

DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2019 // JOY ARULRAJ

LECTURE #17: DETECTION AND SEGMENTATION

CREATING THE NEXT®



ADMINISTRIVIA

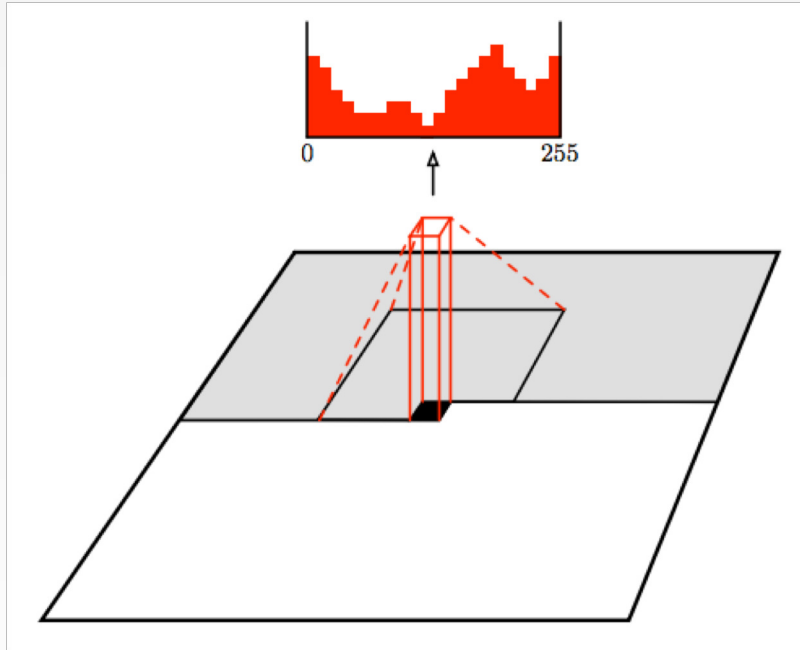
- Reminders
 - Assignment 2 due today
 - Guest lecture on Friday morning
 - Mid-term question bank shared on Piazza

EXTRA CREDIT

- Guest lecture
 - Model-based pricing for ML in a data market
 - Paris Koutris (Wisconsin)
 - **Friday (Oct 25), 11 AM – 12 PM**
 - Student center (Room 321)
 - **5% extra credit**

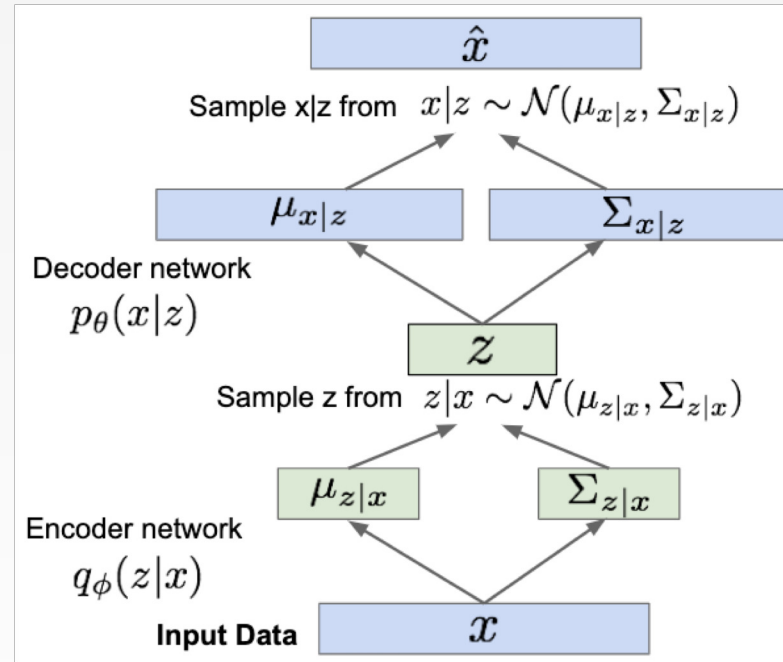
LAST TIME: GENERATIVE MODELS

Autoregressive models: PixelRNN, PixelCNN



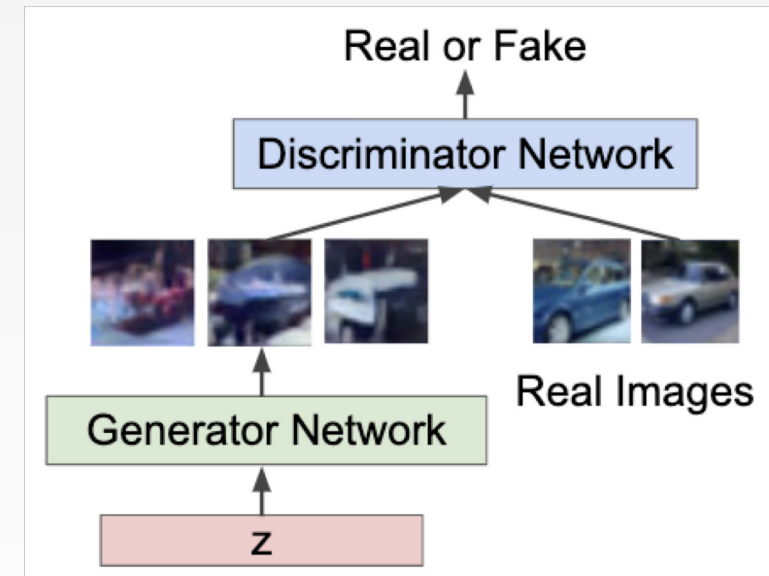
Van der Oord et al, "Conditional image generation with pixelCNN decoders", NIPS 2016

Variational Autoencoders



Kingma and Welling, "Auto-encoding variational bayes", ICLR 2013

Generative Adversarial Networks (GANs)



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

LAST TIME: GAN IMAGES

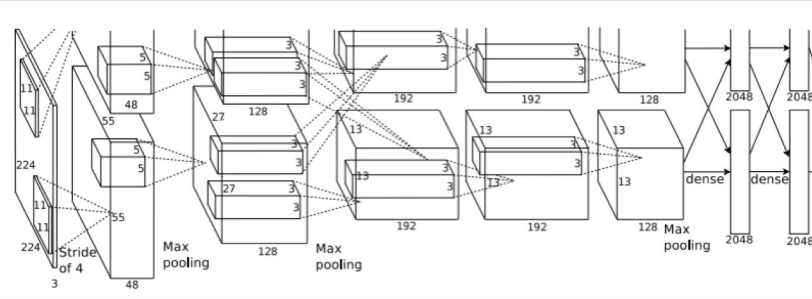


BigGAN, Brock et al., 2019



Progressive GAN, Karras 2018.

SO FAR: IMAGE CLASSIFICATION



**Feature
Vector:
4096**

Fully-Connected:
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01

...

COMPUTER VISION TASKS

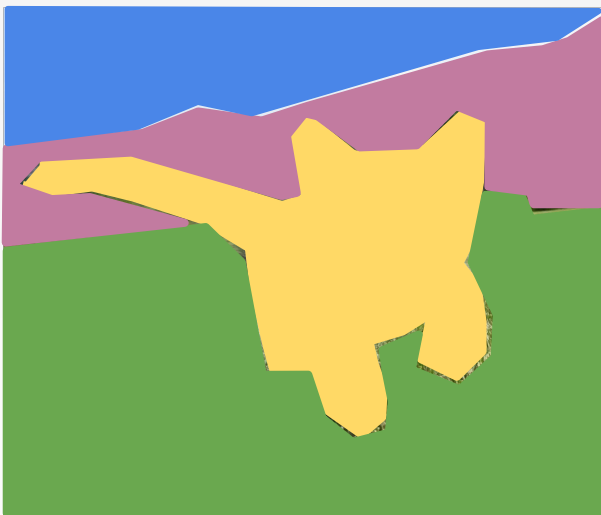
Classification



CAT

No spatial
extent

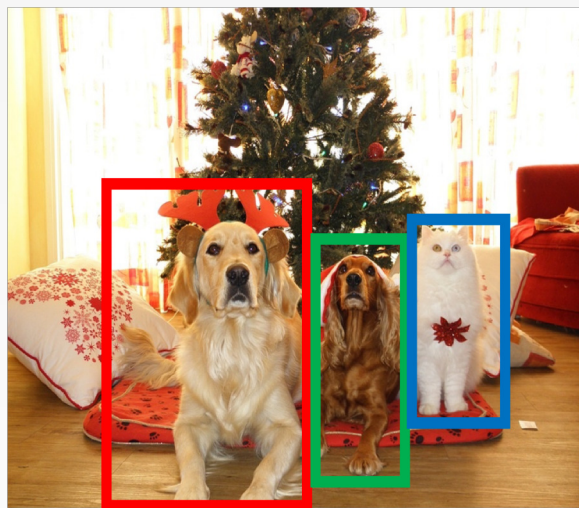
Semantic Segmentation



**GRASS, CAT,
TREE, SKY**

No objects, just
pixels

Object Detection



DOG, DOG, CAT

Multiple
Object

Instance Segmentation



DOG, DOG, CAT

This image is [CC0 public domain](#)

TODAY'S AGENDA

- Segmentation
- Detection



SEGMENTATION

SEMANTIC SEGMENTATION

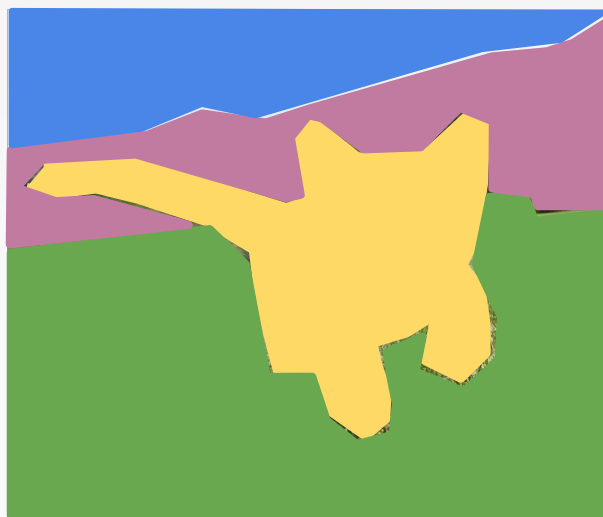
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation

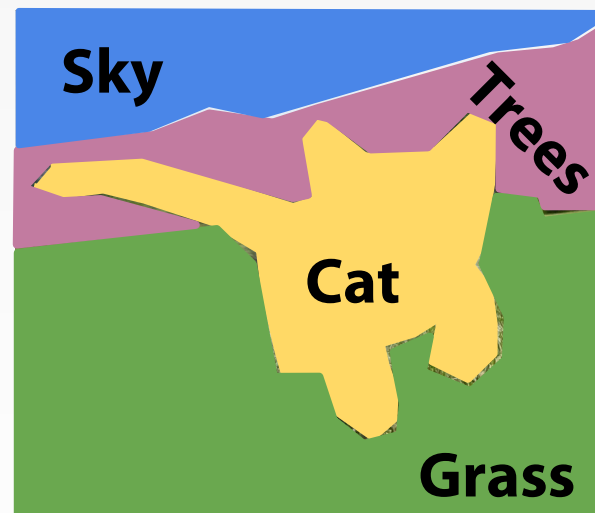


DOG, DOG, CAT

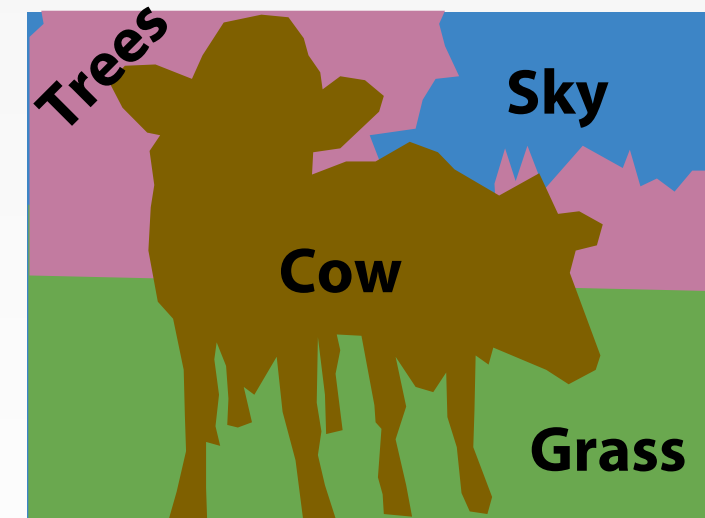
SEMANTIC SEGMENTATION

Label each pixel in the image with a category label

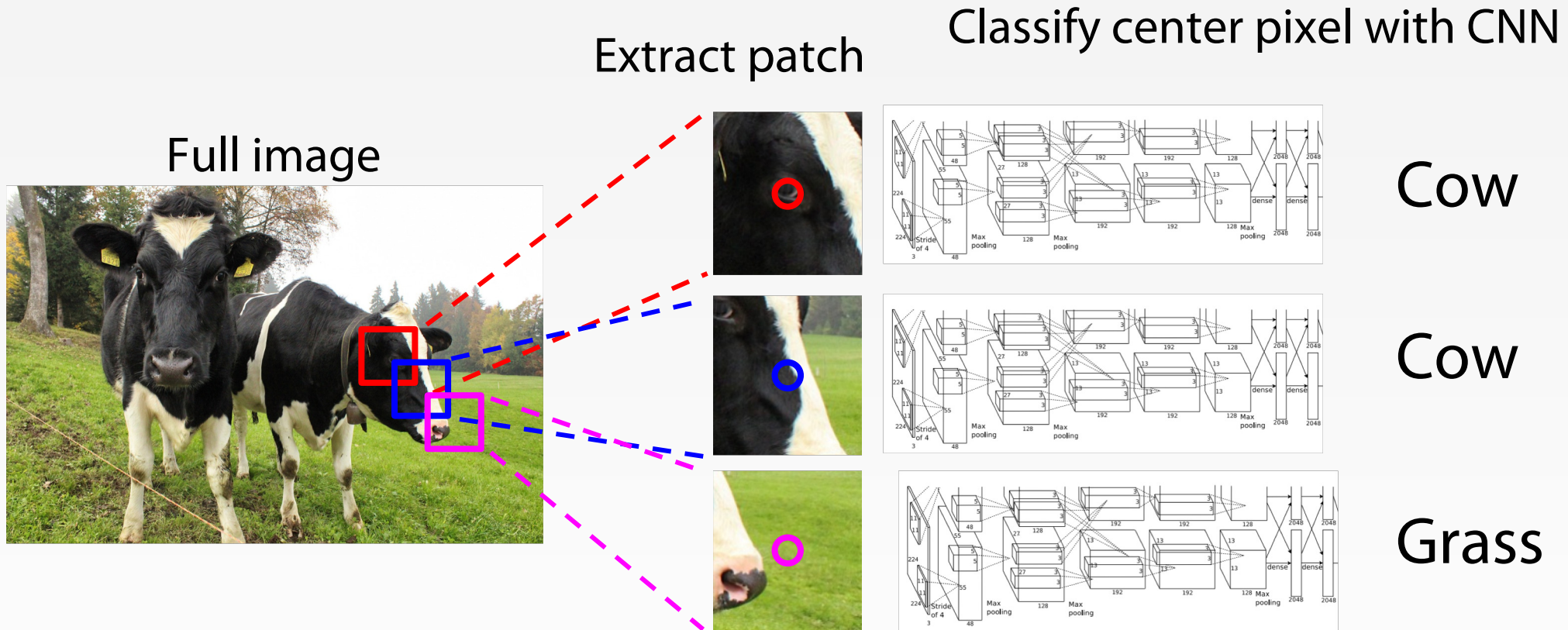
Don't differentiate instances, only care about pixels



[This image is CC0 public domain](#)



SEMANTIC SEGMENTATION IDEA: SLIDING WINDOW



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

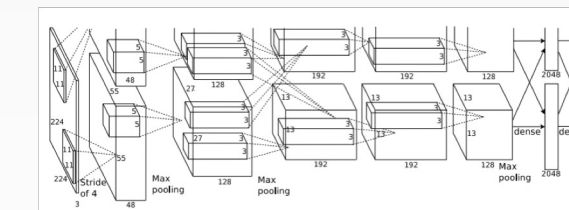
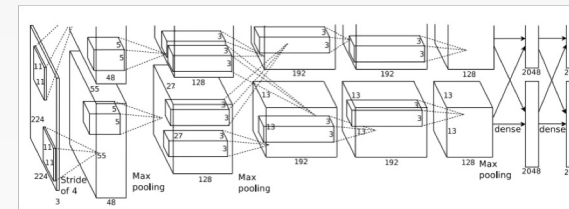
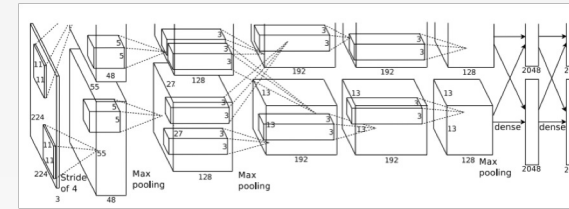
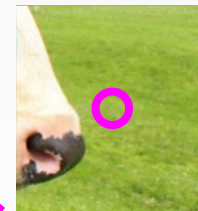
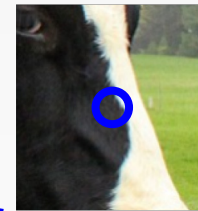
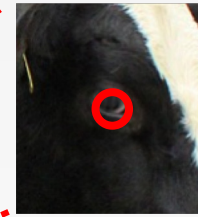
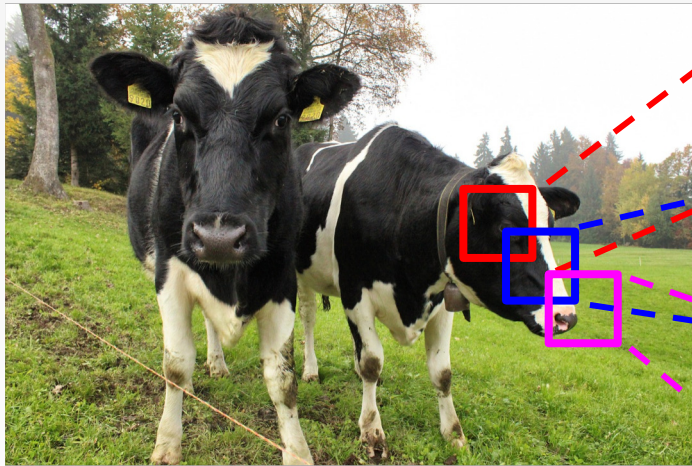
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

SEMANTIC SEGMENTATION IDEA: SLIDING WINDOW

Extract patch

Classify center pixel with CNN

Full image



Cow

Cow

Grass

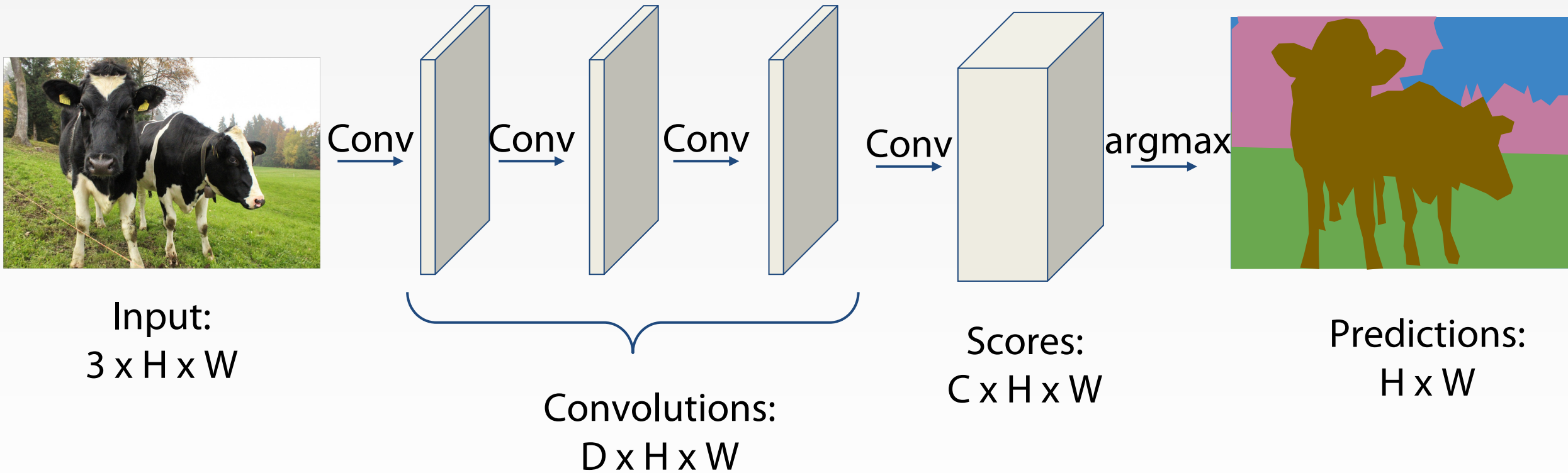
Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

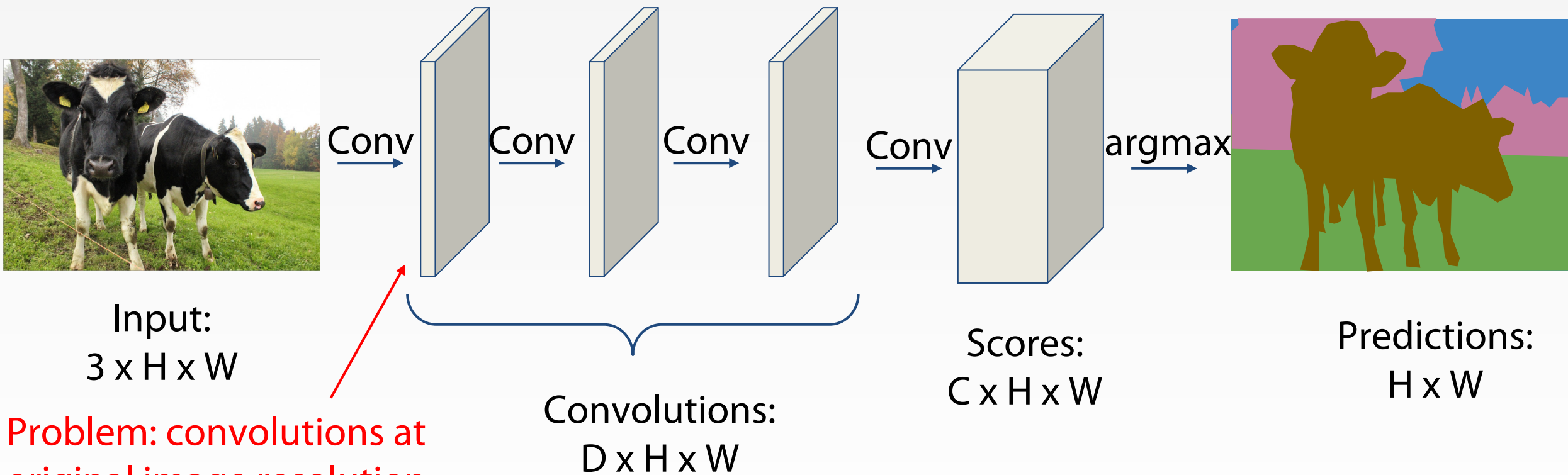
SEMANTIC SEGMENTATION IDEA: FULLY CONVOLUTIONAL

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!



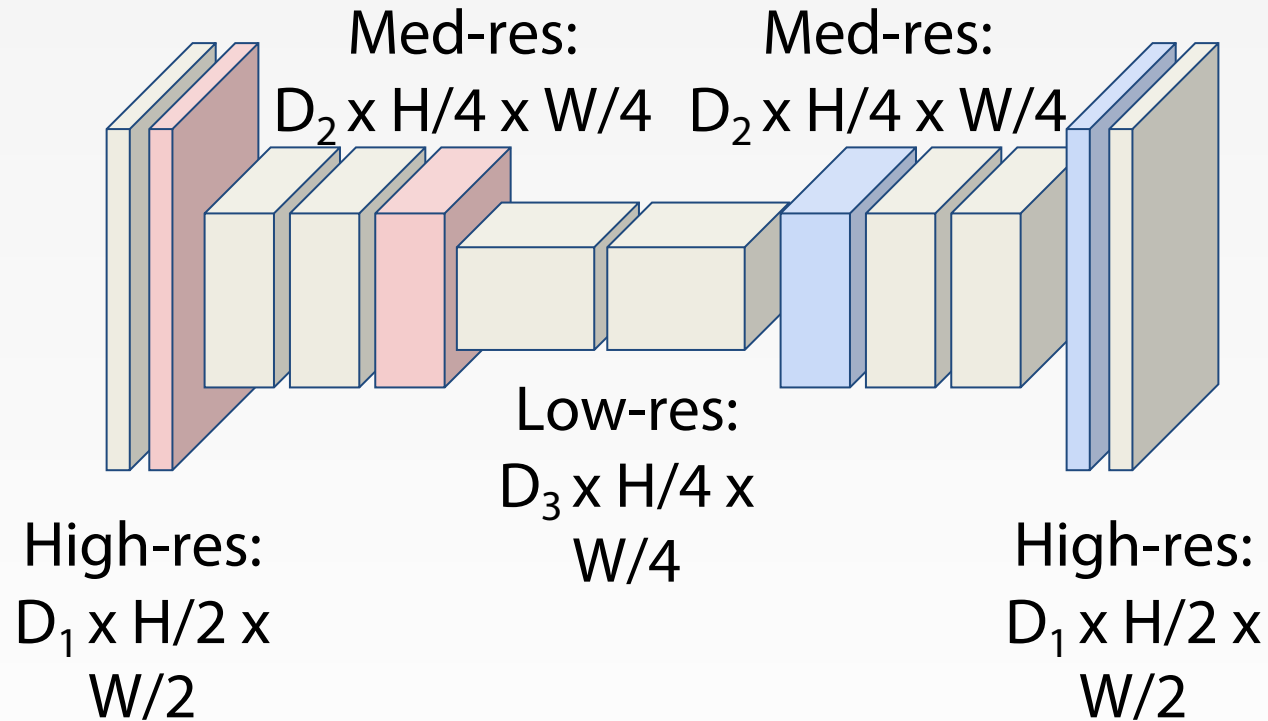
Problem: convolutions at original image resolution will be very expensive ...

SEMANTIC SEGMENTATION IDEA: FULLY CONVOLUTIONAL

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



Input:
 $3 \times H \times W$



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

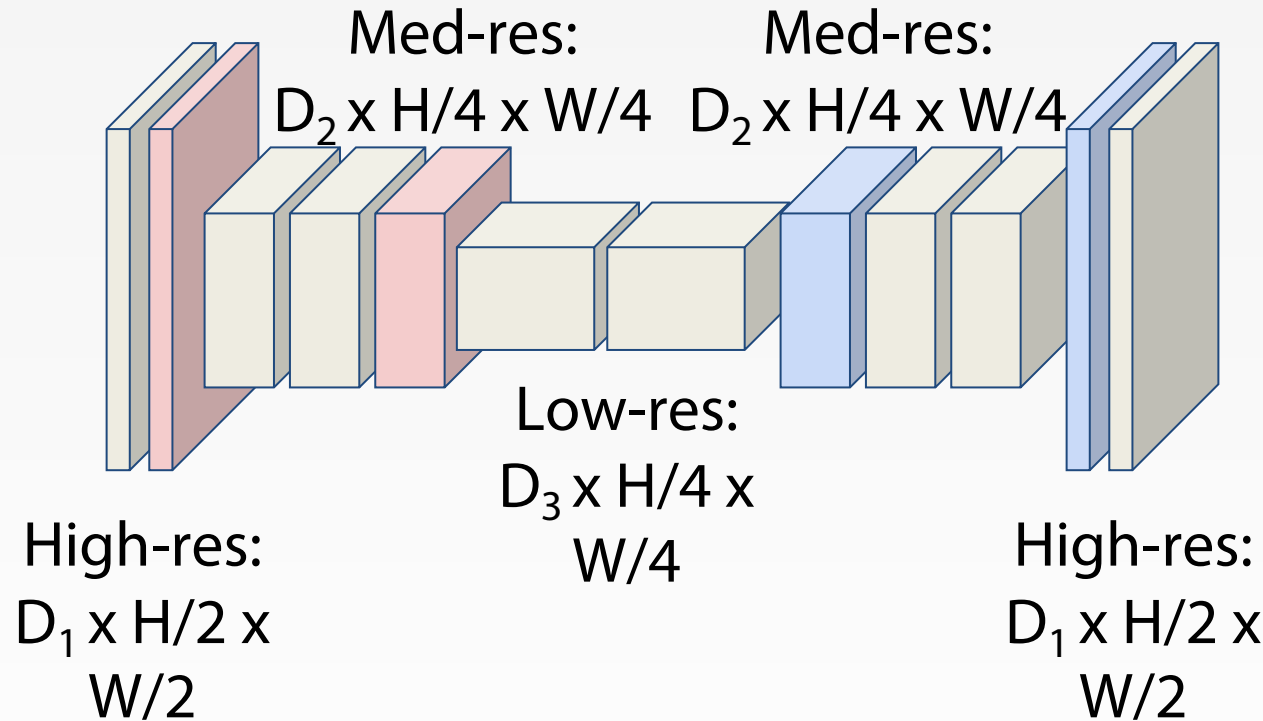
SEMANTIC SEGMENTATION IDEA: FULLY CONVOLUTIONAL

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

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$



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

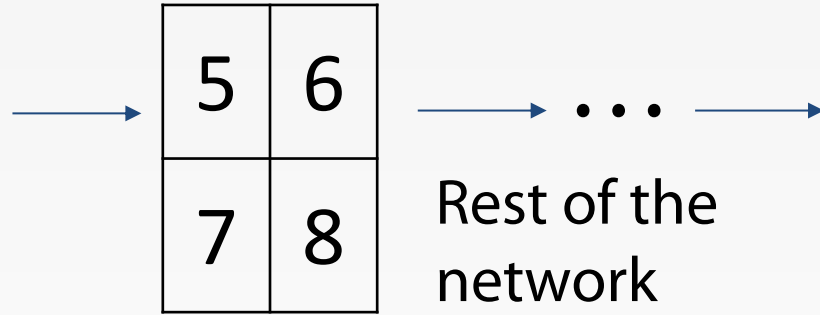
IN-NETWORK UPSAMPLING: "MAX UNPOOLING"

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



Output: 2 x 2

Rest of the network

Max Unpooling

Use positions from pooling layer

1	2
3	4

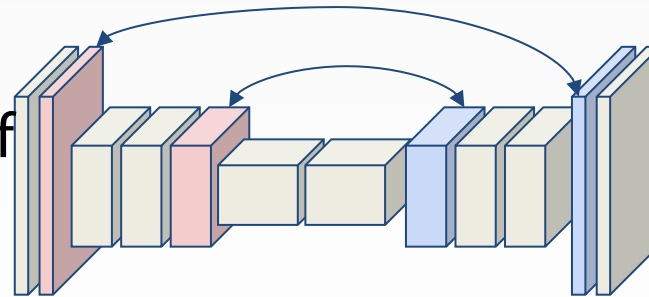
Input: 2 x 2

→

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

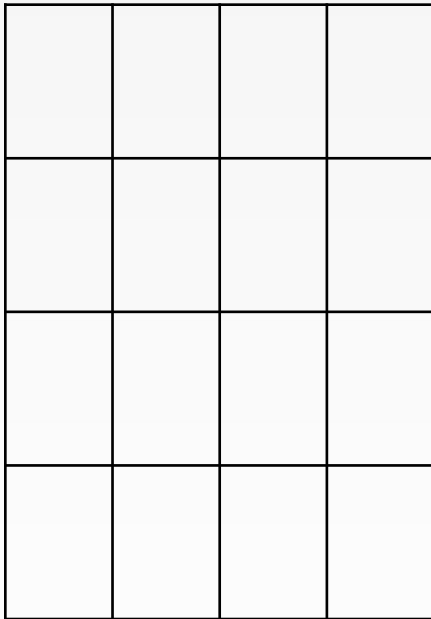
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

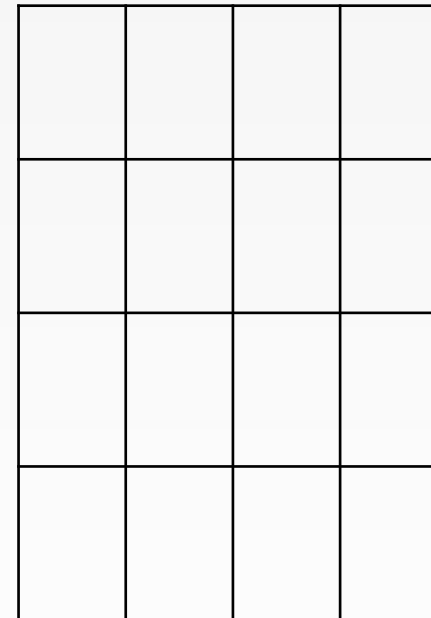


LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



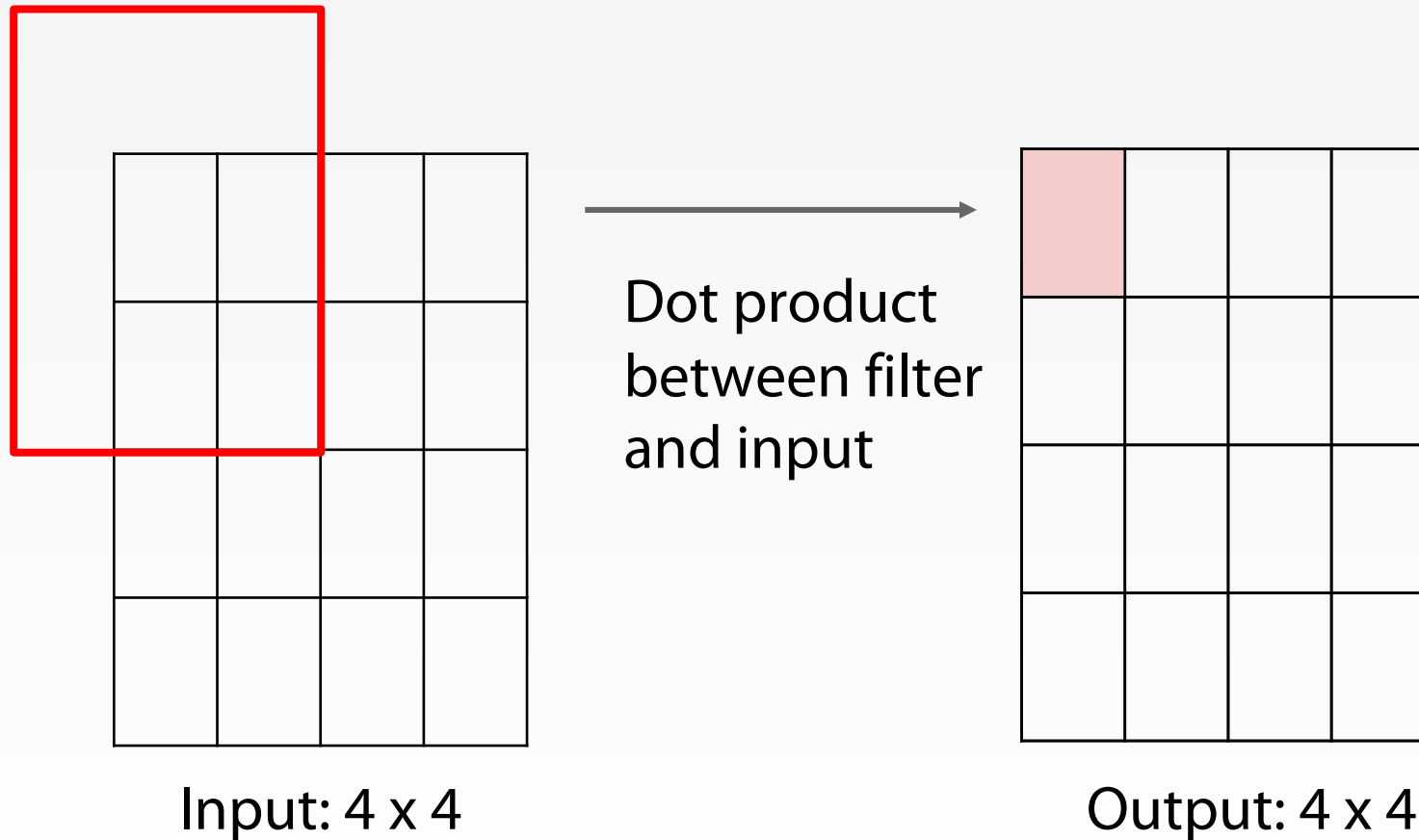
Input: 4 x 4



Output: 4 x 4

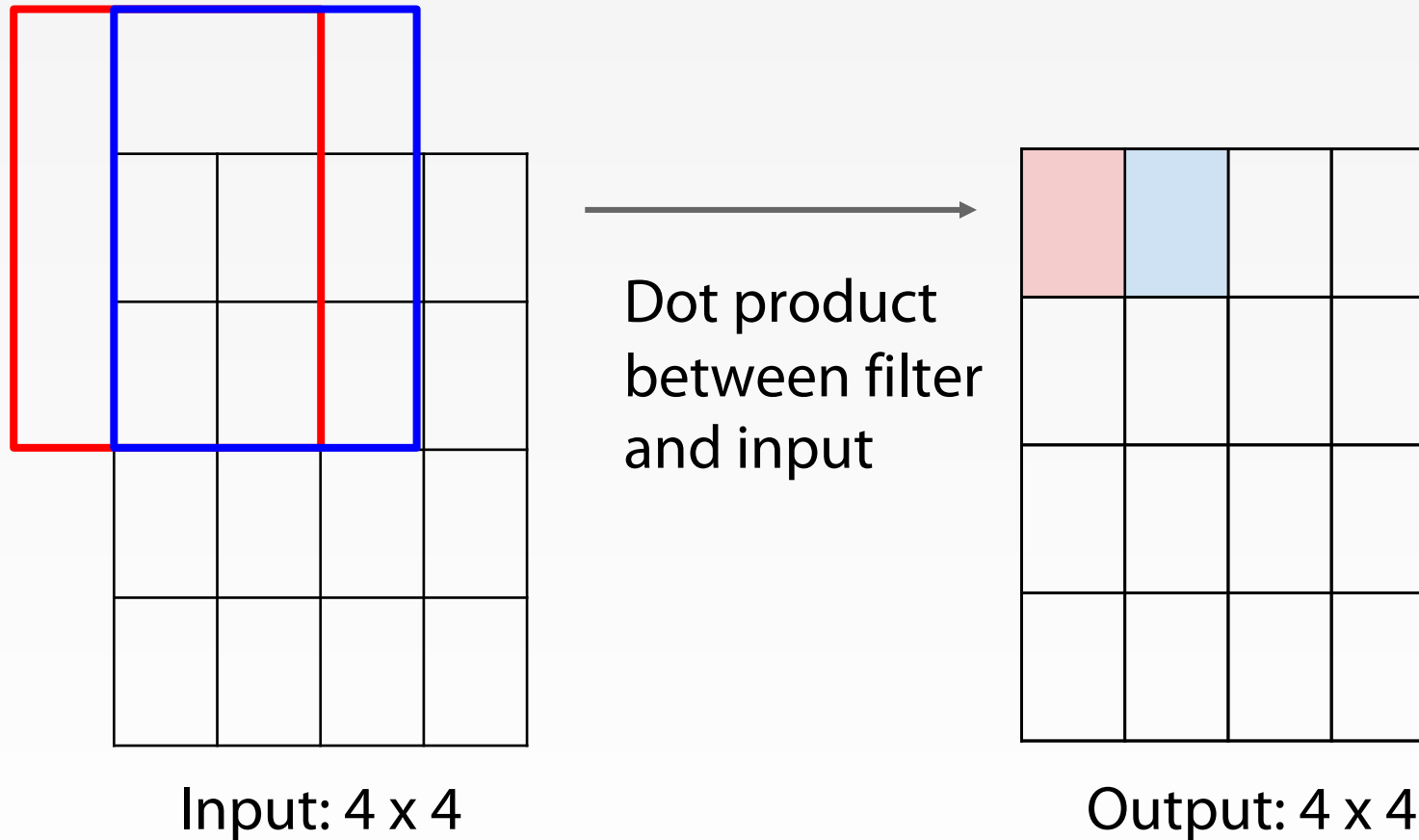
LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



