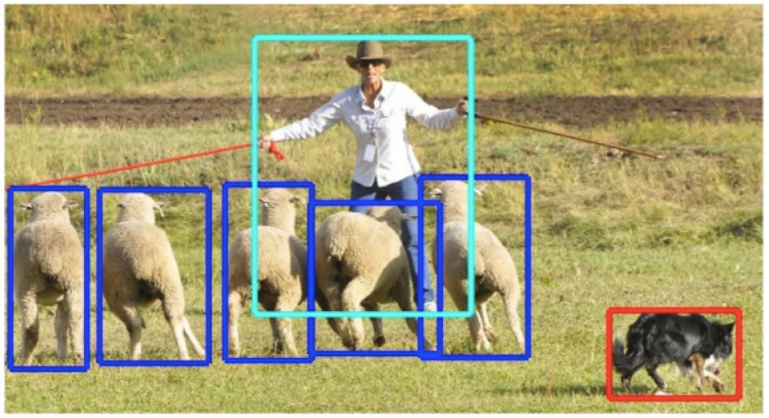
Semantic Segmentation and Dense Labeling

Adapted from: Stanford CS231 , Illinois cs498 , Jeremy Jordan

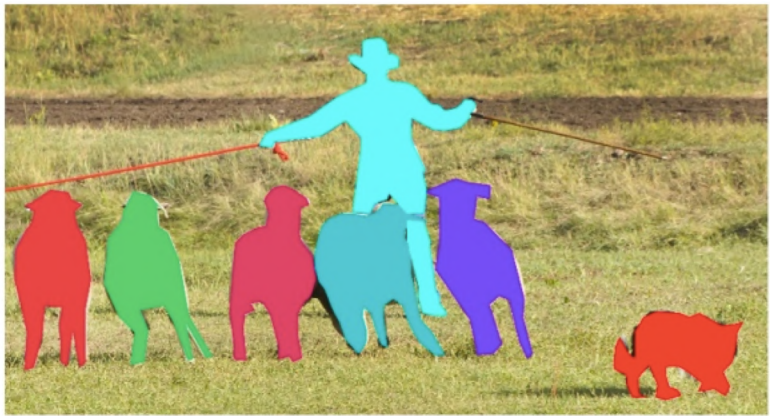
Computer Vision Tasks

image classification



object detection

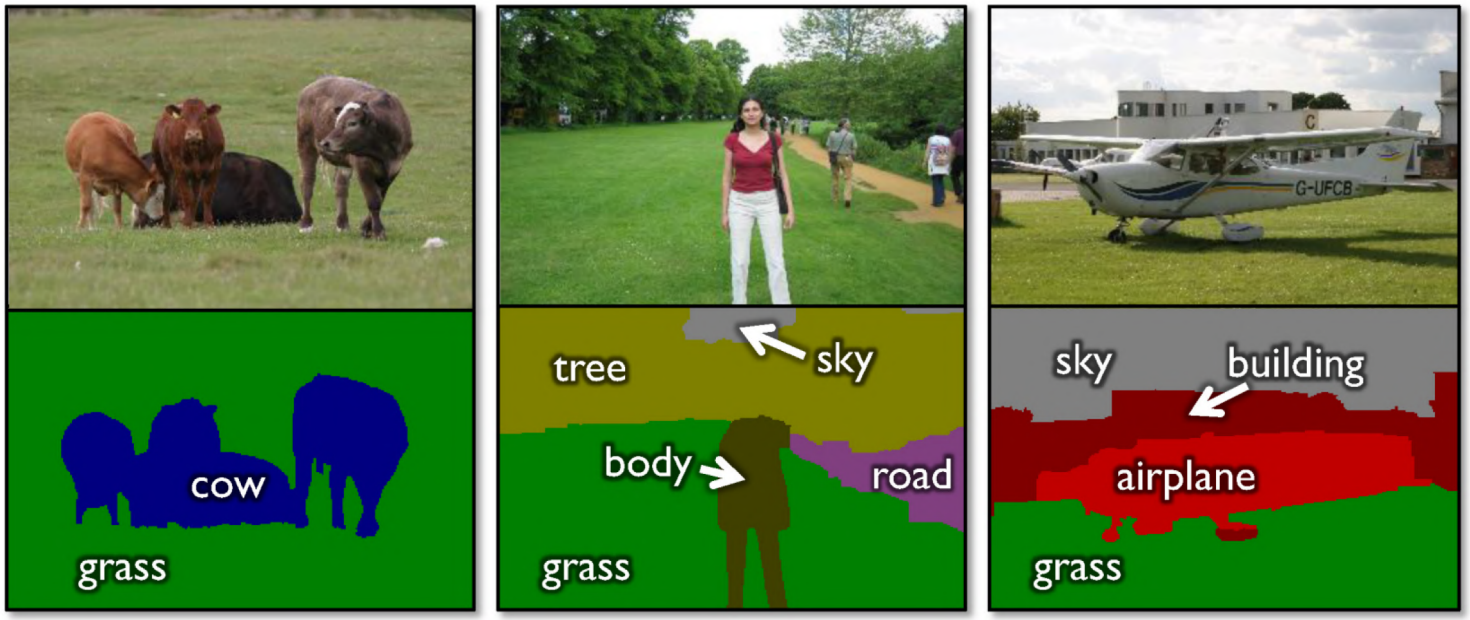
semantic segmentation



instance segmentation

Semantic Segmentation

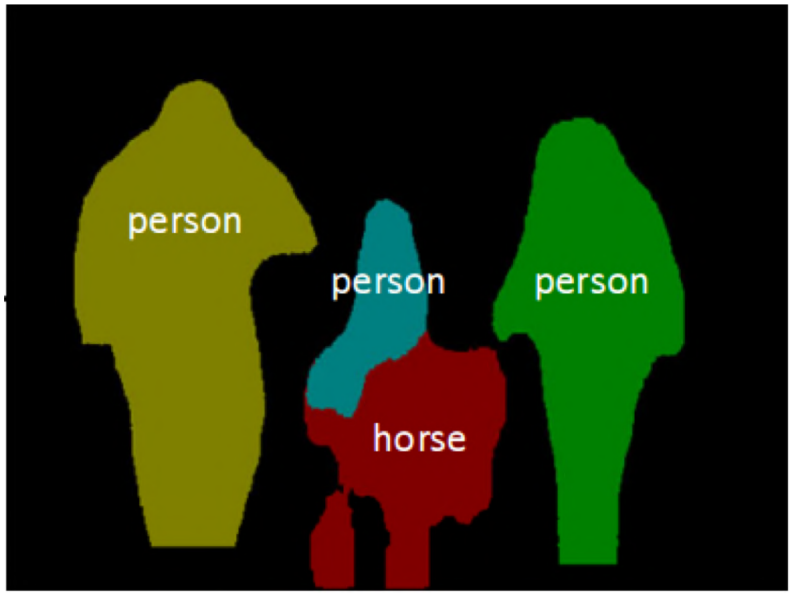
- Label every pixel! (also called: dense labeling)
- Don’t differentiate instances (cows) only care about the category of each pixel.



source: Stanford CS231

Instance Segmentation

- Detect instances. Distinguish between separate objects of the same category.
- “simultaneous detection and segmentation” (SDS)



source: Stanford CS231

Things vs Stuff

THINGS

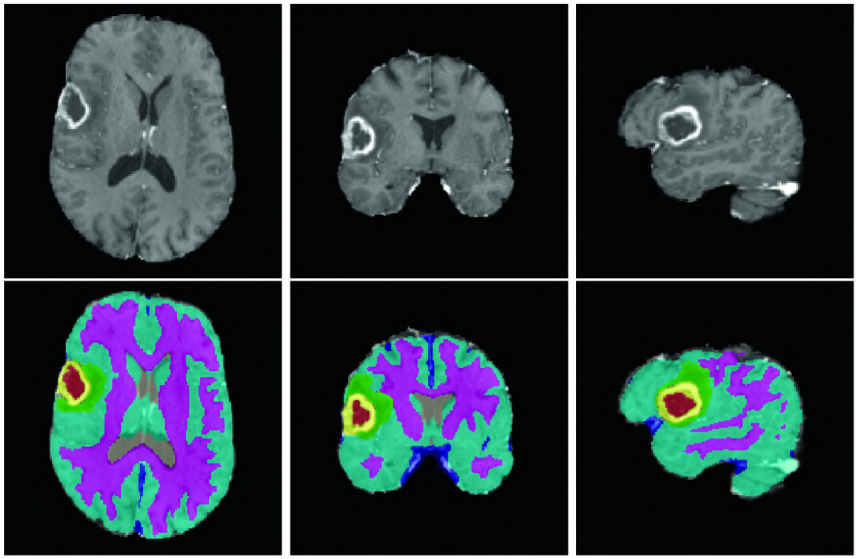
- Person, cat, horse, etc
- A spatially coherent shape
- Individual instances with separate identity

STUFF

- Road, grass, sky etc
- Amorphous, no shape
- No notion of instances
- Can be done at pixel level
- Might be scattered

Segmentation Usage

- Scene understanding
- Autonomous vehicles
- Medical image diagnostics
- Aerial images
- Robotics
- Agriculture



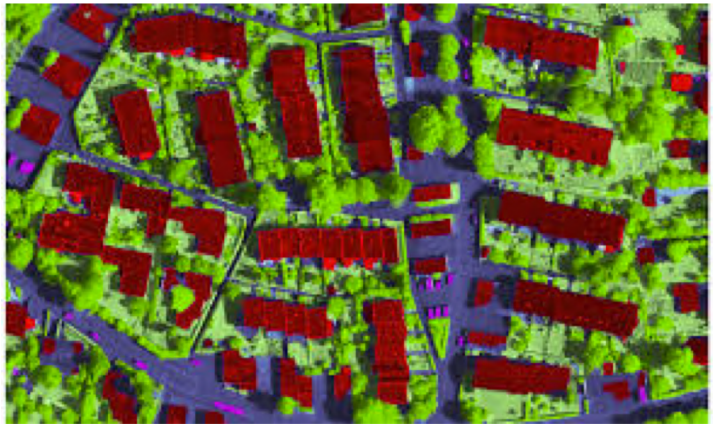


Image source

Representing the Task

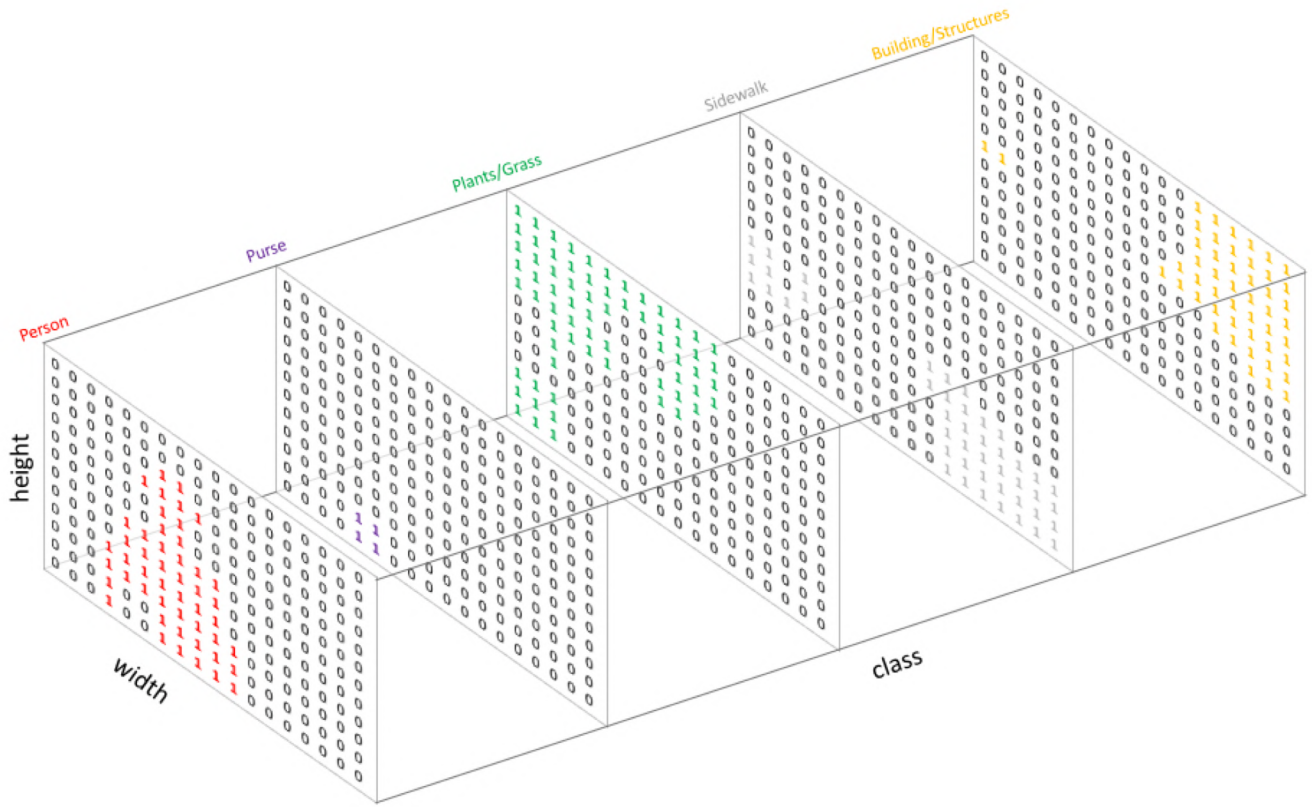
- **Input**: RGB or grayscale image HxWx3
- **Output**: segmentation map HxWx1

0: Background/Unknown
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

Image source

Representing the Task

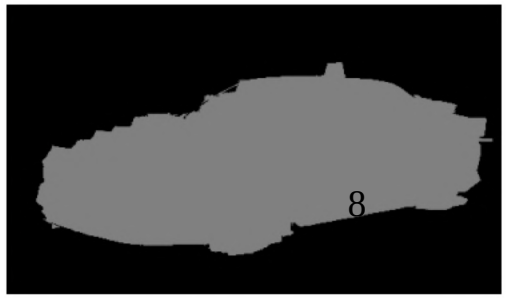
- **Input**: RGB or grayscale image HxWx3
- **Output**: One-hot encoding maps HxWxk



[Image source]

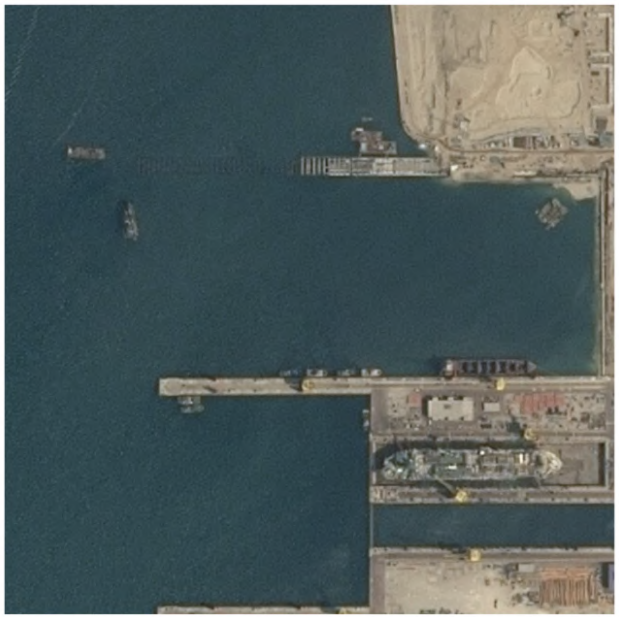
Evaluation Metric

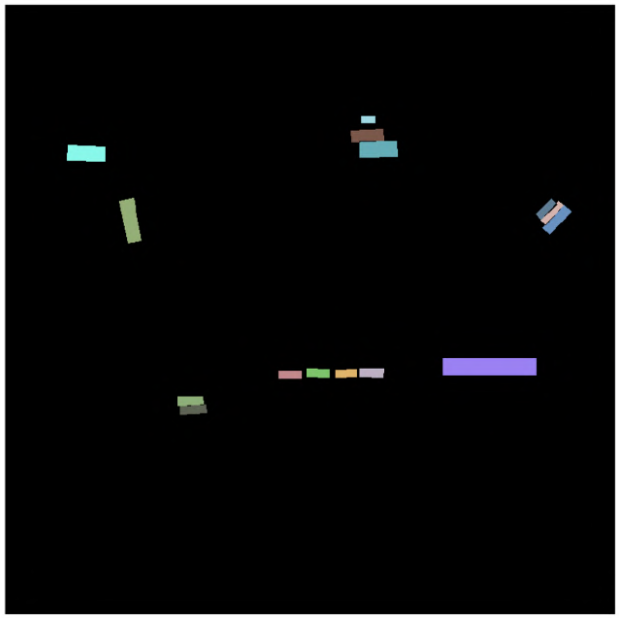
- Pixel classification!
- *Intersection over Union*
  - IoU for a class. Averaged across classes and images
- Pixel Accuracy?
  - Heavily imbalanced classes
  - Common classes are over-emphasized



Evaluation Metric: Pixel Accuracy

Ground Truth

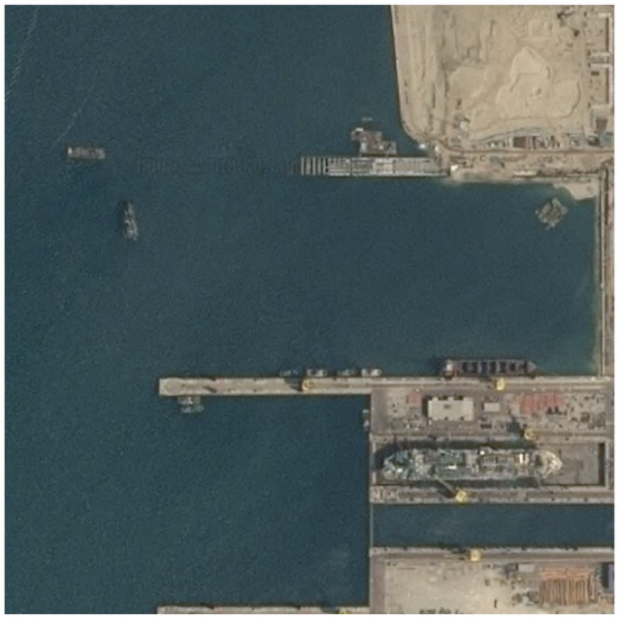


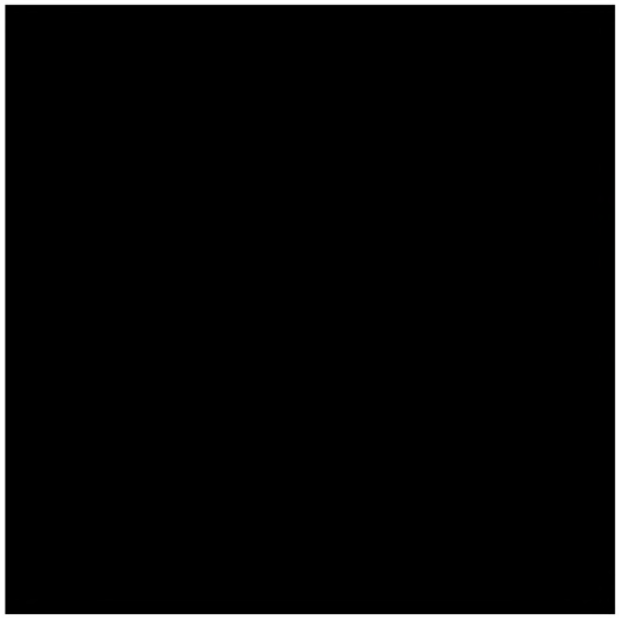


- Segment ships in a satellite image [Image source]
- **Evaluate Pixel accuracy**: % of pixels that are classified correctly.
- Your segmentation accuracy is 95%.
- That's awesome! Let's see how your segmentation looks like!

Evaluation Metric: Pixel Accuracy

Your segmentation results





- Segment ships in a satellite image [Image source]
- **Evaluate Pixel accuracy**: % of pixels that are classified correctly.
- Your segmentation accuracy is 95%.
- That's awesome! Let's see how your segmentation looks like!

