

CNNs Vision Applications

M. Soleymani

Sharif University of Technology

Spring 2026

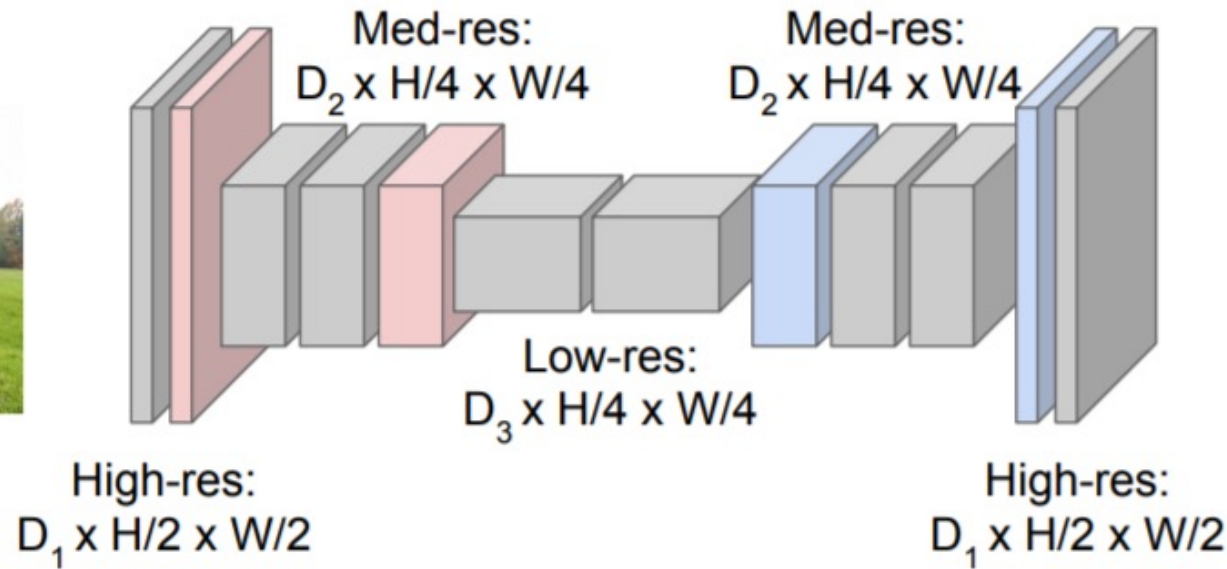
Most slides have been adopted from Fei Fei Li and colleagues lectures, cs231n
and some from Bhiksha Raj, 11-785, CMU

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:
 $H \times W$

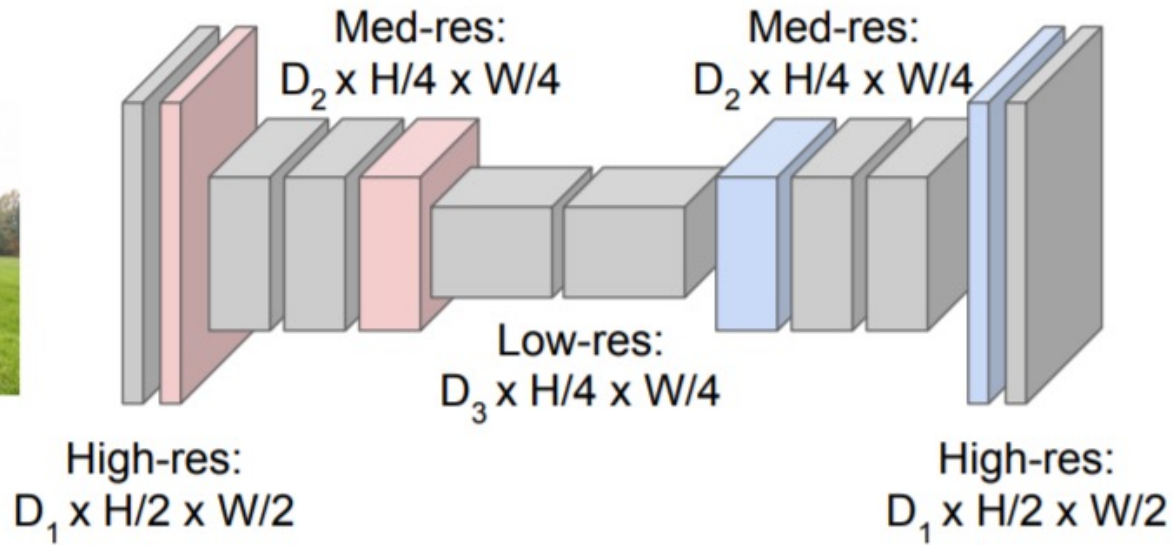
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

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



Upsampling:
???



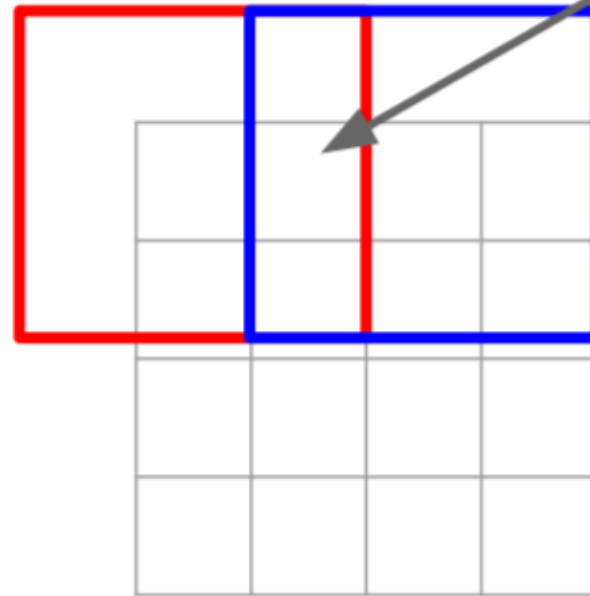
Predictions:
 $H \times W$

Learnable Upsampling: Transposed 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

Other names:

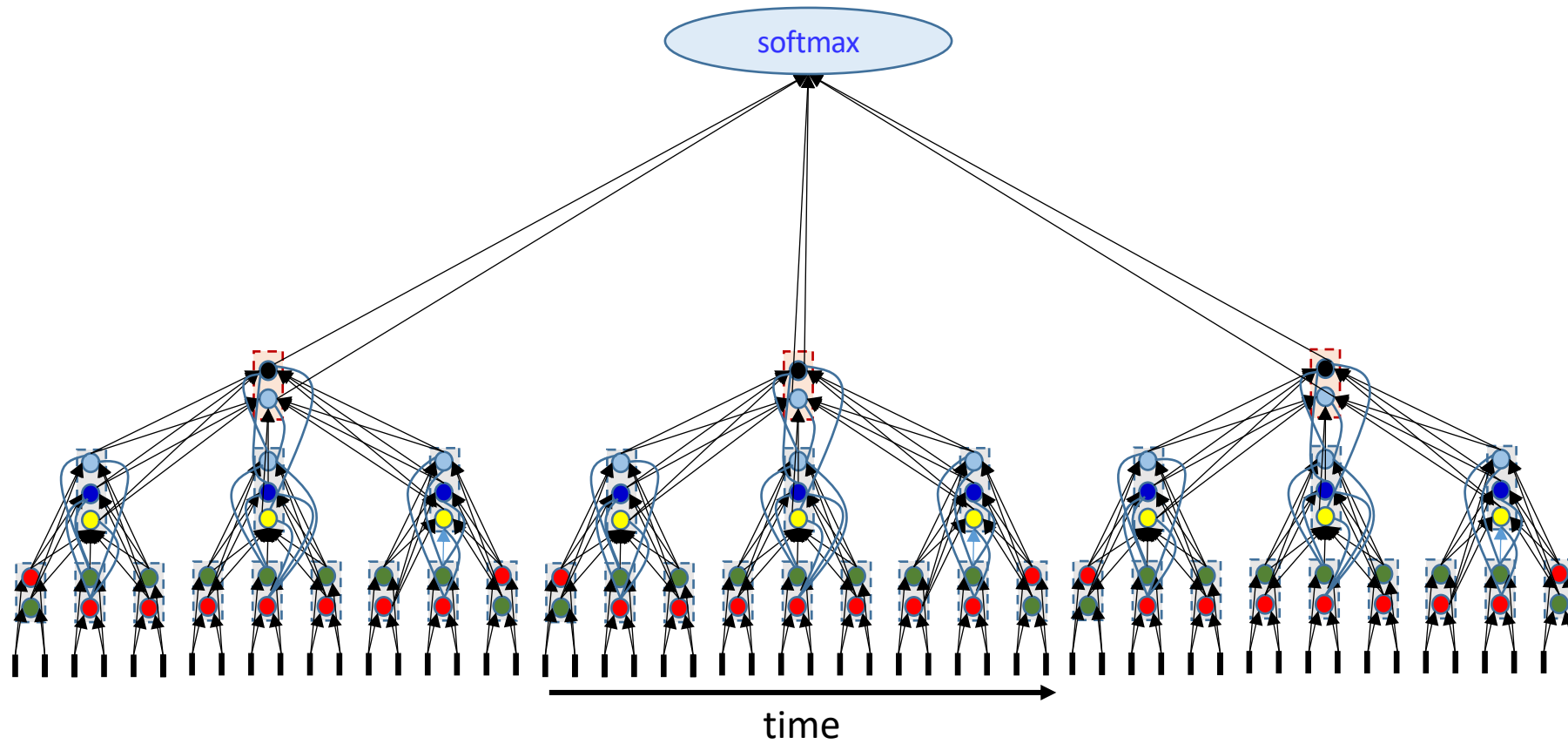
Deconvolution (bad)

Upconvolution

Fractionally strided convolution

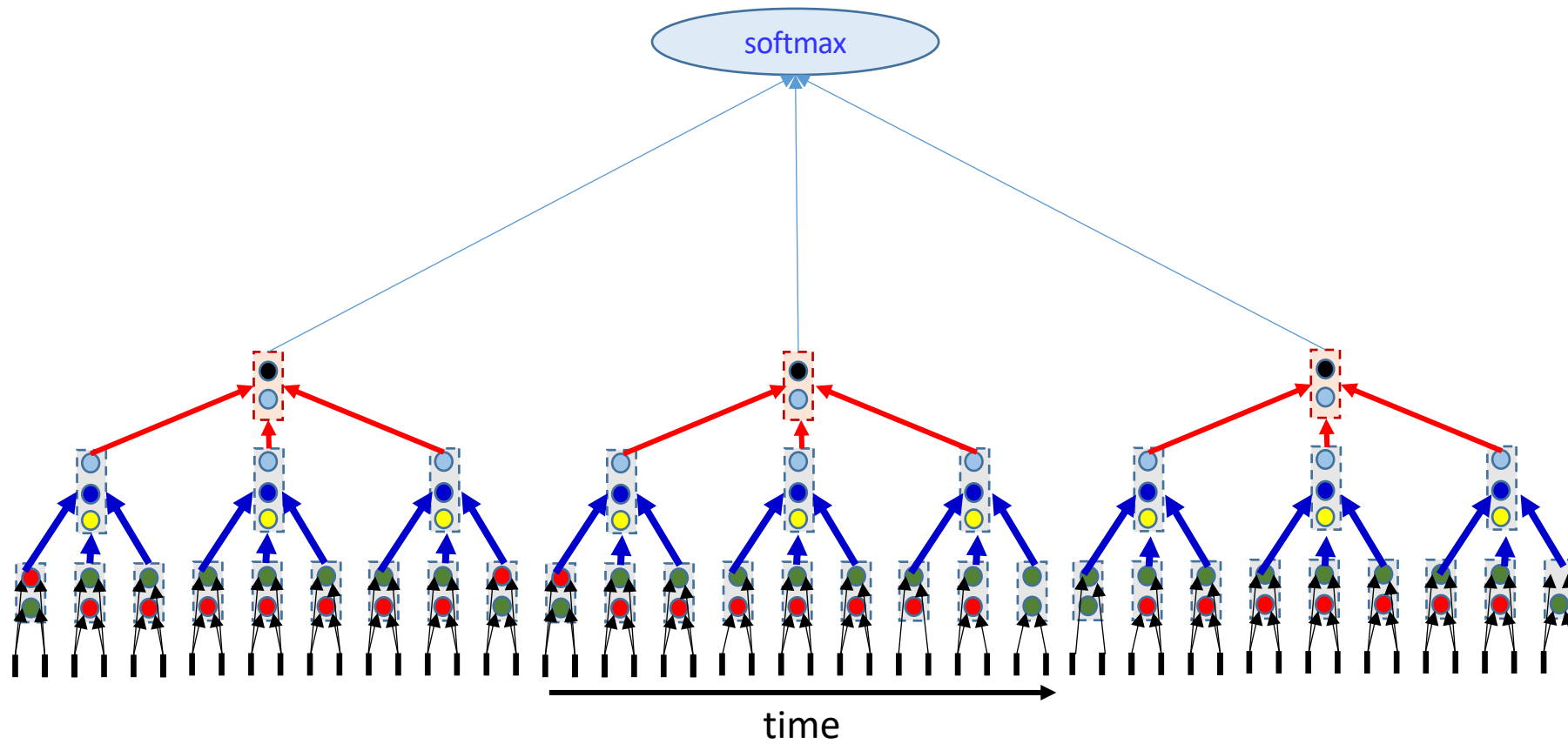
Backward strided convolution

1-D Conv



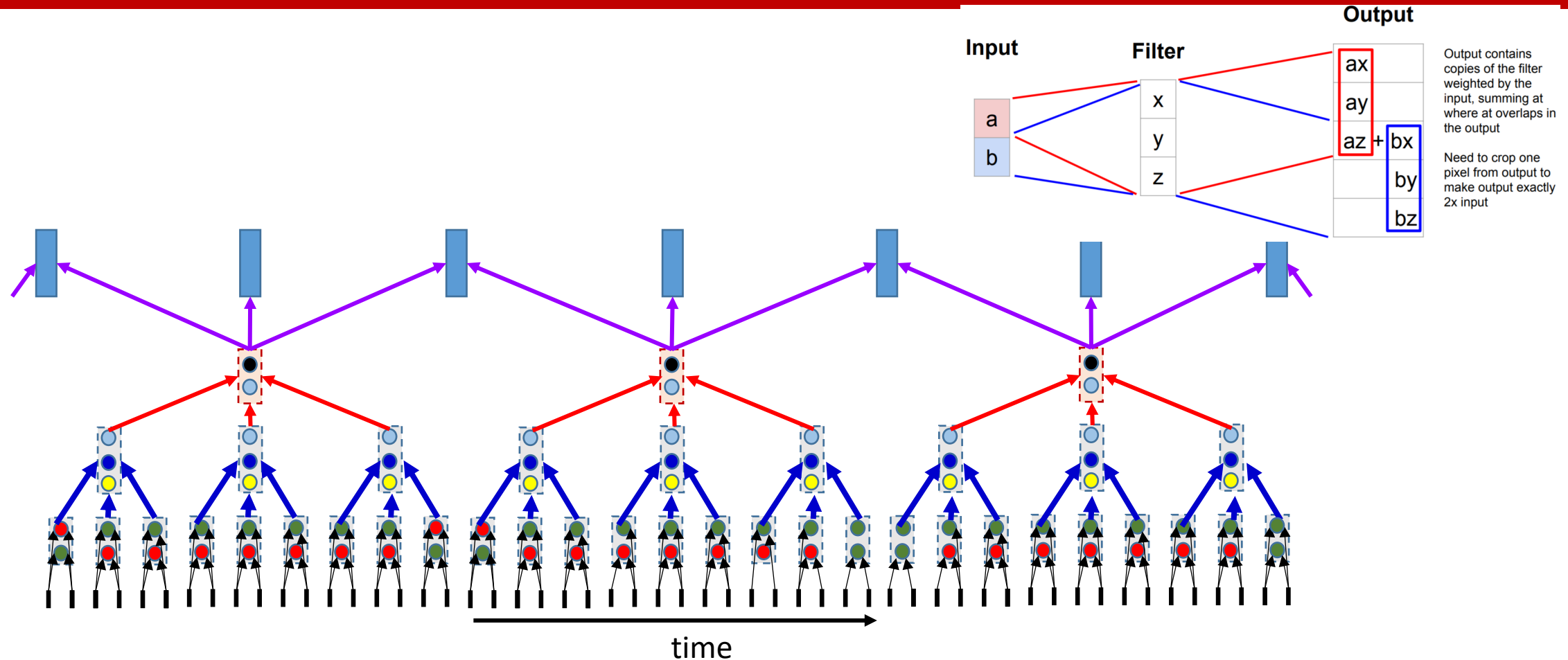
- We've seen this before.. where??

1-D Conv



- Simplified diagram

1D Transposed Conv



- *Maintaining Symmetry:*

- Vertical bars in the 4th layer are regularly arranged w.r.t. bars of layer 3
- The pattern of values of upward weights for each of the three pink (3rd layer) bars is identical

1D Transposed Conv

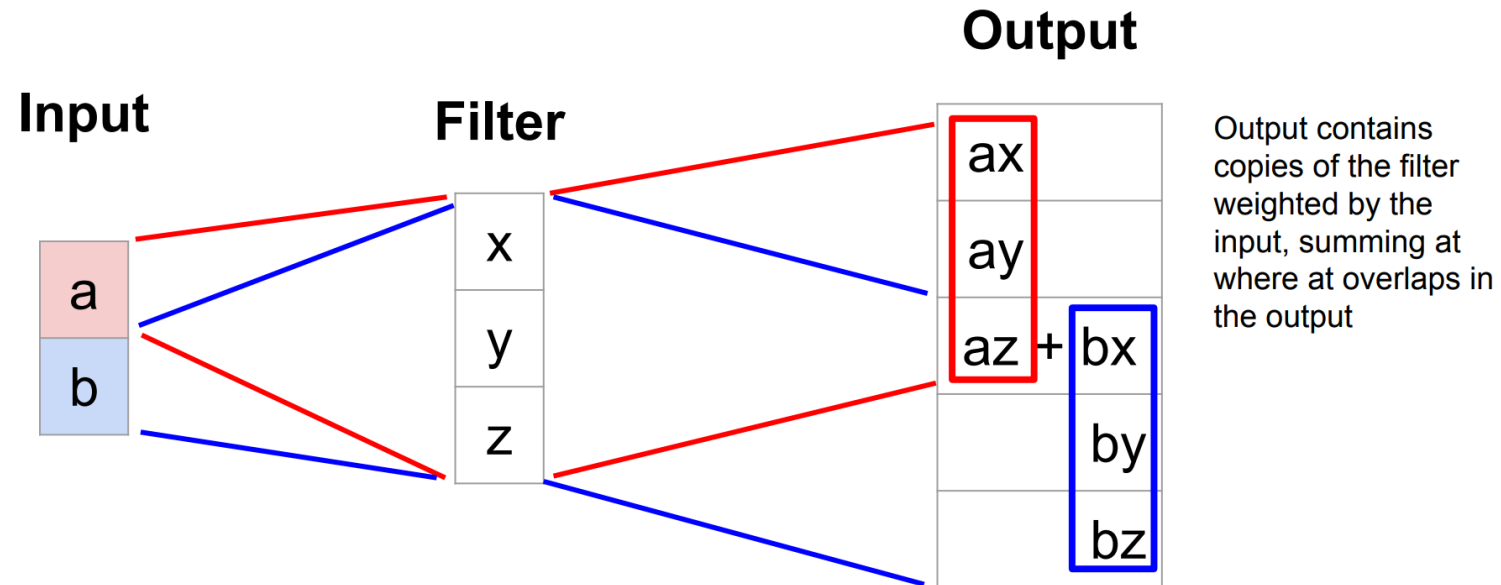
- Connection rules are transposed for expanding layers
 - In shrinking layers, the pattern of *incoming weights* is identical for each bar
 - In expanding layers, the pattern of *outgoing (upward) weights* is identical for each bar

- When thought of as an MLP, can write

$$Z_l = W_l Y_{l-1}$$

- W_l is broader than tall for a shrinking layer
- W_l is taller than broad for an expanding layer
 - Sometimes viewed as the transpose of a broad matrix
- Leading to terminology “transpose convolution”

Learnable Upsampling: 1D Example



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

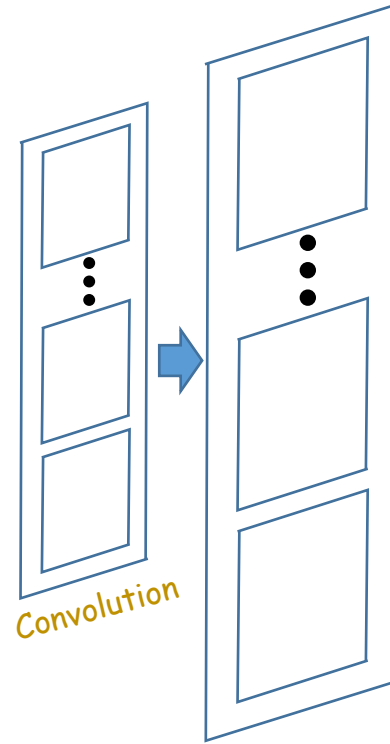
Transposed convolution multiplies by the transpose of the same matrix:

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

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

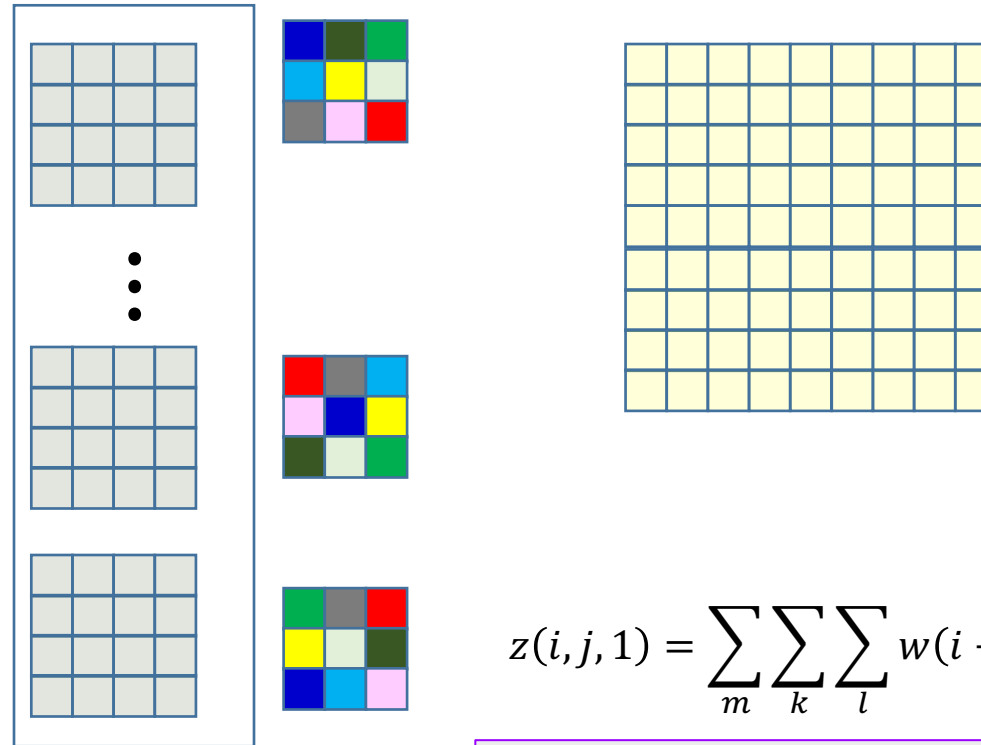
Example: 1D transposed conv, kernel size=3, stride=2, padding=0

In 2-D



- Similar computation

2D expanding convolution

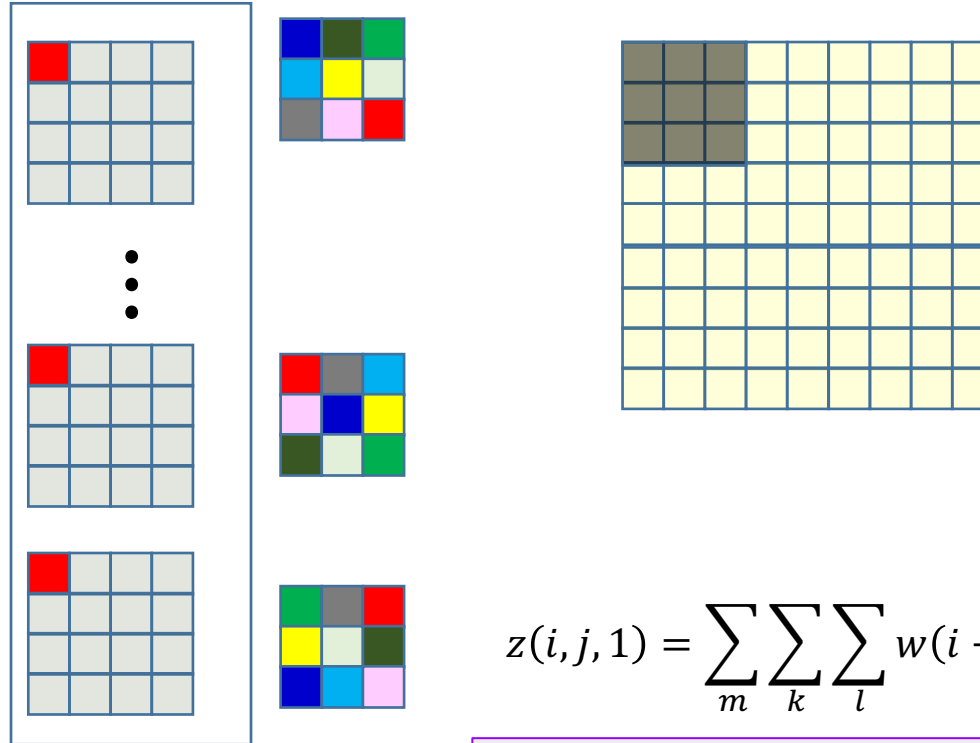


$$z(i, j, 1) = \sum_m \sum_k \sum_l w(i - kb, j - lb, m, 1) I(k, l, m)$$

b is the "stride"
(scaling factor between the sizes of Z and Y)

- Output size is typically an integer multiple of input
 - +1 if filter width is odd
 - Easier to determine assignment of output to input

2D expanding convolution

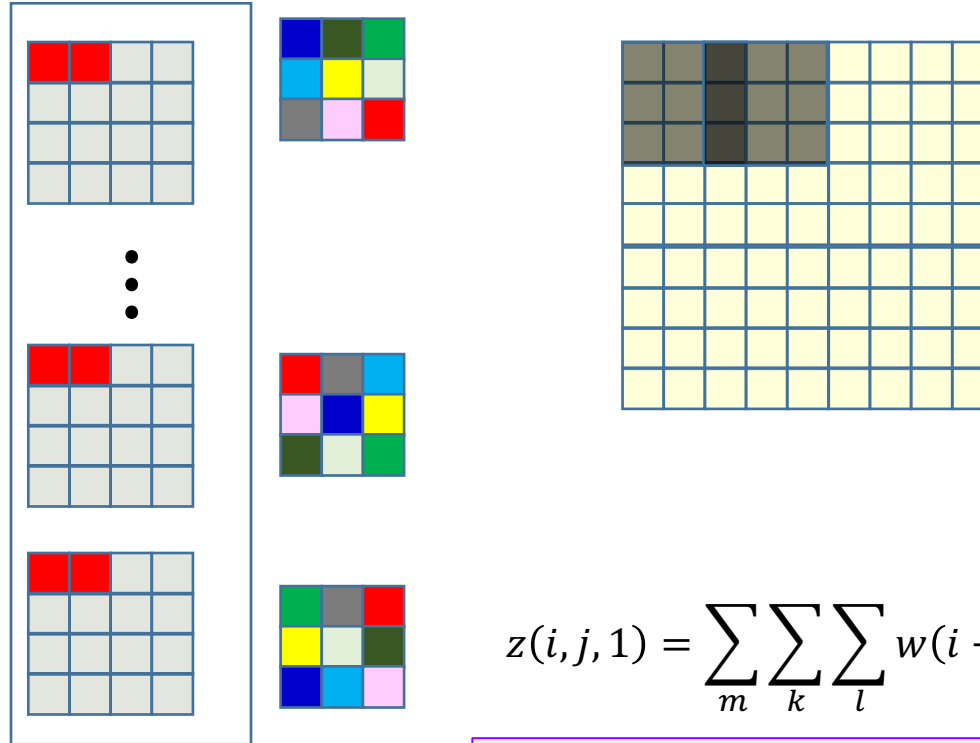


$$z(i, j, 1) = \sum_m \sum_k \sum_l w(i - kb, j - lb, m, 1) I(k, l, m)$$

b is the "stride"
(scaling factor between the sizes of Z and Y)