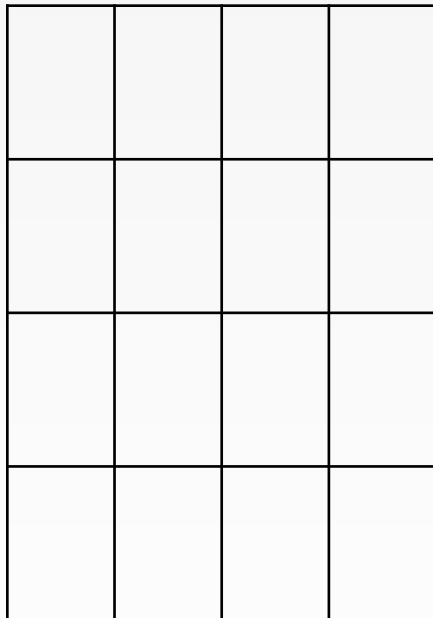
LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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

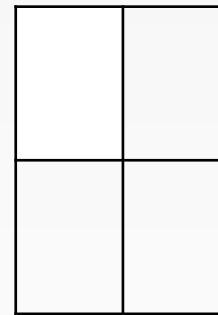


LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



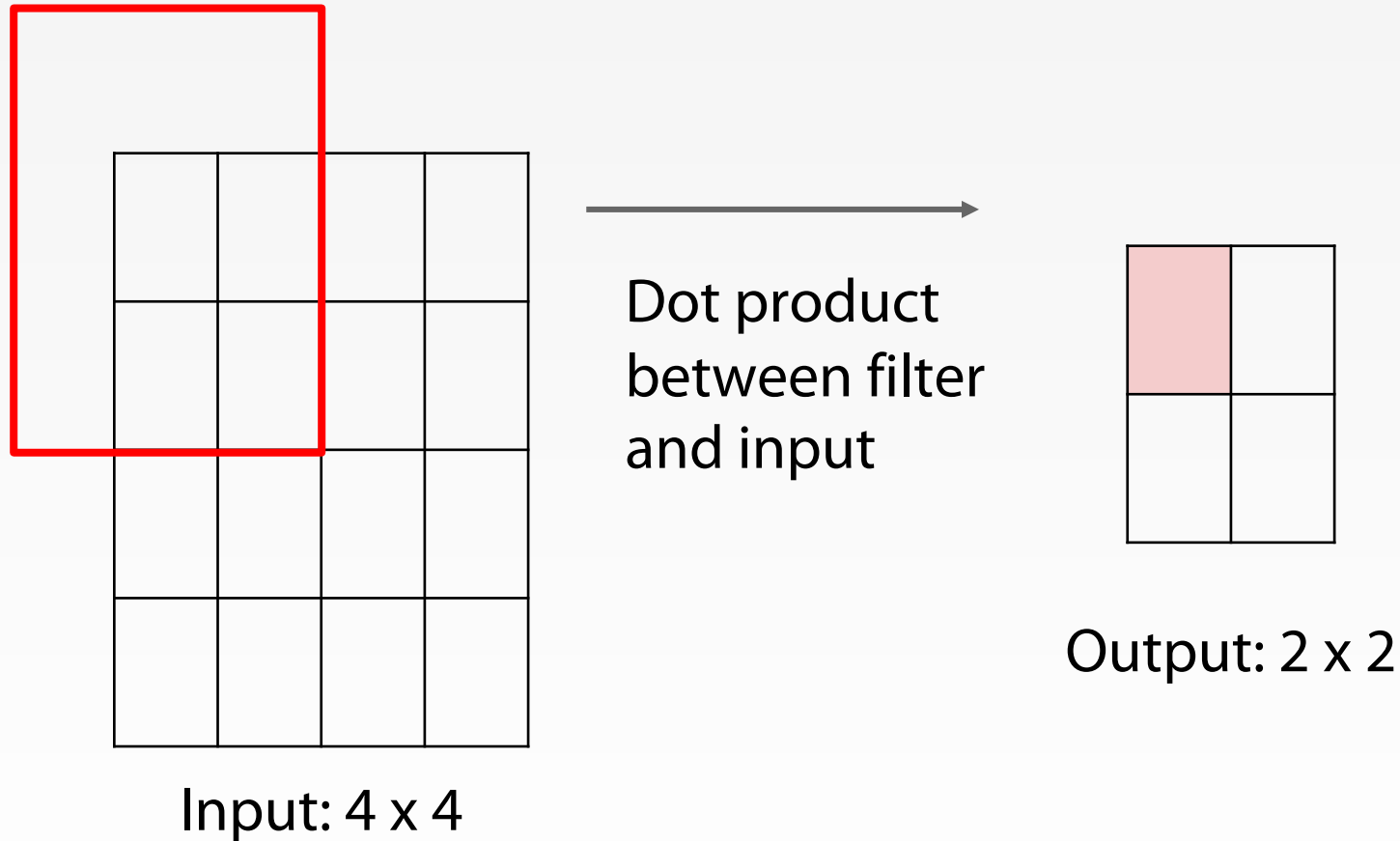
Input: 4 x 4



Output: 2 x 2

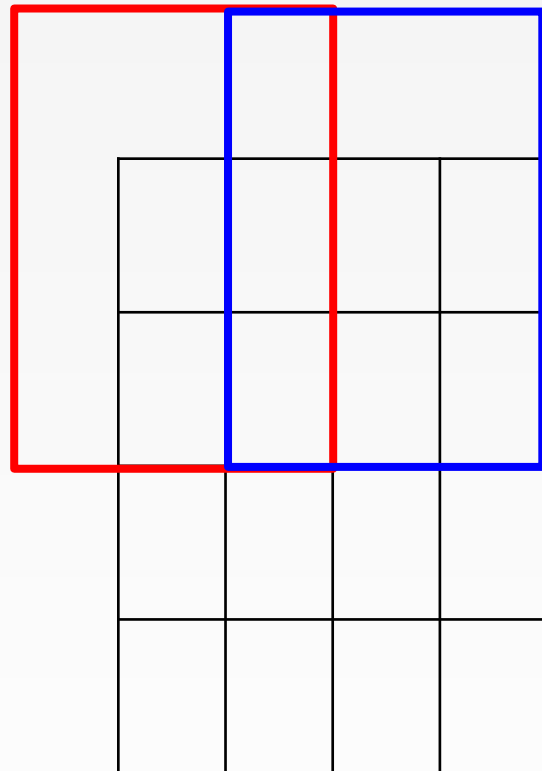
LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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

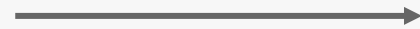


LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

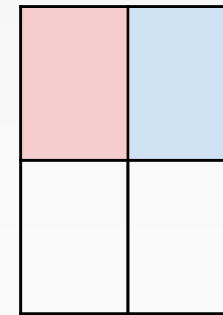
Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4



Dot product
between filter
and input



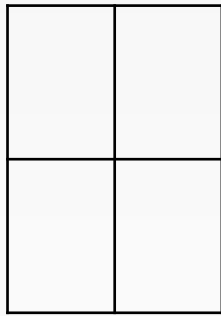
Output: 2 x 2

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

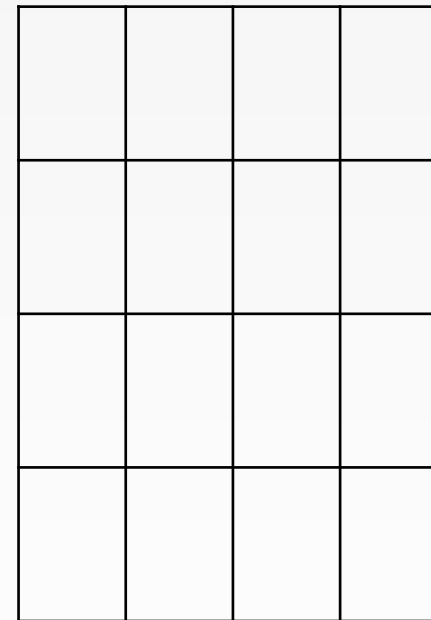
Stride gives ratio between movement in input and output

LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



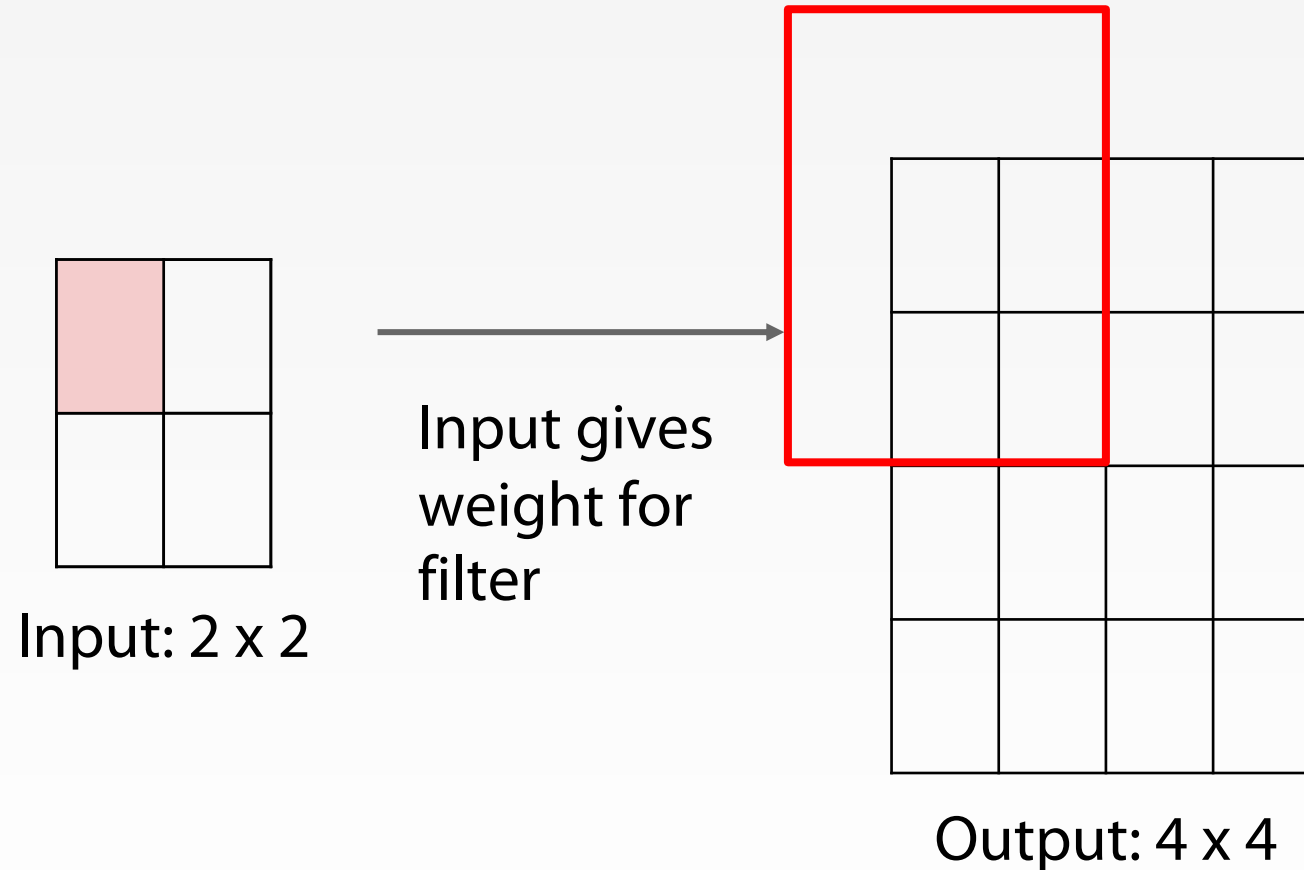
Input: 2 x 2



Output: 4 x 4

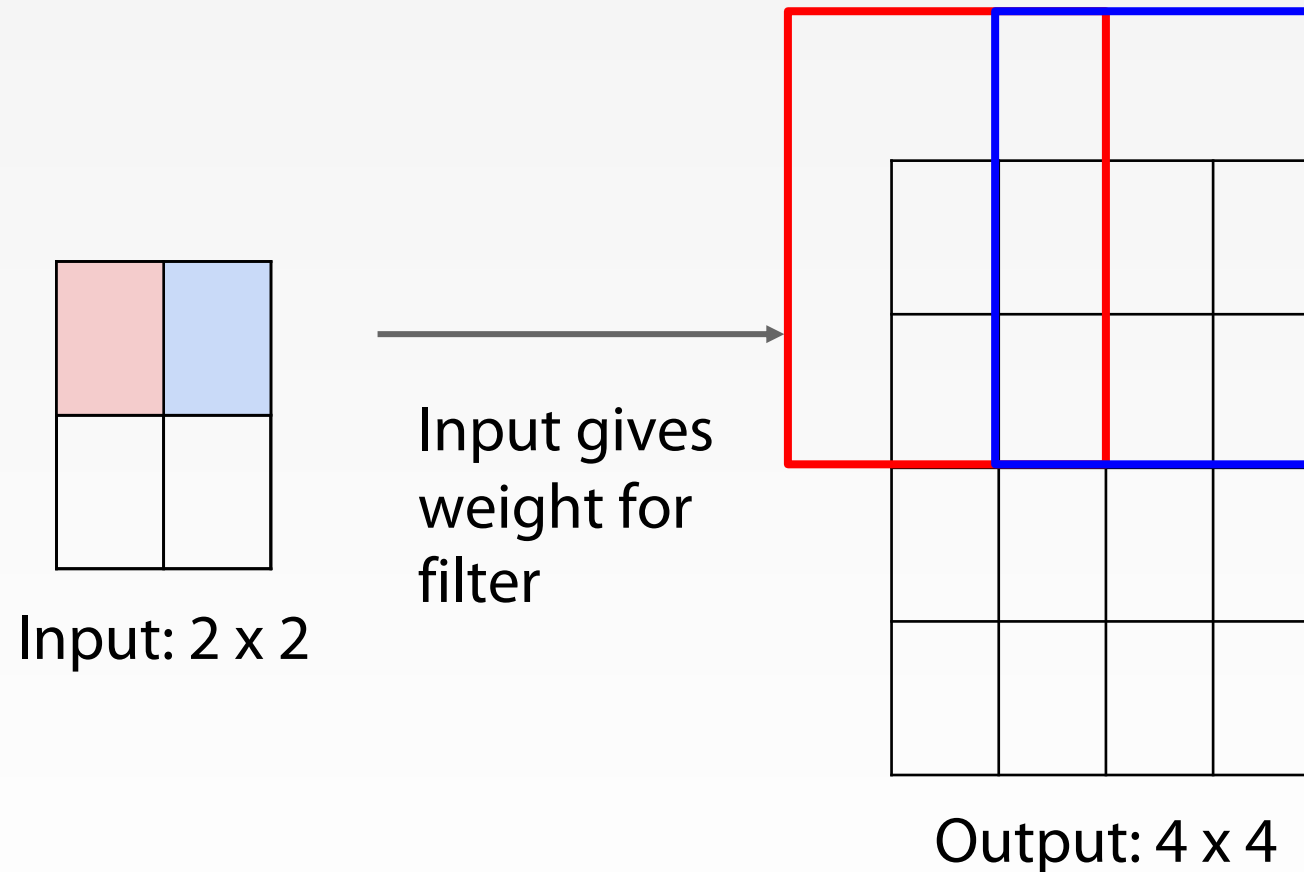
LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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

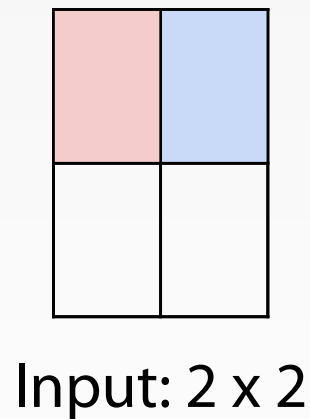


Filter moves 2 pixels in the output for every one pixel in the input

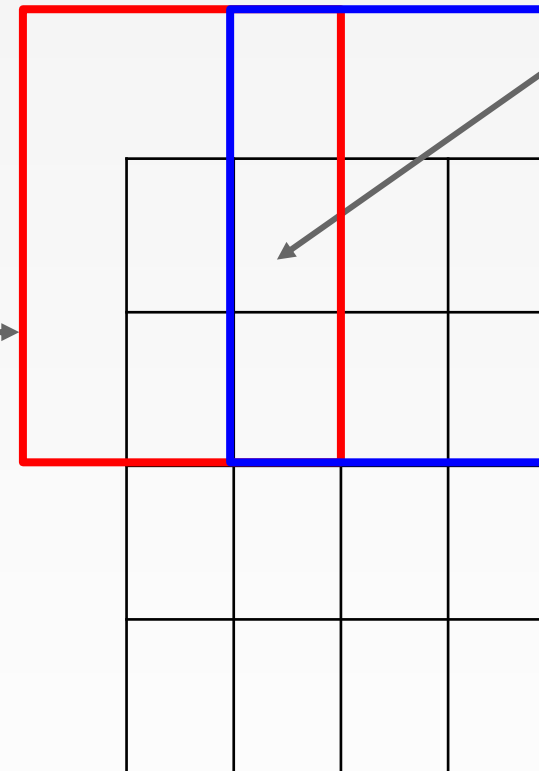
Stride gives ratio between movement in output and input

LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

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



Input gives weight for filter



Sum where output overlaps

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

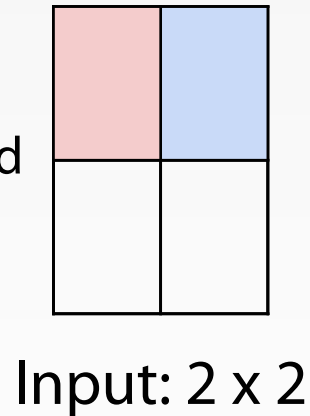
Stride gives ratio between movement in output and input

LEARNABLE UPSAMPLING: TRANSPOSE CONVOLUTION

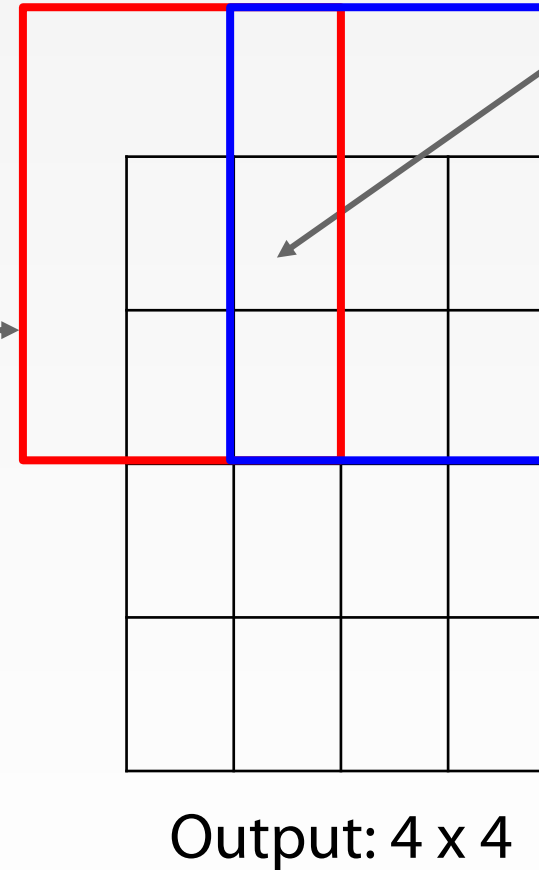
Other names:

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

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



Input gives weight for filter

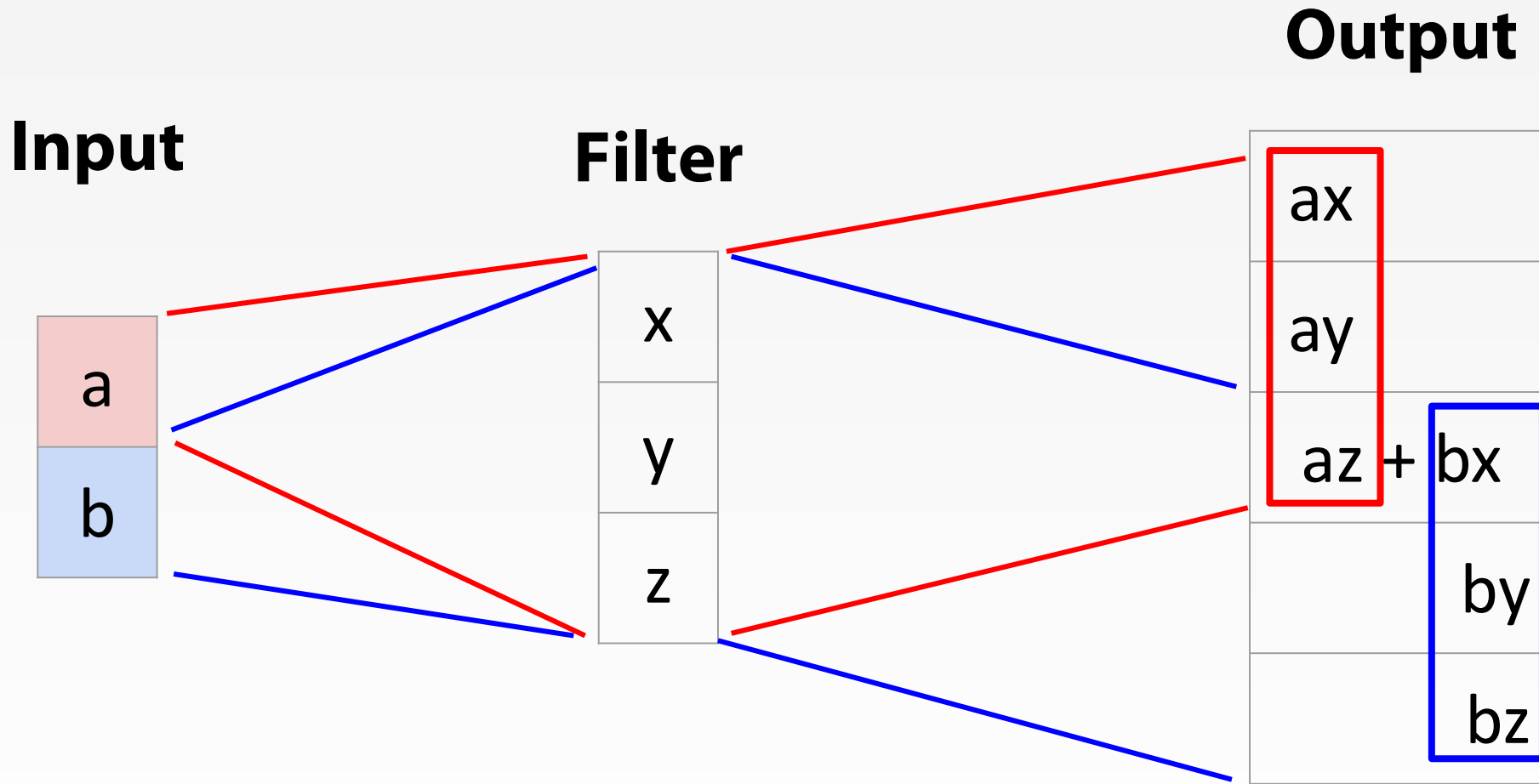


Sum where output overlaps

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: 1D EXAMPLE



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

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

CONVOLUTION AS MATRIX MULTIPLICATION (1D EXAMPLE)

We can express convolution in terms of a matrix multiplication

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

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

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

CONVOLUTION AS MATRIX MULTIPLICATION (1D EXAMPLE)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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

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

CONVOLUTION AS MATRIX MULTIPLICATION (1D EXAMPLE)

We can express convolution in terms of a matrix multiplication

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

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

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

CONVOLUTION AS MATRIX MULTIPLICATION (1D EXAMPLE)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution 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!

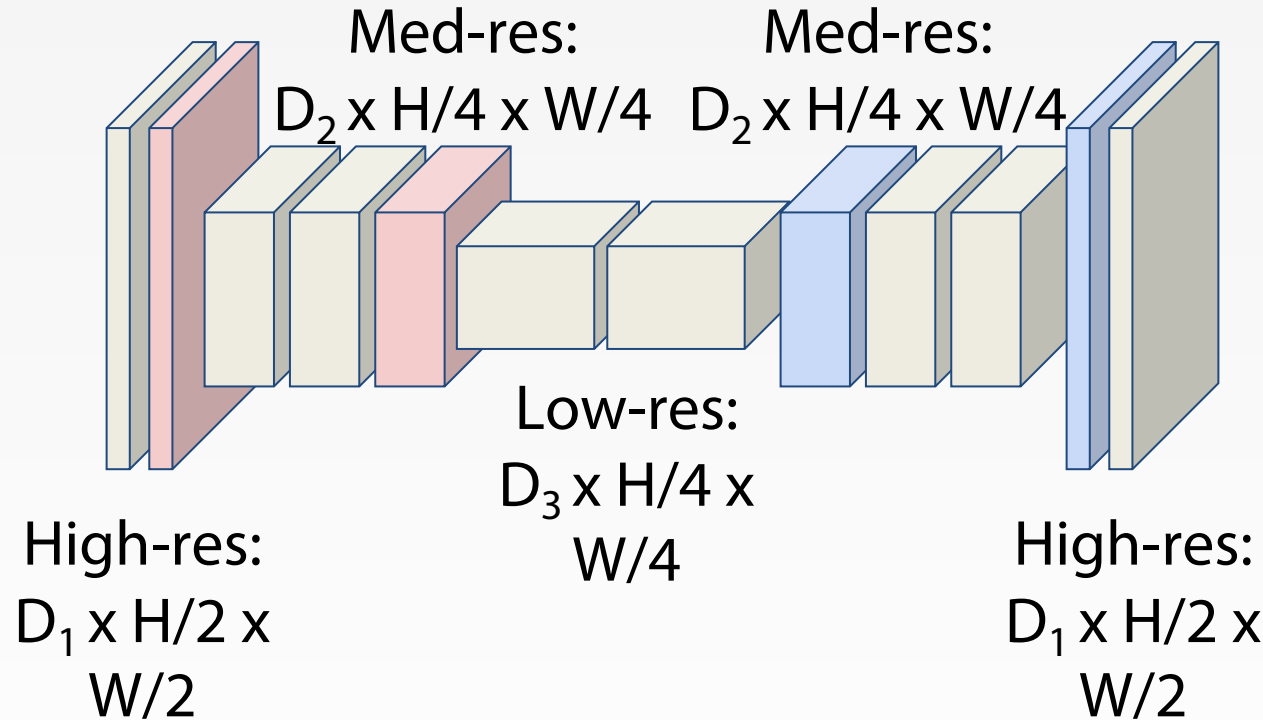
SEMANTIC SEGMENTATION IDEA: FULLY CONVOLUTIONAL

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

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$



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

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

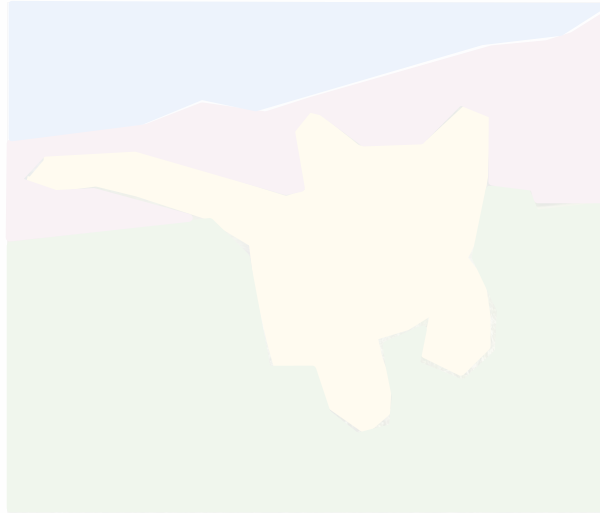
Classification



CAT

No spatial extent

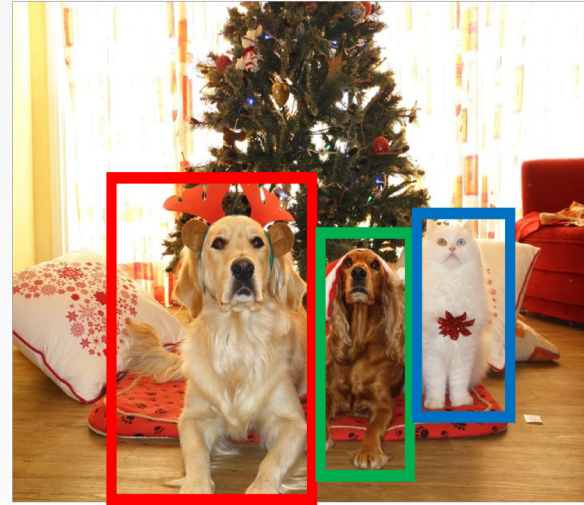
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

OBJECT DETECTION: IMPACT OF DEEP LEARNING

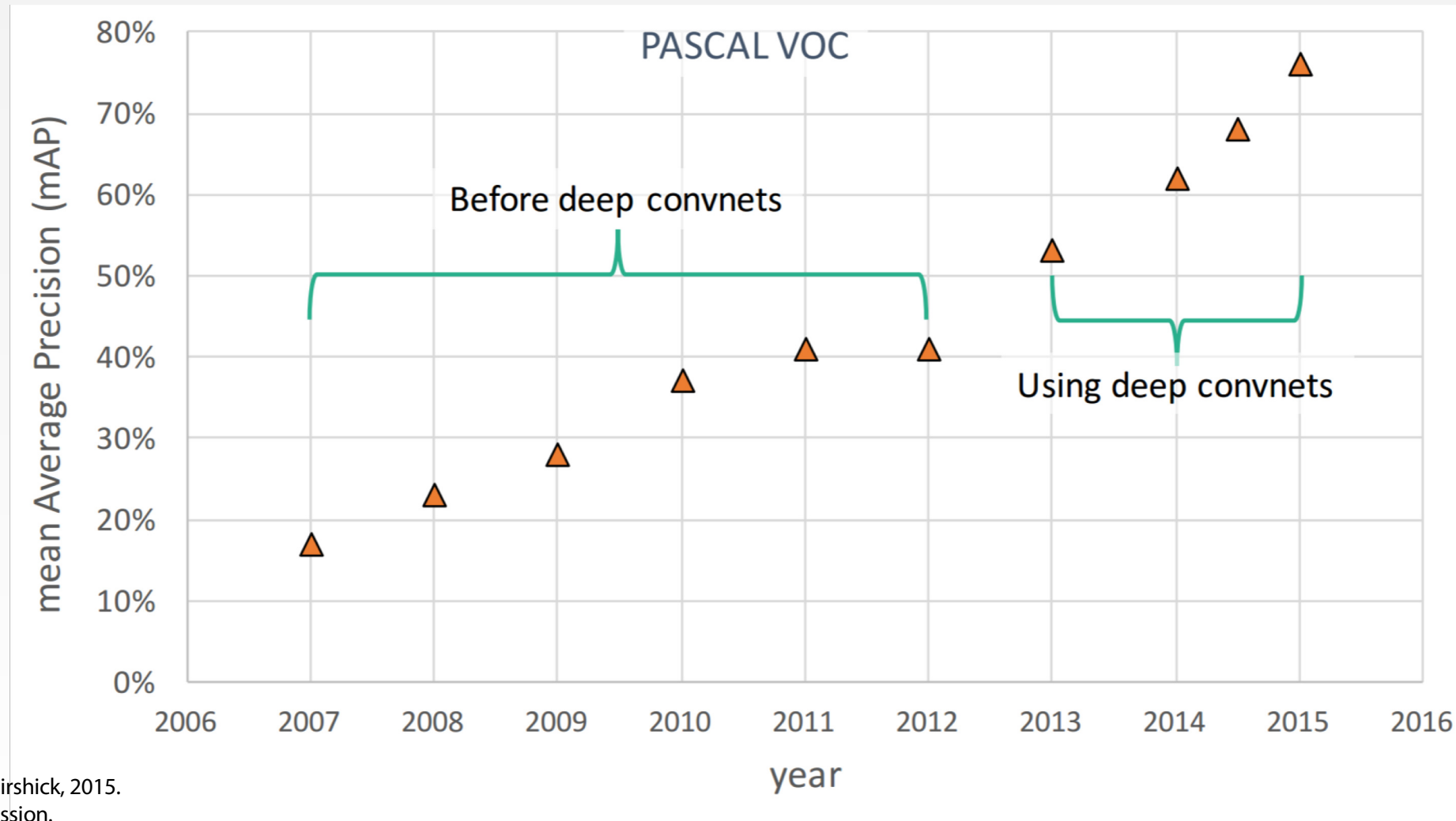


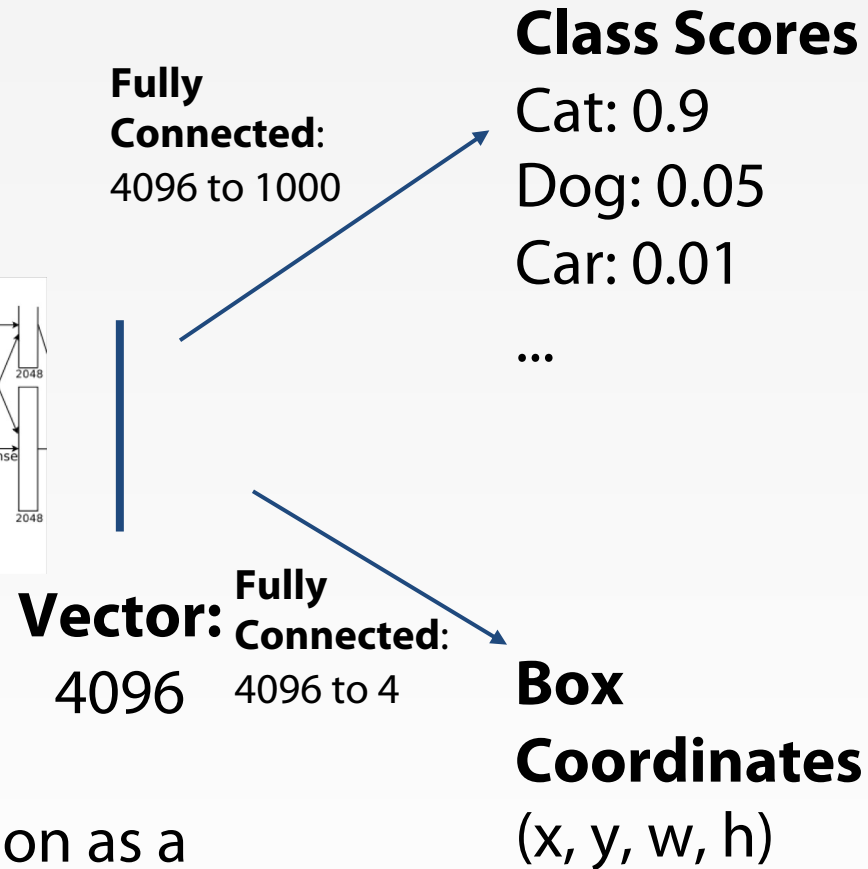
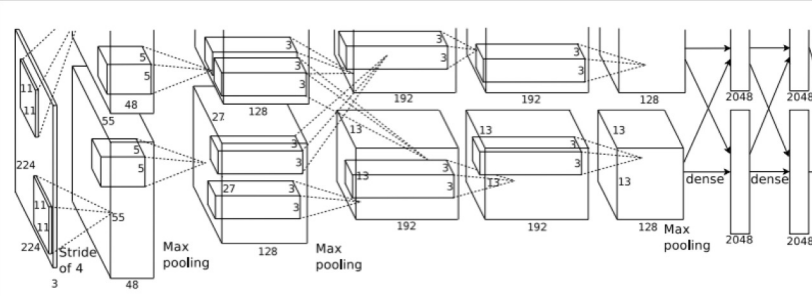
Figure copyright Ross Girshick, 2015.
Reproduced with permission.

