

Panoptic Segmentation

Luiz Velho

Slides from Fei Fei Li and Justin Johnson

Computer Vision Tasks

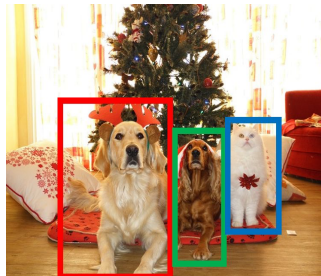
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

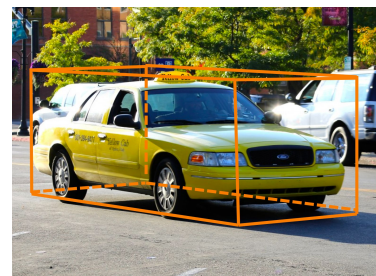
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

Object categories +
3D bounding boxes

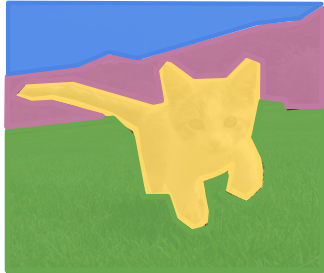
2D Segmentation + *Detection (2D / 3D)*

Panoptic Segmentation

[This image is CC0 public domain](#)

Semantic Segmentation

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

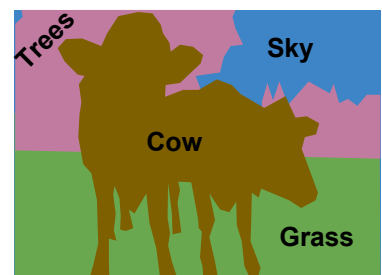
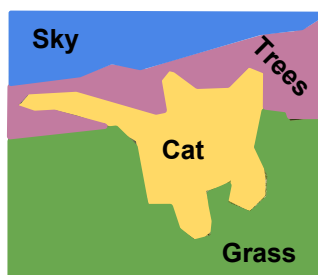
Object categories +
3D bounding boxes

This image is CC0 public domain

Semantic Segmentation

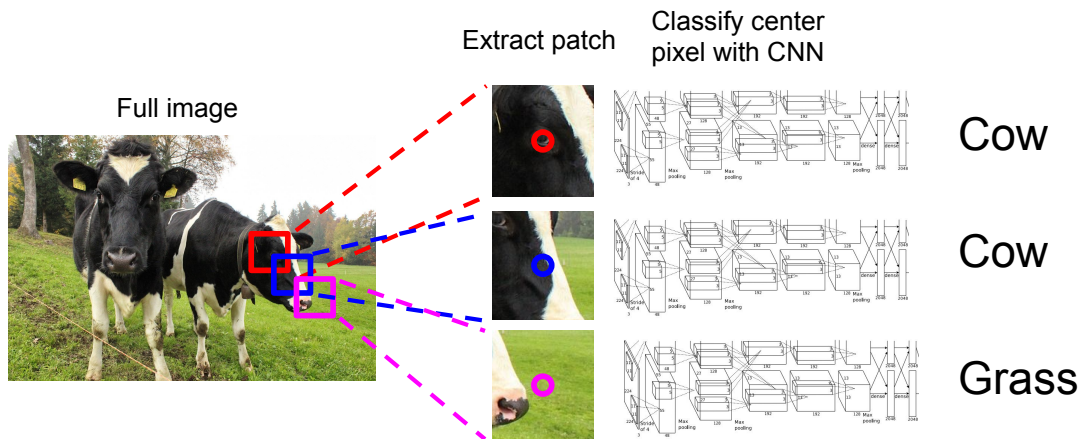
Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



