

DDA4220 · Fall 2026

Lecture 9

Deep Learning and Applications

Object detection and instance segmentation

Zhen Liu

CUHK-Shenzhen

Lecture outline

| 01 The object detection problem

02 Region-based detection

03 Detection evaluation

04 Dense and multiscale detection

05 Instance segmentation

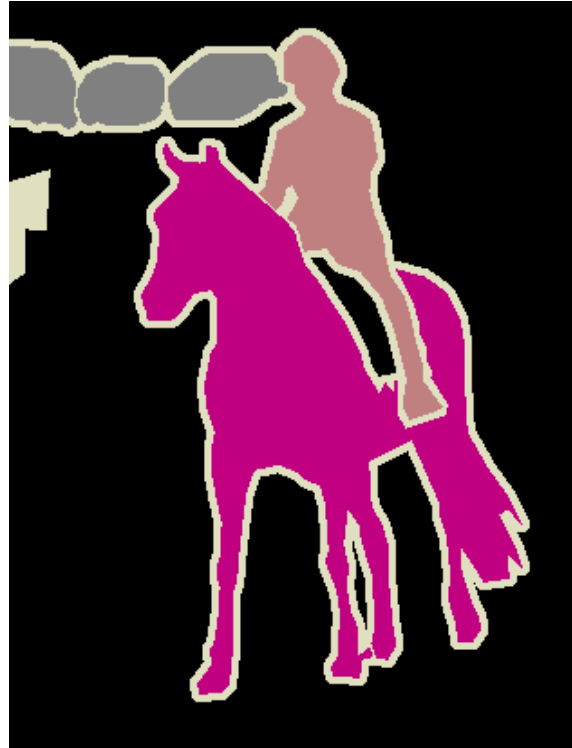
* Optional

Object detection

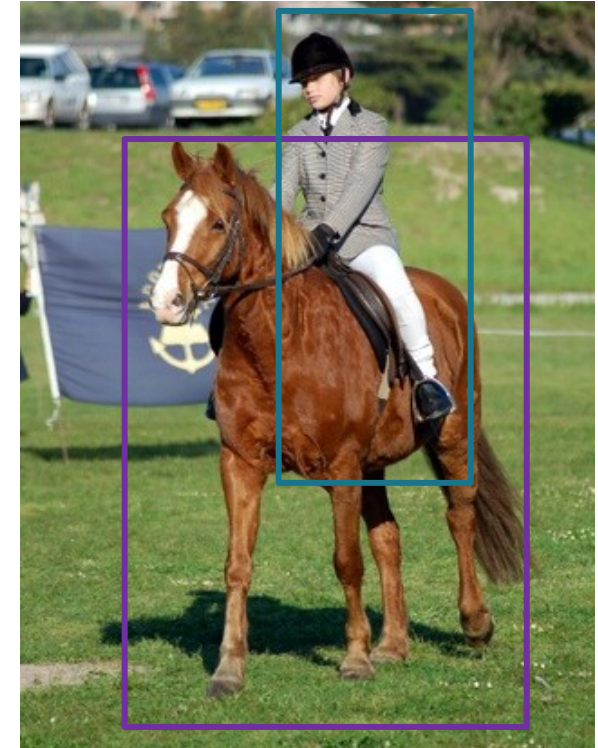
Input image



Semantic labels



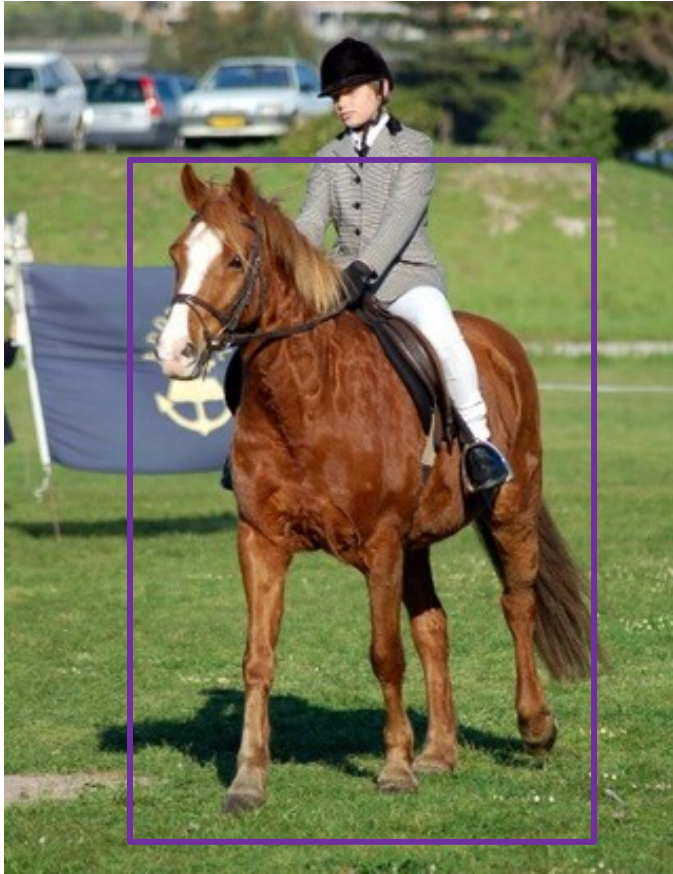
Individual objects



Horse

Person

Detection outputs



Category

Horse

Bounding box

Location and extent in the image

Confidence score

Higher score → stronger predicted detection

Ground truth = annotated category and box

Why object detection is harder

One object



Several objects



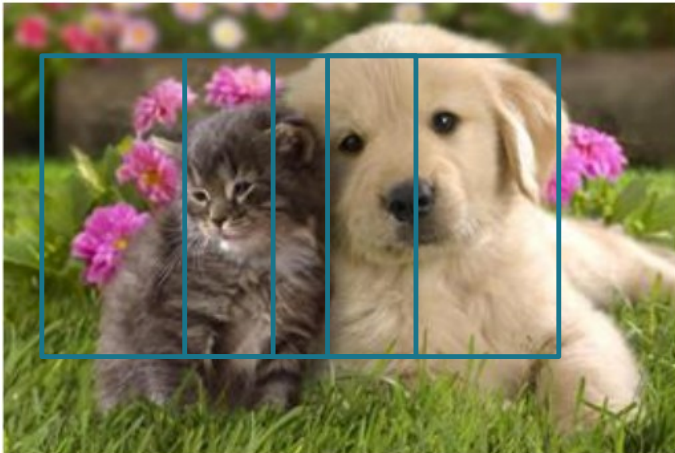
Many objects



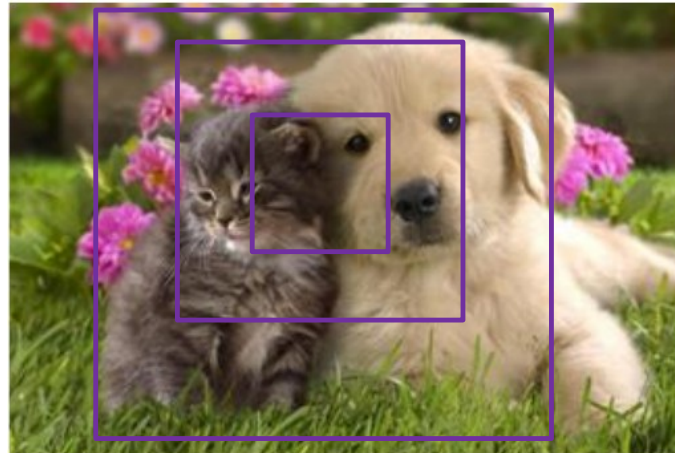
The number, positions and sizes of objects all vary.
On a new image, the detector receives no object boxes.

Sliding-window detection

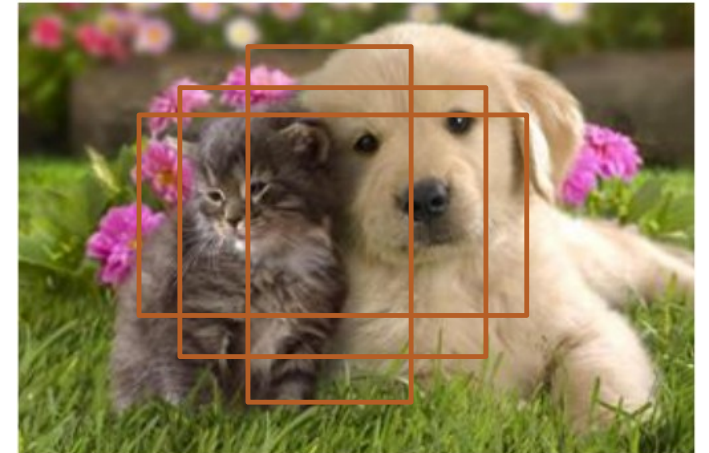
Position



Scale



Aspect ratio

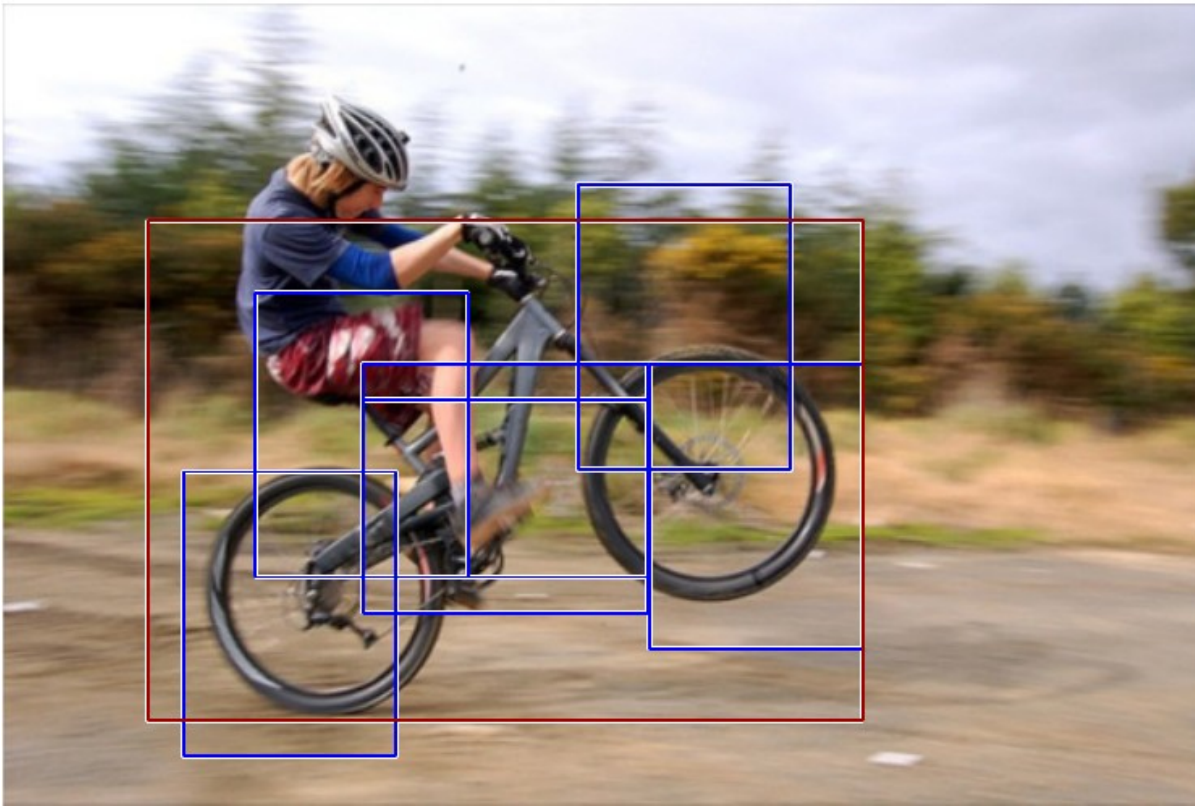


Try windows at different positions, sizes and aspect ratios.

Classify each crop as an object category or background.

Detection before CNNs

Deformable part model



Red: object box Blue: parts

Source: [Discriminatively Trained Part Based Models](#)

Sliding-window detectors

HOG describes local edge directions.
An SVM learns to score each window.

Part-based models

Learn part appearances and allow
their positions to vary.

Lecture outline

01 The object detection problem

| 02 **Region-based detection**

03 Detection evaluation

04 Dense and multiscale detection

05 Instance segmentation

With candidate boxes supplied

Assume approximate candidate boxes are already available.

One candidate crop



Classification

Cat, dog, ..., or background

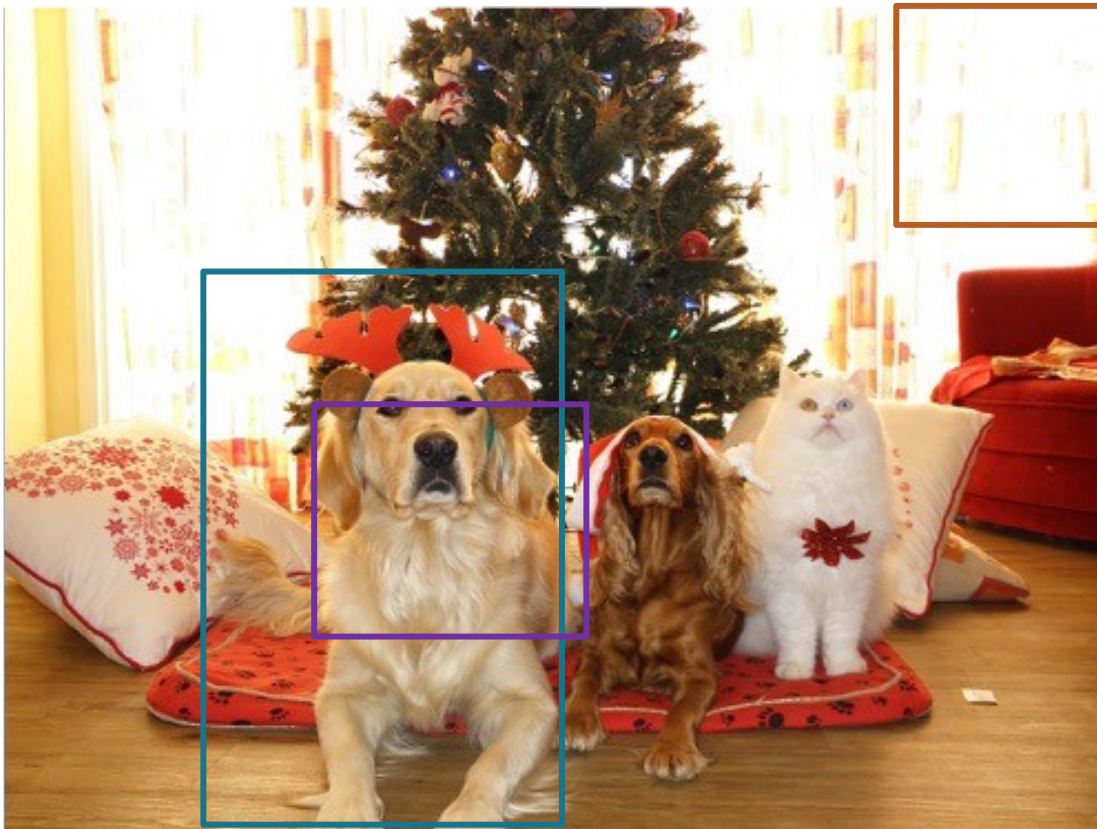
Box regression

Shift and resize the candidate

Apply the same predictor to each candidate region.

Object and background candidates

Candidate regions



Object region



Partial object



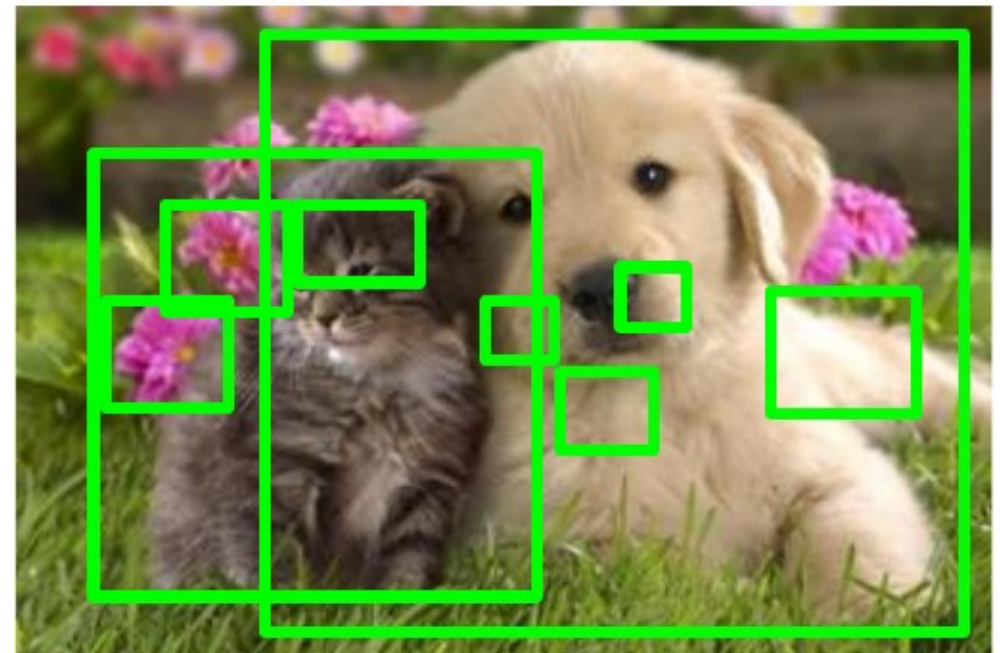
Background

Region proposals

Input image



Region proposals

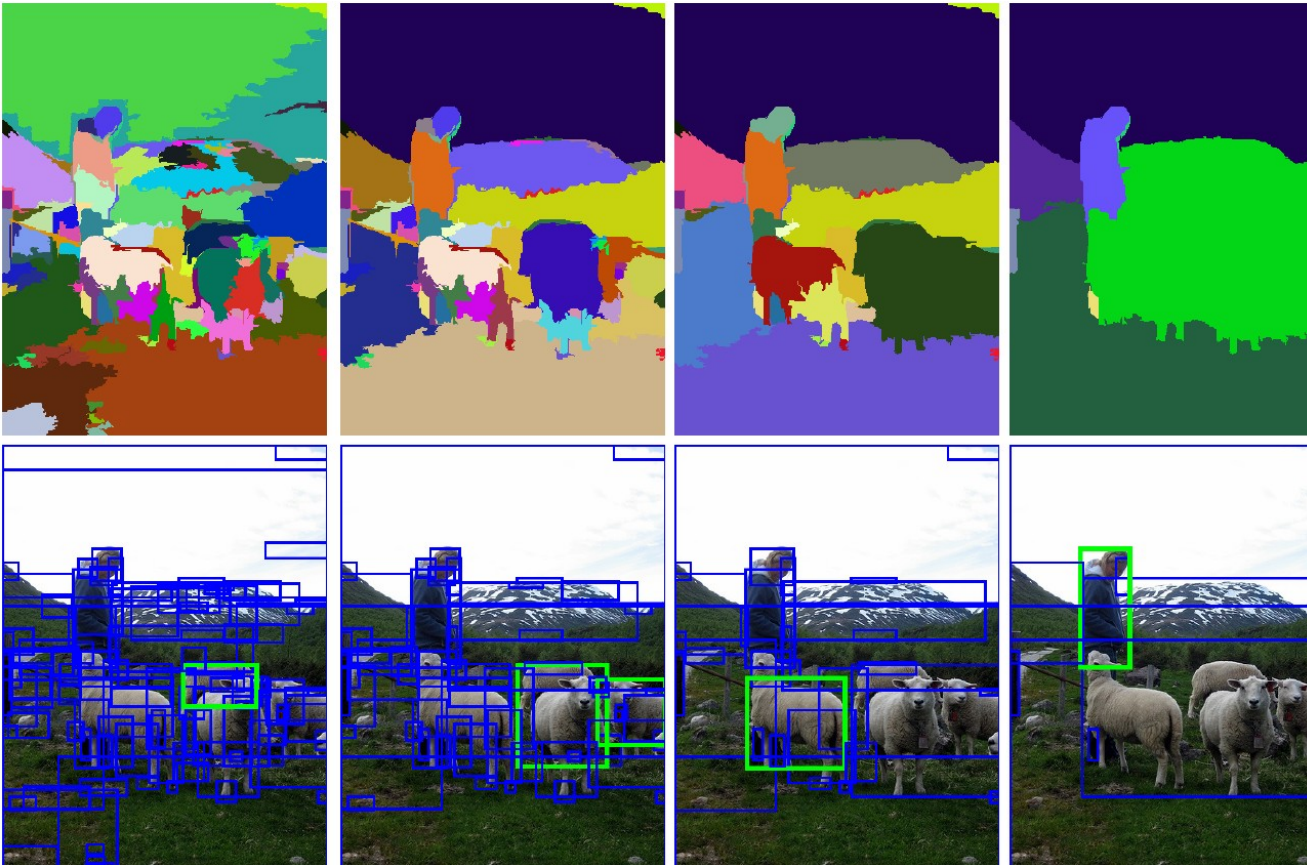


Use image structure to propose likely object regions.

Proposals are approximate boxes without category labels.

Selective Search

Small regions → larger groups



Start with small image regions

Colors mark regions, not classes.

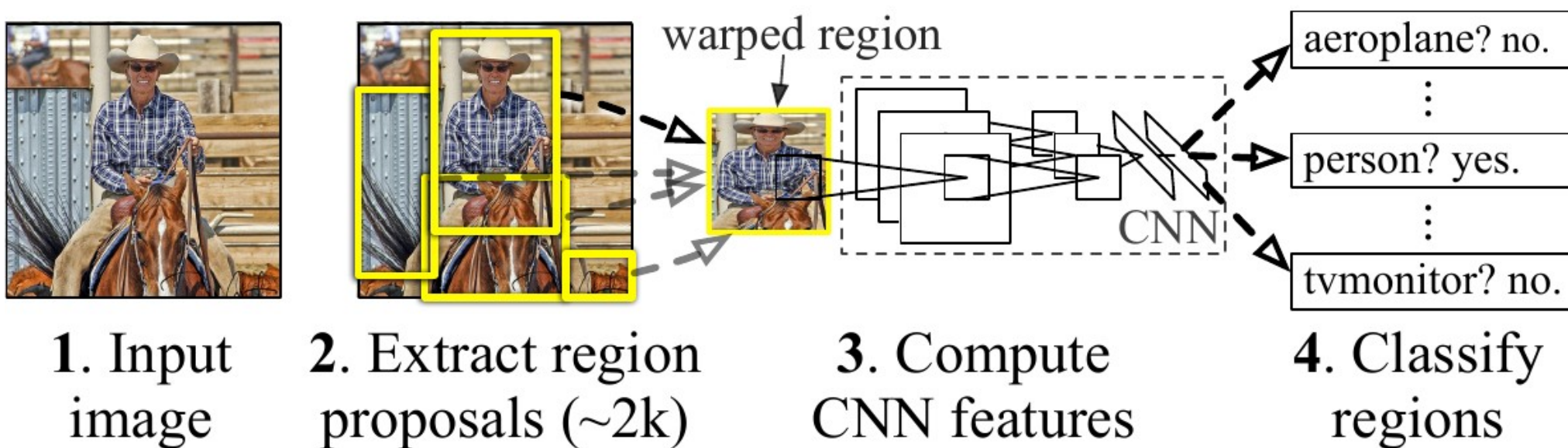
Merge neighboring regions with similar color or texture.

Enclose each region in a box.
Keep boxes from several stages.

Top: grouped regions
Bottom: enclosing boxes

R-CNN

R-CNN: *Regions with CNN features*

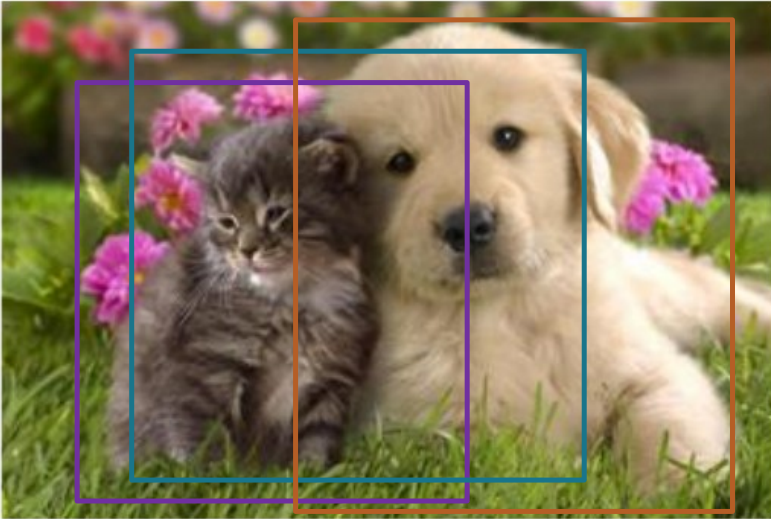


Crop and resize each proposal, then run the same CNN on each crop.

