# Lecture 11: Object Detection and Image Segmentation



## Computer Vision Tasks

Classification | Semantic Segmentation | Object Detection | Instance Segmentation

CAT | GRASS, CAT, TREE, SKY | DOG, DOG, CAT | DOG, DOG, CAT

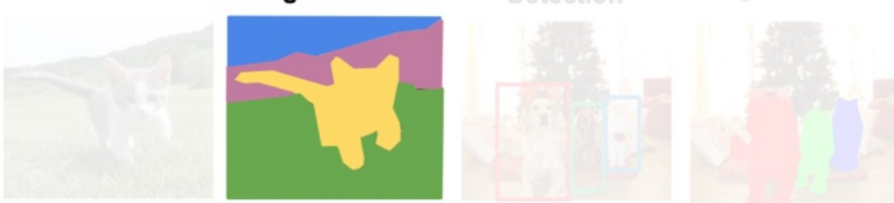
No spatial extent | No objects, just pixels | Multiple Object

This image is CC0 public domain



## Semantic Segmentation

Classification | Semantic Segmentation | Object Detection | Instance Segmentation



CAT | GRASS, CAT, TREE, SKY | DOG, DOG, CAT | DOG, DOG, CAT

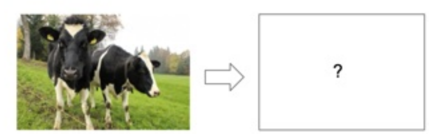
No spatial extent | No objects, just pixels | Multiple Object



## 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

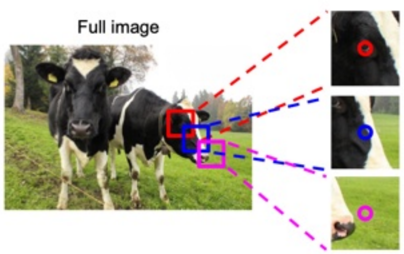

Impossible to classify without context

Q: how do we include context?



## Semantic Segmentation Idea: Sliding Window

Full image



Q: how do we model this?

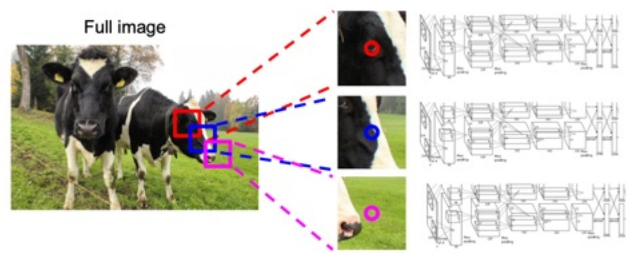


## Semantic Segmentation Idea: Sliding Window

Extract patch

Classify center pixel with CNN

Full image



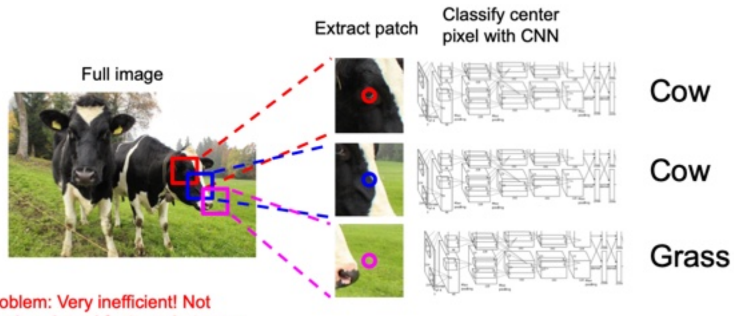
Cow

Cow

Grass

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

## Semantic Segmentation Idea: Sliding Window

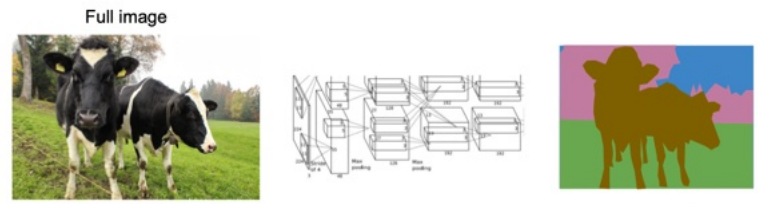



Problem: Very inefficient! Not reusing shared features between overlapping patches

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: Convolution



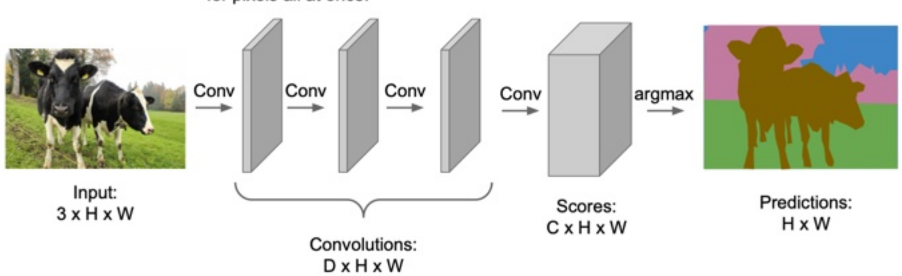

An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

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

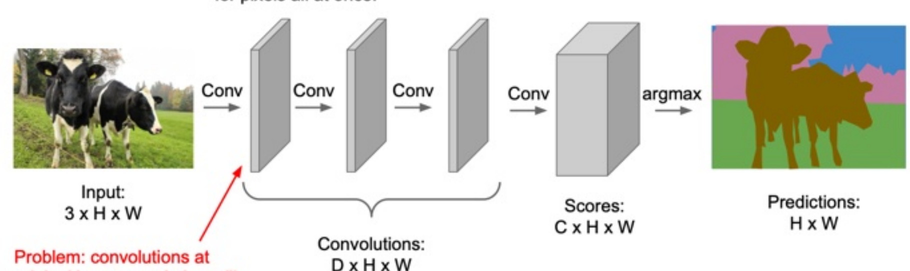
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!






## Semantic Segmentation Idea: Fully Convolutional

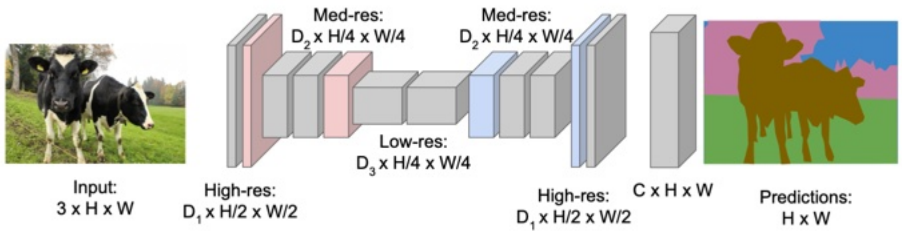
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!




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!




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

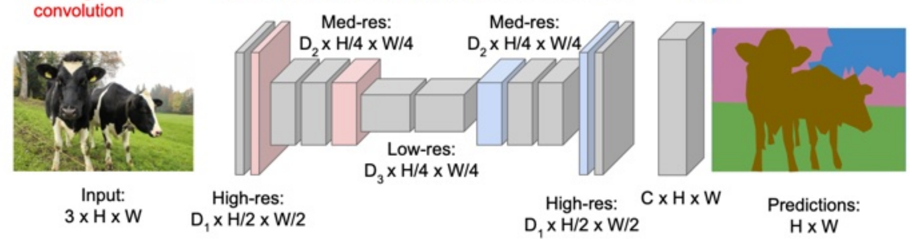


## Semantic Segmentation Idea: Fully Convolutional

**Downsampling**: Pooling, strided convolution

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

**Upsampling**: ???




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



## In-Network upsampling: "Unpooling"

**Nearest Neighbor**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 2 |
| 3 | 3 | 4 | 4 |
| 3 | 3 | 4 | 4 |

Output: 4 x 4

**"Bed of Nails"**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 1 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 |

Output: 4 x 4



## In-Network upsampling: "Max Unpooling"

**Max Pooling**
Remember which element was max!

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

Input: 4 x 4

| | |
|---|---|
| 5 | 6 |
| 7 | 8 |

Output: 2 x 2

Rest of the network

**Max Unpooling**
Use positions from pooling layer

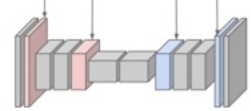
| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 0 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 |

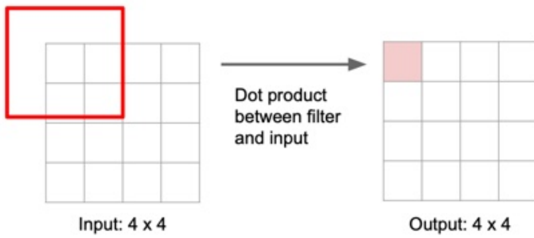
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



## Learnable Upsampling

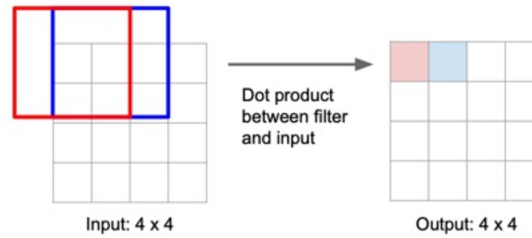
**Recall:** Normal 3 x 3 convolution, stride 1 pad 1






## Learnable Upsampling

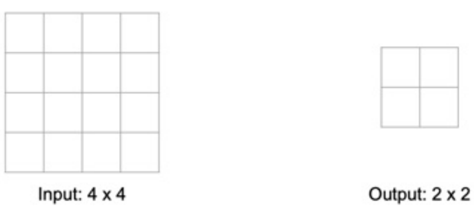
**Recall:** Normal 3 x 3 convolution, stride 1 pad 1






## Learnable Upsampling

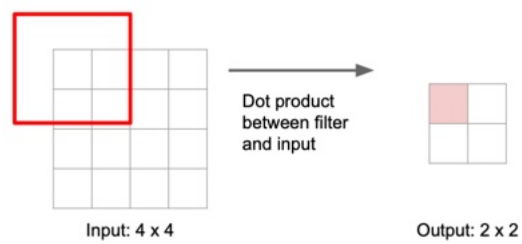
**Recall:** Normal 3 x 3 convolution, stride 2 pad 1





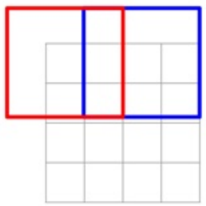

## Learnable Upsampling

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



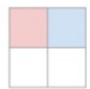
# Learnable Upsampling

**Recall:** Normal 3 x 3 convolution, <u>stride 2</u> 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

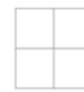
Stride gives ratio between movement in input and output

We can interpret strided convolution as “learnable downsampling”.

