DeepRob

Lecture 12
Object Detectors and Segmentation
University of Minnesota

Project 2—Due today

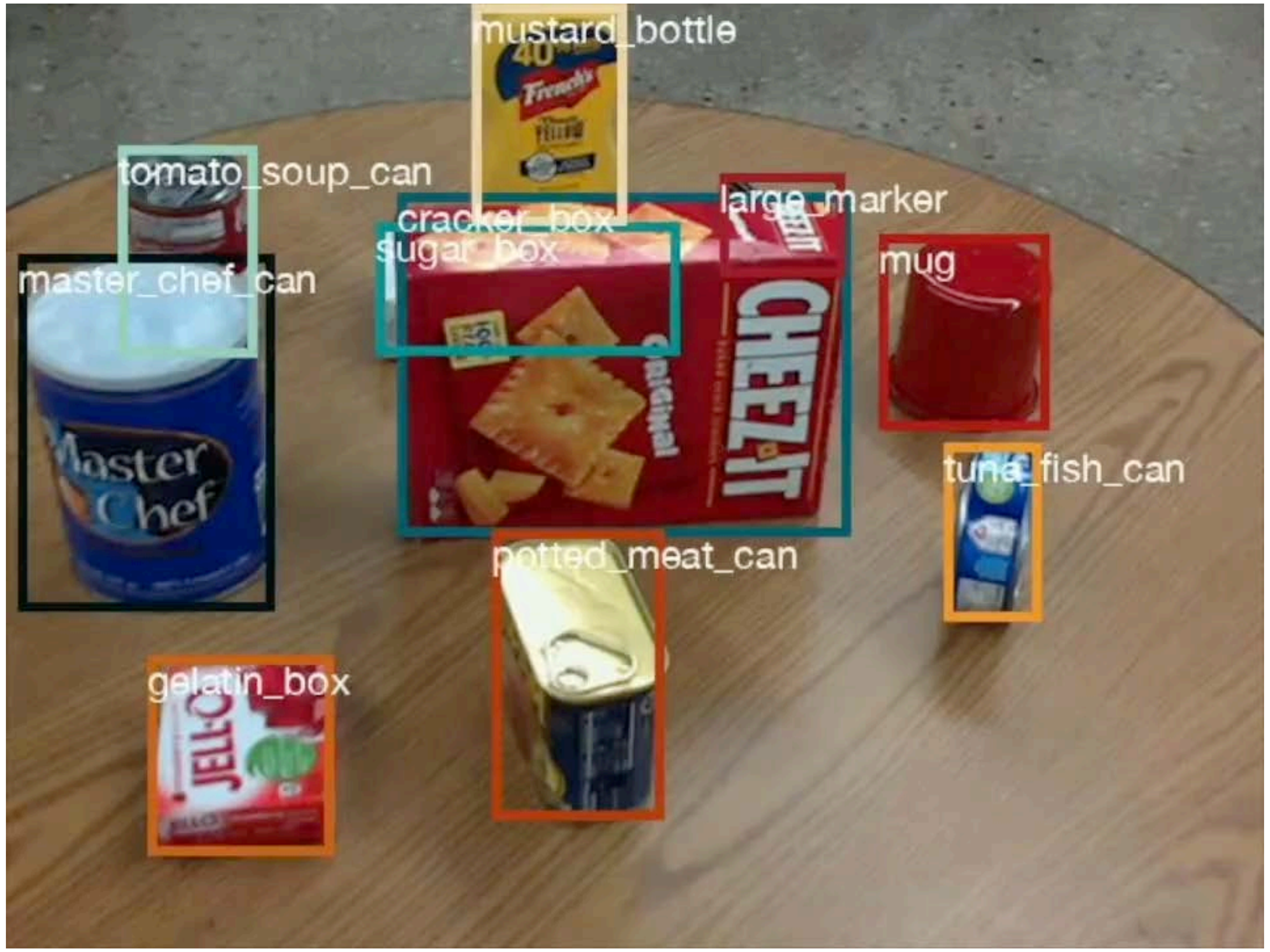
- Instructions available on the website
  - Here: https://rpm-lab.github.io/CSCI5980-F24-DeepRob/projects/project2/

- Implement two-layer neural network and generalize to FCN

- **Due Monday, October 14th, 11:59 PM CT**

Project 3 — Releases today

- Instructions available on the website
  - Here: https://rpm-lab.github.io/CSCI5980-F24-DeepRob/projects/project3/
  - Uses PROPS Detection dataset
- Implement CNN for classification and Faster R-CNN for detection
- Autograder will be available soon!
- **Due Monday, October 28th 11:59 PM CT**



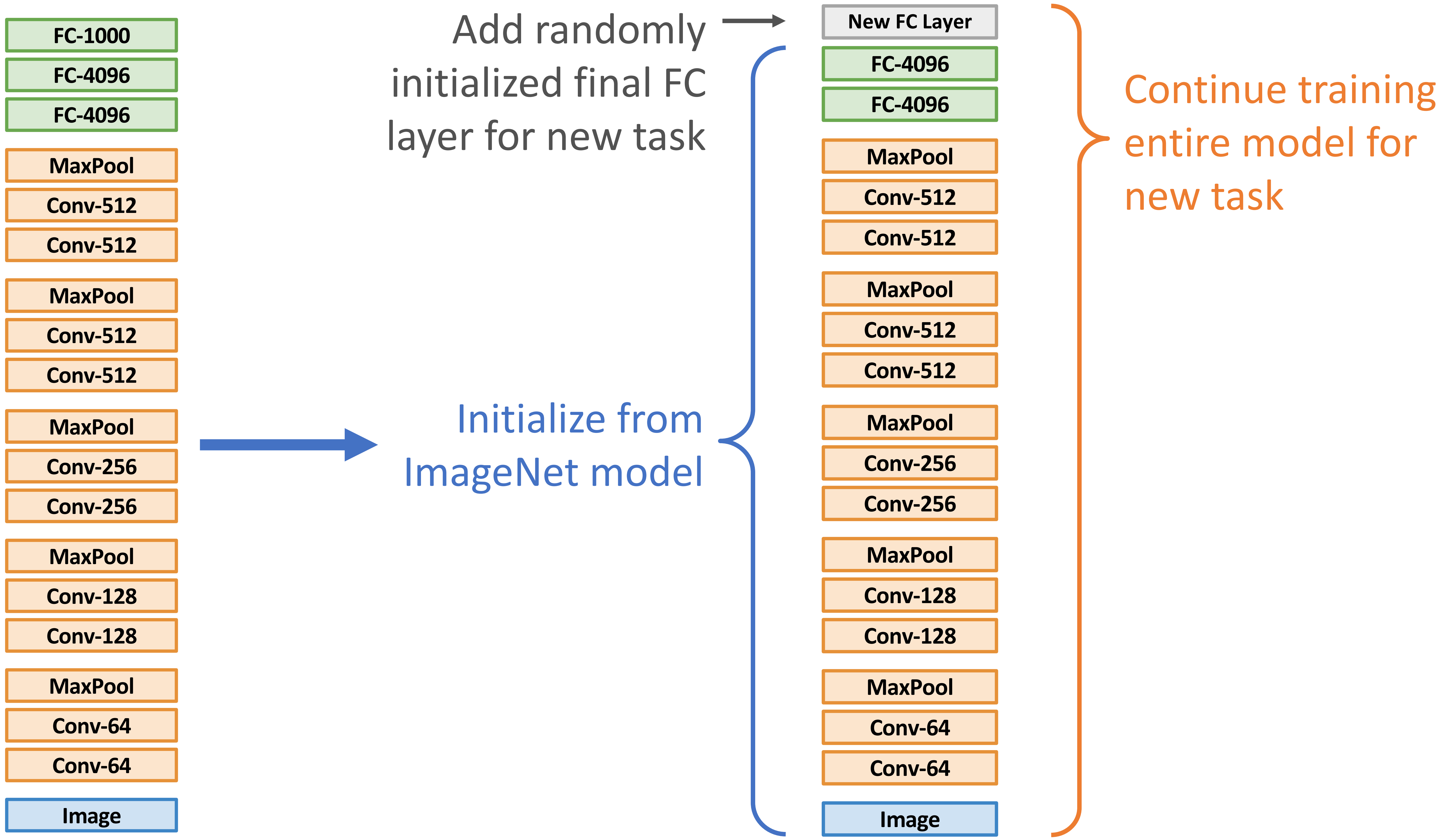
Final Project Proposal—Due Wednesday

DeepRob: What is a Project Proposal?

- Create a 3 page proposal on a google-doc/overleaf 🔥 Please keep the proposal excluding the references to not more than 3 pages.
- Should contain a title that describes the project (keep it simple)
- Should contain full name(s), email addresses of the team.
- Should contain the following sections.
  - **Objective** - What capability does this project aim to give a robot? For example you should be able to say - *"This project aims to impart the capability of .... to the robot. Given a observation in the form of ...., the robot will be able to do ....."*
  - **Input-Output during Inference time** - What are the input and output variables of the system you are building? For example you should be able to say - *"The robot/model takes in RGBD observation I of size $H \times W \times 3$, gripper pose $G \in SE(3)$ and produces action $A \in SE(3)$ ...."*
  - **Method** - What is the algorithm, pipeline, or neural network architecture you are proposing to develop the capability? If it has an algorithm, please describe it. If it is a neural network architecture, describe it. If it is a learning method, what is the training objective, what are the loss functions you will experiment on.
    - **Illustrative figure** - can help quickly understand the method being proposed and the big idea.
  - **Data collection** - Assuming that all the projects in this course is data-driven, where does the data for your project come from (existing datasets, or simulation env) ? Are you going to collect new data?
  - **Evaluation** - How will you evaluate if your method worked? What will you compare with? What is the measure of success?
  - **Resources** - What will be the resources you will use for this project? Is this your desktop or laptop? MSI? Are you using a real-robot setup? If yes, describe the setup. Are you using simulation environment? If yes, describe the setup.
  - **Timeline** - Please plan a weekly schedule and things to accomplish on a weekly basis to successfully finish the project. Do you due diligence to consider other commitments in your semester while creating this timeline for everyone in the group. Discuss this timeline in detail with other members of the group to ensure success. You can tabulate this.
    - Week 10/21-10/25 - Task [member1] - Task[member2] ....
    - Week 10/28-11/01 - Task [member1] - Task[member2] ....
    - ...
  - **Deliverables** - What do you plan to deliver at the end of the project time? Real-world demo? Code for others to use? Make this a technical paper?
  - **Summary of 3 papers** - Please read 3 papers as a group and summarize them with relation to your project. How will you use the techniques from this paper in your work?
  - **References** - Please include any reference material (papers, code, datasets) that you found online that is relevant to your project. This includes all the images you use requires a source citation.
- Please check the grammar or spelling mistakes.
- An upload link will be made available for the submission.

Last time: Transfer Learning

1. Train on ImageNet



Last time: Localization Tasks

Classification

"Chocolate Pretzels"

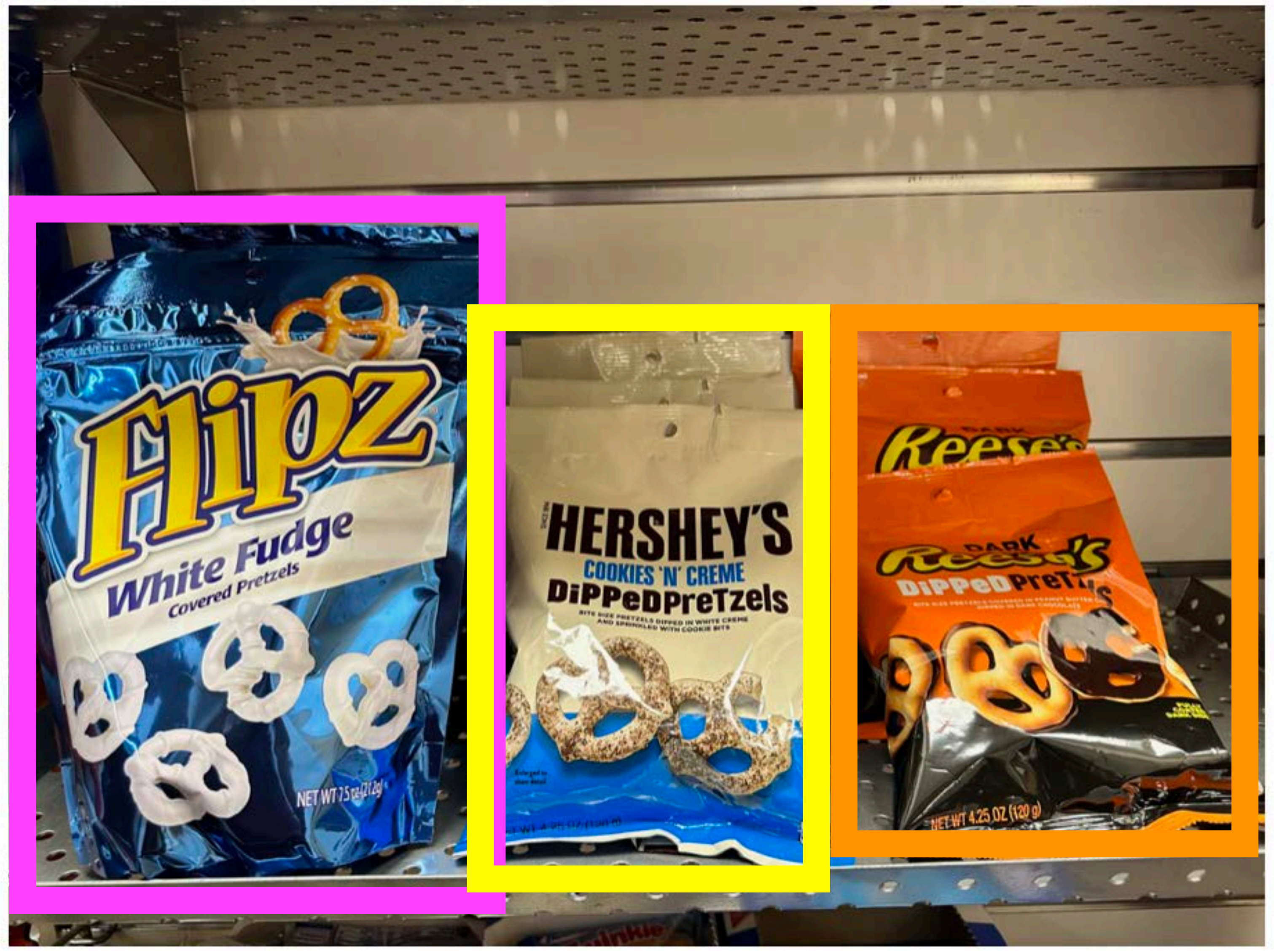
No spatial extent

Semantic Segmentation

Chocolate Pretzels, Shelf

No objects, just pixels

Object Detection



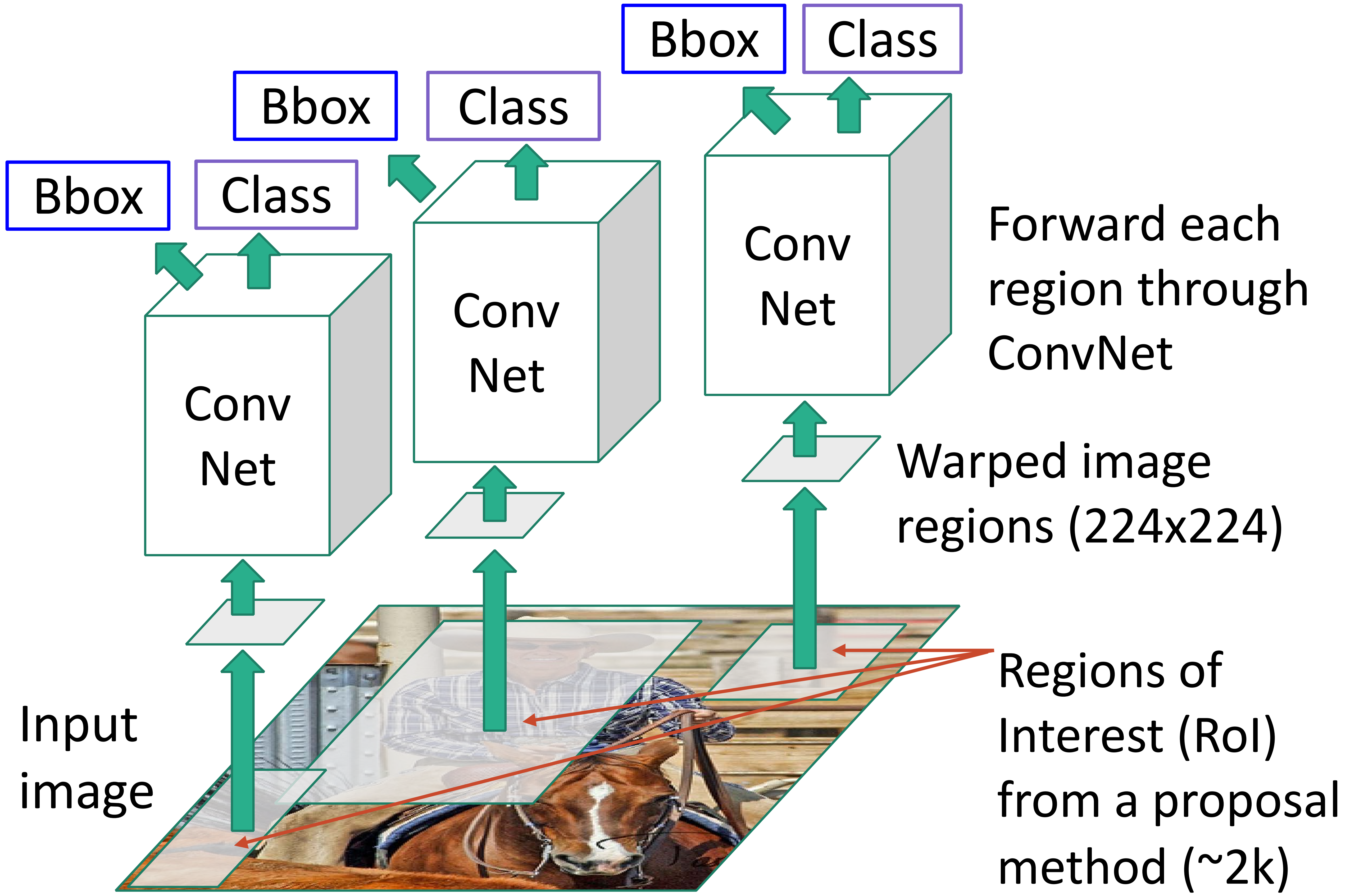
Instance Segmentation

Flipz, Hershey's, Keese's

Multiple objects

Last time: R-CNN

R-CNN: Region-Based CNN

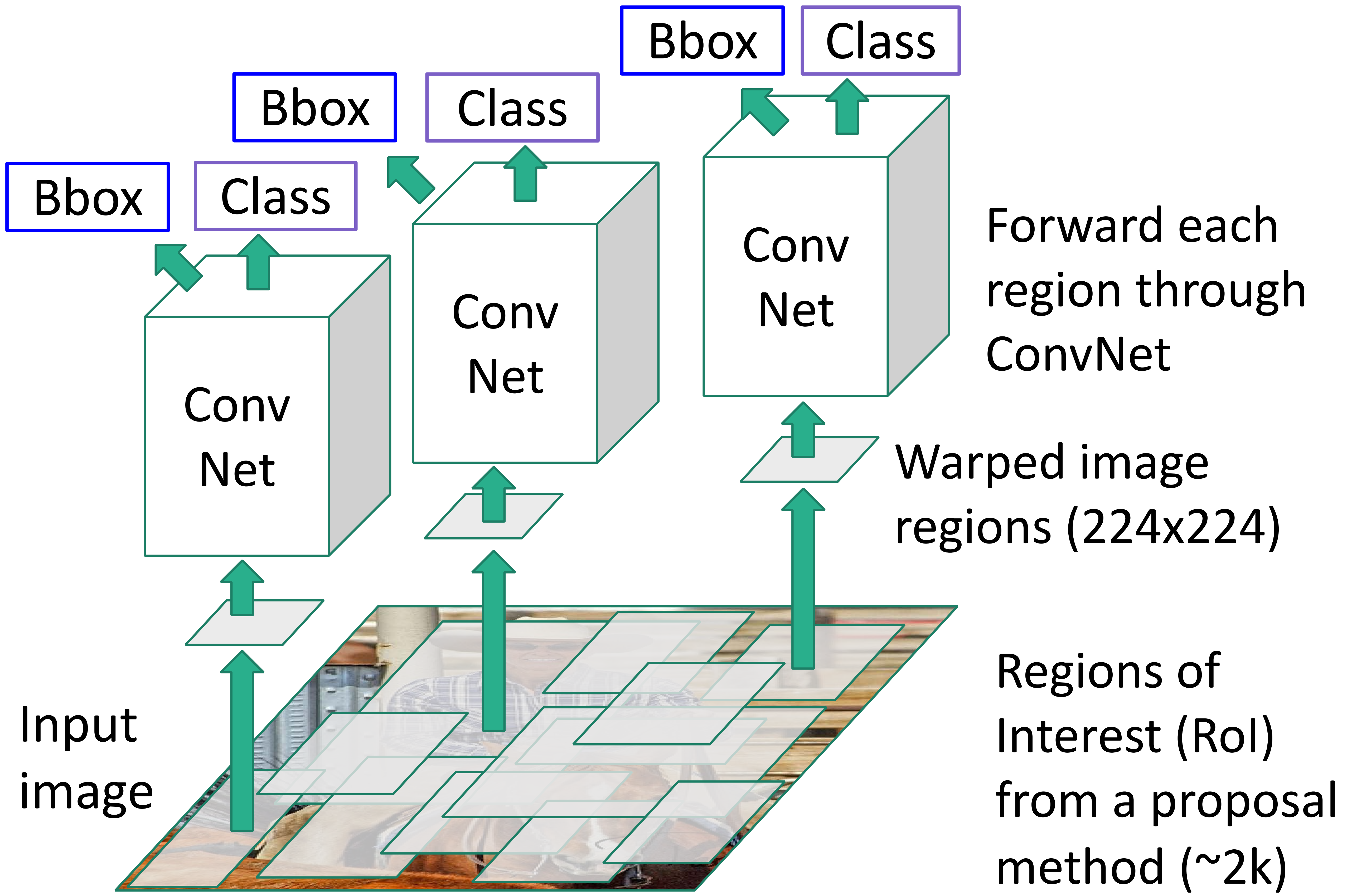


Classify each region

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

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

Last time: R-CNN



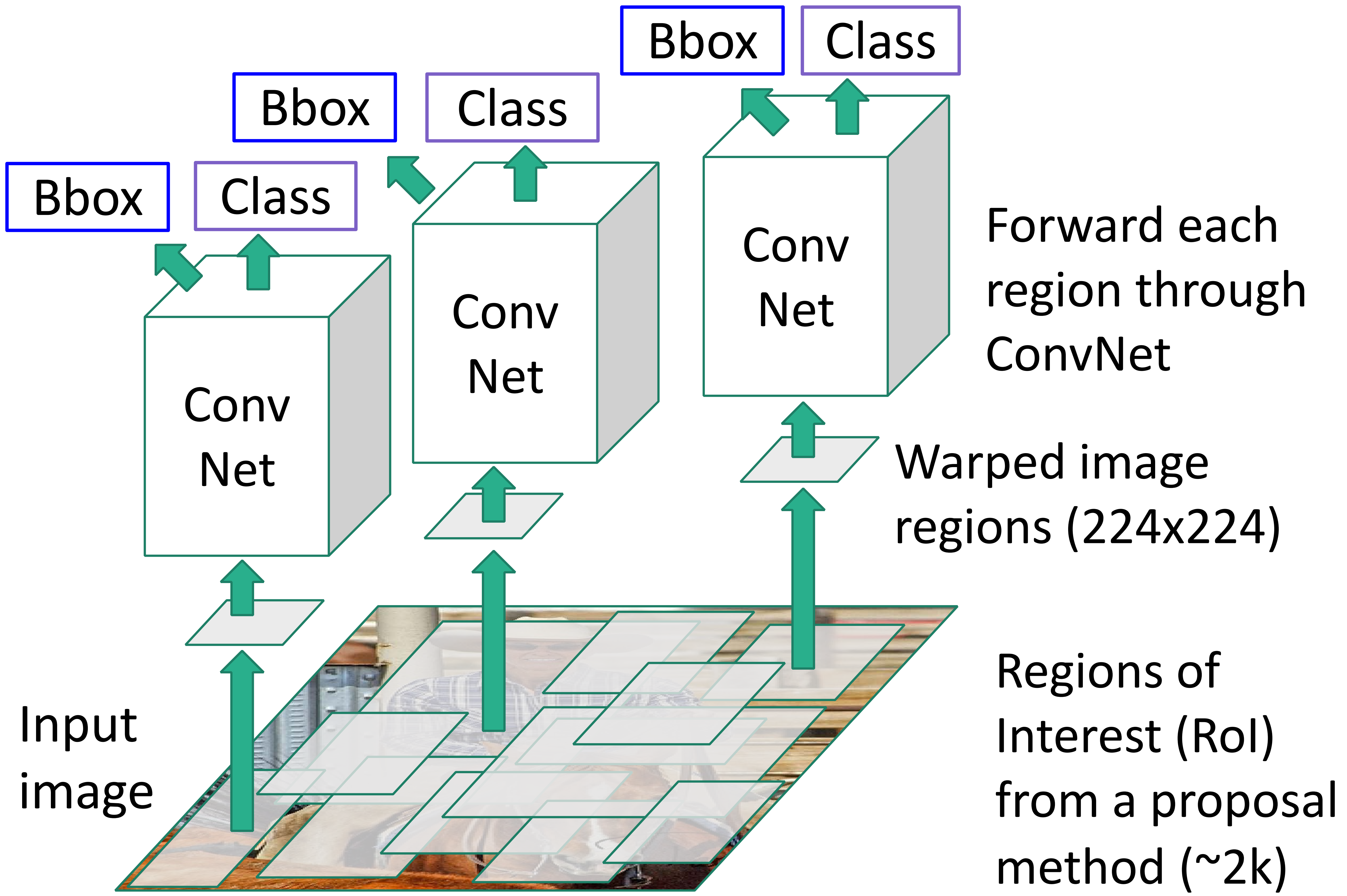
Classify each region

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

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

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

Last time: R-CNN



Classify each region

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

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

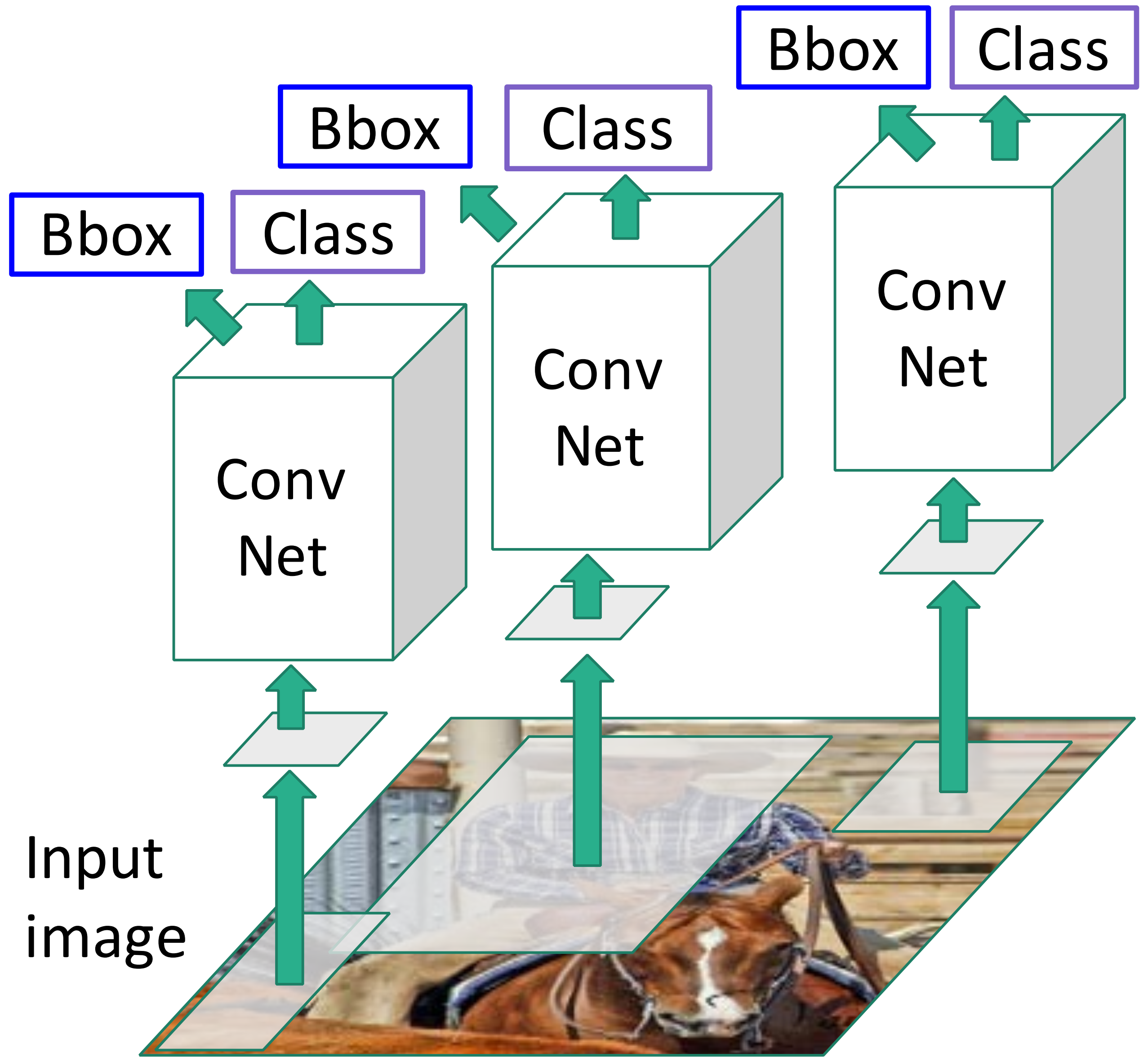
Idea: Overlapping proposals cause a lot of repeated work; same pixels processed many times. Can we avoid this?

