxes first paradigm:

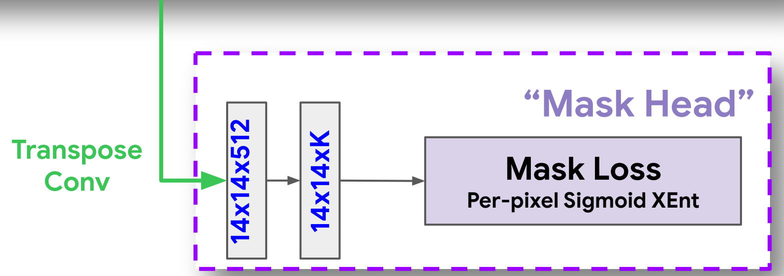
1. Run detector (Faster R-CNN)
2. Produce segmentation relative to each predicted box

Mask R-CNN combines both steps into an end-to-end trainable model

Mask R-CNN by He et al, 2017

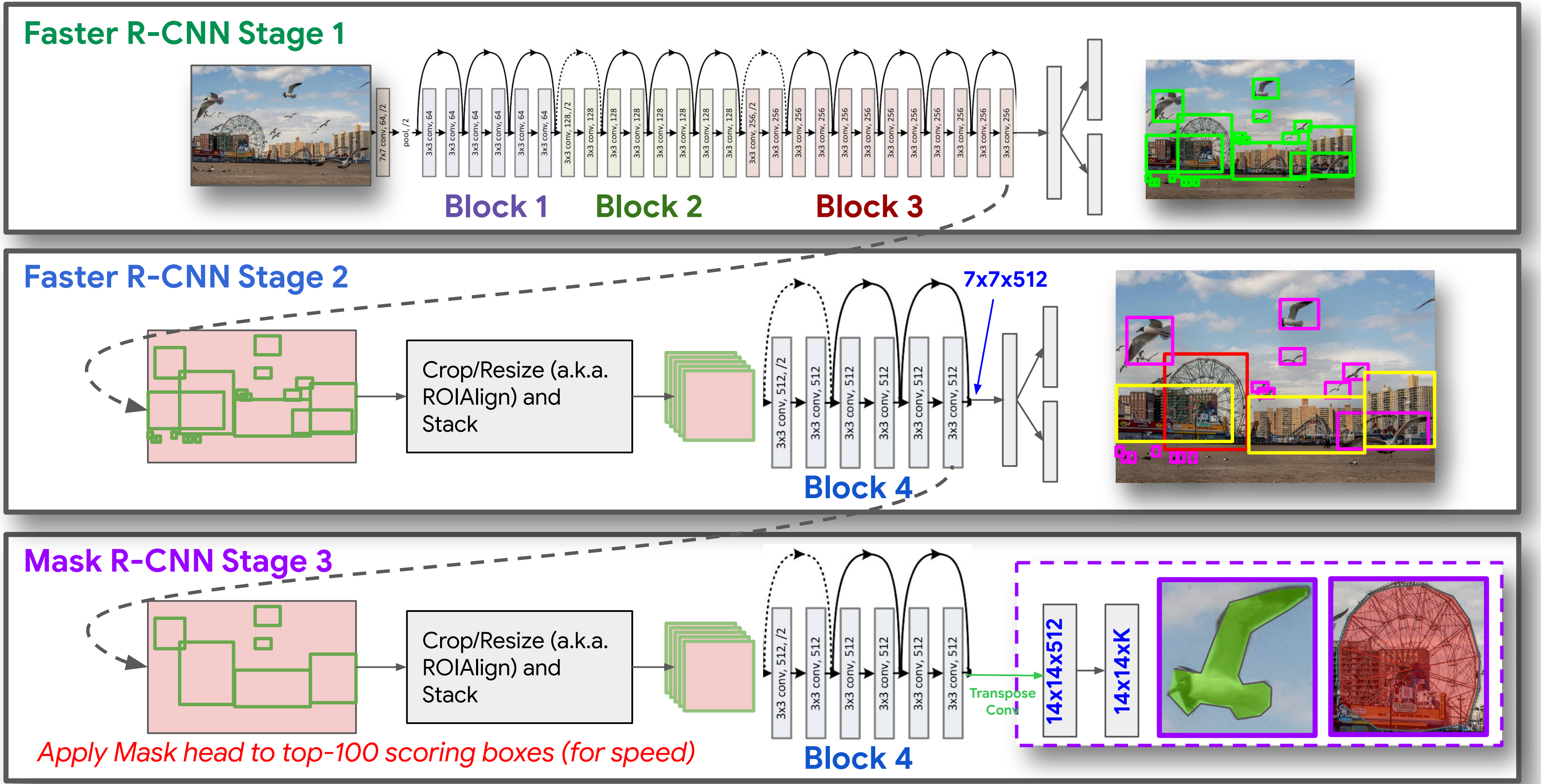
Mask R-CNN Training





Note: exact dimensions in this figure are a bit off as this figure is based on "basic residual unit"

Mask R-CNN Inference

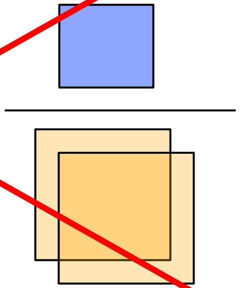


Apply Mask head to top-100 scoring boxes (for speed)

