Object Detection

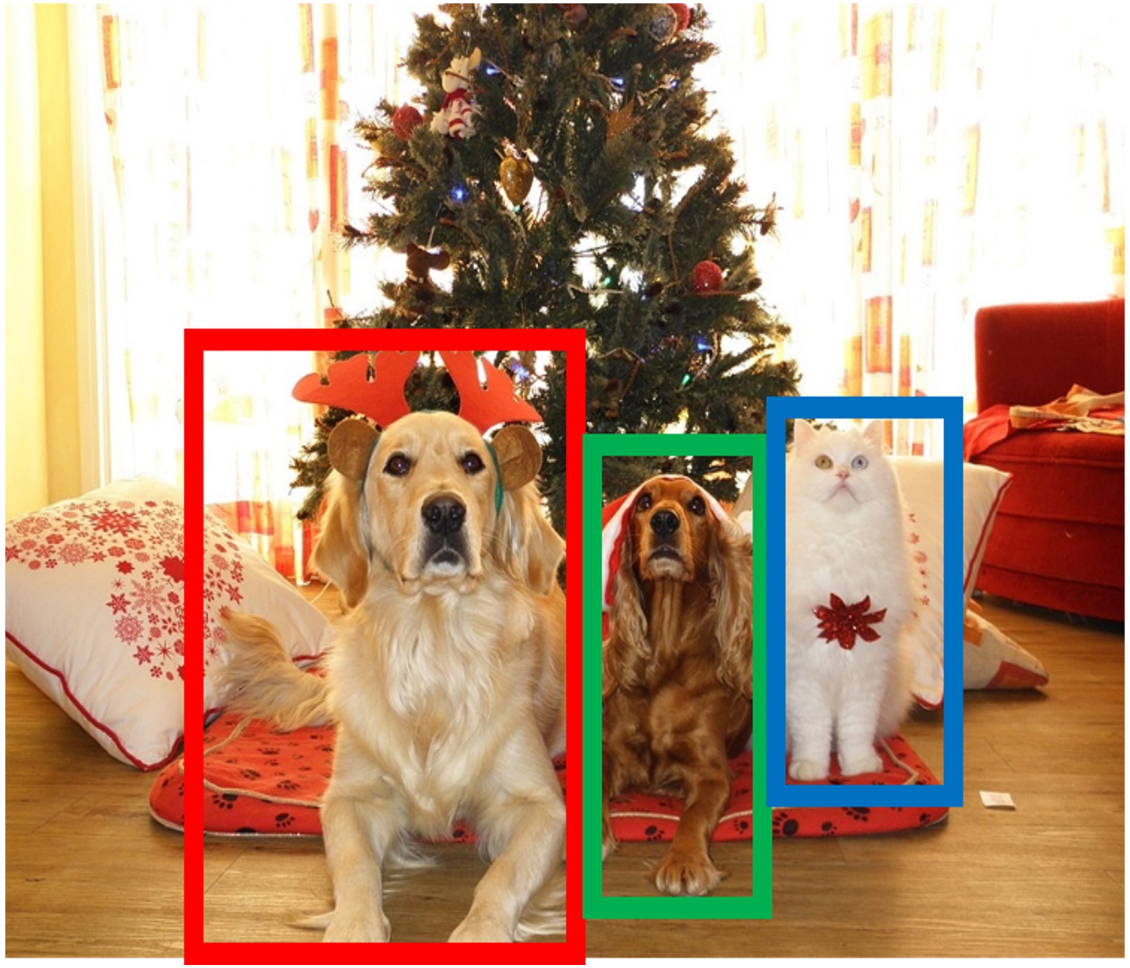
Slides from Justin Johnson

Object Detection: Task Definition

Input: Single RGB Image

Output: A set of detected objects;
For each object predict:

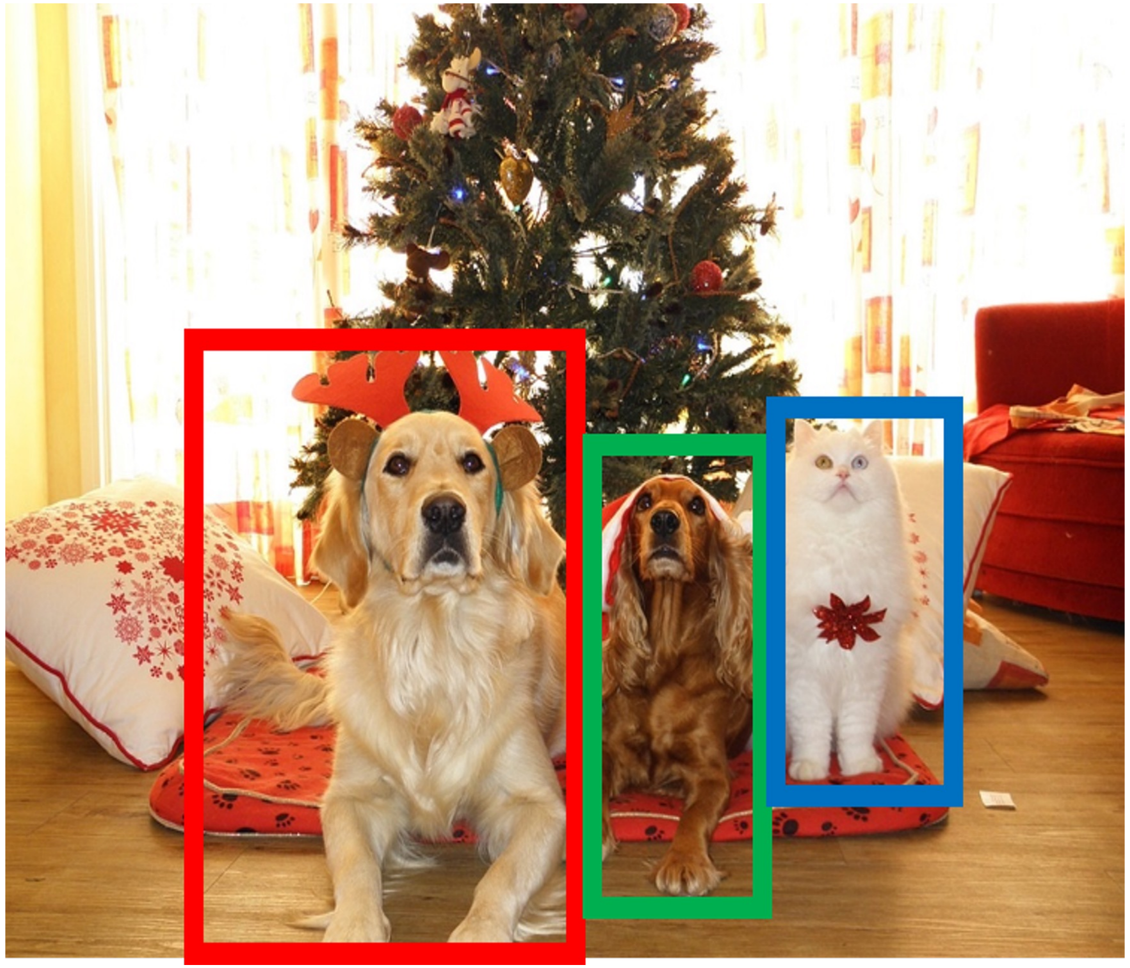
1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)



Slide from Justin Johnson

Object Detection: Challenges

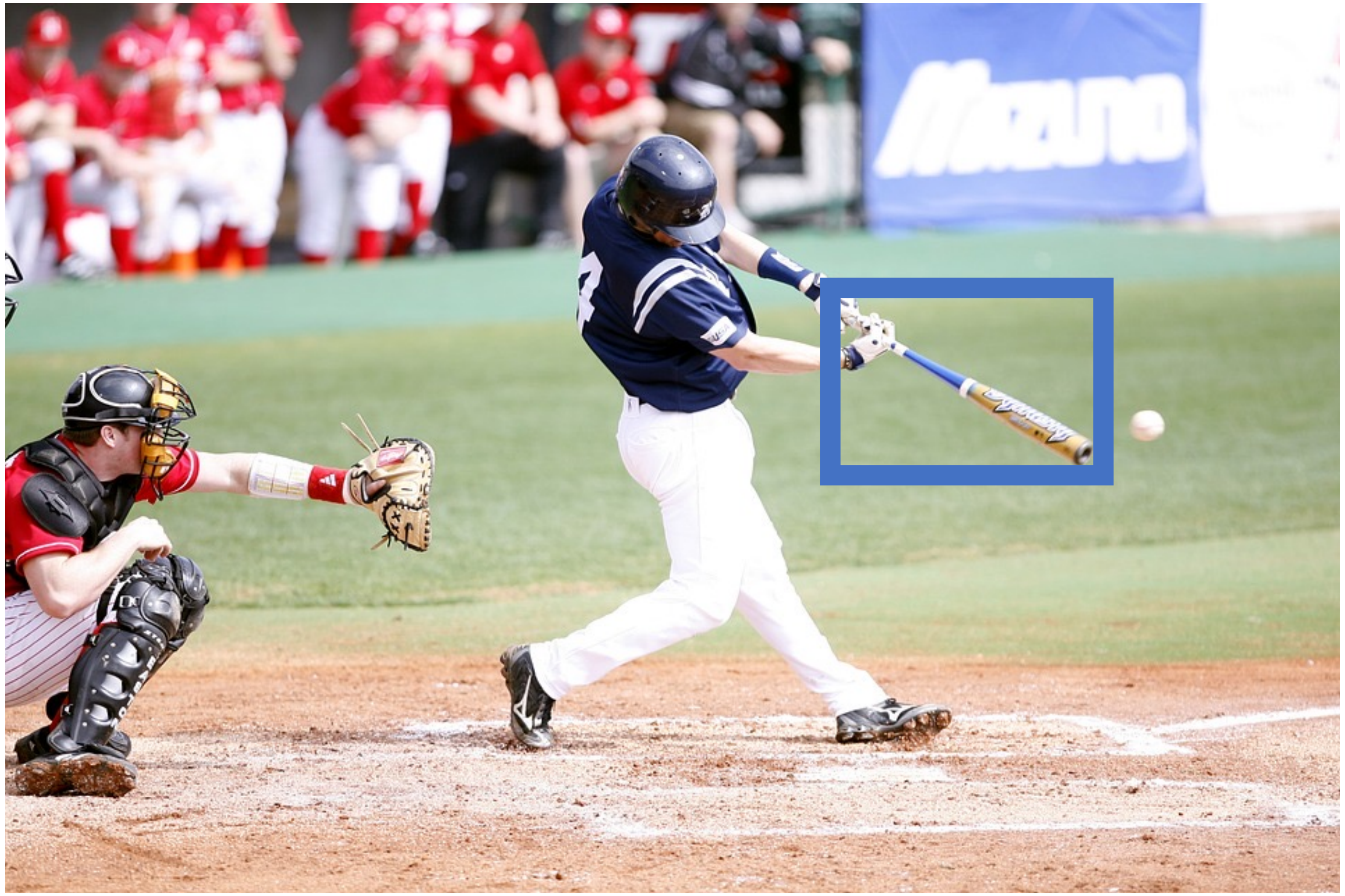
- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of output**: Need to predict ”what” (category label) as well as “where” (bounding box)
- **Large images**: Classification works at 224x224; need higher resolution for detection, often ~800x600



Slide from Justin Johnson

Bounding Boxes

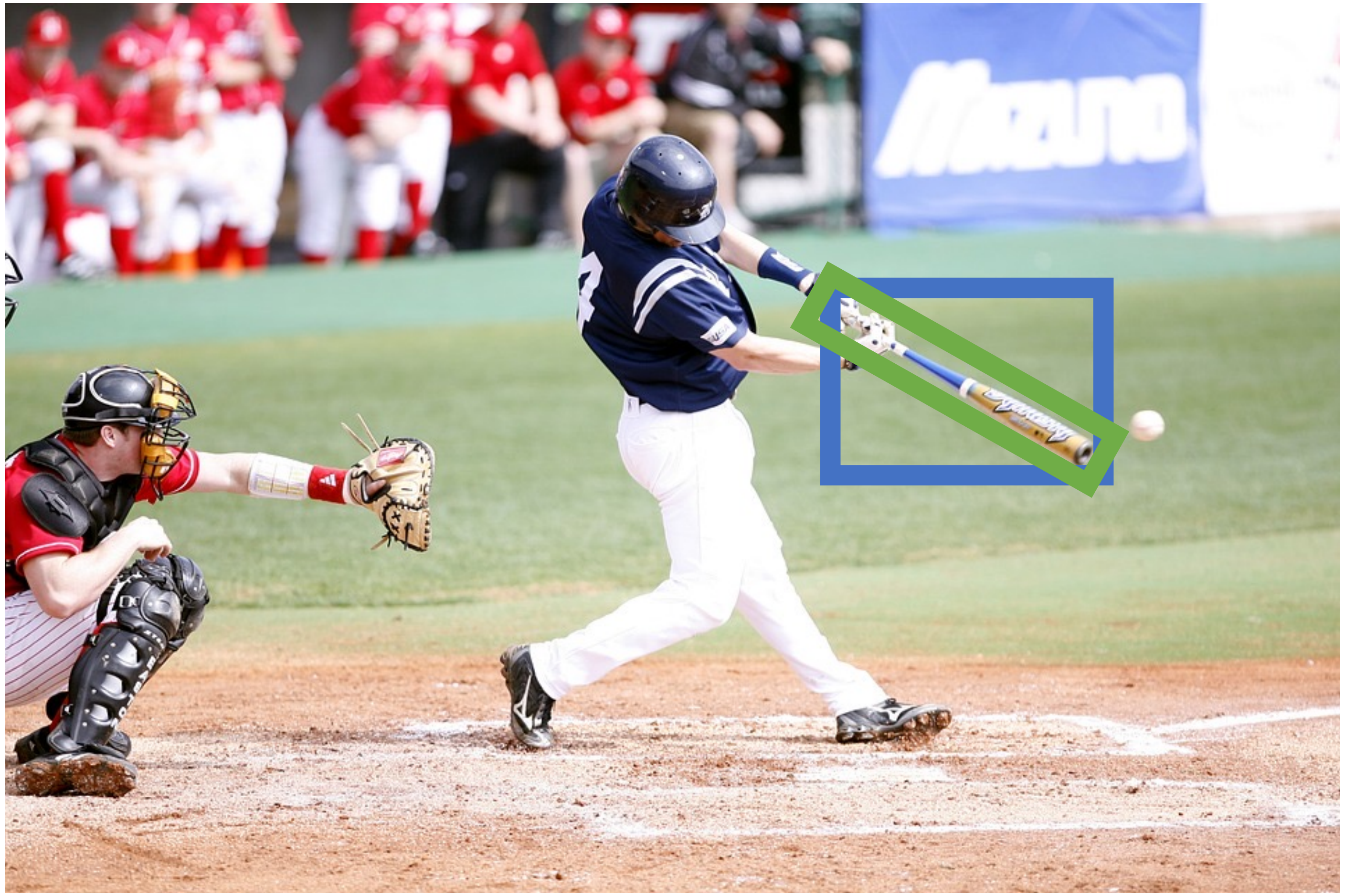
Bounding boxes are typically *axis-aligned*



Bounding Boxes

Bounding boxes are typically *axis-aligned*

Oriented boxes are much less common



Object Detection: Modal vs Amodal Boxes

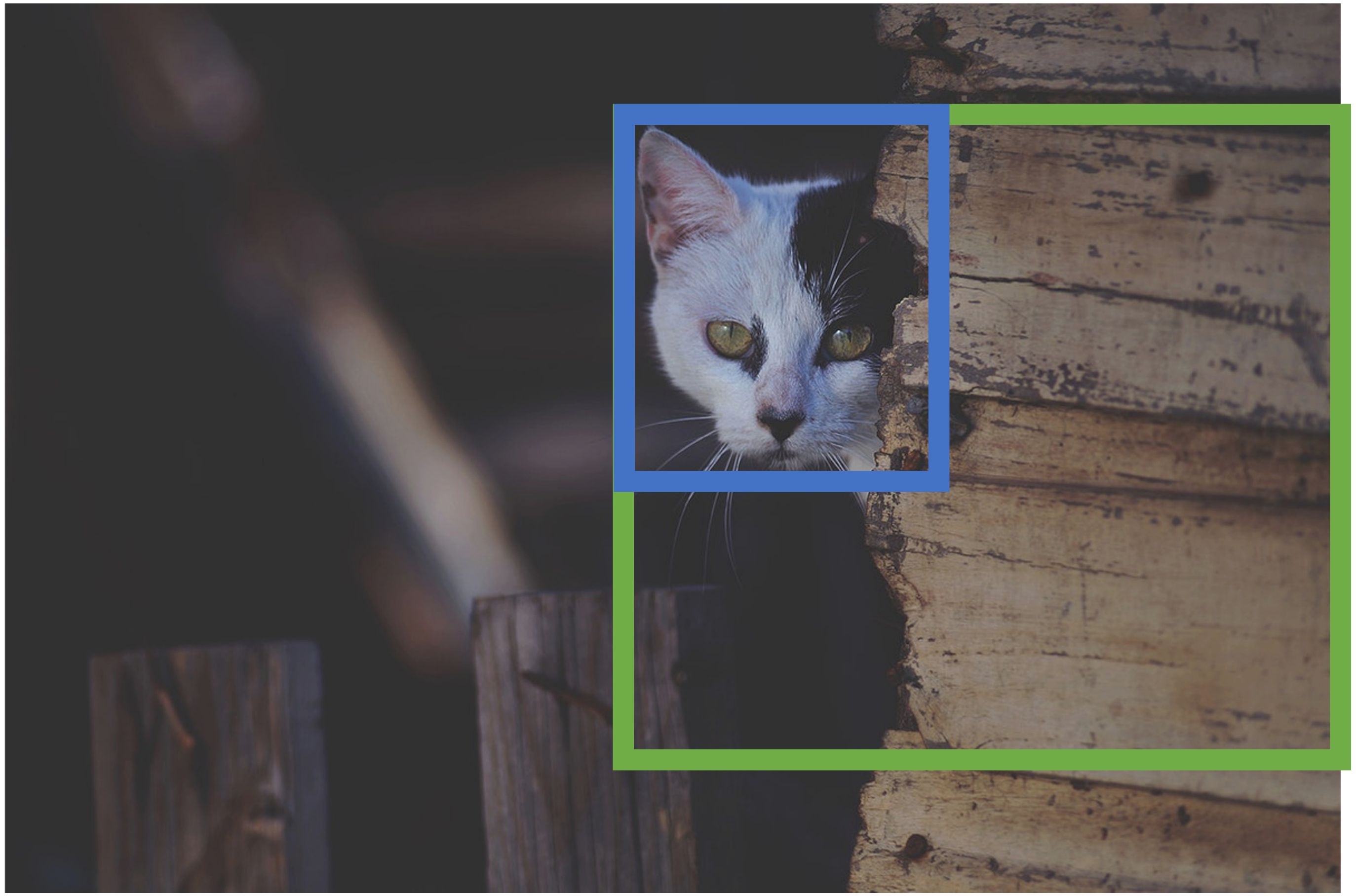
Bounding boxes (usually) cover only the visible portion of the object

Zhu et al, "Semantic Amodal Segmentation", CVPR 2017

Object Detection: Modal vs Amodal Boxes

Bounding boxes (usually) cover only the visible portion of the object

Amodal detection:
box covers the entire extent of the object, even occluded parts



Zhu et al, "Semantic Amodal Segmentation", CVPR 2017

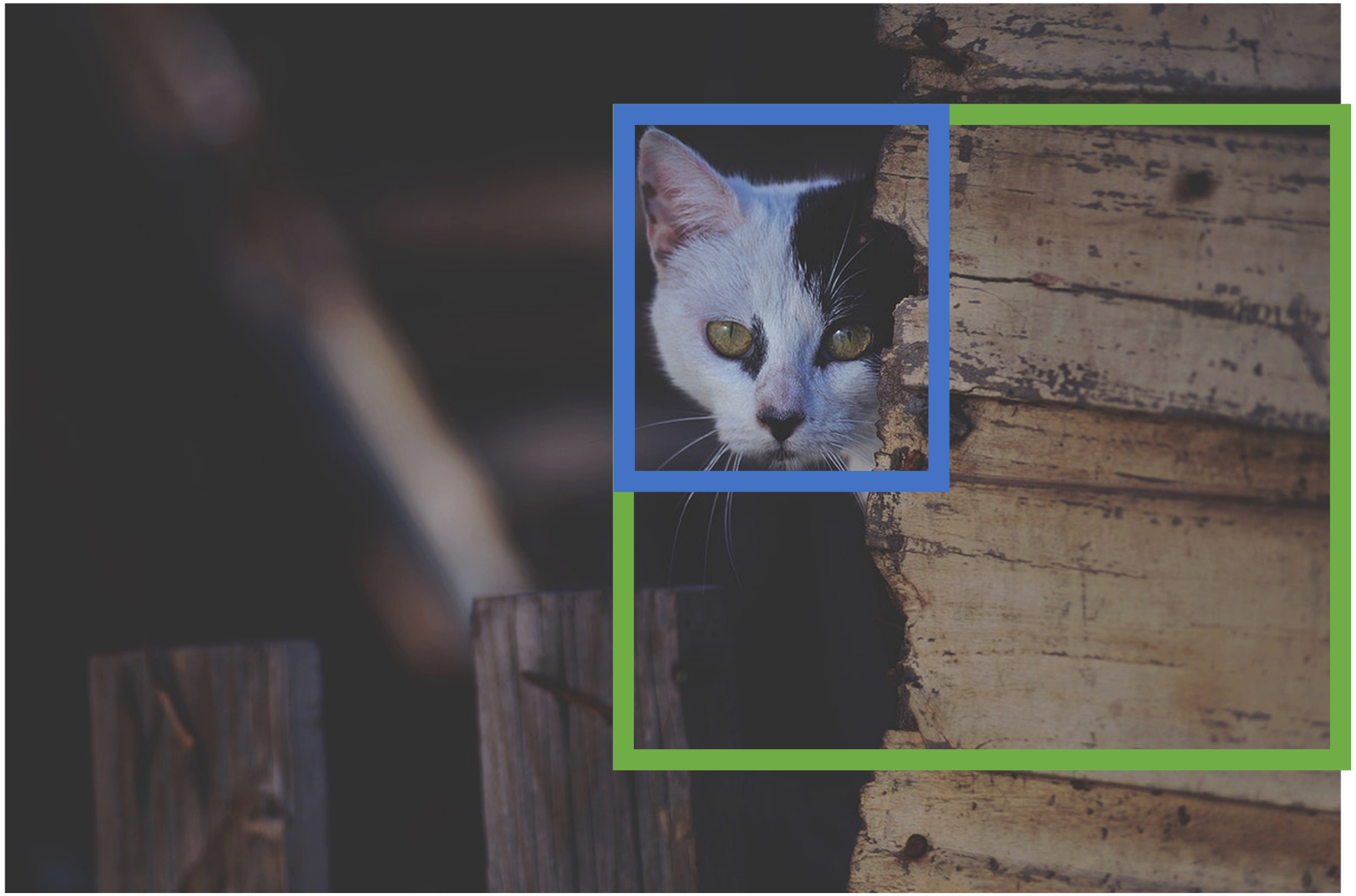
This image is CC0 Public Domain

Slide from Justin Johnson

Object Detection: Modal vs Amodal Boxes

<u>"Modal" detection</u>:
Bounding boxes (usually) cover only the visible portion of the object

<u>Amodal detection</u>:
box covers the entire extent of the object, even occluded parts



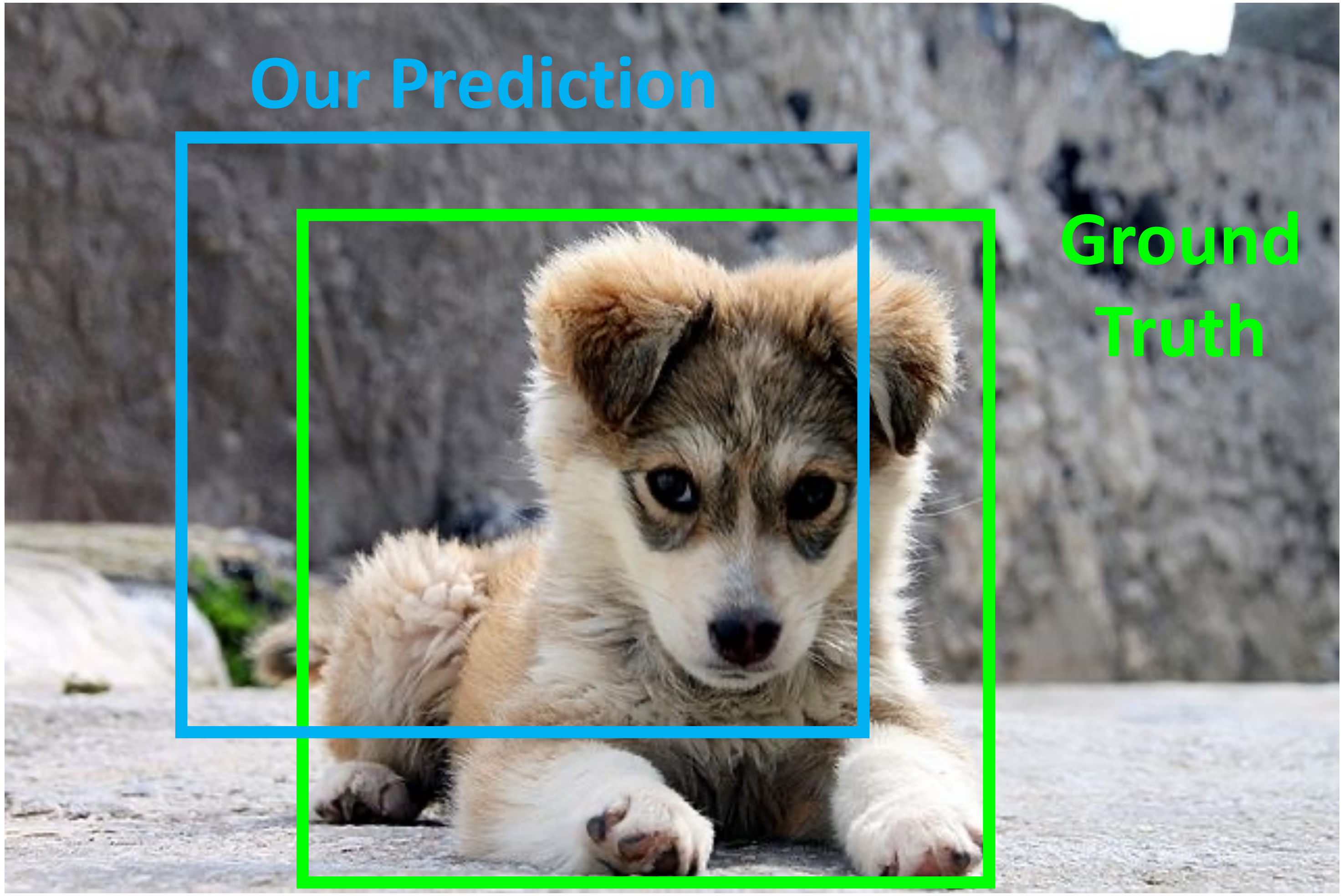
Zhu et al, "Semantic Amodal Segmentation", CVPR 2017

This image is CC0 Public Domain

Slide from Justin Johnson

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

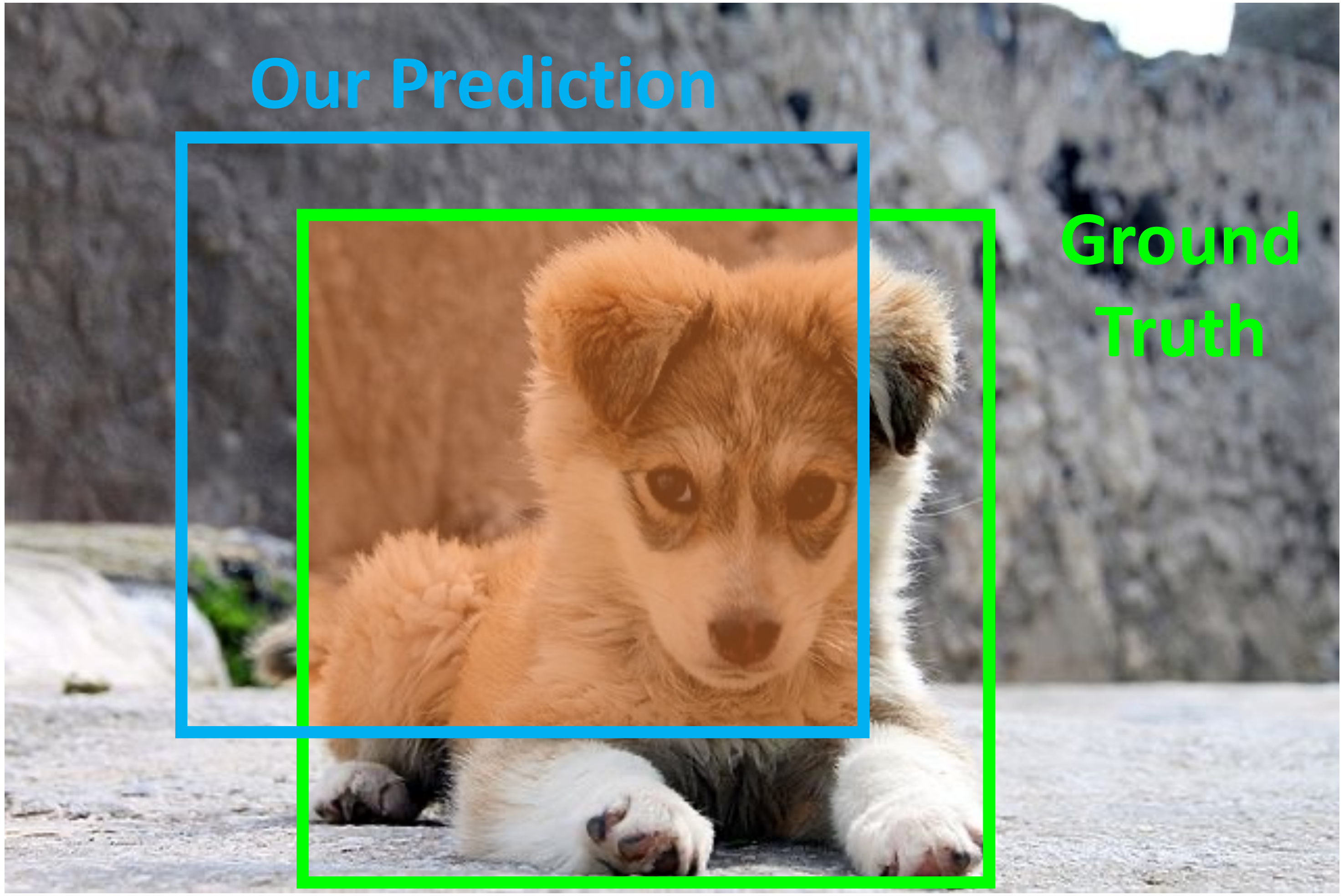


Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

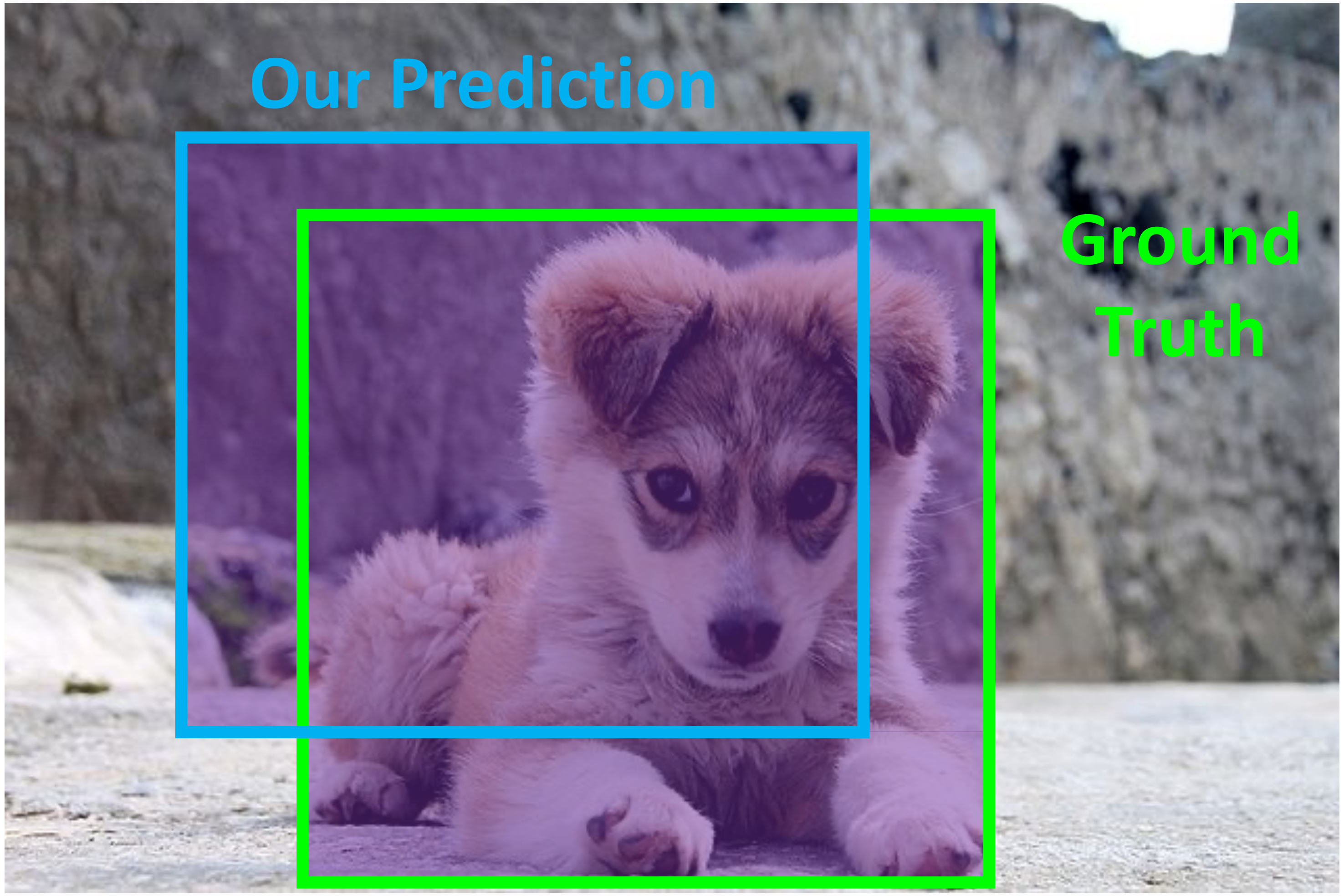


Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



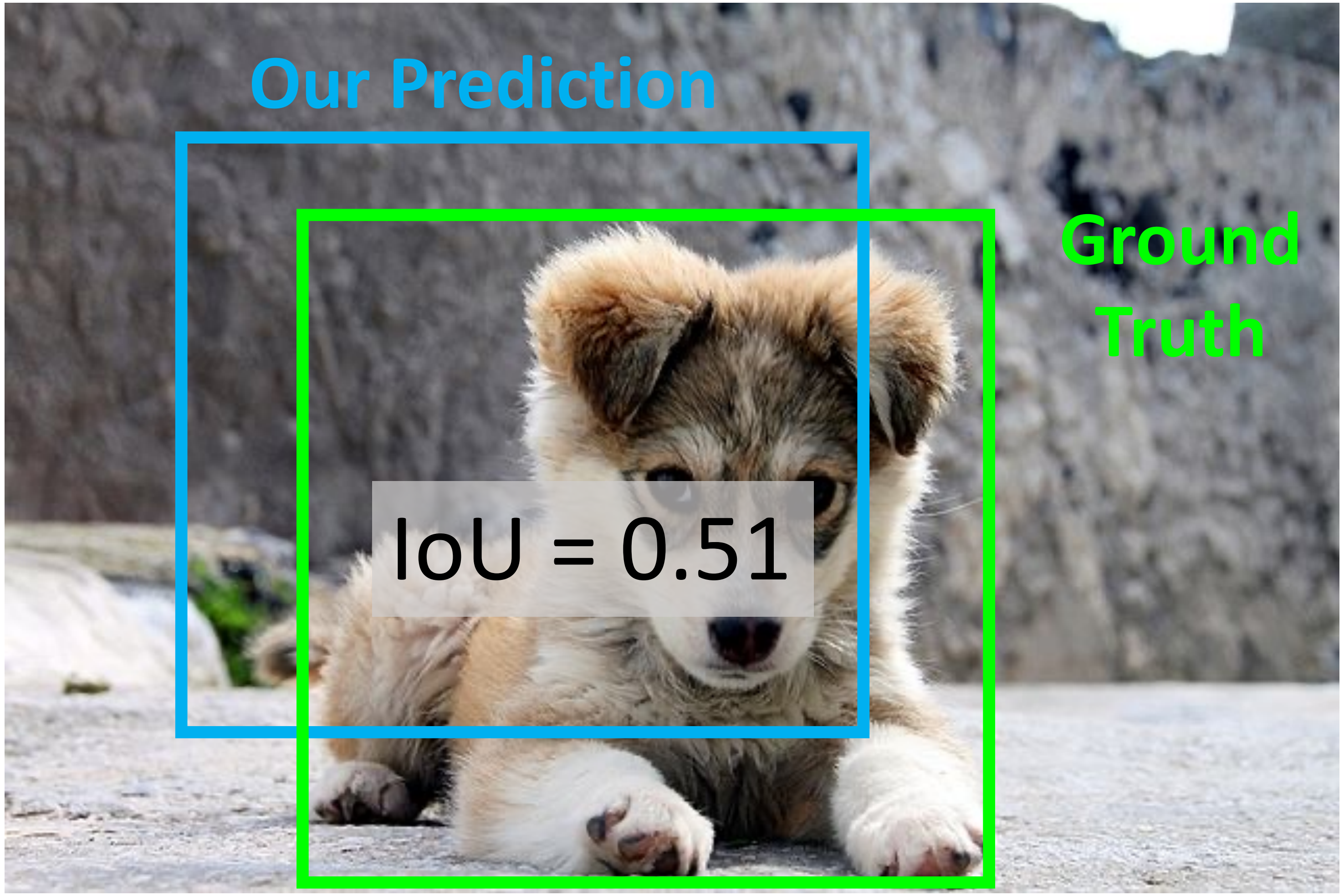
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent"



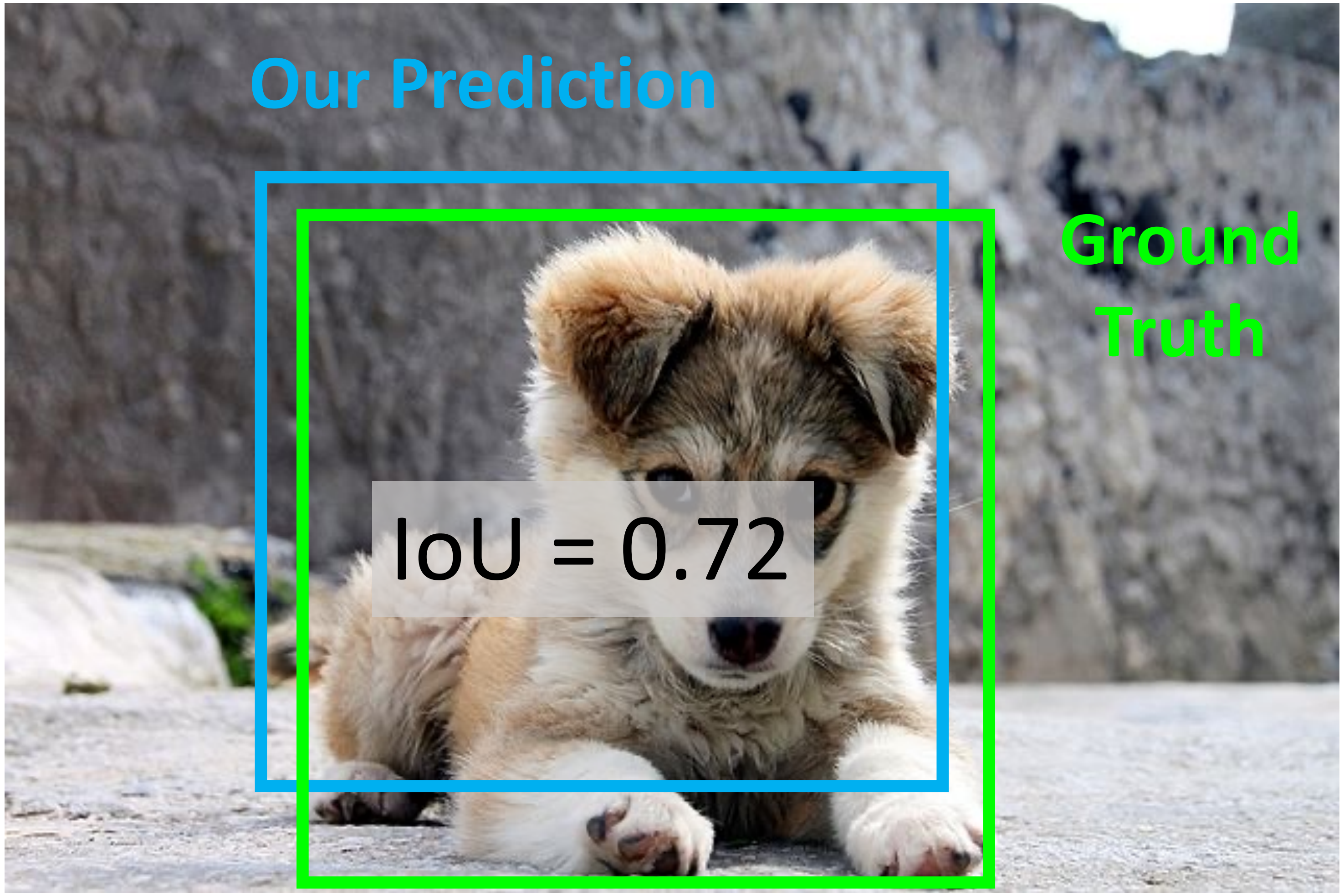
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is “decent”,
IoU > 0.7 is “pretty good”,



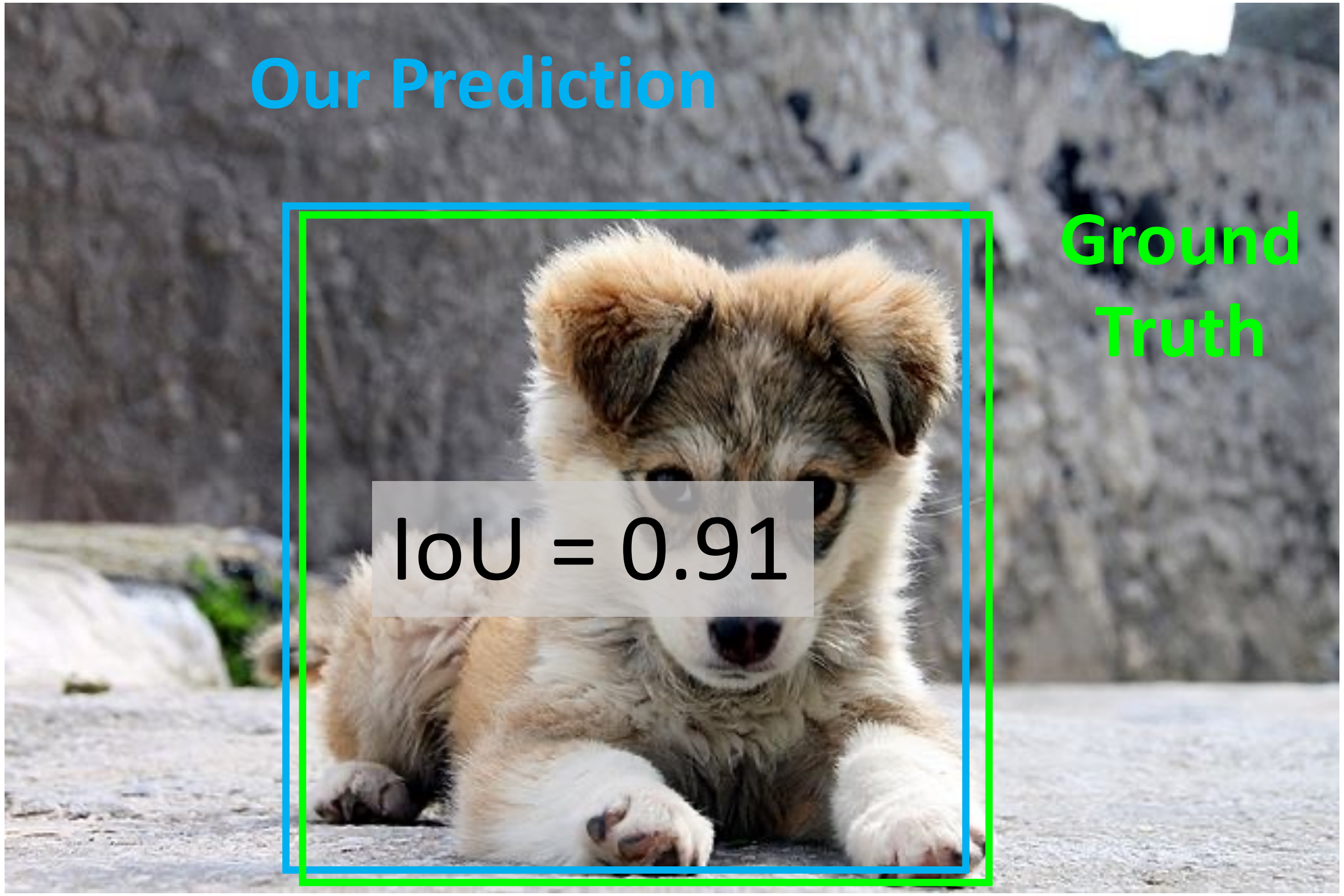
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

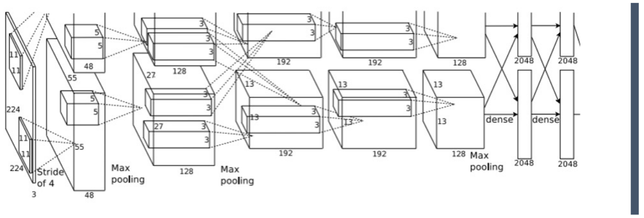
Intersection over Union (IoU)
(Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is “decent”,
IoU > 0.7 is “pretty good”,
IoU > 0.9 is “almost perfect”



Detecting a single object



Vector:
4096

Treat localization as a regression problem!

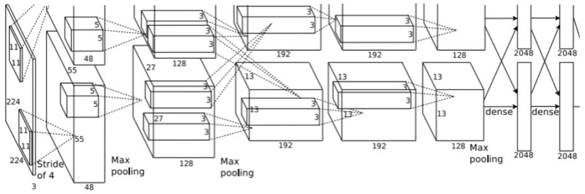
Detecting a single object “What”

Correct label:
Cat

Fully Connected:
4096 to 1000

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

Softmax Loss



Vector:
4096

Treat localization as a regression problem!

Slide from Justin Johnson

Detecting a single object "What"

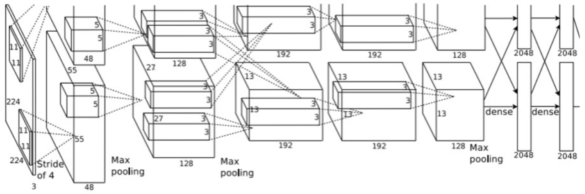
Correct label:
Cat

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

Softmax Loss

Fully Connected:
4096 to 1000

This image is CC0 public domain



Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

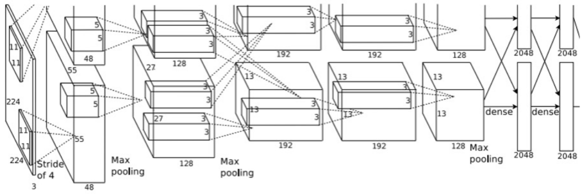
Correct box:
(x', y', w', h')

"Where"

Treat localization as a regression problem!

Detecting a single object

“What”



This image is CC0 public domain

Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

Correct box: (x’, y’, w’, h’)

L2 Loss

Weighted Sum → **Loss**

$L = L_{cls} + \lambda L_{reg}$

Multitask Loss

“Where”

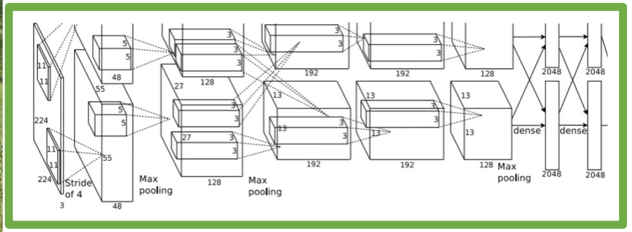
Detecting a single object

"What"

Correct label:
Cat

Often pretrained on ImageNet (Transfer learning)

This image is CC0 public domain



Fully Connected:
4096 to 1000

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

Softmax Loss

Multitask Loss

Weighted Sum

Loss

$L = L_{cls} + \lambda L_{reg}$

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!

"Where"

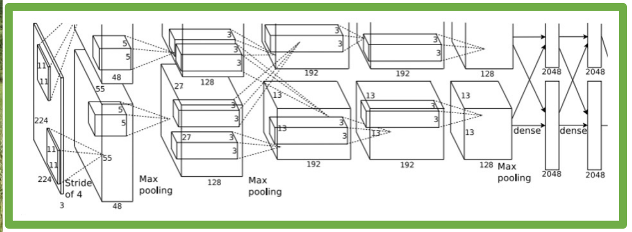
Detecting a single object

"What"

Correct label:
Cat

Often pretrained
on ImageNet
(Transfer learning)

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

Multitask Loss

Weighted Sum

Loss

$L = L_{cls} + \lambda L_{reg}$

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!

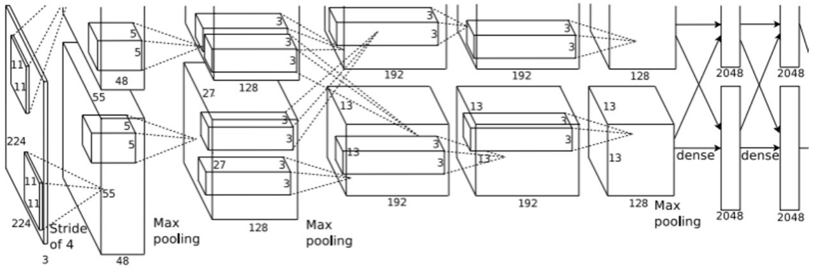
Problem: Images can have more than one object!

"Where"

Slide from Justin Johnson

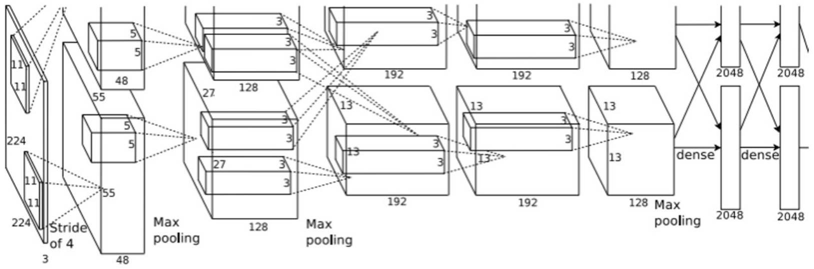
Detecting Multiple Objects

Need different numbers of outputs per image



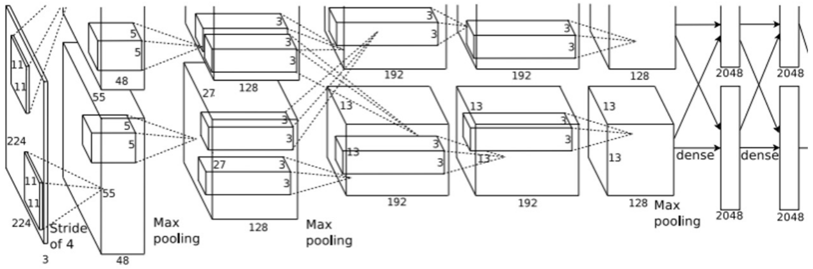
CAT: (x, y, w, h)

4 numbers



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

12 numbers



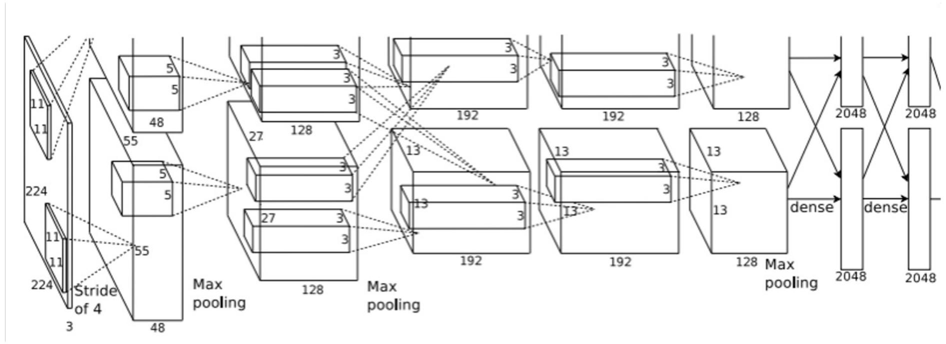
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Slide from Justin Johnson

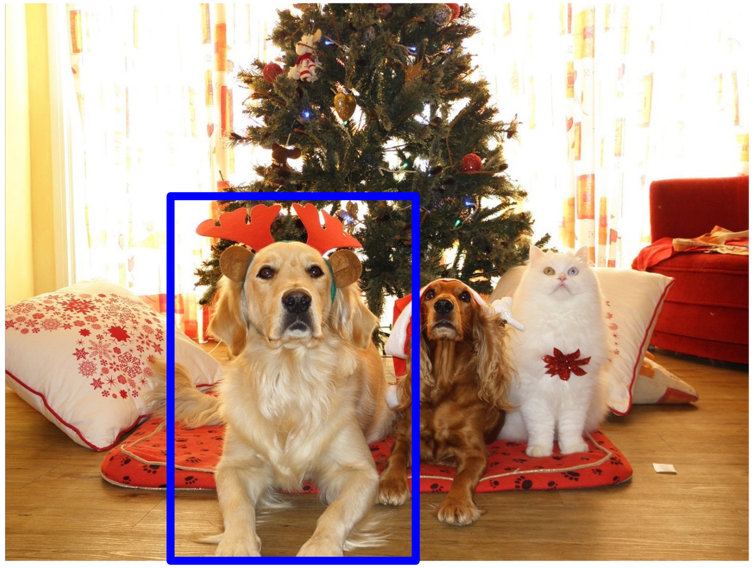
Detecting Multiple Objects: **Sliding Window**

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

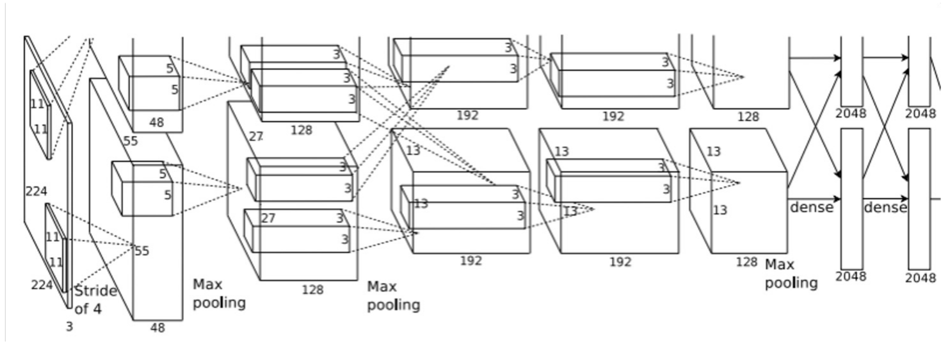


Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: **Sliding Window**



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



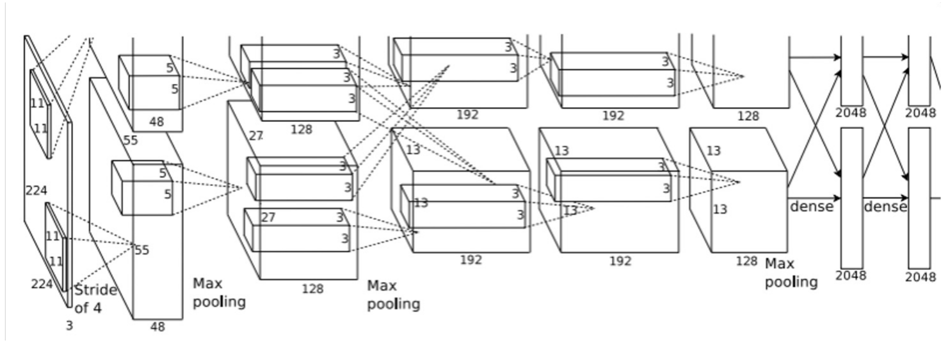
Dog? YES

Cat? NO

Background? NO

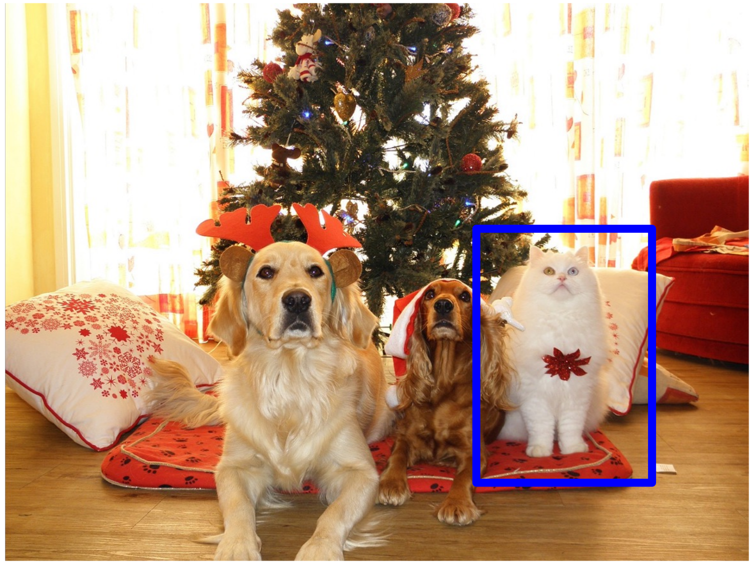
Detecting Multiple Objects: **Sliding Window**

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

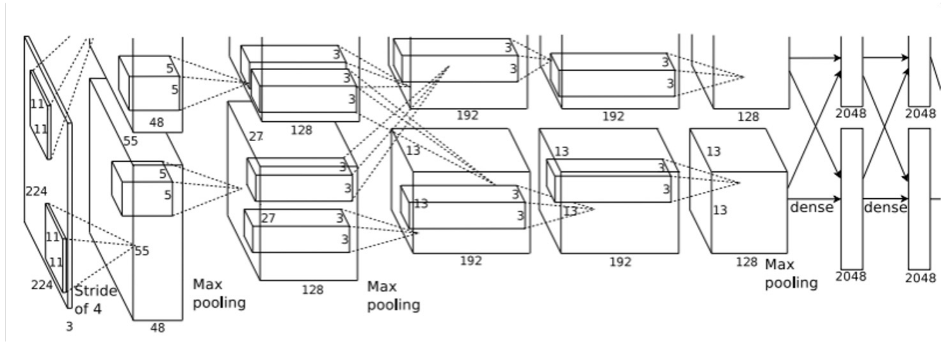


Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

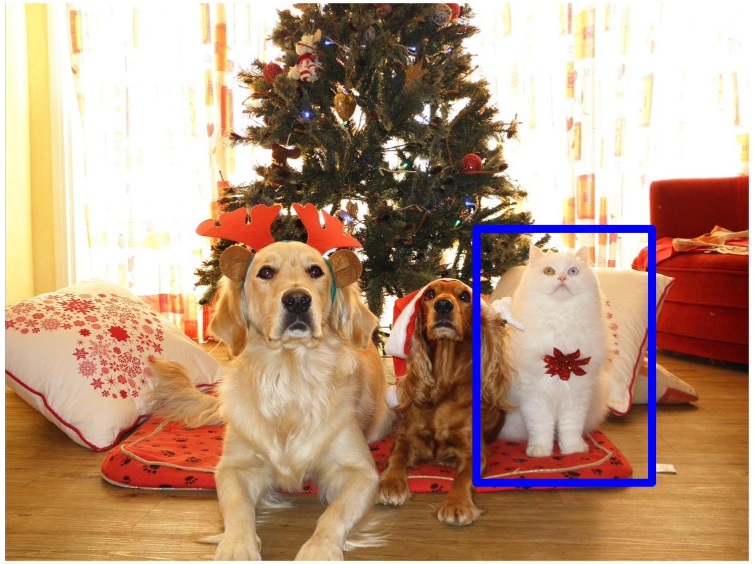


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

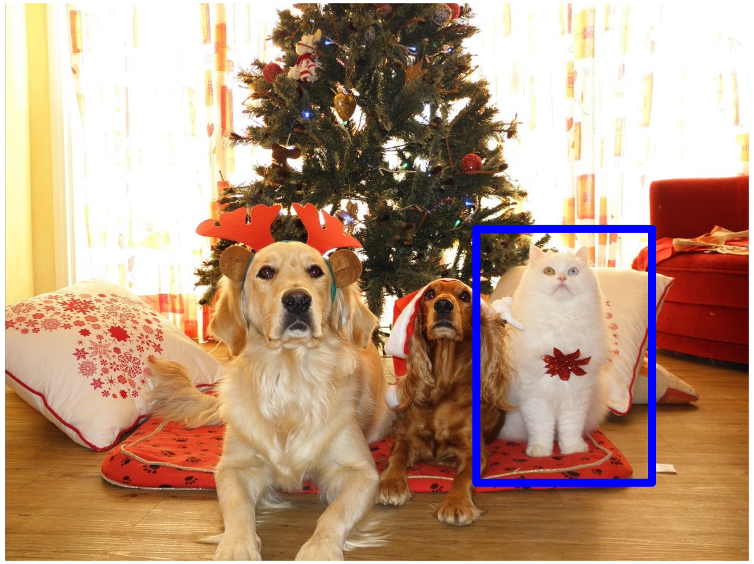
Detecting Multiple Objects: **Sliding Window**



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

Question: How many possible boxes are there in an image of size H x W?

Detecting Multiple Objects: **Sliding Window**



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

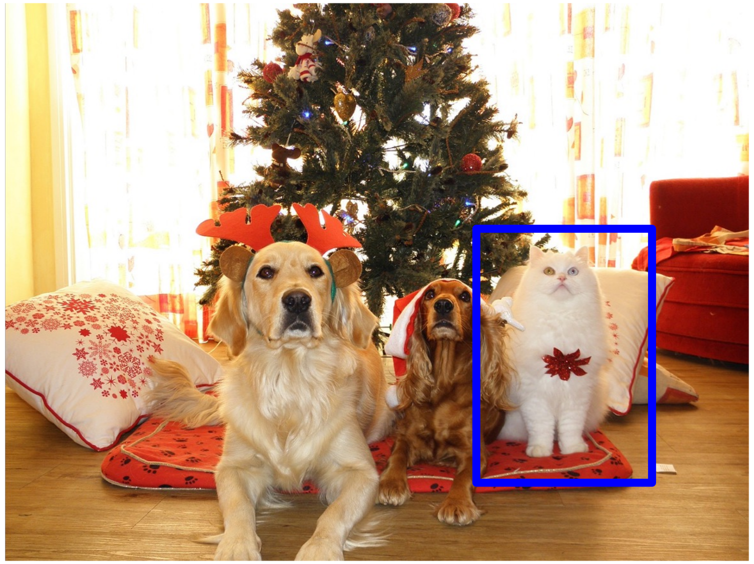
Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Slide from Justin Johnson

Detecting Multiple Objects: **Sliding Window**

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



Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

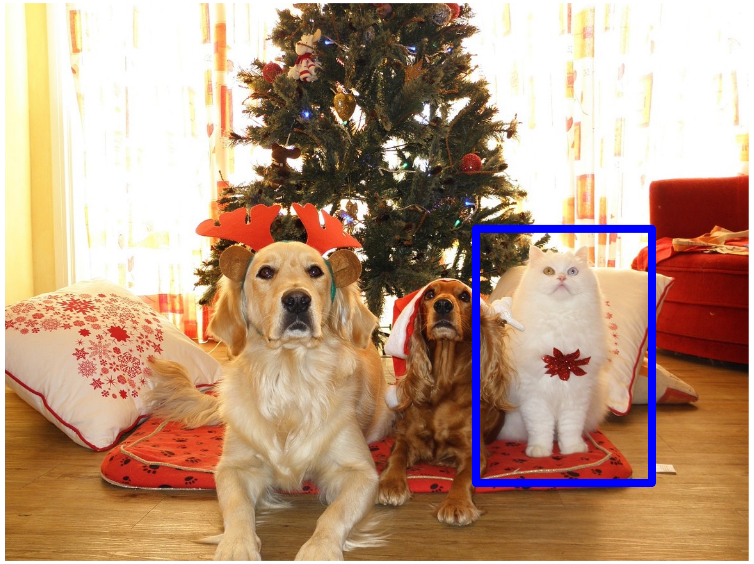
Total possible boxes:

$$\sum_{h=1}^{H} \sum_{w=1}^{W} (W - w + 1)(H - h + 1)$$

$$= \frac{H(H+1)}{2} \frac{W(W+1)}{2}$$

Slide from Justin Johnson

Detecting Multiple Objects: **Sliding Window**



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

800 x 600 image has ~58M boxes! No way we can evaluate them all

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

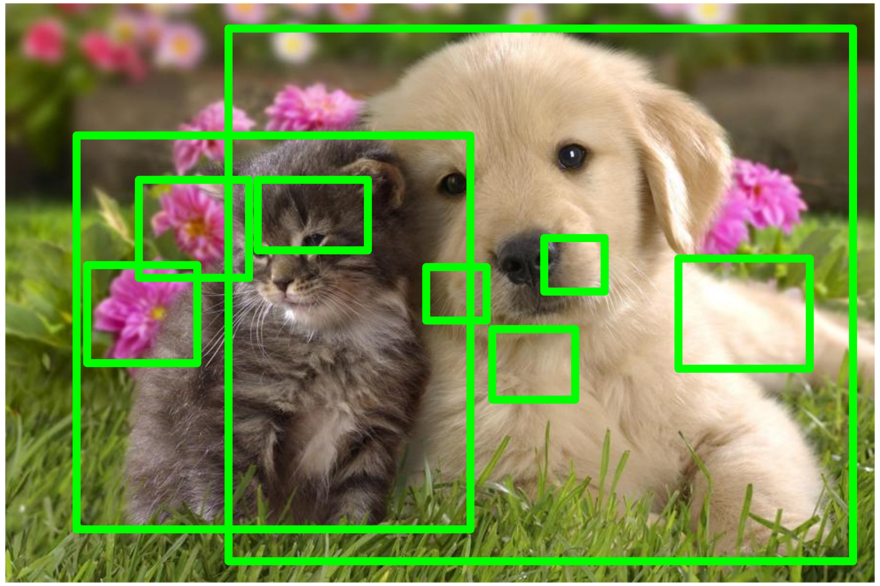
Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- 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: Region-Based 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.

