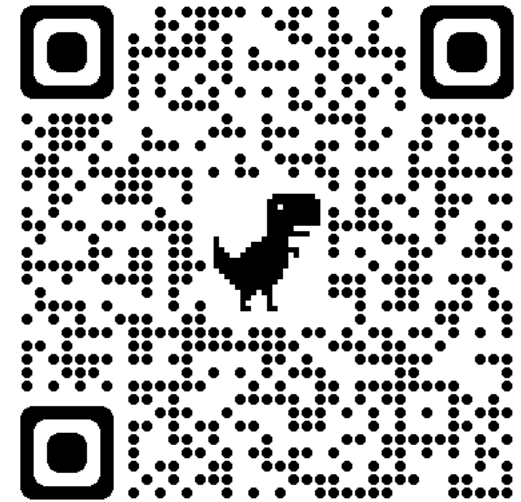


Lecture 20: Object Detection

COMP 590/776: Computer Vision

Instructor: Soumyadip (Roni) Sengupta

TA: Mykhailo (Misha) Shvets

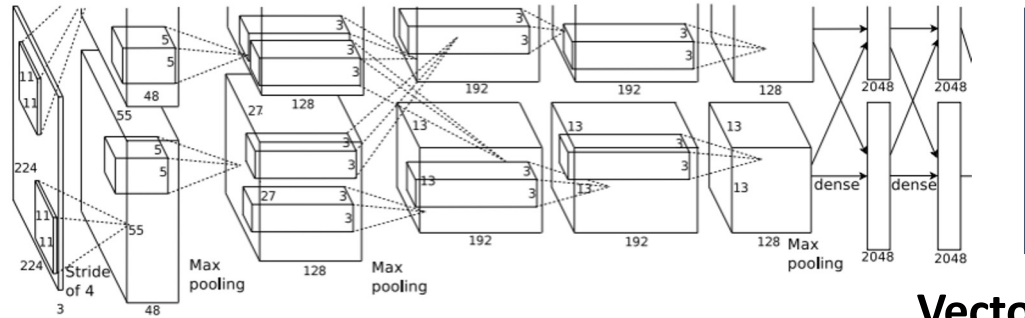


Course Website:
Scan Me!

Slide Credits

- EECS 442/498 [Computer Vision](#), by Justin Johnson & David Fouhey, U Michigan.

So far: Image Classification



Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores

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

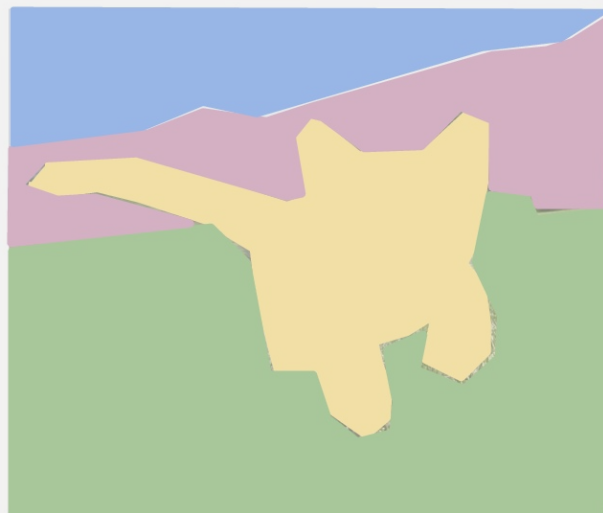
Classification



CAT

No spatial extent

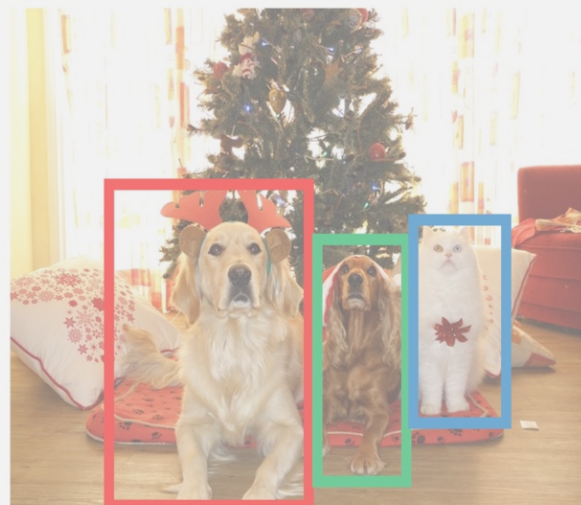
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Objects

Instance Segmentation



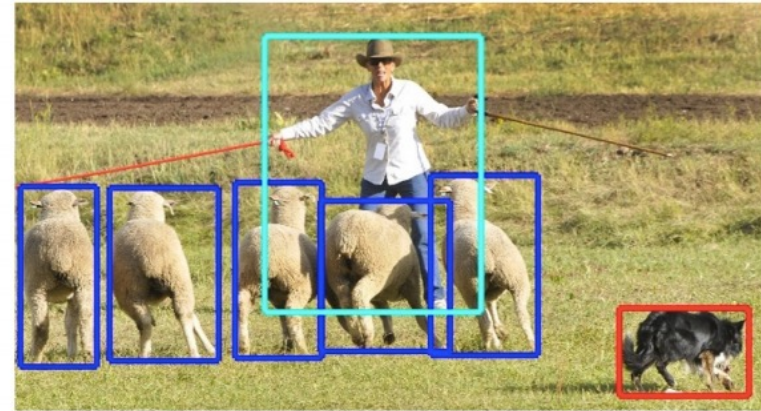
DOG, DOG, CAT

Recognition: What type of output?

Image classification



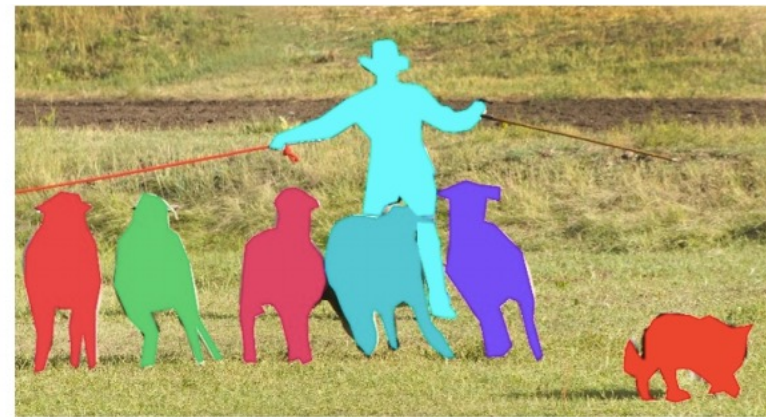
Object detection



Semantic segmentation



Instance segmentation

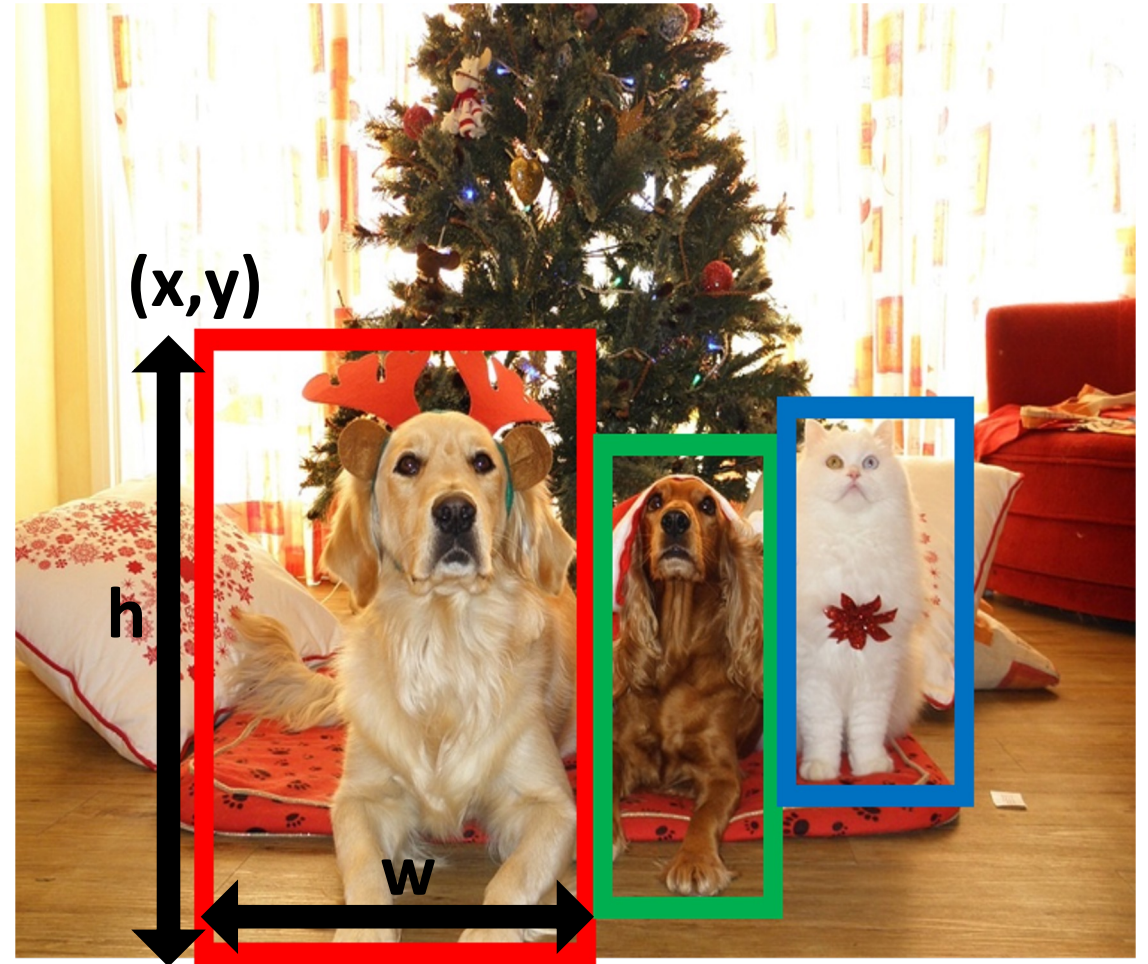


Object Detection: Task Definition

Input: Single RGB Image

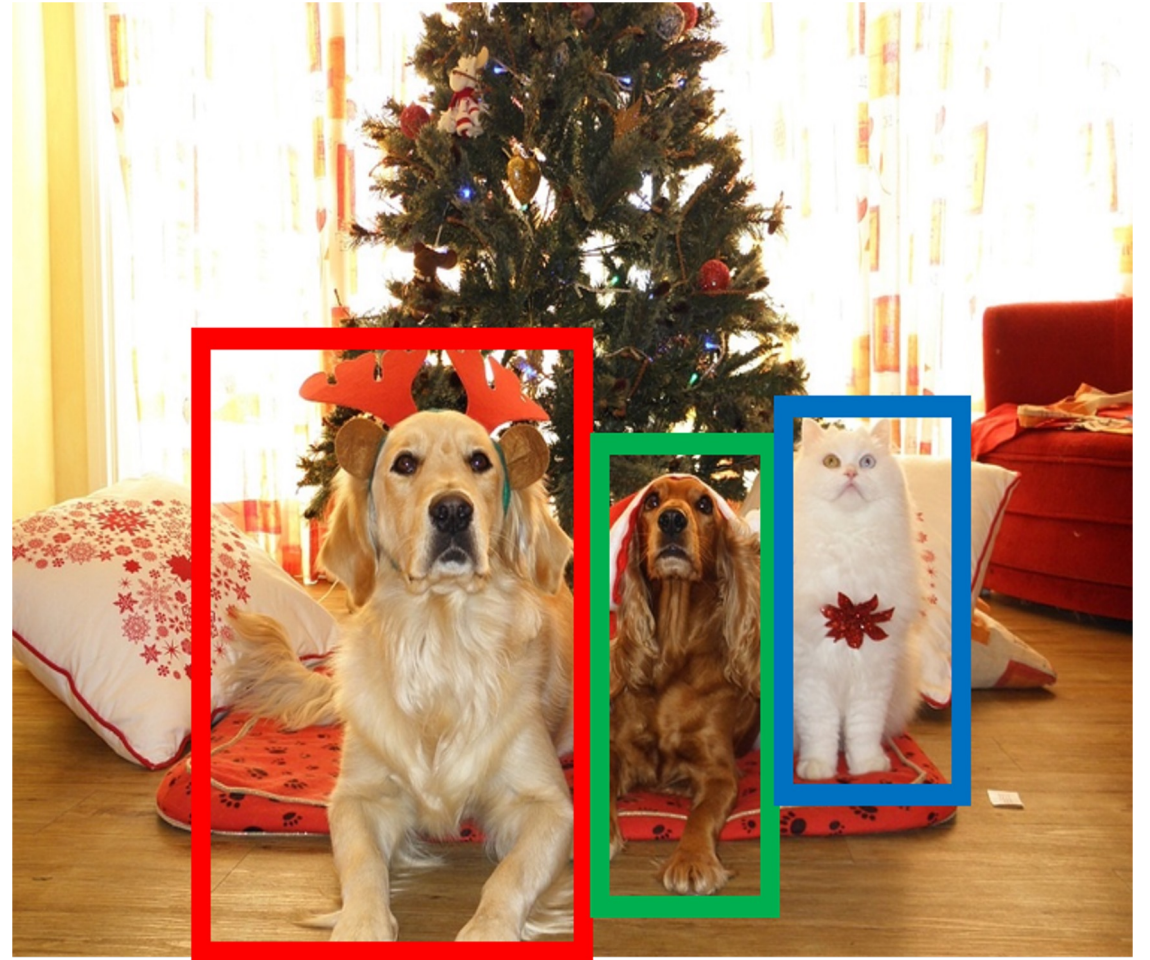
Output: A set of detected objects;
For each object predict:

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



Object Detection: Challenges

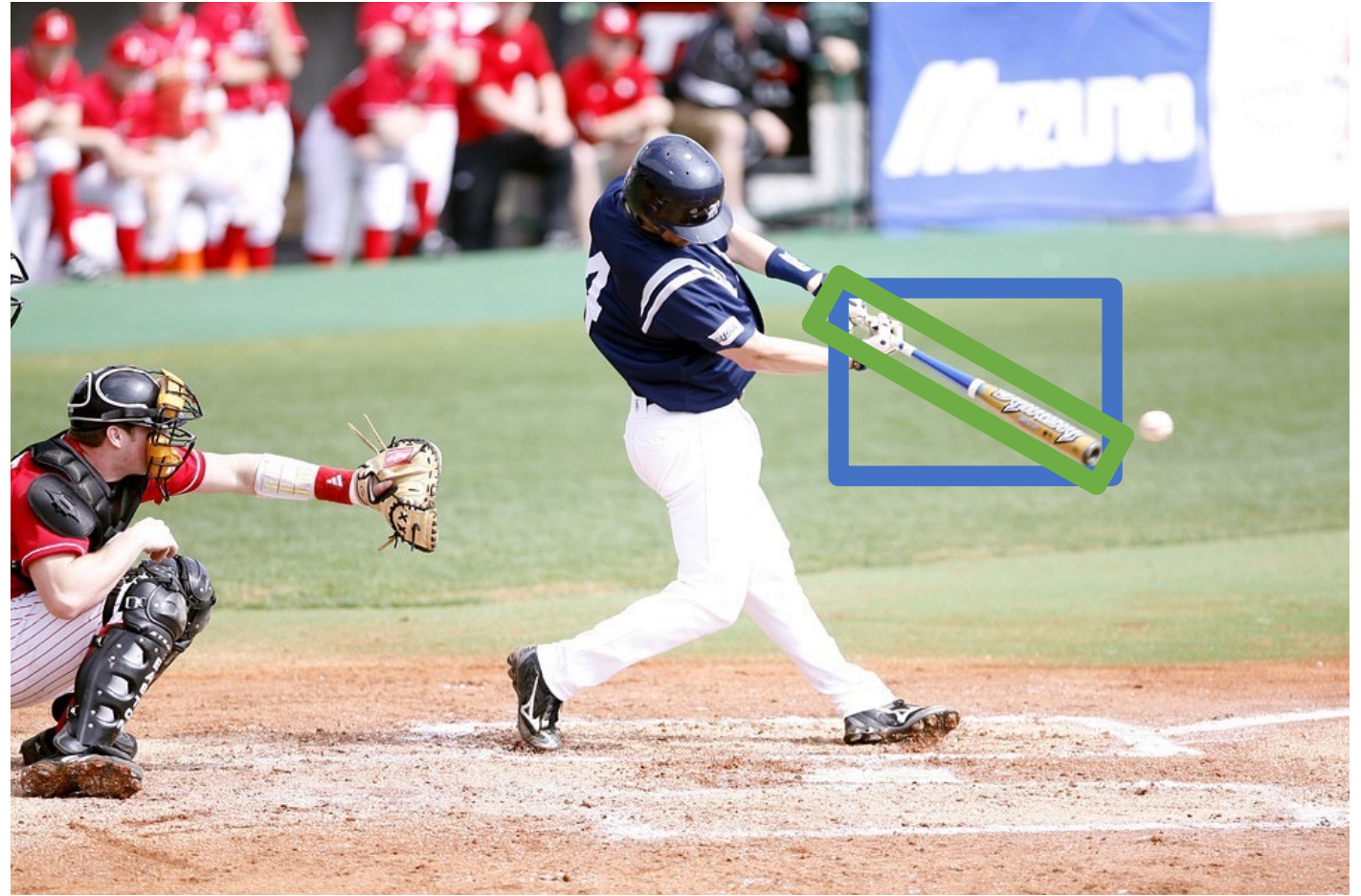
- **Multiple outputs:** Need to output variable numbers of objects per image
- **Multiple types of output:** Need to predict "what" (category label) as well as "where" (bounding box)
- **Large images:** Classification works at 224x224; need higher resolution for detection, often ~800x600



Bounding Boxes

Bounding boxes are typically *axis-aligned*

Oriented boxes are much less common



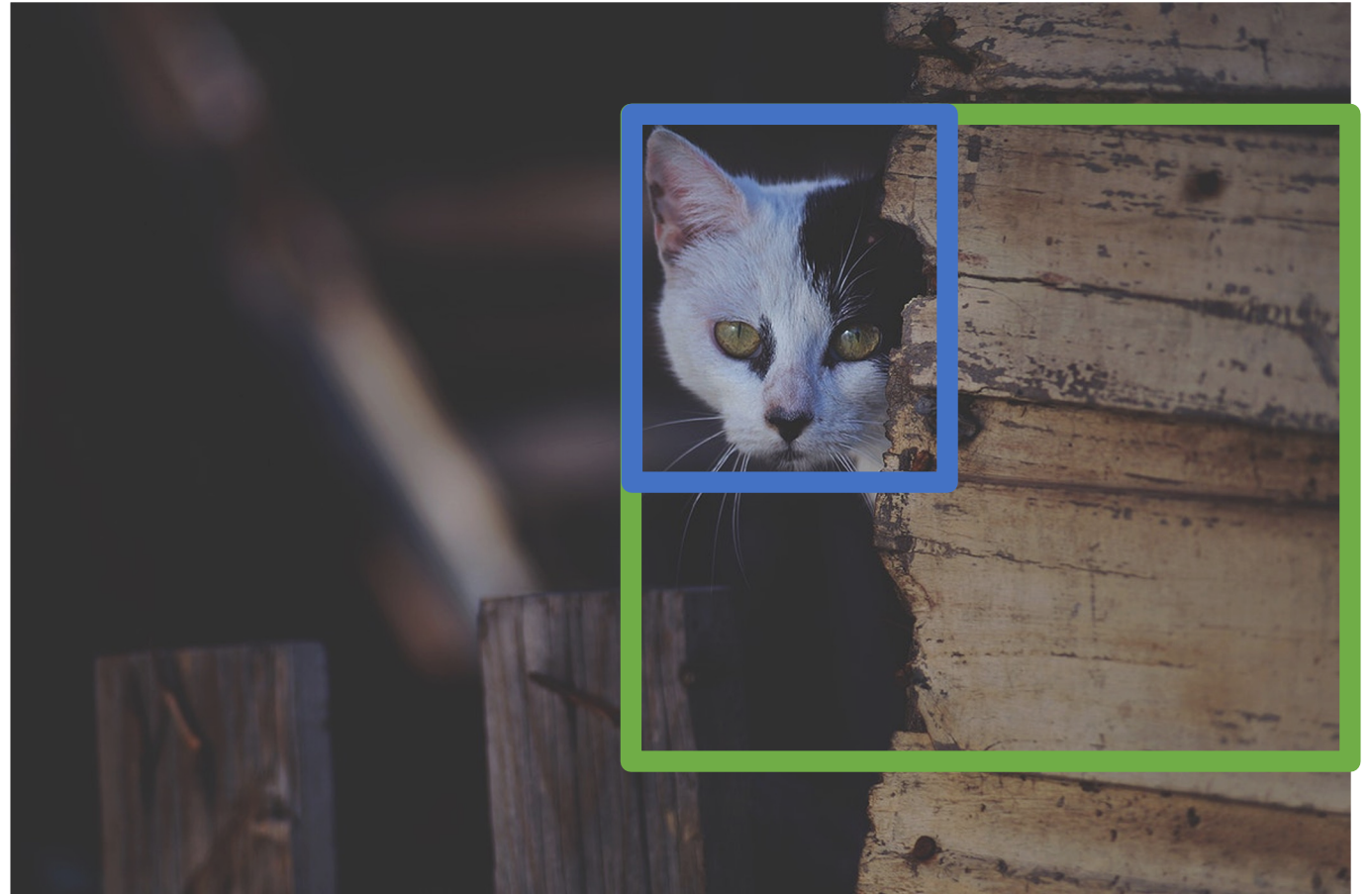
Object Detection: Modal vs Amodal Boxes

“Modal” detection:

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

Amodal detection:

box covers the entire extent of the object, even occluded parts



Today's class

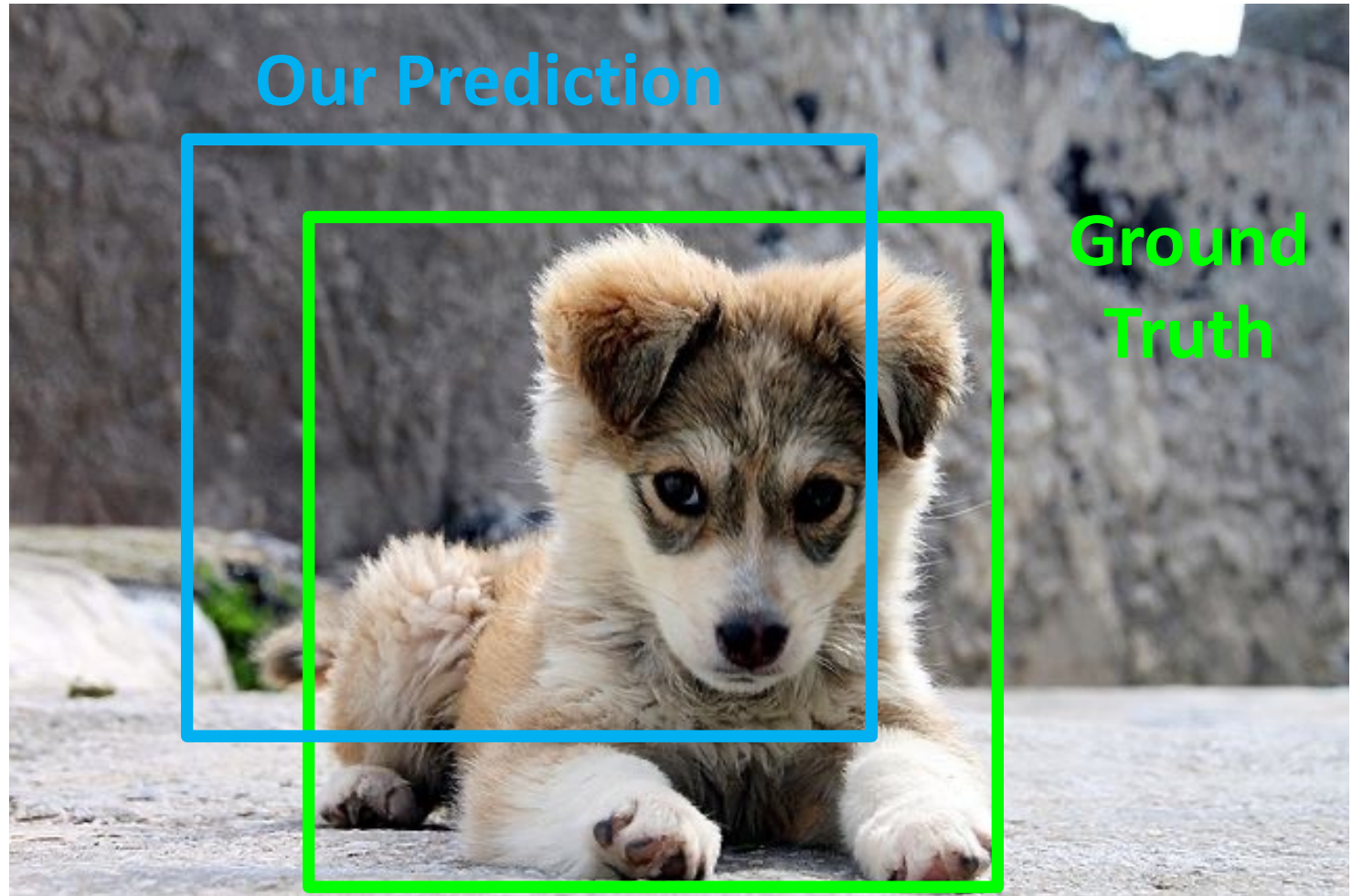
- How do we measure Object Detection accuracy?
- Naïve approaches & R-CNN
- Fast R-CNN
- Region Proposal Network & Faster R-CNN
- Advanced topics:
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

Today's class

- How do we measure Object Detection accuracy?
- Naïve approaches & R-CNN
- Fast R-CNN
- Region Proposal Network & Faster R-CNN
- Advanced topics:
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

Comparing Boxes: Intersection over Union (IoU)

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

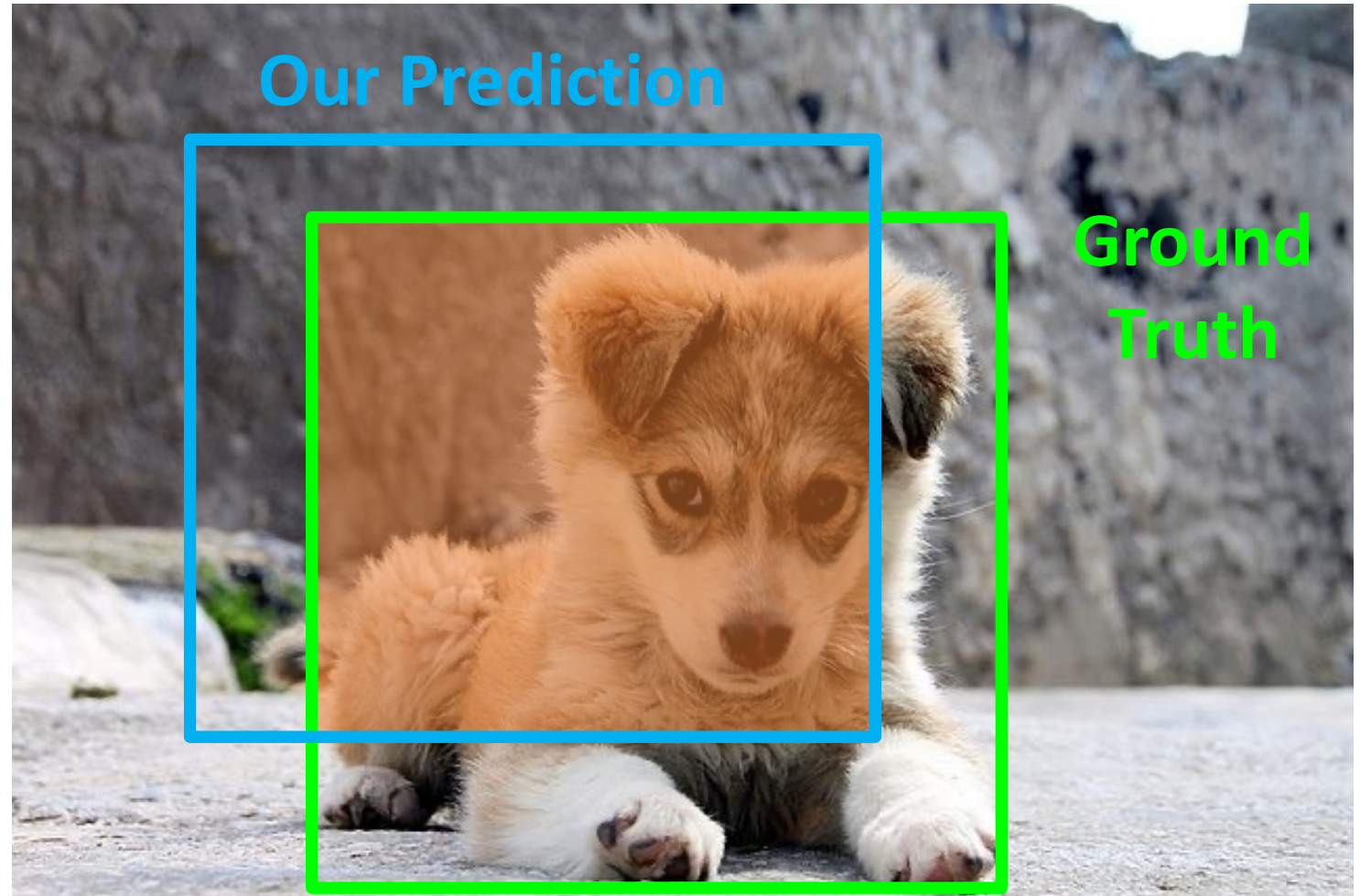


Comparing Boxes: Intersection over Union (IoU)

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

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

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

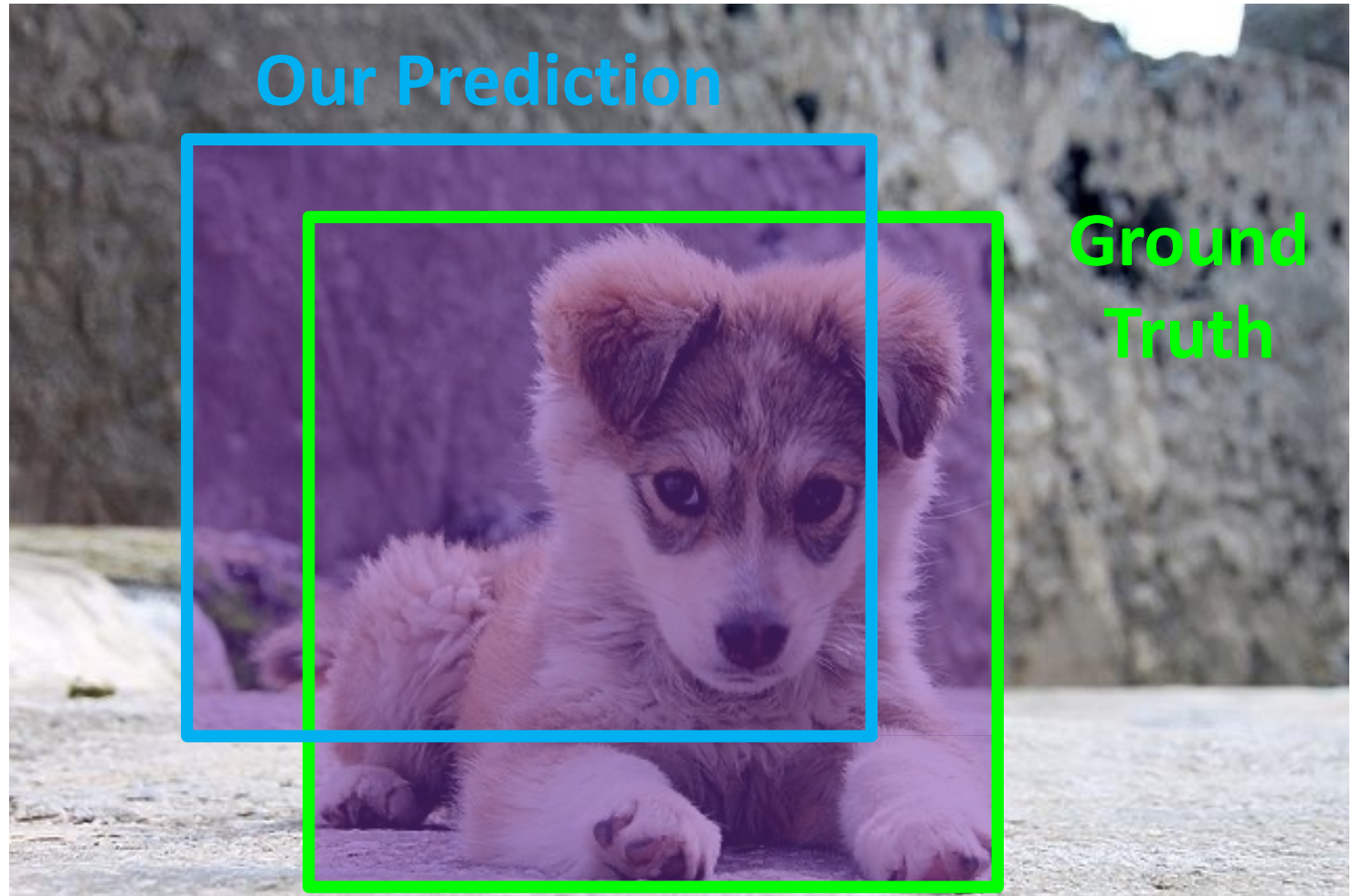


Comparing Boxes: Intersection over Union (IoU)

