Computer Vision

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

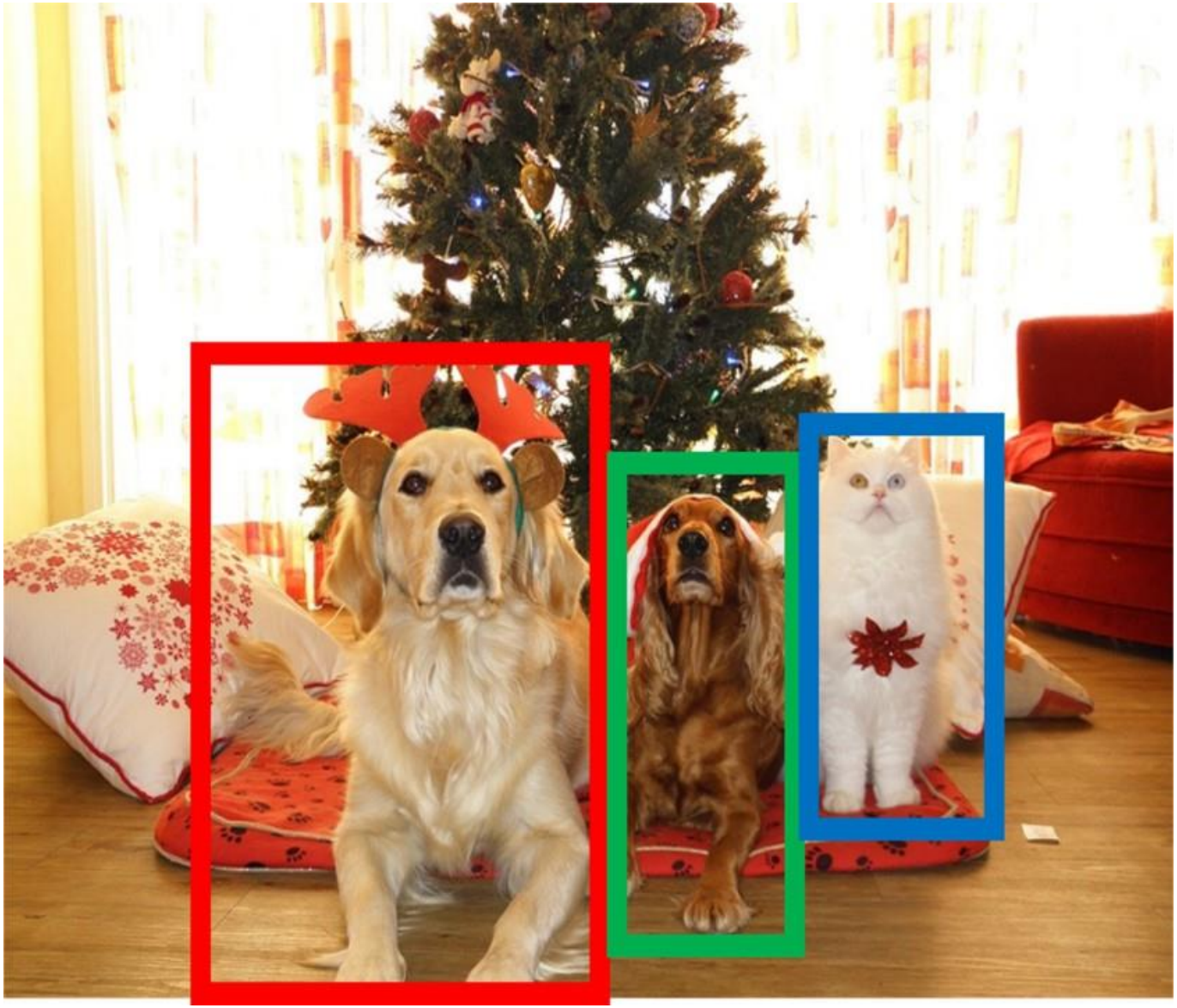
KAUST Academy
King Abdullah University of Science and Technology

January 07, 2024

Object Detection

- **Input:** Single RGB Image
- **Output:** A set of detected objects. For each object predict:
  - Category label (from fixed, known set of categories)
  - Bounding box (four numbers: x, y, width, height)

Object Detection (cont.)

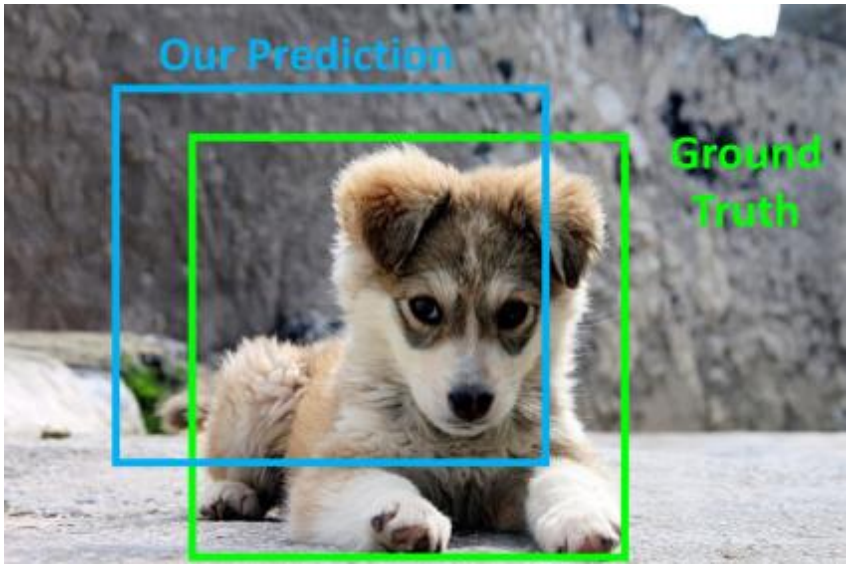


Object Detection: Challenges

- **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 $\sim 800 \times 600$

Comparing Boxes: Intersection over Union (IoU)

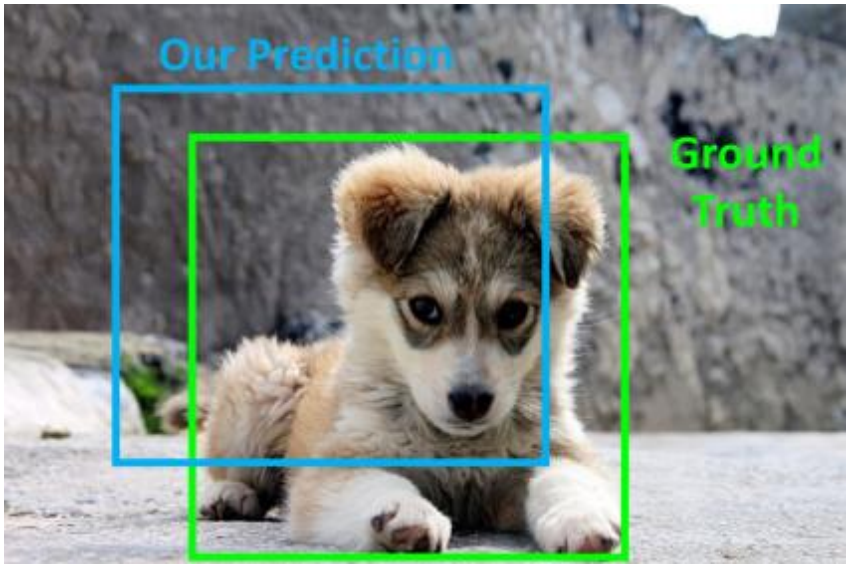
- How can we compare our prediction to the ground-truth box?



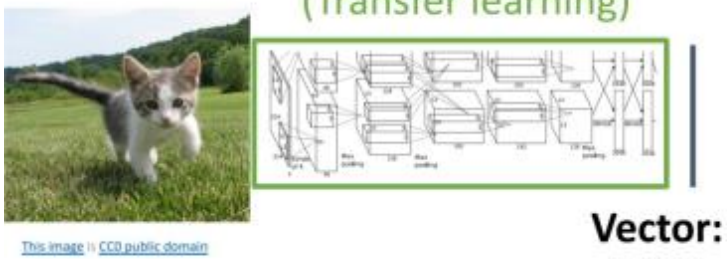
Comparing Boxes: Intersection over Union (IoU)

- How can we compare our prediction to the ground-truth box?
- **Intersection over Union** (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

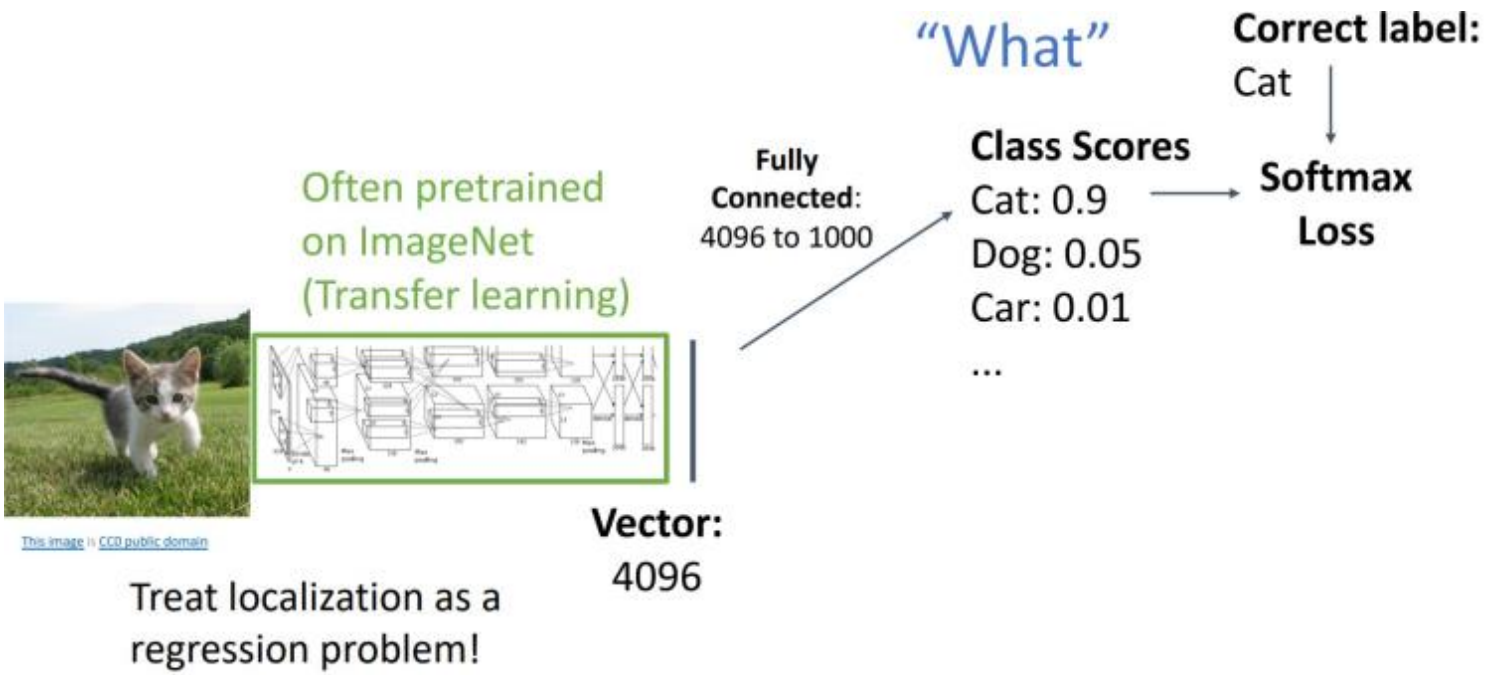
$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$



Detecting a Single Object



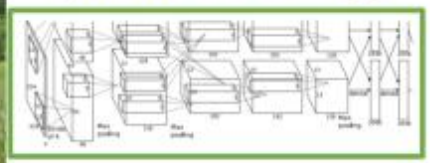
Detecting a Single Object (cont.)



Detecting a Single Object (cont.)

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

"What"

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

"Where"

L2 Loss

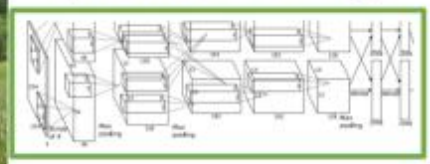
Correct box: (x', y', w', h')

Treat localization as a regression problem!

Detecting a Single Object (cont.)

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

"What"

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

Correct label: Cat

Softmax Loss

Multitask Loss

Weighted Sum → Loss

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

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

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

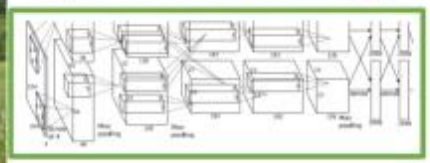
"Where"

Treat localization as a regression problem!

Detecting a Single Object (cont.)

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

"What"

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

Correct label: Cat

Softmax Loss

Multitask Loss

Weighted Sum → Loss

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

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

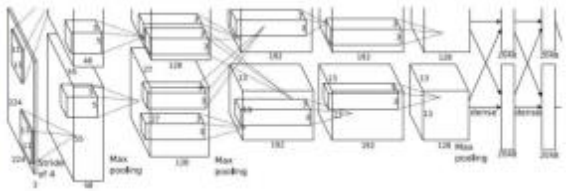
Correct box: (x', y', w', h')

"Where"

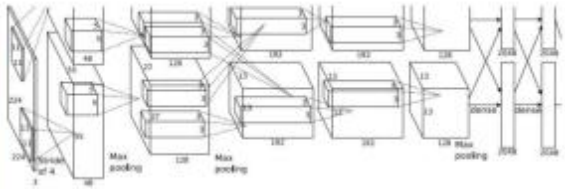
Treat localization as a regression problem!

Problem: Images can have more than one object!

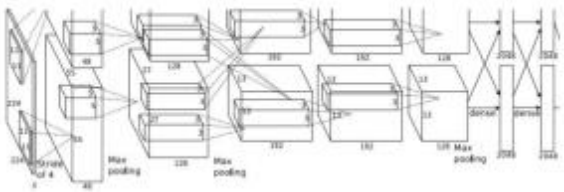
Multiple Objects



CAT: (x, y, w, h)



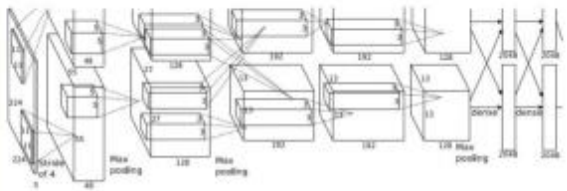
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



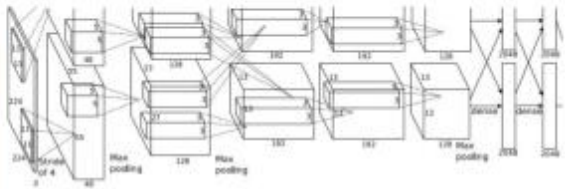
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Multiple Objects (cont.)

Each image needs a different number of outputs!

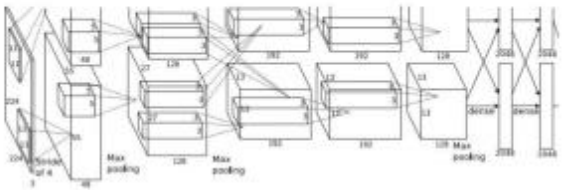


CAT: (x, y, w, h) — 4 numbers



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

12 numbers

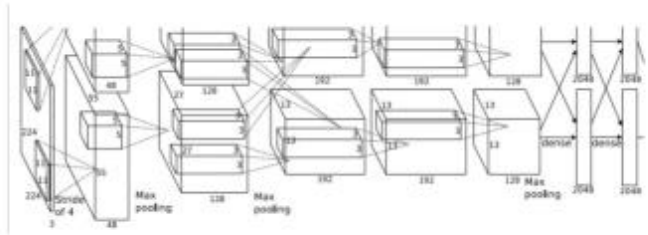


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detecting Multiple Objects: Sliding Window

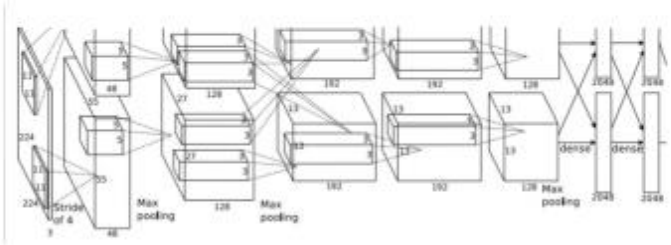
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: Sliding Window (cont.)

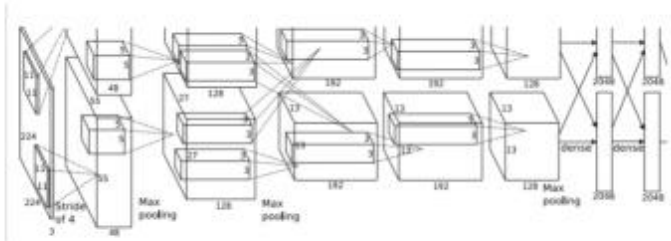
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: Sliding Window (cont.)

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



Dog? NO
Cat? YES
Background? NO

Detecting Multiple Objects: Sliding Window (cont.)

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

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

Detecting Multiple Objects: Sliding Window (cont.)

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

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

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

Detecting Multiple Objects: Sliding Window (cont.)

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

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

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

Total possible boxes:

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

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

Detecting Multiple Objects: Sliding Window (cont.)

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

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

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

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

Total possible boxes:

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

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

Region Proposal

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for "blob-like" image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

R-CNN

Input image

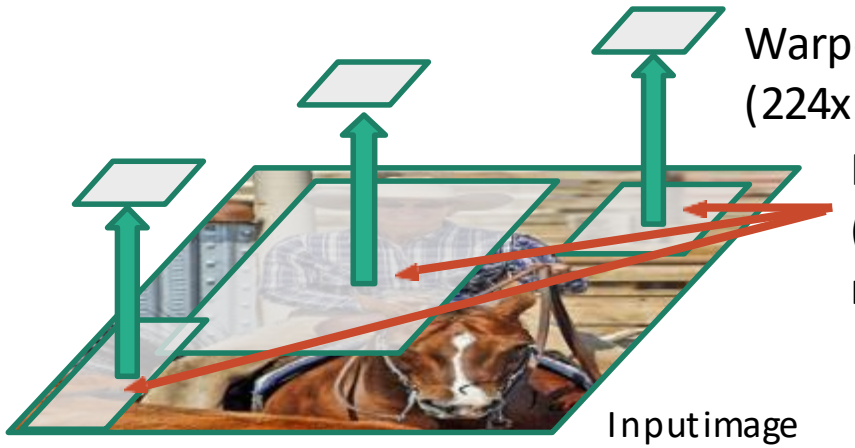
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN

Input image

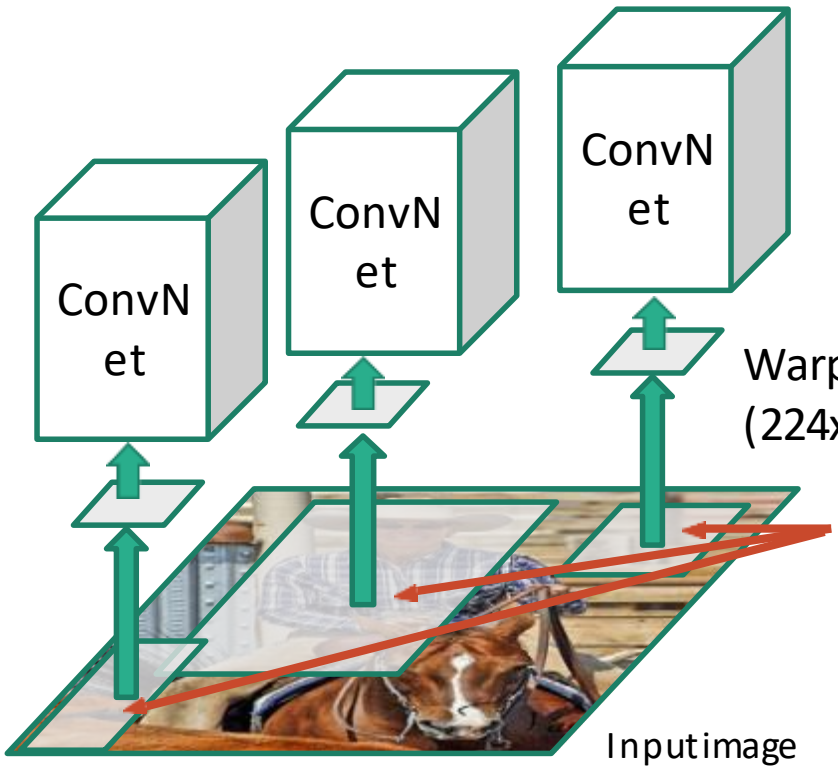
Regions of Interest (RoI) from a proposal method (~2k)

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



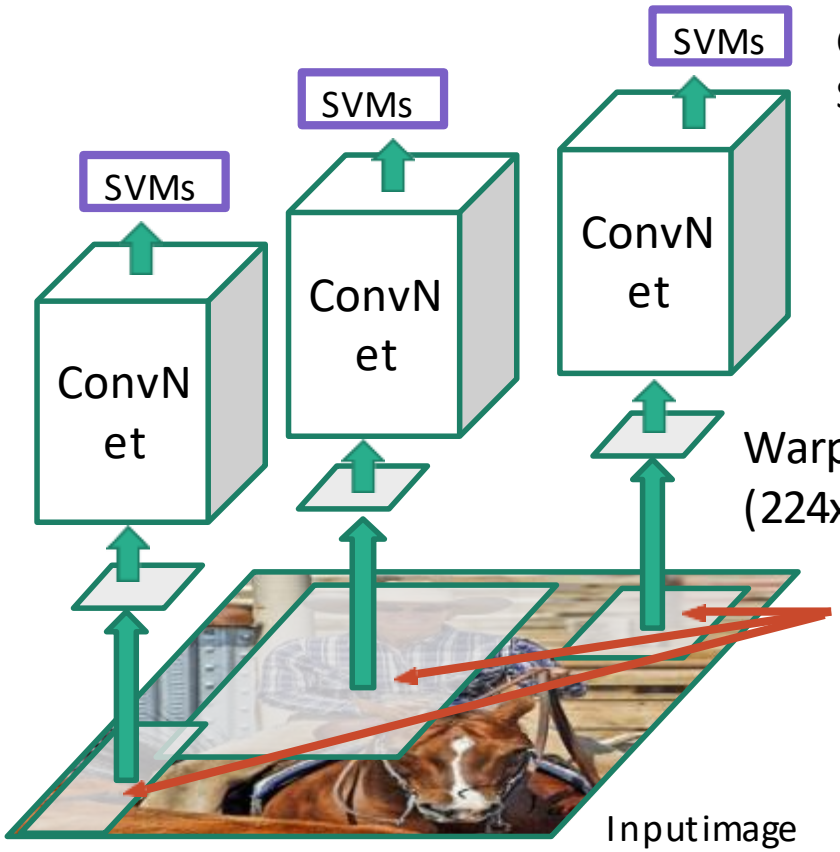
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



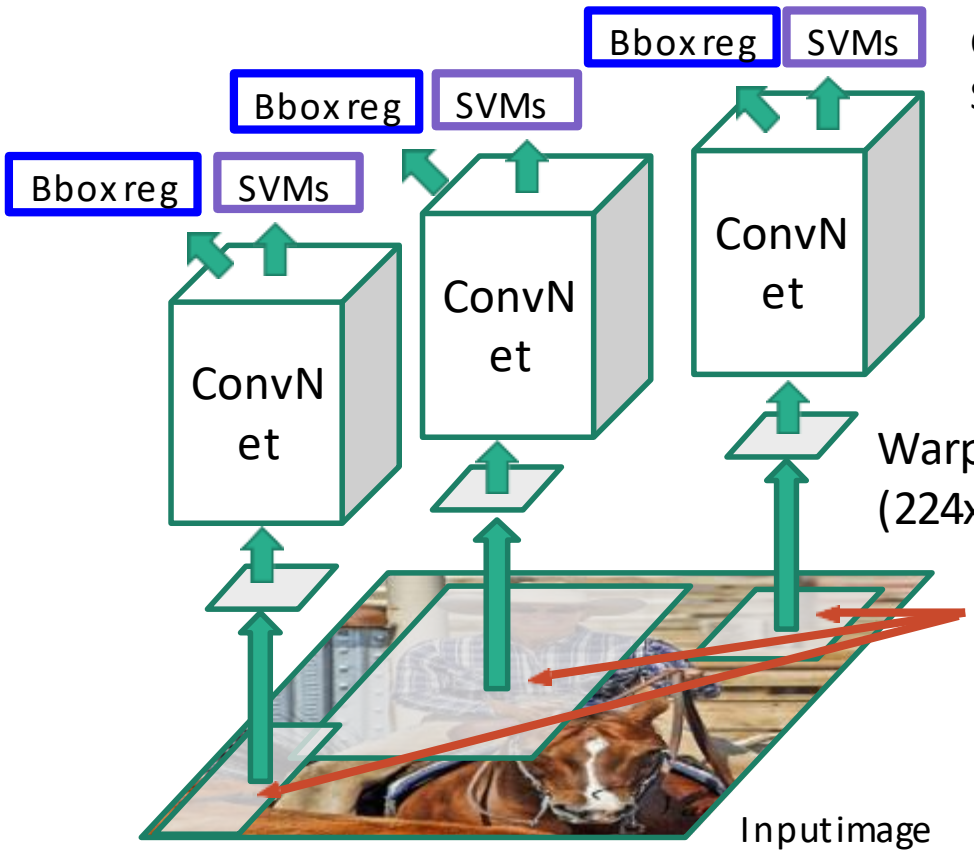
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



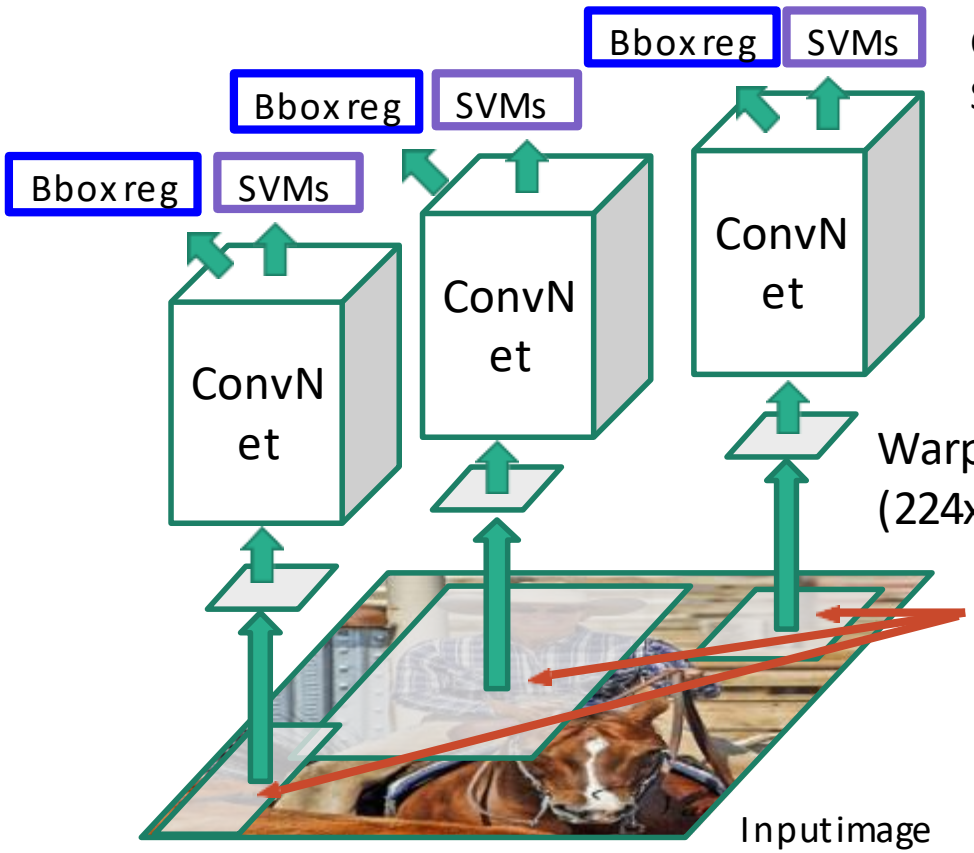
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



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

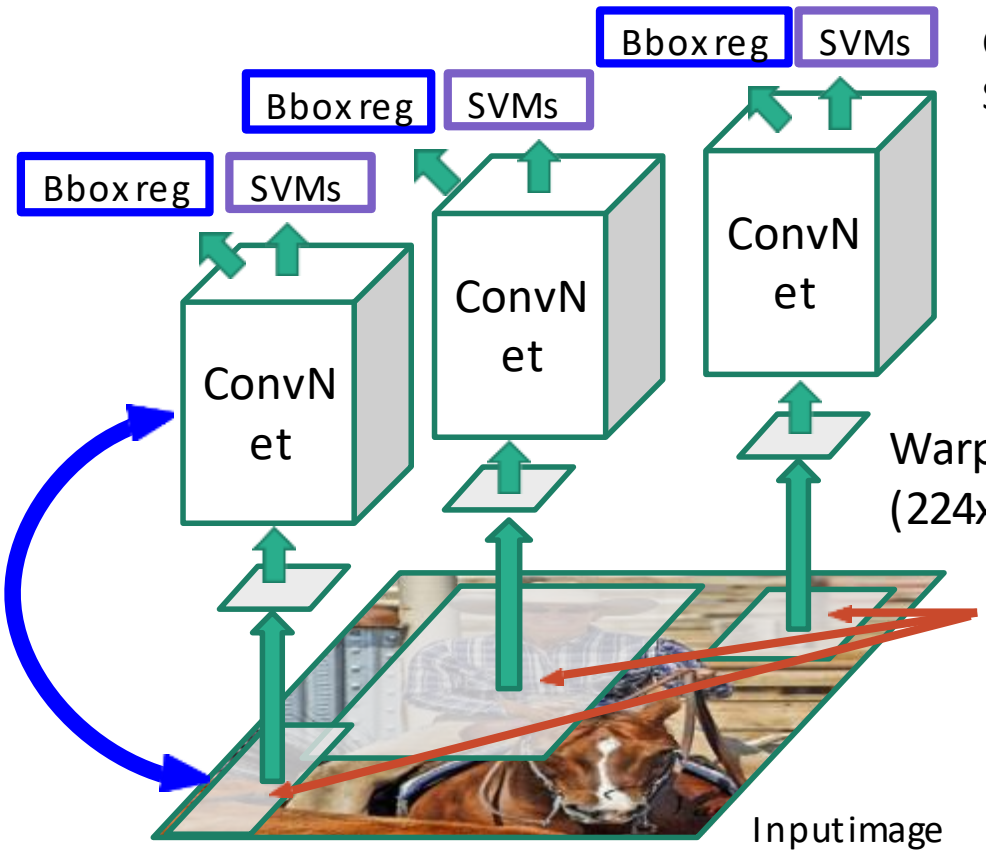


Problem: Very slow! Need to do ~2k independent forward passes for each image!