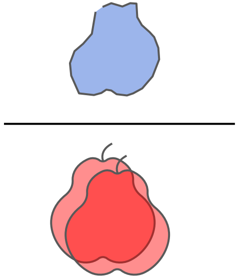
Evaluation for Instance Segmentation

- We care about the same things as object detection
  - E.g. Precision, Recall, Average Precision (AP), mean Average Precision (mAP)

$$\cancel{\texttt{Box IOU} = \frac{\texttt{Intersection}}{\texttt{Union}}}$$



$$\texttt{Mask IOU} = \frac{\texttt{Intersection}}{\texttt{Union}}$$

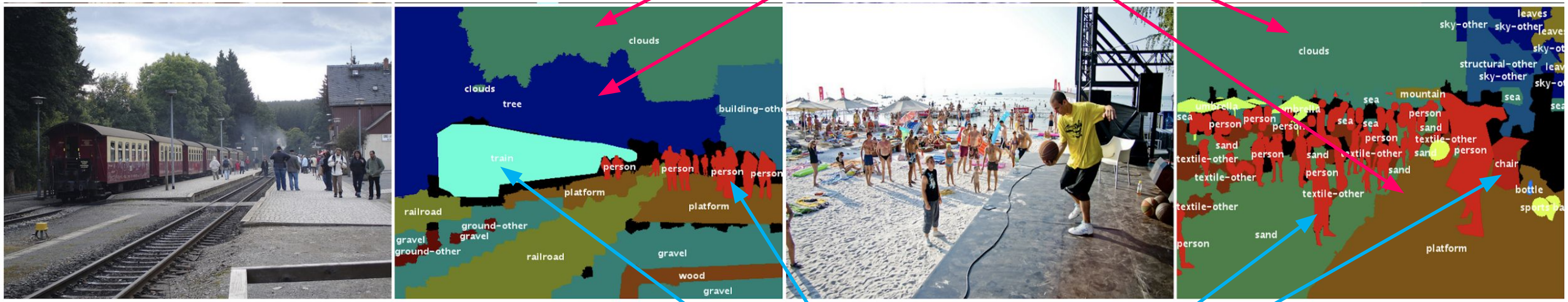


But... with Mask IOU instead of Box IOU

Stuff vs Things

Semantic segmentation makes more sense

Stuff

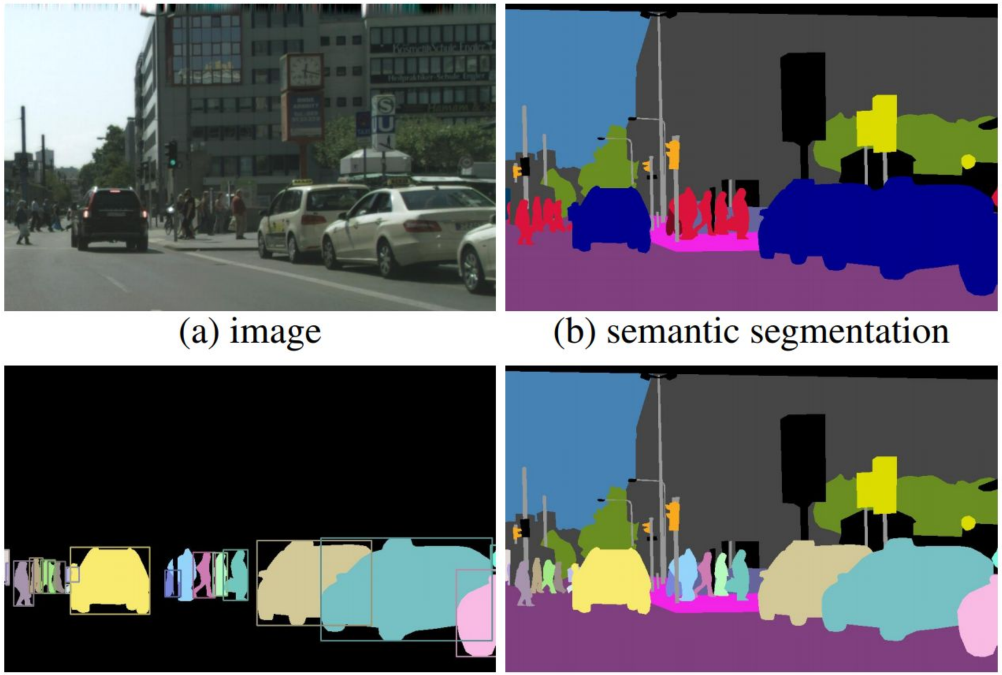


Things

Instance segmentation makes more sense

Figure from Caesar et al 2018

Handle both stuff *and* things: Panoptic Segmentation



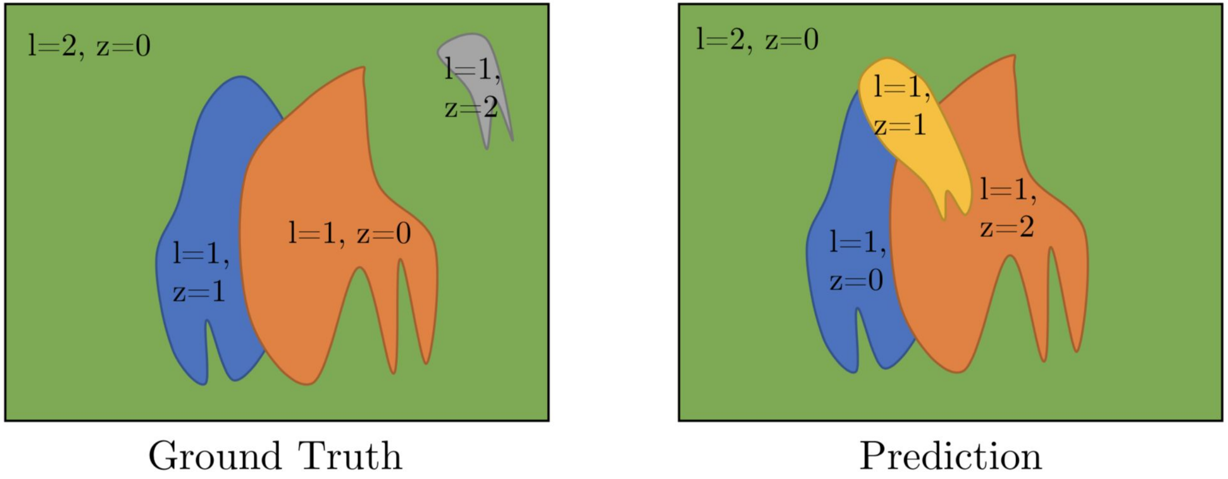
(a) image (b) semantic segmentation

(c) instance segmentation (d) panoptic segmentation

- Assign (category, instance id) pair to each pixel in image.
- Instance label ignored for "stuff" categories.