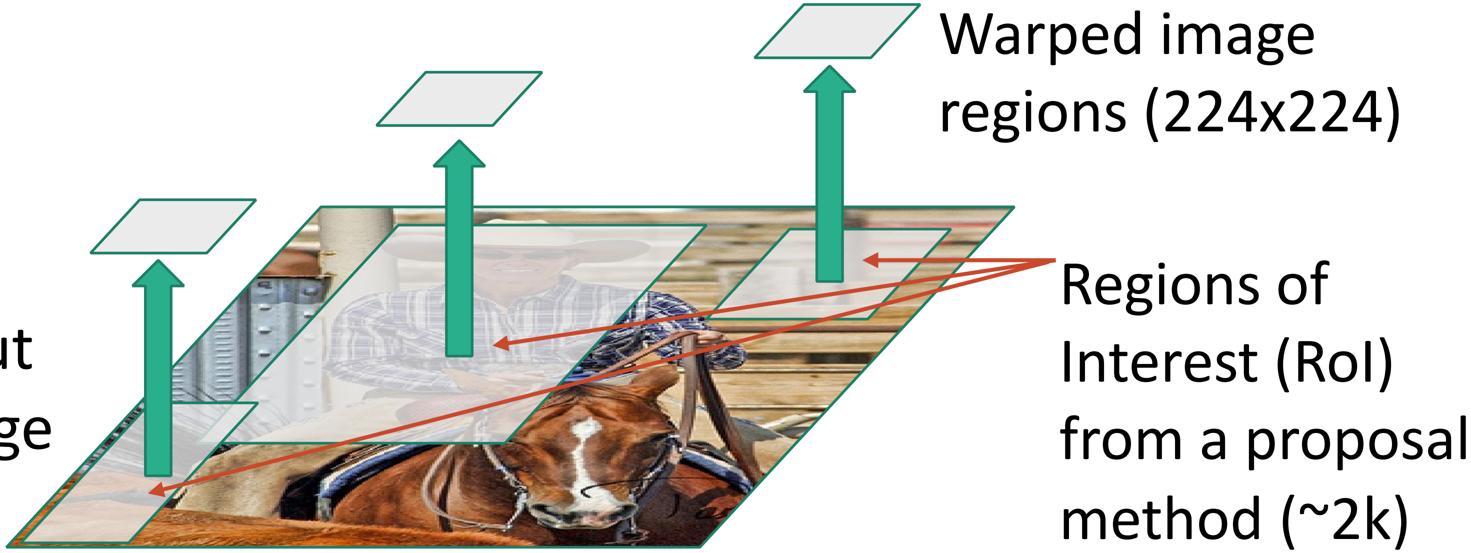
R-CNN: Region-Based CNN

Input image

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

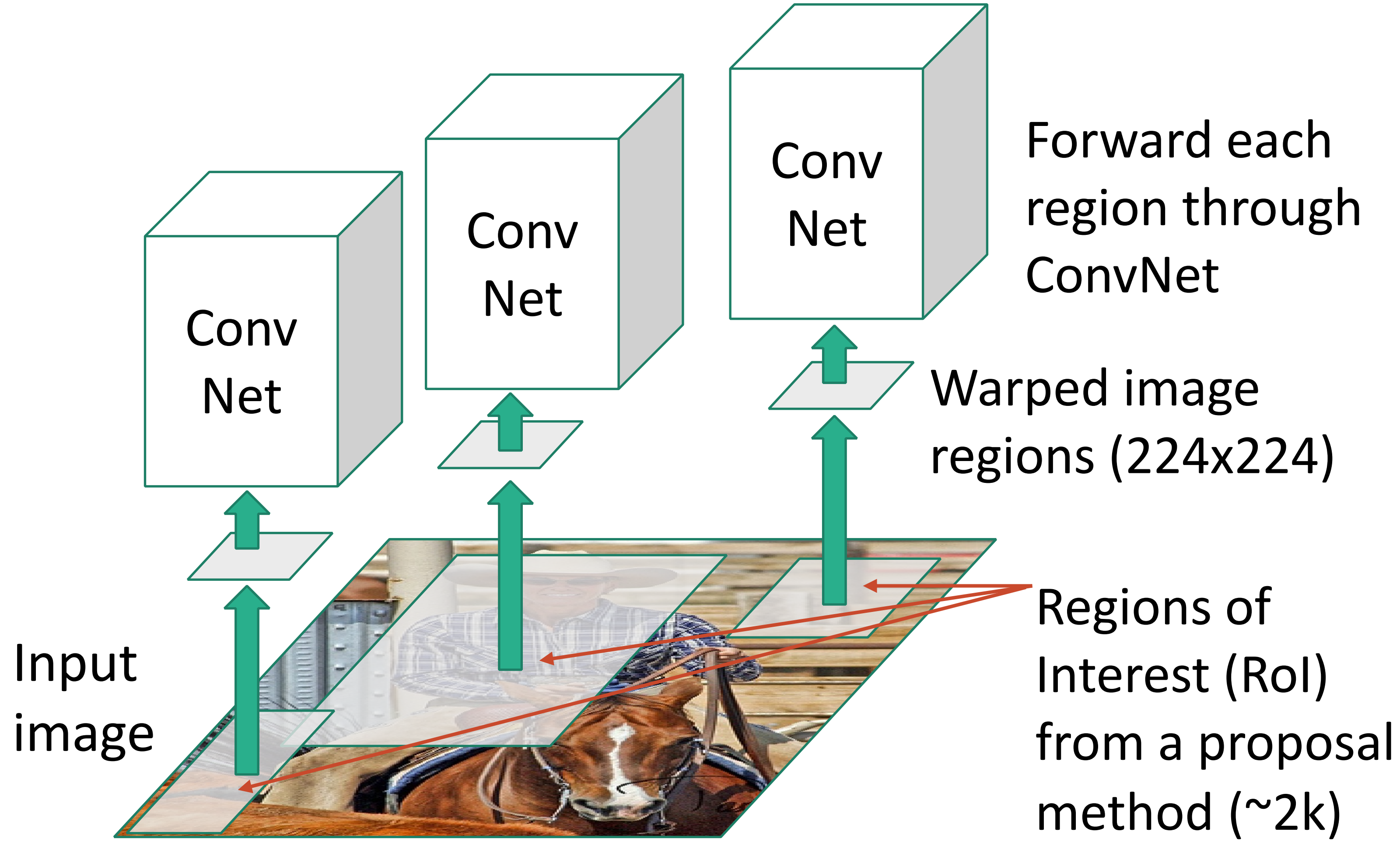
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN: Region-Based CNN



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

R-CNN: Region-Based CNN

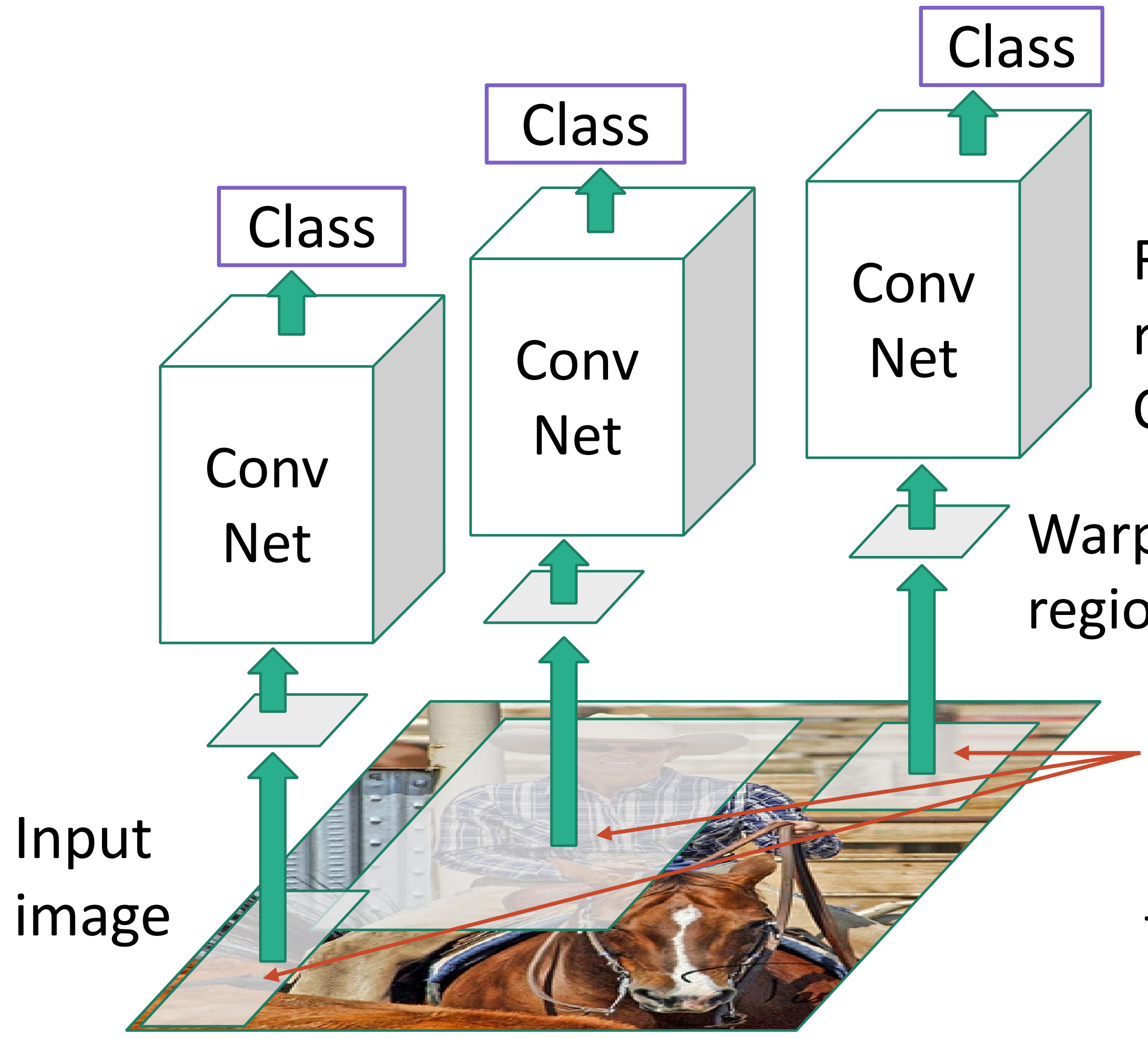


Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](). Reproduced with permission.

Slide from Justin Johnson

R-CNN: Region-Based CNN

Classify each region



Forward each region through ConvNet

Warped image regions (224x224)

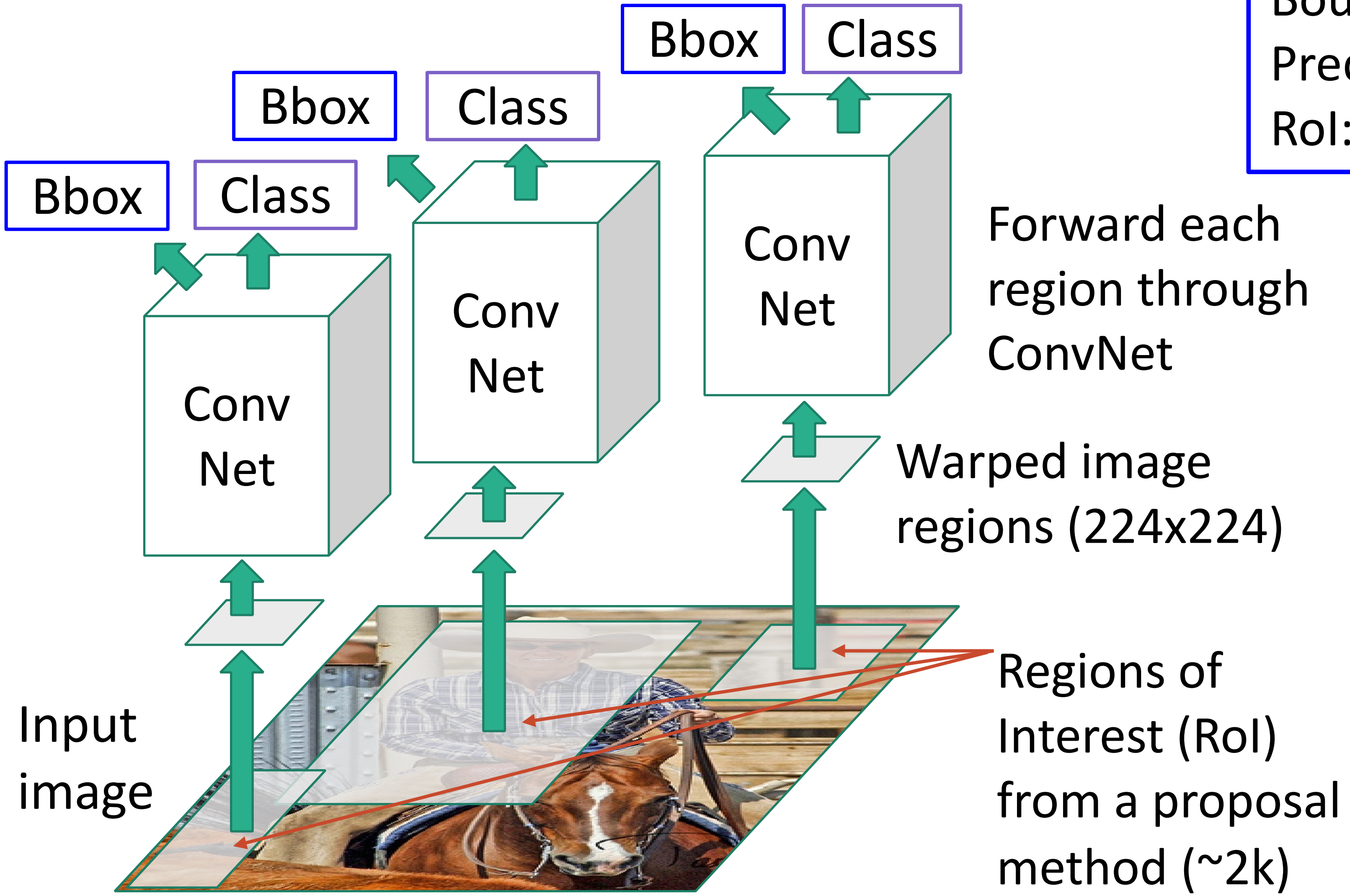
Regions of Interest (RoI) from a proposal method (~2k)

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

R-CNN: Region-Based CNN

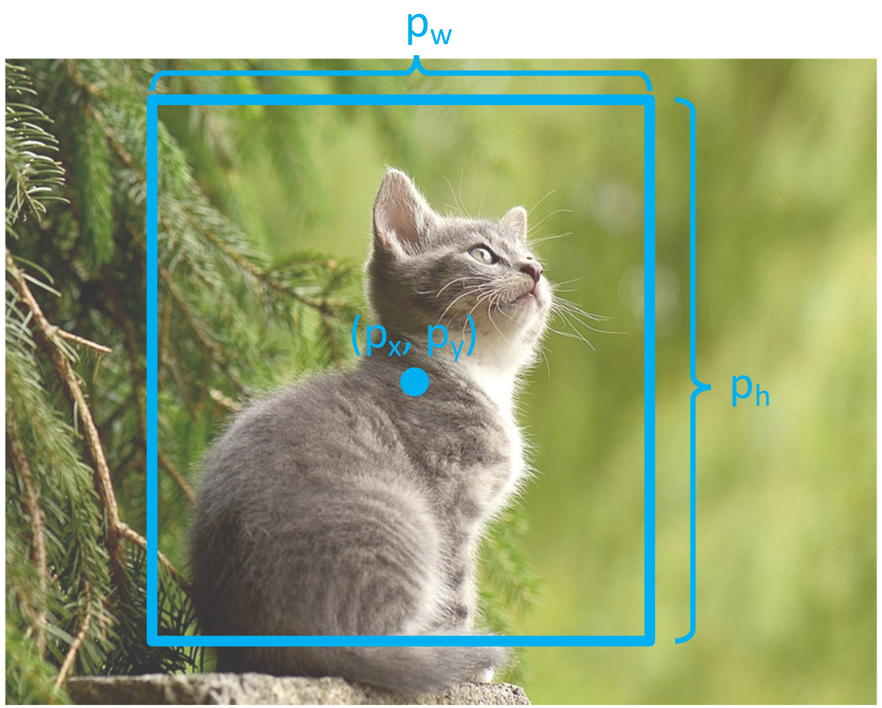
Classify each region

Bounding box regression:
Predict “transform” to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)



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

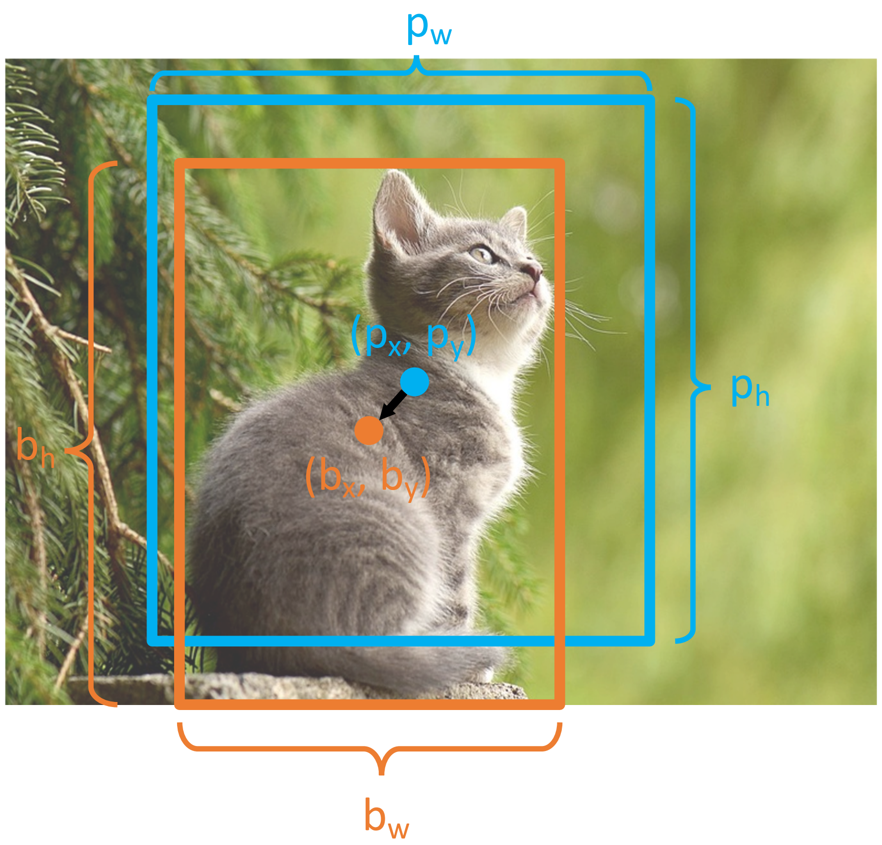
R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is defined by:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$

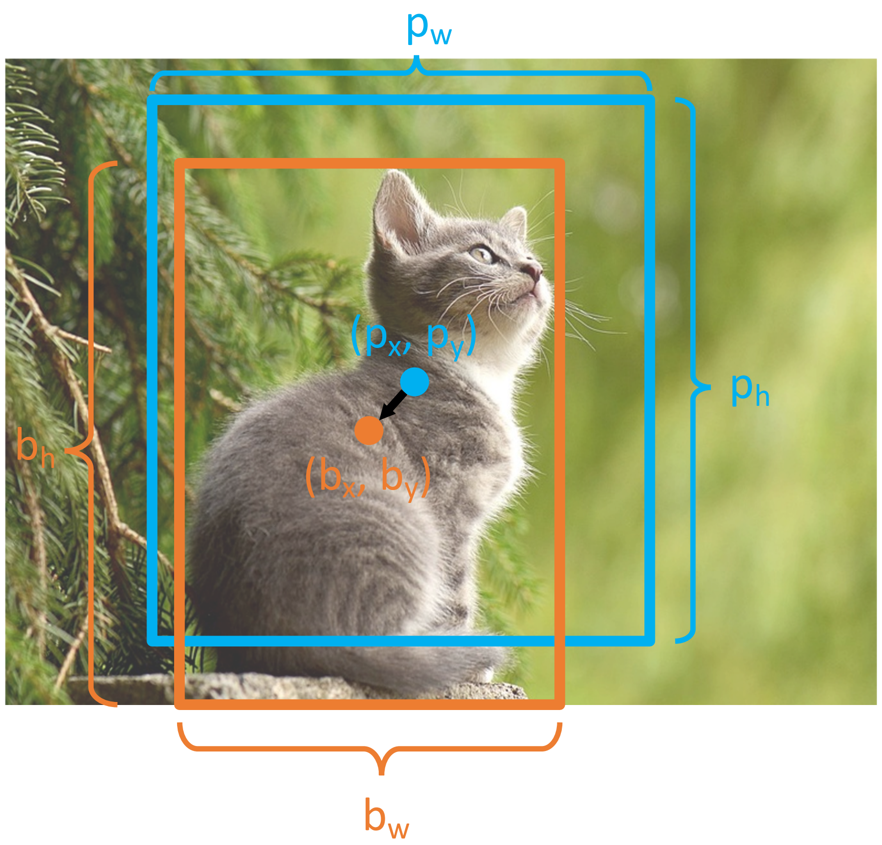
Shift center by amount relative to proposal size

$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

Scale proposal; exp ensures that scaling factor is > 0

Slide from Justin Johnson

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

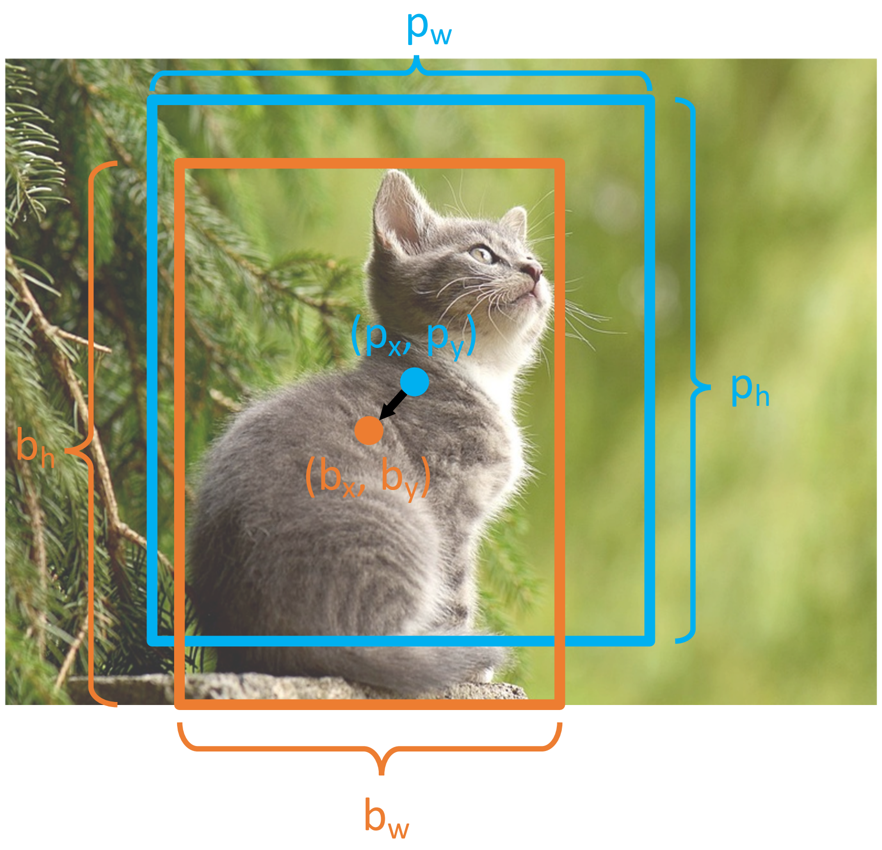
The output box is:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

When transform is 0, output = proposal

L2 regularization encourages leaving proposal unchanged

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

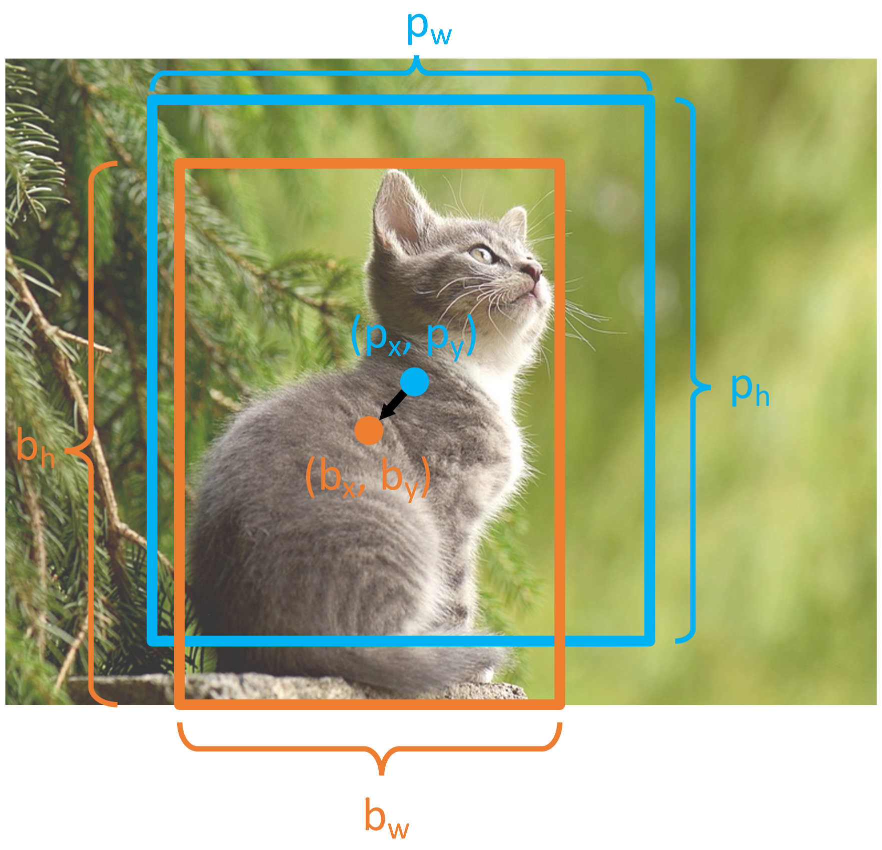
Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

Scale / Translation invariance: Transform encodes *relative* difference between proposal and output; important since CNN doesn't see absolute size or position after cropping

R-CNN: Box Regression



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

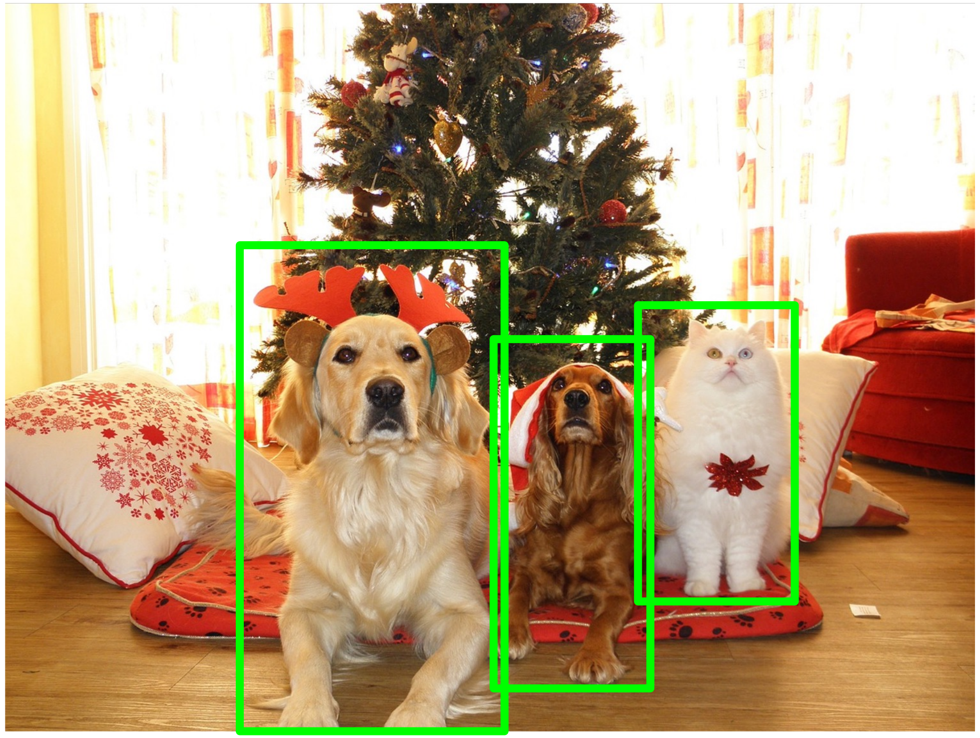
$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

Given proposal and target output, we can solve for the transform the network should output:

$$t_x = (b_x - p_x)/p_w$$
$$t_y = (b_y - p_y)/p_h$$
$$t_w = \log(b_w/p_w)$$
$$t_h = \log(b_h/p_h)$$

R-CNN Training

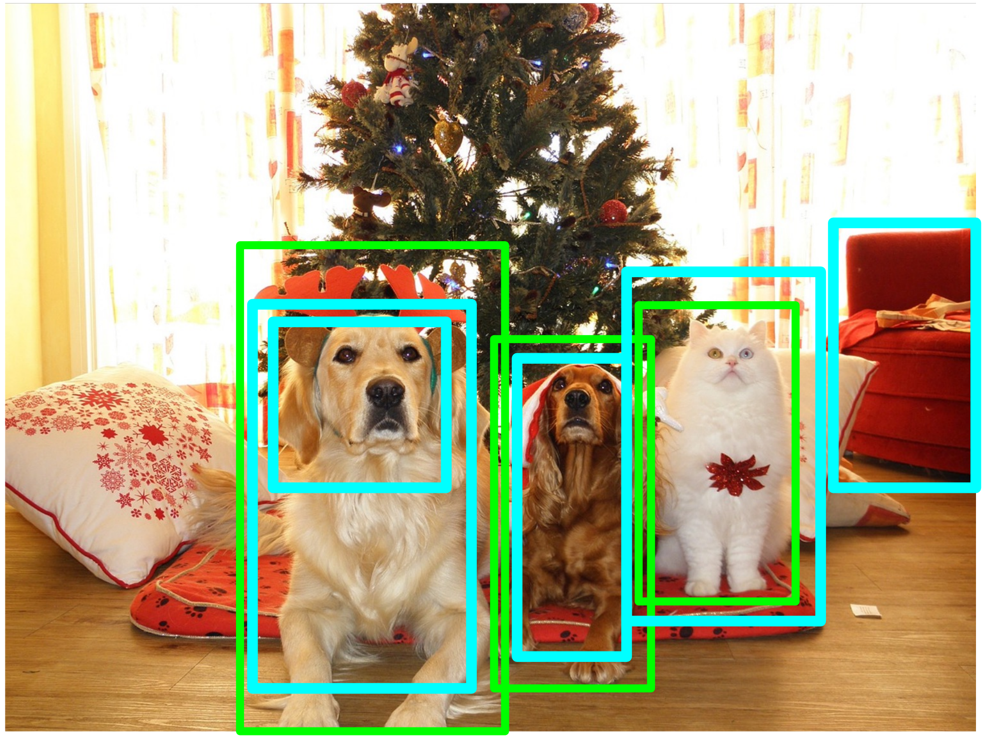
Input Image



Ground-Truth boxes

R-CNN Training

Input Image



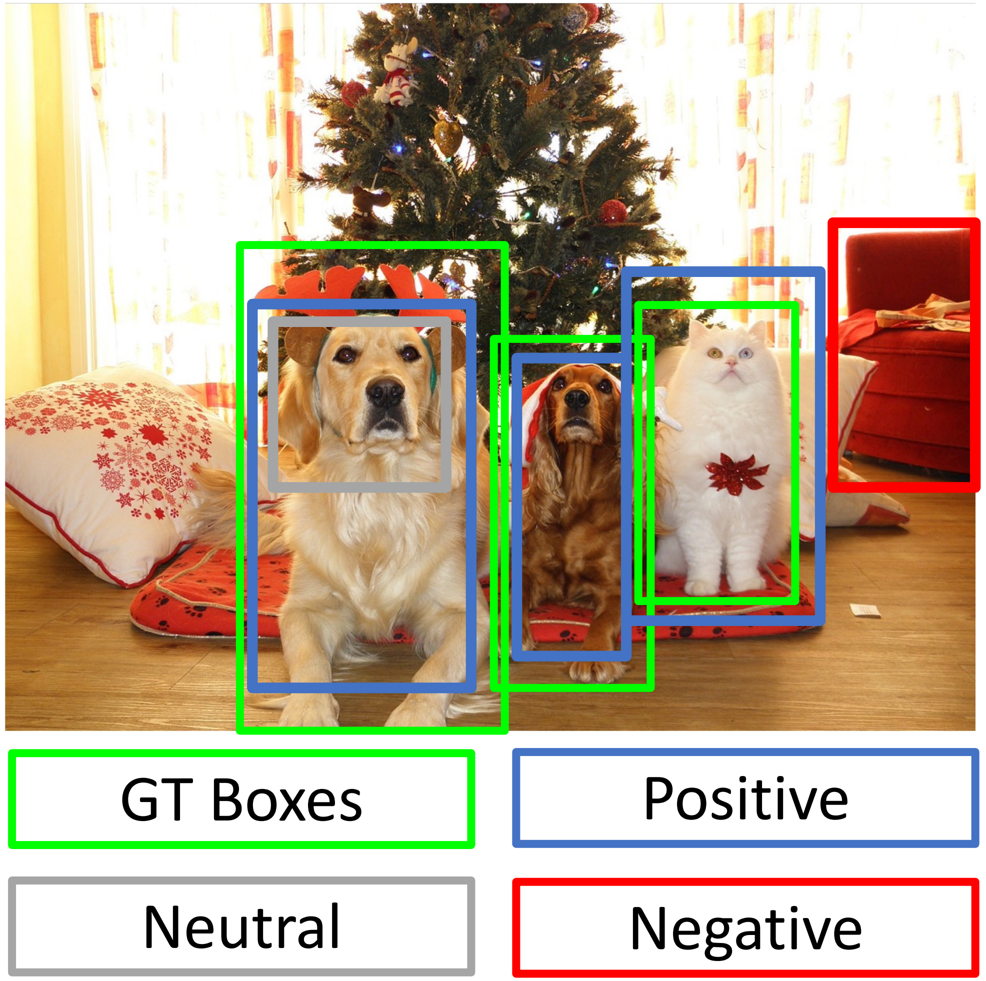
Ground-Truth boxes

Region Proposals

Slide from Justin Johnson

R-CNN Training

Input Image



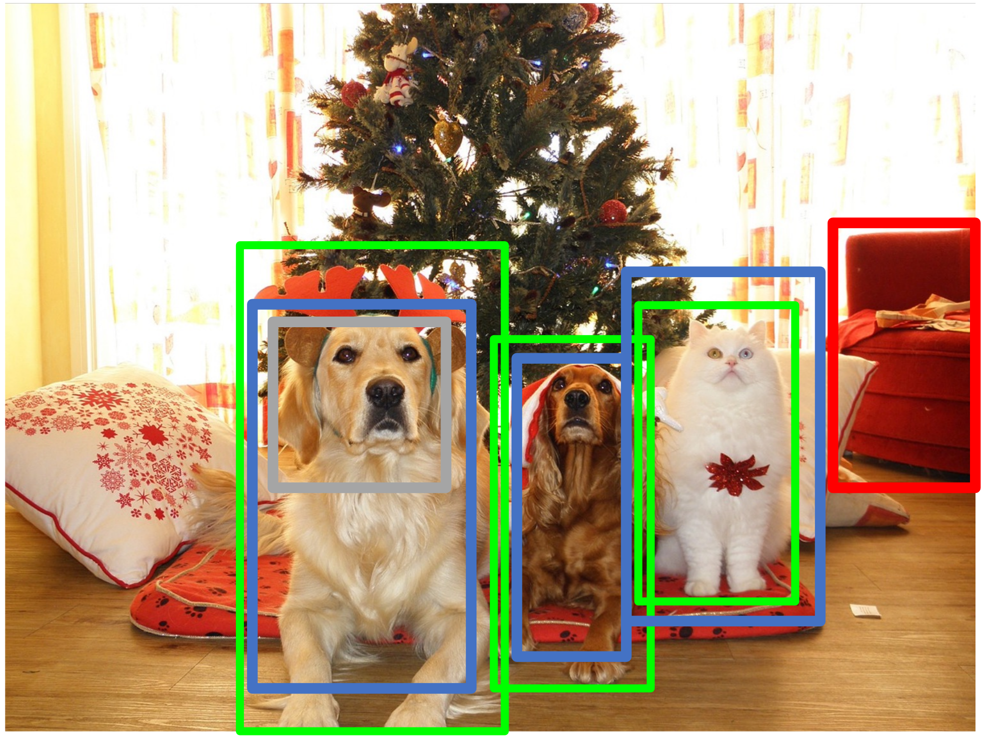
Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes:

Positive: > 0.5 IoU with a GT box
Negative: < 0.3 IoU with all GT boxes
Neutral: between 0.3 and 0.5 IoU with GT boxes

Slide from Justin Johnson

R-CNN Training

Input Image



GT Boxes

Positive

Neutral

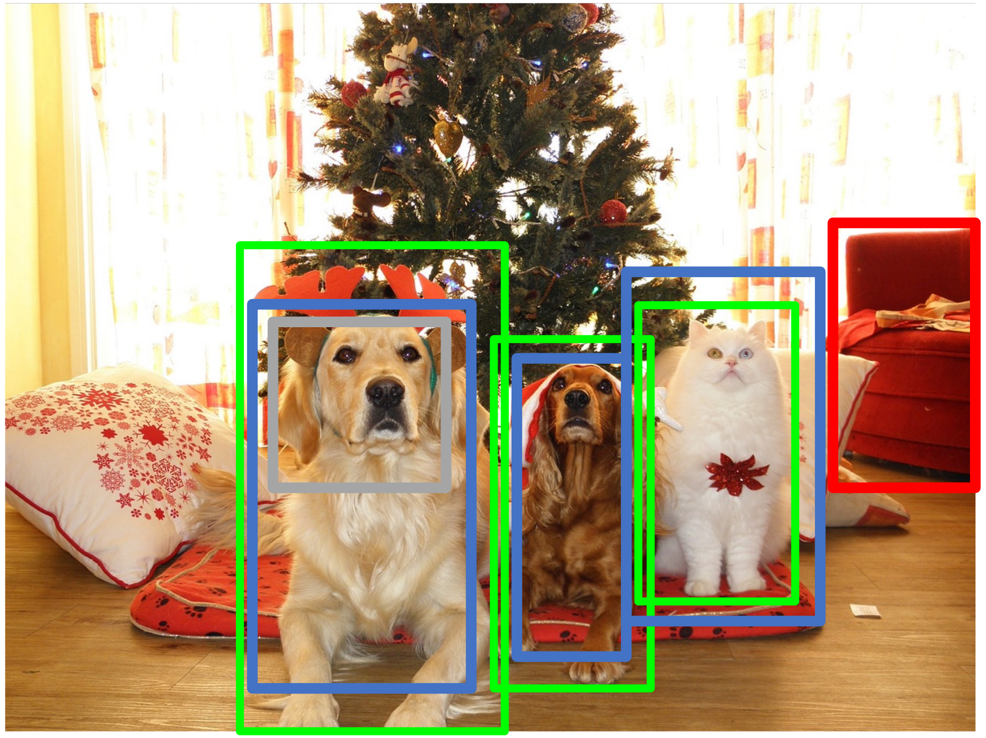
Negative

Crop pixels from each positive and negative proposal, resize to 224 x 224

Slide from Justin Johnson

R-CNN Training

Input Image



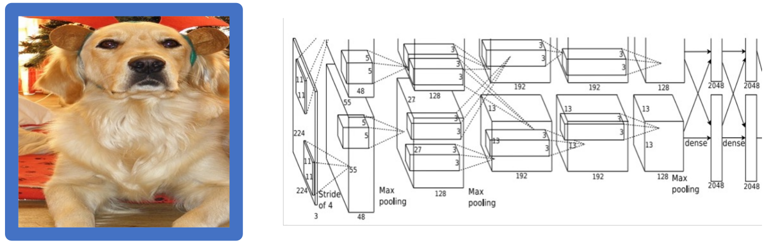
GT Boxes

Positive

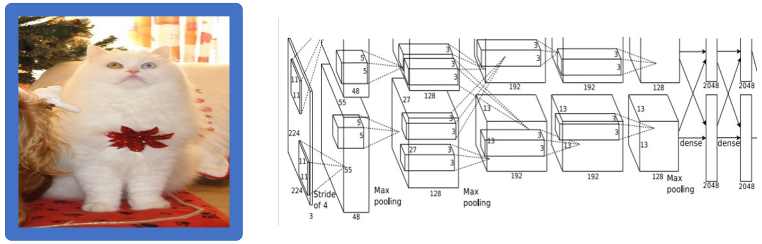
Neutral

Negative

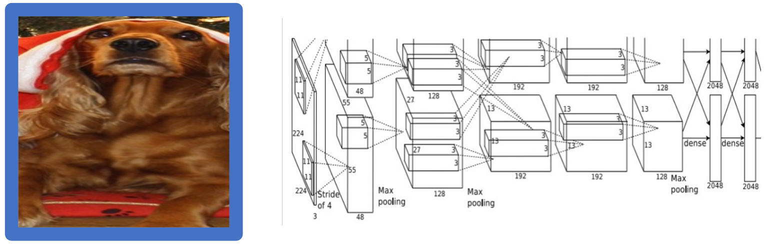
Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class



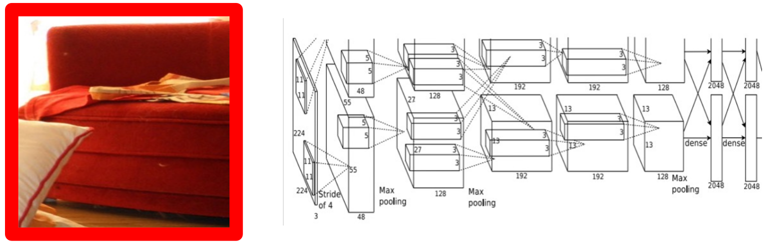
Class target: Dog
Box target: $\longrightarrow$



Class target: Cat
Box target: $\longrightarrow$



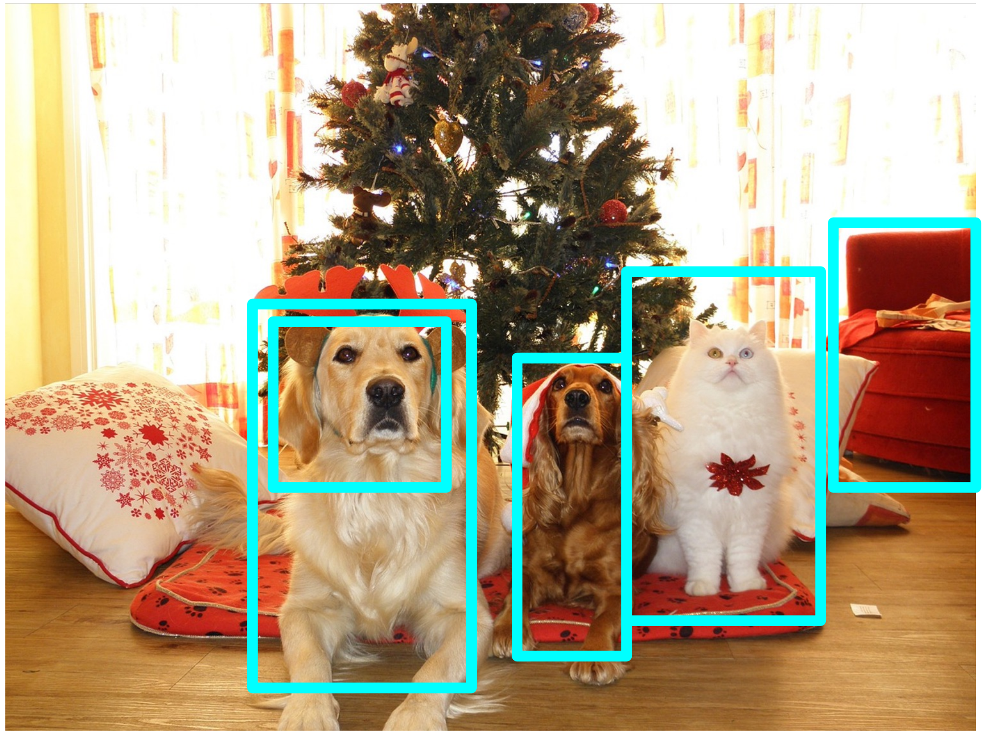
Class target: Dog
Box target: $\longrightarrow$



Class target: Background
Box target: None

R-CNN Test-Time

Input Image



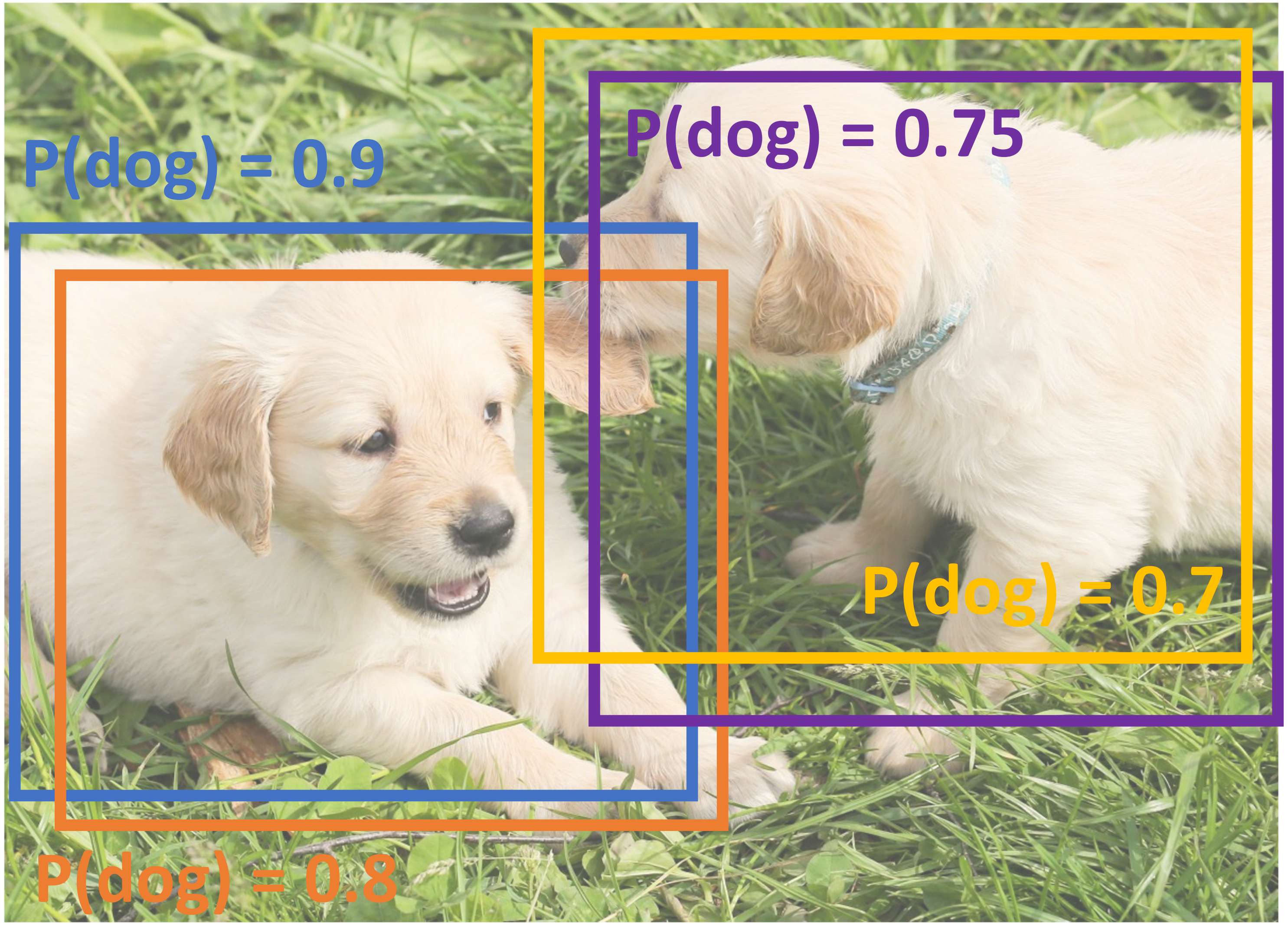
Region Proposals

1. Run proposal method
2. Run CNN on each proposal to get class scores, transforms
3. Threshold class scores to get a set of detections

2 problems:
- CNN often outputs overlapping boxes
- How to set thresholds?

Overlapping Boxes

Problem: Object detectors often output many overlapping detections:



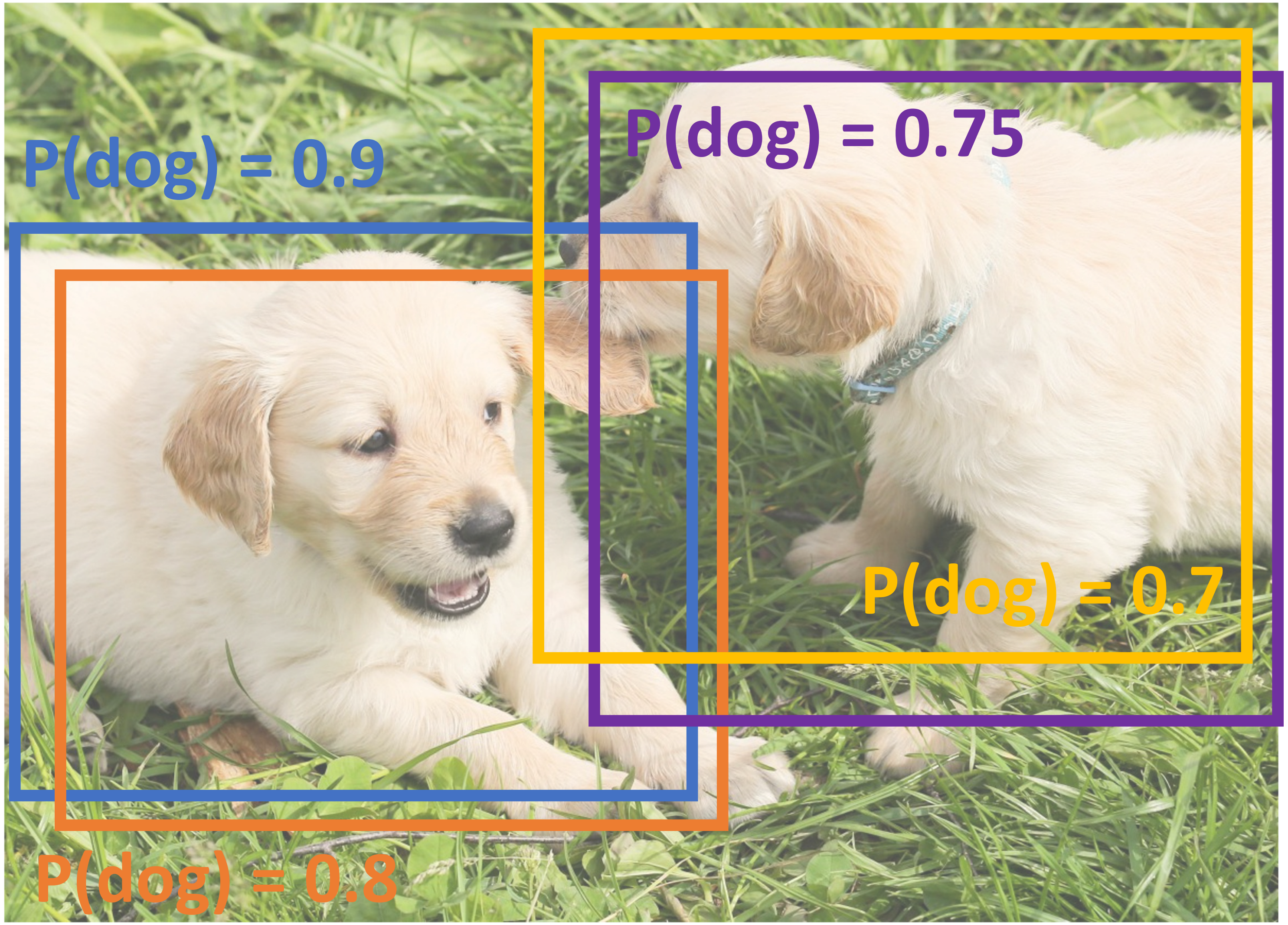
Puppy image is CC0 Public Domain

Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



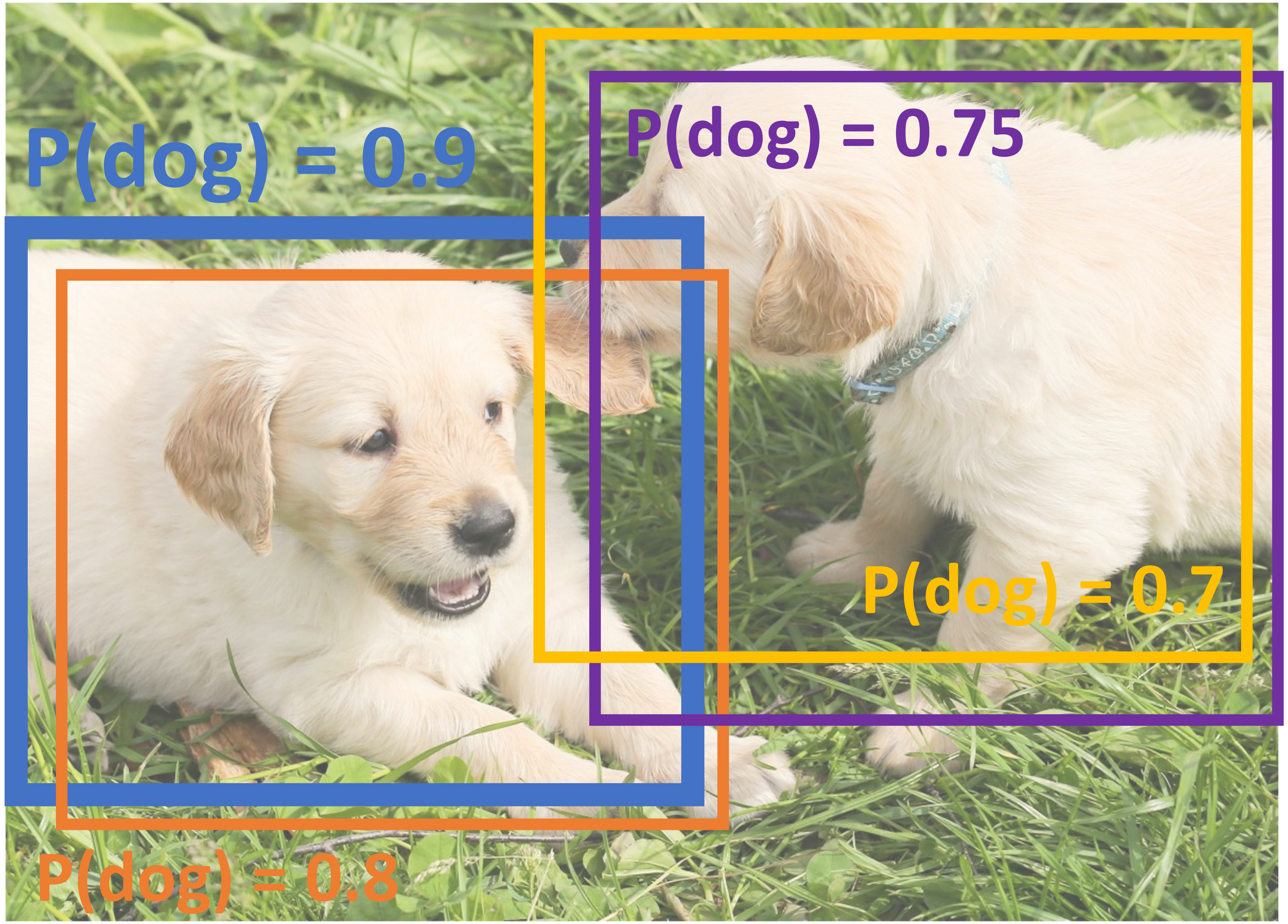
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



Slide from Justin Johnson

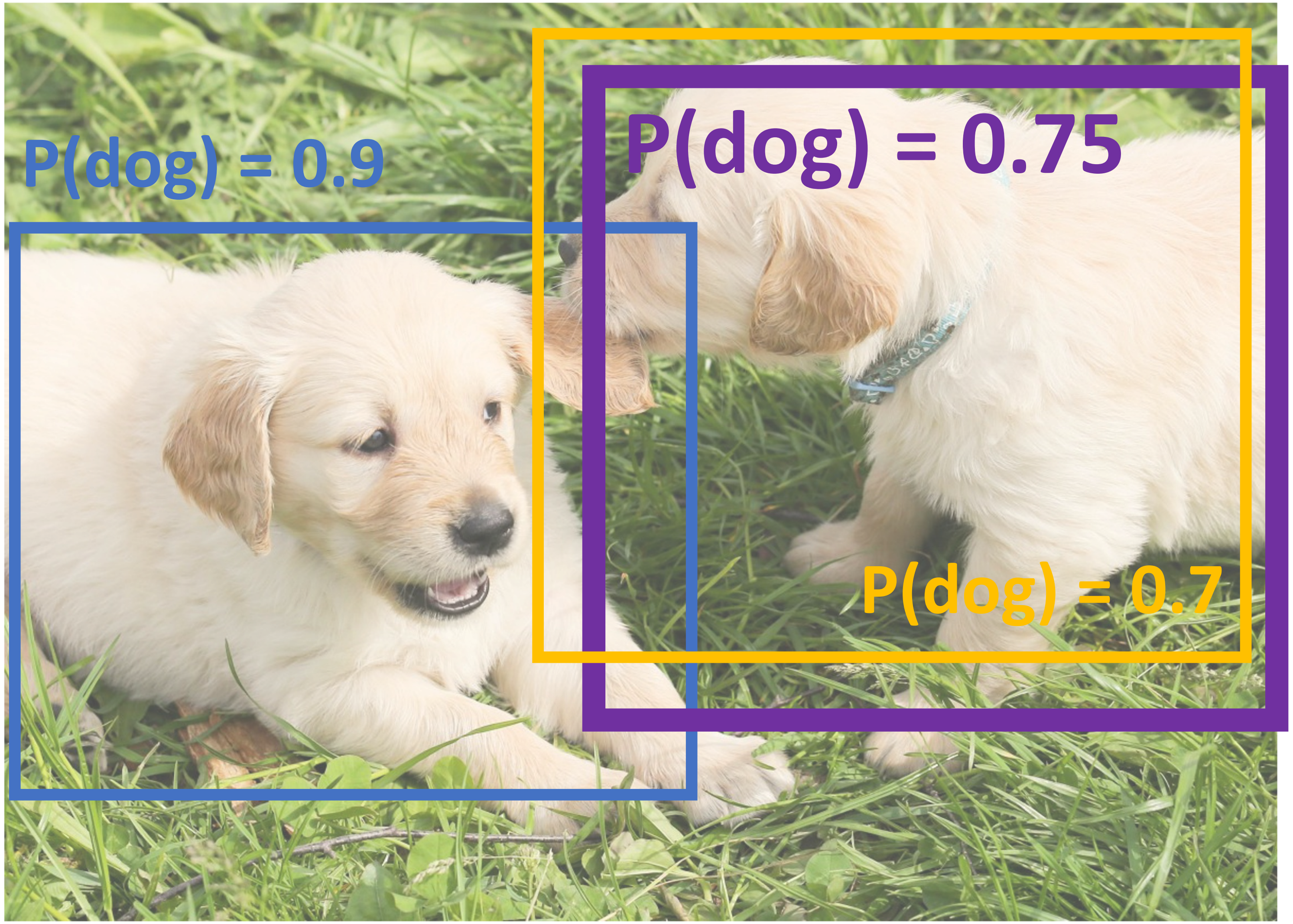
Overlapping Boxes: **Non-Max Suppression (NMS)**

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.74**



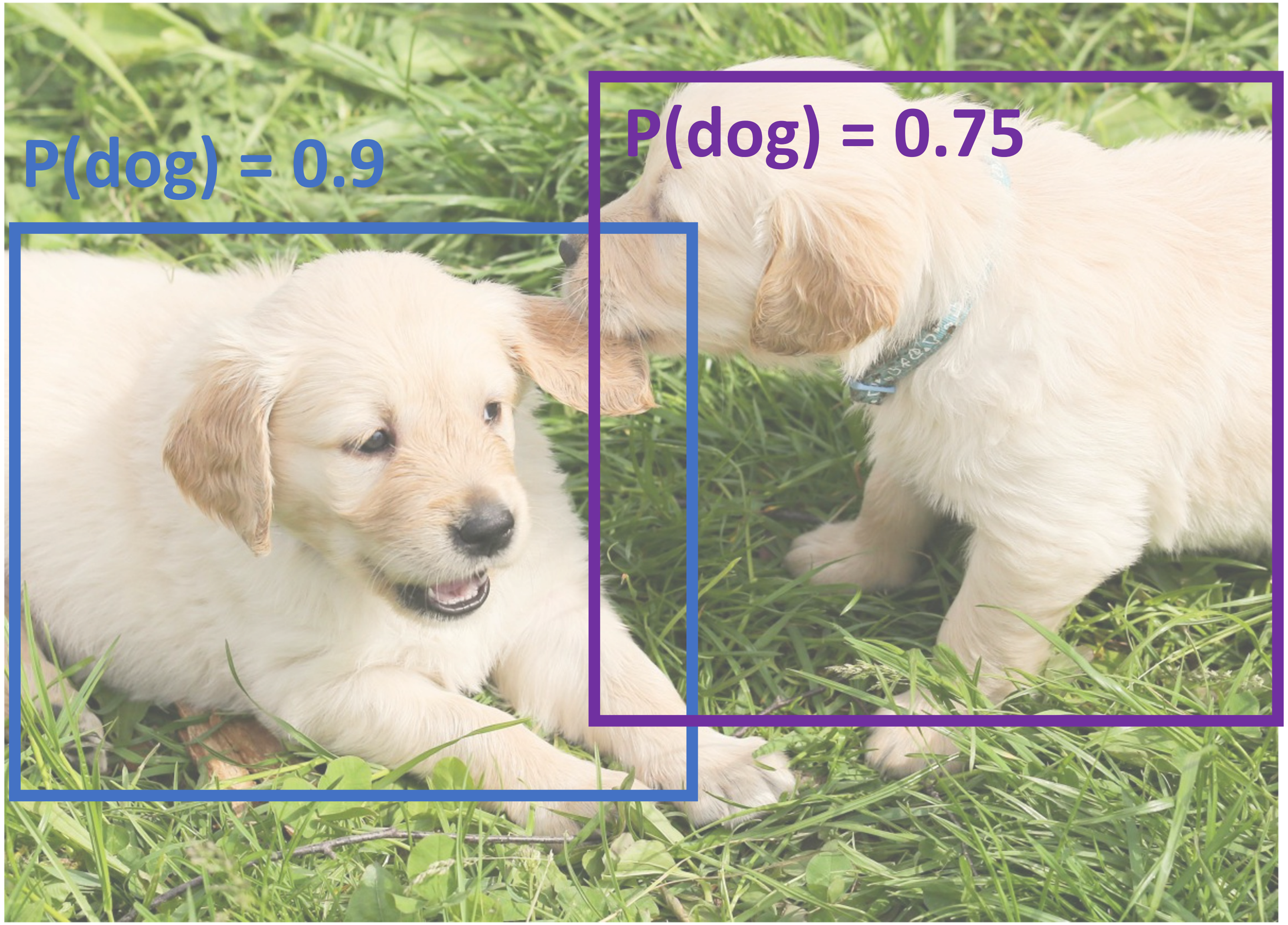
Slide from Justin Johnson

Overlapping Boxes: **Non-Max Suppression (NMS)**

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Slide from Justin Johnson

Overlapping Boxes: **Non-Max Suppression (NMS)**

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

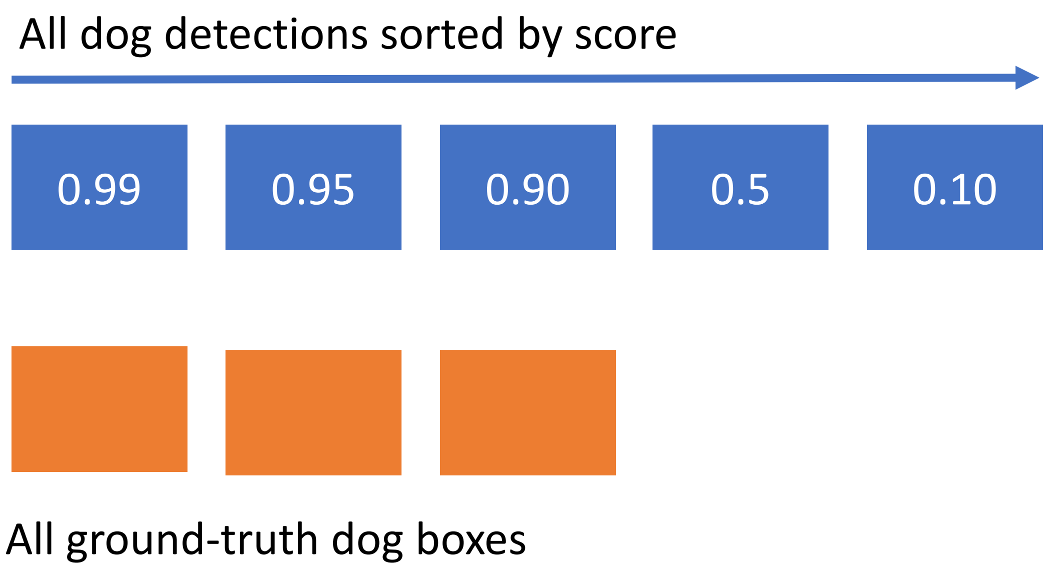
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Problem: NMS may eliminate ”good” boxes when objects are highly overlapping… no good solution =(

Evaluating Object Detectors: Mean Average Precision (mAP)