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

"Slow" R-CNN

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



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

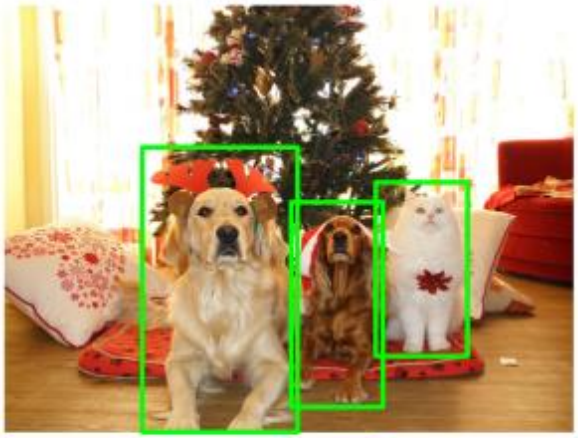
Problem: Very slow! Need to do ~2k independent forward passes for each image!

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

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

"Slow" R-CNN Training

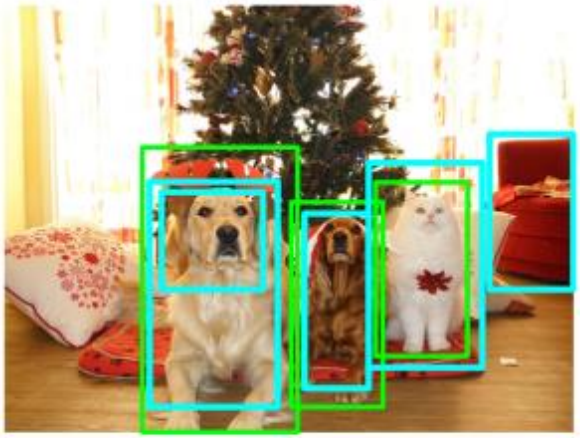
Input Image



Ground-Truth boxes

"Slow" R-CNN Training

Input Image

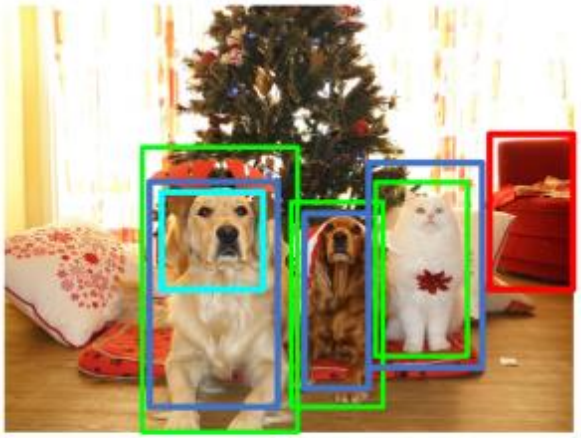


Ground-Truth boxes

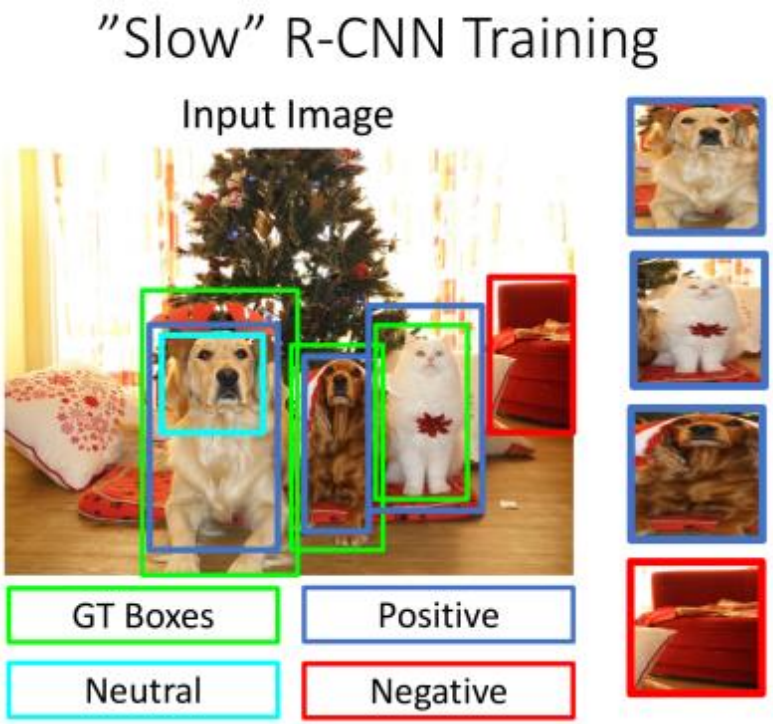
Region Proposals

"Slow" R-CNN Training

Input Image

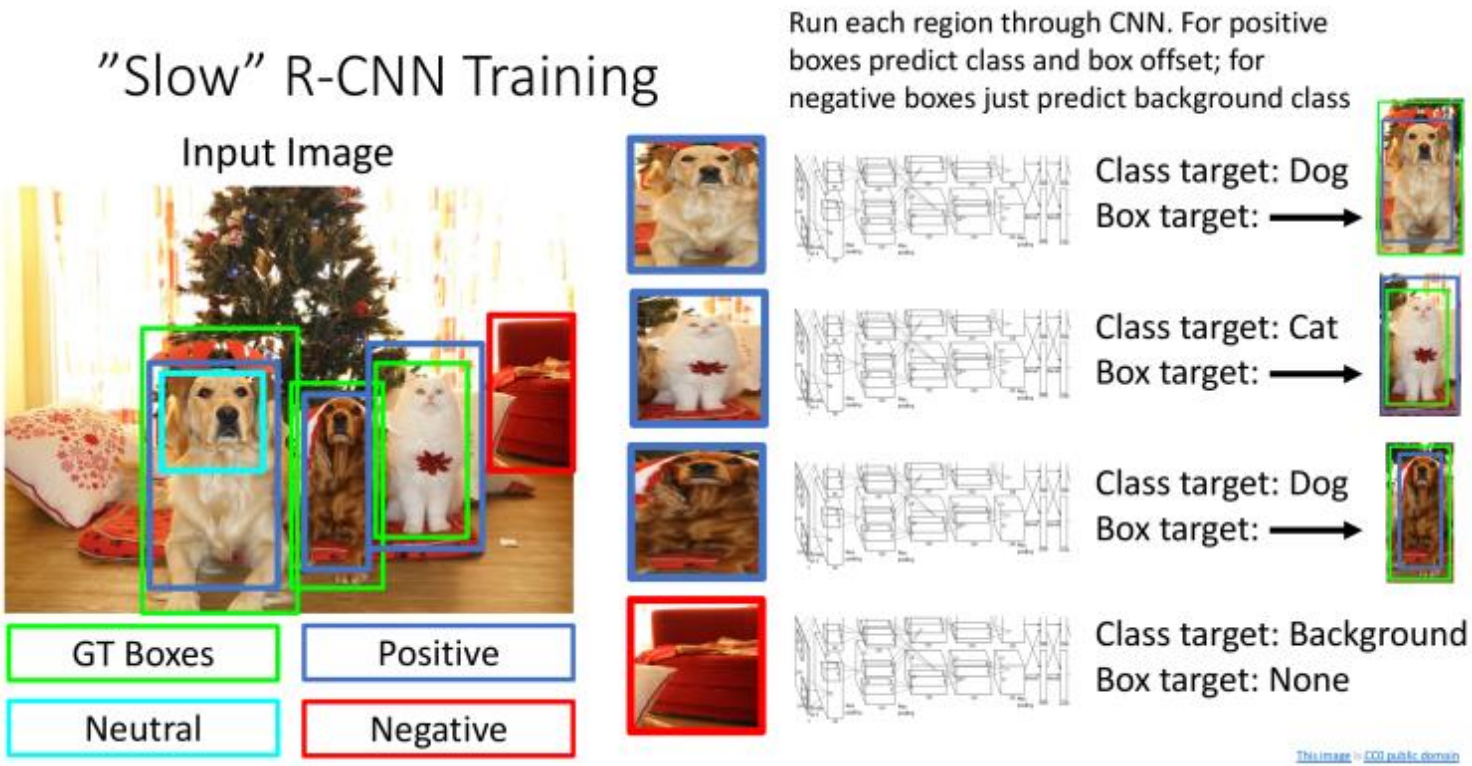


Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes



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

R-CNN

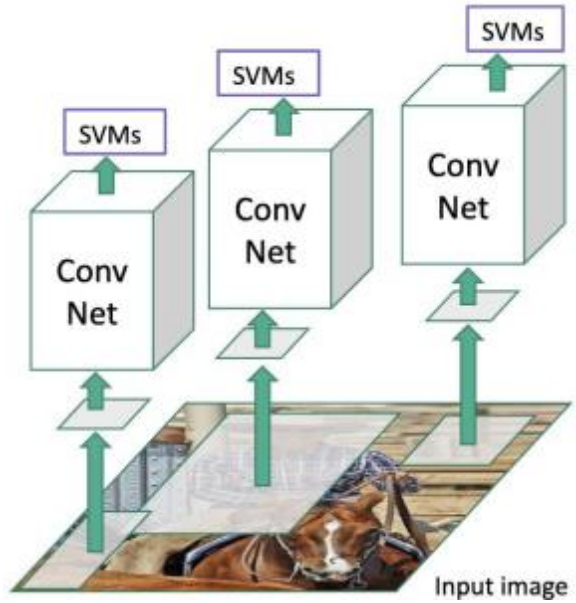

"Slow" R-CNN Training
Input Image
GT Boxes
Positive
Neutral
Negative
Run each region through CNN. For positive boxes predict class and box offset; for negative boxes just predict background class
Class target: Dog
Box target:
Class target: Cat
Box target:
Class target: Dog
Box target:
Class target: Background
Box target: None

Fast R-CNN

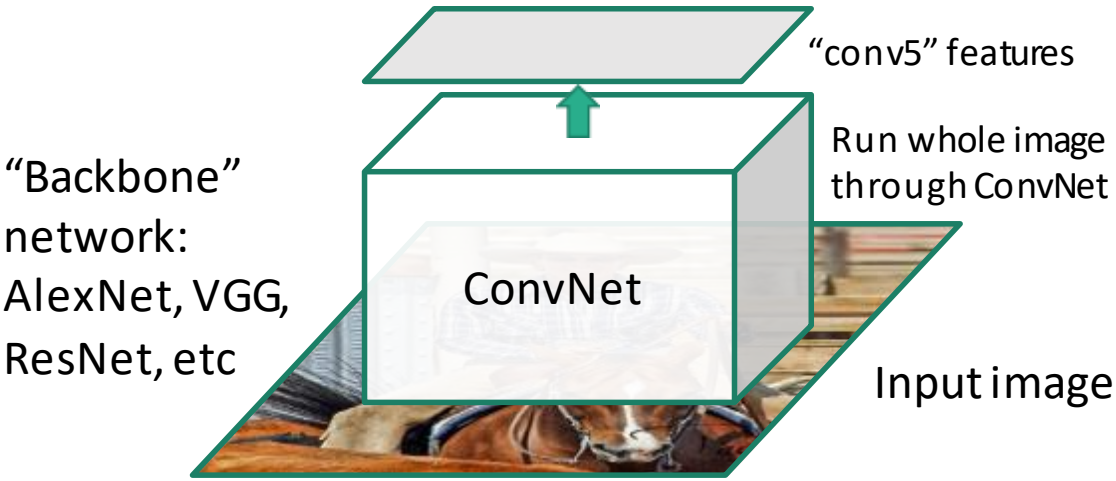
Input image

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

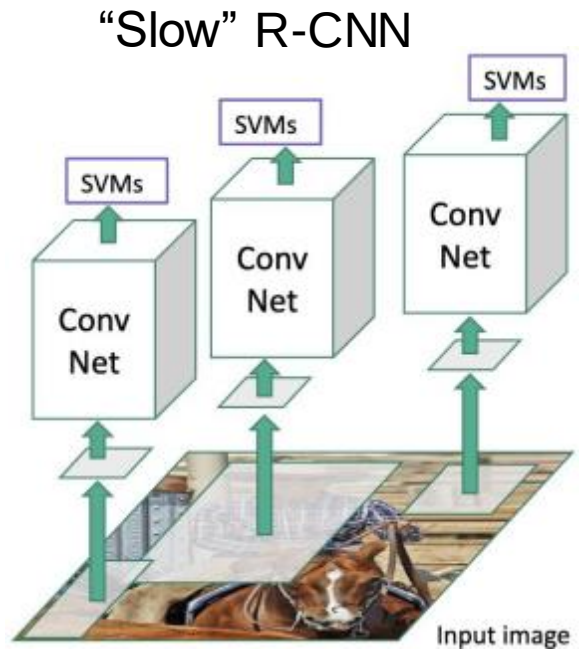
"Slow" R-CNN



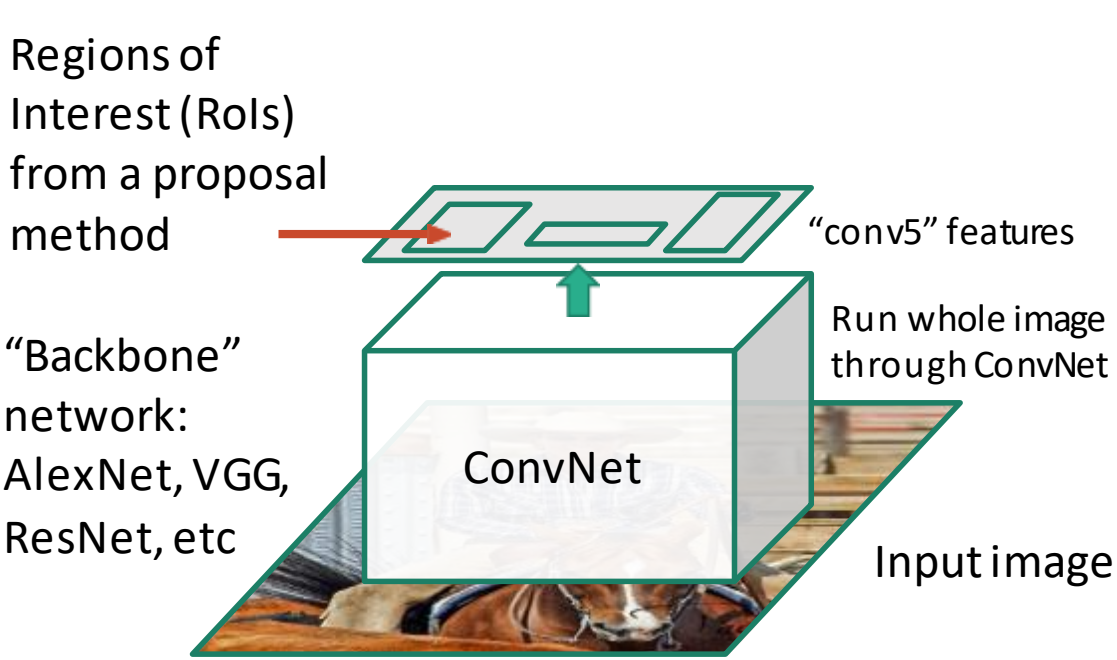
Fast R-CNN



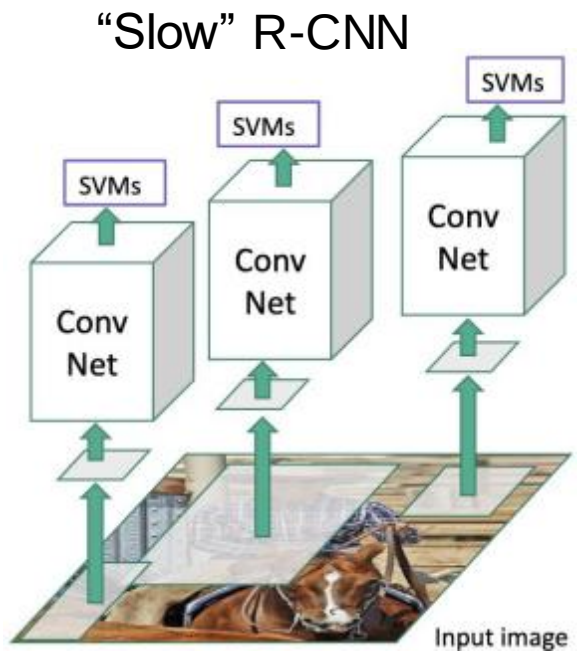
Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



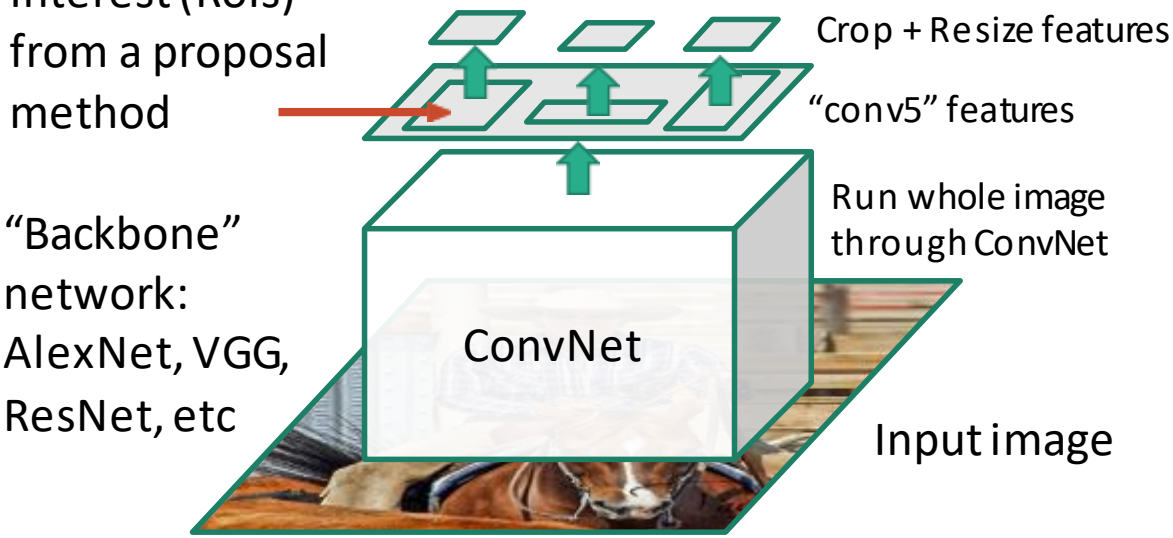
Fast R-CNN



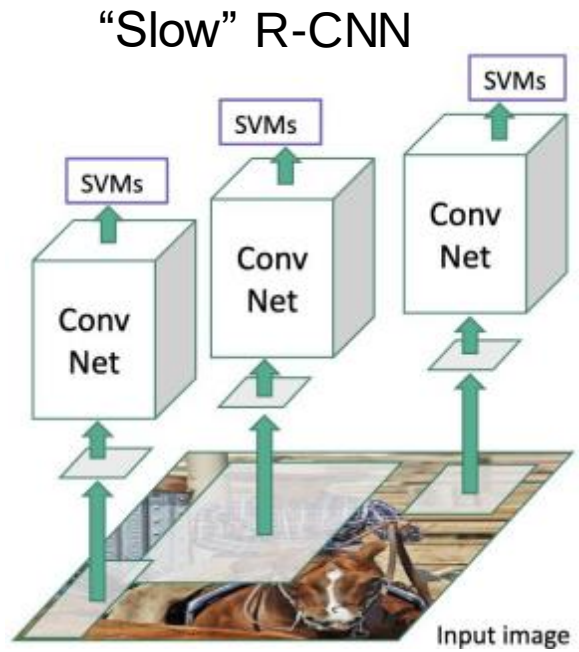
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



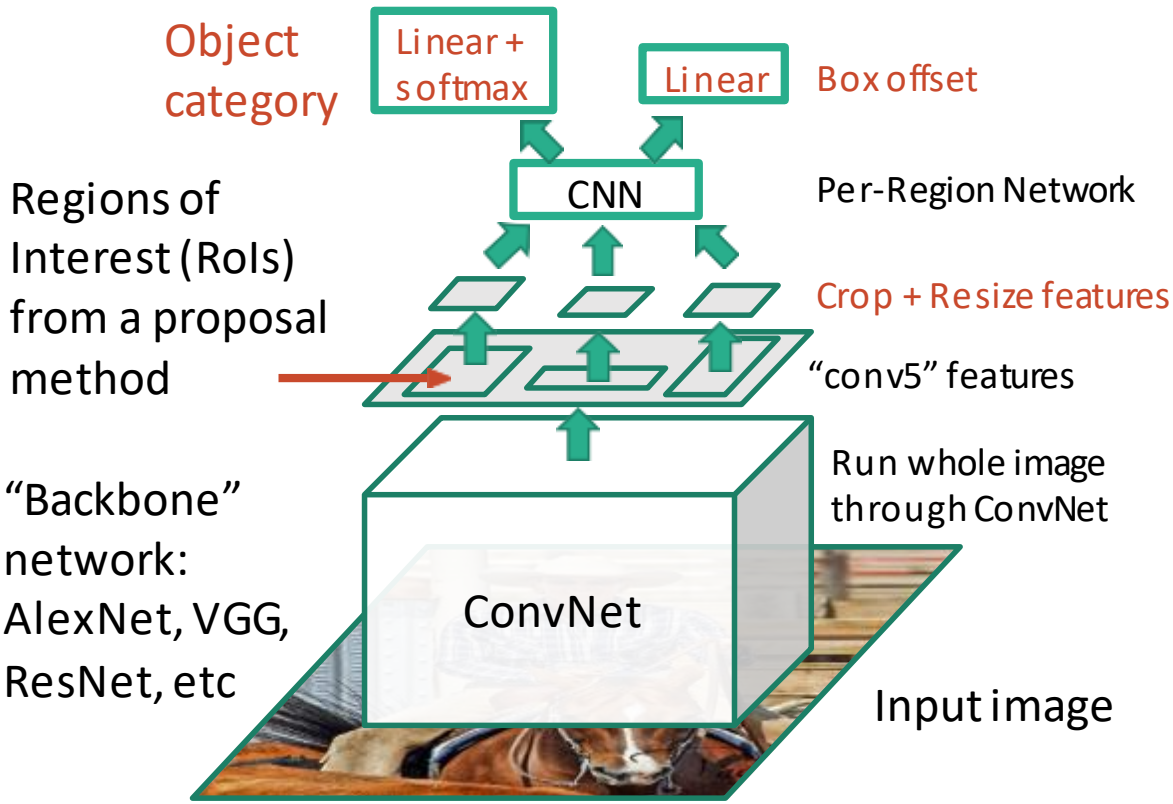
Fast R-CNN



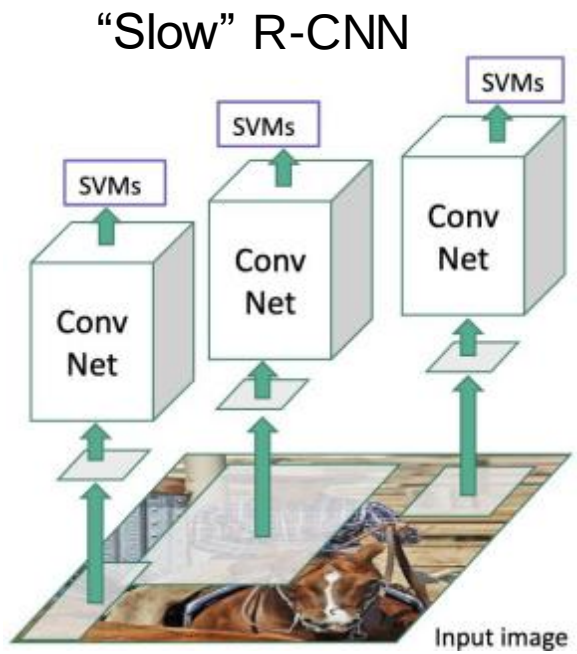
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