- Output size is typically an integer multiple of input
 - +1 if filter width is odd
 - Easier to determine assignment of output to input

2D expanding convolution

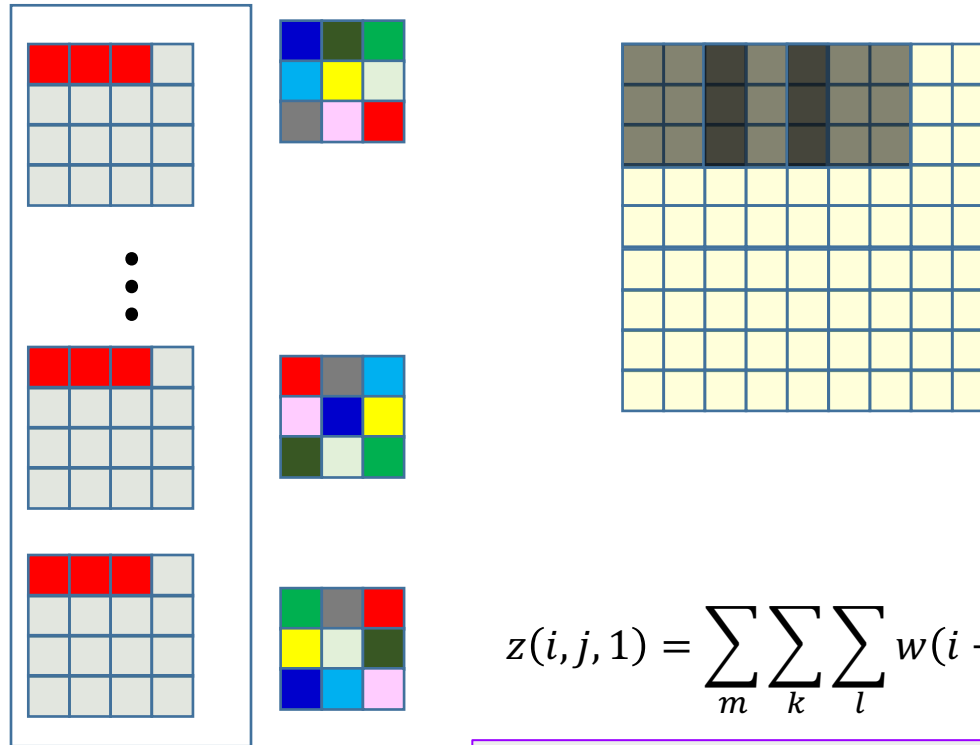


$$z(i, j, 1) = \sum_m \sum_k \sum_l w(i - kb, j - lb, m, 1) I(k, l, m)$$

b is the "stride"
(scaling factor between the sizes of Z and Y)

- Output size is typically an integer multiple of input
 - +1 if filter width is odd
 - Easier to determine assignment of output to input

2D expanding convolution

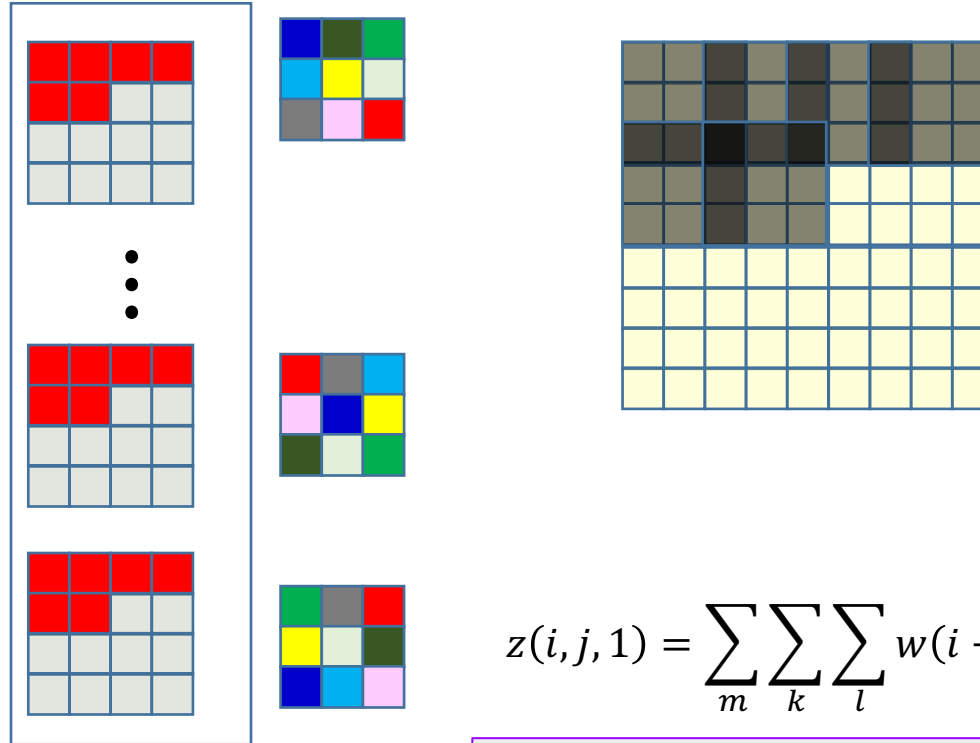


$$z(i, j, 1) = \sum_m \sum_k \sum_l w(i - kb, j - lb, m, 1) I(k, l, m)$$

b is the "stride"
(scaling factor between the sizes of Z and Y)

- Output size is typically an integer multiple of input
 - +1 if filter width is odd
 - Easier to determine assignment of output to input

2D expanding convolution



$$z(i, j, 1) = \sum_m \sum_k \sum_l w(i - kb, j - lb, m, 1) I(k, l, m)$$

b is the "stride"
(scaling factor between the sizes of Z and Y)

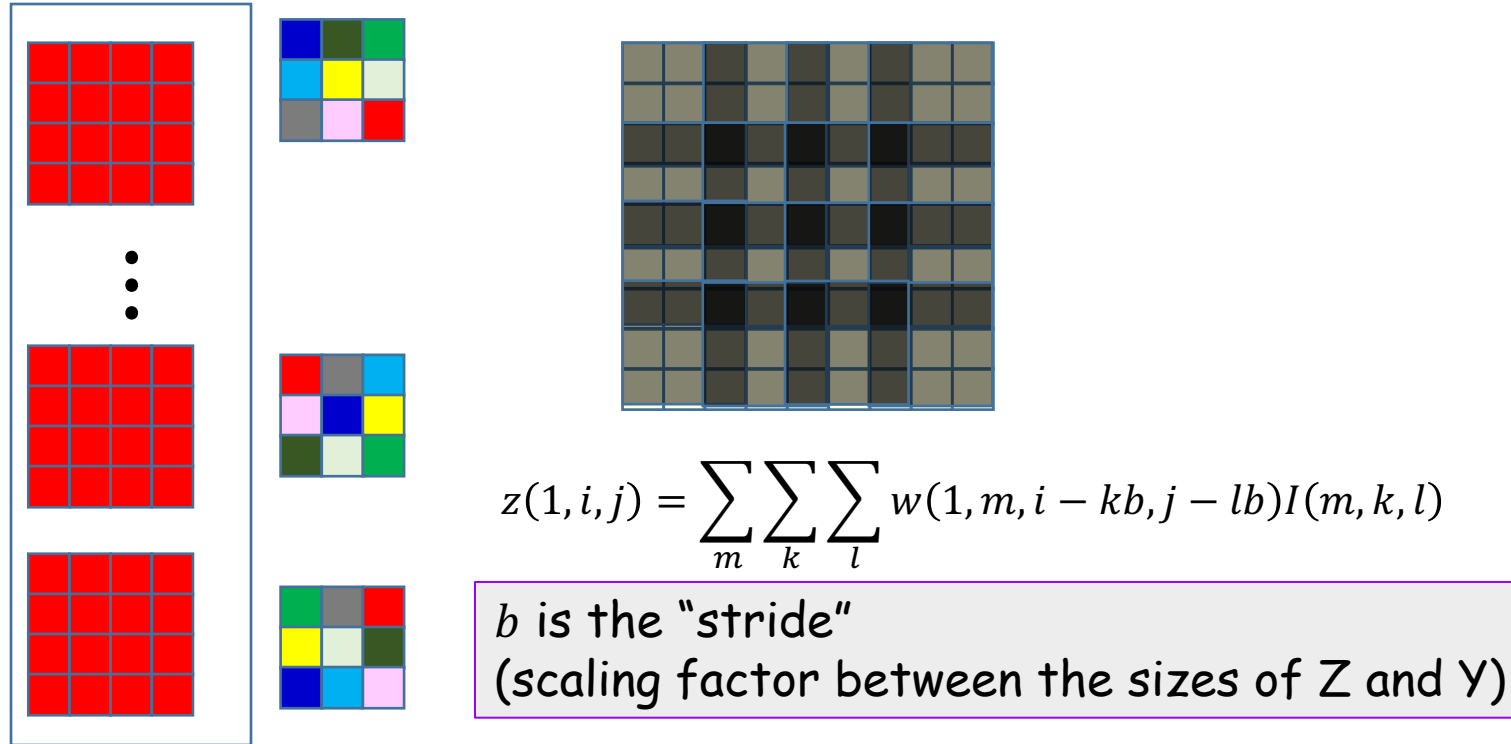
- Output size is typically an integer multiple of input
 - +1 if filter width is odd
 - Easier to determine assignment of output to input

CNN: Expanding convolution layer l

```
Z(l) = zeros(Dl x (Wb+K1) x (Hb+K1)) # b = stride
for j = 1:D1
    for x = 1:W
        for y = 1:H
            for i = 1:D1-1
                for x' = 1:K1
                    for y' = 1:K1
                        z(l,j,(x-1)b+x',(y-1)b+y') +=
                            w(l,j,i,x',y')y(l-1,i,x,y)
```

We leave the rather trivial issue of how to modify this code to compute the derivatives w.r.t w and y to you

2D expanding convolution



- Also called *transpose convolution*
 - If you recast the CNN as a shared-parameter MLP, expanding layers have weight matrices that are taller than wide
- Also called "deconvolution"
 - Strictly speaking, abuse of terminology

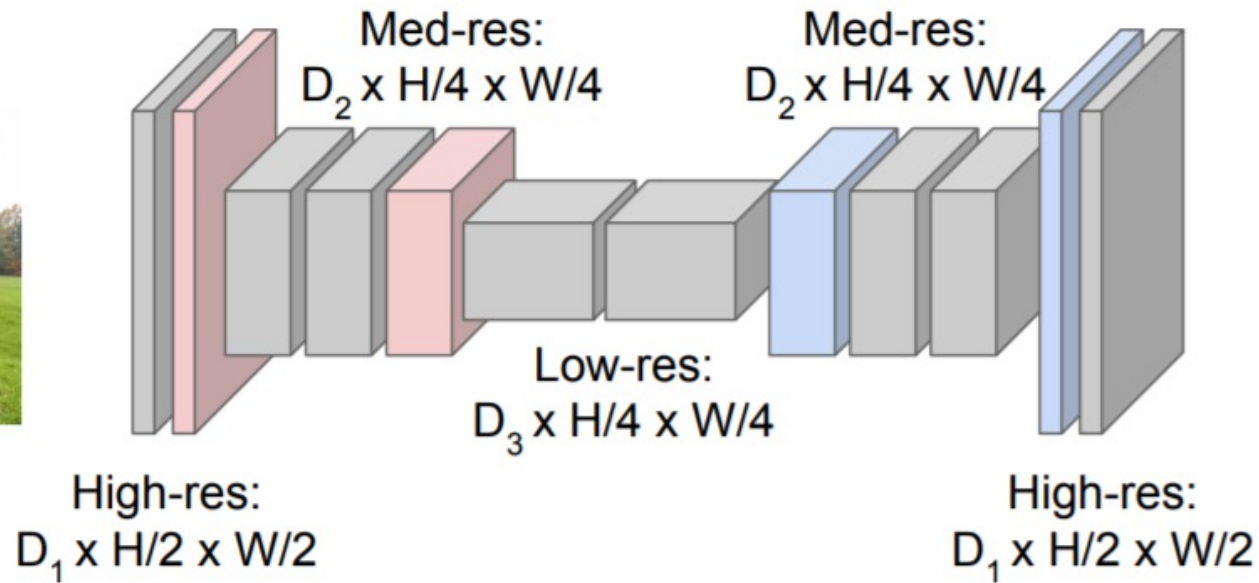
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

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



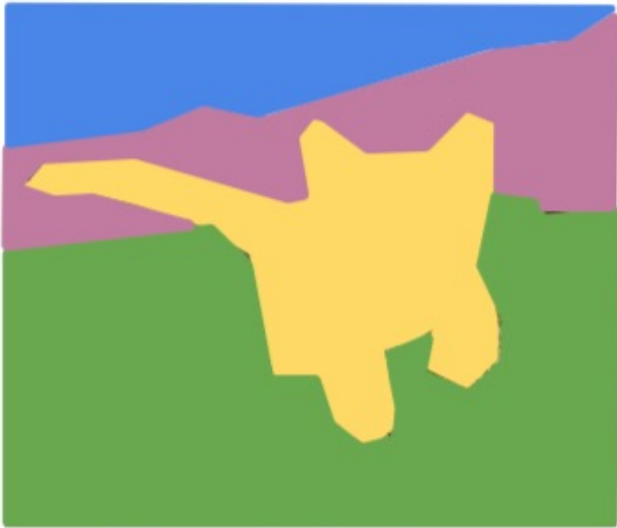
Upsampling:
Unpooling or strided
transpose convolution



Predictions:
 $H \times W$

Computer Vision Tasks

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

[This image is CC0 public domain](#)

Classification + Localization

Classification: C classes

Input: Image

Output: Class label

Evaluation metric: Accuracy



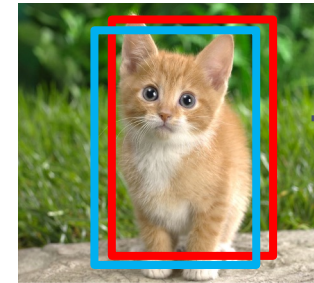
CAT

Localization:

Input: Image

Output: Box in the image (x, y, w, h)

Evaluation metric: Intersection over Union



(x, y, w, h)

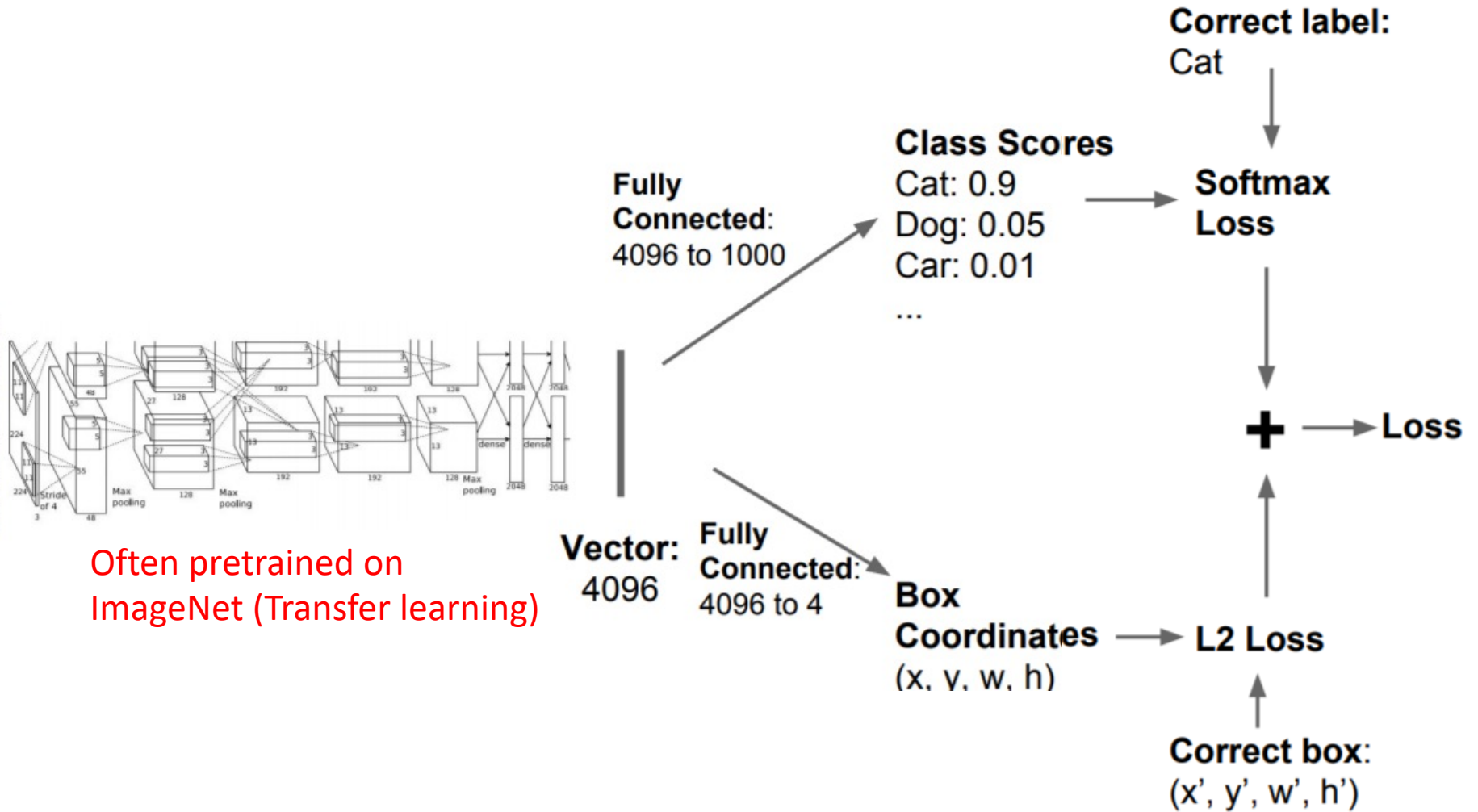
Classification + Localization: Do both

Classification + Localization



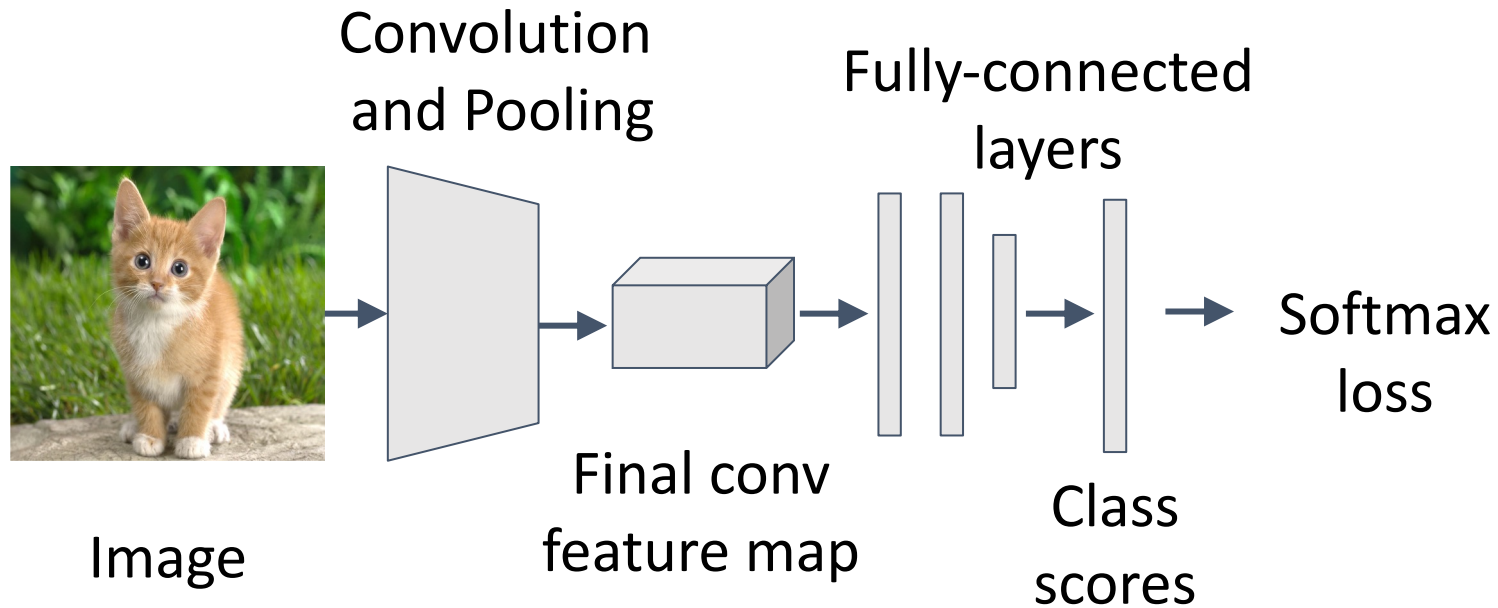
This image is CC0 public domain

Often pretrained on
ImageNet (Transfer learning)



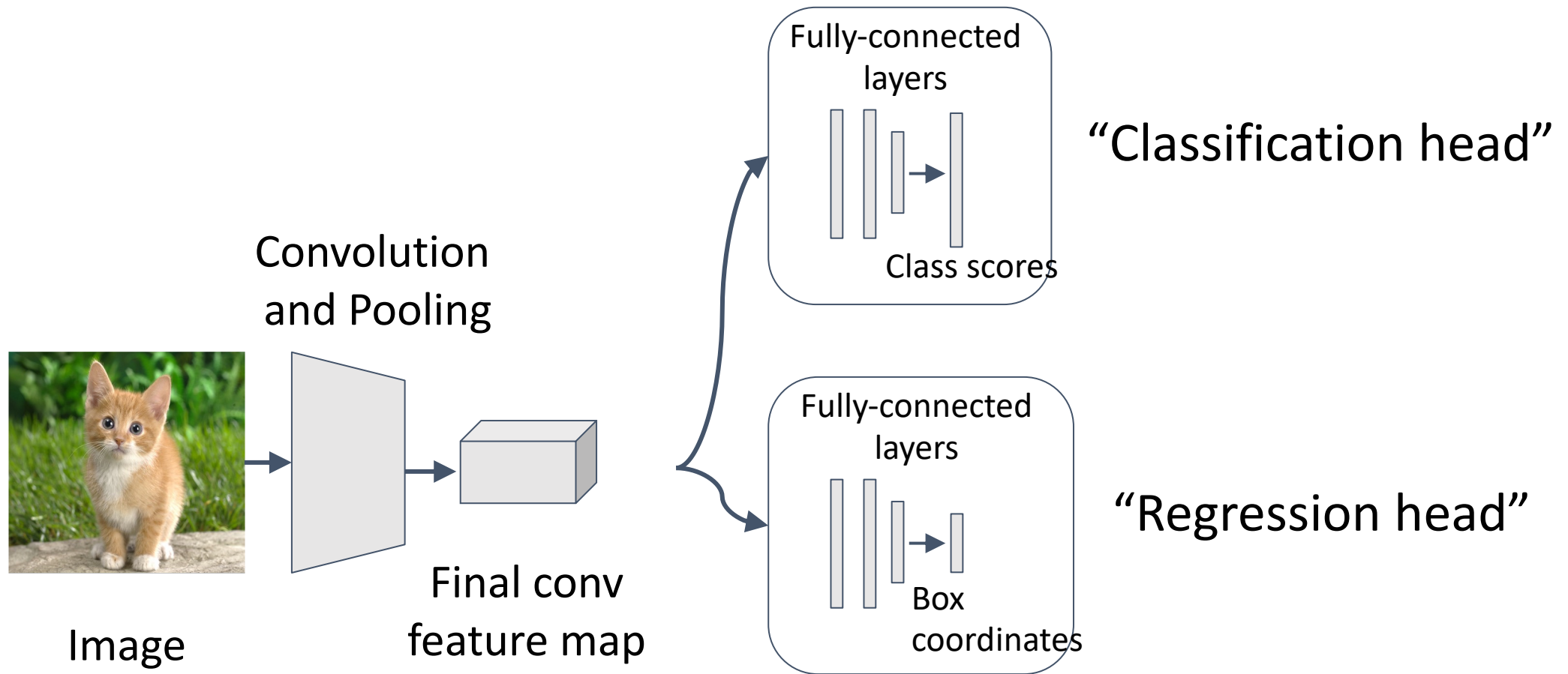
Simple Recipe for Classification + Localization

- **Step 1:** Train (or download) a classification model (e.g., VGG)



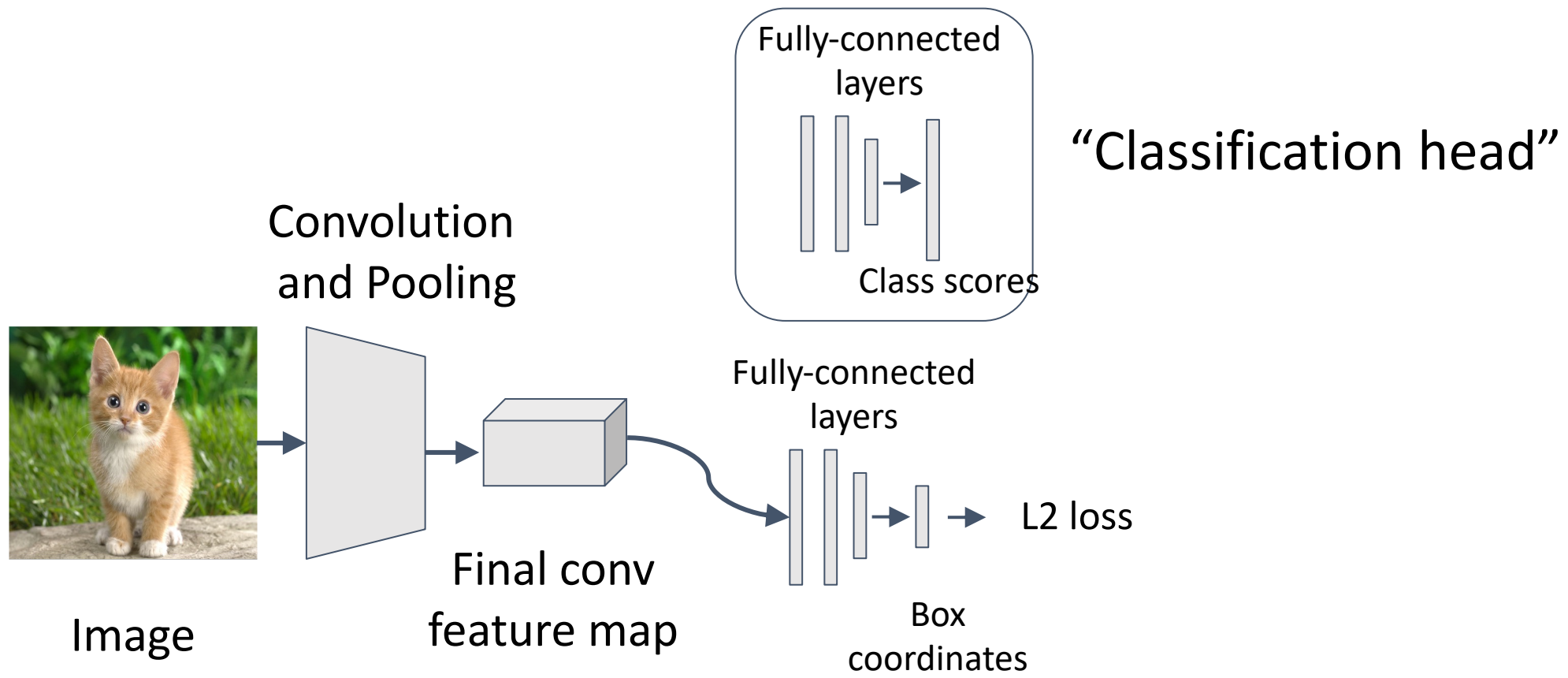
Simple Recipe for Classification + Localization