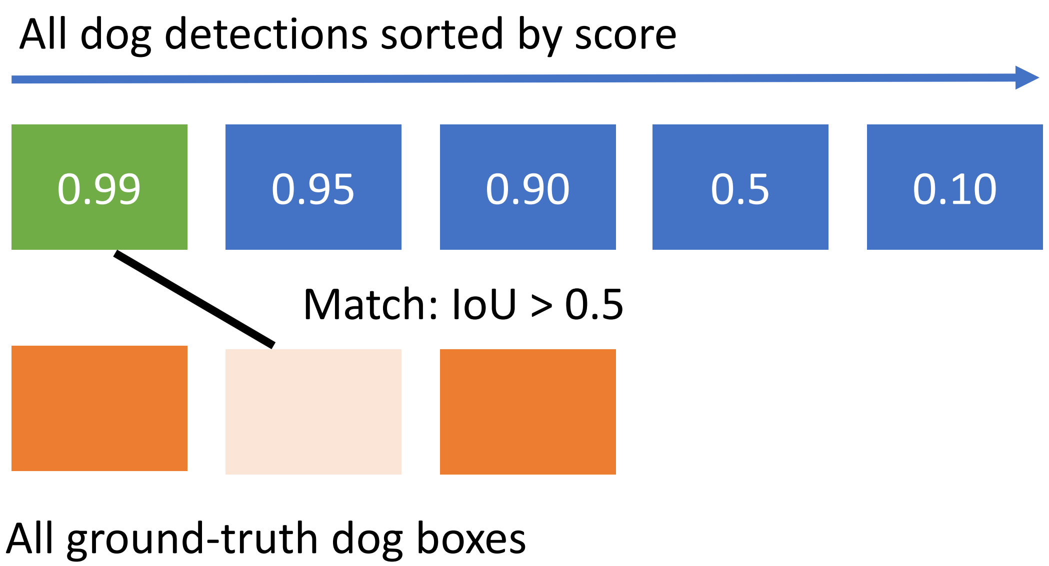
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve

Evaluating Object Detectors: Mean Average Precision (mAP)



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)

Evaluating Object Detectors: Mean Average Precision (mAP)

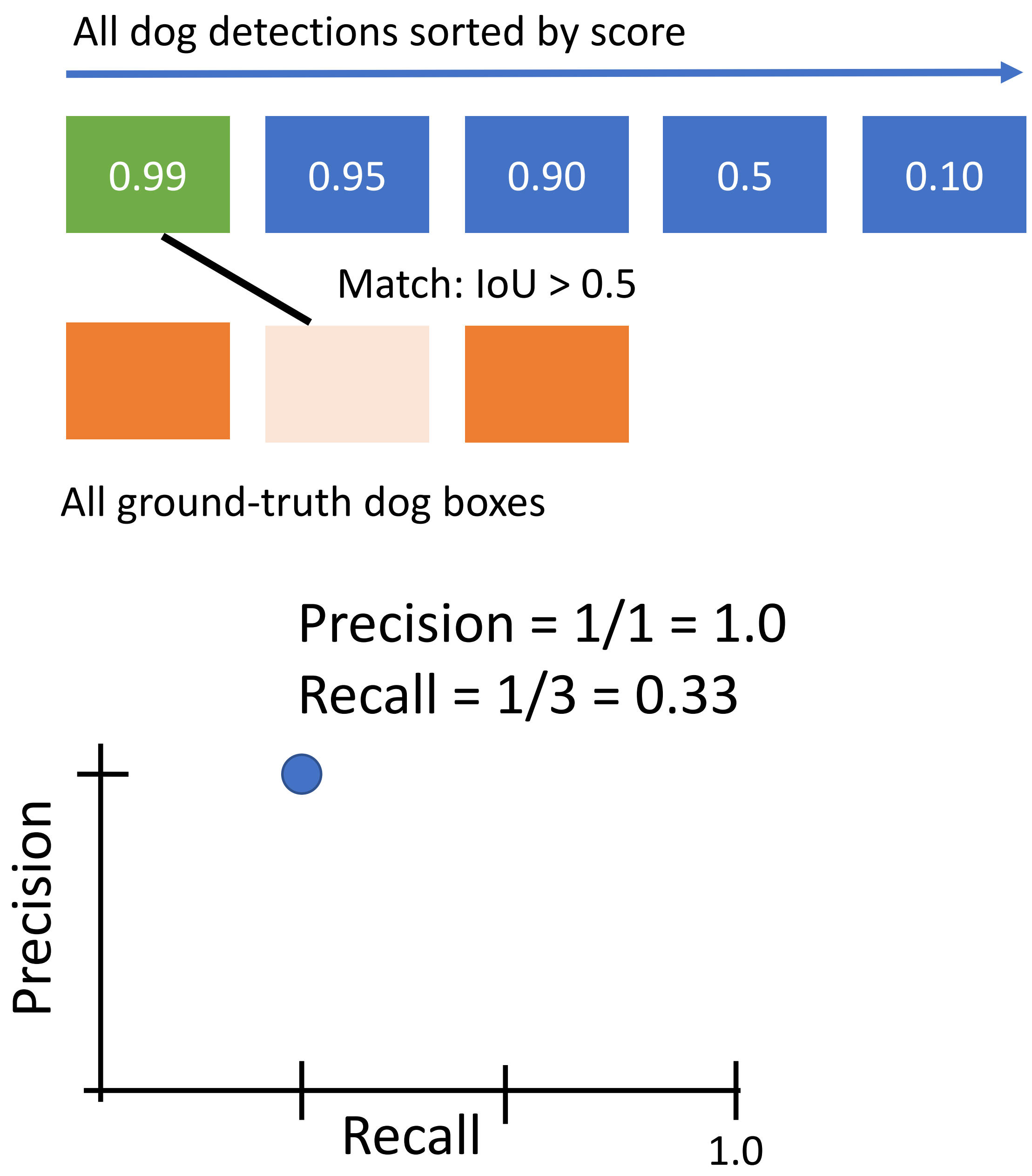


1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative

Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)

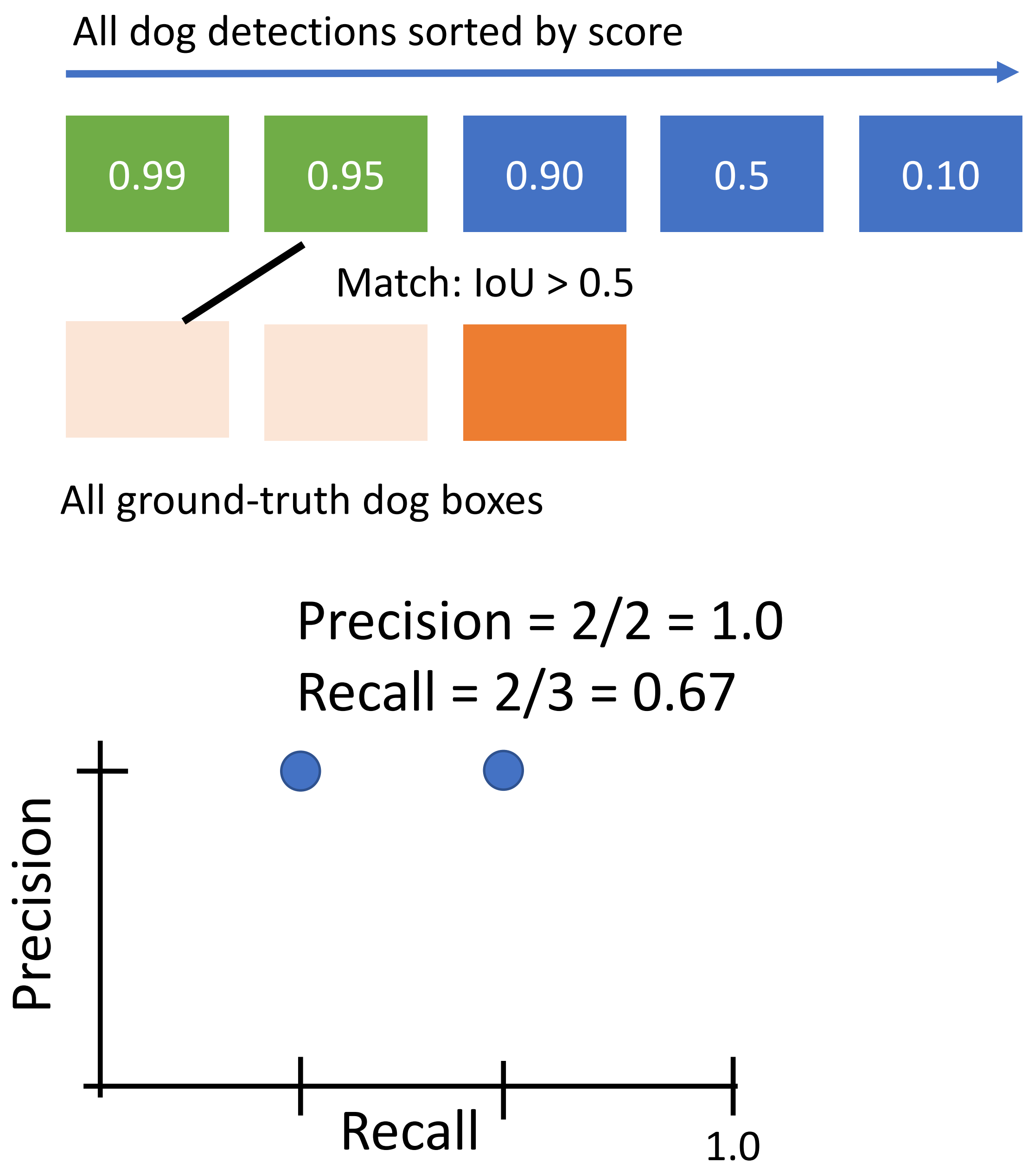
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)

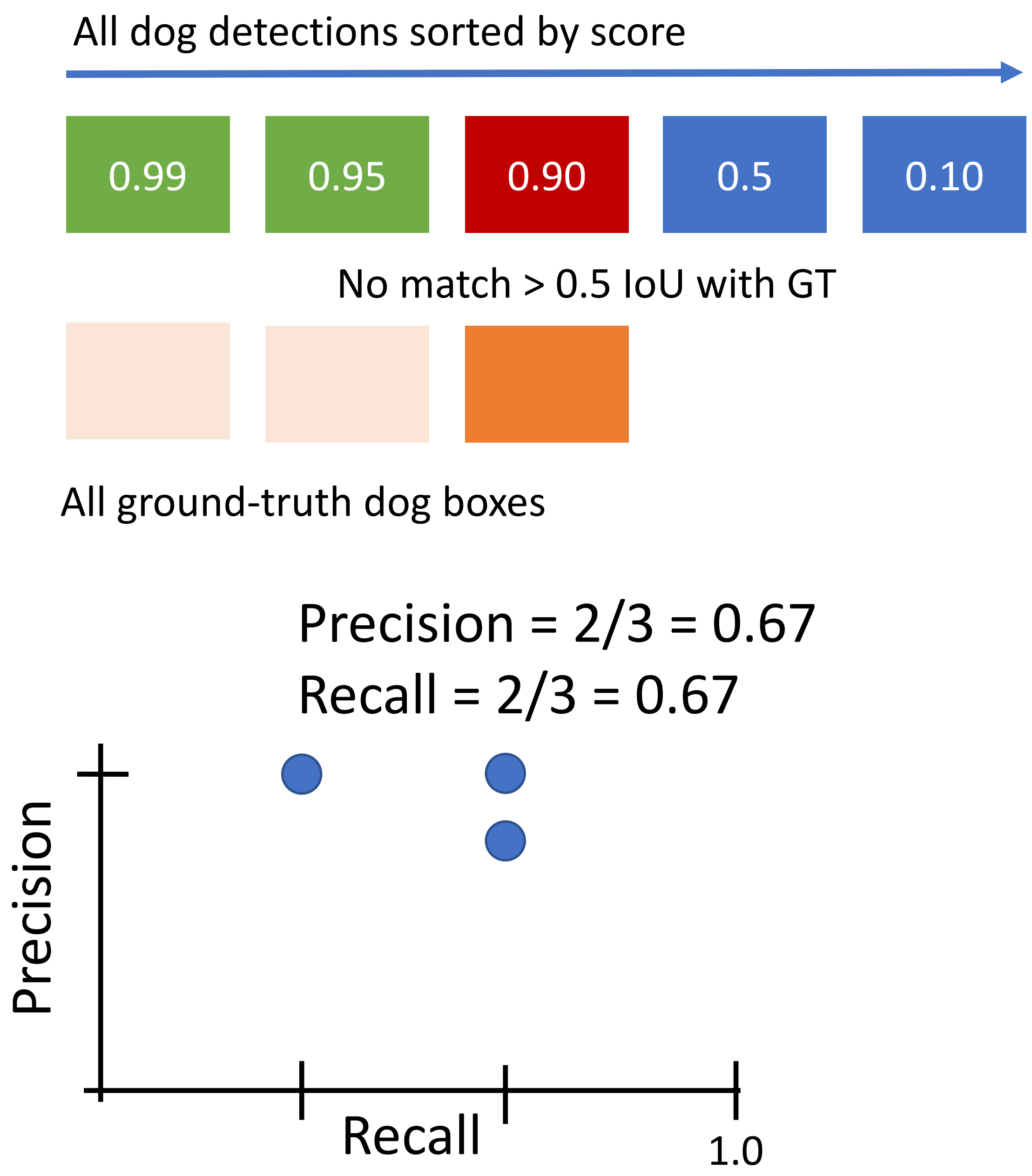
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)

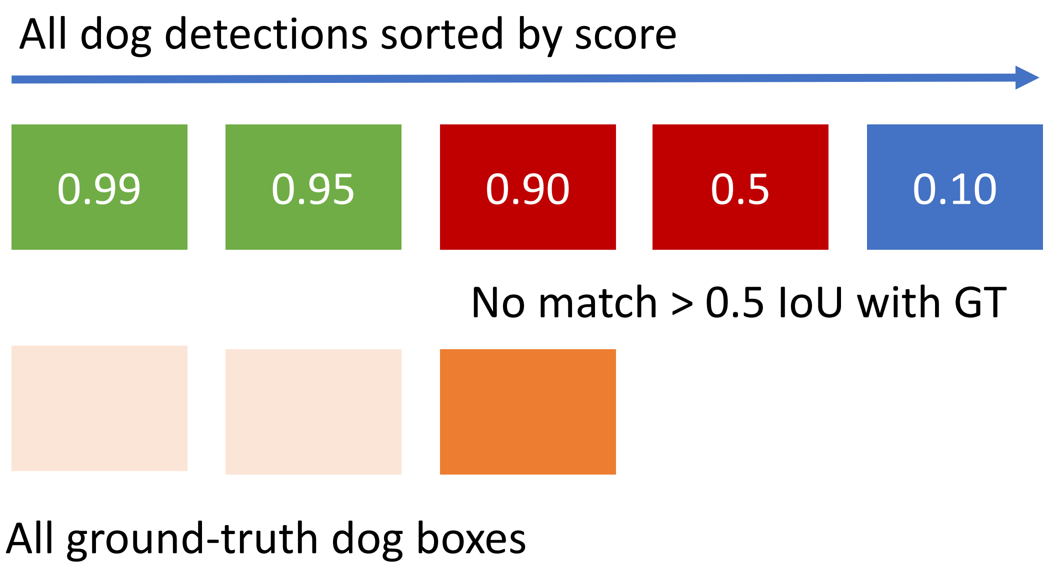
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve



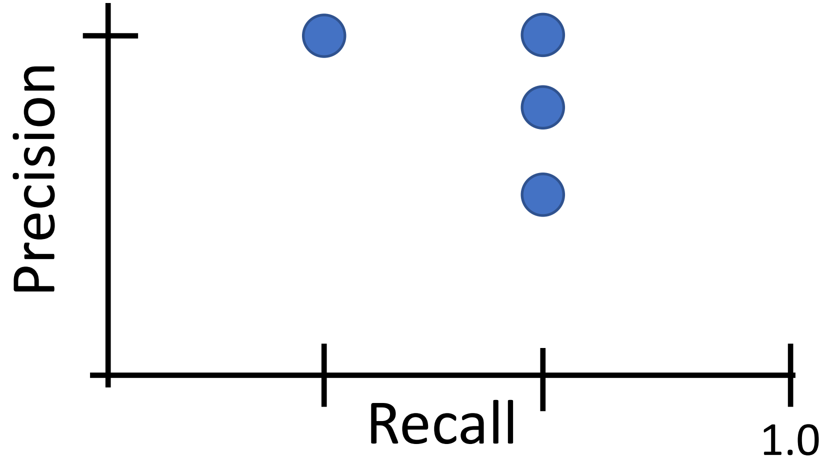
Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

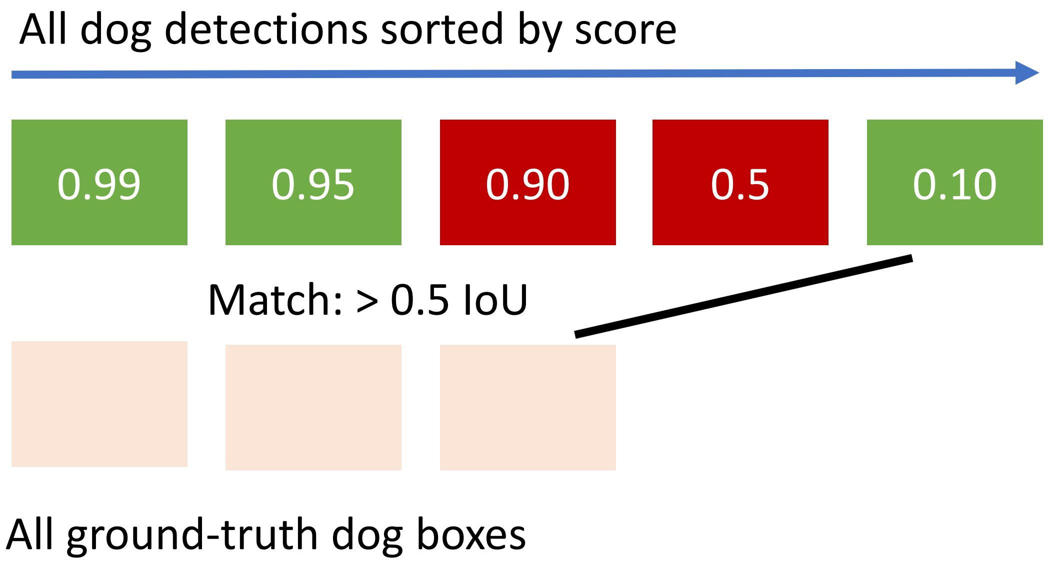


Precision = 2/4 = 0.5
Recall = 2/3 = 0.67

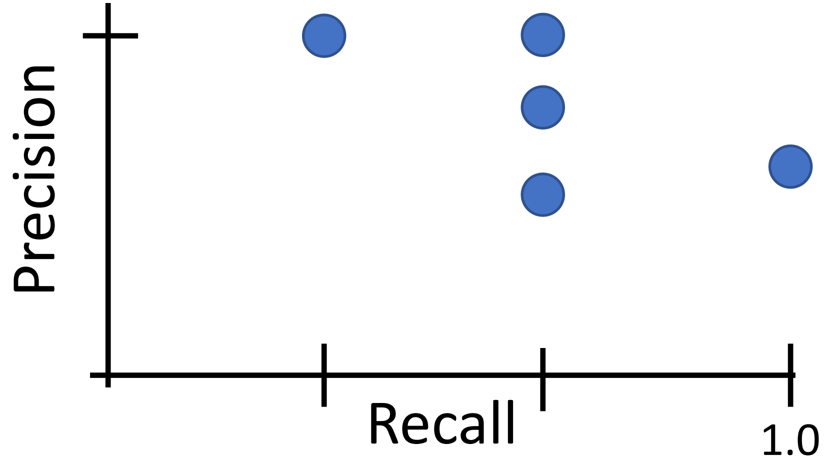


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



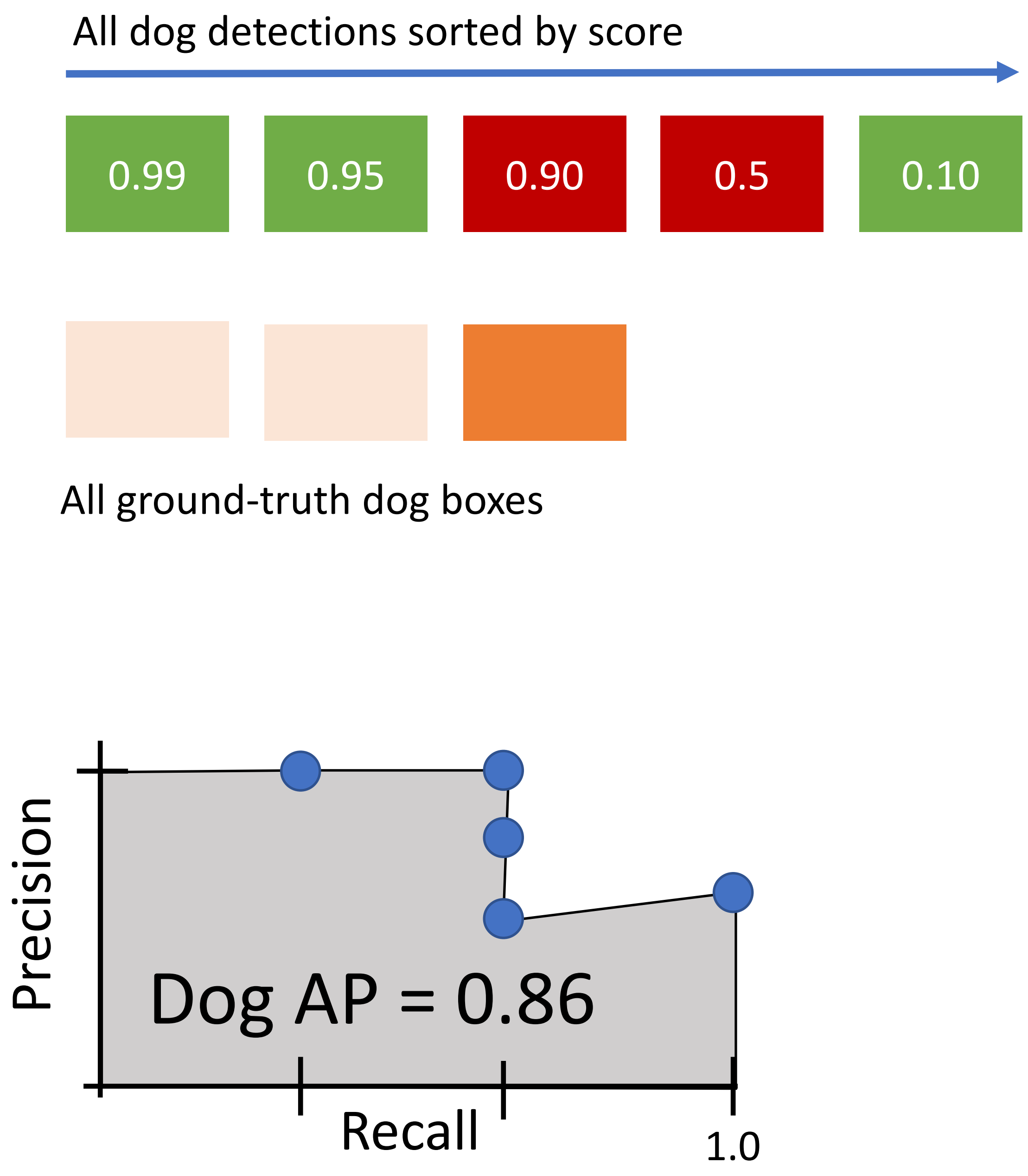
Precision = 3/5 = 0.6
Recall = 3/3 = 1.0



Slide from Justin Johnson

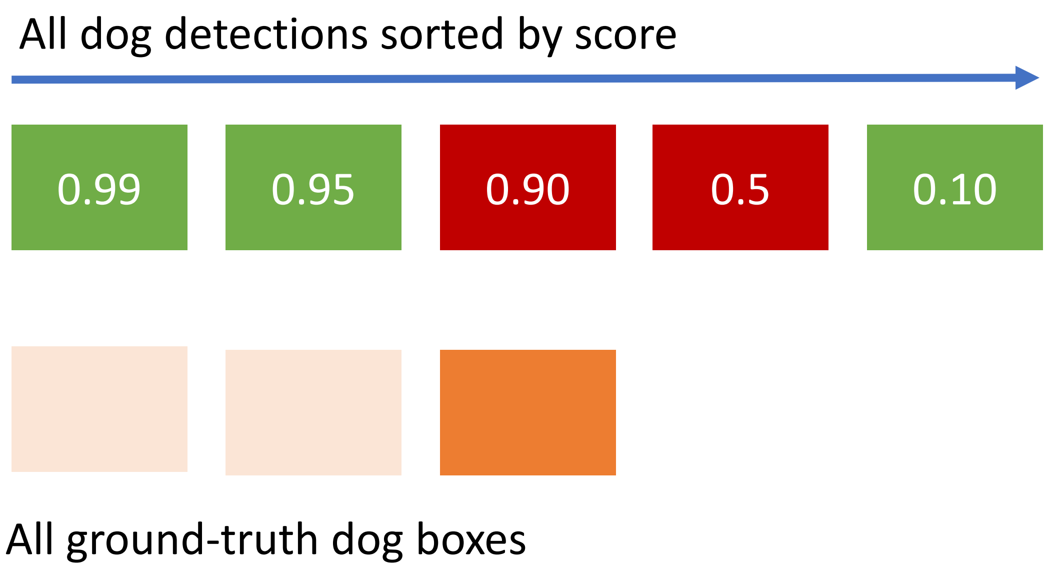
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve



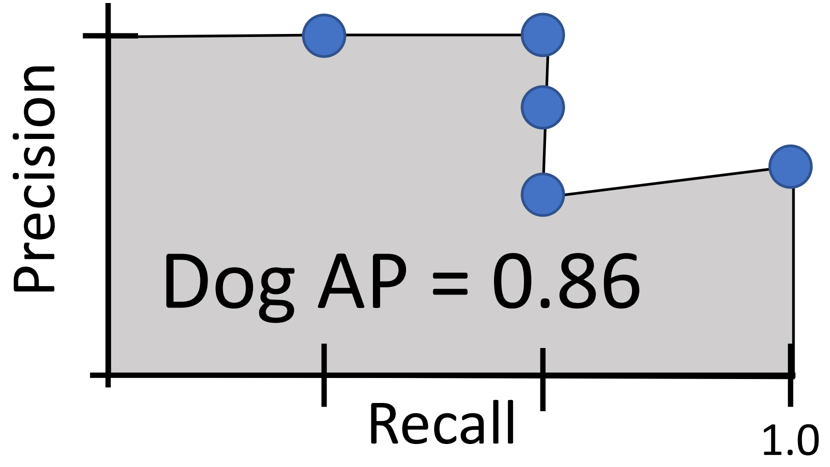
Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



Slide from Justin Johnson

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.2

COCO mAP = 0.4

Slide from Justin Johnson

RCNN Results

VOC 2010 test	mAP
DPM v5 [20]†	33.4
UVA [39]	35.1
Regionlets [41]	39.7
SegDPM [18]†	40.4
R-CNN	50.2
R-CNN BB	**53.7**

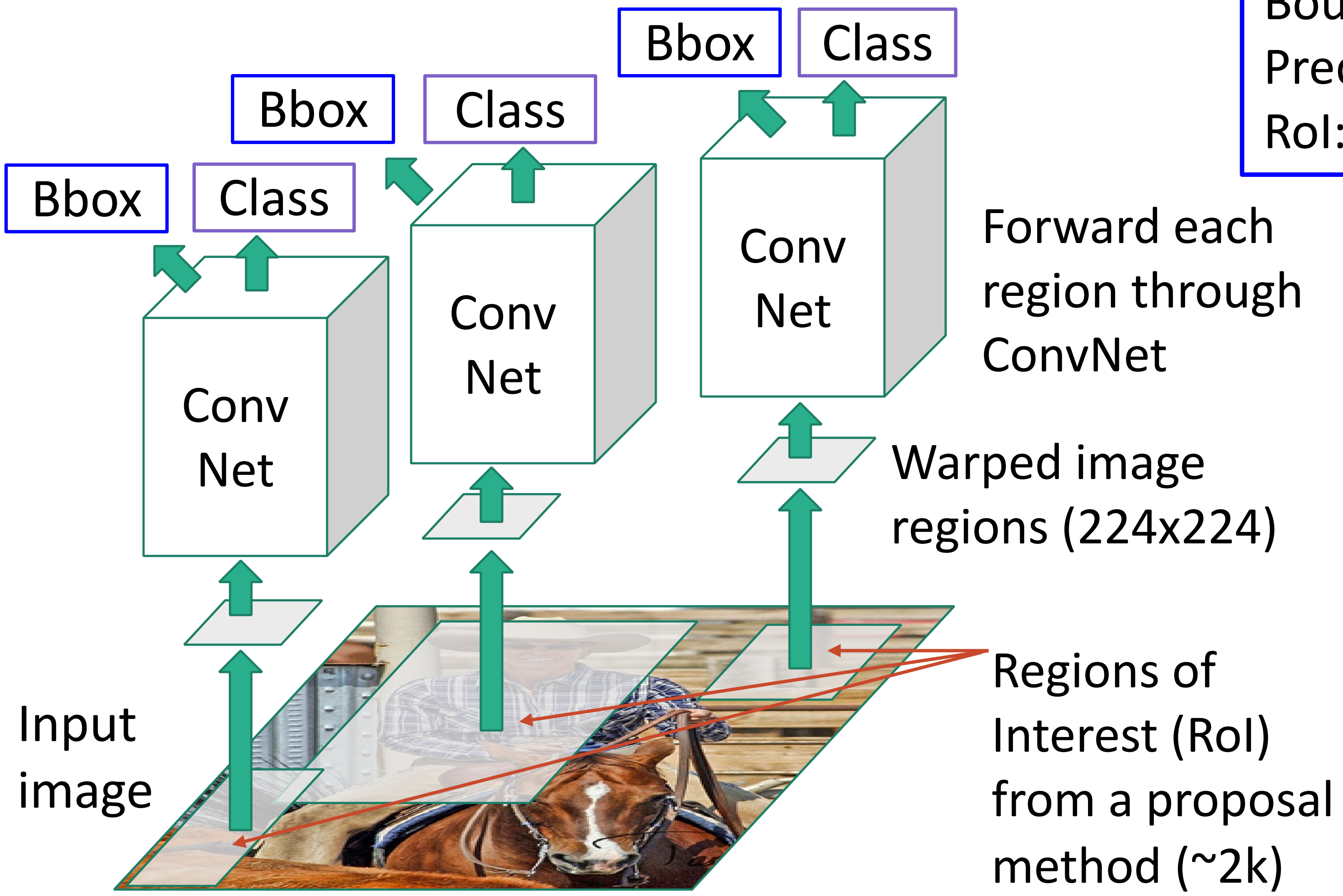
VOC 2007 test	mAP
R-CNN pool_5	44.2
R-CNN fc_6	46.2
R-CNN fc_7	44.7
R-CNN FT pool_5	47.3
R-CNN FT fc_6	53.1
R-CNN FT fc_7	54.2
R-CNN FT fc_7 BB	**58.5**
DPM v5 [20]	33.7
DPM ST [28]	29.1
DPM HSC [31]	34.3

VOC 2007 test	mAP
R-CNN T-Net	54.2
R-CNN T-Net BB	58.5
R-CNN O-Net	62.2
R-CNN O-Net BB	**66.0**

Last Time: R-CNN

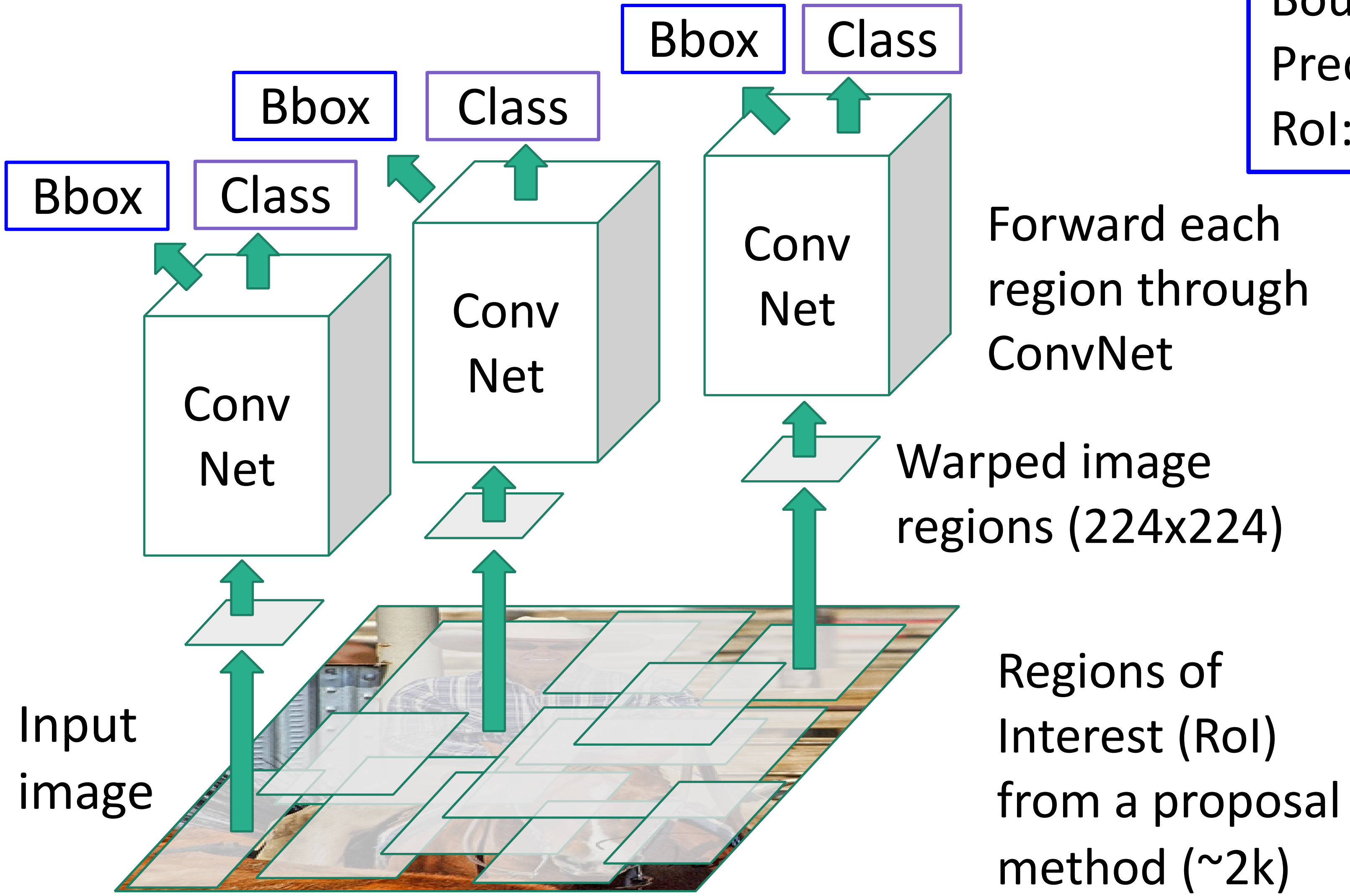
Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)



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

Last Time: R-CNN



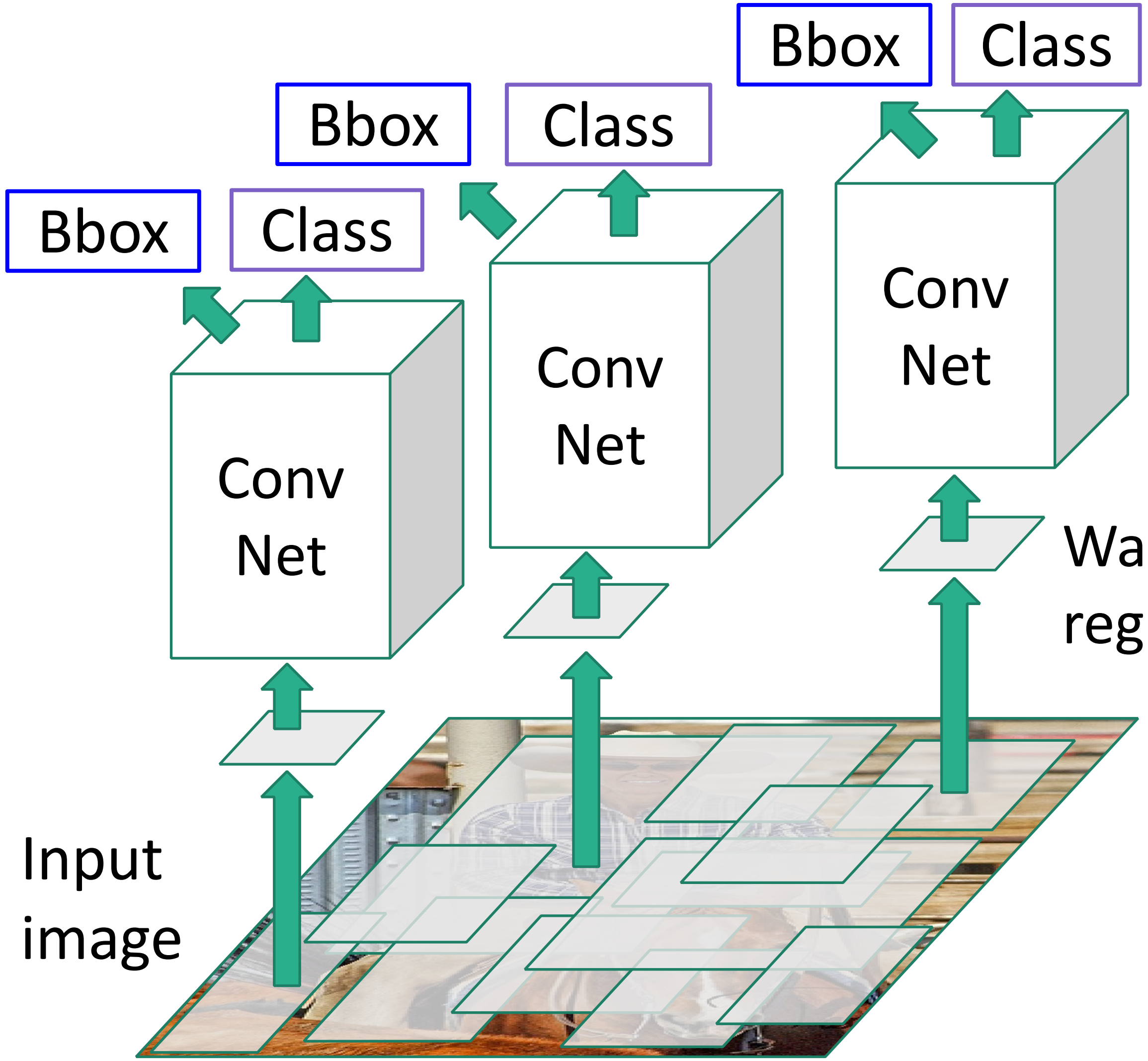
Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Problem: Very slow! Need to do 2000 forward passes through CNN per image

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

Last Time: R-CNN



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

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

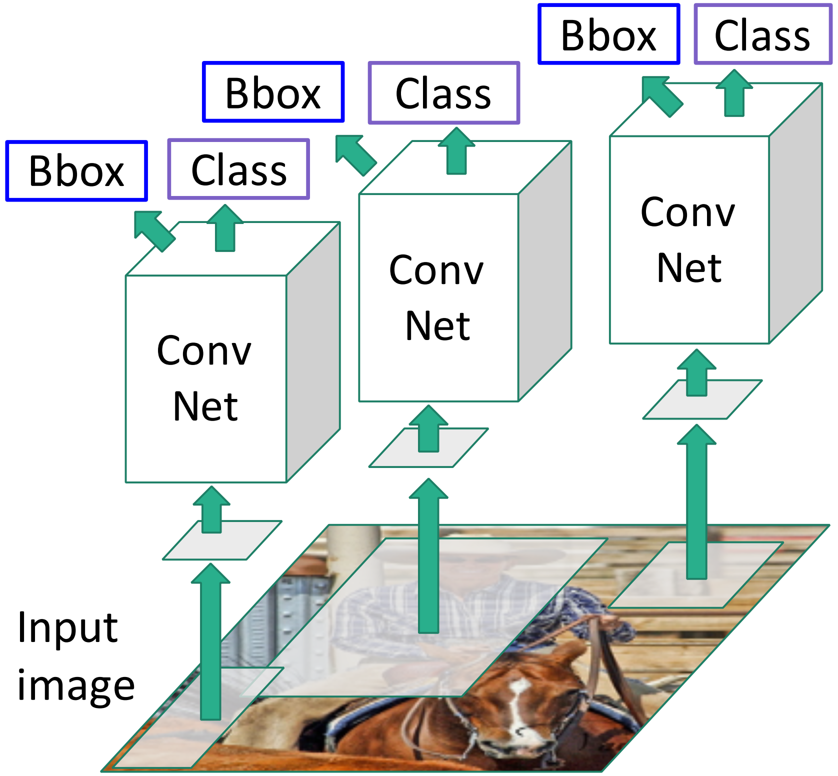
Problem: Very slow! Need to do 2000 forward passes through CNN per image

Idea: Overlapping proposals cause a lot of repeated work: same pixels processed many times. Can we avoid this?

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

"Slow" R-CNN

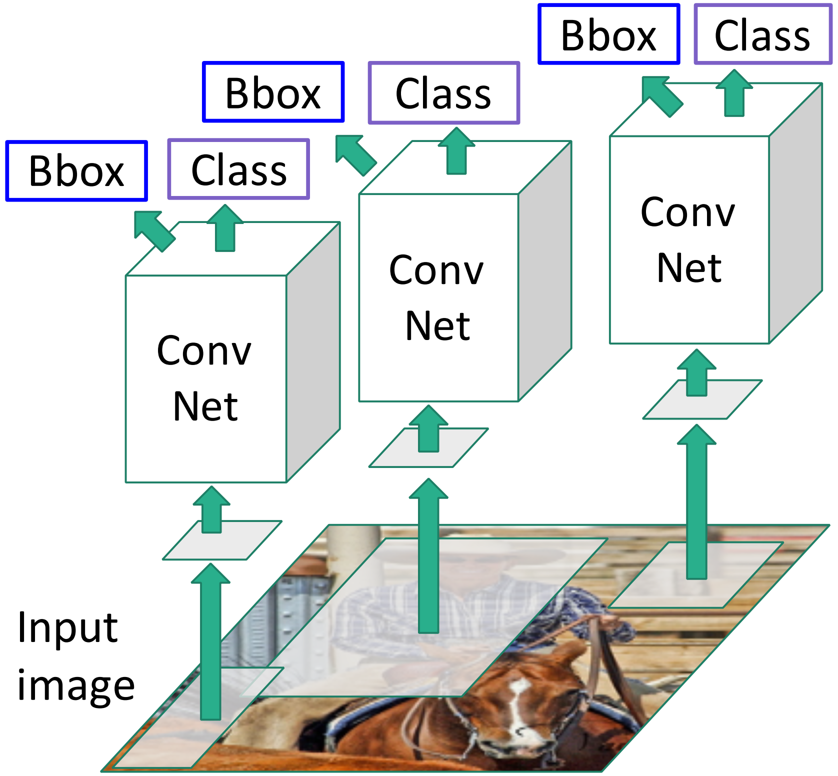
Process each region independently



Fast R-CNN

Input image

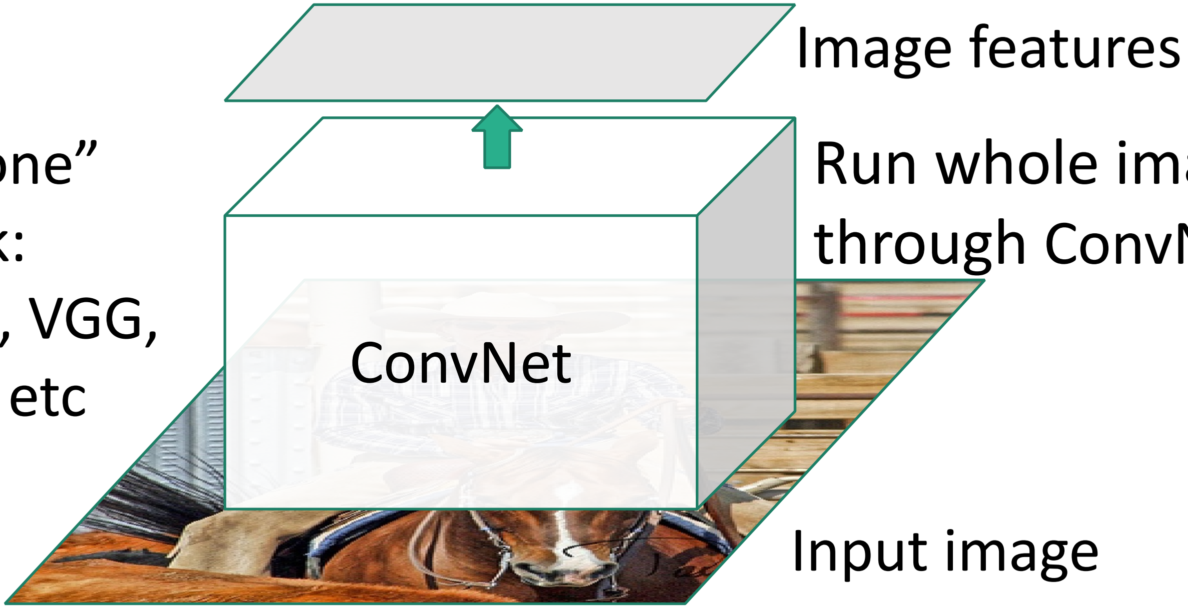
"Slow" R-CNN
Process each region independently



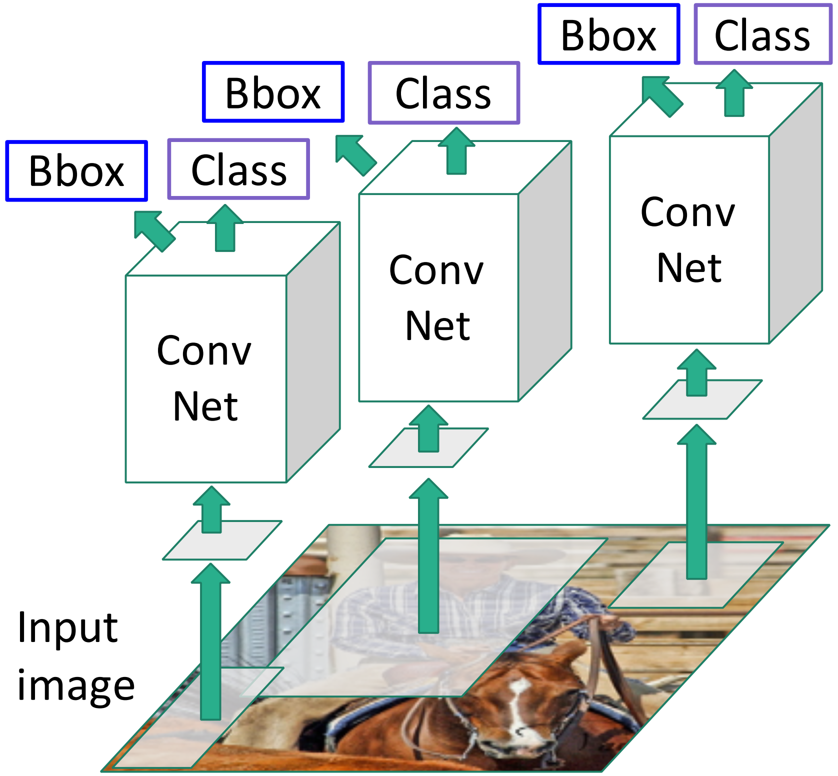
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; [source](). Reproduced with permission.

Fast R-CNN

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



"Slow" R-CNN
Process each region independently



Fast R-CNN

Regions of Interest (RoIs) from a proposal method

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

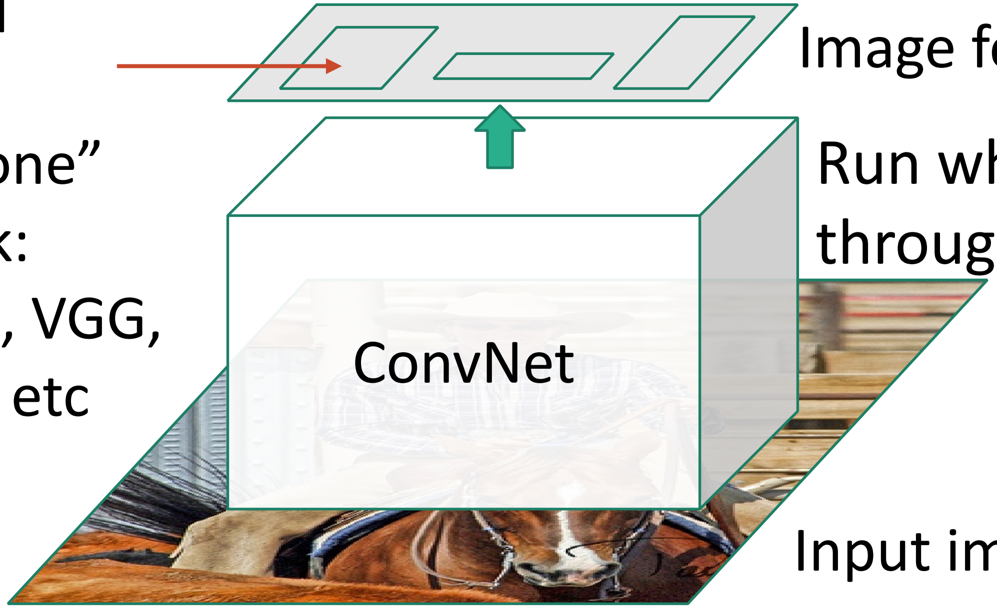


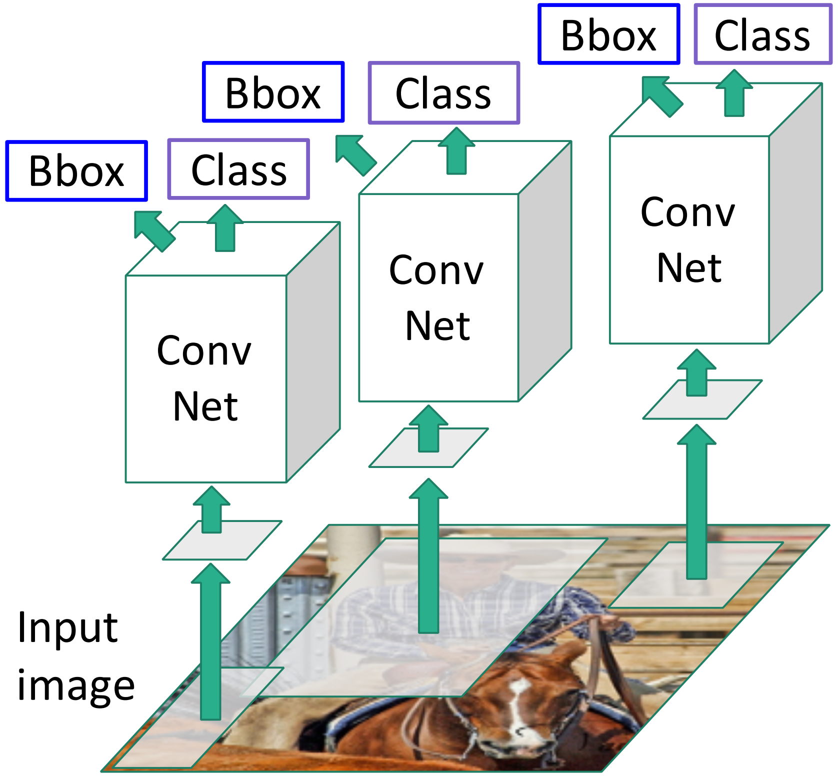
Image features

Run whole image through ConvNet

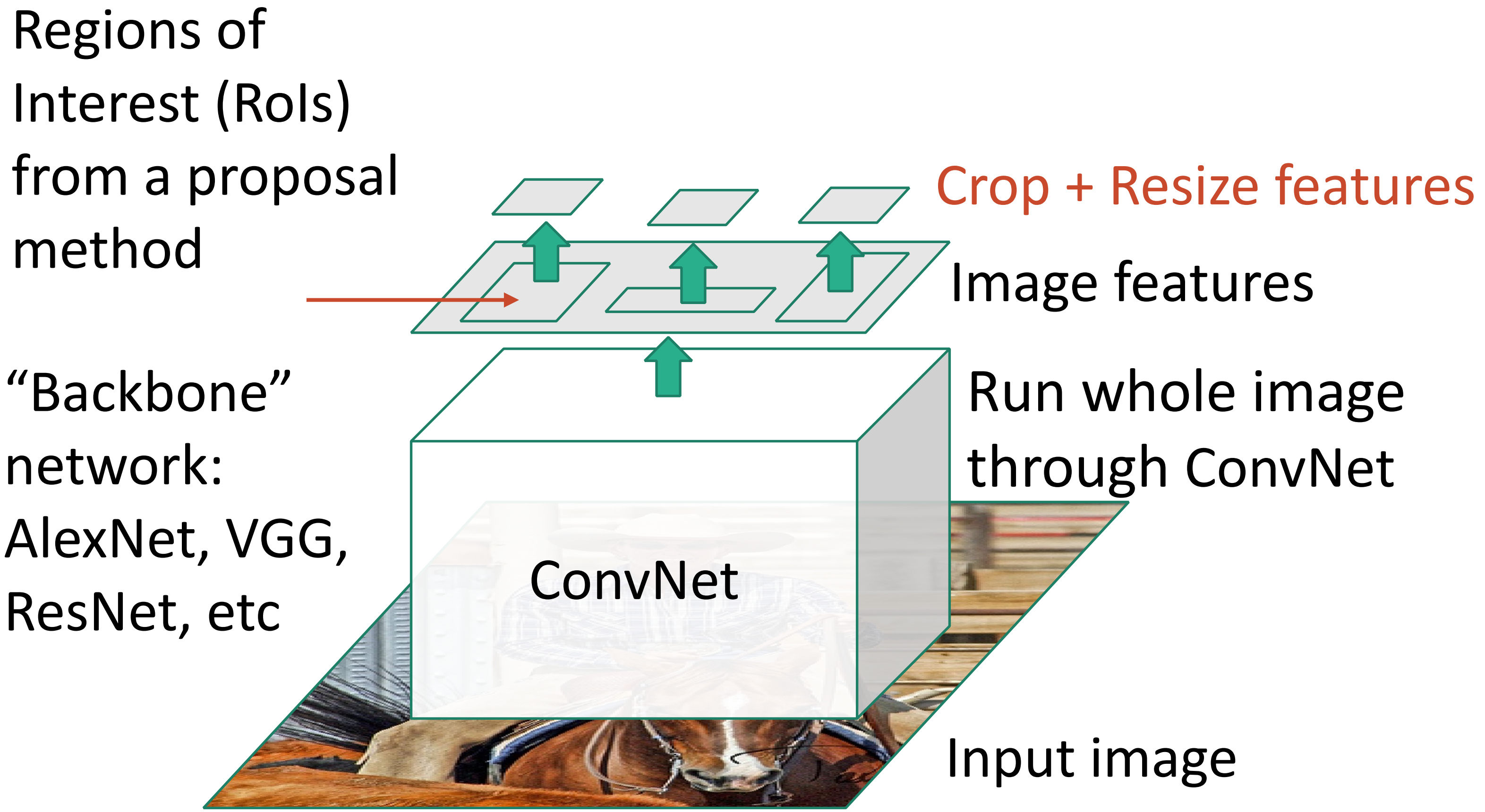
Input image

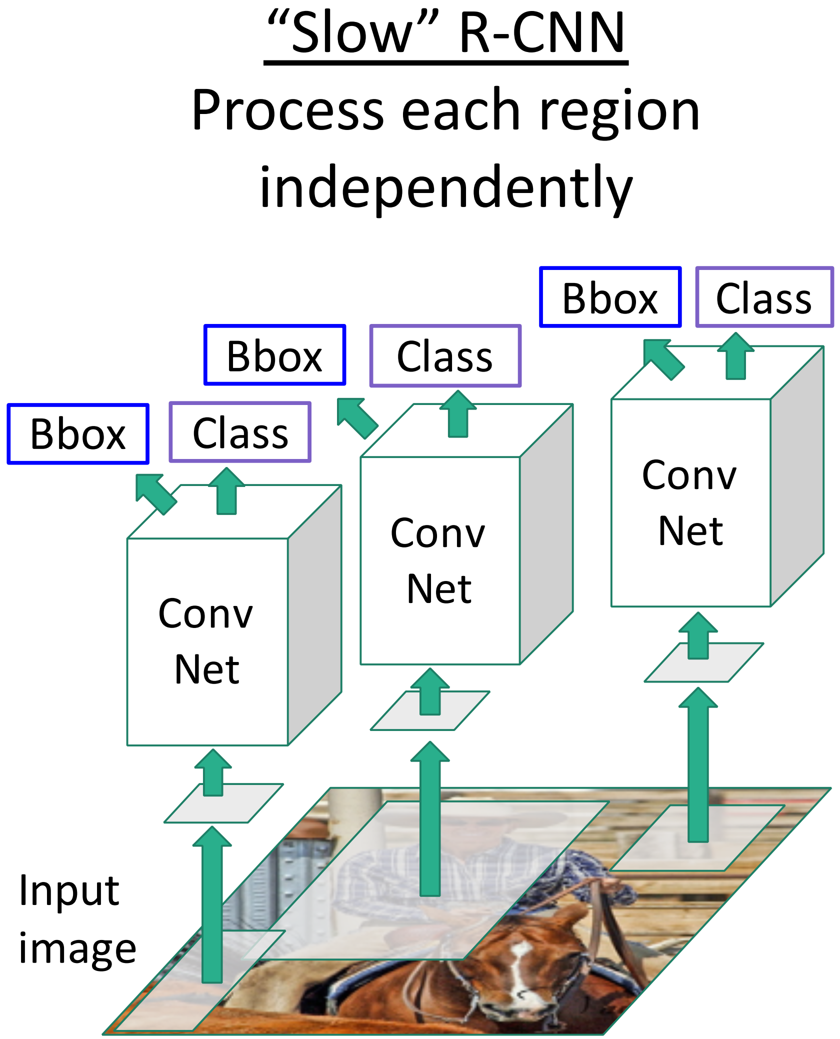
“Slow” R-CNN

Process each region independently



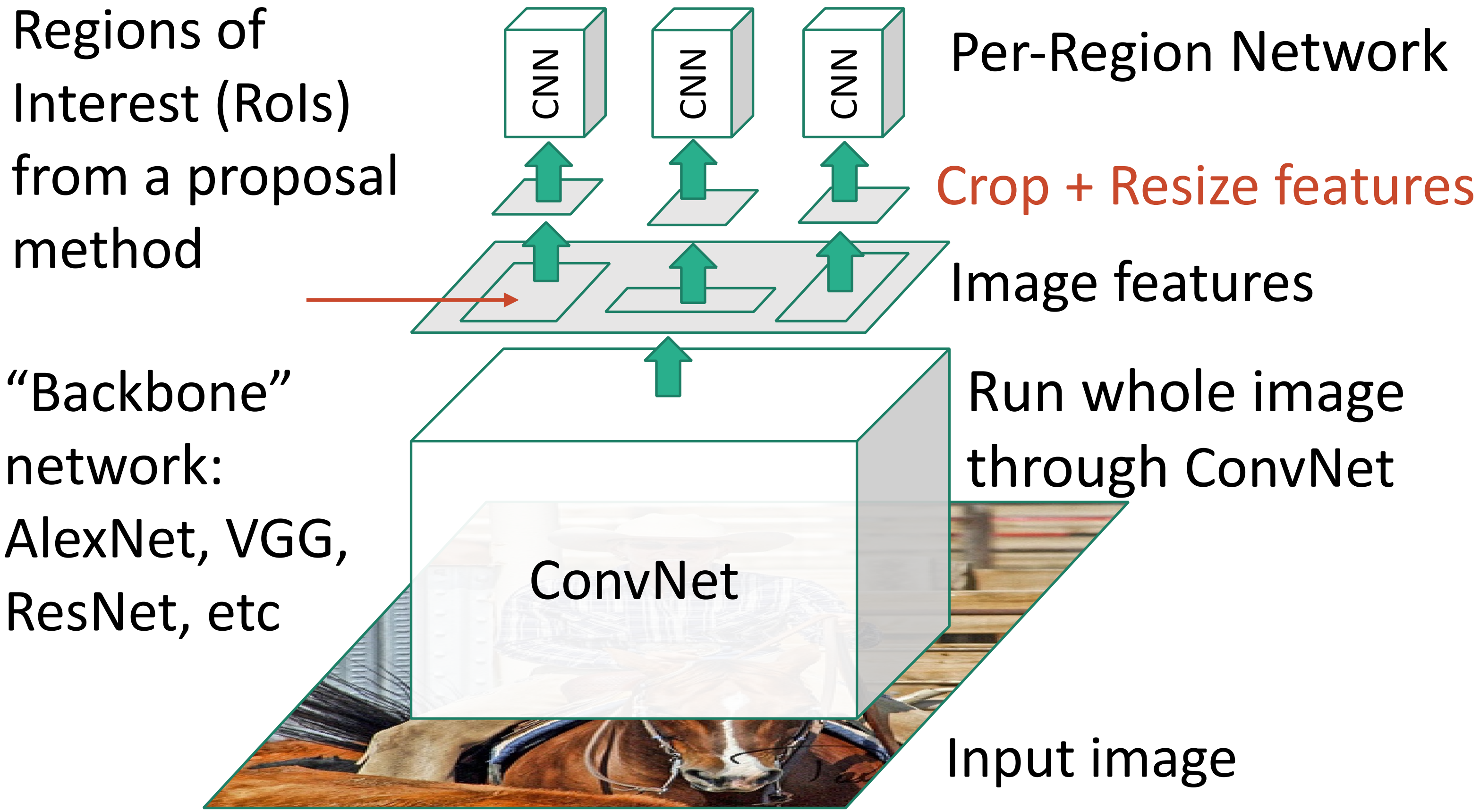
Fast R-CNN

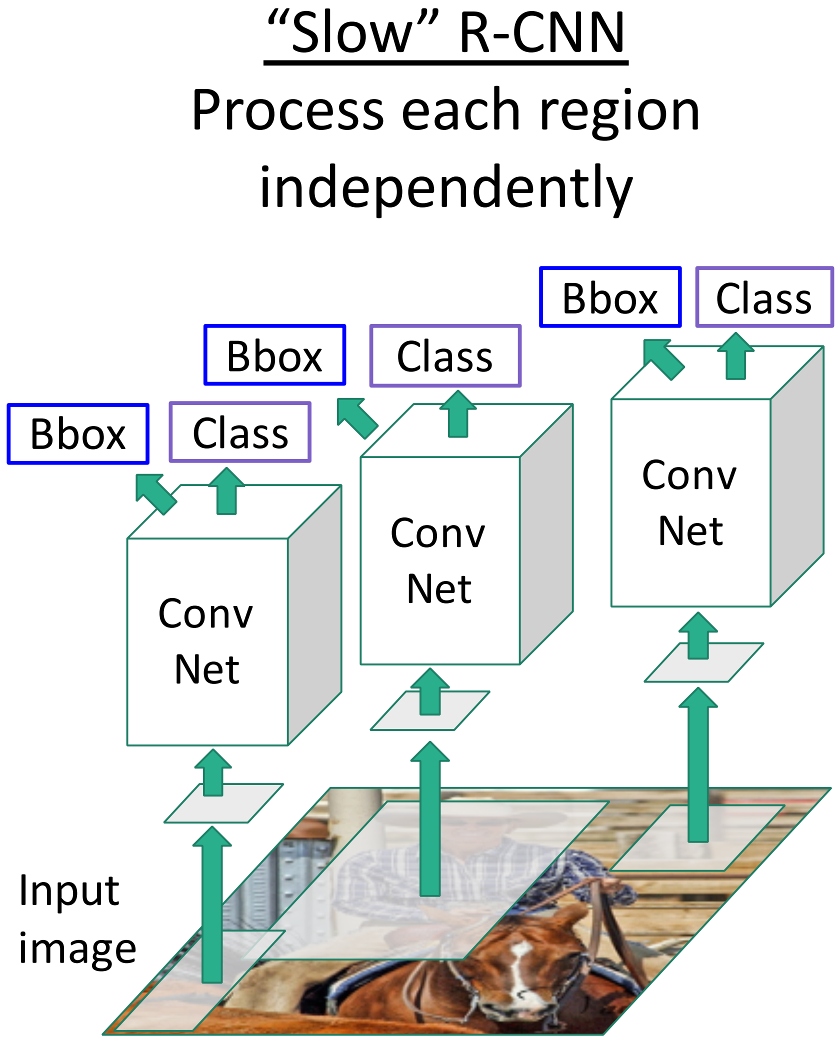




Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

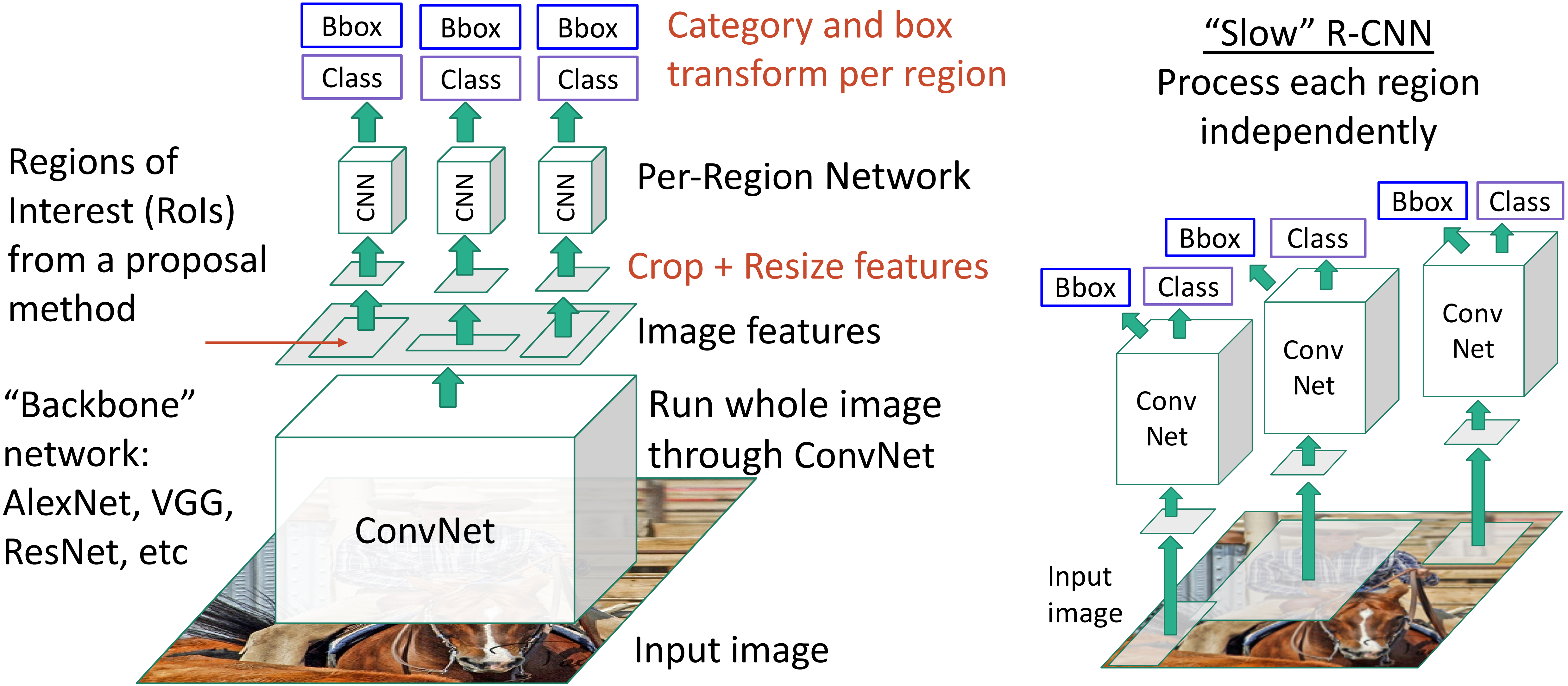
Fast R-CNN





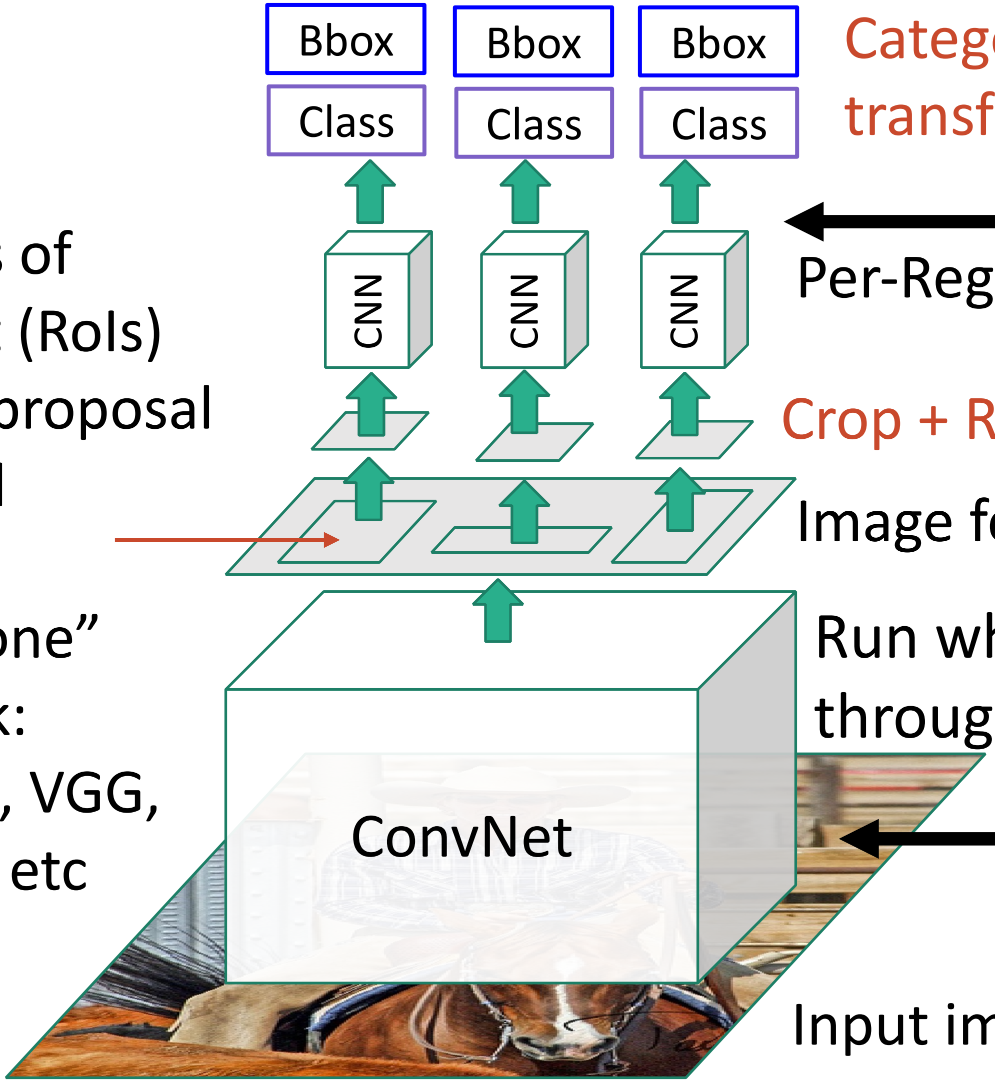
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



