CS163: Deep Learning for Computer Vision

Lecture 14:
Detection + Segmentation

Announcement

- Assignment 3 dues on Nov 24 (this Sunday)
- Assignment 4 will be released today
- Next Wednesday afternoon's lecture (right before thanksgiving break) will be independent study to watch records of research spotlight talks
  - We invite several student researchers (PhD and undergrad students) to talk about their research projects on computer vision
  - Each talk will be 5 – 10 mins, which you will watch and answer one question about the specific project, also you need to raise a question

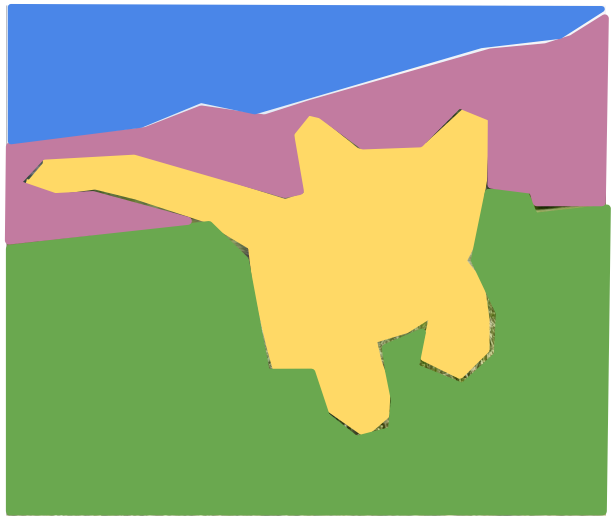
Last Time: Computer Vision Tasks

Classification

CAT

No spatial extent

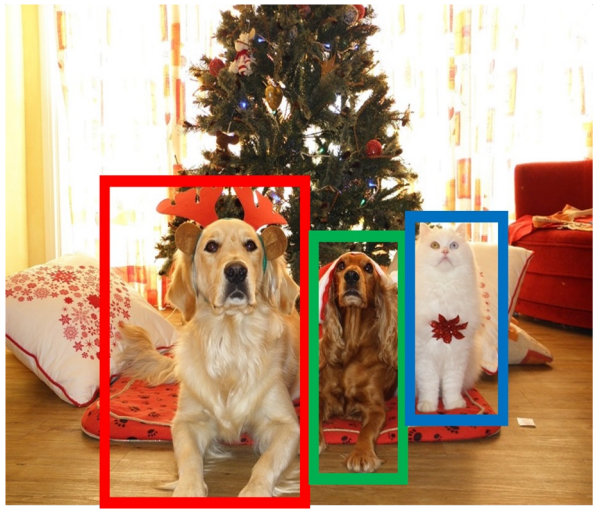
Semantic Segmentation



GRASS, **CAT**, **TREE**, **SKY**

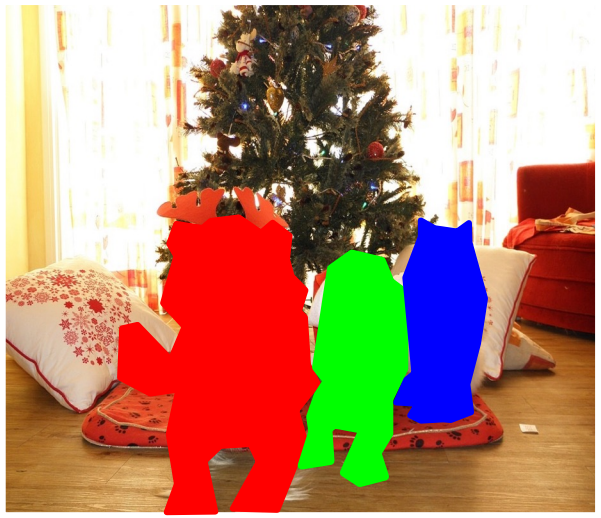
No objects, just pixels

Object Detection



DOG, **DOG**, **CAT**

Instance Segmentation



DOG, **DOG**, **CAT**

Multiple Objects

Last Time: Object Detection

Classification

CAT

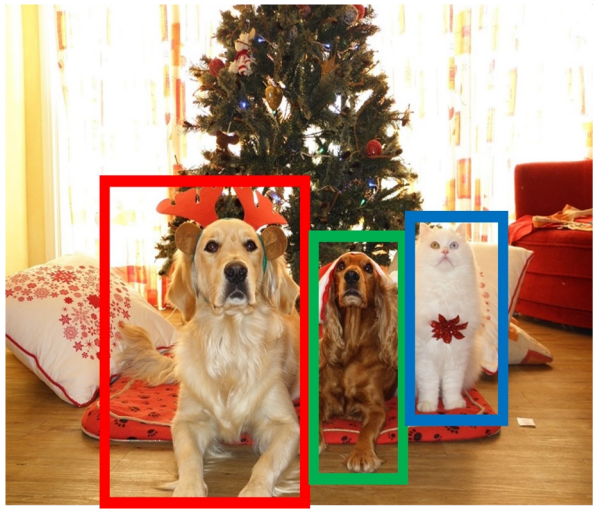
No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



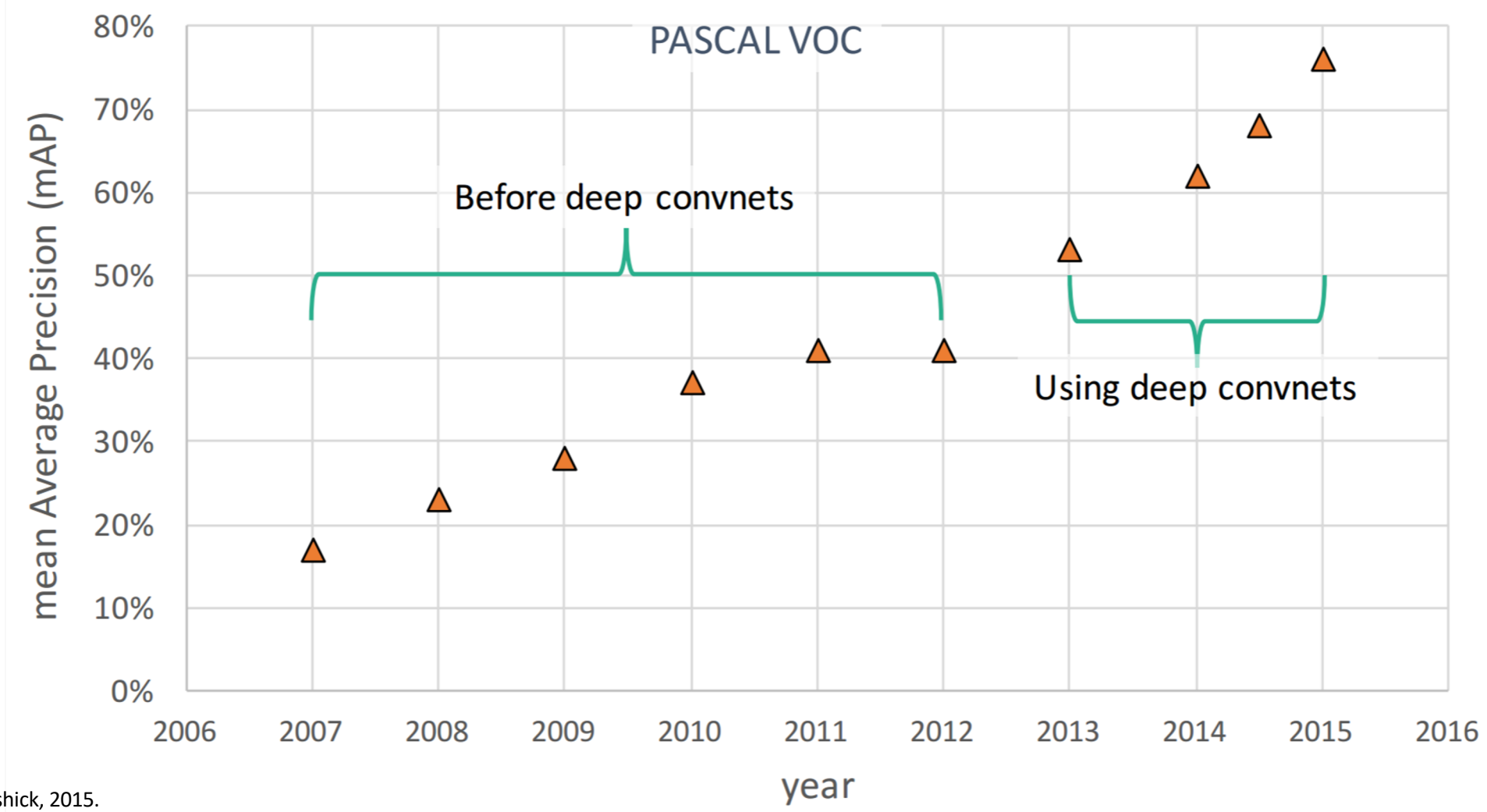
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

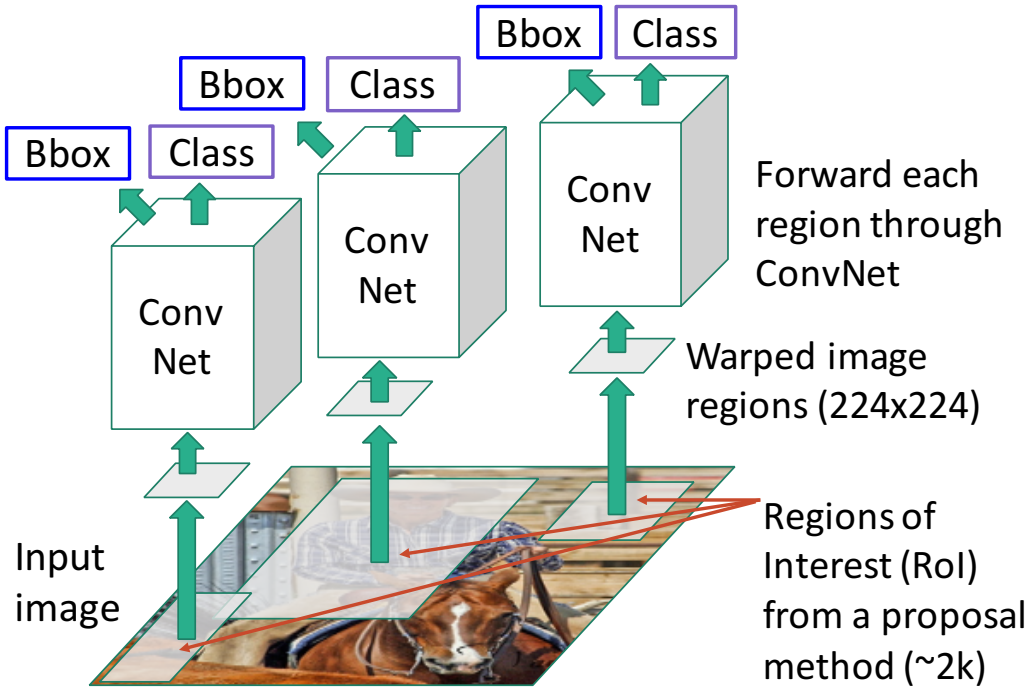
Multiple Objects

Object Detection: Impact of Deep Learning

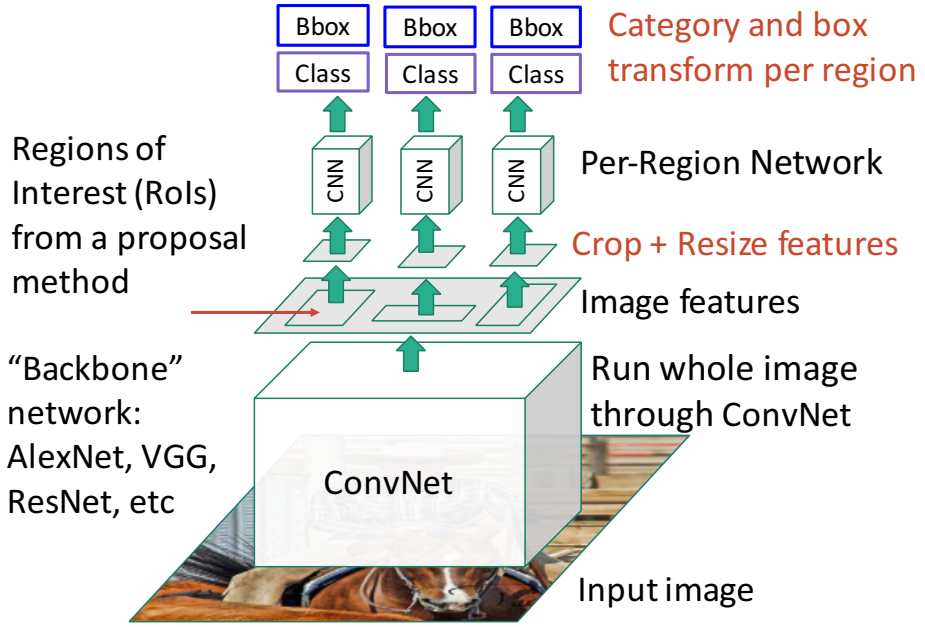


Last Time: Object Detection Methods

"Slow" R-CNN: Run CNN independently for each region

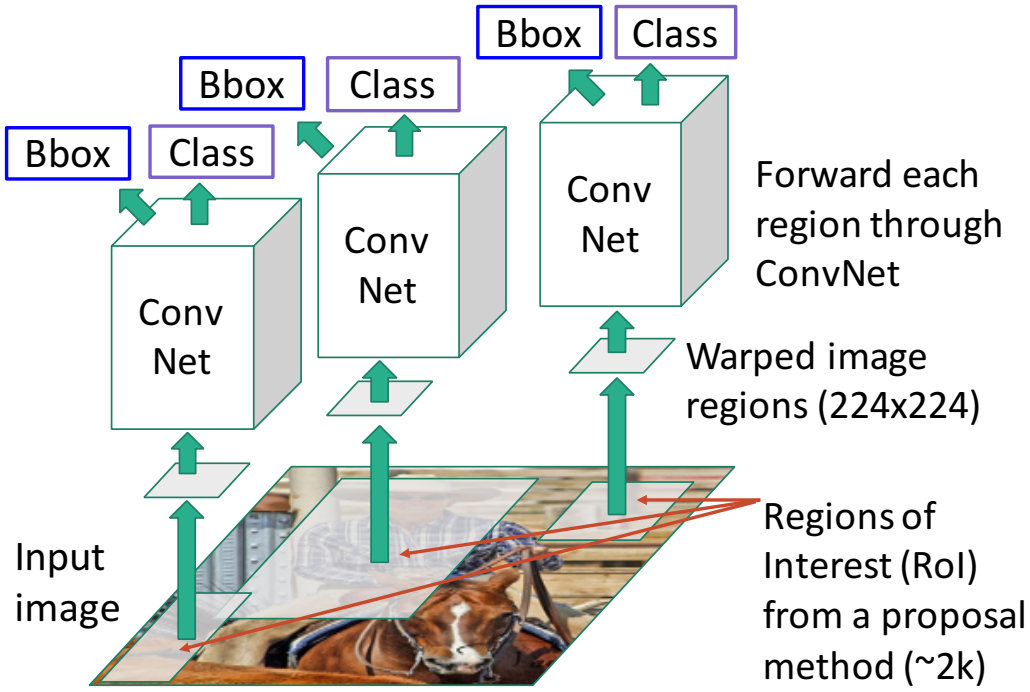


Fast R-CNN: Apply differentiable cropping to shared image features

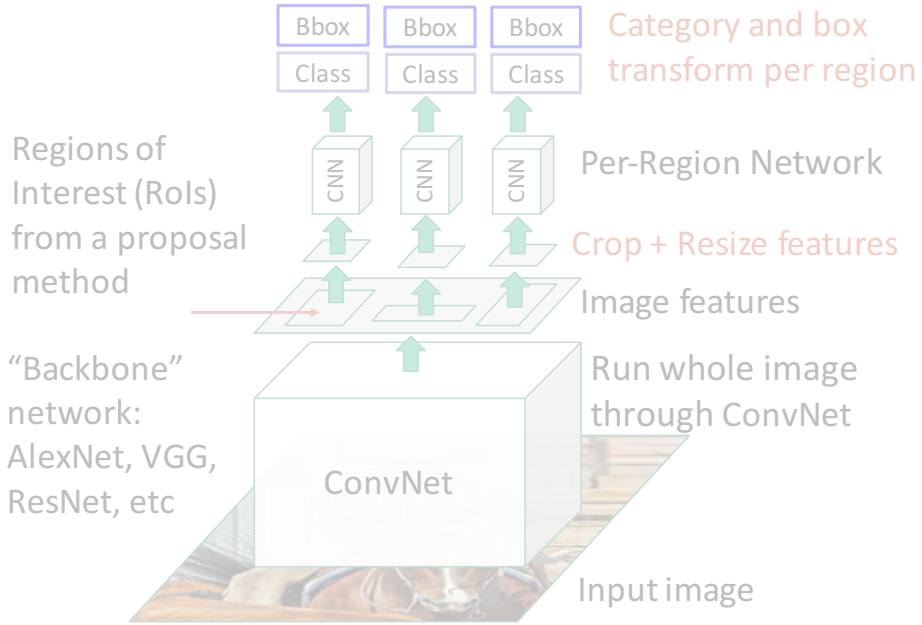


Recap: Slow R-CNN Training

"Slow" R-CNN: Run CNN independently for each region

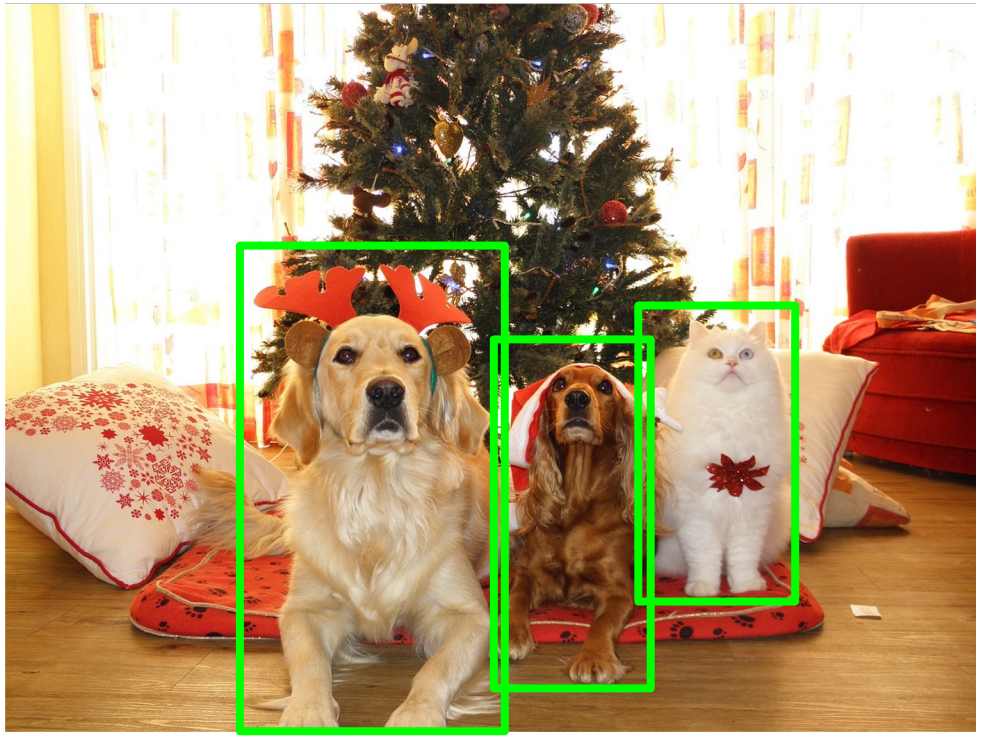


Fast R-CNN: Apply differentiable cropping to shared image features



”Slow” R-CNN Training

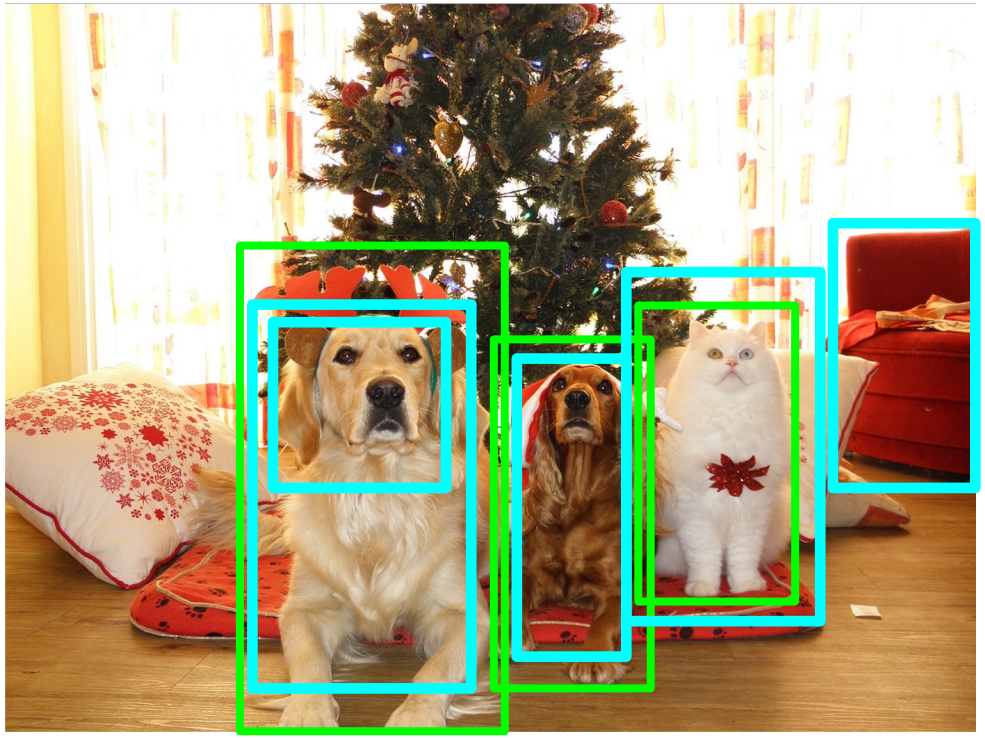
Input Image



Ground-Truth boxes

”Slow” R-CNN Training

Input Image

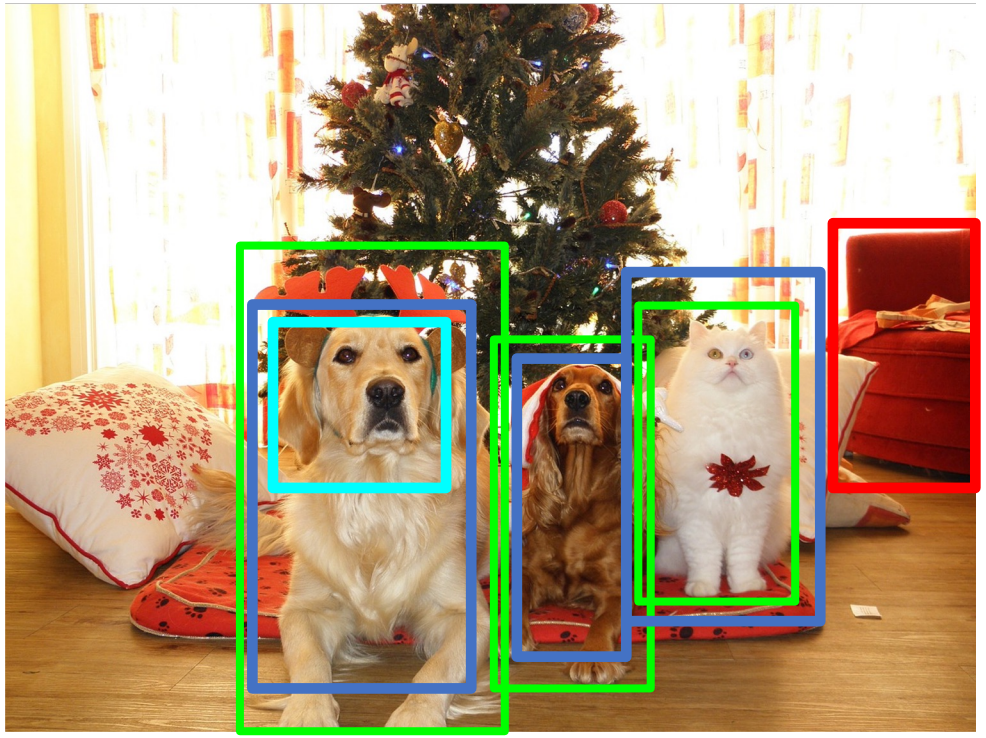


Ground-Truth boxes

Region Proposals

”Slow” R-CNN Training

Input Image



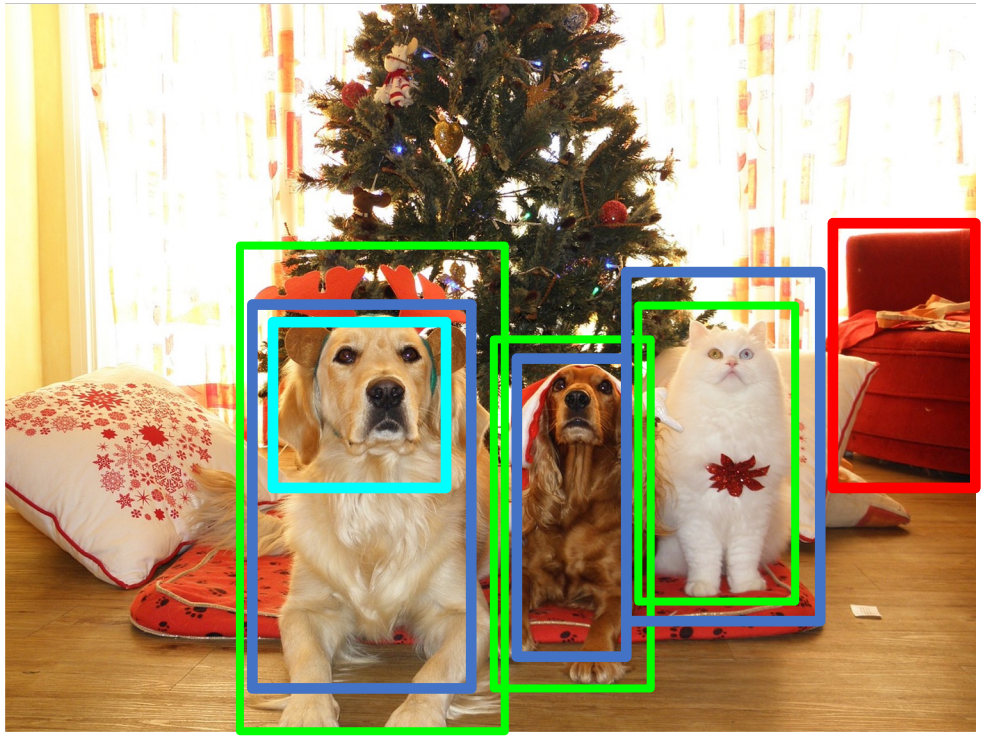
GT Boxes | Positive

Neutral | Negative

Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes

”Slow” R-CNN Training

Input Image



GT Boxes

Positive

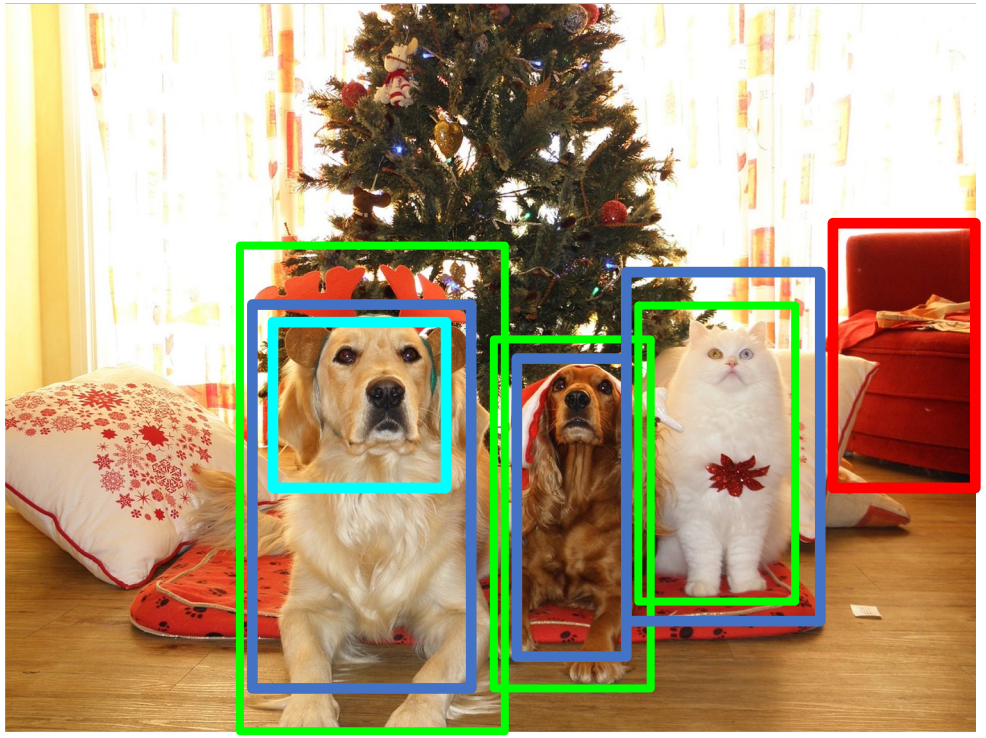
Neutral

Negative

Crop pixels from each positive and negative proposal, resize to 224 x 224

"Slow" R-CNN Training

Input Image



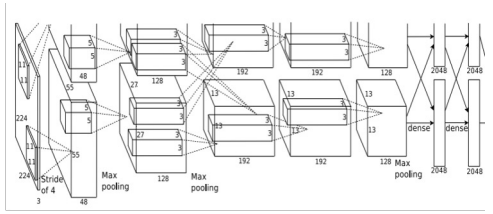
GT Boxes

Positive

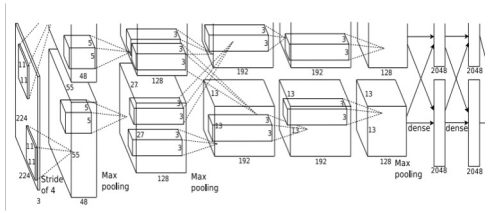
Neutral

Negative

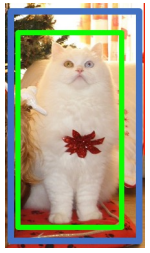
Run each region through CNN. For positive boxes predict class and box offset; for negative boxes just predict background class

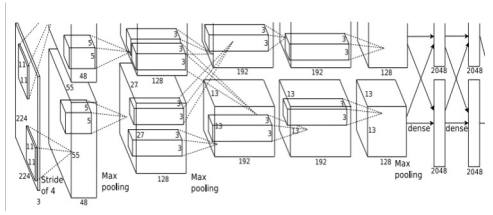


Class target: Dog
Box target: →

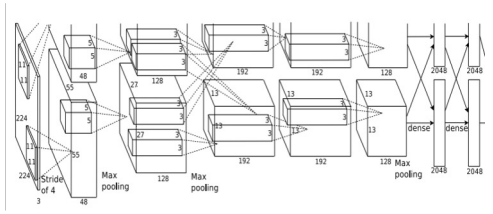


Class target: Cat
Box target: →





Class target: Dog
Box target: →

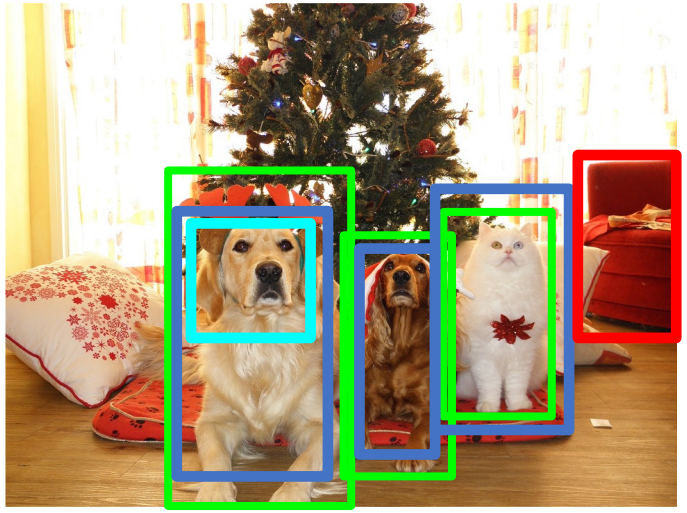


Class target: Background
Box target: None

Fast R-CNN Training

Crop features for each region, use them to predict class and box targets per region

Input Image



Backbone CNN

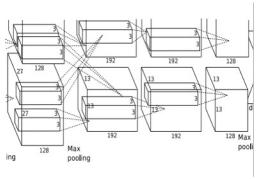
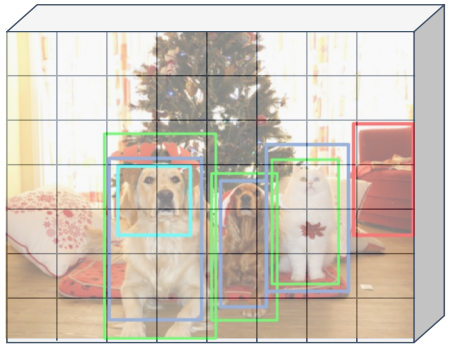
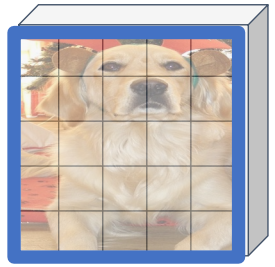
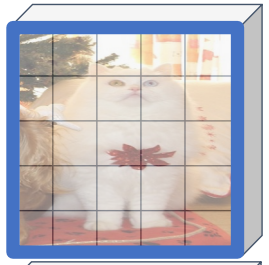


Image Features

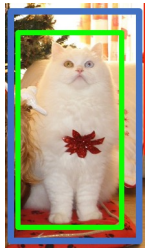


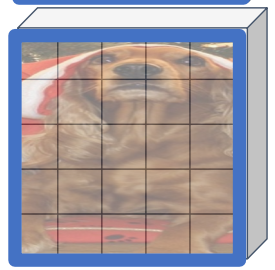


Class target: Dog
Box target:

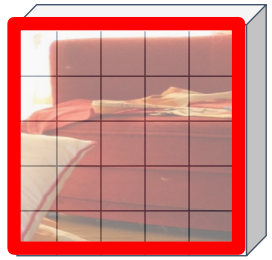


Class target: Cat
Box target:





Class target: Dog
Box target:



Class target: Background
Box target: None

GT Boxes | Positive

Neutral | Negative

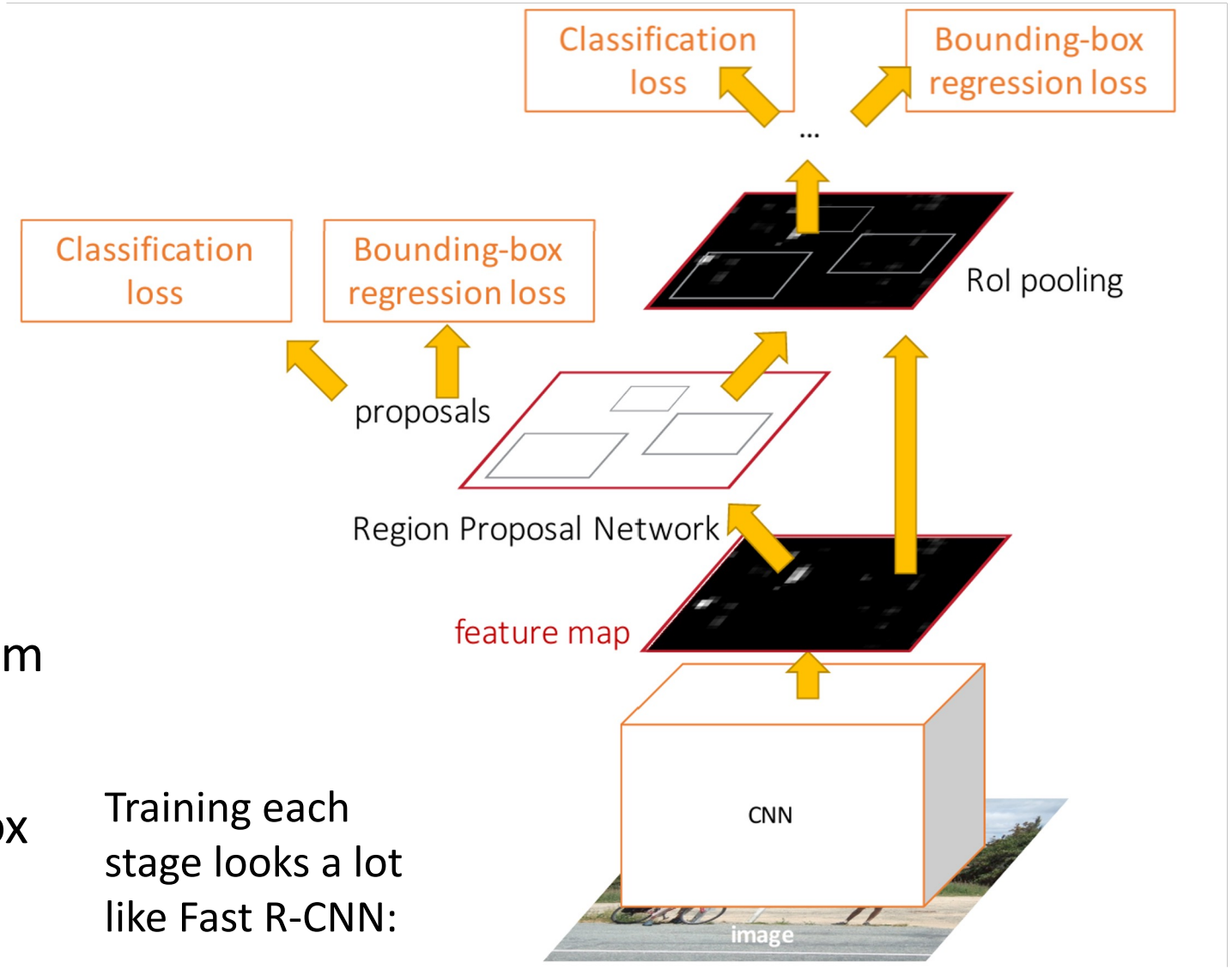
Faster R-CNN: Learnable Region Proposals

Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box

Anchor -> Region Proposal -> Object Box

(Stage 1) (Stage 2)

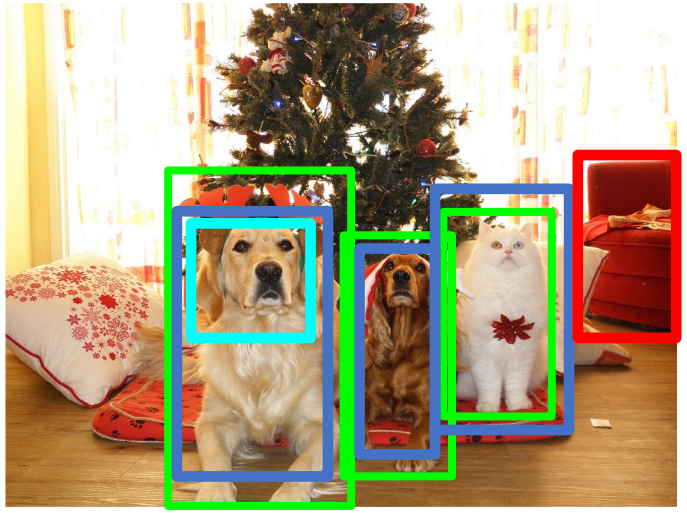


Training each stage looks a lot like Fast R-CNN:

Faster R-CNN Training: RPN Training

RPN predicts Object / Background for each anchor, as well as regresses from anchor to object box

Input Image



GT Boxes | Positive

Neutral | Negative

Backbone CNN

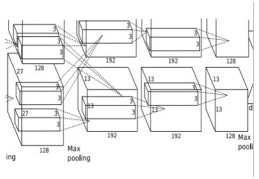
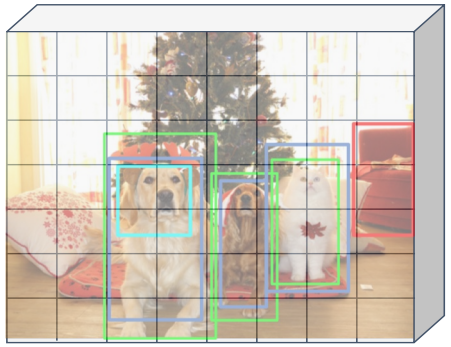
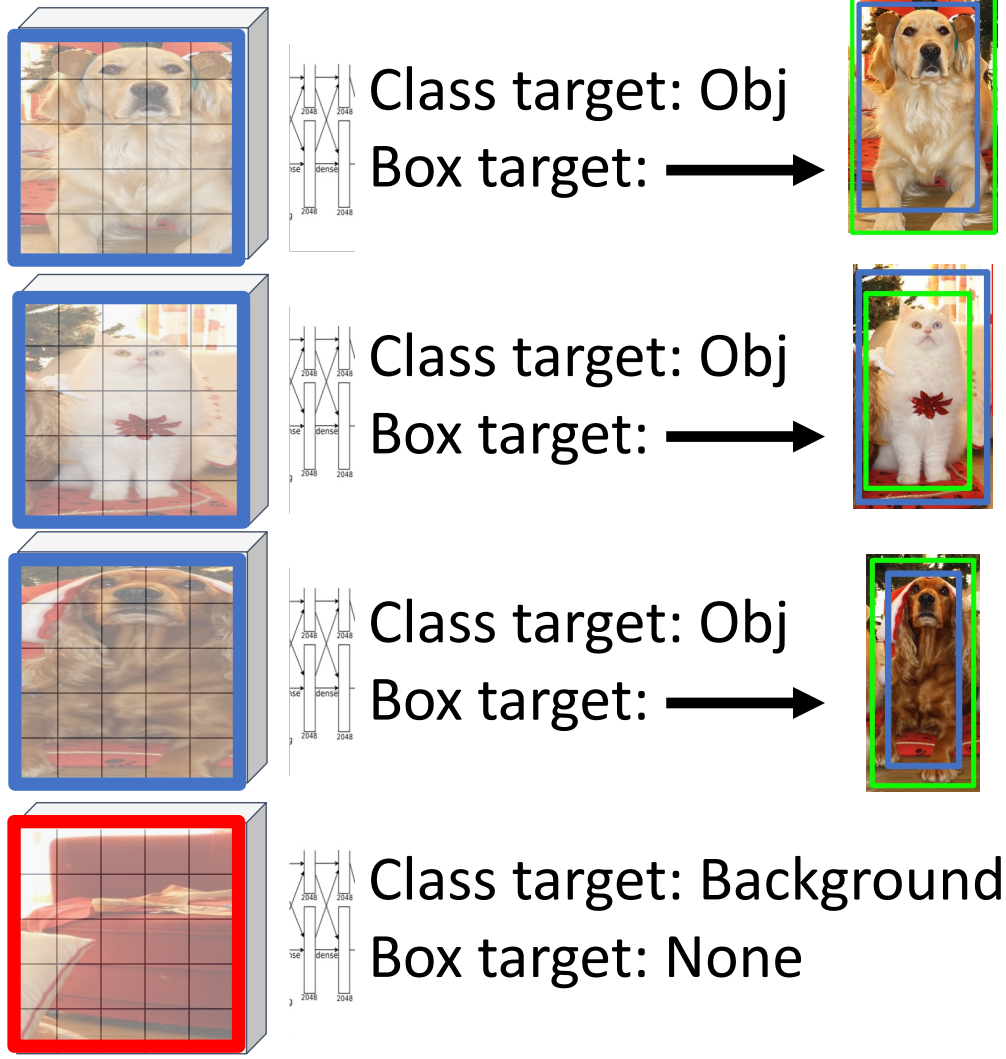


Image Features



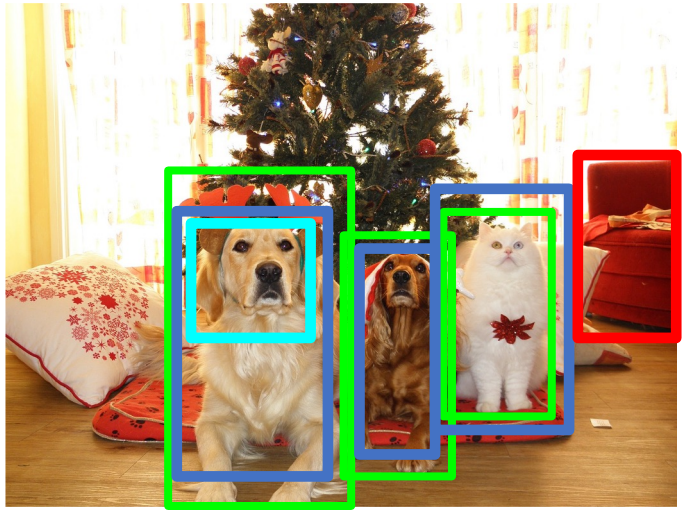
RPN gives lots of **anchors** which we classify as pos / neg / neutral by matching with ground-truth



Faster R-CNN Training: Stage 2

Crop features for each proposal, use them to predict class and box targets per region

Input Image



Backbone CNN

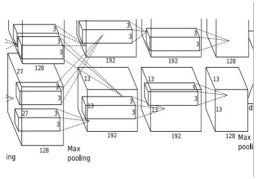
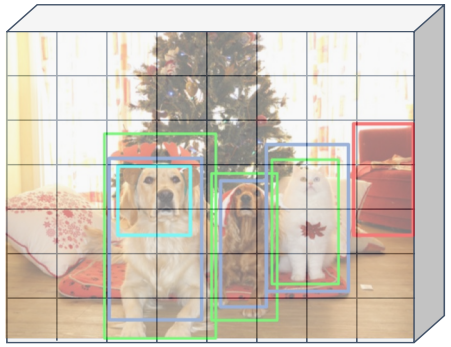
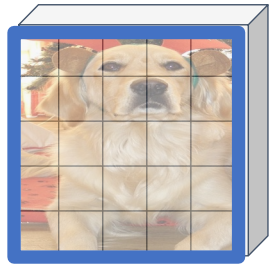
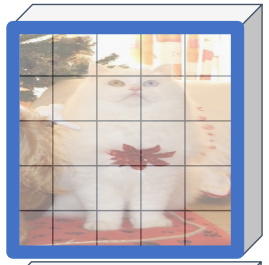


Image Features

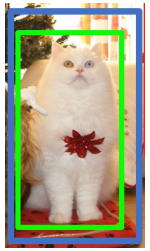


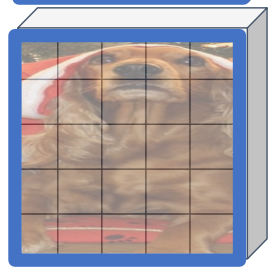


Class target: Dog
Box target:

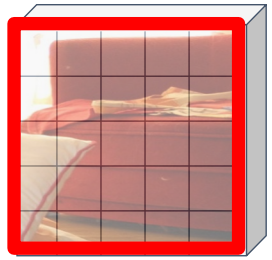


Class target: Cat
Box target:





Class target: Dog
Box target:



Class target: Background
Box target: None

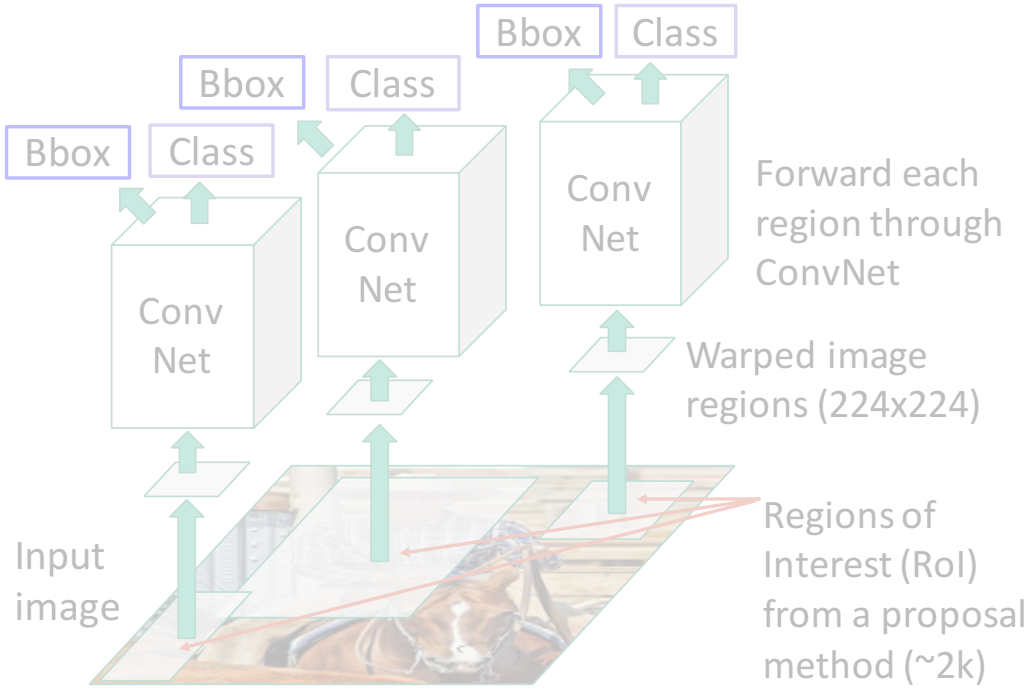
GT Boxes | Positive

Neutral | Negative

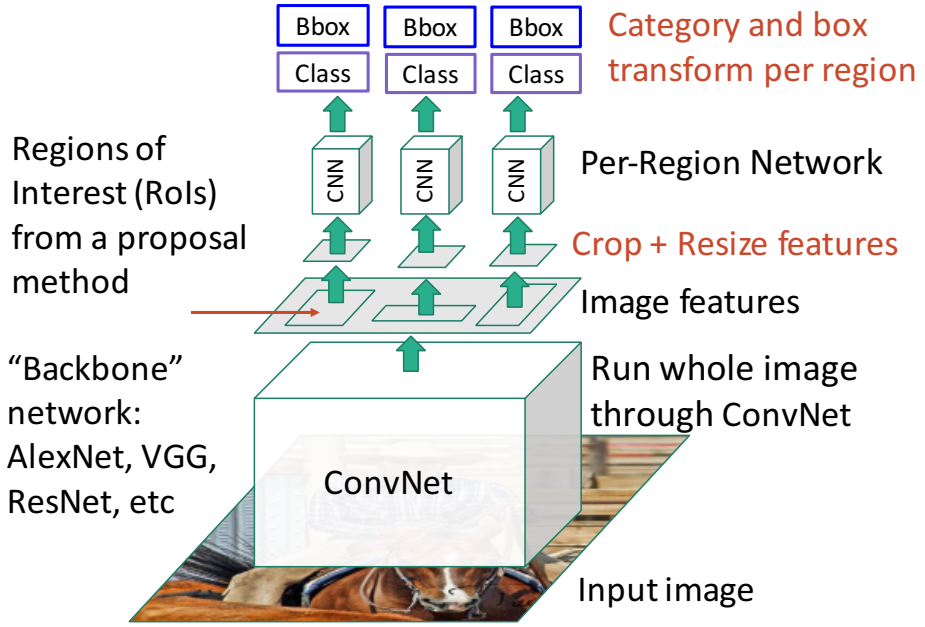
Now proposals come from RPN rather than selective search, but otherwise this works the same as Fast R-CNN training

Recap: Fast R-CNN Feature Cropping

"Slow" R-CNN: Run CNN independently for each region

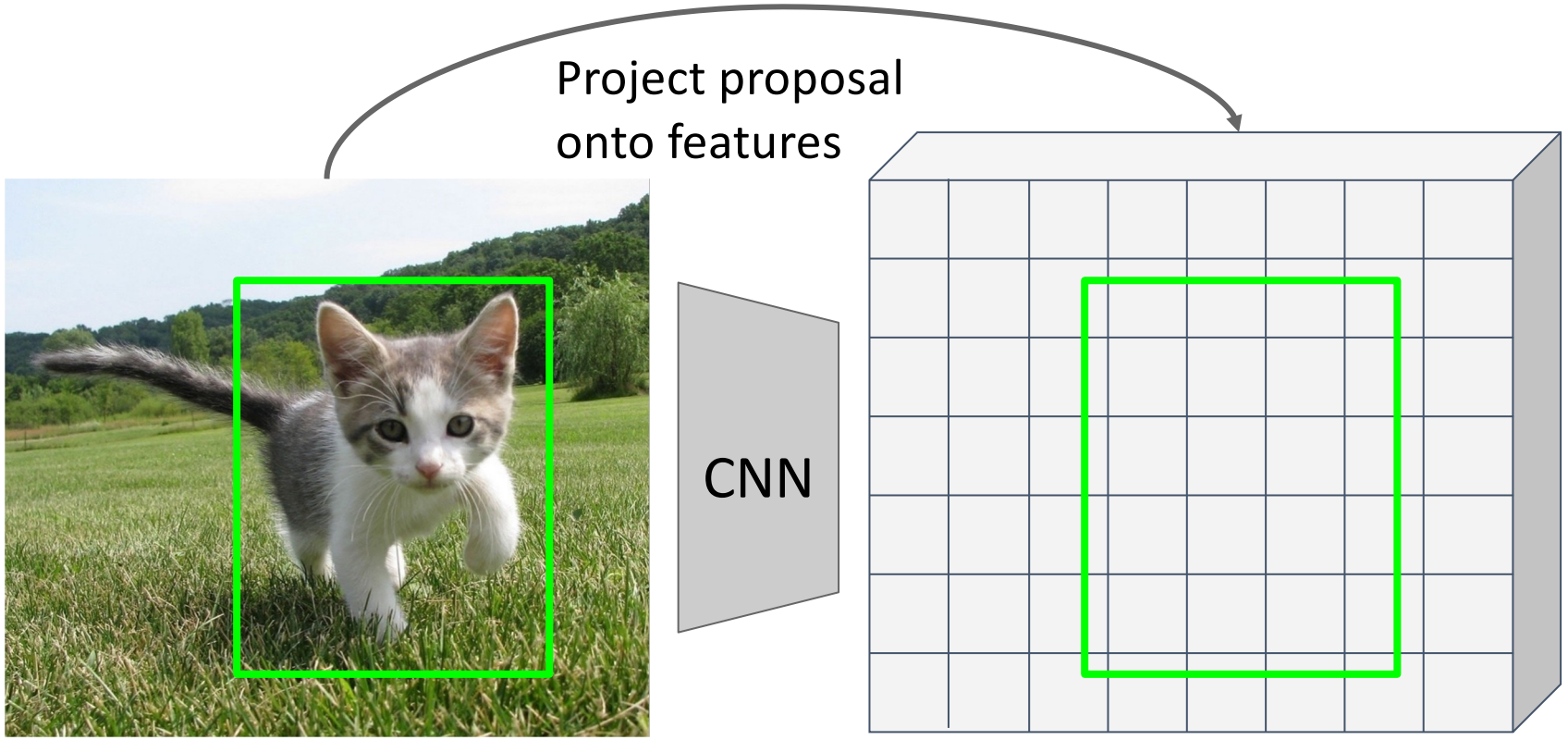


Fast R-CNN: Apply differentiable cropping to shared image features



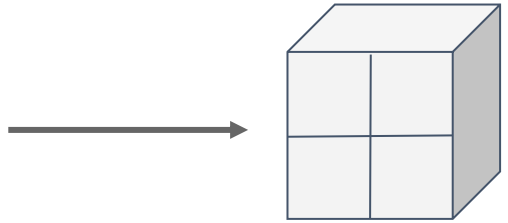
Cropping Features

Goal: Crop features for region proposal, and resize to a fixed size for downstream processing, in a differentiable way



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

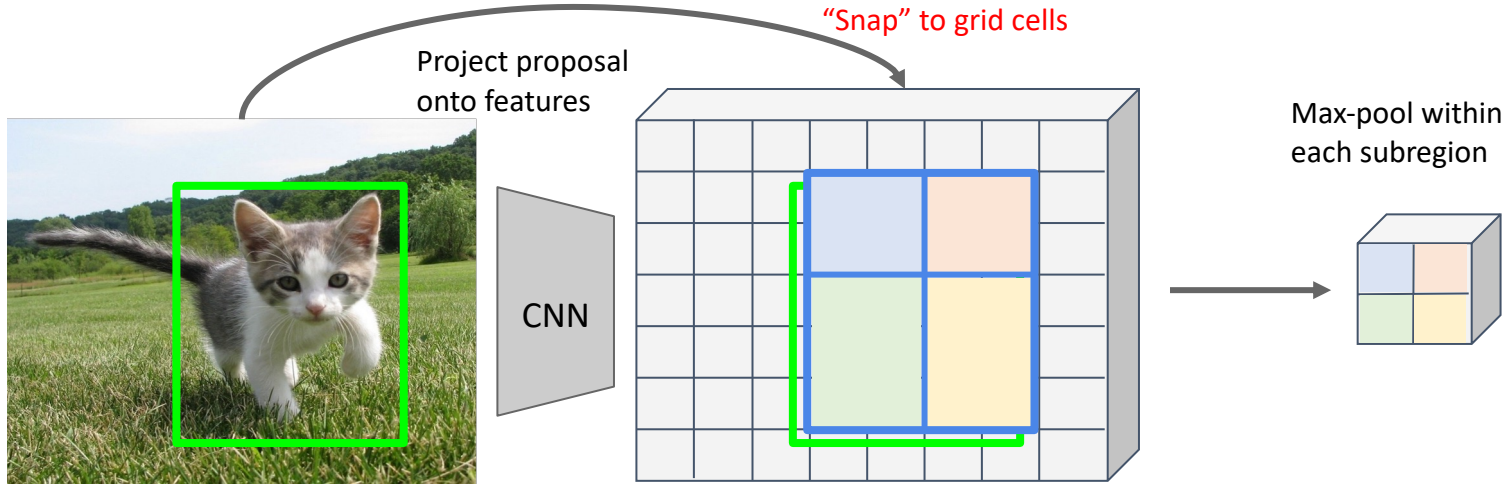


Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

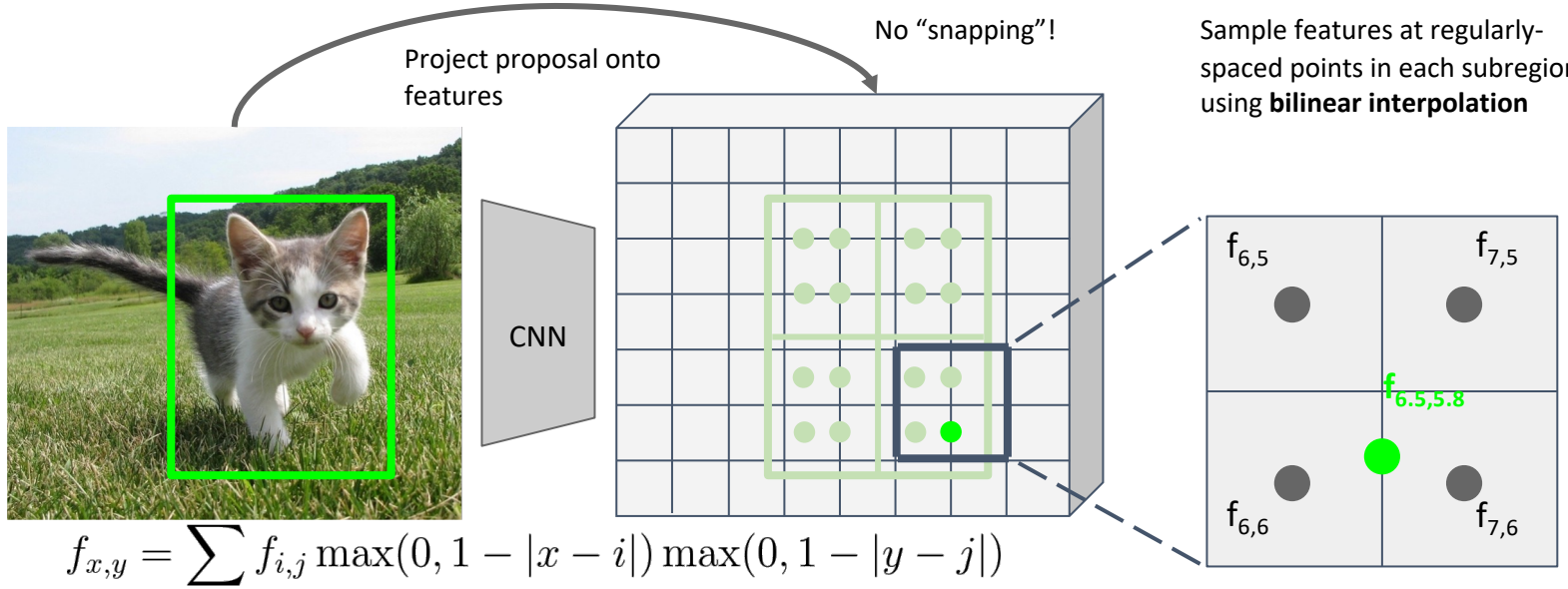
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features

RoI Pool



RoI Align



$$f_{x,y} = \sum f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$$

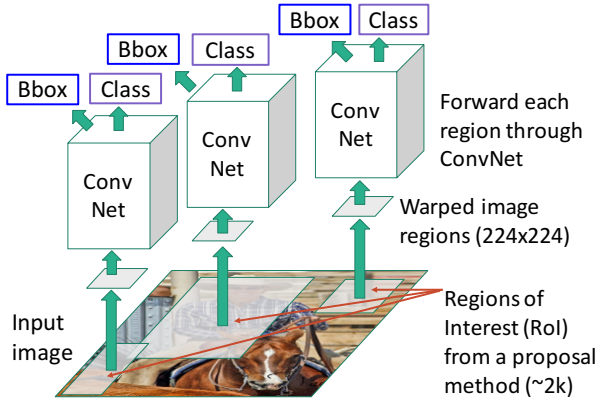
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)

Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

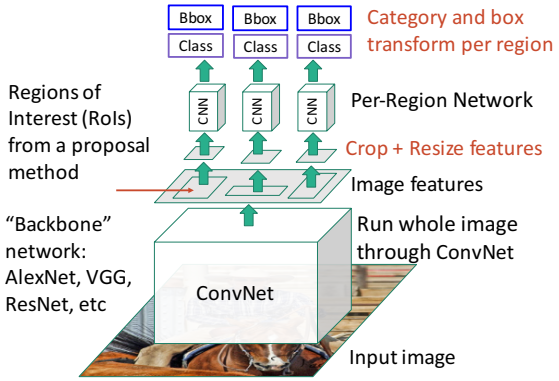
Object Detection Methods

It relies on anchor boxes. Can we do detection without anchors?

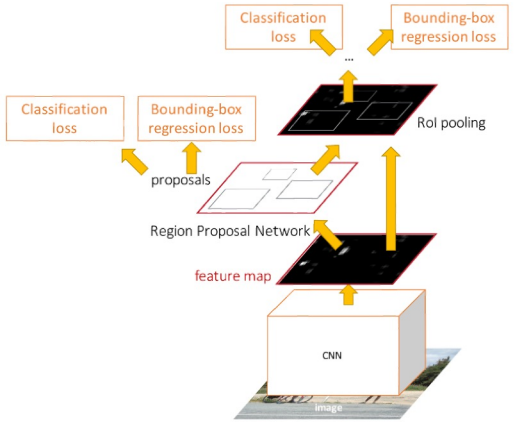
"Slow" R-CNN: Run CNN independently for each region



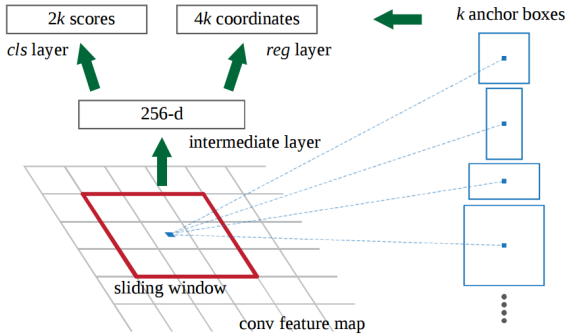
Fast R-CNN: Apply differentiable cropping to shared image features



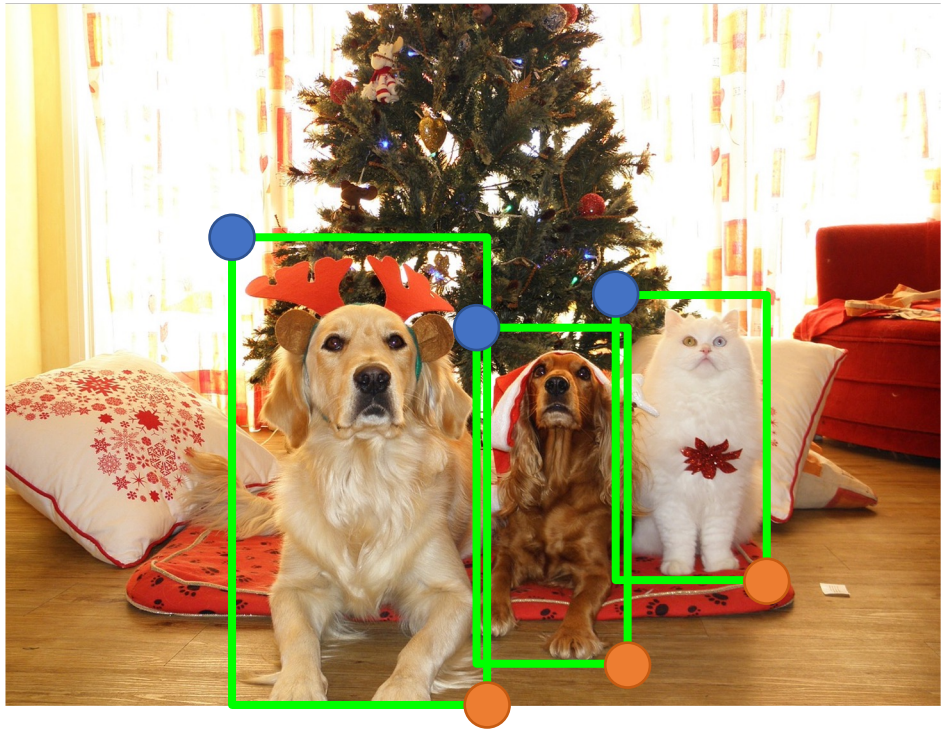
Faster R-CNN: Compute proposals with CNN



Single-Stage (SSD, YOLO): Fully convolutional detector

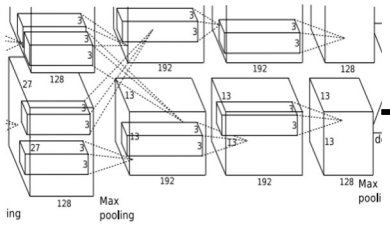


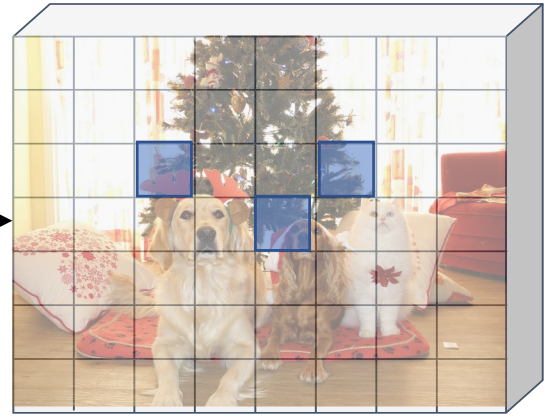
Detection without Anchors (anchor-free methods): CornerNet



Represent bounding boxes by pairs of corners

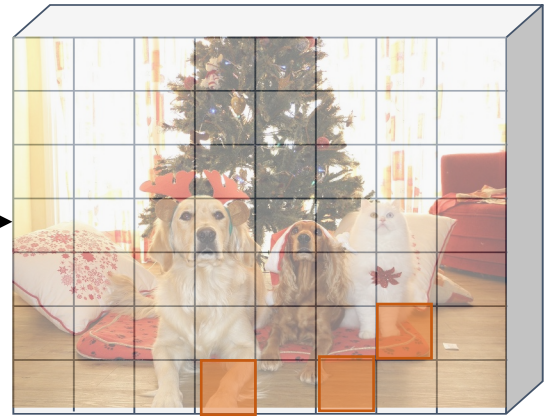
Backbone CNN





Upper left corners
Heatmap: C x H x W
Embeddings: D x H x W

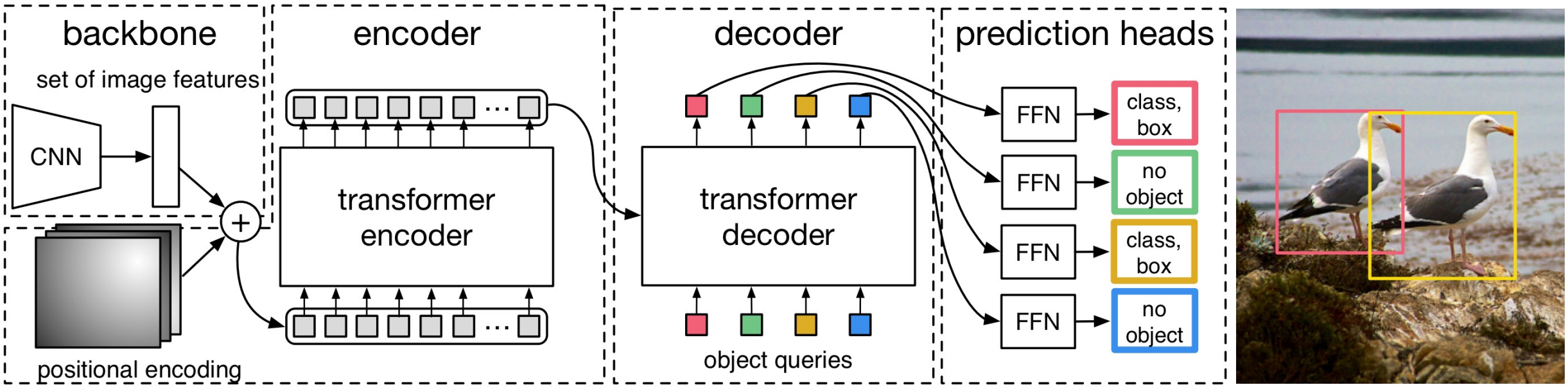
Matching corners are trained to have similar embeddings



Lower right corners
Heatmap: C x H x W
Embeddings: D x H x W

Law and Deng, “CornerNet: Detecting Objects as Paired Keypoints”, ECCV 2018

Detection without Anchors (anchor-free methods): Transformer-based DETR



Carion et al, "DETR: End-to-End Object Detection with Transformers", ECCV 2020

Computer Vision Tasks: Object Detection

Classification

CAT

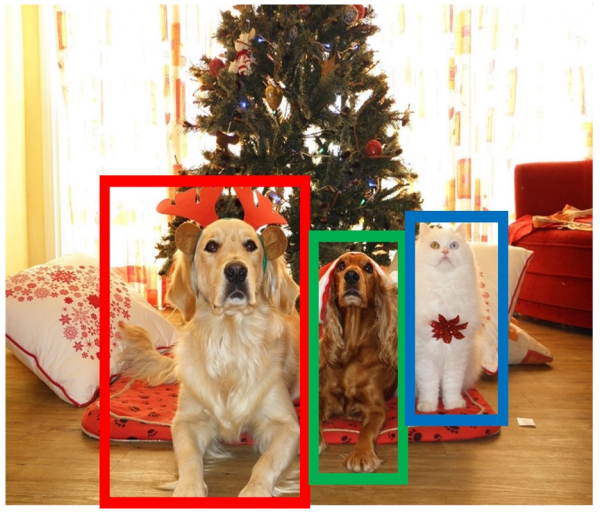
No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Objects

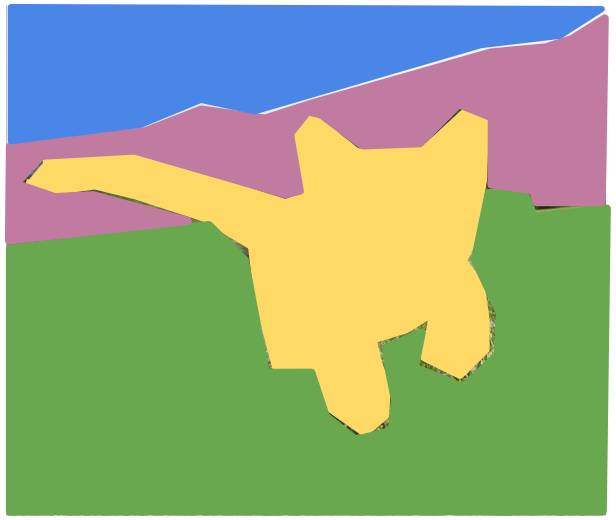
Computer Vision Tasks: Semantic Segmentation

