

# **CS 4644-DL / 7643-A: LECTURE 16**

## **DANFEI XU**

Deep Learning Application to Computer Vision

- Semantic Segmentation
- Object Detection
- Instance Segmentation

# Administrative

- Milestone rubric released.
- Presentation -> video recording.
- Milestone + presentation due Nov 7<sup>th</sup>.
- NO GRACE PERIOD

# The rest of the semester

---

W9: Oct 19	Computer Vision: Detection and Segmentation <a href="#">Slides (pdf)</a>	<ul style="list-style-type: none"><li>• <a href="#">Fully Convolutional Networks for Semantic Segmentation</a></li></ul>
W10: Oct 24	Visualizing Neural Networks <a href="#">Slides (pdf)</a>	<ul style="list-style-type: none"><li>• <a href="#">Understanding Neural Networks Through Deep Visualization</a></li><li>• <a href="#">Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization</a></li></ul>
W10: Oct 26	Generative Models - Autoregressive Models and Variational Autoencoders (VAE) <a href="#">Slides (pdf)</a>	
W11: Oct 31	Generative Models - Denoising Diffusion Probabilistic Models <a href="#">Slides (pdf)</a>	
W11: Nov 2	Self-Supervised Learning <a href="#">Slides (pdf)</a>	
W12: Nov 7	Embodied Reasoning Through Planning with Language and Vision Foundation Models Guest Lecture by <a href="#">Fei Xia</a> , Google DeepMind	
W12: Nov 9	TBD	

# The rest of the semester

---

W13: Nov 14 Graph Neural Networks  
Guest Lecture by [Jiaxuan You](#), NVIDIA Research

---

W13: Nov 16 Reinforcement Learning 1: MDP, Value Iteration, Deep Q Learning.  
PS/HW4 due Nov 11th 11:59pm  
[Slides \(pdf\)](#).

- [Sutton & Bartow Chapter 1](#)
- [Survey paper on Deep RL](#)
- [MDP Notes \(courtesy Byron Boots\)](#)

W14: Nov 21 Reinforcement Learning 2: Actor-Critic, Frontiers.  
[Slides \(pdf\)](#).

- [Sutton & Bartow Chapter 1](#)
- [Survey paper on Deep RL](#)
- [MDP Notes \(courtesy Byron Boots\)](#)

W14: Nov 23 Thanksgiving Holiday - No Class

W15: Nov 28 Foundation Models for Robotics  
Guest Lecture by [Siddharth Karamcheti](#), Stanford University

---

W15: Nov 30 Robot Learning Teaser + Conclusion

W16: Dec 5 Poster Session (Klaus Atrium): 5-7PM

# Image Classification: A core task in Computer Vision



This image by [Nikita](#) is licensed under [CC-BY 2.0](#)

(assume given a set of possible labels)  
{dog, cat, truck, plane, ...}



cat

# Computer Vision Tasks

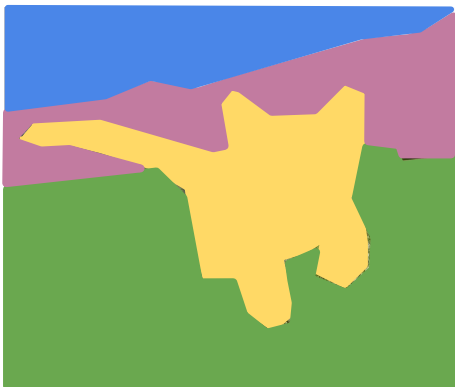
## Classification



**CAT**

No spatial extent

## Semantic Segmentation



**GRASS, CAT,  
TREE, SKY**

No objects, just pixels

## Object Detection



**DOG, DOG, CAT**

Multiple Object

## Instance Segmentation



**DOG, DOG, CAT**

[This image is CC0 public domain](#)

# Semantic Segmentation

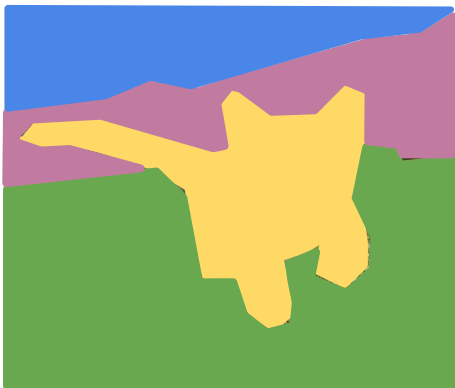
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

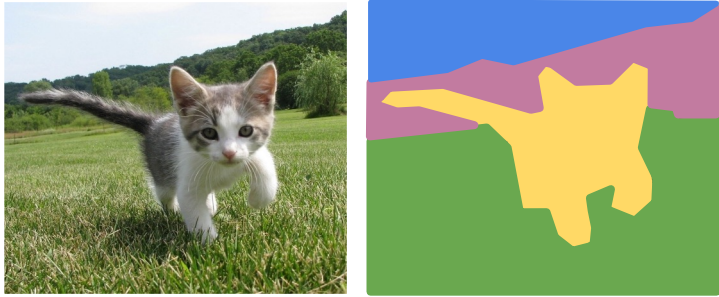
Multiple Object

Instance Segmentation



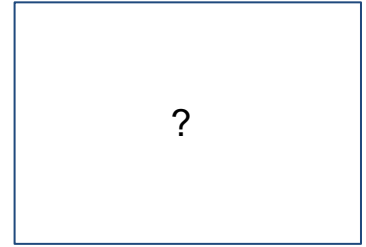
DOG, DOG, CAT

# Semantic Segmentation: The Problem



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

Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.



# Semantic Segmentation Idea: Sliding Window

Full image



# Semantic Segmentation Idea: Sliding Window

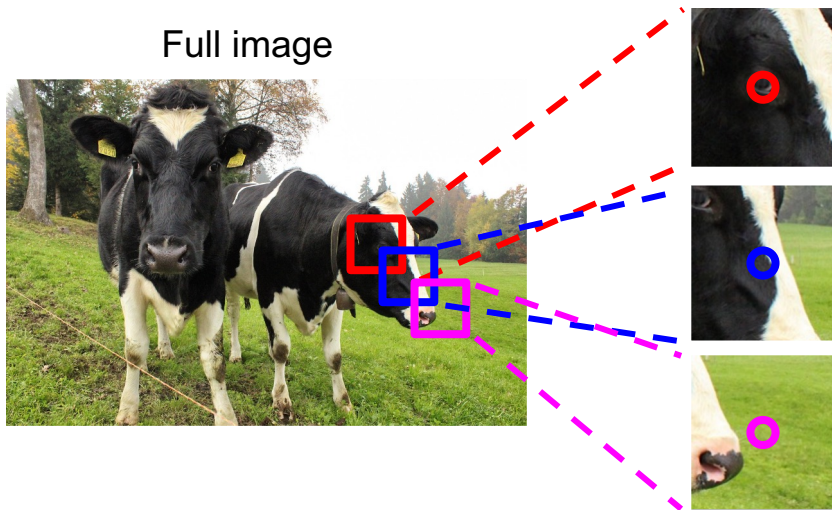
Full image



Impossible to classify without context

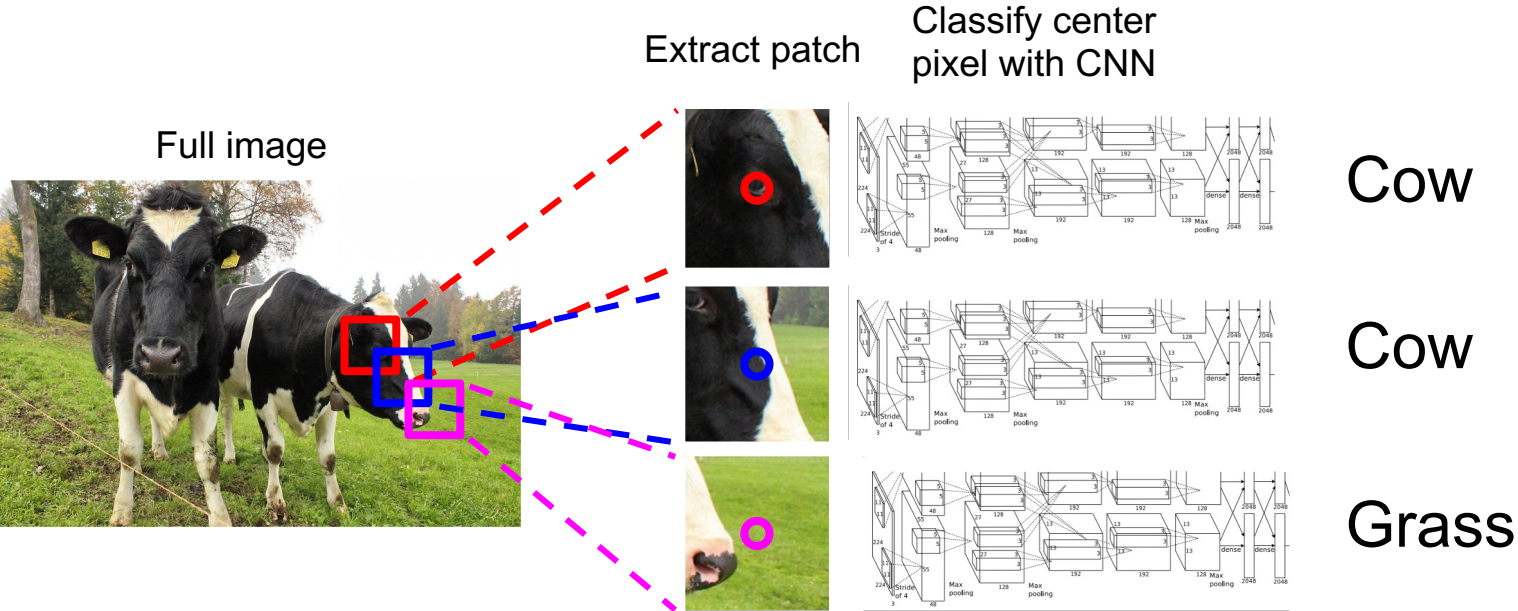
Q: how do we include context?

# Semantic Segmentation Idea: Sliding Window



Q: how do we model this?

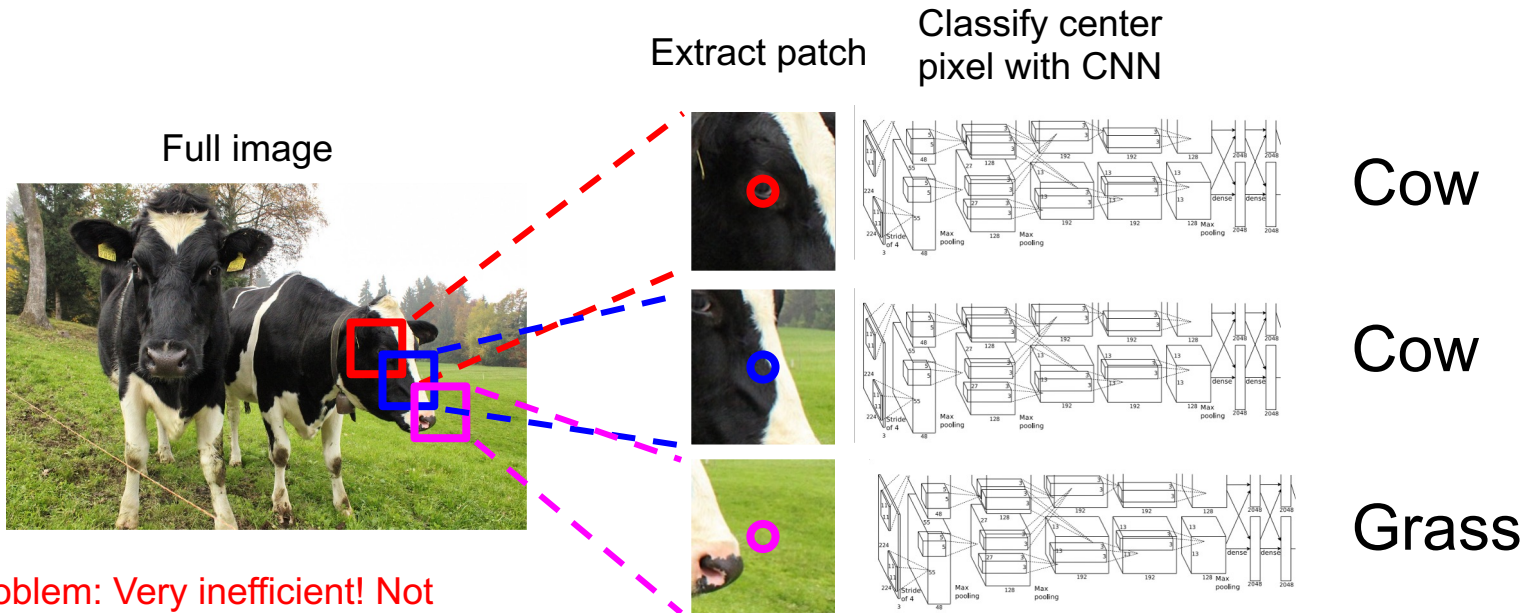
# Semantic Segmentation Idea: Sliding Window



## The “sliding window” approach

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



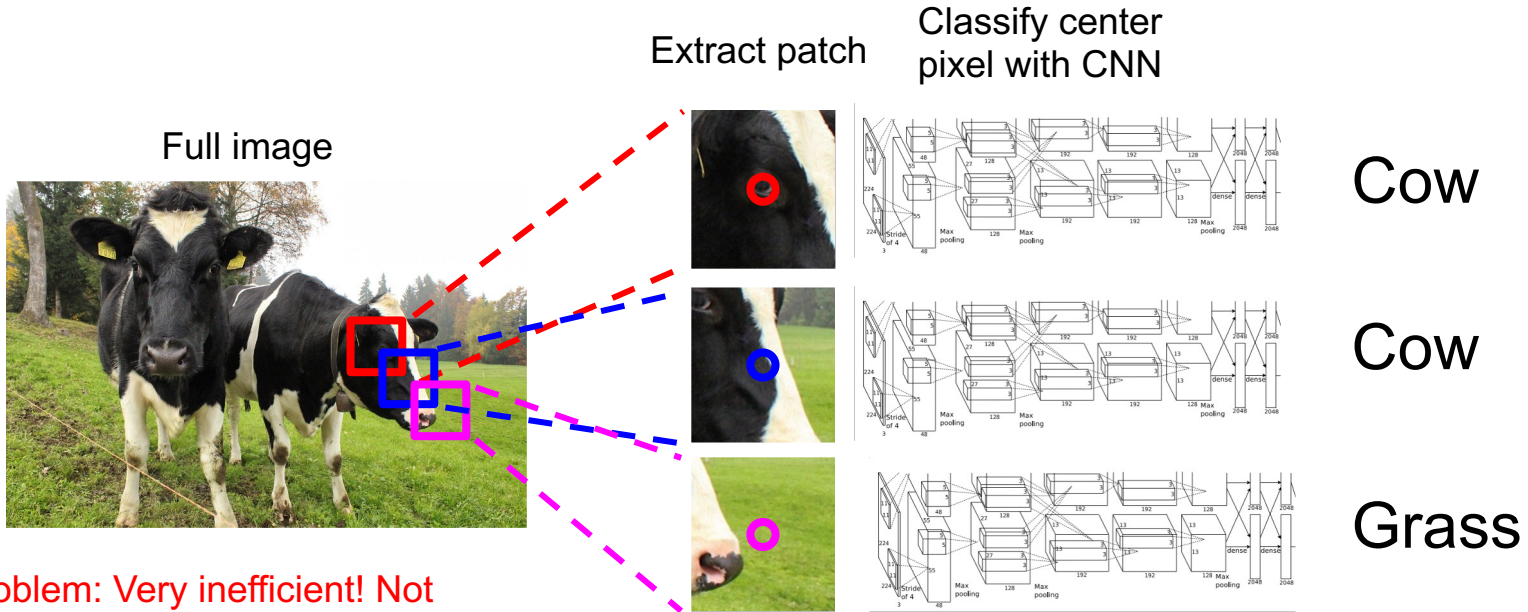
Problem: Very inefficient! Not reusing shared features between overlapping patches

## The “sliding window” approach

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

Observation: lots of duplicate computation in nearby pixels

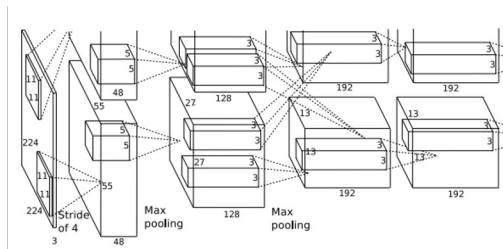
## The “sliding window” approach

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: Convolution

Full image

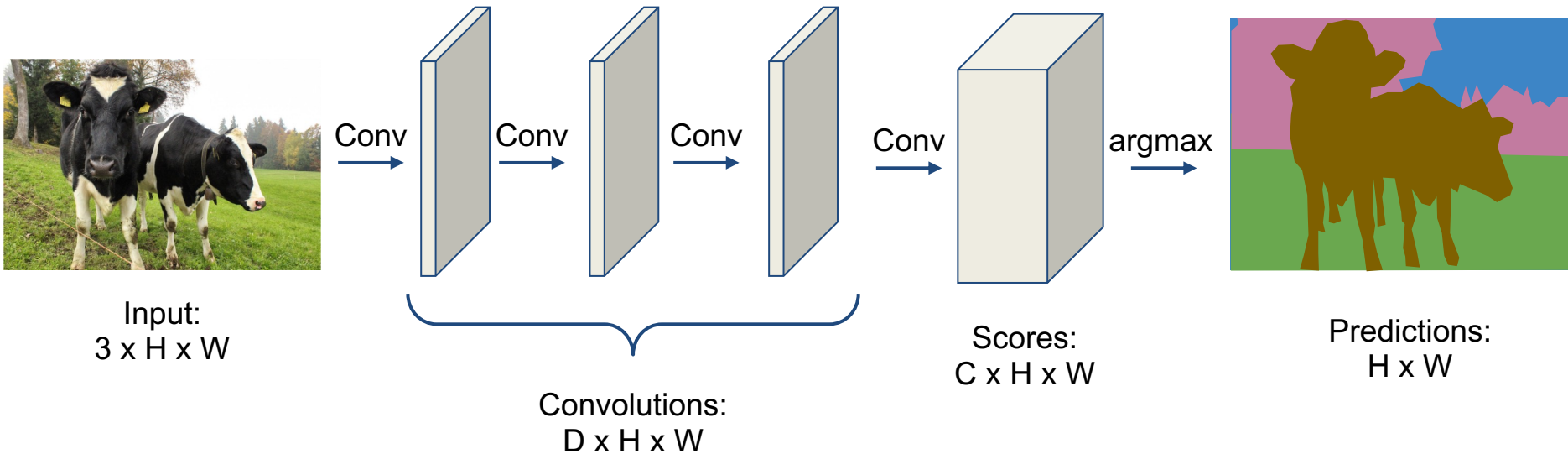


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

**Problem:** classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

# Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!

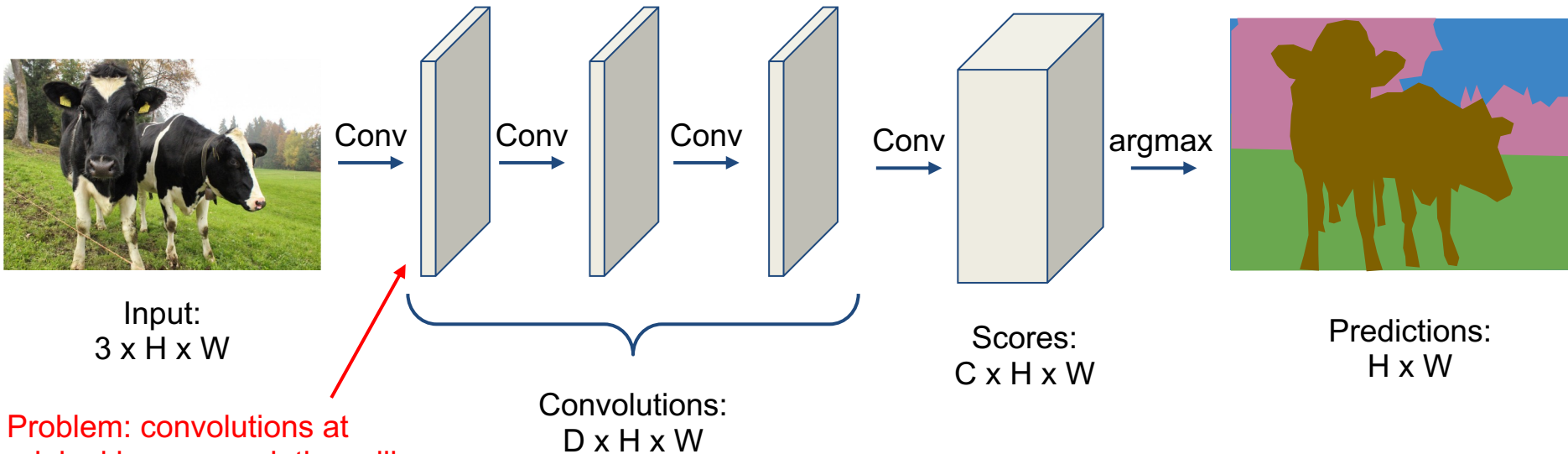


Loss: Pixel-wise cross entropy!



# Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



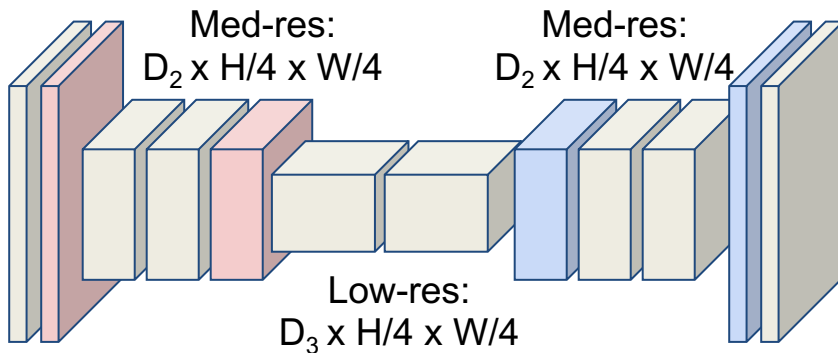
Problem: convolutions at original image resolution will be very expensive ...

# Semantic Segmentation Idea: Fully Convolutional

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



Input:  
 $3 \times H \times W$



High-res:  
 $D_1 \times H/2 \times W/2$

High-res:  
 $D_1 \times H/2 \times W/2$

$C \times H \times W$



Predictions:  
 $H \times W$

# Semantic Segmentation Idea: Fully Convolutional

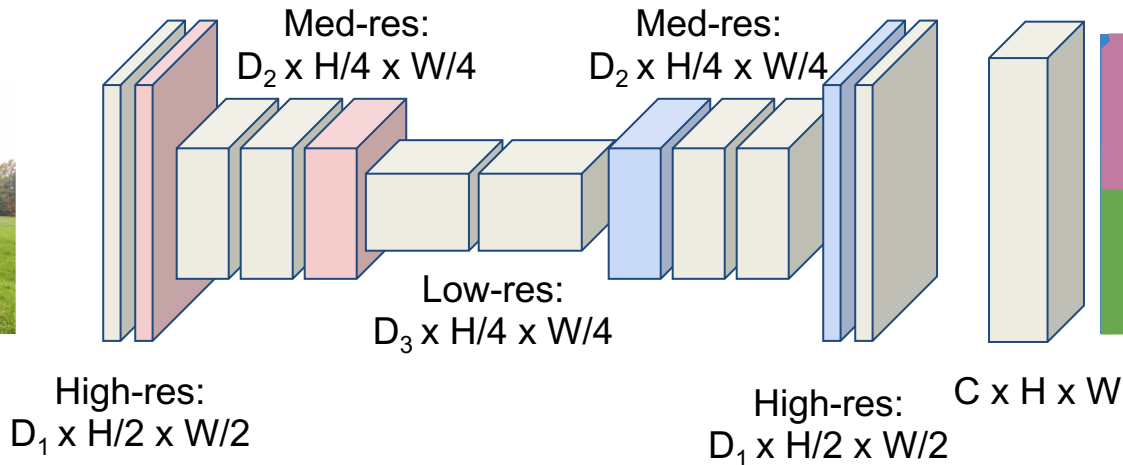
**Downsampling:**  
Pooling, strided  
convolution



Input:  
 $3 \times H \times W$

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

**Upsampling:**  
???



High-res:  
 $D_1 \times H/2 \times W/2$

High-res:  
 $D_1 \times H/2 \times W/2$

$C \times H \times W$

Predictions:  
 $H \times W$

# 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”

## 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

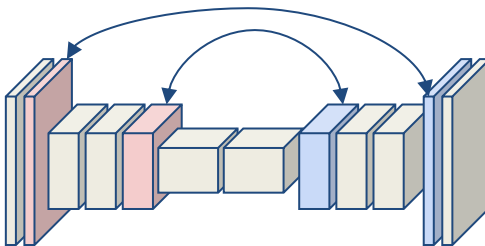
Input: 2 x 2



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

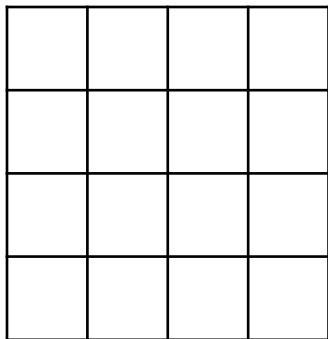
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