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

Fast R-CNN



Category and box transform per region

Per-Region network is relatively lightweight

Regions of Interest (RoIs) from a proposal method

Per-Region Network

Crop + Resize features

Image features

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

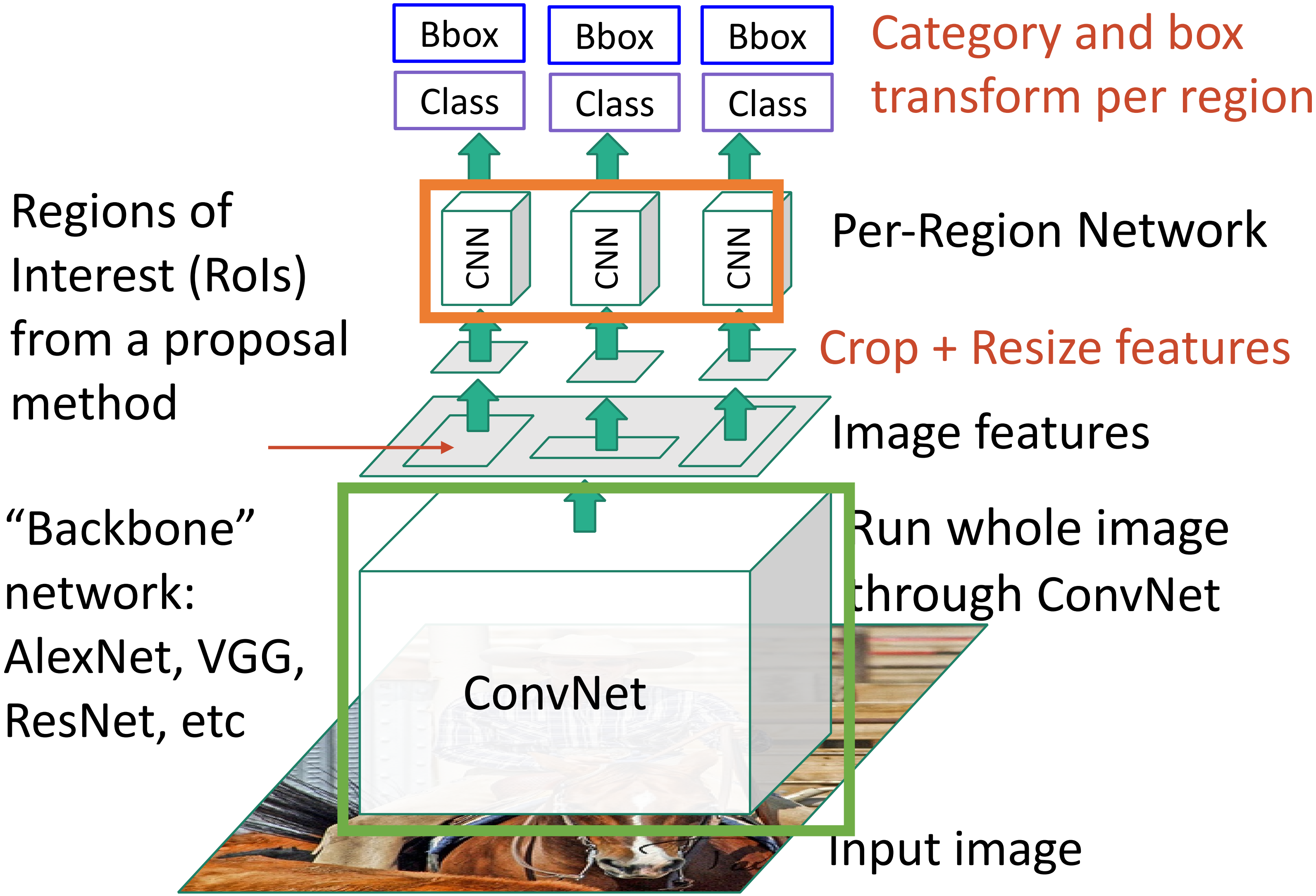
Run whole image through ConvNet

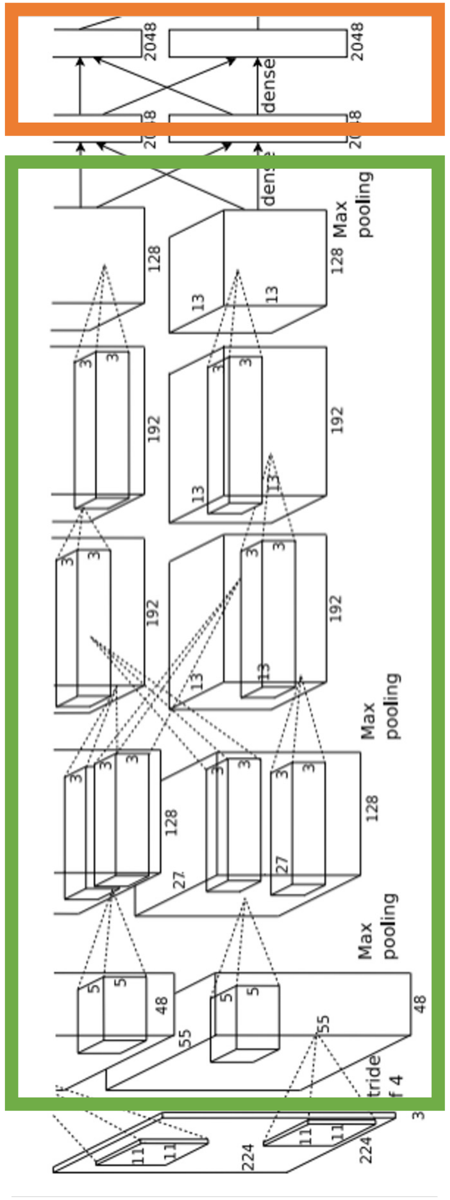
Most of the computation happens in backbone network; this saves work for overlapping region proposals

Input image

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

Fast R-CNN

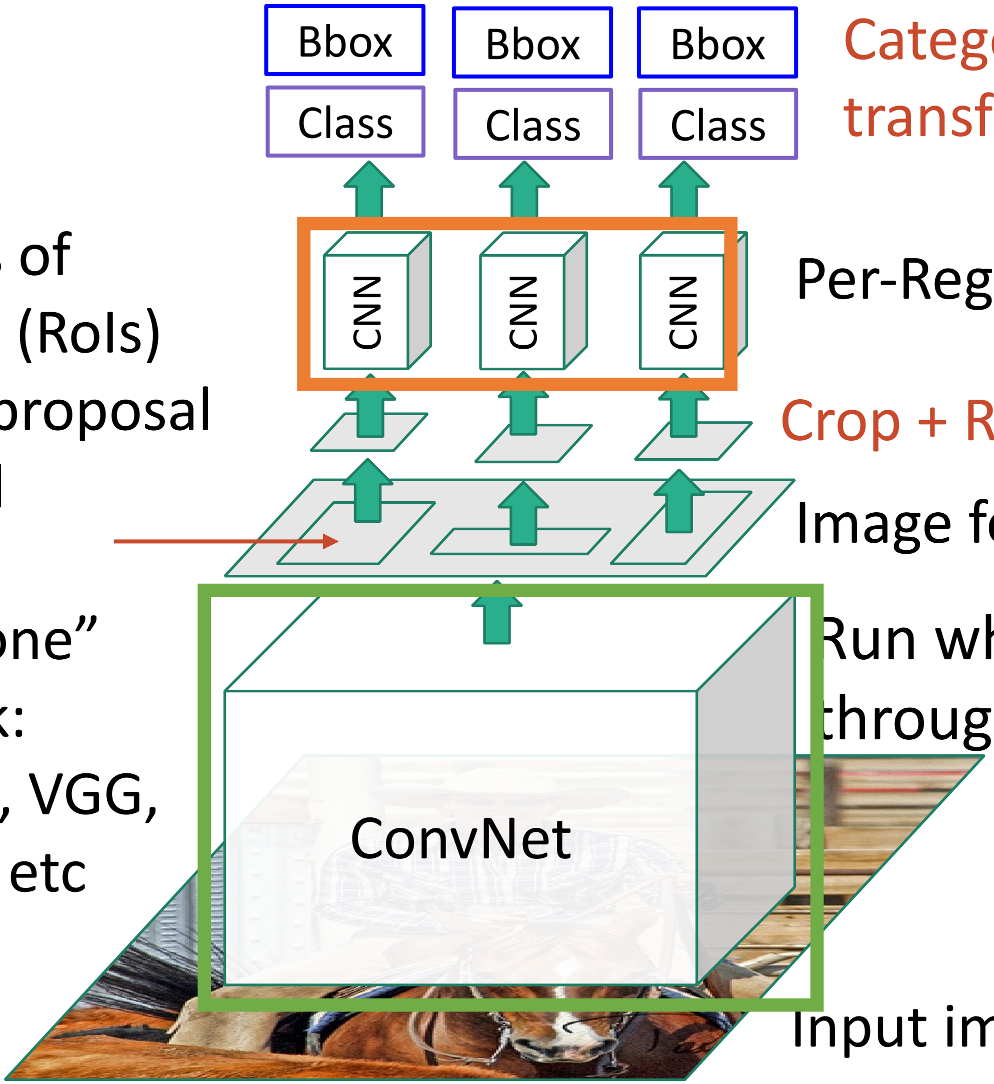




Example: When using AlexNet for detection, five conv layers are used for backbone and two FC layers are used for per-region network

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

Fast R-CNN



Category and box transform per region

Per-Region Network

Crop + Resize features

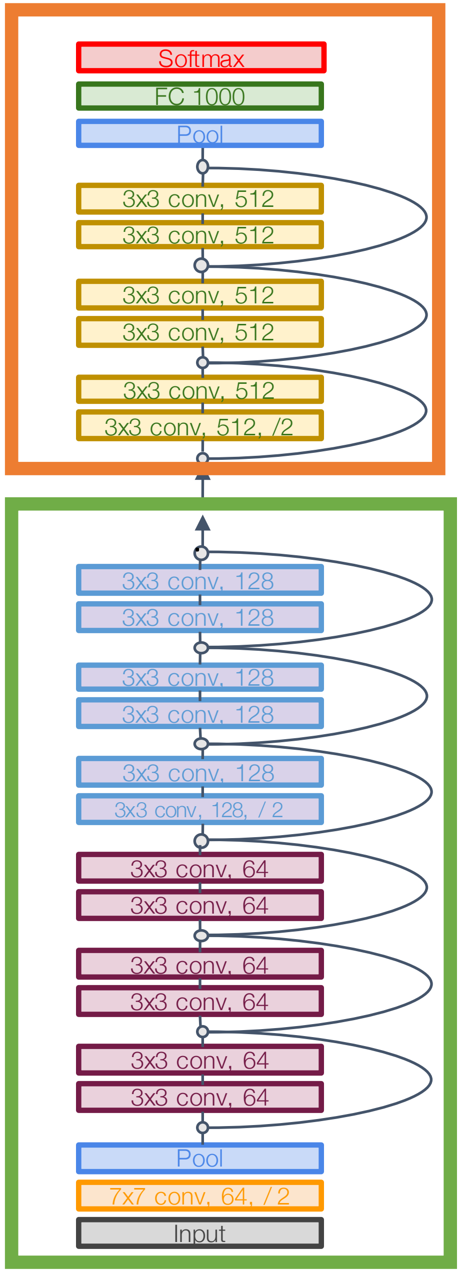
Image features

Run whole image through ConvNet

Input image

Regions of Interest (RoIs) from a proposal method

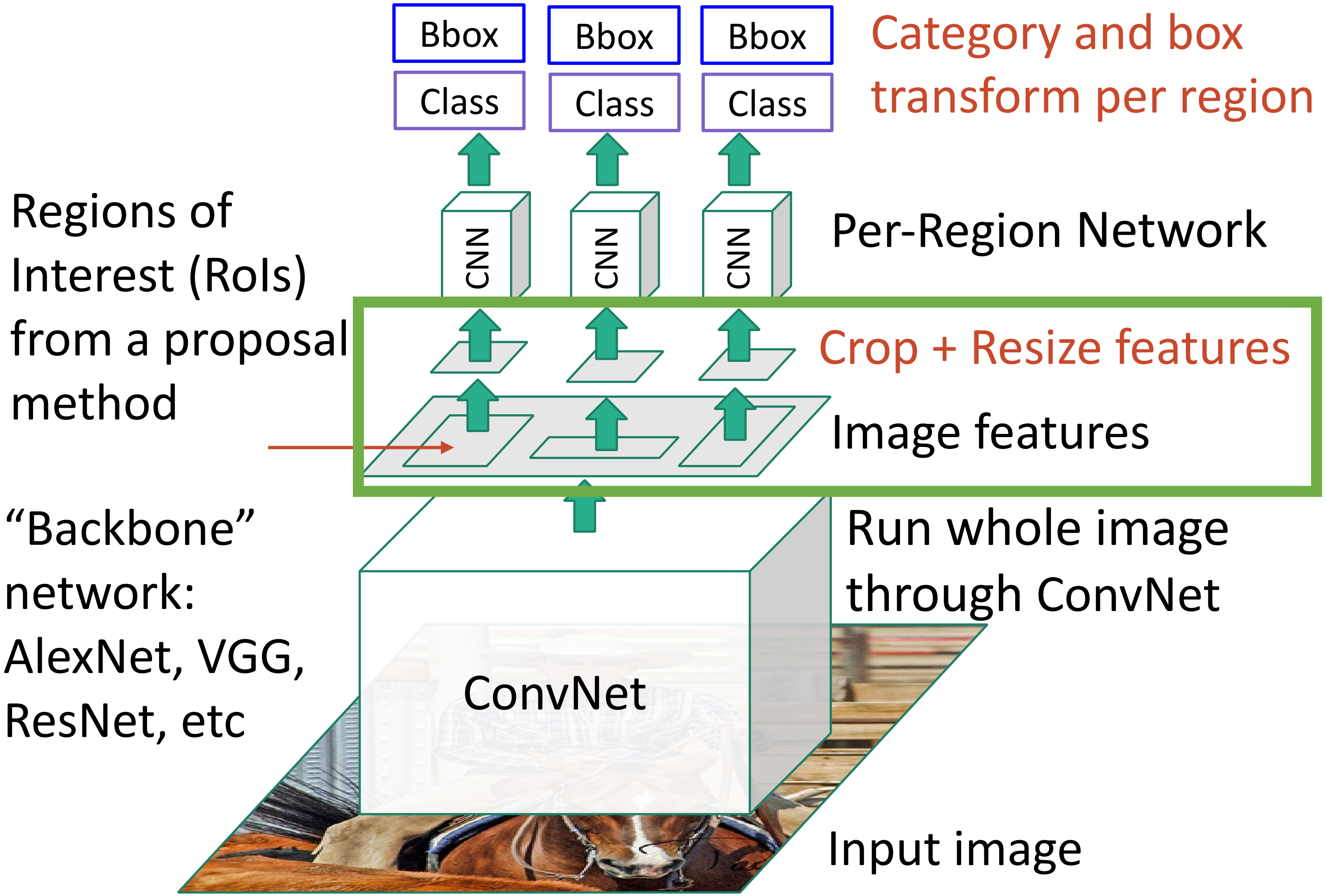
"Backbone" network: AlexNet, VGG, ResNet, etc



Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Slide from Justin Johnson

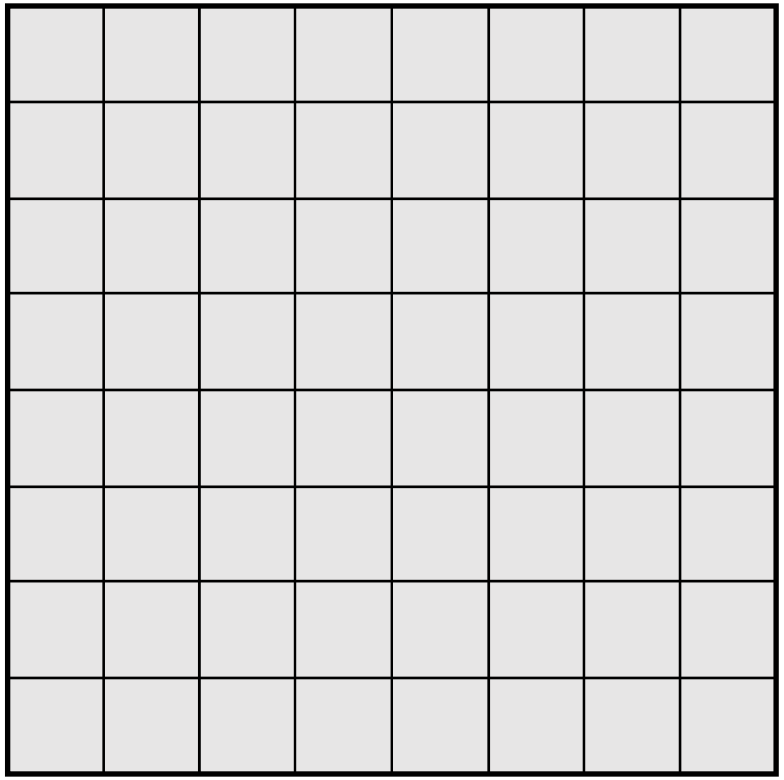
Fast R-CNN



How to crop features?

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

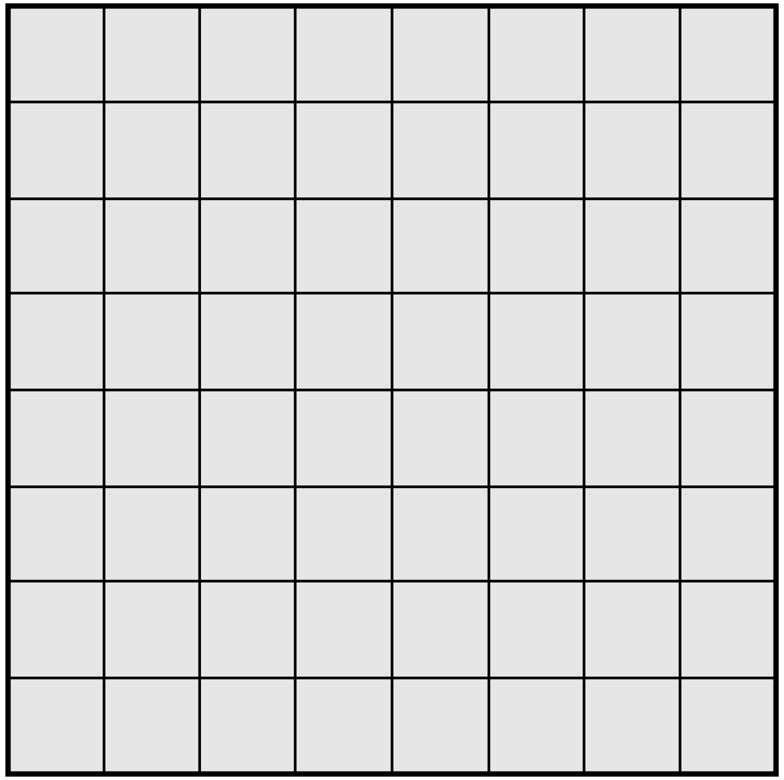
Recall: Receptive Fields



Input Image: 8 x 8

Every position in the output feature map depends on a 3x3 receptive field in the input

3x3 Conv
Stride 1, pad 1

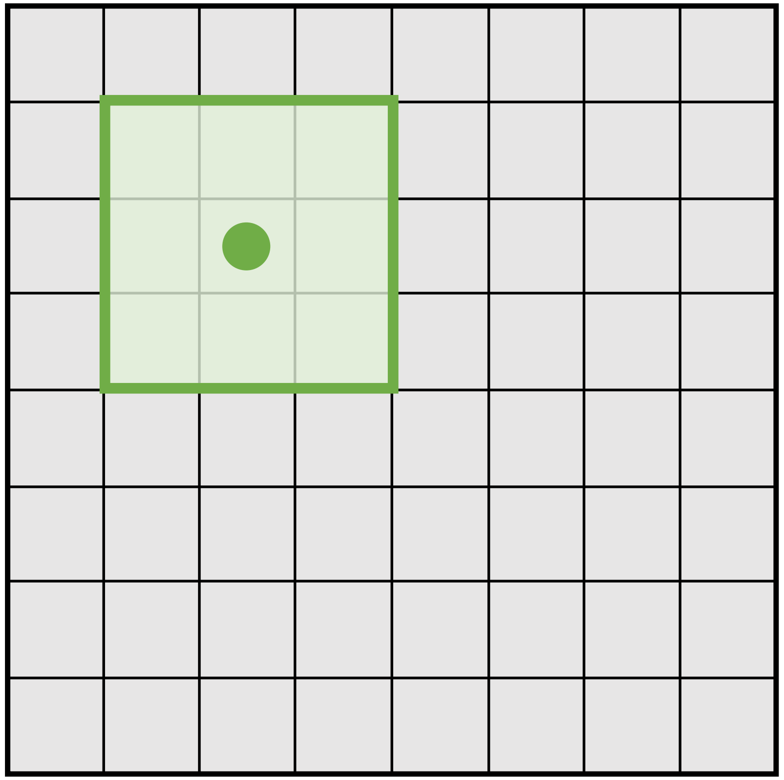


Output Image: 8 x 8

Slide from Justin Johnson

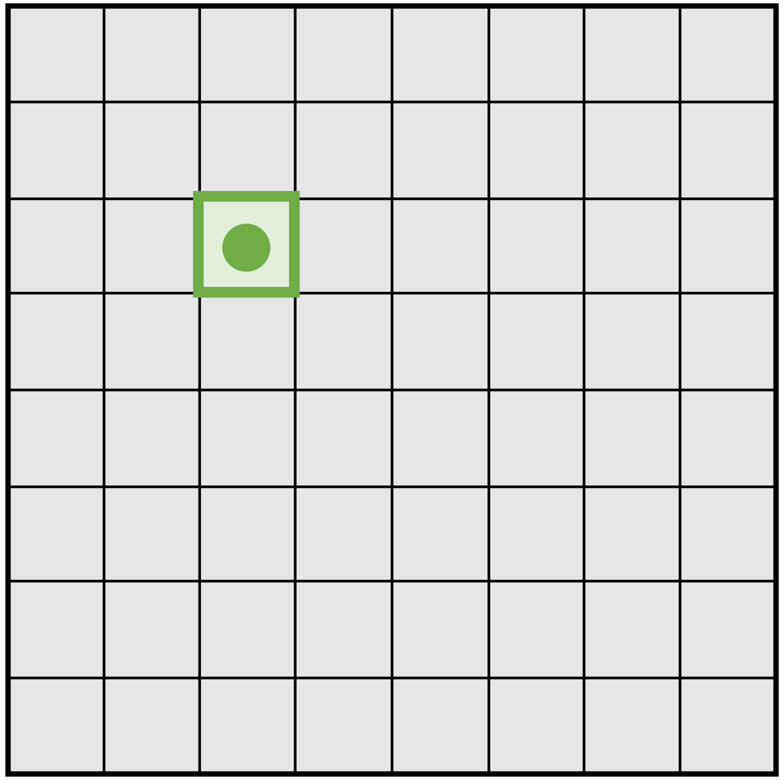
Recall: Receptive Fields

Every position in the output feature map depends on a 3x3 receptive field in the input



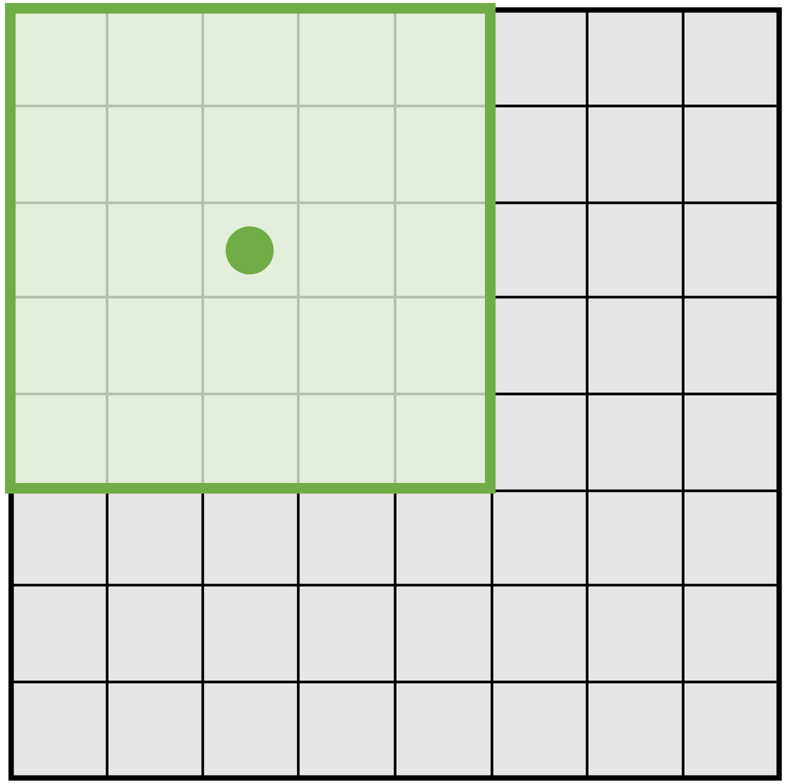
Input Image: 8 x 8

3x3 Conv
Stride 1, pad 1

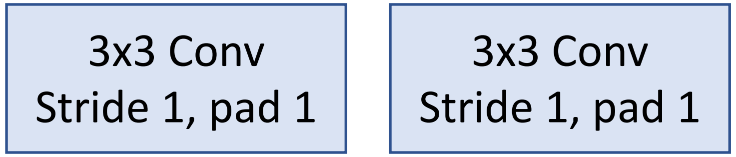


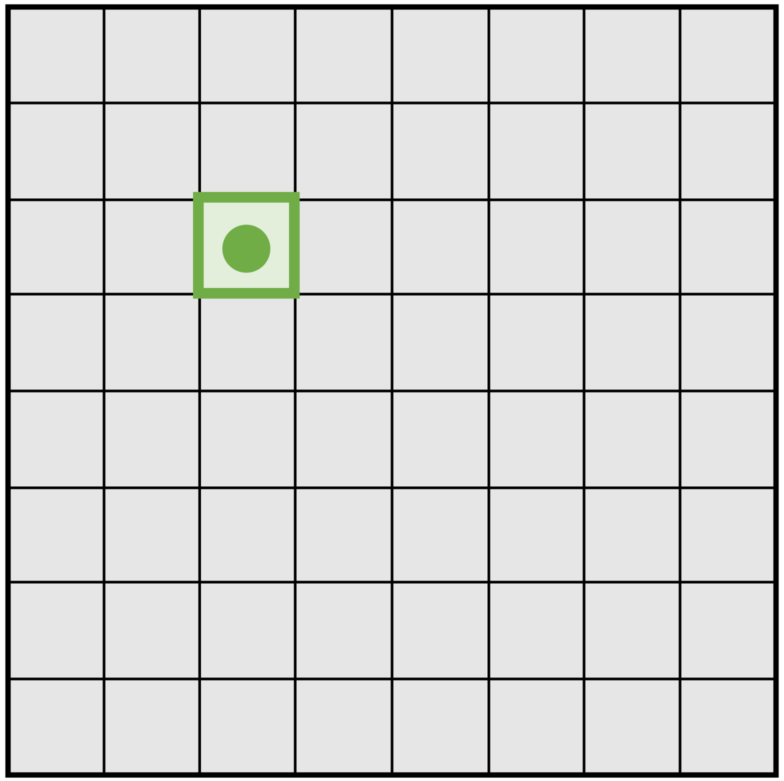
Output Image: 8 x 8

Recall: Receptive Fields



Every position in the output feature map depends on a <u>5x5</u> receptive field in the input

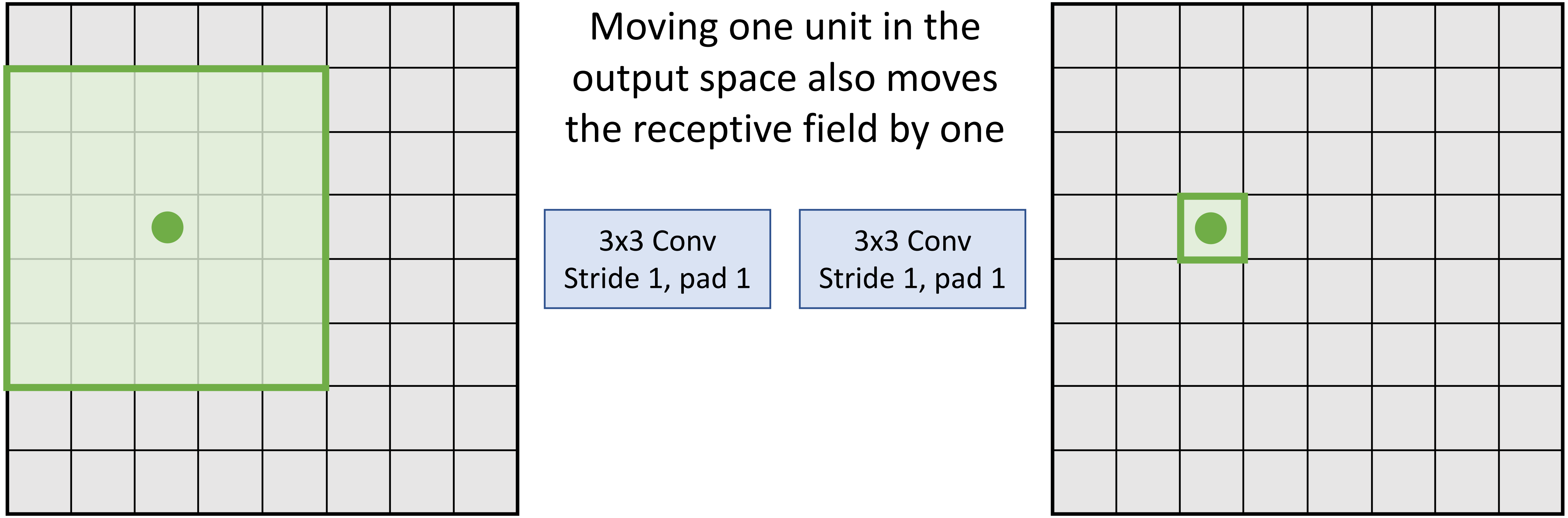




Input Image: 8 x 8

Output Image: 8 x 8

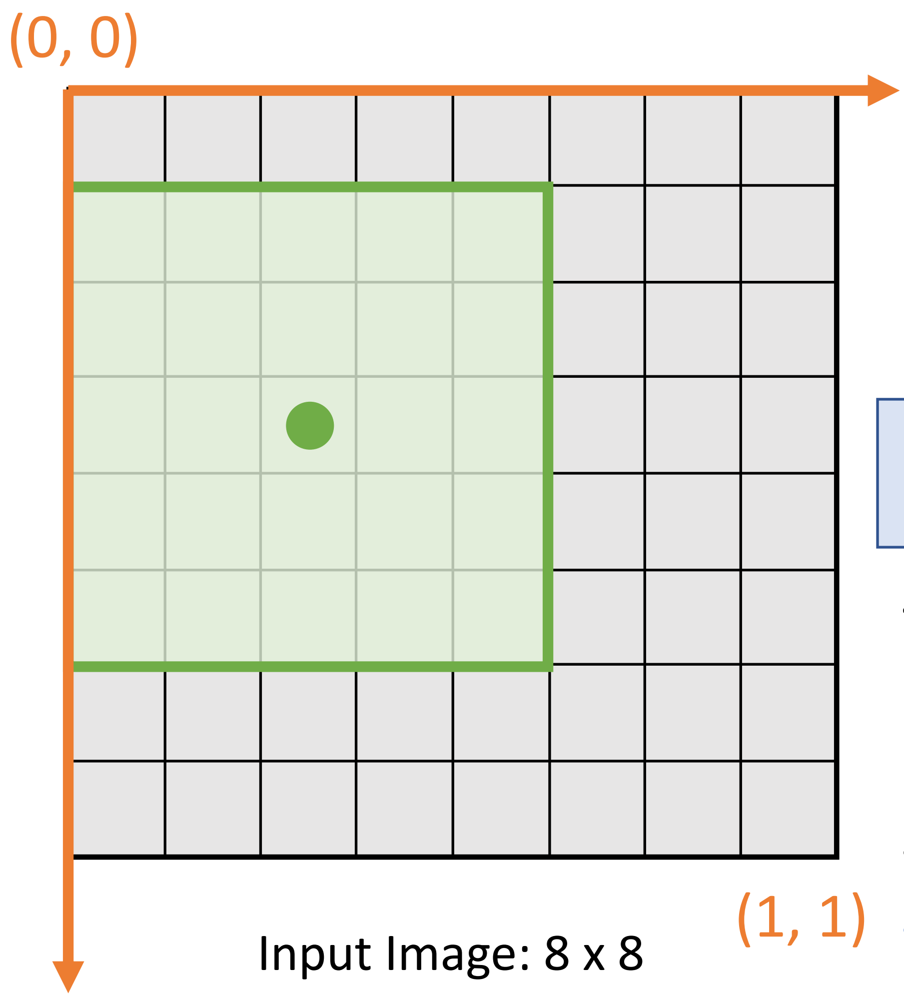
Recall: Receptive Fields



Input Image: 8 x 8

Output Image: 8 x 8

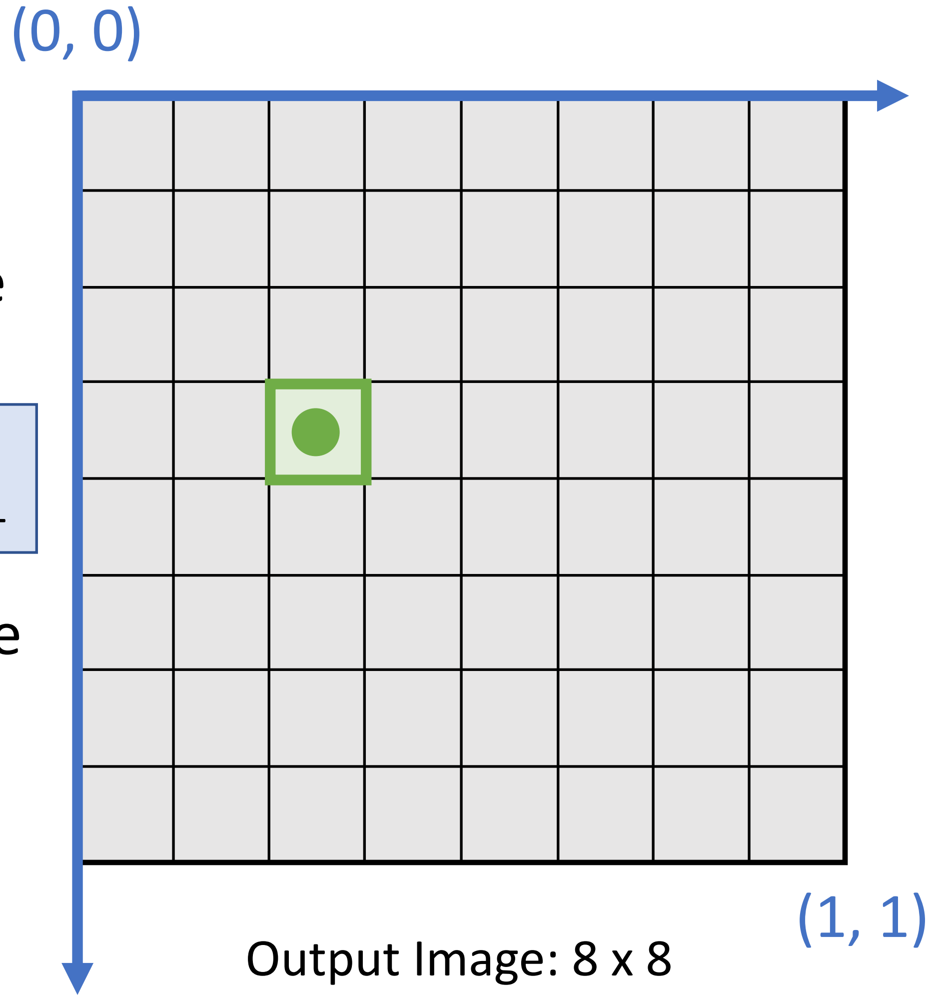
Recall: Receptive Fields



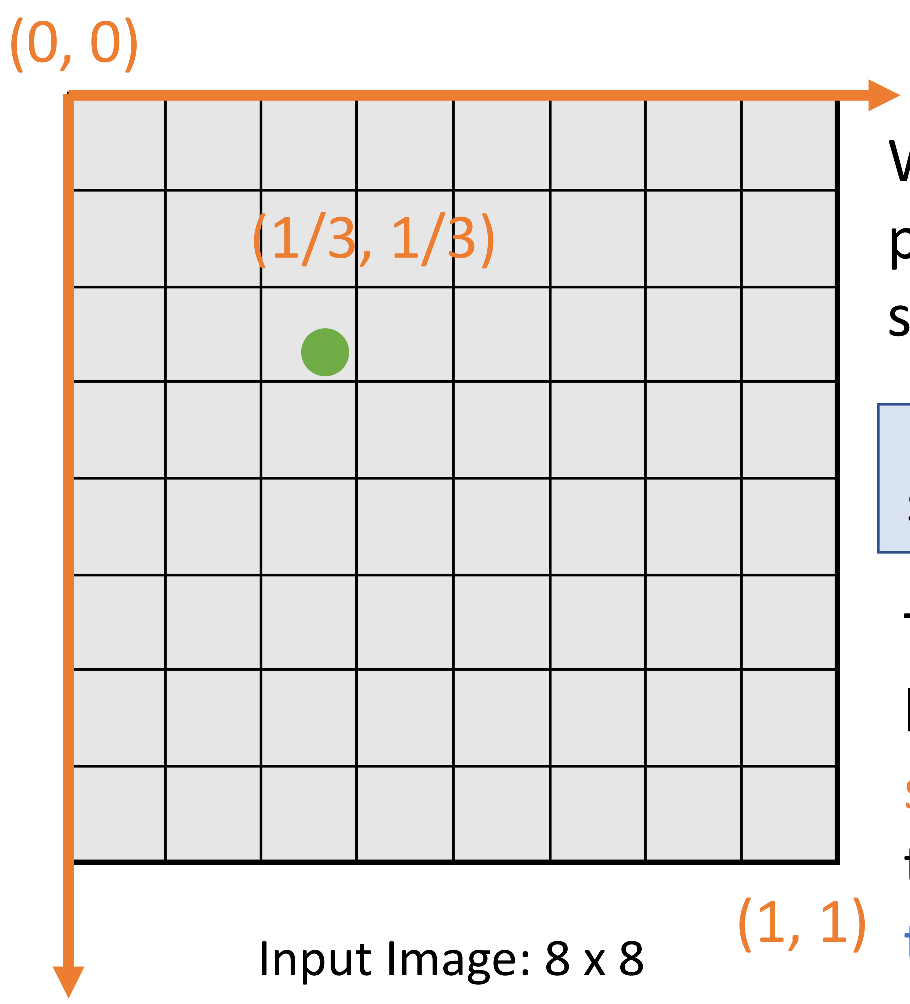
Moving one unit in the output space also moves the receptive field by one

3x3 Conv Stride 1, pad 1	3x3 Conv Stride 1, pad 1

There is a correspondence between the coordinate system of the input and the coordinate system of the output



Projecting Points

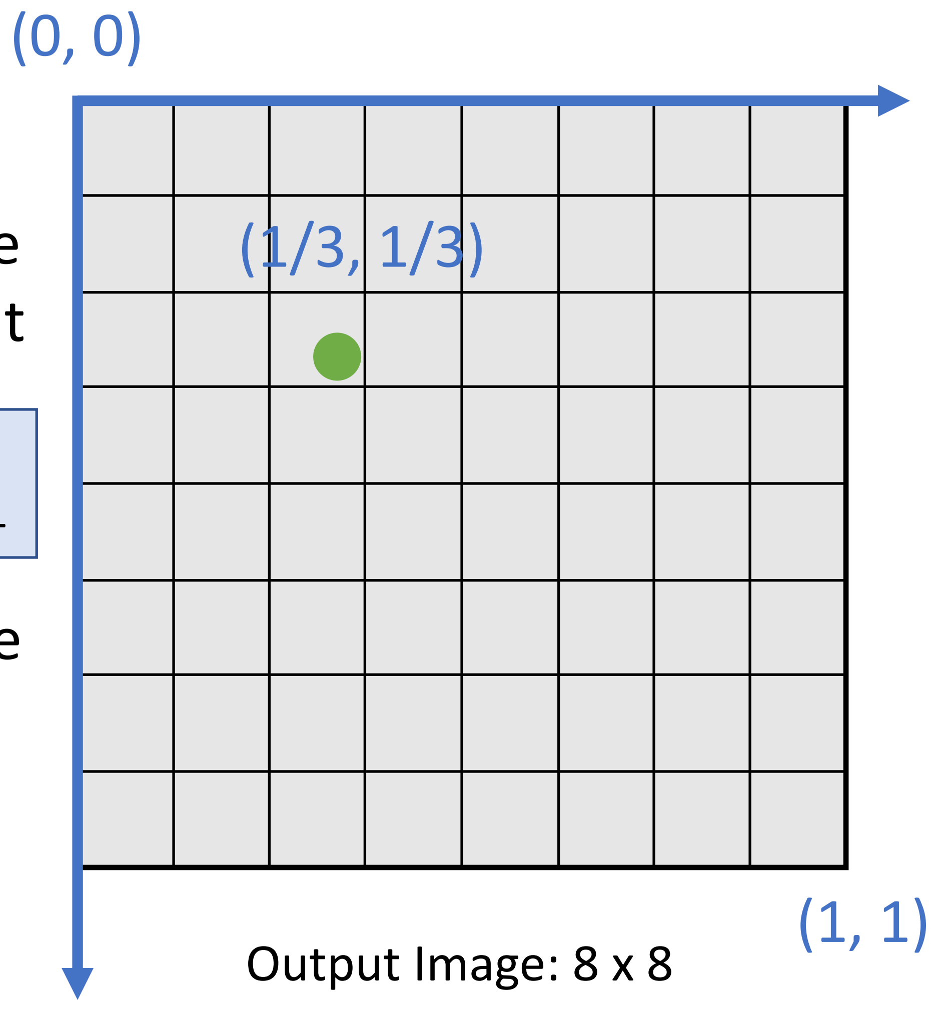


We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

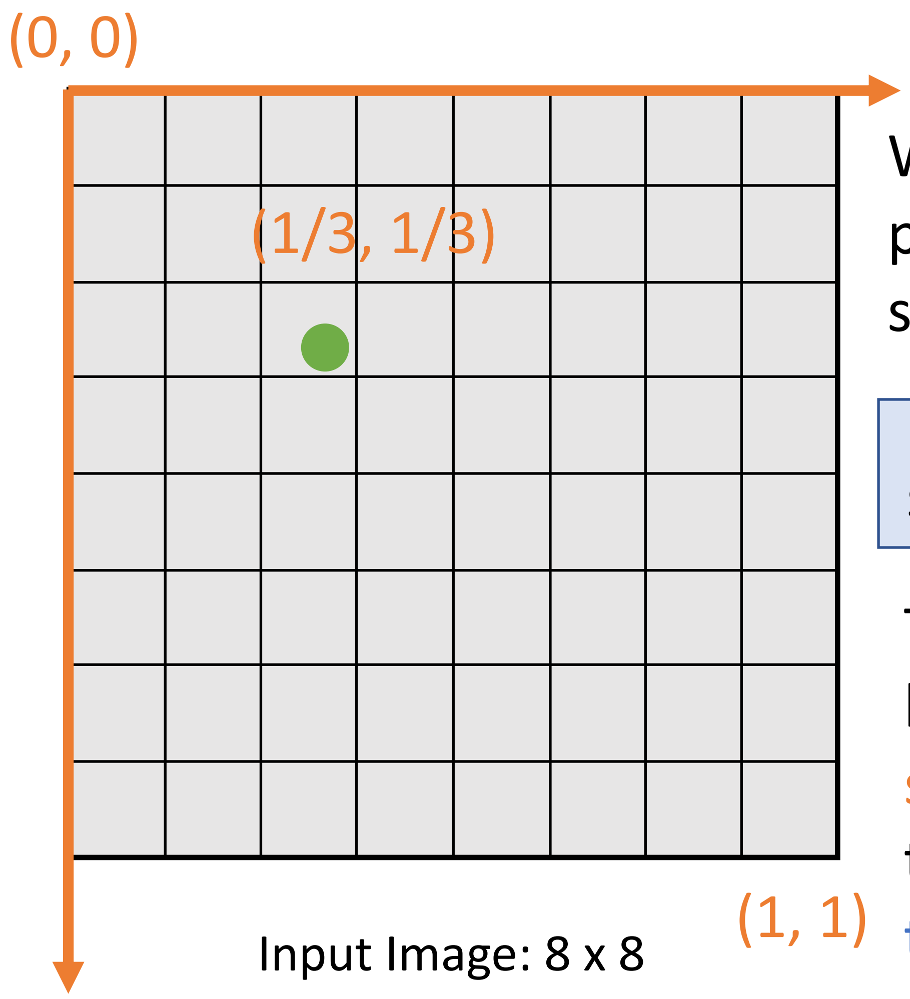
3x3 Conv
Stride 1, pad 1

There is a correspondence between the coordinate system of the input and the coordinate system of the output



Projecting Points

Same logic holds for more complicated CNNs, even if spatial resolution of input and output are different



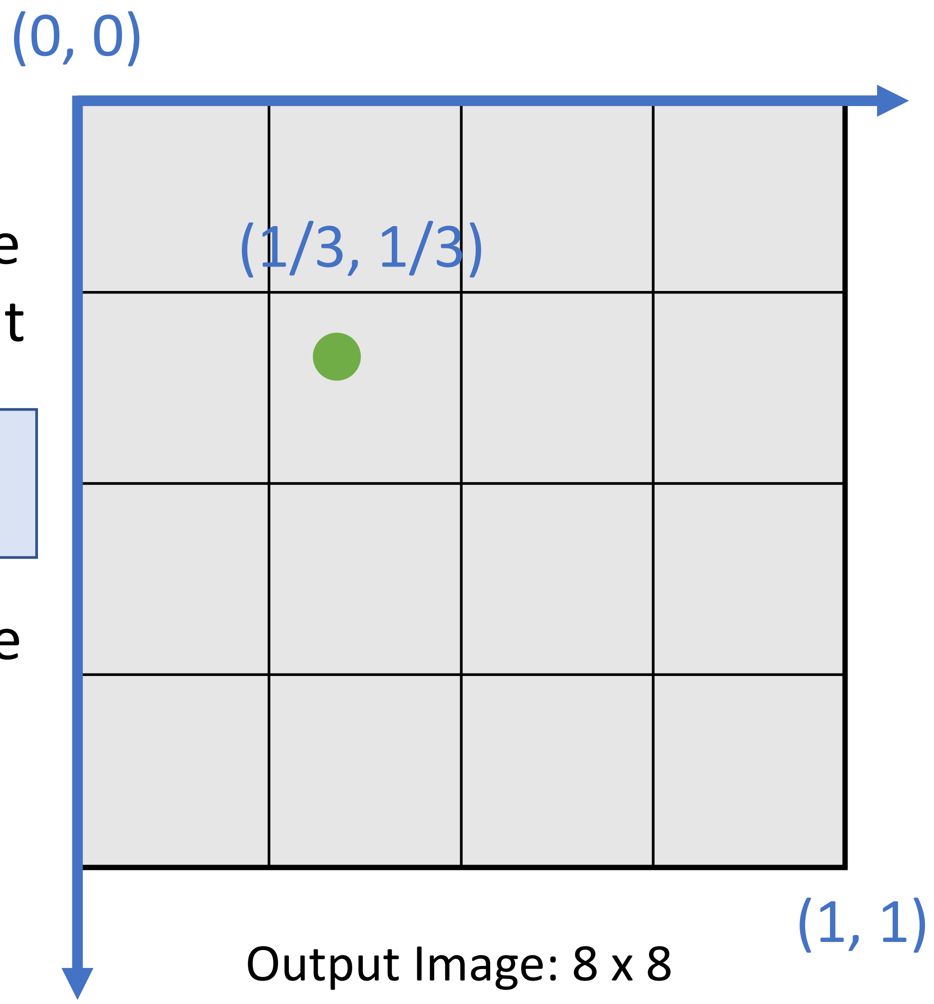
Input Image: 8 x 8

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

2x2 MaxPool
Stride 2

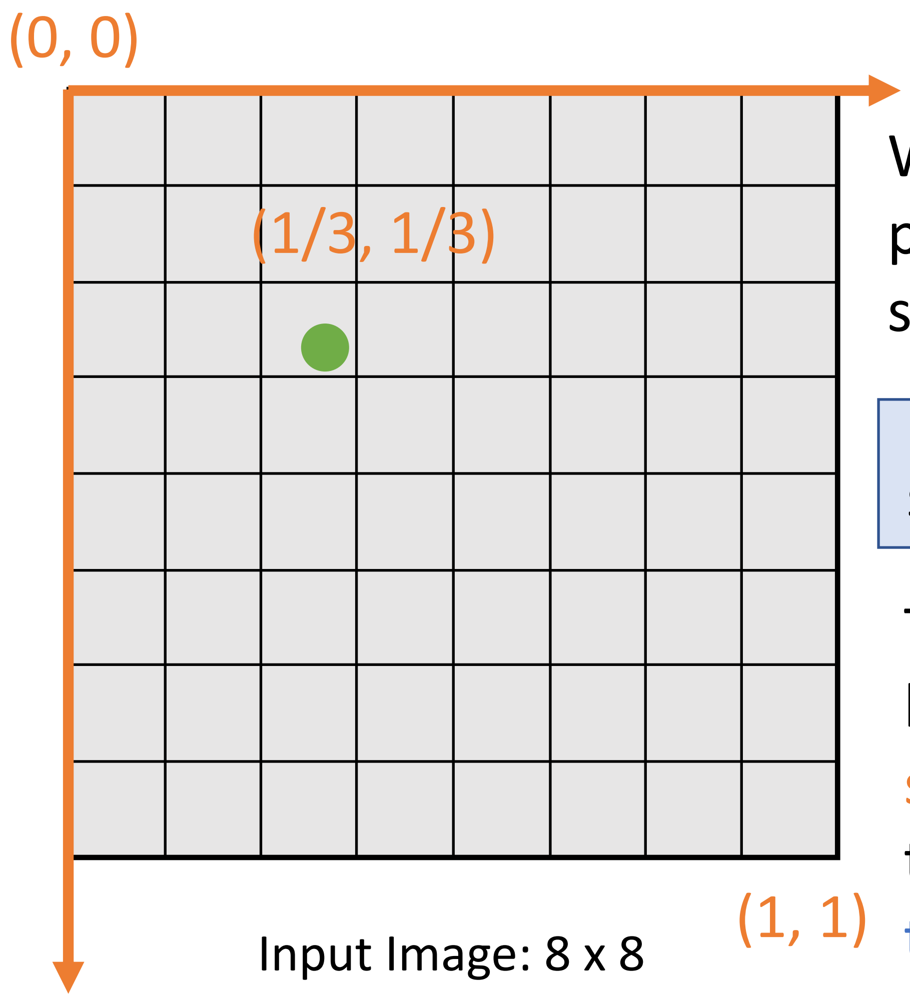
There is a correspondence between the coordinate system of the input and the coordinate system of the output



Output Image: 8 x 8

Projecting Points

Same logic holds for more complicated CNNs, even if spatial resolution of input and output are different



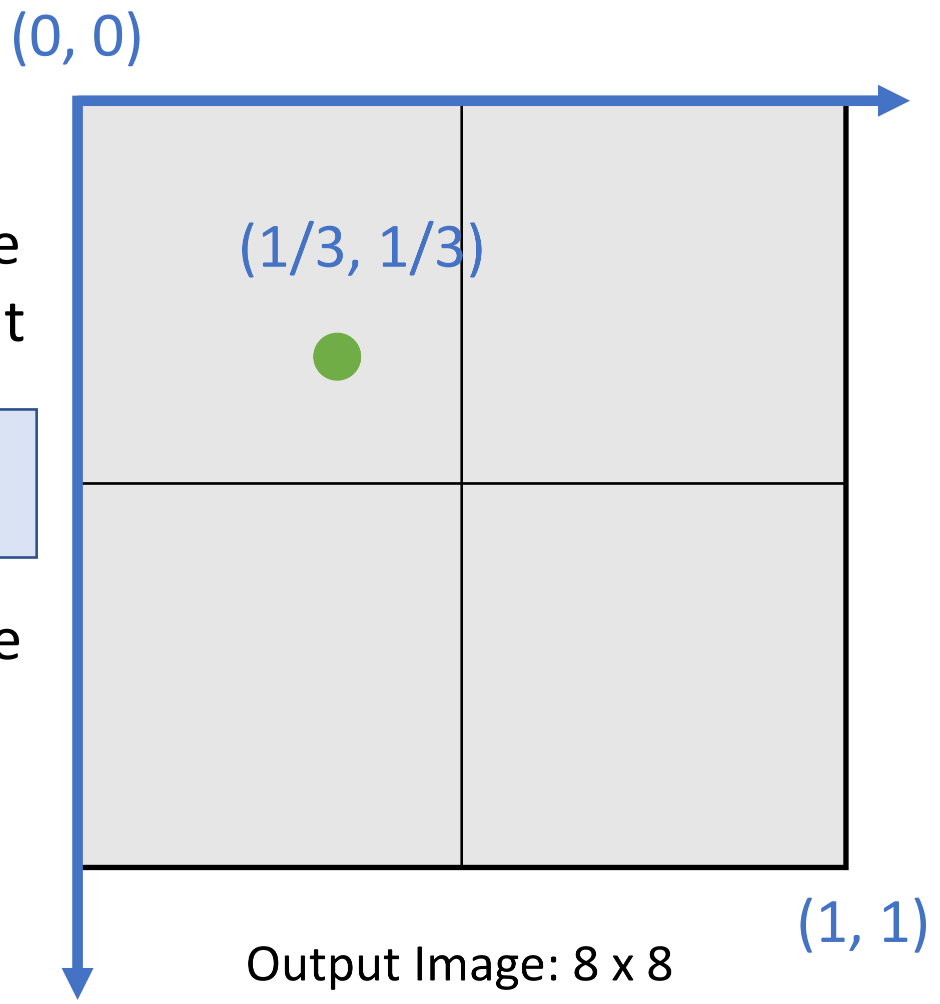
Input Image: 8 x 8

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

4x4 MaxPool
Stride 4

There is a correspondence between the coordinate system of the input and the coordinate system of the output

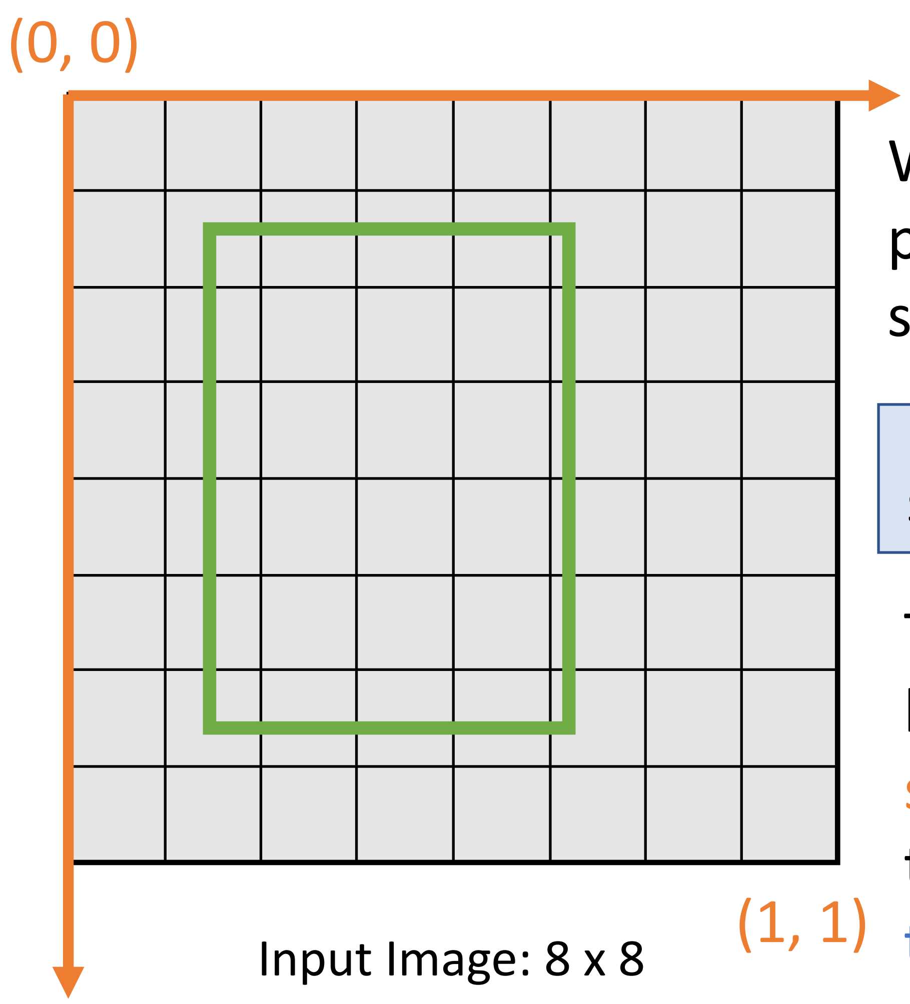


Output Image: 8 x 8

Slide from Justin Johnson

Projecting Boxes

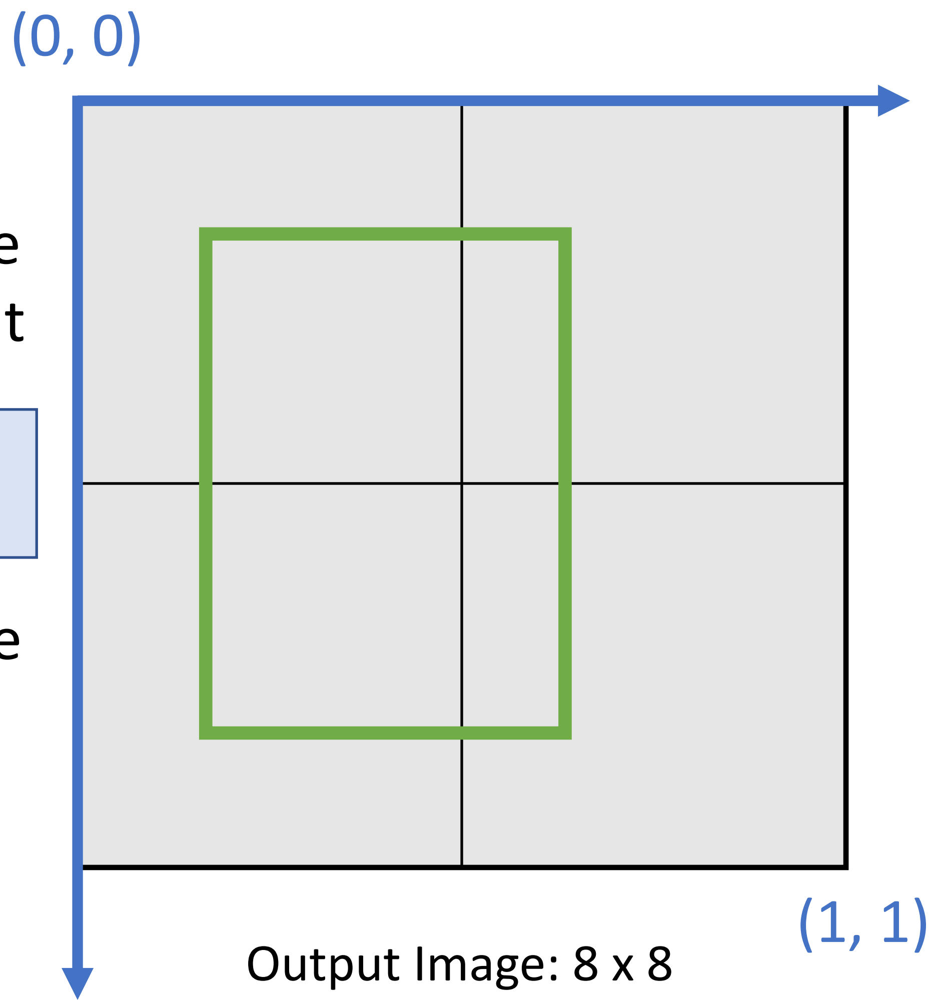
We can use this idea to project **bounding boxes** between an input image and a feature map



We can align arbitrary points between coordinate system of input and output

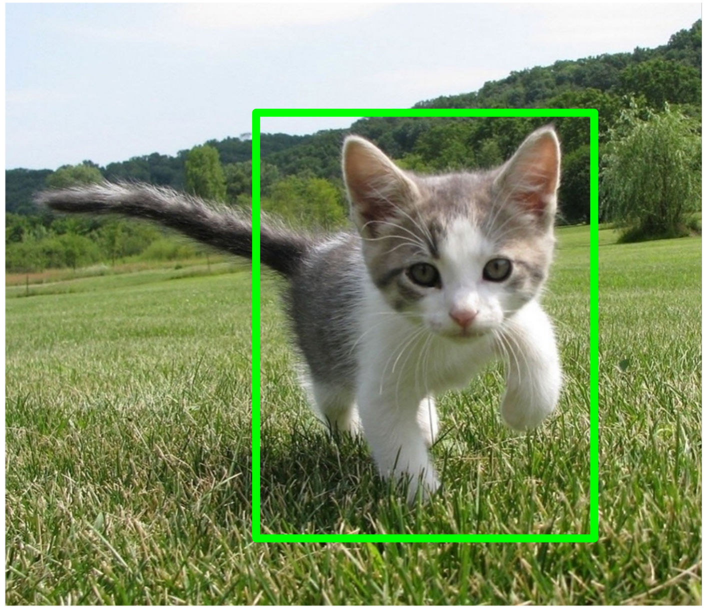
3x3 Conv Stride 1, pad 1	4x4 MaxPool Stride 4

There is a correspondence between the coordinate system of the input and the coordinate system of the output



