

Neurónové siete pre počítačové videnie

9 Detekcia a segmentácia

RNDr. Zuzana Černeková, PhD.

Ing. Viktor Kocur, PhD.

So far: Image Classification



This image is CC0 public domain

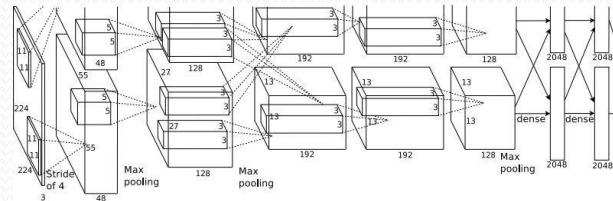


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

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

Computer Vision Tasks

Classification



CAT

No spatial extent

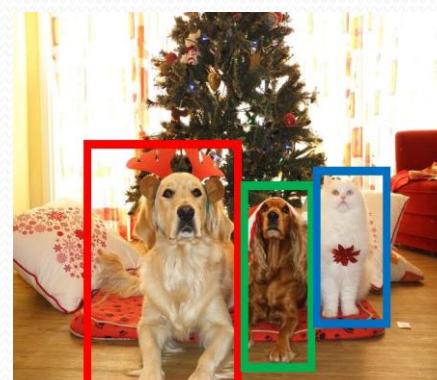
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

Semantic Segmentation

Classification

CAT

No spatial extent

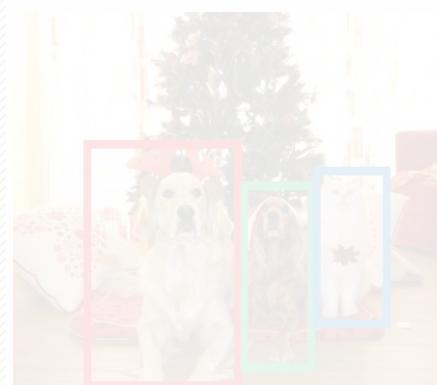
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object
Detection



DOG, DOG, CAT

Instance
Segmentation



DOG, DOG, CAT

Multiple Object

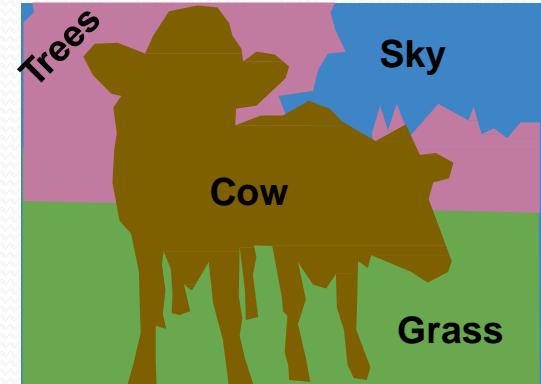
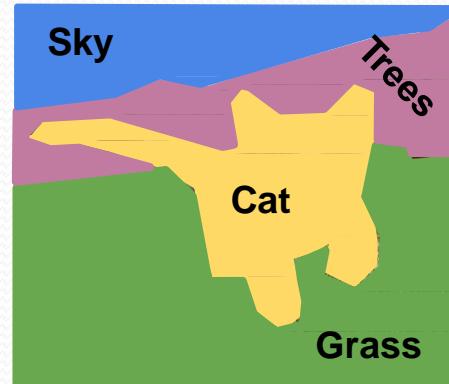
Semantic Segmentation

Label each pixel in the image with a category label

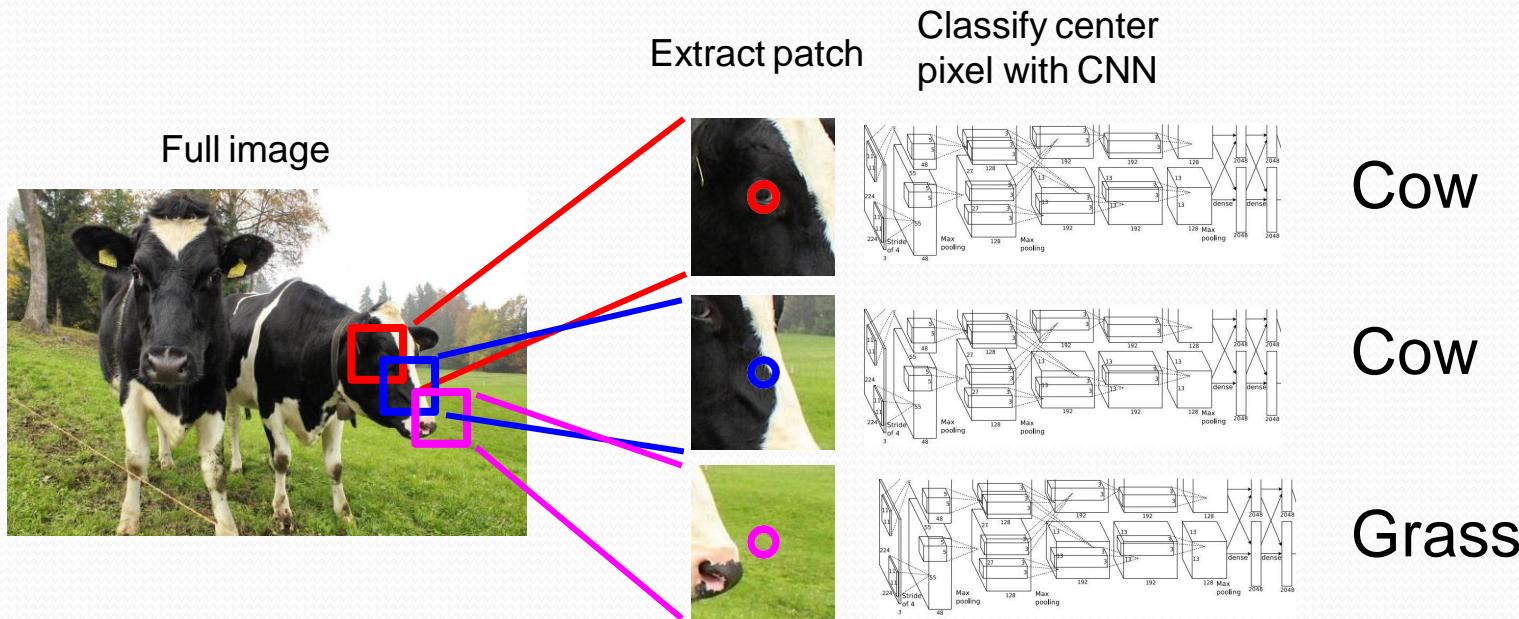
Don't differentiate instances, only care about pixels



[This image is CC0 public domain](#)

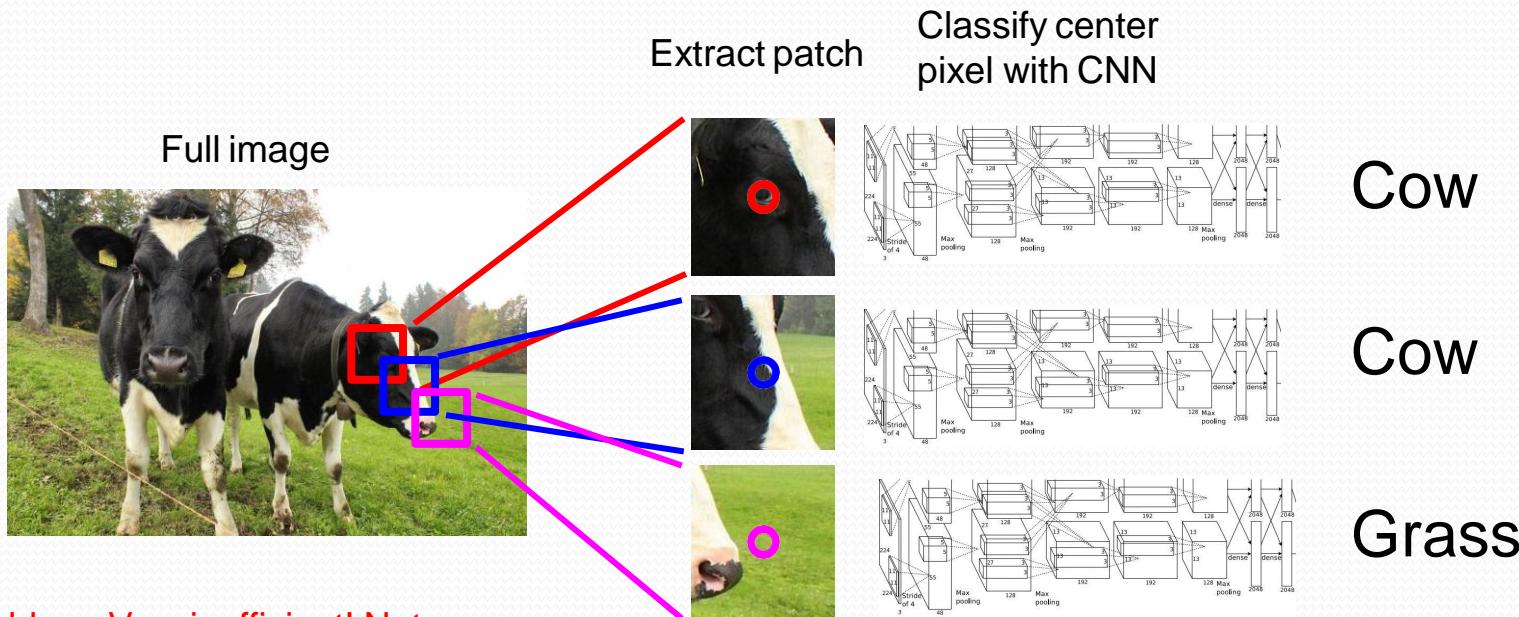


Semantic Segmentation Idea: Sliding Window



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

Semantic Segmentation Idea: Sliding Window

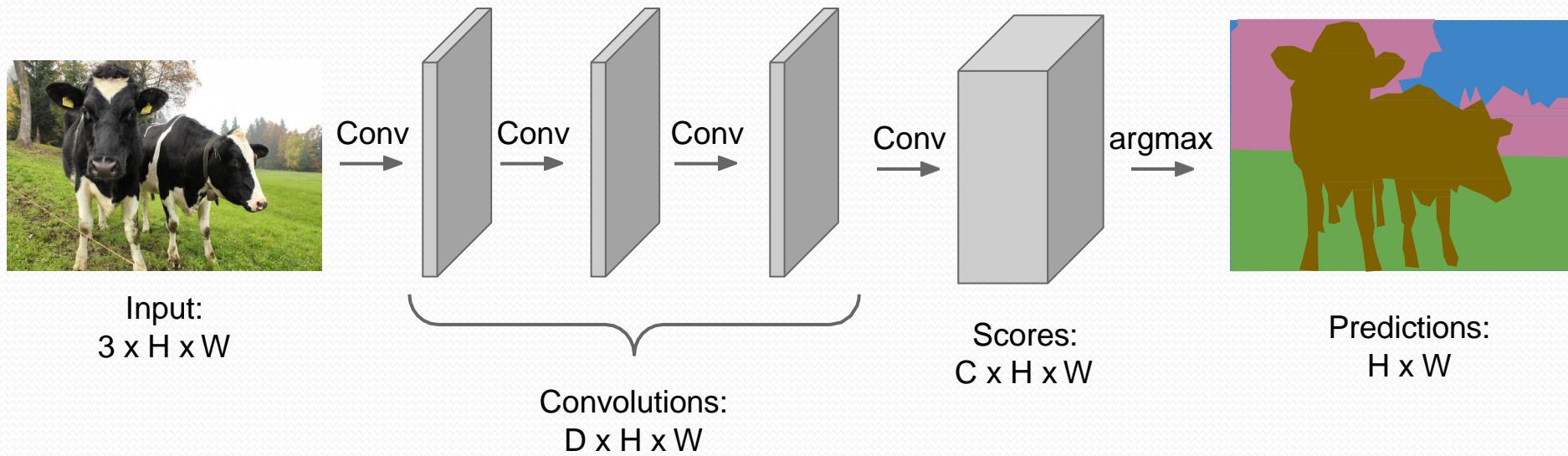


Problem: Very inefficient! Not reusing shared features between overlapping patches

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

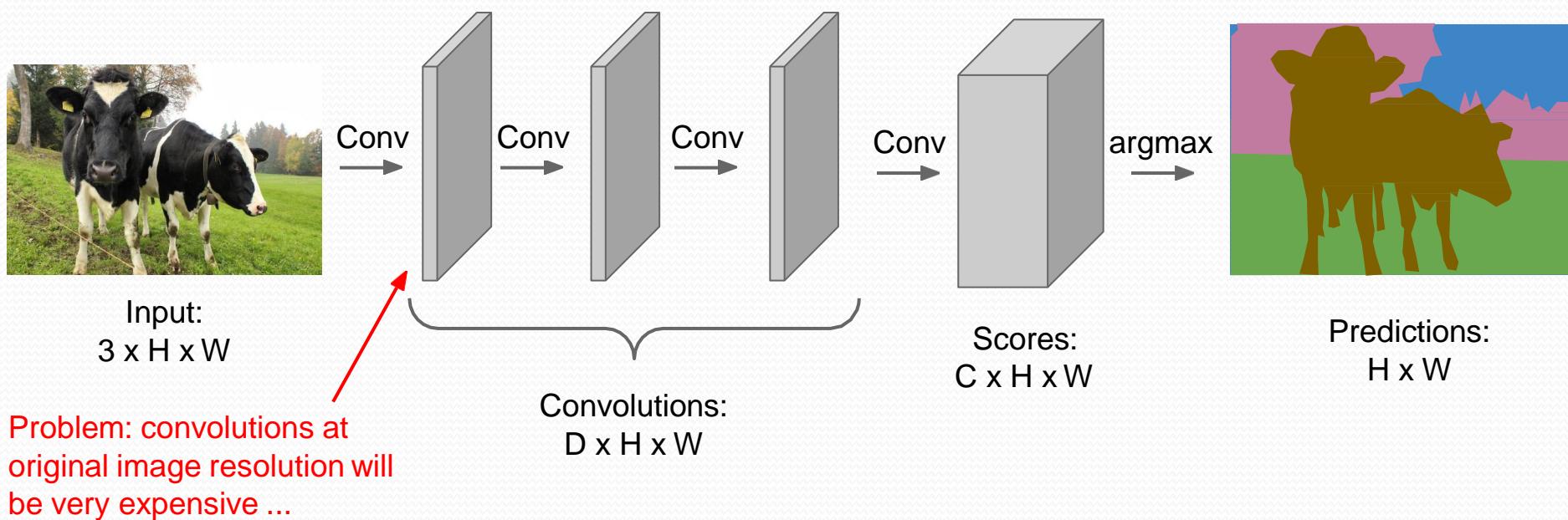
Semantic Segmentation Idea: Fully Convolutional

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



Semantic Segmentation Idea: Fully Convolutional

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

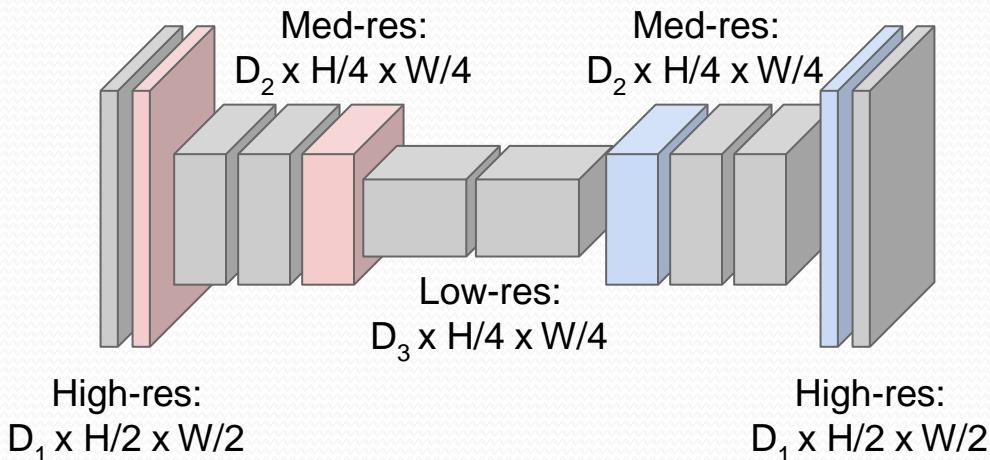


Semantic Segmentation Idea: Fully Convolutional

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



Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

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

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

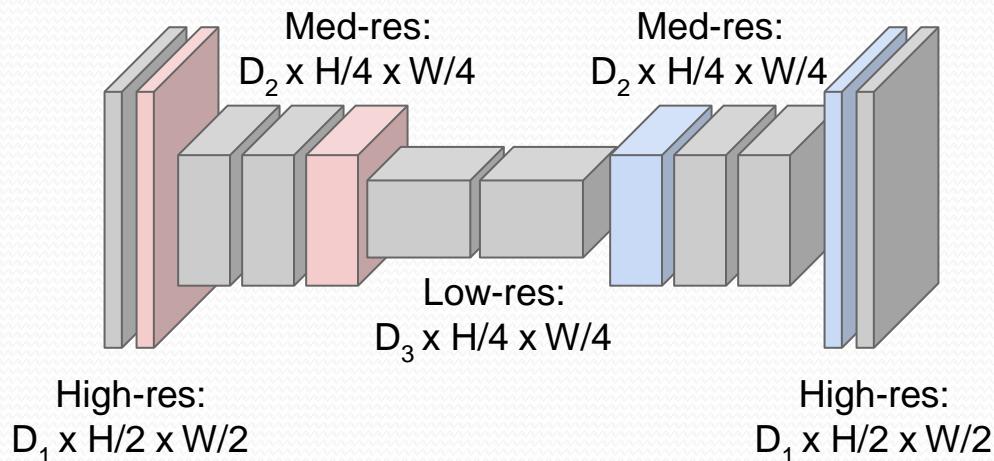
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution

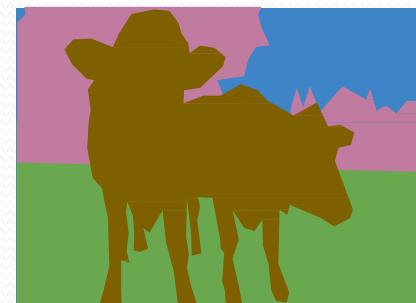


Input:
 $3 \times H \times W$

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



Upsampling:
???



Predictions:
 $H \times W$

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

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network upsampling: “Unpooling”

Nearest Neighbor

1	2
3	4



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

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

1	2
3	4



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

Input: 2 x 2

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

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

Max Unpooling

Use positions from pooling layer

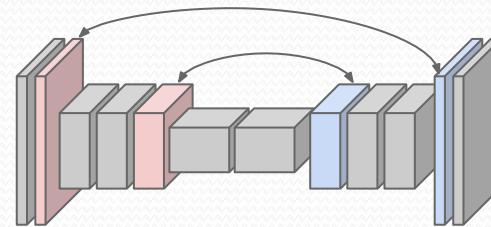
1	2
3	4

Rest of the network

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

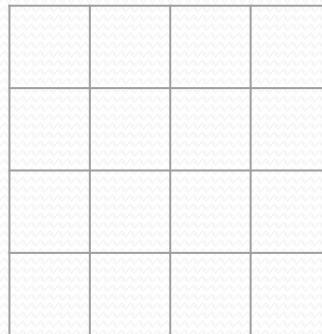
Output: 4 x 4

Corresponding pairs of
downsampling and
upsampling layers

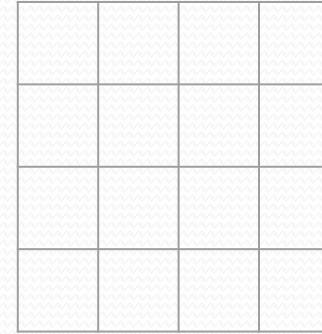


Learnable Upsampling: Transpose Convolution

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



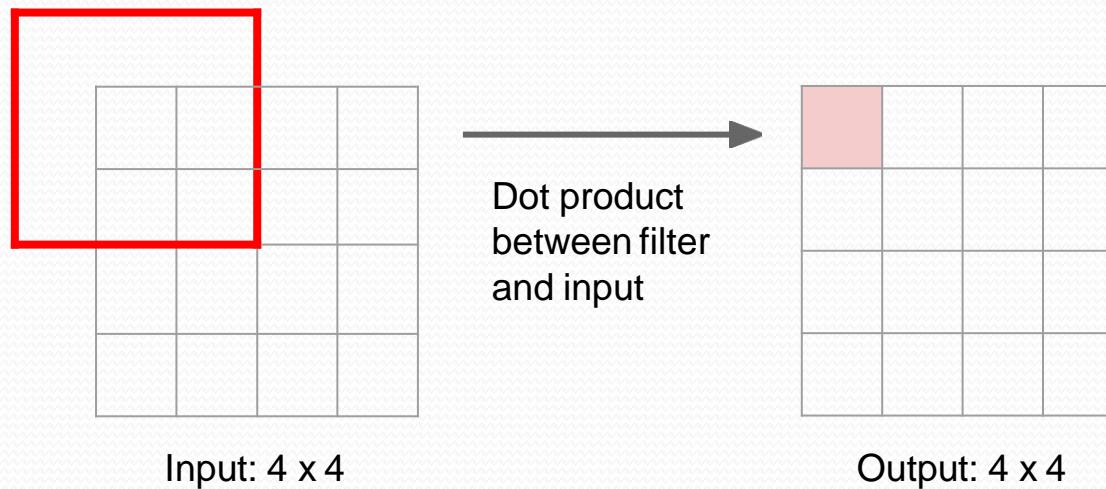
Input: 4×4



Output: 4×4

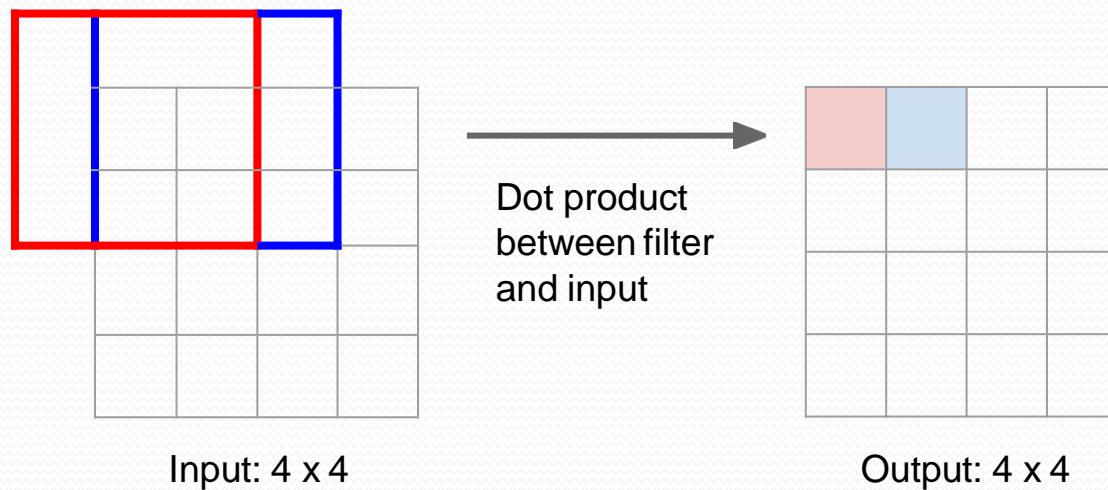
Learnable Upsampling: Transpose Convolution

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



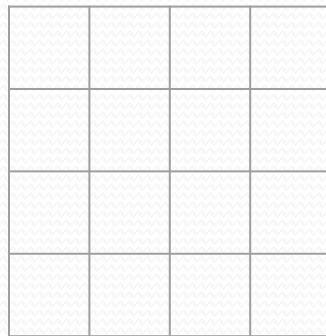
Learnable Upsampling: Transpose Convolution