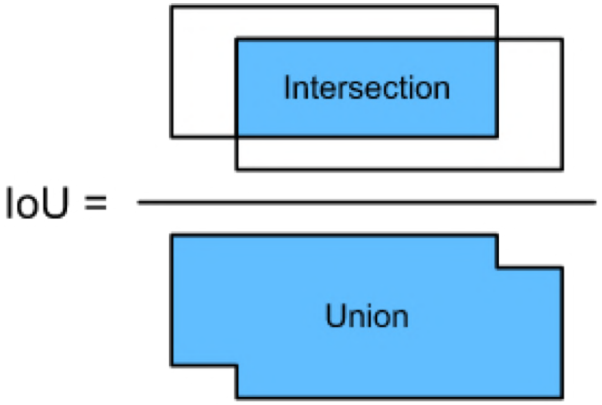
Evaluation Metric: Mean IoU

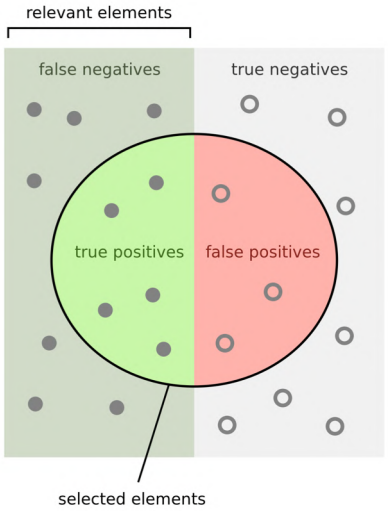
Mean IoU

- **Per-class IoU**: an intersection of the predicted and true sets of pixels for a given class, divided by their union

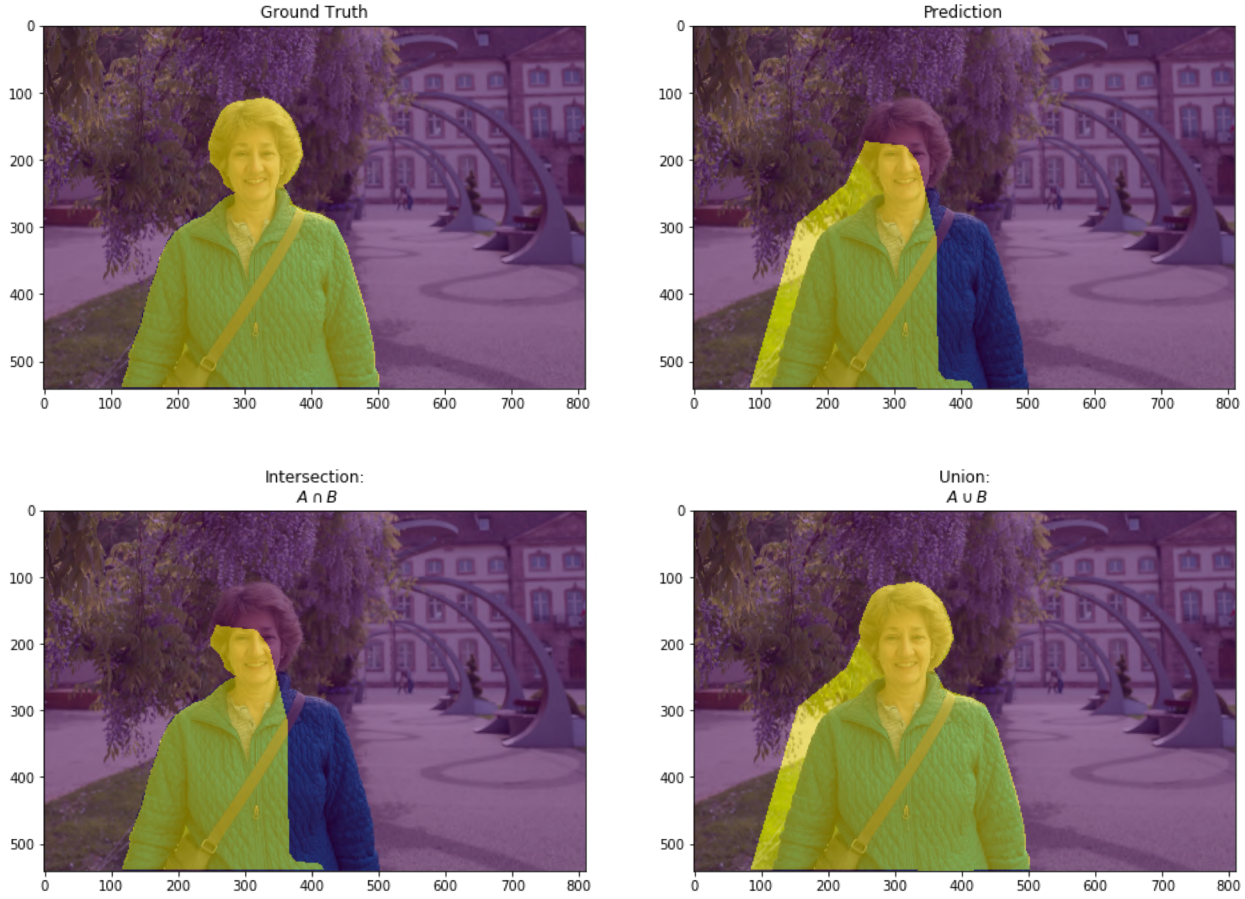
$$IoU = \frac{A \cap B}{A \cup B} = \frac{TP}{TP + FP + FN}$$

- **mIoU:** Mean IoU over all classes





Evaluation Metric: Mean IoU



Evaluation Metric: Mean IoU

$$mean\_IoU = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} IoU(c)$$

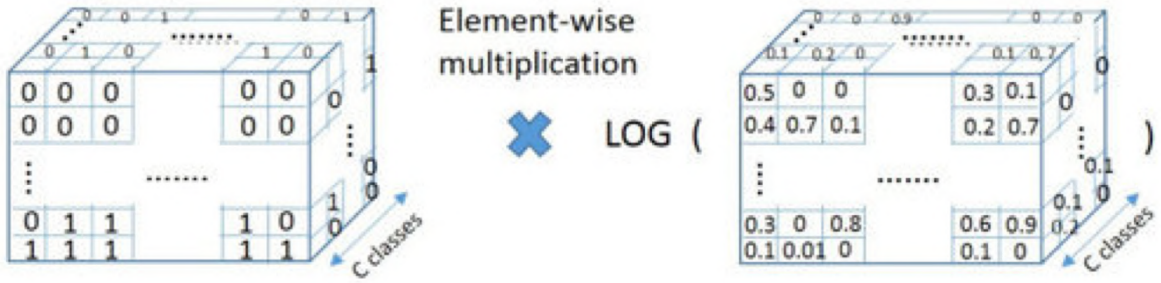
- Mean-IoU compensates for imbalanced classes.
- What is the mIoU of the ship segmentation (assuming 5 out of 100 pixels are the ships)?

class	mean	bg																				
acc	69.6	91.9	85.6	37.3	83.2	62.5	66	85.1	80.7	84.9	27.2	73.3	57.5	78.1	79.2	81.1	77.1	53.6	74	49.2	71.7	63.3

Loss Functions: pixel CE

- The most commonly used: **pixel-wise cross entropy**.
- For each pixel we calculate the CE (cross entropy):

$$-\sum_{i,j}\sum_{k\in C} y_k(i,j)\ \log \hat{y}_k(i,j)$$



One-hot encoded label

Softmax output of CNN

Loss Functions: pixel CE

- The most commonly used: **pixel-wise cross entropy**.
- For each pixel we calculate CE:

$$-\sum_{i,j}\sum_{k \in \mathcal{C}} y_k(i,j)\ \log \hat{y}_k(i,j)$$

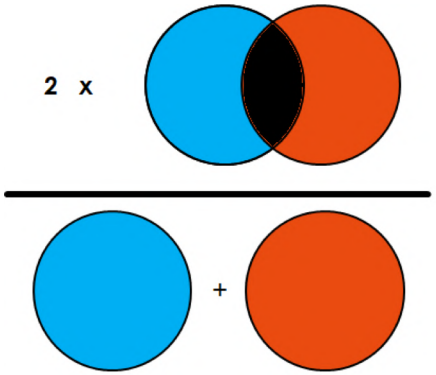
- we're asserting equal learning to each pixel in the image.
- This can be a problem if your various classes have unbalanced representation in the image.
- We can weight this loss for each **output channel** in order to counteract a class imbalance.

Loss Functions: Dice Coefficient

- Another popular loss: **Dice Coefficient / F1**.
- Dice coefficient was originally developed for two sets:

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|} = \frac{2TP}{2TP + FP + FN}$$

- The term $|A \cap B|$ represents the common elements between sets A and B, and $|A| + |B|$ represents the number of elements in sets A and B.



See here a discussion about the difference between IoU and Dice/F1 scores

Loss Functions: Dice Coefficient

- **Dice Loss** is defines as:

$$loss(class_k) = 1 - Dice = 1 - \frac{2\ |A \cap B|}{|A| + |B|}$$

- Soft Dice loss:

$$loss(class_k) = 1 - \frac{2\ \sum_{pixels}\ y_k\, \hat{y}_k}{\sum_{pixels}\ y_k^2 + \sum_{pixels}\ \hat{y}_k^2}$$

- The numerator is concerned with the *common activations* between our prediction and target mask.
- The denominator is concerned with the quantity of activations in each mask (class) *separately*.
- This has the effect of normalizing our loss according to the size of the target mask.

Loss Functions: Dice Coefficient

- In our case: $|A \cap B|$

$$|A \cap B| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{element-wise multiply}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

prediction target

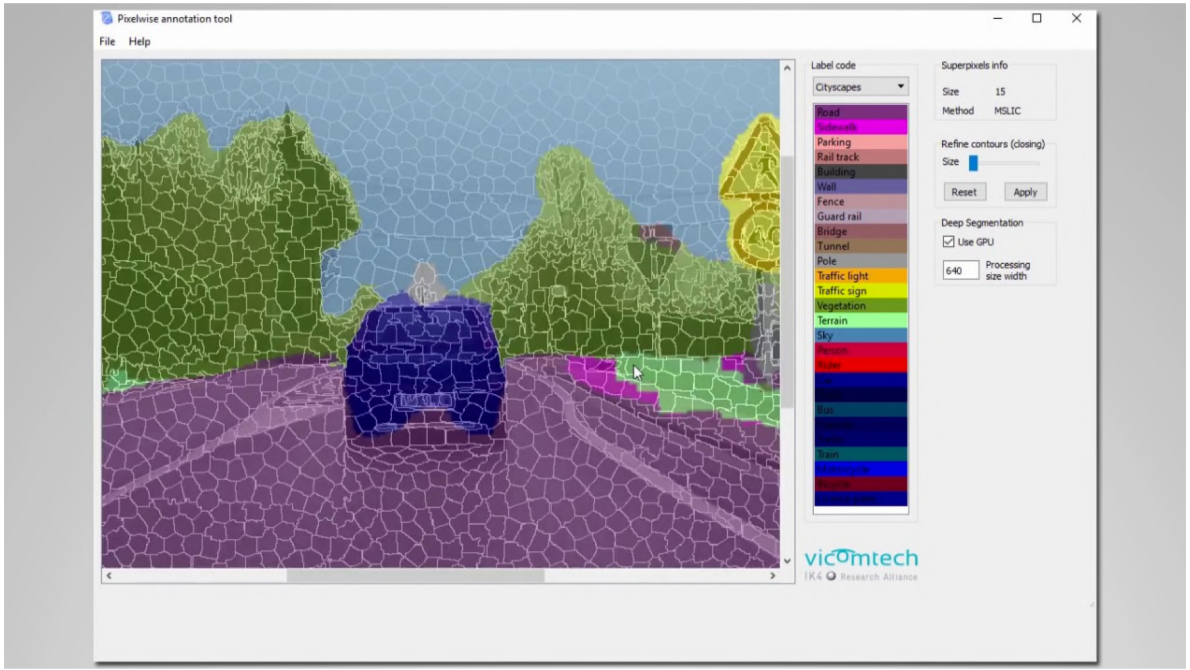
- And $|A| + |B|$

$$|A| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix}^{2\ (optional)} \xrightarrow{\text{sum}} 7.82$$

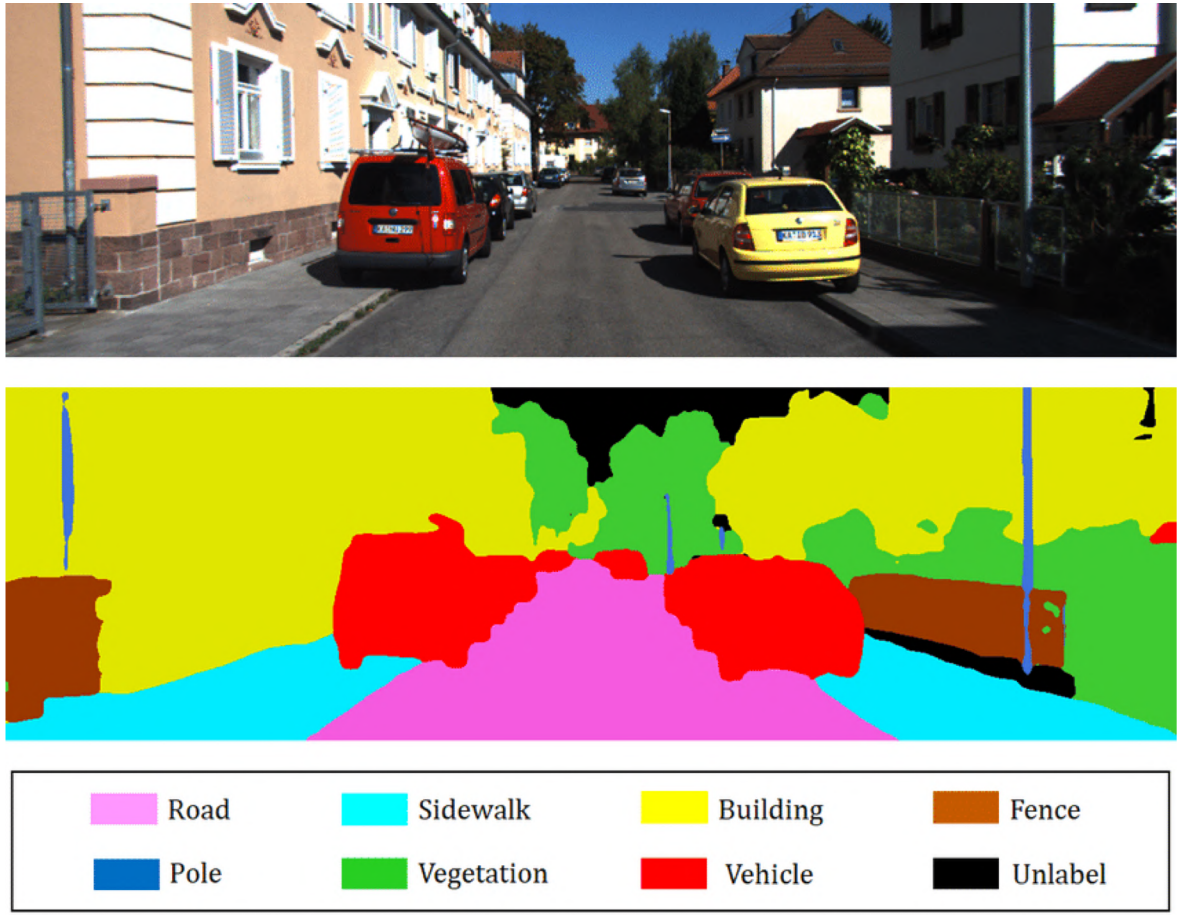
$$|B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{2\ (optional)} \xrightarrow{\text{sum}} 8$$

Challenges in data collection

- Precise localization is hard to annotate
- Annotating every pixel leads to heavy tails
- Common solution: annotate few classes (often things), mark rest as “Other”
- Assist classical tools for over segmentation (super-pixels).
- Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)



Semantic Segmentation

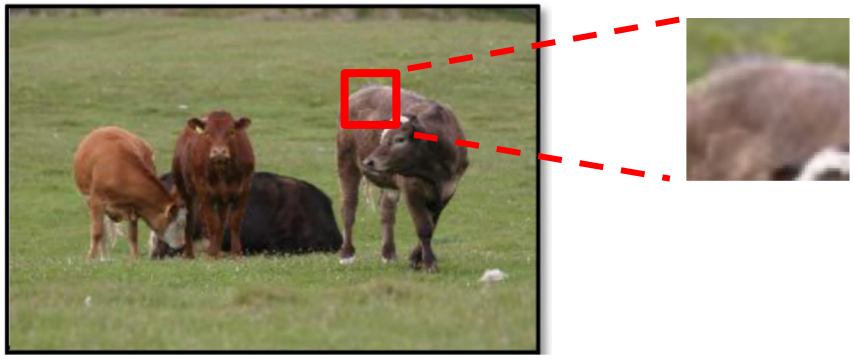


First Try: Sliding window

Source: Stanford CS231n

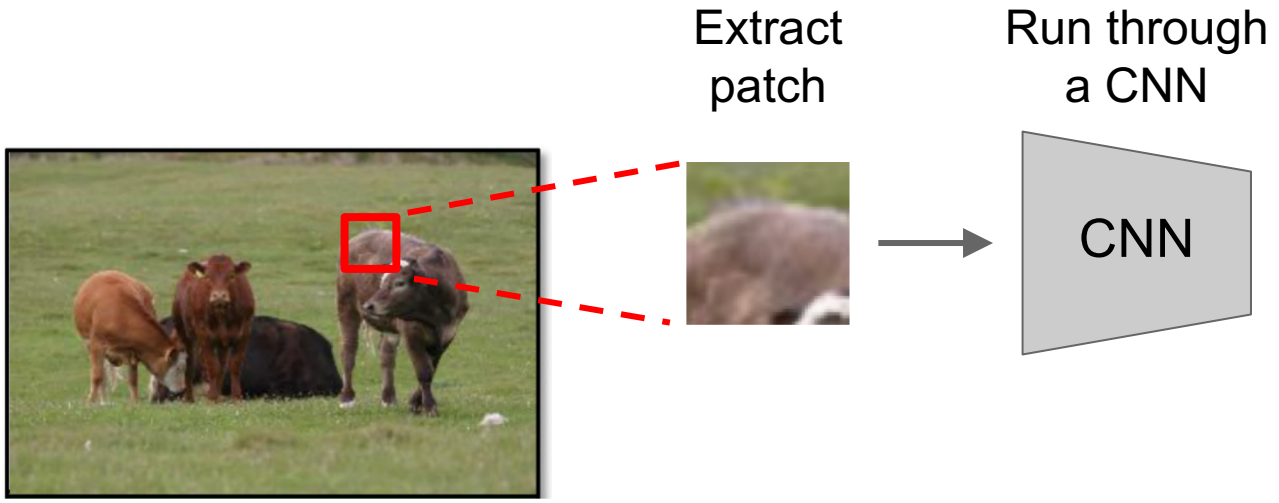
First Try: Sliding window

Extract patch



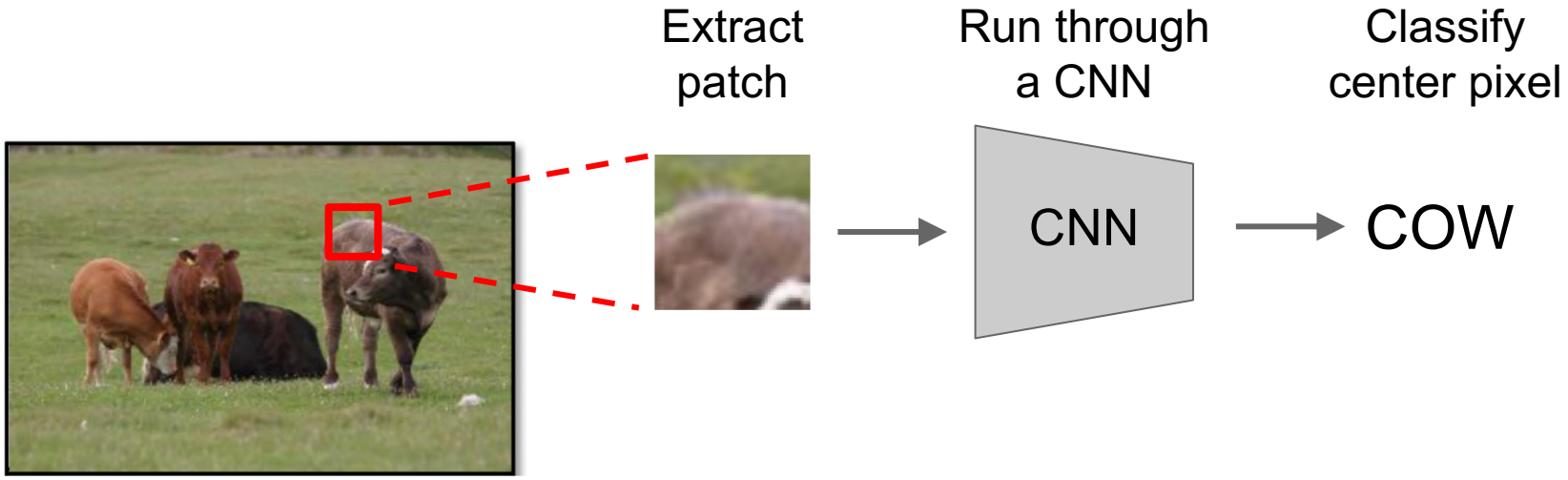
Source: Stanford CS231n

First Try: Sliding window



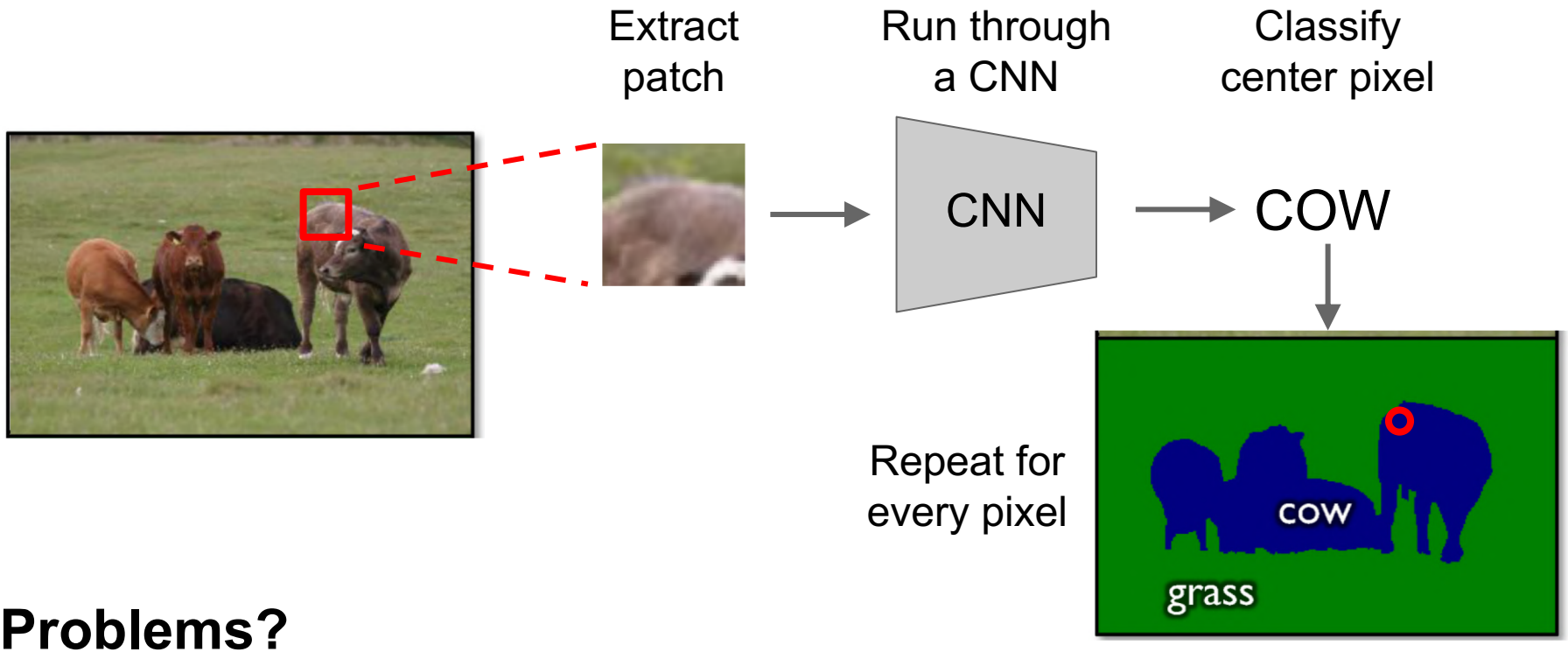
Source: Stanford CS231n

First Try: Sliding window



Source: Stanford CS231n

First Try: Sliding window



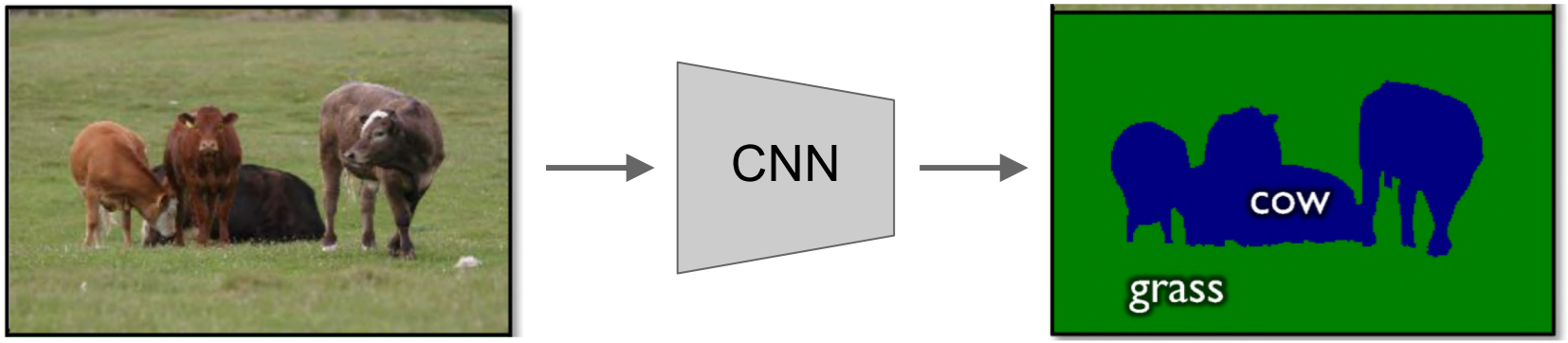
Problems?