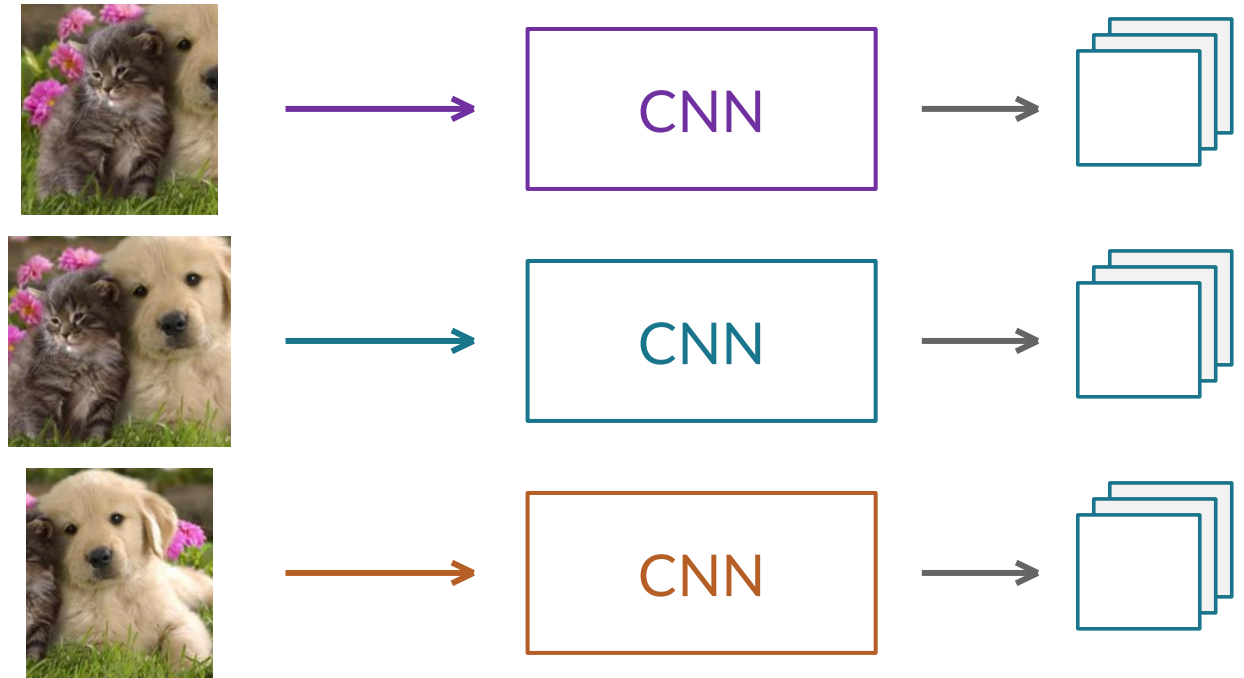
The features support category prediction and box refinement.

Repeated convolution in R-CNN

Overlapping proposals



Crop and resize each proposal

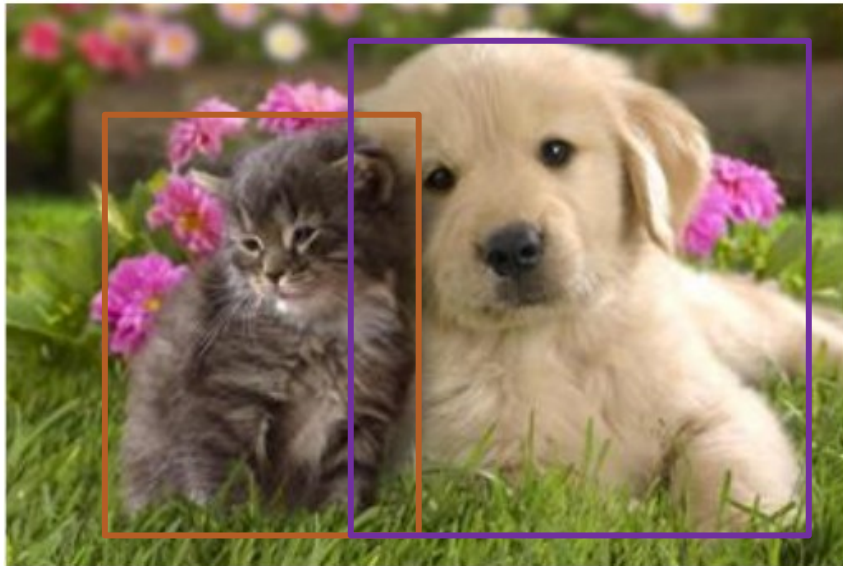


The CNN weights are shared, but overlapping regions are processed repeatedly.

Sharing image features across proposals

Run convolution on the whole image once. Each proposal then selects the corresponding region on the resulting feature map.

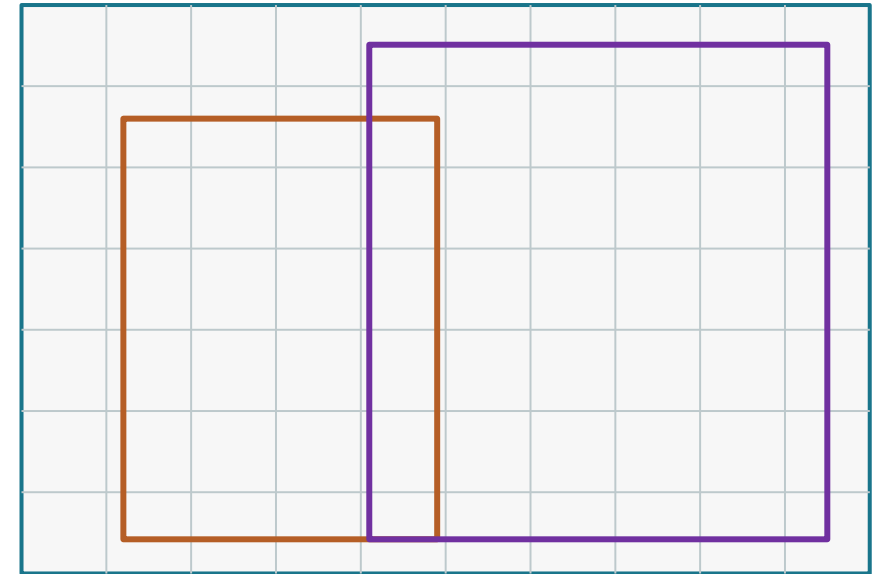
Image and proposal coordinates



One CNN pass



Shared feature map

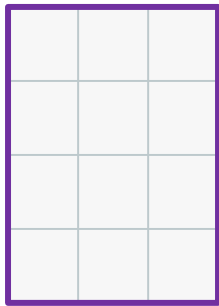
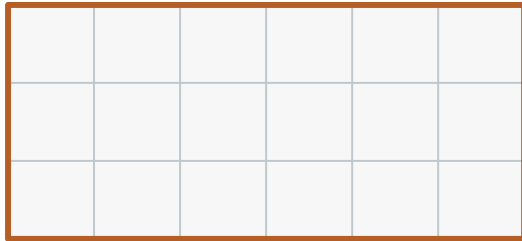


RoI (region of interest): the feature region selected by a proposal.

Fixed-size inputs for the region head

Different-sized regions give vectors of different lengths when flattened.
Fully connected layers require the same input length for every proposal.

Selected feature regions

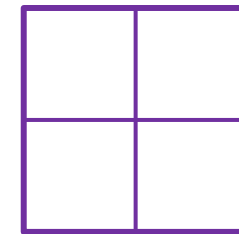
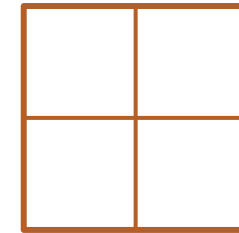


One channel shown

Rol pooling



Fixed output size

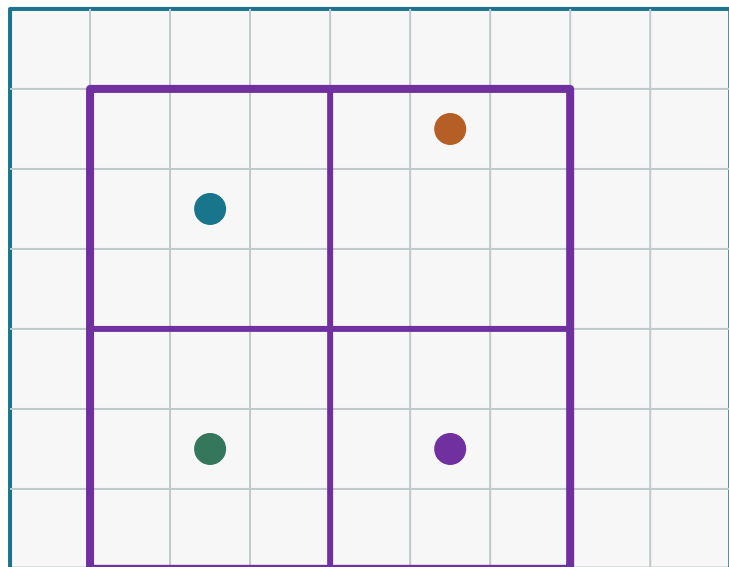


Same grid size → same vector length

Roi pooling

Divide the selected Roi into a fixed grid of bins.
Take the maximum feature value in each bin.

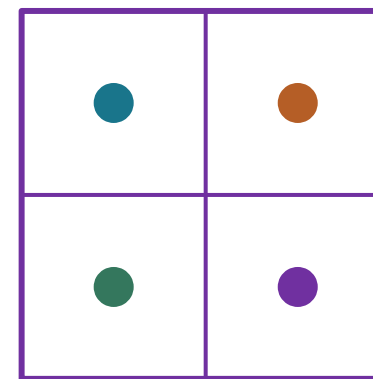
Selected Roi in one channel



Max in each bin



Pooled features

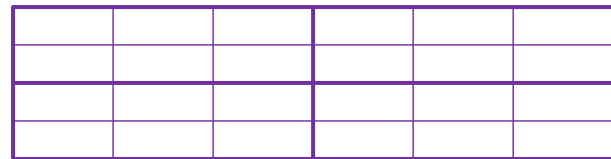
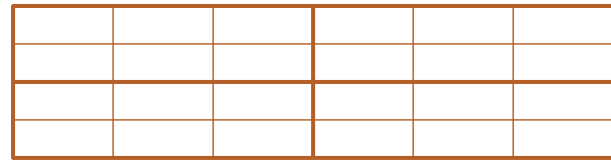
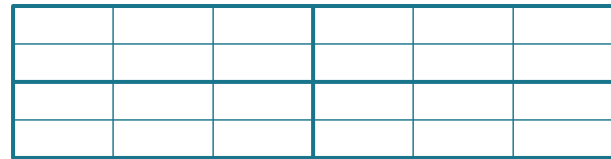


Dots mark the selected maxima. The 2×2 output is just an example.

RoI pooling across feature channels

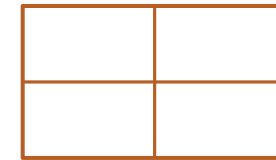
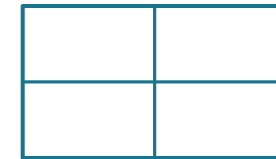
Pool every feature channel separately. The channel count stays the same.

Many values within each channel



$$C \times h_{\text{RoI}} \times w_{\text{RoI}}$$

Fixed grid in each channel

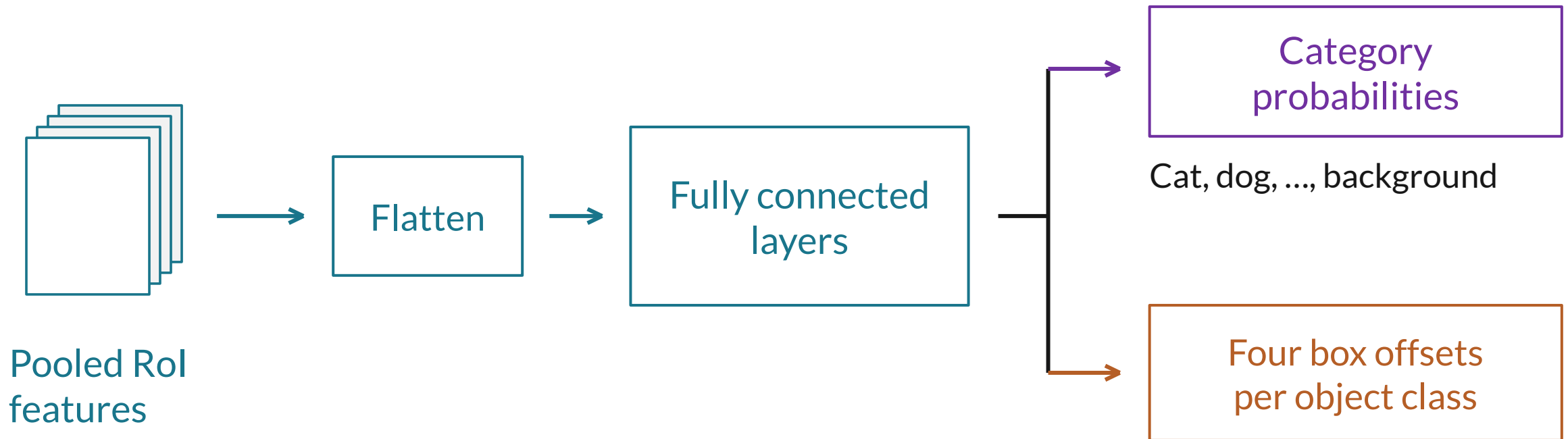


$$C \times 2 \times 2$$

C: channels h, w: region height and width

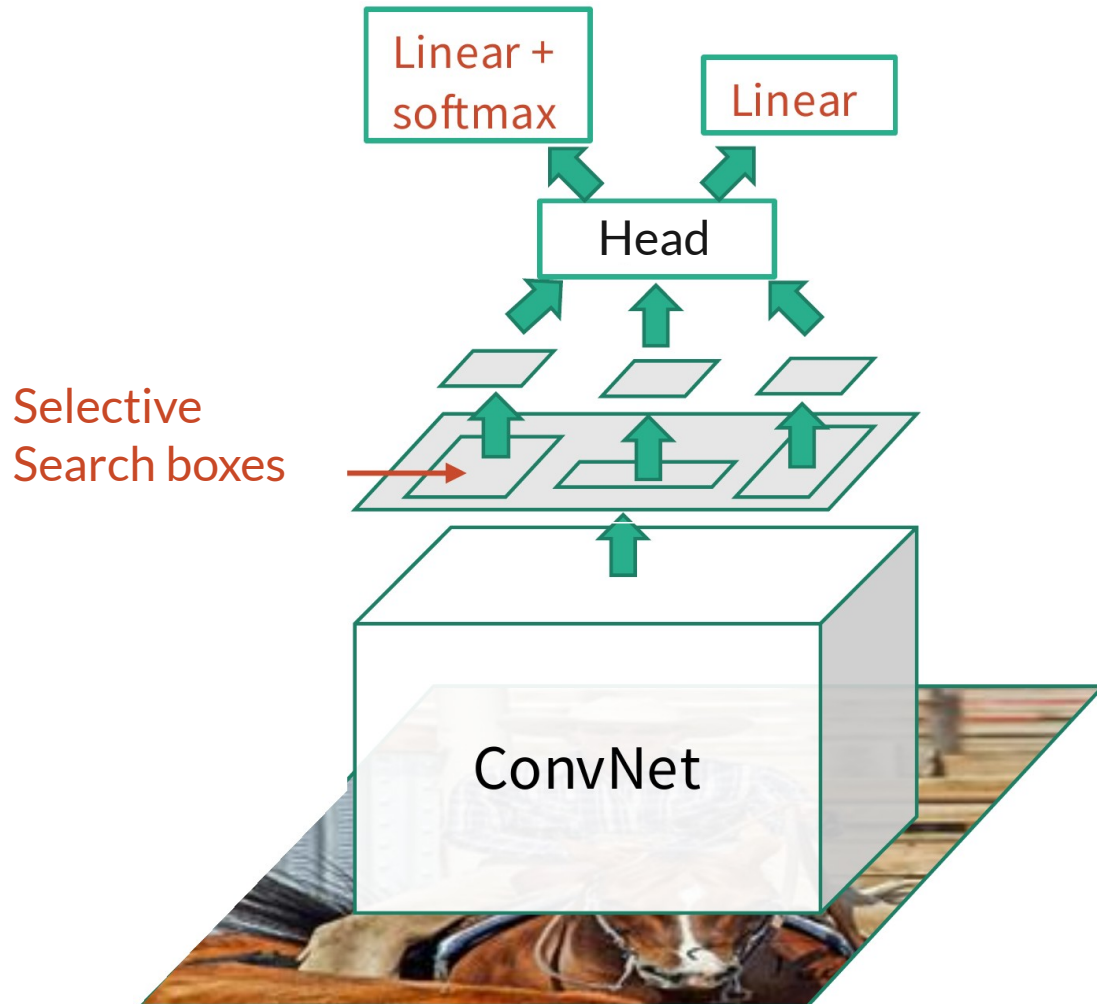
Classification and box refinement

Apply the same prediction layers separately to each RoI.



Box offsets adjust the proposal's center, width and height.

Fast R-CNN



For each region

Category or background
+ box correction

Rol pooling

Extract fixed-size features
at each proposed region.

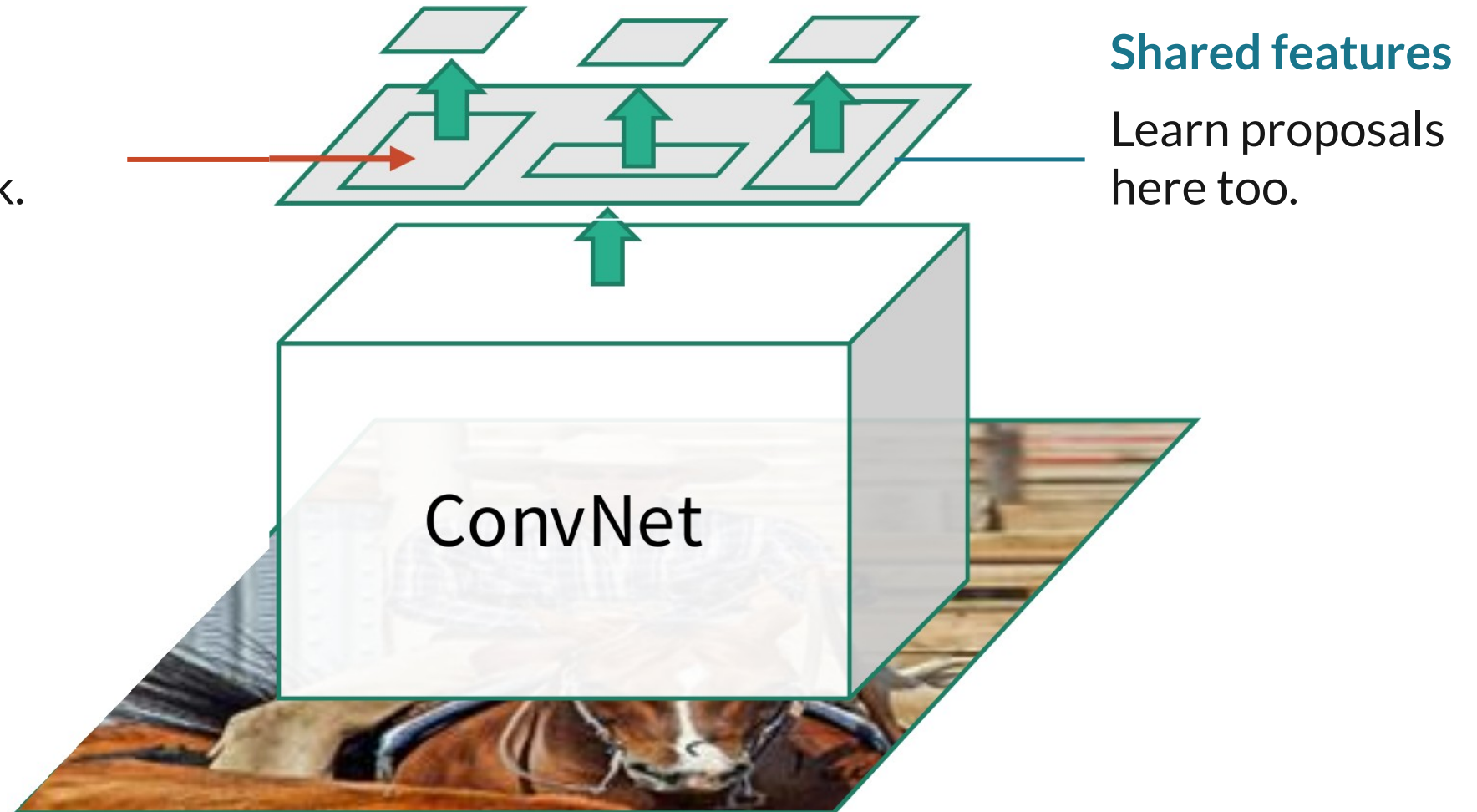
Shared image features

Compute once per image.

Remaining proposal bottleneck

Selective Search

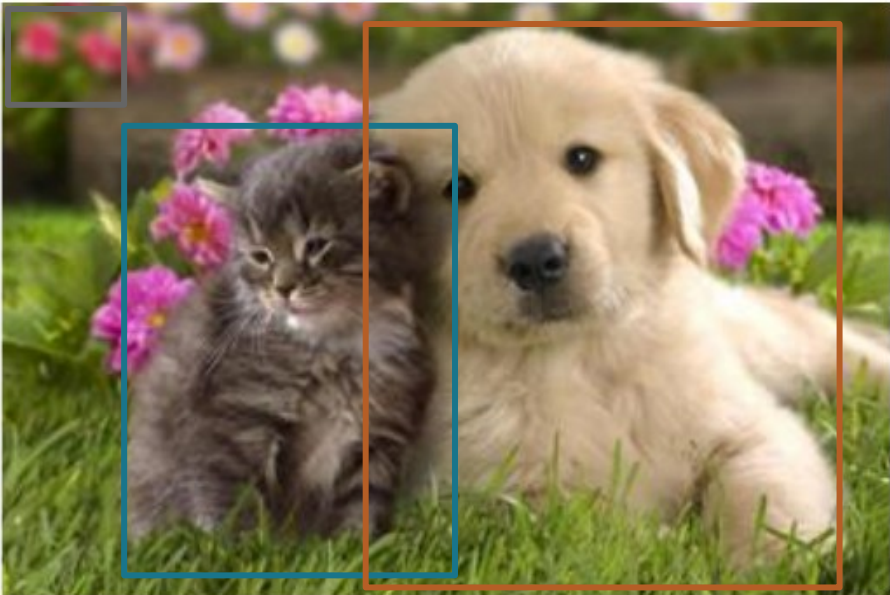
Still a costly search outside the network.



Objectness

A binary classification score: object or background

Select likely object regions before predicting their categories.



Cat region

Dog region

Background

Object

Object

Background

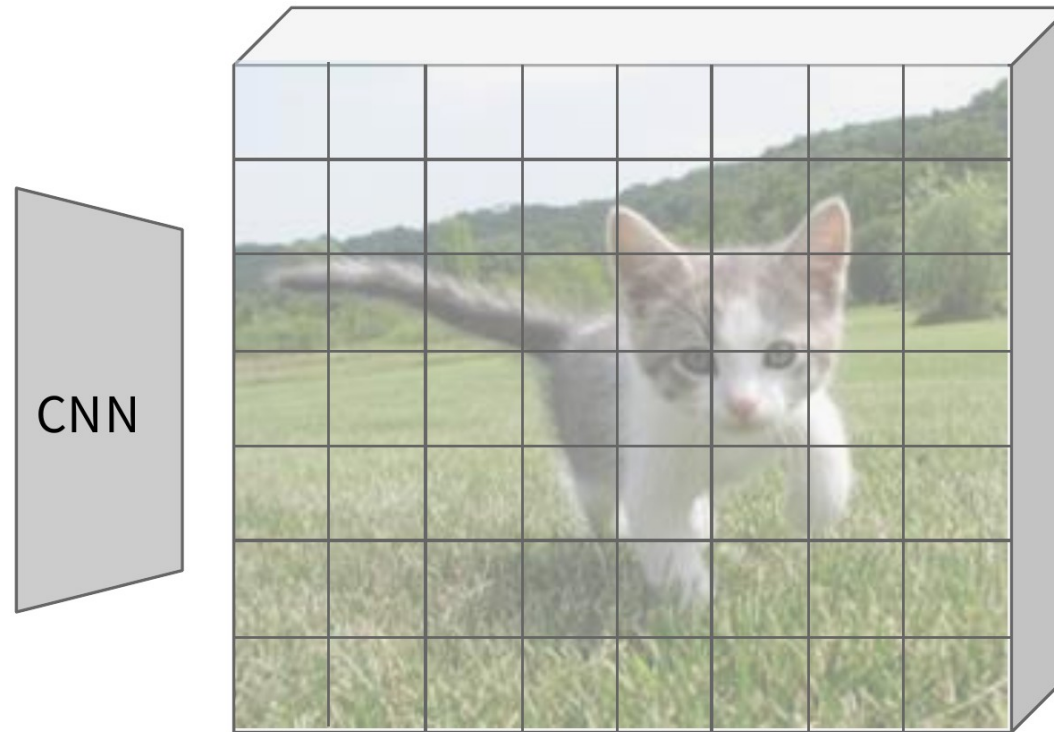
Cat and dog have the same binary target. Their categories come later.

Region proposal network (RPN)

Predict candidate boxes from the shared CNN feature map.