Kirillov et al, 2018

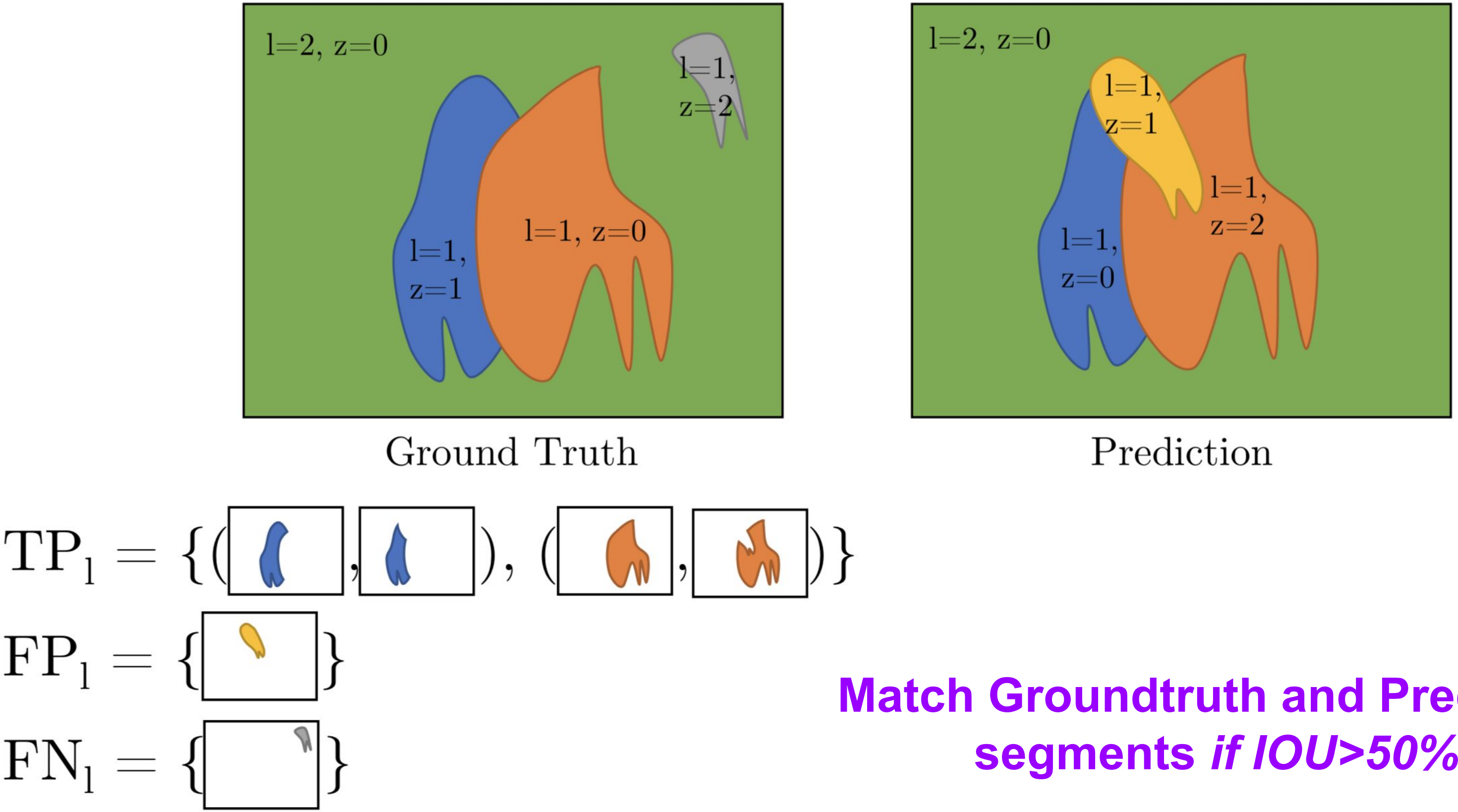
Measuring Panoptic Quality



	mAP	mIOU
Things	Standard for thing categories (instance segmentation)	Does not account for False Positives/Negatives
Stuff	Stuff segments typically do not come with a score needed to compute mAP	Standard for stuff categories (semantic segmentation)