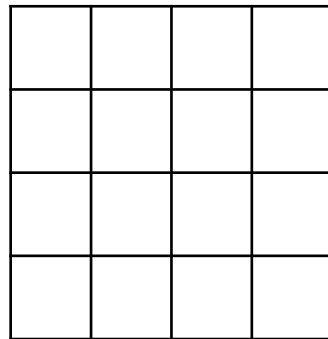


# Learnable Upsampling

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



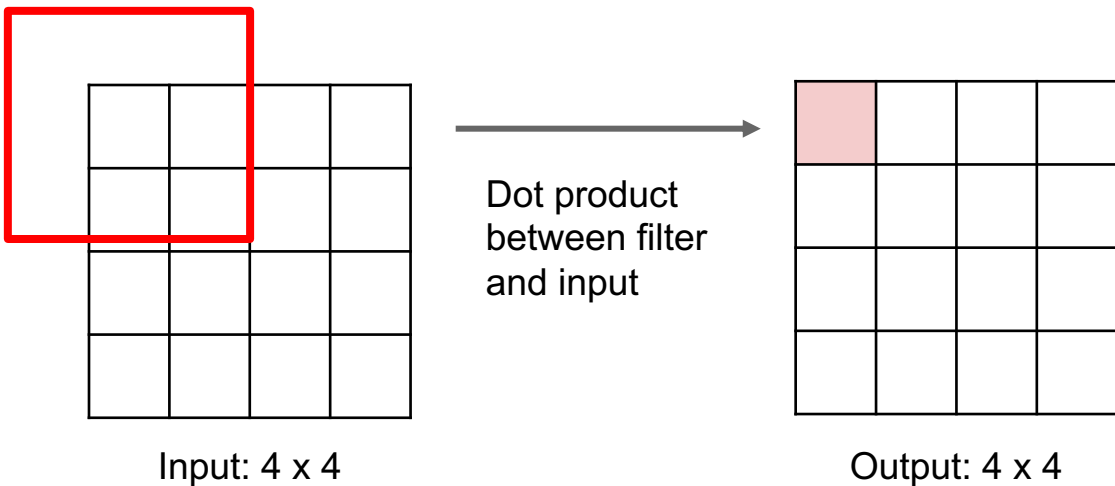
Input: 4 x 4



Output: 4 x 4

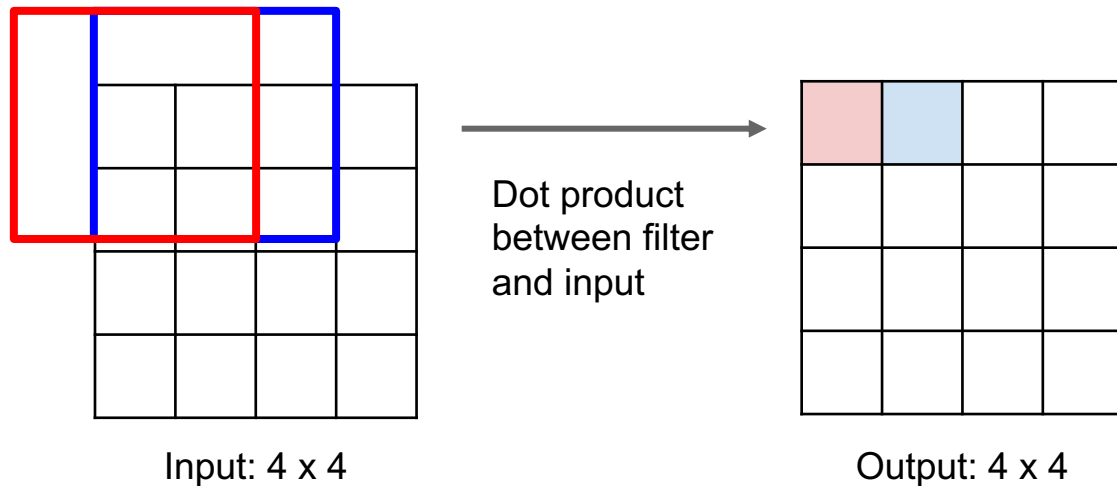
# Learnable Upsampling

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



# Learnable Upsampling

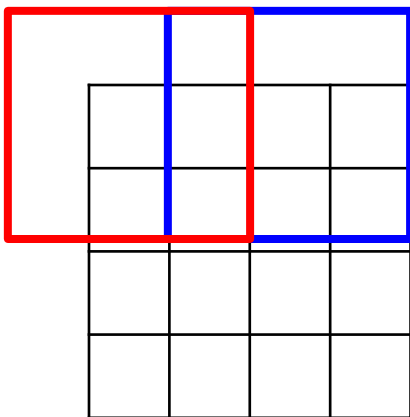
**Recall:** Normal 3 x 3 convolution, stride 1 pad 1





# Learnable Upsampling

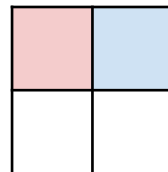
**Recall:** Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4



Dot product  
between filter  
and input



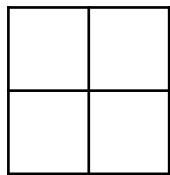
Output: 2 x 2

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

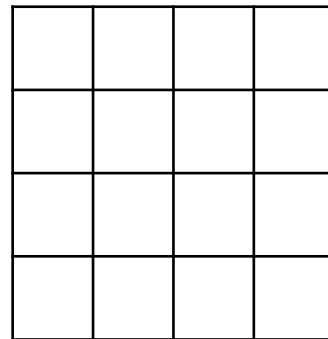
Stride gives ratio between  
movement in input and  
output

# Learnable Upsampling: Transposed Convolution

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



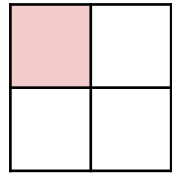
Input: 2 x 2



Output: 4 x 4

# Learnable Upsampling: Transposed Convolution

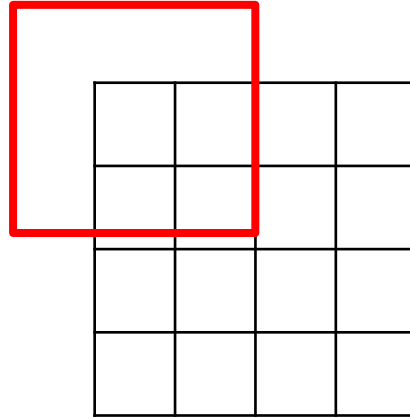
3 x 3 **transpose** convolution, stride 2 pad 1



Input: 2 x 2



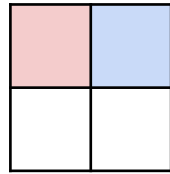
Input gives  
weight for  
filter



Output: 4 x 4

# Learnable Upsampling: Transposed Convolution

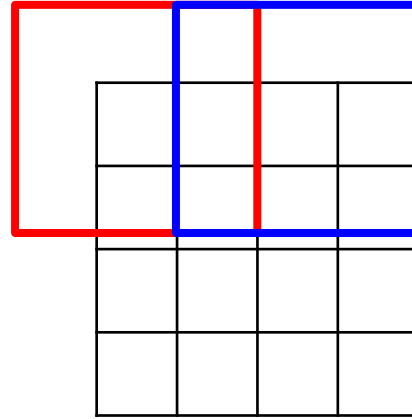
3 x 3 **transpose** convolution, stride 2 pad 1



Input: 2 x 2



Input gives  
weight for  
filter

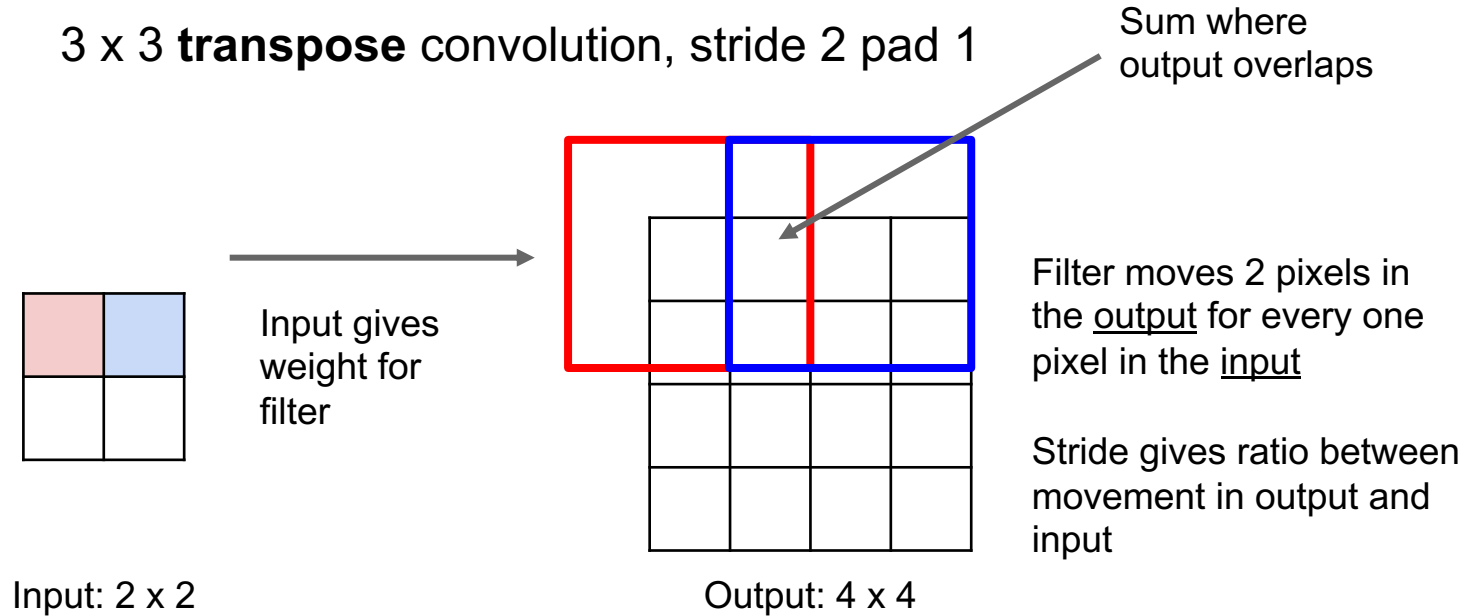


Output: 4 x 4

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

Stride gives ratio between movement in output and input

# Learnable Upsampling: Transposed Convolution



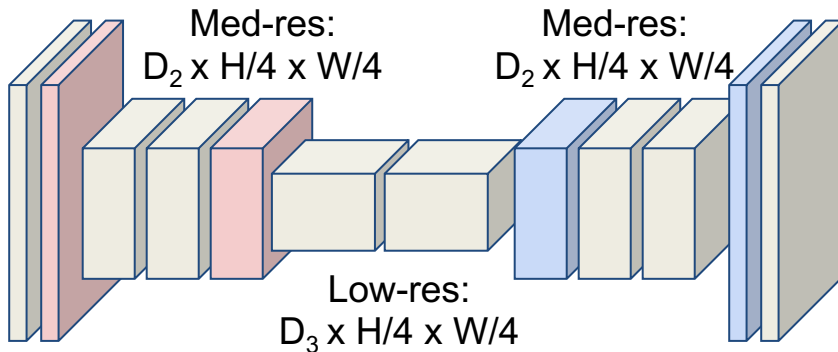
# Semantic Segmentation Idea: Fully Convolutional

**Downsampling:**  
Pooling, strided  
convolution



Input:  
 $3 \times H \times W$

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



High-res:  
 $D_1 \times H/2 \times W/2$

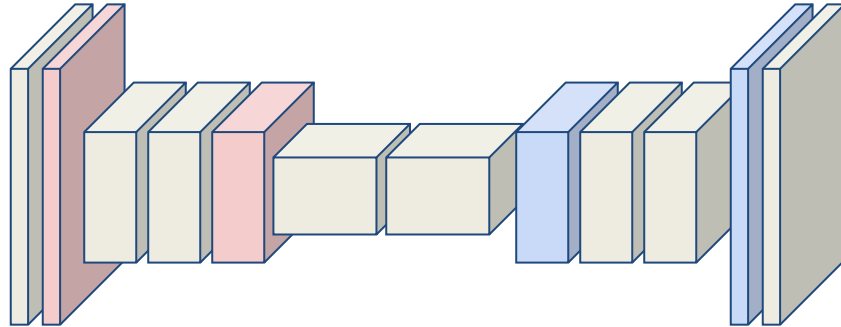
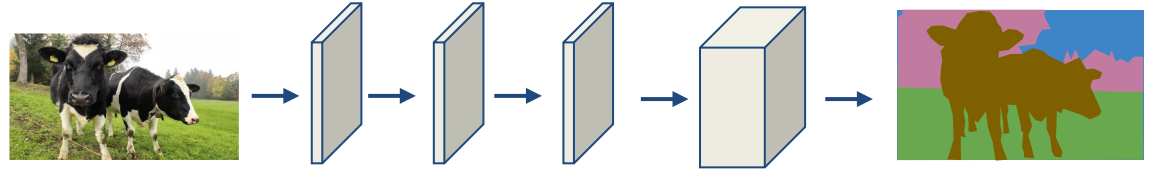
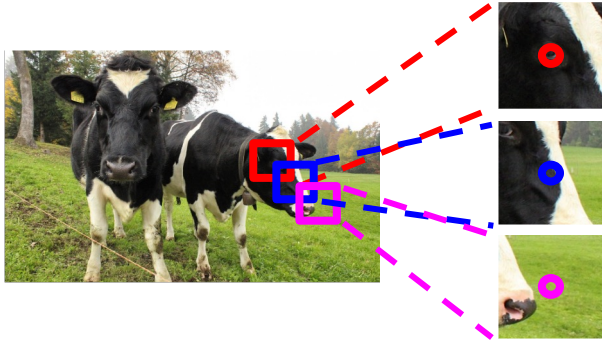
High-res:  
 $D_1 \times H/2 \times W/2$

**Upsampling:**  
Unpooling or strided  
transpose convolution

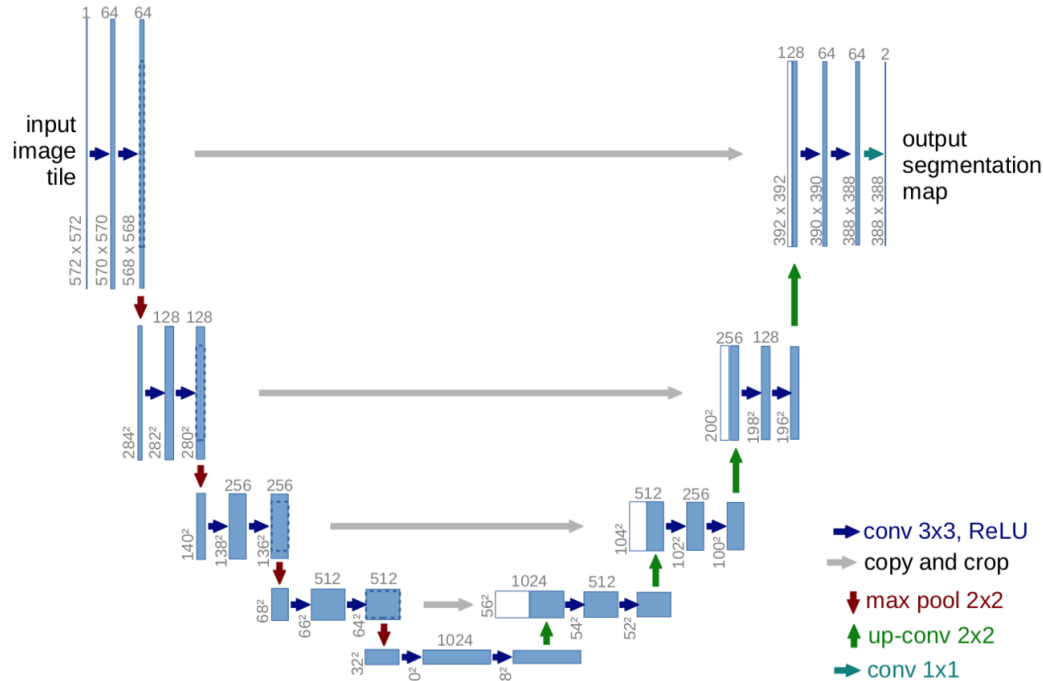


Predictions:  
 $H \times W$

# Semantic Segmentation: Summary



# Semantic Segmentation: U-Net



Idea: Concatenate feature maps from the downsampling stage with the features in the upsampling stage.

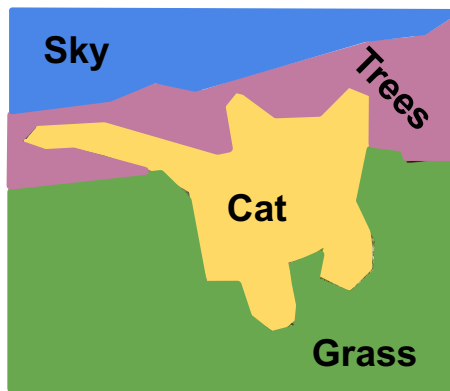
Very commonly used today!



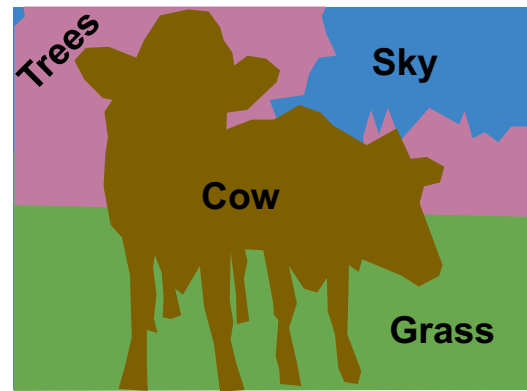
# Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



[This image is CC0 public domain](#)



# Object Detection

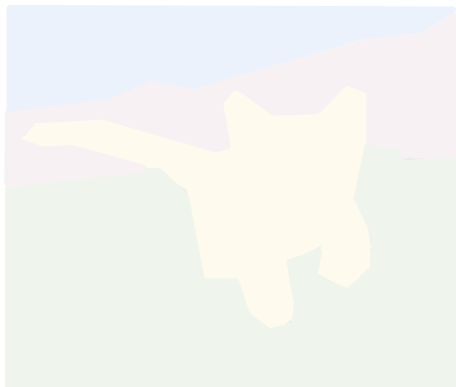
## Classification



CAT

No spatial extent

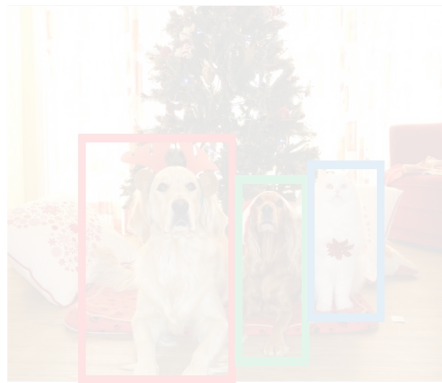
## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

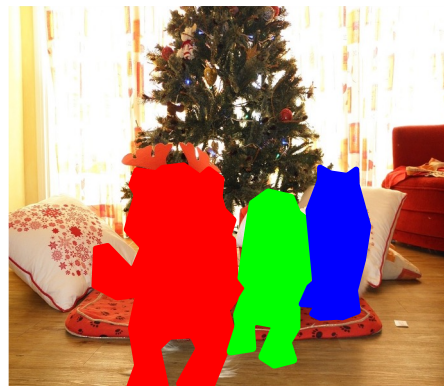
## Object Detection



DOG, DOG, CAT

Multiple Object

## Instance Segmentation



DOG, DOG, CAT

# Object Detection

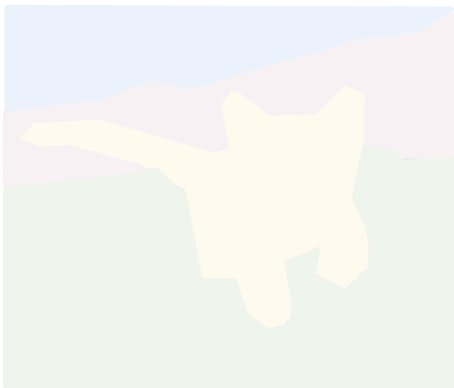
## Classification



CAT

No spatial extent

## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

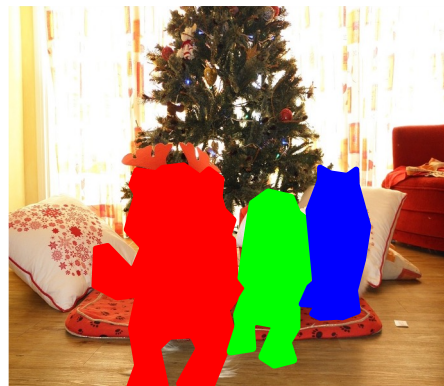
## Object Detection



DOG, DOG, CAT

Multiple Object

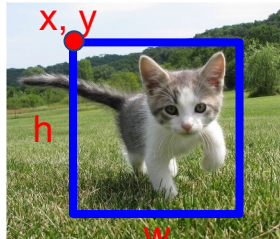
## Instance Segmentation



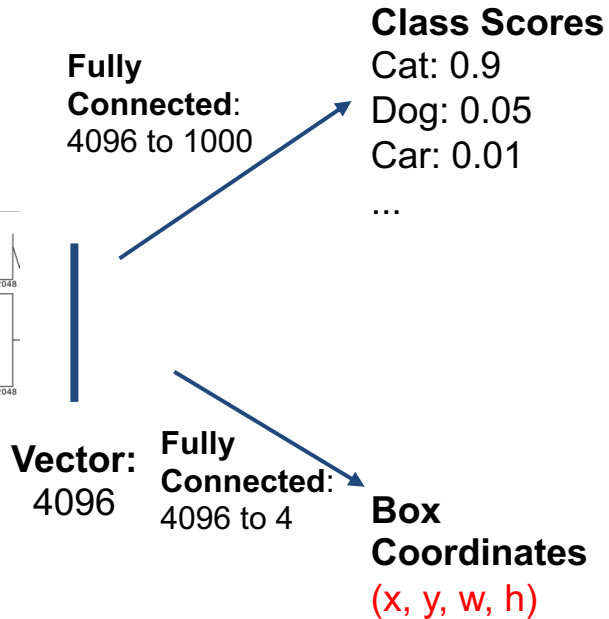
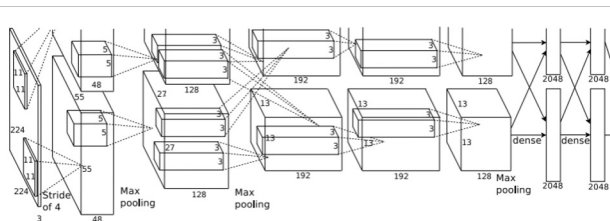
DOG, DOG, CAT

# Object Detection: Single Object

(Classification + Localization)

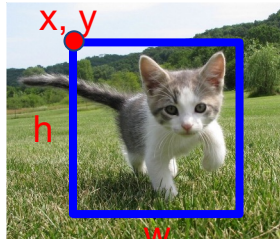


[This image](#) is [CC0 public domain](#)

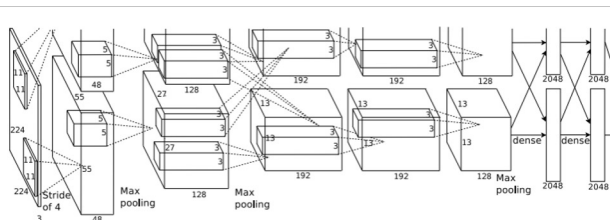


# Object Detection: Single Object

(Classification + Localization)



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



**Vector:**  
4096

**Fully Connected:**  
4096 to 1000

**Class Scores**

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

**Correct label:**  
Cat

**Softmax Loss**

**Fully Connected:**  
4096 to 4

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

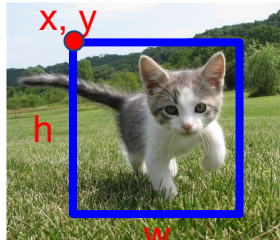
**L2 Loss**

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

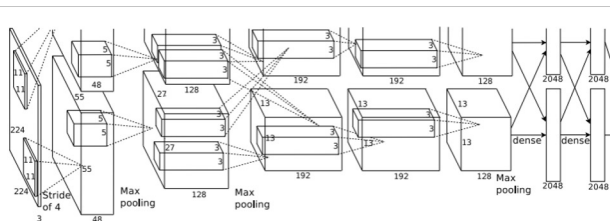
Treat localization as a regression problem!

# Object Detection: Single Object

(Classification + Localization)



[This image is CC0 public domain](#)



**Vector:**  
4096

**Fully Connected:**  
4096 to 1000

**Fully Connected:**  
4096 to 4

**Class Scores**

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

**Multitask Loss**

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

**Correct label:**  
Cat

**Softmax Loss**

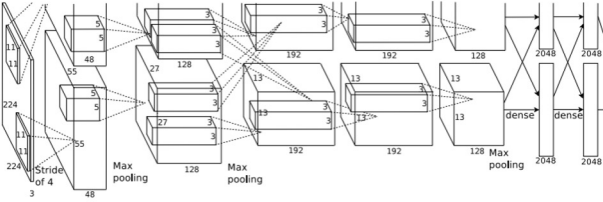
**+** → **Loss**

**L2 Loss**

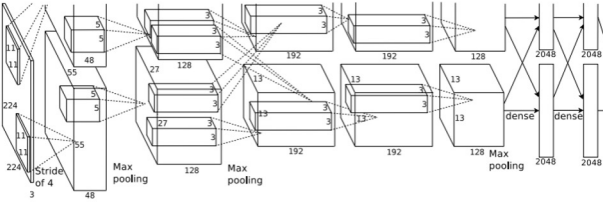
**Correct box:**  
(x', y', w', h')

Treat localization as a regression problem!

# Object Detection: Multiple Objects



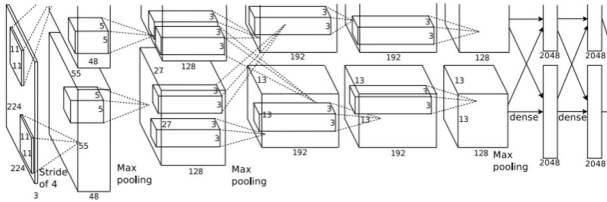
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



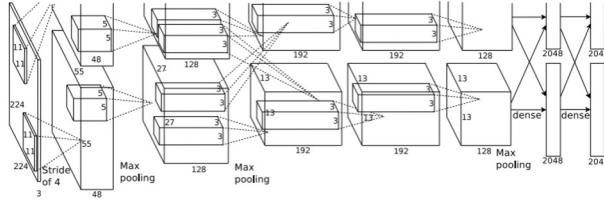
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

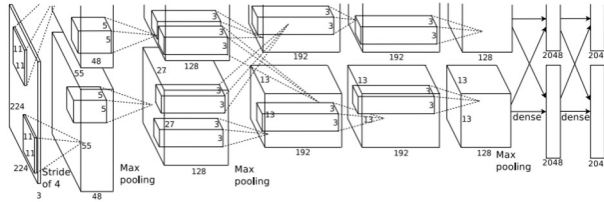
....

# Object Detection: Multiple Objects

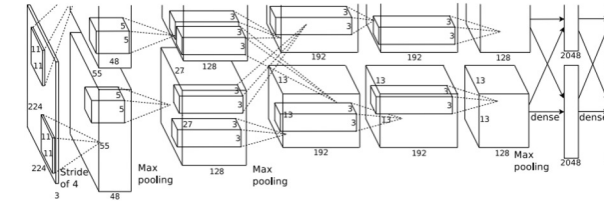
Each image needs a different number of outputs!



CAT:  $(x, y, w, h)$       4 numbers



DOG:  $(x, y, w, h)$   
DOG:  $(x, y, w, h)$       12 numbers  
CAT:  $(x, y, w, h)$



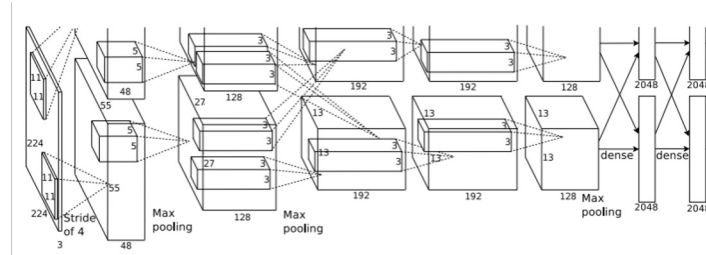
DUCK:  $(x, y, w, h)$       Many numbers!  
DUCK:  $(x, y, w, h)$       Many numbers!

....