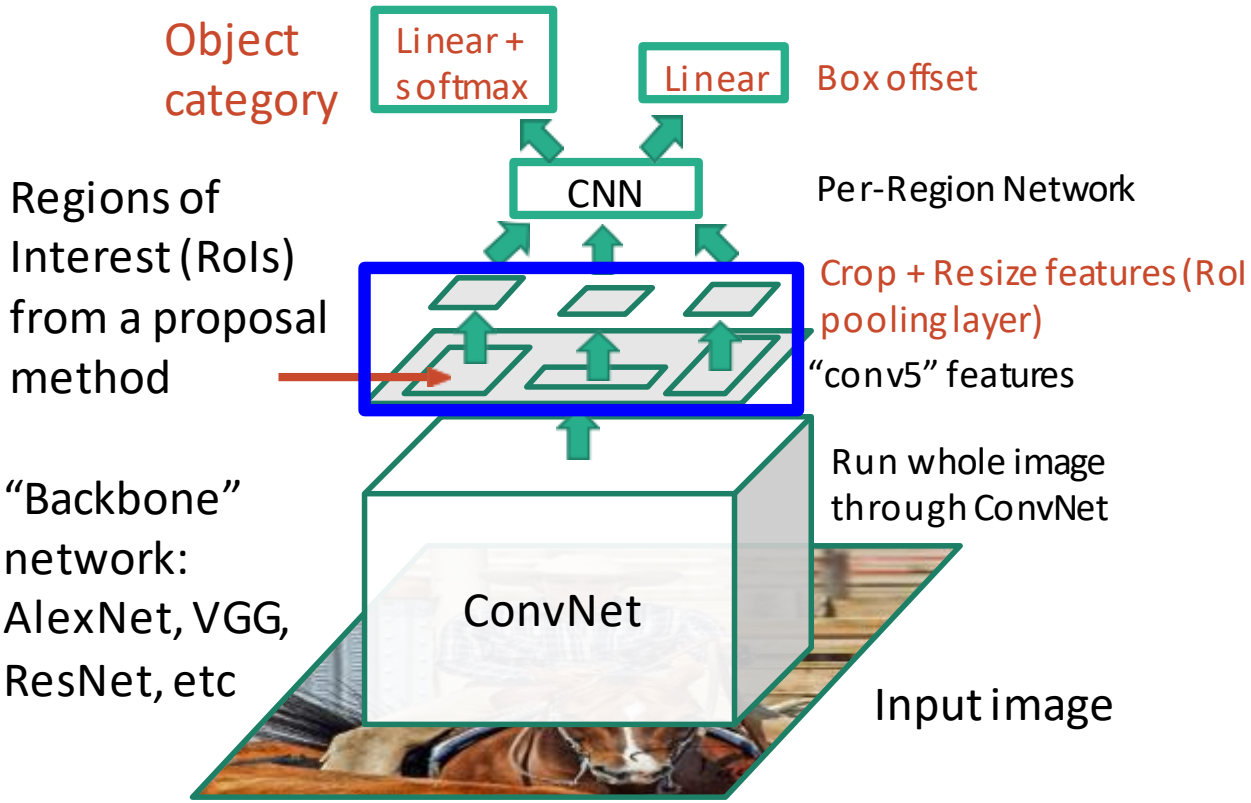
Fast R-CNN



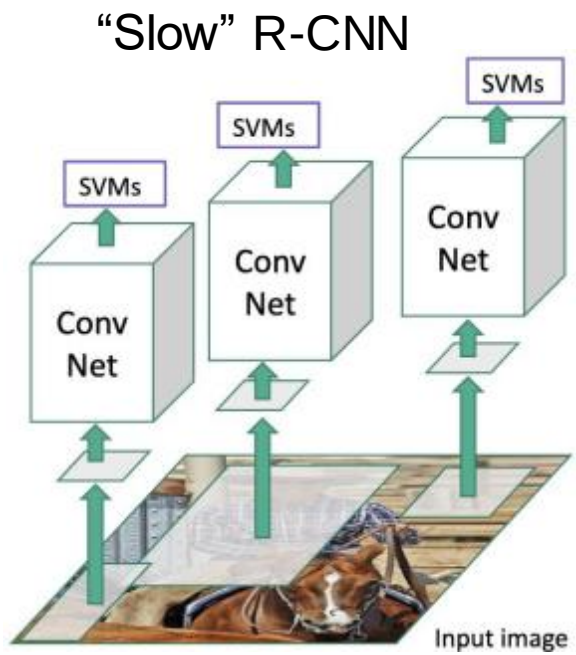
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



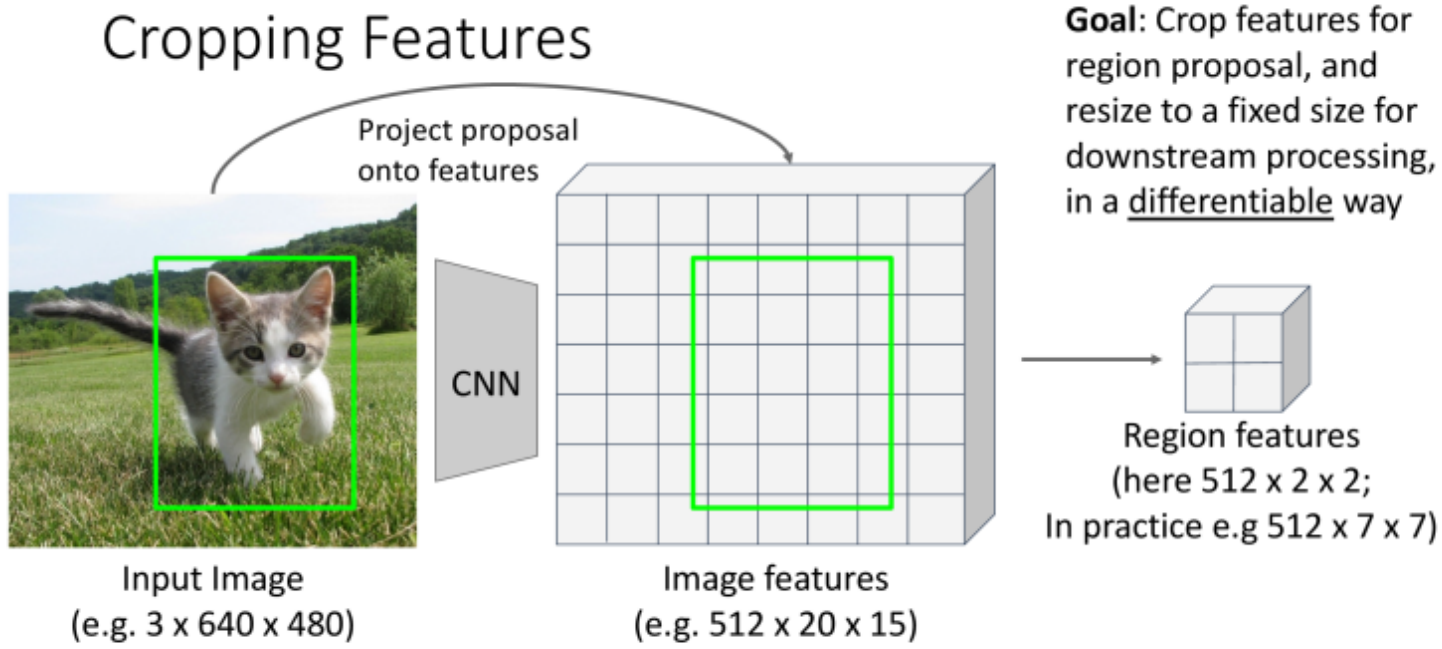
Fast R-CNN



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

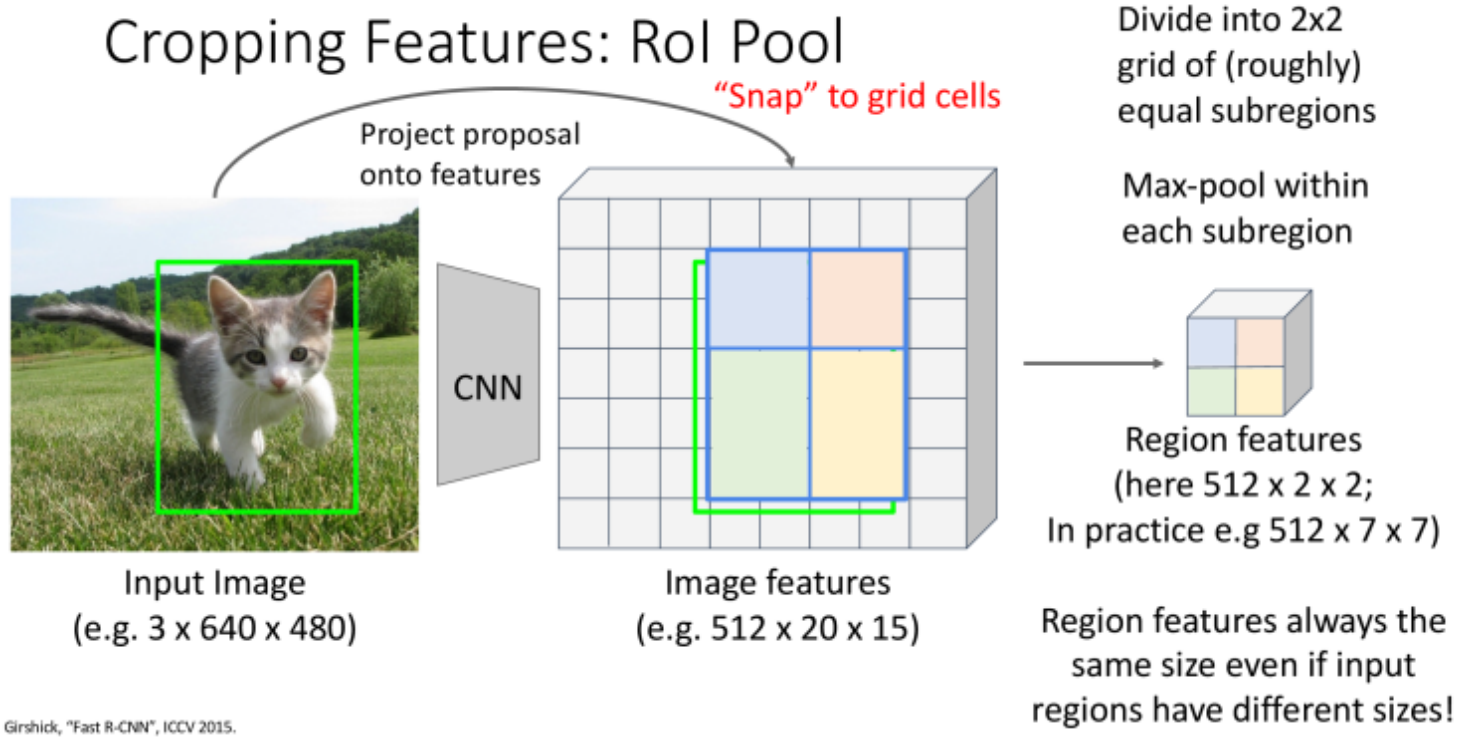


RoI pooling



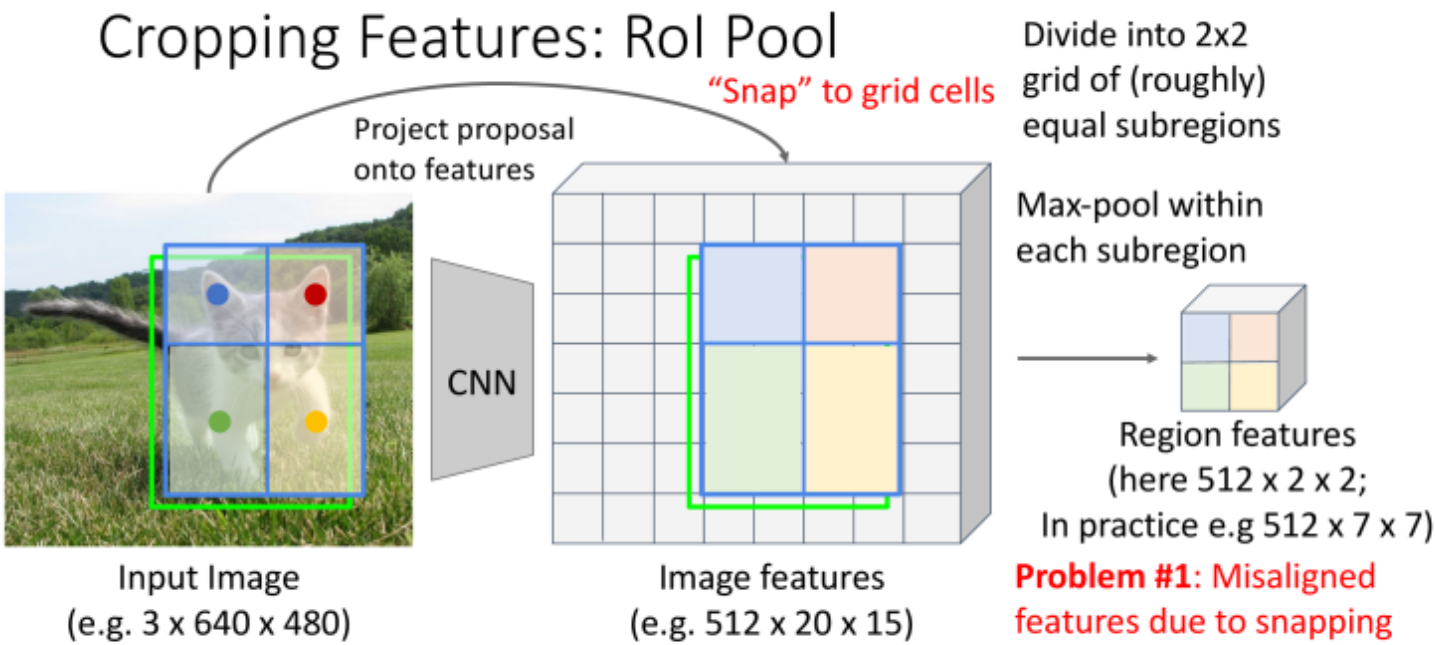
Girshick, "Fast R-CNN", ICCV 2015.

RoI pooling

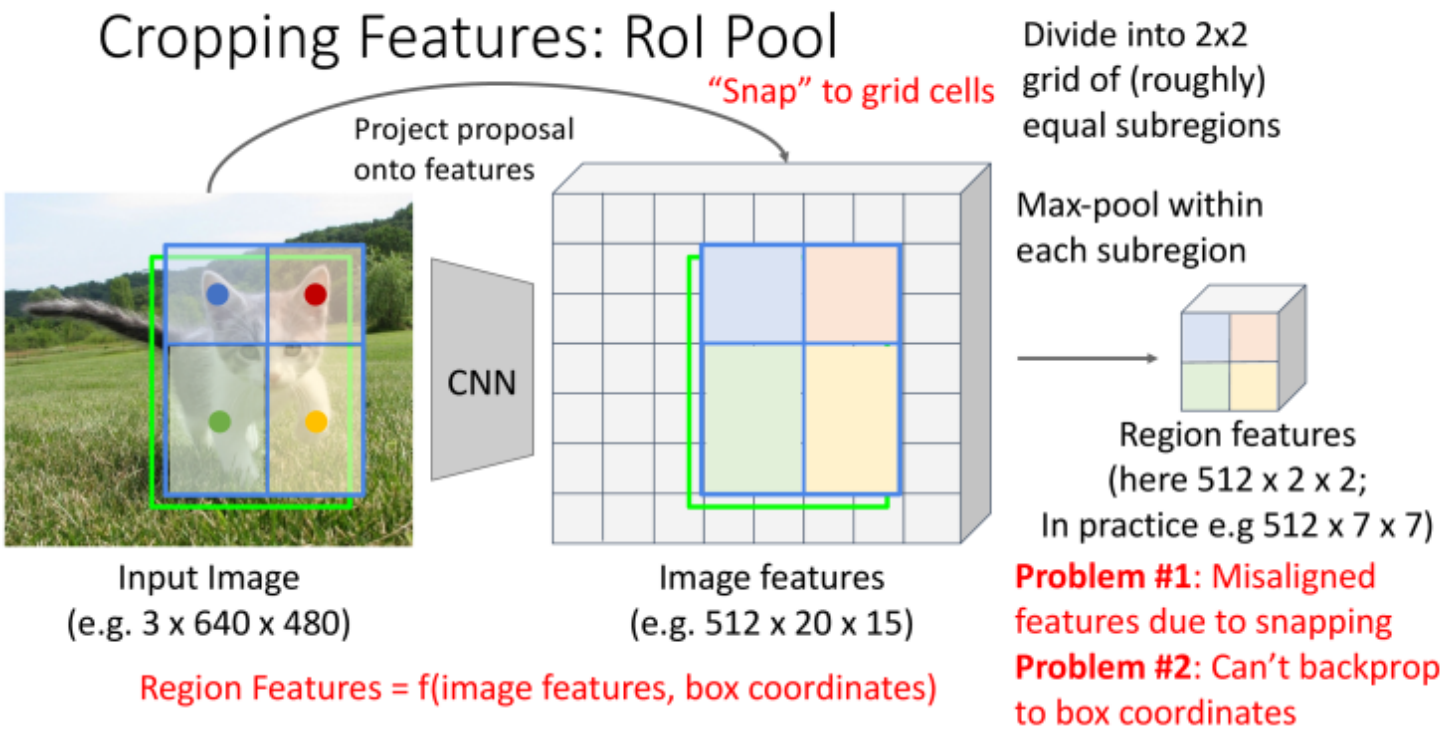


Girshick, "Fast R-CNN", ICCV 2015.

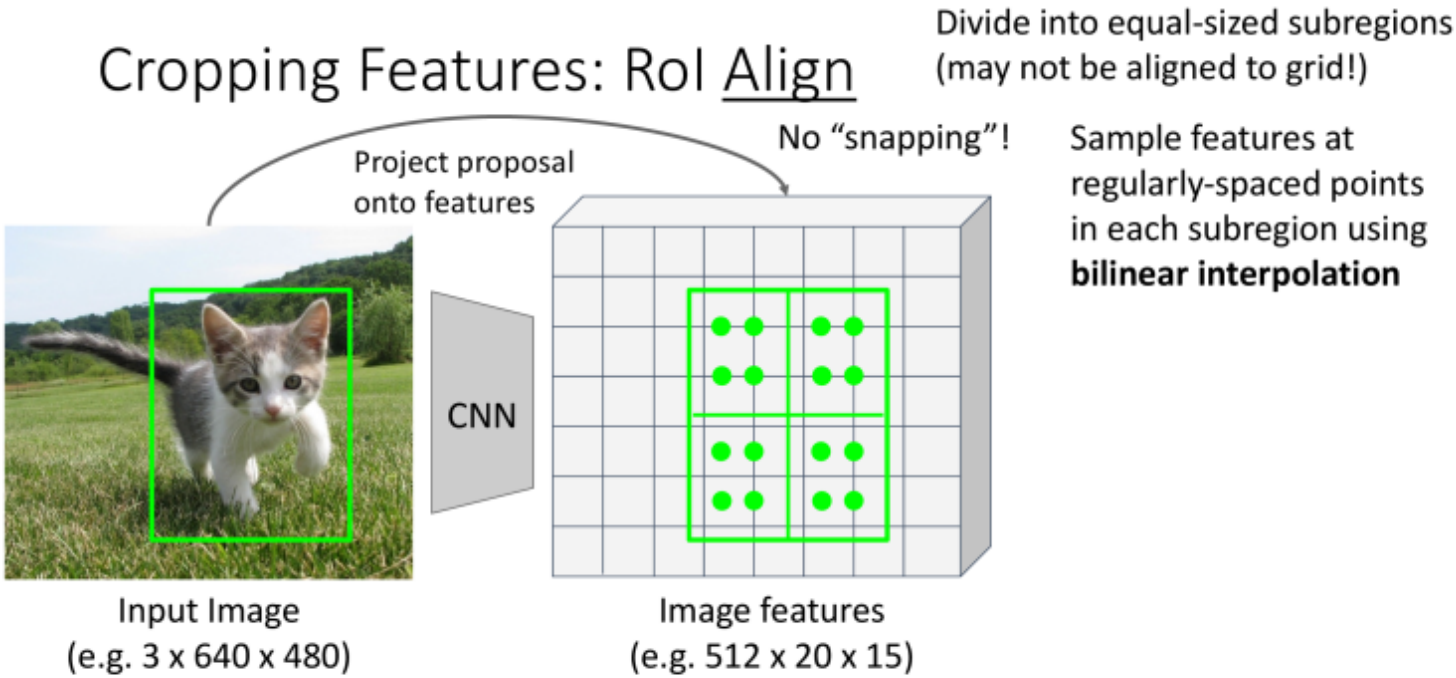
RoI pooling


Cropping Features: RoI Pool
"Snap" to grid cells
Project proposal onto features
CNN
Input Image
(e.g. 3 x 640 x 480)
Image features
(e.g. 512 x 20 x 15)
Divide into 2x2 grid of (roughly) equal subregions
Max-pool within each subregion
Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)
Problem #1: Misaligned features due to snapping

RoI pooling

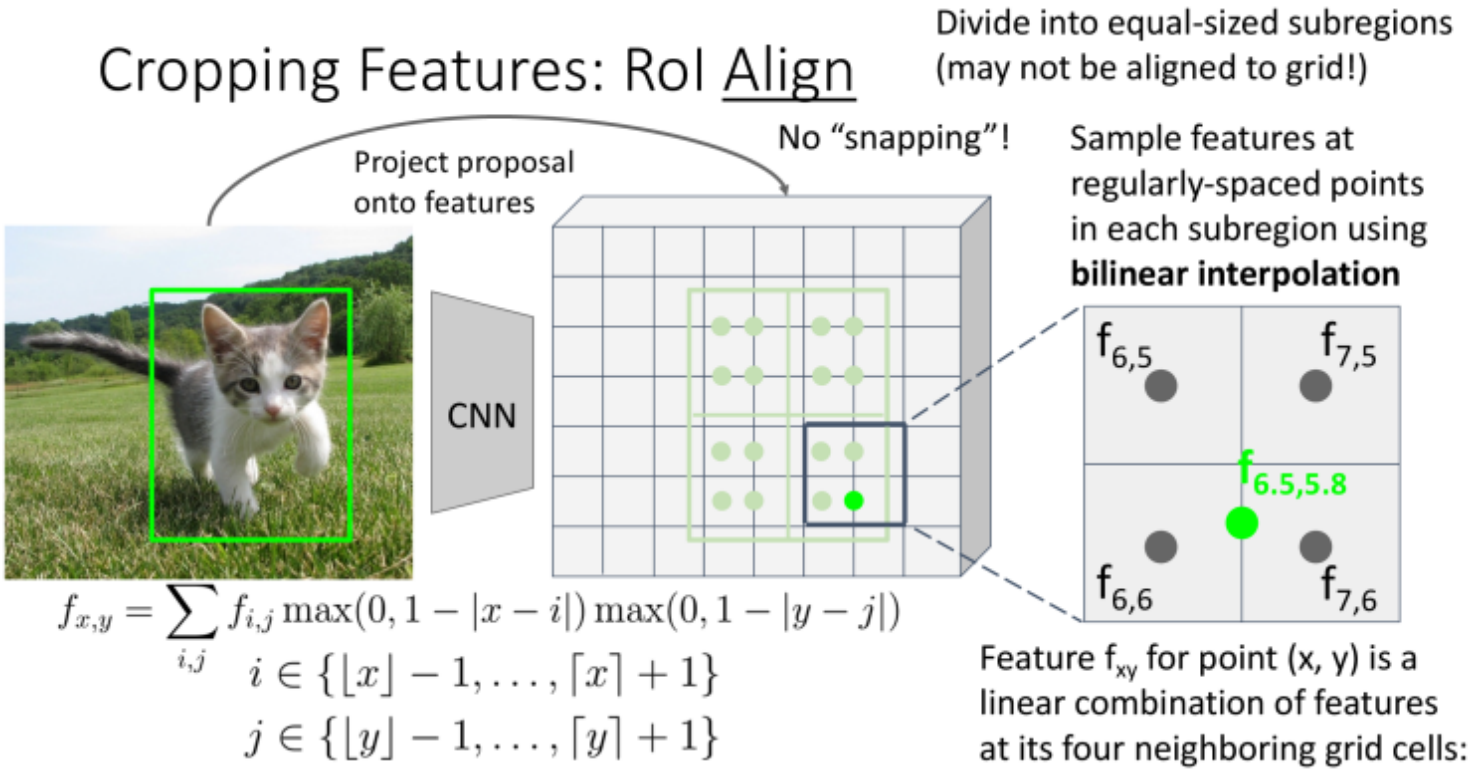

Cropping Features: RoI Pool
"Snap" to grid cells
Project proposal onto features
CNN
Input Image
(e.g. 3 x 640 x 480)
Image features
(e.g. 512 x 20 x 15)
Region Features = f(image features, box coordinates)
Divide into 2x2 grid of (roughly) equal subregions
Max-pool within each subregion
Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)
Problem #1: Misaligned features due to snapping
Problem #2: Can't backprop to box coordinates