- **Step 1:** Train (or download) a classification model (e.g., VGG)
- **Step 2:** Attach new fully-connected “regression head” to the network



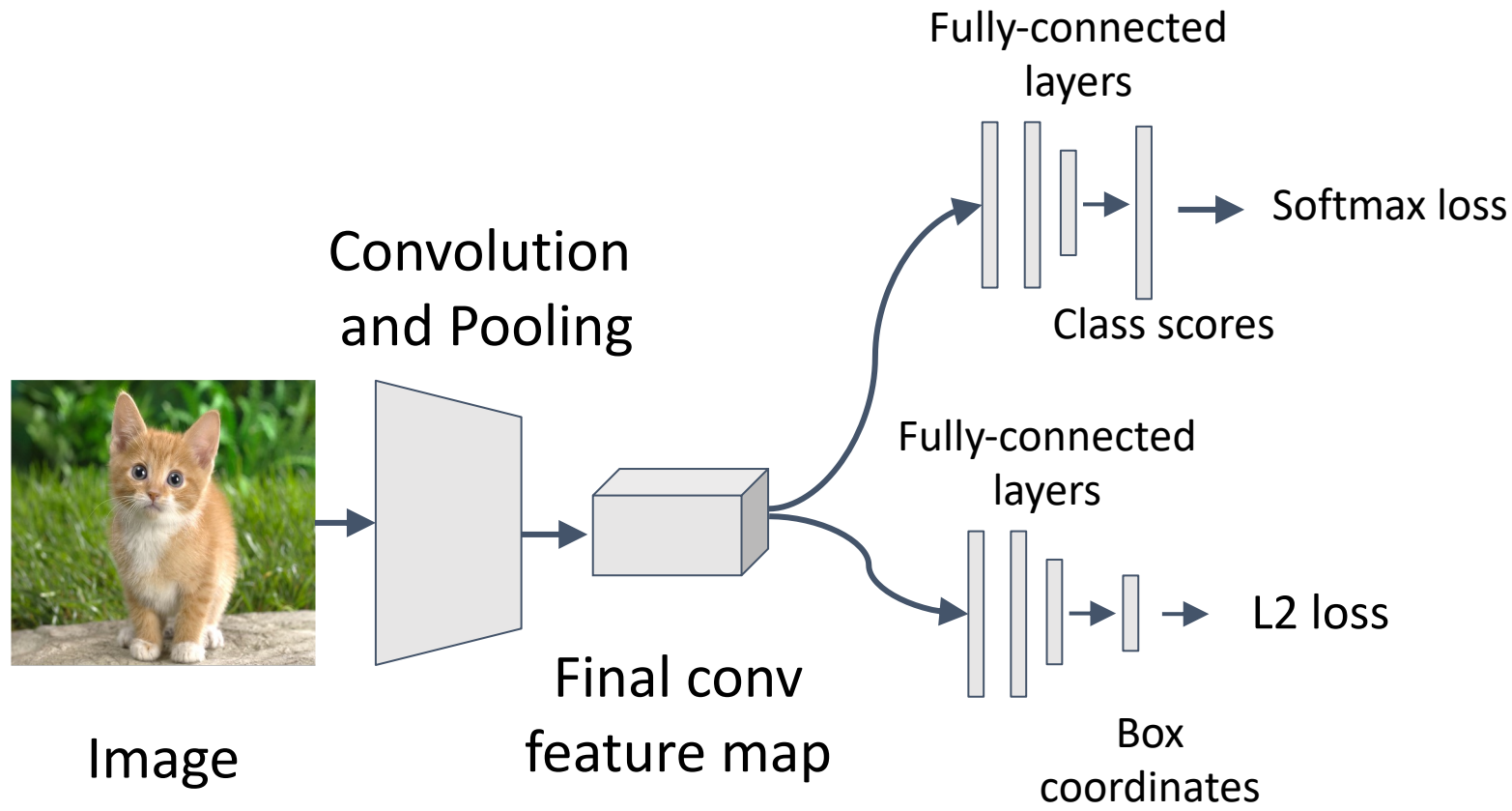
Simple Recipe for Classification + Localization

- **Step 1:** Train (or download) a classification model (e.g., VGG)
- **Step 2:** Attach new fully-connected “regression head” to the network
- **Step 3:** Train the regression head only with SGD and L2 loss

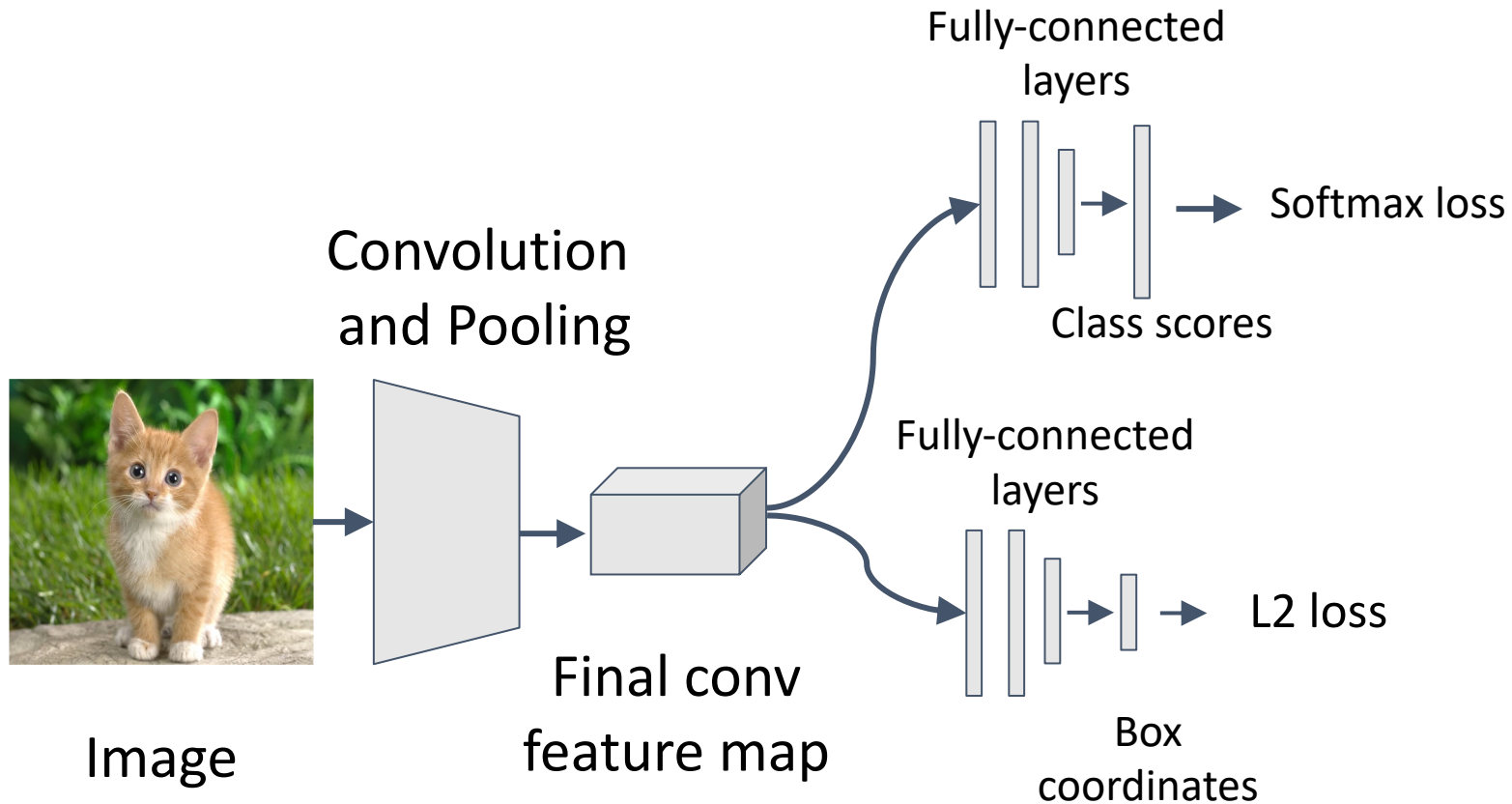


Simple Recipe for Classification + Localization

- **Step 1:** Train (or download) a classification model (e.g., VGG)
- **Step 2:** Attach new fully-connected “regression head” to the network
- **Step 3:** Train the regression head only with SGD and L2 loss



Classification + Localization

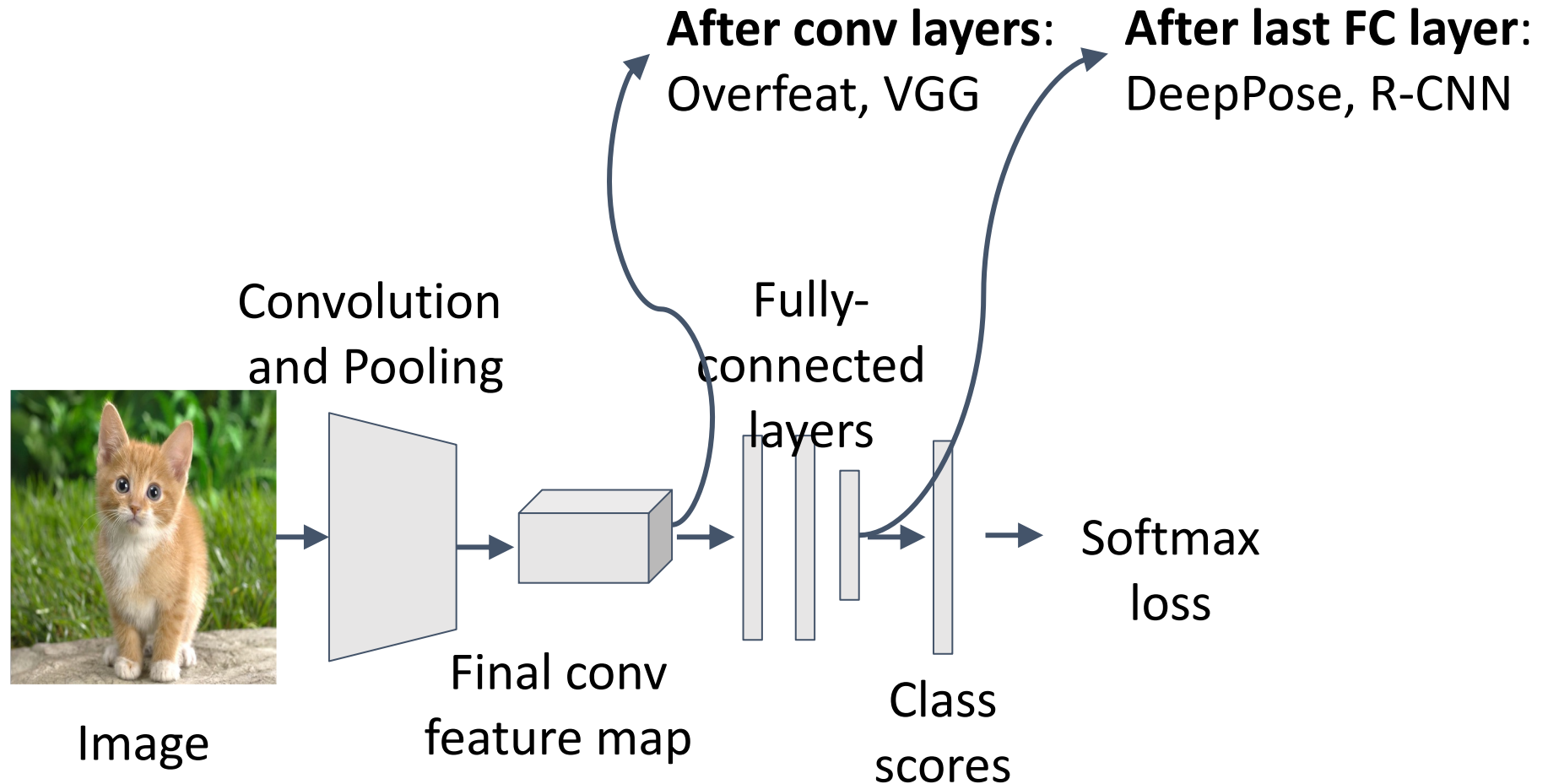


Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

Class specific:
C x 4 numbers
(one box per class)

Where to attach the regression head?



Object detection

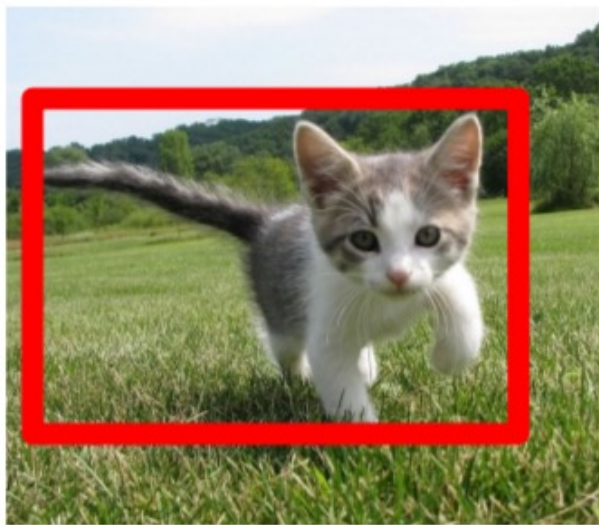
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

**Classification
+ Localization**



CAT

Single Object

**Object
Detection**



DOG, DOG, CAT

Multiple Object

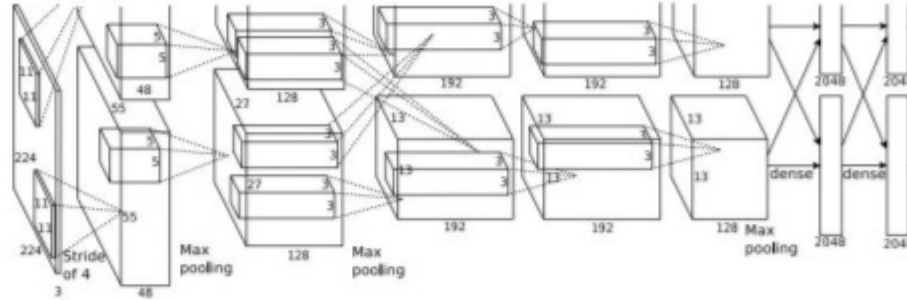
**Instance
Segmentation**



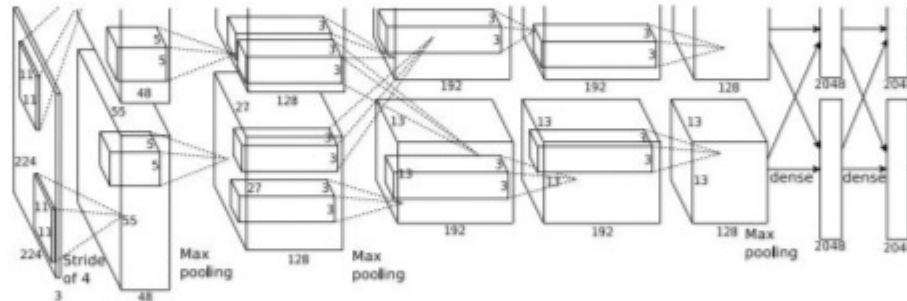
DOG, DOG, CAT

Object Detection as Regression?

Each image needs a different number of outputs!



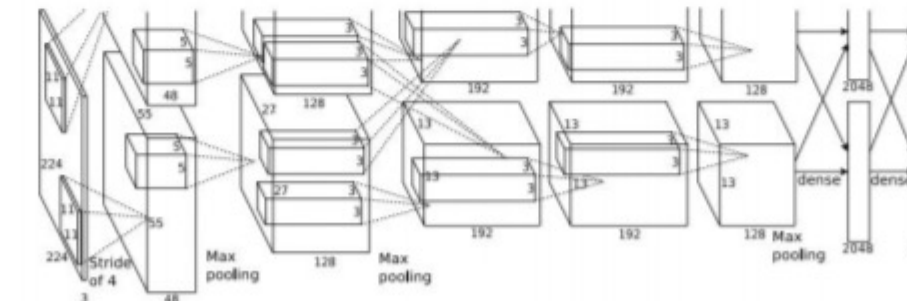
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



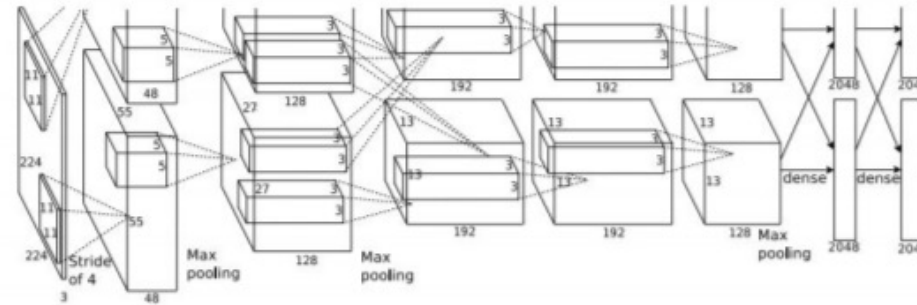
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

....

Object Detection as Classification: Sliding Window

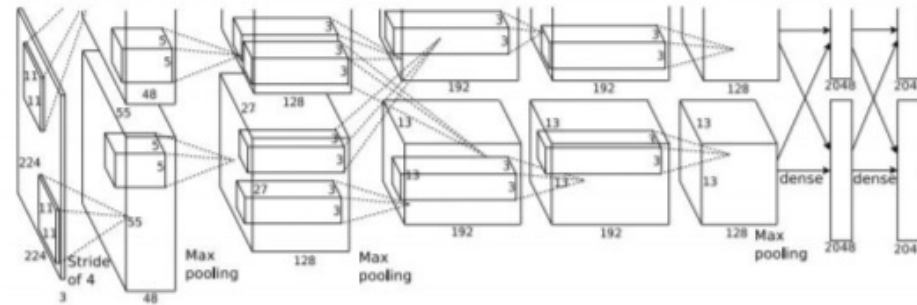
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

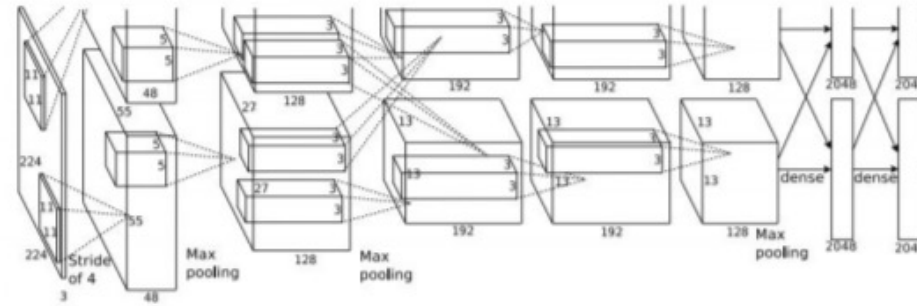
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

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 and scales, very computationally expensive!

Object Detection: Impact of Deep Learning

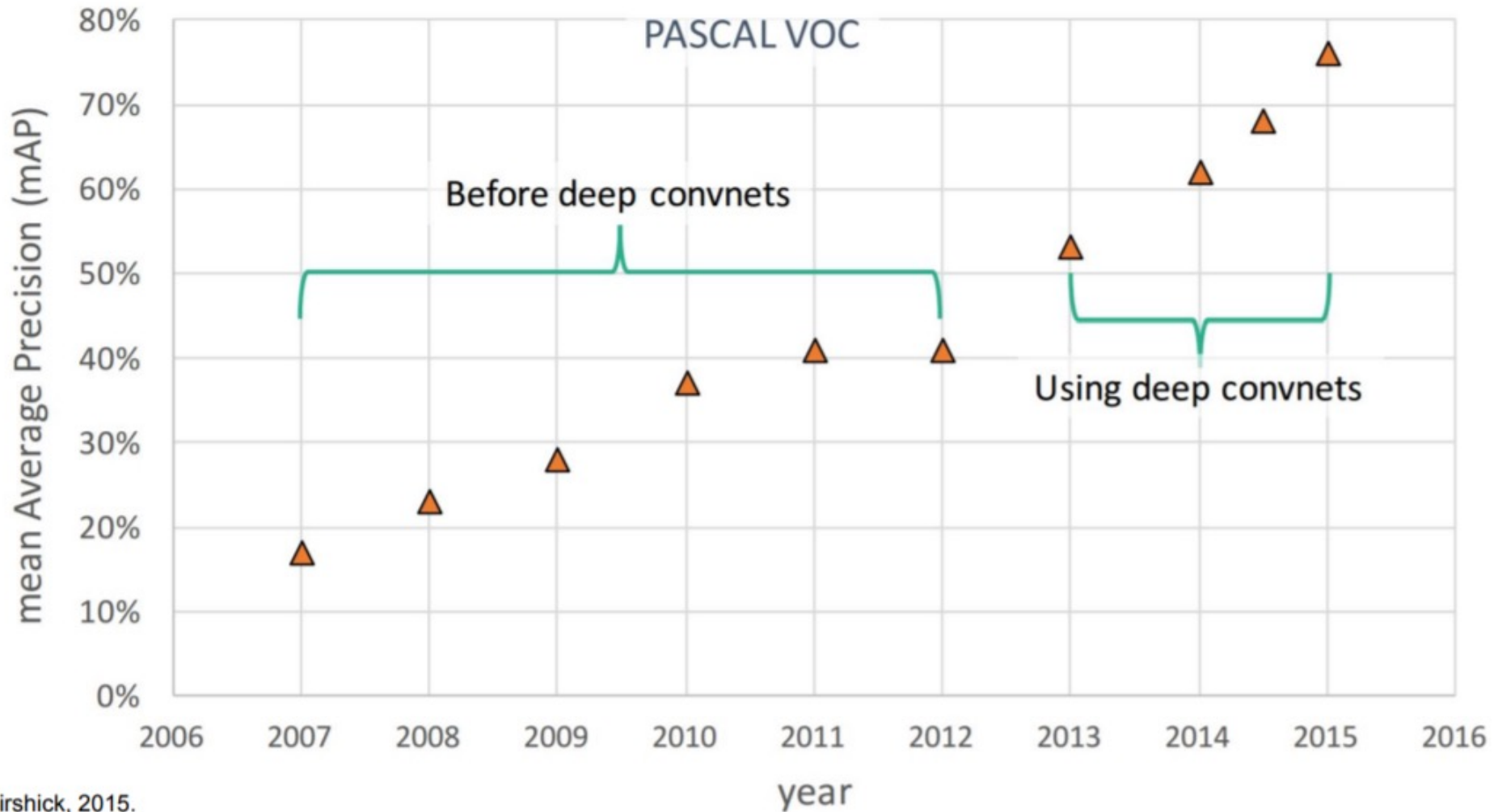
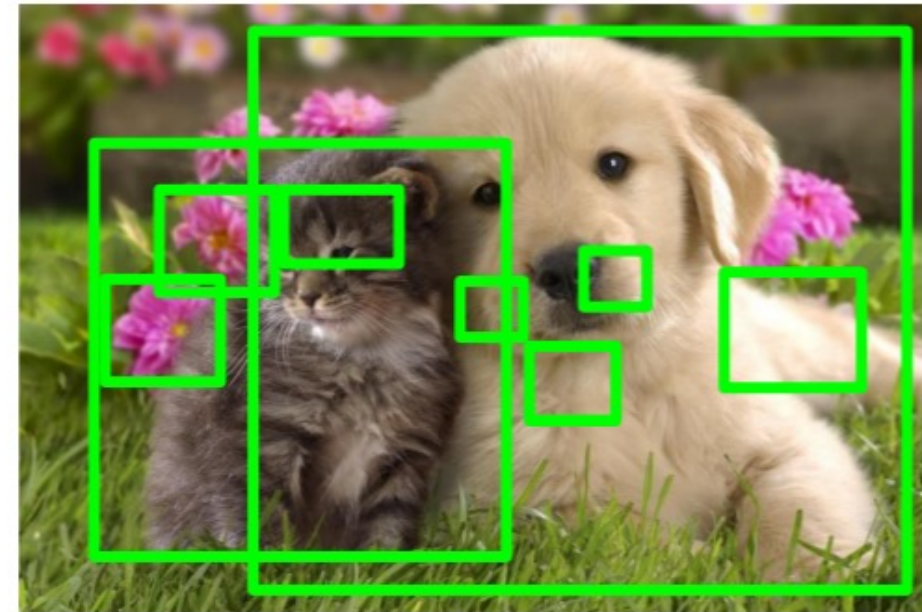


Figure copyright Ross Girshick, 2015.
Reproduced with permission.

