ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 12: Object Detection - part 2

02/19/2025

https://deeprob.org/w25/

ROBOTICS

Today

- Feedback and Recap (5min)
- Object Detection (part 2)
  - Fast R-CNN (25min)
    - Receptive Fields
    - ROI Align
    - ROI Pool
  - Faster R-CNN (25min)
    - Region Proposal Network
  - Mask R-CNN (6min)
  - YOLO (6min)
  - Mesh R-CNN (6min)
- Summary and Takeaways (5min)

Recap: Object Detection

P3 released,
Due March 9, 2025
Start NOW!!!

Classification

"Chocolate Pretzels"

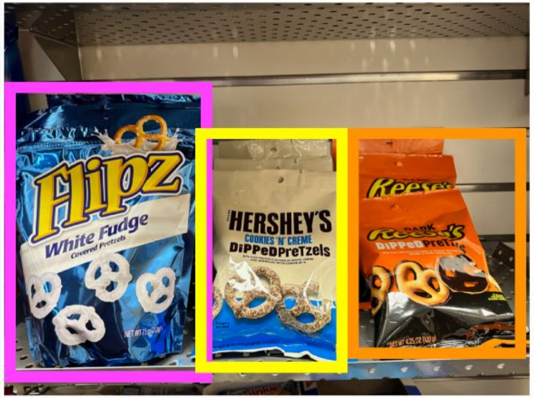
No spatial extent

Semantic Segmentation

Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection

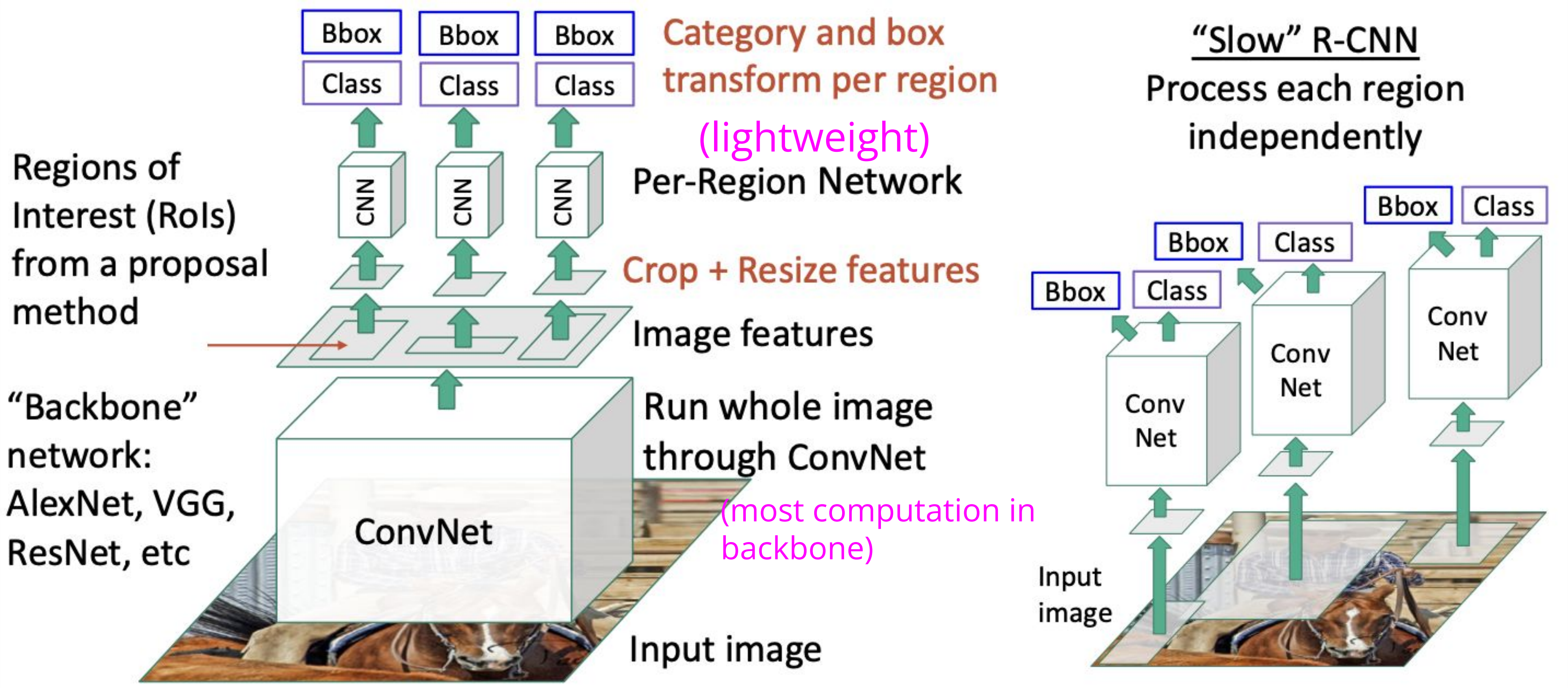


Flipz, Hershey's, Keese's

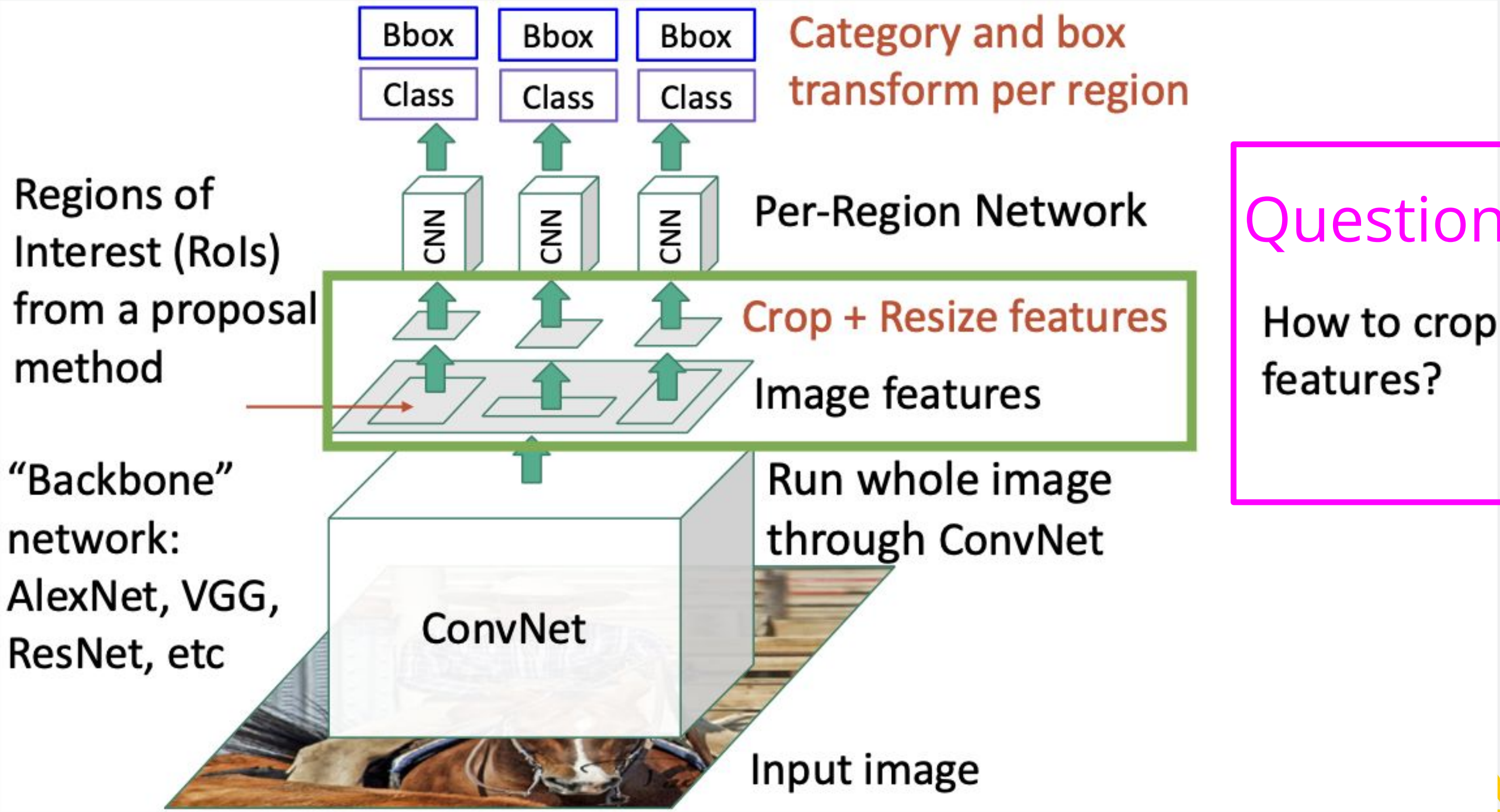
Multiple objects

Instance Segmentation

Recap: Fast R-CNN



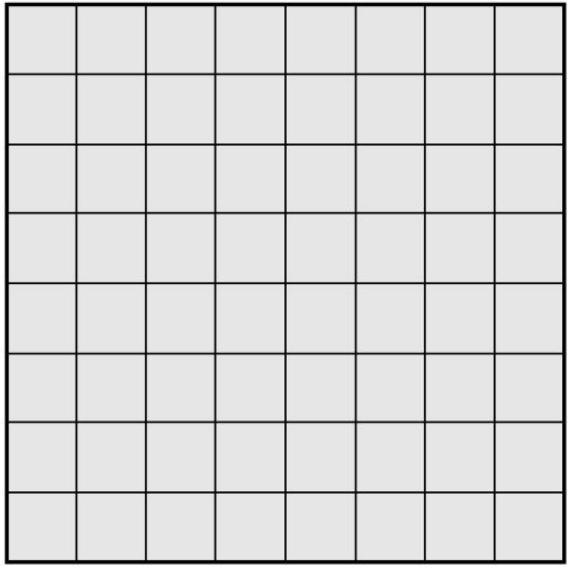
Fast R-CNN



Question:

How to crop features?

