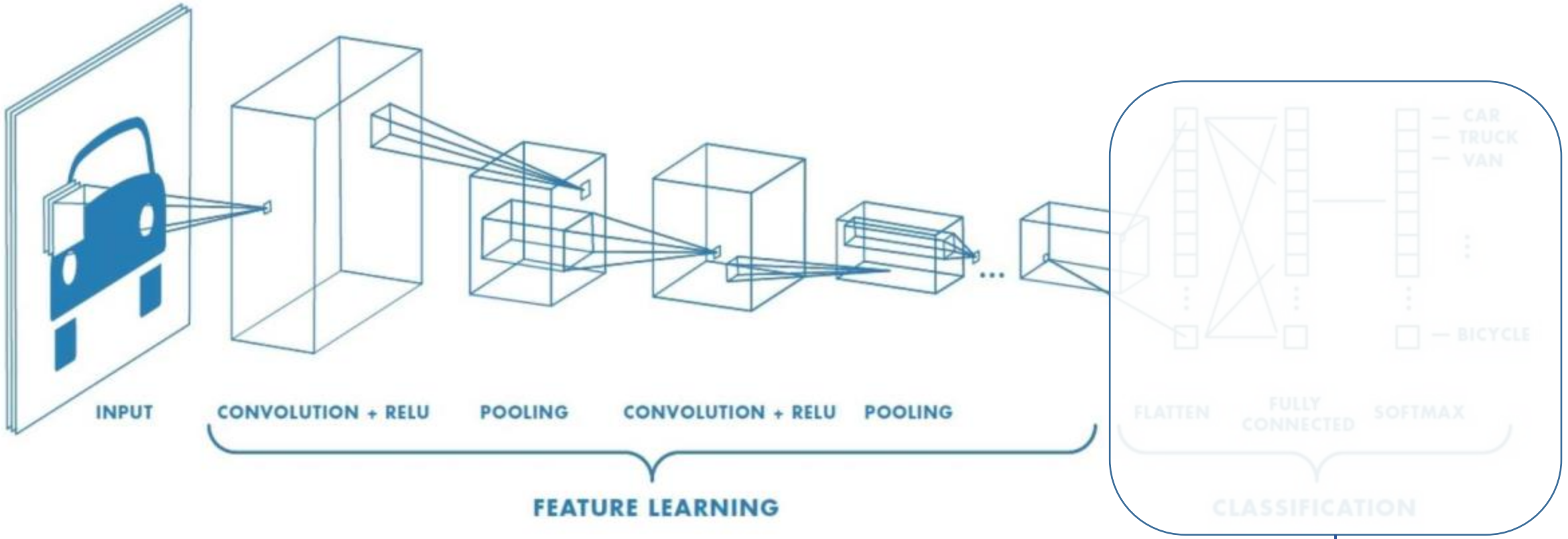
Computer Vision

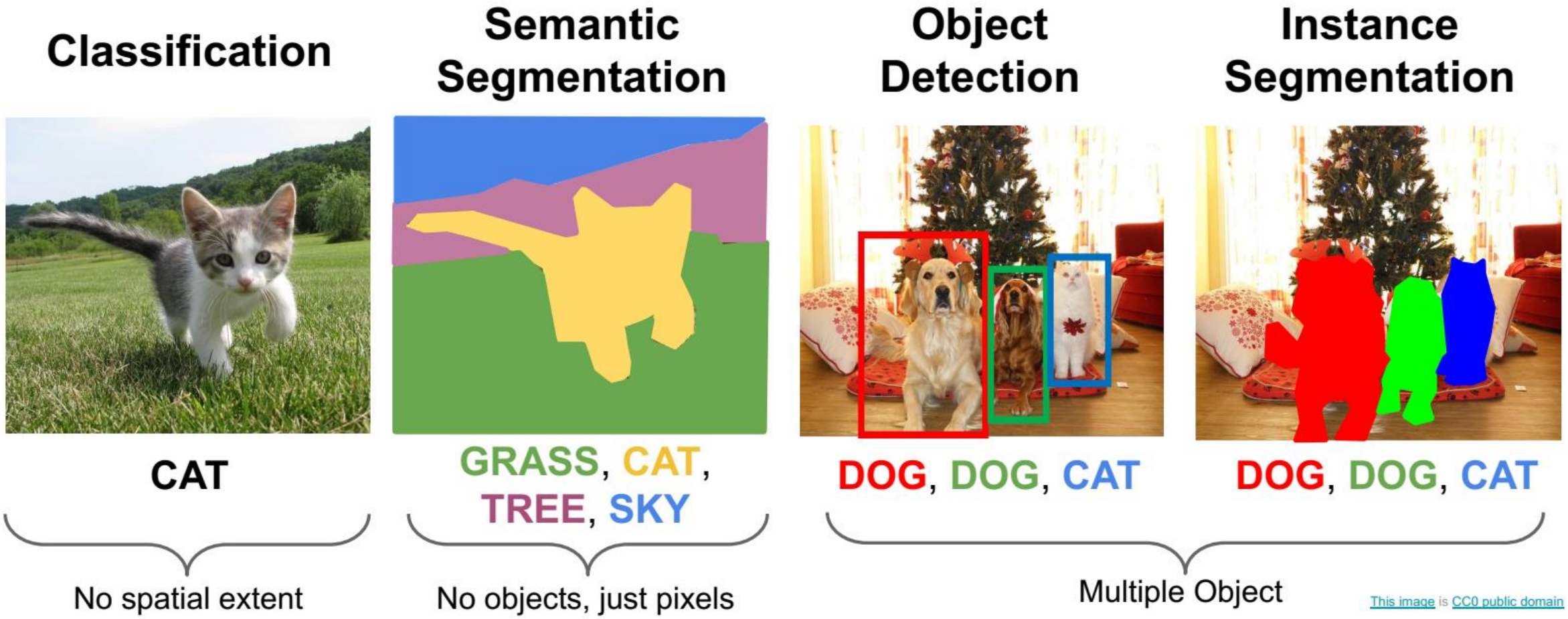
Object Detection & Segmentation

CNNs: *An architecture for many applications*

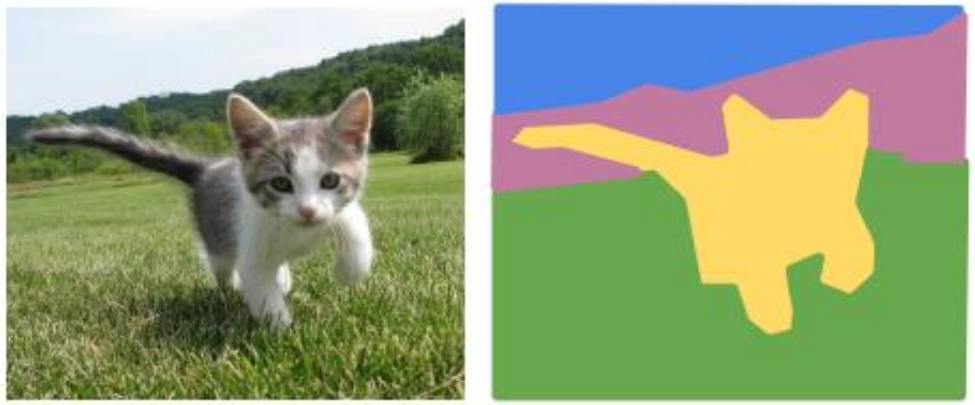


Classification
Object detection
Segmentation
Probabilistic control

Beyond classification

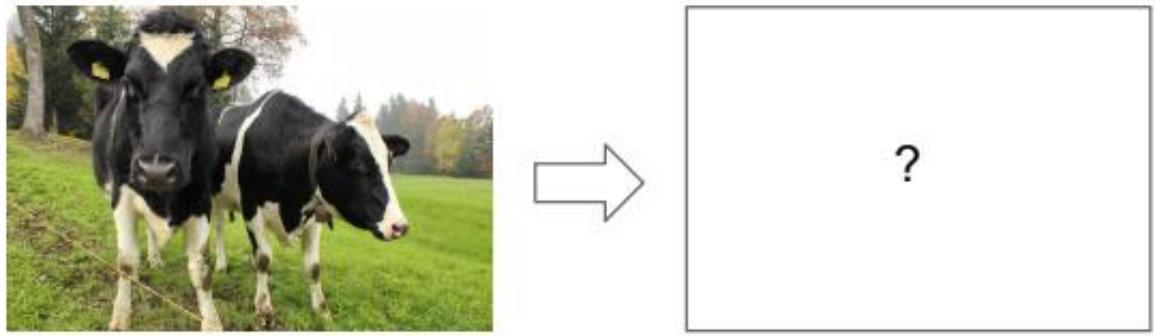


Semantic segmentation: *The problem*



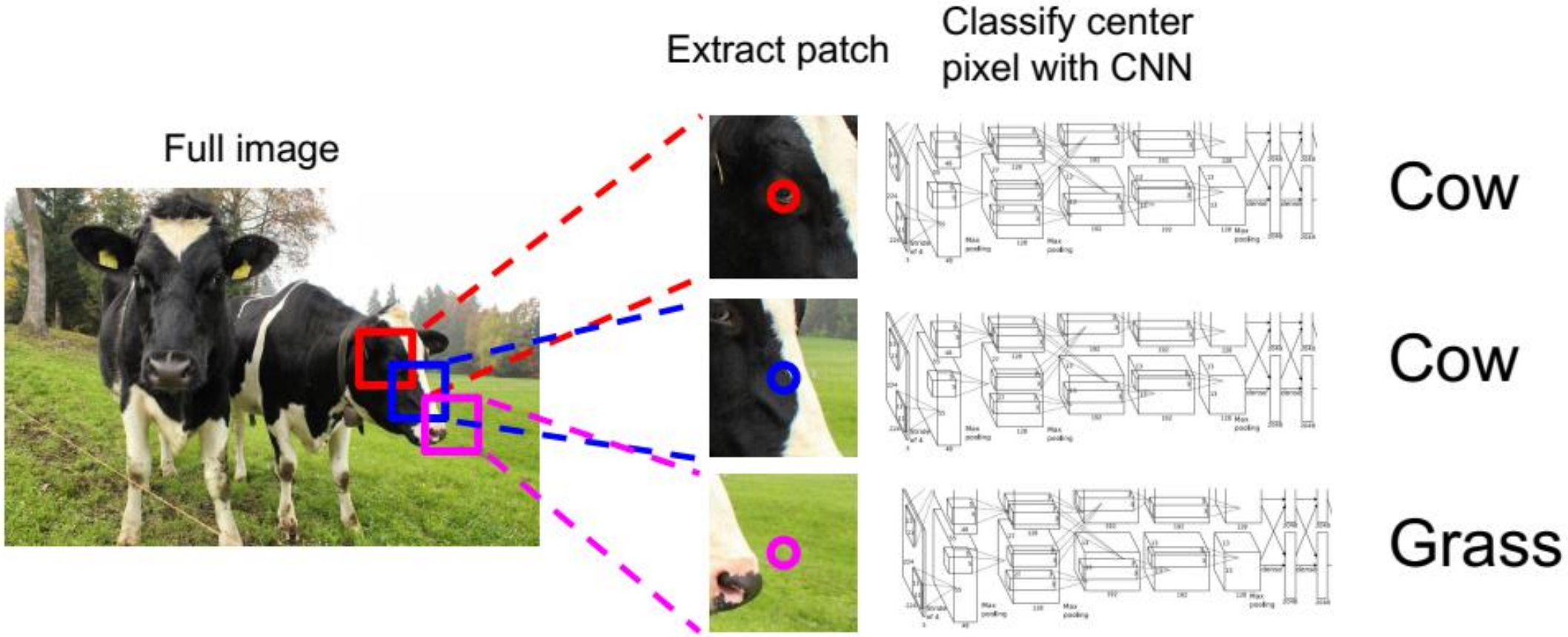
GRASS, CAT,
TREE, SKY, ...

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

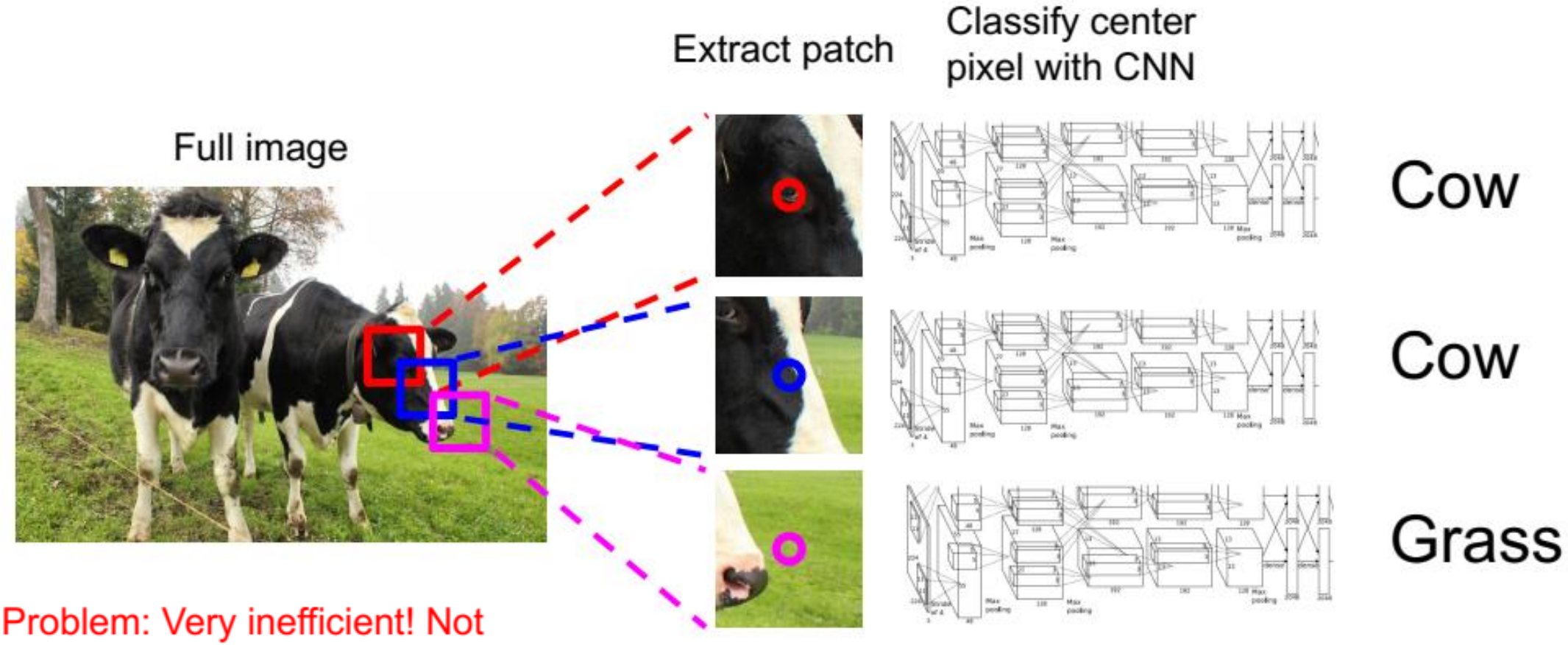


At test time, classify each pixel of a new image.

Semantic segmentation idea: Sliding window

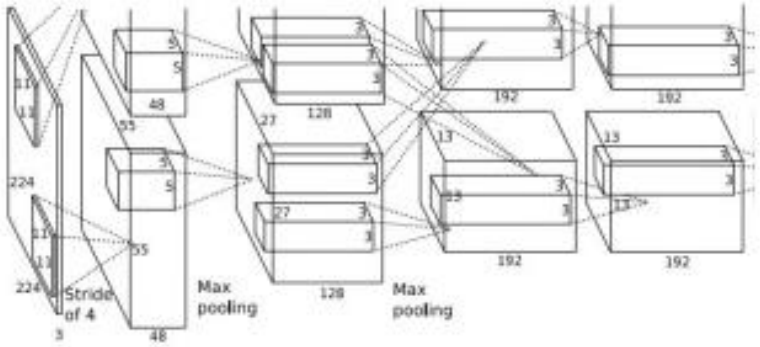


Semantic segmentation idea: Sliding window



Semantic segmentation idea: Convolution

Full image



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

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

Semantic segmentation idea: Fully convolutional

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

Input:
3 x H x W

Conv → Conv → Conv → Conv

Convolutions:
D x H x W

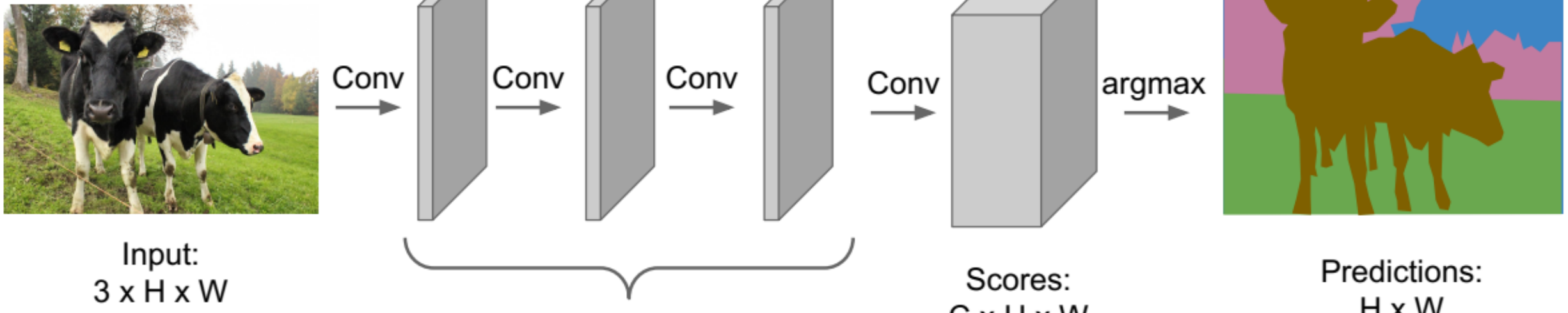
Scores:
C x H x W

argmax →

Predictions:
H x W

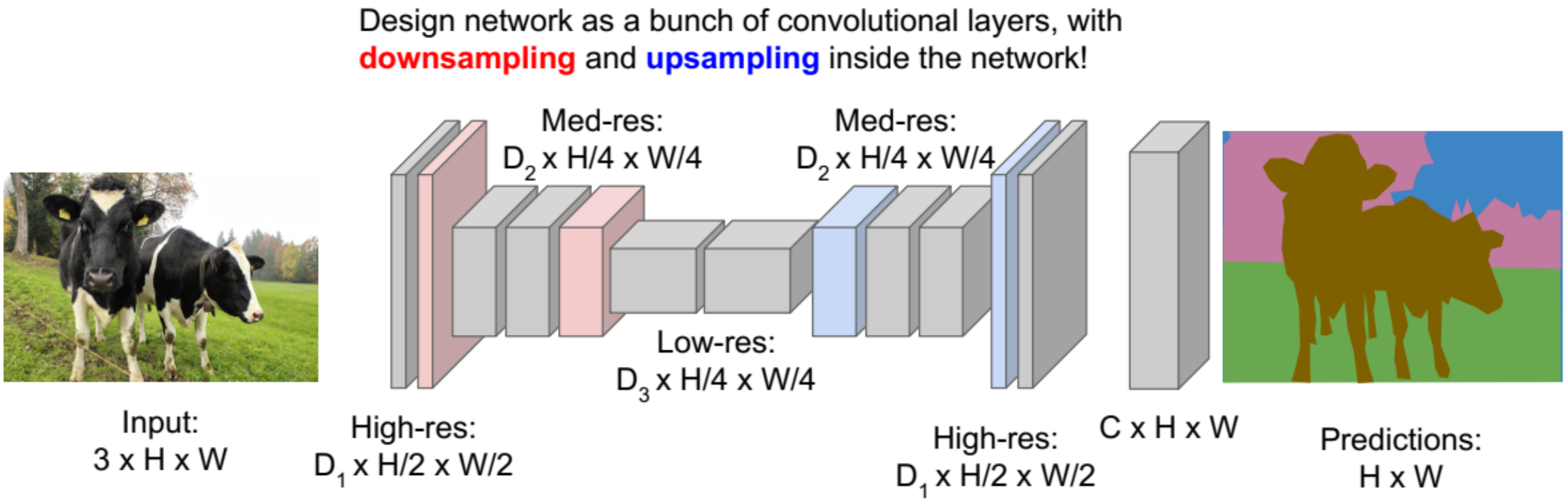
Semantic segmentation idea: Fully convolutional

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



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

Semantic segmentation idea: Fully Convolutional

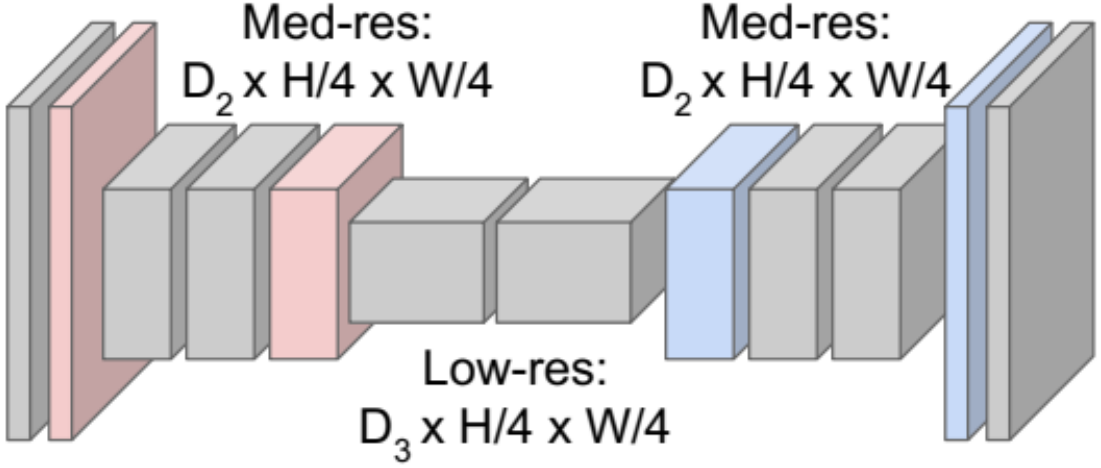


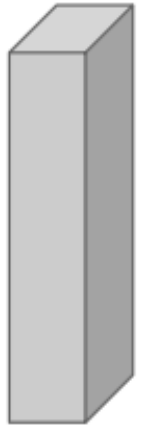
Semantic segmentation idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

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

Upsampling:
???