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

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

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



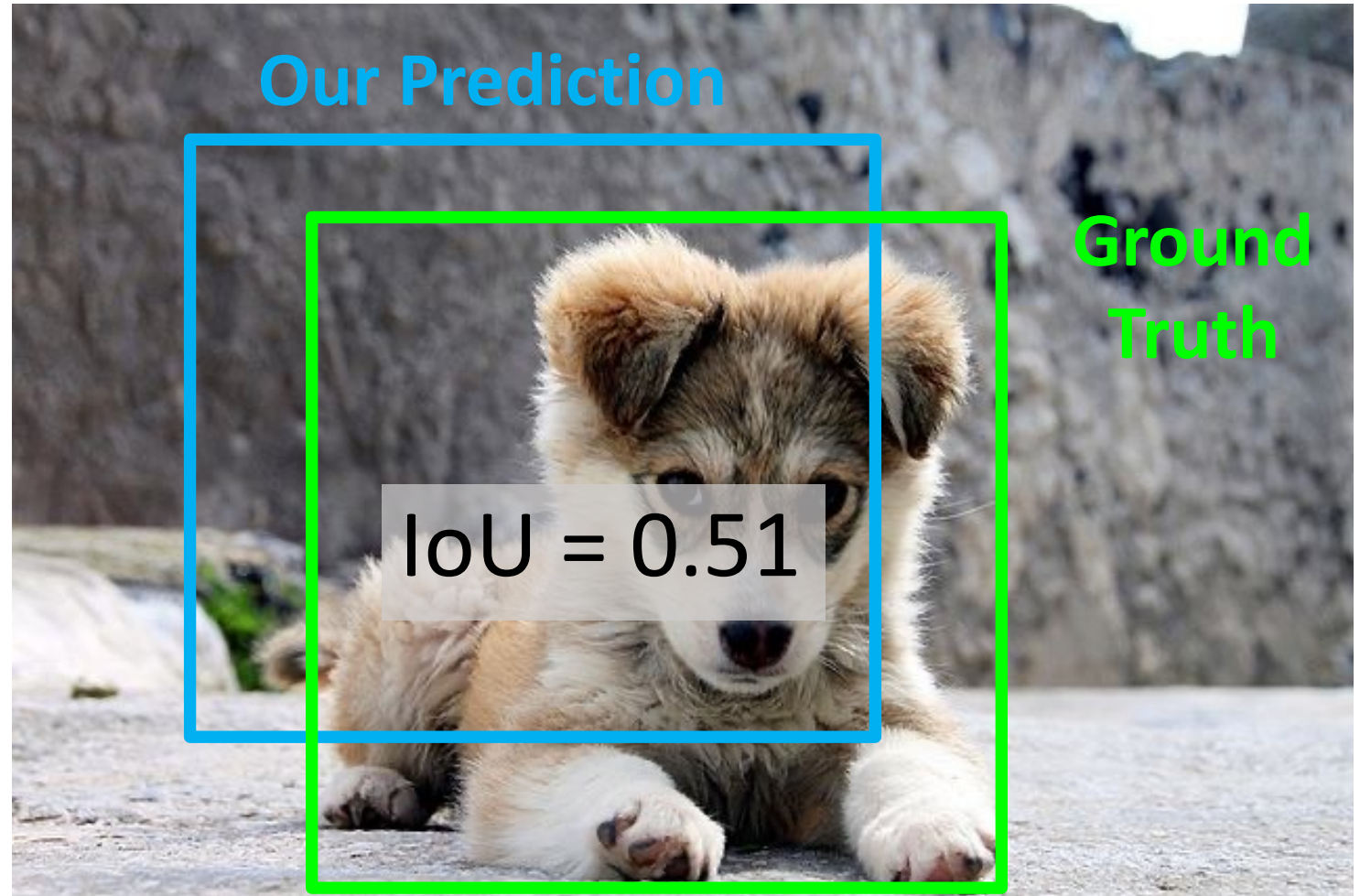
Comparing Boxes: Intersection over Union (IoU)

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

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

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

$\text{IoU} > 0.5$ is “decent”



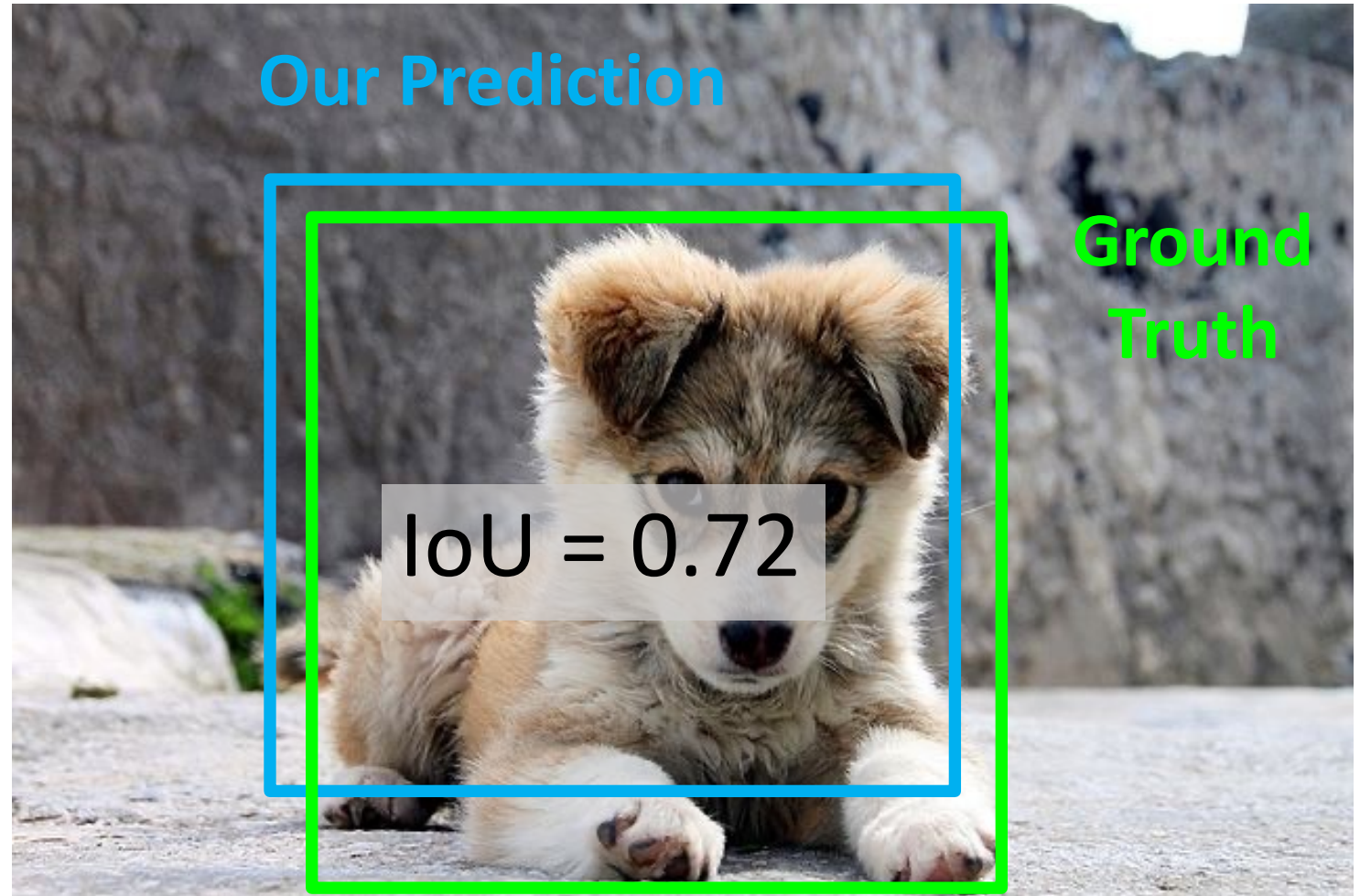
Comparing Boxes: Intersection over Union (IoU)

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

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

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

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



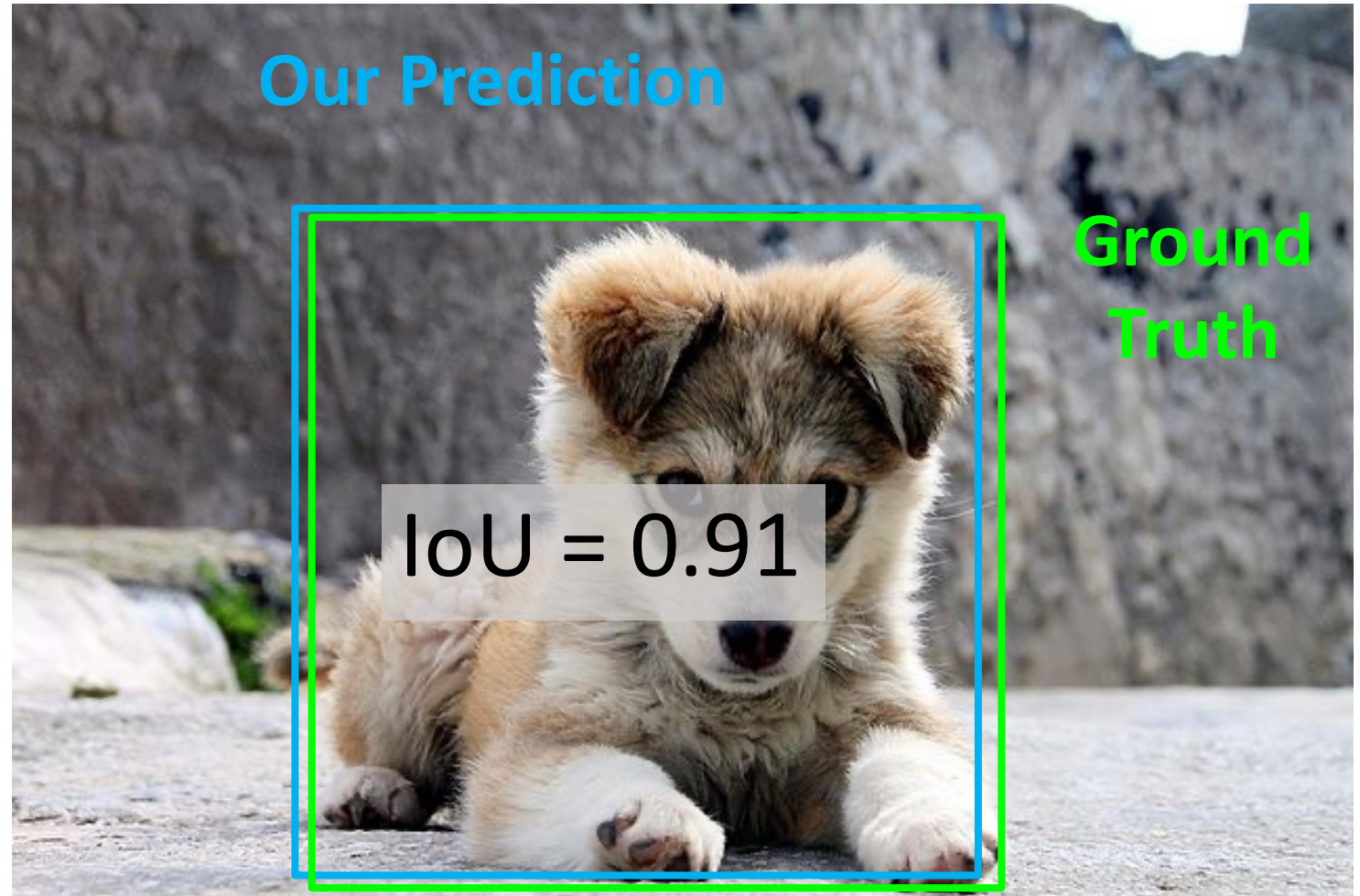
Comparing Boxes: Intersection over Union (IoU)

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

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

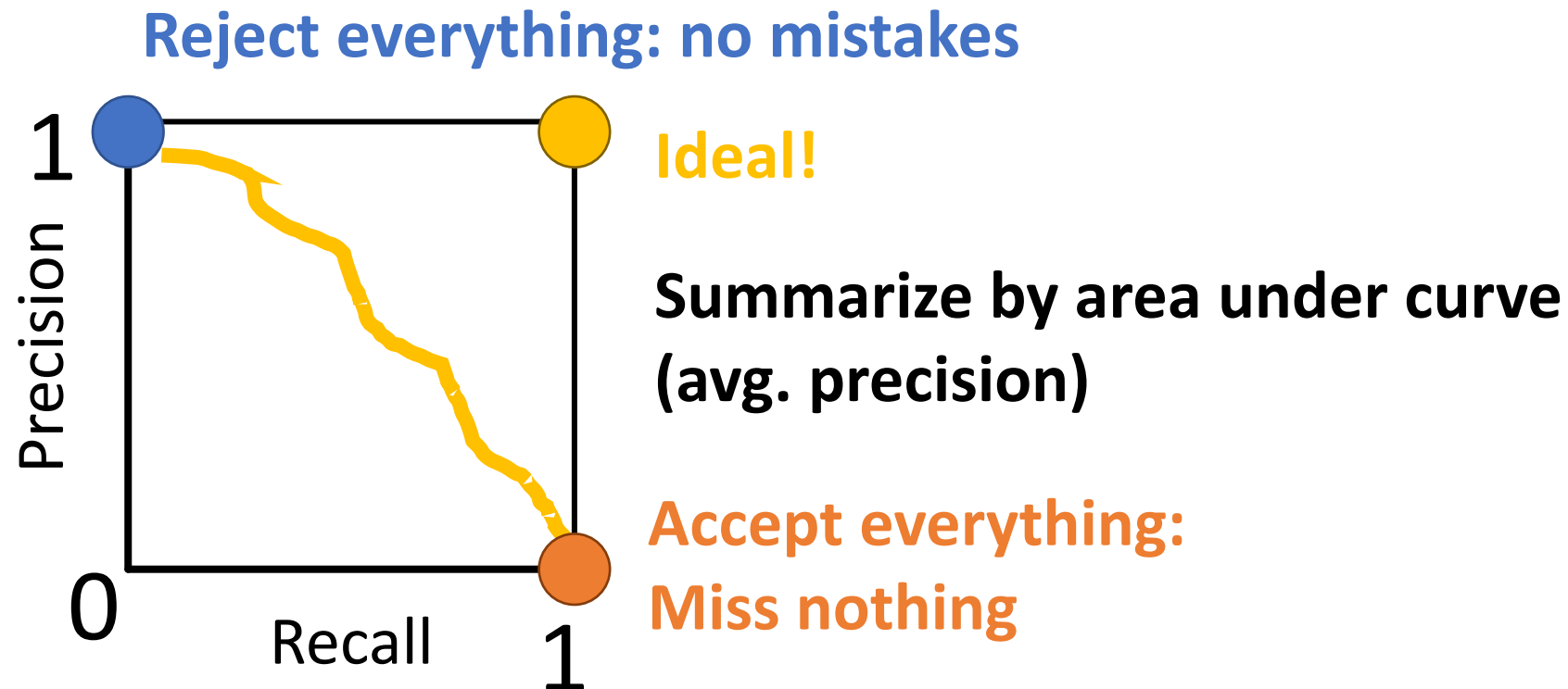
$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$

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



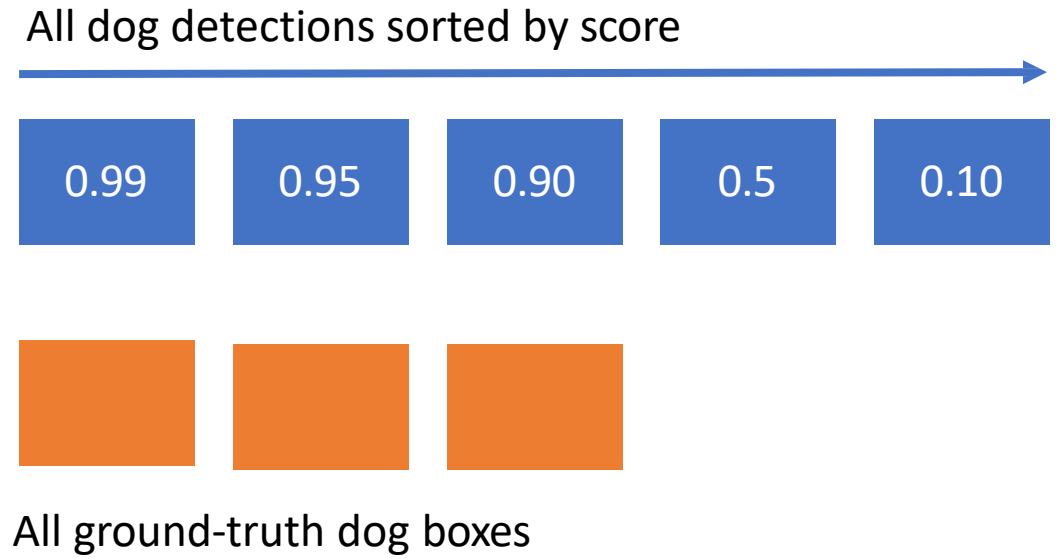
Precision & Recall

- True detection: high intersection over union
- Precision: $\# \text{true detections} / \# \text{detections}$
- Recall: $\# \text{true detections} / \# \text{true positives}$



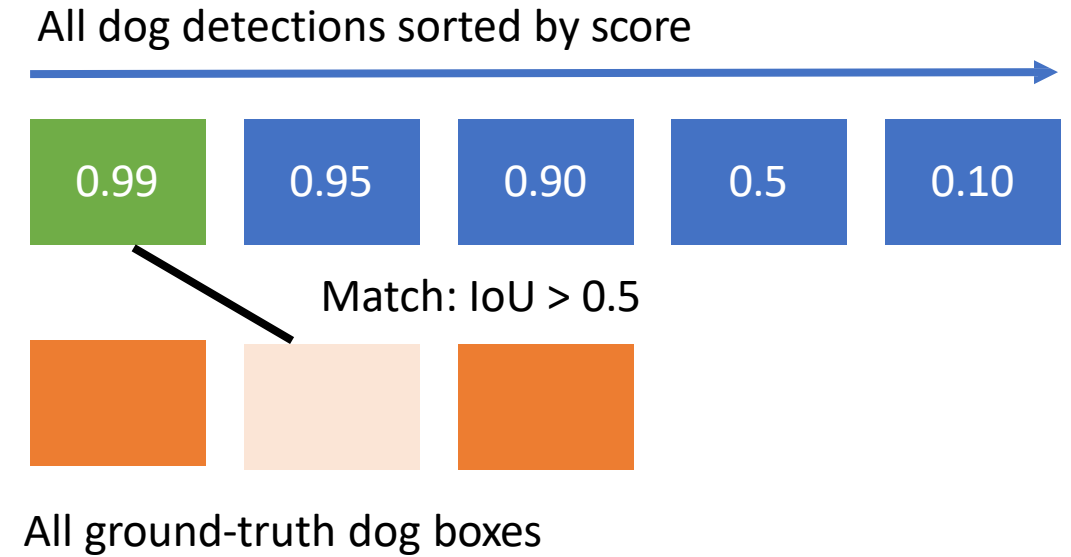
Evaluating Object Detectors: Mean Average Precision (mAP)

1. For each category, compute Average Precision (AP)
= area under Precision vs Recall Curve



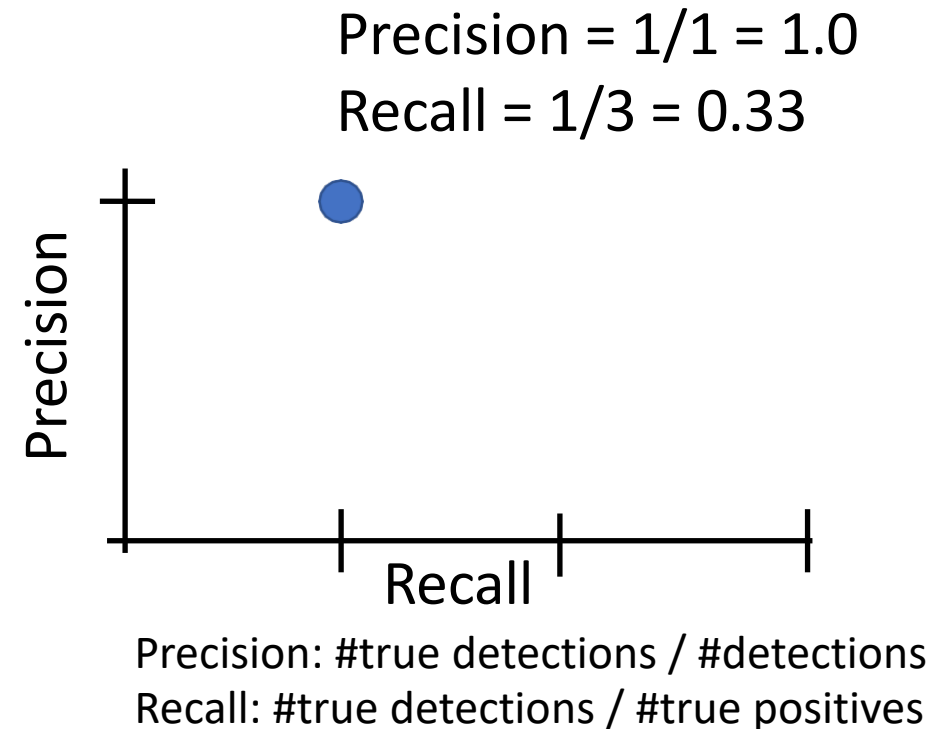
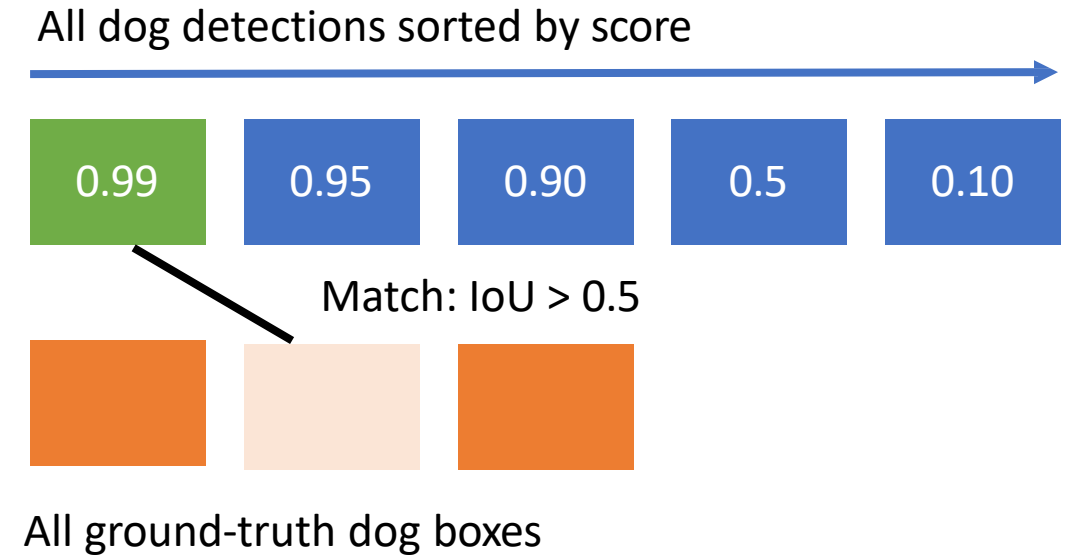
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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 $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative



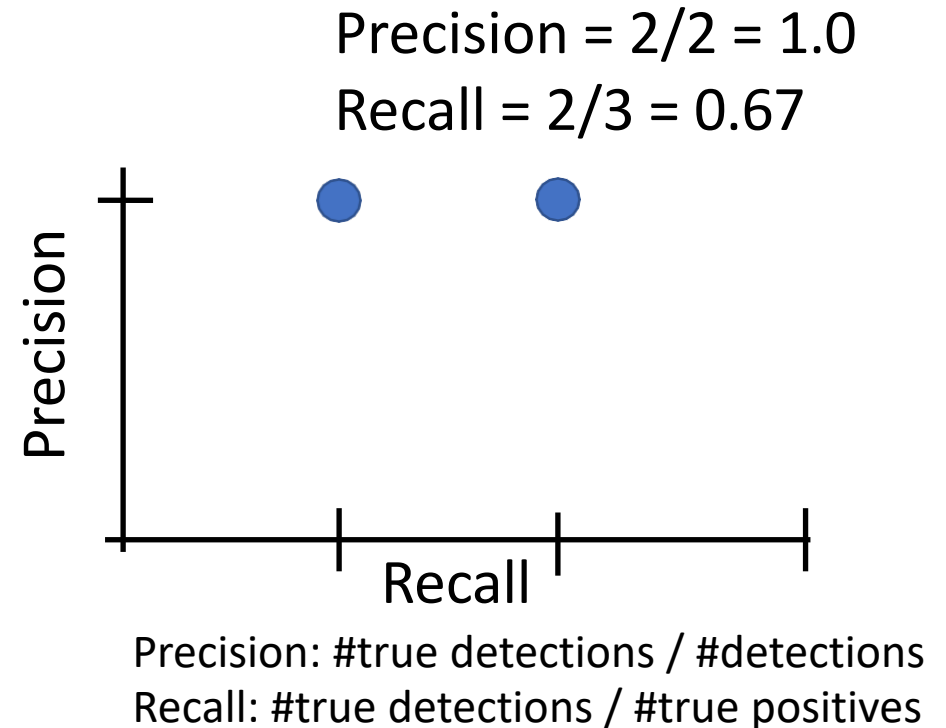
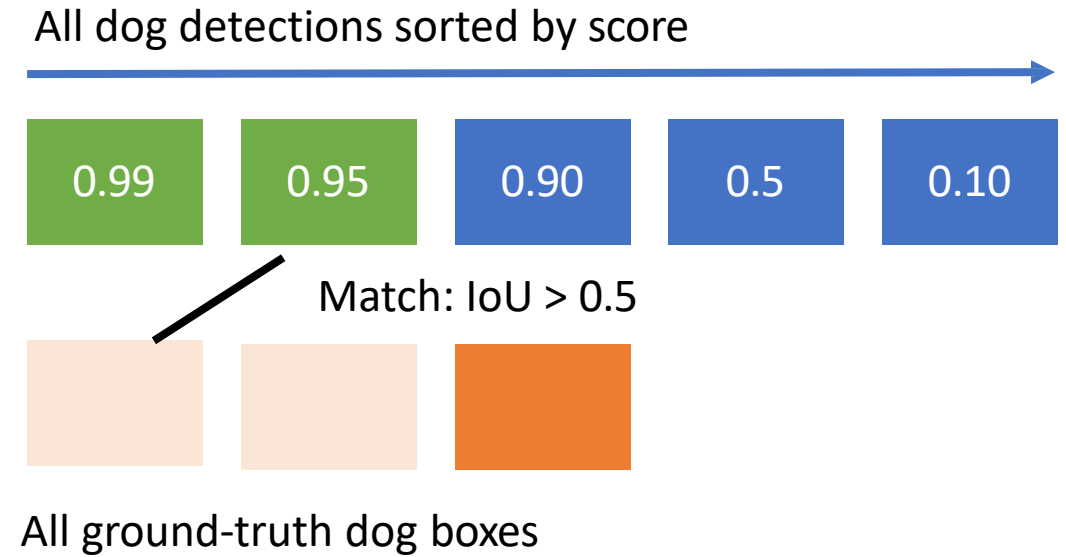
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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



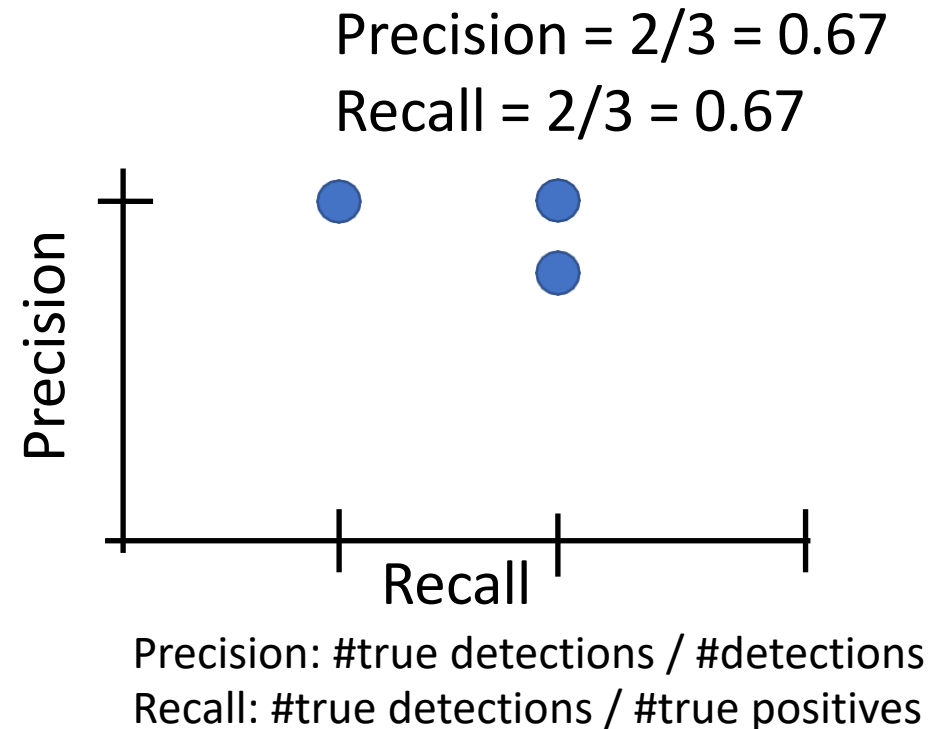
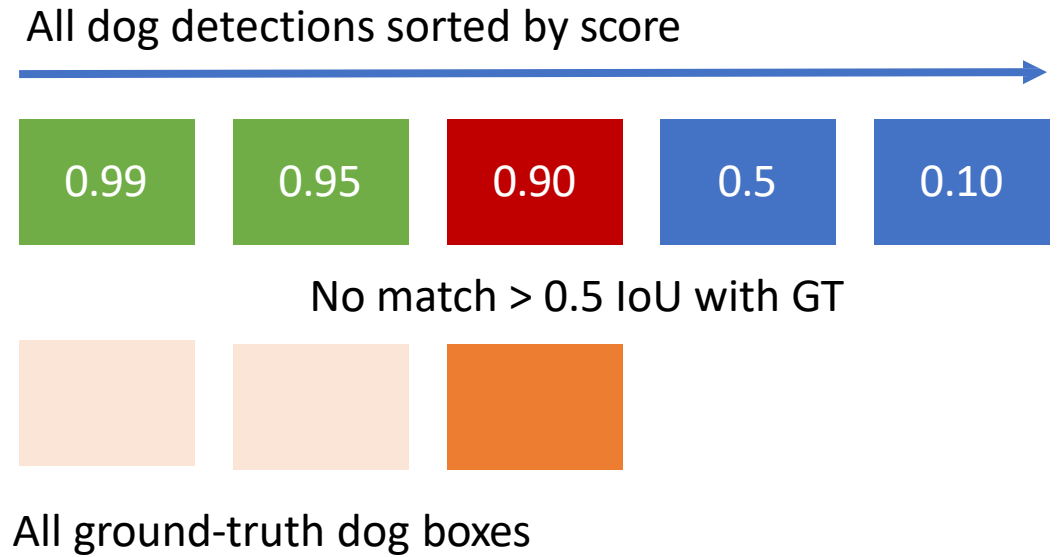
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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



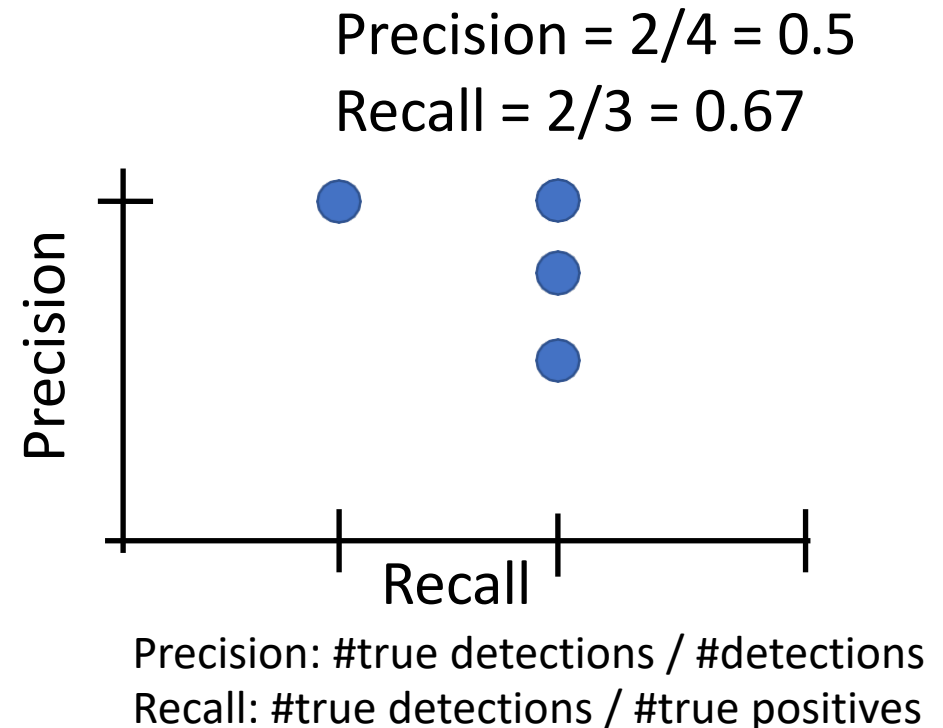
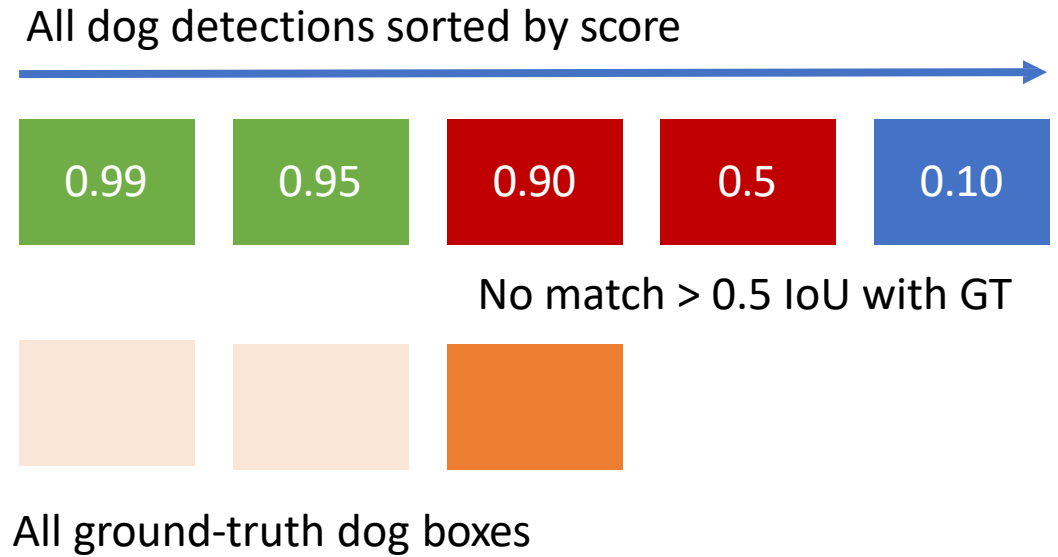
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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