RoI Align

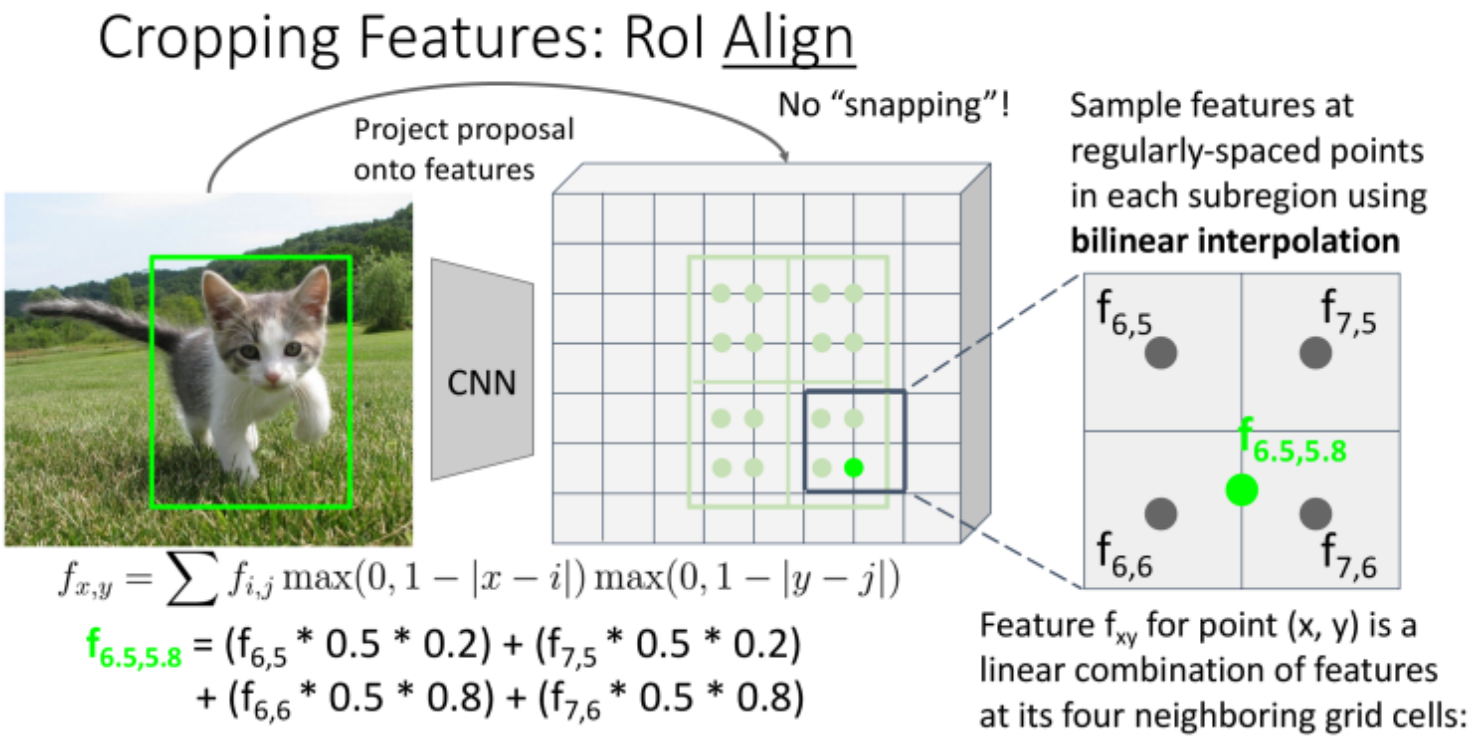


He et al, "Mask R-CNN", ICCV 2017

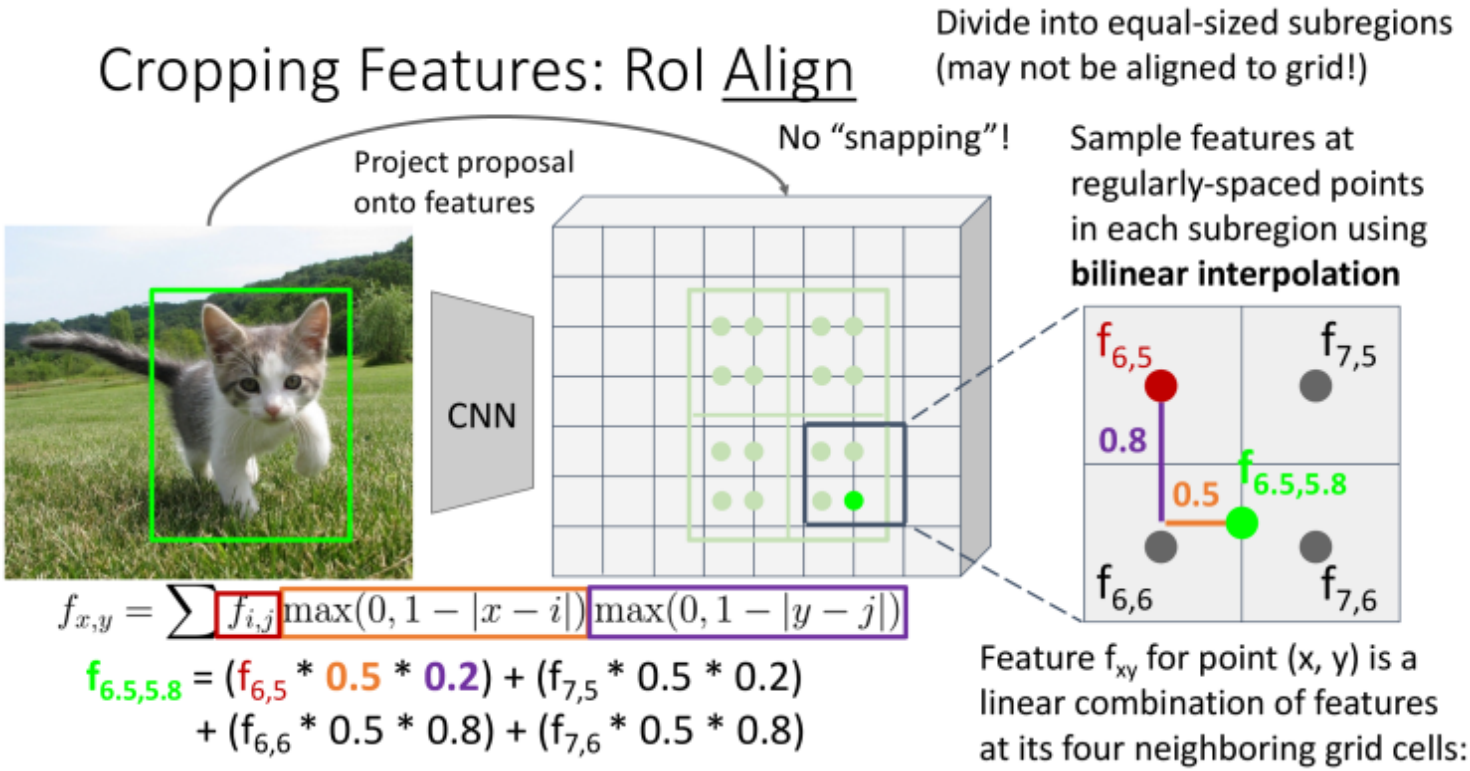
RoI Align


Cropping Features: RoI Align
Divide into equal-sized subregions
(may not be aligned to grid!)
No "snapping"!
Project proposal
onto features
CNN
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation
f6,5
f7,5
f6.5,5.8
f6,6
f7,6
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\}$
$j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$
Feature f_{xy} for point (x, y) is a
linear combination of features
at its four neighboring grid cells:

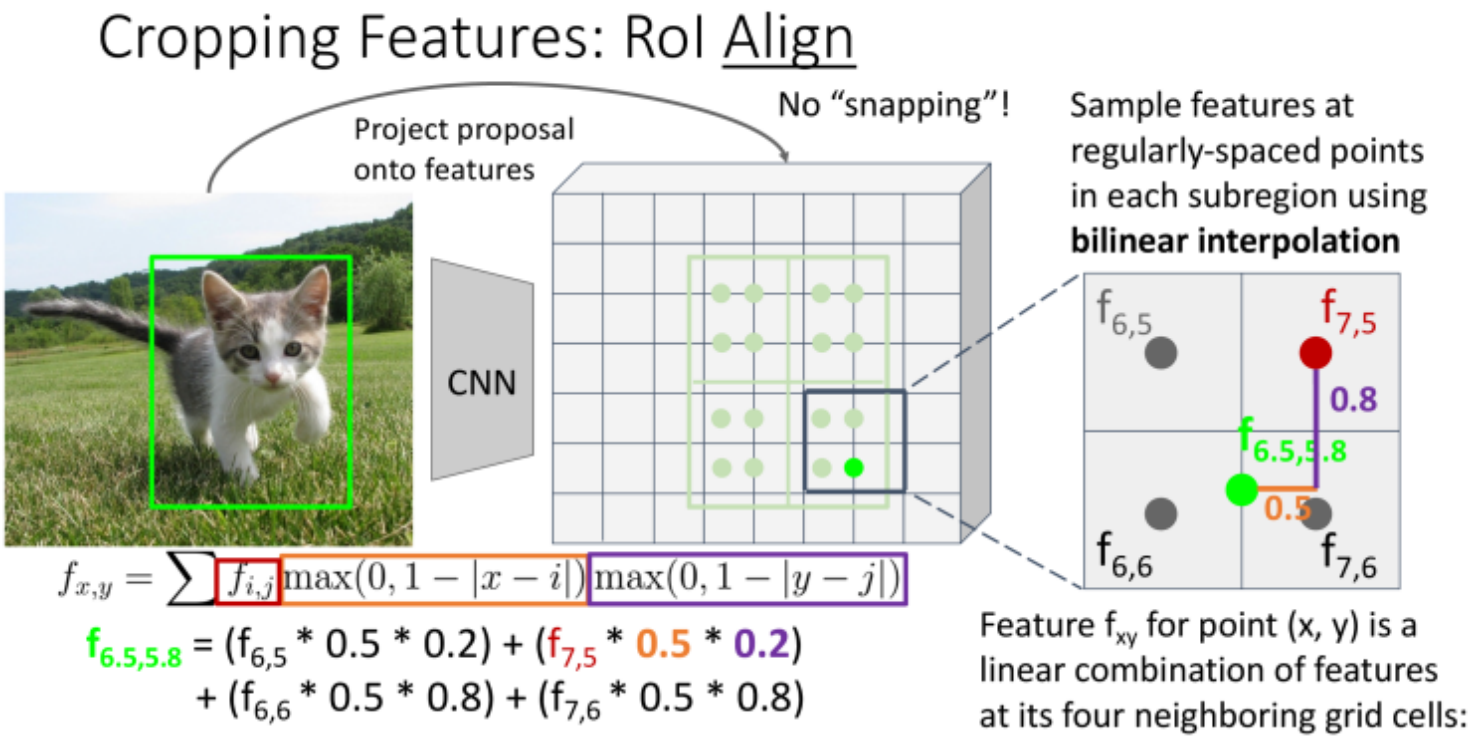
RoI Align


Cropping Features: RoI Align
Project proposal onto features
No "snapping"!
CNN
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**
$f_{6,5}$
$f_{7,5}$
$f_{6.5,5.8}$
$f_{6,6}$
$f_{7,6}$
$f_{x,y} = \sum f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

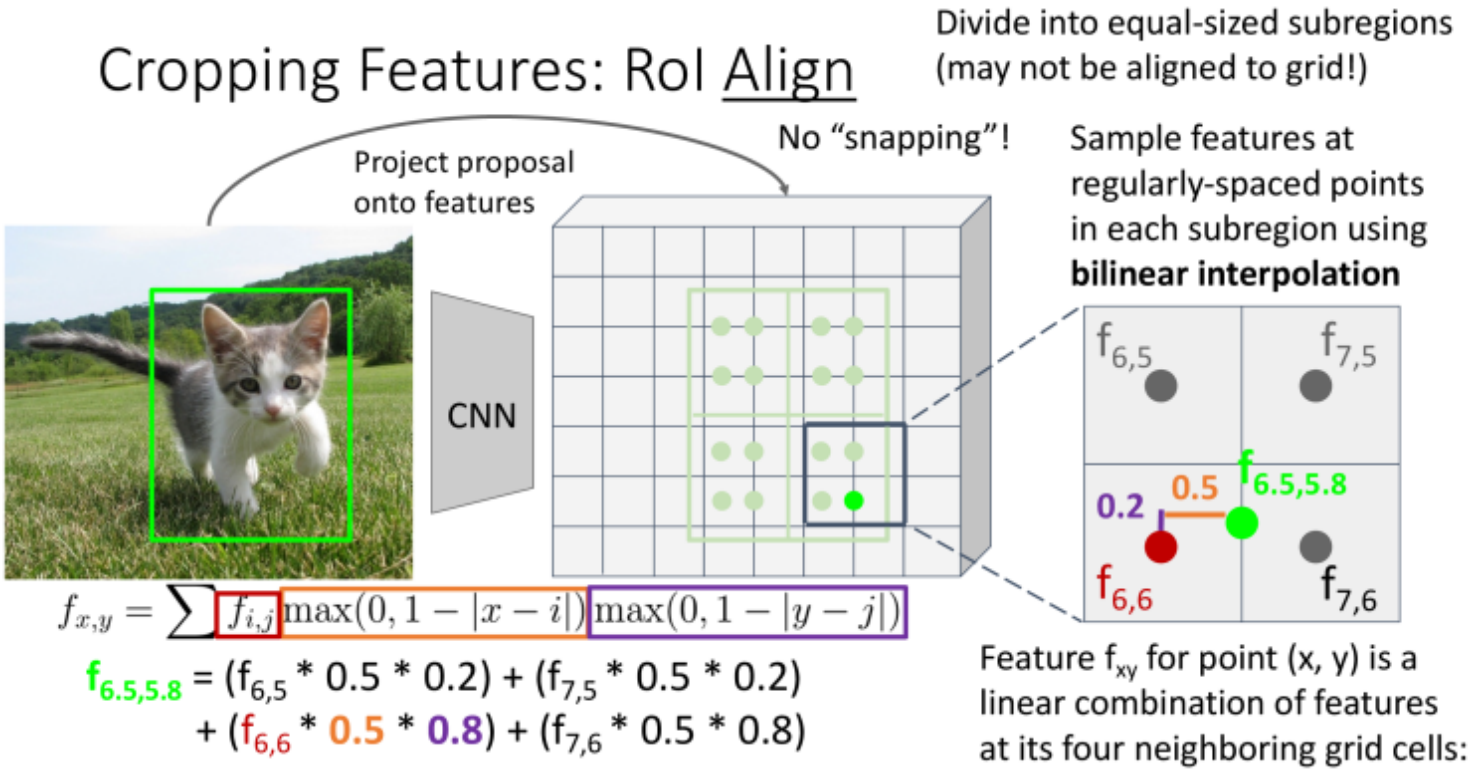
RoI Align


Cropping Features: RoI Align
Divide into equal-sized subregions
(may not be aligned to grid!)
No "snapping"!
Project proposal
onto features
CNN
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)
$f_{6,5}$
$f_{7,5}$
0.8
0.5
$f_{6.5,5.8}$
$f_{6,6}$
$f_{7,6}$
Feature f_{xy} for point (x, y) is a
linear combination of features
at its four neighboring grid cells:

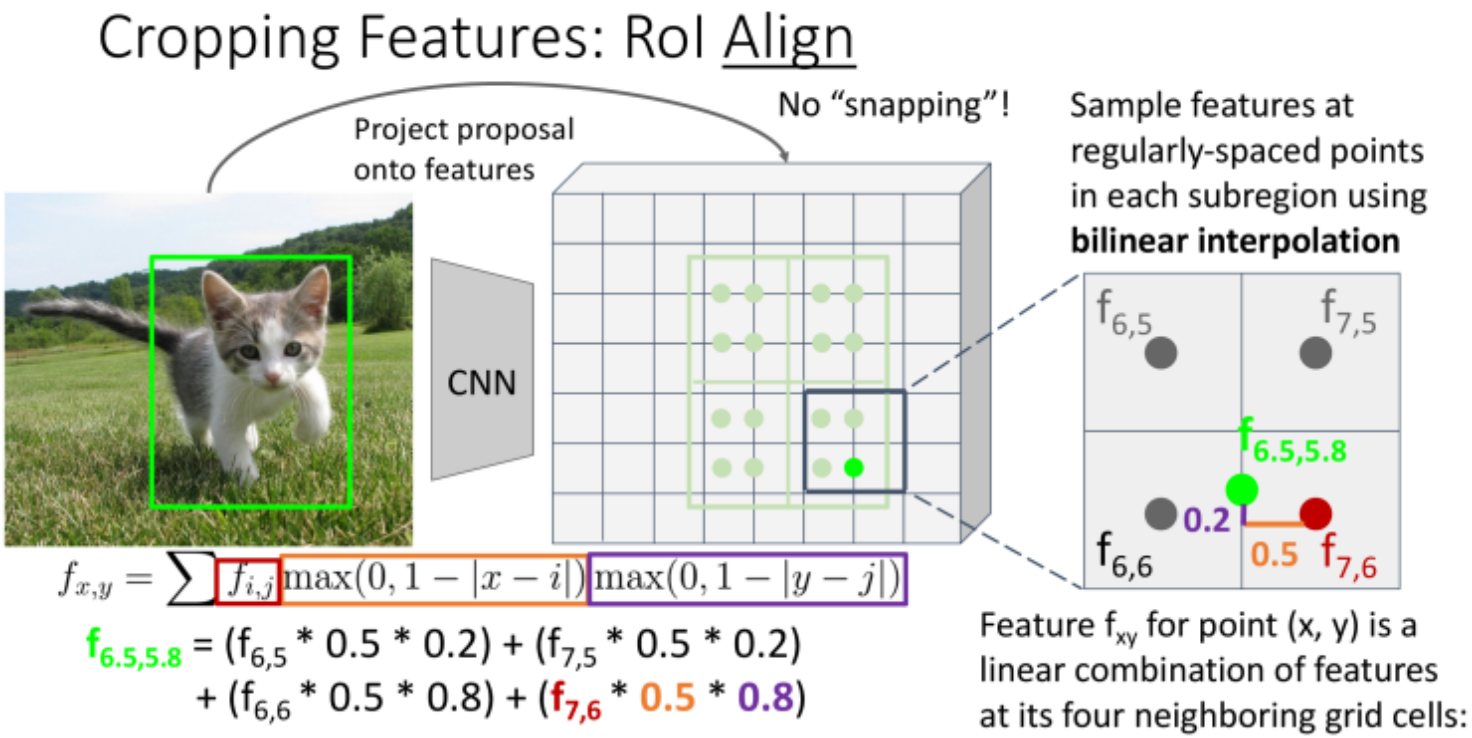
RoI Align


Cropping Features: RoI Align
Project proposal onto features
No "snapping"!
CNN
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)
$f_{6,5}$
$f_{7,5}$
0.8
$f_{6.5,5.8}$
0.5
$f_{6,6}$
$f_{7,6}$
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

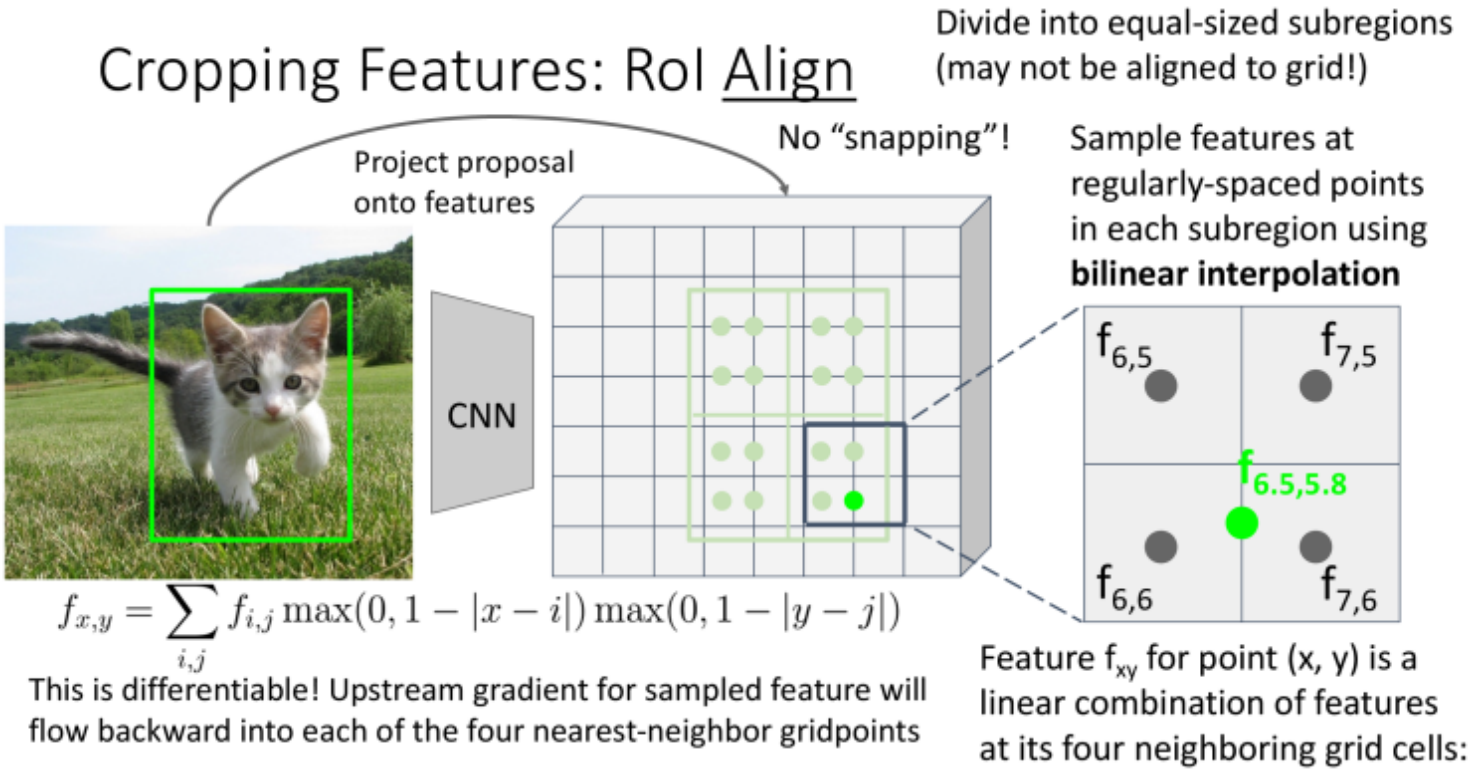
RoI Align


Cropping Features: RoI Align
Divide into equal-sized subregions
(may not be aligned to grid!)
No "snapping"!
Project proposal
onto features
CNN
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$f_{6,5}$
$f_{7,5}$
$f_{6.5,5.8}$
0.5
0.2
$f_{6,6}$
$f_{7,6}$
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)
Feature f_{xy} for point (x, y) is a
linear combination of features
at its four neighboring grid cells:

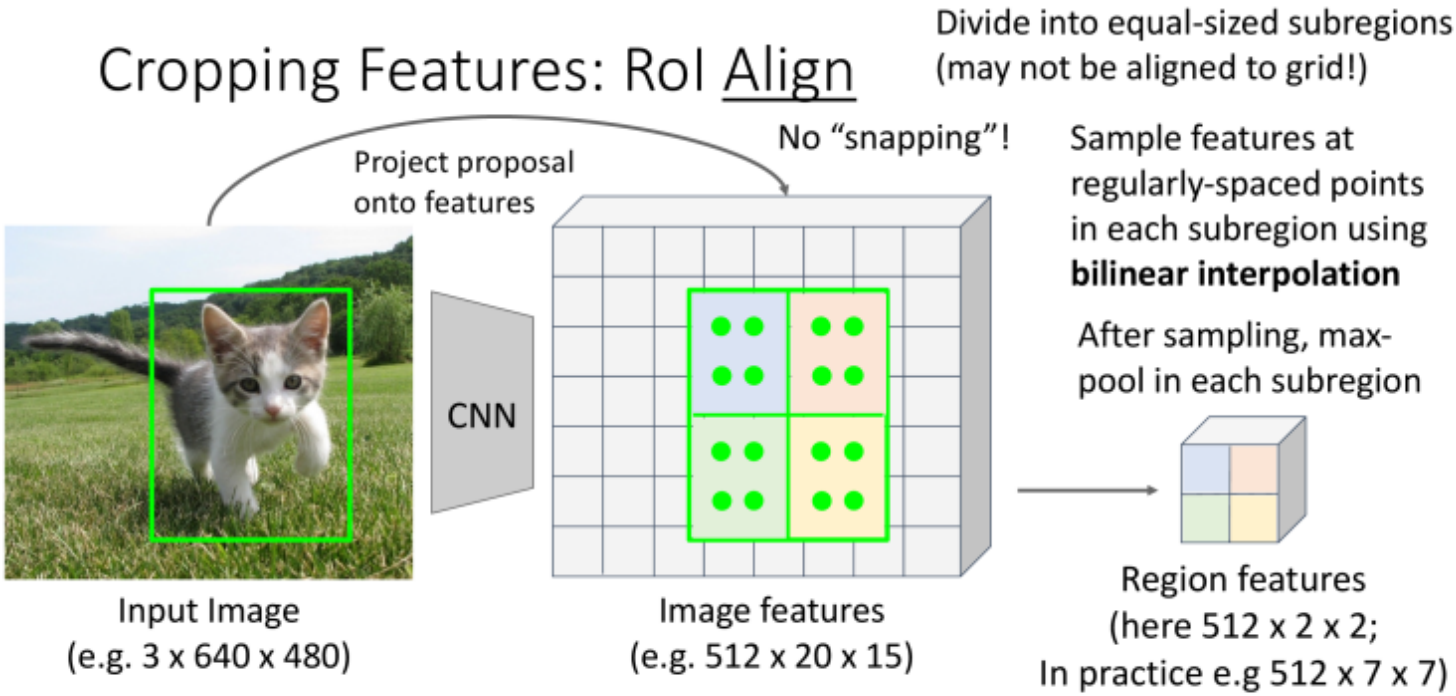
RoI Align


Cropping Features: RoI Align
Project proposal onto features
No "snapping"!
CNN
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
$f_{6,5}$
$f_{7,5}$
$f_{6.5,5.8}$
0.2
$f_{6,6}$
0.5
$f_{7,6}$
$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

RoI Align

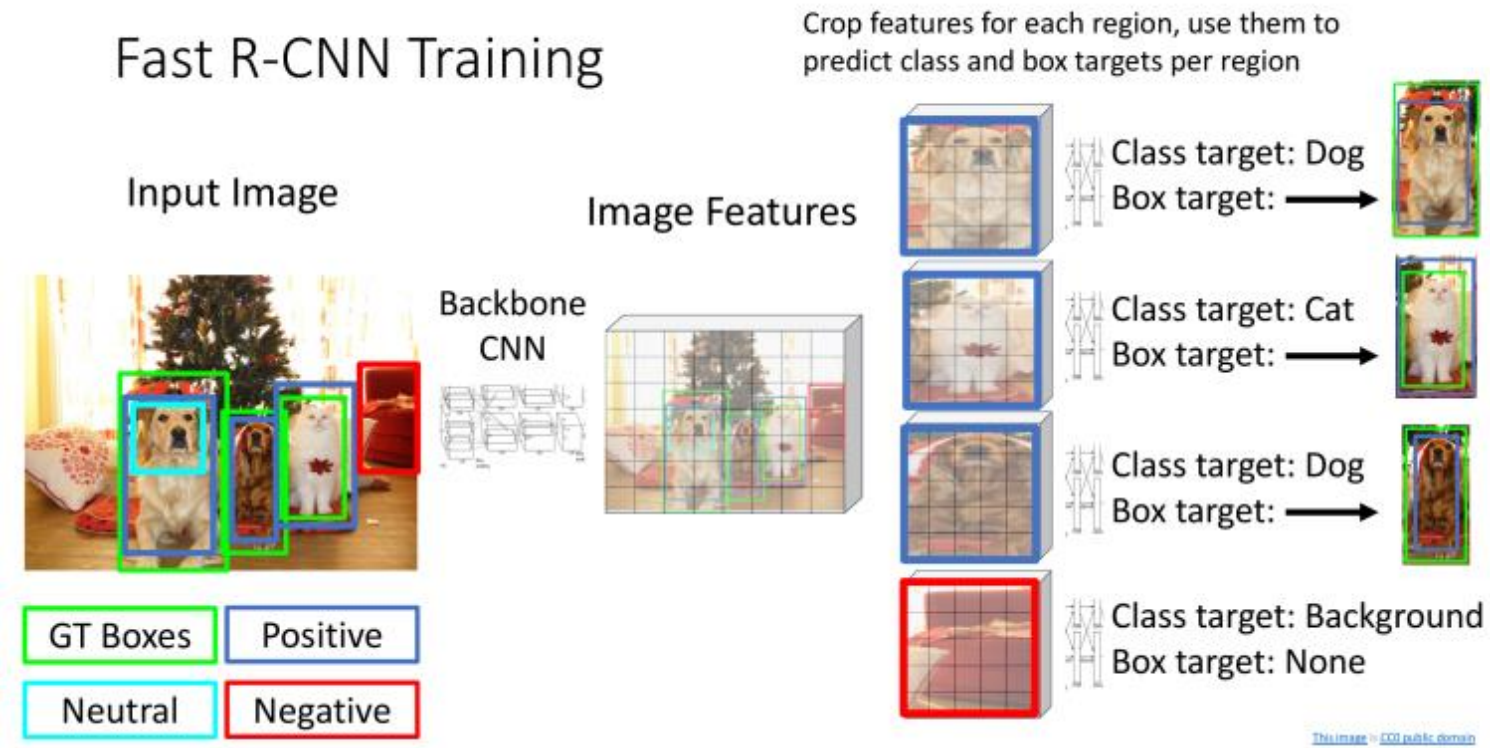

Cropping Features: RoI Align
Divide into equal-sized subregions
(may not be aligned to grid!)
No "snapping"!
Project proposal
onto features
CNN
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation
$f_{6,5}$
$f_{7,5}$
$f_{6.5,5.8}$
$f_{6,6}$
$f_{7,6}$
$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$
This is differentiable! Upstream gradient for sampled feature will
flow backward into each of the four nearest-neighbor gridpoints
Feature f_{xy} for point (x, y) is a
linear combination of features
at its four neighboring grid cells:

RoI Align

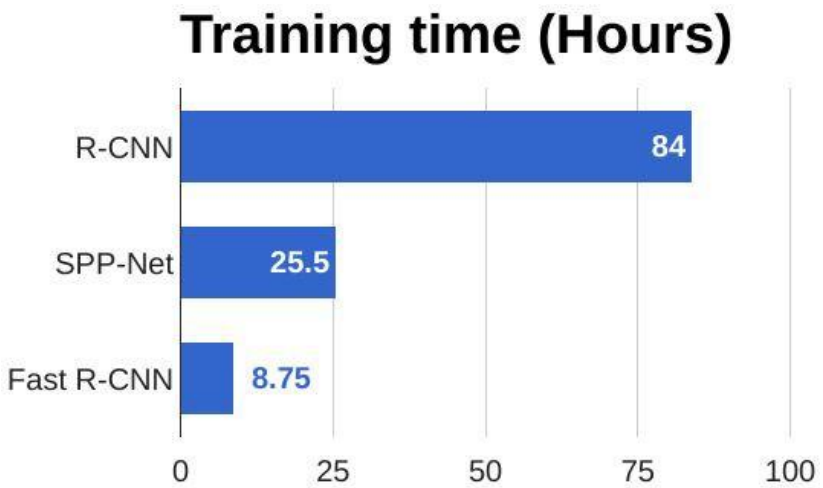


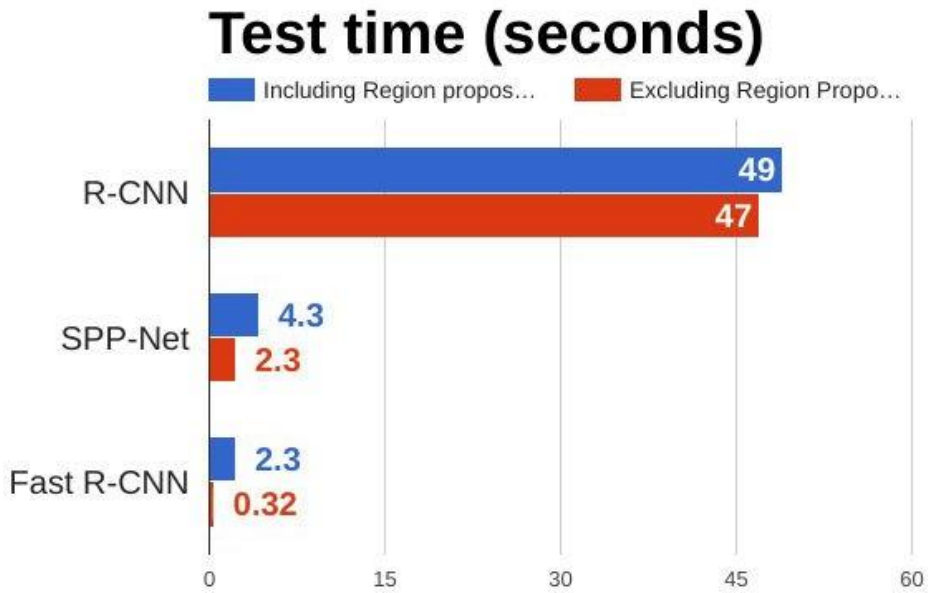
Output features now aligned to input box! And we can backprop to box coordinates!