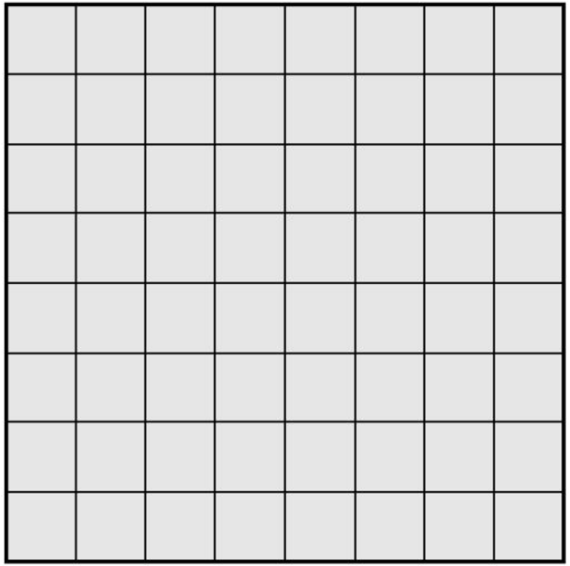
Recall: Receptive Fields



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

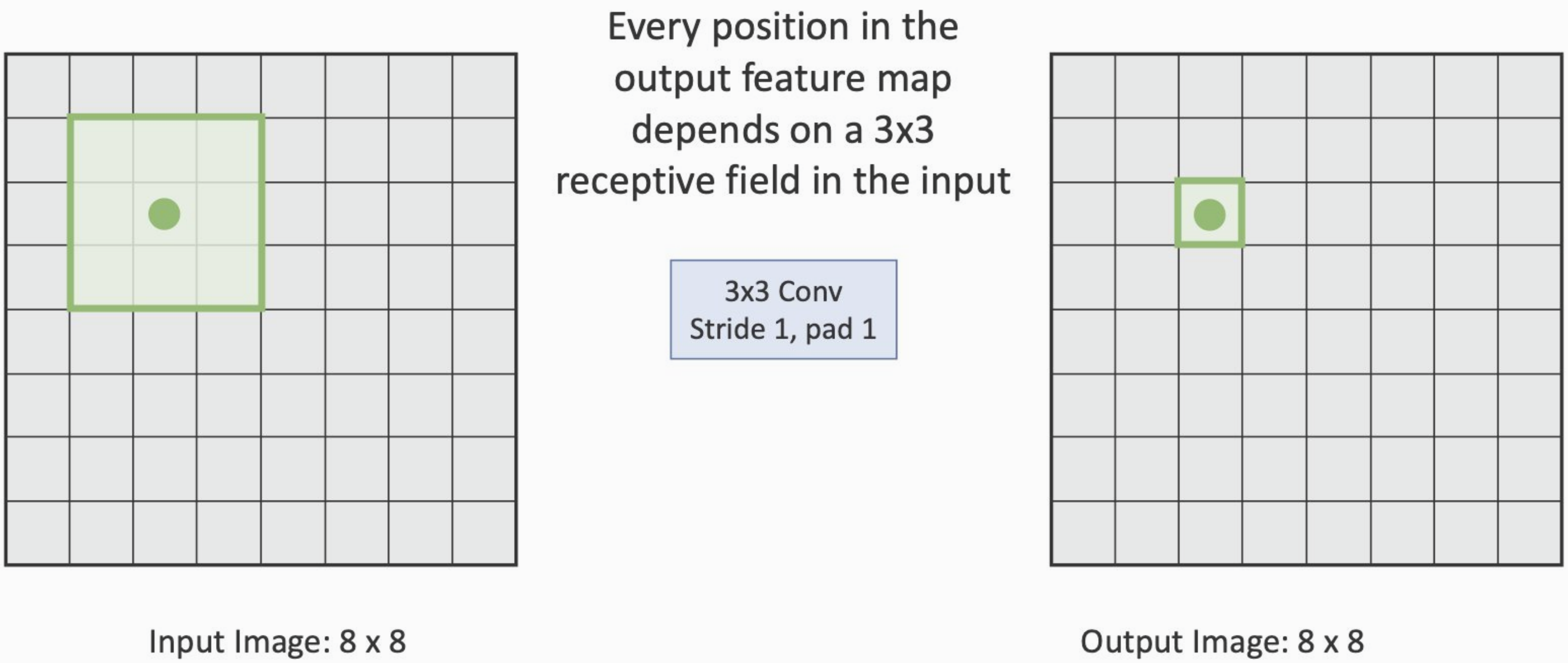
3x3 Conv
Stride 1, pad 1

Input Image: 8 x 8

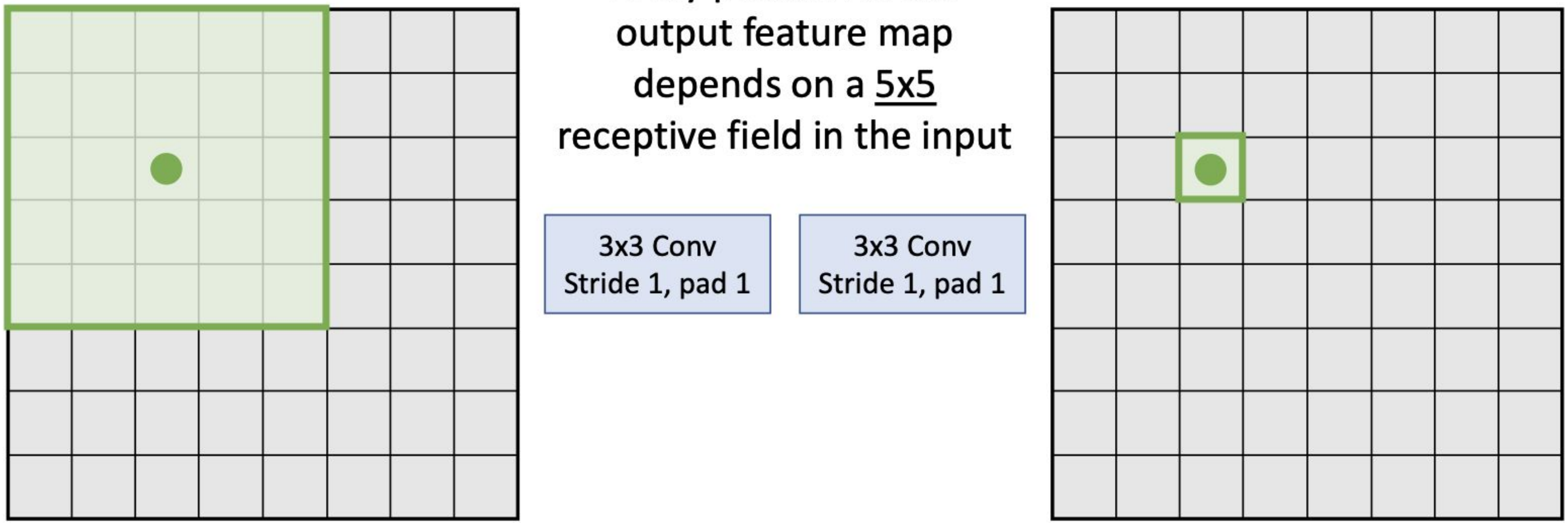


Output Image: 8 x 8

Recall: Receptive Fields



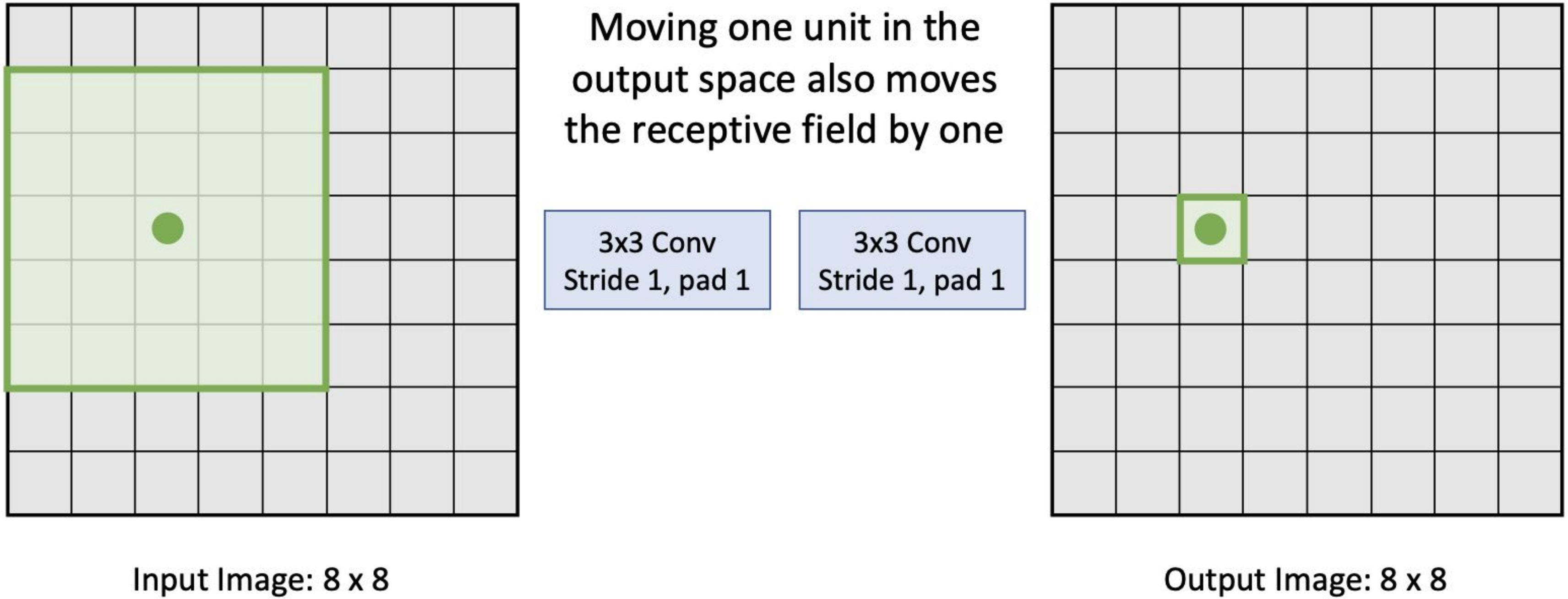
Recall: Receptive Fields



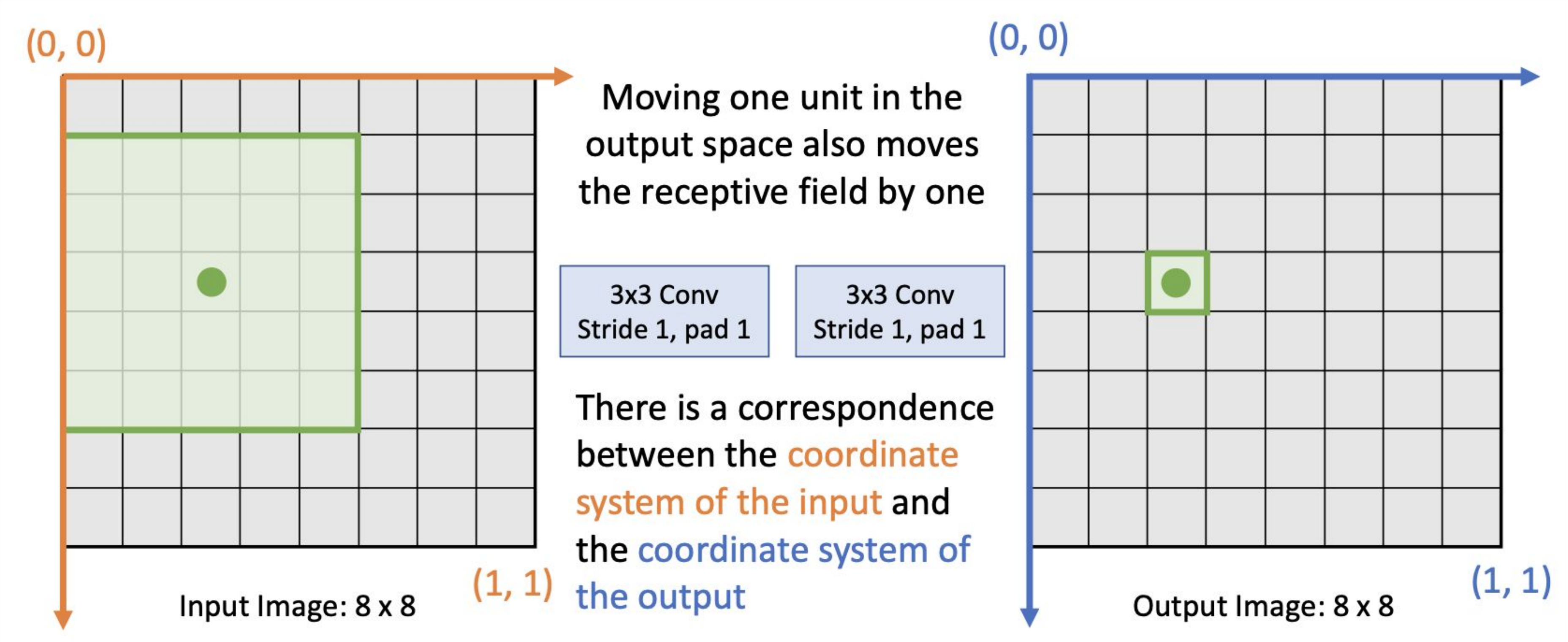
Input Image: 8 x 8

Output Image: 8 x 8

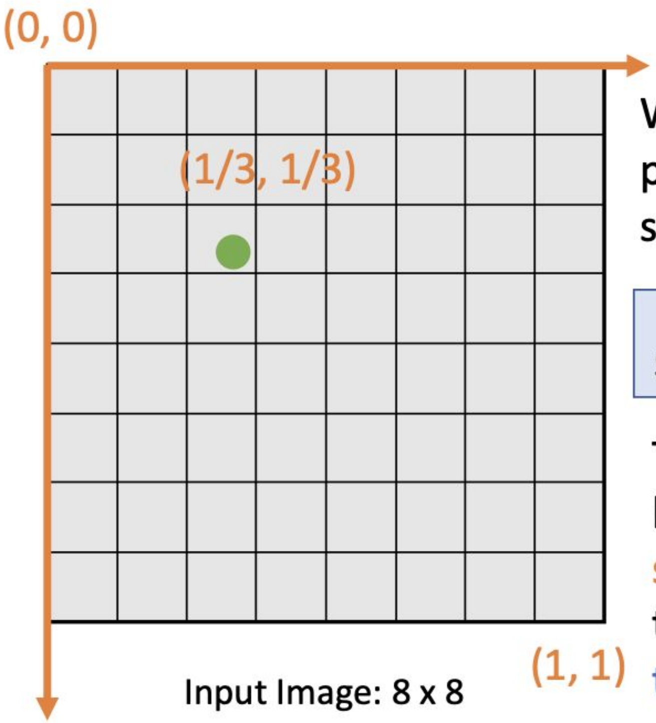
Recall: Receptive Fields



Recall: Receptive Fields



Projecting Points

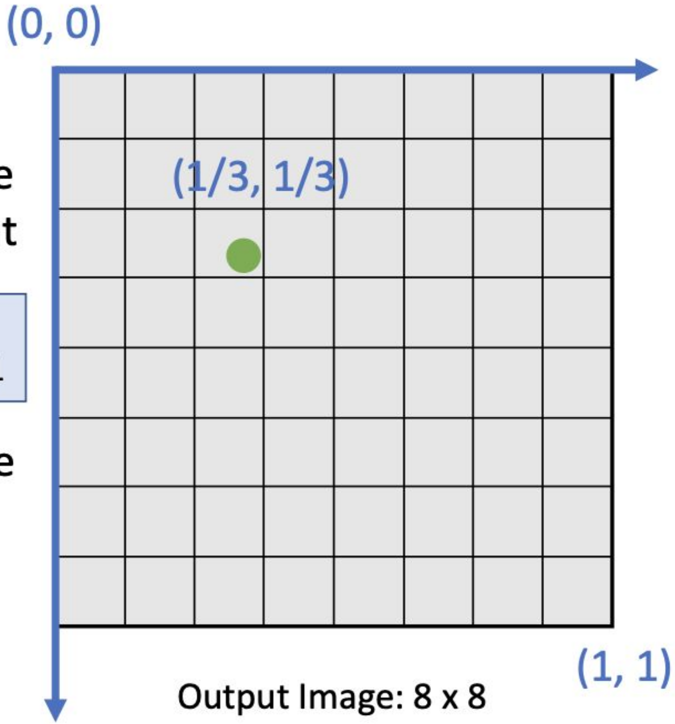


We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

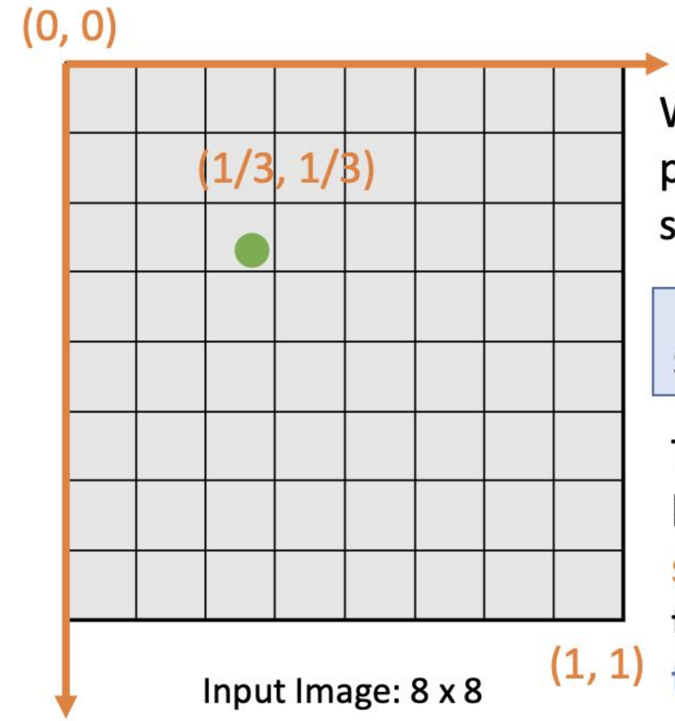
3x3 Conv
Stride 1, pad 1

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



Projecting Points

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

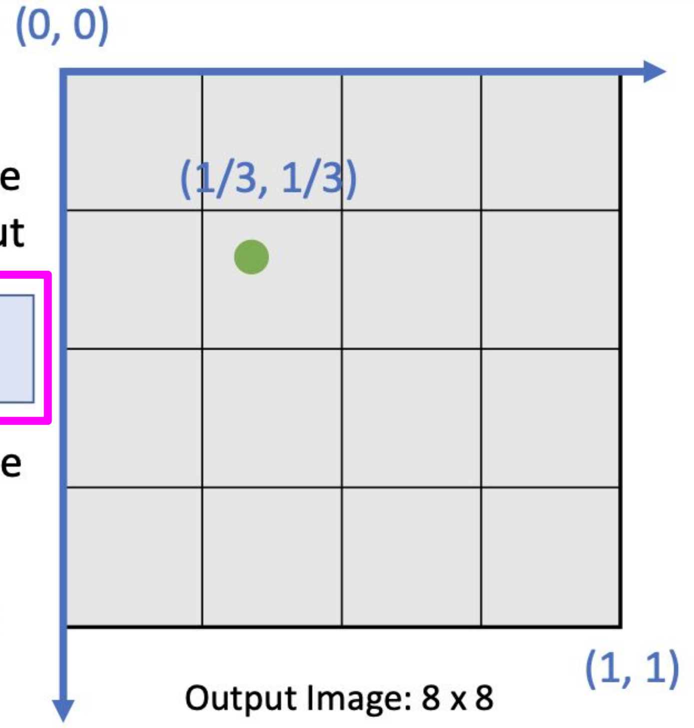


We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

2x2 MaxPool
Stride 2

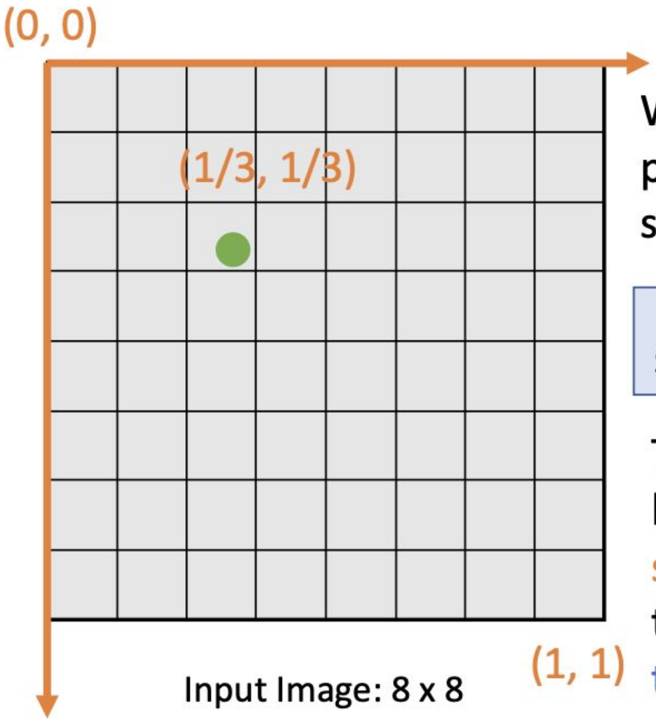
There is a correspondence between the coordinate system of the input and the coordinate system of the output



M | ROBOTICS

Projecting Points

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

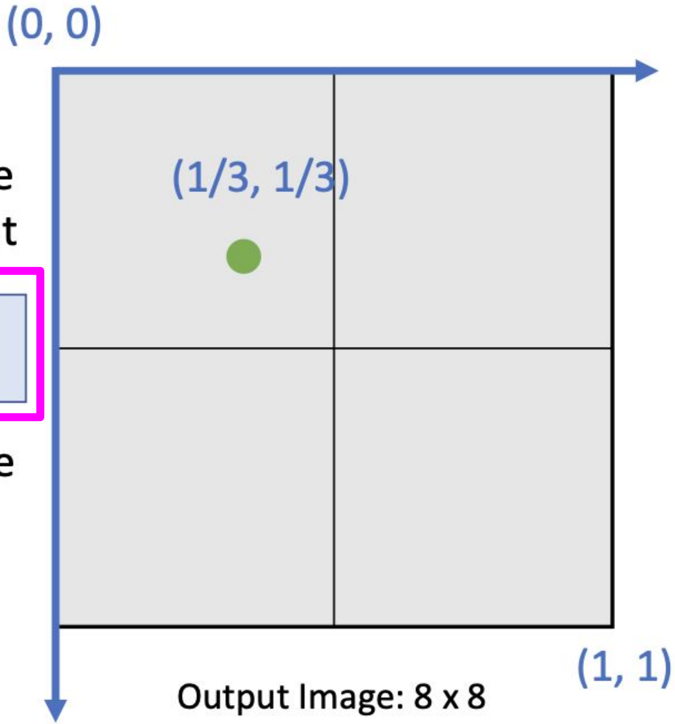


We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

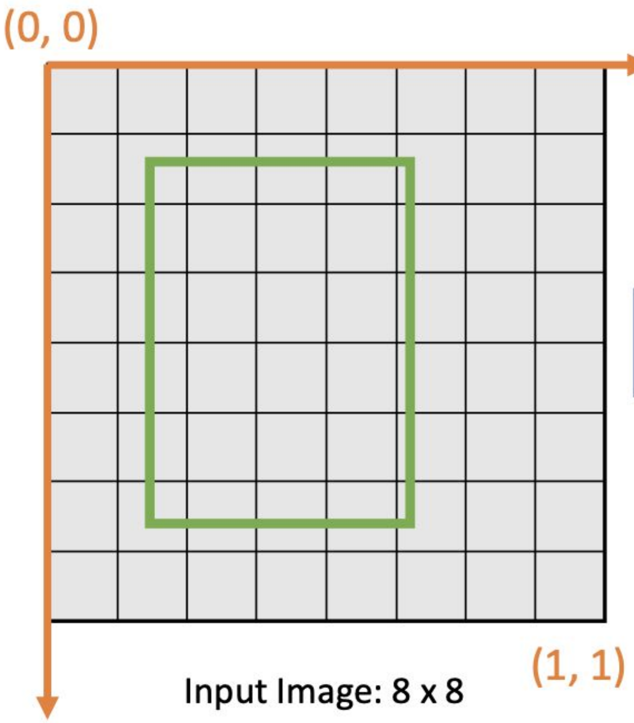
4x4 MaxPool
Stride 4

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



Projecting Points

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

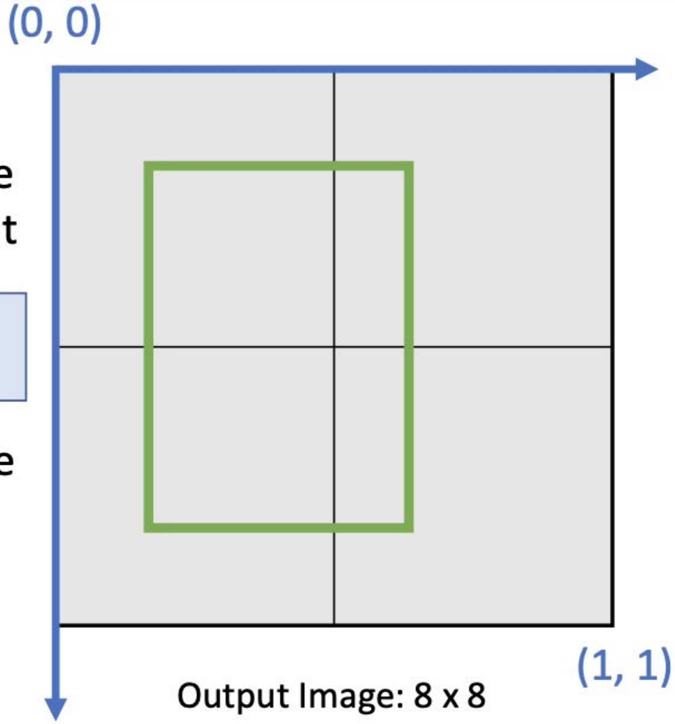


We can align arbitrary points between coordinate system of input and output

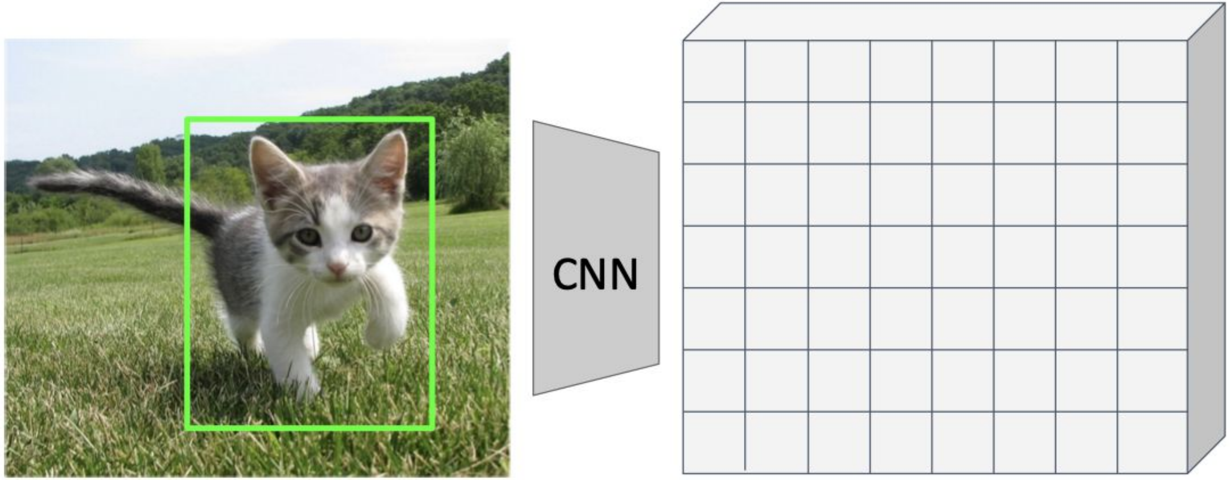
3x3 Conv
Stride 1, pad 1

4x4 MaxPool
Stride 4

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



Cropping Features: RoI Pool



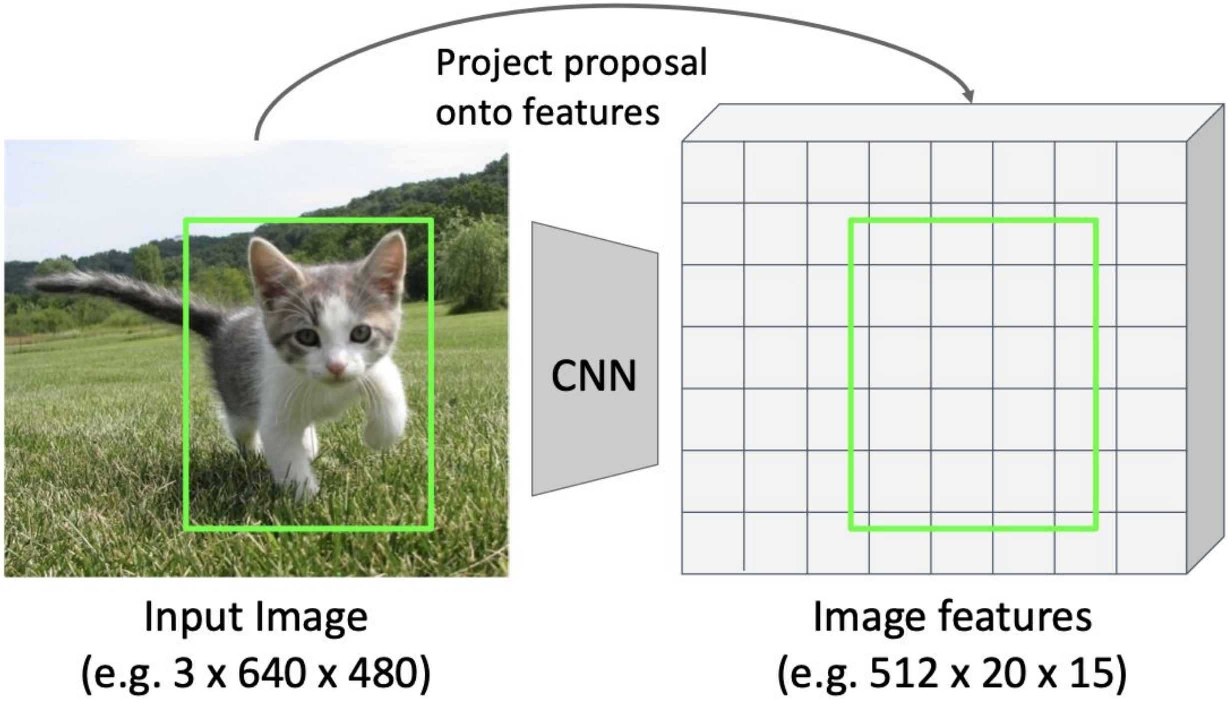
Input Image
(e.g. 3 x 640 x 480)

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

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

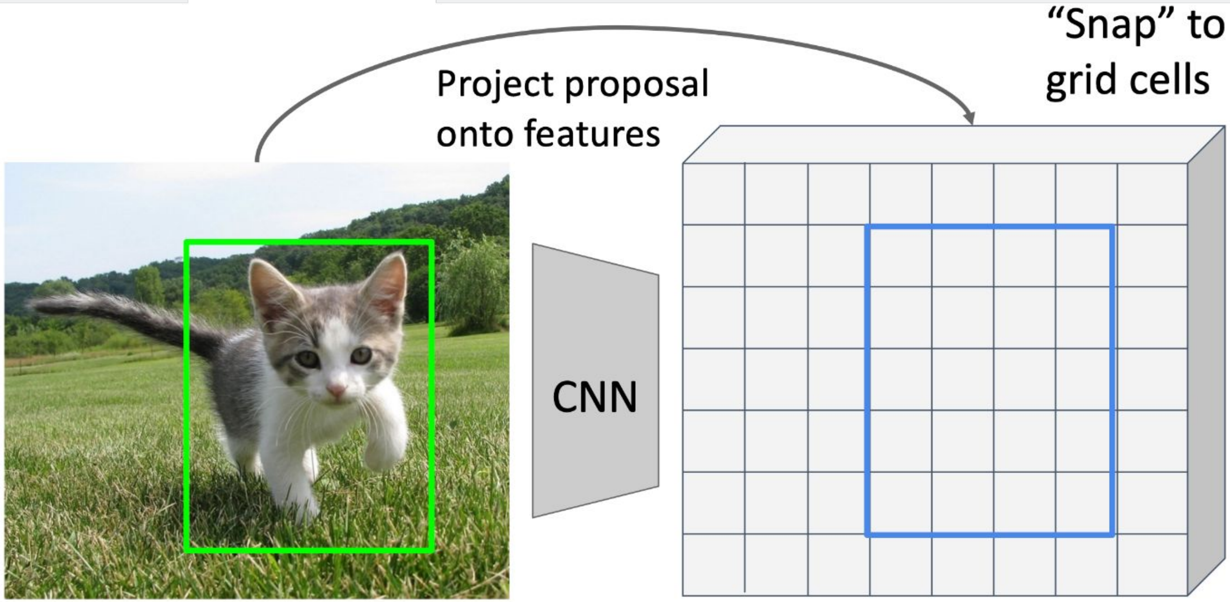
Girshick, “Fast R-CNN”, ICCV 2015.
https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

Cropping Features: RoI Pool



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

Cropping Features: RoI Pool

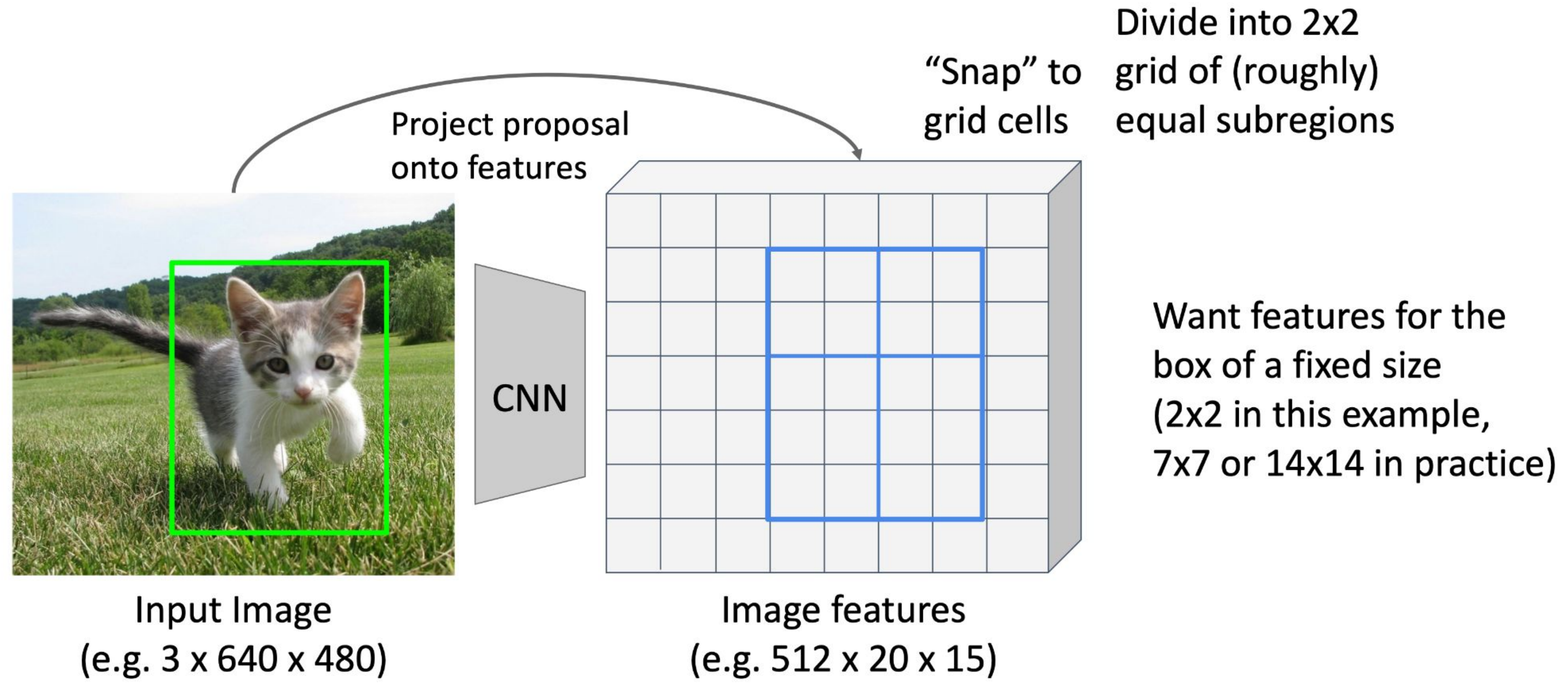


Input Image
(e.g. 3 x 640 x 480)

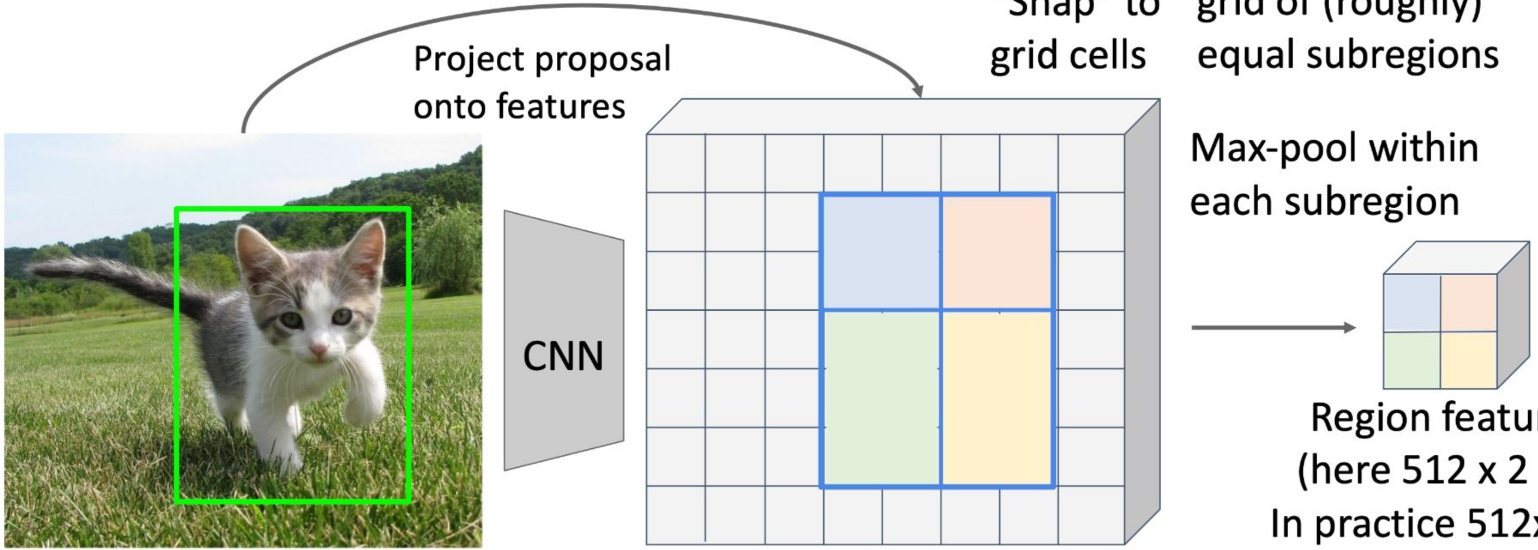
Image features
(e.g. 512 x 20 x 15)

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

Cropping Features: RoI Pool



Cropping Features: RoI Pool

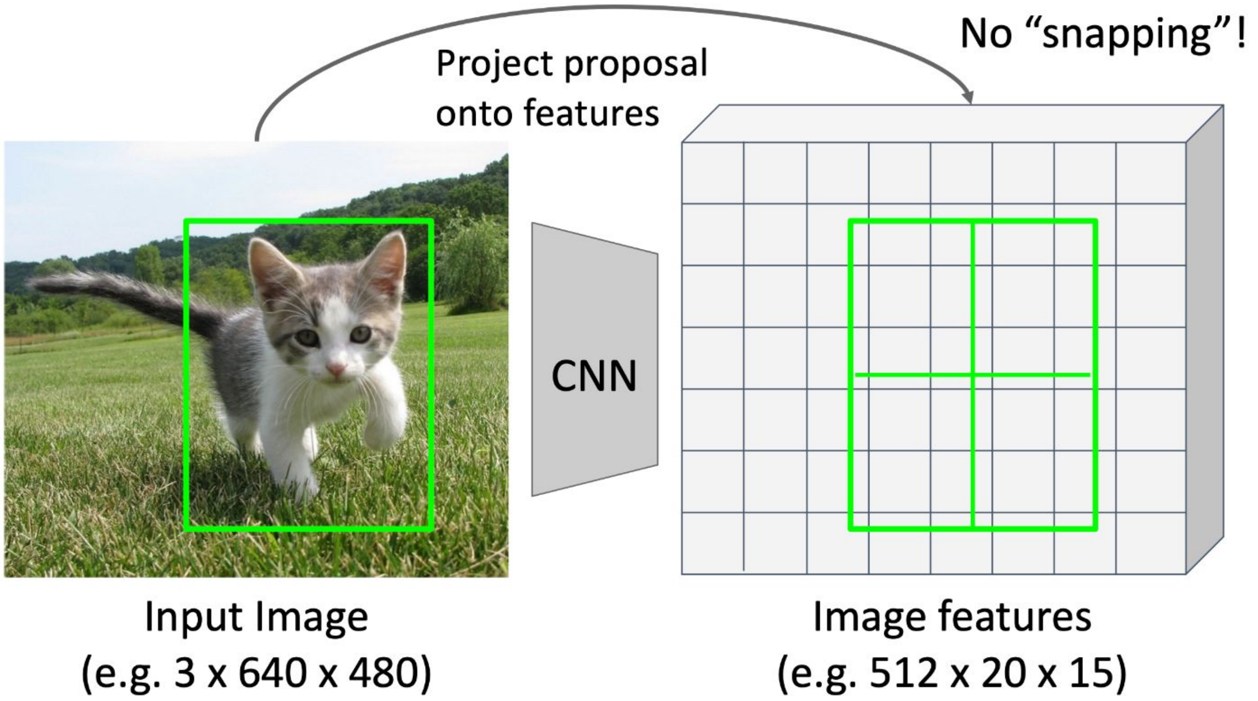


Region features always the same size even if input regions have different sizes!