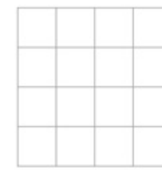


# Learnable Upsampling: Transposed Convolution

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



Input: 2 x 2



Output: 4 x 4



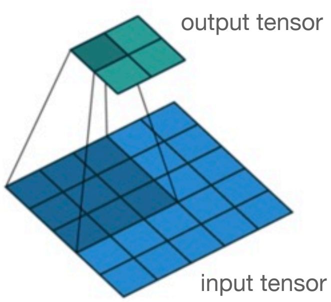



Figure 1: Normal Convolution

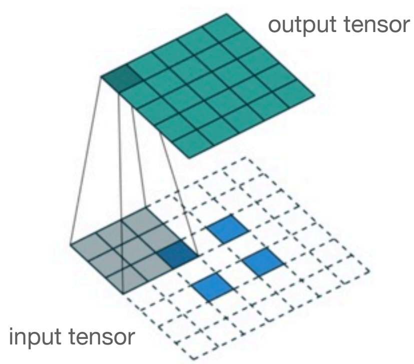
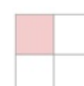



Figure 2: Transposed convolution.

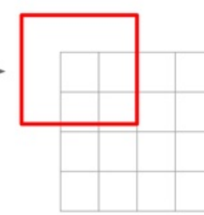
# Learnable Upsampling: Transposed Convolution