Figure by Kirillov et al 2018

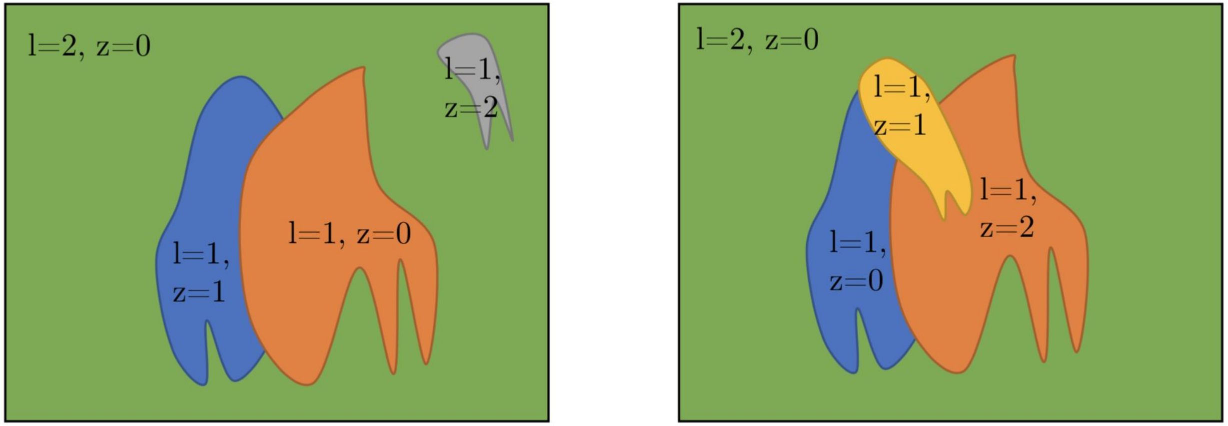
Measuring Panoptic Quality



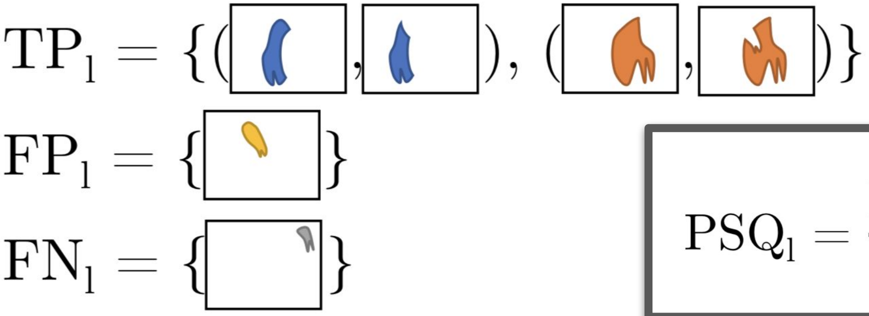
Match Groundtruth and Predicted segments *if IOU>50%*

Figure by Kirillov et al 2018

Measuring Panoptic Quality



Ground Truth Prediction



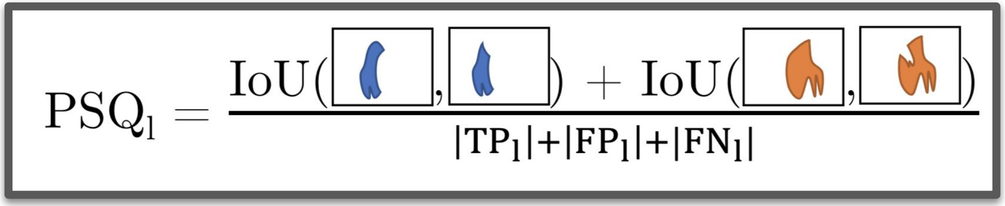
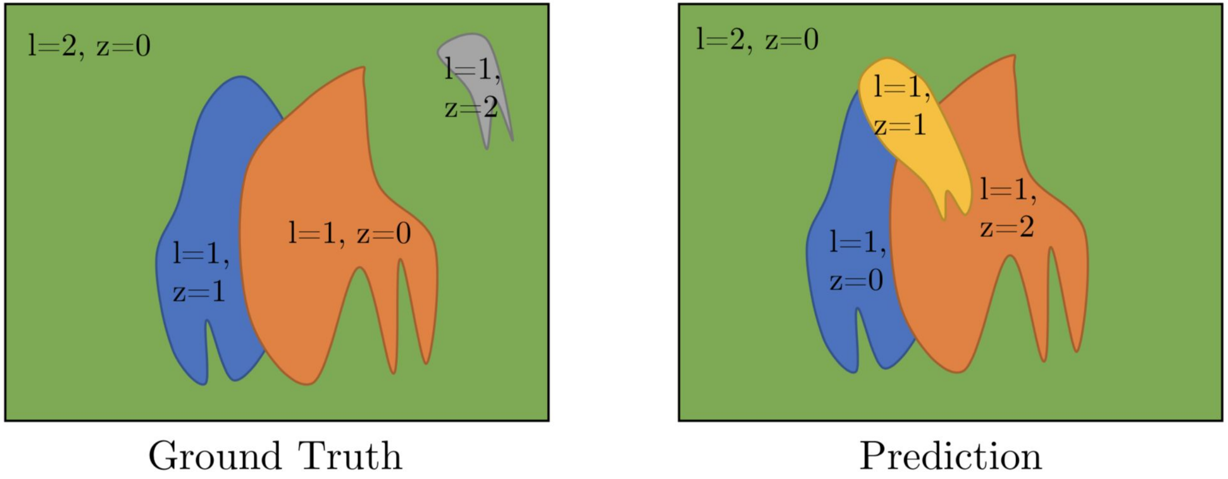


Figure by Kirillov et al 2018

Measuring Panoptic Quality

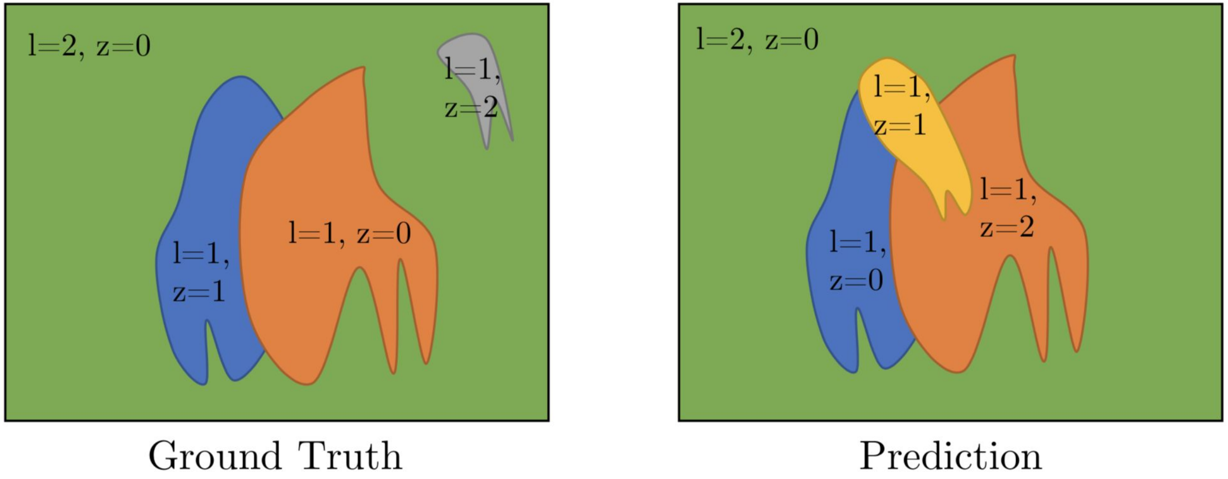


More generally:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathbf{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