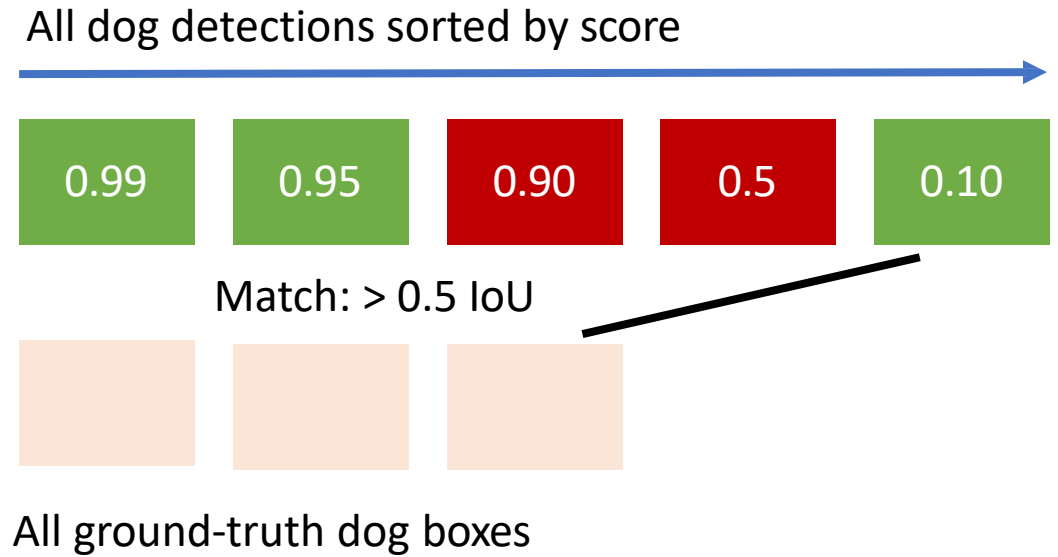
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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



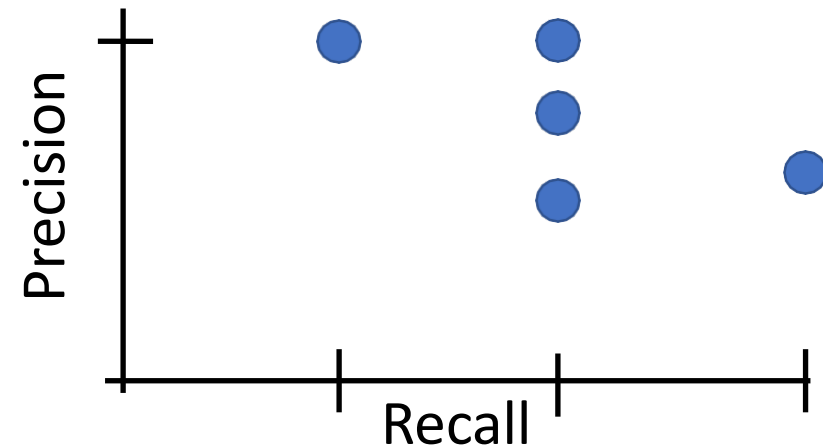
Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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



$$\text{Precision} = 3/5 = 0.6$$

$$\text{Recall} = 3/3 = 1.0$$



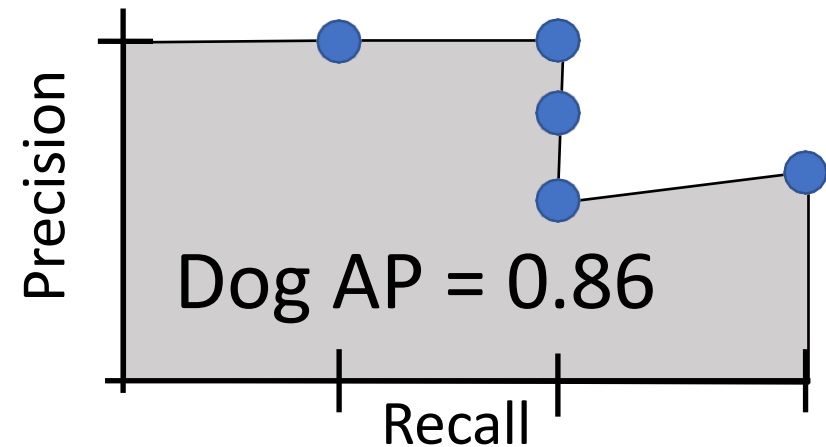
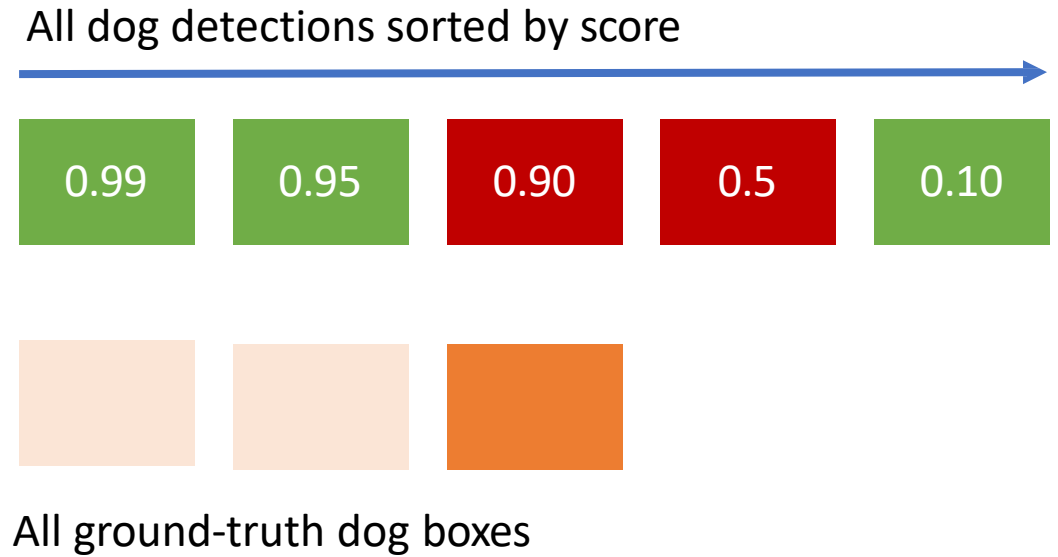
Precision: $\# \text{true detections} / \# \text{detections}$

Recall: $\# \text{true detections} / \# \text{true positives}$

Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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”



Precision: $\frac{\text{\#true detections}}{\text{\#detections}}$
Recall: $\frac{\text{\#true detections}}{\text{\#true positives}}$

Evaluating Object Detectors: Mean Average Precision (mAP)

1. 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
2. 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. 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
2. Mean Average Precision (mAP) = average of AP for each category
3. For “COCO mAP”: Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, ..., 0.95) and take average

$$\text{mAP@0.5} = 0.77$$

$$\text{mAP@0.55} = 0.71$$

$$\text{mAP@0.60} = 0.65$$

...

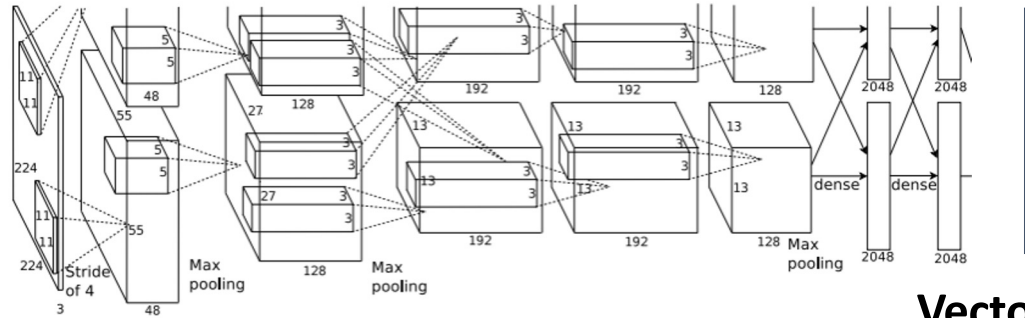
$$\text{mAP@0.95} = 0.2$$

$$\text{COCO mAP} = 0.4$$

Today's class

- How do we measure Object Detection accuracy?
- **Naïve approaches & R-CNN**
- Fast R-CNN
- Region Proposal Network & Faster R-CNN
- Advanced topics:
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

So far: Image Classification



Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores

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

Detecting a single object “What”

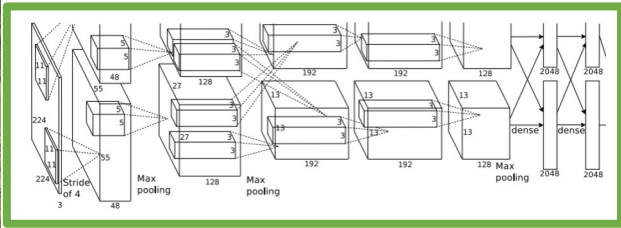
Often pretrained
on ImageNet
(Transfer learning)



[This image is CC0 public domain](#)

Treat localization as a
regression problem!

Problem: Images can have
more than one object!



Vector:
4096

Fully
Connected:
4096 to 1000

Class Scores

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

Correct label:
Cat

**Softmax
Loss**

**Multitask
Loss**

**Weighted
Sum**

Loss

Fully
Connected:
4096 to 4

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

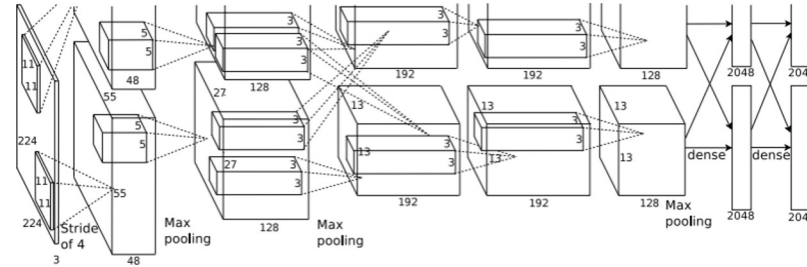
L2 Loss

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

“Where”

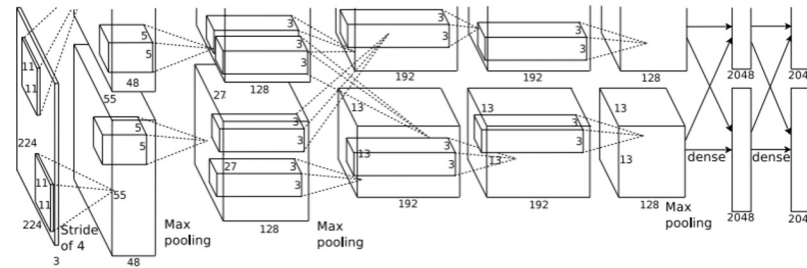
Detecting Multiple Objects

Need different numbers
of outputs per image



CAT: (x, y, w, h)

4 numbers

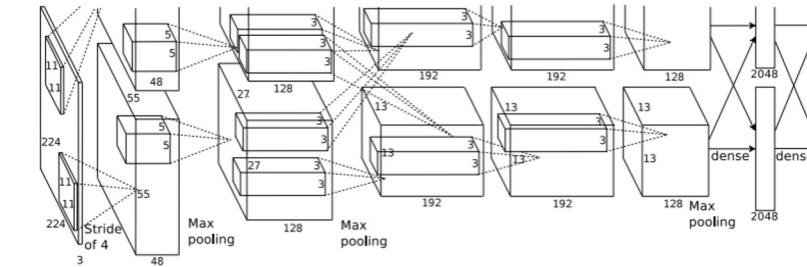


DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)

12 numbers



DUCK: (x, y, w, h)

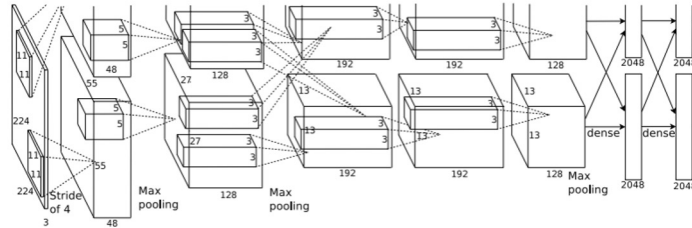
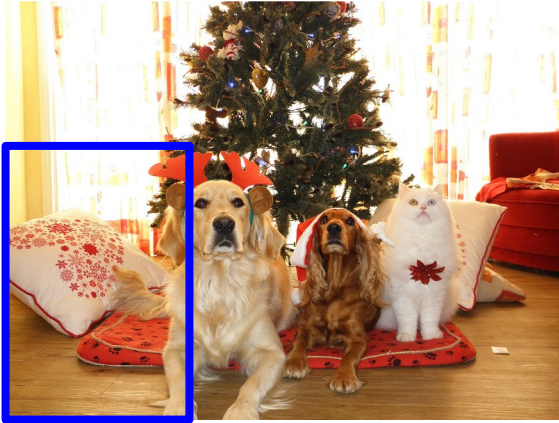
DUCK: (x, y, w, h)

• • •

Many
numbers!

Multiple Objects: Sliding Window

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



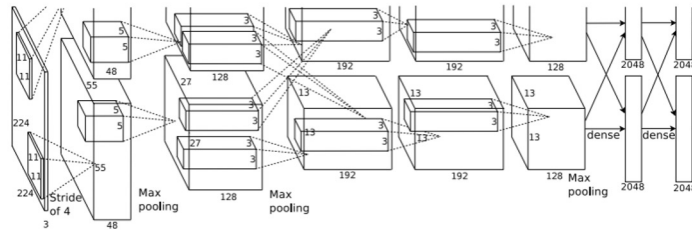
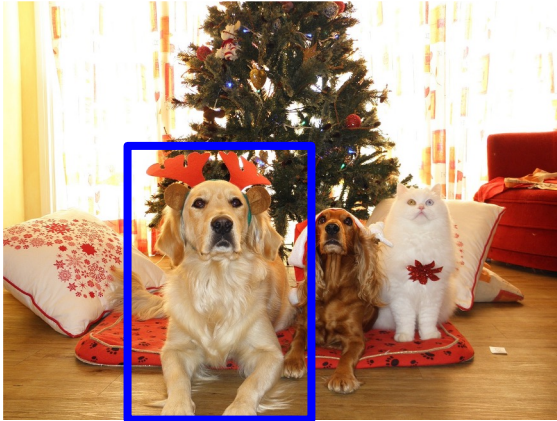
Dog? **NO**

Cat? **NO**

Background? **YES**

Multiple Objects: Sliding Window

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



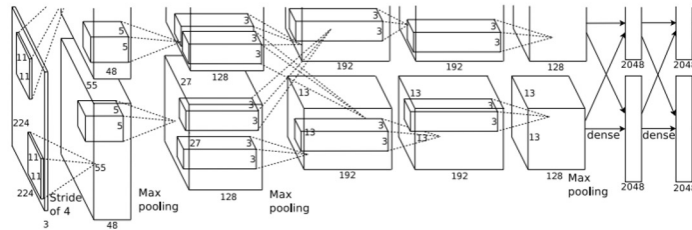
Dog? **YES**

Cat? **NO**

Background? **NO**

Multiple Objects: Sliding Window

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



Dog? YES

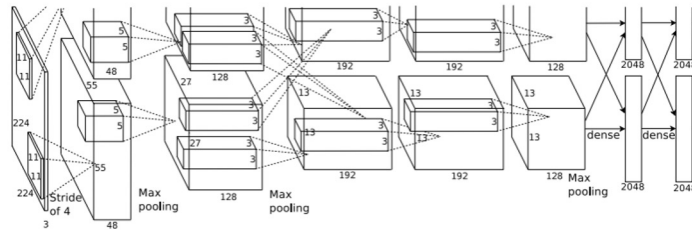
Cat? NO

Background? NO

Multiple Objects: Sliding Window



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

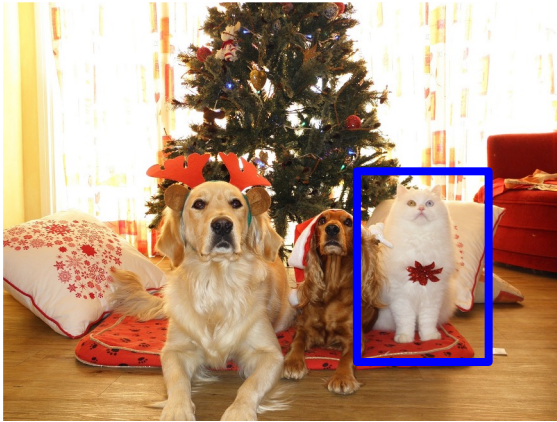


Dog? **NO**
Cat? **YES**
Background? **NO**

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 $H \times W$?

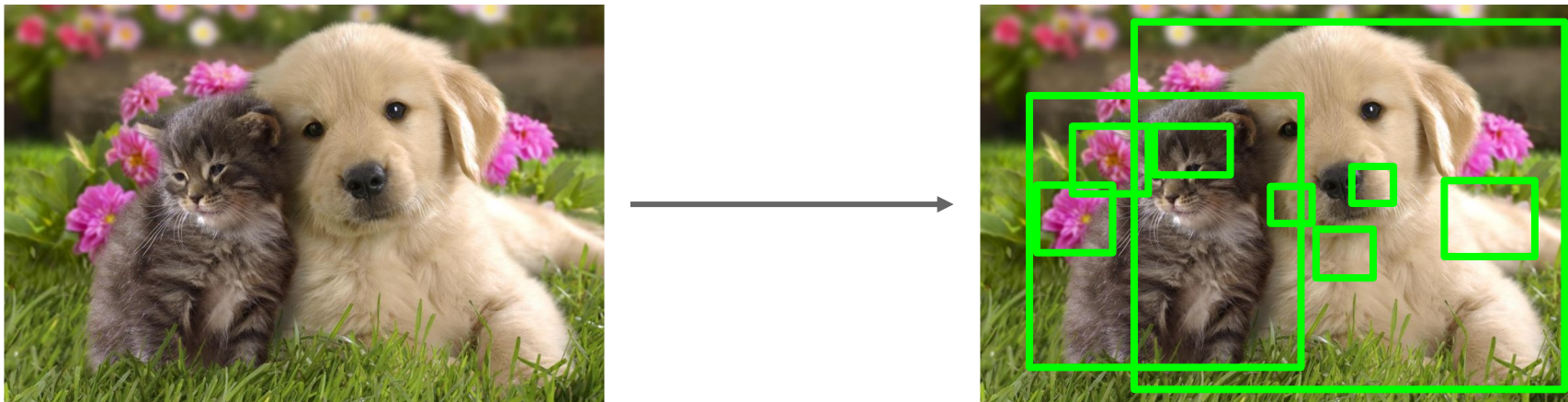
Consider a box of size $h \times w$:
Possible x positions: $W - w + 1$
Possible y positions: $H - h + 1$
Possible positions:
 $(W - w + 1) * (H - h + 1)$

Total possible boxes of different size $h \times w$:

$$\sum_{h=1}^H \sum_{w=1}^W (W - w + 1)(H - h + 1)$$
$$= \frac{H(H+1)}{2} \frac{W(W+1)}{2}$$

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 algorithm** gives 2000 region proposals in a few seconds on CPU



R-CNN: Region-Based CNN

Input
image



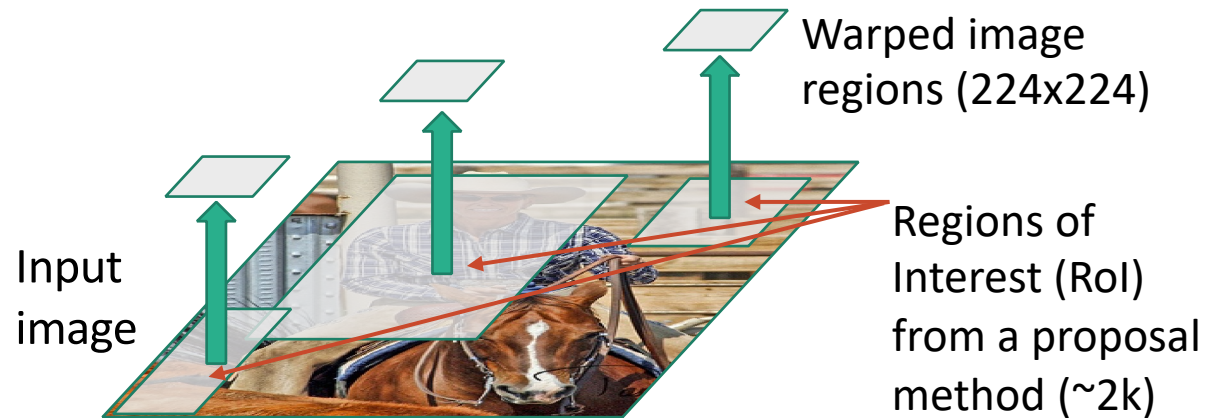
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN: Region-Based CNN



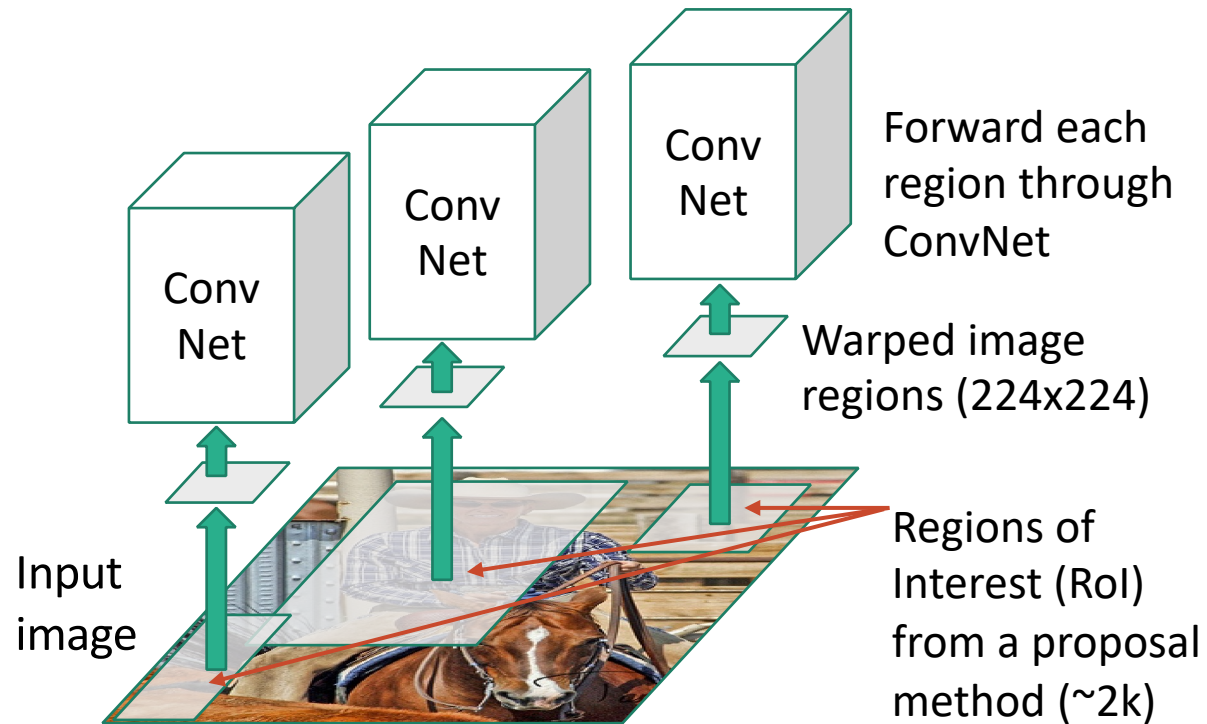
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN: Region-Based CNN



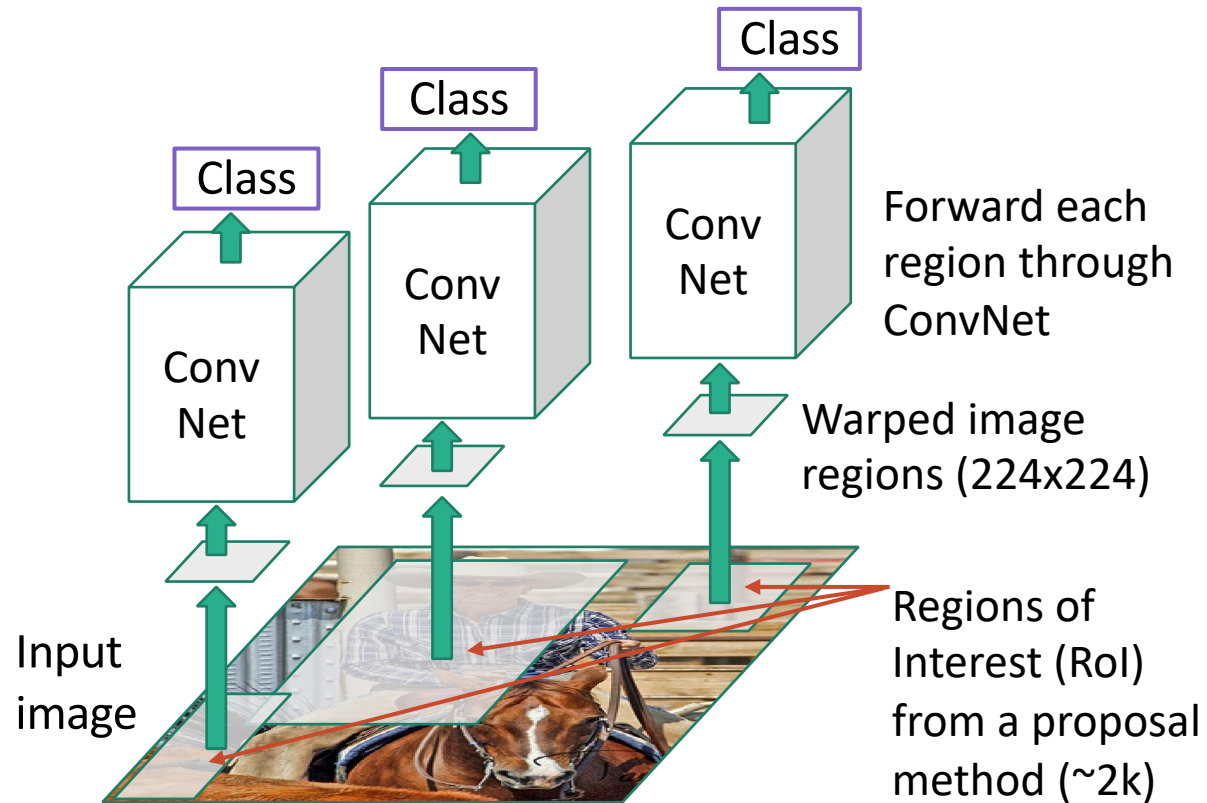
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN: Region-Based CNN