- Very inefficient!
- Does not reuse shared features between overlapping patches.
- Local receptive fields, ignoring contextual information.

Source: Stanford CS231n

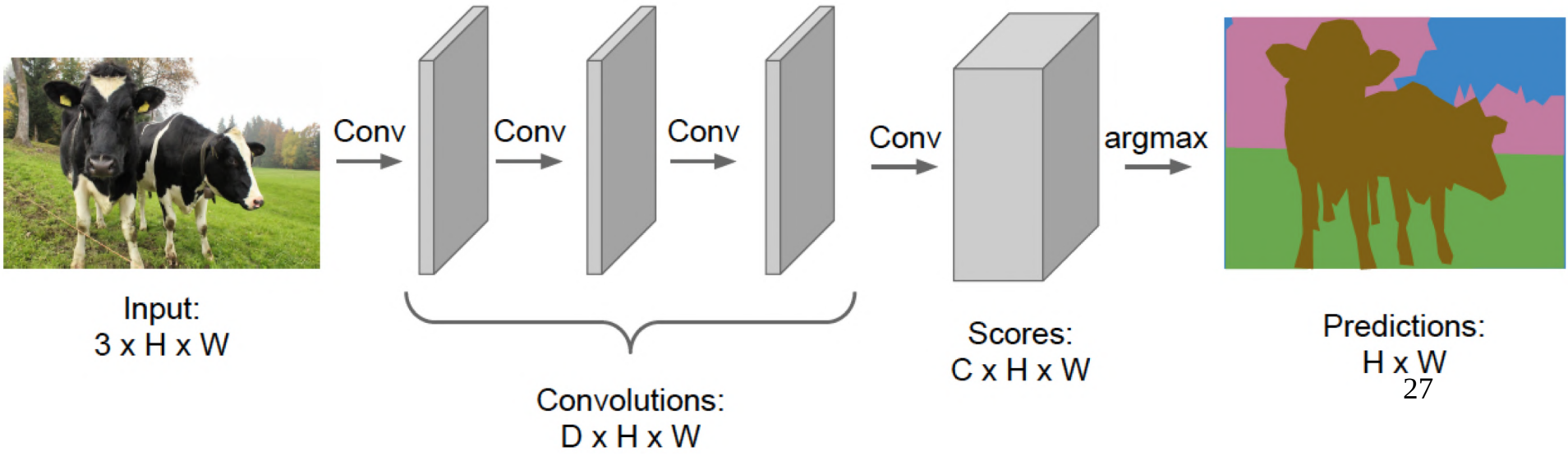
Second Try: Fully convolutional networks

Run “fully convolutional” network
to get all pixels at once



Second Try: Fully convolutional networks

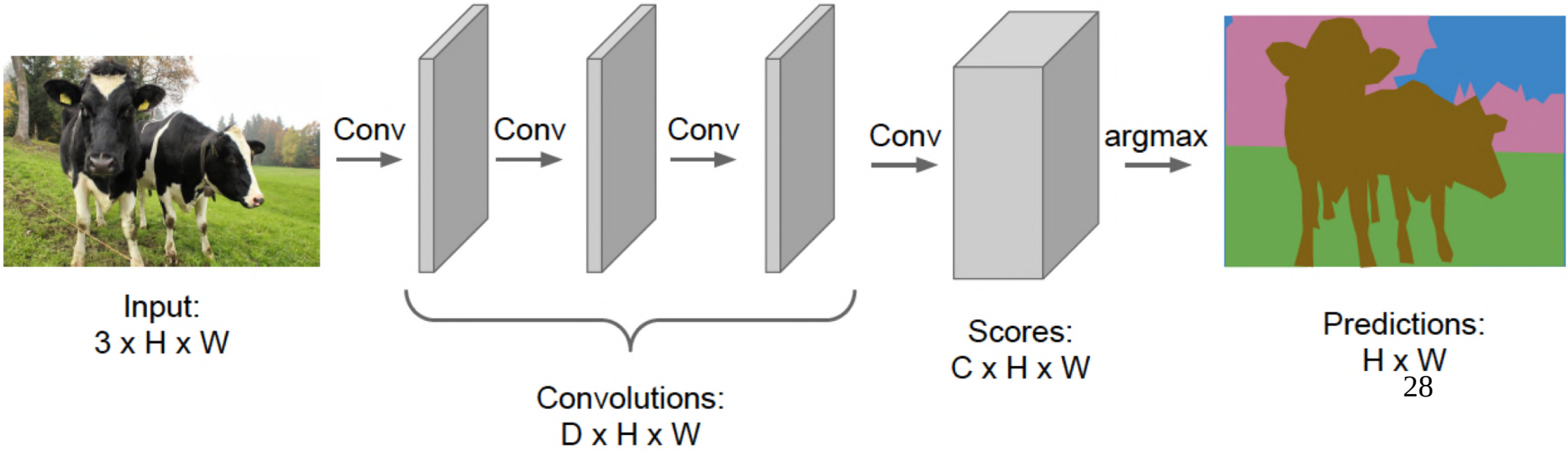
- Design a network with only convolutional layers.
- Stack a bunch of feature maps and output a final segmentation map at once using C 1x1 conv filers.
- Ideally, we want convolutions at full a image resolution.



Second Try: Fully convolutional networks

Problems?

- Computationally expensive (full image resolution).
- **Tradeoff**: Shallow layers tend to learn local pictorial features (where) while deeper layers learn global and semantic concepts (what).
- For segmentation we need both as we care about segmentation (what) at the image resolution (where).



Solution: Encoder-Decoder Structure

- ***Downsample*** the spatial resolution of the input, developing high-level feature(encoder).
- Then ***upsample*** the feature representations into a full-resolution segmentation map (decoder).
- Combines *where* (local, shallow) with *what* (global, deep).

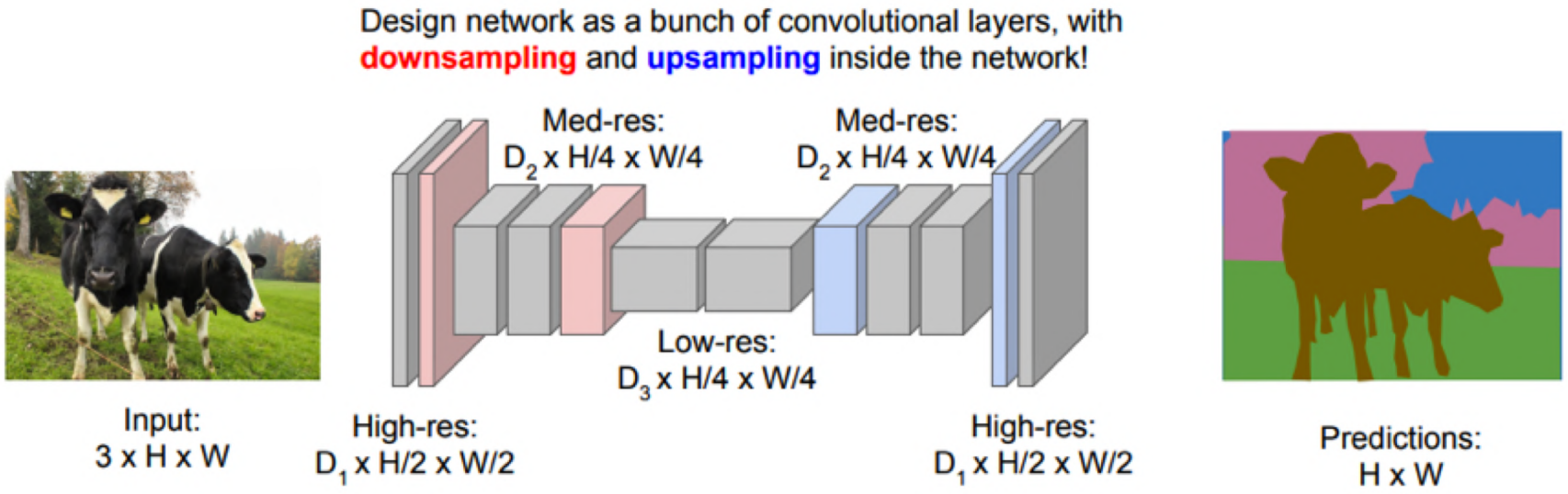
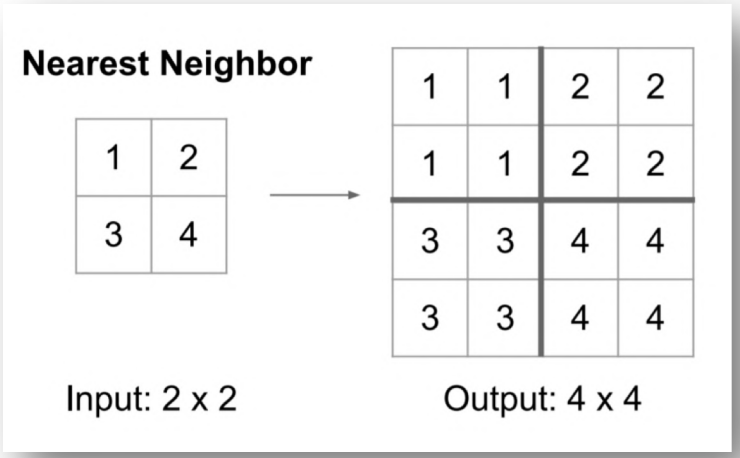


Image source: Stanford CS231n

Methods for Upsampling:

- **Nearest neighbor** upsampling.
- **Bed of nails** + learnable convolution.



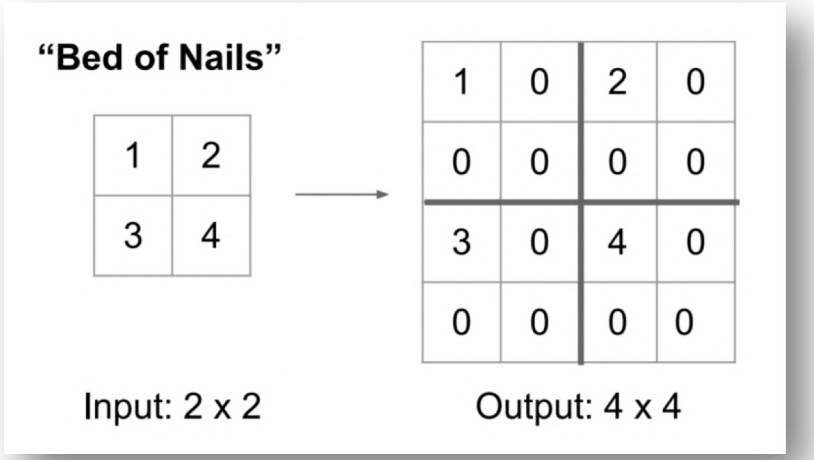
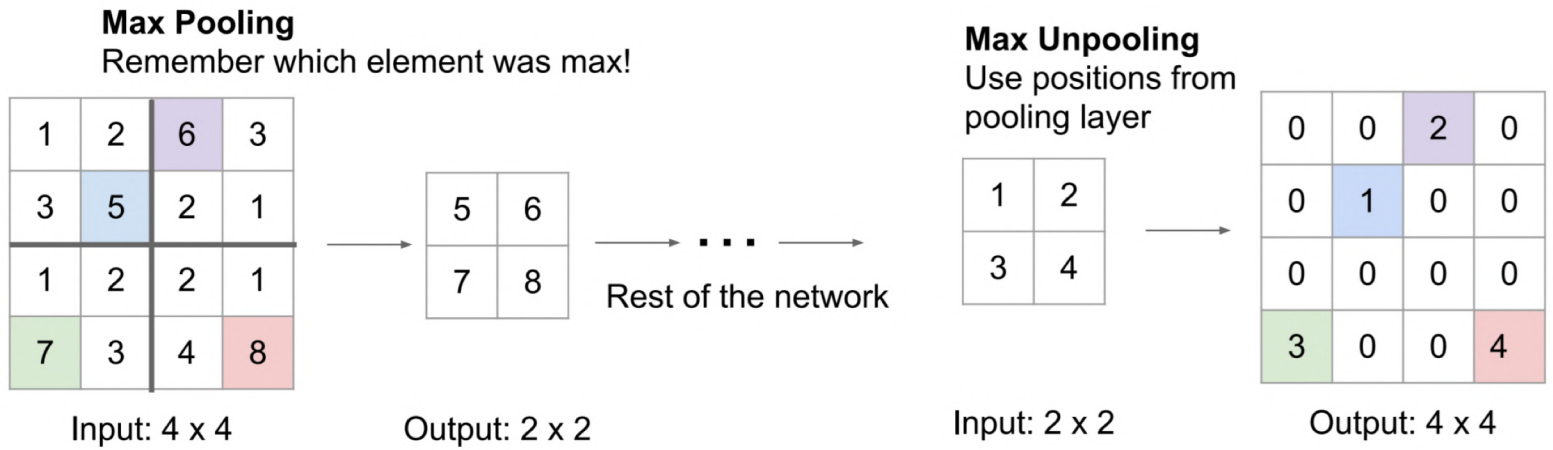
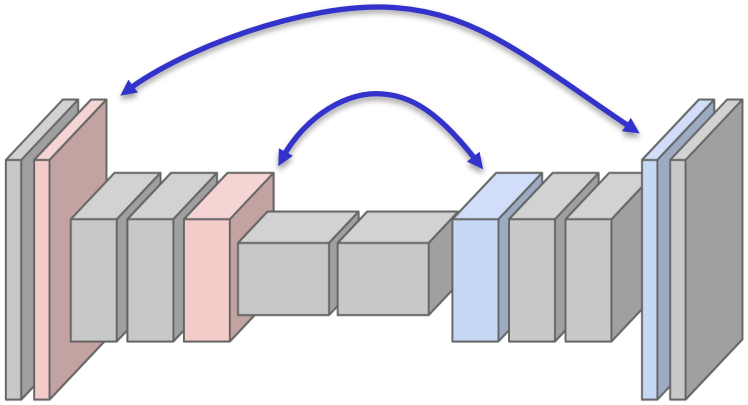


Image source: Stanford CS231n

Methods for Upsampling:

- **Max Unpooling:** Use positions from pooling layer.

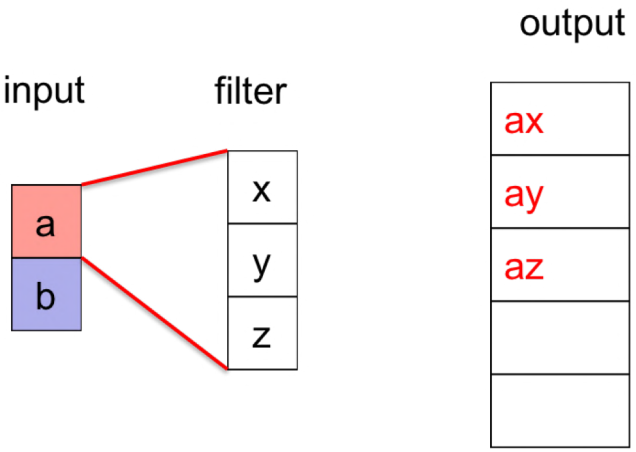




Corresponding pairs of downsampling and upsampling layers

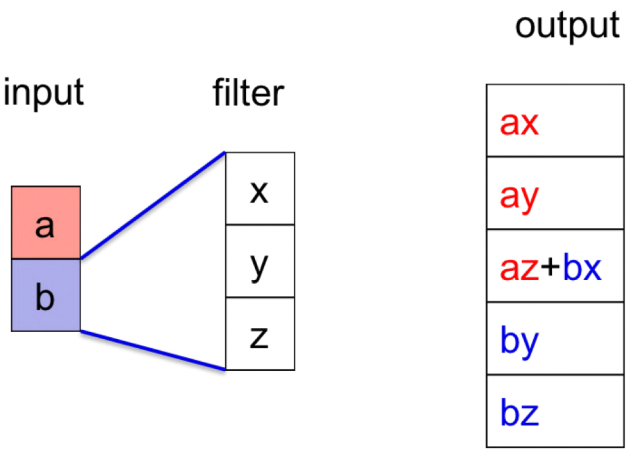
Methods for Upsampling:

- The drawback of previous methods is that upsampling is not learnable.
- **Transpose convolution**: The most common method. It applies the opposite direction of a common convolution.
- *Typical convolution*: Take a dot product between the filter weights and an image window.
- *Transpose convolution*: Take a single value from the image and multiply the filter weights. Project those weighted values into the output feature map.



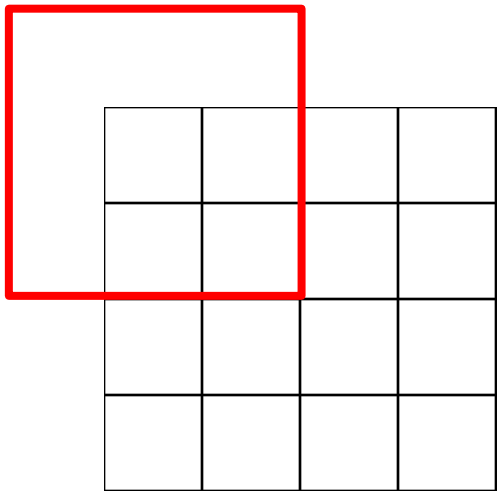
Methods for Upsampling:

- The drawback of previous methods is that upsampling is not learnable.
- **Transpose convolution**: The most common method. It applies the opposite direction of a common convolution.
- *Typical convolution*: Take a dot product between the filter weights and an image window.
- *Transpose convolution*: Take a single value from the image and multiply the filter weights. Project those weighted values into the output feature map.