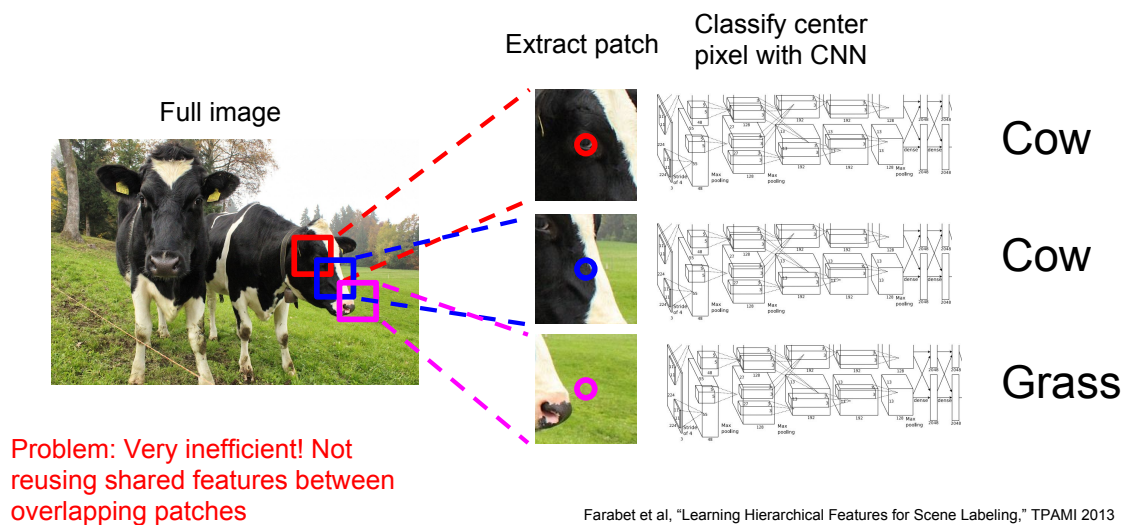
This image is CC0 public domain

Semantic Segmentation Idea: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

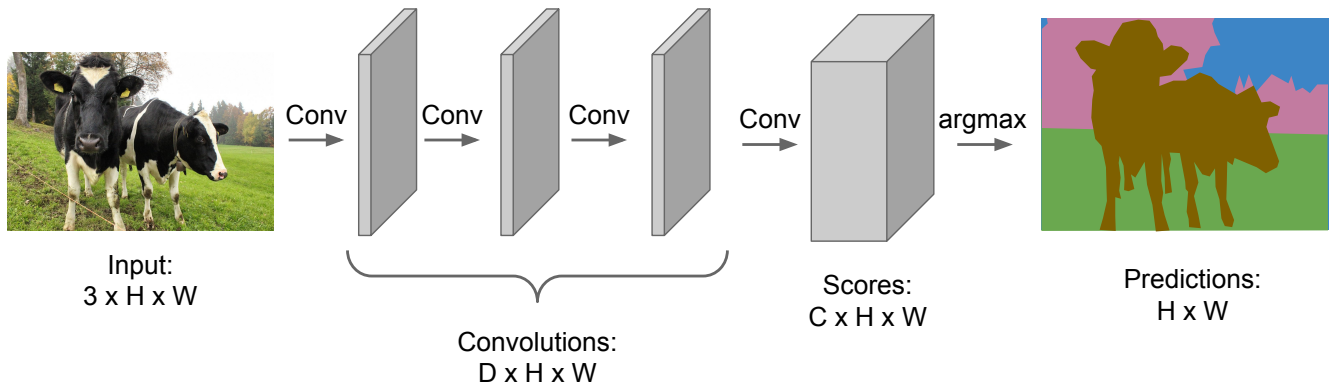
Semantic Segmentation Idea: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

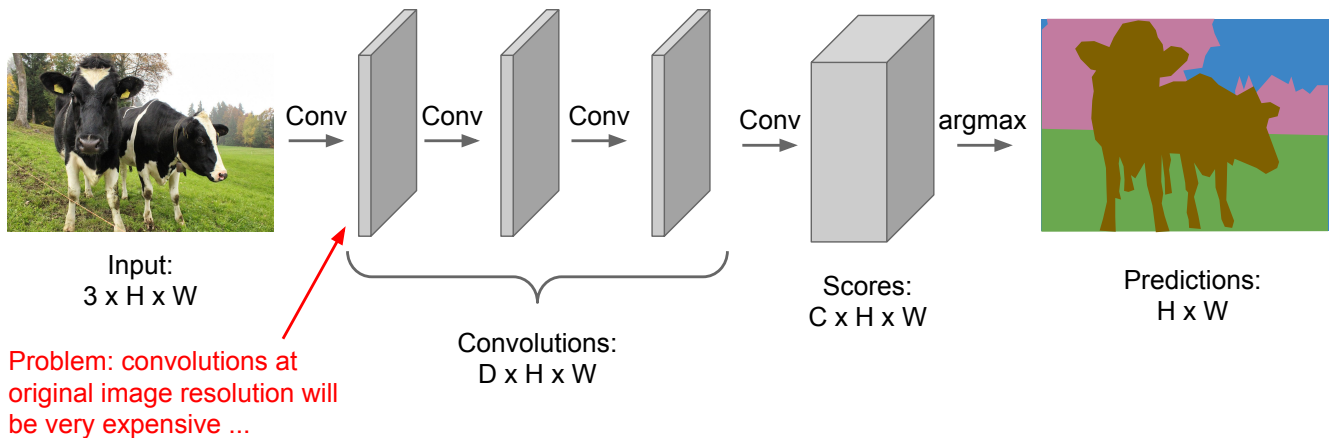
Semantic Segmentation Idea: Fully Convolutional

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



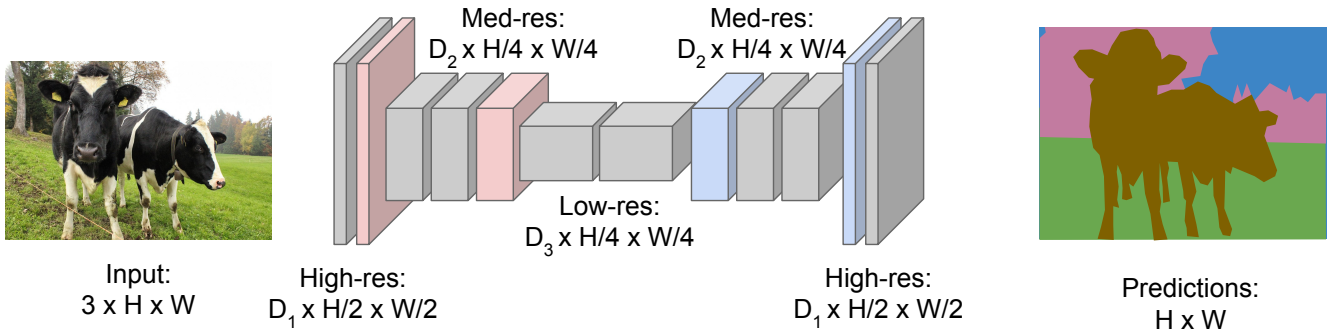
Semantic Segmentation Idea: Fully Convolutional