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



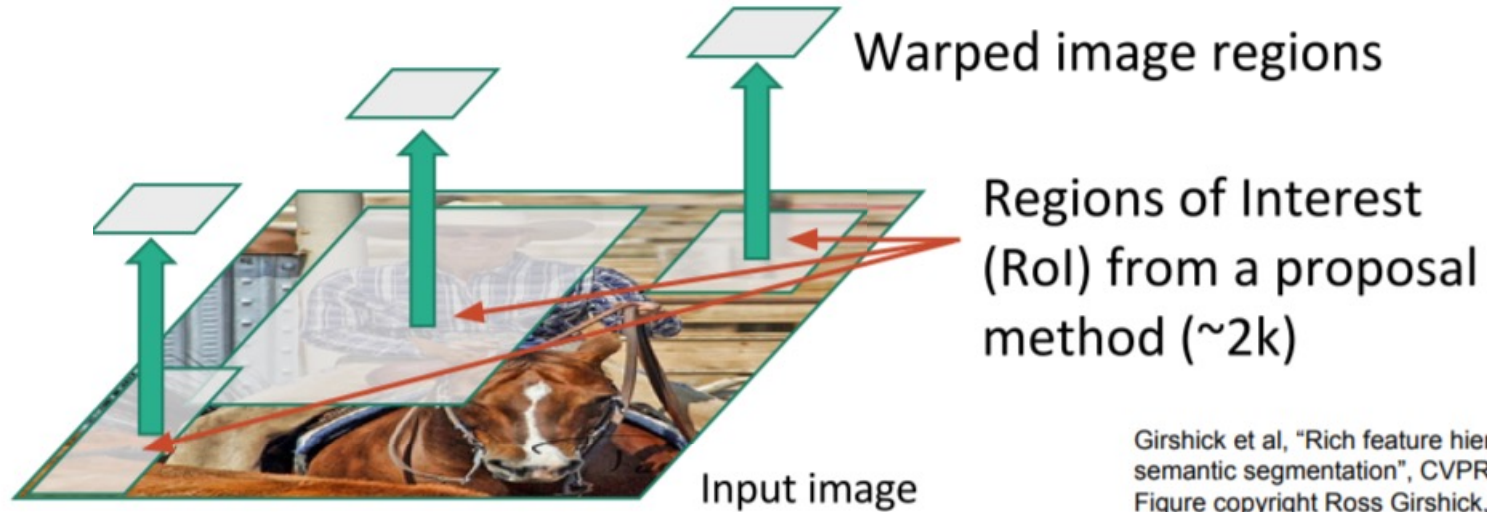
R-CNN



Input image

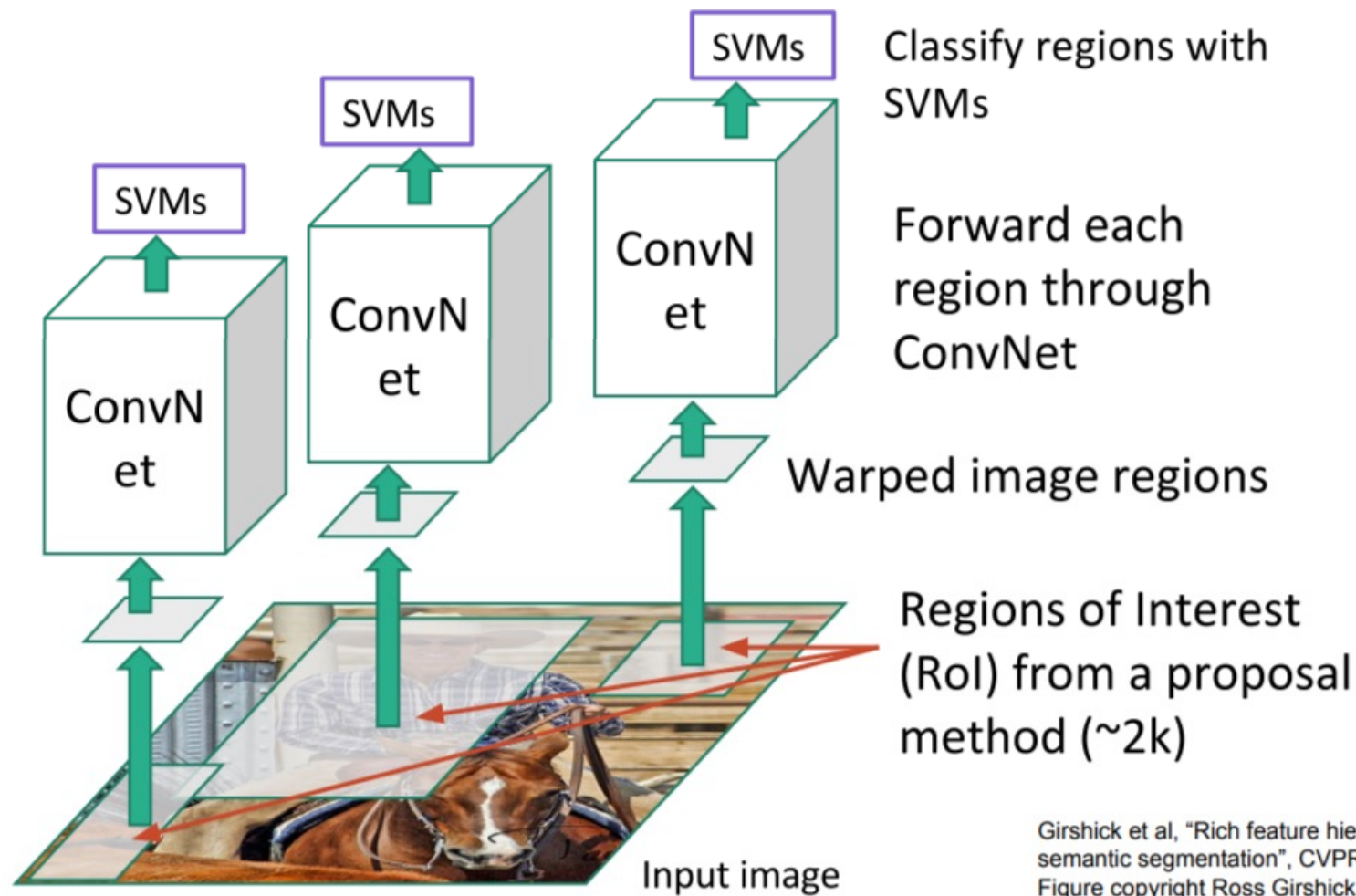
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



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

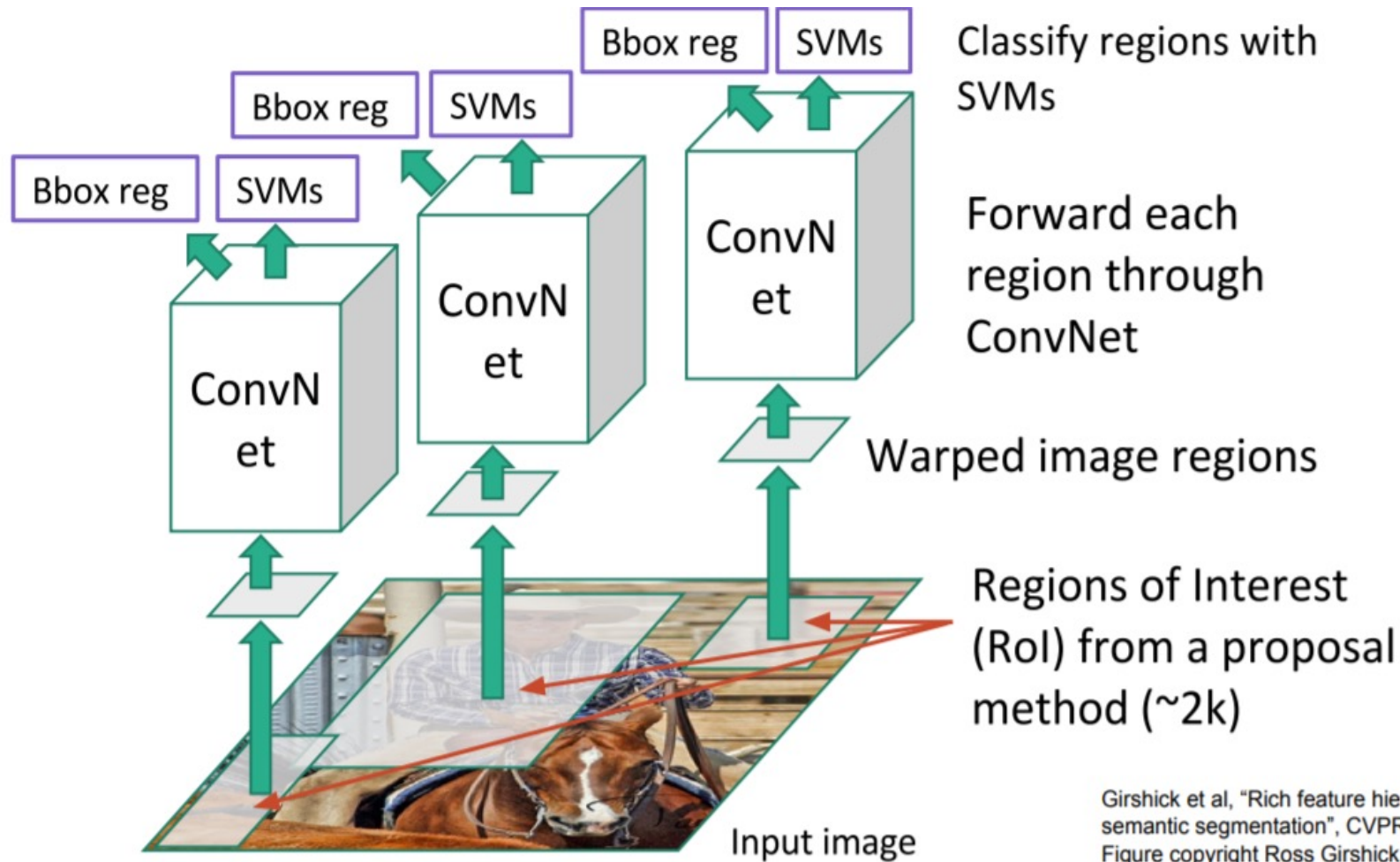


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN

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

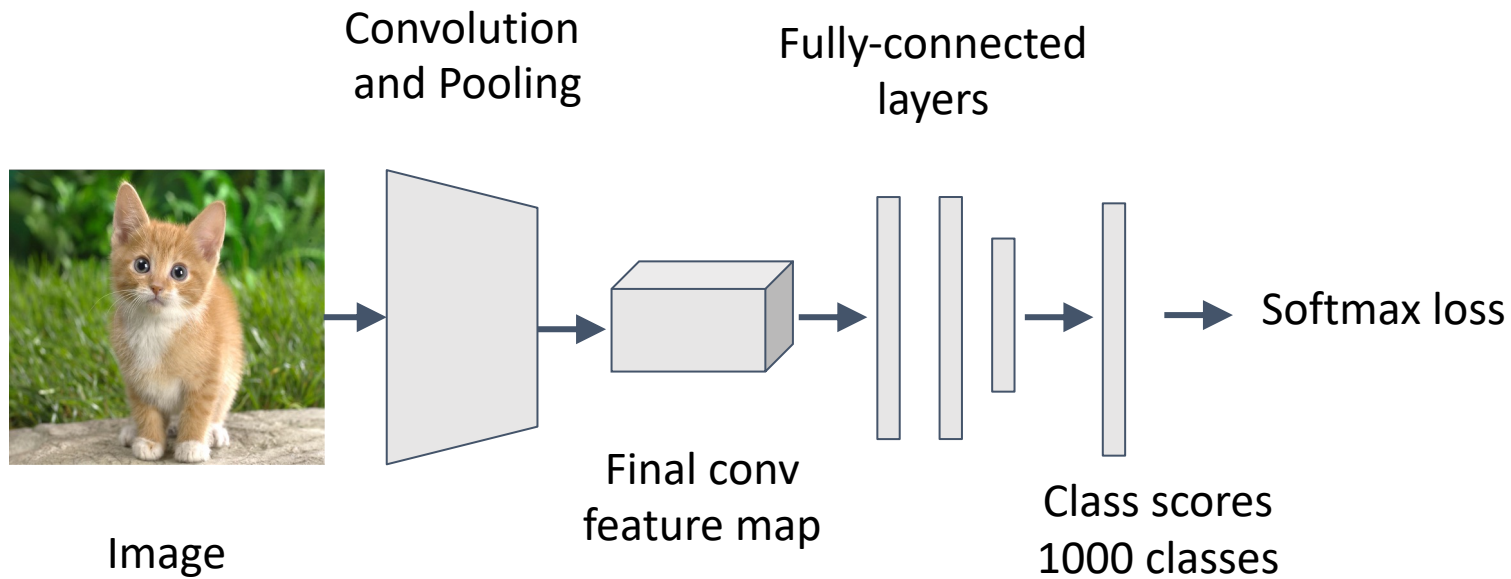
Linear Regression for bounding box



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 Training

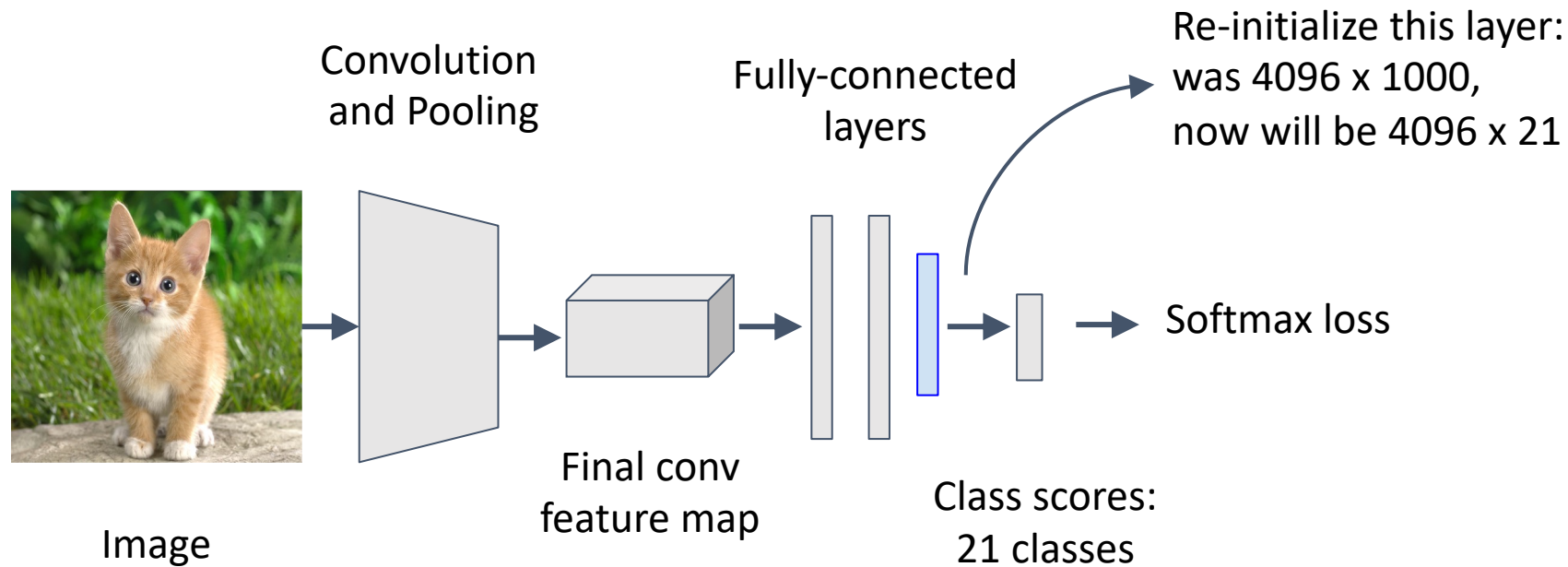
- **Step 1:** Train (or download) a classification model for ImageNet (e.g. VGG)



R-CNN Training

- **Step 2:** Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images

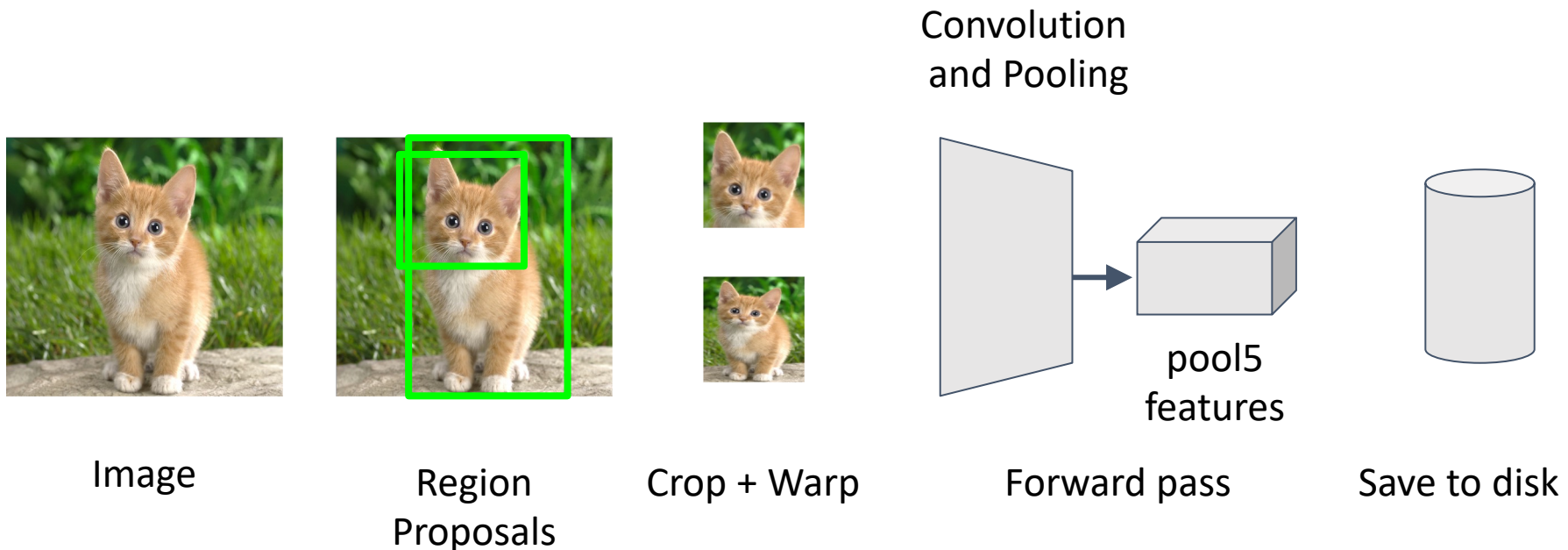


region proposals with **IoU overlap ≥ 0.5 with a ground-truth** box are considered as a sample for that box's class and the rest as background

R-CNN Training

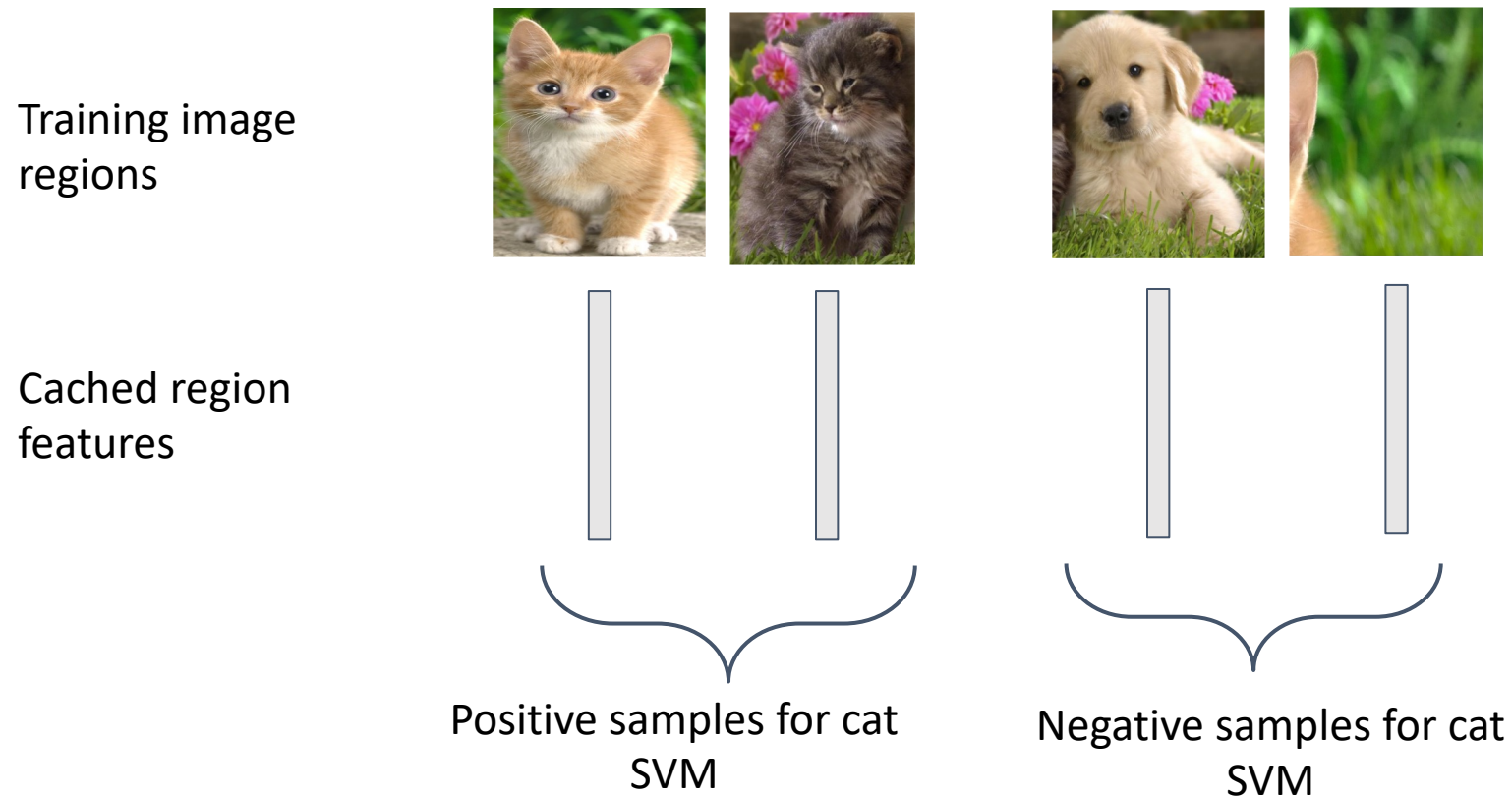
Step 3: Extract features

- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!



R-CNN Training

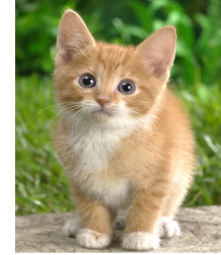
- **Step 4:** Train one binary SVM per class to classify region features



R-CNN Training

- **Step 5** (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals

Training image
regions



Cached region
features



Regression targets
(dx, dy, dw, dh)
Normalized
coordinates

(0, 0, 0, 0)
Proposal is
good

(.25, 0, 0, 0)
Proposal too
far to left

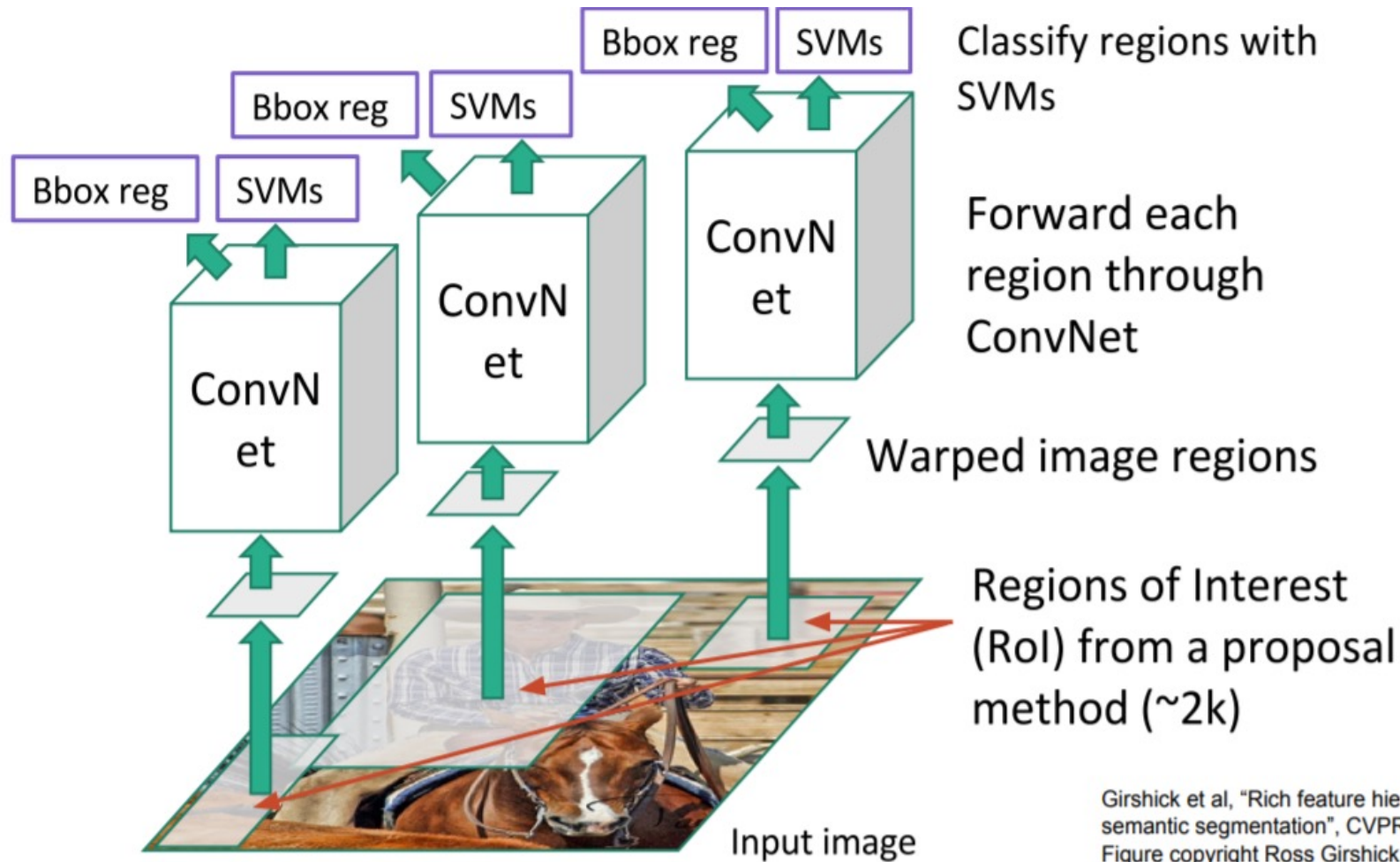
(0, 0, -0.125, 0)
Proposal too
wide

R-CNN: Problems

- Ad hoc training objectives
 - Fine-tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
 - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
 - Fixed by SPP-net [He et al. ECCV14]

R-CNN

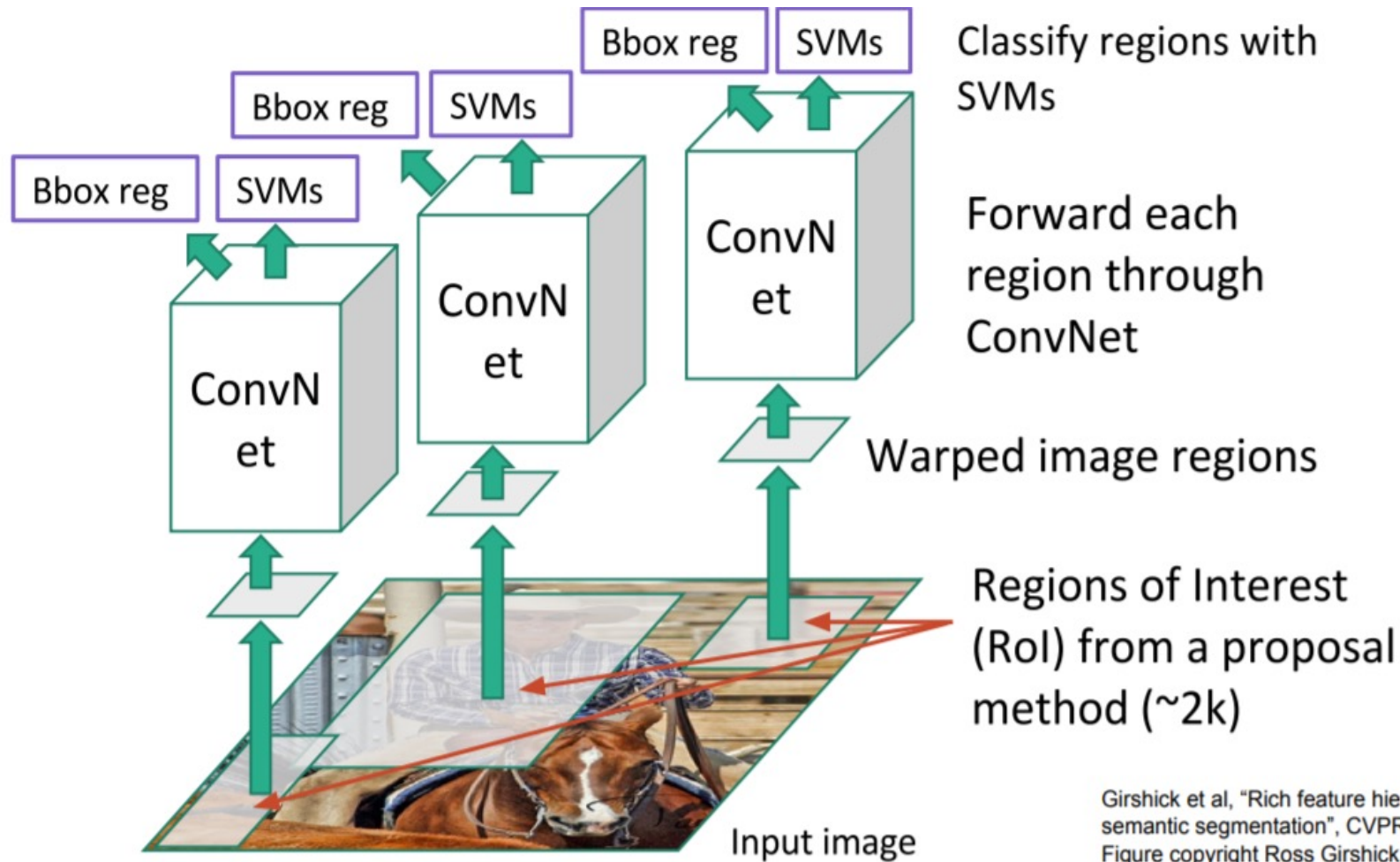
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)
Linear Regression for bounding box