Input image



Shared feature map

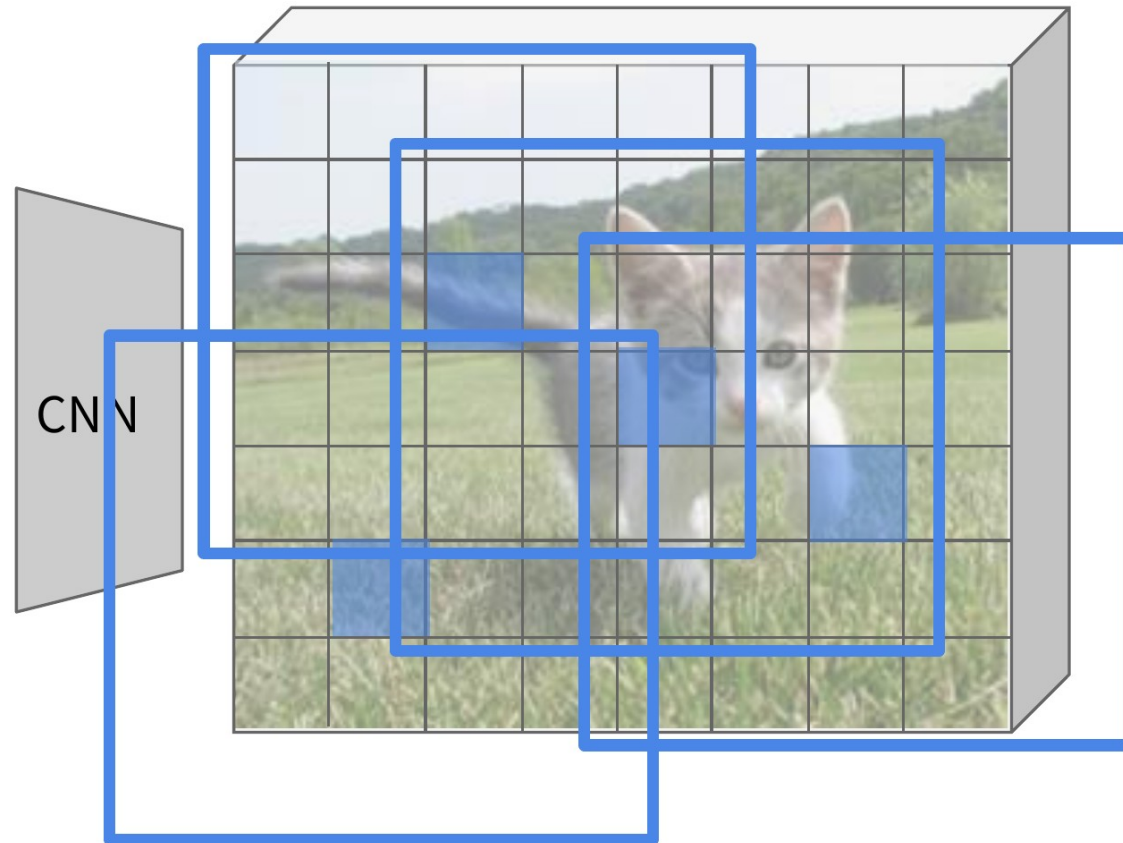
Image overlay shows spatial locations

Anchor boxes

Anchors are predefined starting boxes, repeated across feature locations.



Input image

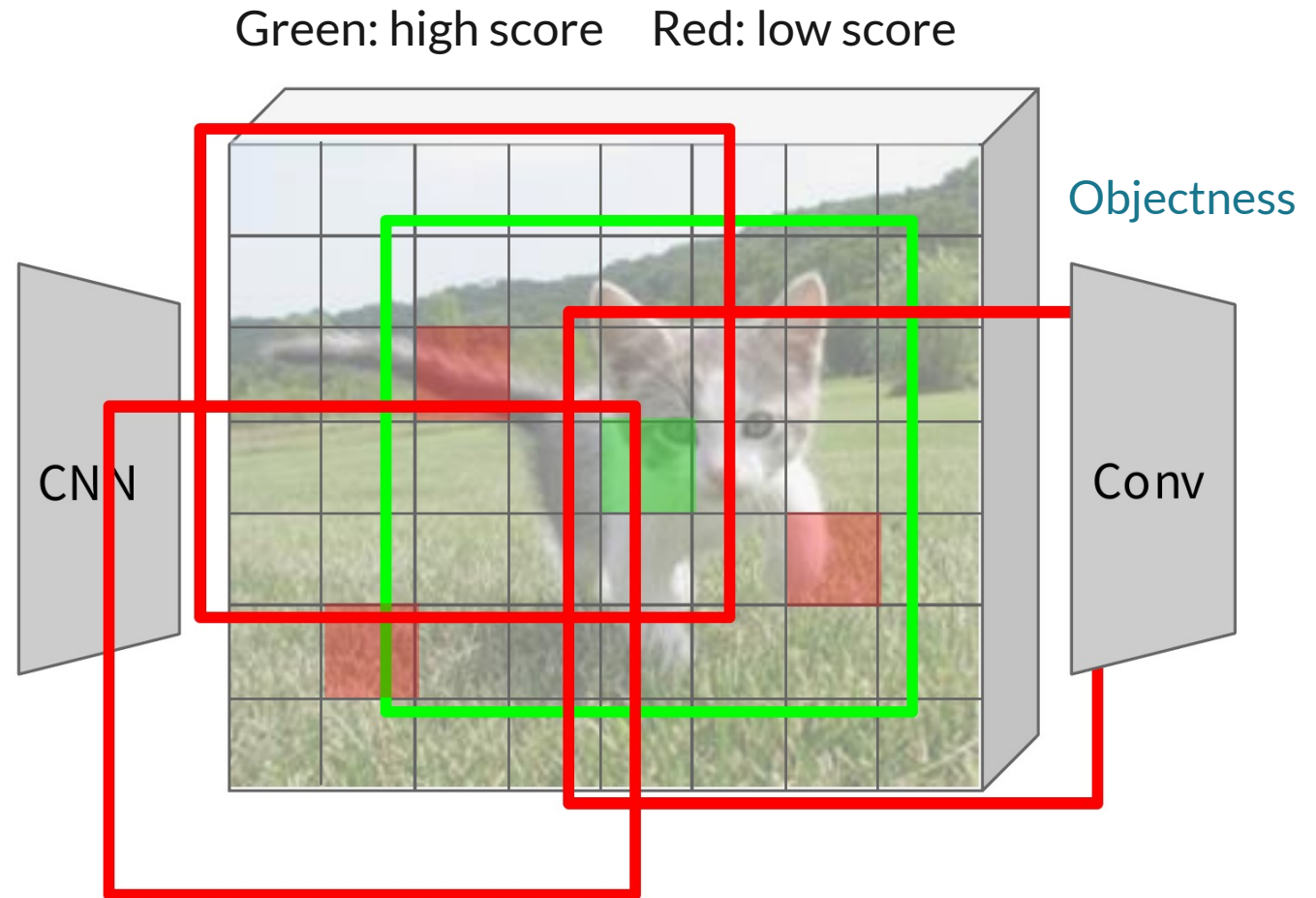


Objectness at each location

Predict an objectness score for each anchor.



Input image

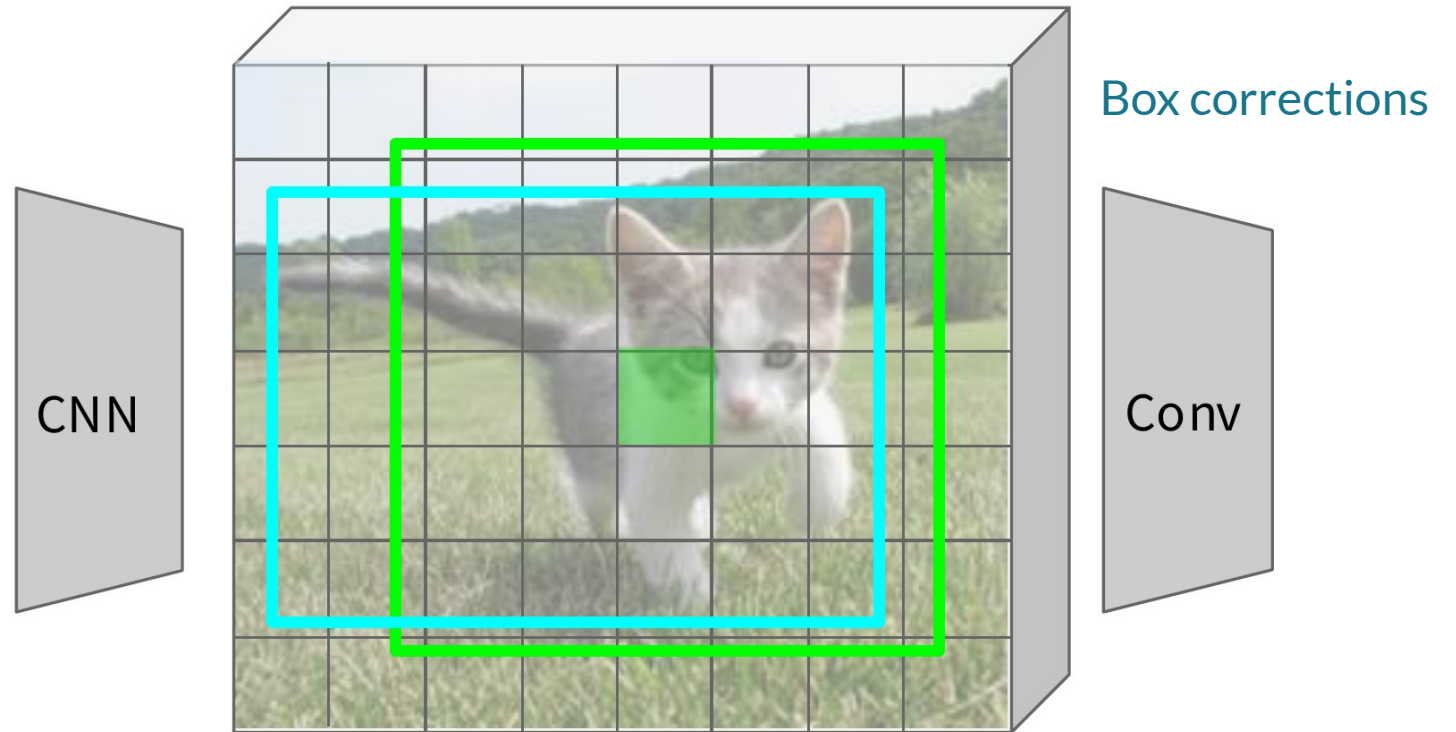


Correcting an anchor

Predict four corrections for each anchor:
shift its center and change its width and height.



Input image



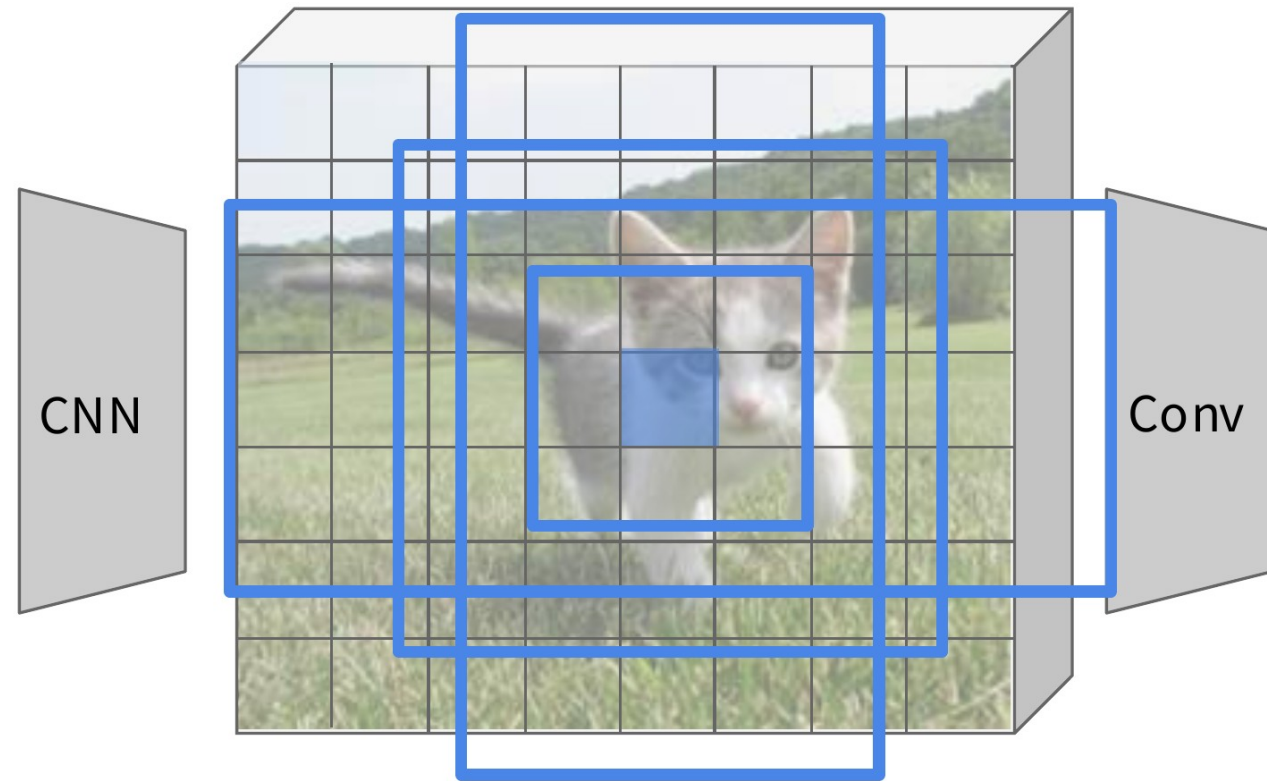
Green: anchor Cyan: training target

Anchor scales and aspect ratios

Tall and wide objects need different starting boxes.
Use several sizes and aspect ratios at each location.

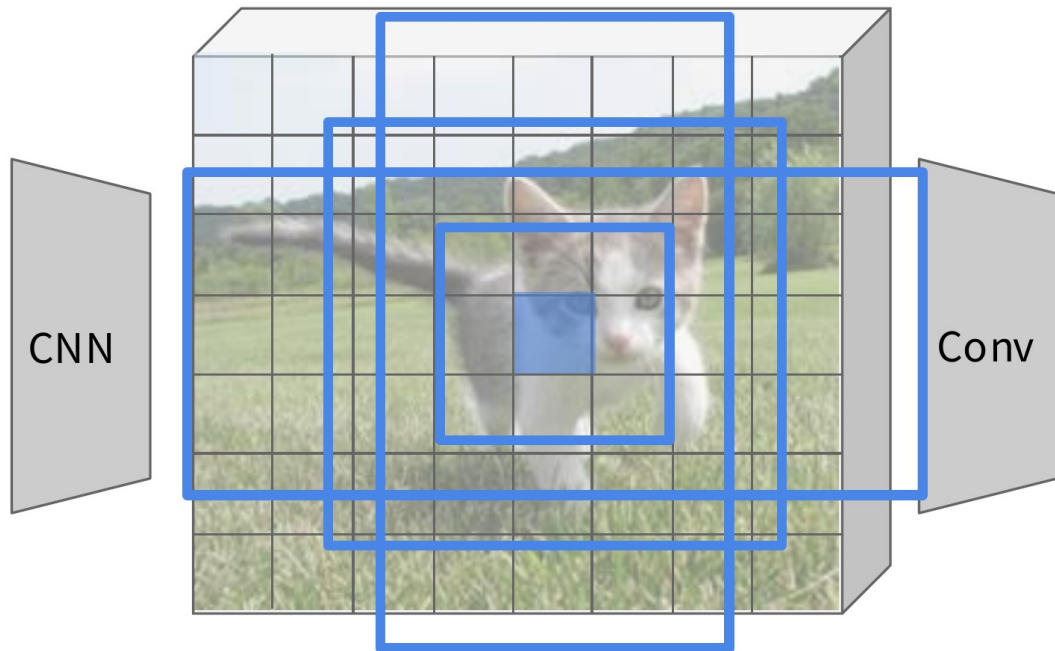


Input image



Several anchors share one center

RPN outputs



At every feature-map location

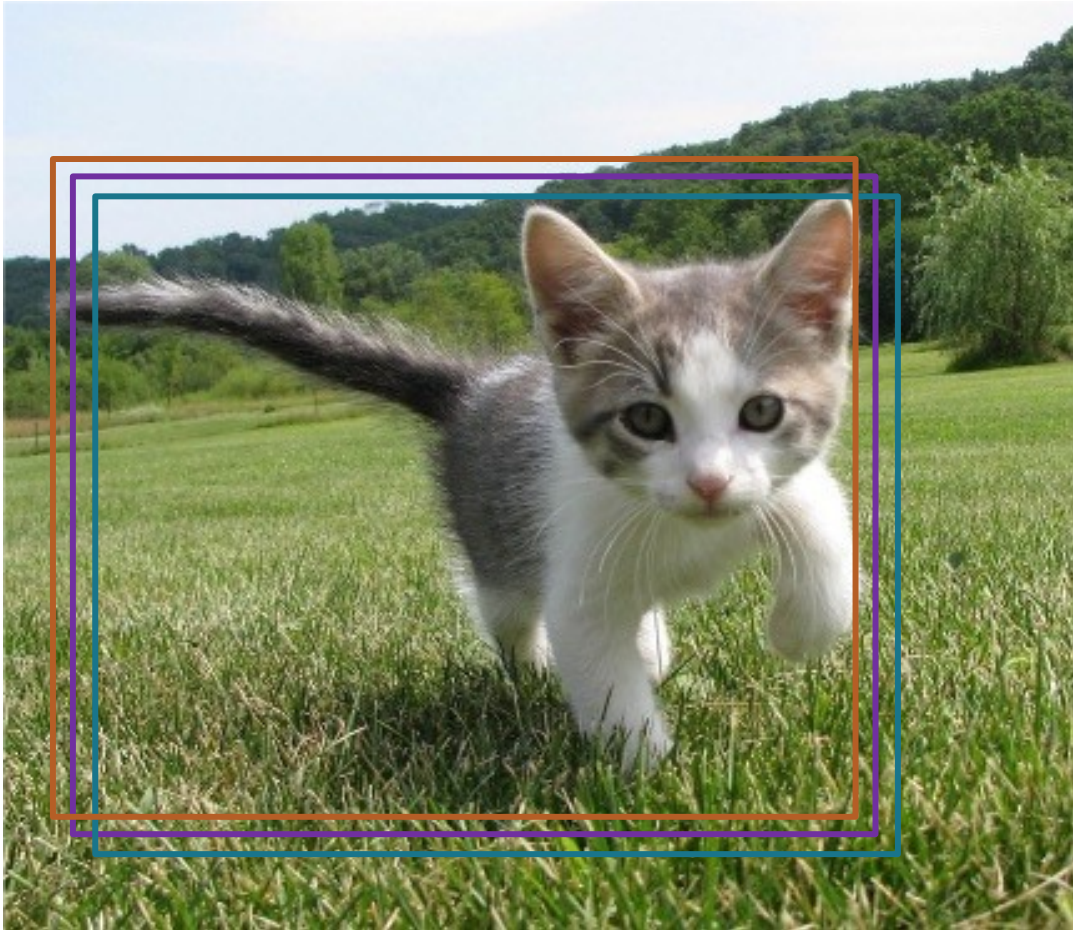
For each anchor shape:

One objectness score

Four box corrections

Shared convolutional head

Duplicate detections



Source: [CS231n](#)

Three boxes, one cat

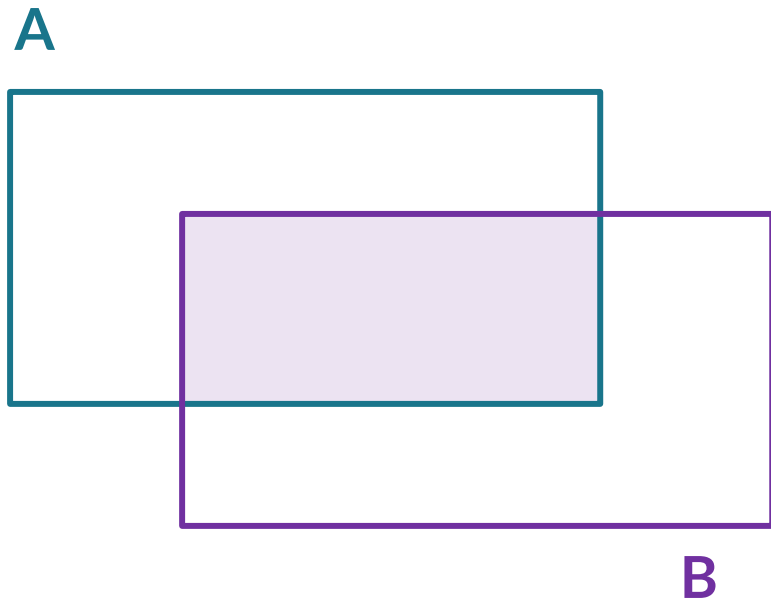
Candidate scores

$$s_A > s_B > s_C$$

— A — B — C

Keep one box for each object.

Intersection over union for boxes



Intersection

Area inside both boxes

Union

Area inside either box

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Non-maximum suppression (NMS)

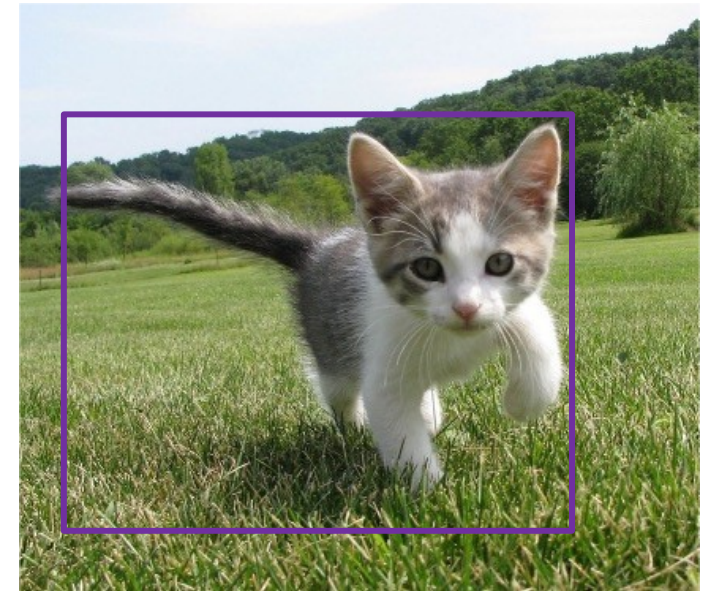
Greedy NMS

Input: scored boxes and IoU threshold τ

While candidates remain:

1. Keep the box with the highest score.
2. Remove it from the candidates.
3. Suppress boxes with $\text{IoU} > \tau$ against the kept box.

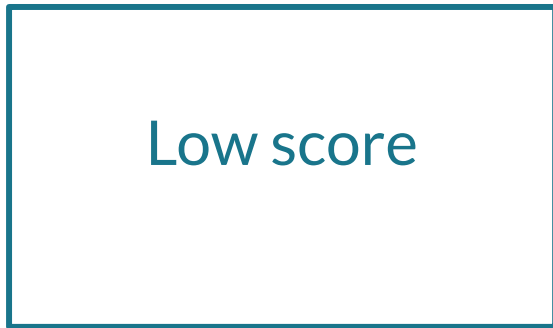
Retain the best-scoring box



Proposal NMS: all candidates
Final NMS: one class at a time

Confidence filtering and overlap suppression

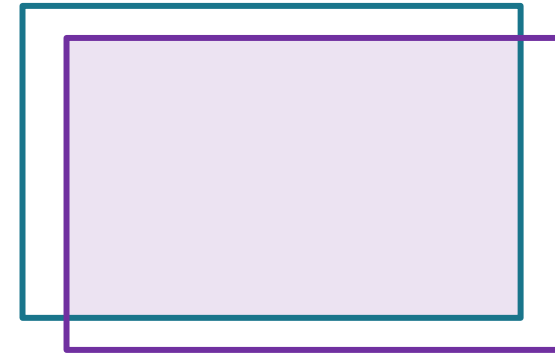
Score threshold



Reject a weak candidate

No neighboring box is needed.

