

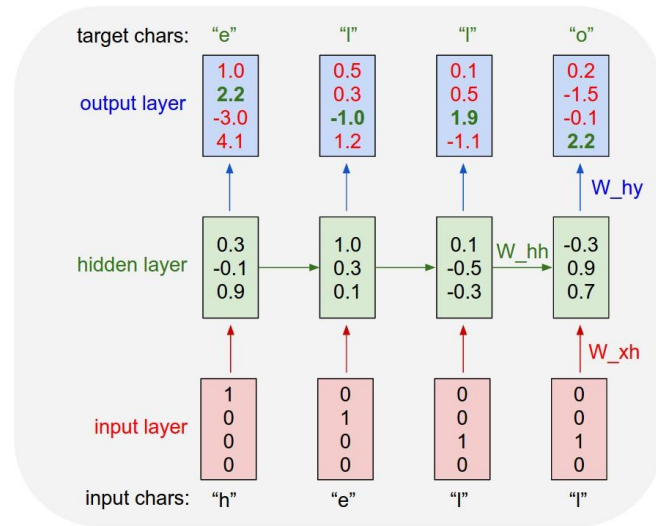
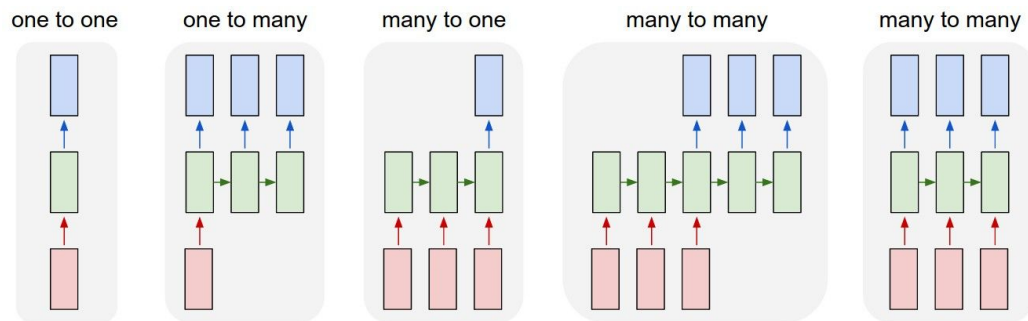
Lecture 11:

Detection and Segmentation

Administrative

1. Midterms being graded
 - **Please don't** discuss midterms until next week - some students not yet taken
2. A2 being graded
3. Project milestones due Wed 5/16

Last Time: Recurrent Networks



Last Time: Recurrent Networks

For $\bigoplus_{i=1,\dots,m}$ where $\mathcal{L}_{m,*} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparico in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ?? . Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. $\square$

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{A}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$ -modules on $\mathcal{C}$ as a $\mathcal{F}$ -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\tilde{M}^* = \mathcal{I}^* \otimes_{\text{Spec}(k)} \mathcal{O}_{S,*} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. $\square$

The result to prove any open covering follows from the less of Example ?? . It may replace S by $X_{\text{spaces}, \text{étale}}$ which gives an open subspace of X and T equal to S_{zar} , see Descent, Lemma ?? . Namely, by Lemma ?? we see that R is geometrically regular over S .

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000ffffffff) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

Last Time: Recurrent Networks

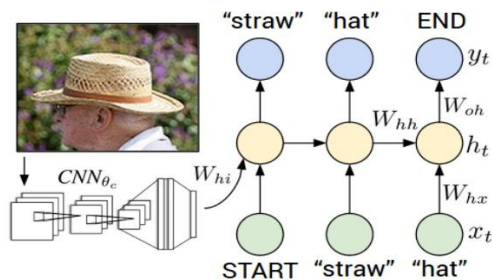


Figure from Karpathy et al., "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015.
Reproduced for educational purposes.



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A woman is holding a cat in her hand



Two people walking on the beach with surfboards



A tennis player in action on the court



A person holding a computer mouse on a desk

So far: Image Classification



This image is CC0 public domain

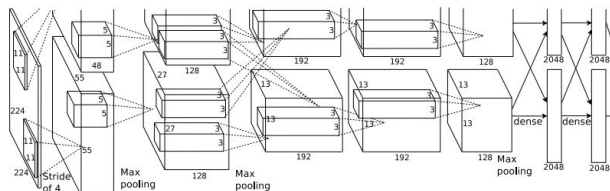


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

→
Fully-Connected:
4096 to 1000

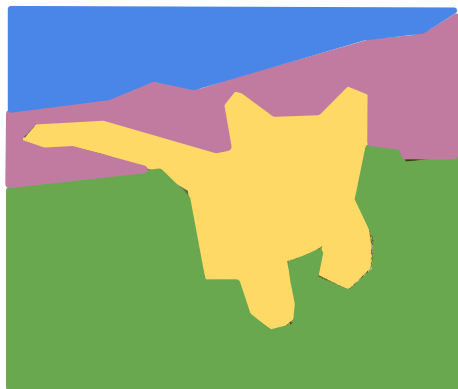
Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Today: Detection, Segmentation

Other Computer Vision Tasks

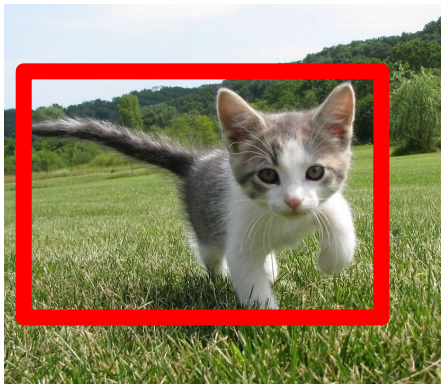
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

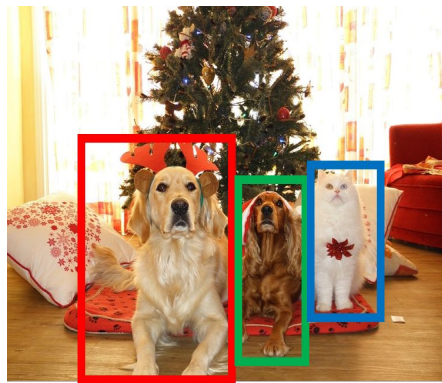
Classification + Localization



CAT

Single Object

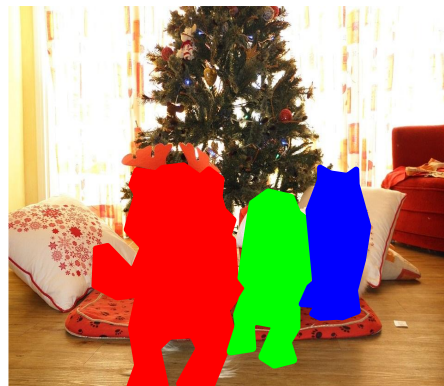
Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

[This image is CC0 public domain](#)

Other Computer Vision Tasks

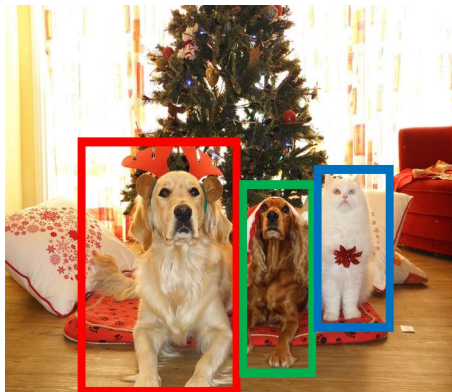
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

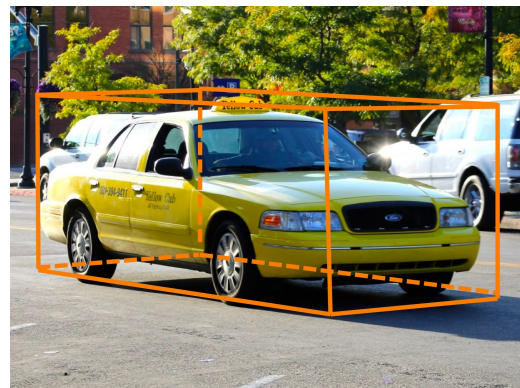
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



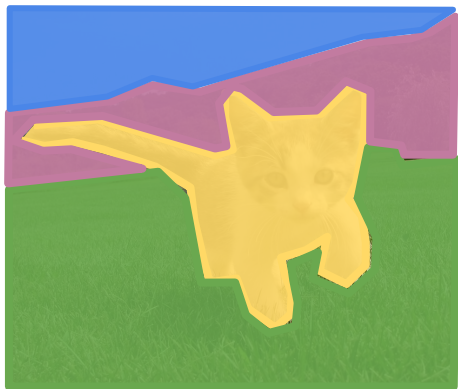
Car

Object categories +
3D bounding boxes

[This image is CC0 public domain](#)

Semantic Segmentation

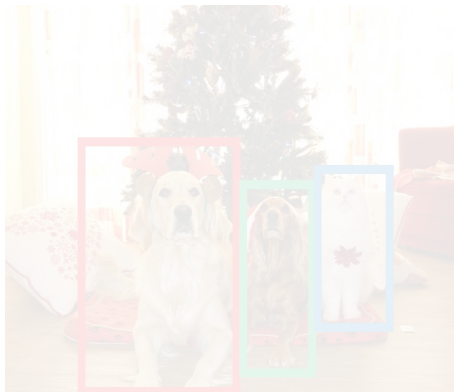
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

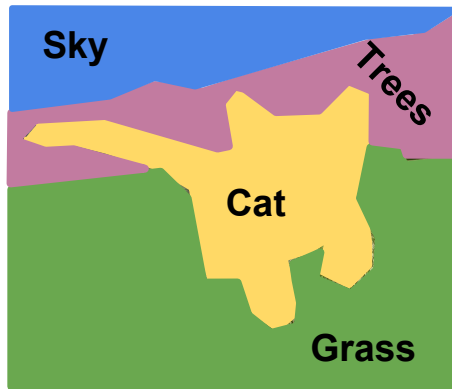
Object categories +
3D bounding boxes

[This image is CC0 public domain](#)

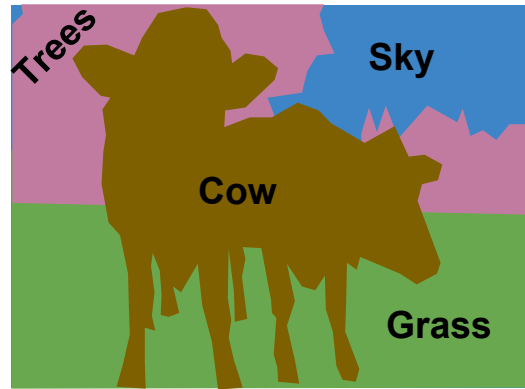
Semantic Segmentation

Label each pixel in the image with a category label

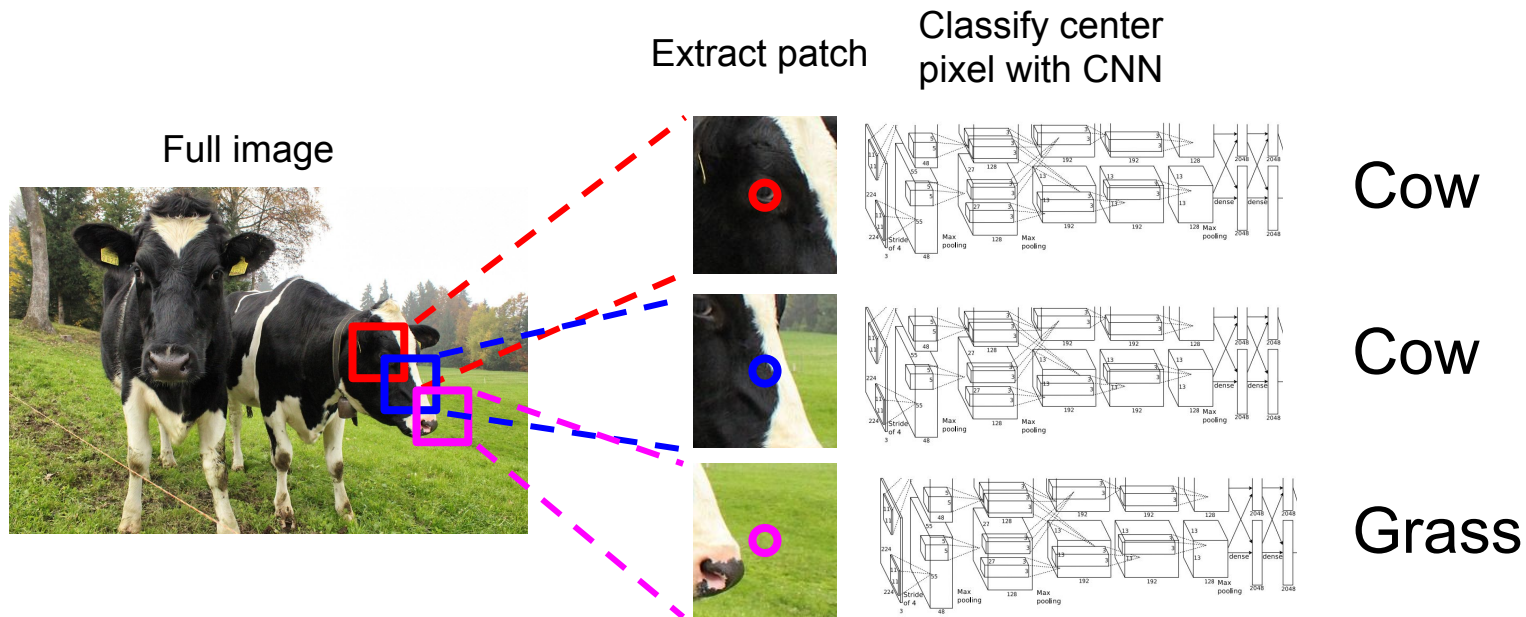
Don't differentiate instances, only care about pixels



[This image is CC0 public domain](#)



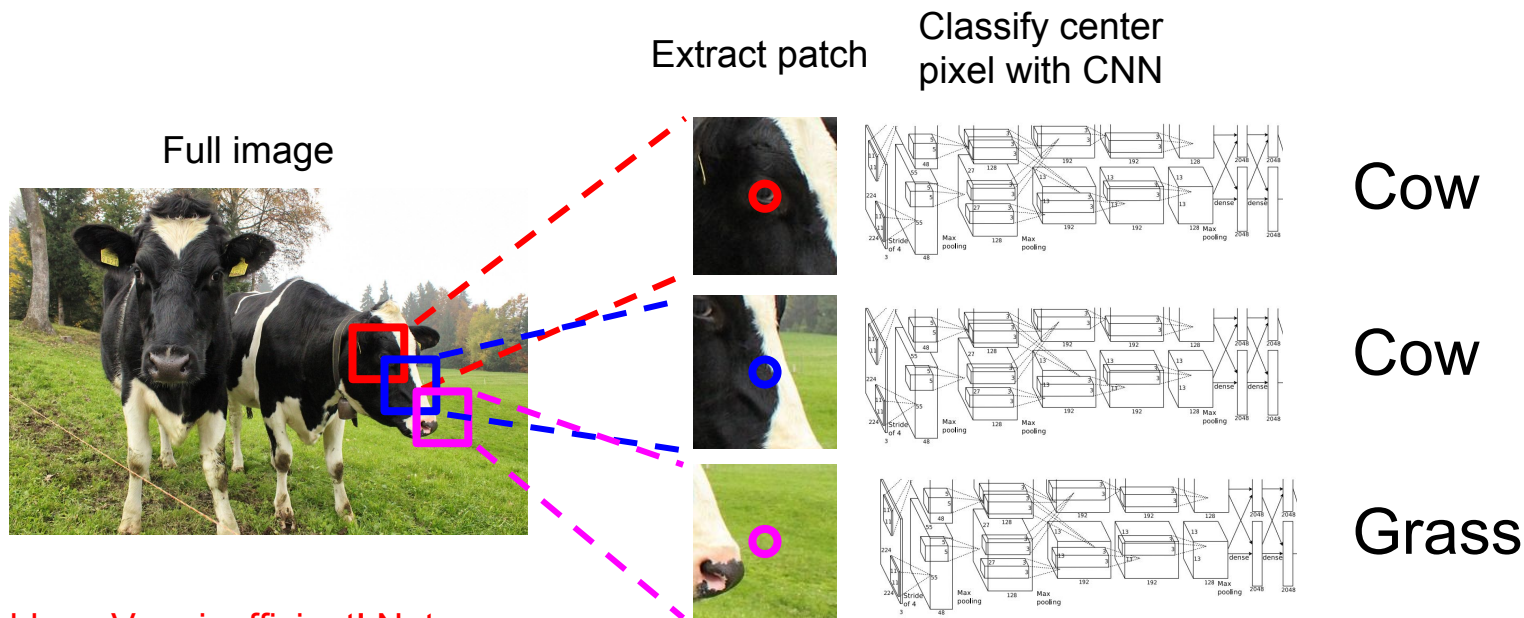
Semantic Segmentation Idea: Sliding Window



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

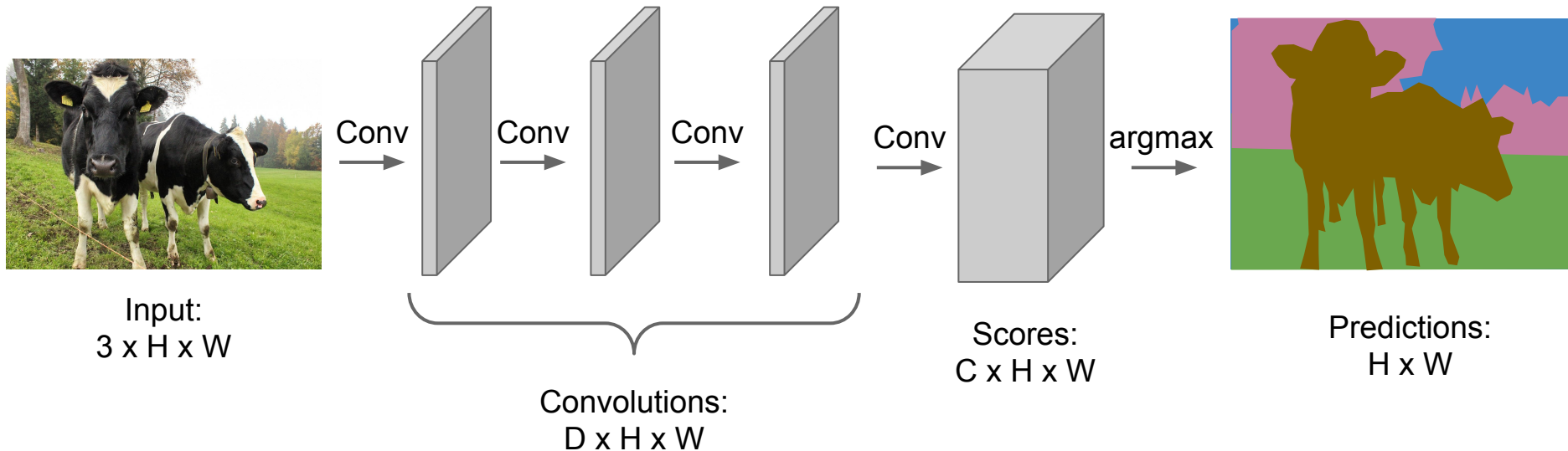


Problem: Very inefficient! Not reusing shared features between overlapping patches

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

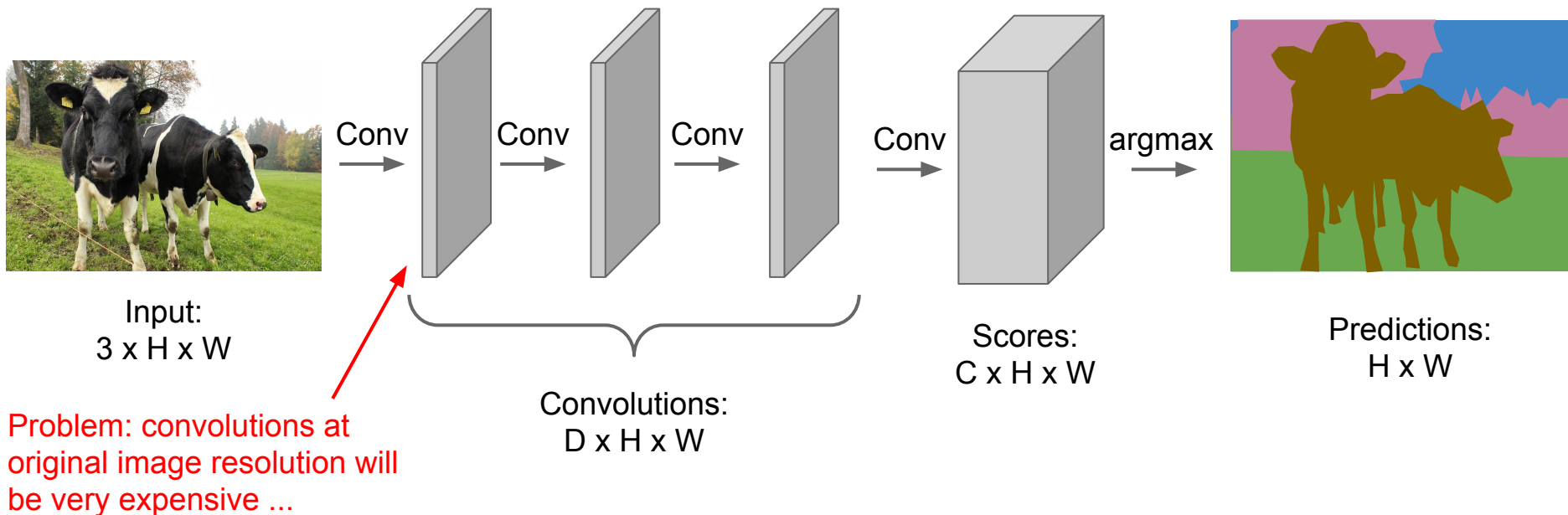
Semantic Segmentation Idea: Fully Convolutional