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



Input: 2 x 2

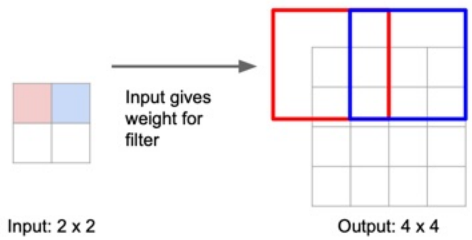
Input gives weight for filter



Output: 4 x 4

## Learnable Upsampling: Transposed Convolution

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



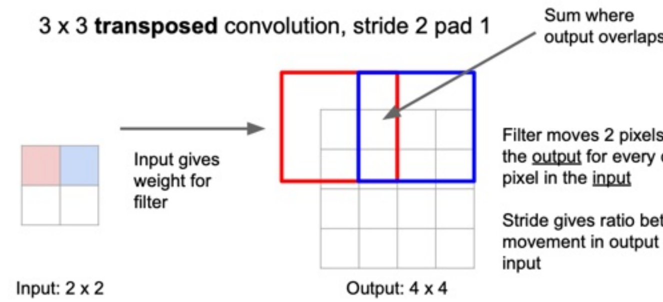

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input



## Learnable Upsampling: Transposed Convolution

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

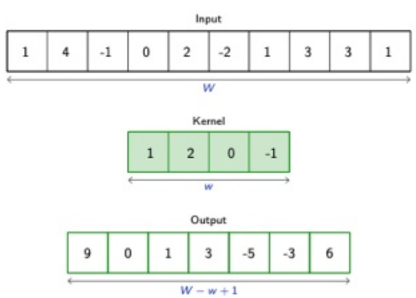



Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

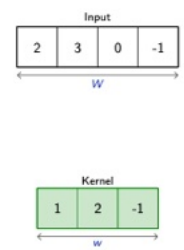


Convolution layer

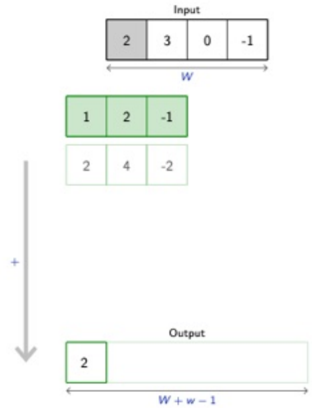





Transposed convolution layer

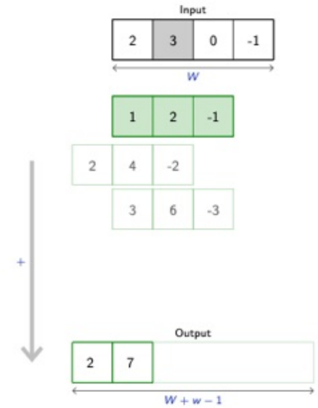


## Transposed convolution layer



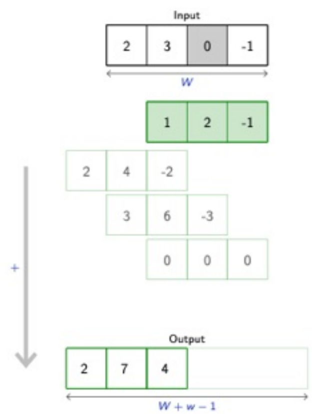



## Transposed convolution layer



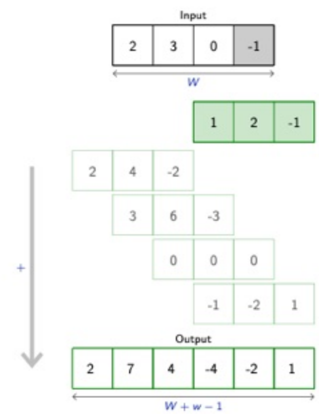



## Transposed convolution layer

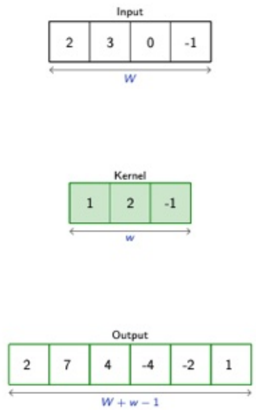





## Transposed convolution layer

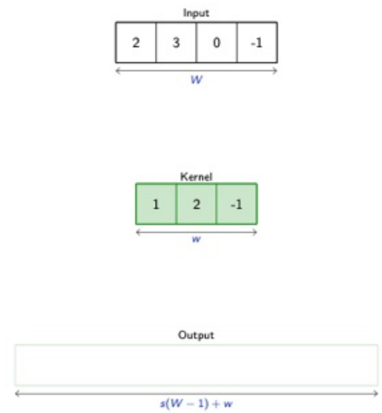


## Transposed convolution layer



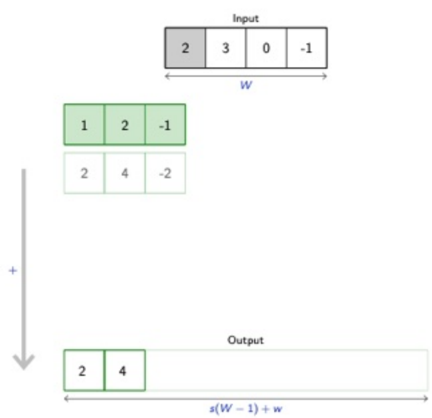



## Transposed convolution layer (stride = 2)



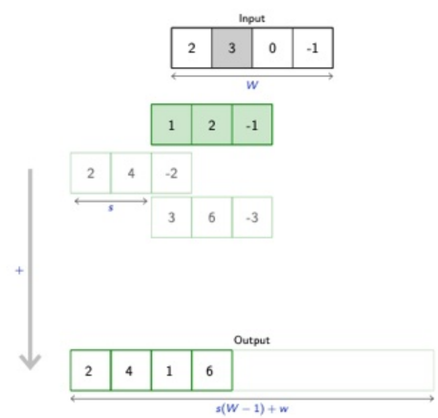


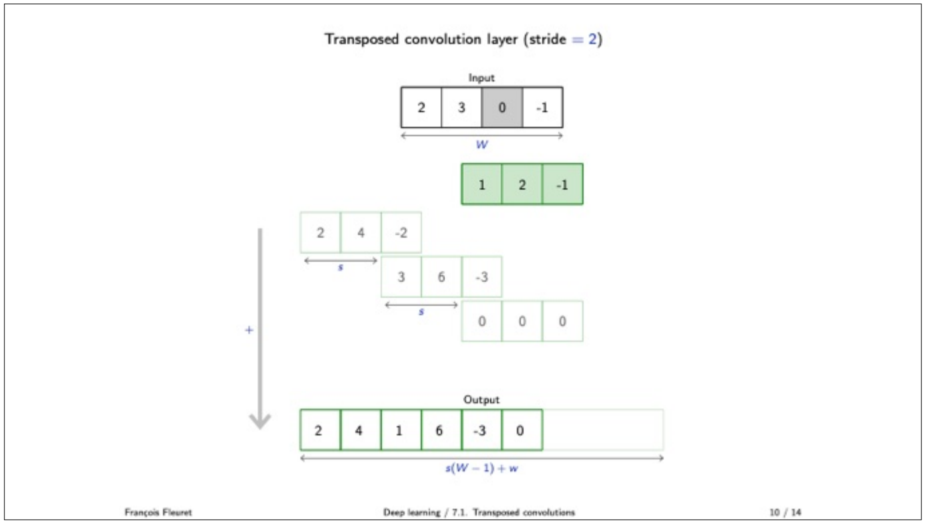

## Transposed convolution layer (stride = 2)

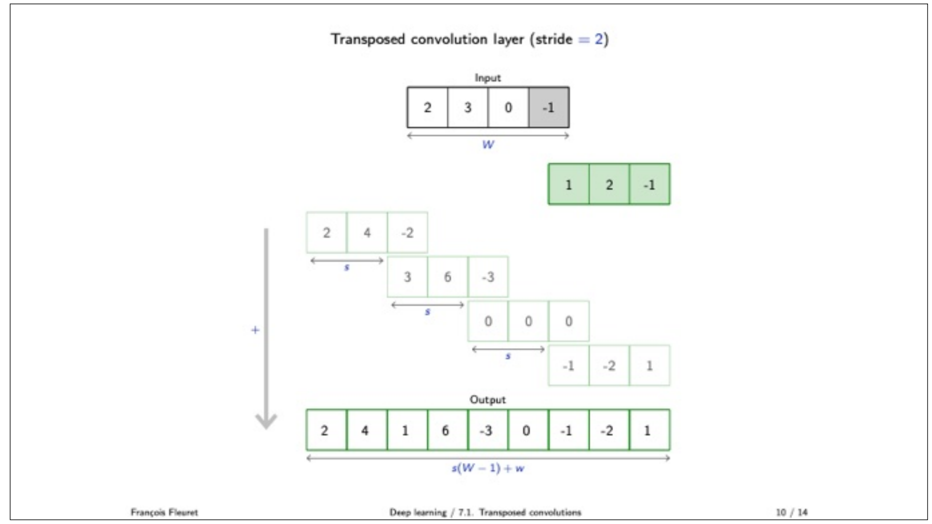


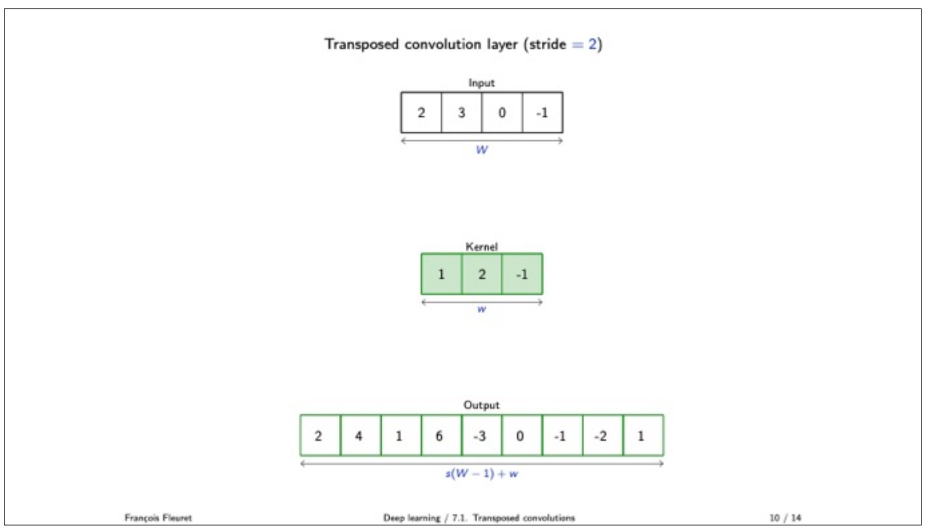


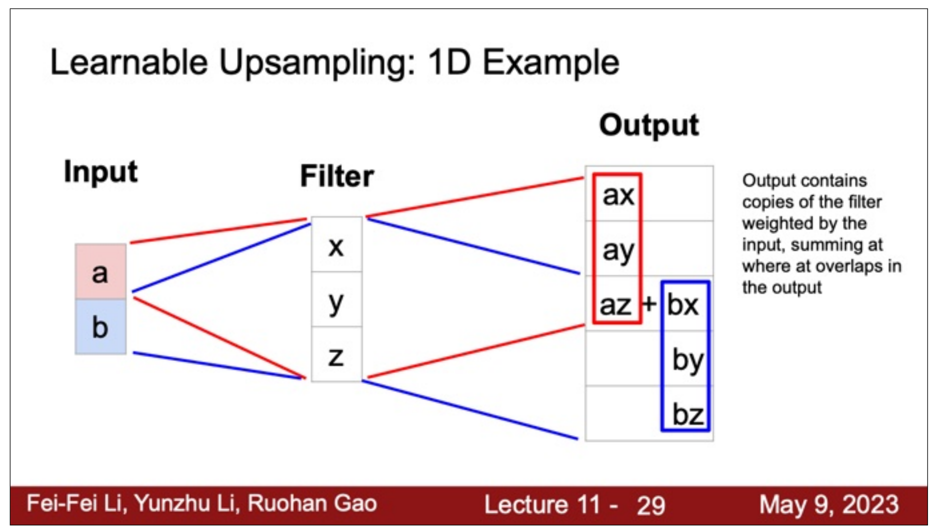

## Transposed convolution layer (stride = 2)




Transposed convolution layer (stride = 2)
Input
2 3 0 -1
W
1 2 -1
2 4 -2
s
3 6 -3
s
0 0 0
+
Output
2 4 1 6 -3 0
s(W − 1) + w
François Fleuret
Deep learning / 7.1. Transposed convolutions
10 / 14


Transposed convolution layer (stride = 2)
Input
2 3 0 -1
W
1 2 -1
2 4 -2
s
3 6 -3
s
0 0 0
s
-1 -2 1
+
Output
2 4 1 6 -3 0 -1 -2 1
s(W − 1) + w
François Fleuret
Deep learning / 7.1. Transposed convolutions
10 / 14


Transposed convolution layer (stride = 2)
Input
2 3 0 -1
W
Kernel
1 2 -1
w
Output
2 4 1 6 -3 0 -1 -2 1
s(W − 1) + w
François Fleuret
Deep learning / 7.1. Transposed convolutions
10 / 14


Learnable Upsampling: 1D Example
Input
a
b
Filter
x
y
z
Output
ax
ay
az + bx
by
bz
Output contains copies of the filter weighted by the input, summing at where at overlaps in the output
Fei-Fei Li, Yunzhu Li, Ruohan Gao
Lecture 11 - 29
May 9, 2023

## 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, <u>stride=2</u>, 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, <u>stride=2</u>, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

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

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

Example: 1D transposed conv, kernel size=3, <u>stride=2</u>, padding=0

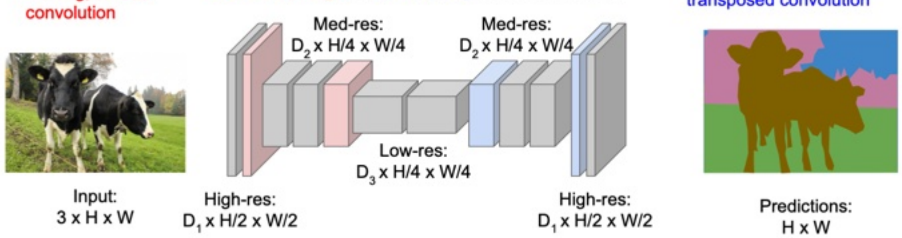


## Semantic Segmentation Idea: Fully Convolutional

**Downsampling**: Pooling, strided convolution

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

**Upsampling**: Unpooling or strided transposed convolution



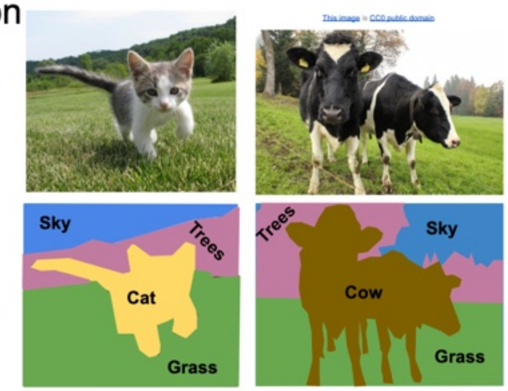

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015



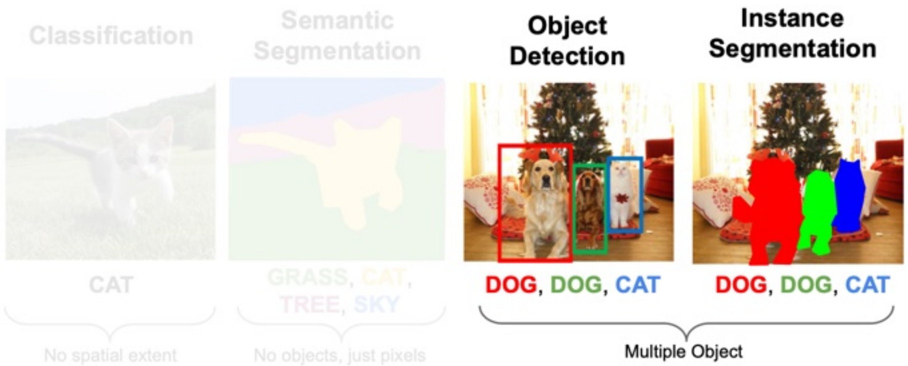
## Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



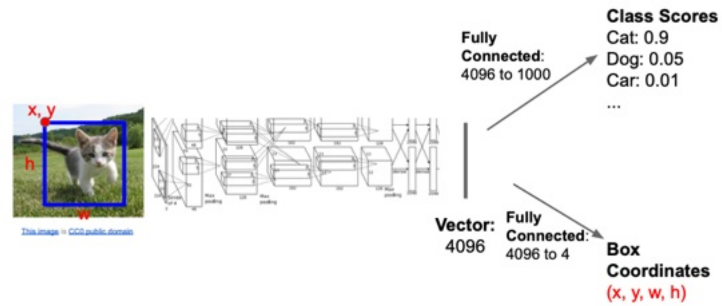
# Object Detection






# Object Detection: Single Object

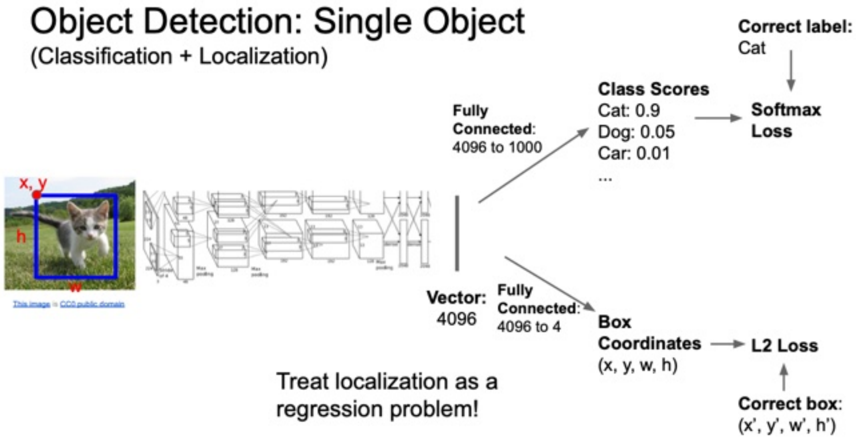
(Classification + Localization)