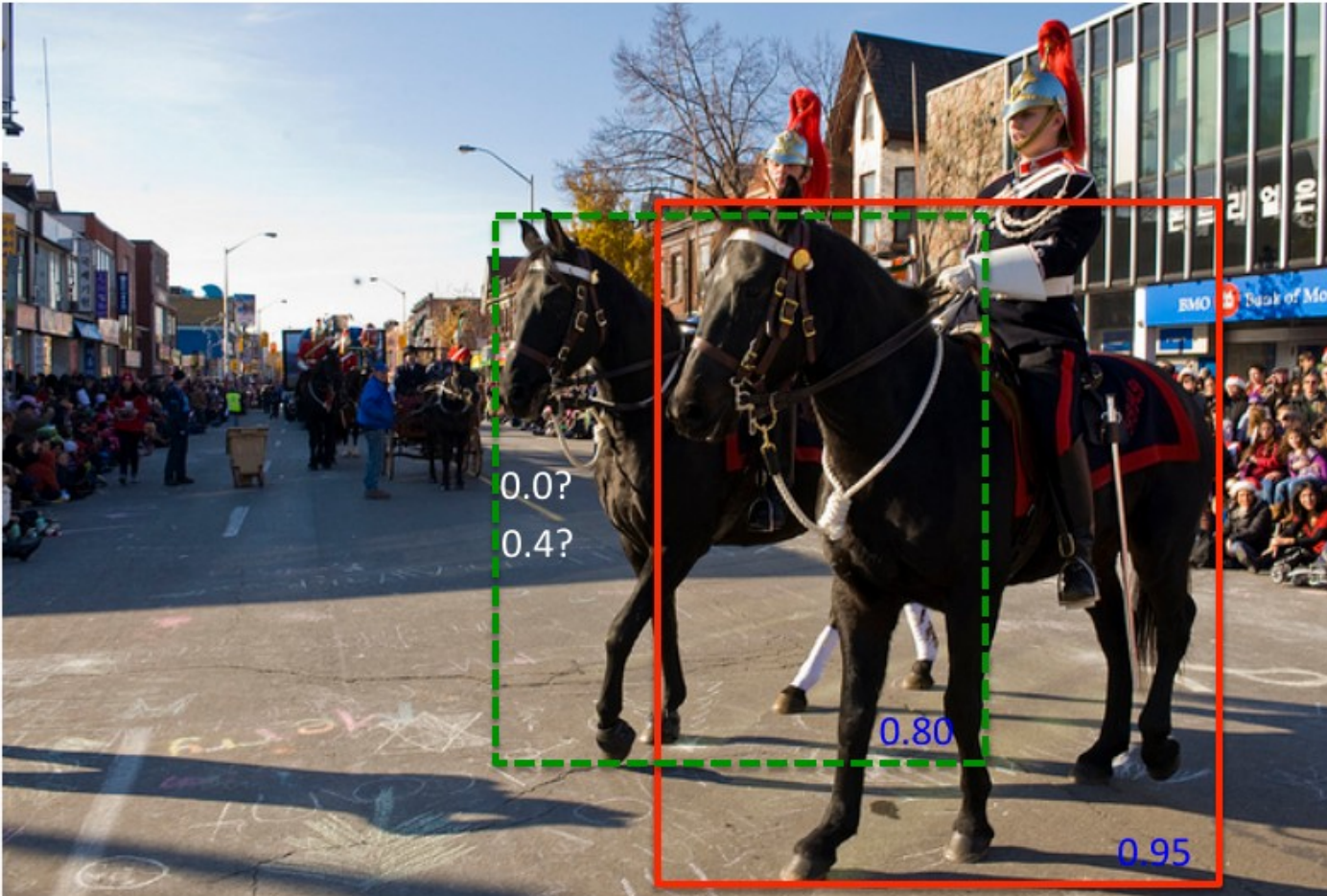
NMS overlap threshold



Suppress a strong duplicate

Requires comparison with a kept box.

Overlapping objects



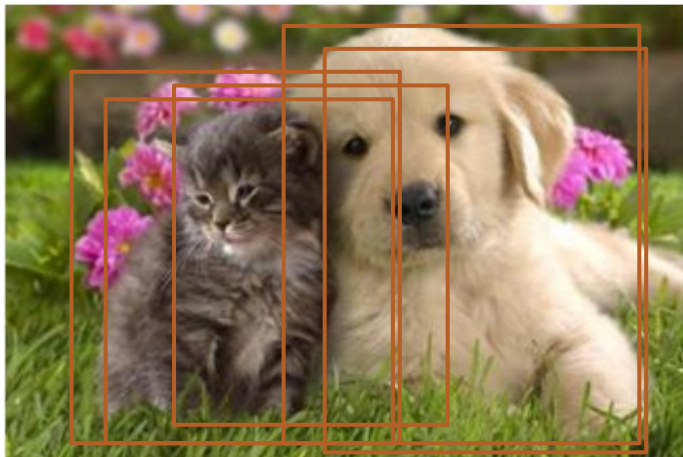
Two overlapping horses

Hard NMS may remove a genuine object.

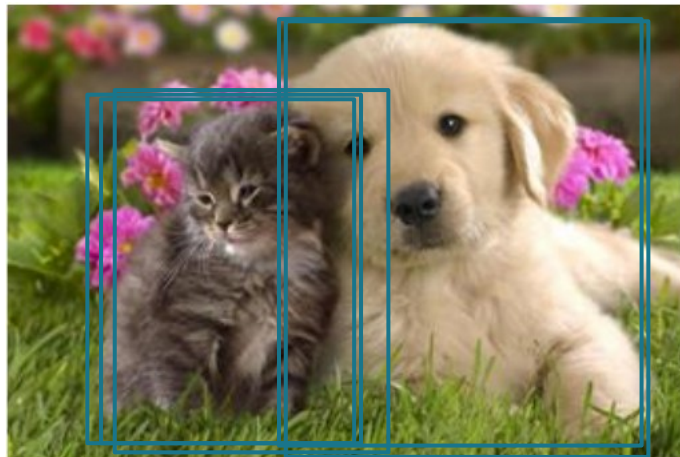
Soft-NMS reduces scores instead of immediately deleting overlapping boxes.

From anchors to proposals

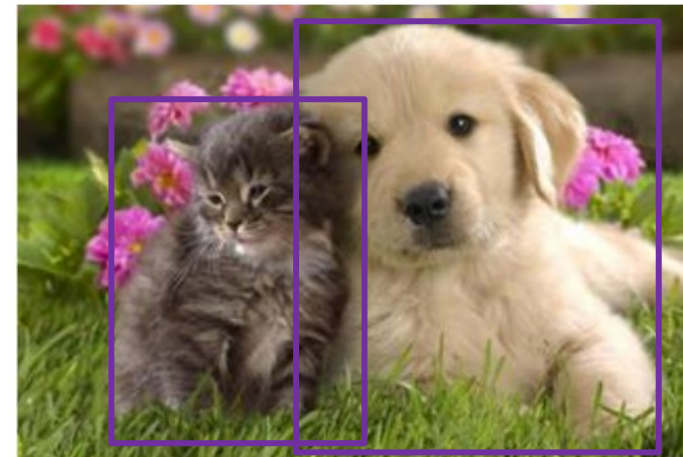
Reference boxes



Corrected candidates



Selected proposals



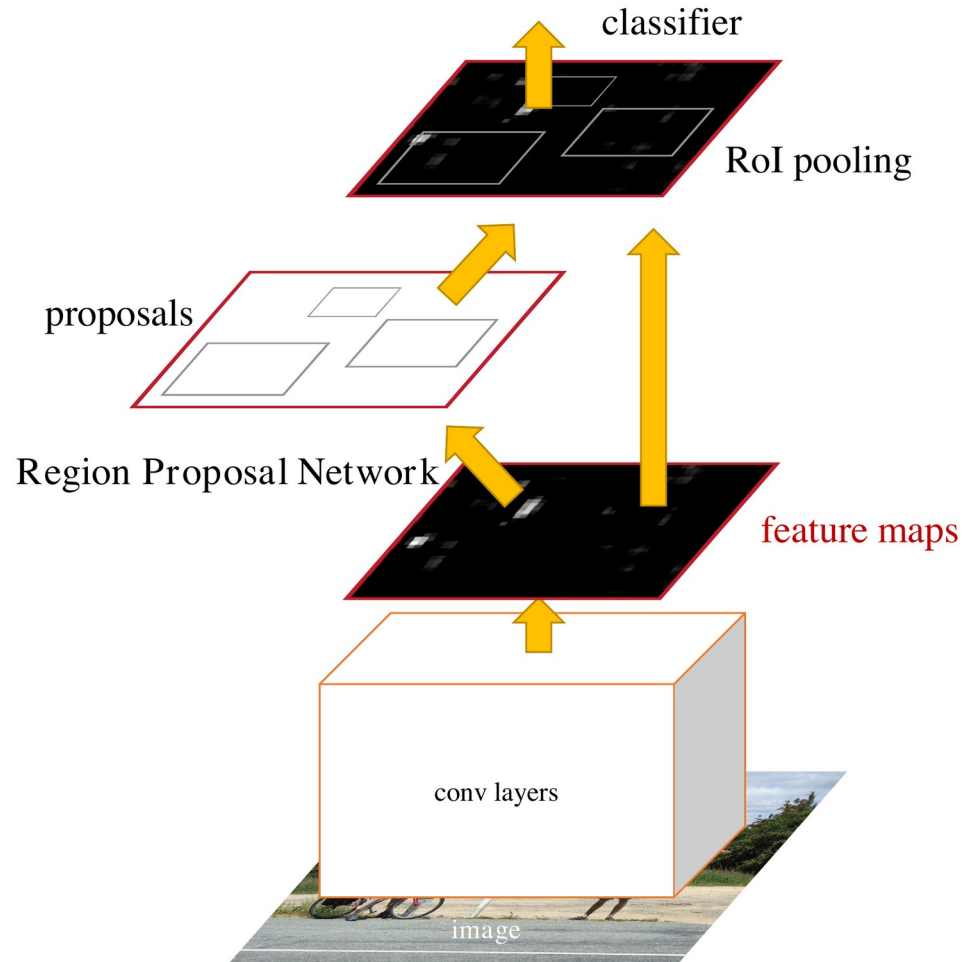
Apply predicted offsets

Rank by objectness

Select with NMS

Proposal NMS ignores category: the RPN has not predicted one.

Faster R-CNN



2 Classify and refine

Pool features for each proposal.
Predict category and a refined box.

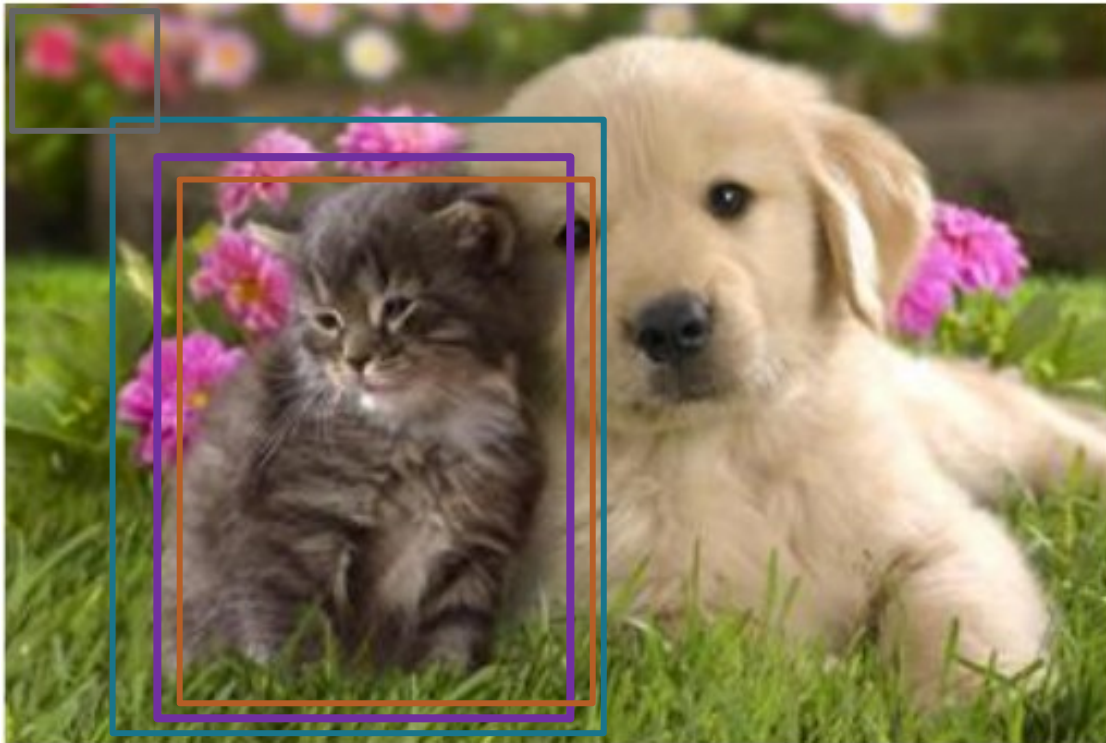
1 Propose regions

RPN predicts objectness
and corrected boxes.

Both stages use the same image features.

Assigning targets to proposals

Match proposals to annotated objects



□ Ground-truth cat box

Positive proposals

Sufficient overlap with the cat

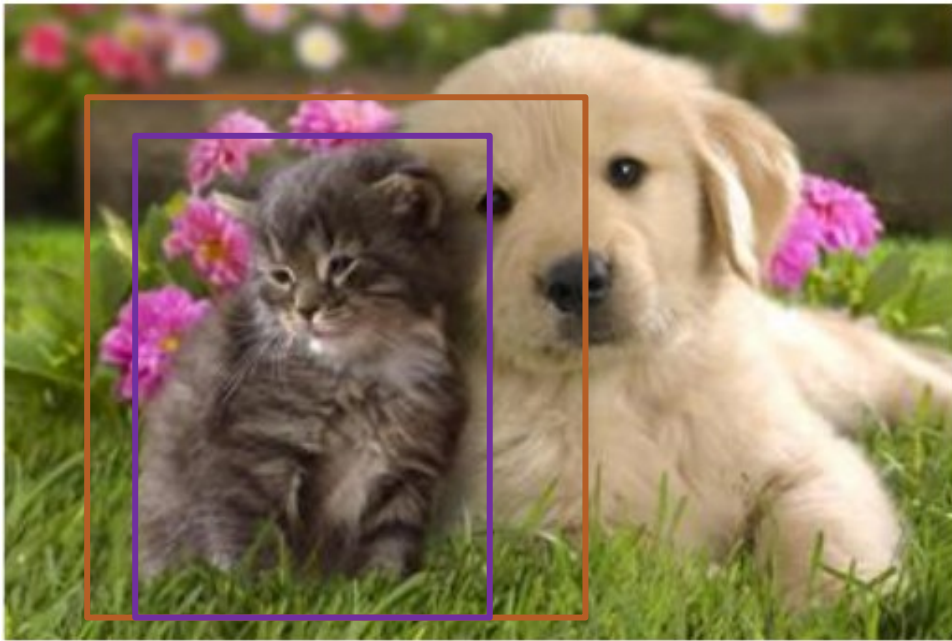
Target: cat + its annotated box

Background proposal

Target: background class only

Several proposals can share one target.

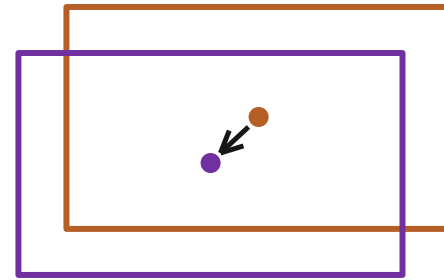
Bounding-box regression



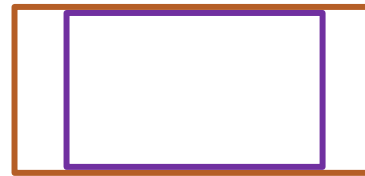
 Proposal

 Target

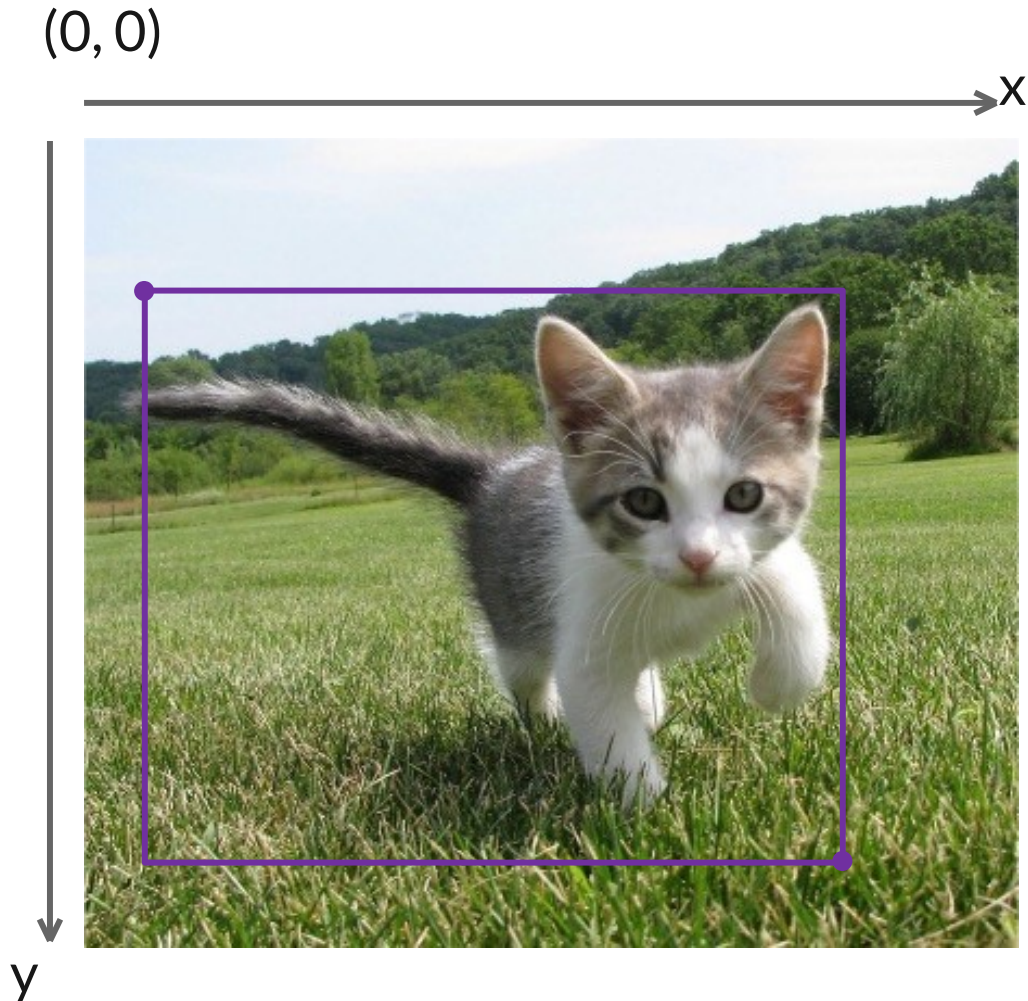
Move the center



Change width and height



Bounding-box coordinates



Top-left corner

$$(x_{\min}, y_{\min})$$

Bottom-right corner

$$(x_{\max}, y_{\max})$$

$$w = x_{\max} - x_{\min}$$

$$h = y_{\max} - y_{\min}$$

Relative box offsets

Center shifts

$$t_x^* = \frac{x^* - x_0}{w_0}$$

$$t_y^* = \frac{y^* - y_0}{h_0}$$

0: starting box *: target box
x, y: center coordinates
w, h: width and height

Size changes

$$t_w^* = \log \frac{w^*}{w_0}$$

$$t_h^* = \log \frac{h^*}{h_0}$$

Zero offsets leave
the starting box unchanged

Training the region detector

Positive region

$$L = L_{\text{cls}}(p, y) + \lambda L_{\text{box}}(t, t^*)$$

Background region

$$L = L_{\text{cls}}(p, y)$$

Classification loss: cross-entropy.

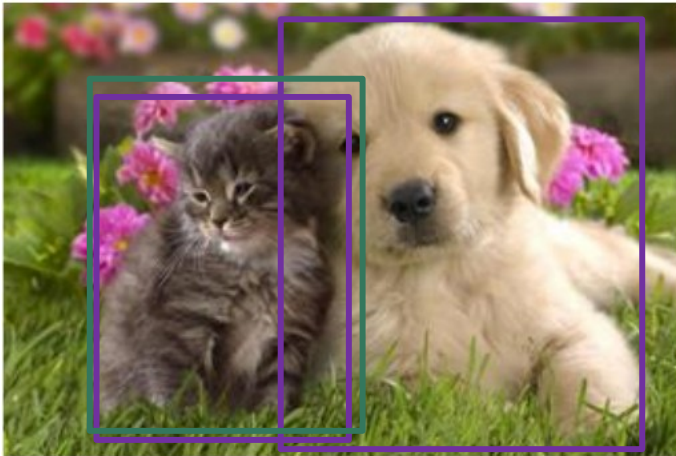
Smooth L1 box loss: quadratic for small offset errors,
linear for larger errors. Background has no box target.

p : class probabilities y : target class

t : predicted offsets t^* : target offsets λ : box-loss weight

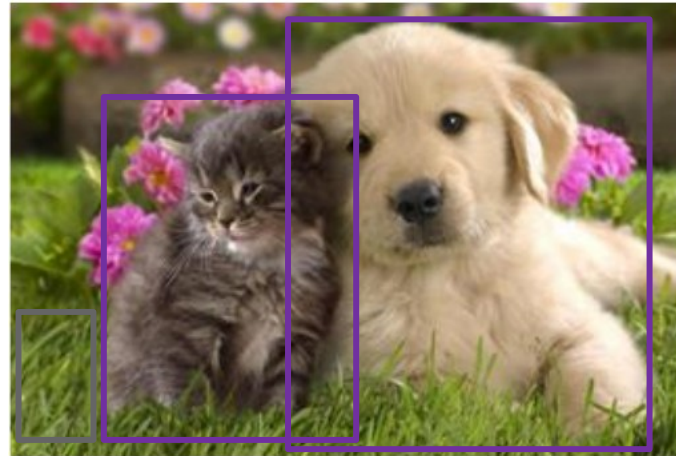
Assigning targets to anchors

Positive



High overlap
Learn objectness + box

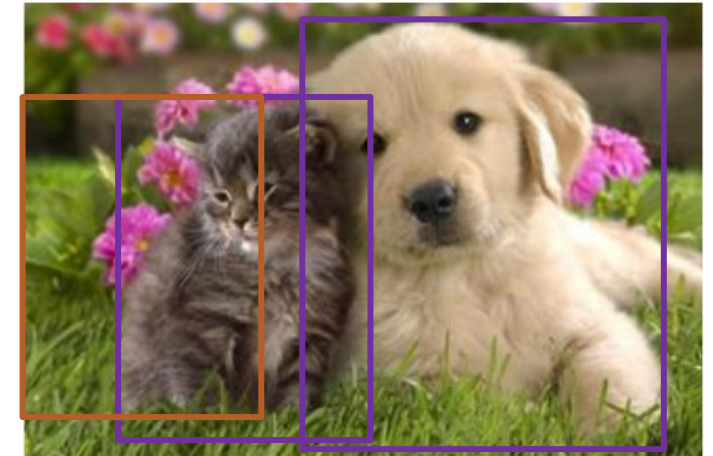
Background



Low overlap
Learn background

Ignore

☐ Annotated objects

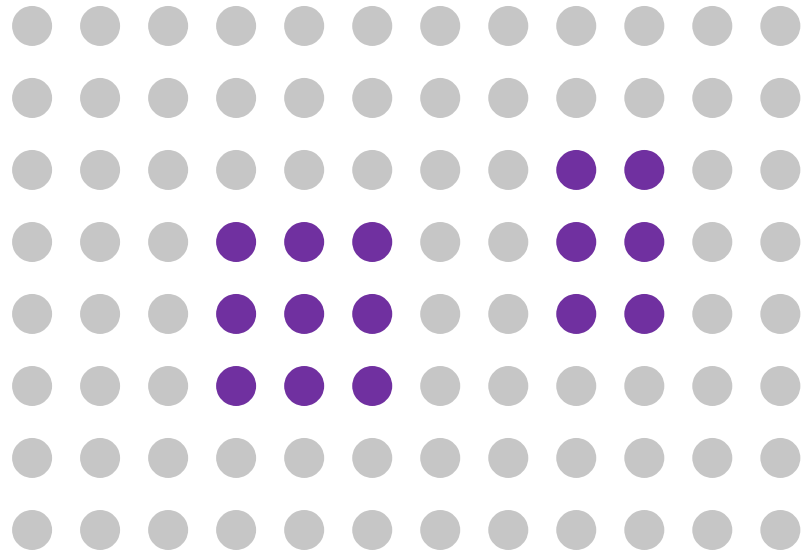


Intermediate overlap
Omit from this loss

Also mark each object's highest-overlap anchor(s) as positive.

Foreground and background training examples

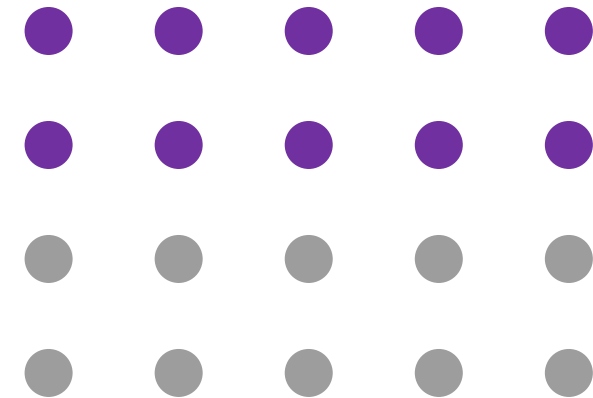
Dense candidates



Sample



Training subset



Limit background examples so they do not dominate the loss.