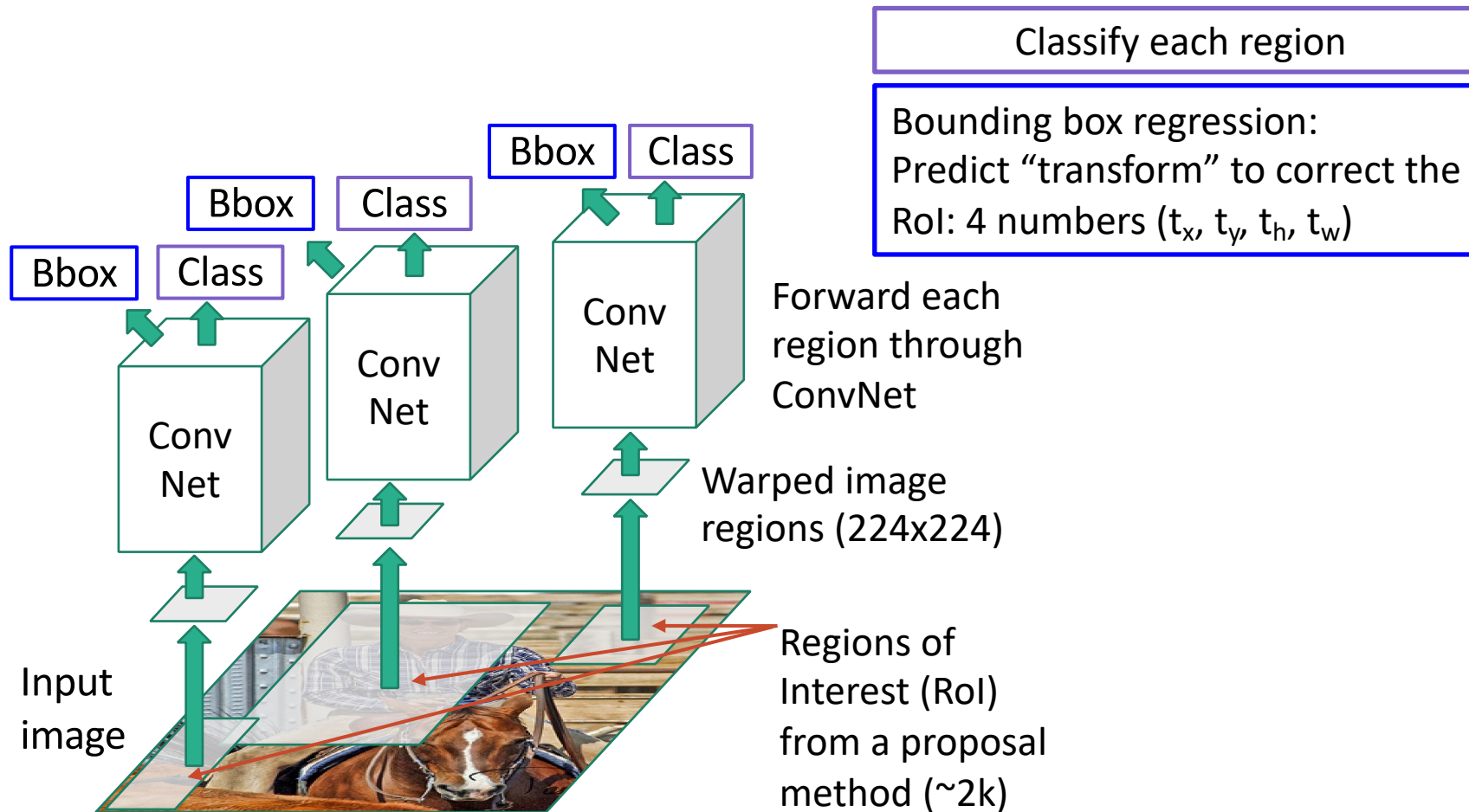


R-CNN: Region-Based CNN

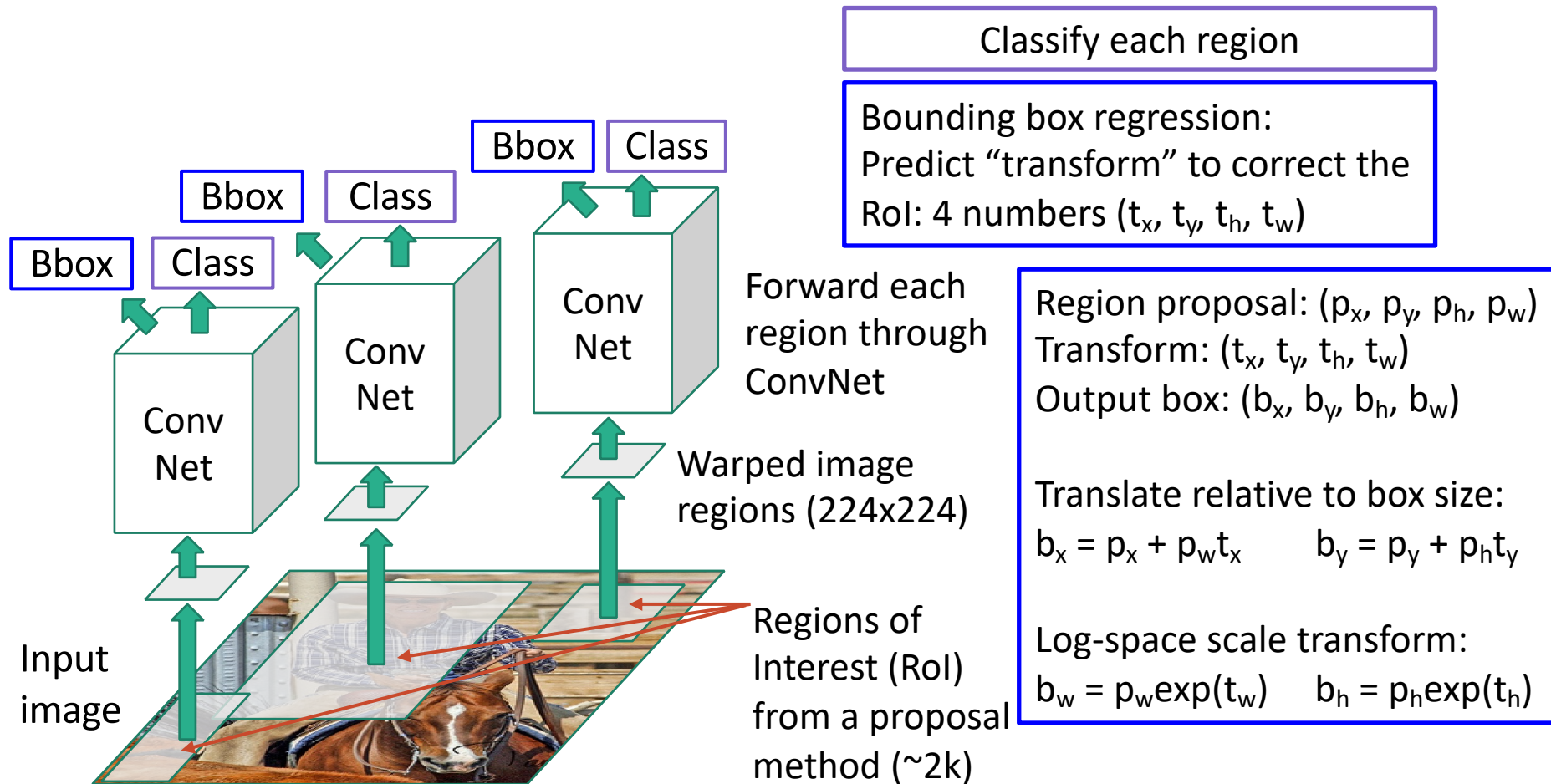
Classify each region



R-CNN: Region-Based CNN

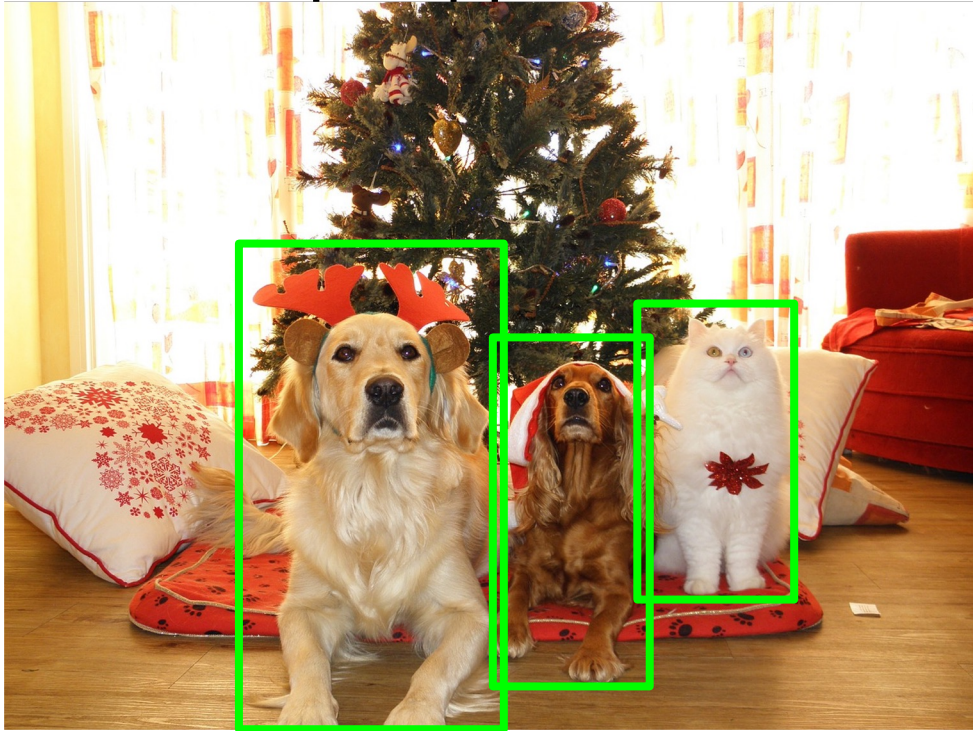


R-CNN: Region-Based CNN



R-CNN

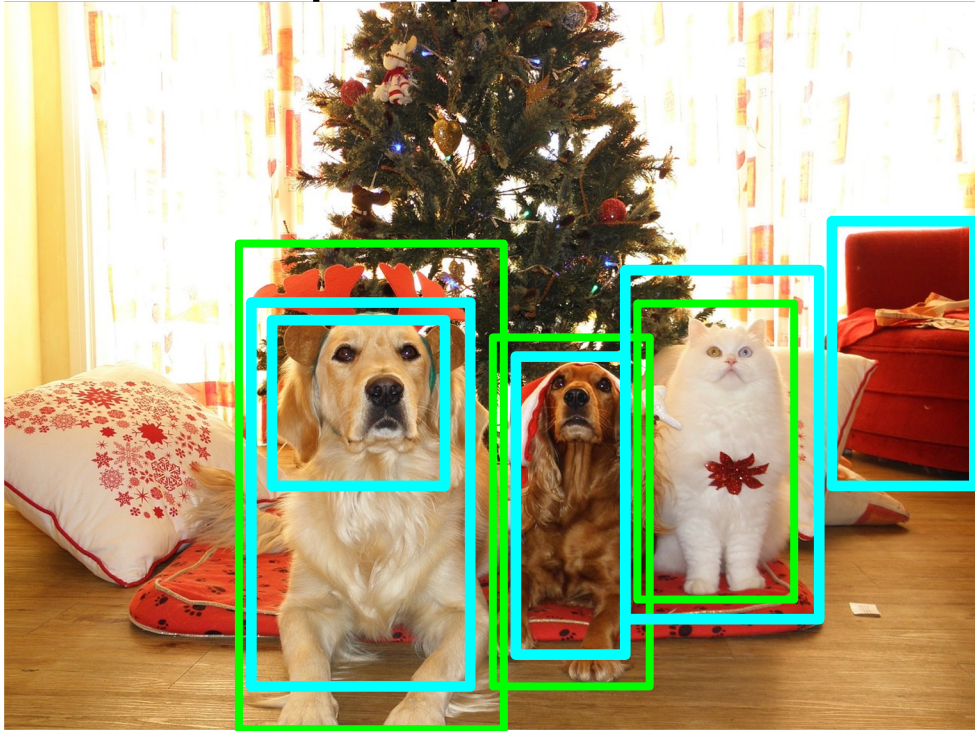
Training



Ground-Truth boxes

R-CNN

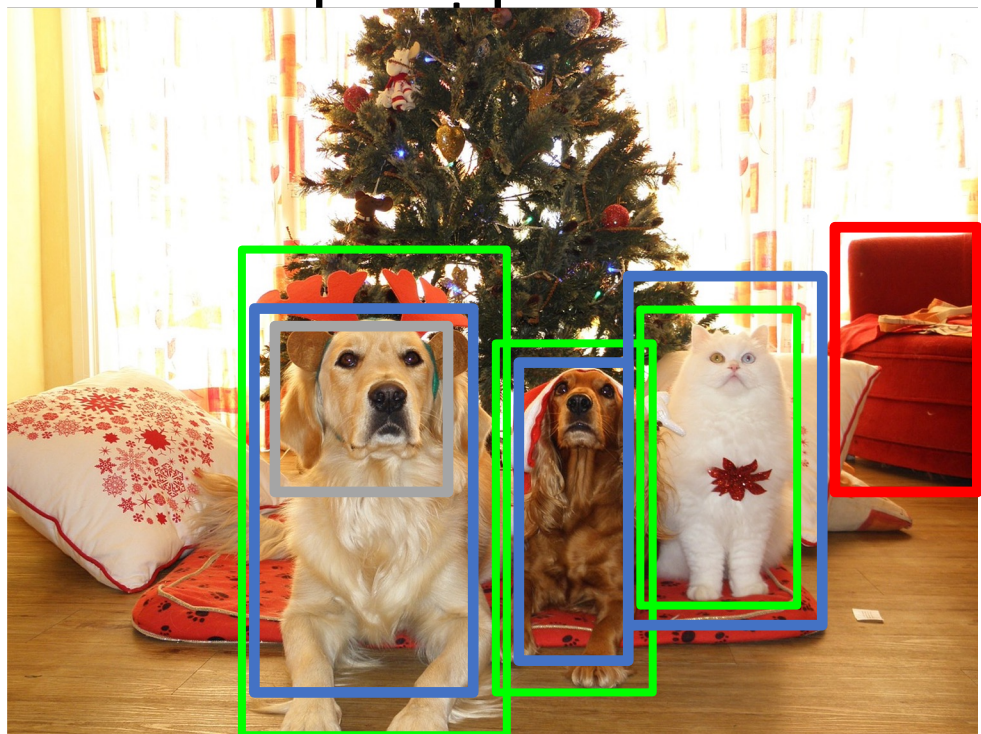
Training



Ground-Truth boxes

Region Proposals

R-CNN Training



GT Boxes

Positive

Neutral

Negative

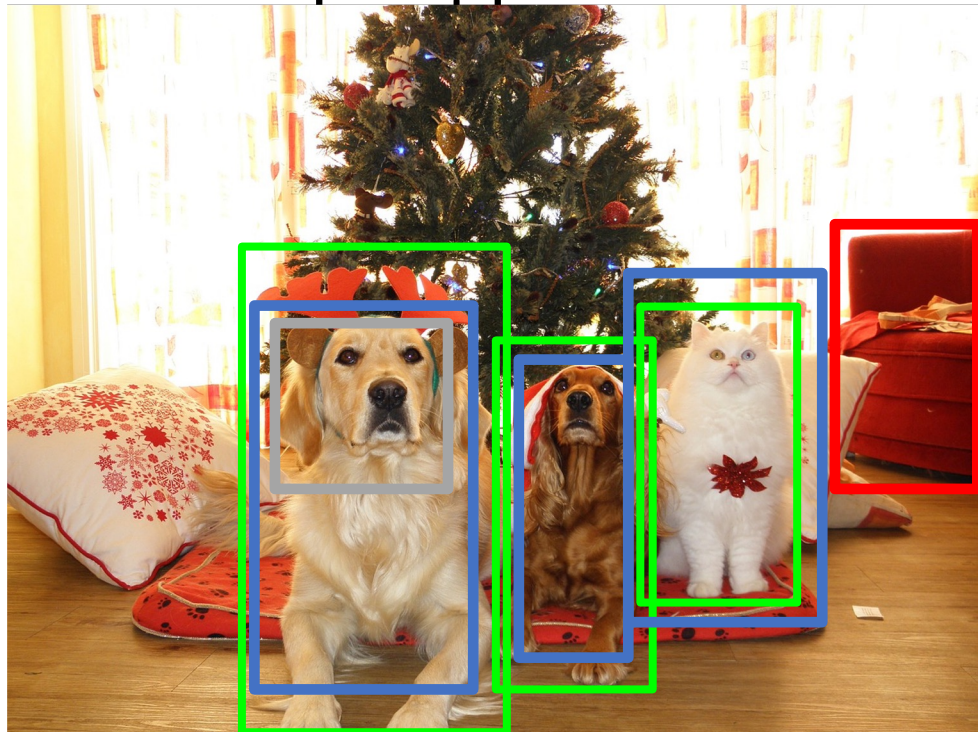
Categorize each region proposal as **positive**, **negative**, or **neutral** based on overlap with ground-truth boxes:

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

R-CNN Training

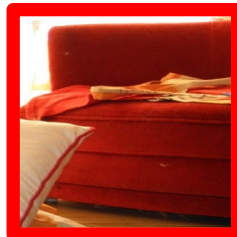
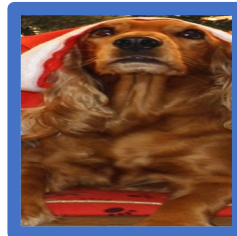
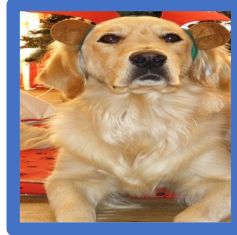


GT Boxes

Positive

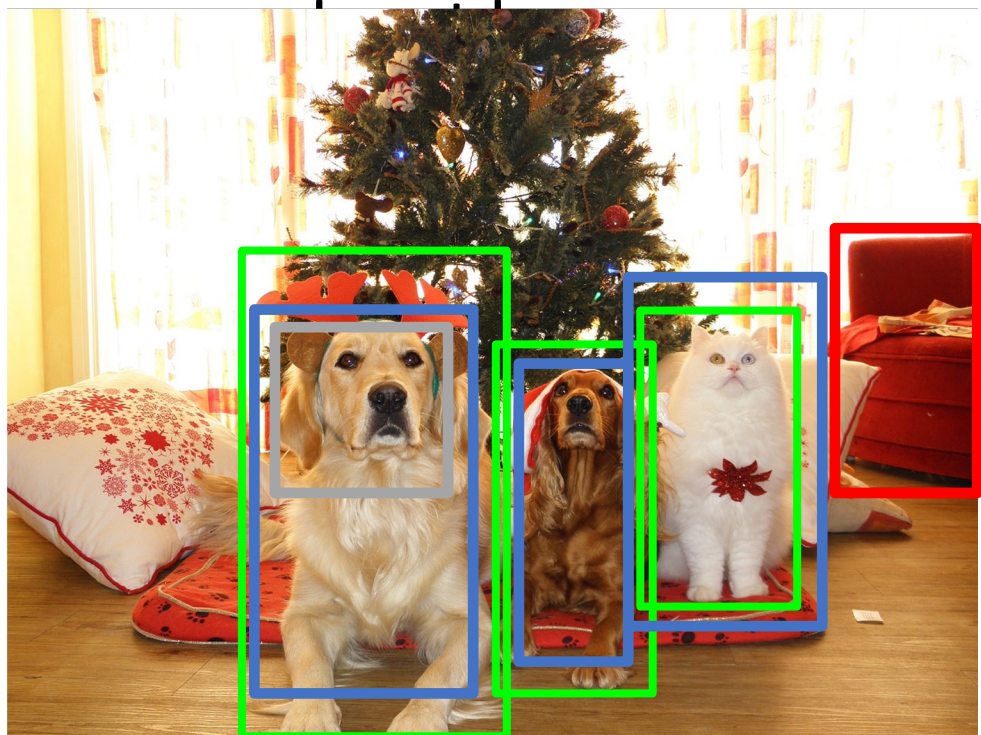
Neutral

Negative



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

R-CNN Training



GT Boxes

Positive

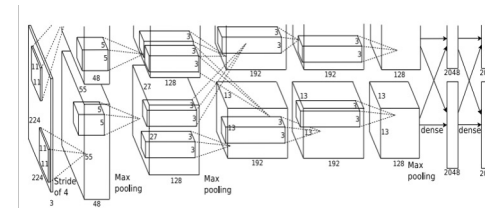
Neutral

Negative

Run each region through CNN

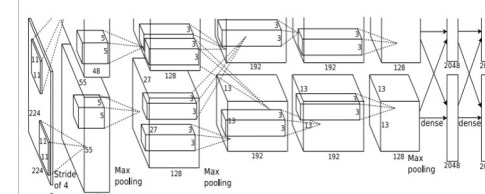
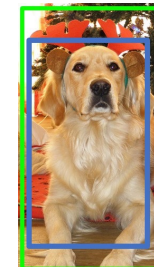
Positive regions: predict class and transform

Negative regions: just predict class



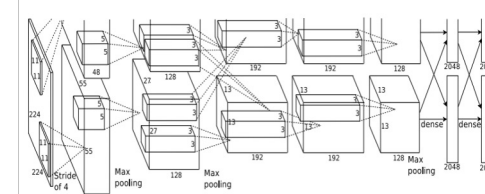
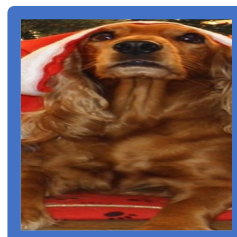
Class target: Dog

Box target: →



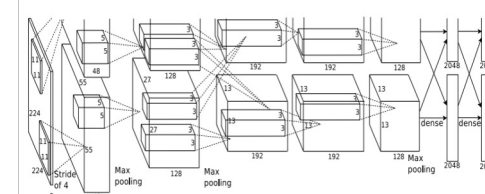
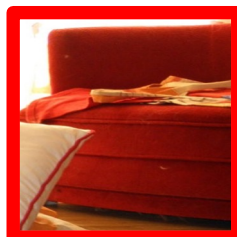
Class target: Cat

Box target: →



Class target: Dog

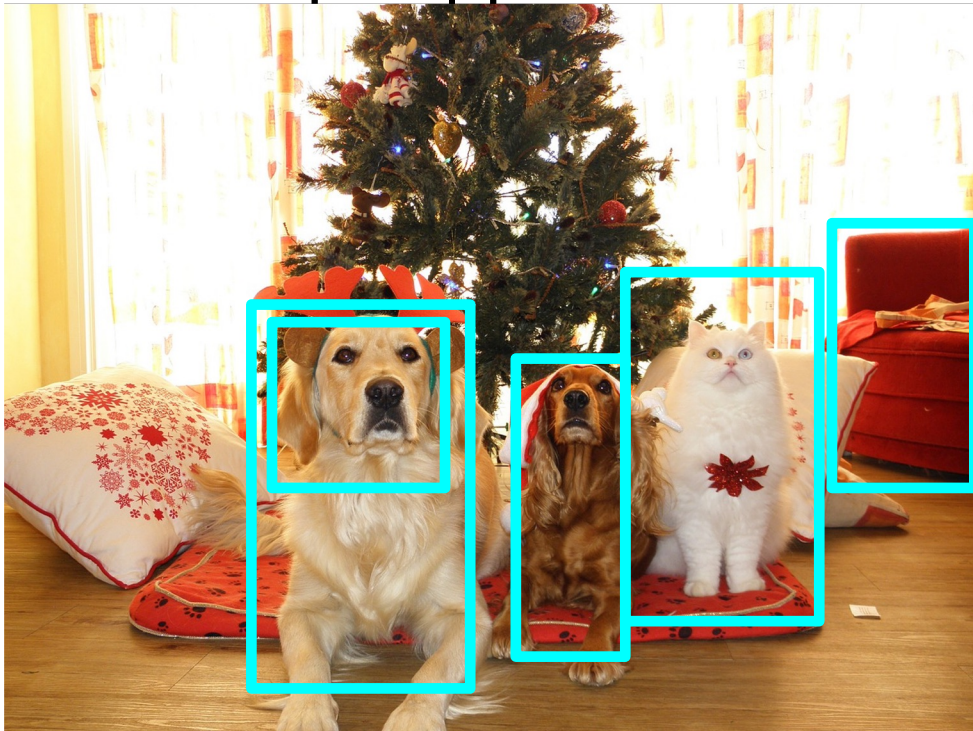
Box target: →



Class target: Background

Box target: None

R-CNN Test-Time



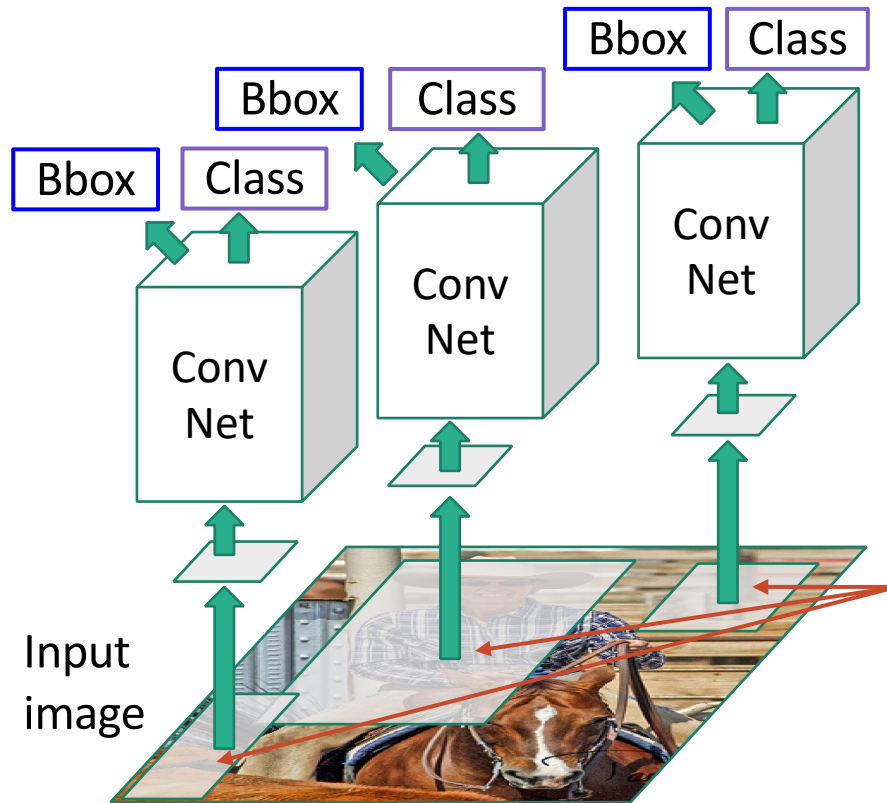
Region Proposals

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
 - Non-maximal suppression
- How to set thresholds?
 - Hyper-parameter

R-CNN: Region-Based CNN



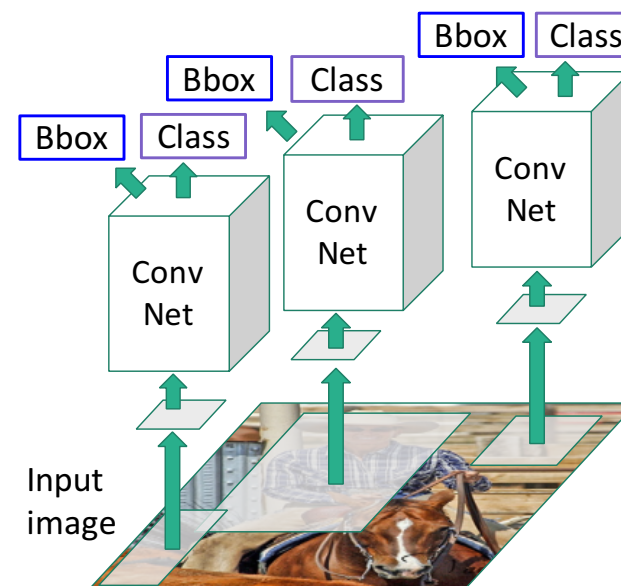
Problem: Very slow!
Need to do ~2k forward passes for each image!

Solution: Run CNN
before cropping!

Today's class

- How do we measure Object Detection accuracy?
- Naïve approaches & R-CNN
- **Fast R-CNN**
- Region Proposal Network & Faster R-CNN
- Advanced topics:
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

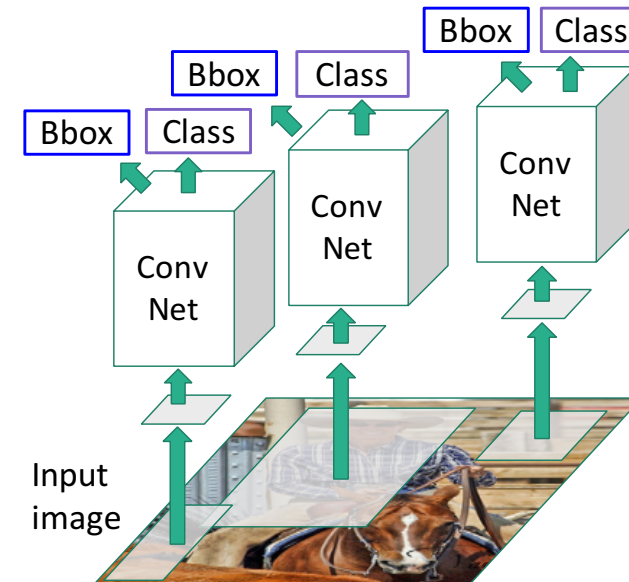
“Slow” R-CNN
Process each region
independently



Fast R-CNN

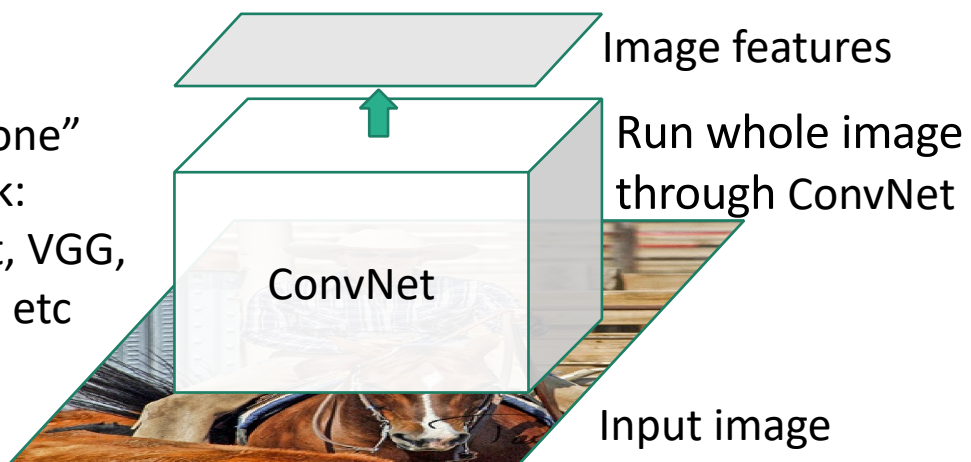


“Slow” R-CNN
Process each region
independently

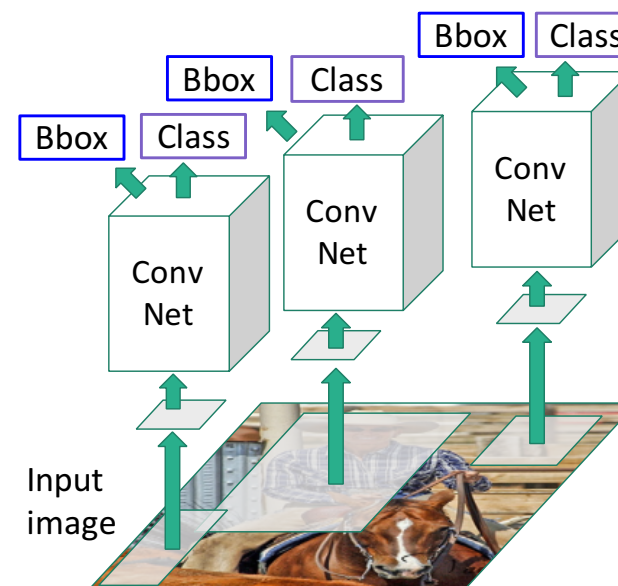


Fast R-CNN

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



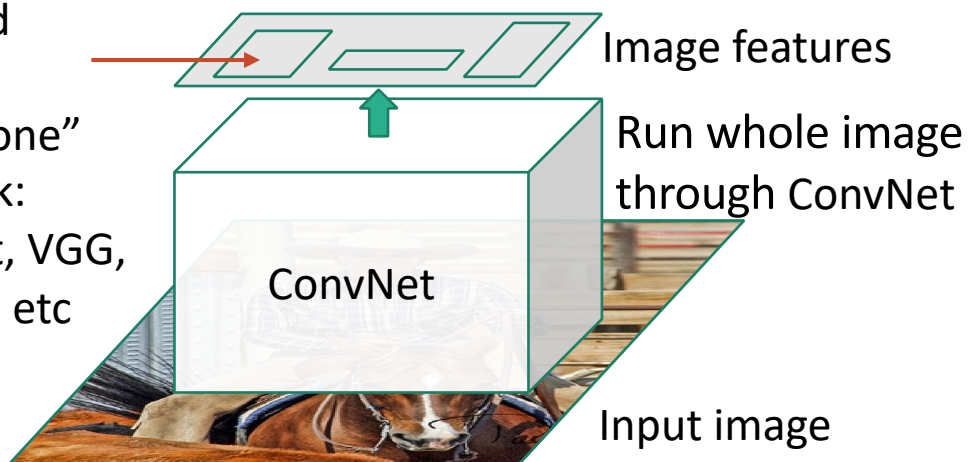
“Slow” R-CNN
Process each region
independently



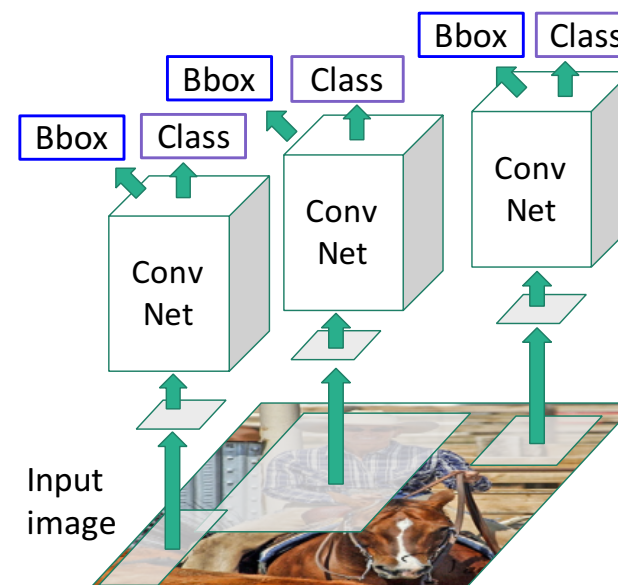
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

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



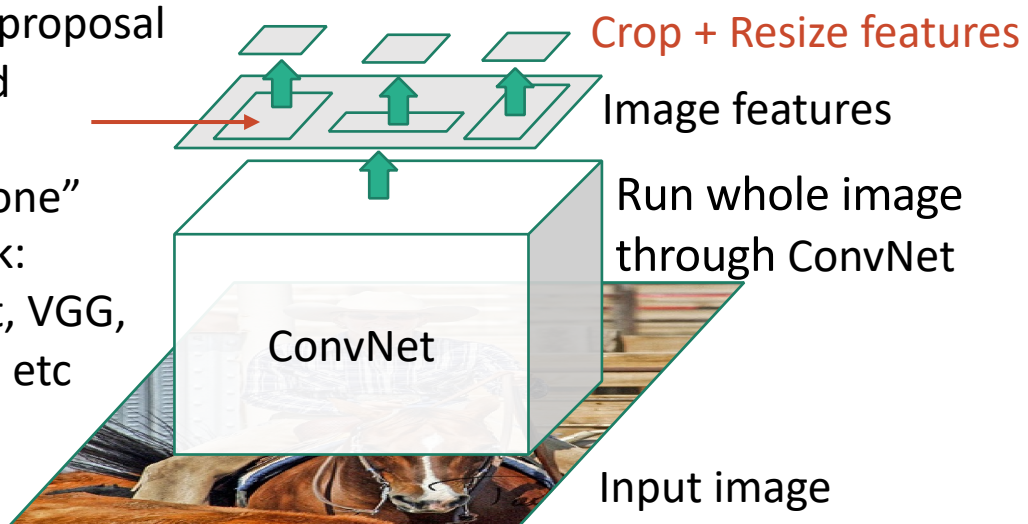
“Slow” R-CNN
Process each region
independently



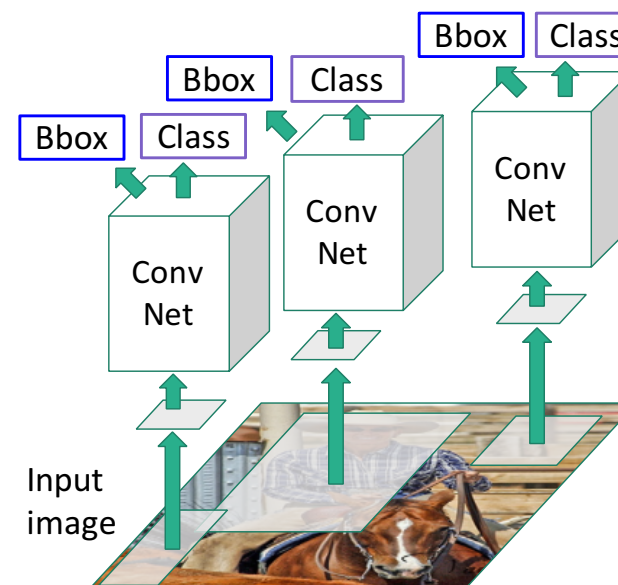
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

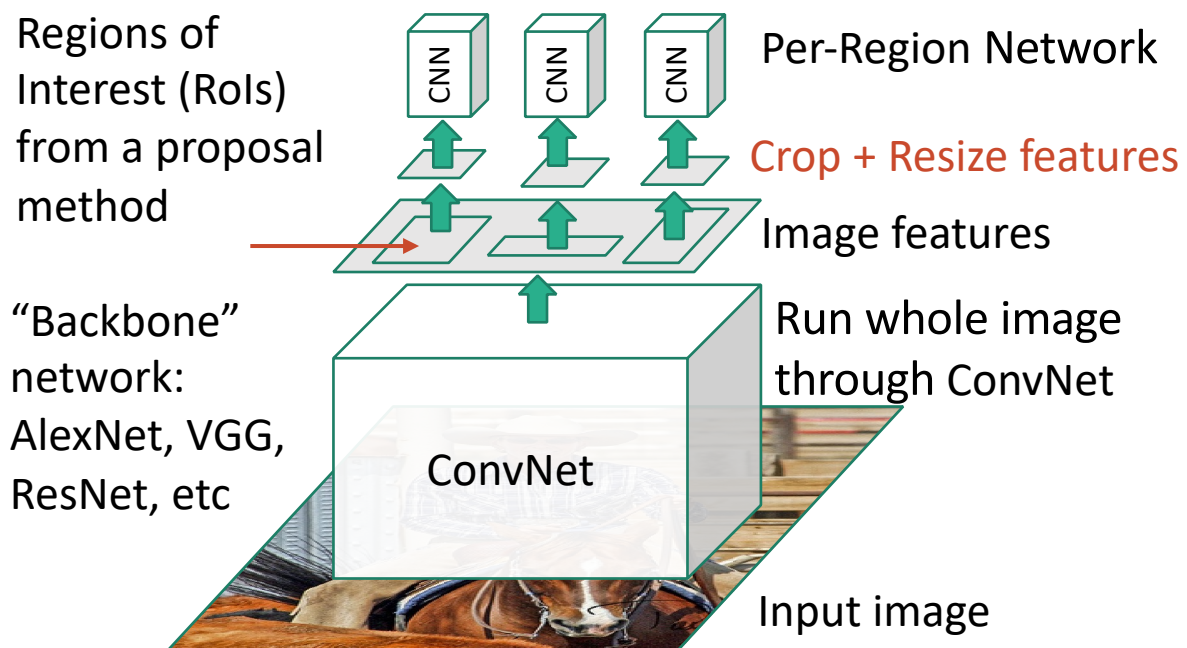
“Backbone” network:
AlexNet, VGG,
ResNet, etc



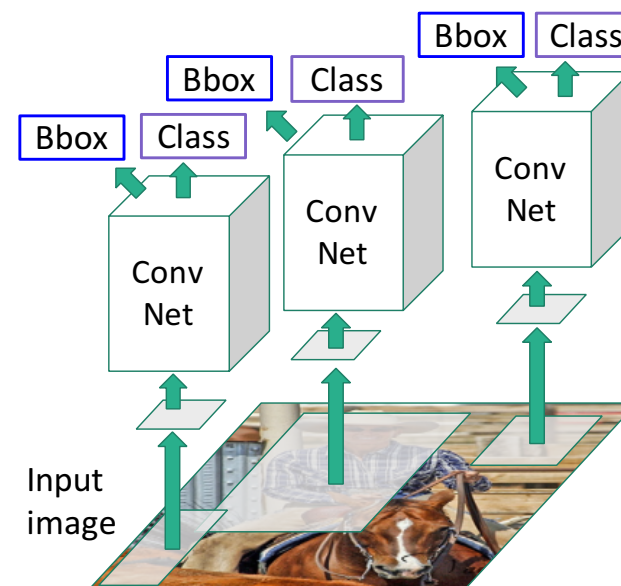
“Slow” R-CNN
Process each region
independently