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

R-CNN

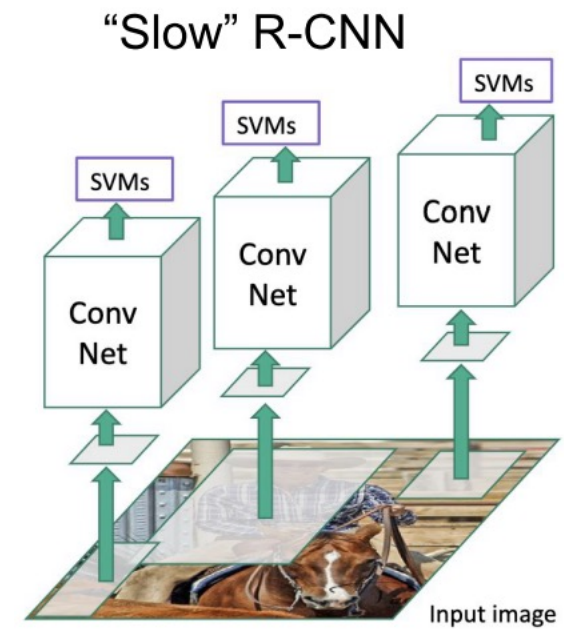
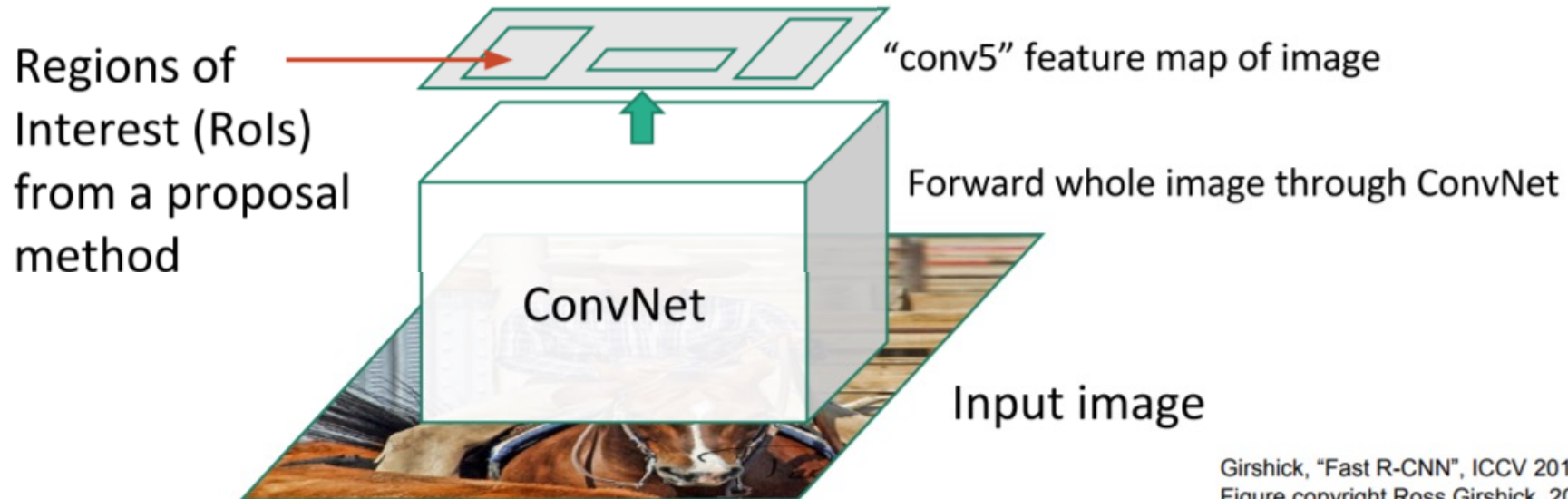
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)
Linear Regression for bounding box



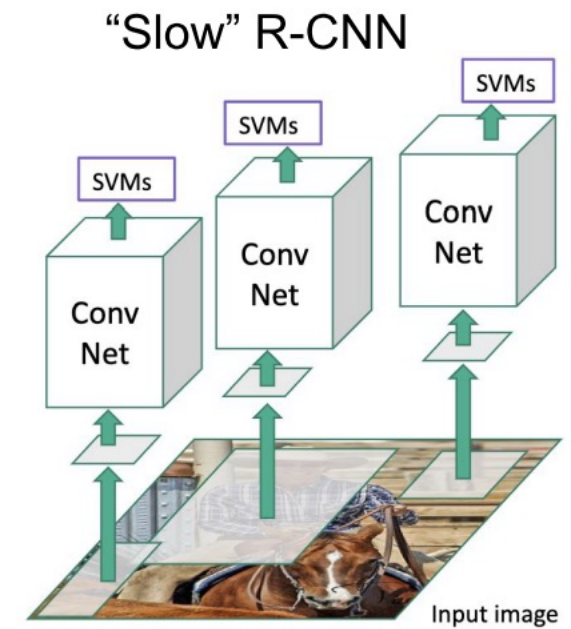
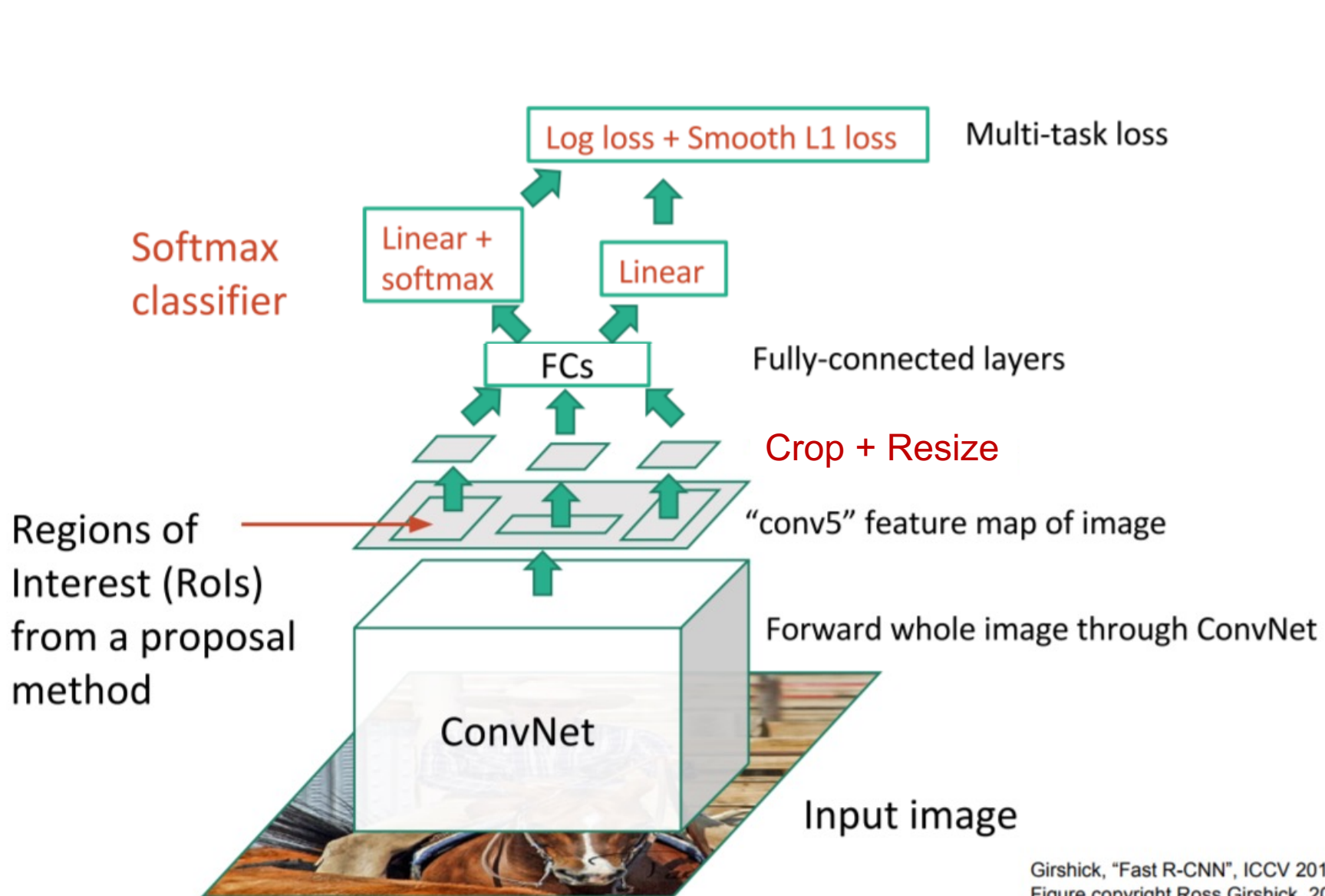
Problem: Very slow!
Need to do ~2k independent forward passes for each image!

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

Fast R-CNN

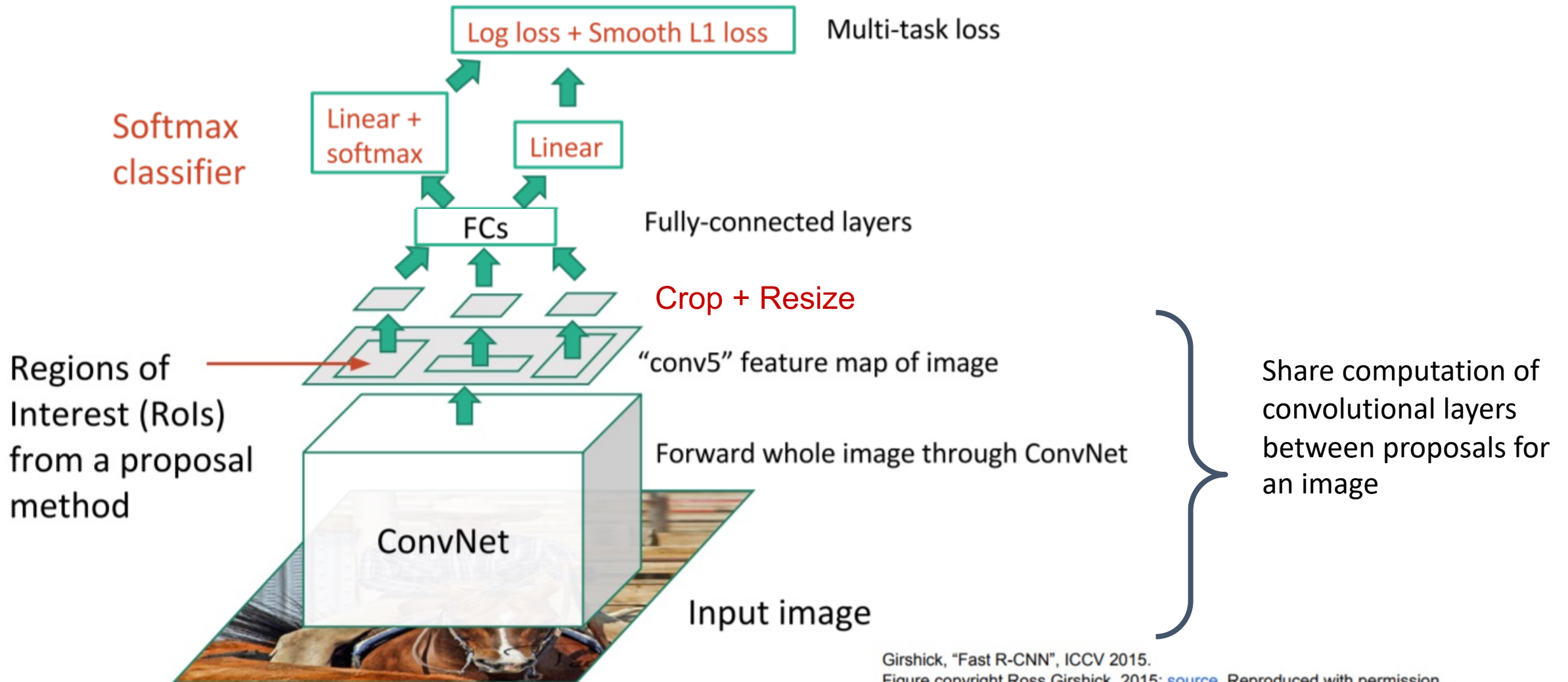


Fast R-CNN

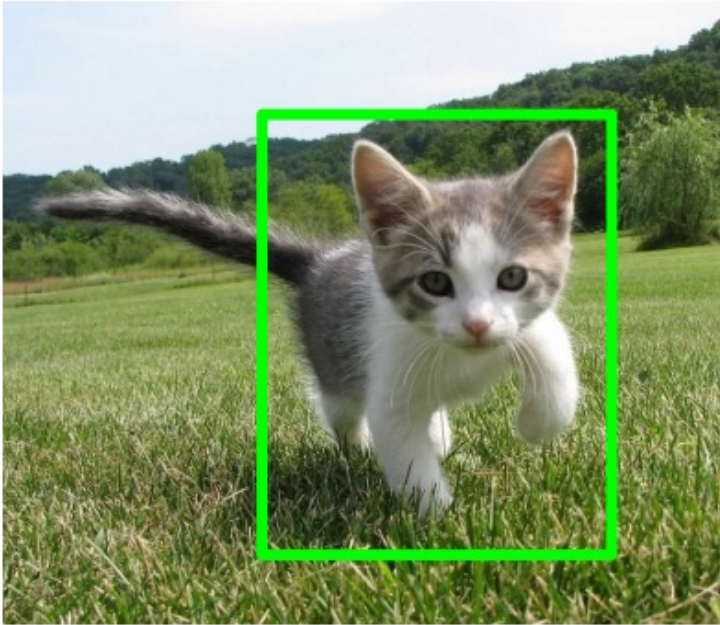


Share computation of convolutional layers between proposals for an image

Fast R-CNN



Cropping Features: RoI Pool



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

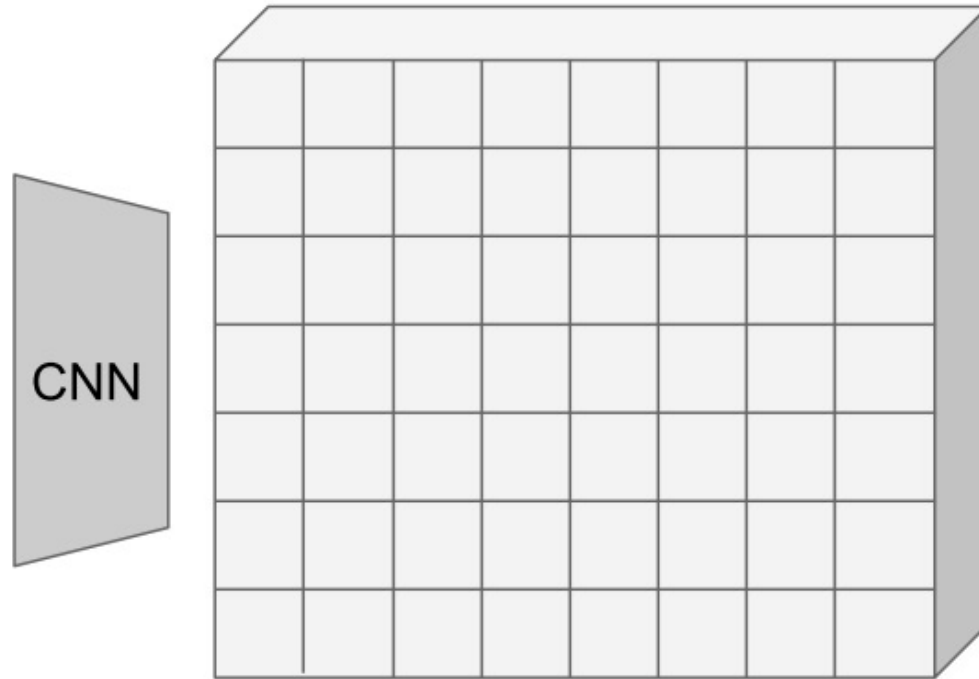
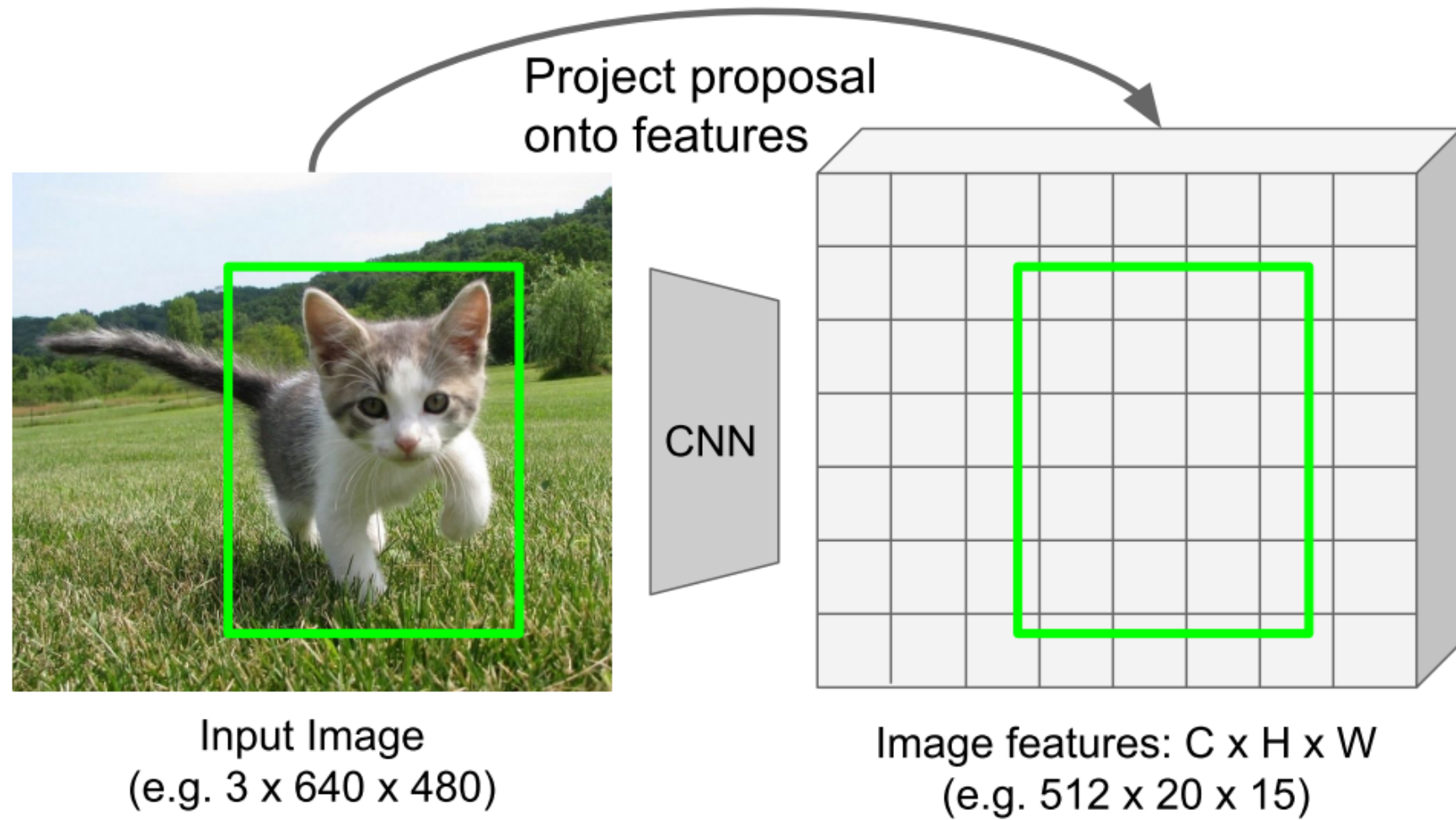
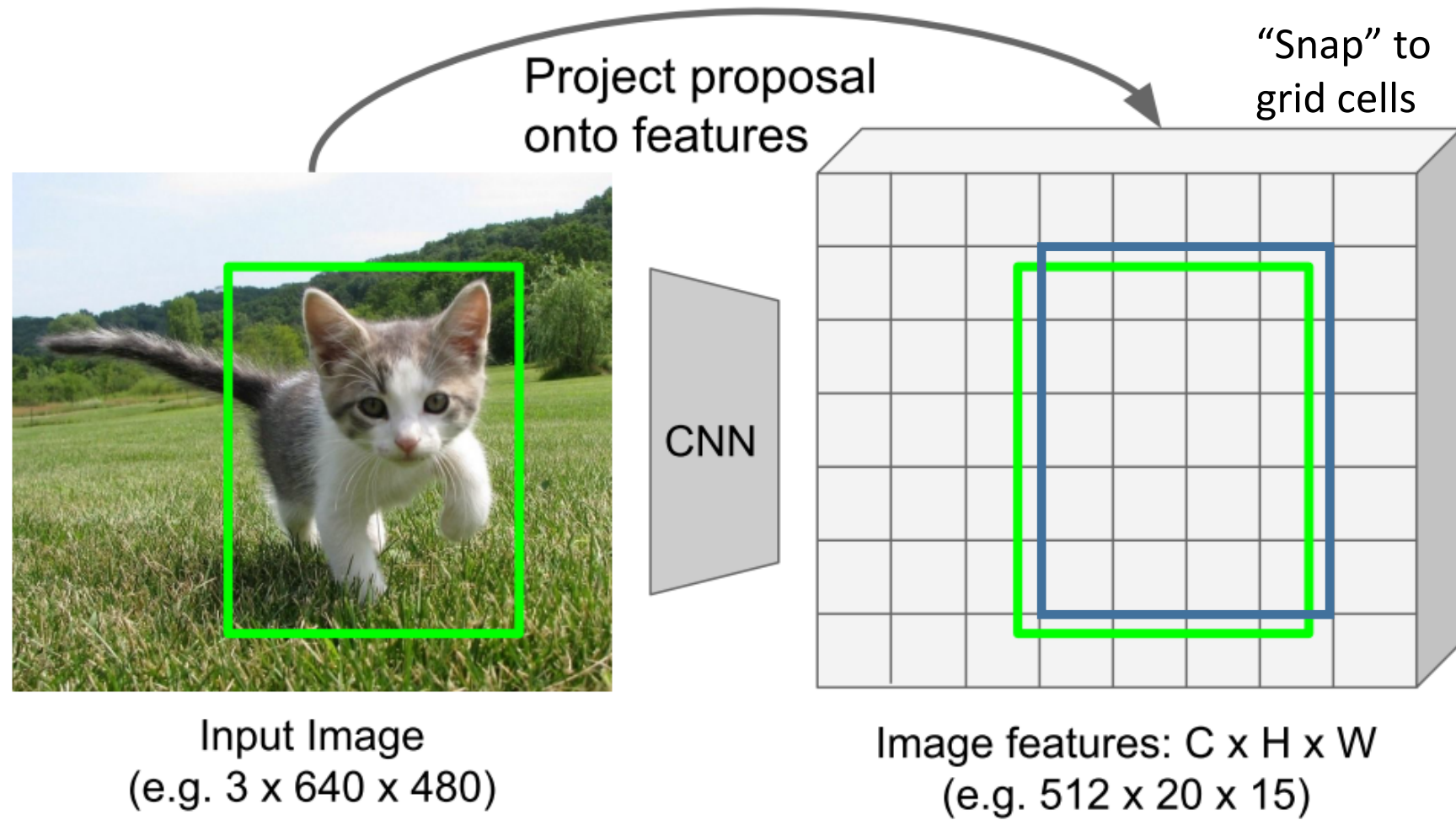


Image features: C x H x W
(e.g. 512 x 20 x 15)