OBJECT DETECTION: SINGLE OBJECT

(CLASSIFICATION + LOCALIZATION)



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



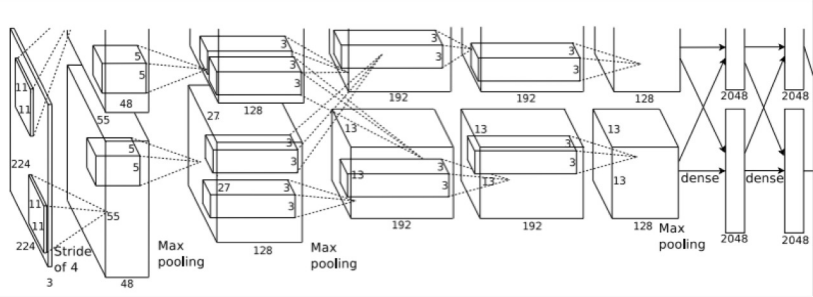
Treat localization as a regression problem!

OBJECT DETECTION: SINGLE OBJECT

(CLASSIFICATION + LOCALIZATION)



This image is [CC0 public domain](#)



Fully Connected:
4096 to 1000

Class Scores

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

Correct label:

Cat

Softmax Loss

Vector: Fully Connected:
4096 4096 to 4

Box

Coordinates

(x, y, w, h)

L2 Loss

Correct box:

(x', y', w', h')

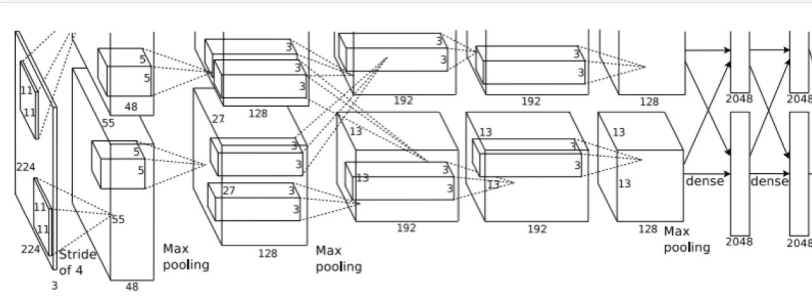
Treat localization as a regression problem!

OBJECT DETECTION: SINGLE OBJECT

(CLASSIFICATION + LOCALIZATION)



This image is [CC0 public domain](#)



Vector: 4096
Fully Connected: 4096 to 4

Treat localization as a regression problem!

Fully Connected: 4096 to 1000

Class Scores

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

Multitask Loss

Correct label:

Cat

Softmax Loss

+ **Loss**

Box Coordinates

(x, y, w, h)

L2 Loss

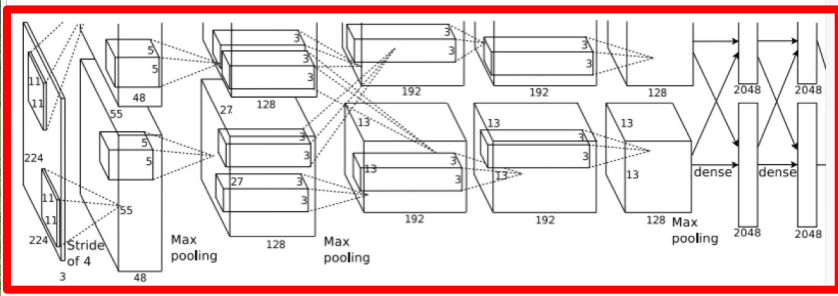
Correct box:
(x', y', w', h')

OBJECT DETECTION: SINGLE OBJECT

(CLASSIFICATION + LOCALIZATION)



This image is [CC0 public domain](#)



Often pretrained on
ImageNet
(Transfer learning)

Treat localization as a
regression problem!

Fully
Connected:
4096 to 1000

Class Scores

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

Multitask Loss

Vector: Fully
4096 Connected:
4096 to 4

Box

Coordinates
(x, y, w, h)

Correct label:

Cat

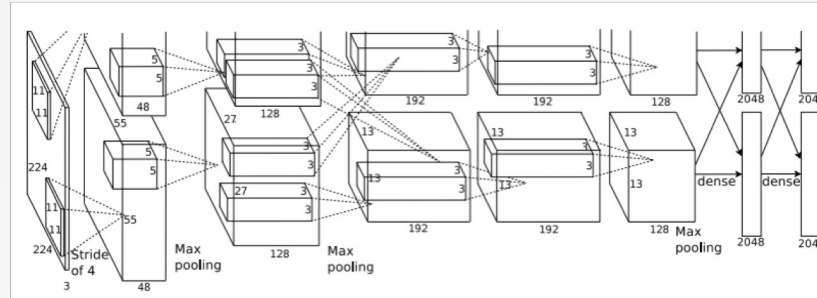
Softmax
Loss

+ Loss

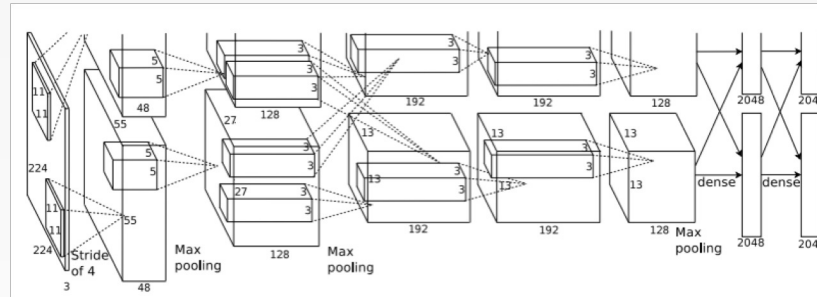
L2 Loss

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

OBJECT DETECTION: MULTIPLE OBJECTS



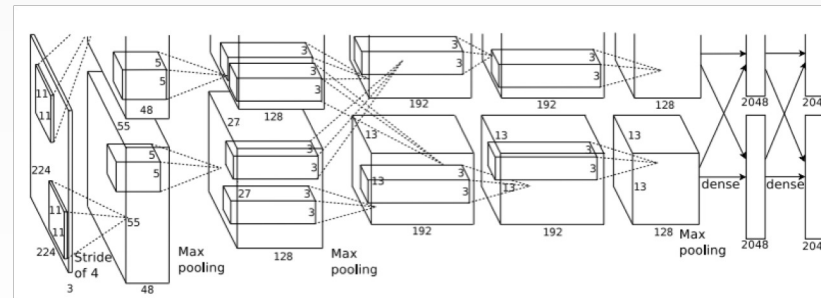
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



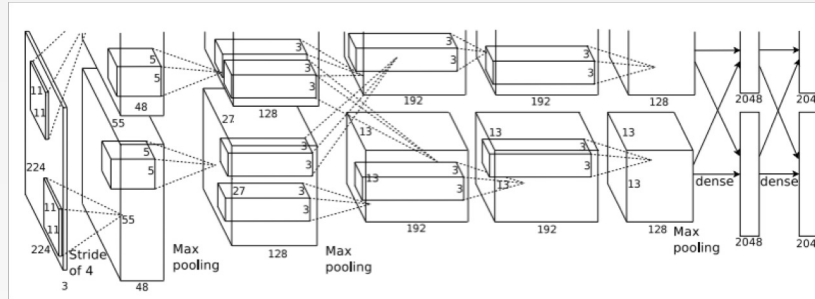
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

....

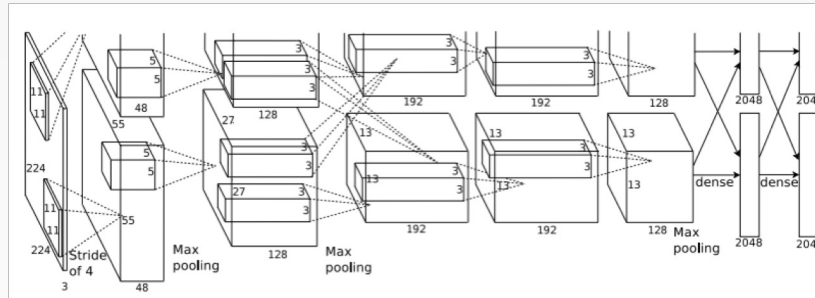
OBJECT DETECTION: MULTIPLE OBJECTS

Each image needs a different number of outputs!



CAT: (x, y, w, h)

4 numbers

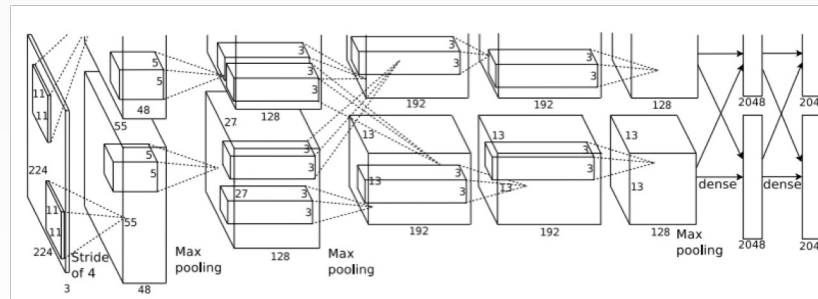


DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)

16 numbers



DUCK: (x, y, w, h)

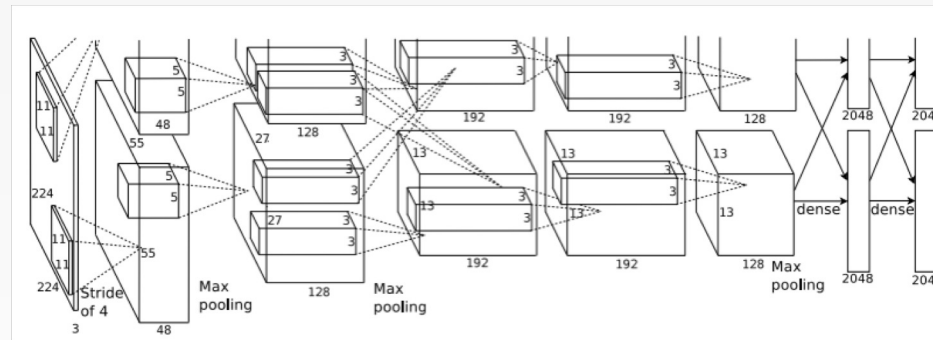
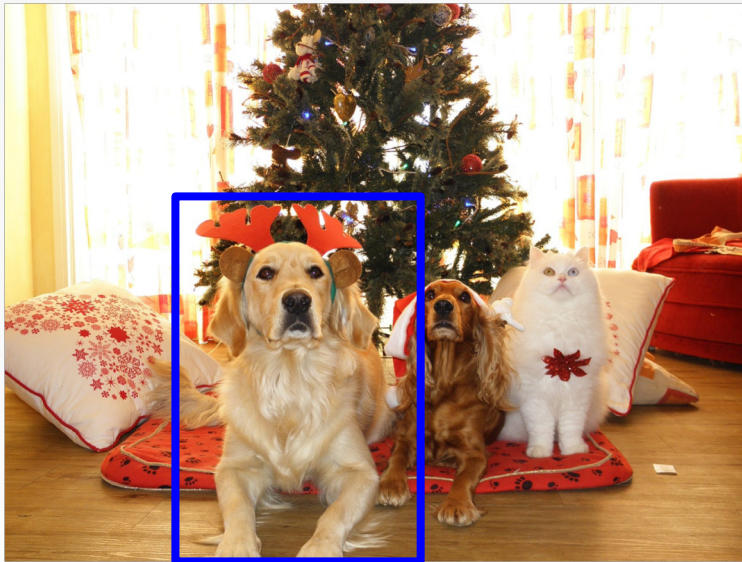
DUCK: (x, y, w, h)

....

Many numbers!

OBJECT DETECTION: MULTIPLE OBJECTS

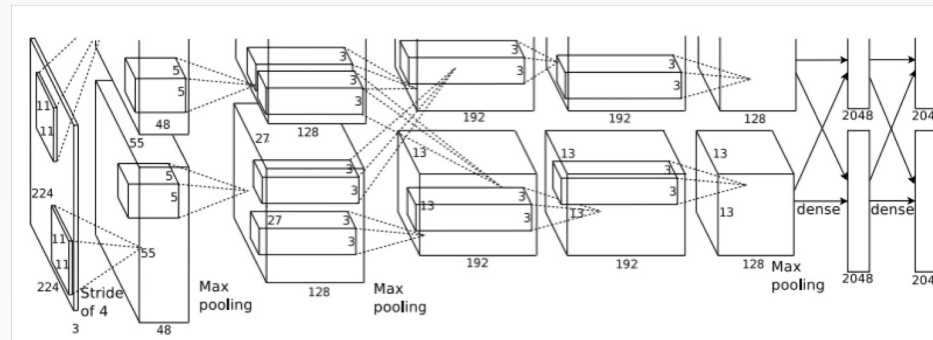
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

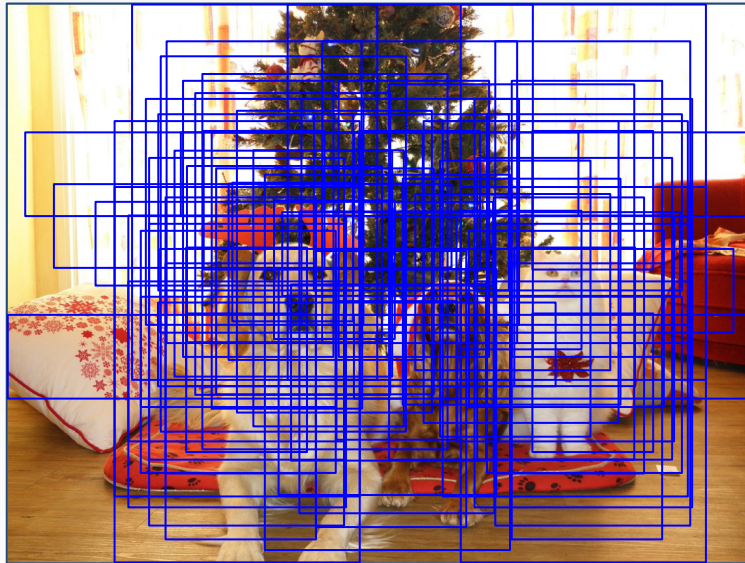
OBJECT DETECTION: MULTIPLE OBJECTS

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

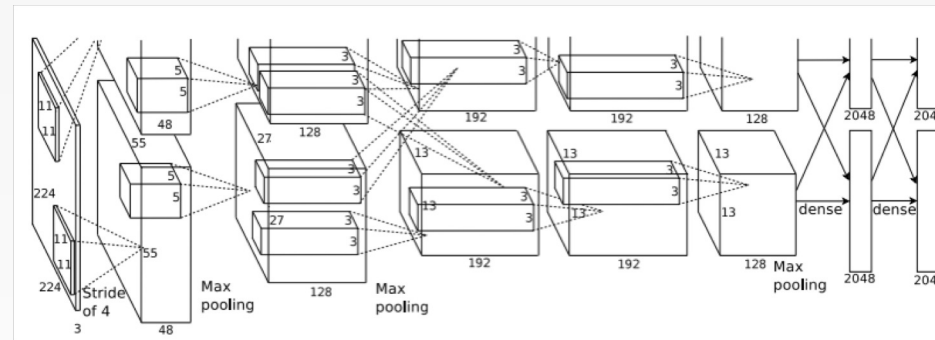


Dog? NO
Cat? YES
Background? NO

OBJECT DETECTION: MULTIPLE OBJECTS



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

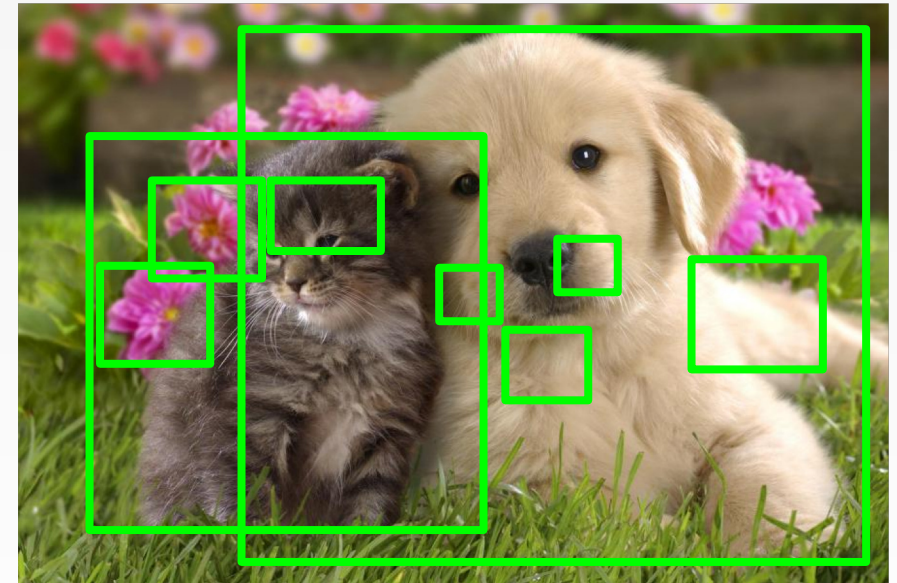


Dog? NO
Cat? NO
Background? YES

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

R-CNN



Input
image

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

R-CNN

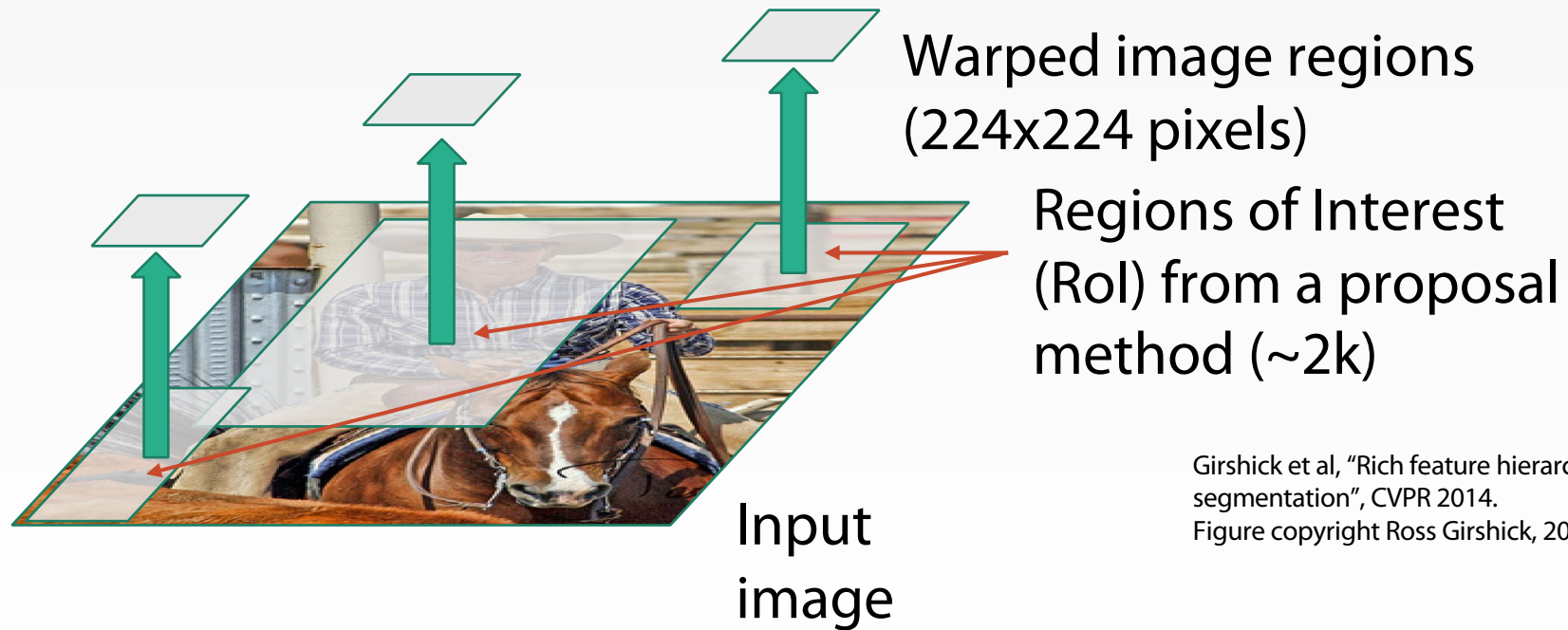


Input
image

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

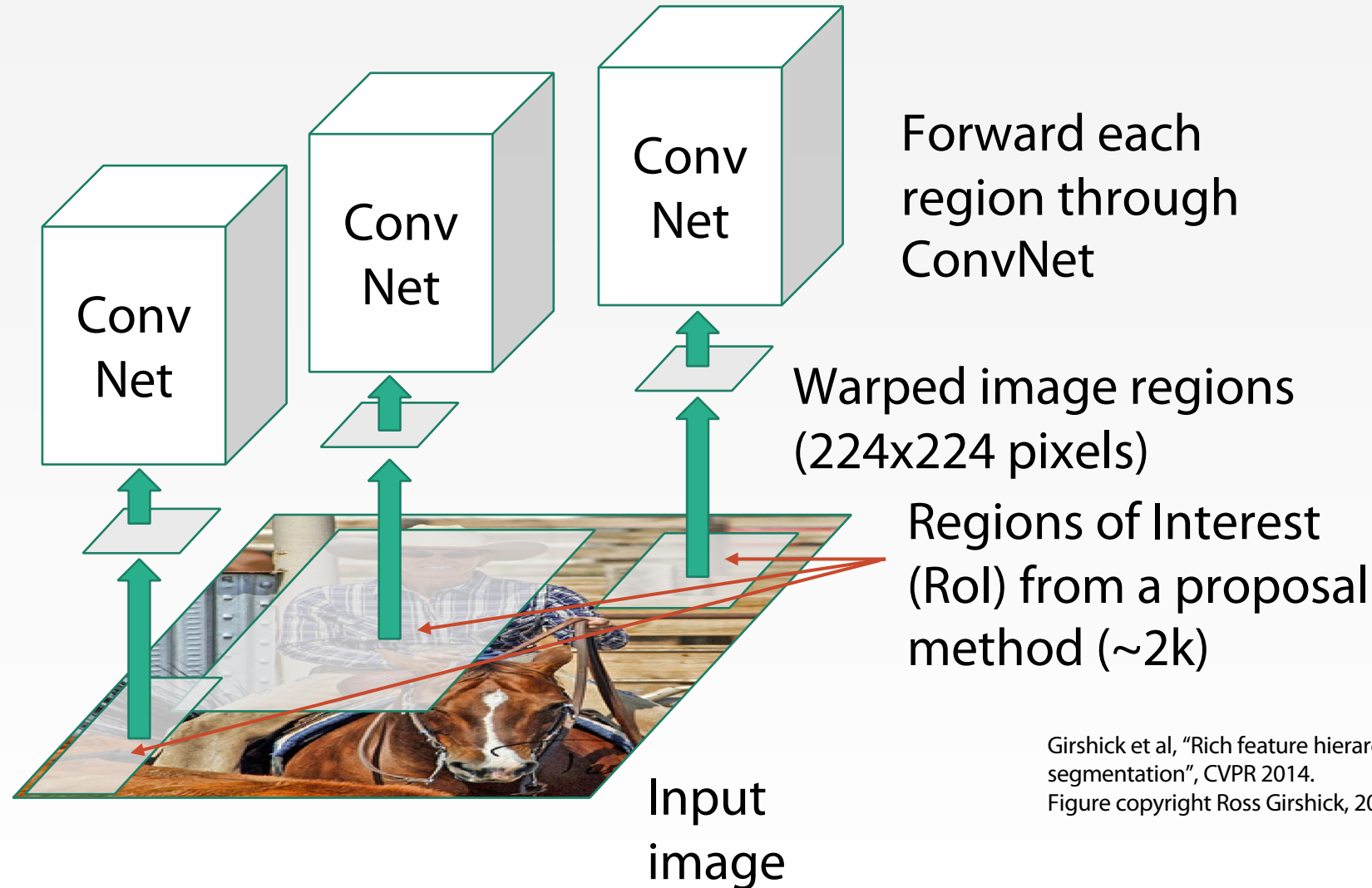
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



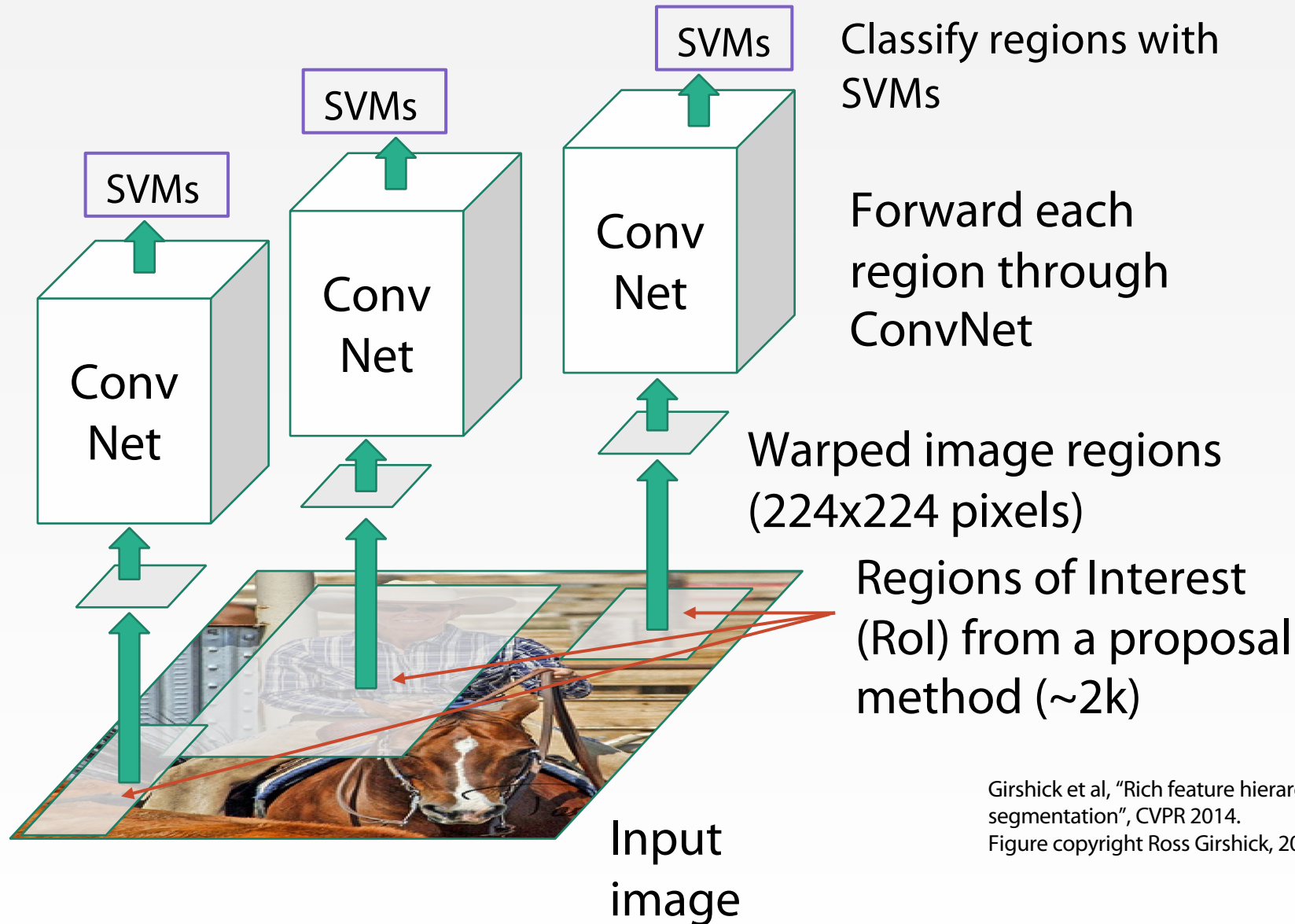
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



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

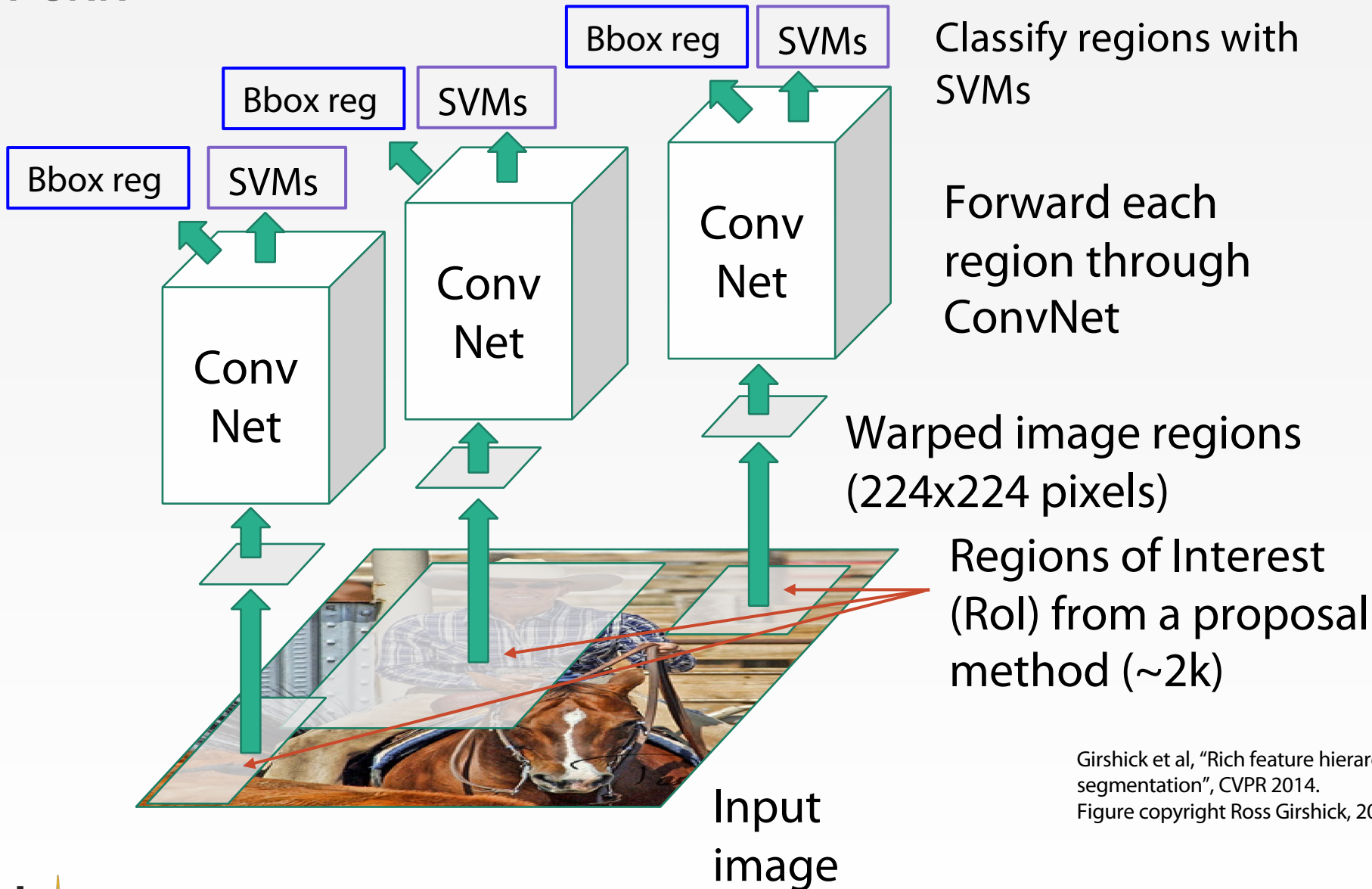
R-CNN



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

R-CNN

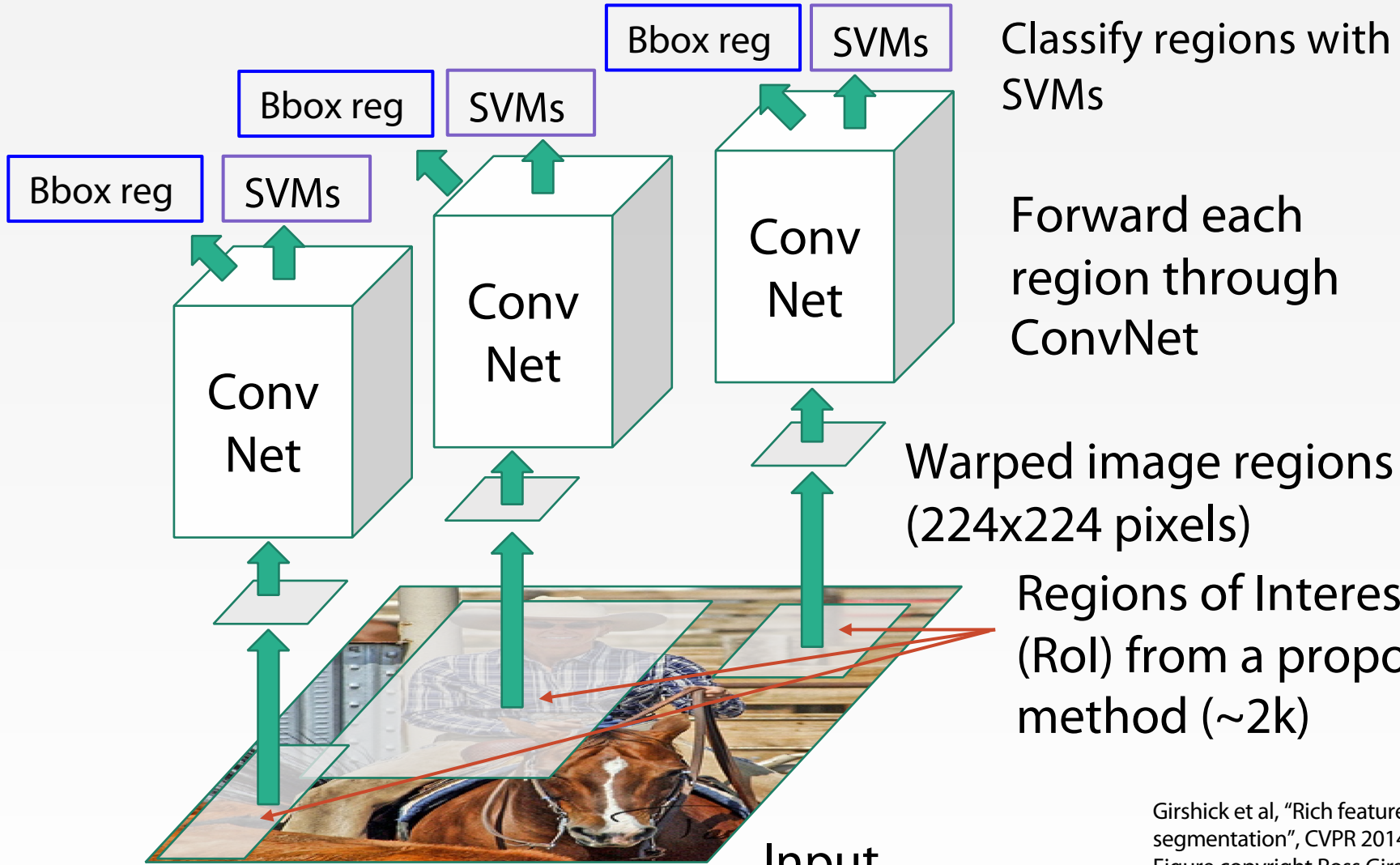
Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