# Object Detection: Single Object

(Classification + Localization)

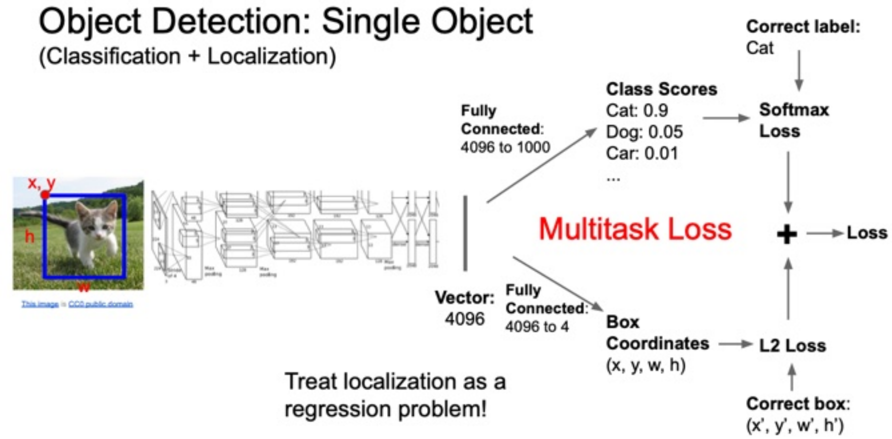



Treat localization as a regression problem!



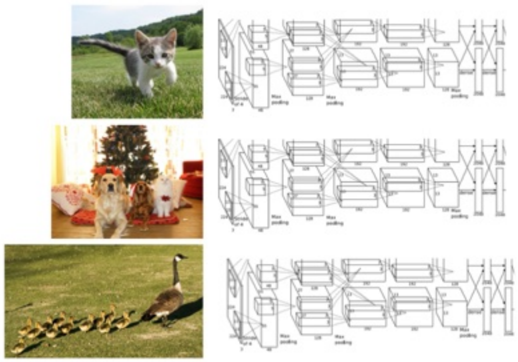
# Object Detection: Single Object

(Classification + Localization)




Treat localization as a regression problem!

## Object Detection: Multiple Objects



CAT: (x, y, w, h)

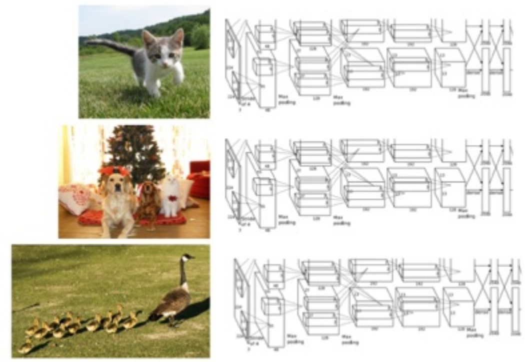
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

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

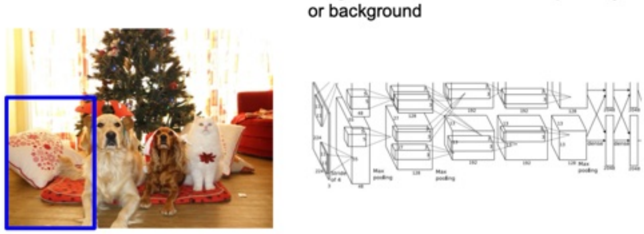
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)
12 numbers

DUCK: (x, y, w, h)
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

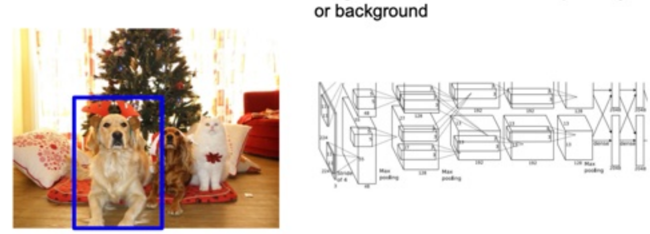


Dog? NO
Cat? NO
Background? YES



## Object Detection: Multiple Objects

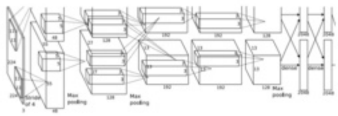
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



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

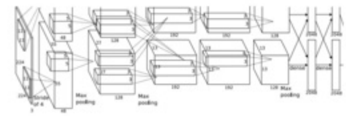


Dog? YES
Cat? NO
Background? NO



## Object Detection: Multiple Objects

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



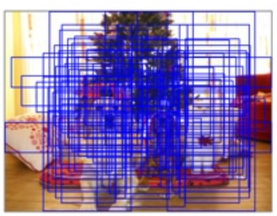
Dog? NO
Cat? YES
Background? NO

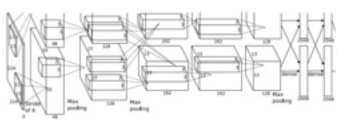
Q: What's the problem with this approach?



## 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!



## 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


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

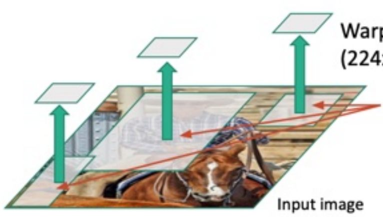


# R-CNN


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



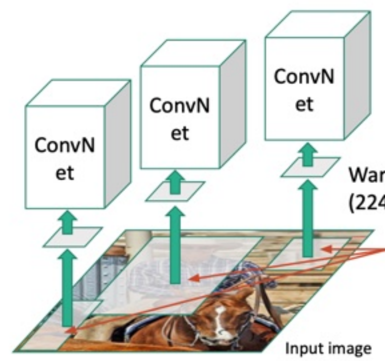
# R-CNN




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

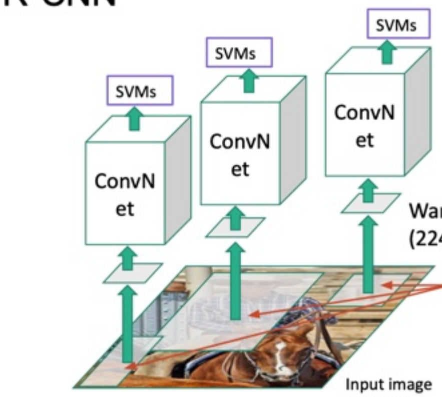


# R-CNN




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




Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

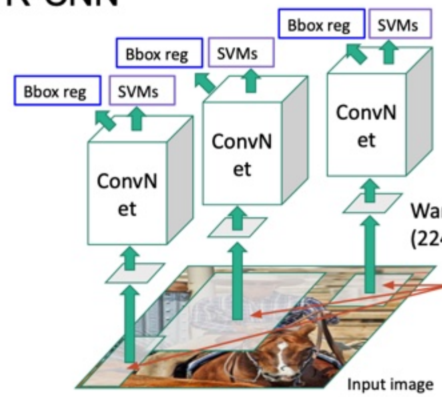
Regions of Interest (RoI) from a proposal method (~2k)

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




# R-CNN

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




Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

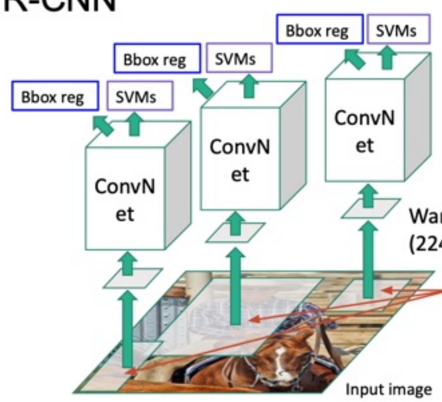
Regions of Interest (RoI) from a proposal method (~2k)

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




# R-CNN

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




Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

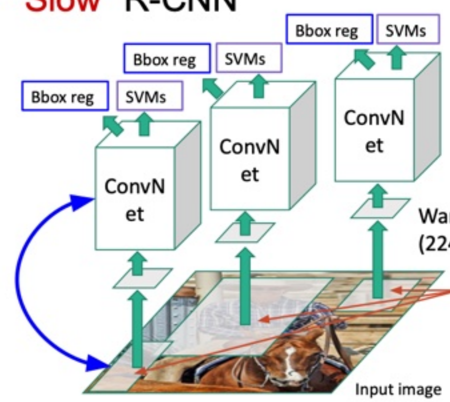
**Problem**: Very slow! Need to do ~2k independent forward passes for each image!