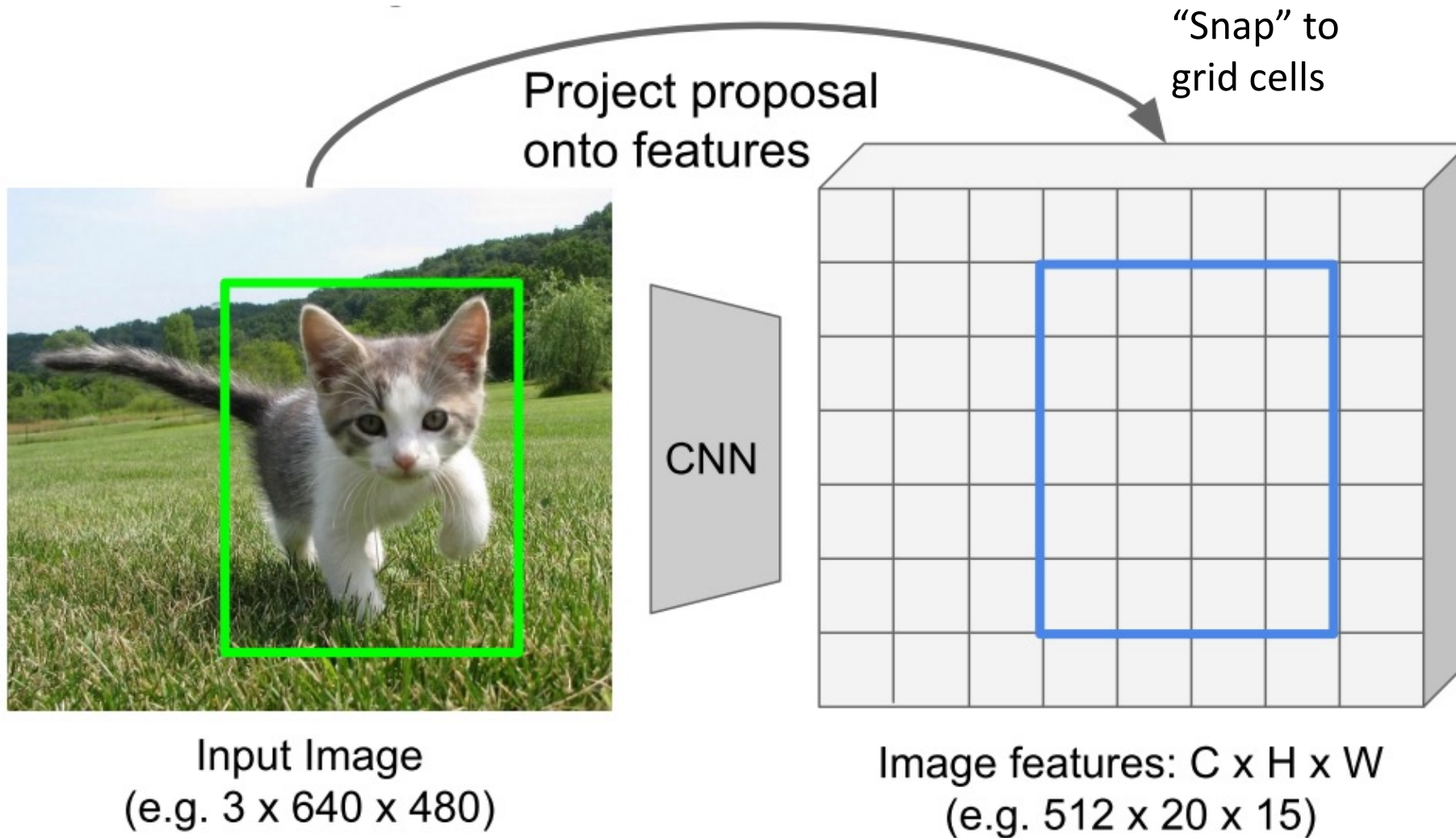
Cropping Features: RoI Pool



Cropping Features: RoI Pool

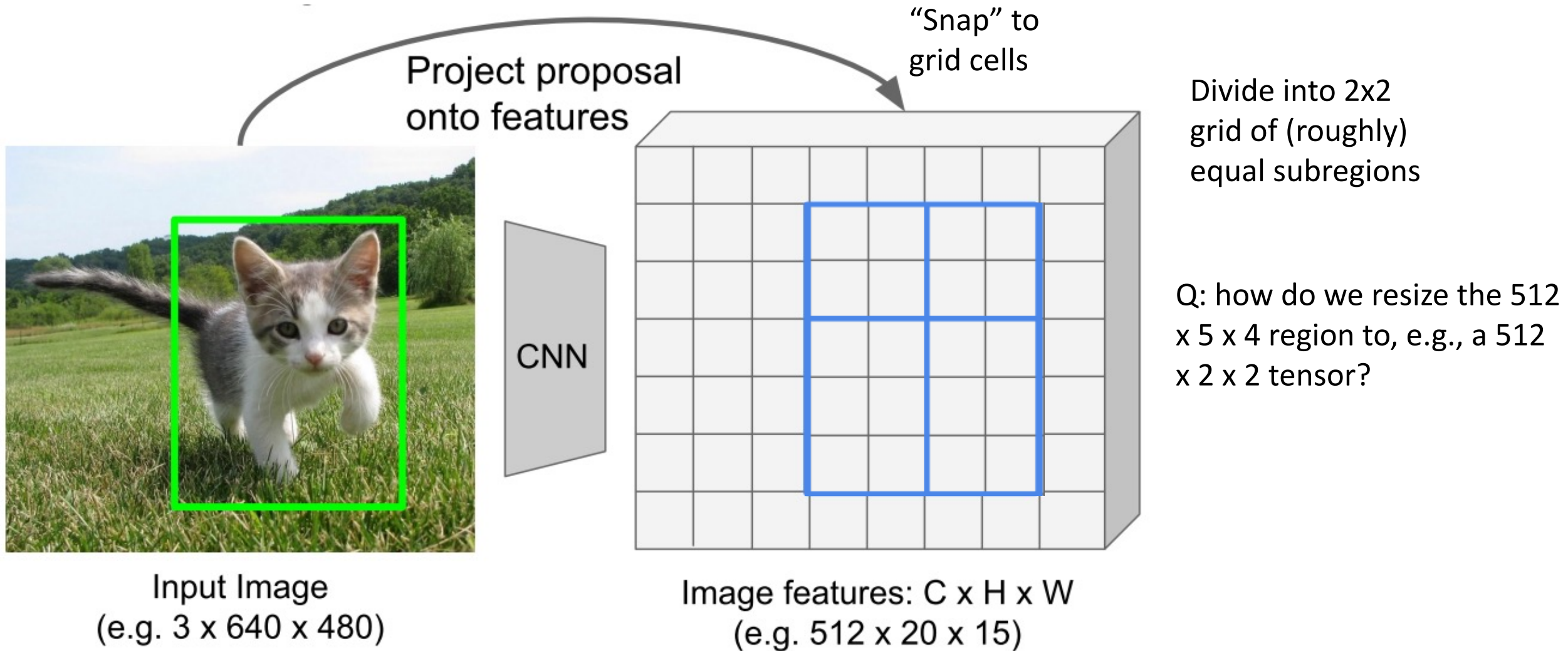


Cropping Features: RoI Pool

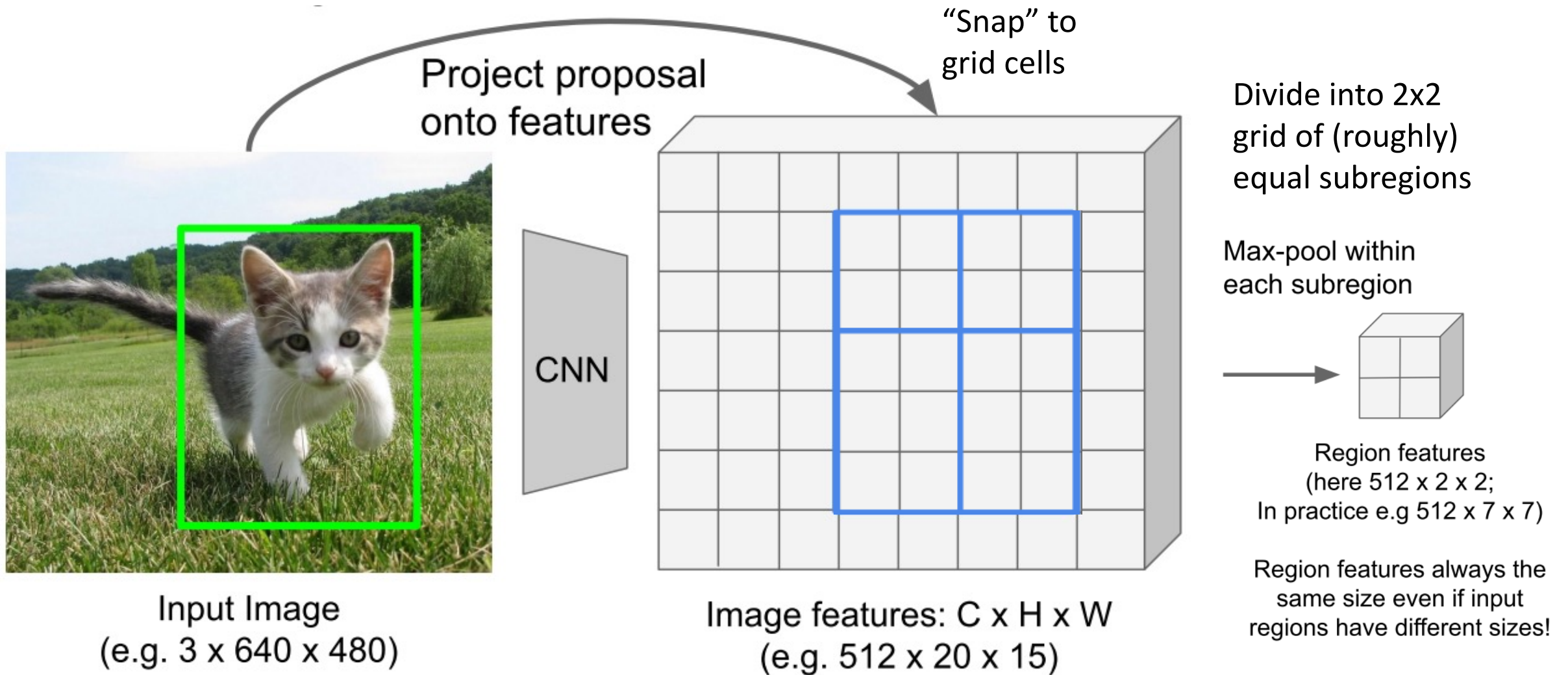


Q: how do we resize the 512 x 5 x 4 region to, e.g., a 512 x 2 x 2 tensor?

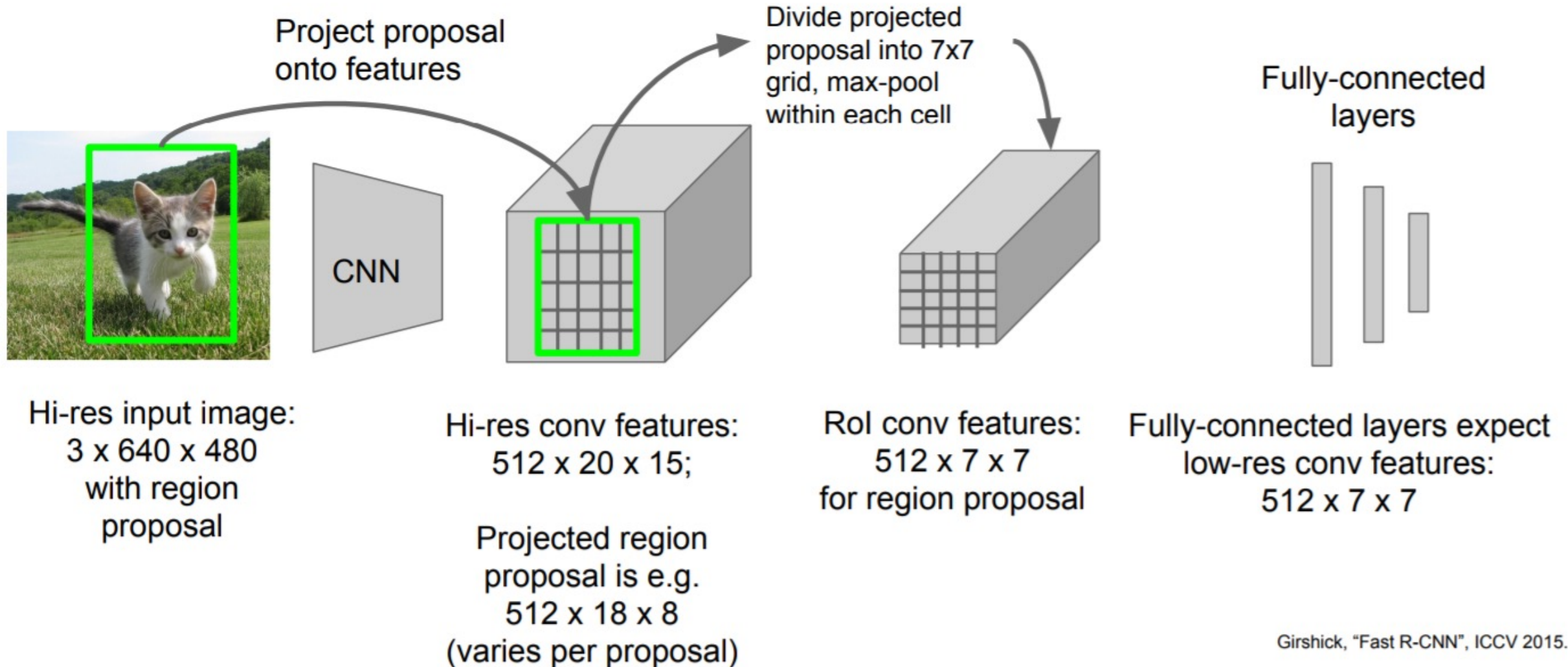
Cropping Features: RoI Pool



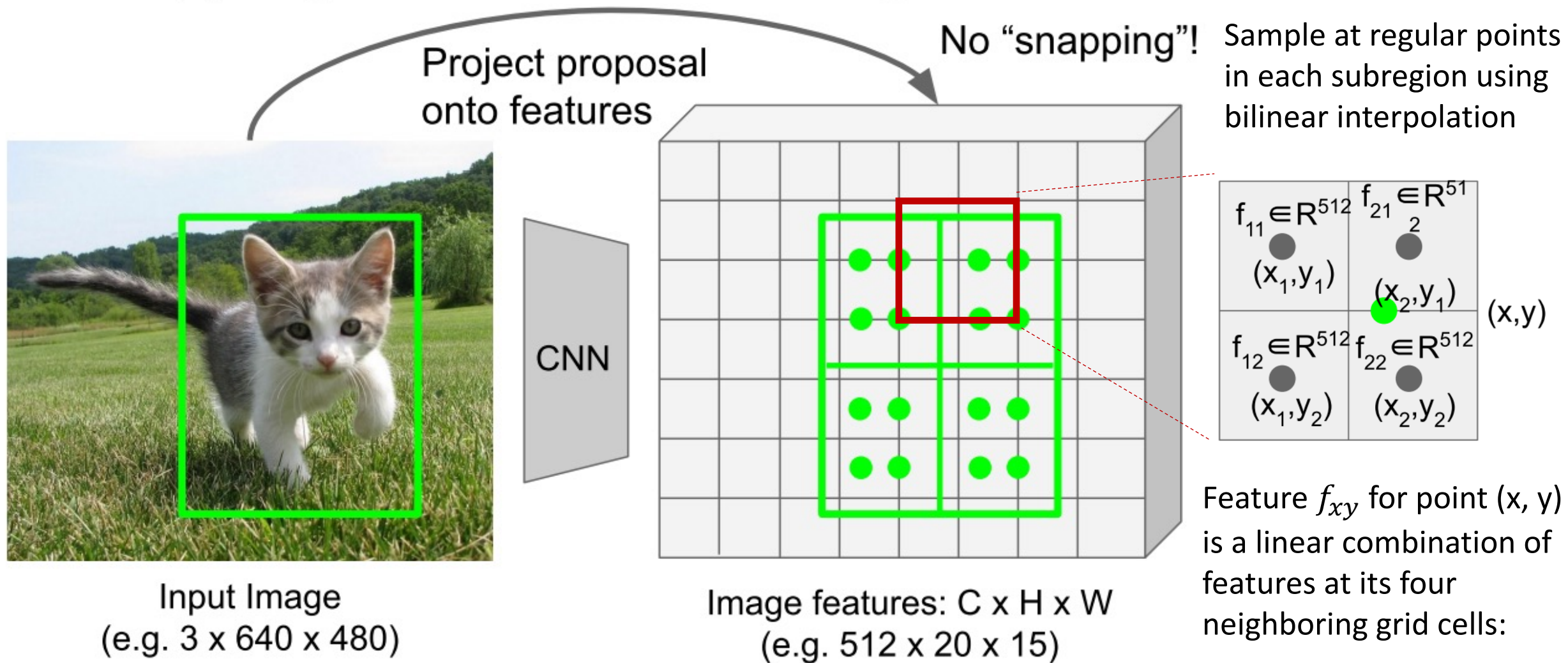
Cropping Features: RoI Pool



Fast R-CNN: RoI Pooling

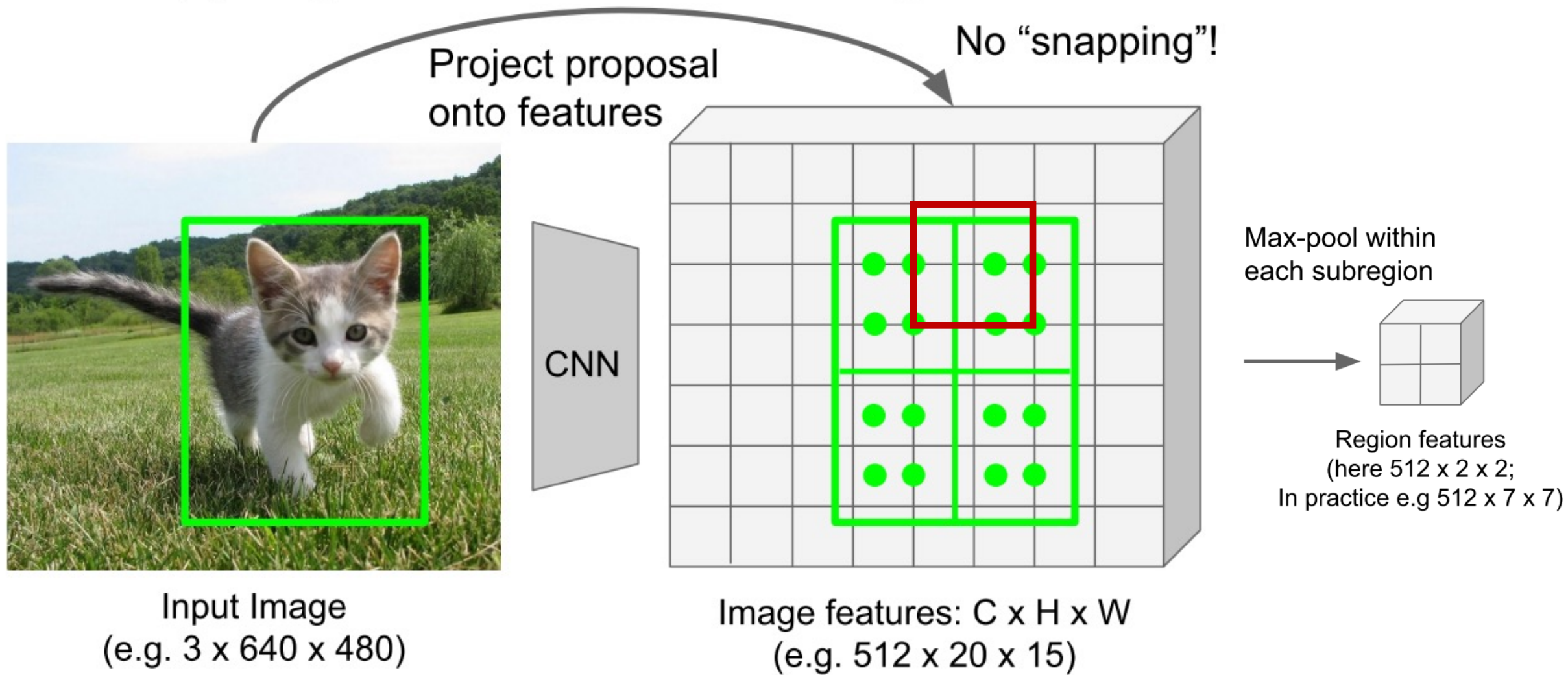


Cropping Features: RoI Align



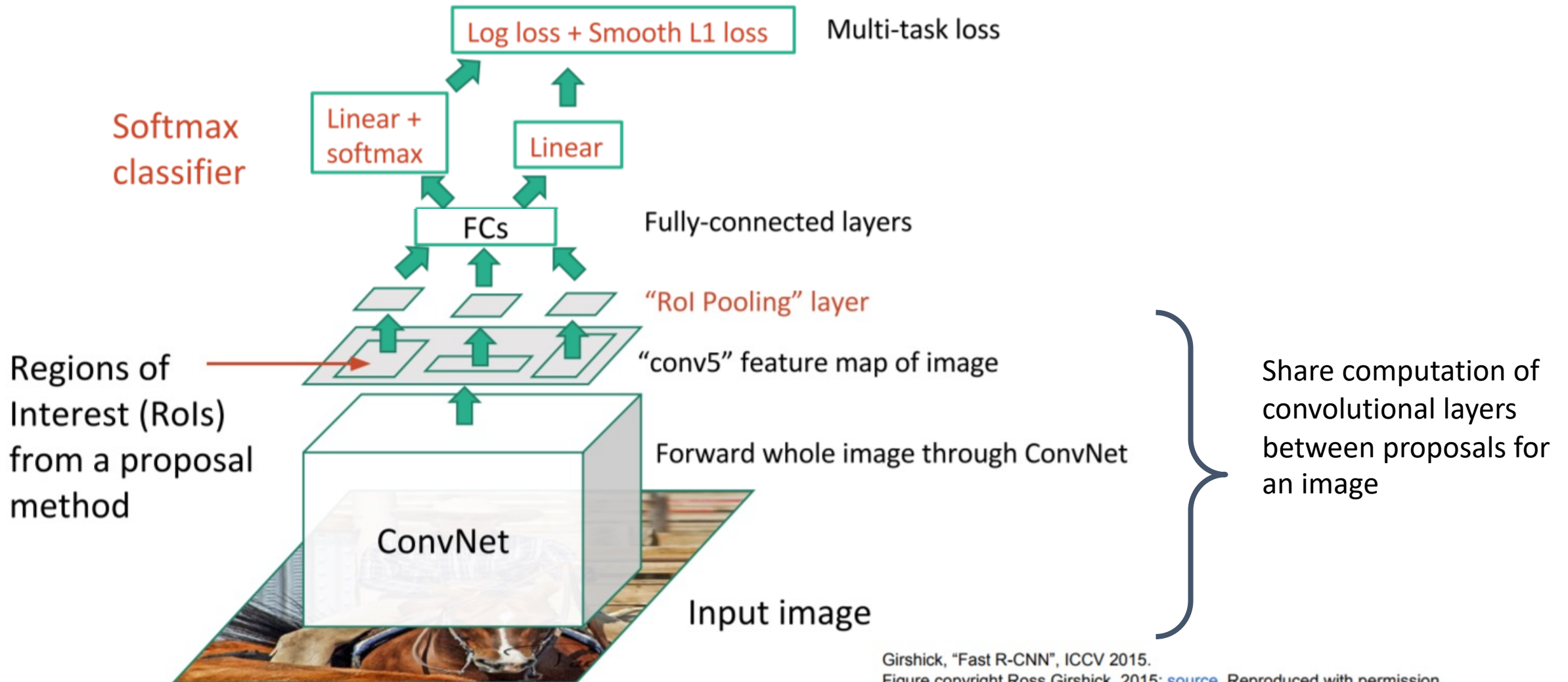
$$f_{xy} = \sum_{i,j=1}^2 f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

Cropping Features: RoI Align



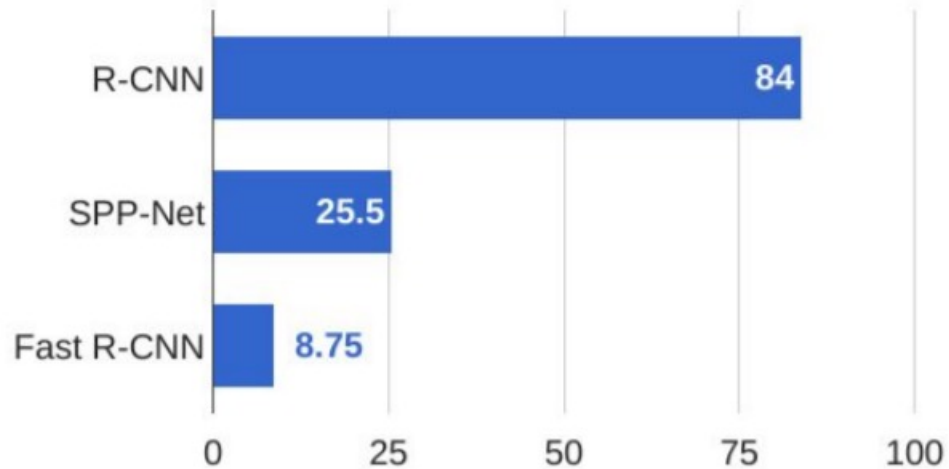
$$f_{xy} = \sum_{i,j=1}^2 f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

Fast R-CNN

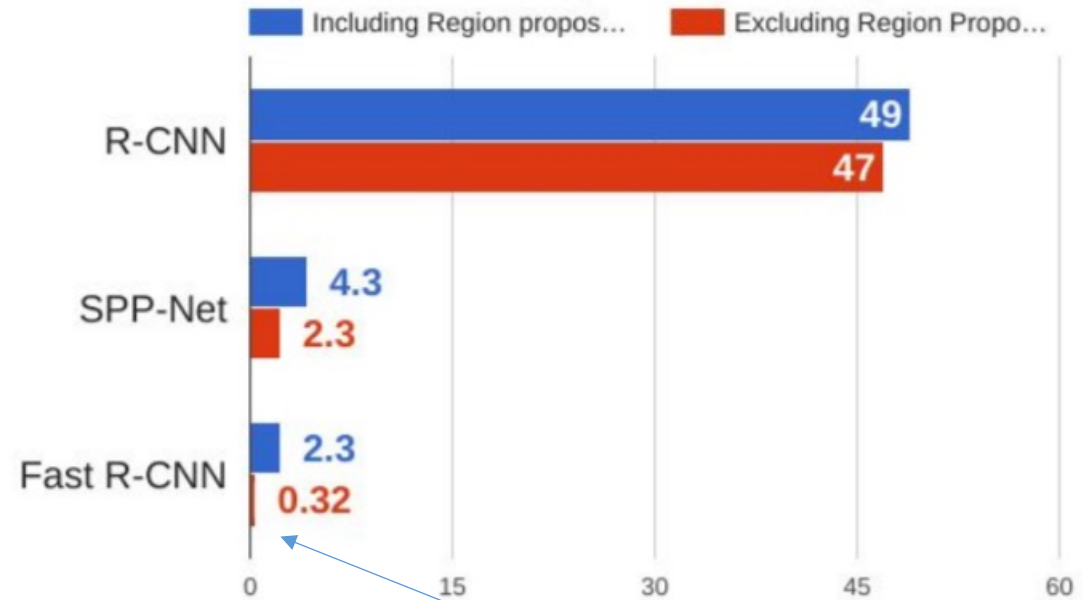


R-CNN vs SPP vs Fast R-CNN

Training time (Hours)



Test time (seconds)

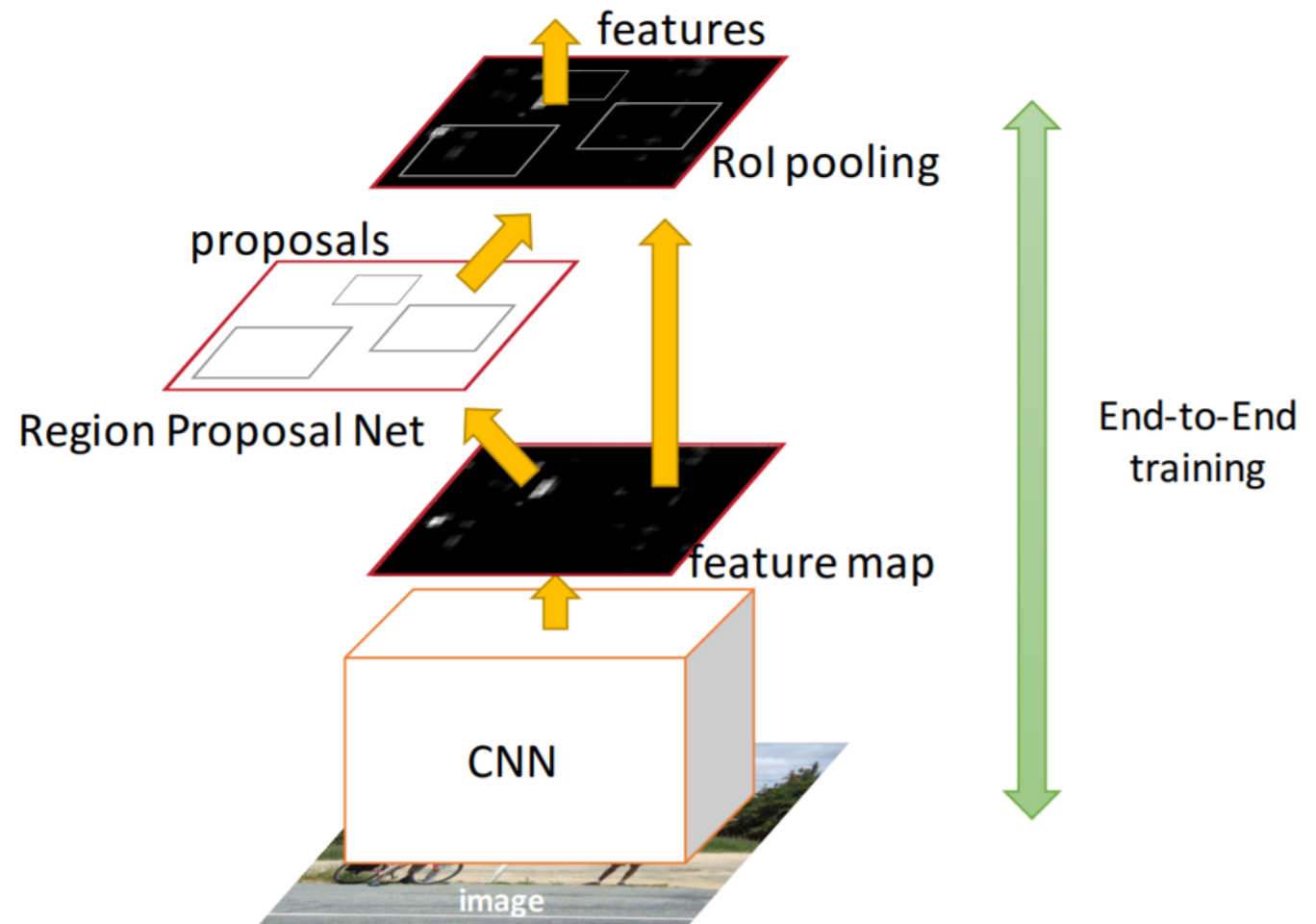


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Problem: Runtime dominated by region proposals!