Figure by Kirillov et al 2018

Measuring Panoptic Quality



More generally: $$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

*(= F_1-score * mIOU)*

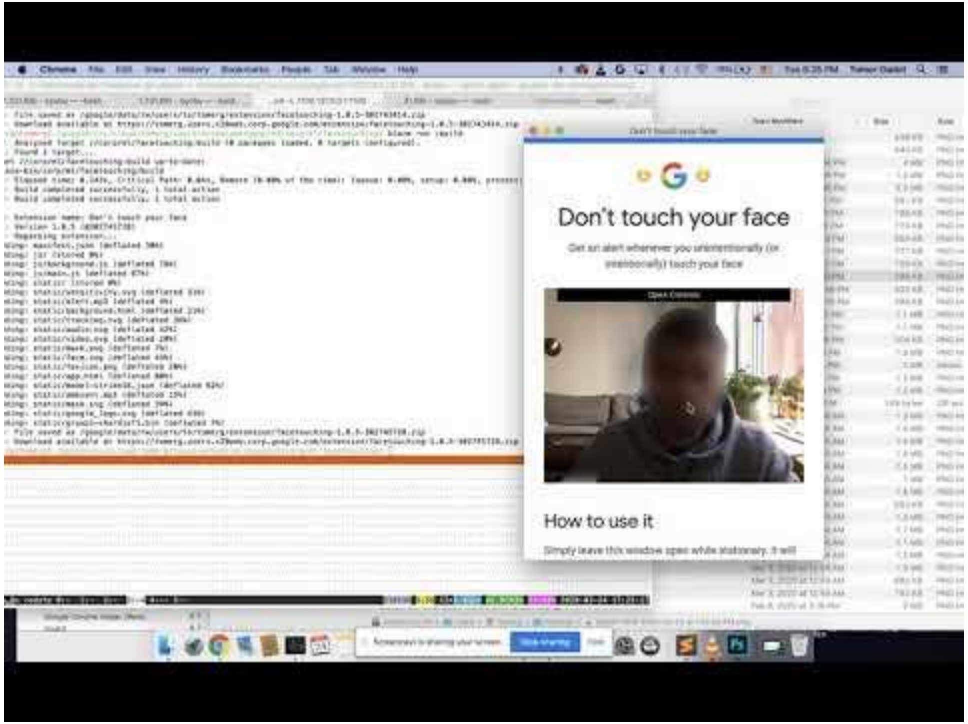
Another common detection metric

Figure by Kirillov et al 2018

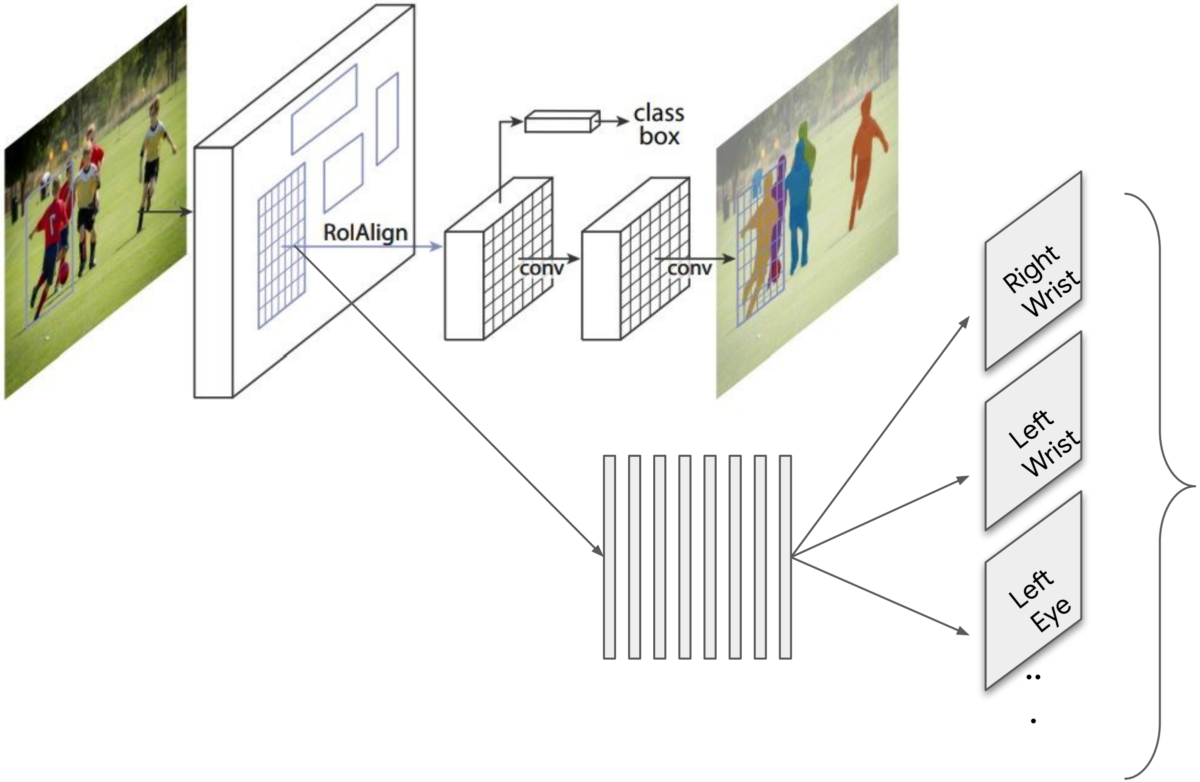
Keypoint Detection

Slide courtesy of George Papandreou, Tyler Zhu


imax PHOTGRAPHY
CHILE
brewers.com


Don't touch your face
How to use it

“Top-down” approach: Mask R-CNN



Predict heatmap for each pose keypoint

He et al, 2017

“Bottom up” approach: Predict keypoint positions (Step 1)