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



Semantic Segmentation Idea: Fully Convolutional

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



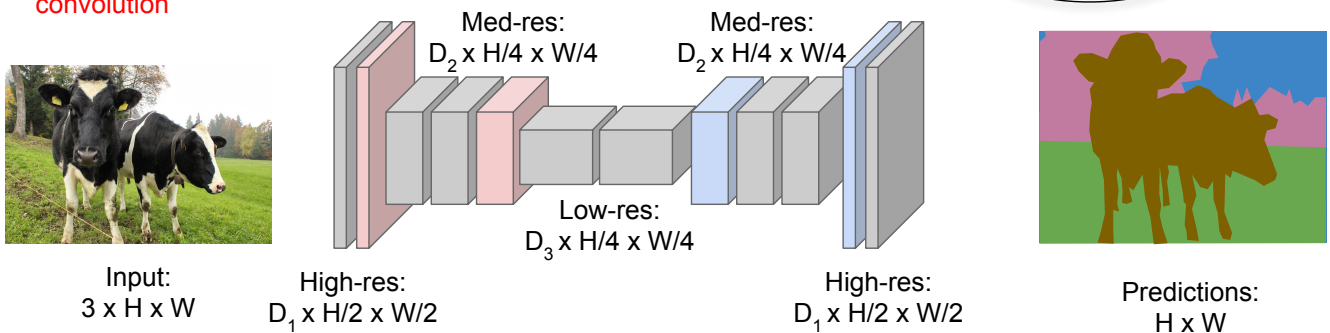
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
 Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation Idea: Fully Convolutional

Downsampling:
 Pooling, strided
 convolution

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

Upsampling:
 ???



OBS: similar to auto-encoders

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”

Nearest Neighbor

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling

Use positions from pooling layer

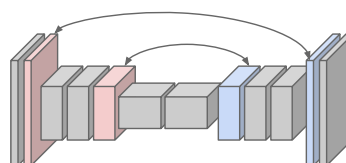
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

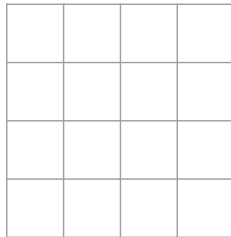
Output: 4 x 4

Corresponding pairs of
downsampling and
upsampling layers

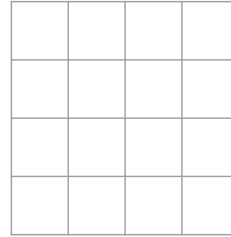


Learnable Upsampling: Transpose Convolution

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



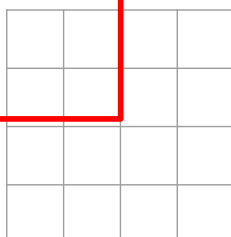
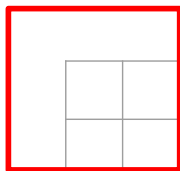
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling: Transpose Convolution

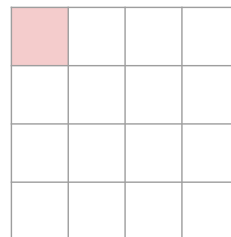
Recall: Normal 3 x 3 convolution, stride 1 pad 1



Input: 4 x 4



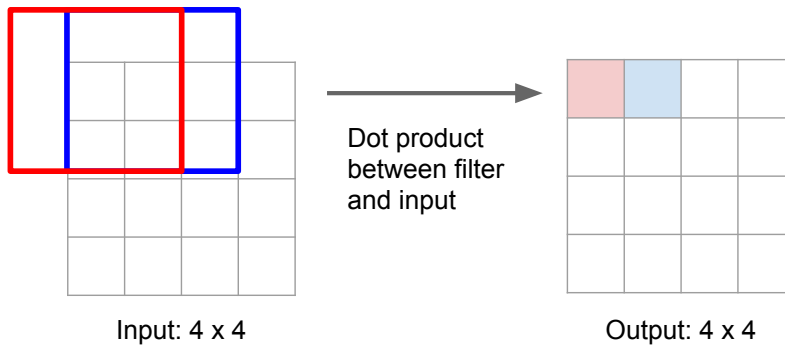
Dot product
between filter
and input



Output: 4 x 4

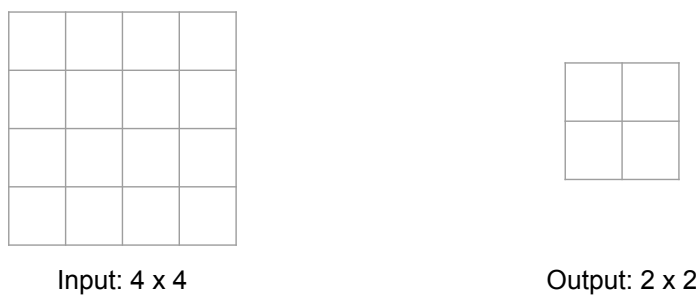
Learnable Upsampling: Transpose Convolution