● Positive: objectness + box loss

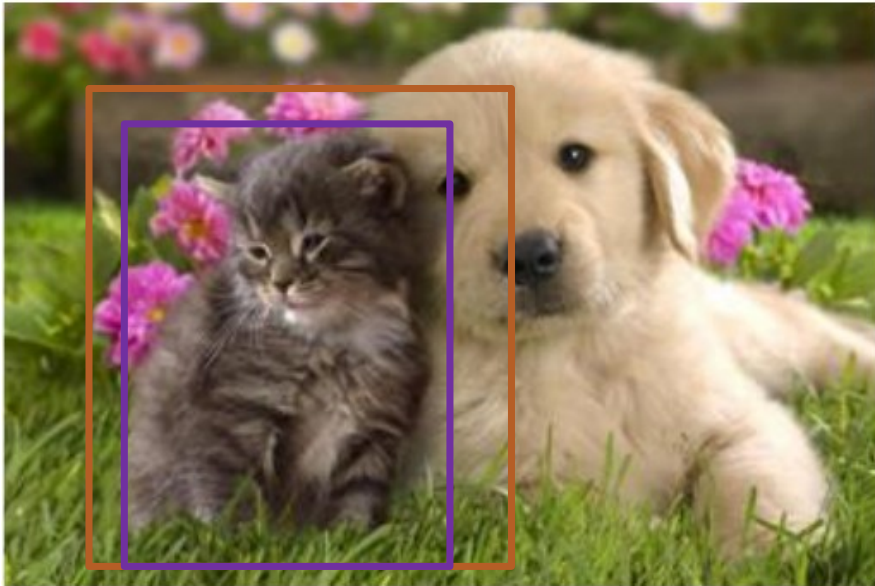
● Negative: objectness loss

Training Faster R-CNN

RPN

Anchor

Annotated box



Objectness loss (target: object)

Box loss: fit the annotated box

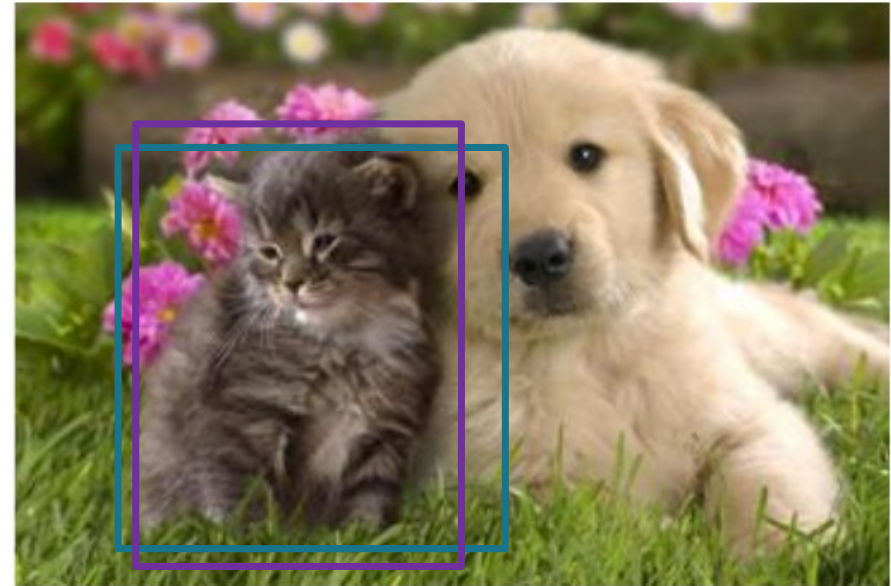
All four losses train the shared CNN.

Source: [CS231n](#)

Region detector

Proposal

Annotated box



Category loss (target: cat)

Box loss: fit the annotated box

Lecture outline

01 The object detection problem

02 Region-based detection

| 03 **Detection evaluation**

04 Dense and multiscale detection

05 Instance segmentation

Matching predictions to objects



Annotated target

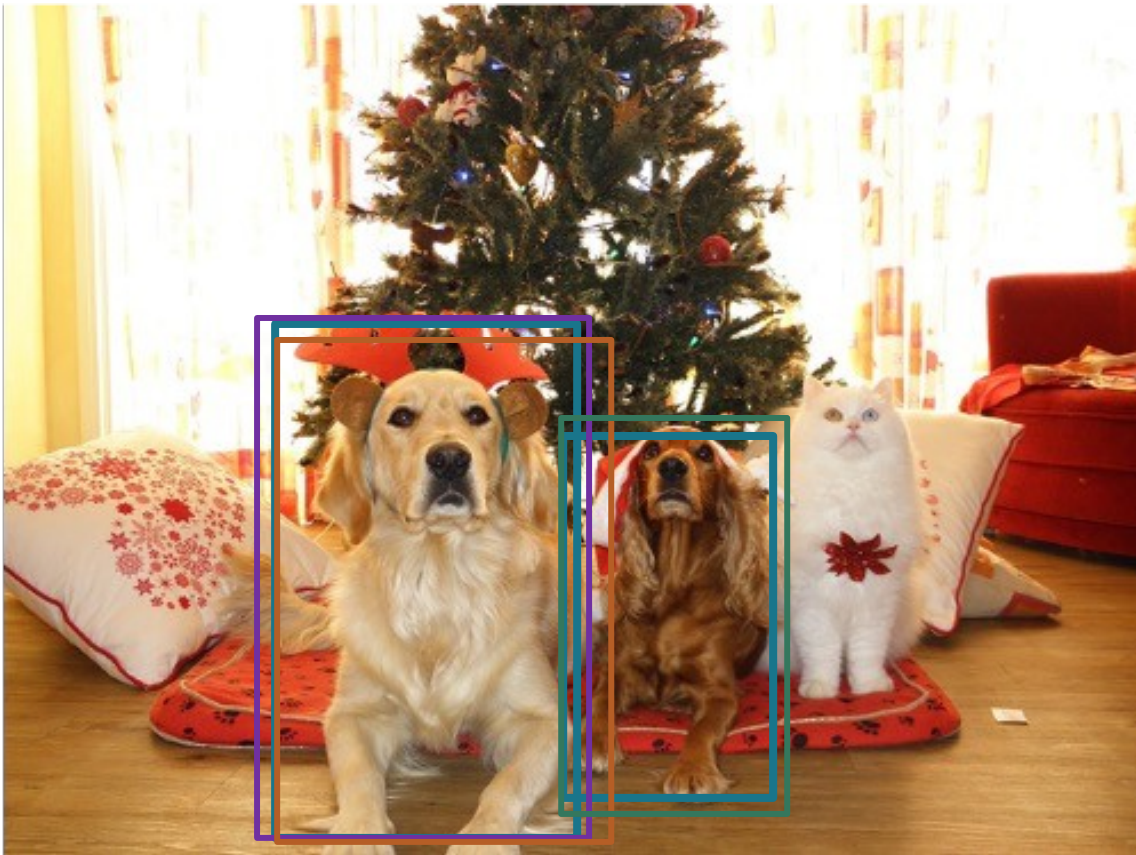
Dog + reference box

Predicted detection

Same image and category
Sufficient box overlap

One target can count only once.

One-to-one matching



Same image and class

Visit detections in score order

- 1 First dog matched
True positive
- 2 Same dog again
False positive
- 3 Second dog matched
True positive

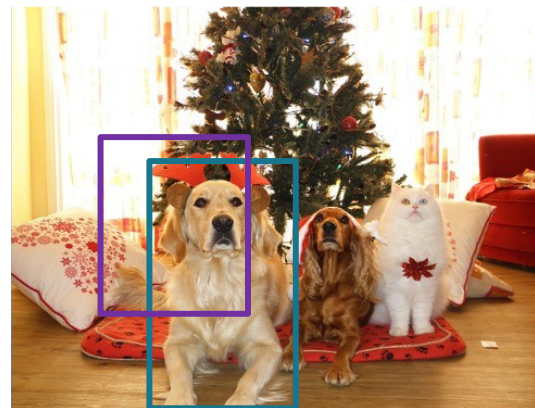
Detection errors

Wrong category



Predicted: cat
True class: dog

Poor localization



Too little overlap
with the true box

Duplicate



Two predictions
for the same dog

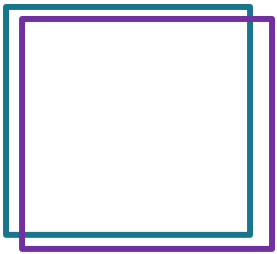
Background



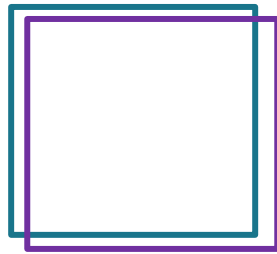
A box containing
only background

Precision and recall

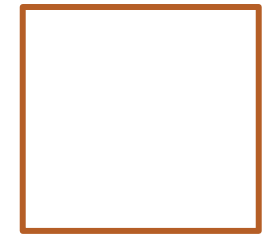
3 targets · 4 retained predictions



2 matched
True positives (TP)



1 missed
False negative (FN)



2 false alarms
False positives (FP)

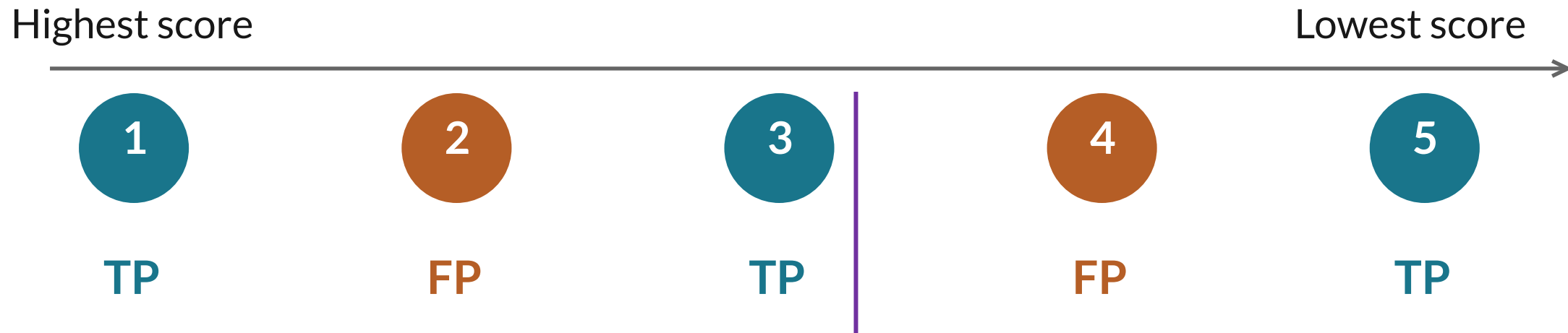
$$\text{Precision} = \frac{TP}{TP + FP} = \frac{2}{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{2}{3}$$

Ranked detections

One category · all evaluation images

Example: 3 annotated objects across the set



The score threshold keeps the first 3 predictions

$$\text{Precision} = \frac{2}{3}, \quad \text{Recall} = \frac{2}{3}$$

Precision-recall curve

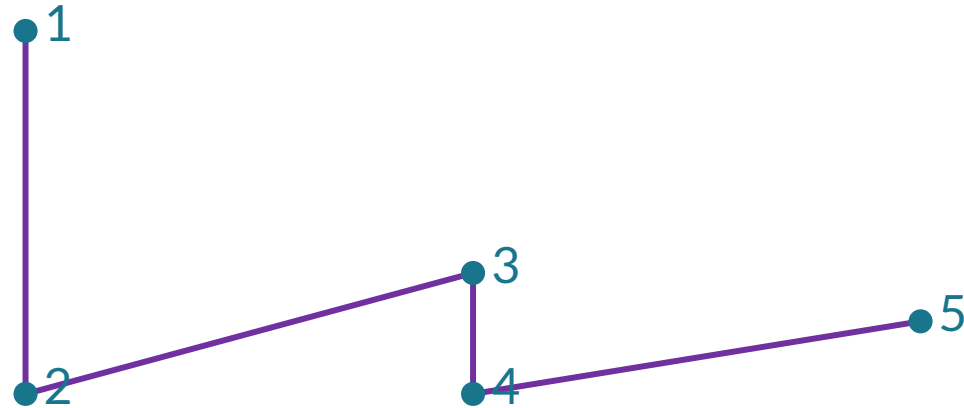
Precision

1

0

Recall

1



Add a false positive

Recall stays fixed;
precision falls.

Add a true positive

Recall increases.

Average precision (AP)

Precision



Area under curve

At each recall level,
use the best precision
at that recall or higher.

Larger area means
a better ranking.

Mean average precision (mAP)

Cat



AP_{cat}

Dog



AP_{dog}

Person

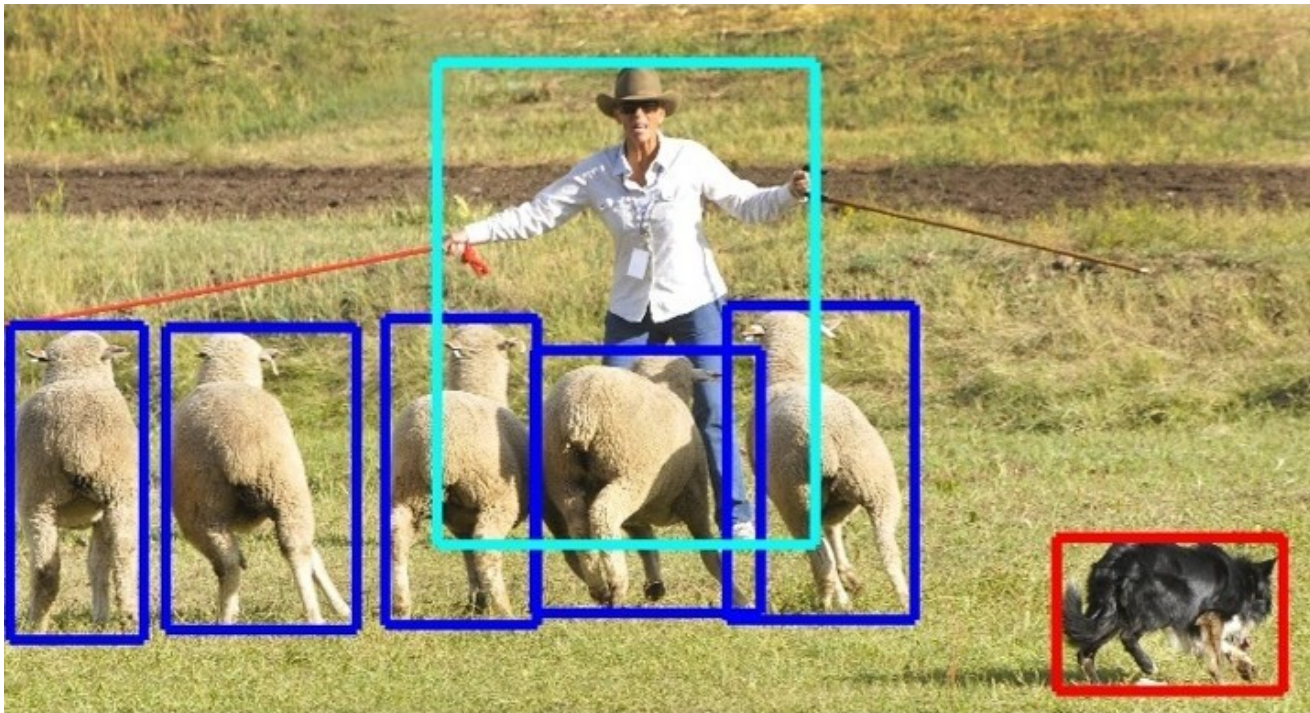


AP_{person}

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^C AP_c$$

C: number of categories

COCO detection benchmark



Common Objects in Context

80 detection categories

Multiple objects, varying size and occlusion

Boxes and instance masks support separate detection tasks.

Localization thresholds in COCO AP

Target

Prediction



$\text{IoU} = 0.60$

AP50: IoU threshold 0.50

Enough overlap to match

AP75: IoU threshold 0.75

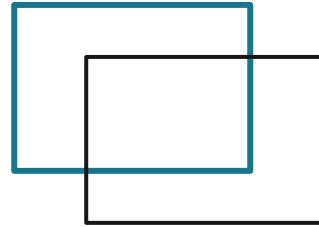
Insufficient overlap

COCO AP averages over IoU thresholds:

0.50, 0.55, ..., 0.95

Three uses of IoU

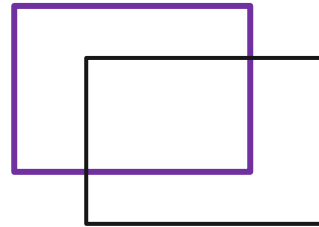
Training



Candidate $\rightarrow$ target

Assign supervision

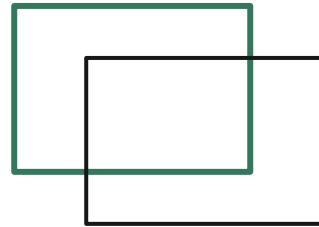
NMS



Prediction $\rightarrow$ prediction

Remove duplicates

Evaluation



Prediction $\rightarrow$ target

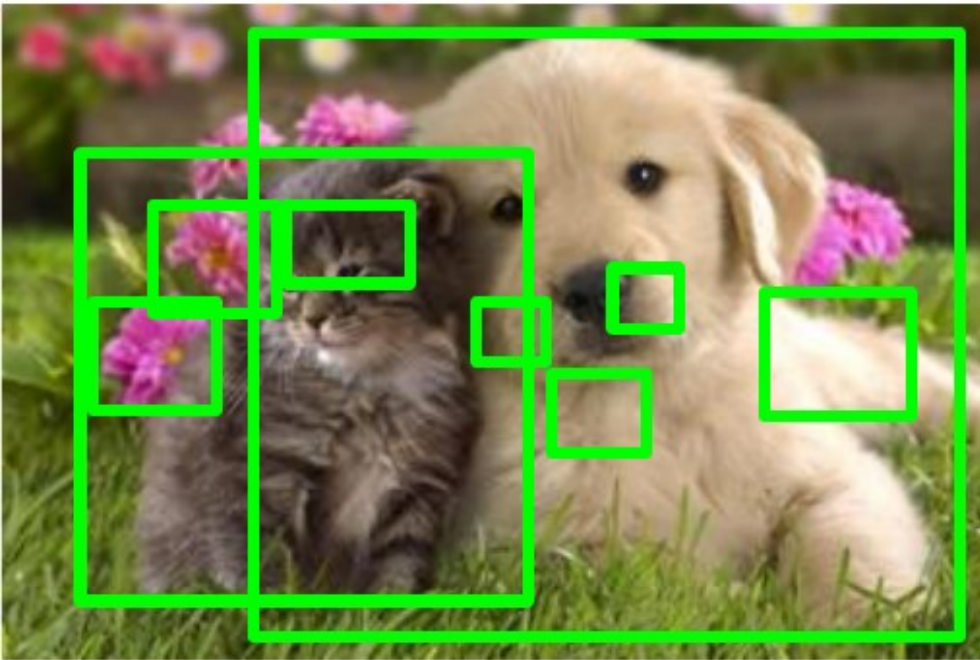
Count correct objects

Lecture outline

- 01 The object detection problem
- 02 Region-based detection
- 03 Detection evaluation
- | 04 **Dense and multiscale detection**
- 05 Instance segmentation

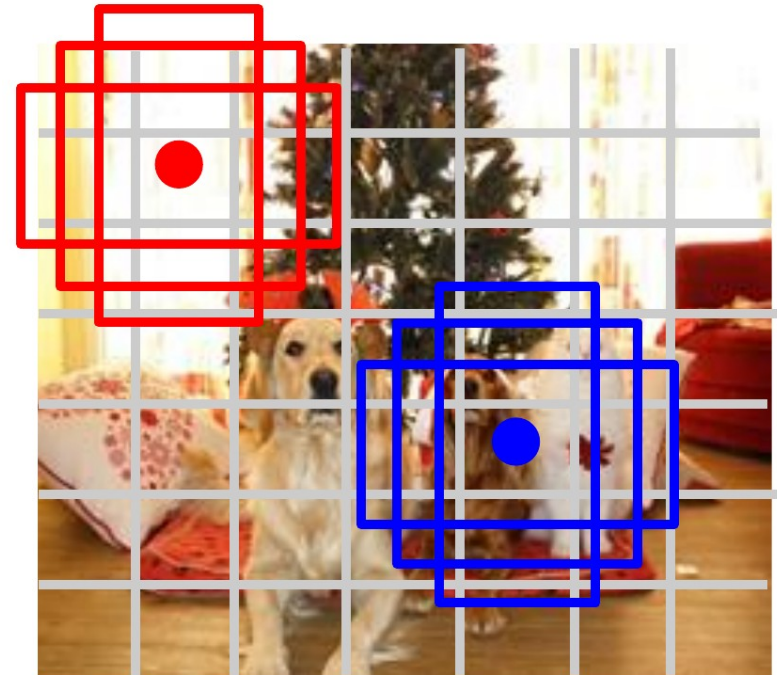
One-stage object detection

Two-stage



Select proposals, then classify and refine each region.

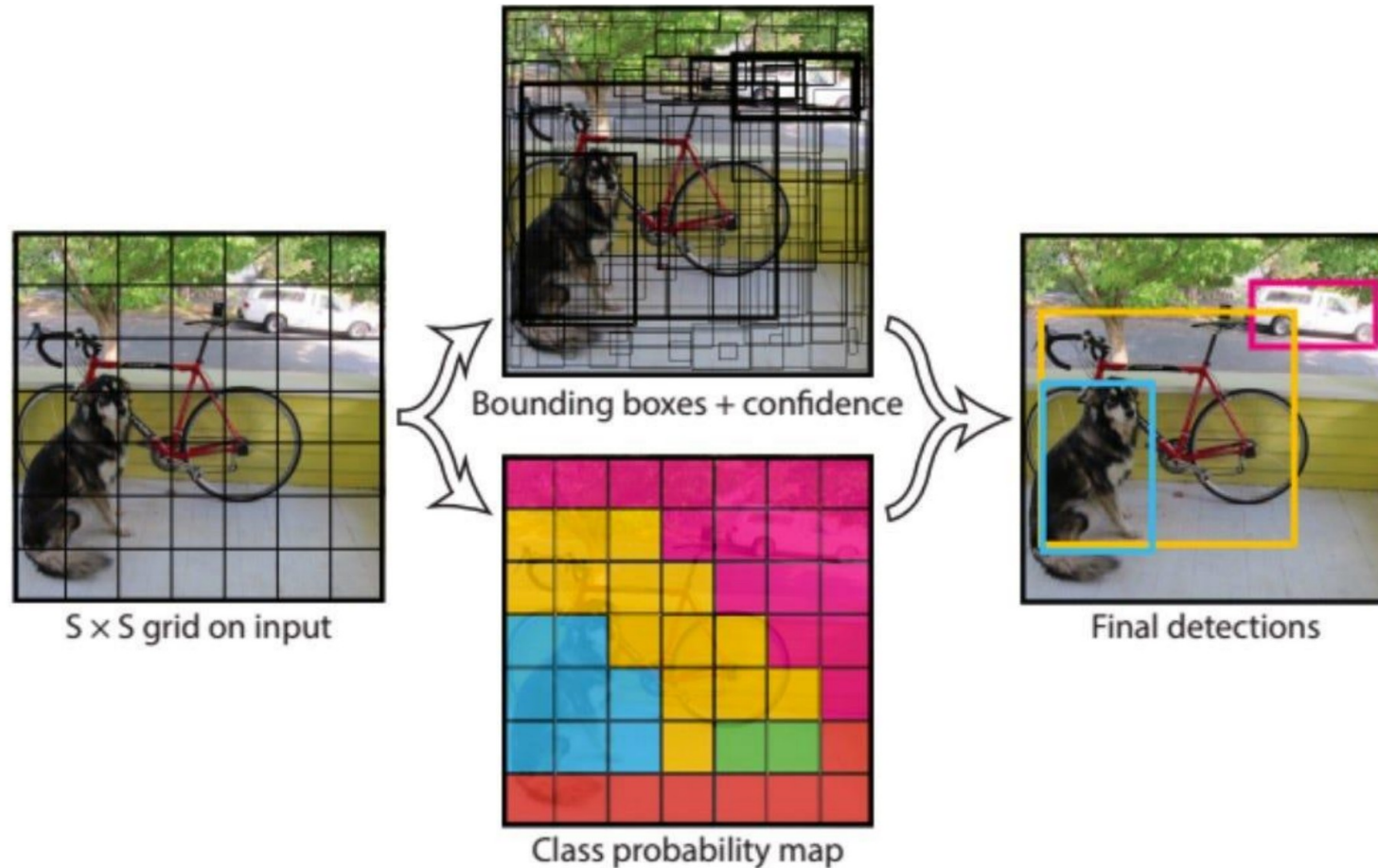
One-stage



Predict class scores and box corrections directly at feature locations.

YOLOv1 (2016)

One image produces a fixed grid of predictions.



Predictions from a grid cell



One cell predicts two boxes

Box 1 $(x, y, w, h, \text{confidence})$

Box 2 $(x, y, w, h, \text{confidence})$

One class-probability vector

Shared by both boxes in the cell.

The cell containing the object's center is responsible for it.

YOLOv1 confidence

Box confidence

Object presence $\times$ box quality

Training target:

0 for background

IoU for a matched object

Predict confidence at test time.