Slide from Justin Johnson

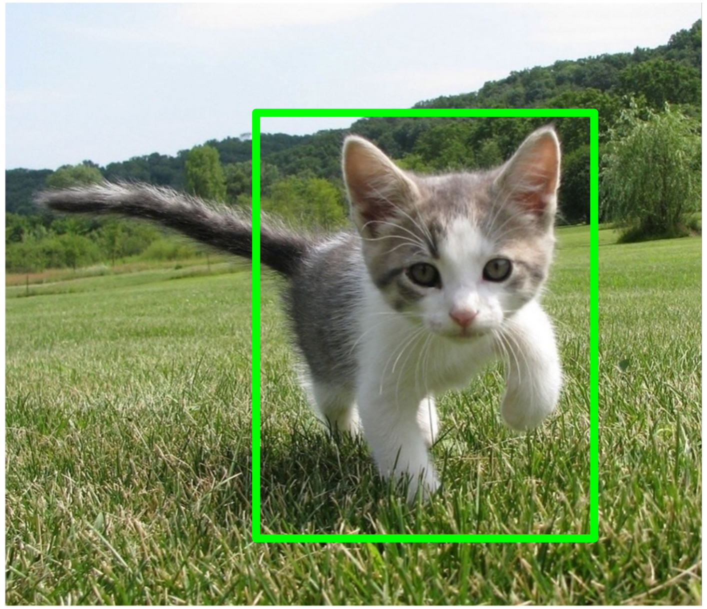
Cropping Features: RoI Pool



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

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



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

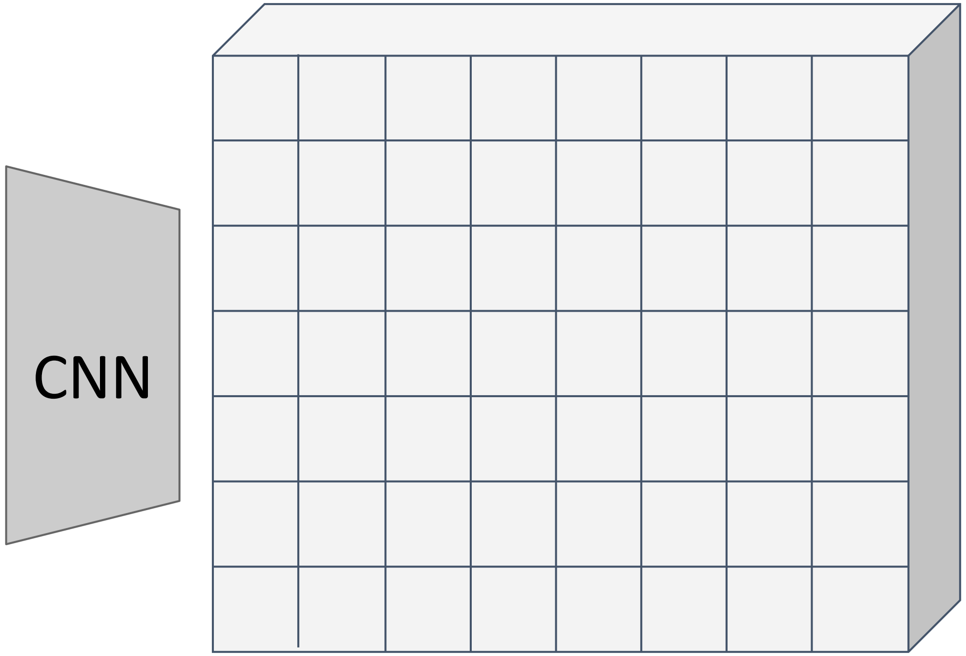
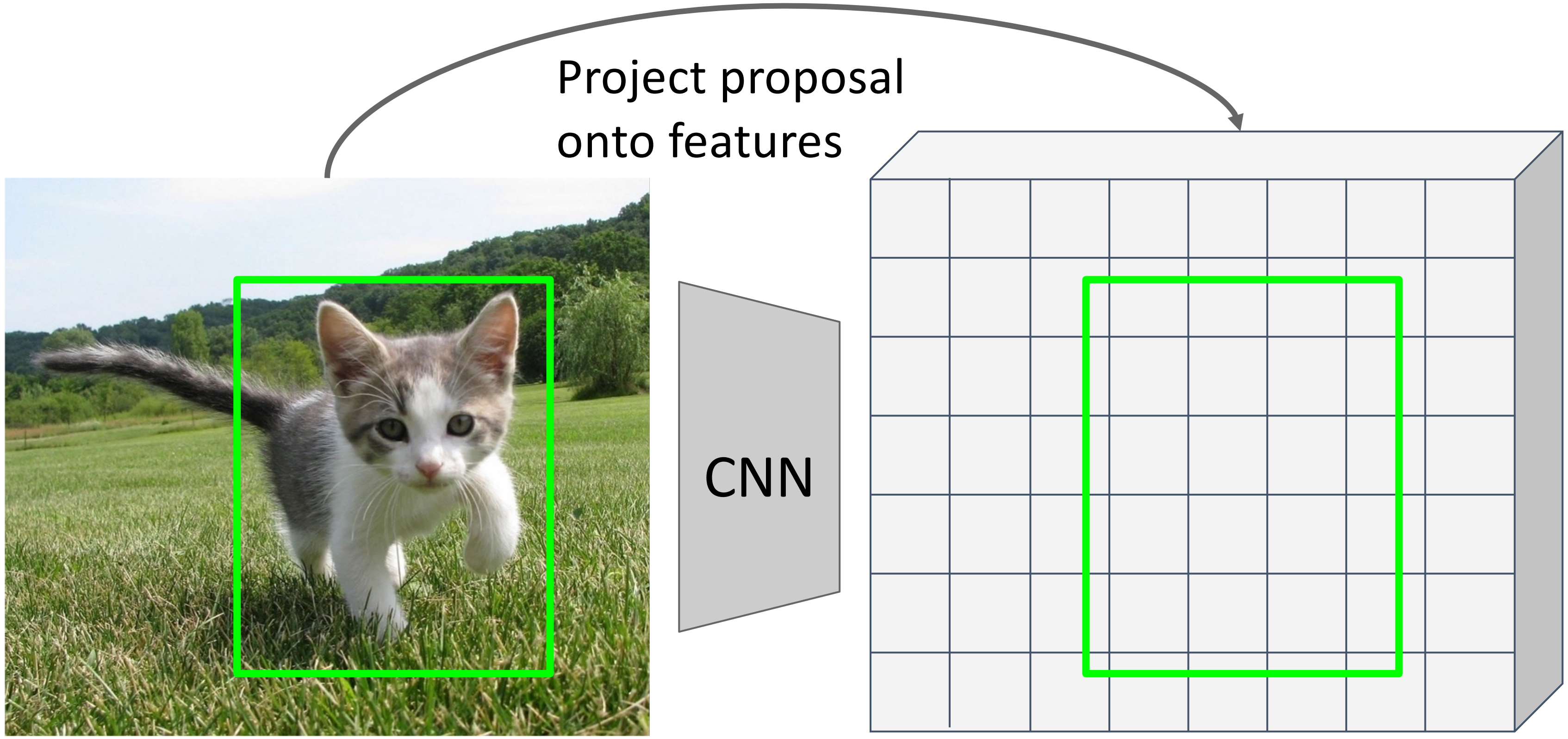


Image features
(e.g. 512 x 20 x 15)

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool



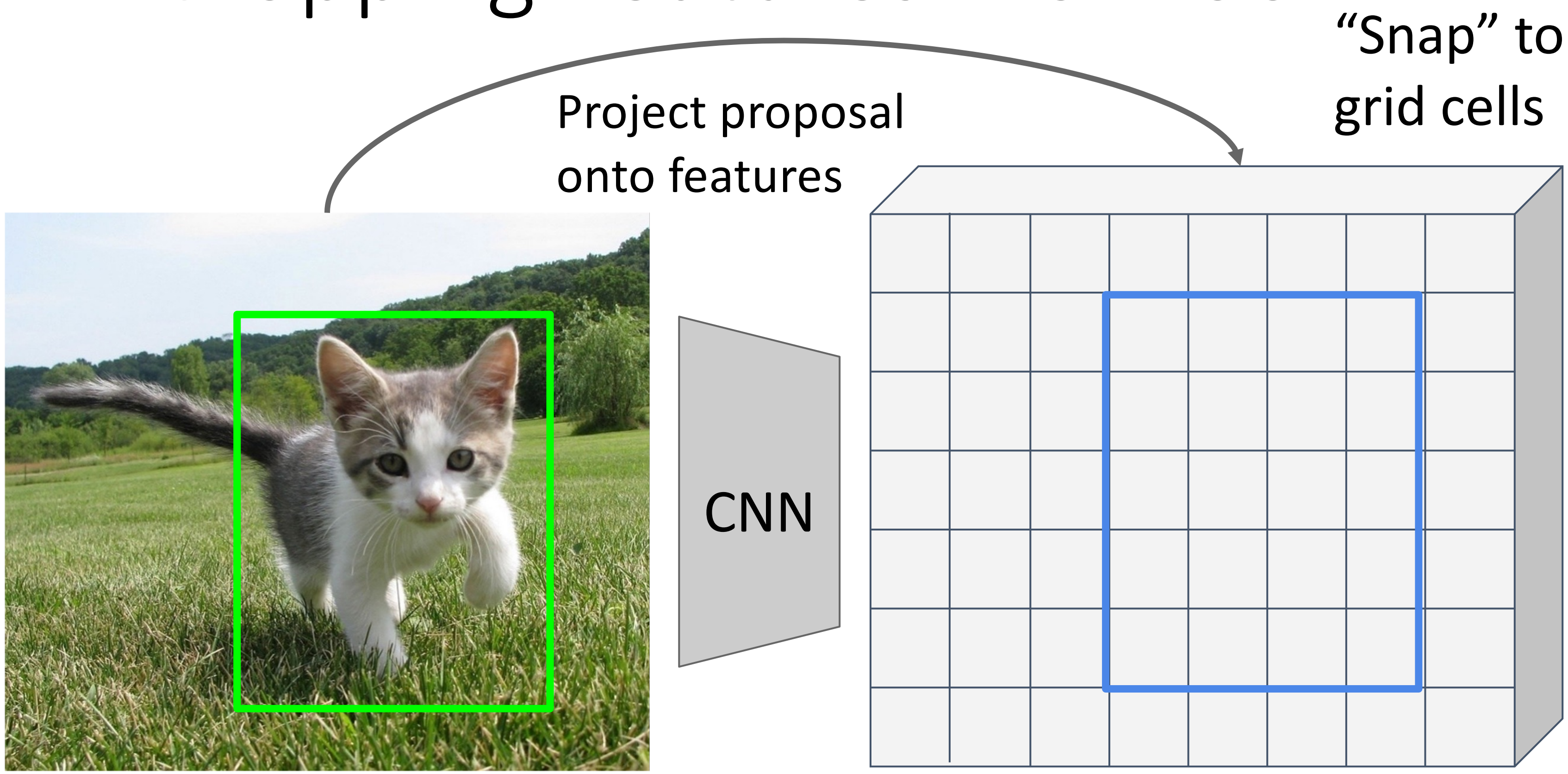
Input Image
(e.g. 3 x 640 x 480)

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

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool



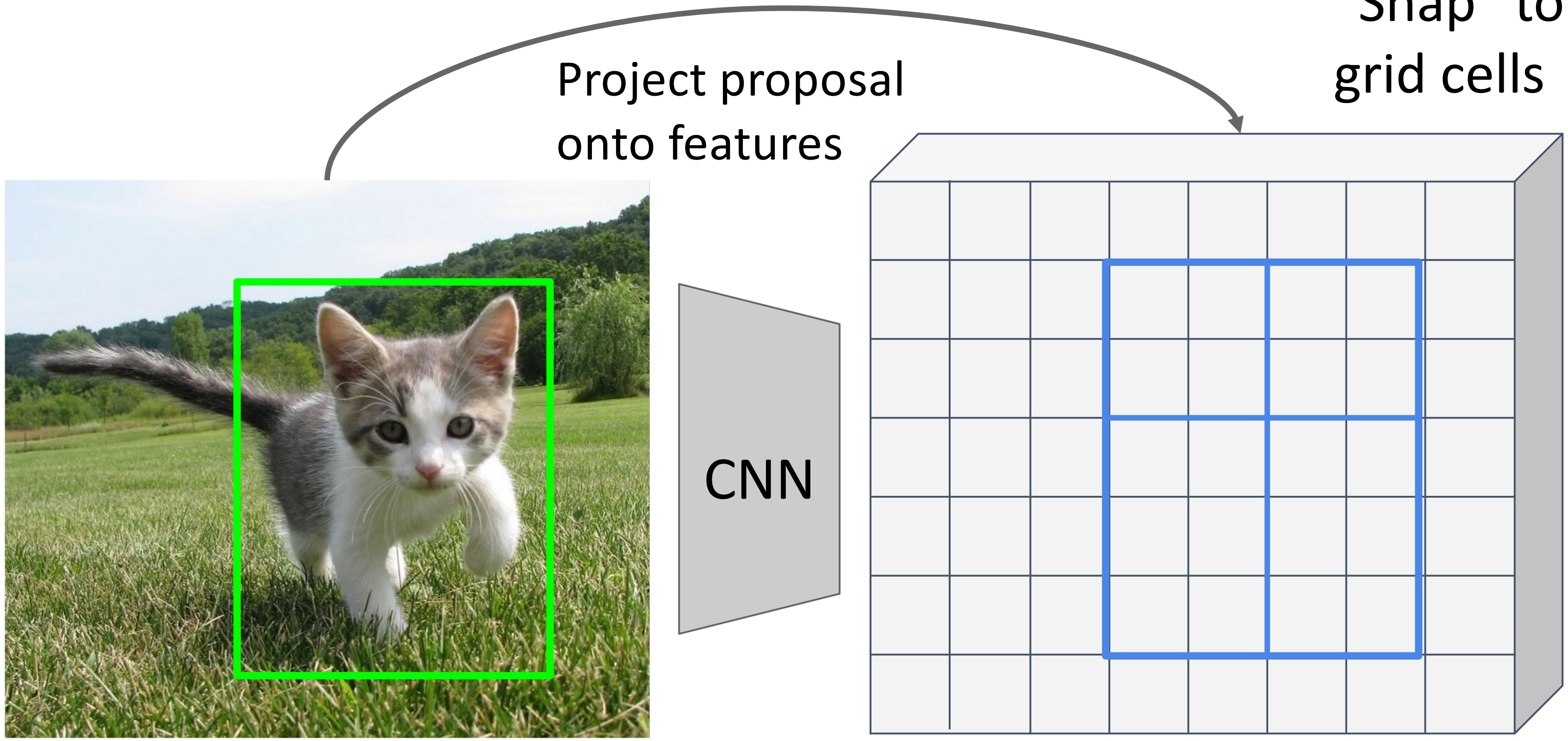
Input Image
(e.g. 3 x 640 x 480)

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

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool



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

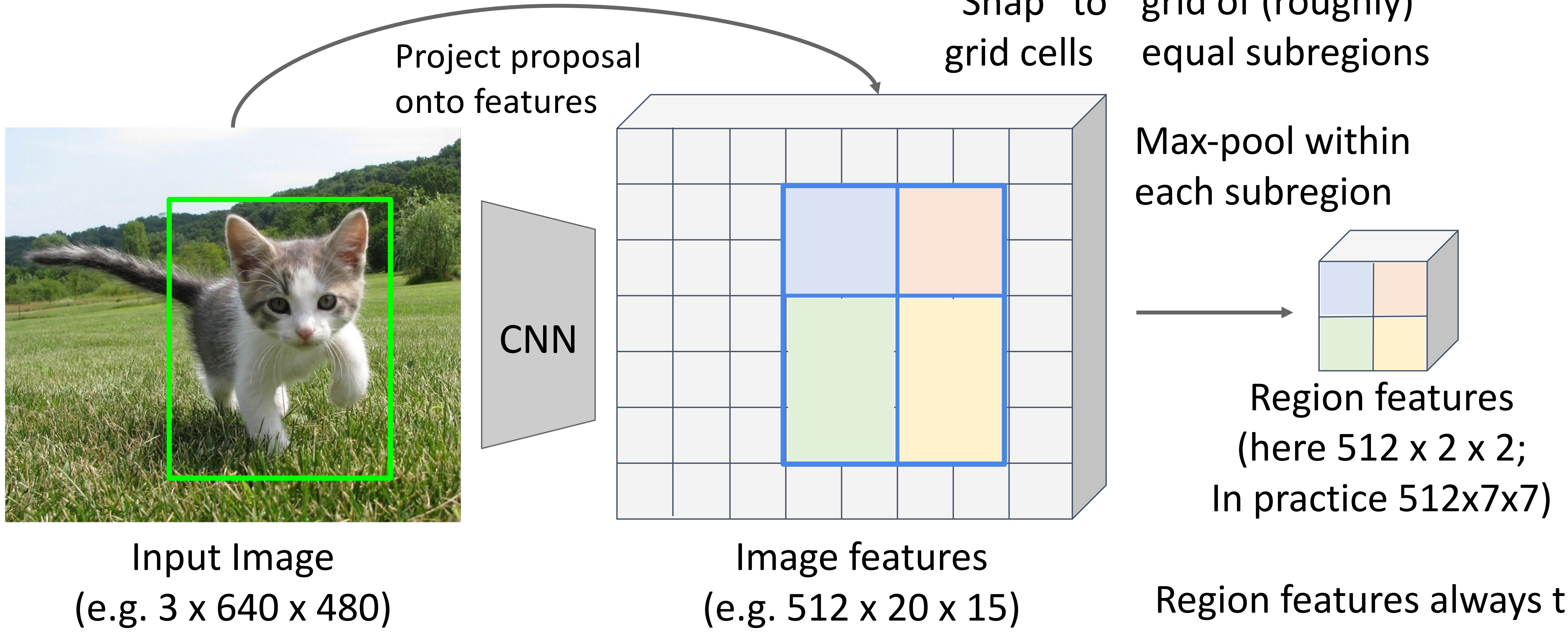
Image features
(e.g. 512 x 20 x 15)

Divide into 2x2 grid of (roughly) equal subregions

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool



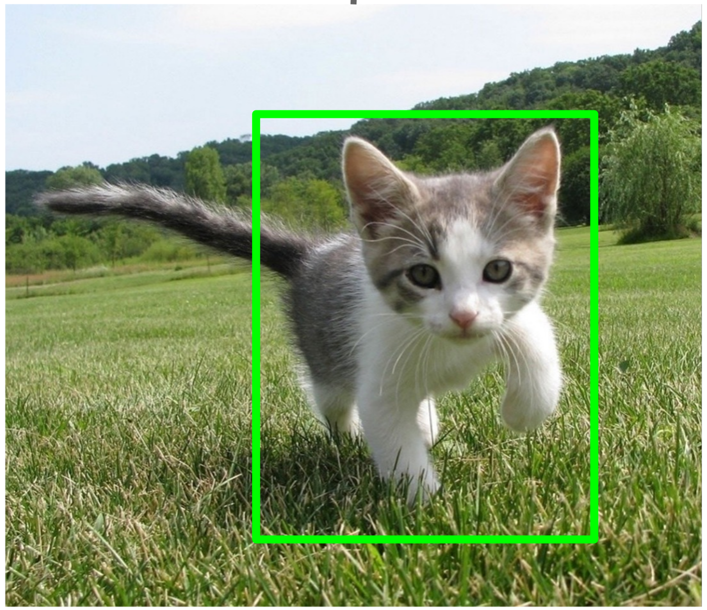
Region features always the same size even if input regions have different sizes!

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

Slide from Justin Johnson

Cropping Features: RoI Pool

Project proposal onto features



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

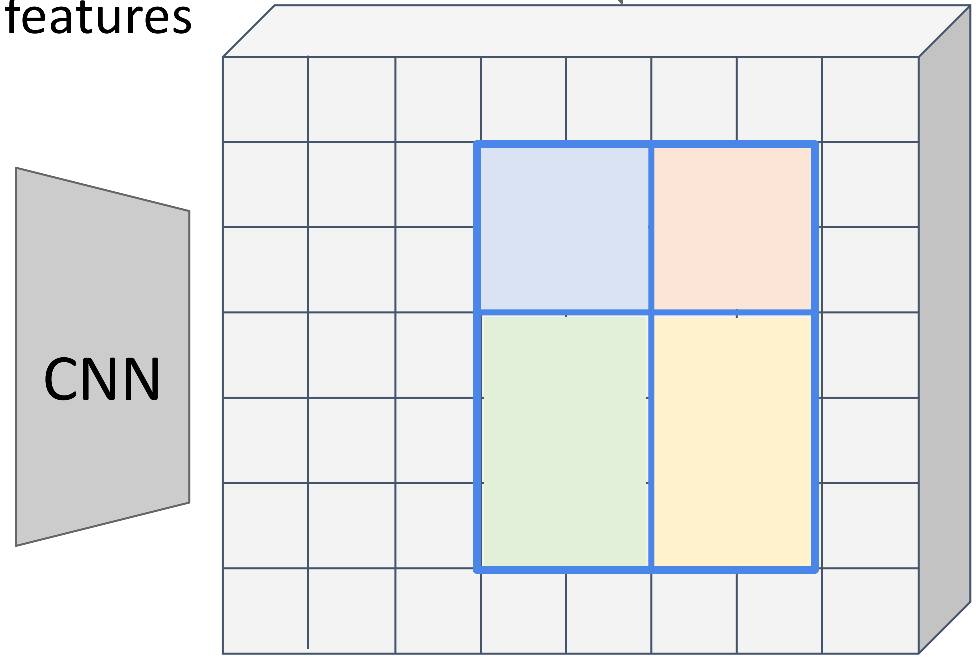
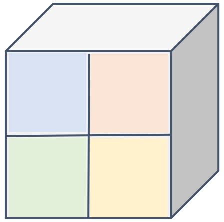


Image features
(e.g. 512 x 20 x 15)

“Snap” to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion



Region features
(here 512 x 2 x 2;
In practice 512x7x7)

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

Problem: Slight misalignment due to snapping; different-sized subregions is weird

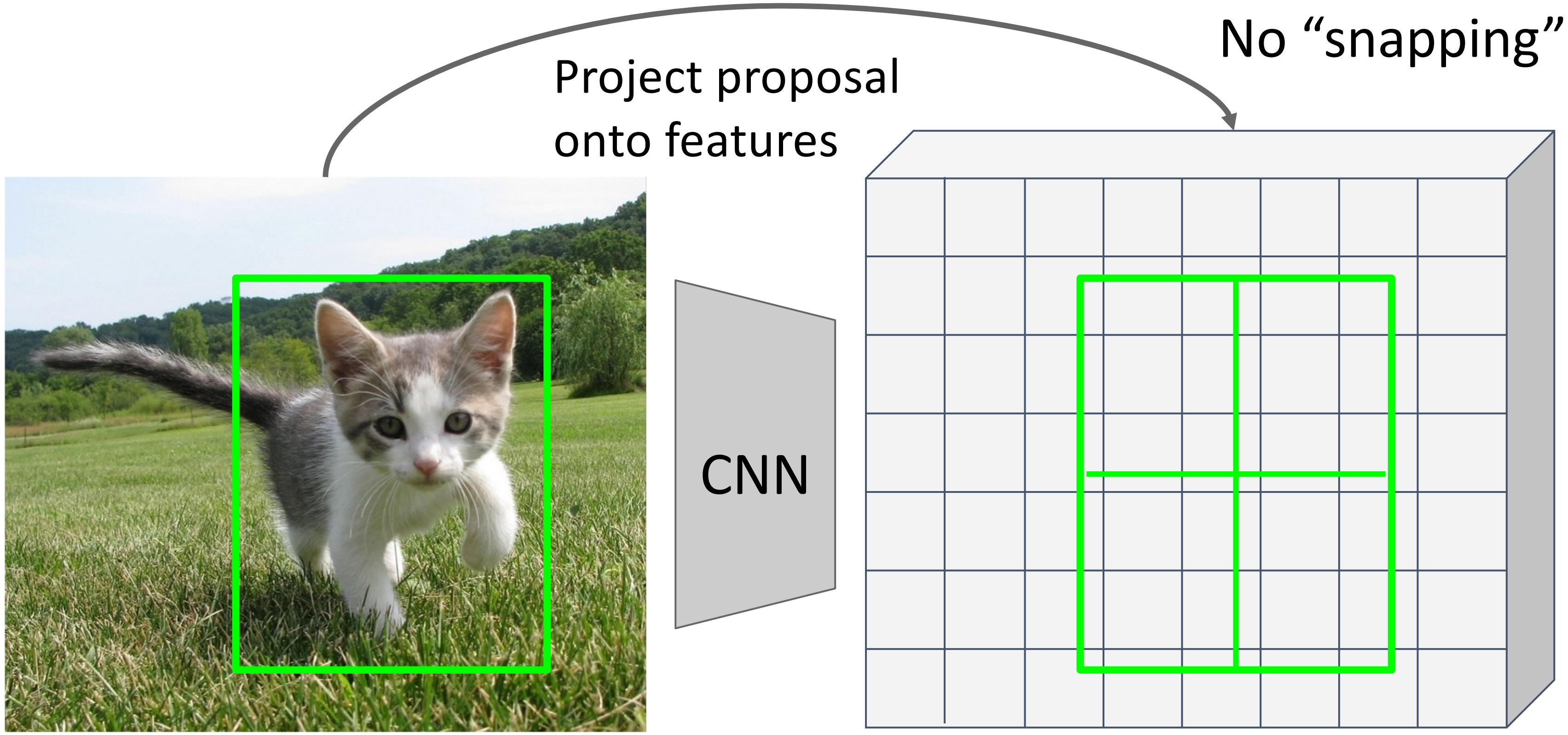
Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Align

Divide into equal-sized subregions
(may not be aligned to grid!)

No "snapping"!

Project proposal
onto features



Want features for the
box of a fixed size
(2x2 in this example,
7x7 or 14x14 in practice)

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

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

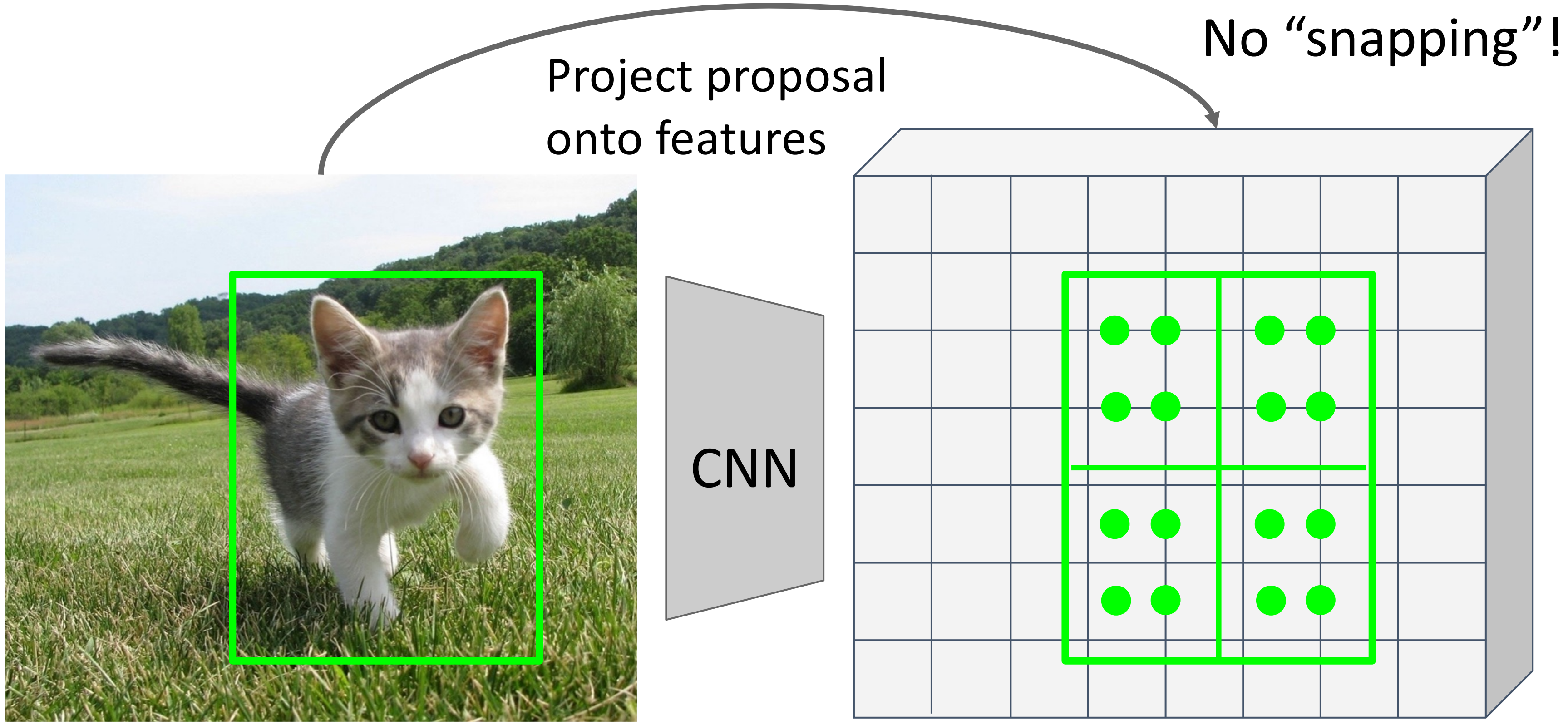
He et al, "Mask R-CNN", ICCV 2017.

Slide from Justin Johnson

Cropping Features: RoI Align

Divide into equal-sized subregions (may not be aligned to grid!)

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**



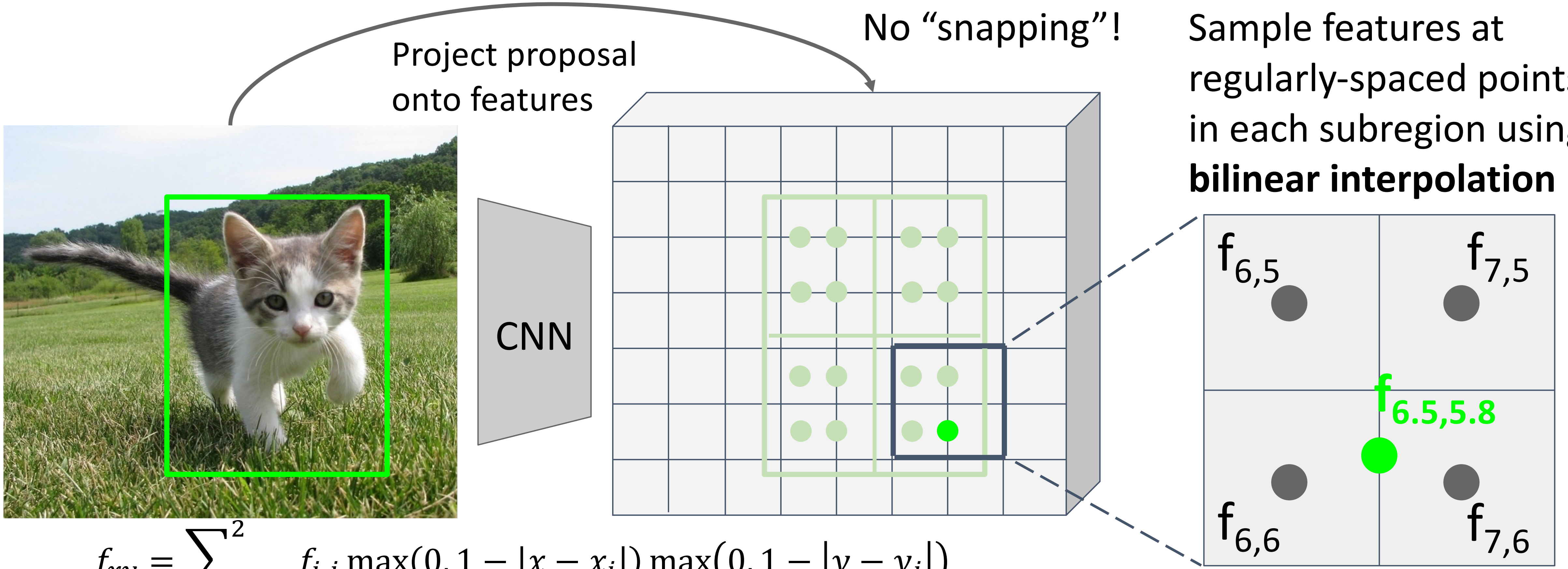
Input Image
(e.g. 3 x 640 x 480)

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

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

Cropping Features: RoI Align

Divide into equal-sized subregions (may not be aligned to grid!)



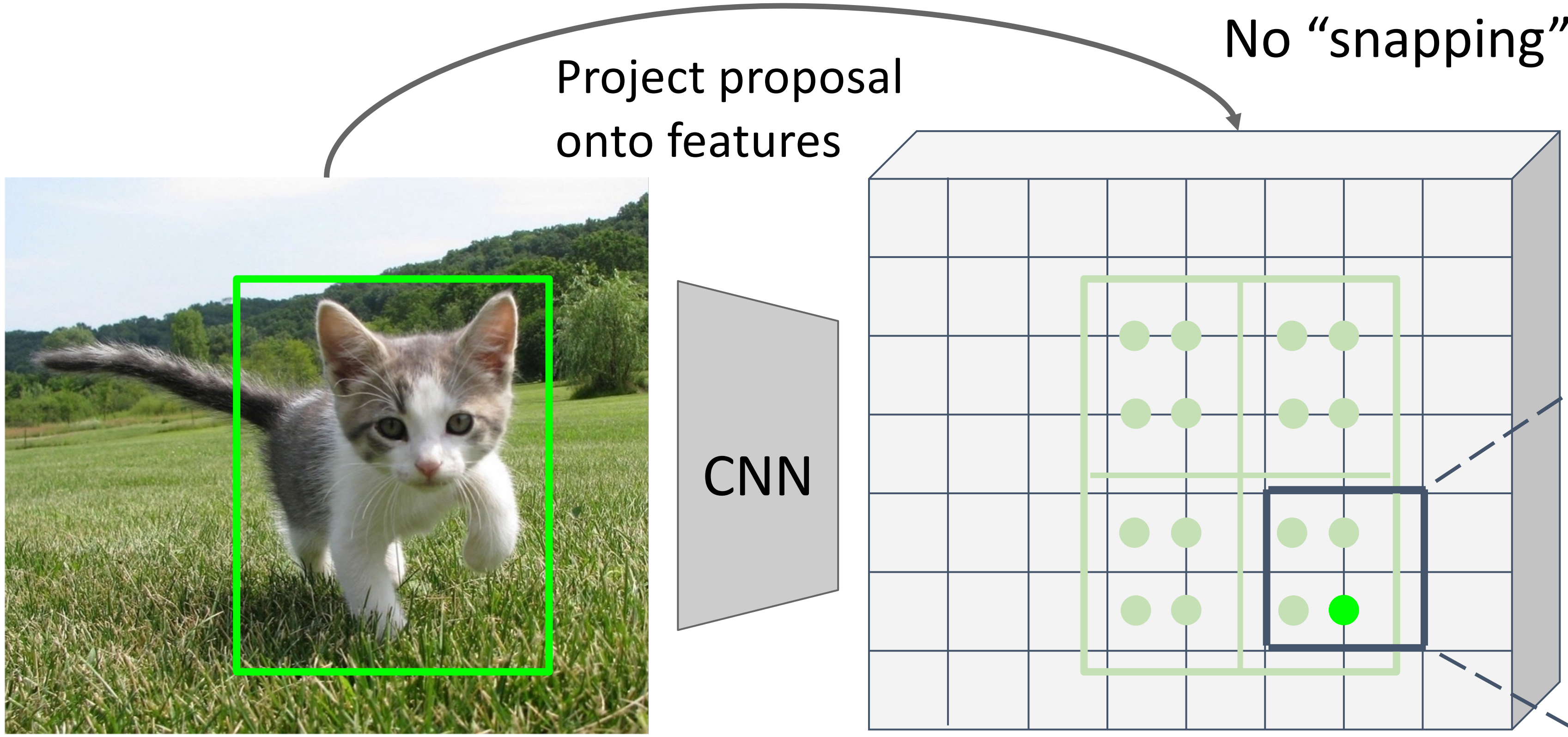
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

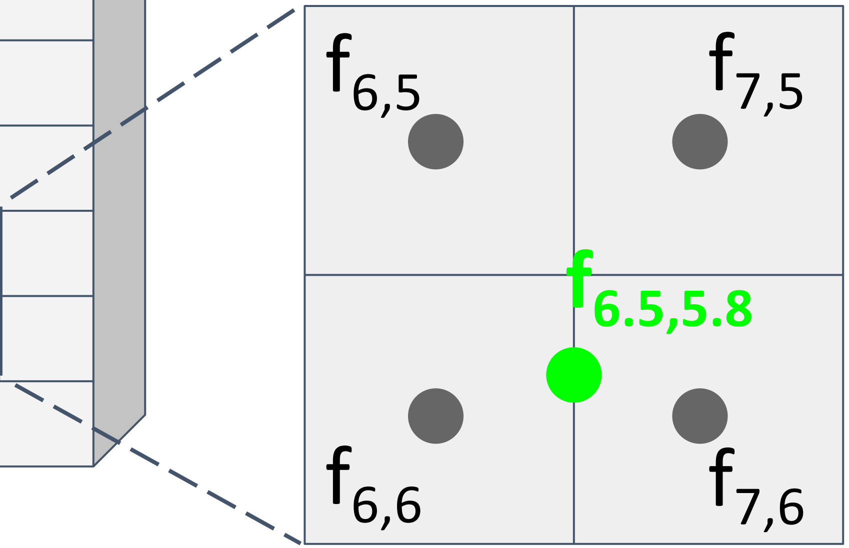
$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

Cropping Features: RoI <u>Align</u>

Divide into equal-sized subregions (may not be aligned to grid!)



Sample features at regularly-spaced points in each subregion using **bilinear interpolation**



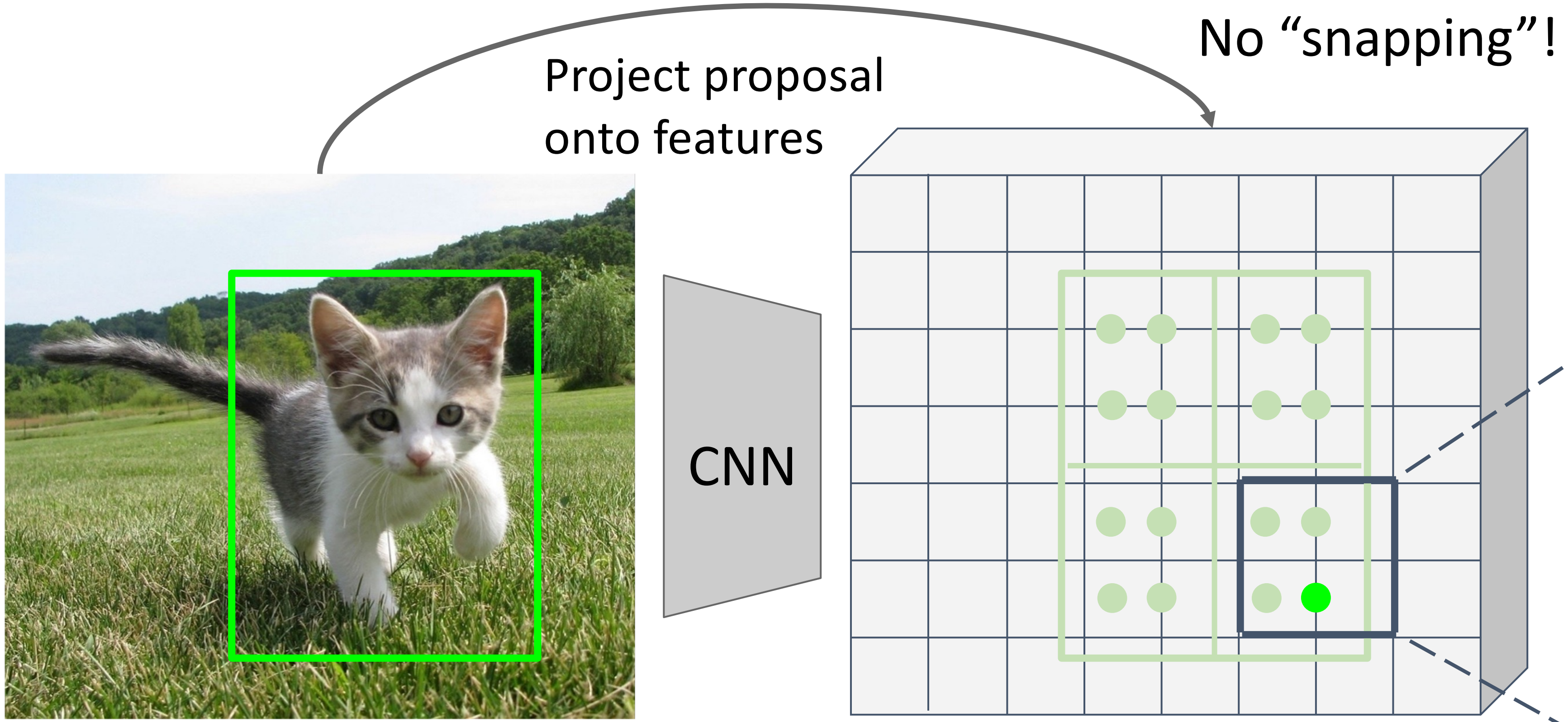
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

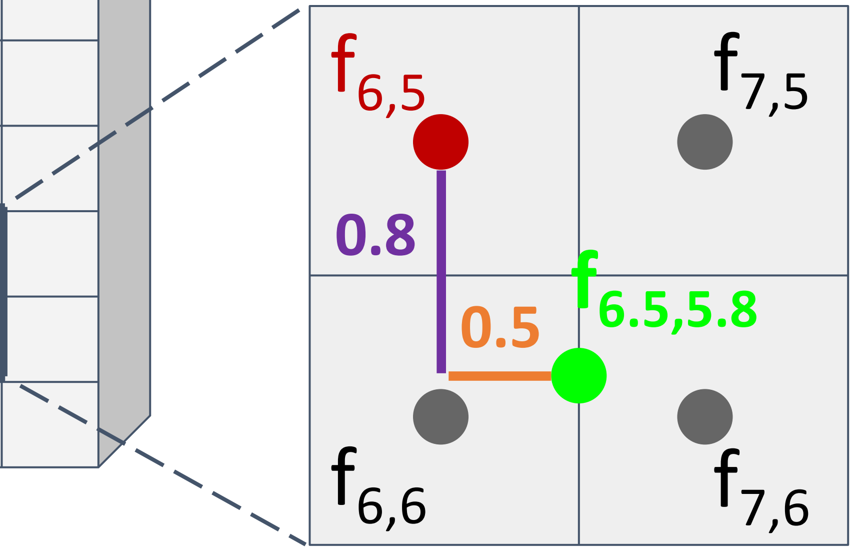
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

Slide from Justin Johnson

Cropping Features: RoI Align



Sample features at regularly-spaced points in each subregion using **bilinear interpolation**



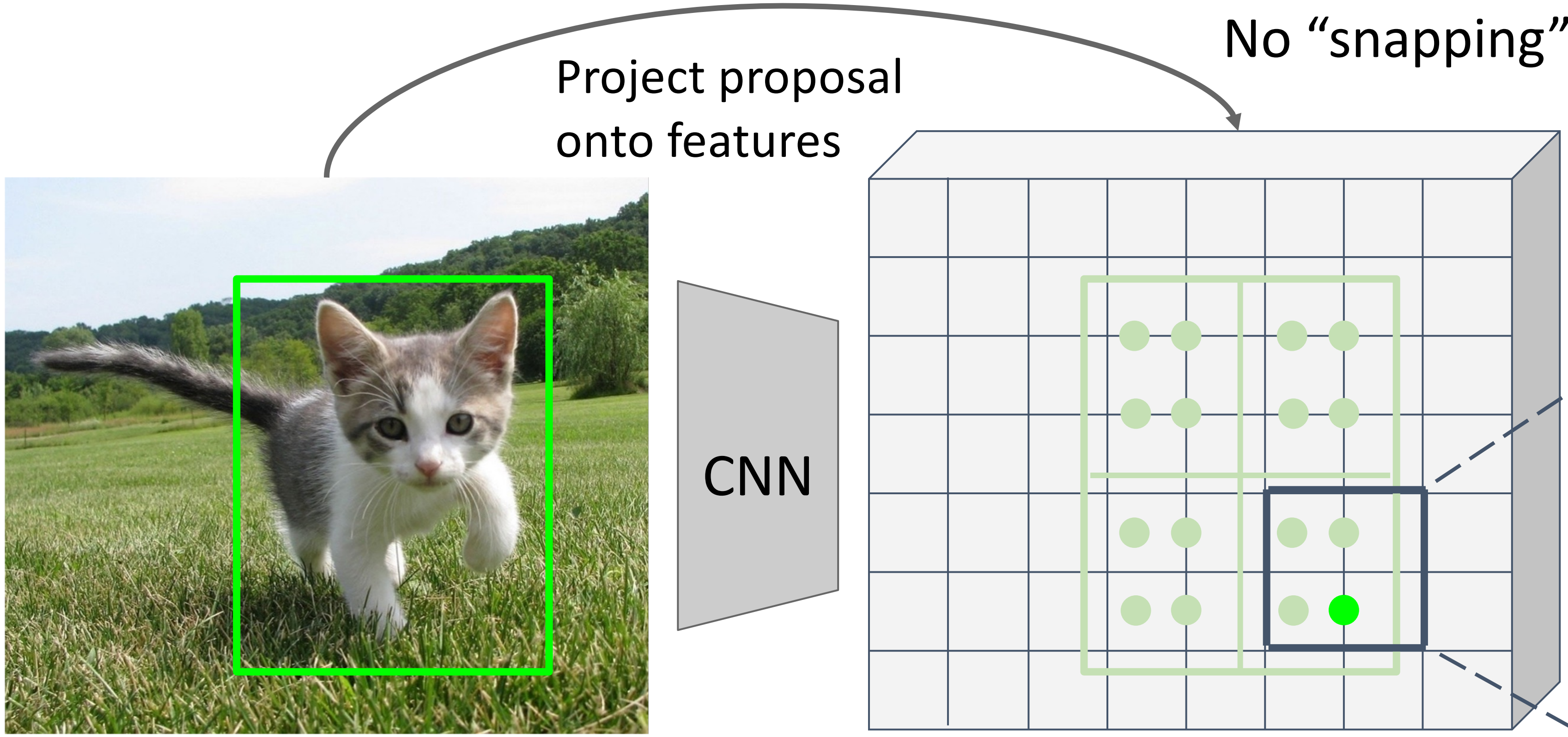
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

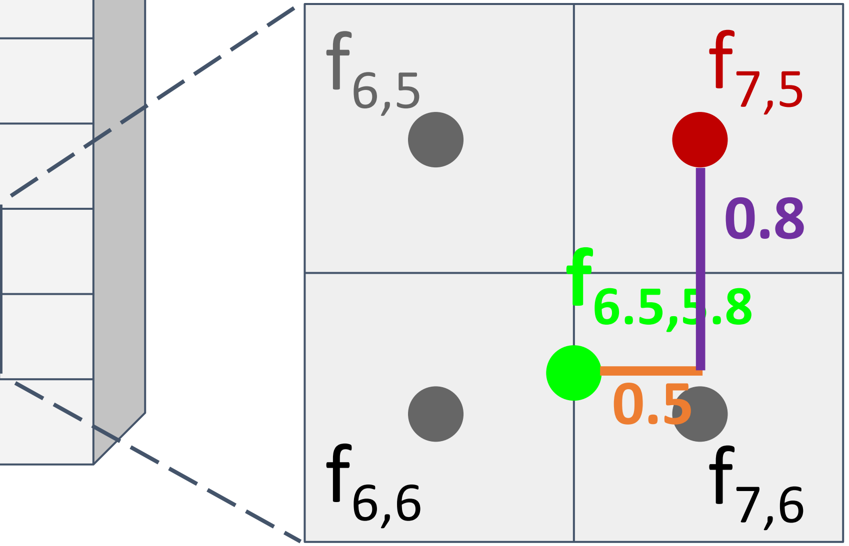
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

Slide from Justin Johnson

Cropping Features: RoI Align



Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

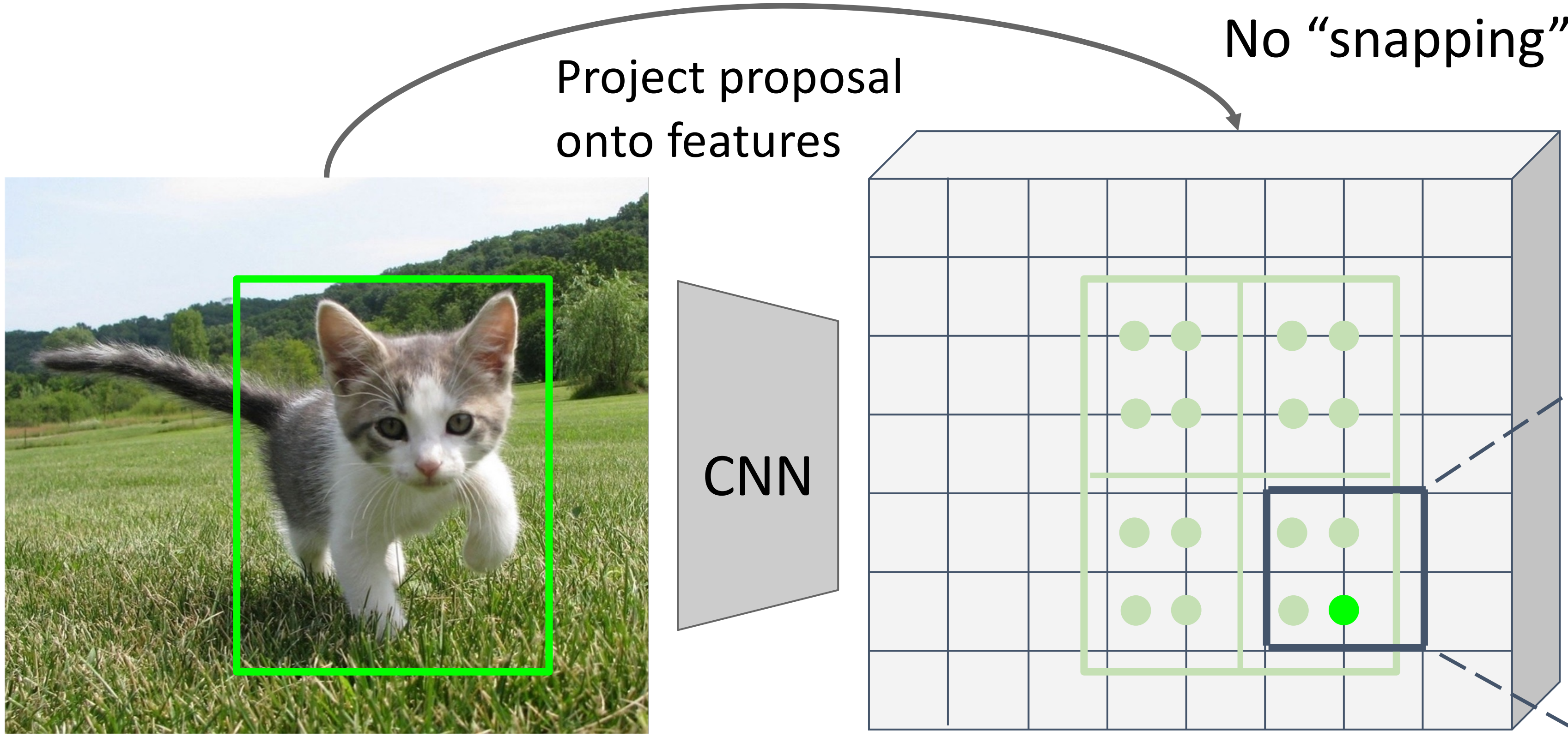


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$\begin{aligned} f_{6.5,5.8} &= (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * \mathbf{0.5} * \mathbf{0.2}) \\ &\quad + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8) \end{aligned}$$

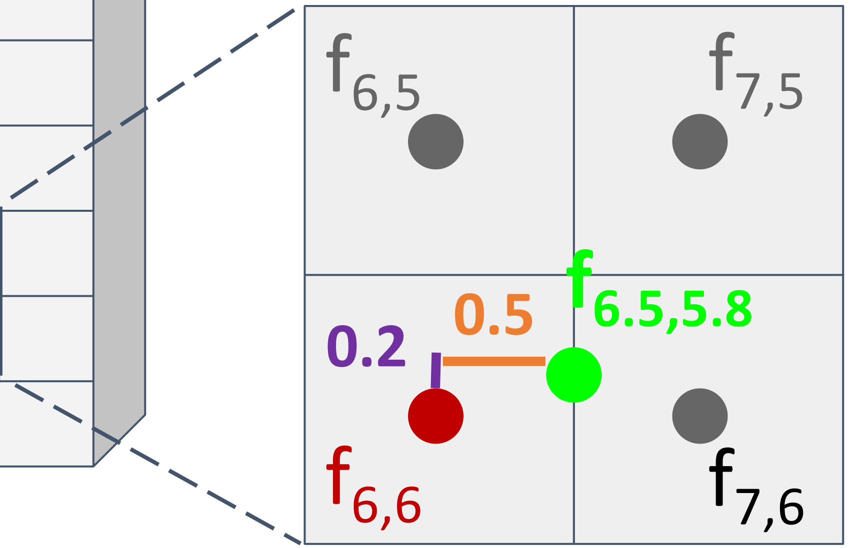
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

Cropping Features: RoI Align



No "snapping"!

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**



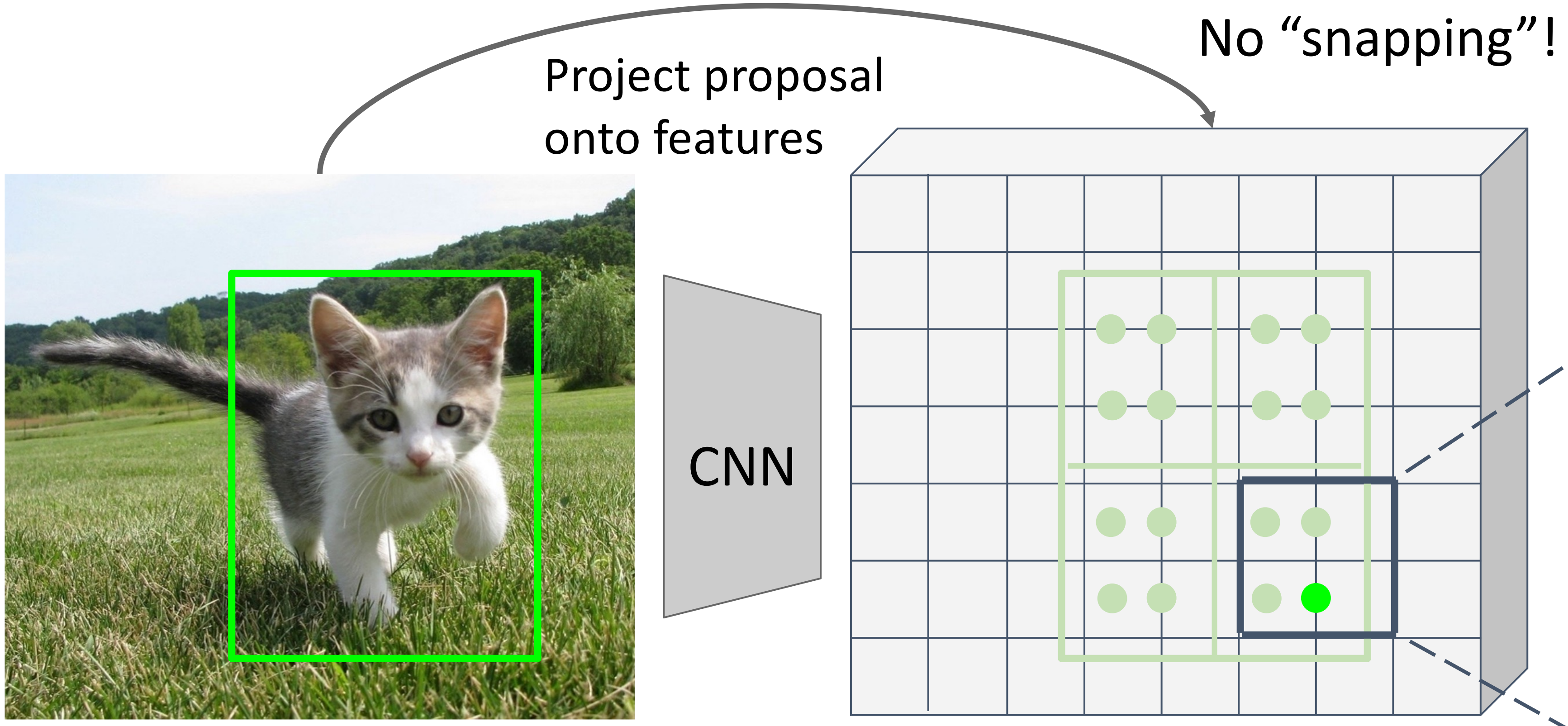
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$\begin{aligned} f_{6.5,5.8} &= (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) \\ &+ (f_{6,6} * \mathbf{0.5} * \mathbf{0.8}) + (f_{7,6} * 0.5 * 0.8) \end{aligned}$$

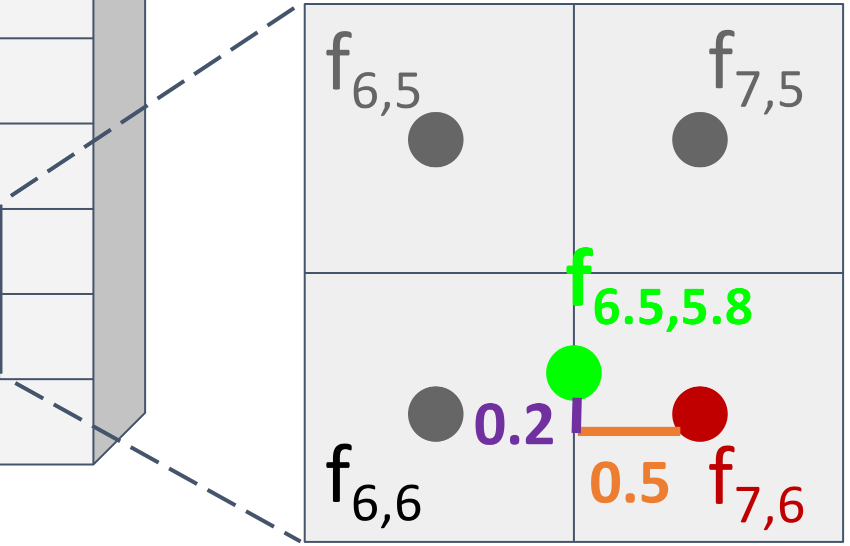
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

Slide from Justin Johnson

Cropping Features: RoI Align



Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

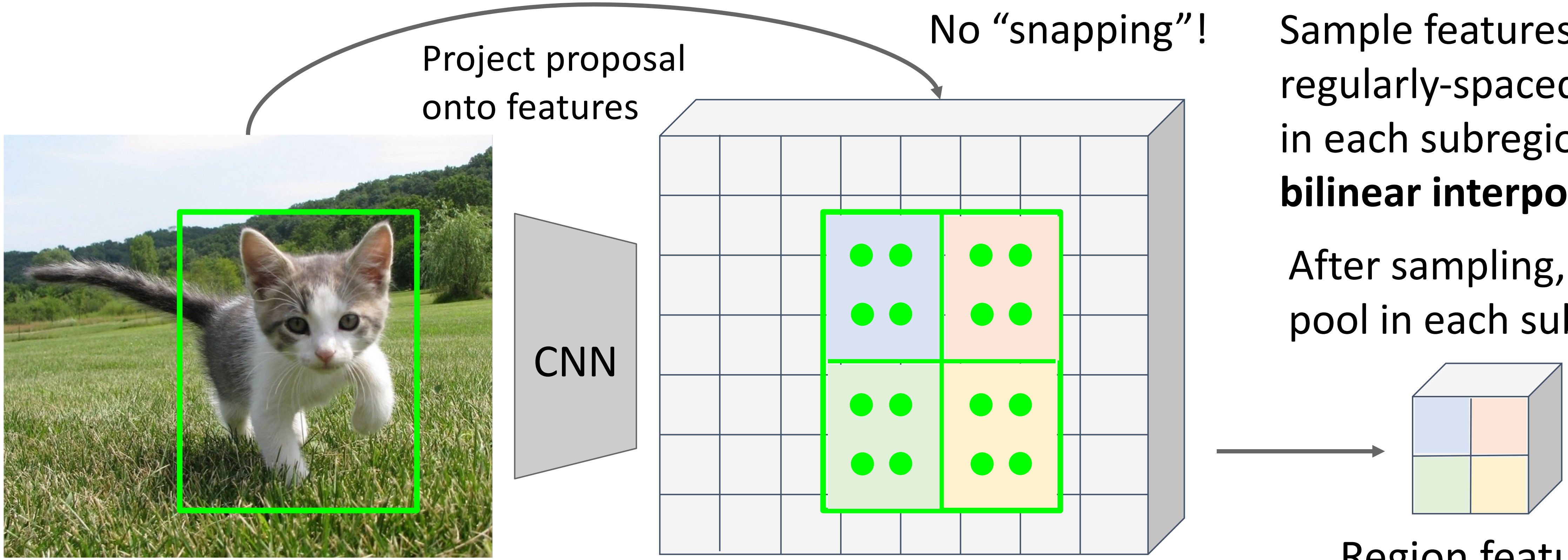


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align



Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

After sampling, max-pool in each subregion

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

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

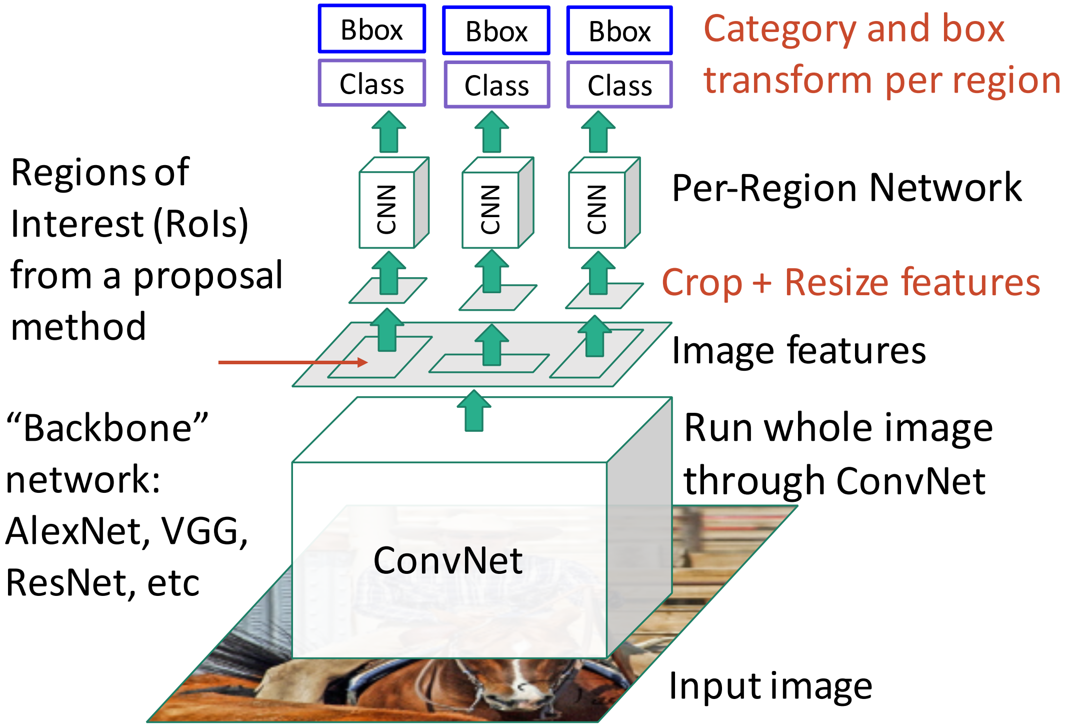
Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

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

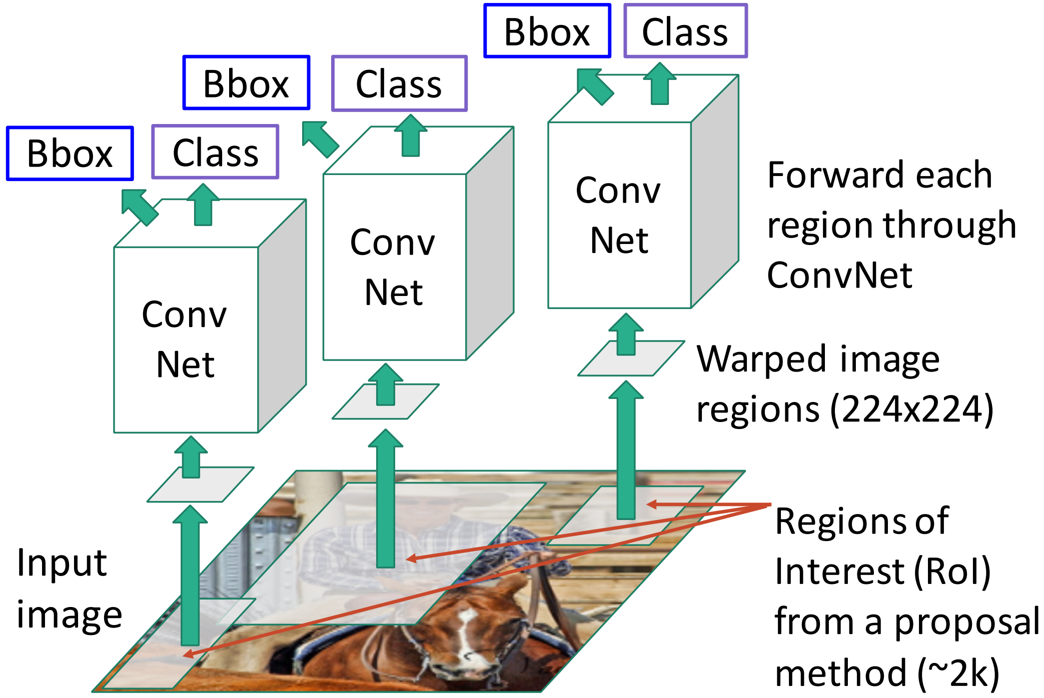
Slide from Justin Johnson

Fast R-CNN vs “Slow” R-CNN

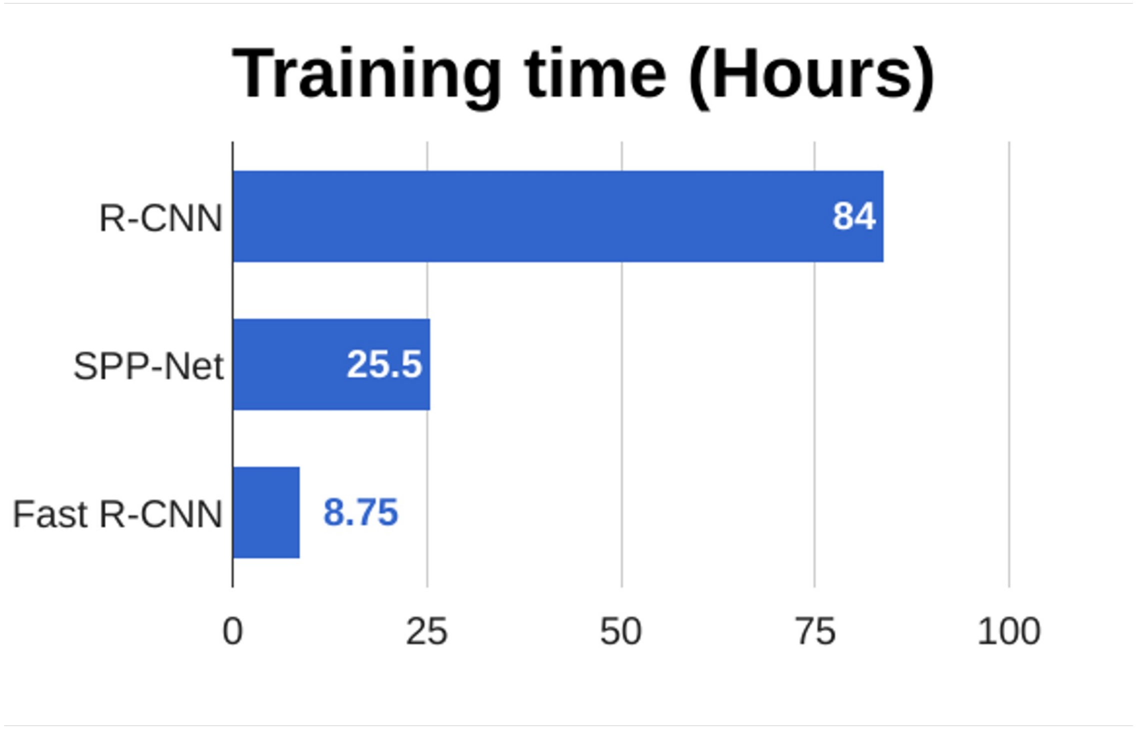
Fast R-CNN: Apply differentiable cropping to shared image features