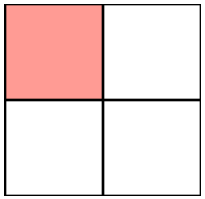
Learnable Upsampling: Conv-transpose

Common 3 x 3 convolution, **stride 2** pad 1



Input: 4 x 4

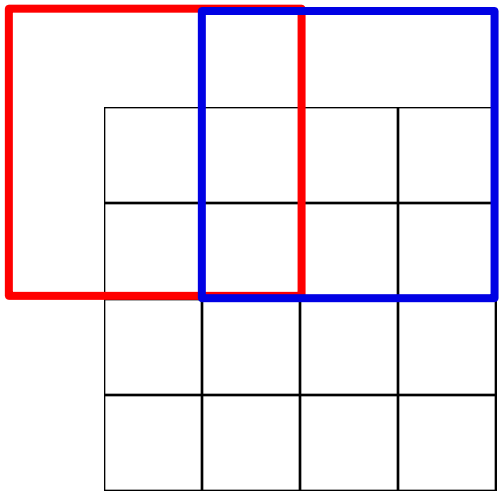
Dot product between filter and input



Output: 2 x 2

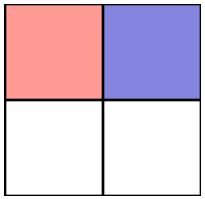
Learnable Upsampling: Conv-transpose

Common 3 x 3 convolution, **stride 2** pad 1



Input: 4 x 4

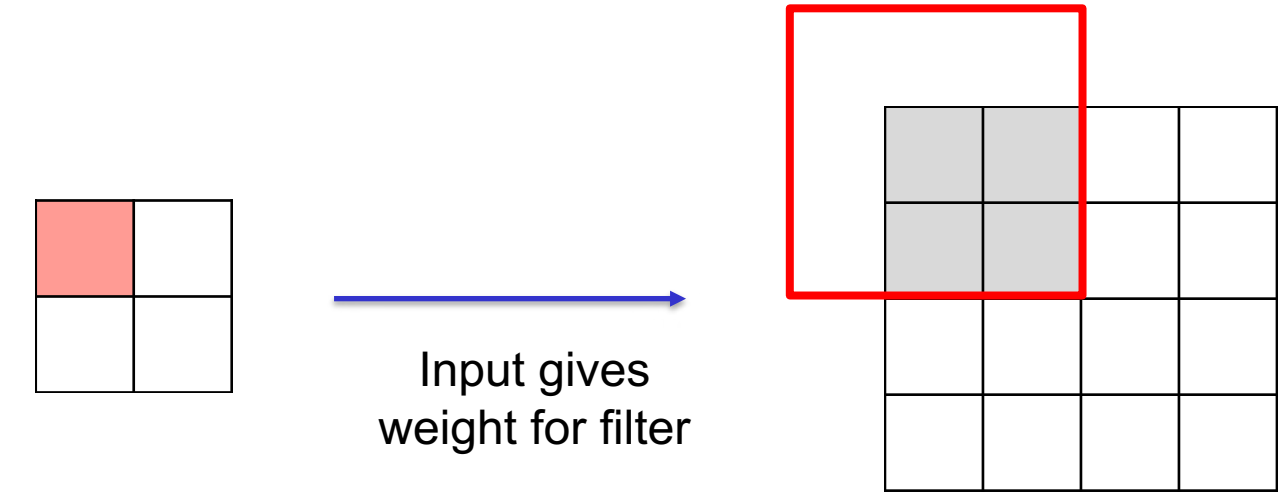
Dot product between filter and input



Output: 2 x 2

Learnable Upsampling: Conv-transpose

3 x 3 **conv-transpose**, **stride 2** pad 1

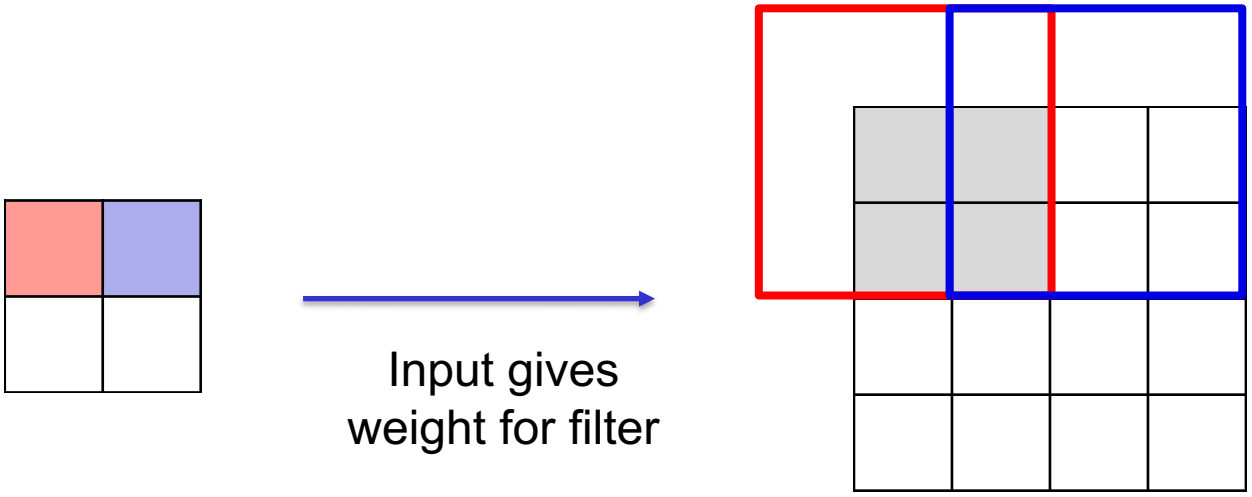


Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Conv-transpose

3 x 3 **conv-transpose**, **stride 2** pad 1

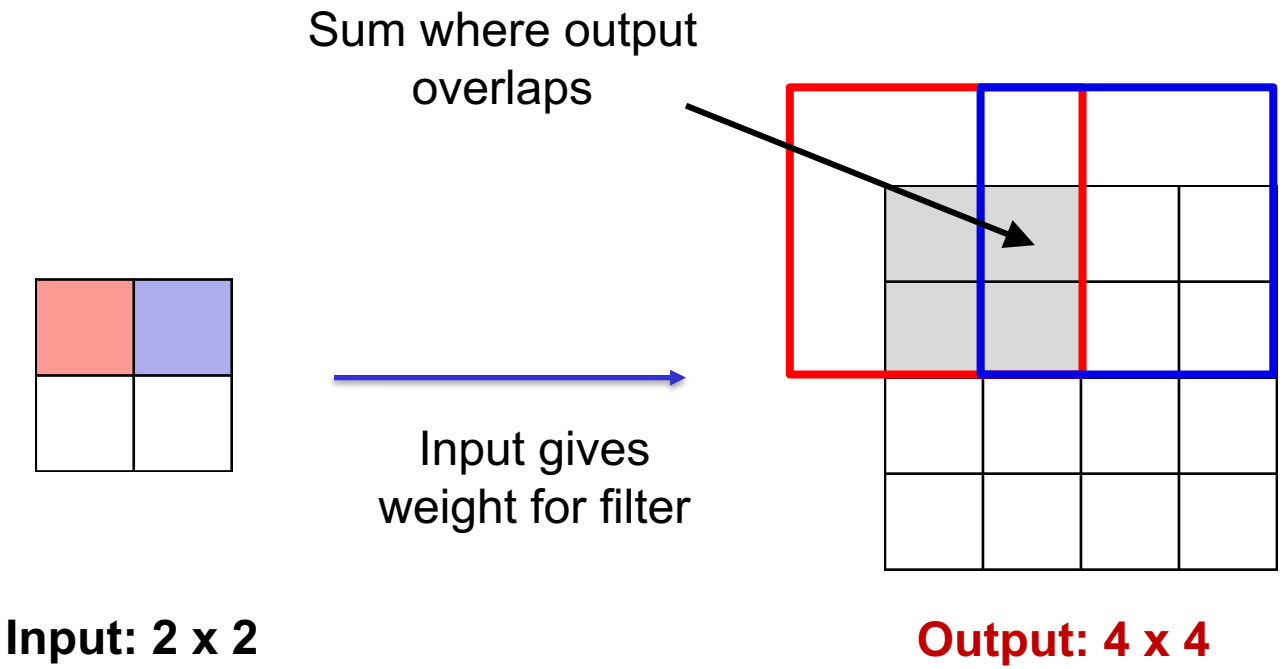


Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Conv-transpose

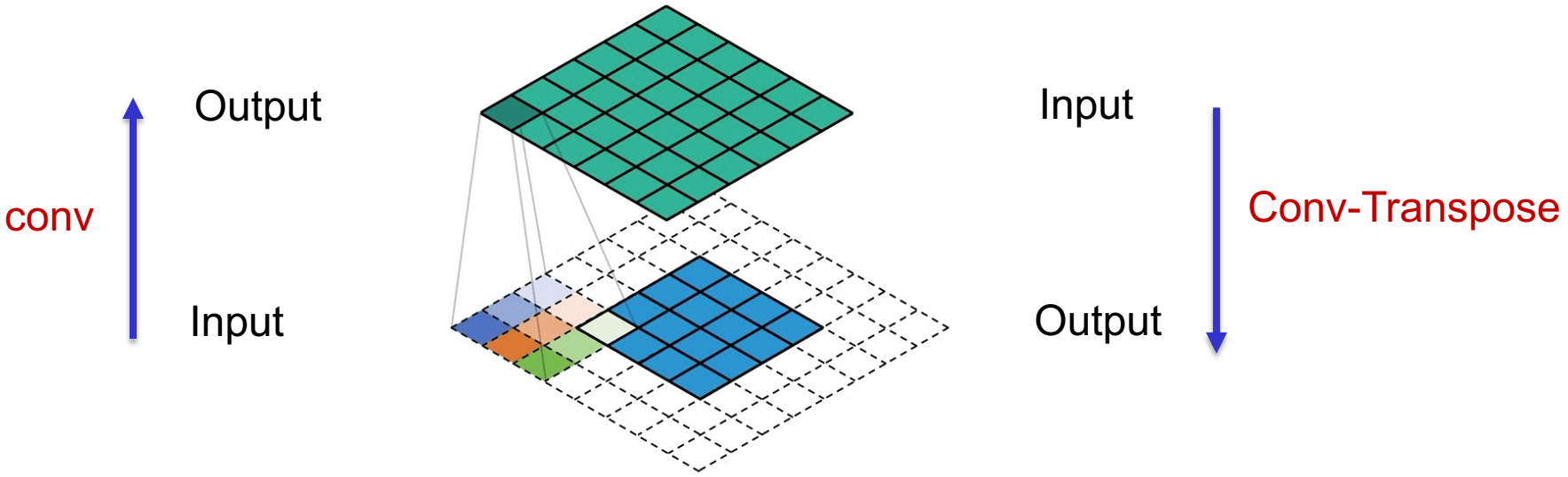
3 x 3 **conv-transpose**, **stride 2** pad 1



- Conv-transpose is similar to the back-prop pass for normal convolution
- The pack-prop operation of conv-transpose is the normal conv.

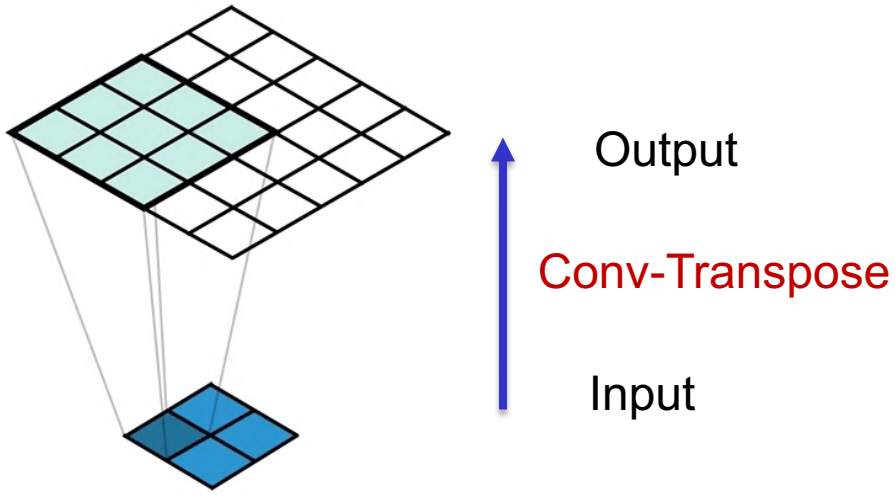
Learnable Upsampling: Conv-transpose

- Conv-transpose is equivalent to applying a regular convolution with the transpose (rot-180) filter (but we don't really care because the filters are learnable).
- Conv-transpose with no padding is equivalent to applying Conv with full padding (P=F-1).



Source gif images

Learnable Upsampling: Striding

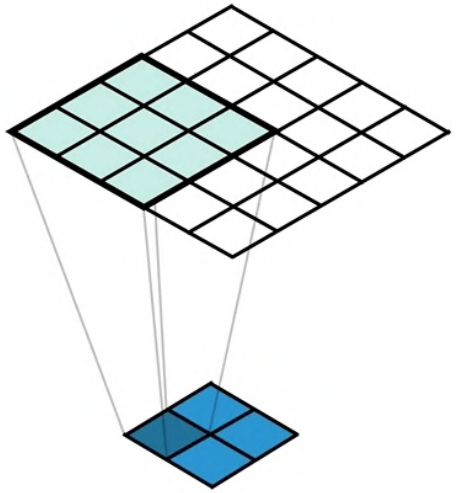


- Striding over the output, **increases the size of the output**.
- Alternative view: A stride of 2 over the output would be equivalent to a stride of 1/2 over the input: a fractional stride.

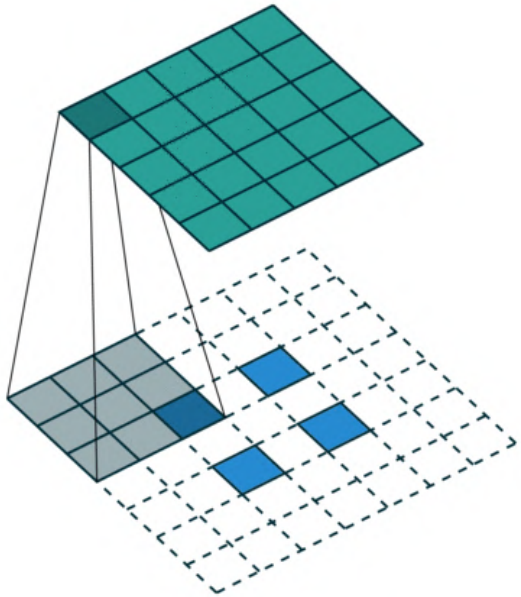
Source gif images

Learnable Upsampling: Striding

Conv 3x3 stride=1/2



Conv 3x3 stride=1



- Stride 1/2 over the input, can be equivalently implemented by introducing empty spaces between input values by the amount proportional to the stride, and then convolve with stride=1

Source gif images

Learnable Upsampling: Conv-transpose

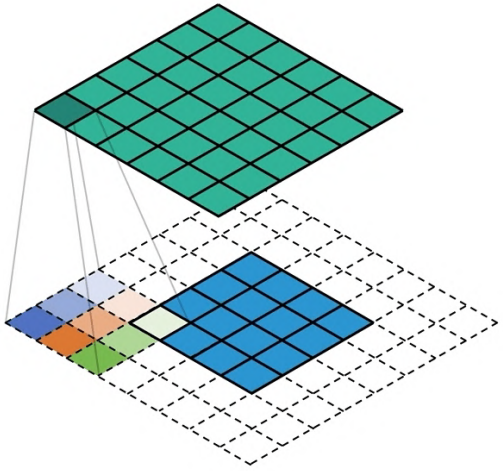
- In regular convolution:

$$outputSize = \frac{inputSize - F + 2P}{stride} + 1$$

- In conv-Transpose:

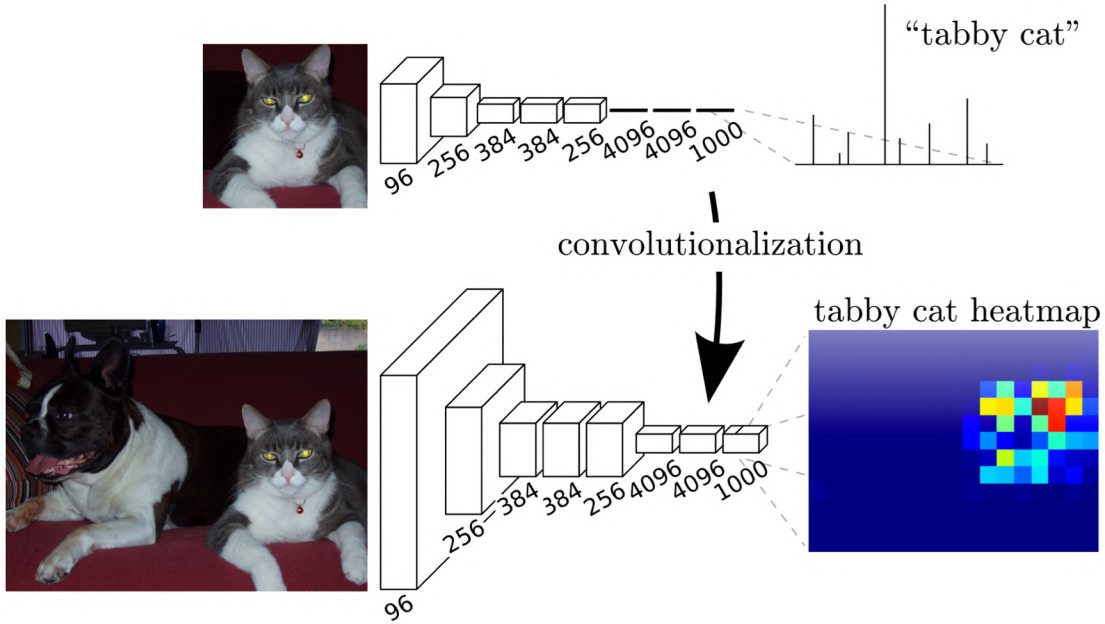
$$inputSize = \frac{OutputSize - F + 2P}{stride} + 1$$

$$outputSize = (inputSize - 1) * stride + F - 2P + output\_padding$$



Segmentation by FCN (Fully Conv. Network)

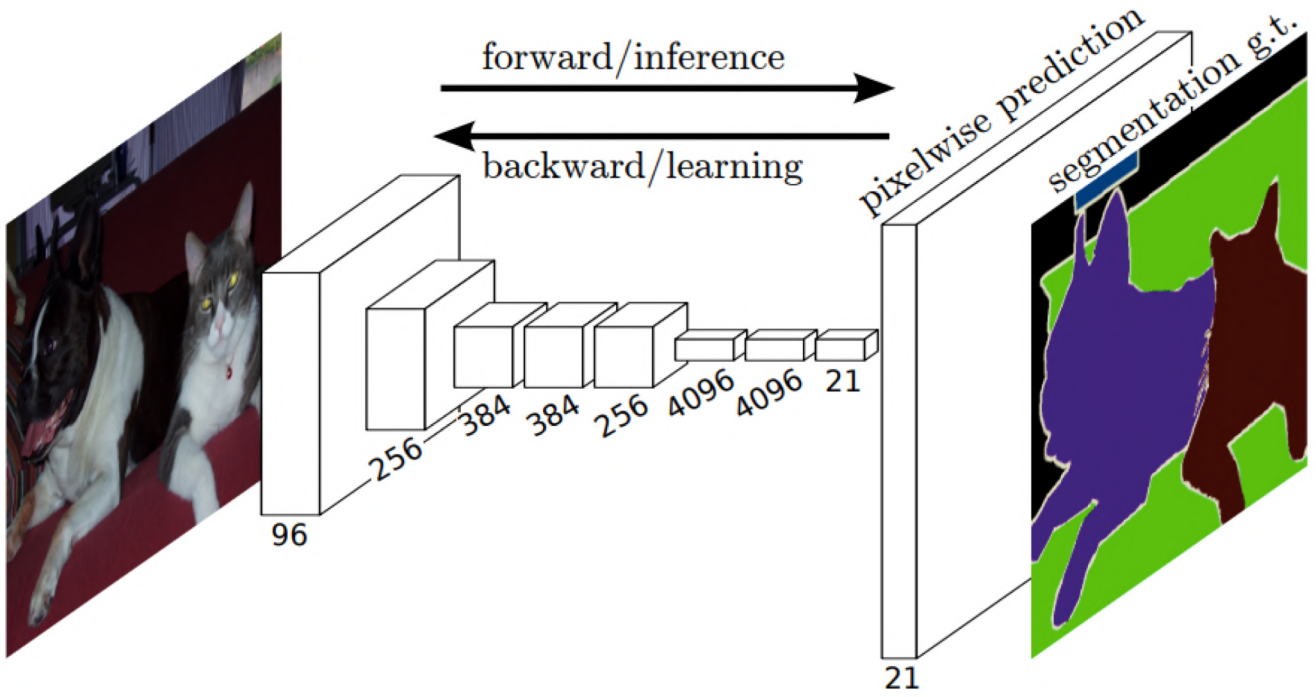
- A “fully convolutional” network is trained end-to-end.
- Adapting AlexNet to to serve as the **encoder** module.
- The **encoder** consists of convolutional layers reducing the input resolution by 32 .



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

Segmentation by FCN (Fully Conv. Network)

- Final encoder convolution with #classes 1x1 filters.
- Upsample back to original size by the **decoder** using convTrans.
- The full network, is trained end-to-end using pixel-wise *cross entropy loss*.



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

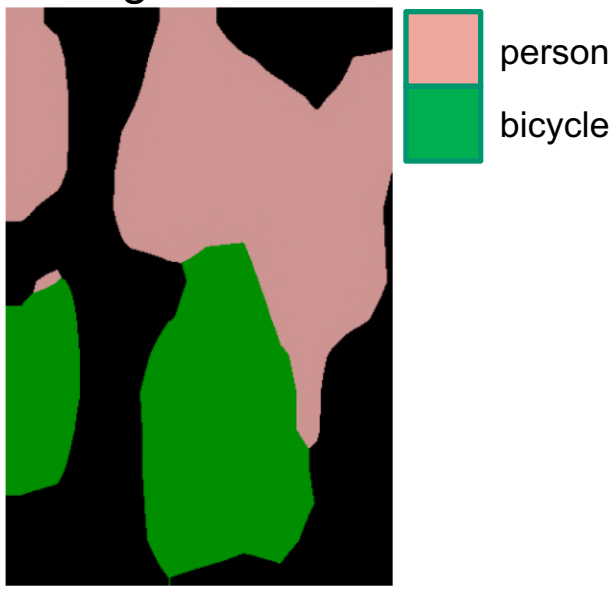
Segmentation by FCN (Fully Conv. Network)

Problems?

- Decoder is fed with coarse feature map ignoring fine feature maps from shallow layers.
- Upsampling from coarse features consists of high-level semantic information but lacks spatial accuracy.

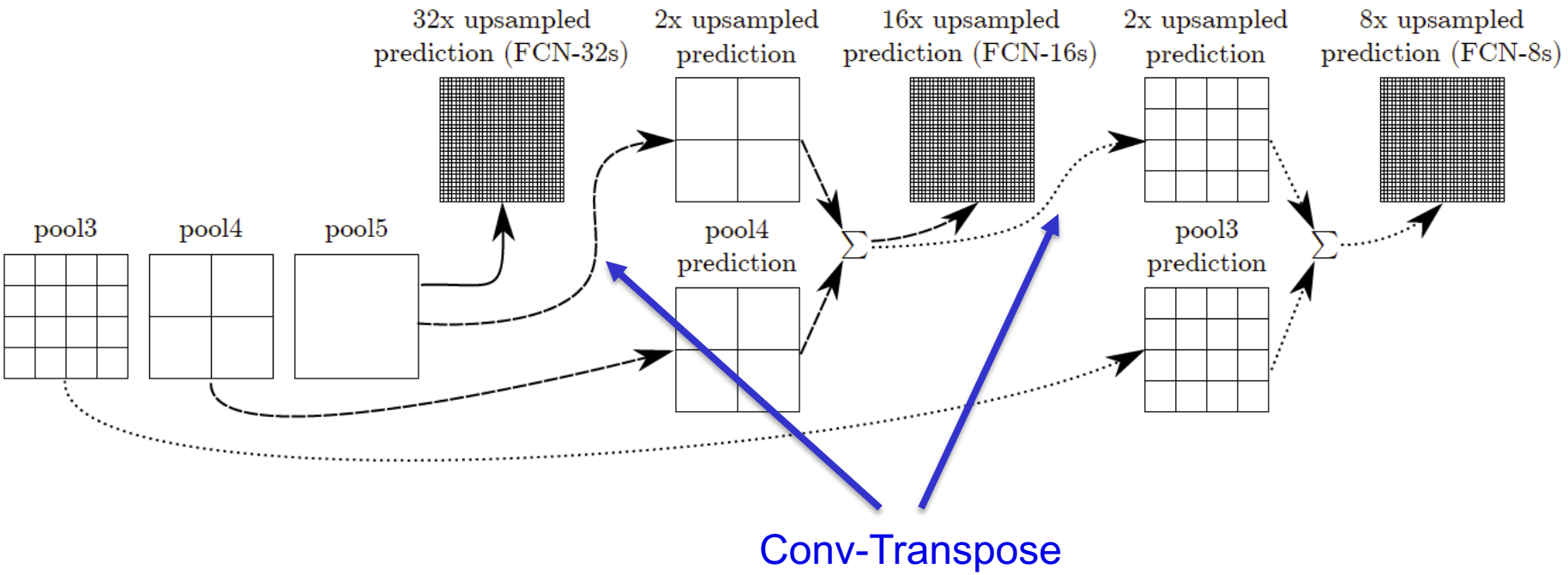
Ground truth

Segmentation



FCN + Skip Connection

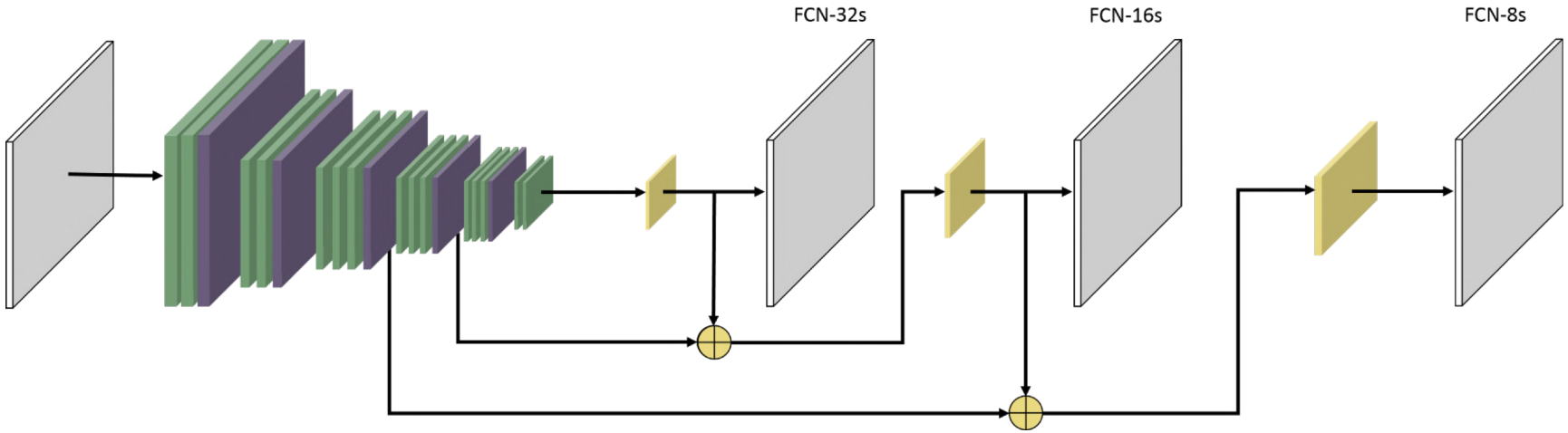
- Skip connections from earlier layers in the network.
- Upsampling (in stages) the representation from shallow layers while adding "skip connections" to higher layers
- Summing the scores from all levels.