Fast R-CNN


Fast R-CNN Training
Input Image
Backbone CNN
Image Features
Crop features for each region, use them to predict class and box targets per region
Class target: Dog
Box target:
Class target: Cat
Box target:
Class target: Dog
Box target:
Class target: Background
Box target: None
GT Boxes
Positive
Neutral
Negative

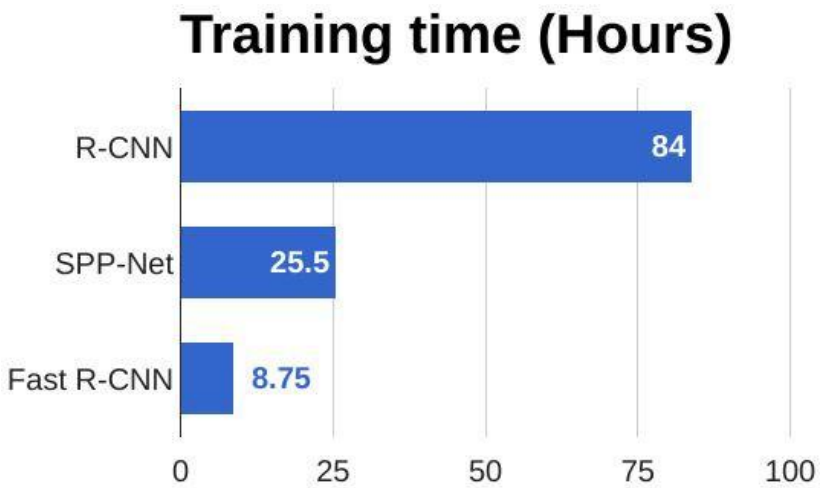
R-CNN vs Fast R-CNN

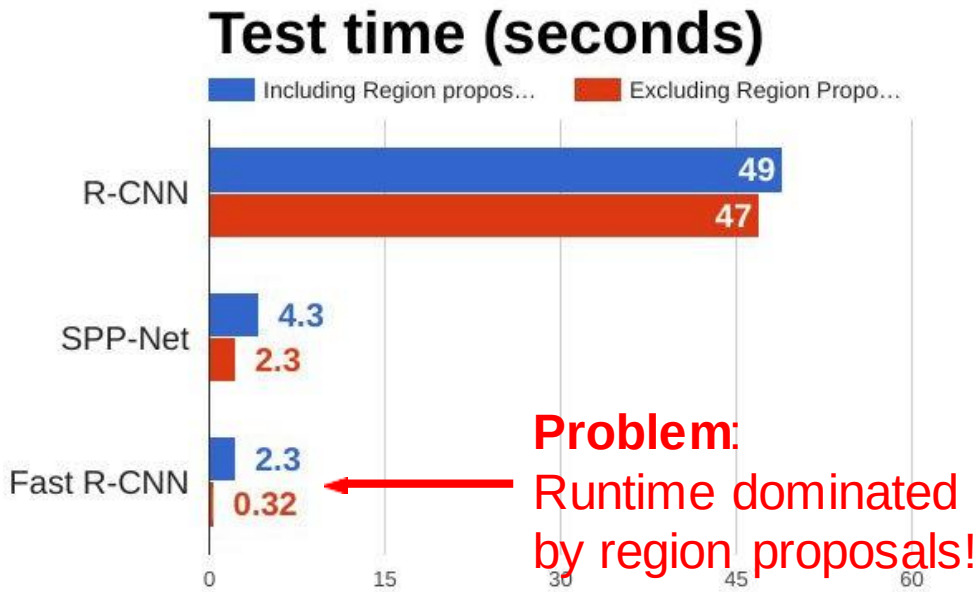




Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

R-CNN vs Fast R-CNN





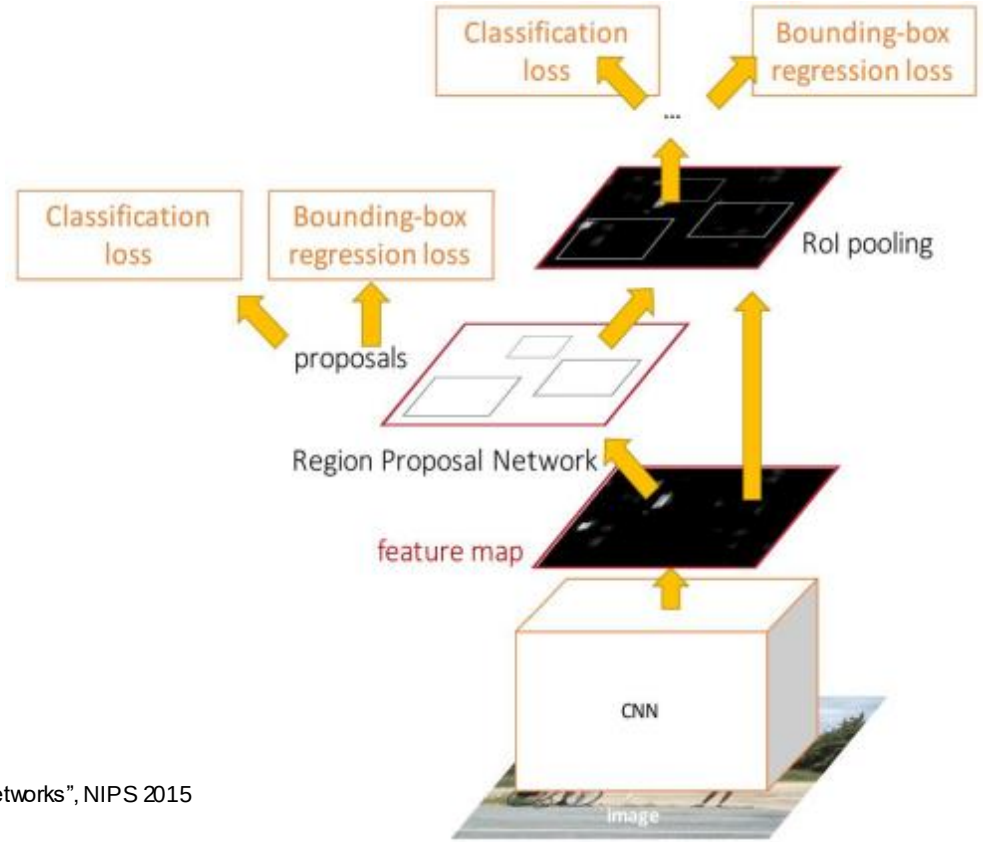
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN

Make CNN do proposals!

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

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



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

Region Proposal Network

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

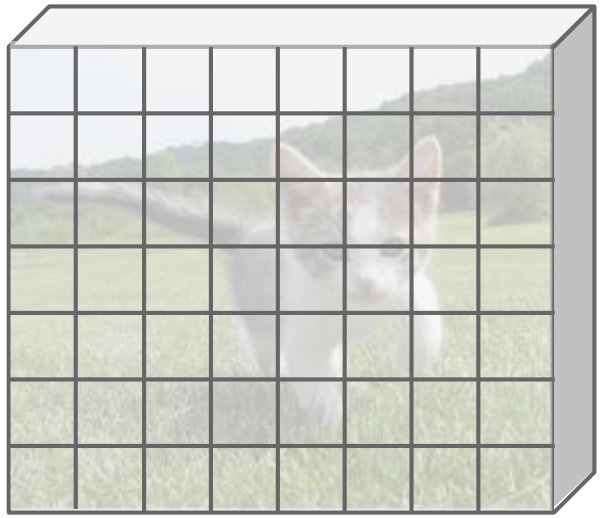


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

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

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

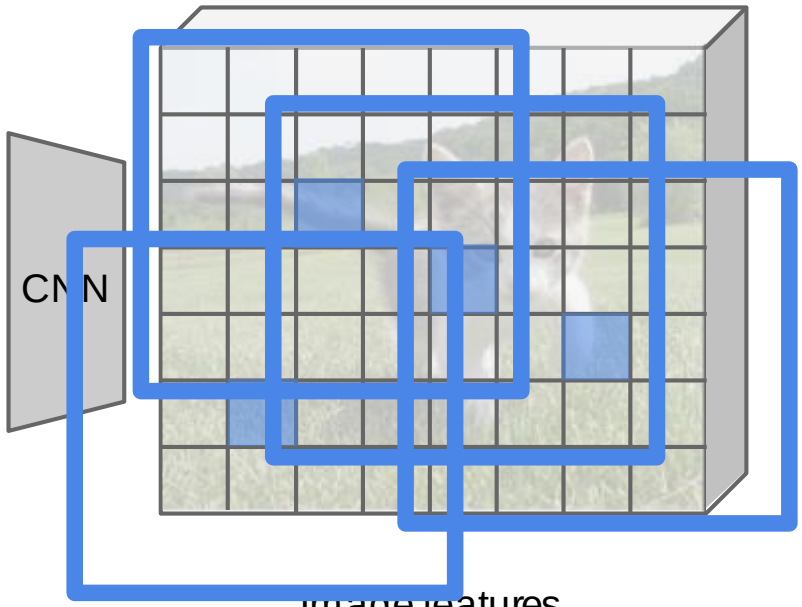


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

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

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

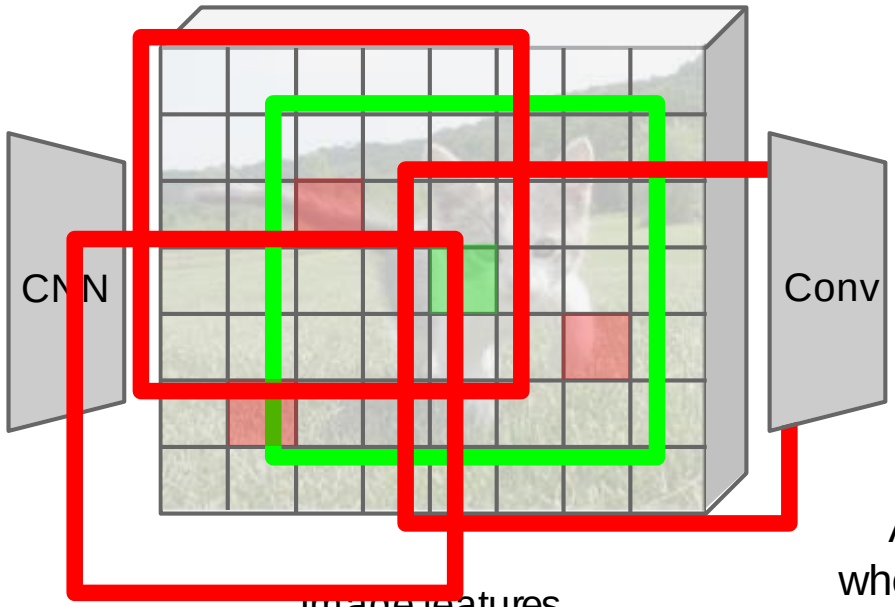


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network

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

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

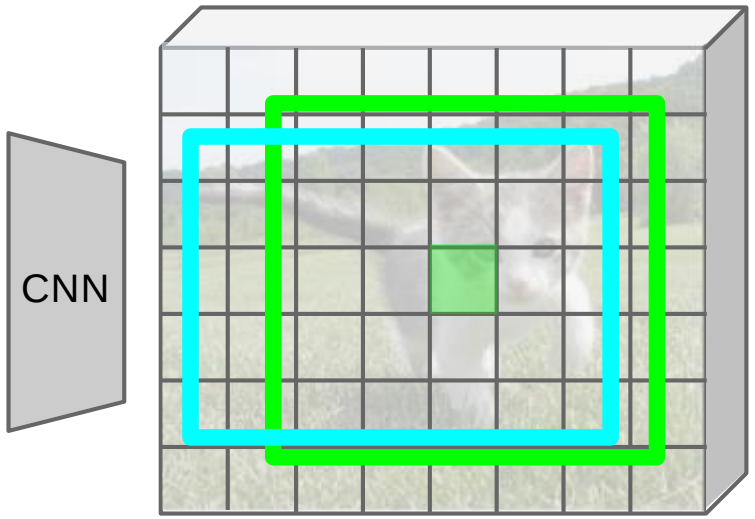


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

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

Region Proposal Network

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

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

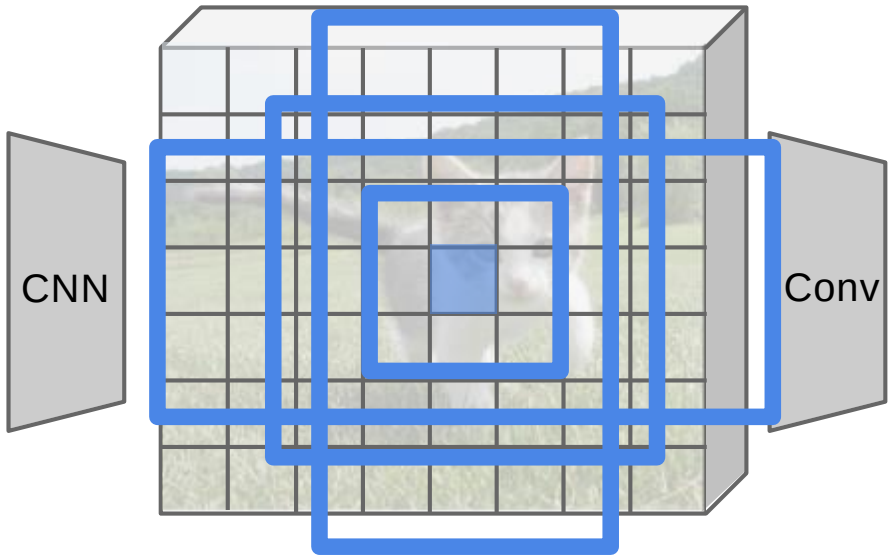


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

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

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

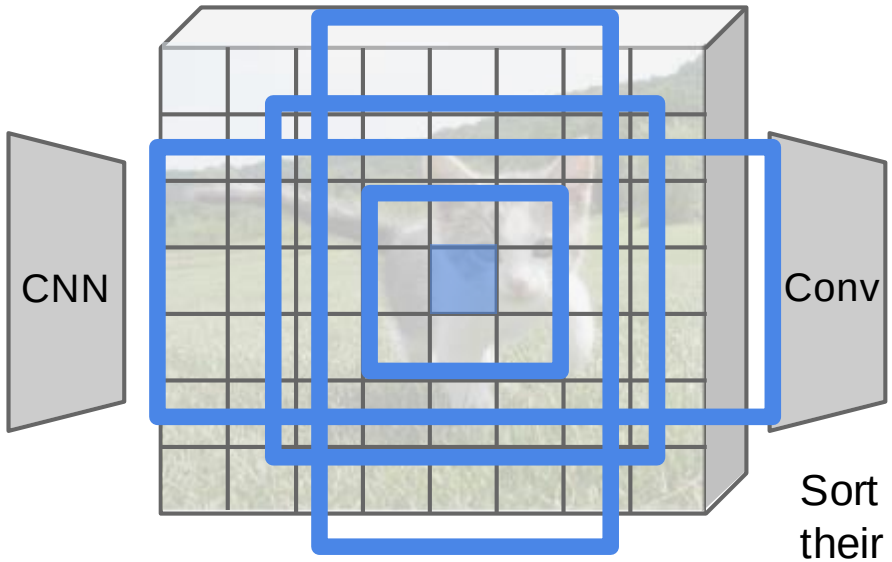
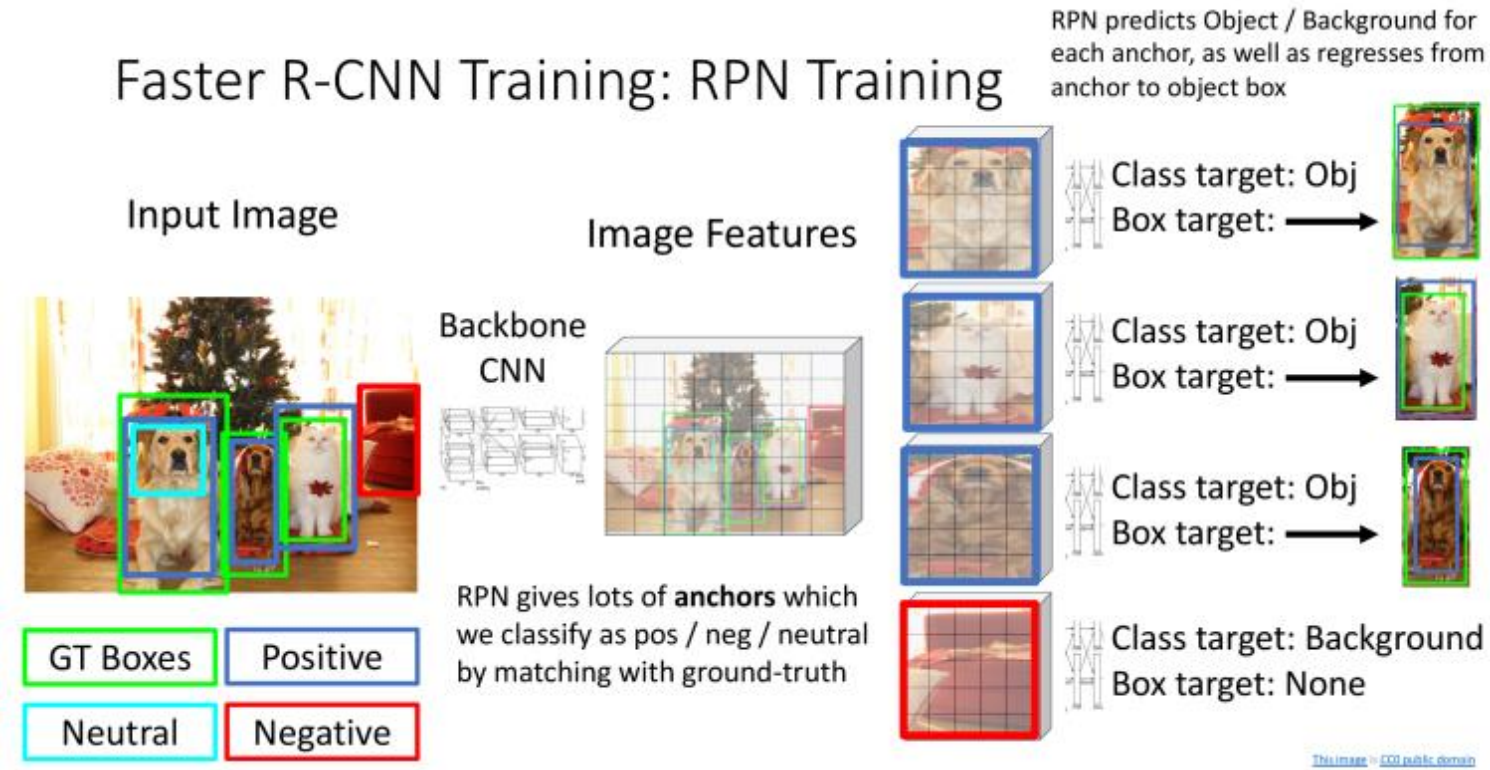


Image features
(e.g. 512 x 20 x 15)

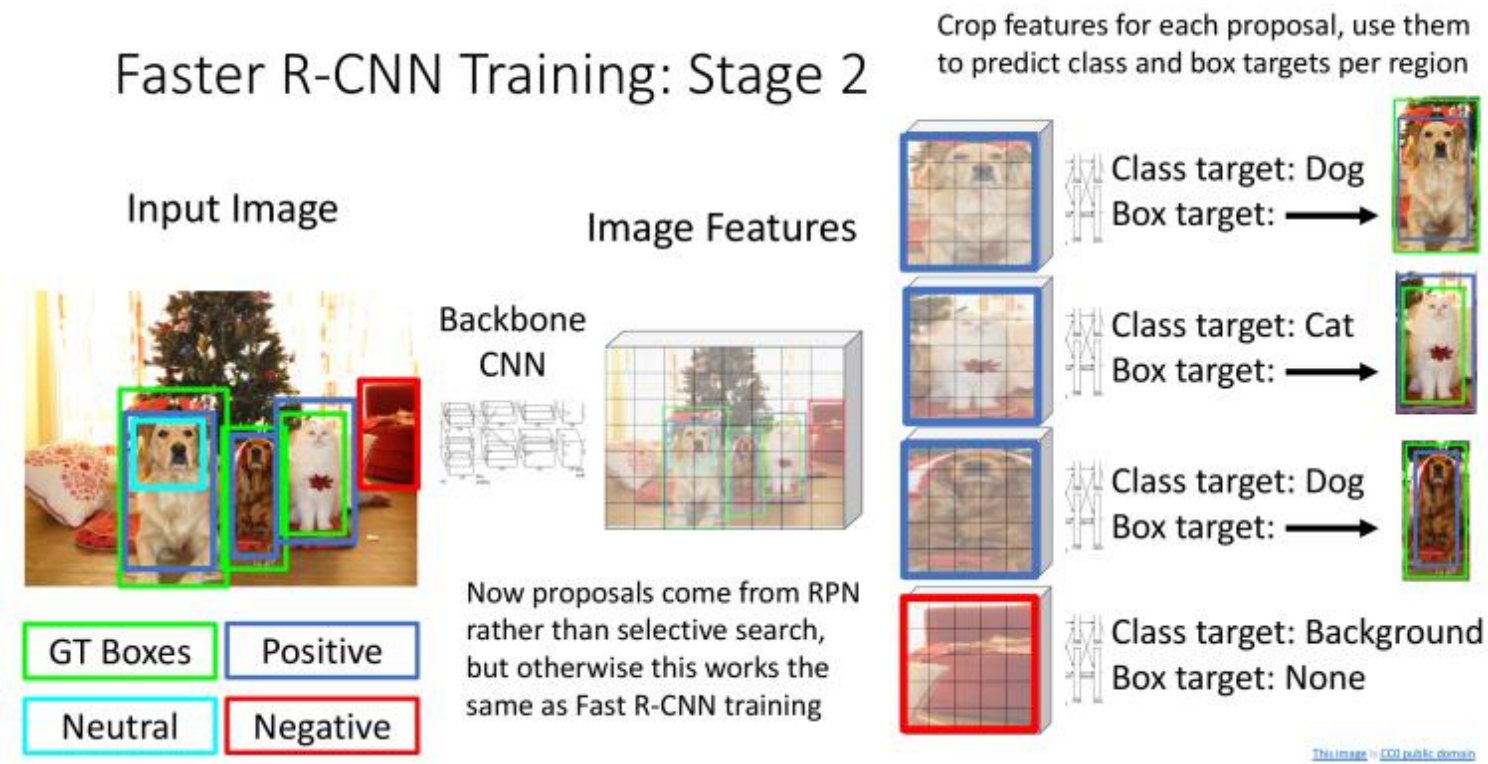
Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

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


Faster R-CNN Training: RPN Training
RPN predicts Object / Background for each anchor, as well as regresses from anchor to object box
Input Image
Image Features
Backbone CNN
Class target: Obj
Box target:
Class target: Obj
Box target:
Class target: Obj
Box target:
RPN gives lots of anchors which we classify as pos / neg / neutral by matching with ground-truth
Class target: Background
Box target: None
GT Boxes
Positive
Neutral
Negative

Faster R-CNN

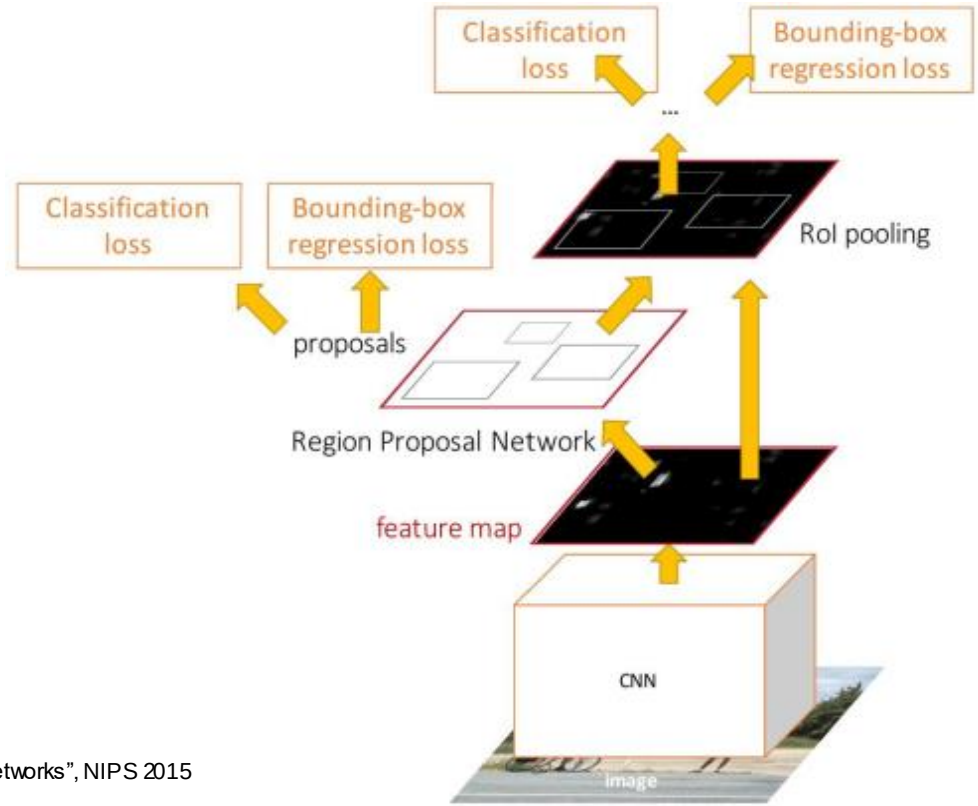

Faster R-CNN Training: Stage 2
Input Image
Backbone CNN
Image Features
GT Boxes
Positive
Neutral
Negative
Now proposals come from RPN rather than selective search, but otherwise this works the same as Fast R-CNN training
Crop features for each proposal, use them to predict class and box targets per region
Class target: Dog
Box target:
Class target: Cat
Box target:
Class target: Dog
Box target:
Class target: Background
Box target: None

Faster R-CNN

Make CNN do proposals!