Class probabilities

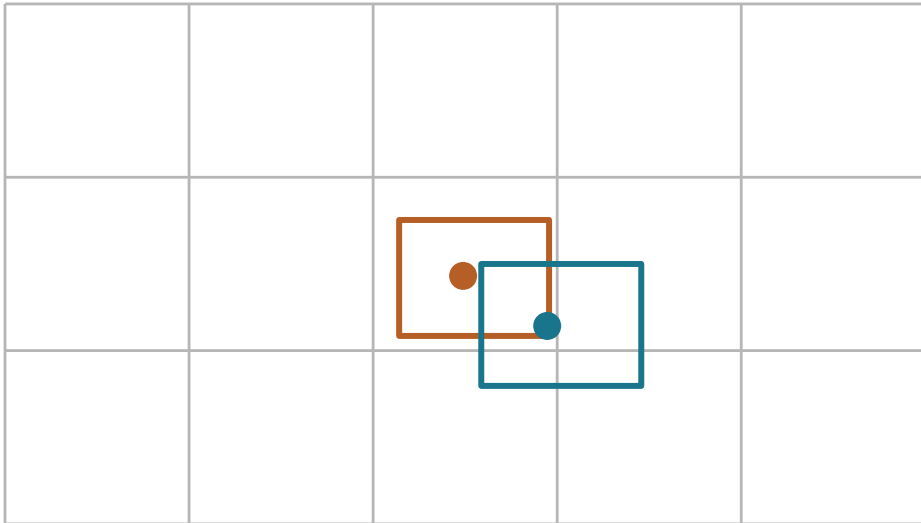
Which category,
given an object is present?

One vector shared by
all boxes in the cell.

$$\text{Class score} = \text{class probability} \times \text{box confidence}$$

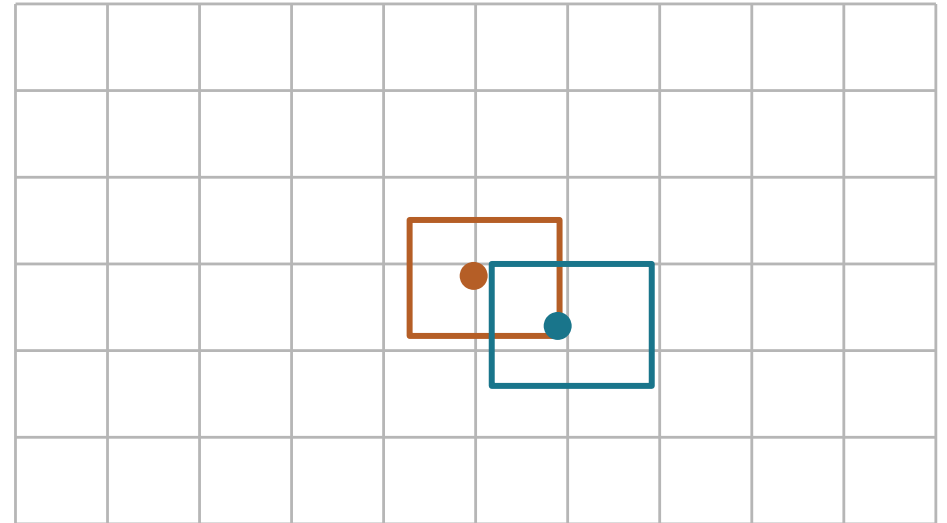
Crowded cells and small objects

Coarse prediction grid



Two objects compete for one cell's class prediction

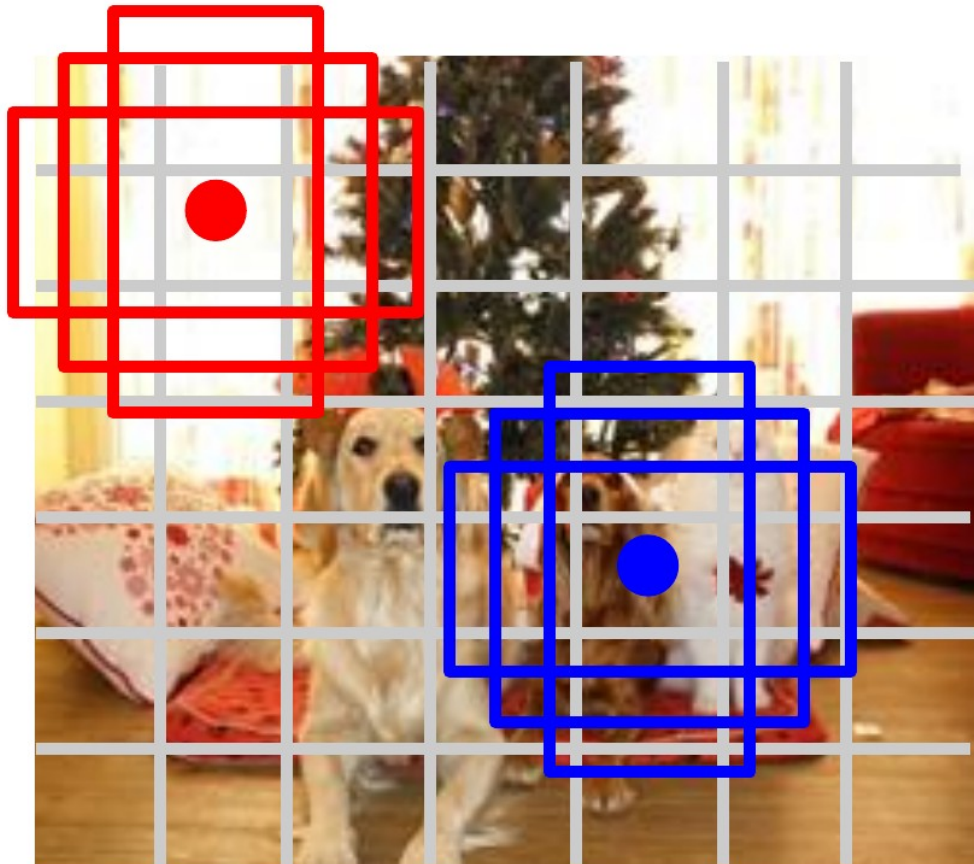
Finer prediction grid



Centers fall into different cells

Dense prediction heads

Anchor-based example



At the blue location

Category scores

Classify each reference box.

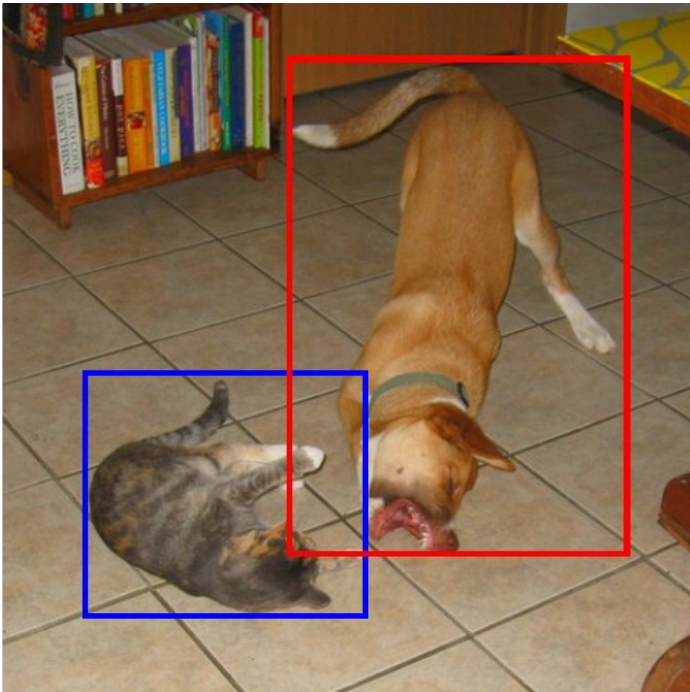
Box corrections

Move and resize each reference box.

The same convolutional predictors run at every feature location.

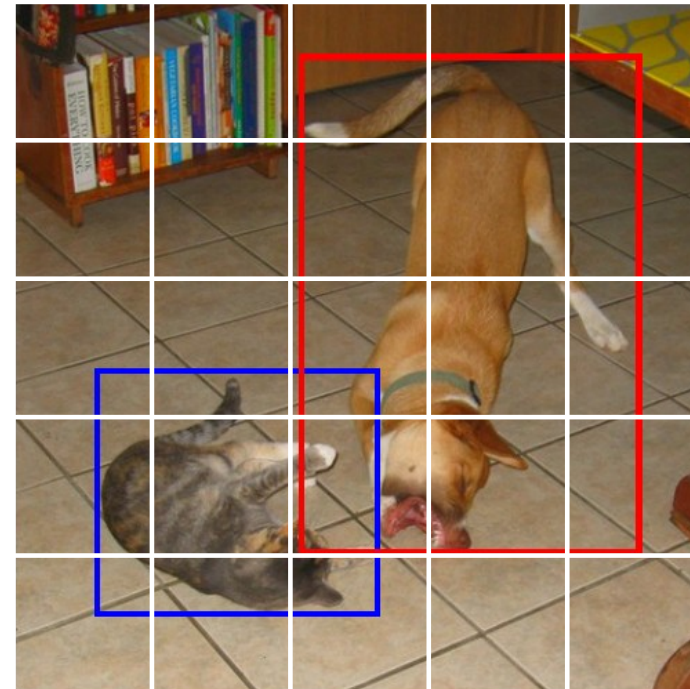
Objects at different scales

Two object scales



Small cat, large dog

Coarse feature grid



Few positions cover the cat

Small and occluded objects



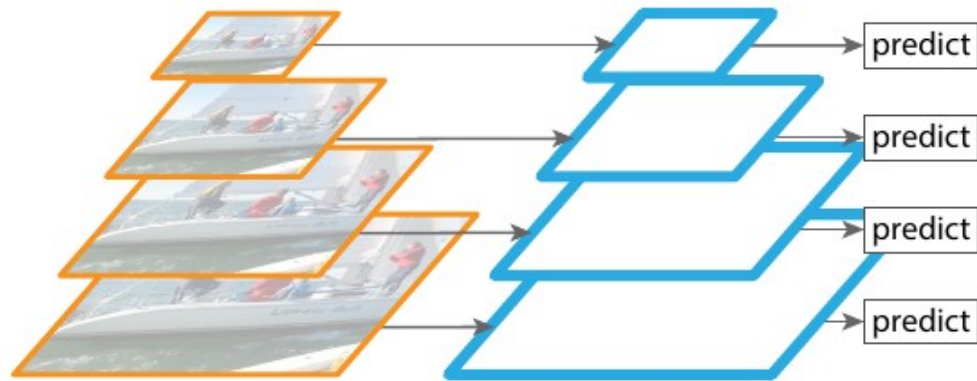
Enlarged region



Small objects and occlusion leave less visual evidence.

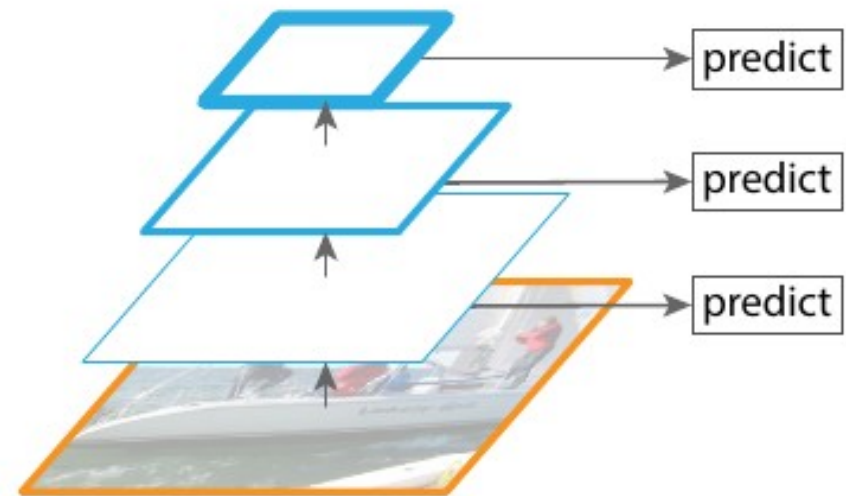
Image and feature pyramids

Image pyramid



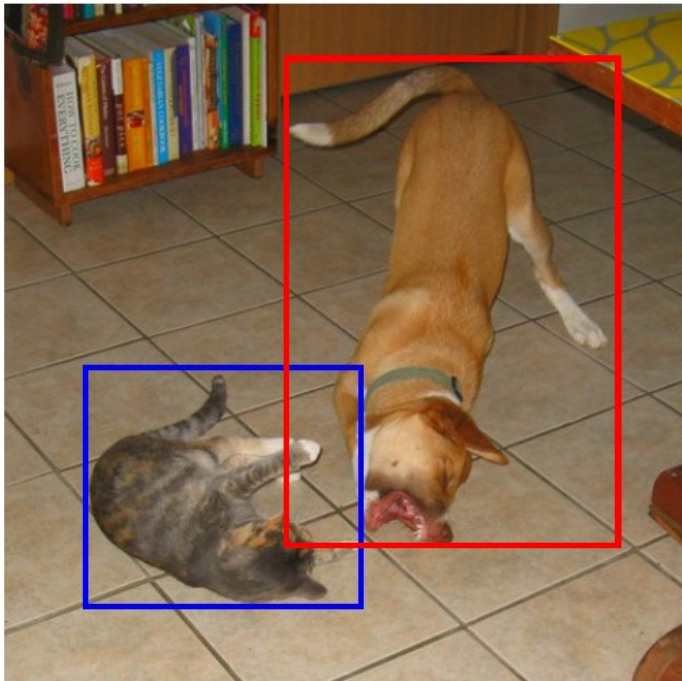
Recompute features for several resized images

CNN feature hierarchy

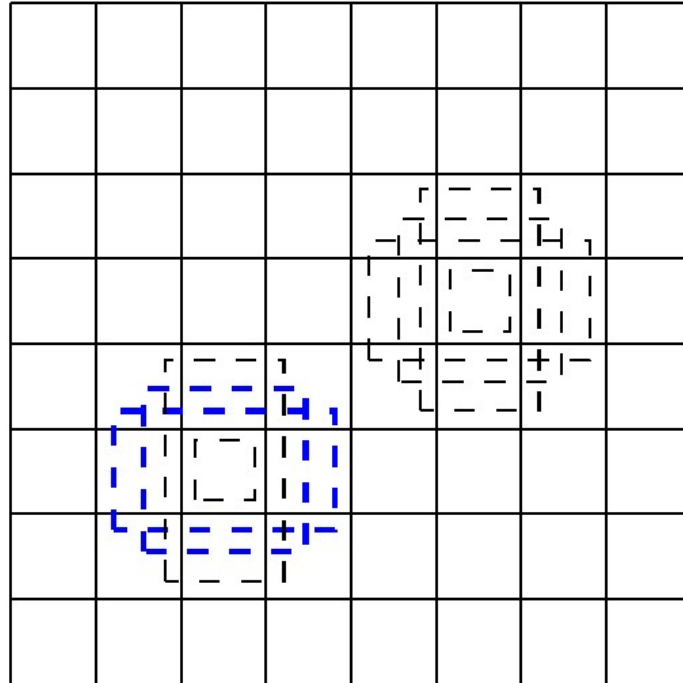


Reuse levels from one backbone pass

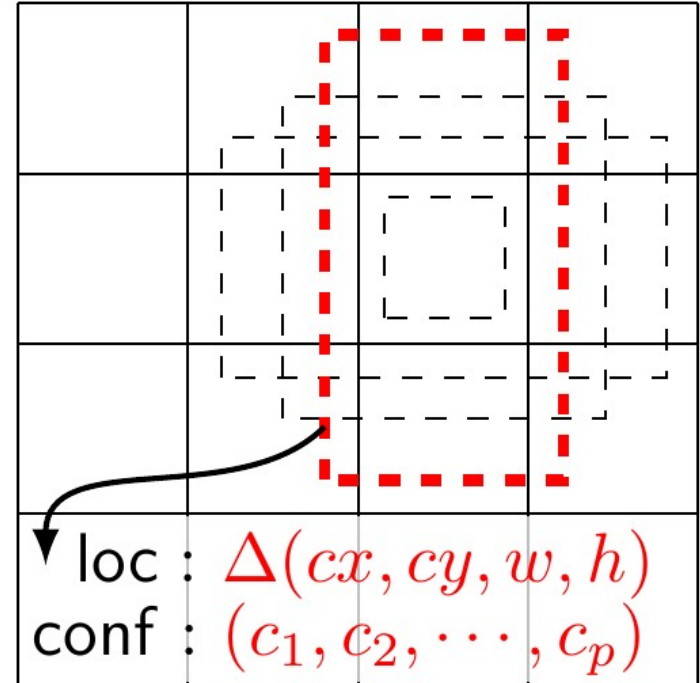
SSD: predictions at multiple scales



(a) Image with GT boxes



(b) 8×8 feature map



loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \dots, c_p)$

(c) 4×4 feature map

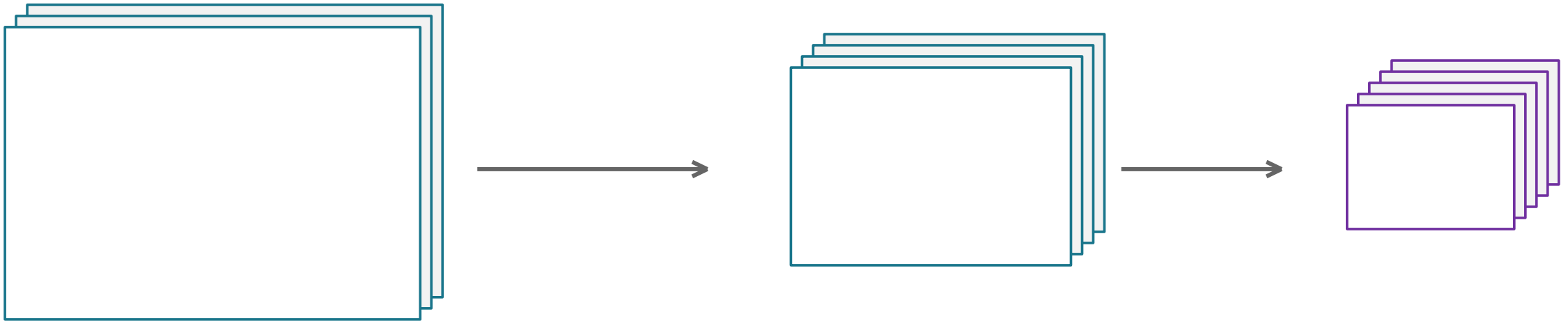
Dashed rectangles: anchor boxes (SSD calls them default boxes)

Finer maps for smaller objects, coarser maps for larger objects

Resolution and semantic information

Fine resolution

Coarse resolution



Fine detail for localization

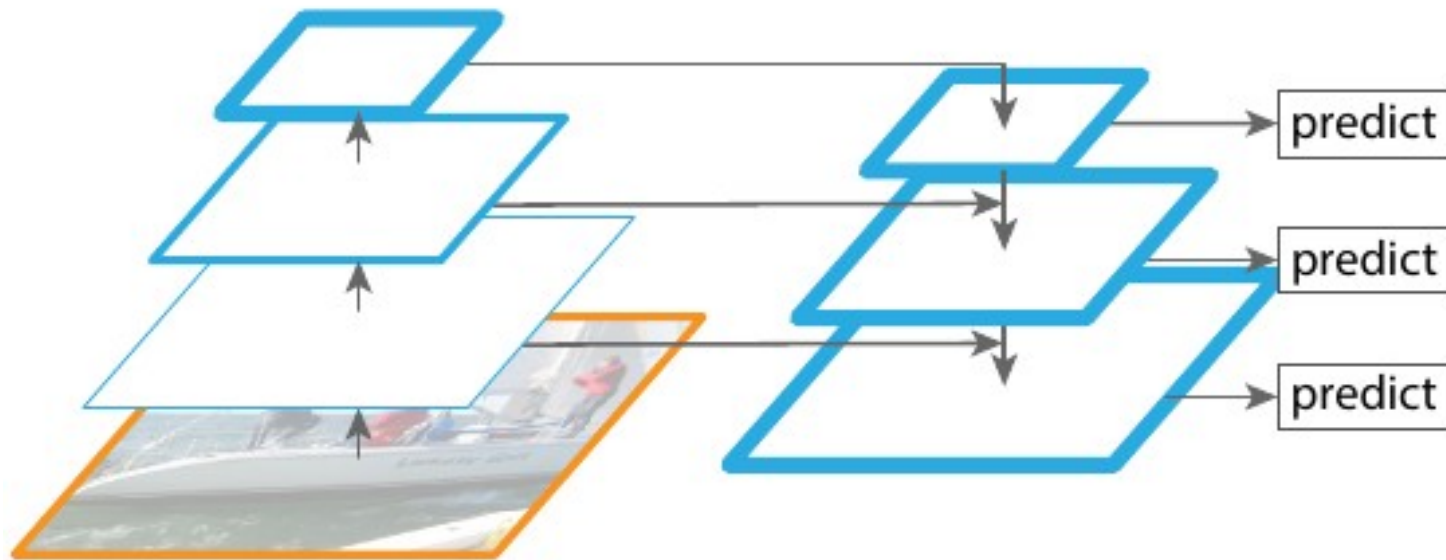
Deeper features for recognition

Bring deeper object features to the finer maps

Feature Pyramid Network (FPN)

Bottom-up

Top-down

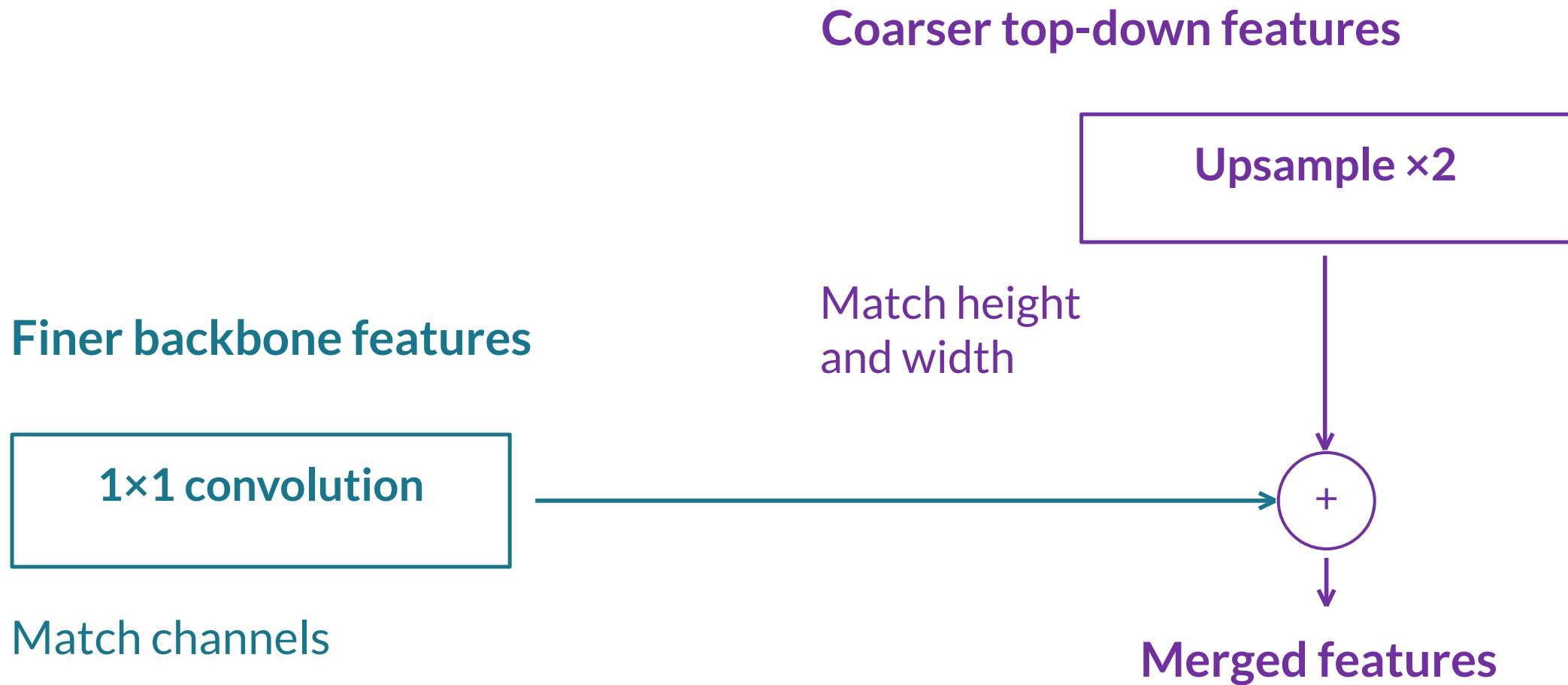


Upsample deeper features

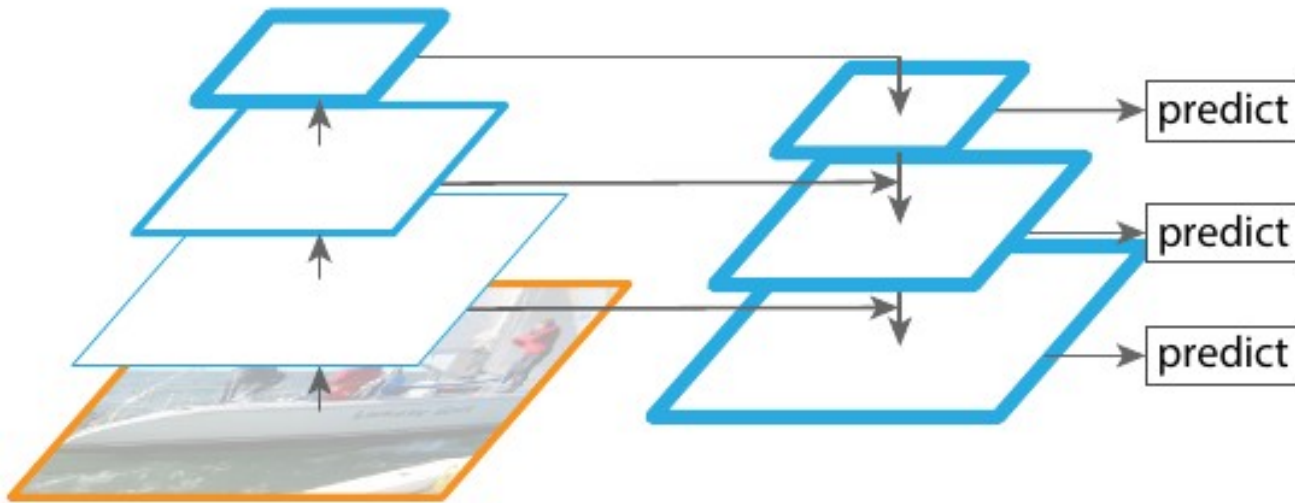
Add lateral spatial detail

Predict at several levels

Lateral fusion



FPN in a detector



Region-based detector

Extract each RoI from a suitable pyramid level.