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

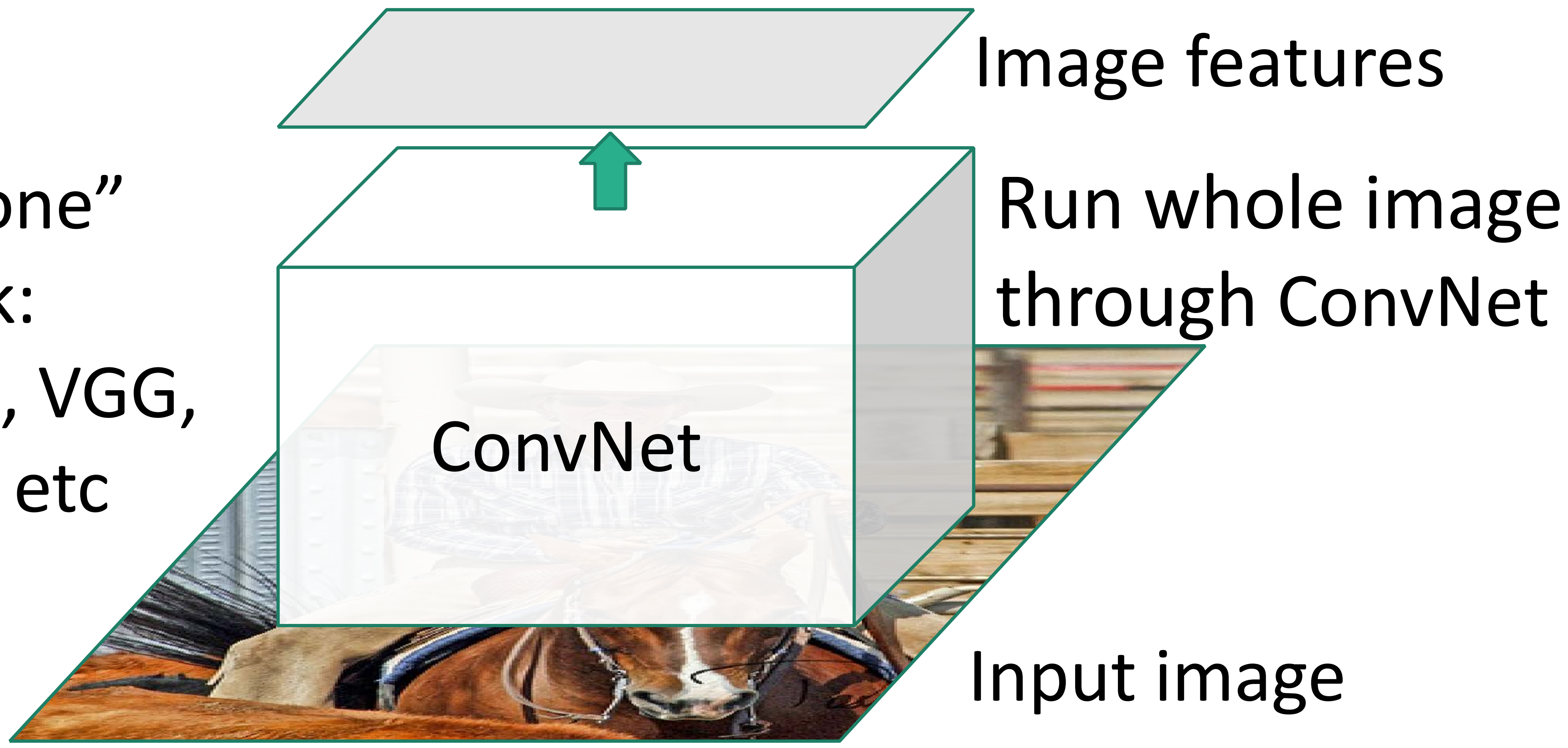
Fast R-CNN

Input image

"Slow" R-CNN
Process each region independently

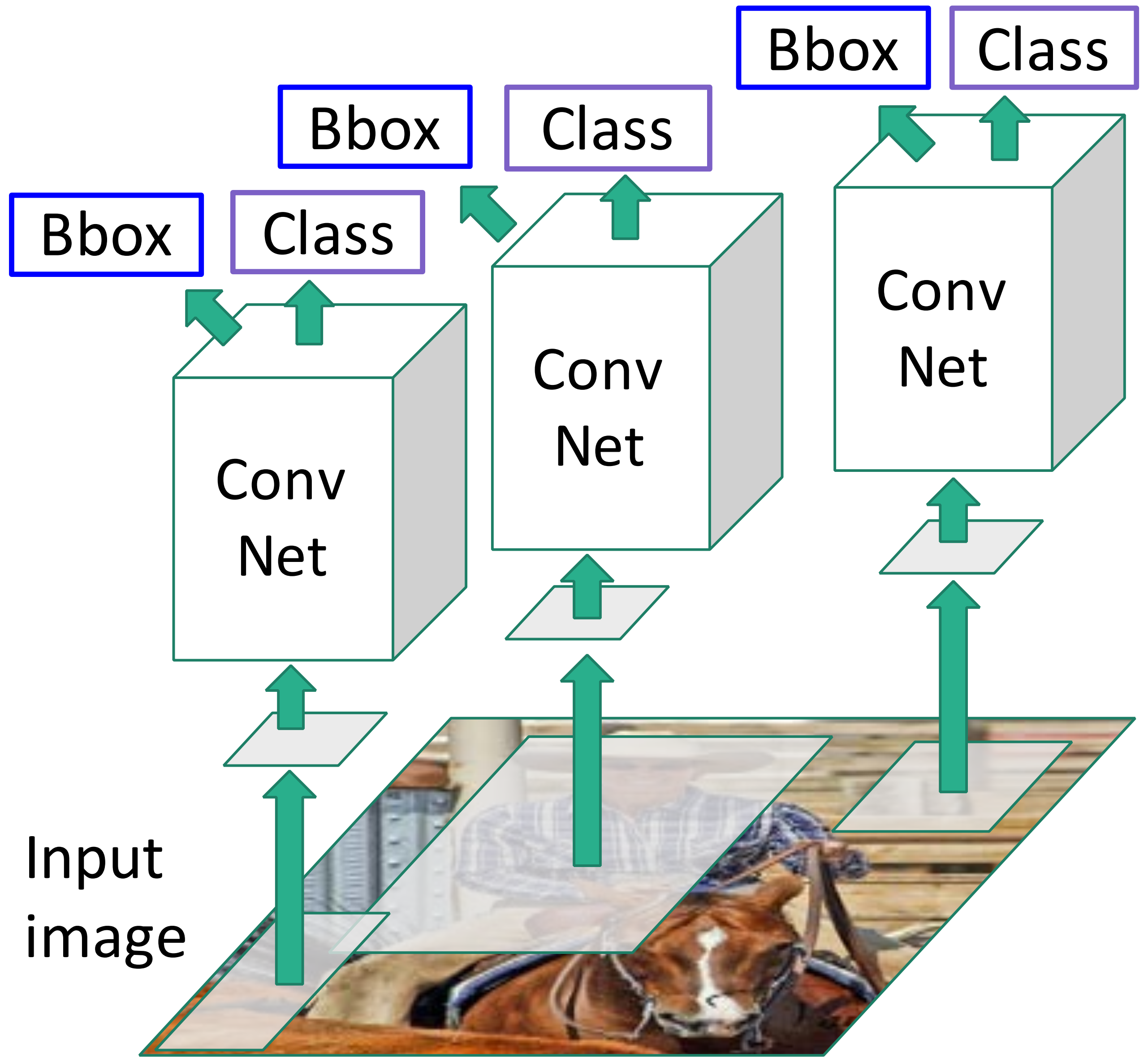


Fast R-CNN

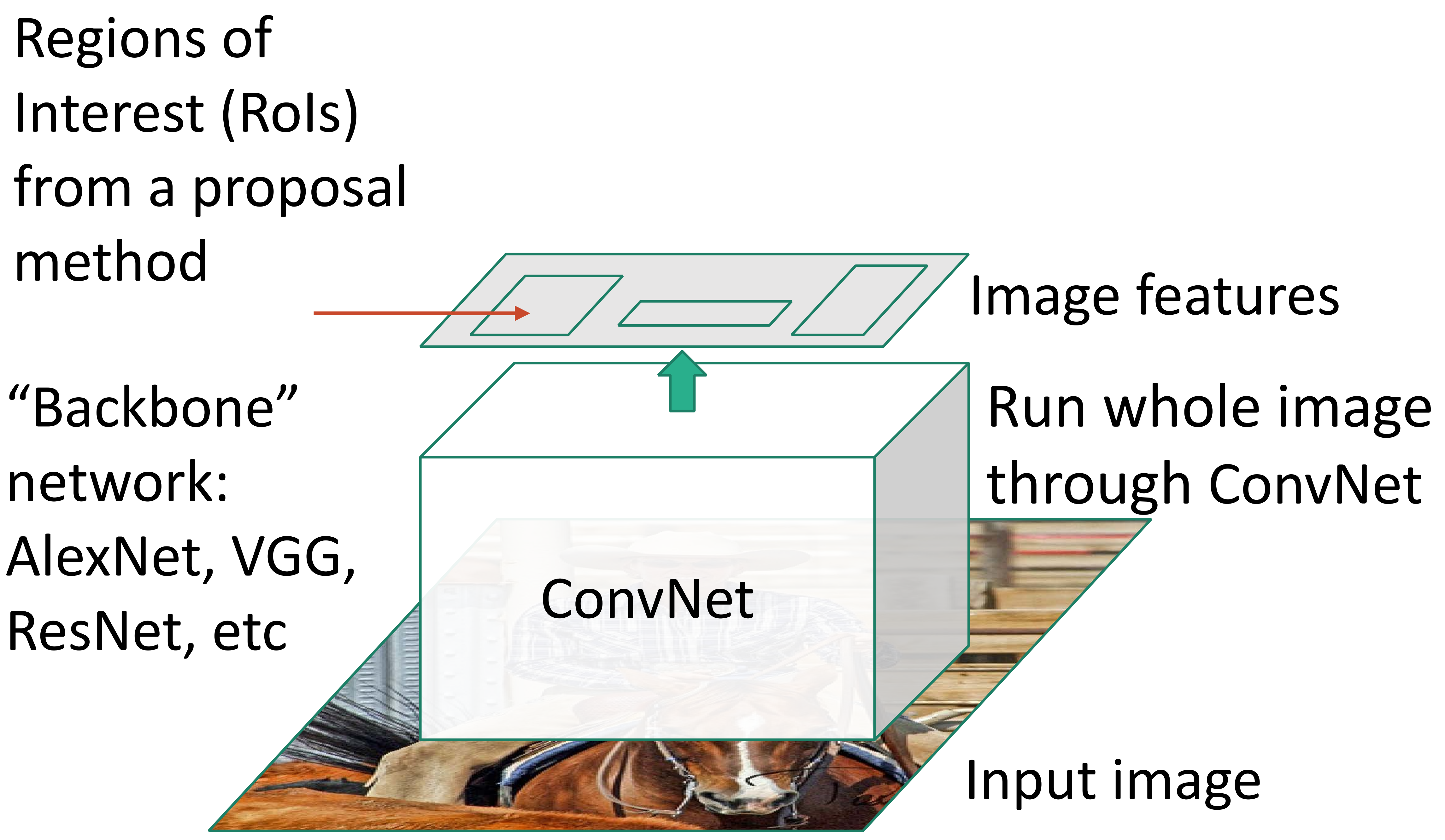


"Backbone" network: AlexNet, VGG, ResNet, etc

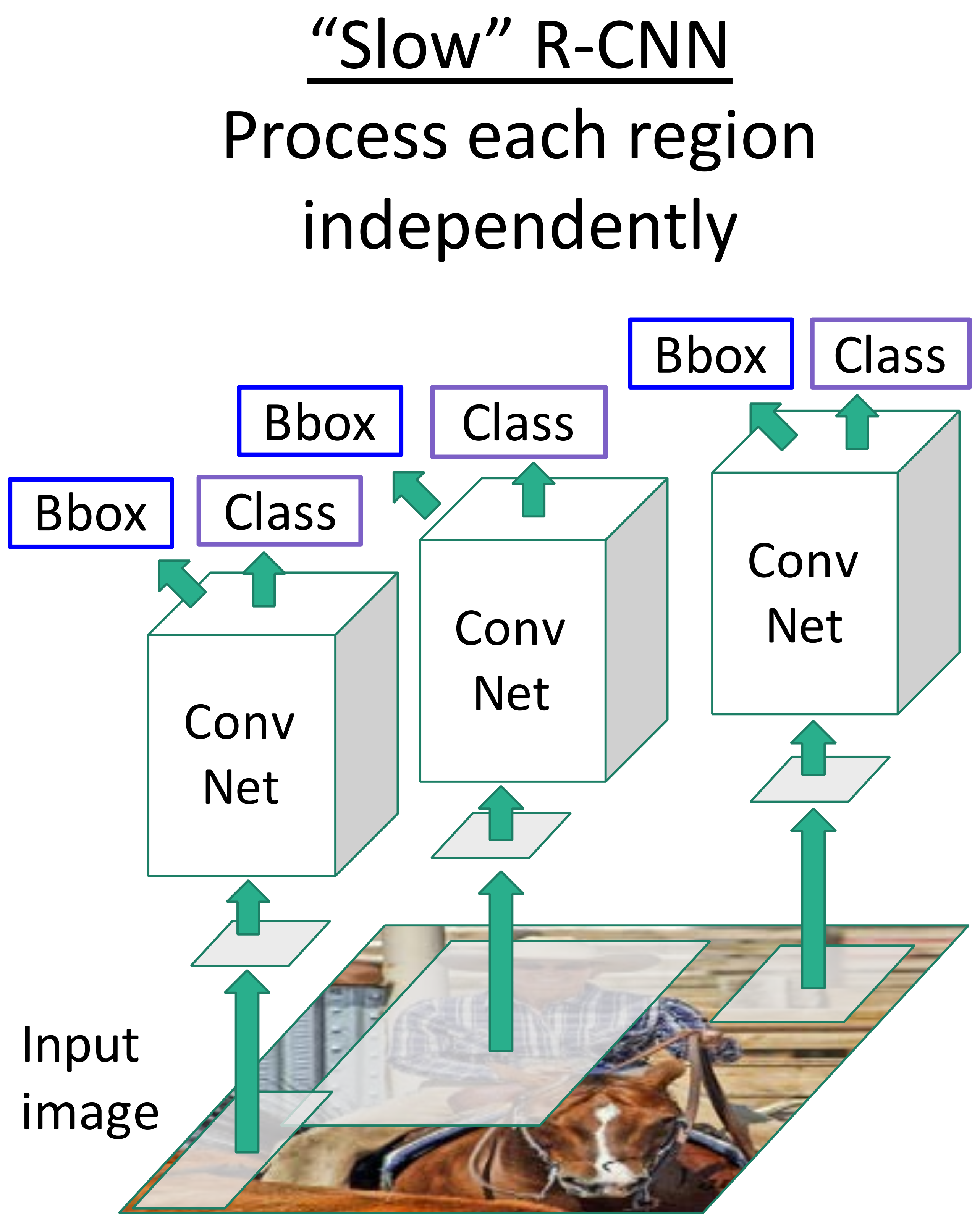
<u>"Slow" R-CNN</u>
Process each region independently



Fast R-CNN

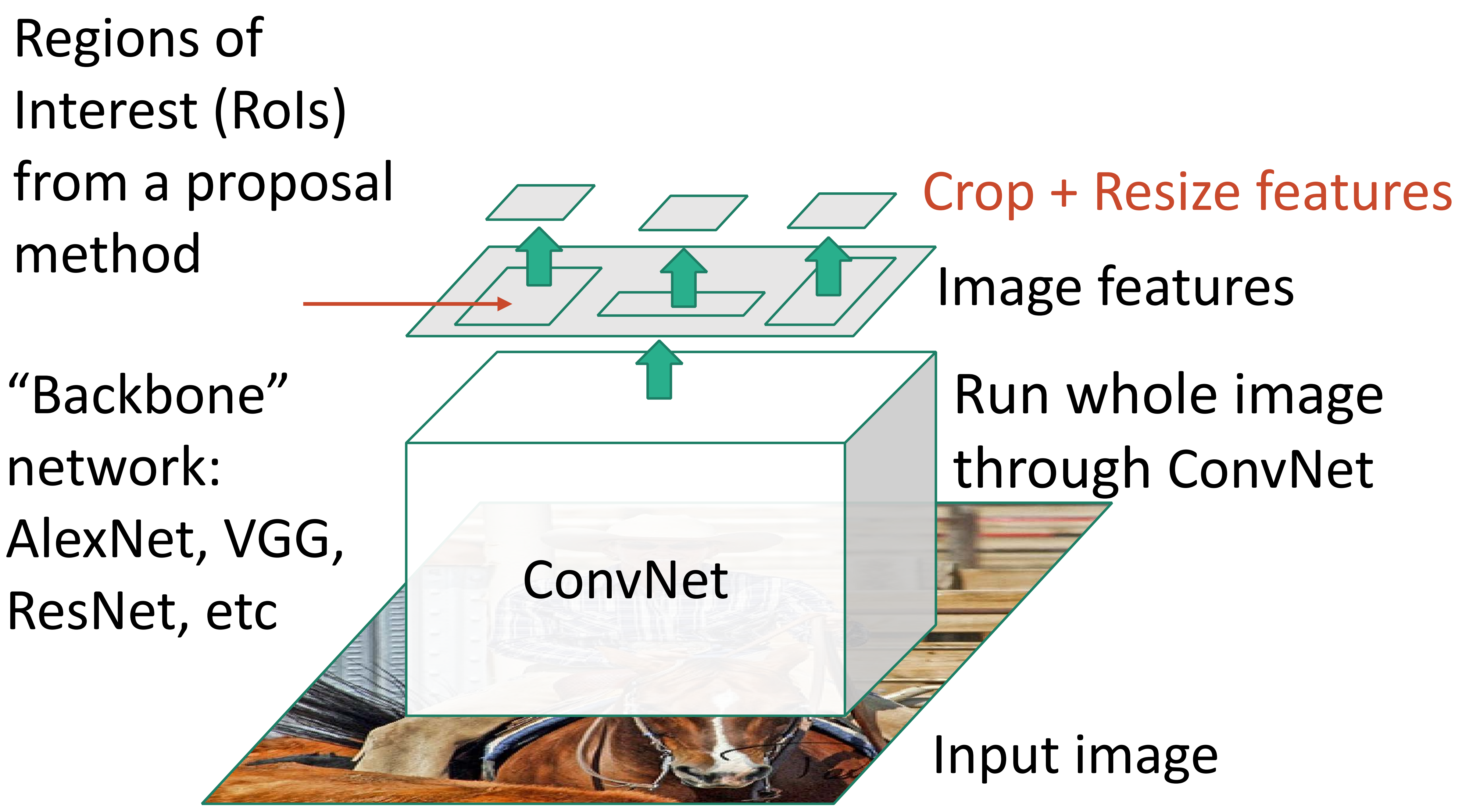


Regions of Interest (RoIs) from a proposal method
Image features
"Backbone" network: AlexNet, VGG, ResNet, etc
Run whole image through ConvNet
ConvNet
Input image

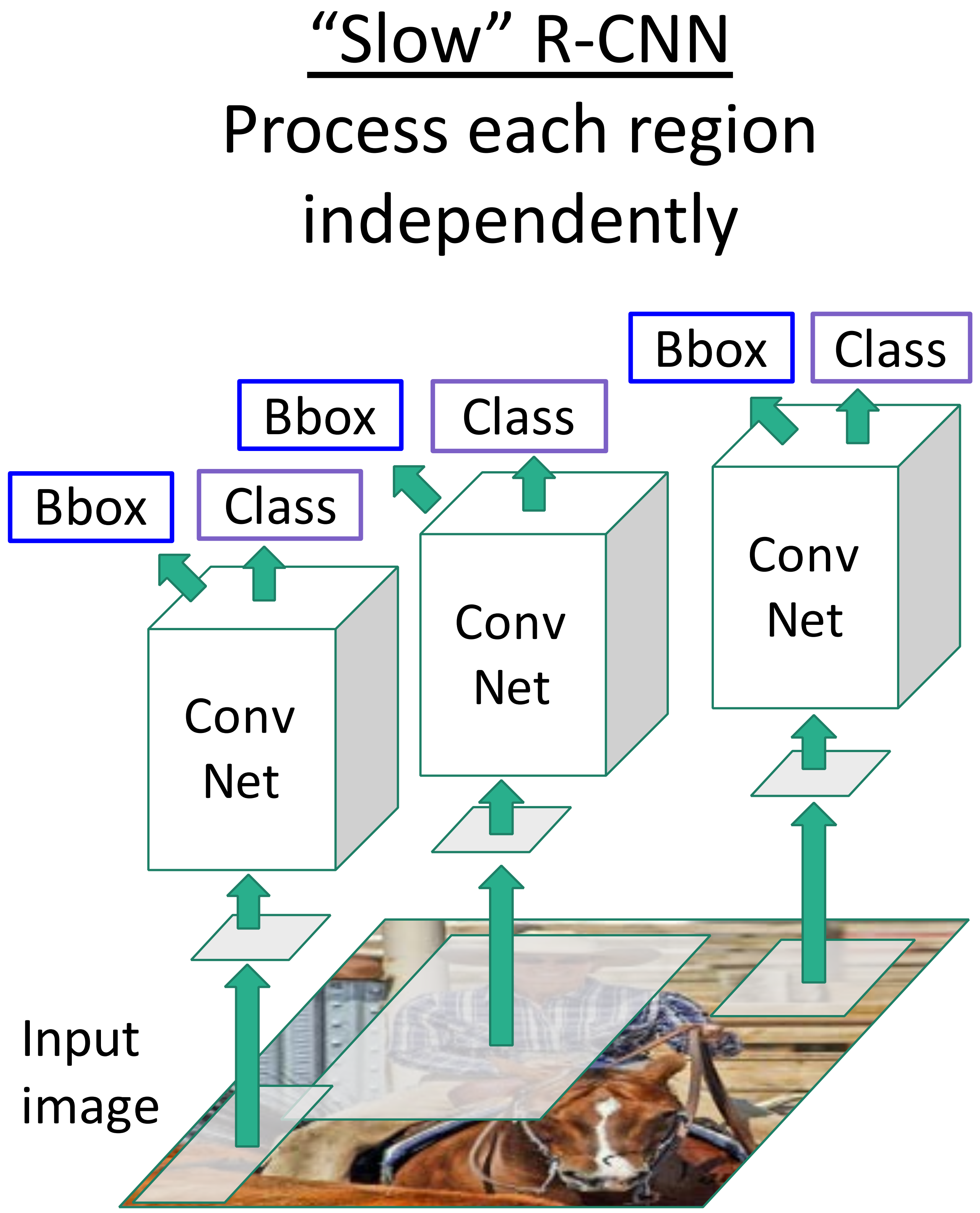


"Slow" R-CNN
Process each region independently
Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Input image

Fast R-CNN

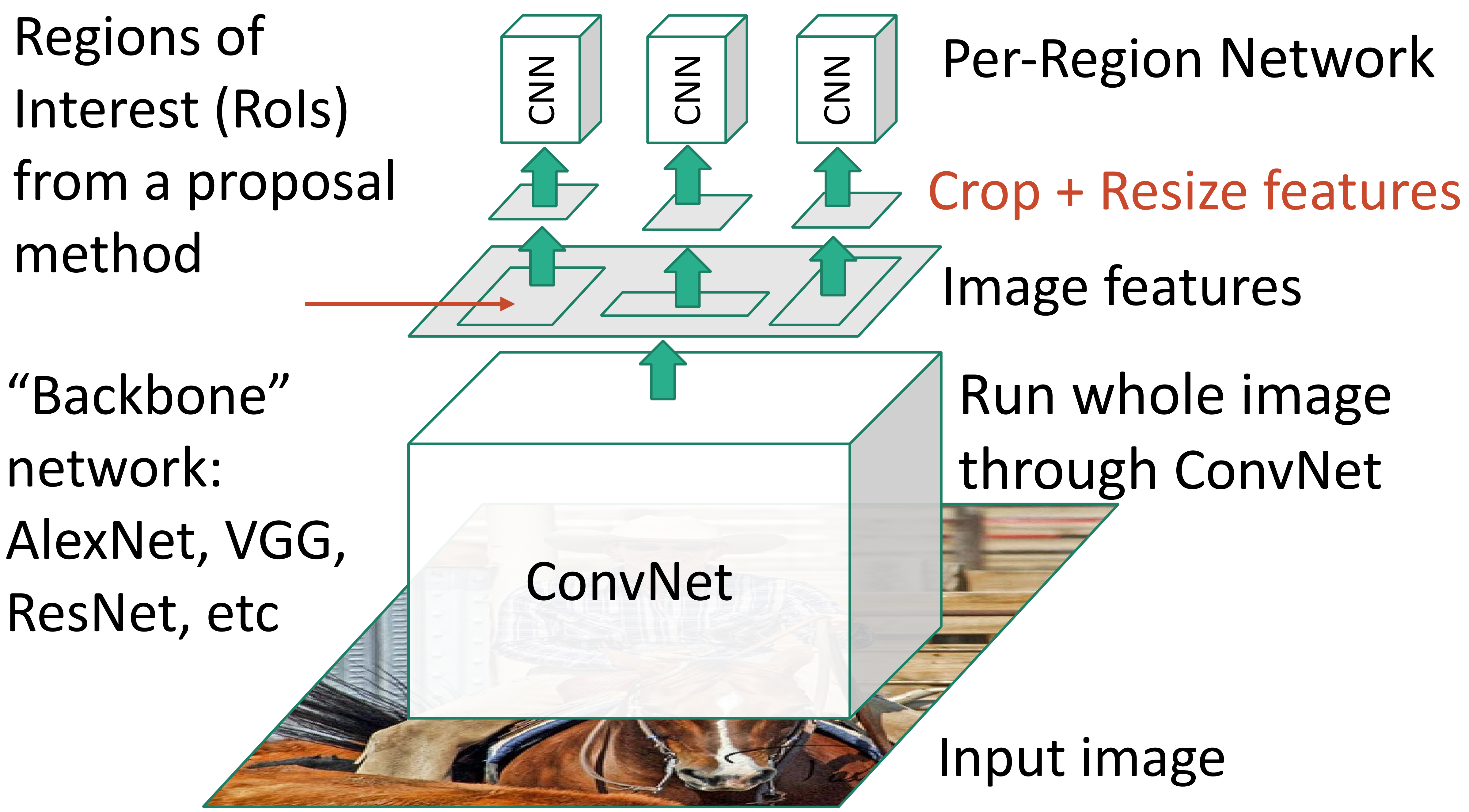


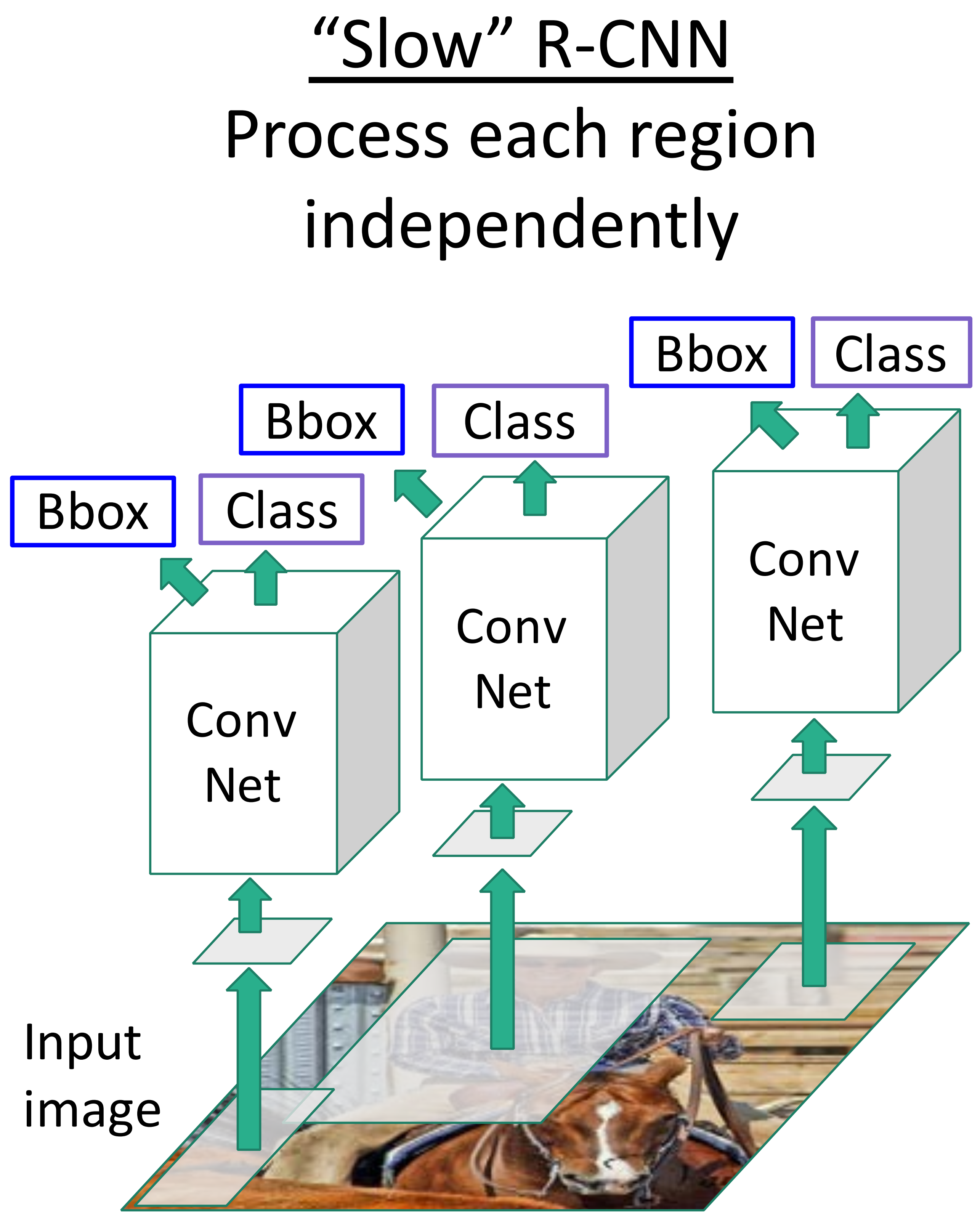
Regions of Interest (RoIs) from a proposal method
"Backbone" network: AlexNet, VGG, ResNet, etc
ConvNet
Crop + Resize features
Image features
Run whole image through ConvNet
Input image



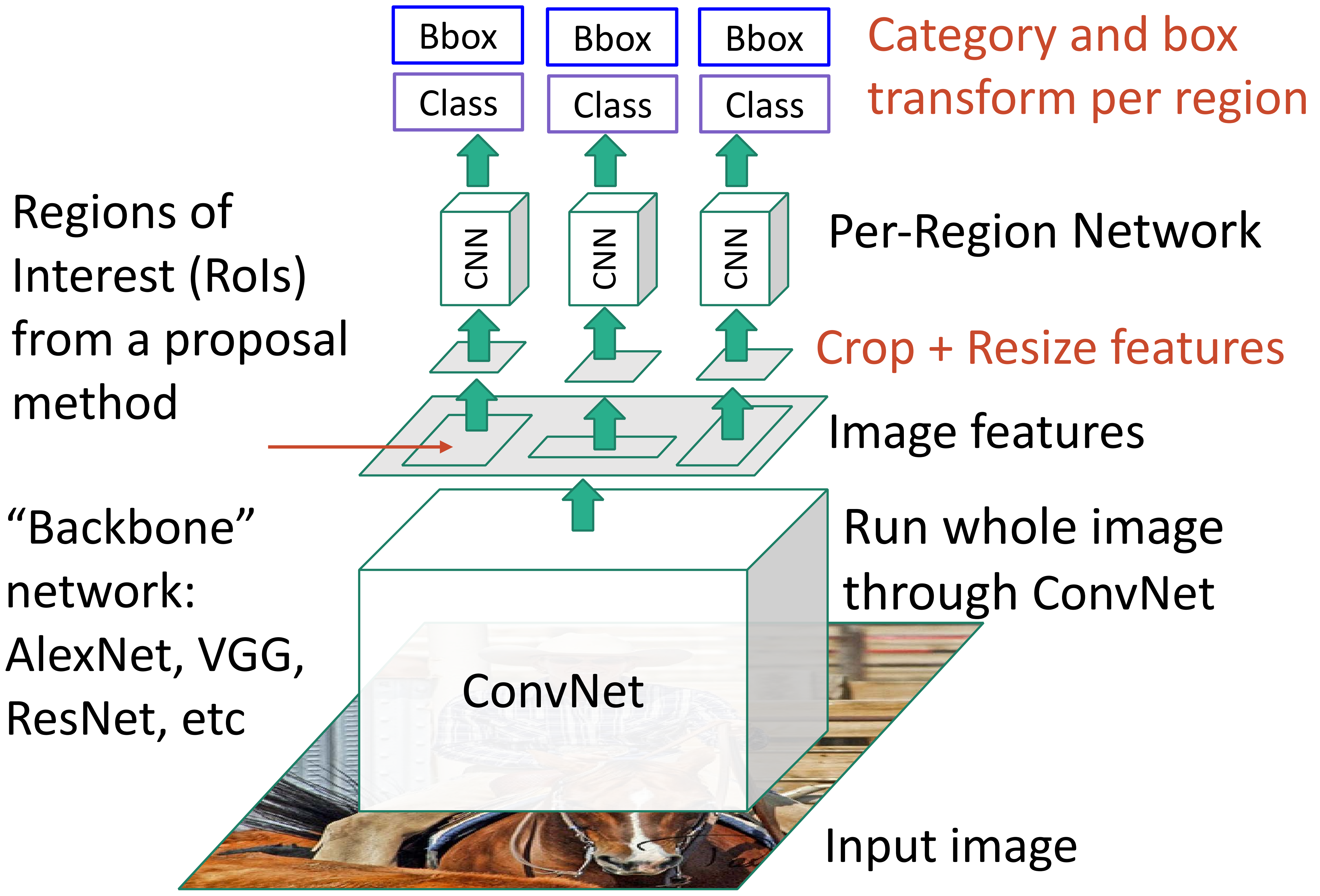
"Slow" R-CNN
Process each region independently
Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Input image

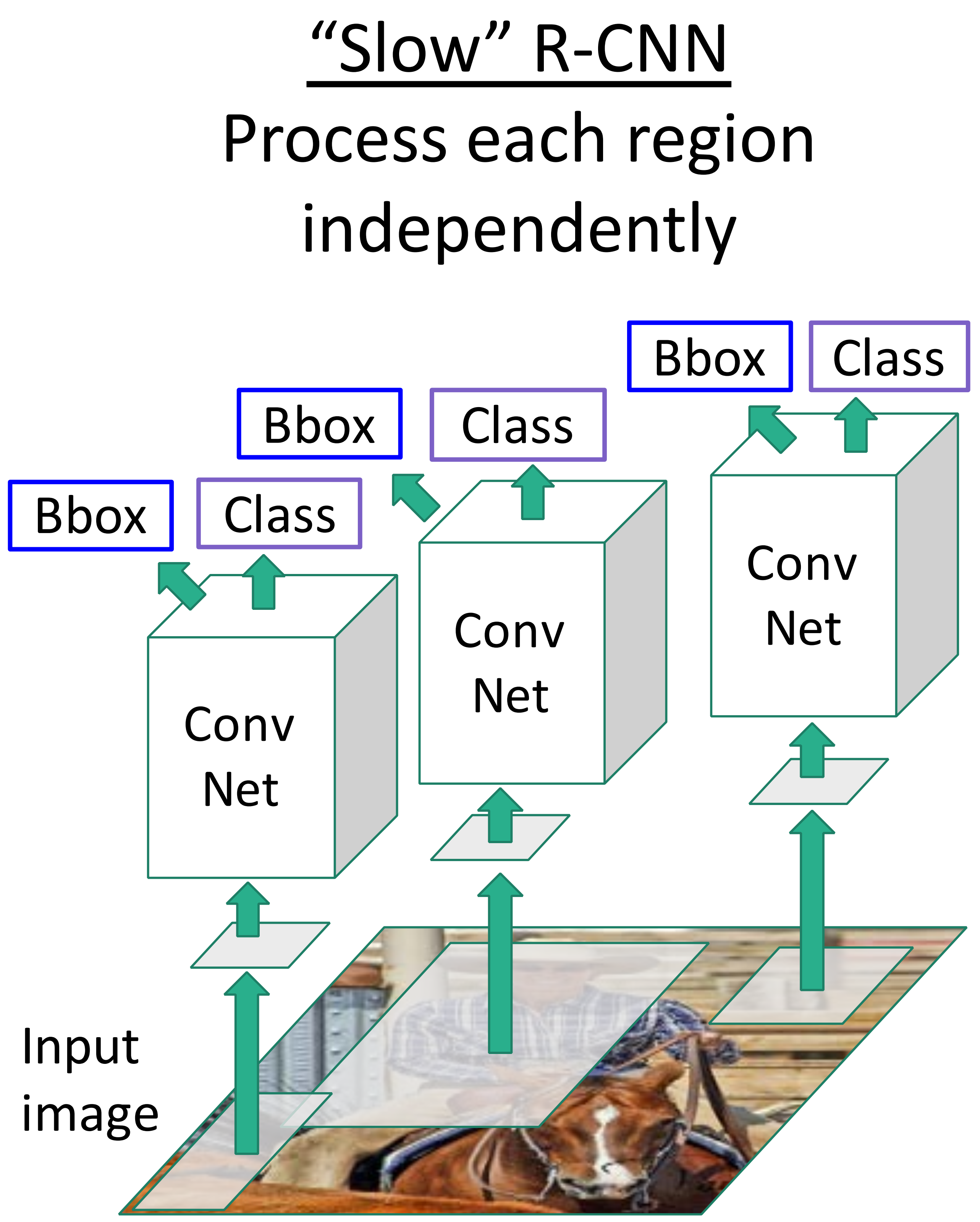
Fast R-CNN


Regions of Interest (RoIs) from a proposal method
"Backbone" network: AlexNet, VGG, ResNet, etc
CNN
CNN
CNN
ConvNet
Per-Region Network
Crop + Resize features
Image features
Run whole image through ConvNet
Input image

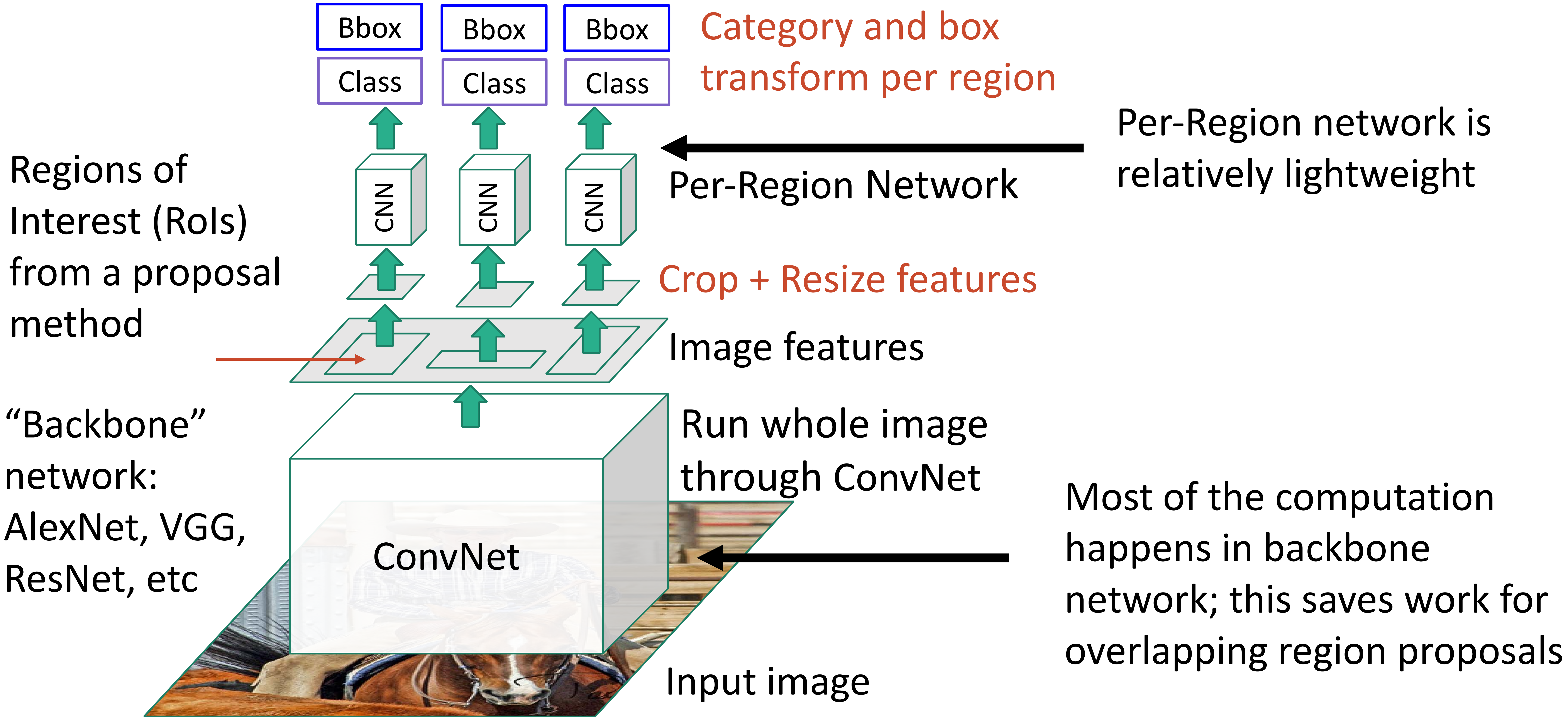

"Slow" R-CNN
Process each region independently
Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Input image