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

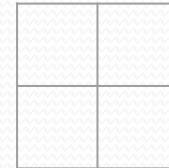


Learnable Upsampling: Transpose Convolution

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



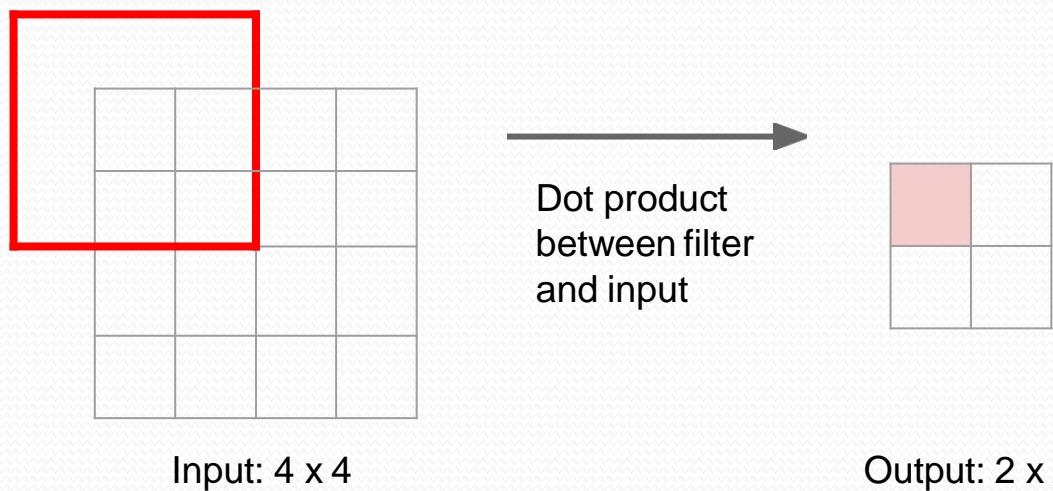
Input: 4×4



Output: 2×2

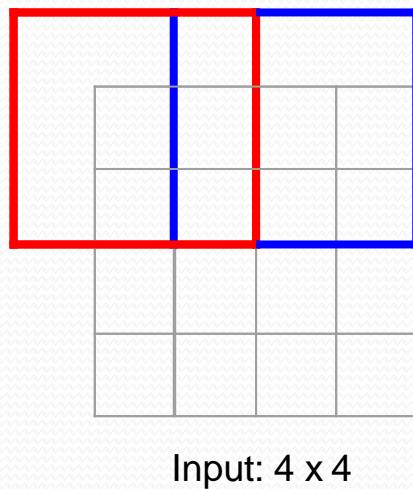
Learnable Upsampling: Transpose Convolution

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

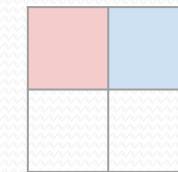


Learnable Upsampling: Transpose Convolution

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



Dot product
between filter
and input

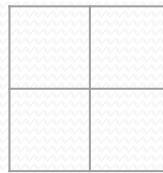


Filter moves 2 pixels in
the input for every one
pixel in the output

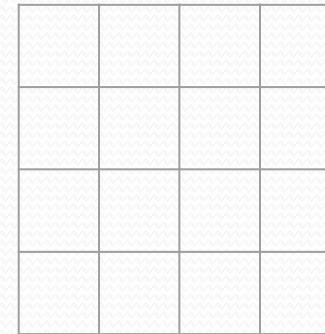
Stride gives ratio between
movement in input and
output

Learnable Upsampling: Transpose Convolution

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



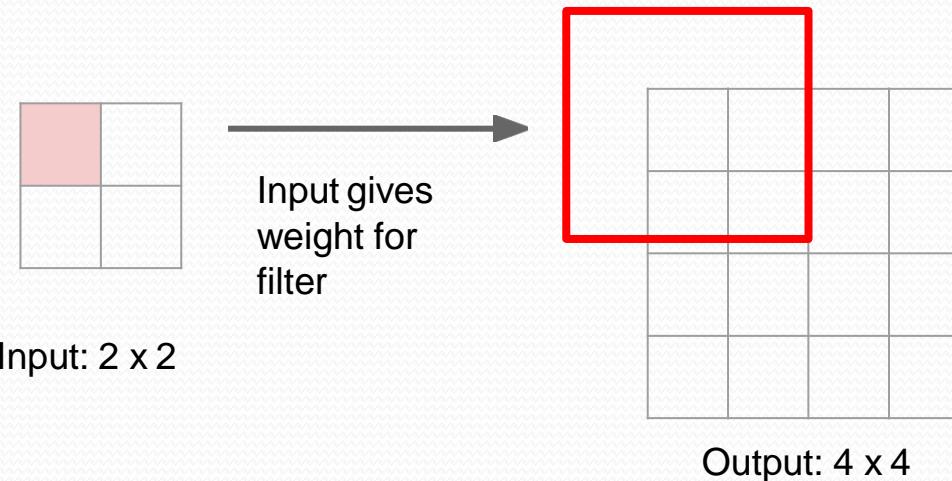
Input: 2 x 2



Output: 4 x 4

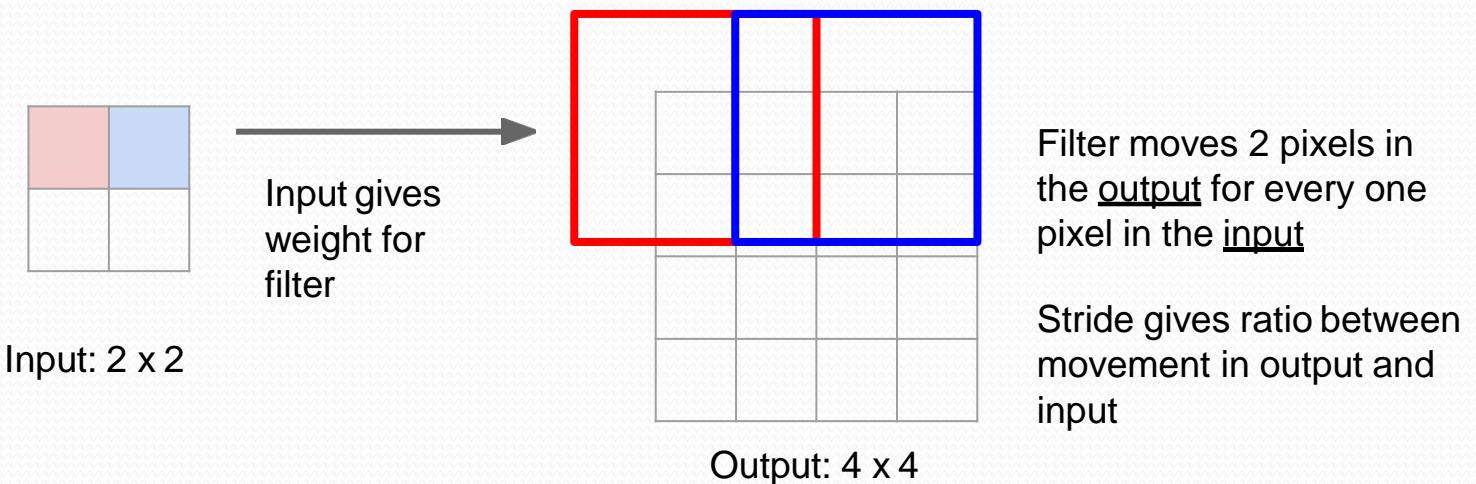
Learnable Upsampling: Transpose Convolution

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



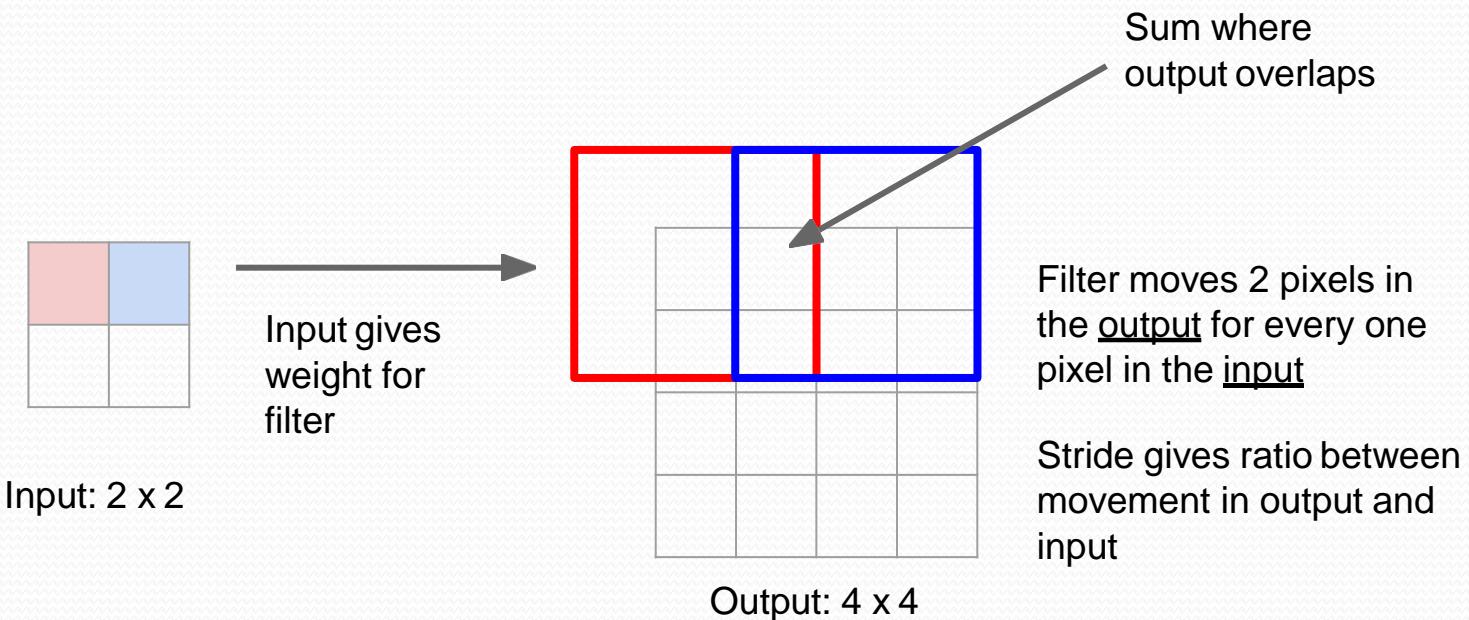
Learnable Upsampling: Transpose Convolution

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



Learnable Upsampling: Transpose Convolution

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

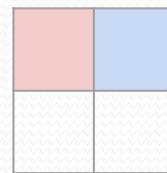


Learnable Upsampling: Transpose Convolution

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

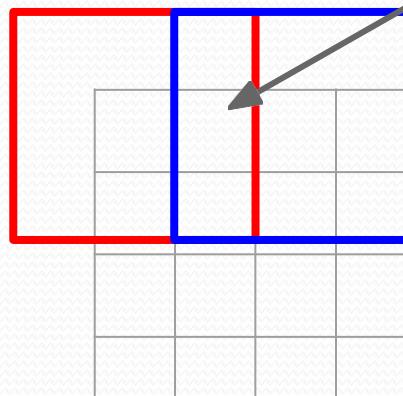
Other names:

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



Input: 2 x 2

Input gives weight for filter



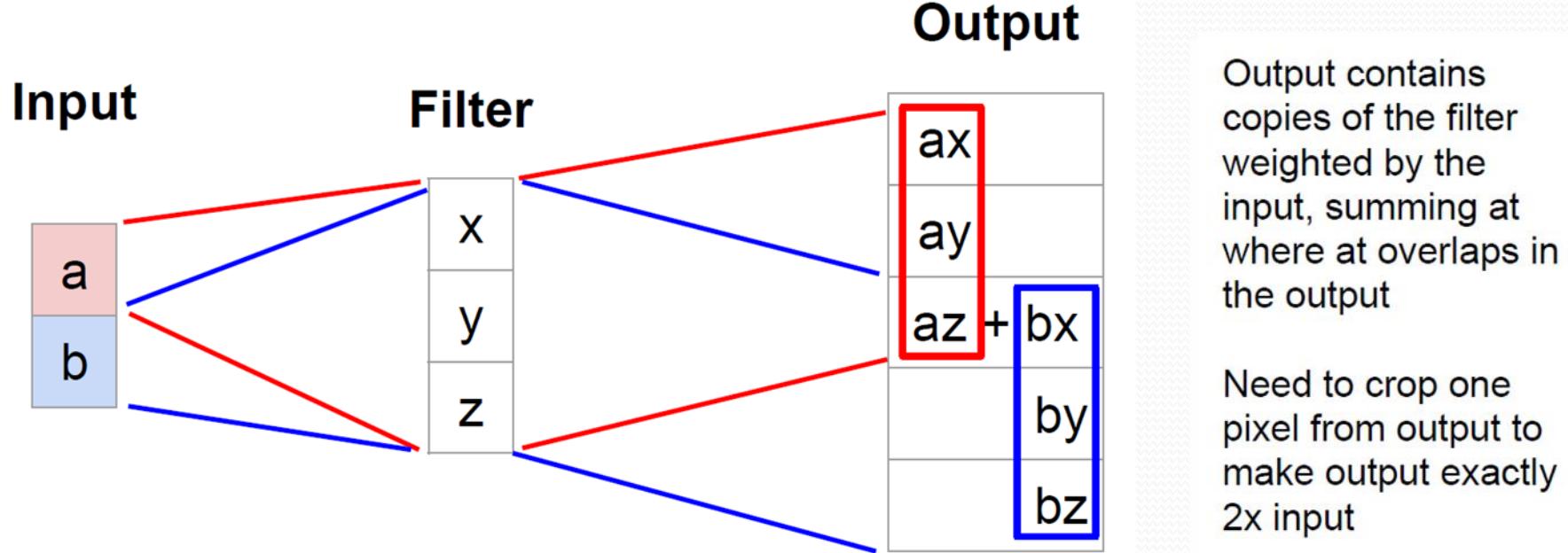
Output: 4 x 4

Sum where output overlaps

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: Transpose Convolution



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

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

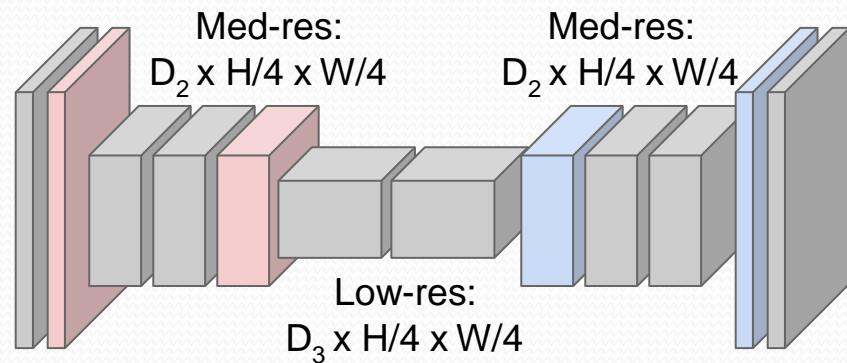
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

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



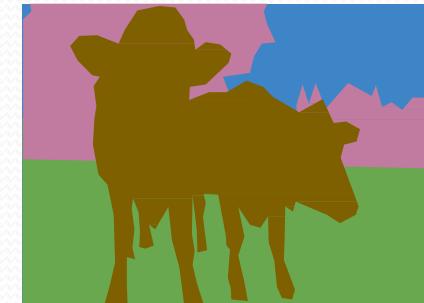
High-res:
 $D_1 \times H/2 \times W/2$

Med-res:
 $D_2 \times H/4 \times W/4$

Low-res:
 $D_3 \times H/4 \times W/4$

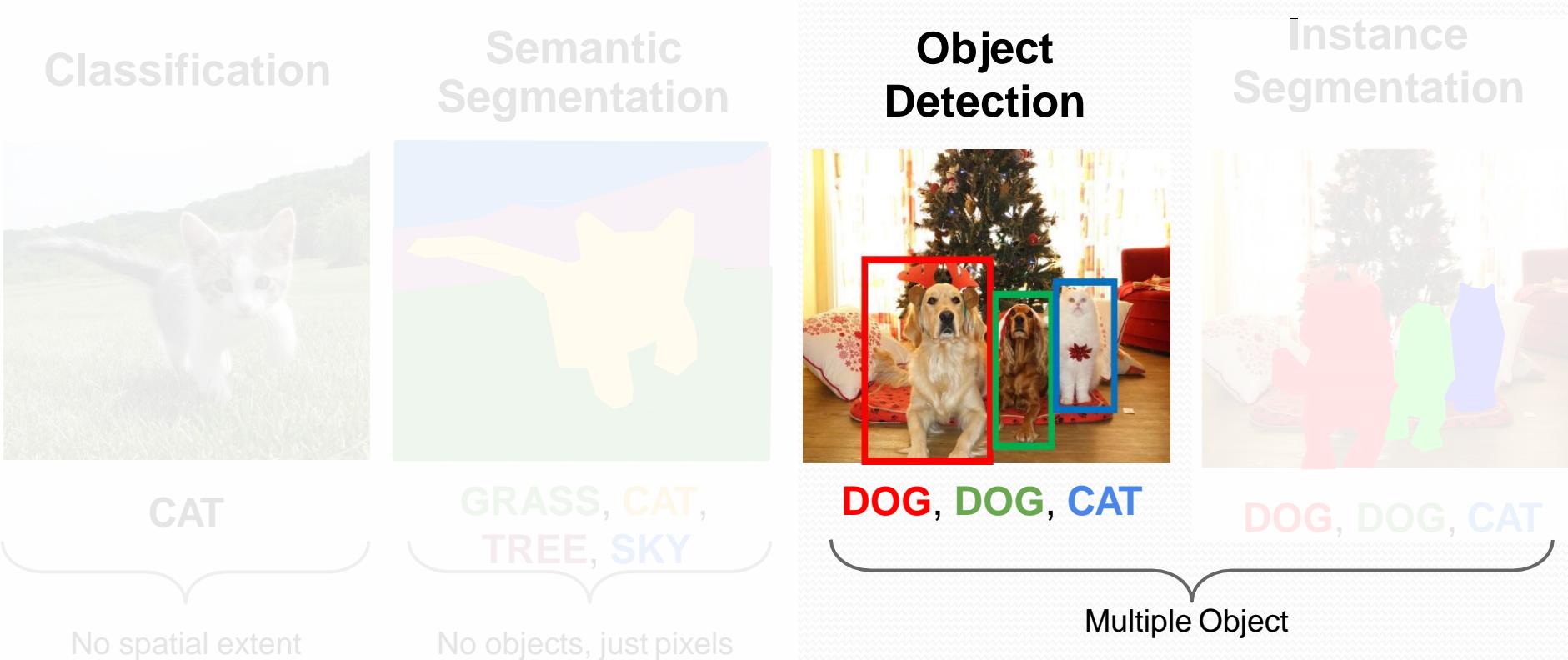
High-res:
 $D_1 \times H/2 \times W/2$

Upsampling:
Unpooling or strided
transpose convolution

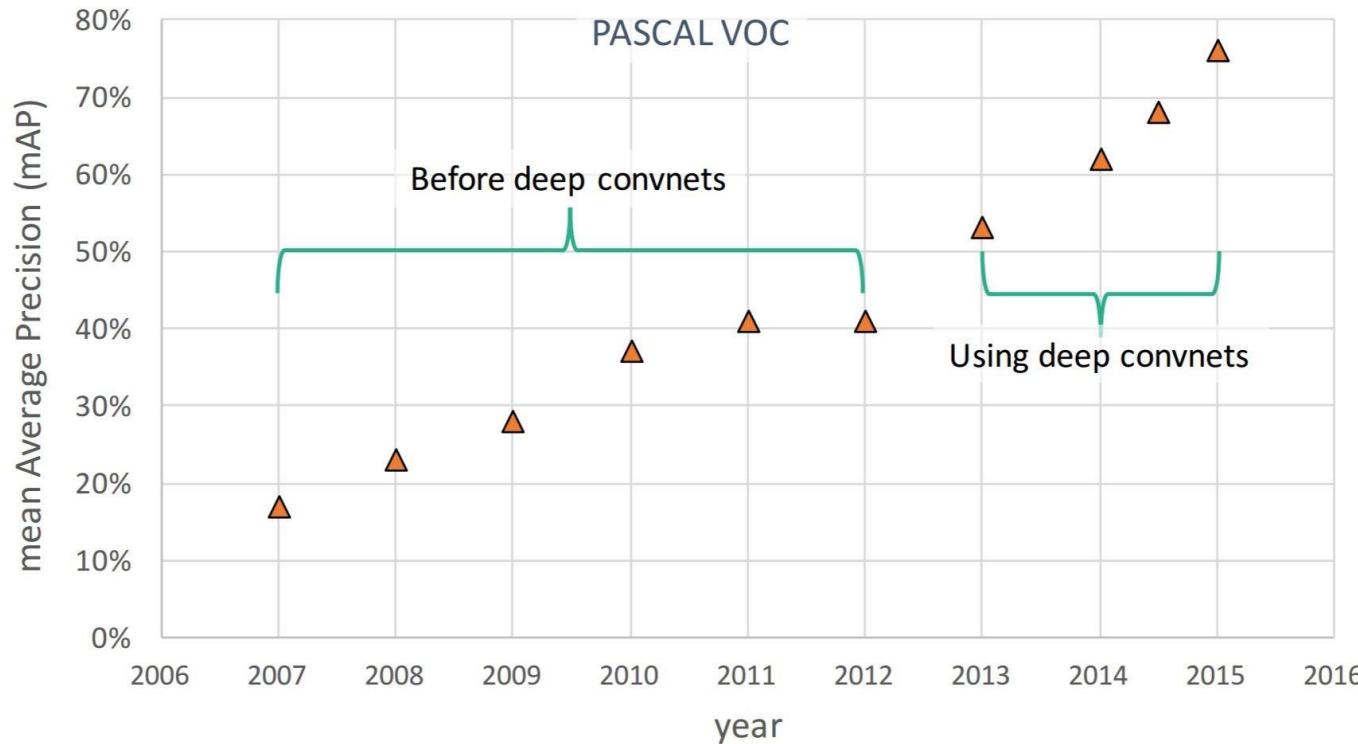


Predictions:
 $H \times W$

Object Detection



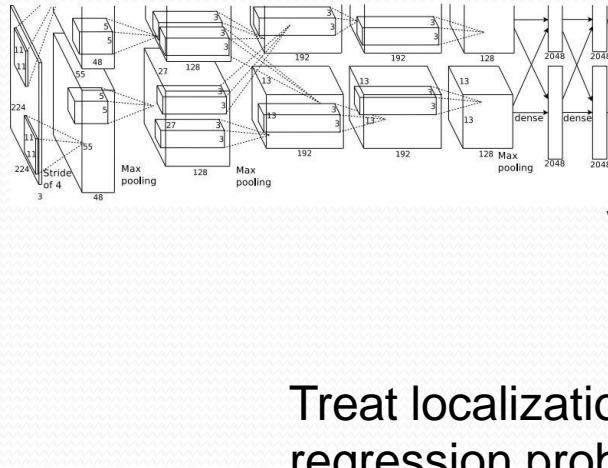
Object Detection: Impact of Deep Learning



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
...

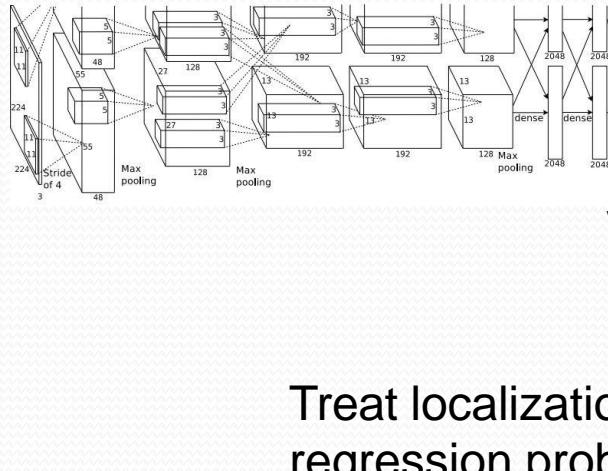
**Box
Coordinates**
(x, y, w, h)

Treat localization as a
regression problem!