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

Cropping Features: RoI Align

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



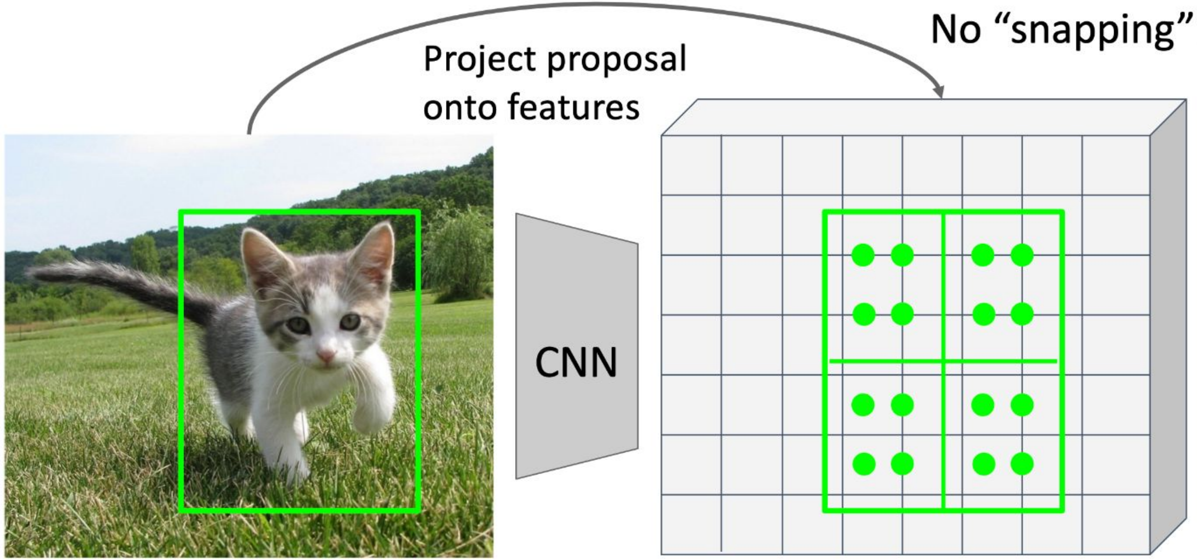
Want features for the
box of a fixed size
(2x2 in this example,
7x7 or 14x14 in practice)

Cropping Features: RoI Align

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

No “snapping”!

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



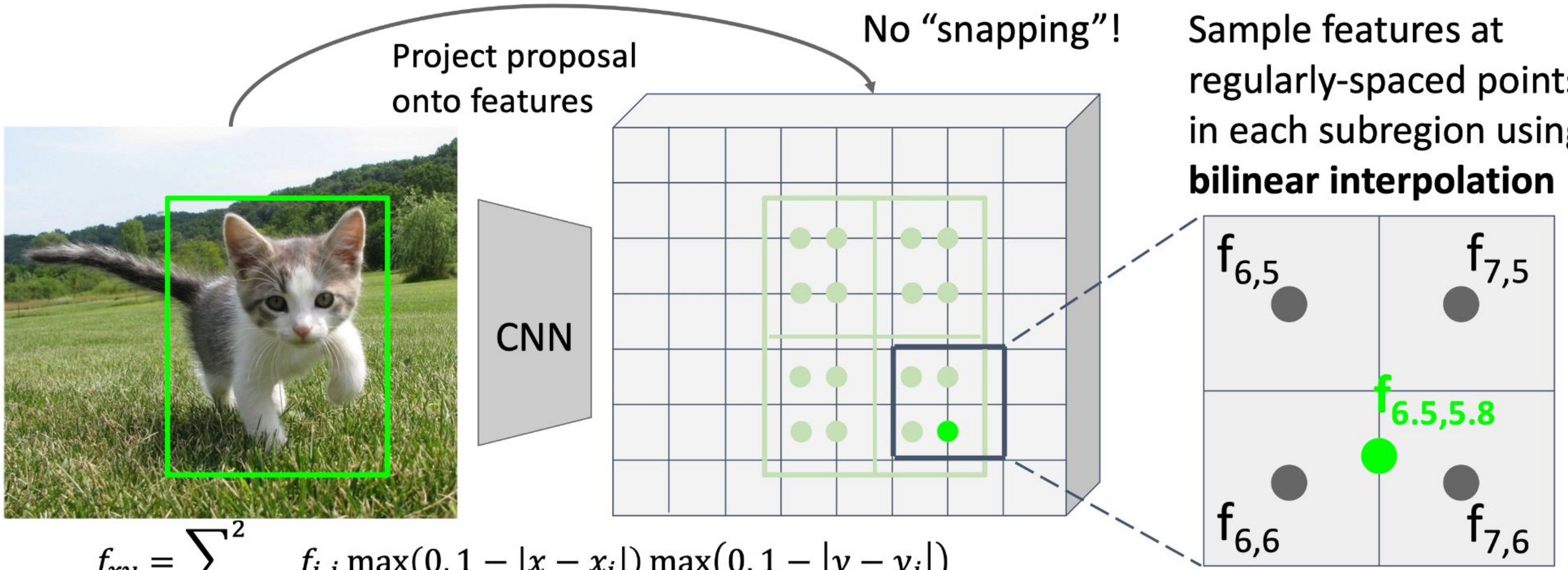
Input Image
(e.g. 3 x 640 x 480)

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

Cropping Features: RoI Align

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

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



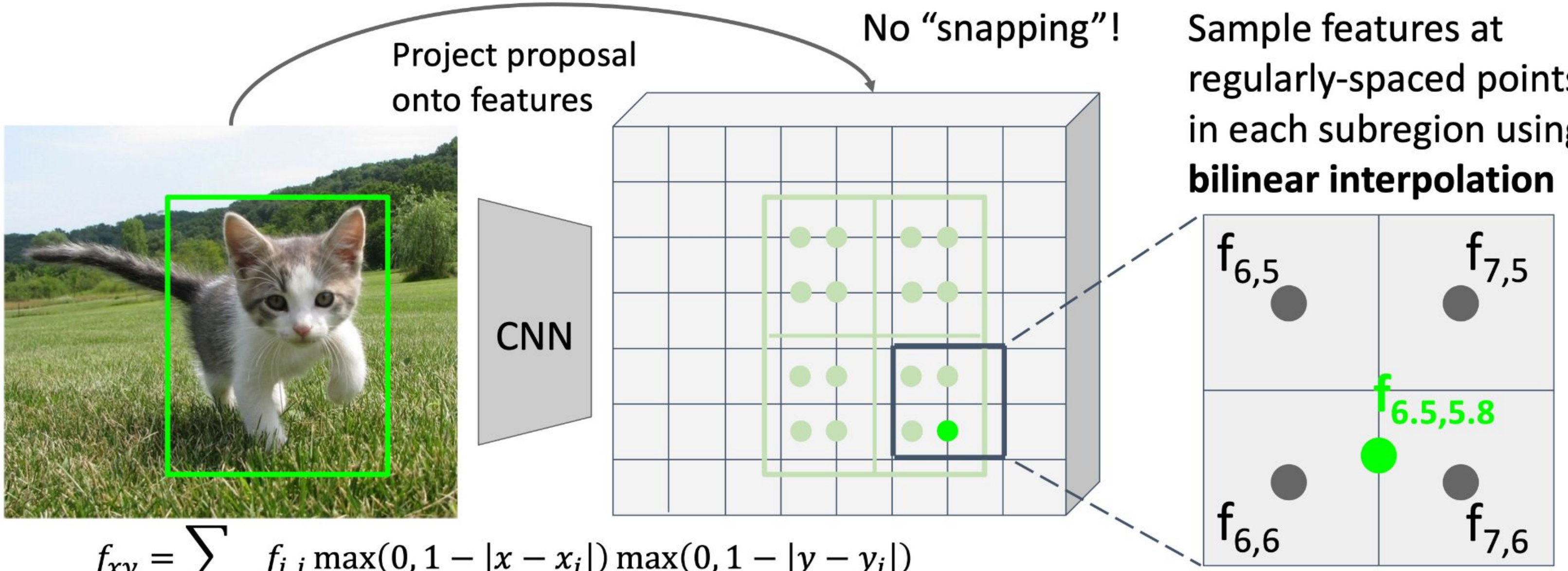
$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

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

Cropping Features: RoI Align

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

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

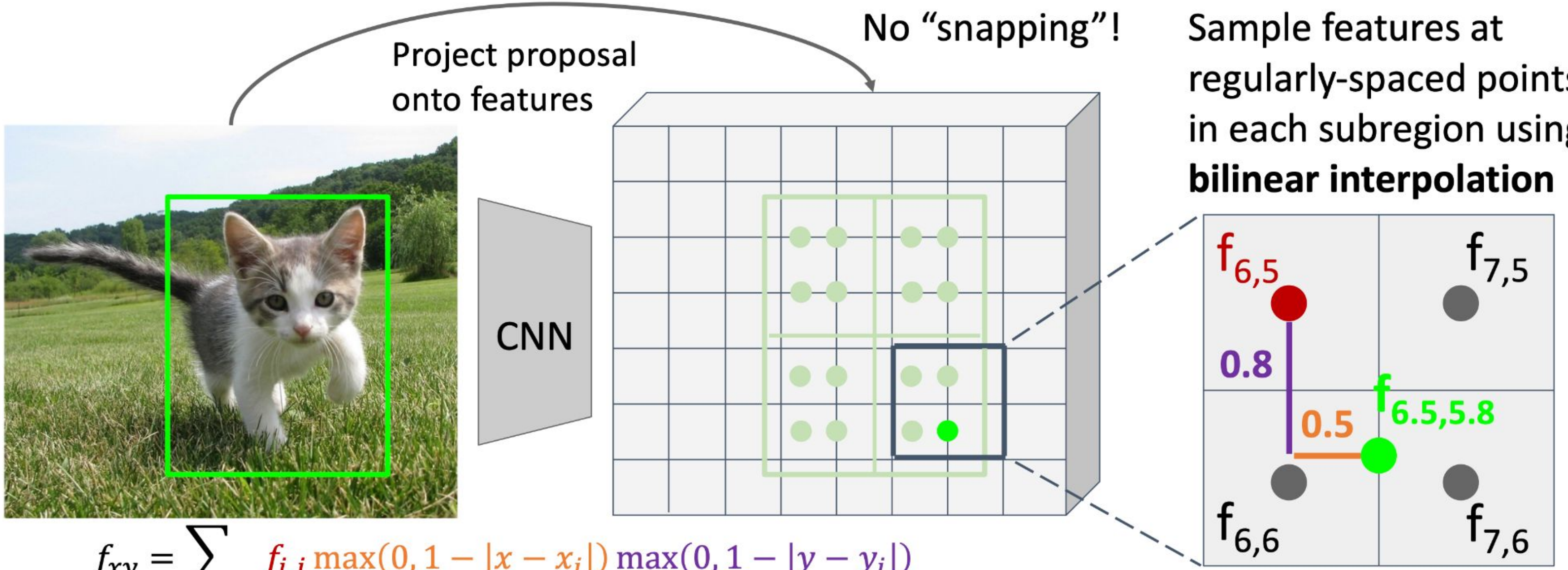


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

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

Cropping Features: RoI Align



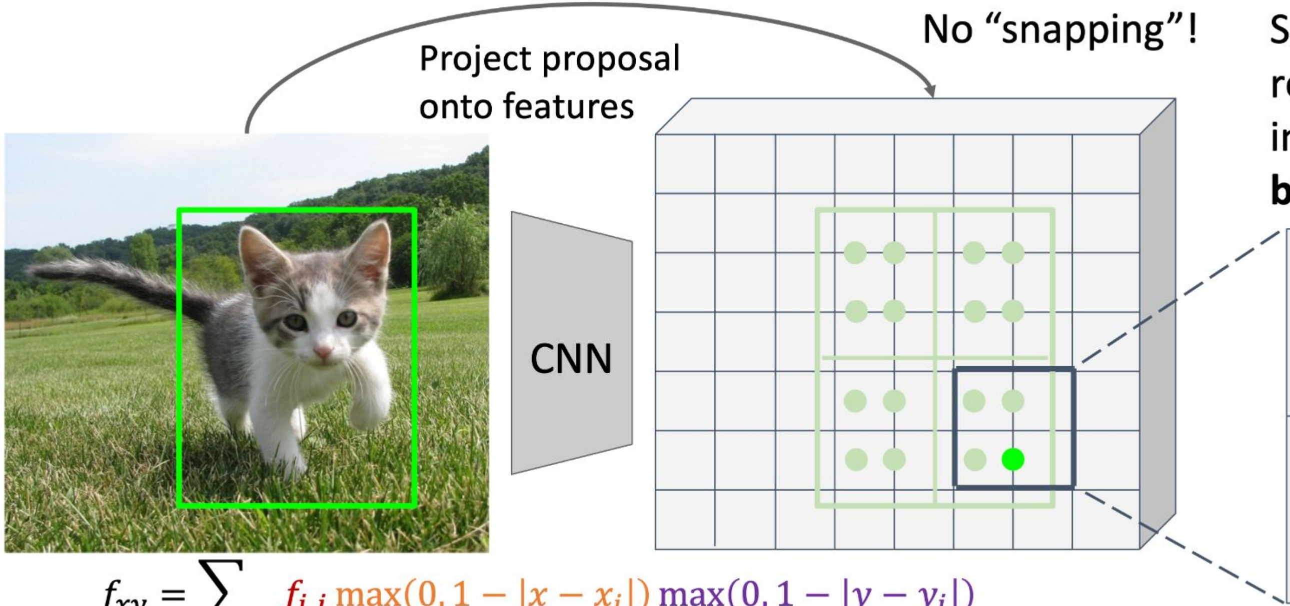
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

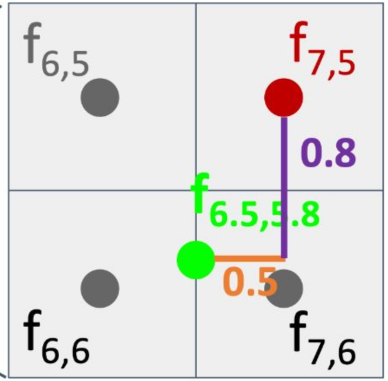
$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align



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

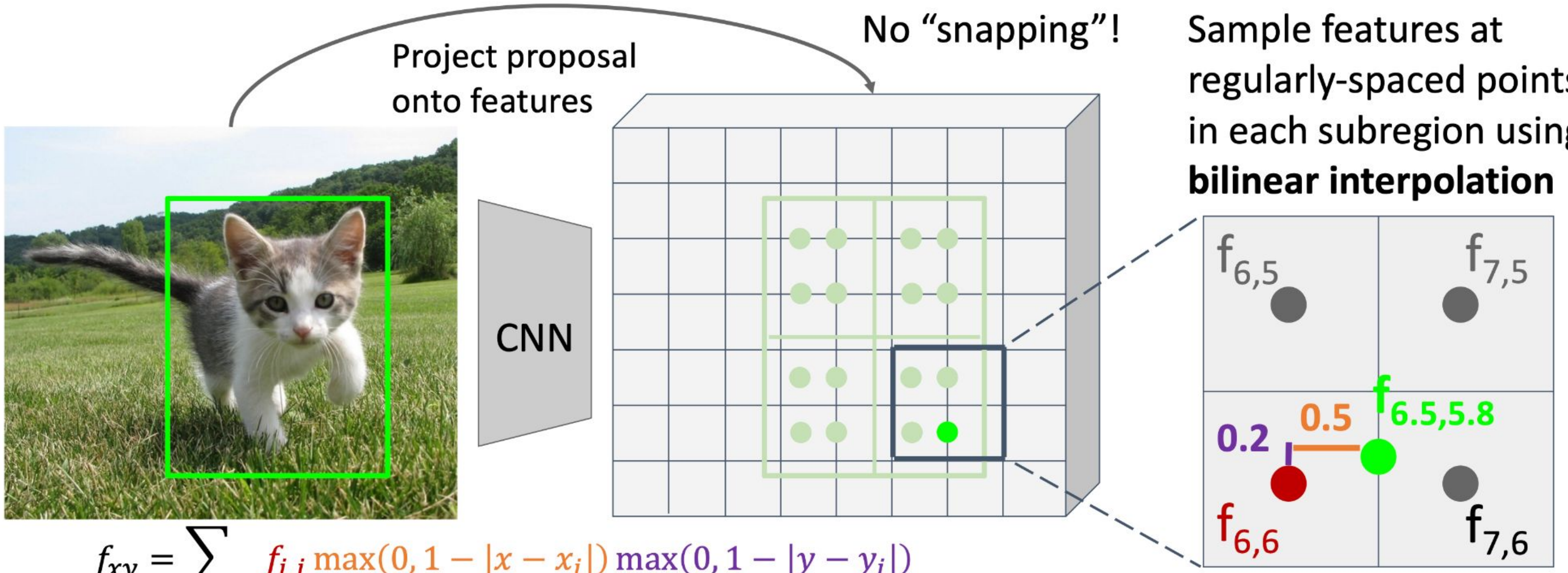


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

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

Cropping Features: RoI Align

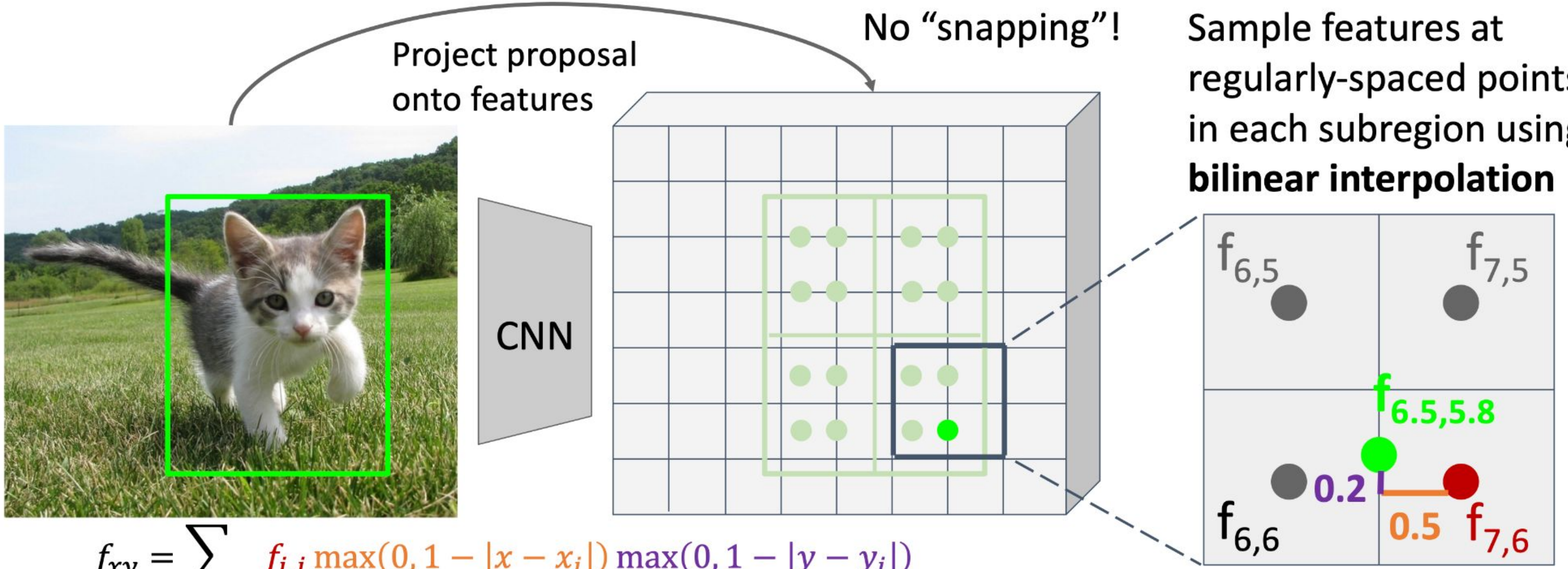


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

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

Cropping Features: RoI Align



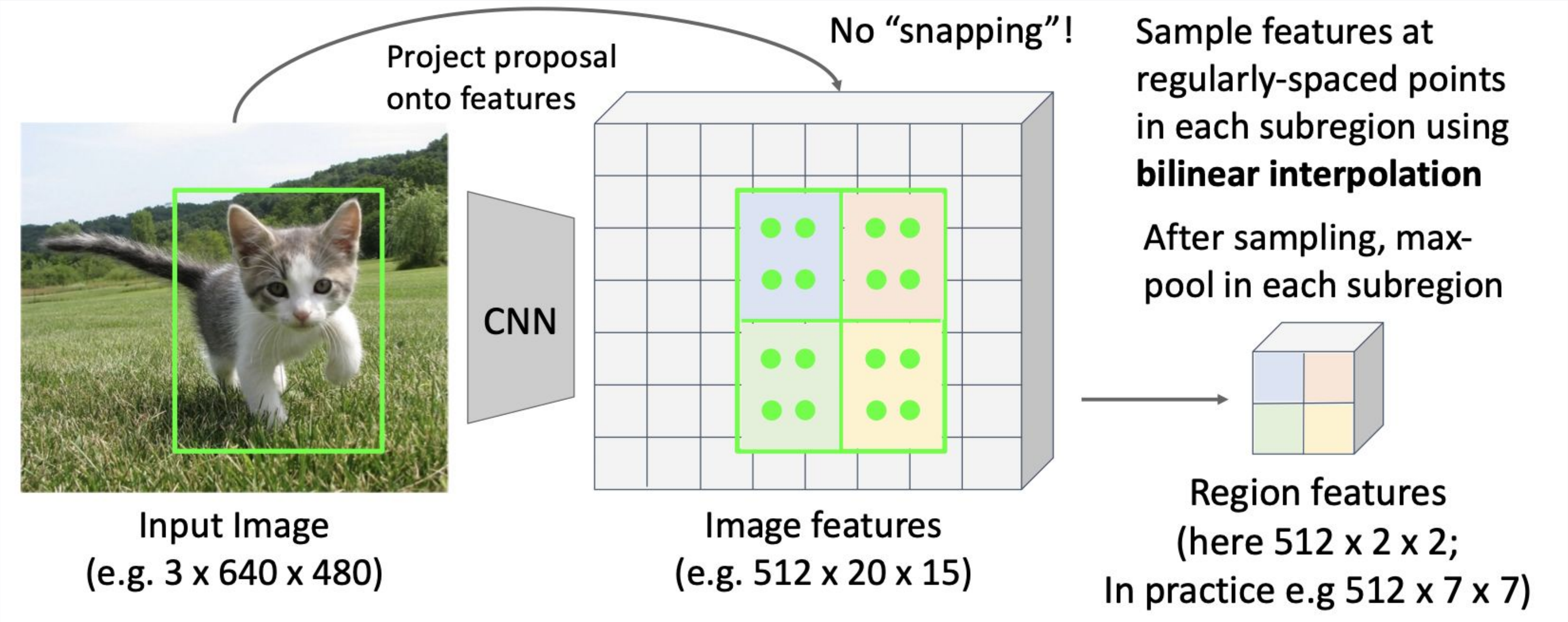
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