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



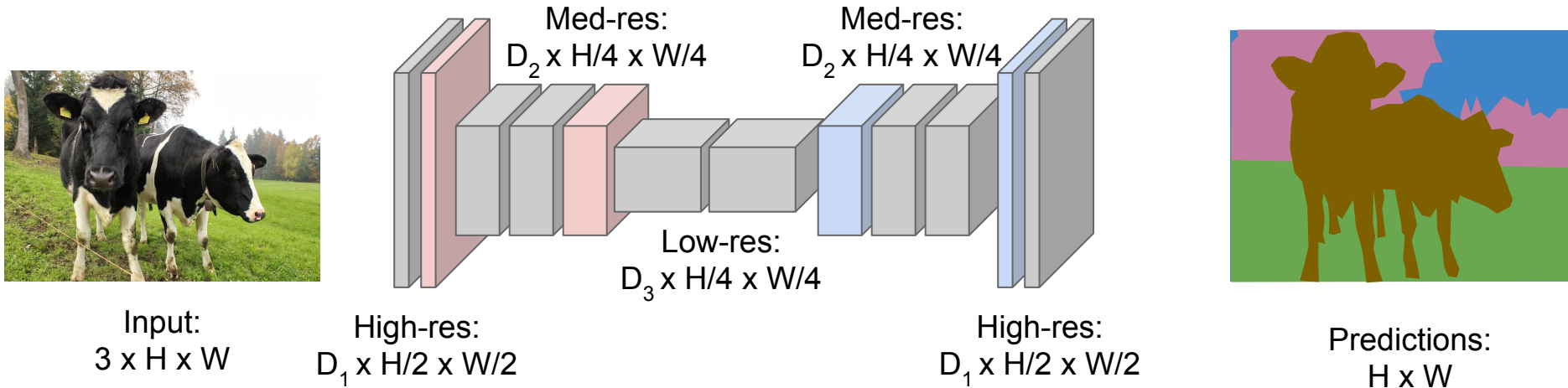
Semantic Segmentation Idea: Fully Convolutional

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

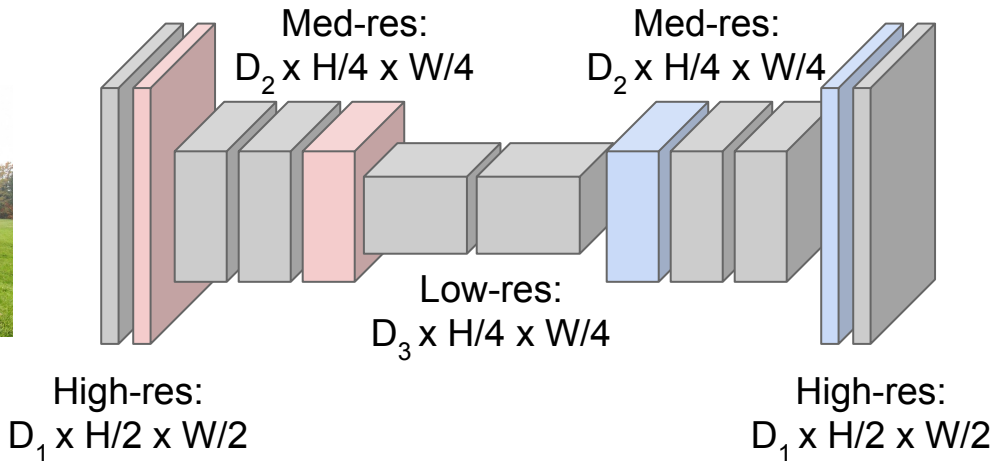
Downsampling:
Pooling, strided
convolution

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

Upsampling:
???



Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

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



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

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

1	2
3	4

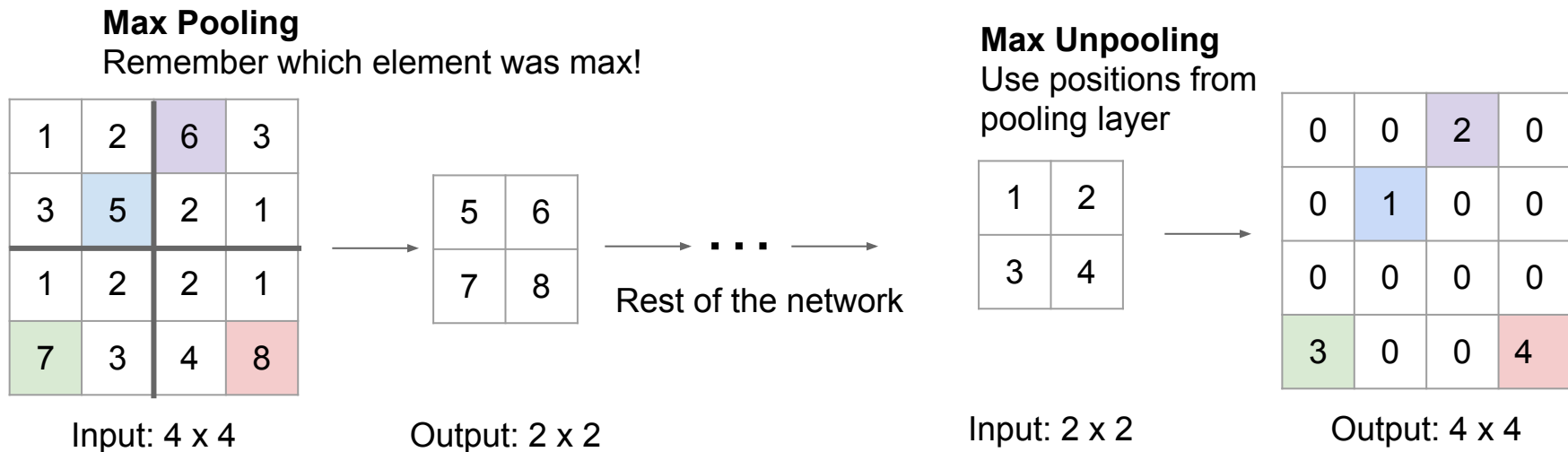


1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

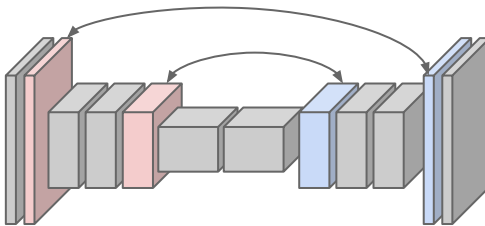
Input: 2 x 2

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

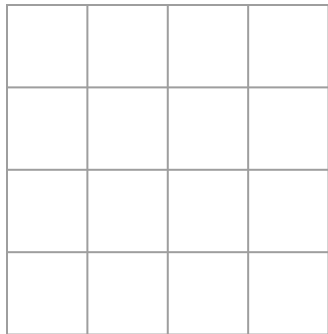


Corresponding pairs of
downsampling and
upsampling layers

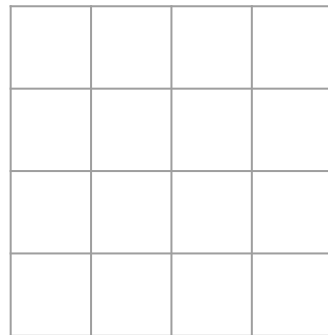


Learnable Upsampling: Transpose Convolution

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



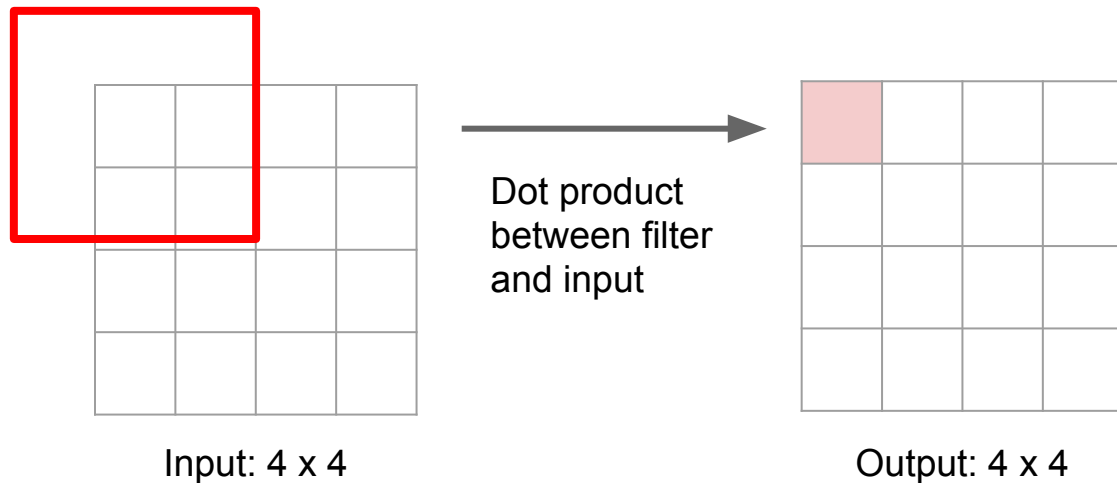
Input: 4 x 4



Output: 4 x 4

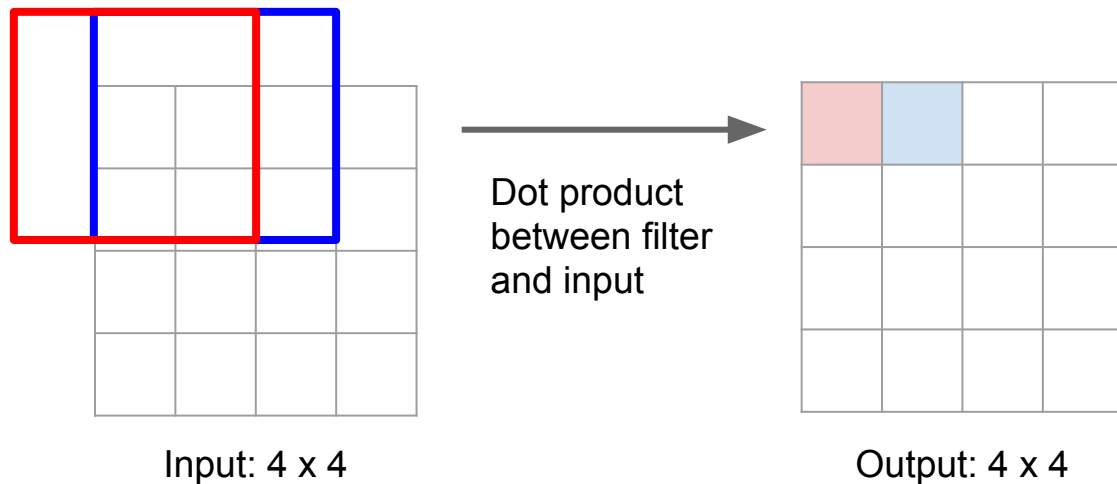
Learnable Upsampling: Transpose Convolution

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



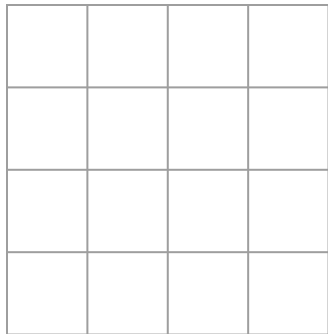
Learnable Upsampling: Transpose Convolution

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

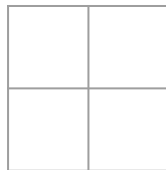


Learnable Upsampling: Transpose Convolution

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



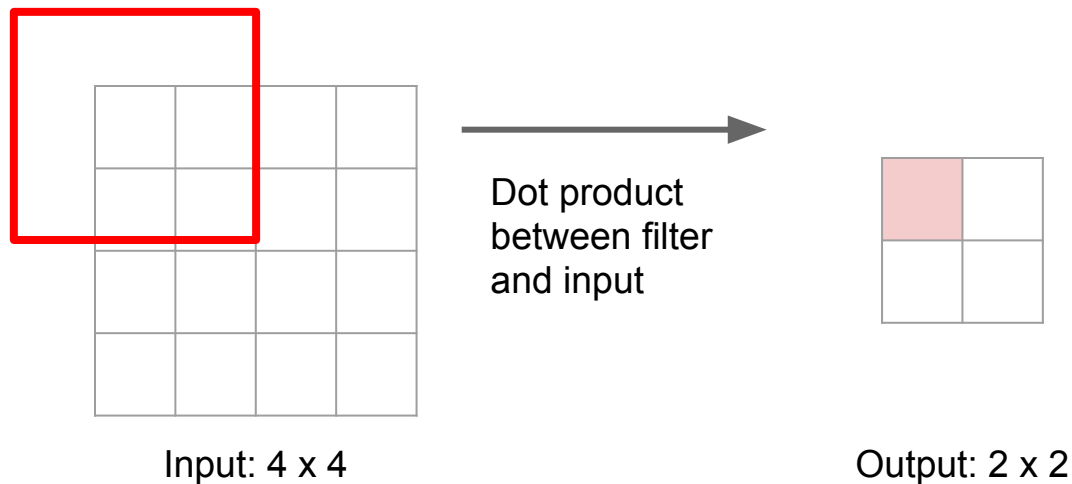
Input: 4 x 4



Output: 2 x 2

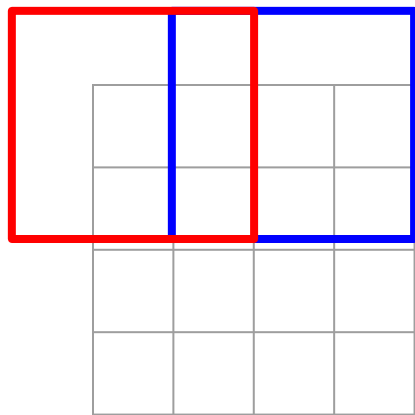
Learnable Upsampling: Transpose Convolution

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



Learnable Upsampling: Transpose Convolution

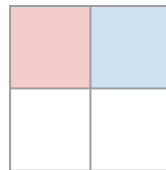
Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4



Dot product
between filter
and input



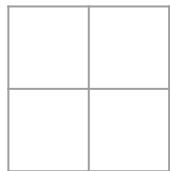
Output: 2 x 2

Filter moves 2 pixels in
the input for every one
pixel in the output

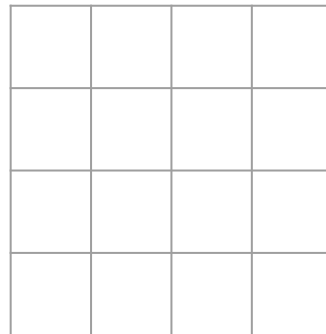
Stride gives ratio between
movement in input and
output

Learnable Upsampling: Transpose Convolution

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



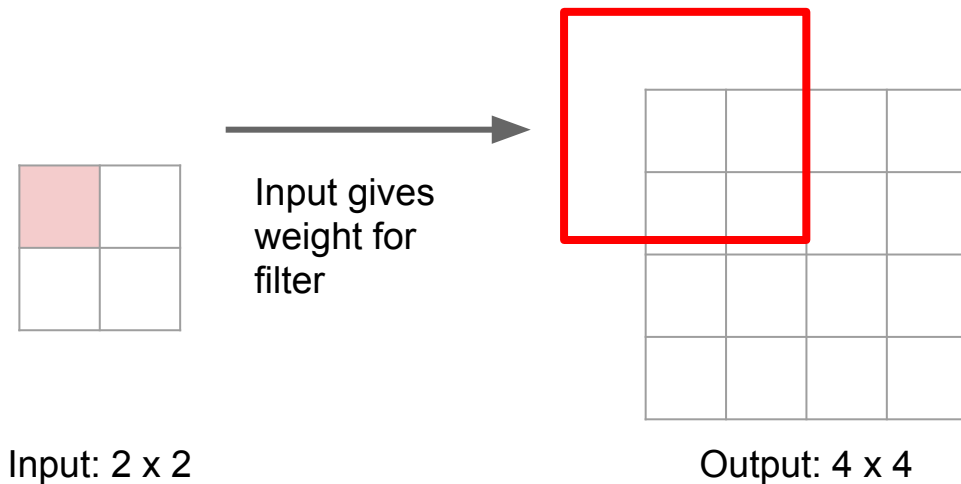
Input: 2 x 2



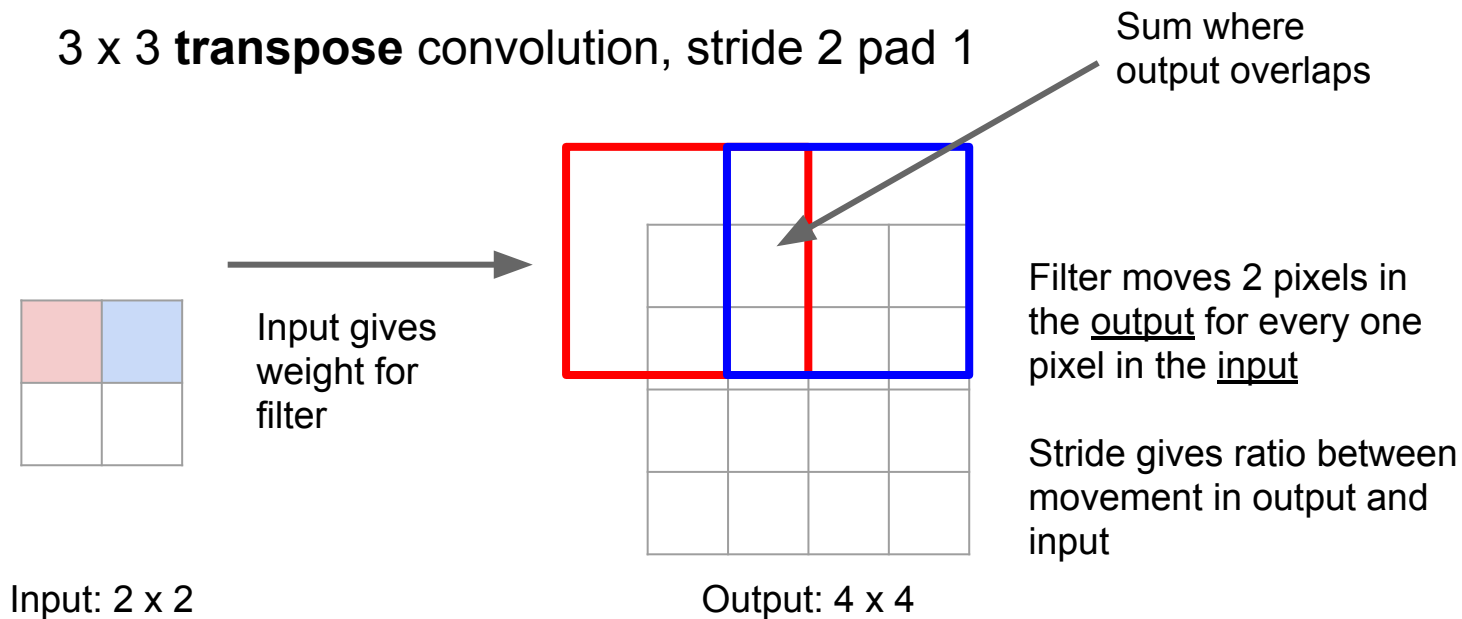
Output: 4 x 4

Learnable Upsampling: Transpose Convolution

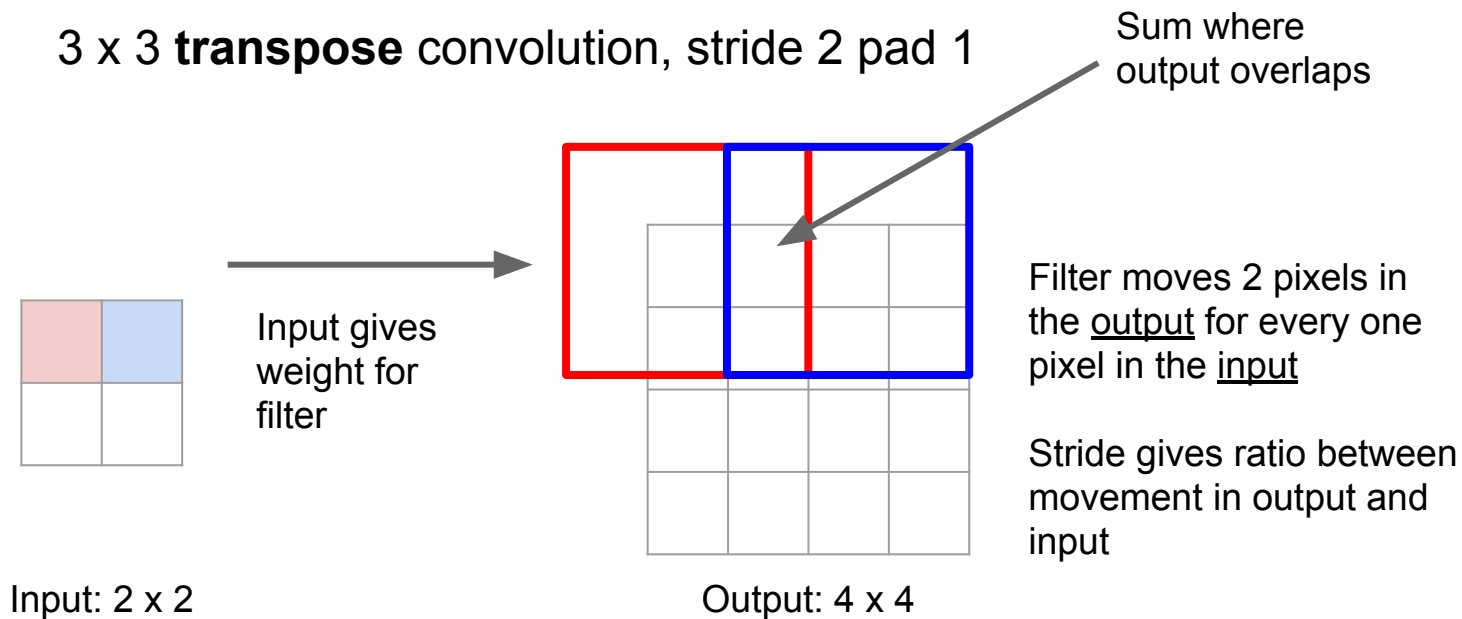
3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: Transpose Convolution