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




# "Slow" R-CNN

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




Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

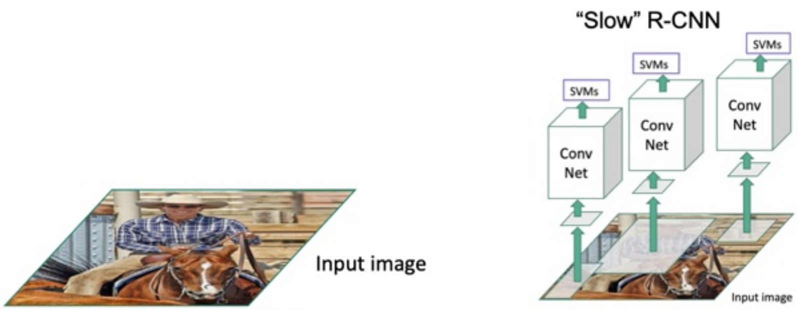
Regions of Interest (RoI) from a proposal method (~2k)

**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.

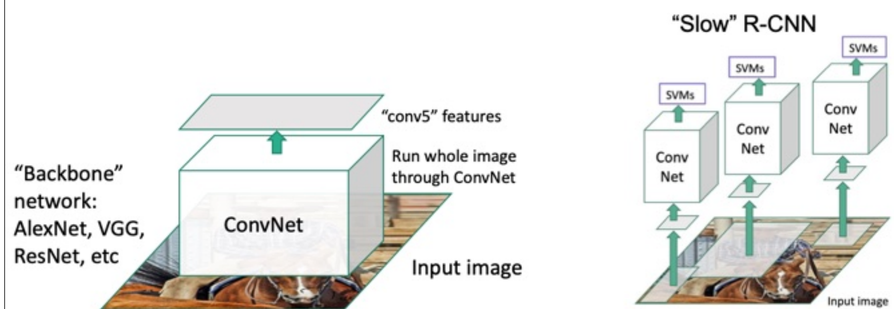
# Fast R-CNN







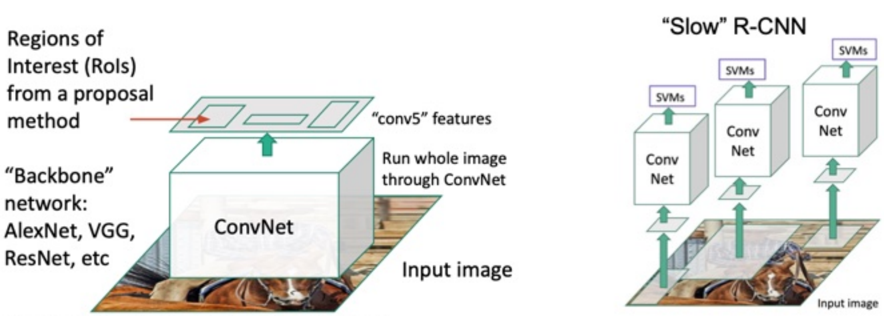

# Fast R-CNN







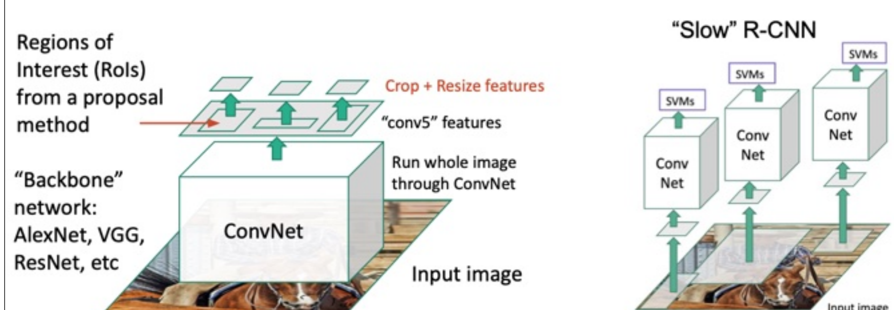

# Fast R-CNN



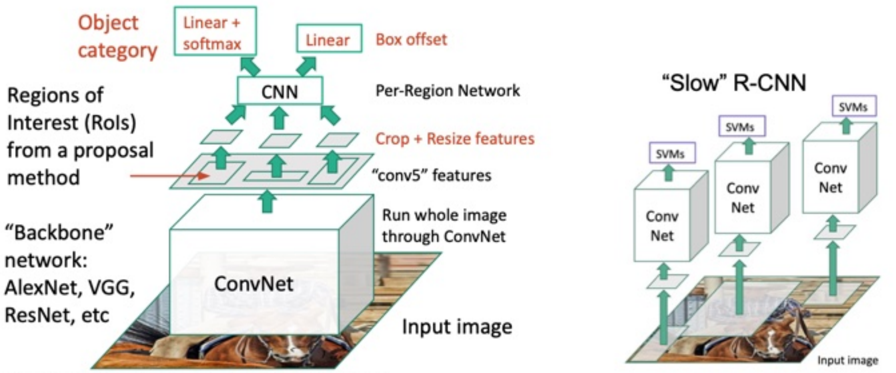





# Fast R-CNN



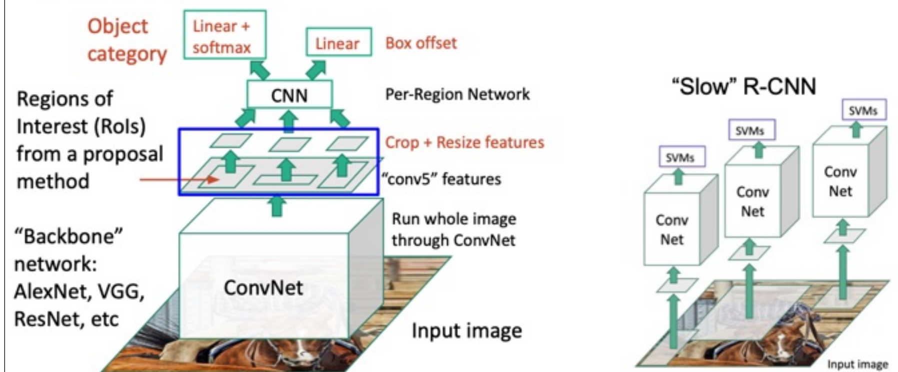
# Fast R-CNN




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



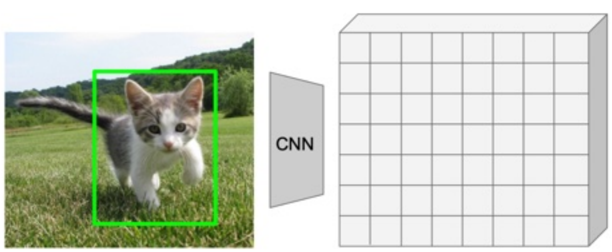
# Fast R-CNN




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



# Cropping Features: RoI Pool




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

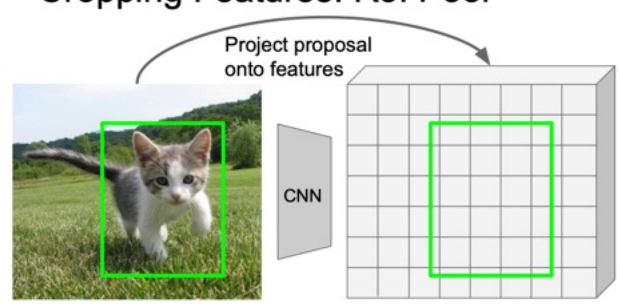
Image features: C x H x W
(e.g. 512 x 20 x 15)

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

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



# Cropping Features: RoI Pool



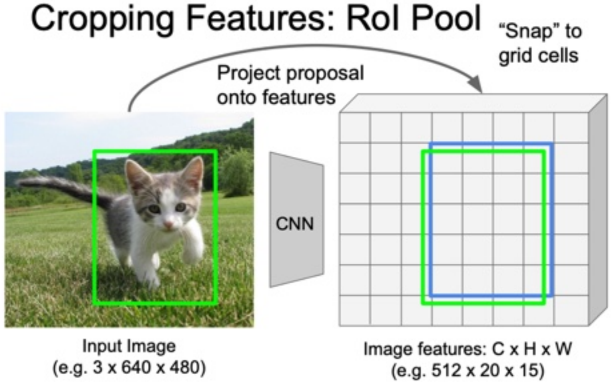

Input Image
(e.g. 3 x 640 x 480)

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

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

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

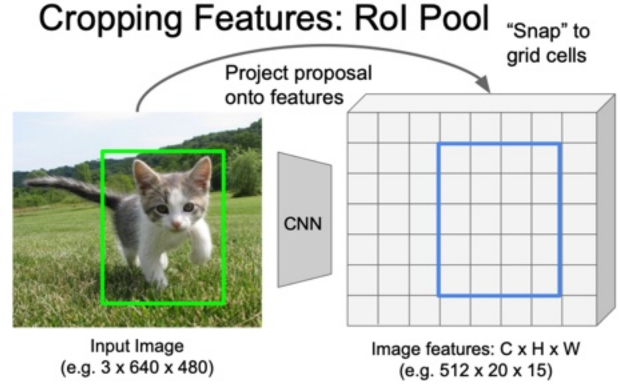
# Cropping Features: RoI Pool




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



# Cropping Features: RoI Pool

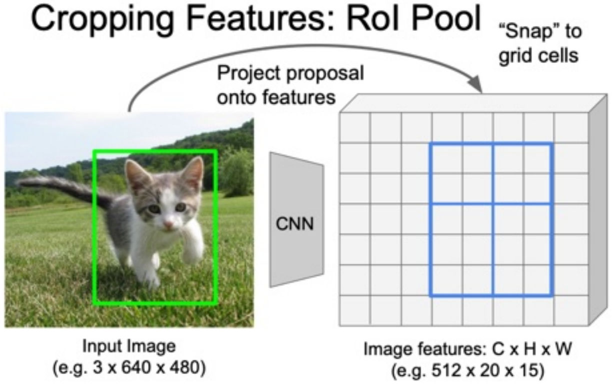



Q: how do we resize the 512 x 5 x 4 region to, e.g., a 512 x 2 x 2 tensor?.

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



# Cropping Features: RoI Pool




Divide into 2x2 grid of (roughly) equal subregions

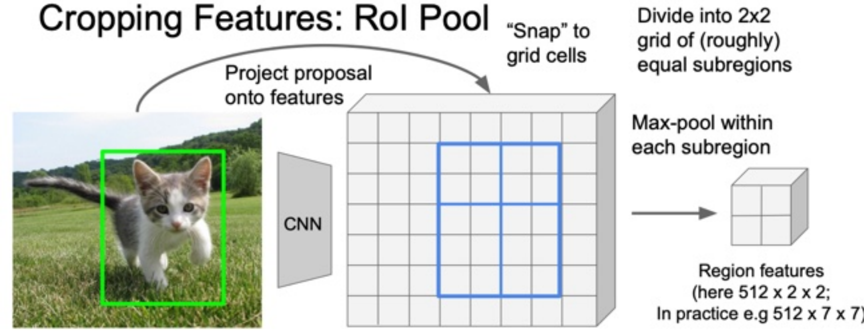
Q: how do we resize the 512 x 5 x 4 region to, e.g., a 512 x 2 x 2 tensor?.