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

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

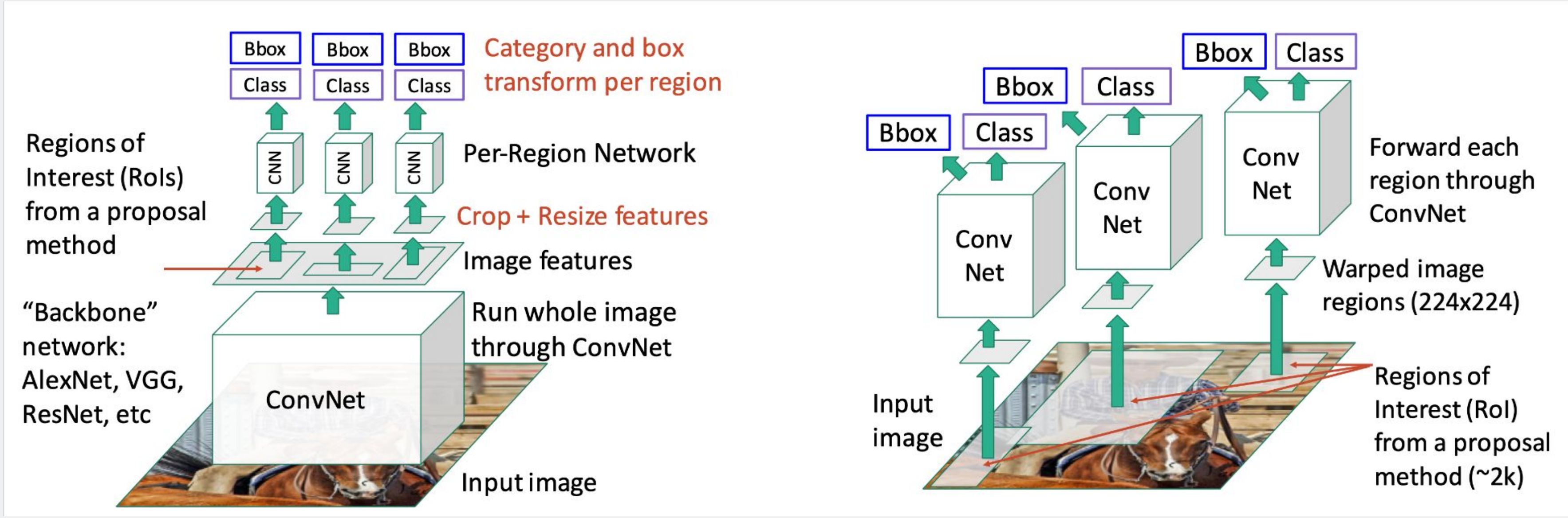
Cropping Features: RoI Pool



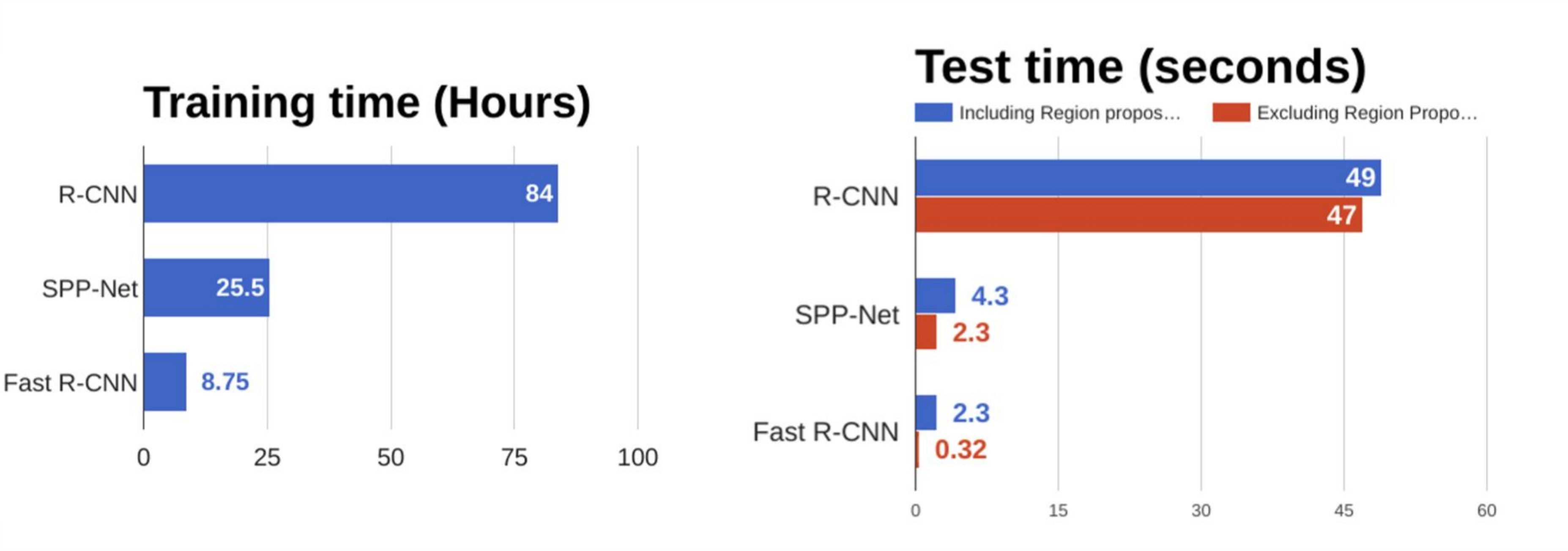
Fast R-CNN vs “Slow” R-CNN

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

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

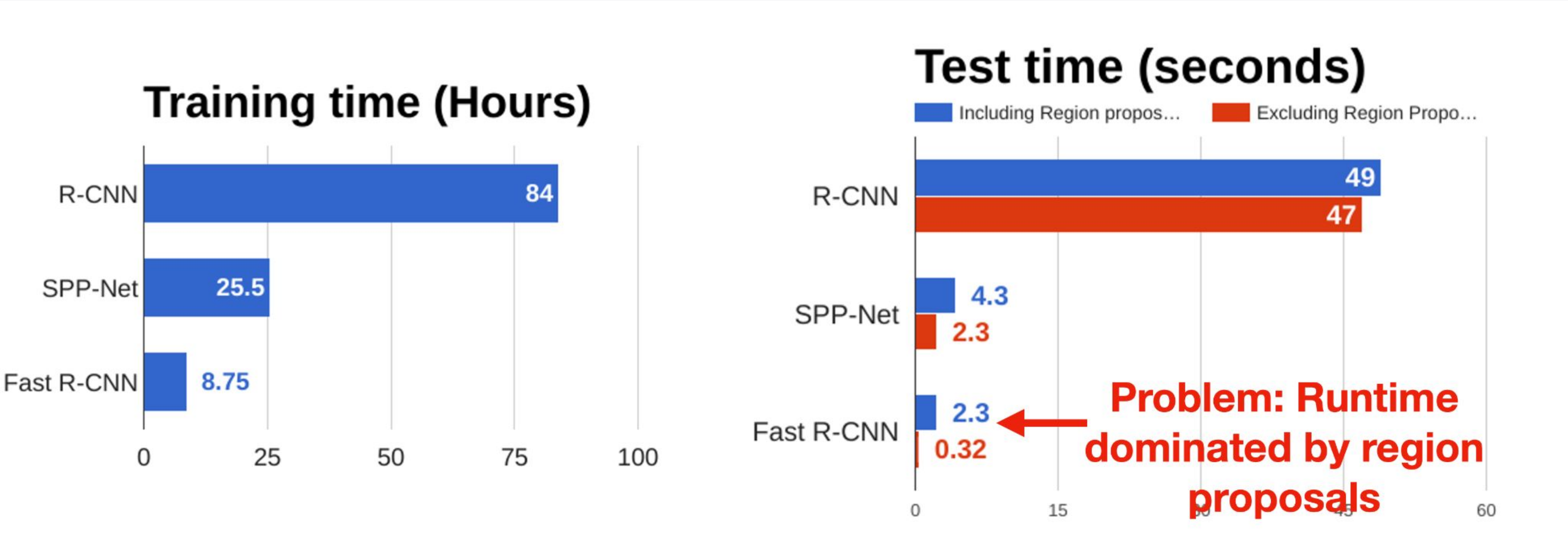


Fast R-CNN vs “Slow” R-CNN

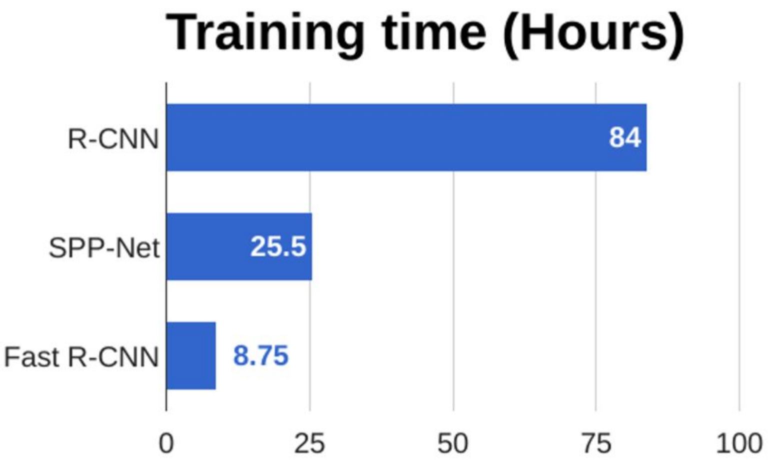


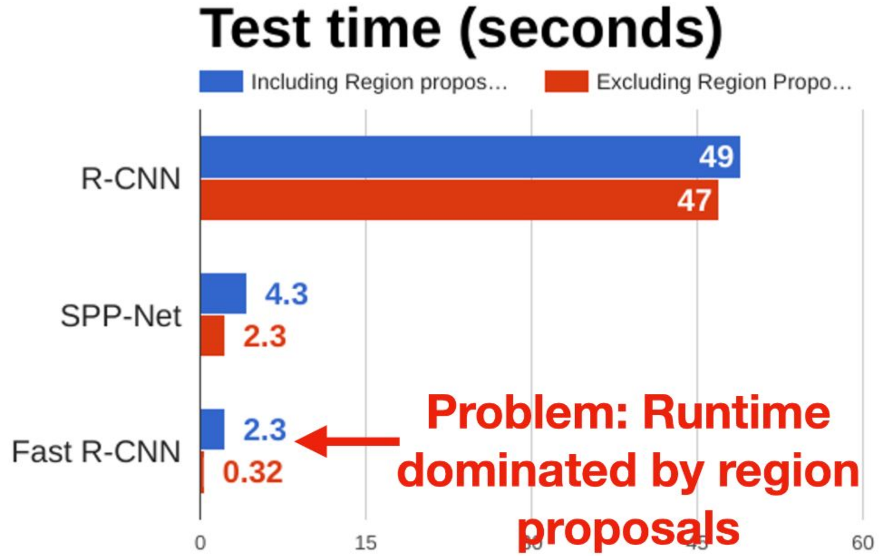
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs “Slow” R-CNN



Fast R-CNN vs "Slow" R-CNN





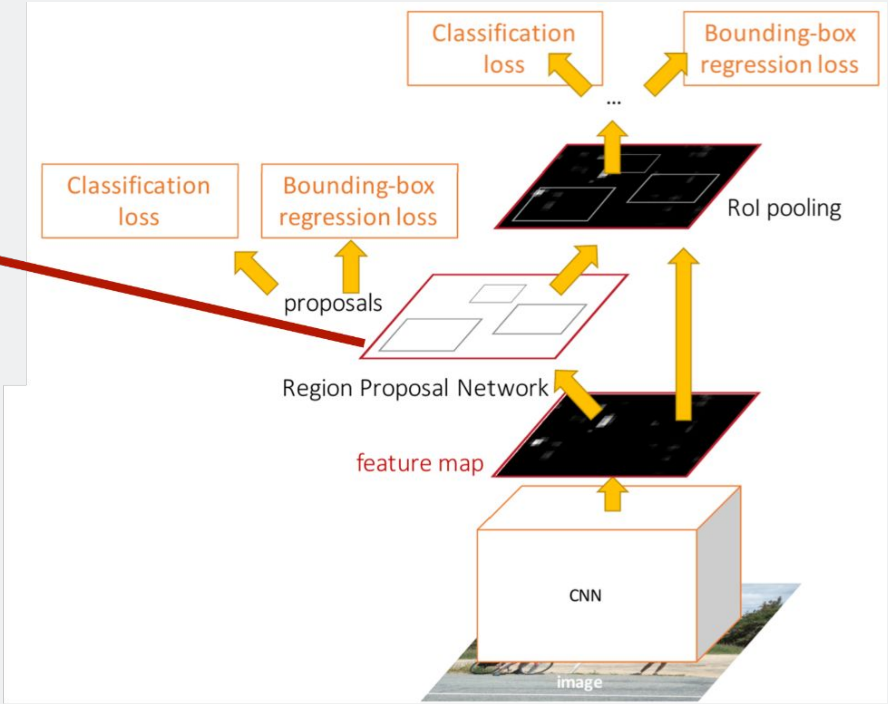
Problem: Runtime dominated by region proposals

Recall: Region proposals computed by heuristic "Selective search" algorithm on CPU — let's learn them with a CNN

Faster R-CNN: Learnable Region Proposals

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

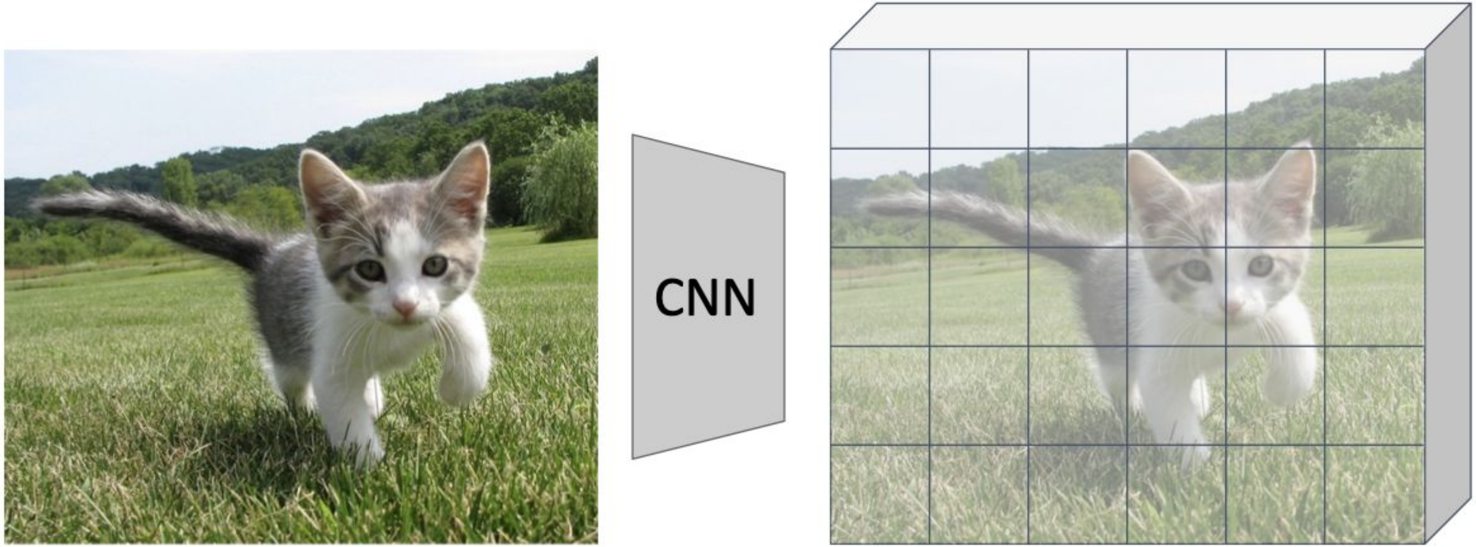
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



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

Region Proposal Network (RPN)

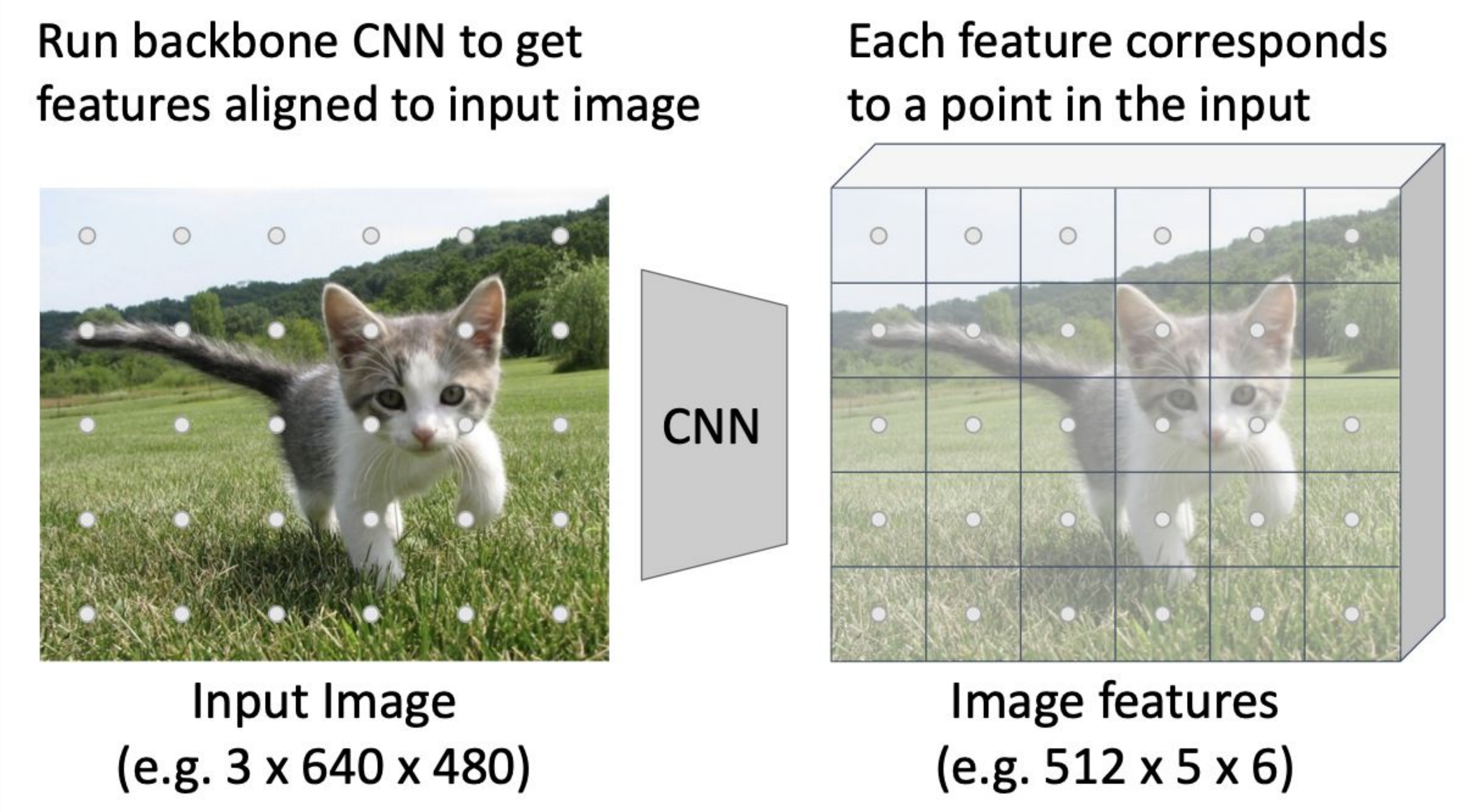
Run backbone CNN to get features aligned to input image



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

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

Region Proposal Network (RPN)

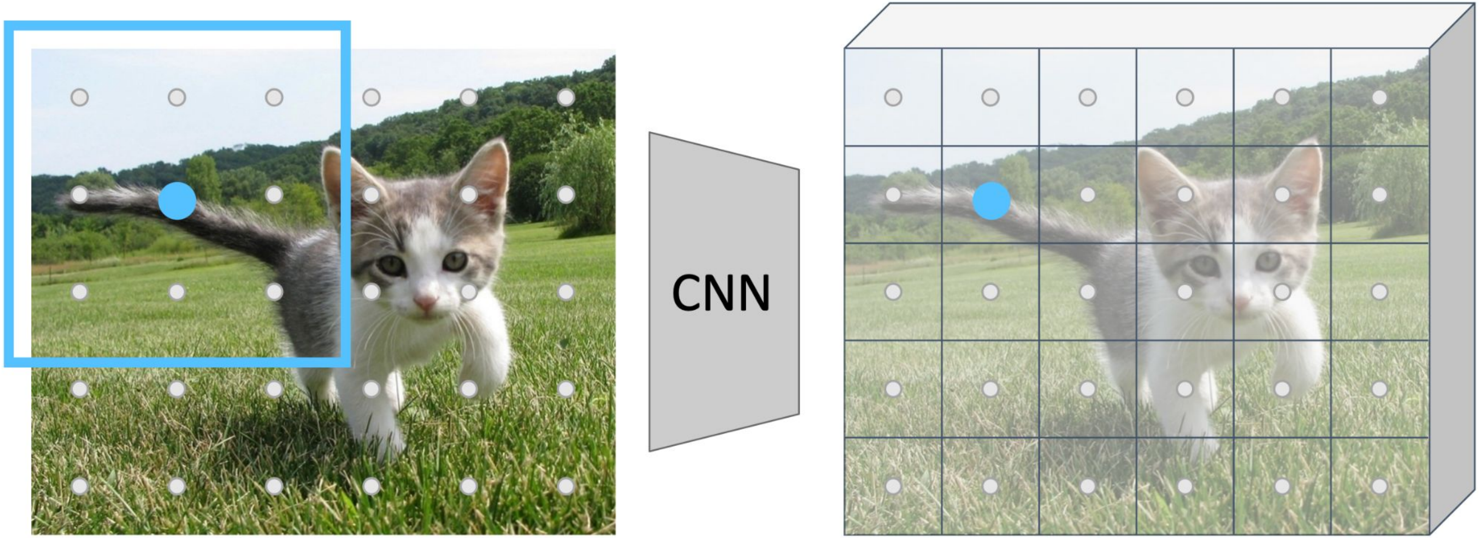


Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

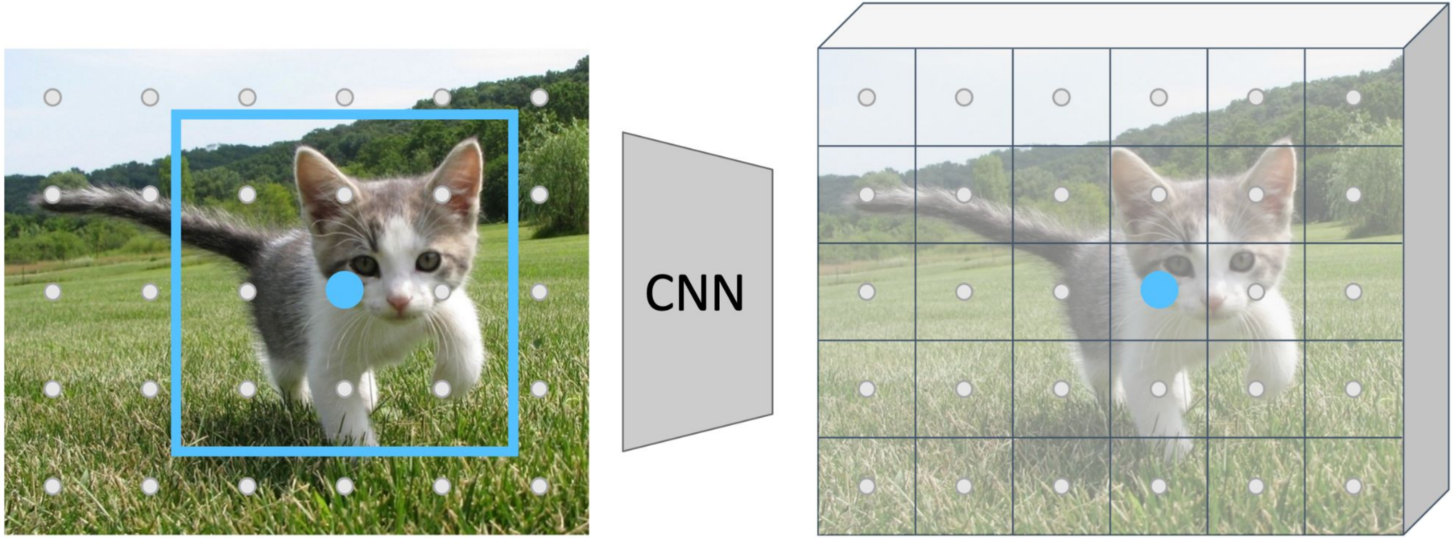
Image features
(e.g. 512 x 5 x 6)