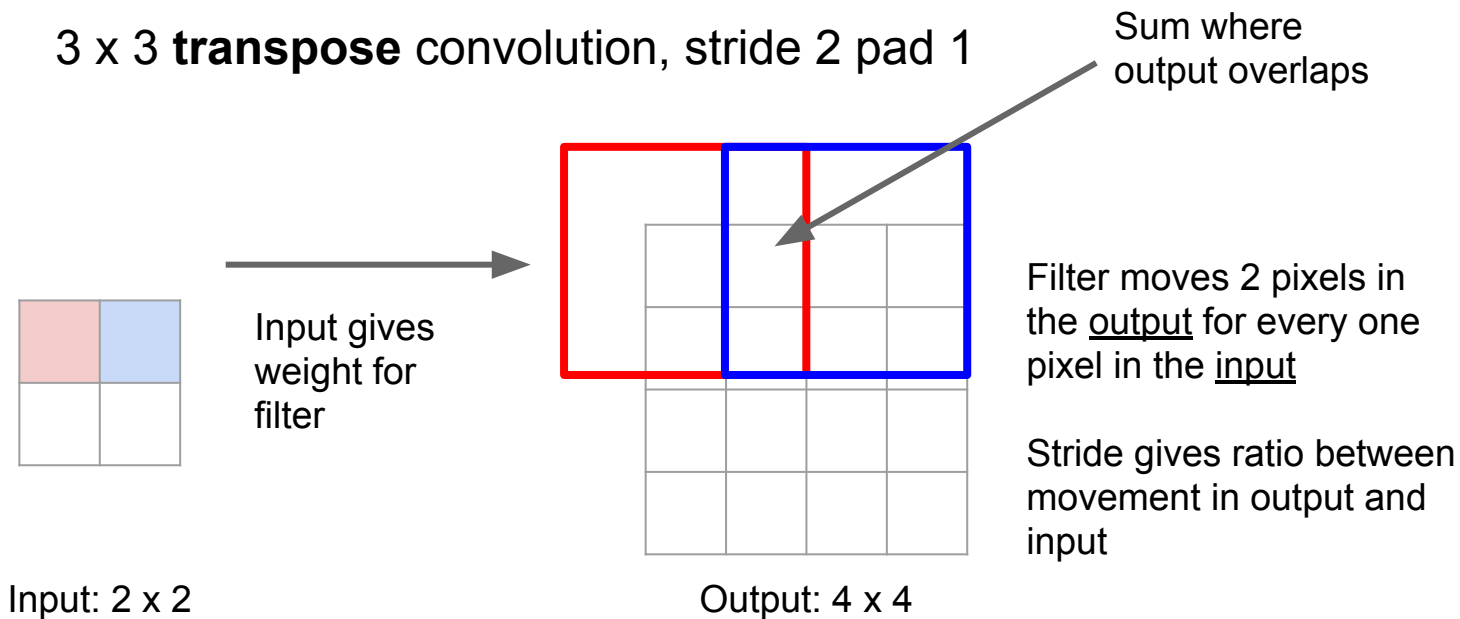
Learnable Upsampling: Transpose Convolution



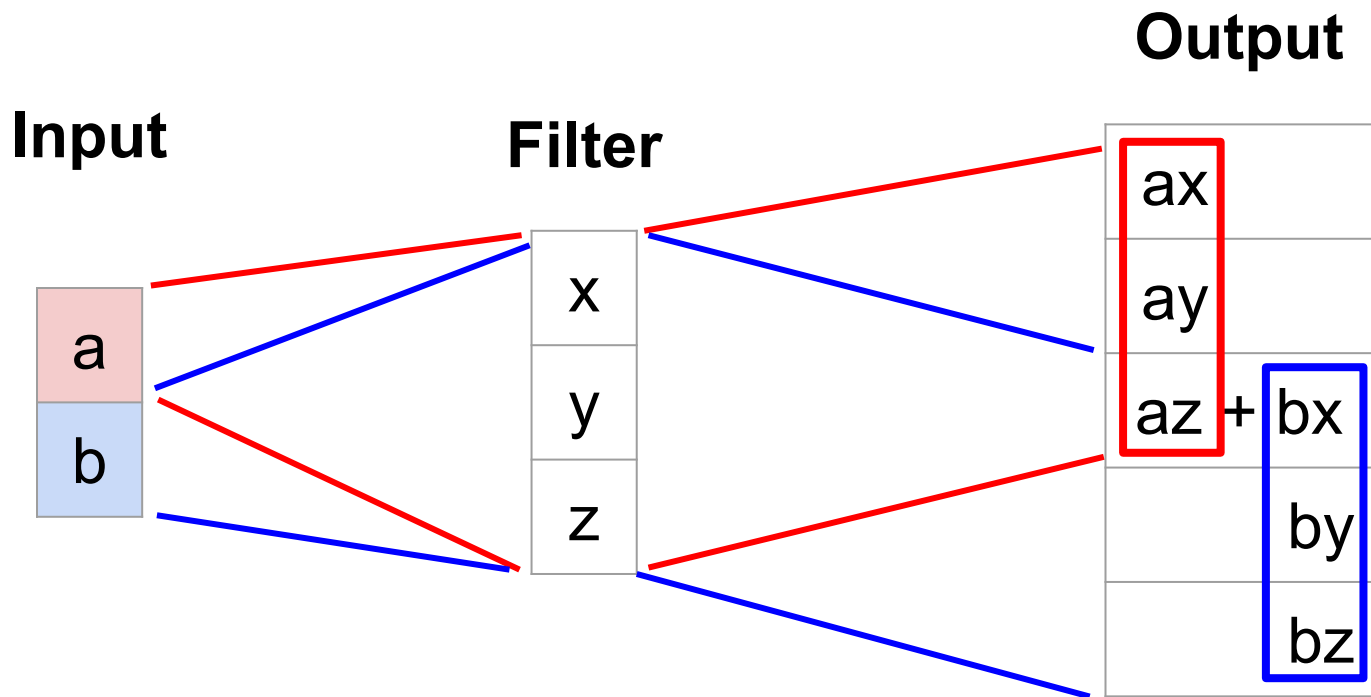
Learnable Upsampling: Transpose Convolution

Other names:

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



Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

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

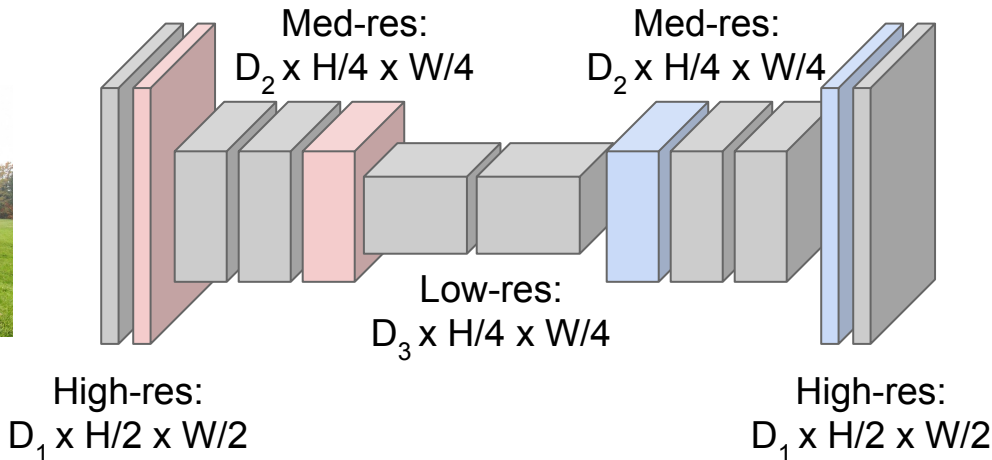
Downsampling:
Pooling, strided
convolution

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

Upsampling:
Unpooling or strided
transpose convolution



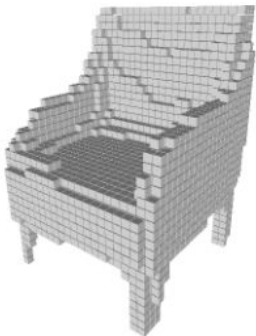
Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

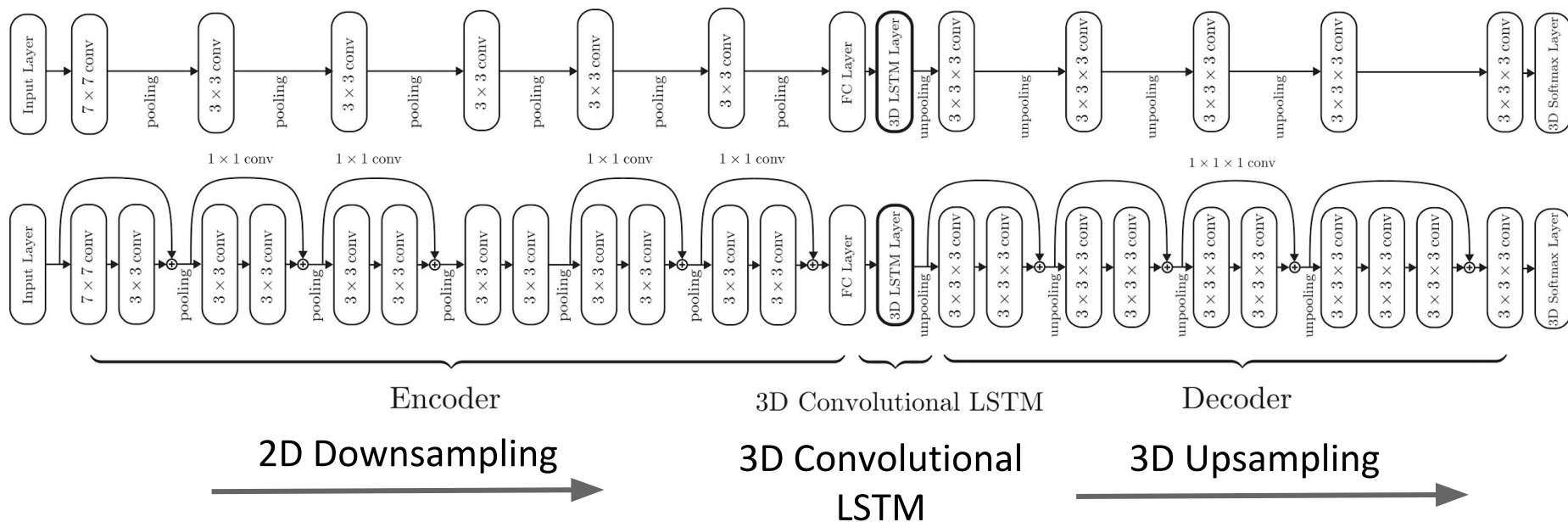
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Aside: Multi-view 3D Reconstruction

Image 1	Image 2	Image 3	Reconstruction
			

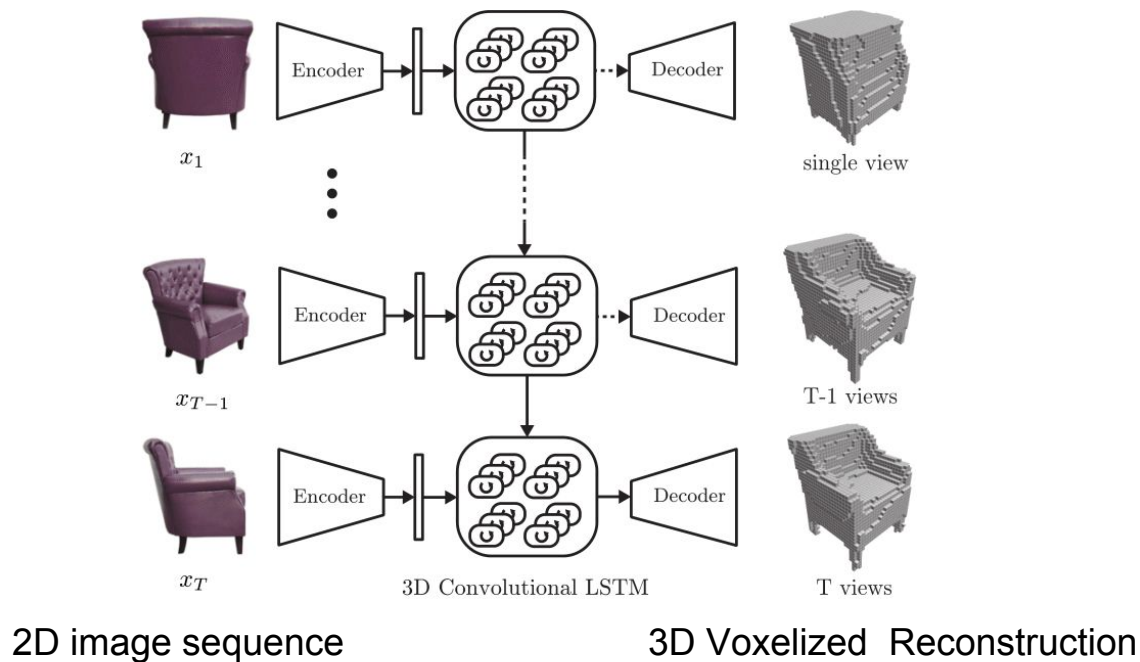
Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016, October). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In European Conference on Computer Vision (pp. 628-644). Springer, Cham.

Aside: Multi-view 3D Reconstruction



Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016, October). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In European Conference on Computer Vision (pp. 628-644). Springer, Cham.

Aside: Multi-view 3D Reconstruction



Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016, October). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In European Conference on Computer Vision (pp. 628-644). Springer, Cham.

2D Object Detection

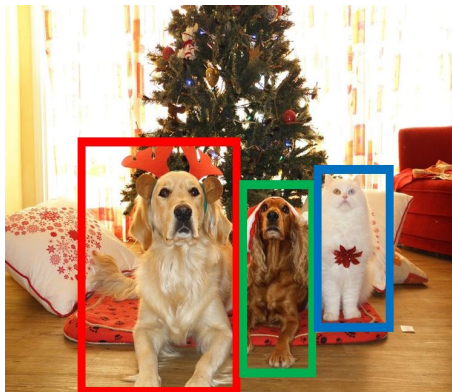
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

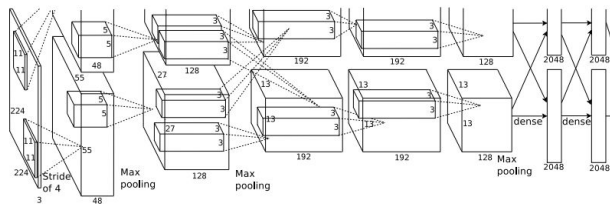
Object categories +
3D bounding boxes

[This image is CC0 public domain](#)

Classification + Localization



[This image](#) is [CC0 public domain](#)



**Fully
Connected:**
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector:
4096

**Fully
Connected:**
4096 to 4

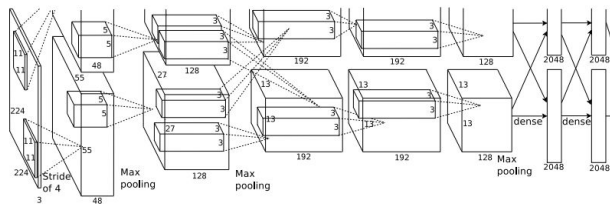
**Box
Coordinates**
(x, y, w, h)

Treat localization as a
regression problem!

Classification + Localization



[This image](#) is [CC0 public domain](#)



Fully Connected:
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax Loss

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

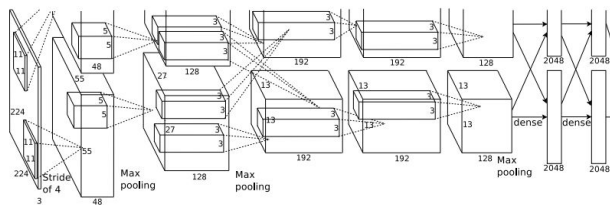
Correct box:
(x', y', w', h')

Treat localization as a regression problem!

Classification + Localization



[This image](#) is [CC0 public domain](#)



Fully
Connected:
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax
Loss

Multitask Loss

+

Loss

Vector:
4096

Fully
Connected:
4096 to 4

Box
Coordinates
(x, y, w, h)

L2 Loss

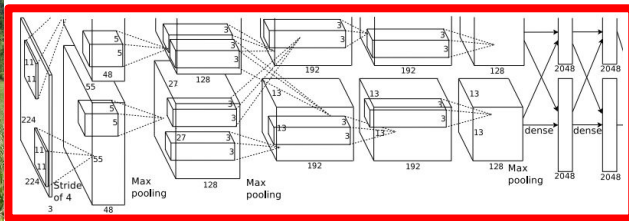
Correct box:
(x', y', w', h')

Treat localization as a
regression problem!

Classification + Localization



[This image is CC0 public domain](#)



Often pretrained on ImageNet
(Transfer learning)

Treat localization as a
regression problem!

Fully
Connected:
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax
Loss

+ → Loss

Vector:
4096

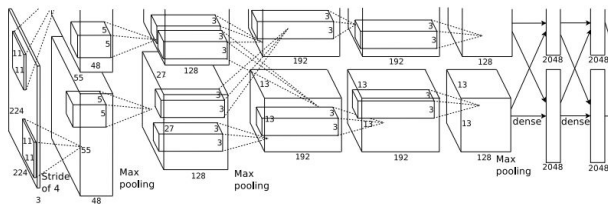
Fully
Connected:
4096 to 4

Box
Coordinates
(x, y, w, h)

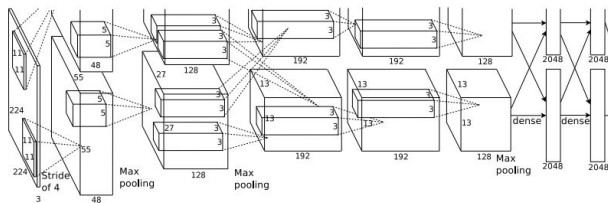
L2 Loss

Correct box:
(x', y', w', h')

Object Detection as Regression?



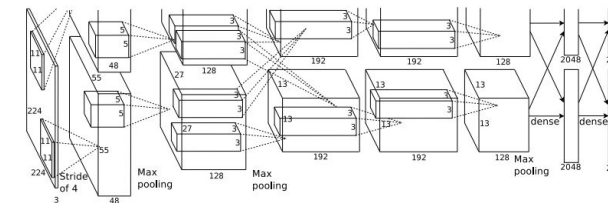
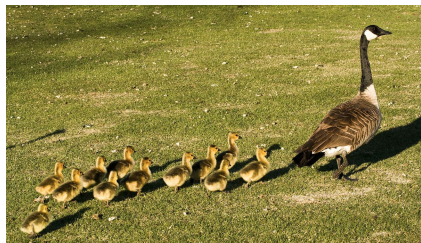
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



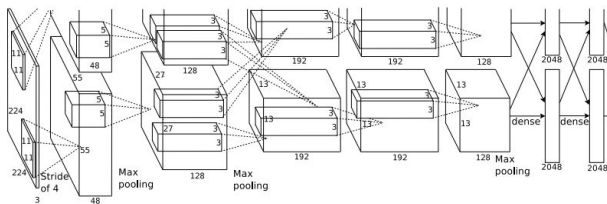
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

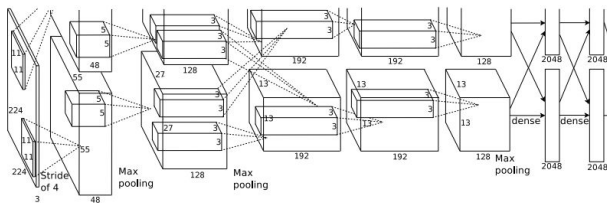
....

Object Detection as Regression?

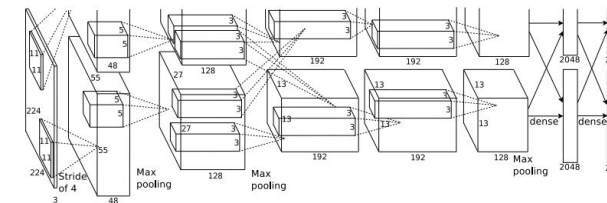
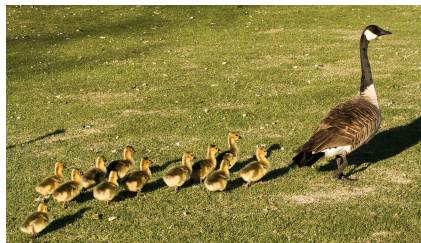
Each image needs a
different number of outputs!