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



# Cropping Features: RoI Pool

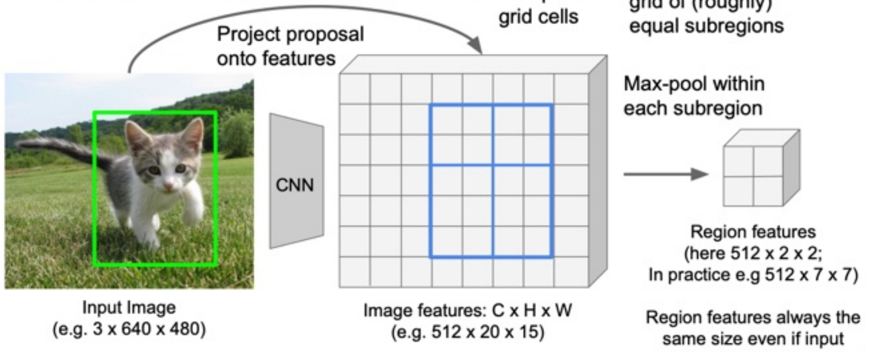
Divide into 2x2 grid of (roughly) equal subregions




Region features always the same size even if input regions have different sizes!

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

# Cropping Features: RoI Pool




Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

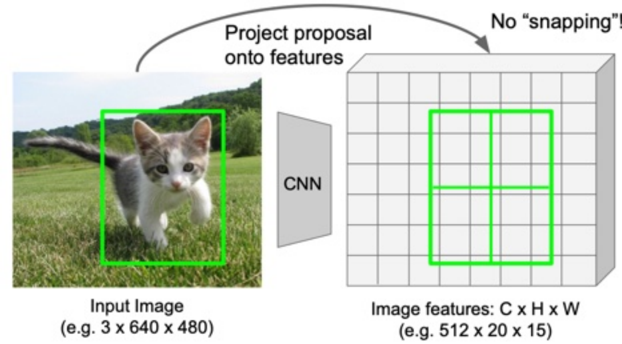
Region features always the same size even if input regions have different sizes!

**Problem**: Region features slightly misaligned

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



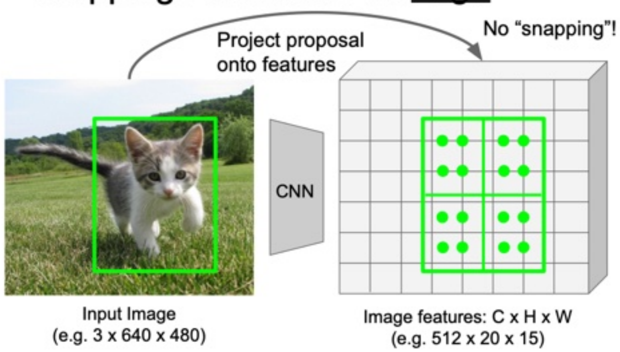
# Cropping Features: RoI Align




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



# Cropping Features: RoI Align



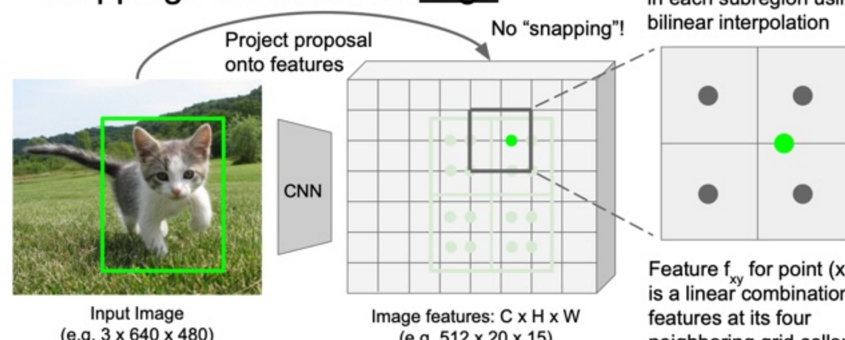

Sample at regular points in each subregion using bilinear interpolation

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



# Cropping Features: RoI Align

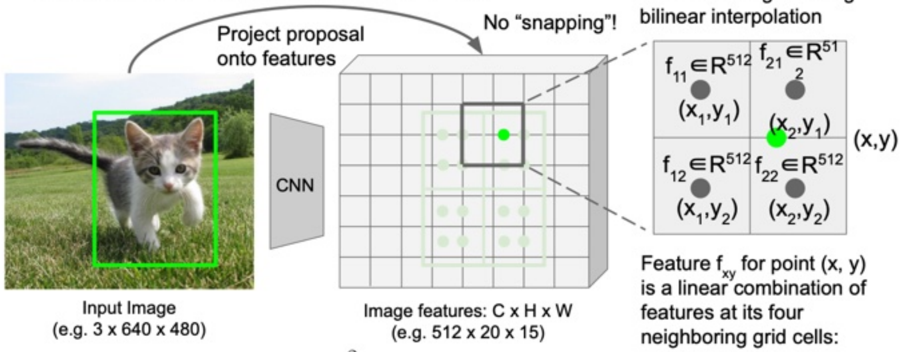
Sample at regular points in each subregion using bilinear interpolation




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

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

# Cropping Features: RoI Align



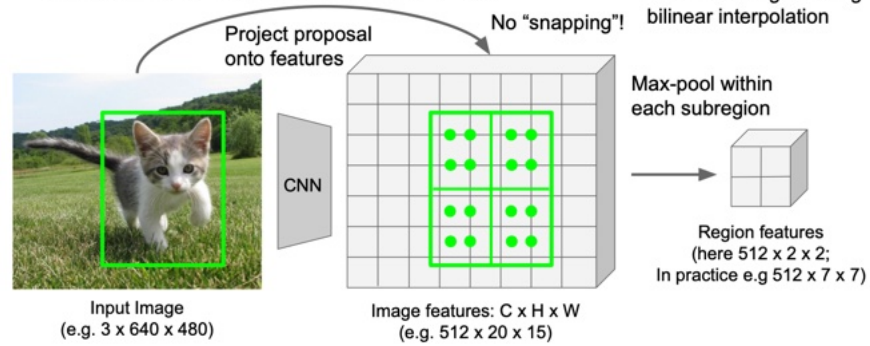

Feature $f_{xy}$ for point (x, y) is a linear combination of features at its four neighboring grid cells:

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

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



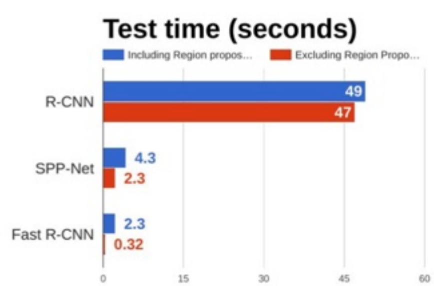
# Cropping Features: RoI Align




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



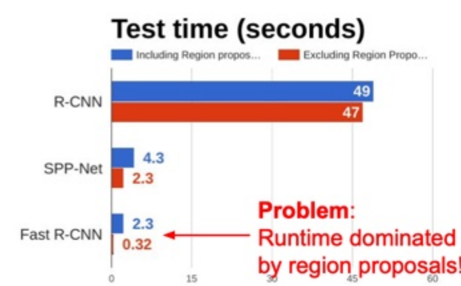
# R-CNN vs Fast R-CNN


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

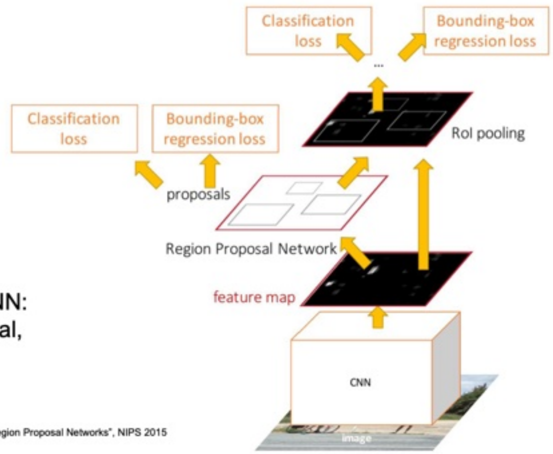




Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

# Faster R-CNN:
Make CNN do proposals!

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

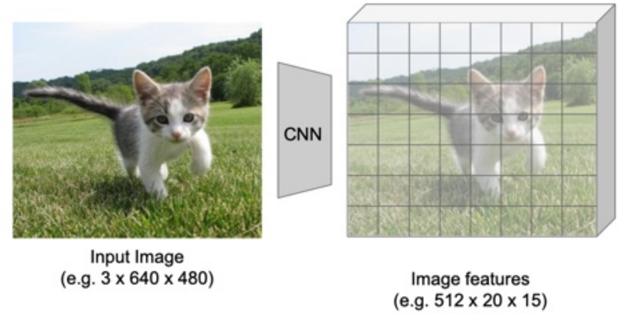
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one