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

FCN + Skip Connection

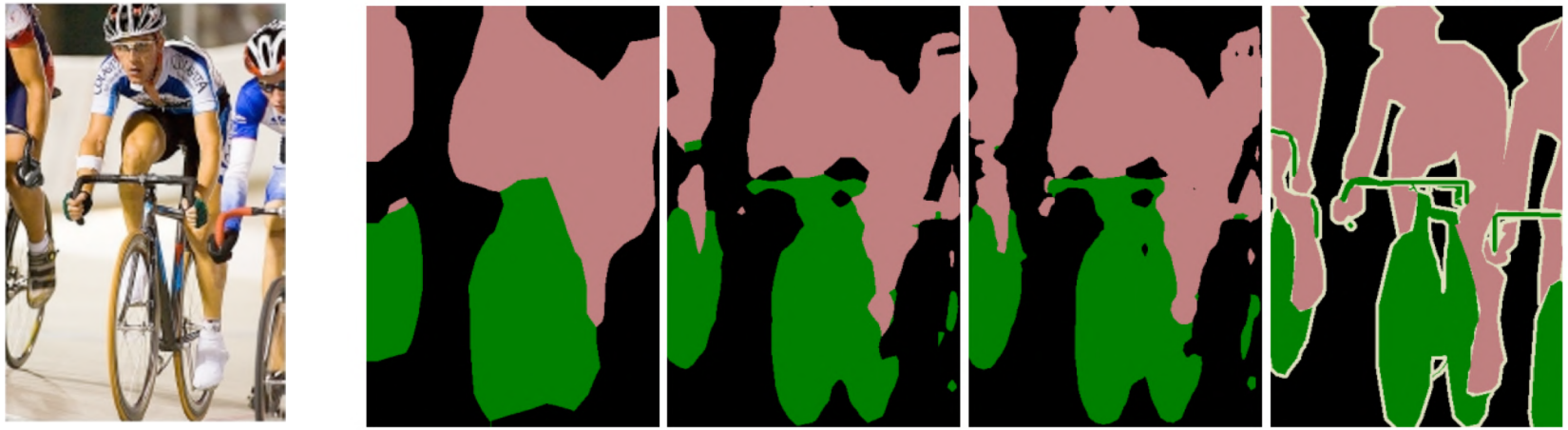
- Skip connections from earlier layers in the network
- Upsampling (in stages) the representation from shallow layers while adding "skip connections" to higher layers
- Summing the scores from all levels.



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

FCN + Skip Connection

- Skip connections from earlier layers in the network
- Upsampling (in stages) the representation from shallow layers while adding "skip connections" to higher layers
- Summing the scores from all levels.



Skip connections = Better results

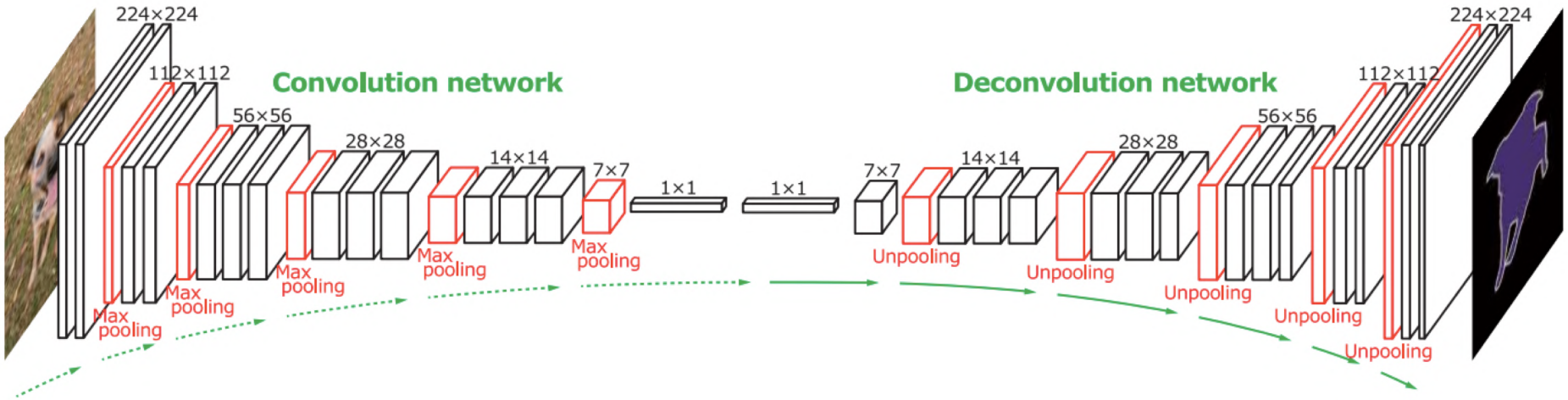
Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

FCN Results

PASCAL VOC 2012	mIoU
Hypercolumns	59.2
ZoomOut	64.4
FCN-8	62.2
DeconvNet	69.6
Ensemble of DeconvNet and FCN	71.7

DeconvNet

- In the FCN architecture the scores are calculated in multi-depth layers and then fused together.
- Calculating scores for each particular layer is lacking as the scores take into account only partial information.
- In the **DeconvNet**, scores are calculated only at the last layer, where all the information is available.



H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet

- Uses VGGNet as backbone.
- The first part (encoder) is a convolution network with conv and pooling layers.
- The second part (decoder) is the deconvolution network.
- It performs **max-unpooling** and convTrans at each layer, thus needs to remember the positions of maximum value when doing max pooling.
- The convTrans layers densify the sparse activations obtained by unpooling.

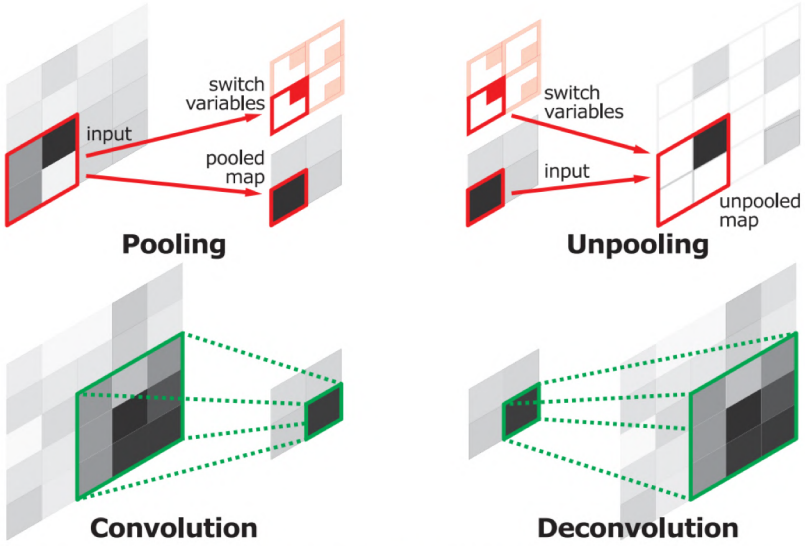
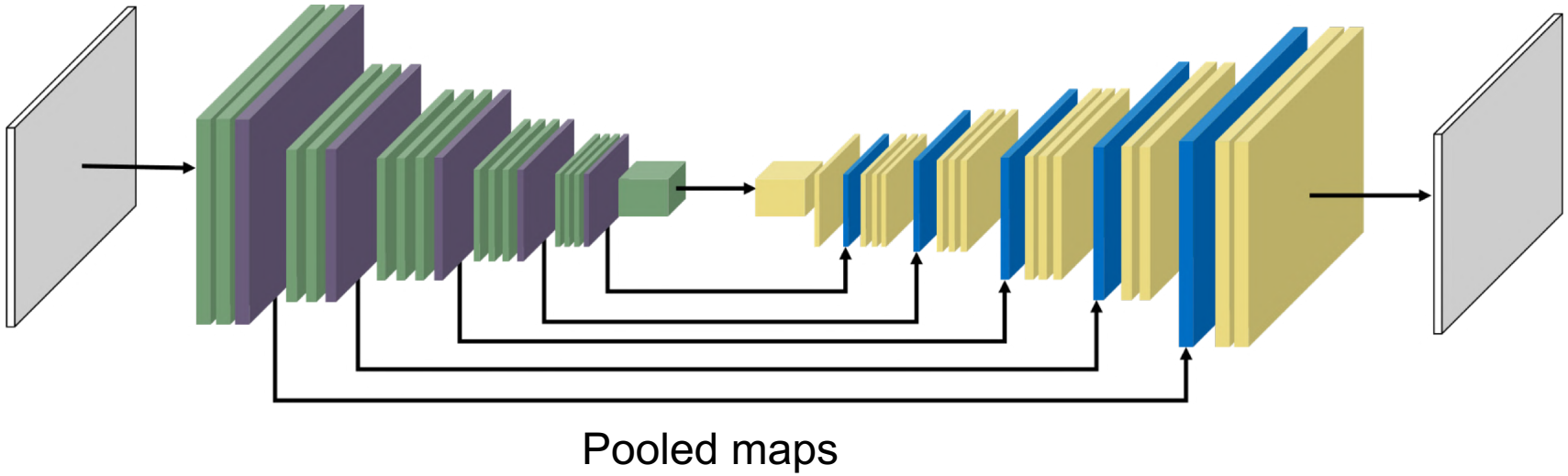


Figure 3. Illustration of deconvolution and unpooling operations.

H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

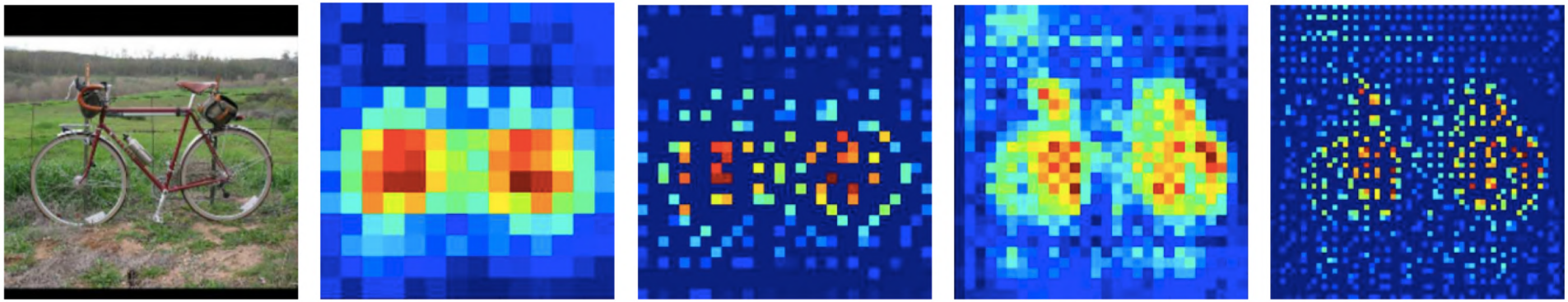
DeconvNet

- Uses VGGNet as backbone.
- The first part (encoder) is a convolution network with conv and pooling layers.
- The second part (decoder) is the deconvolution network.

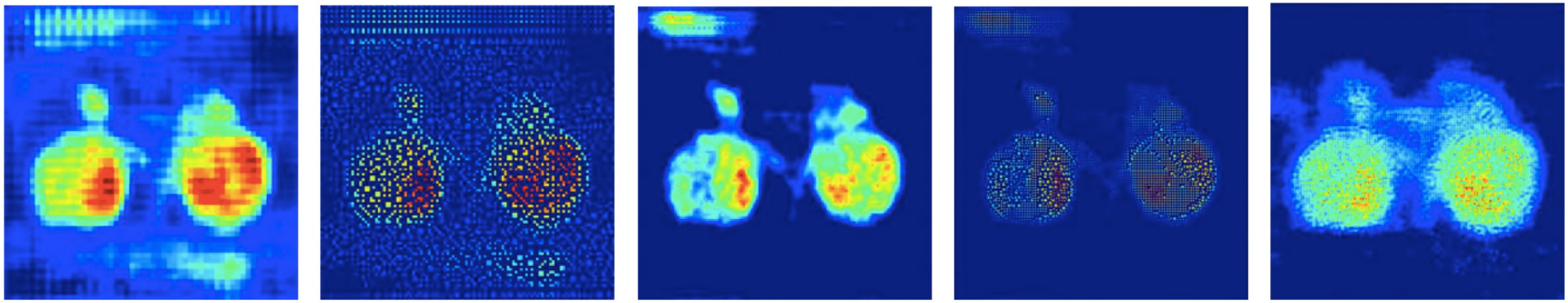


H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet



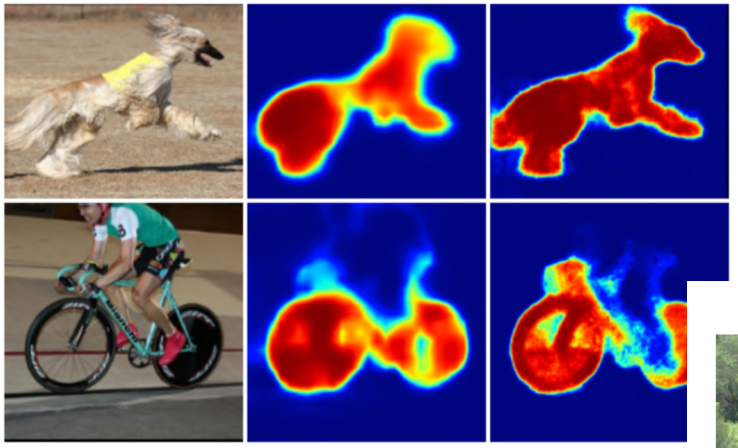
Original image | 14x14 deconv | 28x28 unpooling | 28x28 deconv | 54x54 unpooling



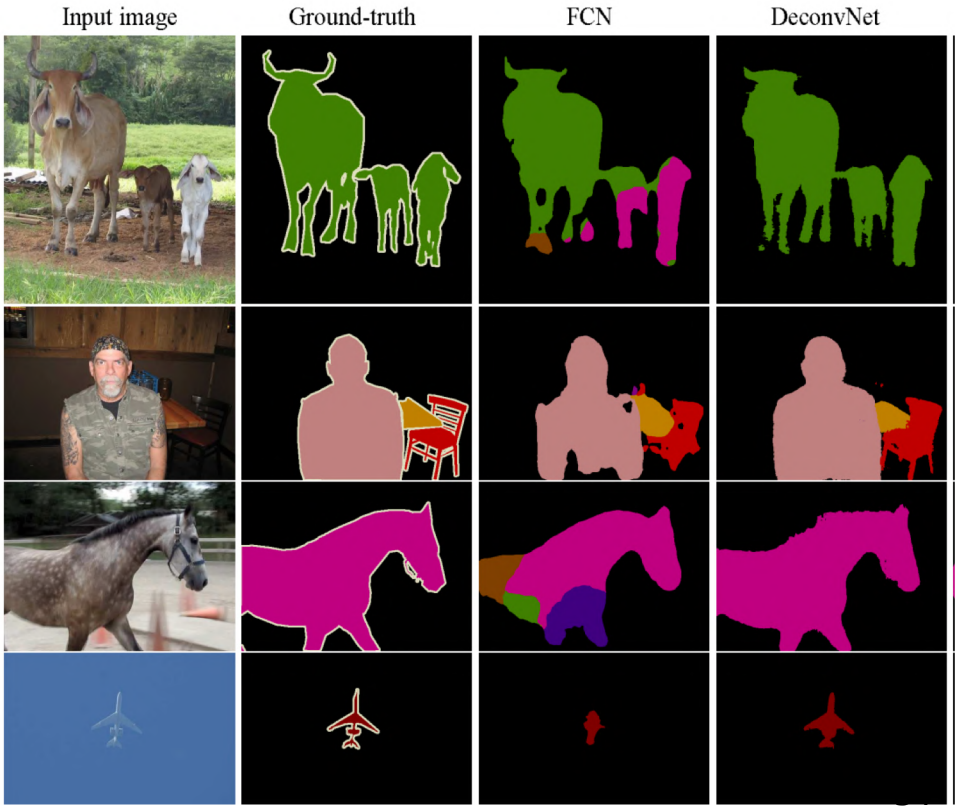
54x54 deconv | 112x112 unpooling | 112x112 deconv | 224x224 unpooling | 224x224 deconv

H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet



Input Image (Left), FCN-8s (Middle), DeconvNet (Right)



H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

DeconvNet

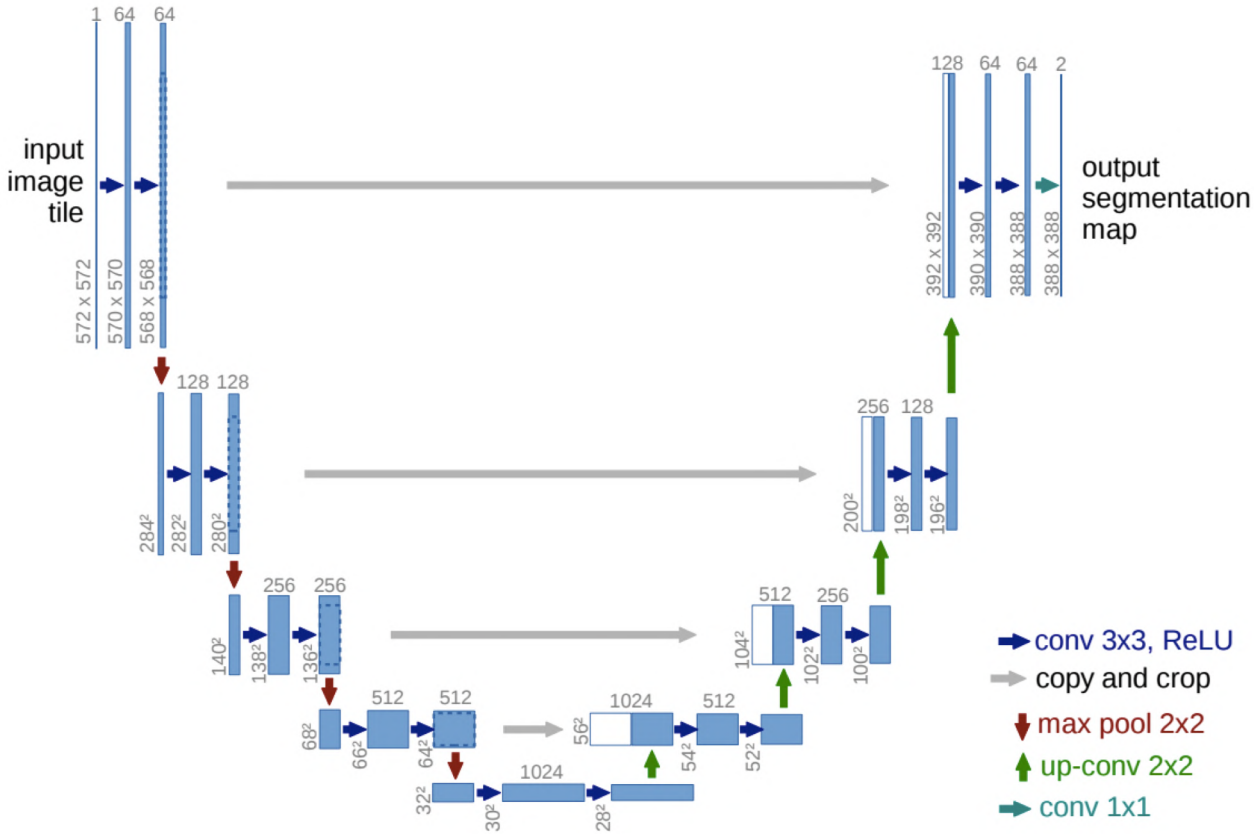
- DeconvNet takes 6 days to train on a TitanX.
- Results are really good.

PASCAL VOC 2012	mIoU
Hypercolumns	59.2
ZoomOut	64.4
FCN-8	62.2
DeconvNet	69.6
Ensemble of DeconvNet and FCN	71.7

H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

U-Net

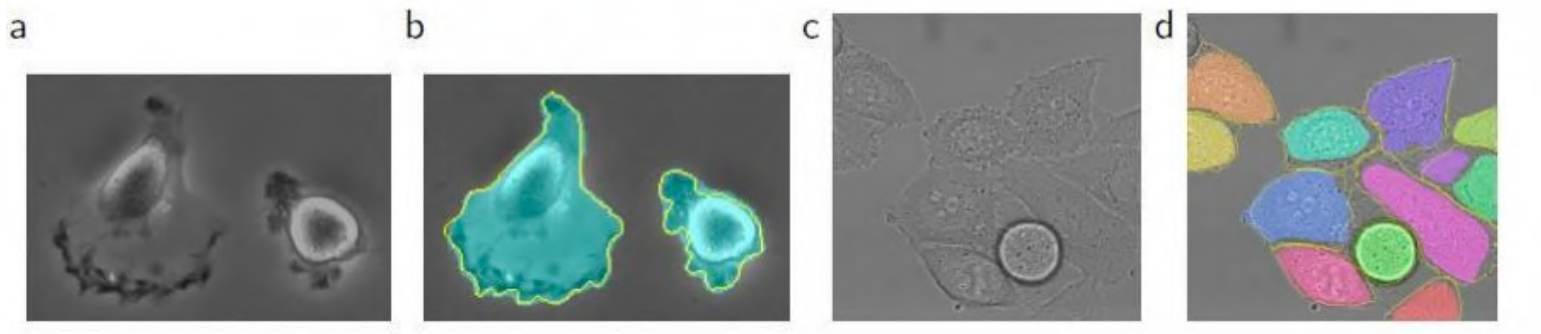
- Like FCN, fuses up-sampled higher-level feature maps with higher-res, lower-level feature maps.
- Unlike FCN, fuse by concatenation, predict at the end.