CAT: (x, y, w, h) 4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h) 16 numbers
CAT: (x, y, w, h)

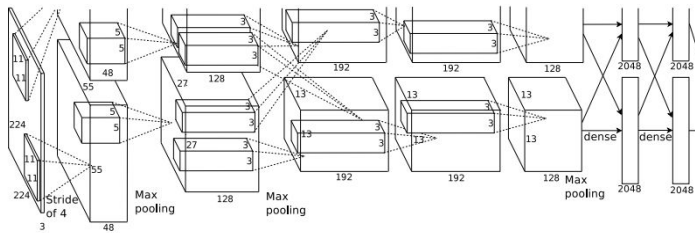


DUCK: (x, y, w, h) Many
DUCK: (x, y, w, h) numbers!

....

Object Detection as Classification: Sliding Window

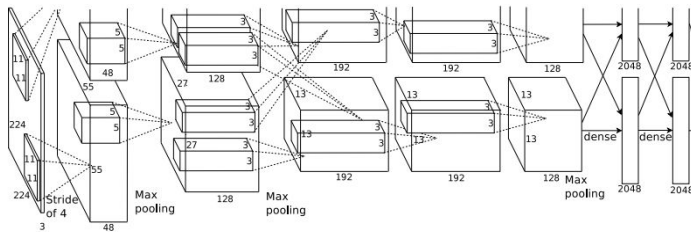
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

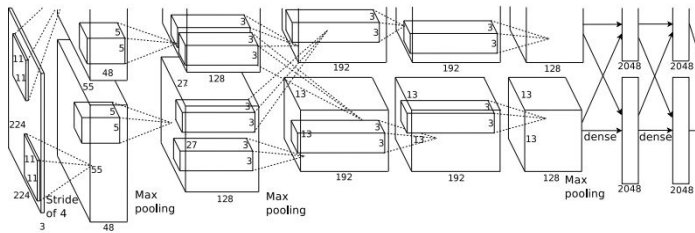
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

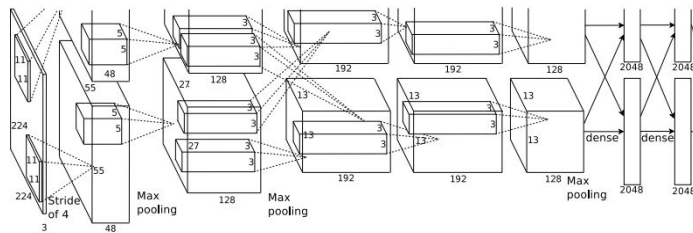
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

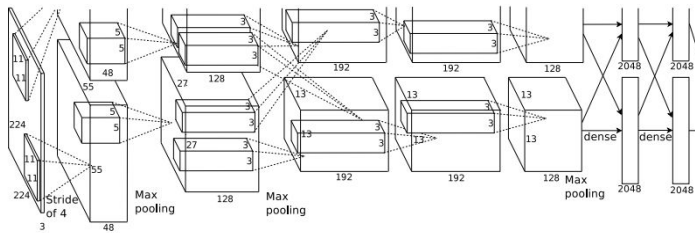
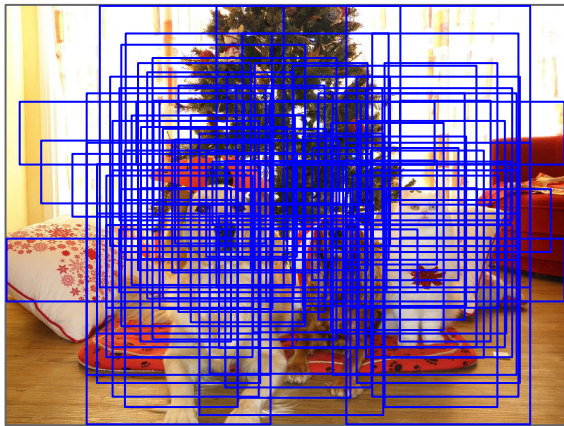
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

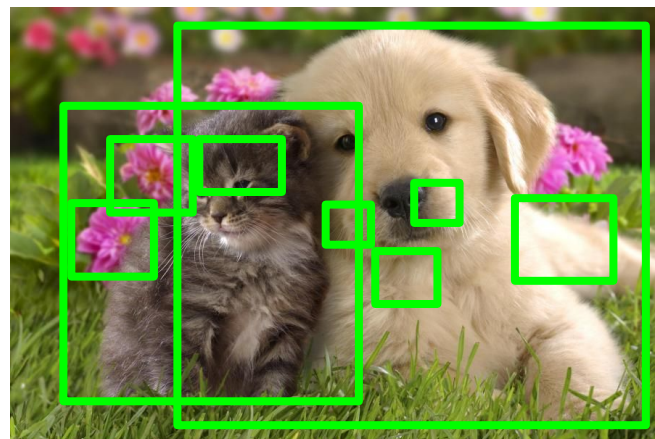


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals / Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014

Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

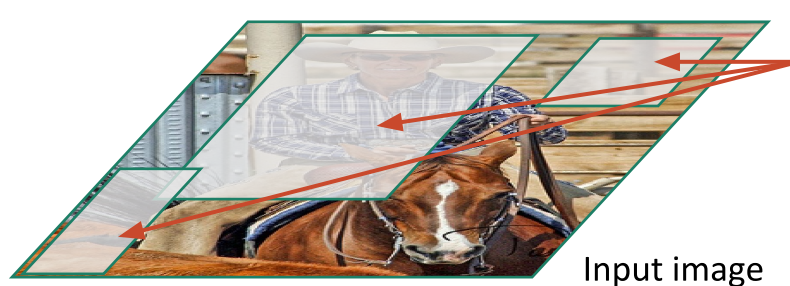
R-CNN



Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

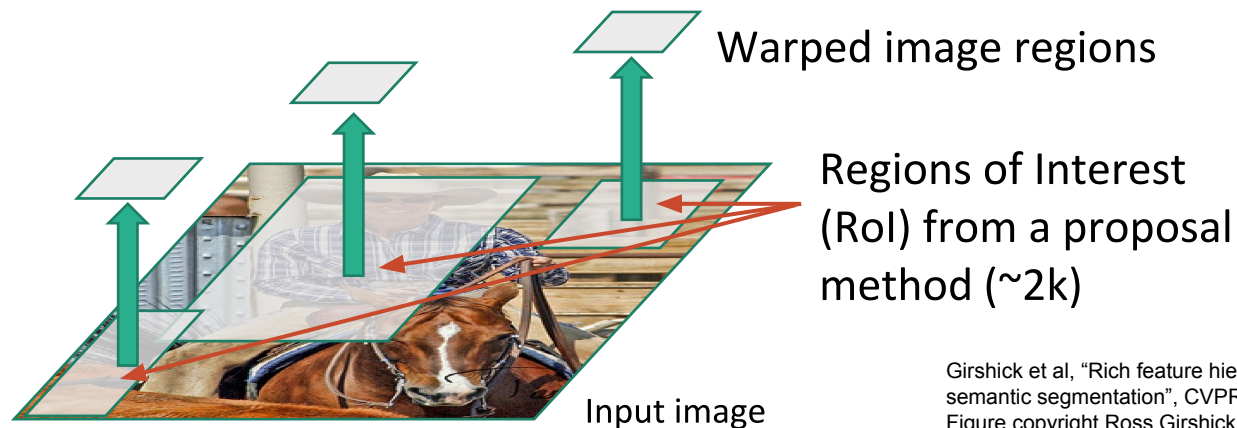
R-CNN



Regions of Interest
(RoI) from a proposal
method (~2k)

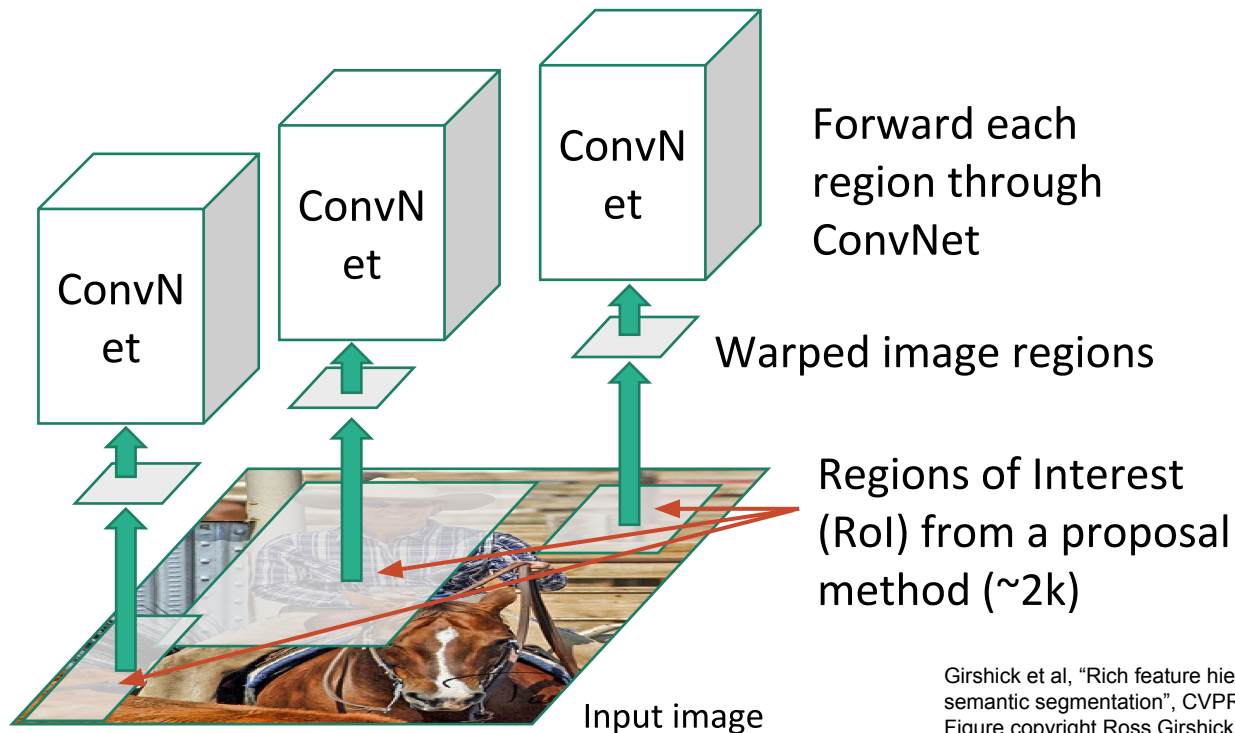
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



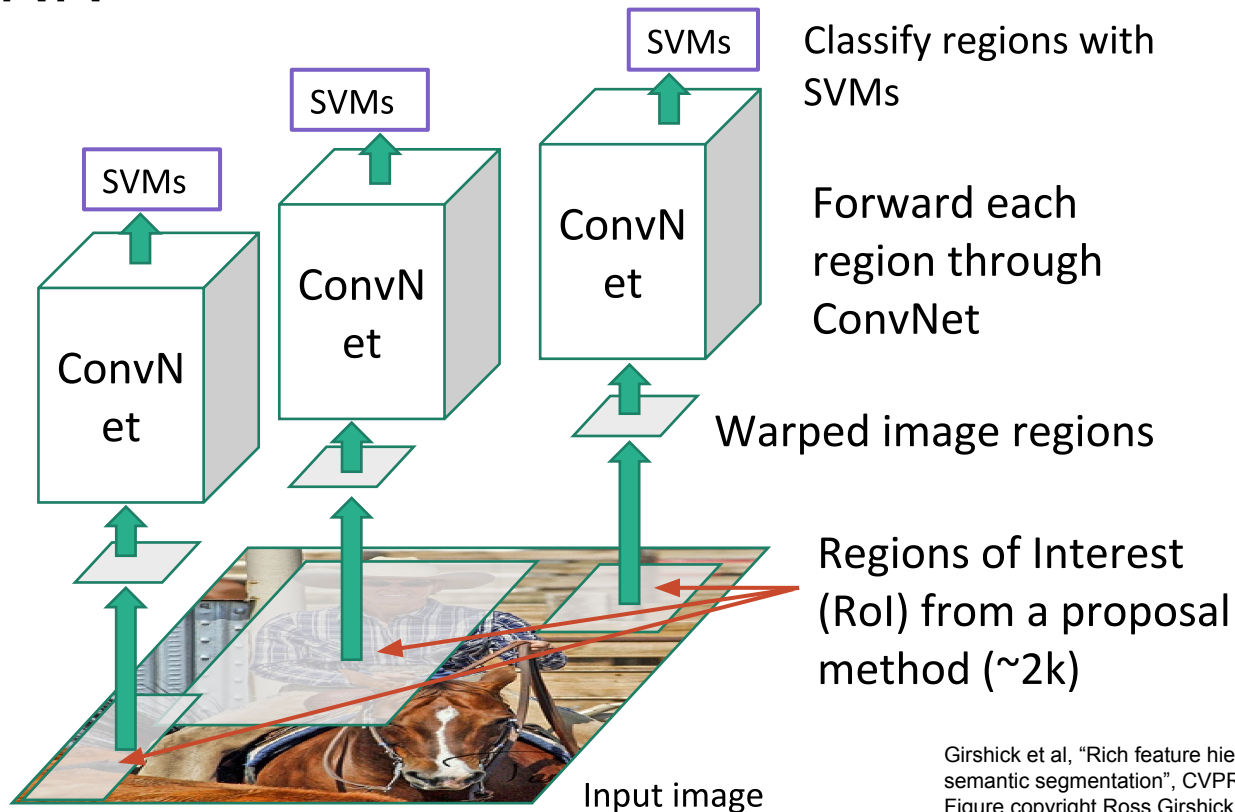
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



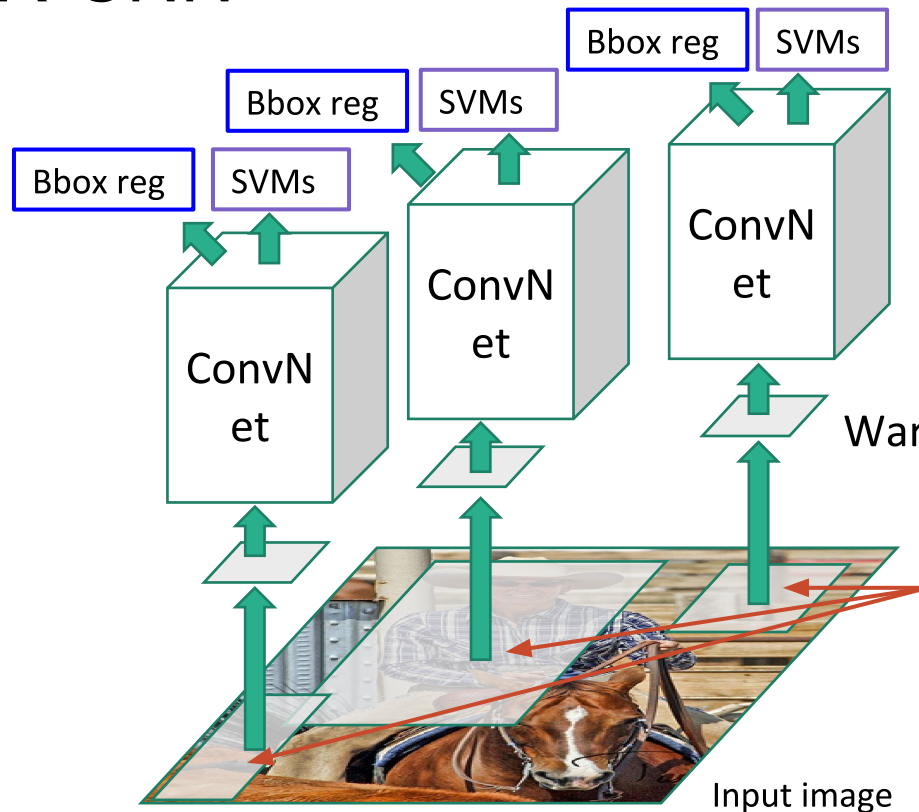
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

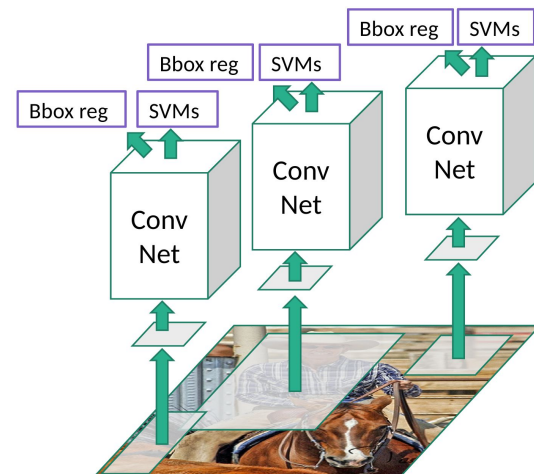
Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN: Problems

- Ad hoc training objectives
 - Fine-tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
 - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
 - Fixed by SPP-net [He et al. ECCV14]



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Slide copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN

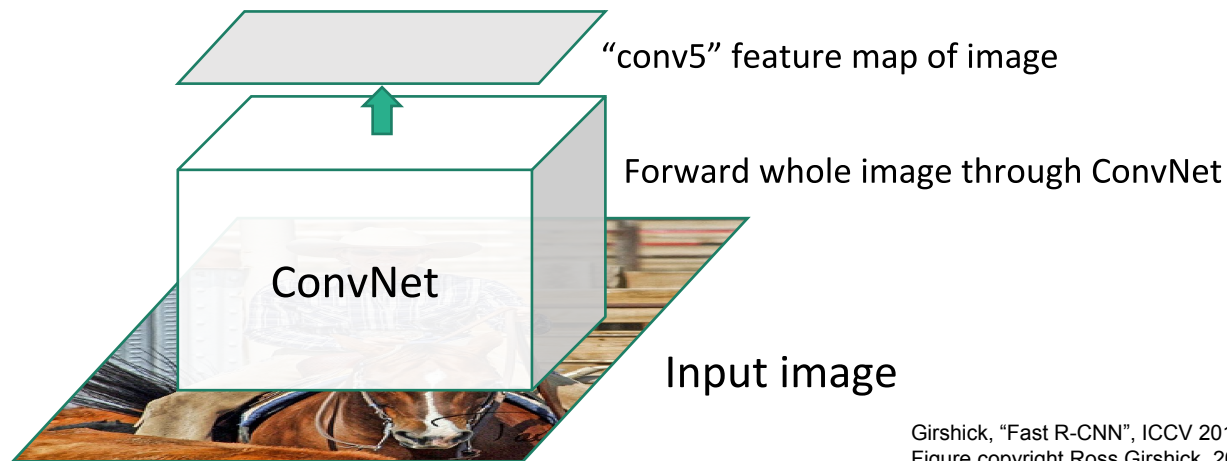


Input image

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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

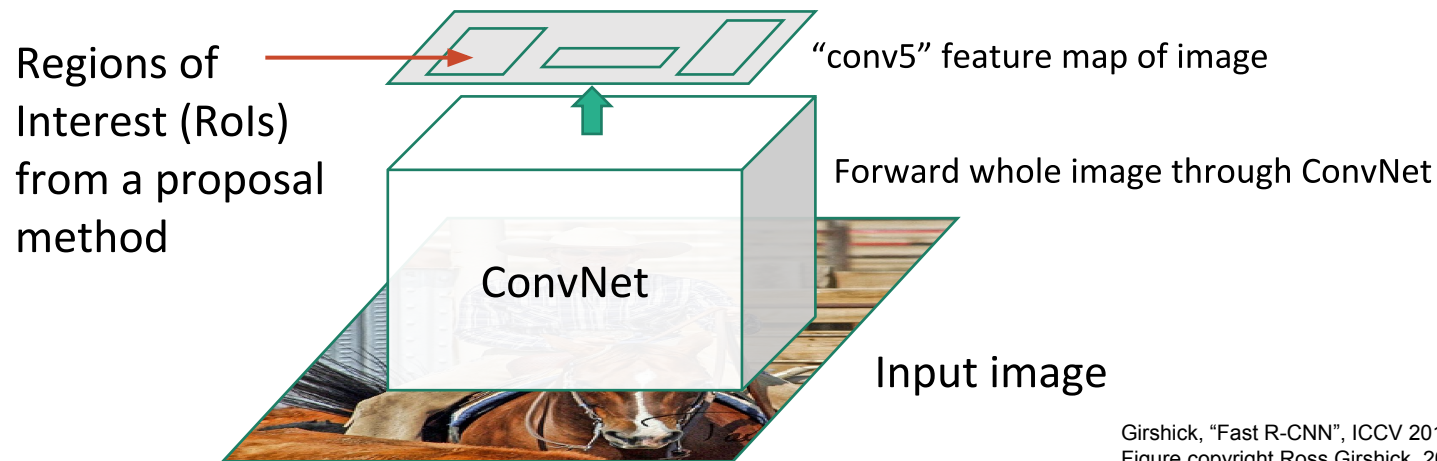
Fast R-CNN



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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