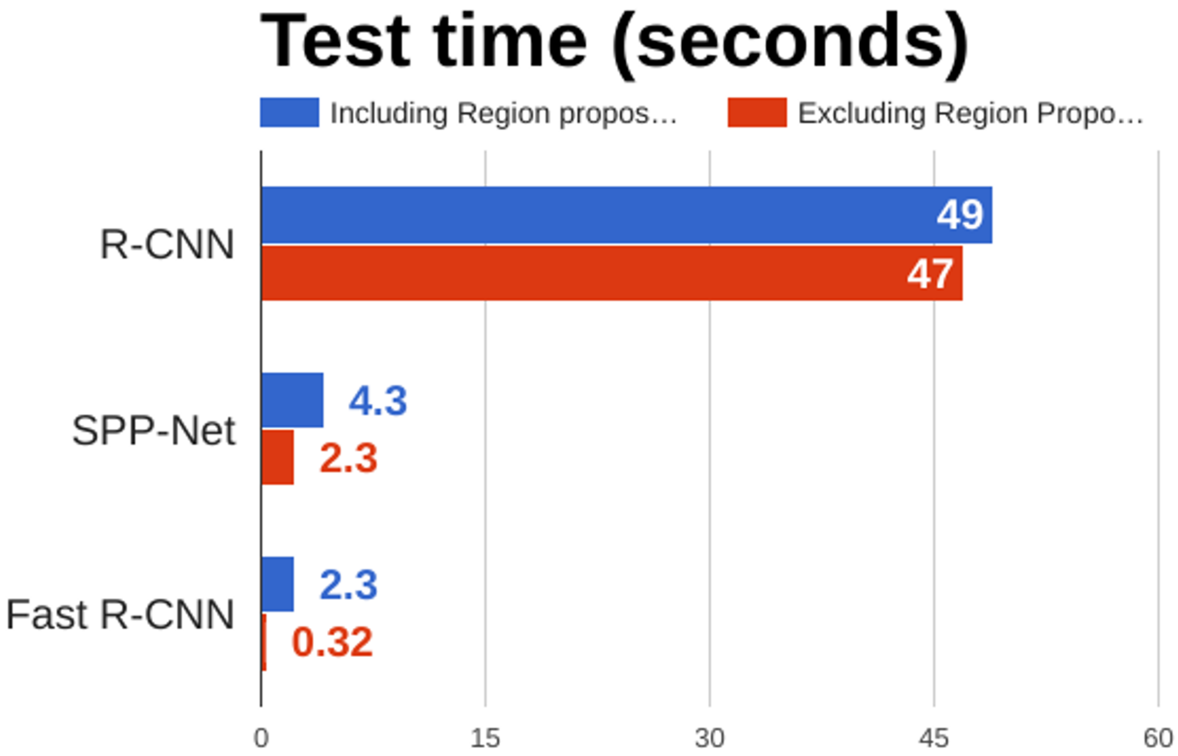


“Slow” R-CNN: Apply differentiable cropping to shared image features



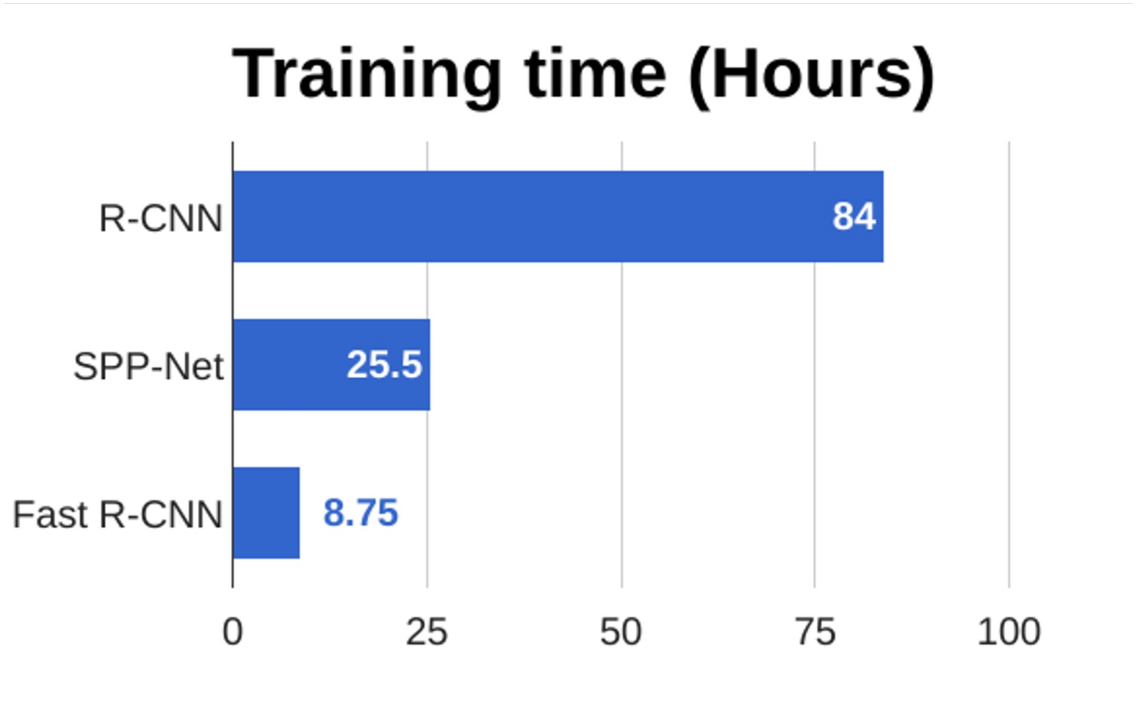
Fast R-CNN vs "Slow" R-CNN

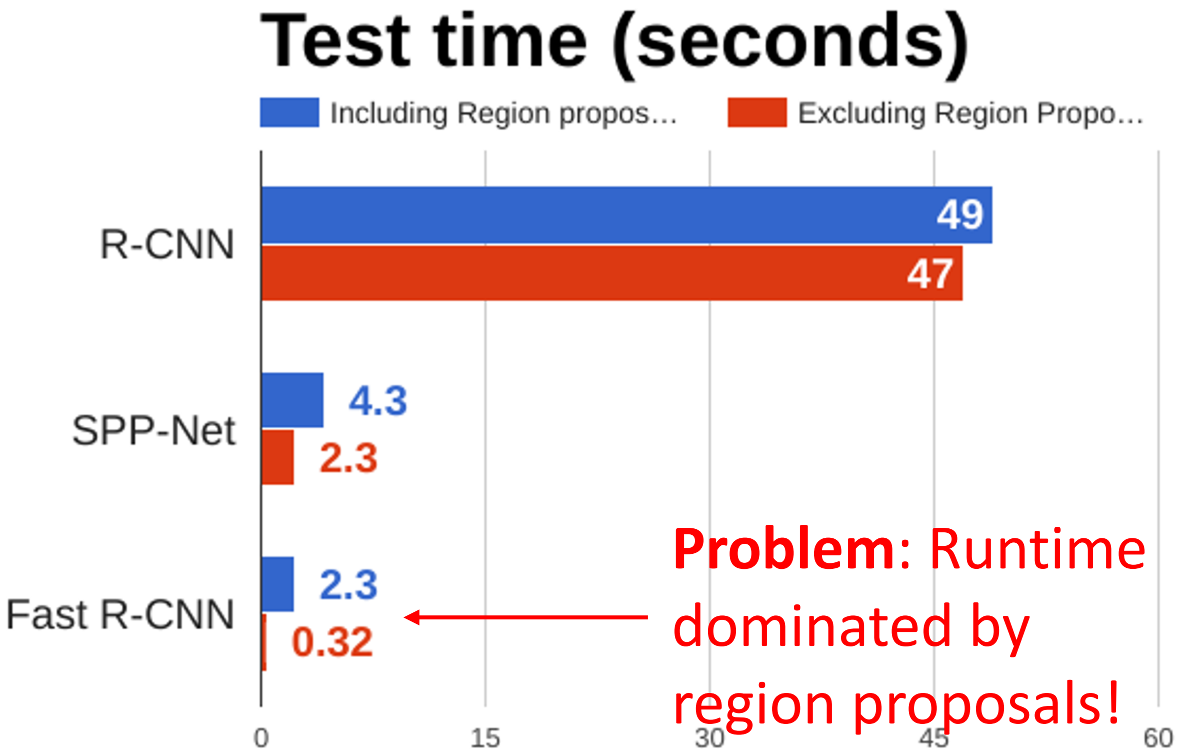




Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN vs “Slow” R-CNN

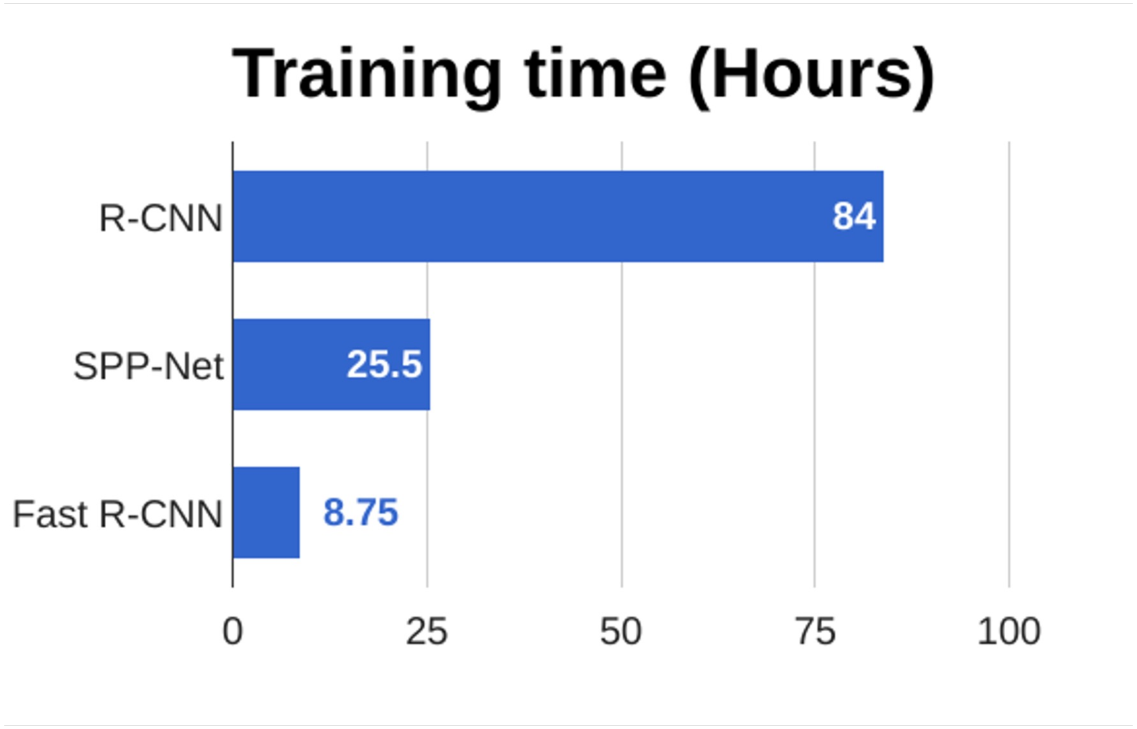


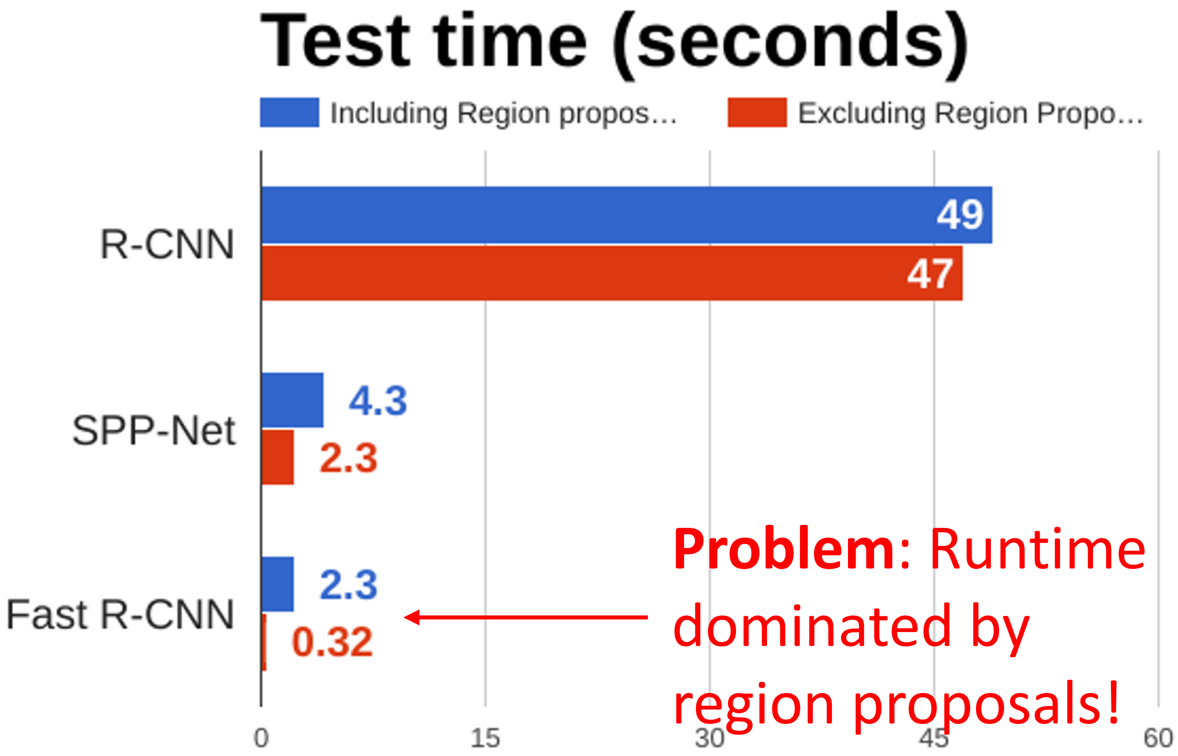


Problem: Runtime dominated by region proposals!

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

Fast R-CNN vs “Slow” R-CNN





Recall: Region proposals computed by heuristic ”Selective Search” algorithm on CPU -- let’s learn them with a CNN instead!

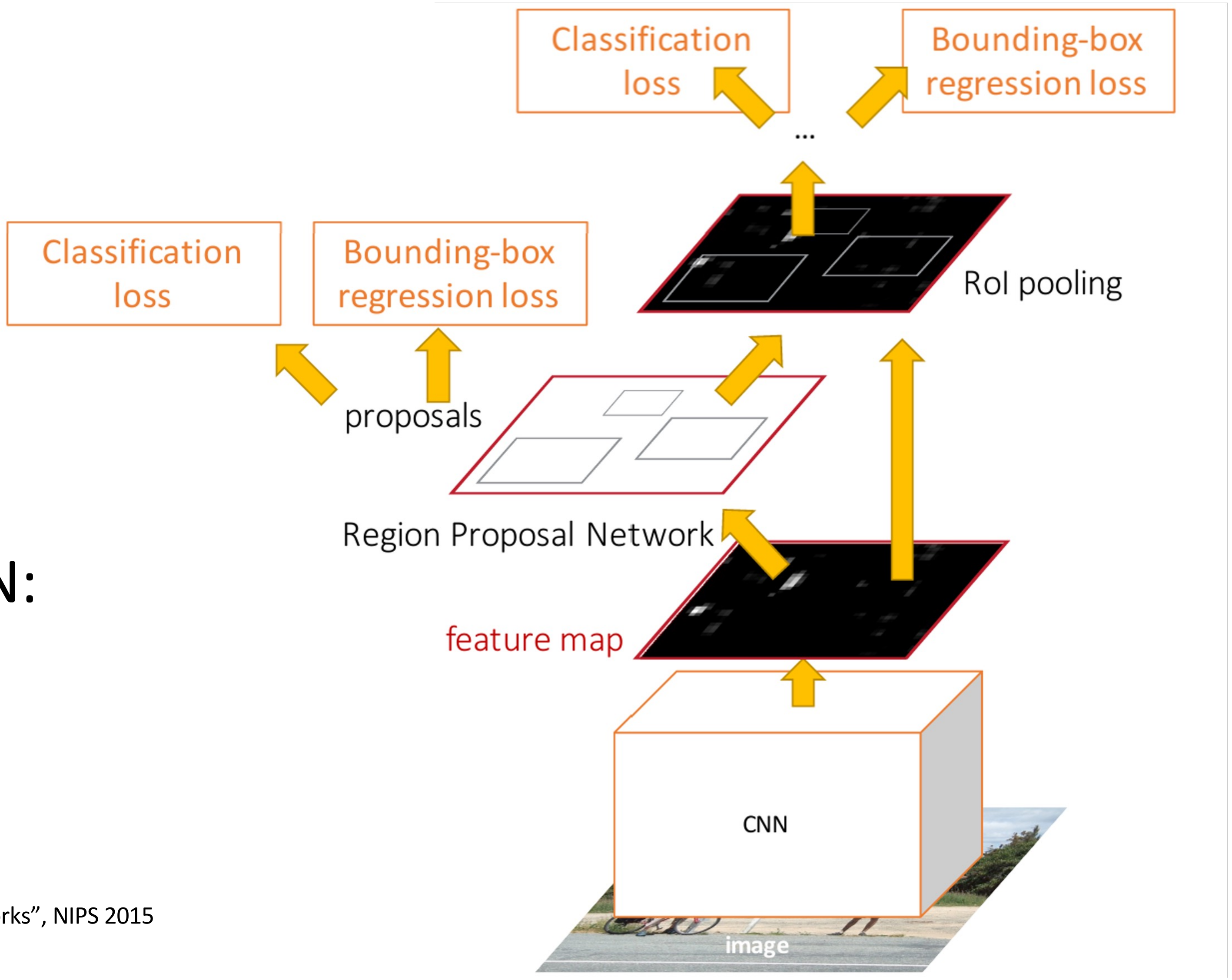
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Slide from Justin Johnson

Faster R-CNN: Learnable Region Proposals

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

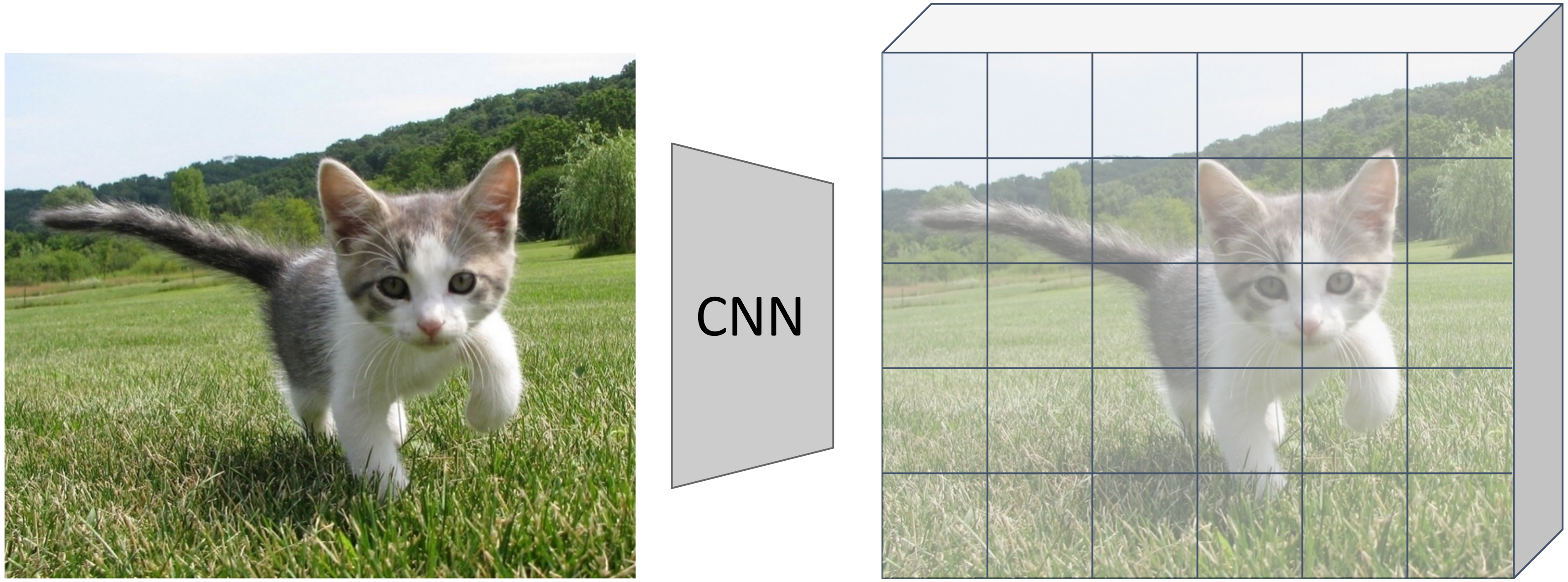
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



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

Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image



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

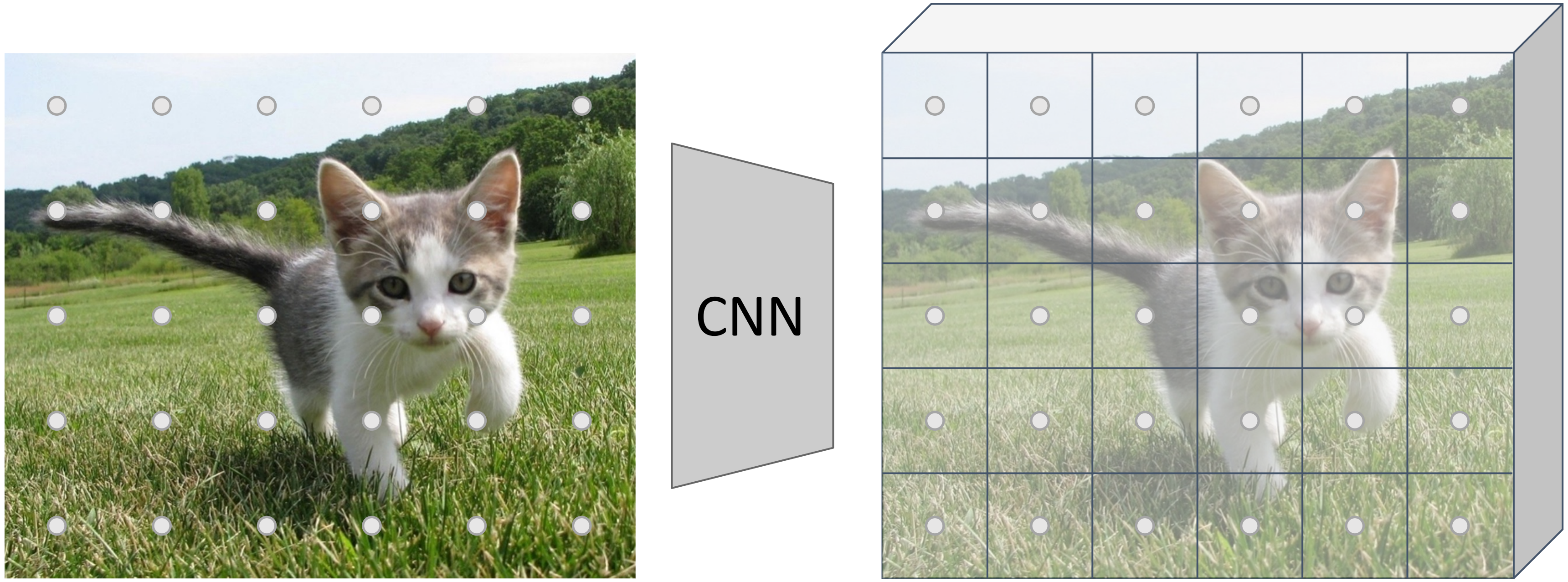
Image features
(e.g. 512 x 5 x 6)

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

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

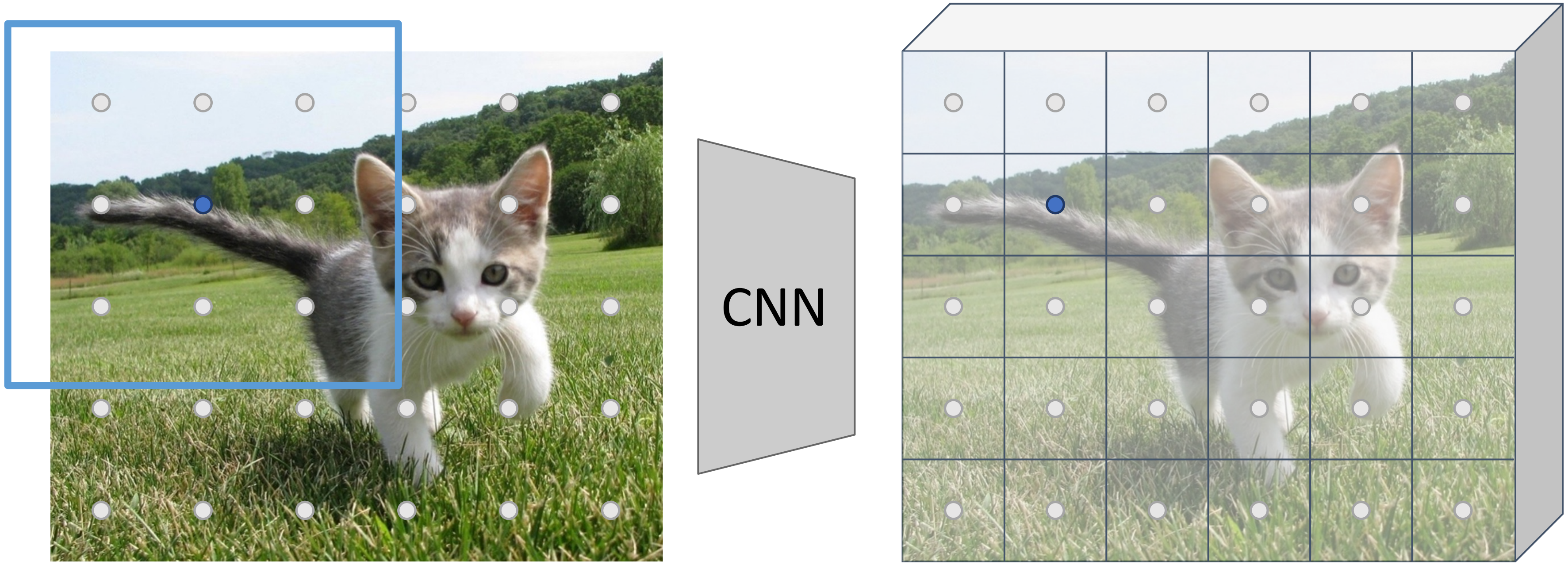
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

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

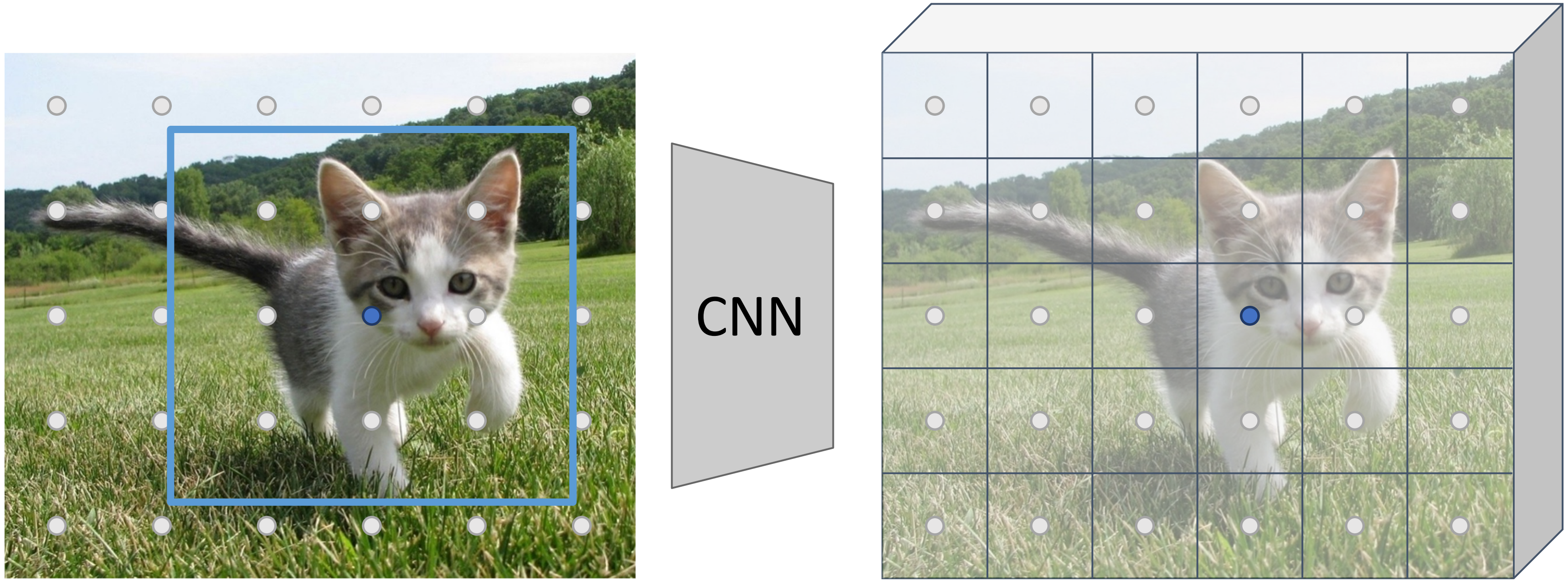
Slide from Justin Johnson

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

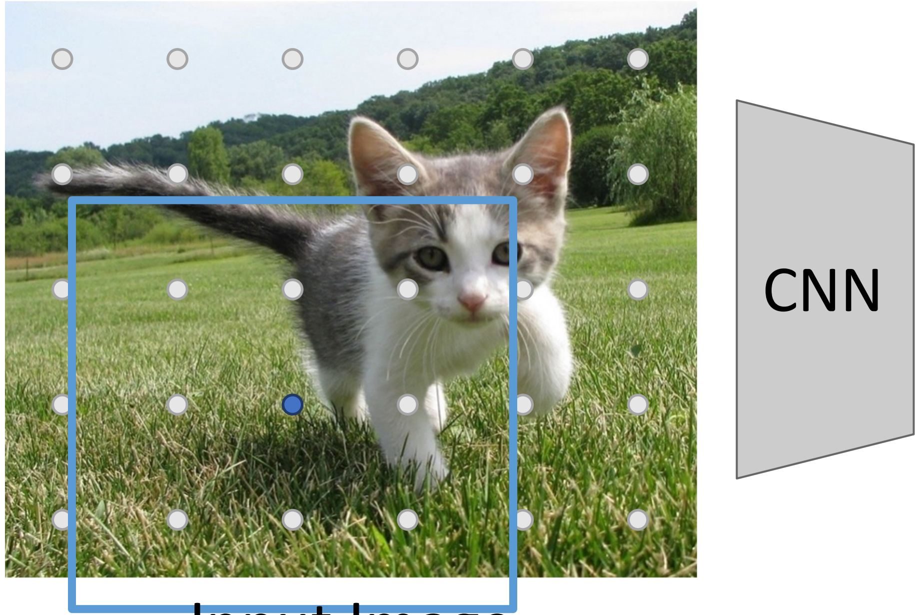
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

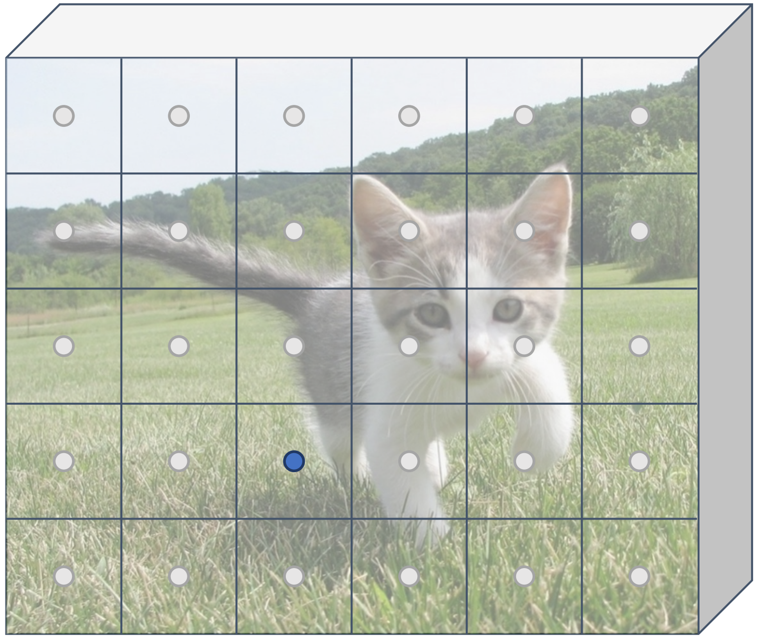


Image features
(e.g. 512 x 5 x 6)

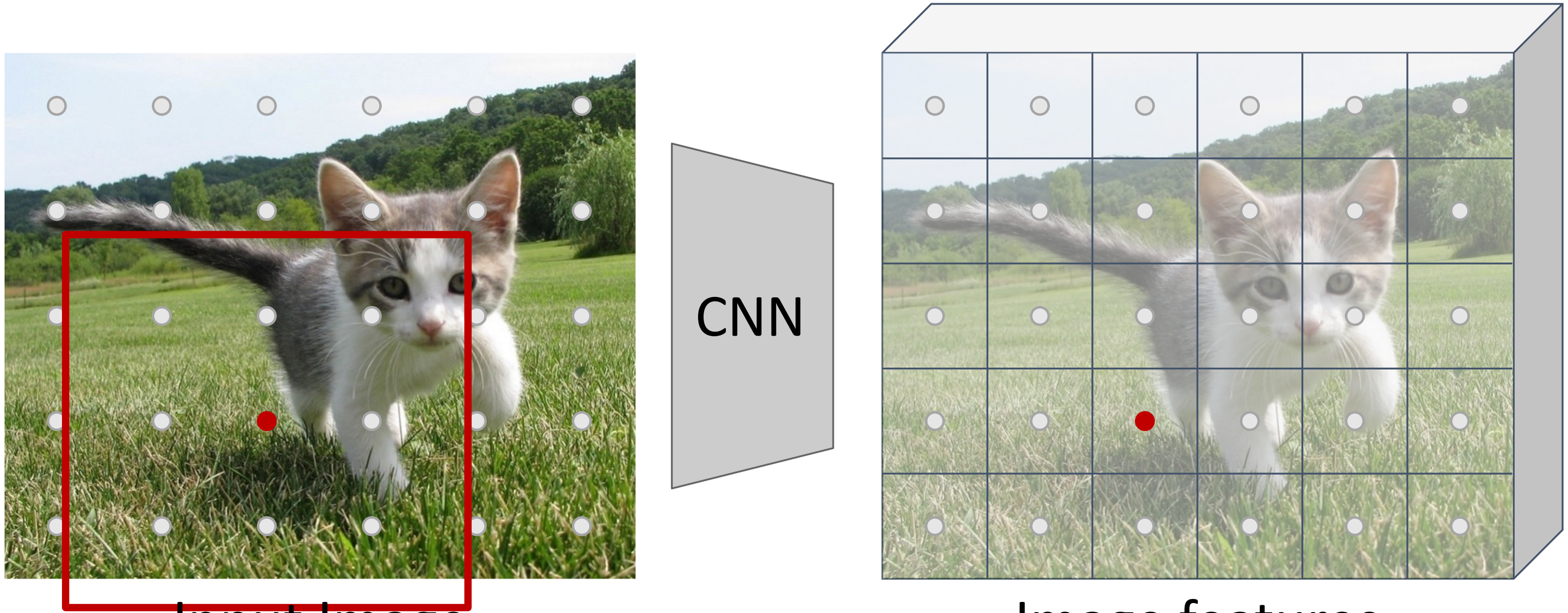
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

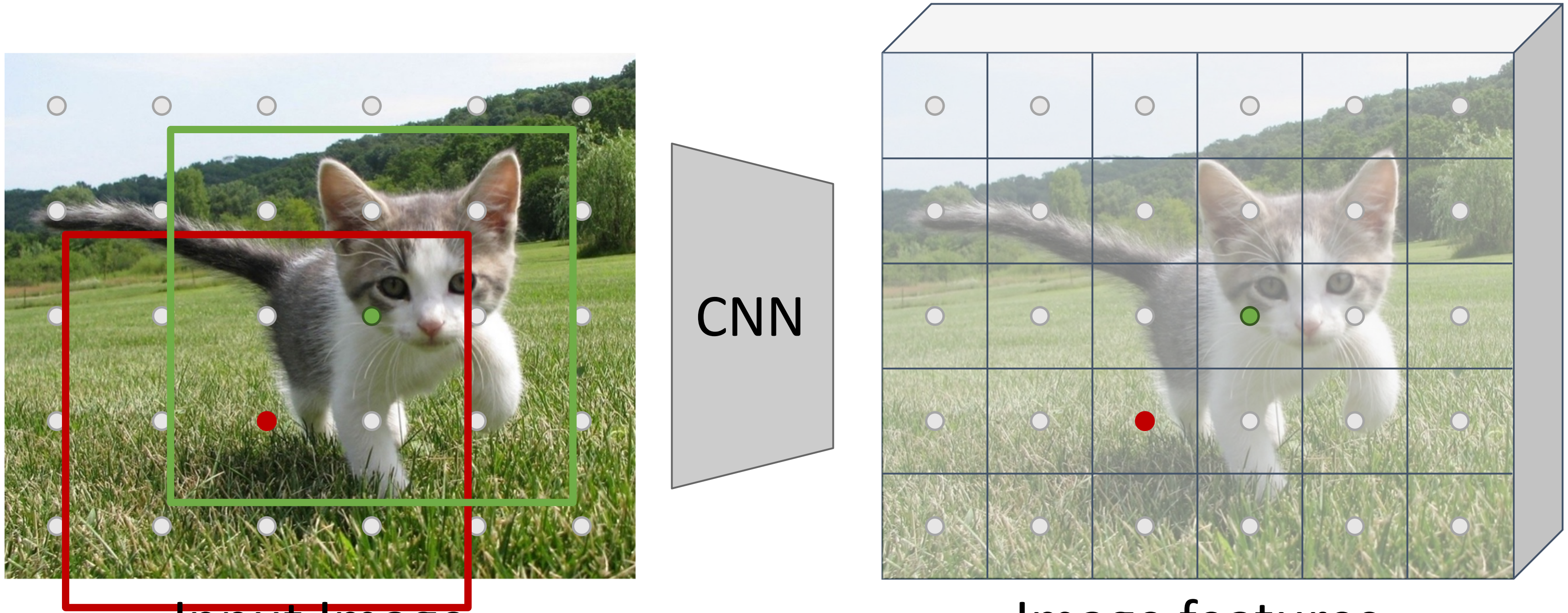
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

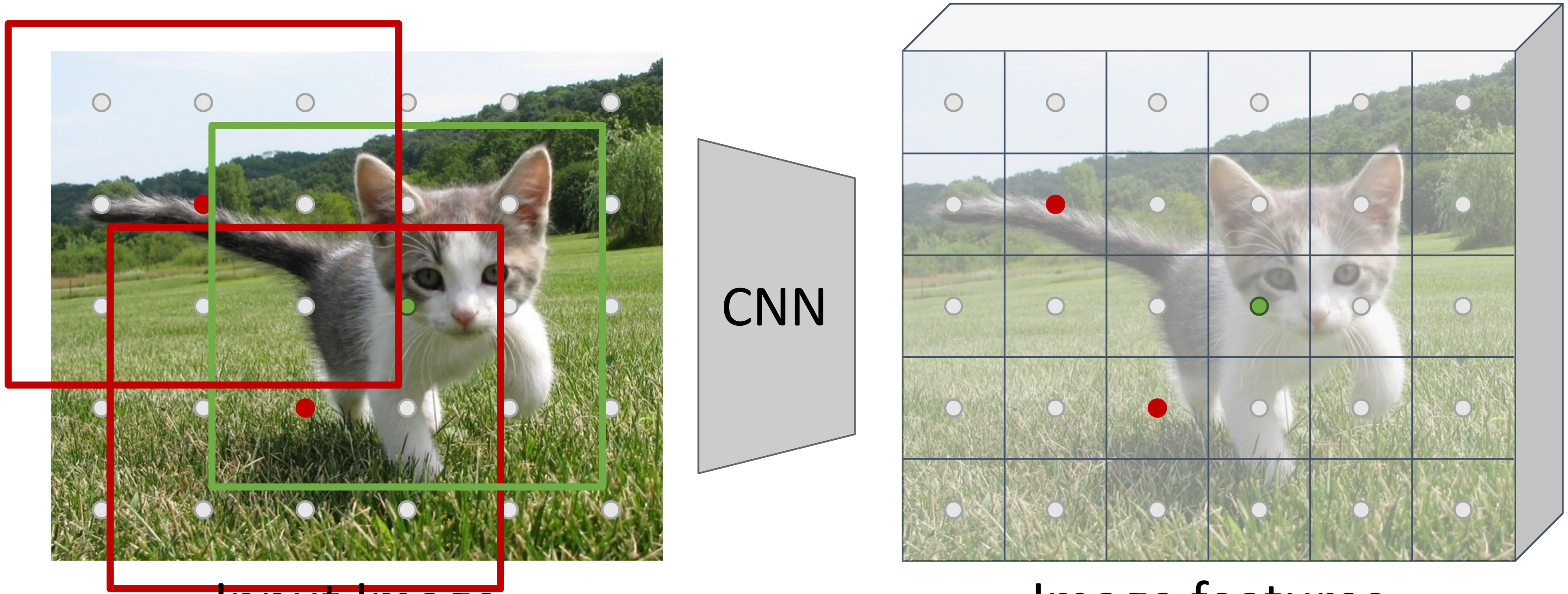
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

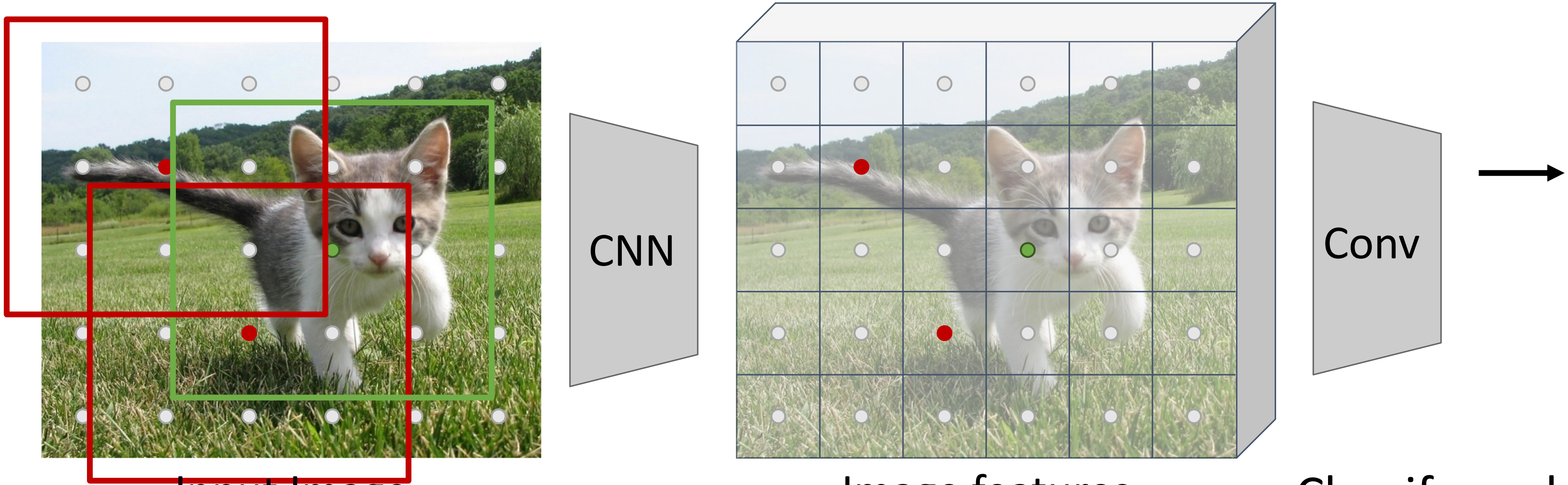
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Predict object vs not object scores for all anchors with a conv layer (512 input filters, 2 output filters)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Anchor is object?
2 x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

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

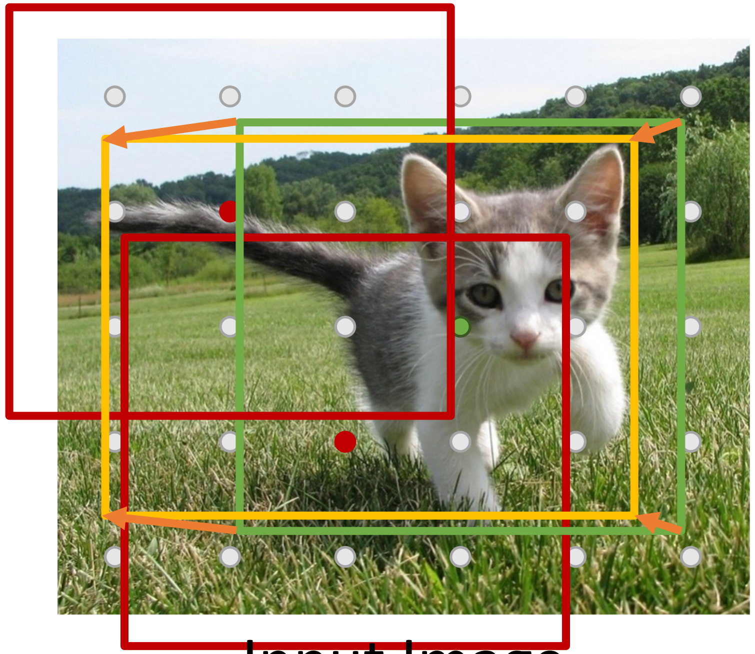
Slide from Justin Johnson

Region Proposal Network (RPN)

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

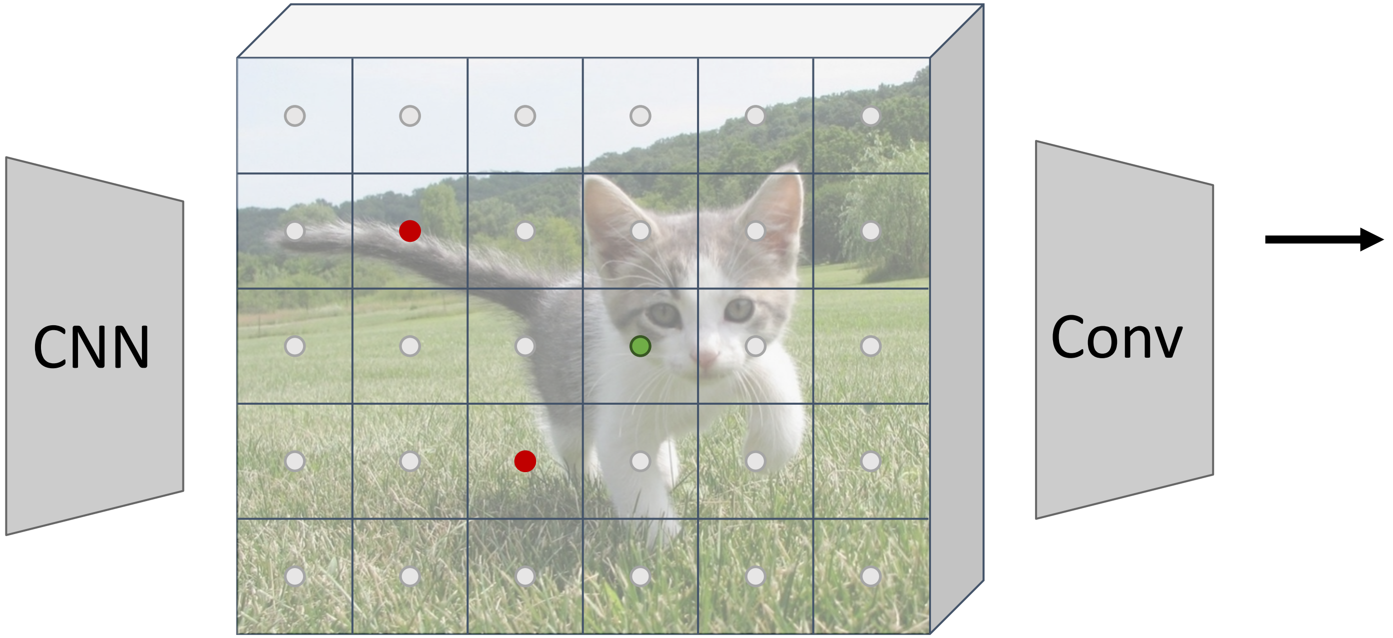


Image features
(e.g. 512 x 5 x 6)

Anchor is object?
2 x 5 x 6

Classify each anchor as positive (object) or negative (no object)

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

Slide from Justin Johnson

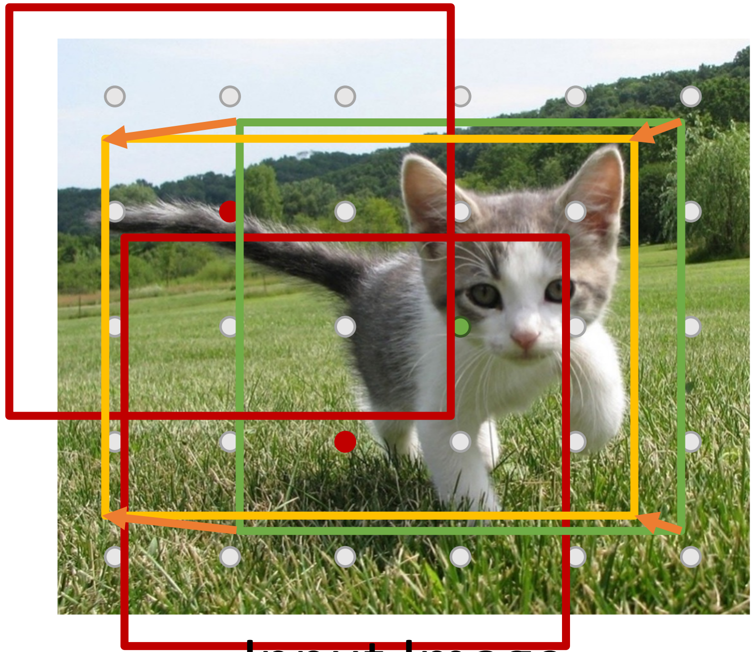
Region Proposal Network (RPN)

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)

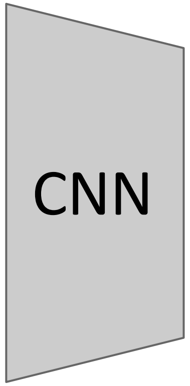
Predict transforms with conv

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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



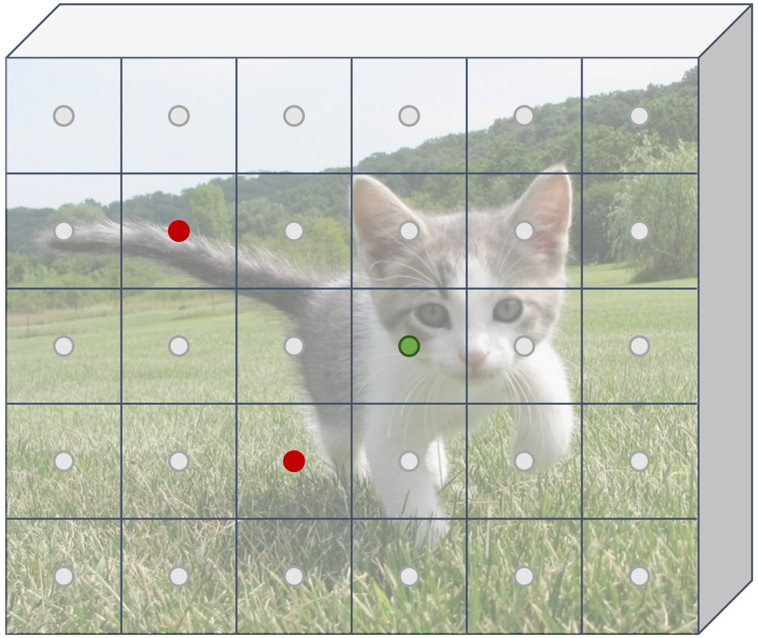
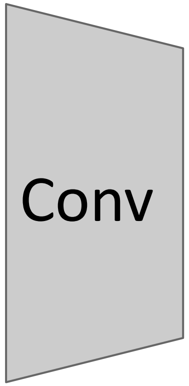


Image features
(e.g. 512 x 5 x 6)



→ Anchor is object?
2 x 5 x 6

→ Anchor transforms
4 x 5 x 6

Classify each anchor as positive (object) or negative (no object)

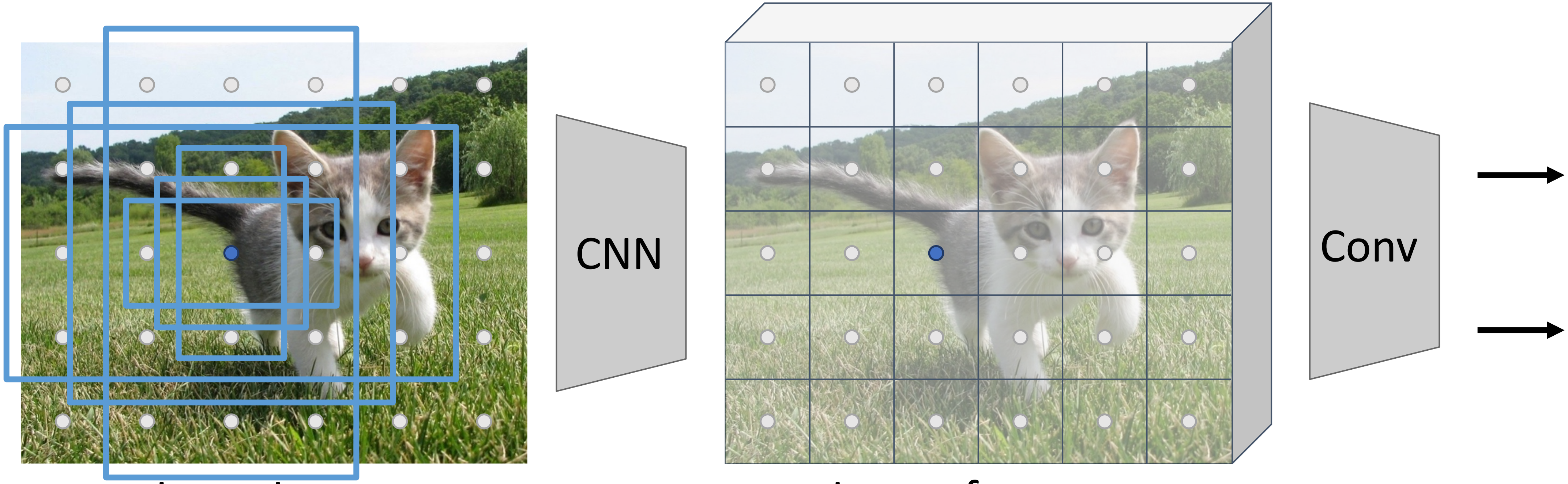
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

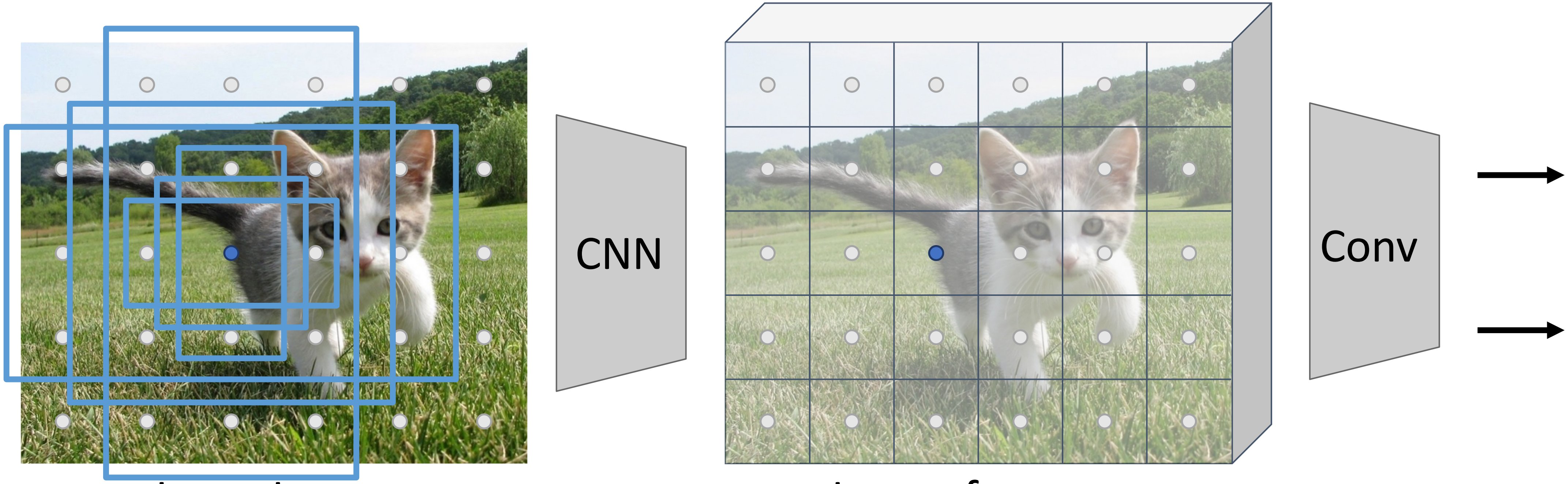
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

During training, supervised positive / negative anchors and box transforms like R-CNN

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

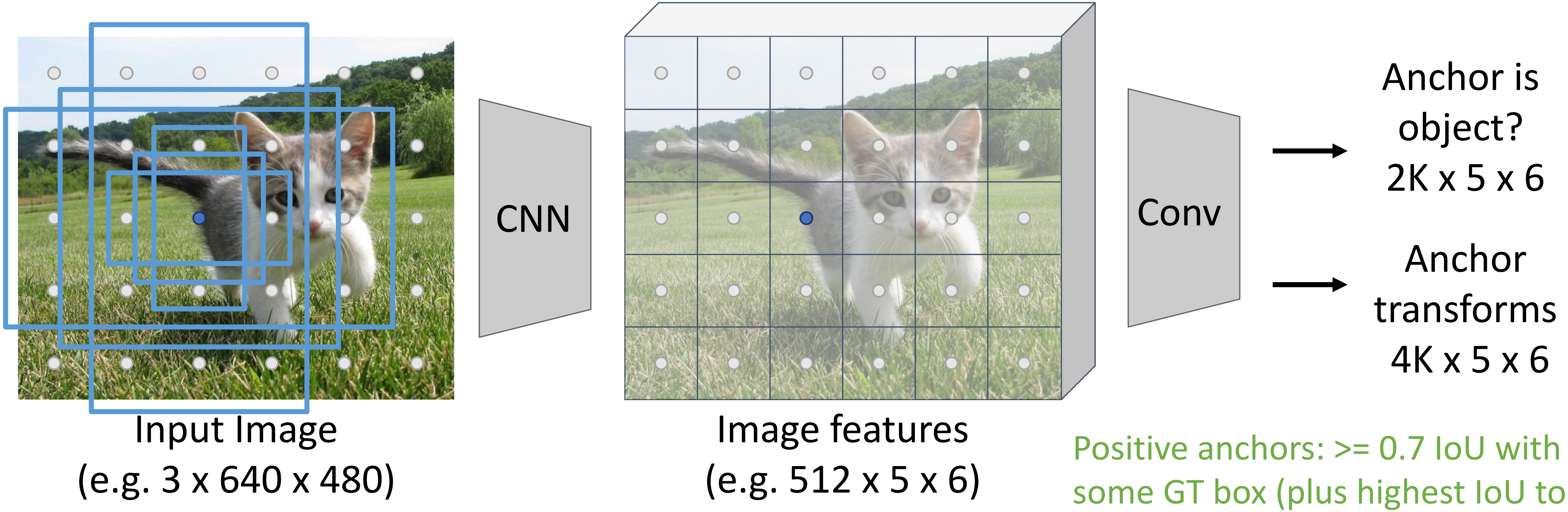
Slide from Justin Johnson

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Positive anchors: >= 0.7 IoU with some GT box (plus highest IoU to each GT)

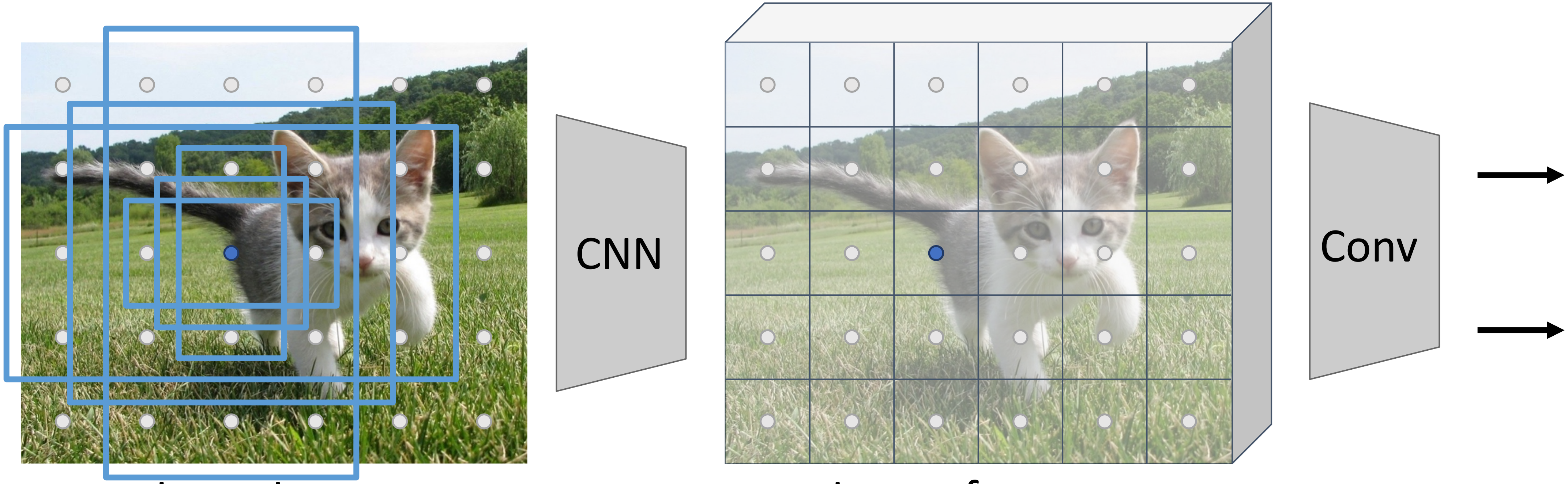
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

Negative anchors: < 0.3 IoU with all GT boxes. Don't supervised transforms for negative boxes.