Input:
3 x H x W

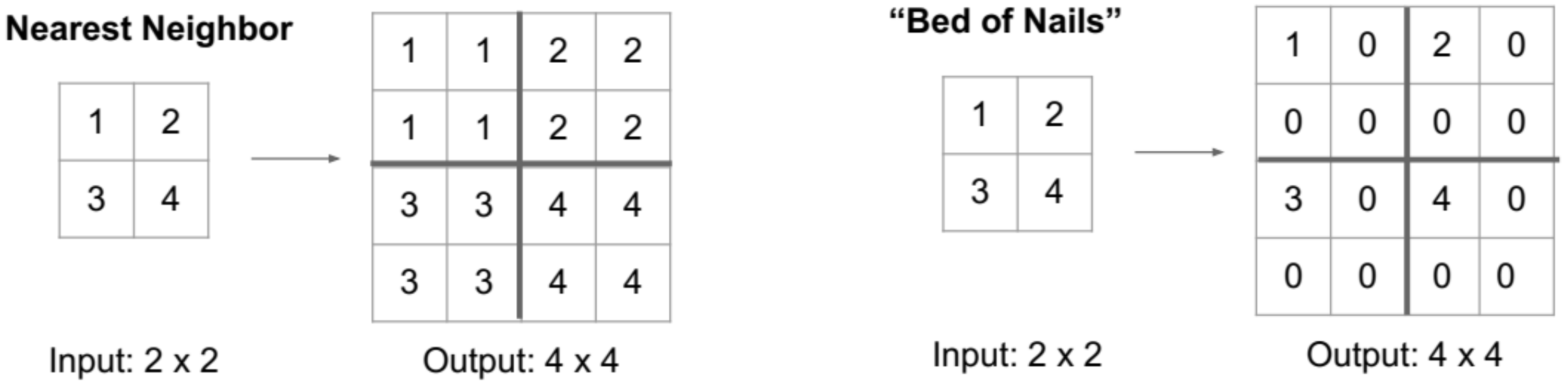
High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

C x H x W

Predictions:
H x W

In-network upsampling: “Unpooling”



In-network upsampling: "*Max Unpooling*"

Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

...

Rest of the network

Max Unpooling
Use positions from pooling layer

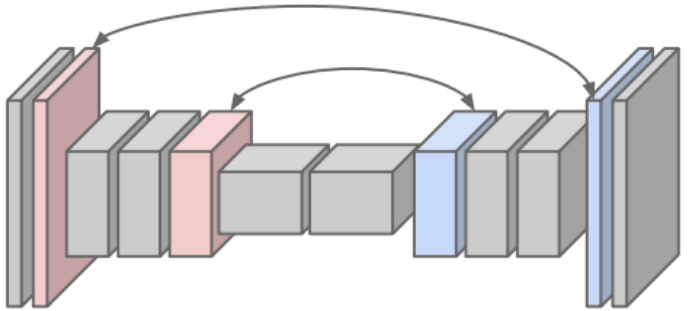
1	2
3	4

Input: 2 x 2

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

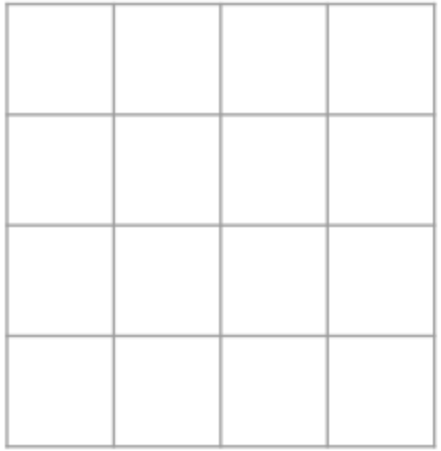
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

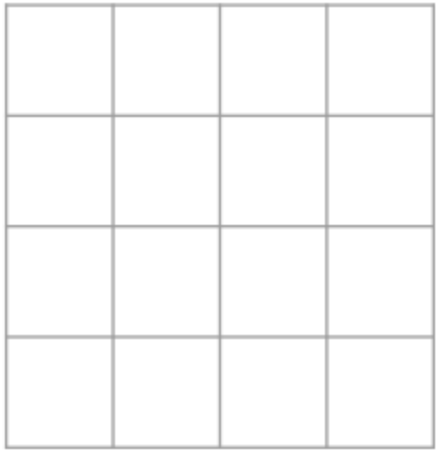


Learnable upsampling

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



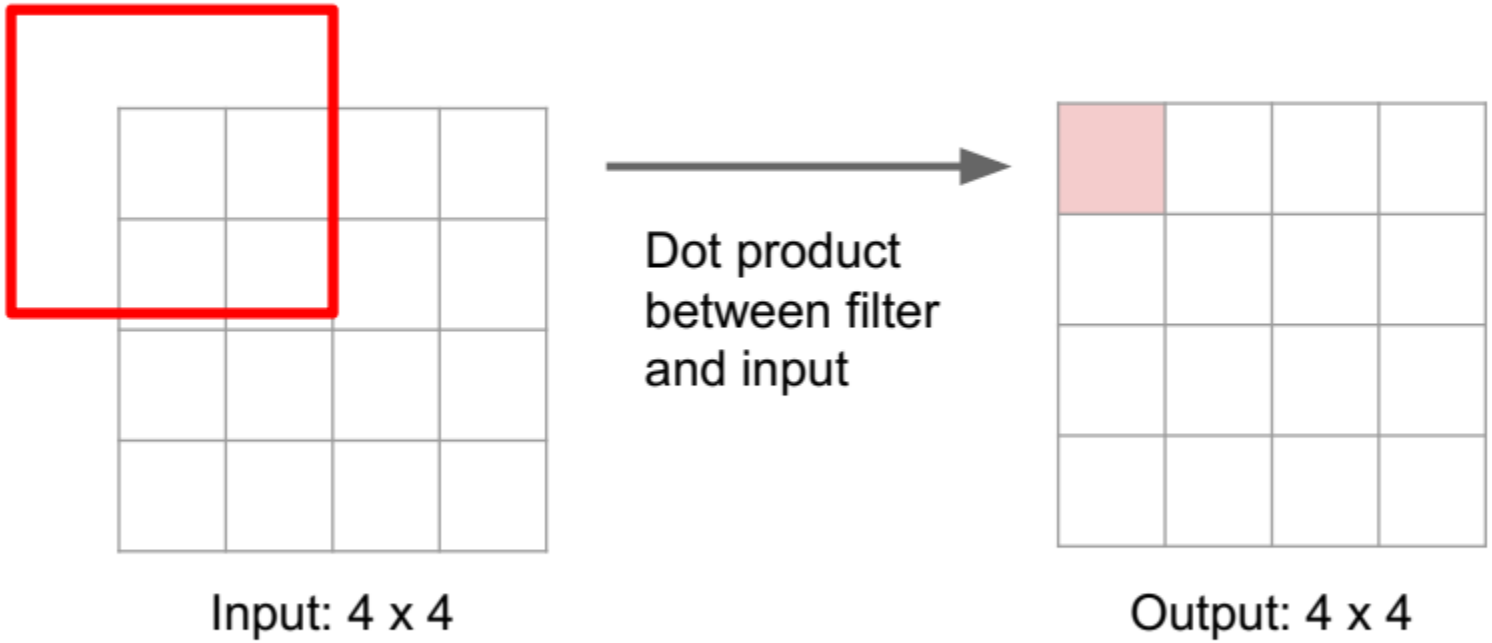
Input: 4 x 4



Output: 4 x 4

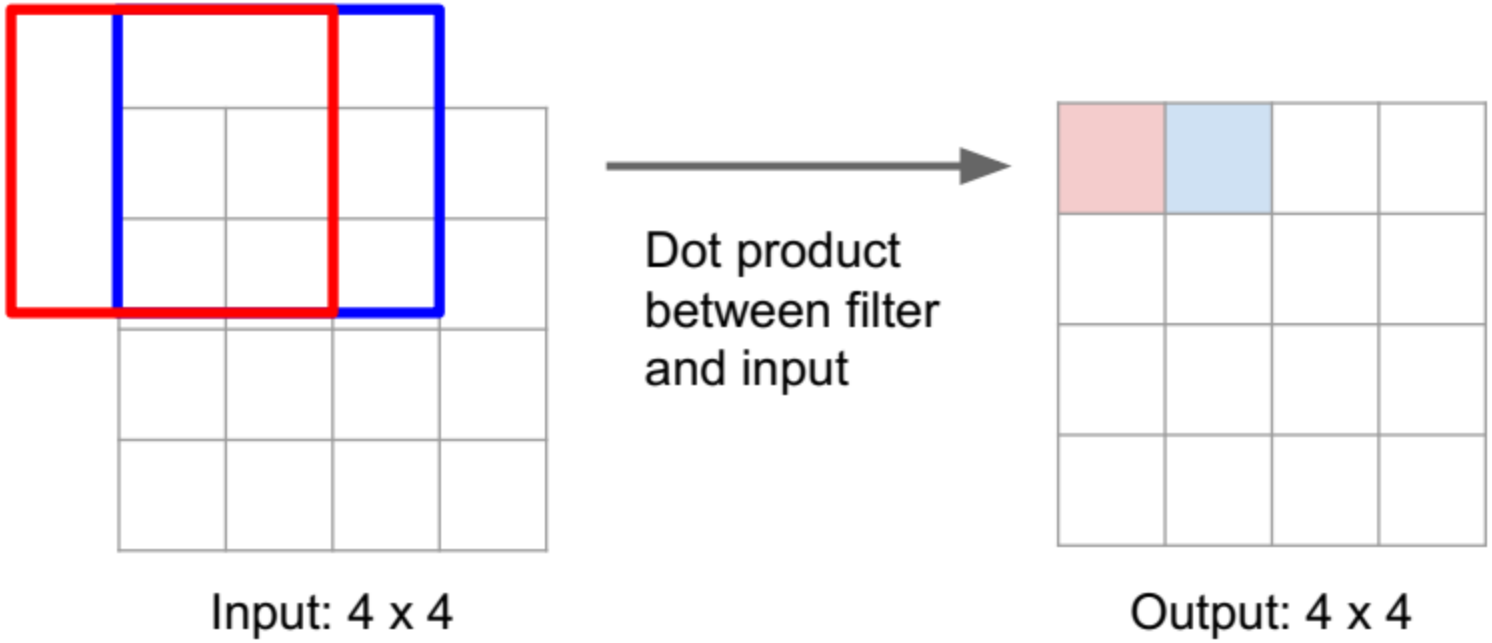
Learnable upsampling

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



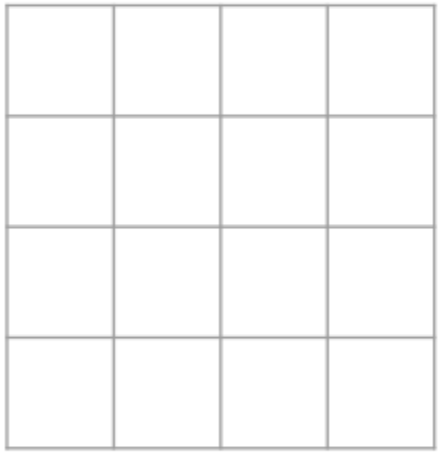
Learnable upsampling

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

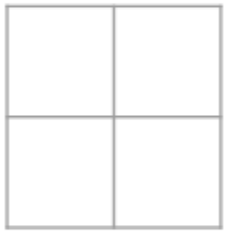


Learnable upsampling

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



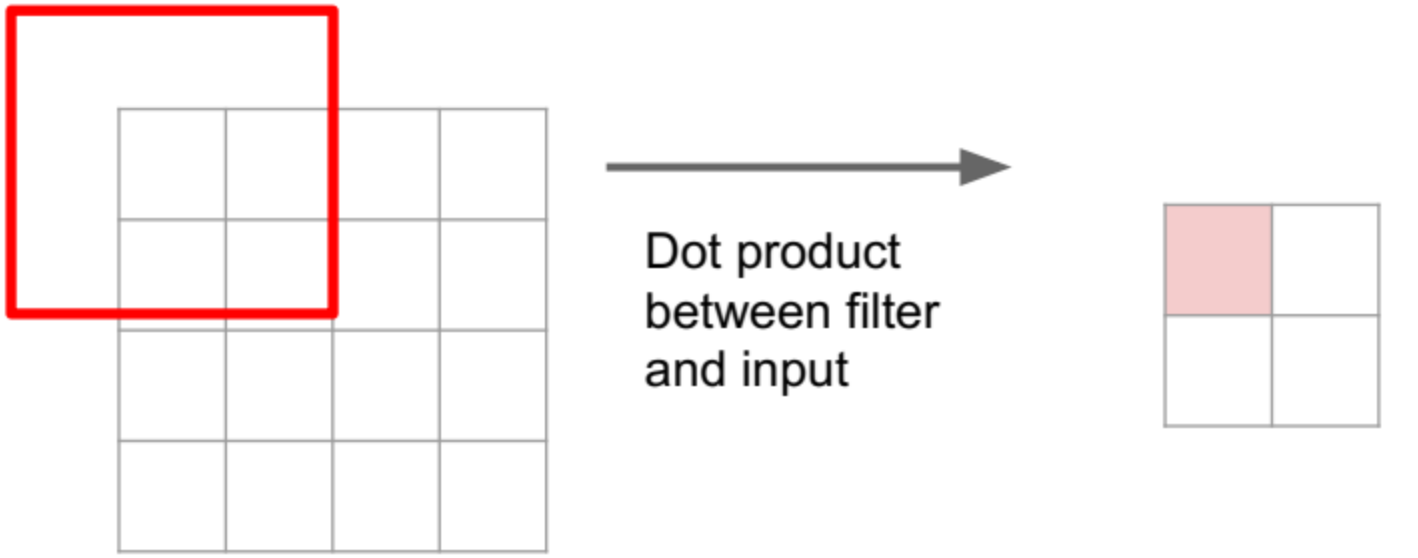
Input: 4 x 4



Output: 2 x 2

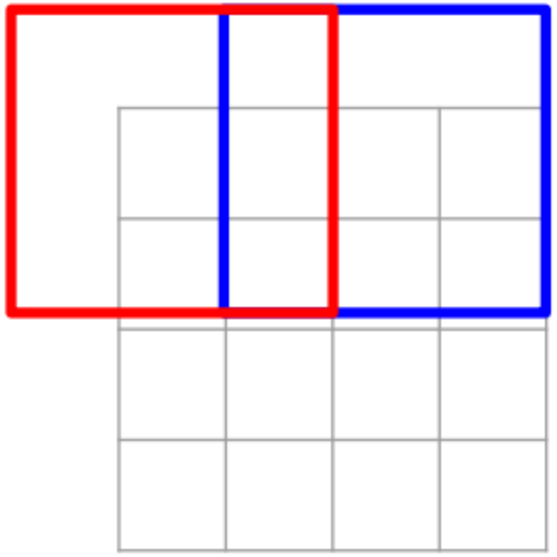
Learnable upsampling

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



Learnable upsampling

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



Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

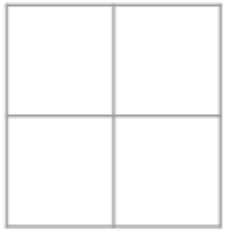
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

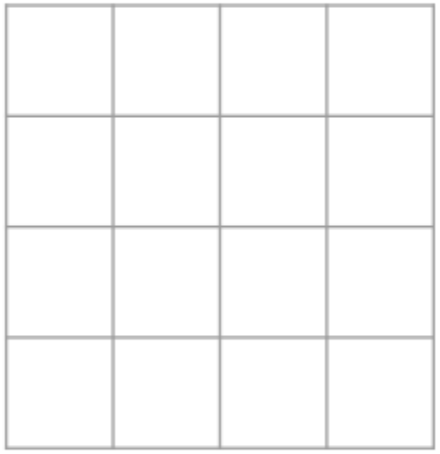
We can interpret strided convolution as “learnable downsampling”.