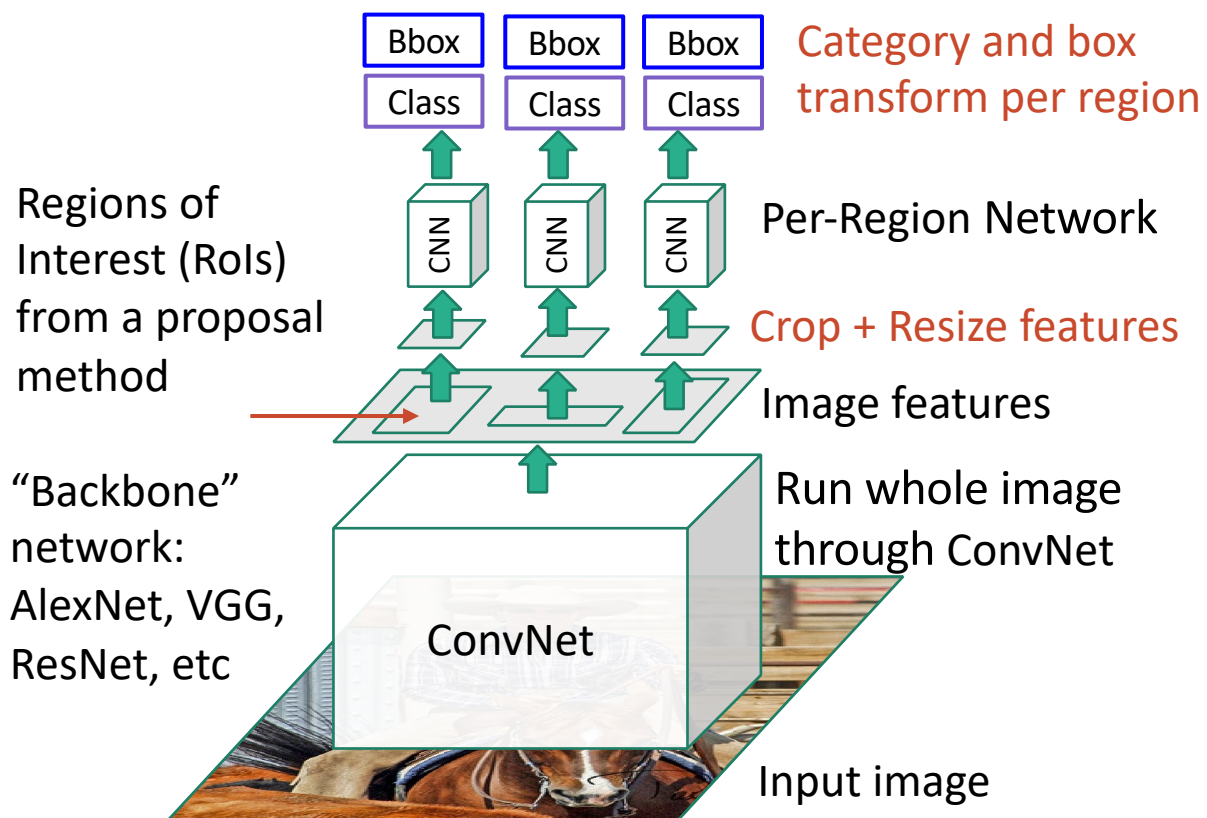
Fast R-CNN



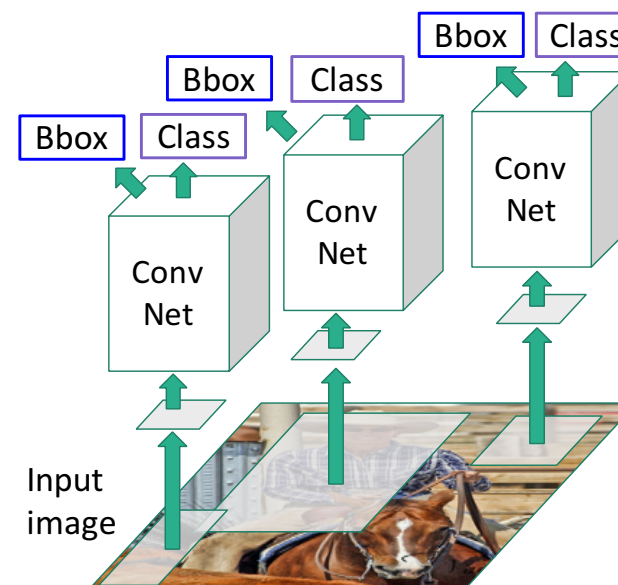
“Slow” R-CNN Process each region independently



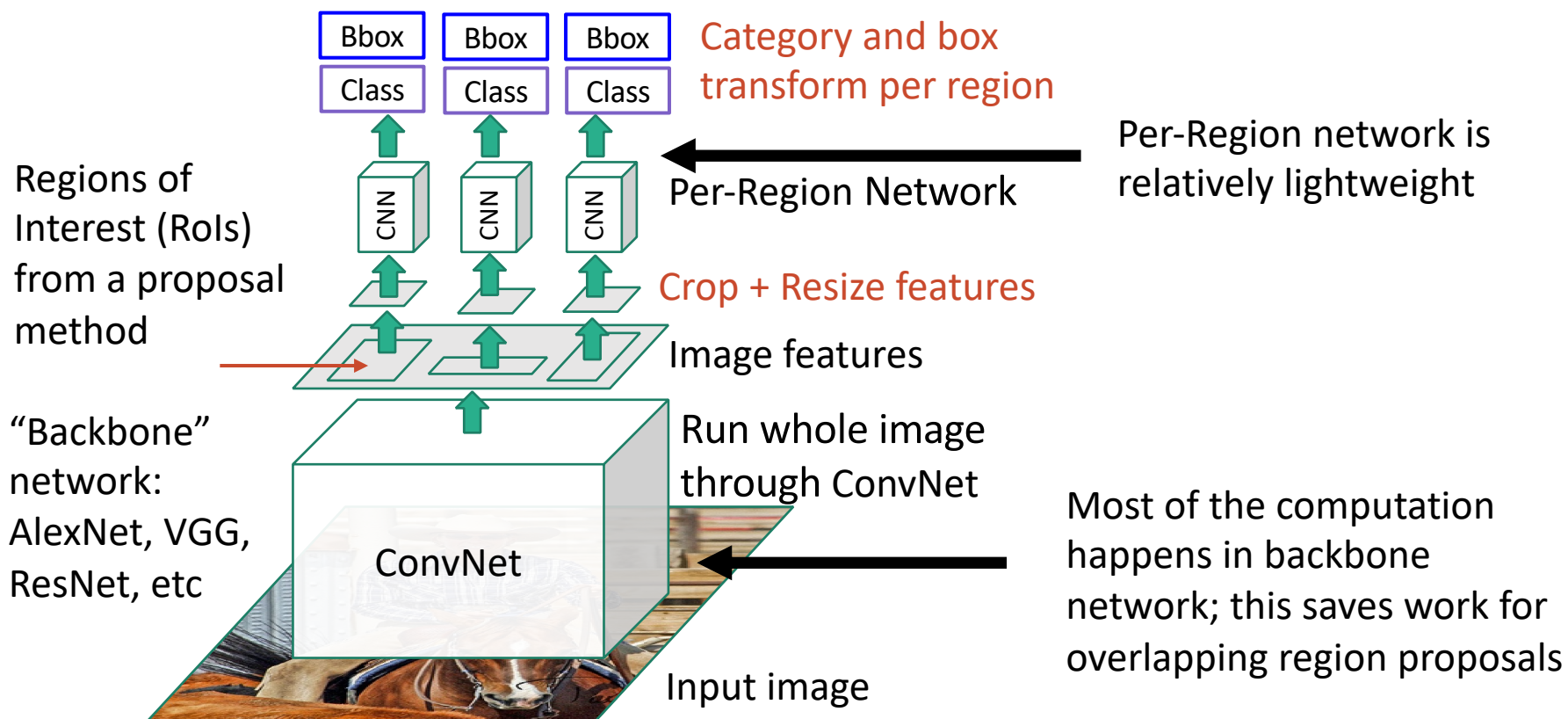
Fast R-CNN



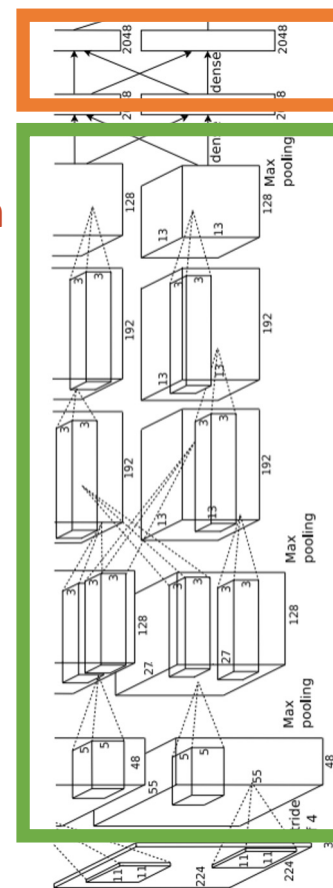
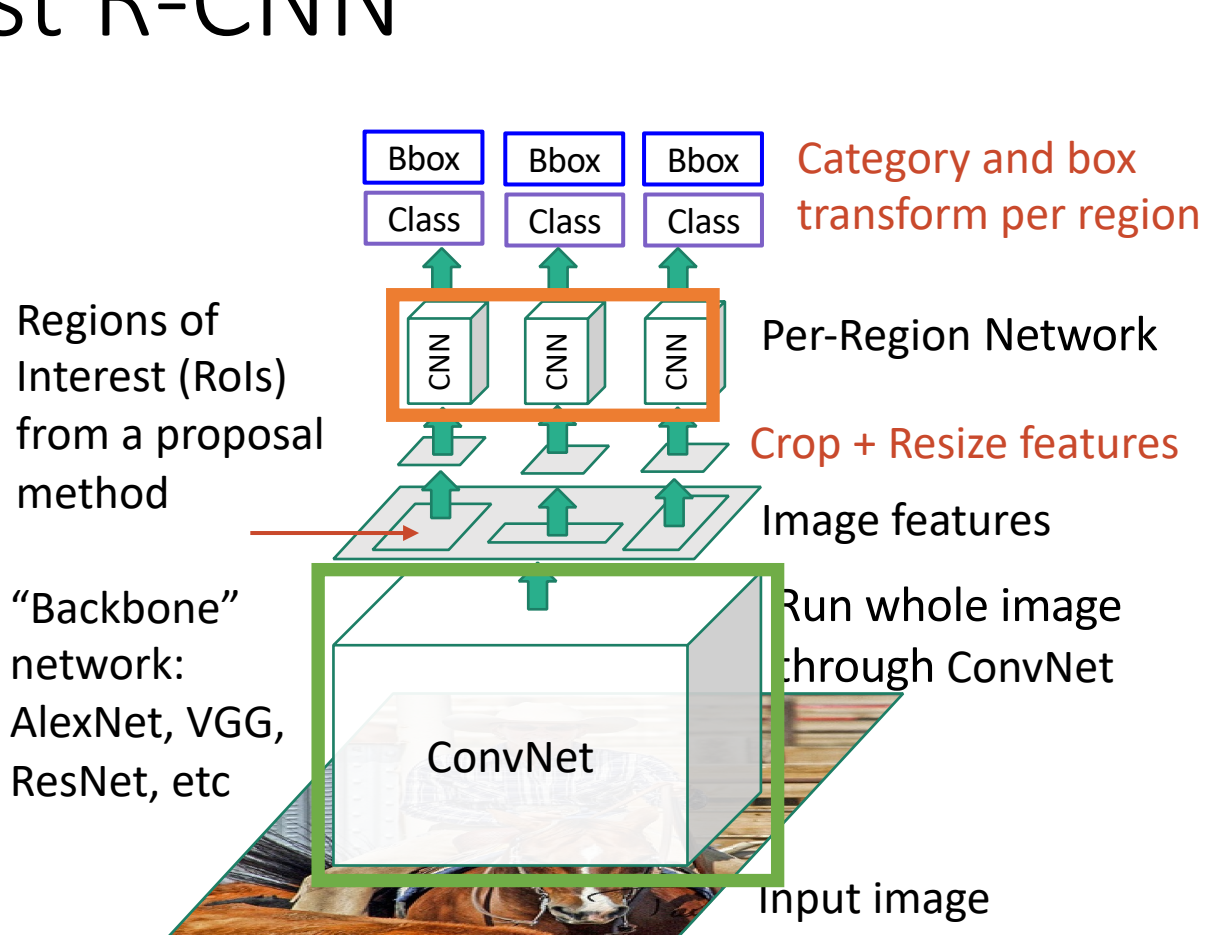
"Slow" R-CNN
Process each region
independently



Fast R-CNN

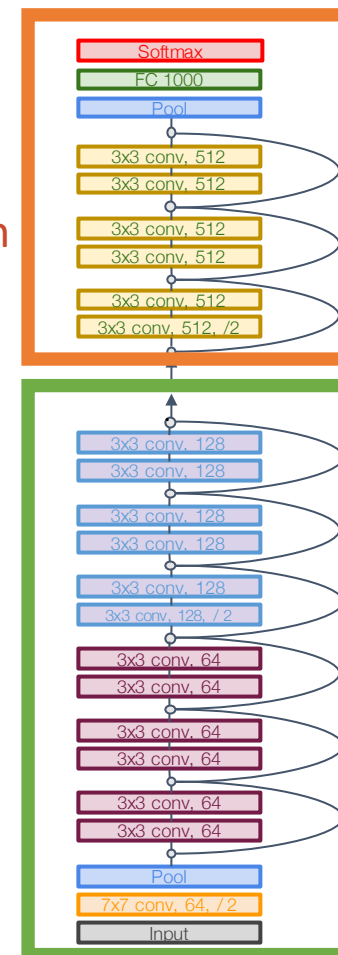
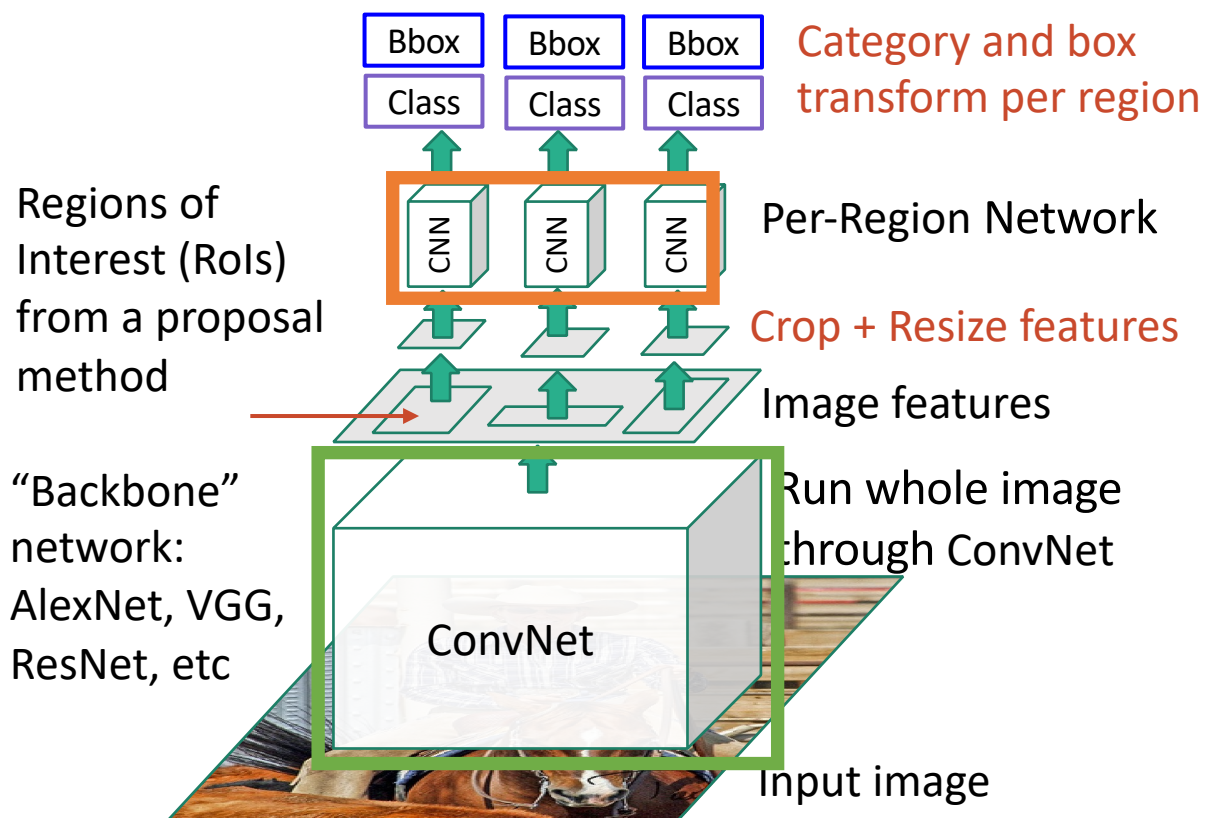


Fast R-CNN



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

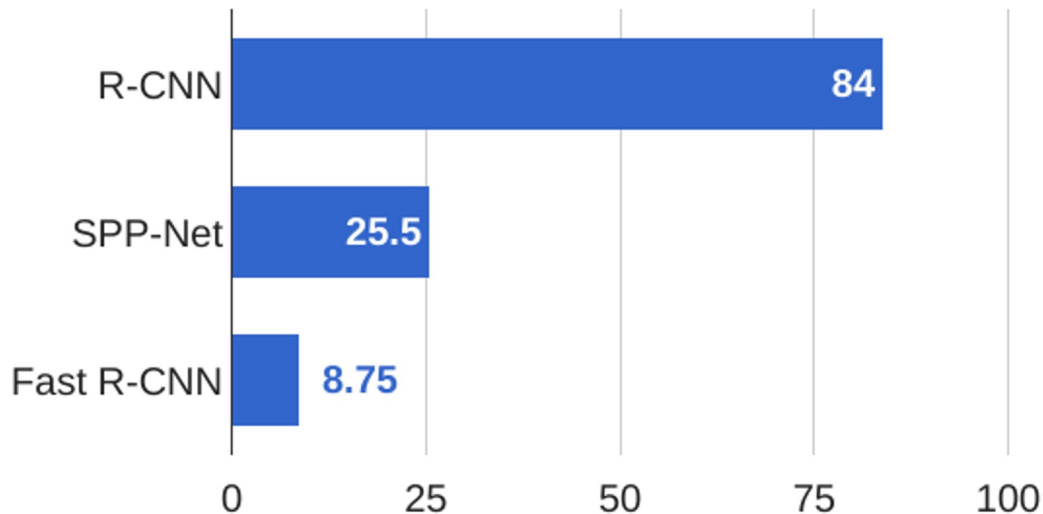
Fast R-CNN



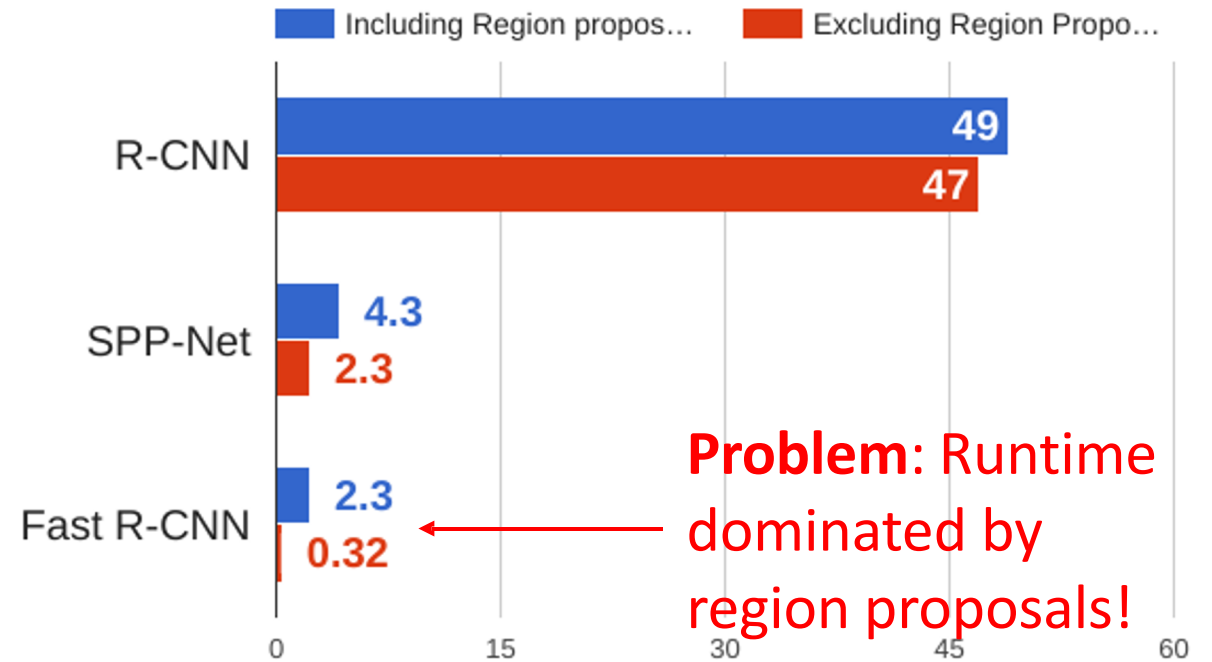
Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Fast R-CNN vs “Slow” R-CNN

Training time (Hours)



Test time (seconds)



Problem: Runtime dominated by region proposals!

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

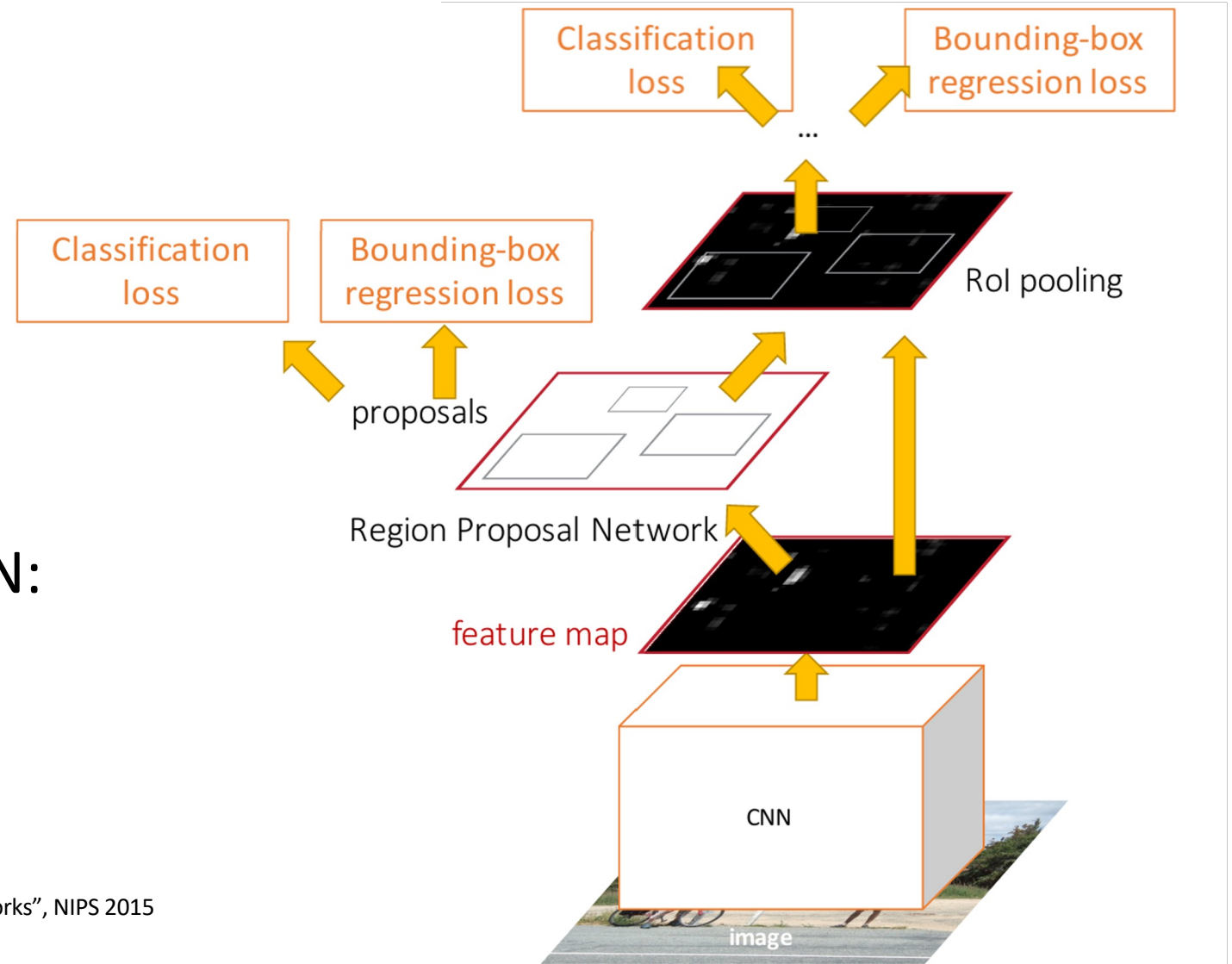
Today's class

- How do we measure Object Detection accuracy?
- Naïve approaches & R-CNN
- Fast R-CNN
- **Region Proposal Network & Faster R-CNN**
- Advanced topics:
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

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



Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image



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

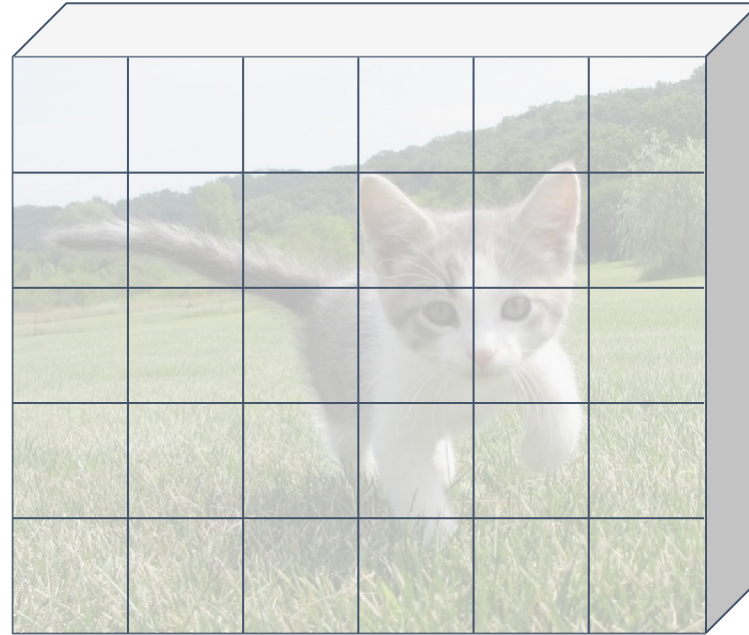


Image features
(e.g. 512 x 5 x 6)

Region Proposal Network (RPN)

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

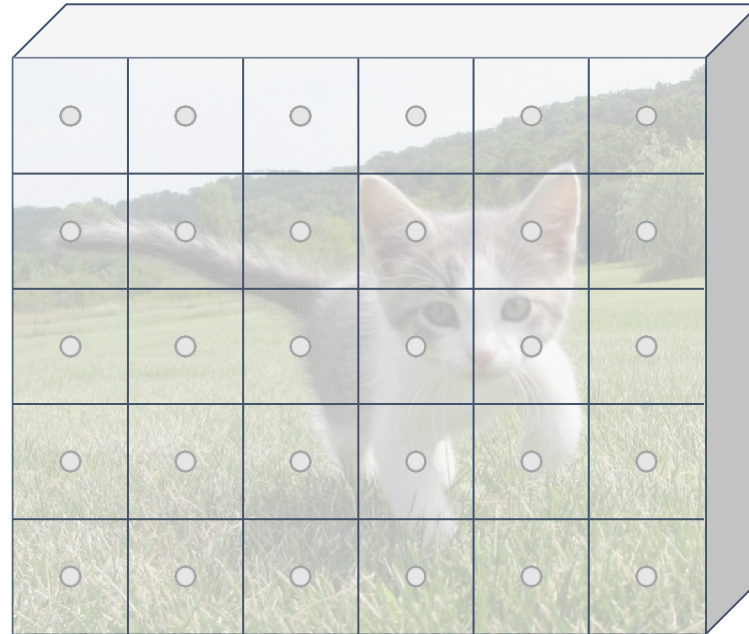


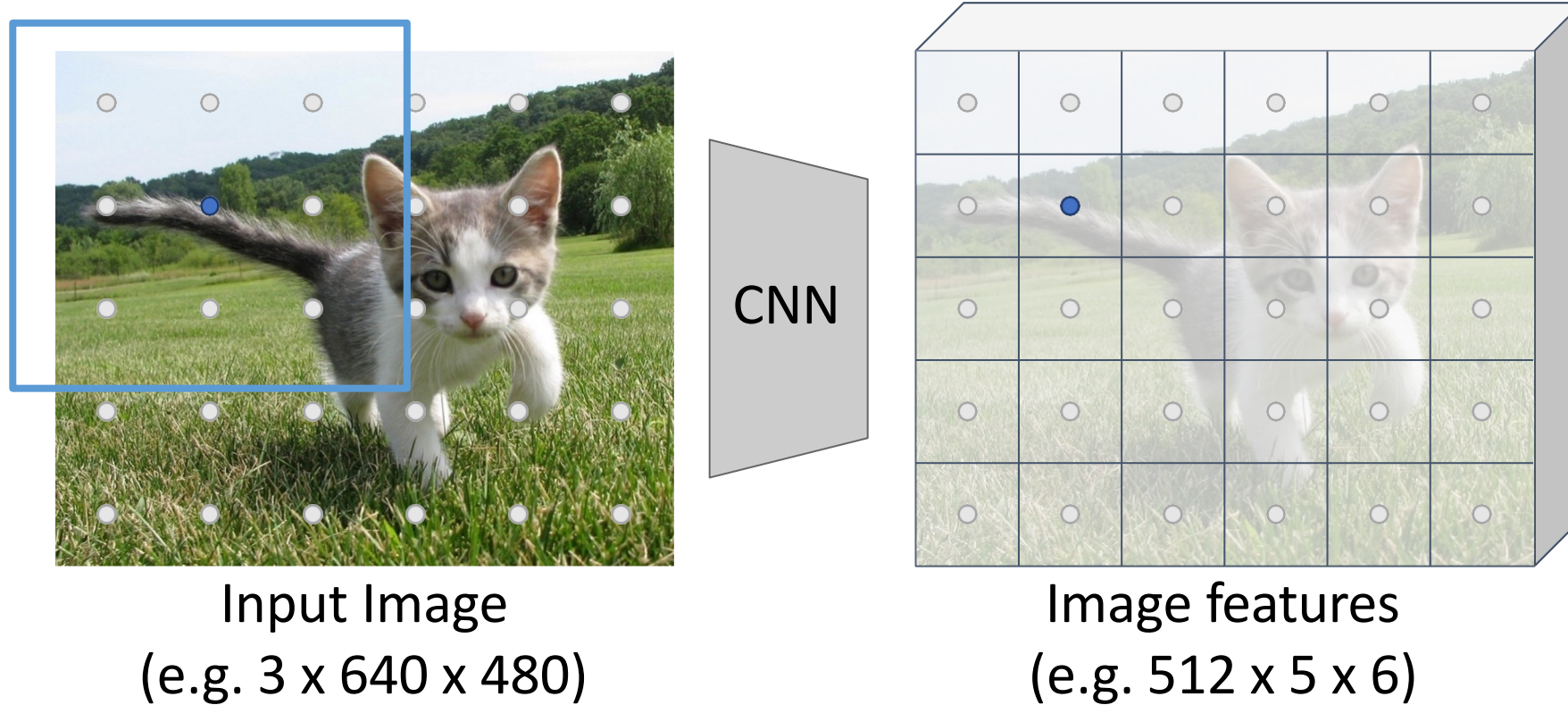
Image features
(e.g. 512 x 5 x 6)

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

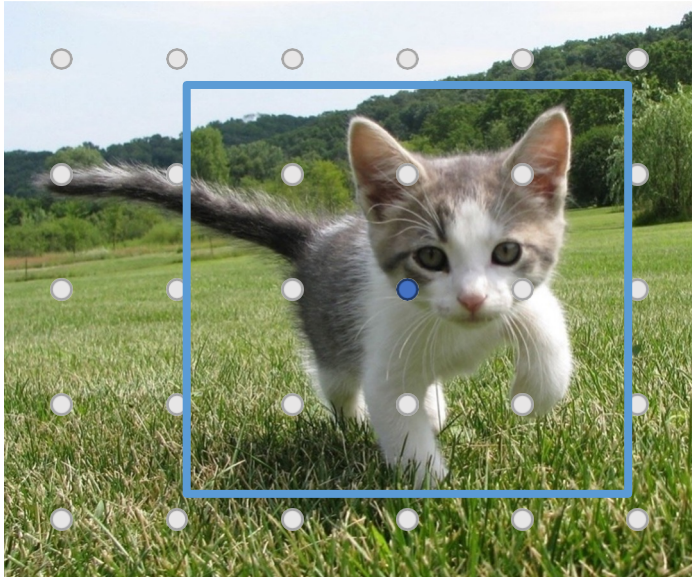
Each feature corresponds
to a point in the input