Classification

CAT

No spatial extent

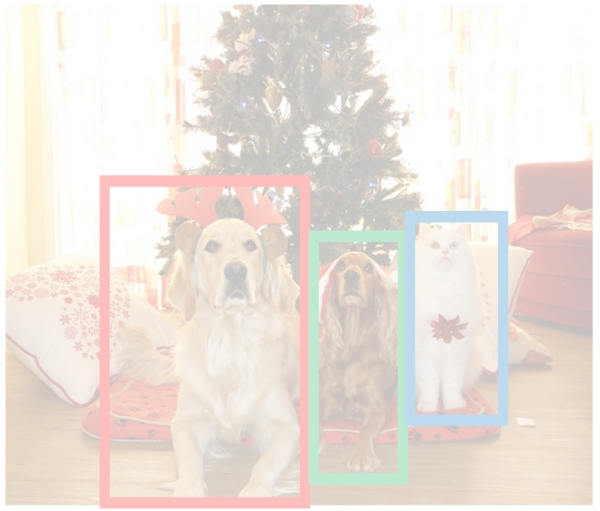
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

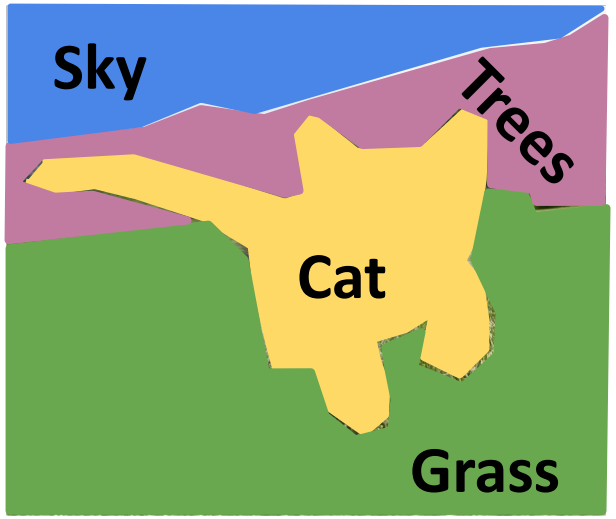
DOG, DOG, CAT

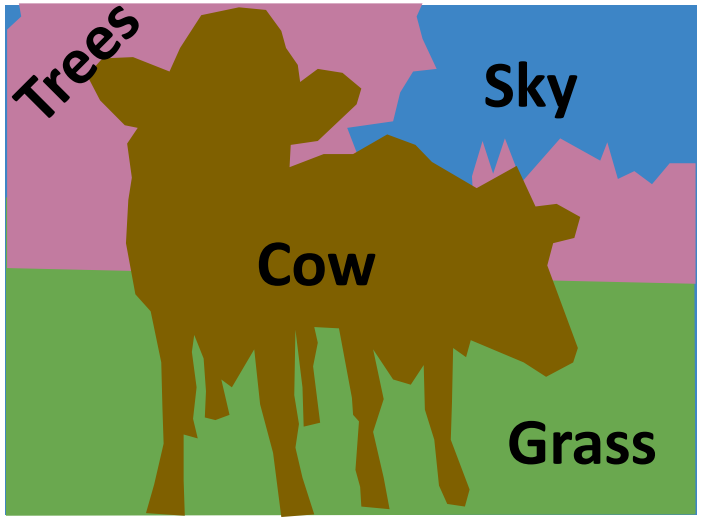
Multiple Objects

Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels

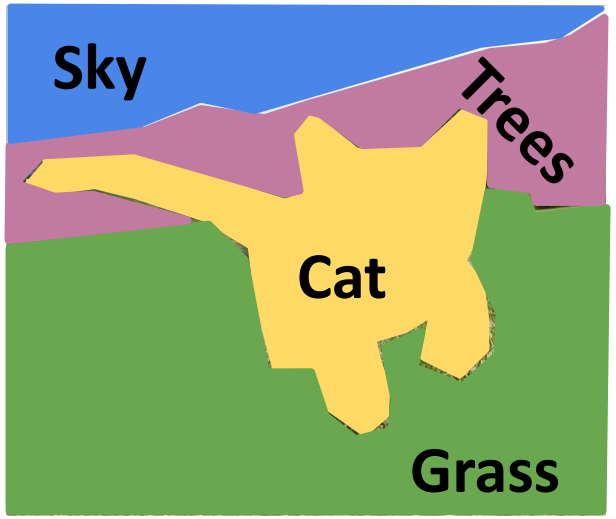


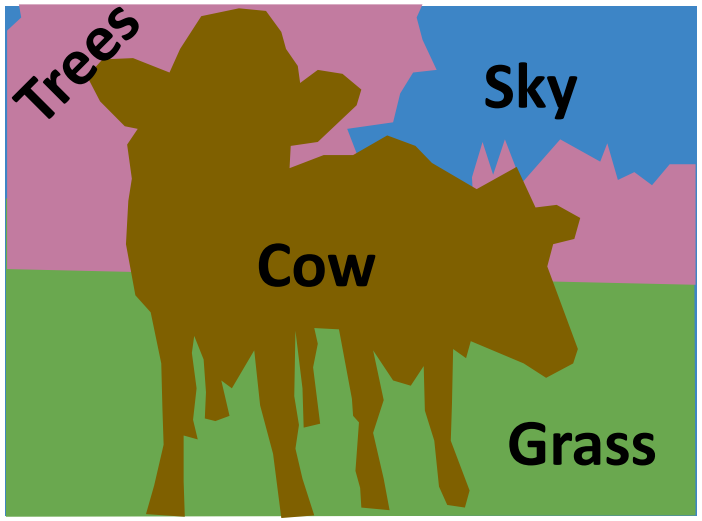


Things and Stuff

Things: Object categories that can be separated into object instances
(e.g. cats, cars, person)

Stuff: Object categories that cannot be separated into instances
(e.g. sky, grass, water, trees)

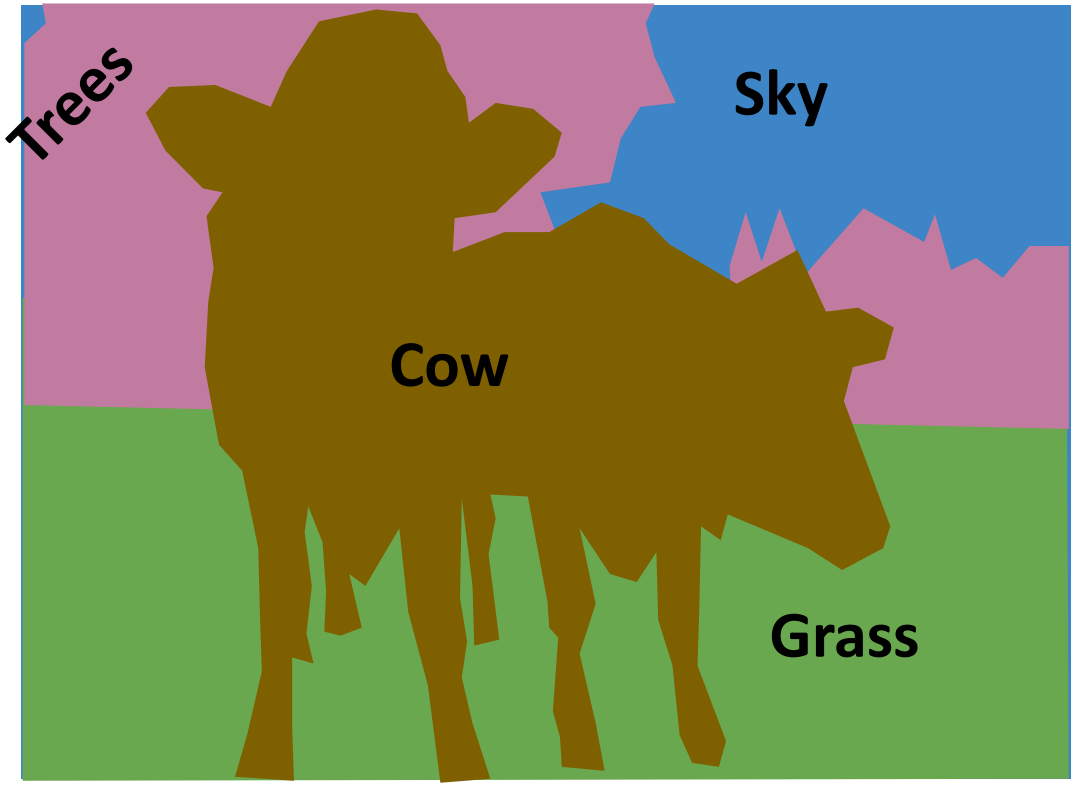




Object Detection vs Semantic Segmentation

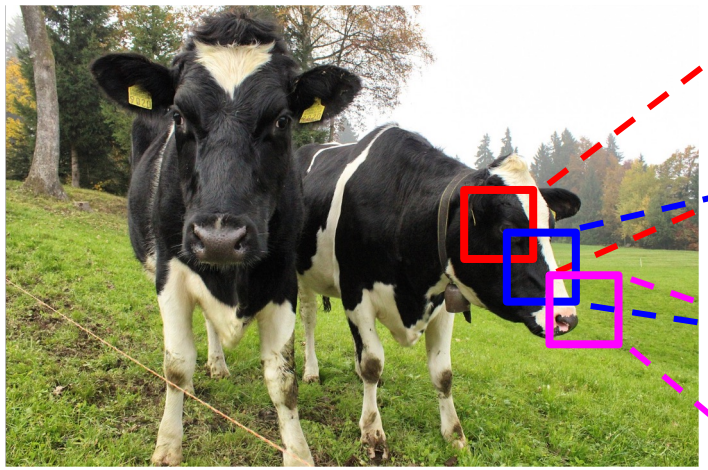
Object Detection: Detects individual object instances, but only gives box
(Only things!)

Semantic Segmentation: Gives per-pixel labels, but merges instances (Both things and stuff) where each pixel has a label

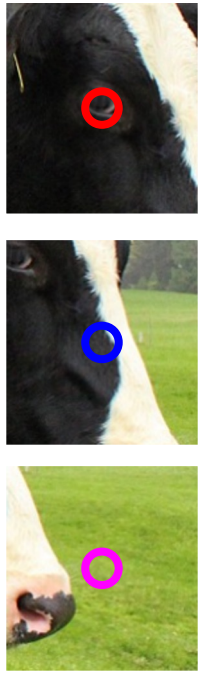


Semantic Segmentation Idea: Sliding Window

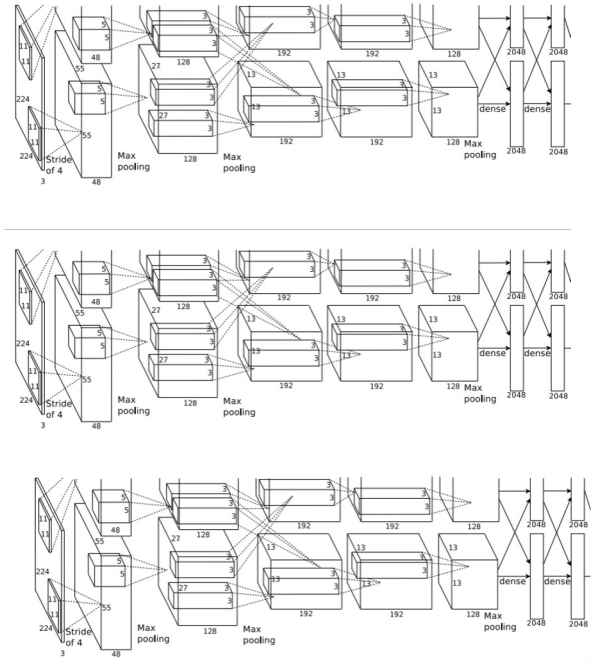
Full image



Extract patch



Classify center pixel with CNN



Cow

Cow

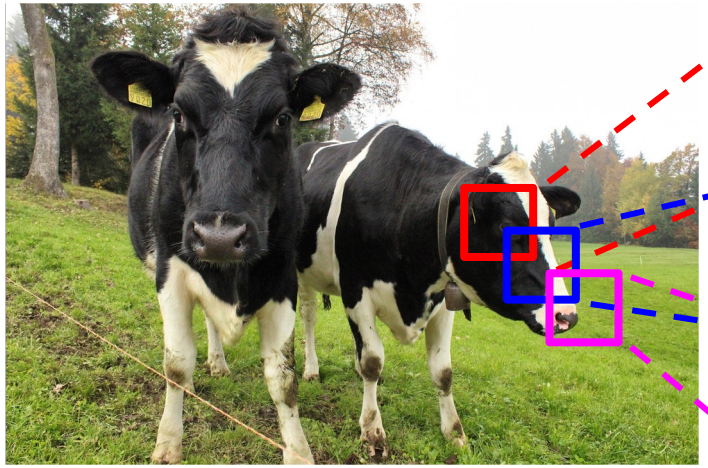
Grass

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

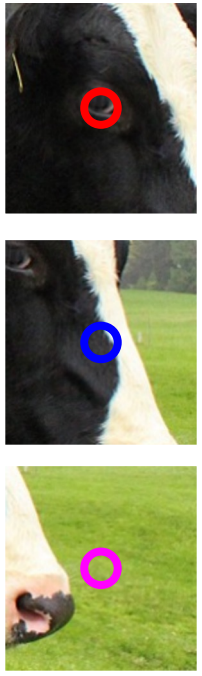
Semantic Segmentation Idea: Sliding Window

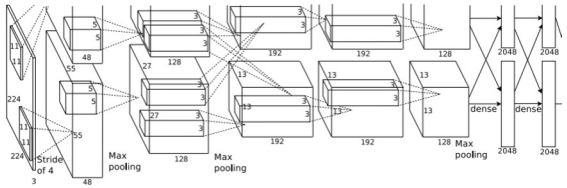
Full image



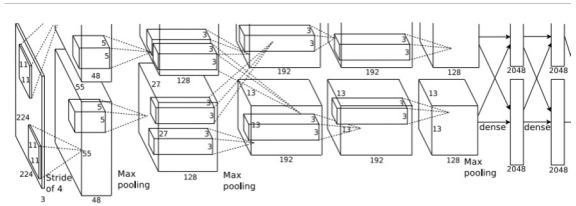
Extract patch

Classify center pixel with CNN

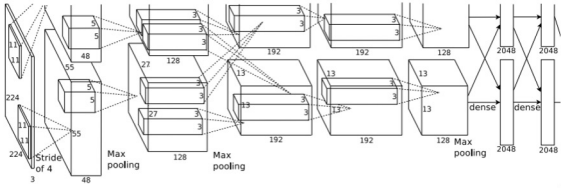




Cow



Cow



Grass

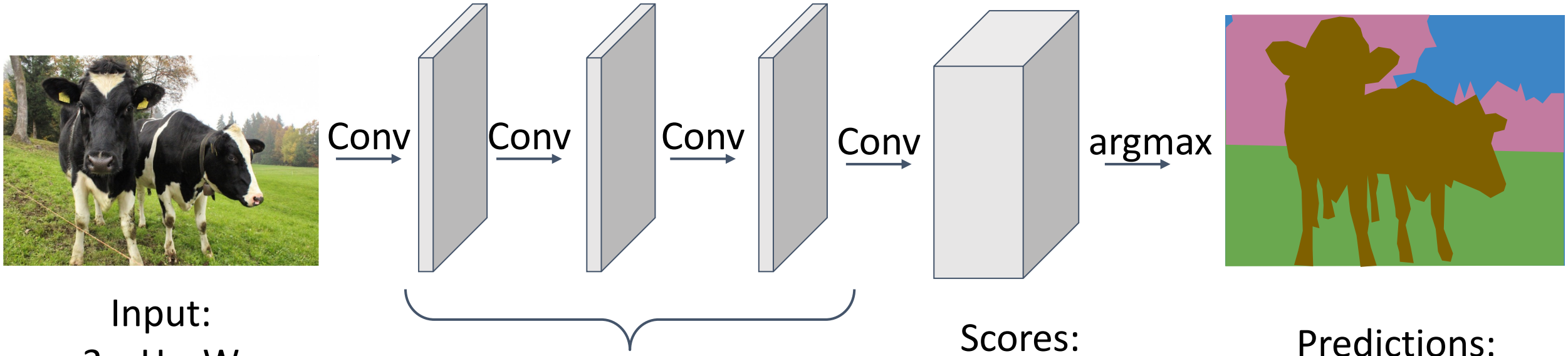
Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W

Convolutions:
D x H x W

Scores:
C x H x W

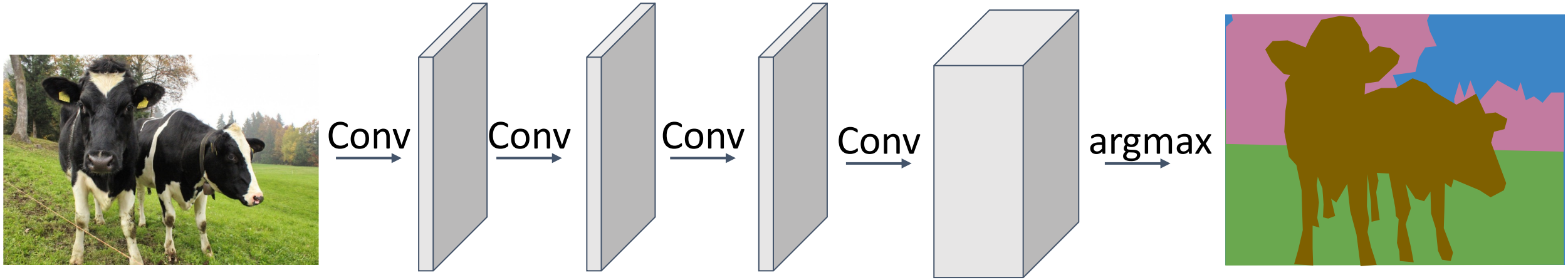
Predictions:
H x W

Loss function: Per-Pixel cross-entropy

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



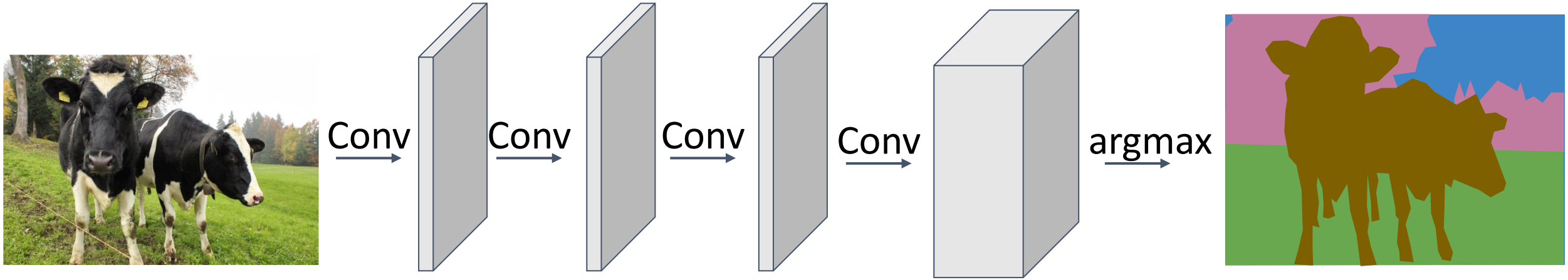
Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

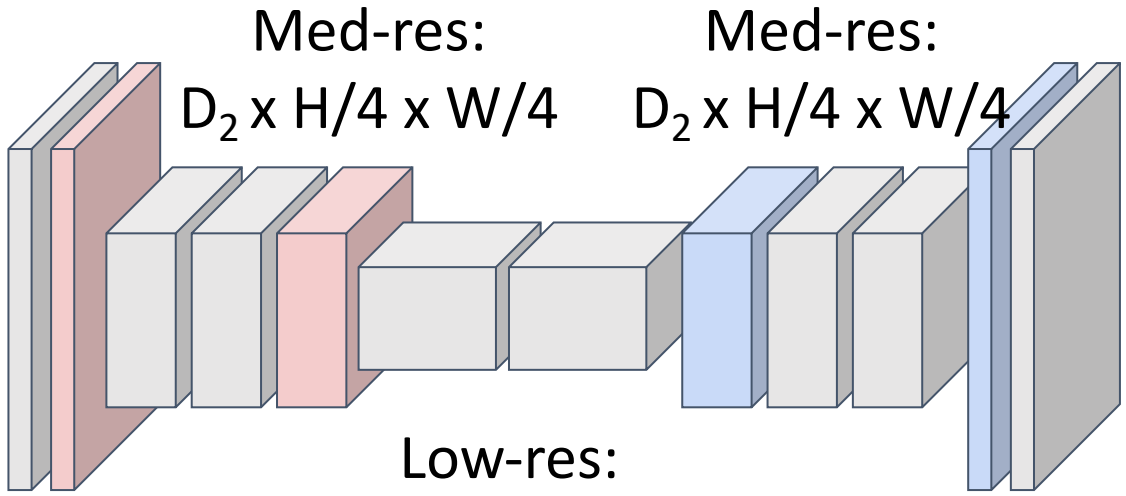
Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



Predictions:
H x W

Or it is also called Encoder-Decoder Architecture

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

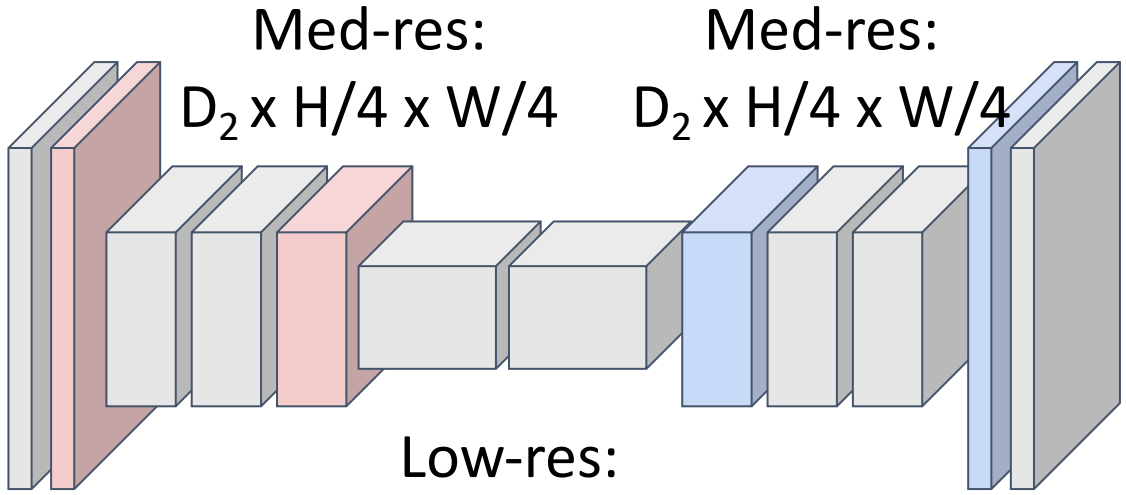
Semantic Segmentation: Fully Convolutional Network

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: ???

Input:
3 x H x W



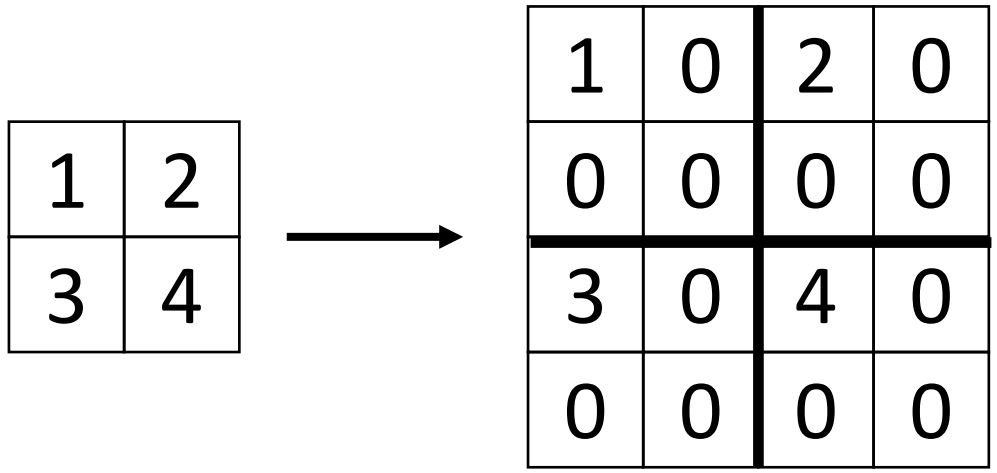
Predictions:
H x W

Or it is also called Encoder-Decoder Architecture

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network Upsampling: “Unpooling”

Simple

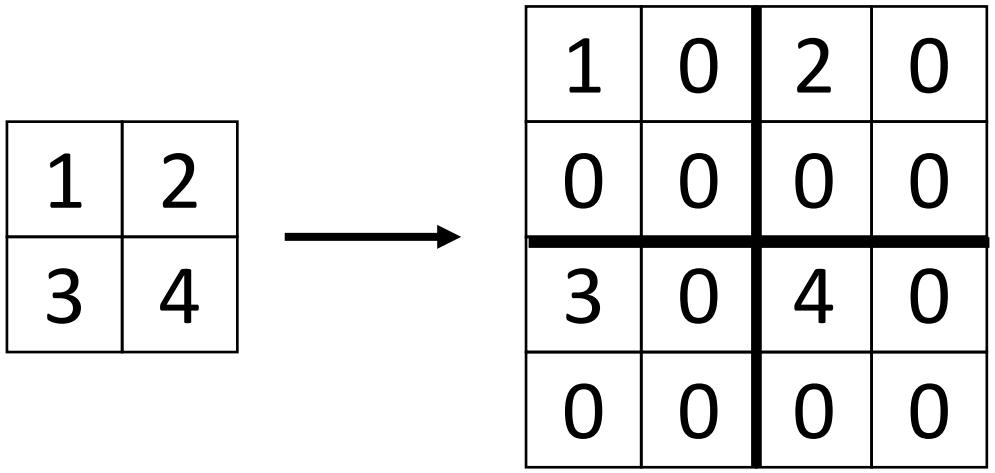


Input
C x 2 x 2

Output
C x 4 x 4

In-Network Upsampling: “Unpooling”

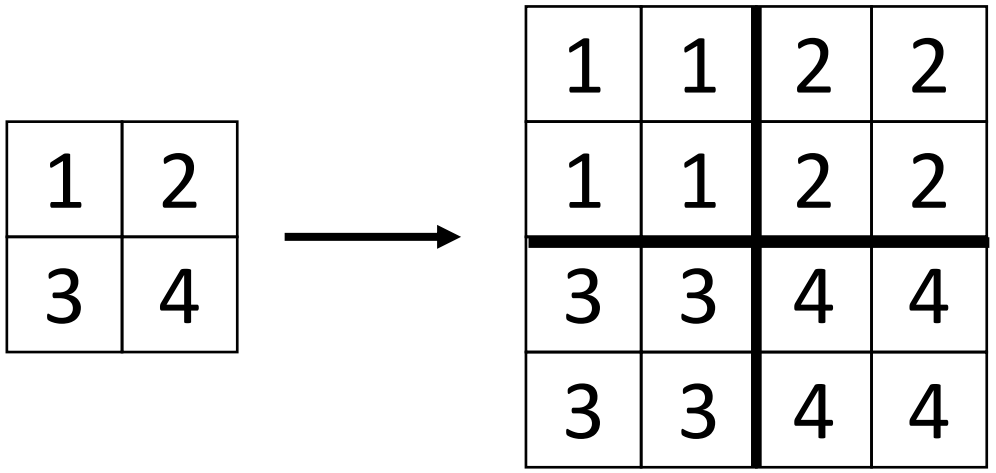
Simple



Input
C x 2 x 2

Output
C x 4 x 4

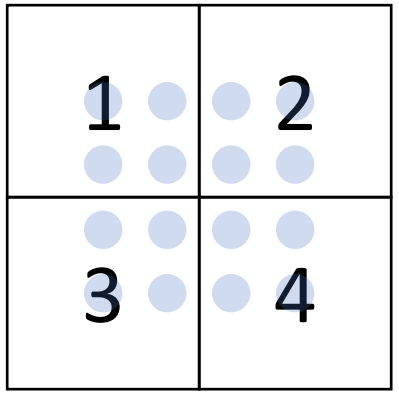
Nearest Neighbor



Input
C x 2 x 2

Output
C x 4 x 4

In-Network Upsampling: Bilinear Interpolation



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: C x 2 x 2

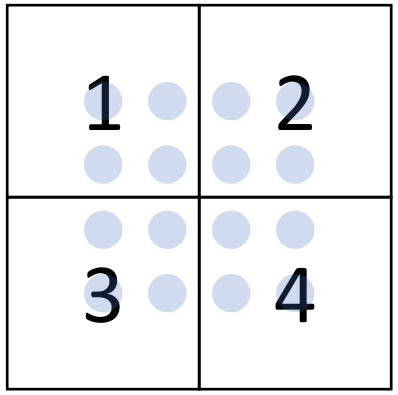
Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$$

$$i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\}$$
$$j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

In-Network Upsampling: Bicubic Interpolation



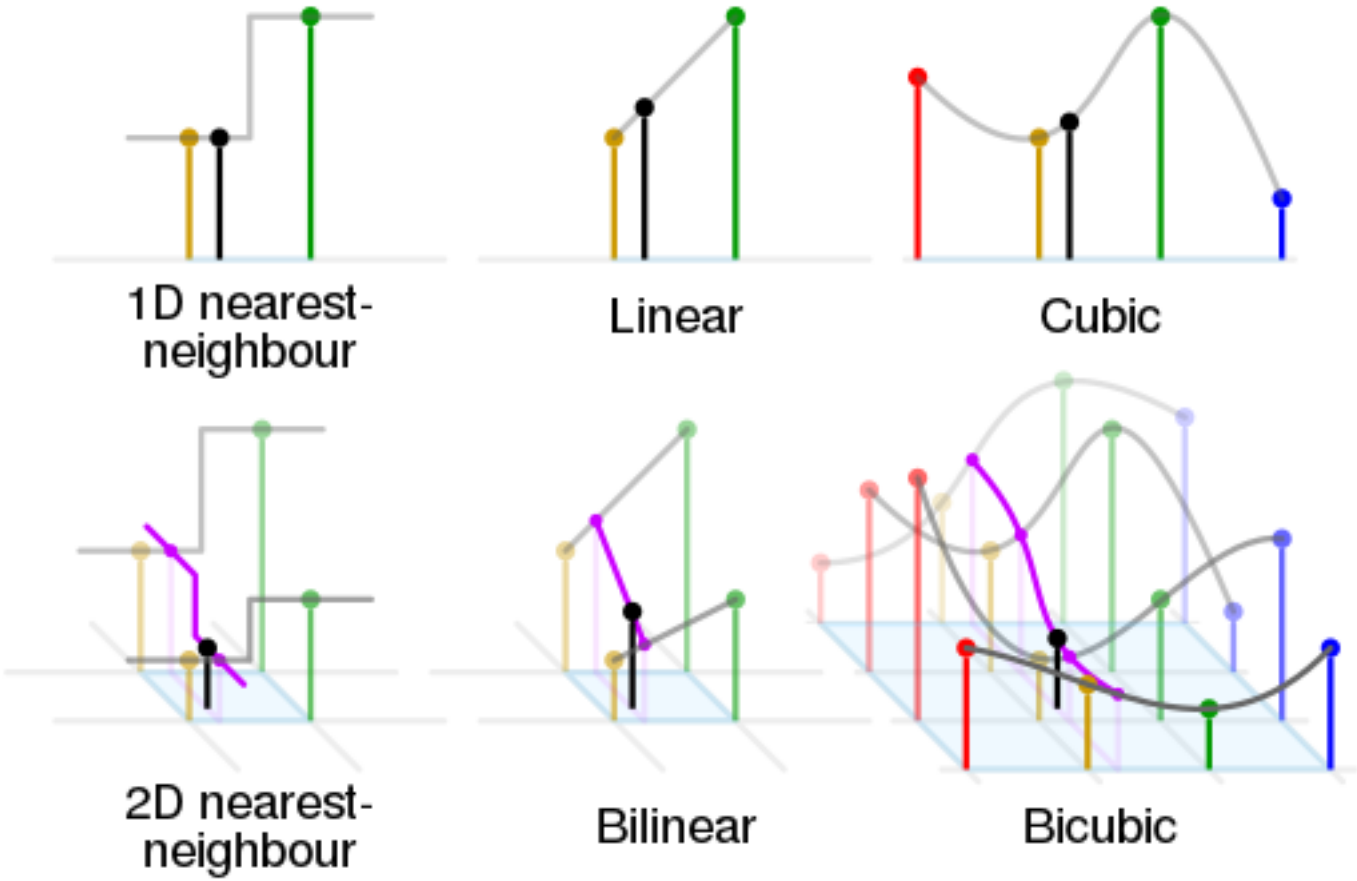
0.68	1.02	1.56	1.89
1.35	1.68	2.23	2.56
2.44	2.77	3.32	3.65
3.11	3.44	3.98	4.32

Input: C x 2 x 2

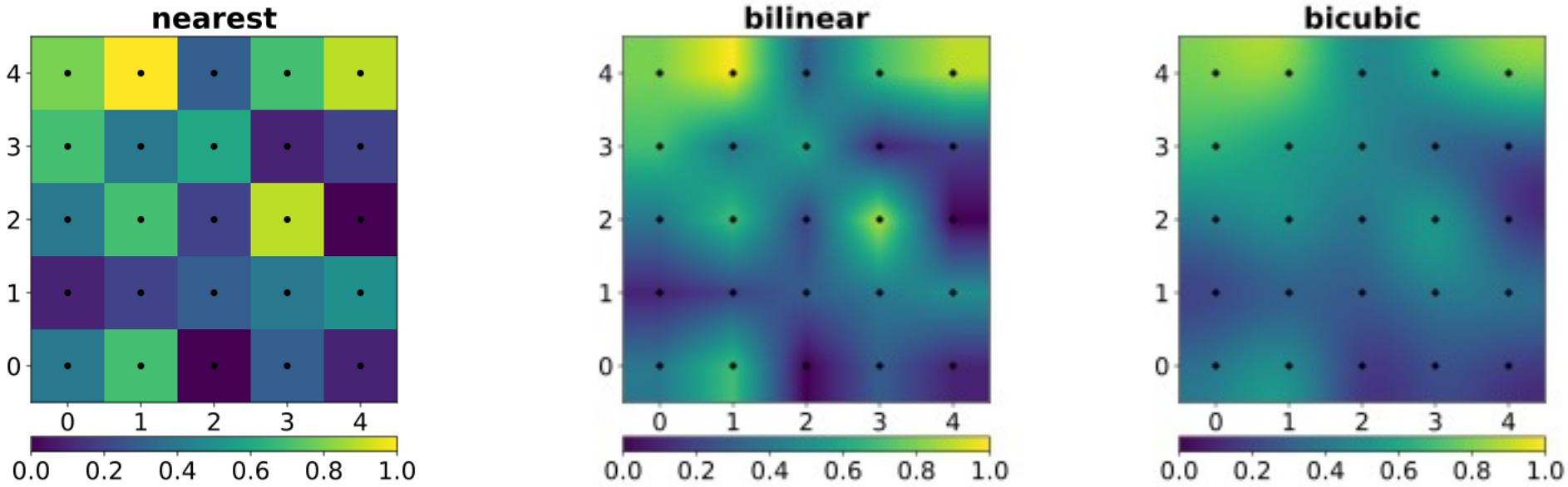
Output: C x 4 x 4

Use **three** closest neighbors in x and y to construct **cubic** approximations
(This is how we normally resize images!)