Learnable upsampling: *Transposed convolution*

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

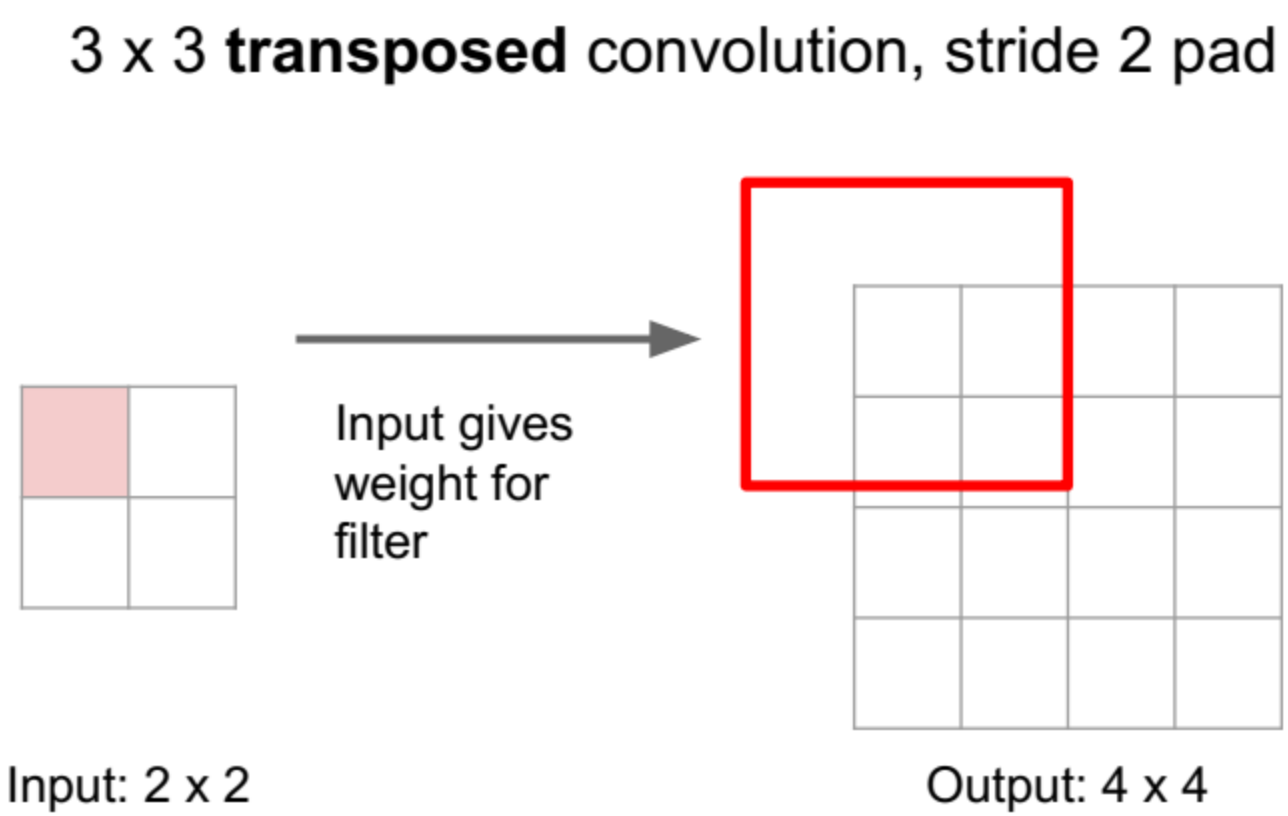


Input: 2 x 2

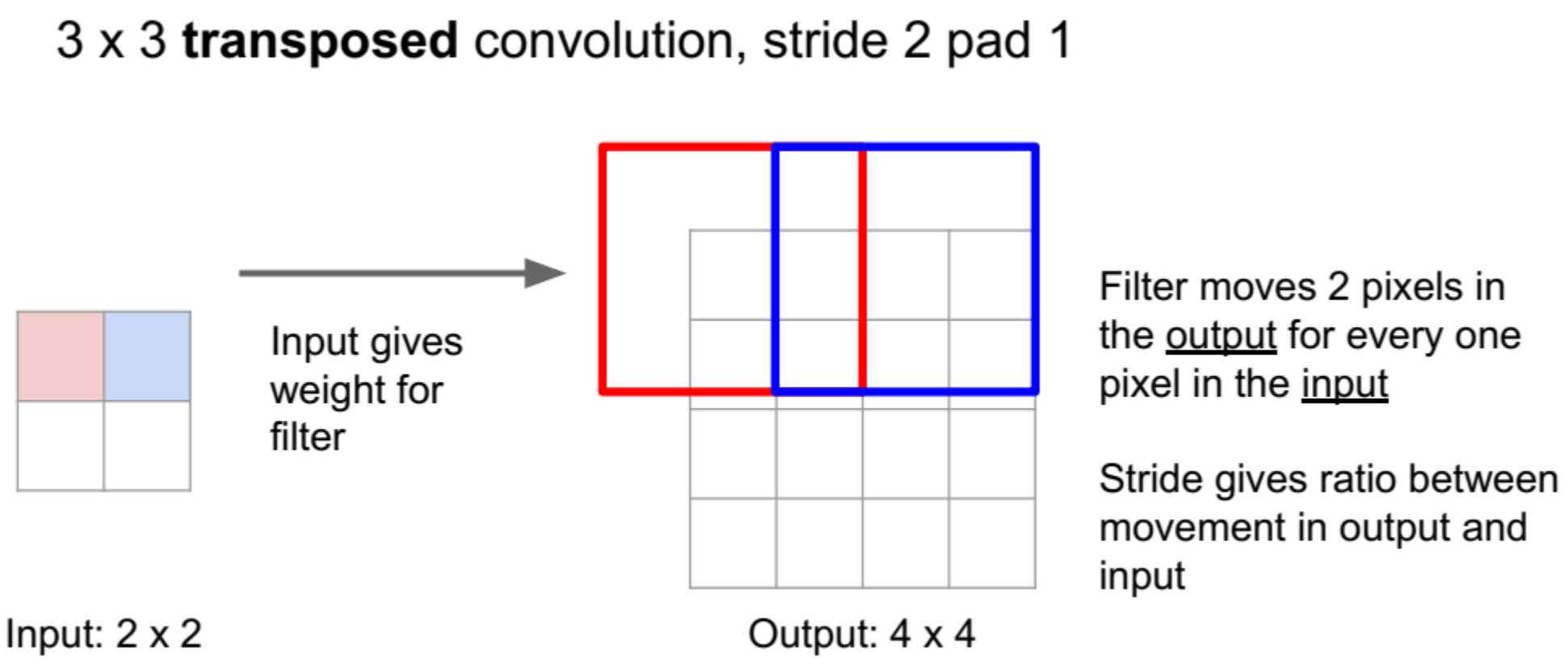


Output: 4 x 4

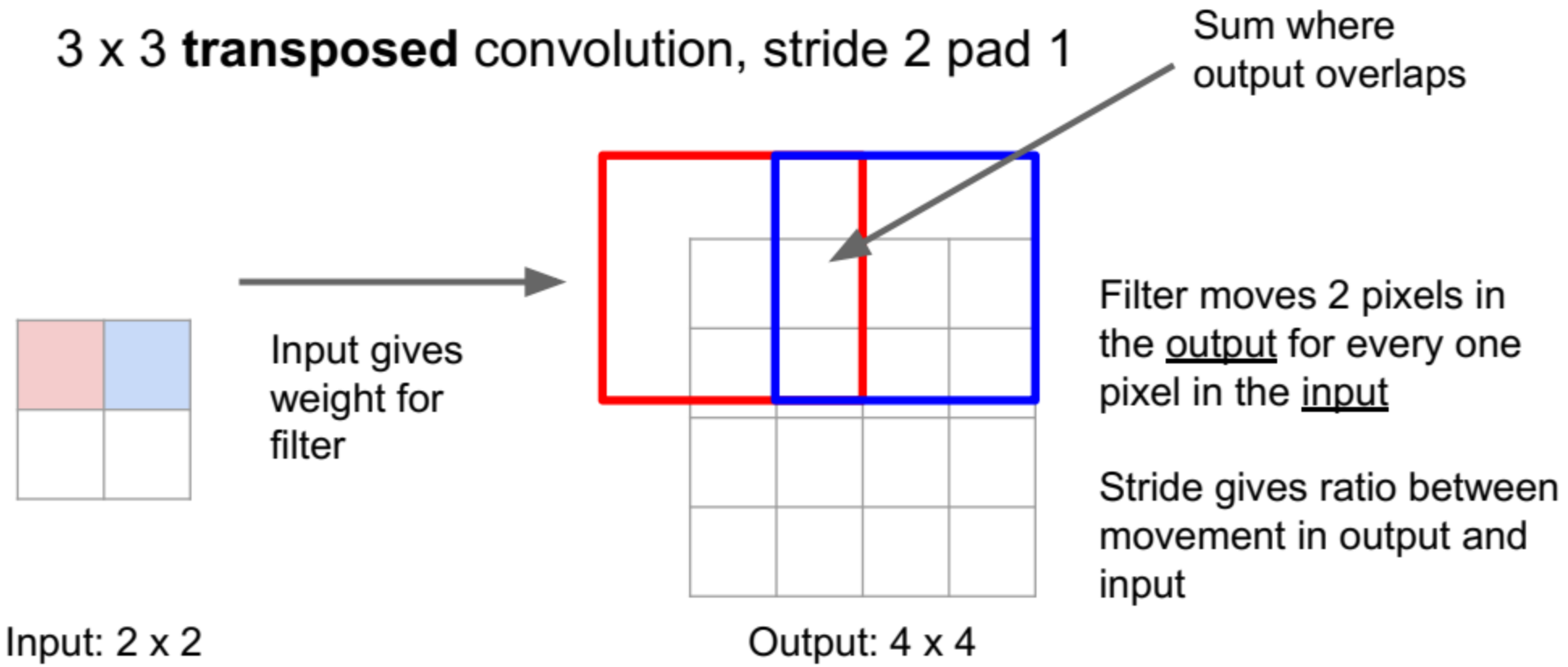
Learnable upsampling: *Transposed convolution*



Learnable upsampling: *Transposed convolution*

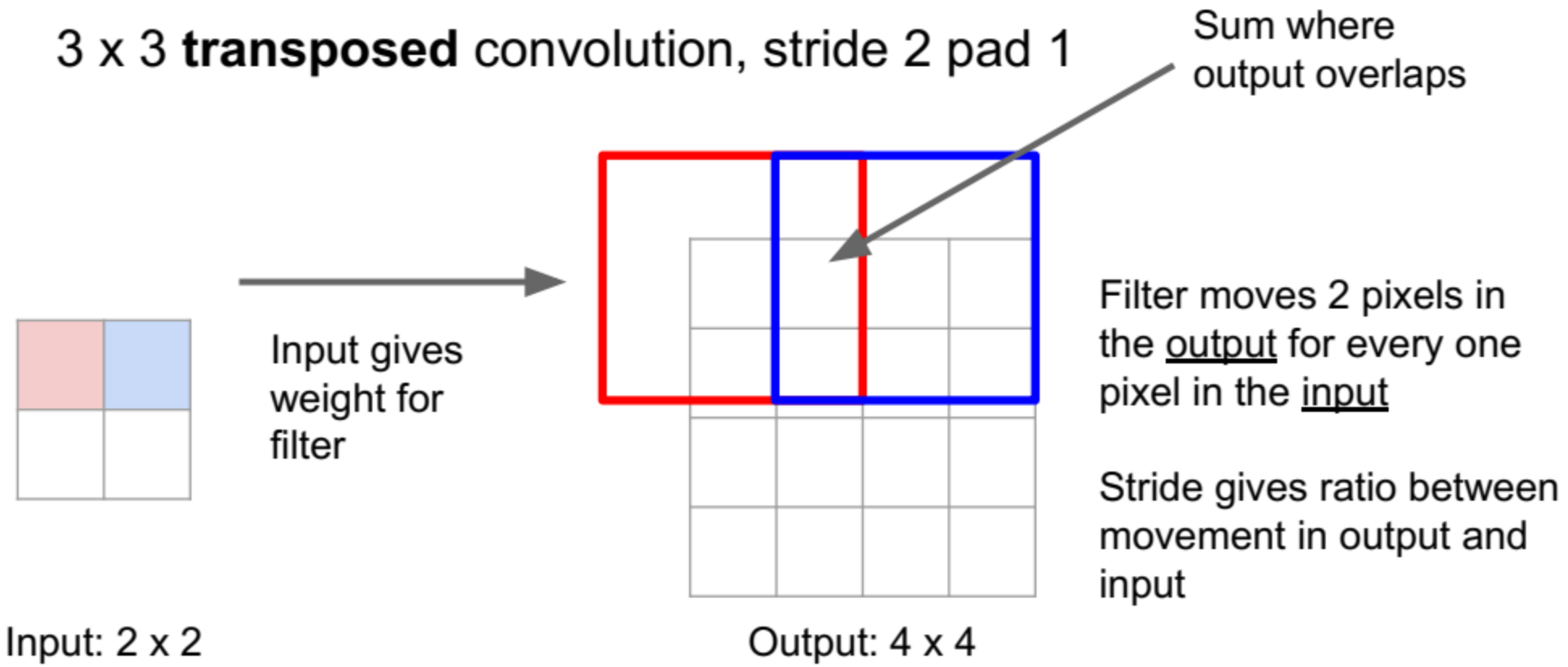


Learnable upsampling: *Transposed convolution*

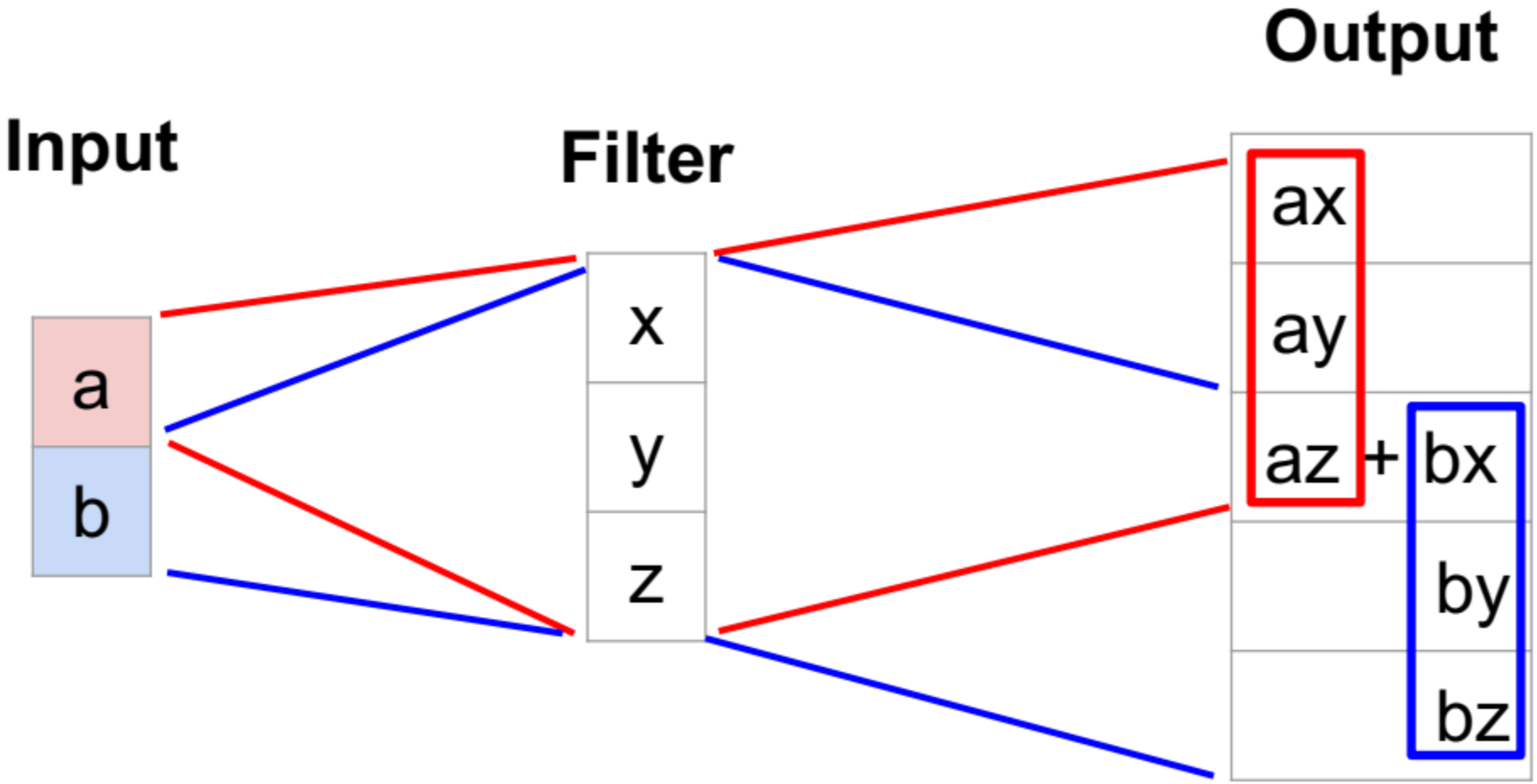


Learnable upsampling: *Transposed convolution*

Q: Why is it called transposed convolution?



Learnable upsampling: *1D example*



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

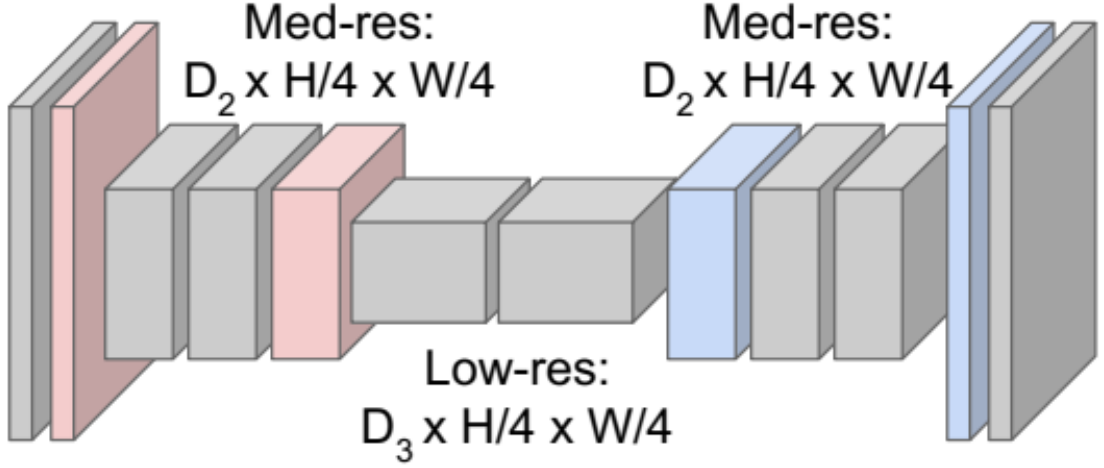
Semantic segmentation idea: Fully convolutional

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transposed convolution

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

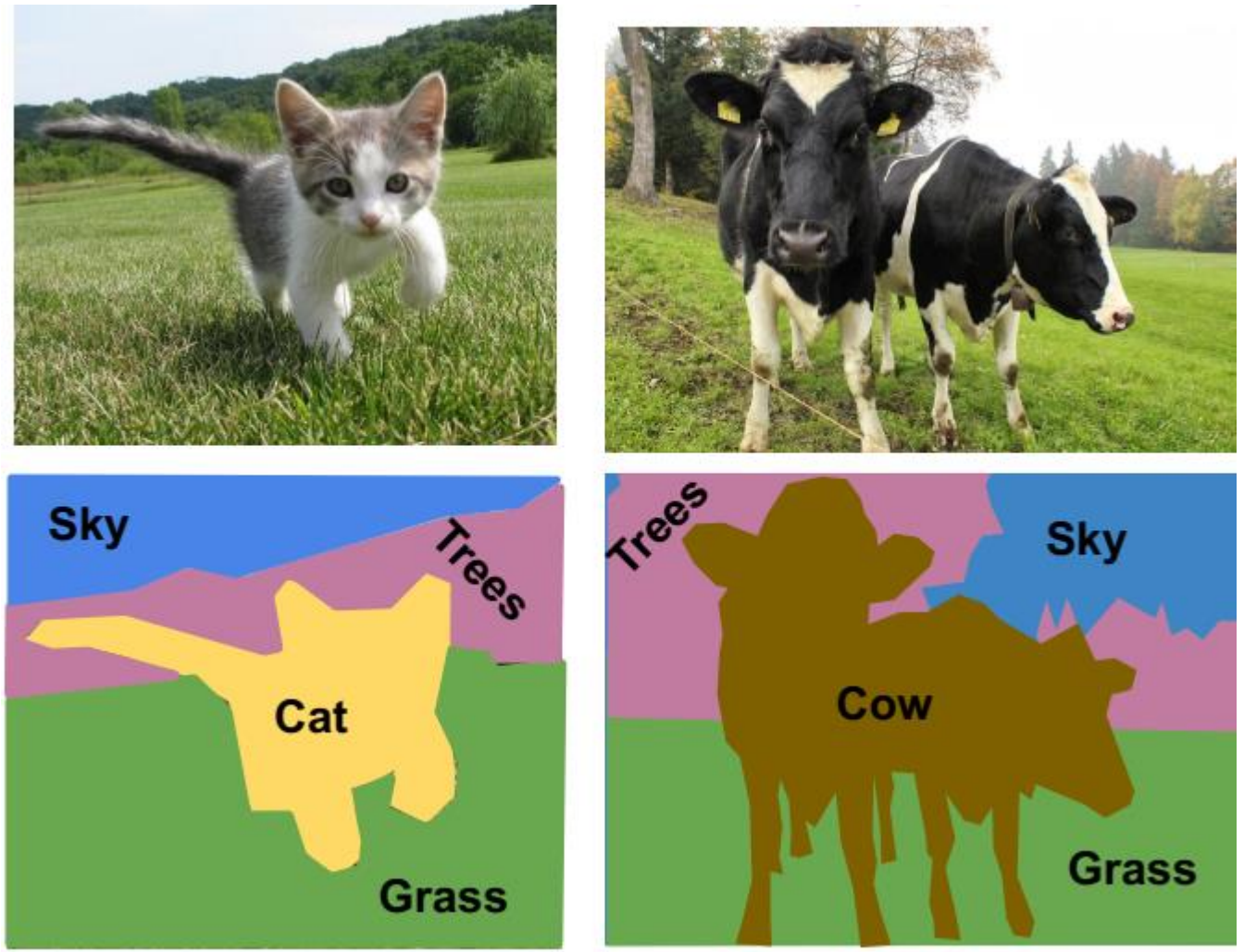
C x H x W

Predictions:
H x W

Semantic segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels



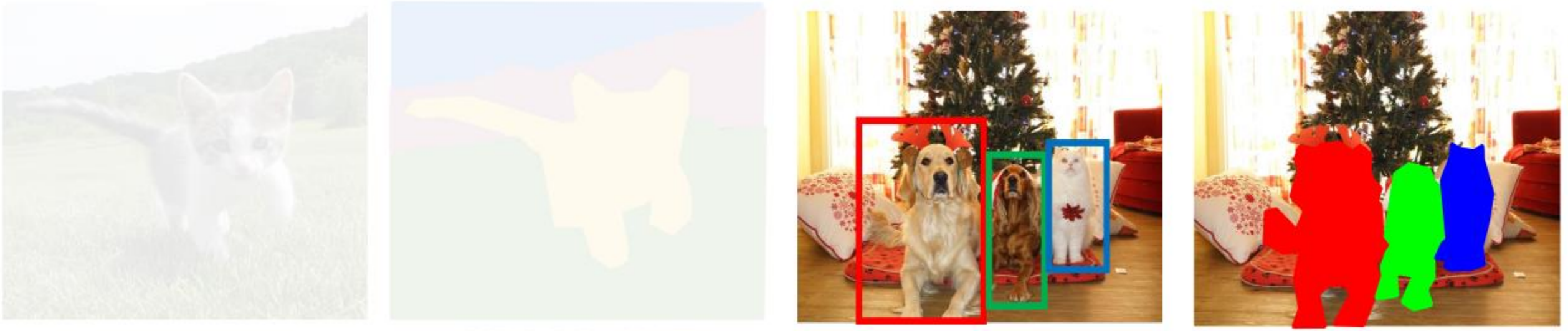
Object detection

Classification

Semantic Segmentation

Object Detection

Instance Segmentation



CAT

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

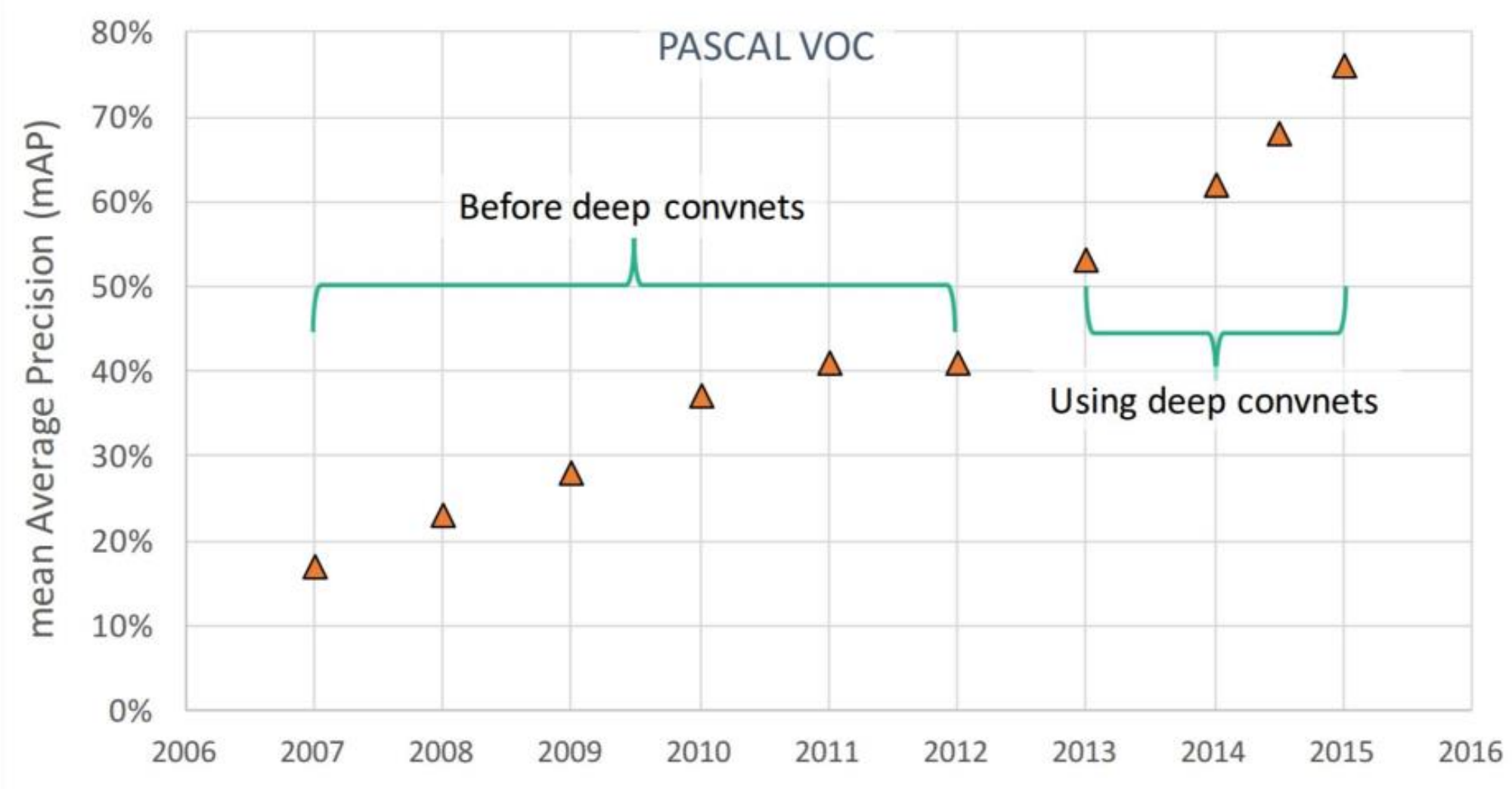
DOG, DOG, CAT

No spatial extent

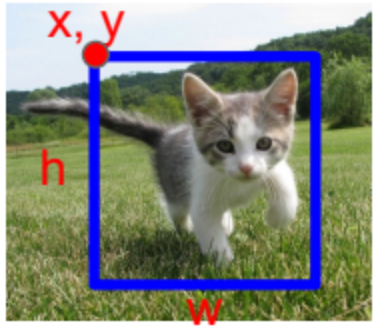
No objects, just pixels

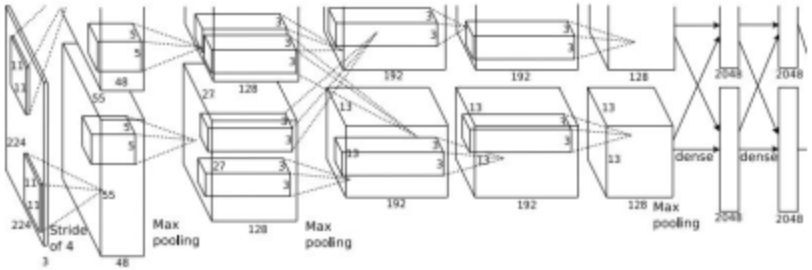
Multiple Object

Object detection



Object detection: *Single object*
(classification + localization)