# Object Detection: Multiple Objects

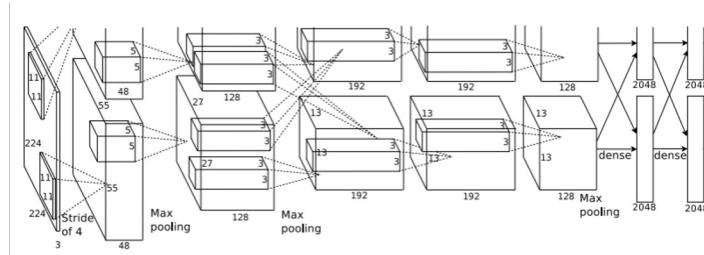
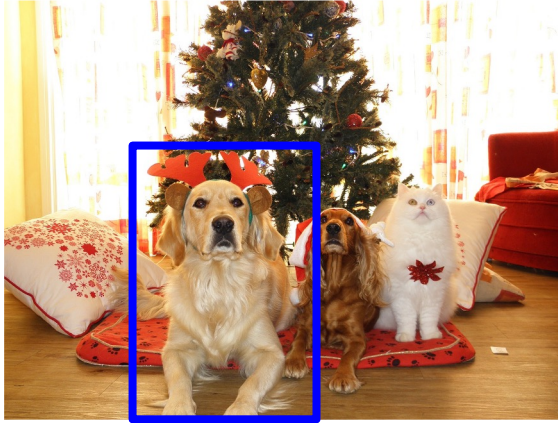
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO  
Cat? NO  
Background? YES

# Object Detection: Multiple Objects

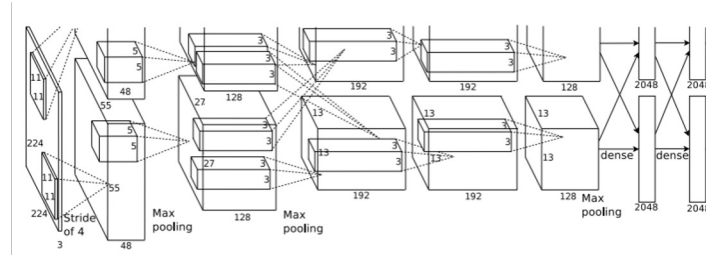
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES  
Cat? NO  
Background? NO

# Object Detection: Multiple Objects

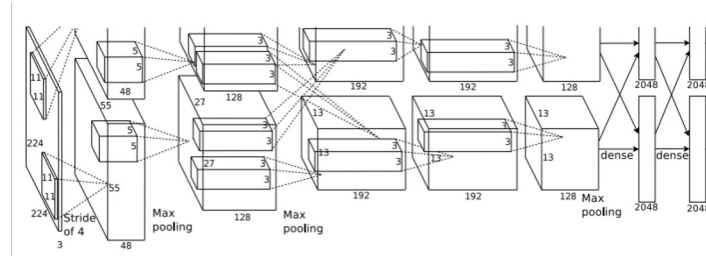
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES  
Cat? NO  
Background? NO

# Object Detection: Multiple Objects

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

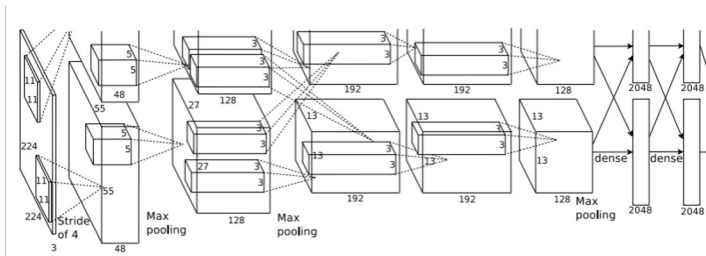
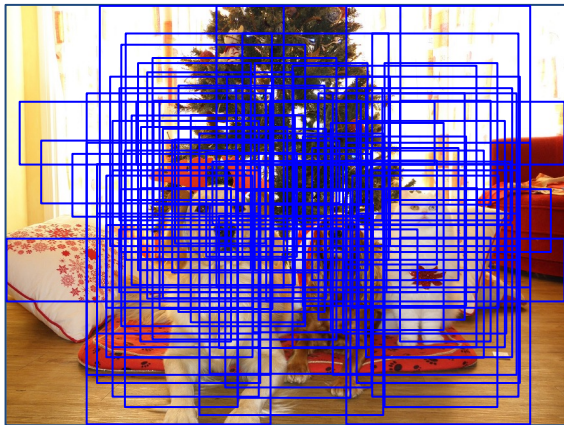


Dog? NO  
Cat? YES  
Background? NO

Q: What's the problem with this approach?

# Object Detection: Multiple Objects

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



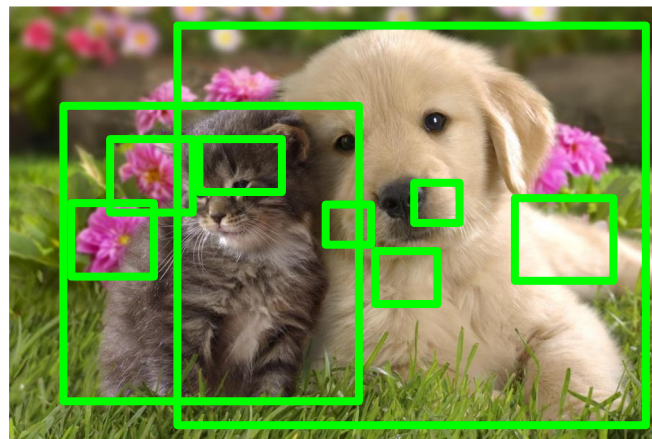
Dog? NO  
Cat? YES  
Background? NO

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

**Need to find promising regions**

# 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



# 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

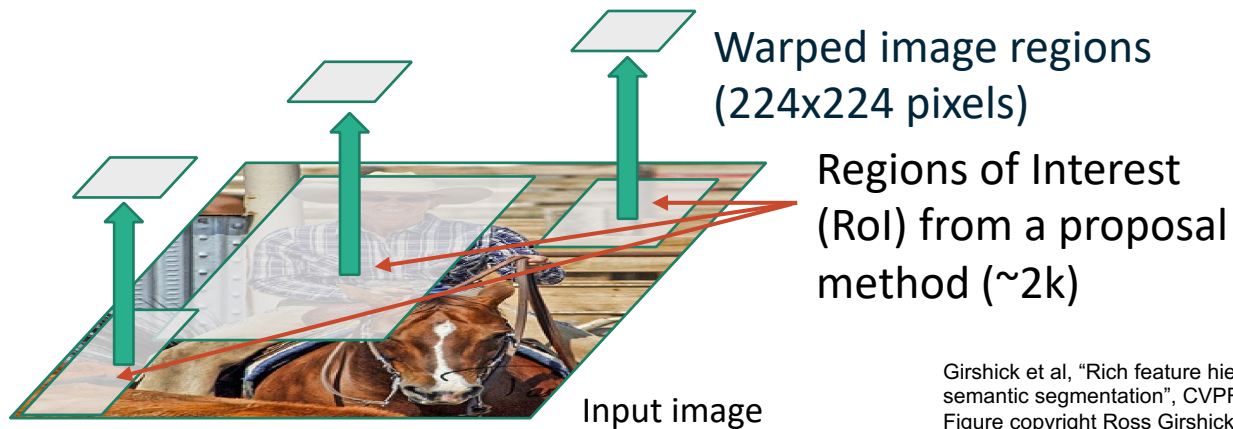


Regions of Interest  
(RoI) from a proposal  
method (~2k)

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

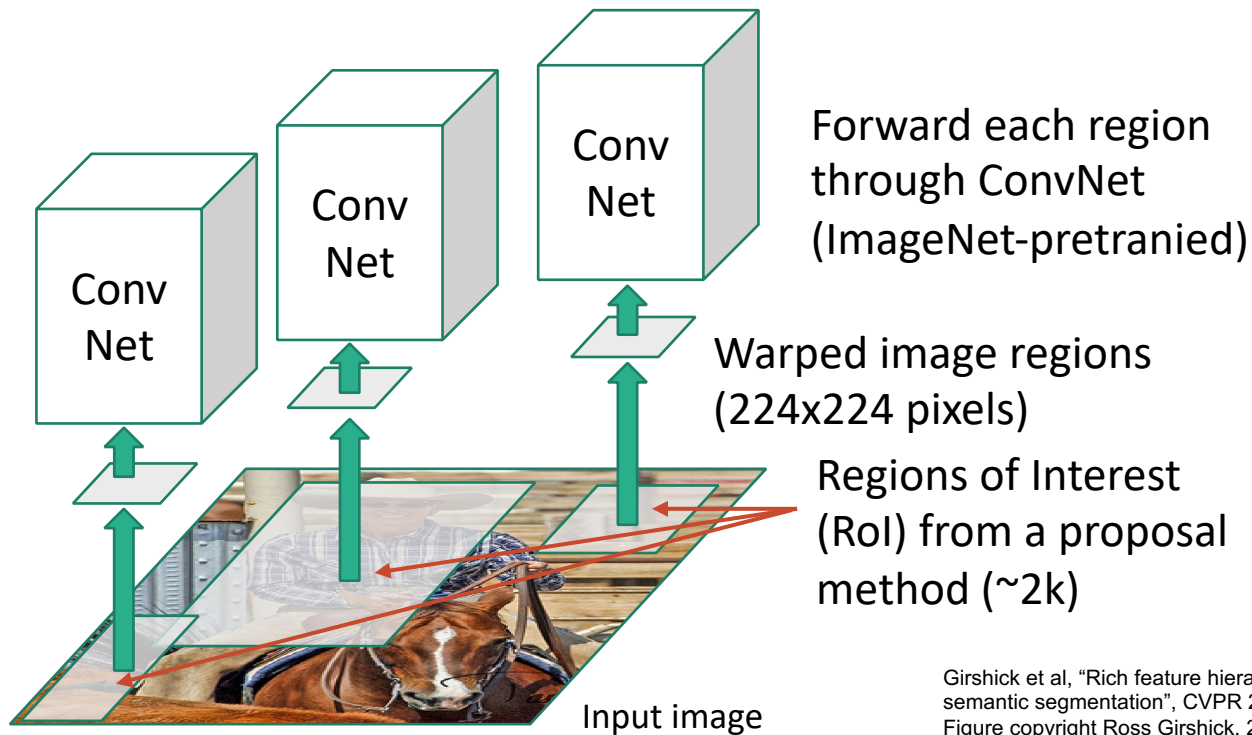


# R-CNN



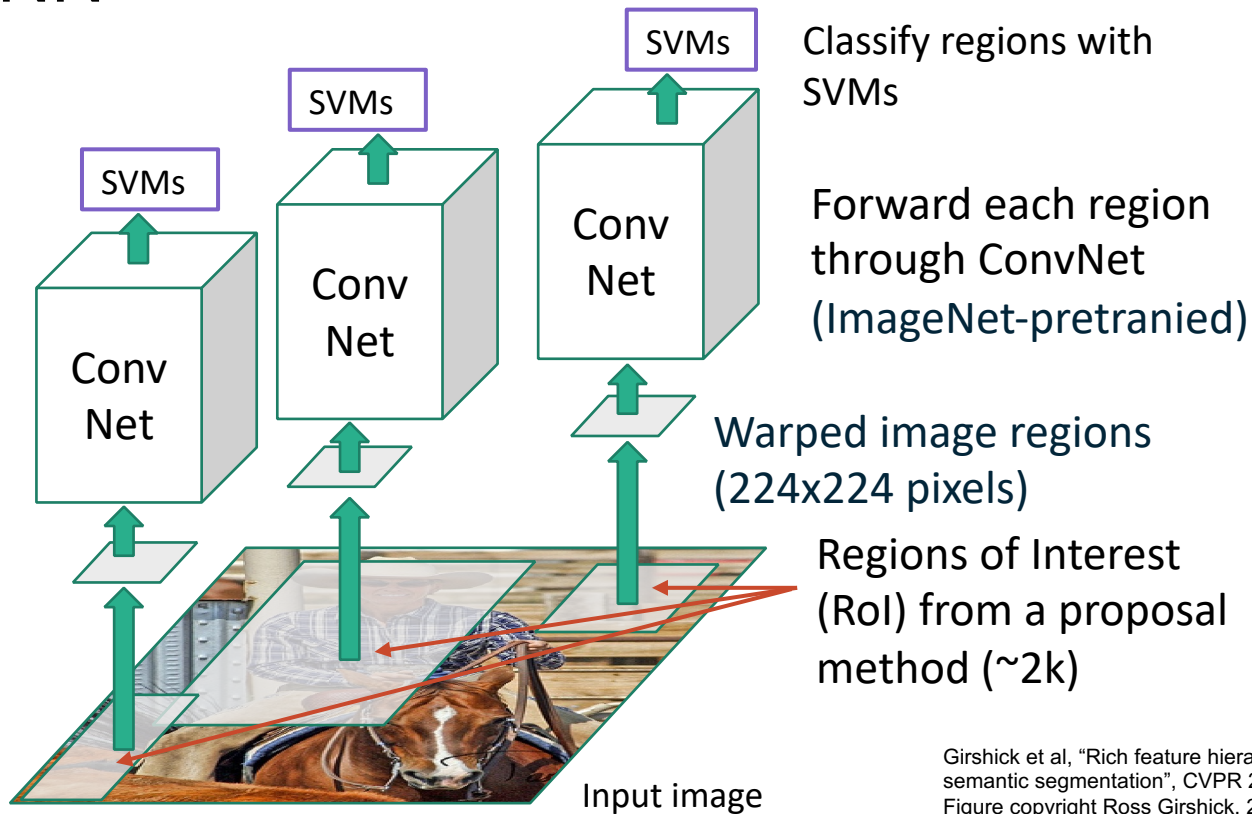
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.  
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

# 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.

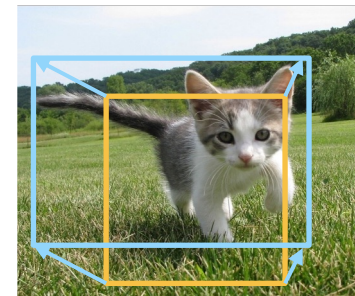
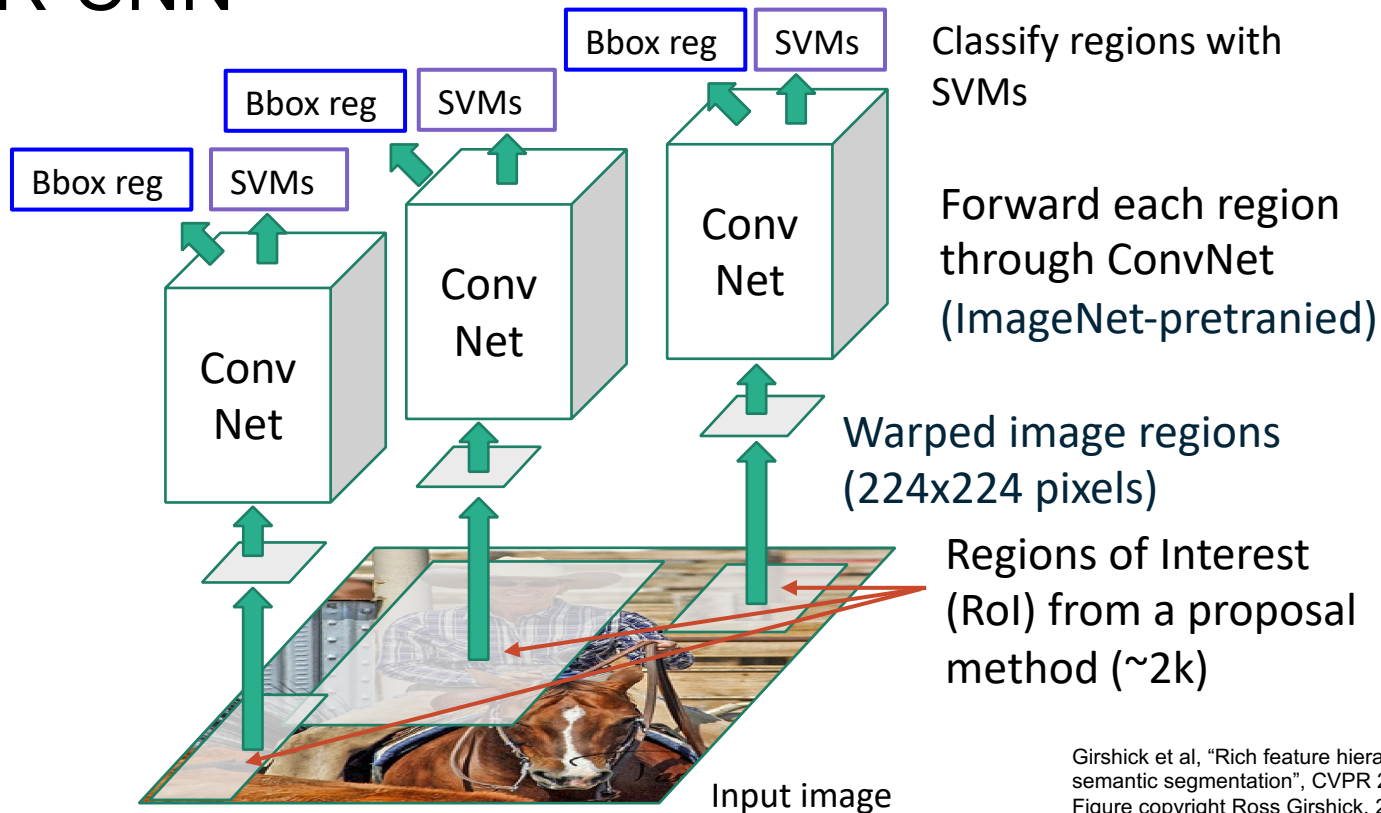
# R-CNN



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

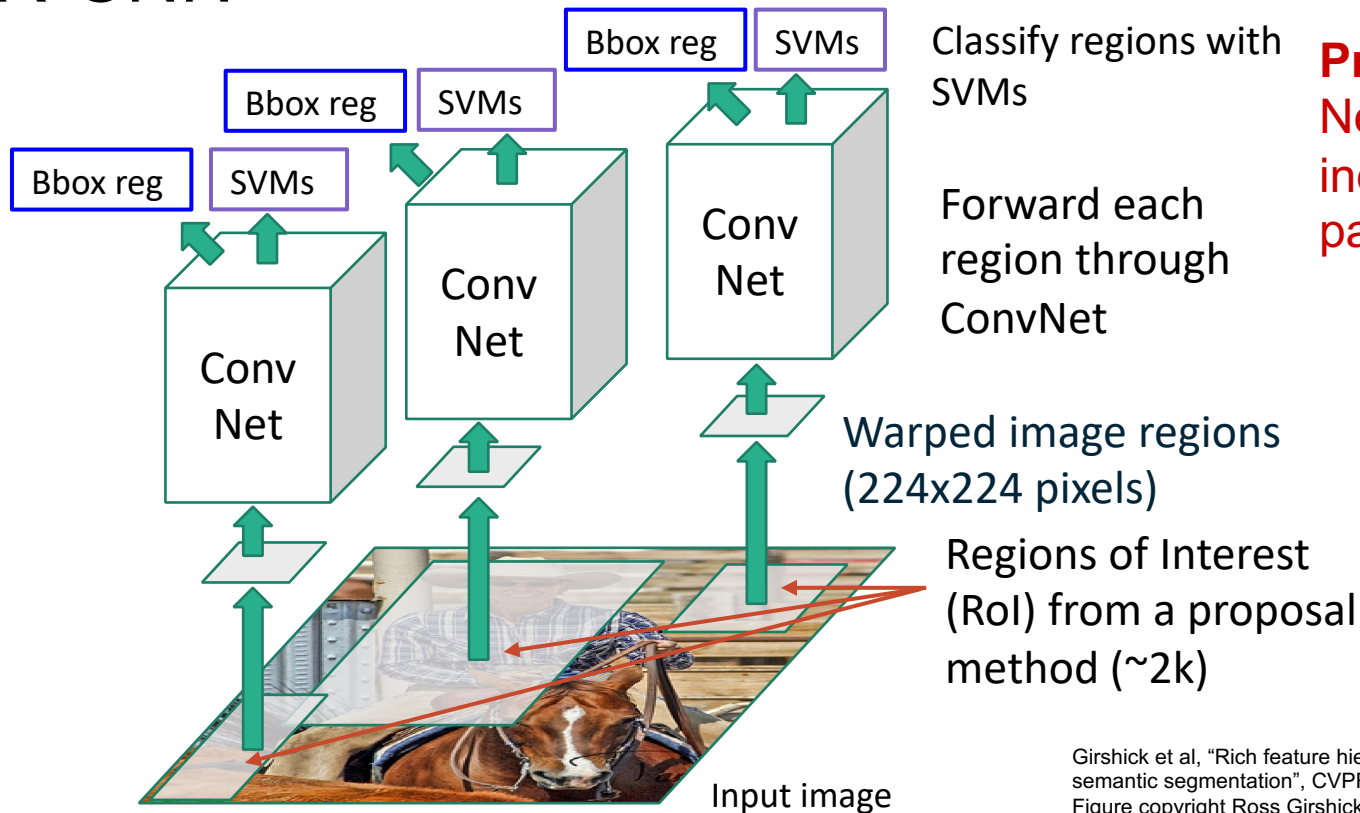
Predict “**corrections**” to the RoI: 4 numbers: (dx, dy, dw, dh)



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

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

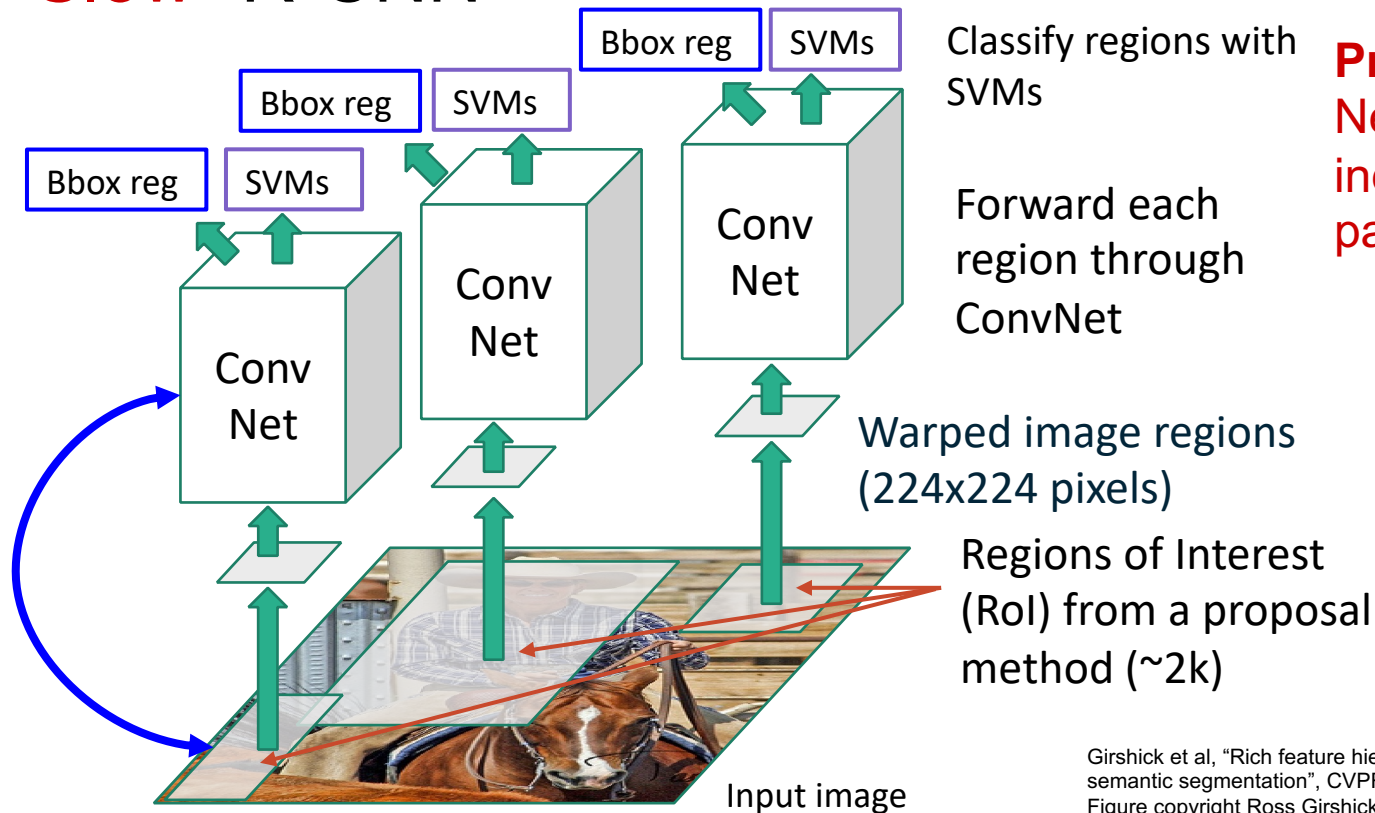


**Problem: Very slow!**  
Need to do ~2k  
independent forward  
passes for each image!

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

# “Slow” R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



**Problem: Very slow!**  
Need to do ~2k  
independent forward  
passes for each image!

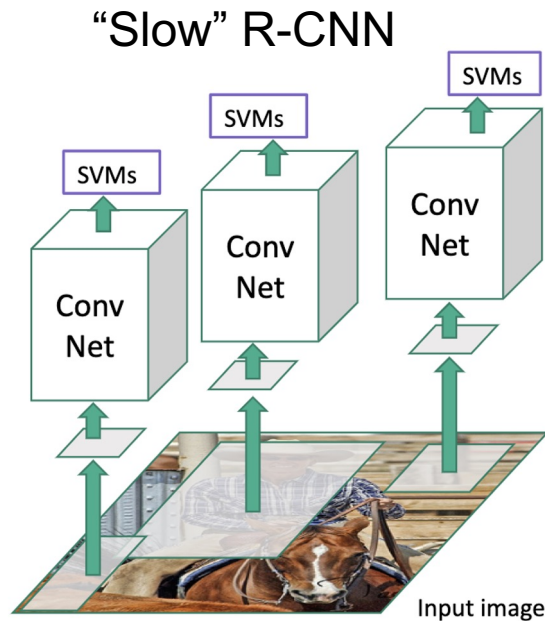
**Idea: Pass the  
image through  
convnet before  
cropping! Crop the  
conv feature instead!**