Jointly train with 4 losses:

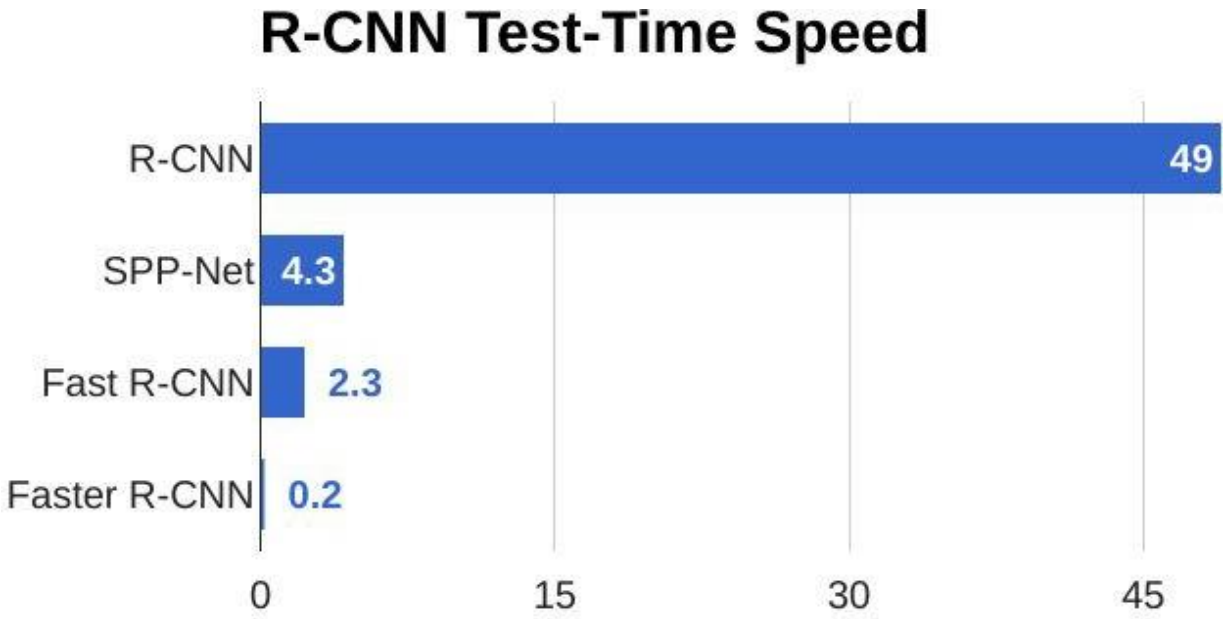
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



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

Faster R-CNN

Make CNN do proposals!



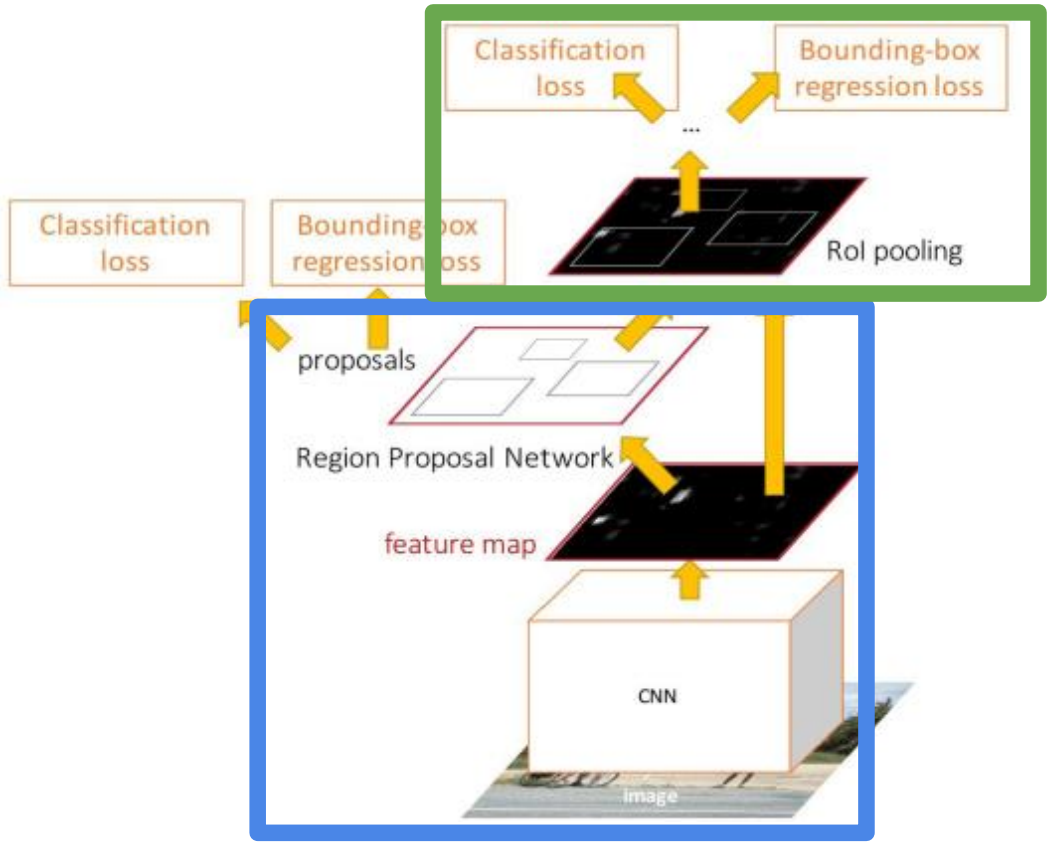
Faster R-CNN

Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

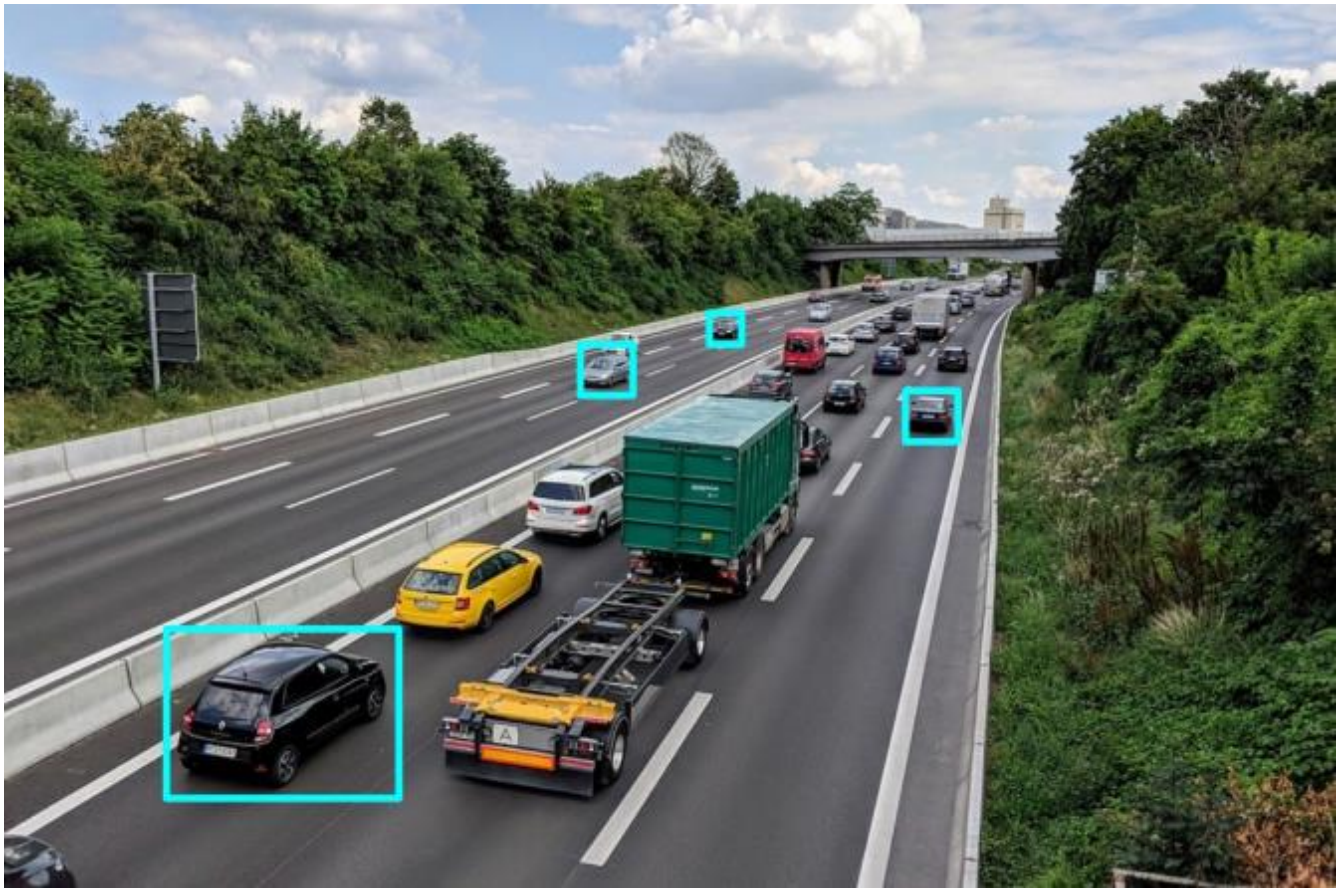
First stage: Run once per image
- Backbone network
- Region proposal network

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



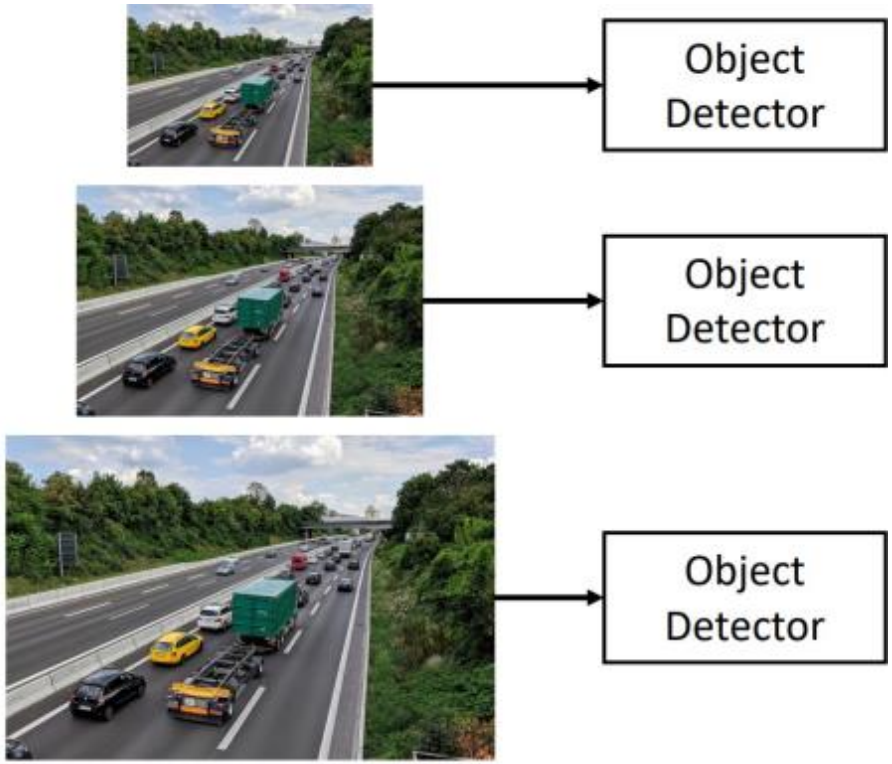
Dealing with Scale

- 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.



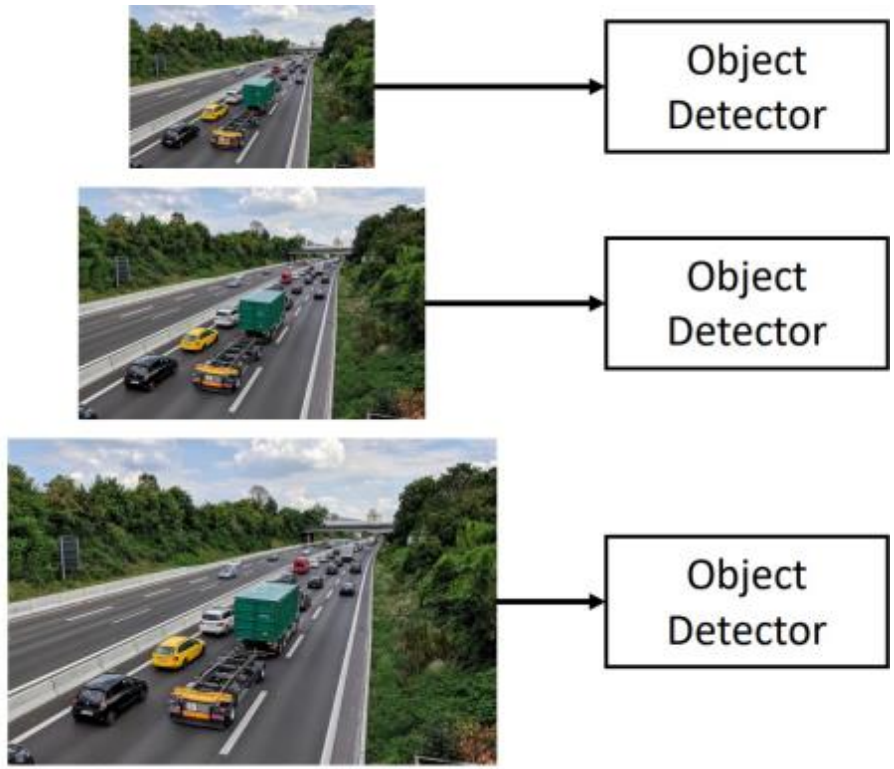
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

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

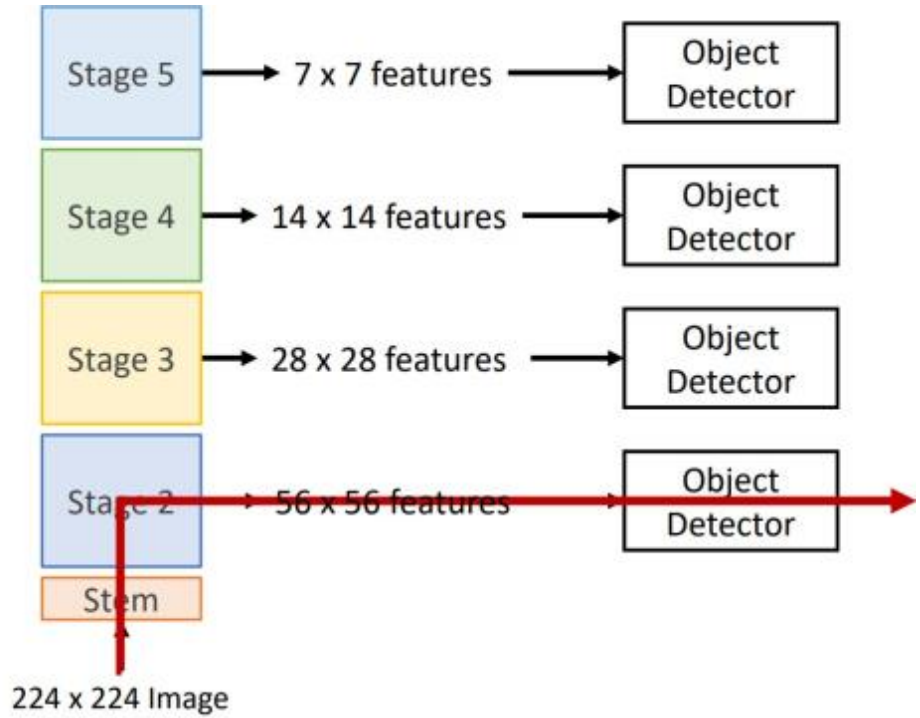
Problem: Expensive! Don't share any computation between scales

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



Dealing with Scale: Image Pyramid

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



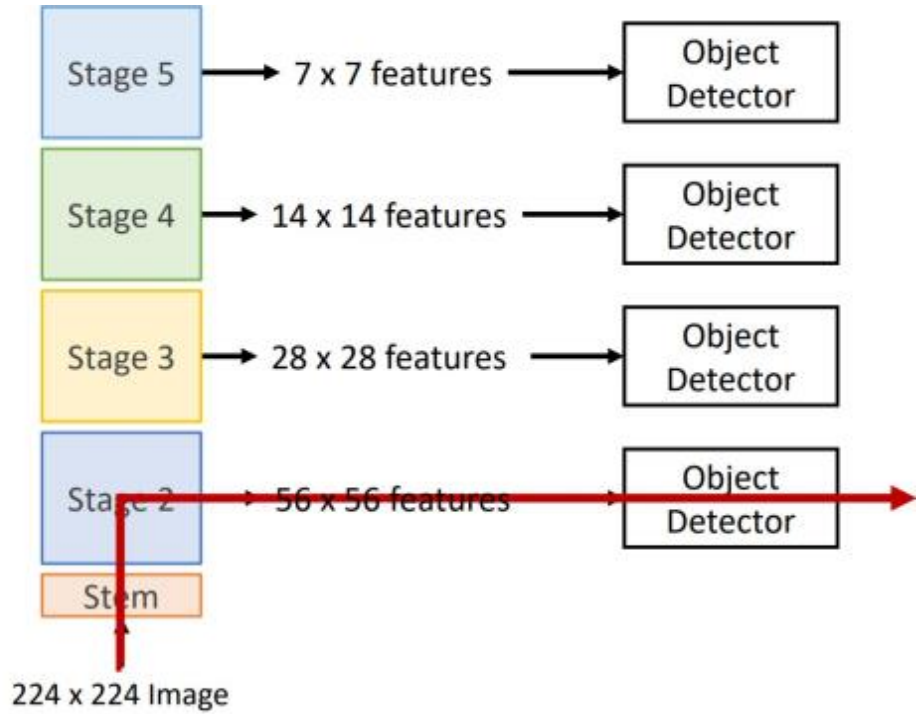
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

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

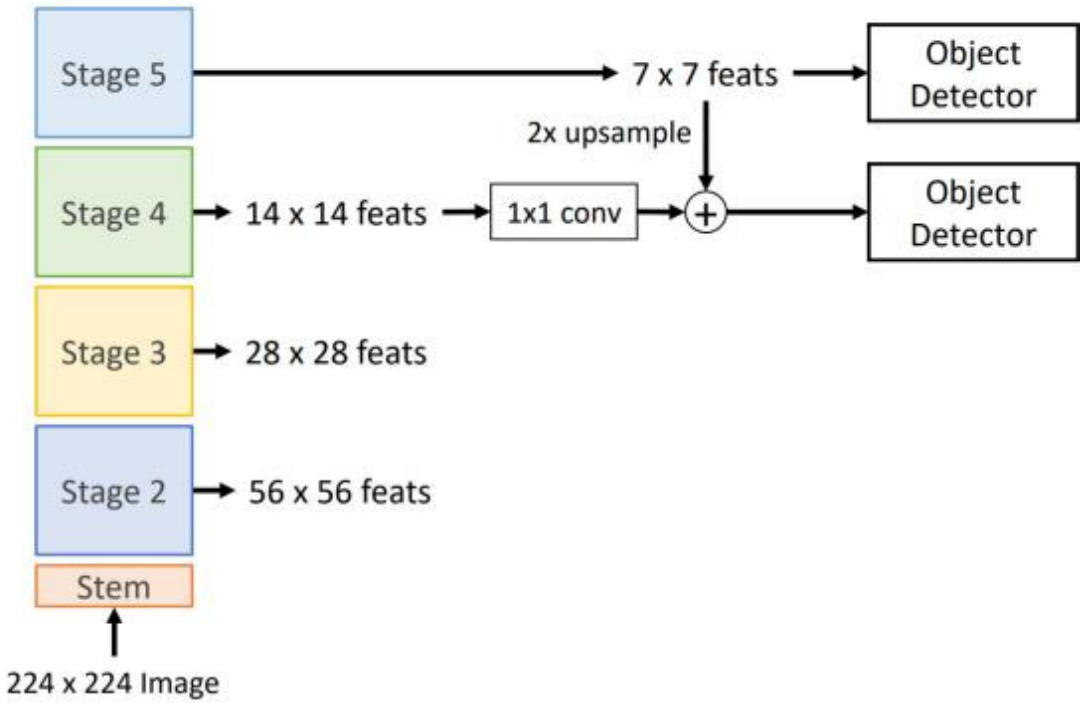
Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

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



Dealing with Scale: Feature Pyramid Network

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



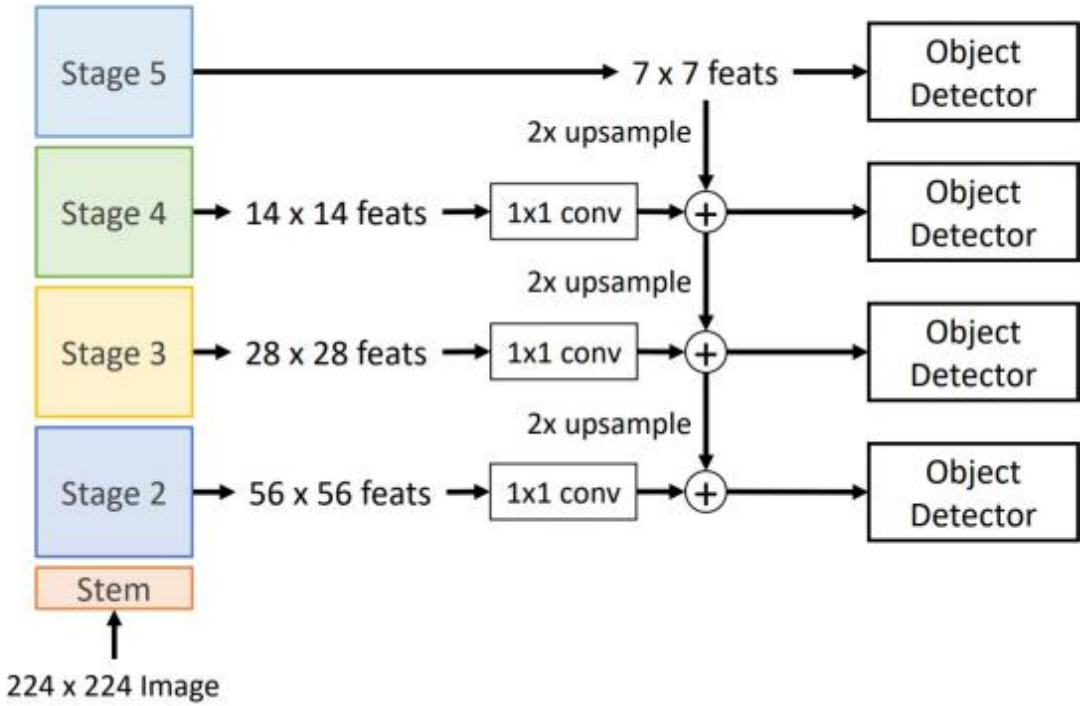
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network (cont.)

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

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

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

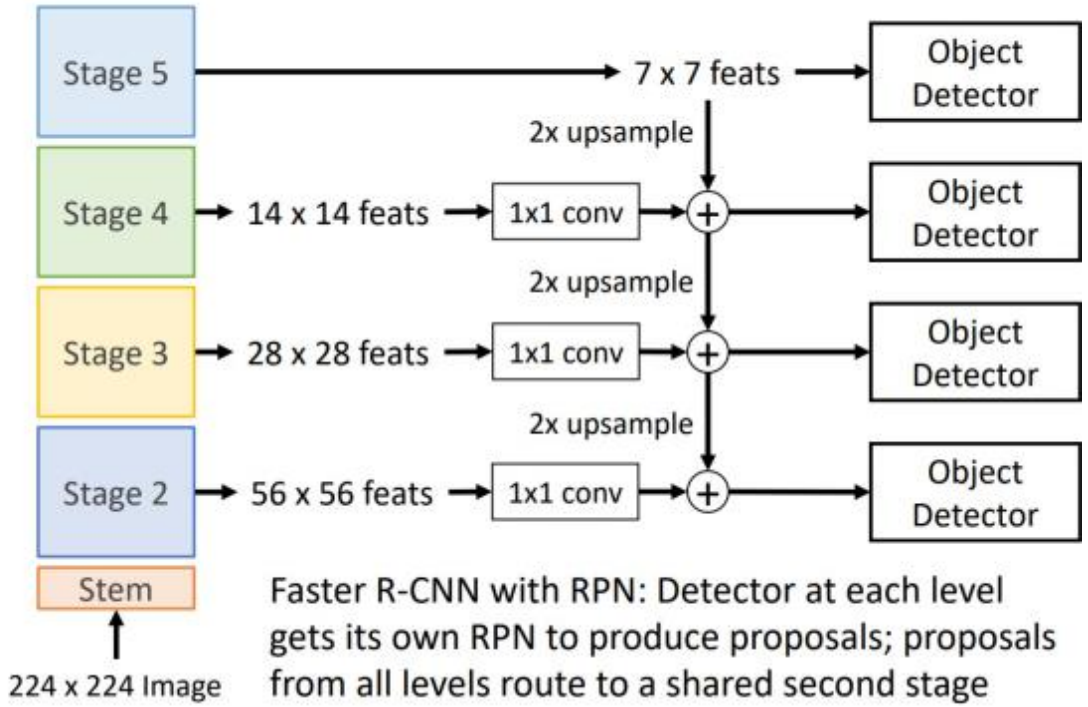


Dealing with Scale: Feature Pyramid Network (cont.)