Different interpolation methods



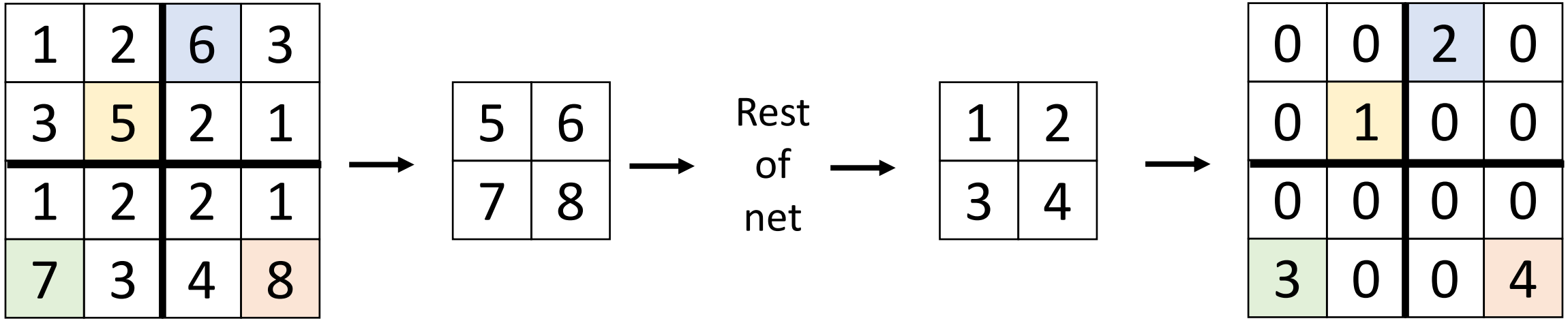
https://en.wikipedia.org/wiki/Bicubic_interpolation

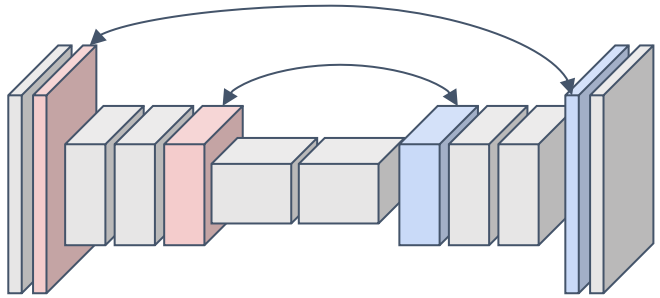

nearest
bilinear
bicubic
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0

In-Network Upsampling: “Max Unpooling”

Max Pooling: Remember which position had the max

Max Unpooling: Place into remembered positions



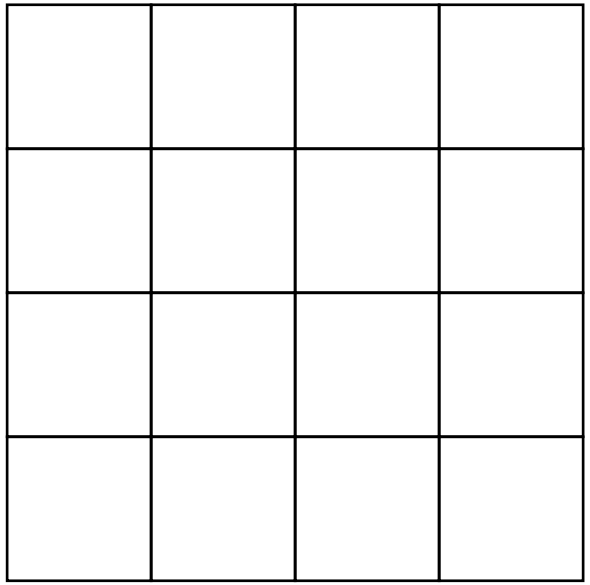


Pair each downsampling layer with an upsampling layer

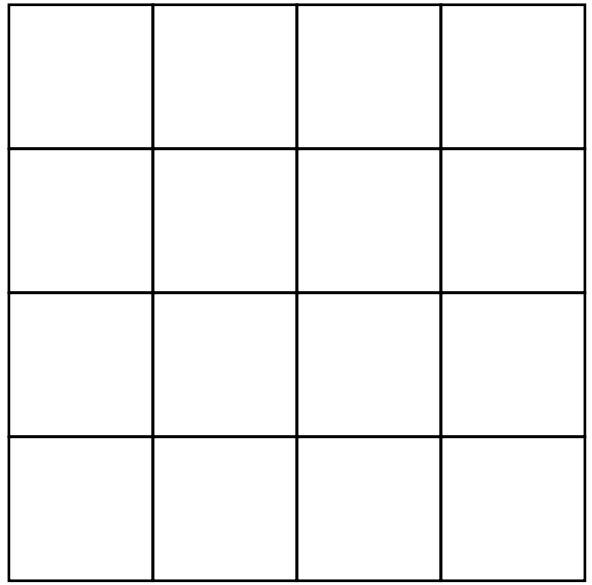
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1



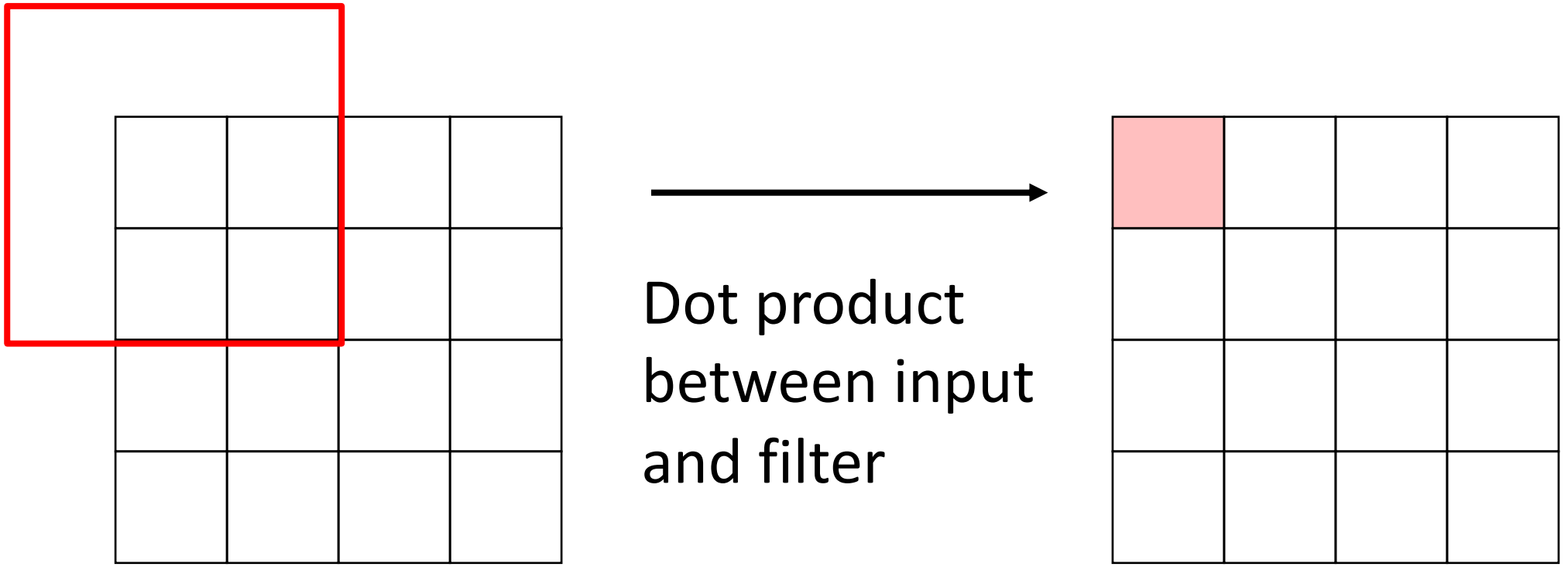
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1

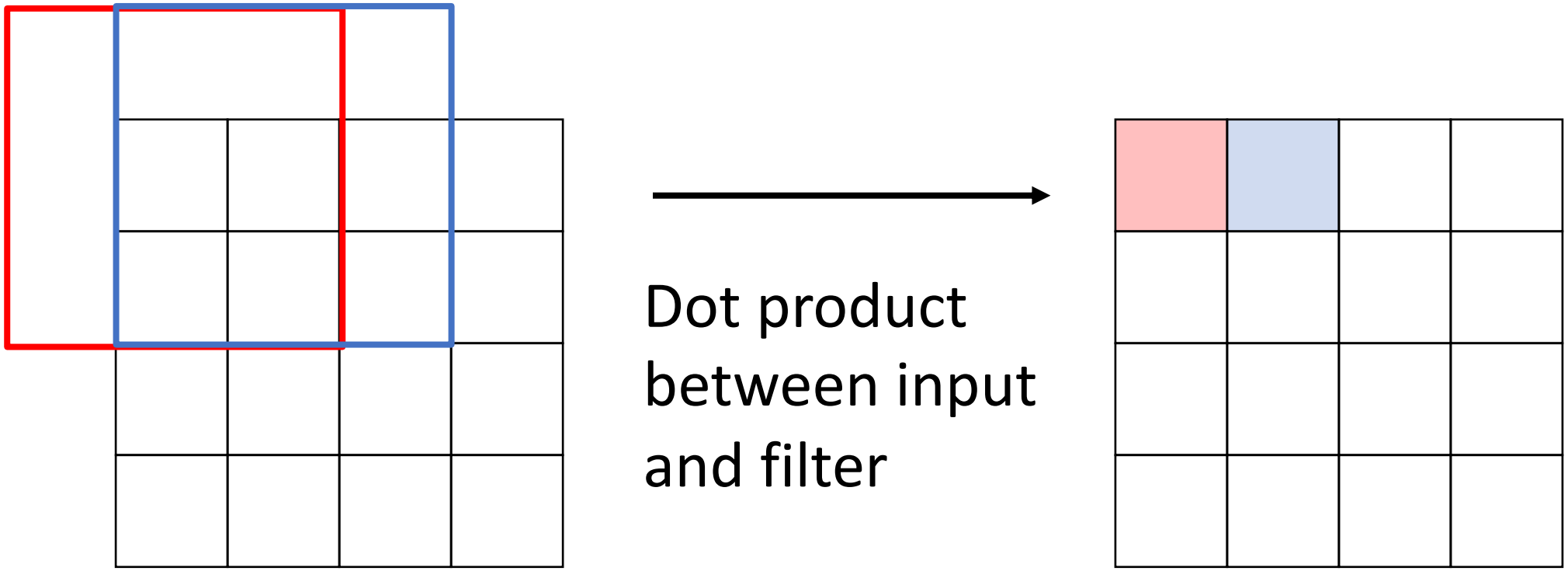


Input: 4 x 4

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1



Input: 4 x 4

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

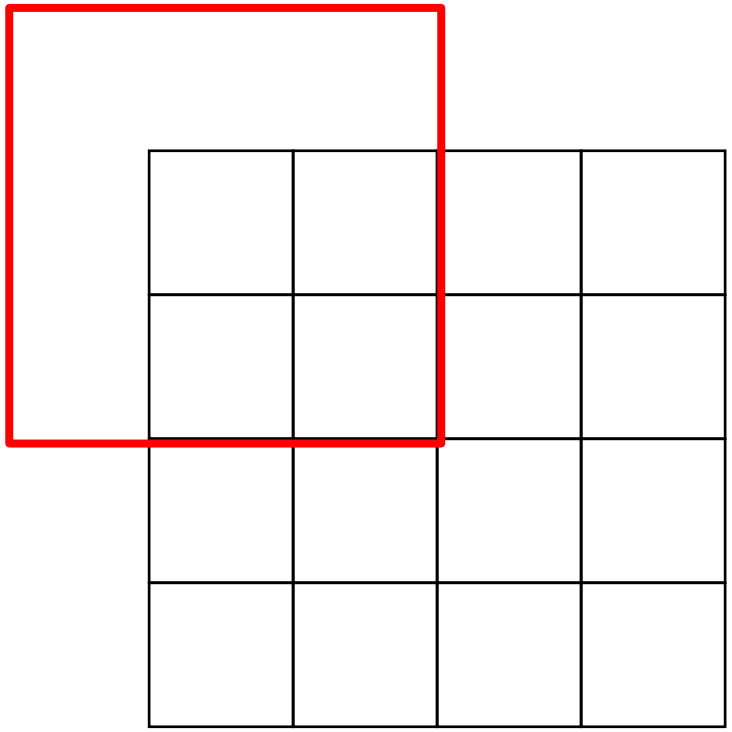
Recall: Normal 3 x 3 convolution, stride 2, pad 1

Input: 4 x 4

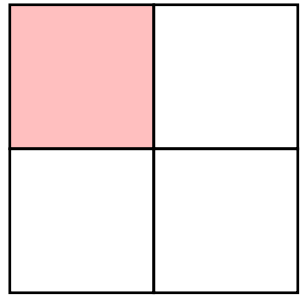
Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Dot product between input and filter

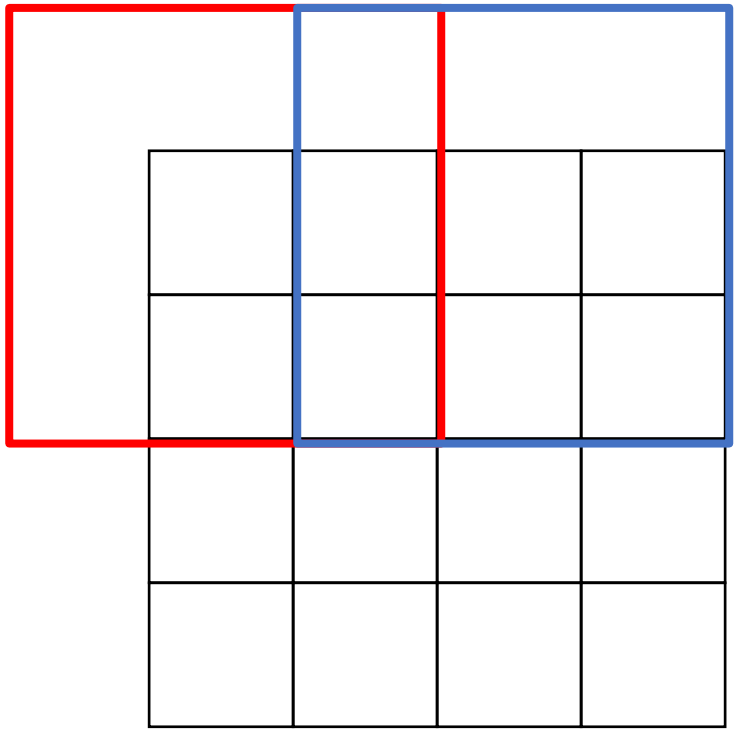


Input: 4 x 4

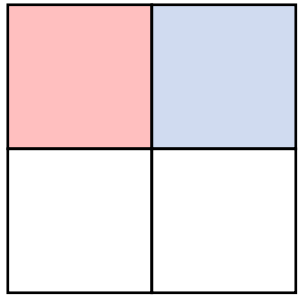
Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Dot product between input and filter

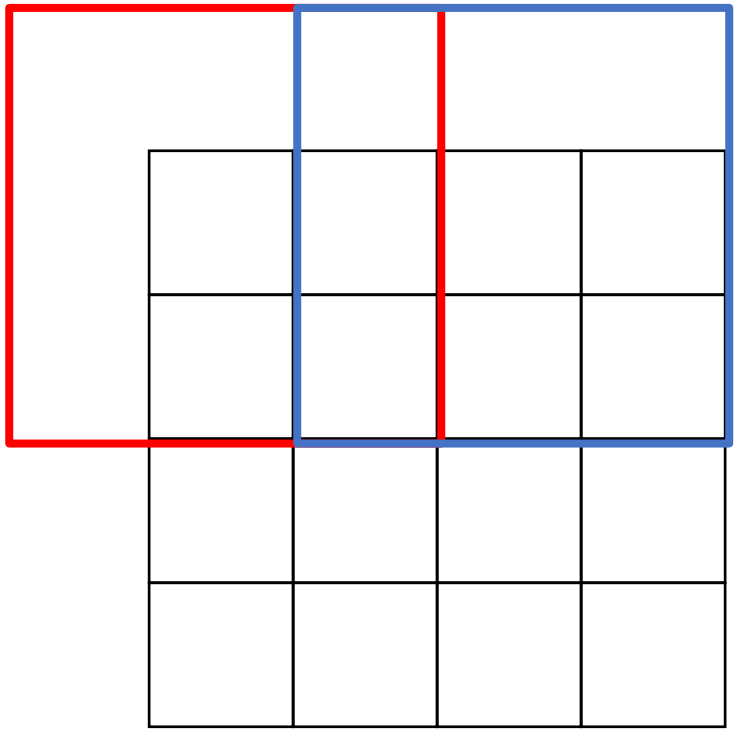


Input: 4 x 4

Output: 2 x 2

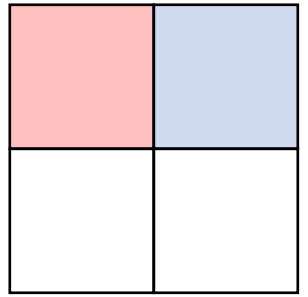
Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Convolution with stride > 1 is “Learnable Downsampling”
How about “Learnable Upsampling”?

Dot product between input and filter

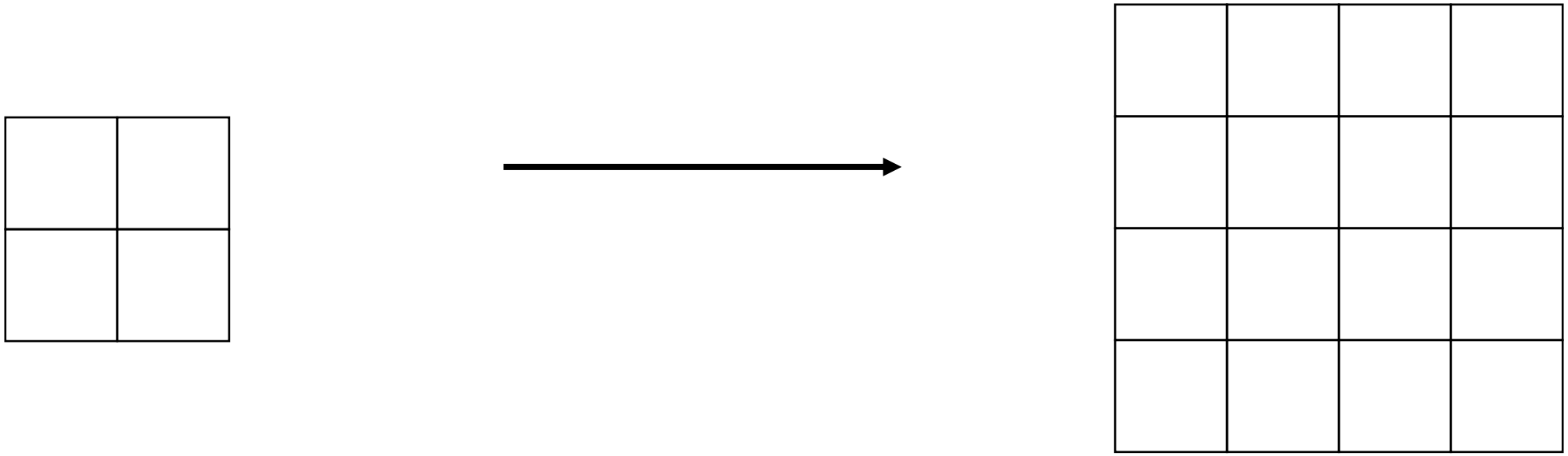


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transposed Convolution

3 x 3 **convolution transpose**, stride 2

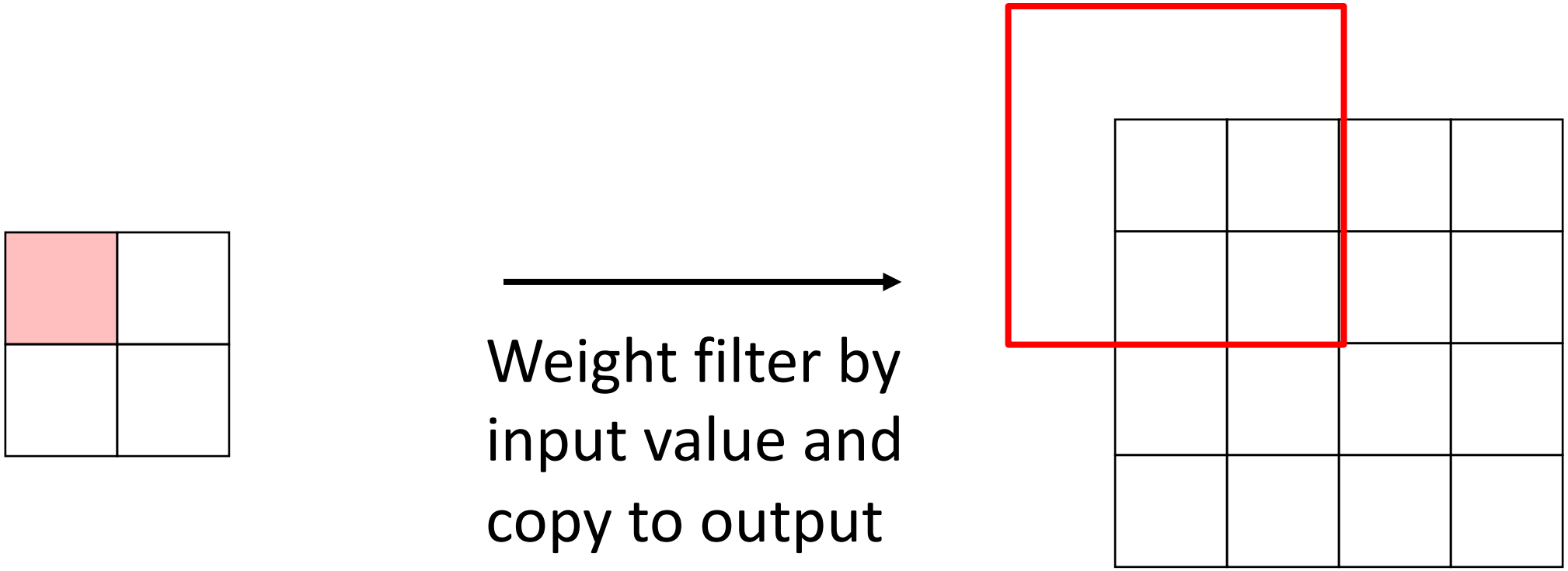


Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

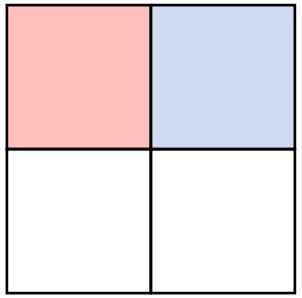
3 x 3 **convolution transpose**, stride 2



Learnable Upsampling: Transposed Convolution

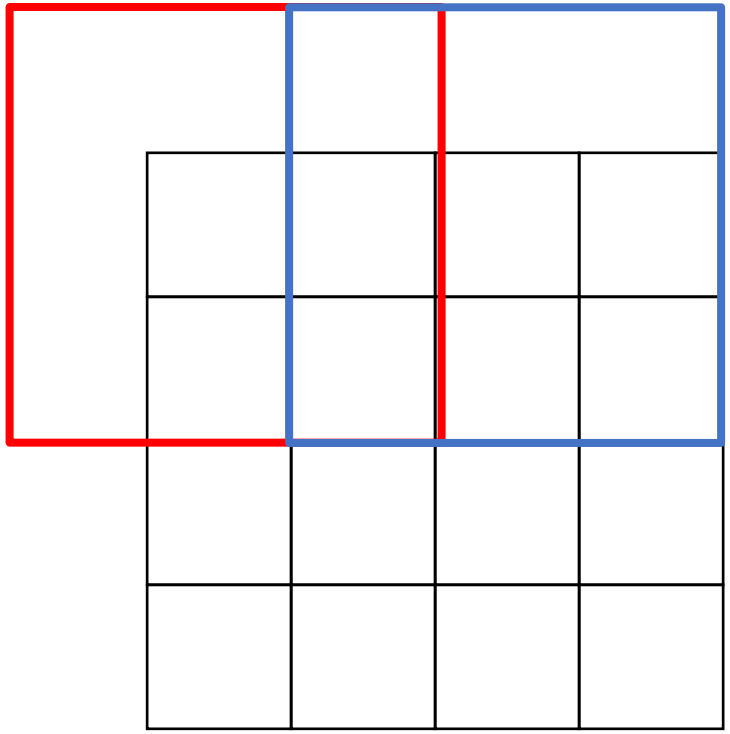
3 x 3 **convolution transpose**, stride 2

Filter moves 2 pixels in output
for every 1 pixel in input



Input: 2 x 2

Weight filter by
input value and
copy to output

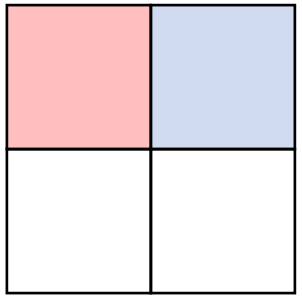


Output: 4 x 4

Learnable Upsampling: Transposed Convolution

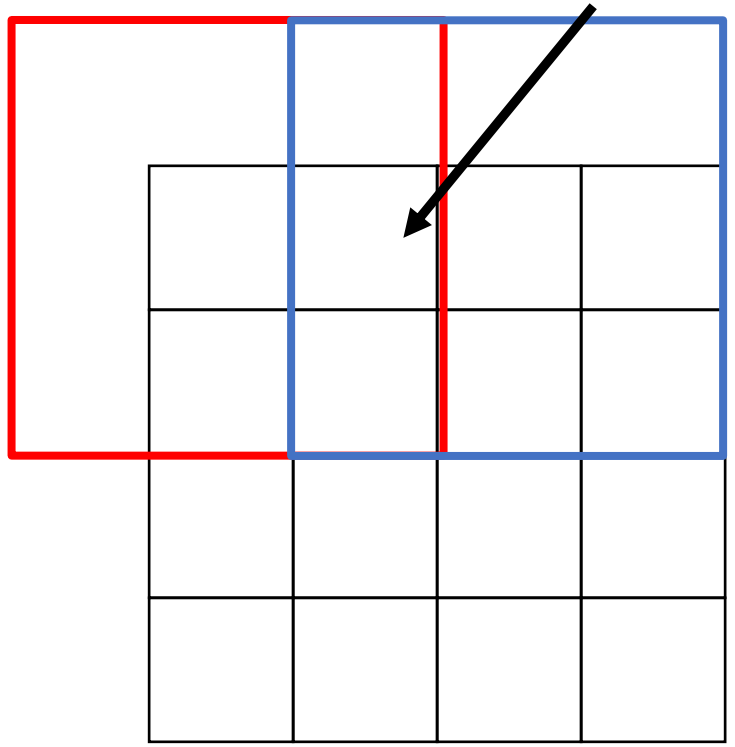
3 x 3 **convolution transpose**, stride 2

Filter moves 2 pixels in output
for every 1 pixel in input



Input: 2 x 2

Weight filter by input value and copy to output



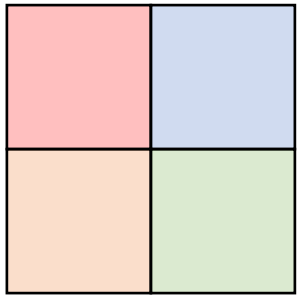
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

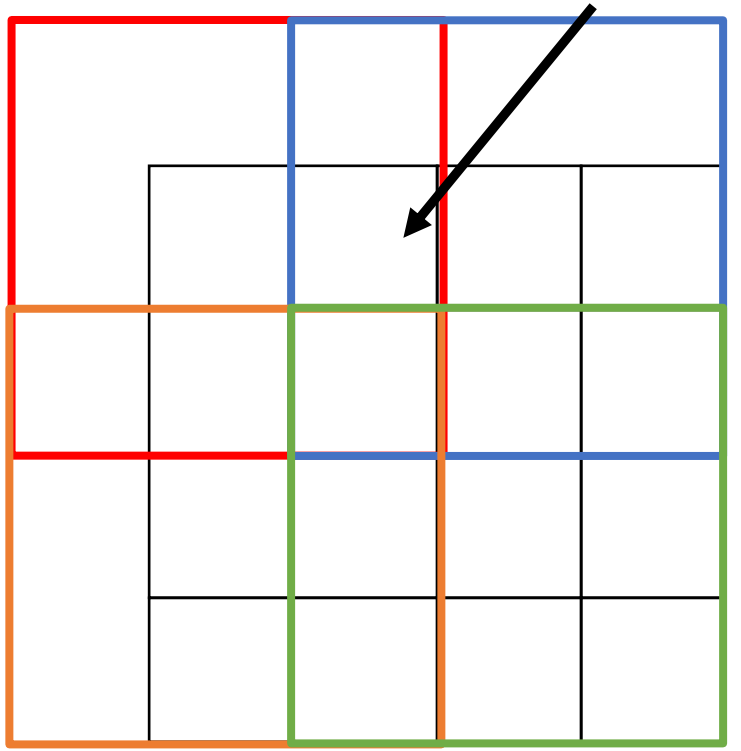
3 x 3 **convolution transpose**, stride 2

Sum where output overlaps

This gives 5x5 output – need to trim one pixel from top and left to give 4x4 output



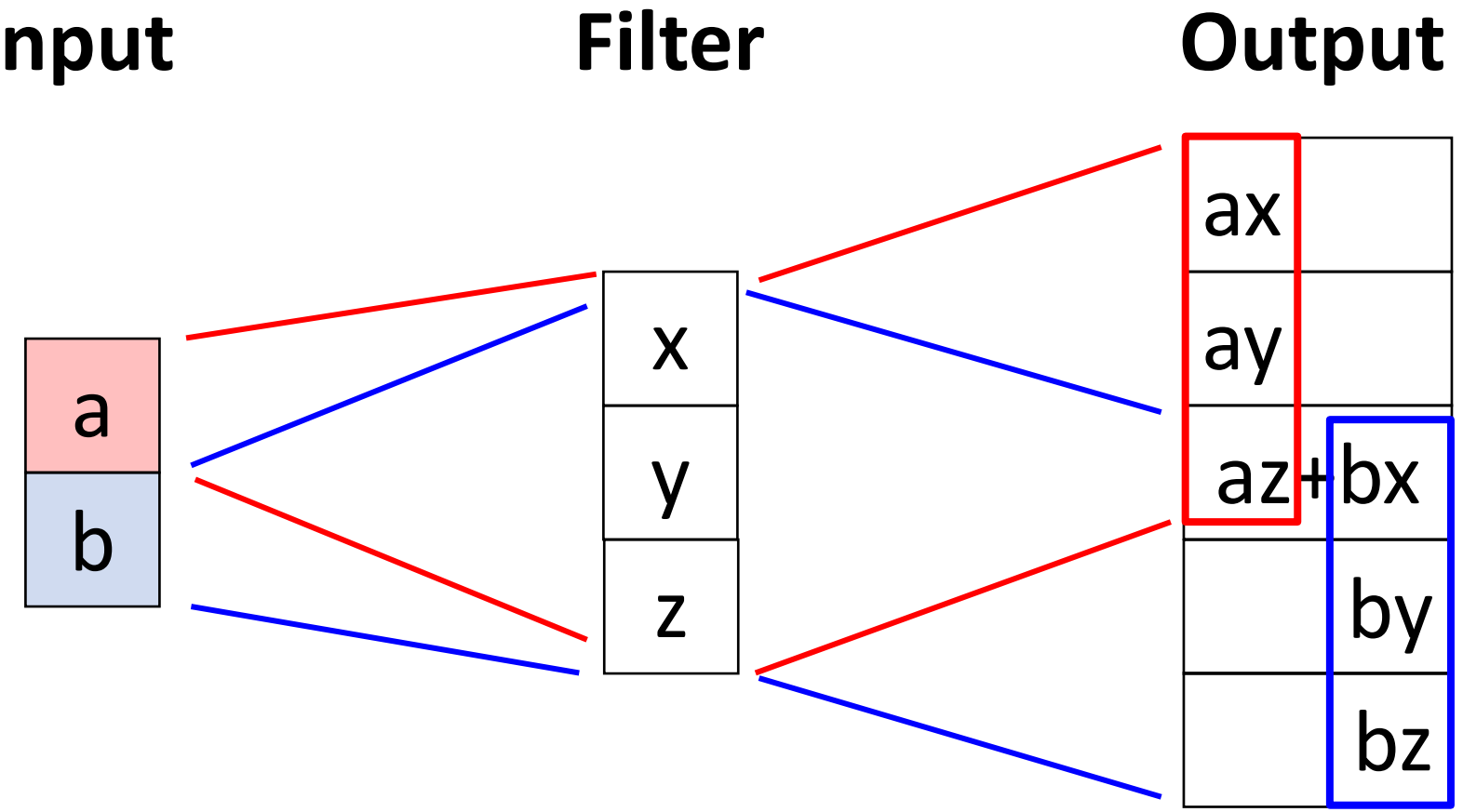
Weight filter by input value and copy to output



Input: 2 x 2

Output: 4 x 4

Transposed Convolution: 1D example

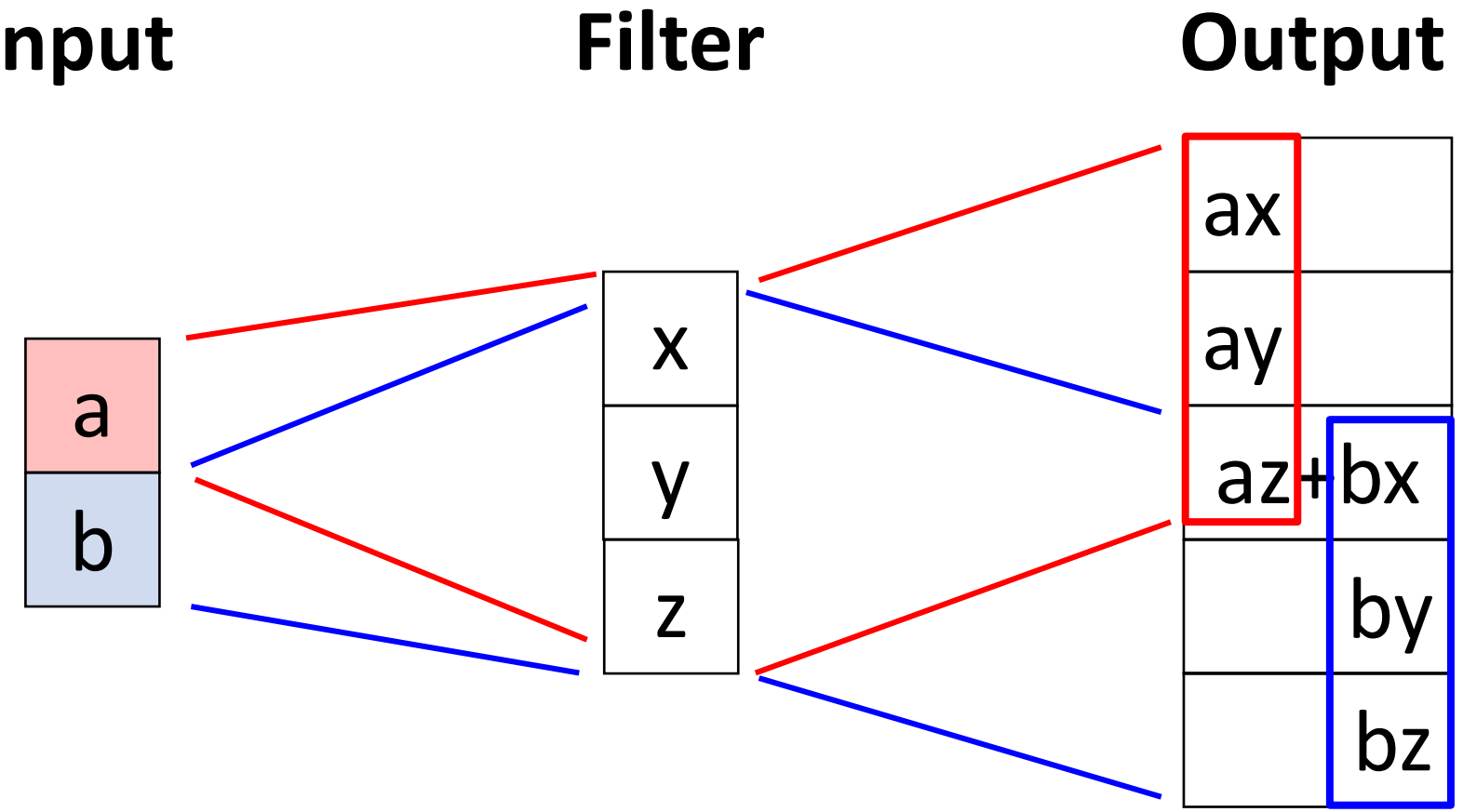


Output has copies of filter weighted by input

Stride 2: Move 2 pixels output for each pixel in input

Sum at overlaps

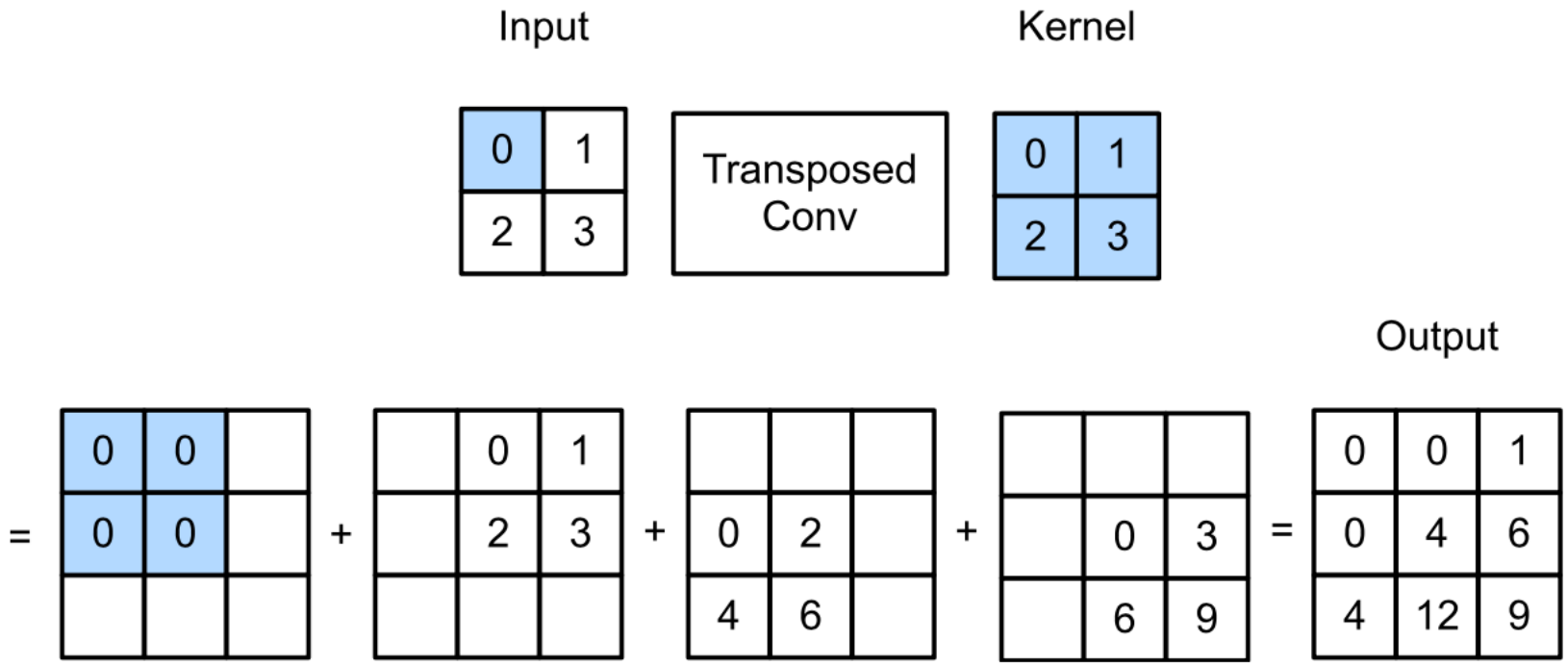
Transposed Convolution: 1D example



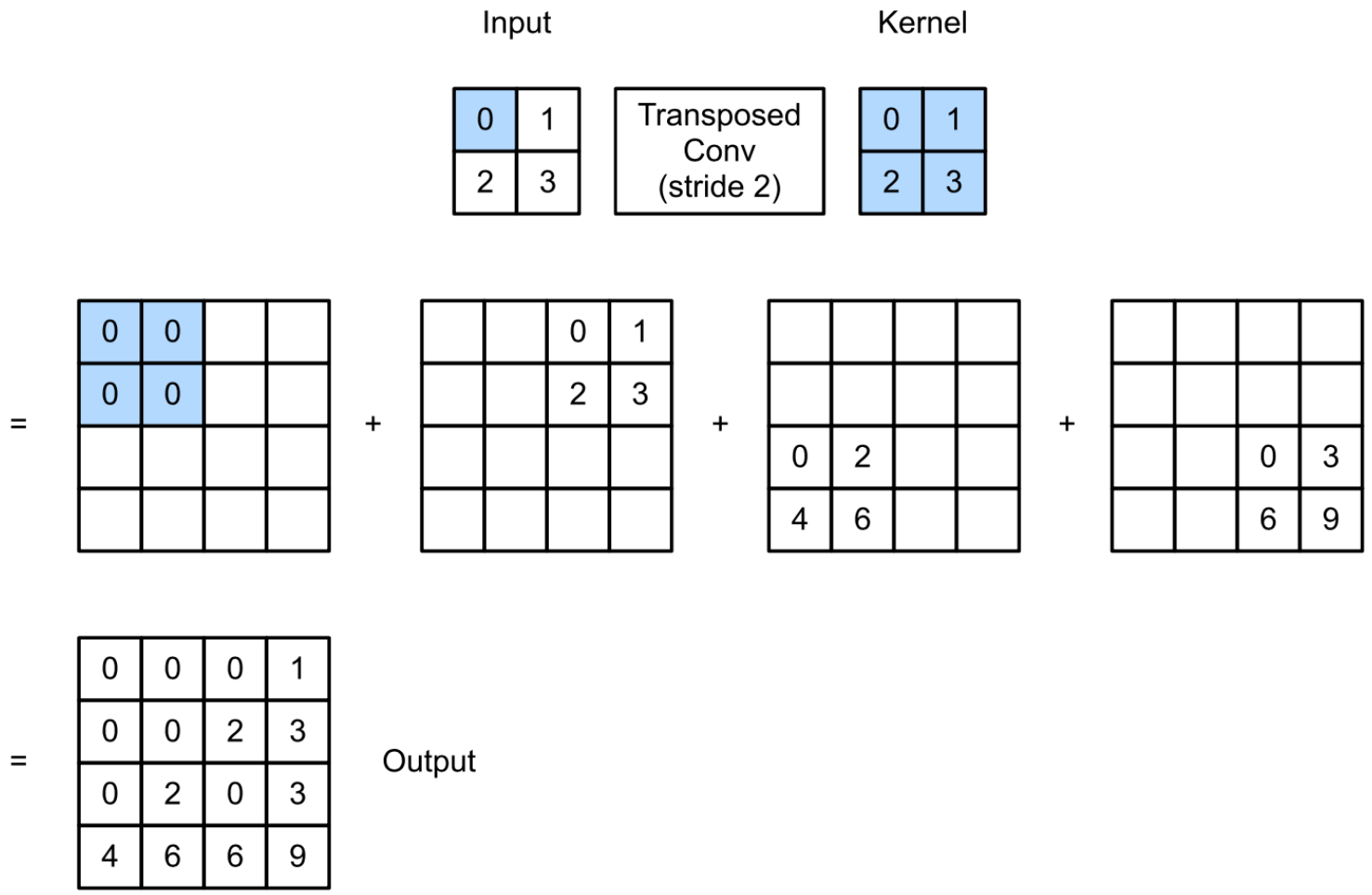
This has many names:

- Deconvolution (bad)!
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution
- Transposed Convolution (best name)

Transposed Convolution: 2D example



Transposed Convolution: 2D example



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, transposed conv is just a regular conv (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, transposed convolution cannot be expressed as normal conv

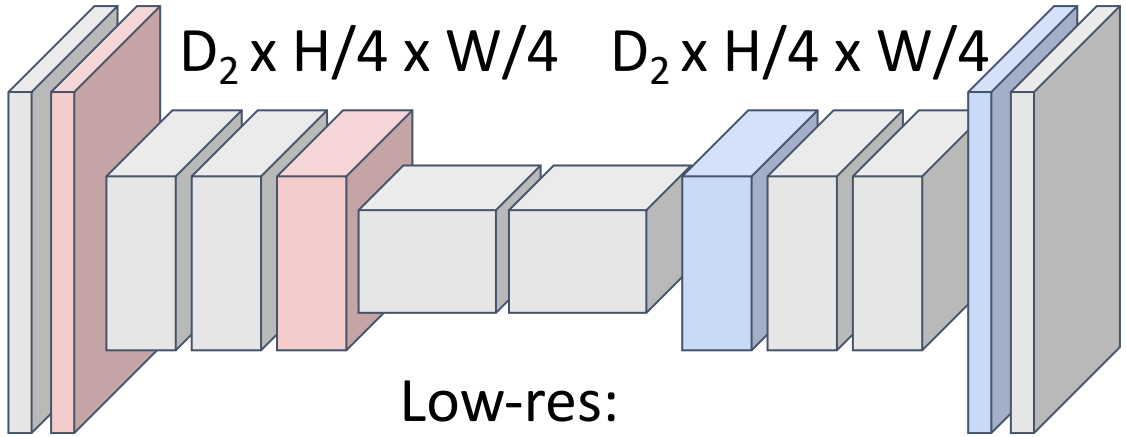
Semantic Segmentation: Fully Convolutional Network

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: Interpolation, transposed conv

Input:
3 x H x W



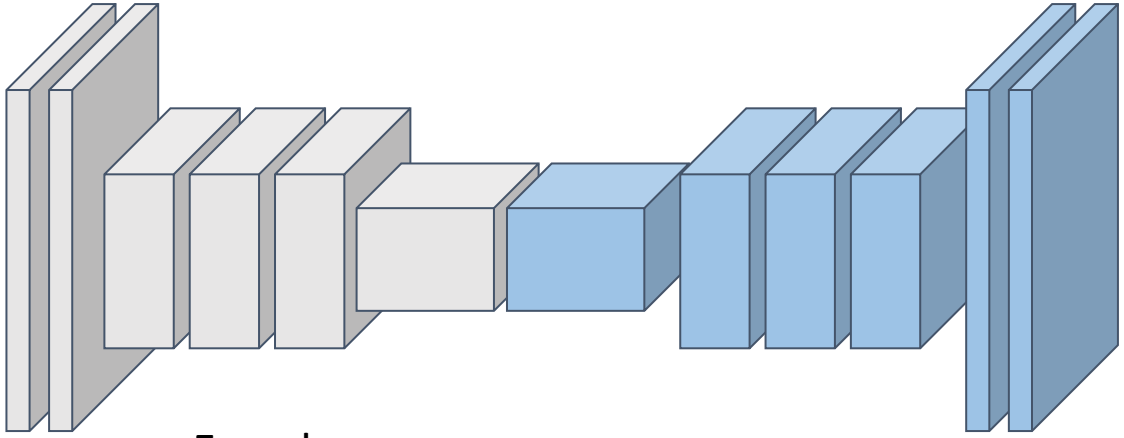
Predictions:
H x W

Loss function: Per-Pixel cross-entropy

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic segmentation networks (encoder+decoder)

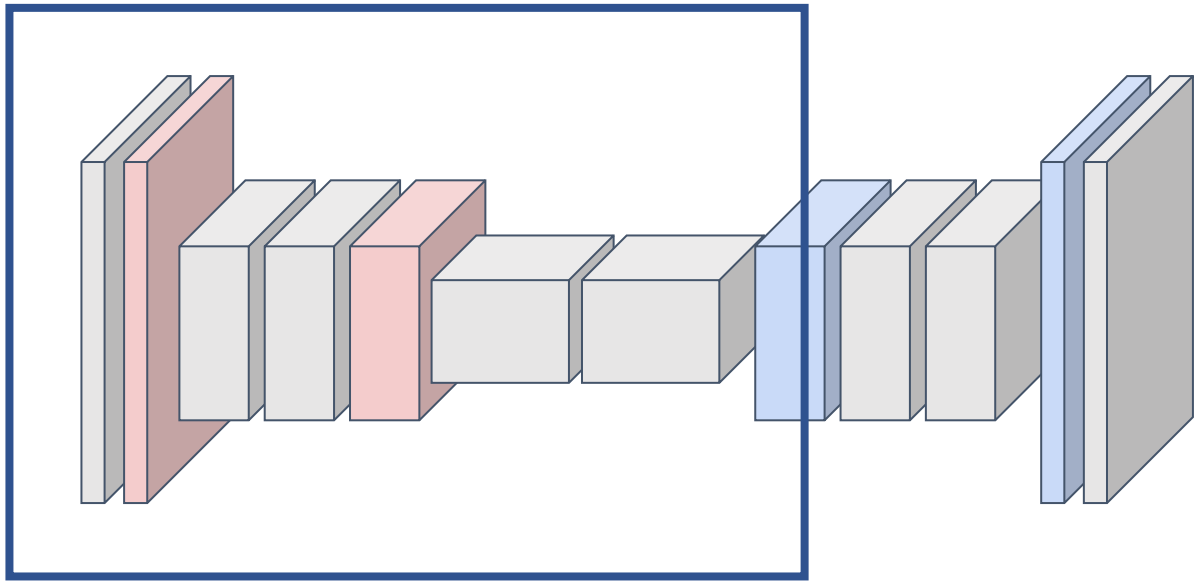
Encoder side: How you can better encode spatial context information?
Decoder side: How you can maintain the spatial structure of the mask?



See the list of recent models at https://github.com/open-mmlab/mmsegmentation

Innovation on encoder: How to better encode the scene context

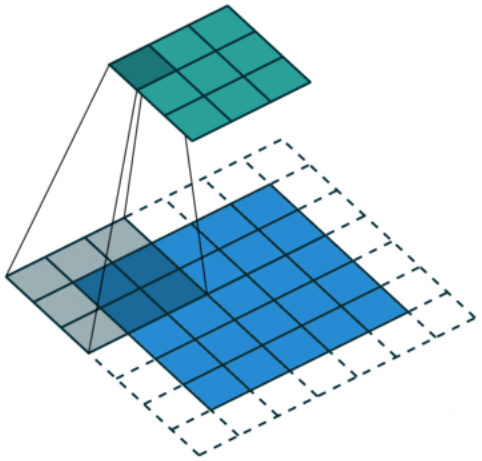
Input:
3 x H x W



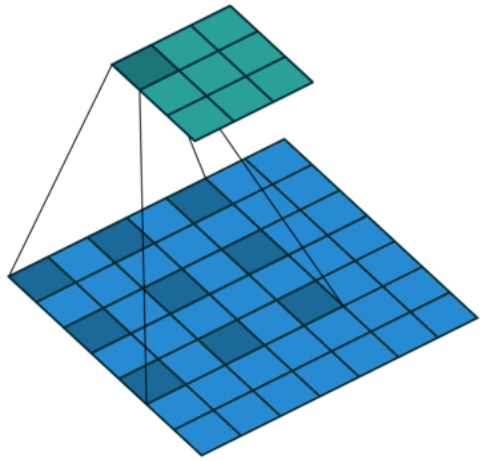
Predictions:
H x W

Dilated Convolution / Astrous Convolution

Standard Convolution



Dilated Convolution: a larger receptive field

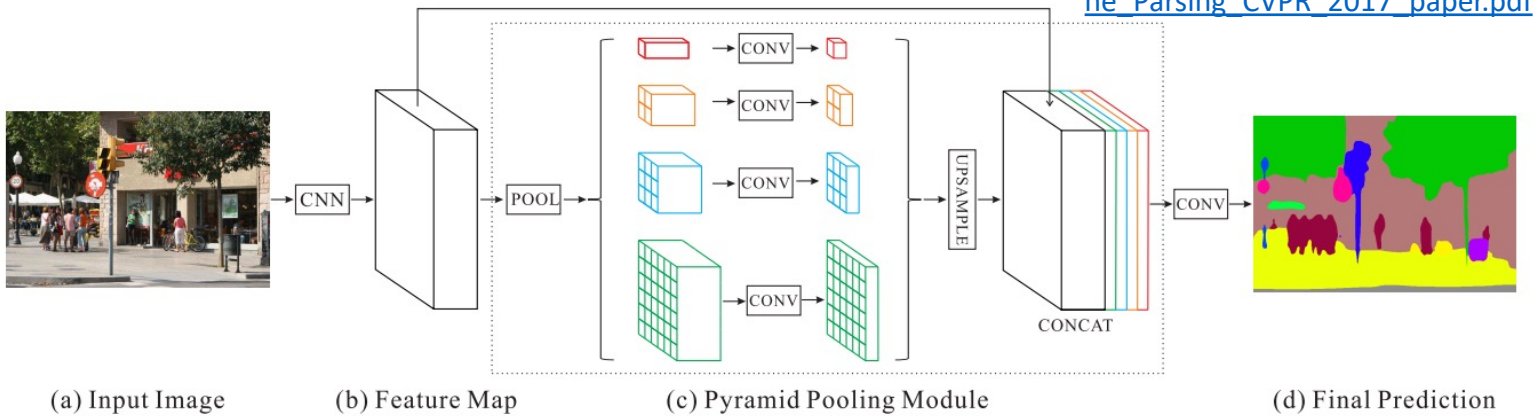


[Multi-Scale Context Aggregation by Dilated Convolutions](#)

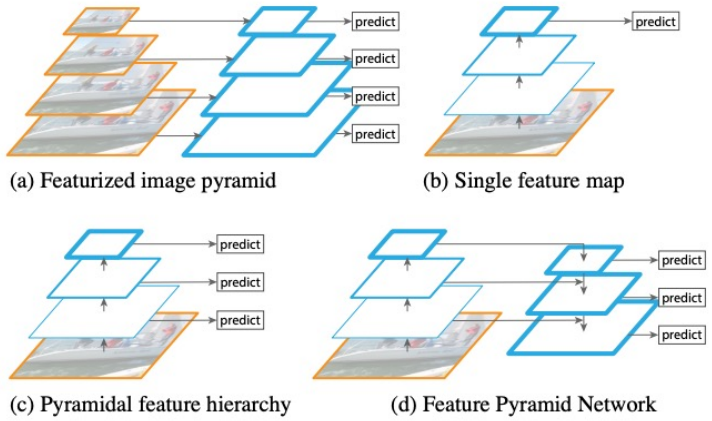
Feature Pyramid Structures

Pyramid Scene Parsing Network (PSPnet)

https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhao_Pyramid_Scene_Parsing_CVPR_2017_paper.pdf

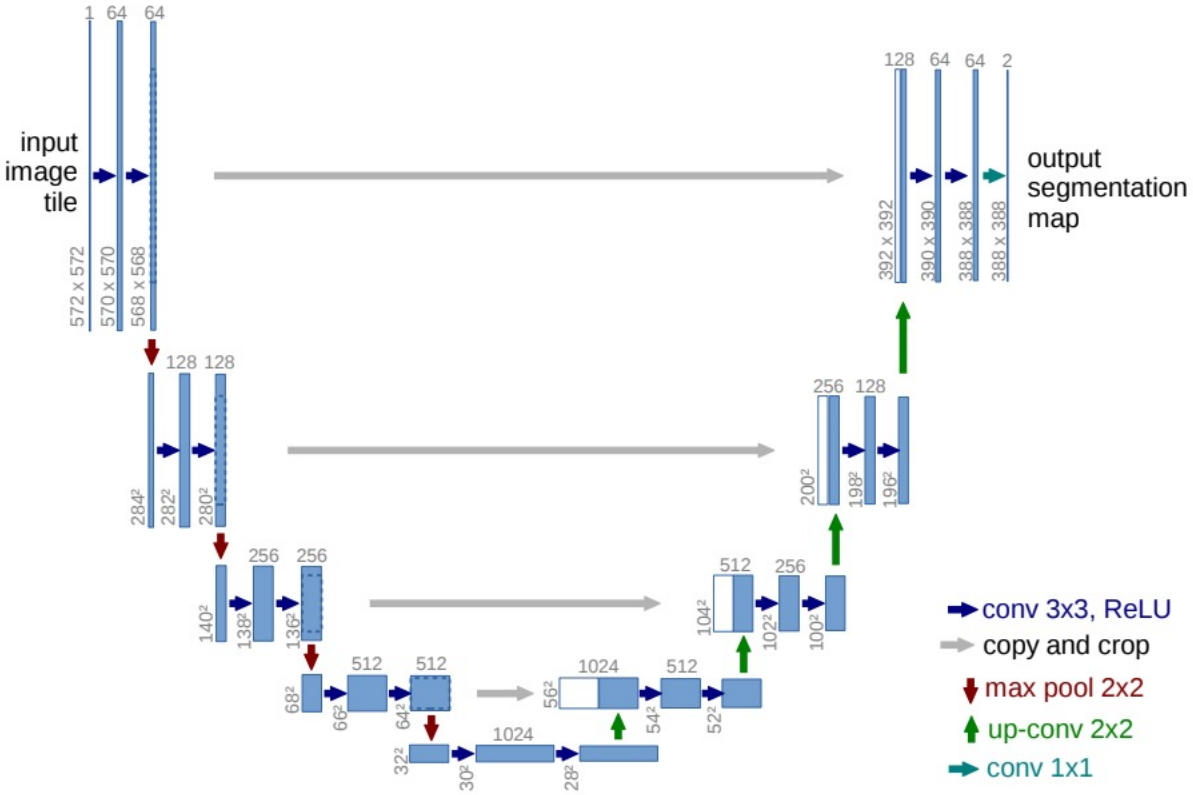


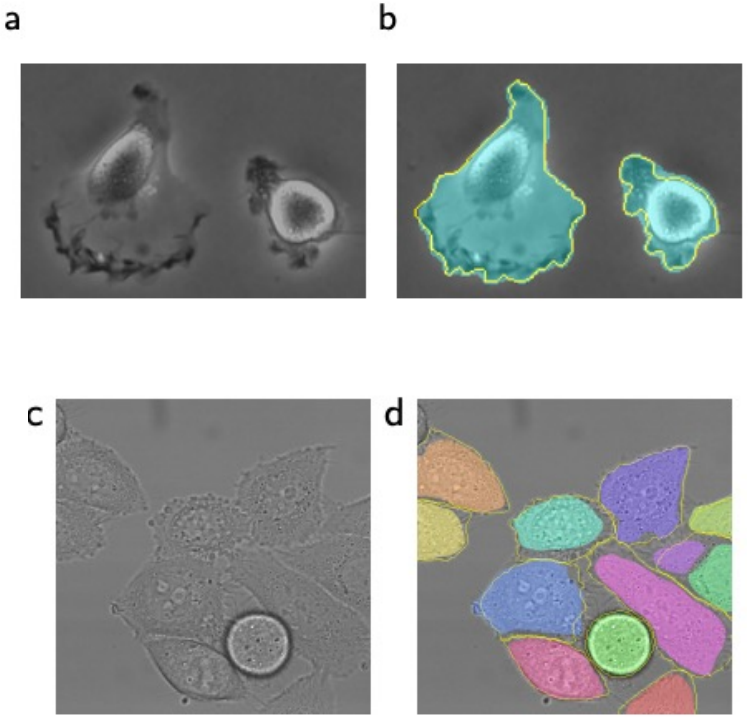
Feature Pyramid Networks (FPN)



https://arxiv.org/pdf/1612.03144.pdf

U-Net: a popular network for biomedical image segmentation (>38,000 citations)





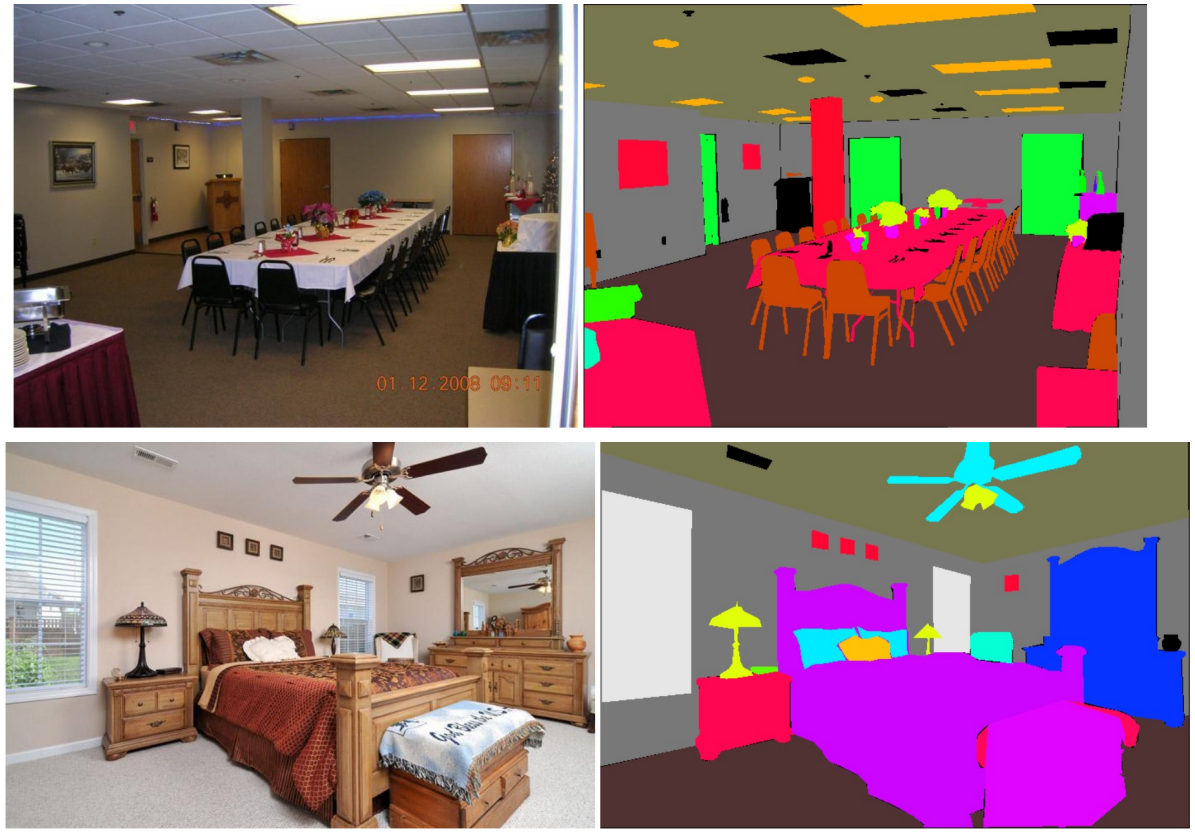
U-Net: Convolutional Networks for Biomedical Image Segmentation https://arxiv.org/pdf/1505.04597.pdf
Olaf Ronneberger, Philipp Fischer, Thomas Brox

Semantic segmentation datasets

Cityscapes Dataset
(30 classes and 5 K images in driving scenes)

https://www.cityscapes-dataset.com/

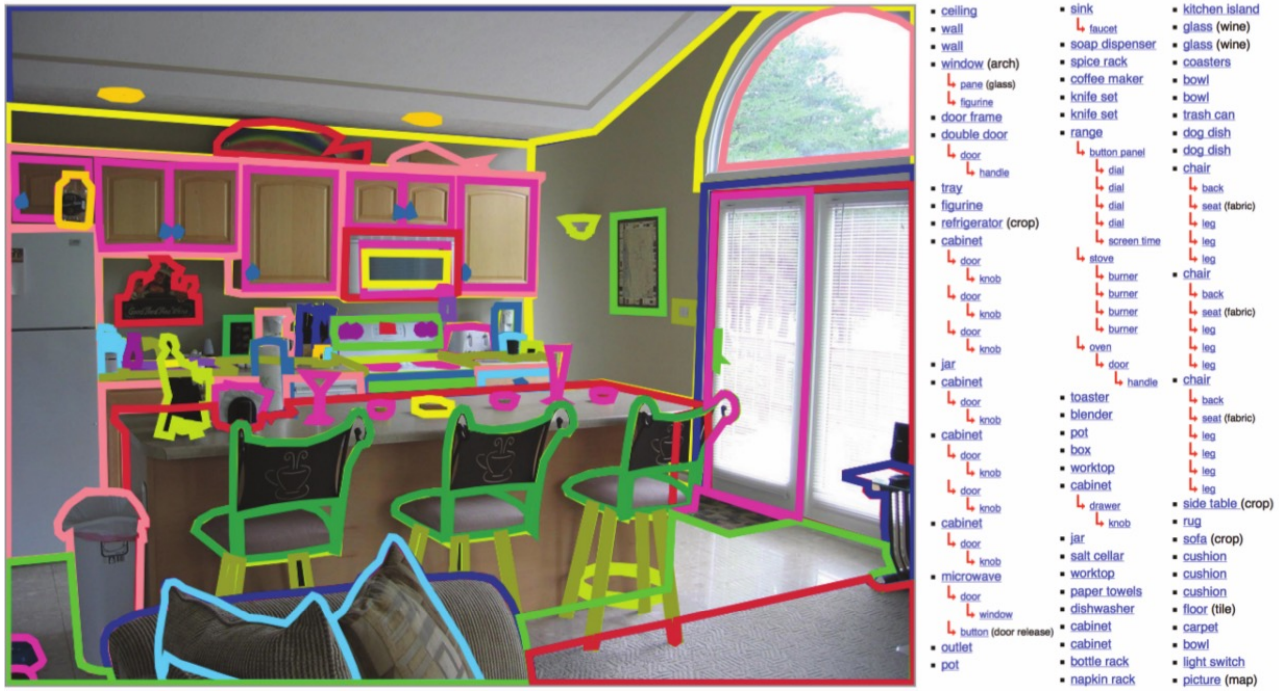
MIT ADE20K
(150 classes, 25K training images)



Scene Parsing through ADE20K Dataset. B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso and A. Torralba. Computer Vision and Pattern Recognition (CVPR), 2017.

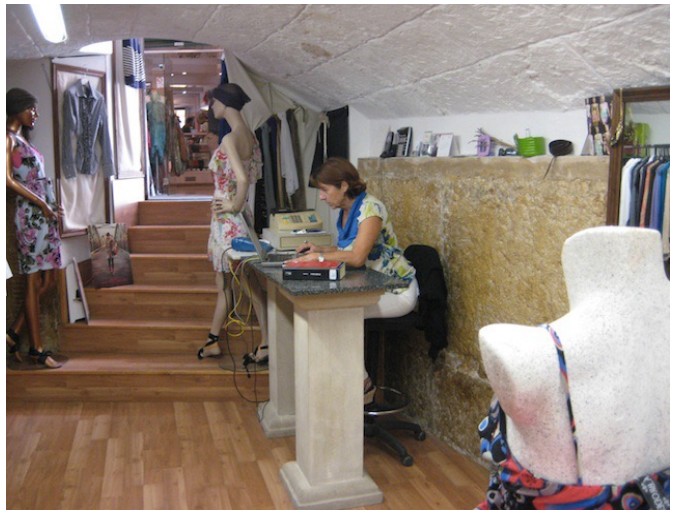
Constructing ADE20K Dataset

- Annotating each object instances in a scene
- Single expert annotator for a few years of work



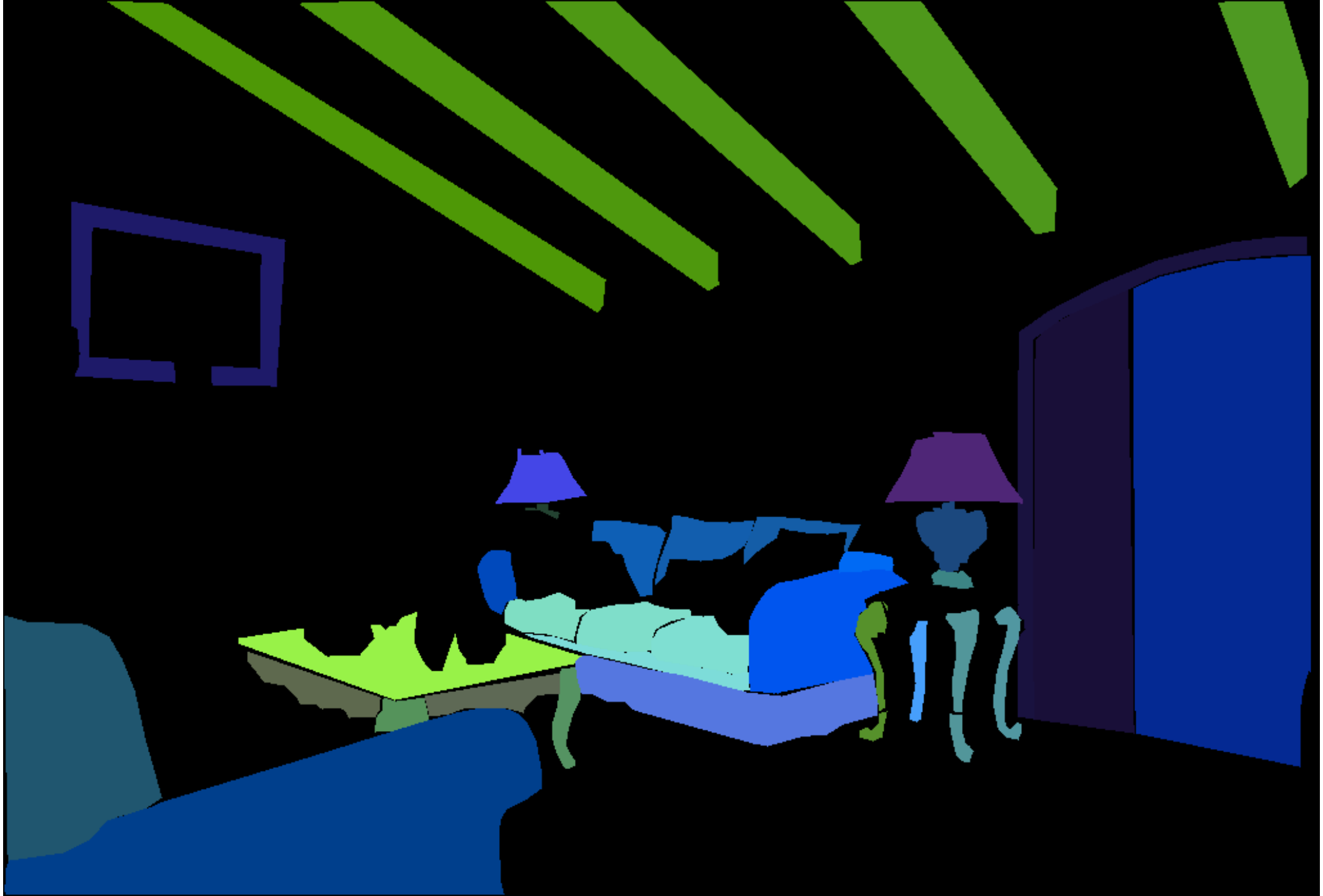
Labelme Annotation Tool

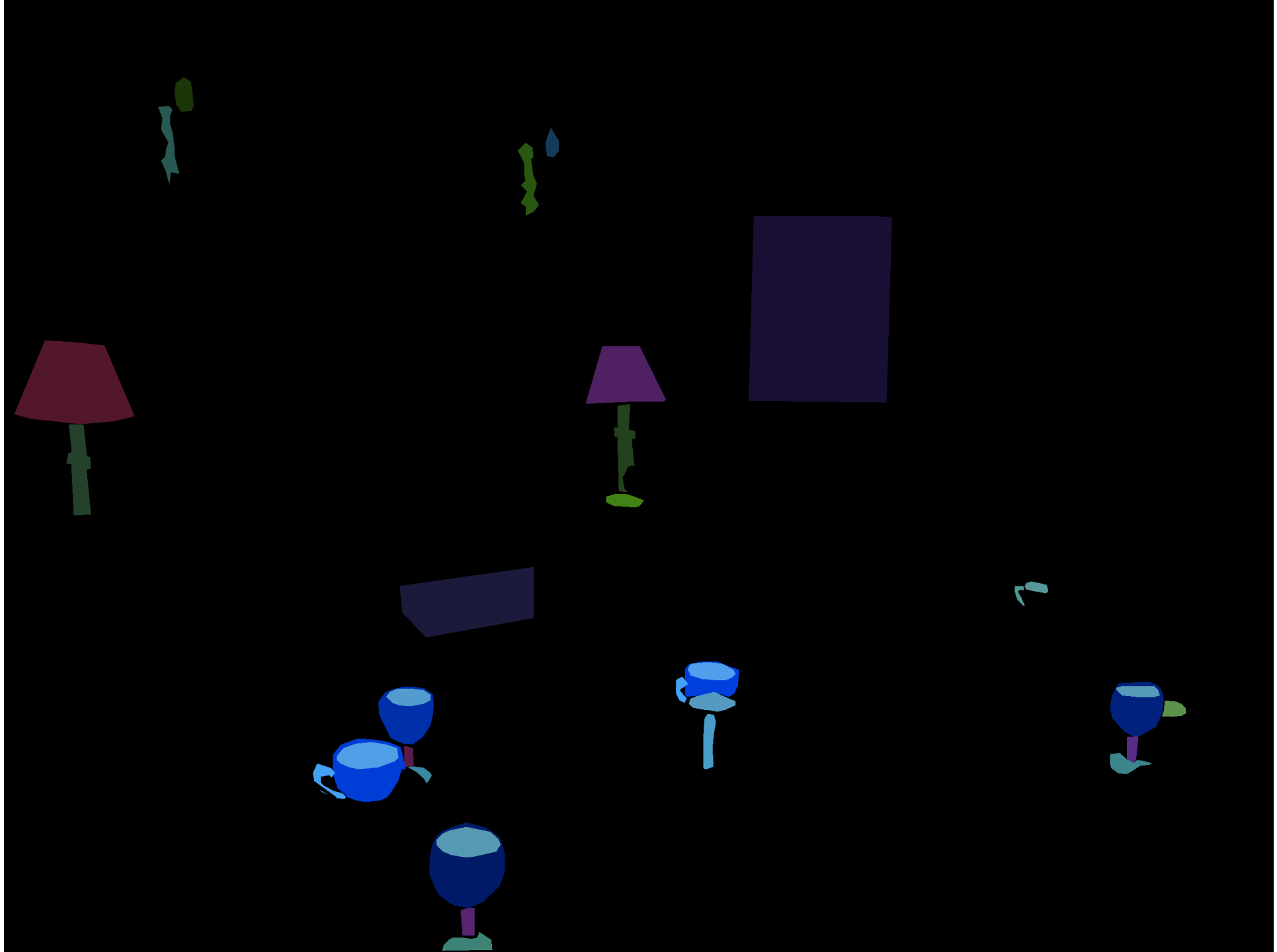
Ms. Adela Barriuso



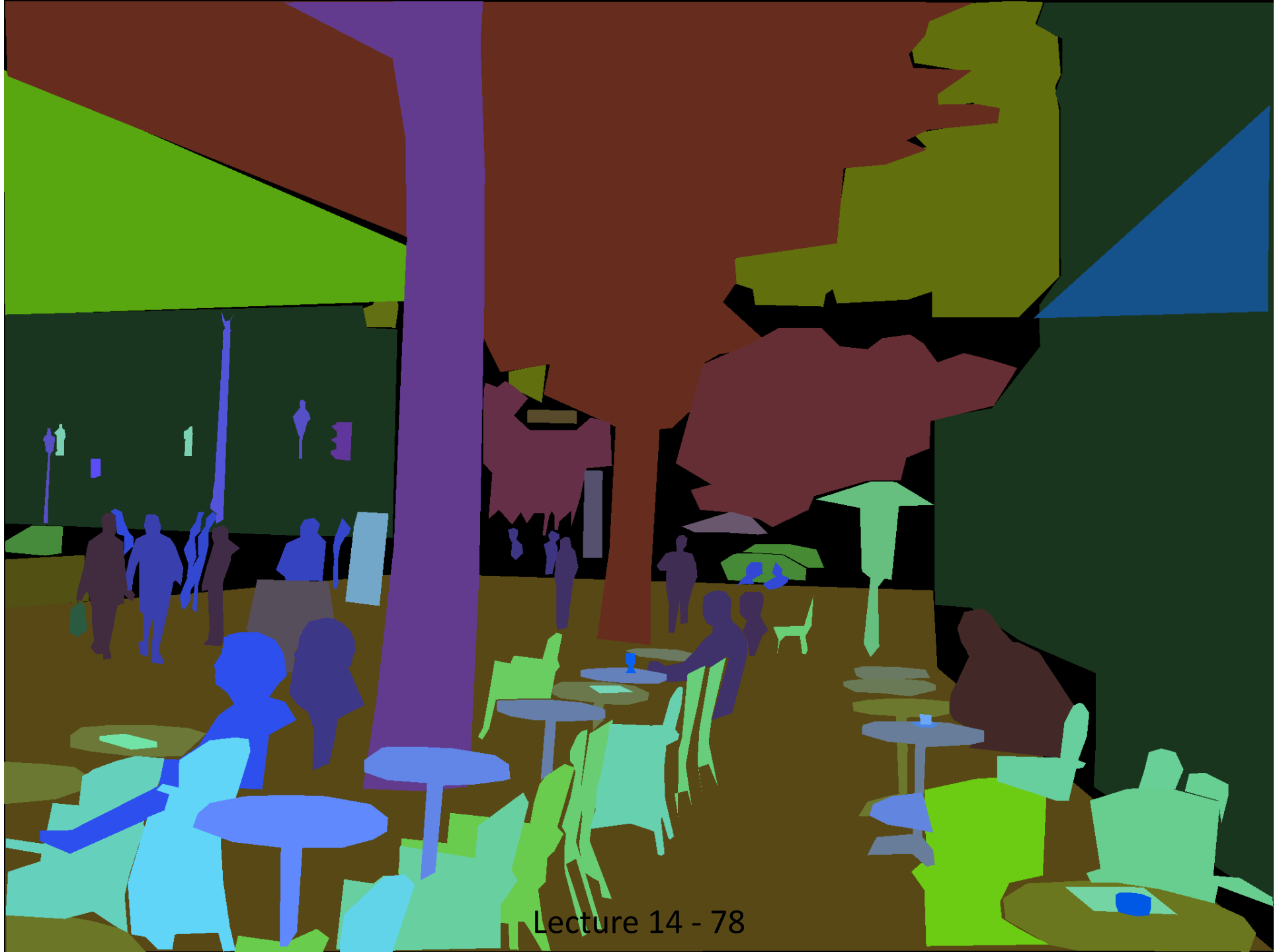
http://groups.csail.mit.edu/vision/datasets/ADE20K/