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

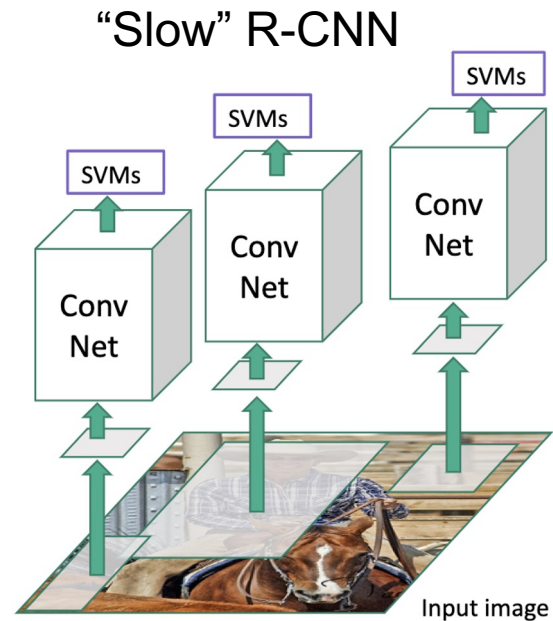
# Fast R-CNN



Input image



# Fast R-CNN

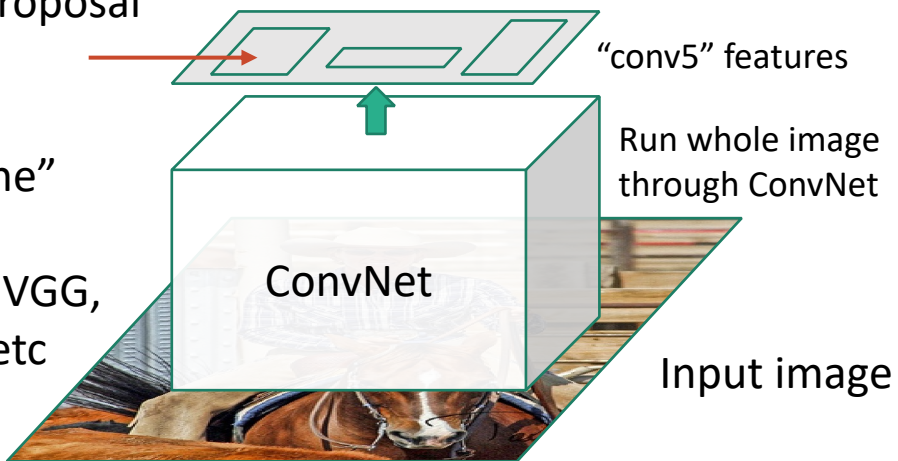




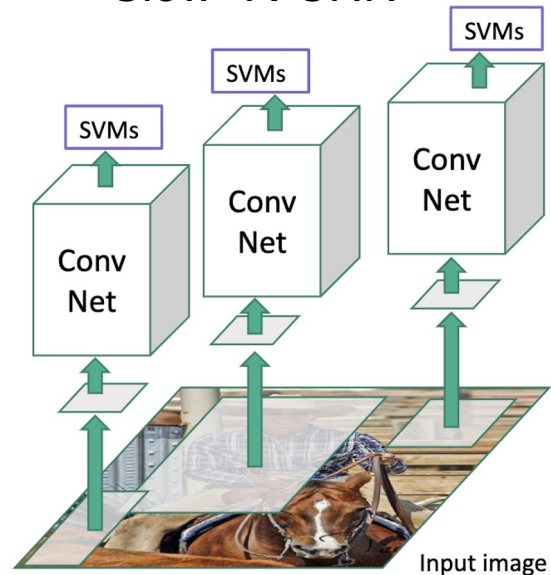
# Fast R-CNN

Regions of Interest (Rois) from a proposal method

“Backbone” network:  
AlexNet, VGG,  
ResNet, etc



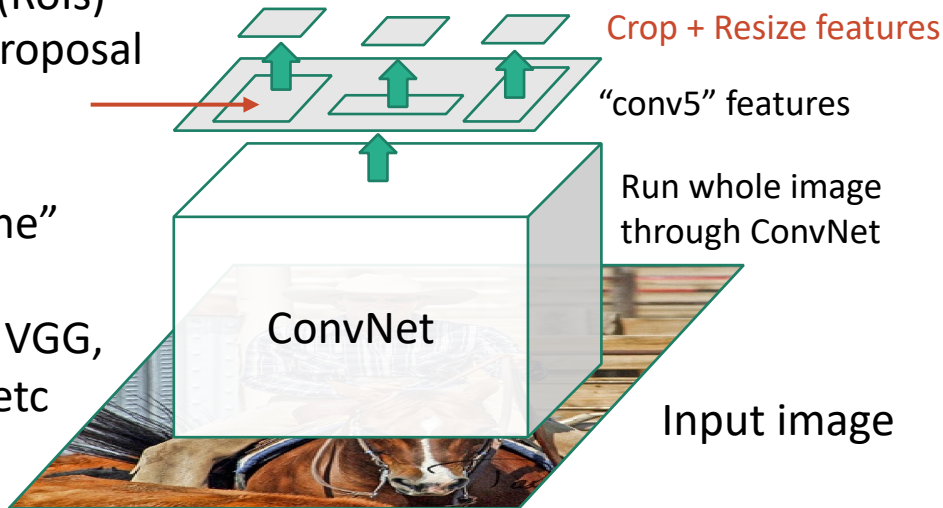
## “Slow” R-CNN



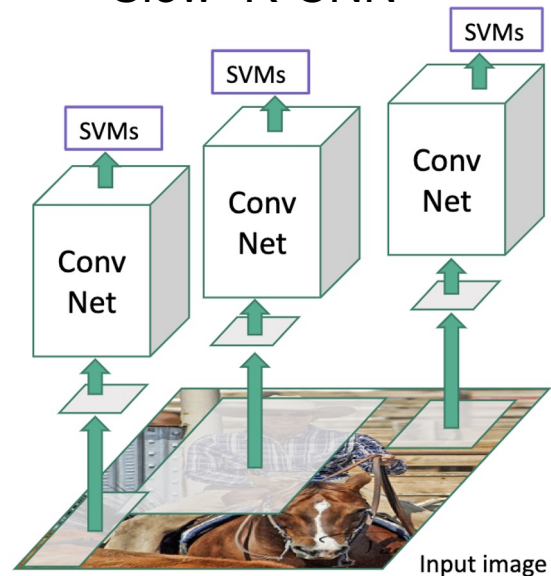
# Fast R-CNN

Regions of Interest (Rois) from a proposal method

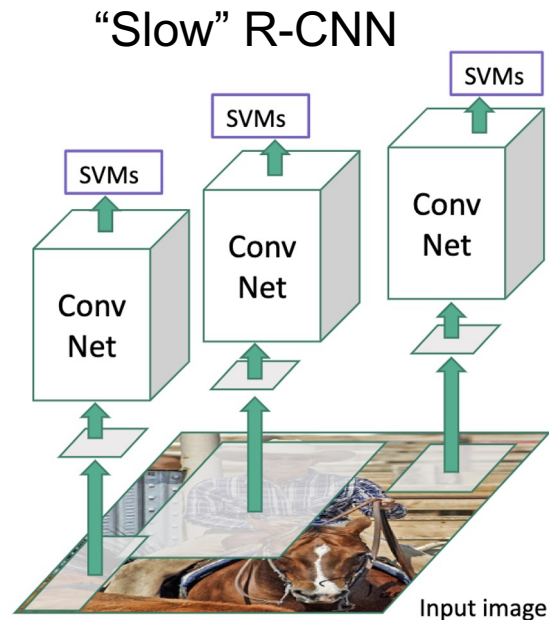
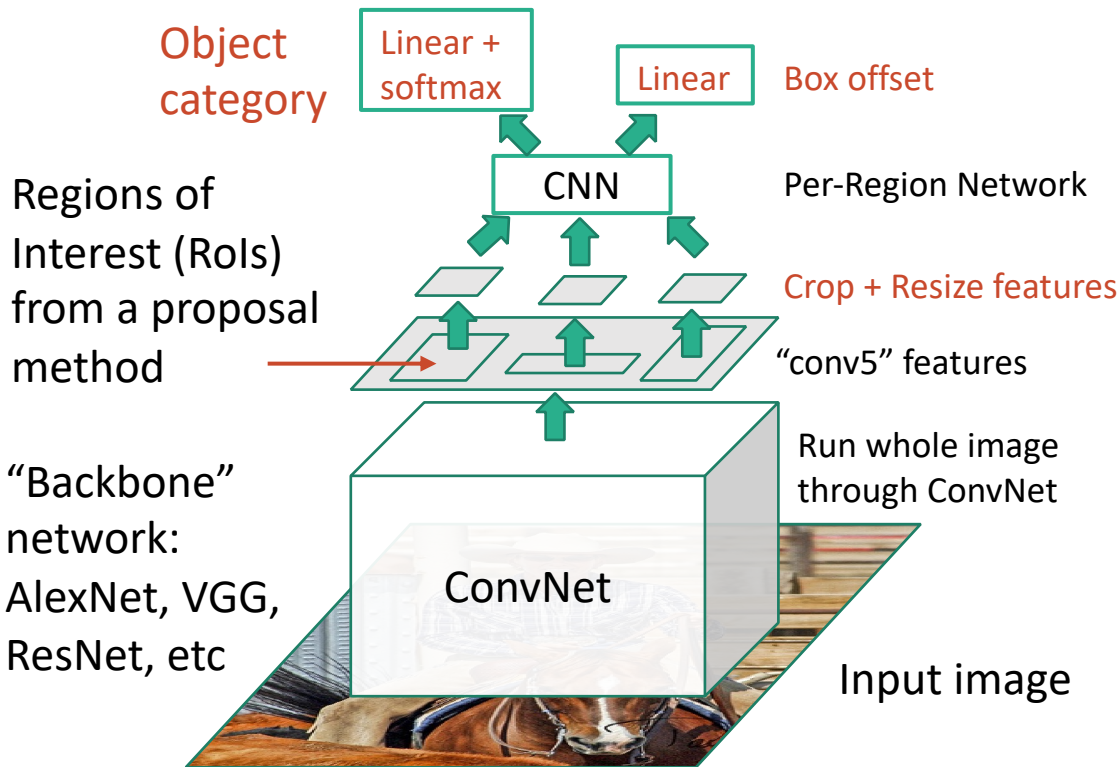
“Backbone” network:  
AlexNet, VGG,  
ResNet, etc



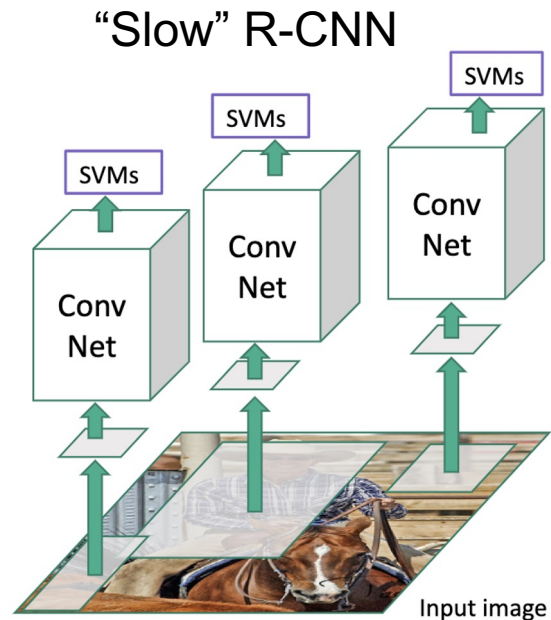
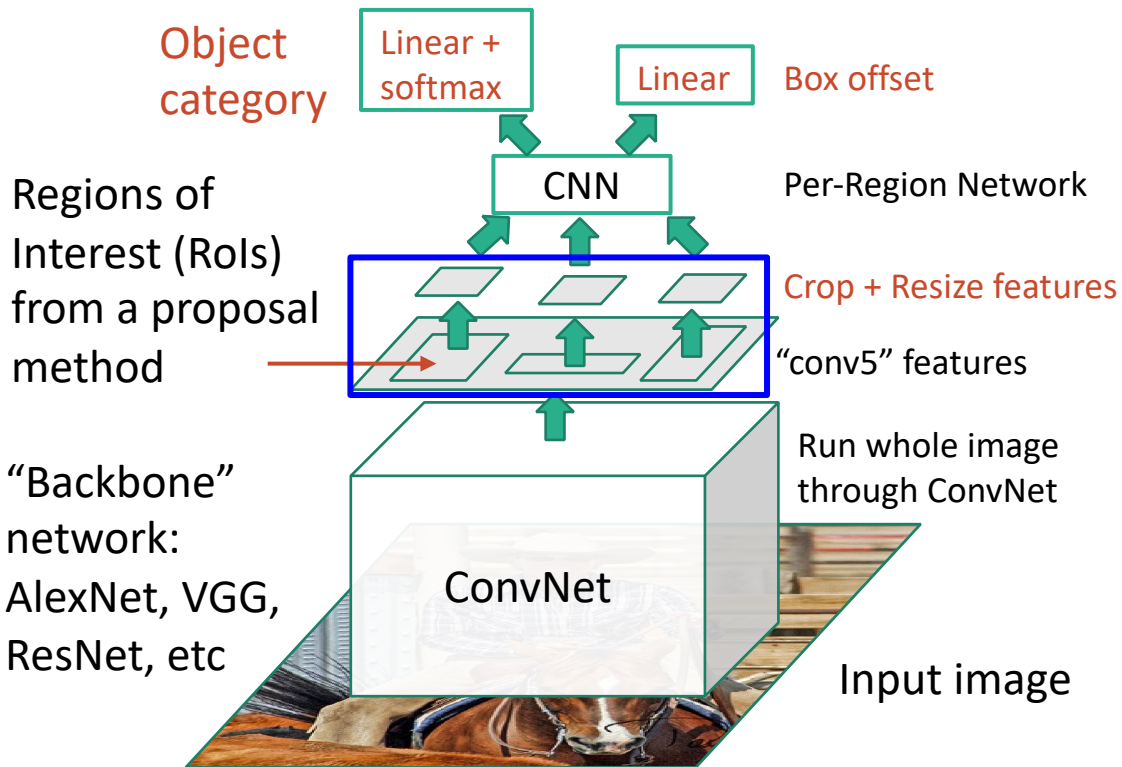
## “Slow” R-CNN



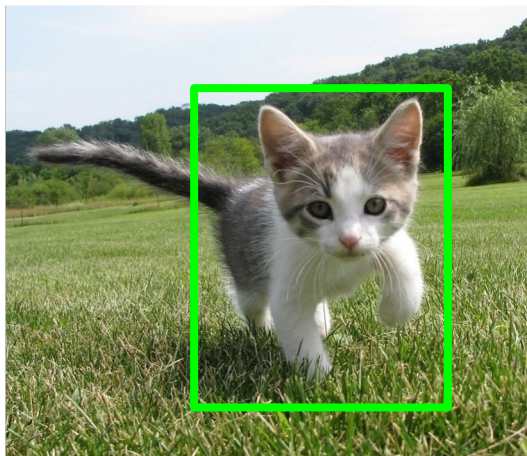
# Fast R-CNN



# Fast R-CNN



# Cropping Features: RoI Pool



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

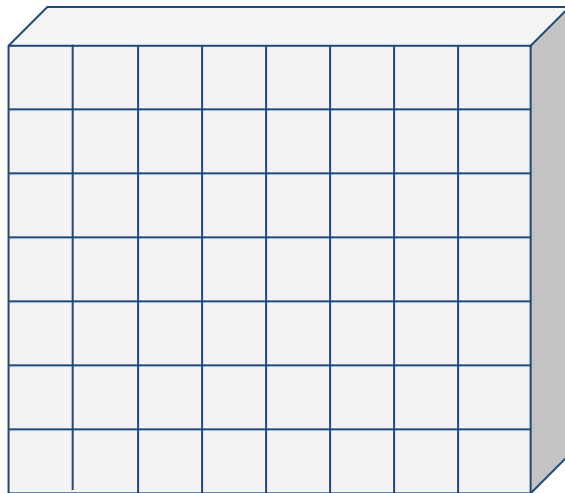
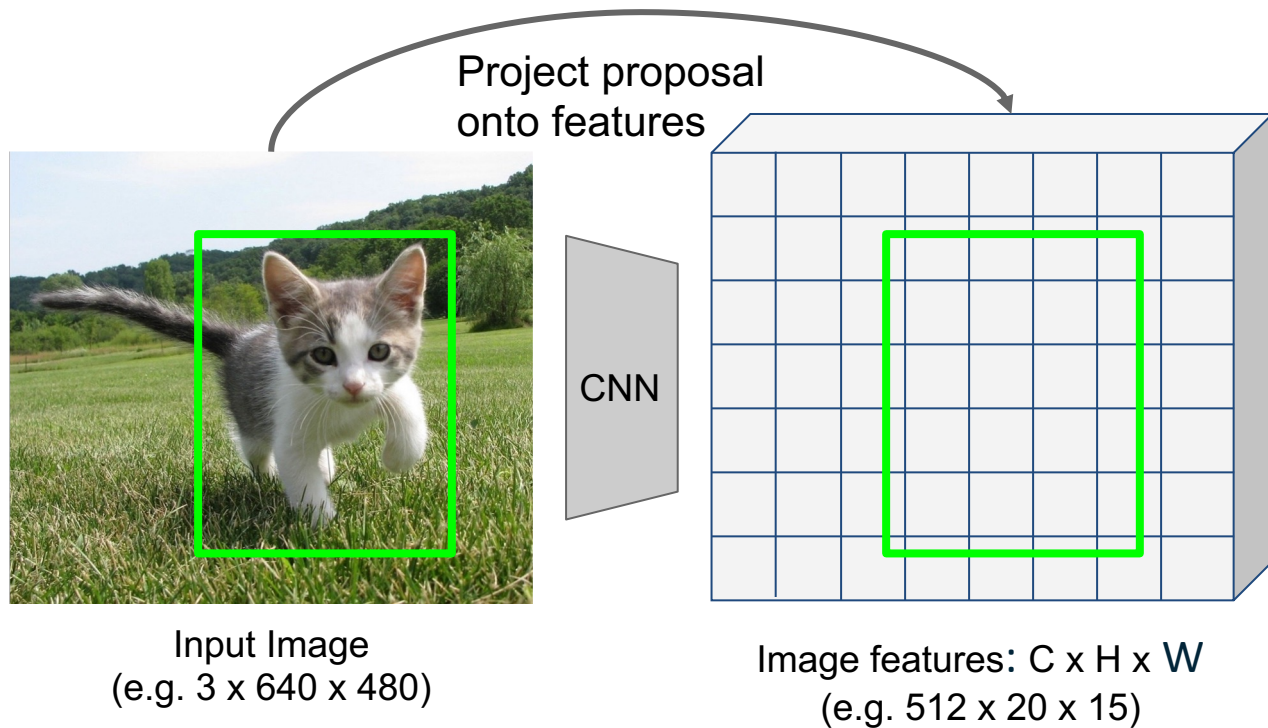


Image features:  $C \times H \times W$   
(e.g. 512 x 20 x 15)

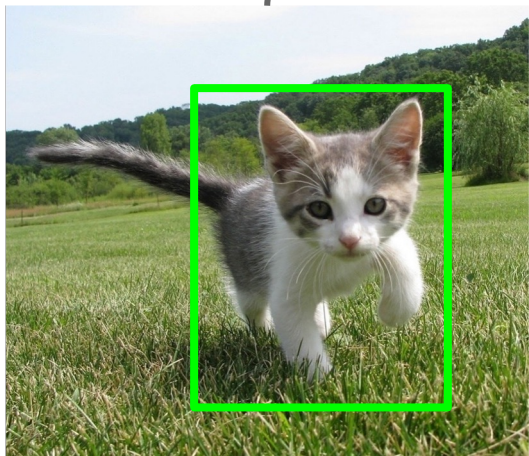
# Cropping Features: RoI Pool



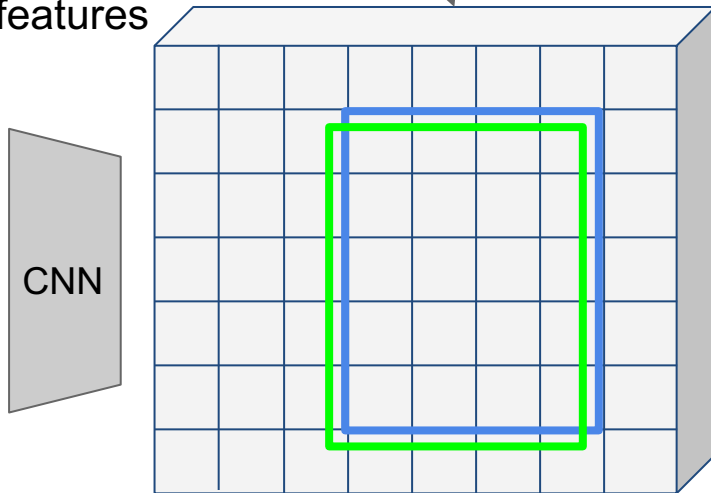
# Cropping Features: RoI Pool

“Snap” to grid cells

Project proposal  
onto features



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



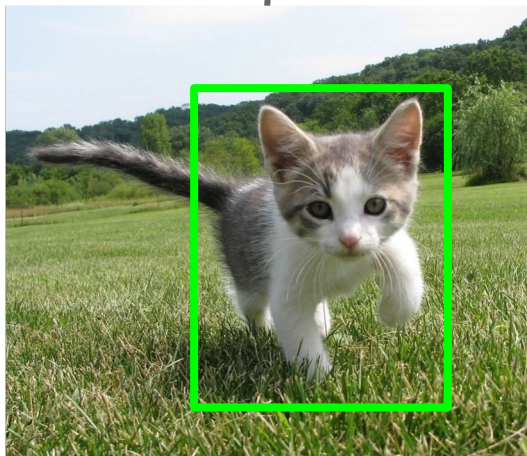
CNN

Image features:  $C \times H \times W$   
(e.g. 512 x 20 x 15)

# Cropping Features: RoI Pool

“Snap” to grid cells

Project proposal  
onto features



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

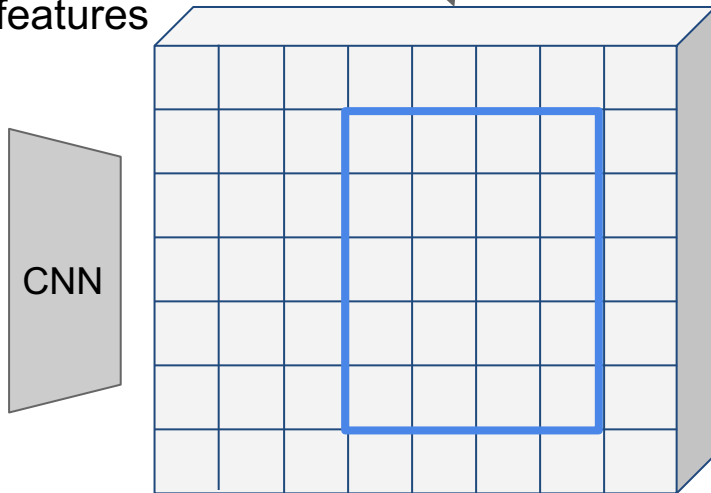
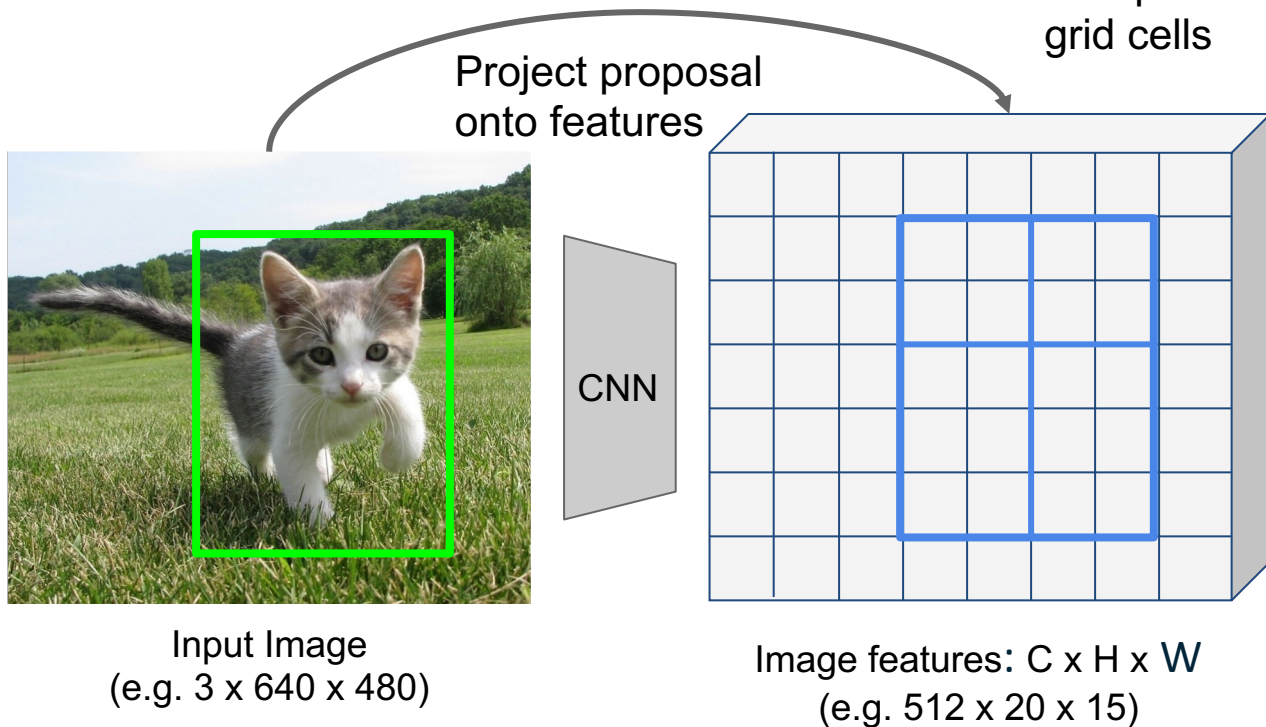


Image features:  $C \times H \times W$   
(e.g. 512 x 20 x 15)

Q: how do we resize the 512 x 20 x 15 region to, e.g., a 512 x 2 x 2 tensor?.



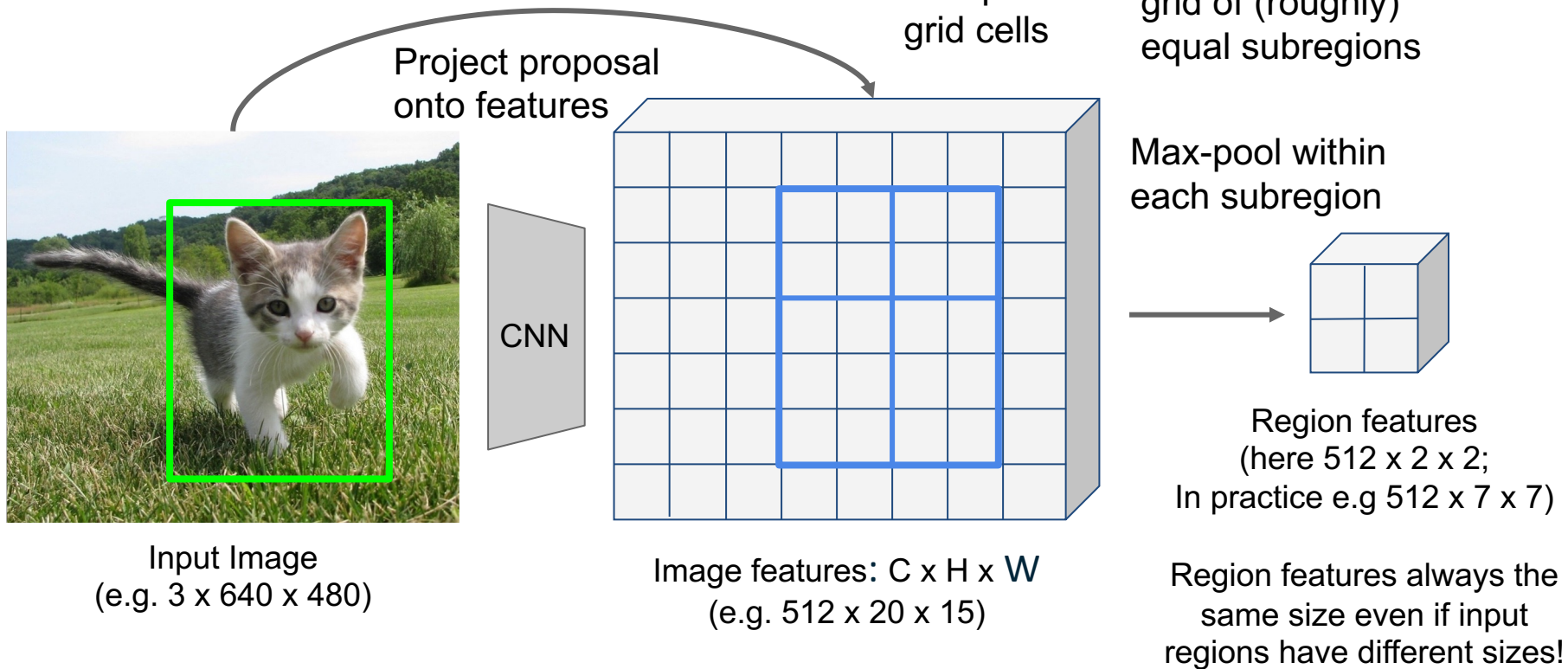
# Cropping Features: RoI Pool



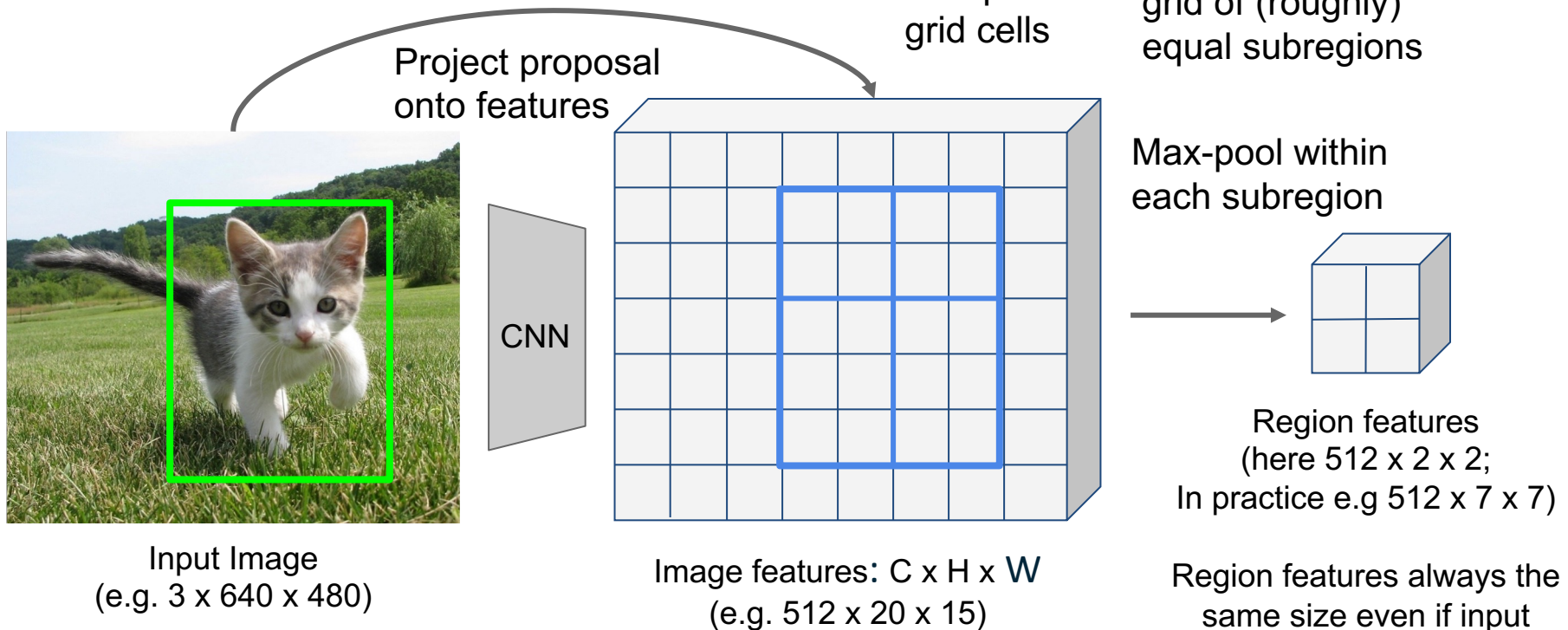
Divide into 2x2  
grid of (roughly)  
equal subregions

Q: how do we resize the 512  
x 20 x 15 region to, e.g., a  
512 x 2 x 2 tensor?.