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



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



Each feature corresponds
to a point in the input

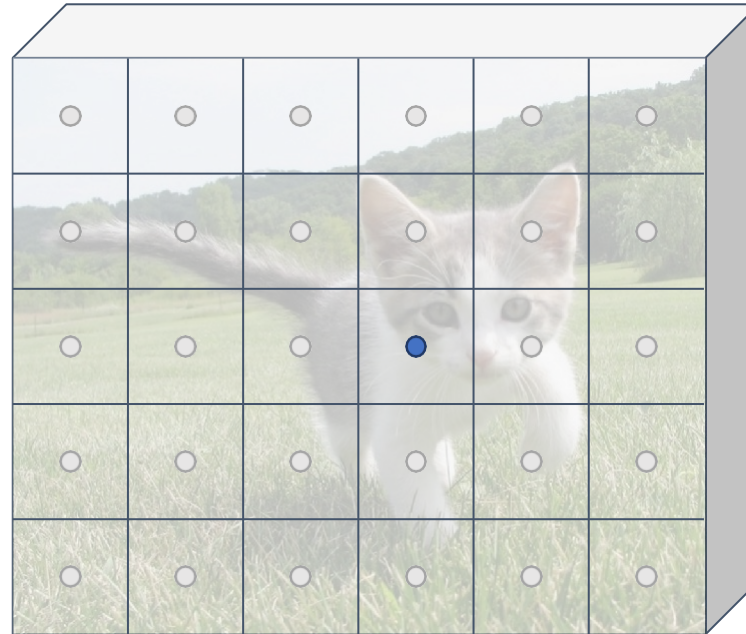
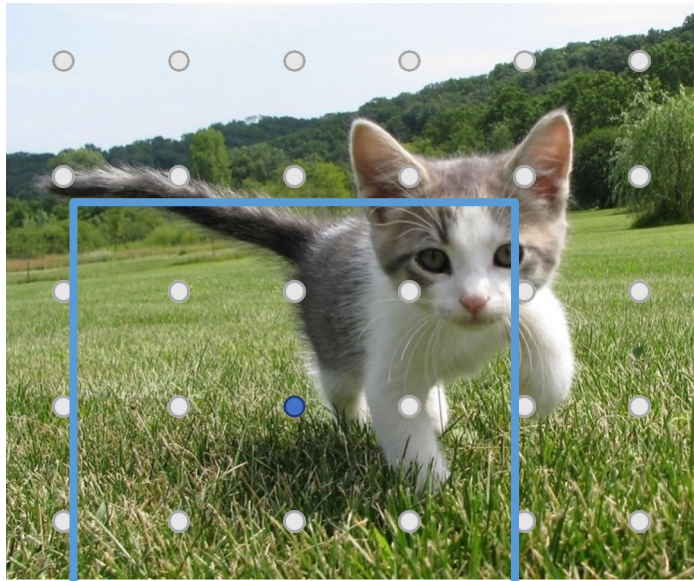


Image features
(e.g. 512 x 5 x 6)

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



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



Each feature corresponds
to a point in the input

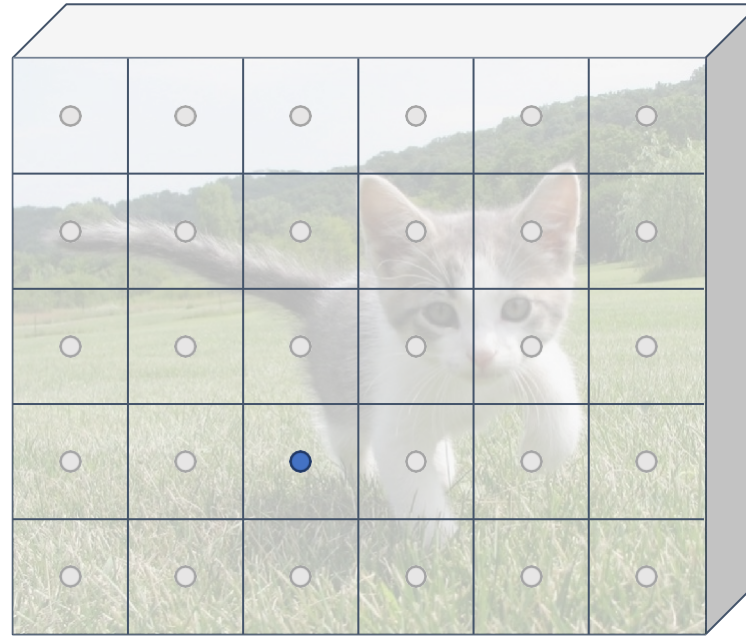
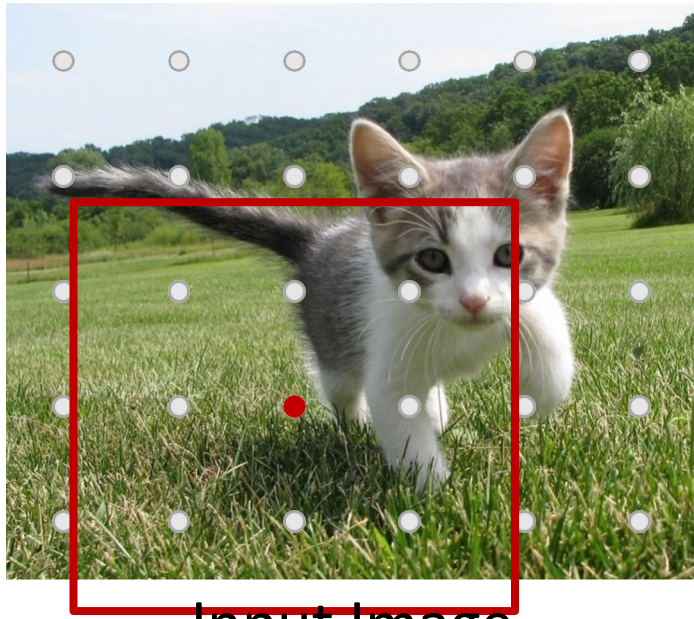


Image features
(e.g. 512 x 5 x 6)

Region Proposal Network (RPN)

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

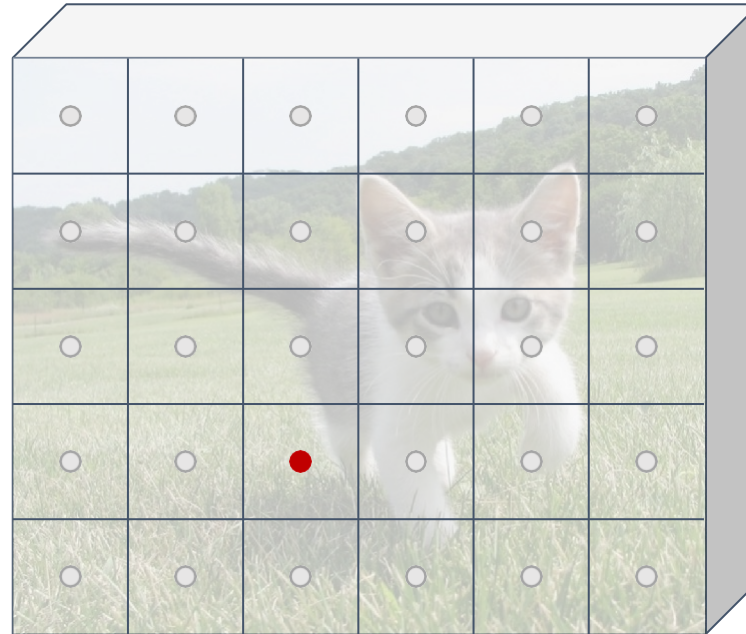


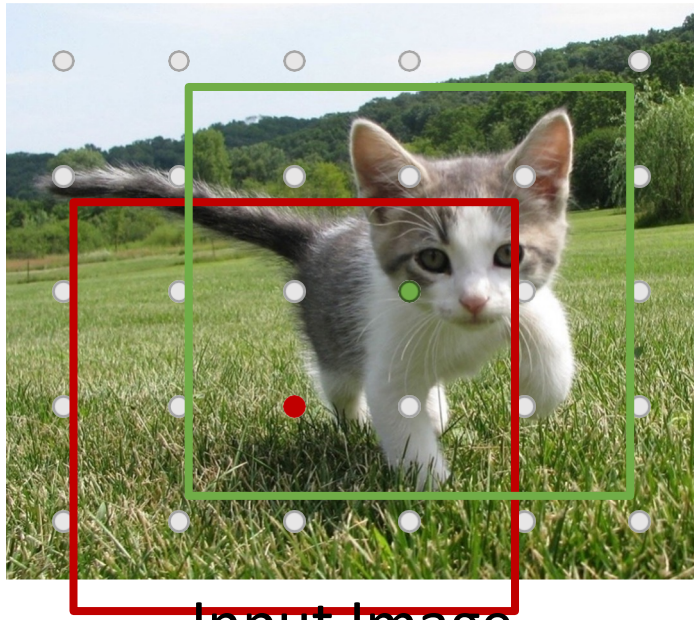
Image features
(e.g. 512 x 5 x 6)

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

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

Region Proposal Network (RPN)

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

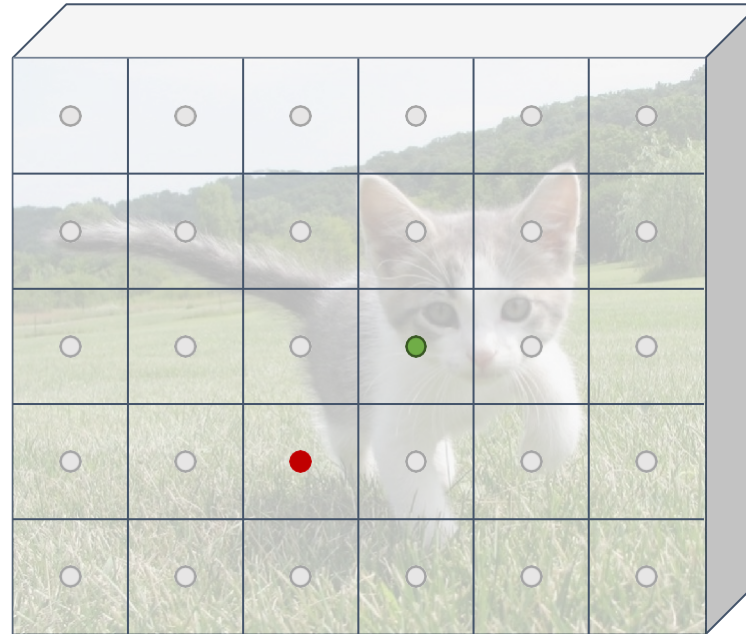


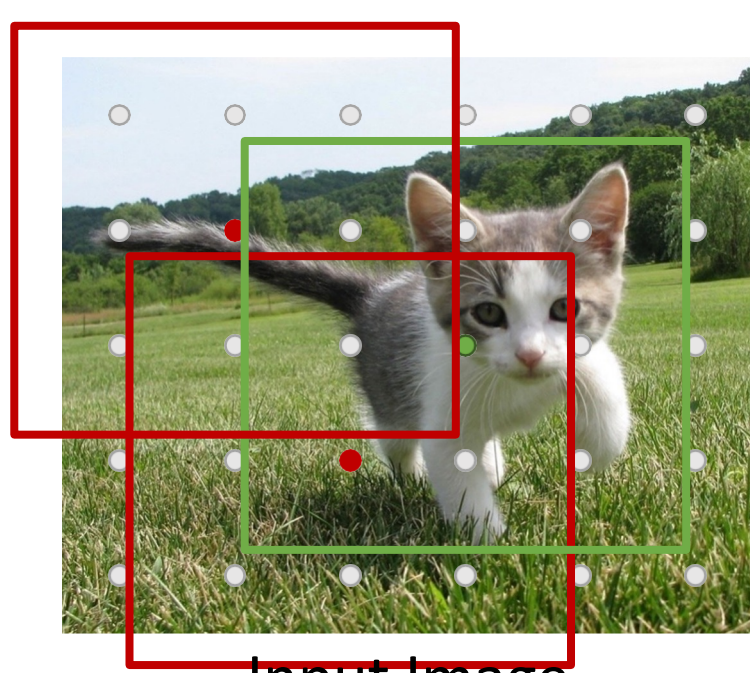
Image features
(e.g. 512 x 5 x 6)

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

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

Region Proposal Network (RPN)

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

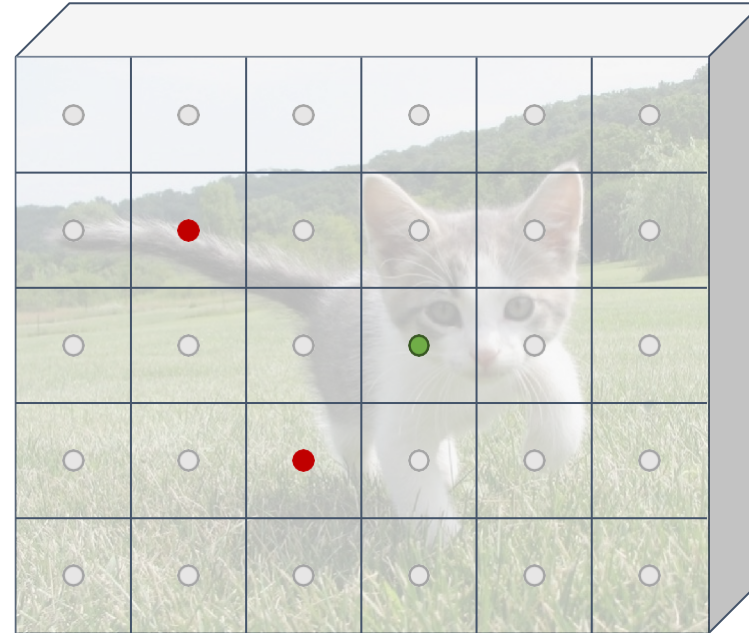


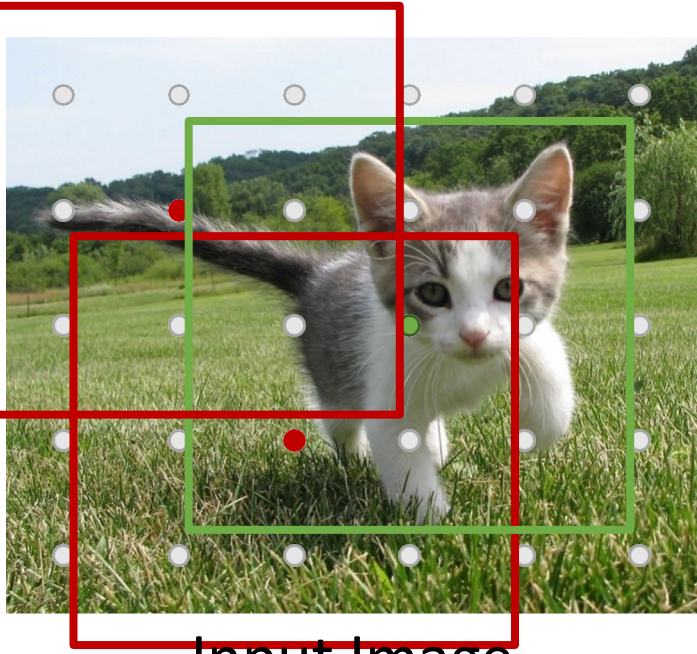
Image features
(e.g. 512 x 5 x 6)

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

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

Region Proposal Network (RPN)

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

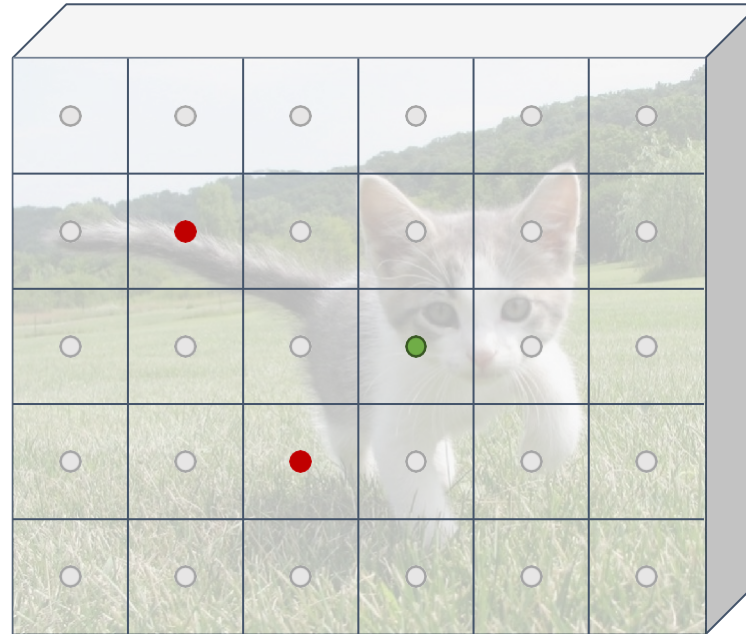


Image features
(e.g. 512 x 5 x 6)

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

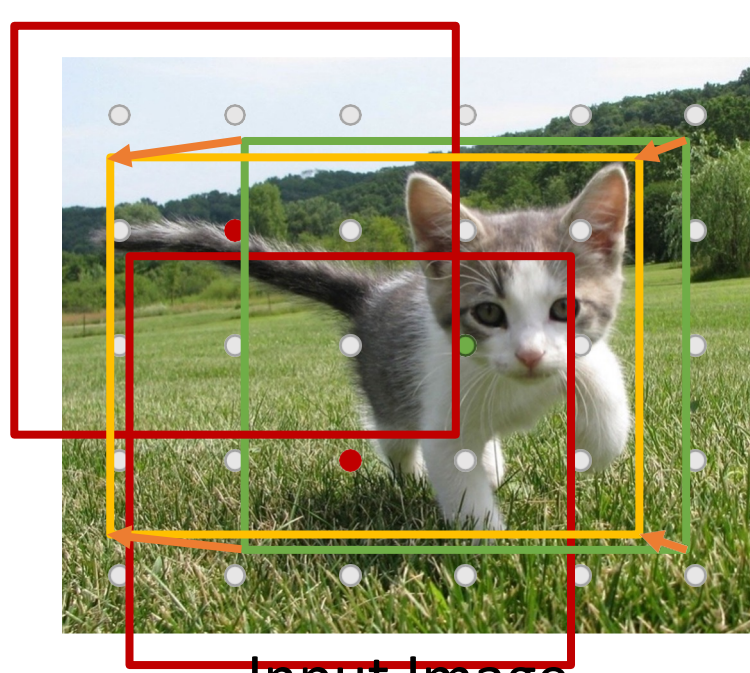


Anchor is
object?
2 x 5 x 6

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

Region Proposal Network (RPN)

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

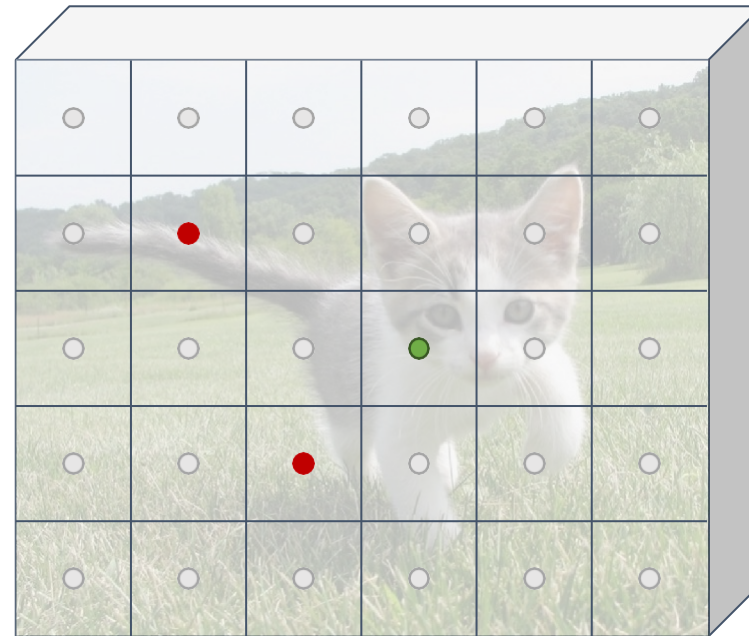


Image features
(e.g. 512 x 5 x 6)

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



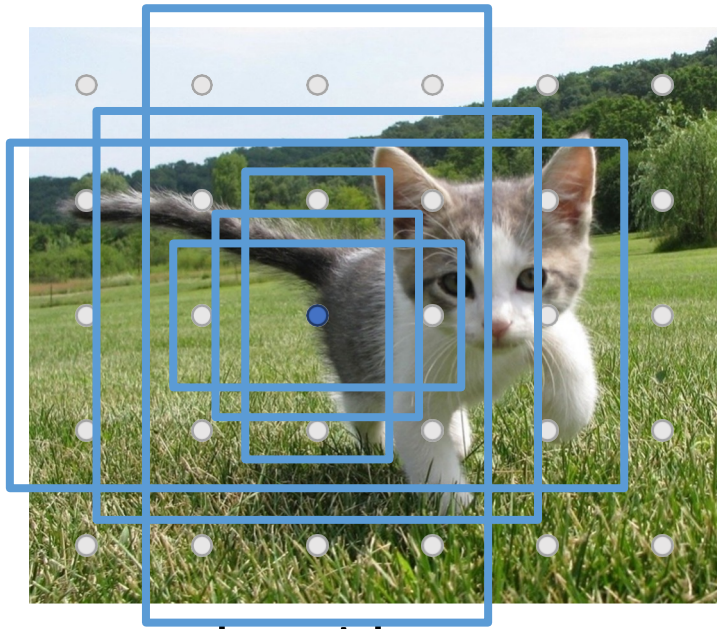
Anchor is
object?
2 x 5 x 6

Anchor
transforms
4 x 5 x 6

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

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



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

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

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

Each feature corresponds to a point in the input

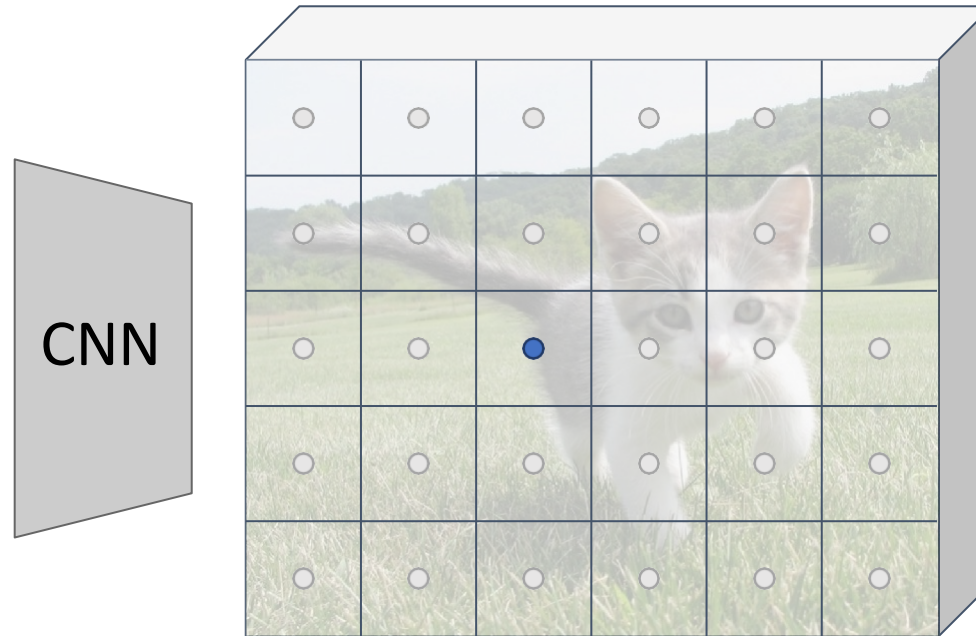


Image features
(e.g. 512 x 5 x 6)

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



Anchor is object?
2K x 5 x 6

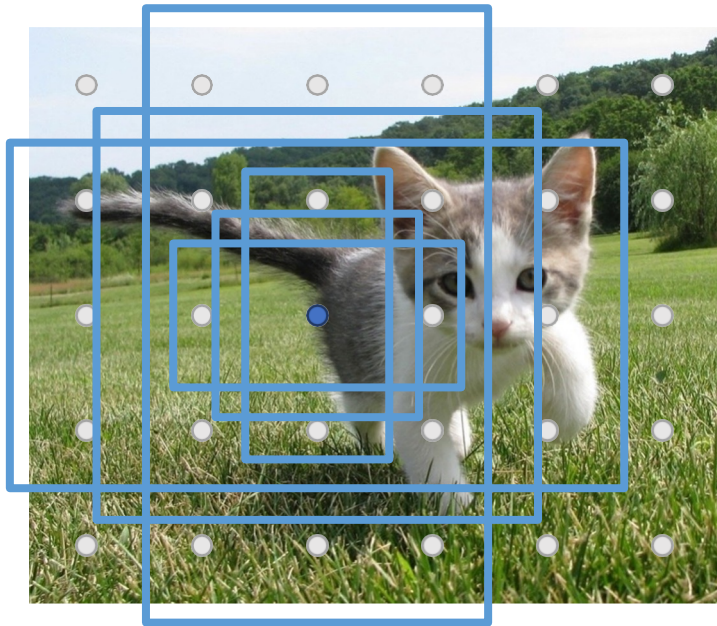
Anchor transforms
4K x 5 x 6

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

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

Region Proposal Network (RPN)

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

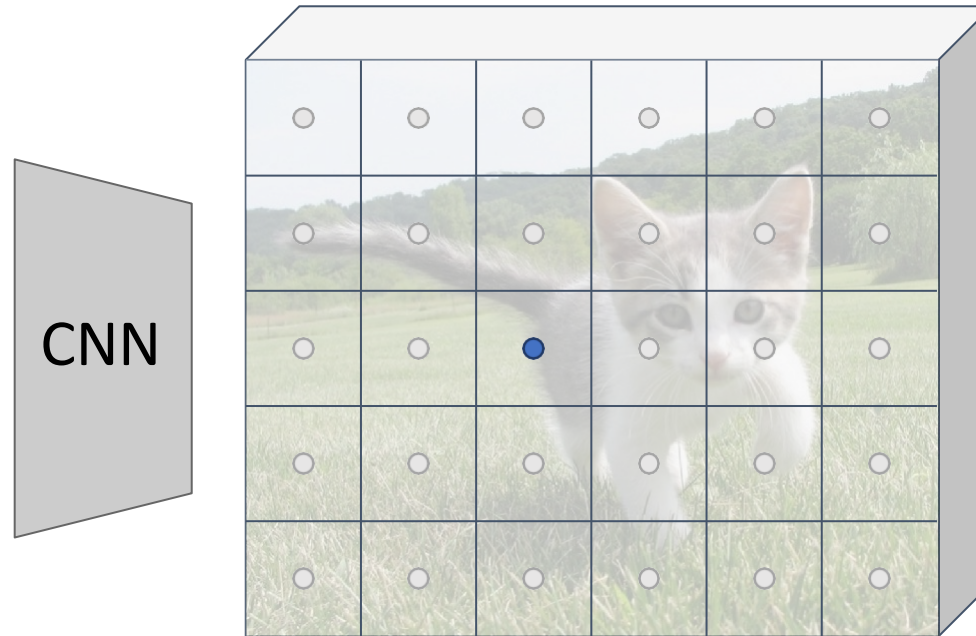


Image features
(e.g. 512 x 5 x 6)

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



Anchor is
object?
 $2K \times 5 \times 6$

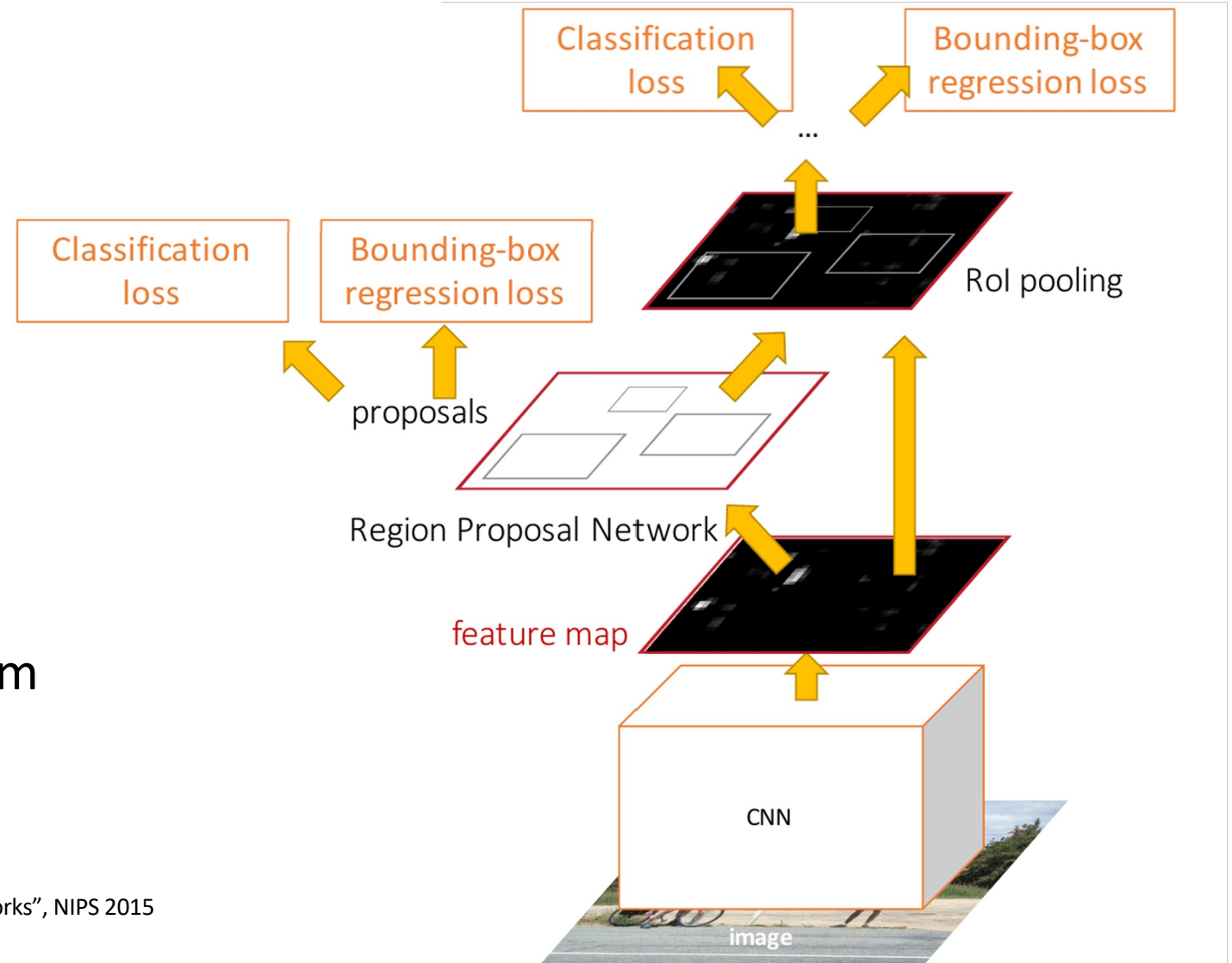
Anchor
transforms
 $4K \times 5 \times 6$

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

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



Faster R-CNN: Learnable Region Proposals

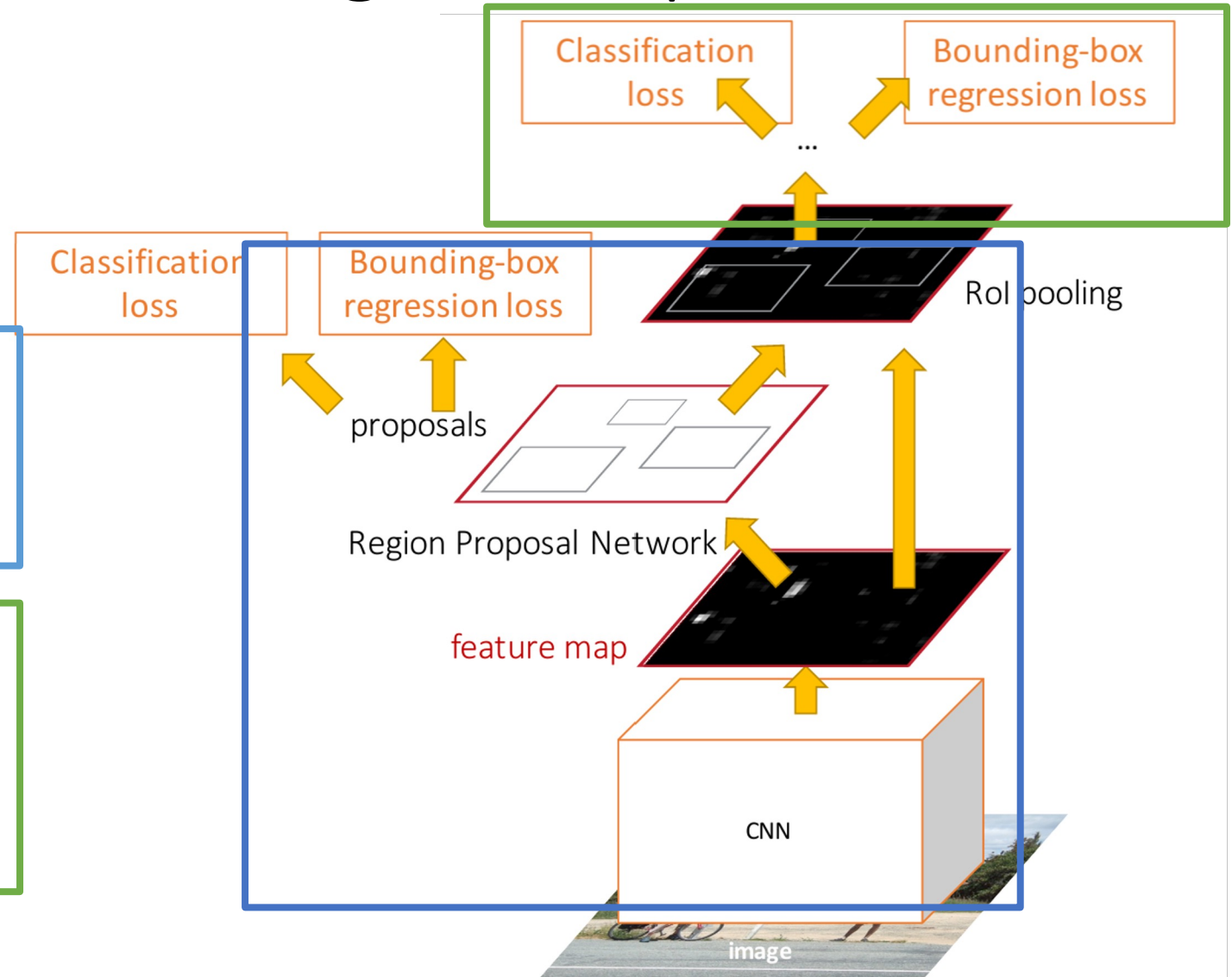
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image