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

R-CNN

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



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

Classify regions with SVMs
Forward each region through ConvNet

Warped image regions (224x224 pixels)

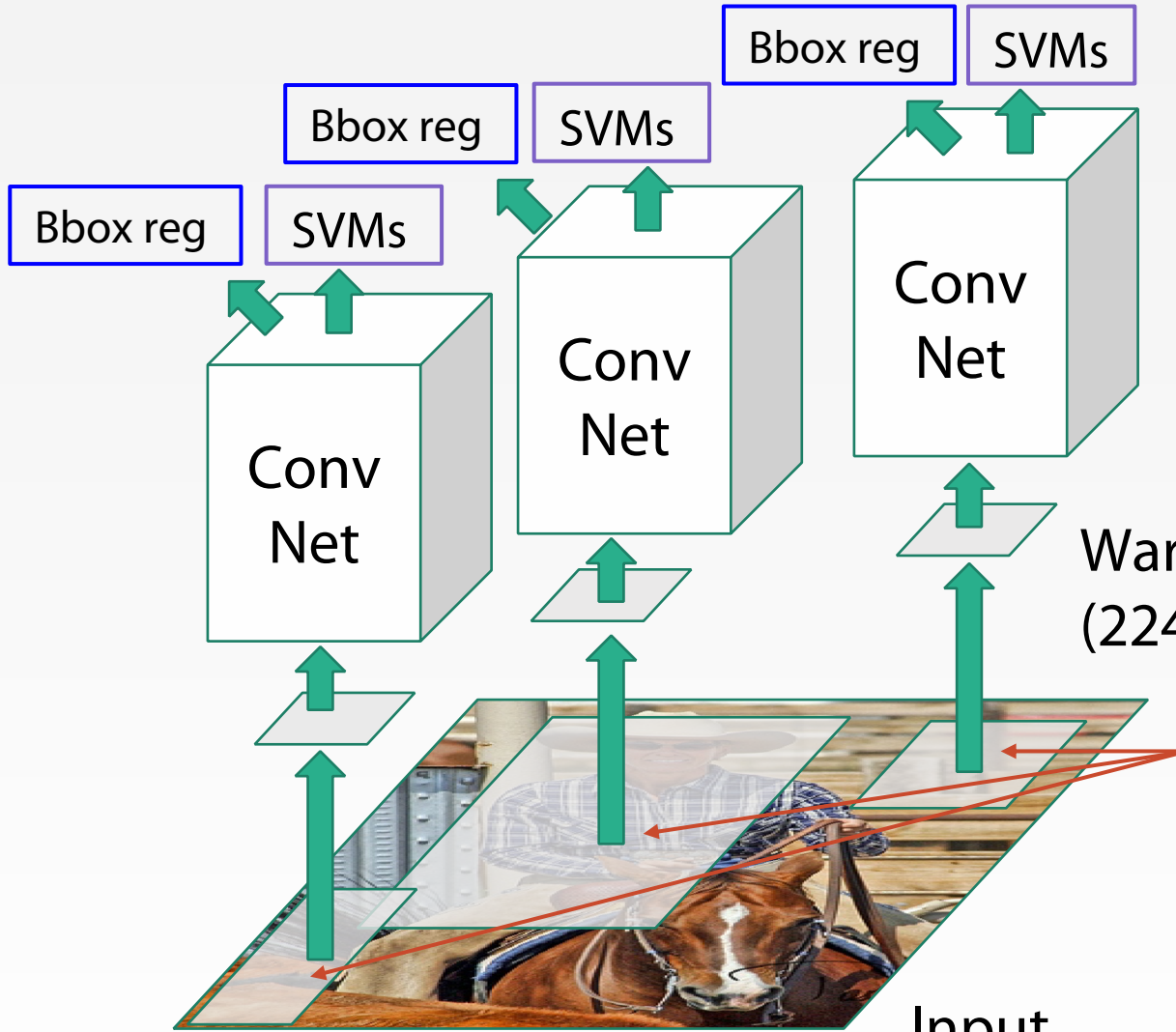
Regions of Interest (RoI) from a proposal method (~2k)

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

Input image

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)

Input image

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

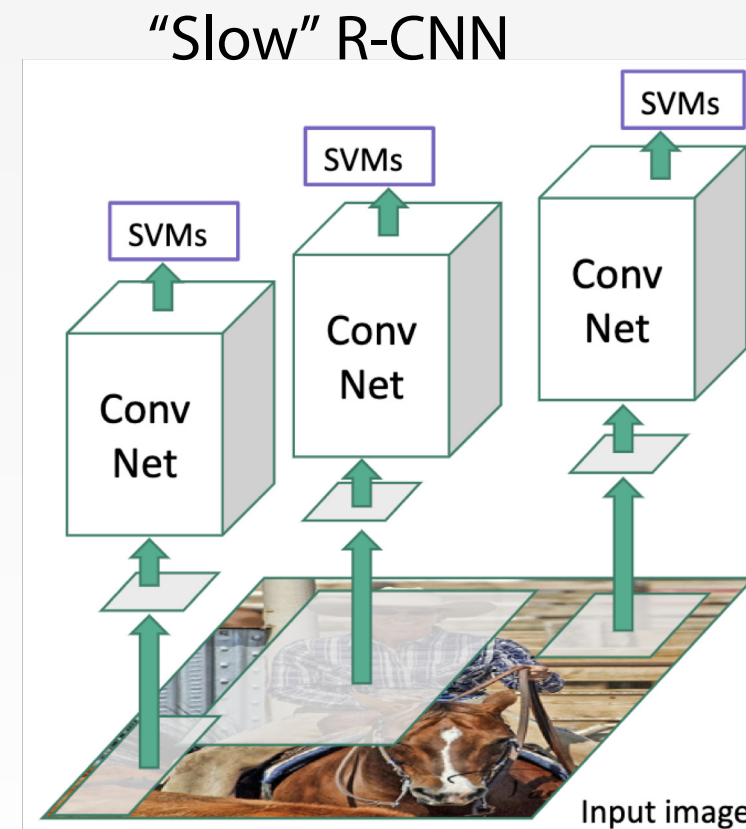


FAST R-CNN

FAST R-CNN

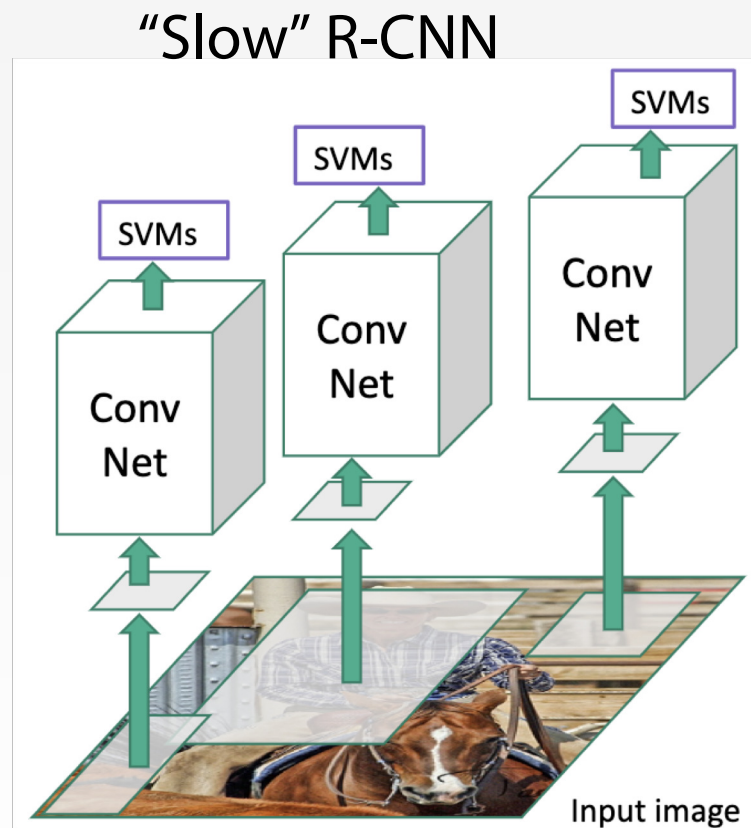
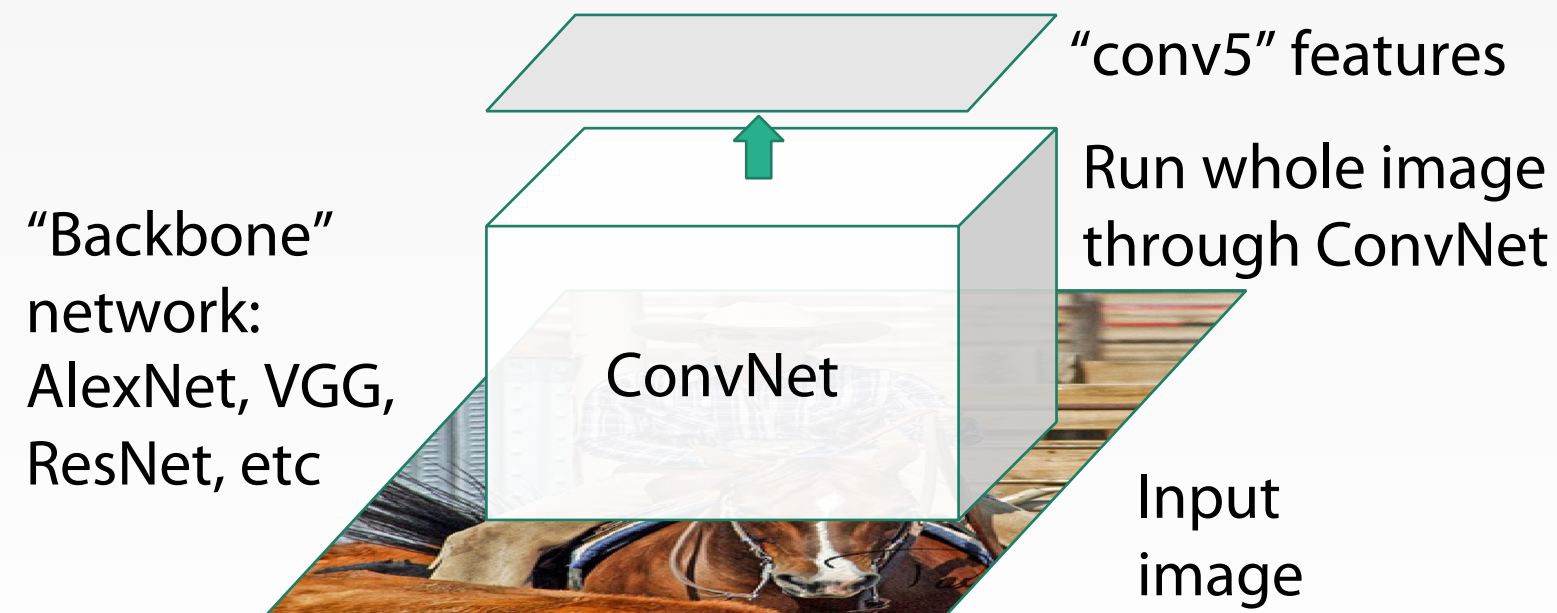


Input image



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

FAST R-CNN

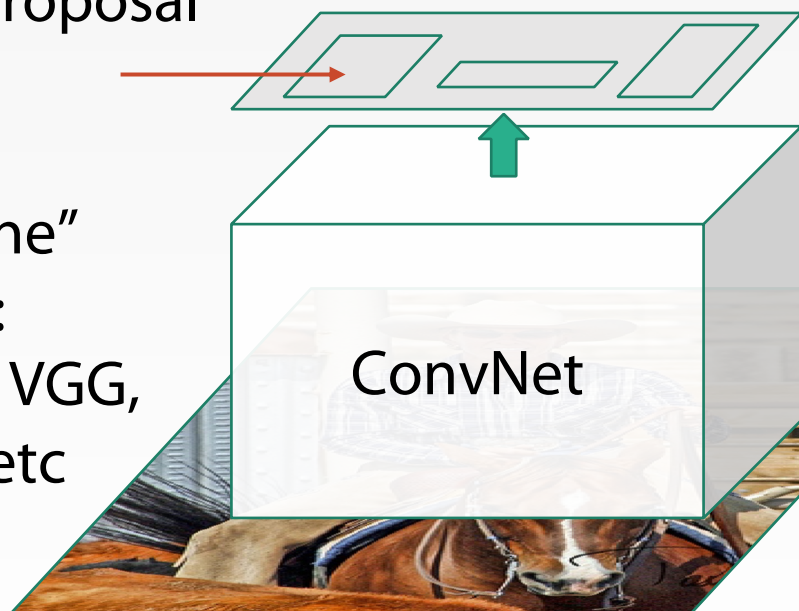


Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

FAST R-CNN

Regions of Interest (Rois) from a proposal method

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

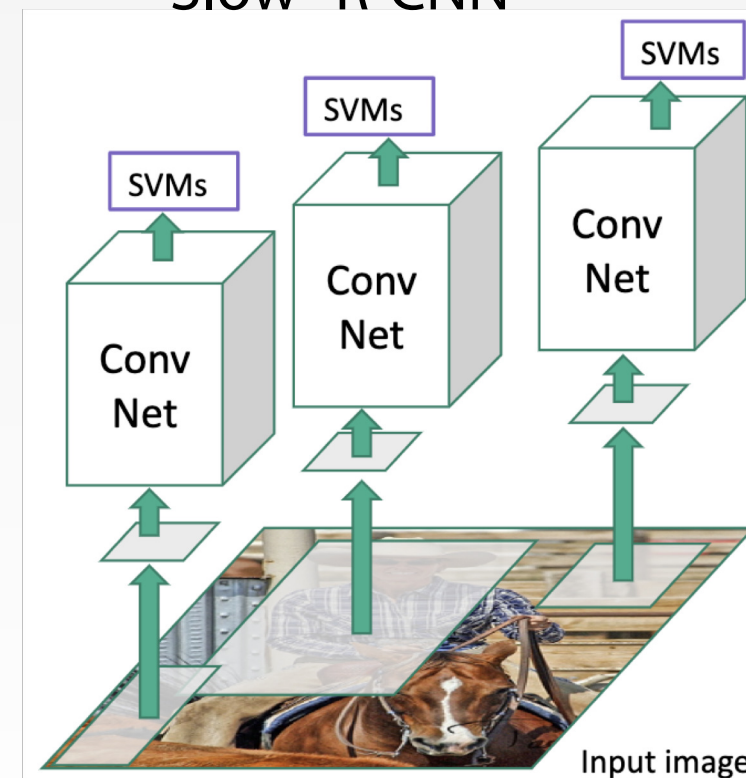


“conv5” features

Run whole image through ConvNet

Input image

“Slow” R-CNN

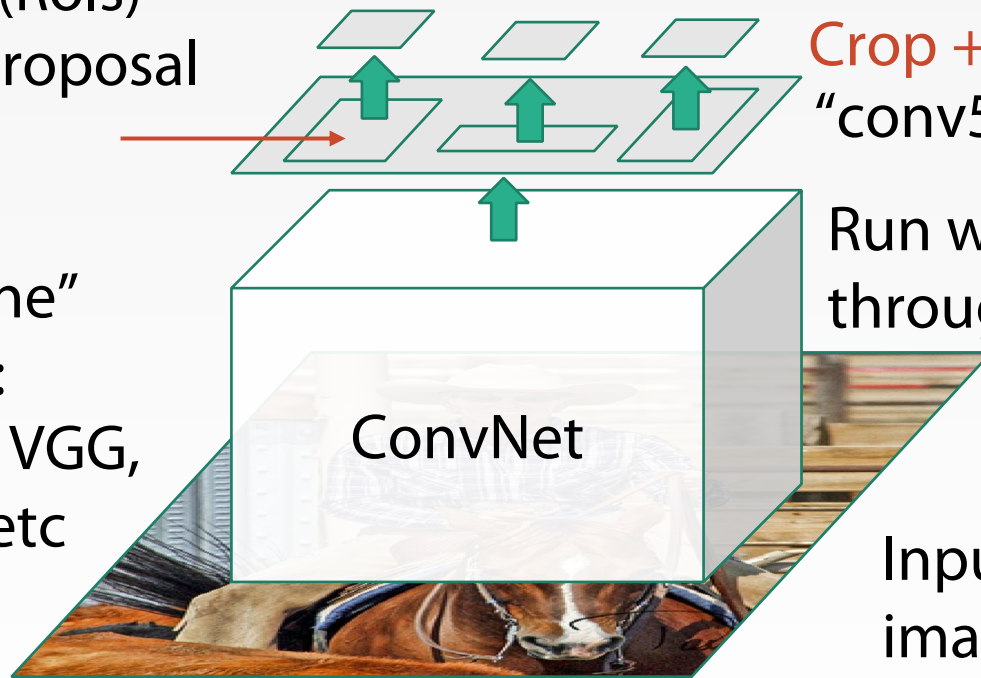


Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

FAST R-CNN

Regions of Interest (Rois) from a proposal method

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

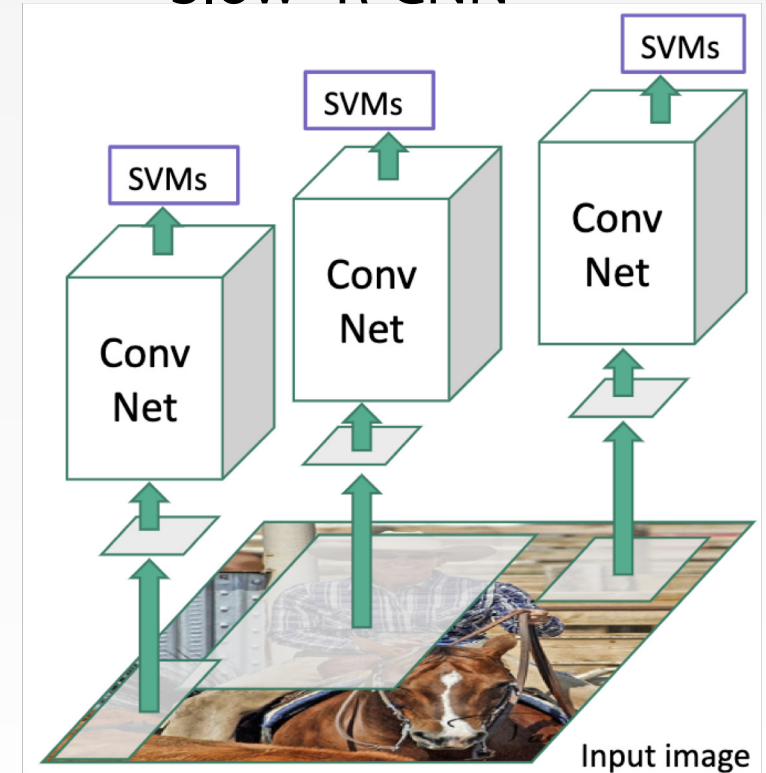


Crop + Resize features
“conv5” features

Run whole image
through ConvNet

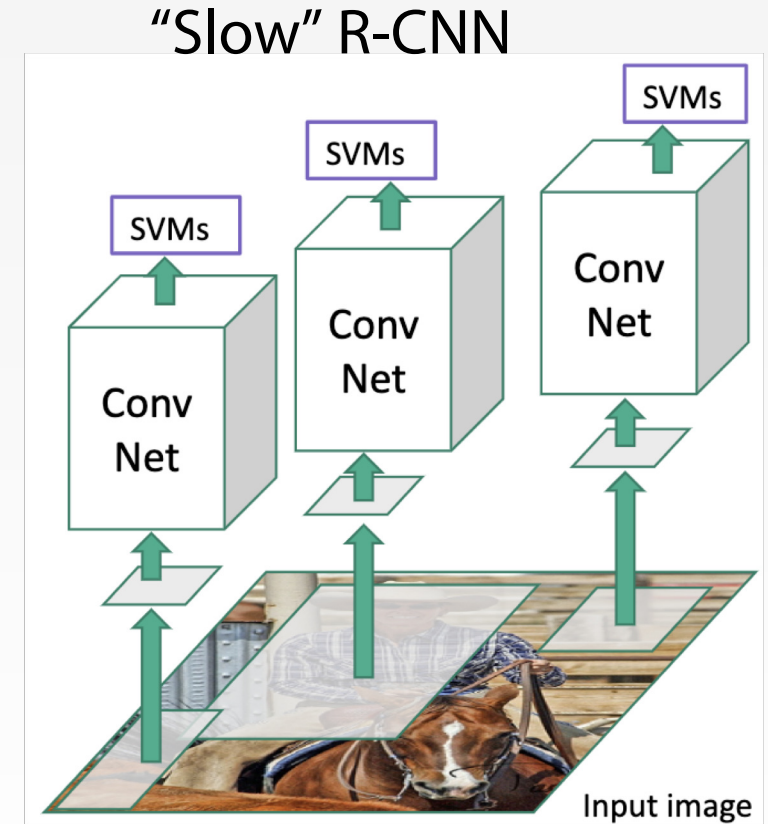
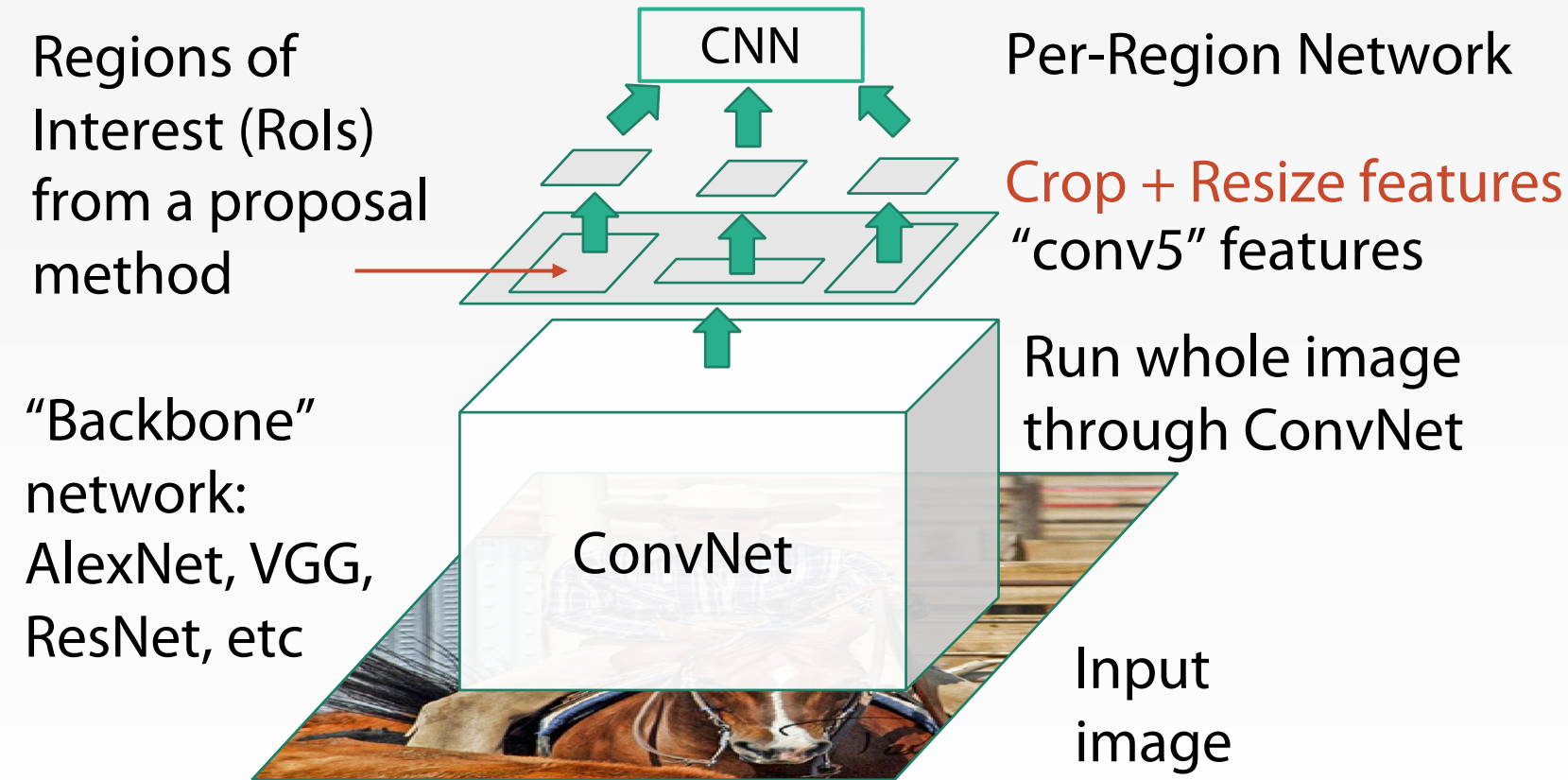
Input
image

“Slow” R-CNN



Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

FAST R-CNN



Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

FAST R-CNN

Object category

Linear + softmax

Linear

Box offset

CNN

Per-Region Network

Regions of Interest (Rois) from a proposal method

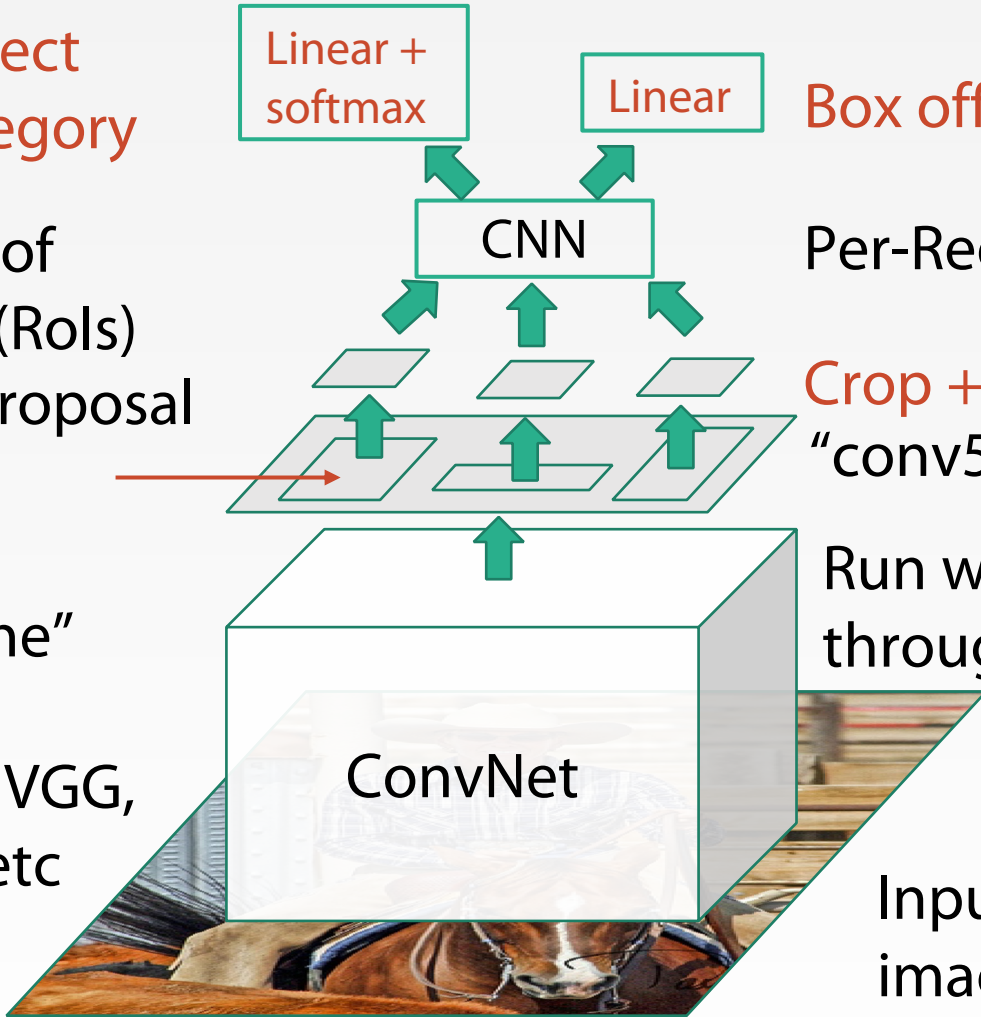
Crop + Resize features "conv5" features

Run whole image through ConvNet

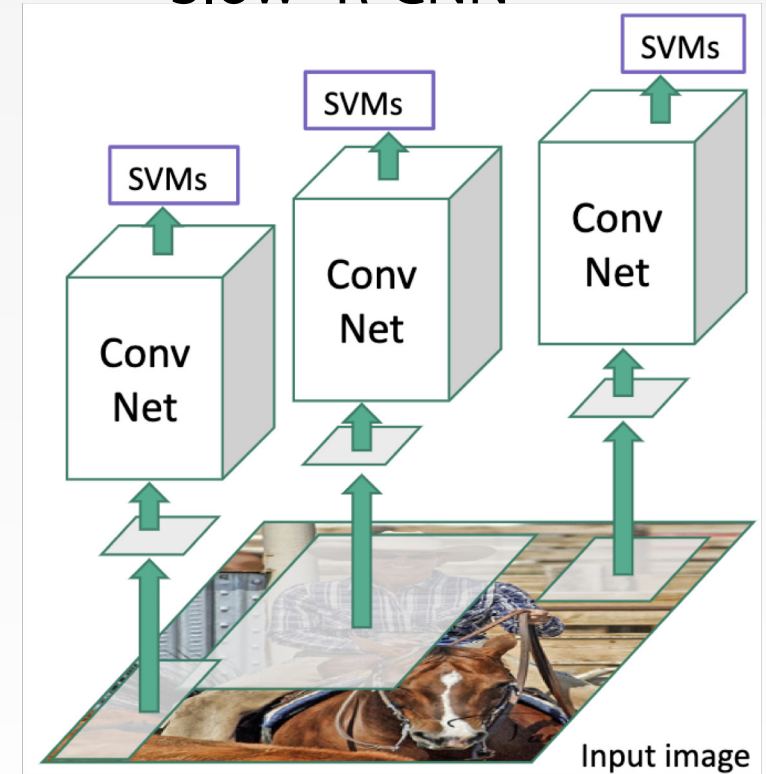
"Backbone" network: AlexNet, VGG, ResNet, etc

ConvNet

Input image

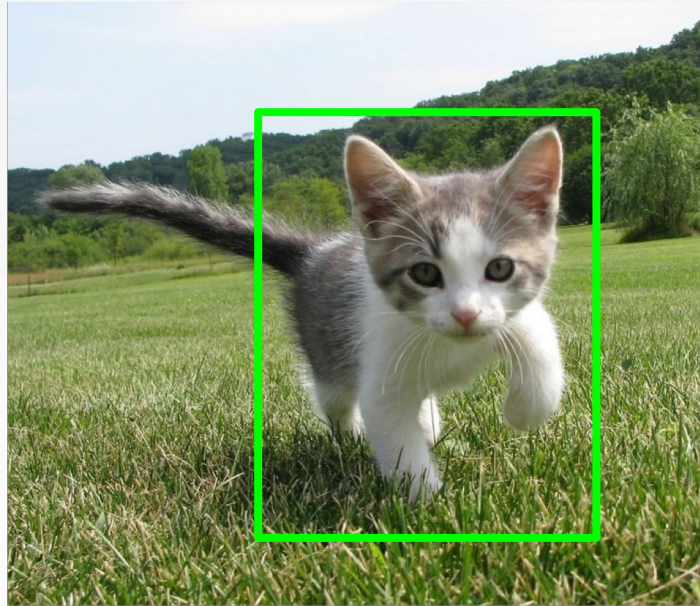


"Slow" R-CNN



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)

CNN

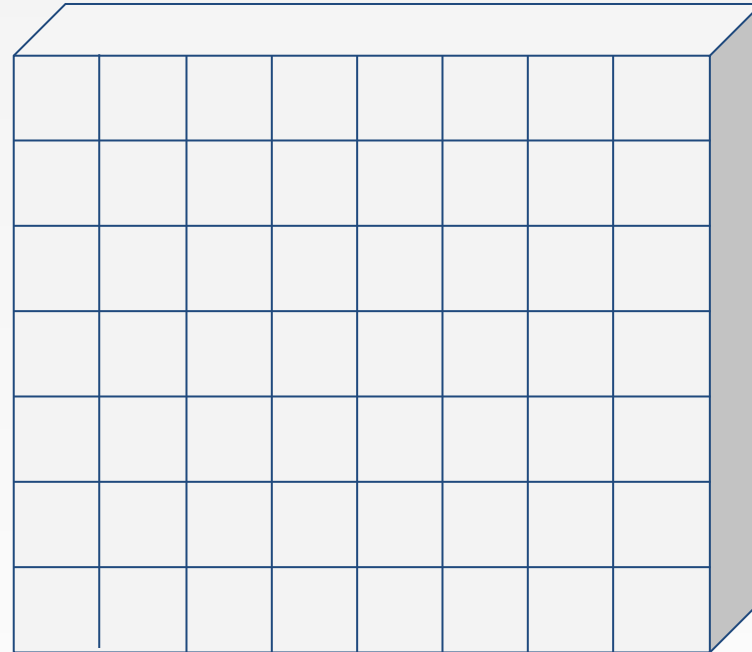
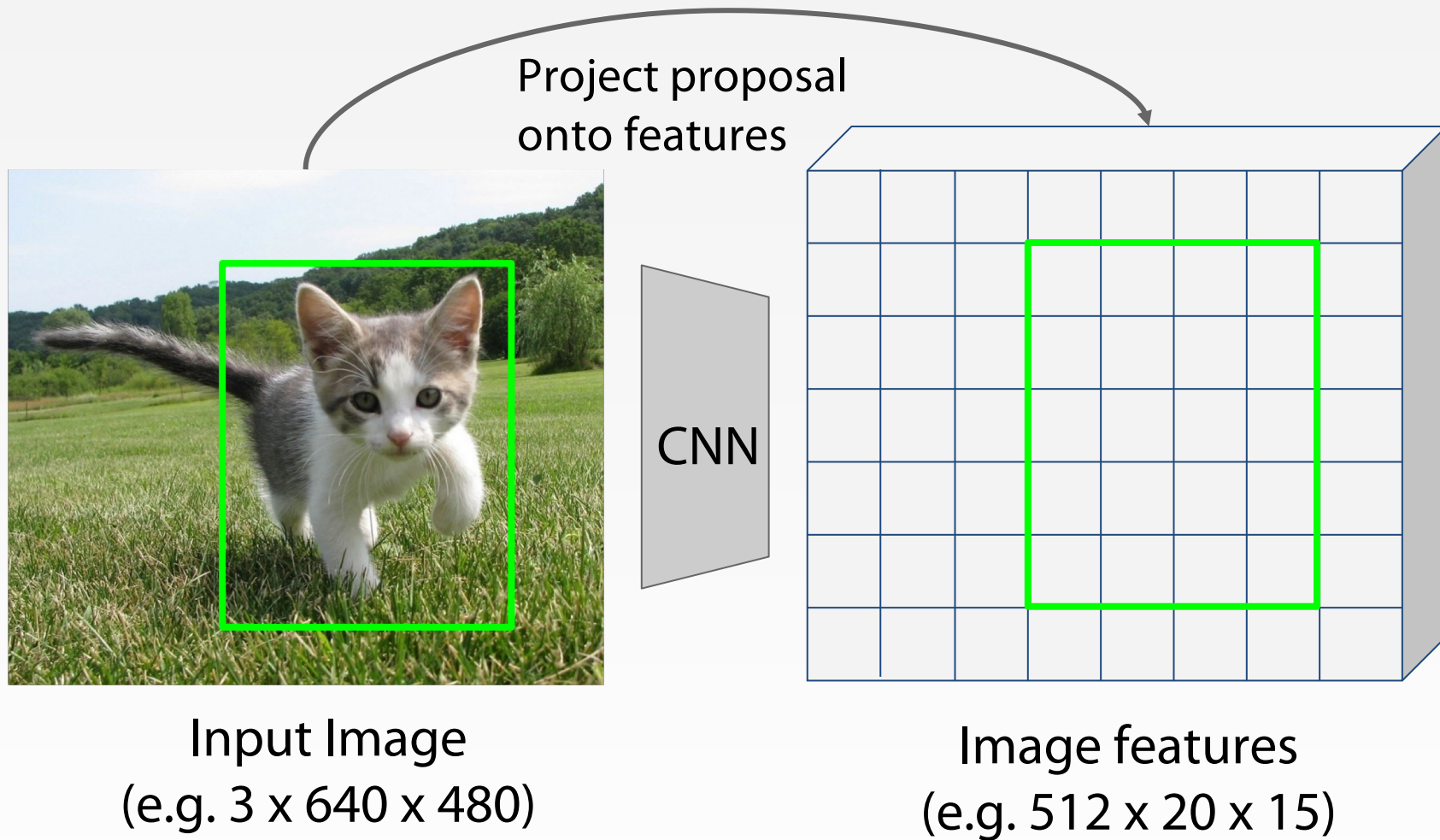


Image features
(e.g. 512 x 20 x 15)

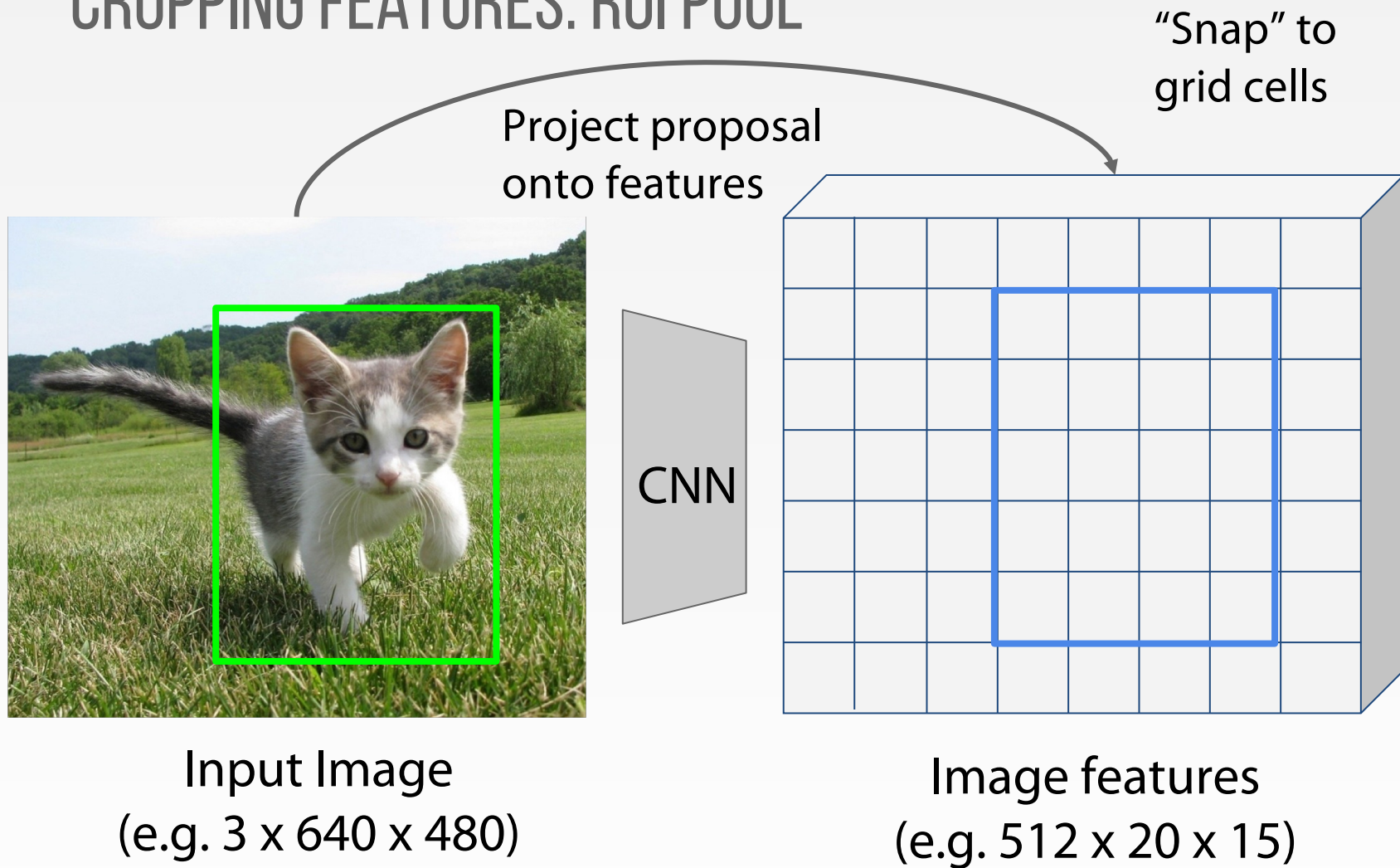
Girshick, "Fast R-CNN", ICCV 2015.