Vector: 4096

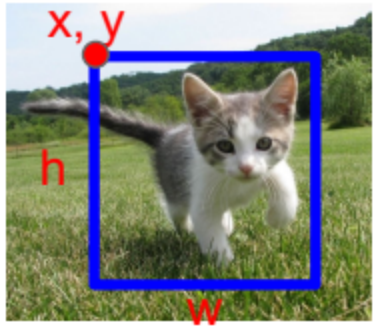
Fully Connected: 4096 to 1000

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

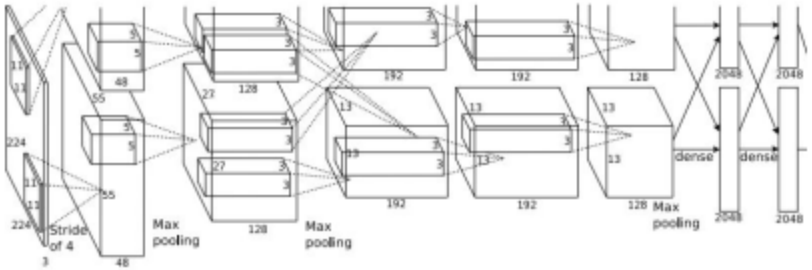
Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

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
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

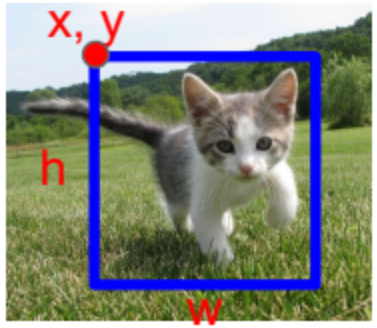
Box Coordinates (x, y, w, h)

L2 Loss

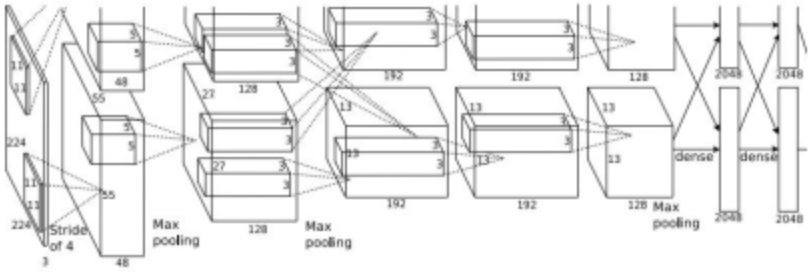
Correct box: (x', y', w', h')

Treat localization as a regression problem!

Object detection: *Single object*
(classification + localization)



This image is CC0 public domain



Correct label:
Cat

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

Fully Connected:
4096 to 1000

Softmax Loss

Multitask Loss

+ → **Loss**

Vector:
4096

Fully Connected:
4096 to 4

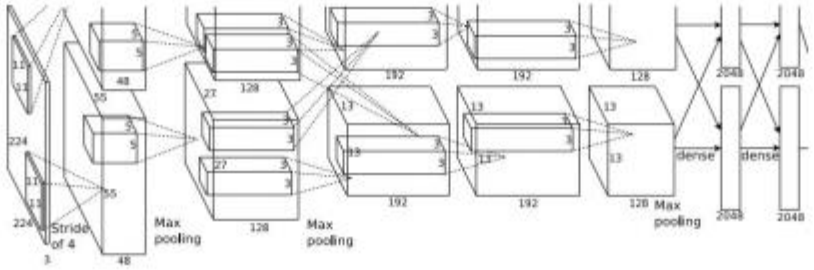
Box Coordinates
(x, y, w, h)

L2 Loss

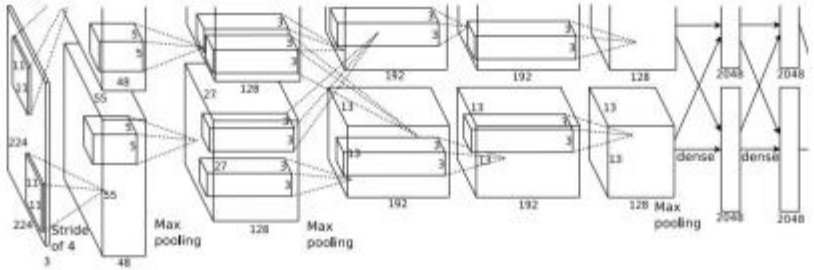
Correct box:
(x', y', w', h')

Treat localization as a regression problem!

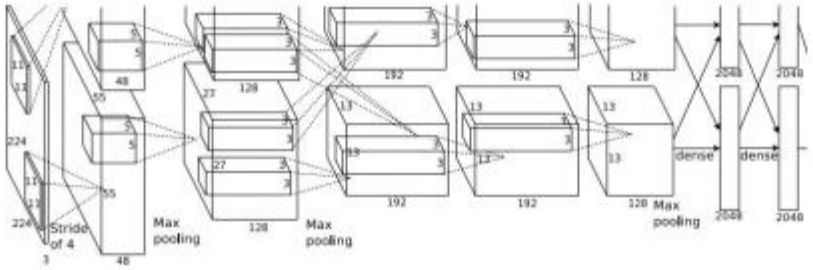
Object detection: *Multiple objects*



CAT: (x, y, w, h)

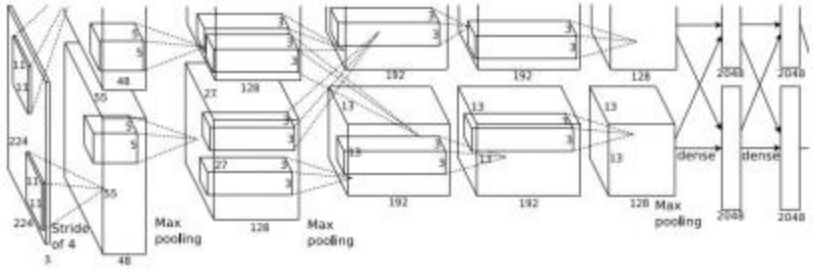


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



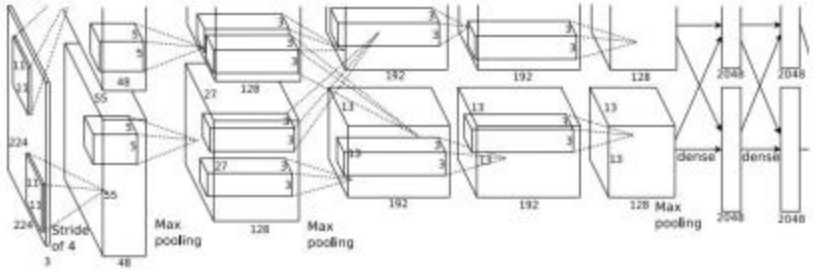
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Object detection: *Multiple objects*



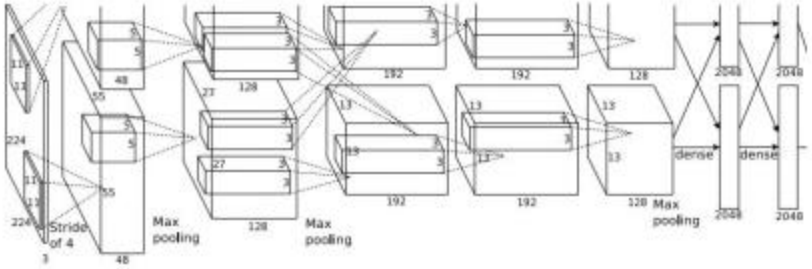
CAT: (x, y, w, h)

4 numbers



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

12 numbers



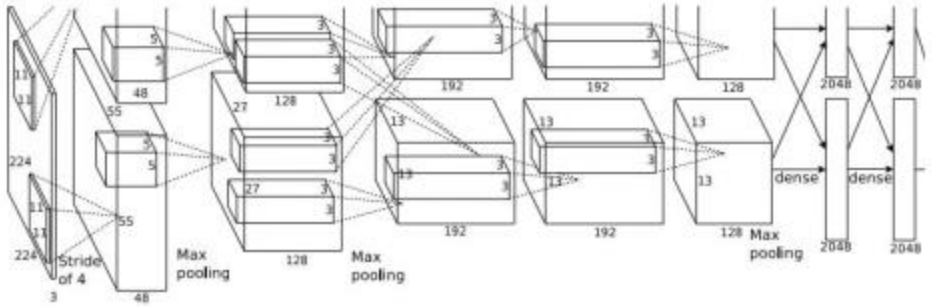
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Each image needs a different number of outputs!

Object detection: *Multiple objects*

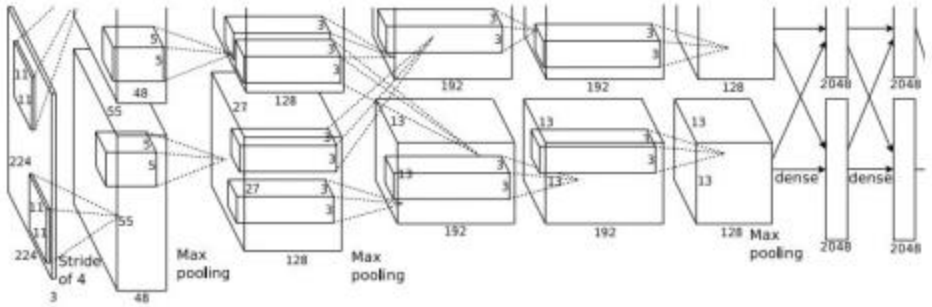
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object detection: *Multiple objects*

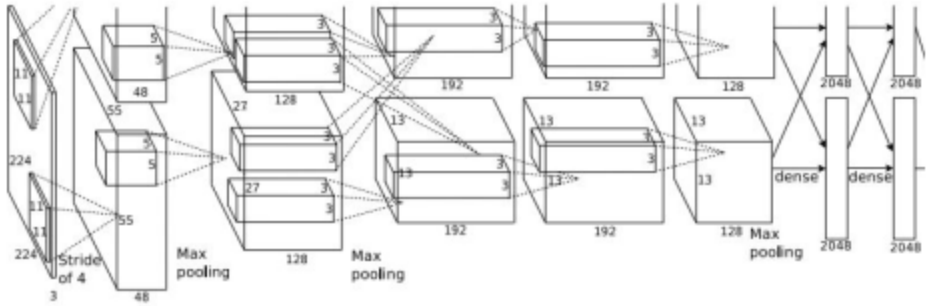
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object detection: *Multiple objects*

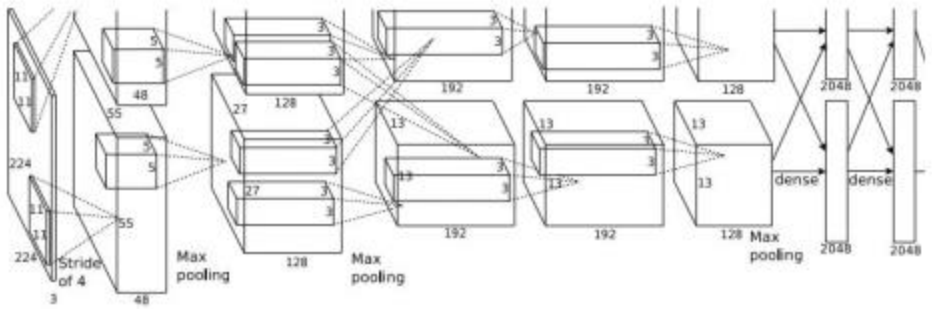
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object detection: *Multiple objects*

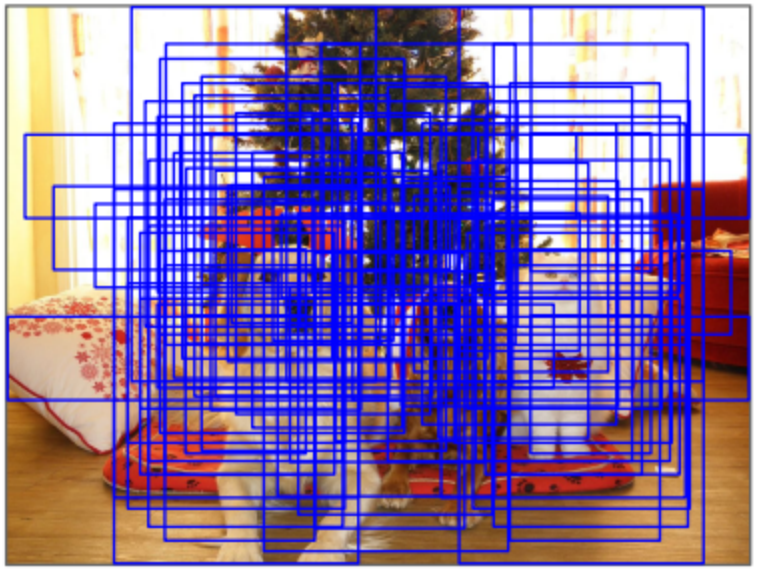
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



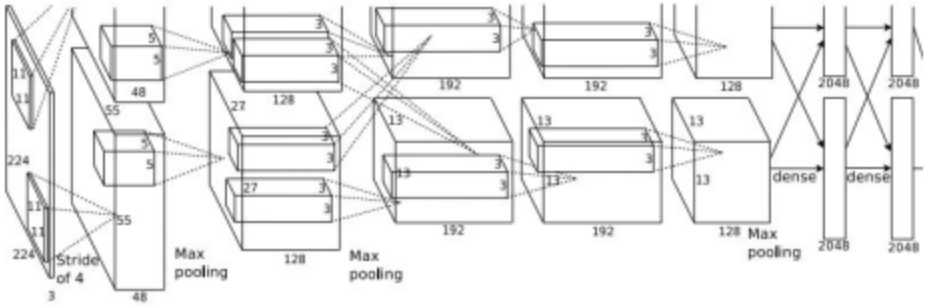
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Object detection: *Multiple objects*



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

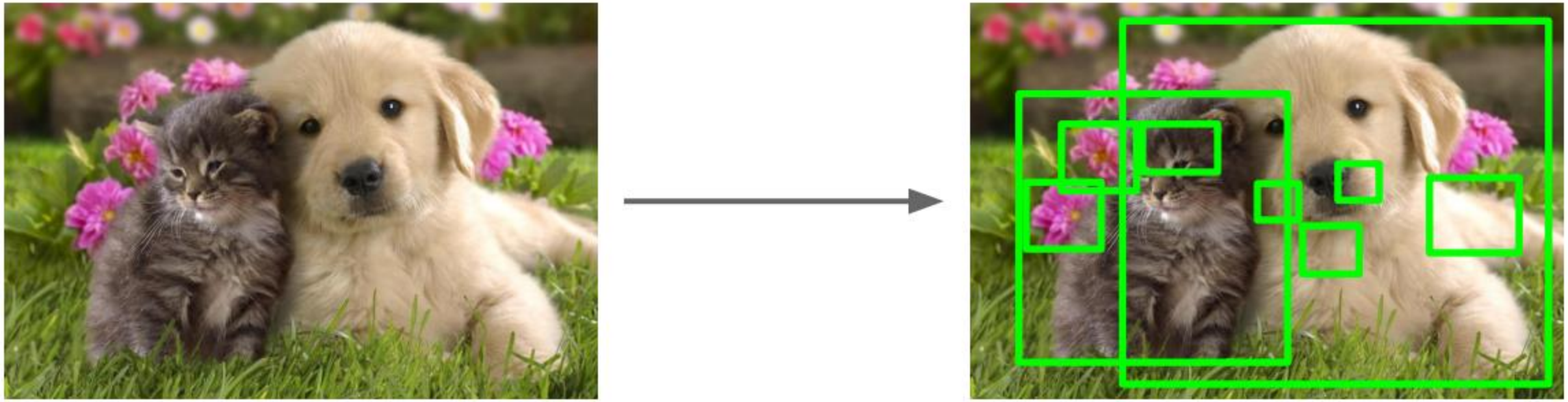


Dog? NO
Cat? YES
Background? NO

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

Region proposals: Selective search

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



Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

R-CNN

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.

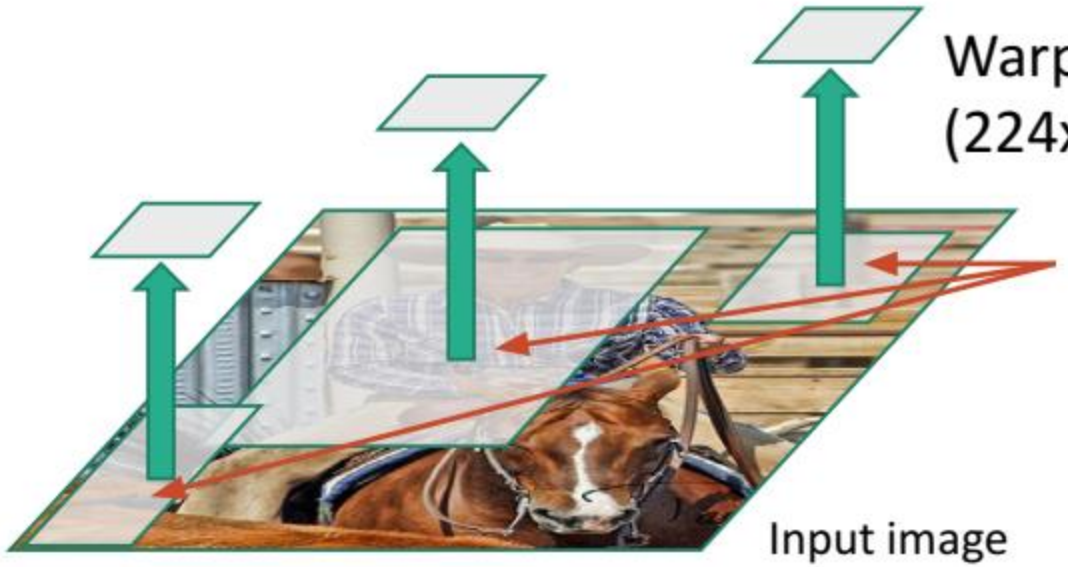
R-CNN

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



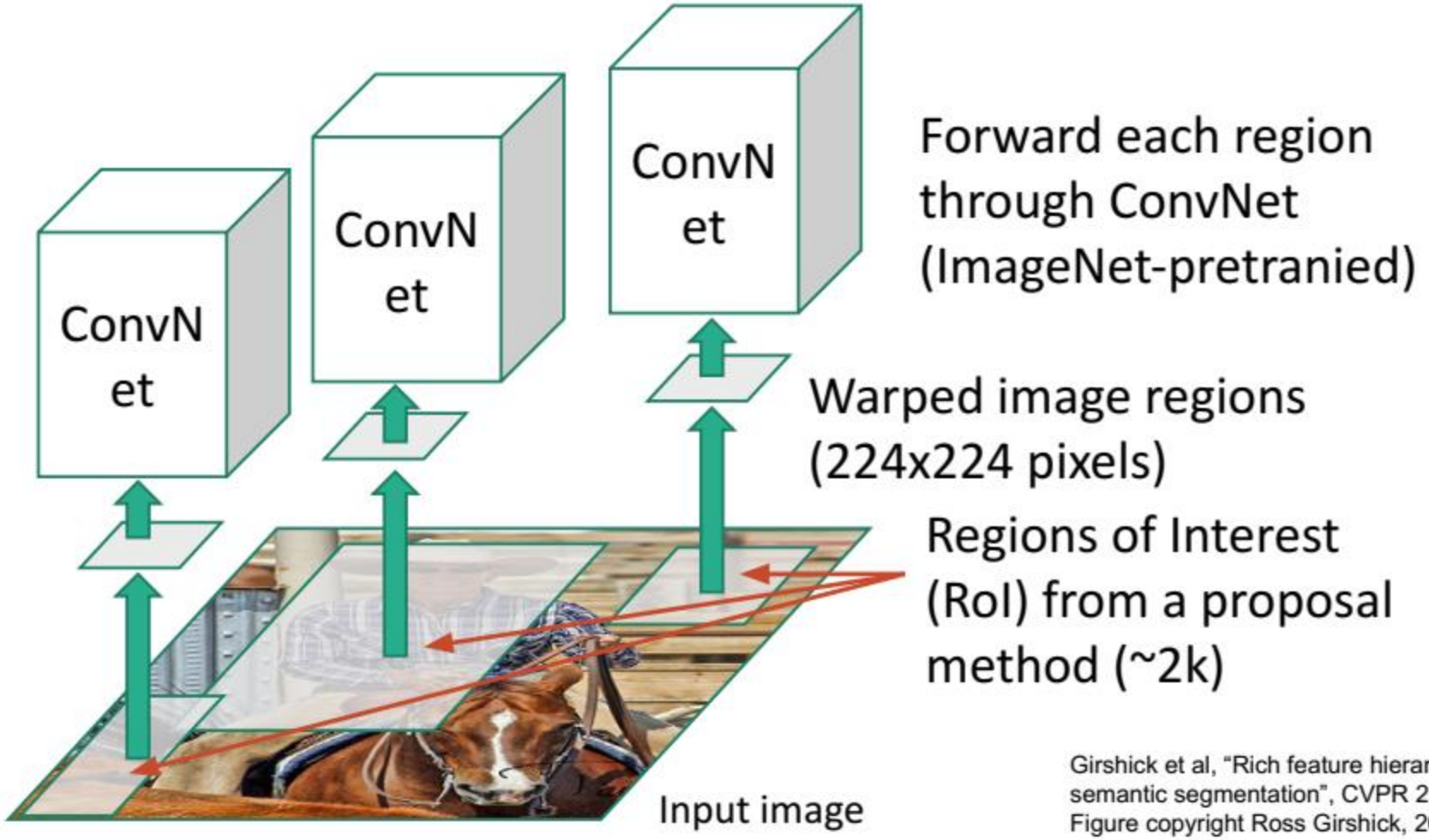
Warped image regions
(224x224 pixels)