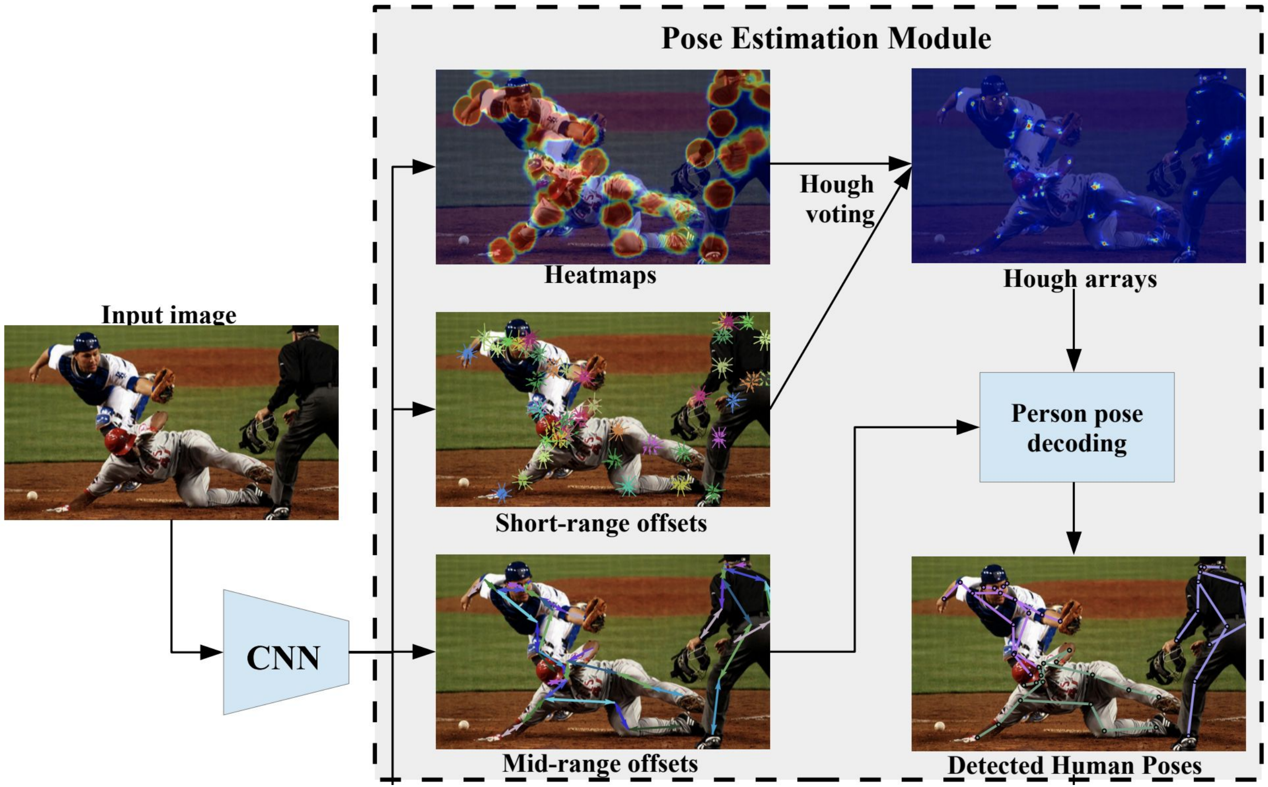
Image credit: DeeperCut paper

"Bottom up" approach: Group keypoints (Step 2)

vs.

Image Source

Example “bottom up” method: PersonLab



Papandreou, Zhu et al, 2018

“Bottom up” vs “Top down”

Performance on COCO keypoints task

	AP	$AP^{.50}$	$AP^{.75}$	AP^{M}	AP^{L}	AR	$AR^{.50}$	$AR^{.75}$	AR^{M}	AR^{L}
Bottom-up methods:										
CMU-Pose [32] (+refine)	0.618	0.849	0.675	0.571	0.682	0.665	0.872	0.718	0.606	0.746
Assoc. Embed. [2] (multi-scale)	0.630	0.857	0.689	0.580	0.704	-	-	-	-	-
Assoc. Embed. [2] (mscale, refine)	0.655	0.879	0.777	0.690	0.752	0.758	0.912	0.819	0.714	0.820
Top-down methods:										
Mask-RCNN [34]	0.631	0.873	0.687	0.578	0.714	0.697	0.916	0.749	0.637	0.778
G-RMI *COCO-only* [33]	0.649	0.855	0.713	0.623	0.700	0.697	0.887	0.755	0.644	0.771
PersonLab (ours):										
ResNet101 (single-scale)	0.655	0.871	0.714	0.613	0.715	0.701	0.897	0.757	0.650	0.771
ResNet152 (single-scale)	**0.665**	0.880	0.726	0.624	0.723	0.710	0.903	0.766	0.661	0.777
ResNet101 (multi-scale)	0.678	0.886	0.744	0.630	0.748	0.745	0.922	0.804	0.686	0.825
ResNet152 (multi-scale)	**0.687**	0.890	0.754	0.641	0.755	0.754	0.927	0.812	0.697	0.830

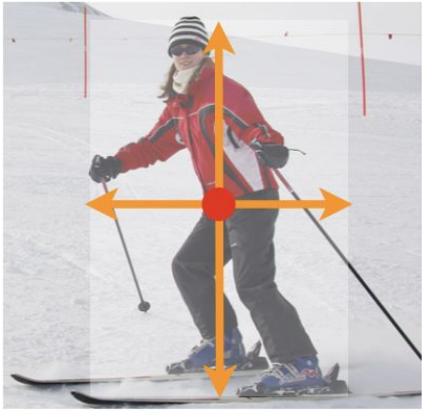
Papandreou, Zhu et al, 2018

Another example “bottom up” method: “Objects as Points”