Fast R-CNN


Bbox
Class
Bbox
Class
Bbox
Class
Category and box transform per region
CNN
CNN
CNN
Per-Region Network
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
Image features
"Backbone" network: AlexNet, VGG, ResNet, etc
Run whole image through ConvNet
ConvNet
Input image

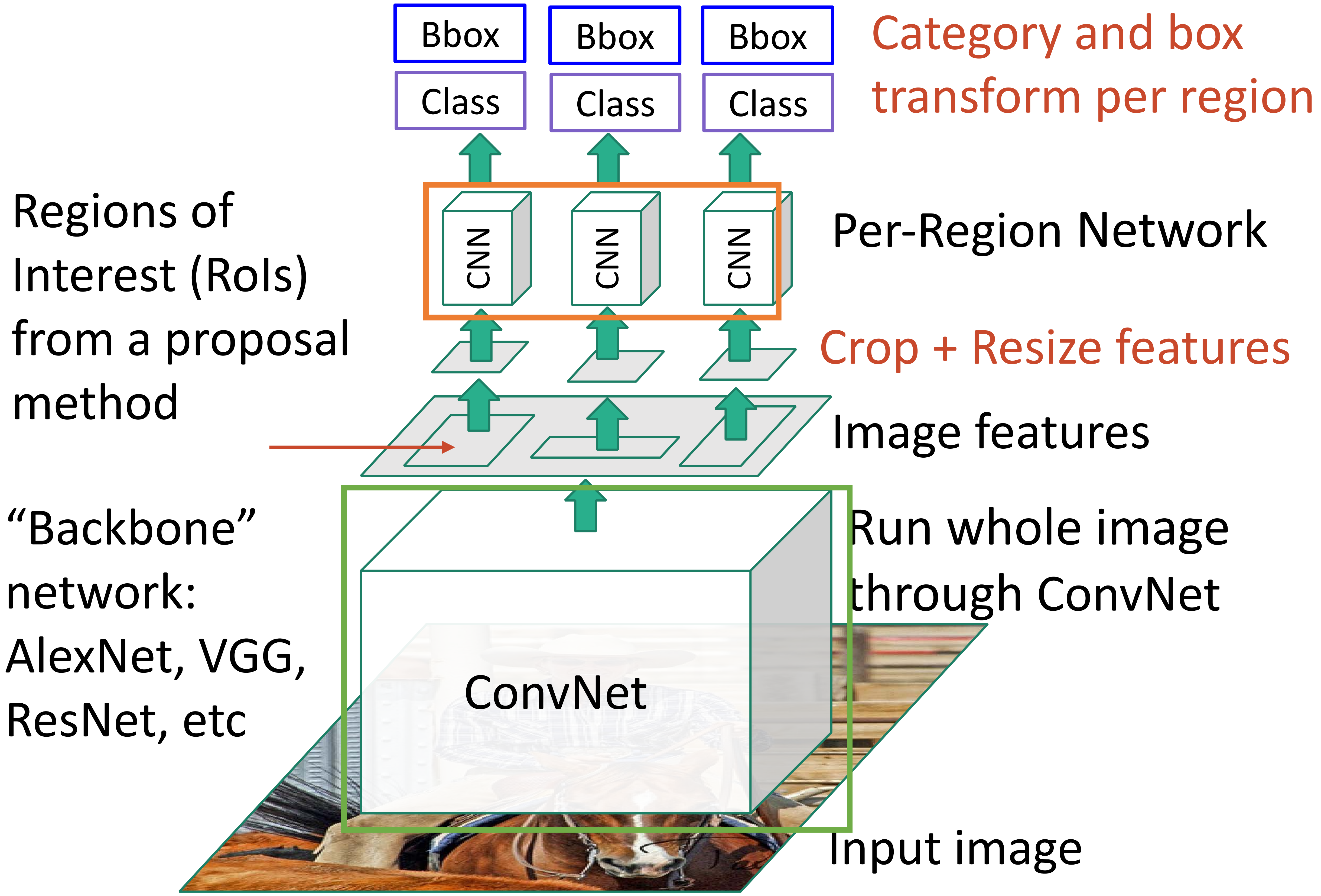

"Slow" R-CNN
Process each region independently
Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Input image

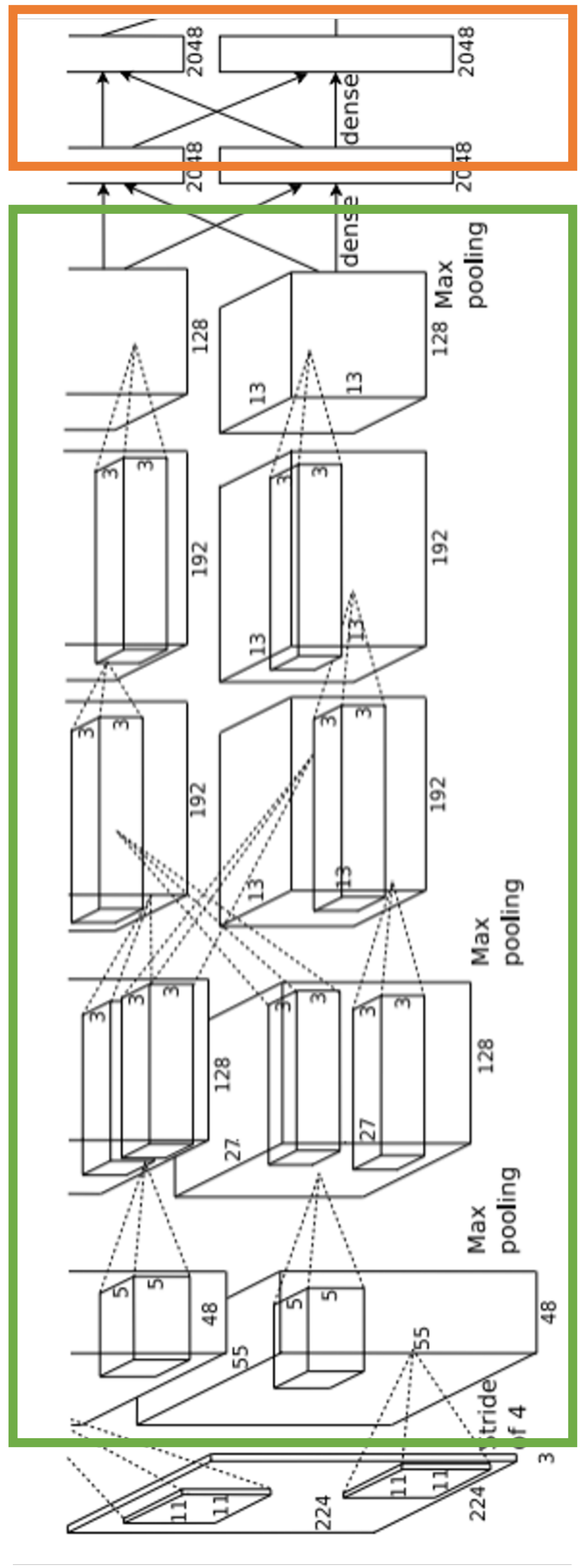
Fast R-CNN



Bbox
Bbox
Bbox
Class
Class
Class
Category and box transform per region
CNN
CNN
CNN
Per-Region Network
Per-Region network is relatively lightweight
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
Image features
"Backbone" network: AlexNet, VGG, ResNet, etc
ConvNet
Run whole image through ConvNet
Most of the computation happens in backbone network; this saves work for overlapping region proposals
Input image

Fast R-CNN





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

Fast R-CNN

Category and box transform per region

Per-Region Network

Crop + Resize features

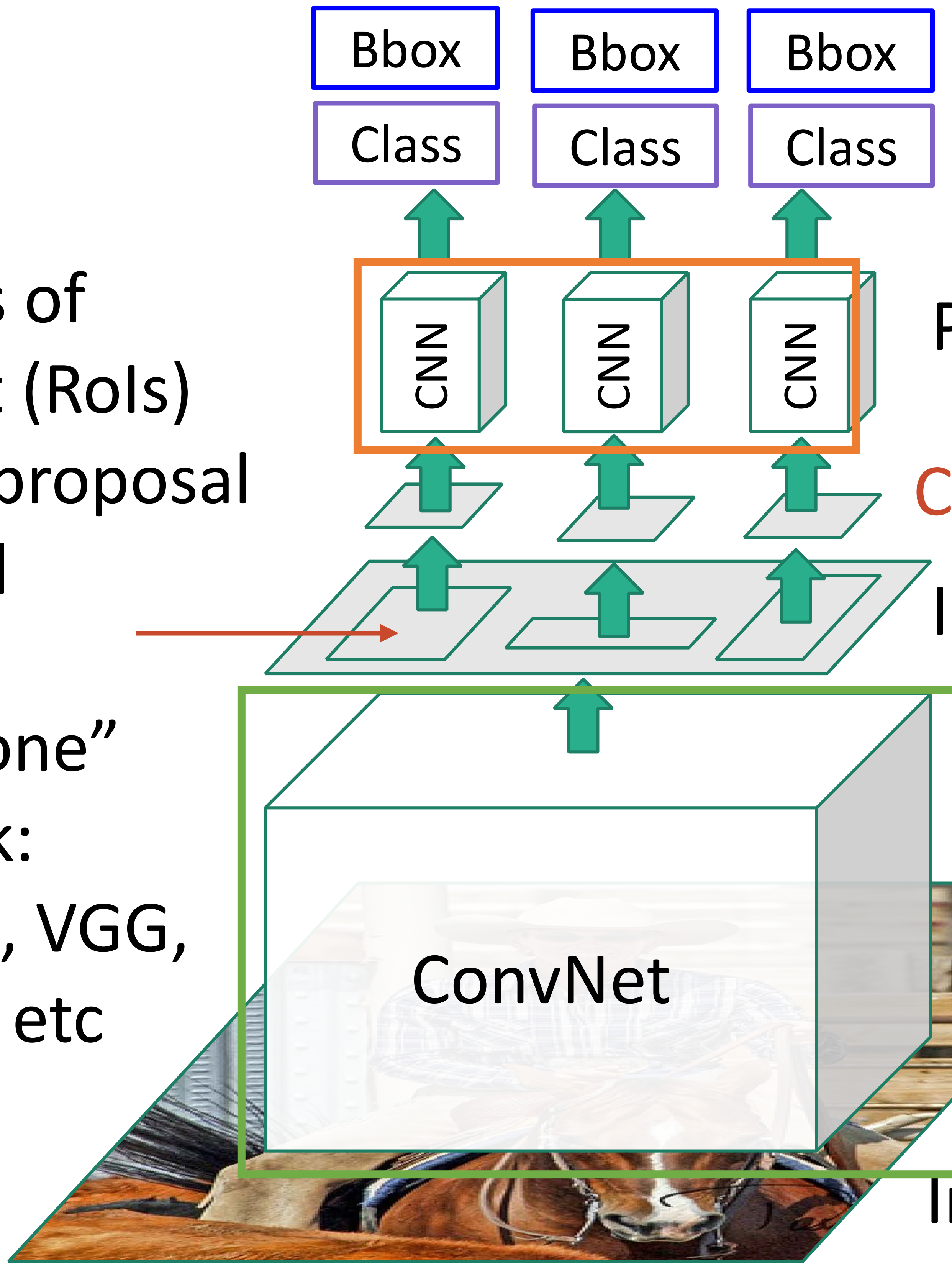
Image features

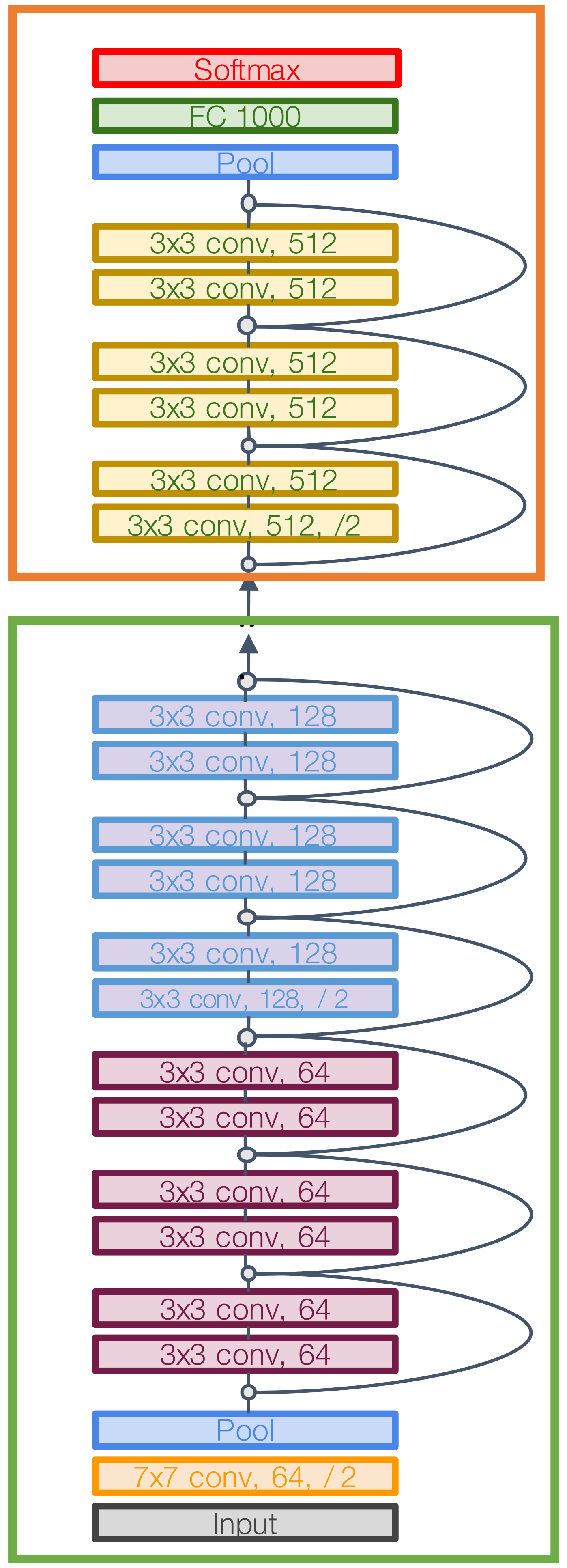
Run whole image through ConvNet

Input image

Regions of Interest (RoIs) from a proposal method

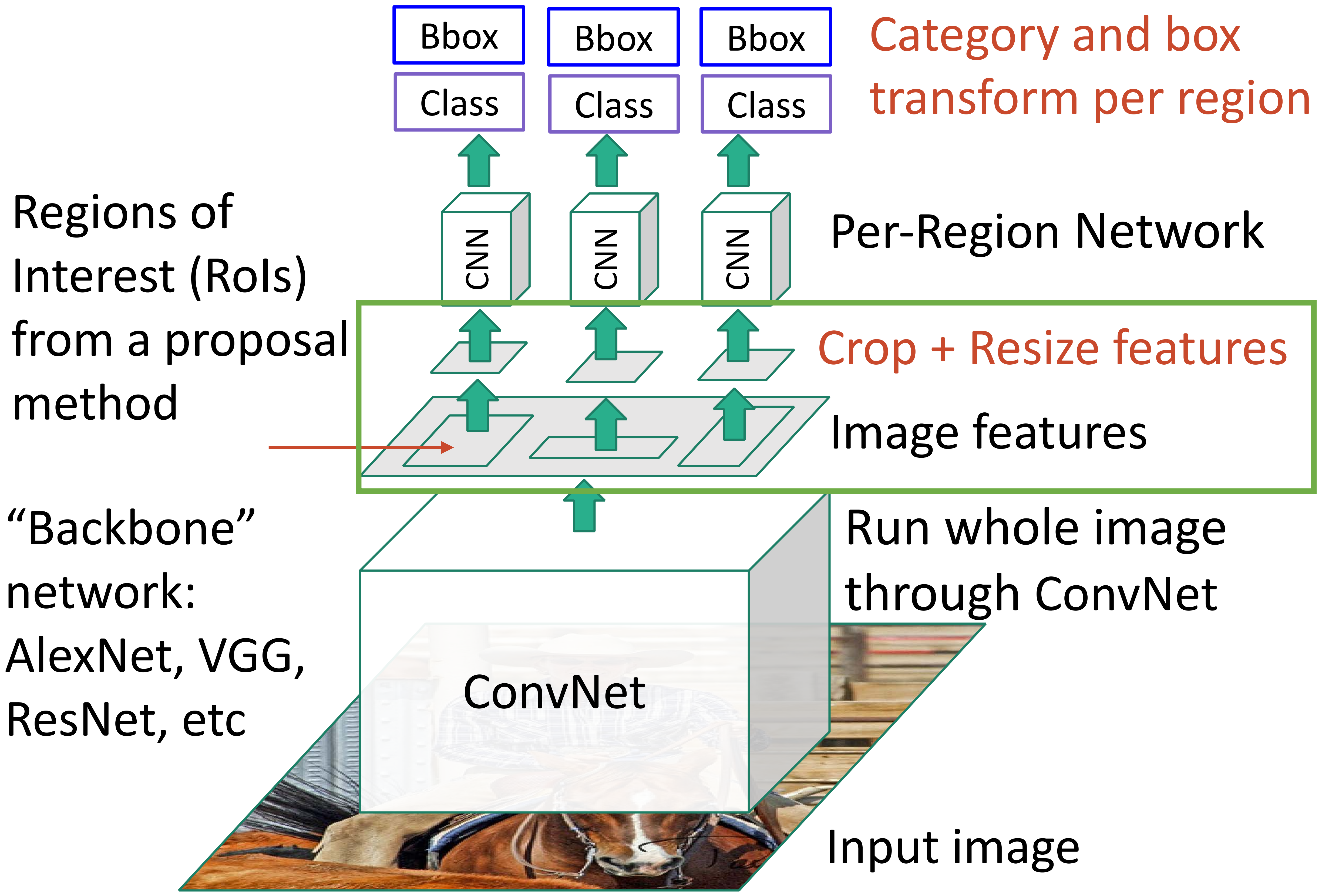
“Backbone” network: AlexNet, VGG, ResNet, etc





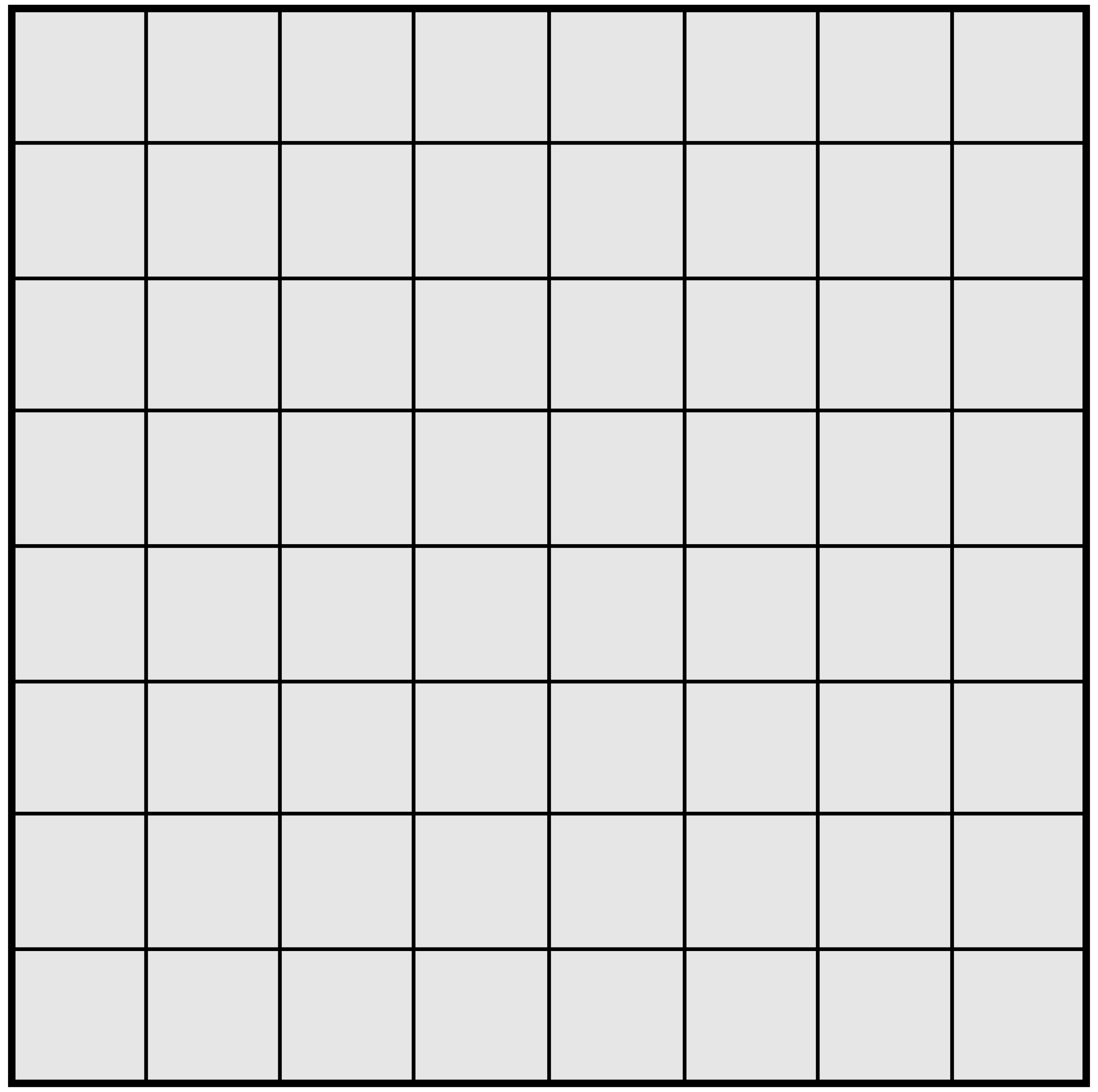
Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Fast R-CNN



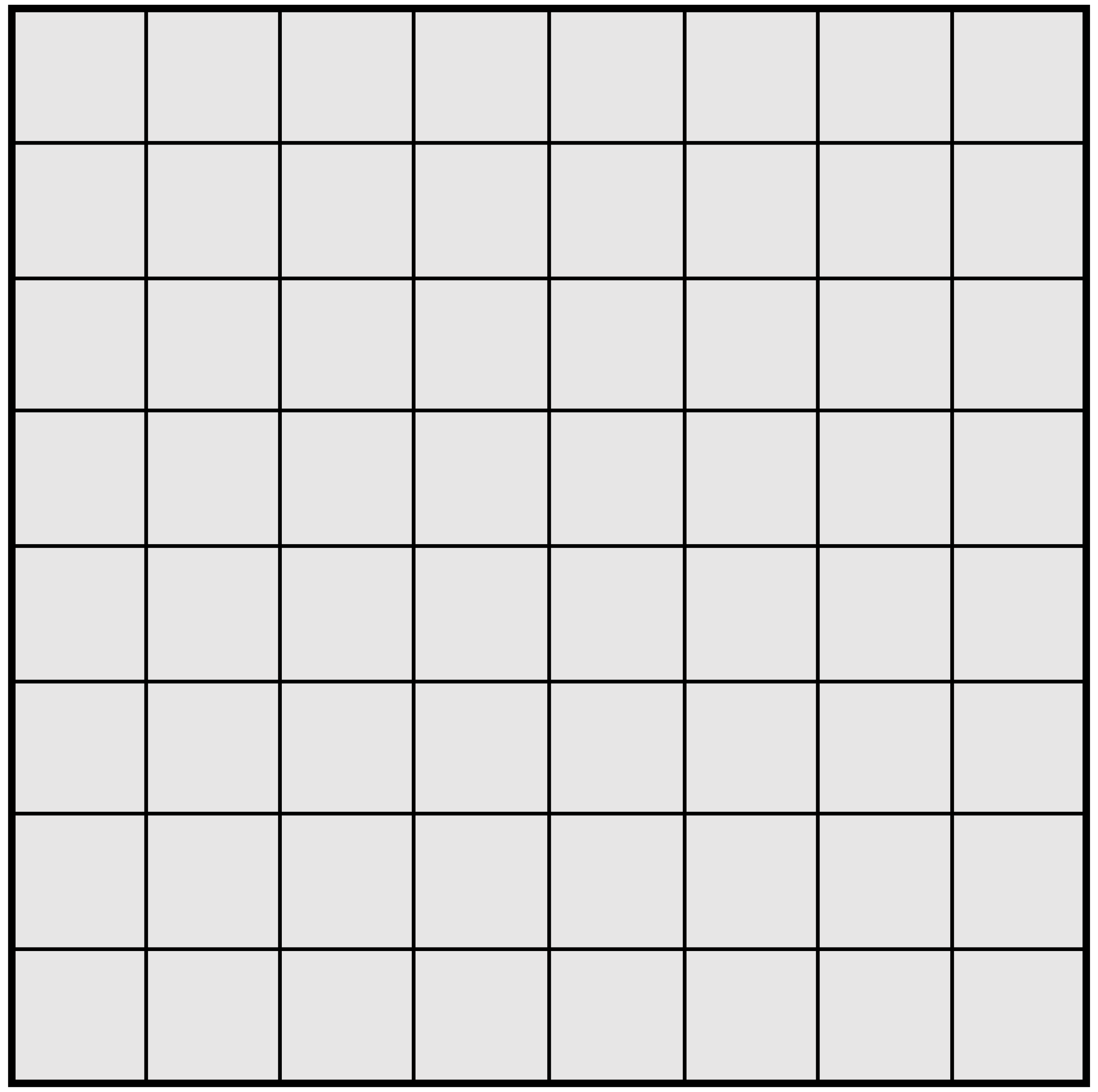
How to crop features?

Recall: Receptive Fields



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

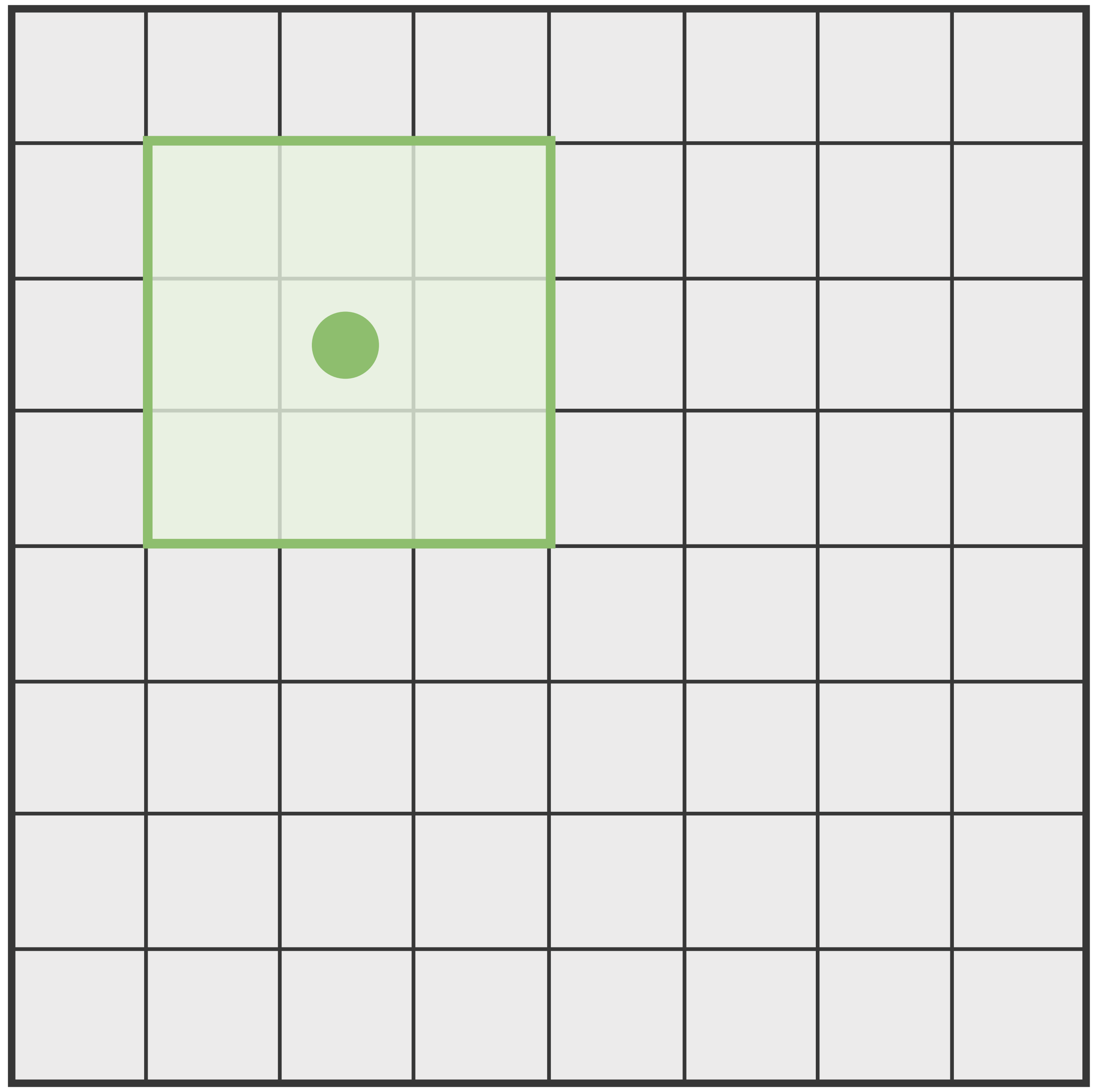
3x3 Conv
Stride 1, pad 1



Input Image: 8 x 8

Output Image: 8 x 8

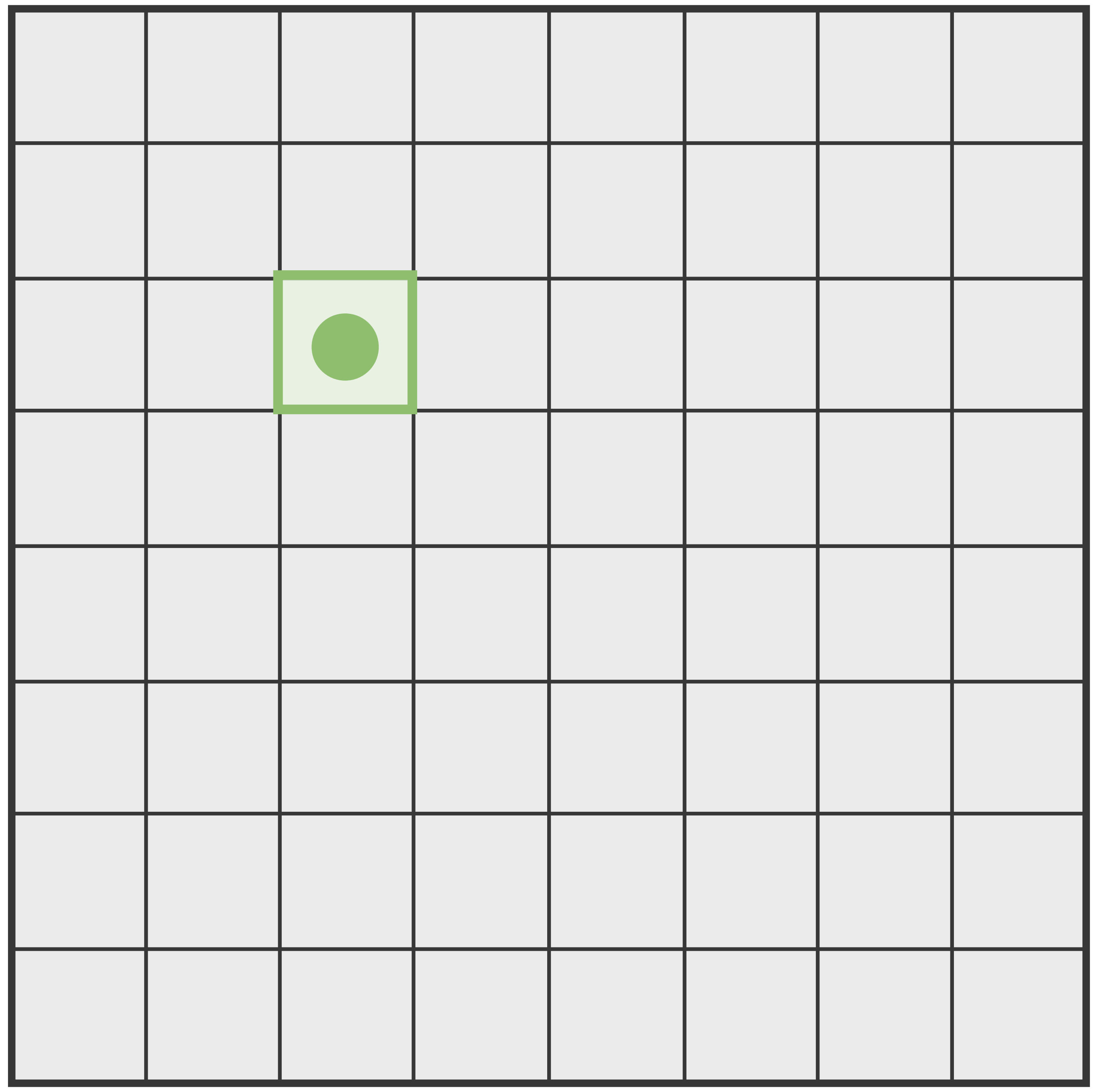
Recall: Receptive Fields



Input Image: 8 x 8

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

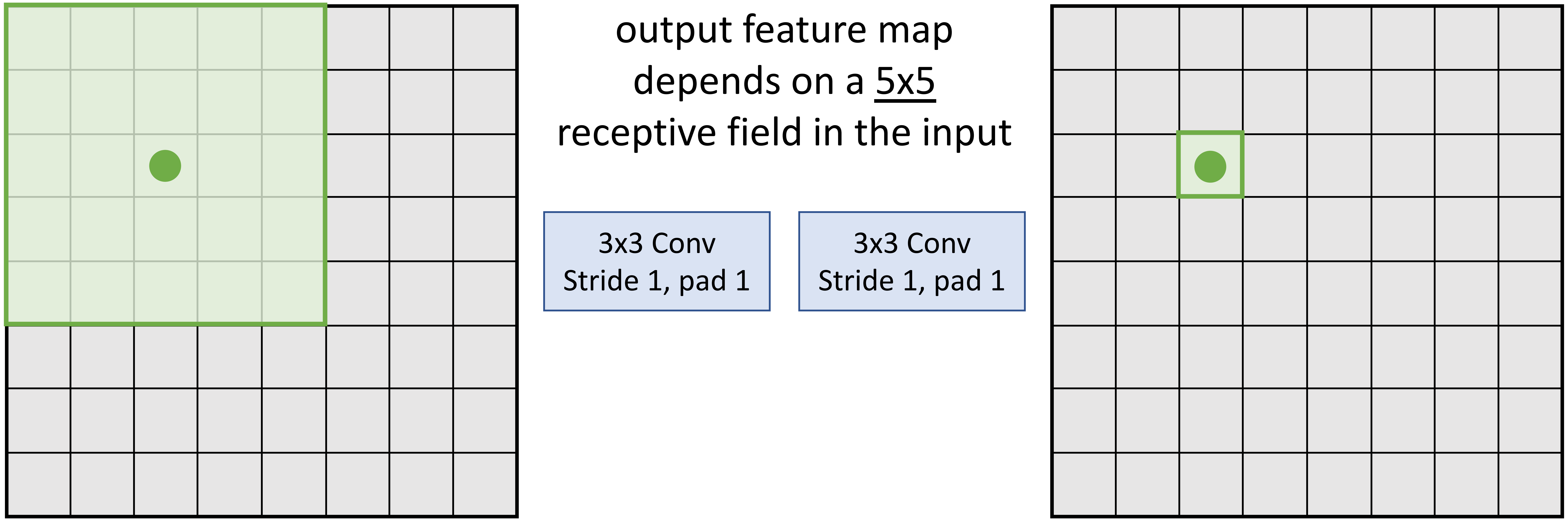
3x3 Conv
Stride 1, pad 1



Output Image: 8 x 8

Recall: Receptive Fields

Every position in the output feature map depends on a 5x5 receptive field in the input