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

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

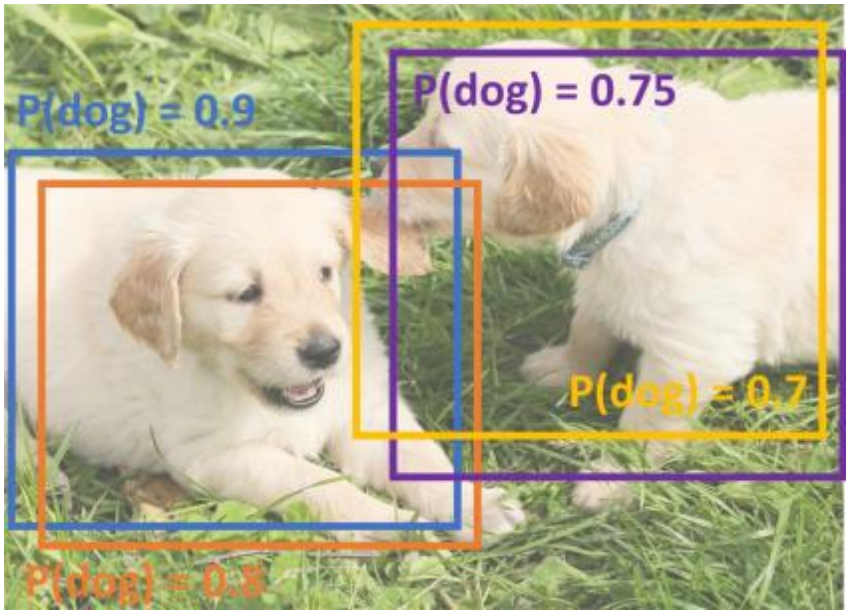
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



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

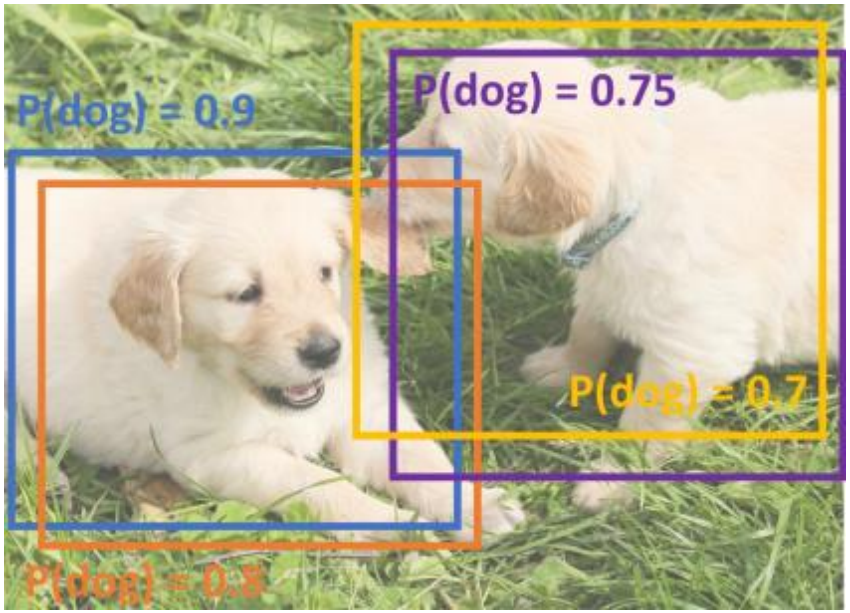
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections



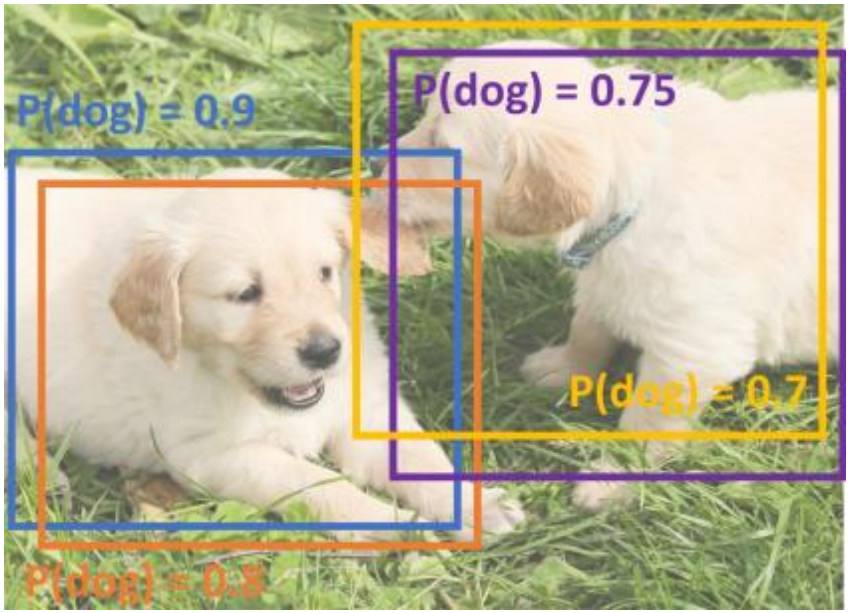
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)



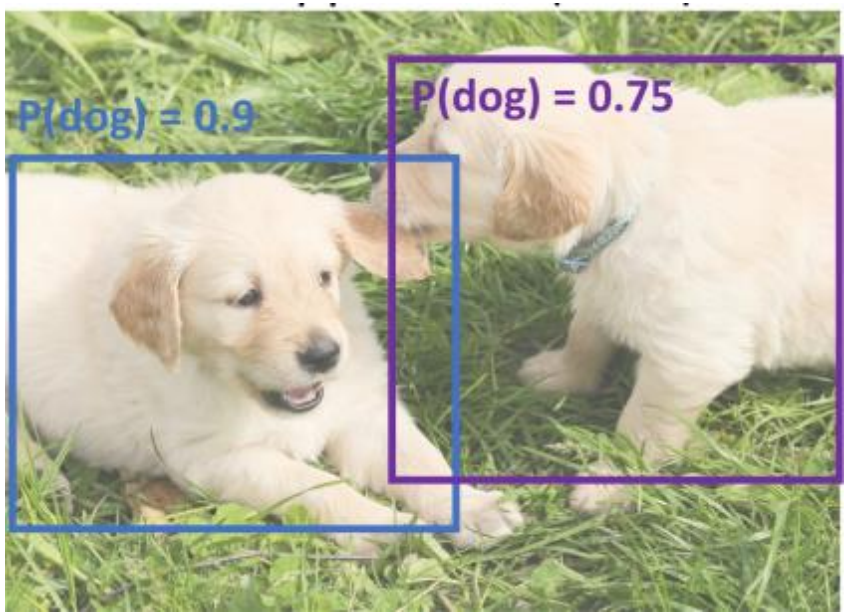
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU > threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

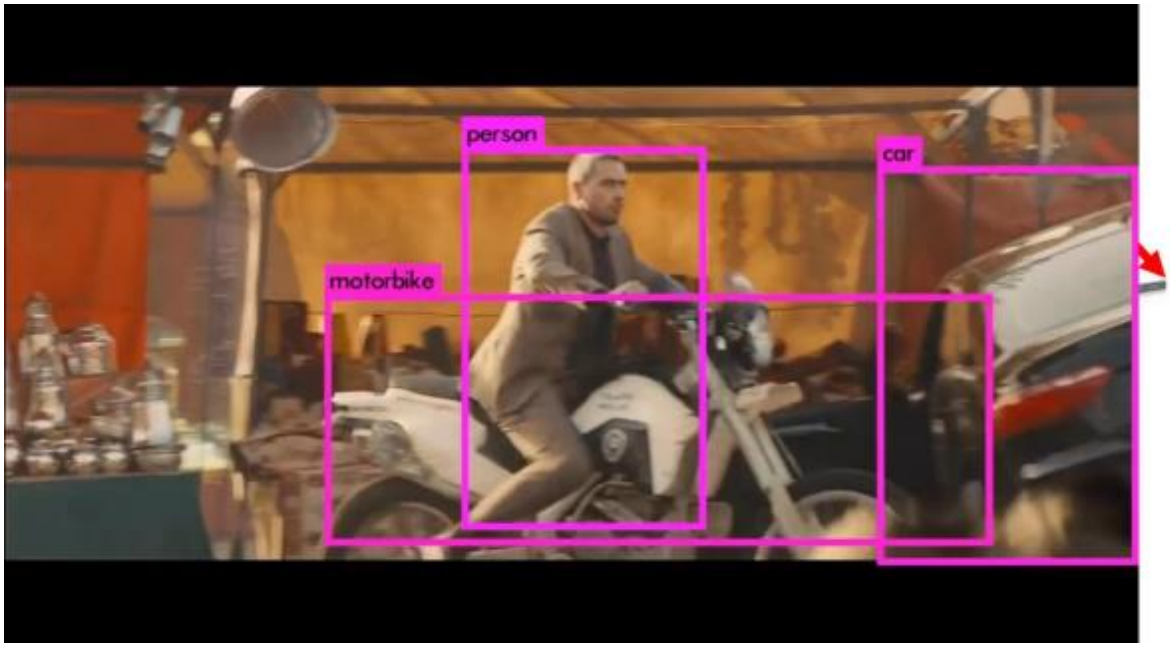
- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU > threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU > threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1
- **Problem:** NMS may eliminate "good" boxes when objects are highly overlapping. . . no good solution =(

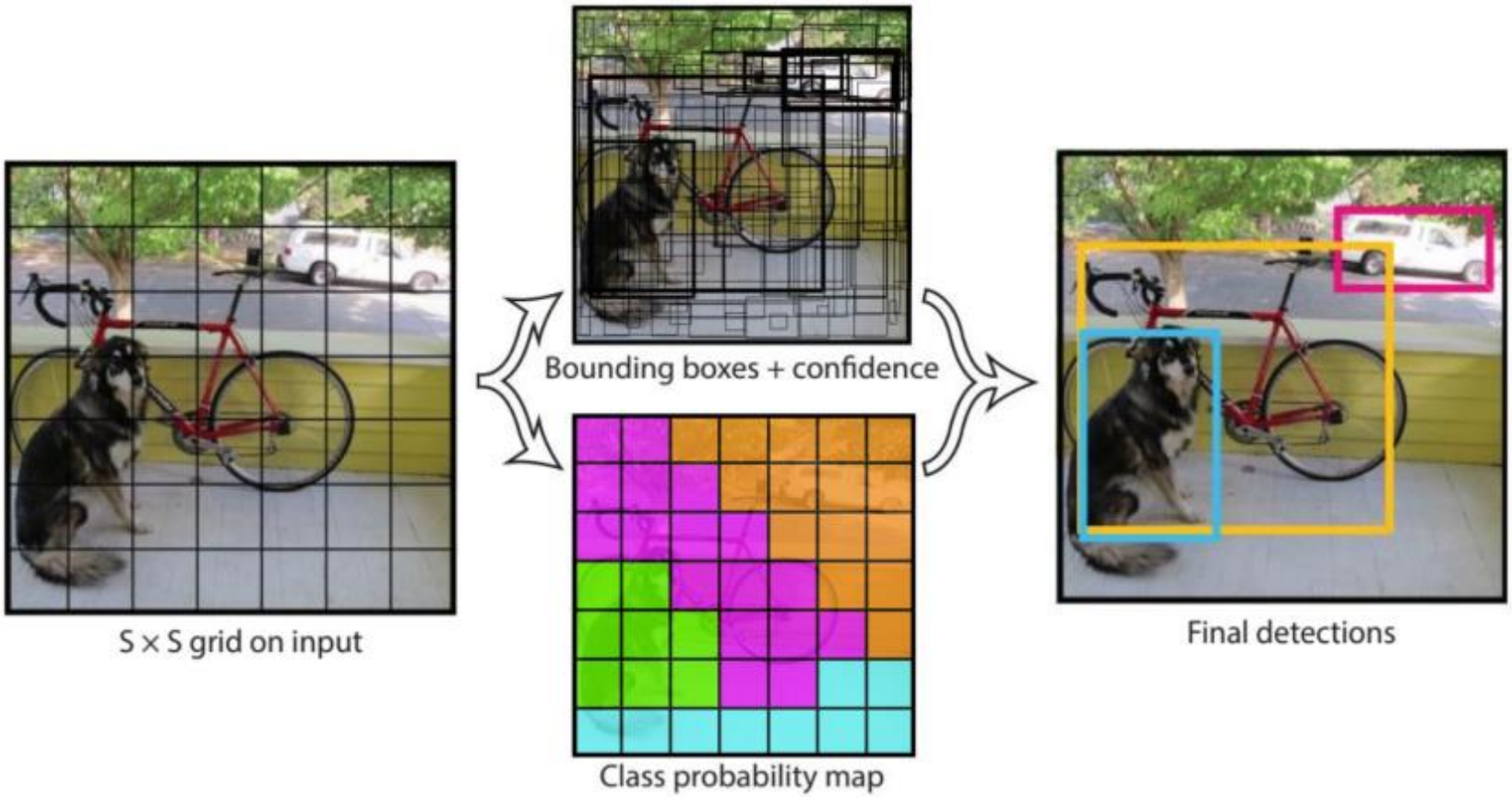
Single Shot Object Detection



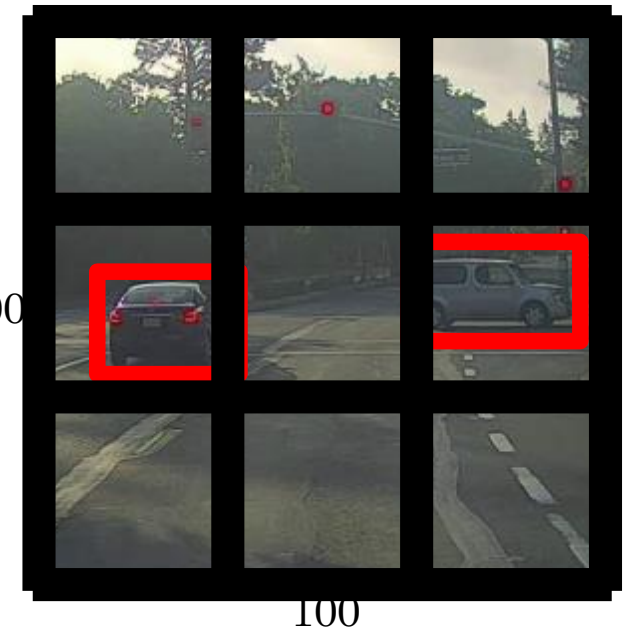
Single Shot:
SSD, YOLO ...

Fast
High false rate

YOLO - Overview



YOLO algorithm

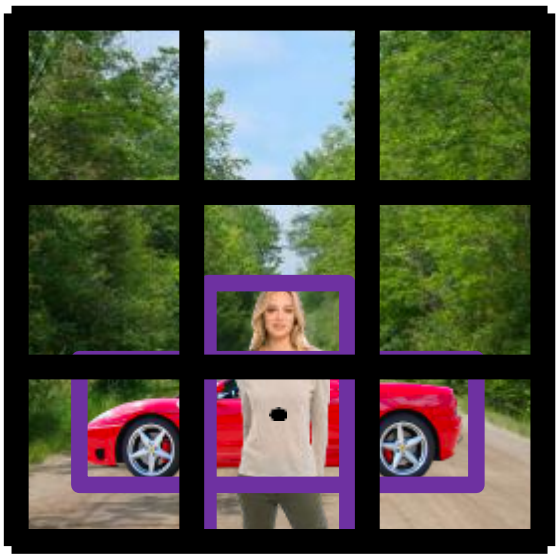


Labels for training

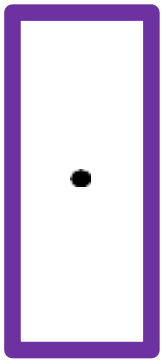
For each grid cell:

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

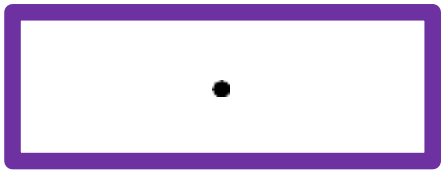
YOLO algorithm



Anchor box 1:



Anchor box 2:

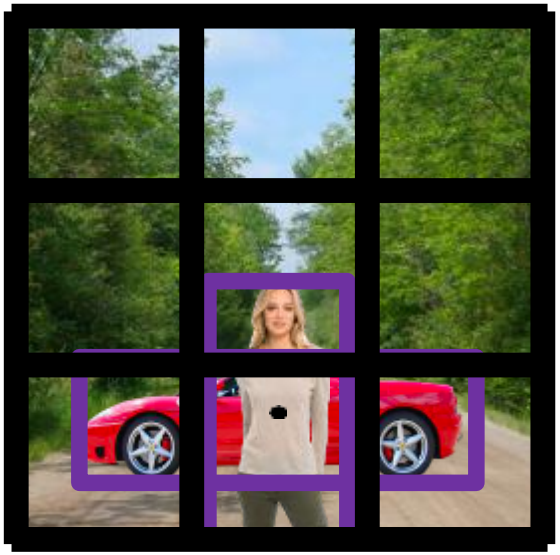


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

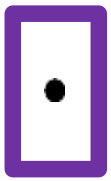
[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

YOLO algorithm



Anchor box 1:

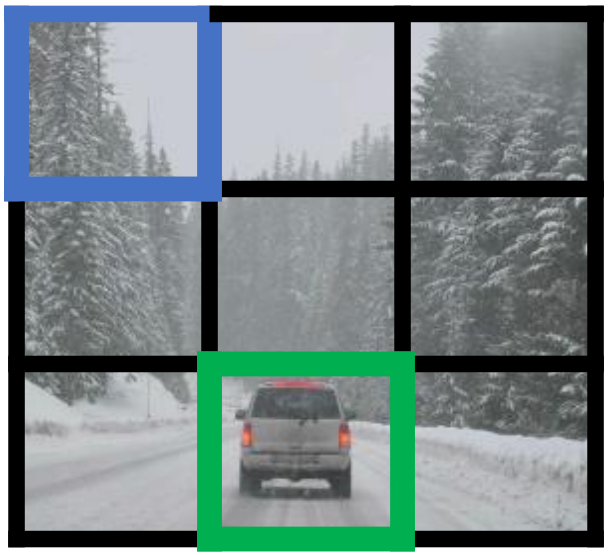


Anchor box 2:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Andrew Ng

YOLO algorithm



1 - pedestrian
2 - car
3 - motorcycle

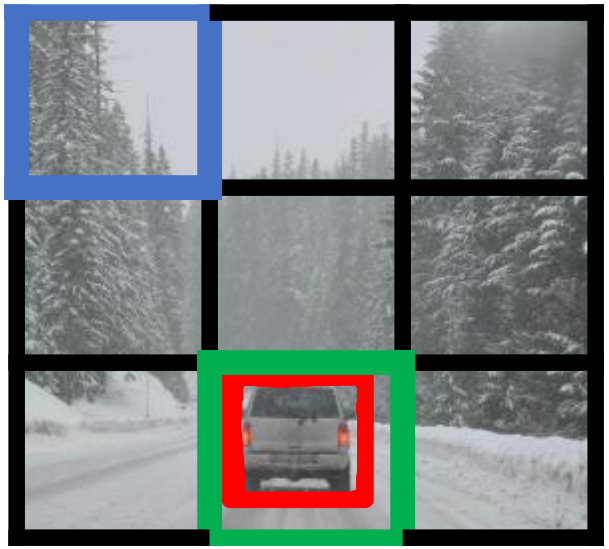
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

y is $3 \times 3 \times 2 \times 8$

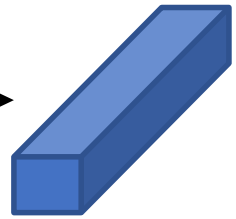
[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

YOLO algorithm



$$\longrightarrow \quad \cdots \quad \longrightarrow$$

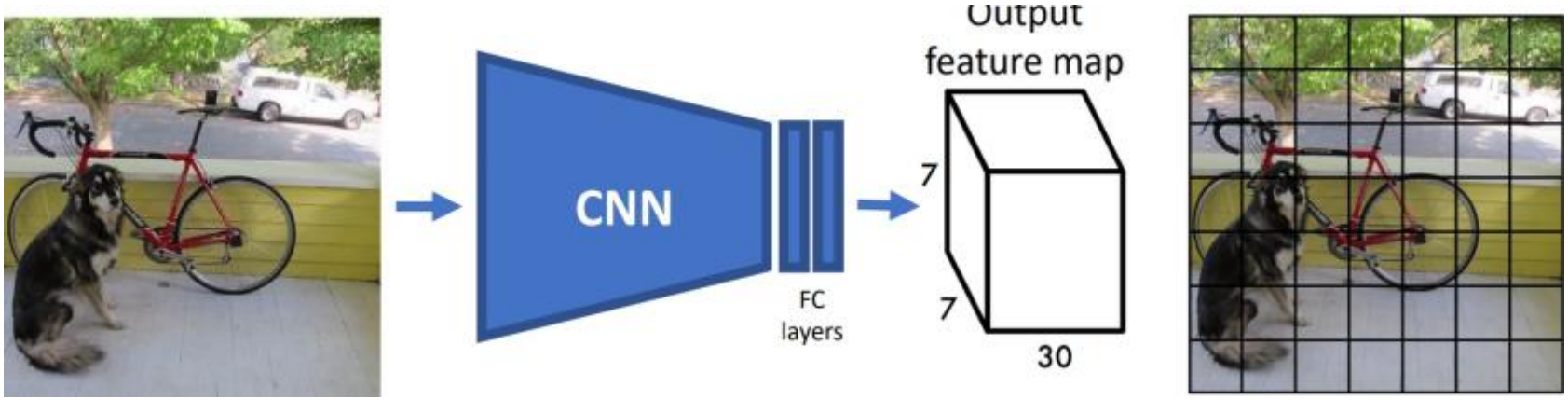


$3\times3\times2\times8$

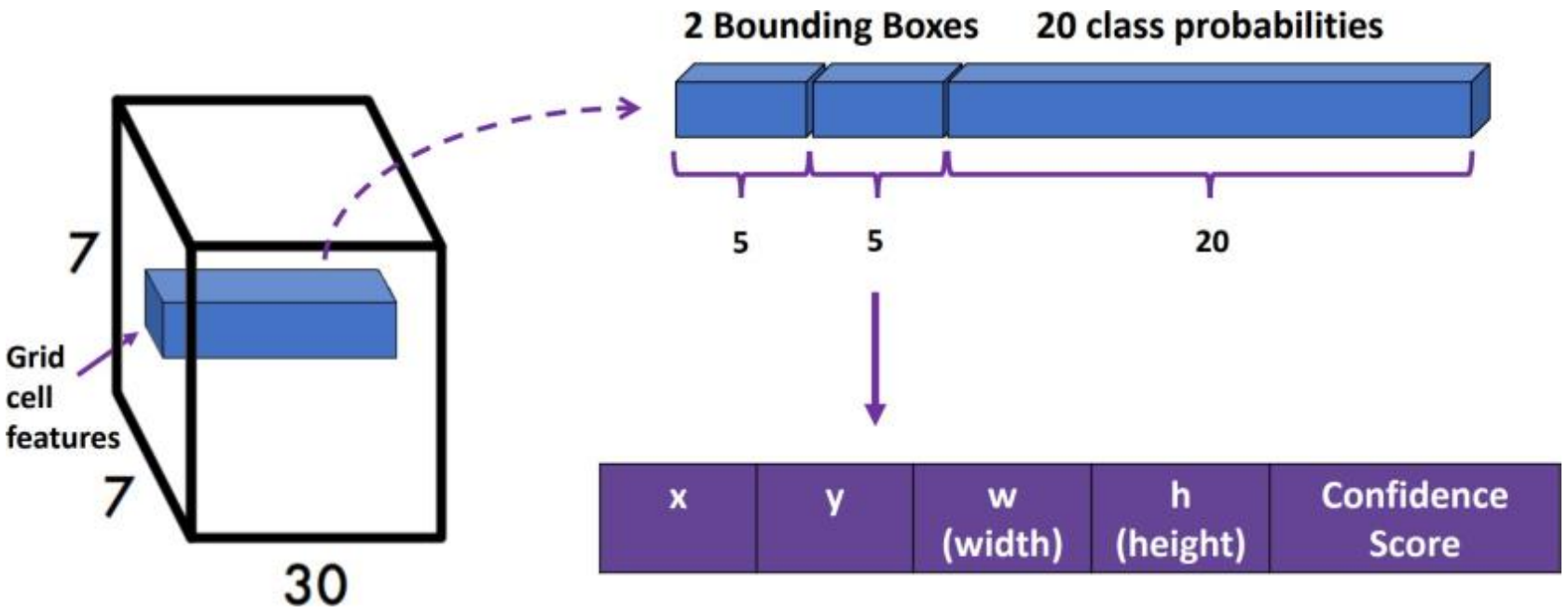
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