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

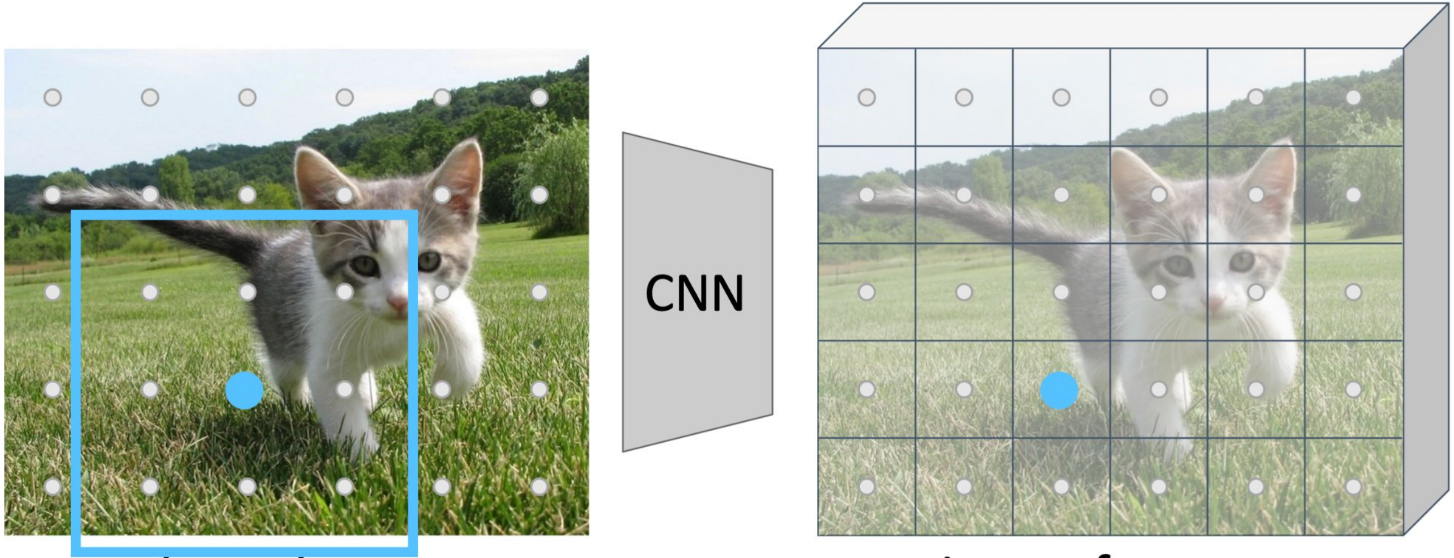
Image features
(e.g. 512 x 5 x 6)

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

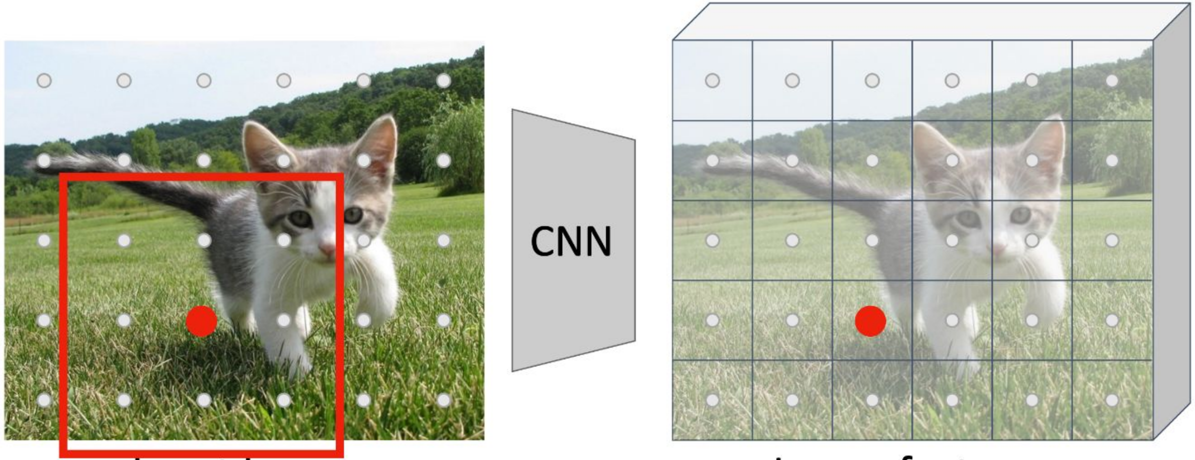
Image features
(e.g. 512 x 5 x 6)

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

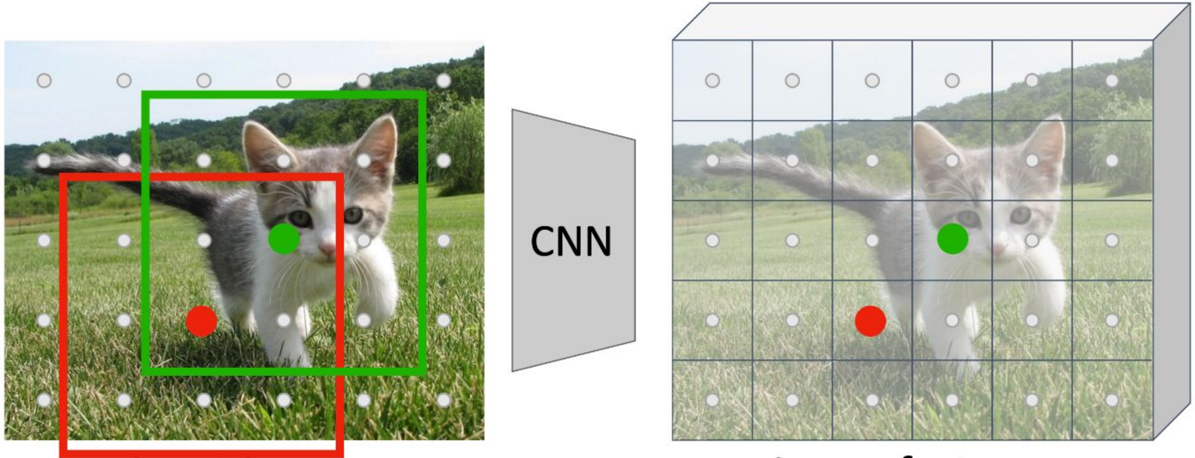
Classify each anchor as positive (object) or negative (no object)

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

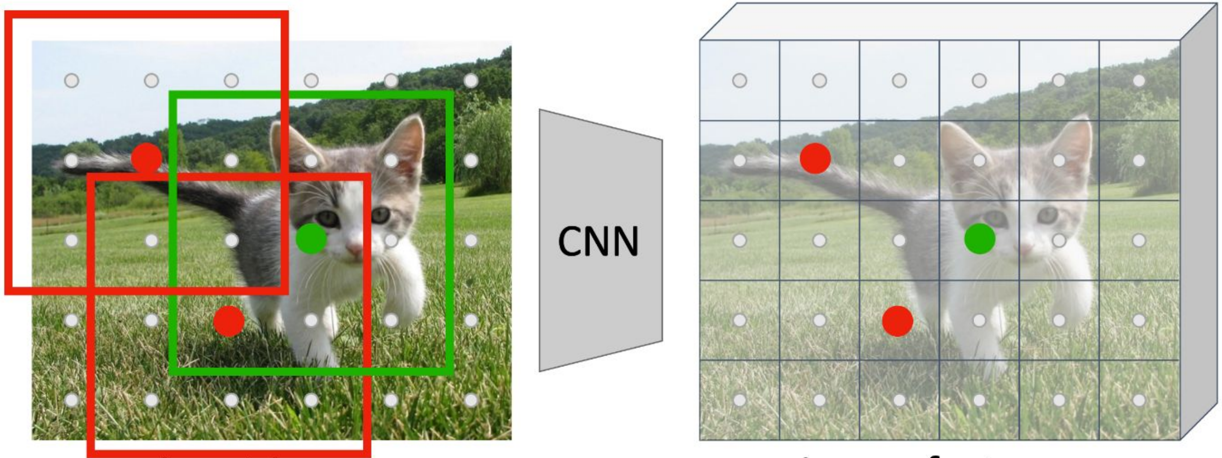
Classify each anchor as positive (object) or negative (no object)

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

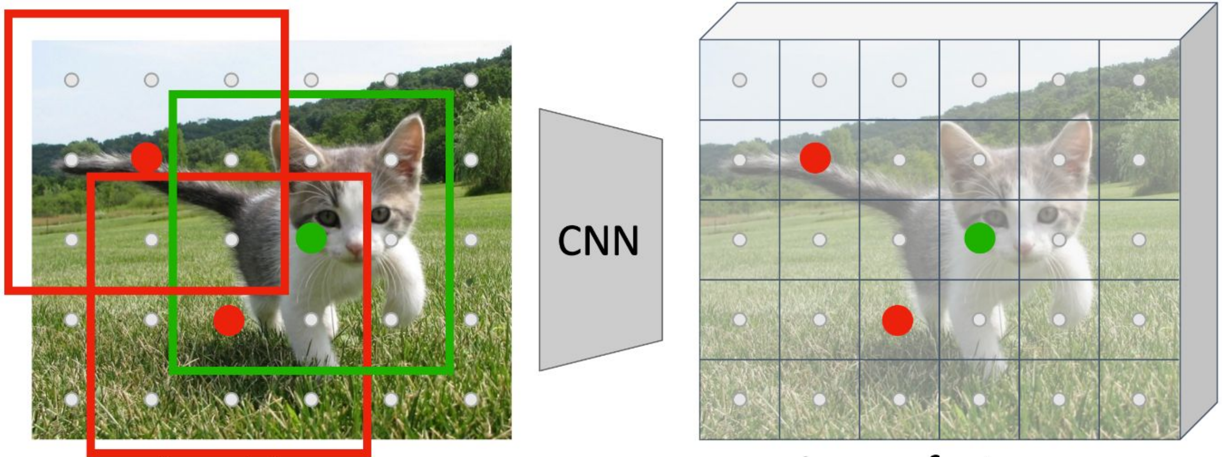
Image features
(e.g. 512 x 5 x 6)

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

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

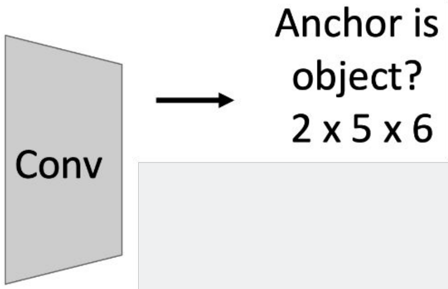
Each feature corresponds to a point in the input



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

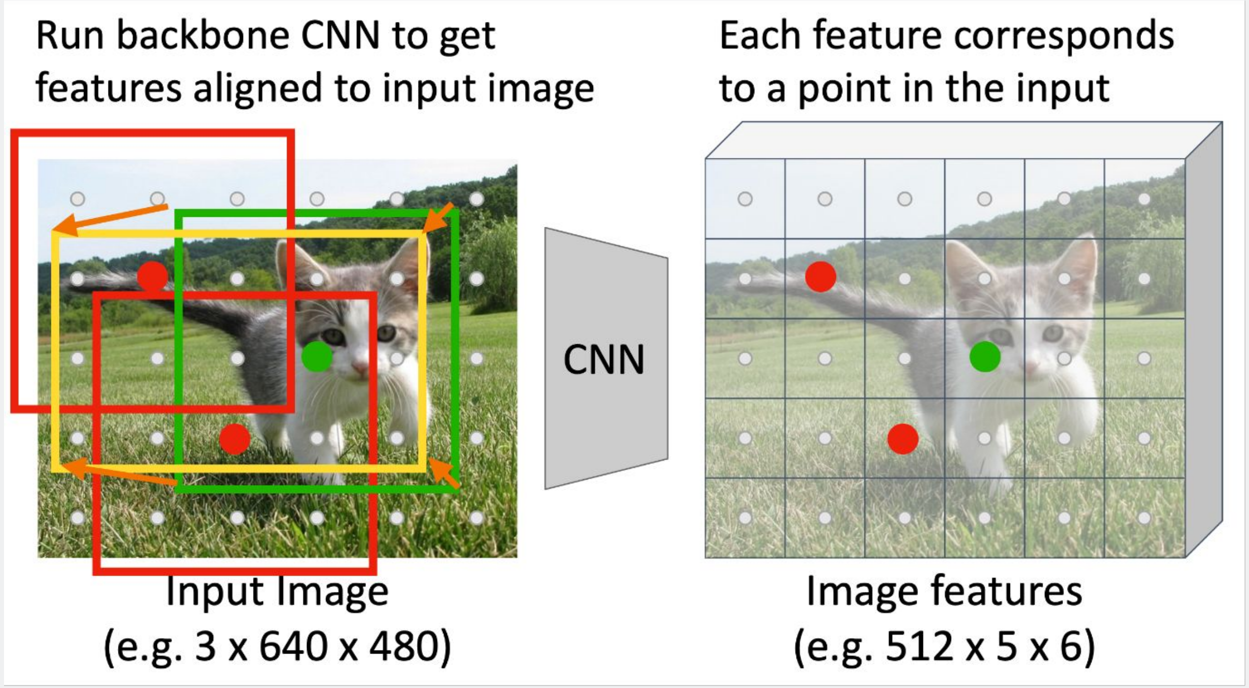
Image features
(e.g. 512 x 5 x 6)

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

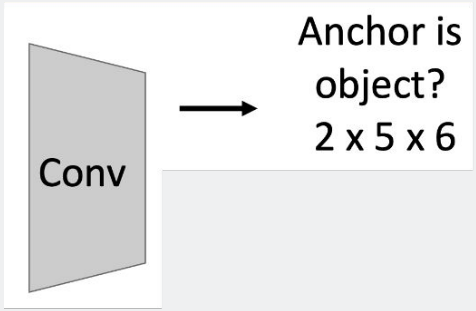


Classify each anchor as positive (object) or negative (no object)

Region Proposal Network (RPN)

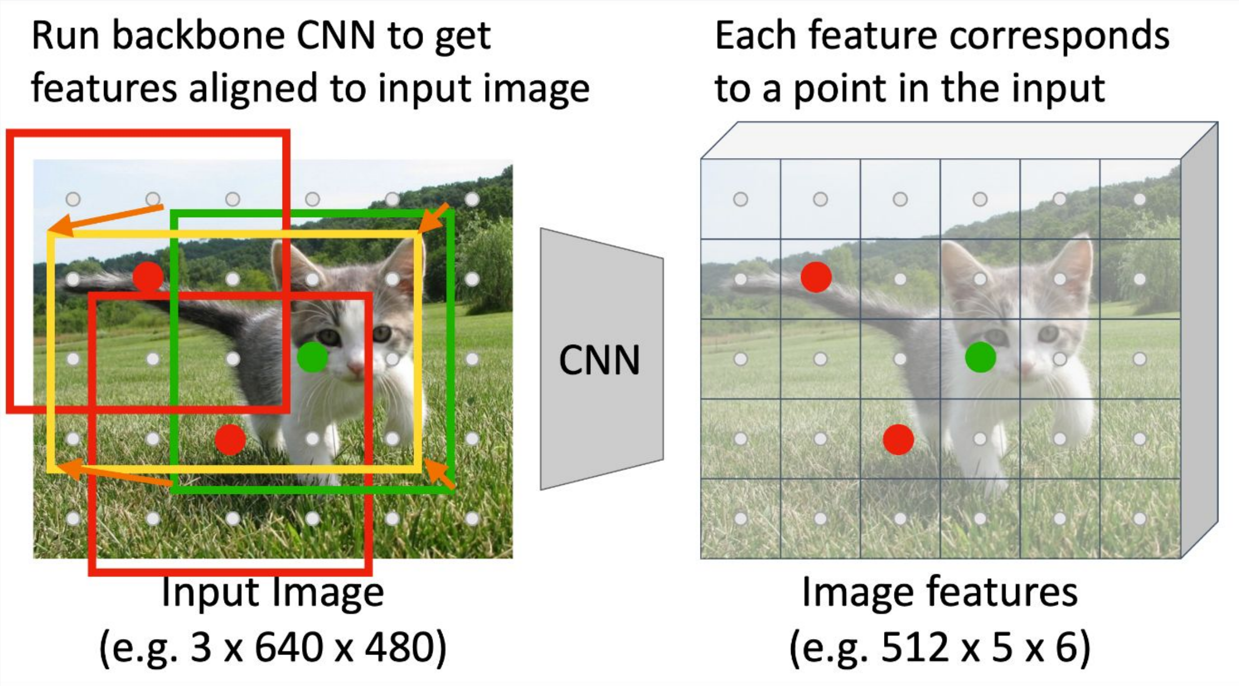


For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)

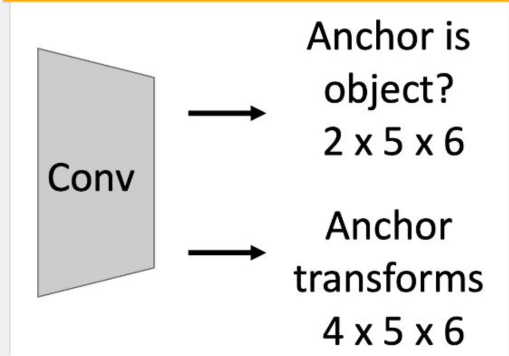


Classify each anchor as positive (object) or negative (no object)

Region Proposal Network (RPN)

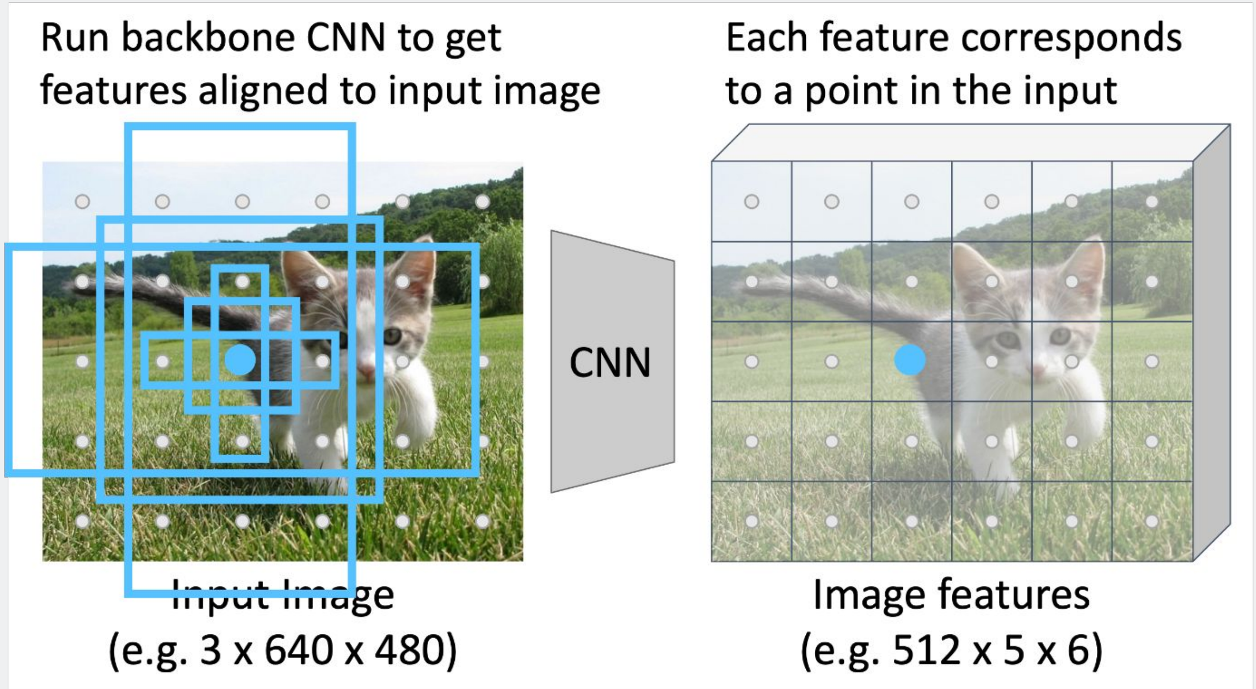


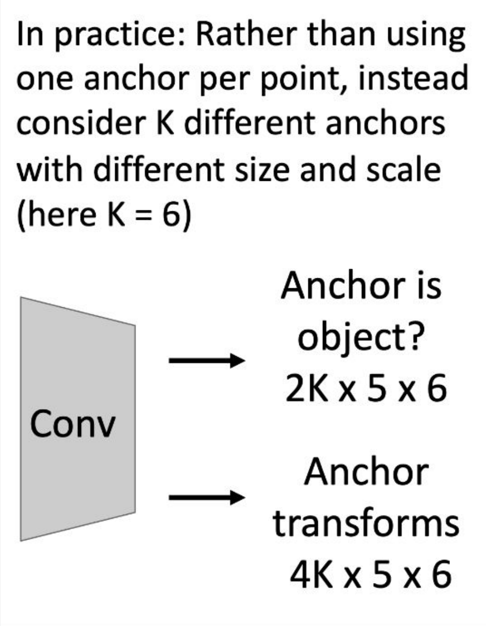
For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)



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

Region Proposal Network (RPN)

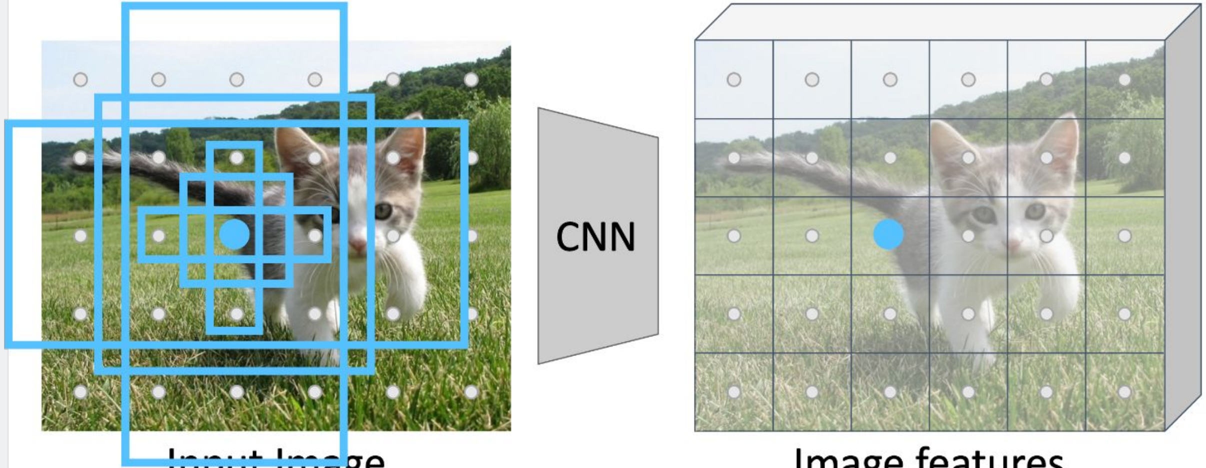




Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

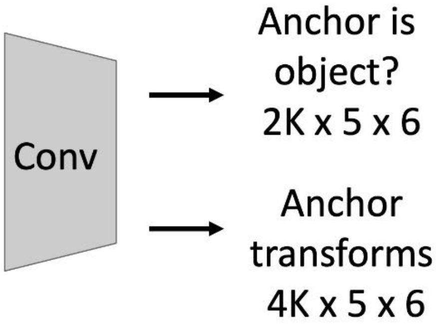
Each feature corresponds to a point in the input



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

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

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

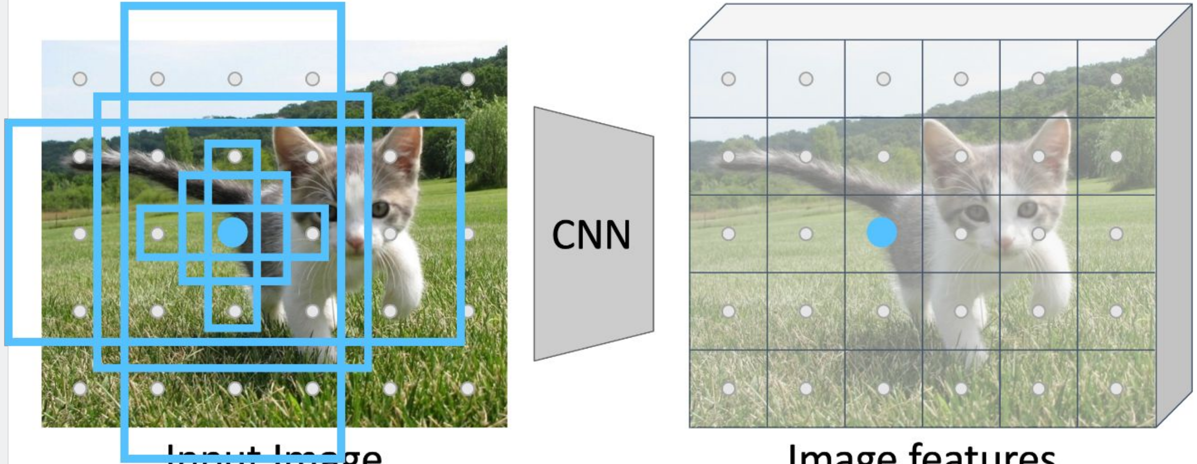


During training, supervised positive / negative anchors and box transforms like R-CNN

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

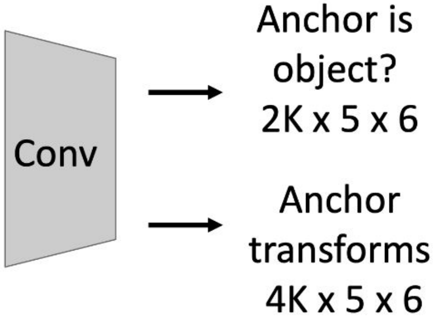
Each feature corresponds to a point in the input