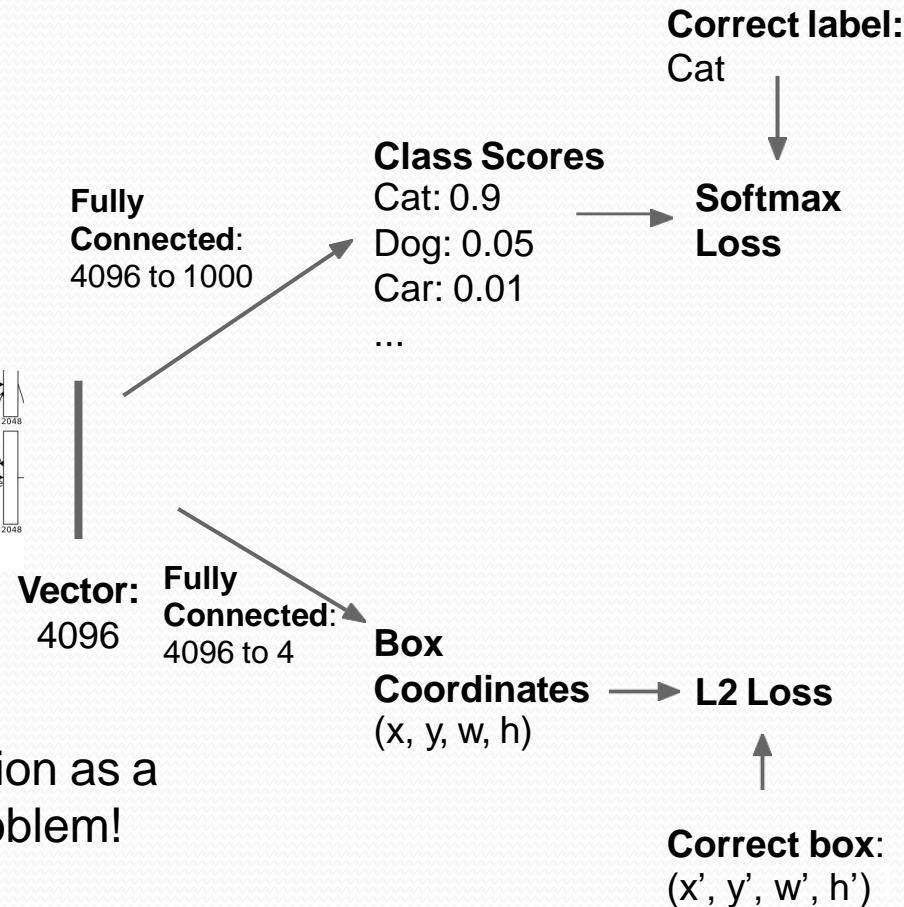
Object Detection: Single Object (Classification + Localization)



This image is CC0 public domain.



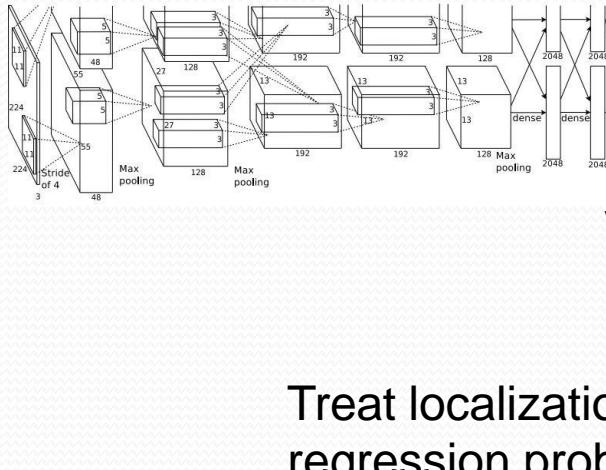
Treat localization as a
regression problem!



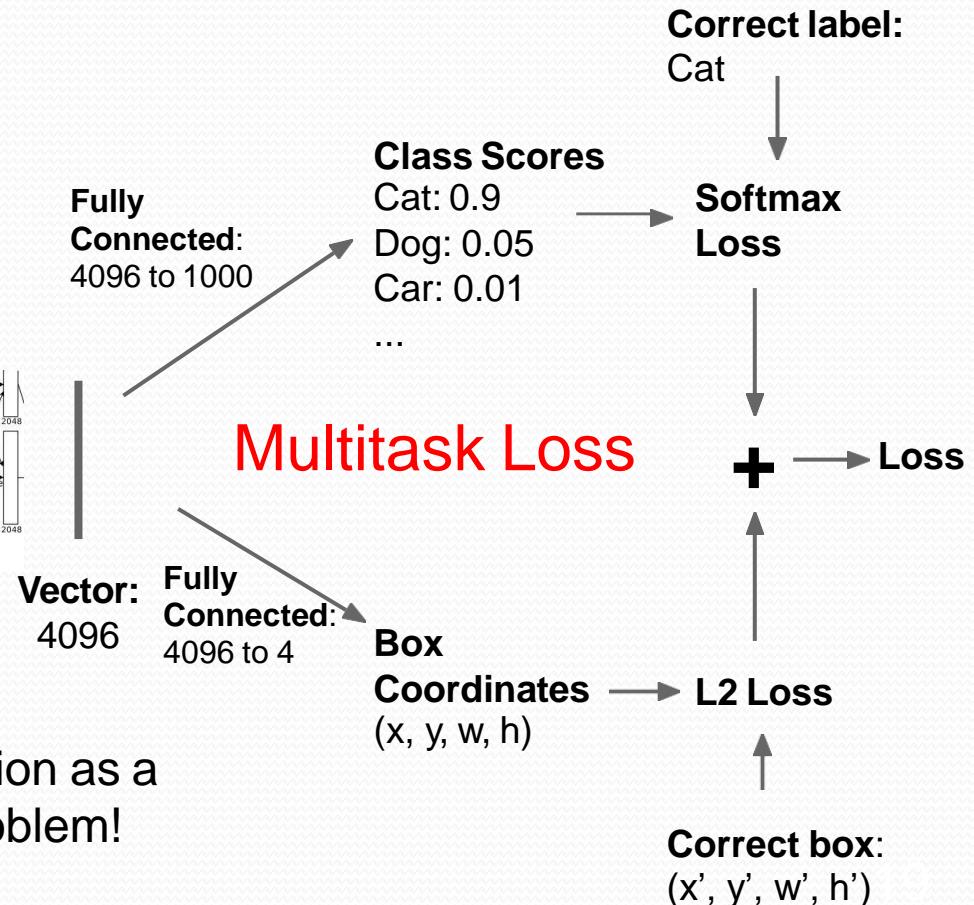
Object Detection: Single Object (Classification + Localization)



This image is CC0 public domain.



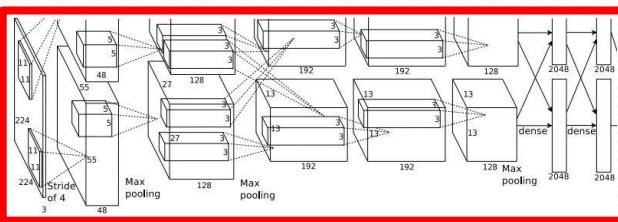
Treat localization as a
regression problem!



Object Detection: Single Object (Classification + Localization)

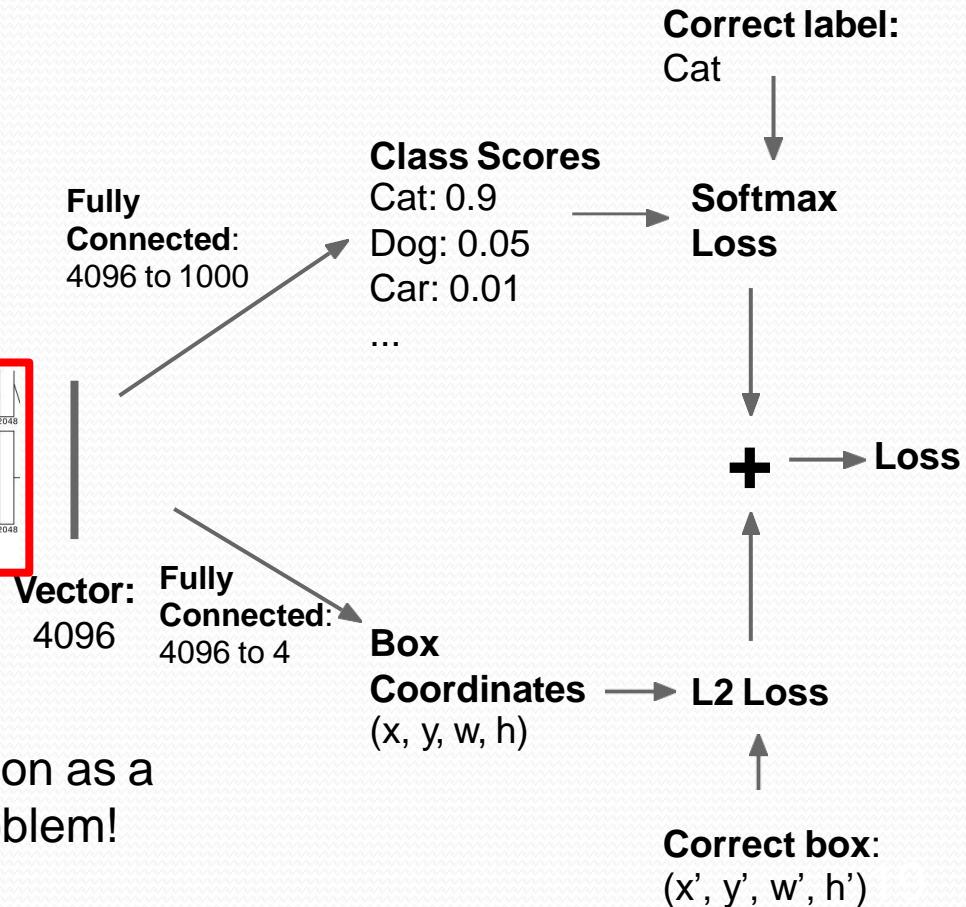


This image is CC0 public domain.

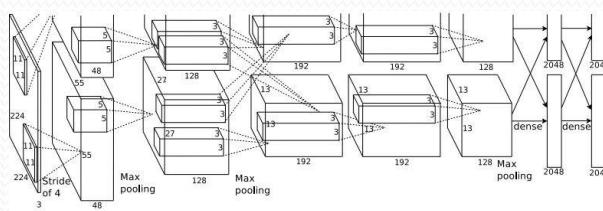


Often pretrained on ImageNet
(Transfer learning)

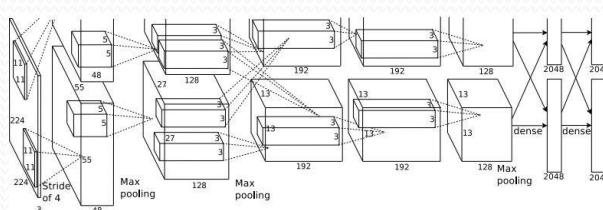
Treat localization as a
regression problem!



Object Detection: Multiple Objects



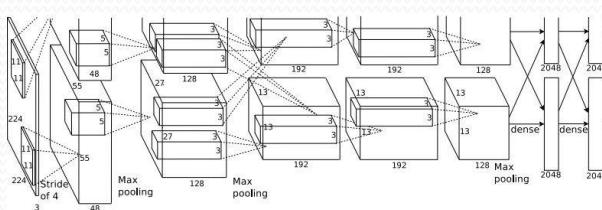
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



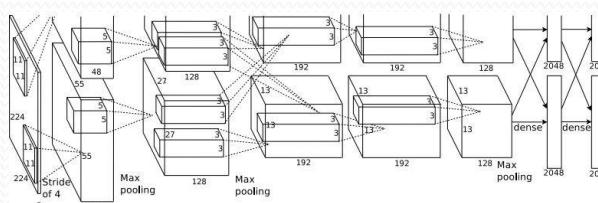
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

...

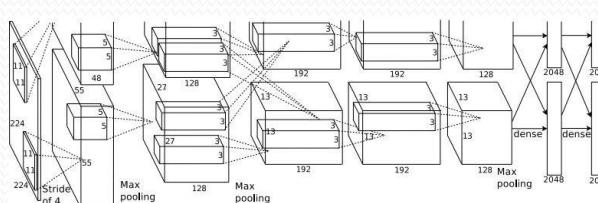
Object Detection: Multiple Objects

Each image needs a different number of outputs!



CAT: (x, y, w, h)

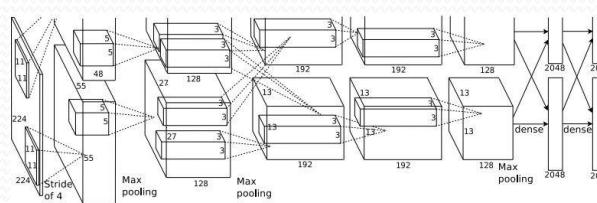
4 numbers



DOG: (x, y, w, h)

12 numbers

CAT: (x, y, w, h)



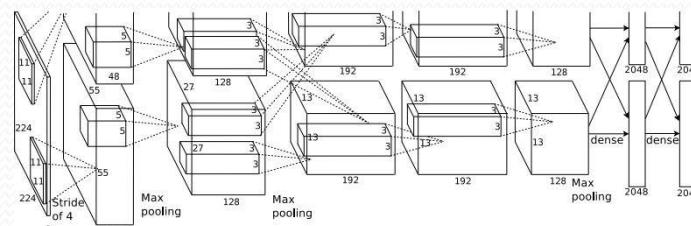
DUCK: (x, y, w, h)

Many numbers!

...

Object Detection: Multiple Objects

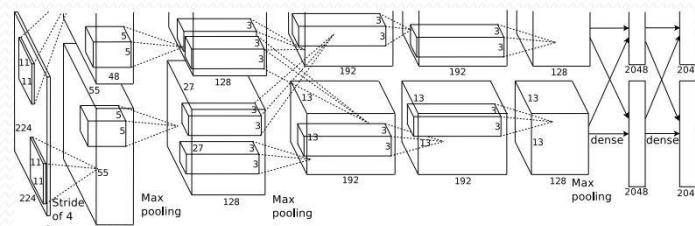
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

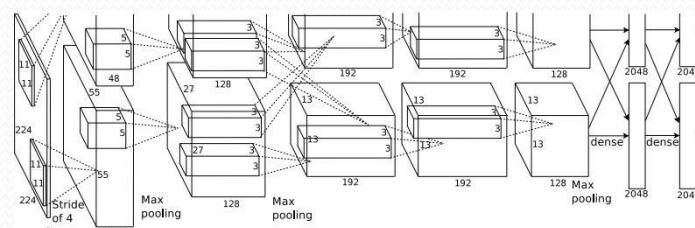
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

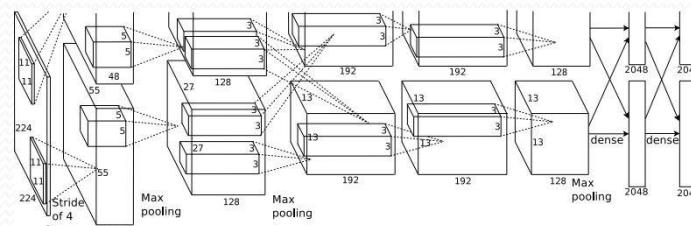
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

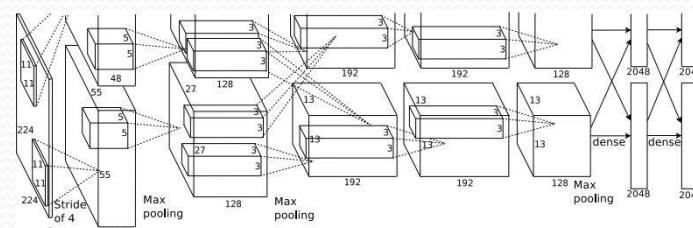
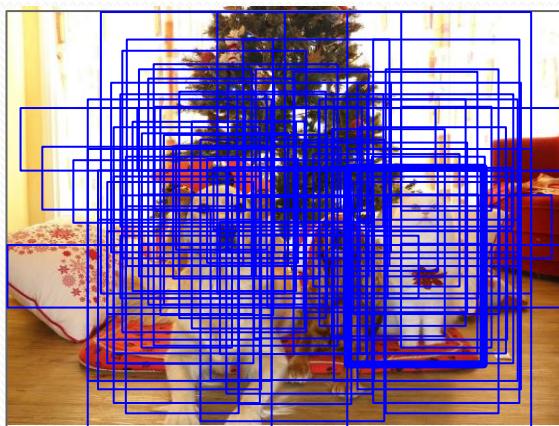
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Object Detection: Multiple Objects

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

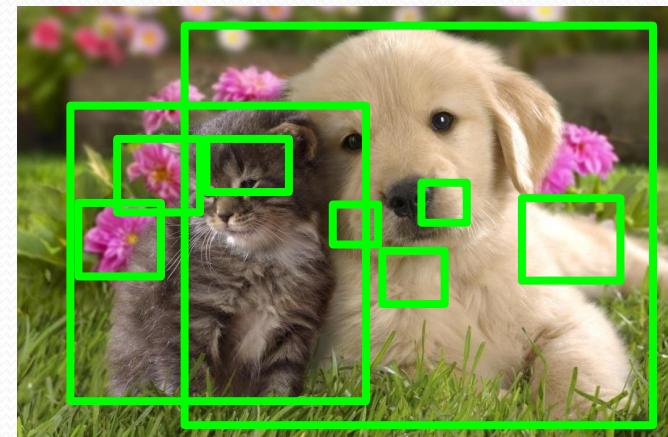


Dog? NO
Cat? YES
Background? NO

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

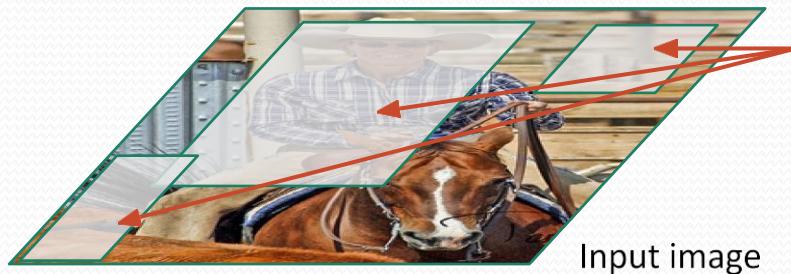
R-CNN



Input image

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

R-CNN

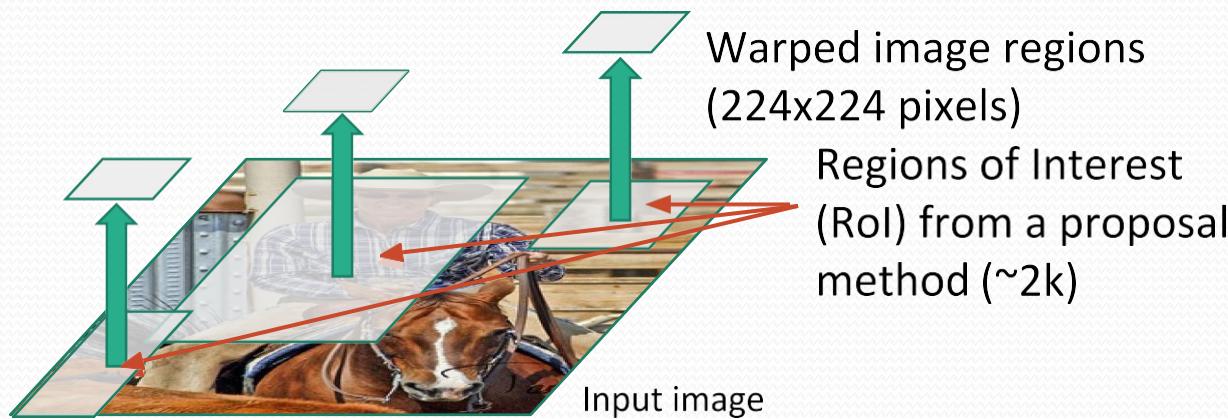


Input image

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

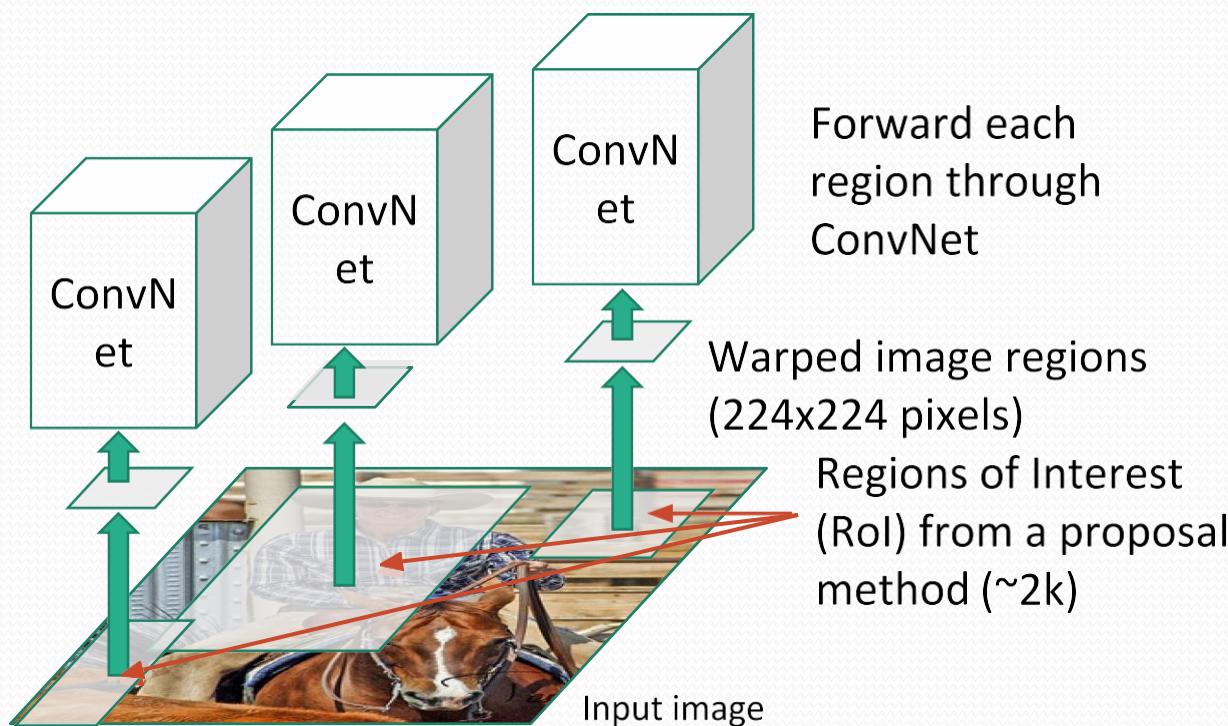
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



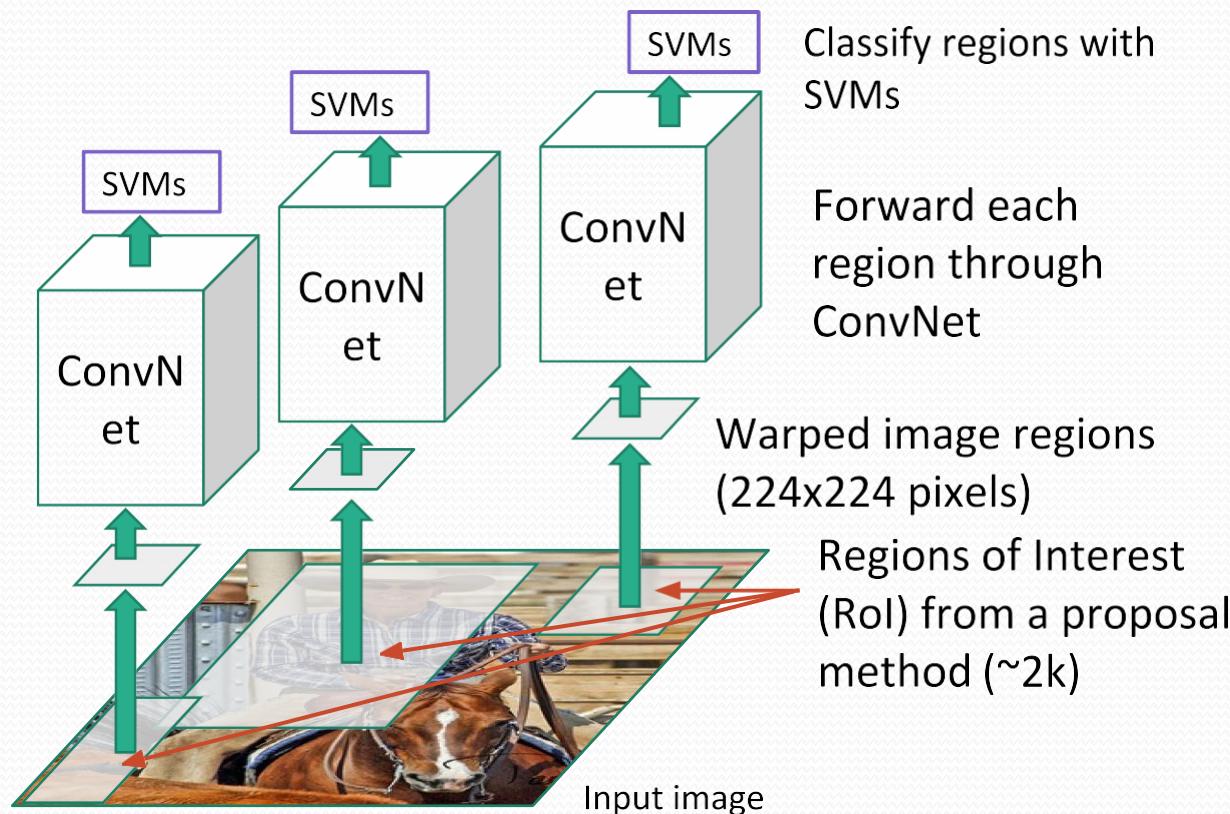
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