# Cropping Features: RoI Pool

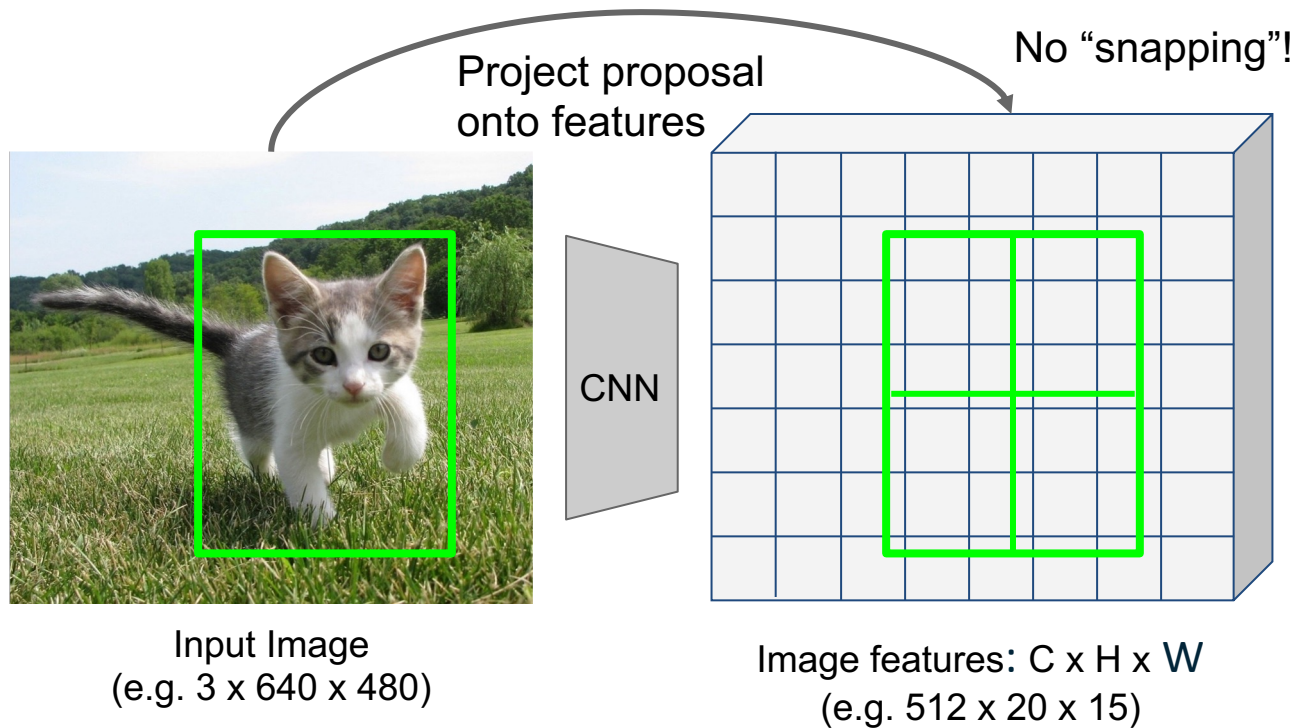


# Cropping Features: RoI Pool



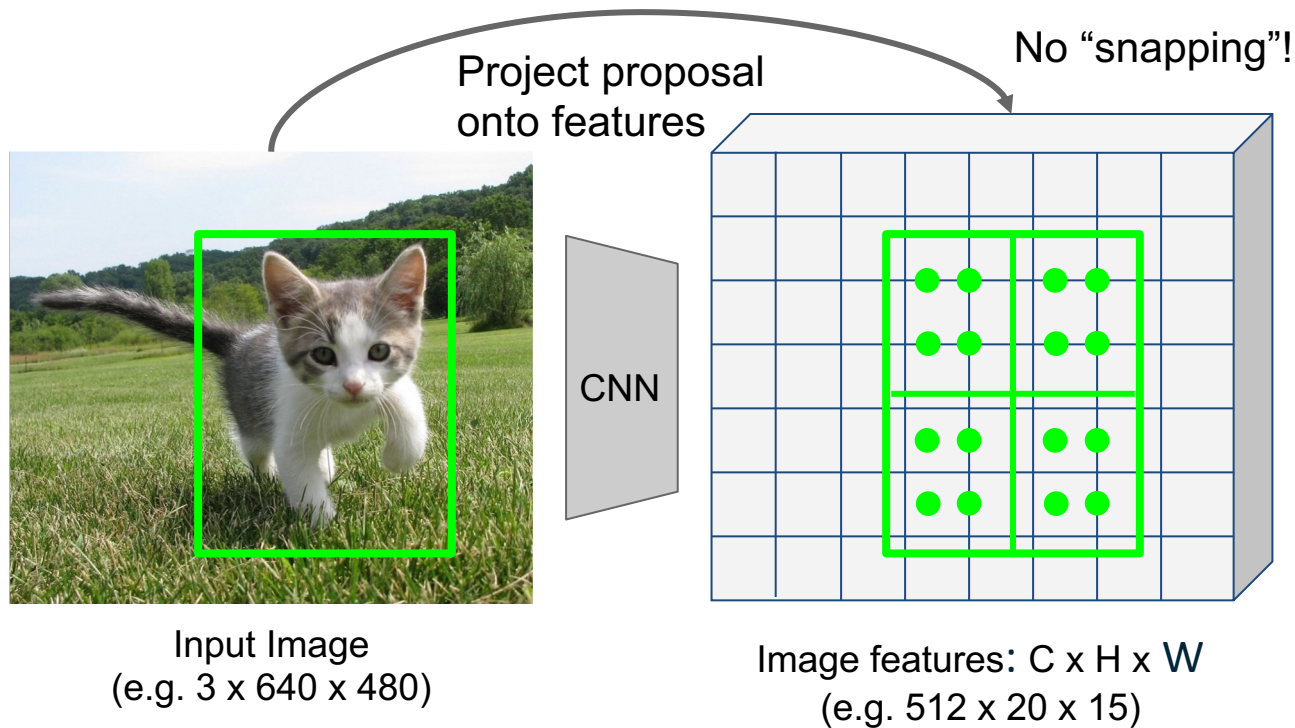
**Problem: Region features slightly misaligned**

# Cropping Features: RoI Align

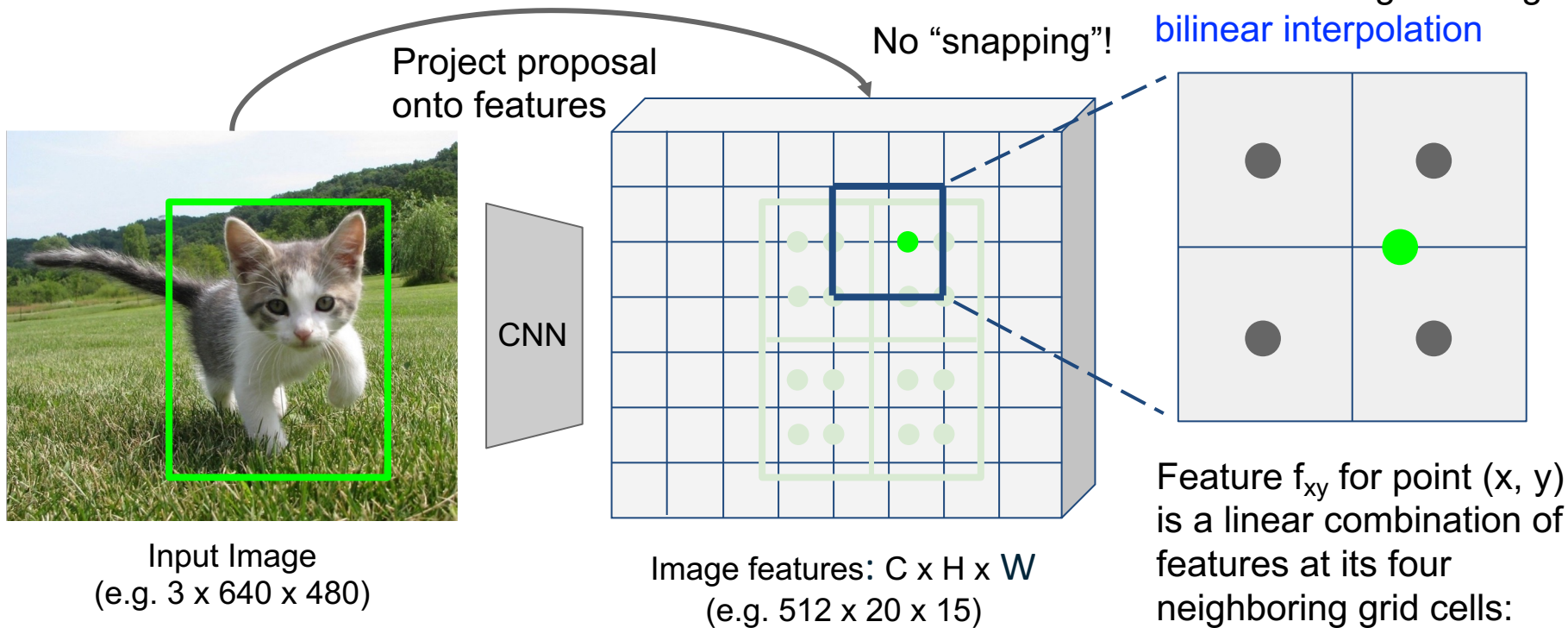


# Cropping Features: RoI Align

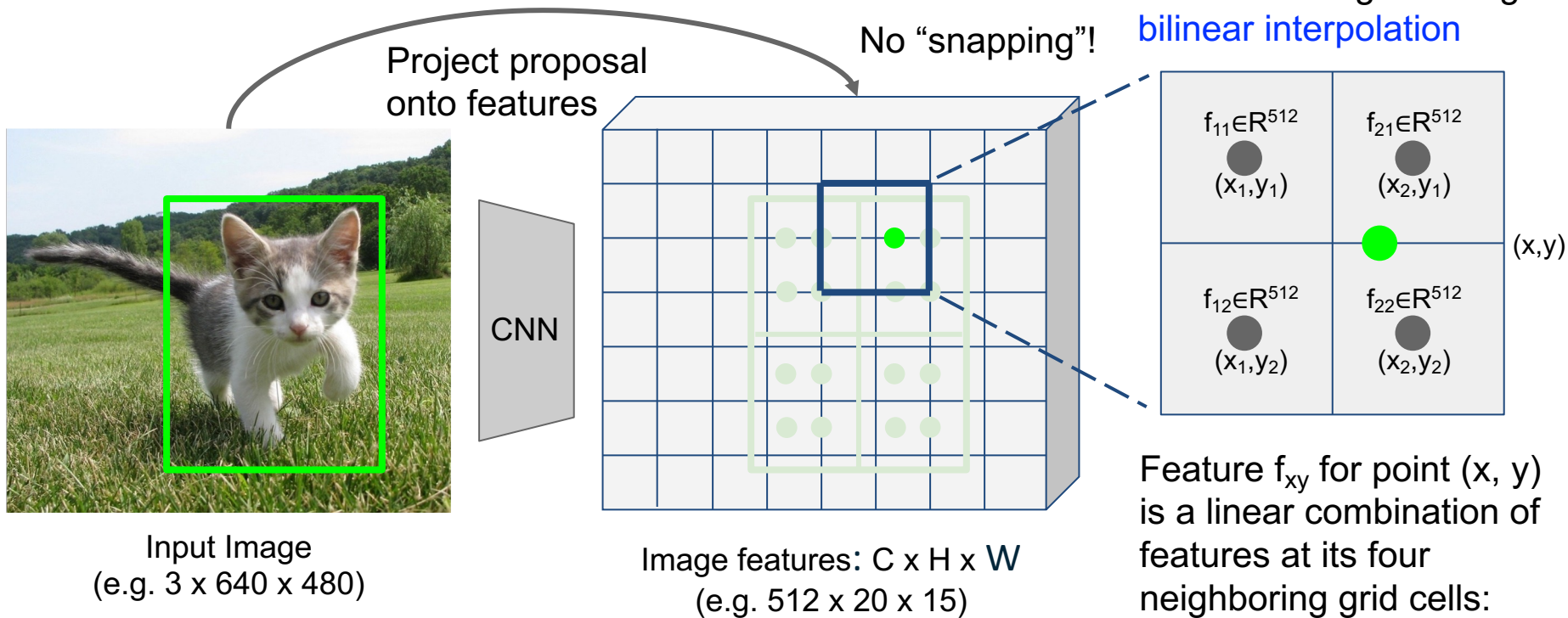
Sample at regular points in each subregion using **bilinear interpolation**



# Cropping Features: RoI Align

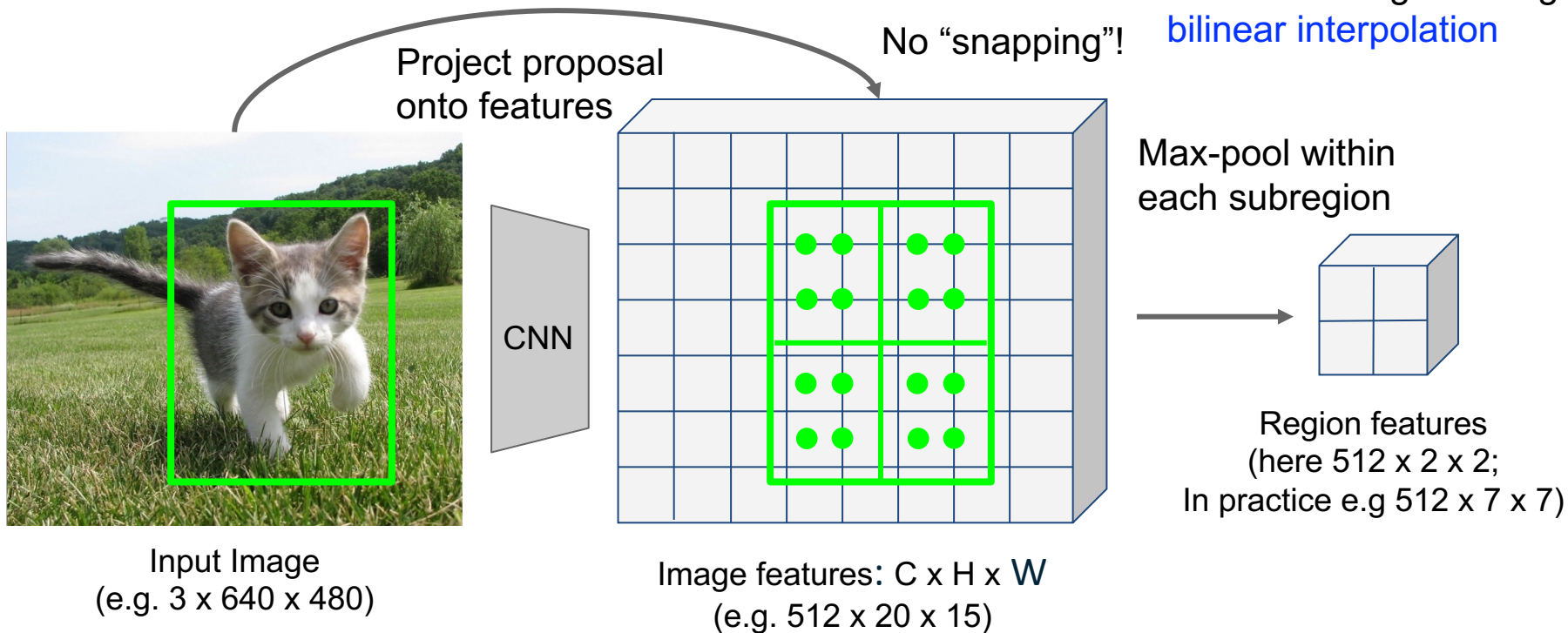


# Cropping Features: RoI Align



$$f_{xy} = \sum_{i,j=1}^2 f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

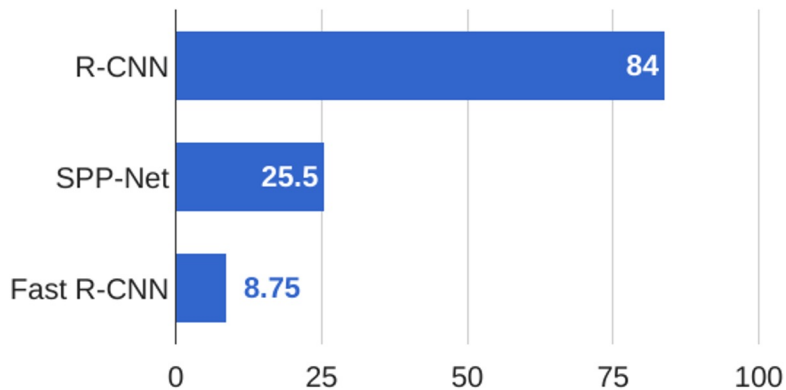
# Cropping Features: RoI Align



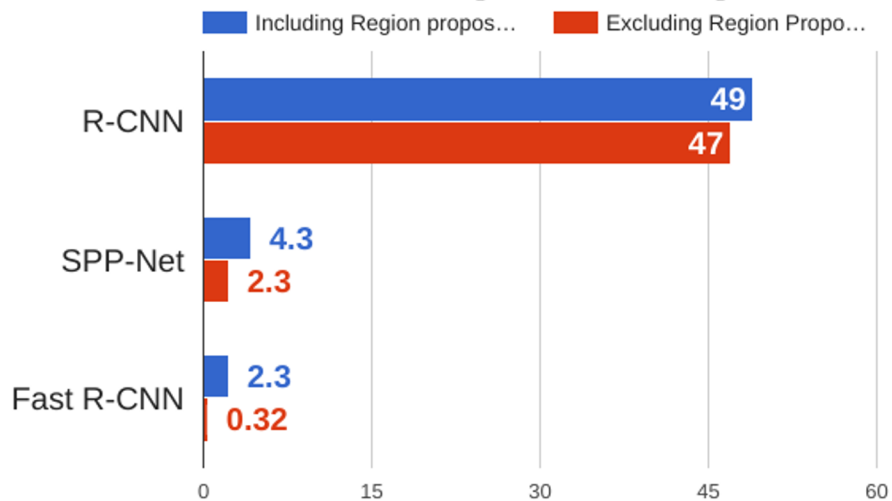


# R-CNN vs Fast R-CNN

## Training time (Hours)



## Test time (seconds)



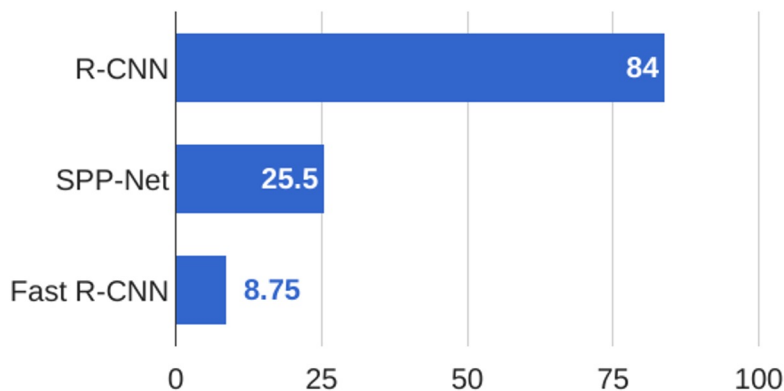
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

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

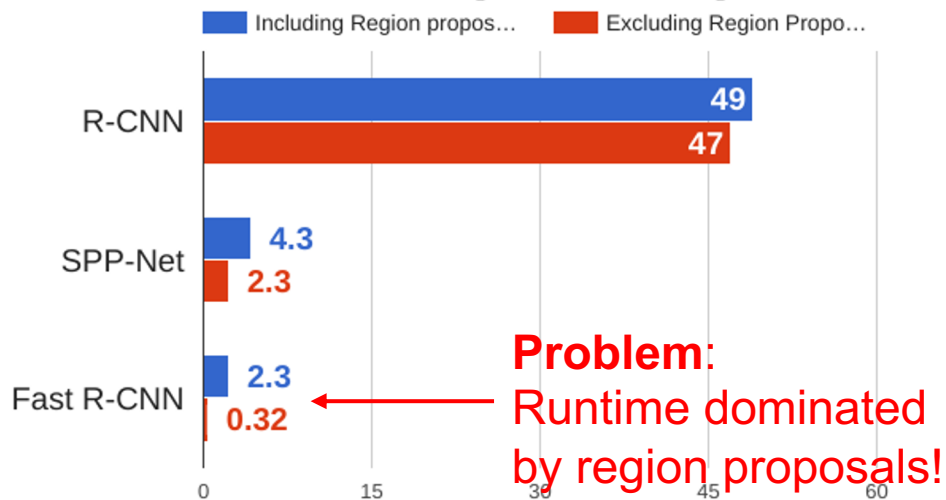
Girshick, "Fast R-CNN", ICCV 2015

# R-CNN vs 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

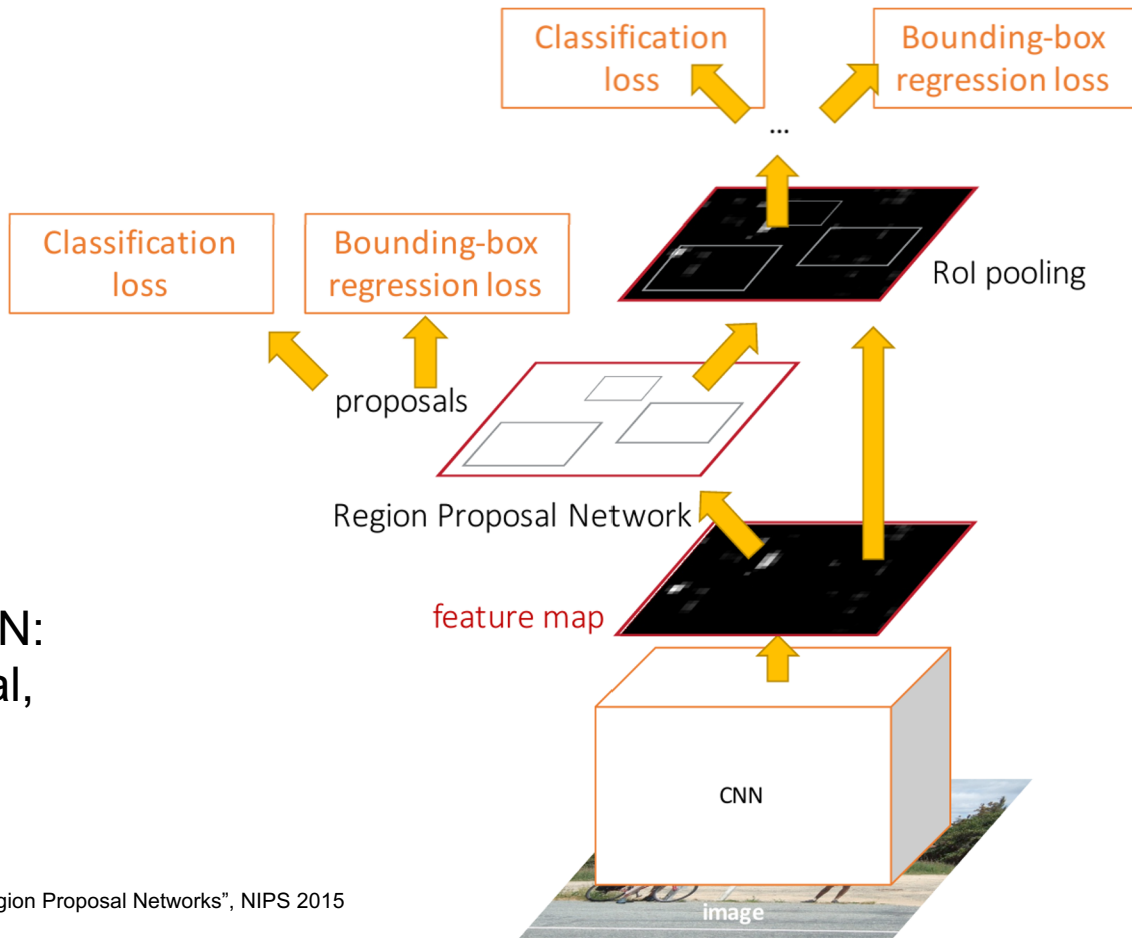
Girshick, "Fast R-CNN", ICCV 2015

# Faster R-CNN:

Make CNN do proposals!