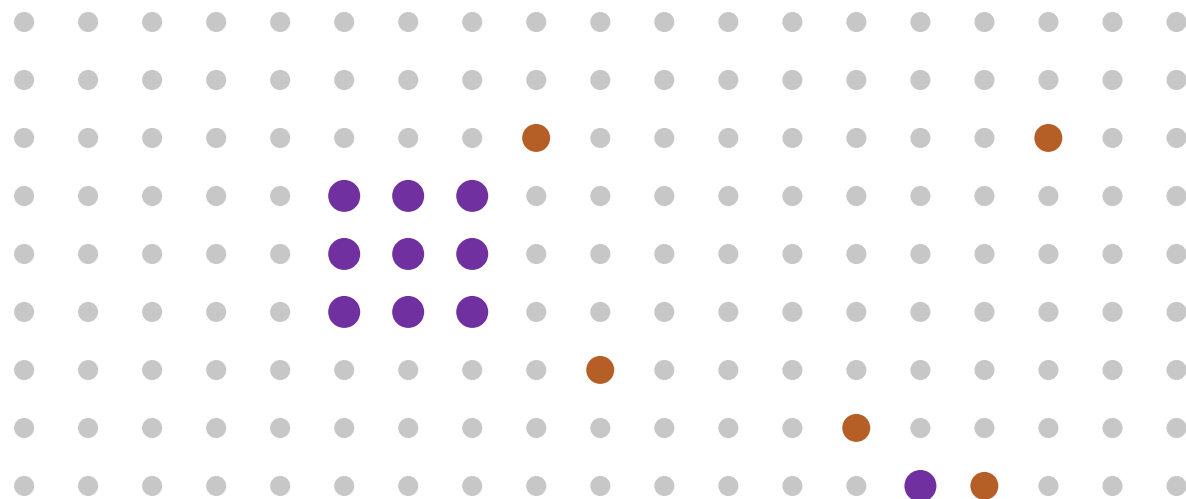
Dense detector

Predict categories and boxes at every location of each level.

Foreground-background imbalance

Sigmoid scores: one binary decision per category

Dog vs not dog • each dot is a candidate



- Dog: target 1
- Hard negatives: 0
- Easy negatives: 0

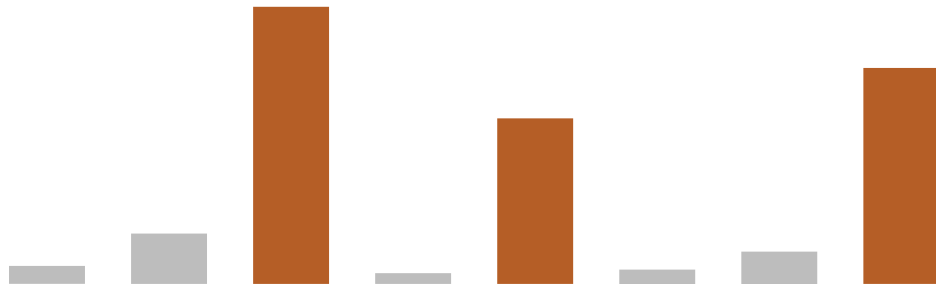
Background has target 0 for every category.

Many easy negatives can dominate the total loss.

Selecting useful negatives

Hard-negative mining

Per-candidate classification loss



Keep the high-loss negatives

Confidence-dependent weighting



Keep all negatives.
Downweight the easy ones.

Focal loss

p_t Probability assigned to the correct binary target

$$p_t = p \text{ for target } 1, \quad p_t = 1 - p \text{ for target } 0$$

p : predicted probability of this class

Cross-entropy

$$L_{\text{CE}} = -\log p_t$$

Focal loss

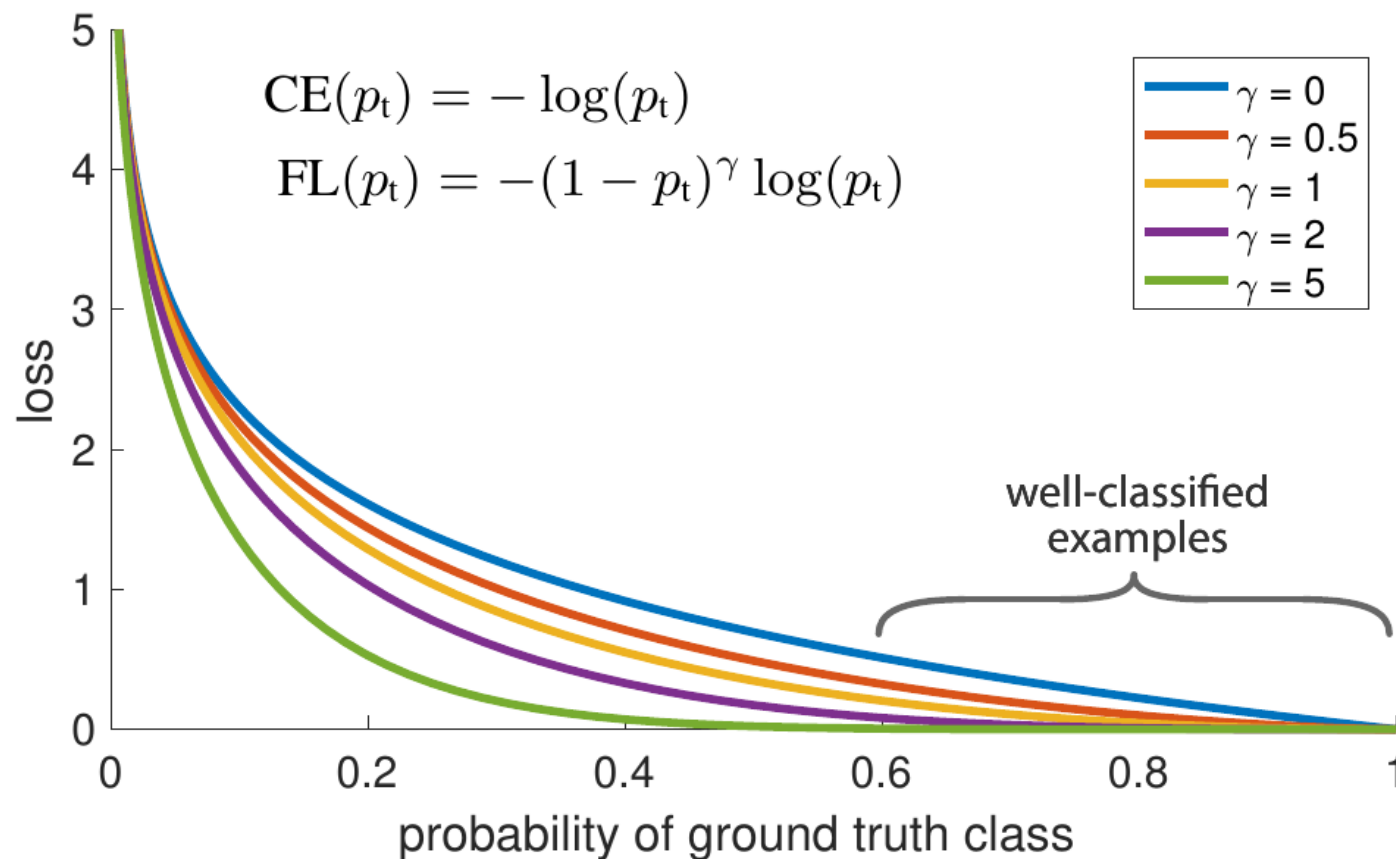
$$L_{\text{focal}} = -(1 - p_t)^\gamma \log p_t$$

γ controls how strongly easy examples are downweighted.

Easy example: multiplier near zero

Hard example: multiplier closer to one

Effect of focal loss



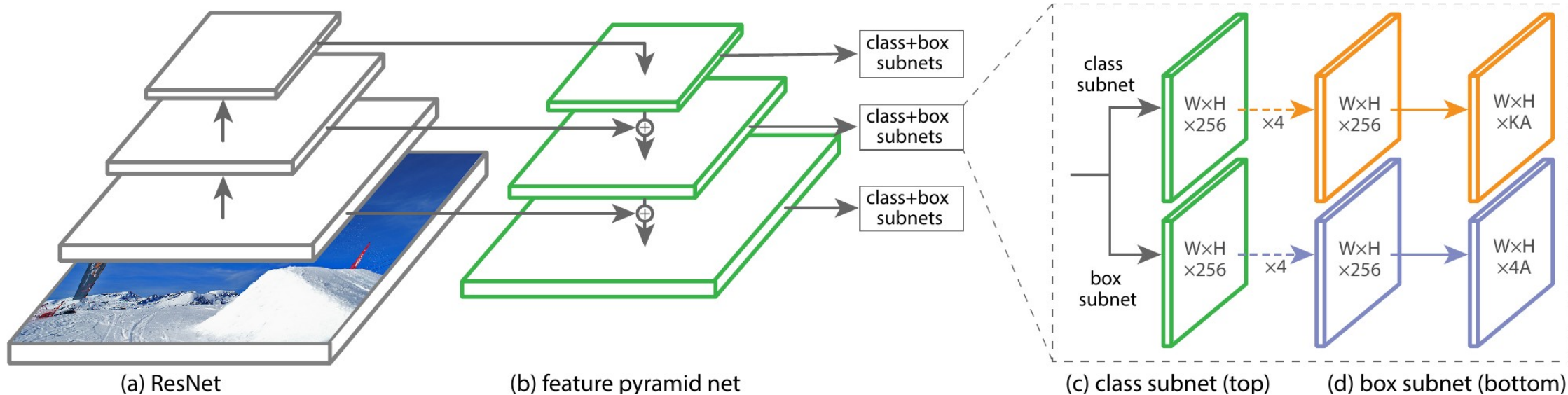
Poor prediction

Retain a substantial loss

Confident and correct

Reduce the loss sharply

RetinaNet



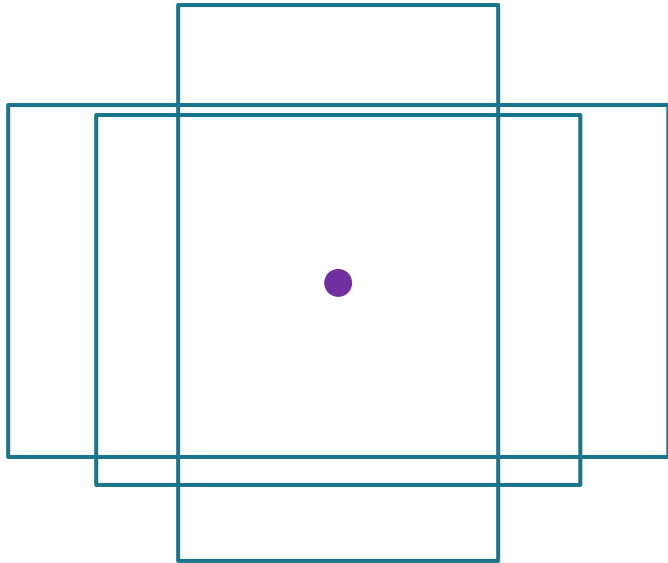
FPN + dense heads, trained with focal loss

A: anchors per location K: object categories

Category sigmoid scores directly rank boxes. No separate objectness head.

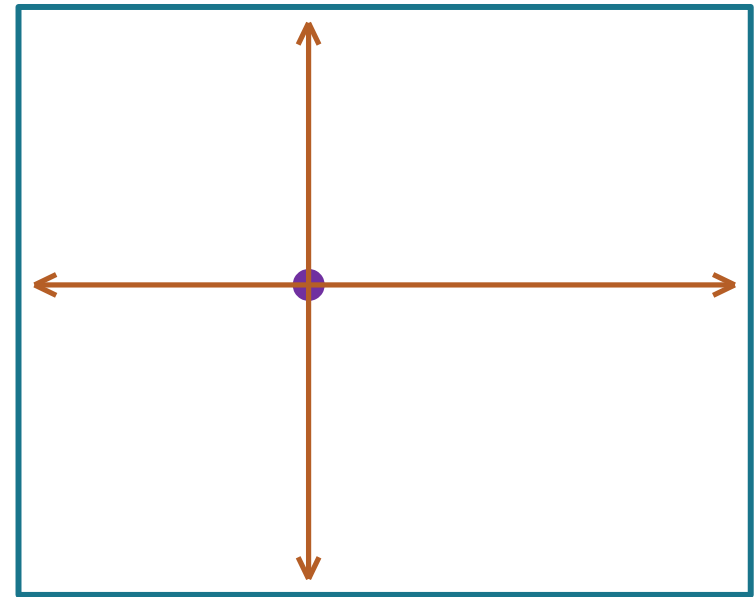
From anchor boxes to locations

Anchor boxes



Choose template sizes and shapes,
then predict corrections

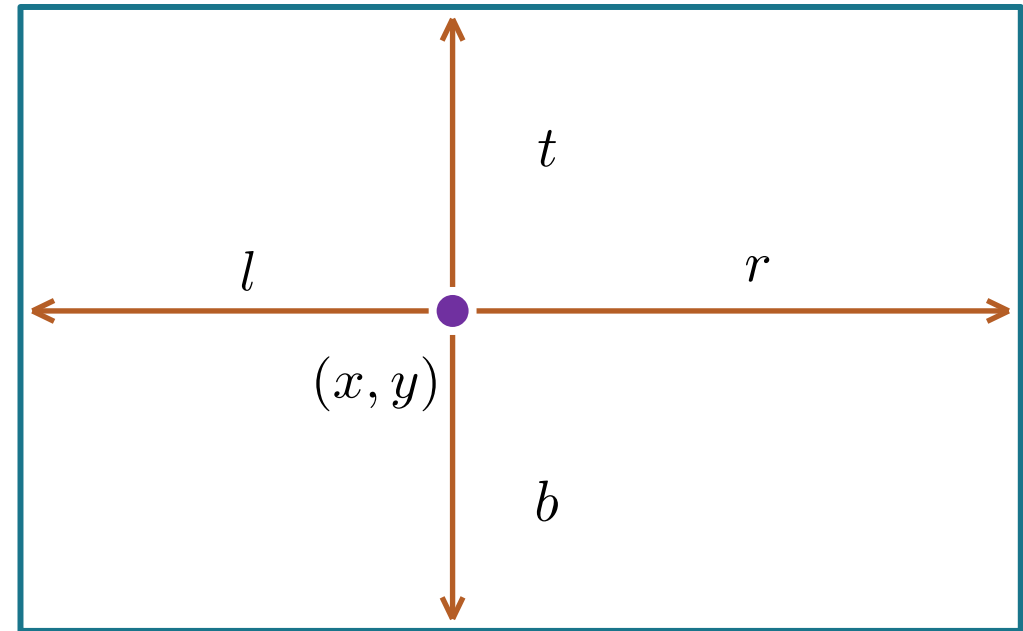
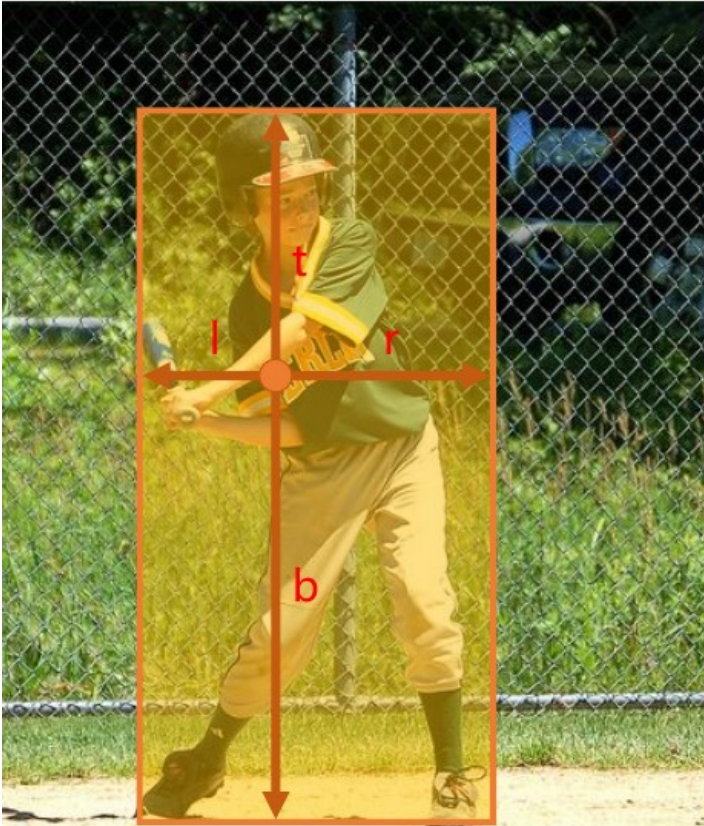
Reference location



Predict side distances directly.
No box templates to choose.

FCOS box coordinates

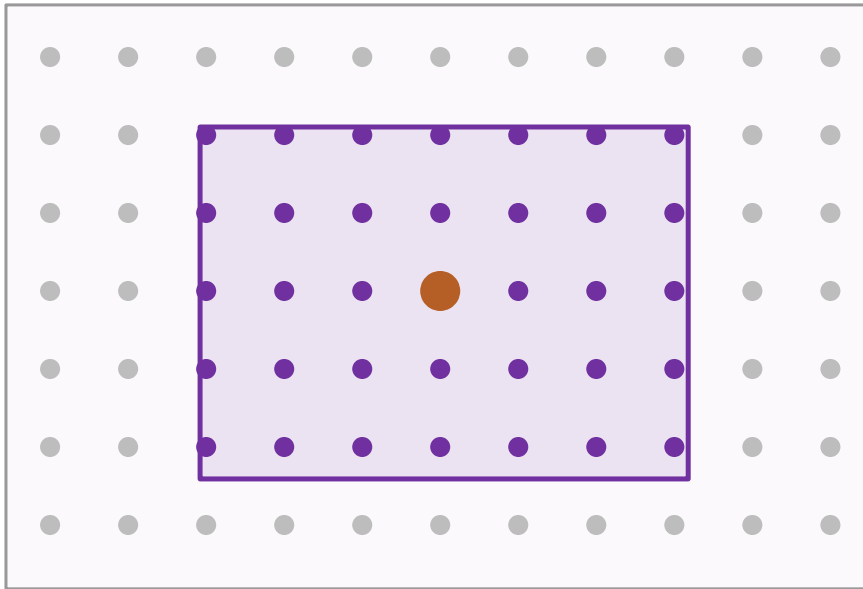
Fully convolutional one-stage object detection



$$(x - l, y - t, x + r, y + b)$$

FCOS training targets

Box annotation



Targets at eligible locations

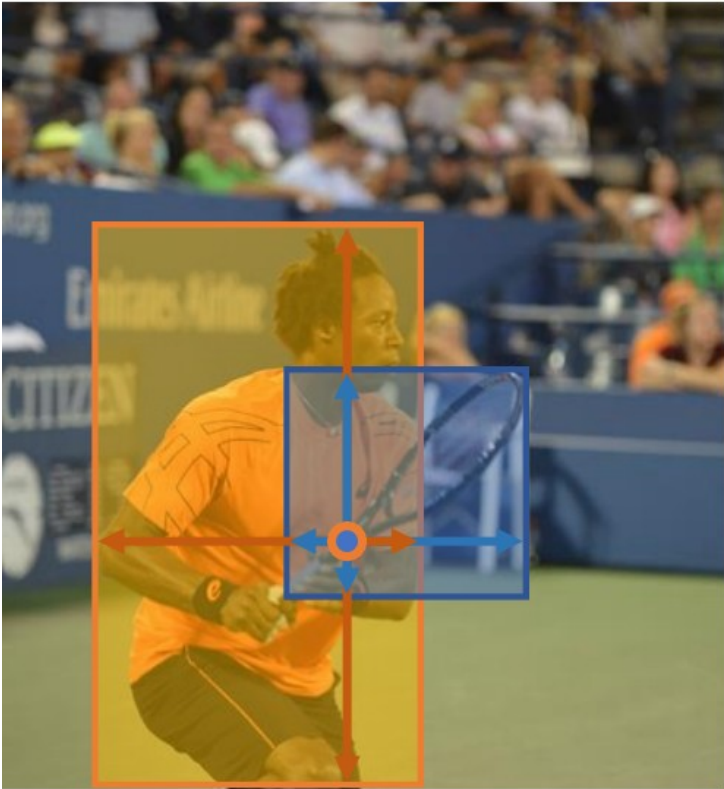
Class: the assigned object

Box: distances to its sides

Outside eligible boxes: background

Box interiors are not segmentation masks

Multiple objects and pyramid levels



One point can lie in two boxes

Use a suitable pyramid level

Fine maps: smaller box extents

Coarse maps: larger box extents

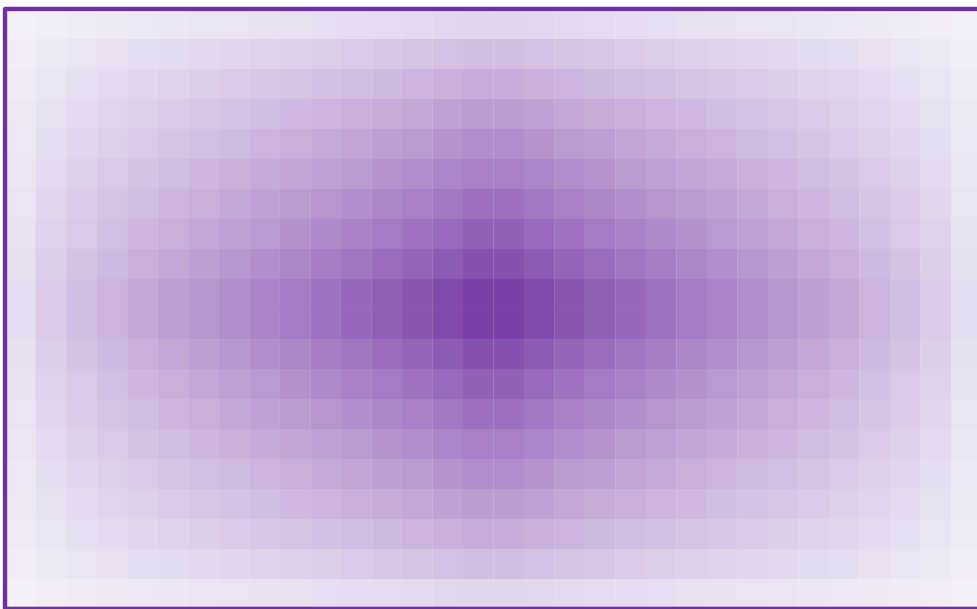
If eligible boxes still overlap

Assign the smaller eligible box

Center-ness

Off-center locations can predict poorer boxes

Training target



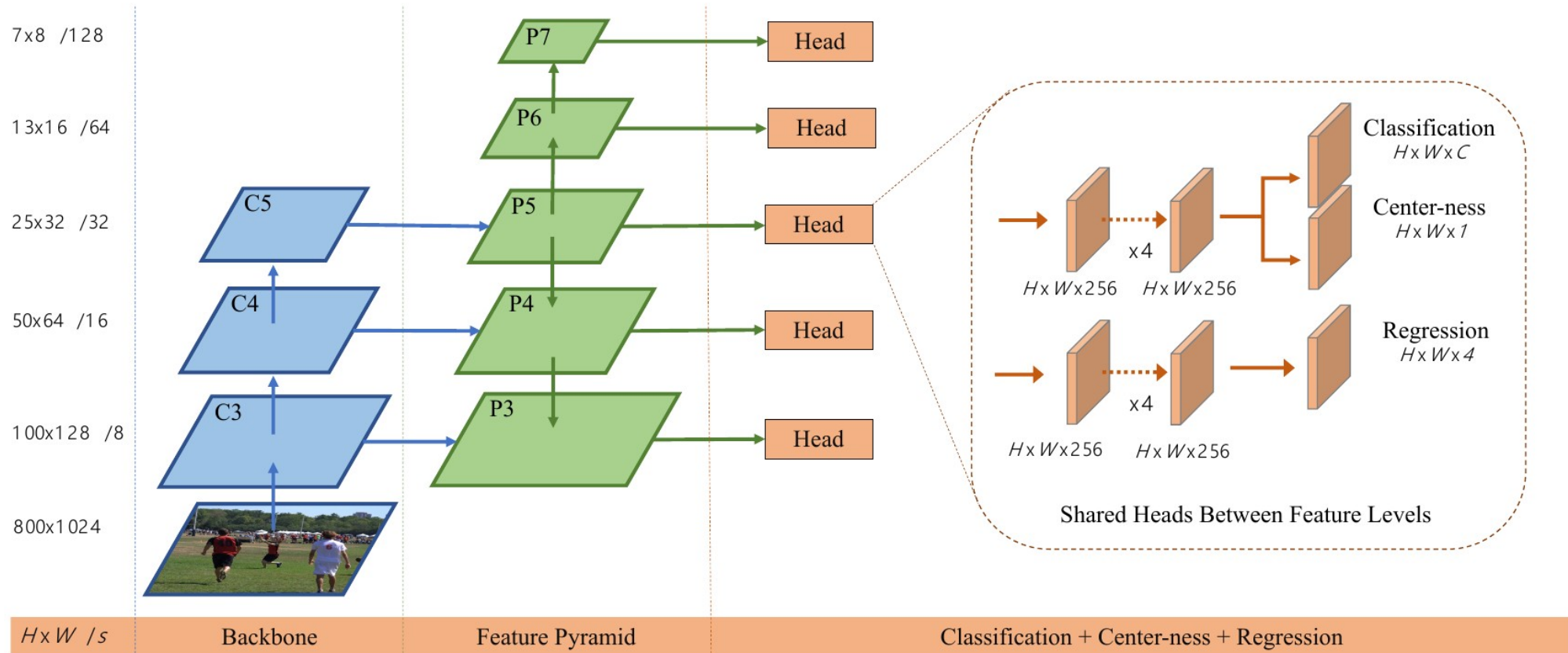
Lower near edges **Higher at center**

$$\sqrt{\frac{\min(l, r)}{\max(l, r)} \frac{\min(t, b)}{\max(t, b)}}$$

Center-ness measures location quality, not objectness.