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

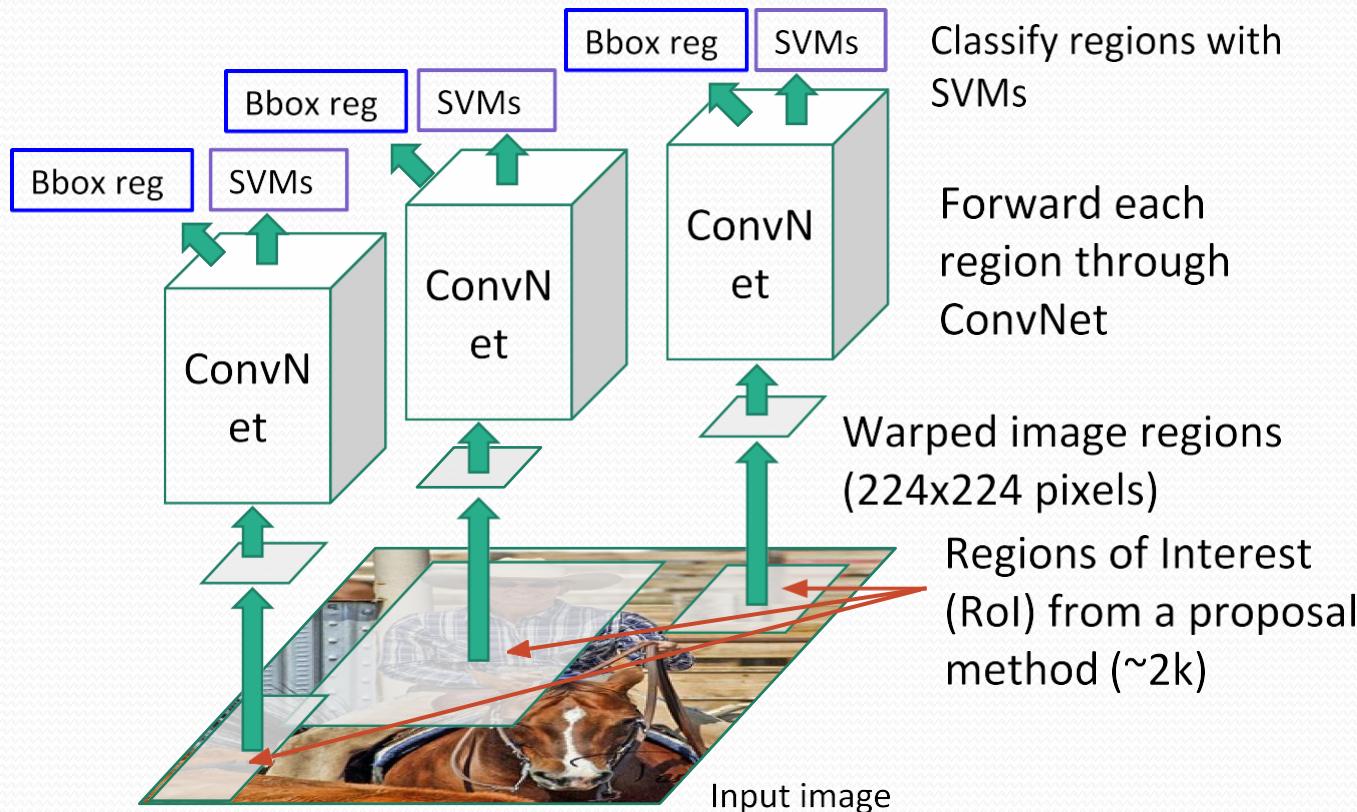
R-CNN



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

R-CNN

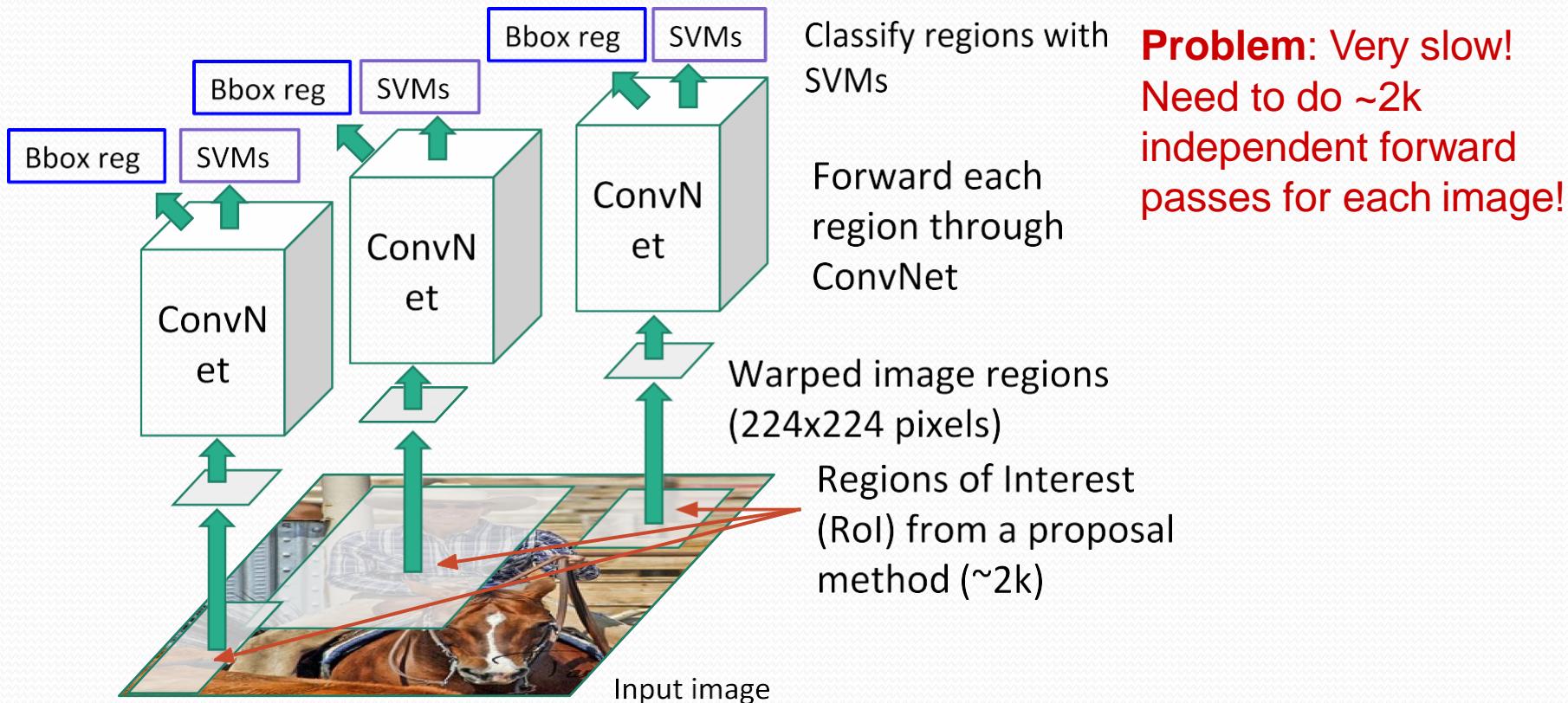
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



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

R-CNN

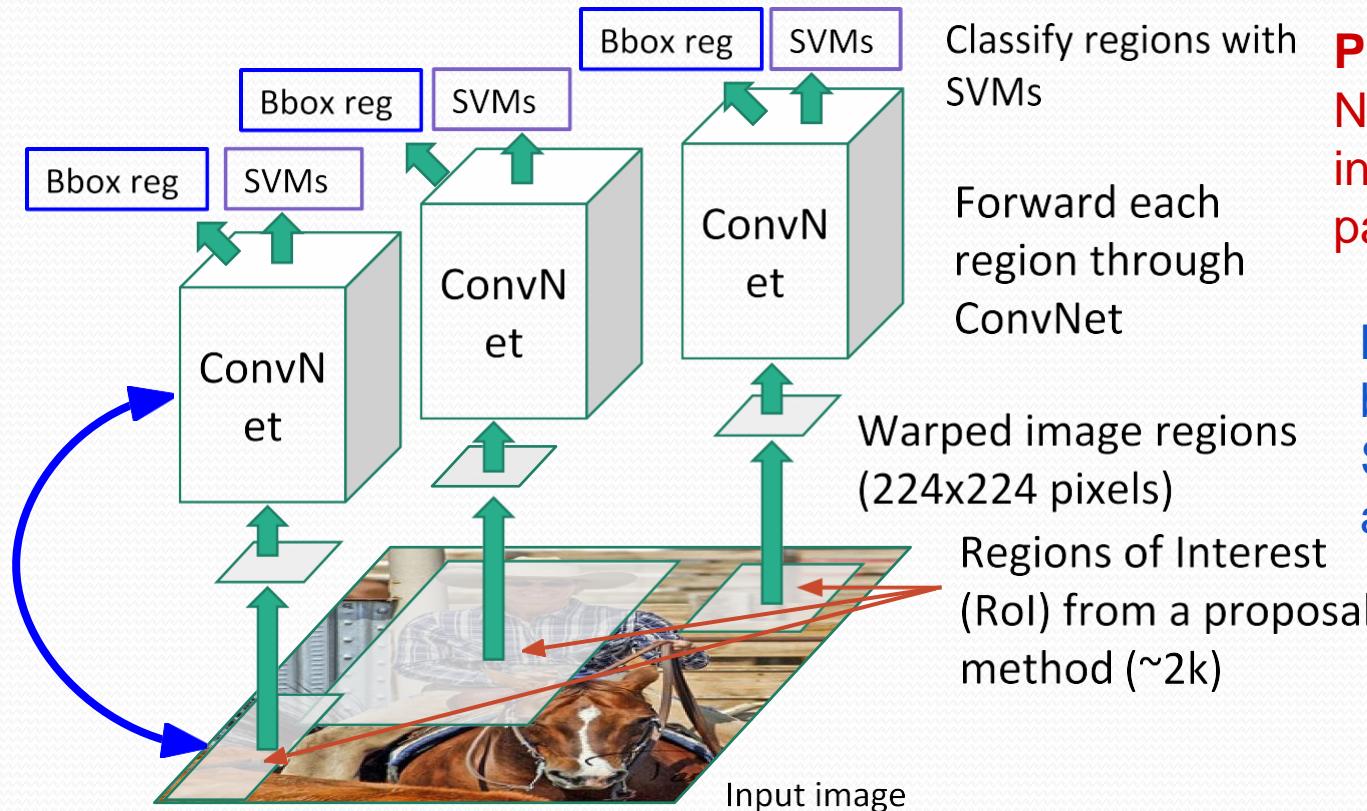
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



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)



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

Idea: Process image before cropping!
Swap convolution and cropping!

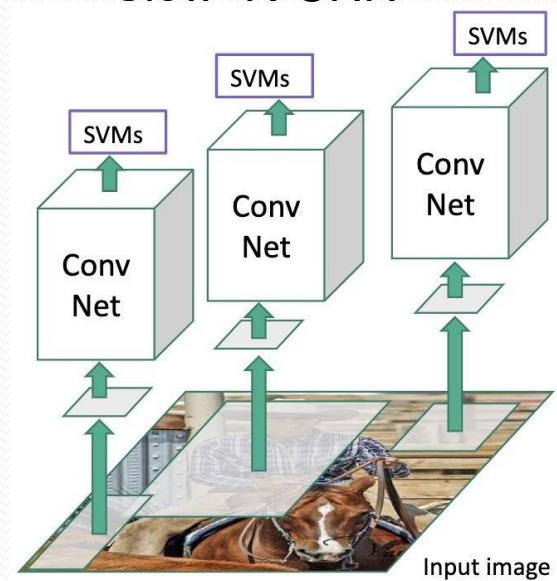
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

Fast R-CNN



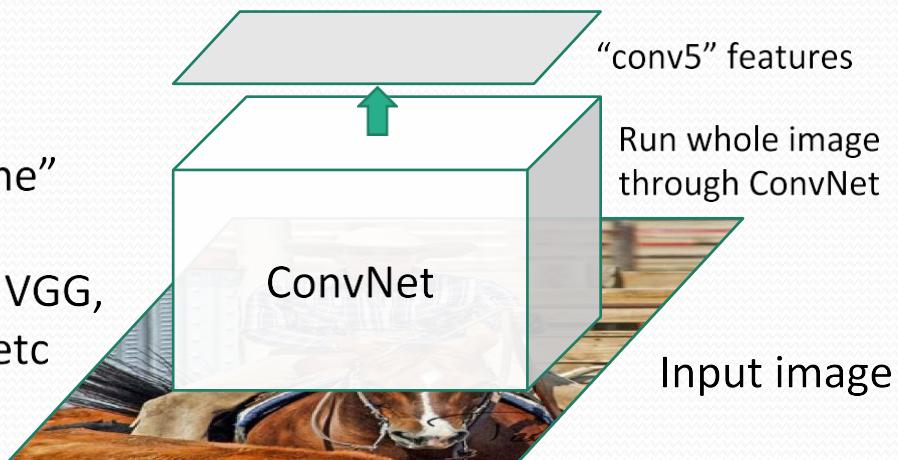
Input image

“Slow” R-CNN

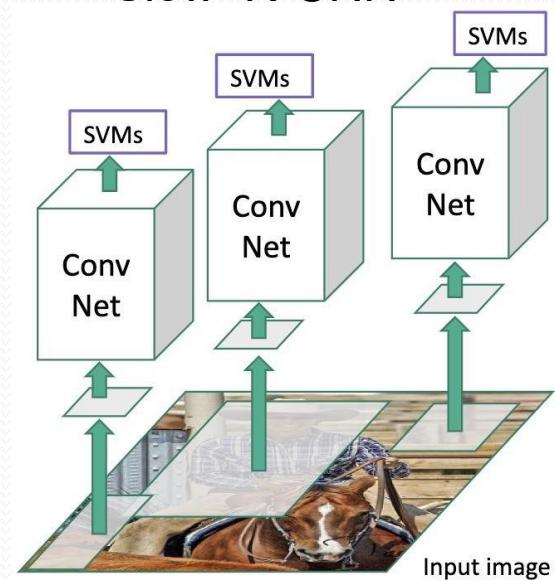


Fast R-CNN

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



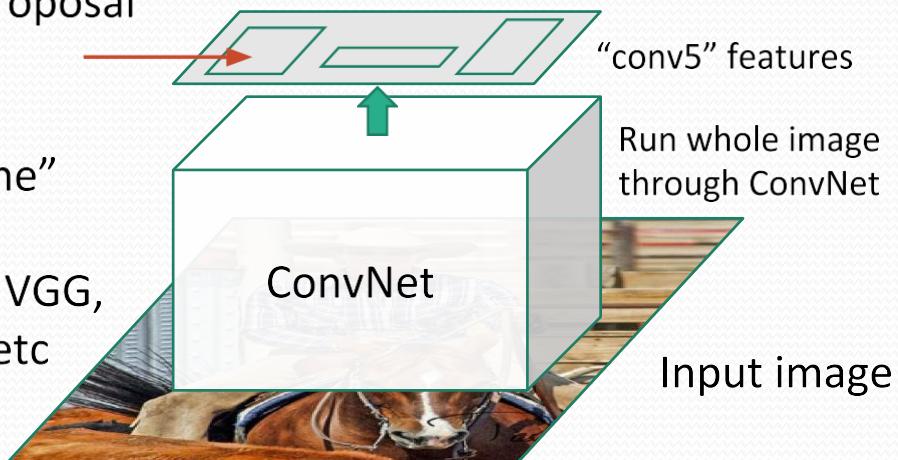
“Slow” R-CNN



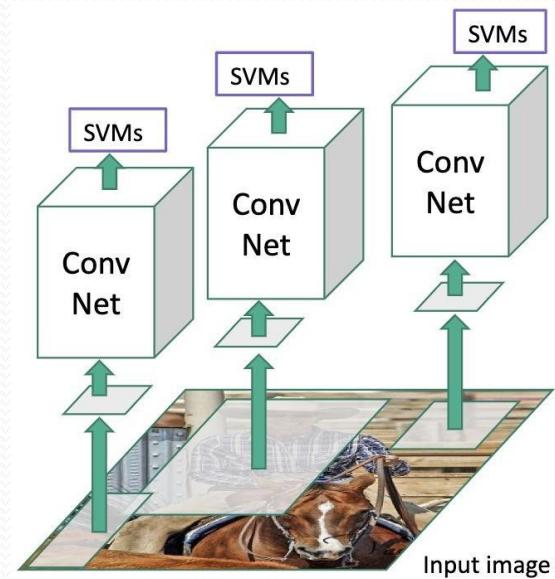
Fast R-CNN

Regions of Interest (RoIs)
from a proposal
method

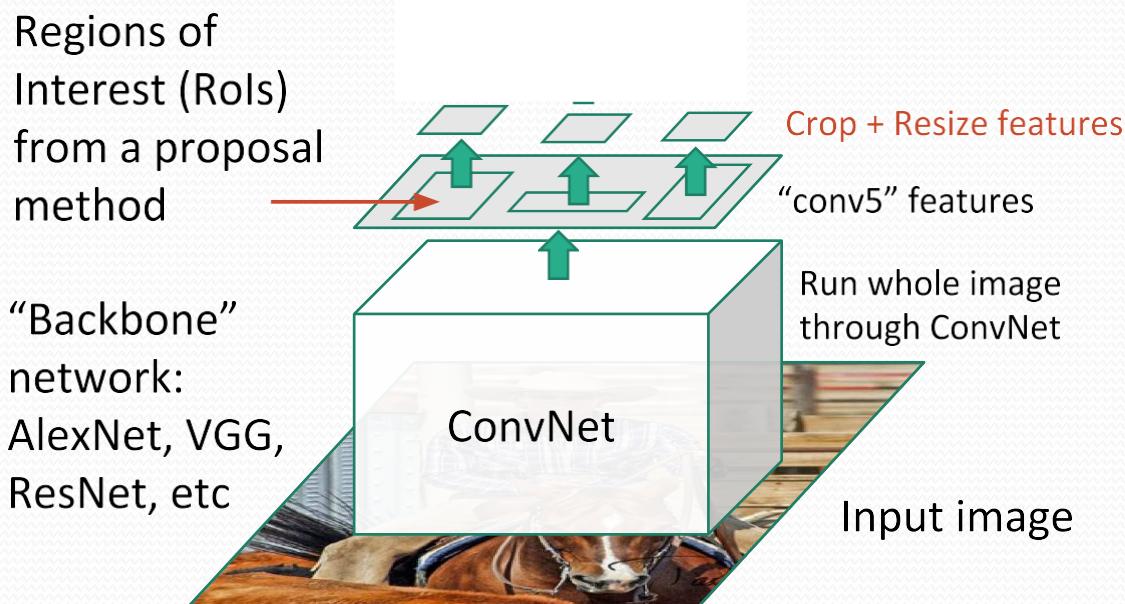
“Backbone”
network:
AlexNet, VGG,
ResNet, etc



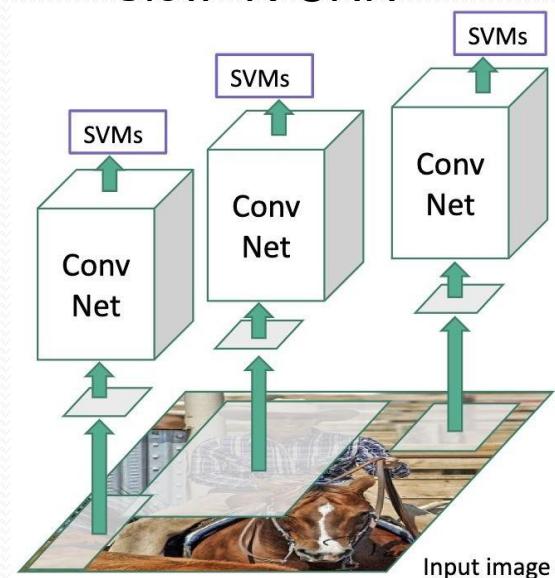
“Slow” R-CNN



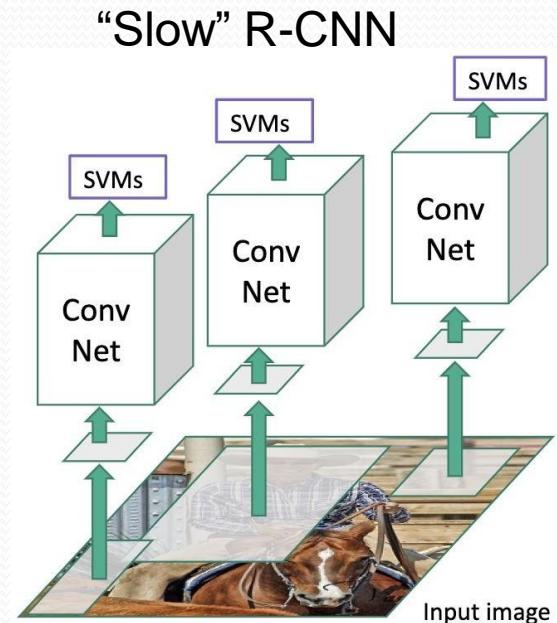
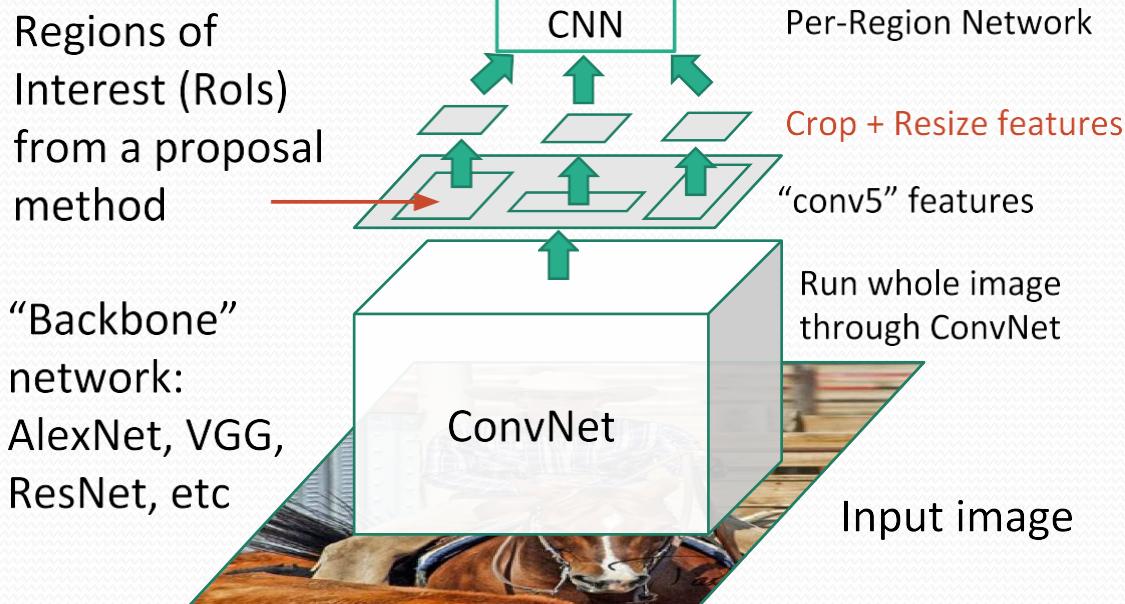
Fast R-CNN