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

Otherwise same as Fast R-CNN:  
Crop features for each proposal,  
classify each one



# Region Proposal Network



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

CNN

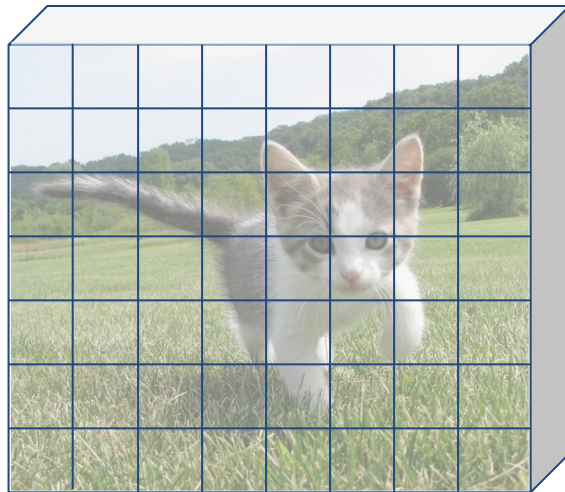


Image features  
(e.g. 512 x 20 x 15)

# Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map



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

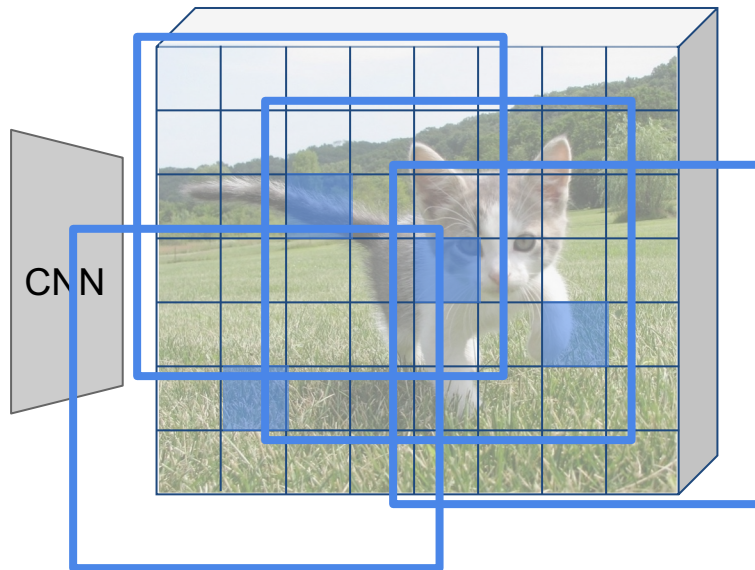


Image features  
(e.g. 512)

# Region Proposal Network

Example: 20 x 15 **anchor box** uniformly sampled on the feature map



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

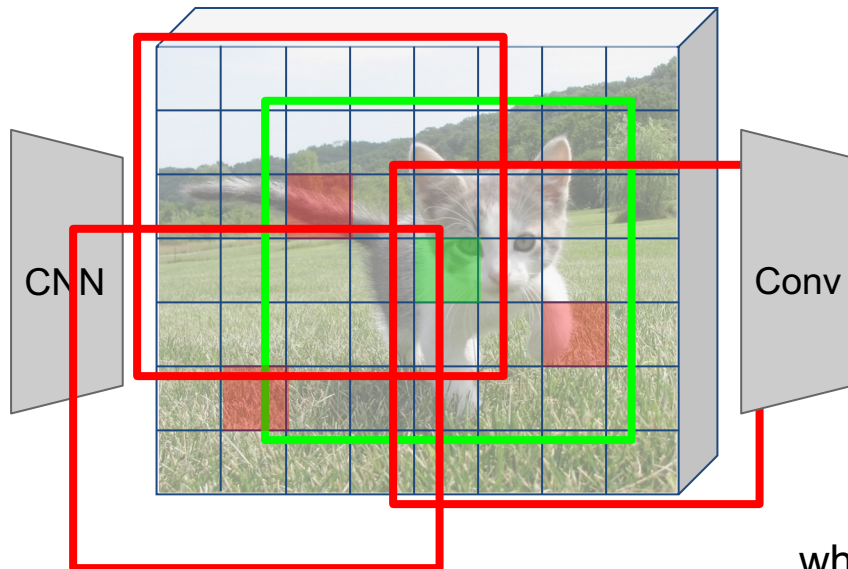


Image features  
(e.g. 512)

Anchor is an object?  
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

# Region Proposal Network



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

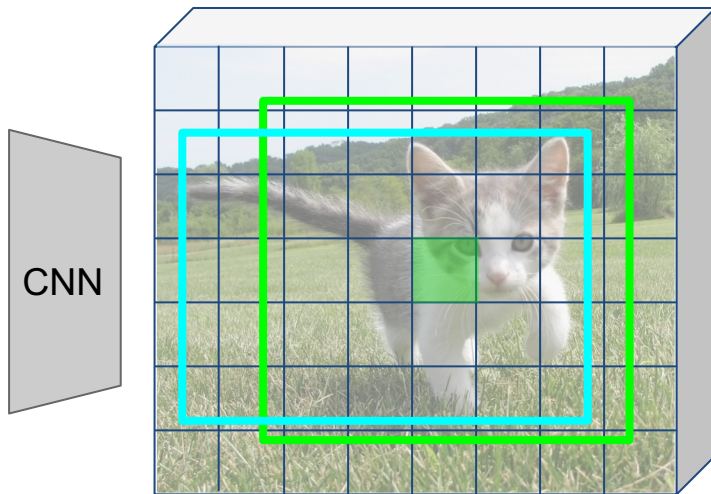


Image features  
(e.g. 512)

Example: 20 x 15  
**anchor box** uniformly  
sampled on the feature  
map



Conv

- Anchor is an object?  
1 x 20 x 15
- Box corrections  
4 x 20 x 15

For positive boxes, also predict  
a corrections from the anchor  
to the ground-truth box (regress  
4 numbers per pixel)

# Region Proposal Network

In practice use  $K$  different anchor boxes of different size / scale at each point



Input Image  
(e.g.  $3 \times 640 \times 480$ )

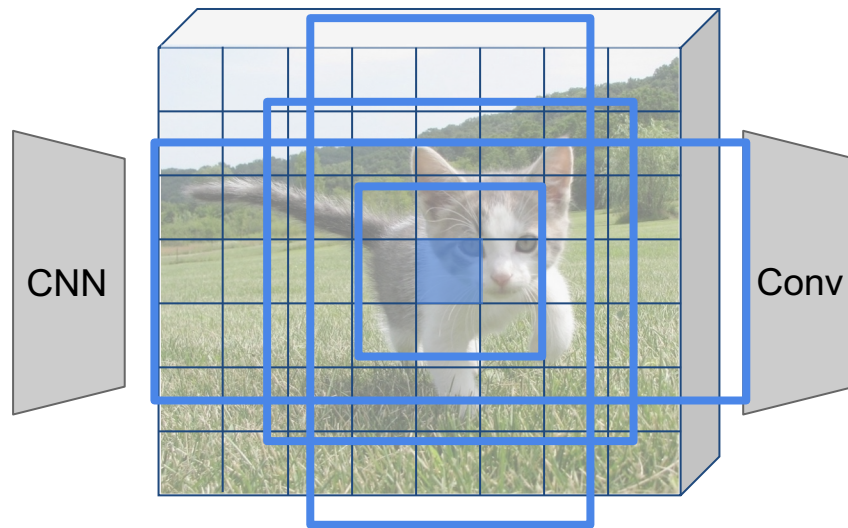


Image features  
(e.g. 512)

Anchor is an object?  
 $\mathbf{K} \times 20 \times 15$

Box transforms  
 $\mathbf{4K} \times 20 \times 15$



# Region Proposal Network

In practice use  $K$  different anchor boxes of different size / scale at each point



Input Image  
(e.g.  $3 \times 640 \times 480$ )

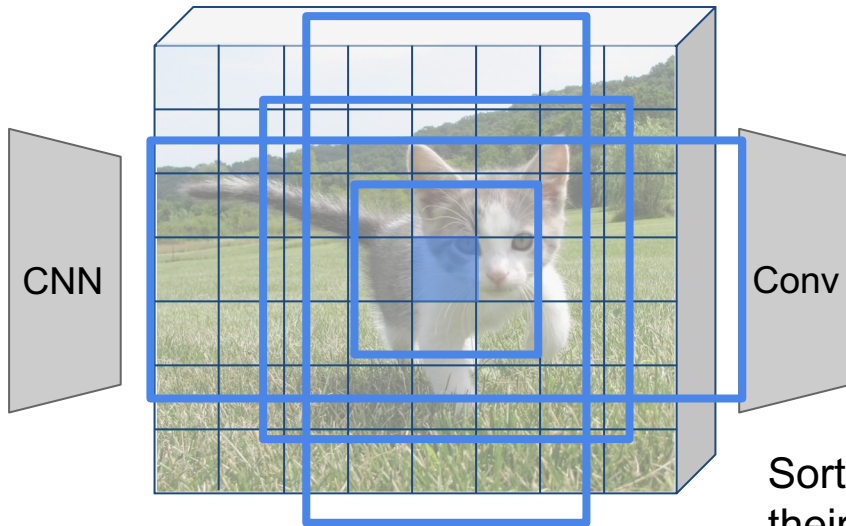


Image features  
(e.g. 512)

Anchor is an object?  
 $K \times 20 \times 15$

Box transforms  
 $4K \times 20 \times 15$

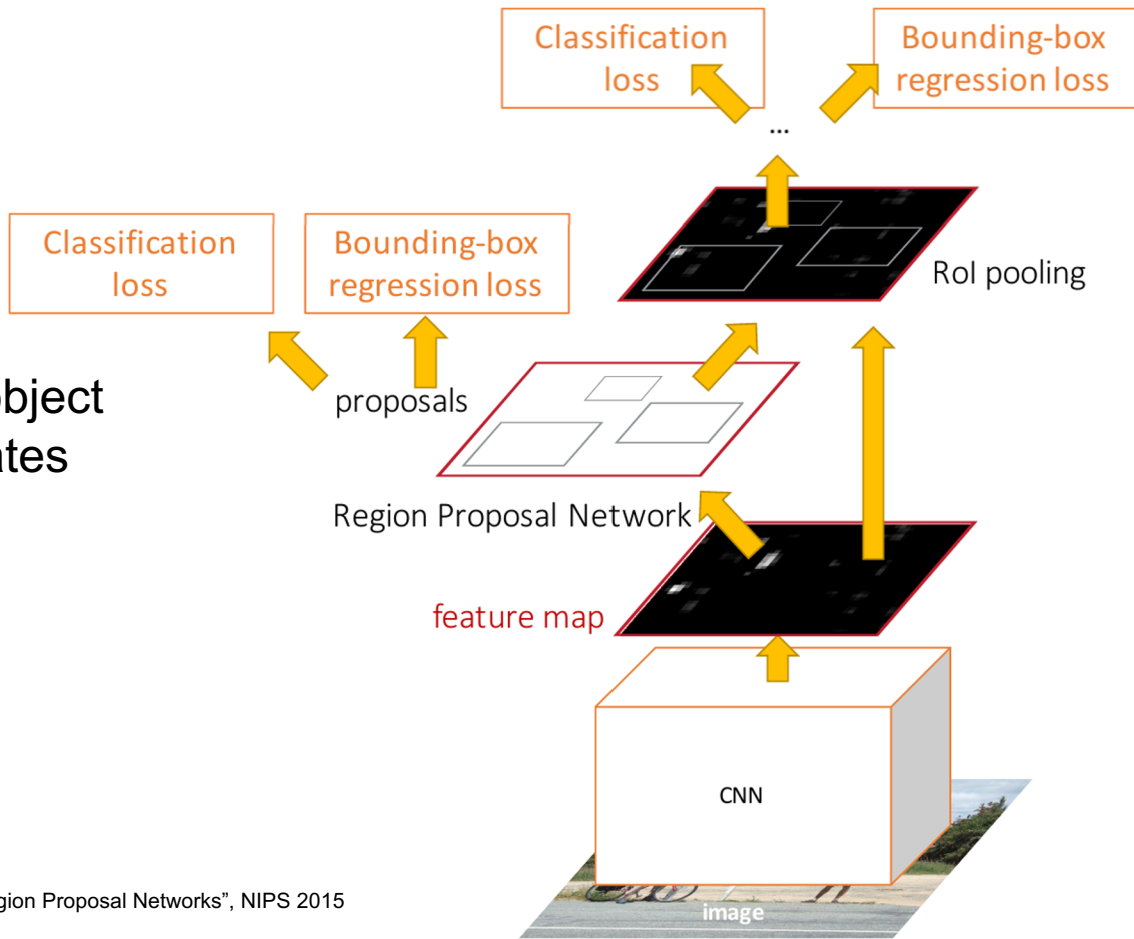
Sort the  $K \times 20 \times 15$  boxes by their “objectness” score, take top  $\sim 300$  as our proposals

# Faster R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

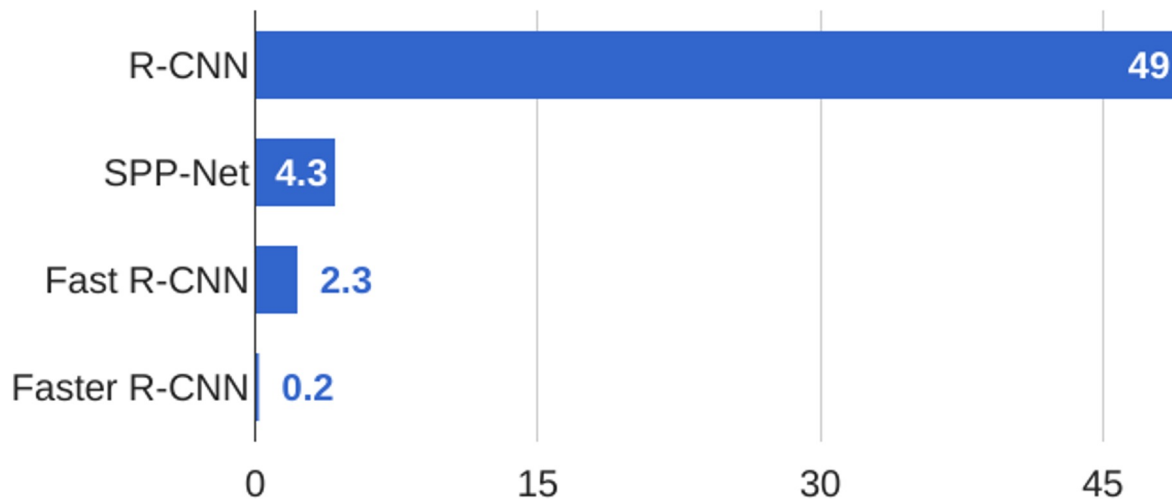
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



# Faster R-CNN:

Make CNN do proposals!

## R-CNN Test-Time Speed

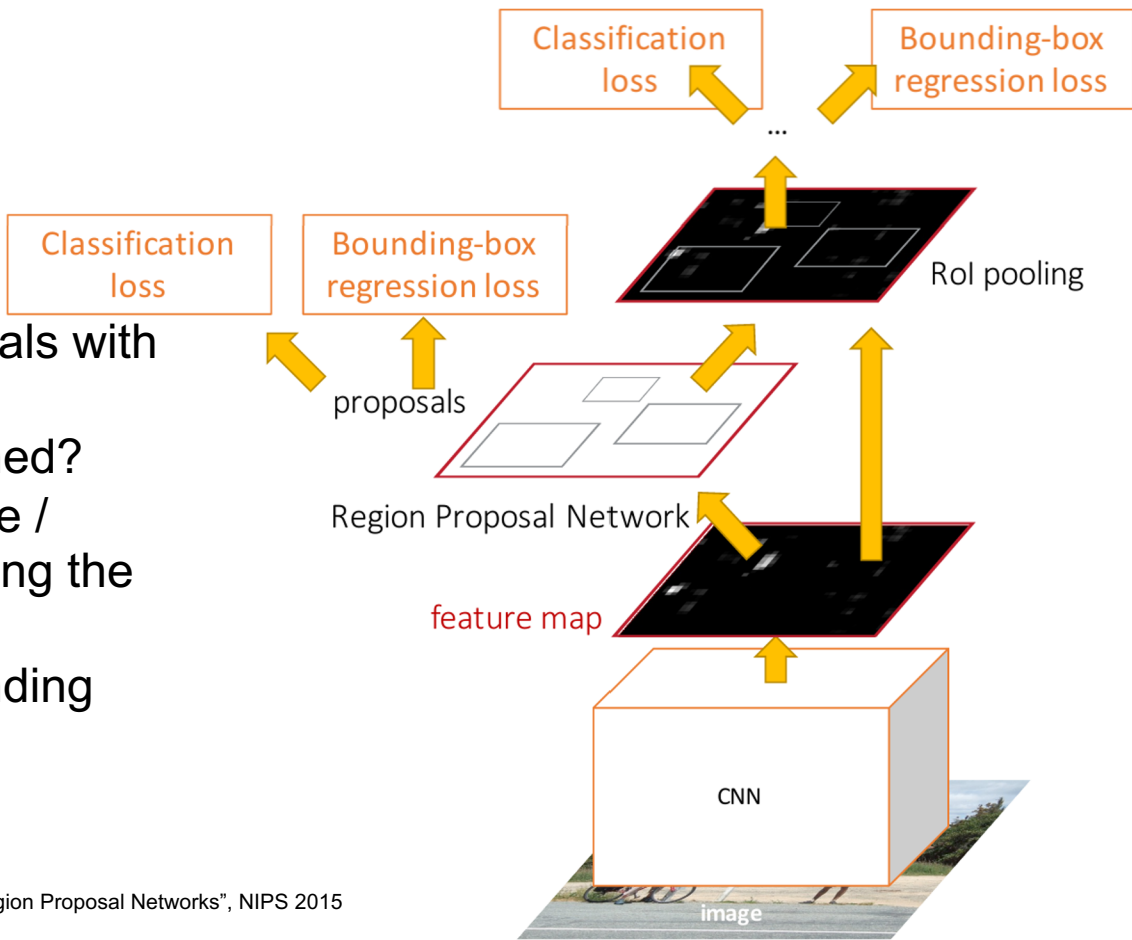


# Faster R-CNN:

Make CNN do proposals!

Glossing over many details:

- Ignore overlapping proposals with **non-max suppression**
- How are anchors determined?
- How do we sample positive / negative samples for training the RPN?
- How to parameterize bounding box regression?



# Faster R-CNN:

Make CNN do proposals!

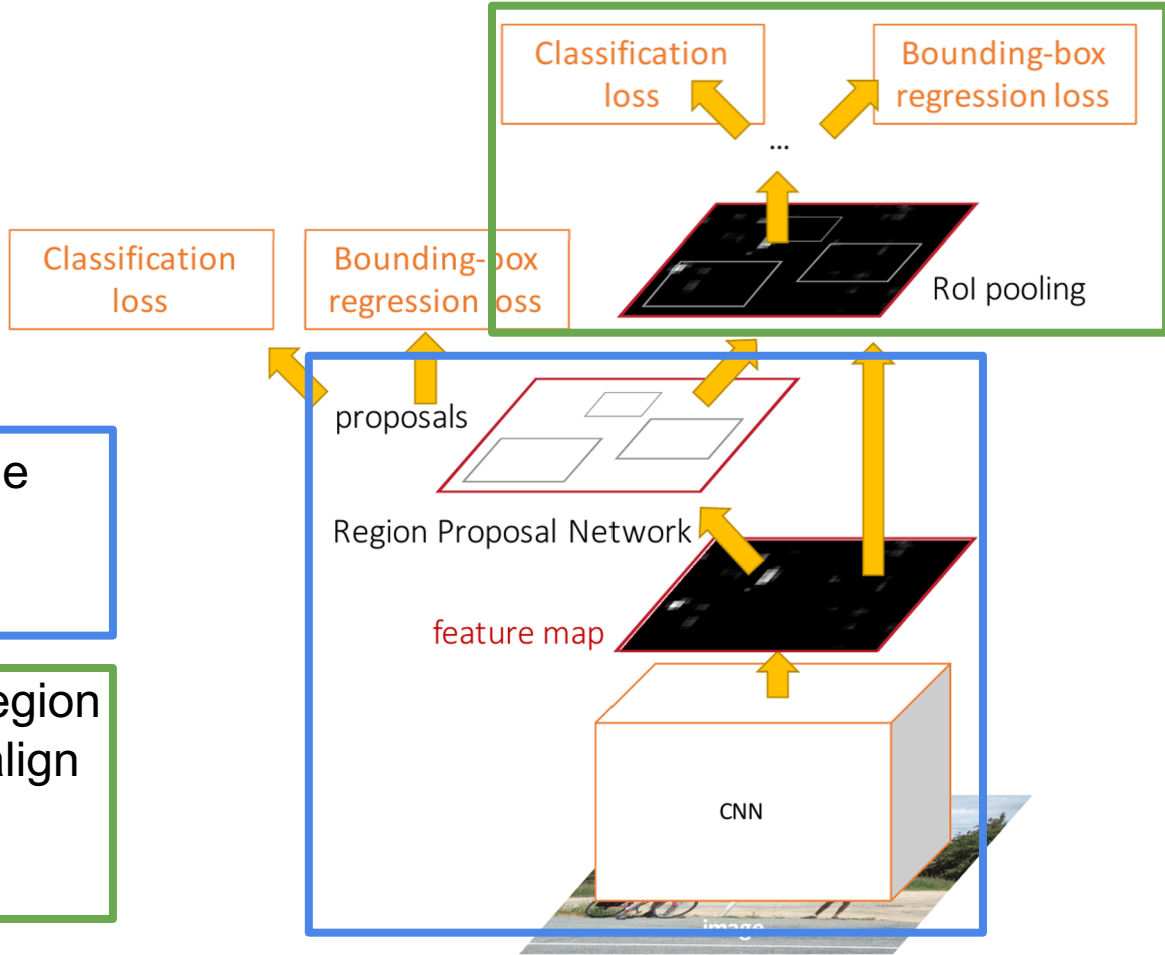
Faster R-CNN is a  
**Two-stage object detector**

First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



# Faster R-CNN:

Make CNN do proposals!

Faster R-CNN is a  
**Two-stage object detector**

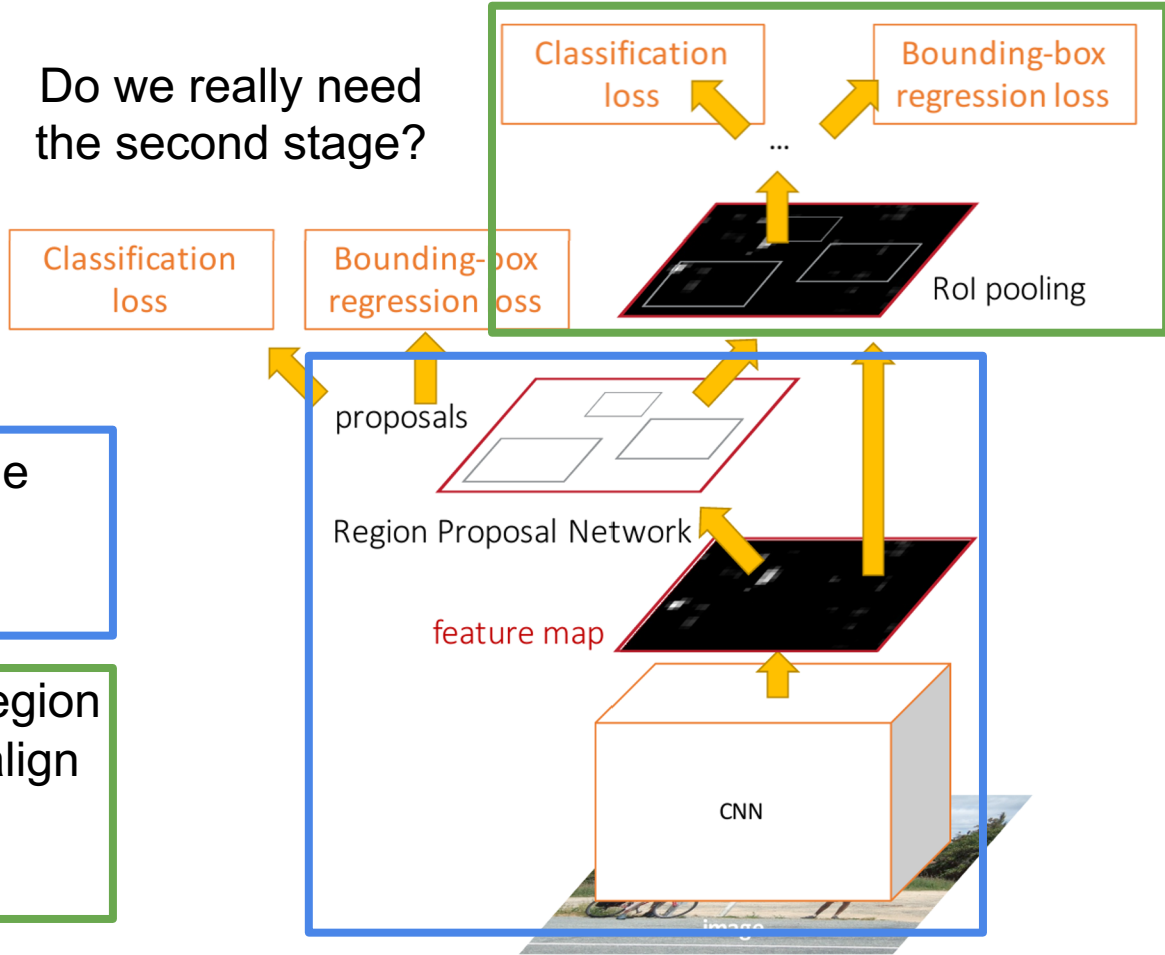
First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

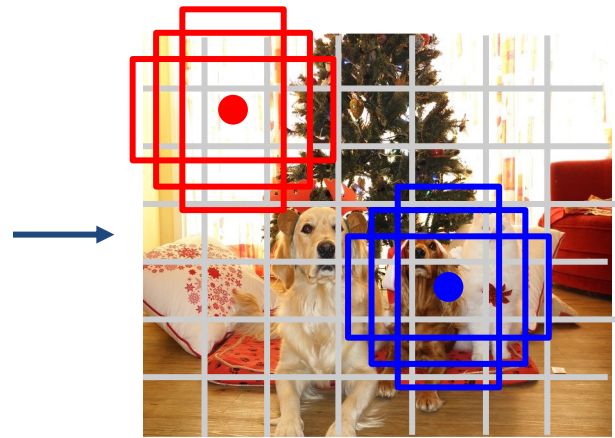
Do we really need  
the second stage?



# Single-Stage Object Detectors: YOLO / SSD / RetinaNet



Input image  
 $3 \times H \times W$



Divide image into grid  
 $7 \times 7$   
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:  $(dx, dy, dh, dw, \text{confidence})$
  - Predict scores for each of  $C$  classes (including background as a class)
  - Looks a lot like RPN, but category-specific!

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  
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

# Object Detection: Lots of variables ...

## Backbone Network

VGG16

ResNet-101

Inception V2

Inception V3

Inception

ResNet

MobileNet

## “Meta-Architecture”

Two-stage: Faster R-CNN

Single-stage: YOLO / SSD

Hybrid: R-FCN

## Image Size

## # Region Proposals

...