O. Ronneberger, P. Fischer, T. Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

U-Net

- U-Net architecture is used mainly in biomedical images.
- It is applied to a cell segmentation task in light microscopic images.
- It achieves outstanding performance on different biomedical segmentation applications.



Result on the ISBI cell tracking challenge. (a) part of an input image of the PhC-U373 data set. (b) Segmentation result (cyan mask) with the manual ground truth (yellow border) (c) input image of the DIC-HeLa data set. (d) Segmentation result (random colored masks) with the manual ground truth (yellow border).

O. Ronneberger, P. Fischer, T. Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

Upsampling Architectures

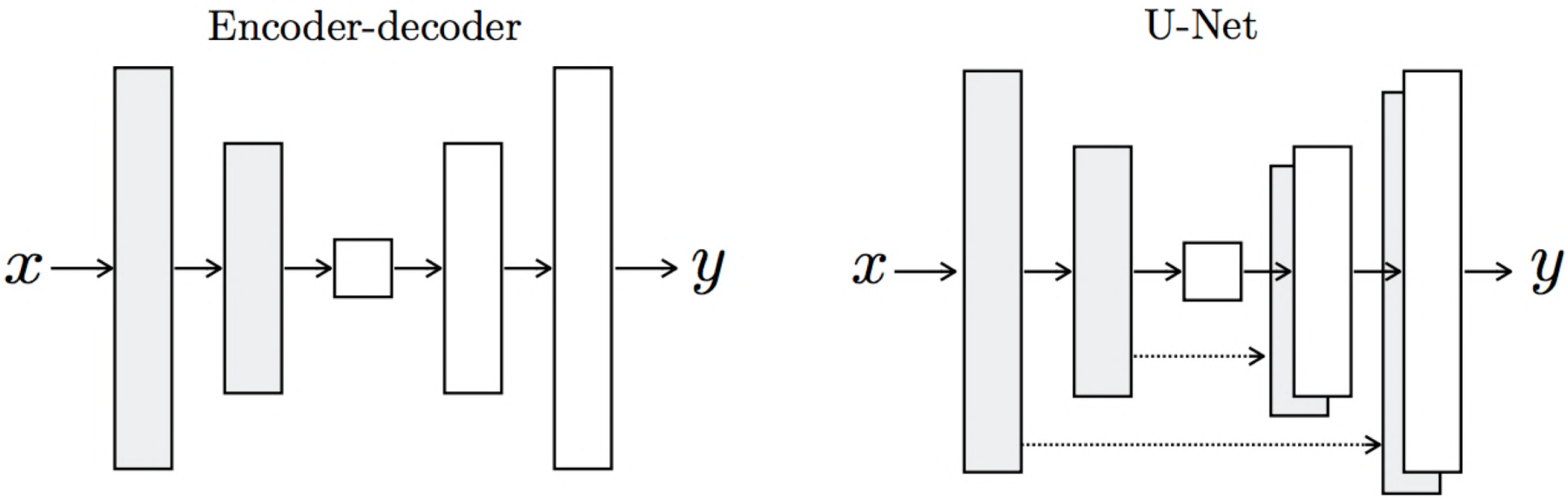
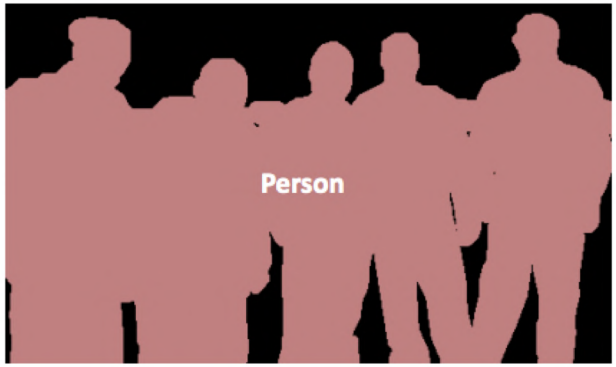


Figure source

Instance Segmentation

Object Detection

✓



Semantic Segmentation

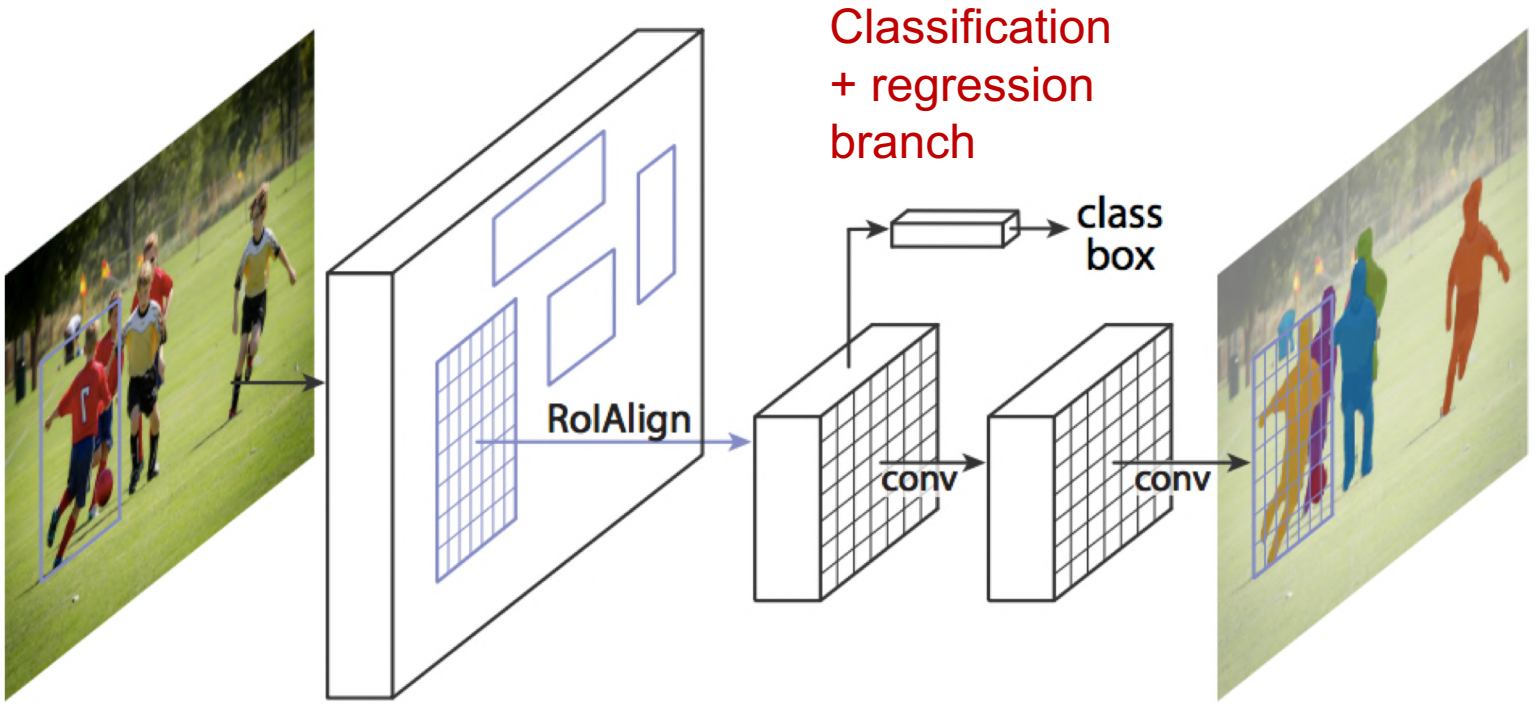
✓

Instance Segmentation

?

Mask R-CNN

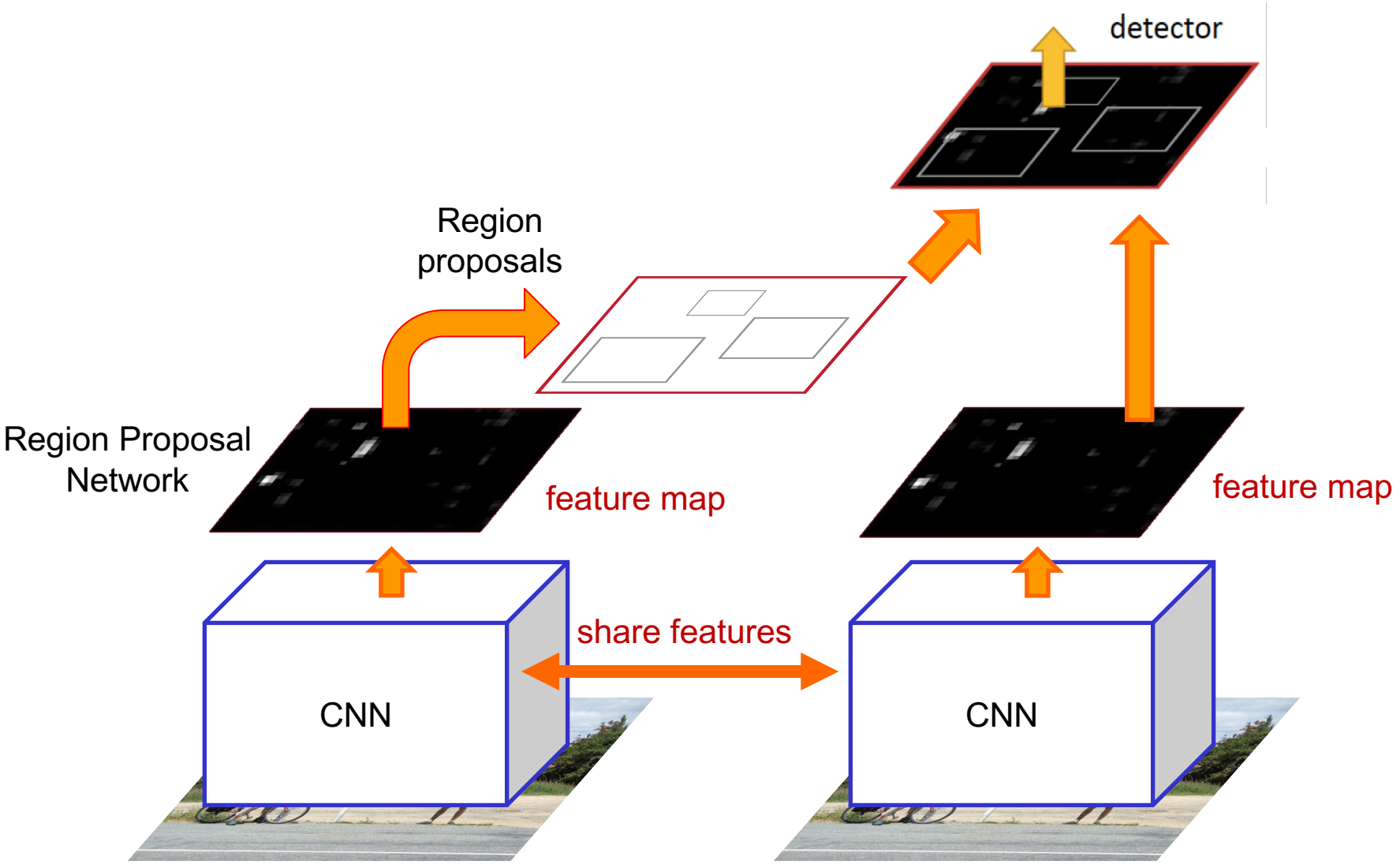
- Mask R-CNN = Faster R-CNN + FCN on RoI's



Mask branch: separately predict segmentation for each possible class

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

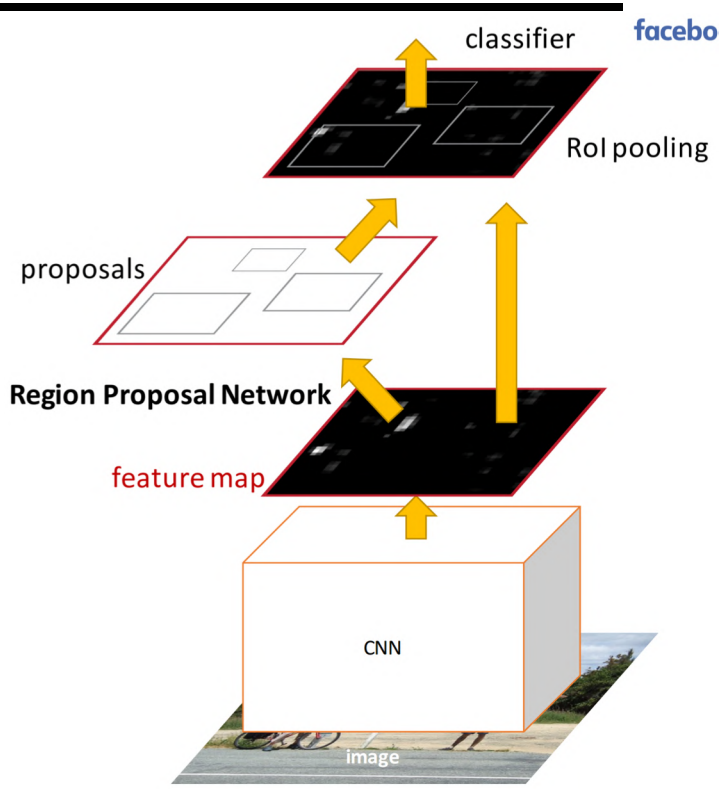
Reminder: Faster R-CNN

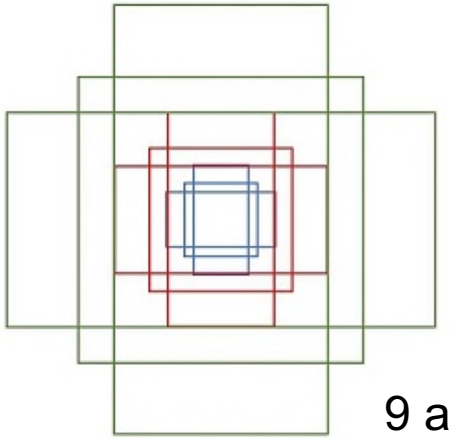


S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS15

Reminder: Faster R-CNN: RPN

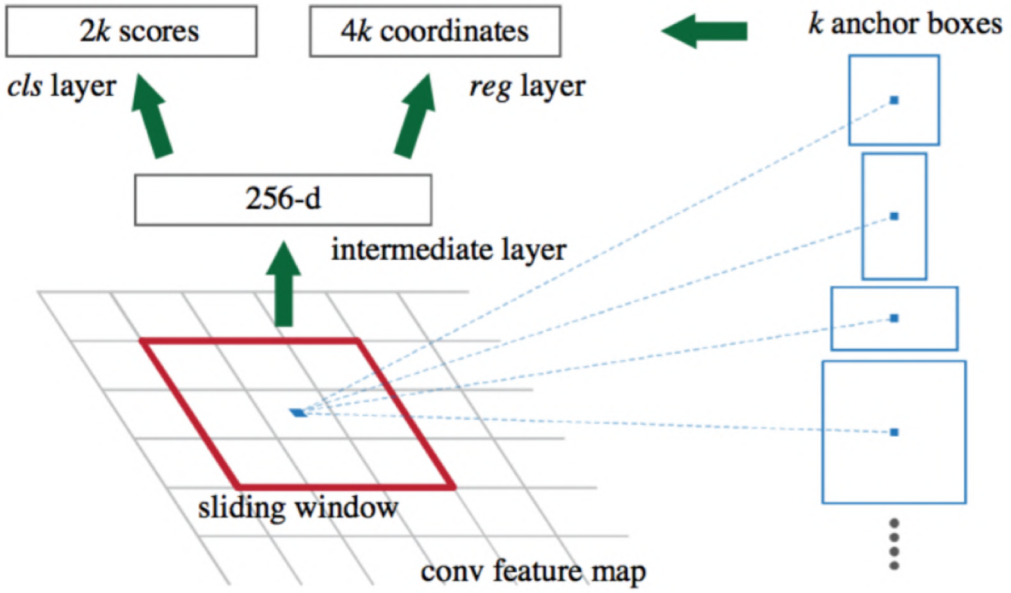
- **Region Proposal Network (RPN)** is trained to produce region proposals.
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor.
- At each location the RPN examines k **anchor boxes** that vary in size and aspect-ratios





9 anchor boxes

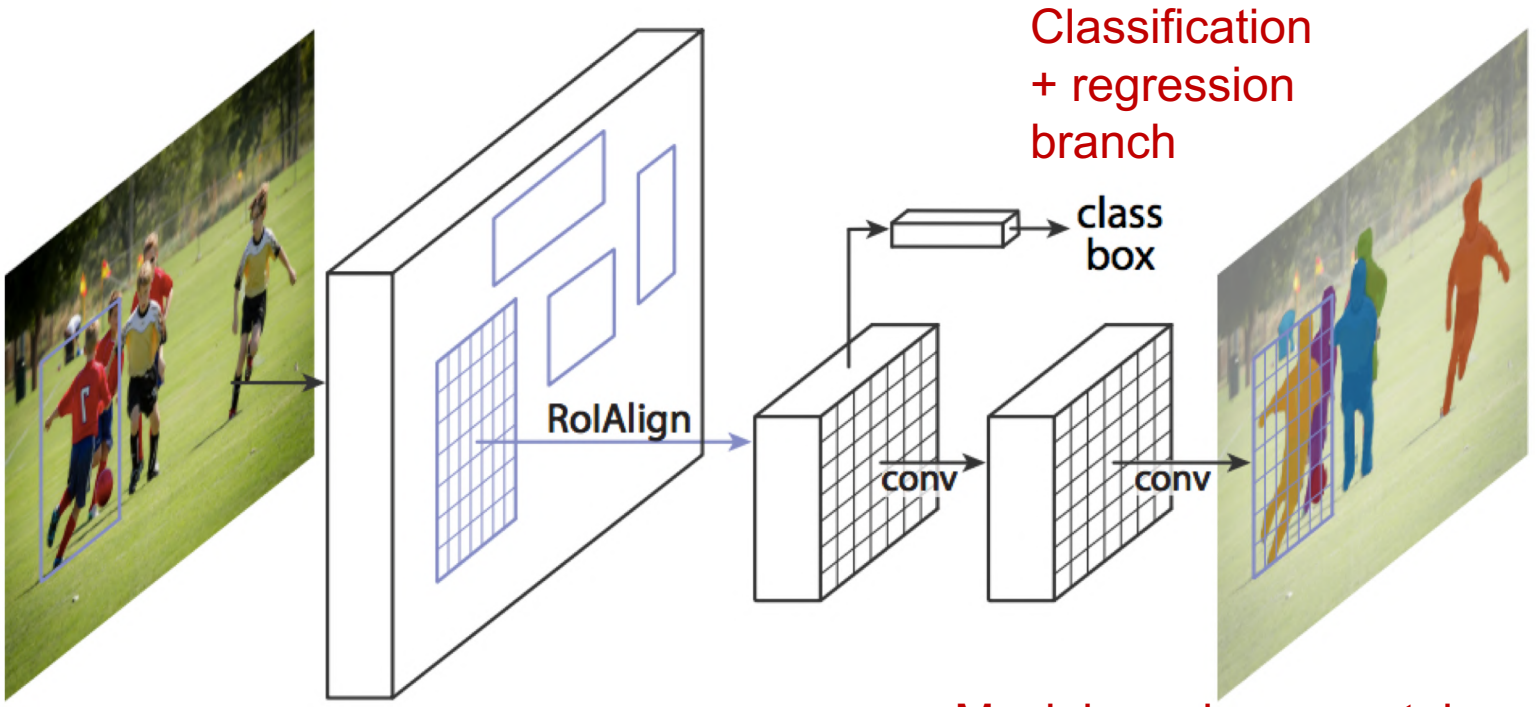
Reminder: Faster R-CNN: RPN



- At each location, hypothesize the k anchor boxes
- Build a small network for:
  - classifying object/non-object
  - regressing bbox locations

Mask R-CNN

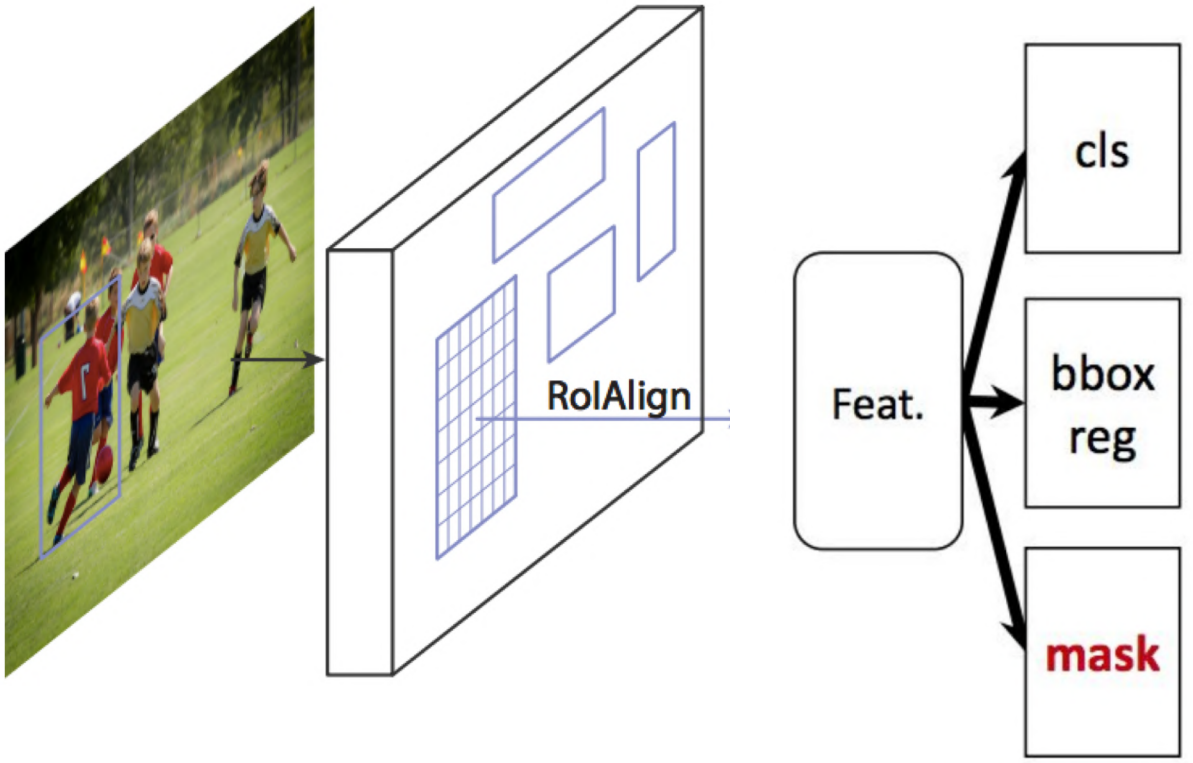
- Mask R-CNN = Faster R-CNN + FCN on RoIs