CROPPING FEATURES: ROI POOL



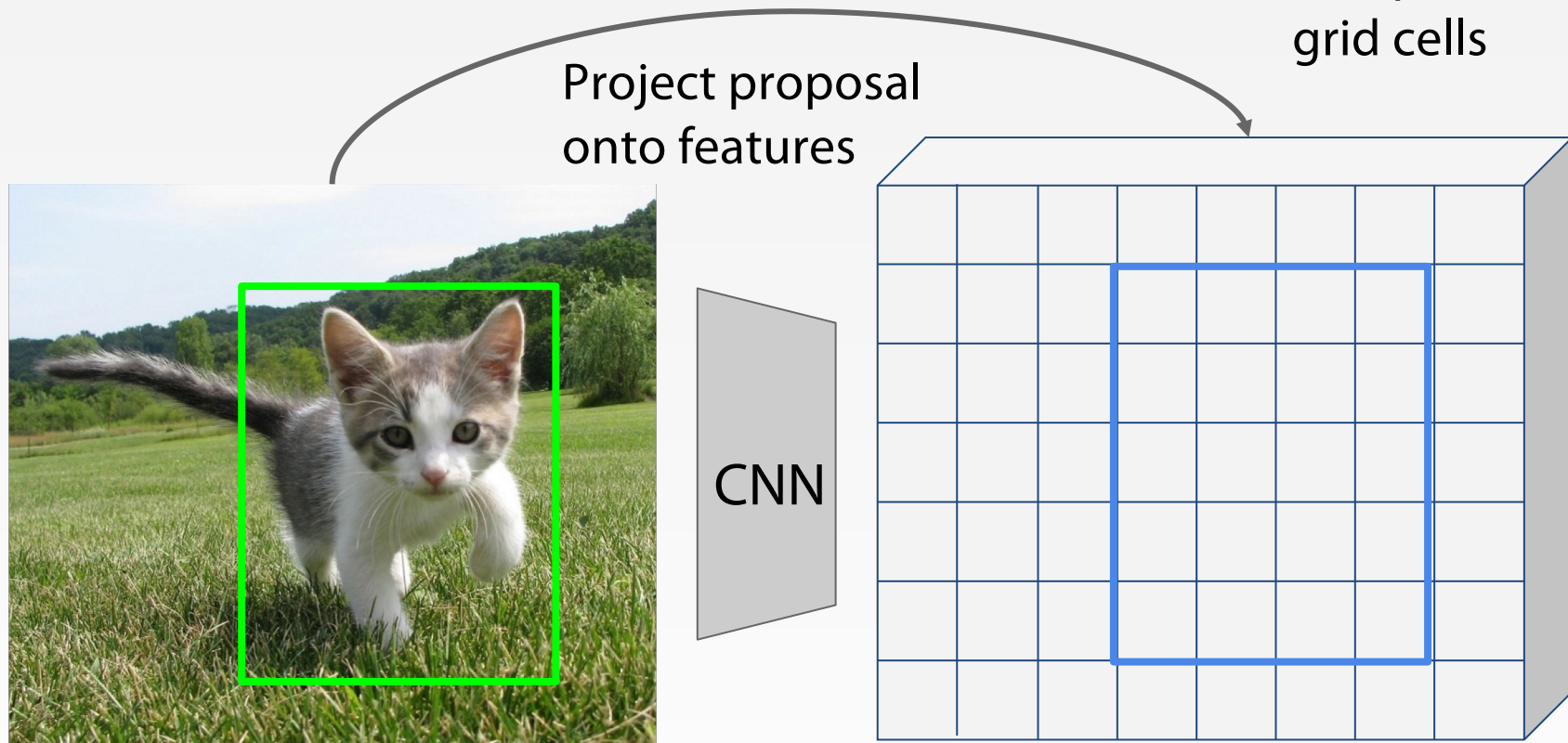
Girshick, "Fast R-CNN", ICCV 2015.

CROPPING FEATURES: ROI POOL



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

CROPPING FEATURES: ROI POOL



Divide into 2x2 grid of (roughly) equal subregions

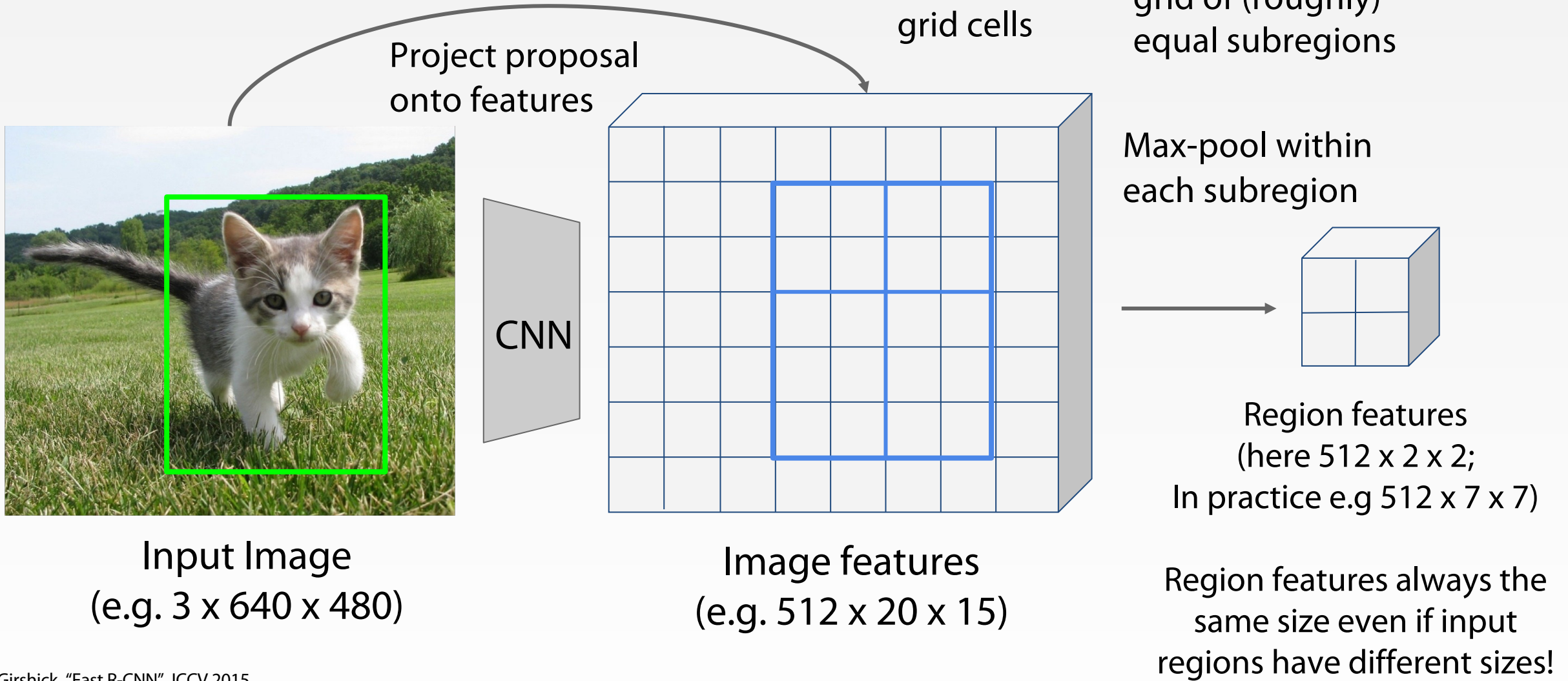
Input Image
(e.g. 3 x 640 x 480)

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

CNN

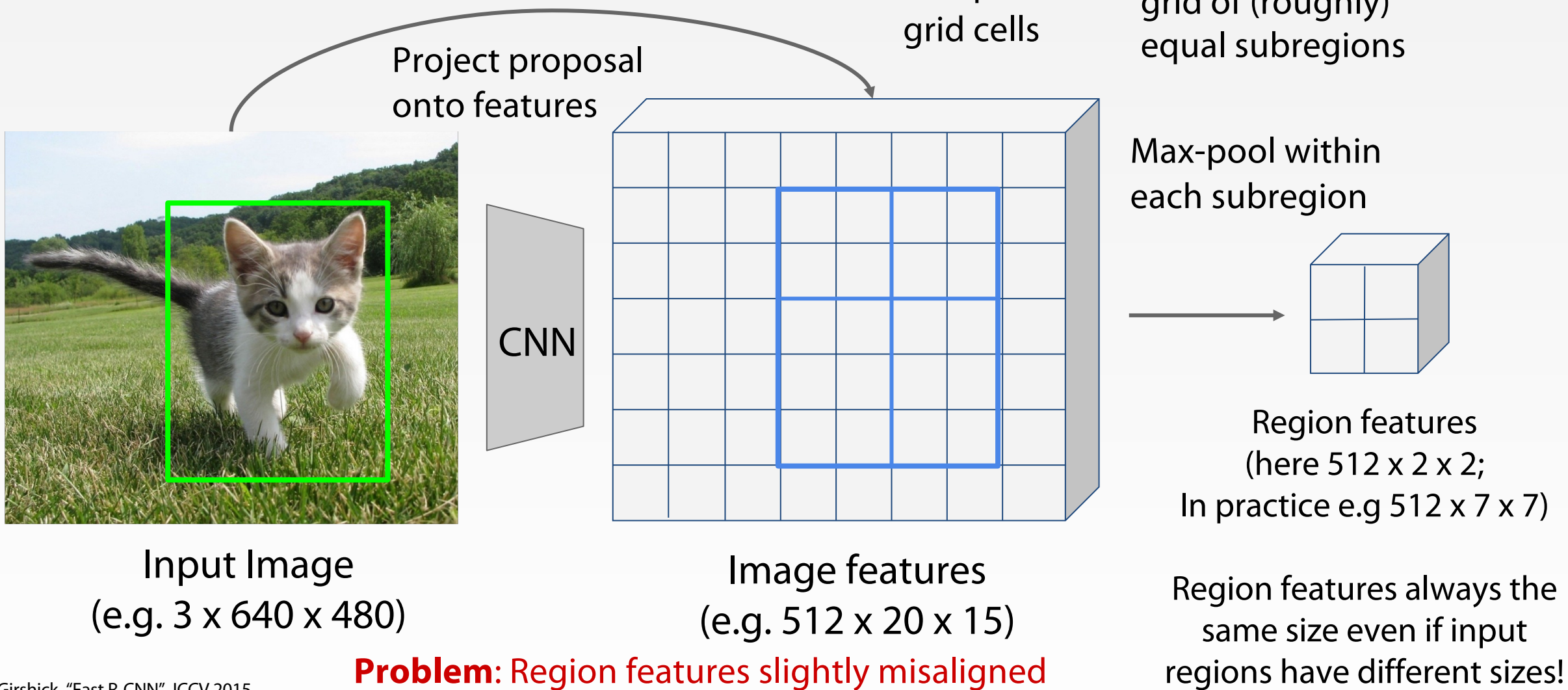
Girshick, "Fast R-CNN", ICCV 2015.

CROPPING FEATURES: ROI POOL



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

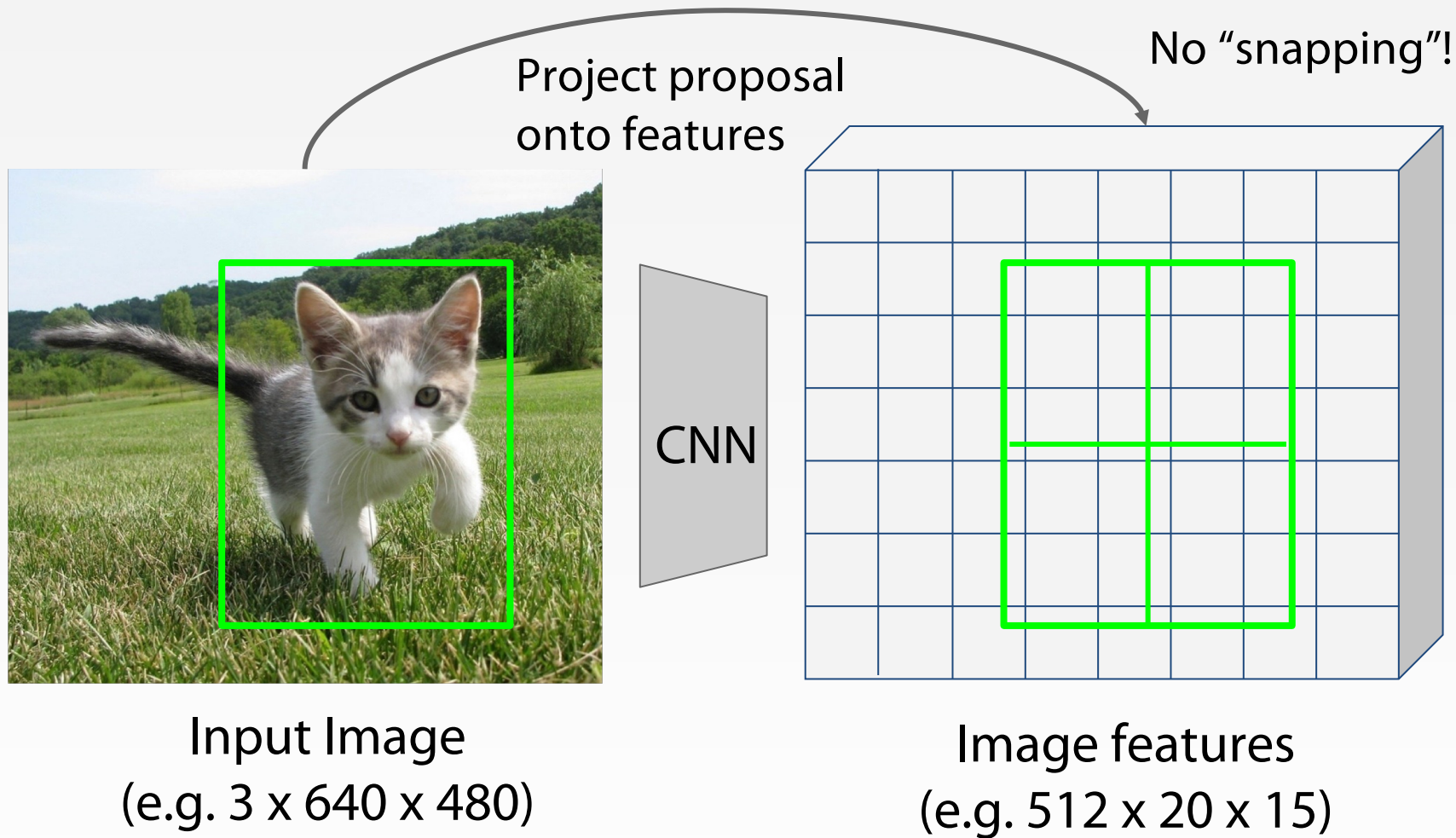
CROPPING FEATURES: ROI POOL



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

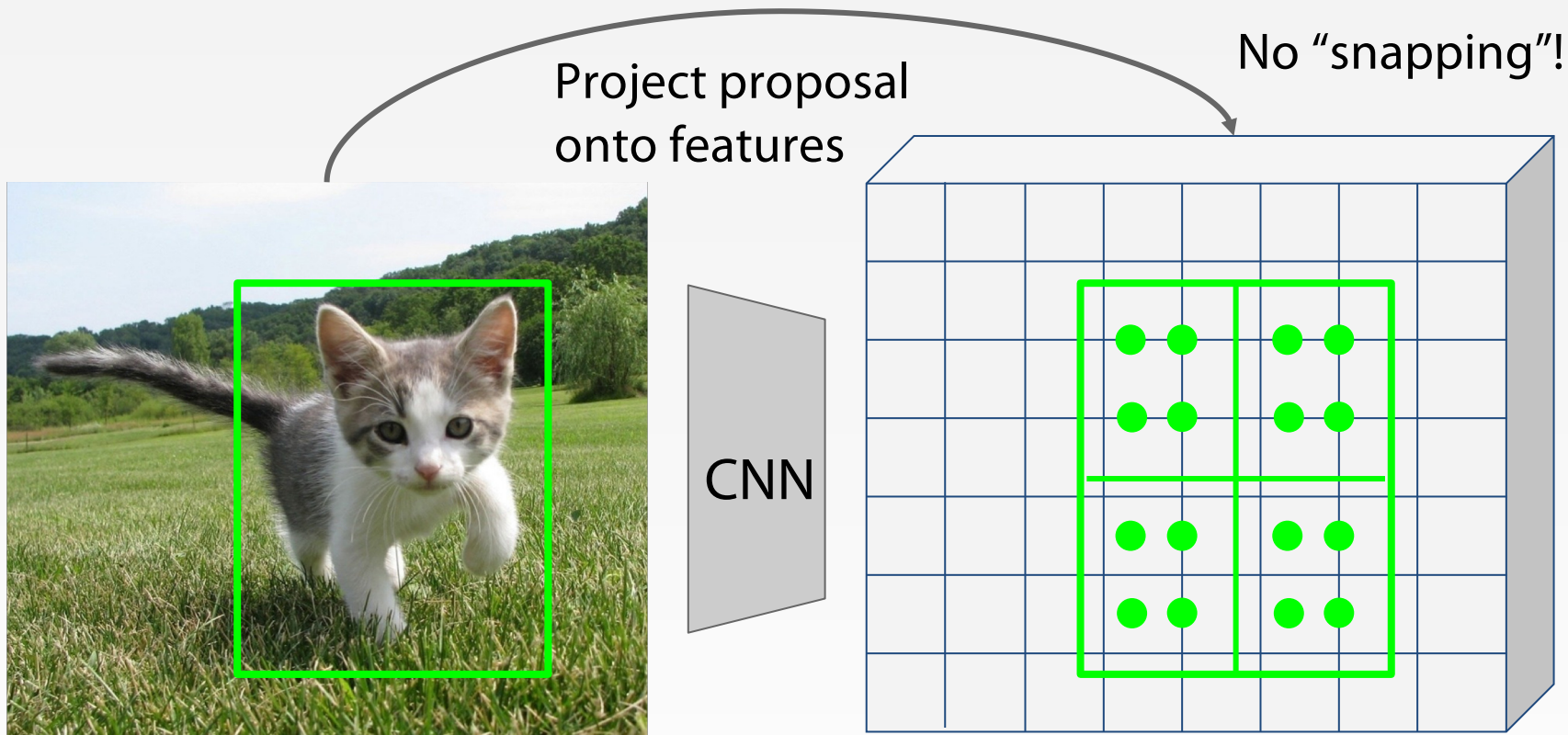
Problem: Region features slightly misaligned

CROPPING FEATURES: ROI ALIGN



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

CROPPING FEATURES: ROI ALIGN

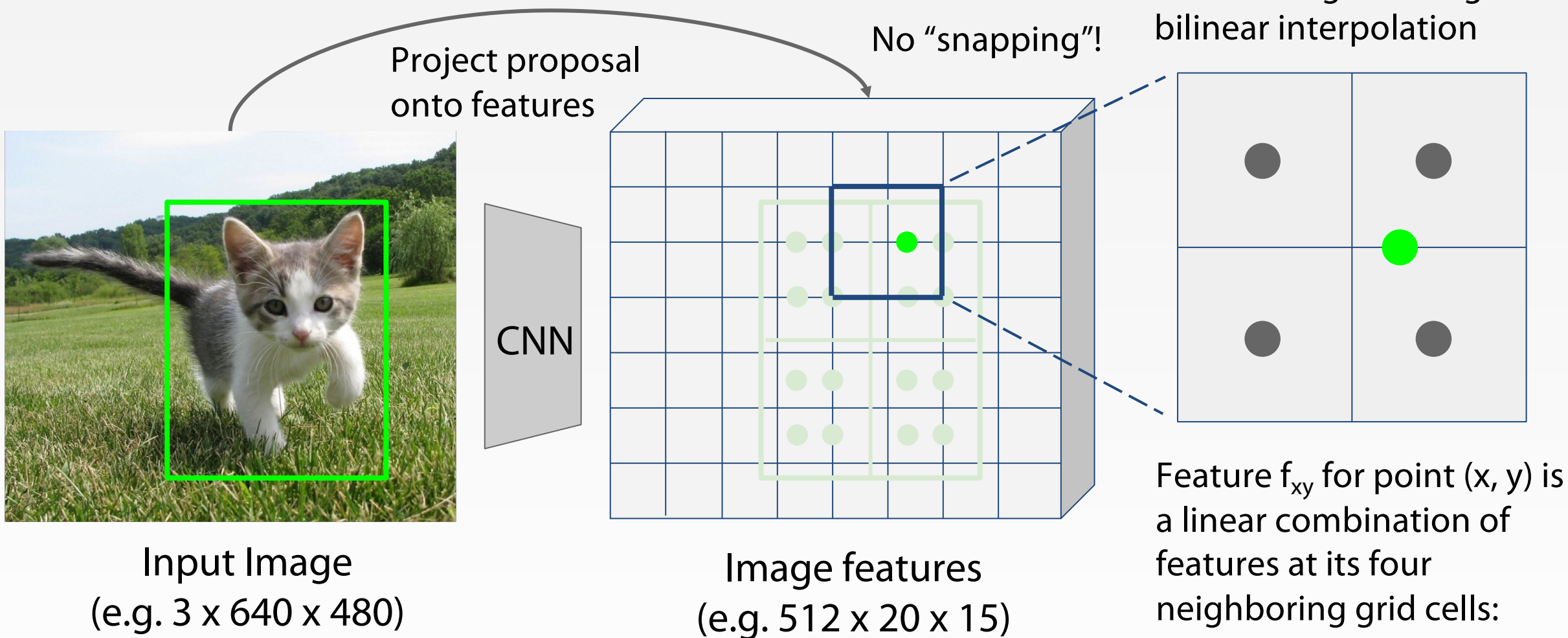


Input Image
(e.g. 3 x 640 x 480)

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

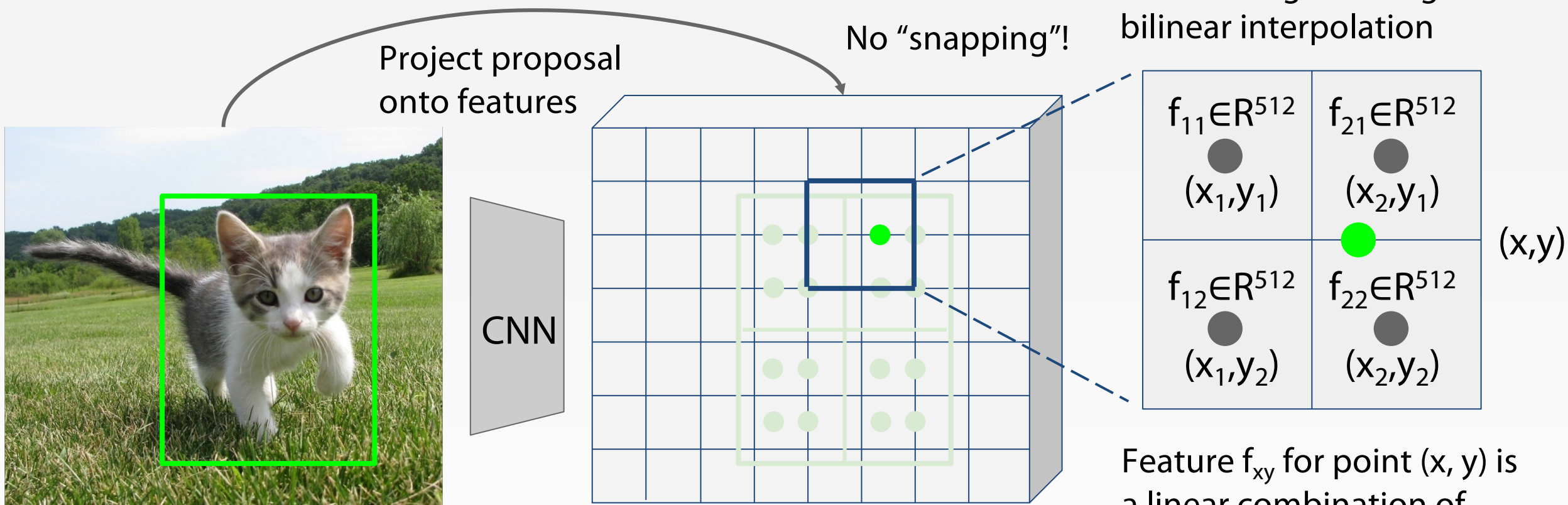
Sample at regular points in each subregion using bilinear interpolation

CROPPING FEATURES: ROI ALIGN



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

CROPPING FEATURES: ROI ALIGN



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

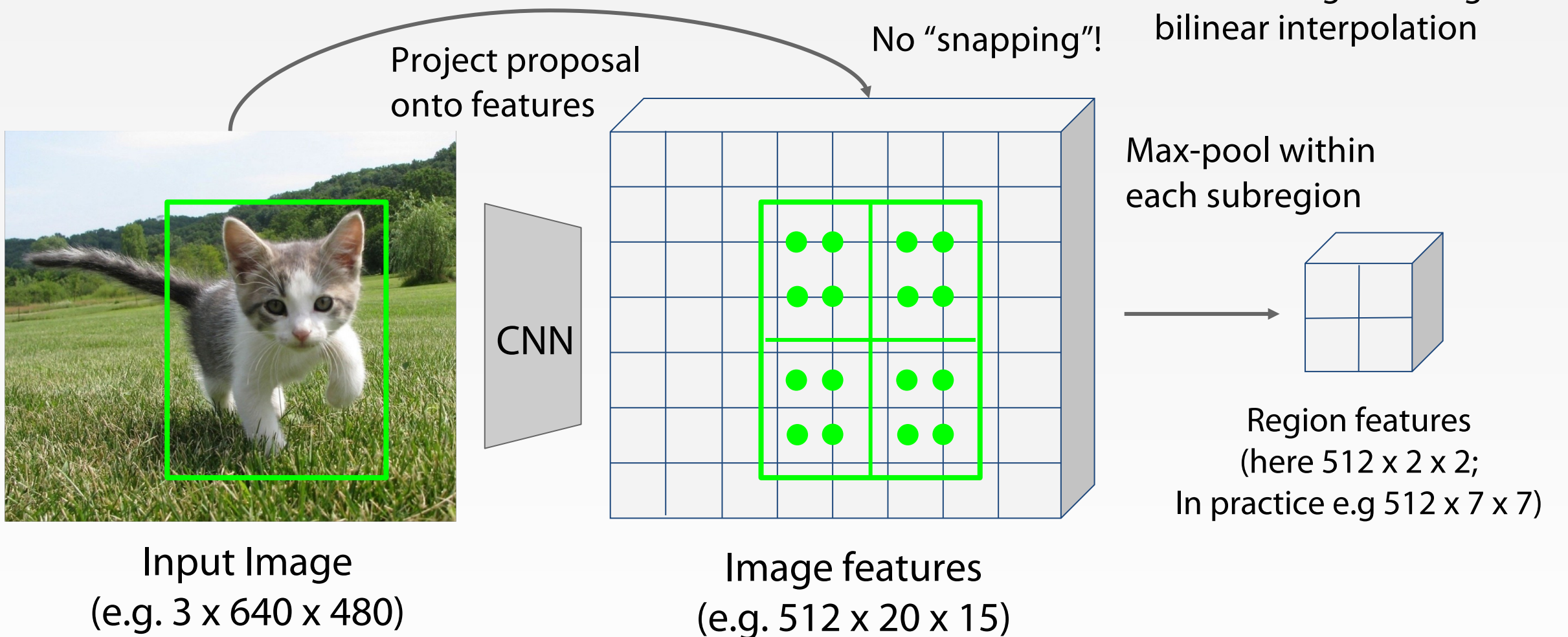
Image features
(e.g. 512 x 20 x 15)

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

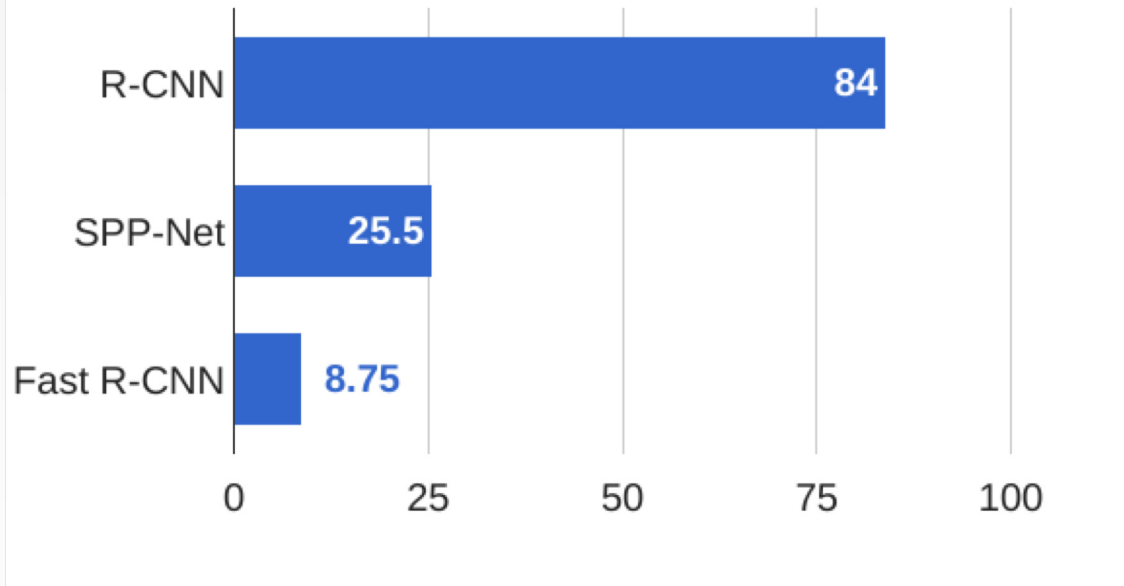
CROPPING FEATURES: ROI ALIGN



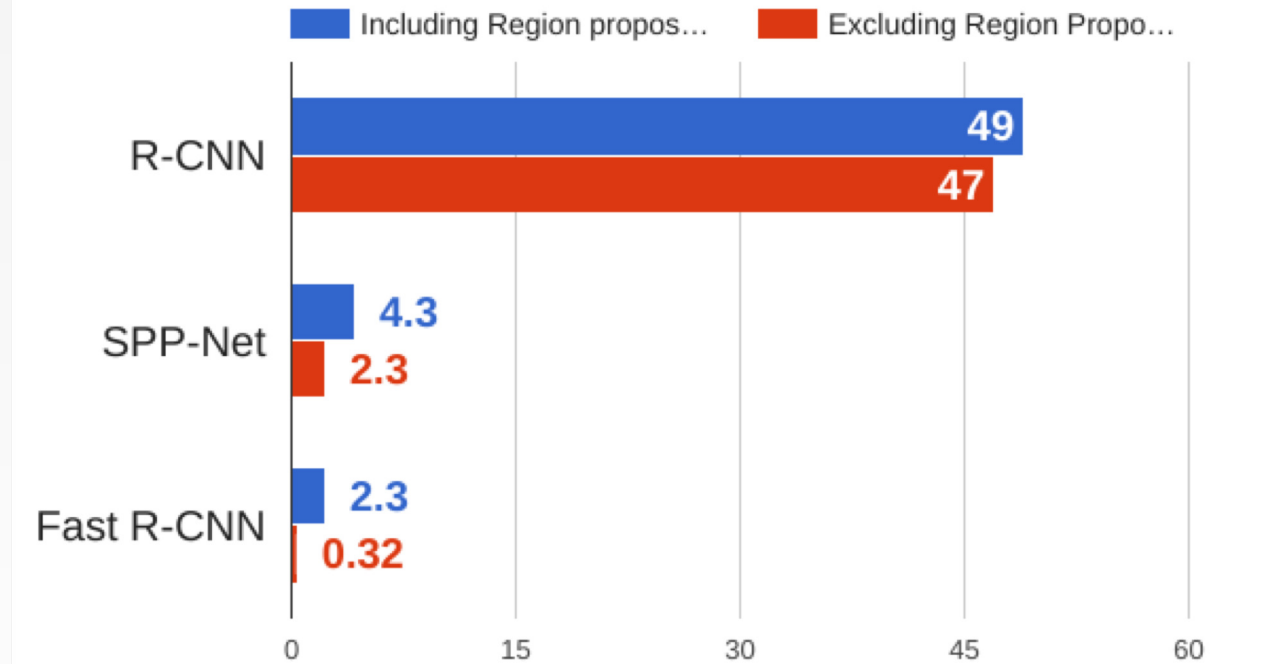
He et al, "Mask R-CNN", ICCV 2017

R-CNN VS FAST R-CNN

Training time (Hours)