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

Input image

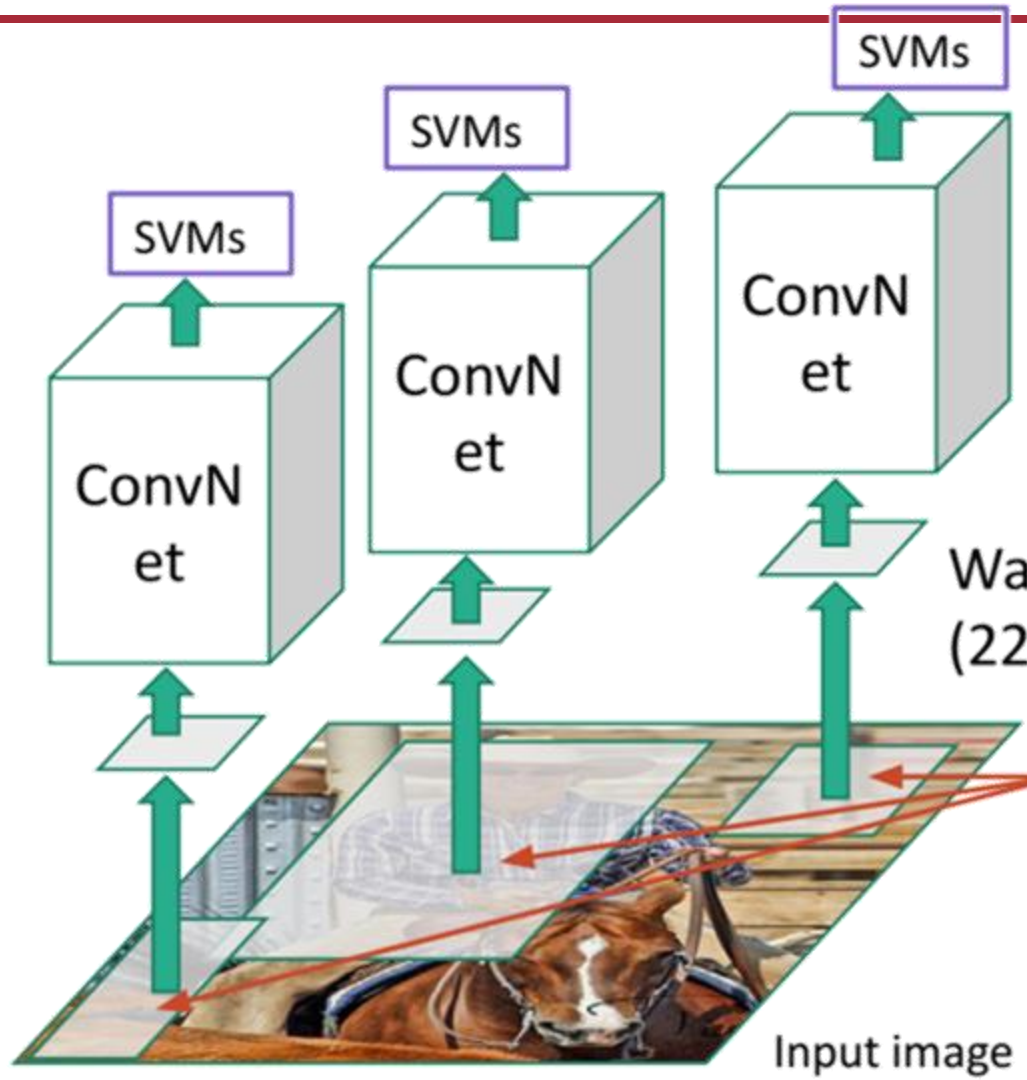
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



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

R-CNN



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

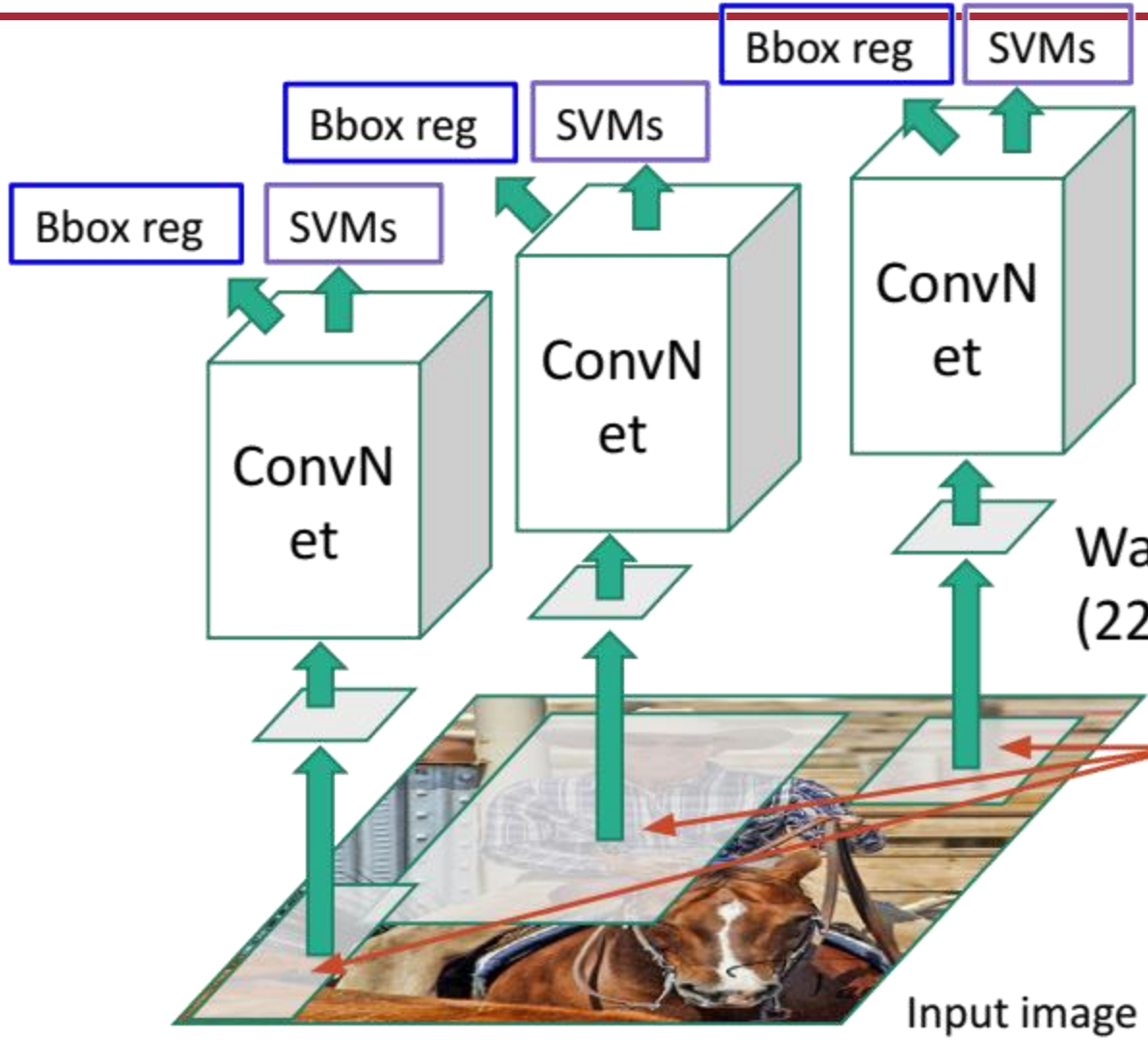
Warped image regions (224x224 pixels)

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

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

R-CNN

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



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

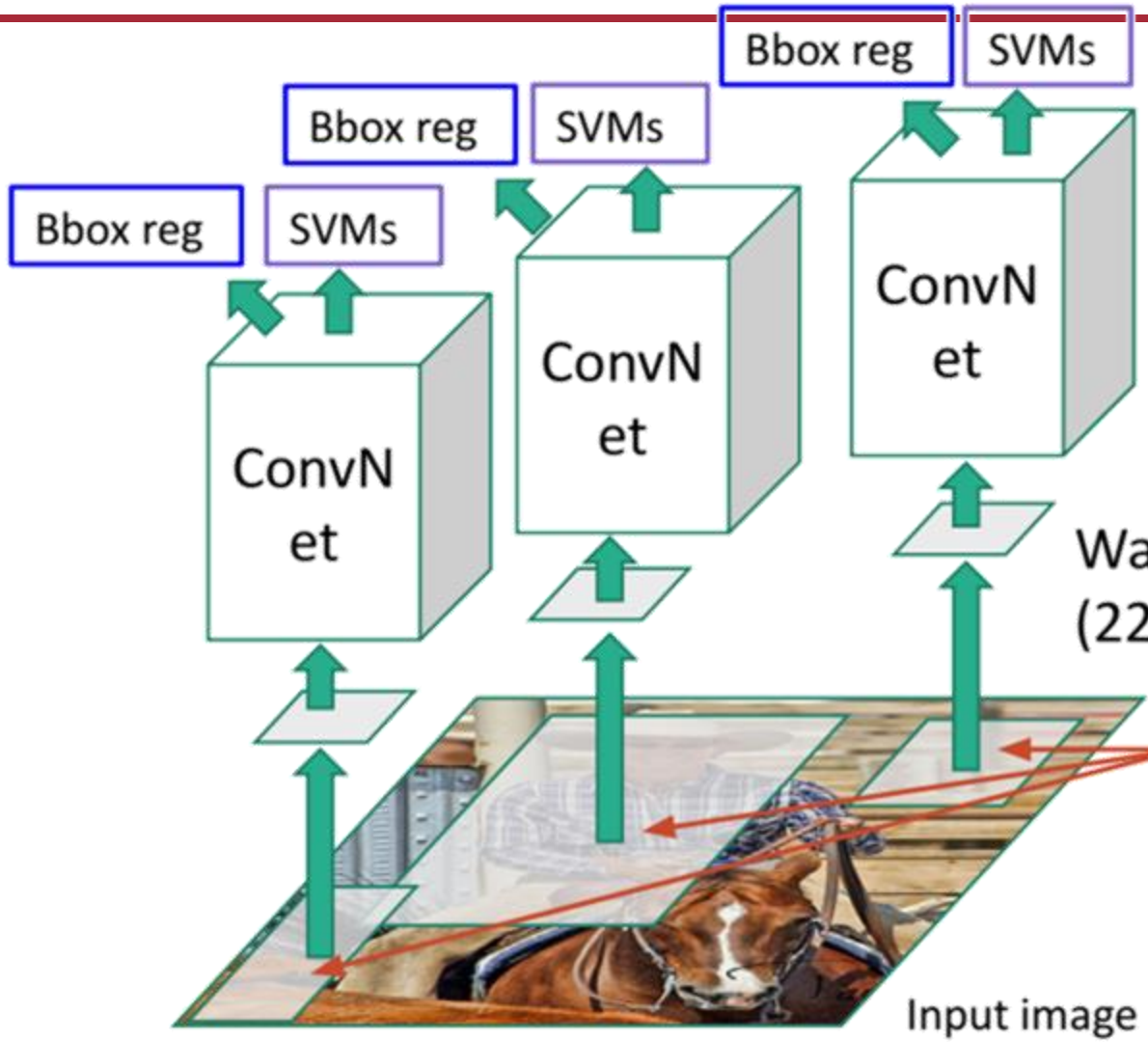
Warped image regions (224x224 pixels)

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

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

R-CNN

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



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

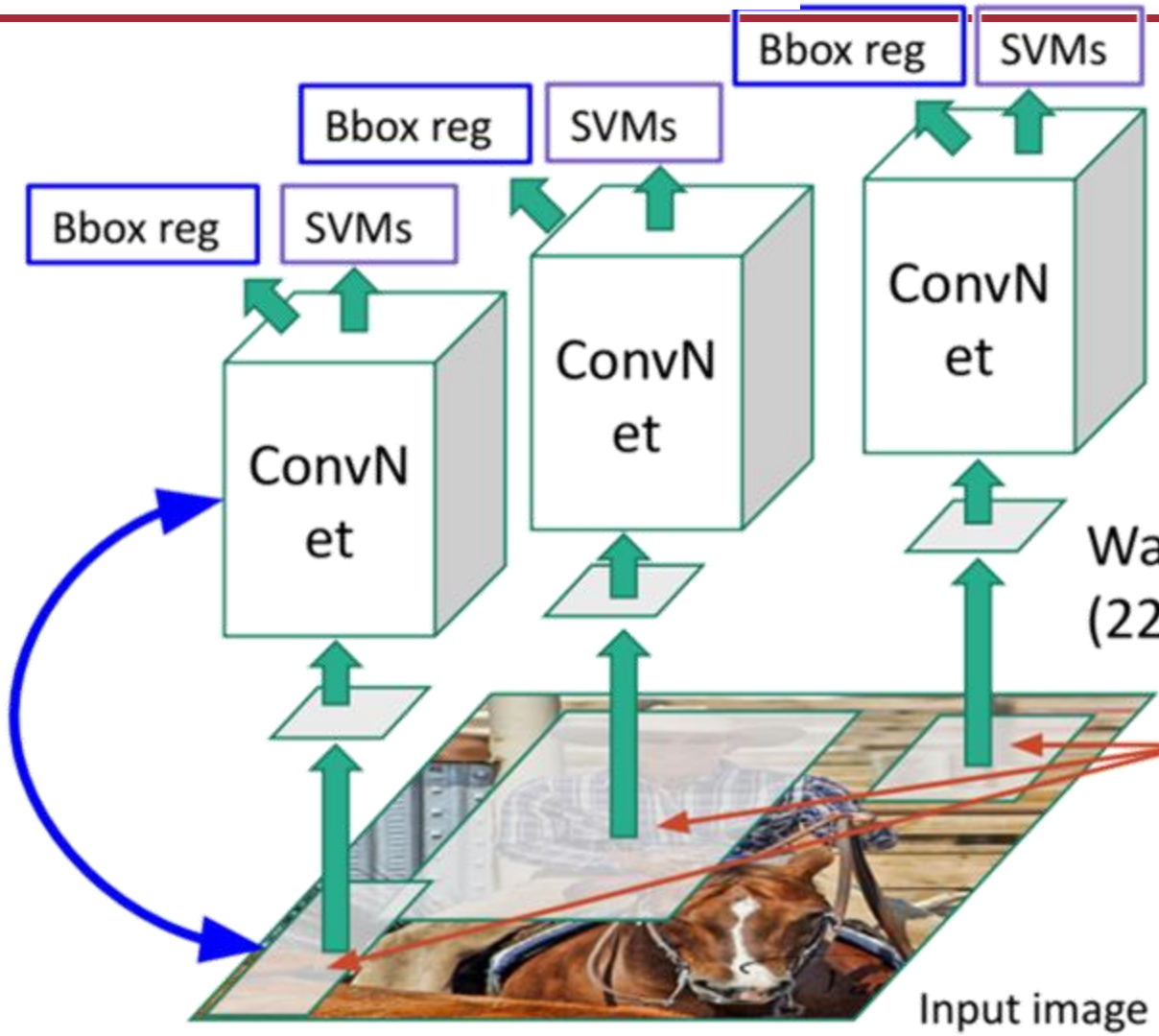
Regions of Interest (RoI) from a proposal method (~2k)

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

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

"Slow" R-CNN

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



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

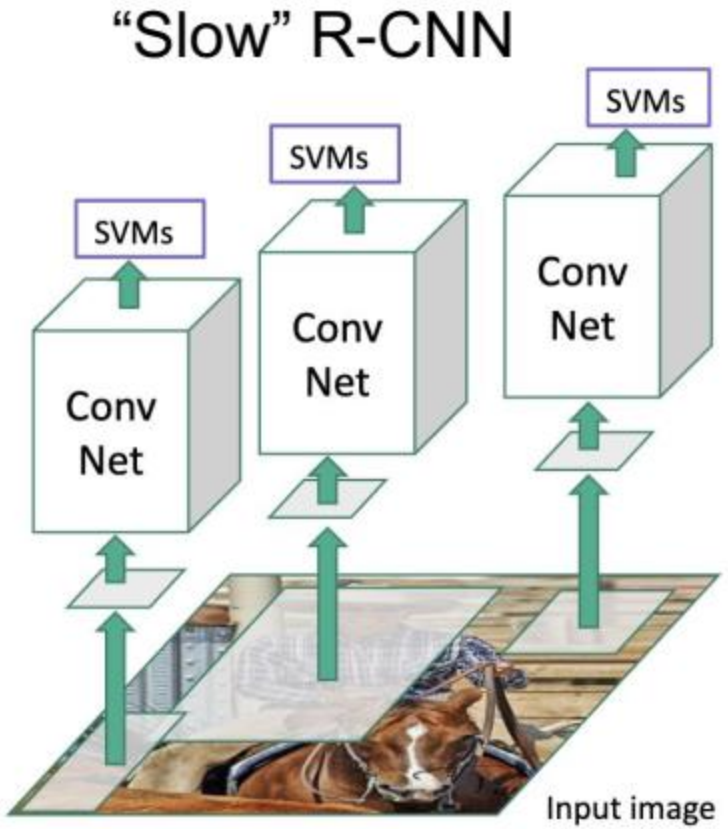
Problem: Very slow! Need to do ~2k independent forward passes for each image!