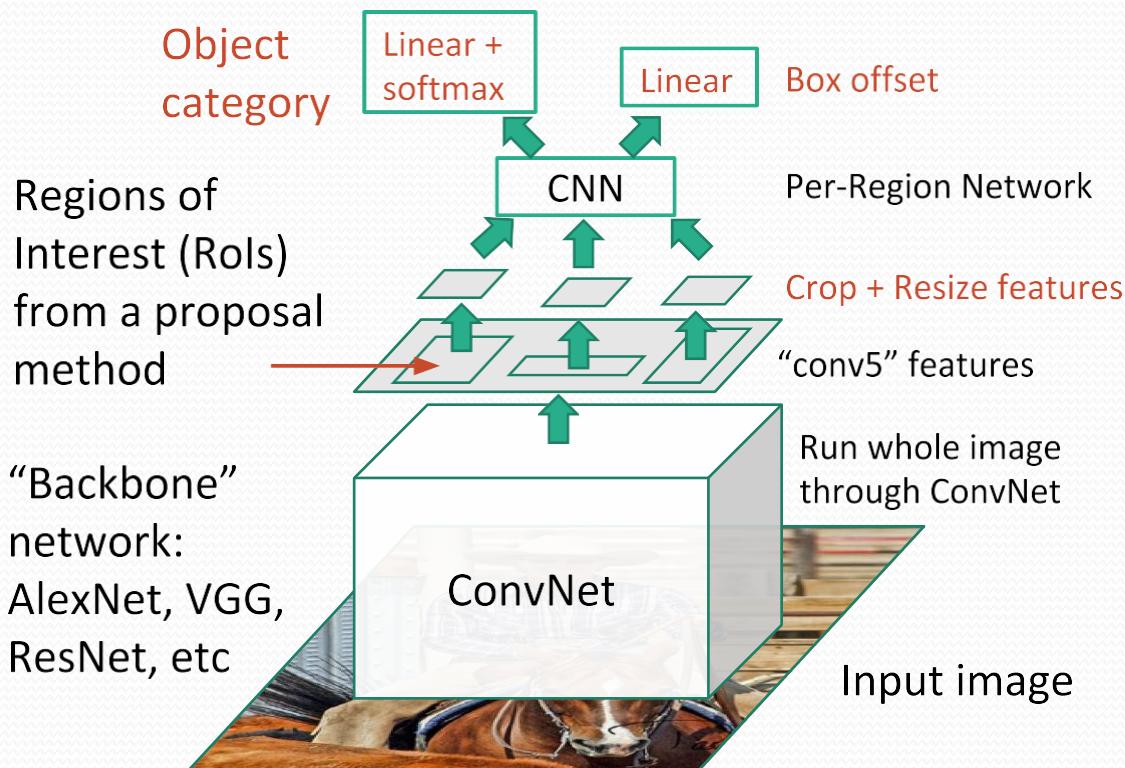
“Slow” R-CNN



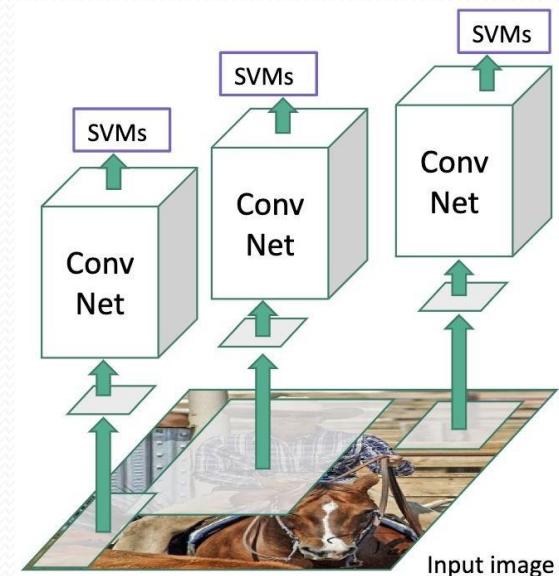
Fast R-CNN



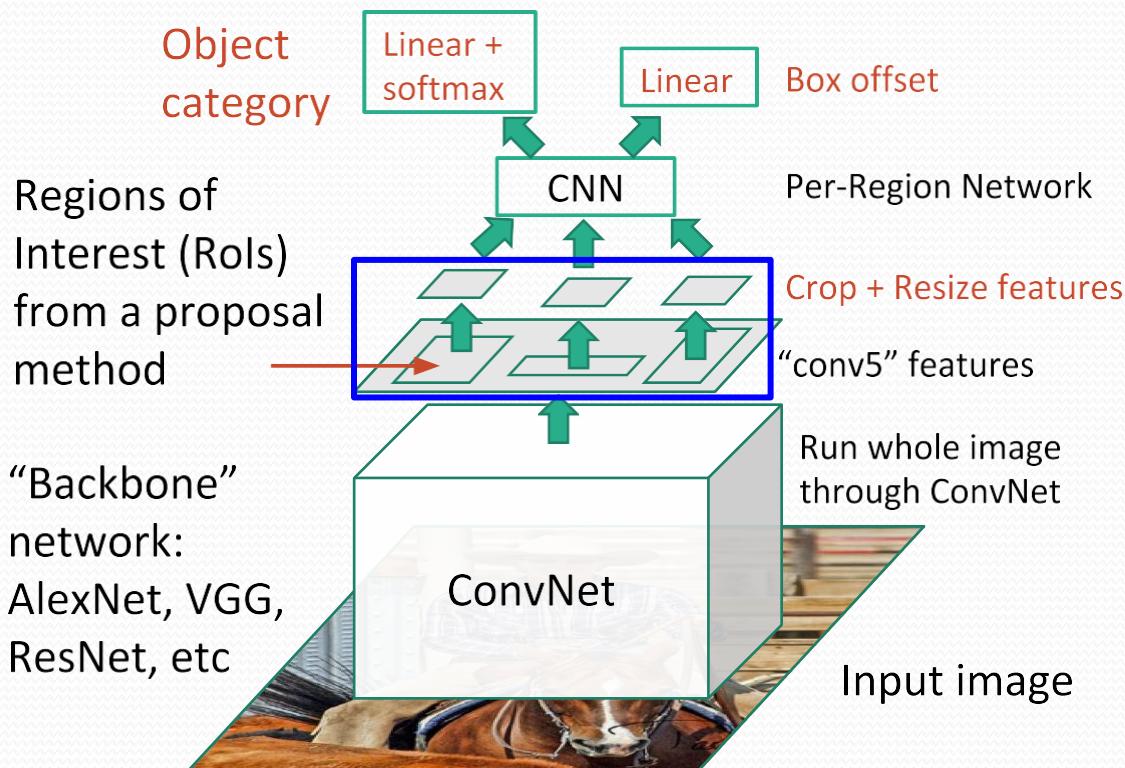
Fast R-CNN



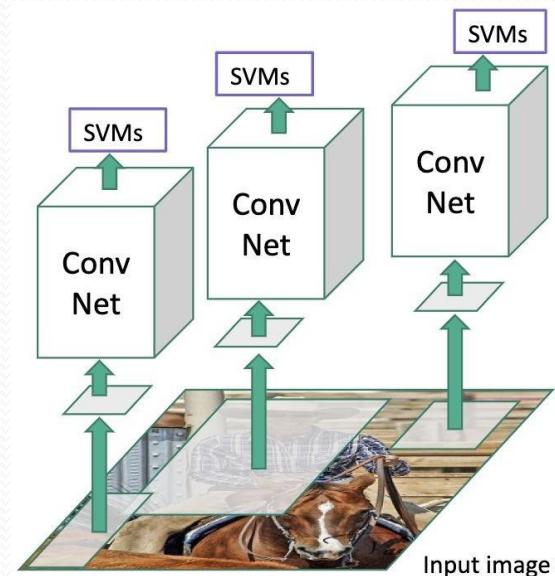
“Slow” R-CNN



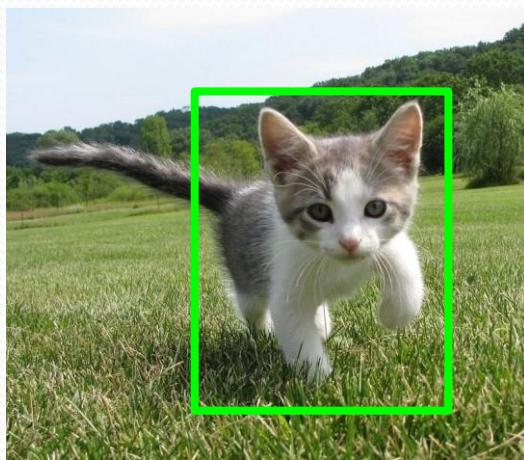
Fast R-CNN



“Slow” R-CNN



Cropping Features: RoI Pool



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

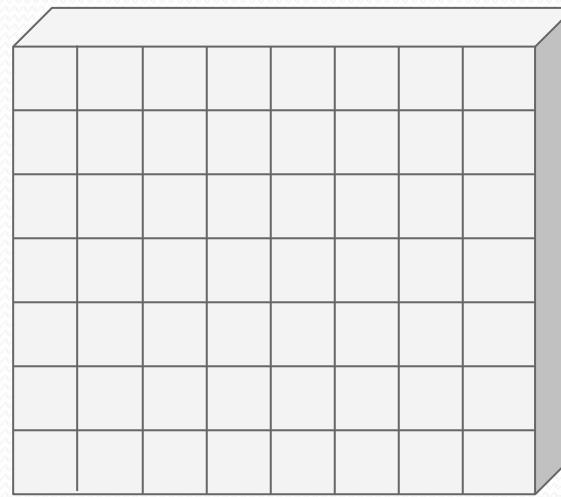
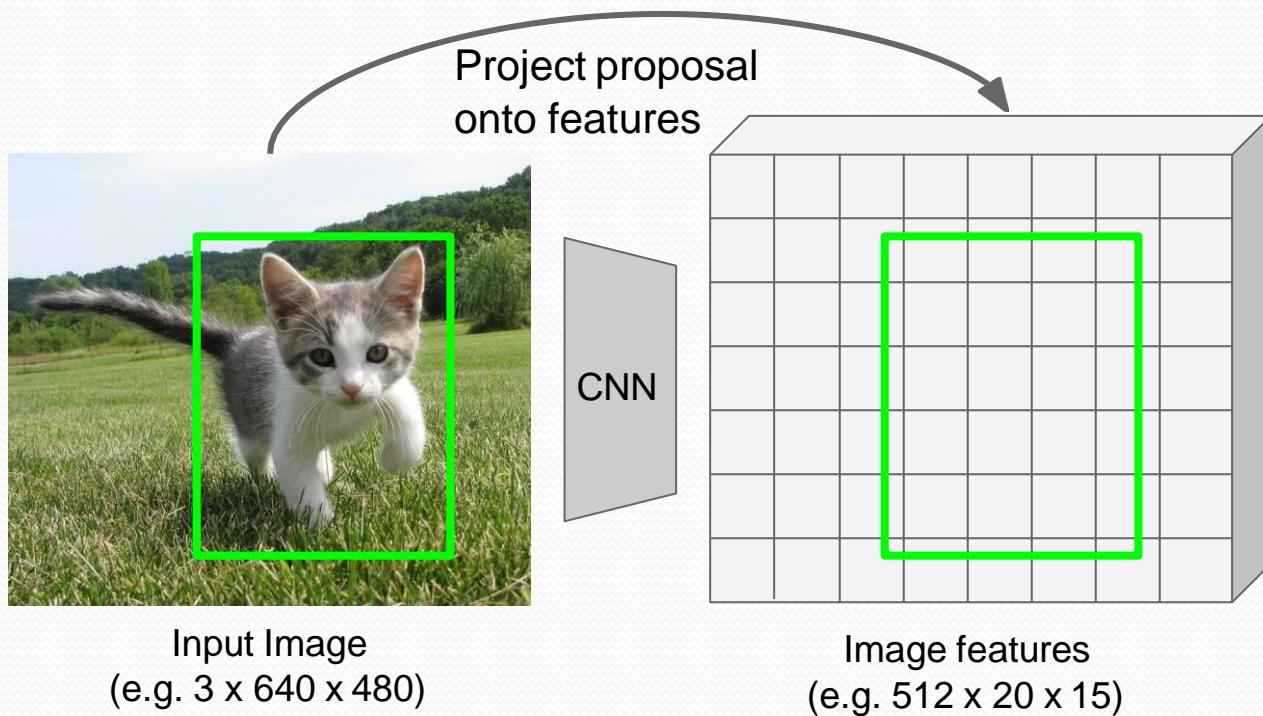


Image features
(e.g. $512 \times 20 \times 15$)

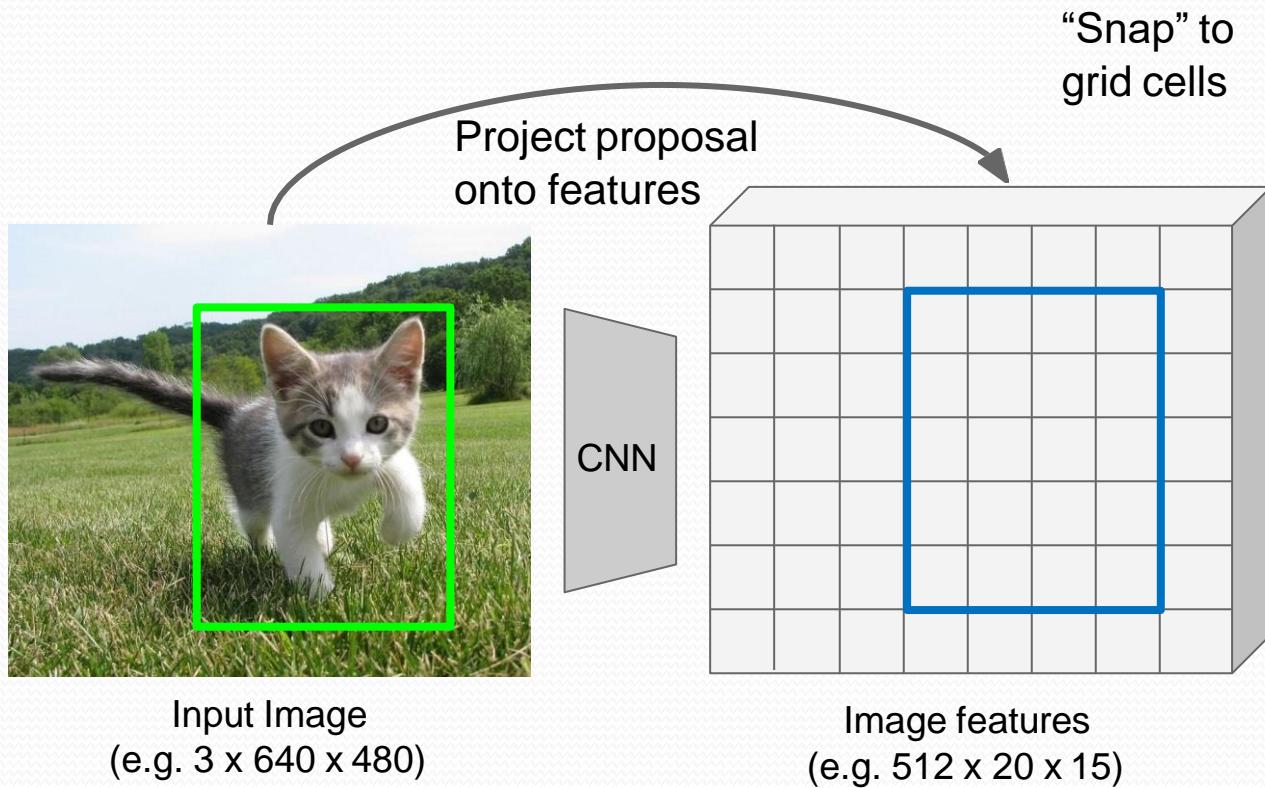
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



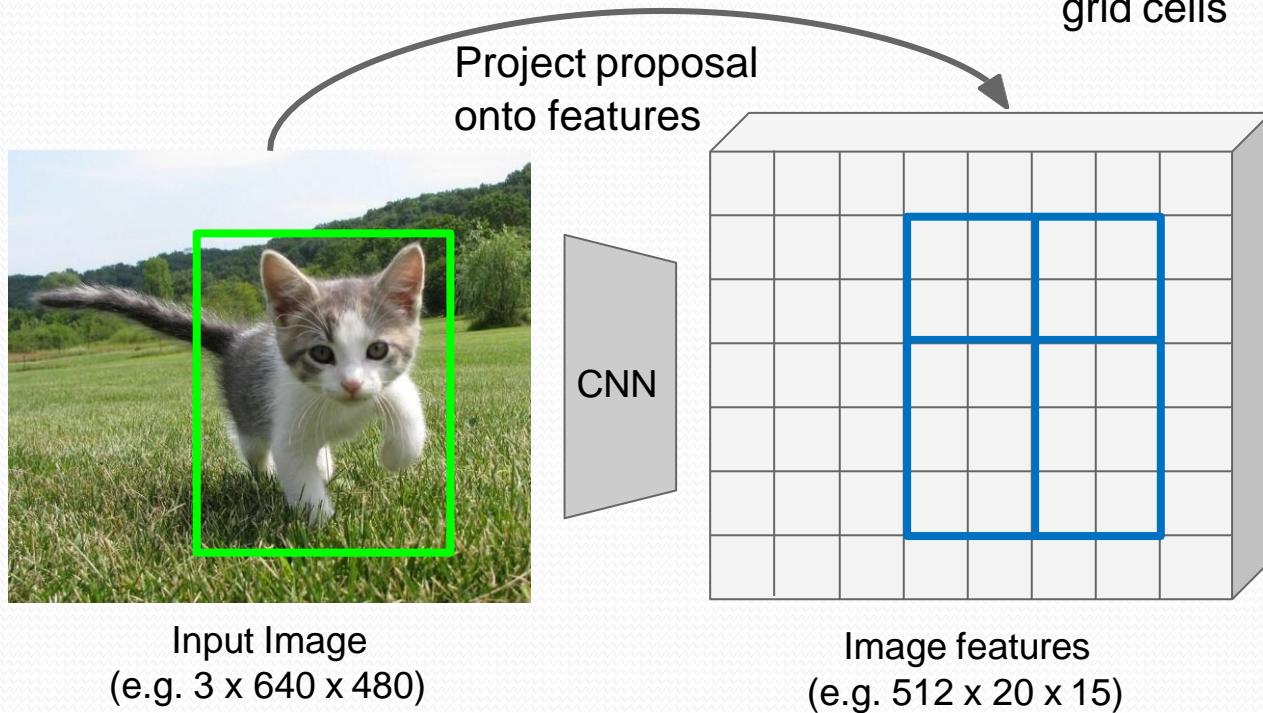
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



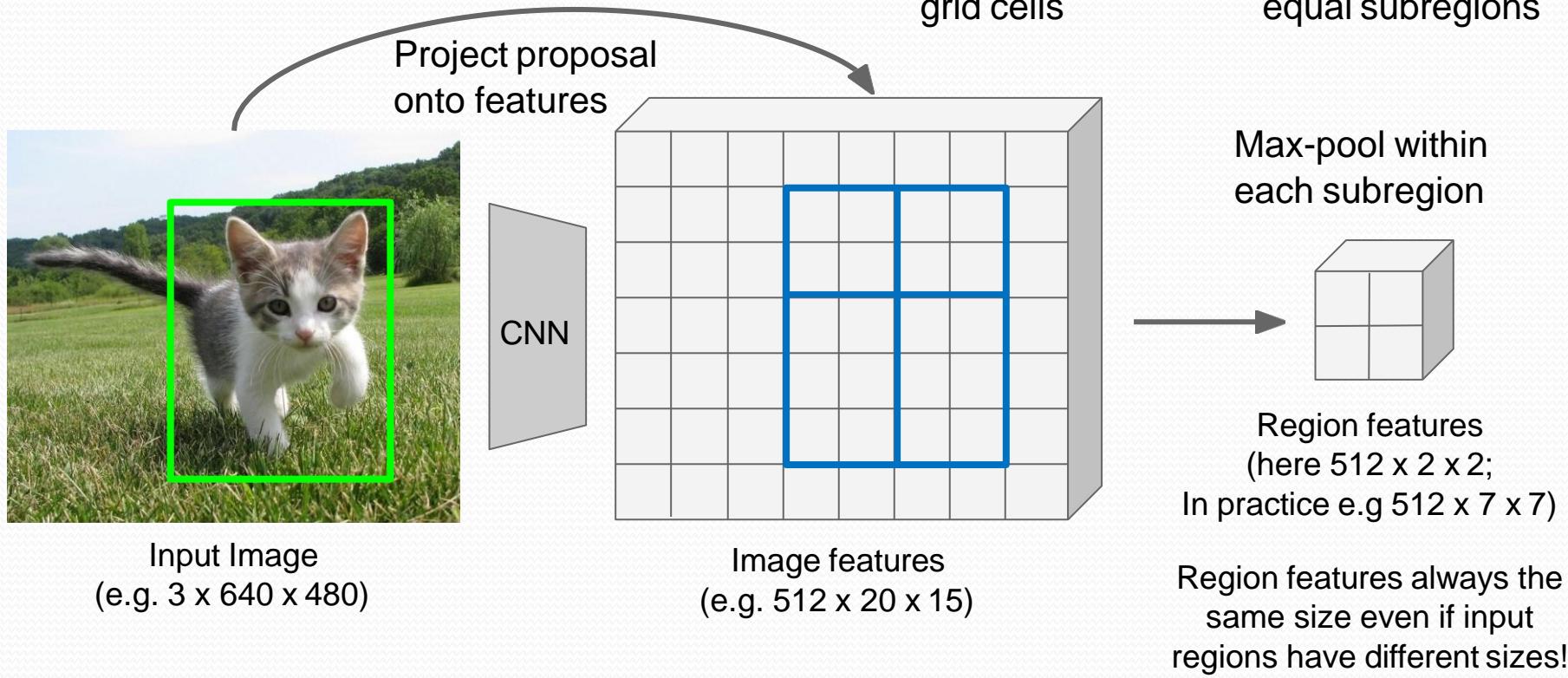
Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



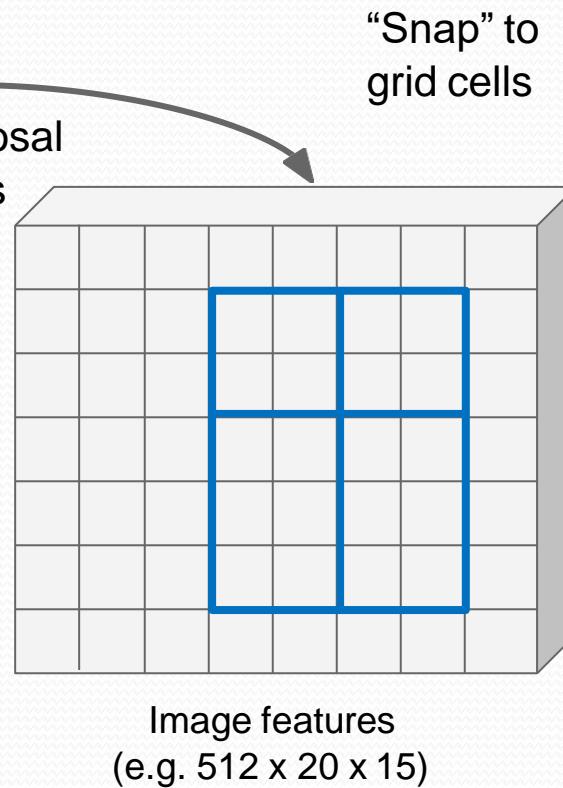
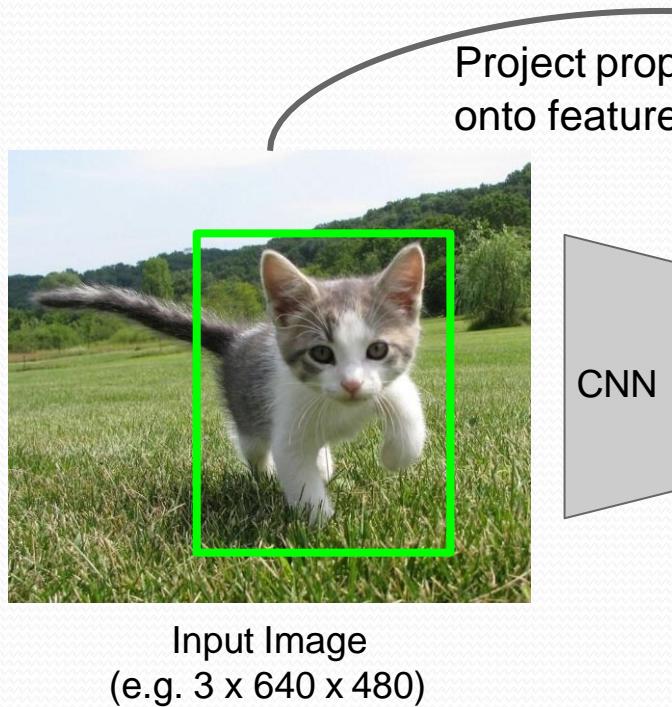
Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



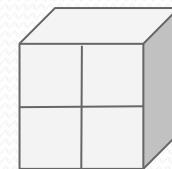
Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