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

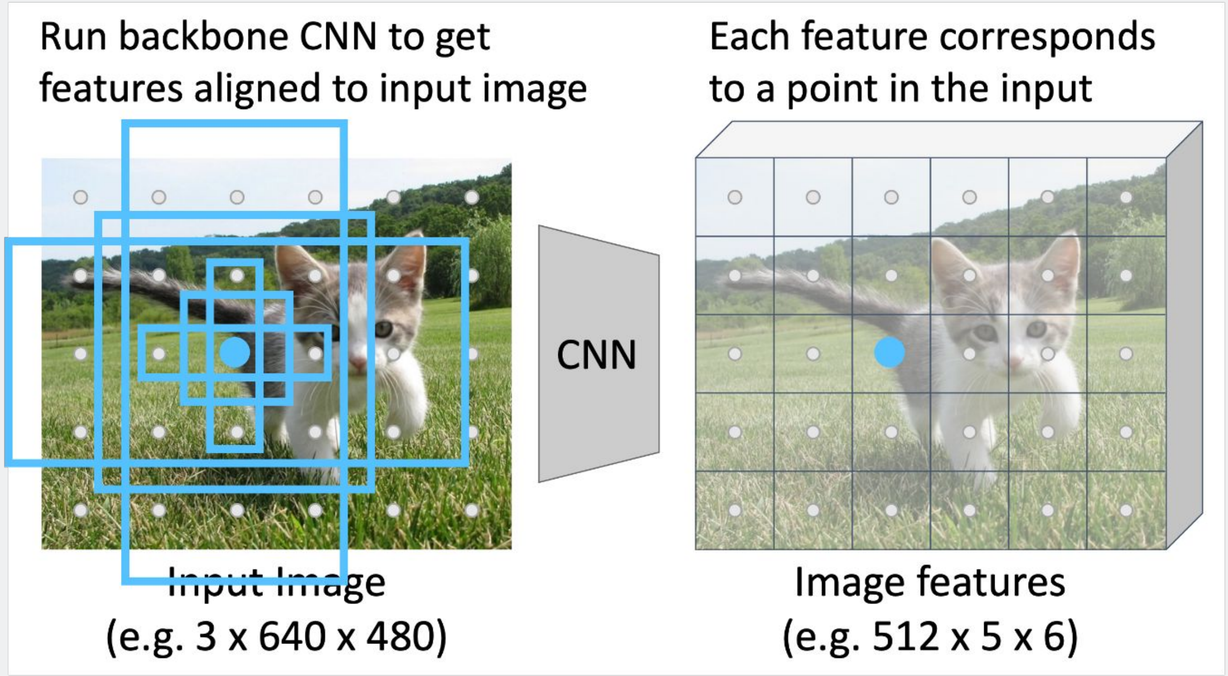
Image features
(e.g. 512 x 5 x 6)

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

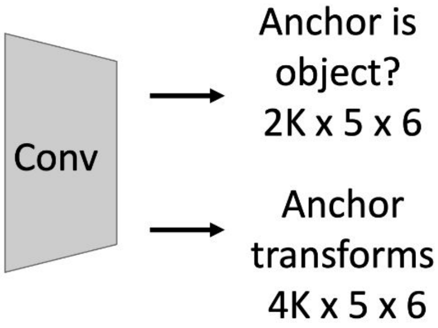


Positive anchors: >= 0.7 IoU with some GT box (plus highest IoU to each GT)

Region Proposal Network (RPN)



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

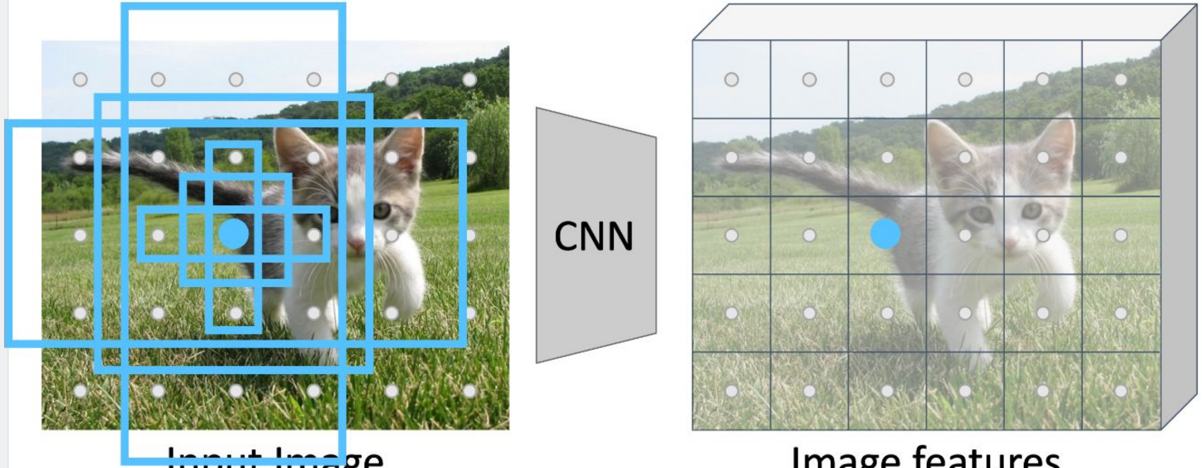


Negative anchors: < 0.3 IoU with all GT boxes. Don't supervised transforms for negative boxes.

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

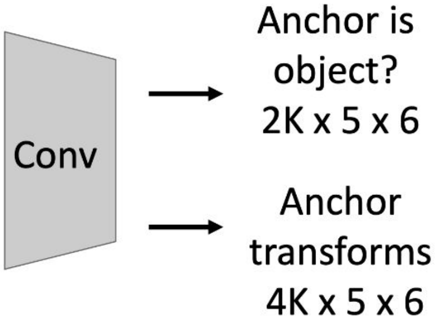
Each feature corresponds to a point in the input



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

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

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

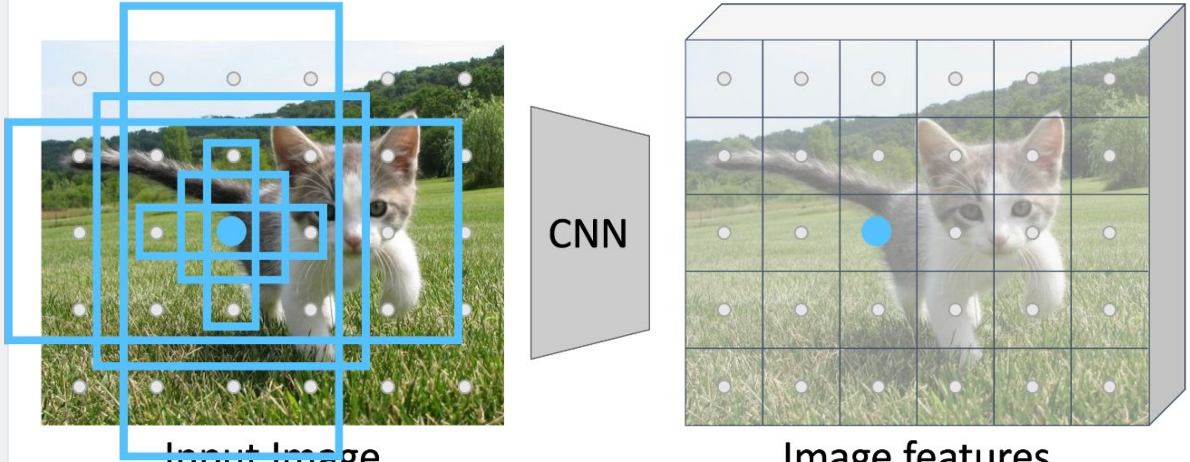


Neutral anchors: between 0.3 and 0.7 IoU with all GT boxes; ignored during training

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

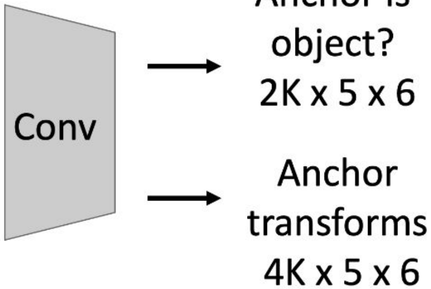
Each feature corresponds to a point in the input



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

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

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

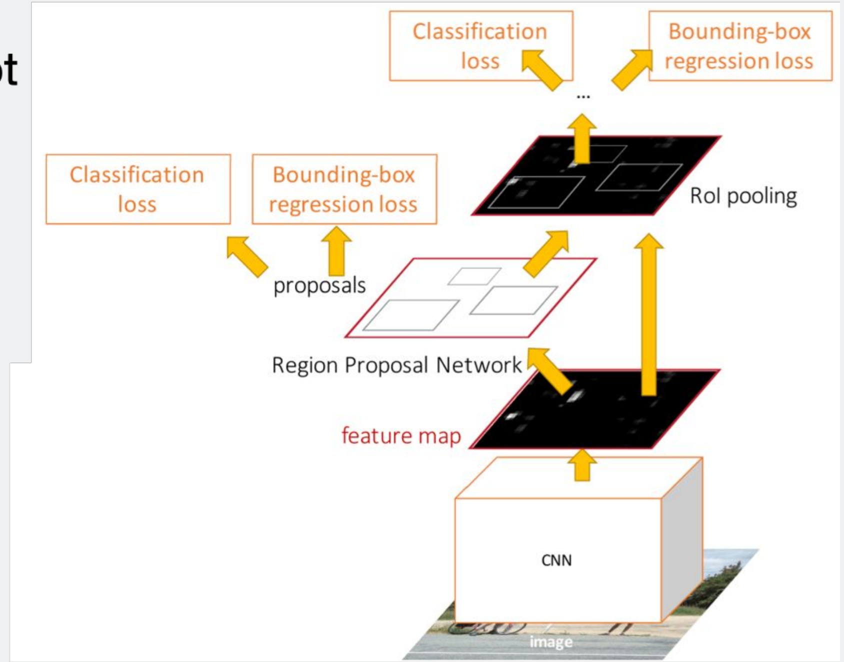


At test-time, sort all K*5*6 boxes by their positive score, take top 300 as our region proposals

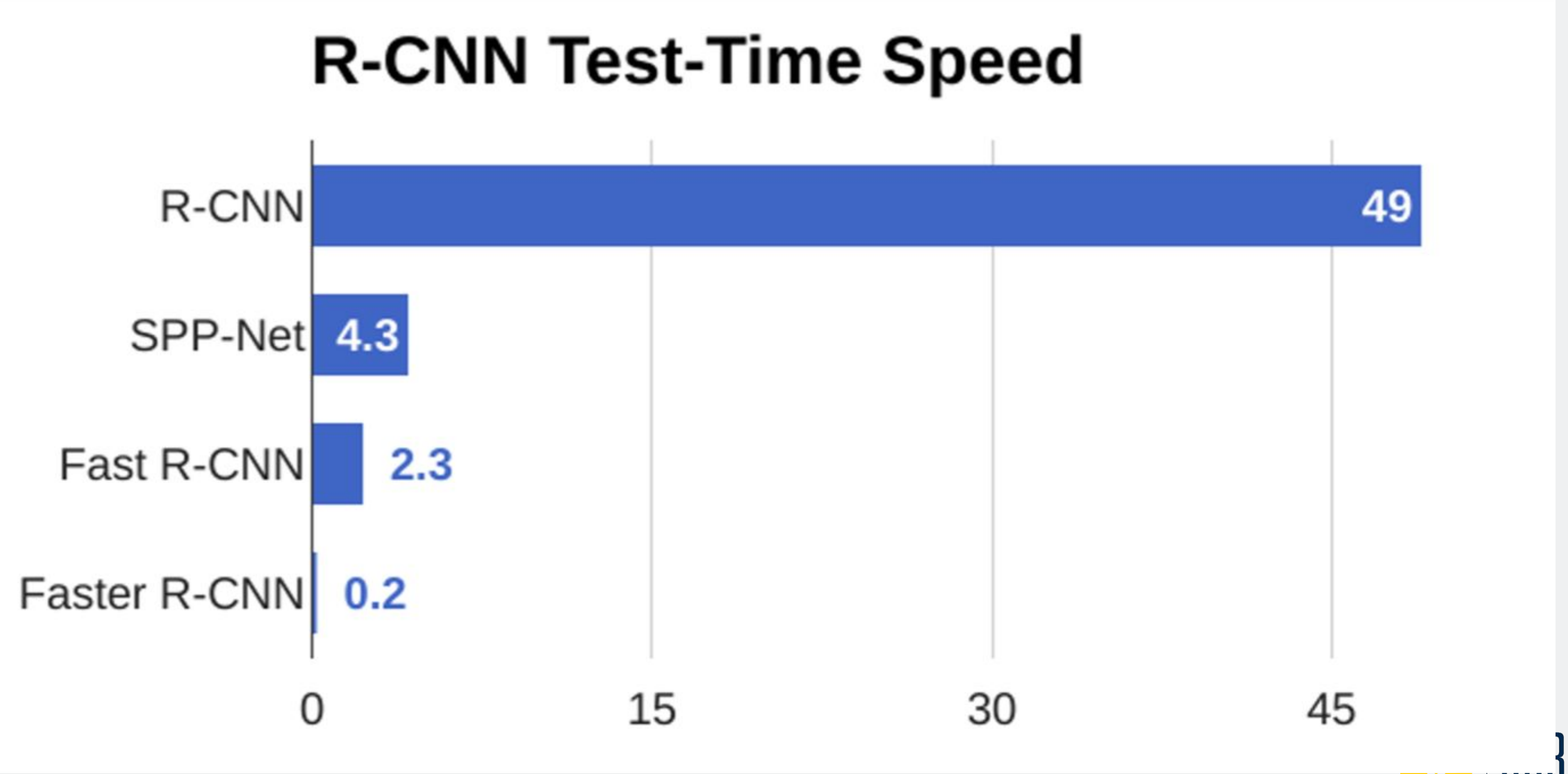
Faster R-CNN: Learnable Region Proposals

Jointly train four losses:

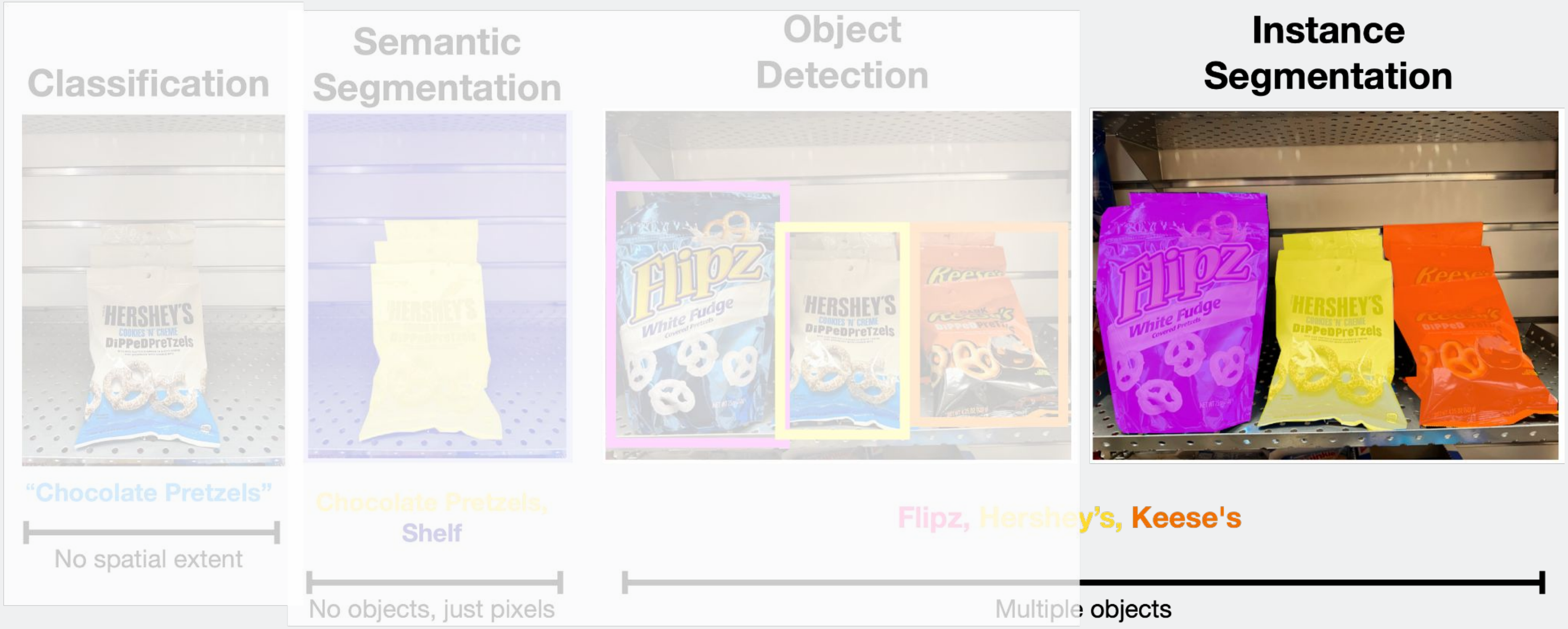
1. **RPN classification:** anchor box is object / not an object
2. **RPN regression:** predict transform from anchor box to proposal box
3. **Object classification:** classify proposals as background / object class
4. **Object regression:** predict transform from proposal box to object box



Faster R-CNN: Learnable Region Proposals



Extend Faster R-CNN to Image Segmentation: Mask R-CNN



Extend Faster R-CNN to Image Segmentation: Mask R-CNN

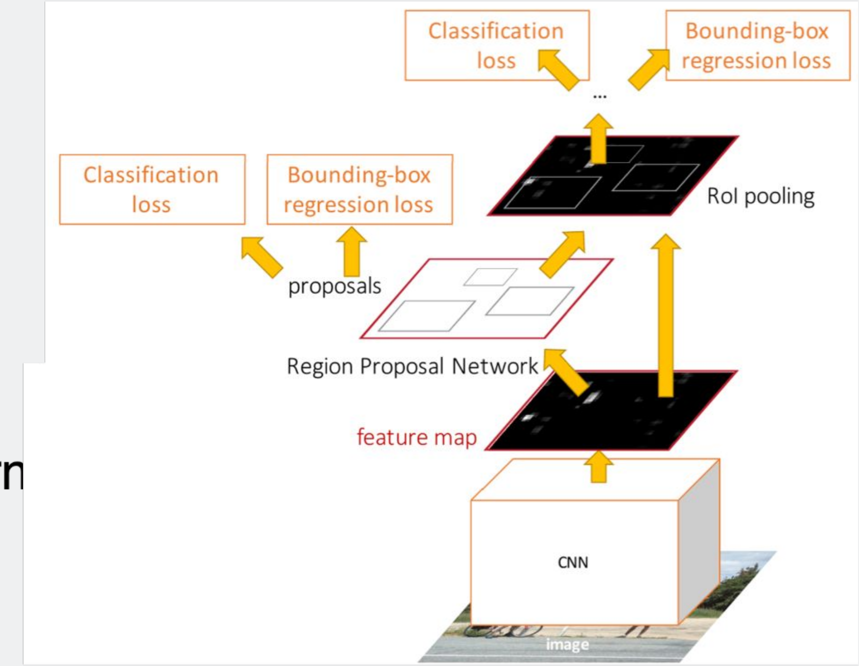
Instance Segmentation
Detect all objects in the image and identify the pixels that belong to each object (Only things!)

Approach
Perform object detection then predict a segmentation mask for each object detected!

Extend Faster R-CNN to Mask R-CNN

Faster R-CNN

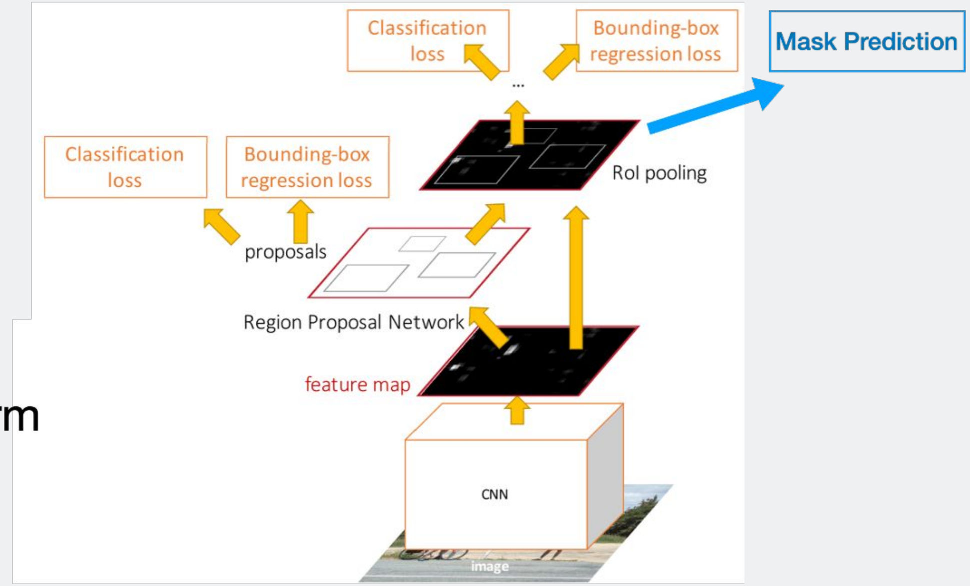
1. **Feature Extraction** at the image-level
2. **Regions of Interest** proposal from feature map
3. **In Parallel**
   1. **Object classification:** classify proposals
   2. **Object regression:** predict transform from proposal box to object box



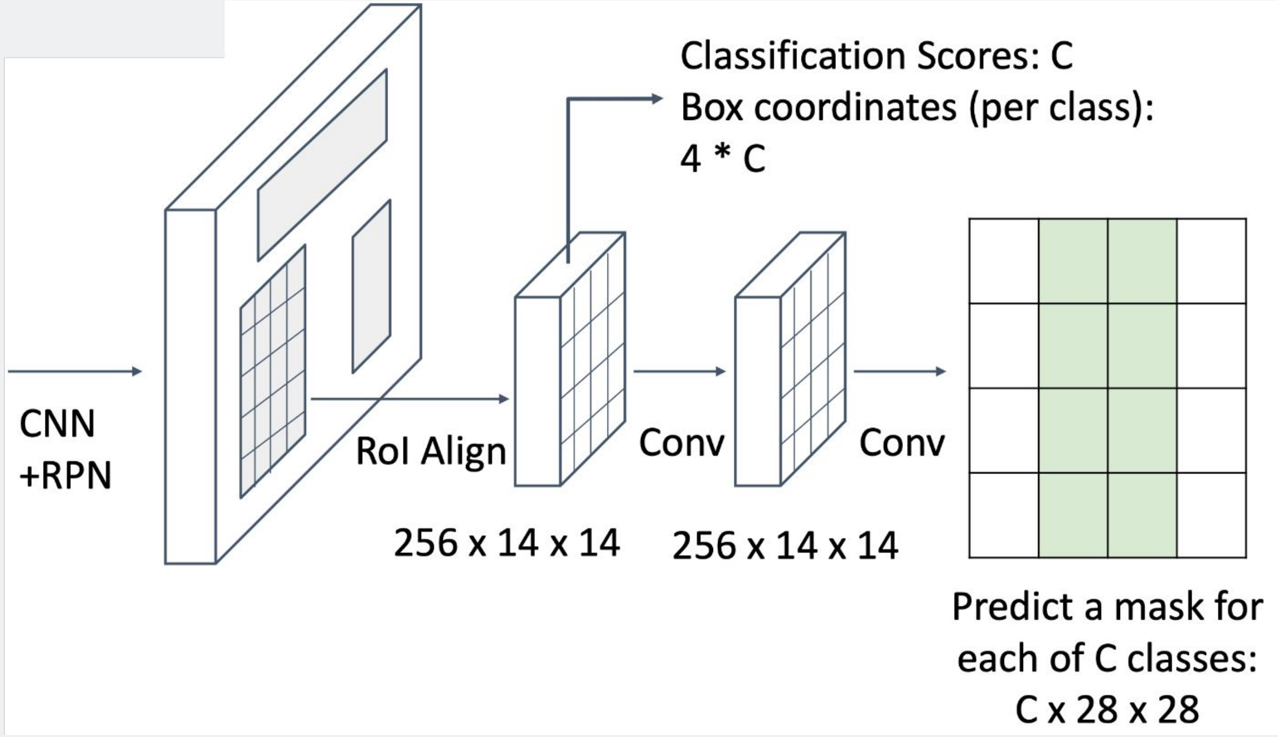
Extend Faster R-CNN to Mask R-CNN

Mask R-CNN

1. **Feature Extraction** at the image-level
2. **Regions of Interest** proposal from feature map
3. **In Parallel**
   a. **Object Classification:** classify proposals
   b. **Object Regression:** predict transform from proposal box to object box
   c. **Mask Prediction:** predict a binary mask for every region