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



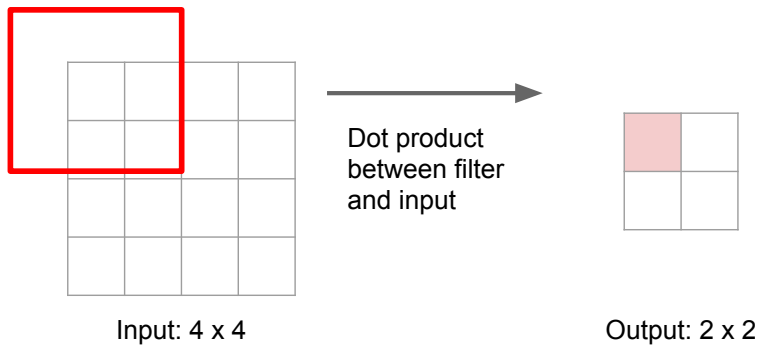
Learnable Upsampling: Transpose Convolution

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



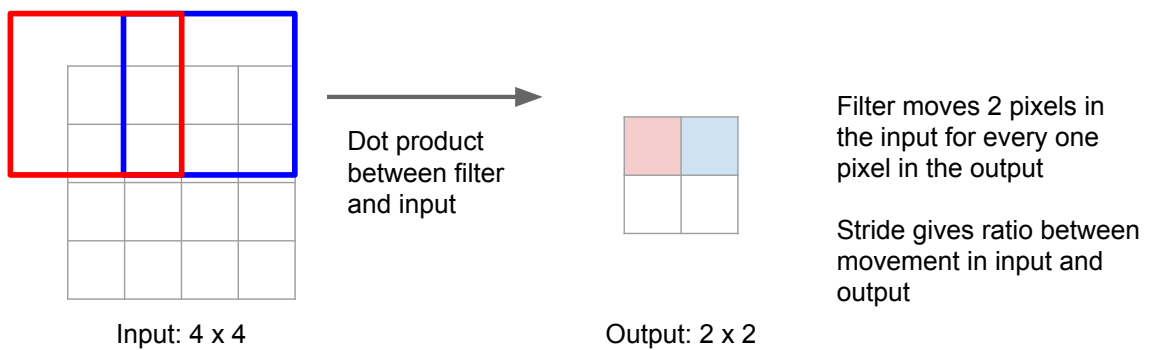
Learnable Upsampling: Transpose Convolution

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



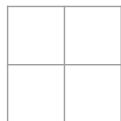
Learnable Upsampling: Transpose Convolution

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

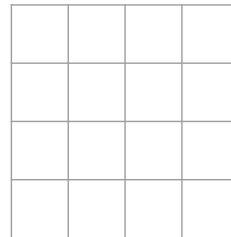


Learnable Upsampling: Transpose Convolution

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



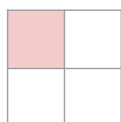
Input: 2 x 2



Output: 4 x 4

Learnable Upsampling: Transpose Convolution

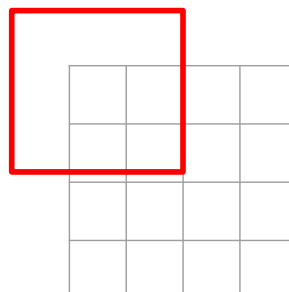
3 x 3 **transpose** convolution, stride 2 pad 1