Divide into 2×2 grid of (roughly) equal subregions

Max-pool within each subregion



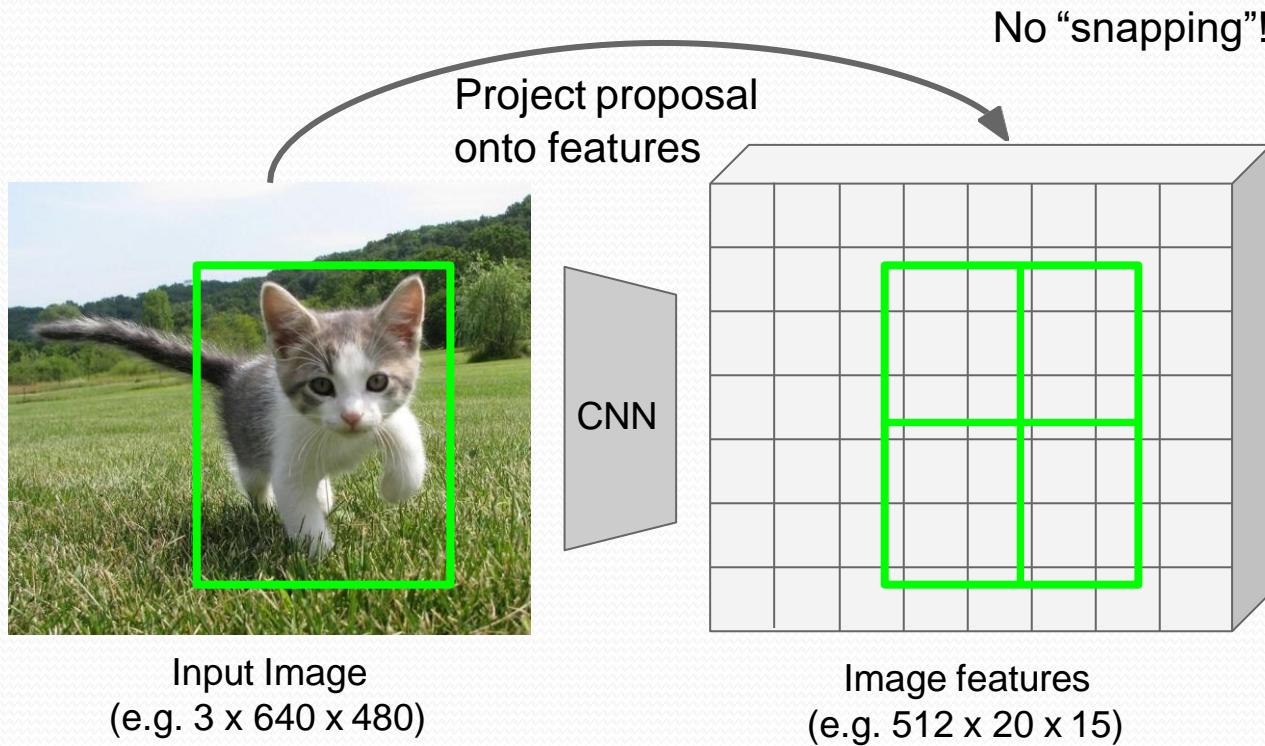
Region features
(here $512 \times 2 \times 2$;
In practice e.g. $512 \times 7 \times 7$)

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

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

Problem: Region features slightly misaligned

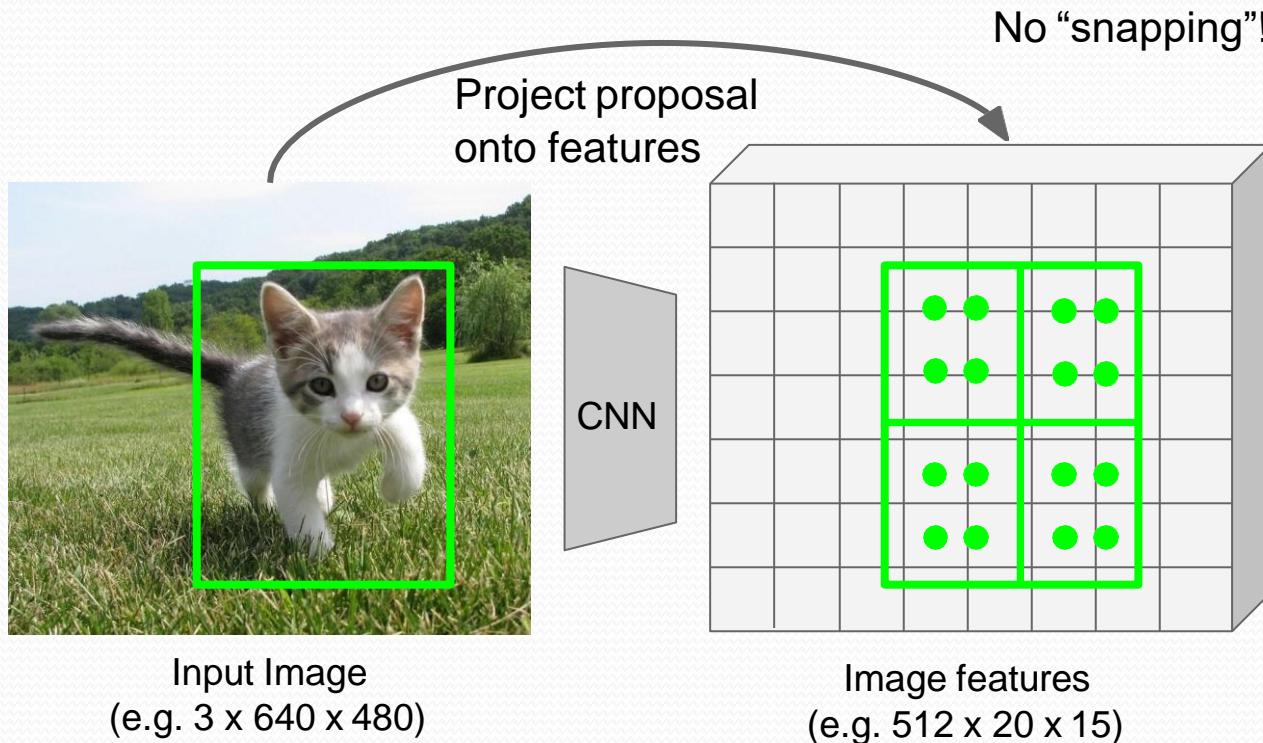
Cropping Features: RoI Align



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

Cropping Features: RoI Align

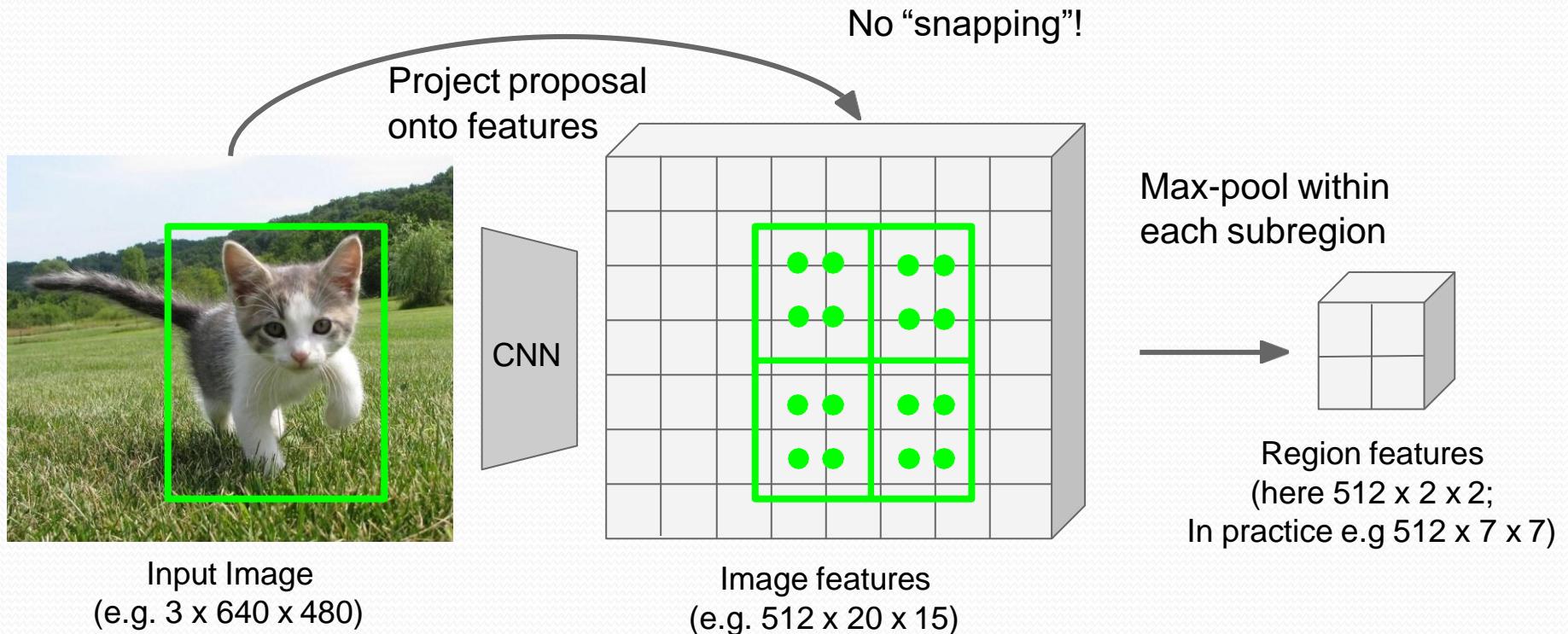
Sample at regular points
in each subregion using
bilinear interpolation



Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Align

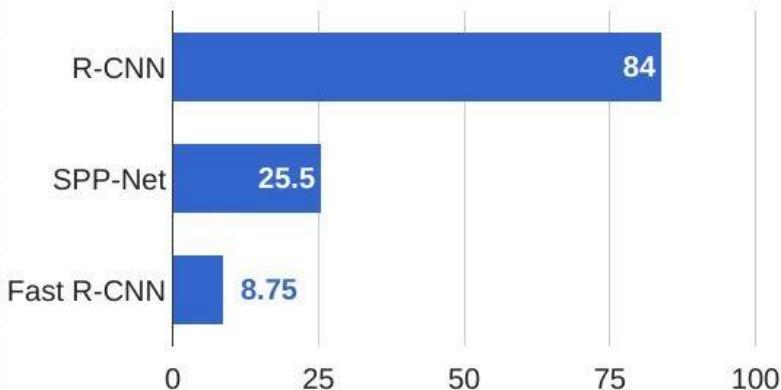
Sample at regular points
in each subregion using
bilinear interpolation



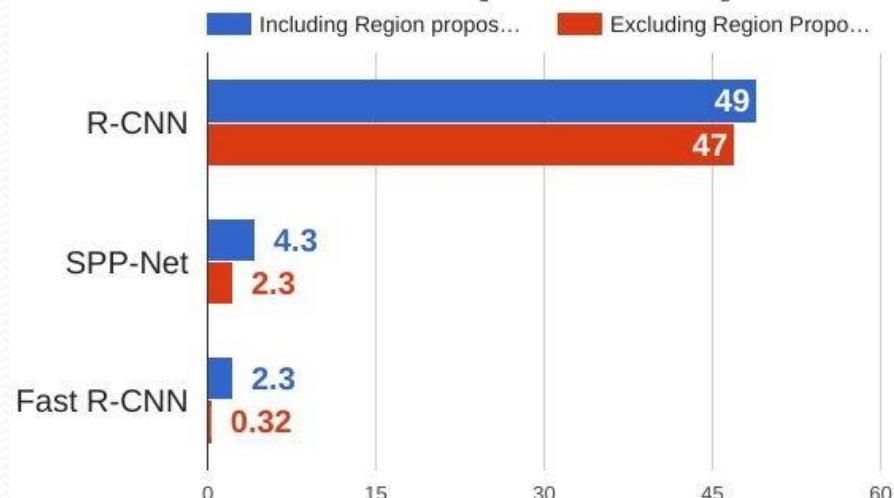
Girshick, "Fast R-CNN", ICCV 2015.

R-CNN vs Fast R-CNN

Training time (Hours)



Test time (seconds)

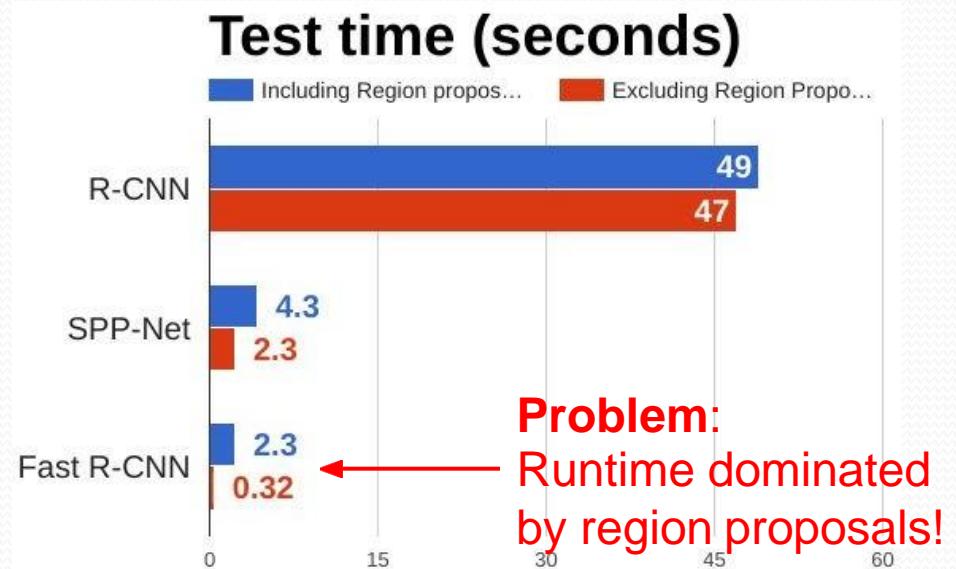


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

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

Girshick, "Fast R-CNN", ICCV 2015

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

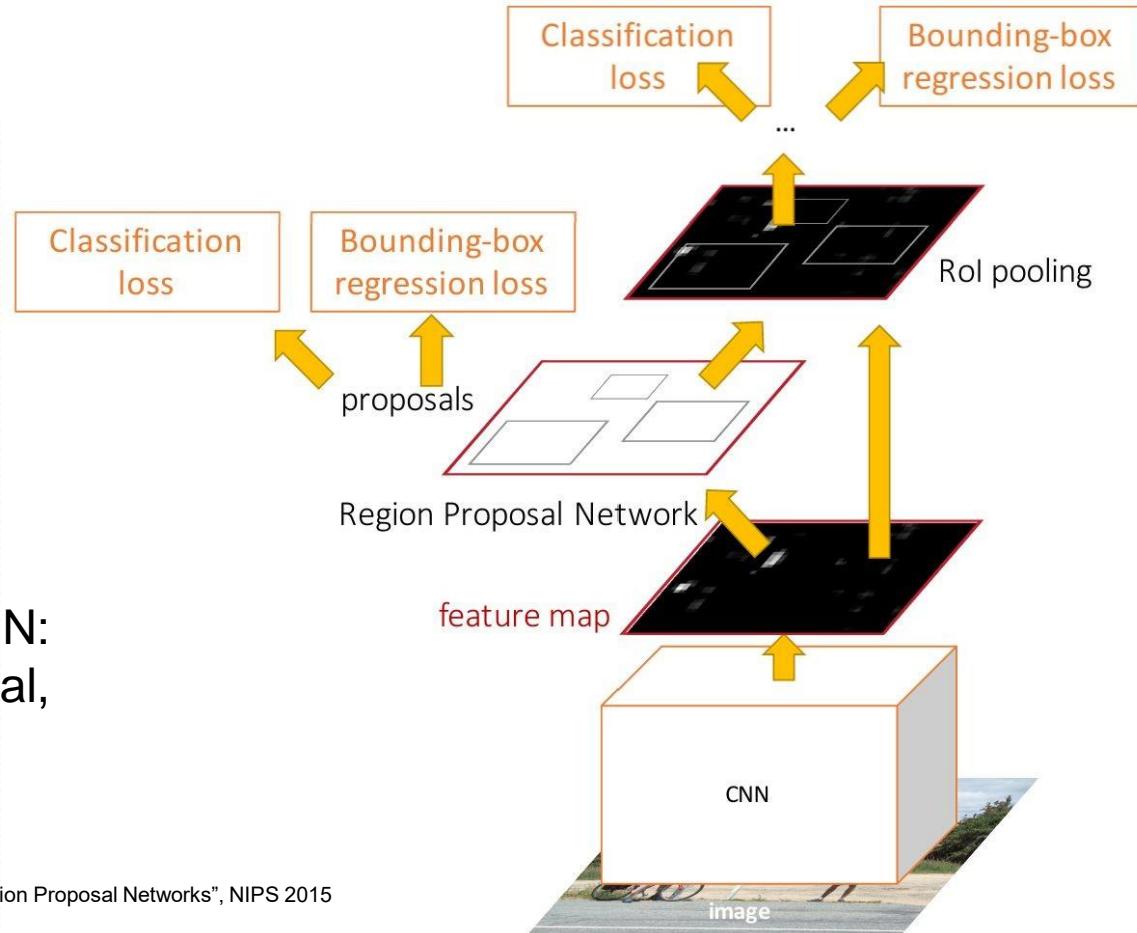
Girshick, "Fast R-CNN", ICCV 2015

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



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

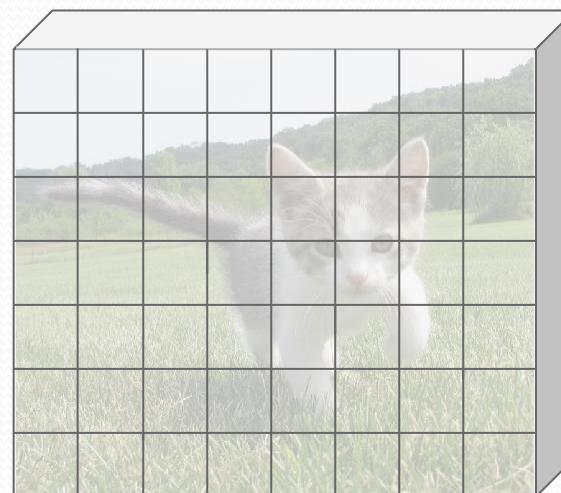
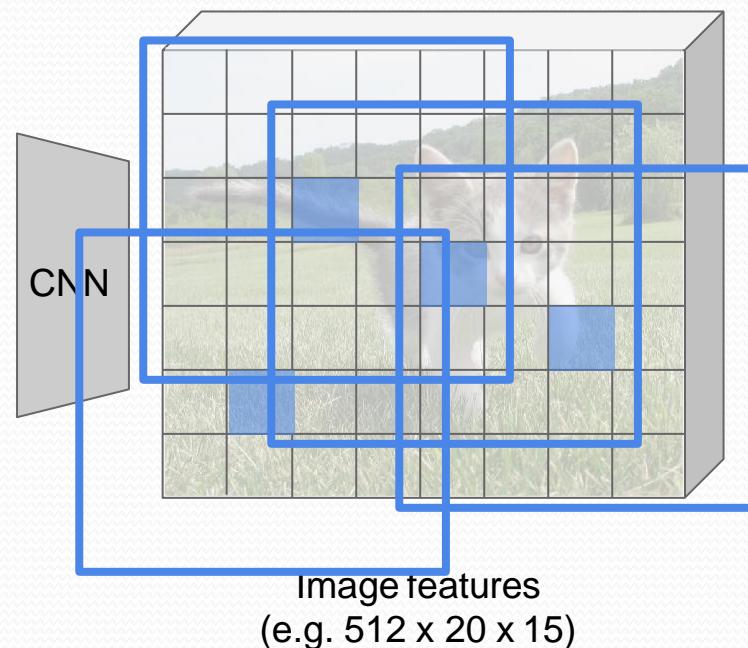


Image features
(e.g. $512 \times 20 \times 15$)

Region Proposal Network



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

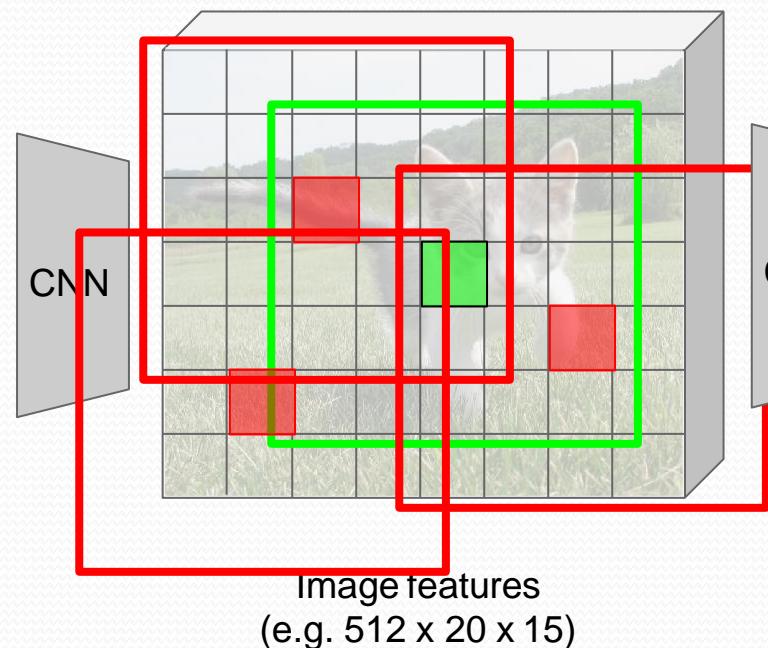


Imagine an **anchor box** of fixed size at each point in the feature map

Region Proposal Network



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



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

Anchor is an object?
 $1 \times 20 \times 15$

At each point, predict
whether the corresponding
anchor contains an object
(per-pixel logistic regression)

Region Proposal Network



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

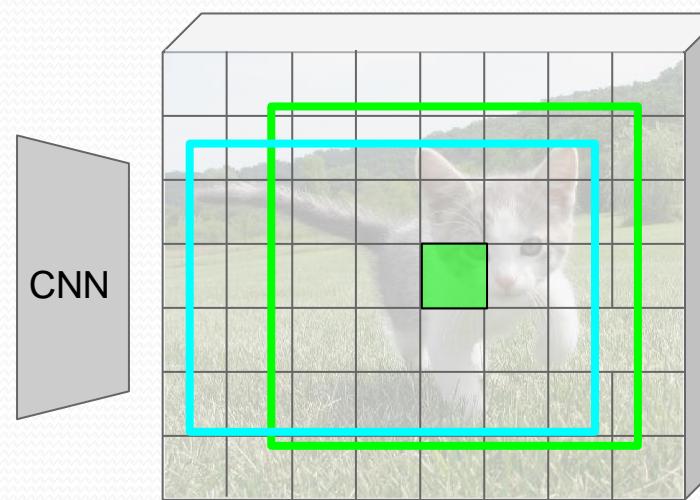


Image features
(e.g. $512 \times 20 \times 15$)

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

Conv → Anchor is an object?
 $1 \times 20 \times 15$

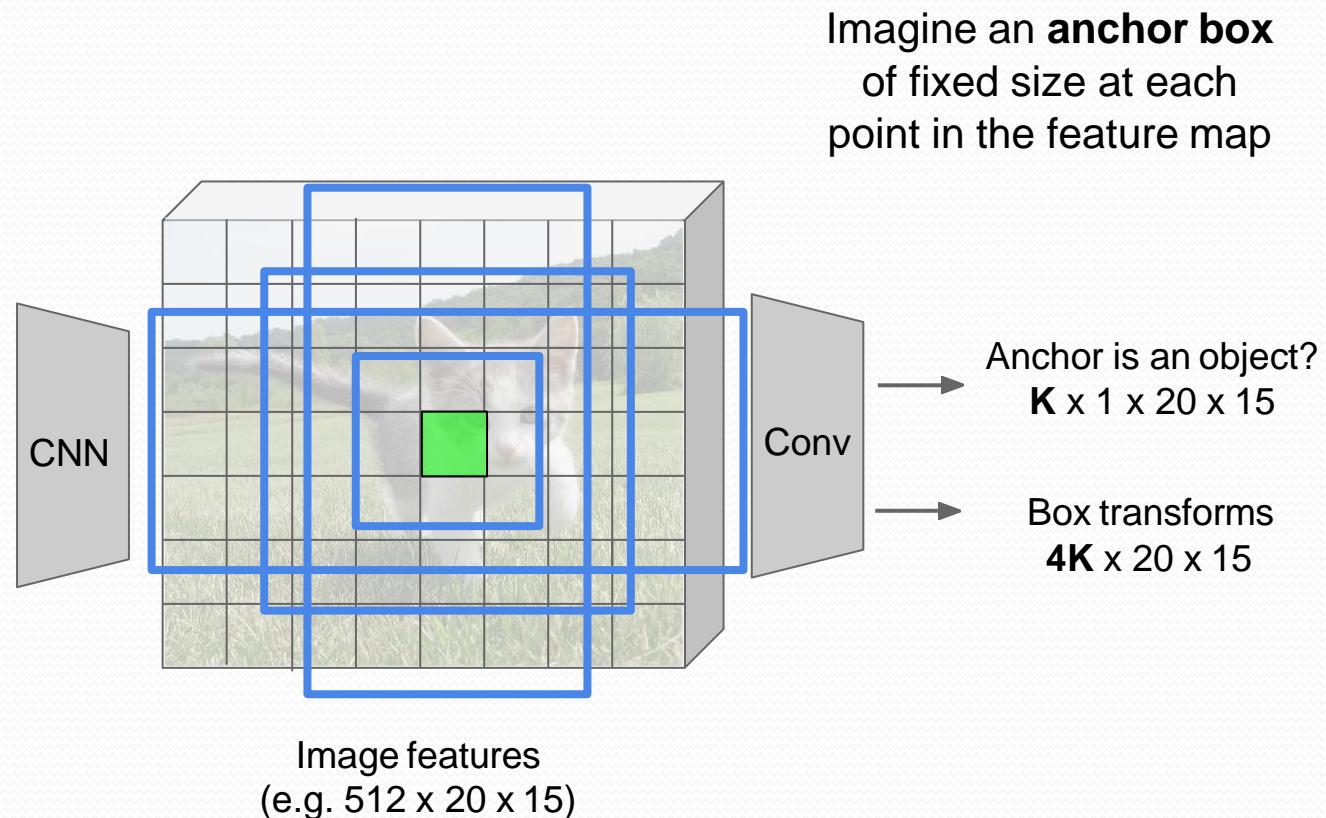
Conv → Box transforms
 $4 \times 20 \times 15$