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

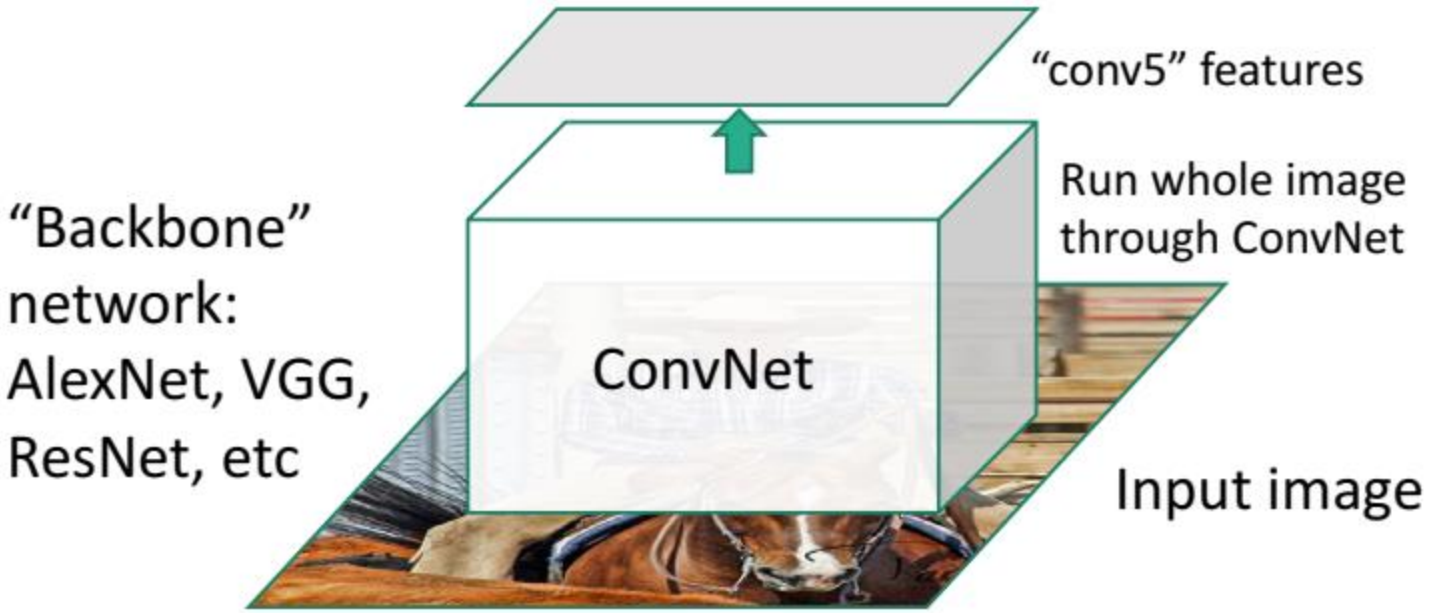
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

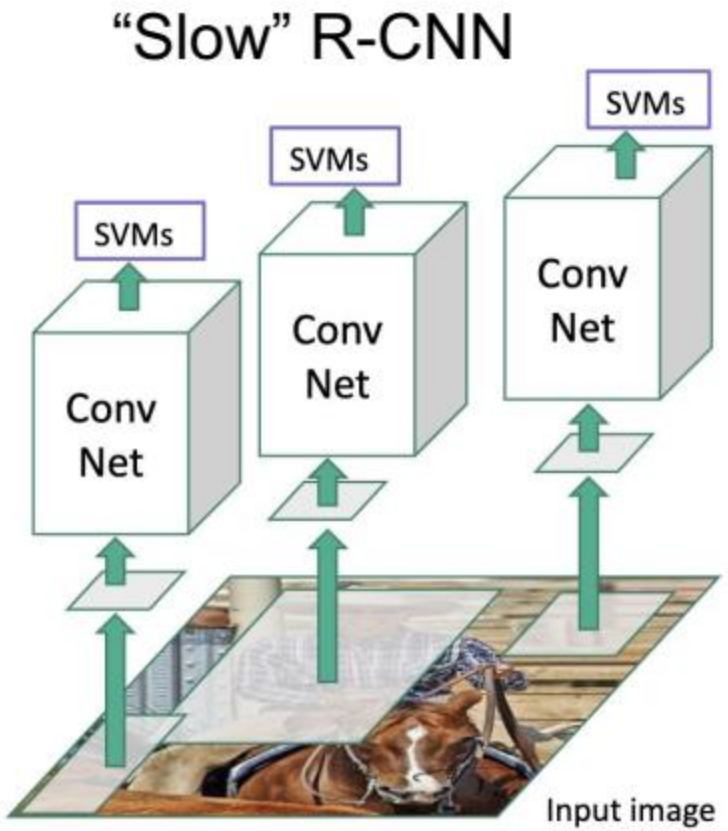
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



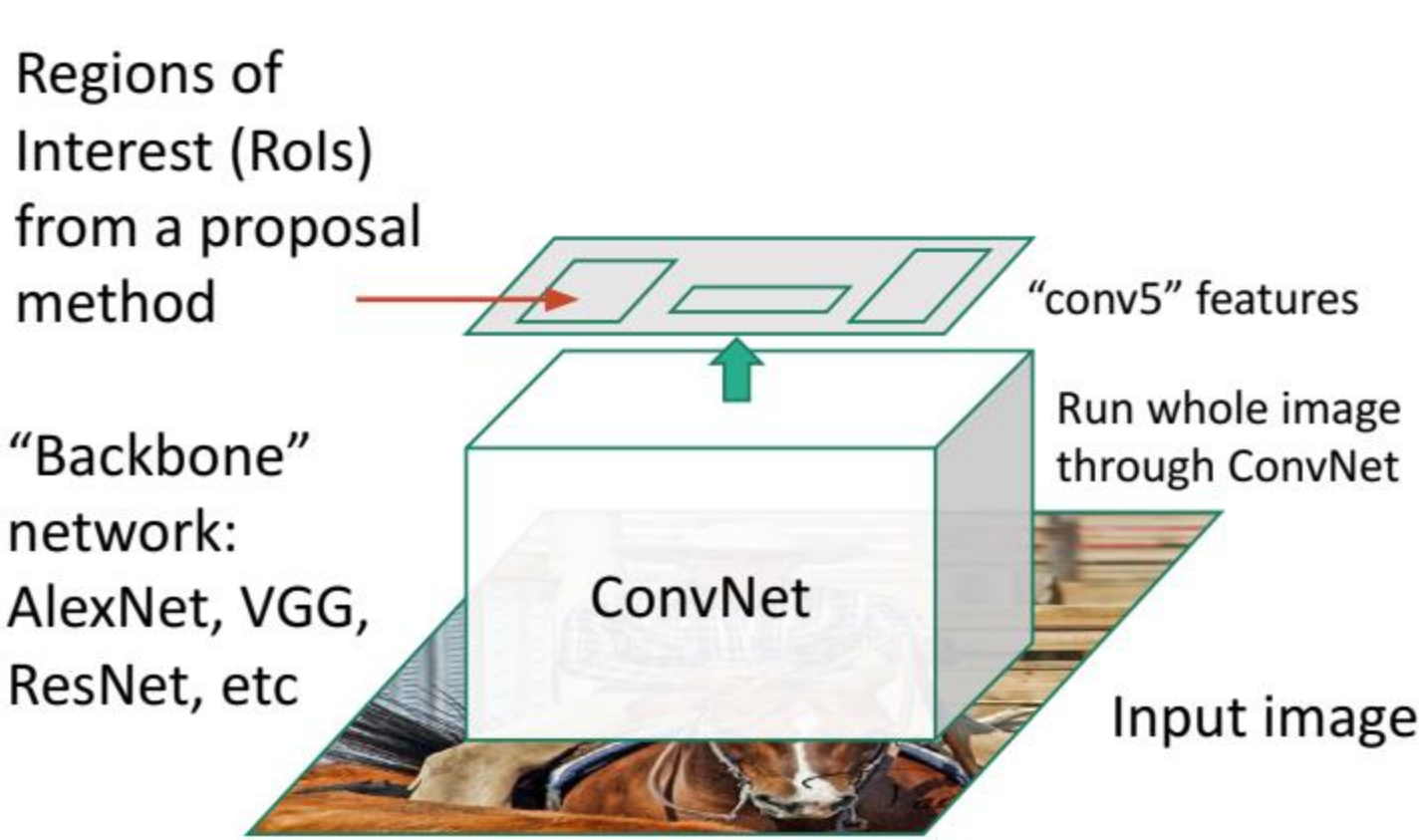
Fast R-CNN



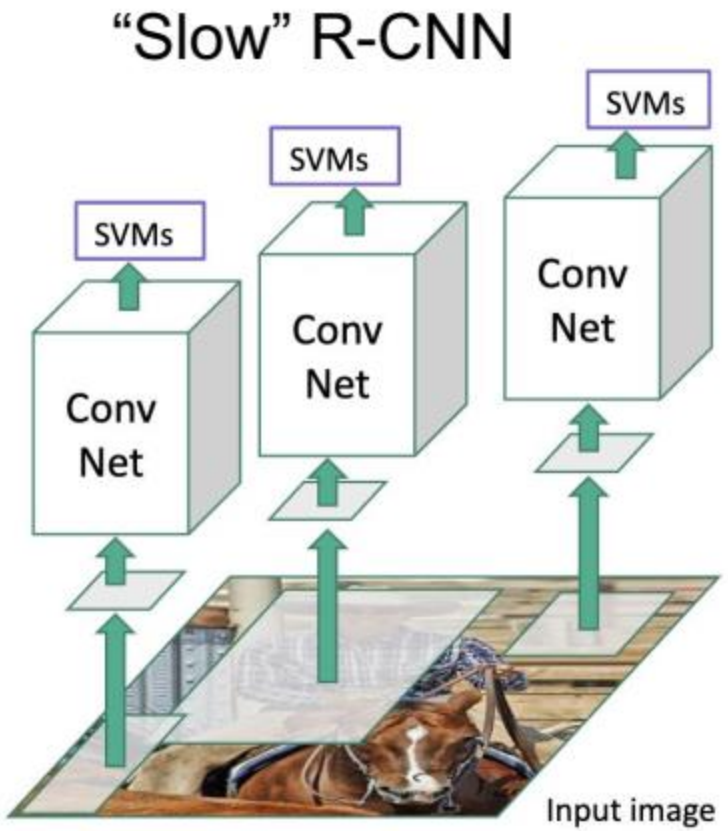
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



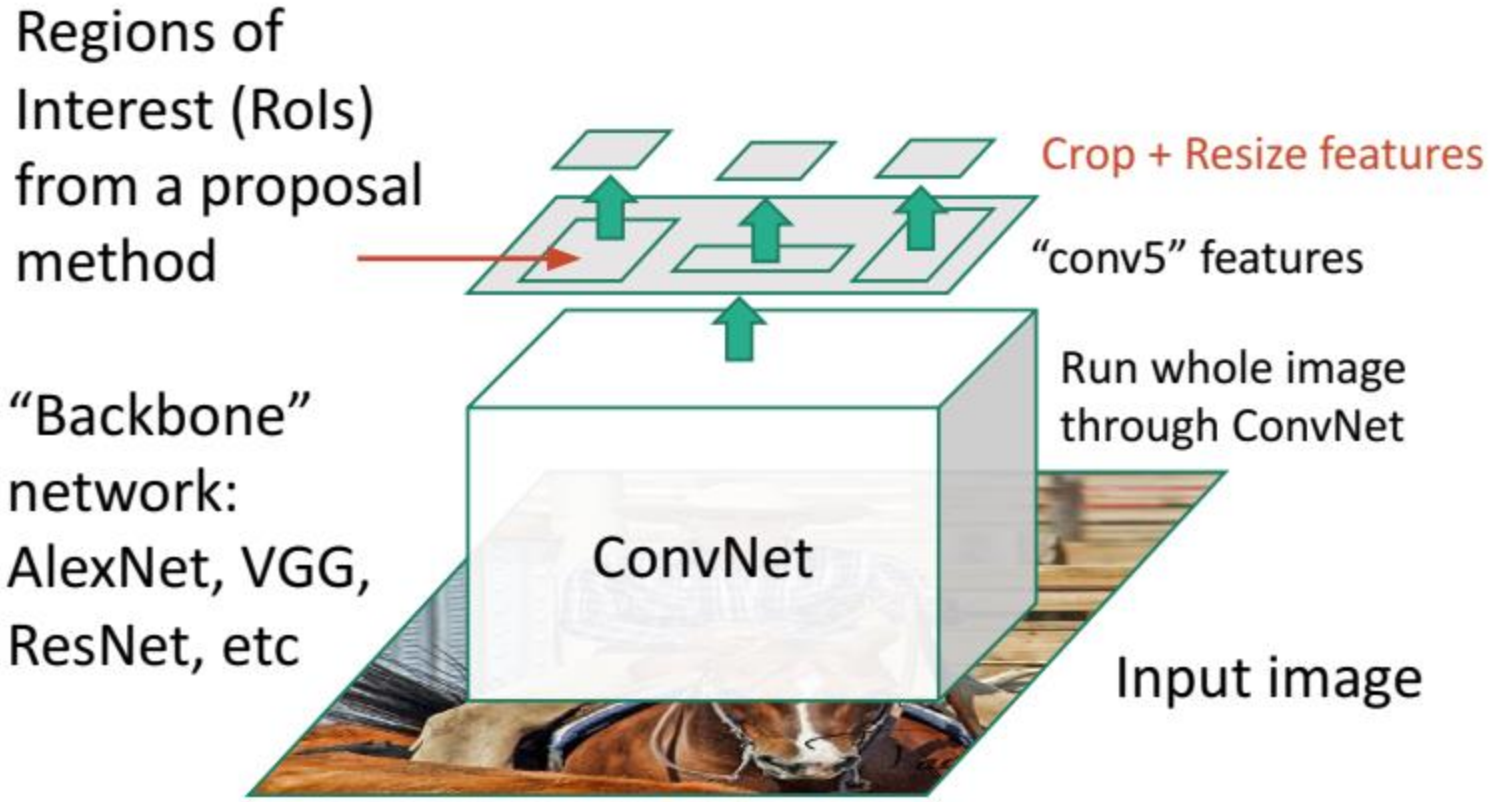
Fast R-CNN



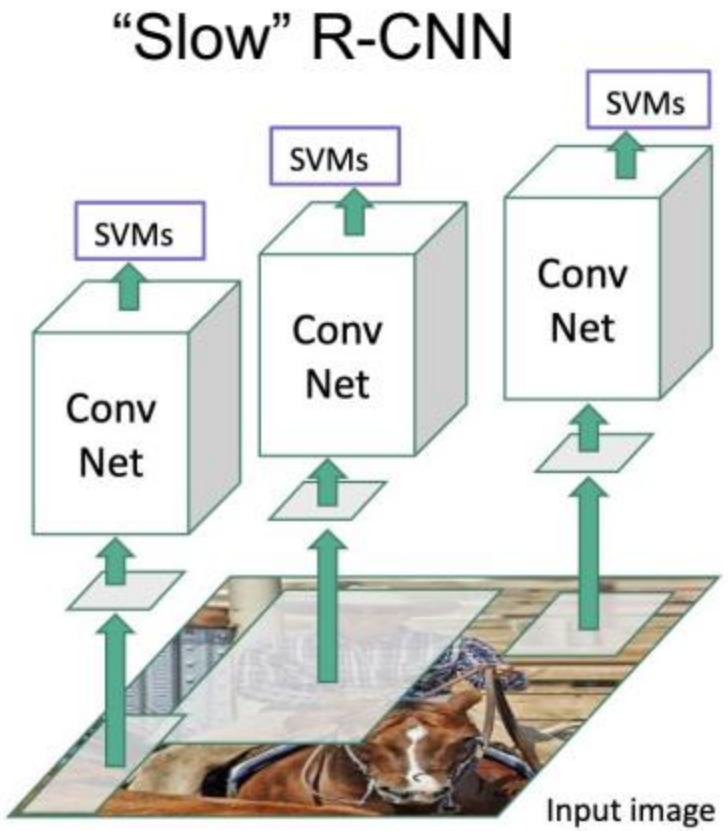
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



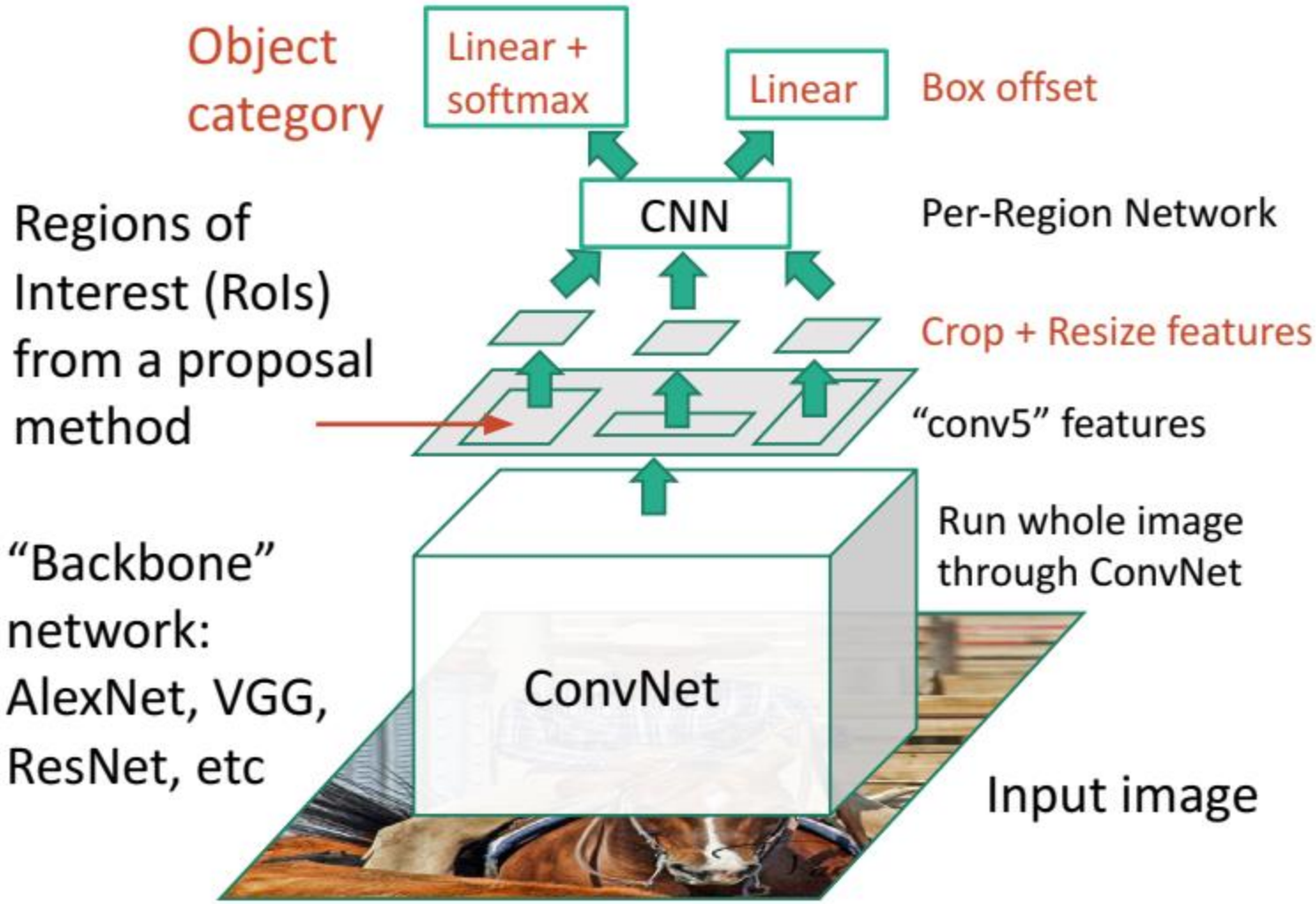
Fast R-CNN



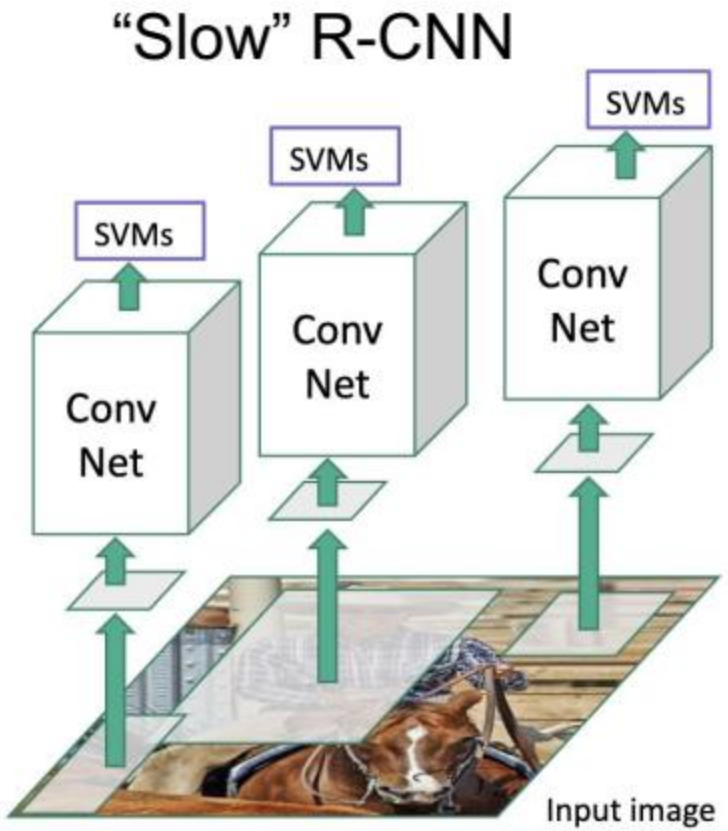
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