Mask R-CNN



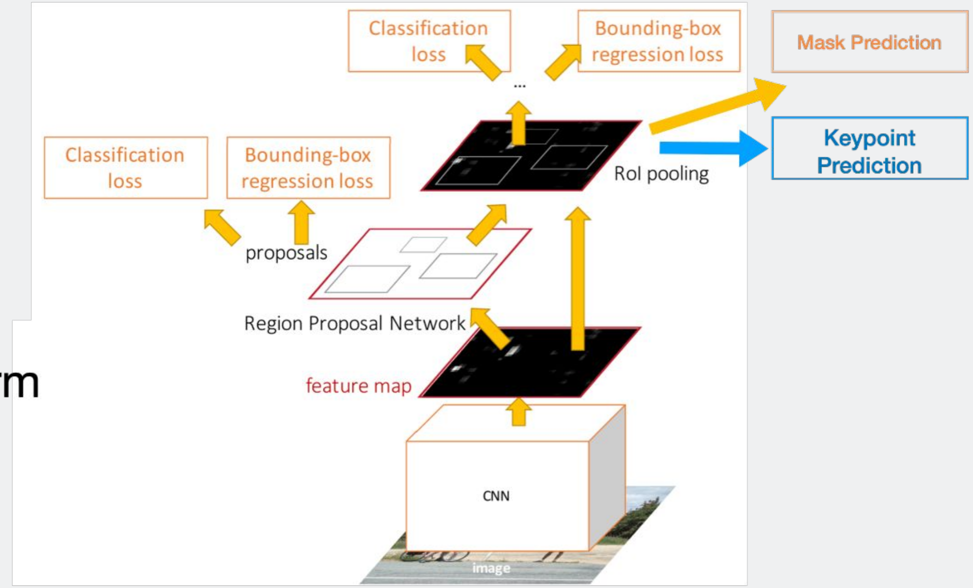
He et al., "Mask R-CNN", ICCV 2017
https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

Mask R-CNN

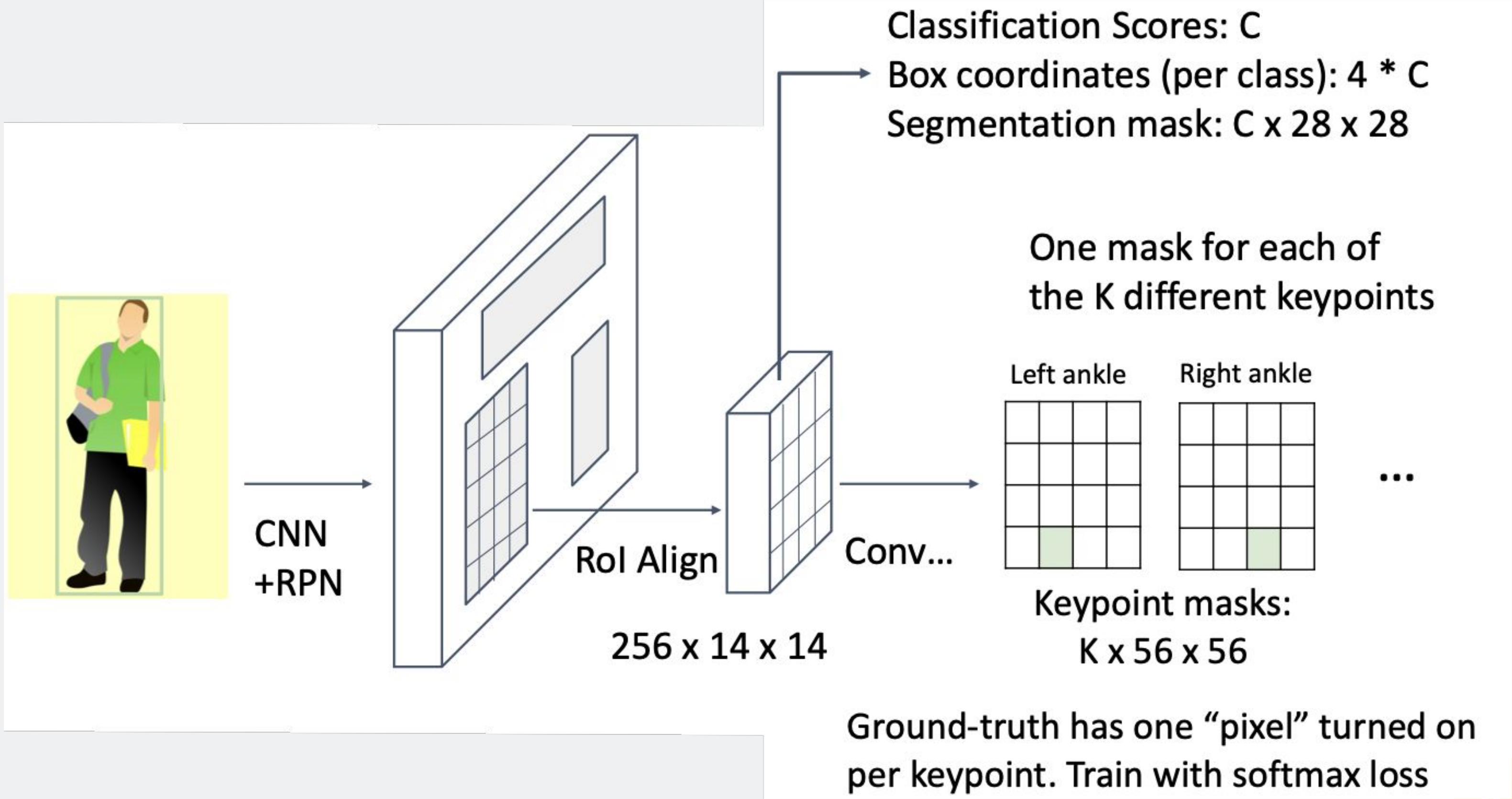
Mask R-CNN for Human Pose Estimation

Mask R-CNN

1. **Feature Extraction** at the image-level
2. **Regions of Interest** proposal from feature map
3. **In Parallel**
   a. **Object Classification:** classify proposals
   b. **Object Regression:** predict transform from proposal box to object box
   c. **Mask Prediction:** predict a binary mask for every region
   d. **Keypoint Prediction:** predict binary mask for human key points



Mask R-CNN for Human Pose Estimation



ROBOTICS

Mask R-CNN for Human Pose Estimation

Detection without Anchors - Transformers

(more details later)

+ No RPN
+ No Anchors
+ No NMS

- Slow to train
- Worse than Faster R-CNN for smaller objects

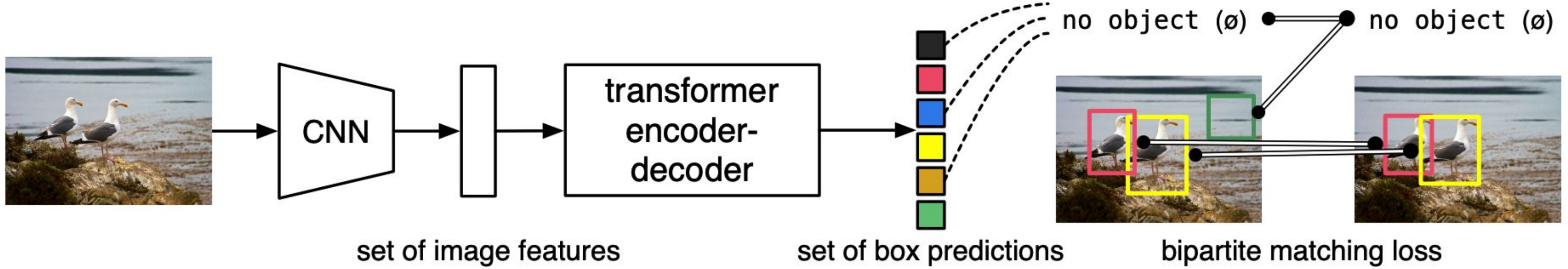


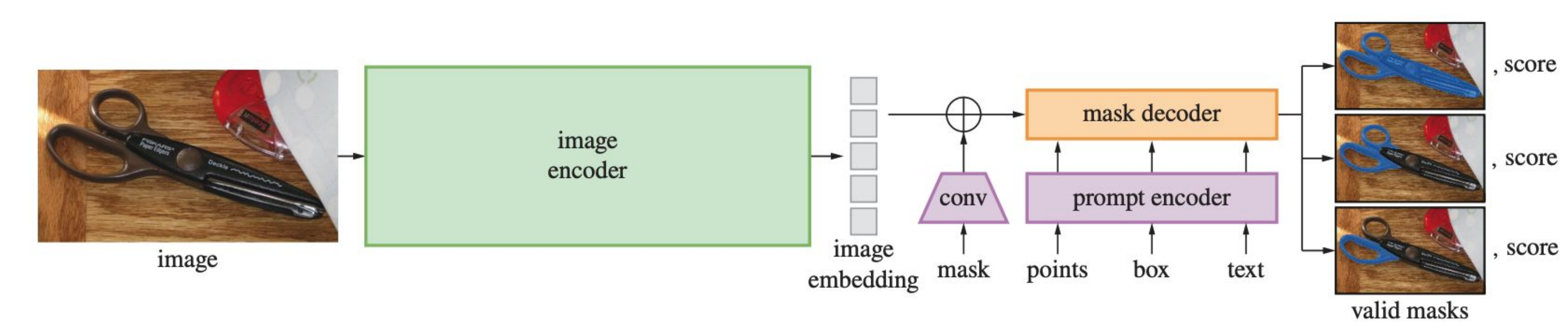
Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "**no object**" (∅) class prediction.

https://arxiv.org/abs/2005.12872

SAM2

Under “in_class” folder
“20250218_discussion_image_predictor_example.ipynb”

https://ai.meta.com/sam2/



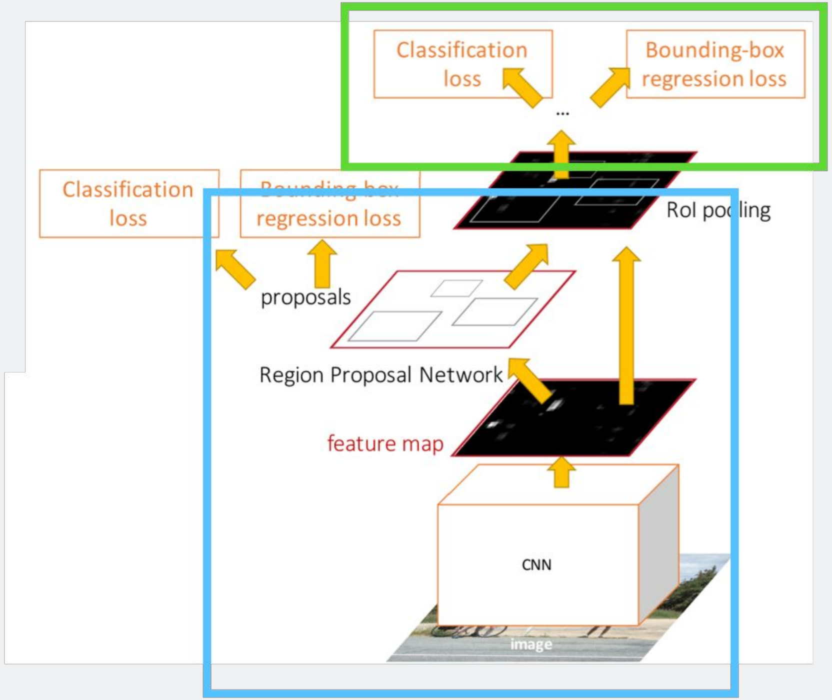
- pre-trained Vision Transformer (ViT)
- NLP prompt

Two Stage vs One Stage Detectors

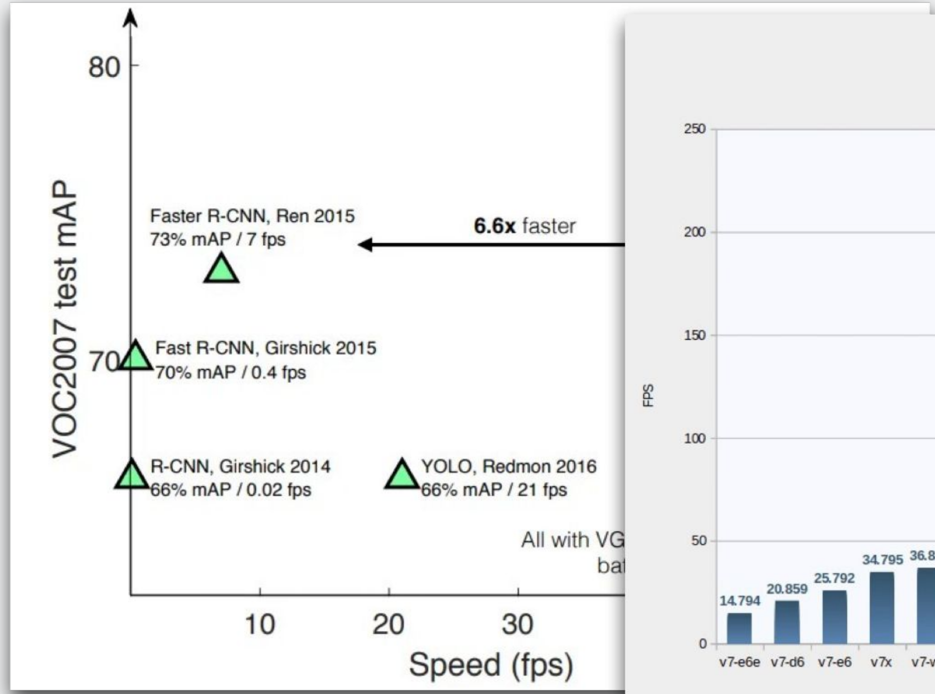
Faster R-CNN is a two-stage object detector

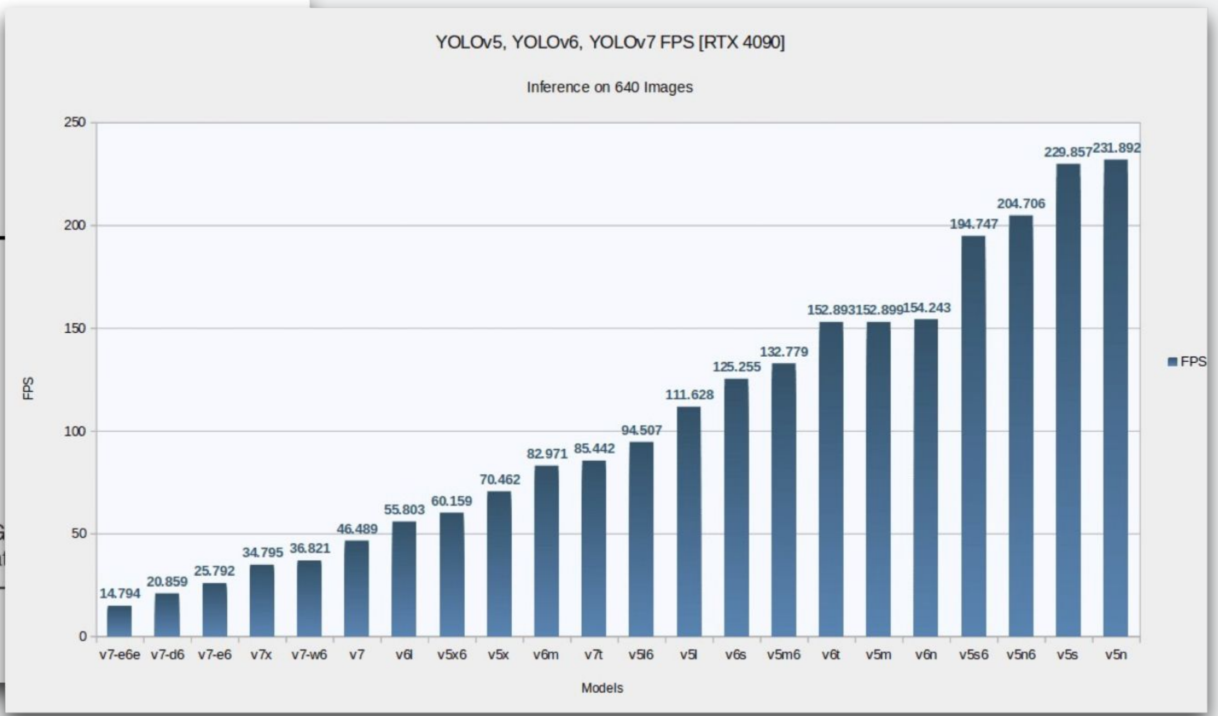
First stage: Run once per image
- Backbone Network
- Region Proposal Network

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



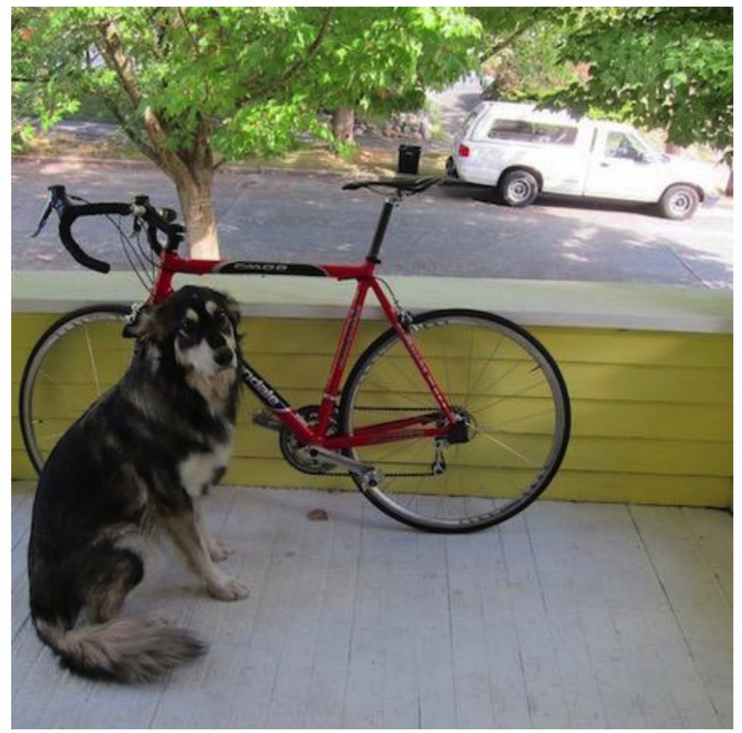
Two Stage vs One Stage Detectors



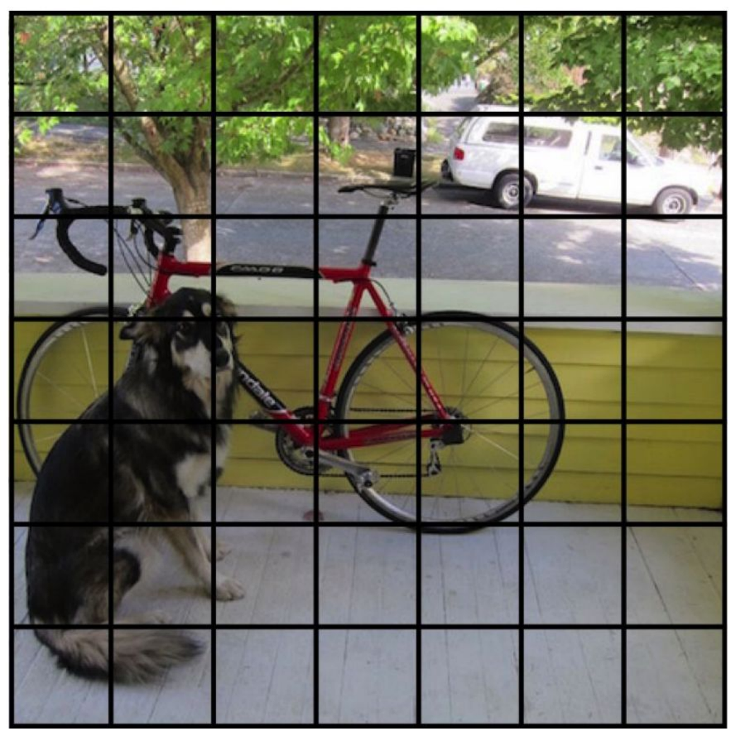


https://pjreddie.com/yolo/

YOLO

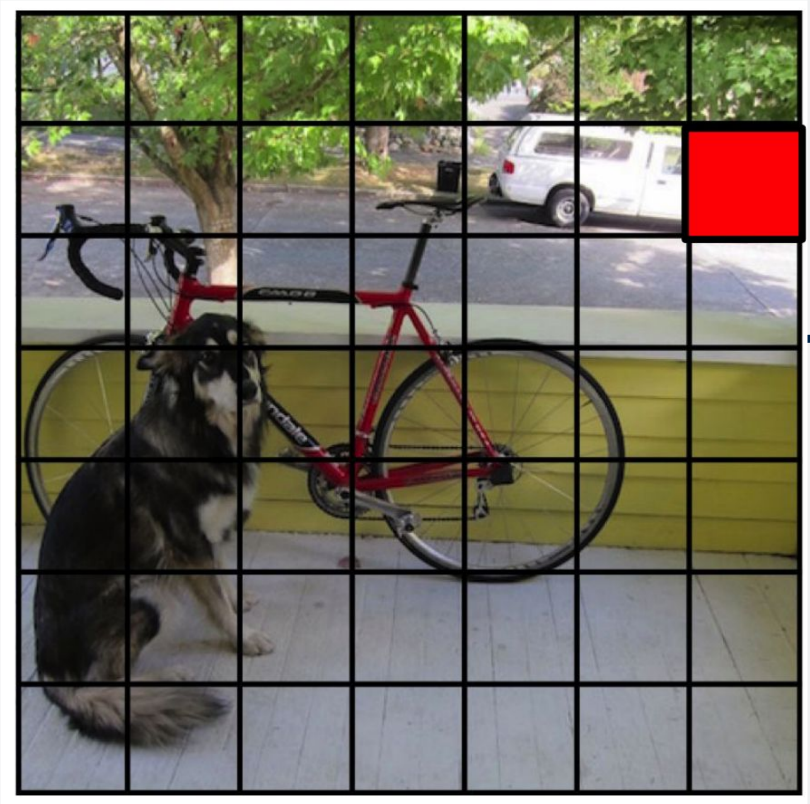


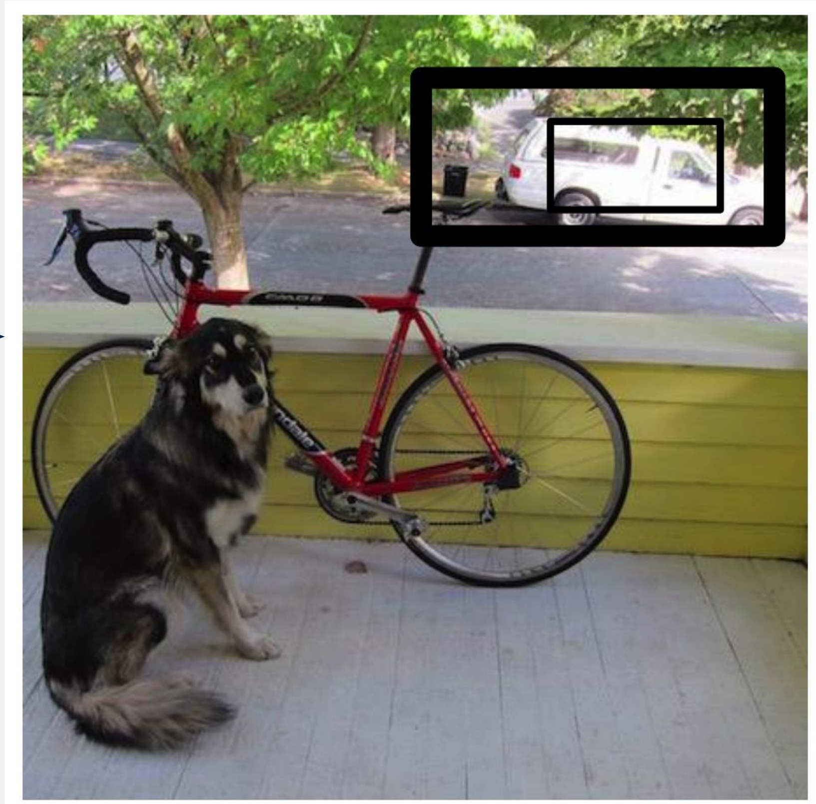
Split into SxS grid



YOLO

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

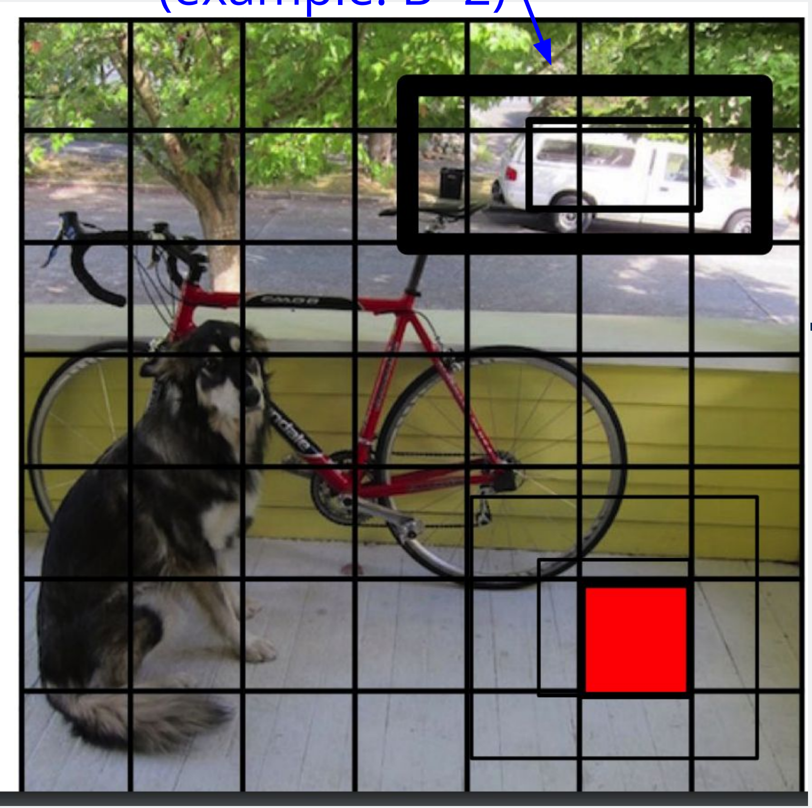




YOLO

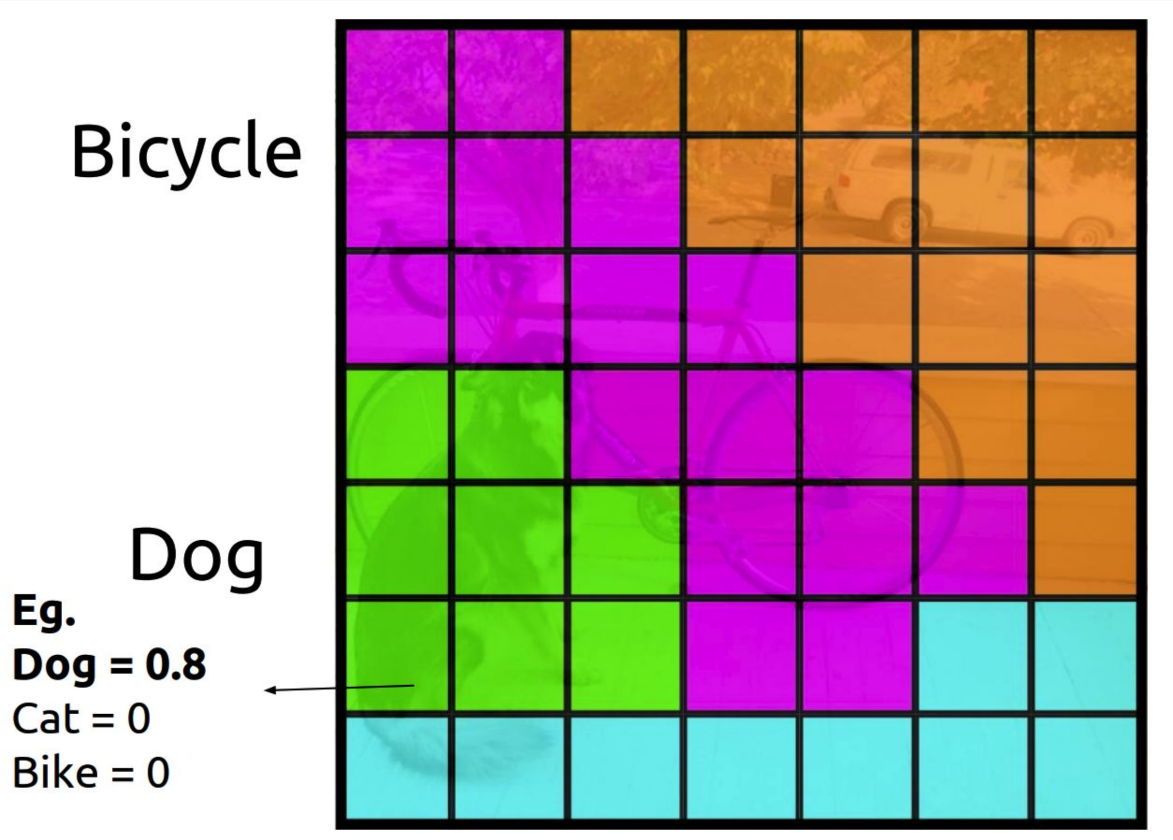
Each cell predicts boxes and confidences