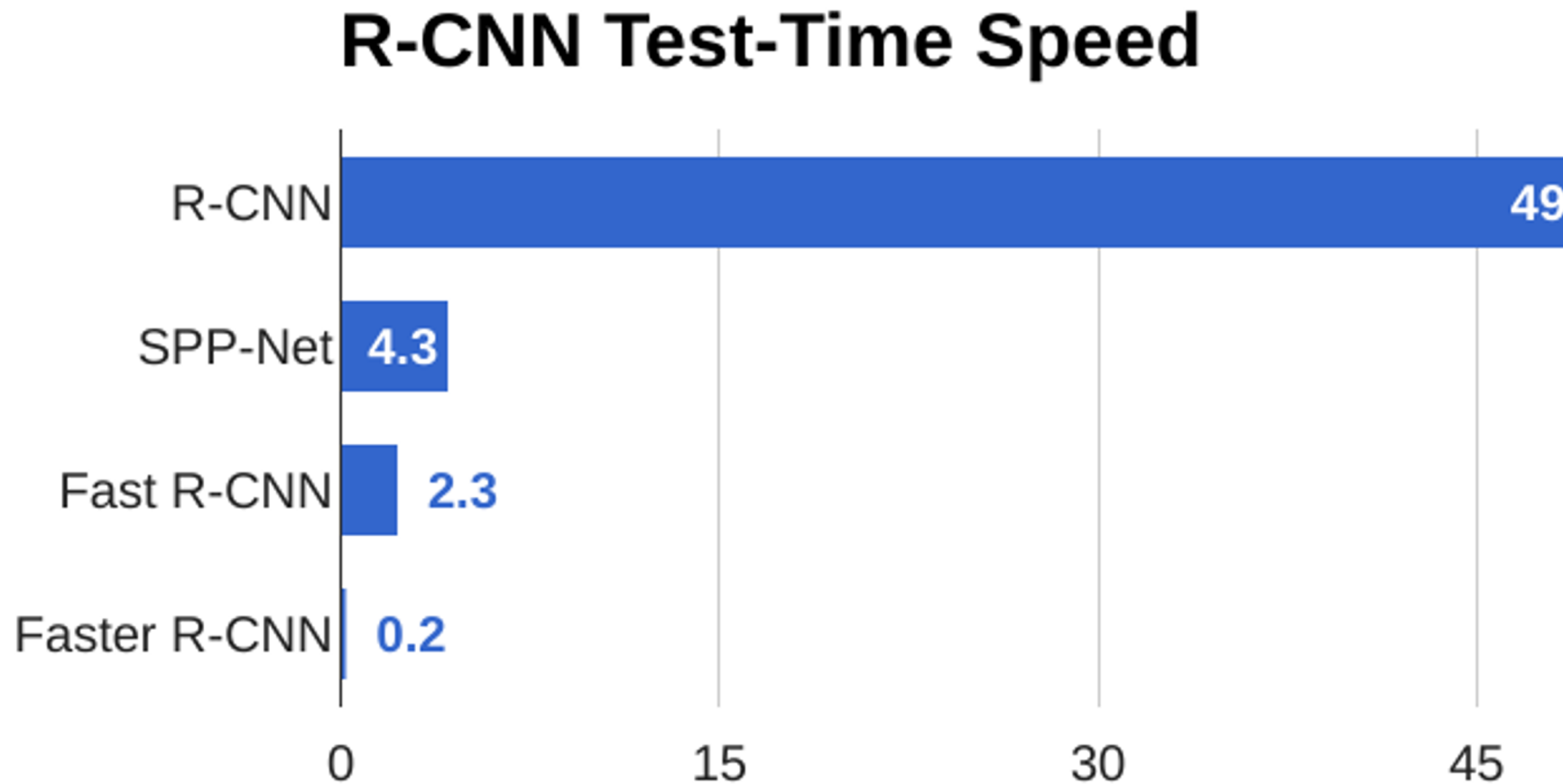
- Backbone network
- Region proposal network

Second stage: Run once per region

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



Faster R-CNN: Learnable Region Proposals

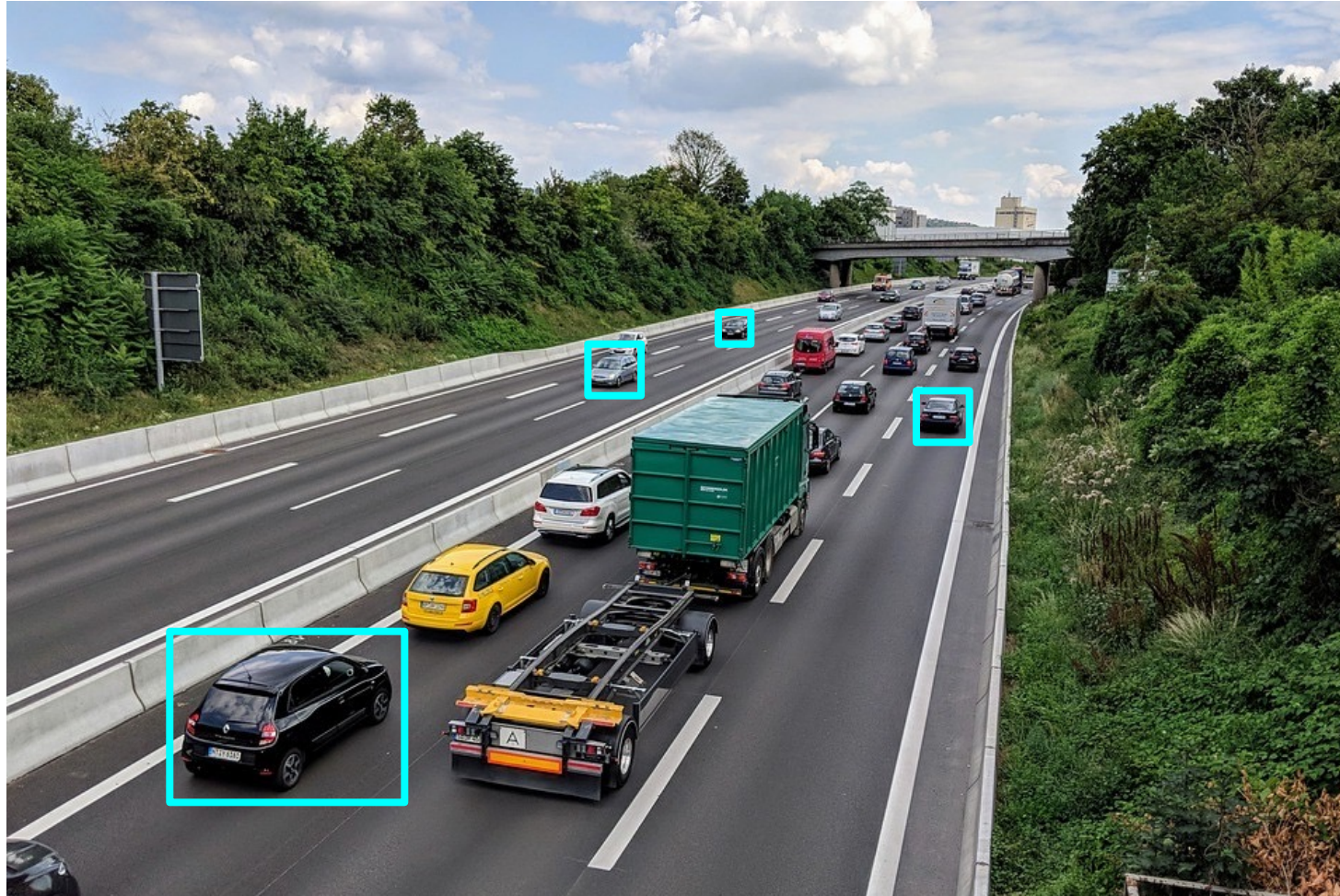


Today's class

- How do we measure Object Detection accuracy?
- Naïve approaches & R-CNN
- Fast R-CNN
- Region Proposal Network & Faster R-CNN
- **Advanced topics:**
 - Feature Pyramid Networks to detect at scales
 - Single Shot detection

Dealing with Scale

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



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



Object
Detector



Object
Detector

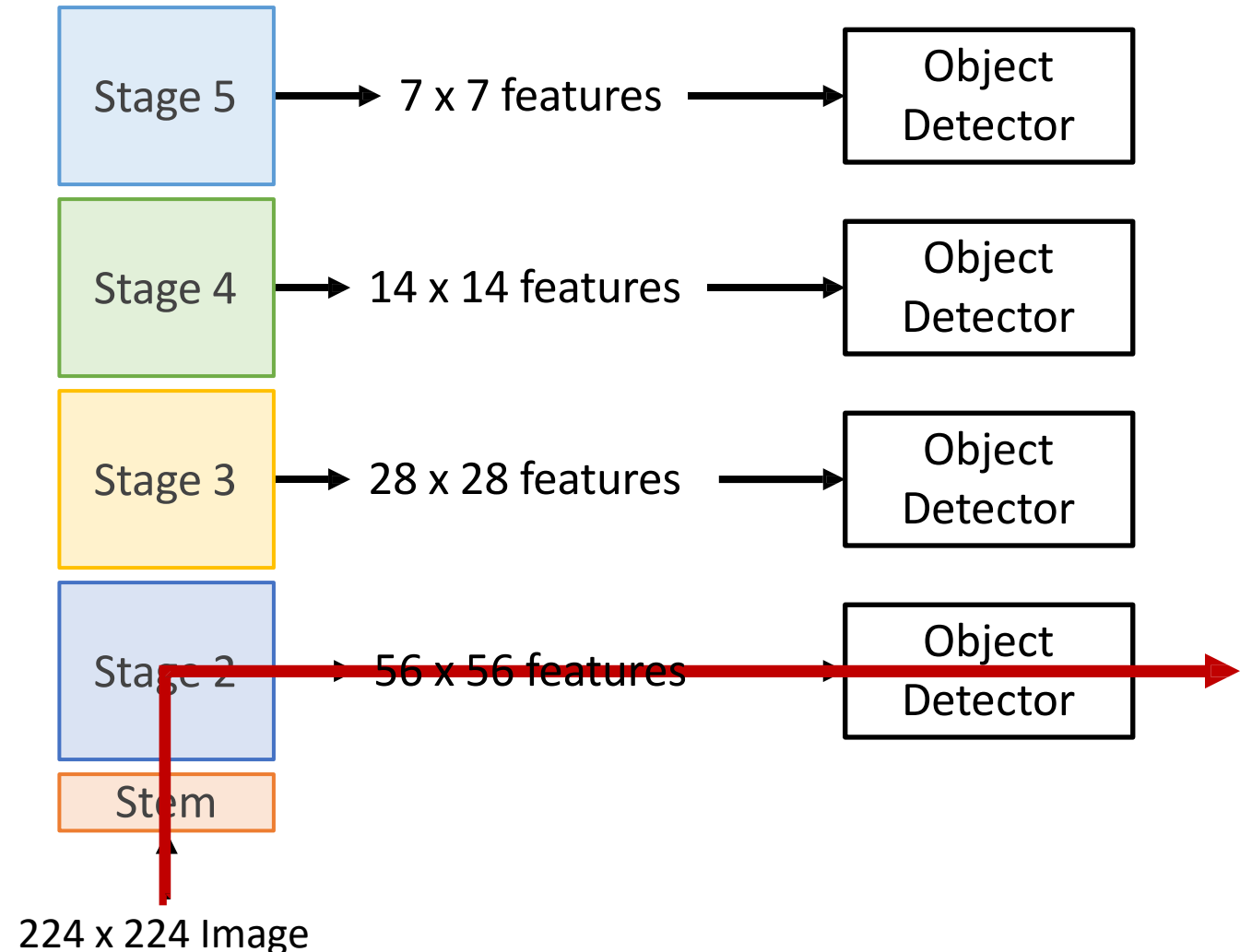


Object
Detector

Dealing with Scale: Multiscale Features

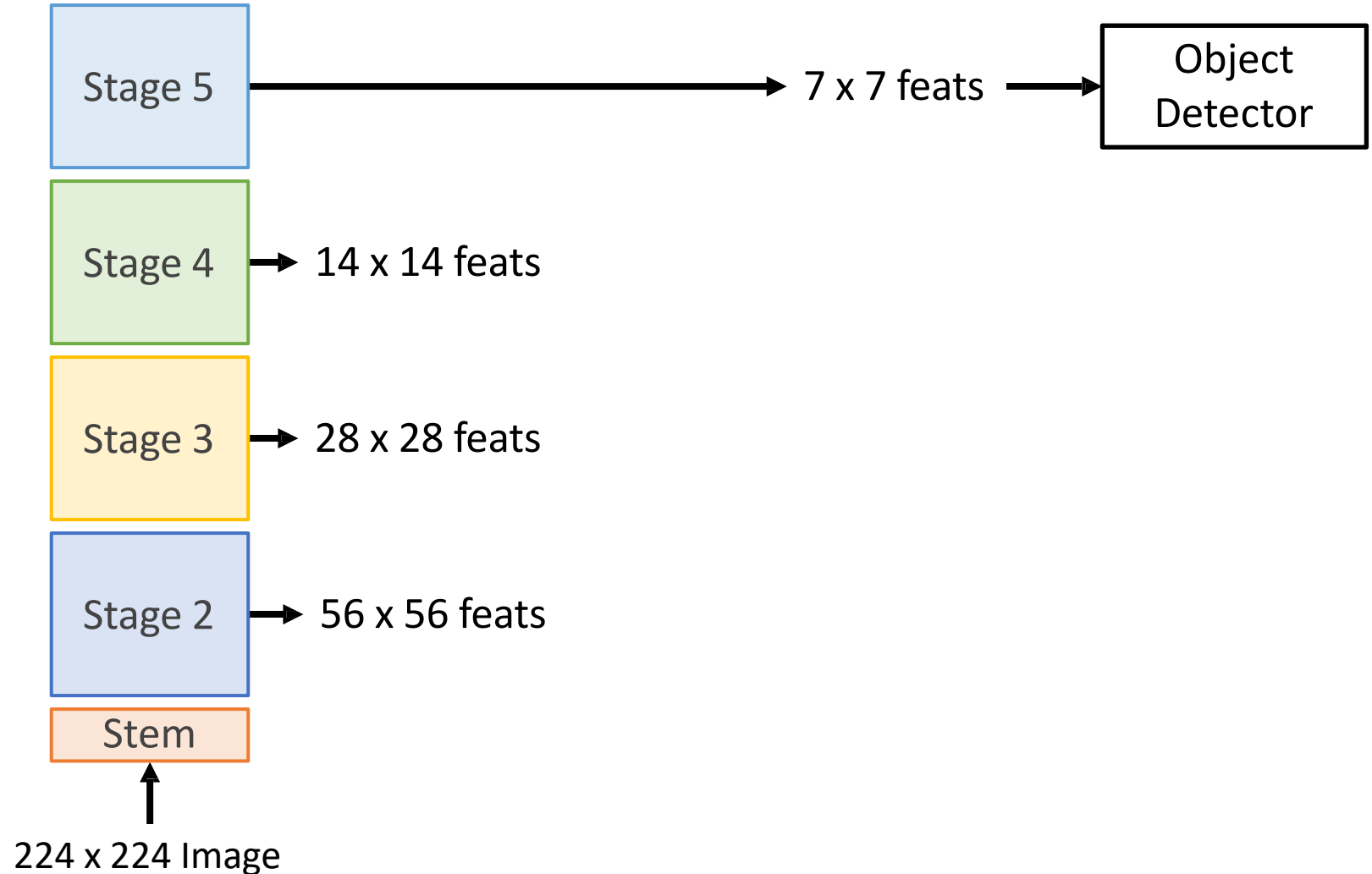
CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

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



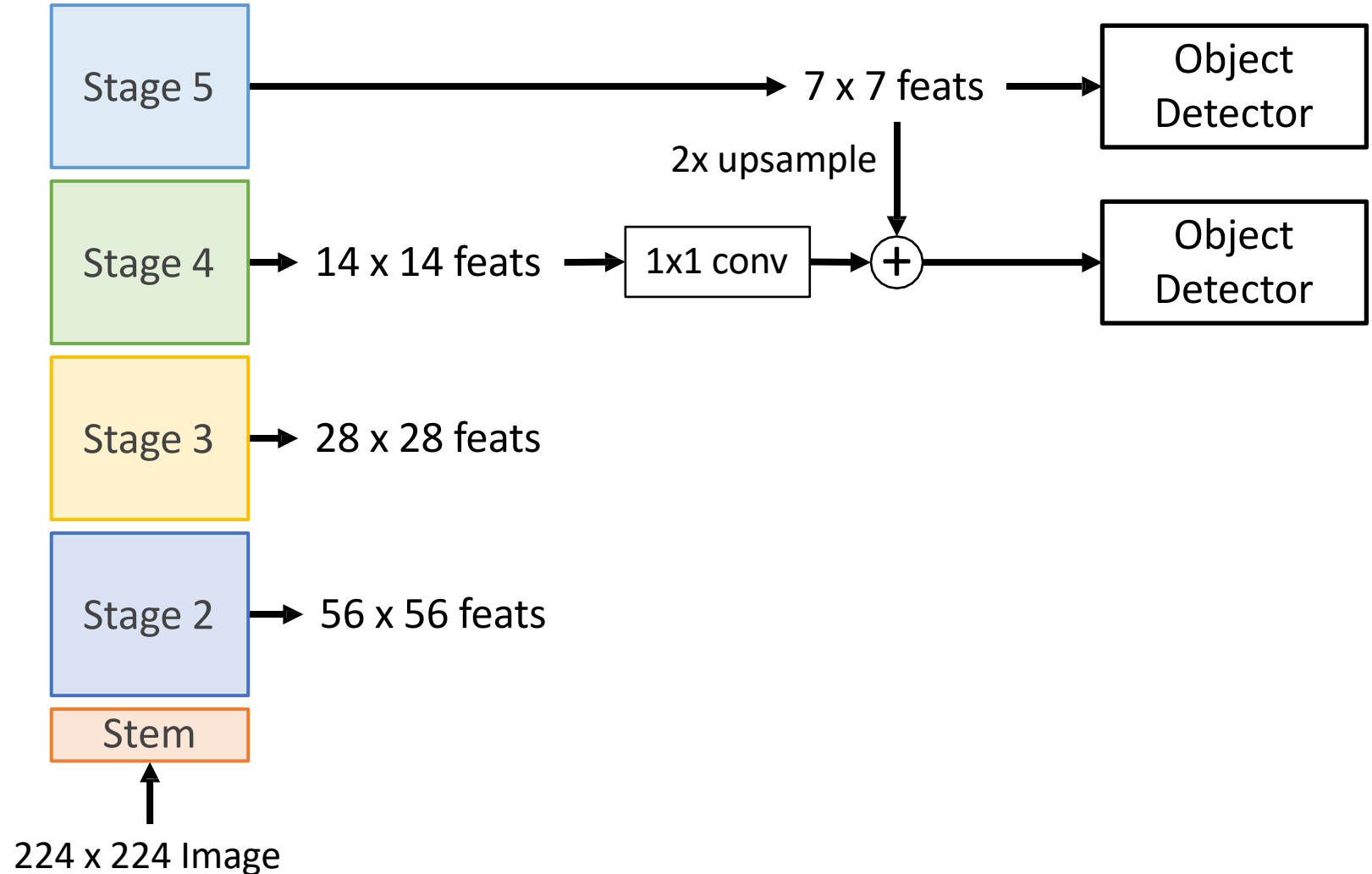
Dealing with Scale: Feature Pyramid Network

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



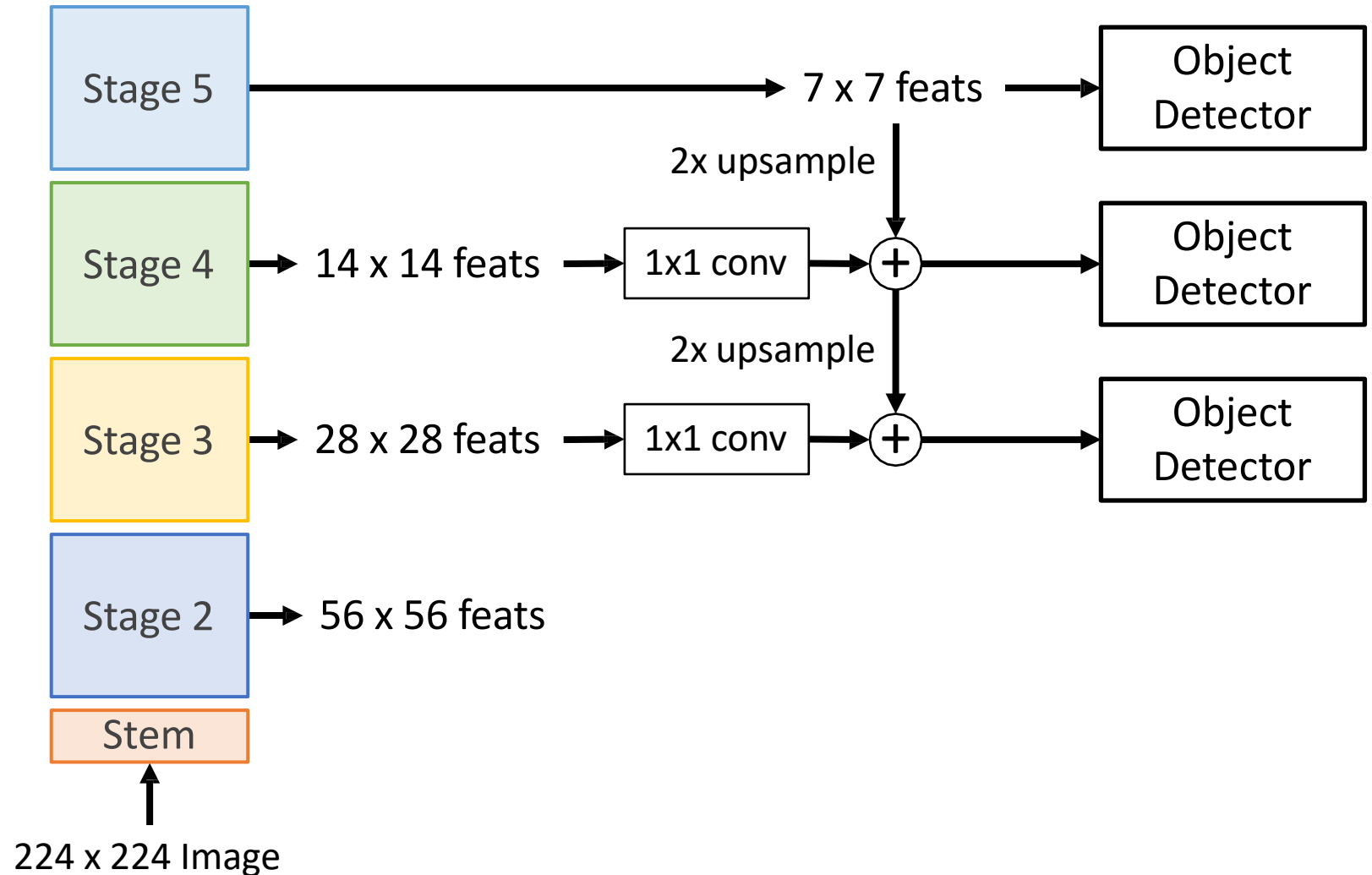
Dealing with Scale: Feature Pyramid Network

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



Dealing with Scale: Feature Pyramid Network

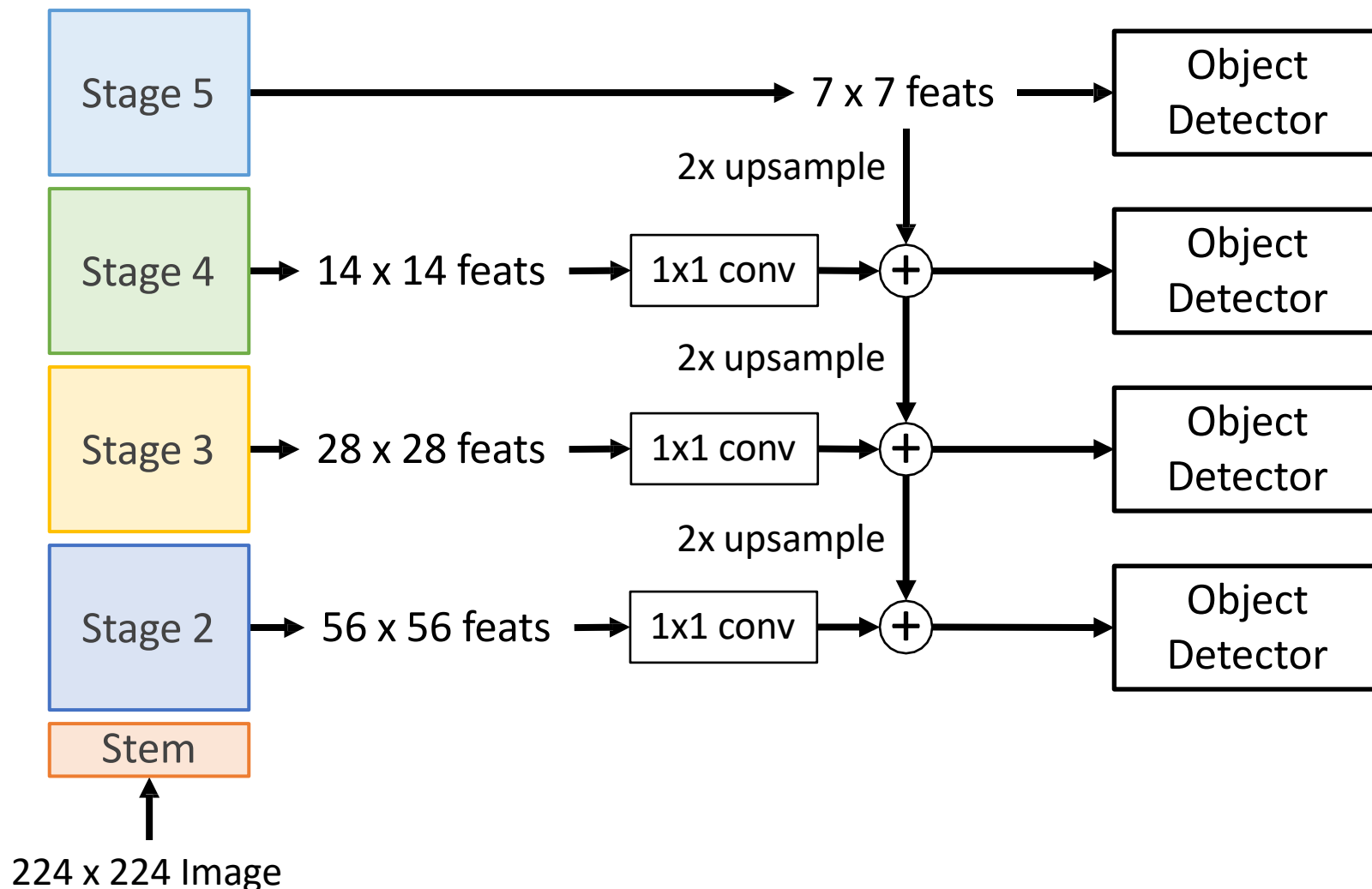
Add *top down connections* that feed information from high level features back down to lower level features



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: Learnable Region Proposals

Question: Do we really need the second stage?

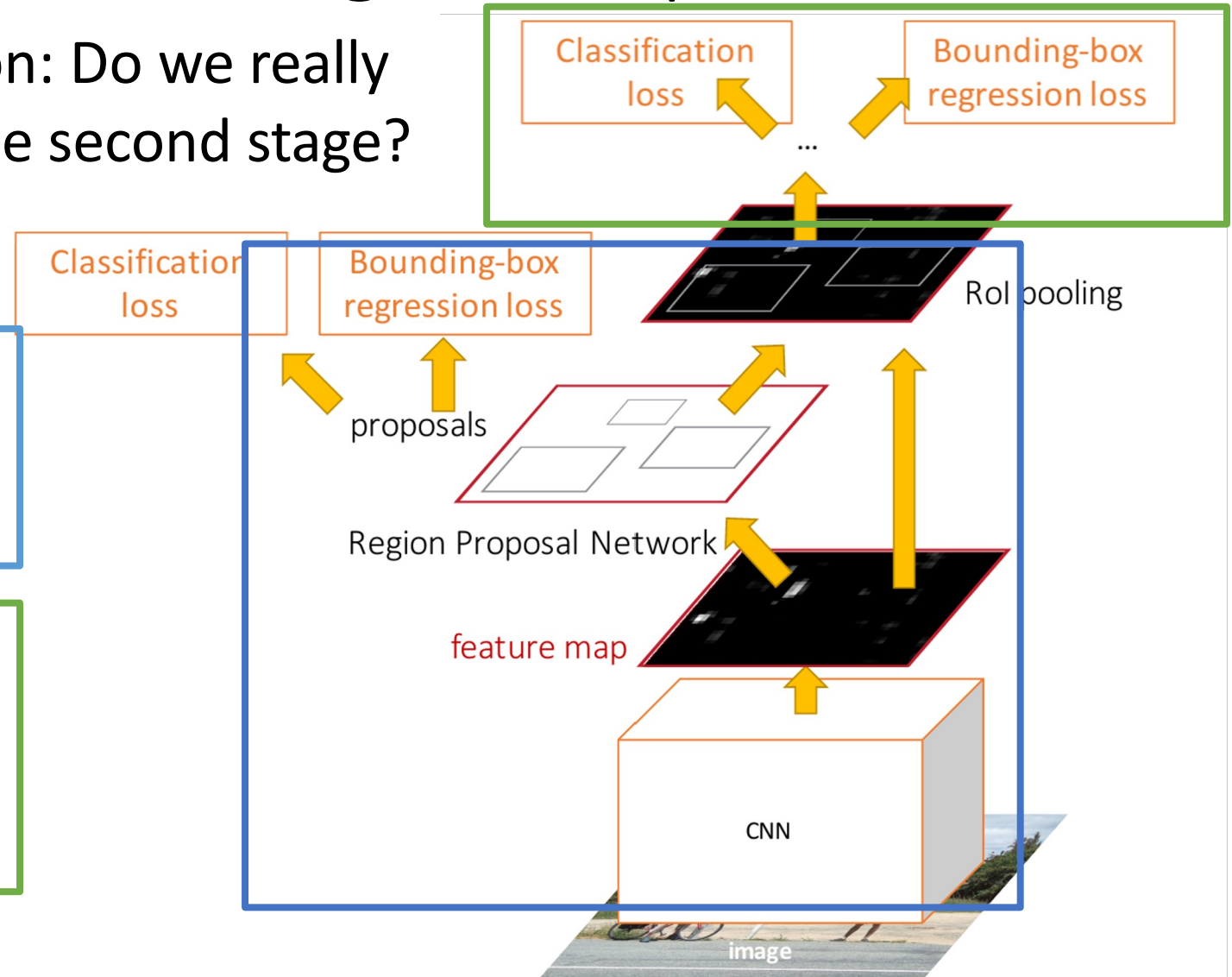
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image

- Backbone network
- Region proposal network

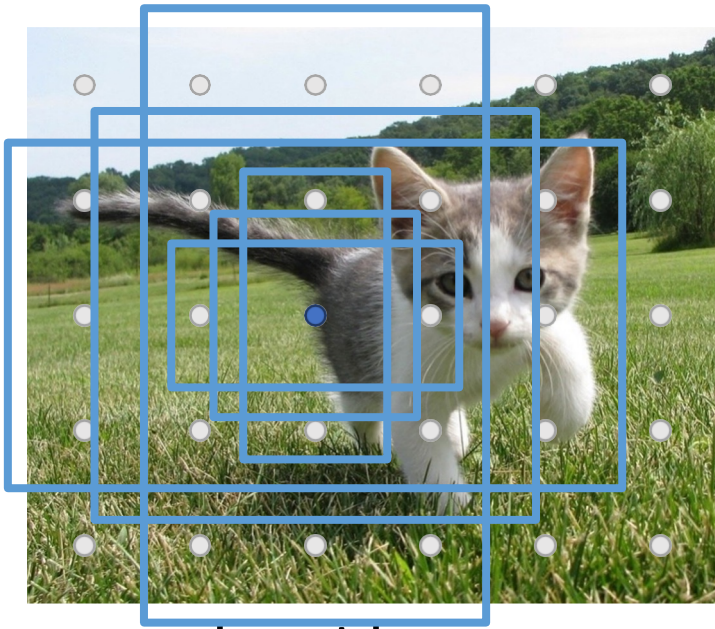
Second stage: Run once per region

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



Single-Stage Detectors: RetinaNet

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

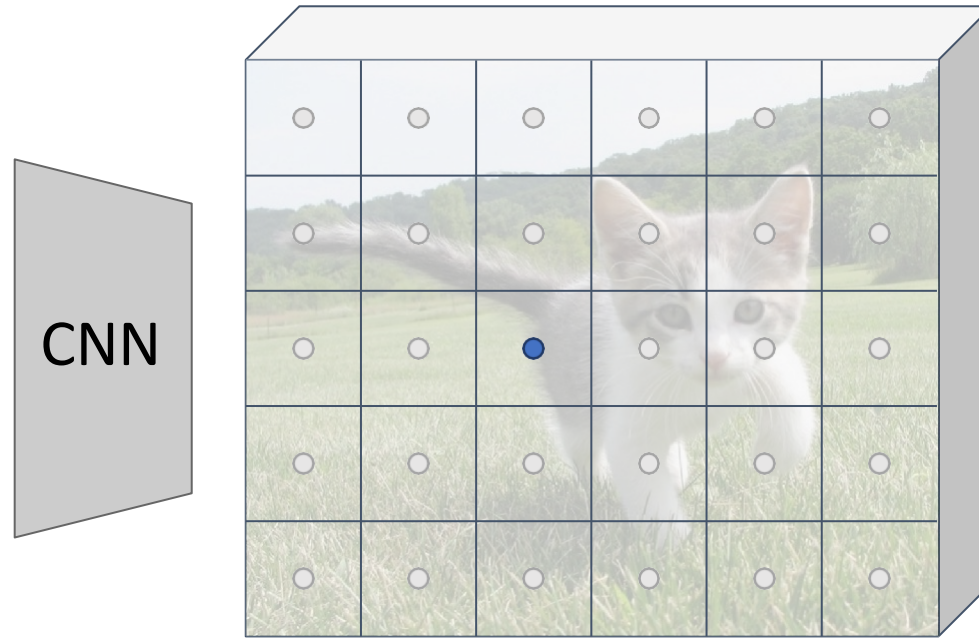


Image features
(e.g. 512 x 5 x 6)