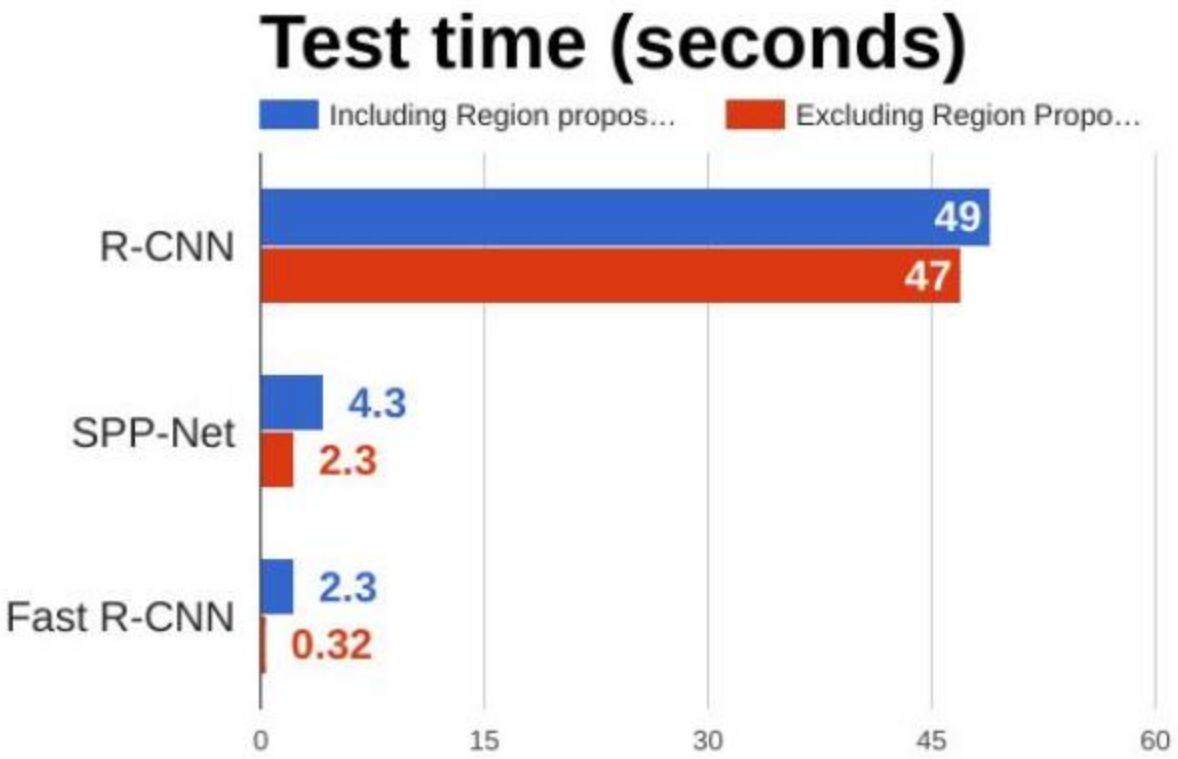
Fast R-CNN



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

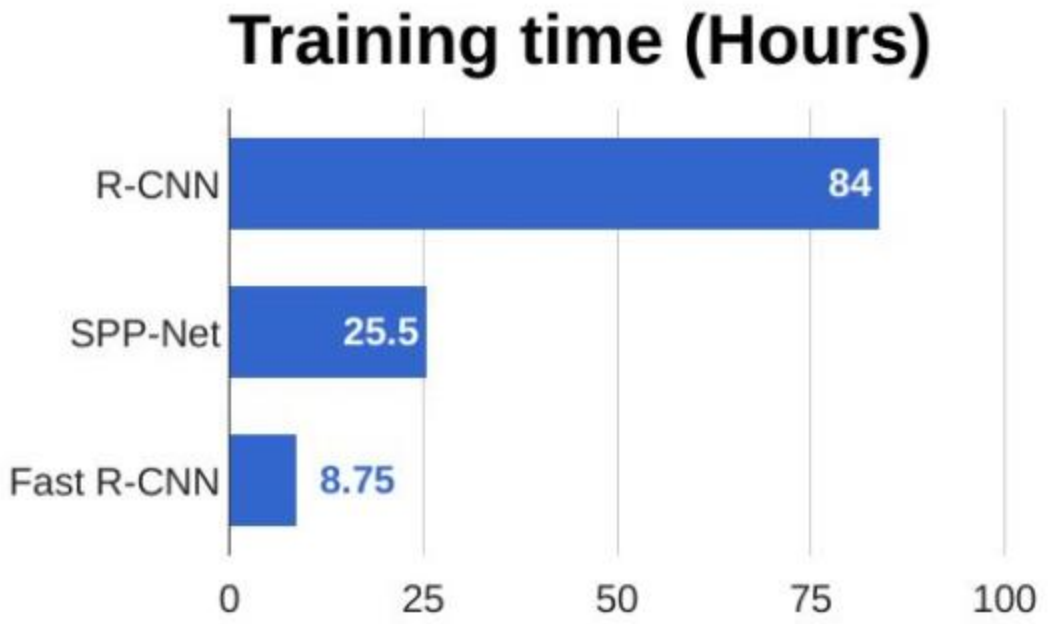


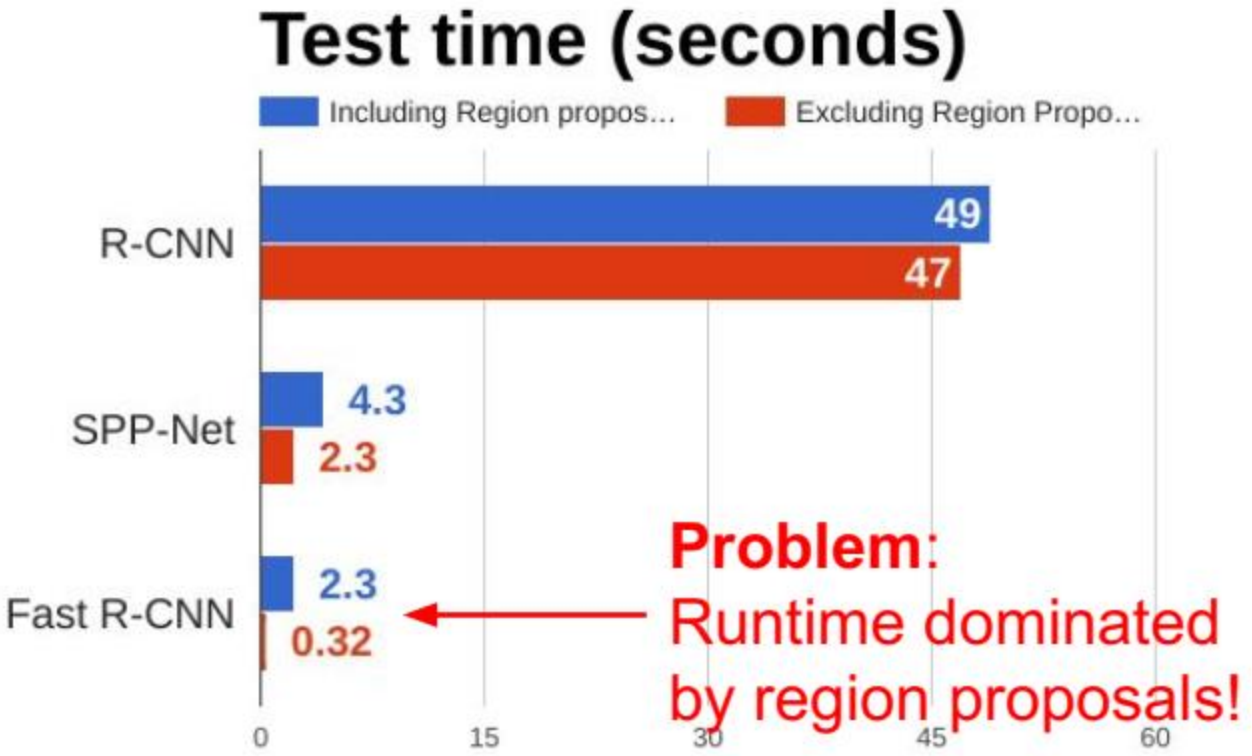
R-CNN vs Fast R-CNN



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

R-CNN vs Fast R-CNN





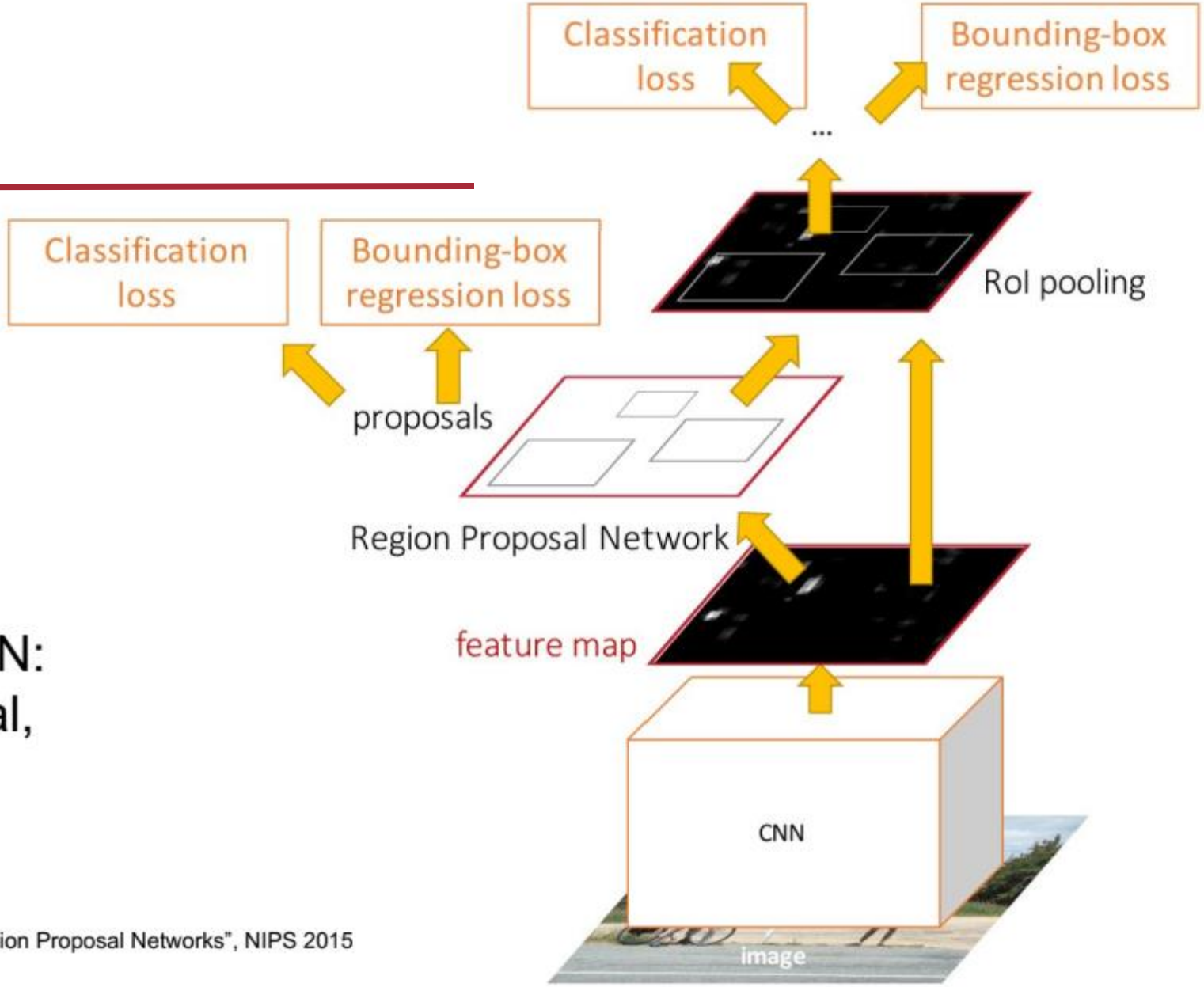
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN:

Make CNN do proposals!

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

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



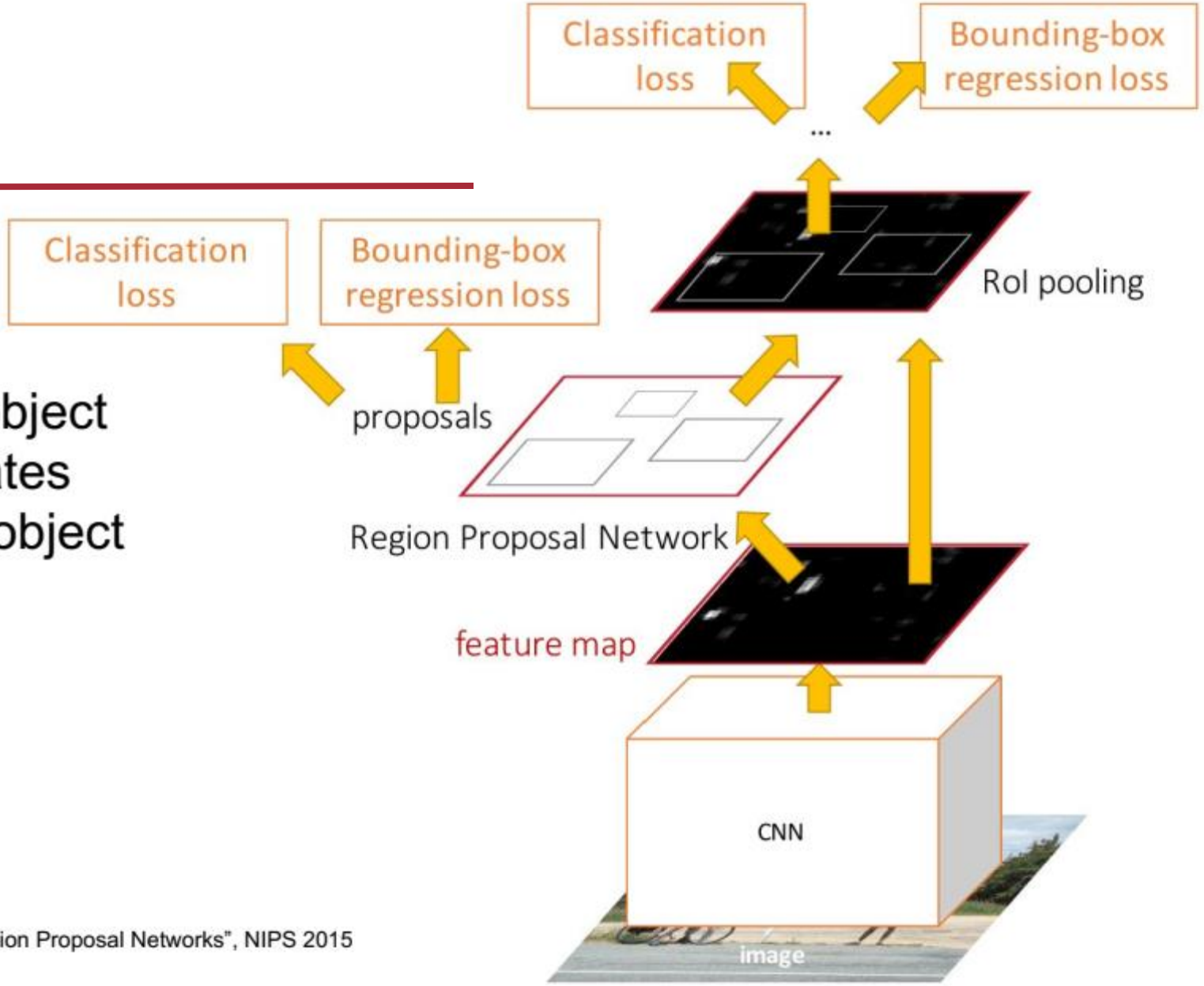
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

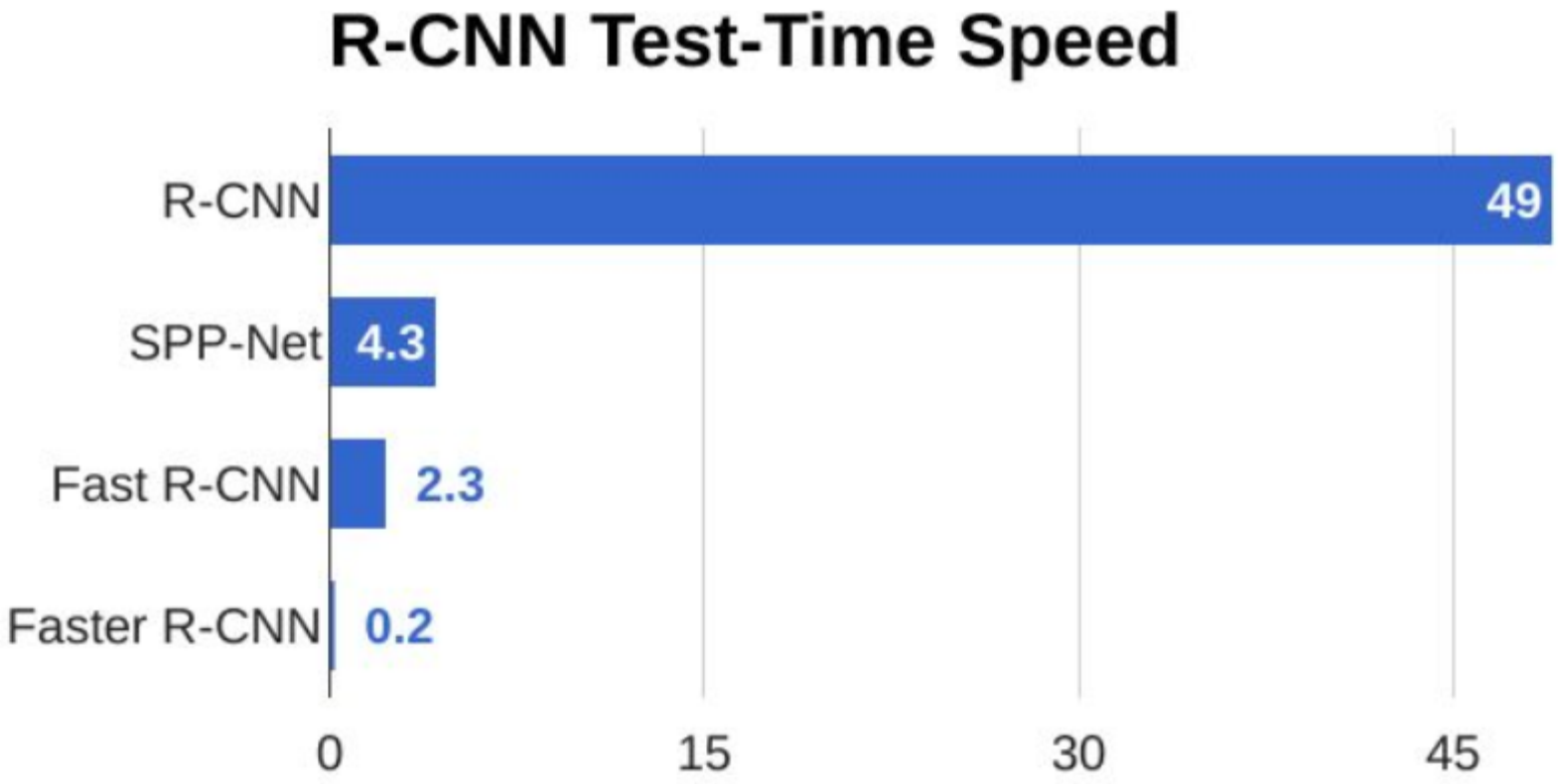
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



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!



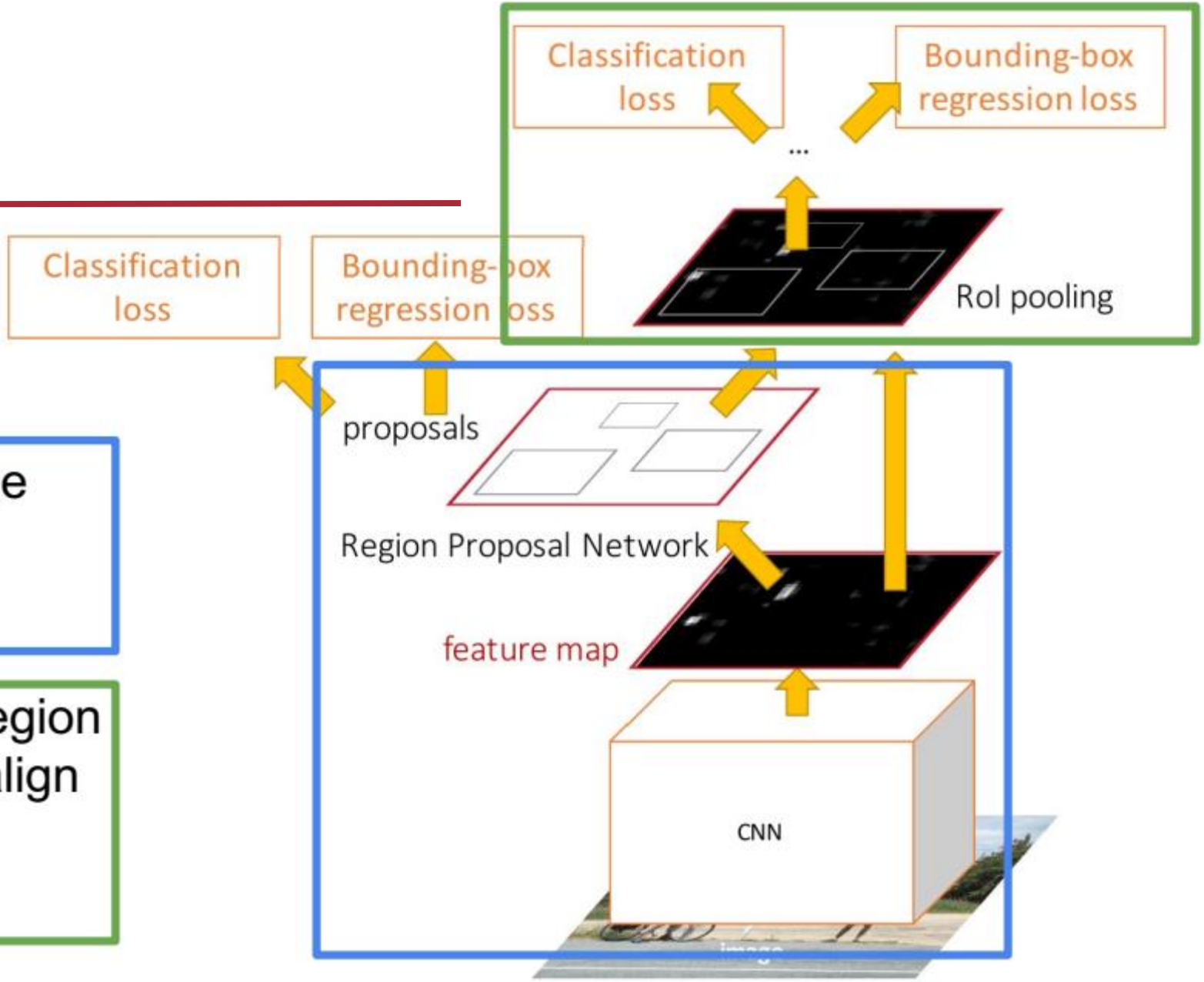
Faster R-CNN:

Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

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

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



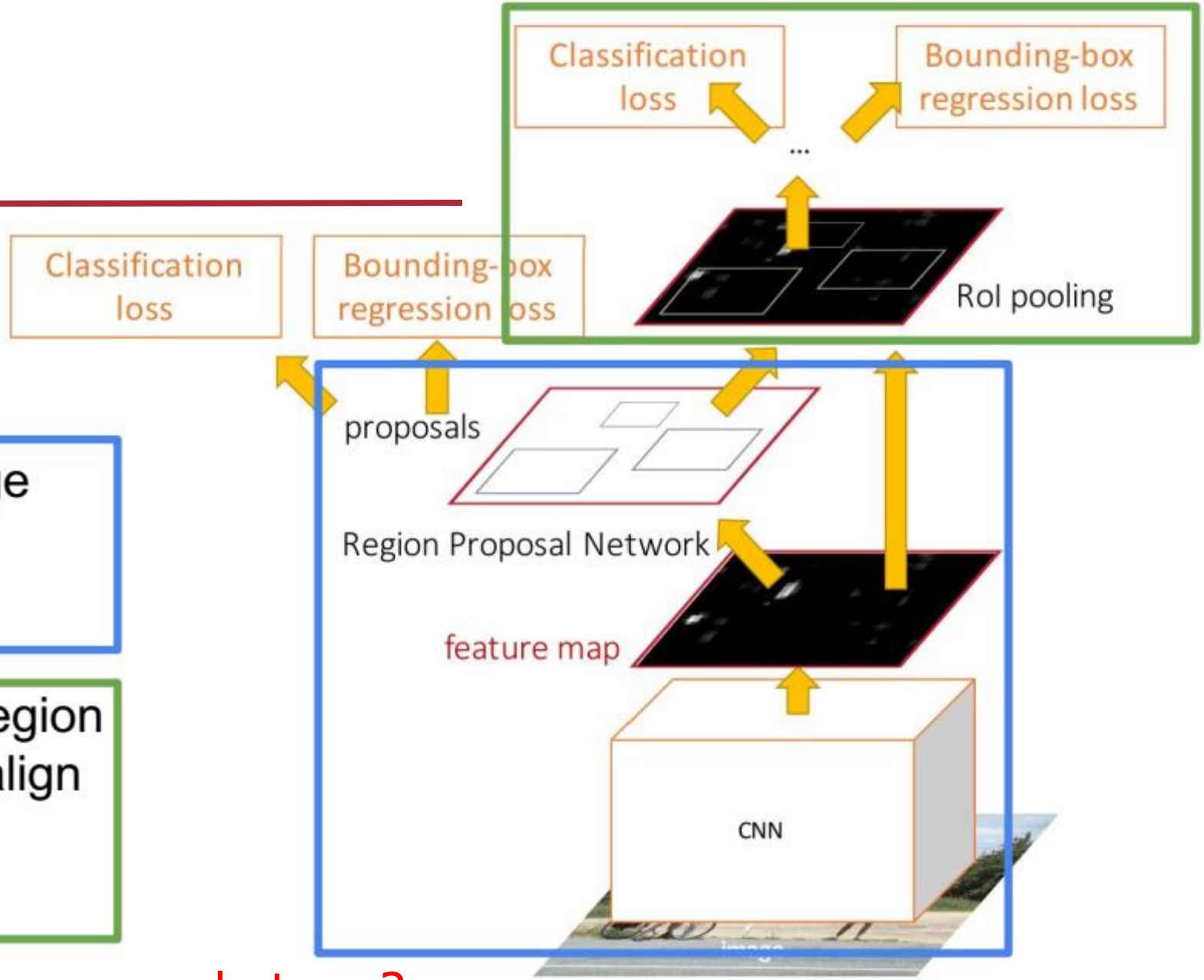
Faster R-CNN:

Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

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

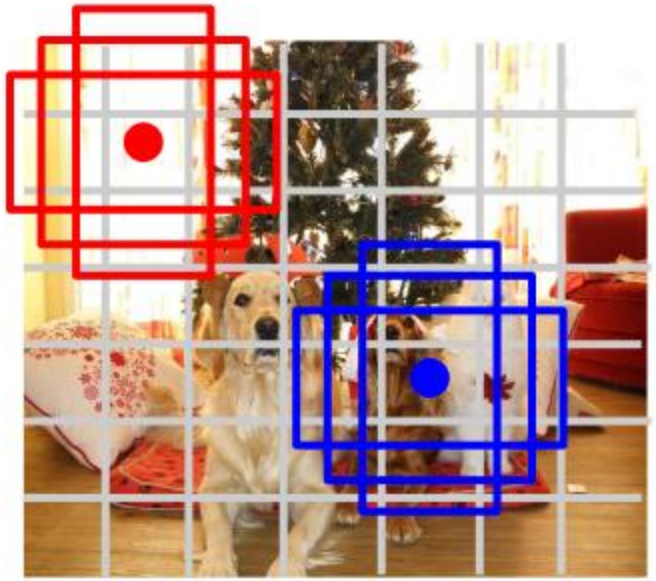
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Do we really need the second stage?

Single-stage object detectors: *YOLO/ SSD/ RetinaNet*

Input image
3 x H x W



Divide image into grid
7 x 7

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

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

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

Object detection: *Lots of variables …*

Backbone Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

"Meta-Architecture"
Two-stage: Faster R-CNN
Single-stage: YOLO / SSD
Hybrid: R-FCN

Image Size
 # Region Proposals
…

Takeaways
Faster R-CNN is slower but more accurate

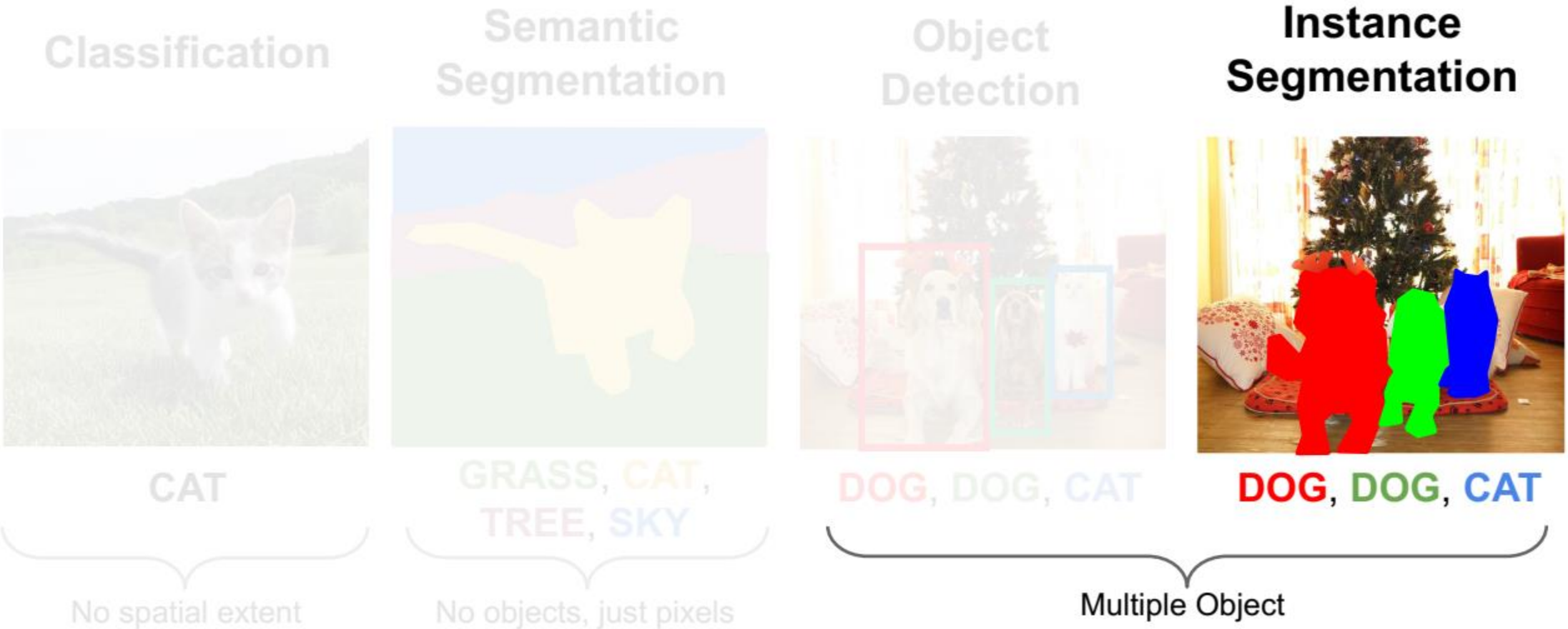
SSD is much faster but not as accurate

Bigger / Deeper backbones work better

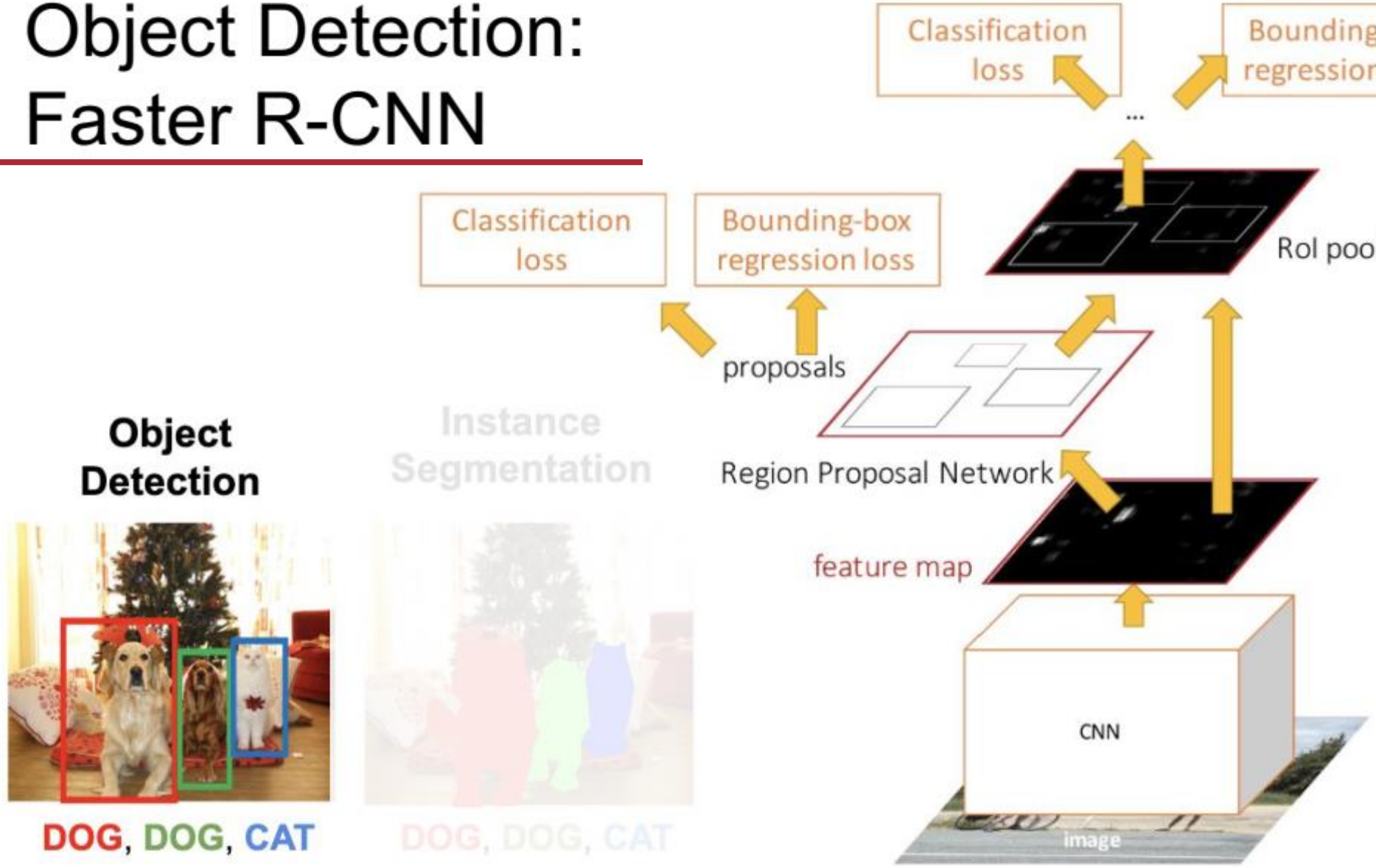
Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017
Zou et al, "Object Detection in 20 Years: A Survey", arXiv 2019

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

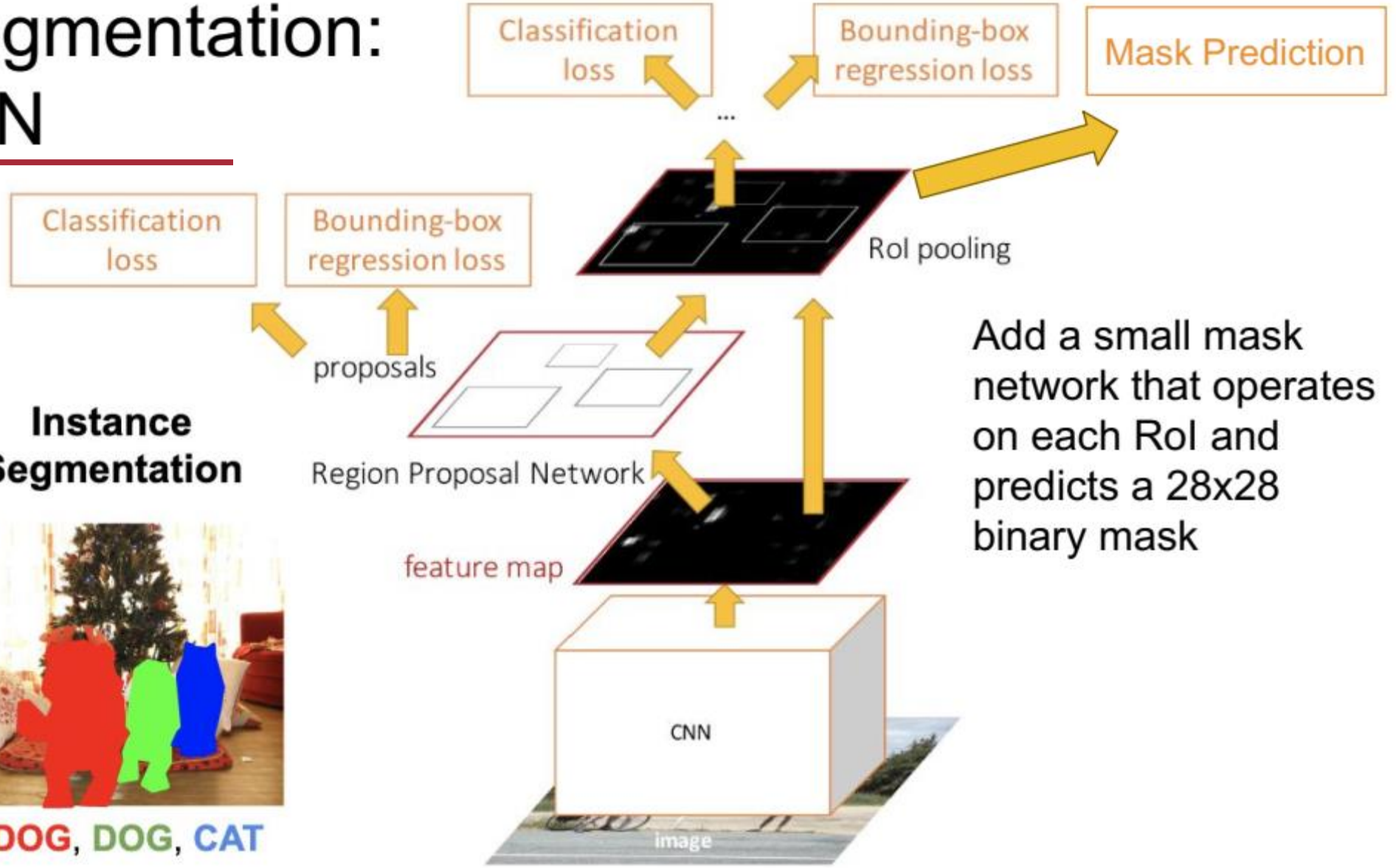
Instance segmentation



Object Detection: Faster R-CNN

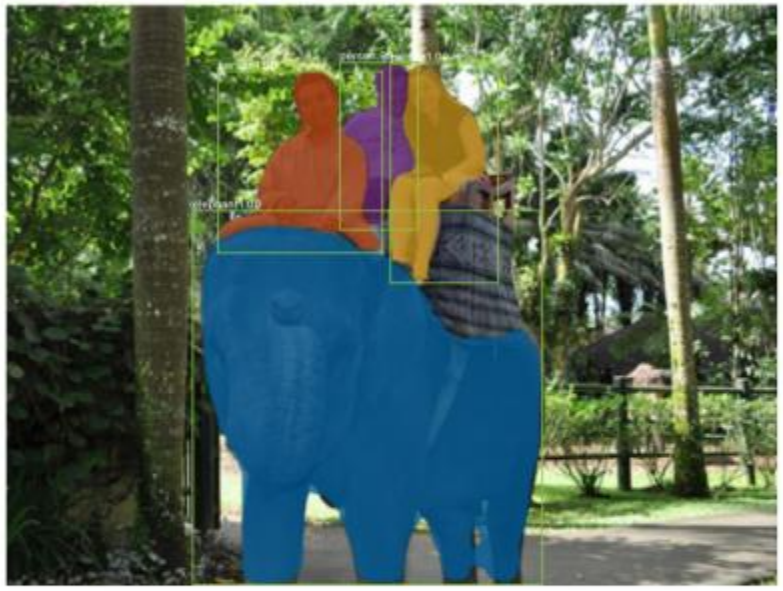


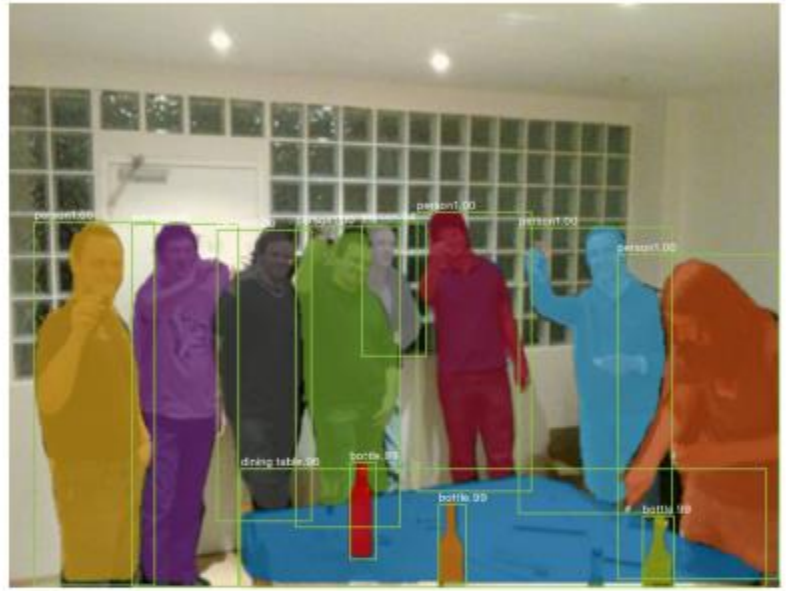
Instance Segmentation: Mask R-CNN



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

Mask R-CNN: Very good results!



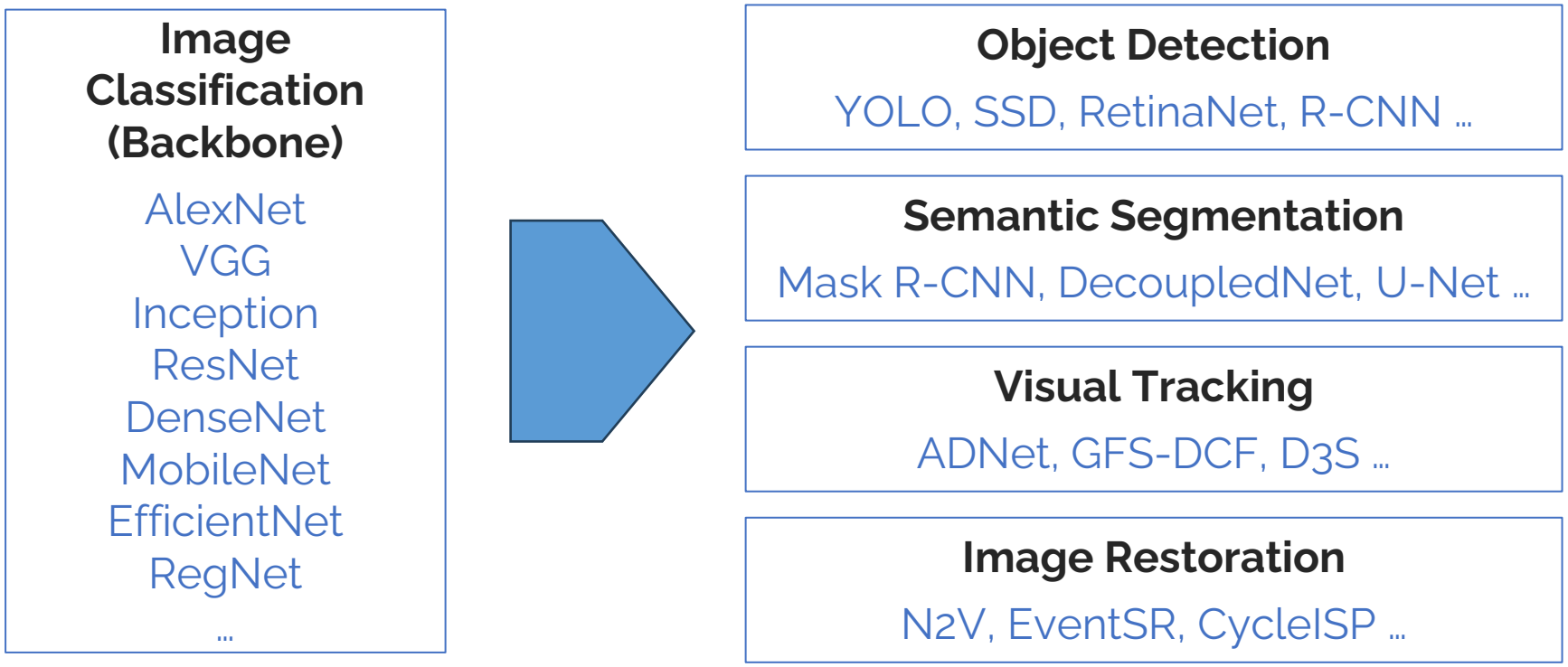


He et al, “Mask R-CNN”, ICCV 2017

Mask R-CNN: Also does pose

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

Computer Vision Tasks



Backbone Networks

Meta Architectures

Thanks