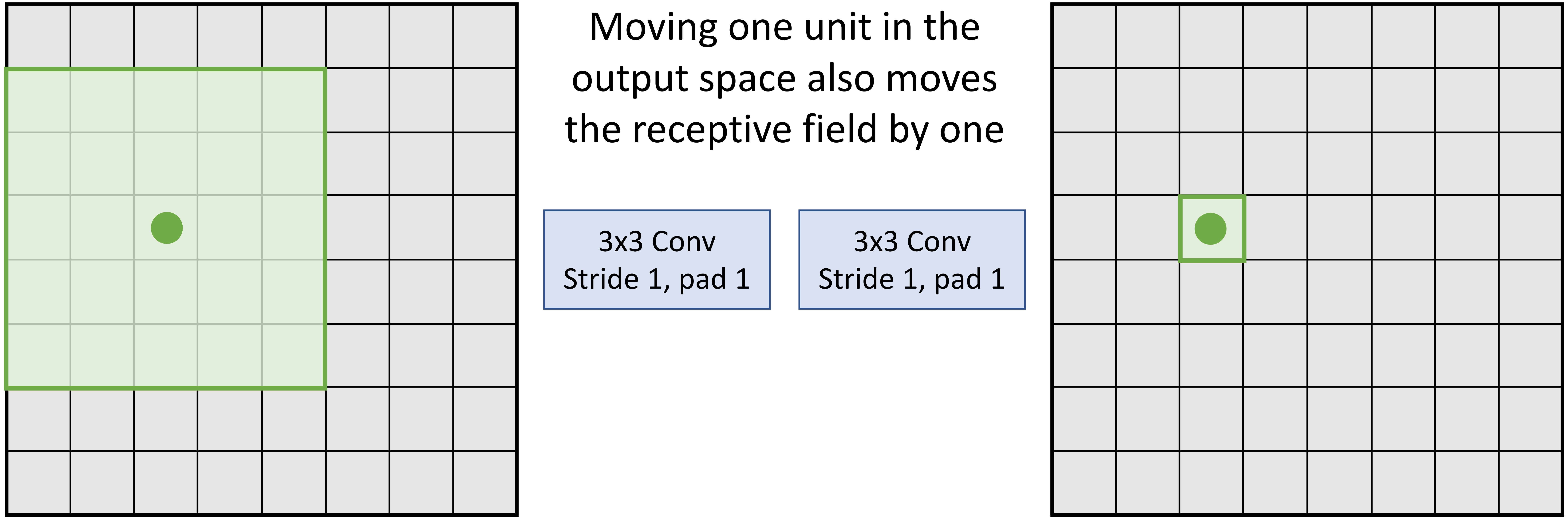


Input Image: 8 x 8

Output Image: 8 x 8

Recall: Receptive Fields

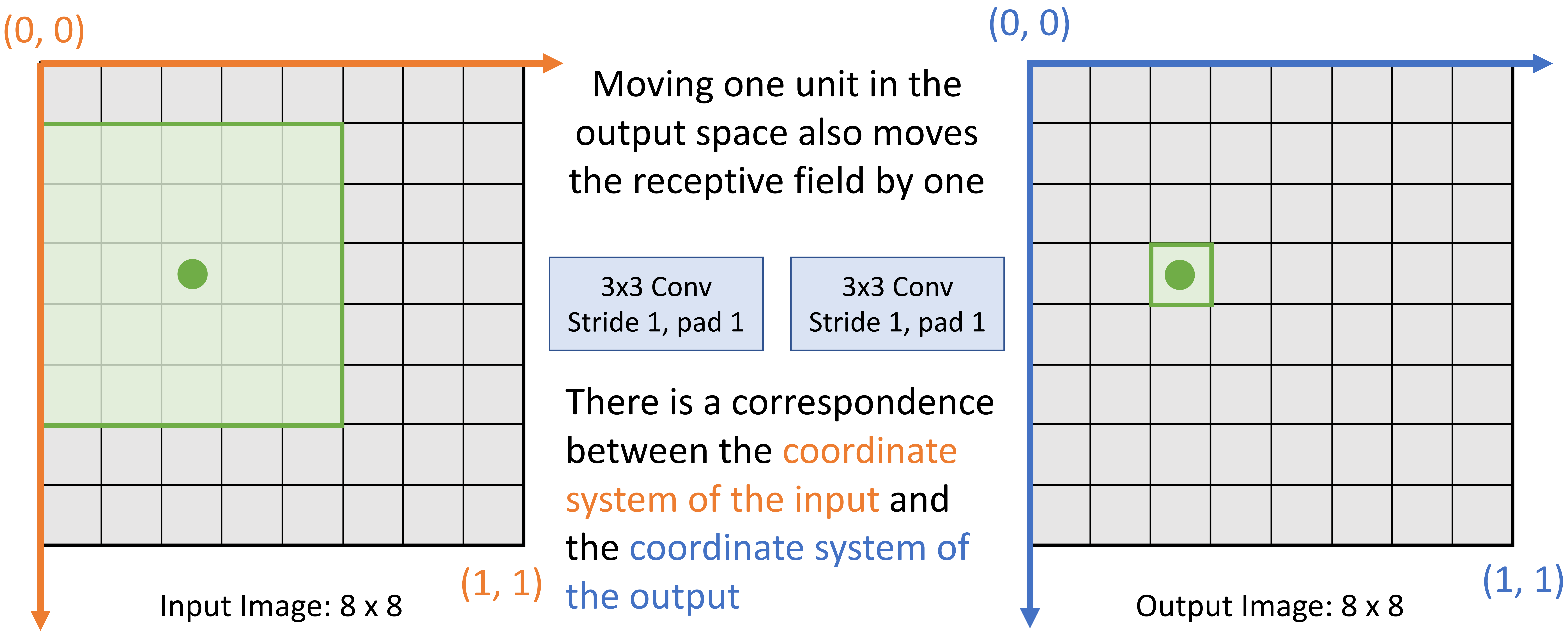
Moving one unit in the output space also moves the receptive field by one



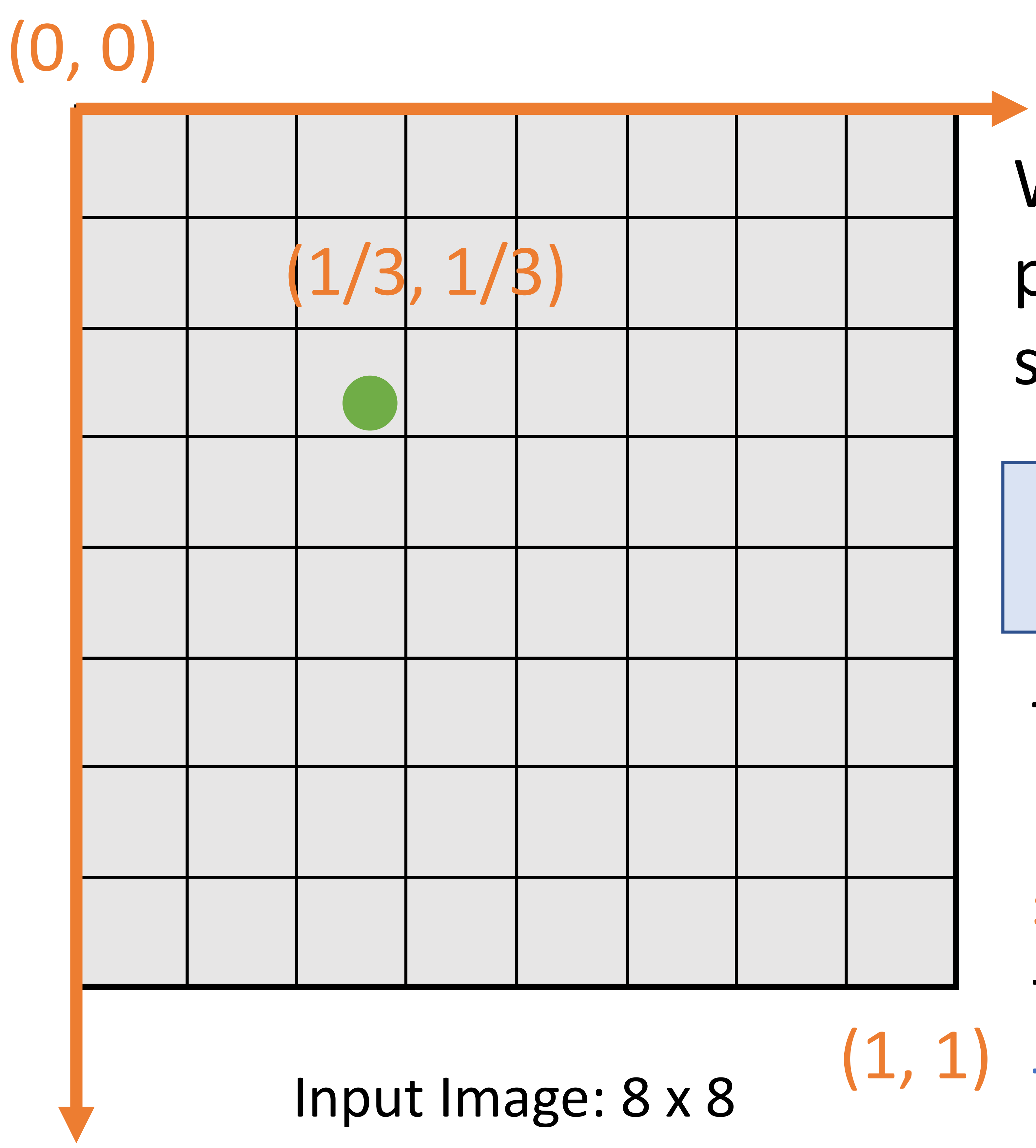
Input Image: 8 x 8

Output Image: 8 x 8

Recall: Receptive Fields



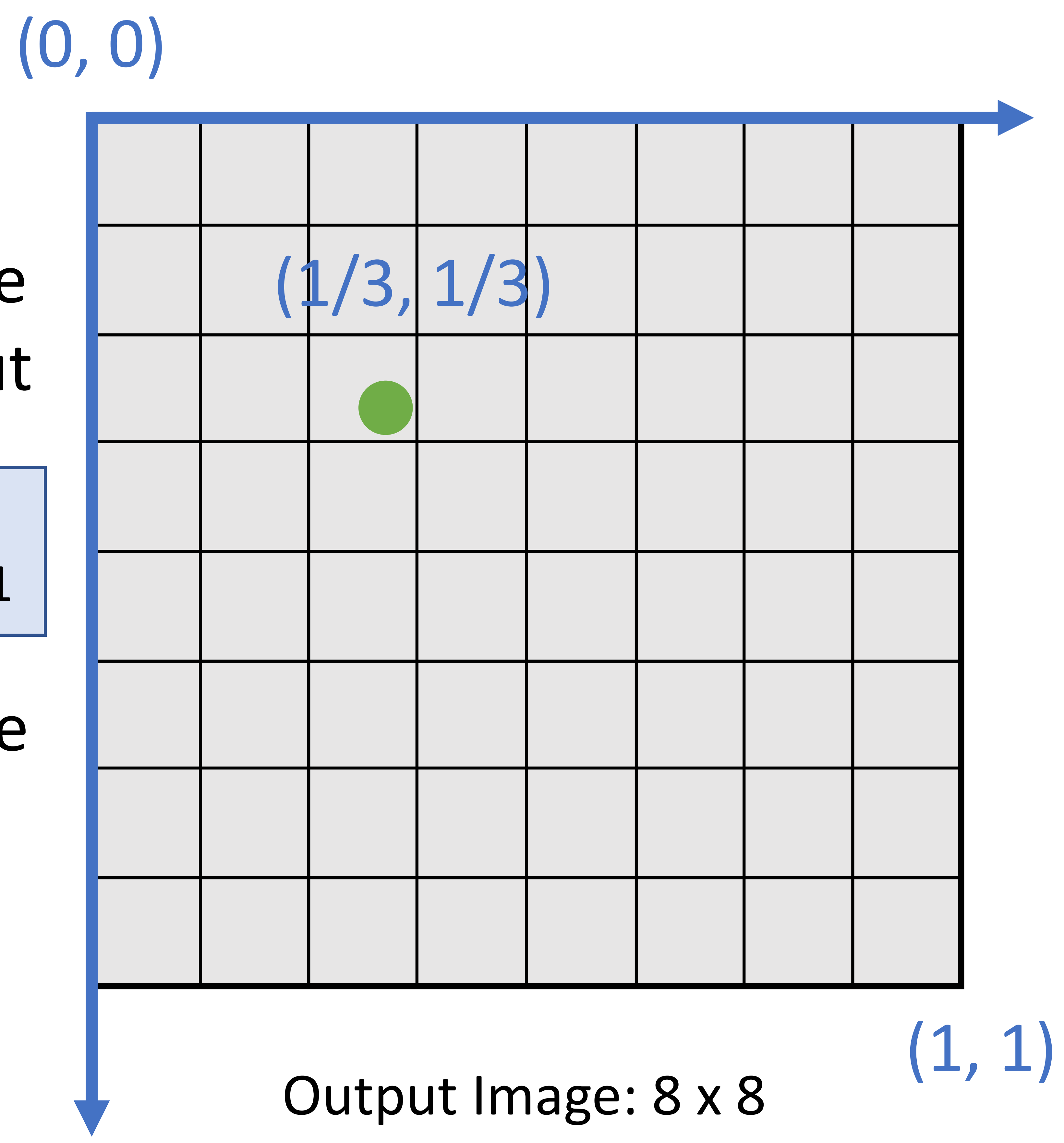
Projecting Points



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

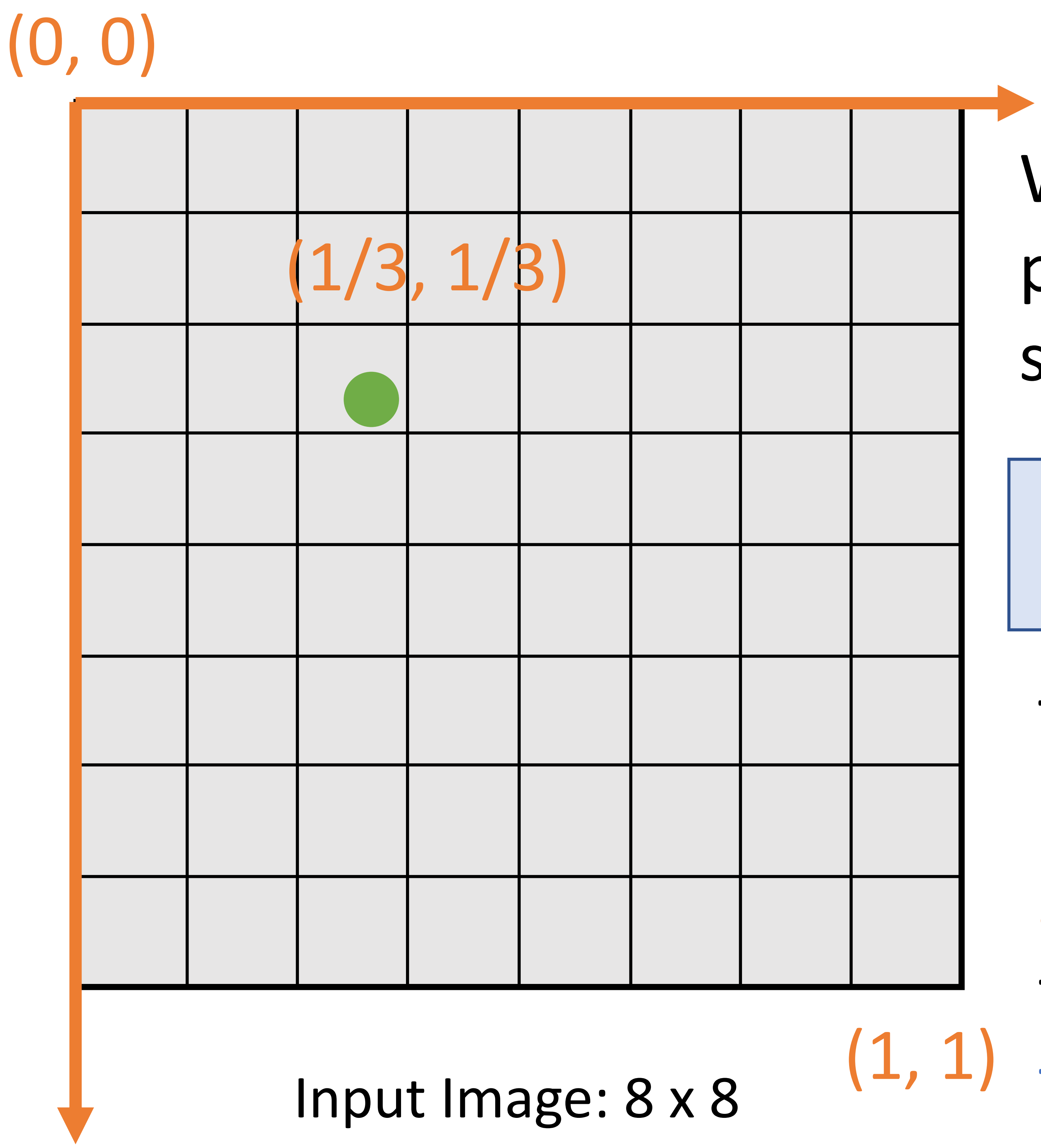
3x3 Conv Stride 1, pad 1	3x3 Conv Stride 1, pad 1

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



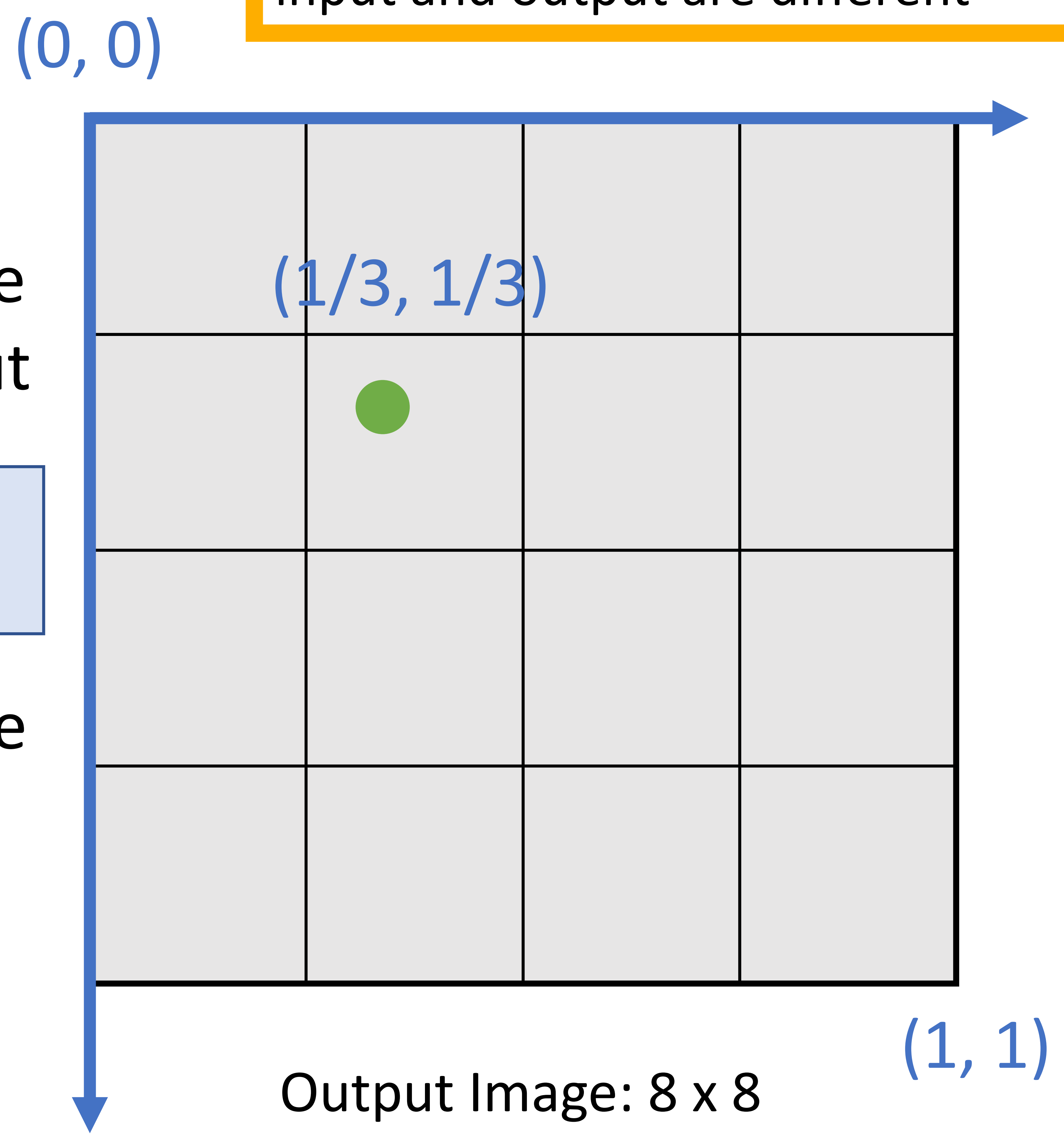
Input Image: 8 x 8

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

3x3 Conv
Stride 1, pad 1

2x2 MaxPool
Stride 2

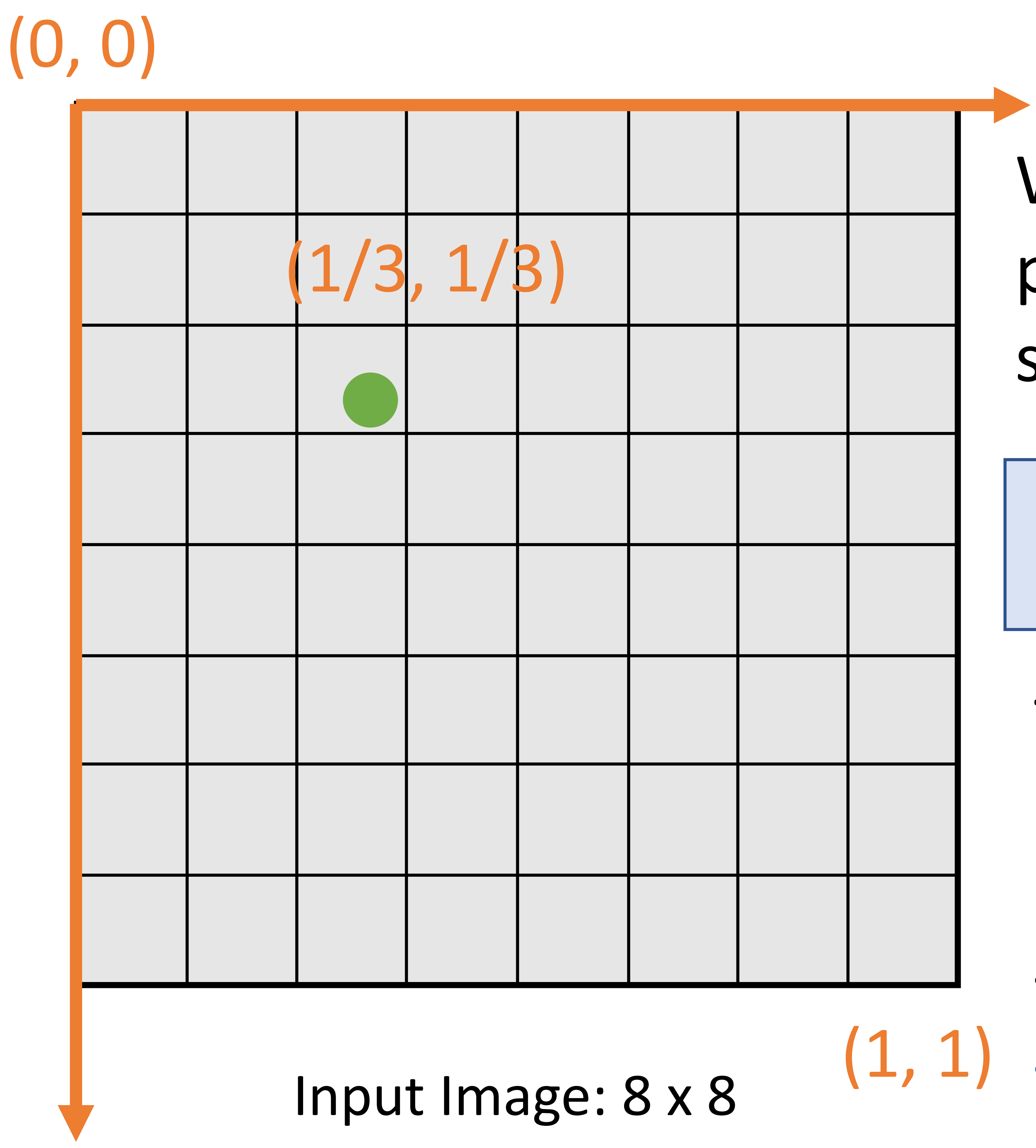
There is a correspondence between the coordinate system of the input and the coordinate system of the output



Output Image: 8 x 8

Projecting Points

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



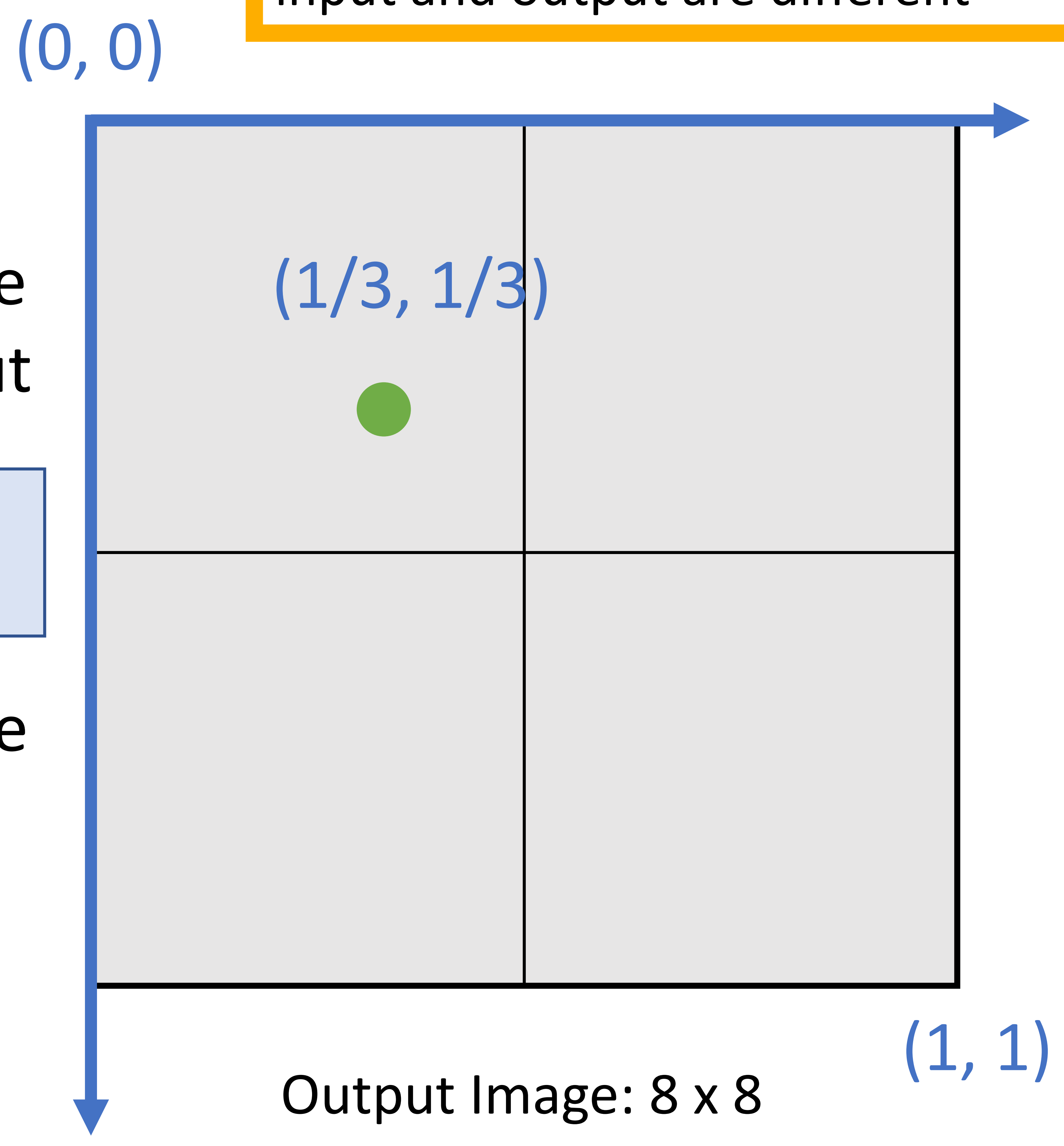
Input Image: 8 x 8

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

3x3 Conv
Stride 1, pad 1

4x4 MaxPool
Stride 4

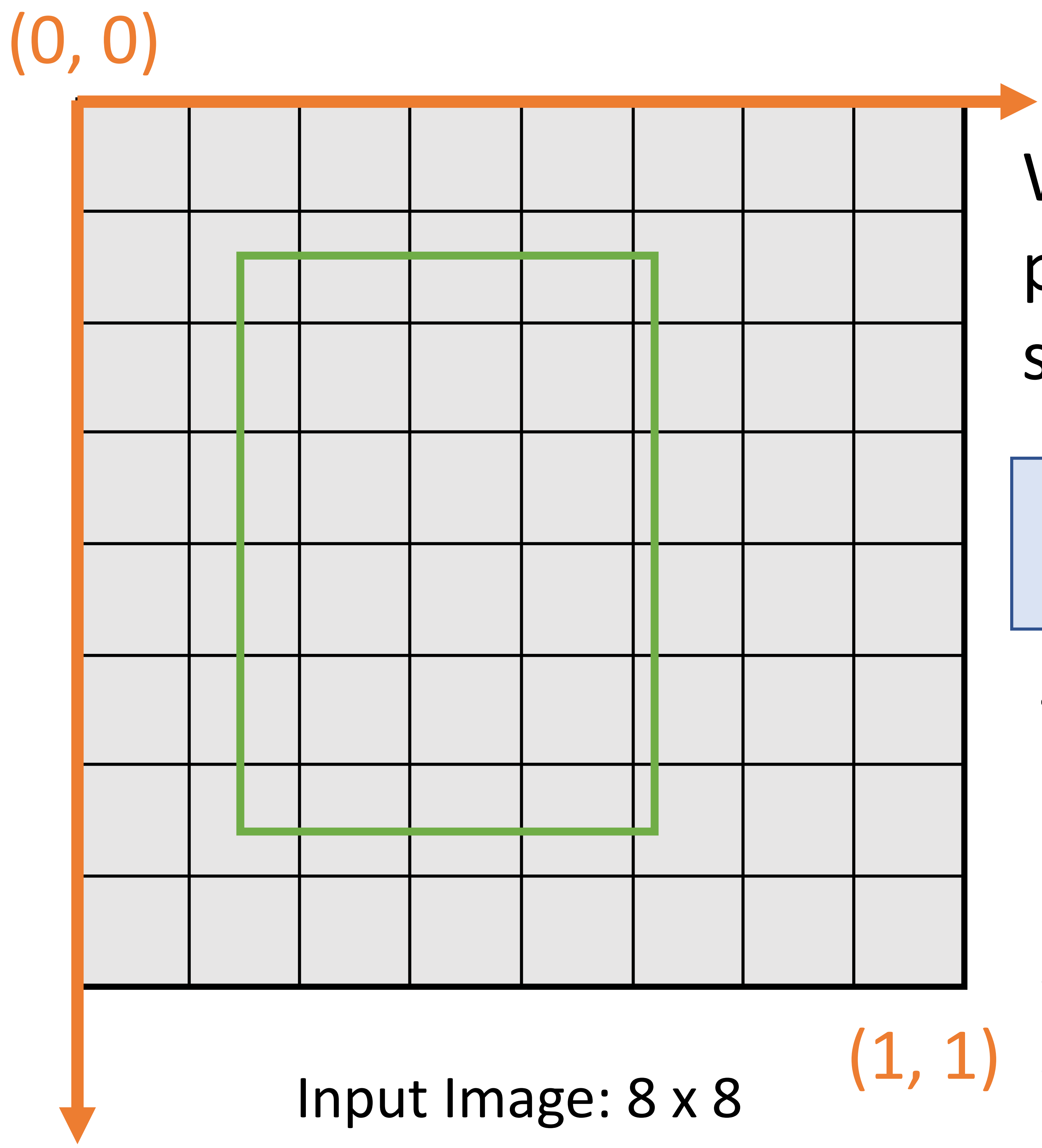
There is a correspondence between the coordinate system of the input and the coordinate system of the output