Faster R-CNN

- Make CNN do proposals!
 - Solely based on CNN
 - No external modules
- Each step is end-to-end



Region Proposal Network



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

CNN

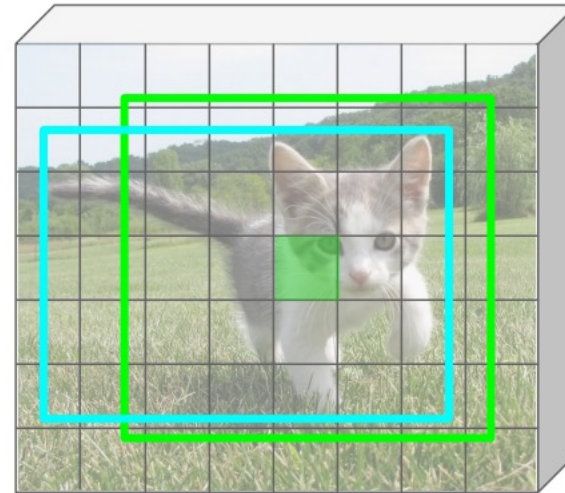


Image features
(e.g. 512 x 20 x 15)

Conv

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

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

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

Region Proposal Network



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

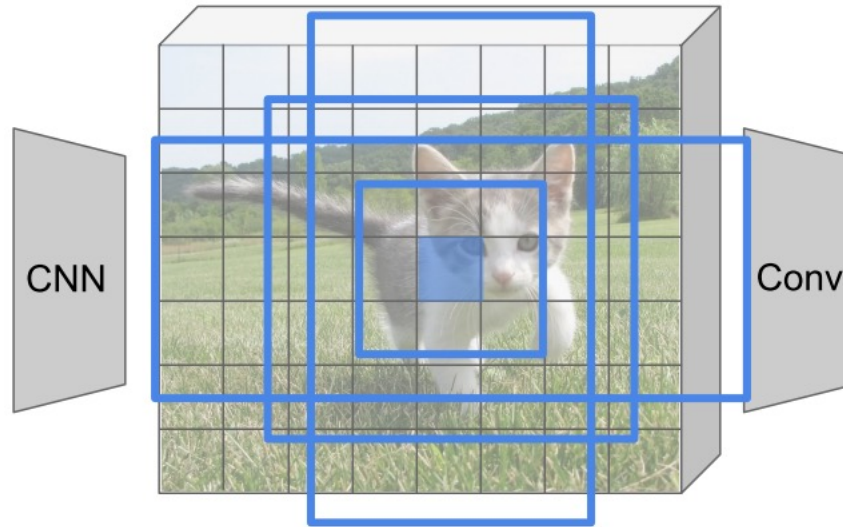


Image features
(e.g. 512 x 20 x 15)

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

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

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

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

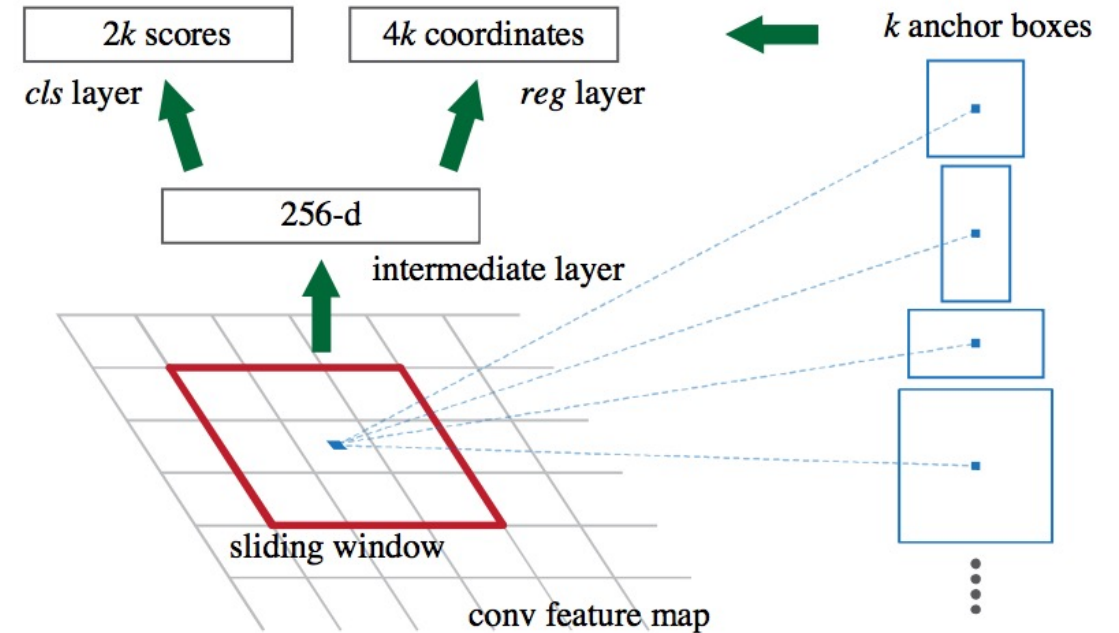
Faster R-CNN: Region Proposal Network

Use **K anchor boxes** at each location
(e.g. $K=3 \times 3$ showing boxes of 3 different sizes and 3 different aspect ratio at that location)

Mini-network operates in a sliding-window fashion (shared across all spatial locations)

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



Faster R-CNN: Region Proposal Network

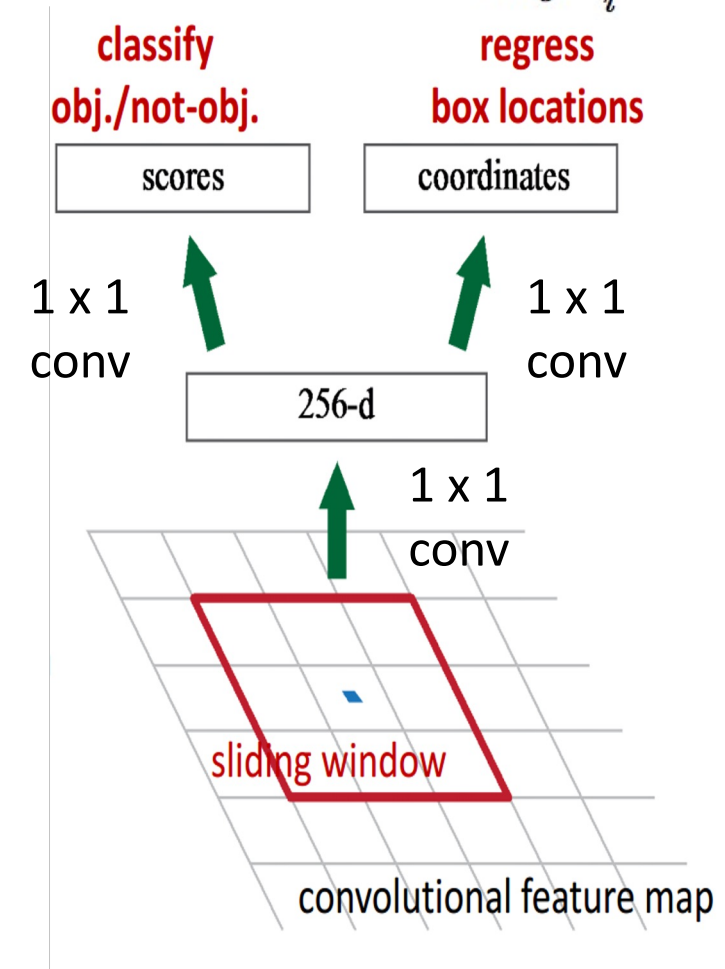
Slide a small window on the feature map

Build a small network for:

- classifying object or not-object, and
- regressing bbox locations

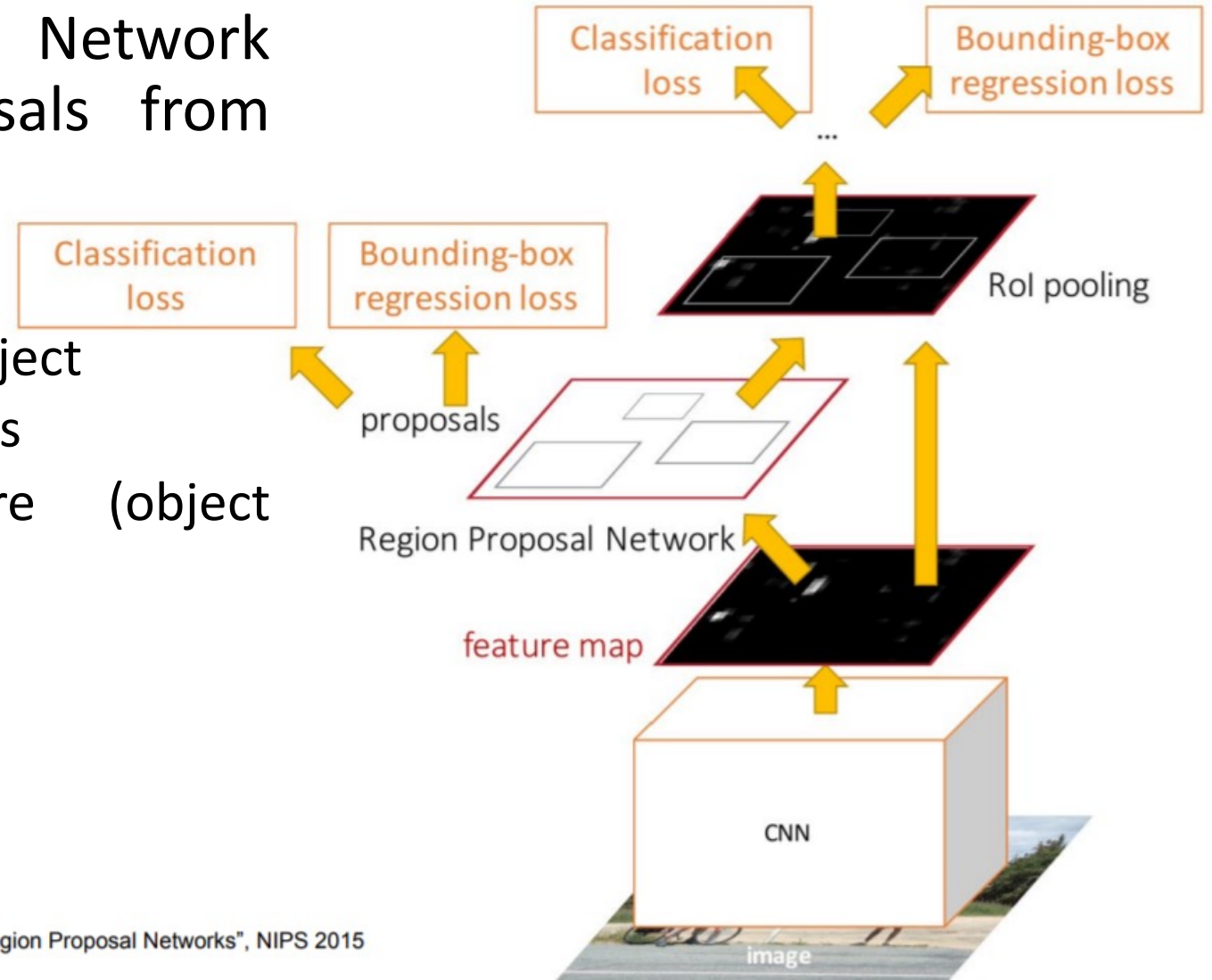
Box regression provides finer localization information with reference to the sliding window

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

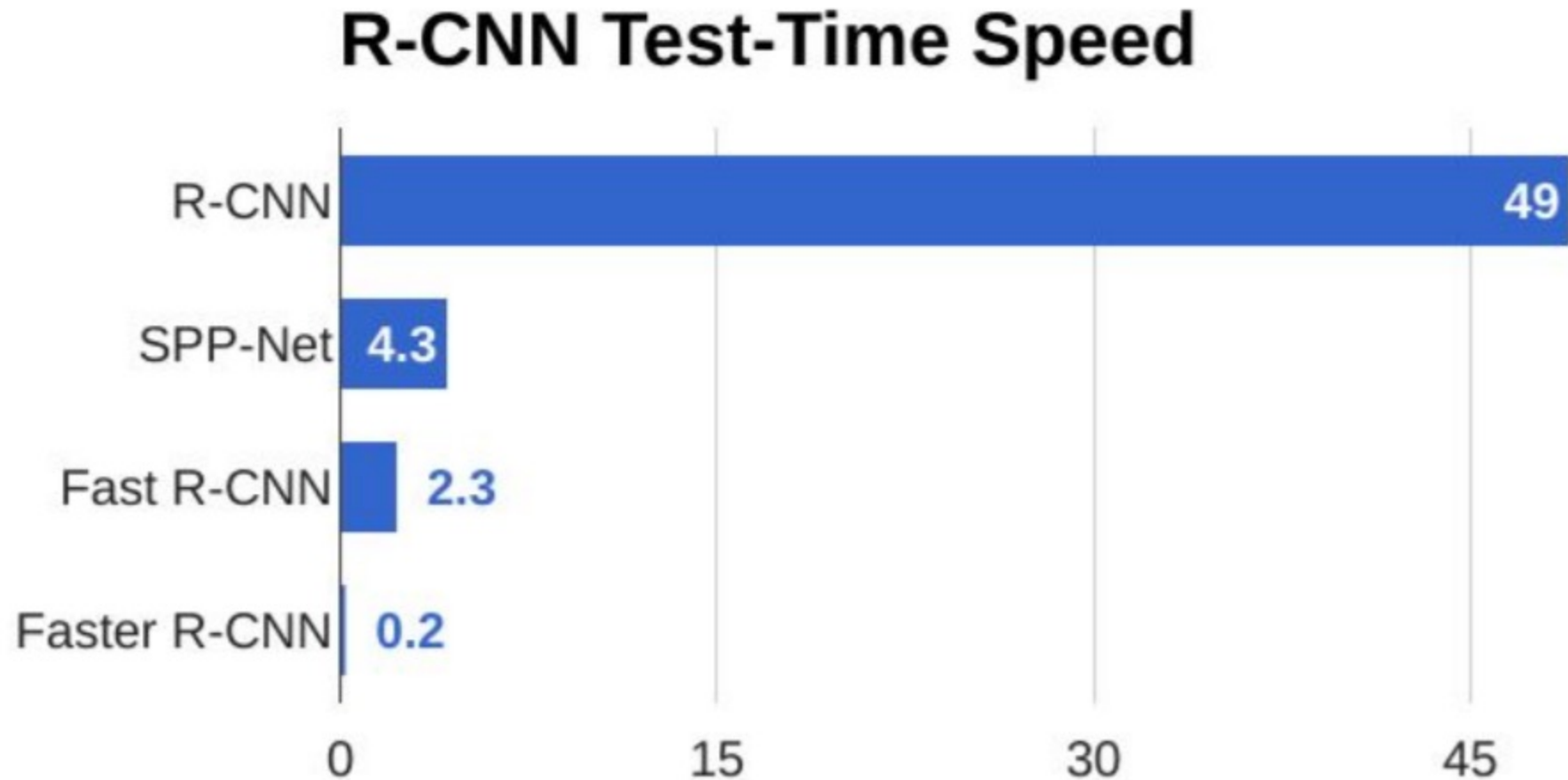


Faster R-CNN

- Insert Region Proposal Network (RPN) to predict proposals from features Jointly
- Train with 4 losses:
 - RPN classify object / not object
 - RPN regress box coordinates
 - Final classification score (object classes)
 - Final box coordinates

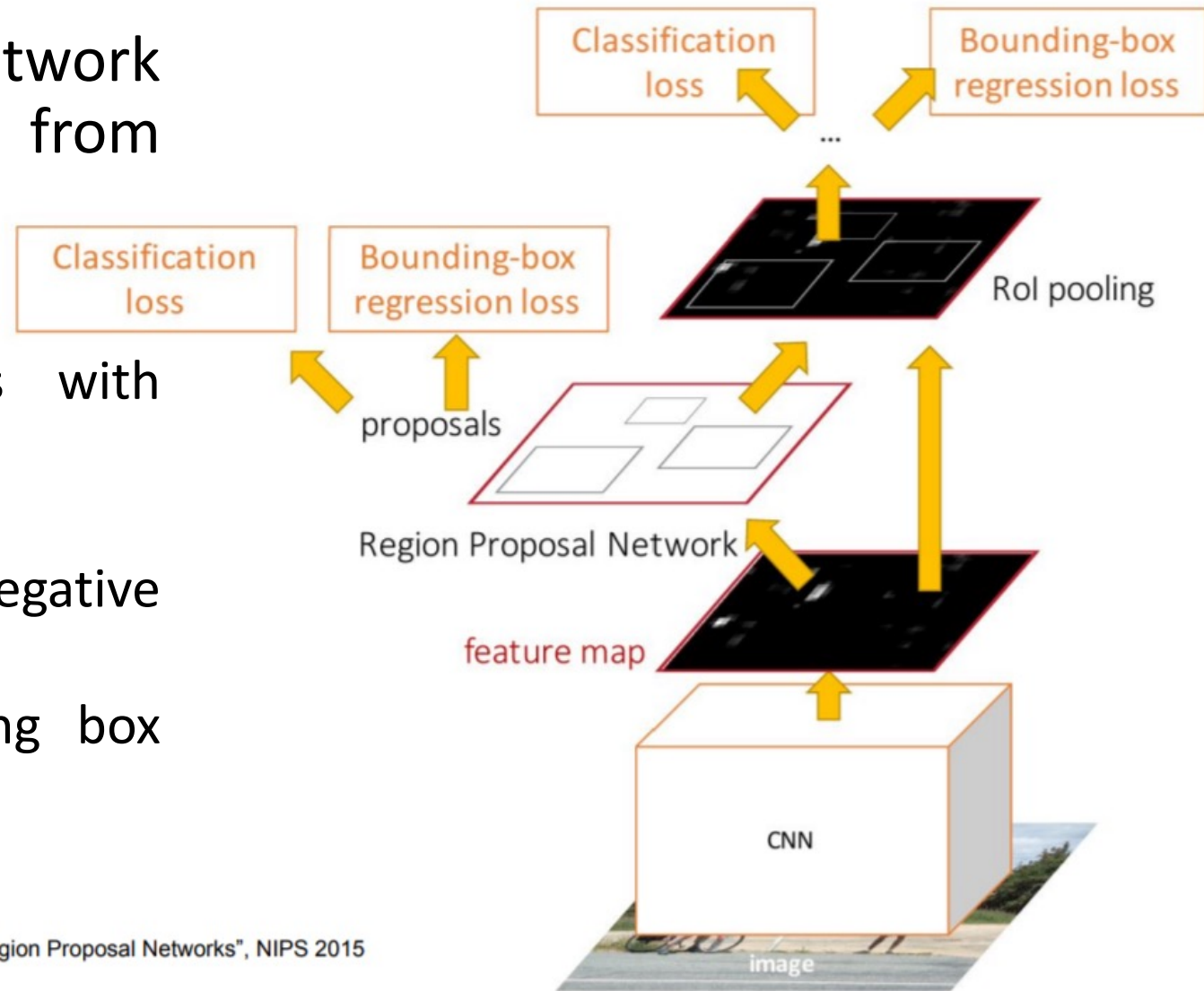


Faster R-CNN: Make CNN do proposals!



Faster R-CNN

- Insert Region Proposal Network (RPN) to predict proposals from features Jointly
- Glossing over many details:
 - Ignore overlapping proposals with **non-max suppression**
 - How are anchors determined?
 - How do we sample positive / negative samples for training the RPN?
 - How to parameterize bounding box regression?



Faster R-CNN

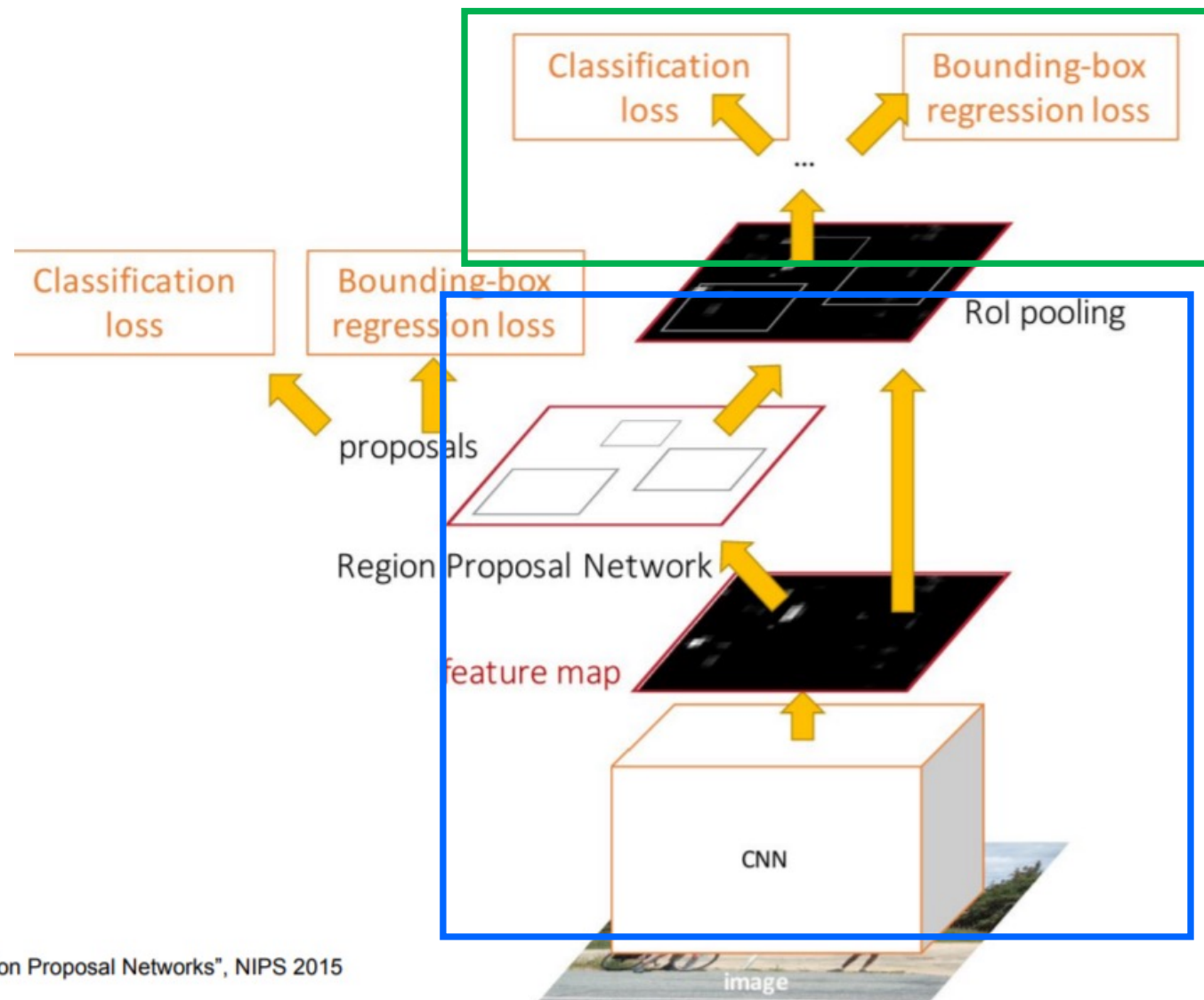
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

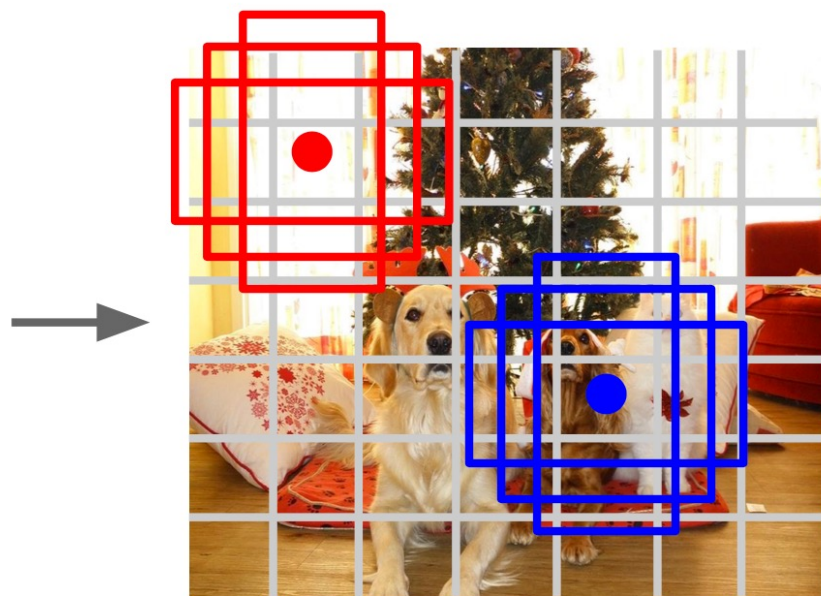
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Detection without Proposals: YOLO/SSD



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

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

Within each grid cell:

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

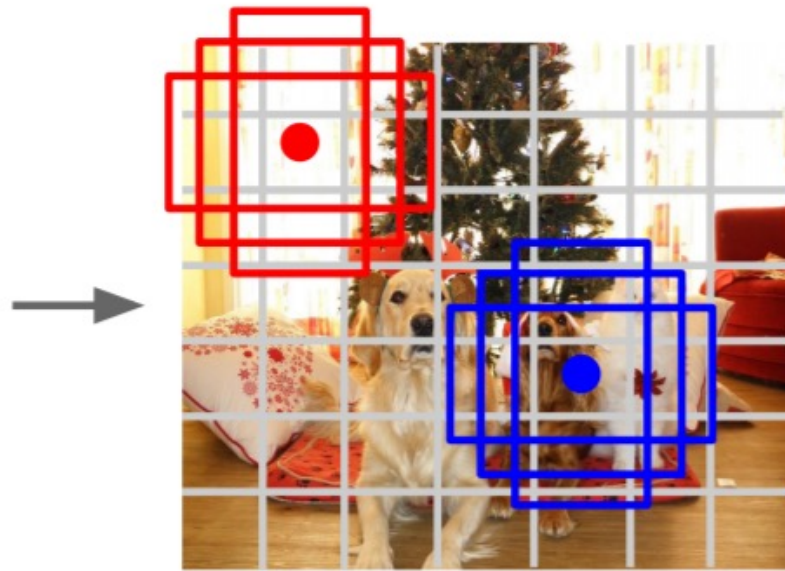
Output:
 $7 \times 7 \times (5 * B + C)$

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network! →



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

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

Within each grid cell:

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

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

SSD is much faster but not as
accurate as faster R-NN

Object Detection: Lots of Variables ...

Base Network

VGG16

ResNet-101

Inception V2

Inception V3

Inception

ResNet

MobileNet

Object Detection architecture

Faster R-CNN

R-FCN

SSD

Image Size

Region Proposals

...