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

Region Proposal Network



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



Region Proposal Network



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

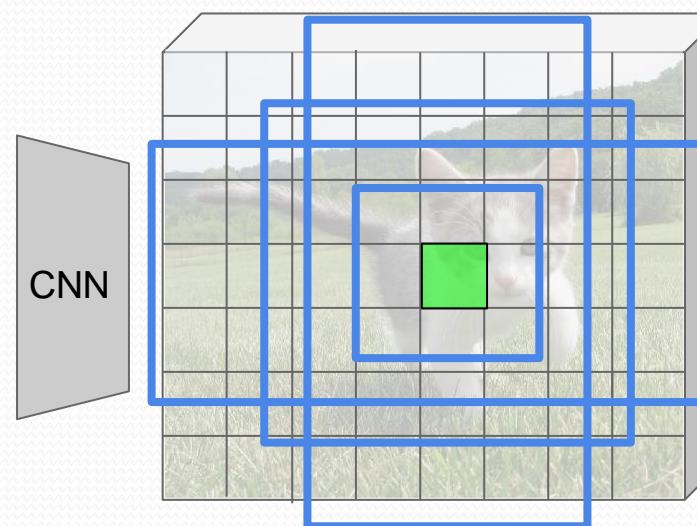


Image features
(e.g. $512 \times 20 \times 15$)

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

→ Anchor is an object?
 $K \times 1 \times 20 \times 15$

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

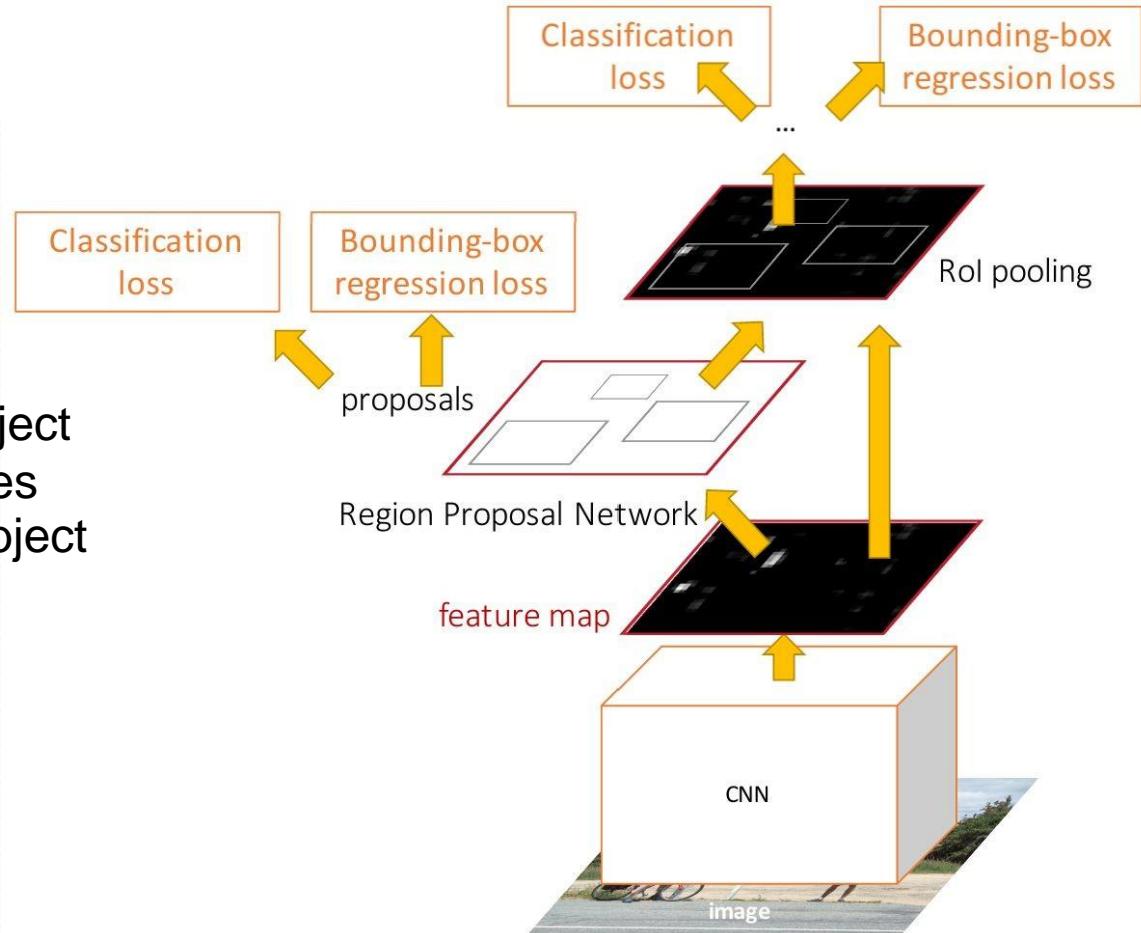
Sort the K^*20^*15 boxes by
their “object” score, take top
~300 as our proposals

Faster R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

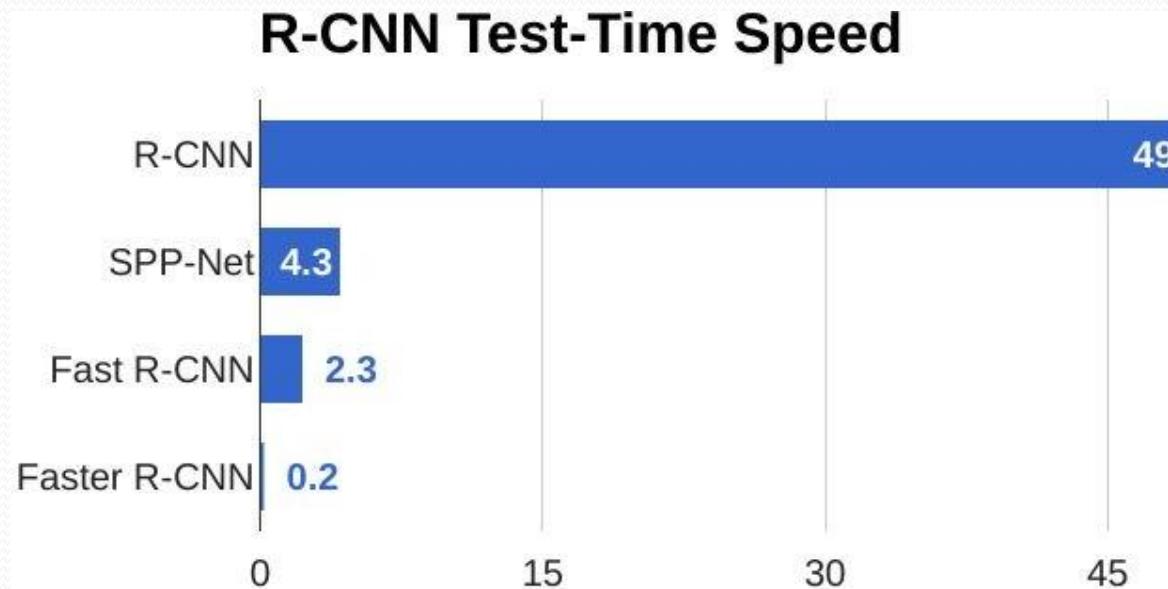
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



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

Faster R-CNN:

Make CNN do proposals!



Faster R-CNN:

Make CNN do proposals!

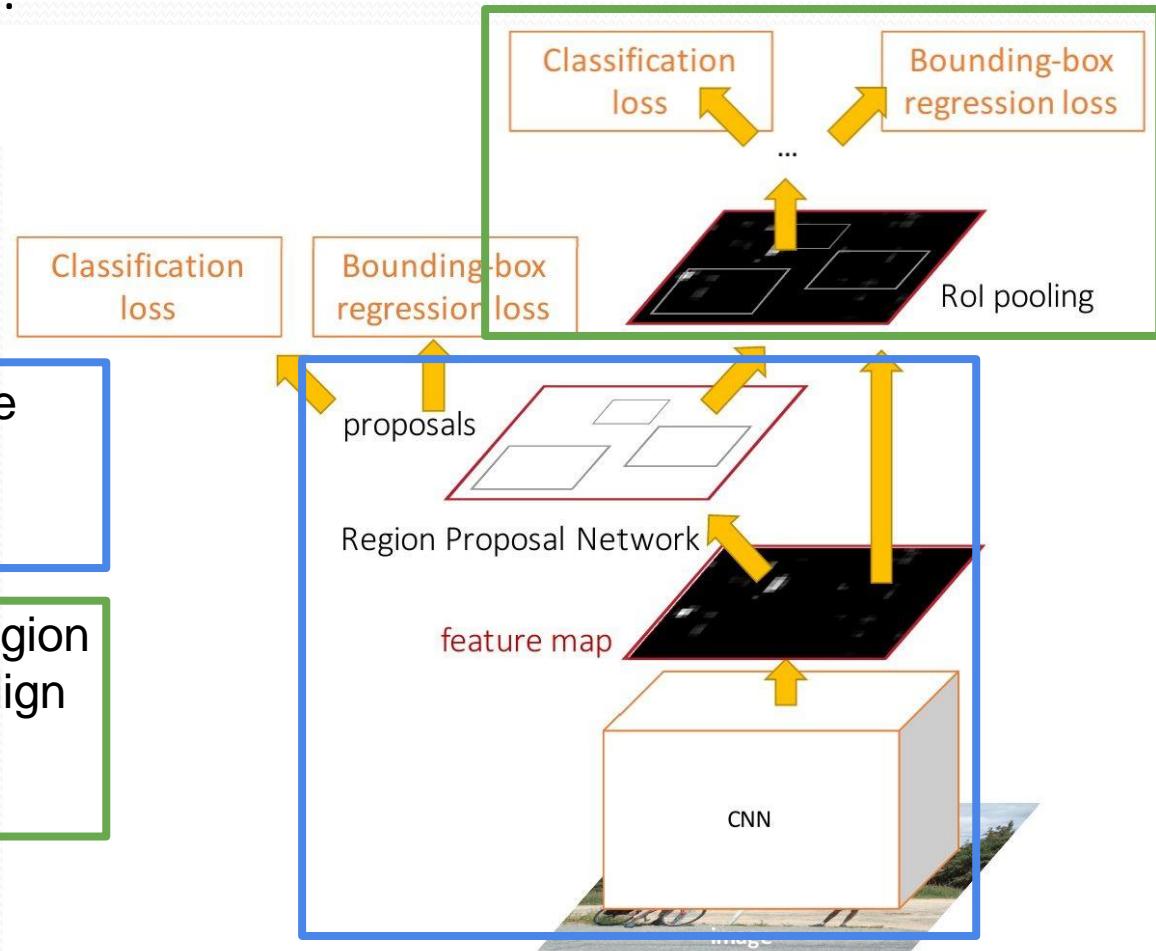
Faster R-CNN is a
Two-stage object detector

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

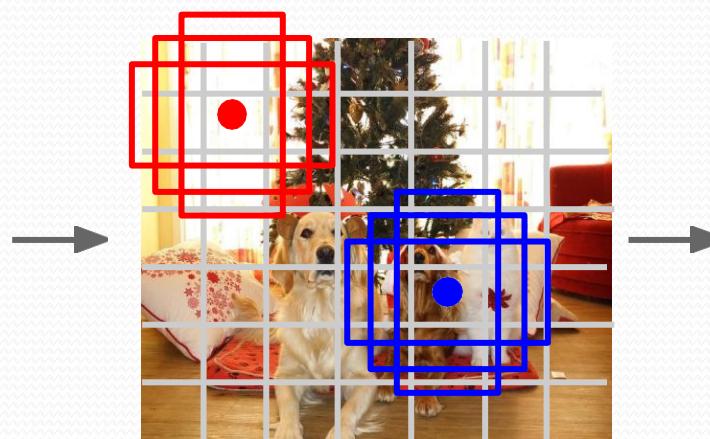


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Single-Stage Object Detectors: YOLO / SSD / RetinaNet



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

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

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

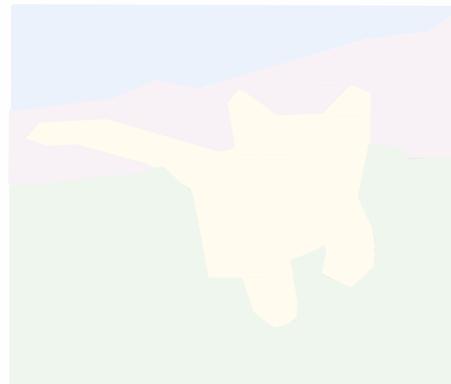
Classification



CAT

No spatial extent

Semantic
Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object
Detection



DOG, DOG, CAT

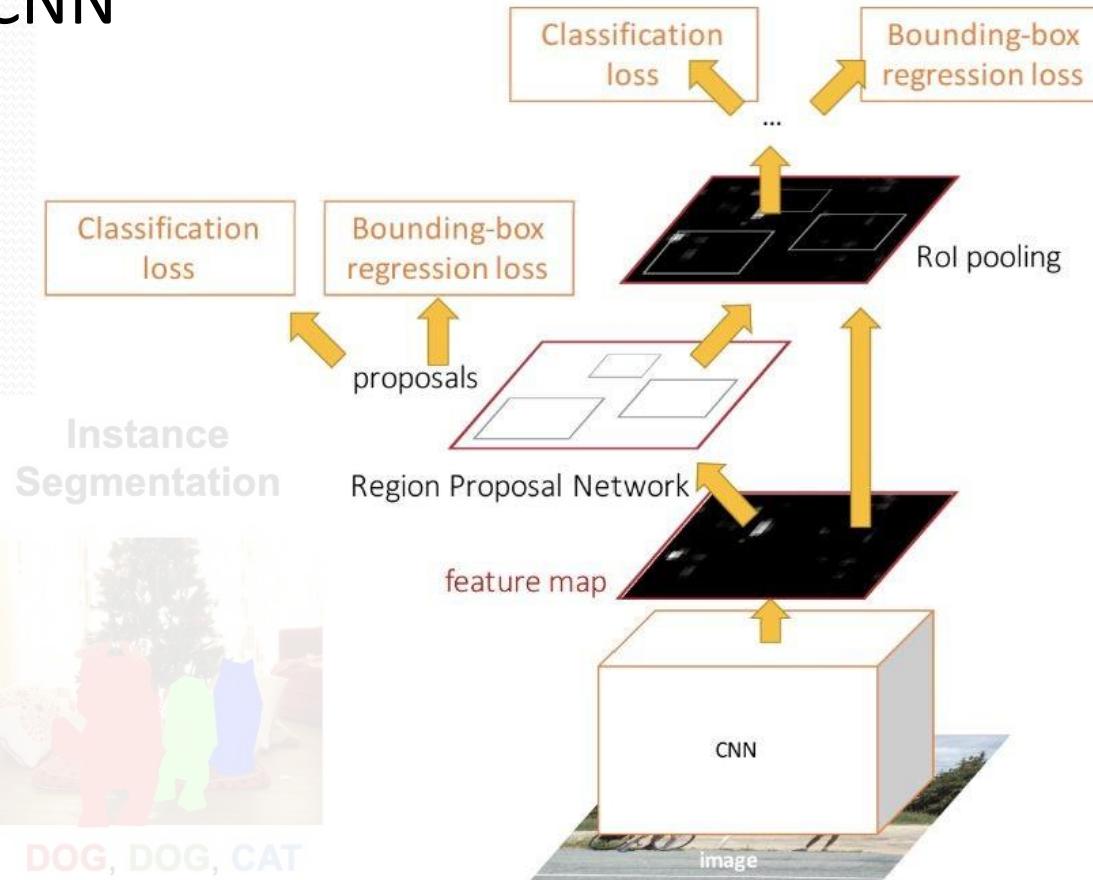
Multiple Object

Instance
Segmentation



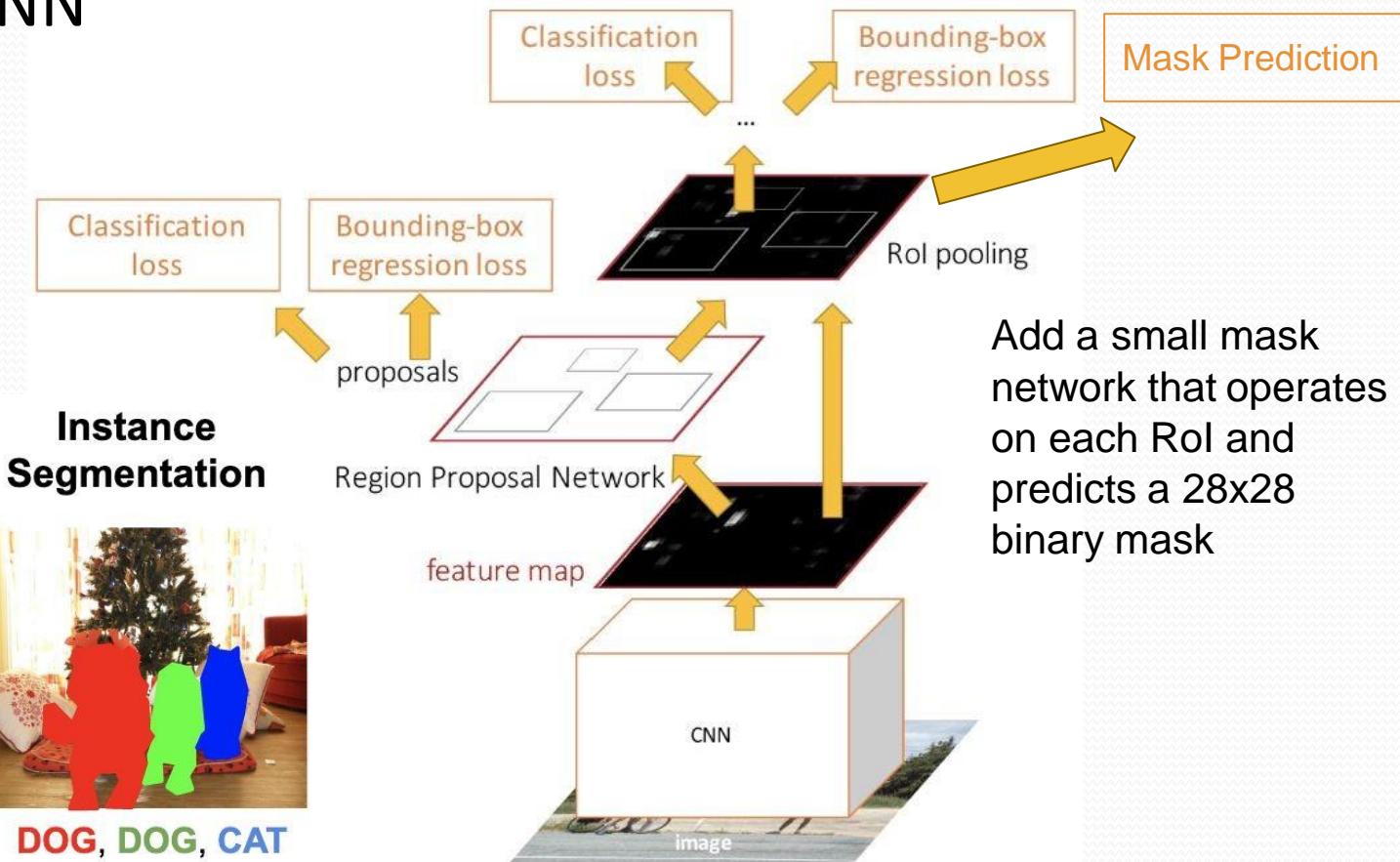
DOG, DOG, CAT

Object Detection: Faster R-CNN



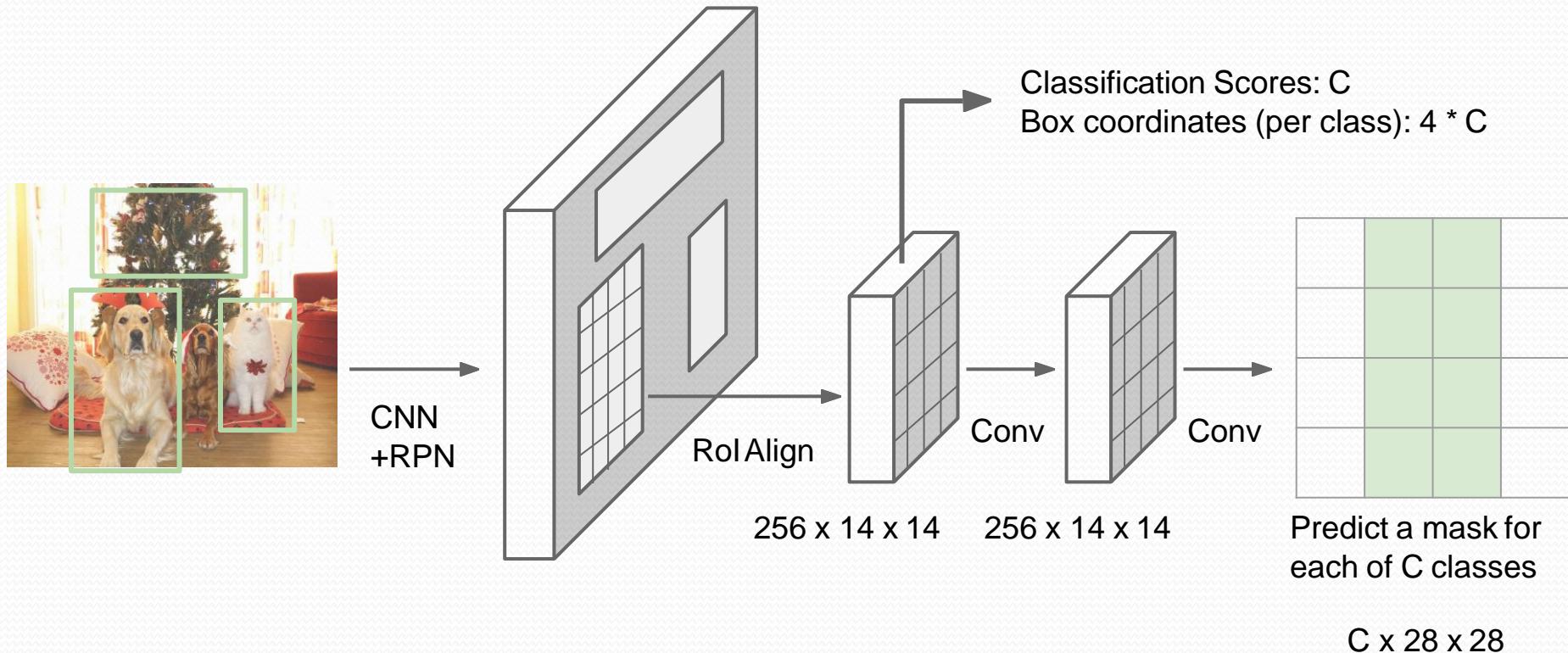
He et al., "Mask R-CNN", ICCV 2017

Instance Segmentation: Mask R-CNN



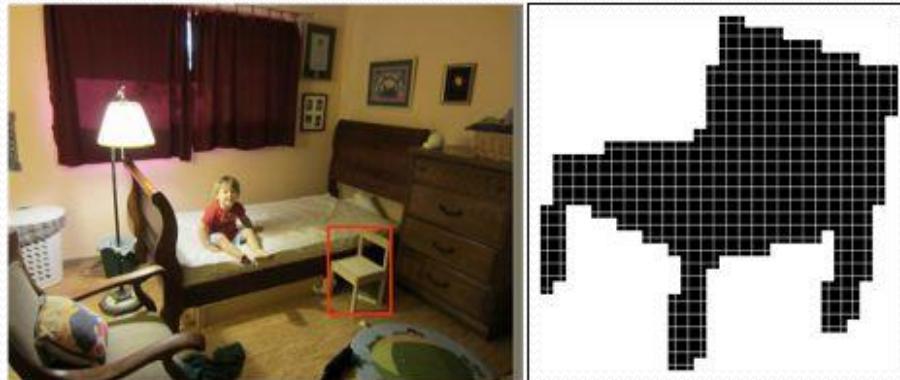
He et al., "Mask R-CNN", ICCV 2017

Mask R-CNN

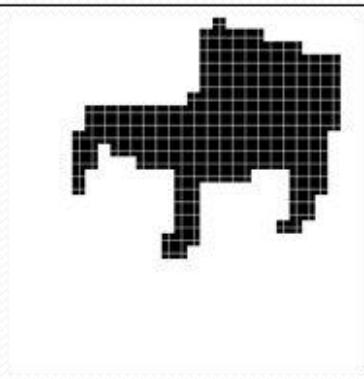
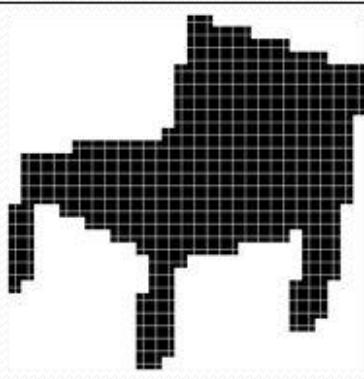