Output Image: 8 x 8

Projecting Points

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

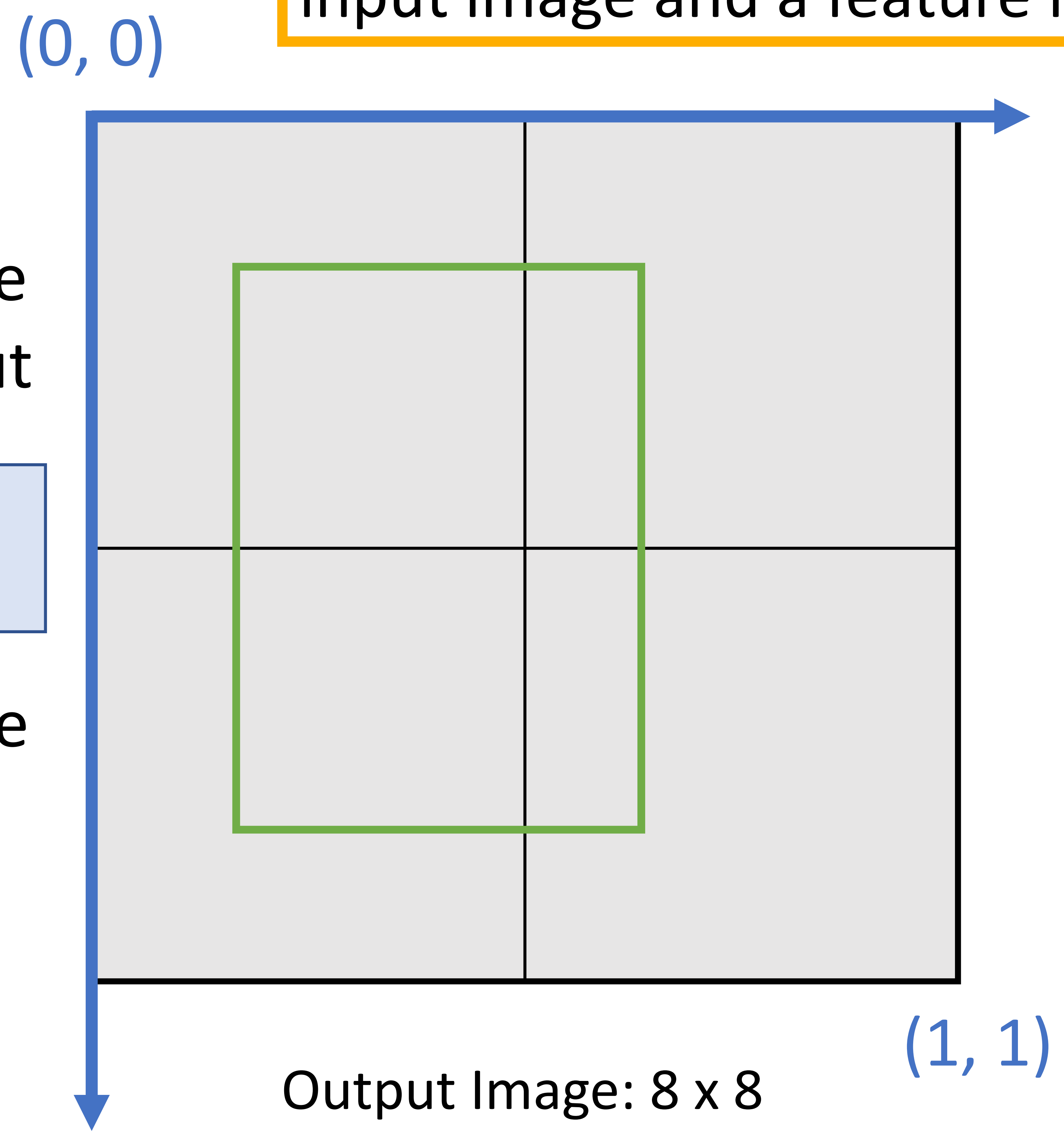


We can align arbitrary points between coordinate system of input and output

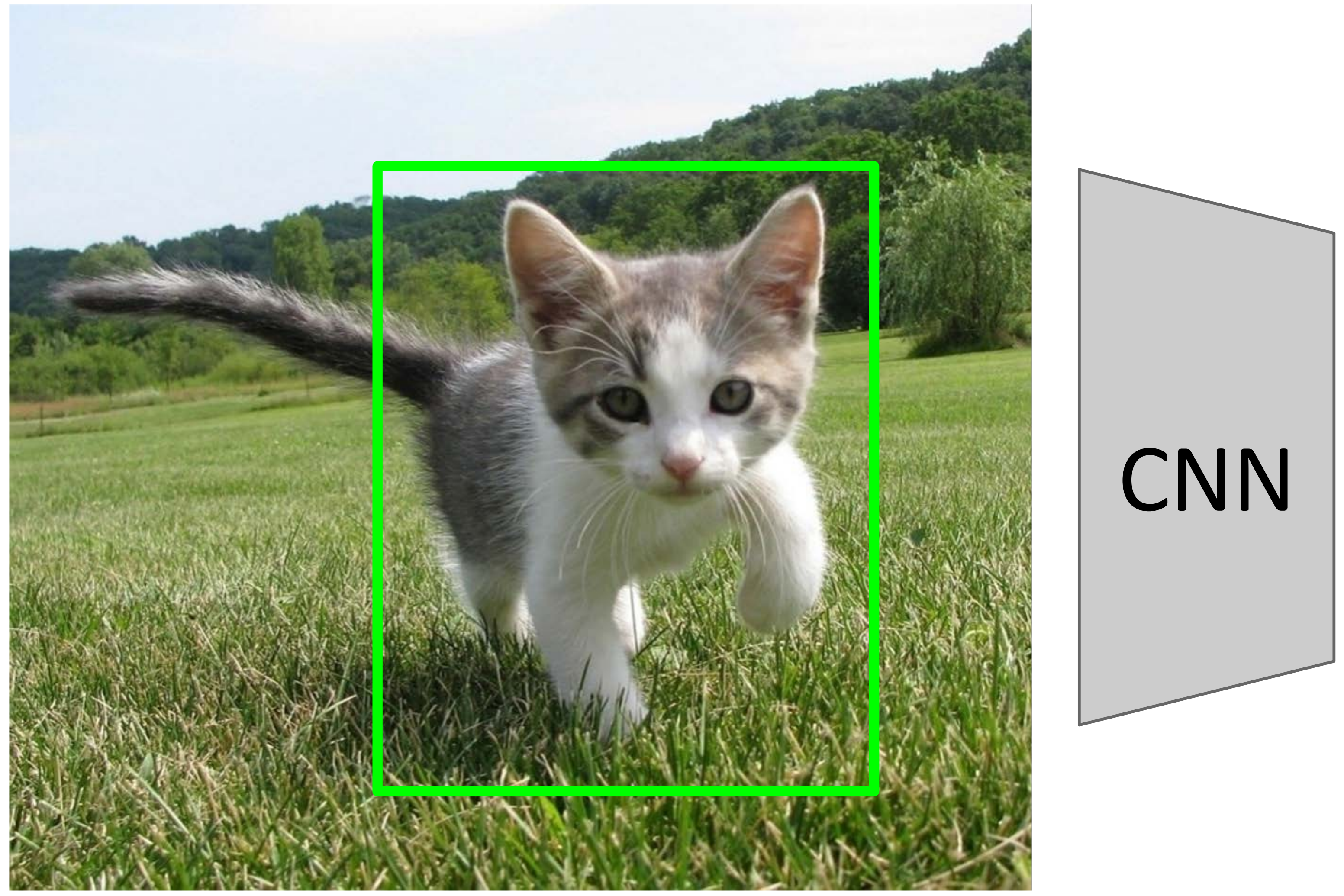
3x3 Conv
Stride 1, pad 1

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)

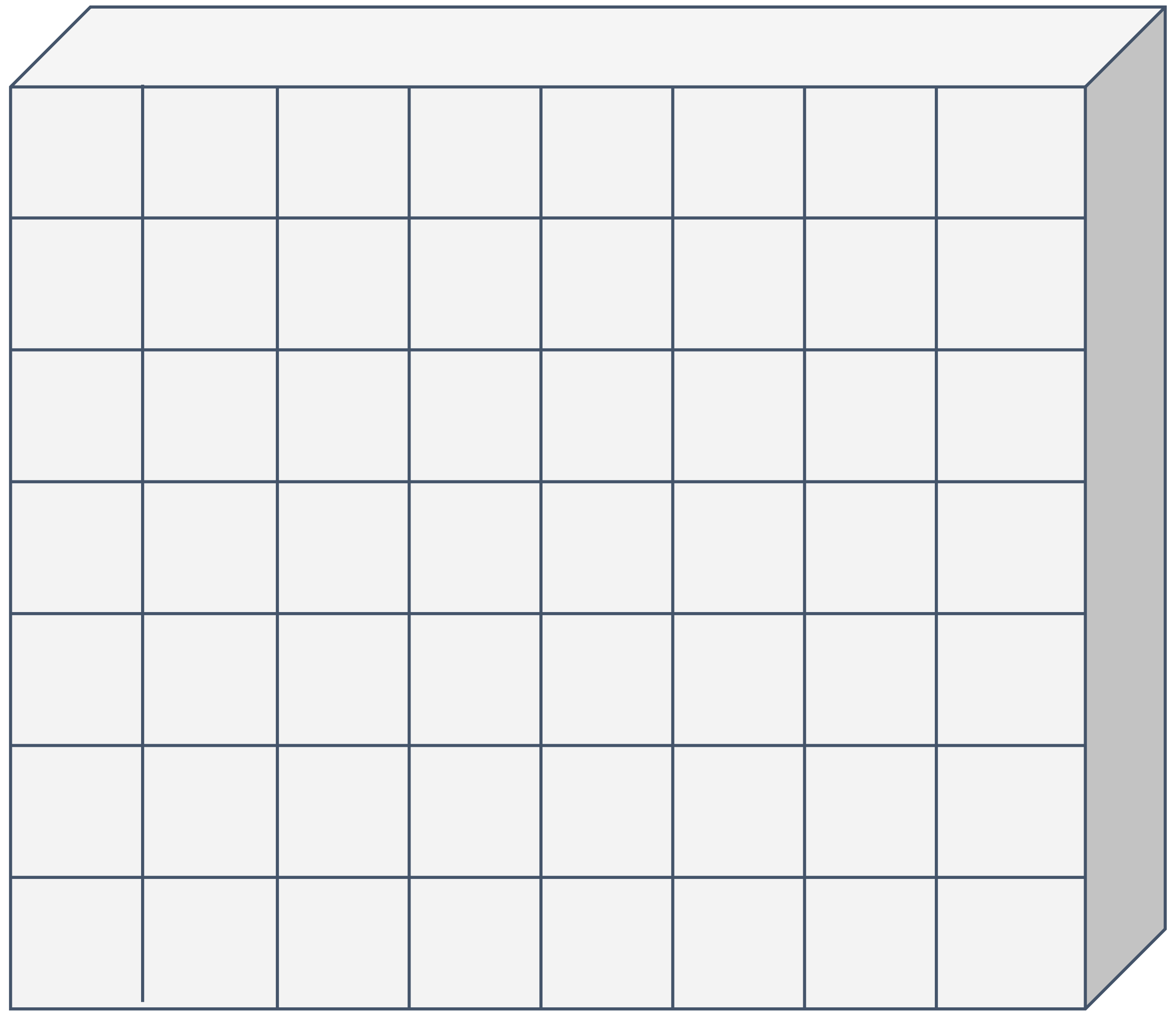
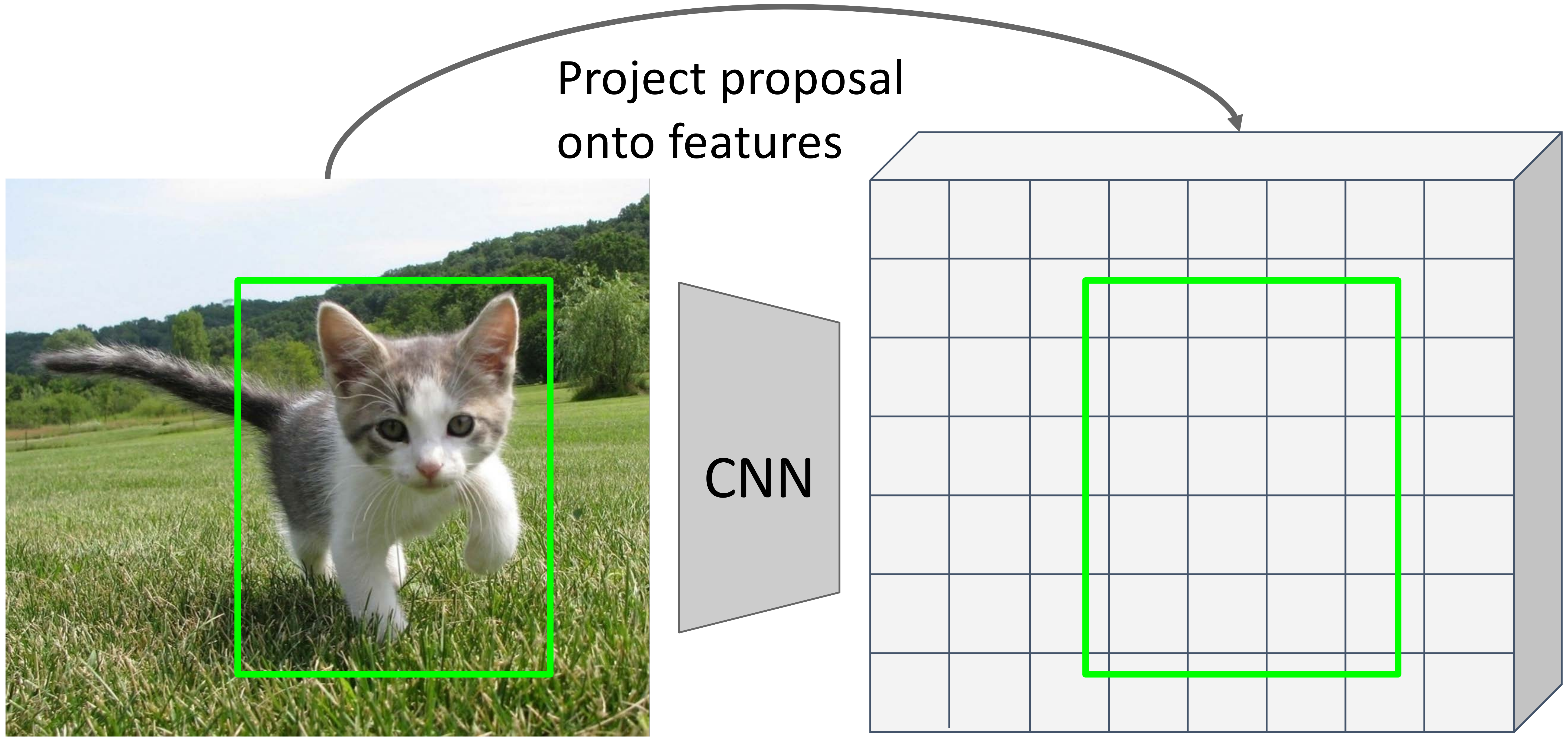


Image features
(e.g. 512 x 20 x 15)

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

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



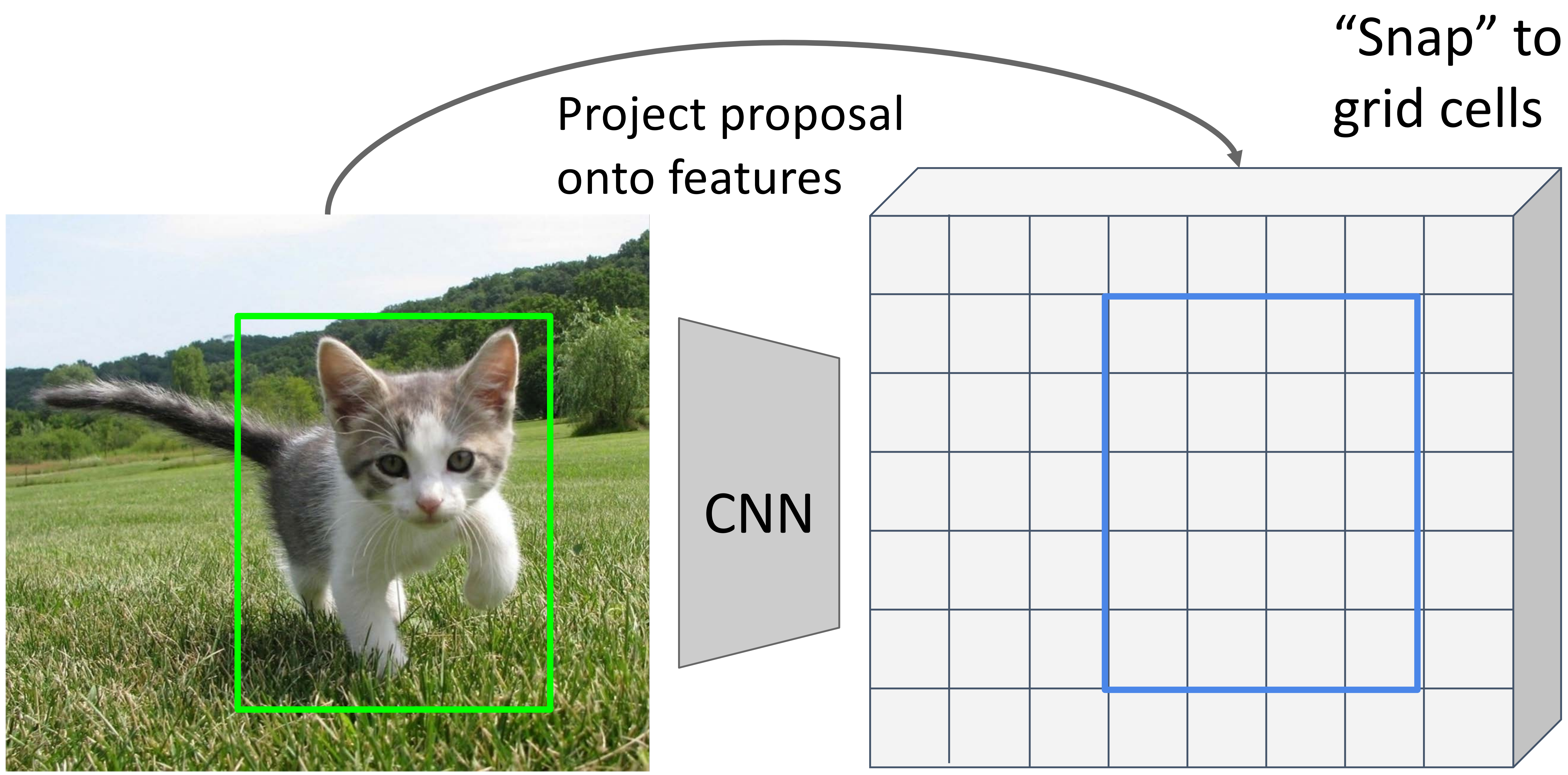
Input Image
(e.g. 3 x 640 x 480)

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

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

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



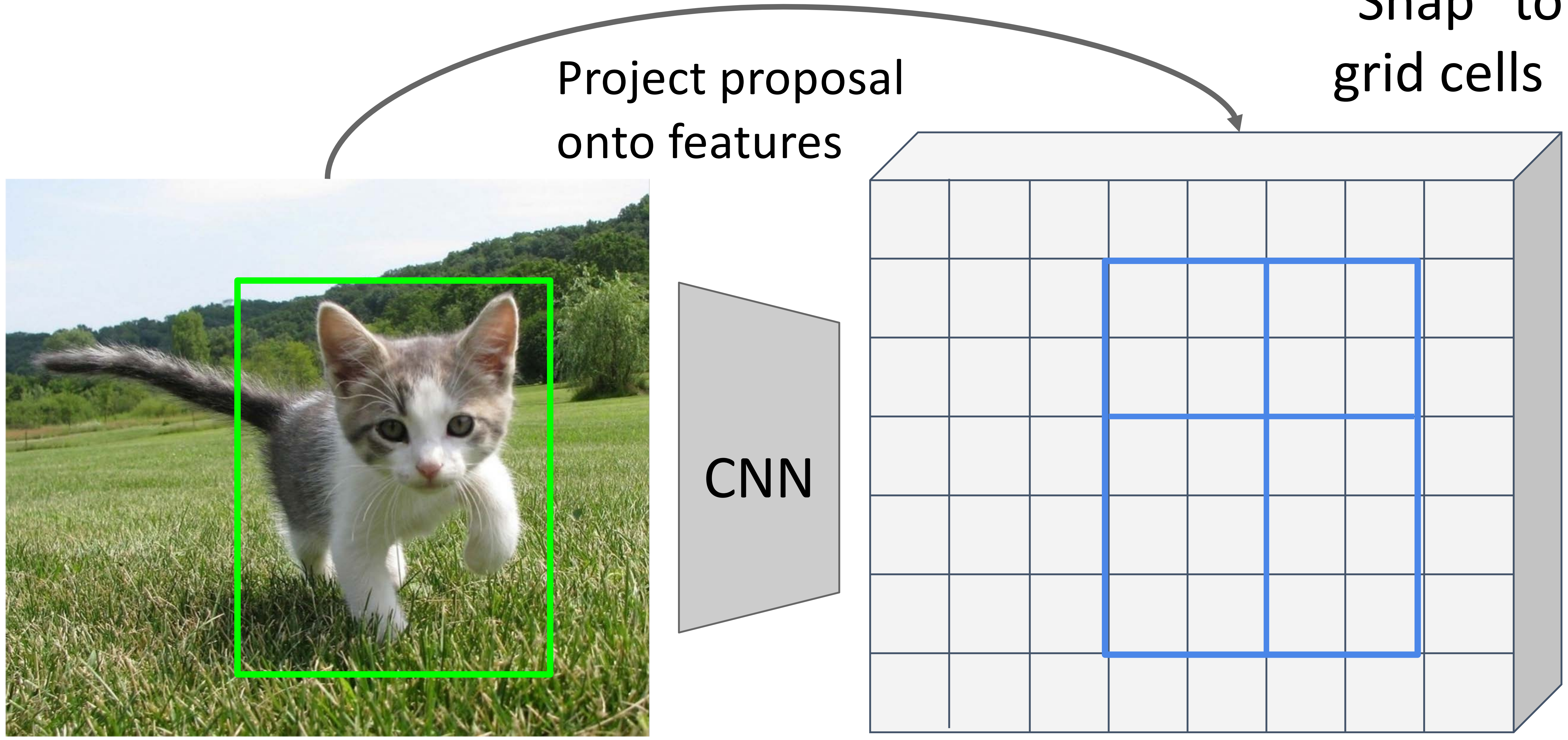
Input Image
(e.g. 3 x 640 x 480)

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

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

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

Cropping Features: RoI Pool



"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

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

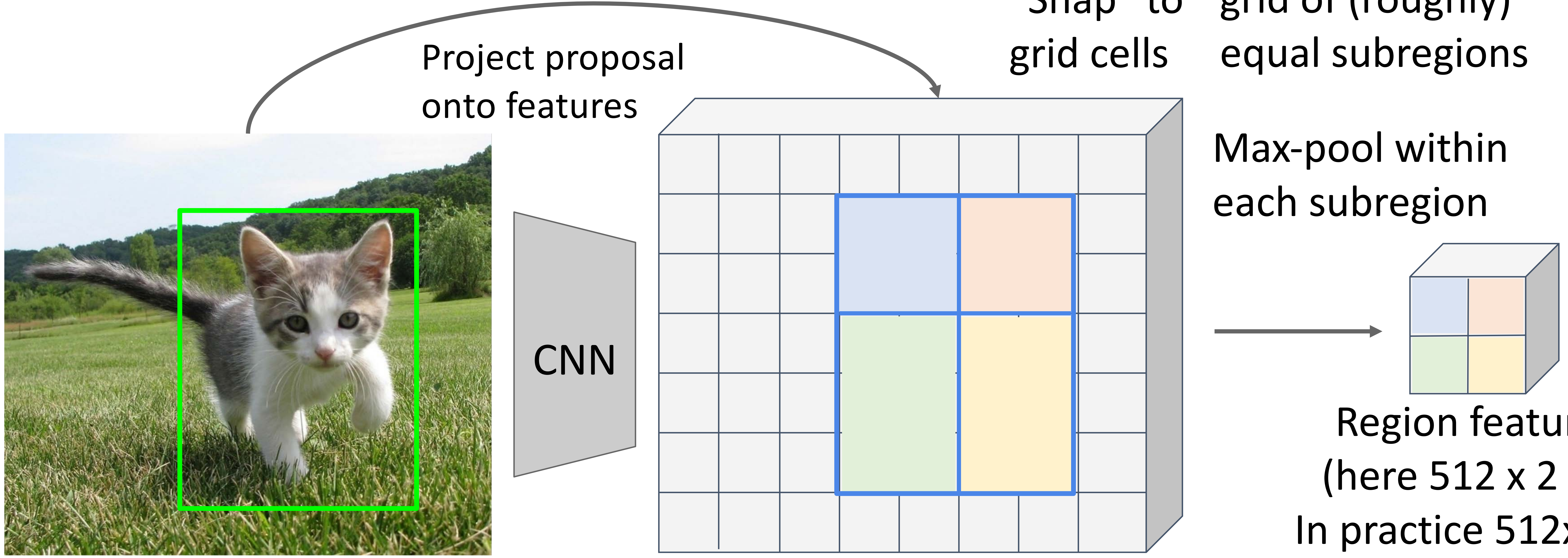
Input Image
(e.g. 3 x 640 x 480)

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

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

Cropping Features: RoI Pool

Divide into 2x2 grid of (roughly) equal subregions



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

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

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

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

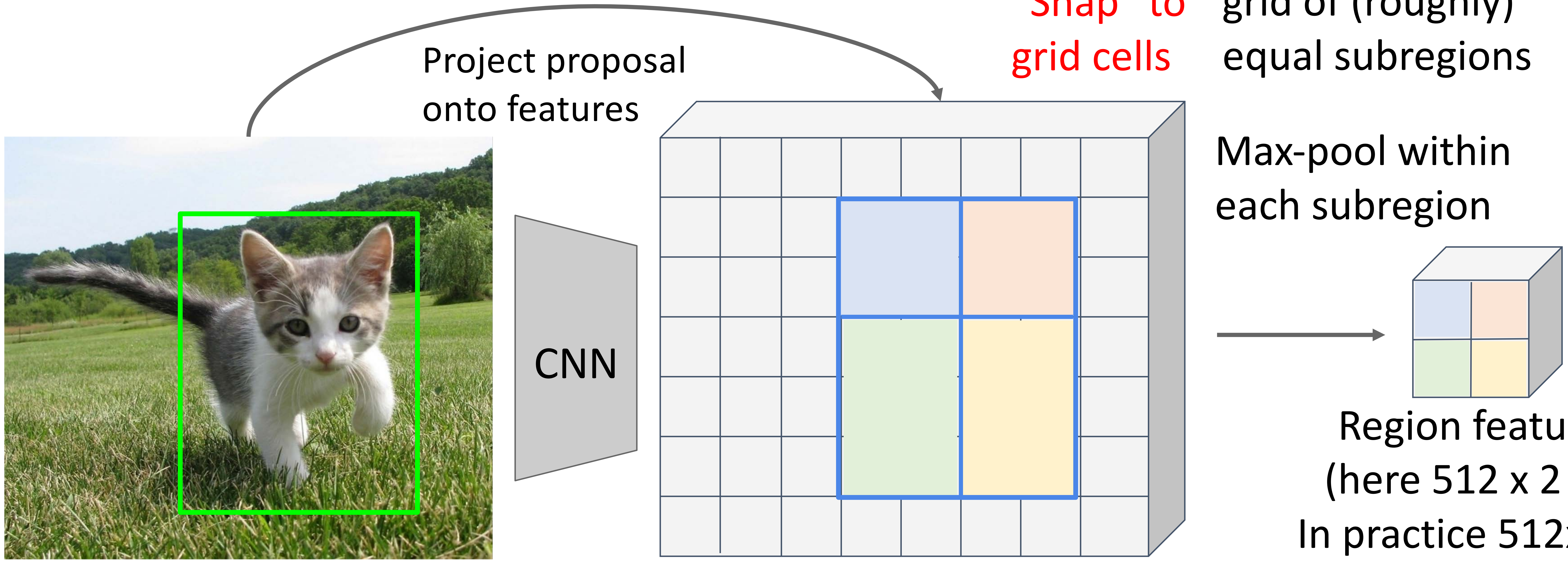
Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool

Divide into 2x2 grid of (roughly) equal subregions

“Snap” to grid cells

Max-pool within each subregion



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

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

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

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

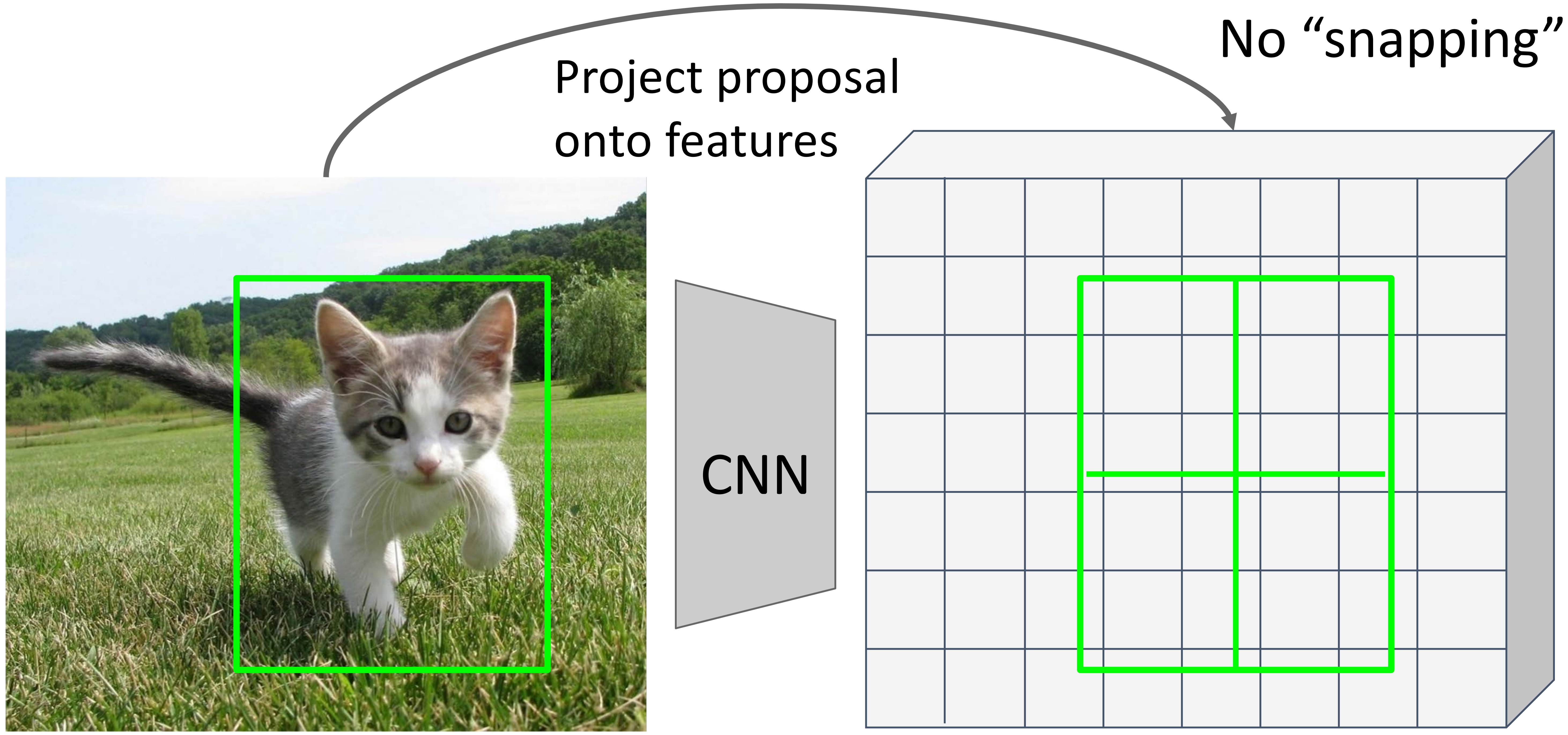
Problem: Slight misalignment due to snapping; different-sized subregions is weird

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Align

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

No "snapping"!



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

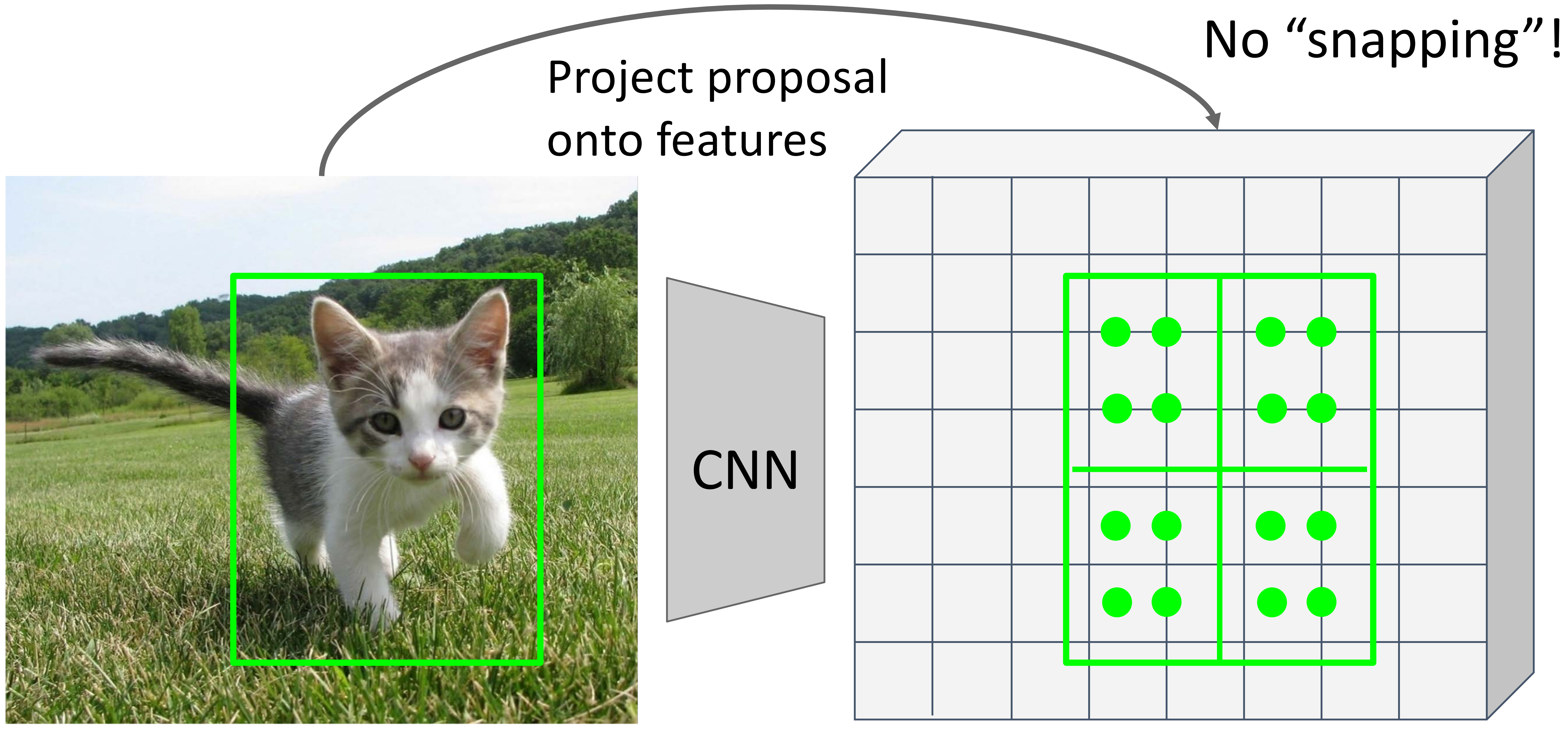
Image features
(e.g. 512 x 20 x 15)

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

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

Cropping Features: RoI Align

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



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

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

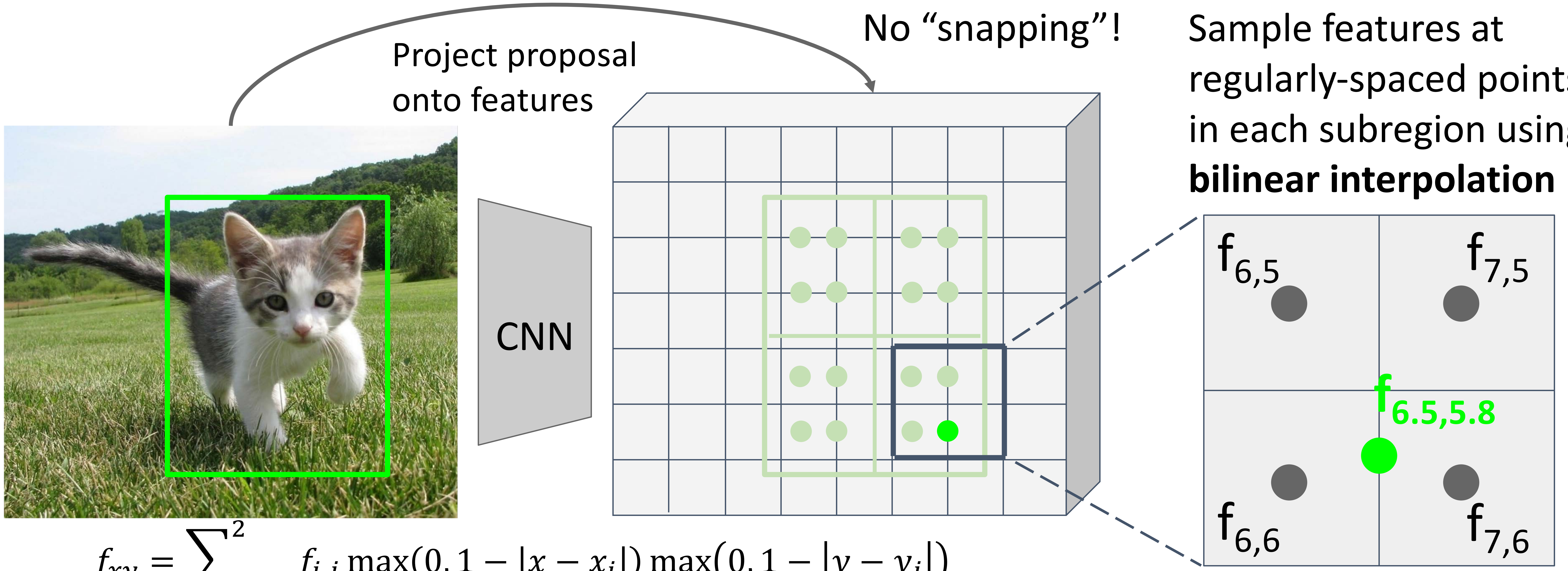
Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