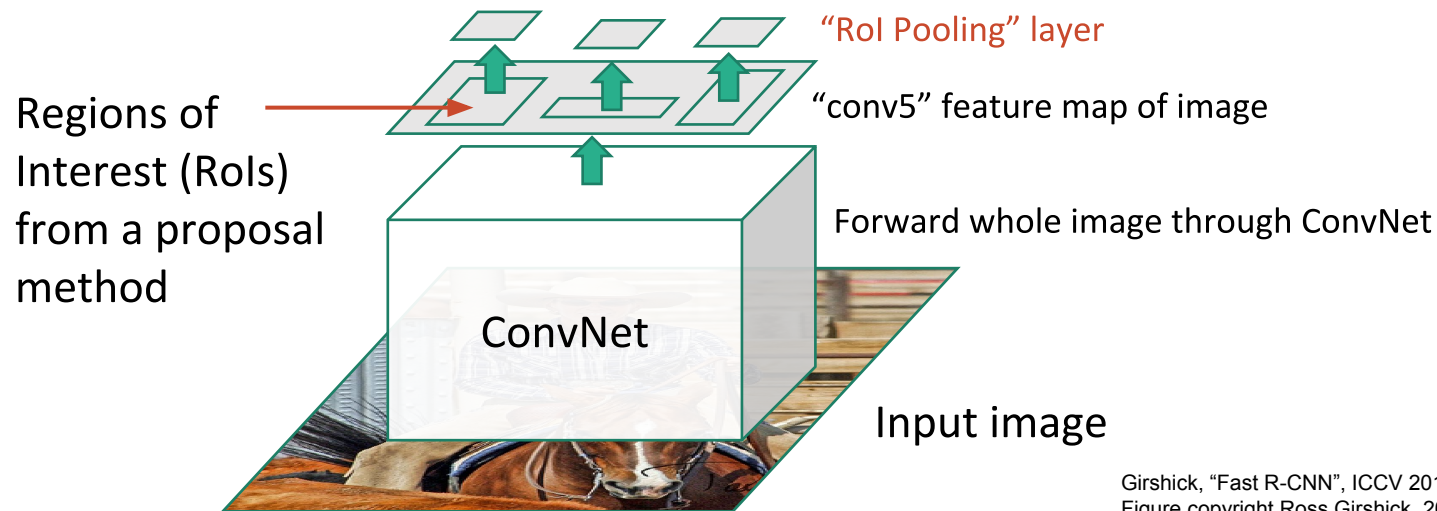
Fast R-CNN



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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

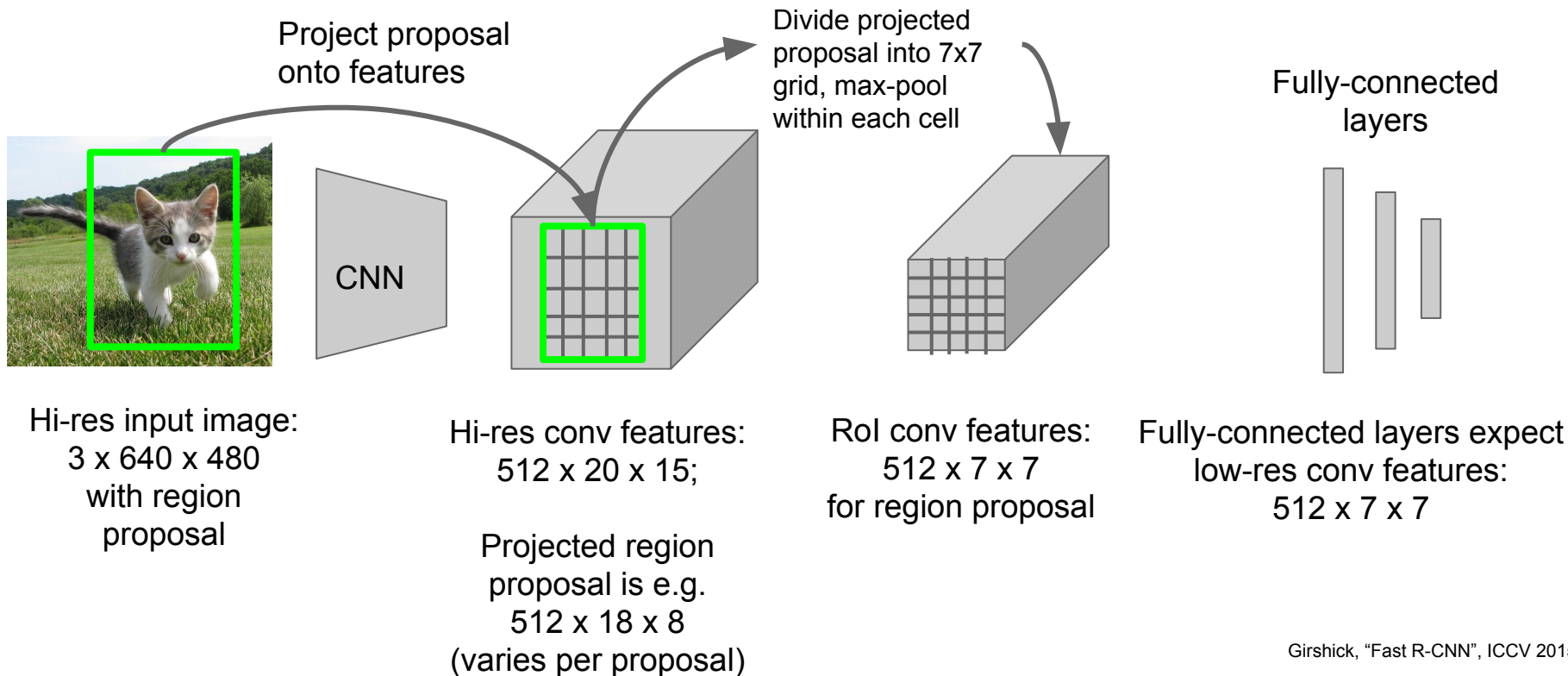
Fast R-CNN



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

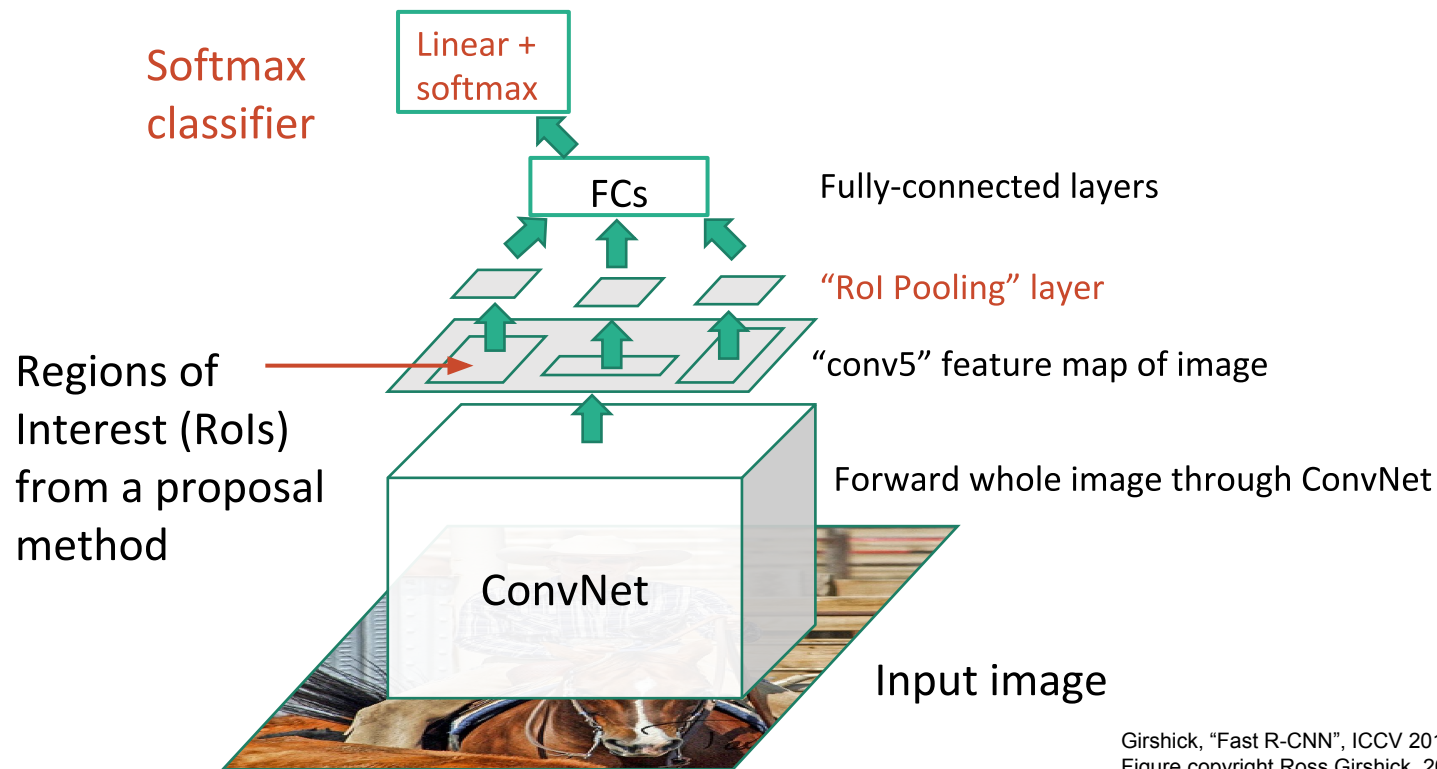
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN: RoI Pooling



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

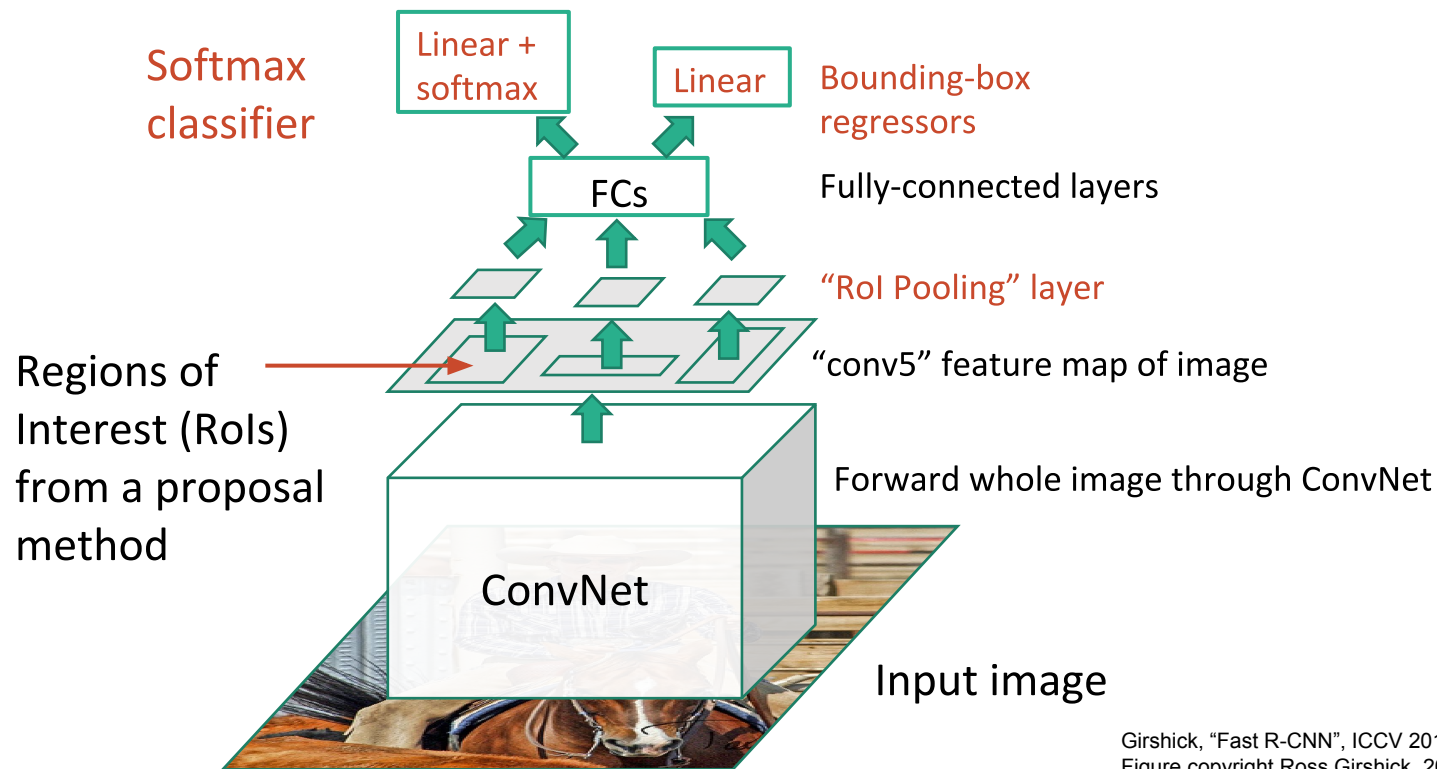
Fast R-CNN



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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

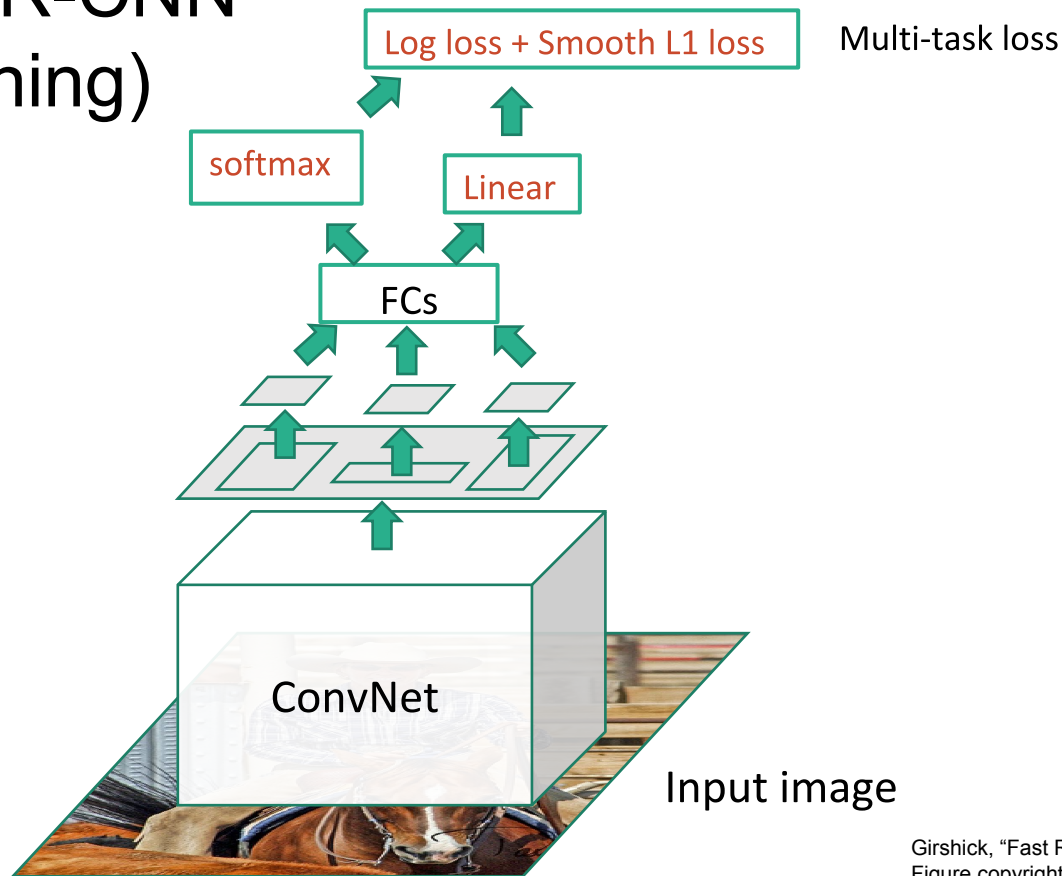
Fast R-CNN



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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

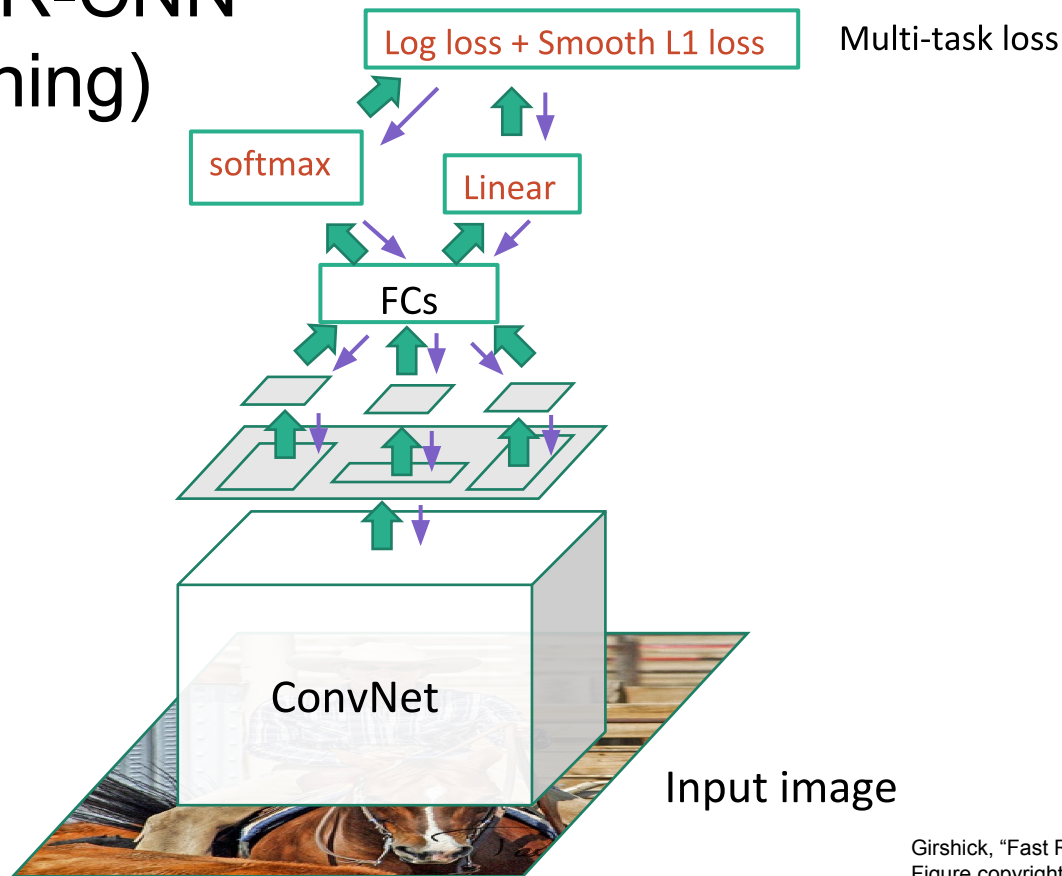
Fast R-CNN (Training)



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

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN (Training)

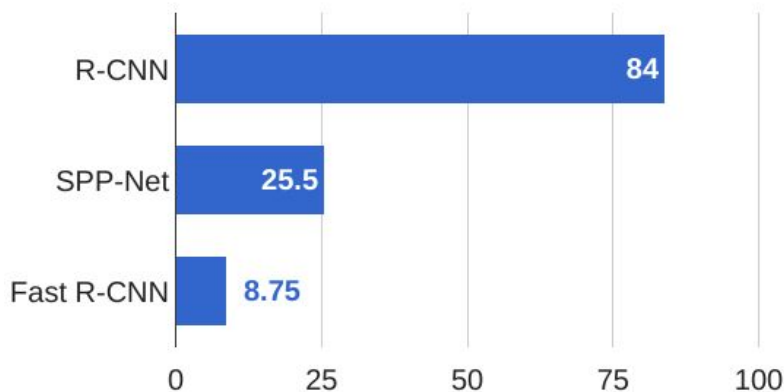


Girshick, "Fast R-CNN", ICCV 2015.