Predict heatmap for each pose keypoint

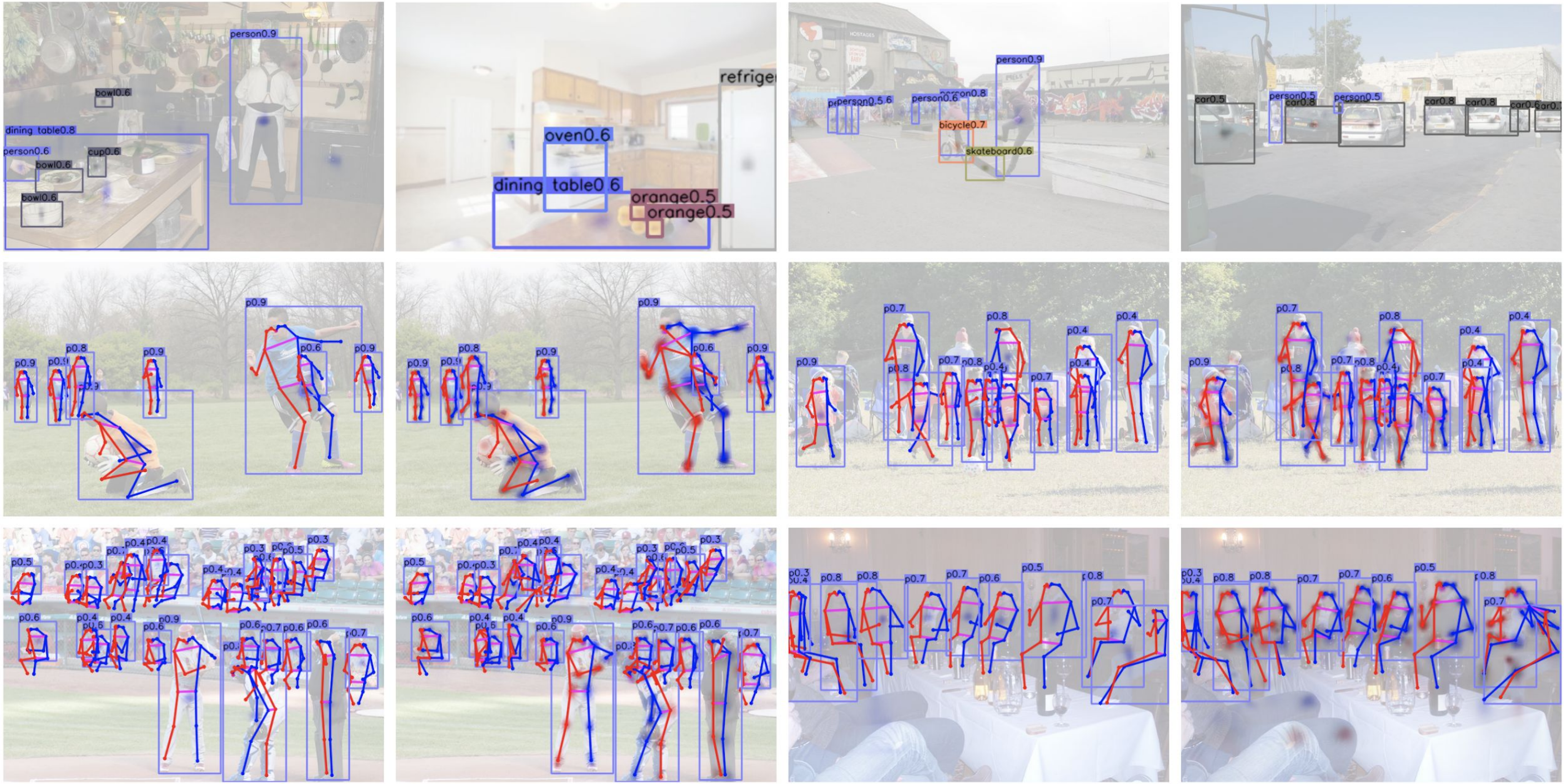
Predict heatmap for object center

Predict offset to each pose keypoint



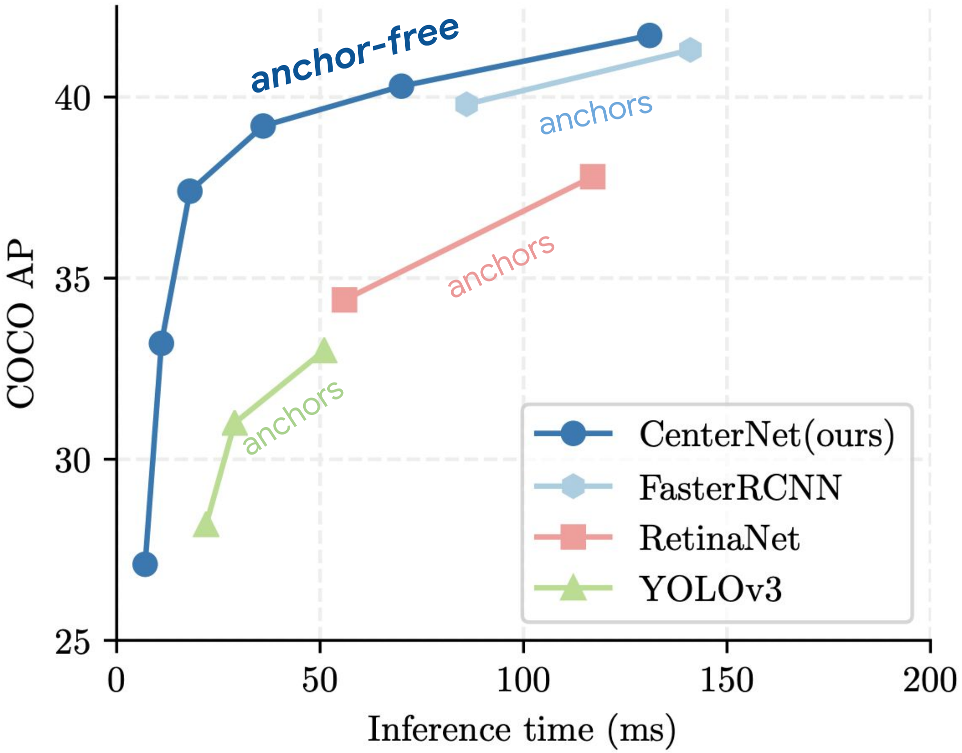
Predict object height/width

Zhou et al, 2019



Zhou et al, 2019

New kid on the block: "Anchor-free" object detection