At test time: class score × predicted center-ness, then NMS

FCOS: from features to detections



Decode distances to boxes, rank by class score $\times$ center-ness, then NMS.

Lecture outline

- 01 The object detection problem
- 02 Region-based detection
- 03 Detection evaluation
- 04 Dense and multiscale detection
- | 05 **Instance segmentation**

Instance segmentation

Input image



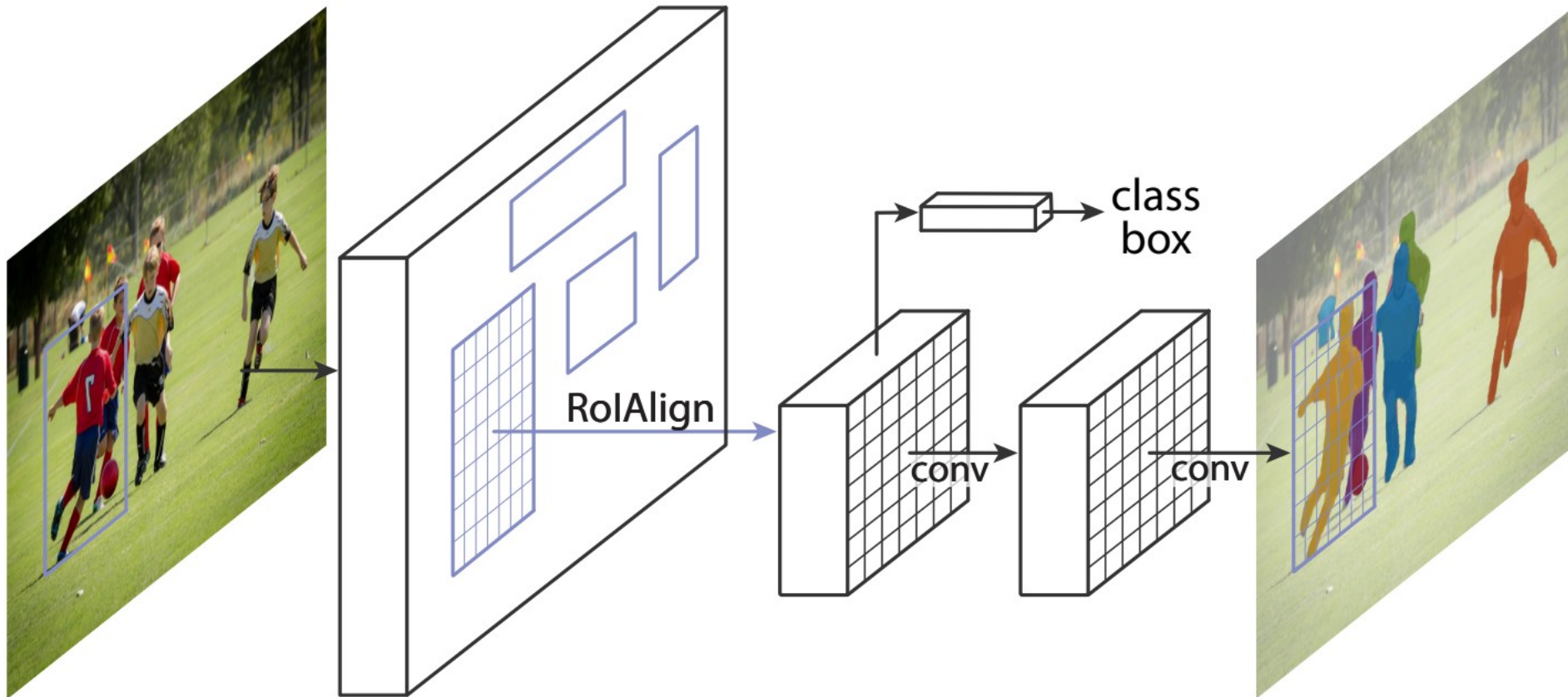
Separate masks for separate people



Given a candidate object region, predict which pixels belong to that object.

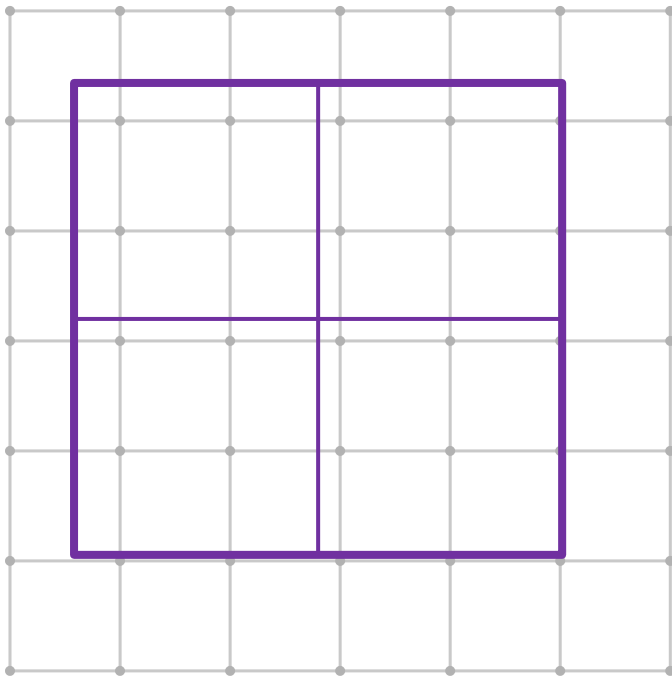
Mask R-CNN

Faster R-CNN + a spatial mask branch



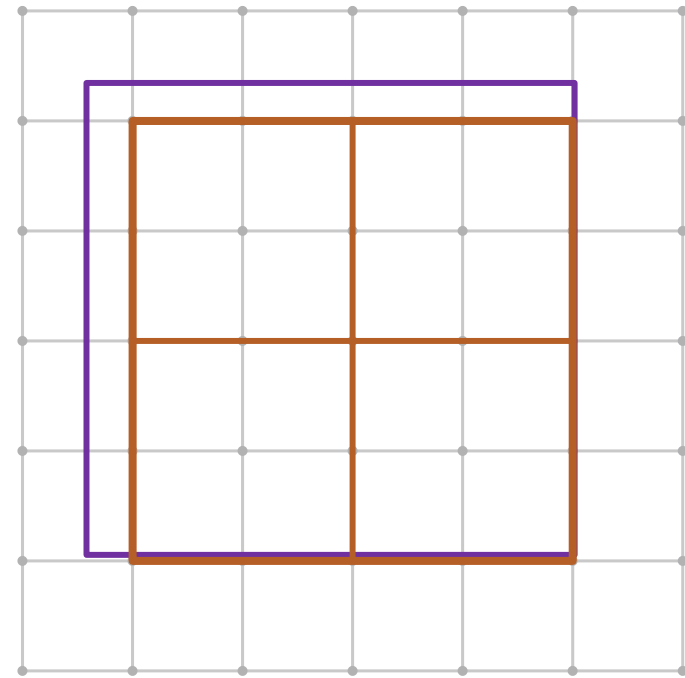
Region alignment

Continuous region



Feature locations lie on the gray grid.

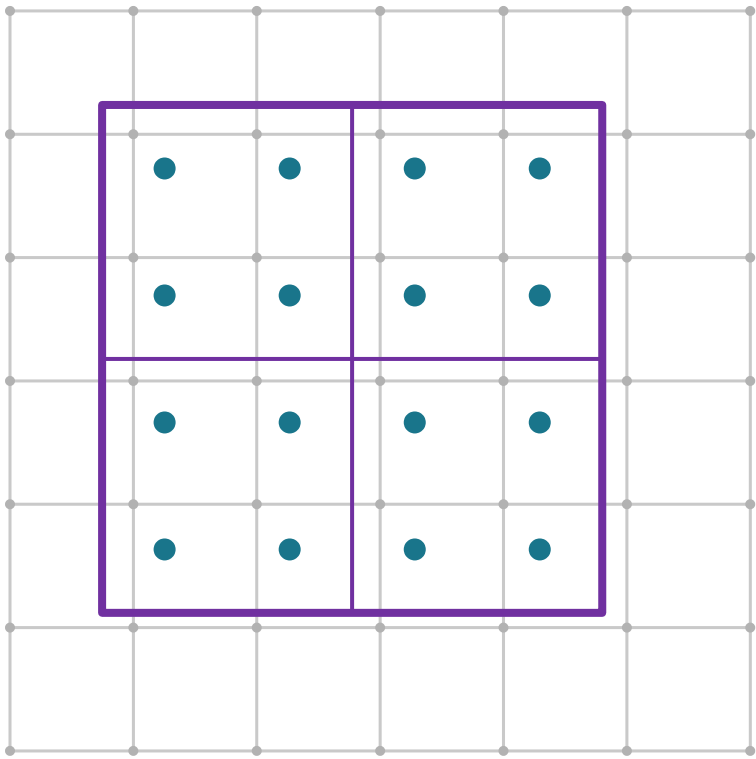
RoI pooling: rounded bins



Rounding can misalign
the mask with the object.

RoAlign

Keep the region coordinates



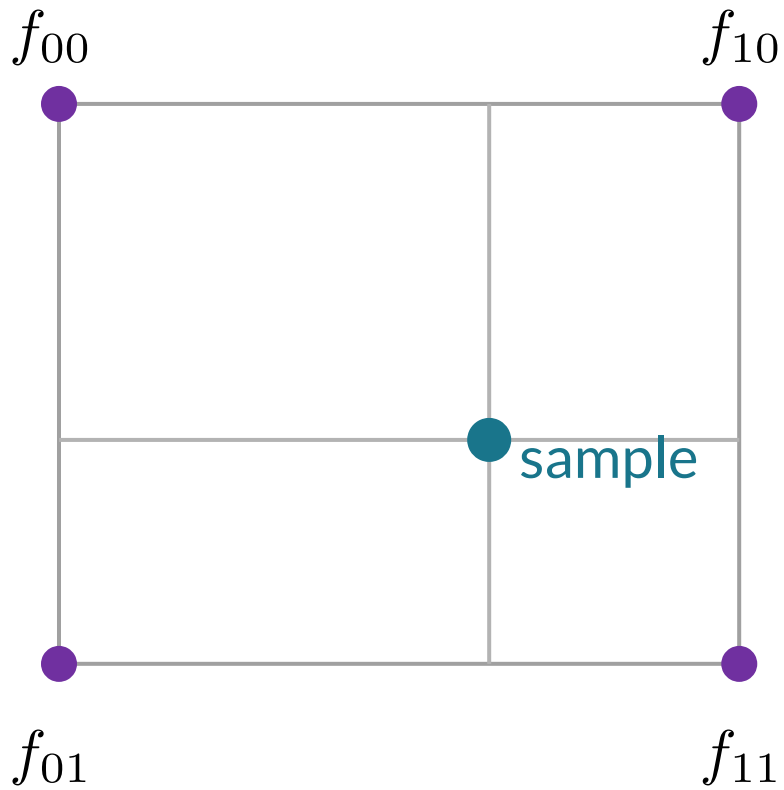
Sample at regular locations inside each bin.

Interpolate the feature values.

Average the samples in each bin.

Repeat for every feature channel.

Sampling between feature locations

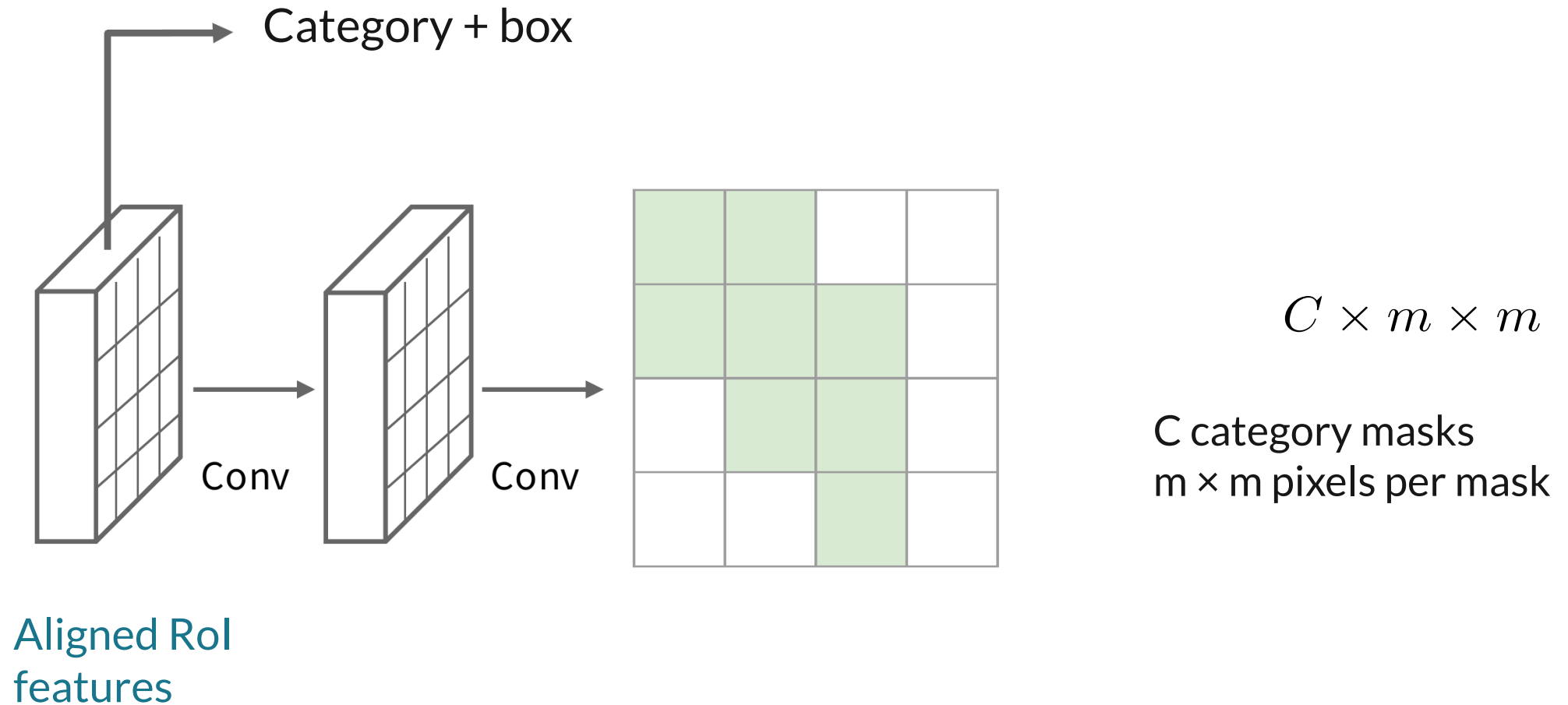


**Weighted average
of four neighbors**

More weight on closer values.

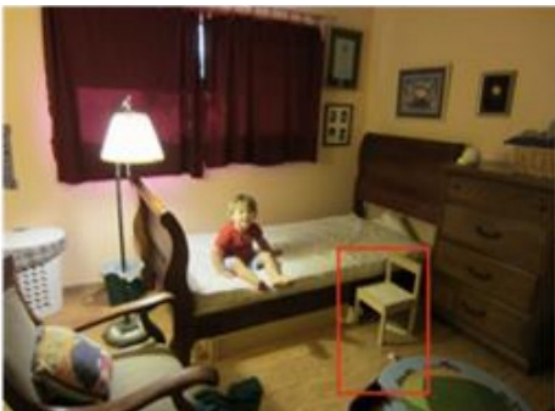
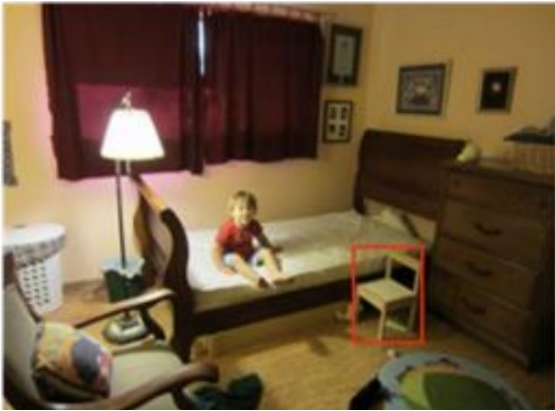
The same interpolation weights are
used for every channel.

The mask head



Mask training targets

Positive region



Binary target

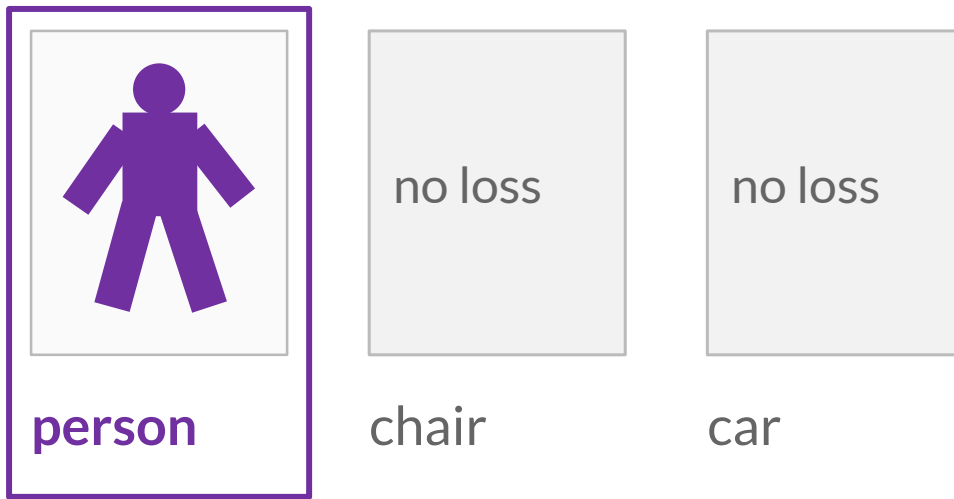


Black: the matched chair
White: everything else

Crop the matched chair's mask
to each proposal.

Binary mask prediction

Training region: person



Supervise only the true class's mask.

Binary membership at each location

$$-y \log p - (1 - y) \log(1 - p)$$

y : belongs to this instance

p : predicted membership probability

Sigmoid + binary cross entropy

Training Mask R-CNN

Joint objective

$$L = L_{\text{RPN}} + L_{\text{class}} + L_{\text{box}} + L_{\text{mask}}$$

Region proposal network

Objectness + box correction

Second-stage region

Positive: class + box + mask

Negative: background class

Shared features receive gradients from all prediction branches.

Placing masks in the image

Mask for the predicted class

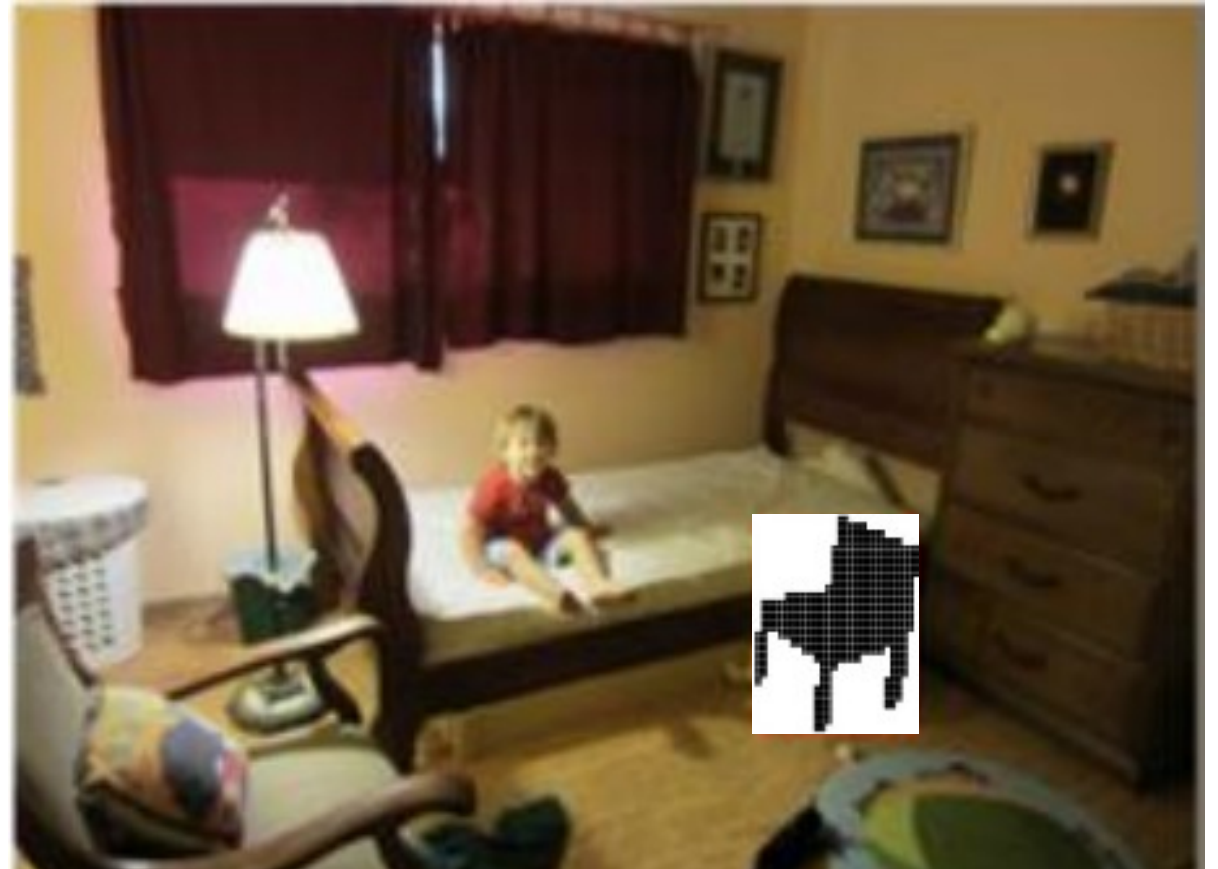


Resize into the detected box.
Threshold the mask probabilities.

Example: chair annotation

Source: [CS231n 2026](#)

Mask in image coordinates



Instance masks in complex scenes



Evaluating instance masks

Correct outline



Same box, different mask



Ground truth

Prediction

$$\text{IoU}_{\text{mask}} = \frac{|M_{\text{pred}} \cap M_{\text{true}}|}{|M_{\text{pred}} \cup M_{\text{true}}|}$$

Use mask overlap in the same ranked matching and AP procedure.

Detection targets in Torchvision

```
images = [image_a, image_b]  # RGB tensors in [0, 1]
targets = [
    {"boxes": boxes_a,        # [N, 4]: x1, y1, x2, y2
     "labels": labels_a,      # [N]: foreground class IDs
     "masks": masks_a},      # [N, H, W]: binary masks
    target_b,                 # annotations for image_b
]
```

One row per object, in the same order in all three fields.

Torchvision: training and inference

Training: targets $\rightarrow$ losses

```
model.train()
losses = model(images, targets) # dictionary of loss terms
loss = sum(losses.values())      # use the usual backward + step
```

Inference: images $\rightarrow$ detections

```
model.eval()
with torch.no_grad():
    predictions = model(images) # one dictionary per image
result = predictions[0]         # boxes, labels, scores, masks
```

Next: sequence models

Order changes meaning

The dog chased the cat.

The cat chased the dog.

-
- Language modeling: predict the next word
 - Recurrent neural networks
 - Long-range dependencies and LSTMs