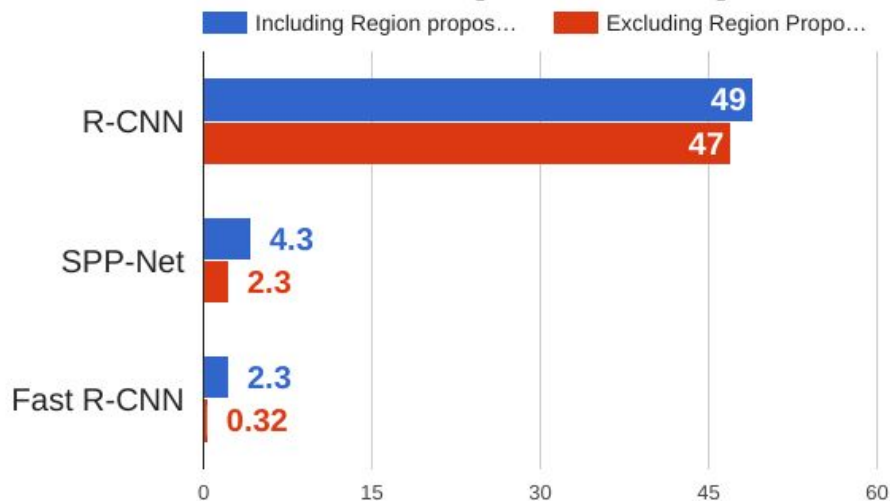
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN vs SPP vs Fast R-CNN

Training time (Hours)



Test time (seconds)



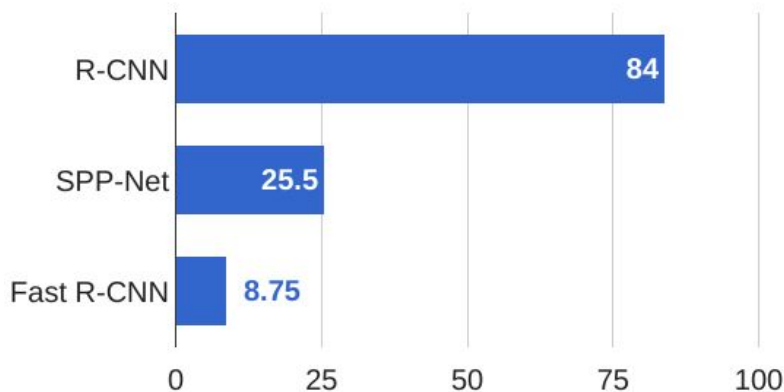
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014

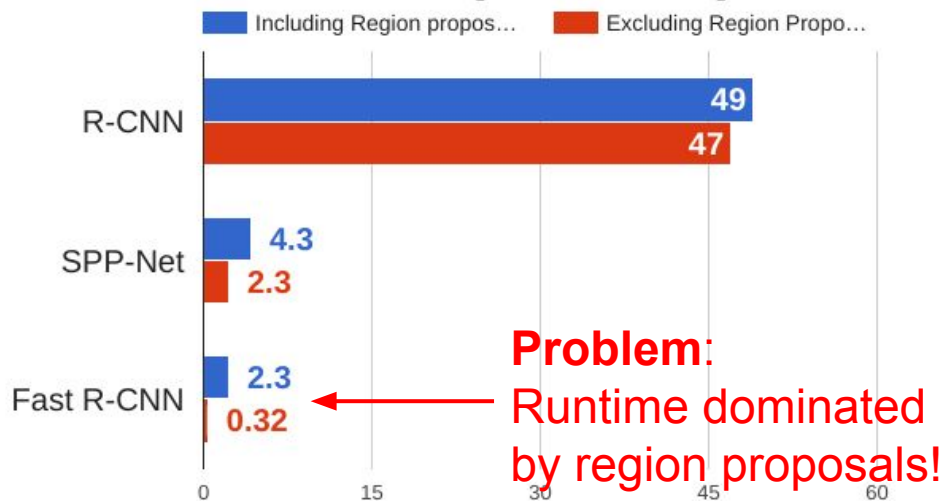
Girshick, "Fast R-CNN", ICCV 2015

R-CNN vs SPP vs Fast R-CNN

Training time (Hours)



Test time (seconds)



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

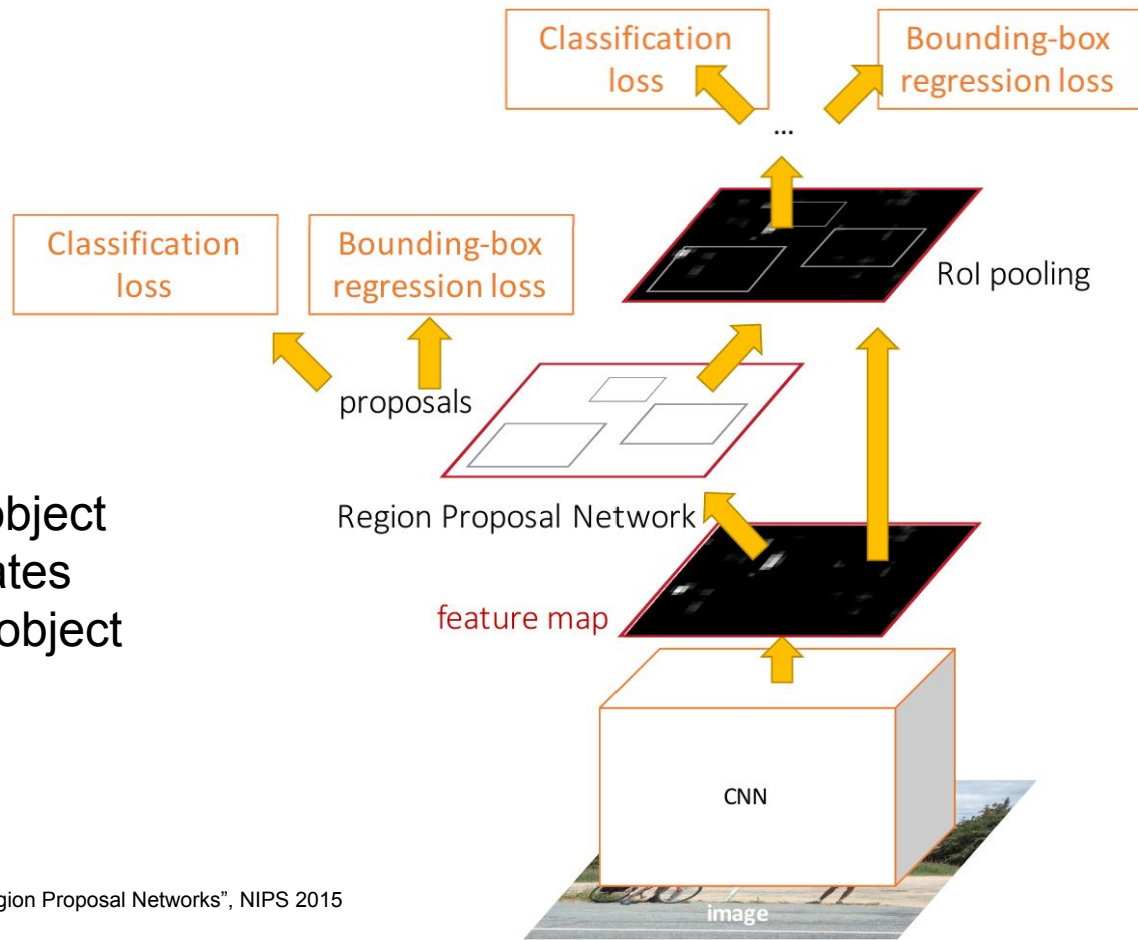
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

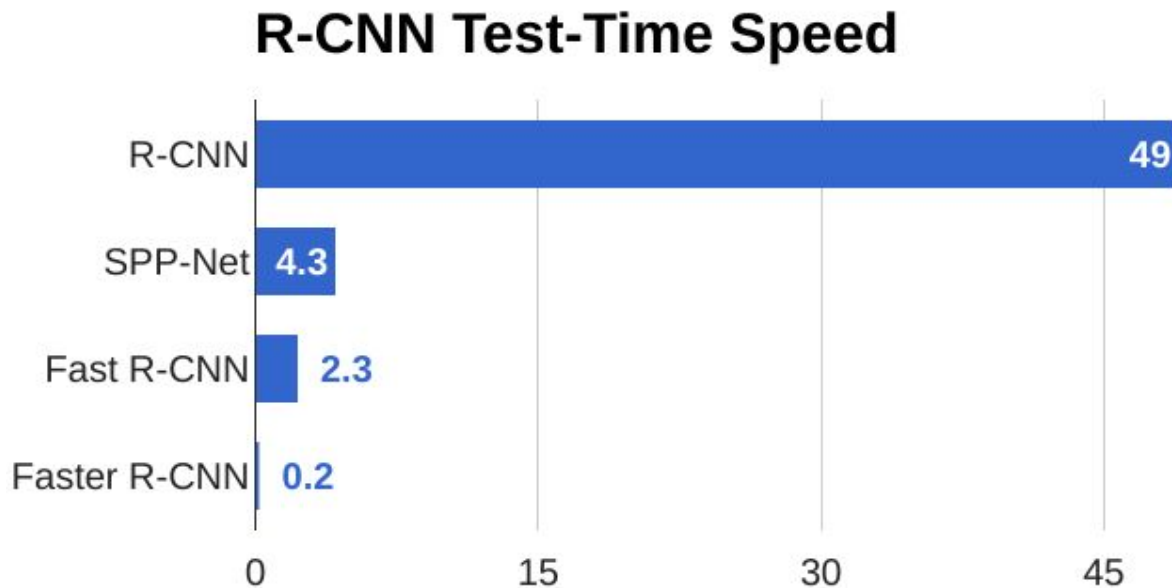
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



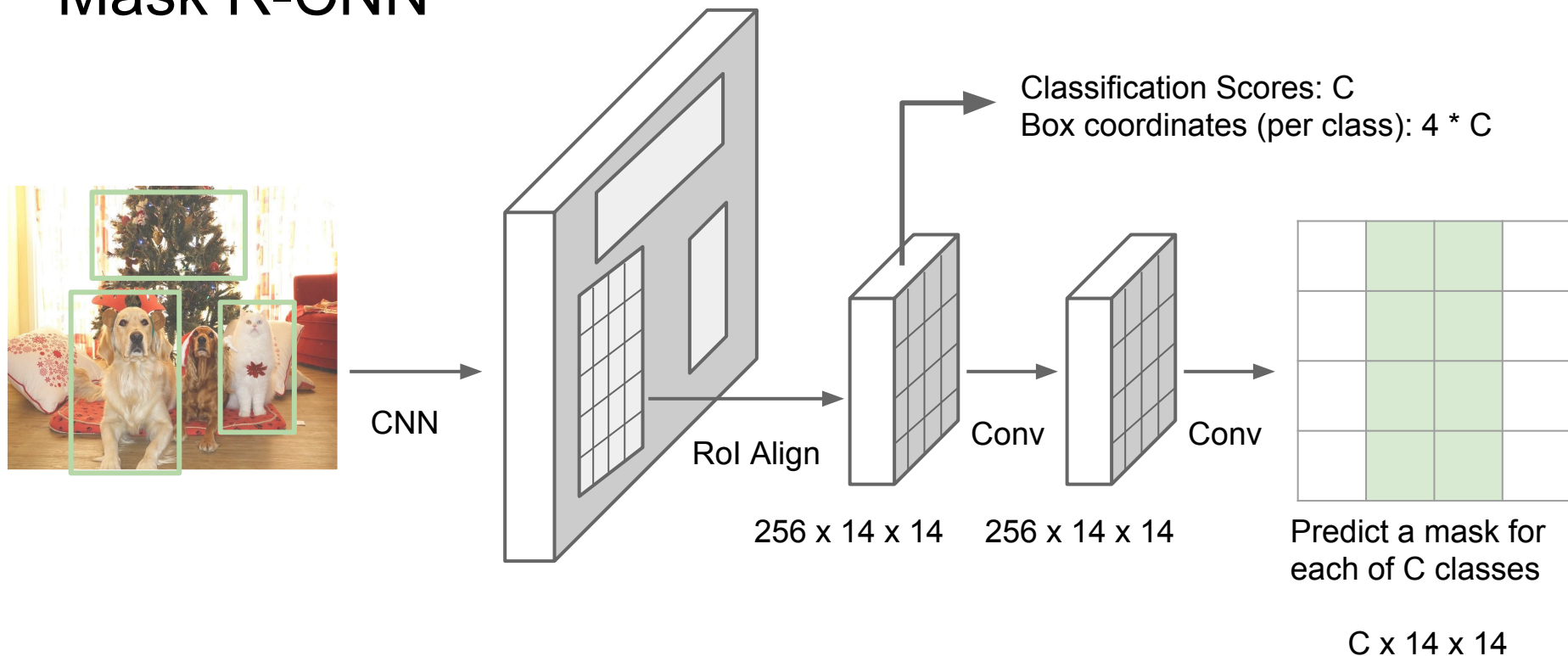
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN:

Make CNN do proposals!

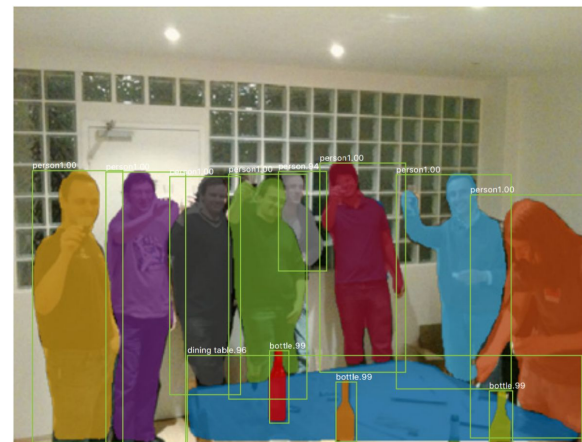
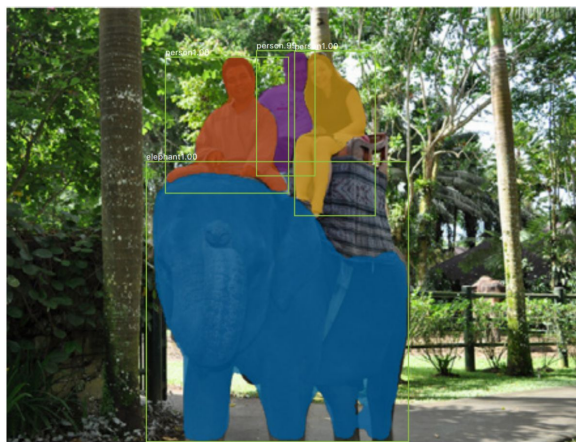


Mask R-CNN



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

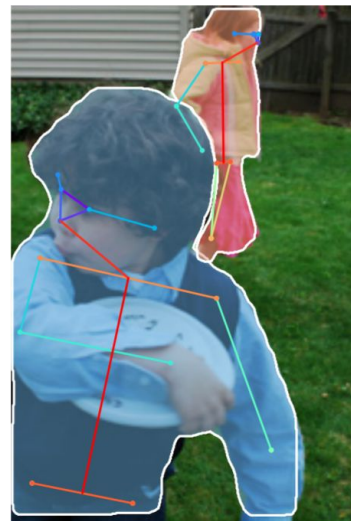
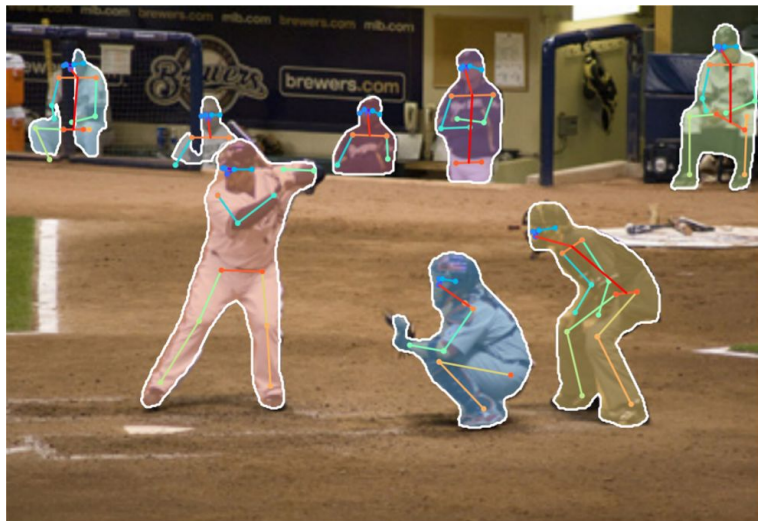
Mask R-CNN: Very Good Results!



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

Mask R-CNN

Also does pose

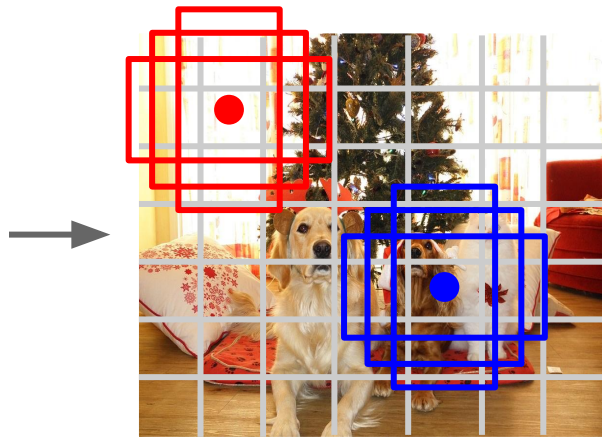


He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

Detection without Proposals: YOLO / SSD



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
($dx, dy, dh, dw, confidence$)
- Predict scores for each of C classes (including background as a class)

Output:
 $7 \times 7 \times (5 * B + C)$

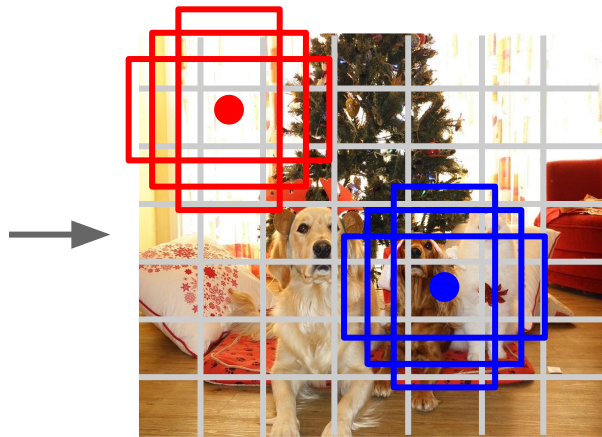
Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
($dx, dy, dh, dw, confidence$)
- Predict scores for each of C classes (including background as a class)

Output:
 $7 \times 7 \times (5 * B + C)$

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

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

Object Detection: Impact of Deep Learning

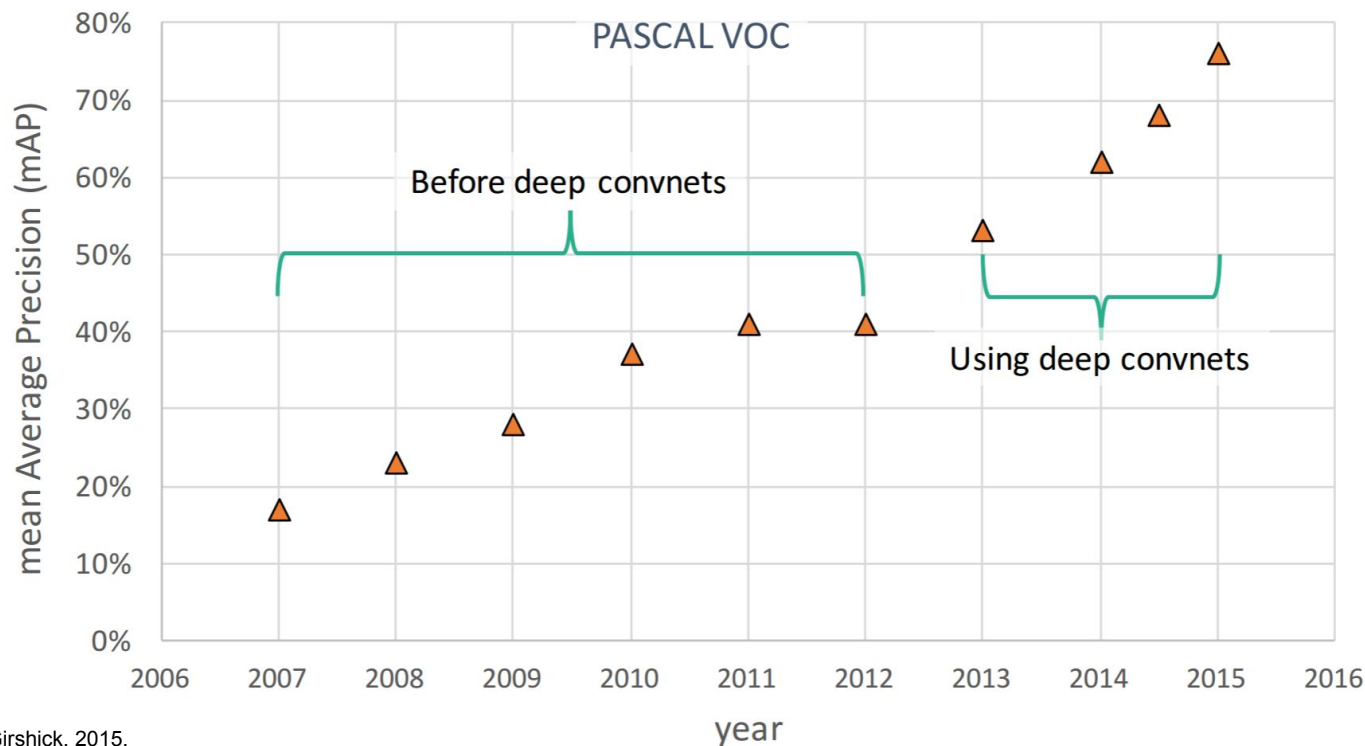


Figure copyright Ross Girshick, 2015.
Reproduced with permission.

Open Source Frameworks

Lots of good implementations on GitHub!