Cropping Features: RoI Align

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

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



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

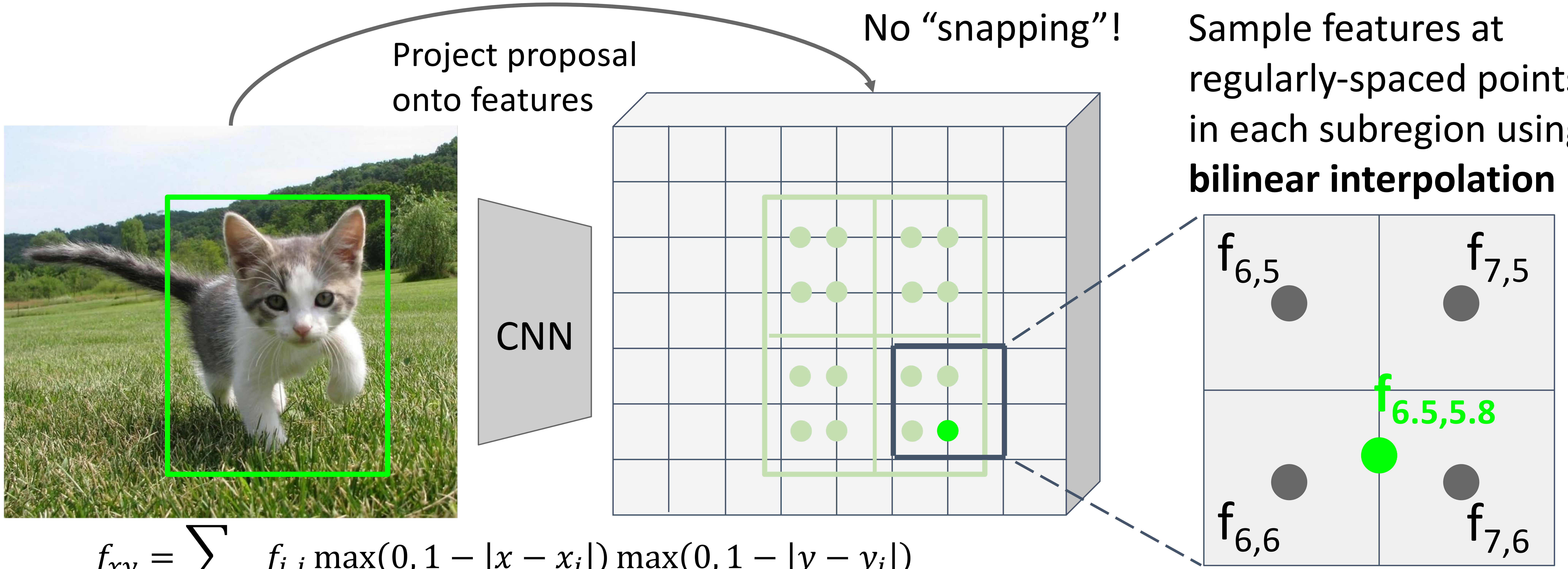
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

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

Cropping Features: RoI Align

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

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



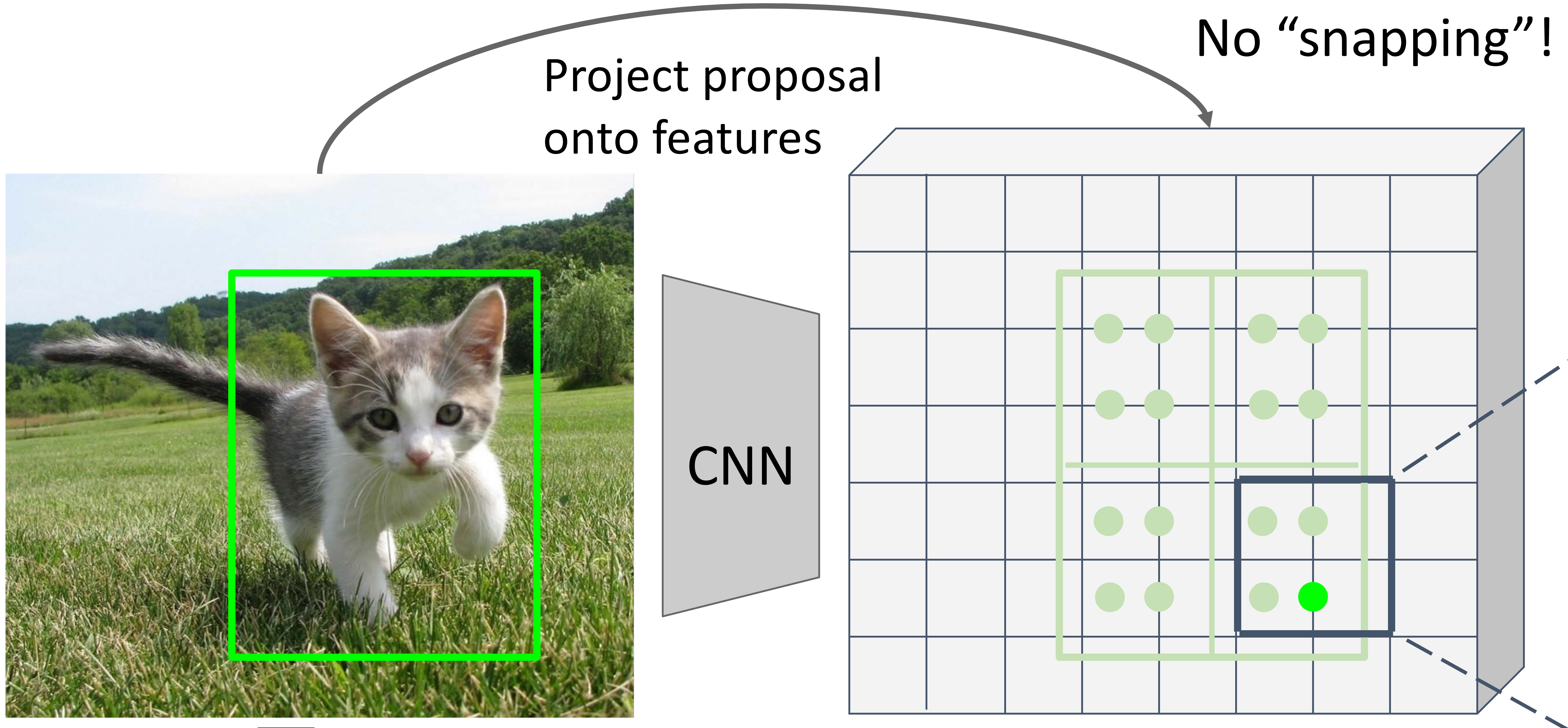
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

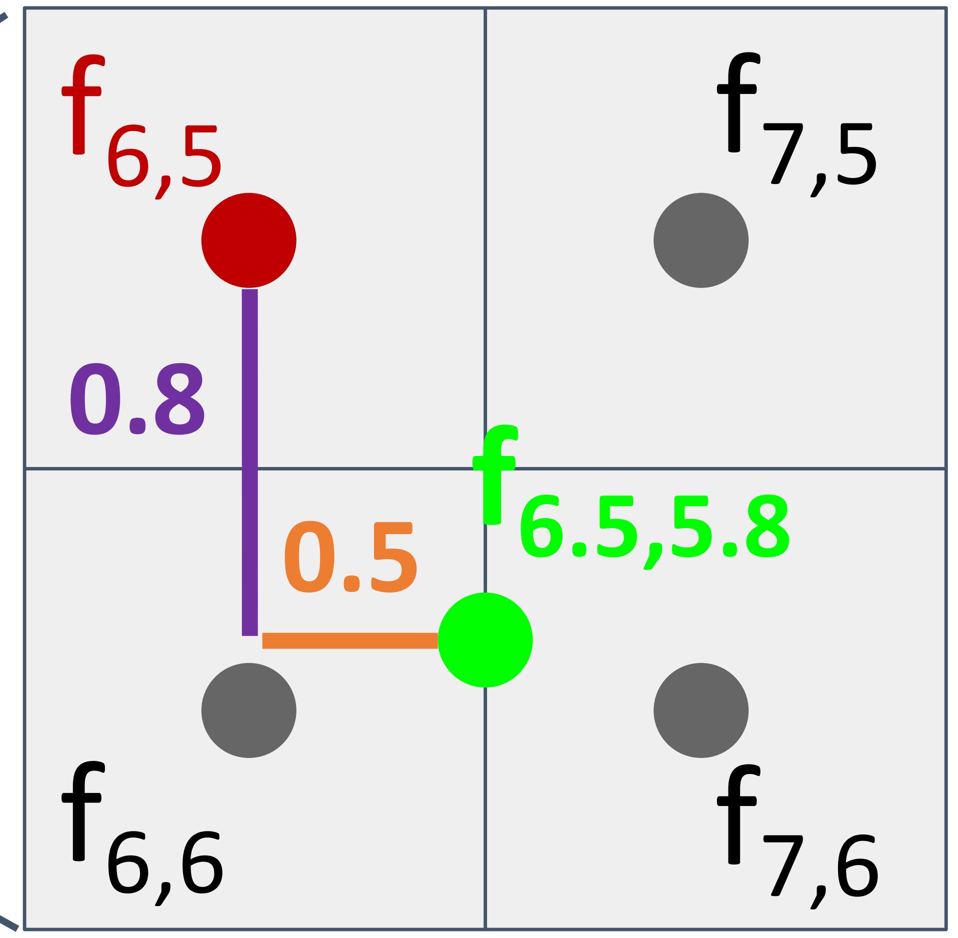
$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align



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



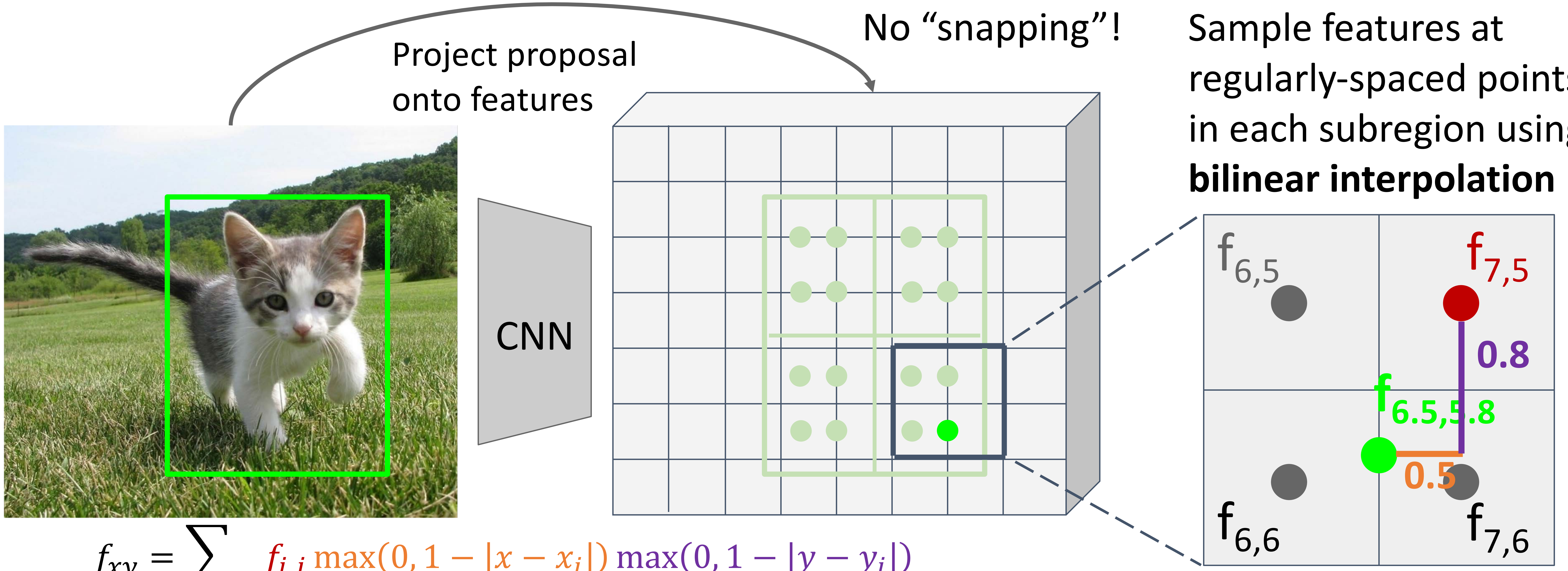
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

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

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

Cropping Features: RoI Align



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

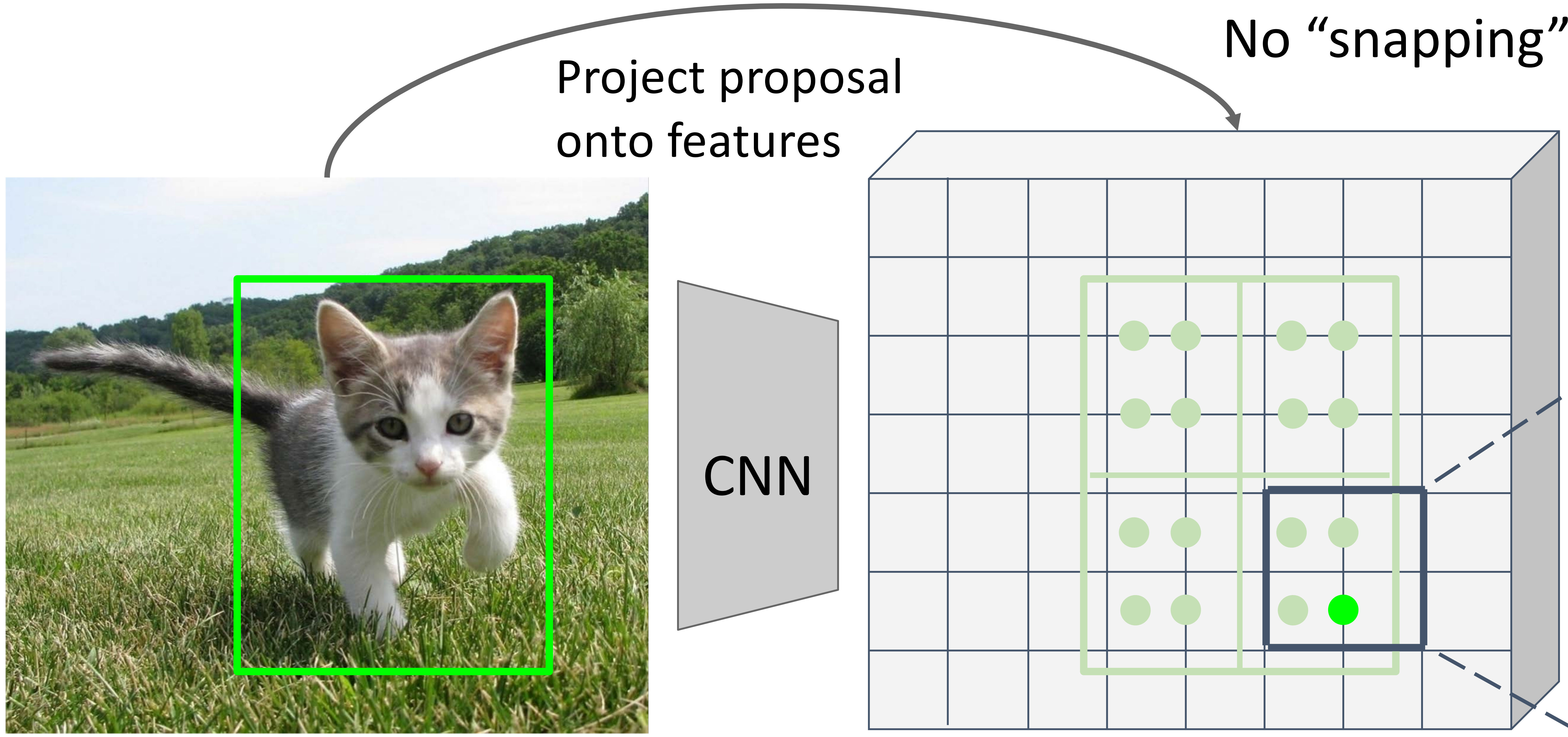
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

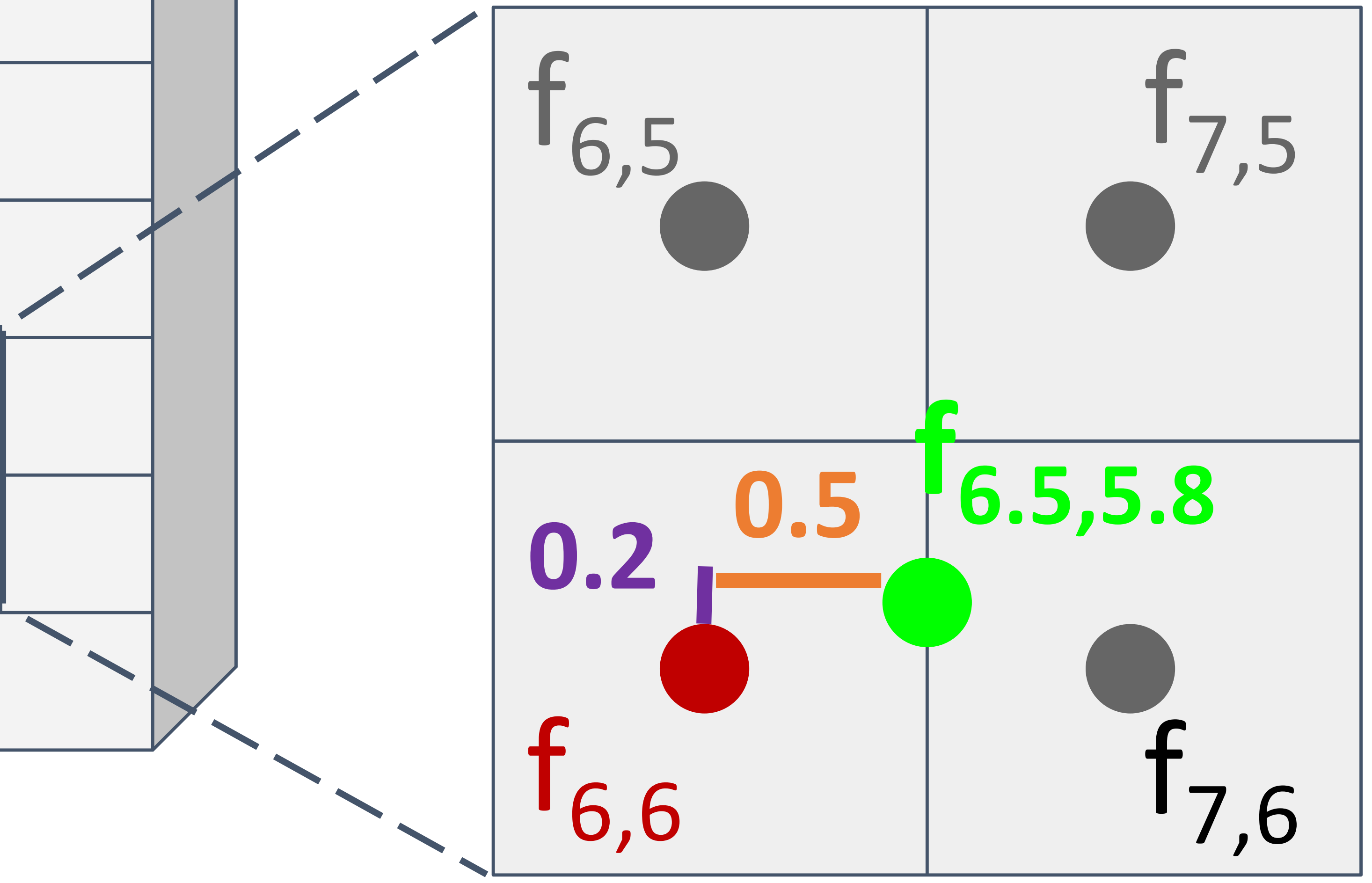
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

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

Cropping Features: RoI Align



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



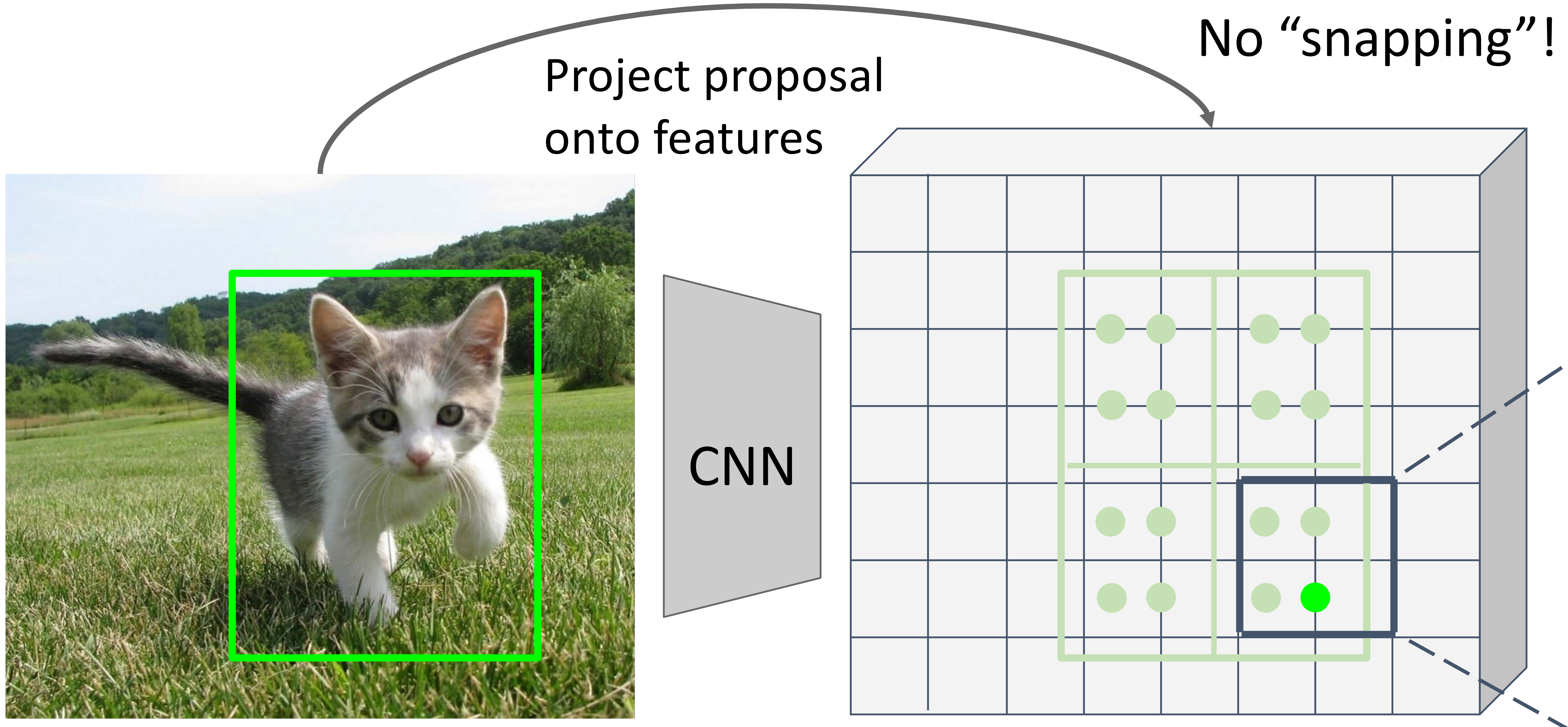
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

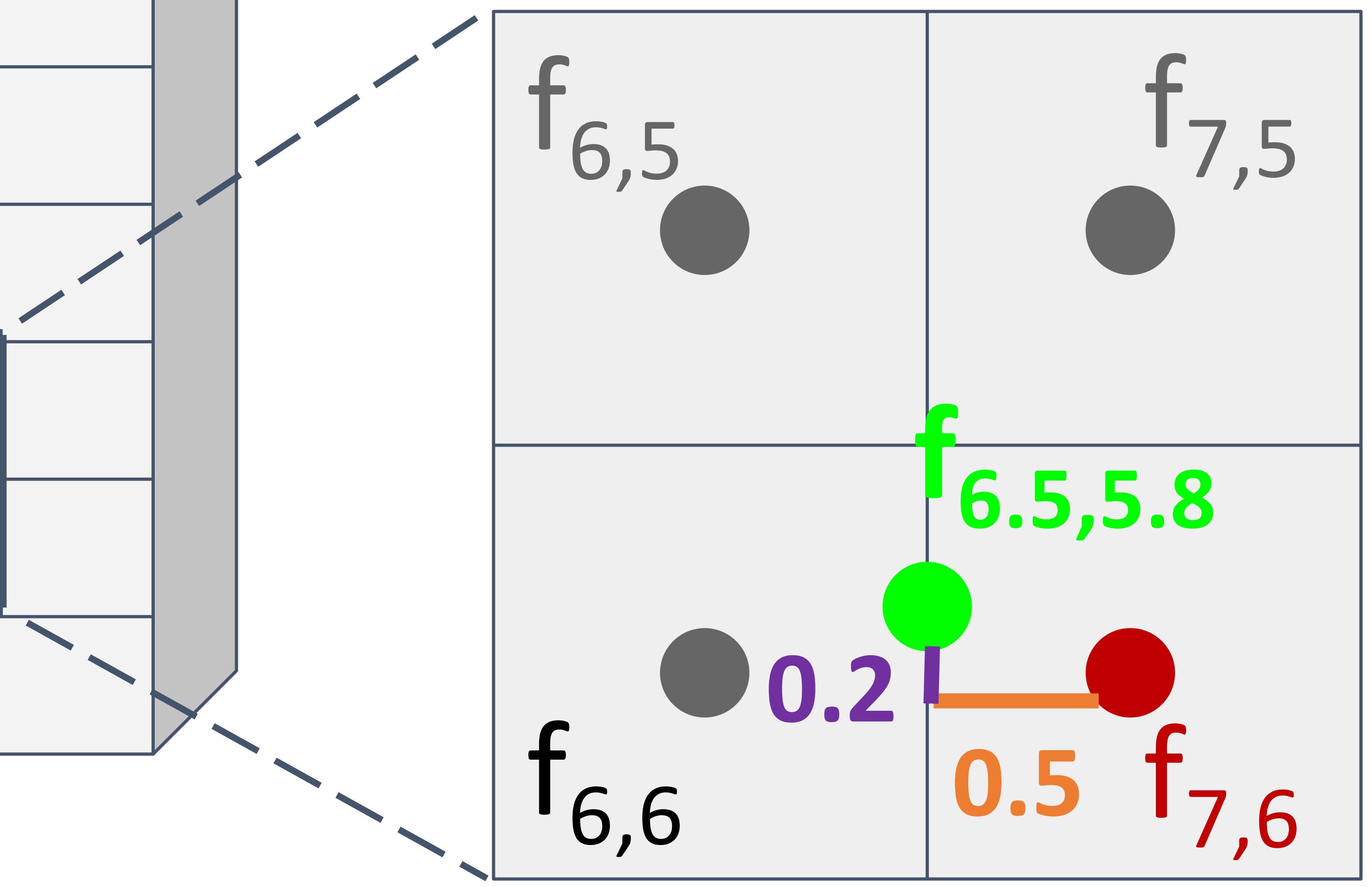
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align



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



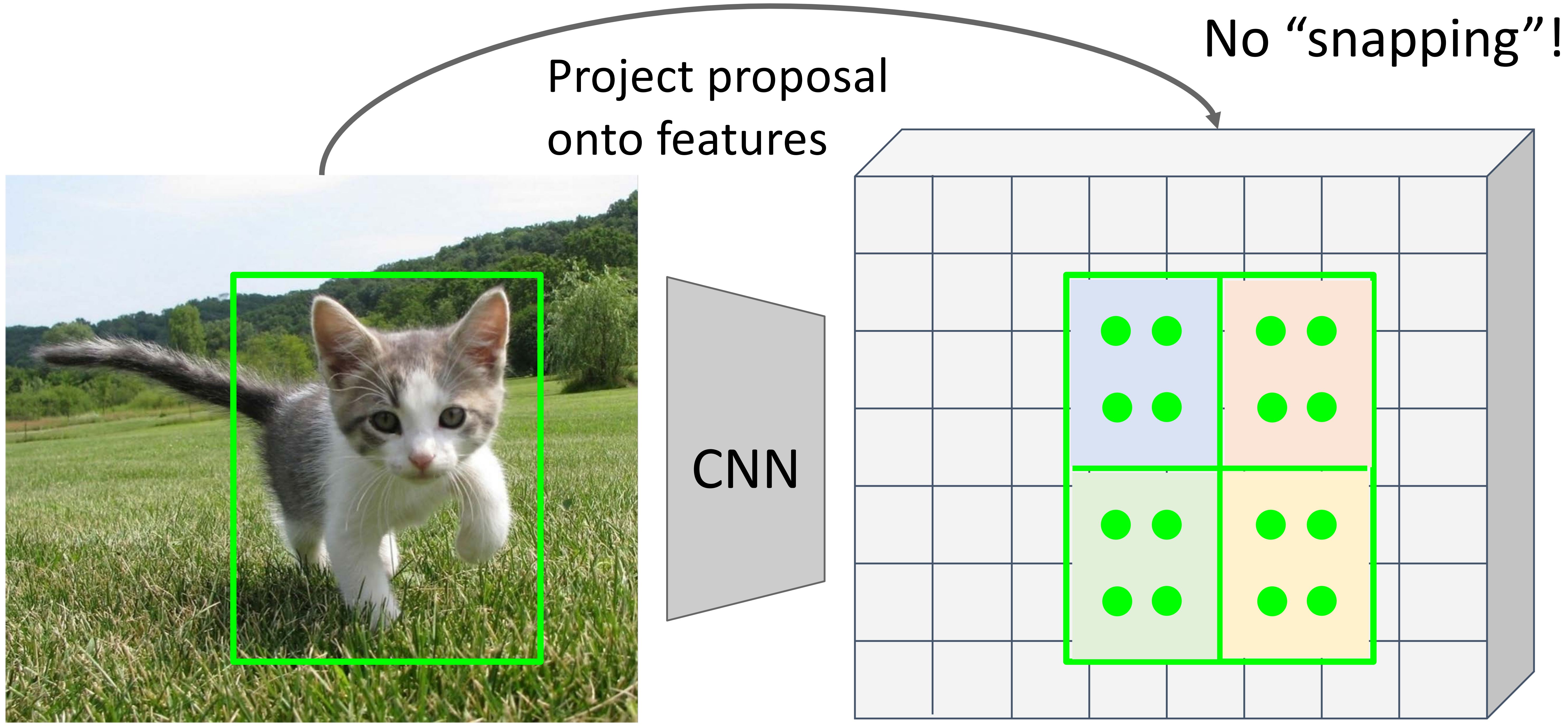
$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

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

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

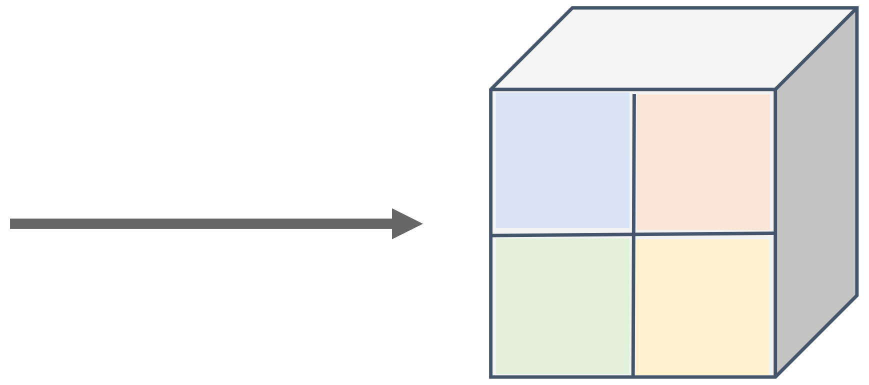
He et al, "Mask R-CNN", ICCV 2017

Cropping Features: RoI Align



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

After sampling, max-pool in each subregion



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

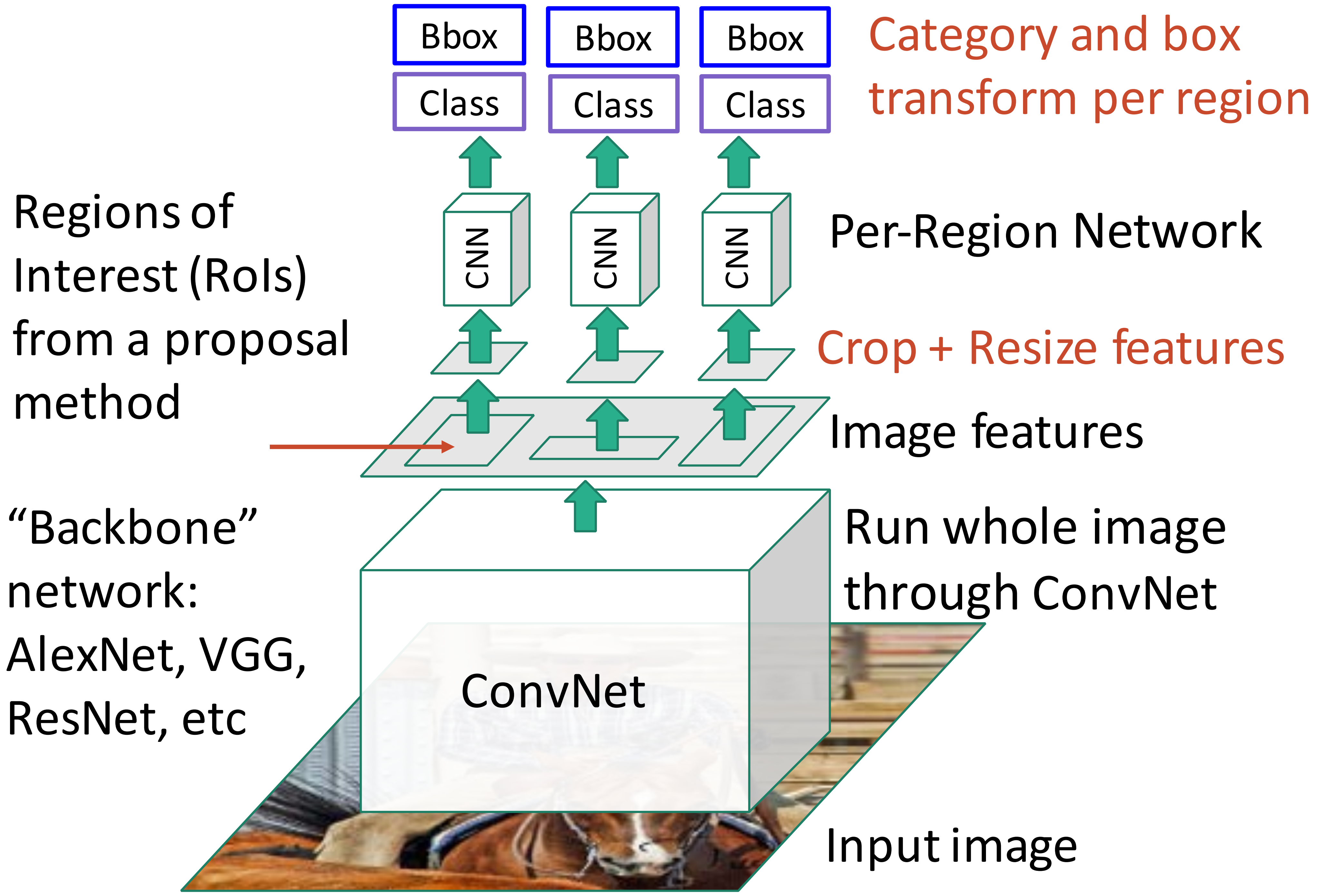
Image features
(e.g. 512 x 20 x 15)