©davidpalermo.com

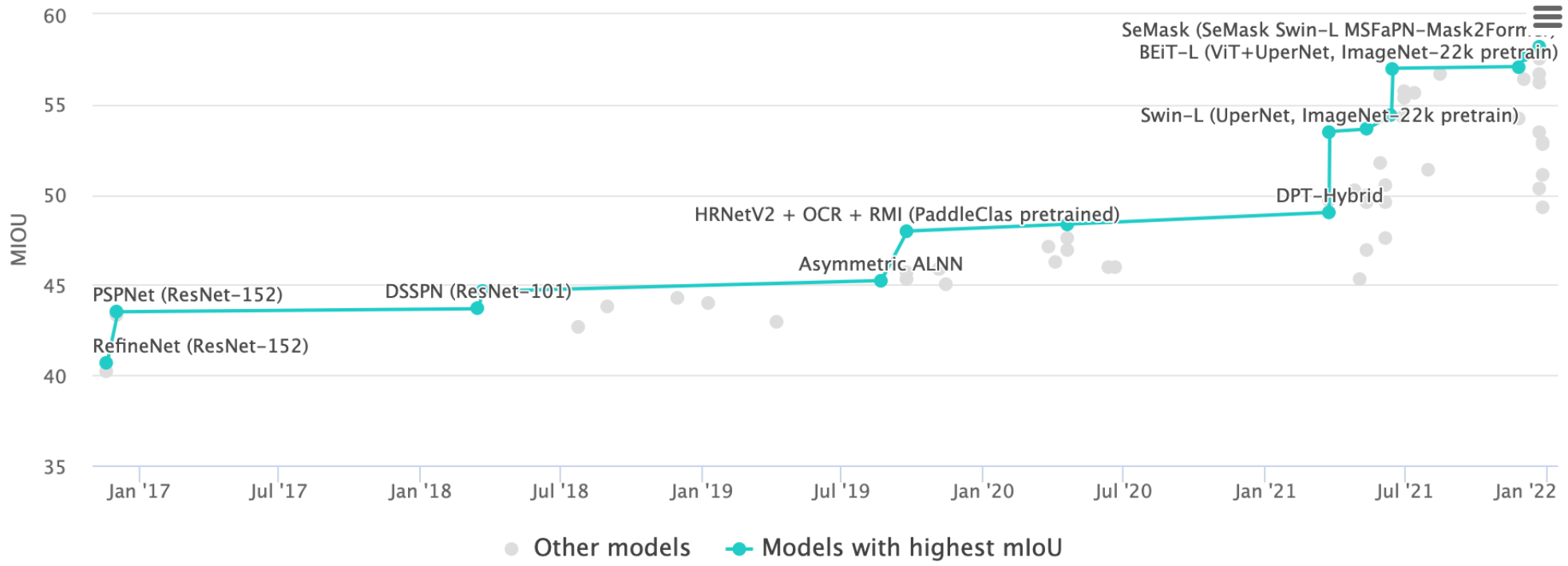





METROPOLITAIN
Glaces & Sorbets
Berthillon
Crêpes



Progress of semantic segmentation on ADE20K

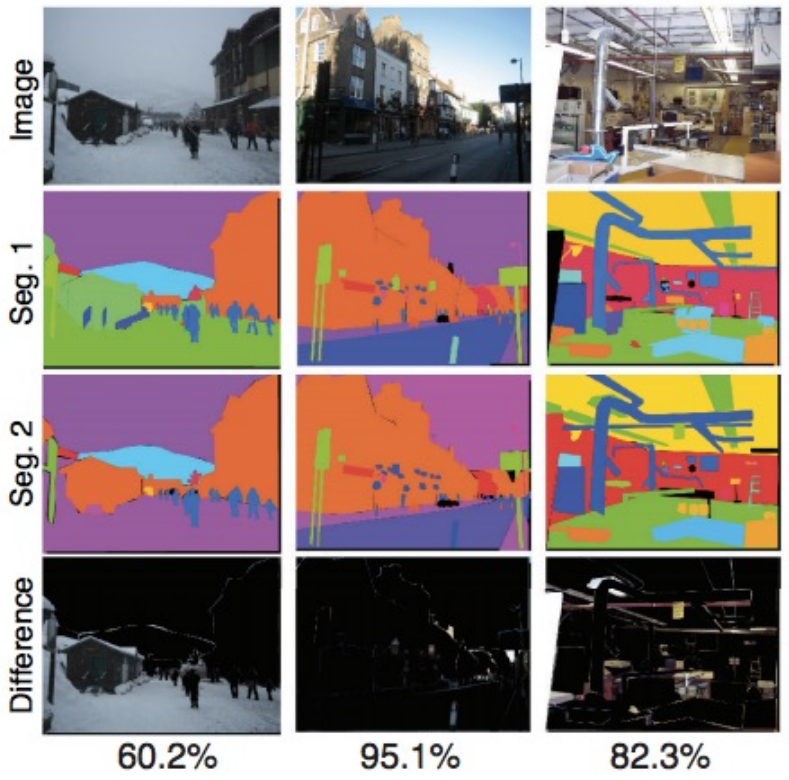


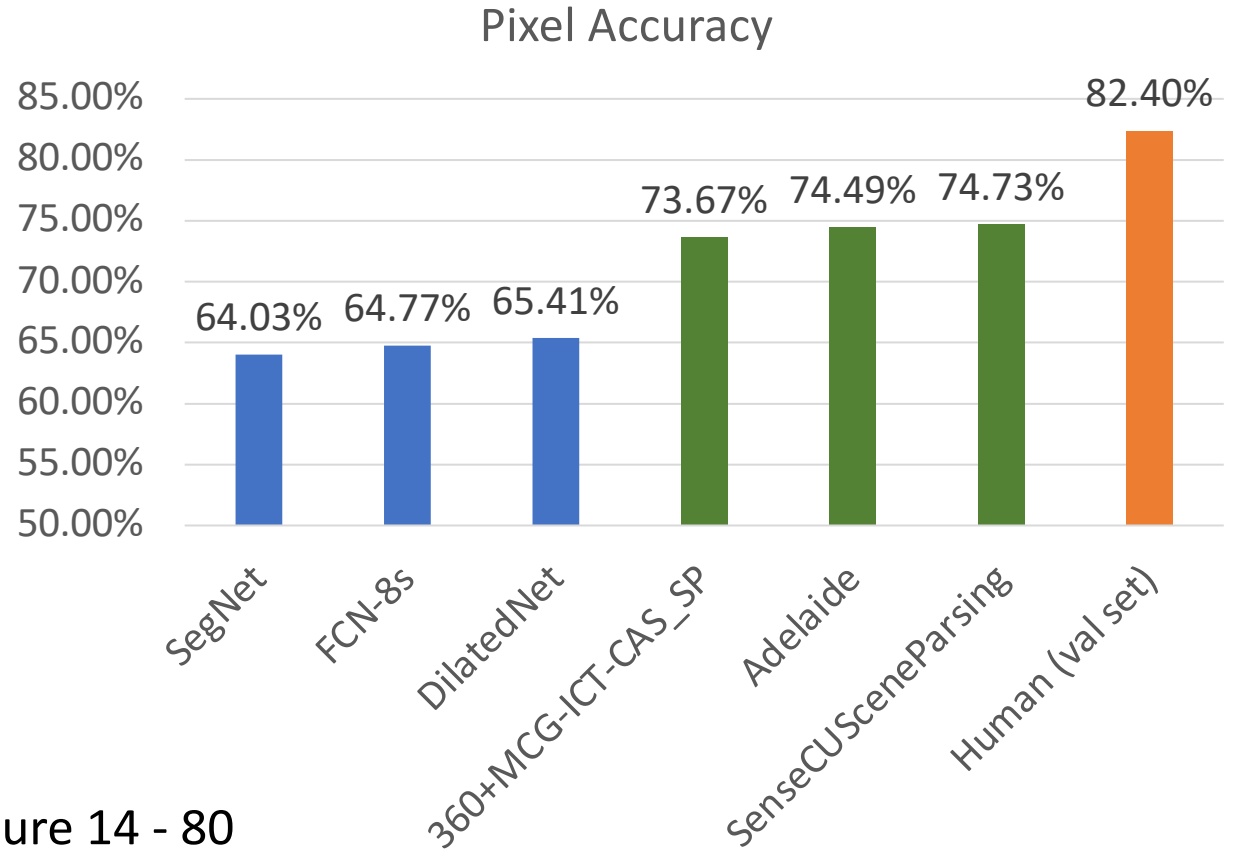
https://paperswithcode.com/sota/semantic-segmentation-on-ade20k-val

Data Consistency and Human Performance

- 61 images from val set are re-annotated after 6 months.

82.4% pixels got the same label.





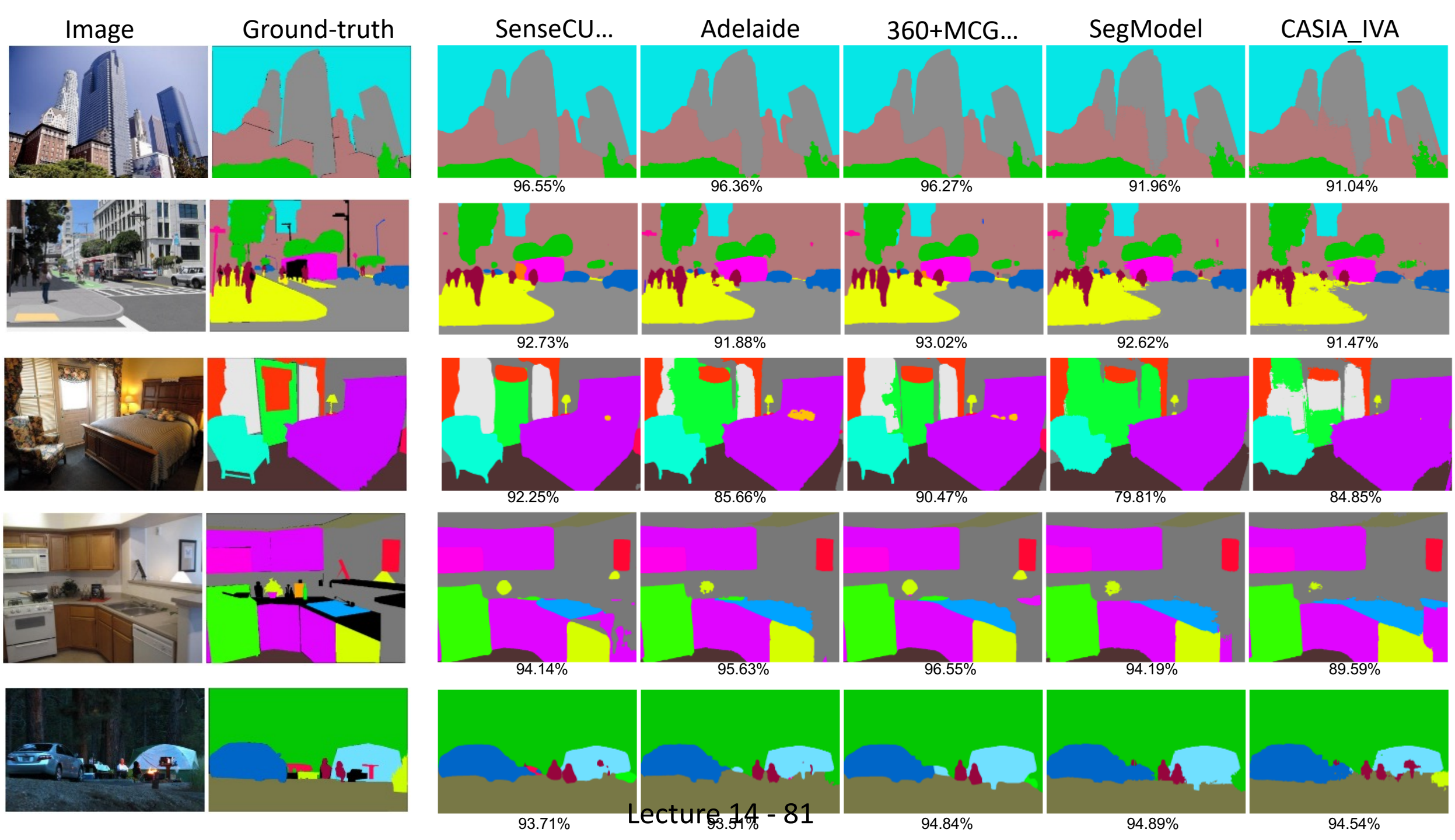


Image
Ground-truth
SenseCU…
Adelaide
360+MCG…
SegModel
CASIA_IVA
96.55%
96.36%
96.27%
91.96%
91.04%
92.73%
91.88%
93.02%
92.62%
91.47%
92.25%
85.66%
90.47%
79.81%
84.85%
94.14%
95.63%
96.55%
94.19%
89.59%
93.71%
93.51%
94.84%
94.89%
94.54%

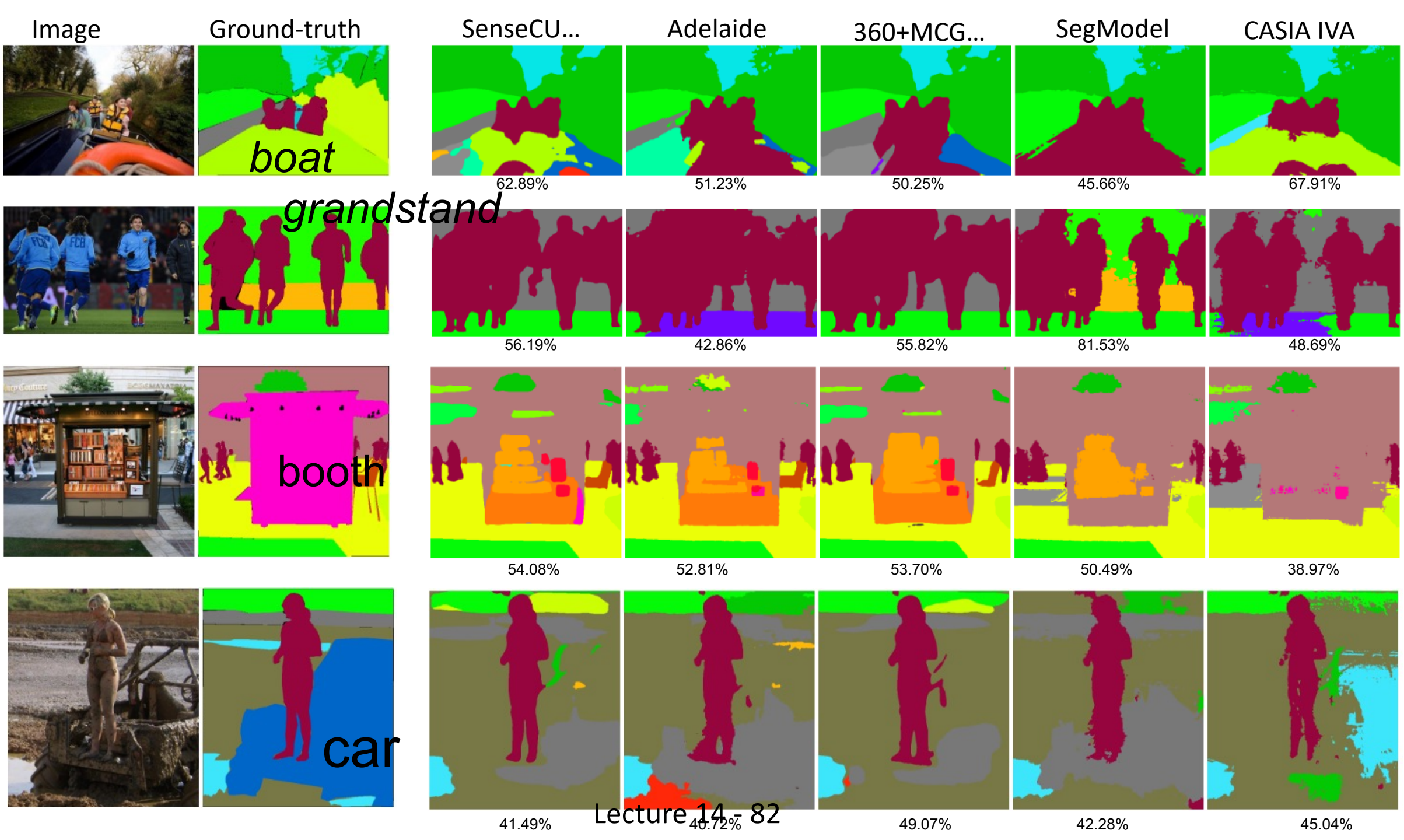
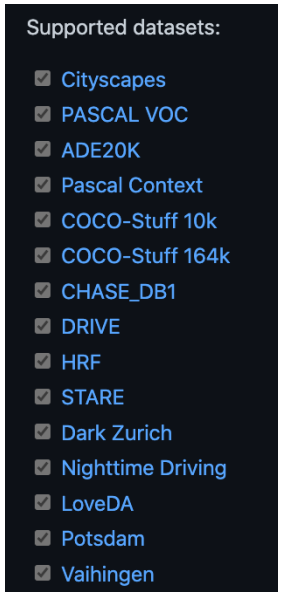


Image
Ground-truth
SenseCU…
Adelaide
360+MCG…
SegModel
CASIA IVA
boat
62.89%
51.23%
50.25%
45.66%
67.91%
grandstand
56.19%
42.86%
55.82%
81.53%
48.69%
booth
54.08%
52.81%
53.70%
50.49%
38.97%
car
41.49%
40.72%
49.07%
42.28%
45.04%

Semantic segmentation libraries

- Semantic segmentation on MIT ADE20K:
  - Link: https://github.com/CSAILVision/semantic-segmentation-pytorch
  - Colab: https://colab.research.google.com/github/CSAILVision/semantic-segmentation-pytorch/blob/master/notebooks/DemoSegmenter.ipynb
- mmSegmentattion: a zoo of any segmentation NNs
  - Link: https://github.com/open-mmlab/mmsegmentation



Demo video of semantic segmentation on CityScapes

Computer Vision Tasks: Instance Segmentation

Classification

CAT

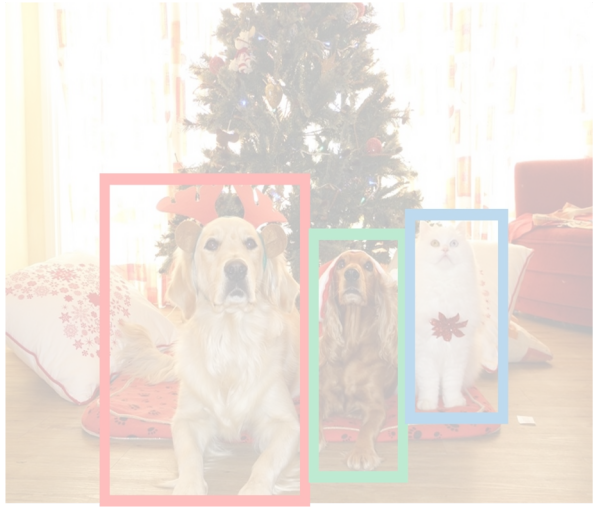
No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

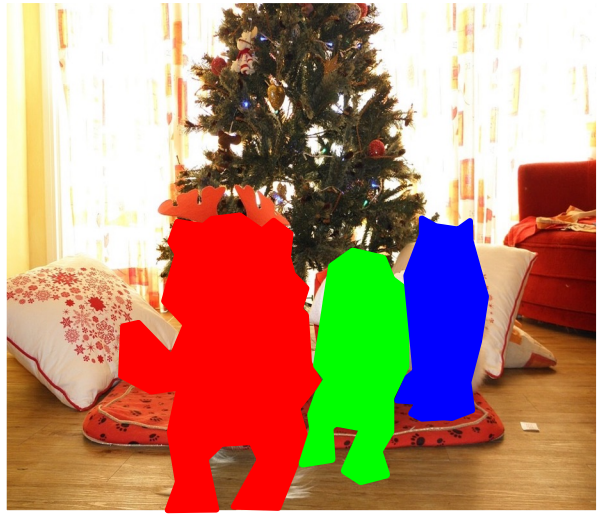
Object Detection



DOG, DOG, CAT

Multiple Objects

Instance Segmentation

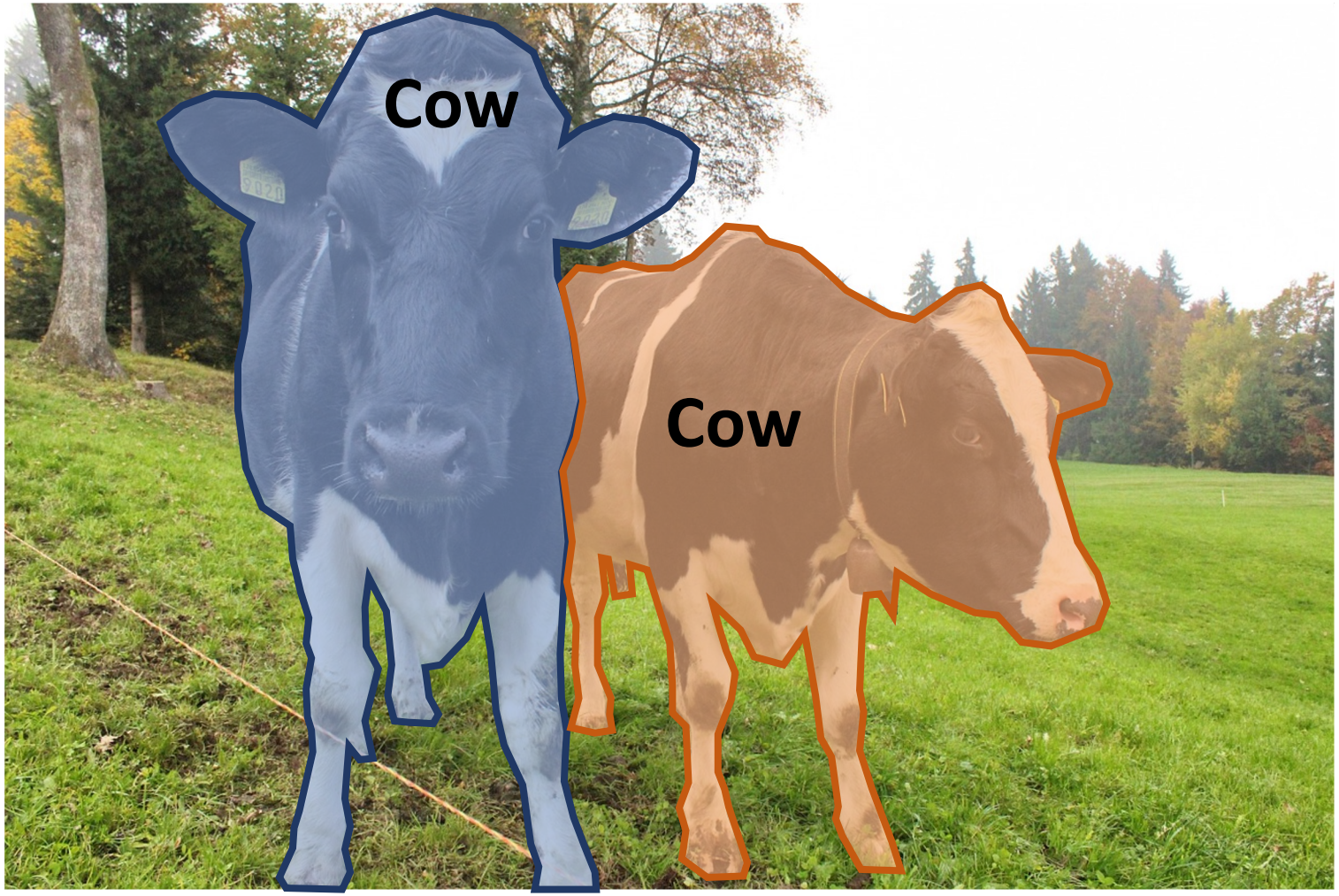


DOG, DOG, CAT

Multiple Objects

Computer Vision Tasks: Instance Segmentation

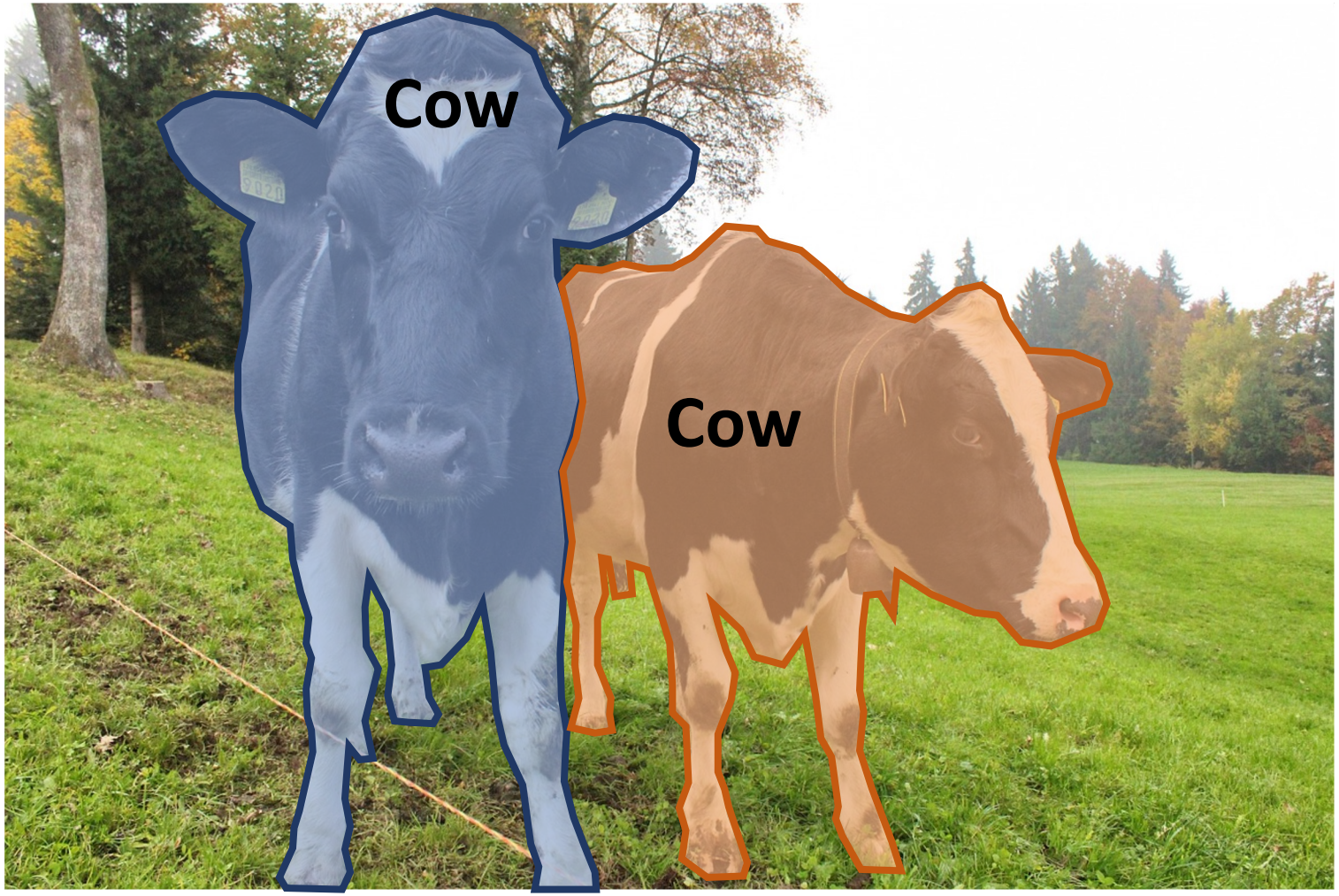
Instance Segmentation: Detect all objects in the image, and identify the pixels that belong to each object (Only things!)



Computer Vision Tasks: Instance Segmentation

Instance Segmentation: Detect all objects in the image, and identify the pixels that belong to each object (Only things!)

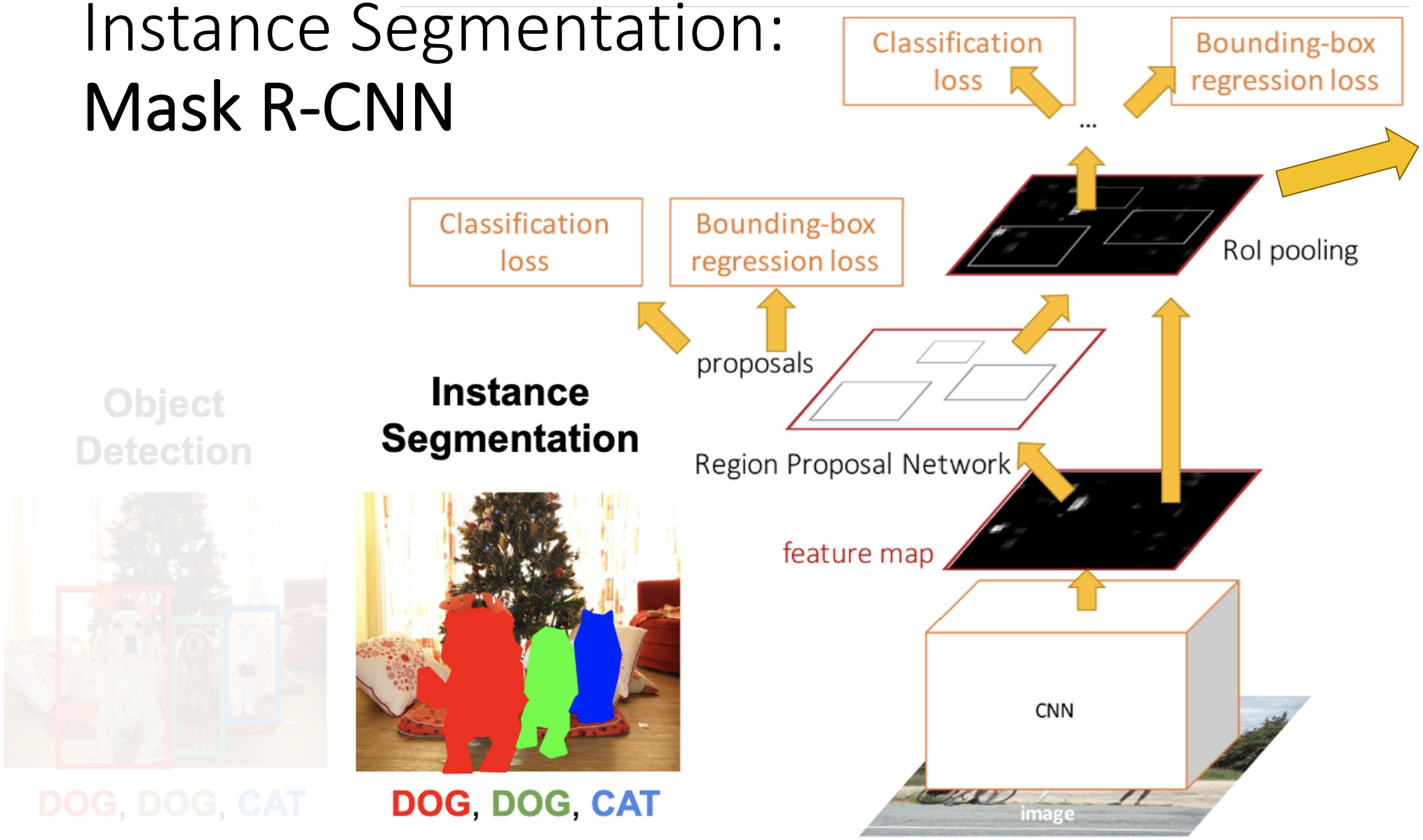
Approach: Perform object detection, then predict a segmentation mask for each object!