Input: 2 x 2

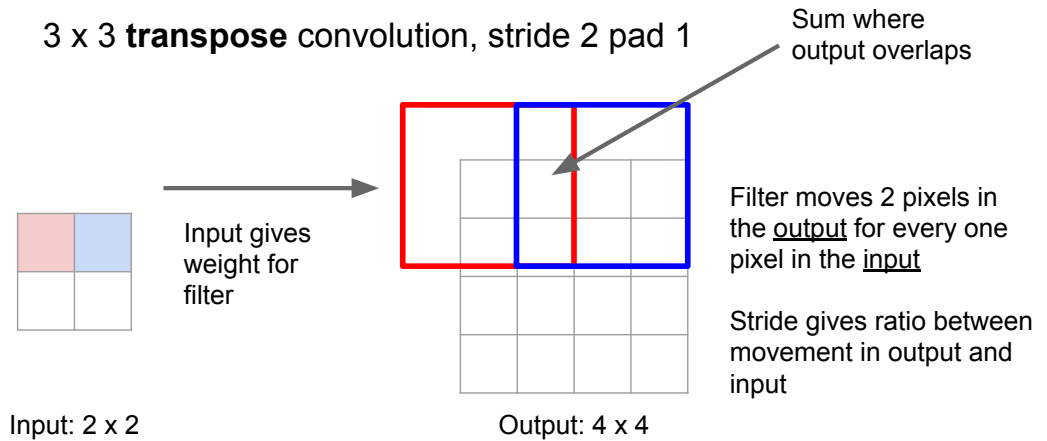


Input gives
weight for
filter



Output: 4 x 4

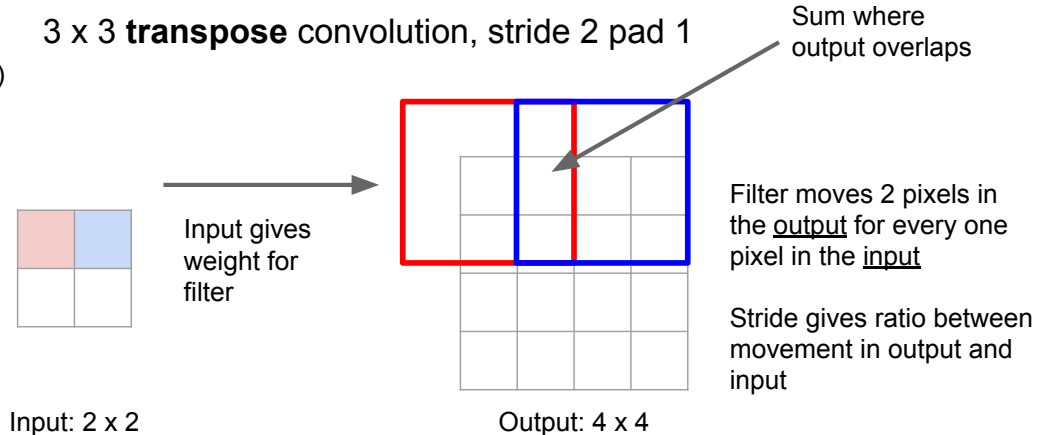
Learnable Upsampling: Transpose Convolution



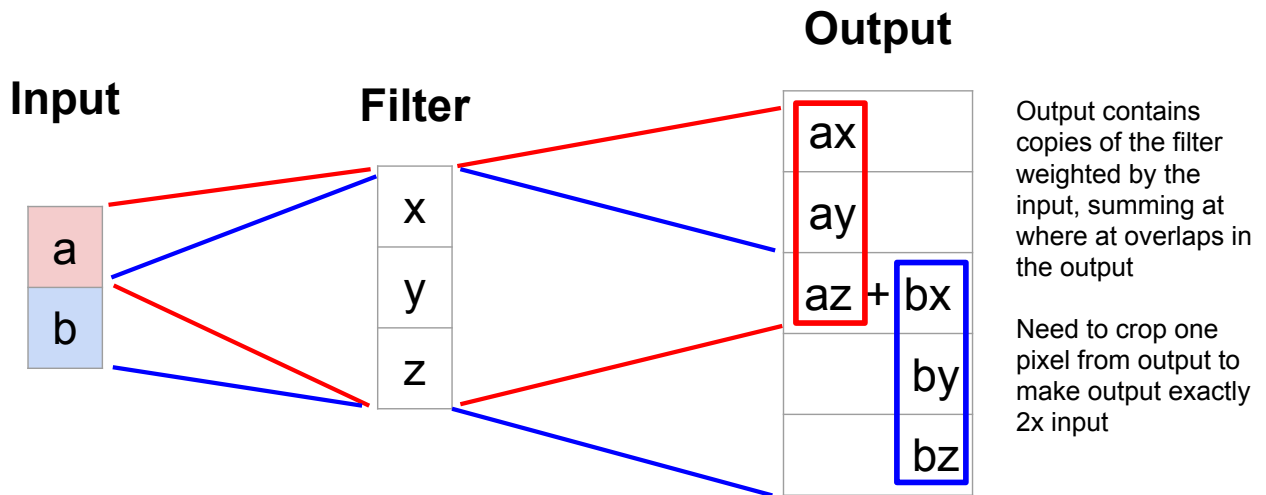
Learnable Upsampling: Transpose Convolution

Other names:

- Deconvolution (bad)
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution



Learnable Upsampling: 1D Example



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

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

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

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

Convolution transpose 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, convolution transpose is just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

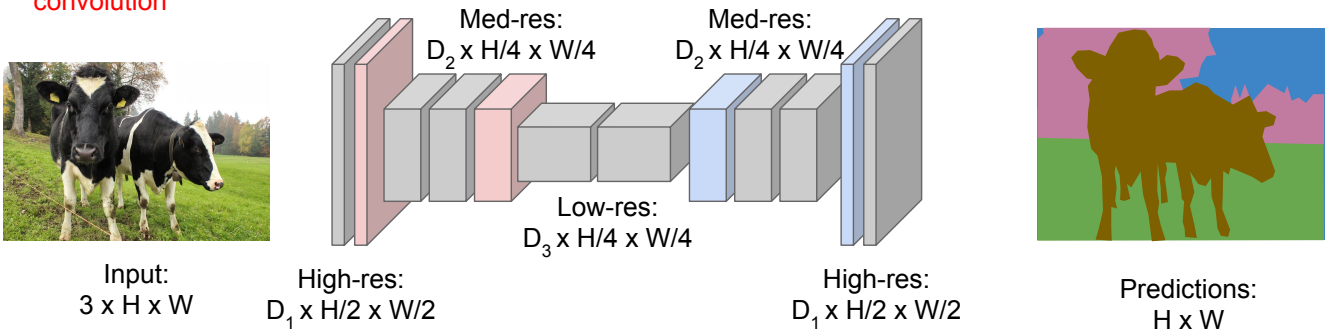
When stride>1, convolution transpose is no longer a normal convolution!

Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transpose convolution



2D Object Detection

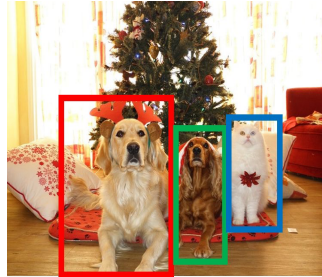
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection

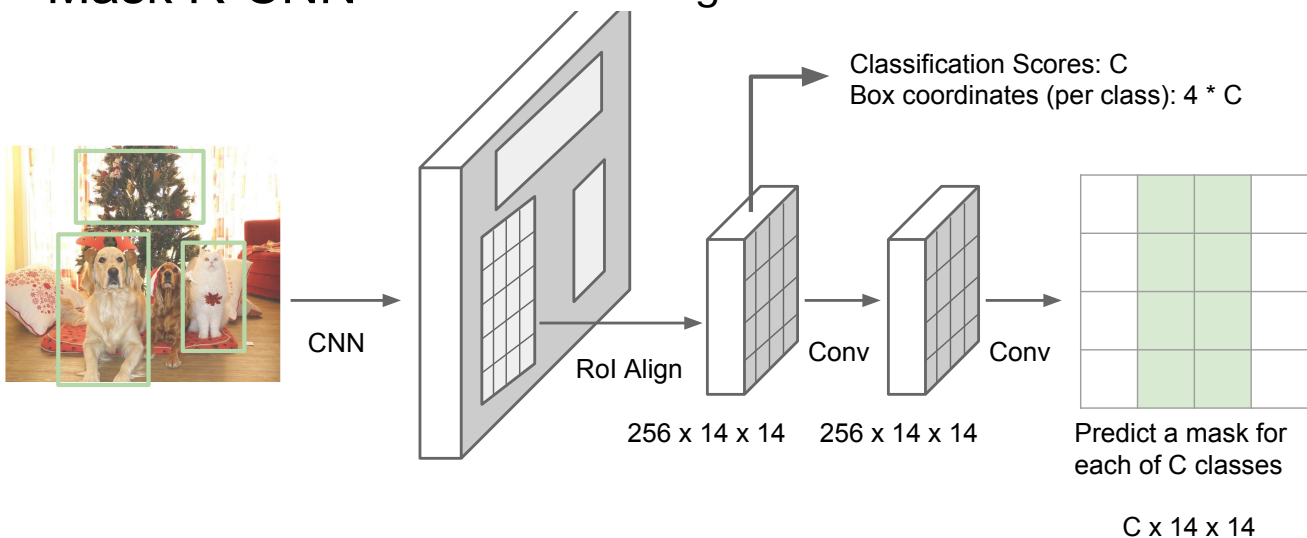


Car

Object categories +
3D bounding boxes

This image is CC0 public domain

Mask R-CNN : Detection + Segmentation



Mask R-CNN: Very Good Results!



He et al, "Mask R-CNN", arXiv 2017
 Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
 Reproduced with permission.

Object Detection: Lots of variables ...

Base Network

VGG16
 ResNet-101
 Inception V2
 Inception V3
 Inception
 ResNet
 MobileNet

Object Detection architecture

Faster R-CNN
 R-FCN
 SSD

Image Size

Region Proposals

...

Takeaways

Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
 Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
 Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
 Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
 MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

3D Object Detection

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

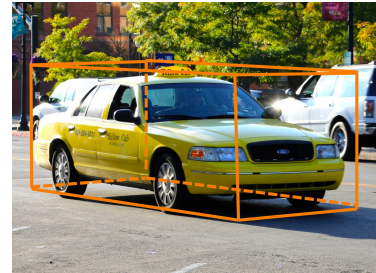
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

Object categories +
3D bounding boxes

This image is CC0 public domain

Simplified Camera Model

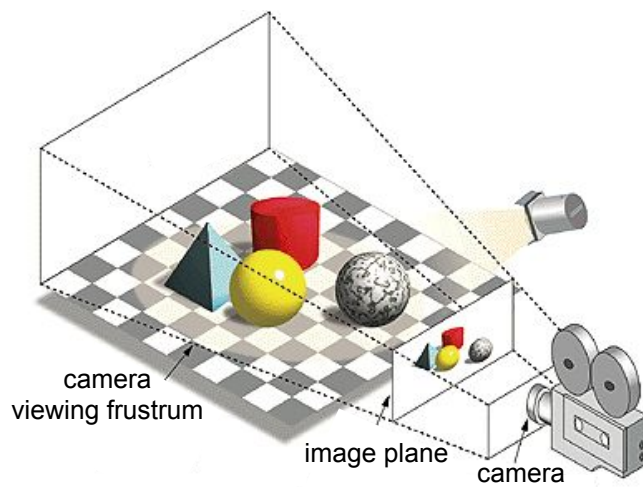
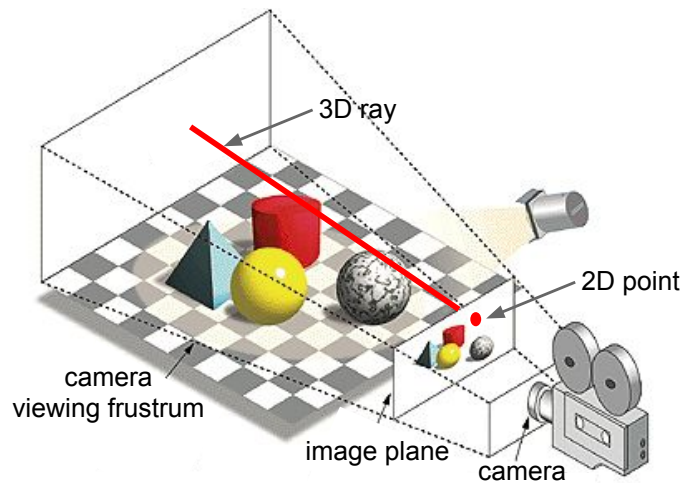


Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

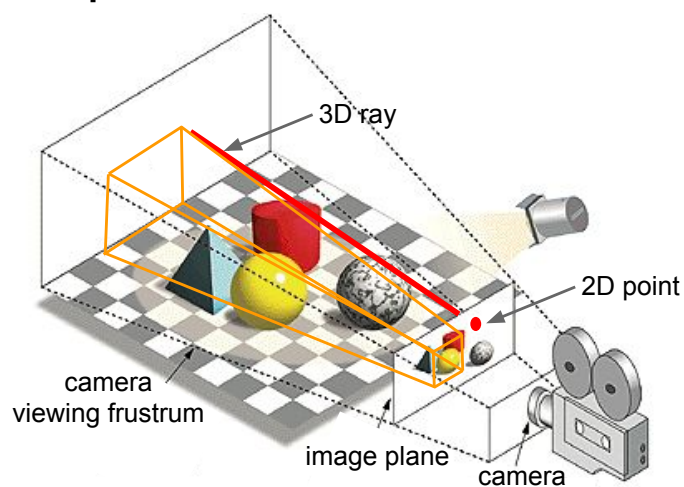
Simplified Camera Model



A point on the image plane corresponds to a **ray** in the 3D space

Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

Simplified Camera Model

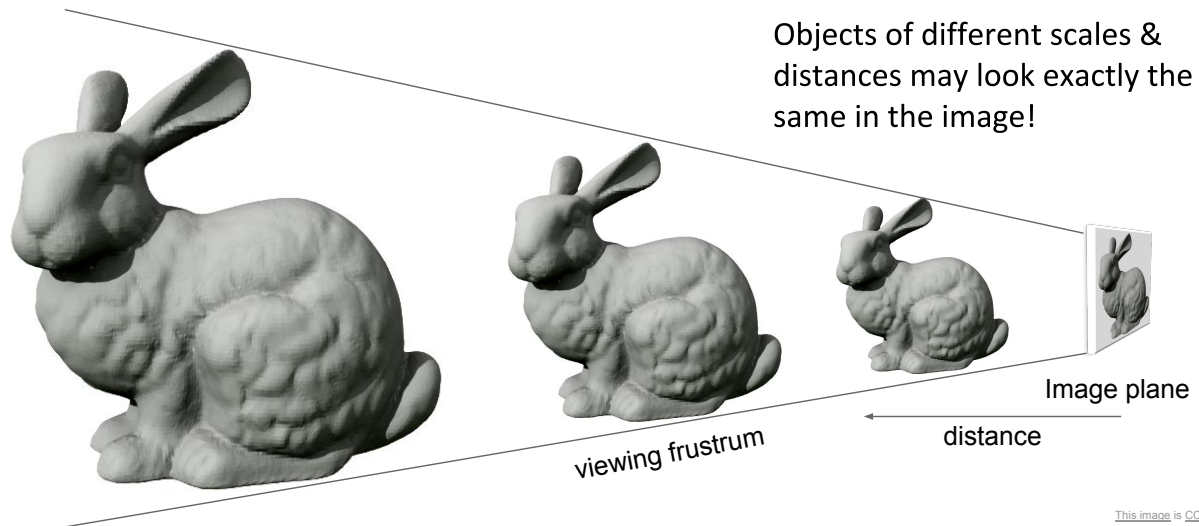


A point on the image plane corresponds to a **ray** in the 3D space

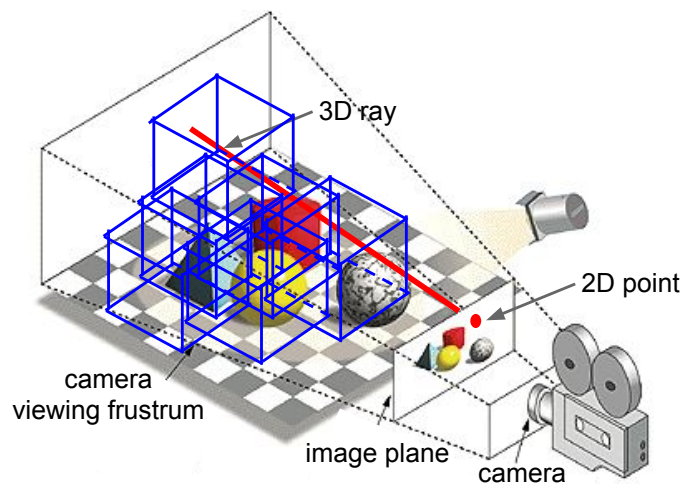
A 2D bounding box on an image is a **frustrum** in the 3D space

Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

Scale & Distance Ambiguity



Simplified Camera Model



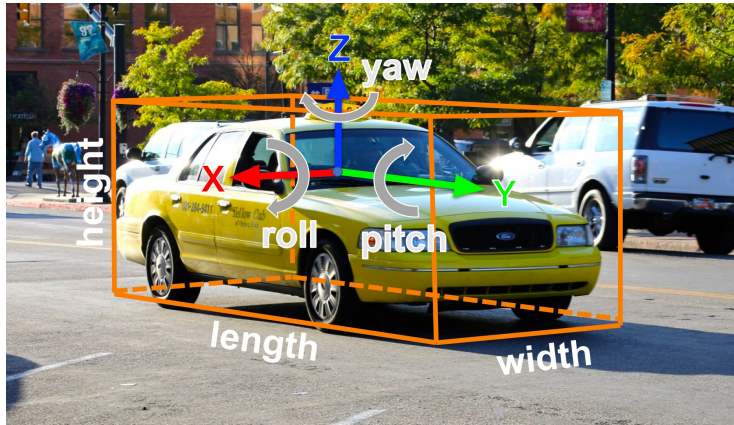
A point on the image plane corresponds to a **ray** in the 3D space

A 2D bounding box on an image is a **frustum** in the 3D space

Localize an object in 3D:
The object can be anywhere in the **camera viewing frustum**!

Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

3D Object Detection



2D Object Detection:

2D bounding box

(x, y, w, h)

3D Object Detection:

3D oriented bounding box

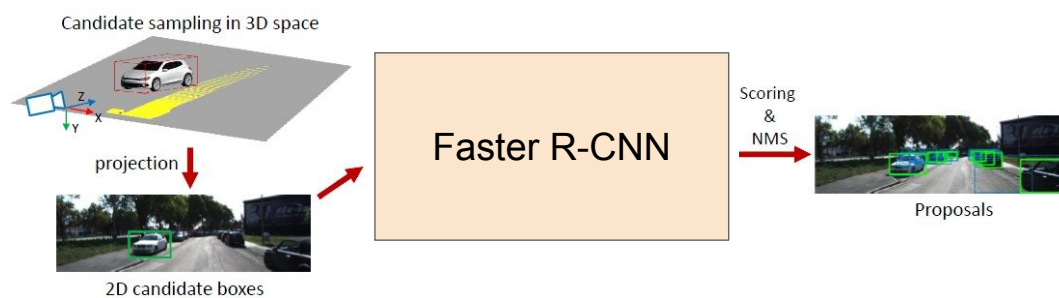
$(x, y, z, w, h, l, r, p, y)$

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

This image is CC0 public domain

3D Object Detection: Monocular Camera



- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

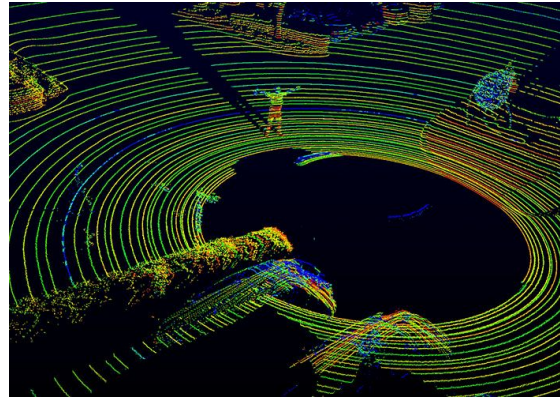
Chen, Xiaoqi, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "Monocular 3d object detection for autonomous driving." CVPR 2016.

3D Object Detection: Camera + LiDAR

This image is CC0 public domain

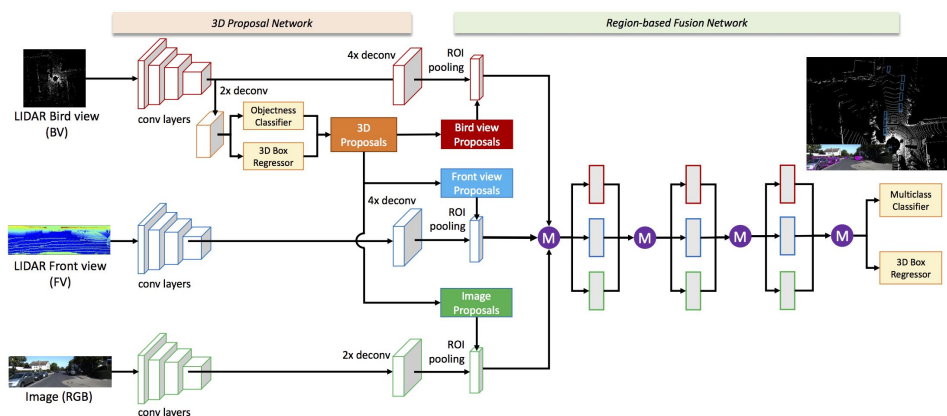


Velodyne (HDL-64e)



3D Point Cloud

3D Object Detection: Camera + LiDAR



- Combine 3D proposals from multiple views & sensors
- regress 3D box parameters + class score

Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." CVPR 2017

3D Object Detection: Camera + LiDAR



Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." *CVPR* 2017

Multi-View 3D Object Detection Detection Network for Autonomous Driving

Xiaozhi Chen¹, Huimin Ma¹, Ji Wan², Bo Li², Tian Xia²

¹Tsinghua University ²Baidu Inc.

Recap

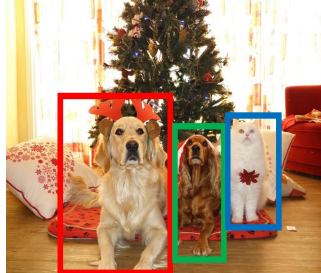
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

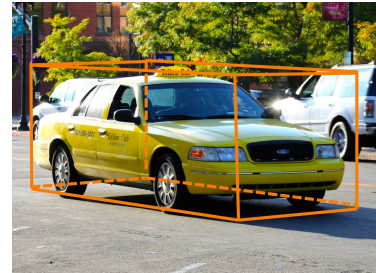
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

Object categories +
3D bounding boxes

[This image is CC0 public domain](#)

Detection + Segmentation (Mask R-CNN)

*What about
Semantic Instance Segmentation
of 3D Point Clouds ?*