He et al, "Mask R-CNN", arXiv 2017

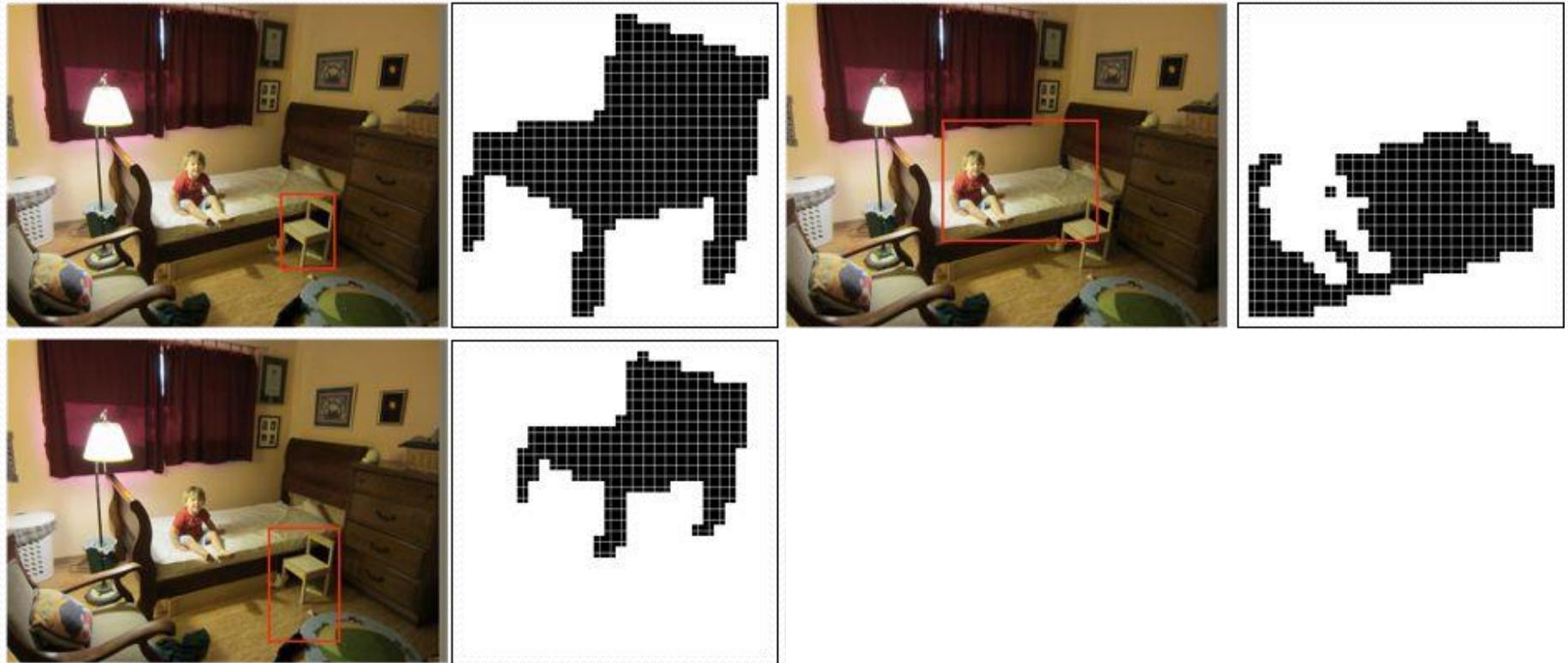
Mask R-CNN: Example Mask Training Targets



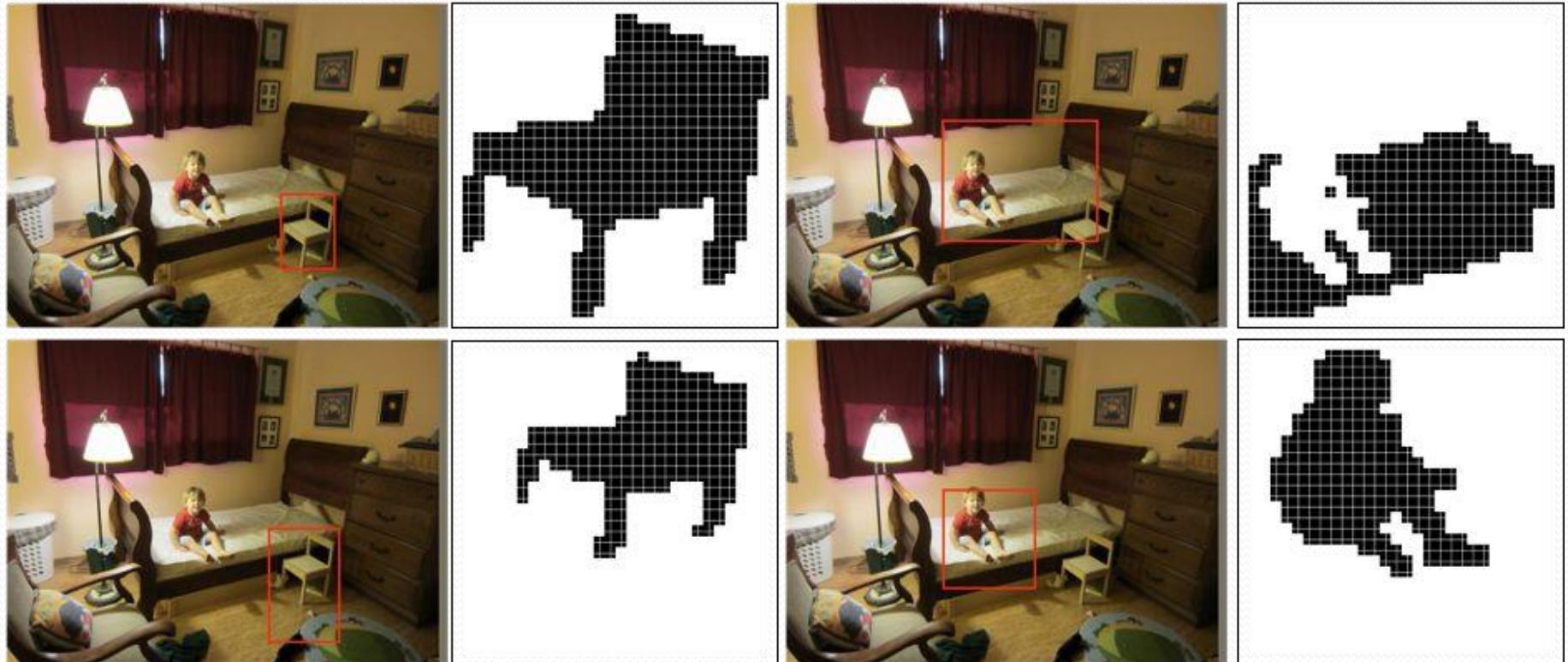
Mask R-CNN: Example Mask Training Targets



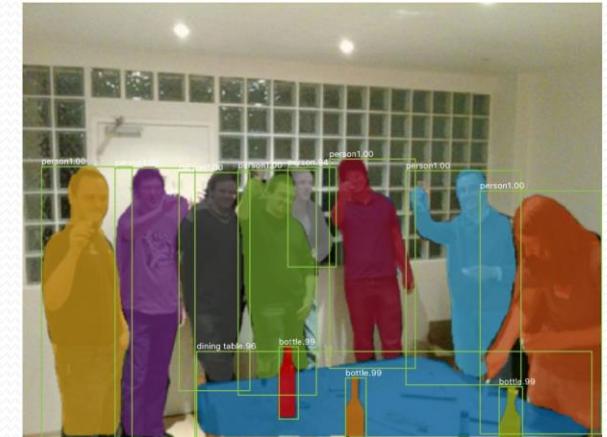
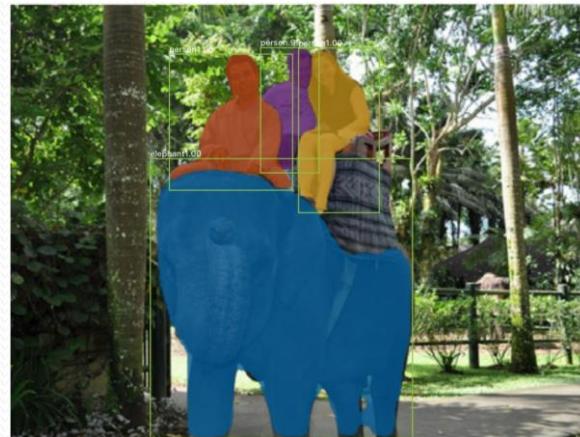
Mask R-CNN: Example Mask Training Targets



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

Faster RCNN, SSD, RFCN, Mask R-CNN

Caffe2 Detectron:

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

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

Finetune on your own dataset with pre-trained models

Computer Vision Tasks

Classification



CAT

No spatial extent

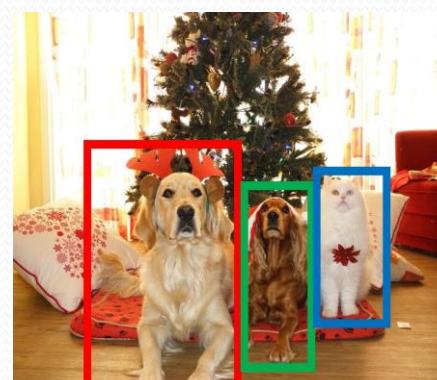
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Object

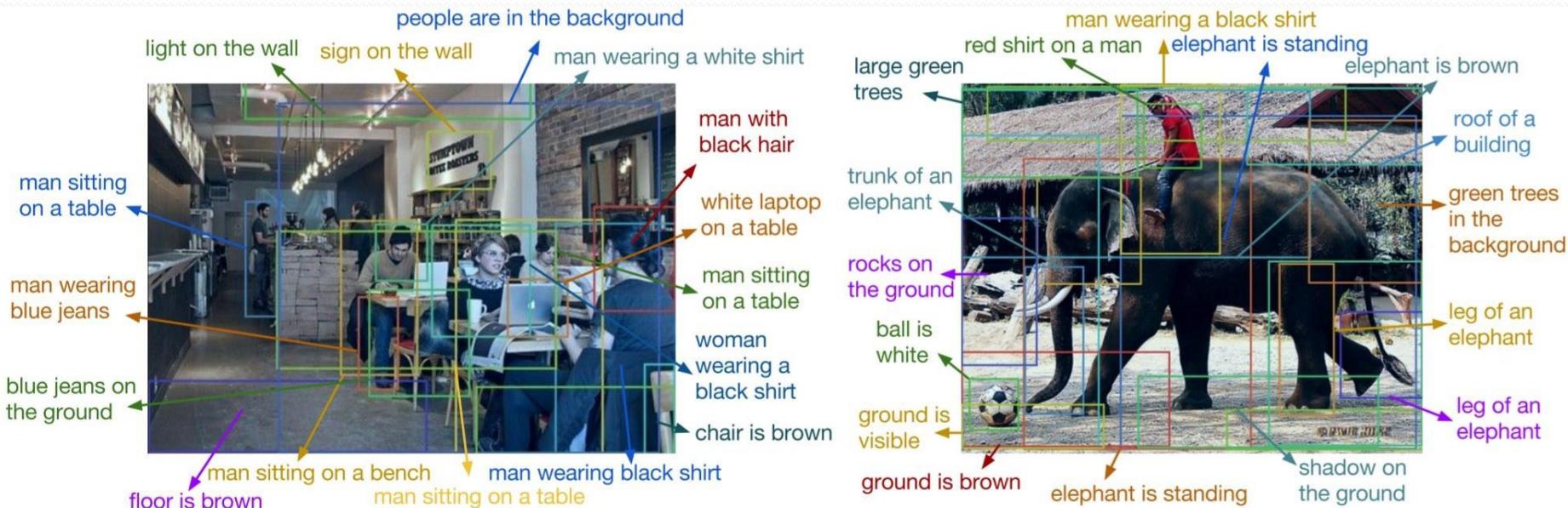
Instance Segmentation



DOG, DOG, CAT

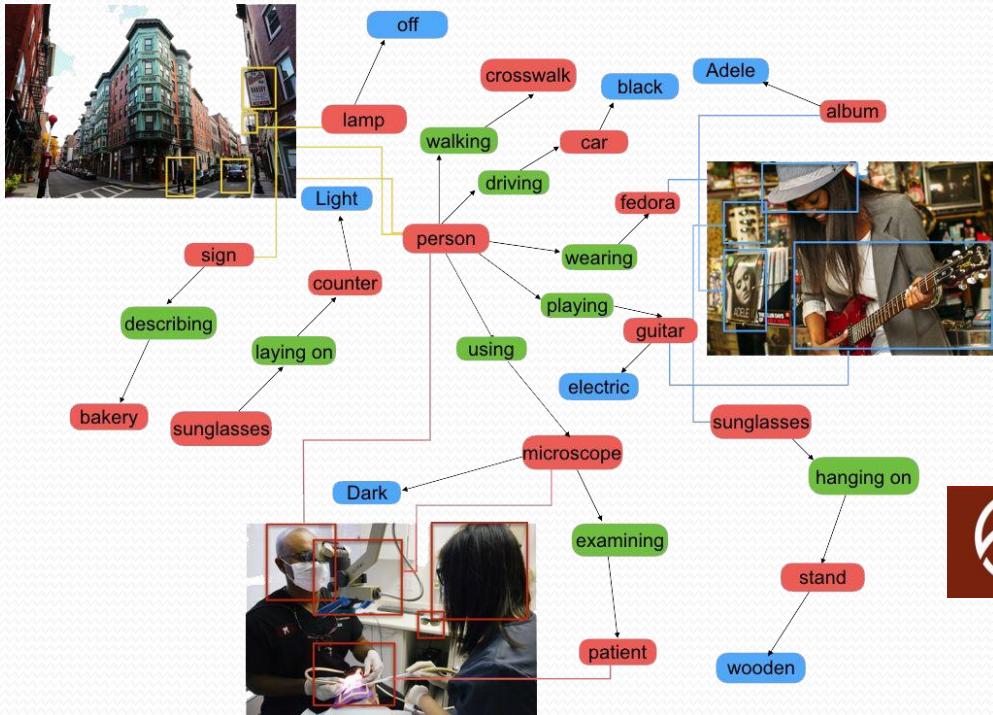
Beyond 2D Object Detection...

Object Detection + Captioning = Dense Captioning



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

Objects + Relationships = Scene Graphs

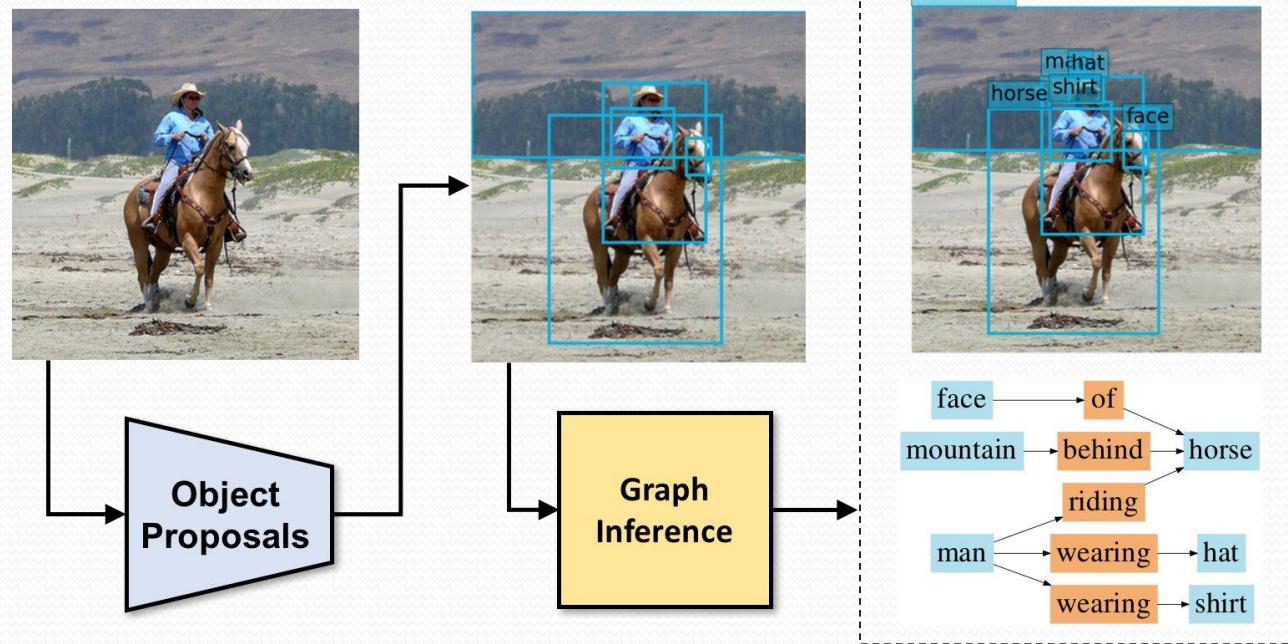


108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets



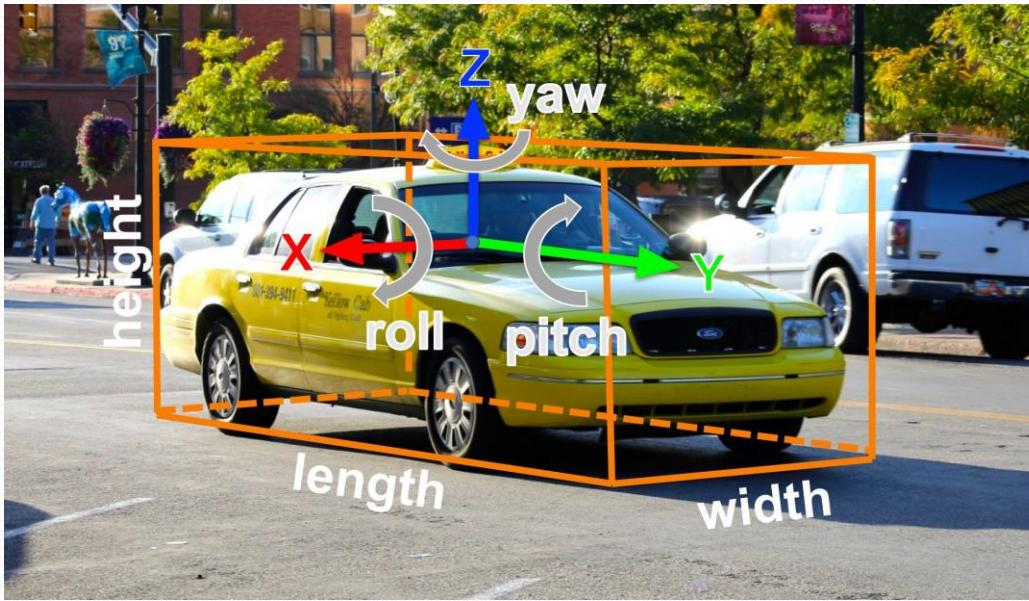
Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." International Journal of Computer Vision 123, no. 1 (2017): 32-73.

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)

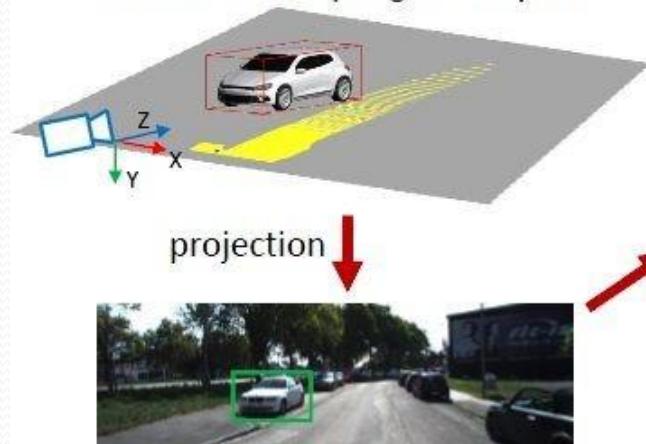
3D Object Detection:
3D oriented bounding box
 $(x, y, z, w, h, l, r, p, y)$

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

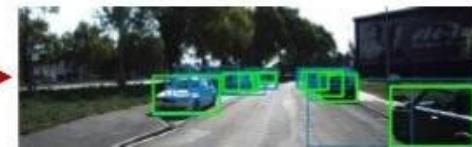
3D Object Detection: Monocular Camera

Candidate sampling in 3D space



Faster R-CNN

Scoring
&
NMS



Proposals

2D candidate boxes

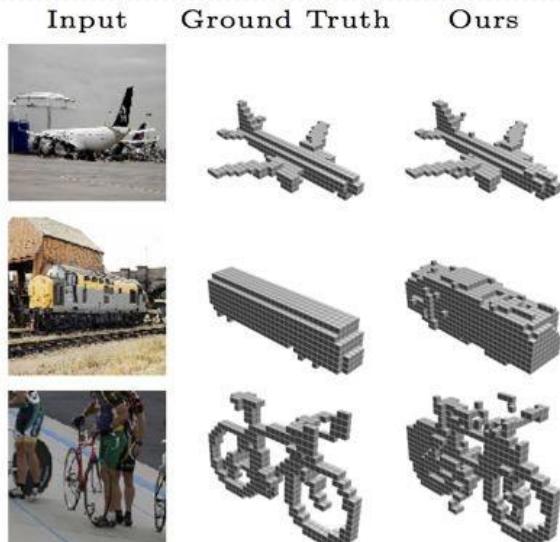
- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

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

3D Shape Prediction

Voxel:

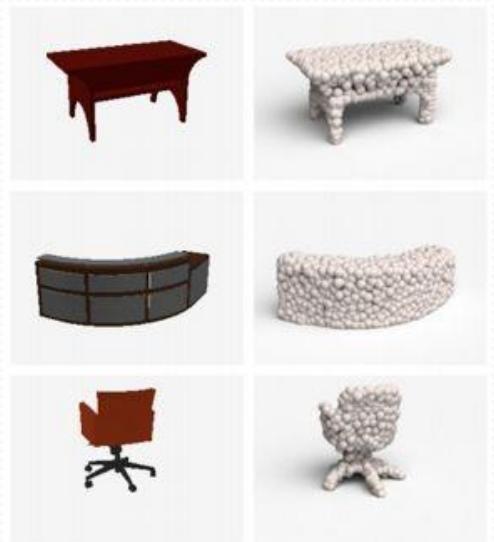
$D \times D \times D$ binary



Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

Pointcloud:

$V \times 3$ float



Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Mesh:

$V \times 3$ float, $F \times 3$ int



Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

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: Generative Models, Generative Adversarial Networks (GANs)