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

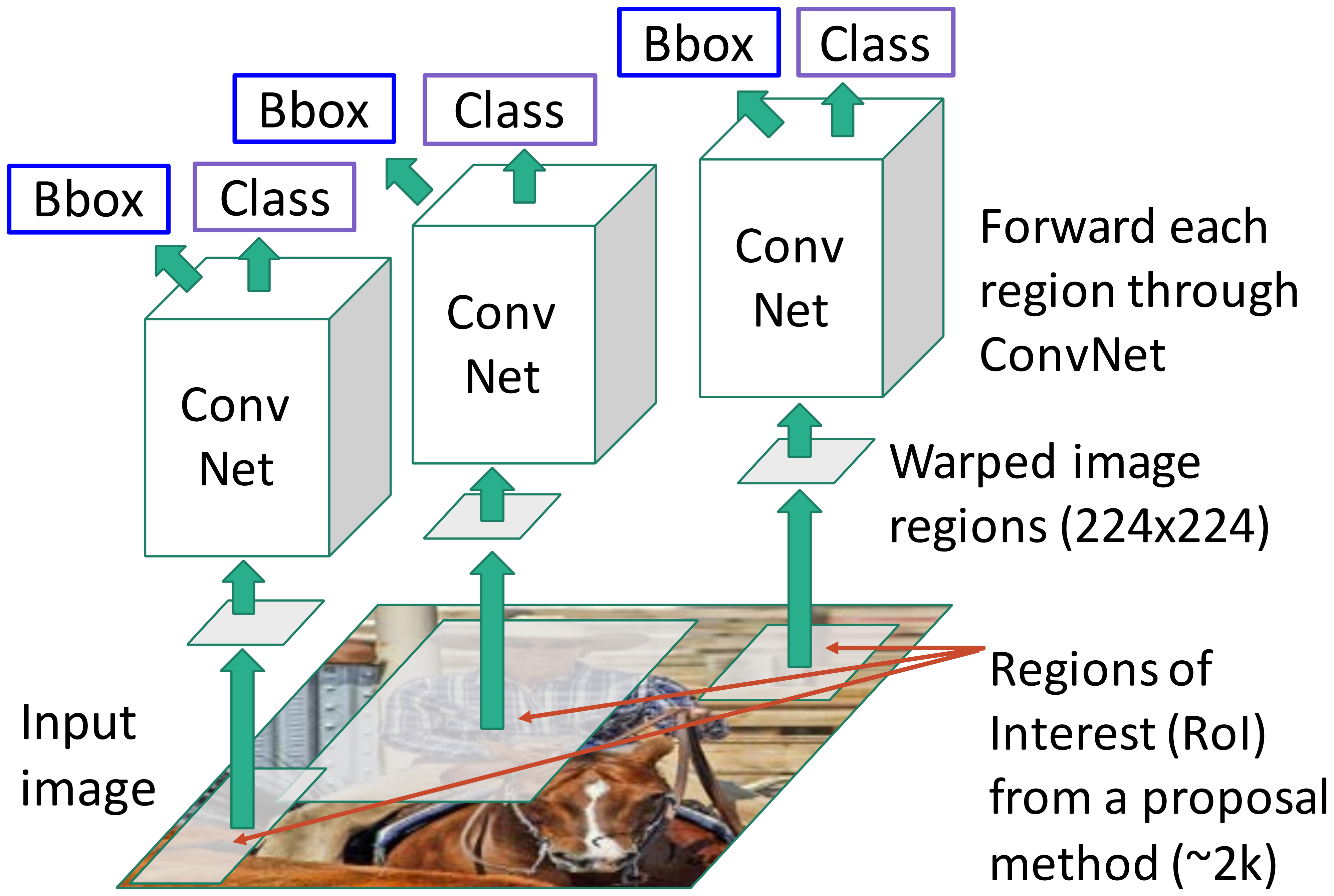
He et al, "Mask R-CNN", ICCV 2017

Fast R-CNN vs “Slow” R-CNN

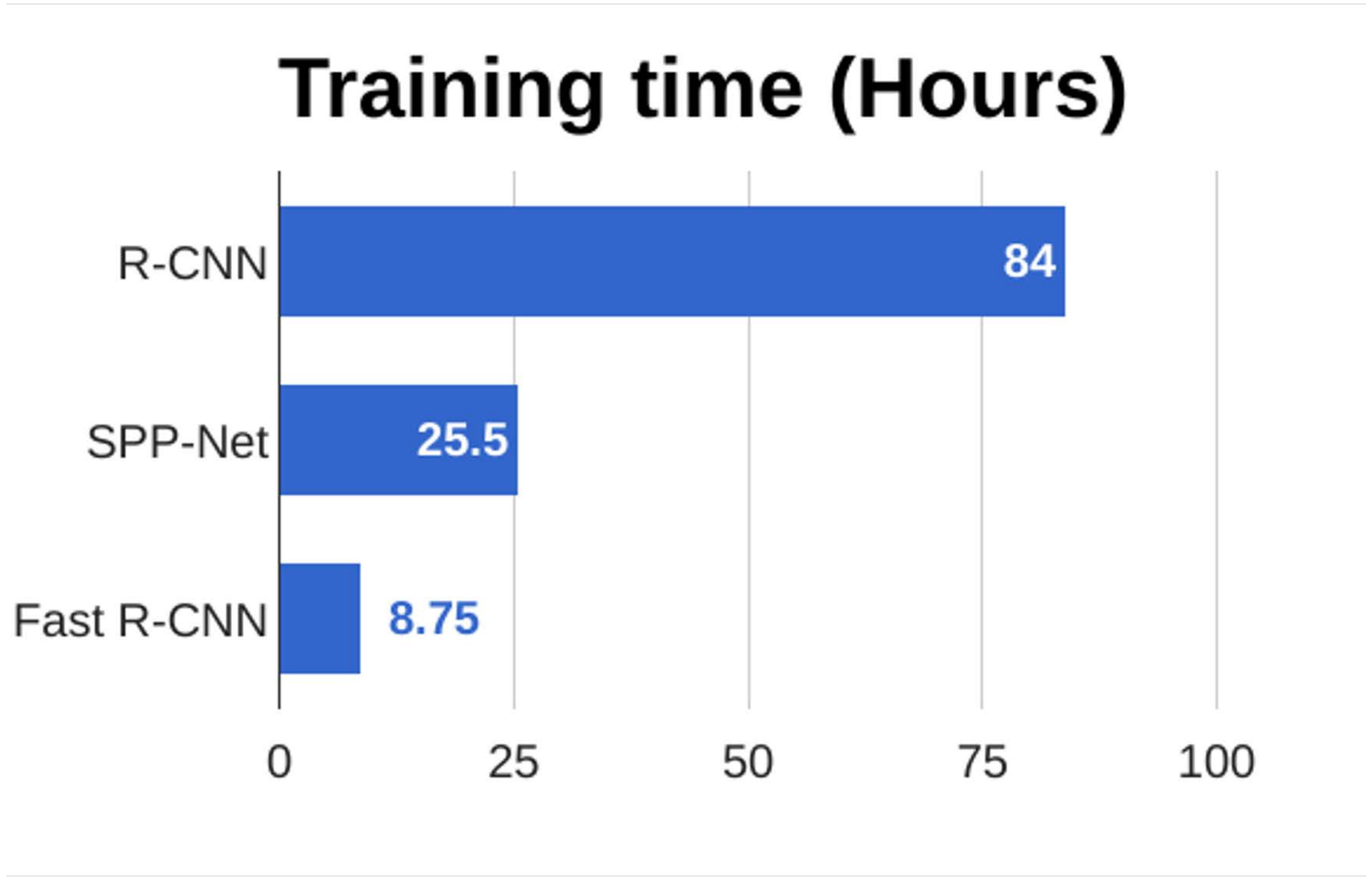
Fast R-CNN: Apply differentiable cropping to shared image features

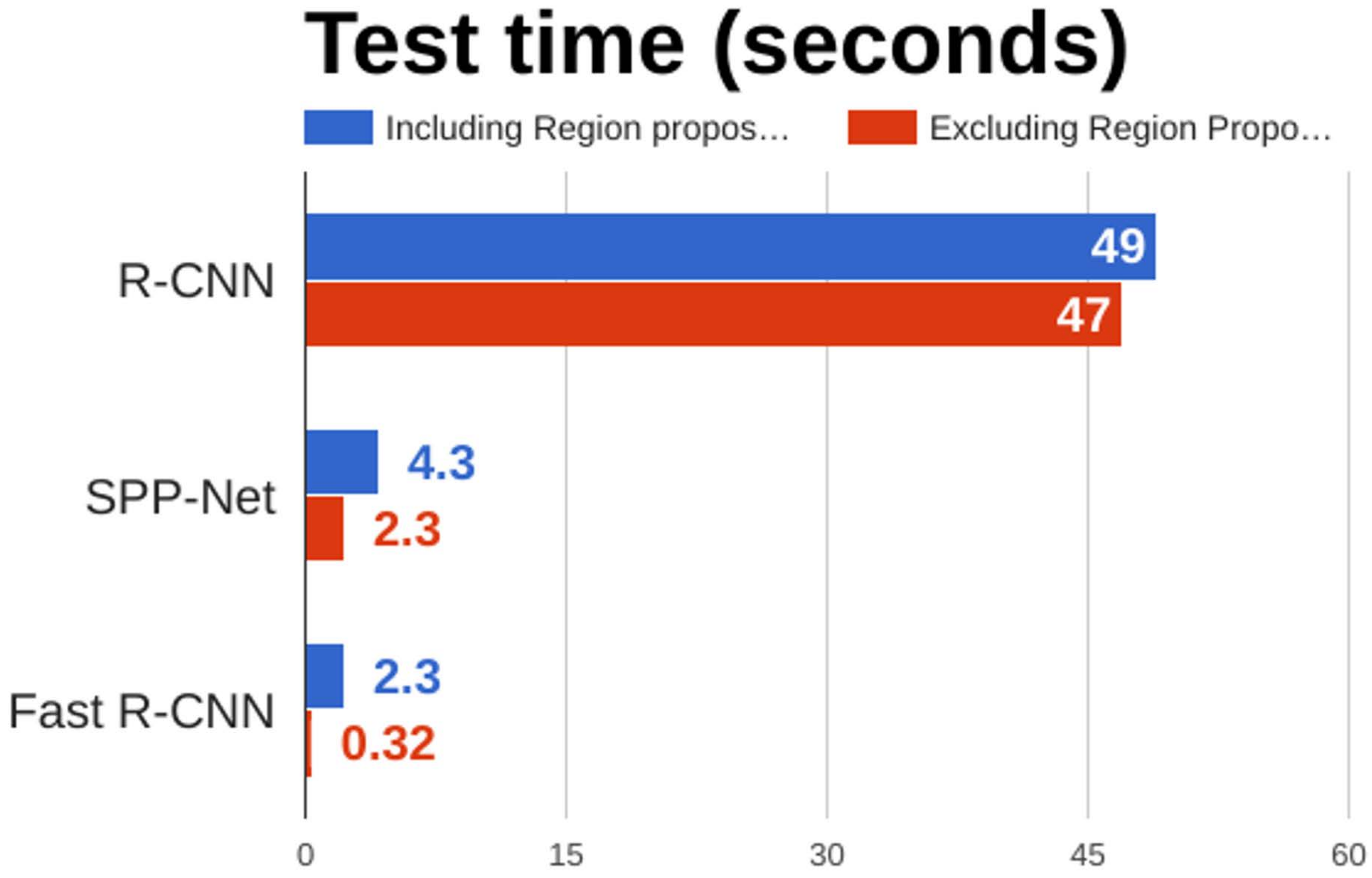


“Slow” R-CNN: Apply differentiable cropping to shared image features



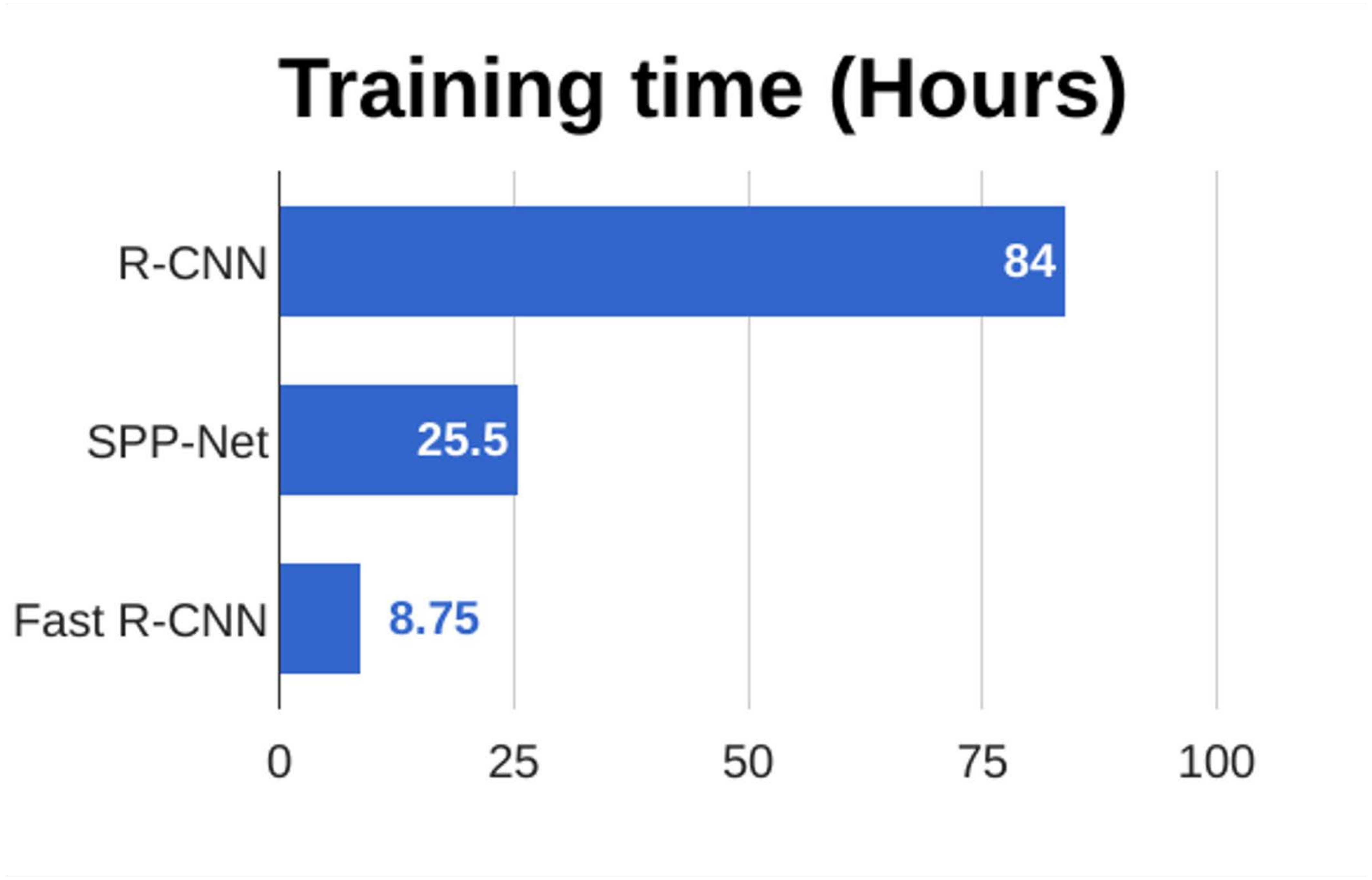
Fast R-CNN vs “Slow” R-CNN

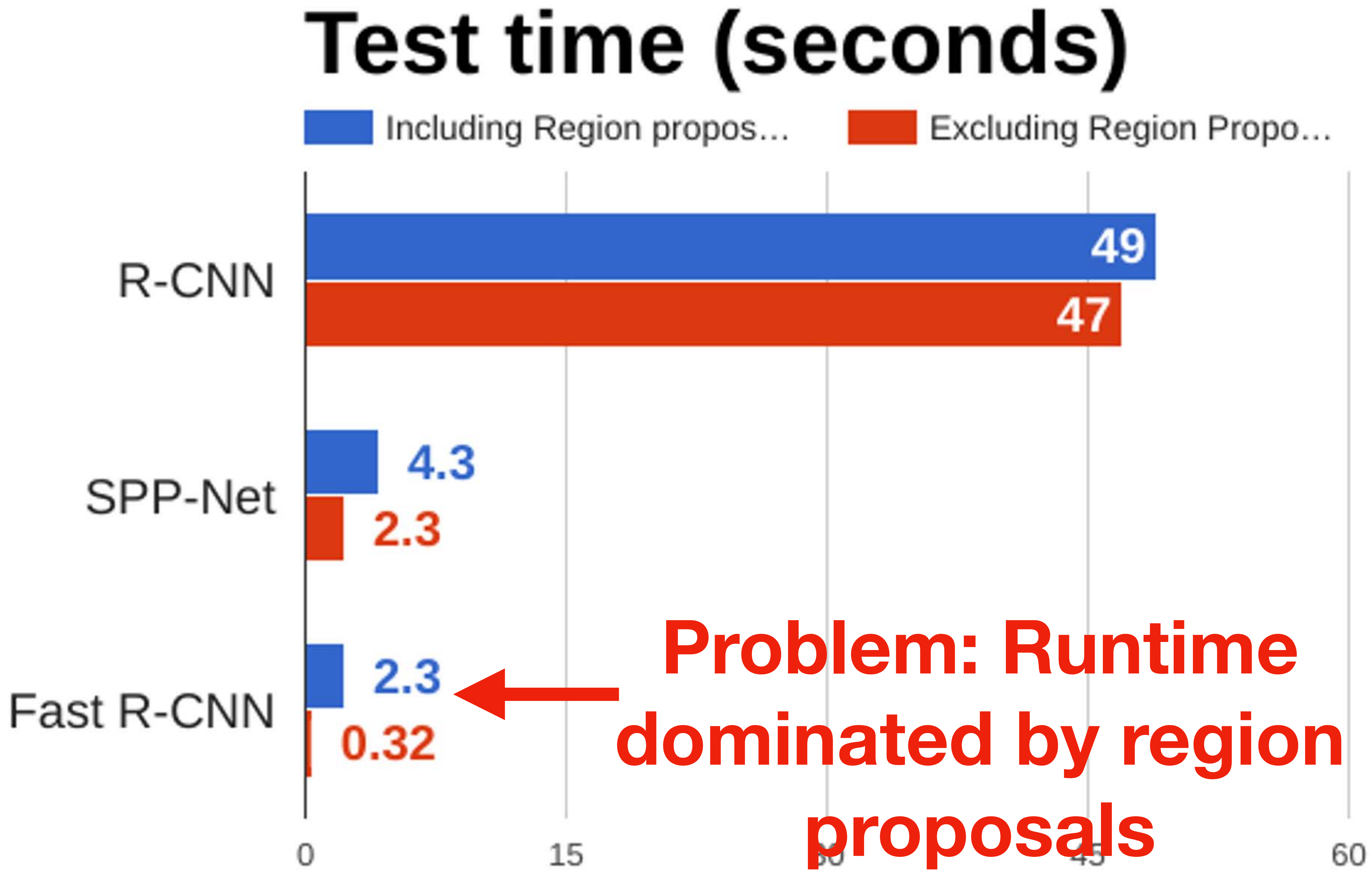




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

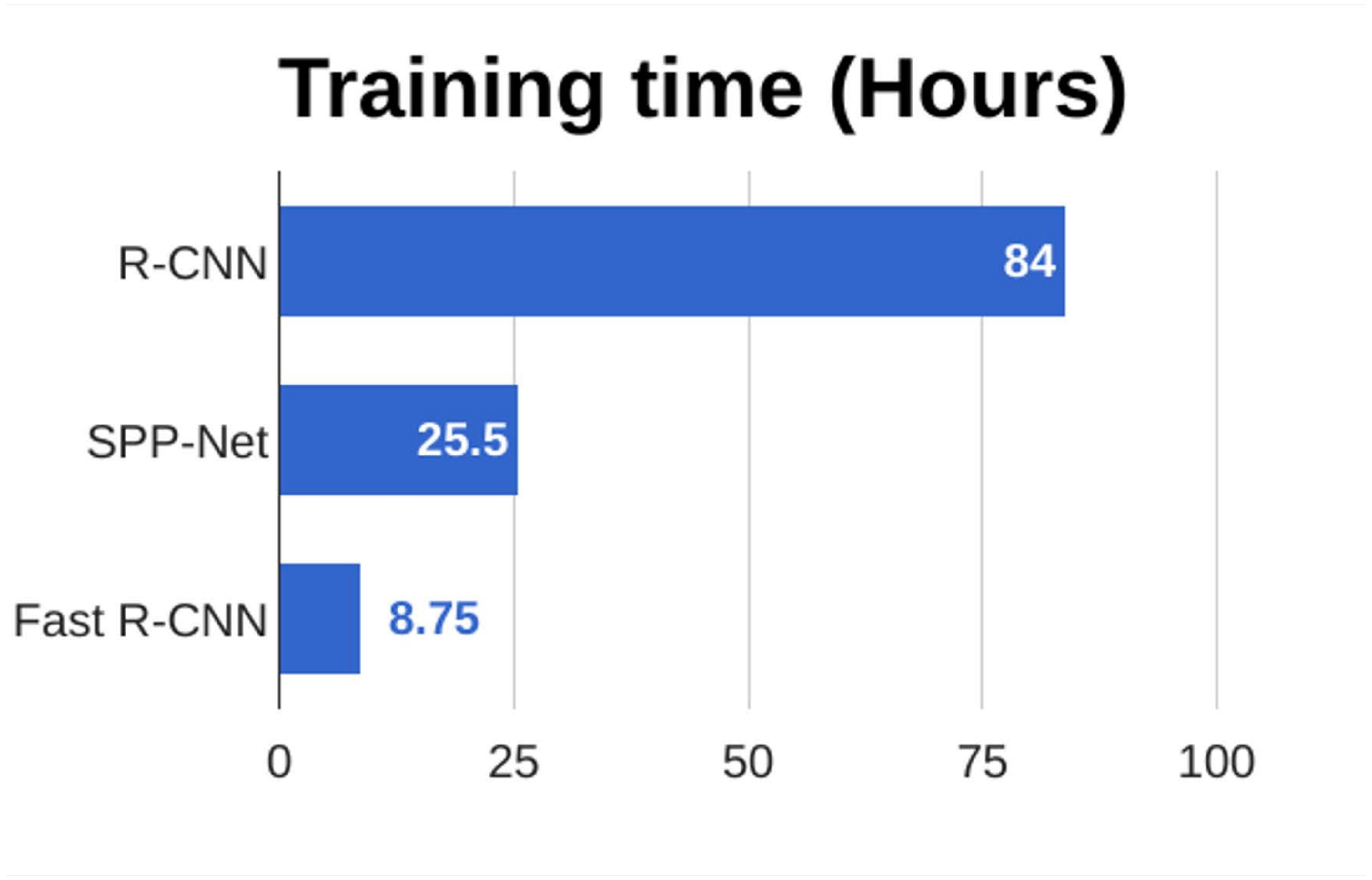
Fast R-CNN vs “Slow” R-CNN

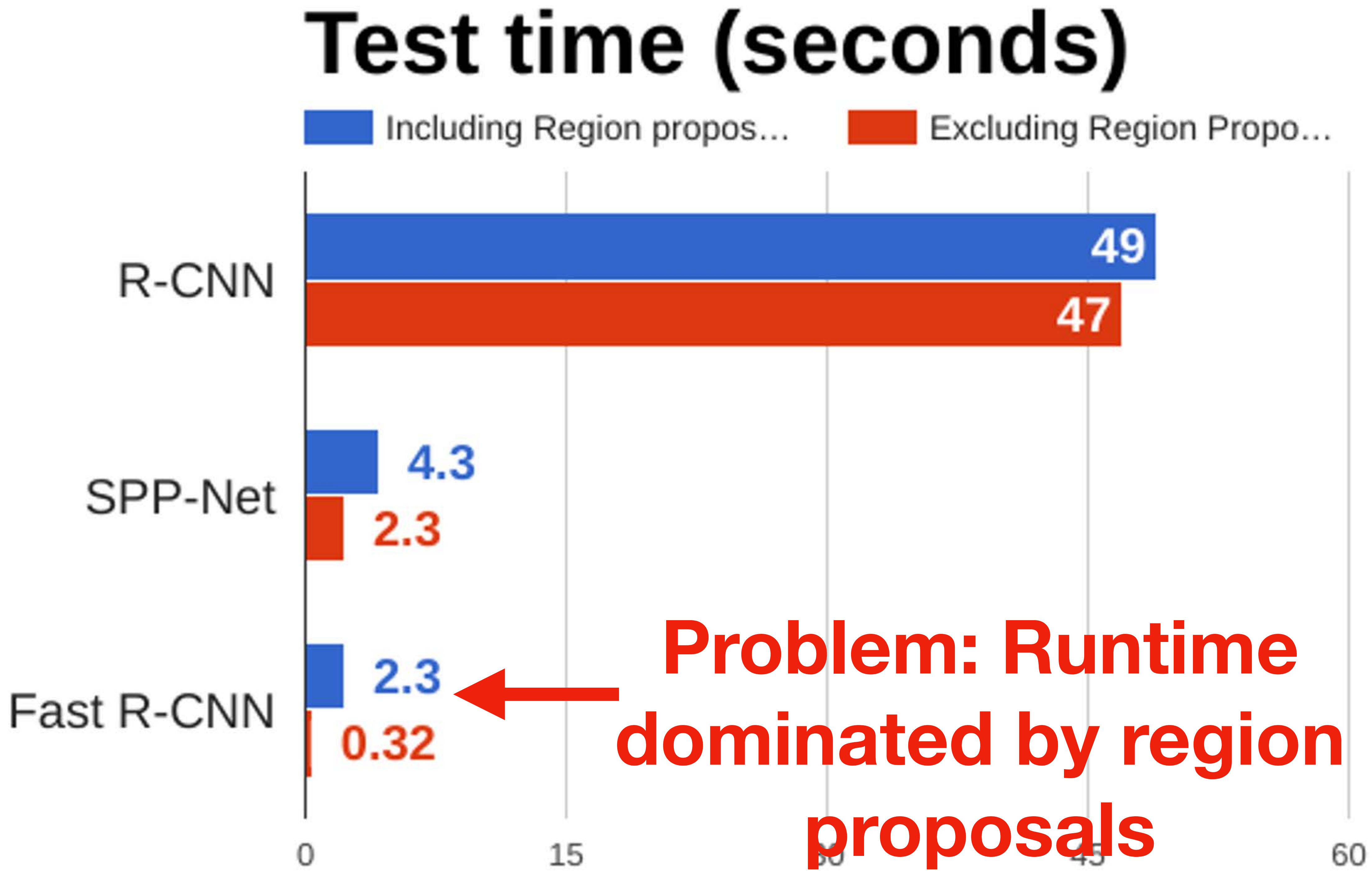




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs "Slow" R-CNN





Problem: Runtime dominated by region proposals

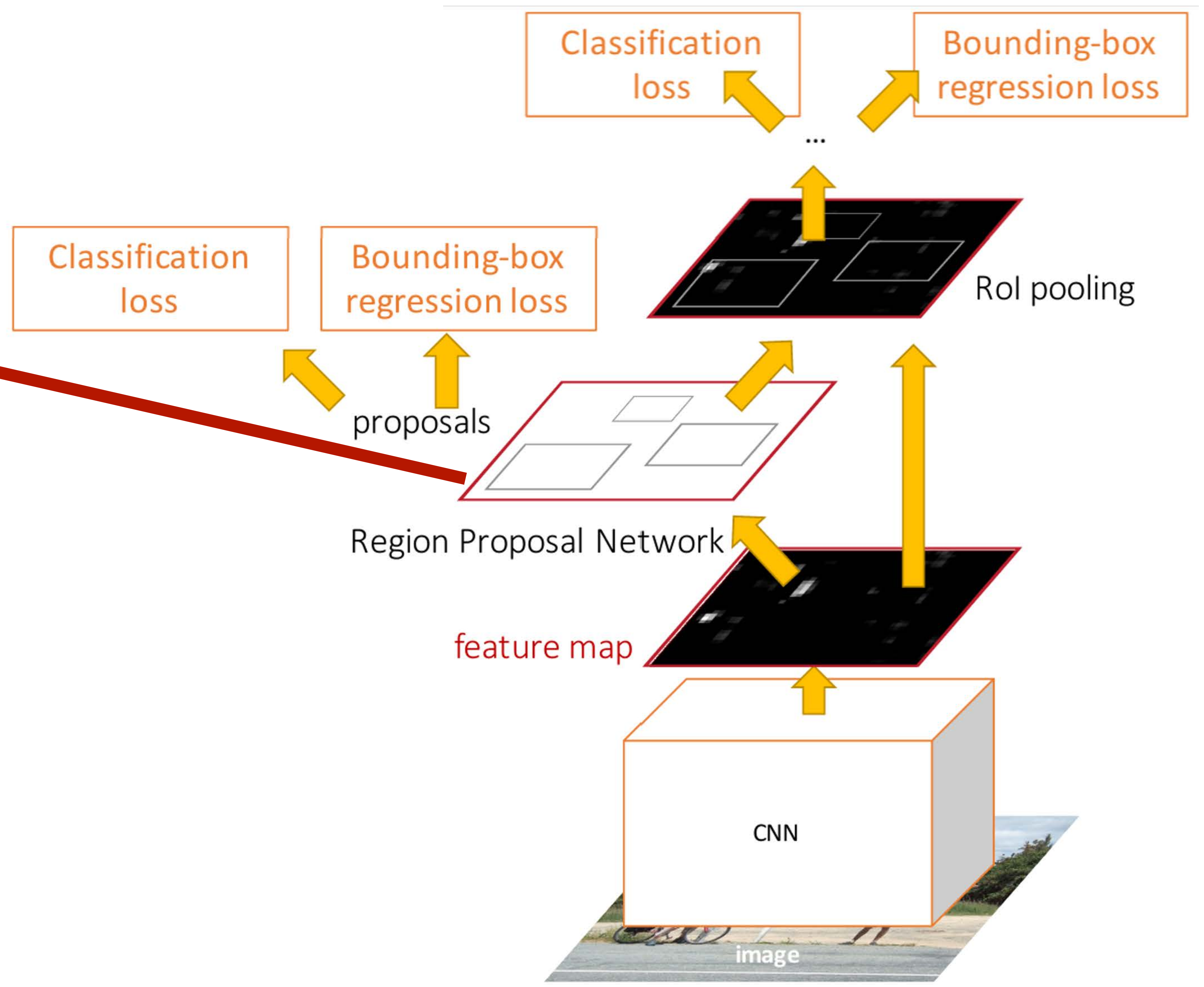
Recall: Region proposals computed by heuristic "Selective search" algorithm on CPU — let's learn them with a CNN

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: Learnable Region Proposals

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

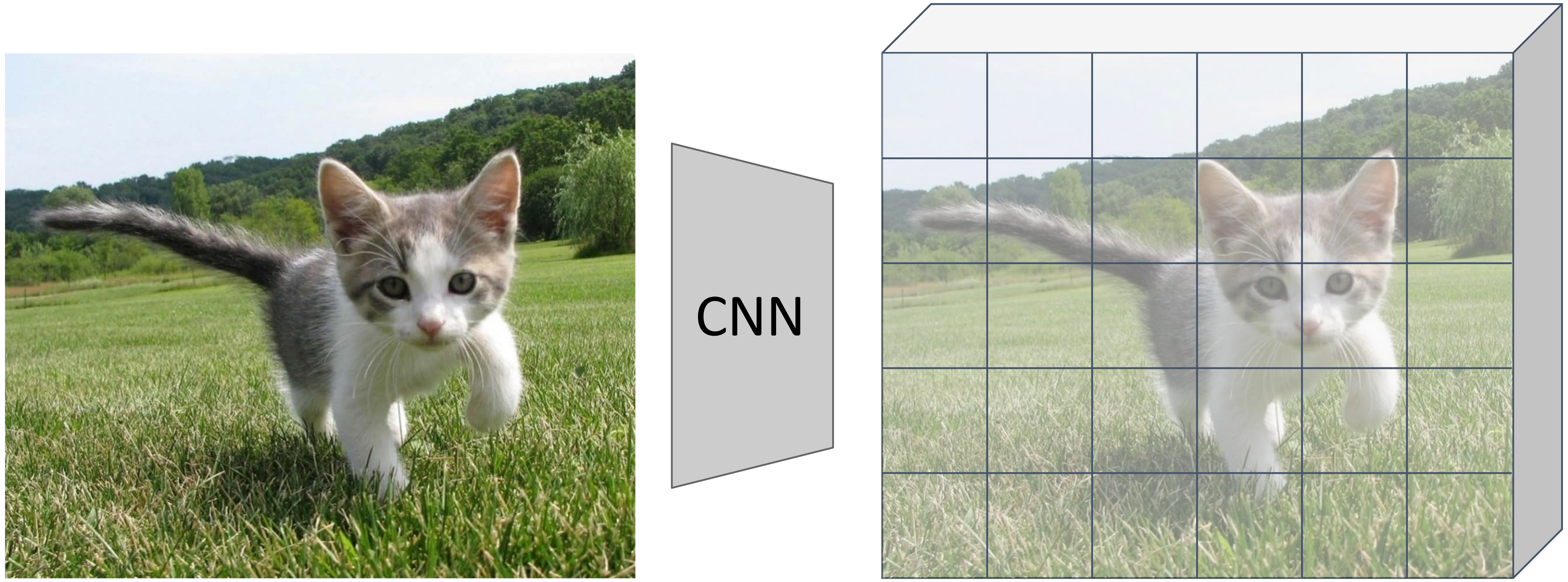
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



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

Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image



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

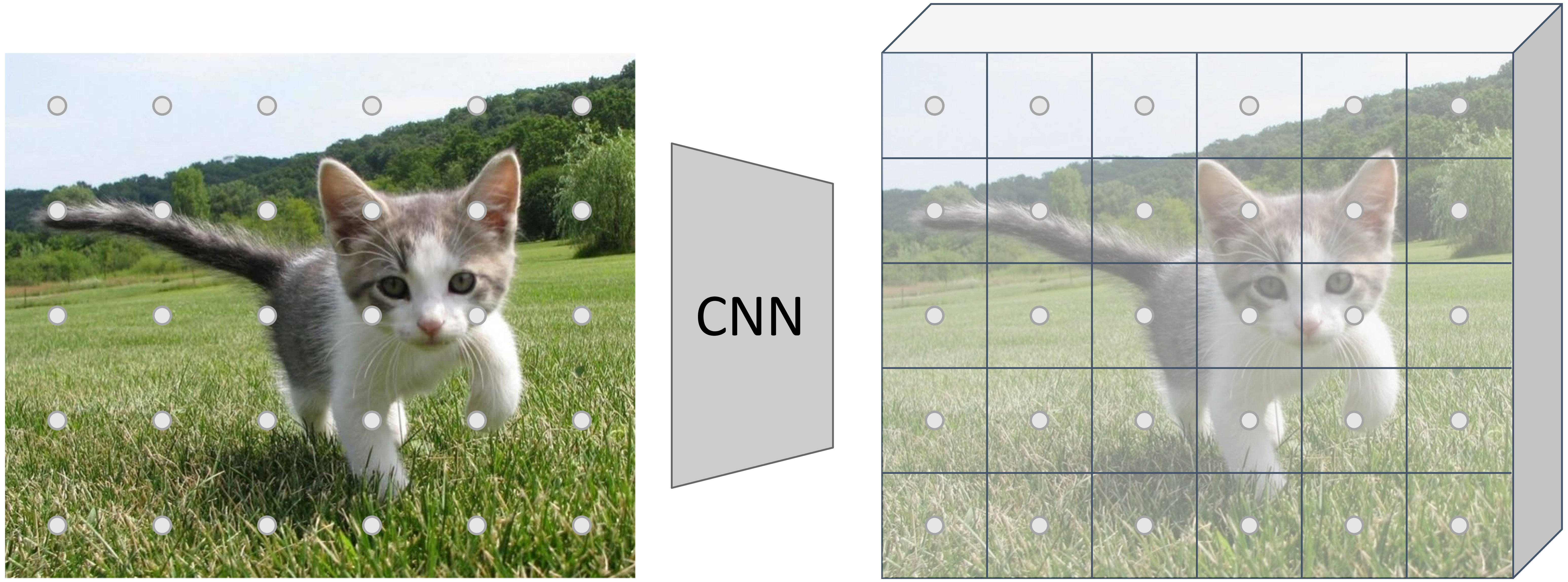
Image features
(e.g. 512 x 5 x 6)