Test time (seconds)



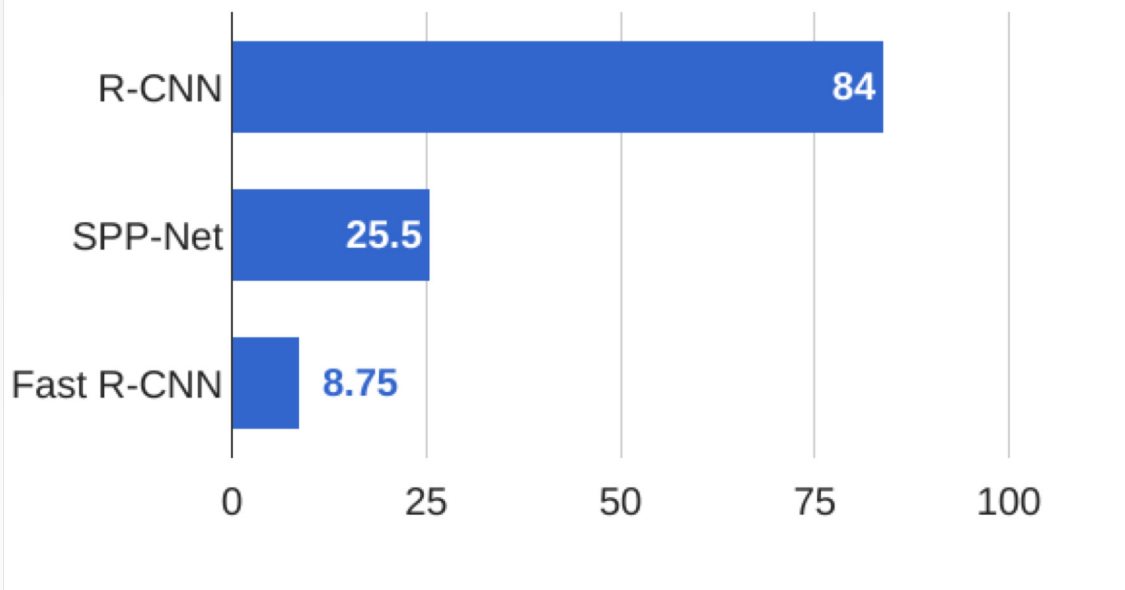
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

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

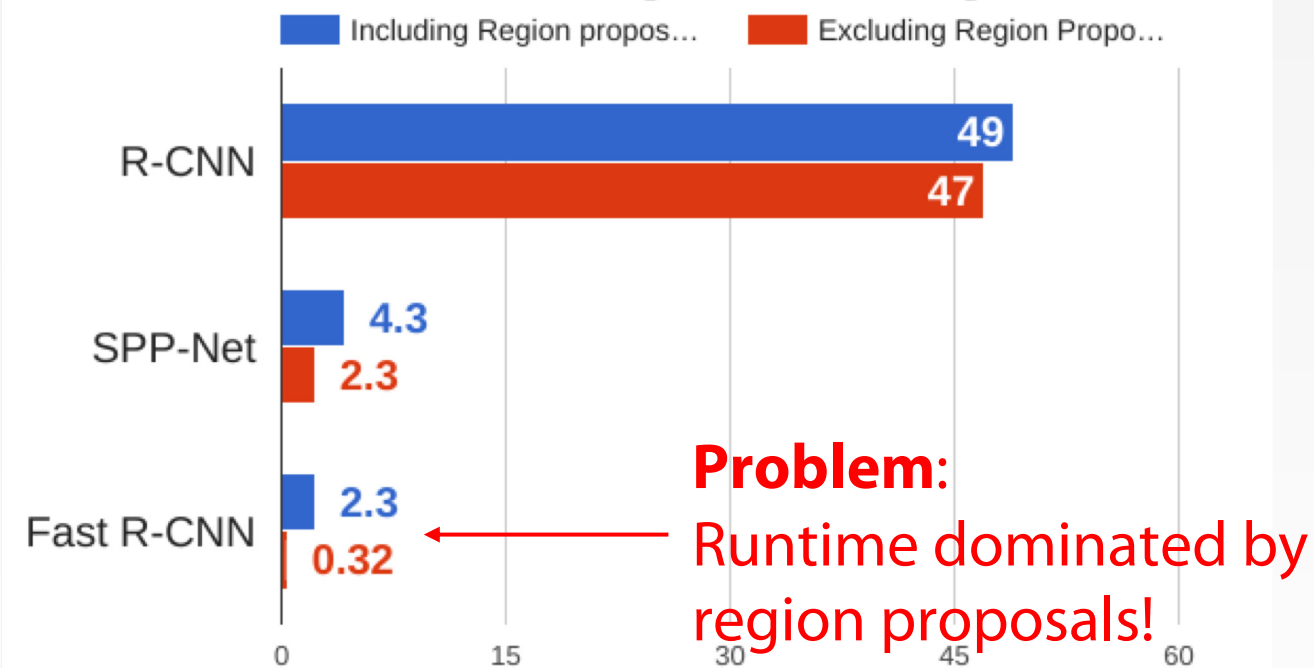
Girshick, "Fast R-CNN", ICCV 2015

R-CNN VS FAST R-CNN

Training time (Hours)



Test time (seconds)



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

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

Girshick, "Fast R-CNN", ICCV 2015

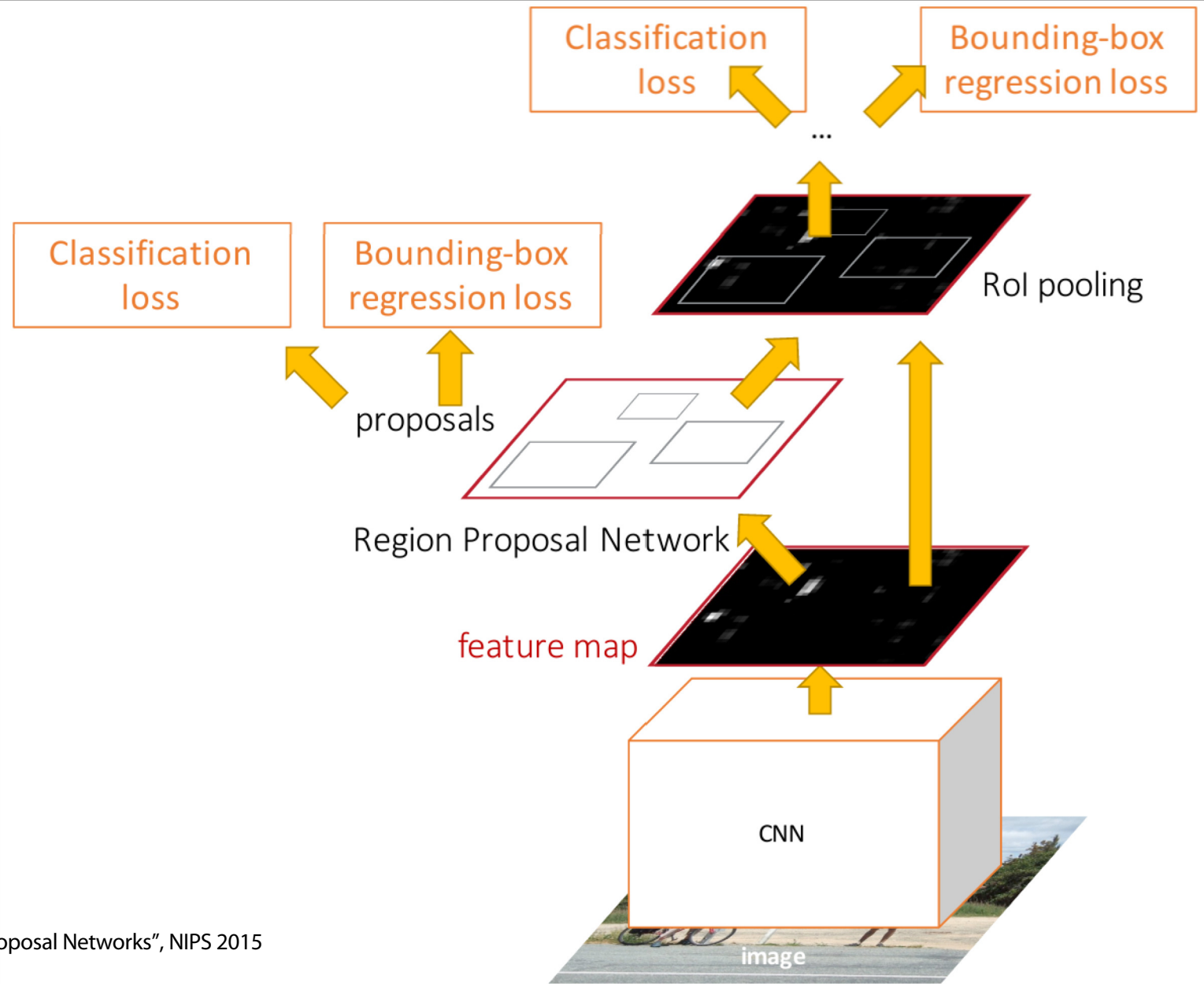


FASTER R-CNN

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 x 640 x 480)

CNN

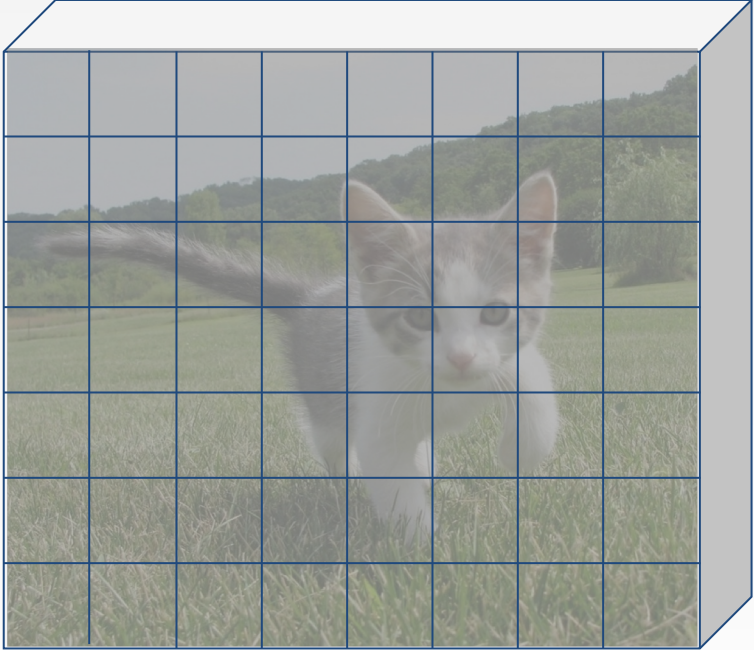


Image features
(e.g. 512 x 20 x 15)

REGION PROPOSAL NETWORK

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



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

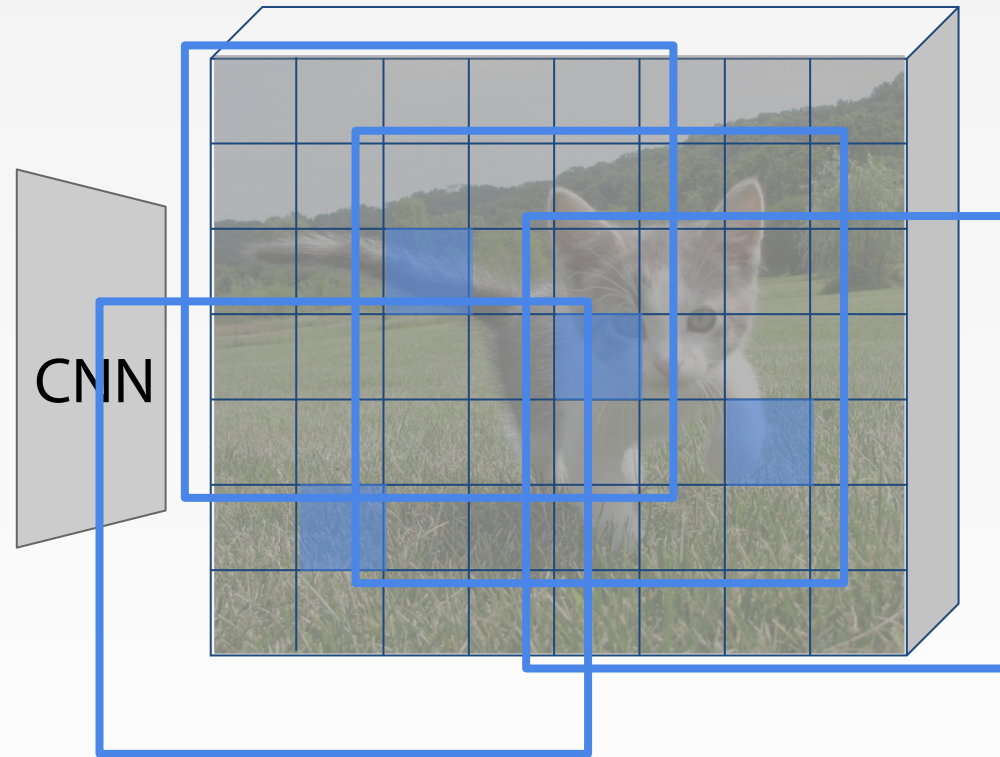


Image features
(e.g. 512 x 20 x 15)

REGION PROPOSAL NETWORK



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

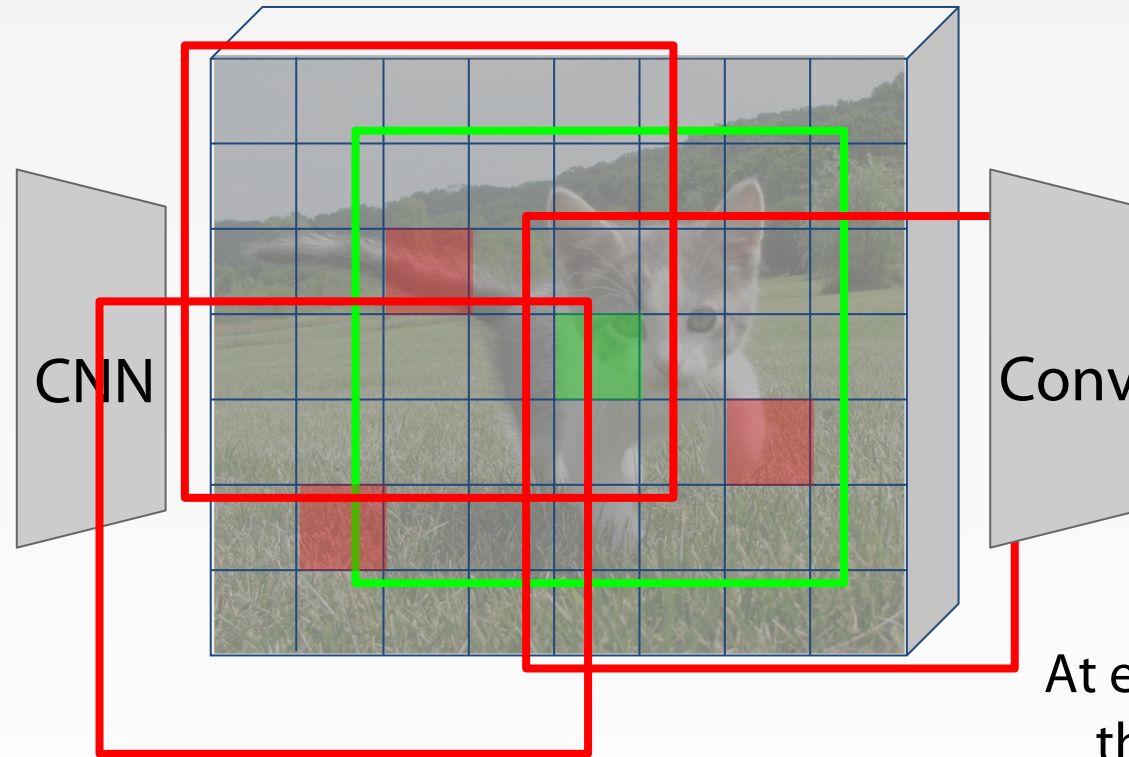


Image features
(e.g. 512 x 20 x 15)

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

Anchor is an object?
1 x 20 x 15

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

REGION PROPOSAL NETWORK



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

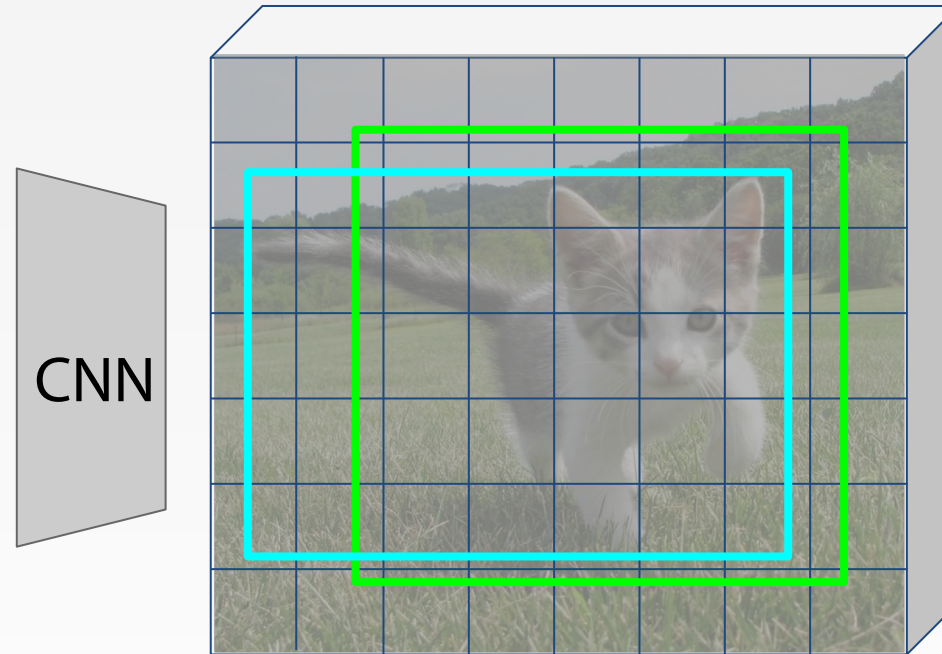


Image features
(e.g. 512 x 20 x 15)

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



Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

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

REGION PROPOSAL NETWORK

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



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

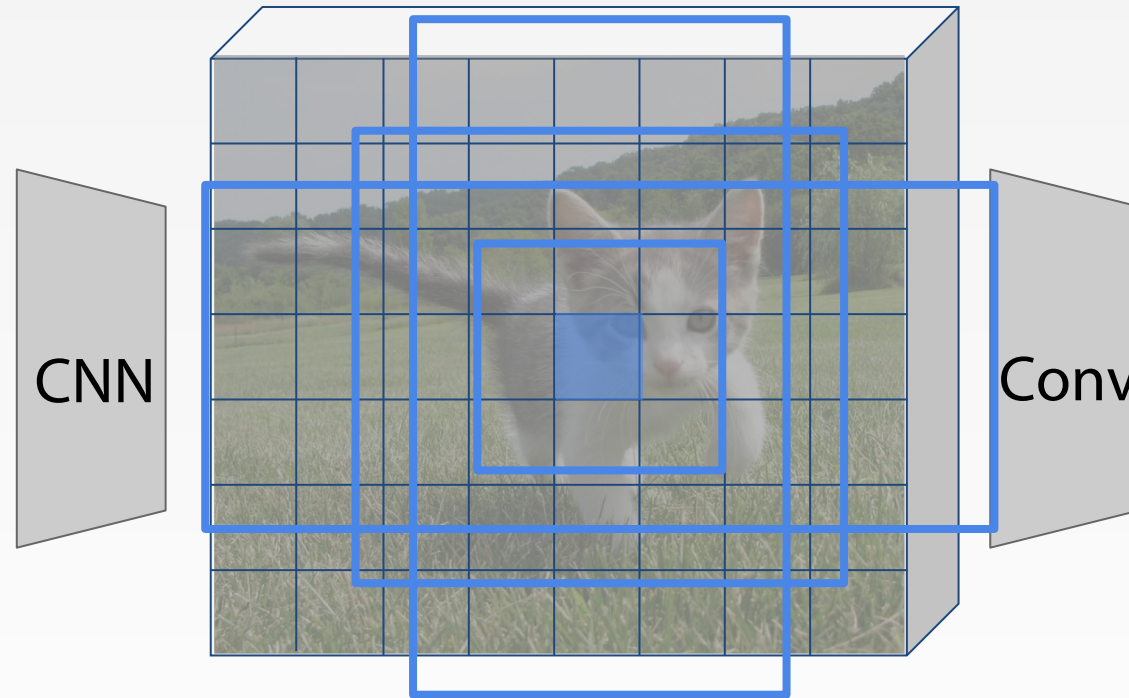


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?

1K x 20 x 15

Box transforms

4K x 20 x 15

REGION PROPOSAL NETWORK

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



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

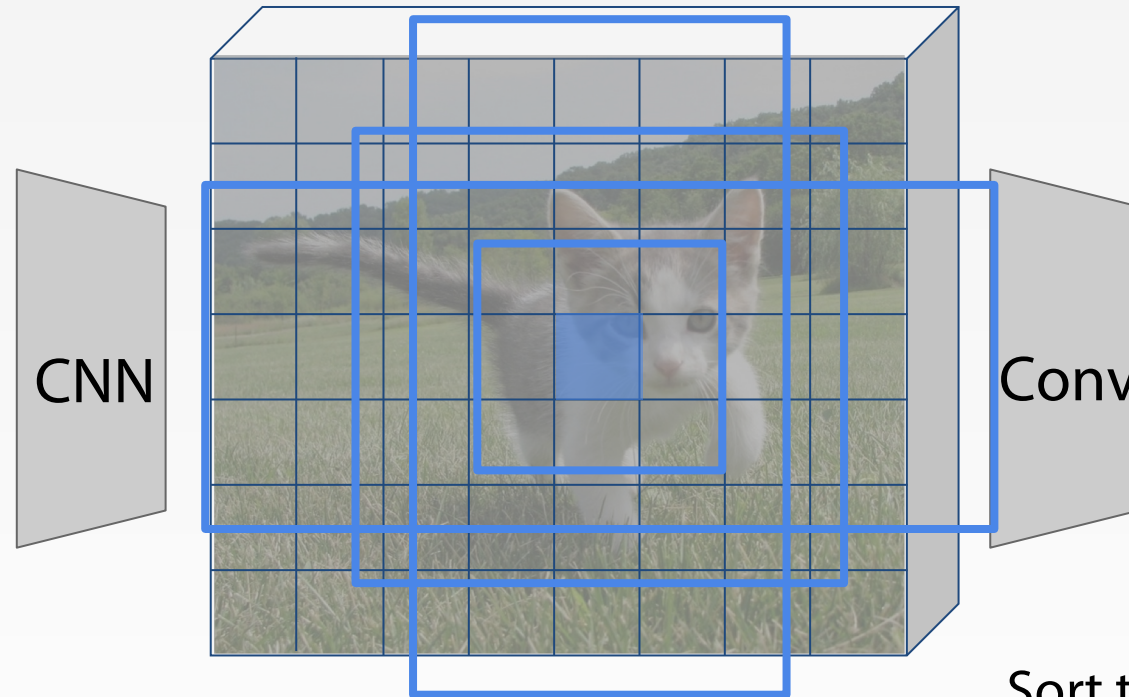


Image features
(e.g. $512 \times 20 \times 15$)

Conv

Anchor is an object?

1K $\times 20 \times 15$

Box transforms

4K $\times 20 \times 15$

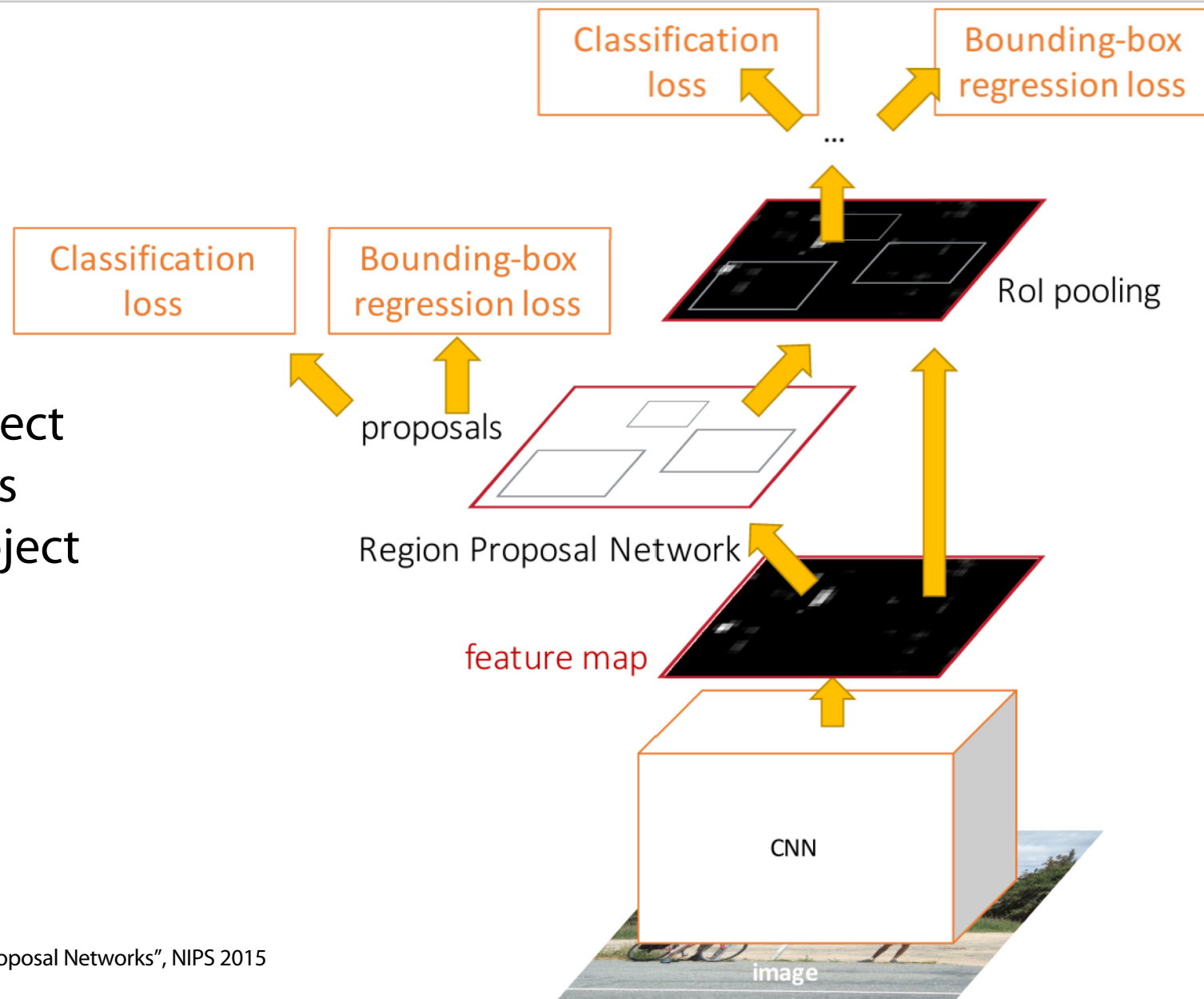
Sort the $K \times 20 \times 15$ boxes by their "object" score, take top ~ 300 as our proposals

FASTER R-CNN:

MAKE CNN DO PROPOSALS!

Jointly train with 4 losses:

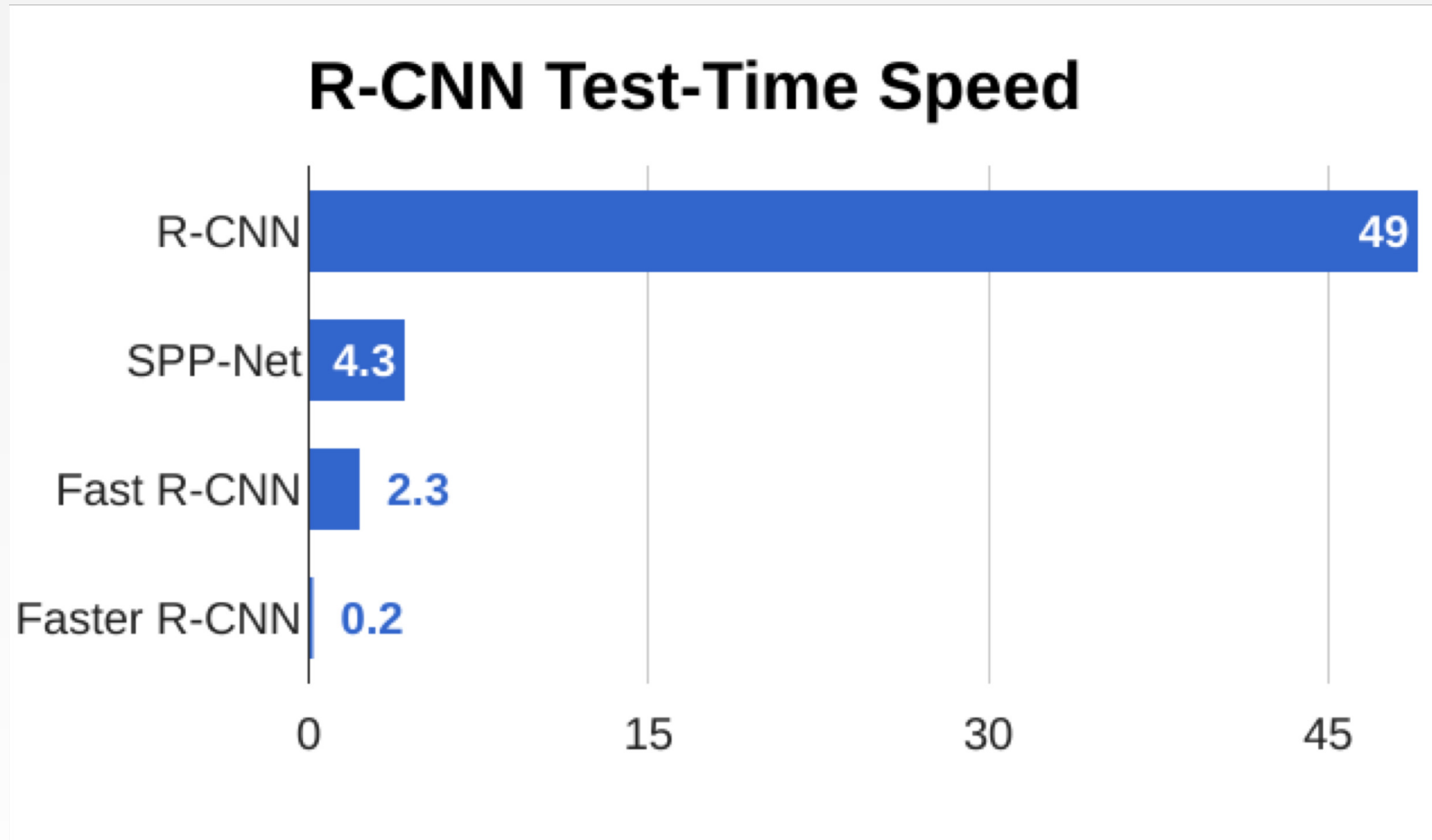
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



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

FASTER R-CNN:

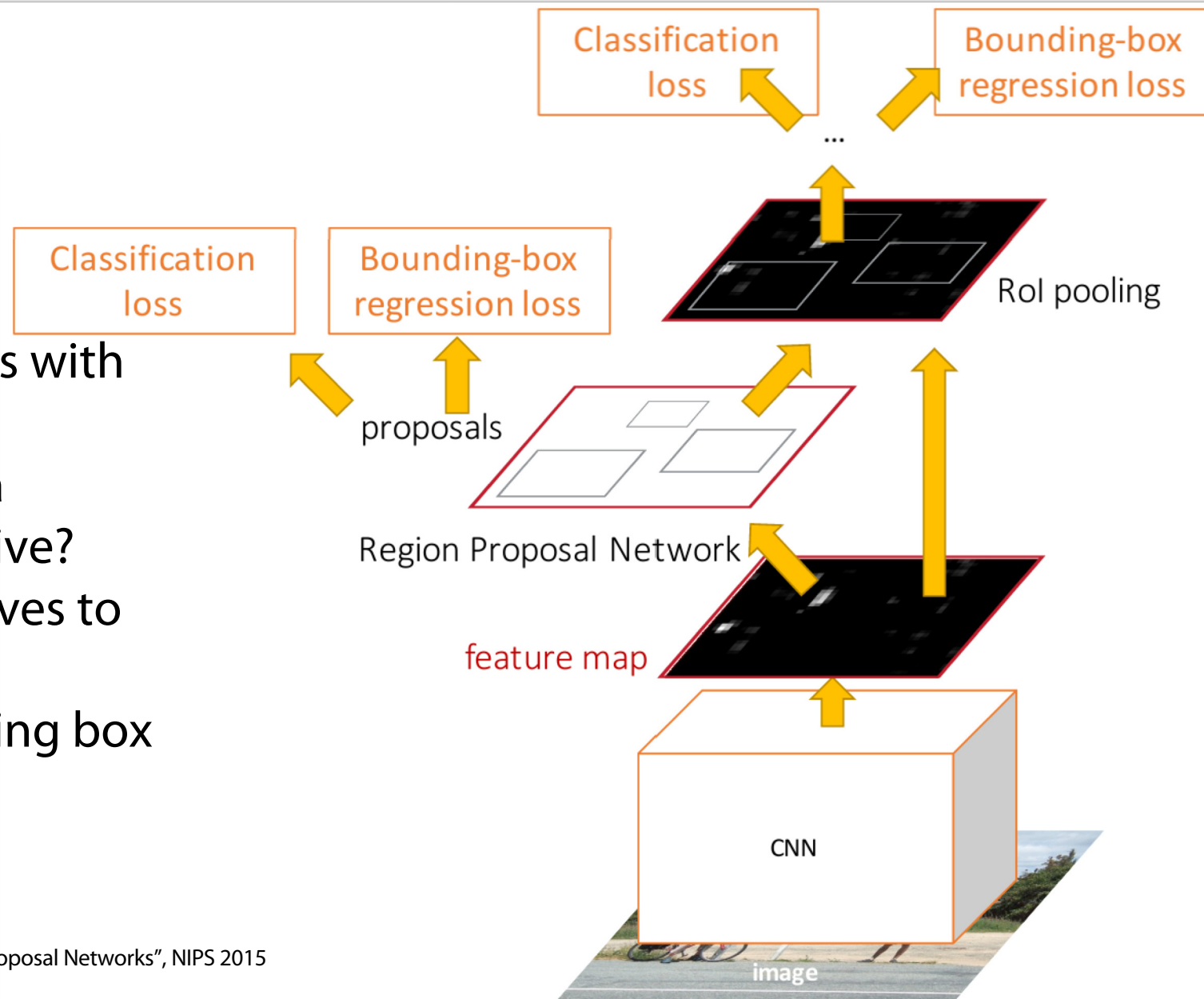
MAKE CNN DO PROPOSALS!



FASTER R-CNN: MAKE CNN DO PROPOSALS!

Glossing over many details:

- Ignore overlapping proposals with **non-max suppression**
- How to determine whether a proposal is positive or negative?
- How many positives / negatives to send to second stage?
- 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

FASTER R-CNN:

MAKE CNN DO PROPOSALS!