Object Detection: Faster R-CNN

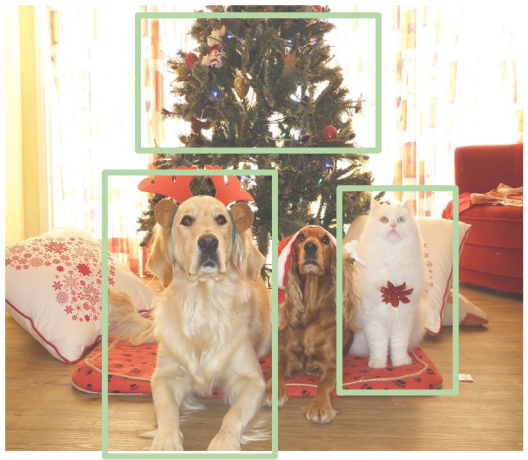
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NeurIPS 2015

Instance Segmentation: Mask R-CNN

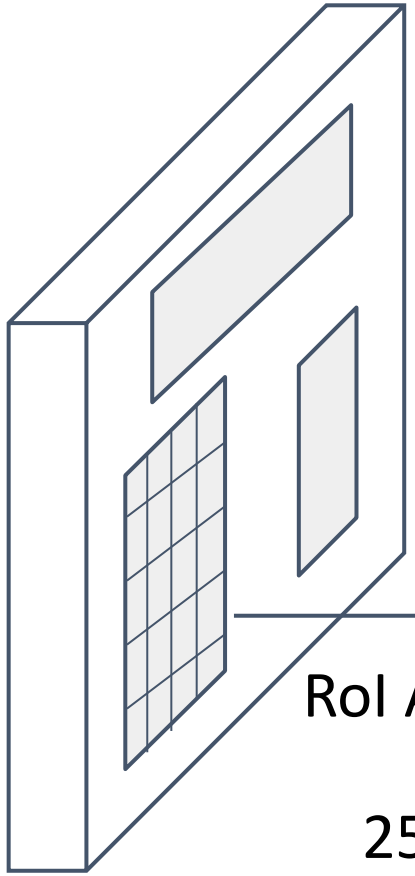


He et al, "Mask R-CNN", ICCV 2017

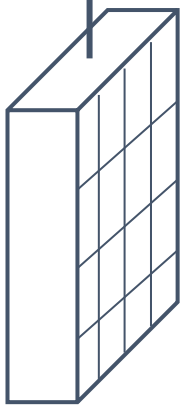
Mask R-CNN



CNN
+RPN

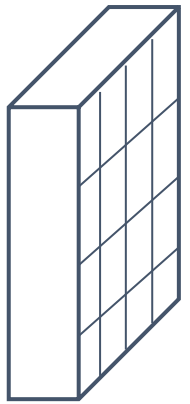


RoI Align



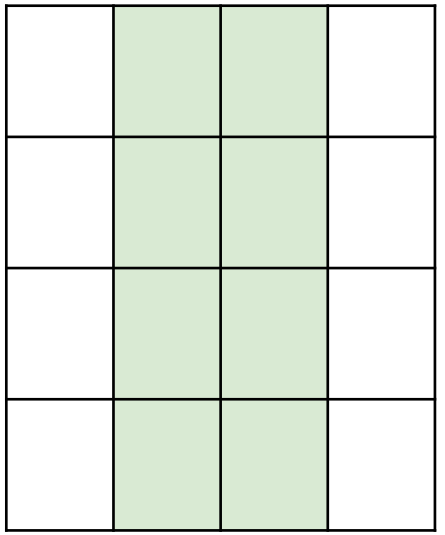
256 x 14 x 14

Conv



256 x 14 x 14

Conv

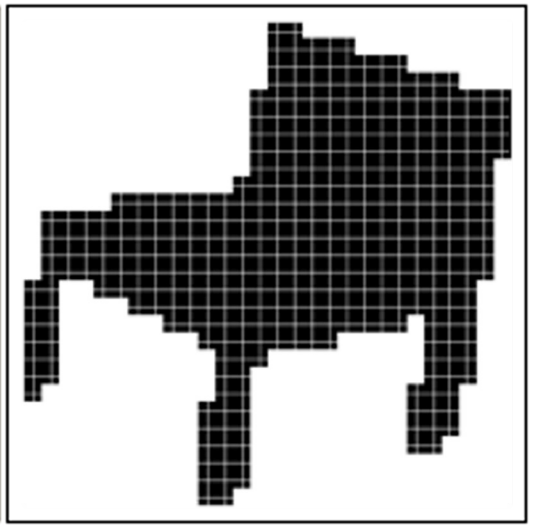


Classification Scores: C
Box coordinates (per class):
4 * C

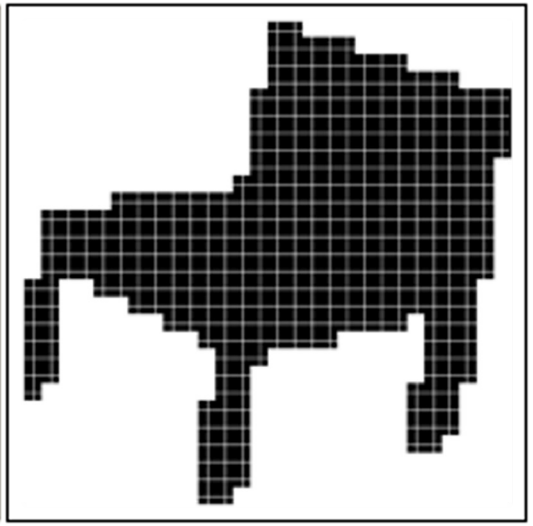
Predict a mask for
each of C classes:
C x 28 x 28 (mask size)

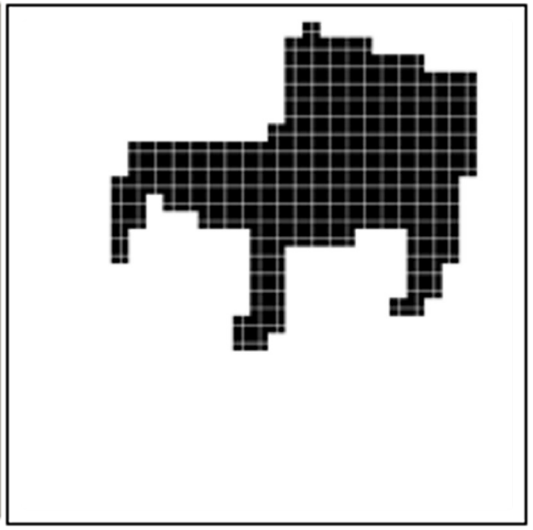
He et al, “Mask R-CNN”, ICCV 2017

Mask R-CNN: Example Training Targets

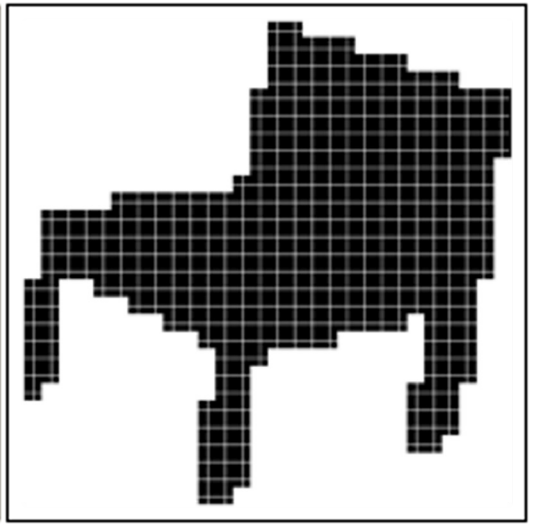
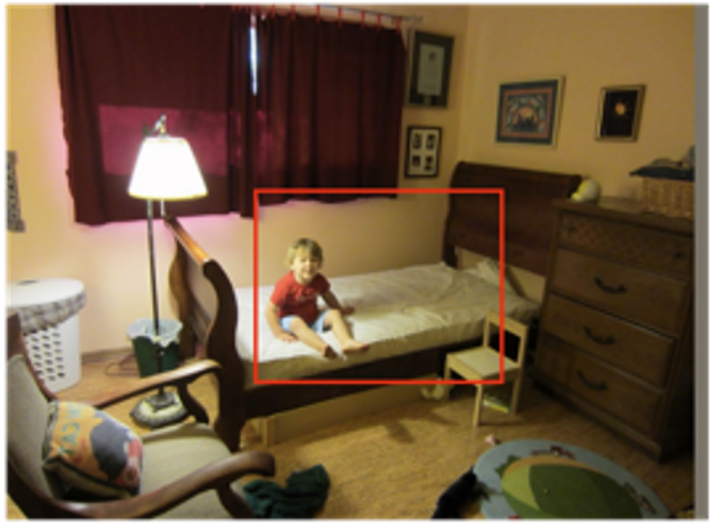
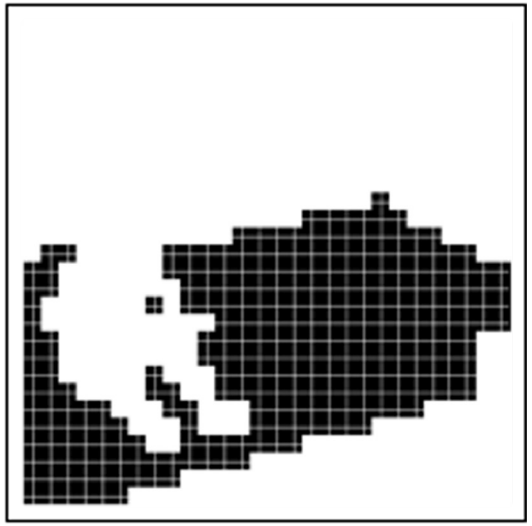


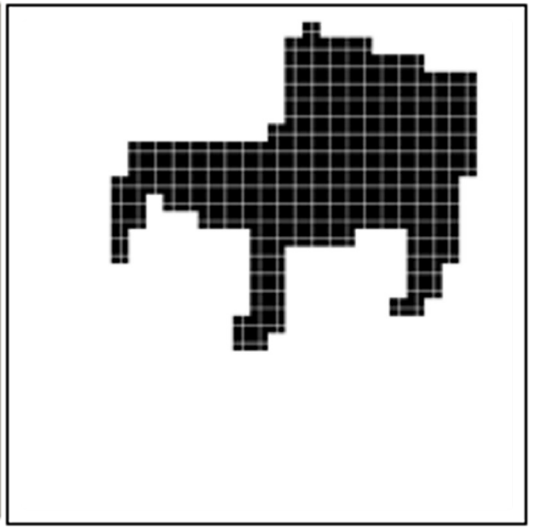
Mask R-CNN: Example Training Targets



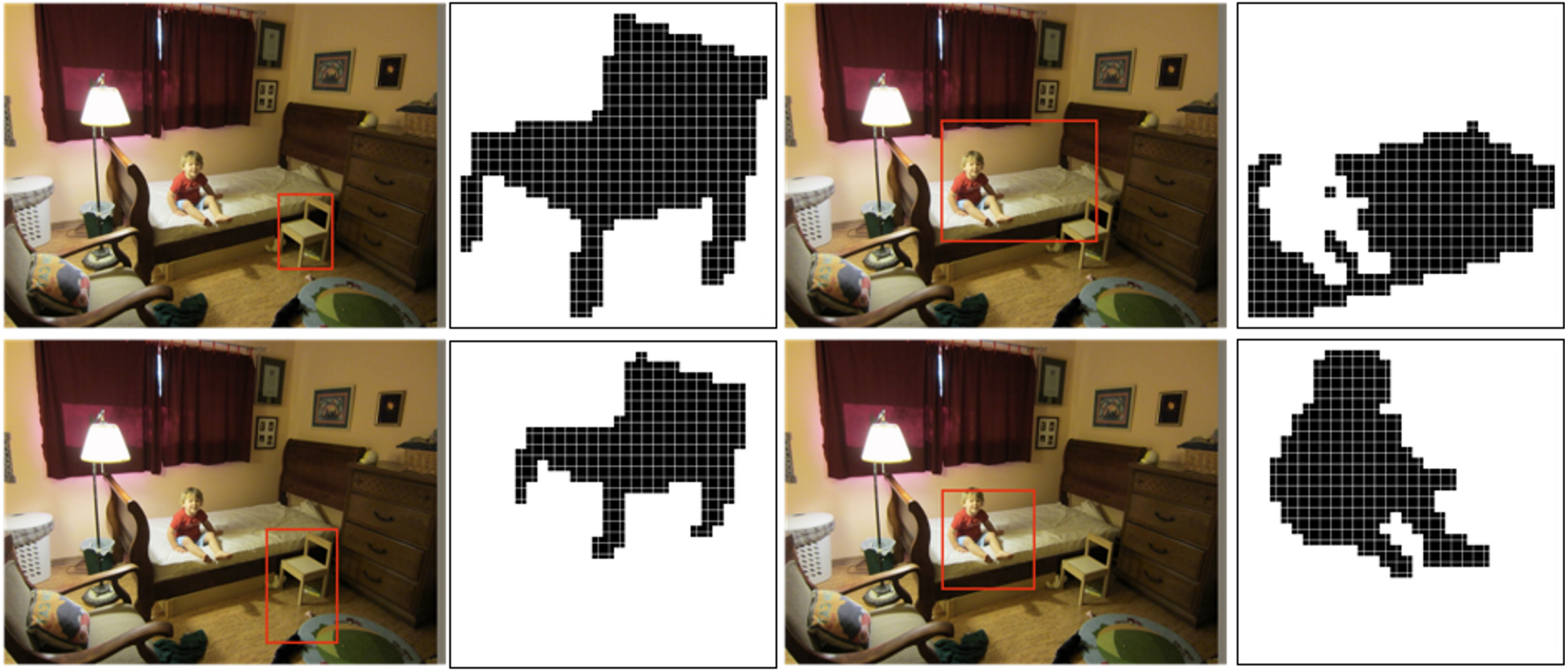


Mask R-CNN: Example Training Targets

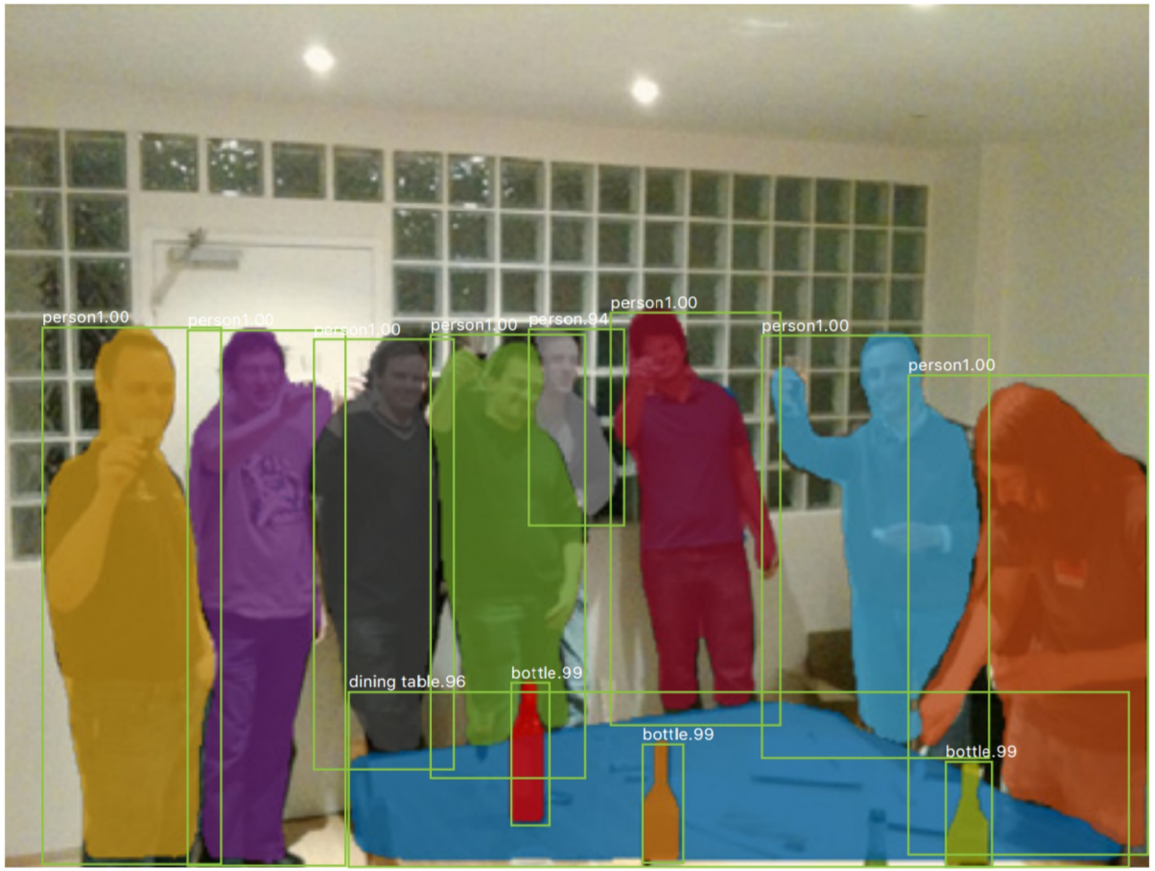







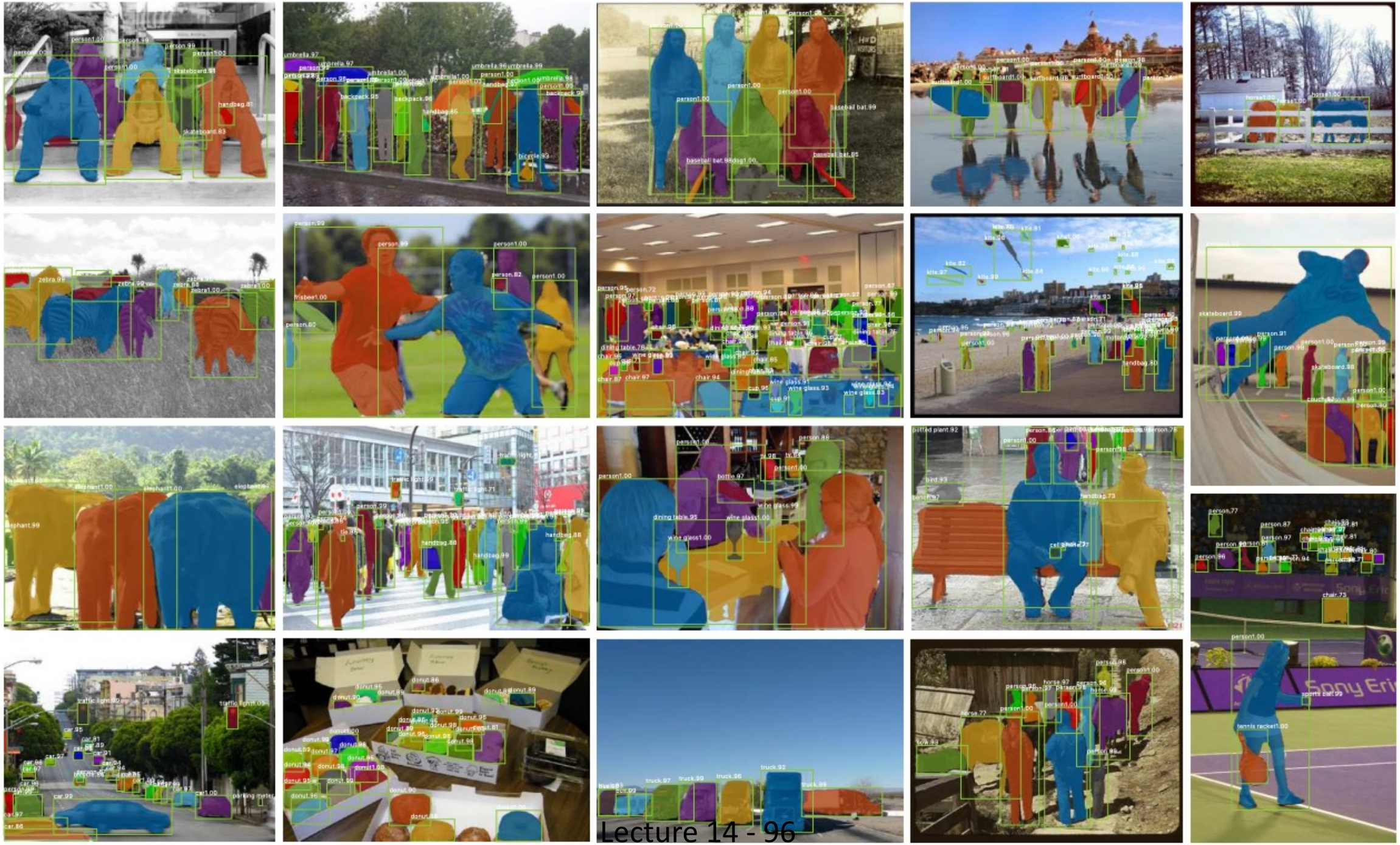
Mask R-CNN: Example Training Targets



Mask R-CNN: Very Good Results! Computer vision seems solved!



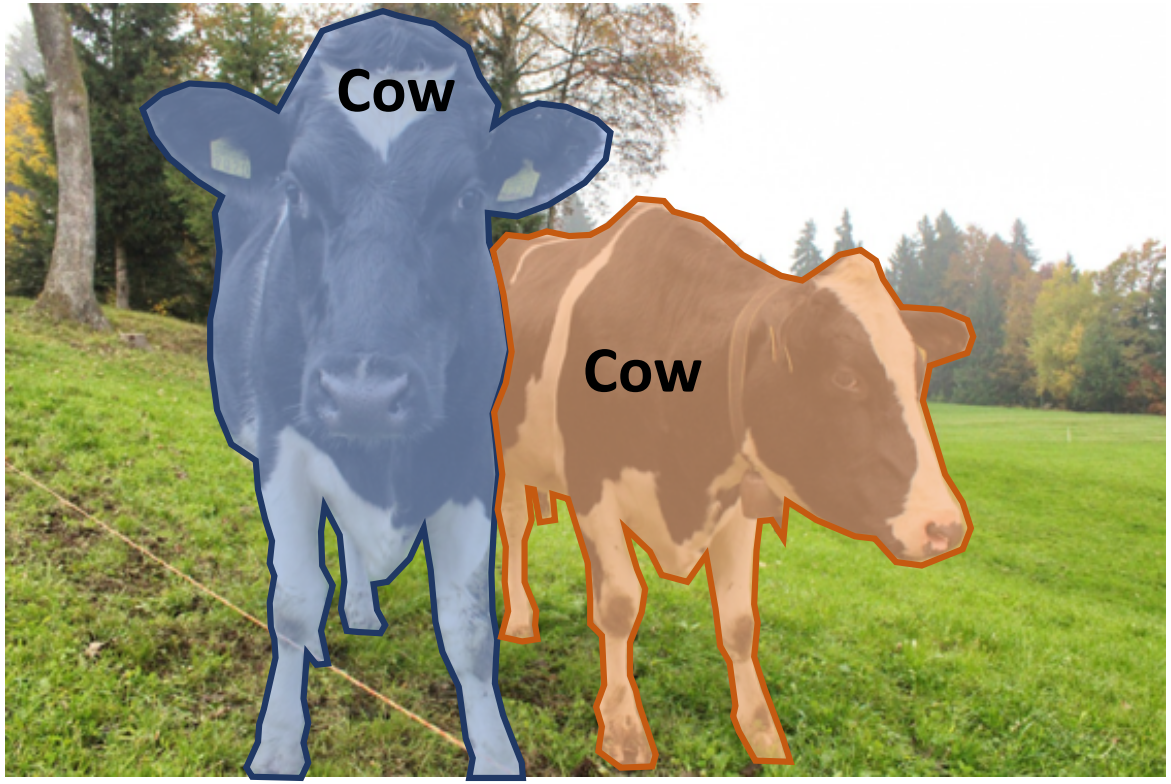
Mask R-CNN: by Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick. ICCV'17, Best Paper Award:
https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf



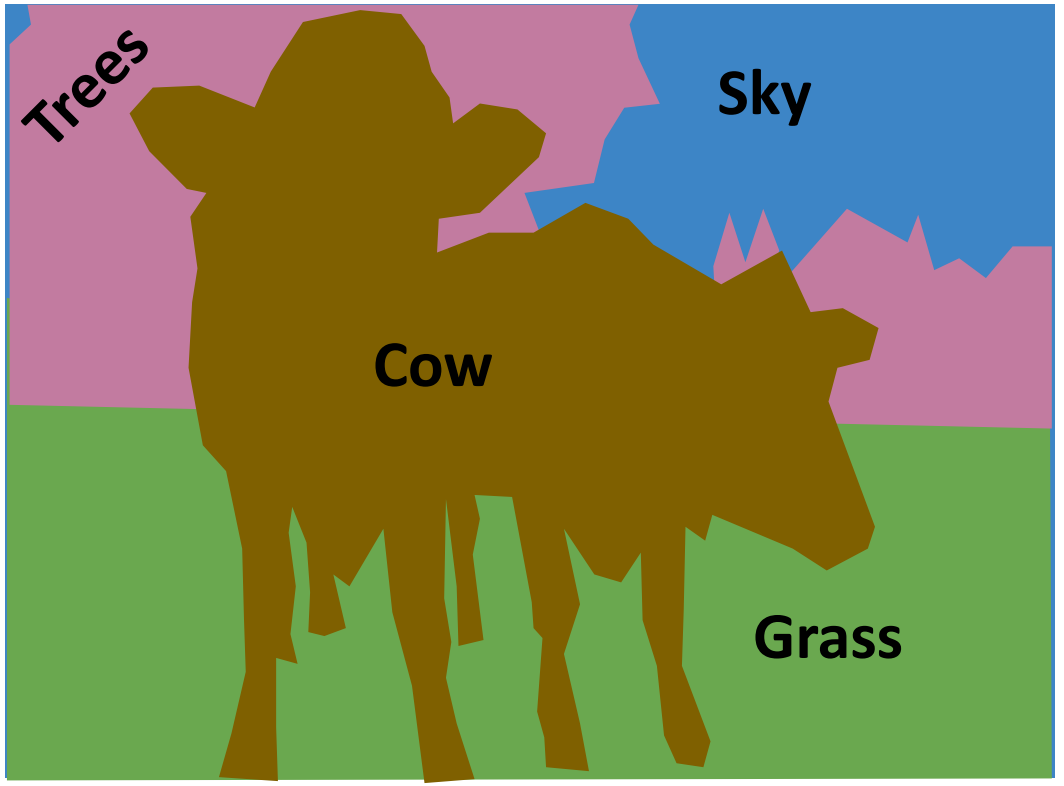
Lecture 14 - 97 https://www.youtube.com/watch?v=OOT3UIXZztE

Beyond Instance Segmentation

Instance Segmentation: Separate object instances, but only things



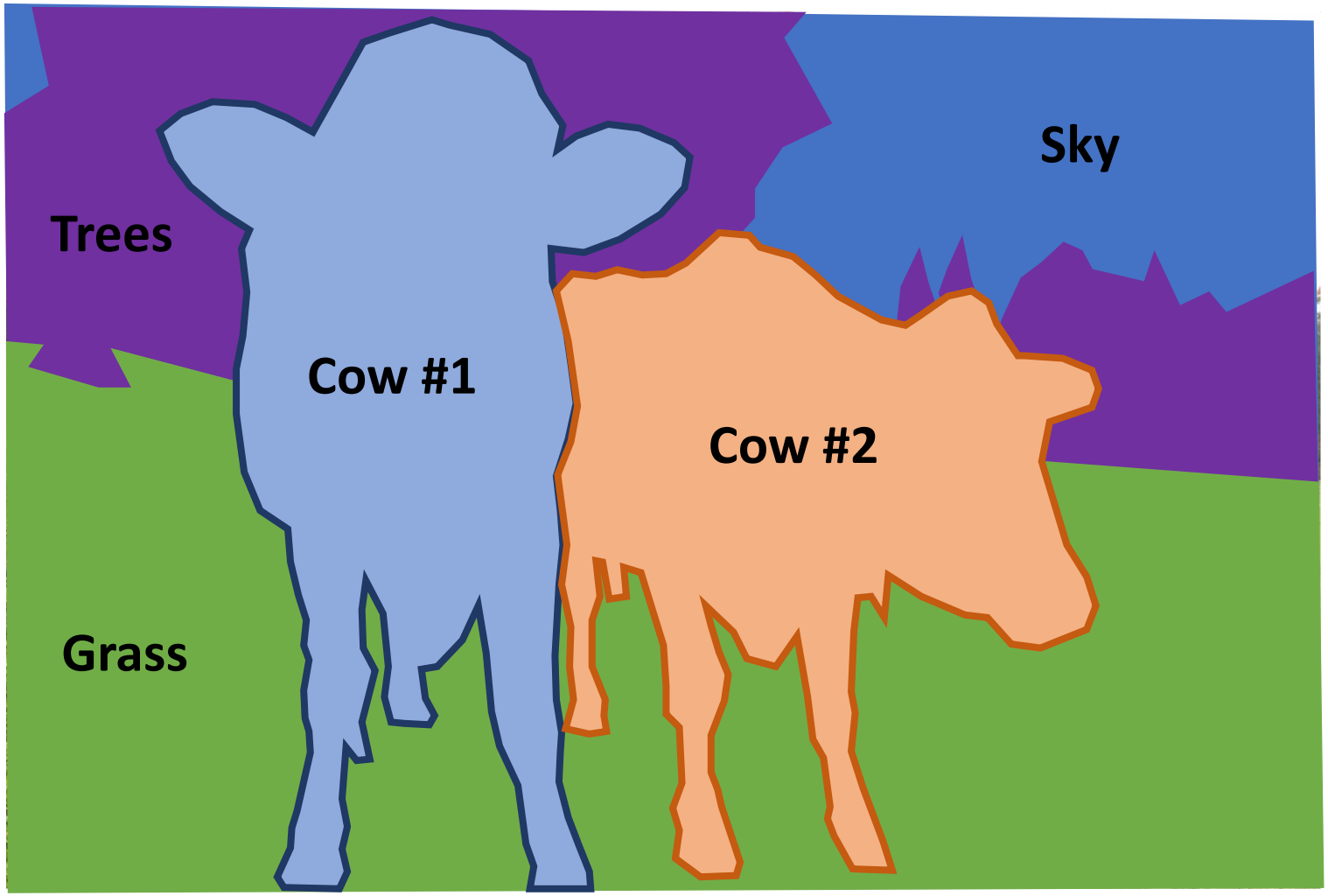
Semantic Segmentation: Identify both things and stuff, but doesn't separate instances



Beyond Instance Segmentation: **Panoptic Segmentation**

Label all pixels in the image (both things and stuff)

For "thing" categories also separate into instances



Kirillov et al, "Panoptic Segmentation", CVPR 2019
Kirillov et al, "Panoptic Feature Pyramid Networks", CVPR 2019

Beyond Instance Segmentation: **Panoptic Segmentation**

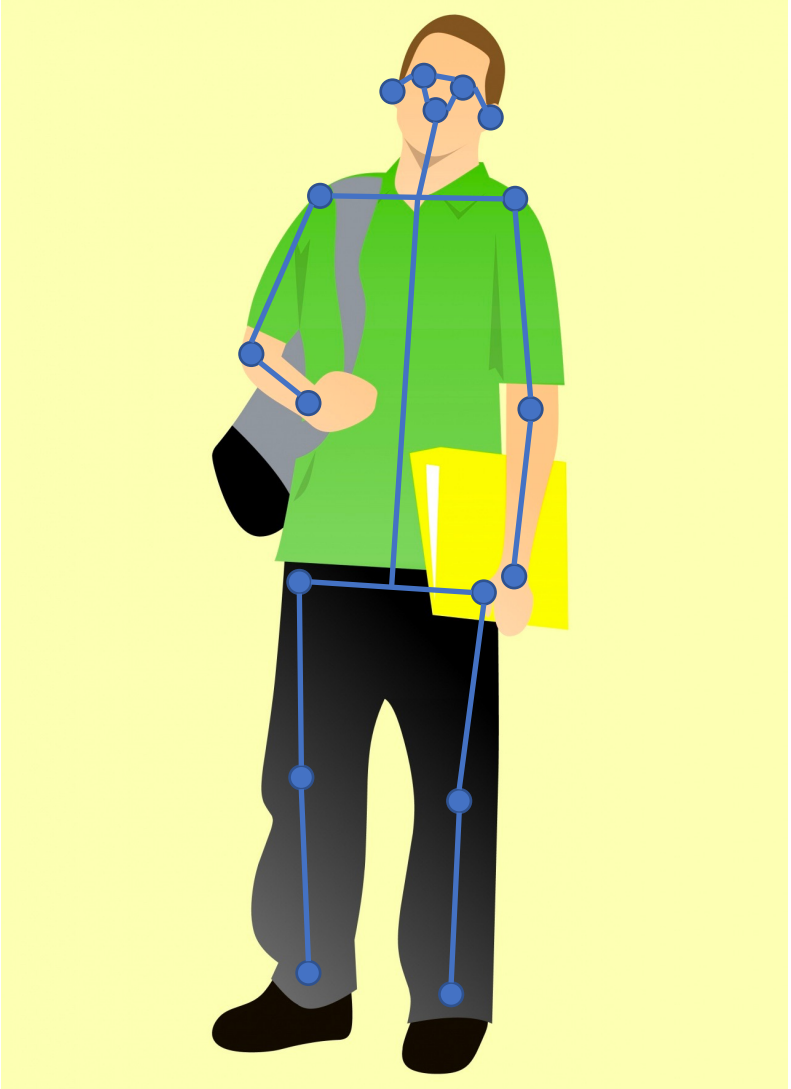
Kirillov et al, "Panoptic Feature Pyramid Networks", CVPR 2019

Beyond Instance Segmentation: Human Keypoints

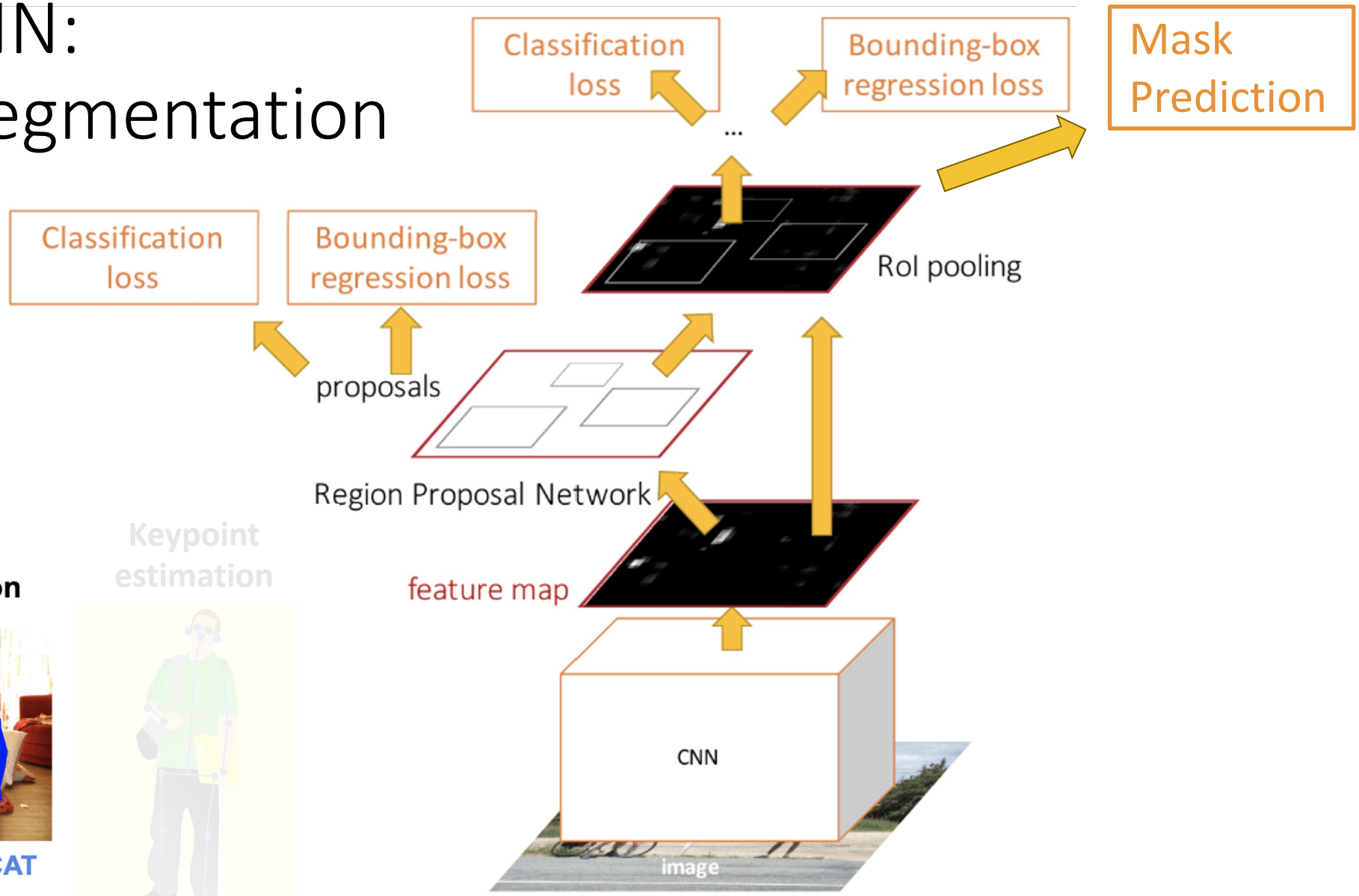
Represent the pose of a human
by locating a set of **keypoints**

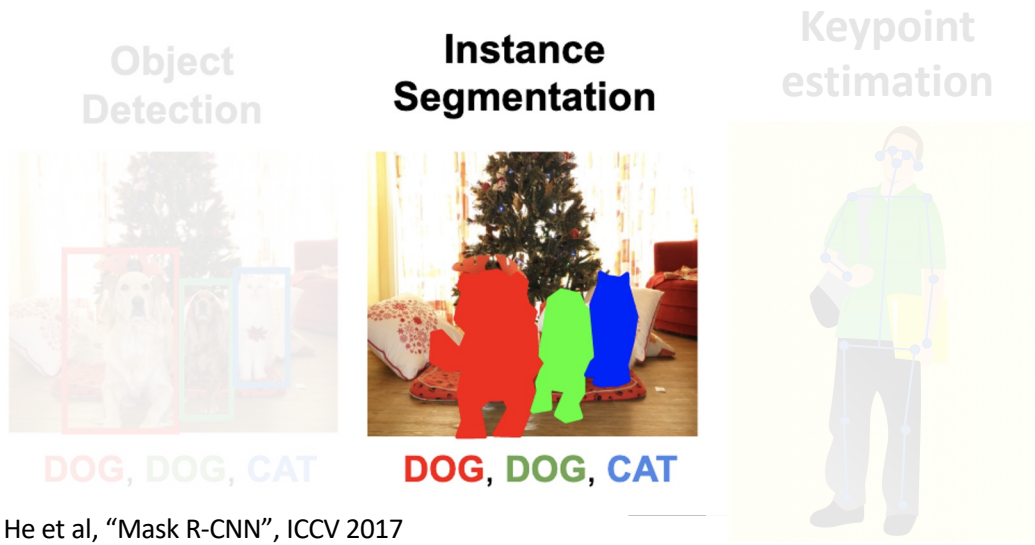
e.g. 17 keypoints:

- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle



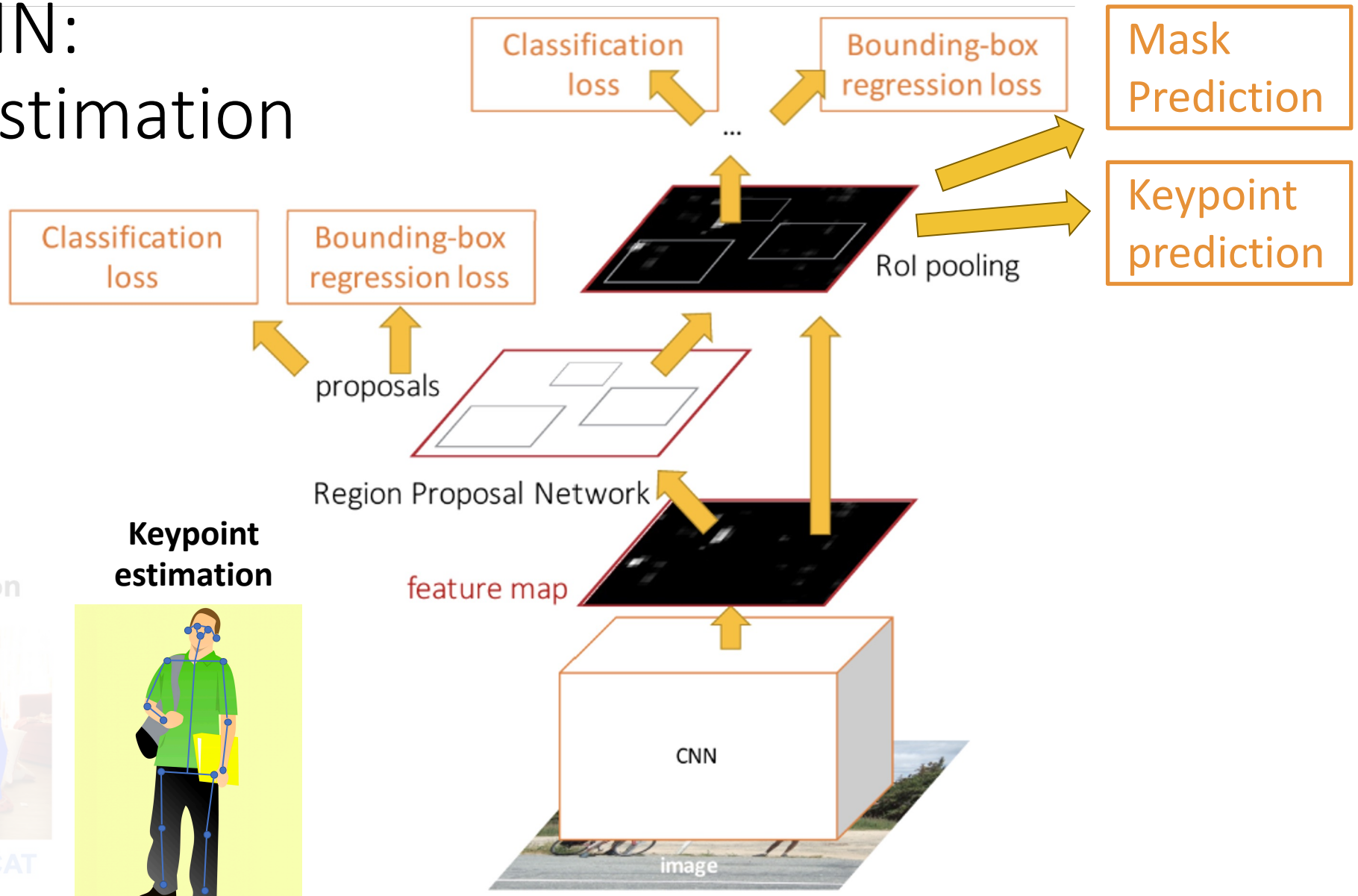
Mask R-CNN: Instance Segmentation





He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN: Keypoint Estimation



Object Detection

DOG, DOG, CAT

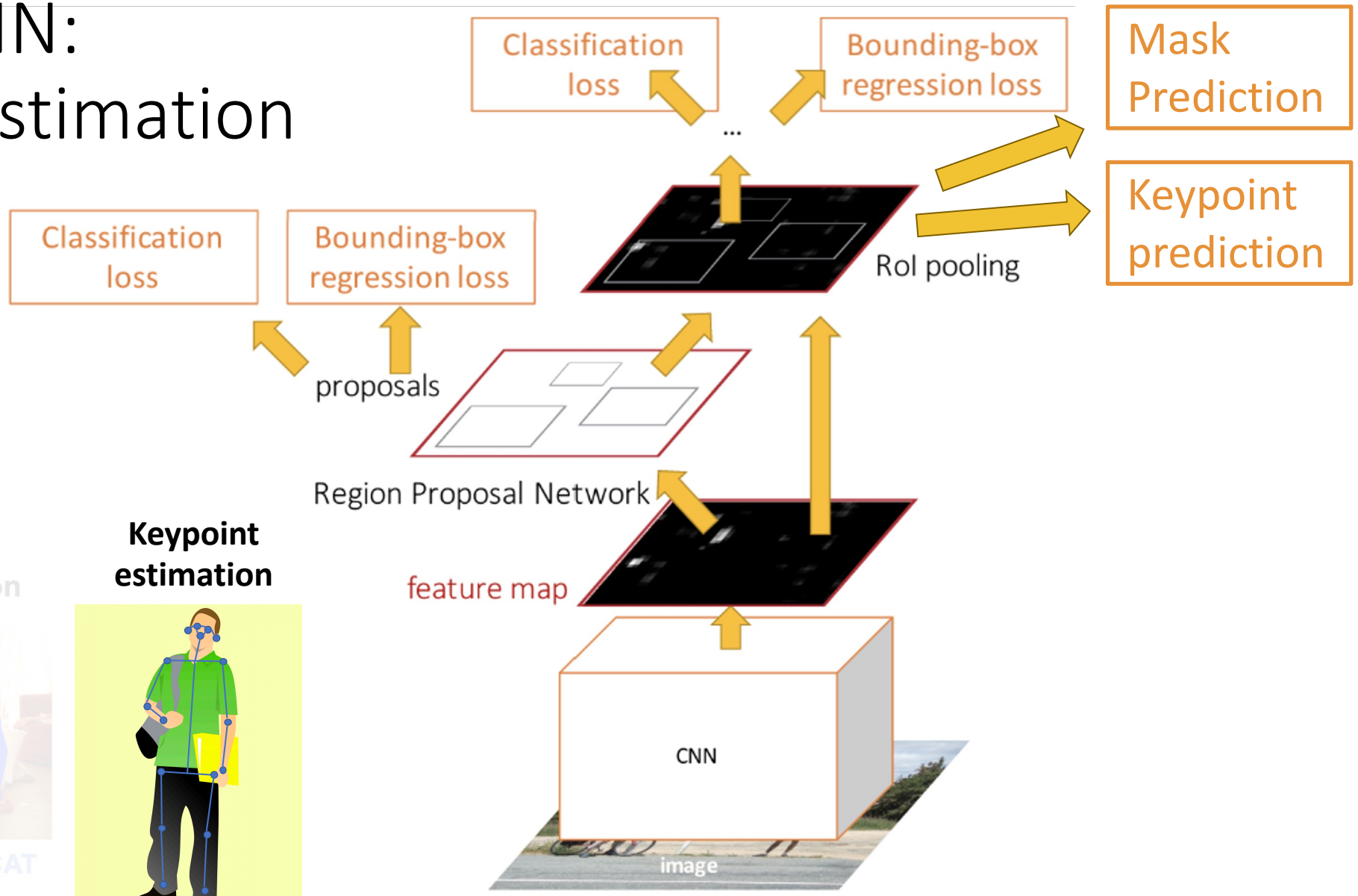
Instance Segmentation

DOG, DOG, CAT

Keypoint estimation

He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN: Keypoint Estimation



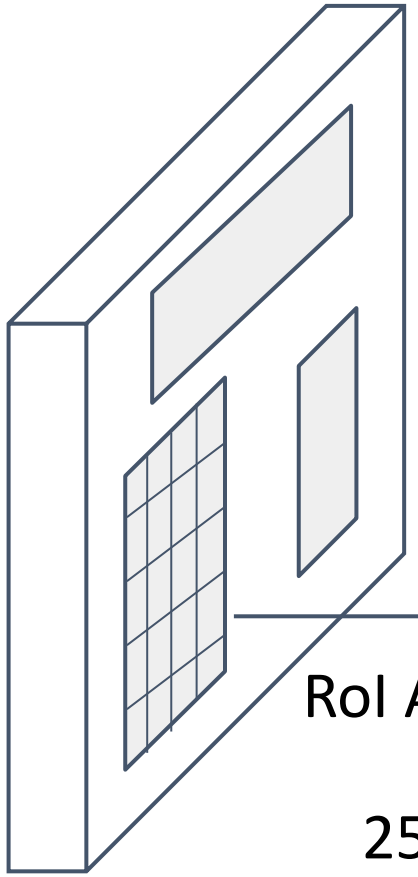
Keypoint estimation

He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN: Keypoints

Classification Scores: C
Box coordinates (per class): 4 * C
Segmentation mask: C x 28 x 28

CNN
+RPN

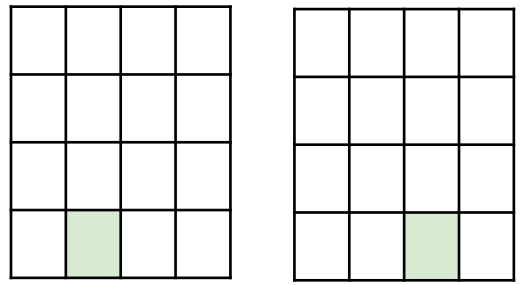


RoI Align

256 x 14 x 14

Conv…

One mask for each of
the K different keypoints



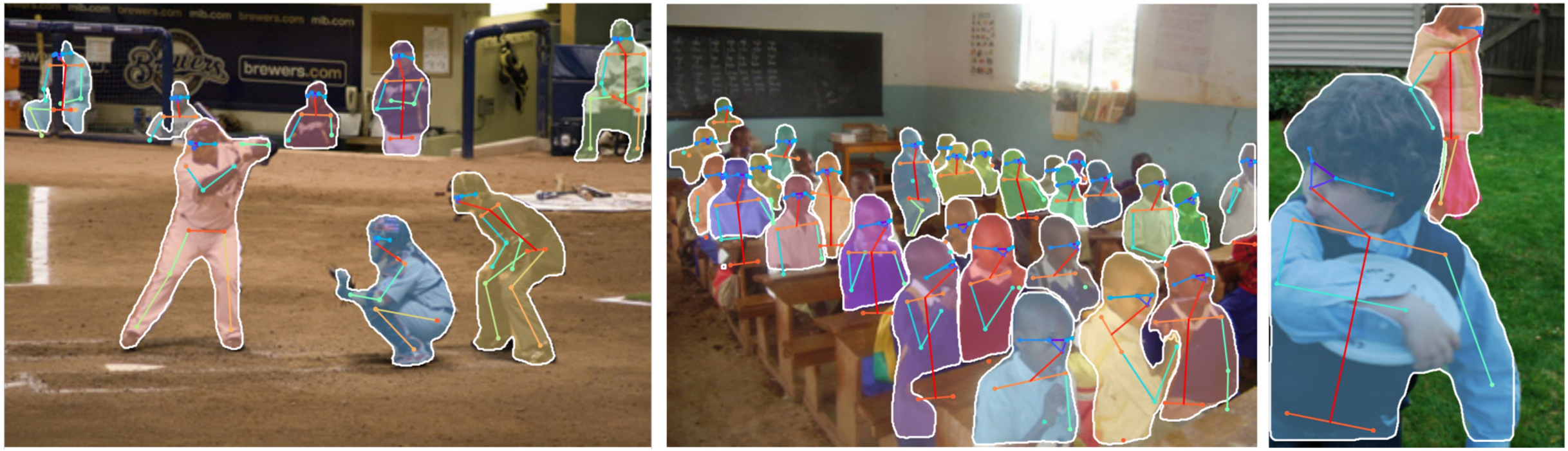
…

Keypoint masks:
K x 56 x 56

Ground-truth has one “pixel” turned on
per keypoint. Train with softmax loss

He et al, “Mask R-CNN”, ICCV 2017

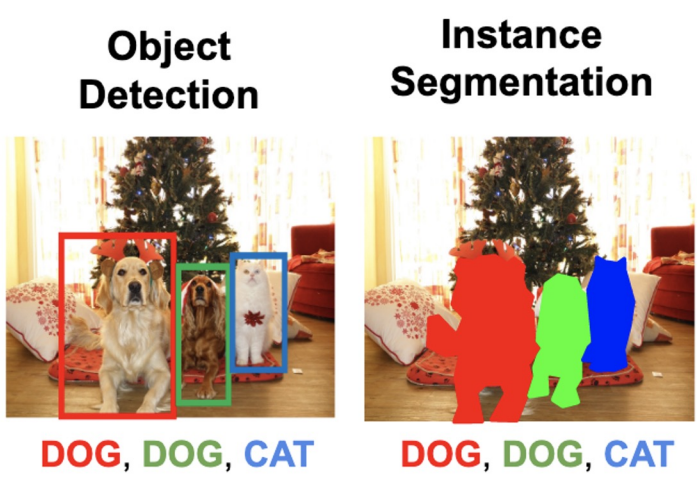
Joint Instance Segmentation and Pose Estimation



He et al, “Mask R-CNN”, ICCV 2017

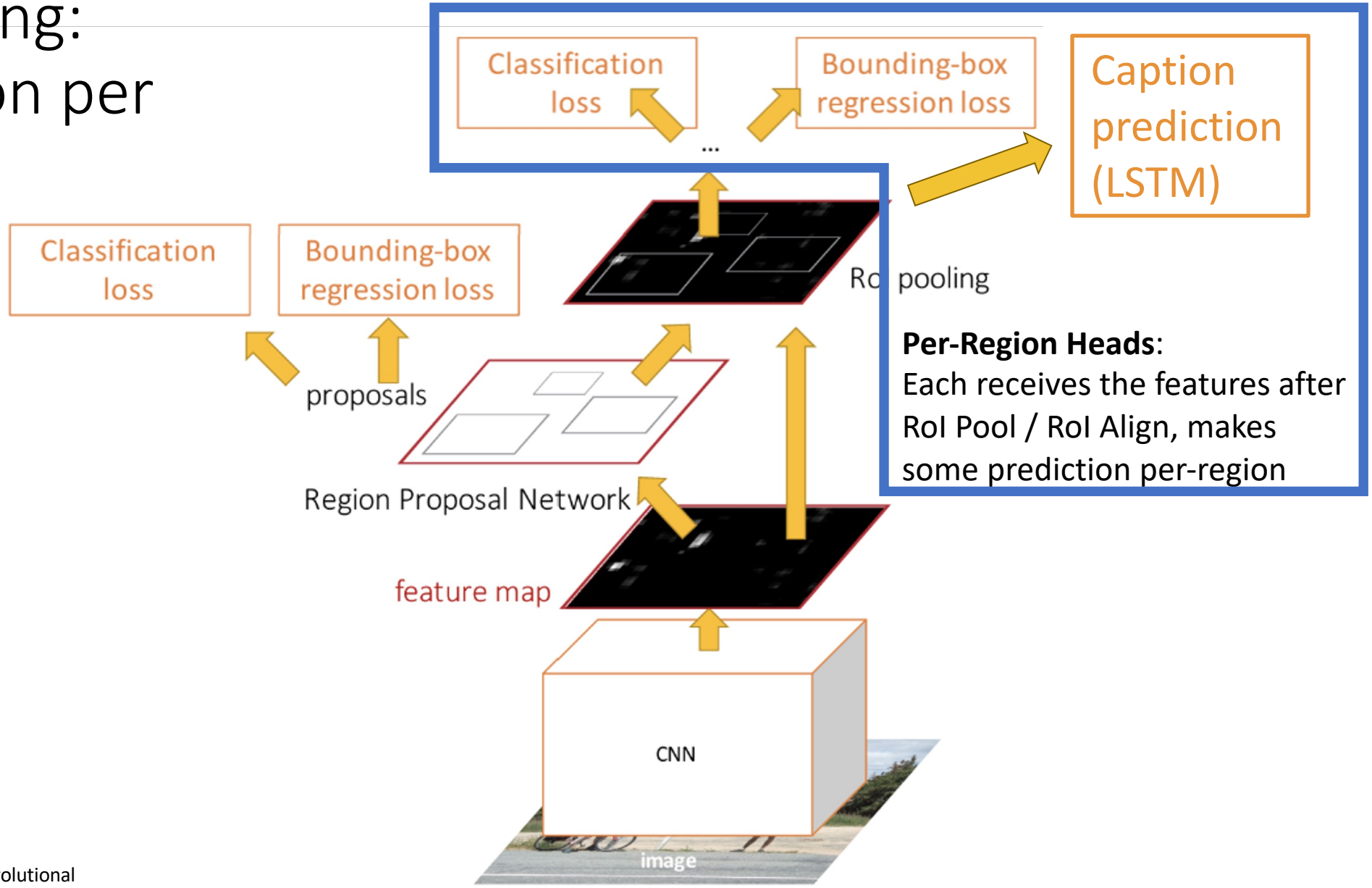
Mask R-CNN for pose estimation and instance segmentation demo video

General Idea: Add Per-Region "Heads" to Mask R-CNN!



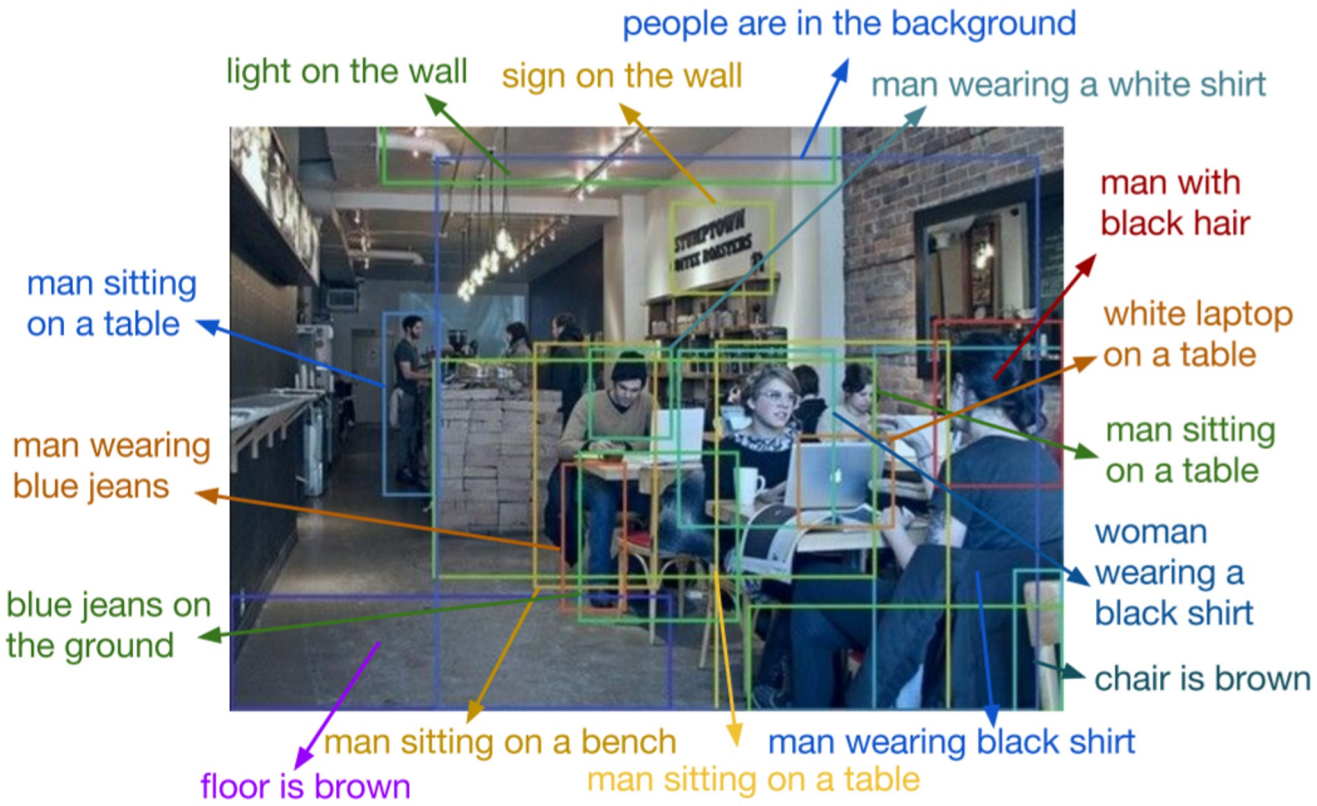
He et al, "Mask R-CNN", ICCV 2017

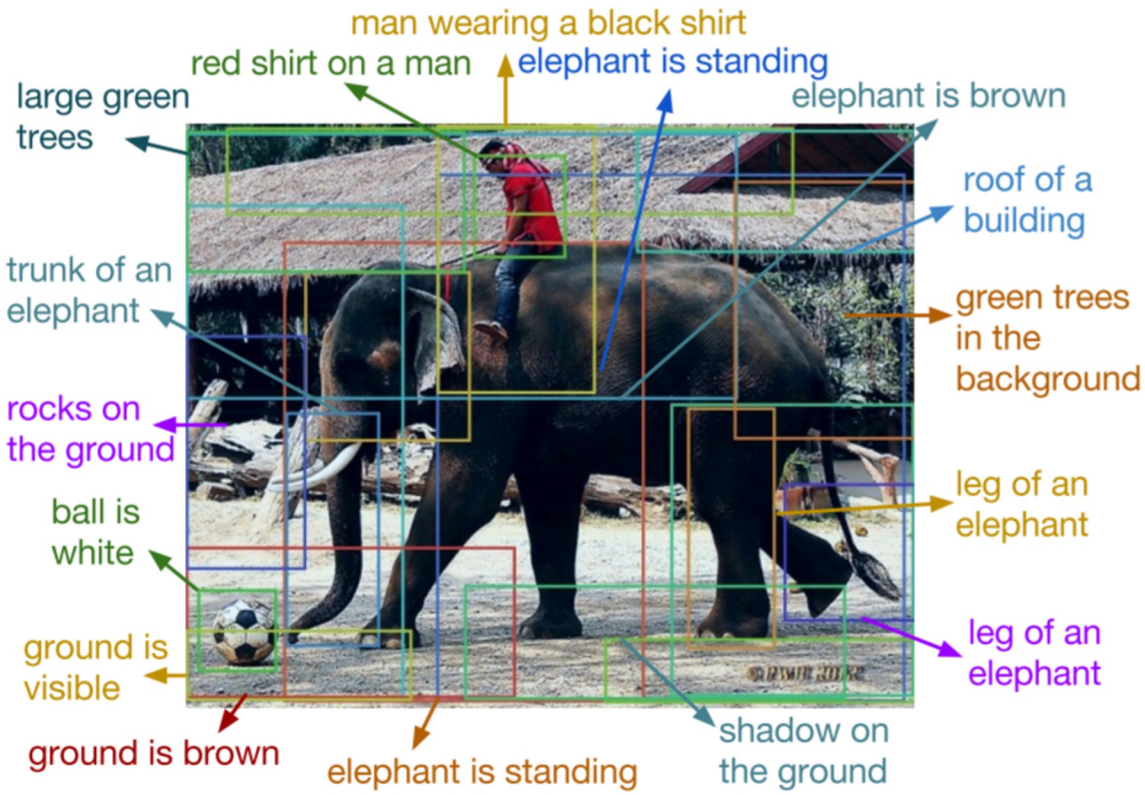
Dense Captioning: Predict a caption per region



Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016

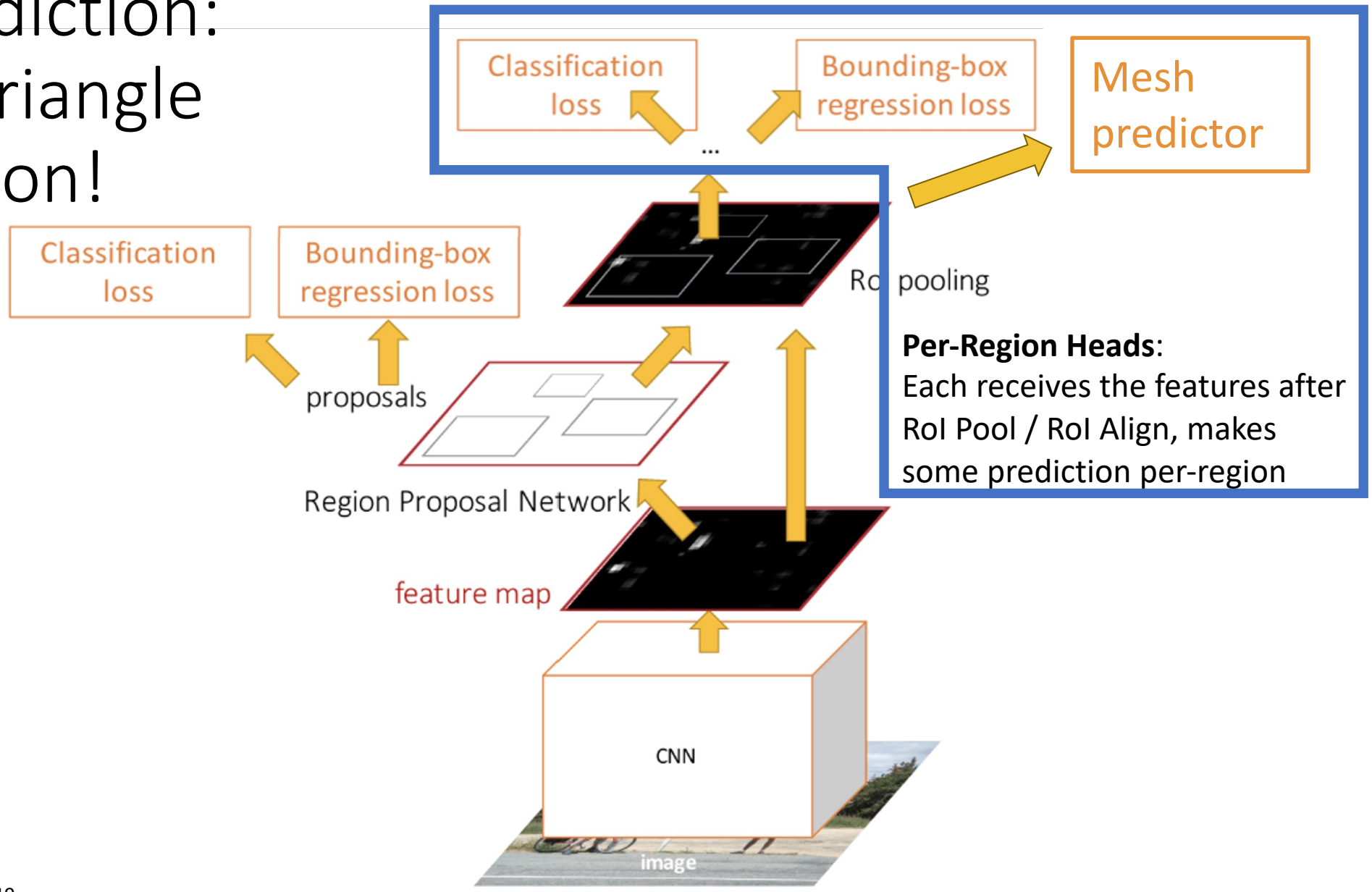
Dense Captioning





Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016

3D Shape Prediction: Predict a 3D triangle mesh per region!

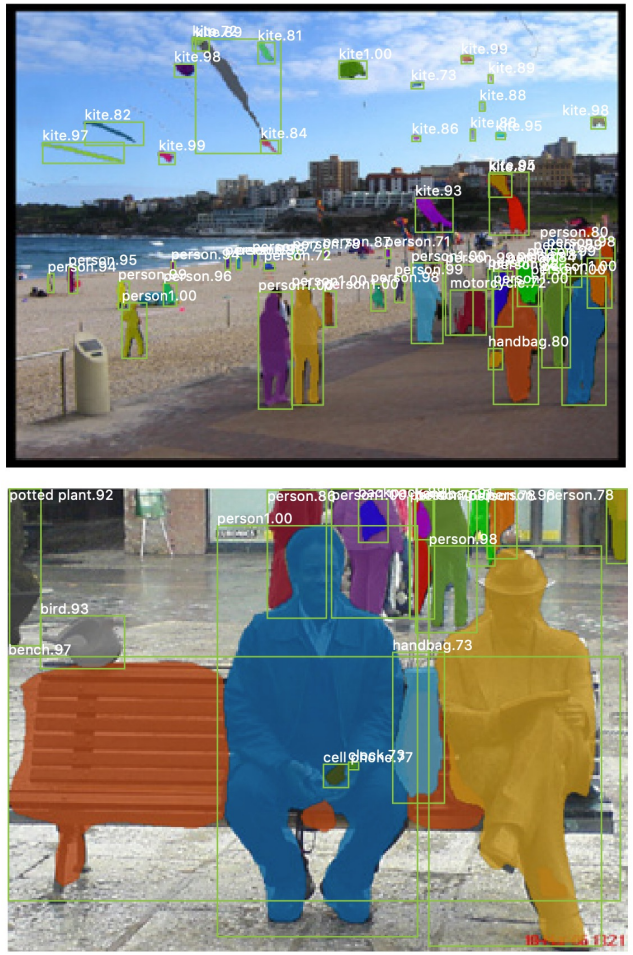


Per-Region Heads:
Each receives the features after RoI Pool / RoI Align, makes some prediction per-region

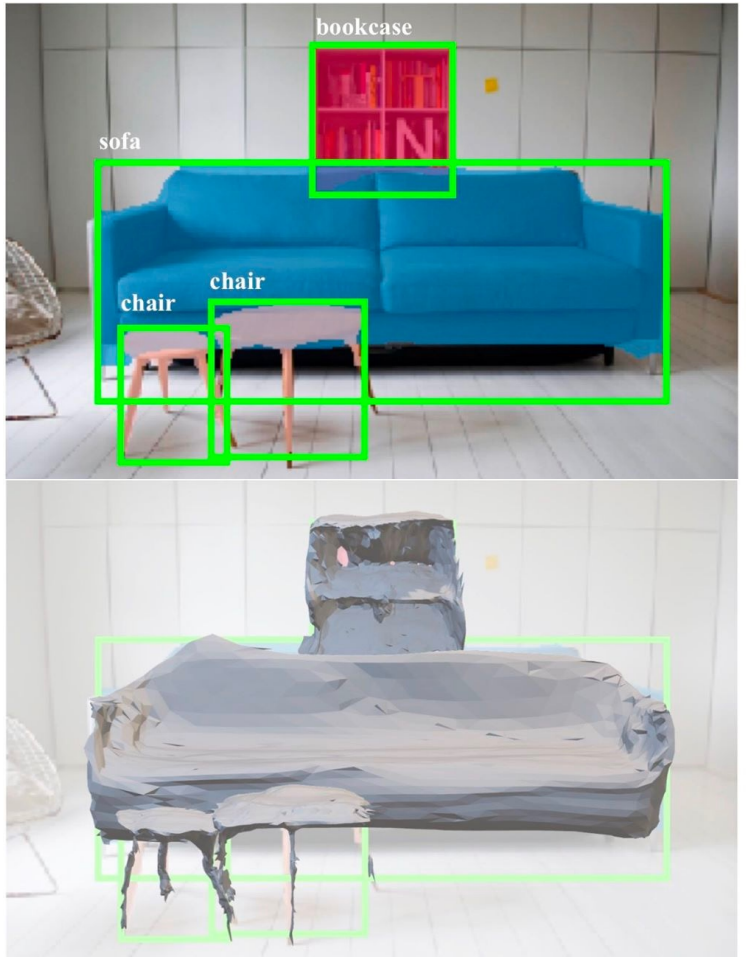
Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

3D Shape Prediction: Mask R-CNN + Mesh Head

Mask R-CNN:
2D Image -> 2D shapes



Mesh R-CNN:
2D Image -> **3D** shapes



More details next time!

He, Gkioxari, Dollár, and Girshick, "Mask R-CNN", ICCV 2017

Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

Toward a foundation model for image segmentation

Segment Anything (SAM)

https://arxiv.org/pdf/2304.02643.pdf

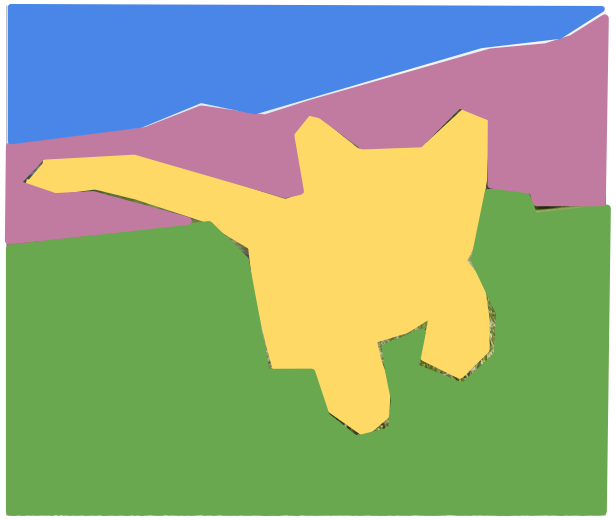
Summary: Many Computer Vision Tasks!

Classification

CAT

No spatial extent

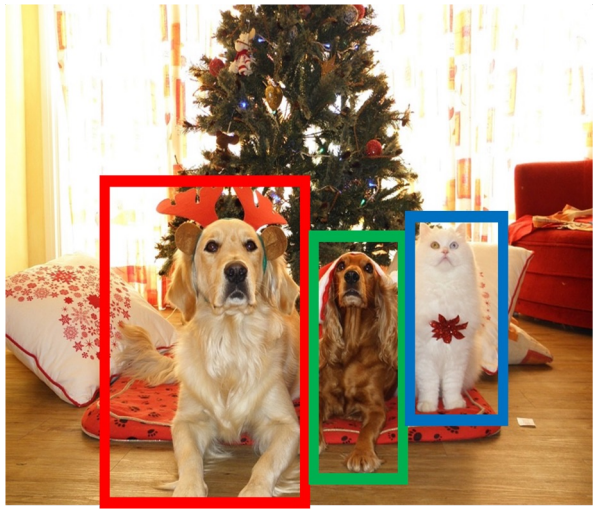
Semantic Segmentation



GRASS, **CAT**, **TREE**, **SKY**

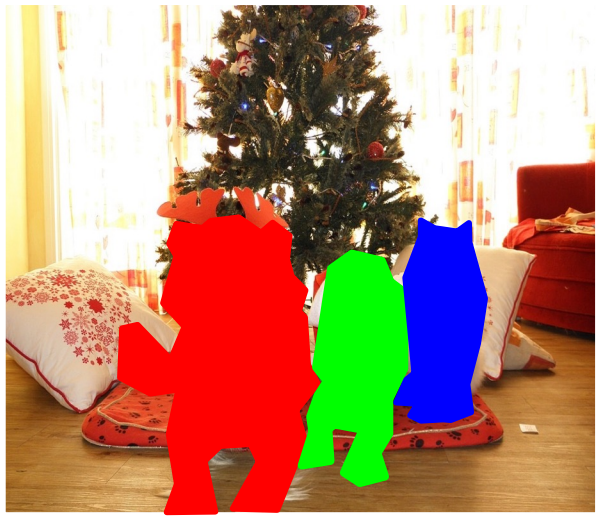
No objects, just pixels

Object Detection



DOG, **DOG**, **CAT**

Instance Segmentation



DOG, **DOG**, **CAT**

Multiple Objects

Reading and Practice

Textbook D2L:

Chapter 14.8.4 Mask R-CNN

Chapter 14.9 Semantic Segmentation and the Dataset

Chapter 14.10 Transposed Convolution

Chapter 14.11 Fully Convolutional Networks