(example: B=2)



YOLO

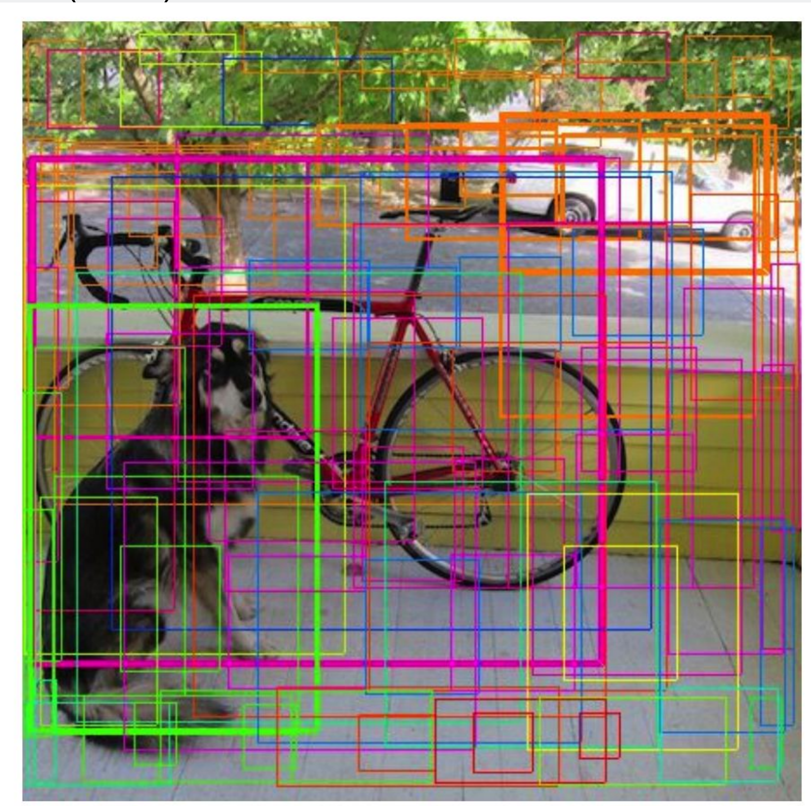
Each cell also predicts a class probability
Conditioned on object: P(Class | Object)



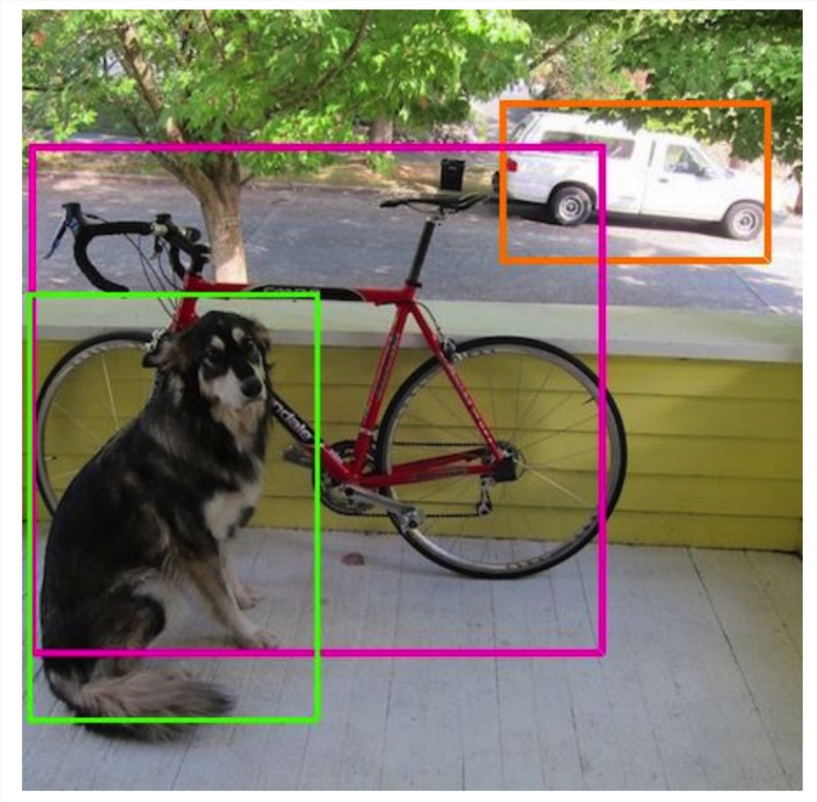
YOLO

Combine boxes and class probabilities
Apply NMS

P(class|Object) * P(Object)
=P(class)



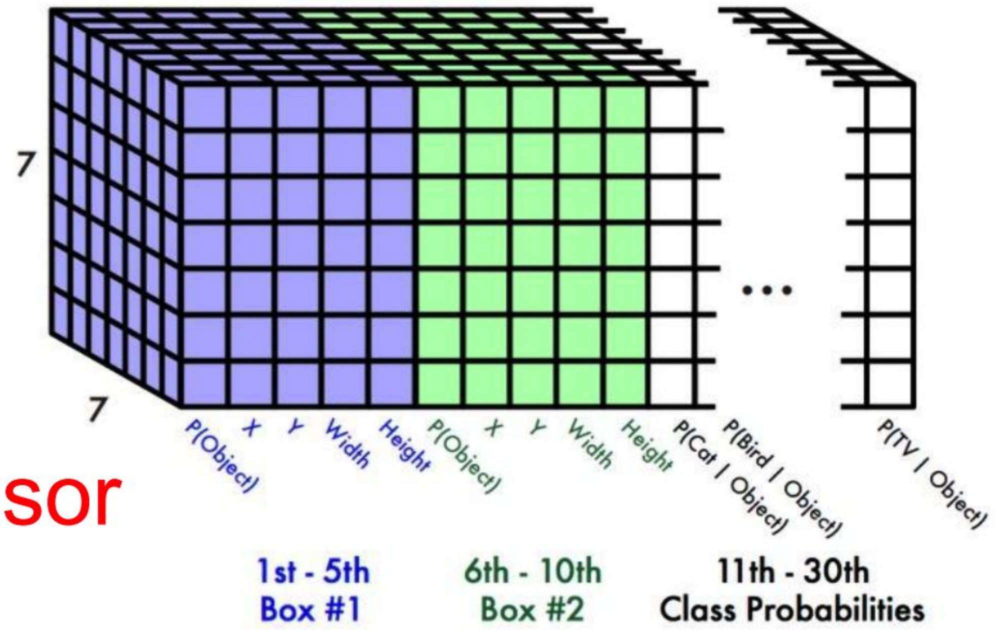
NMS



YOLO

Each cell predicts:

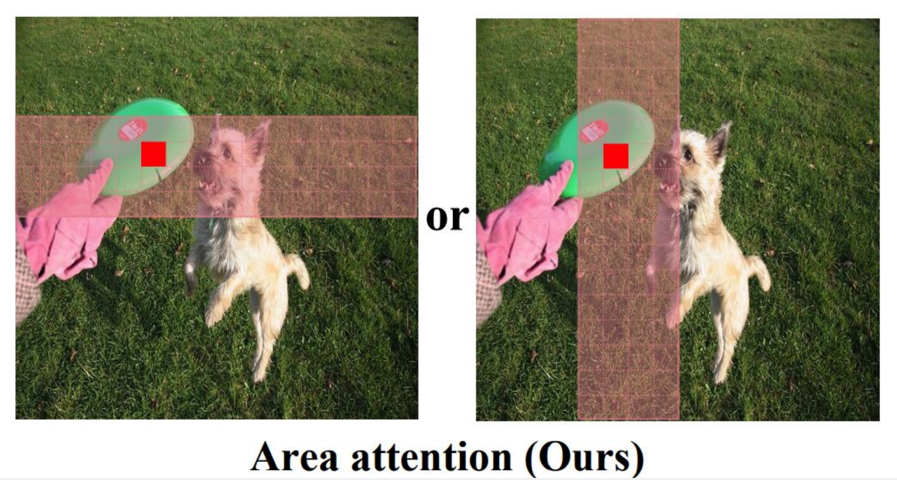
- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

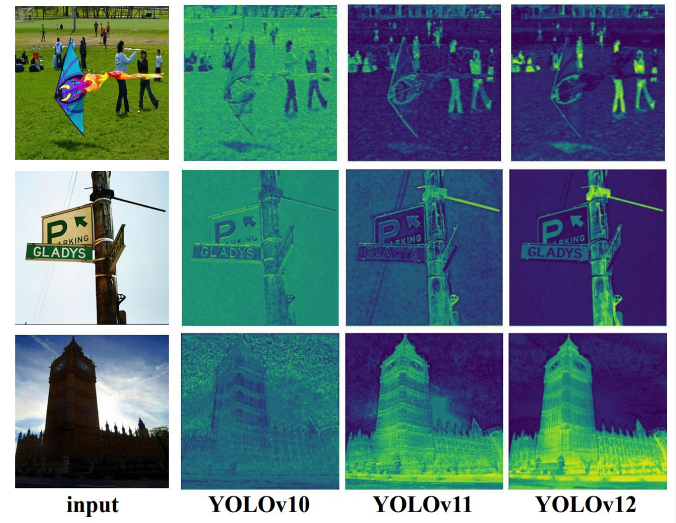


S * S * (B * 5 + C) tensor

YOLO Variants

https://arxiv.org/abs/1506.02640 (original YOLO paper, 2015)
https://arxiv.org/abs/2304.00501 (from YOLOv1 to YOLOv8 and YOLO-NAS, 2023)
https://arxiv.org/abs/2402.13616 (YOLOv9, Feb. 2024)
https://arxiv.org/abs/2405.14458 (YOLOv10, May 2024)
https://arxiv.org/abs/2410.17725 (YOLOv11, Oct. 2024)
https://www.arxiv.org/abs/2502.12524 (**YOLOv12**, Feb.18, 2025)
https://github.com/sunsmarterjie/yolov12 (YOLOv12 pytorch, Attention-centric YOLO)





heatmap comparison

3D Shape Prediction: Mesh R-CNN

Mask R-CNN:
2D Image -> 2D shapes

Mesh R-CNN:
2D image -> 3D triangle meshes

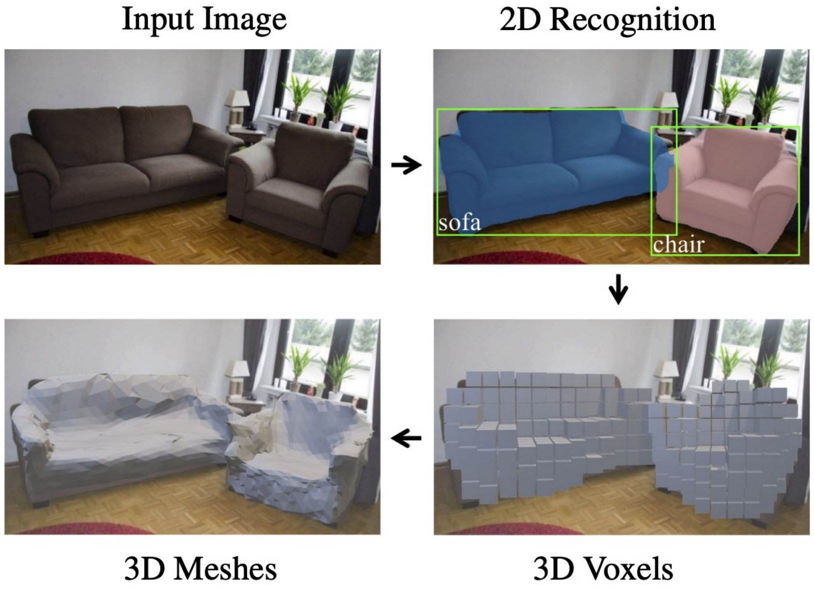


Figure 1. Mesh R-CNN takes an input image, predicts object instances in that image and infers their 3D shape. To capture diversity in geometries and topologies, it first predicts coarse voxels which are refined for accurate mesh predictions.

https://arxiv.org/abs/1906.02739

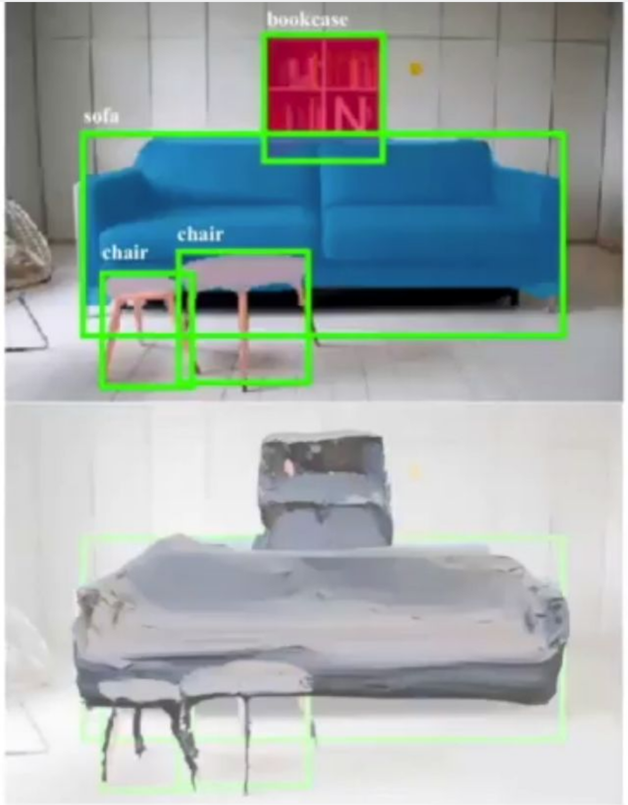
3D Shape Prediction: Mesh R-CNN

Input: Single RGB image

Output:

Mask R-CNN
- A set of detected objects
- For each object:
  - Bounding box
  - Category label
  - Instance segmentation

Mesh head
  - 3D triangle mesh



Attach a customized head that operates on each RoI coming out of Mask R-CNN

3D Shape Prediction: Mesh R-CNN

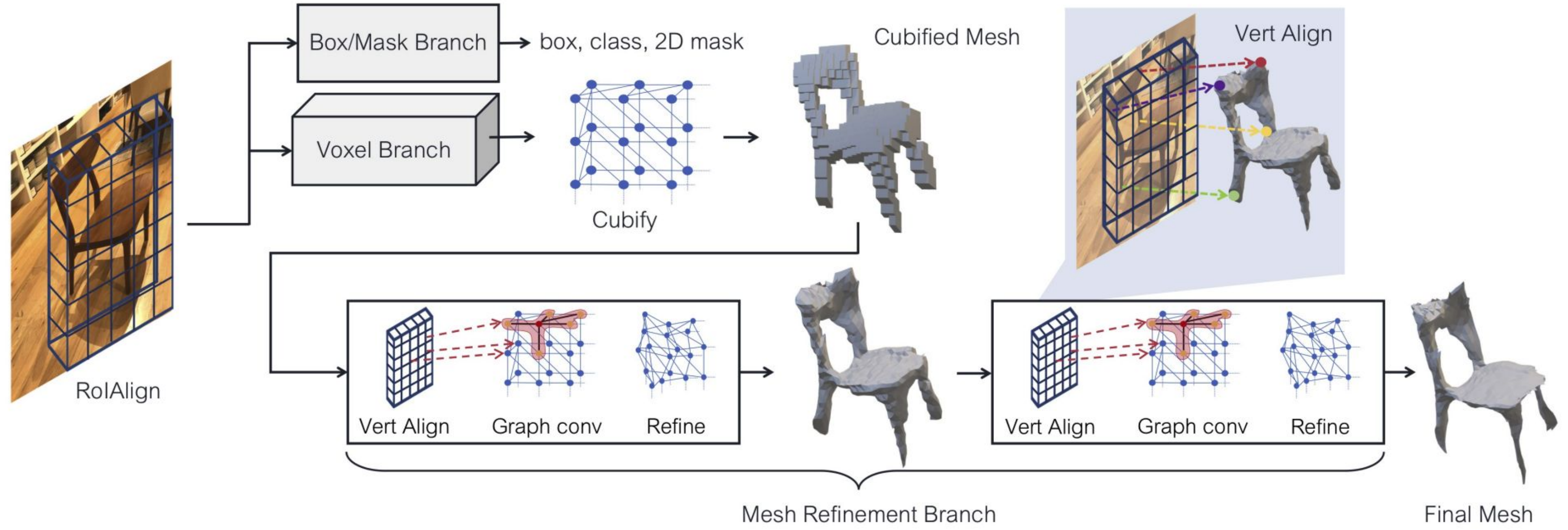
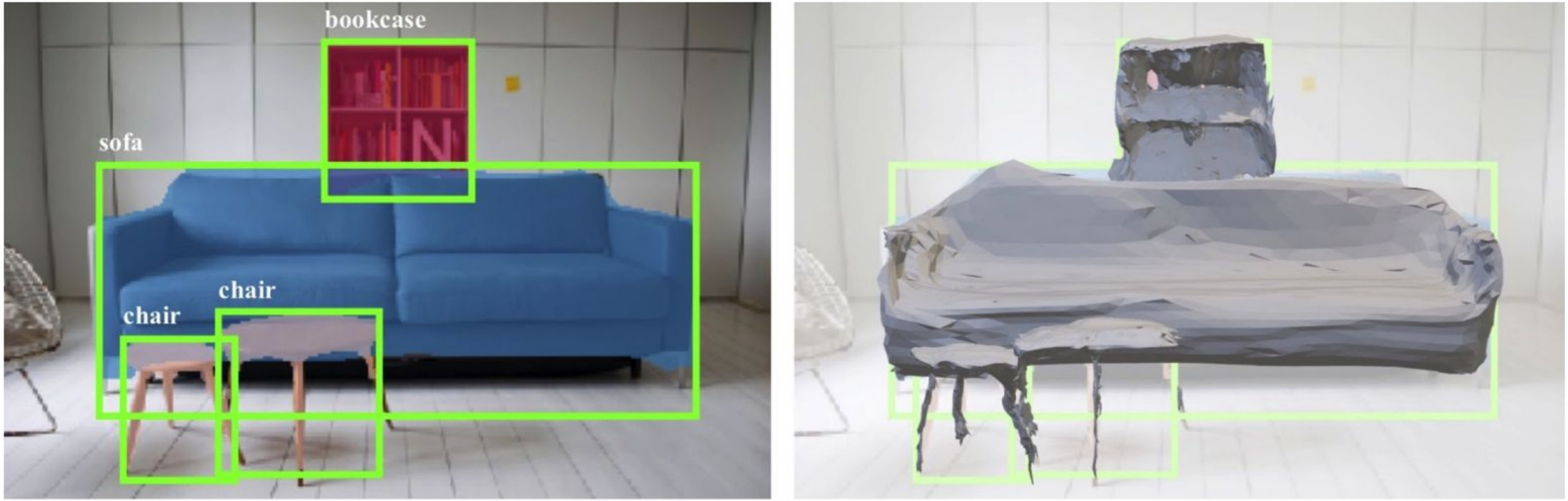


Figure 3. System overview of Mesh R-CNN. We augment Mask R-CNN with 3D shape inference. The *voxel branch* predicts a coarse shape for each detected object which is further deformed with a sequence of refinement stages in the *mesh refinement branch*.

3D Shape Prediction: Mesh R-CNN

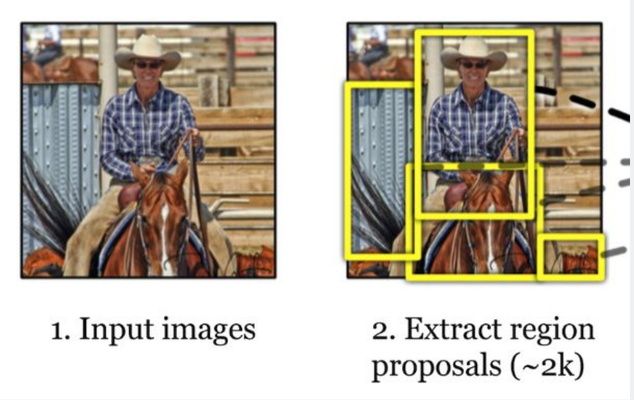


https://proceedings.neurips.cc/paper_files/paper/2024/file/29c8c615b3187ee995029284702d3f43-Paper-Conference.pdf (2024 NIPS, 3D Scene Diffusion)

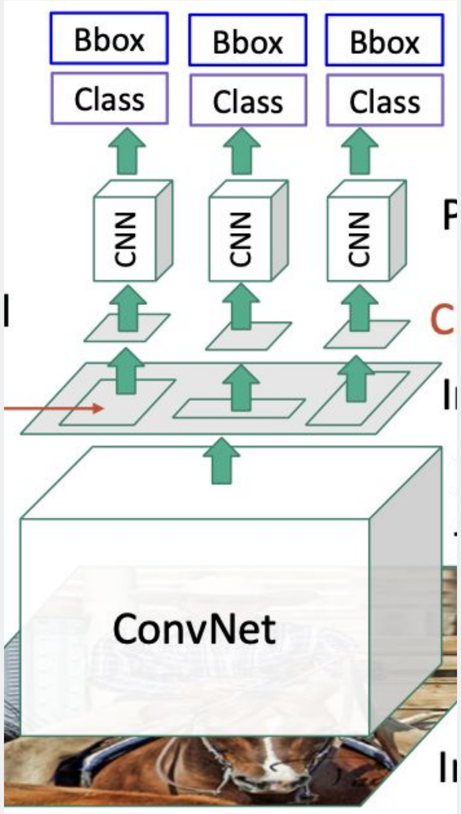
Summary

- **P3 released, Due March 9, 2025 Start NOW!!!**
- **Also, Canvas Quizzes**

"R-CNN Family"



Region proposals



ROI Pool
ROI Align

RPNs

Instance Segmentation