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



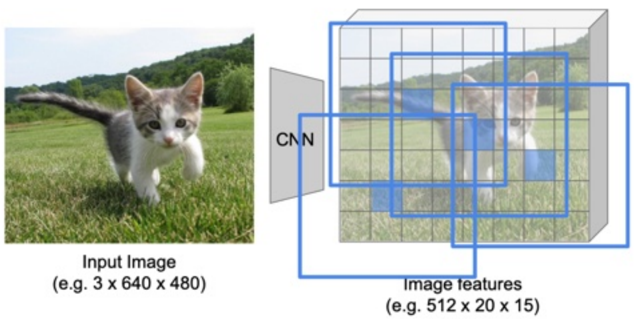
# Region Proposal Network






# Region Proposal Network

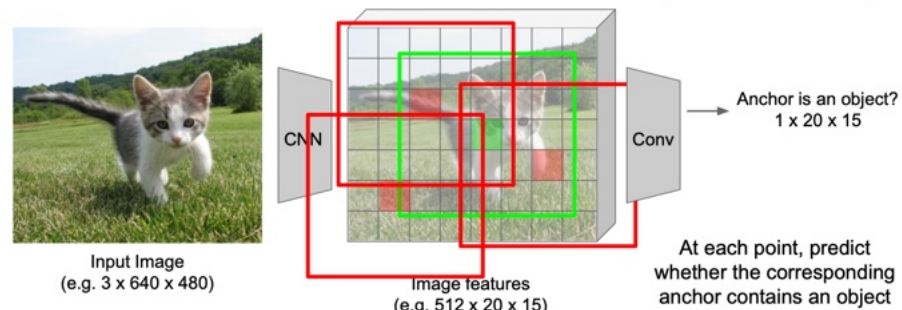
Imagine an **anchor box** of fixed size at each point in the feature map






# Region Proposal Network

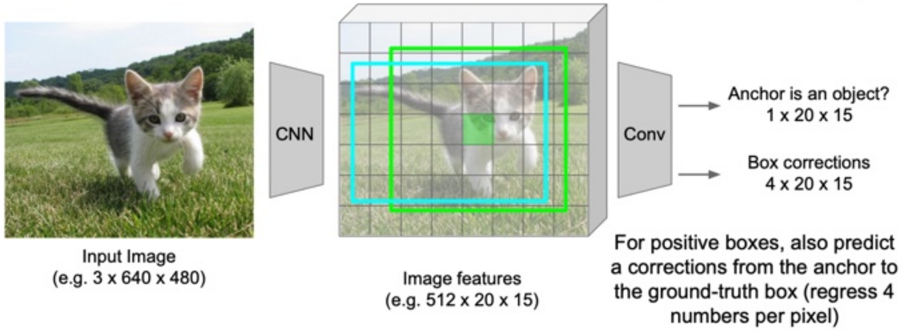
Imagine an **anchor box** of fixed size at each point in the feature map




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

## Region Proposal Network

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

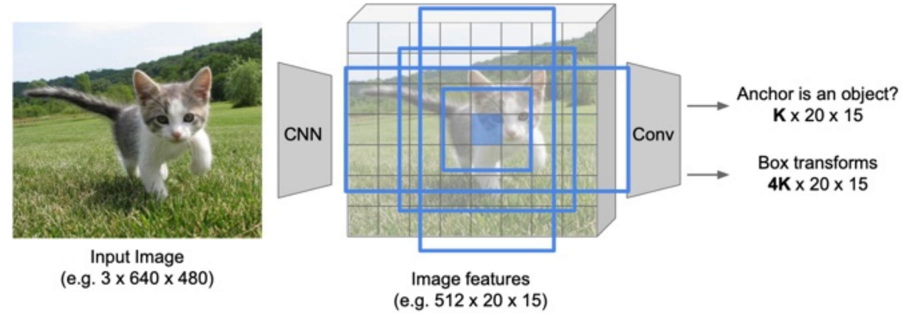



For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)



## Region Proposal Network

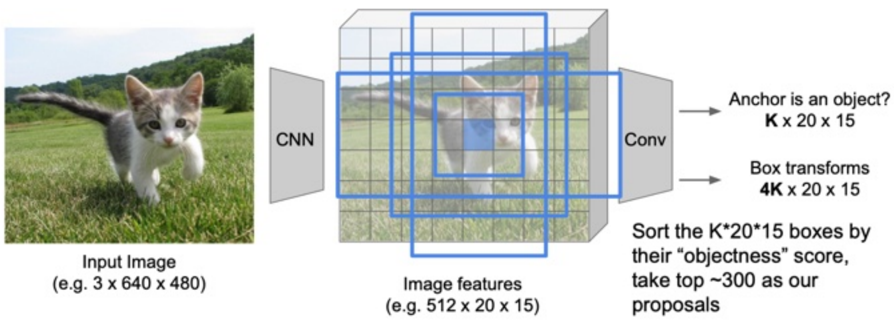
In practice use K different anchor boxes of different size / scale at each point






## Region Proposal Network

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




Sort the K*20*15 boxes by their "objectness" score, take top ~300 as our proposals

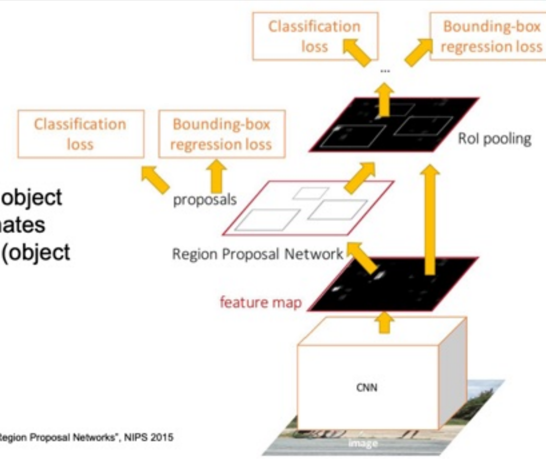


## Fast**er** R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

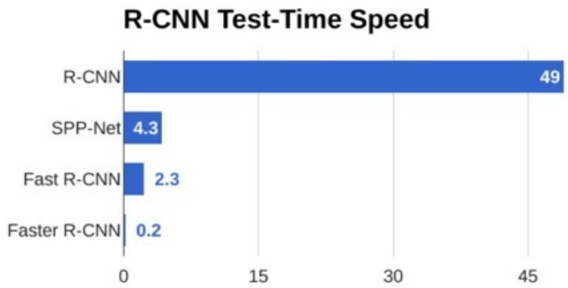
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates




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

# Fast**er** R-CNN:

Make CNN do proposals!

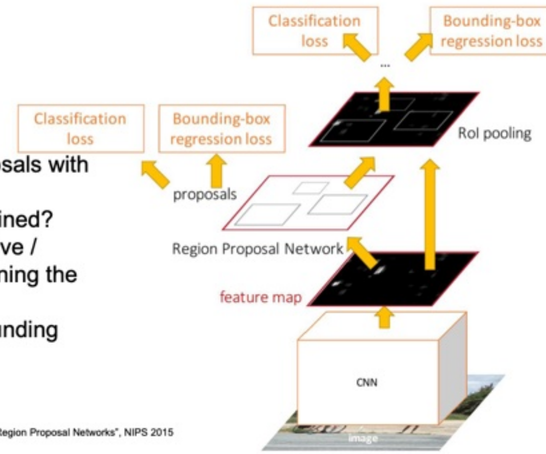





# Fast**er** R-CNN:

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?




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



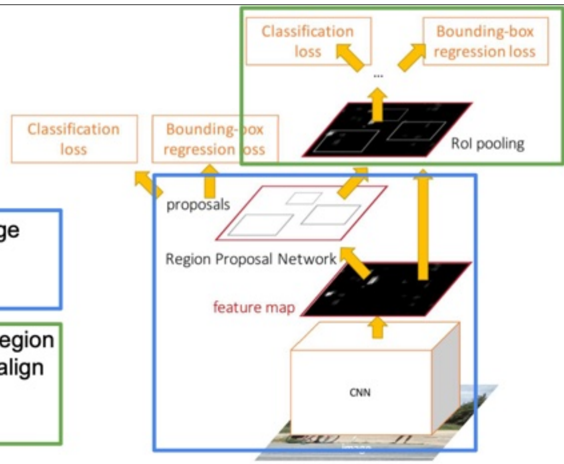
# Fast**er** R-CNN:

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






# Fast**er** R-CNN:

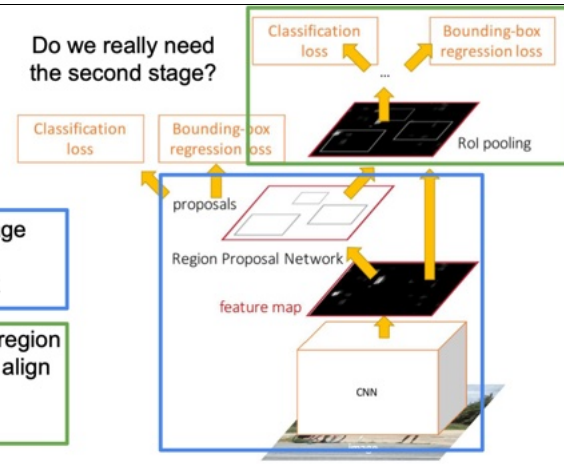
Make CNN do proposals!

Do we really need the second stage?