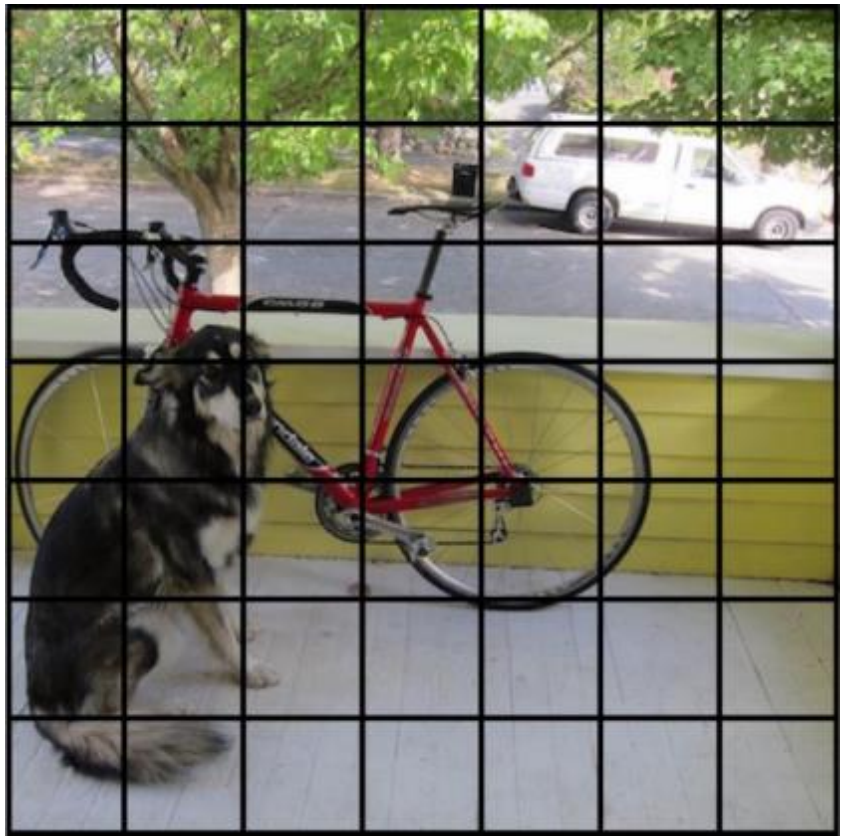
Andrew Ng

YOLO

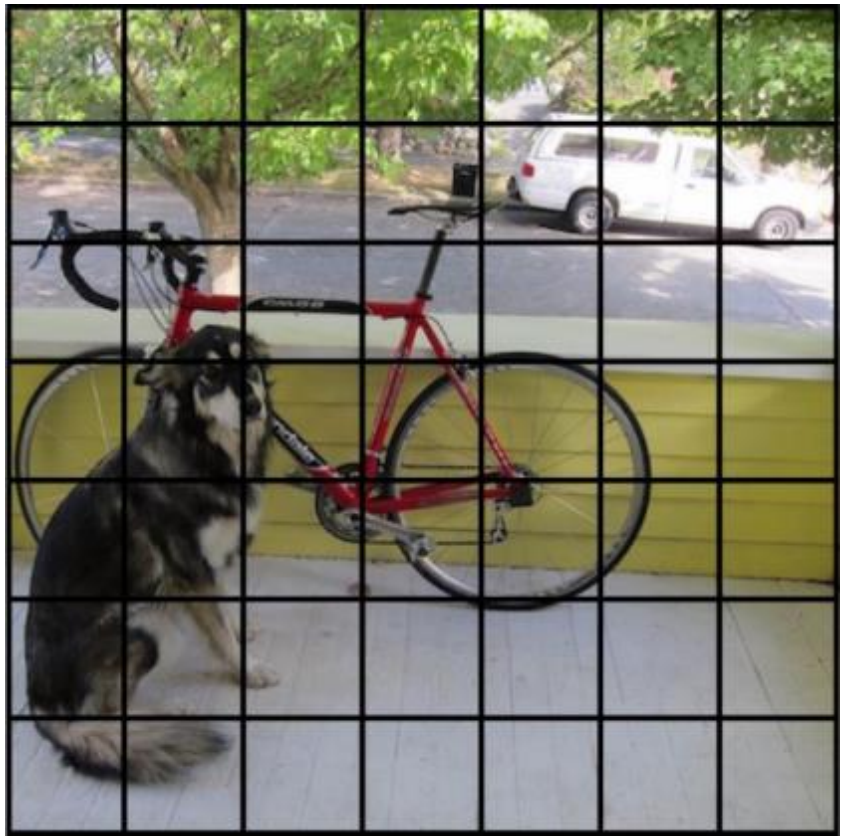


YOLO


2 Bounding Boxes
20 class probabilities
5
5
20
7
Grid cell features
7
30
x
y
w (width)
h (height)
Confidence Score

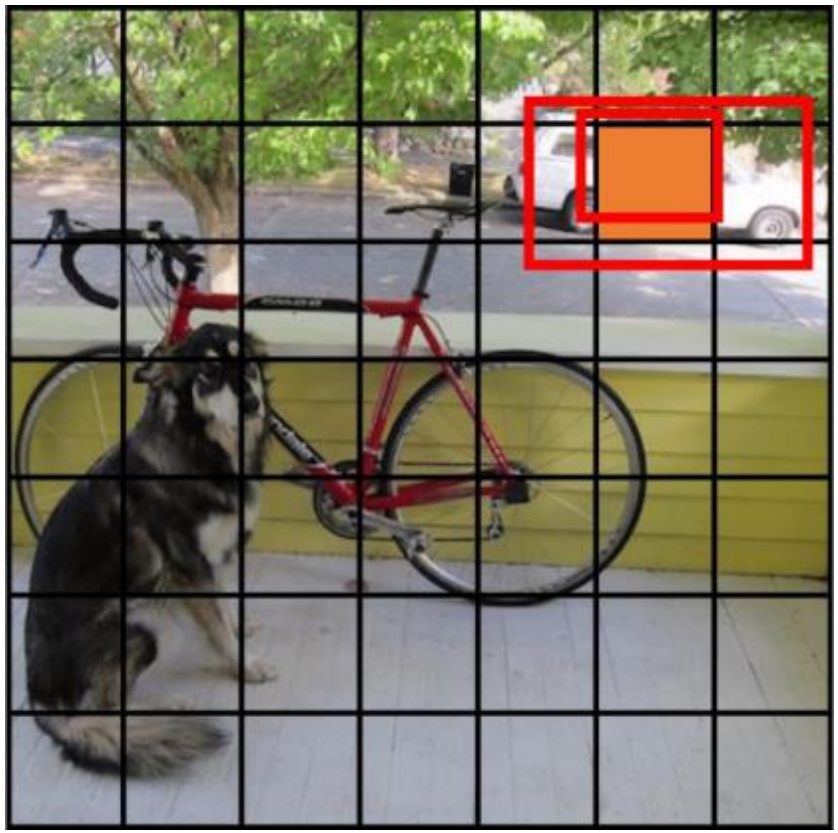


Each cell predicts



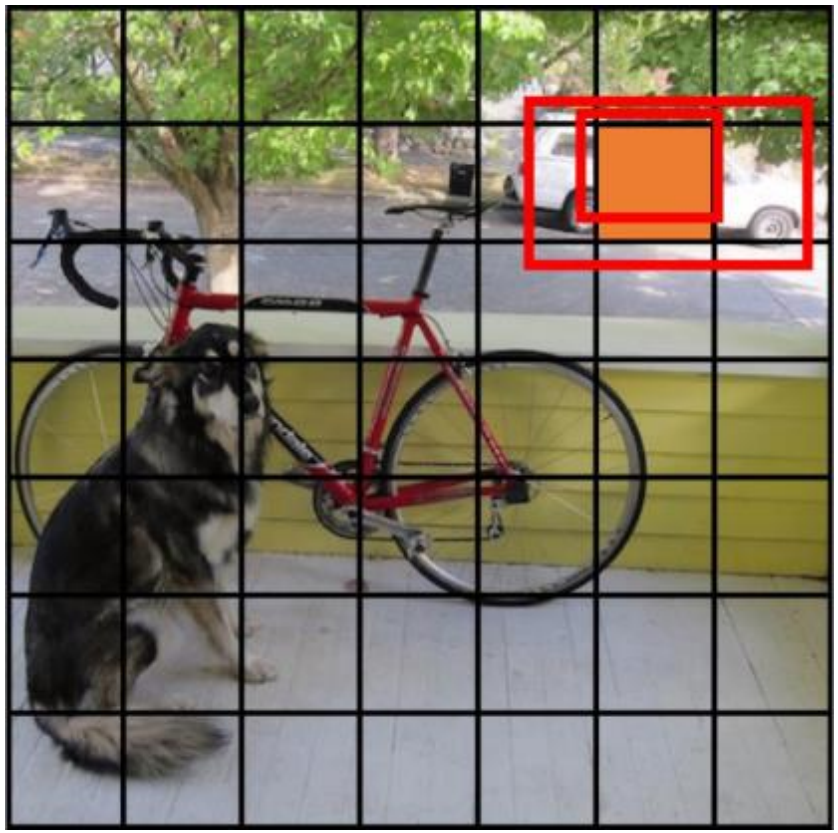
Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score



Each cell predicts

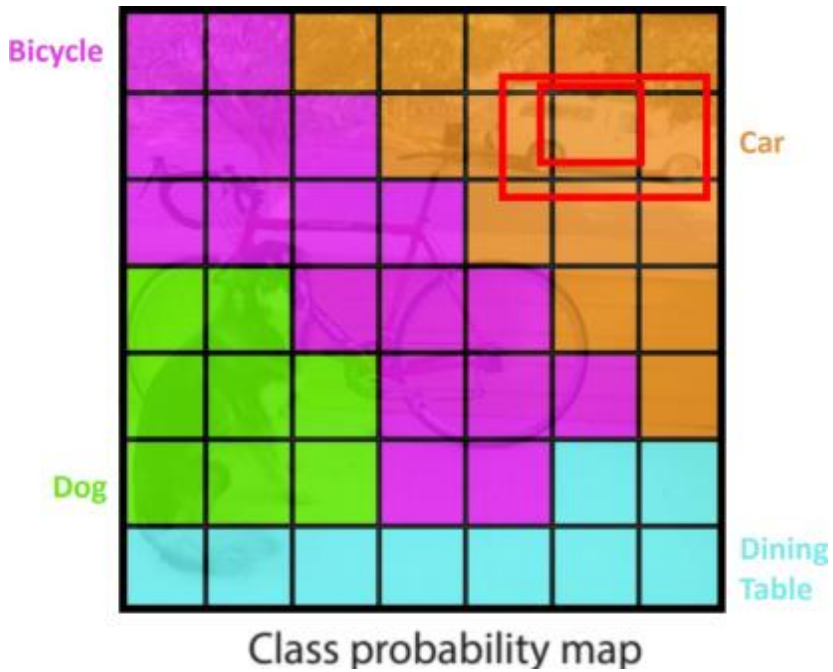
- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities



YOLO

Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities



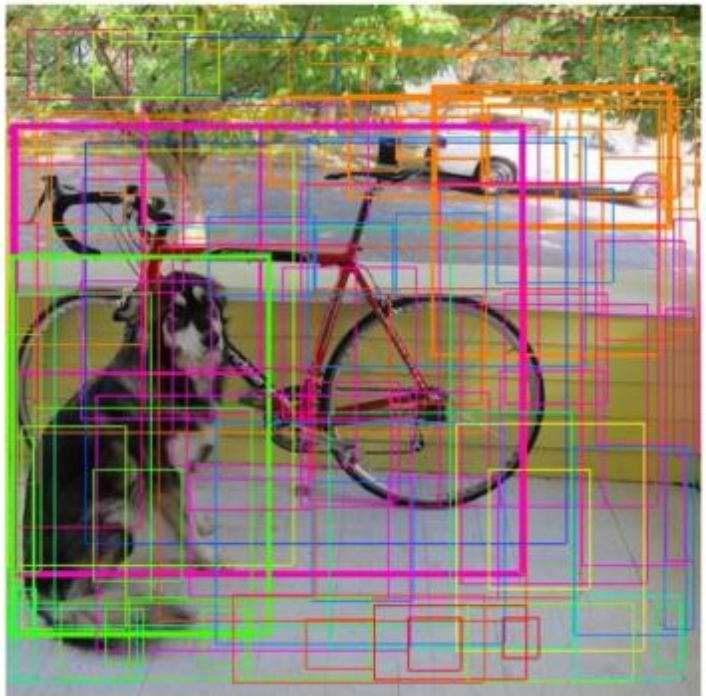
Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

SxSxB Bounding-Boxes (S=7,B=2 → 96 Bboxs)

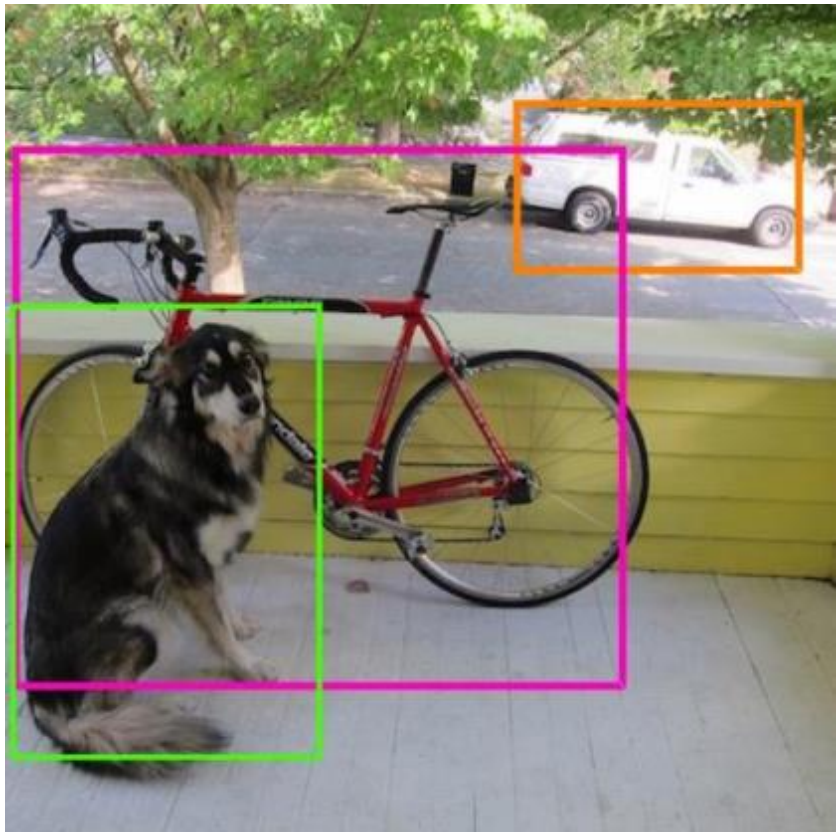


YOLO

Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

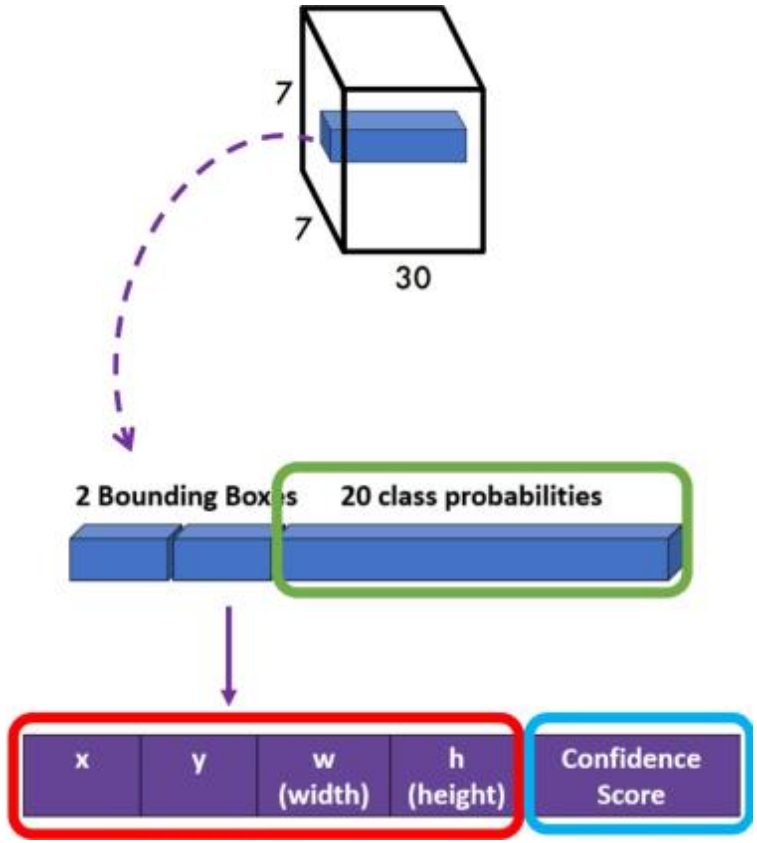
- Apply Non-Maximum Suppression



YOLO - Loss Function

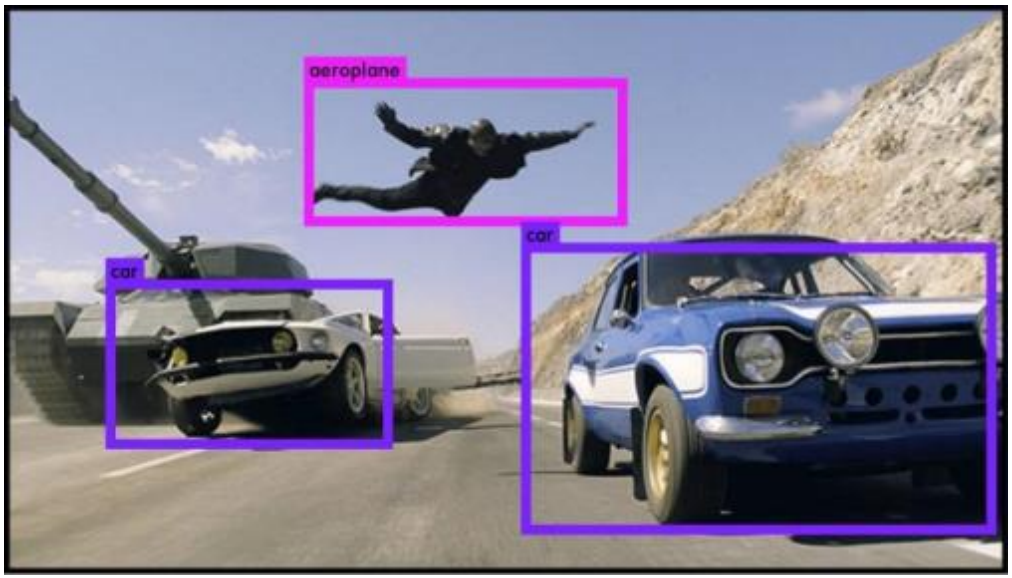
YOLO – Loss function

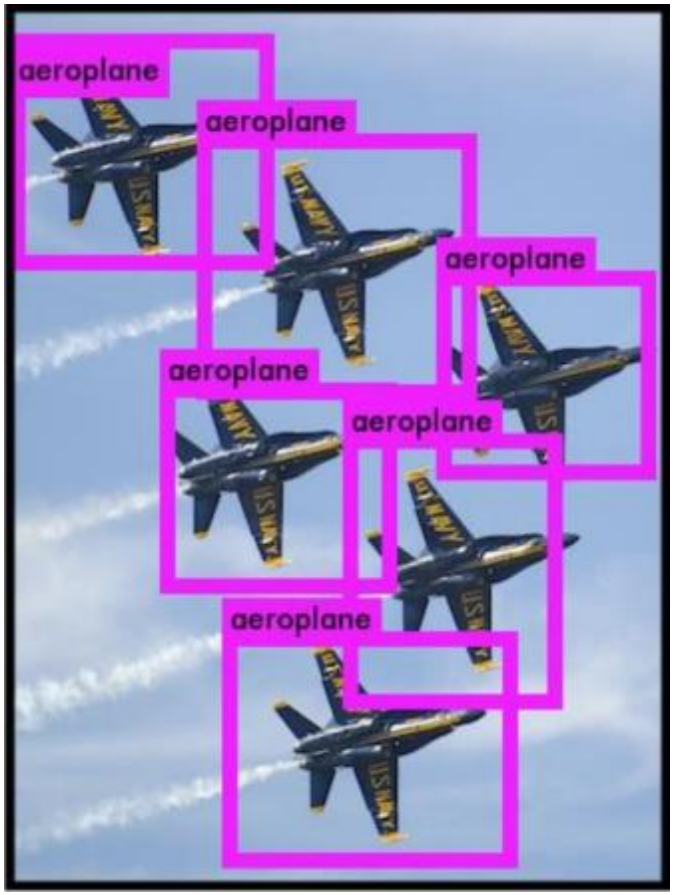
$$\mathcal{L} = \mathcal{L}_{Localization\ Loss} + \mathcal{L}_{Confidence\ Loss} + \mathcal{L}_{Classification\ Loss}$$



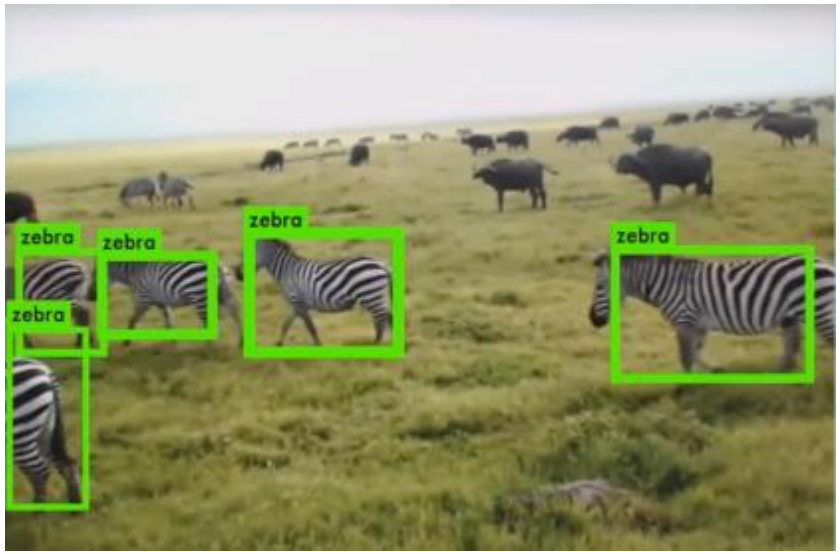
YOLO - Benefits

- Fast. Good for real-time processing
- End-to-end training



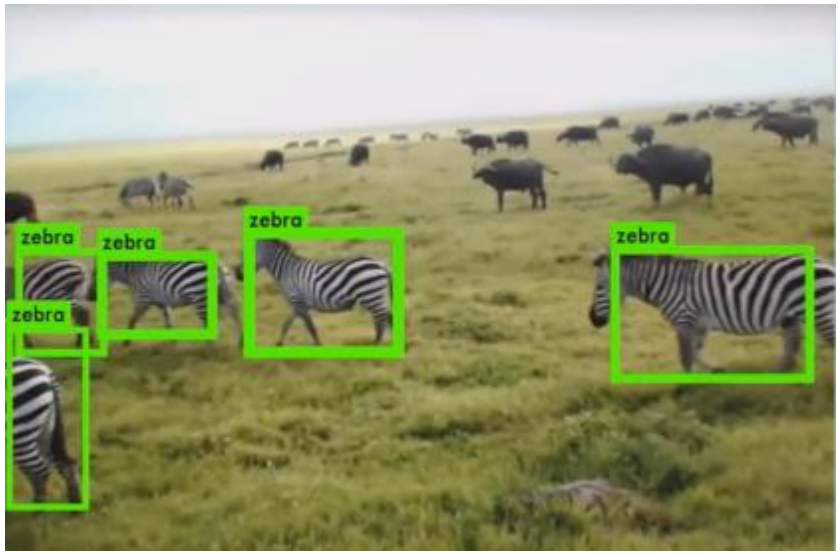


YOLO - Limitations



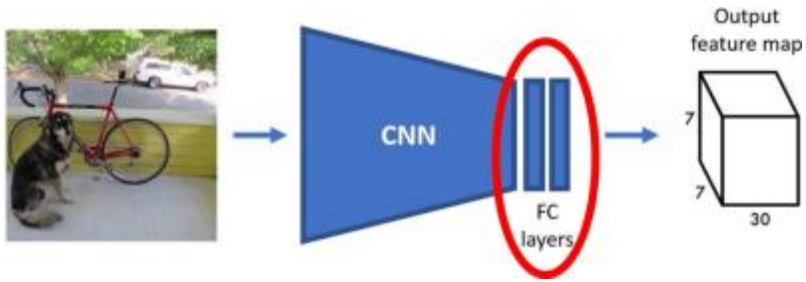
YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions



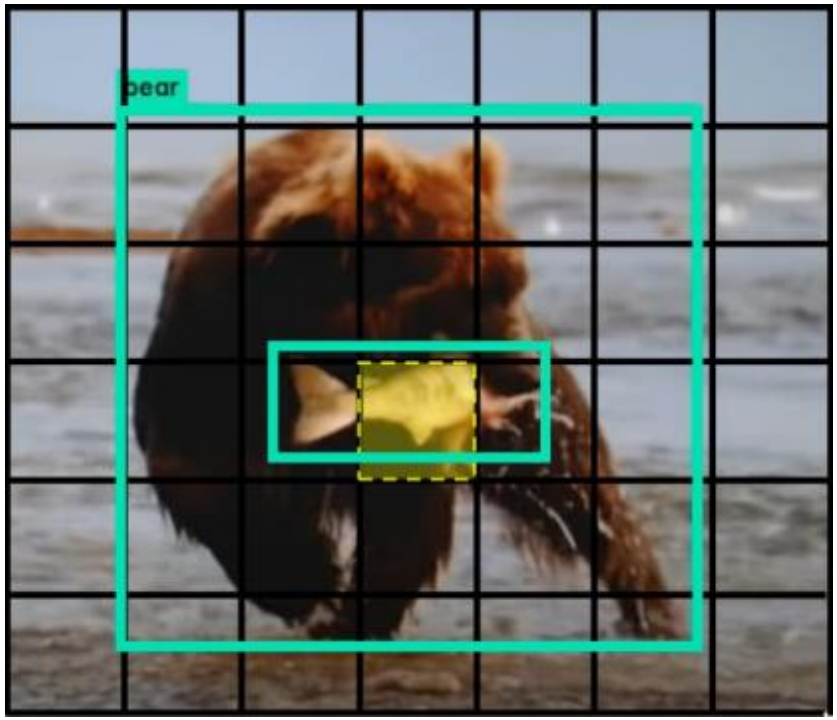
YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions
- Fixed input size



YOLO - Limitations

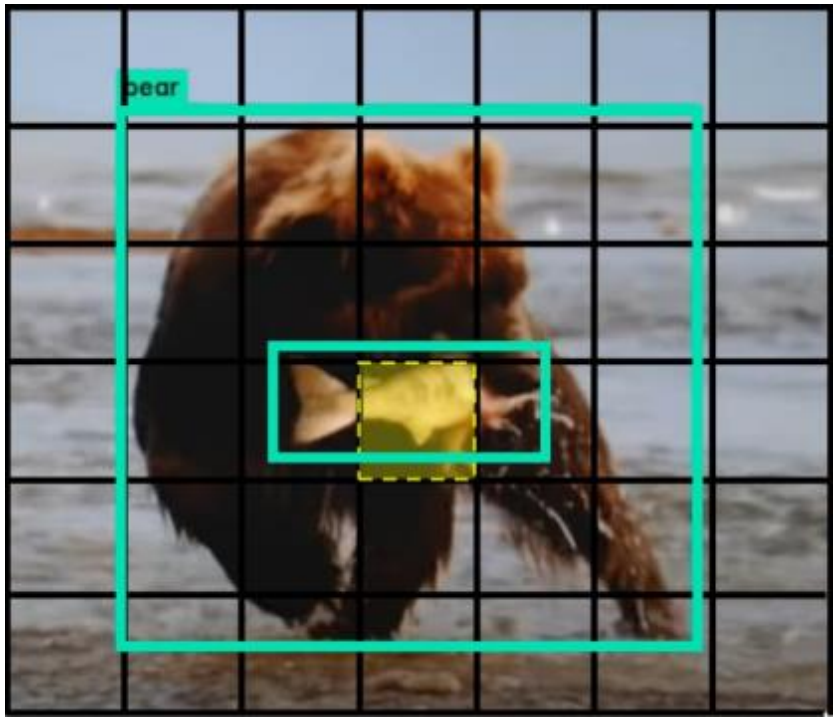
- Difficult to detect small objects
- Coarse predictions
- Fixed input size
- A grid cell can predict only one class



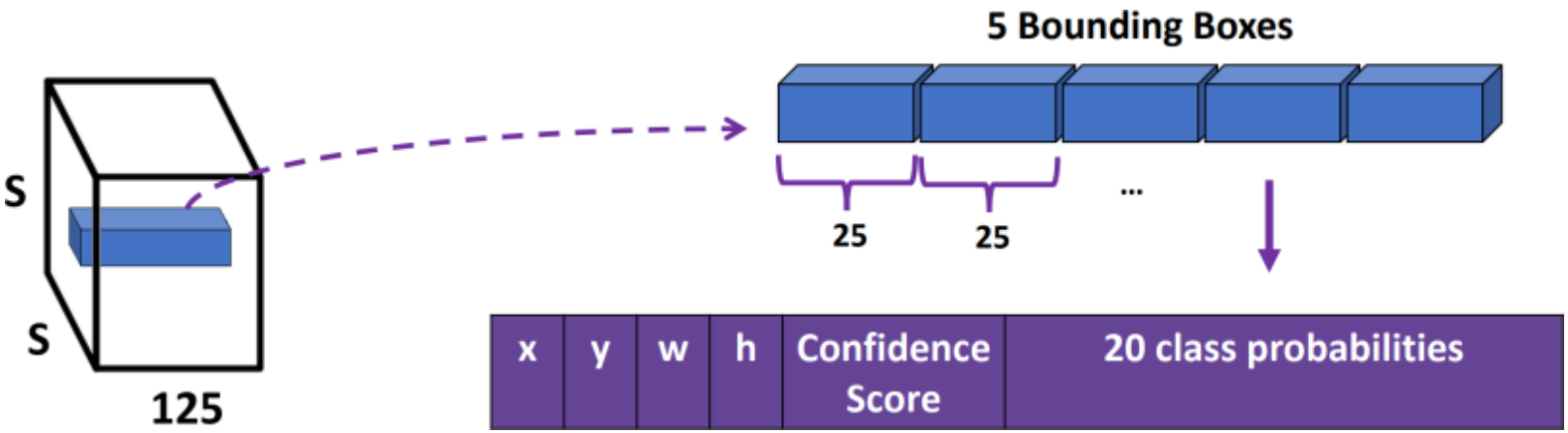
YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions
- Fixed input size
- A grid cell can predict only one class

- Solutions:
  - Remove fc layers!
  - Predict class per bbox (not per cell)



- Removed fully connected layers
- A grid cell predicts class probabilities for each box



There's always room for improvement!

- YOLOv3
  - J. Redmon, A. Farhadi. Yolov3: An incremental improvement, 2018
- YOLOv4
  - A. Bochkovskiy, C. Wang, H. Liao. Yolov4: Optimal speed and accuracy of object detection (Feb. 2020)
- YOLOv5
  - YOLOv5 by ultralytics (June 2020)
- PP-YOLO
  - X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding, S. Wen. Pp-yolo: An effective and efficient implementation of object detector (June 2020)
- PP-YOLOv2 (2021)
  - J. X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, X. Hu, D. Yu, Y. Ma, O. Yoshie. PP-YOLOv2: A Practical Object Detector (2021)

References

These slides have been adapted from

- Fei-Fei Li, Yunzhu Li & Ruohan Gao, Stanford CS231n: Deep Learning for Computer Vision
- Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
- Justin Johnson, UMich EECS 498.008/598.008: Deep Learning for Computer Vision
- cs231n@stanford
- EECS 498.008 / 598.008
- CNN DeepLearning.AI