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

Mask R-CNN

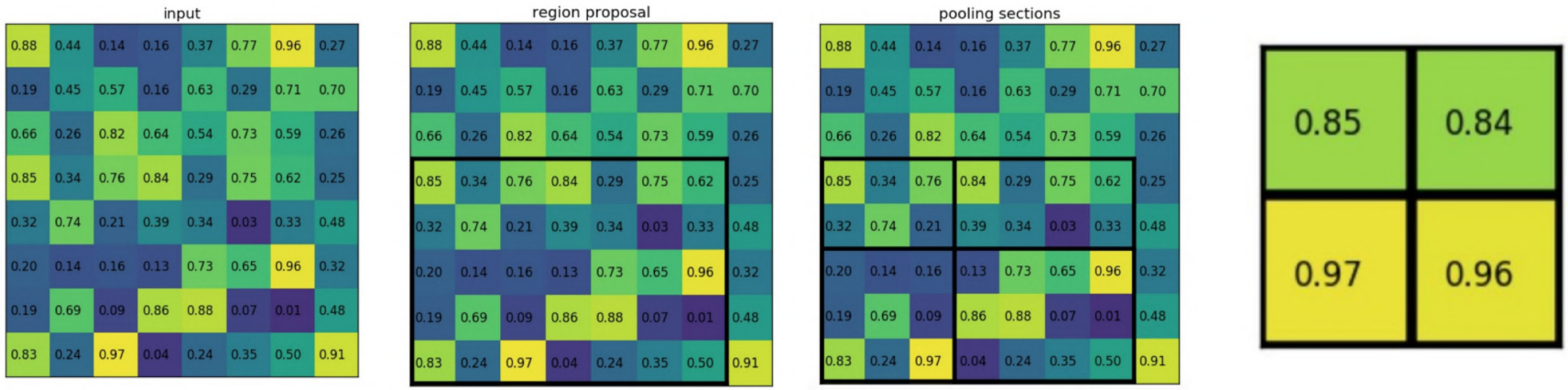
- Mask R-CNN = Faster R-CNN + FCN on RoIs



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

RoIAlign vs. RoIPool

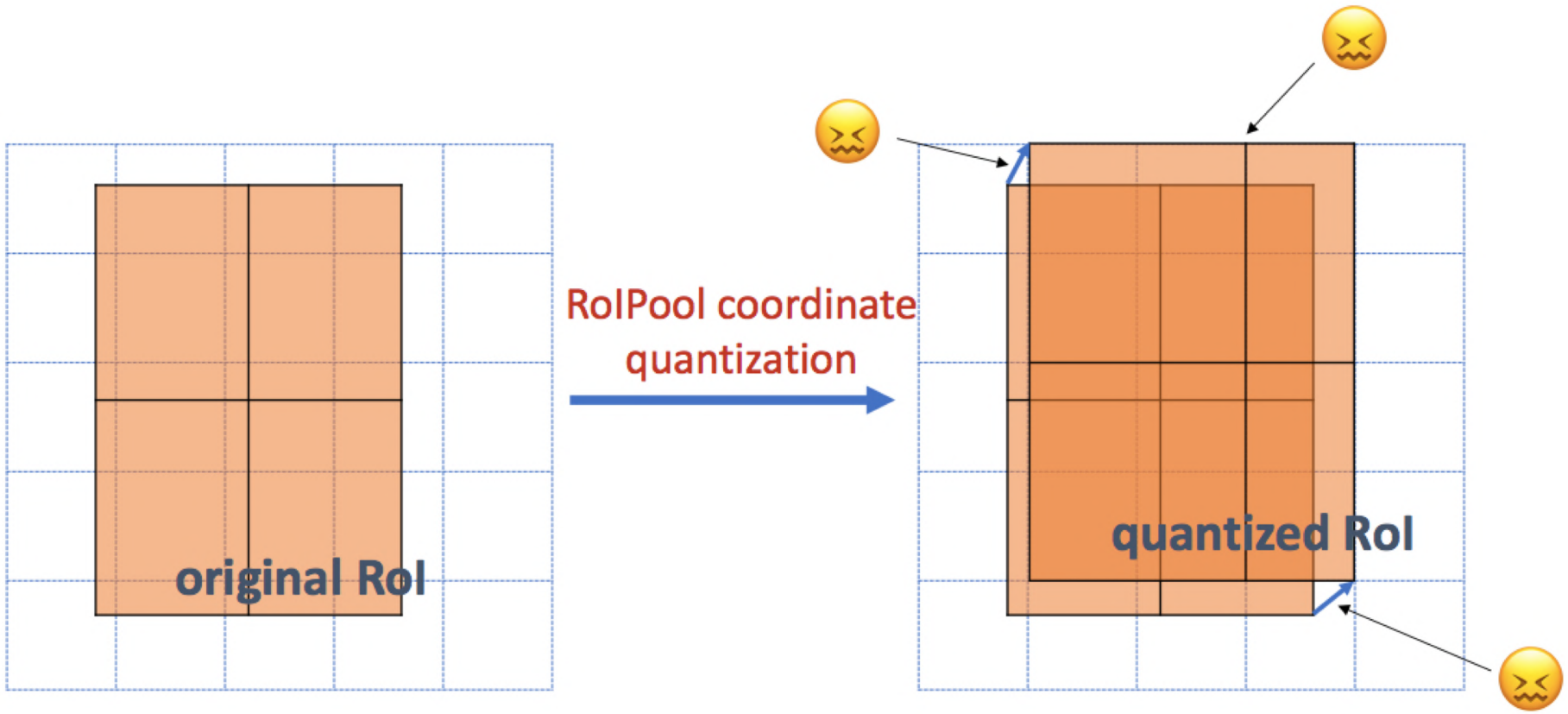
- RoIPool: Proposed in the original Faster R-CNN.
- Nearest neighbor quantization.
- Not translation-equivariant.



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

RoIAlign vs. RoIPool

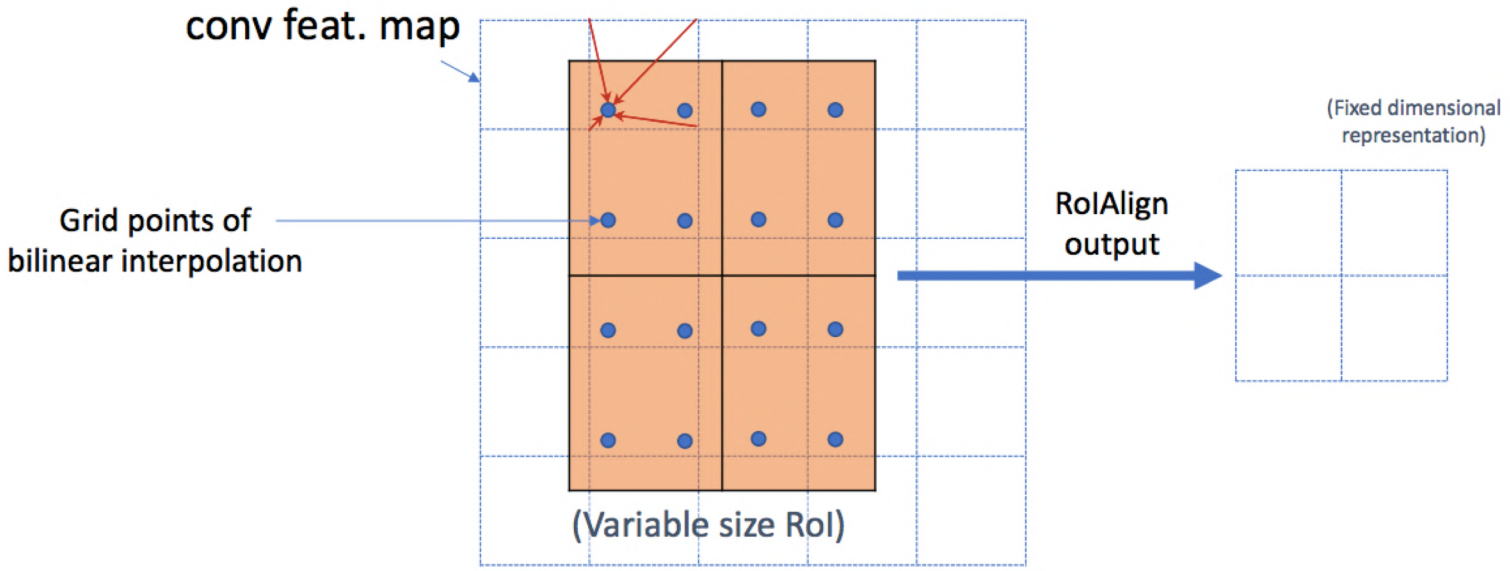
- RoIPool: nearest neighbor quantization



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

RoIAlign vs. RoIPool

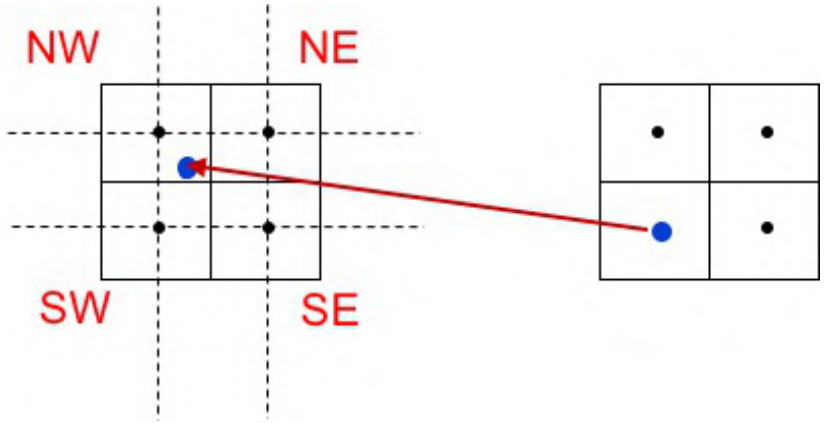
- RoIAlign: Suggested in Mask R_CNN
- Bilinear interpolation then Max-pooling



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

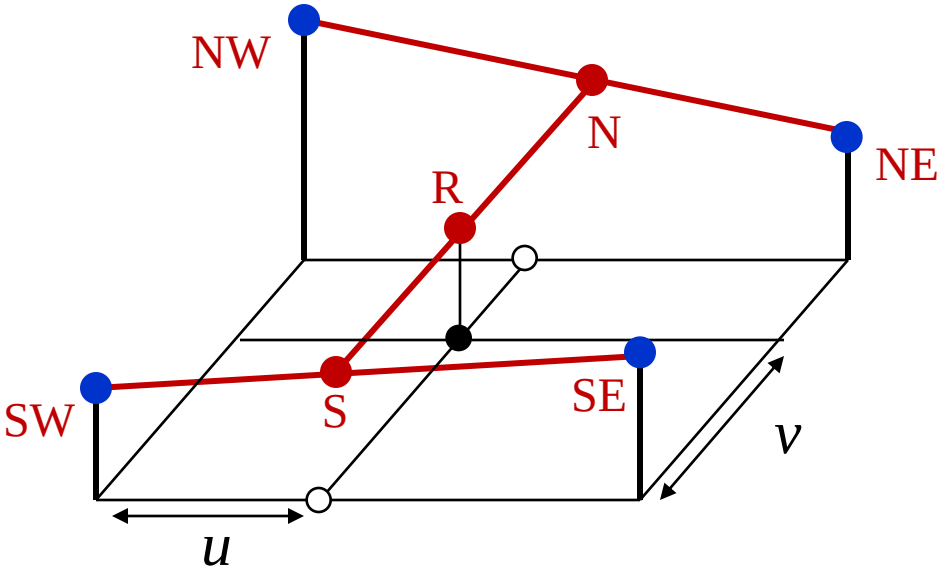
Bilinear interpolation

- Upsampling is performed using **bilinear interpolation**.



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

Bilinear interpolation



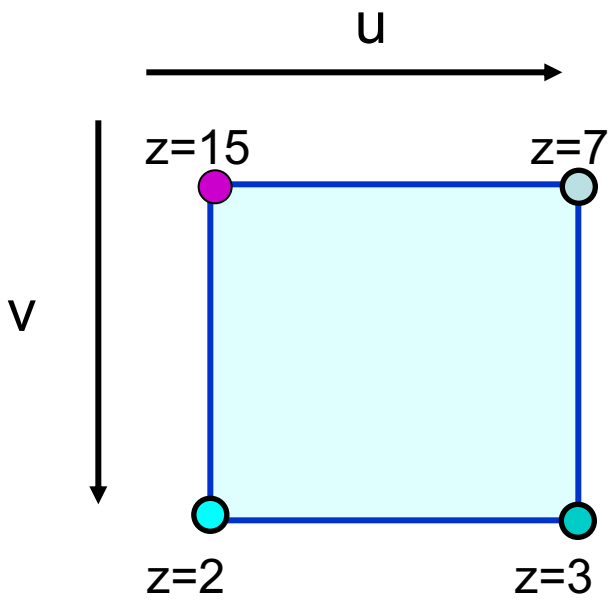
$$u, v \in [0,1]$$

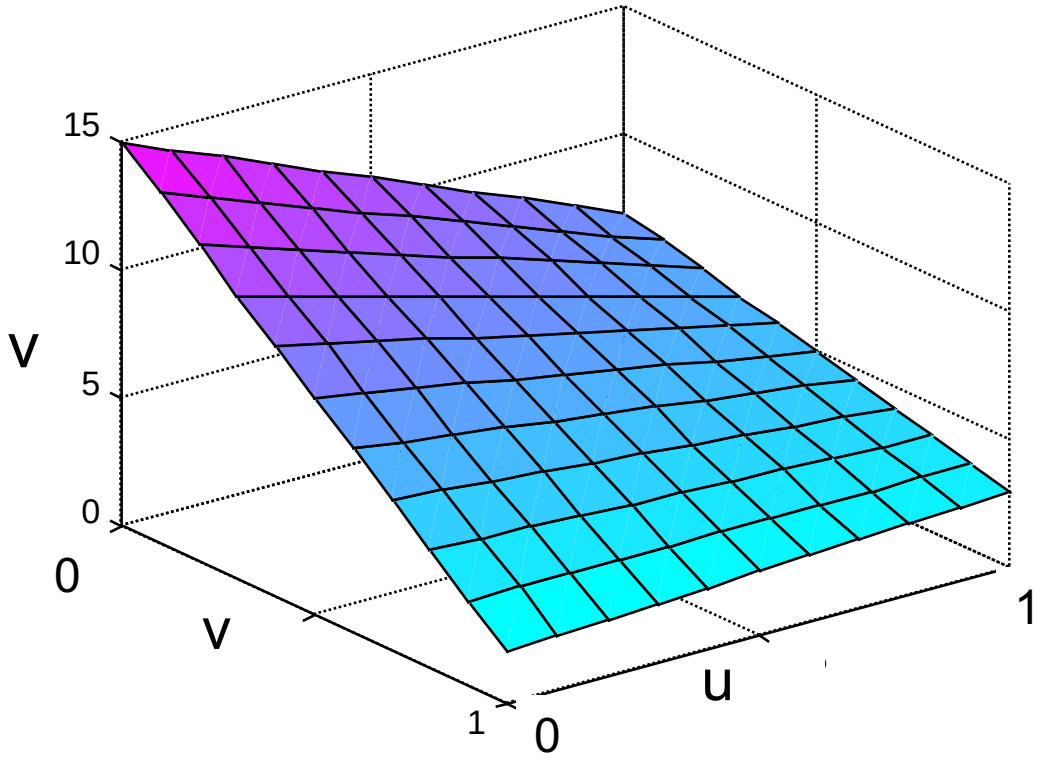
$$S(u) = u\,SE + (1-u)\,SW$$

$$N(u) = u\,NE + (1-u)\,NW$$

$$R(u,v) = v\,N(u) + (1-v)\,S(u)$$

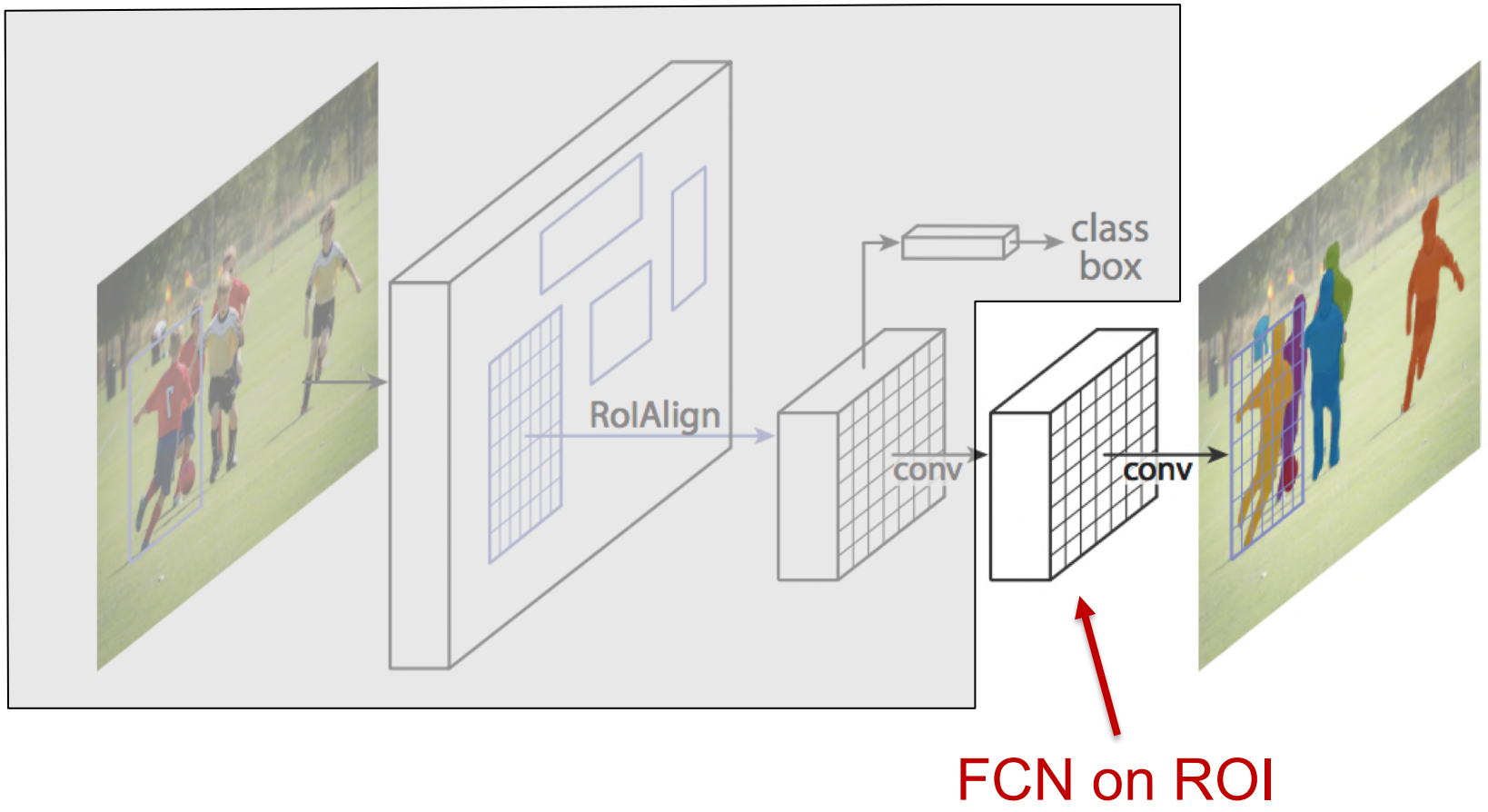
Bilinear example





Mask R-CNN

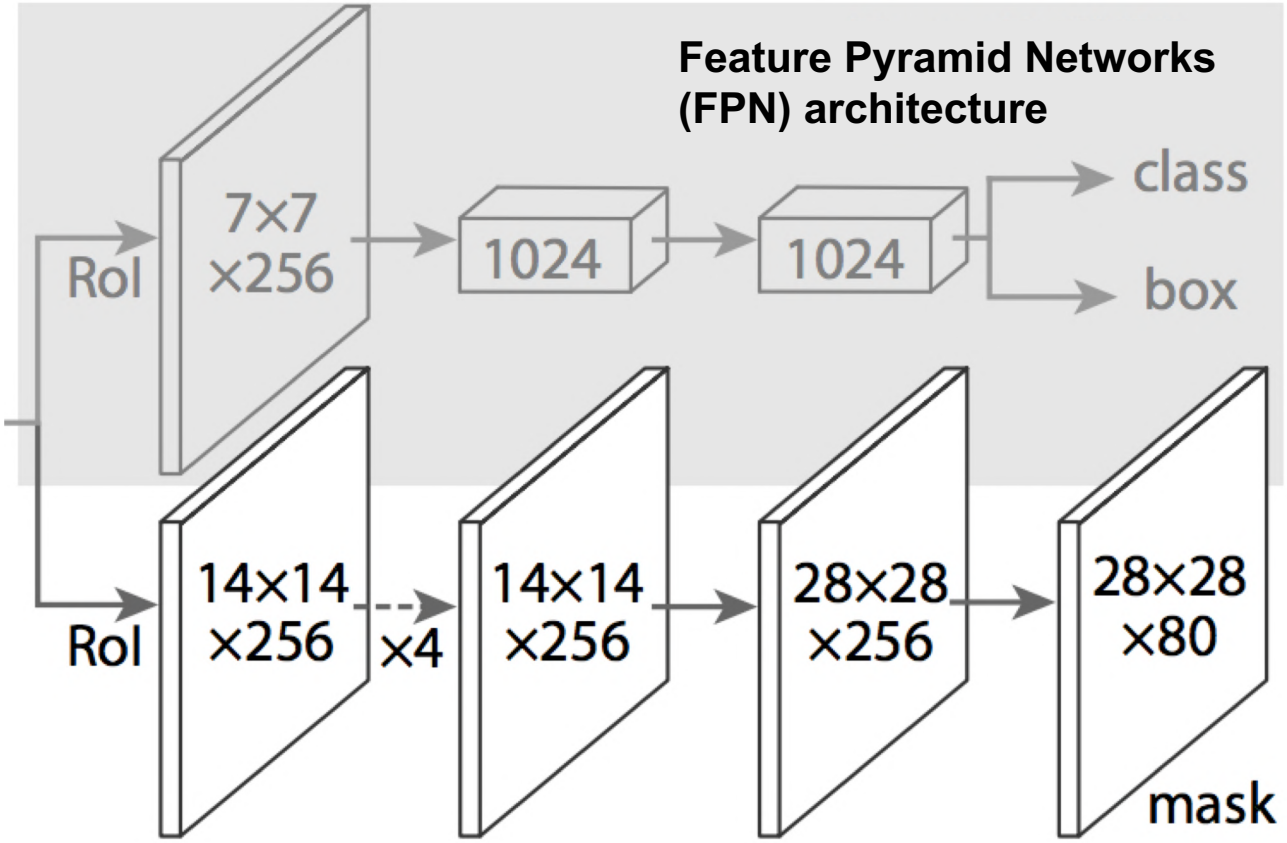
- Mask R-CNN = Faster R-CNN + FCN on RoIs