## Takeaways

Faster R-CNN is slower  
but more accurate

SSD is much faster but  
not as accurate

Bigger / Deeper  
backbones work better

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

Zou et al, “Object Detection in 20 Years: A Survey”, arXiv 2019

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



# Instance Segmentation

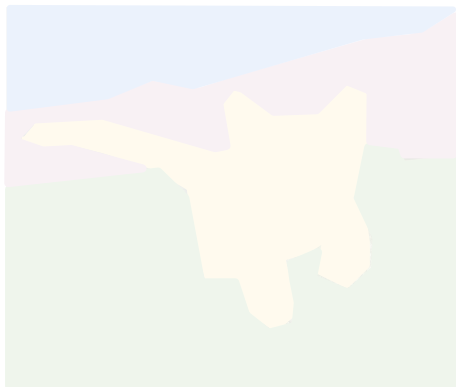
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

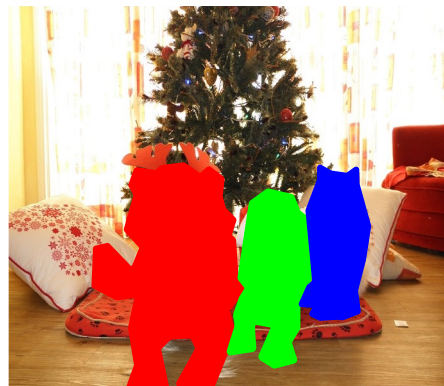
Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

# Object Detection: Faster R-CNN

## Object Detection

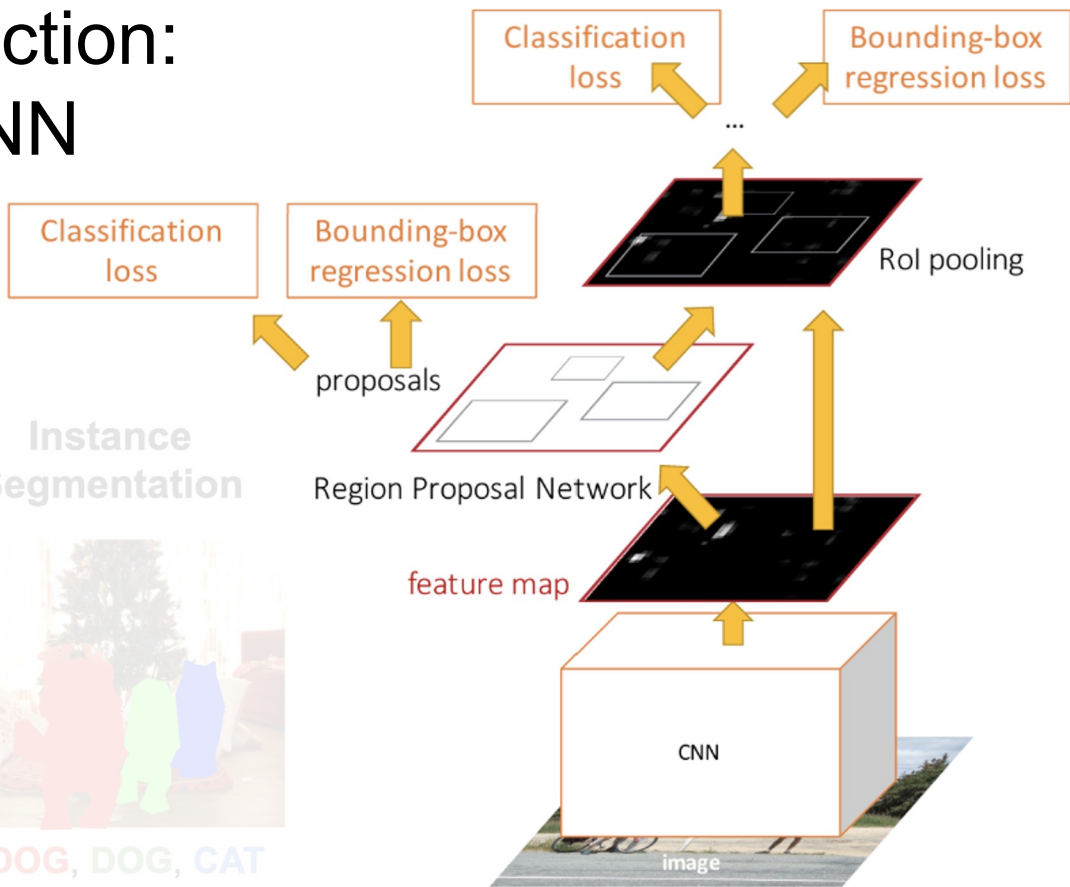


DOG, DOG, CAT

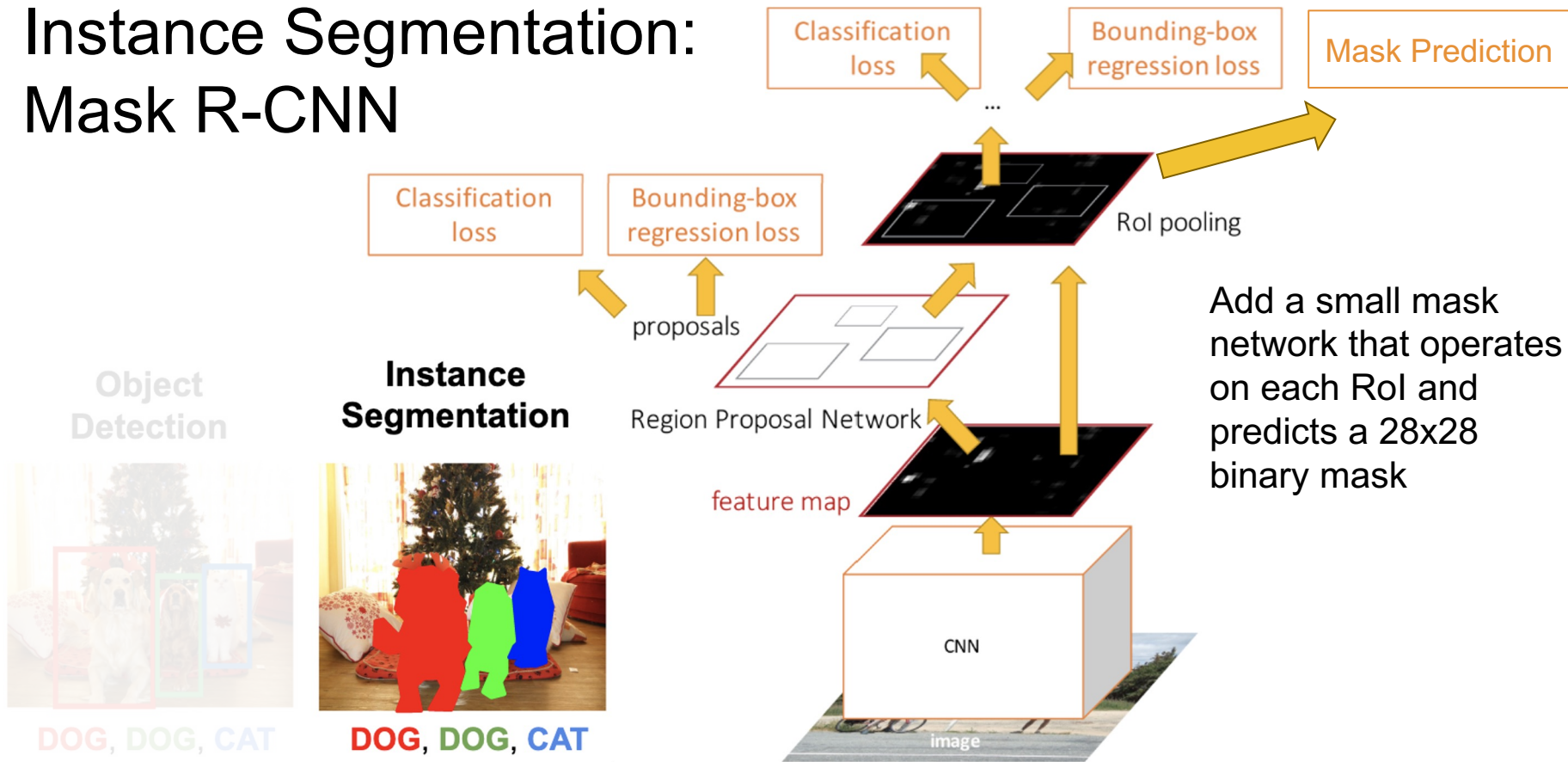
## Instance Segmentation



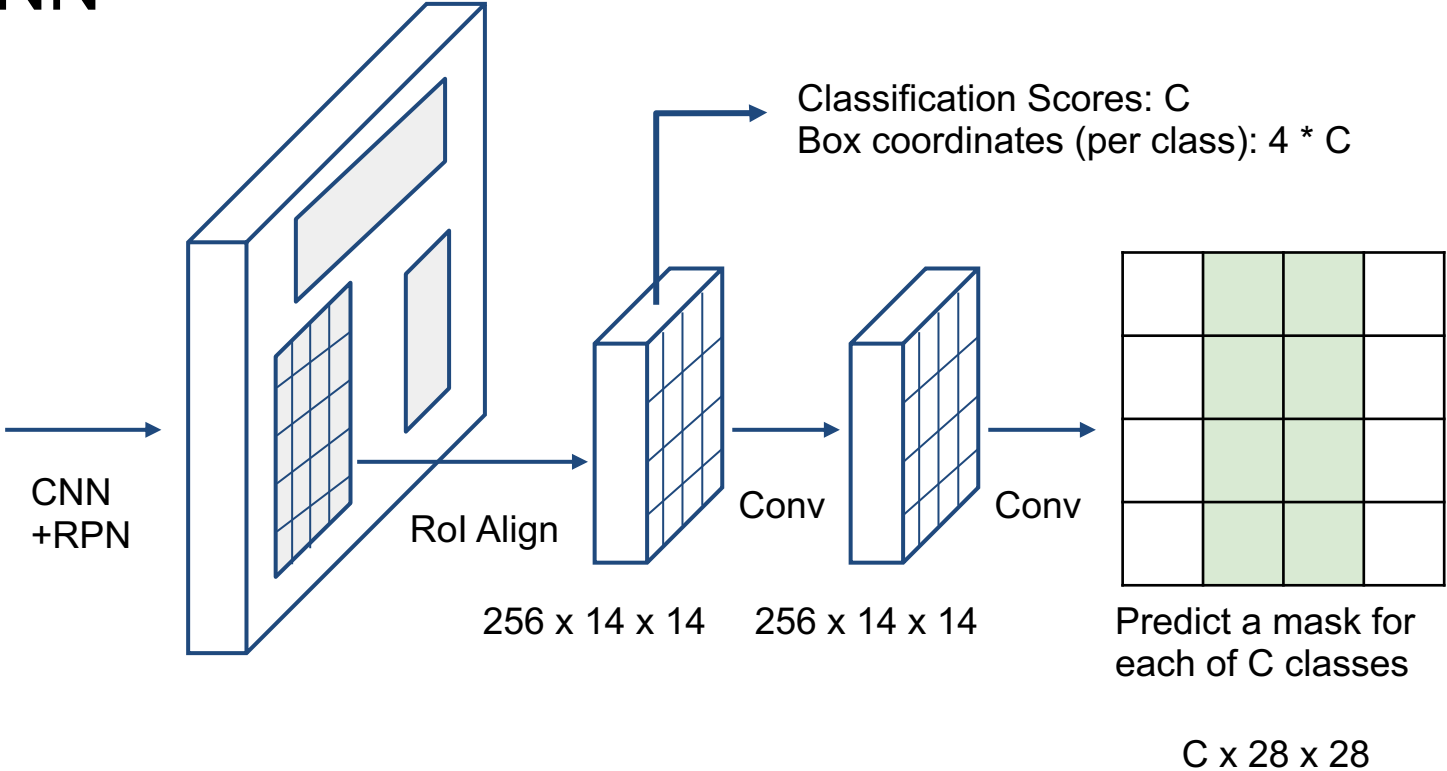
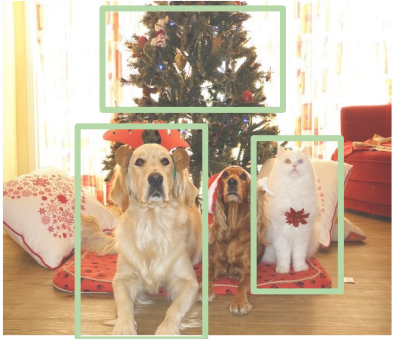
DOG, DOG, CAT



# Instance Segmentation: Mask R-CNN

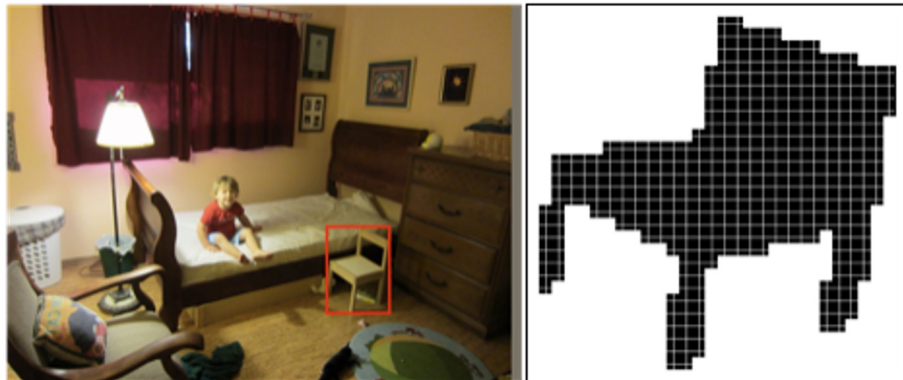


# Mask R-CNN

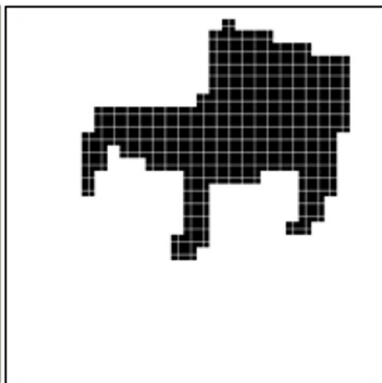
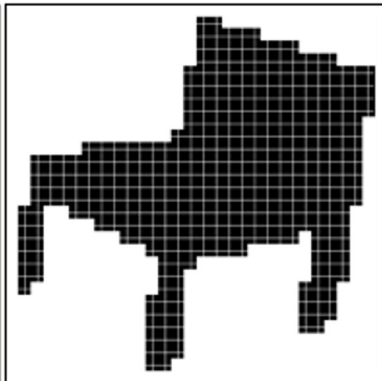
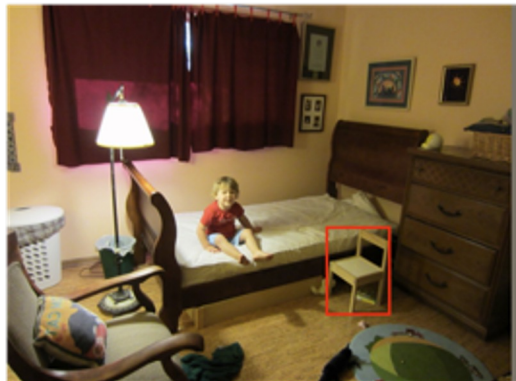


He et al, "Mask R-CNN", arXiv 2017

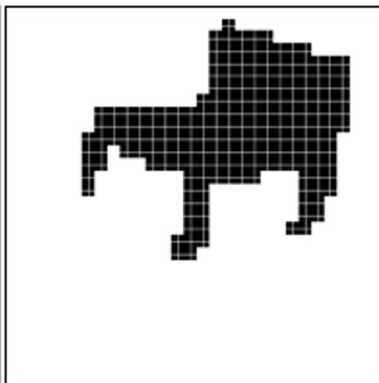
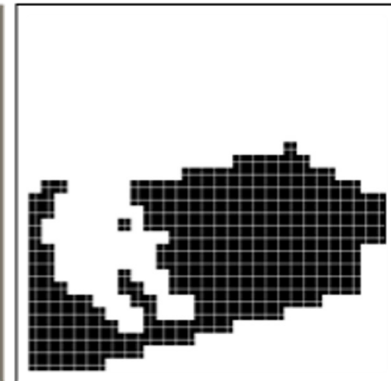
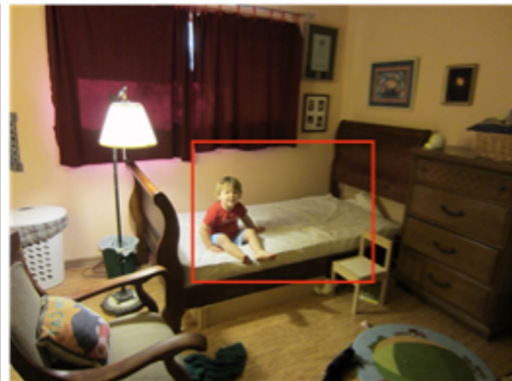
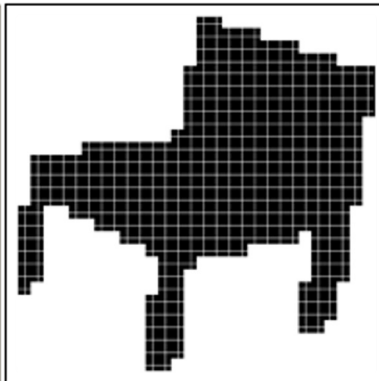
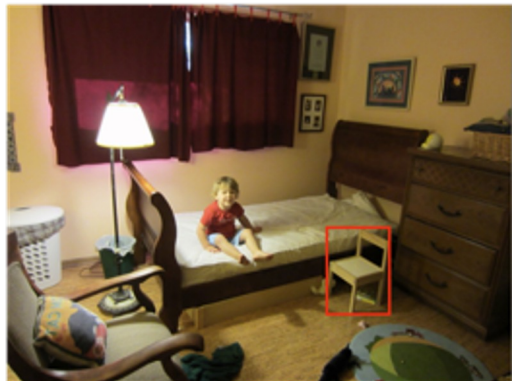
# Mask R-CNN: Example Mask Training Targets



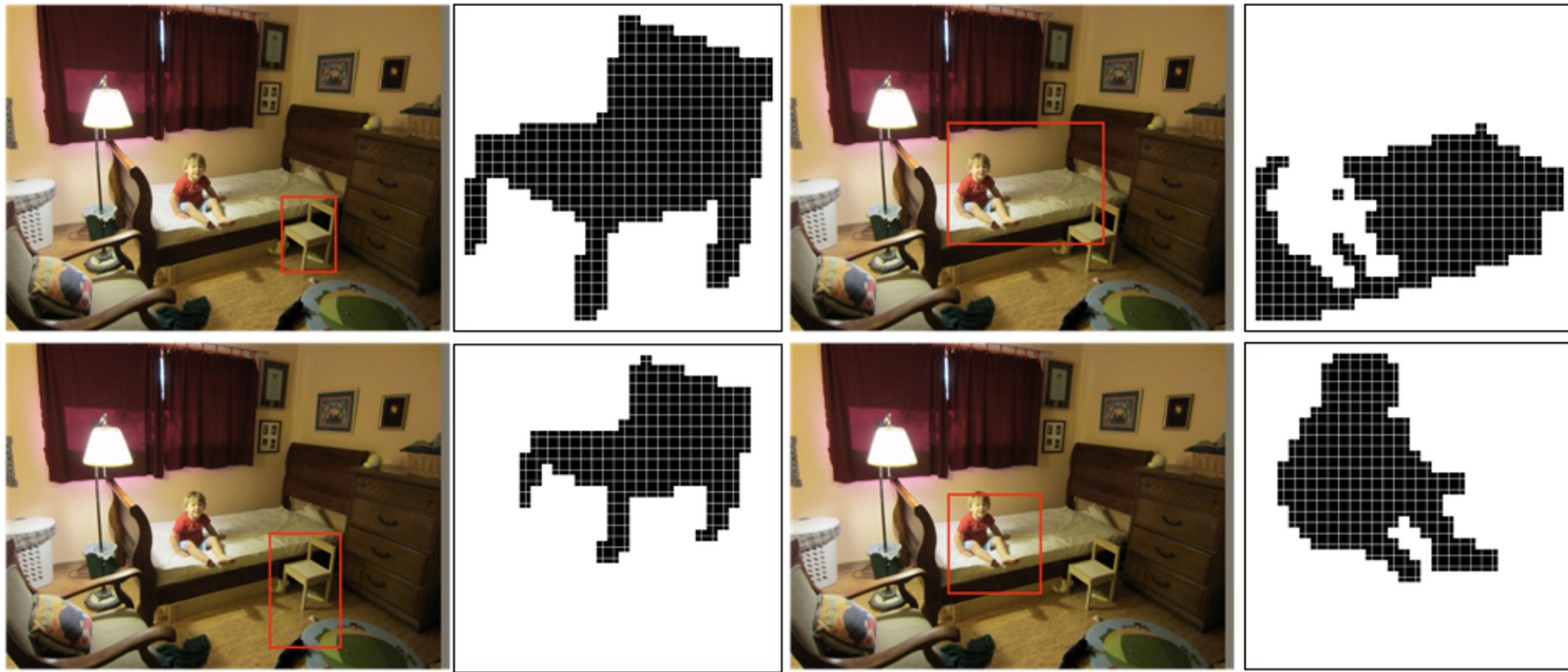
# Mask R-CNN: Example Mask Training Targets



# Mask R-CNN: Example Mask Training Targets



# Mask R-CNN: Example Mask Training Targets



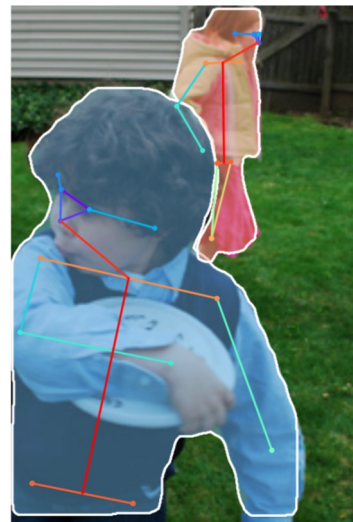
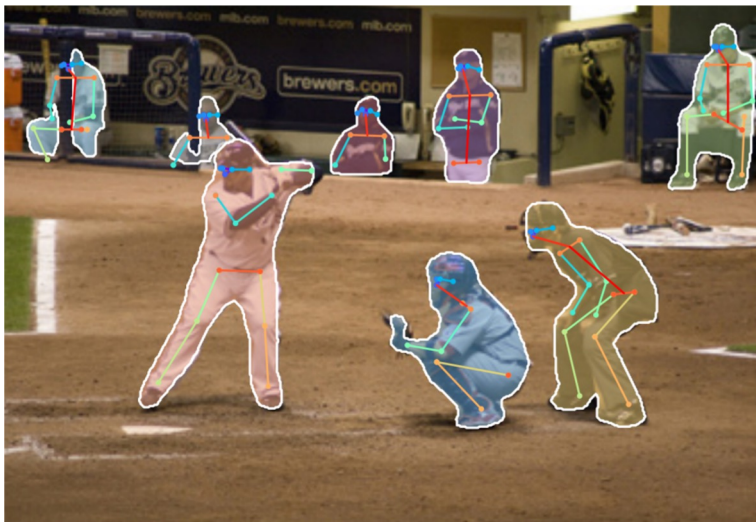


# Mask R-CNN: Very Good Results!



# Mask R-CNN

## Also does pose



# Open Source Frameworks

Lots of good implementations on GitHub!

TensorFlow Detection API:

[https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection)

Faster RCNN, SSD, RFCN, Mask R-CNN, ...

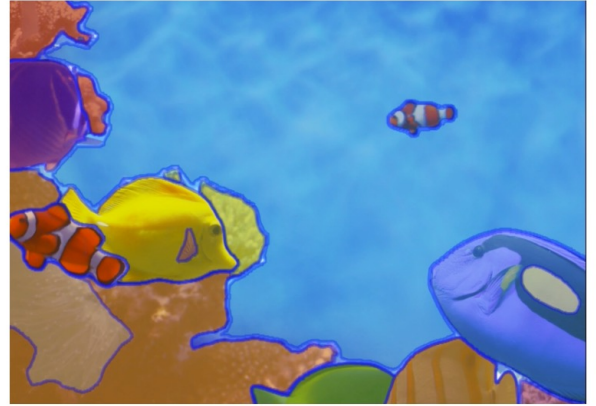
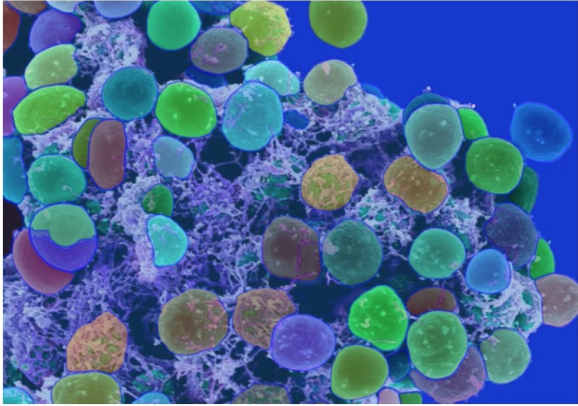
Detectron2 (PyTorch)

<https://github.com/facebookresearch/detectron2>

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

Finetune on your own dataset with pre-trained models

# Foundation Image Segmentation Models



SegmentAnything (Meta AI, 2023)

Try it yourself! <https://segment-anything.com/demo#>

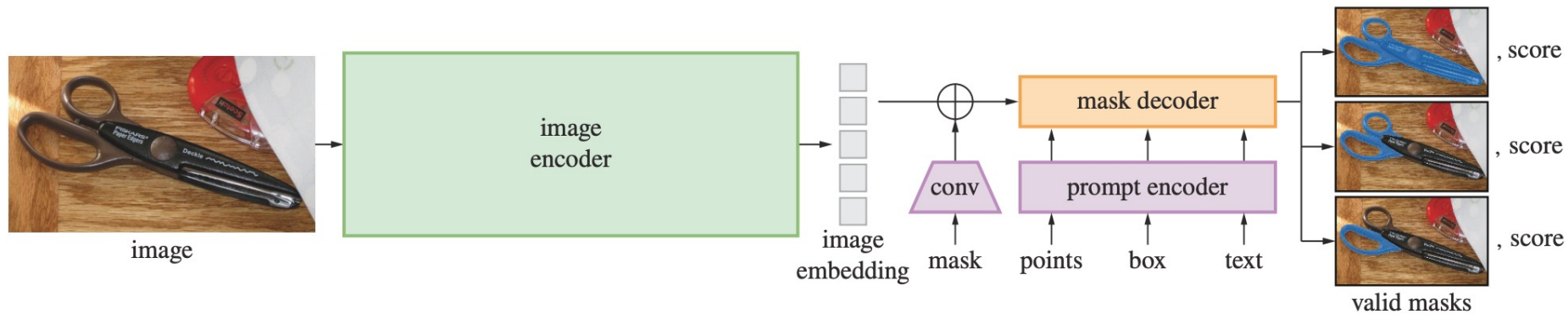
# Foundation Image Segmentation Models



SegmentAnything (Meta AI, 2023)