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

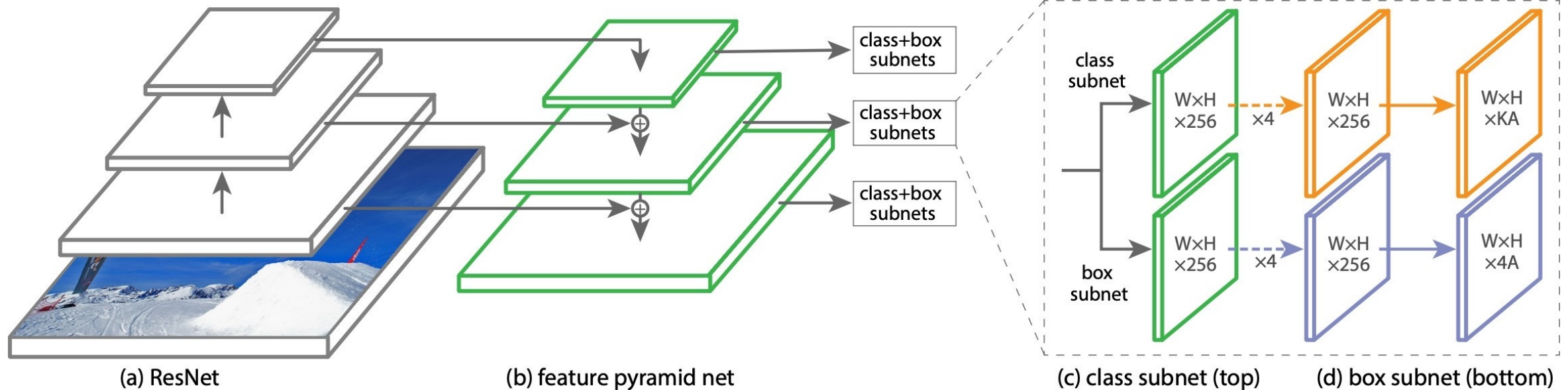


Anchor
classification
 $2K \times (C+1) \times 5 \times 6$

Anchor
transforms
 $4K \times 5 \times 6$

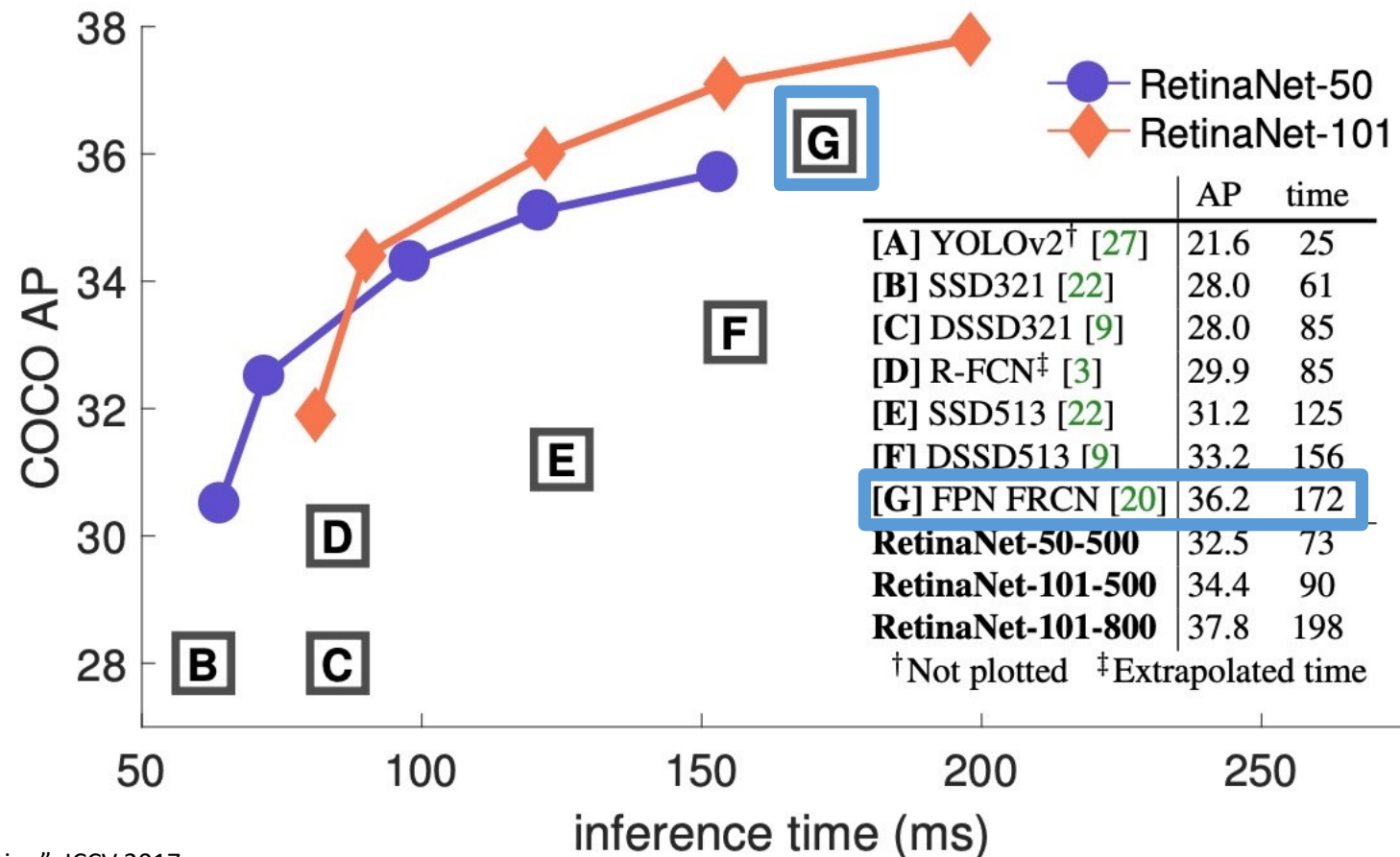
Single-Stage Detectors: RetinaNet

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



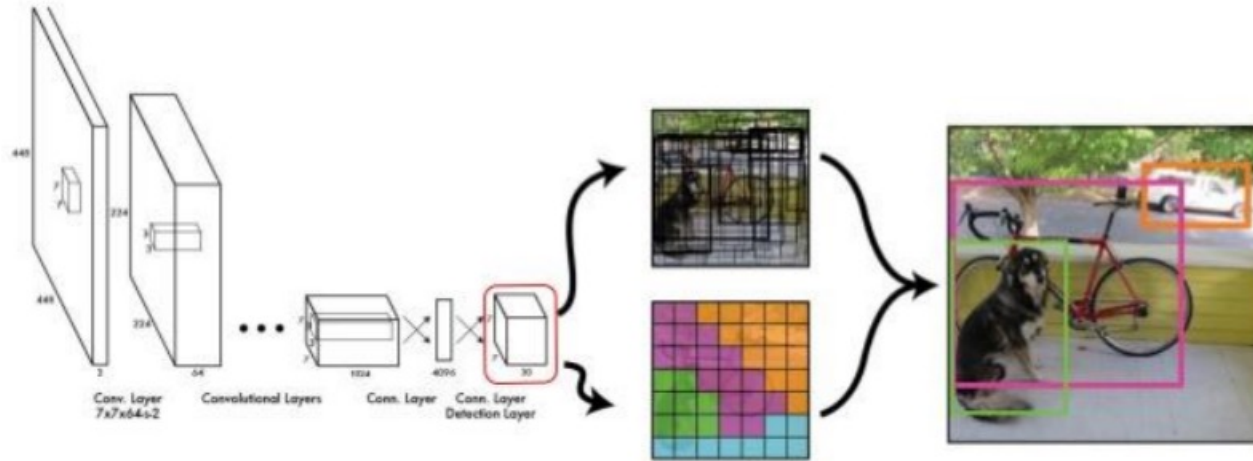
Single-Stage Detectors: RetinaNet

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

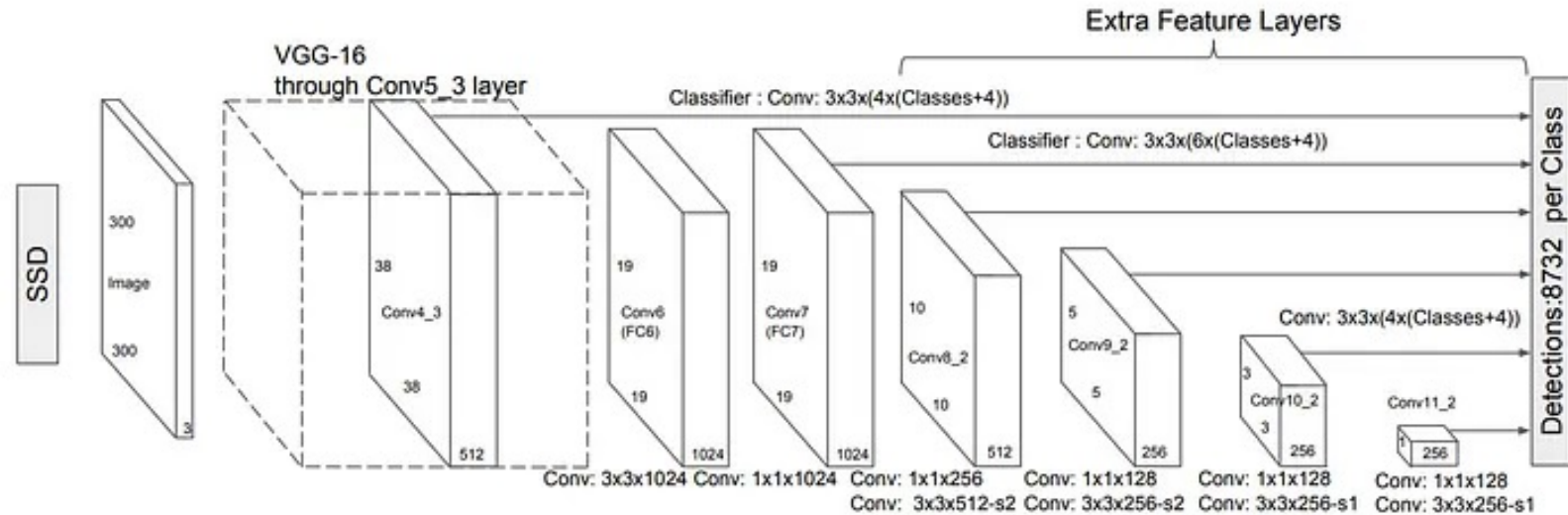


Faster R-CNN
with Feature
Pyramid
Network

Other popular Single-Stage Detectors



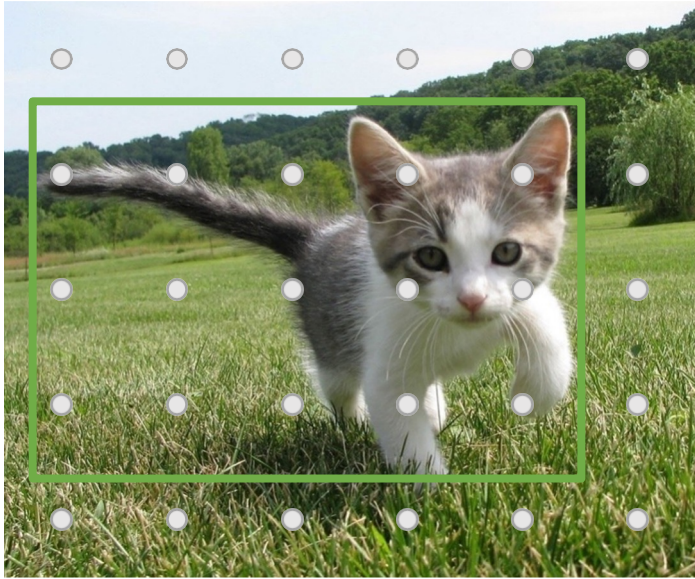
YOLO (You Only Look Once) – v1 to v7 versions



SSD Multibox detector, developed at UNC

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

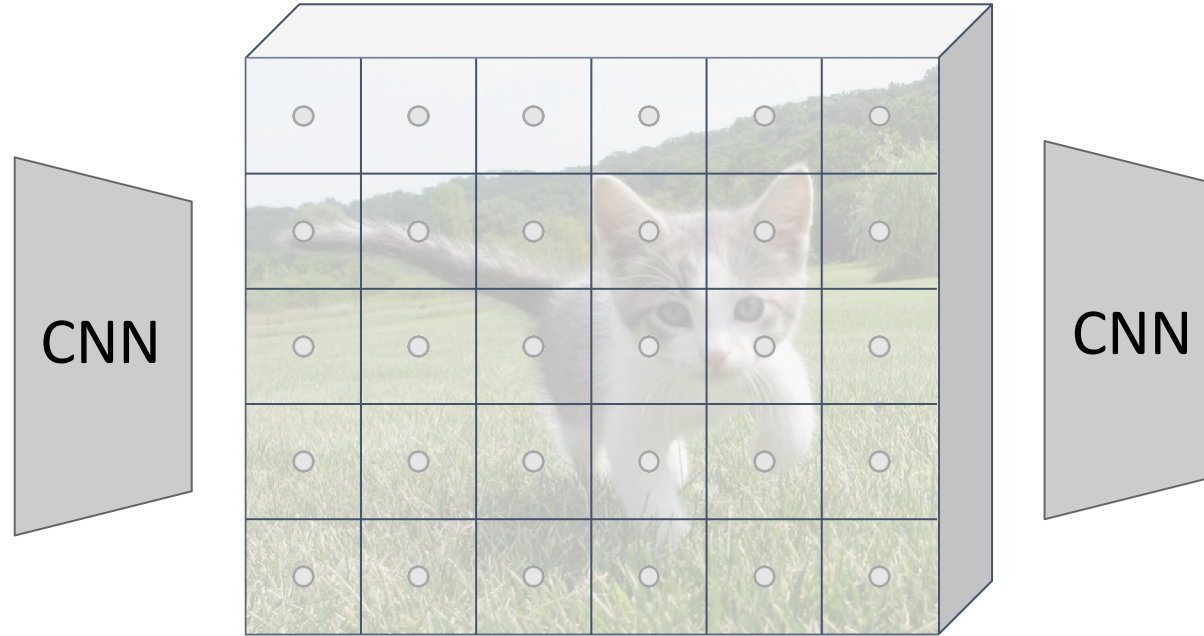
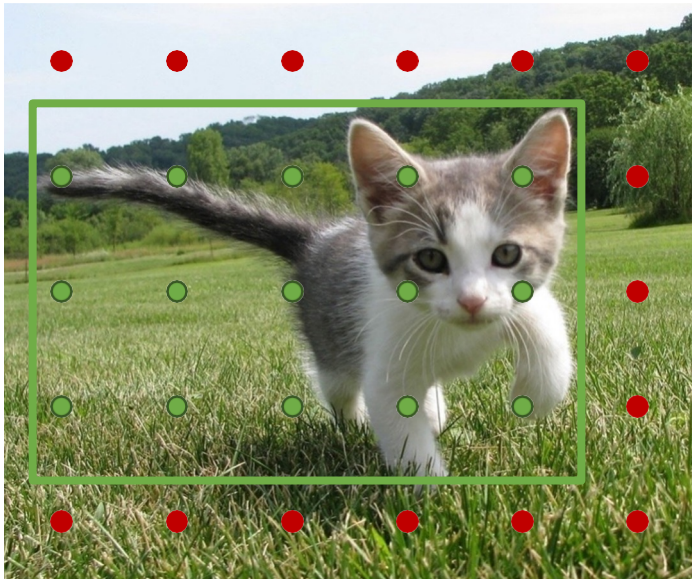


Image features
(e.g. 512 x 5 x 6)

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

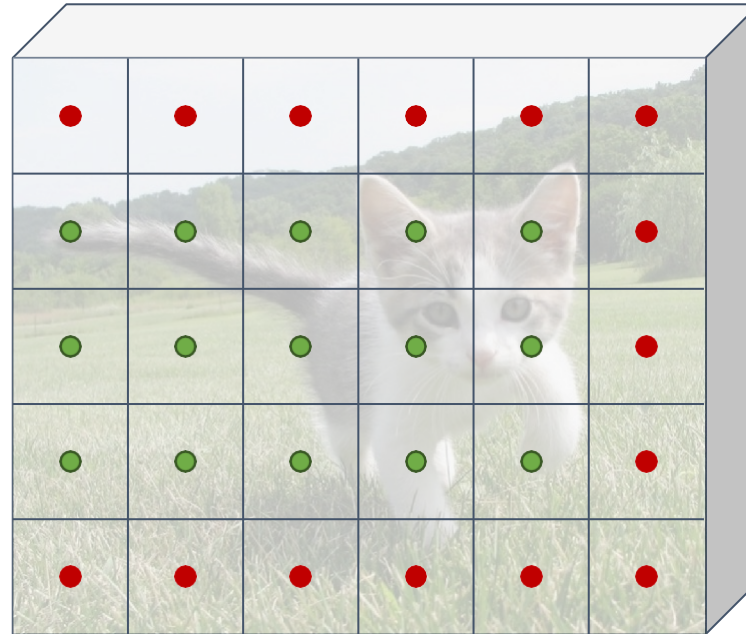


Image features
(e.g. 512 x 5 x 6)

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

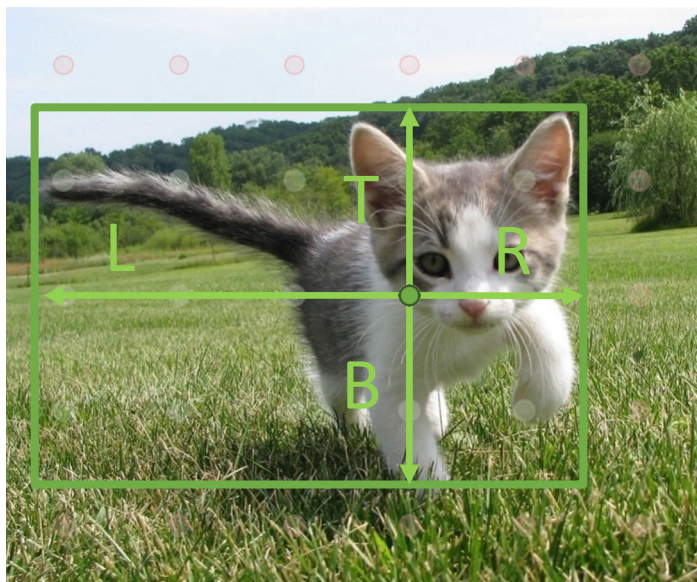
Train independent per-category logistic regressors



→ Class scores
 $C \times 5 \times 6$

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

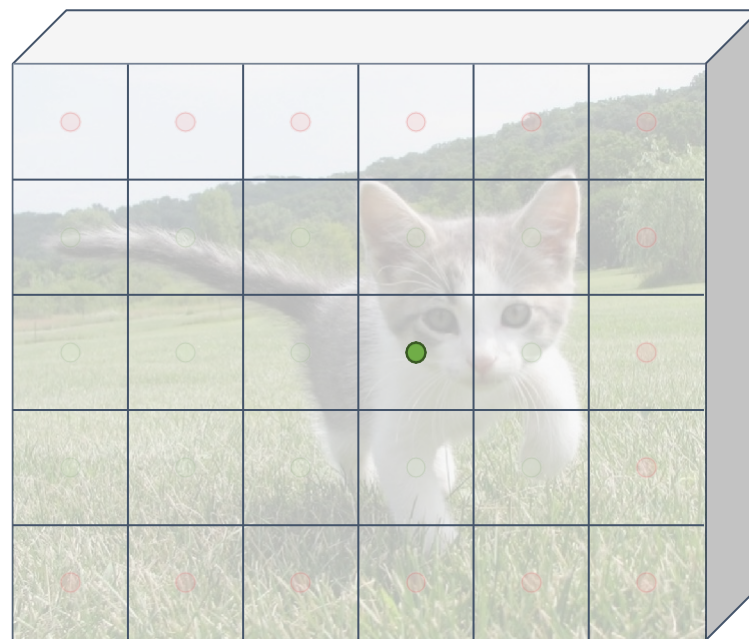


Image features
(e.g. 512 x 5 x 6)

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

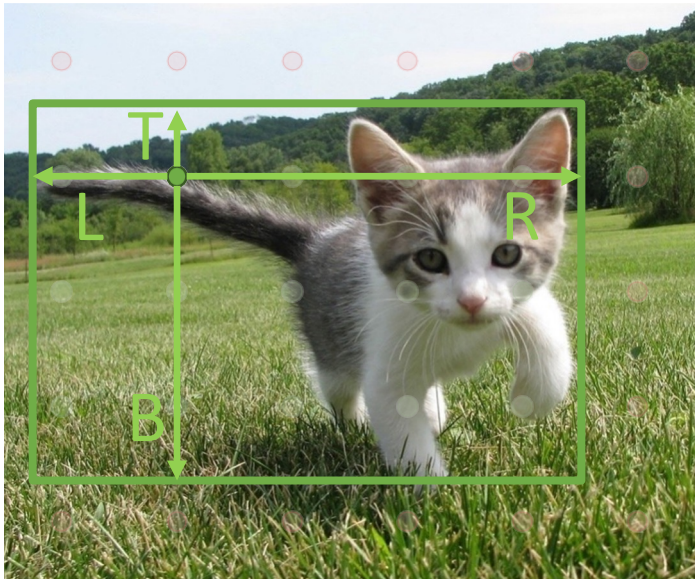


→ Class scores
 $C \times 5 \times 6$

→ Box edges
 $4 \times 5 \times 6$

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

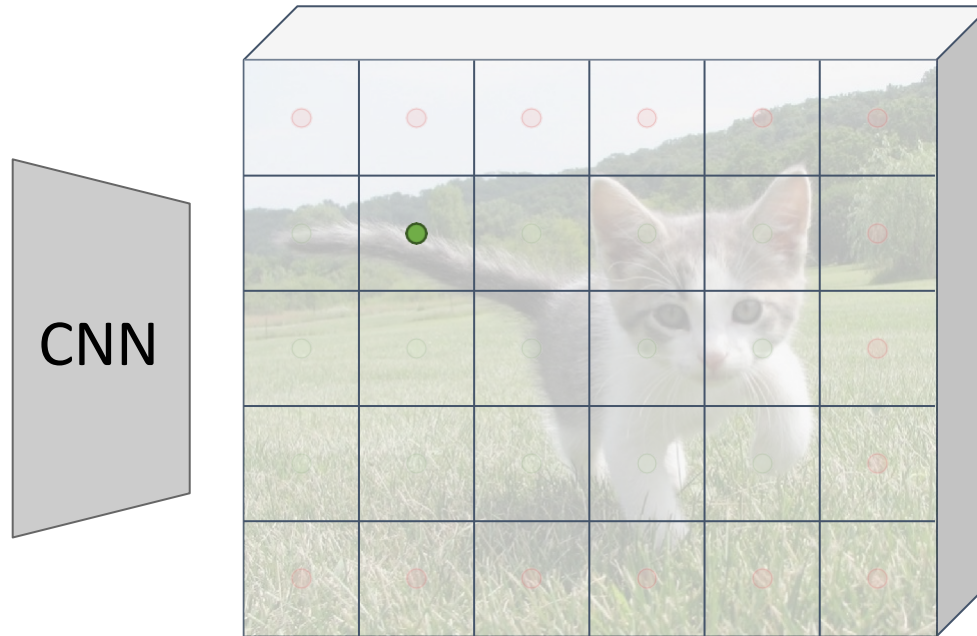
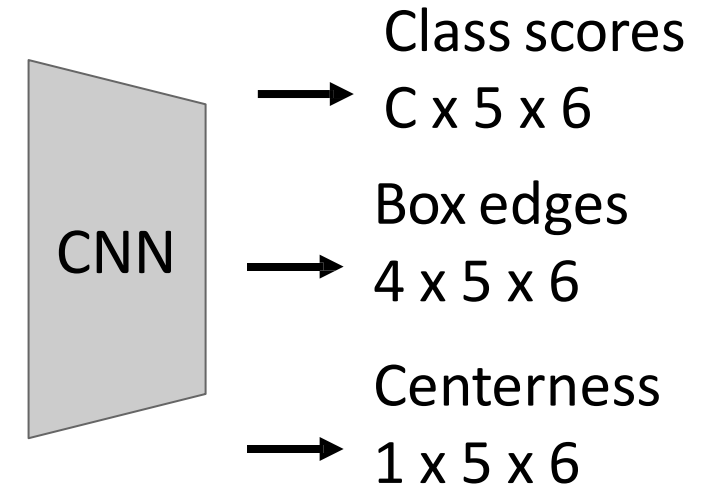


Image features
(e.g. 512 x 5 x 6)

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



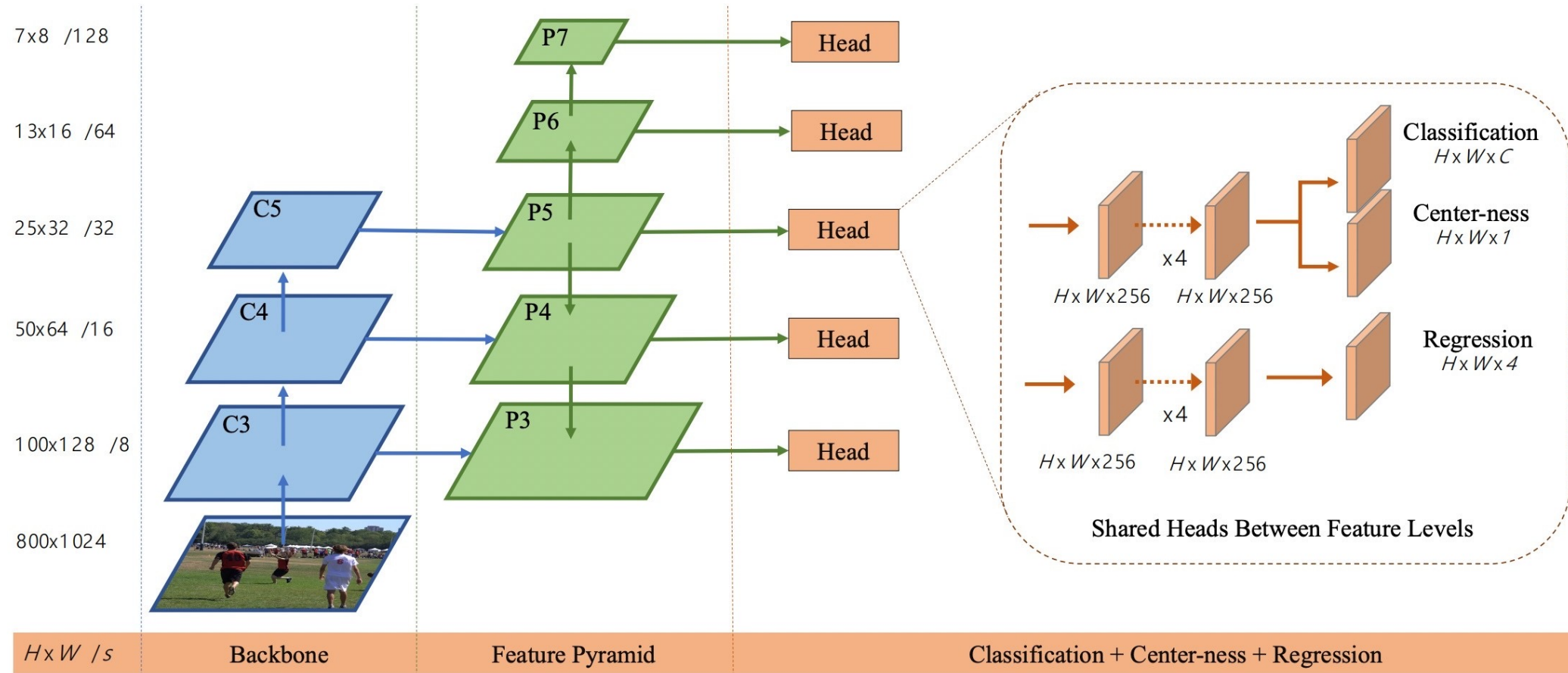
$$centerness = \sqrt{\frac{\min(L, R)}{\max(L, R)} \cdot \frac{\min(T, B)}{\max(T, B)}}$$

Ranges from 1 at box center to 0 at box edge

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

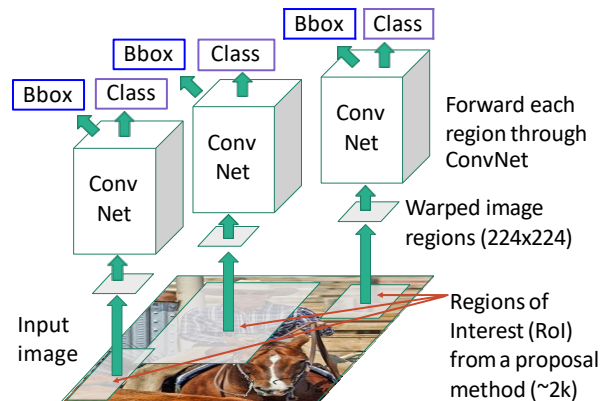
Single-Stage Detectors: FCOS (“Anchor-free” detector)

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

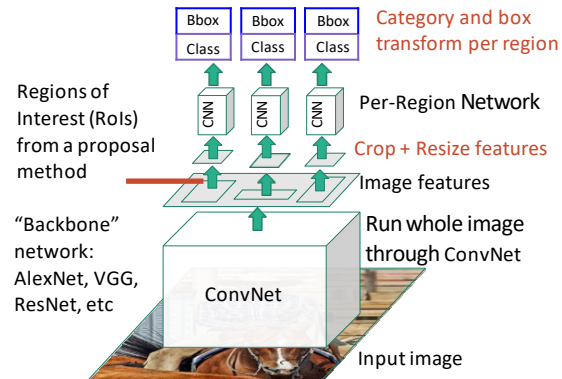


Summary

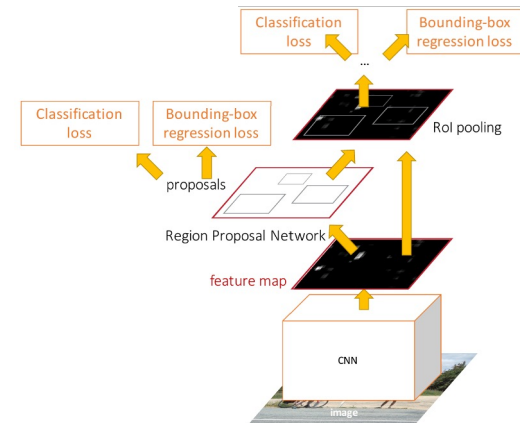
“Slow” R-CNN: Run CNN independently for each region



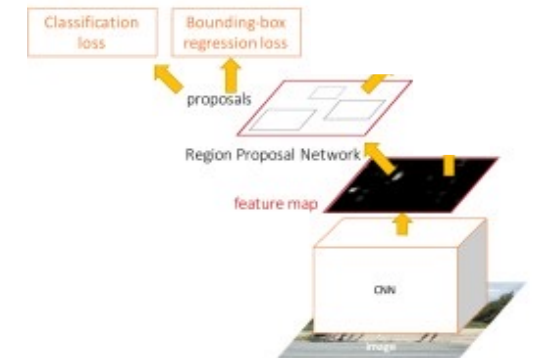
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



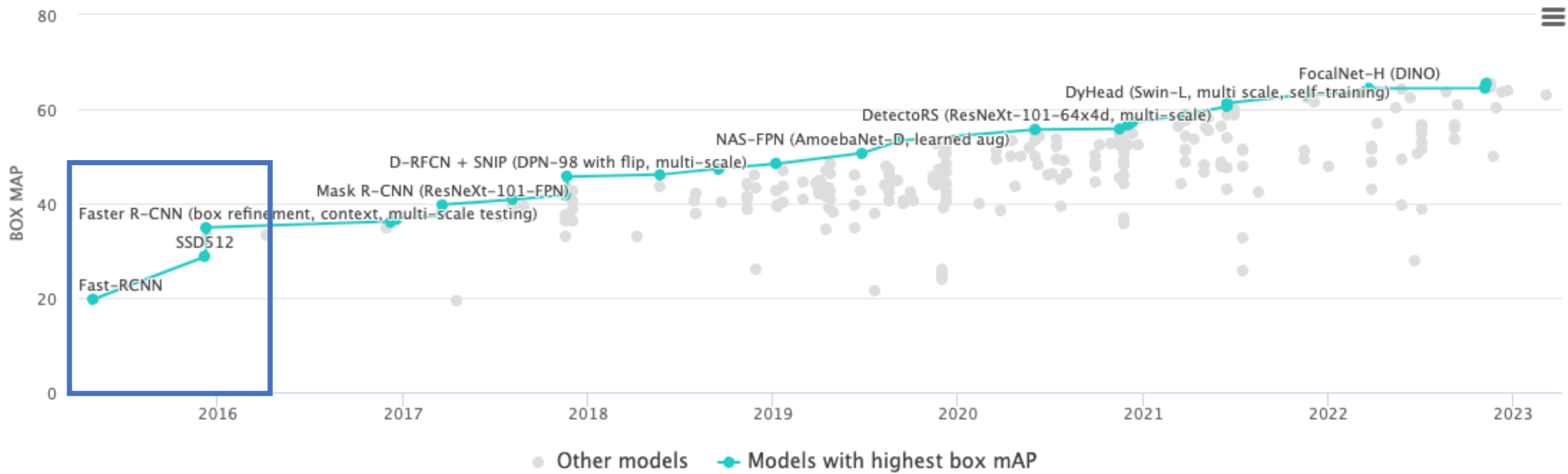
With anchors: RetinaNet
Anchor-Free: FCOS

Object Detection on COCO test-dev

Leaderboard

Dataset

View box mAP by Date for All models



Current SOTA in object detection

InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions

Wenhai Wang^{1*}, Jifeng Dai^{2,1*}, Zhe Chen^{3,1*}, Zhenhang Huang^{1*}, Zhiqi Li^{3,1*}, Xizhou Zhu^{4*},
Xiaowei Hu¹, Tong Lu³, Lewei Lu⁴, Hongsheng Li⁵, Xiaogang Wang^{4,5}, Yu Qiao^{1✉}

¹ Shanghai AI Laboratory ² Tsinghua University

³ Nanjing University ⁴ SenseTime Research ⁵ The Chinese University of Hong Kong

<https://github.com/OpenGVLab/InternImage>