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

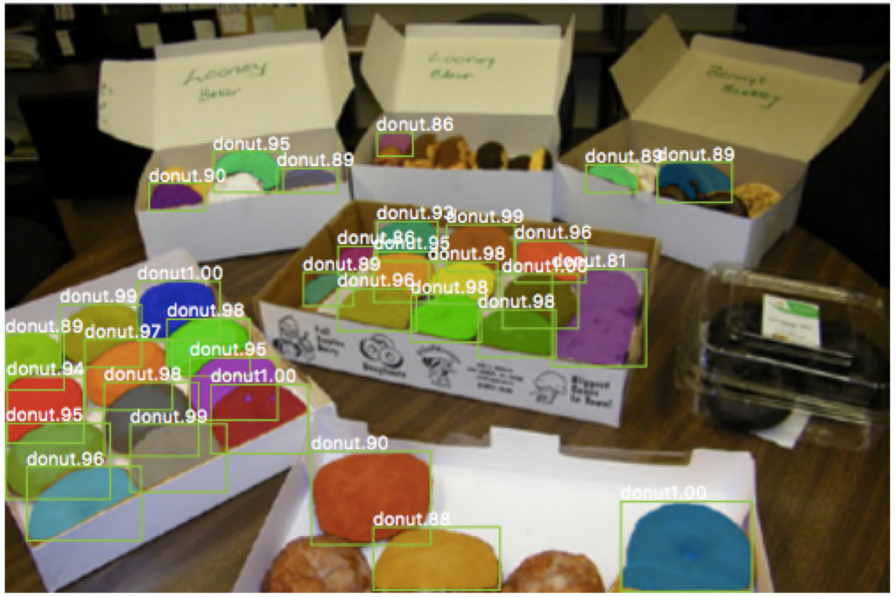
Mask R-CNN

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

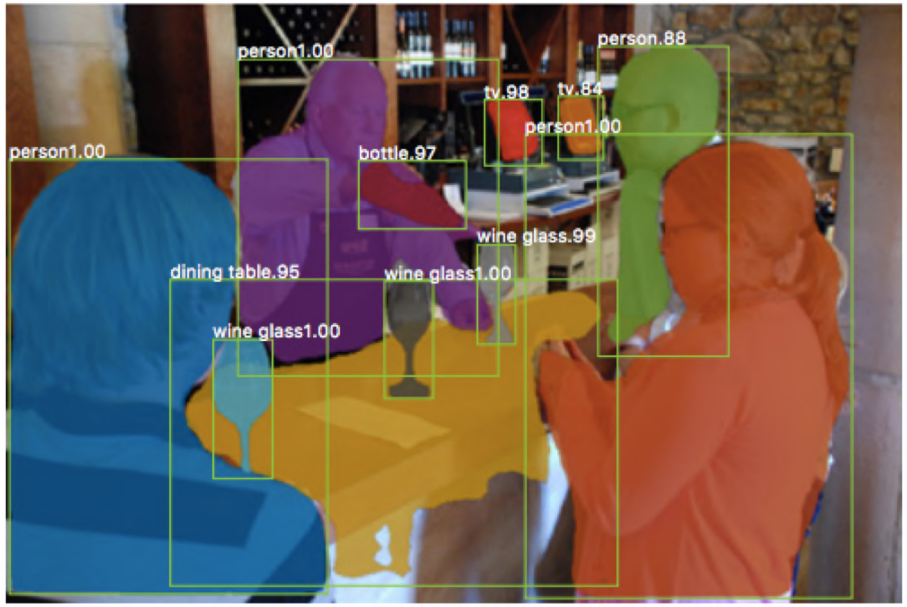


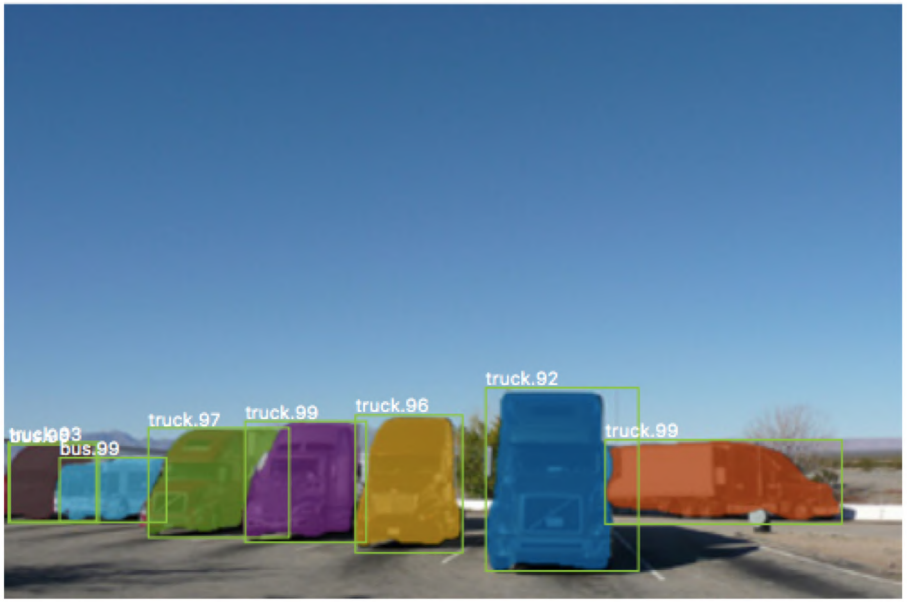
K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)

Example results



Example results





Example results

- Mask R-CNN also estimates human pose

Figure 7. Keypoint detection results on COCO `test` using Mask R-CNN (ResNet-50-FPN), with person segmentation masks predicted from the same model. This model has a keypoint AP of 63.1 and runs at 5 fps.

Instance segmentation results on COCO

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

AP at different IoU thresholds

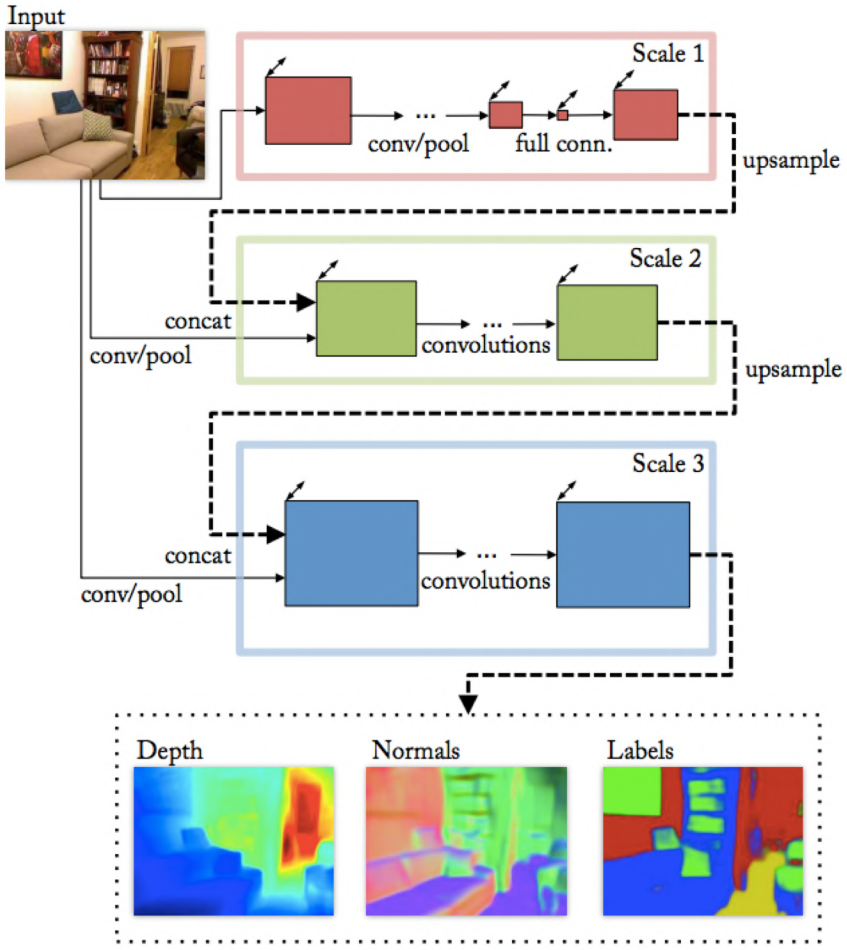
AP for different size instances

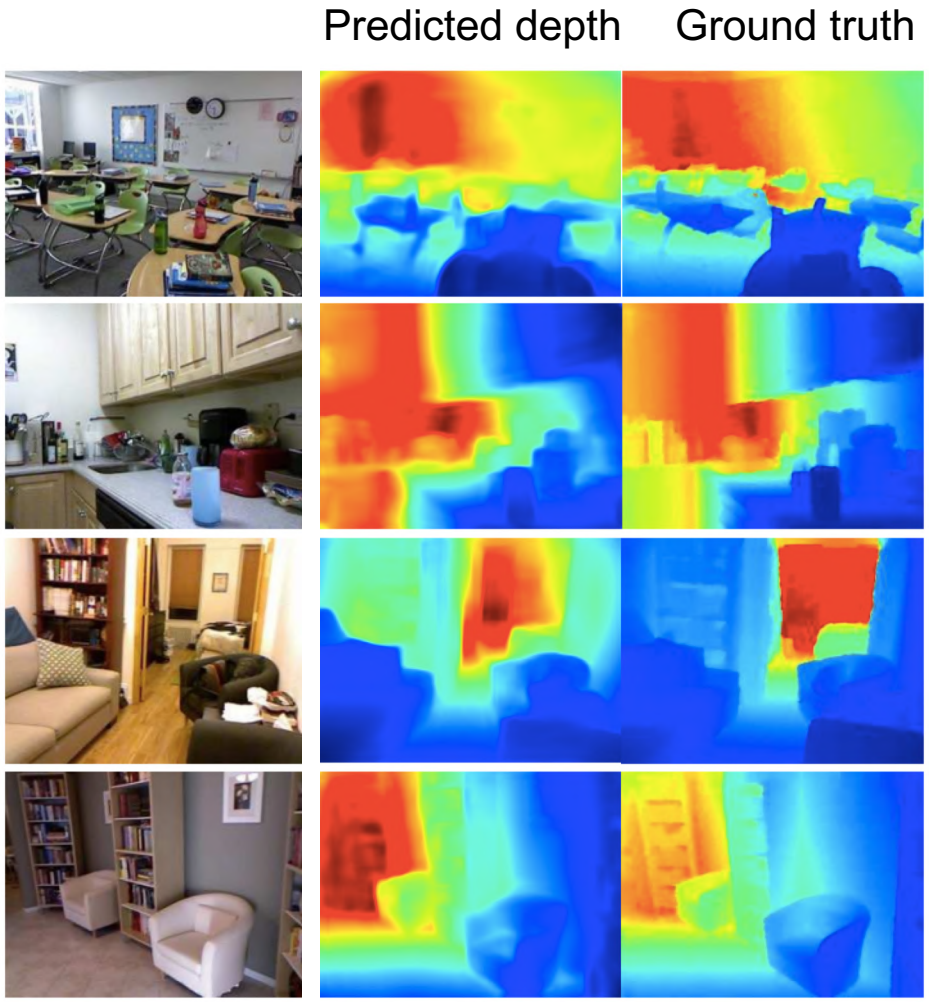
K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)

Other Dense Prediction Tasks

- Depth estimation
- Surface normal estimation
- Colorization
- ….

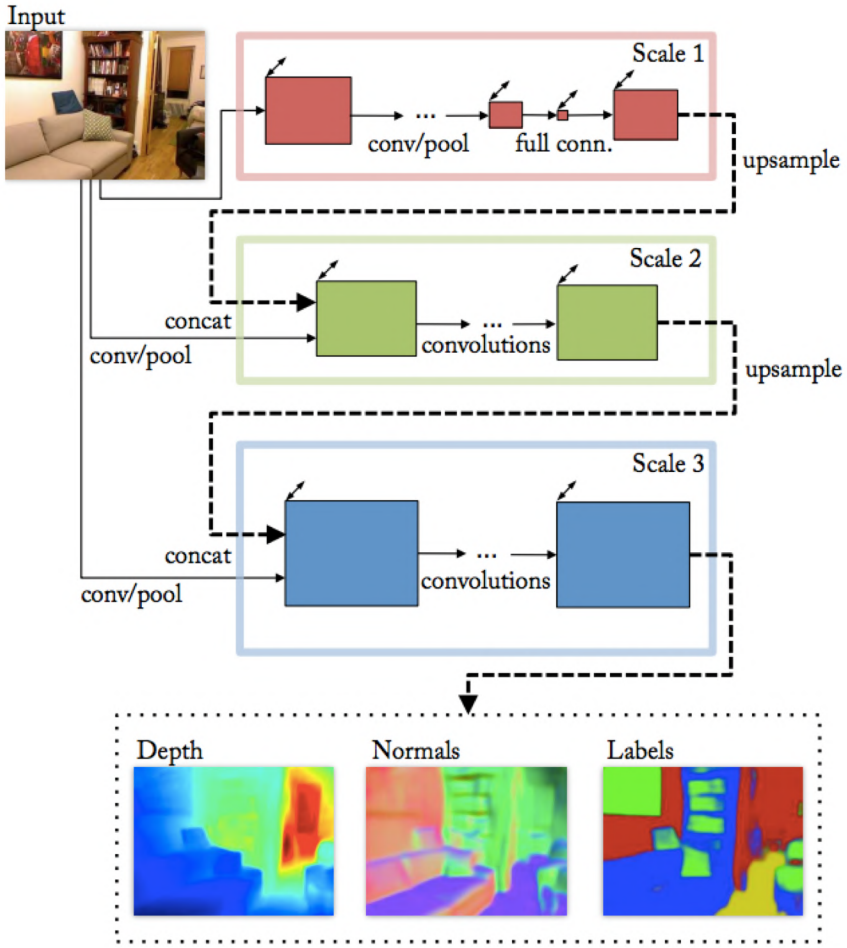
Depth and normal estimation

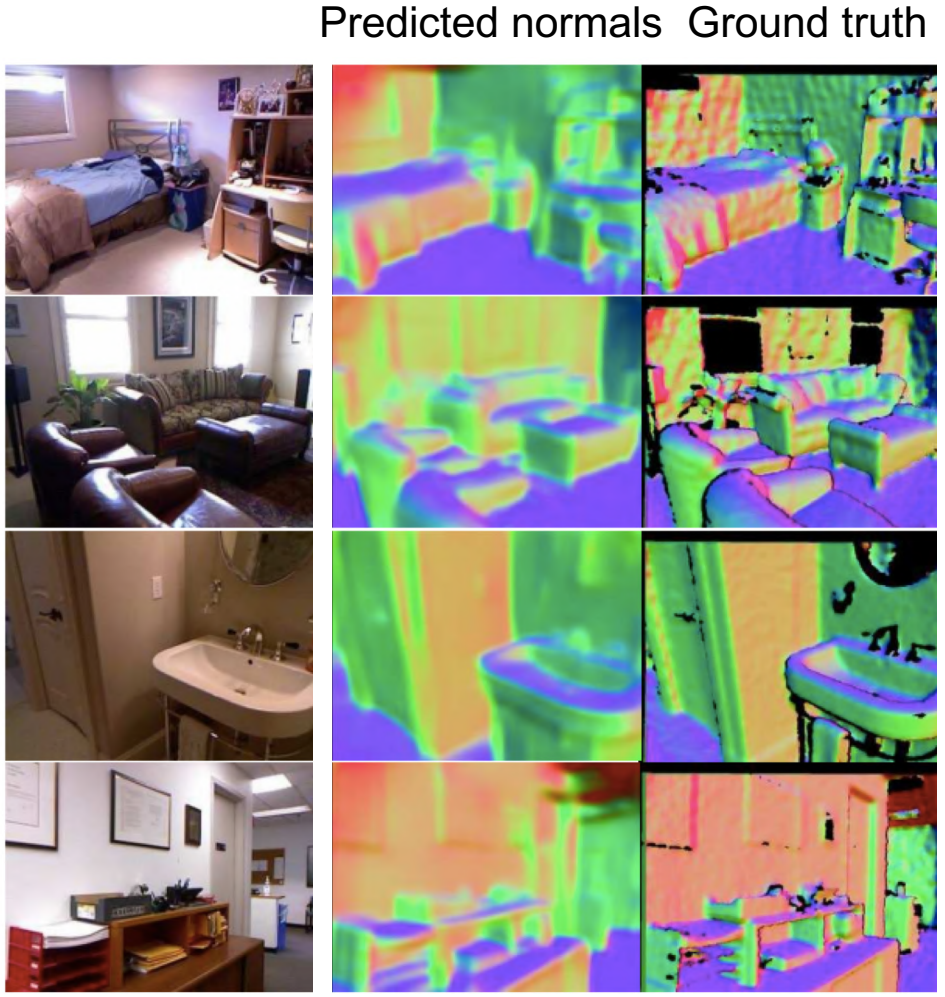




D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015

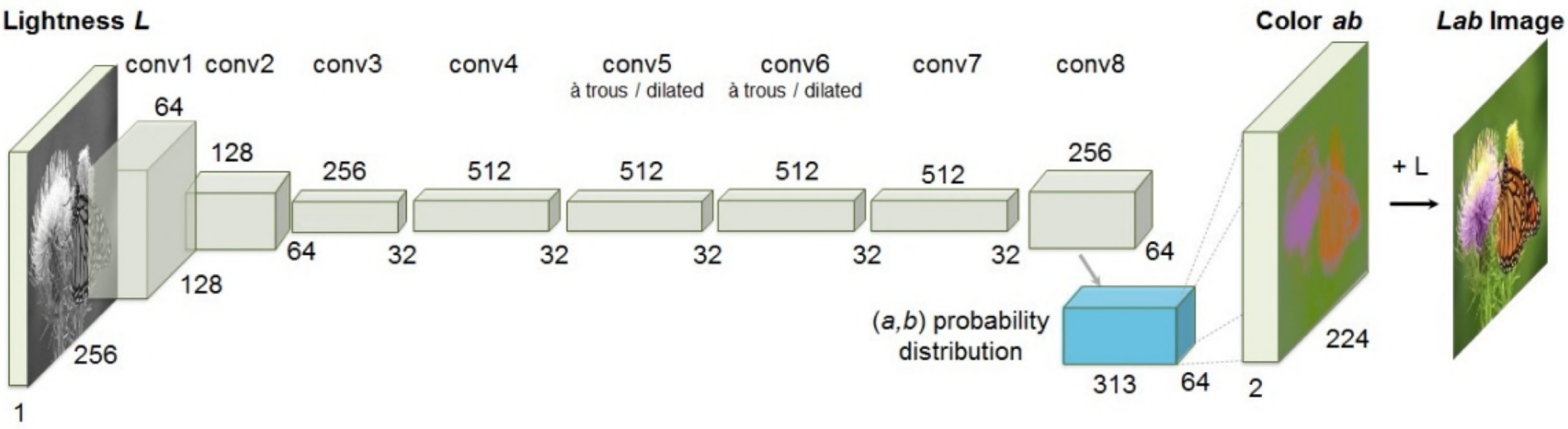
Depth and normal estimation

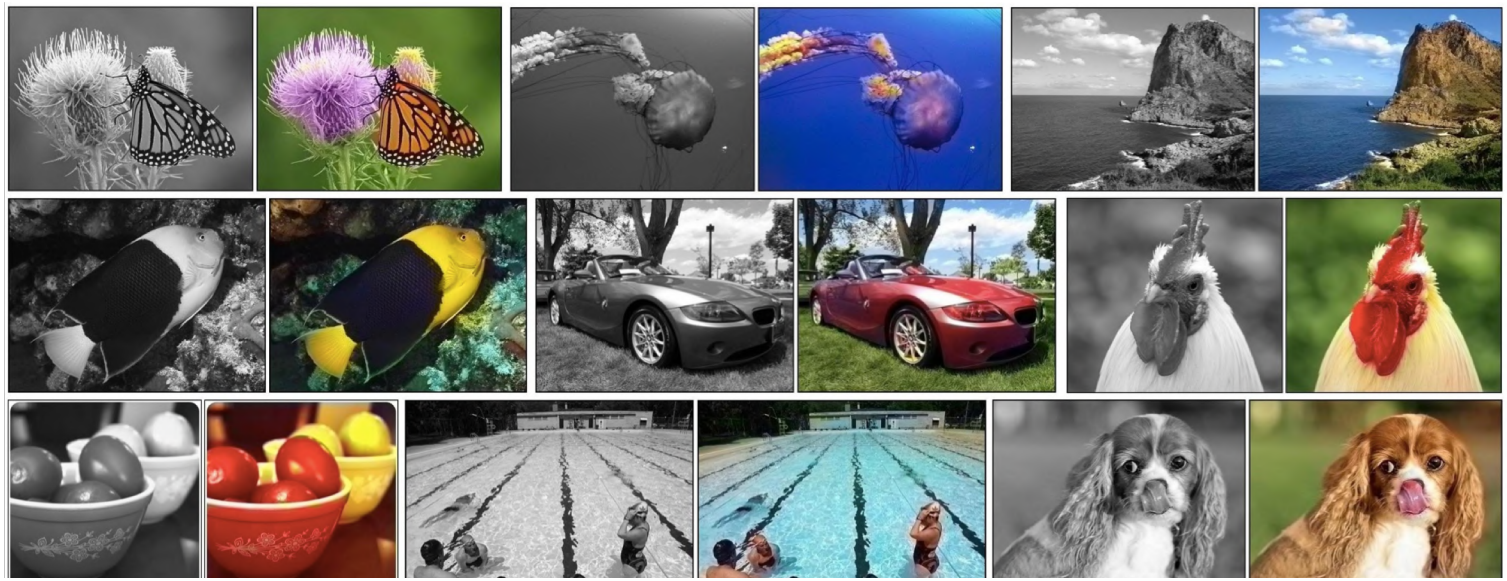




D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015

Colorization





R. Zhang, P. Isola, and A. Efros, Colorful Image Colorization, ECCV 2016

Summary

Semantic segmentation

- Classify all pixels (sliding window)
- Fully convolutional models, downsample then upsample
- Learnable upsampling: fractionally strided convolution
- Skip connections can help

Instance Segmentation

- Detect instance, generate mask
- Similar pipelines to object detection

Other dense prediction:

- Color, depth, normal, material, intrinsic properties, tissues, tumors, etc.

THE END