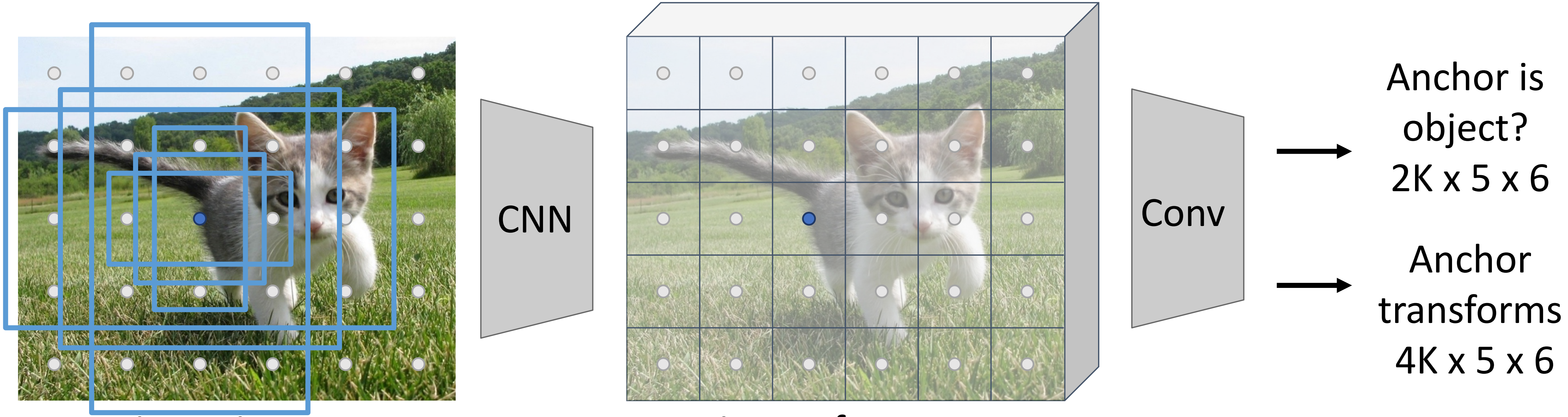
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Neutral anchors: between 0.3 and 0.7 IoU with all GT boxes; ignored during training

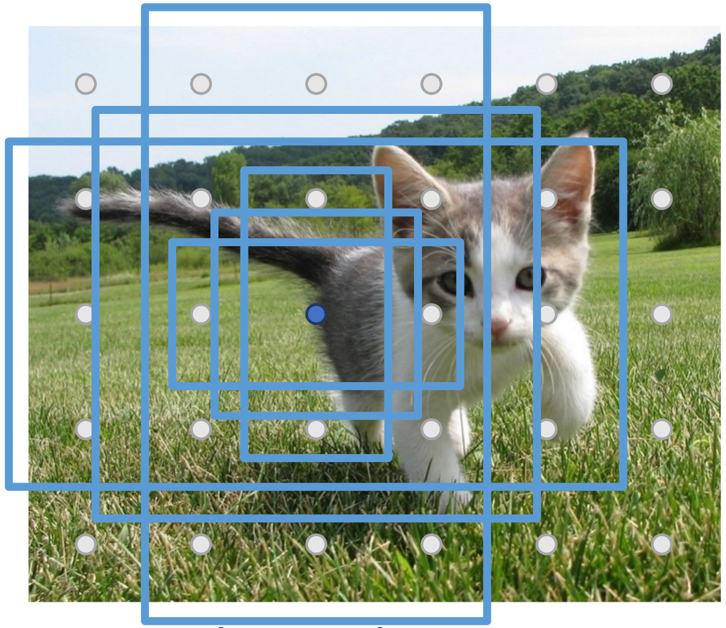
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Slide from Justin Johnson

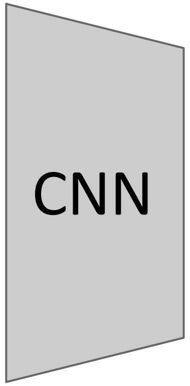
Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image



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



Each feature corresponds to a point in the input

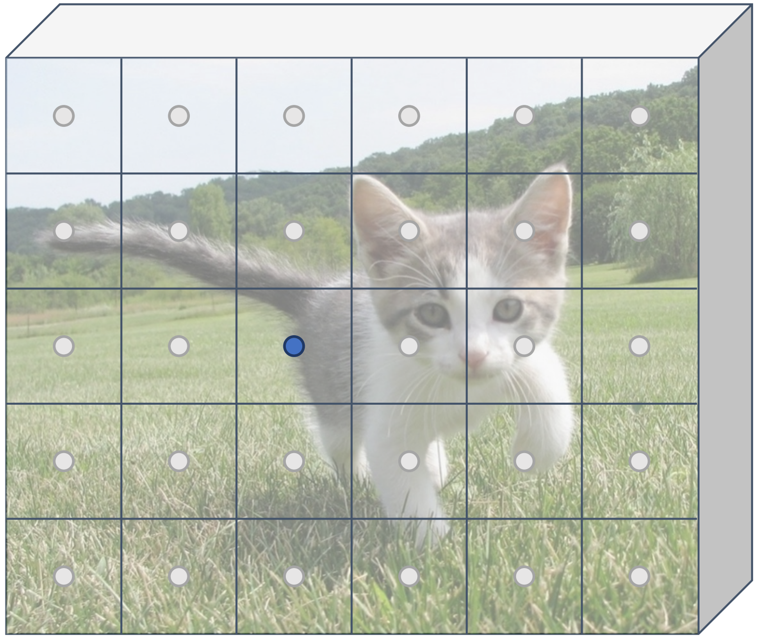
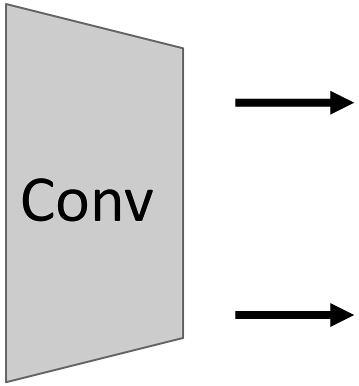


Image features
(e.g. 512 x 5 x 6)



Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

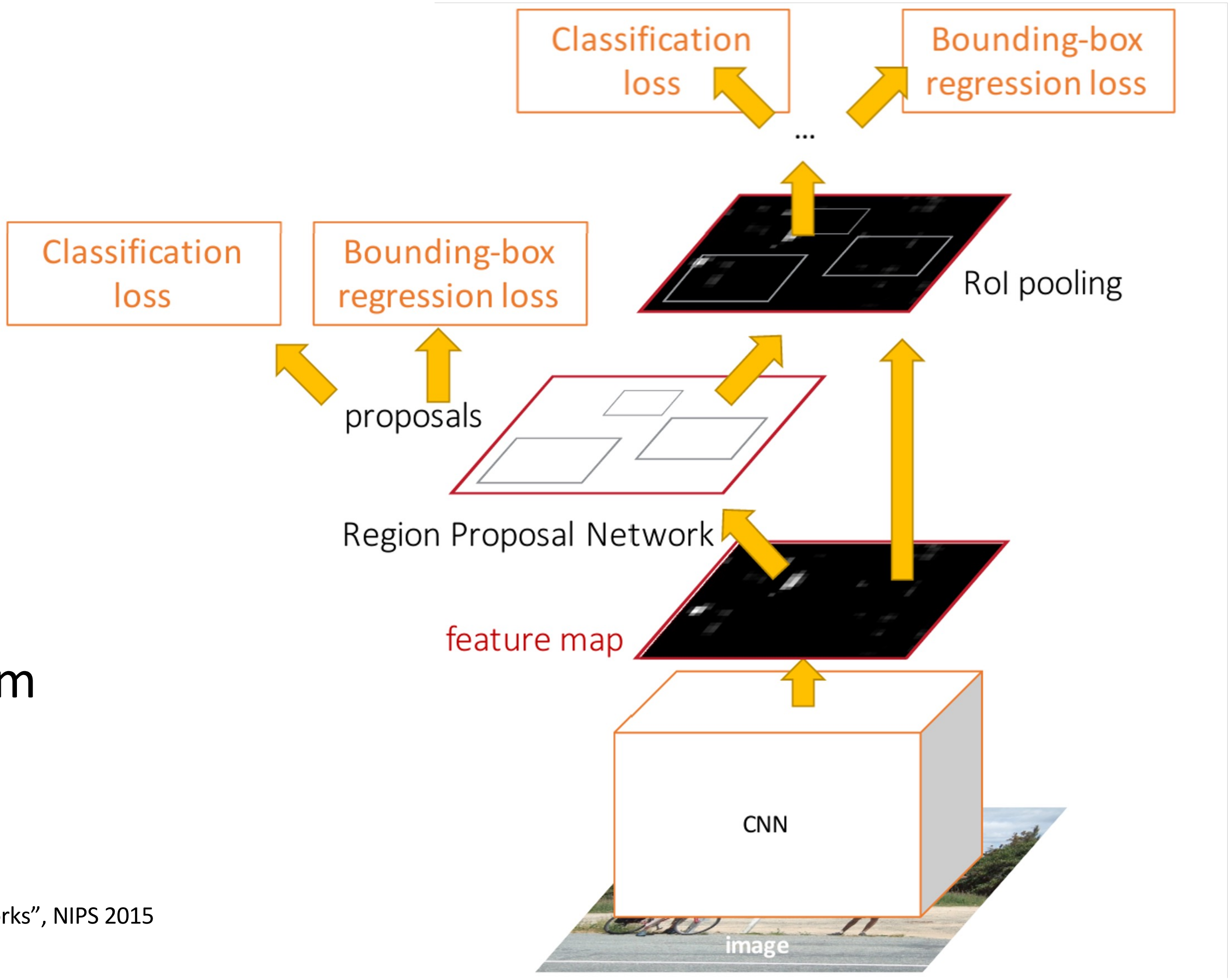
At test-time, sort all K*5*6 boxes by their positive score, take top 300 as our region proposals

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

Faster R-CNN: Learnable Region Proposals

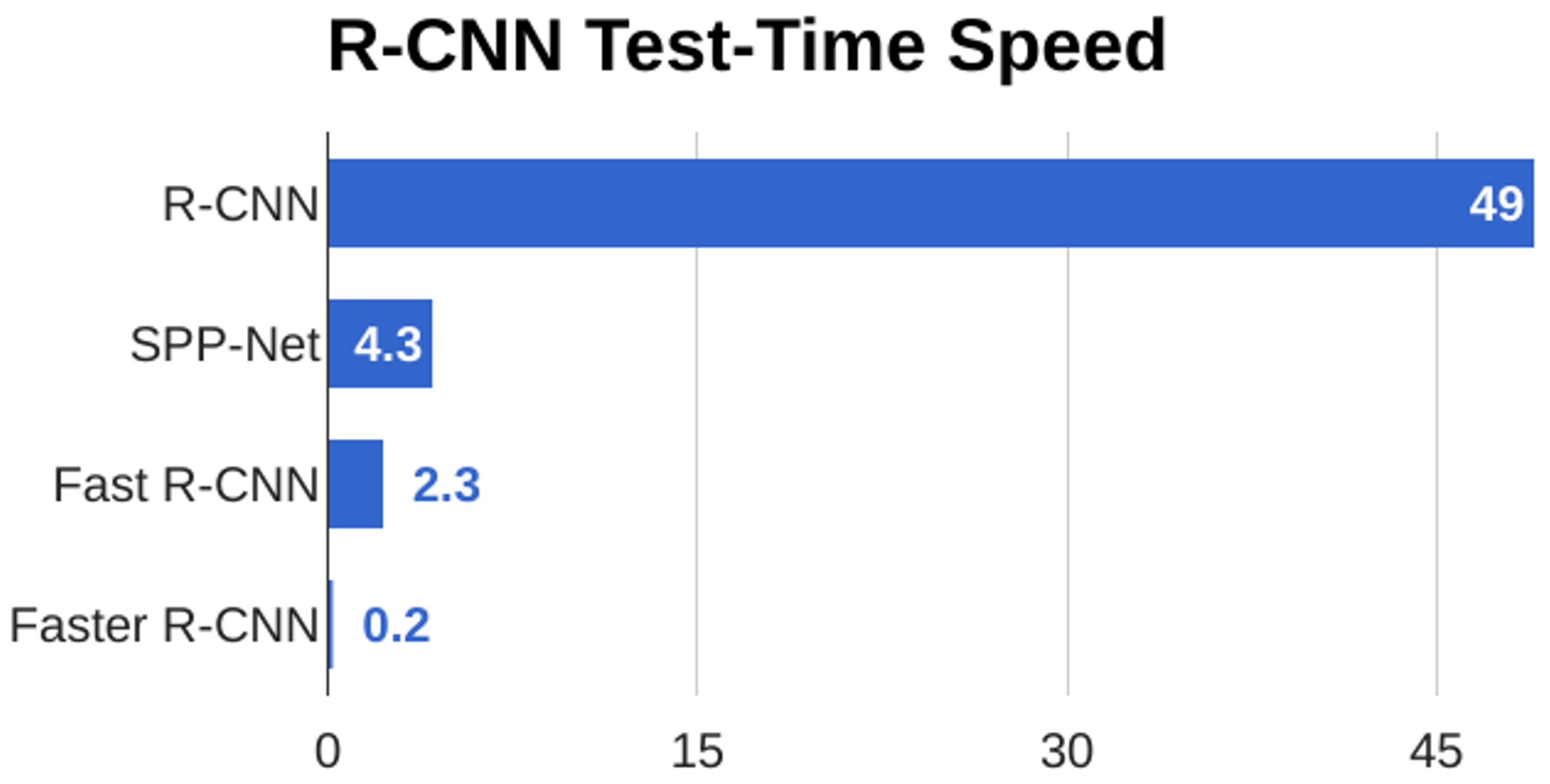
Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



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: Learnable Region Proposals

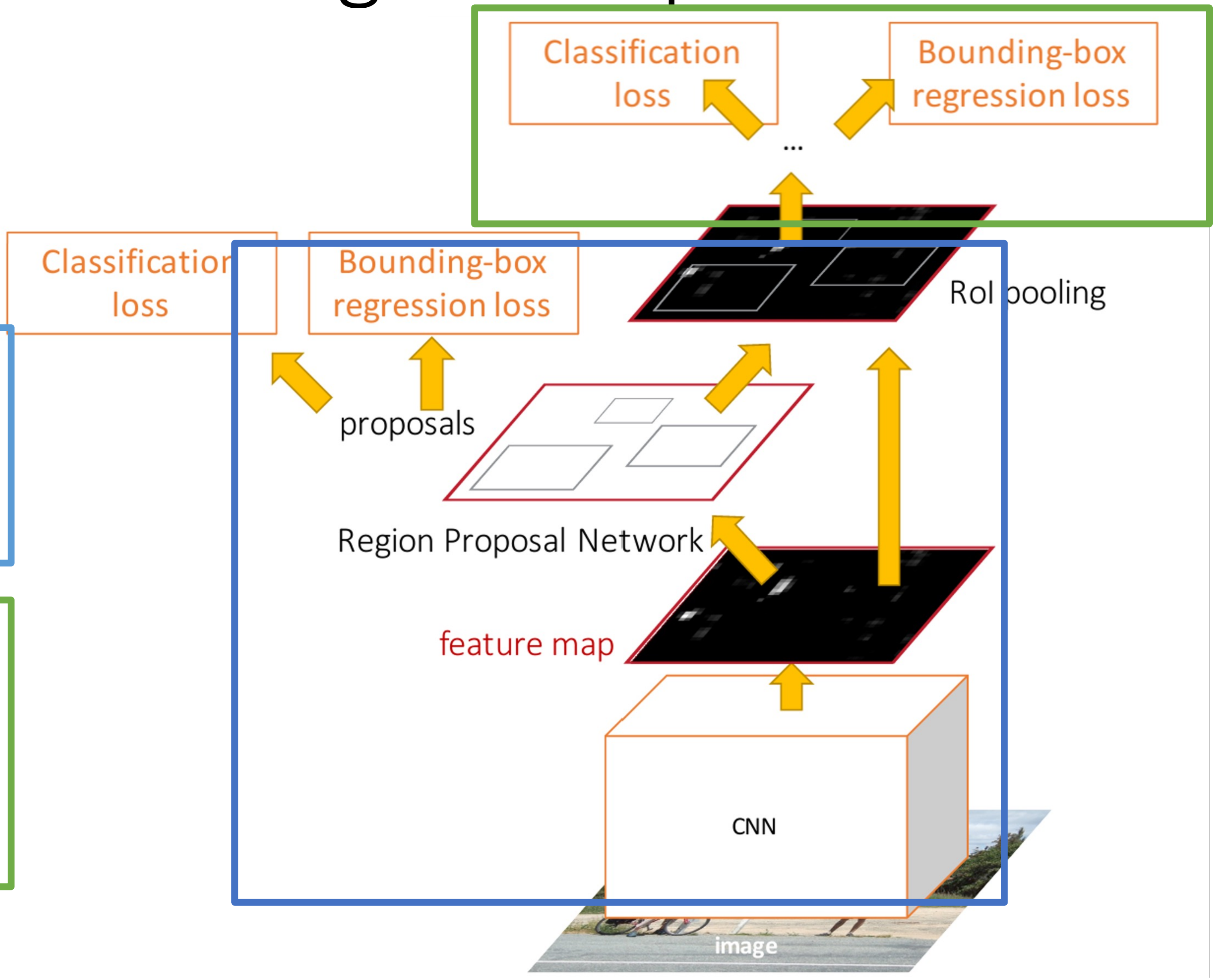


Faster R-CNN: Learnable Region Proposals

Faster R-CNN is a
Two-stage object detector

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

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



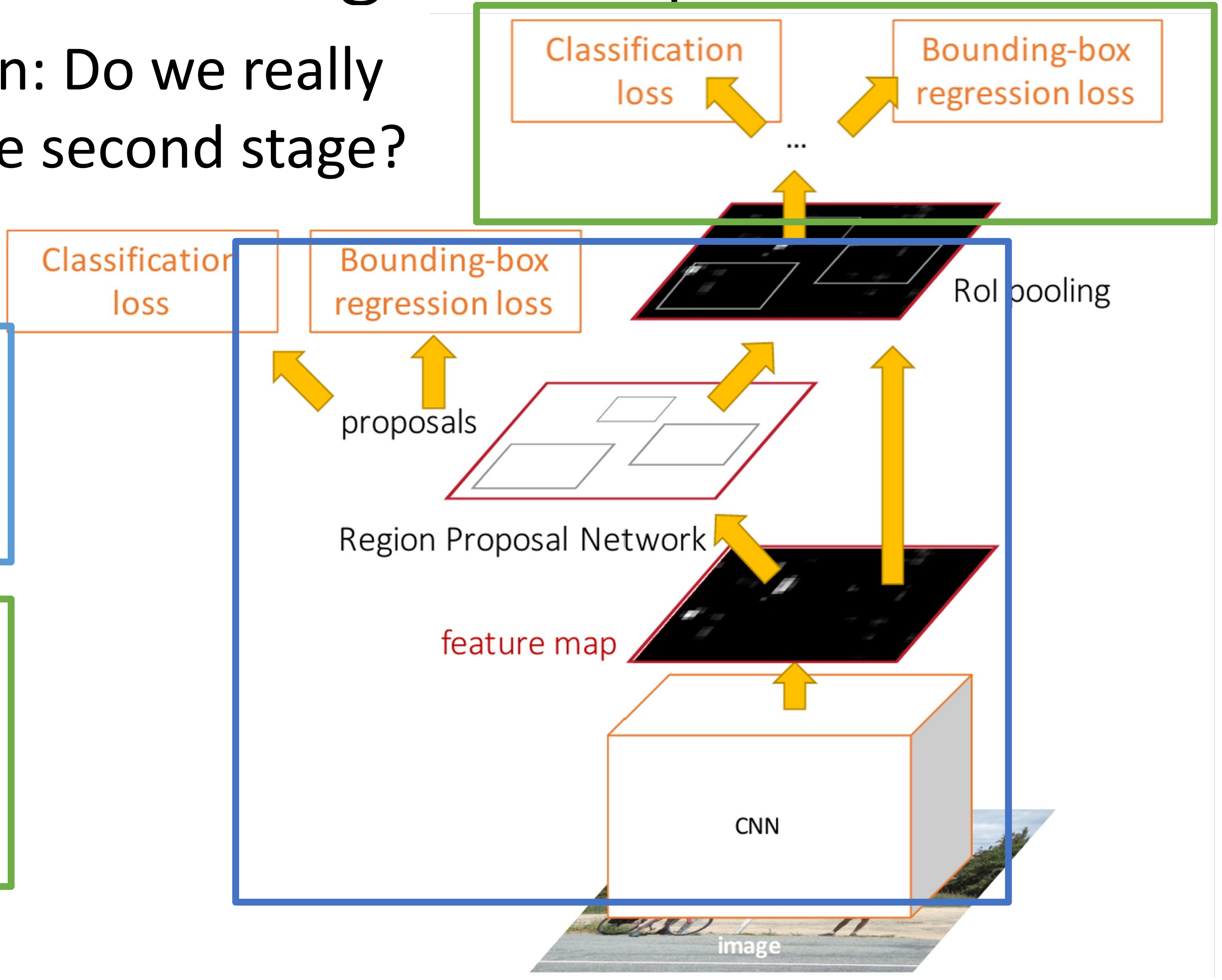
Fast<u>er</u> R-CNN: Learnable Region Proposals

Question: Do we really need the second stage?

Faster R-CNN is a
Two-stage object detector

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

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

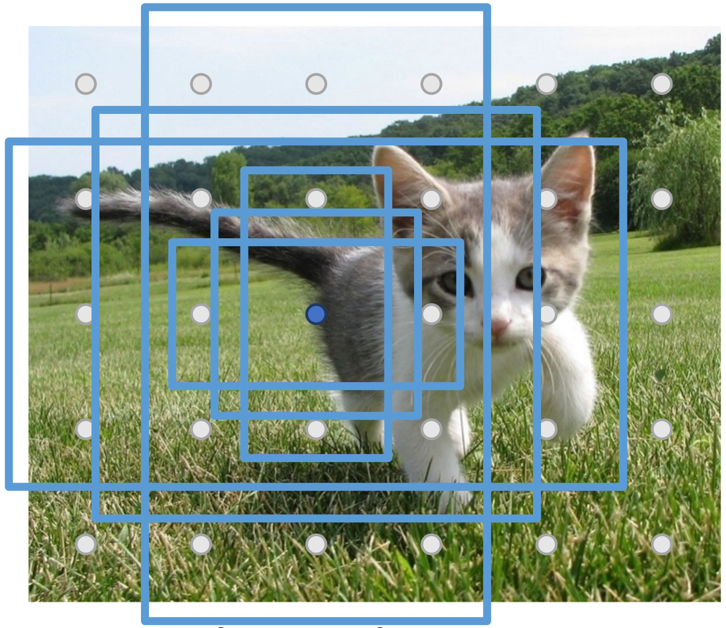


Slide from Justin Johnson

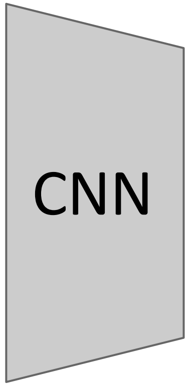
Single-Stage Detectors: RetinaNet

Similar to RPN – but rather than classify anchors as object/no object, directly predict object category (among C categories) or background

Run backbone CNN to get features aligned to input image



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



Each feature corresponds to a point in the input

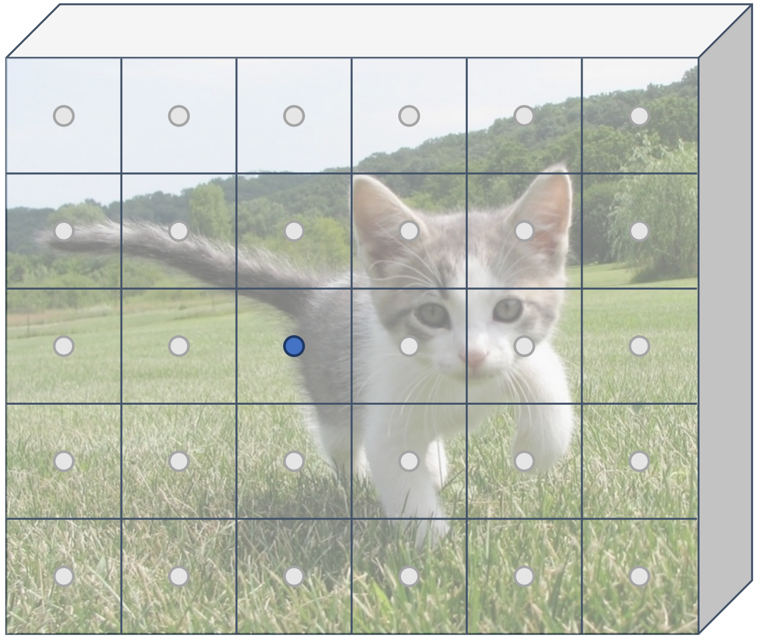
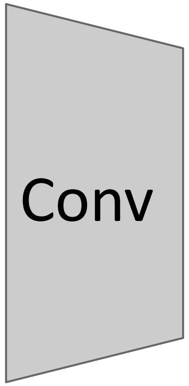


Image features
(e.g. 512 x 5 x 6)



→ Anchor classification
2K*(C+1) x 5 x 6

→ Anchor transforms
4K x 5 x 6

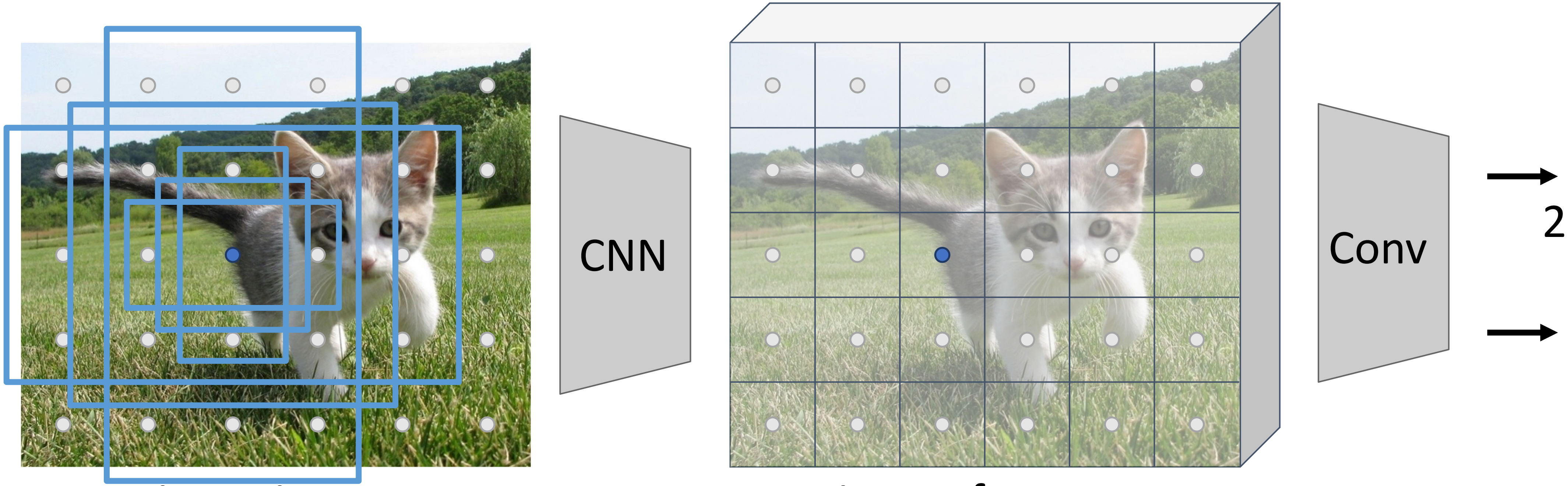
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Single-Stage Detectors: RetinaNet

Problem: class imbalance – many more background anchors vs non-background

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

Anchor classification
2K*(C+1) x 5 x 6

Anchor transforms
4K x 5 x 6

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Slide from Justin Johnson

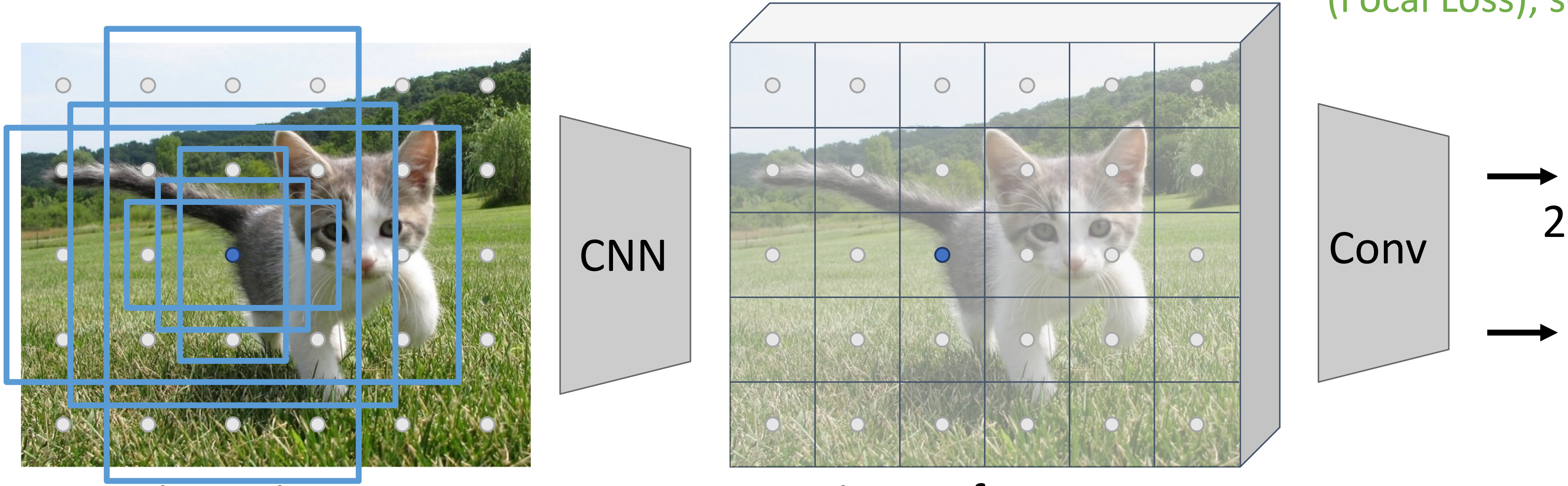
Single-Stage Detectors: RetinaNet

Problem: class imbalance – many more background anchors vs non-background

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

Solution: new loss function (Focal Loss); see paper



Anchor classification 2K*(C+1) x 5 x 6

Anchor transforms 4K x 5 x 6

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

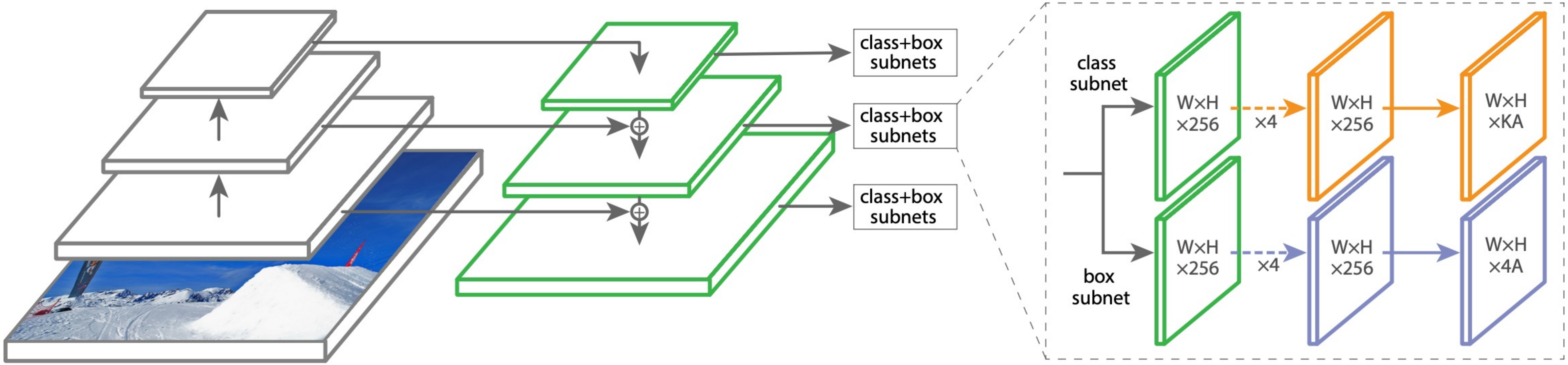
Image features (e.g. 512 x 5 x 6)

$$\mathrm{CE}(p_t) = -\log(p_t)$$
$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$$

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Single-Stage Detectors: RetinaNet

In practice, RetinaNet also uses Feature Pyramid Network to handle multiscale



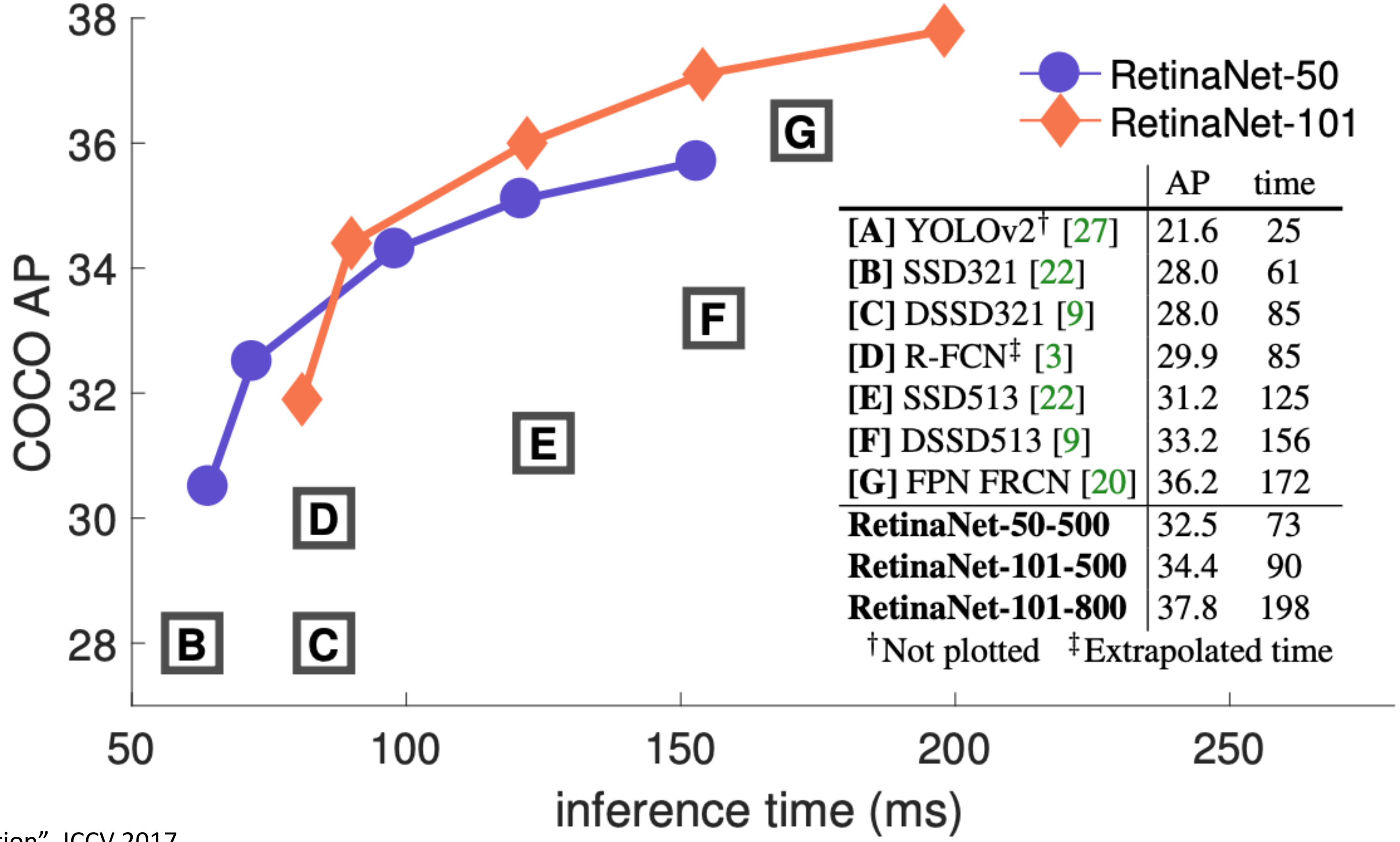
(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Figure credit: Lin et al, ICCV 2017

Single-Stage Detectors: RetinaNet

Single-Stage detectors can be much faster than two-stage detectors



	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

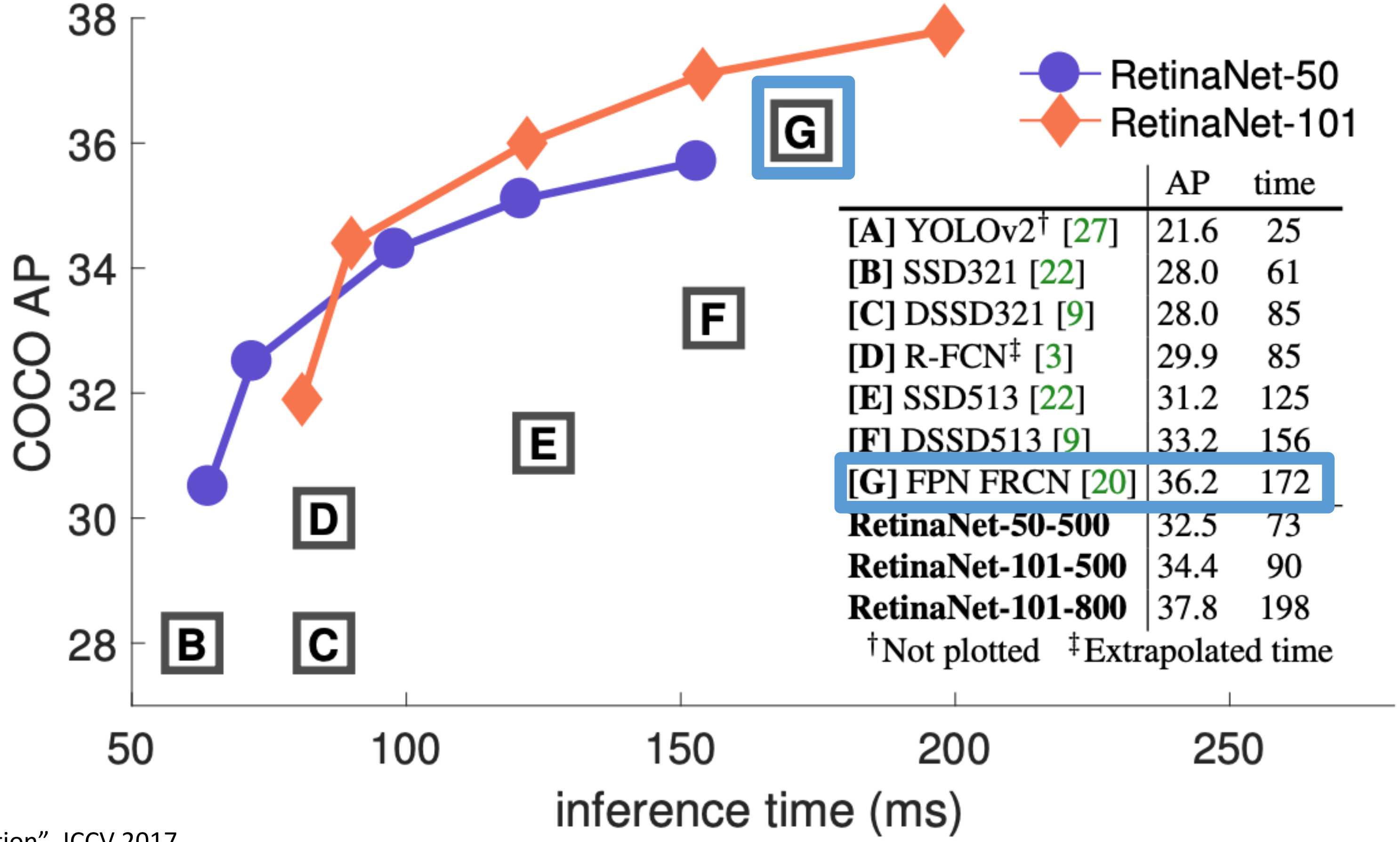
†Not plotted ‡Extrapolated time

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Figure credit: Lin et al, ICCV 2017

Single-Stage Detectors: RetinaNet

Single-Stage detectors can be much faster than two-stage detectors



	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Faster R-CNN with Feature Pyramid Network

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Figure credit: Lin et al, ICCV 2017

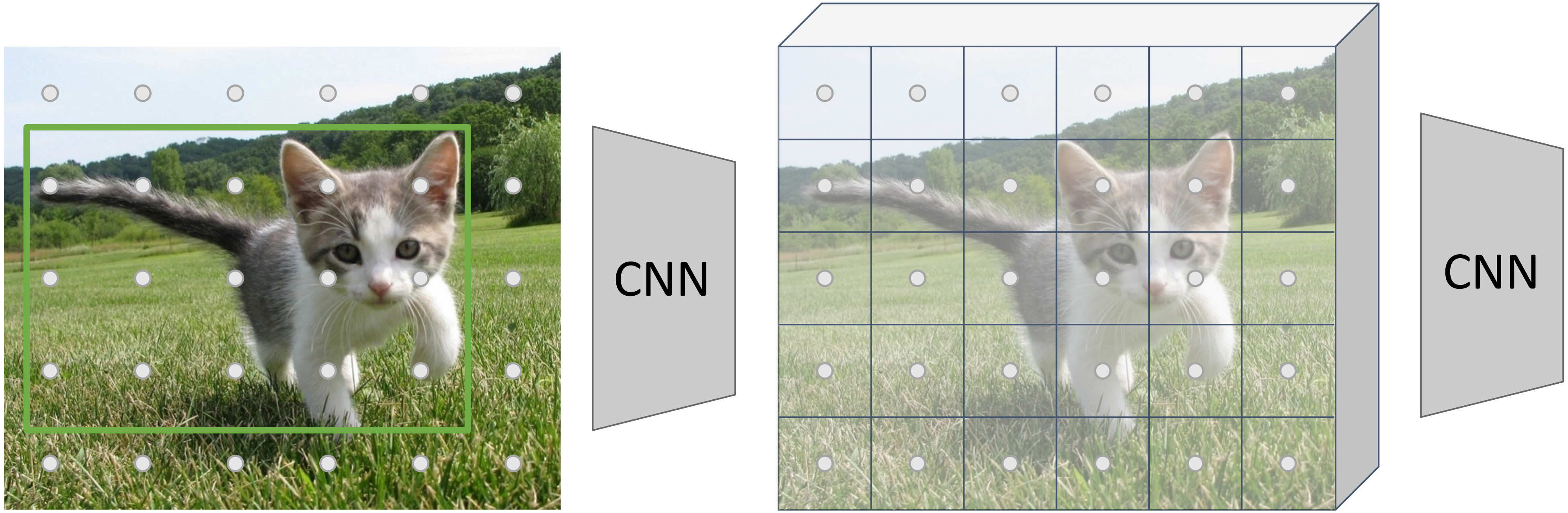
Slide from Justin Johnson

Single-Stage Detectors: FCOS

"Anchor-free" detector

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

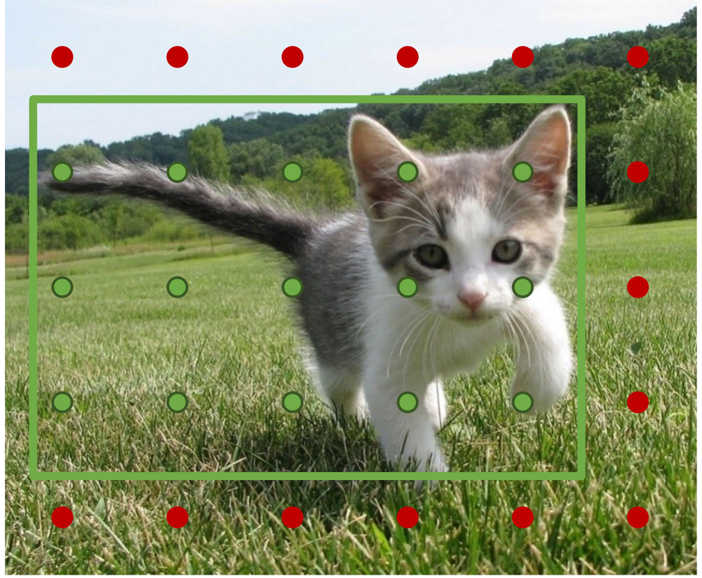
Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Slide from Justin Johnson

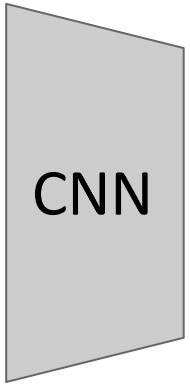
Single-Stage Detectors: FCOS

“Anchor-free” detector

Run backbone CNN to get features aligned to input image



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



Each feature corresponds to a point in the input

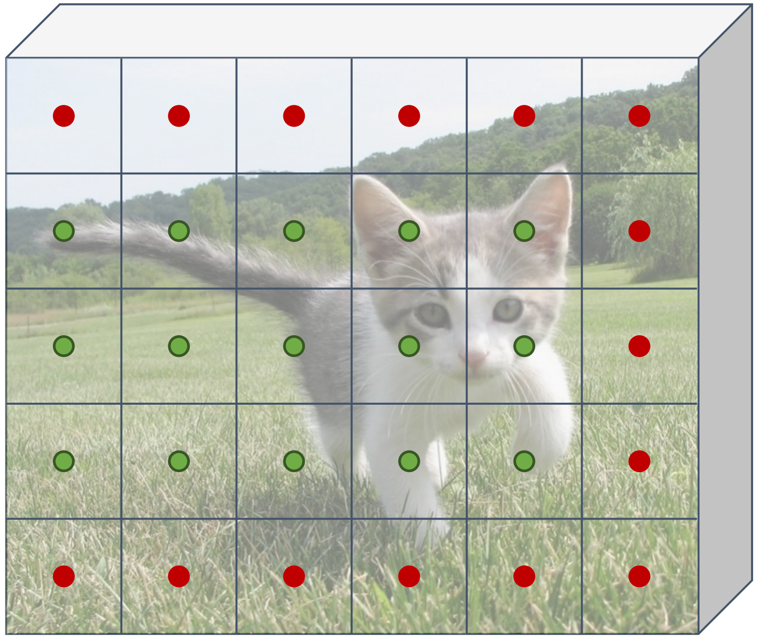
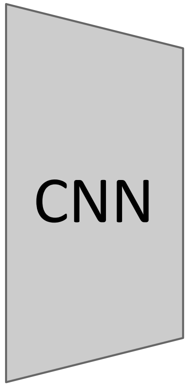


Image features
(e.g. 512 x 5 x 6)

Classify points as positive if they fall into a GT box, or negative if they don’t

Train independent per-category logistic regressors



→ Class scores
C x 5 x 6

Tian et al, “FCOS: Fully Convolutional One-Stage Object Detection”, ICCV 2019

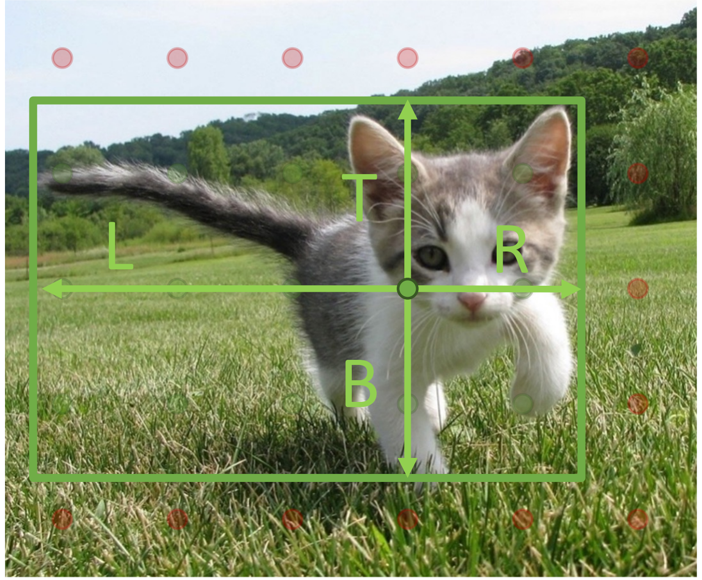
Single-Stage Detectors: FCOS

"Anchor-free" detector

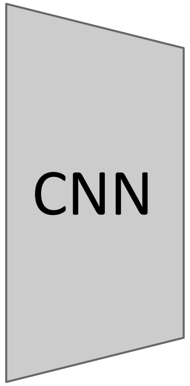
Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

For positive points, also regress distance to left, right, top, and bottom of ground-truth box (with L2 loss)



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



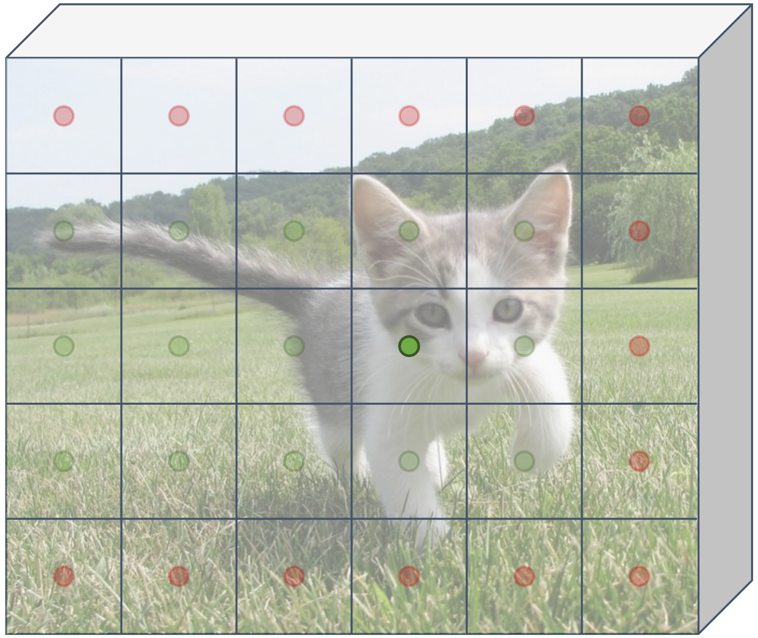
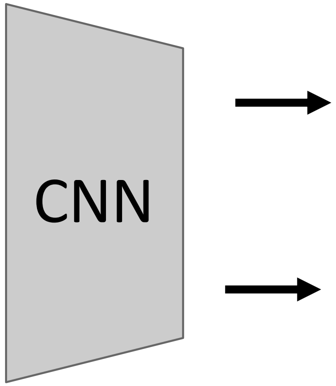


Image features
(e.g. 512 x 5 x 6)



Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

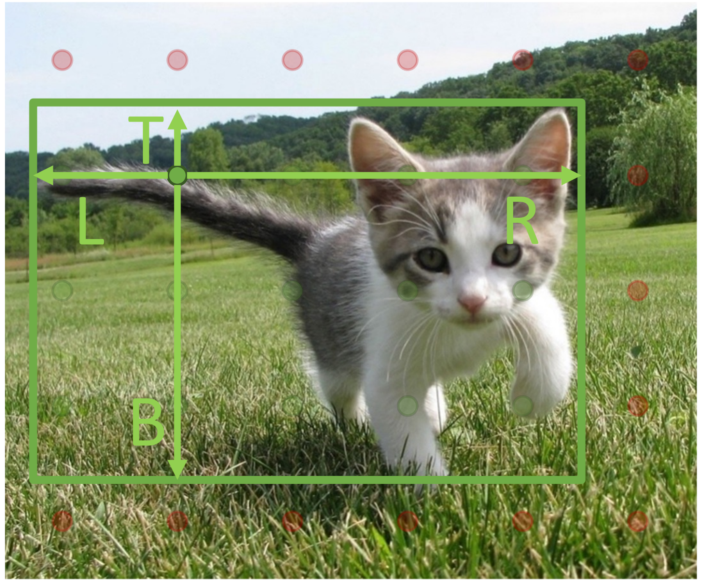
Single-Stage Detectors: FCOS

"Anchor-free" detector

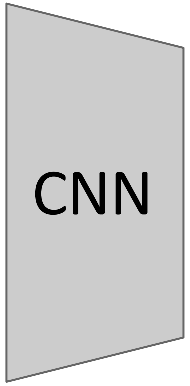
Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

For positive points, also regress distance to left, right, top, and bottom of ground-truth box (with L2 loss)



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



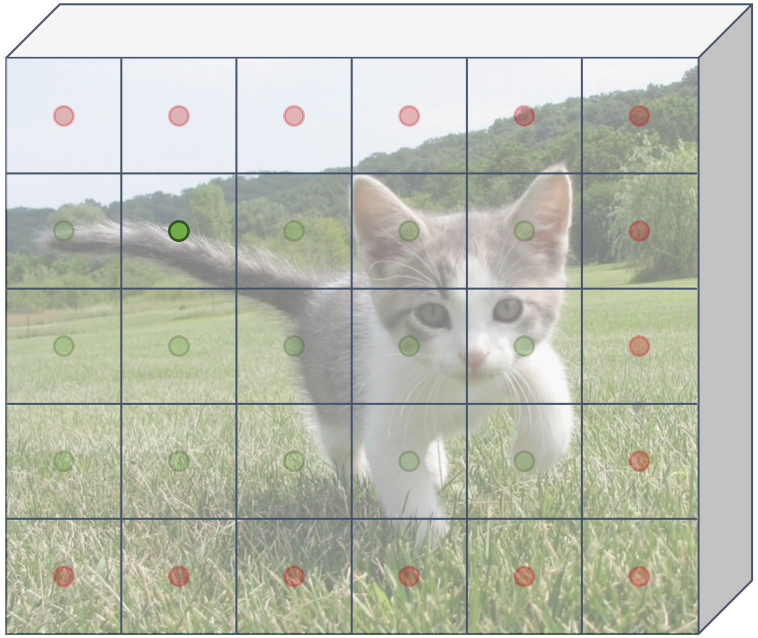
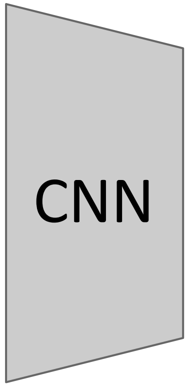


Image features
(e.g. 512 x 5 x 6)



→ Class scores
C x 5 x 6

→ Box edges
4 x 5 x 6

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

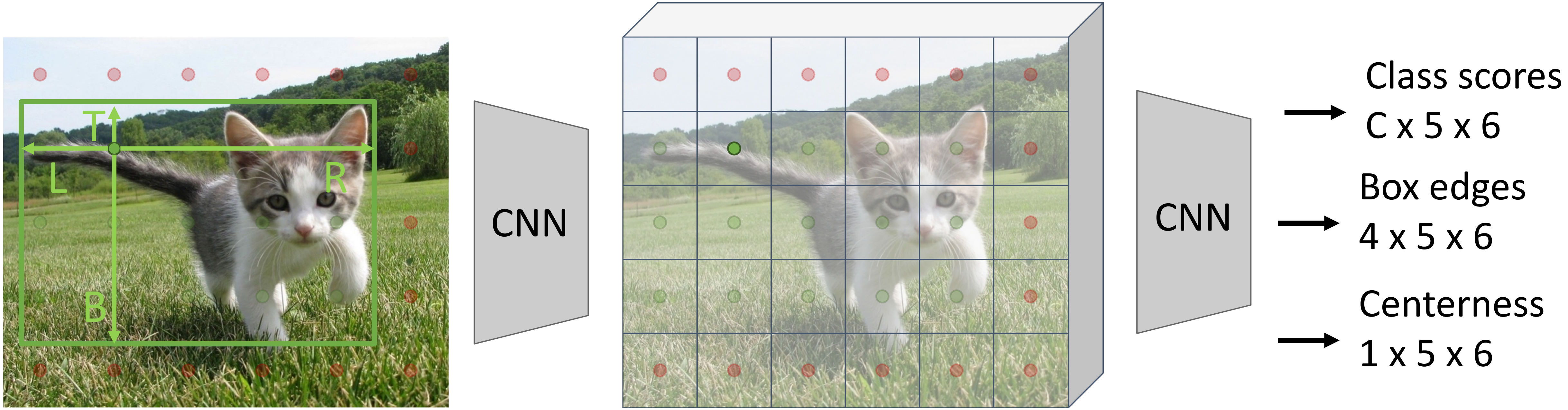
Single-Stage Detectors: FCOS

"Anchor-free" detector

Finally, predict "centerness" for all positive points (using logistic regression loss)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

Image features
(e.g. 512 x 5 x 6)

$$centerness = \sqrt{\frac{\min(L,R)}{\max(L,R)} \cdot \frac{\min(T,B)}{\max(T,B)}}$$

Ranges from 1 at box center to 0 at box edge

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

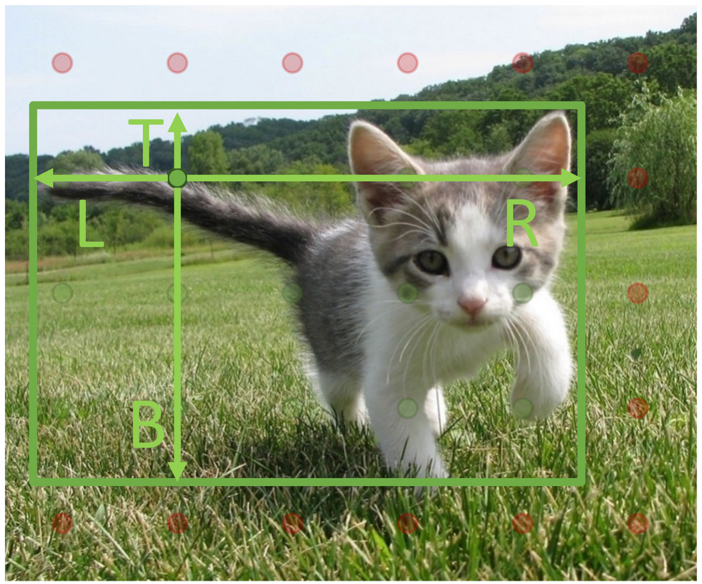
Single-Stage Detectors: FCOS

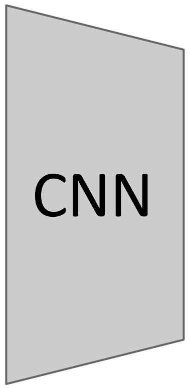
"Anchor-free" detector

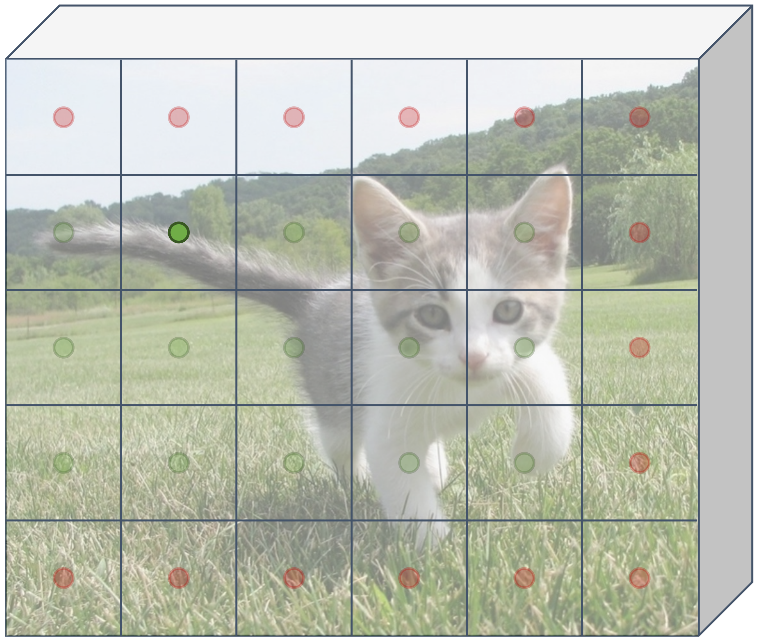
Test-time: predicted "confidence" for the box from each point is product of its class score and centerness

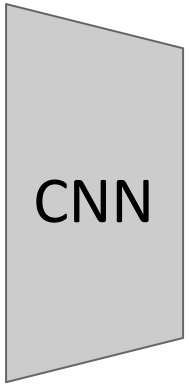
Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input









Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Centerness
1 x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

$$centerness = \sqrt{\frac{\min(L,R)}{\max(L,R)} \cdot \frac{\min(T,B)}{\max(T,B)}}$$

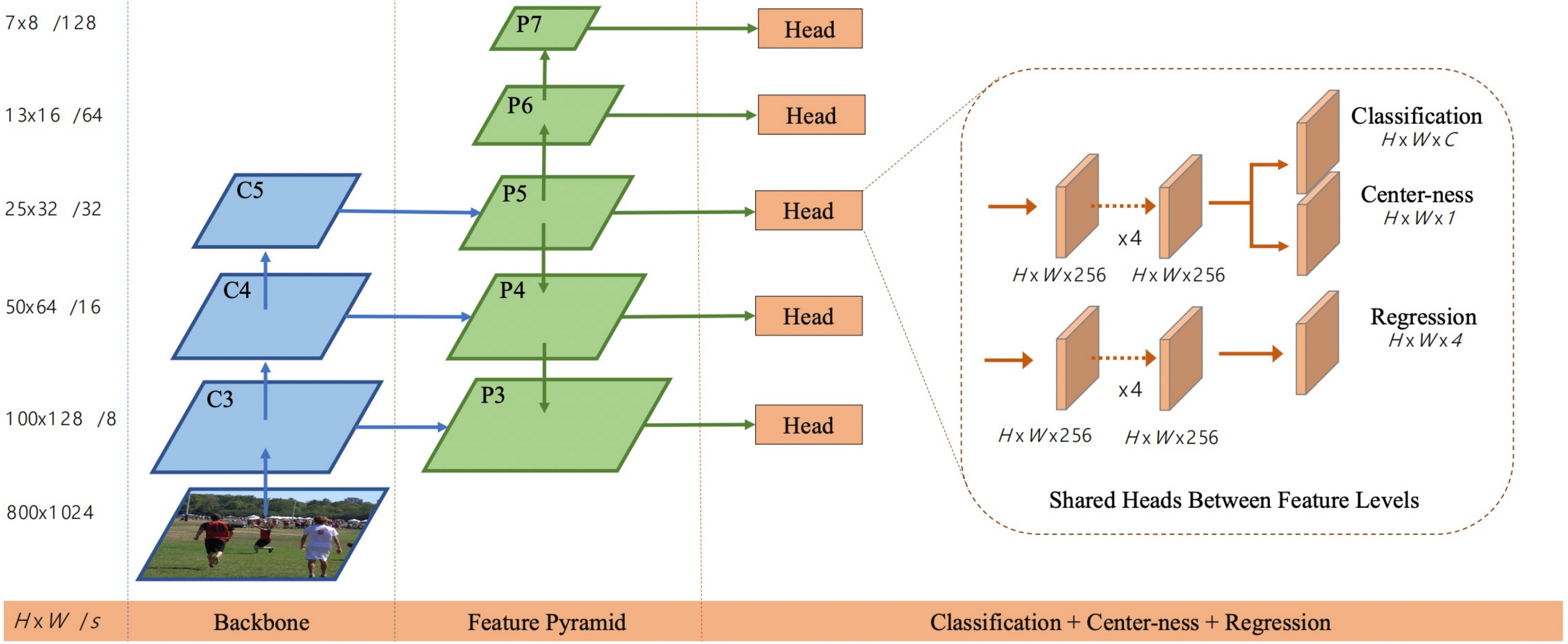
Ranges from 1 at box center to 0 at box edge

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

FCOS also uses a Feature Pyramid Network with heads shared across stages



Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

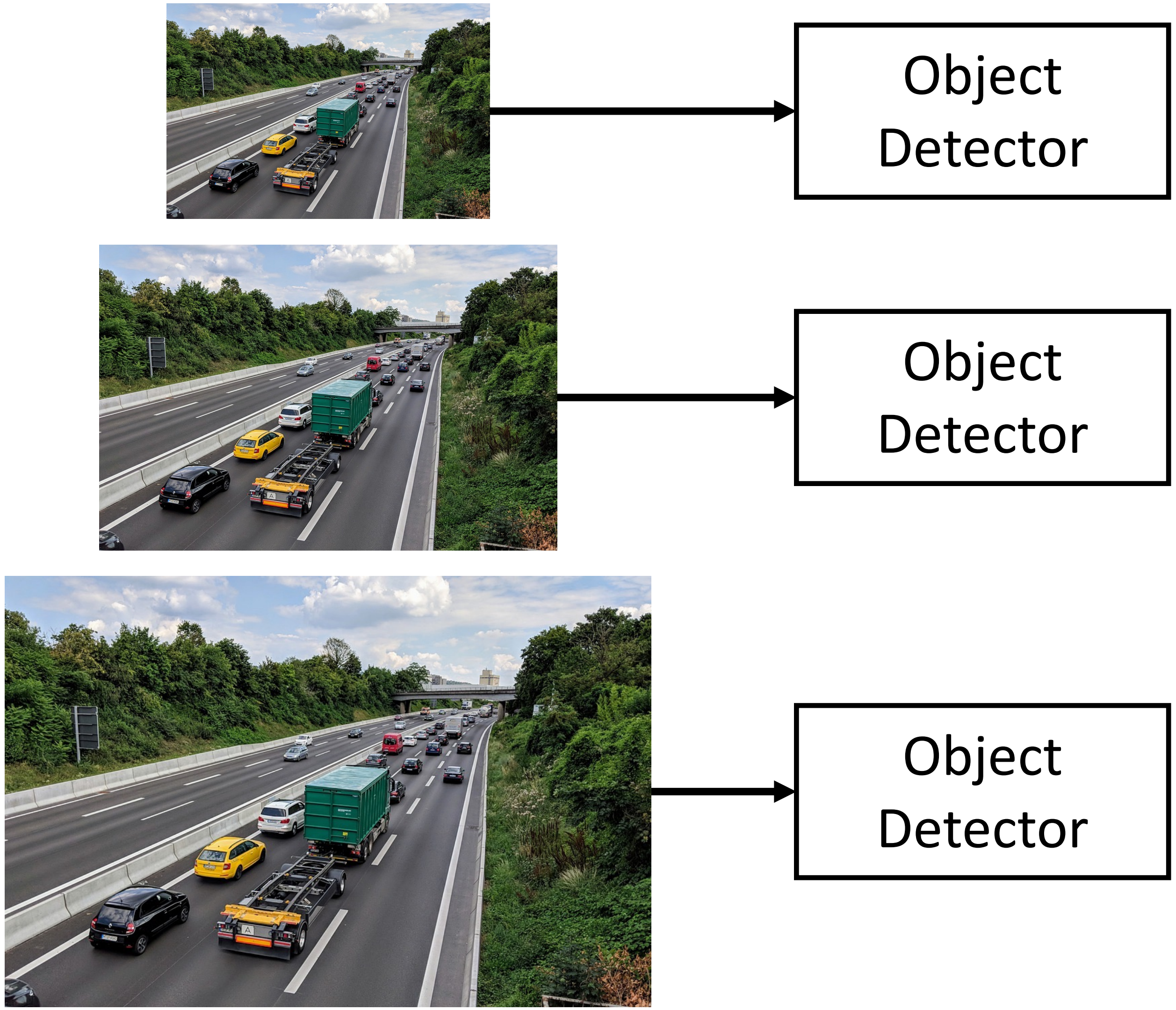
Dealing with Scale

We need to detect objects of many different scales.
How to improve *scale invariance* of the detector?