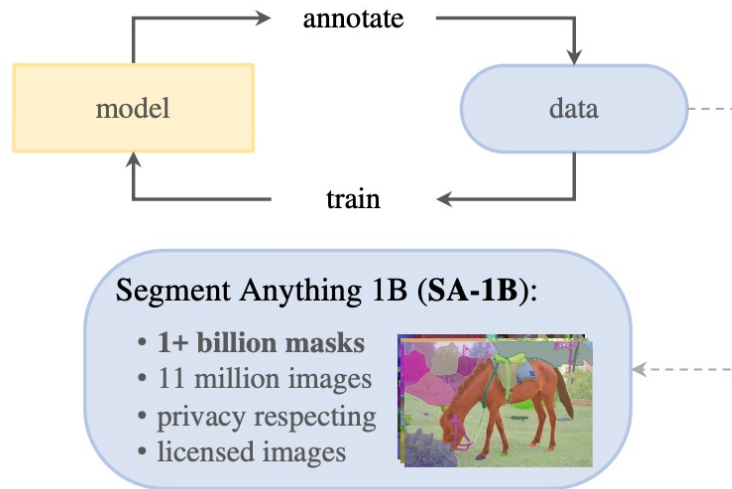
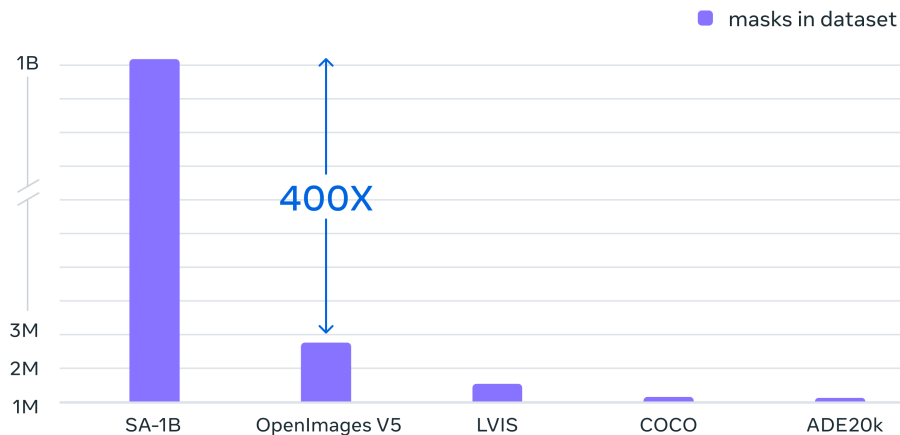
Try it yourself! <https://segment-anything.com/demo#>

# Foundation Image Segmentation Models



SegmentAnything (Meta AI, 2023)

# Foundation Image Segmentation Models

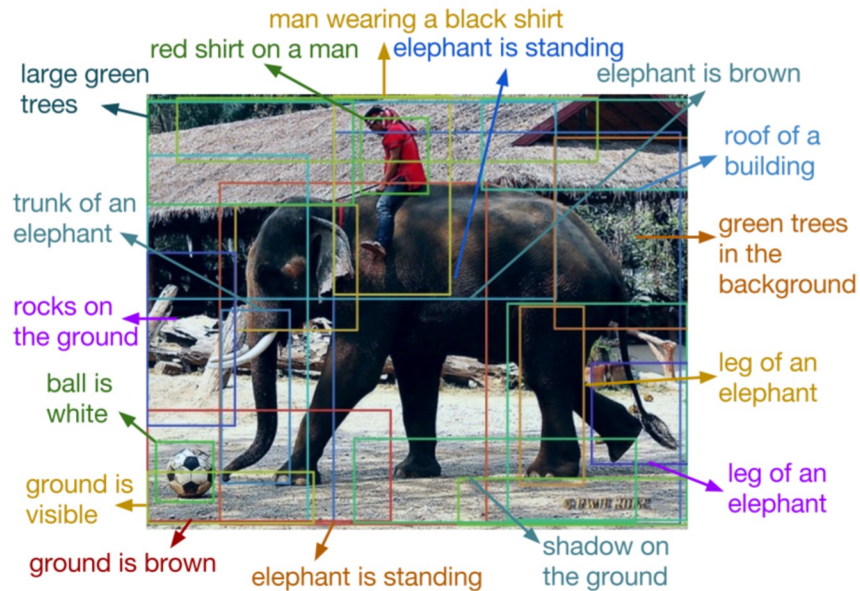
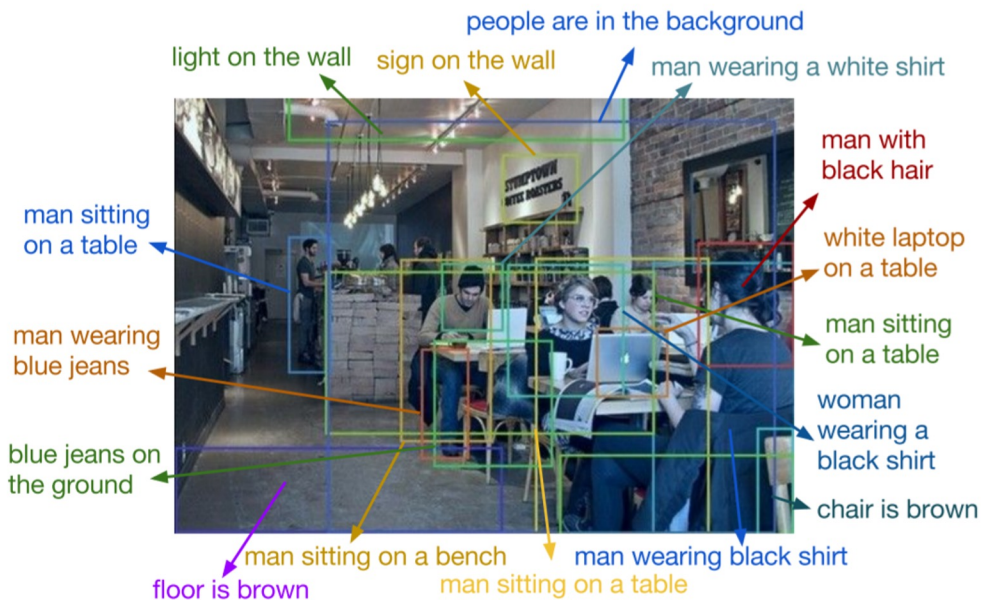


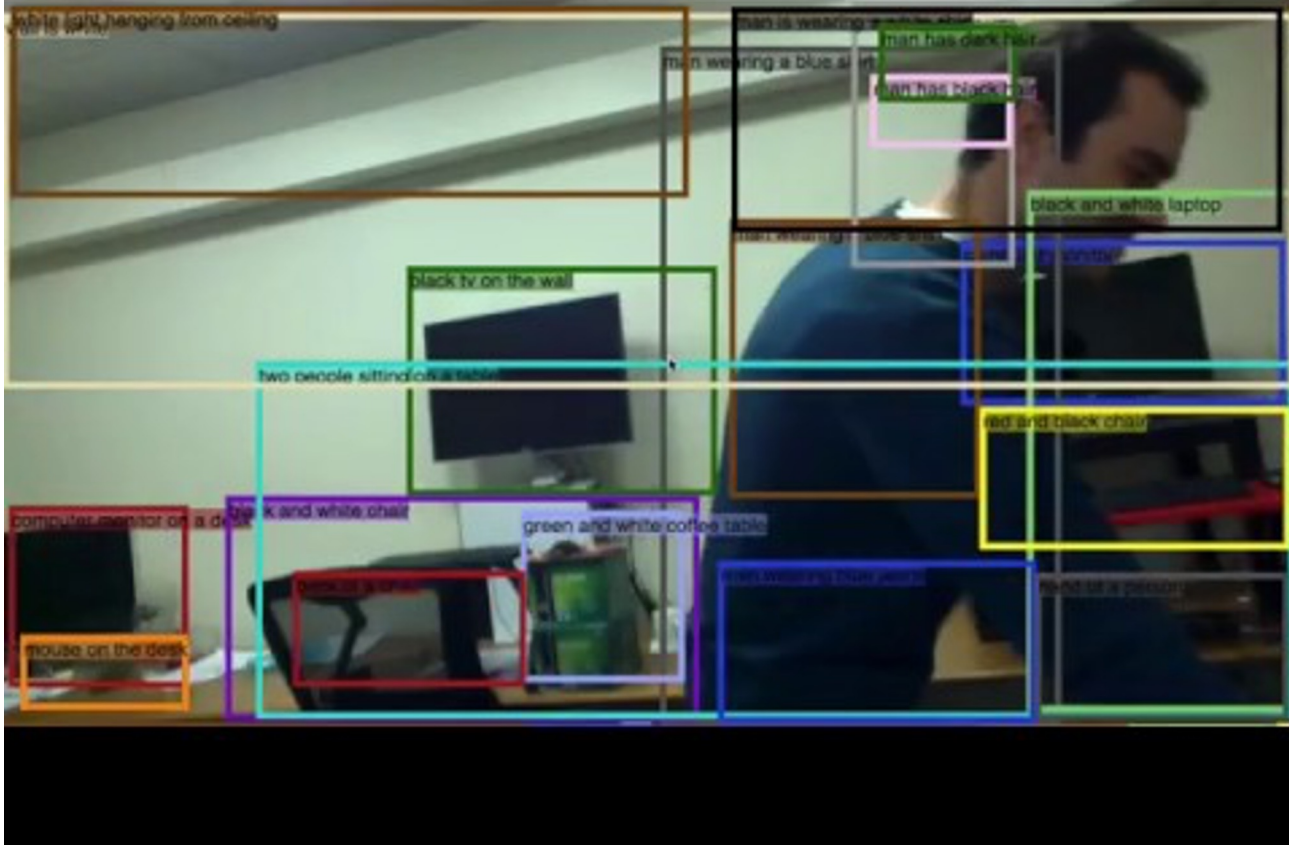
SegmentAnything (Meta AI, 2023)

Beyond 2D Object Detection...

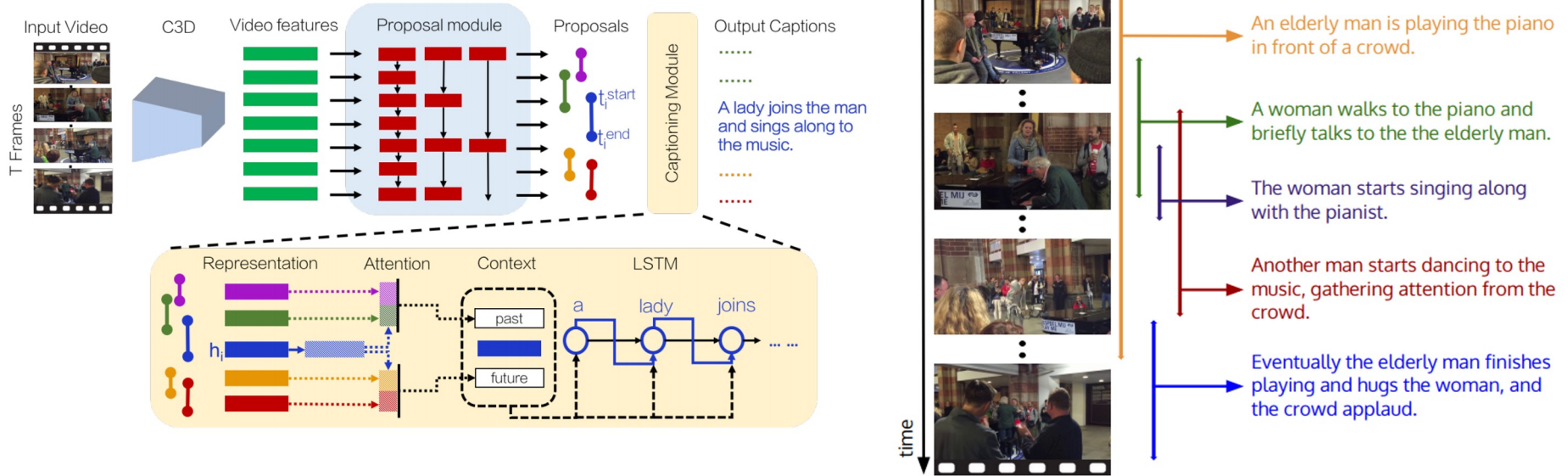


# Object Detection + Captioning = Dense Captioning

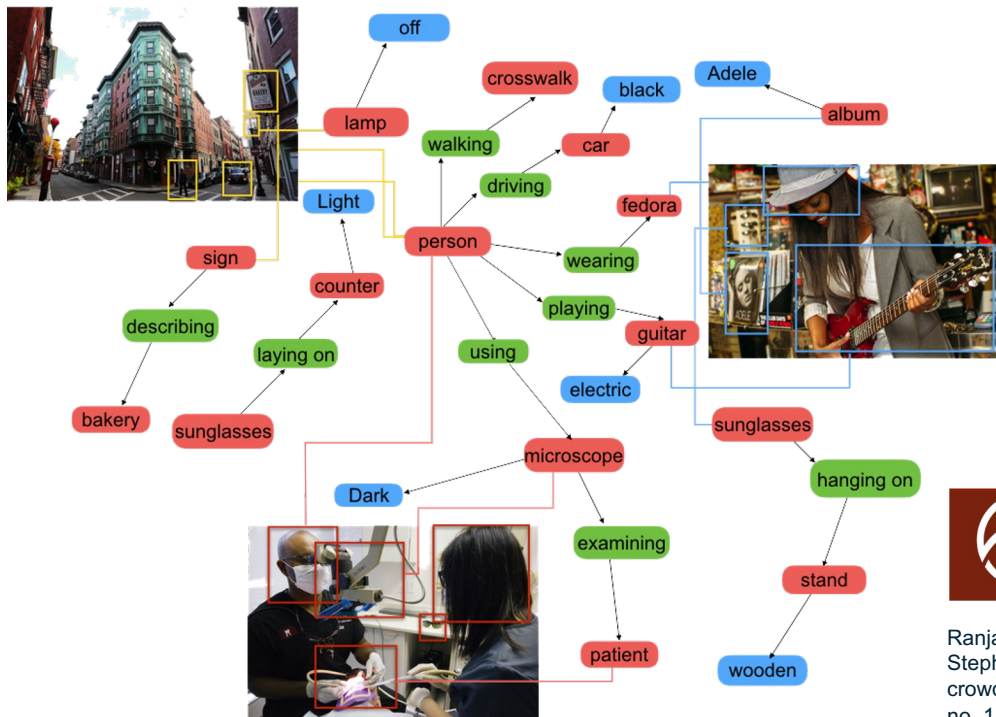




# Dense Video Captioning



# Objects + Relationships = Scene Graphs



108,077 Images

5.4 Million Region Descriptions

1.7 Million Visual Question Answers

3.8 Million Object Instances

2.8 Million Attributes

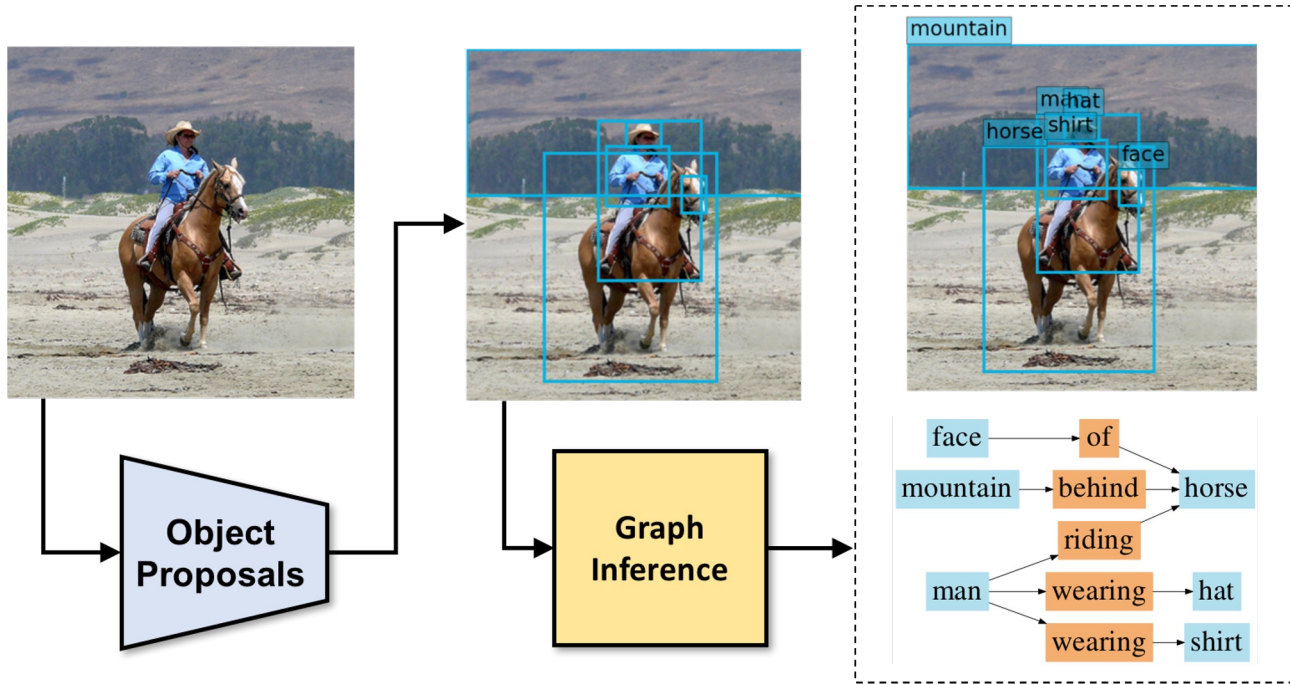
2.3 Million Relationships

Everything Mapped to Wordnet Synsets

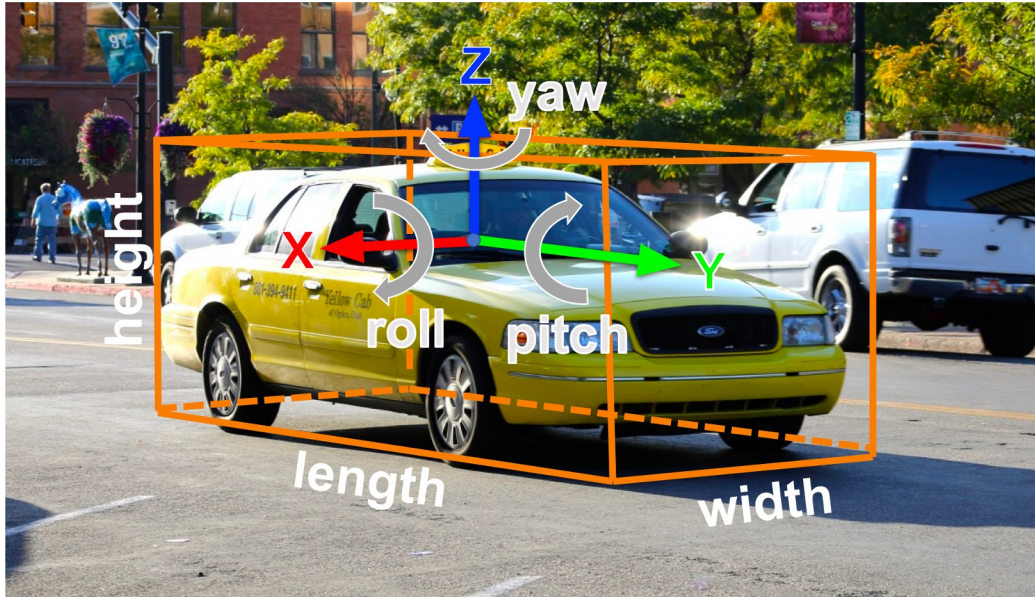
 **VISUALGENOME**

Ranjay Krishna, 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.

# Scene Graph Prediction



# 3D Object Detection



2D Object Detection:

2D bounding box

$(x, y, w, h)$

3D Object Detection:

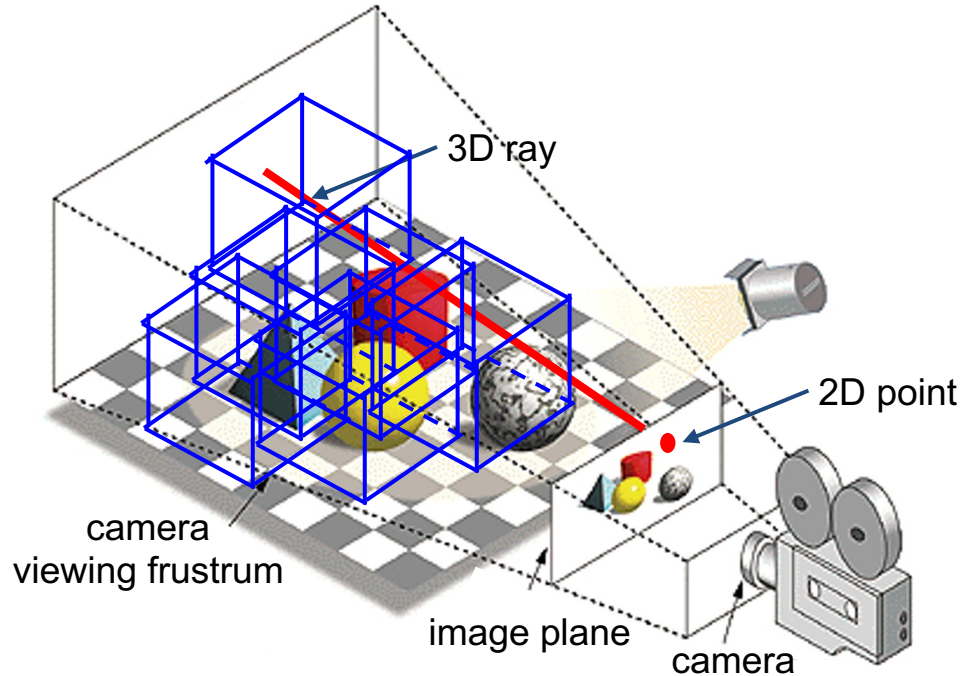
3D oriented bounding box

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

Simplified bbox: no roll & pitch

Much harder problem than 2D  
object detection!

# 3D Object Detection: Simple Camera Model

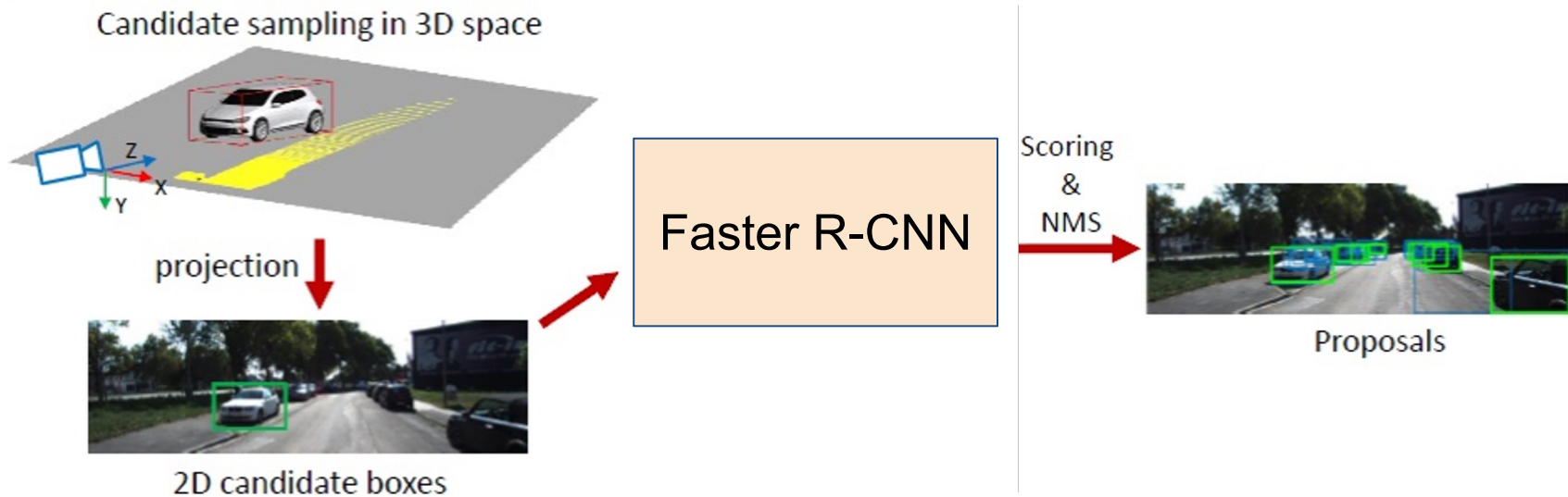


A point on the image plane corresponds to a **ray** in the 3D space

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

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

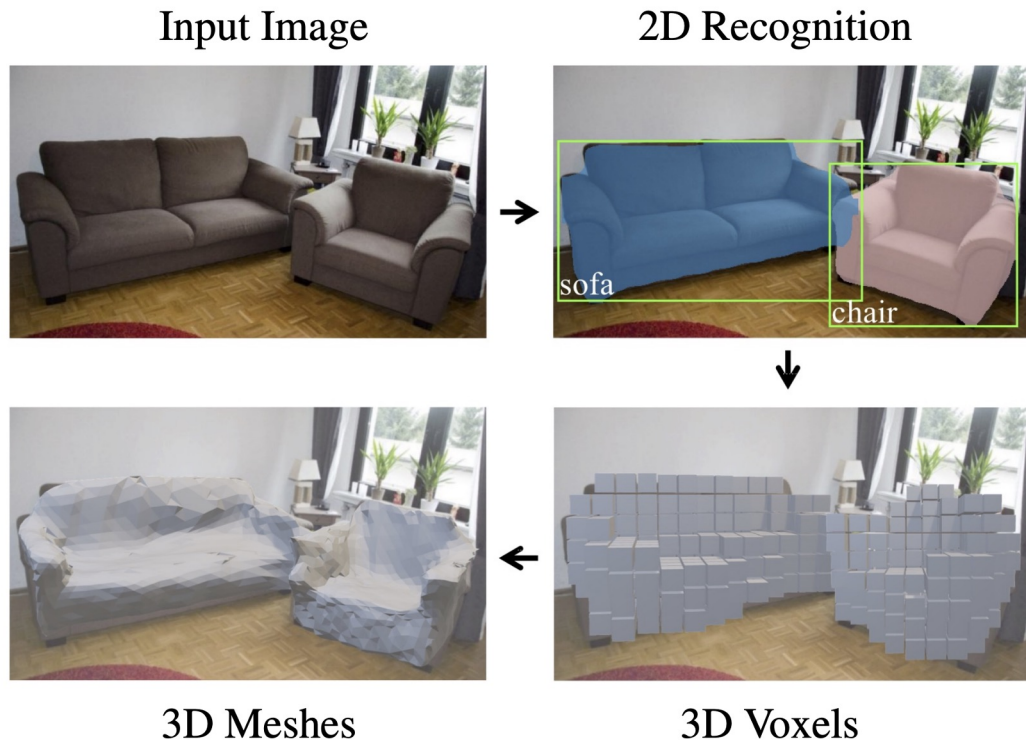
# 3D Object Detection: Monocular Camera



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



# 3D Shape Prediction: Mesh R-CNN



# Recap: Lots of computer vision tasks!

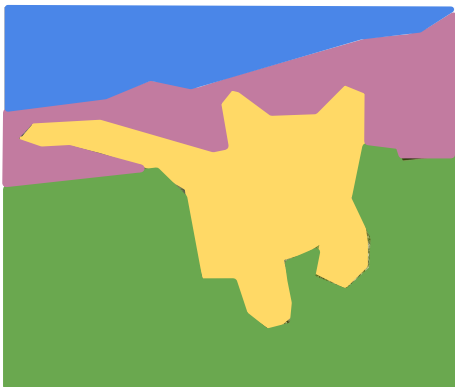
## Classification



**CAT**

No spatial extent

## Semantic Segmentation



**GRASS, CAT,  
TREE, SKY**

No objects, just pixels

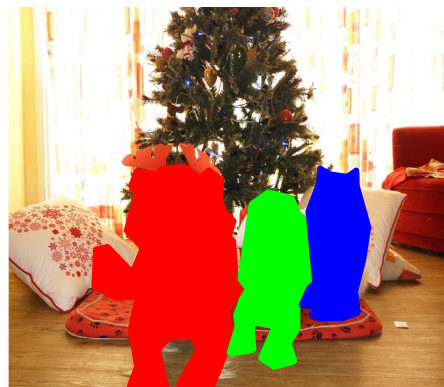
## Object Detection



**DOG, DOG, CAT**

Multiple Object

## Instance Segmentation



**DOG, DOG, CAT**

Next time: Visualizing Neural Networks