TensorFlow Detection API:

https://github.com/tensorflow/models/tree/master/research/object_detection

Faster RCNN, SSD, RFCN, Mask R-CNN

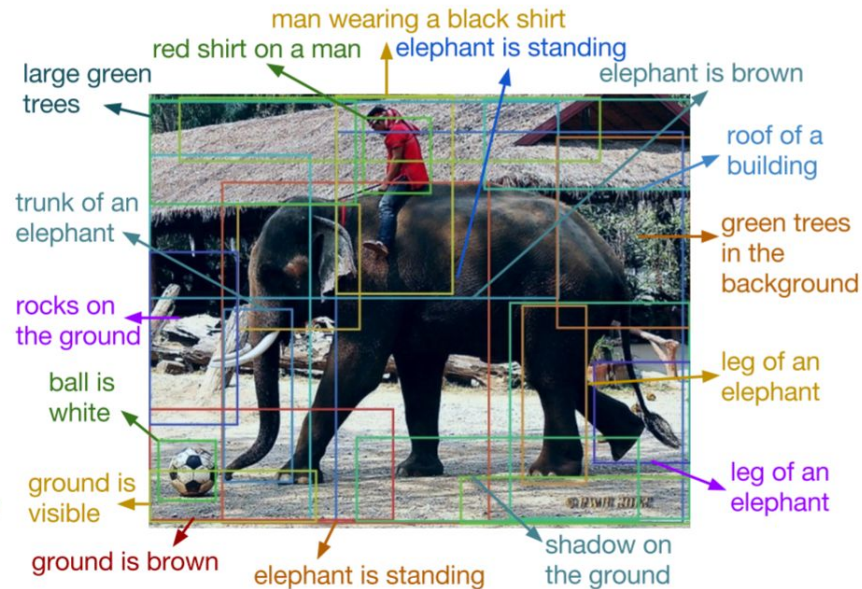
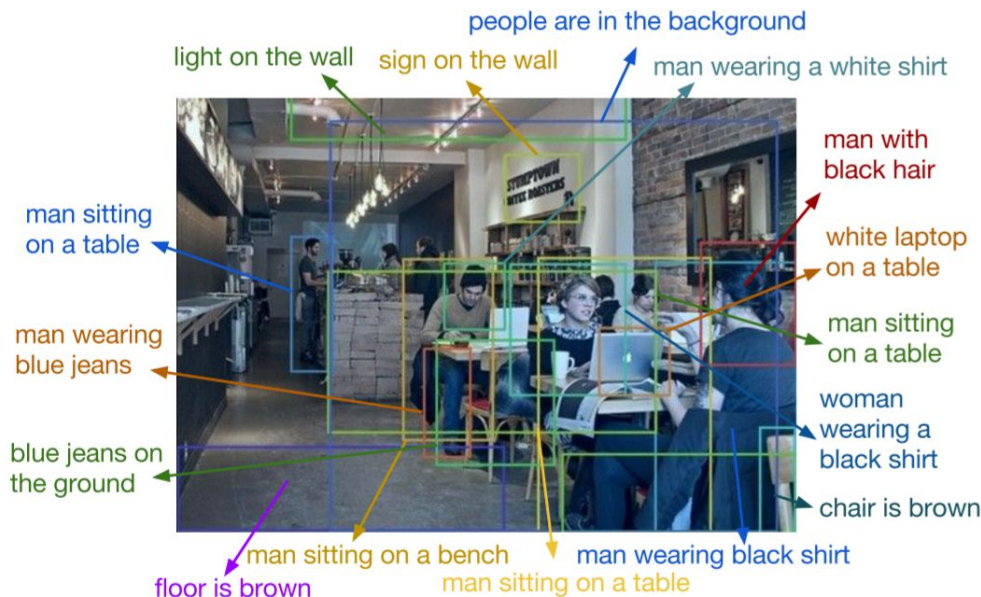
Caffe2 Detectron:

<https://github.com/facebookresearch/Detectron>

Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN

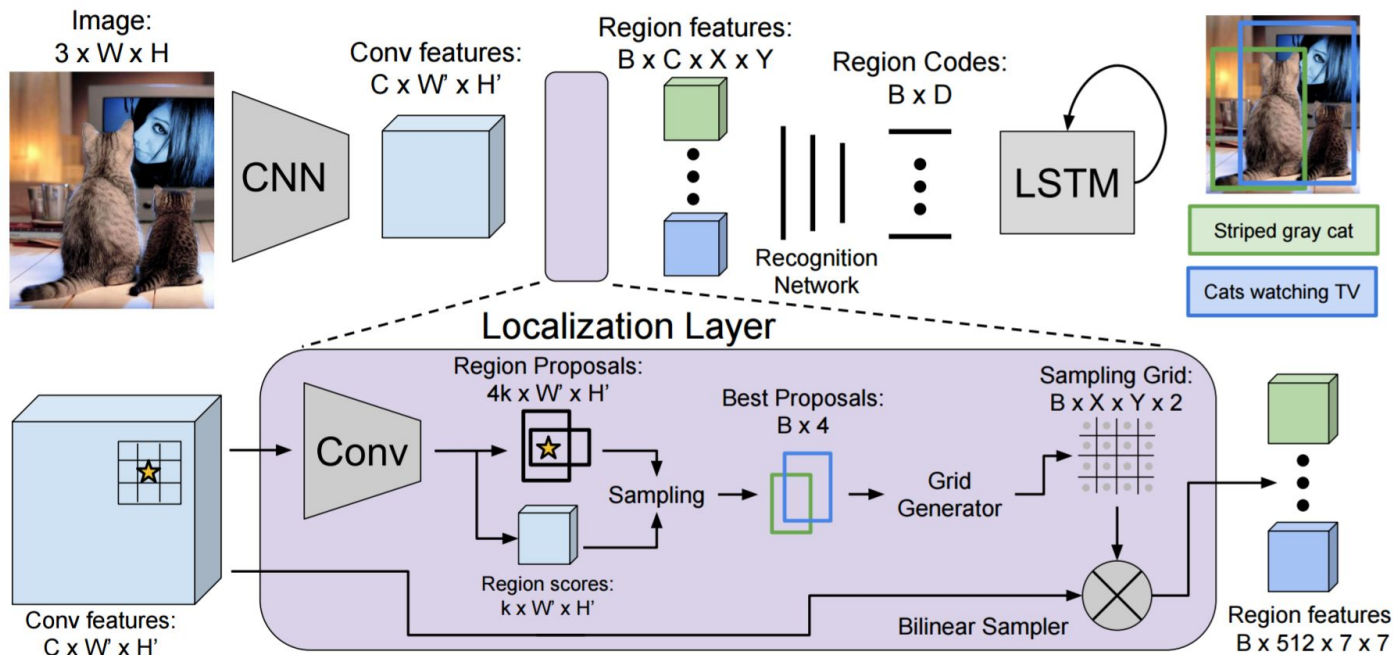
Finetune on your own dataset with pre-trained models

Aside: Object Detection + Captioning = Dense Captioning

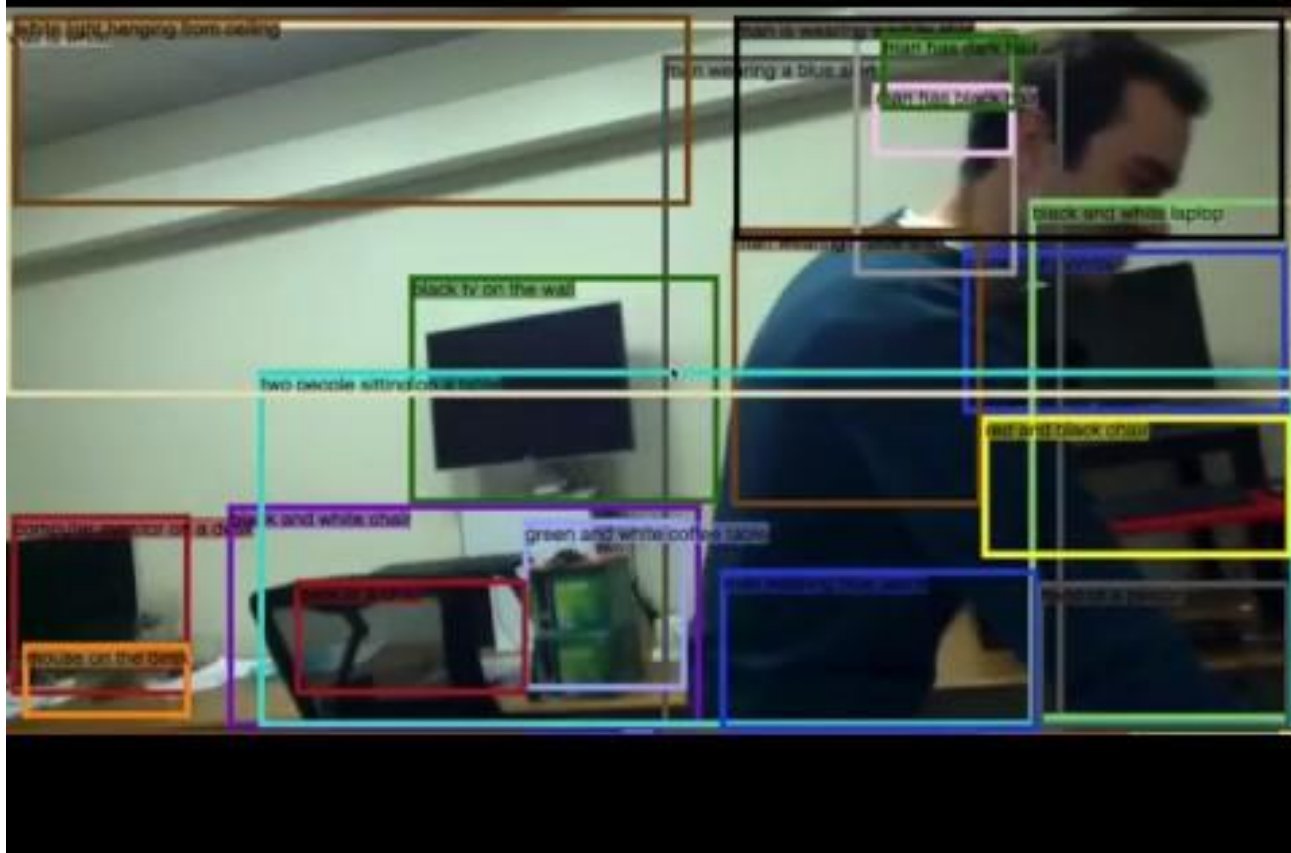


Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.

Aside: Object Detection + Captioning = Dense Captioning



Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.



The diagram illustrates a semantic network with the following nodes and relationships:

- Central Node:** person
- Actions (Green nodes):** walking, driving, wearing, playing, using, examining
- Objects/Entities (Red nodes):** crosswalk, car, fedora, guitar, electric, microscope, patient, lamp, counter, laying on, sunglasses, hanging on, stand, wooden
- States/Qualities (Blue nodes):** off, Light, sign, Dark, black, Adele, album, fedora, electric, Dark, wooden
- Other (Pink node):** bakery

Relationships (Edges):

- person → walking → crosswalk
- person → walking → car
- person → driving → car
- person → wearing → fedora
- person → playing → guitar
- person → using → electric
- person → using → microscope
- person → examining → patient
- person → lamp
- person → Light
- person → sign
- person → counter
- person → laying on
- person → sunglasses
- person → hanging on
- person → stand
- person → wooden
- lamp → off
- Light → sign
- sign → describing → bakery
- counter → laying on → sunglasses
- hanging on → stand → wooden

Image Annotations:

- Top Left Image (Street Scene):** Annotated with bounding boxes for 'sign', 'lamp', and 'crosswalk'.
- Top Right Image (Person Playing Guitar):** Annotated with bounding boxes for 'person', 'guitar', 'fedora', 'black', and 'sunglasses'.
- Bottom Left Image (Person Examining Patient):** Annotated with bounding boxes for 'person', 'microscope', 'patient', and 'Dark'.

5.4 Million Region Descriptions

3.8 Million Object Instances

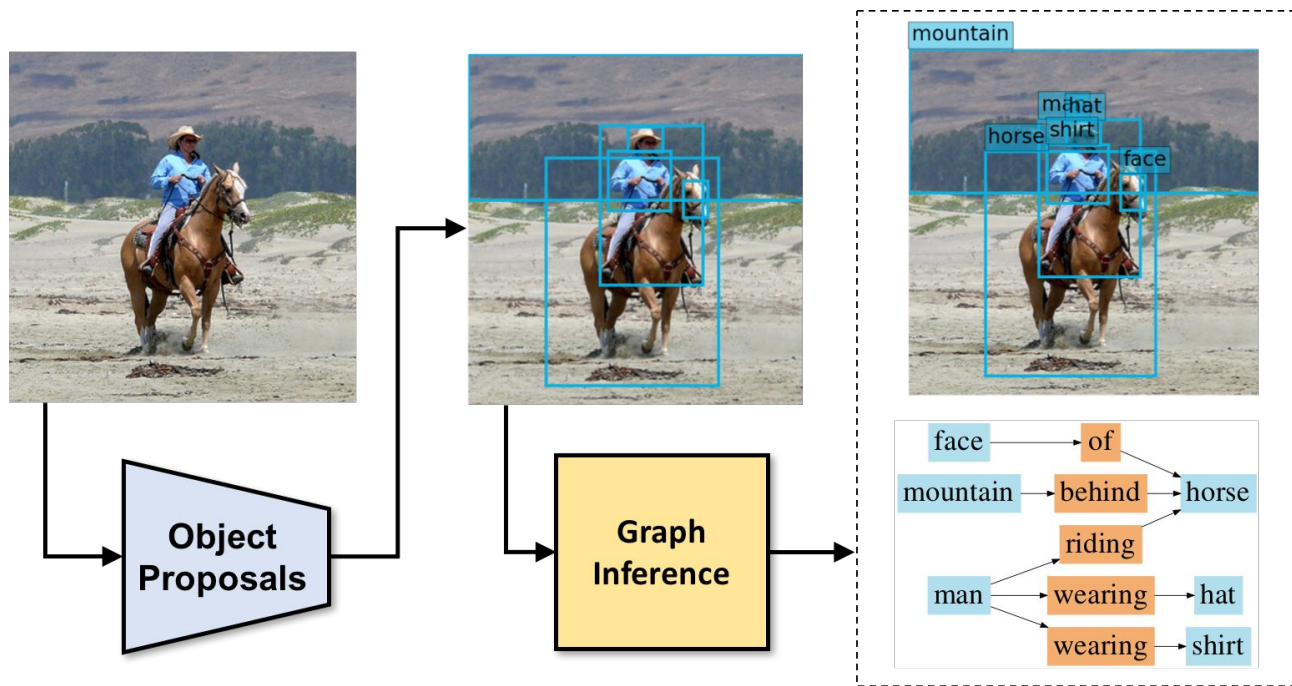
2.3 Million Relationships

Everything Mapped to Wordnet Synsets



Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." *International Journal of Computer Vision* 123, no. 1 (2017): 32-73.

Aside: Scene Graph Generation



Xu, Zhu, Choy, and Fei-Fei, "Scene Graph Generation by Iterative Message Passing", CVPR 2017
Figure copyright IEEE, 2018. Reproduced for educational purposes.

3D Object Detection

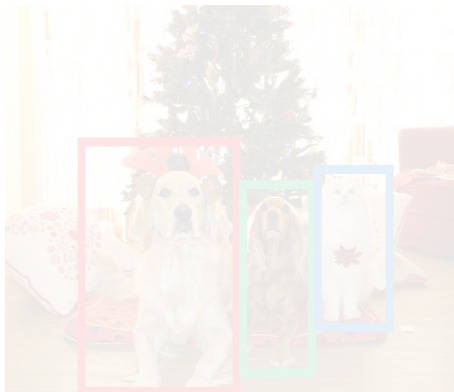
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

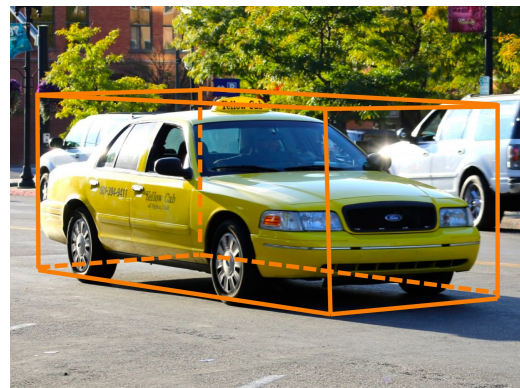
2D Object Detection



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

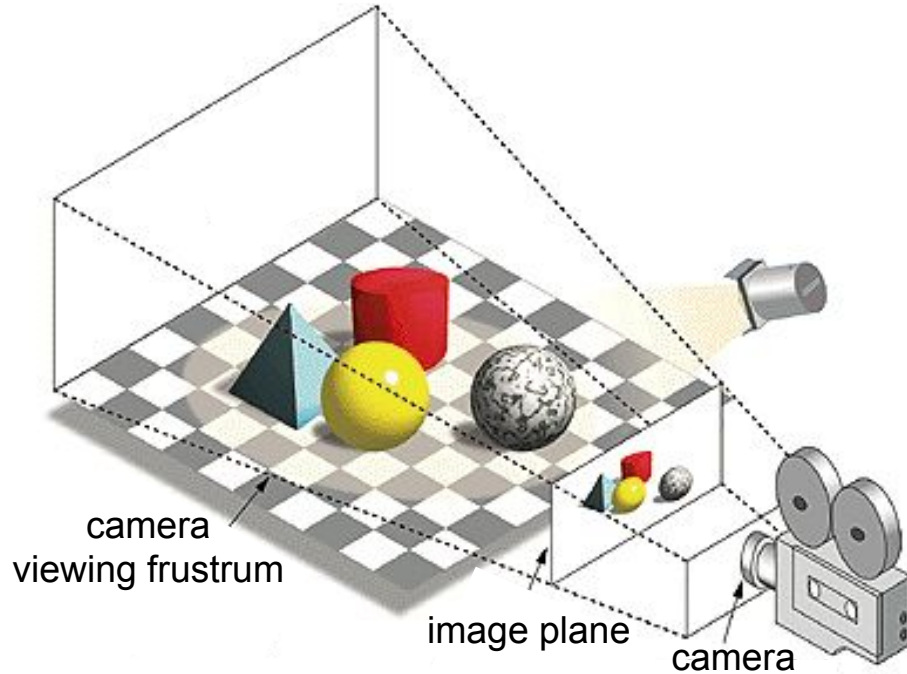
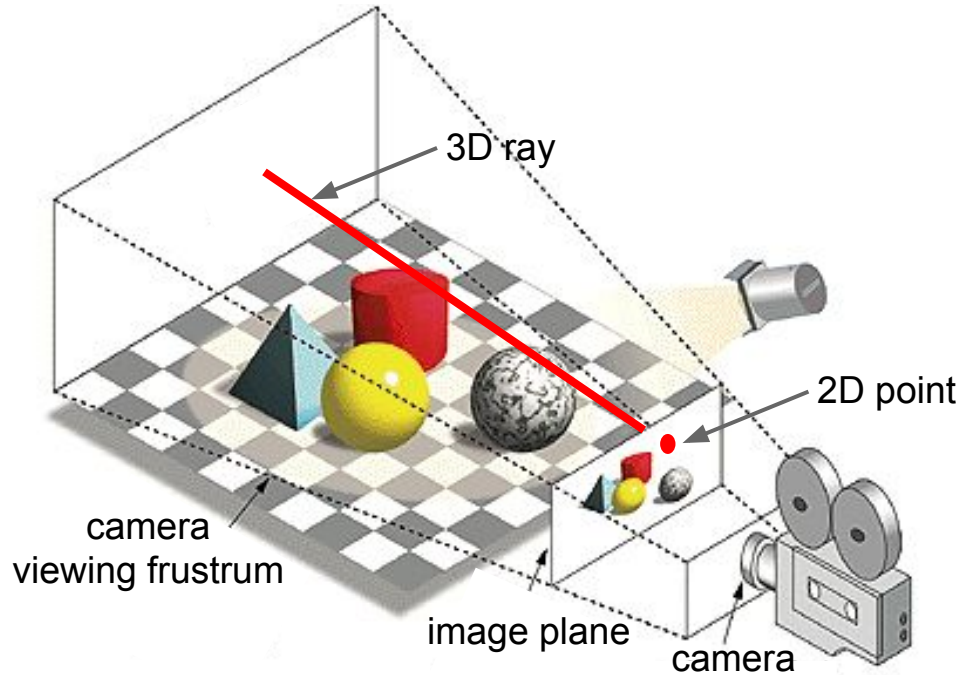


Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

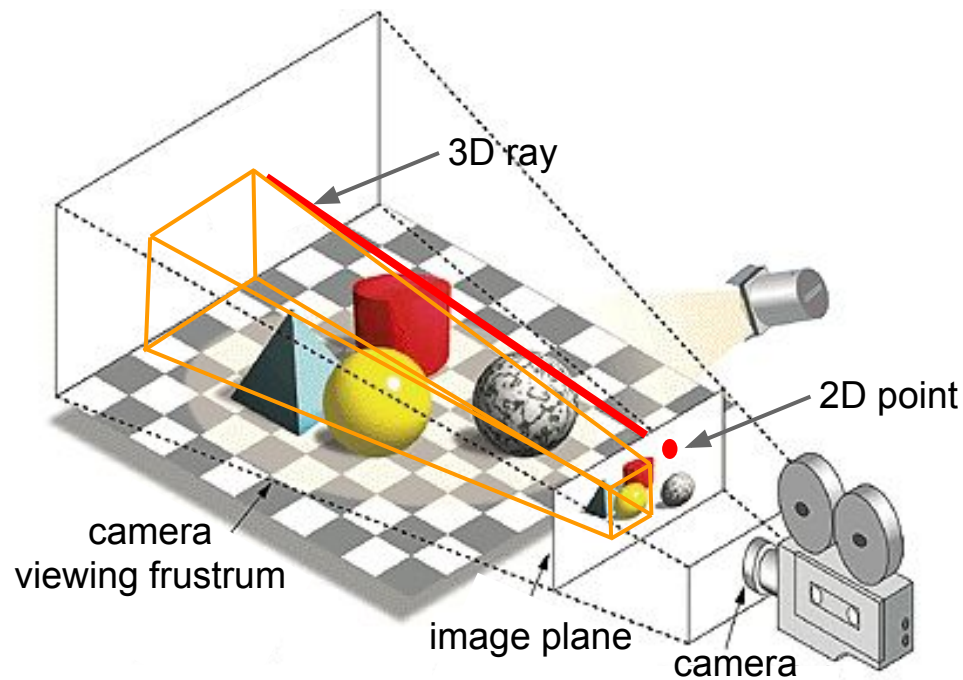
Simplified Camera Model



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

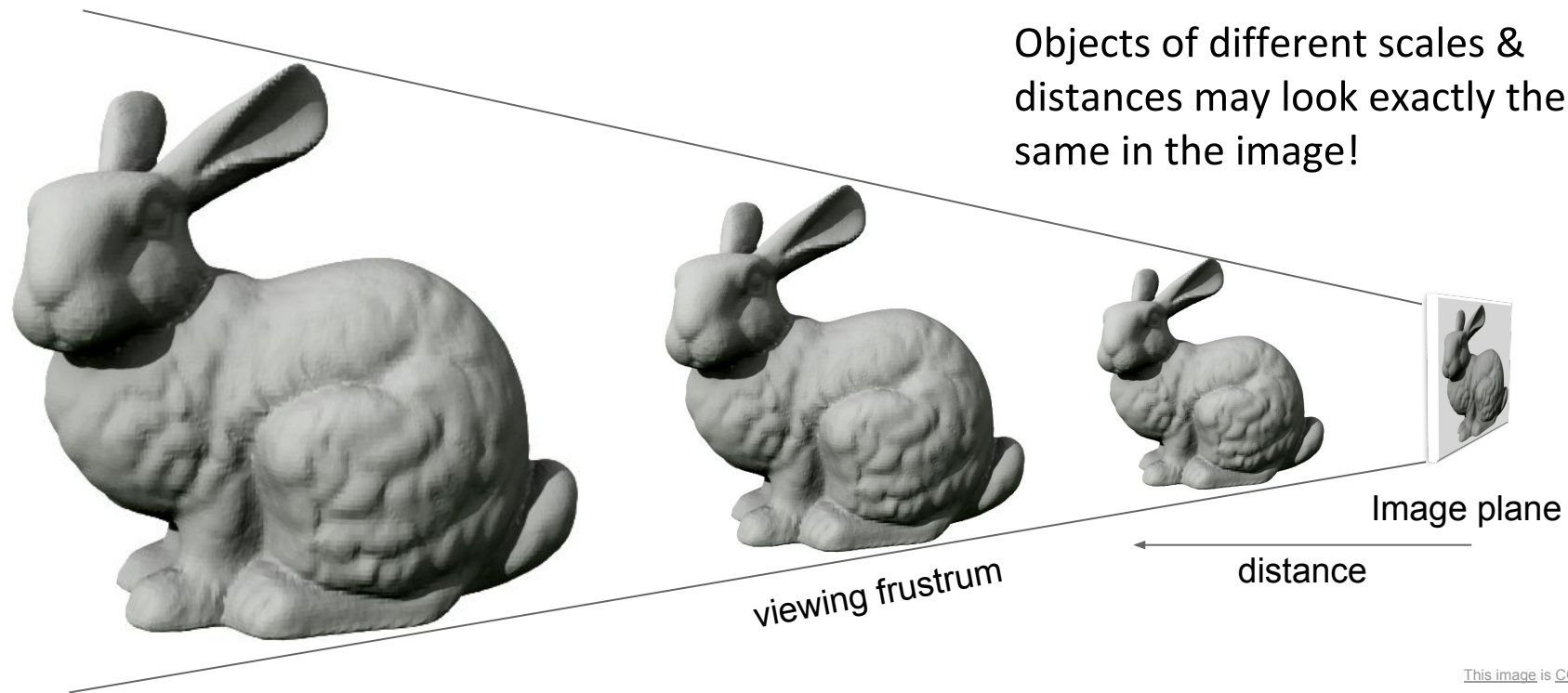


A point on the image plane corresponds to a **ray** in the 3D space

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

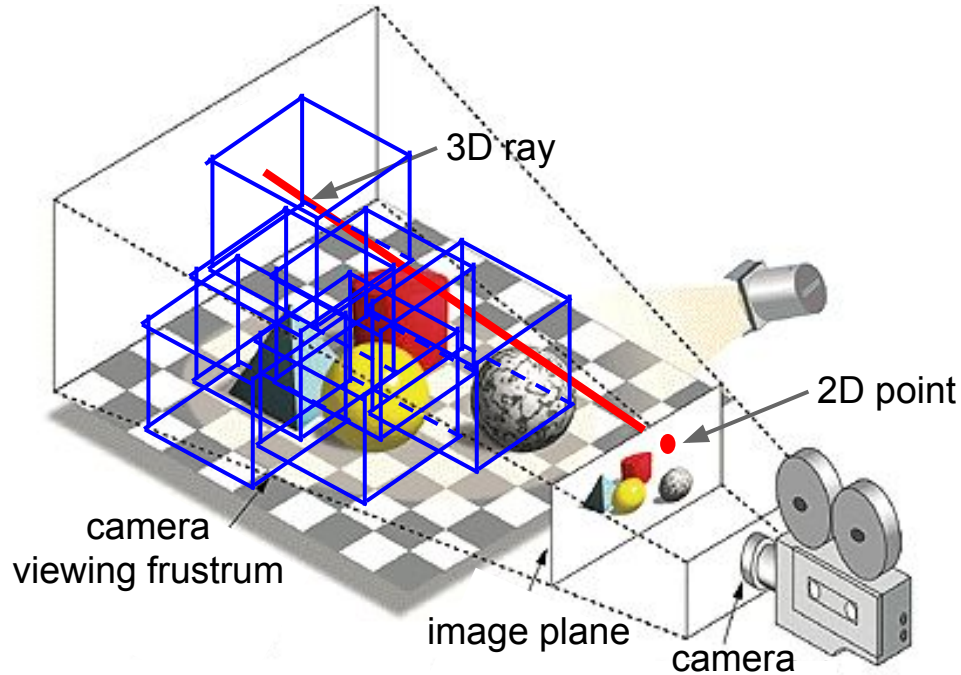
Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

Scale & Distance Ambiguity



[This image is CC0 public domain](#)

Simplified Camera Model



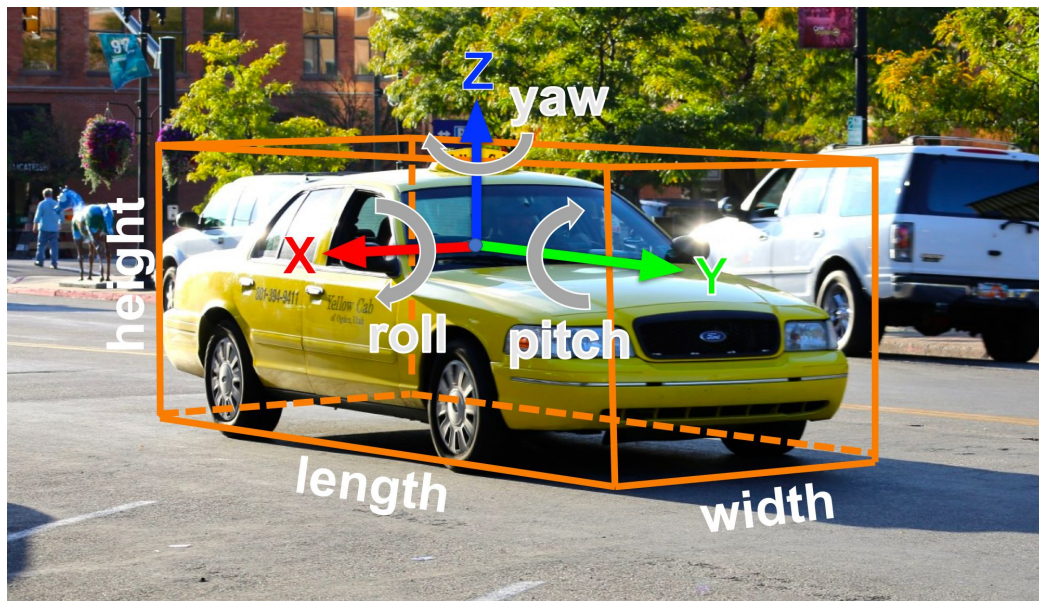
A point on the image plane corresponds to a **ray** in the 3D space

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

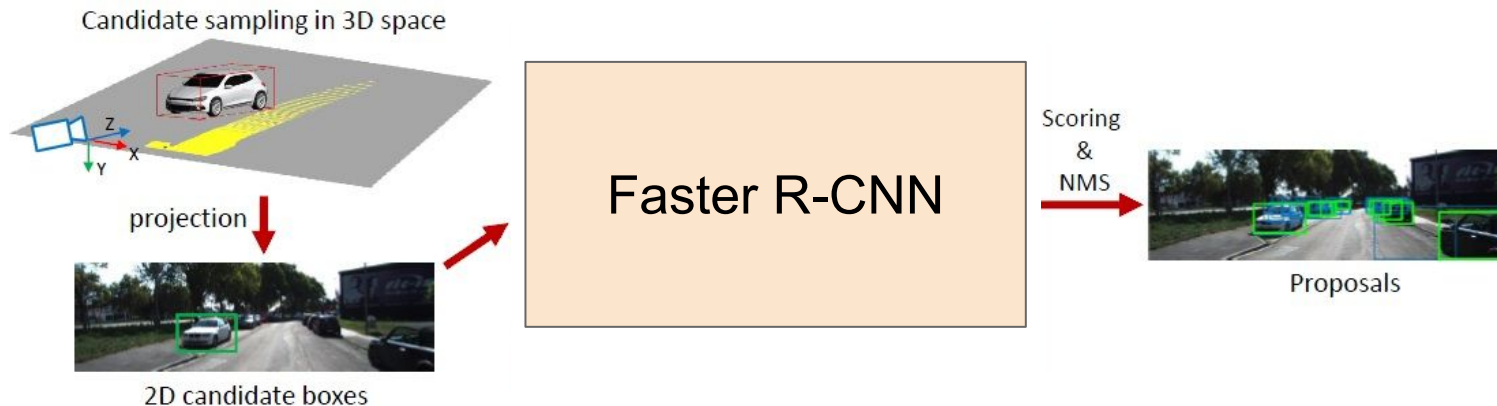
Much harder problem than 2D
object detection!

[This image is CC0 public domain](#)

3D Object Detection



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: Monocular View



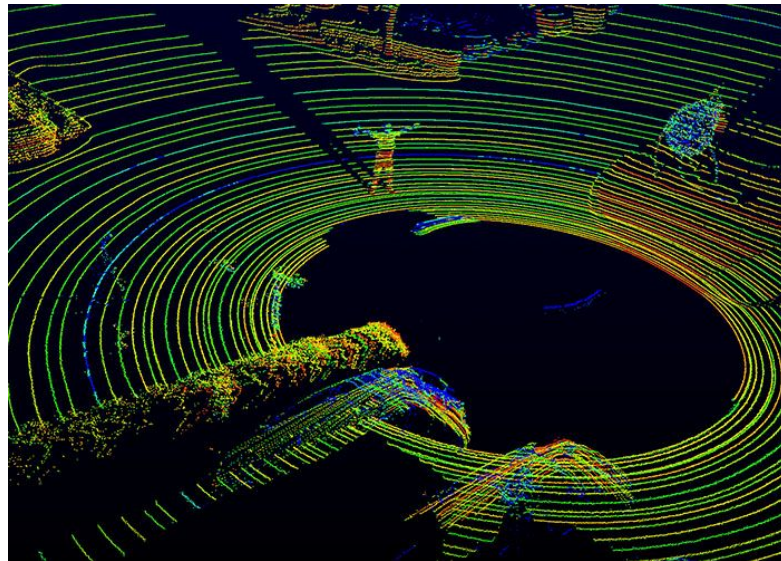
Chen, Xiaozi, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "Monocular 3d object detection for autonomous driving." CVPR 2016.

3D Object Detection: Camera + LiDAR

[This image is CC0 public domain](#)

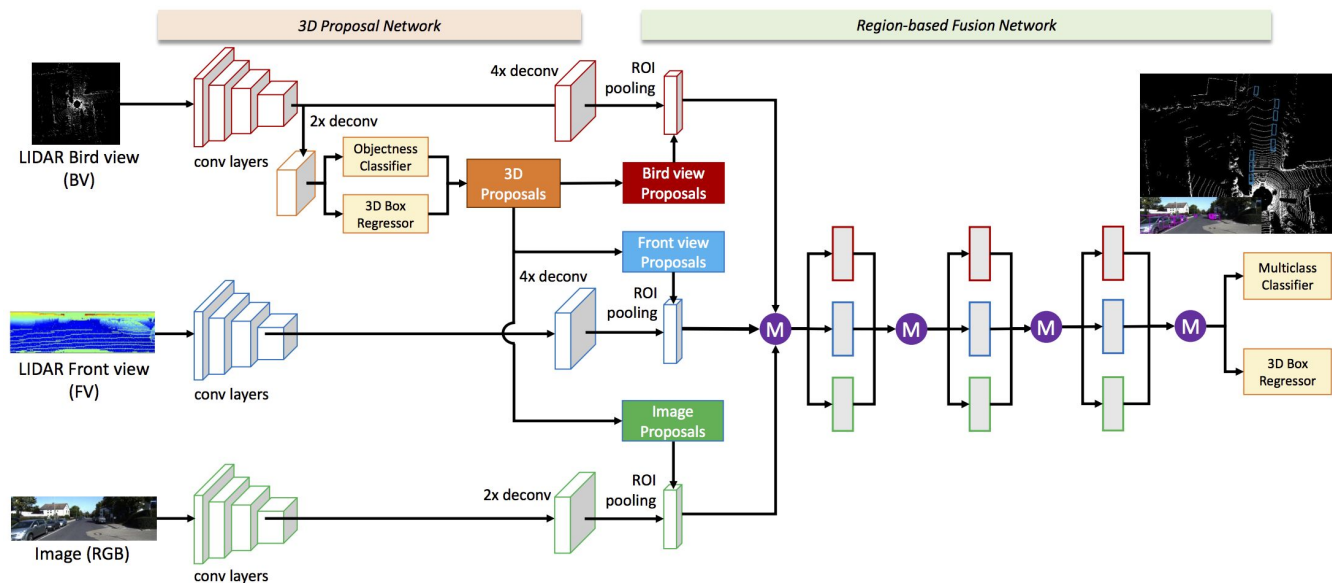


Velodyne (HDL-64e)



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



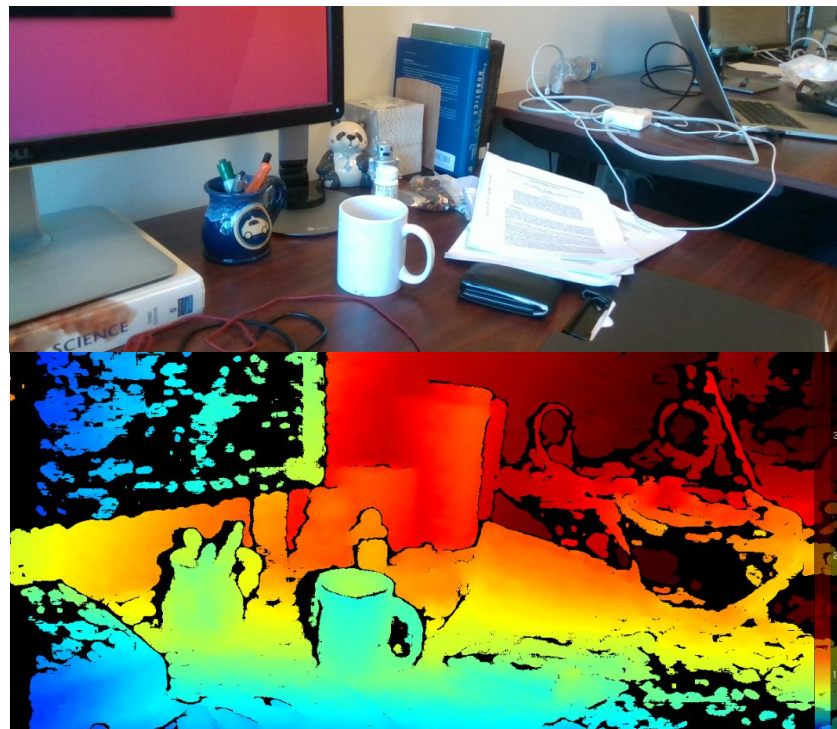
Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." *CVPR* 2017

RGB-Depth Camera

[This image is CC0 public domain](#)



Kinect (Xbox One)



RGB

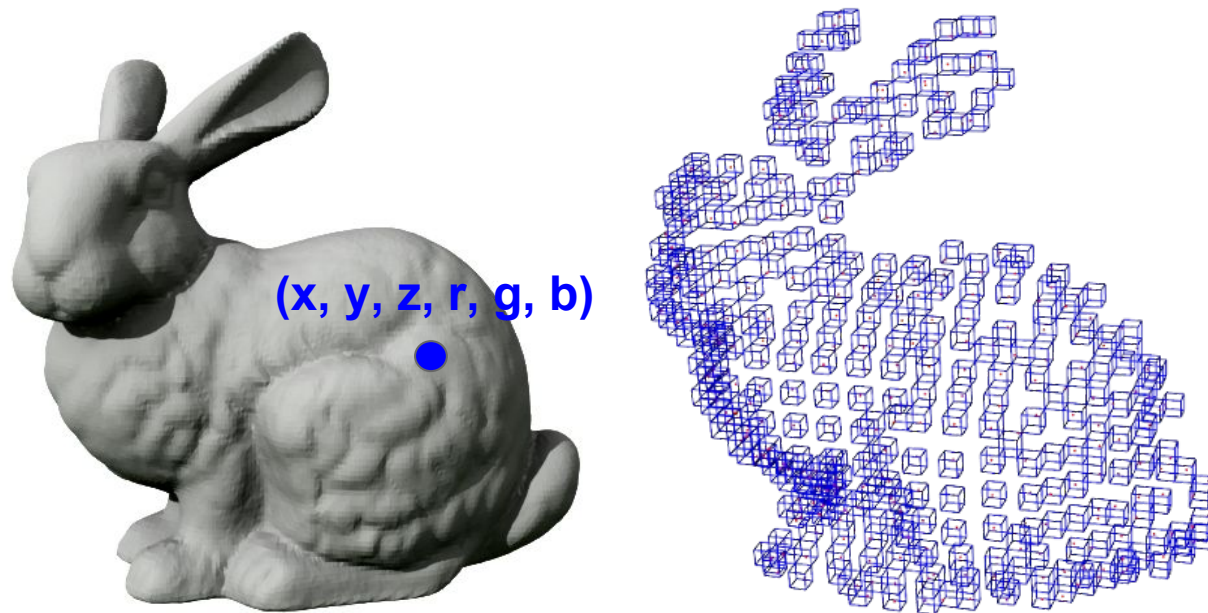
Depth

RGB-Depth Camera



Registered RGB + depth point cloud

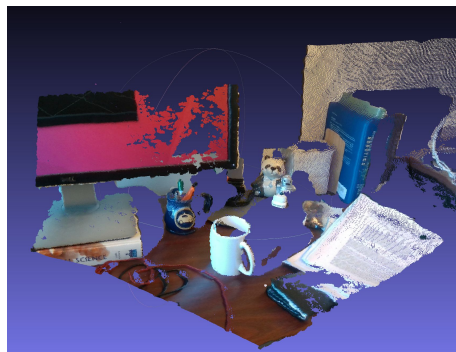
Point Cloud Voxelization



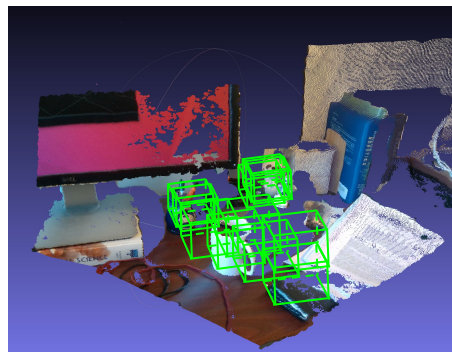
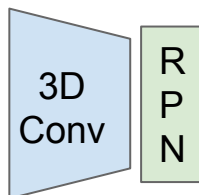
1. Capture RGB-D point cloud of a scene.
2. Partition the 3D space into a regular 3D grid.
3. For each grid cell that has a point fall into it, fill the cell with the RGB value of that point.

A bit like “3D image”

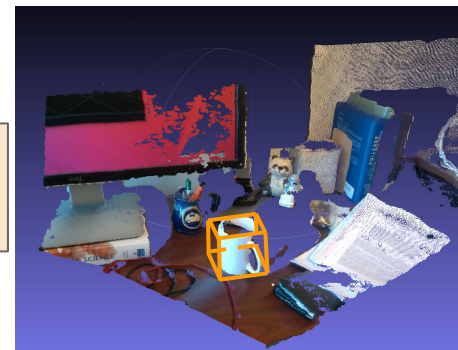
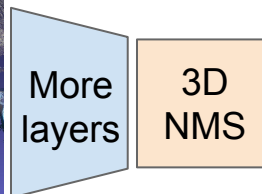
3D Object Detection: RGB-Depth Camera



Voxelized RGB-D point cloud



3D region proposals



Object categories +
3D bounding boxes

“Faster RCNN in 3D”

S. Song, and J. Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. CVPR 2016

Recap

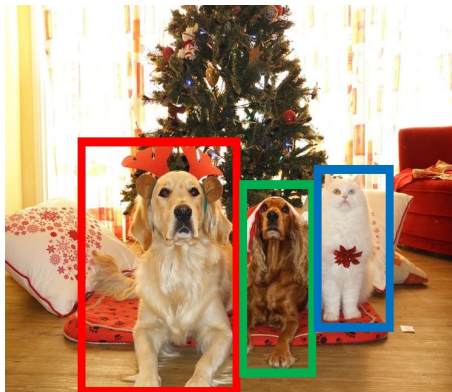
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

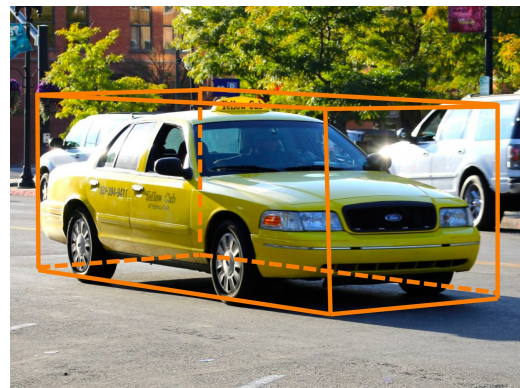
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

Object categories +
3D bounding boxes

[This image is CC0 public domain](#)

Next time:
Visualizing CNN features
DeepDream + Style Transfer