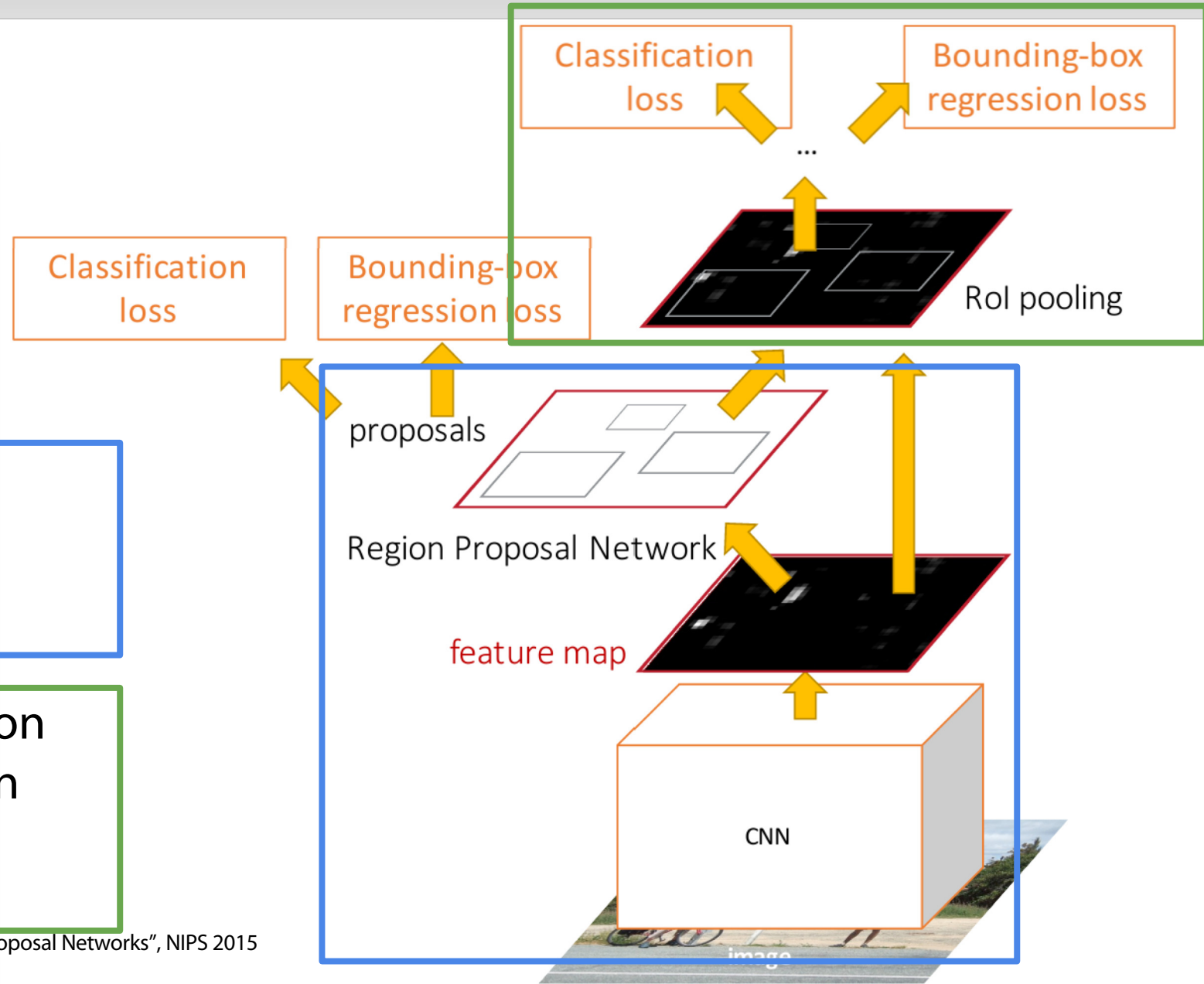
Faster R-CNN is a Two-stage object detector

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
Figure copyright 2015, Ross Girshick; reproduced with permission

FASTER R-CNN:

MAKE CNN DO PROPOSALS!

Faster R-CNN is a Two-stage object detector

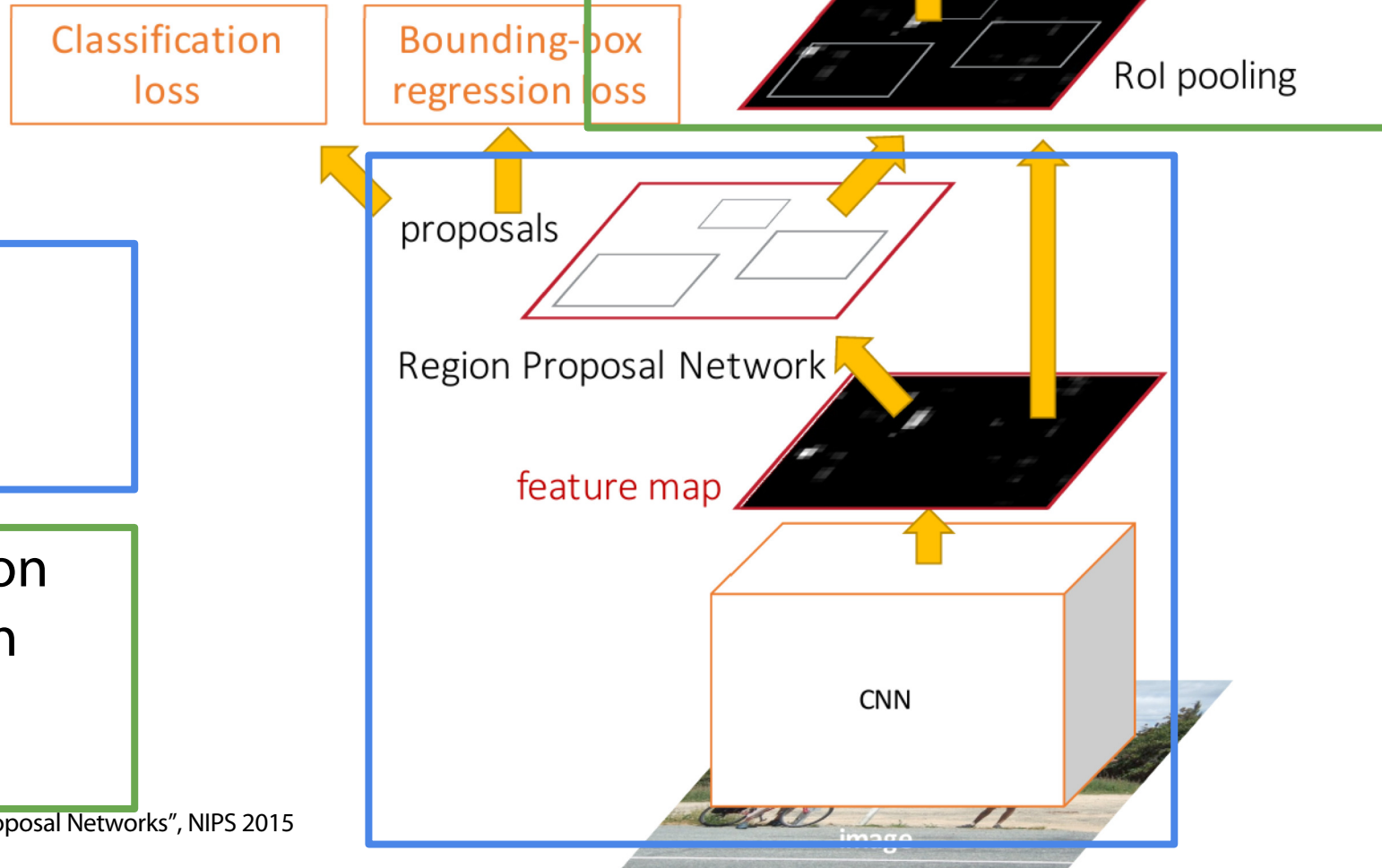
First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

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

Do we really need the second stage?



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

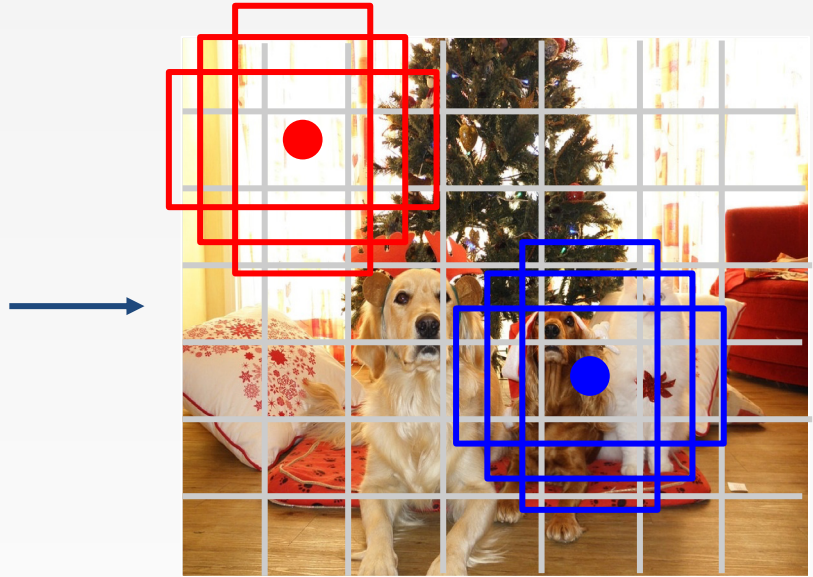


SINGLE-STAGE OBJECT DETECTORS

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

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

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

OBJECT DETECTION: LOTS OF VARIABLES ...

Backbone Network

VGG16

ResNet-101

Inception V2

Inception V3

Inception ResNet

MobileNet

“Meta-Architecture”

Two-stage: Faster R-CNN

Single-stage: YOLO / SSD

Hybrid: R-FCN

Image Size

Region Proposals

...

Takeaways

Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Bigger / Deeper backbones work better

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

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

R-FCN: Dai et al, “R-FCN: Object Detection via Region-based Fully Convolutional Networks”, NIPS 2016

Inception-V2: Ioffe and Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, ICML 2015

Inception V3: Szegedy et al, “Rethinking the Inception Architecture for Computer Vision”, arXiv 2016

Inception ResNet: Szegedy et al, “Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning”, arXiv 2016

MobileNet: Howard et al, “Efficient Convolutional Neural Networks for Mobile Vision Applications”, arXiv 2017



INSTANCE SEGMENTATION

INSTANCE SEGMENTATION

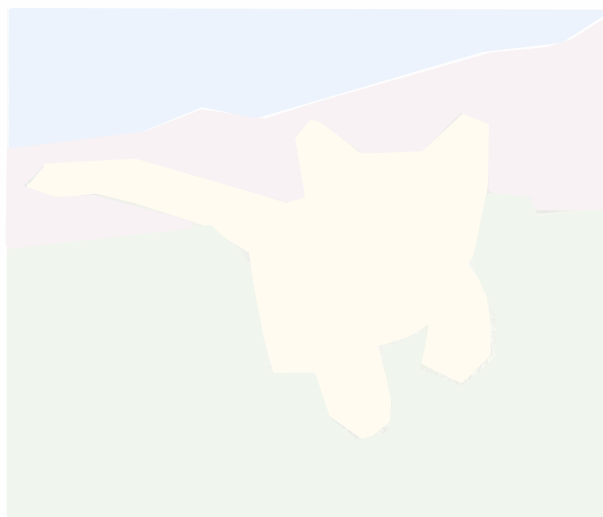
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

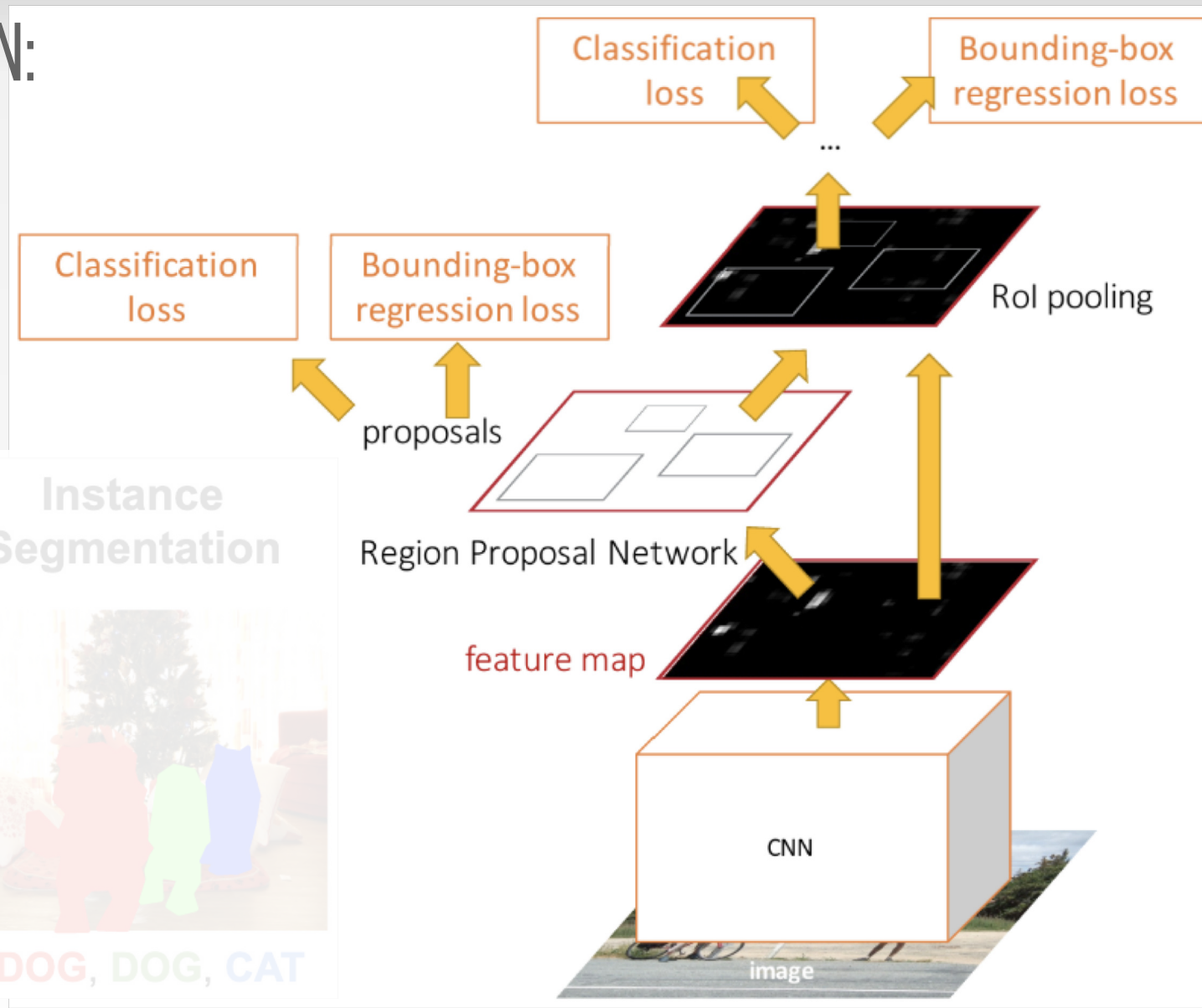
Multiple Object

Instance Segmentation

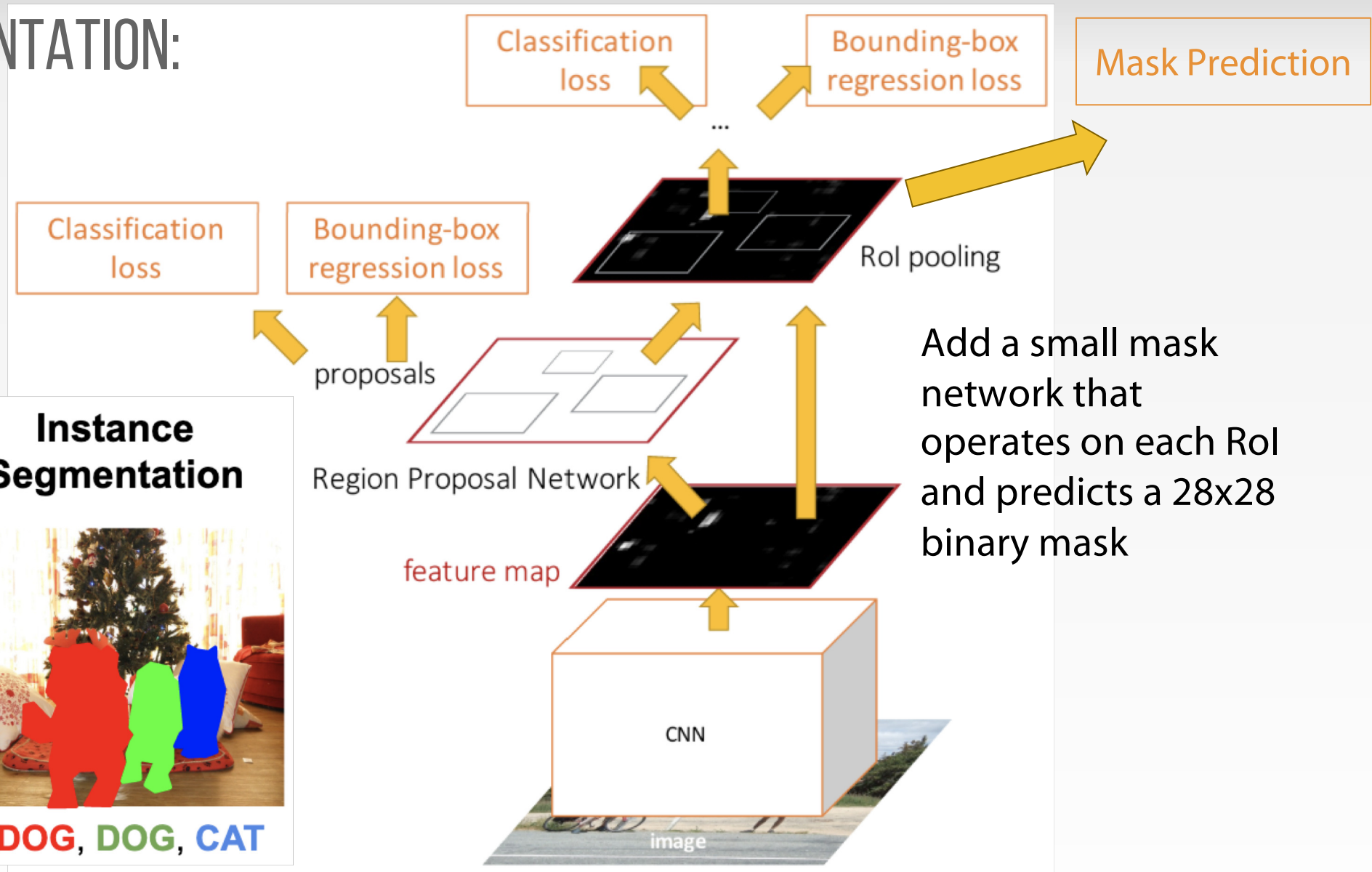
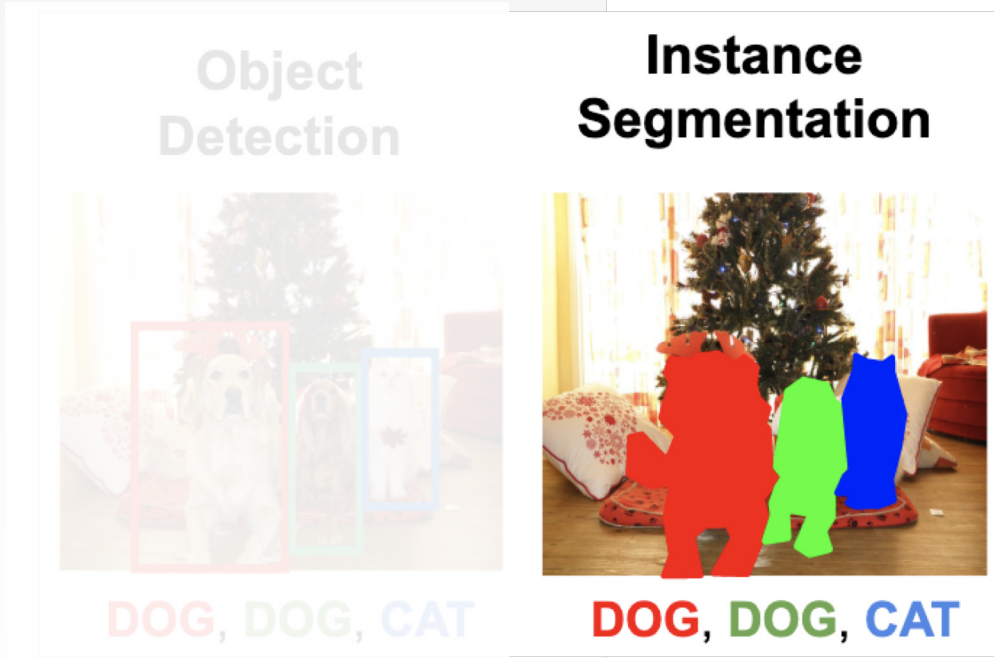


DOG, DOG, CAT

OBJECT DETECTION: FASTER R-CNN



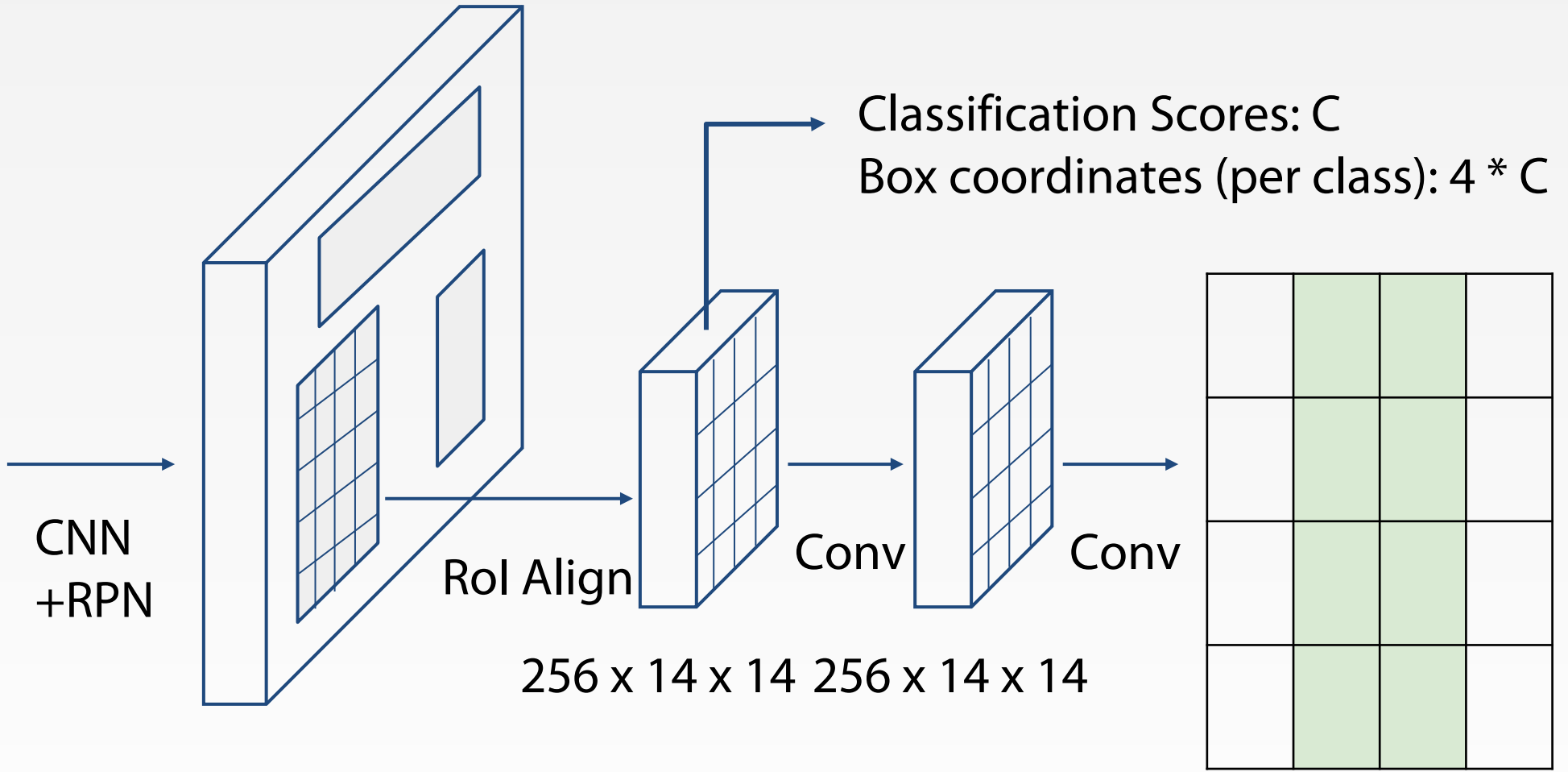
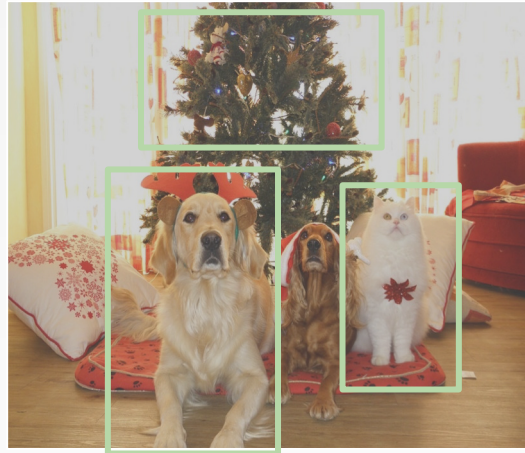
INSTANCE SEGMENTATION: MASK R-CNN



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

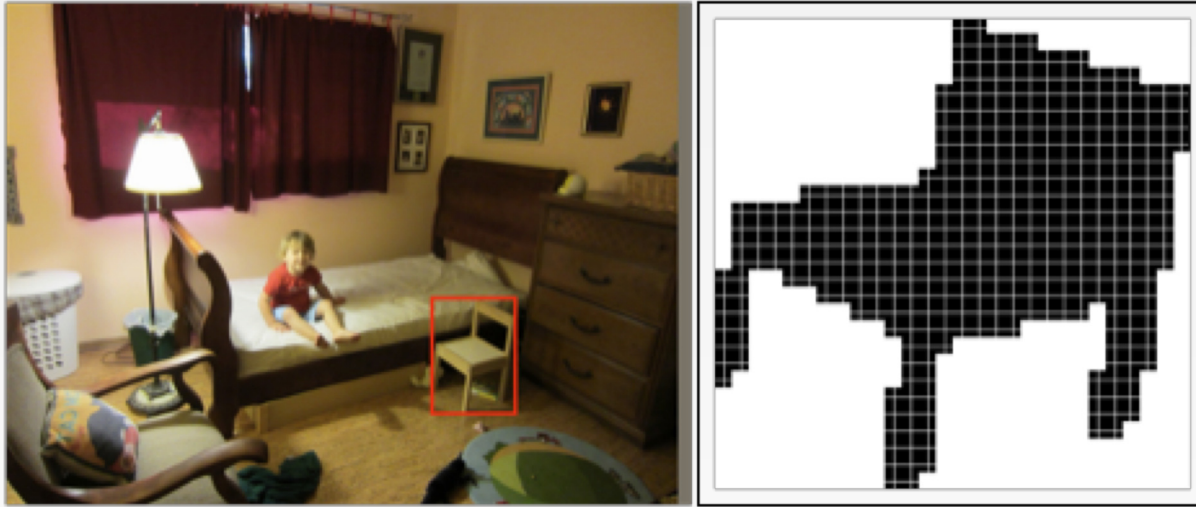
He et al, "Mask R-CNN", ICCV 2017

MASK R-CNN

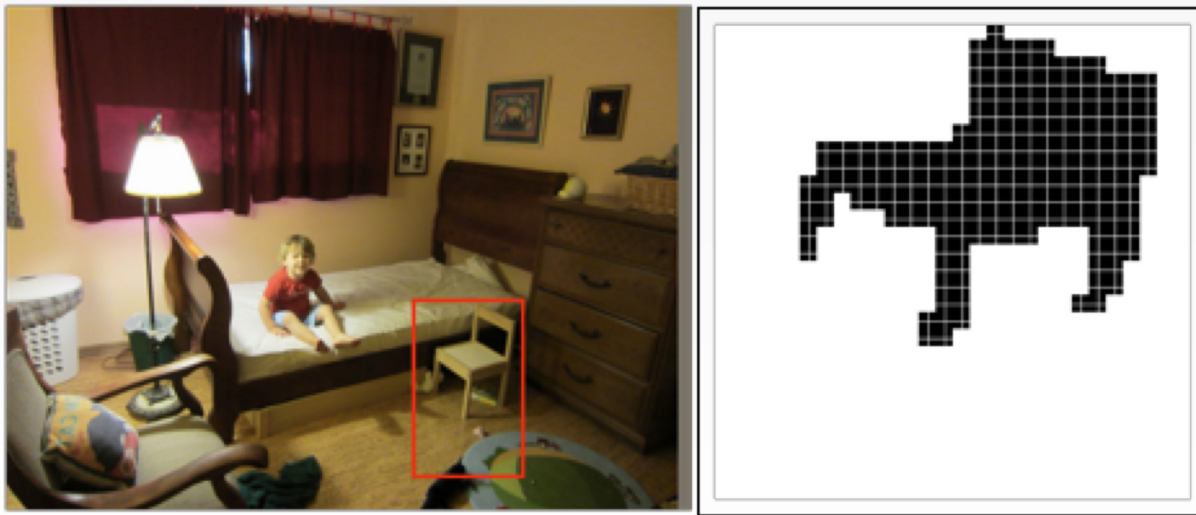
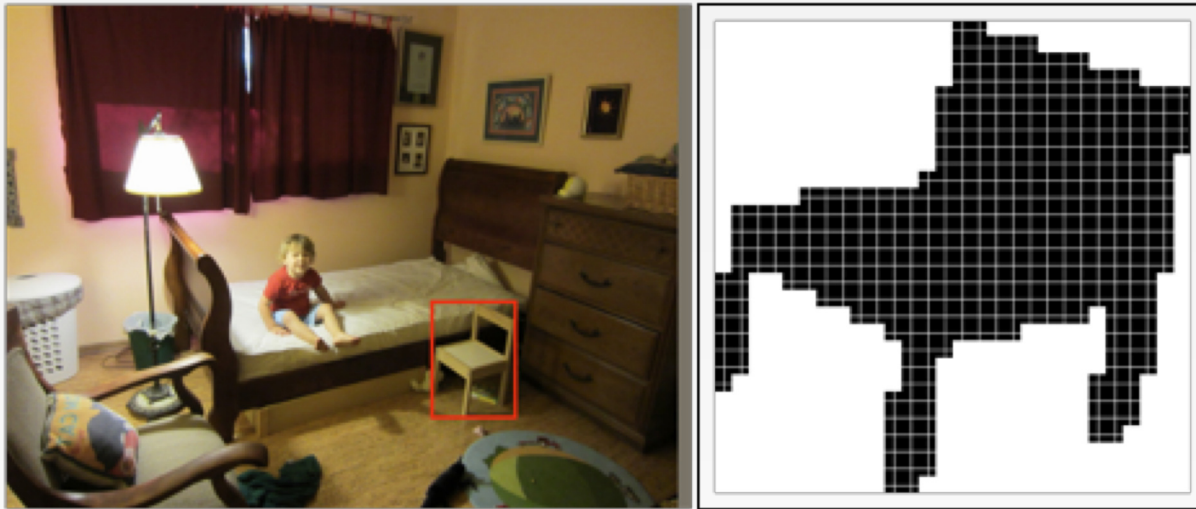


He et al, "Mask R-CNN", ICCV 2017

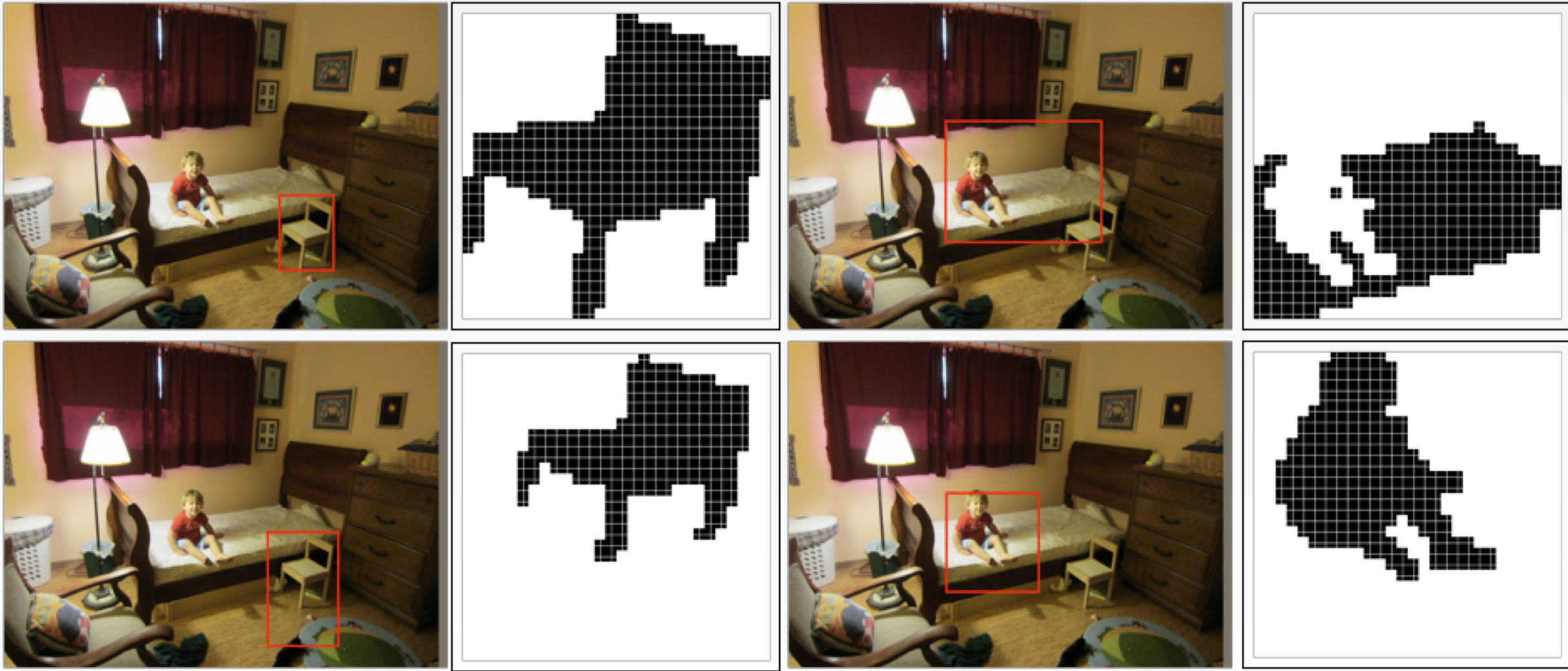
MASK R-CNN: EXAMPLE MASK TRAINING TARGETS



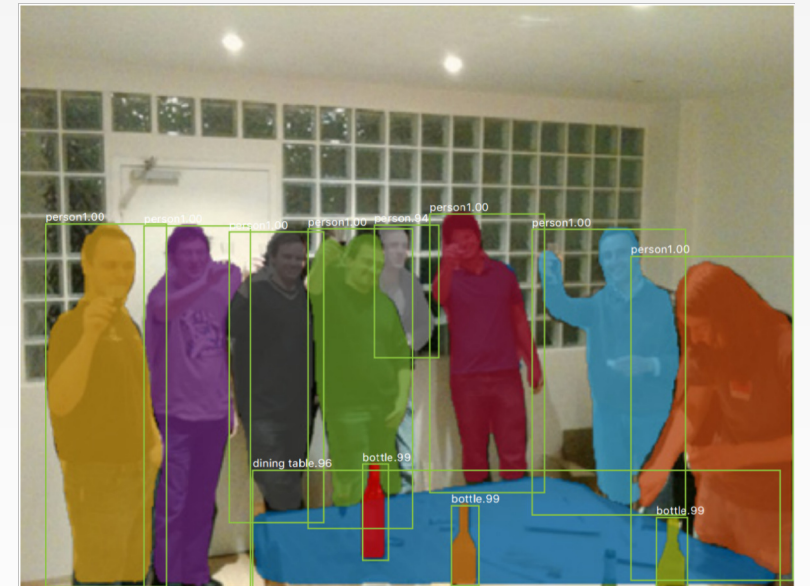
MASK R-CNN: EXAMPLE MASK TRAINING TARGETS



MASK R-CNN: EXAMPLE MASK TRAINING TARGETS

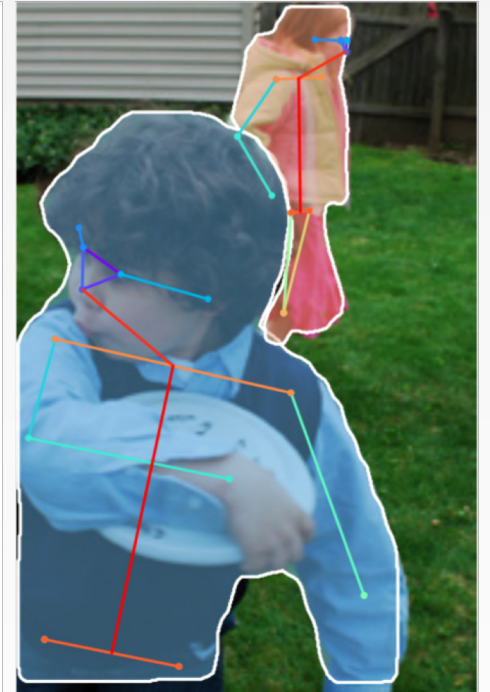
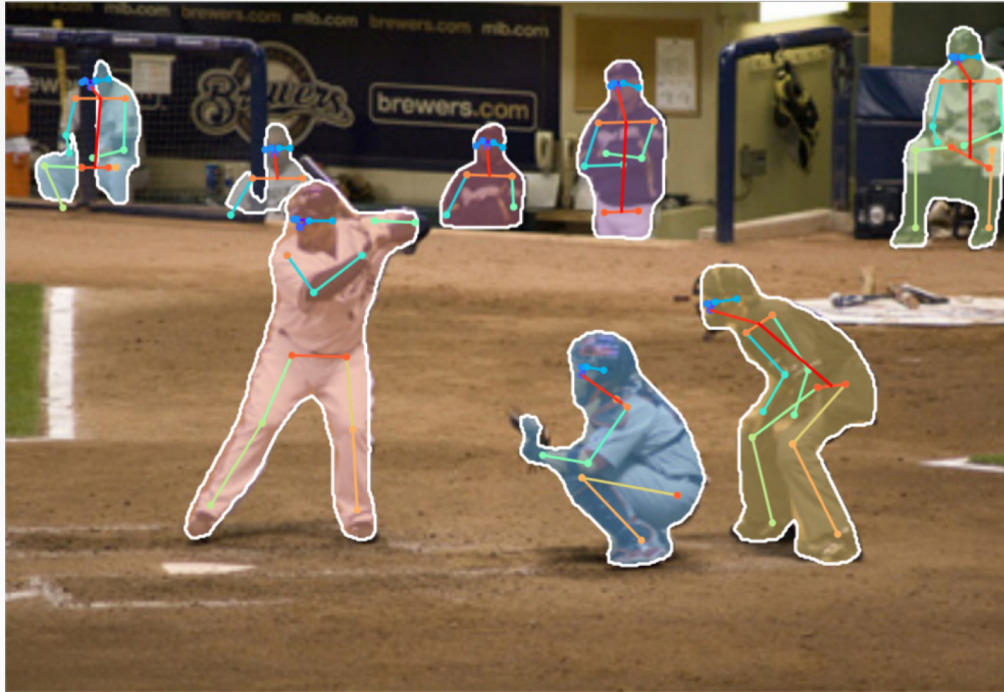


MASK R-CNN: VERY GOOD RESULTS!