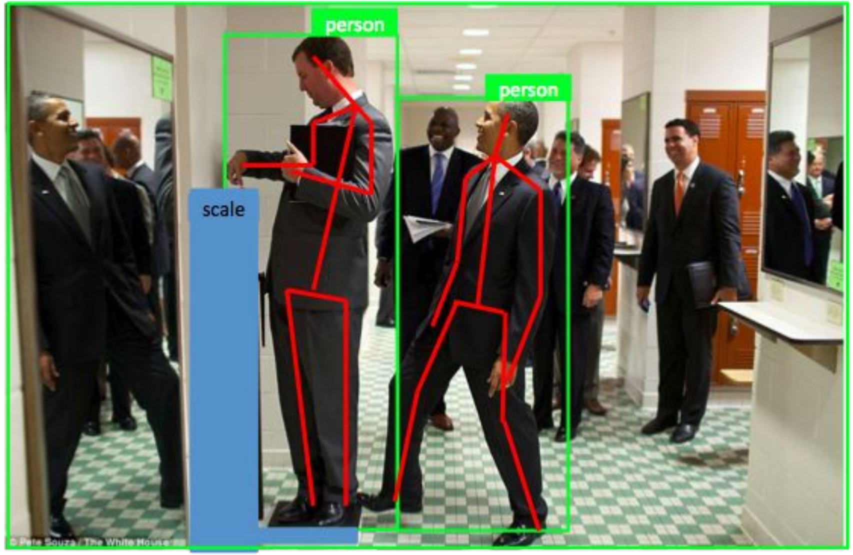


Zhou et al, 2019

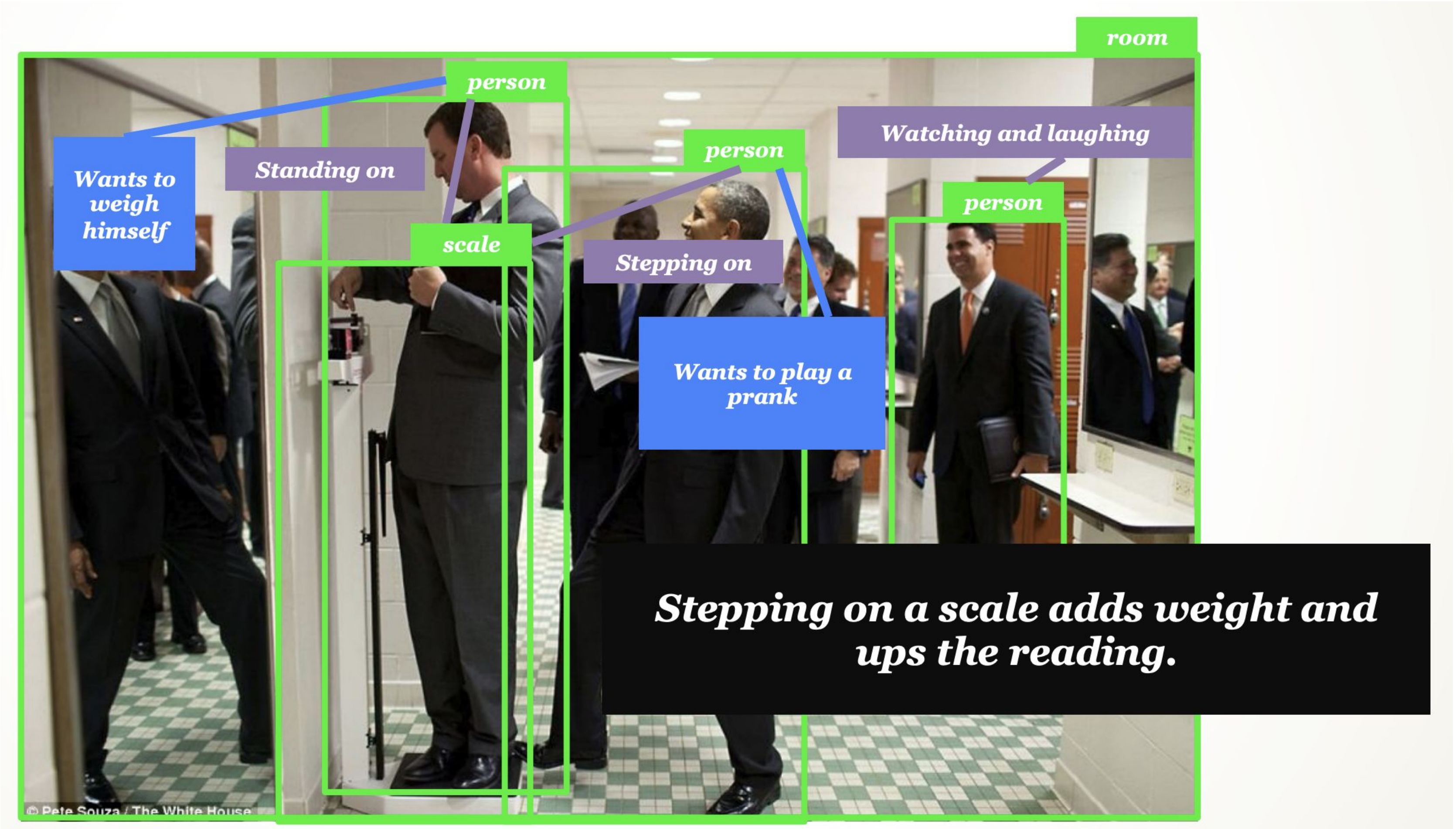
Wrap up

Dense Prediction (pixel level prediction)

- Semantic Segmentation
- Instance Segmentation
- Panoptic Segmentation
- Keypoint Estimation



Technology!



ImageNet: Where have we been? Where are we going? by Fei Fei Li and Jia Deng