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

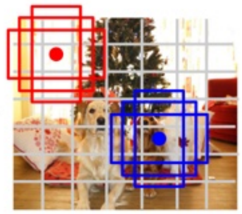
First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



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

Input image
3 x H x W



Divide image into grid
7 x 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)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (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




**Instance Segmentation**

DOG, DOG, CAT

Multiple Object

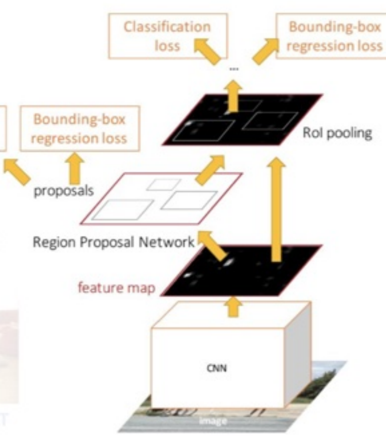


## Object Detection: Faster R-CNN

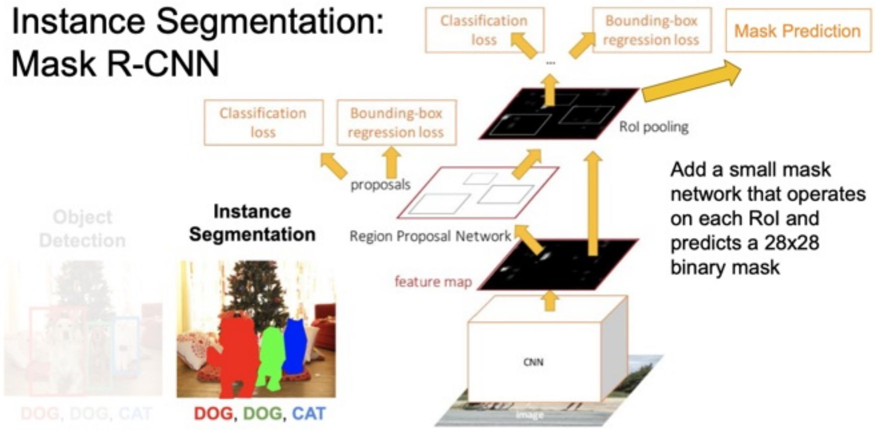
**Object Detection**

DOG, DOG, CAT



# Instance Segmentation: Mask R-CNN

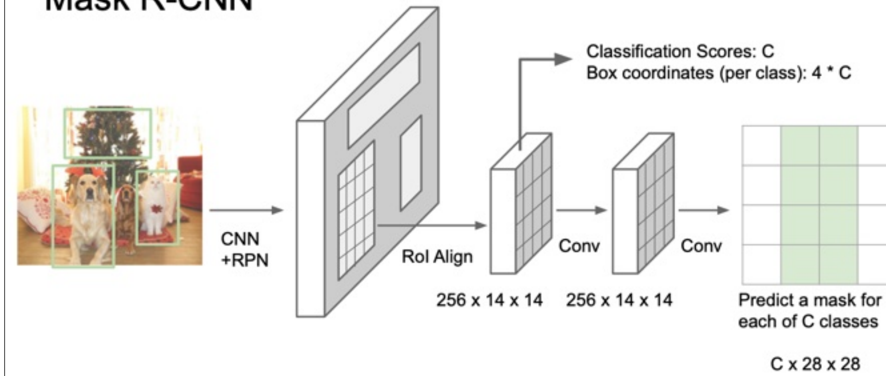



Add a small mask network that operates on each RoI and predicts a 28x28 binary mask

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



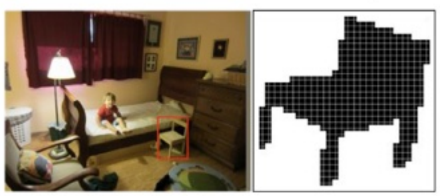
# Mask R-CNN




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

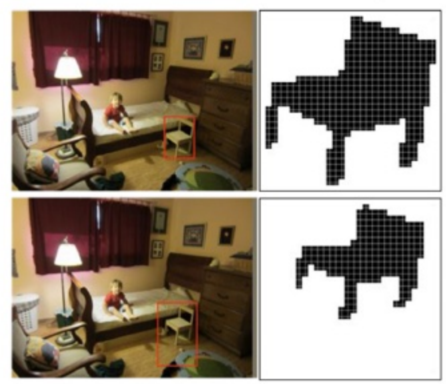


# Mask R-CNN: Example Mask Training Targets

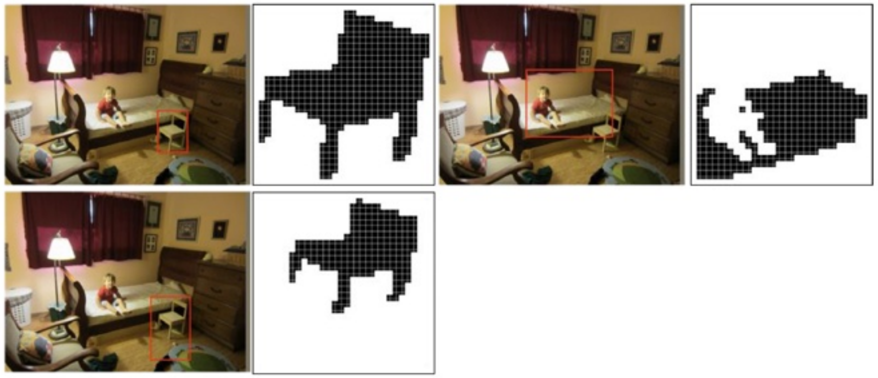




# Mask R-CNN: Example Mask Training Targets

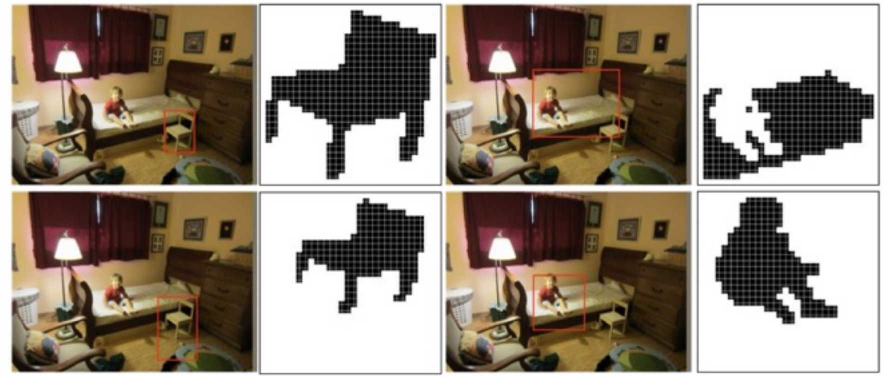


## Mask R-CNN: Example Mask Training Targets



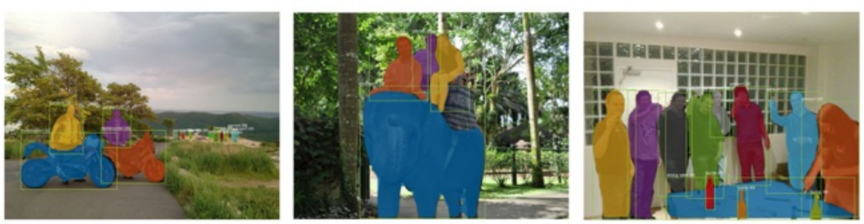


## Mask R-CNN: Example Mask Training Targets





## Mask R-CNN: Very Good Results!



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



## Mask R-CNN
## Also does pose

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

# 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, ...

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



# Beyond 2D Object Detection...

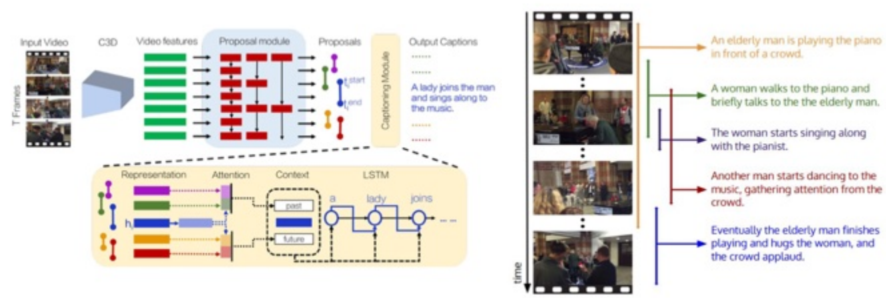


# Object Detection + Captioning = Dense Captioning


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

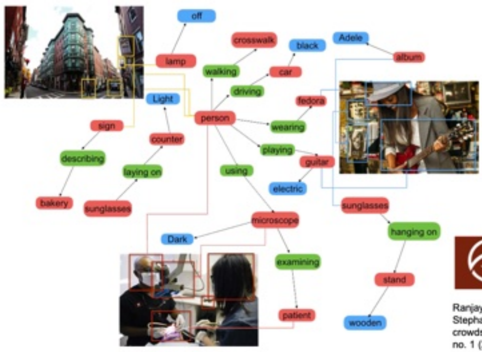



# Dense Video Captioning




Ranjay Krishna et al., "Dense-Captioning Events in Videos", ICCV 2017

# Objects + Relationships = Scene Graphs



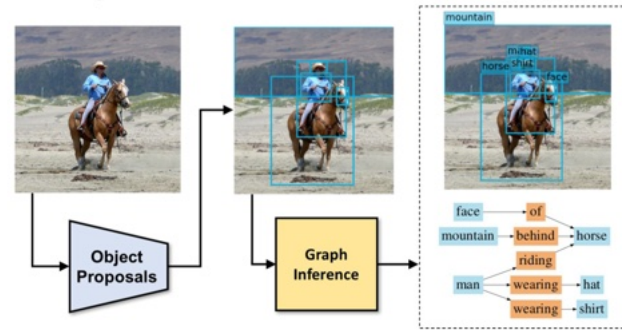

108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets

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.



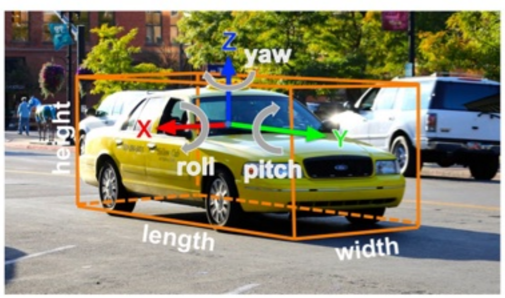
# Scene Graph Prediction




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




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)

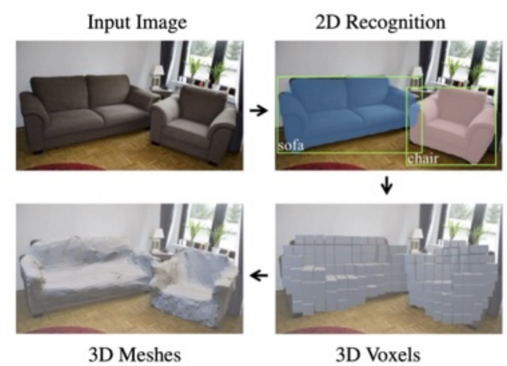
Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

This image is CC0 public domain



# 3D Shape Prediction: Mesh R-CNN




Gkioxari et al., Mesh RCNN, ICCV 2019