Takeaways

Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

[Zou et al, "Object Detection in 20 Years: A Survey", arXiv 2019](#)

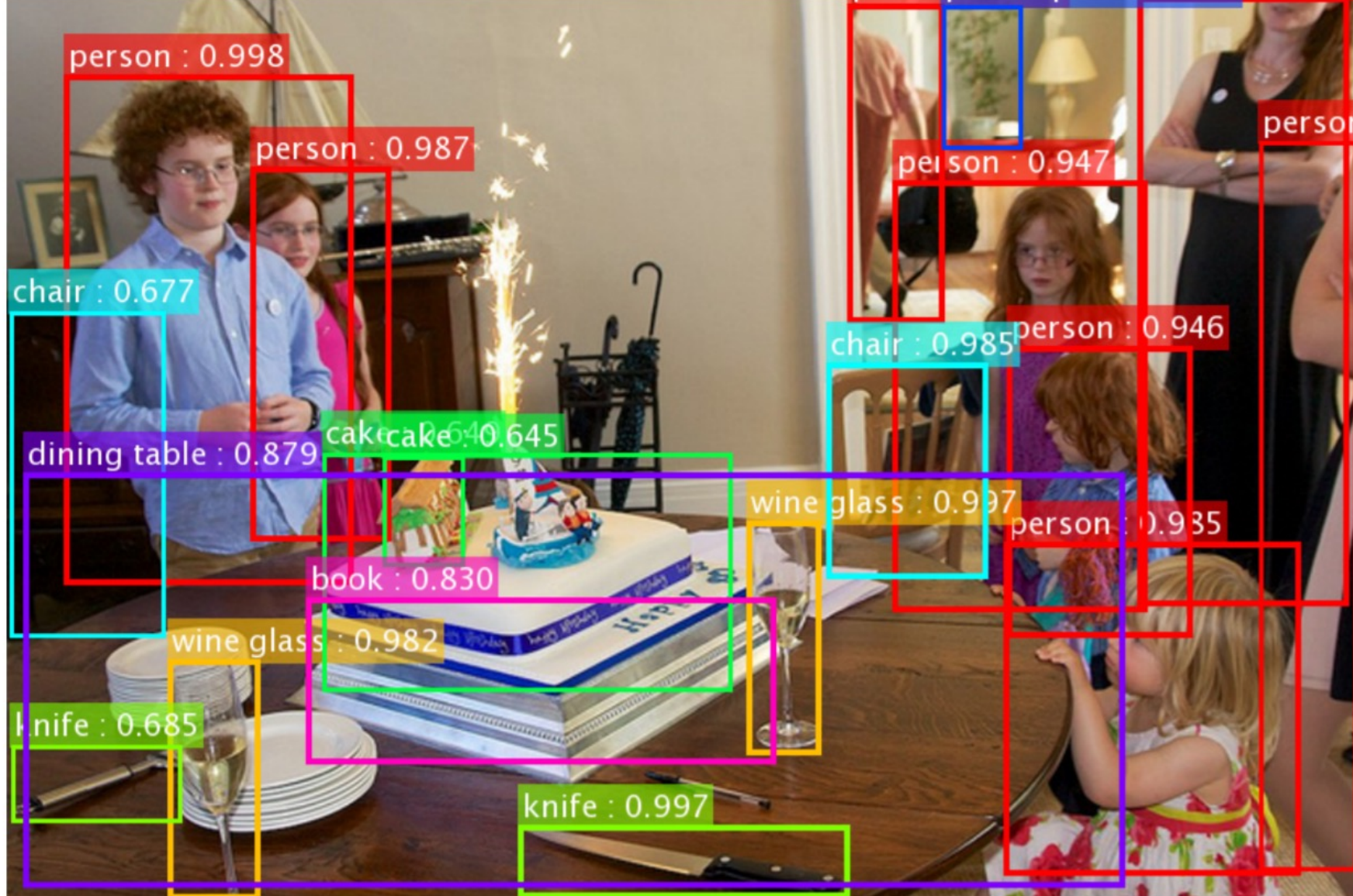
[Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017](#)

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



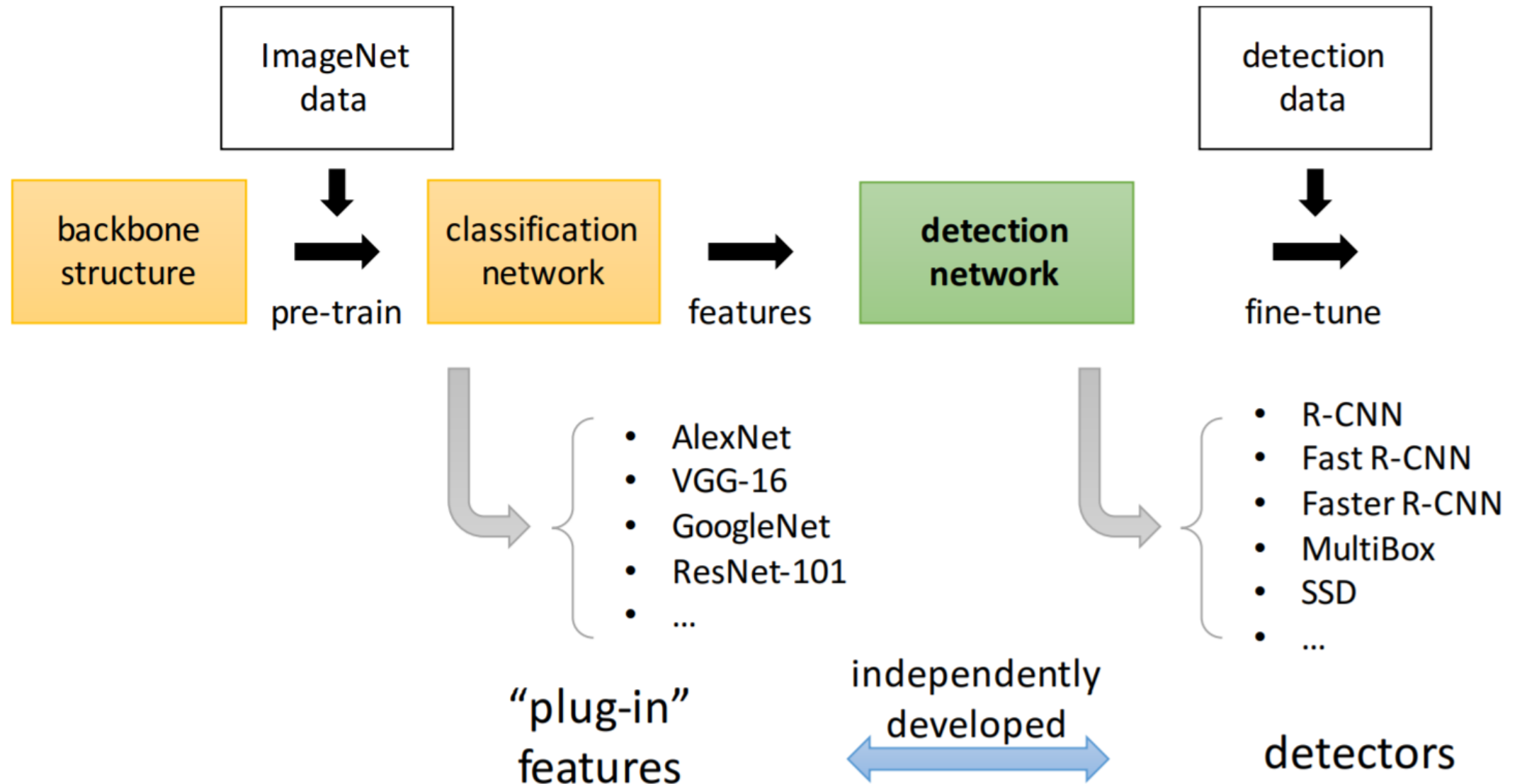
ResNet's object detection result on COCO

*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

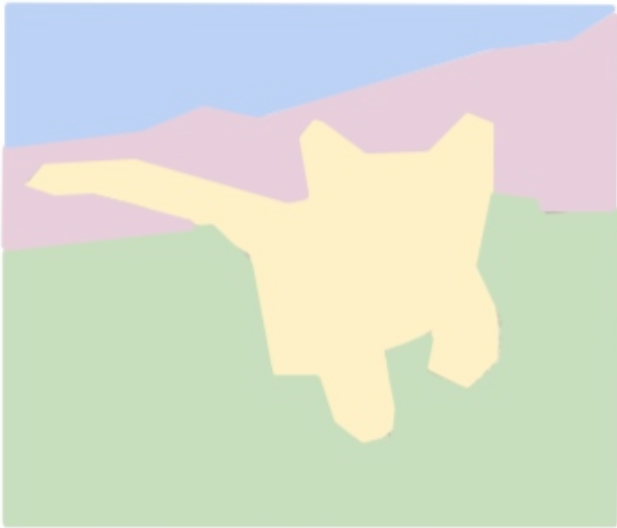
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

Object Detection



Instance Segmentation

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

**Classification
+ Localization**



CAT

Single Object

**Object
Detection**



DOG, DOG, CAT

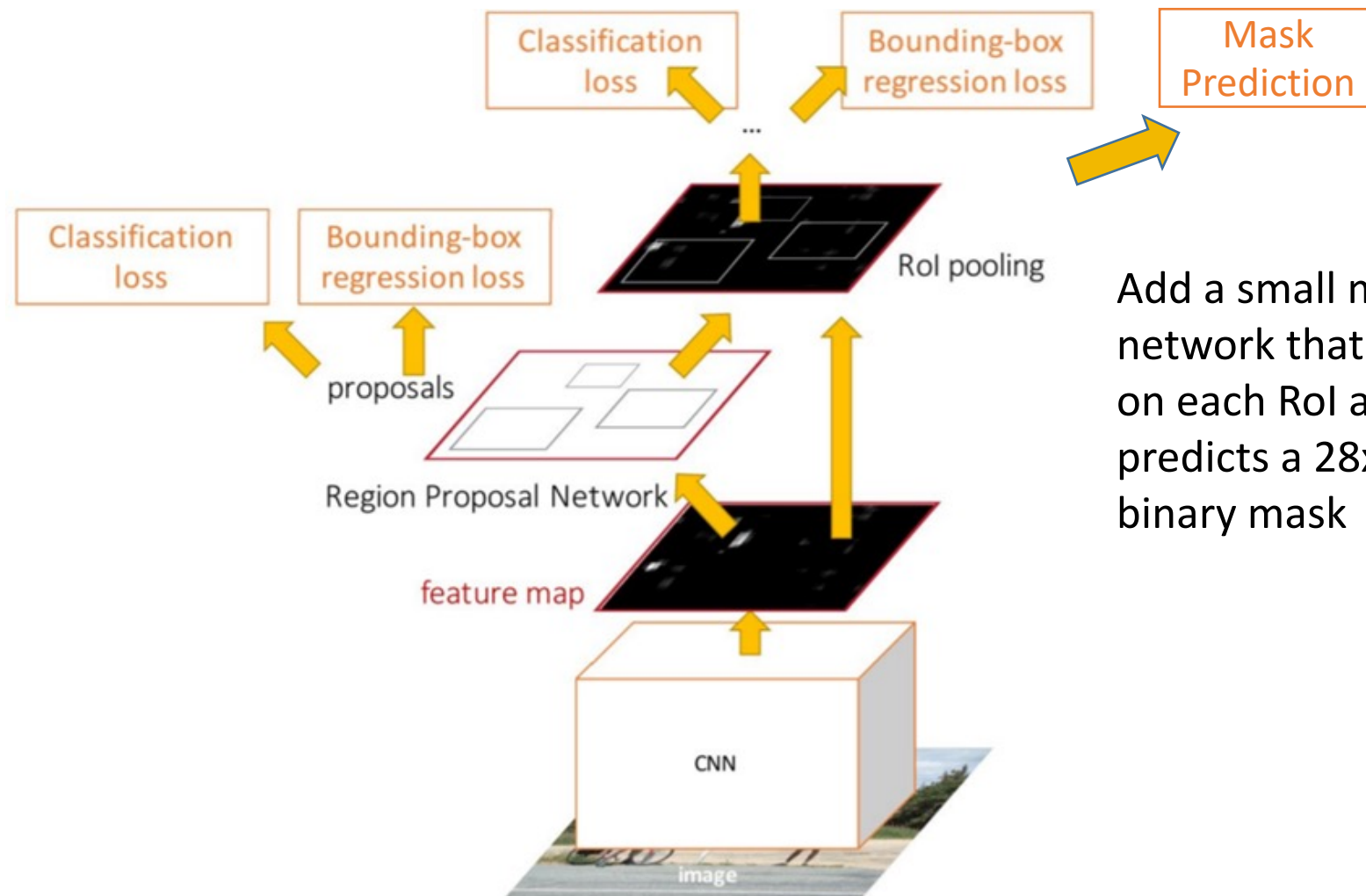
Multiple Object

**Instance
Segmentation**

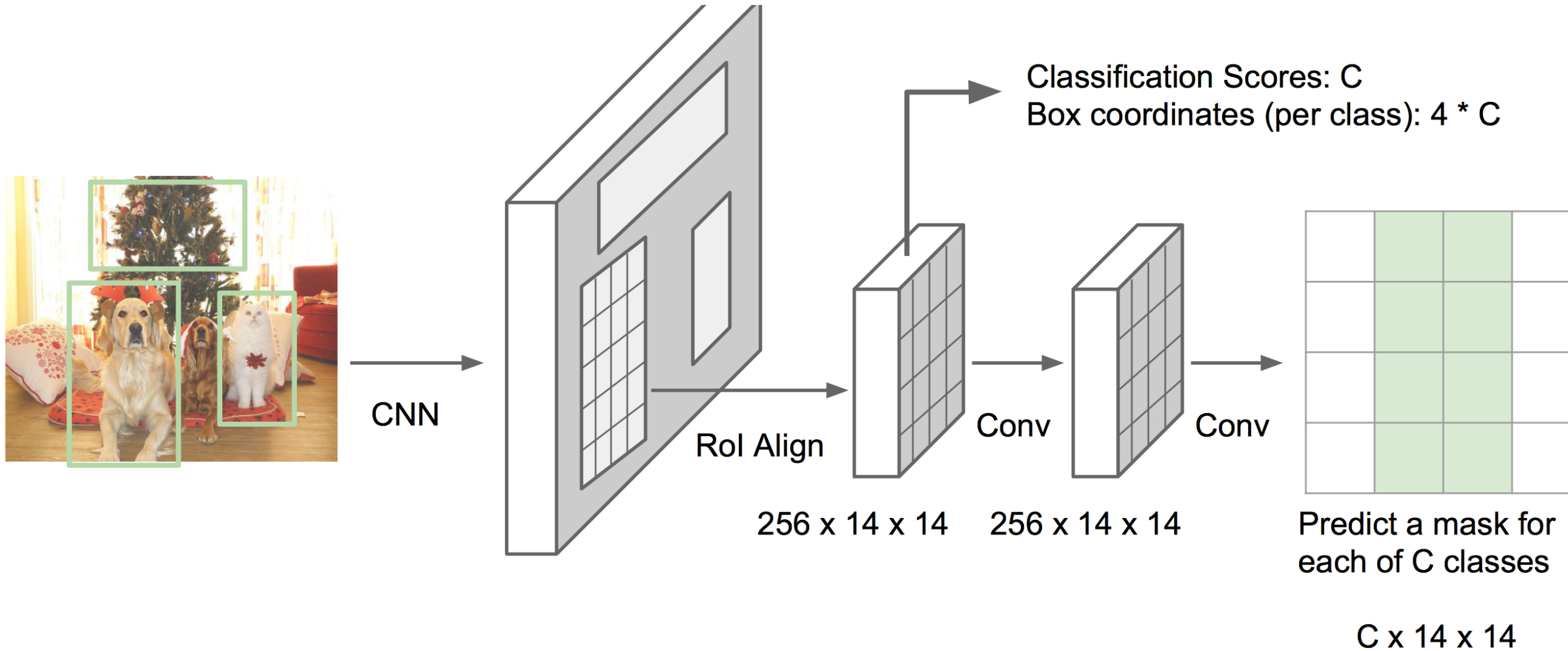


DOG, DOG, CAT

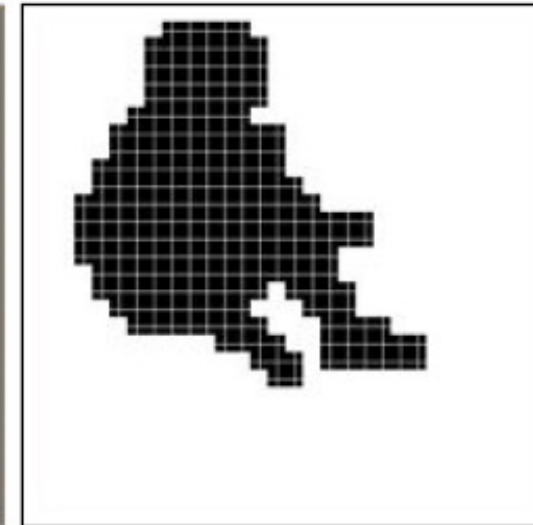
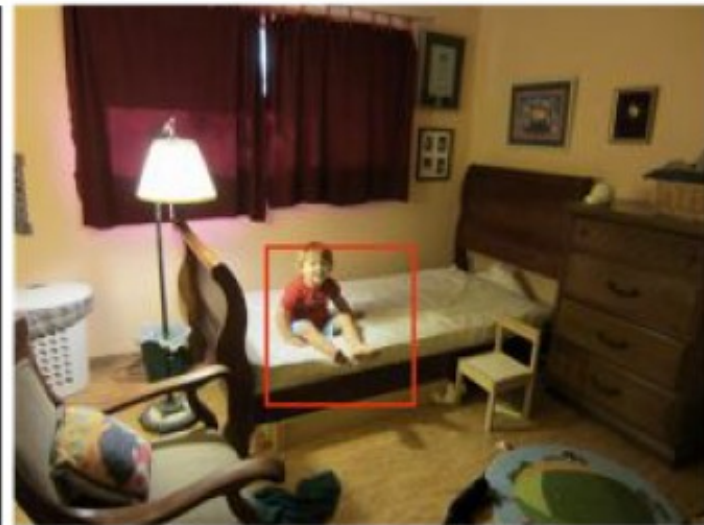
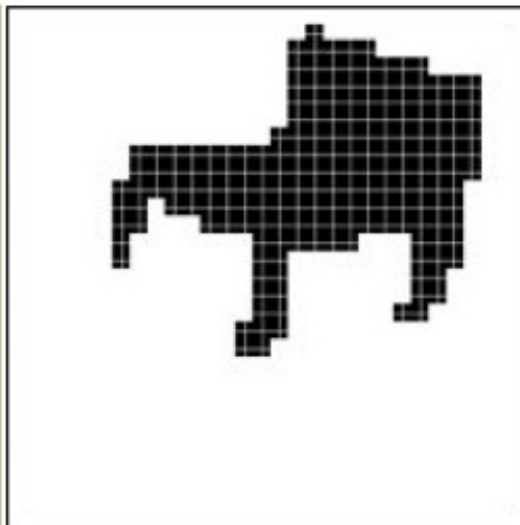
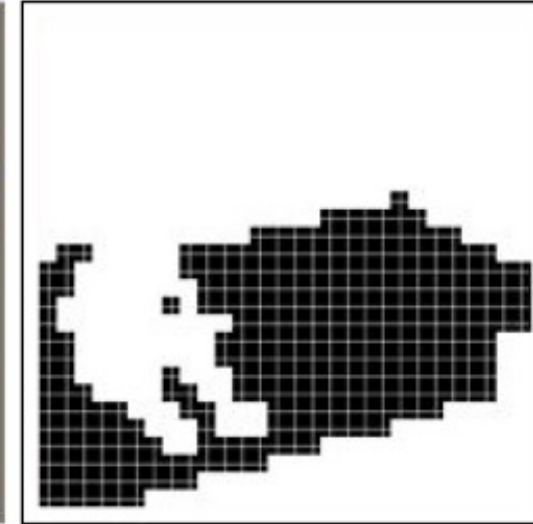
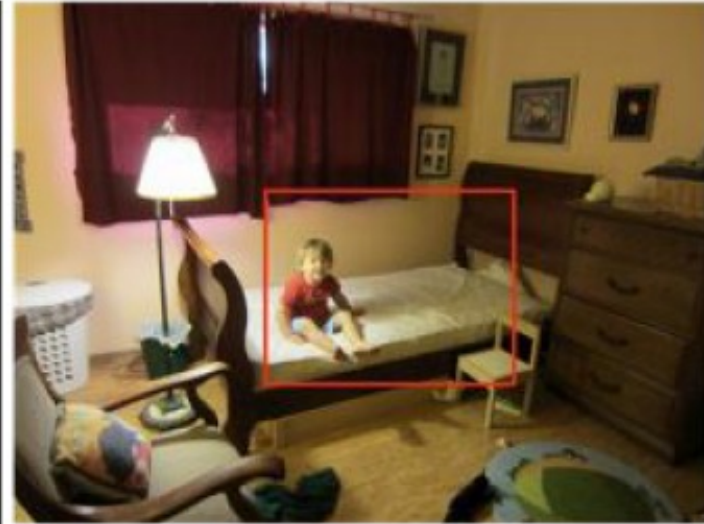
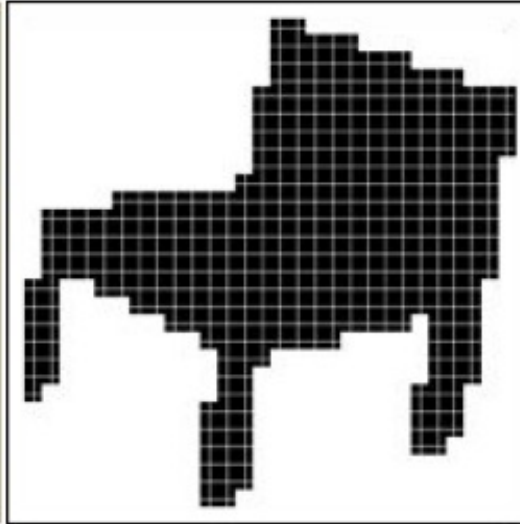
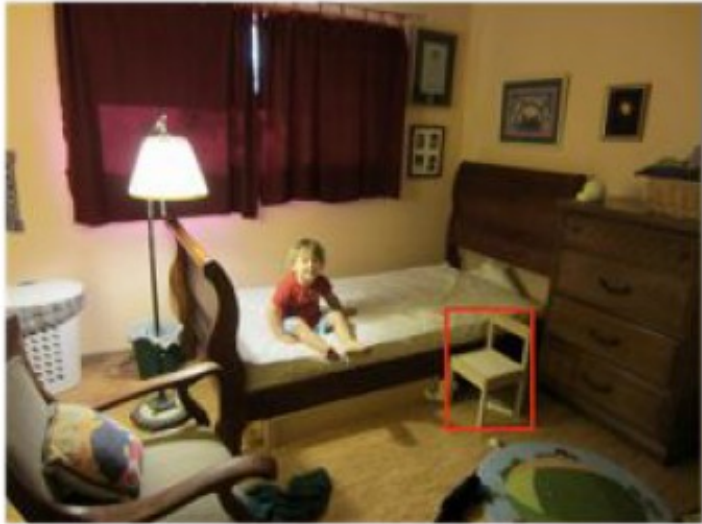
[This image is CC0 public domain](#)



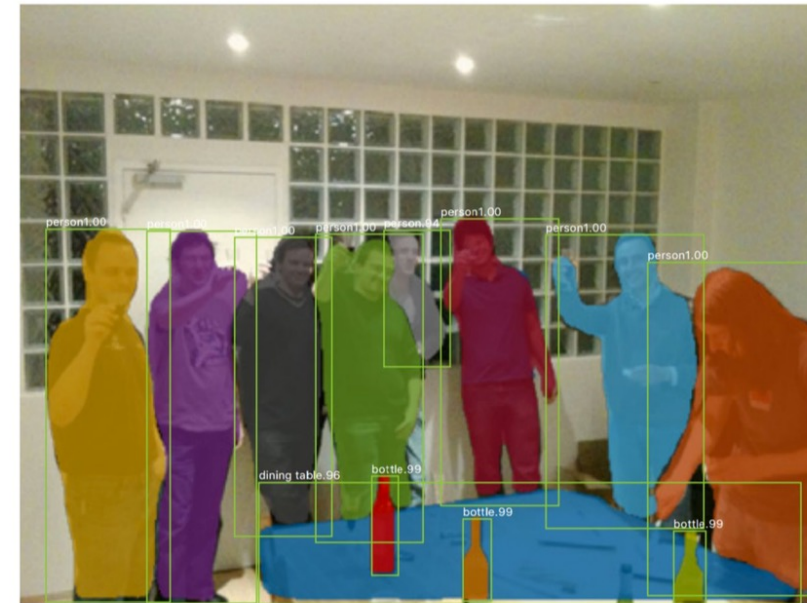
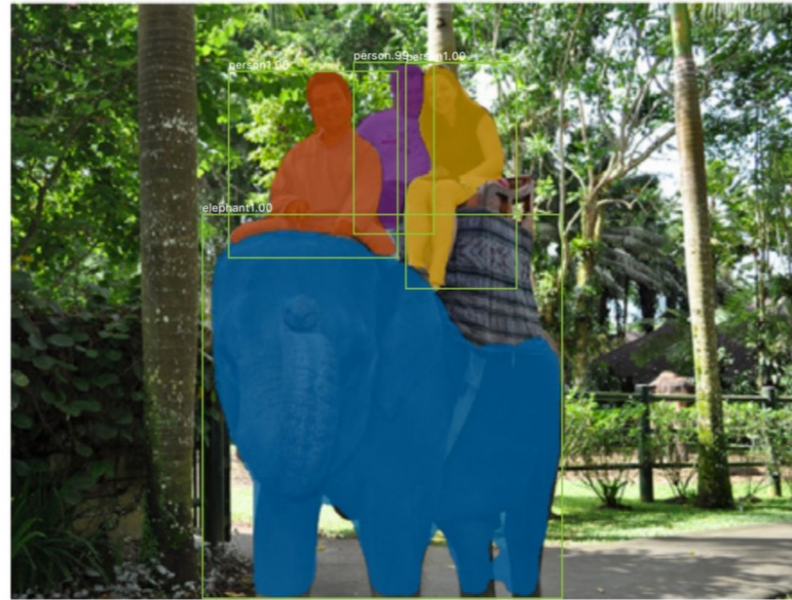
Mask R-CNN



Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Very Good Results!



He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

Mask R-CNN: Also does pose



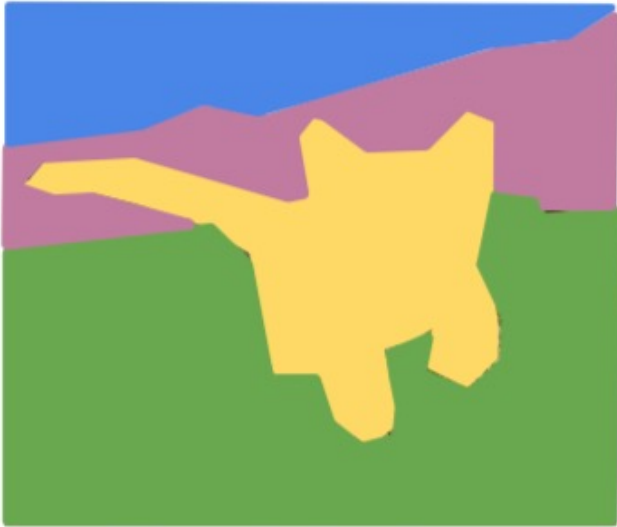
He et al, "Mask R-CNN", arXiv 2017

Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.

Reproduced with permission.

Summary

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation



DOG, DOG, CAT

[This image is CC0 public domain](#)

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 Detection:
 - <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