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

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

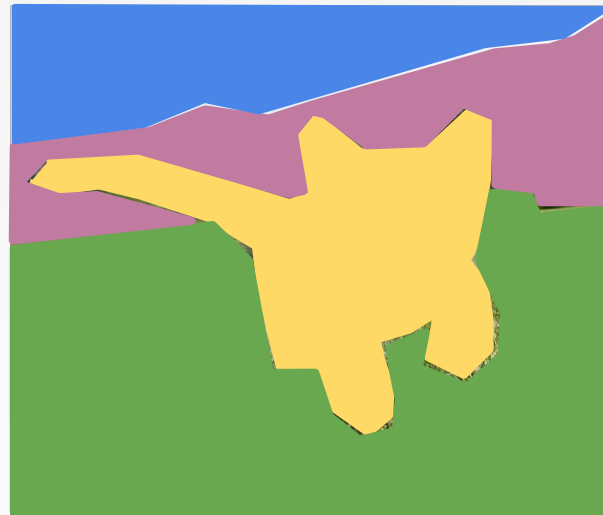
Classification



CAT

No spatial
extent

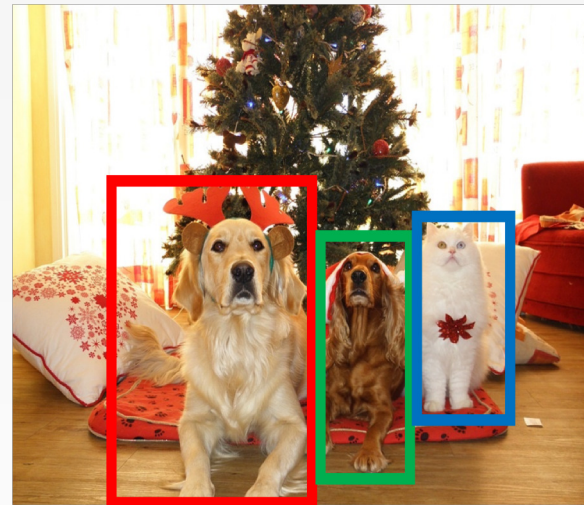
Semantic Segmentation



**GRASS, CAT,
TREE, SKY**

No objects, just
pixels

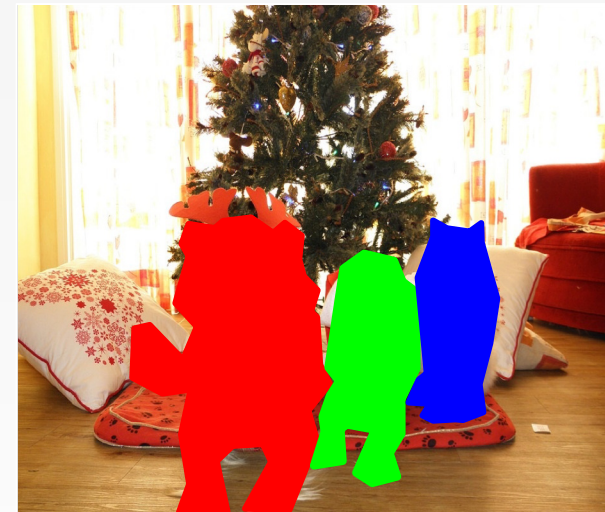
Object Detection



DOG, DOG, CAT

Multiple
Object

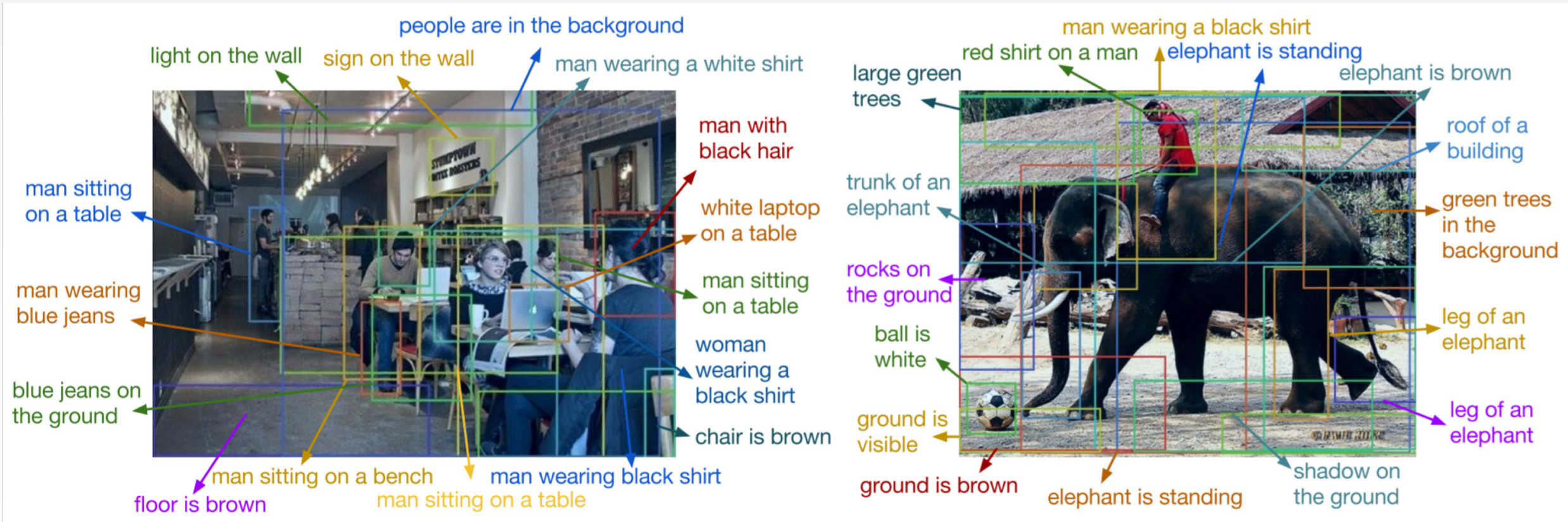
Instance Segmentation



DOG, DOG, CAT

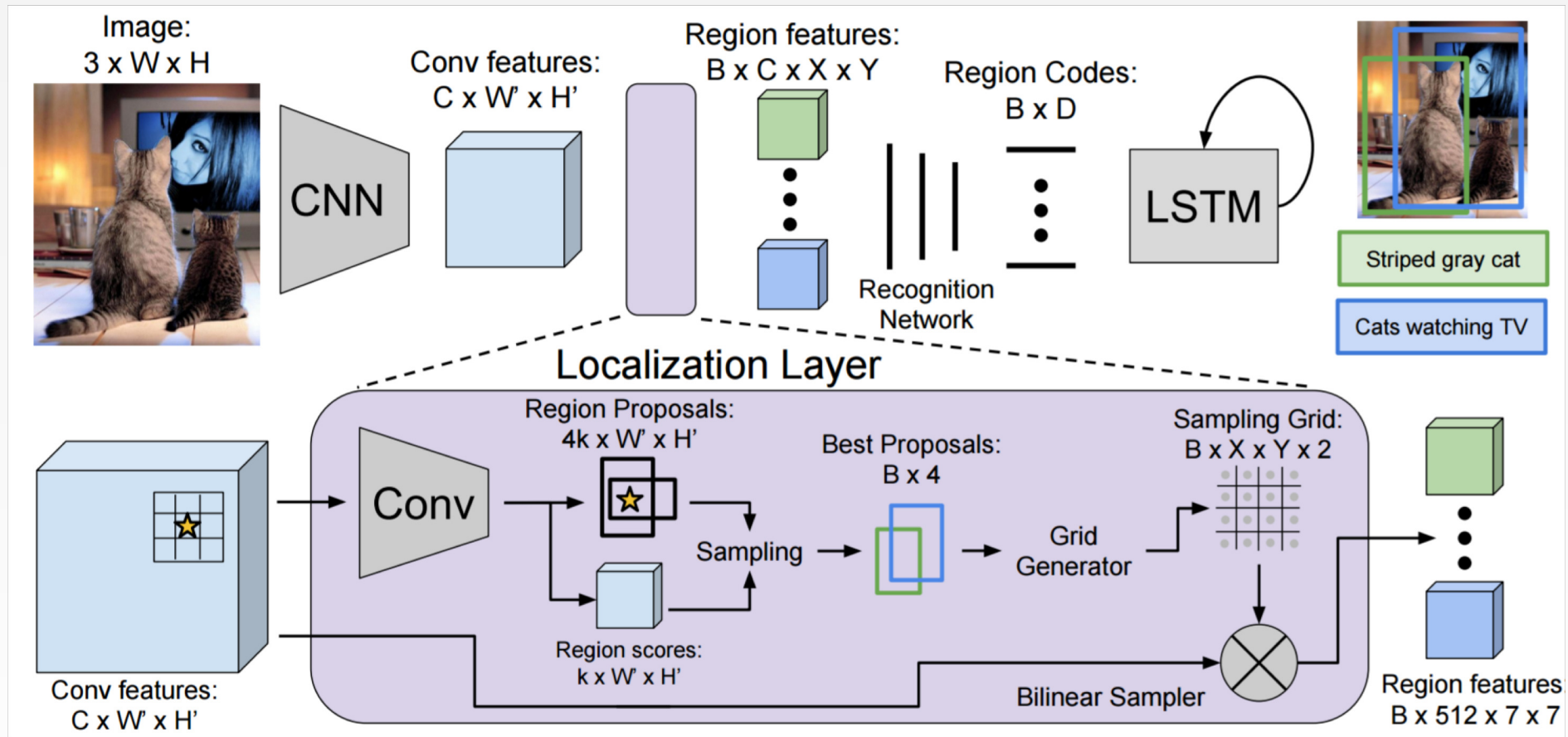
BEYOND 2D OBJECT DETECTION...

OBJECT DETECTION + CAPTIONING = DENSE CAPTIONING



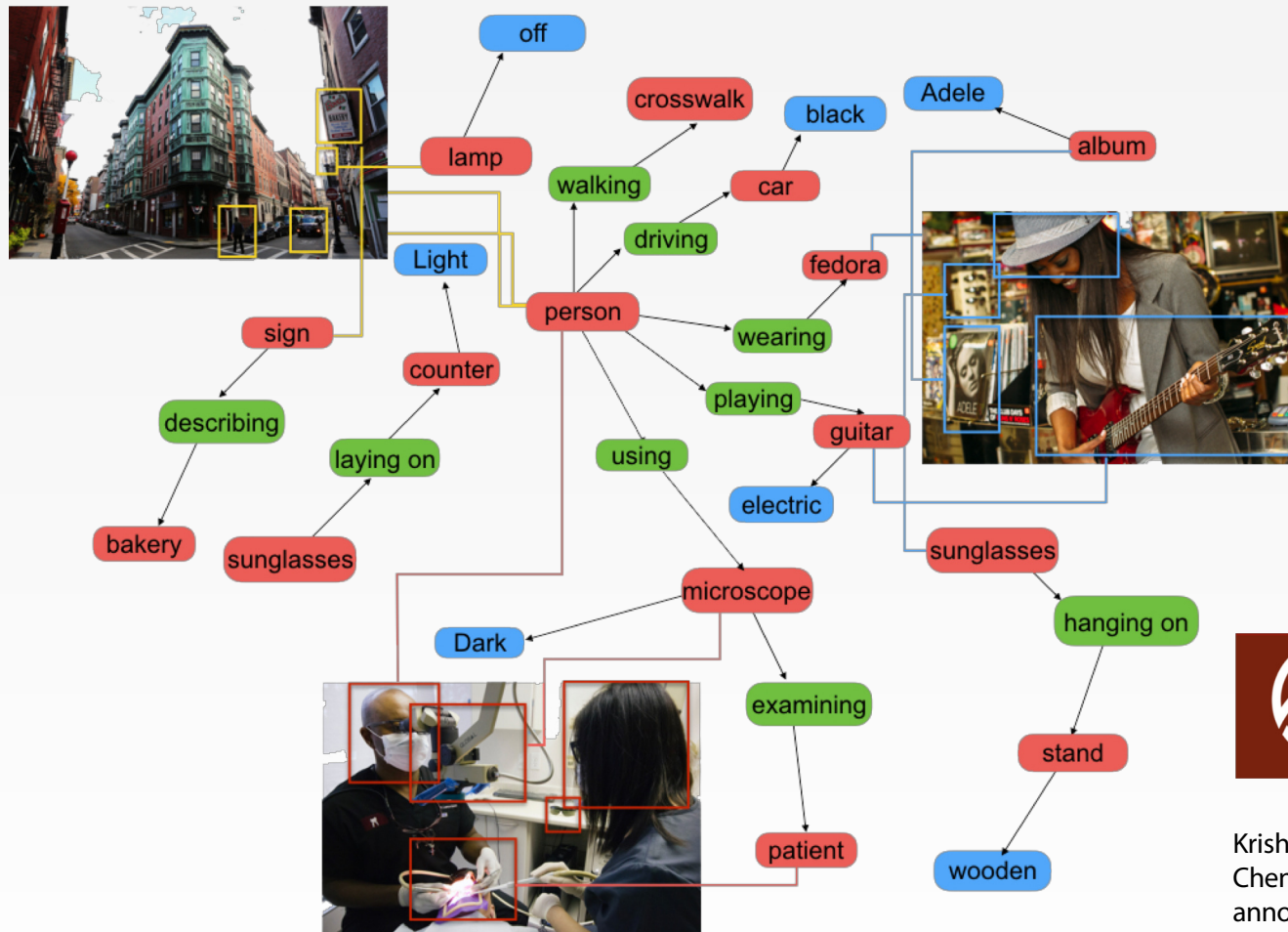
Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.

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

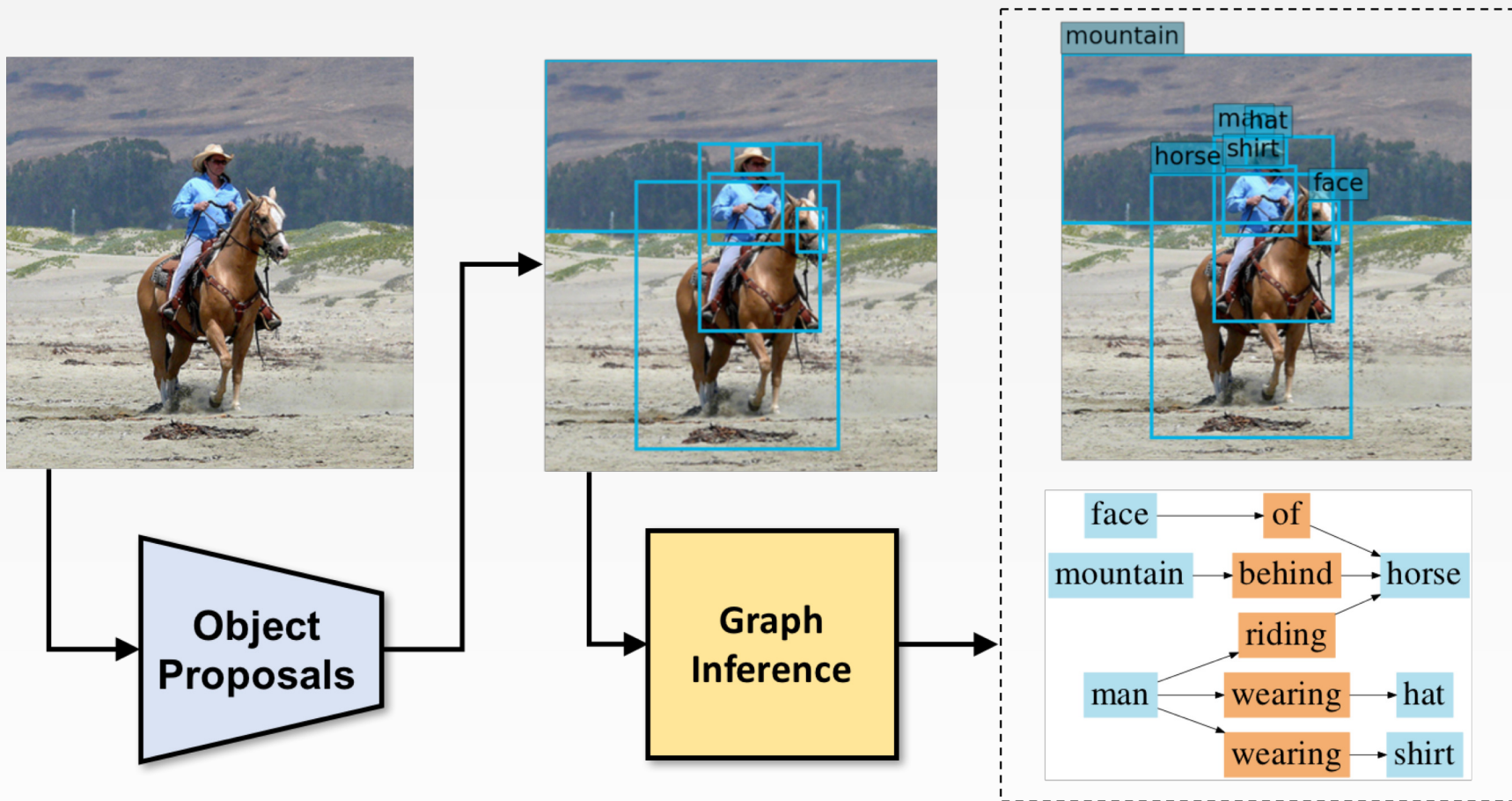


108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets



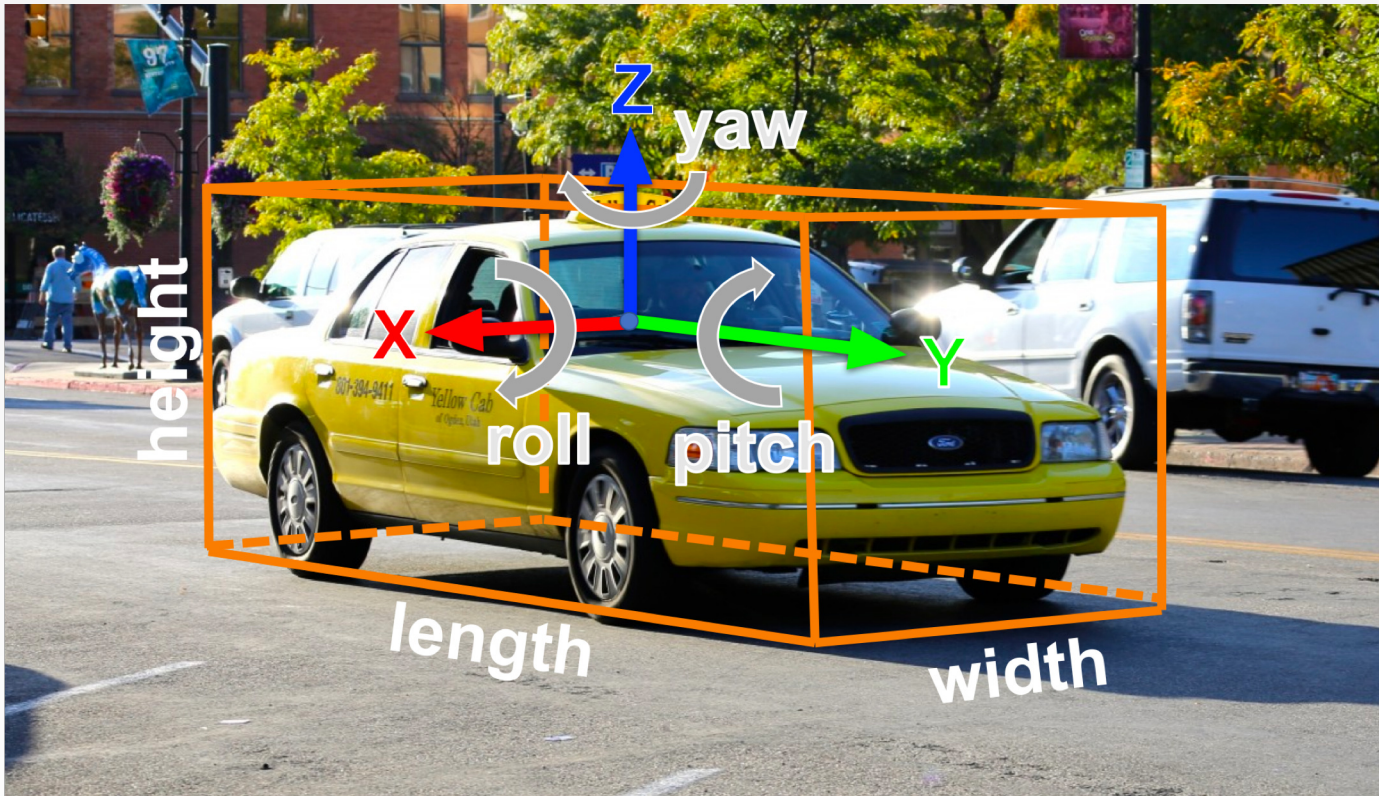
Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." International Journal of Computer Vision 123, no. 1 (2017): 32-73.

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



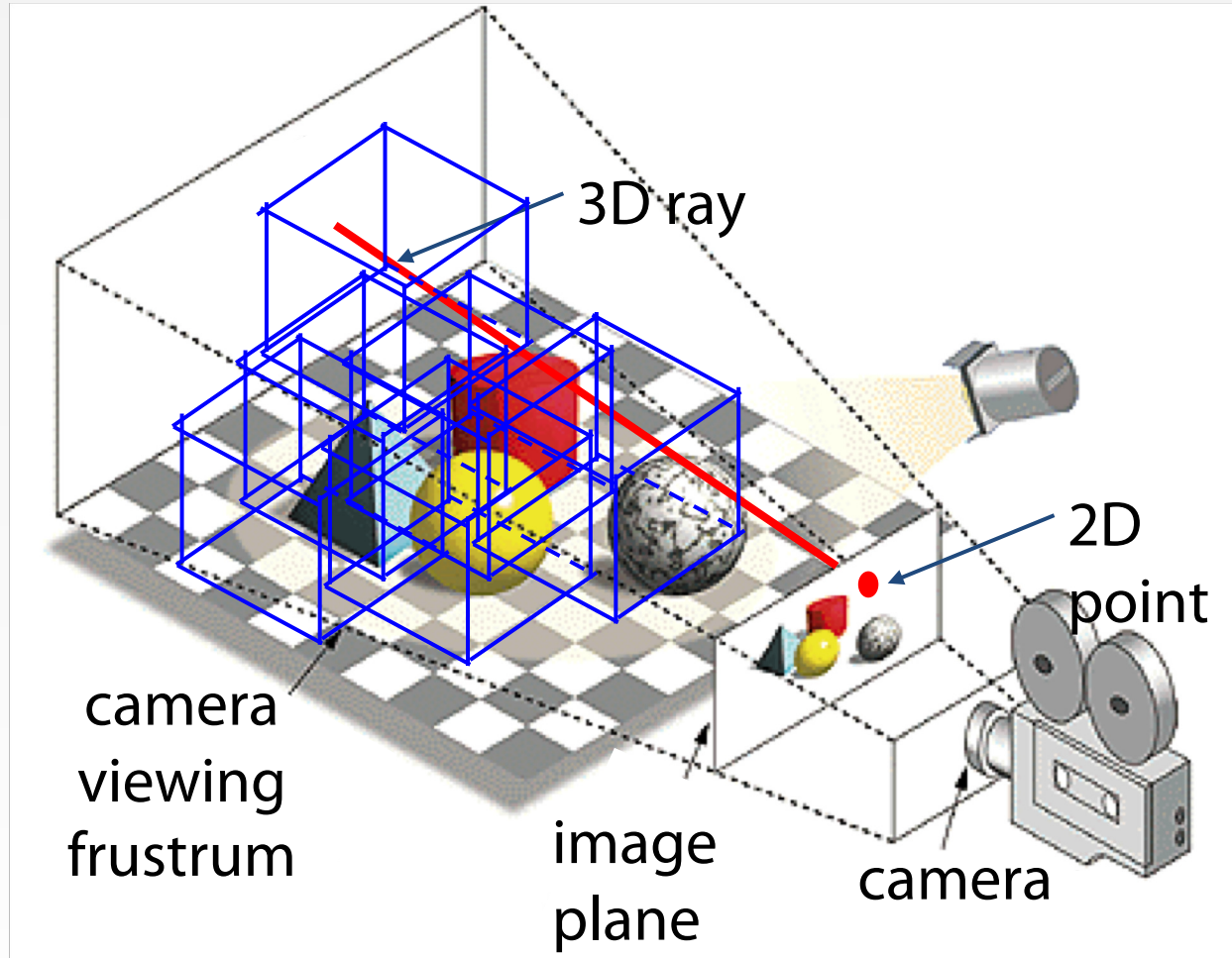
2D Object Detection:
2D bounding box
(x, y, w, h)

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: SIMPLE CAMERA MODEL



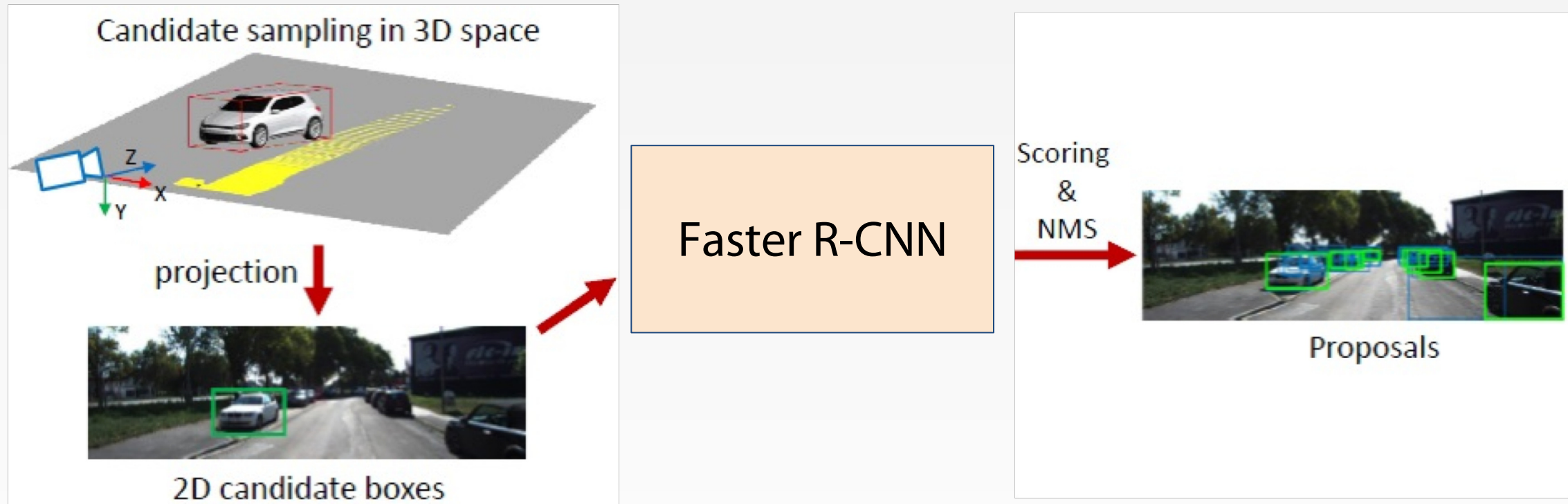
A point on the image plane corresponds to a **ray** in the 3D space

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

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

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

3D OBJECT DETECTION: MONOCULAR CAMERA

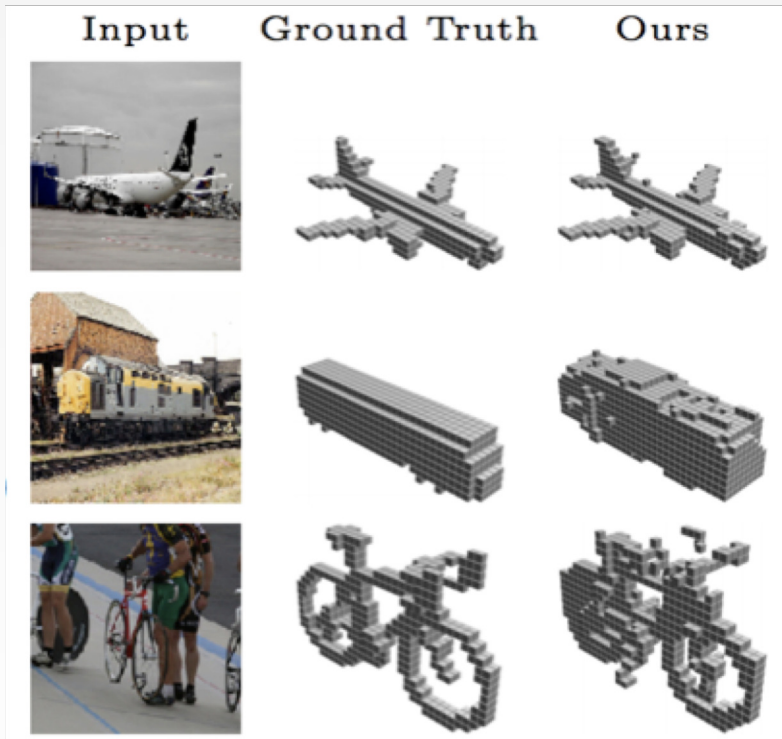


- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

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

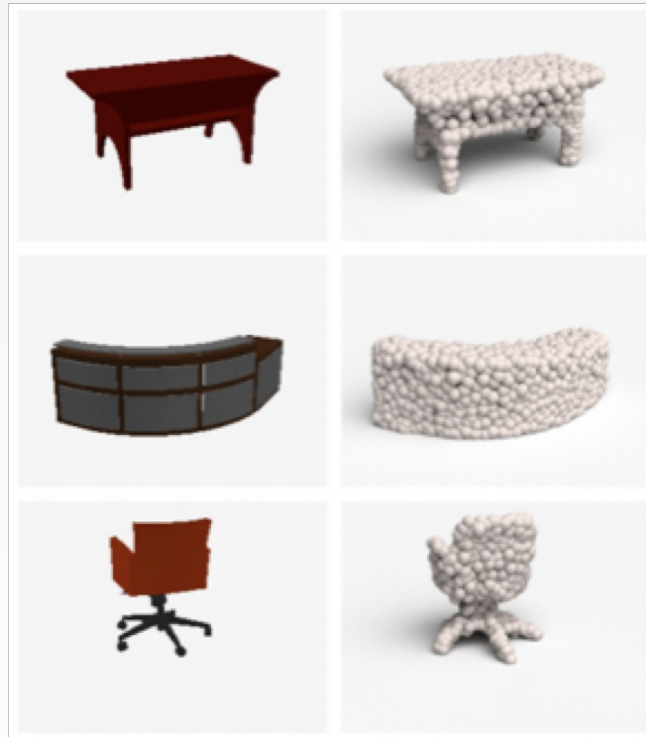
3D SHAPE PREDICTION

Voxel:
D x D x D binary



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

Pointcloud:
V x 3 float



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

Mesh:
V x 3 float, F x 3 int



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

RECAP: LOTS OF COMPUTER VISION TASKS!

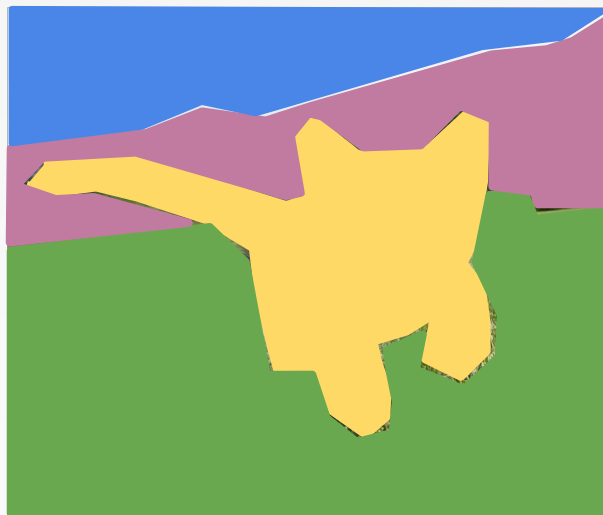
Classification



CAT

No spatial
extent

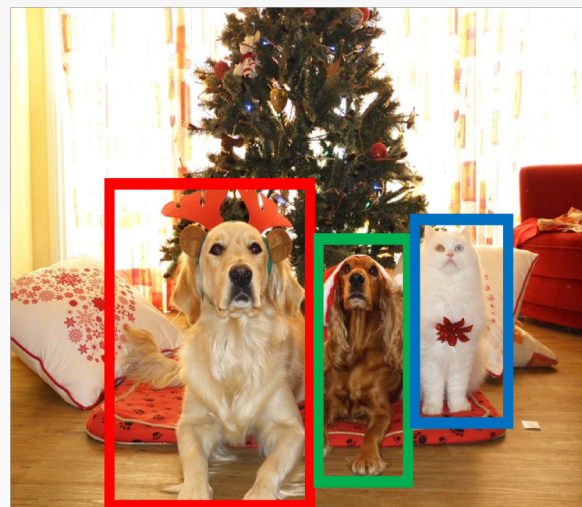
Semantic Segmentation



**GRASS, CAT,
TREE, SKY**

No objects, just
pixels

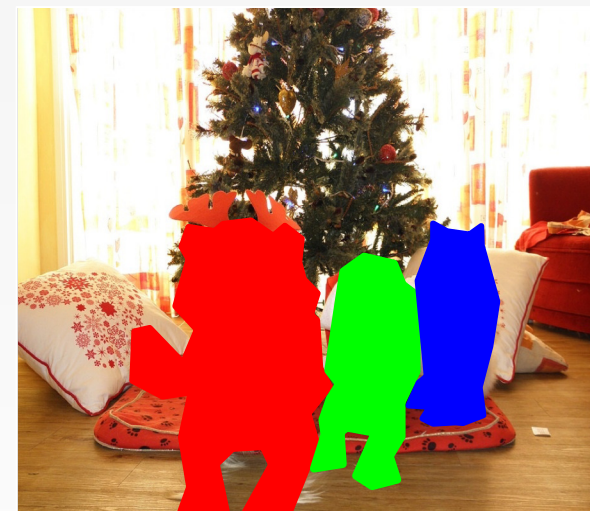
Object Detection



DOG, DOG, CAT

Multiple
Object

Instance Segmentation



DOG, DOG, CAT

NEXT TIME:

VISUALIZING CNN FEATURES

DEEPAIDREAM + STYLE TRANSFER