Slide from Justin Johnson

Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

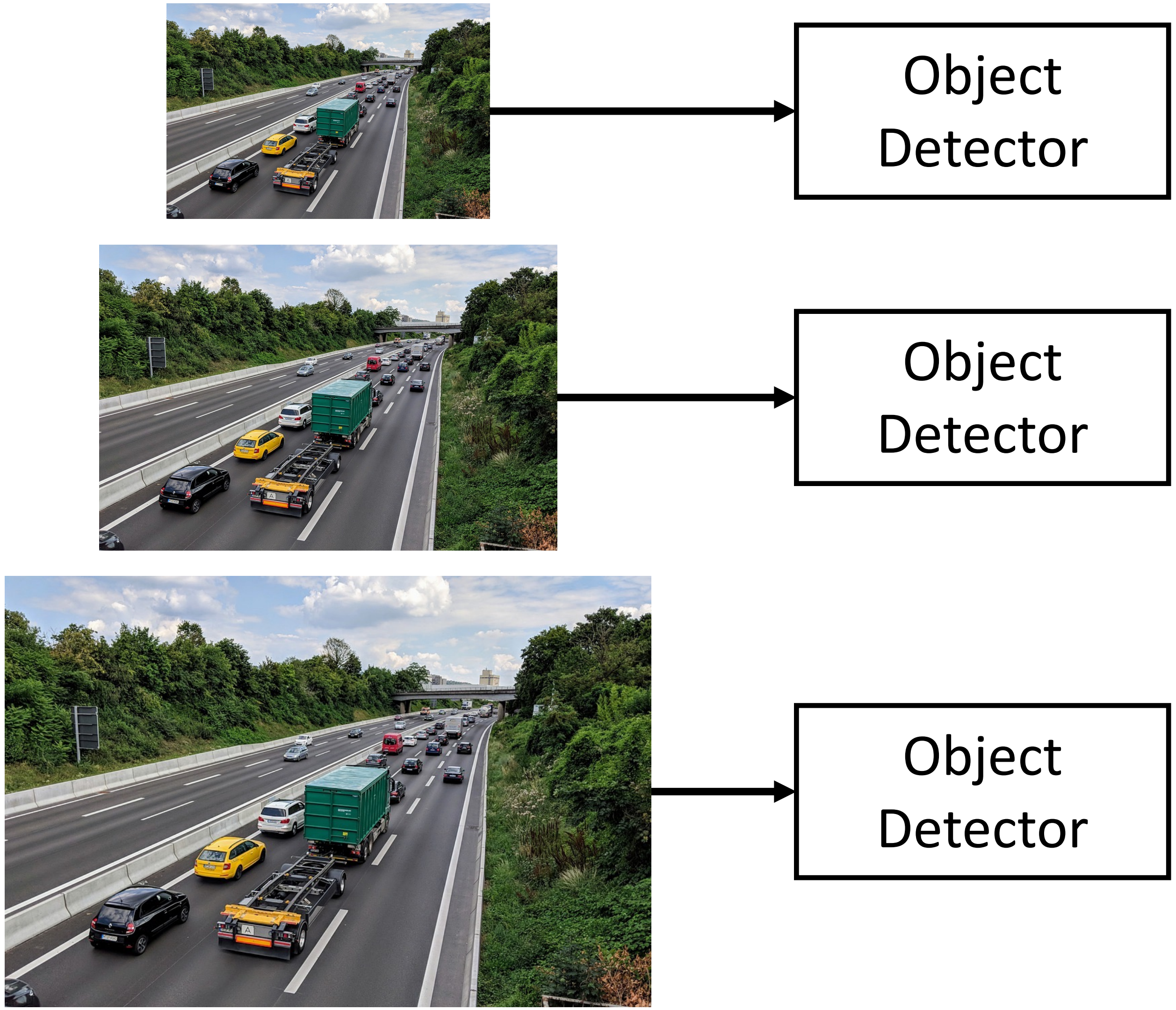


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

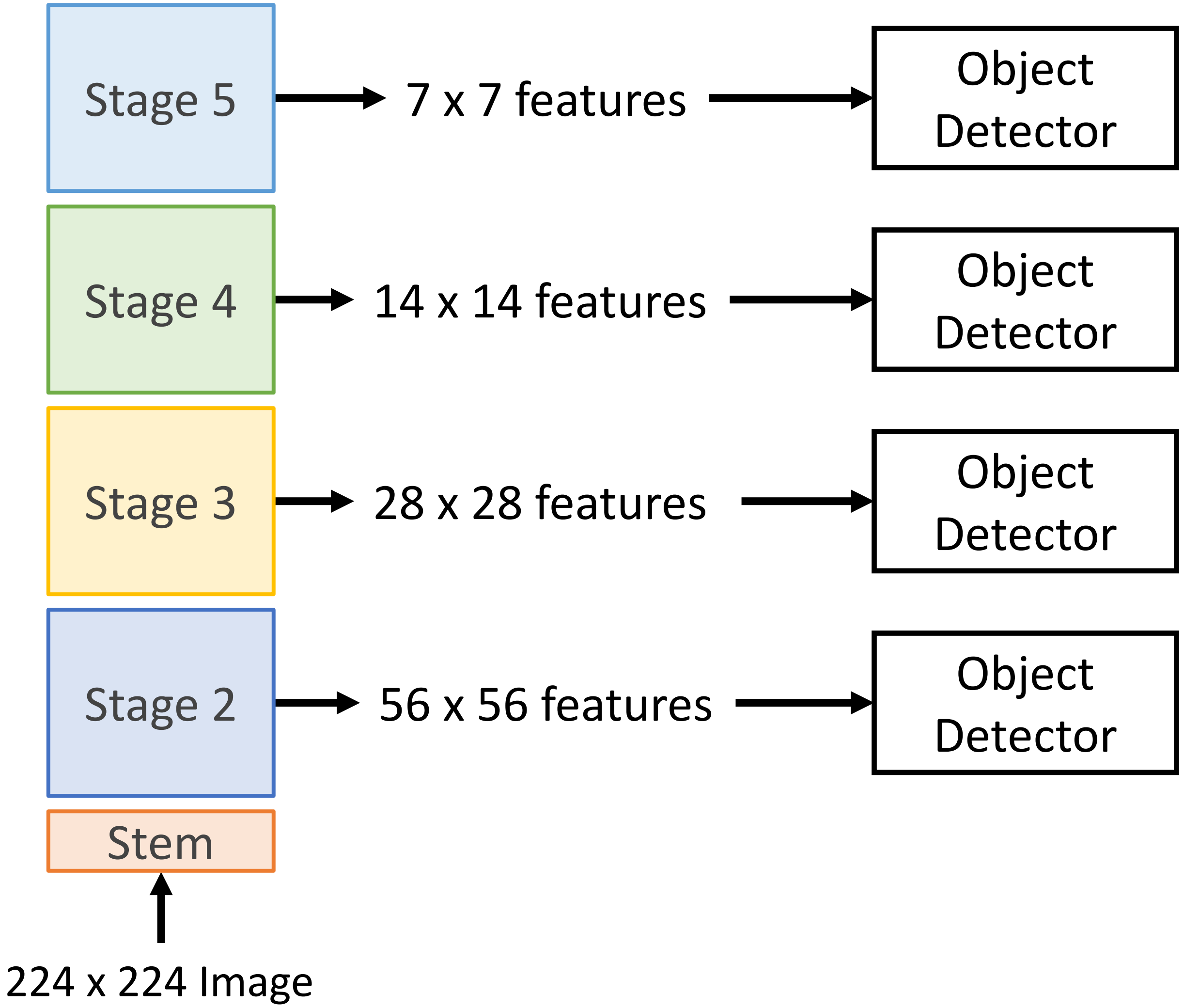
Problem: Expensive! Don’t share any computation between scales



Lin et al, “Feature Pyramid Networks for Object Detection”, ICCV 2017

Dealing with Scale: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level



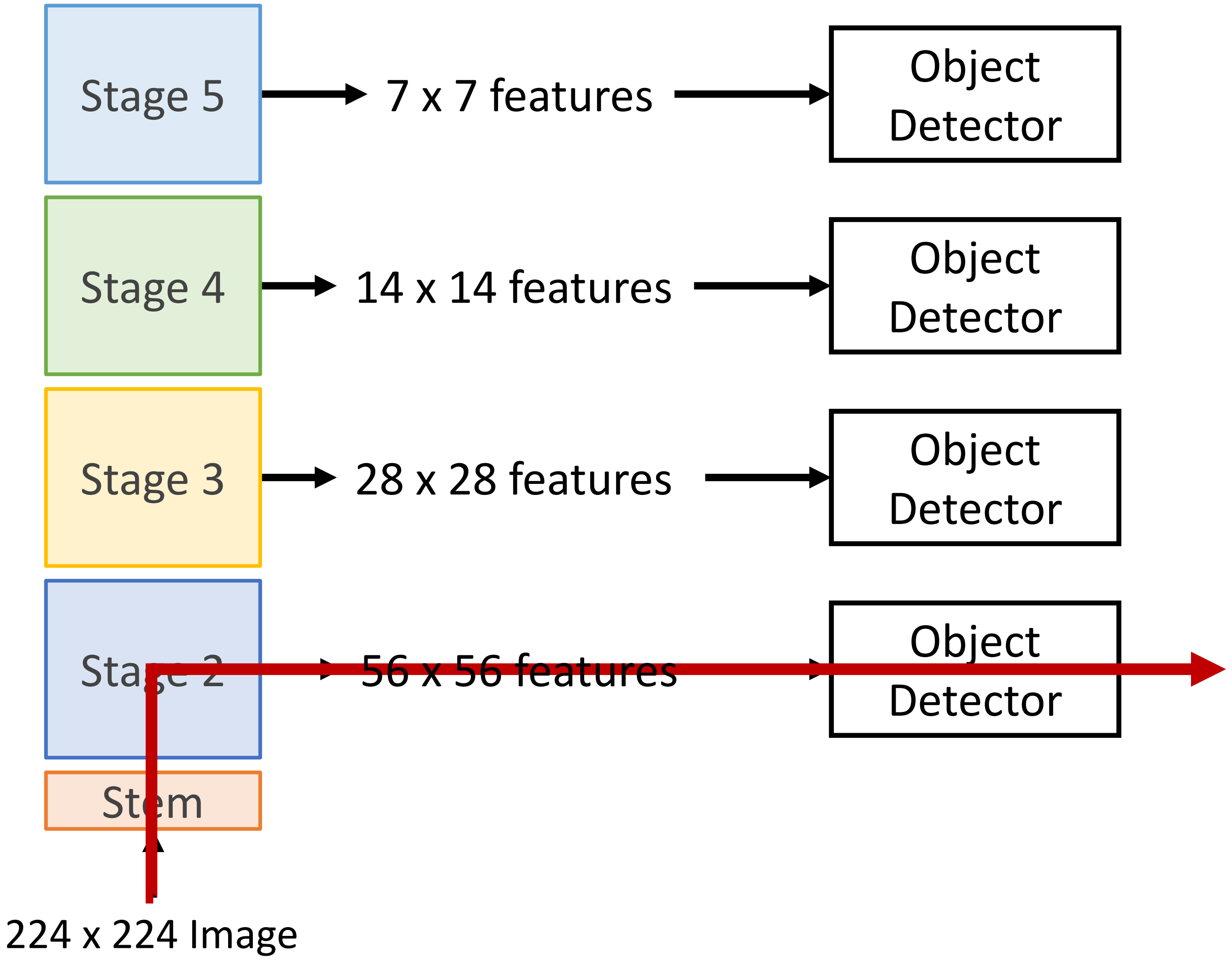
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Slide from Justin Johnson

Dealing with Scale: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

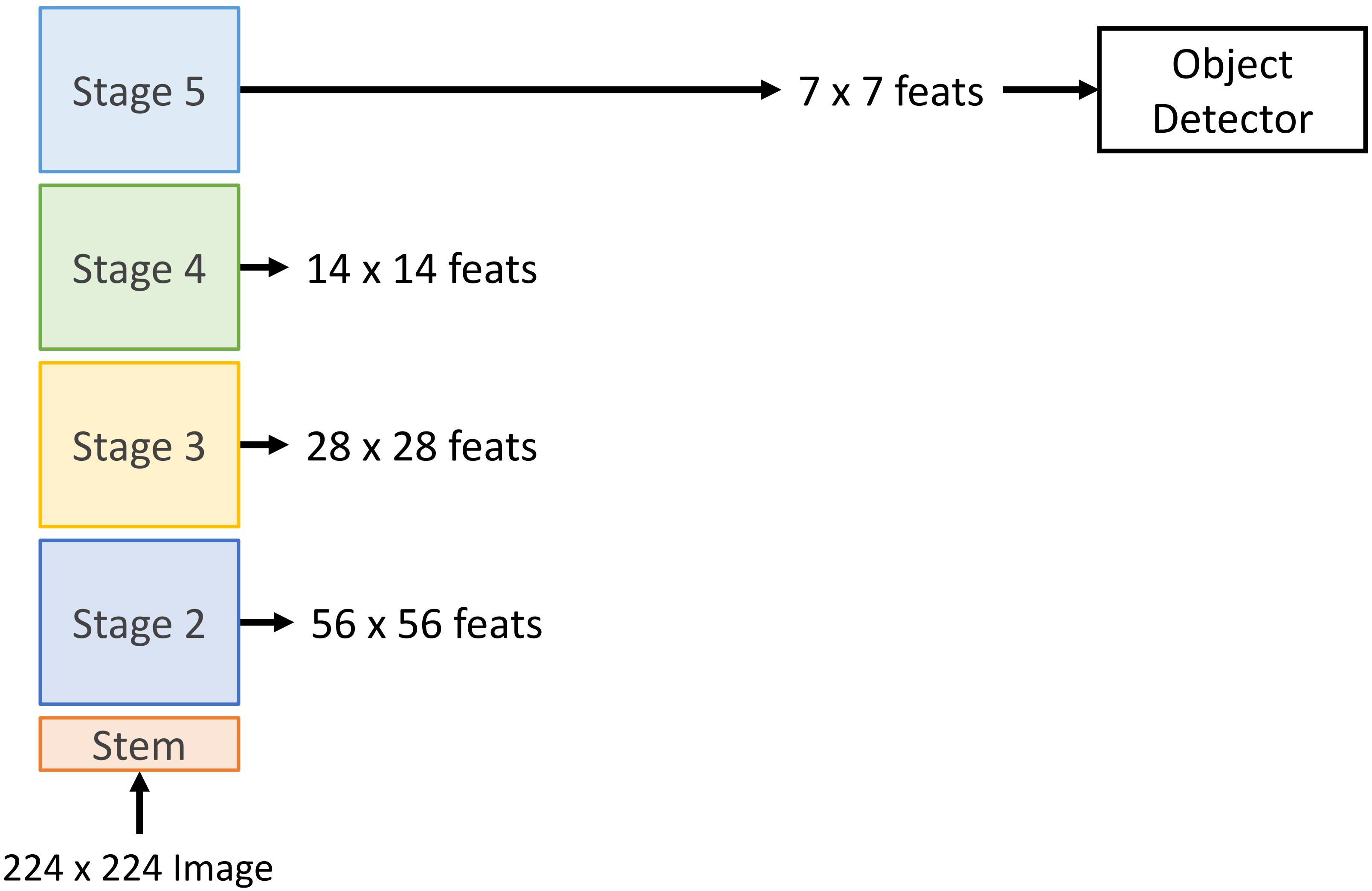
Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

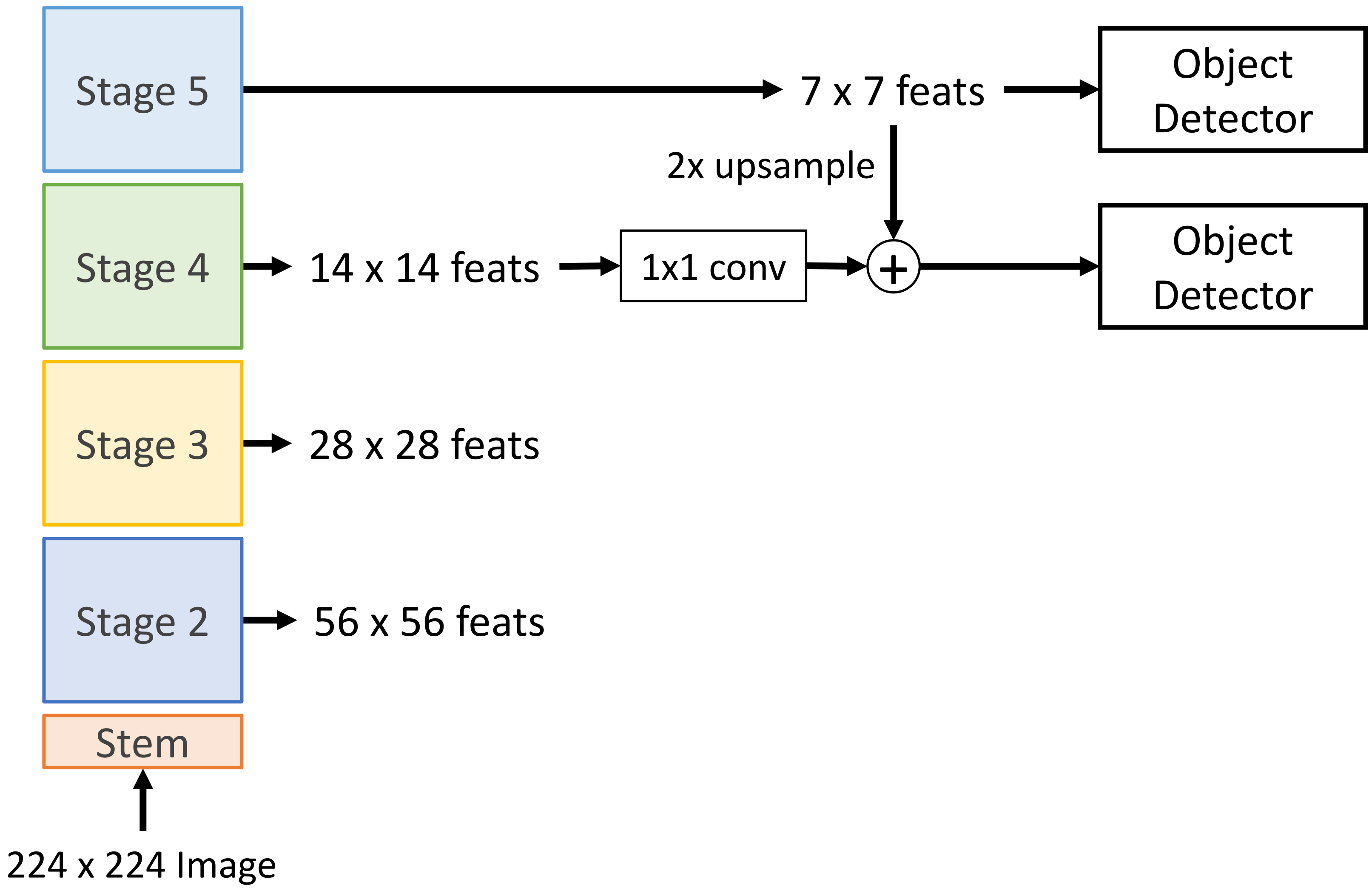
Add *top down connections* that feed information from high level features back down to lower level features



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

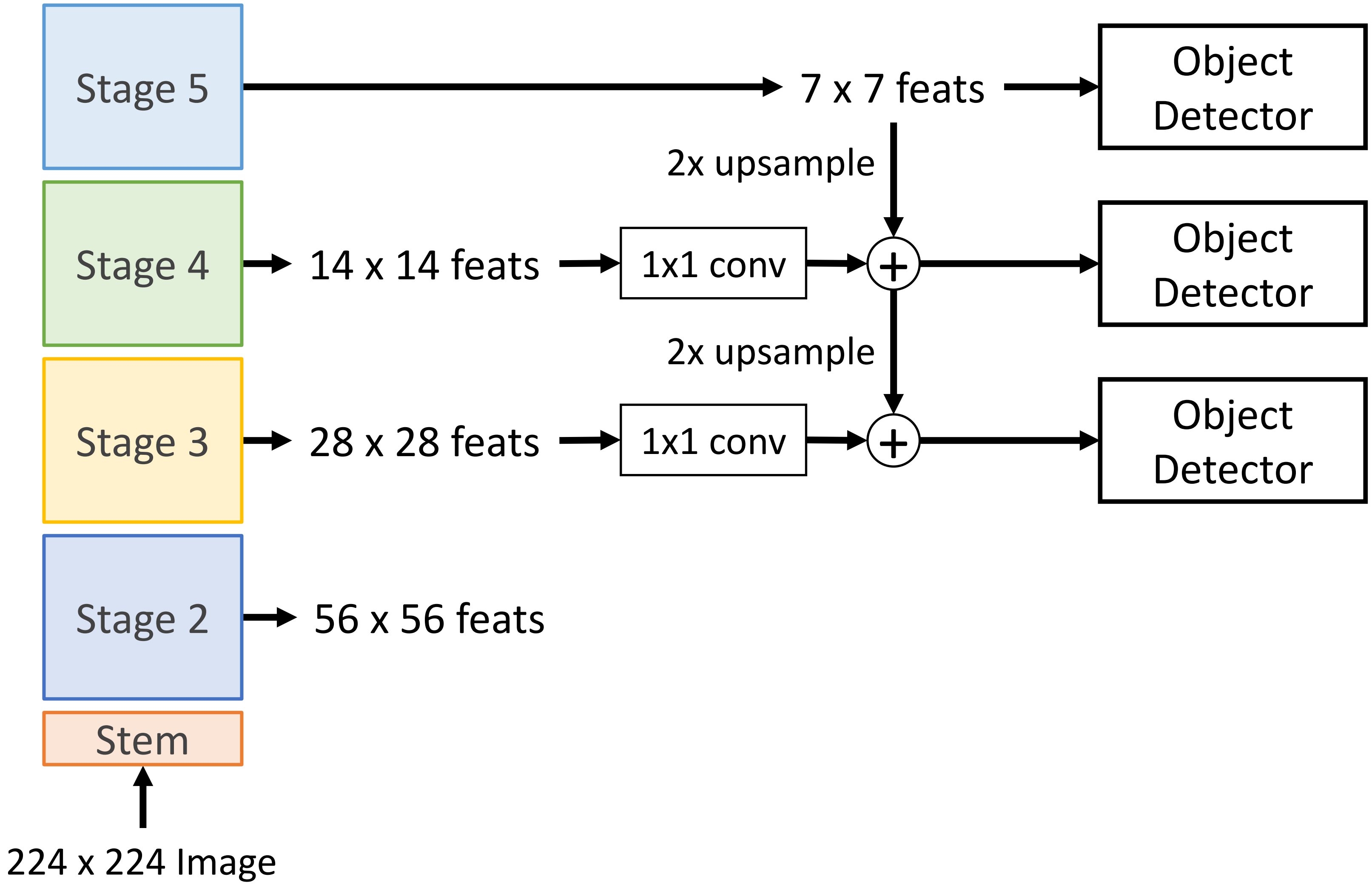


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Slide from Justin Johnson

Dealing with Scale: Feature Pyramid Network

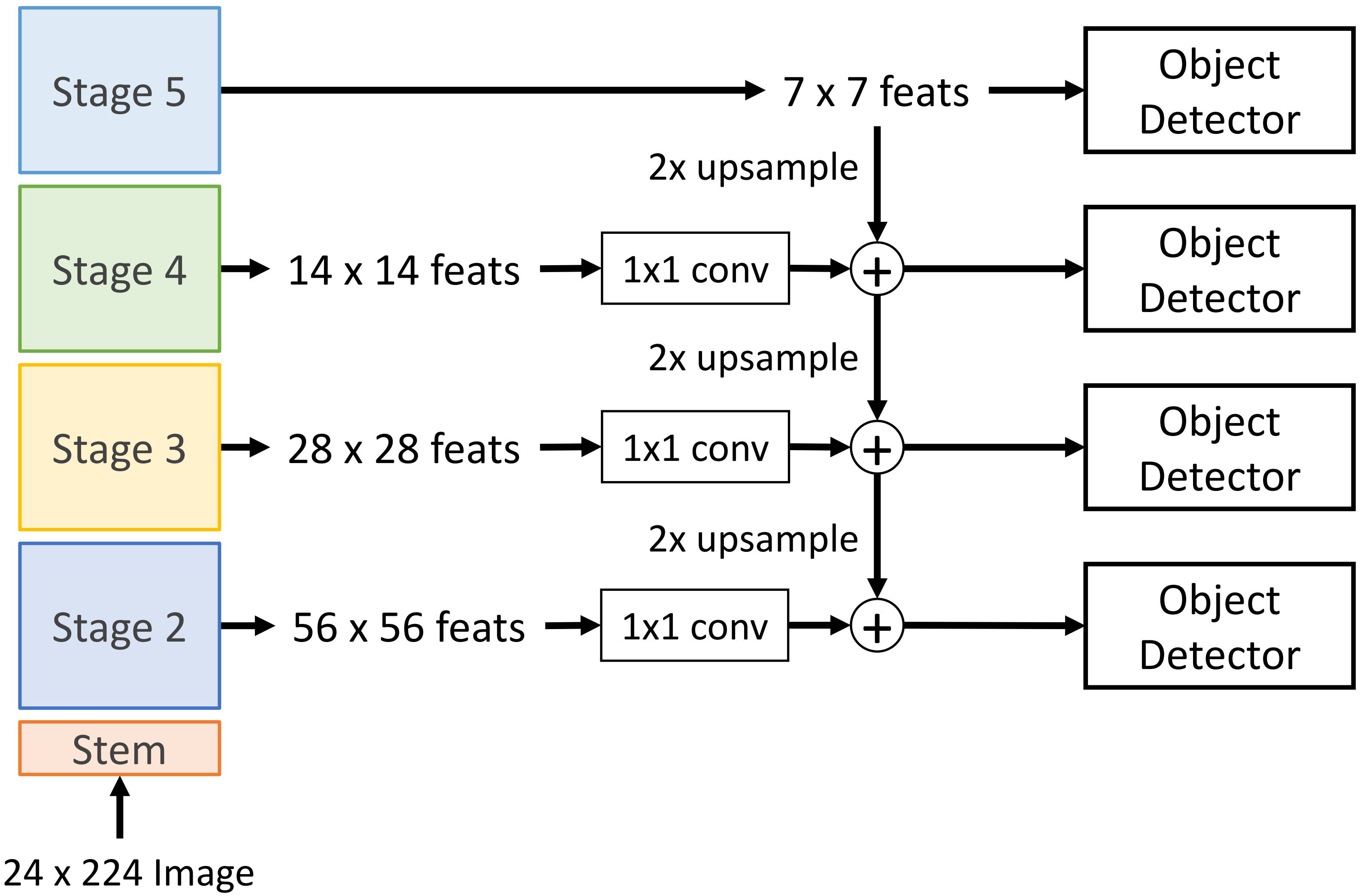
Add *top down connections* that feed information from high level features back down to lower level features



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features



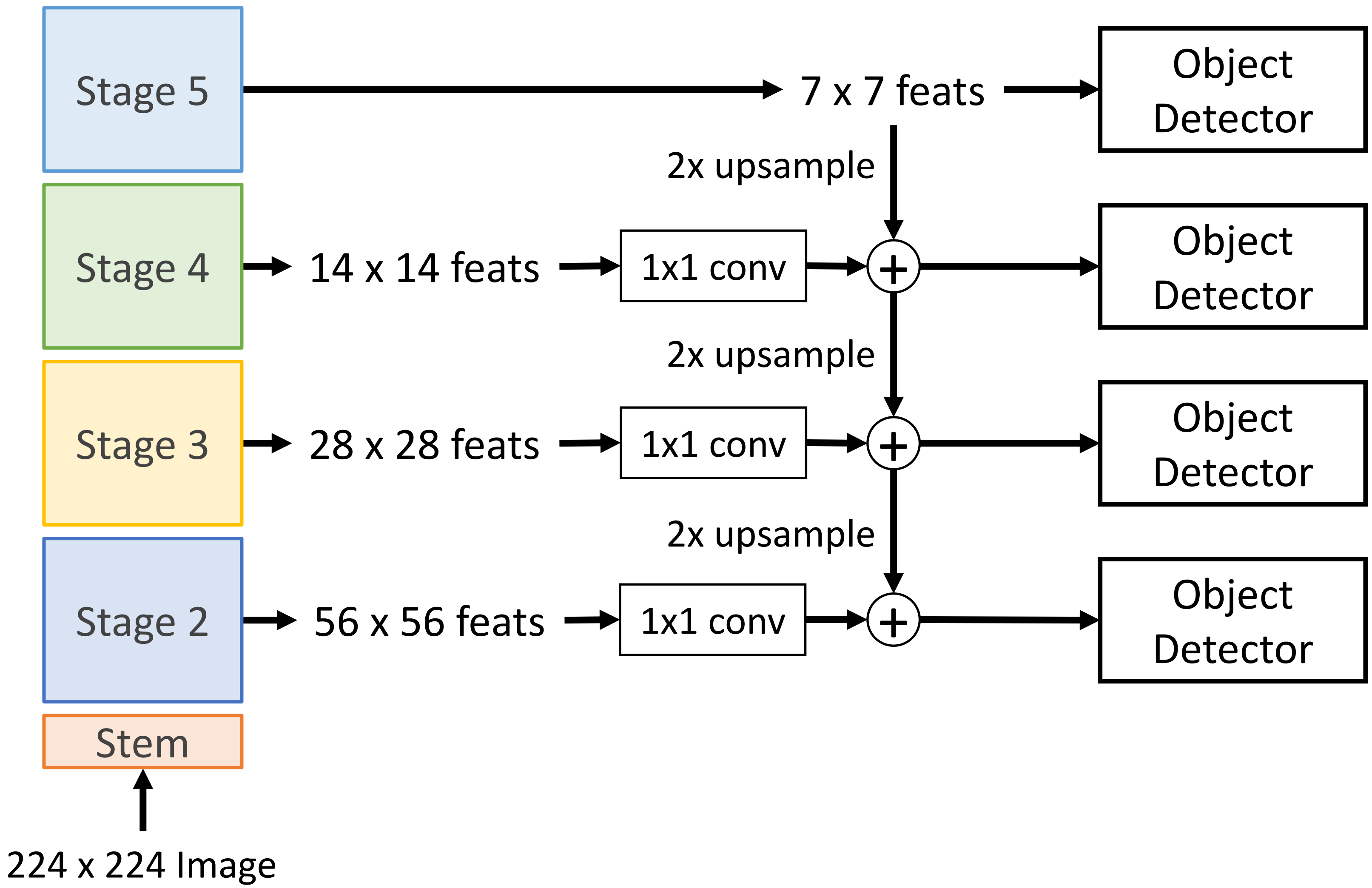
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Slide from Justin Johnson

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice



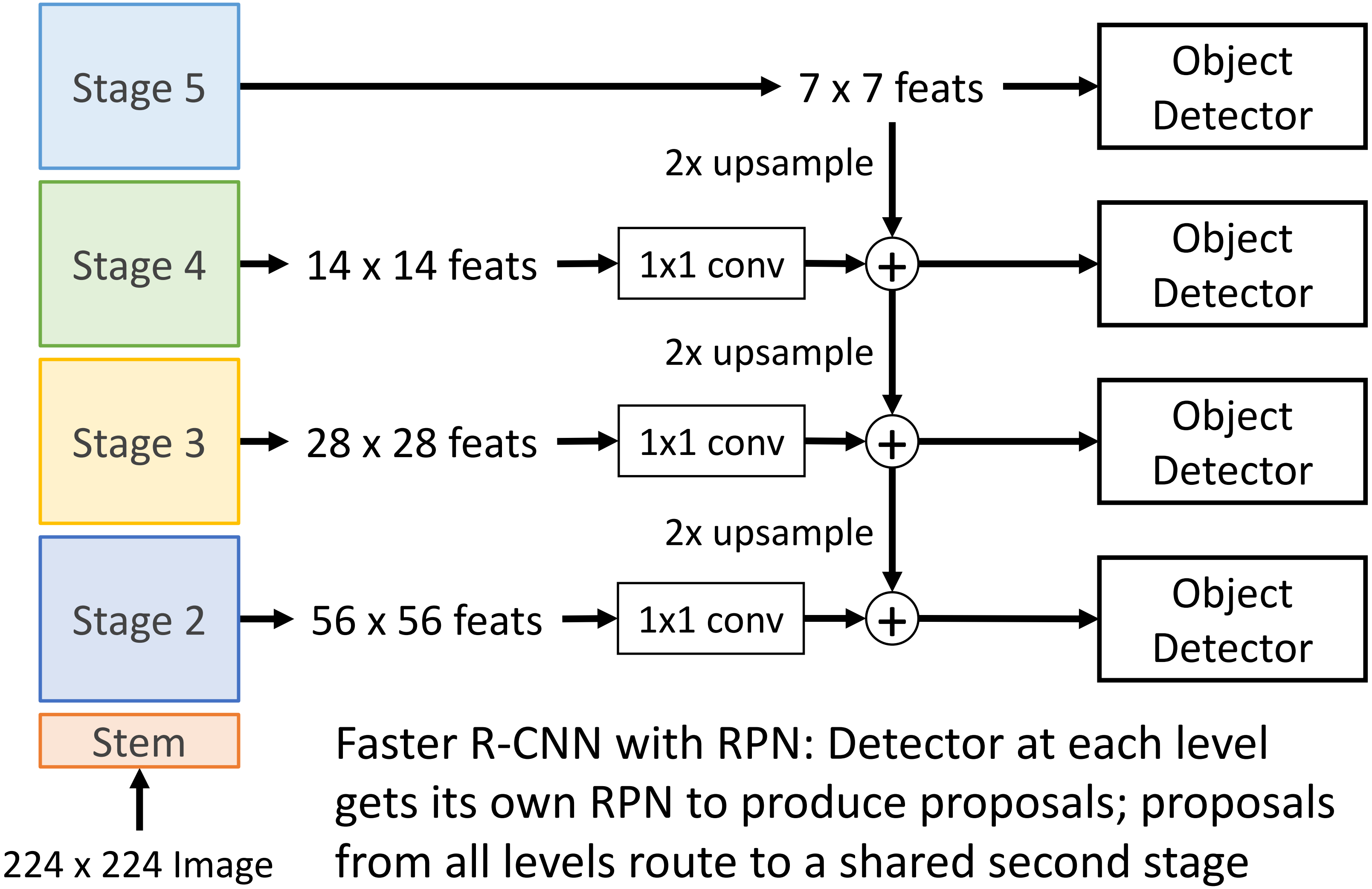
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Slide from Justin Johnson

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice



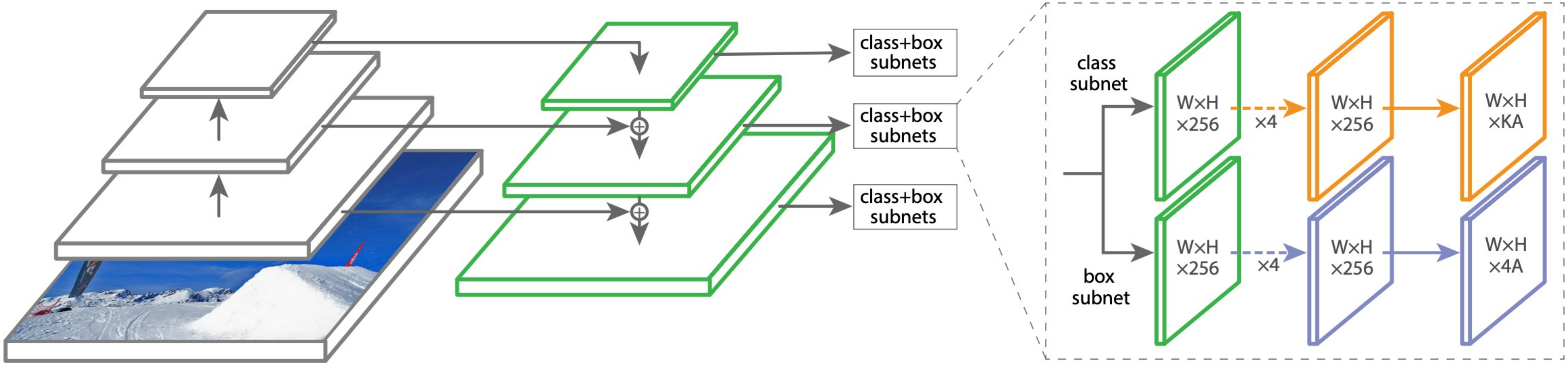
Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Slide from Justin Johnson

Single-Stage Detectors: RetinaNet

In practice, RetinaNet also uses Feature Pyramid Network to handle multiscale



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

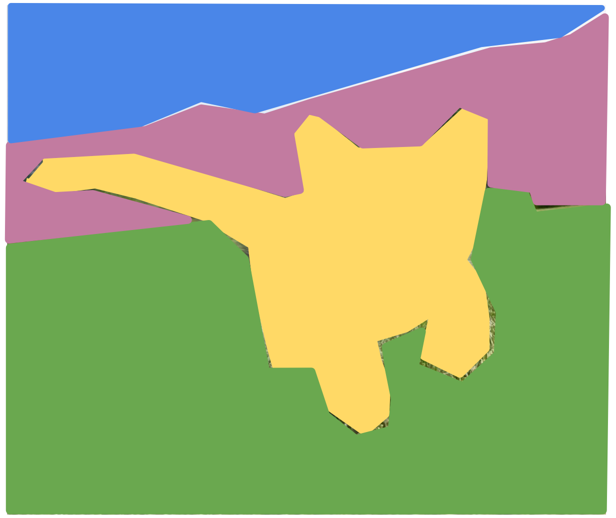
Figure credit: Lin et al, ICCV 2017

Beyond Image Classification

Classification

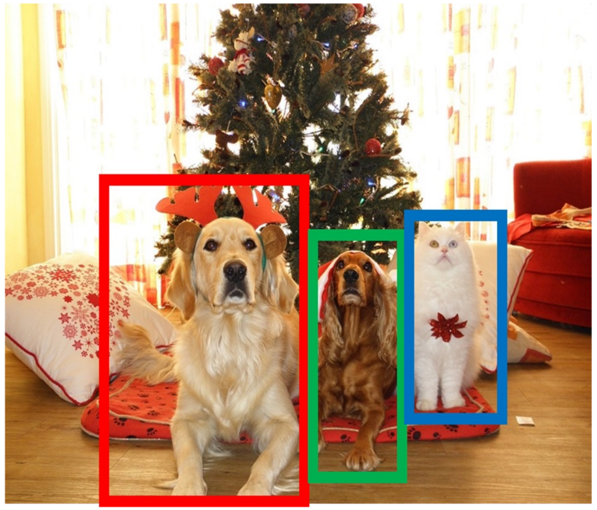
CAT

Semantic Segmentation



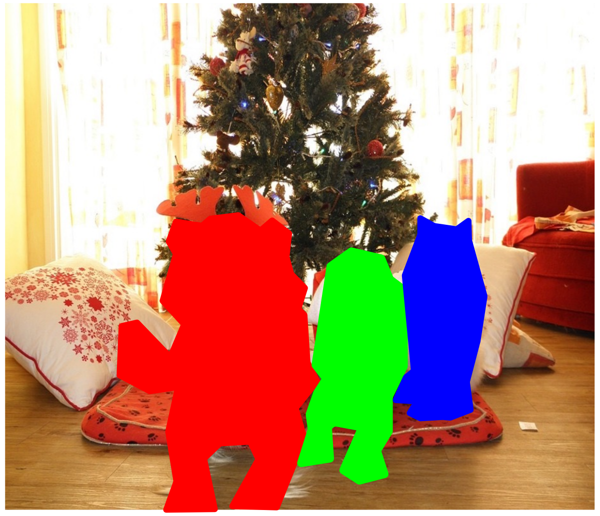
GRASS, CAT, TREE, SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

No spatial extent

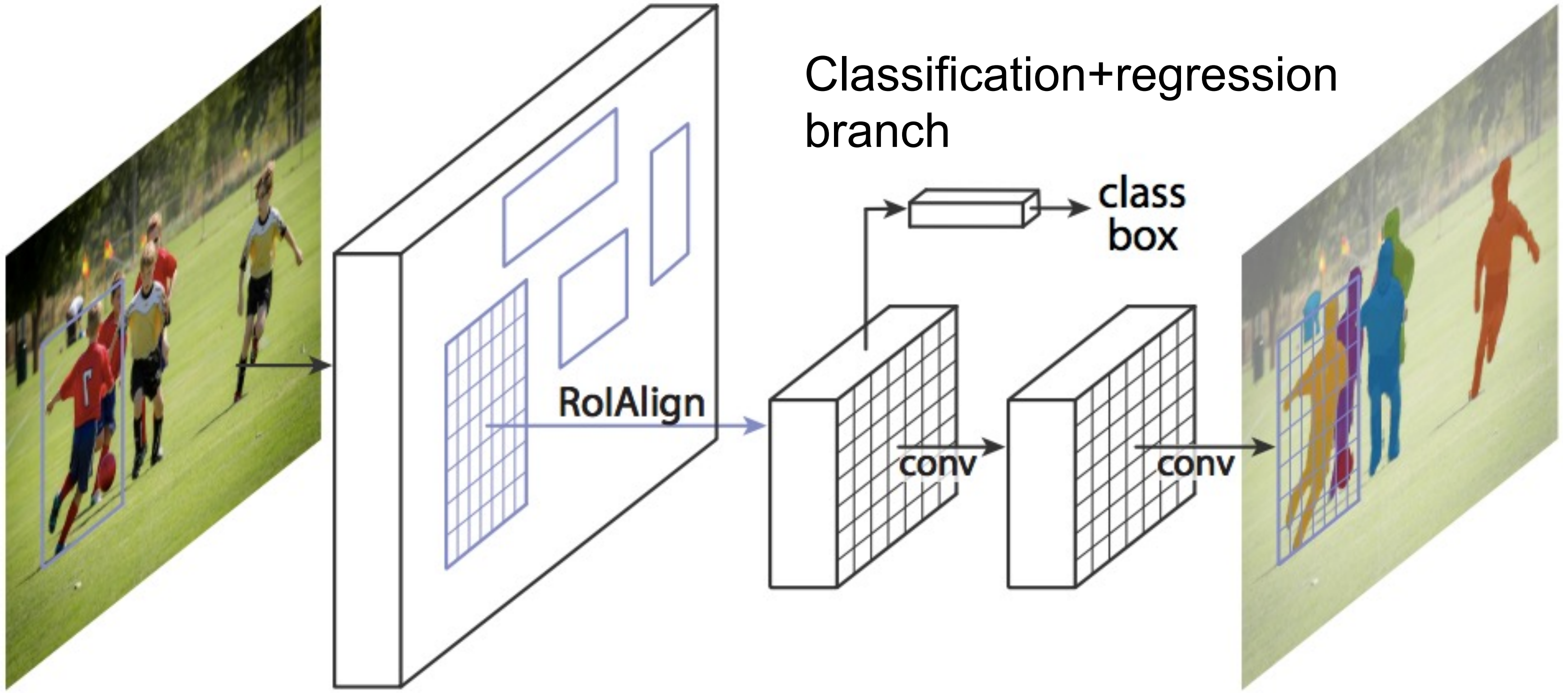
No objects, just pixels

Multiple Objects

Mask R-CNN

- Mask R-CNN = Faster R-CNN + FCN on RoIs

Classification+regression branch

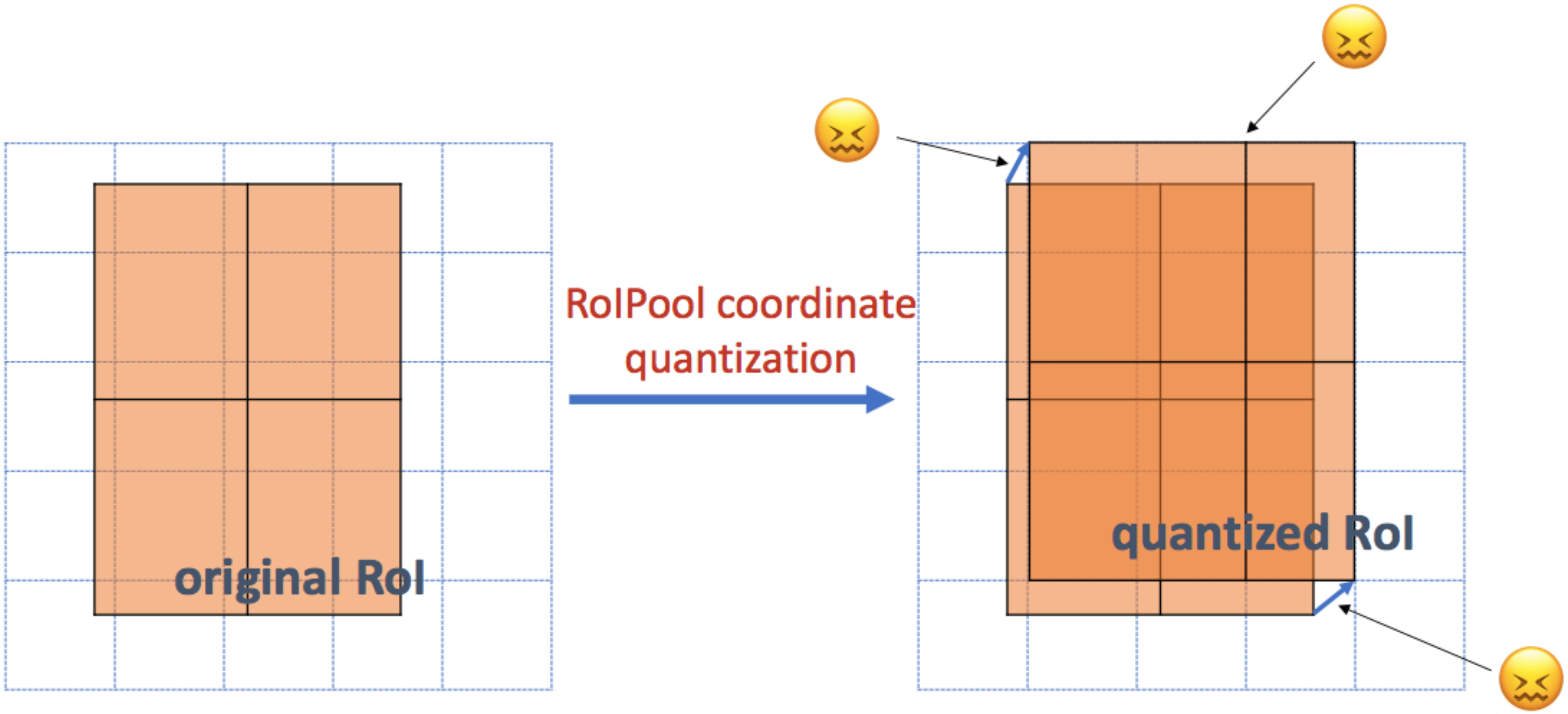


Mask branch: separately predict segmentation for each possible class

K. He, G. Gkioxari, P. Dollar, and R. Girshick, [Mask R-CNN](), ICCV 2017 (Best Paper Award)

RoIAlign vs. RoIPool

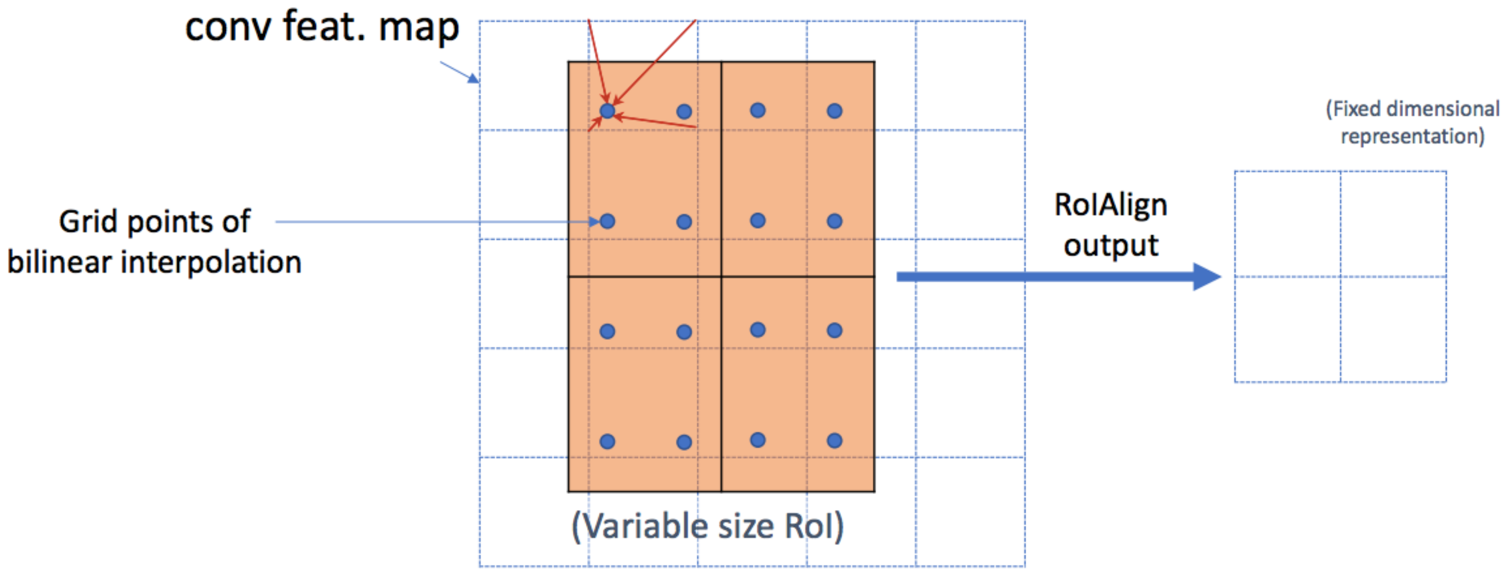
- RoIPool: nearest neighbor quantization



K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)

RoIAlign vs. RoIPool

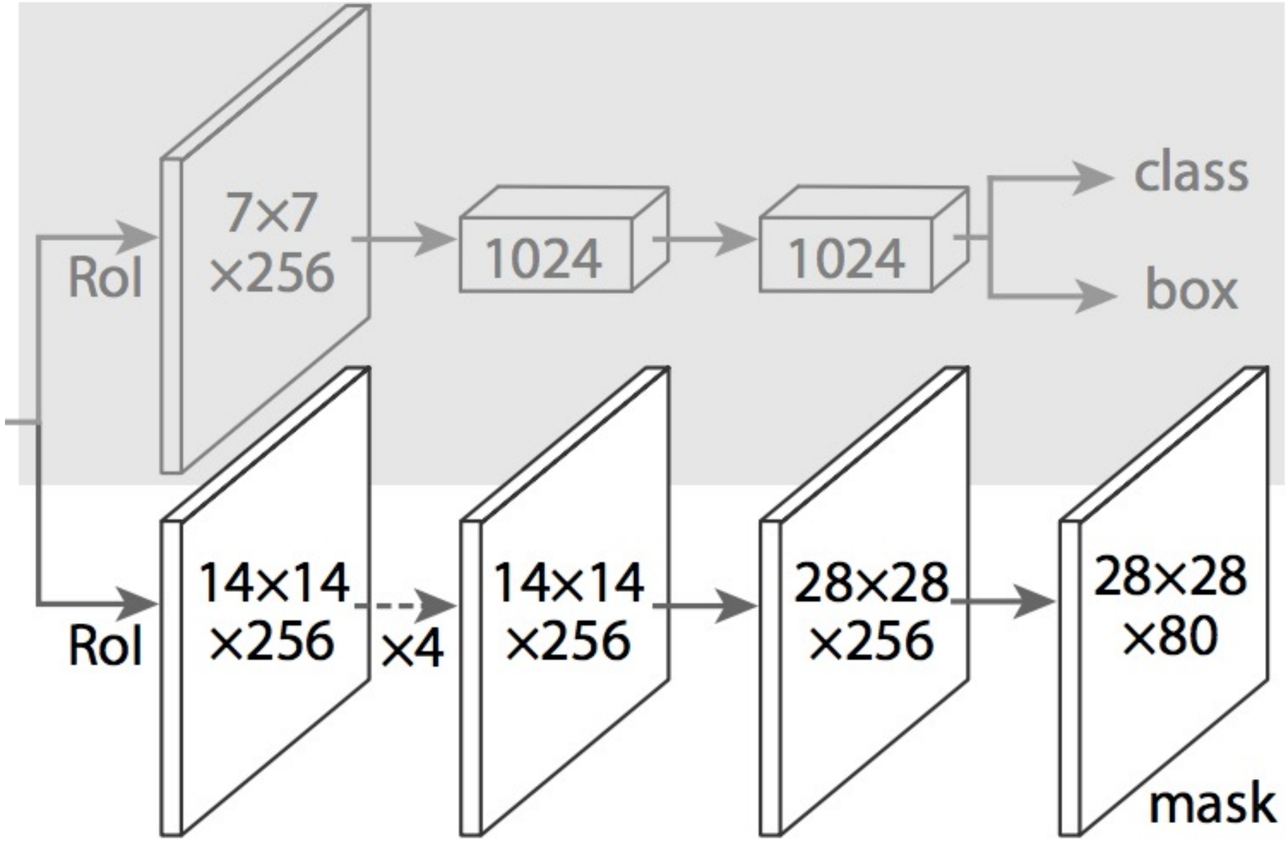
- RoIPool: nearest neighbor quantization
- RoIAlign: bilinear interpolation



K. He, G. Gkioxari, P. Dollar, and R. Girshick, [Mask R-CNN](), ICCV 2017 (Best Paper Award)

Mask R-CNN

- From RoIAlign features, predict class label, bounding box, and segmentation mask



Classification/regression head from an established object detector (e.g., FPN)

Separately predict binary mask for each class with per-pixel sigmoids, use average binary cross-entropy loss

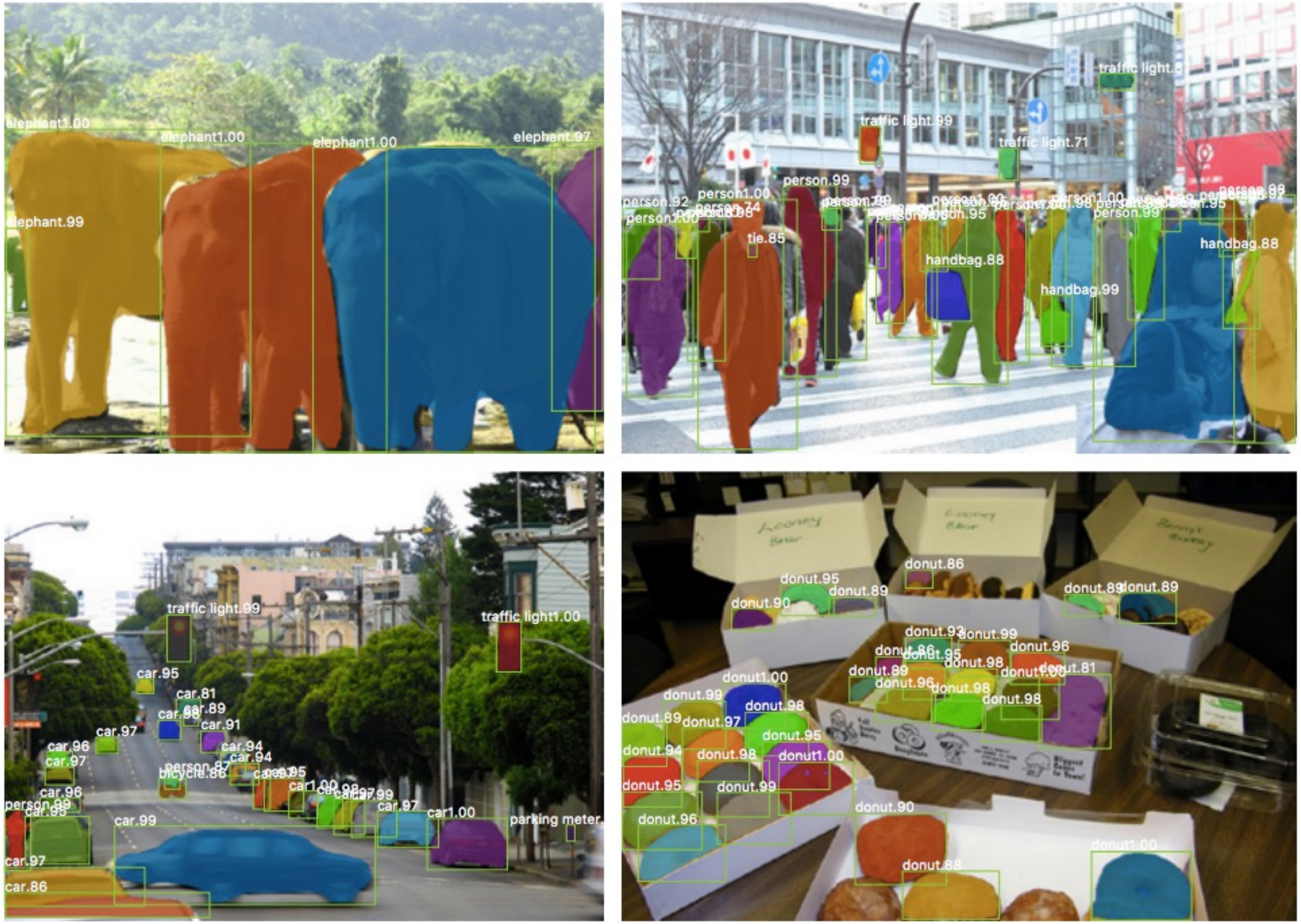
K. He, G. Gkioxari, P. Dollar, and R. Girshick, [Mask R-CNN](), ICCV 2017 (Best Paper Award)

Mask R-CNN

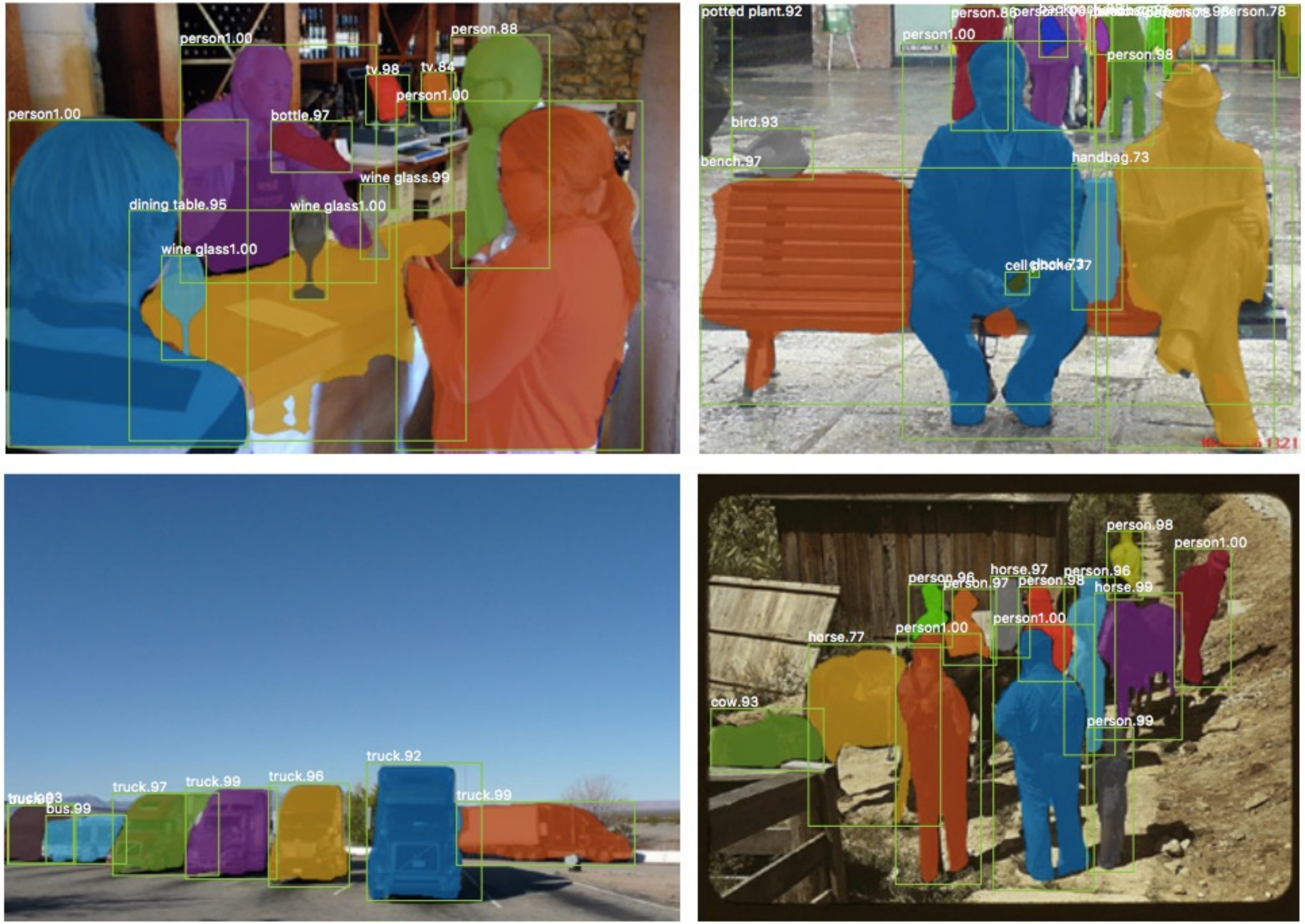
Validation image with box detection shown in red

K. He, G. Gkioxari, P. Dollar, and R. Girshick, [Mask R-CNN](),
ICCV 2017 (Best Paper Award)

Example results



Example results



Instance segmentation results on COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

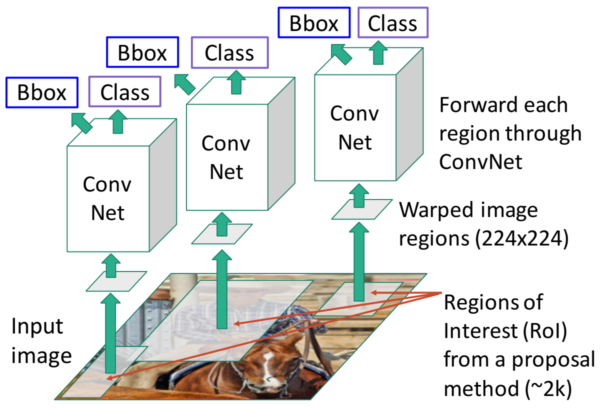
AP at different IoU thresholds

AP for different size instances

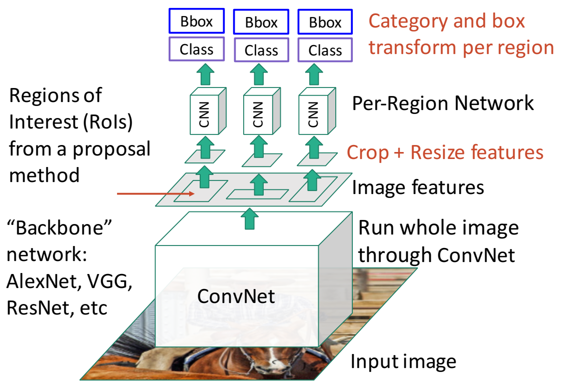
K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)

Summary

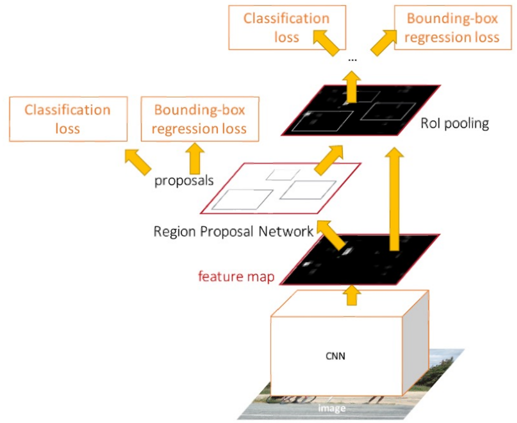
"Slow" R-CNN: Run CNN independently for each region



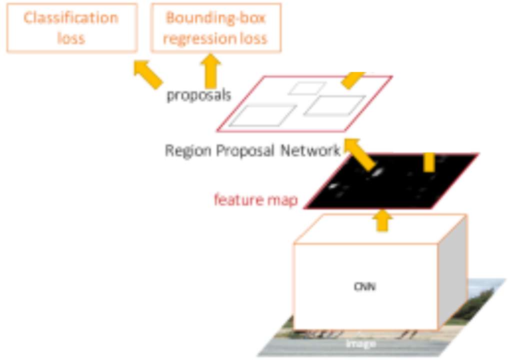
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



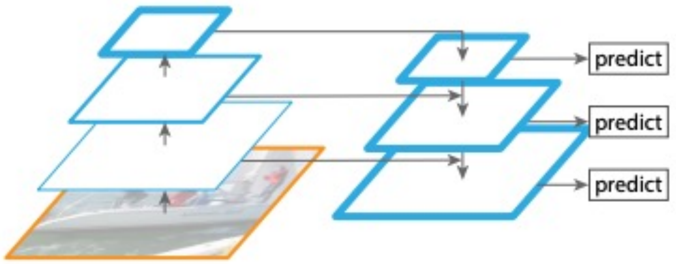
Single-Stage: Fully convolutional detector / RetinaNet



Bounding Box Regression

RoIPool / RoIAlign

Feature Pyramid Network



Summary

Object detection is the task of localizing objects with bounding boxes

Intersection over Union (IoU) quantifies differences between bounding boxes

The **R-CNN** object detector processes **region proposals** with a CNN

At test-time, eliminate overlapping detections using **non-max suppression (NMS)**

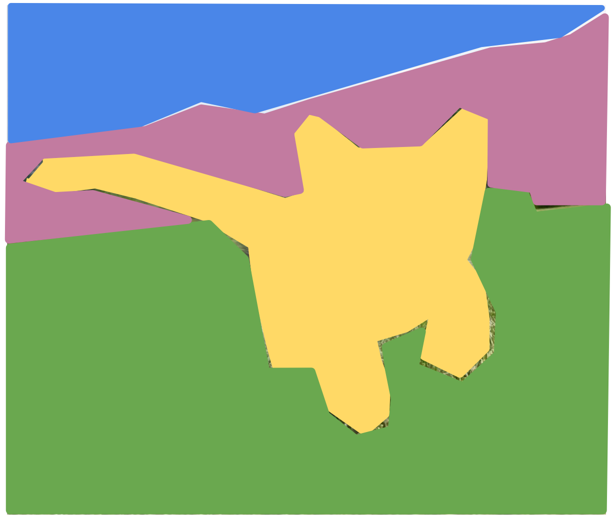
Evaluate object detectors using **mean average precision (mAP)**

Summary: Beyond Image Classification

Classification

CAT

Semantic Segmentation

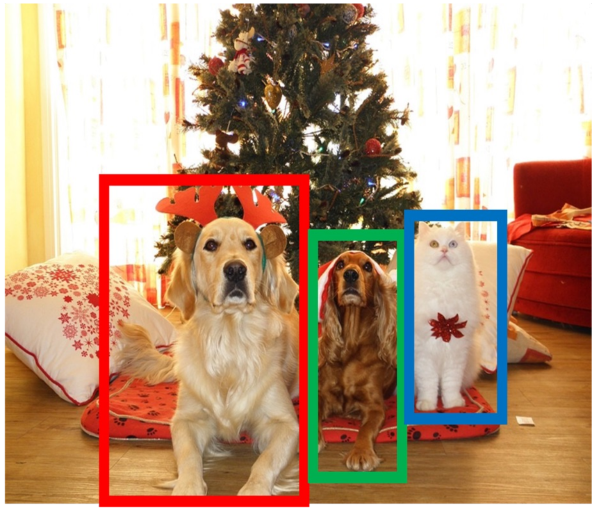


GRASS, CAT, TREE, SKY

No spatial extent

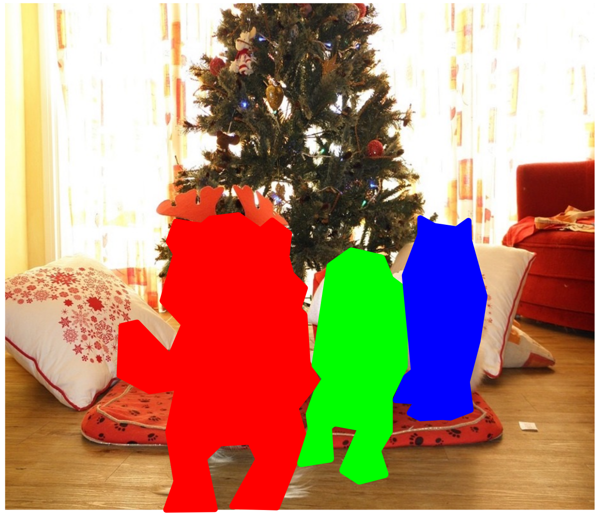
No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Objects