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

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

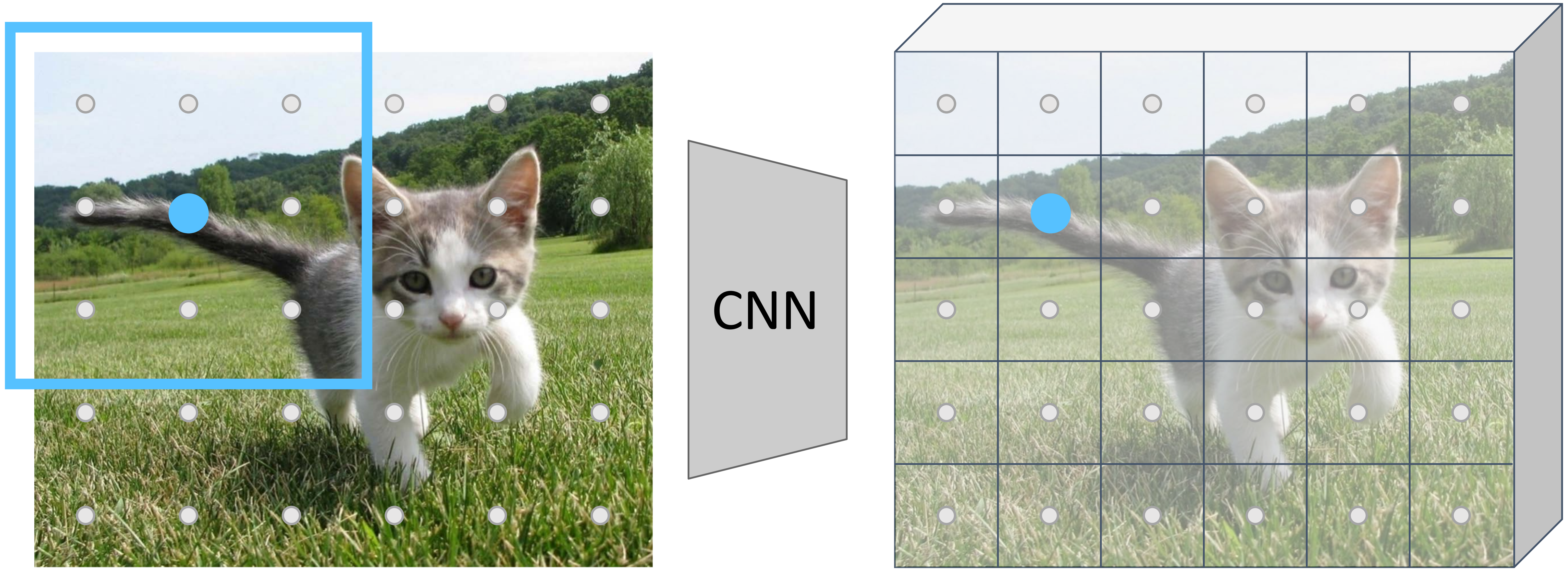
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

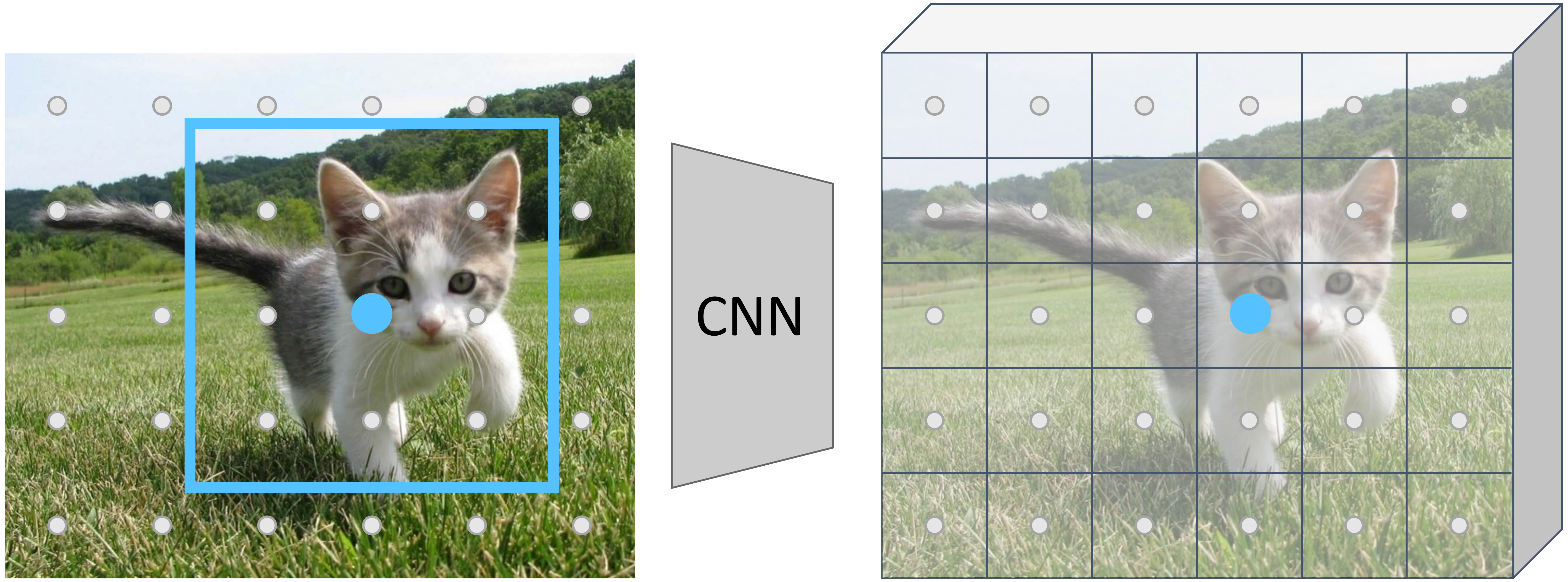
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

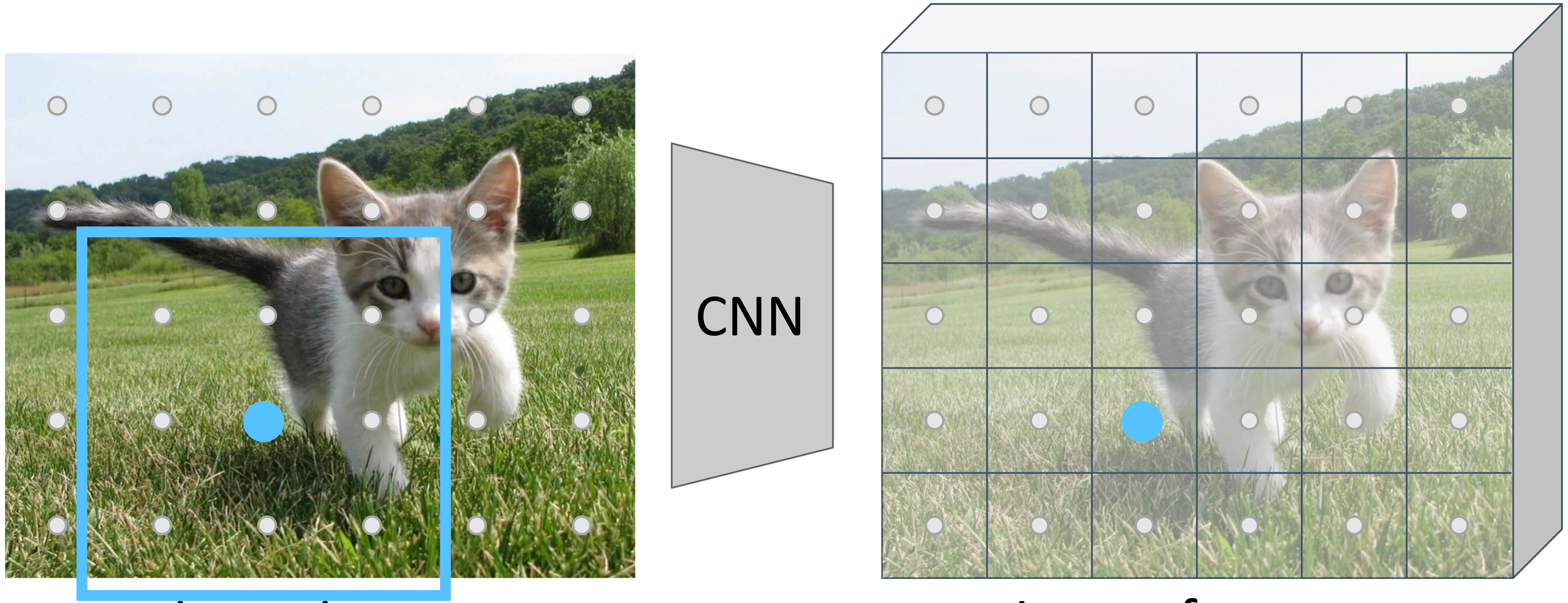
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

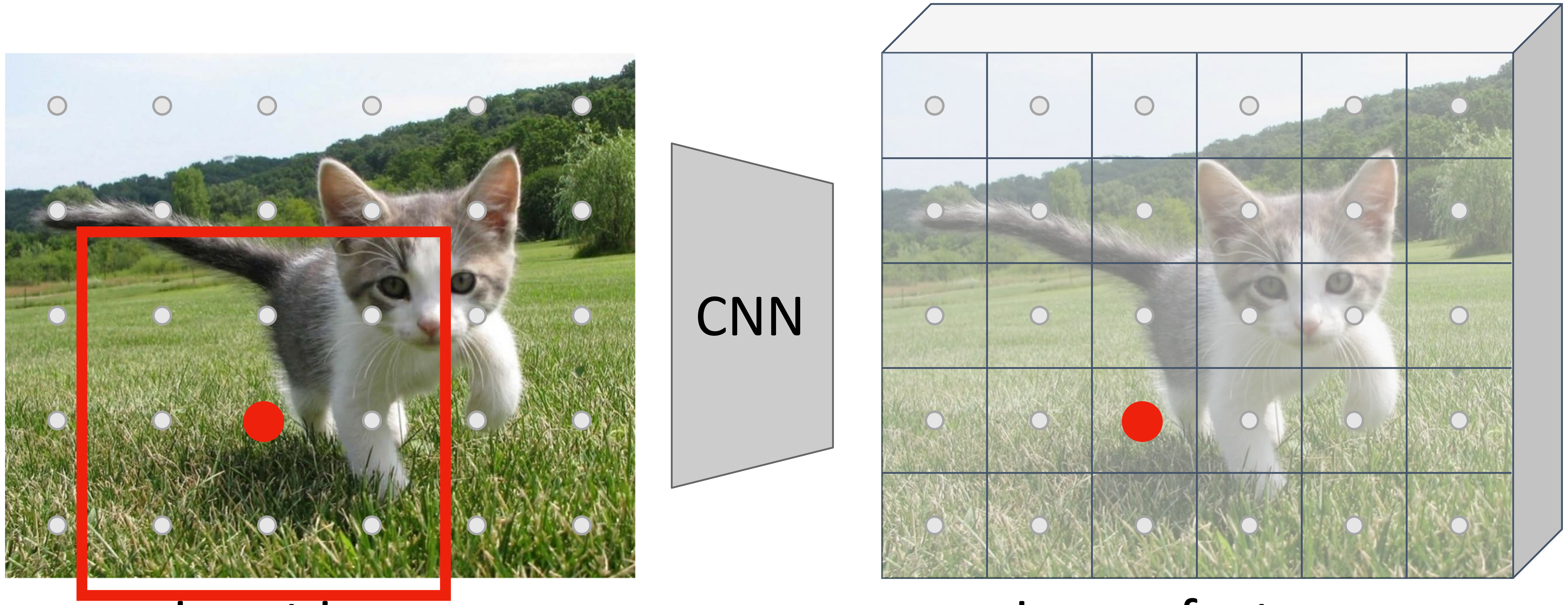
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

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

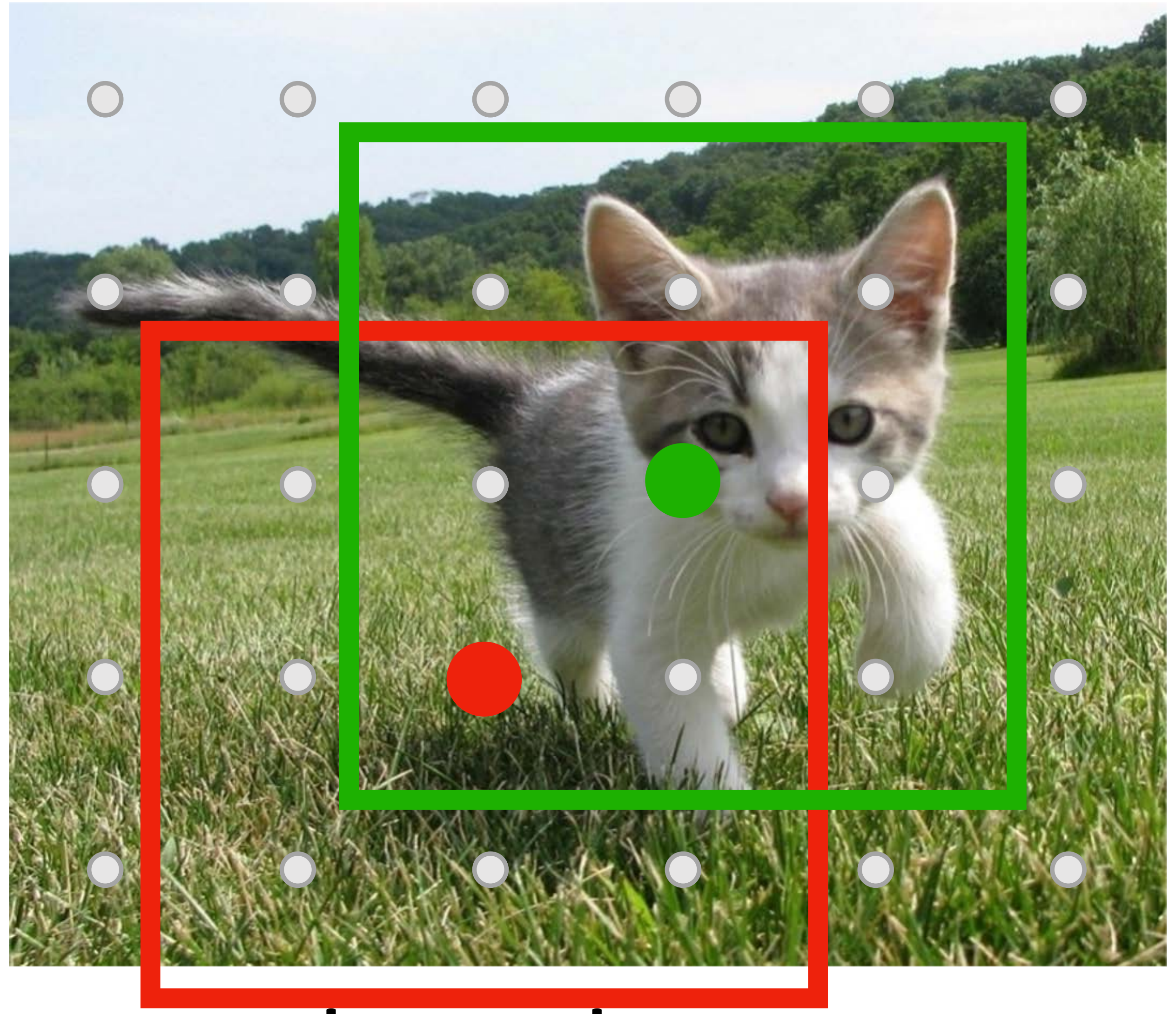
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

CNN

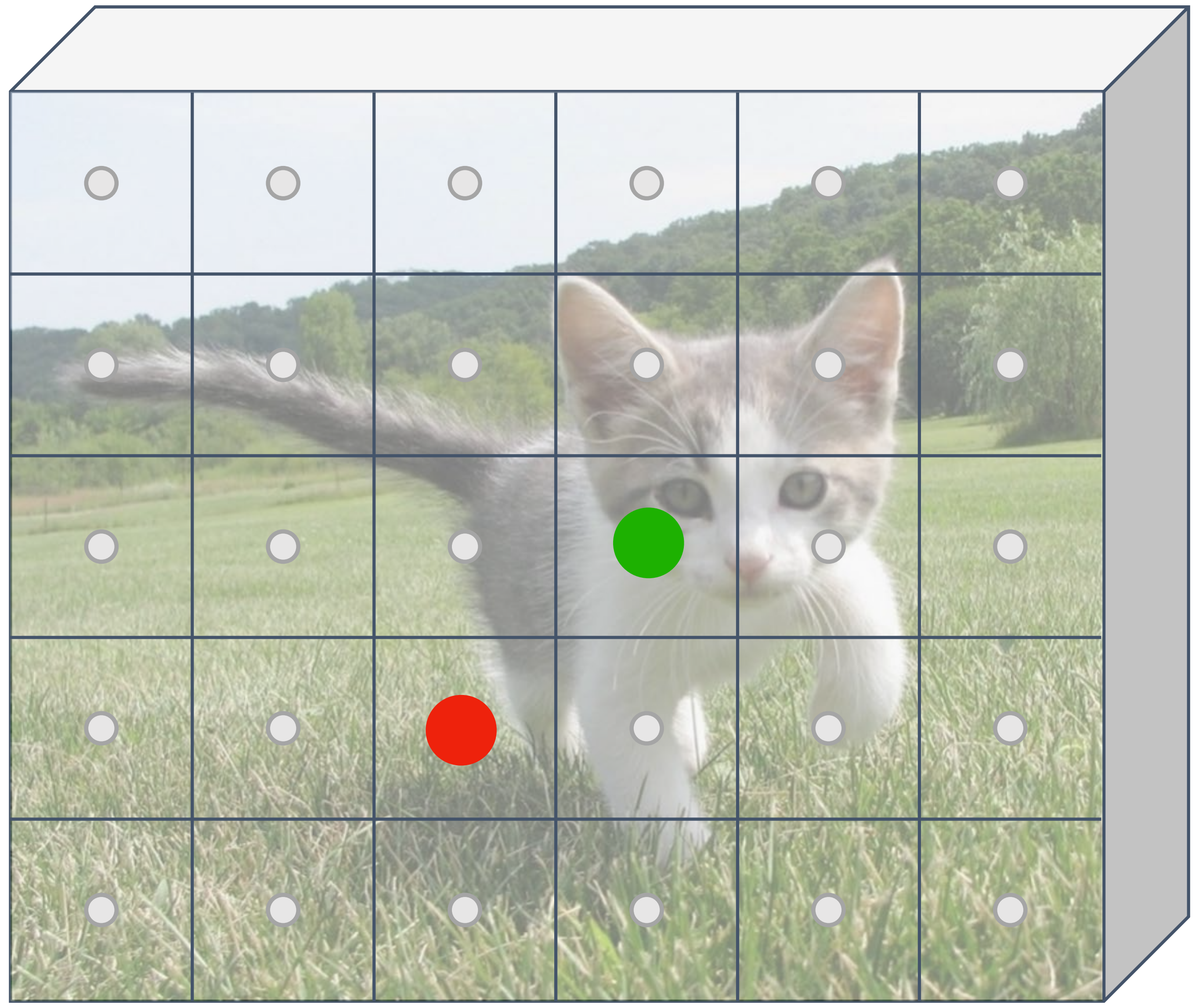


Image features
(e.g. 512 x 5 x 6)

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

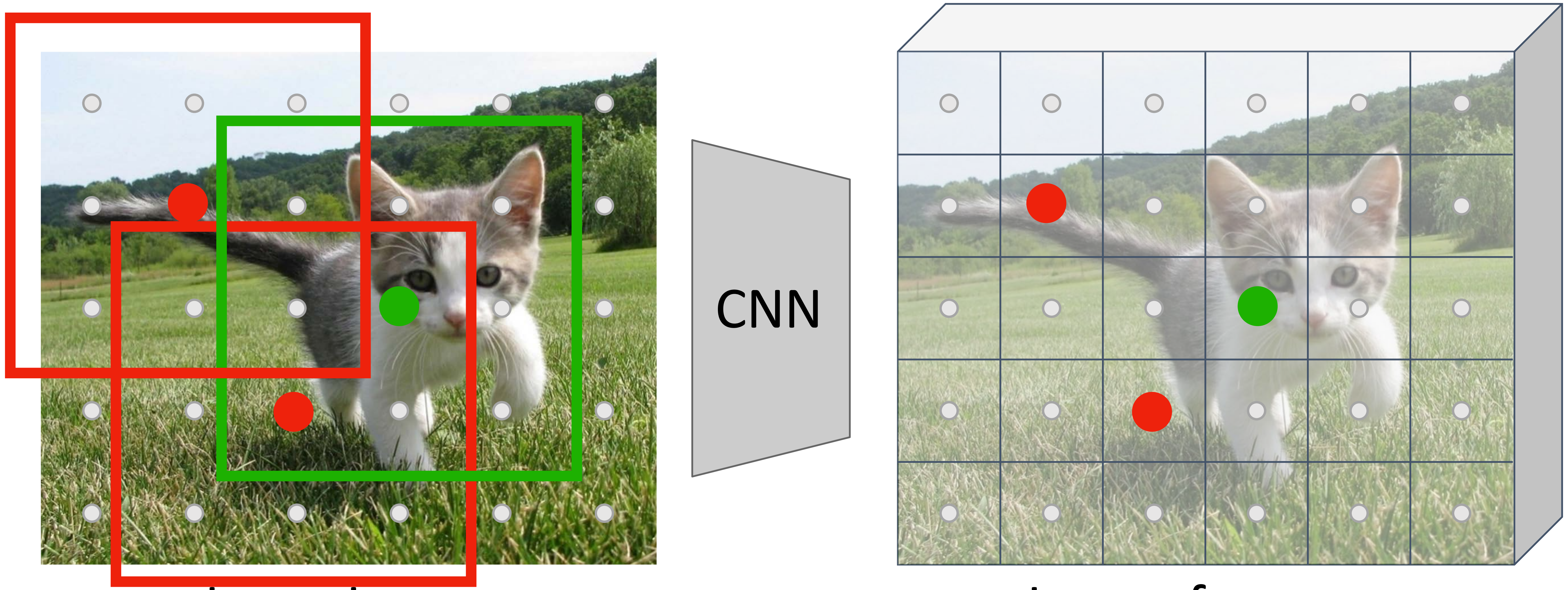
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

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

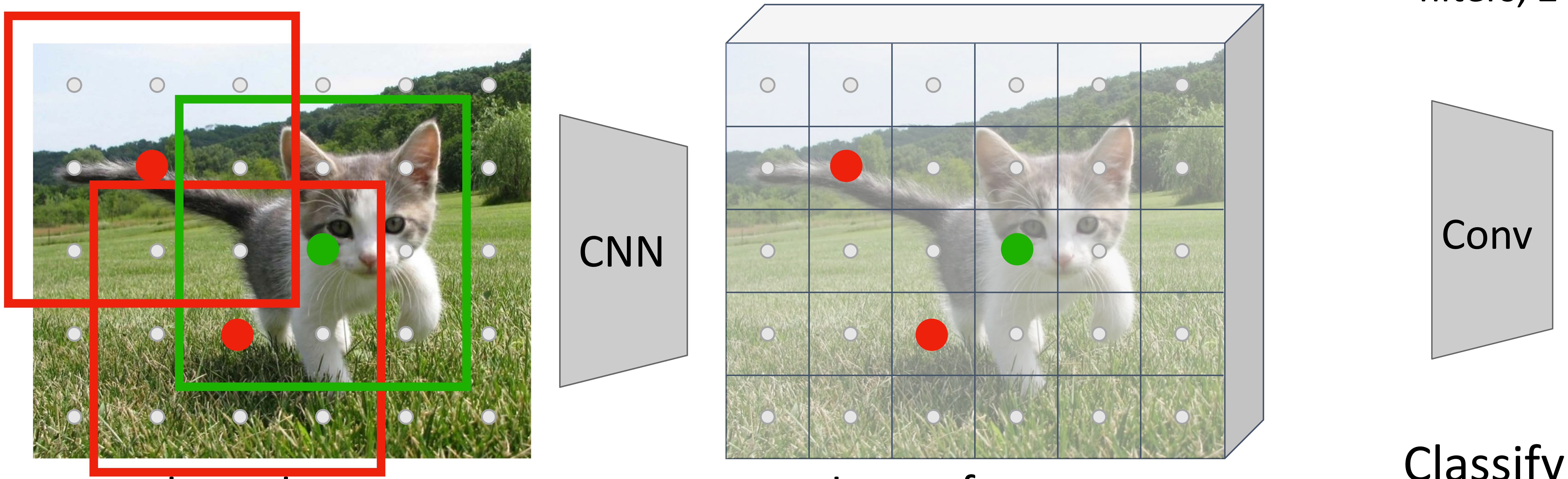
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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



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

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

Anchor is object?
2 x 5 x 6

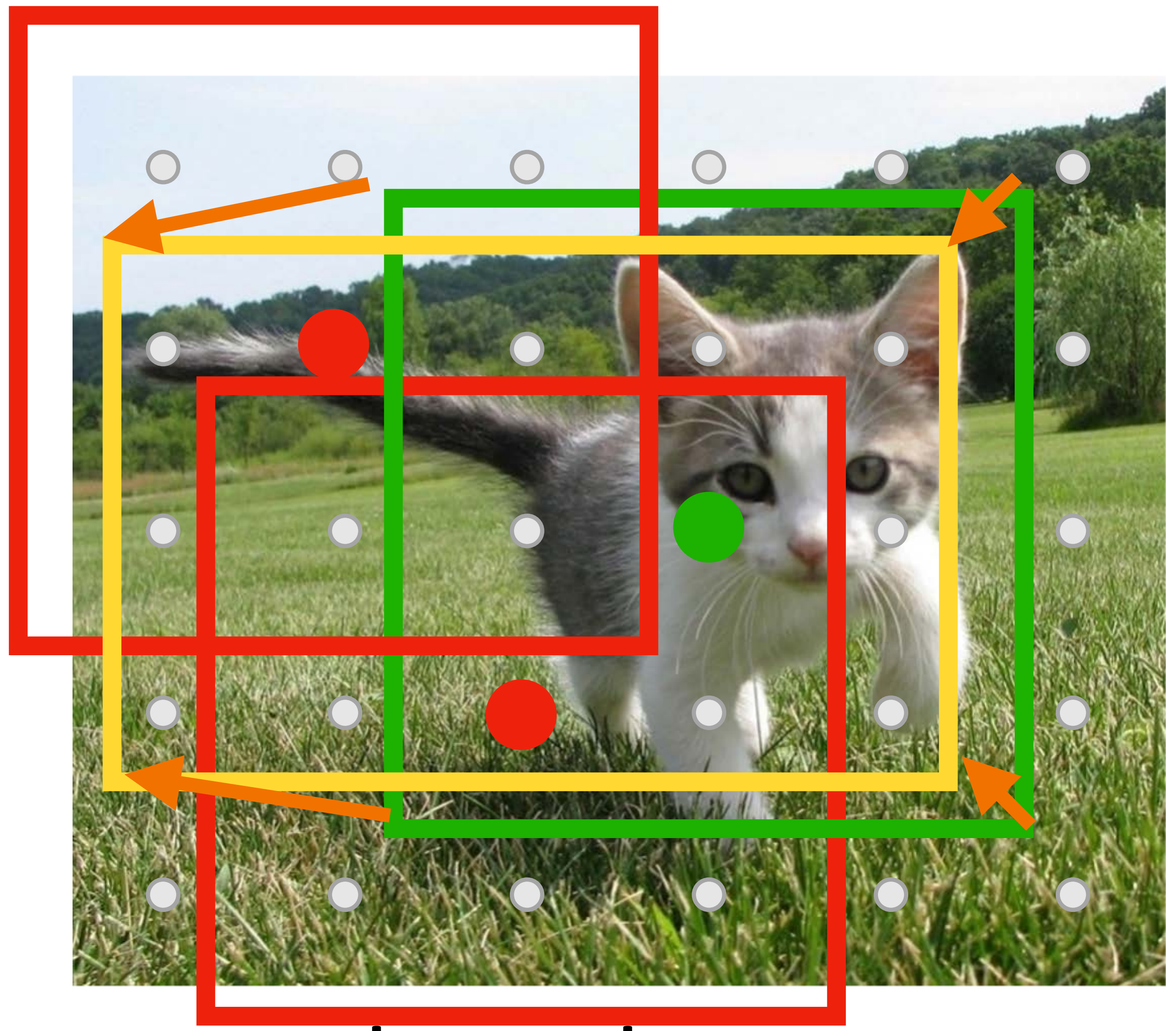
Classify each anchor as positive (object) or negative (no object)

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

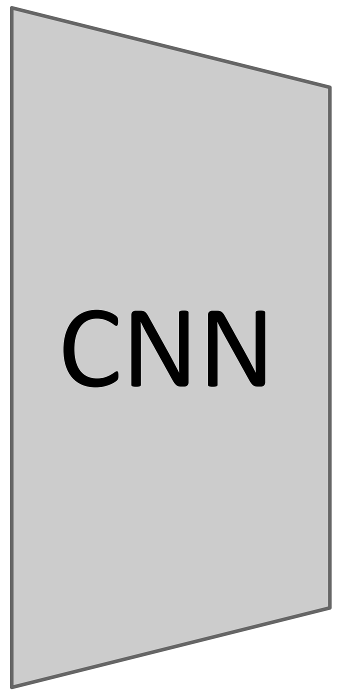
Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image



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



Each feature corresponds to a point in the input

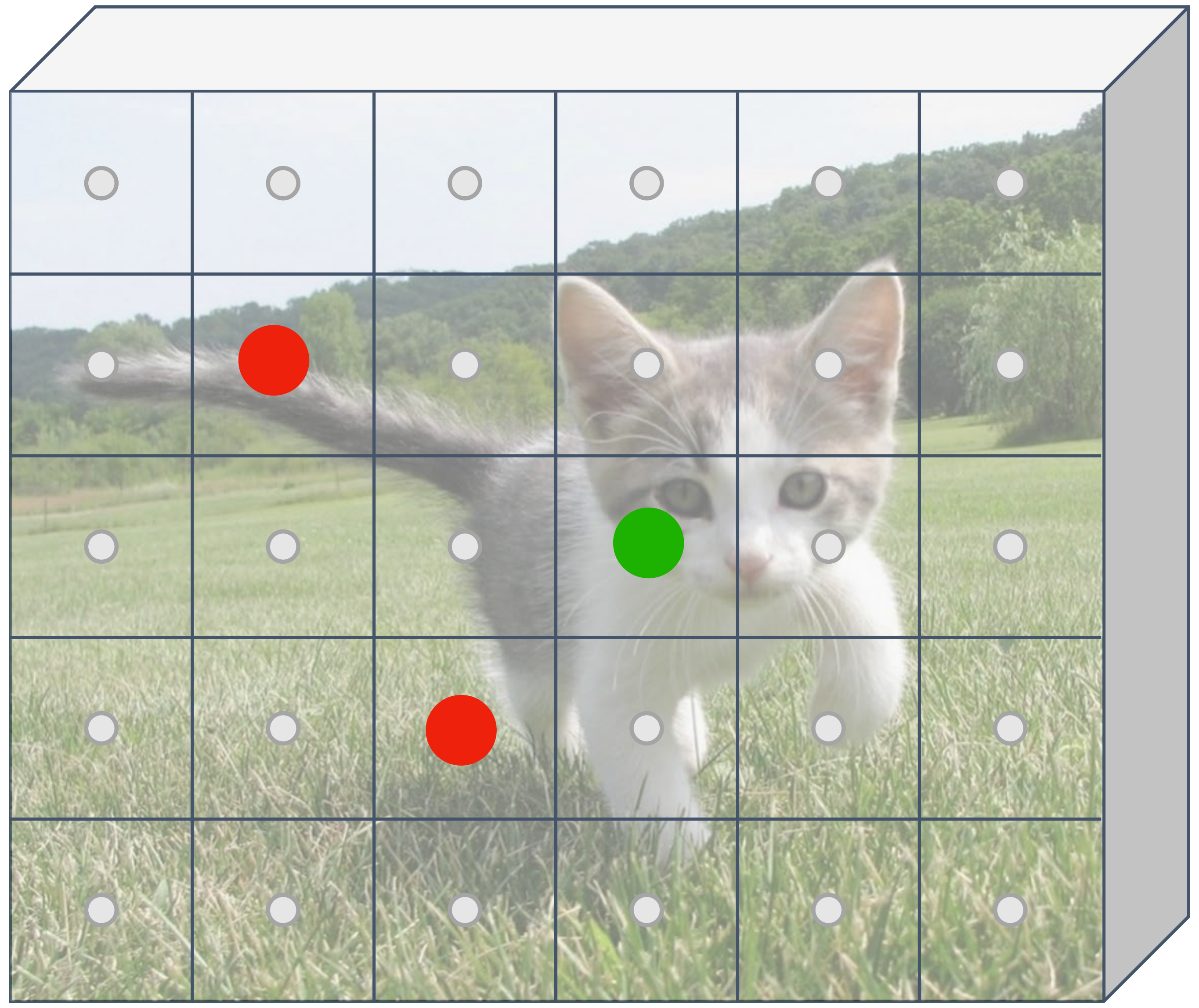
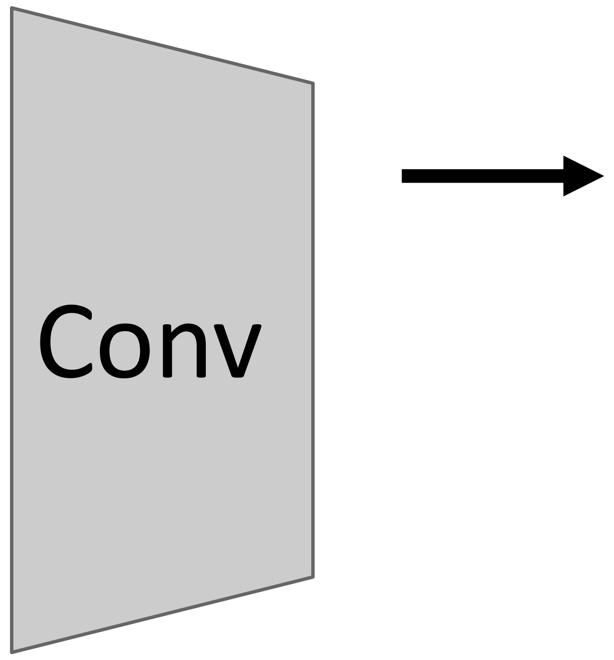


Image features
(e.g. 512 x 5 x 6)



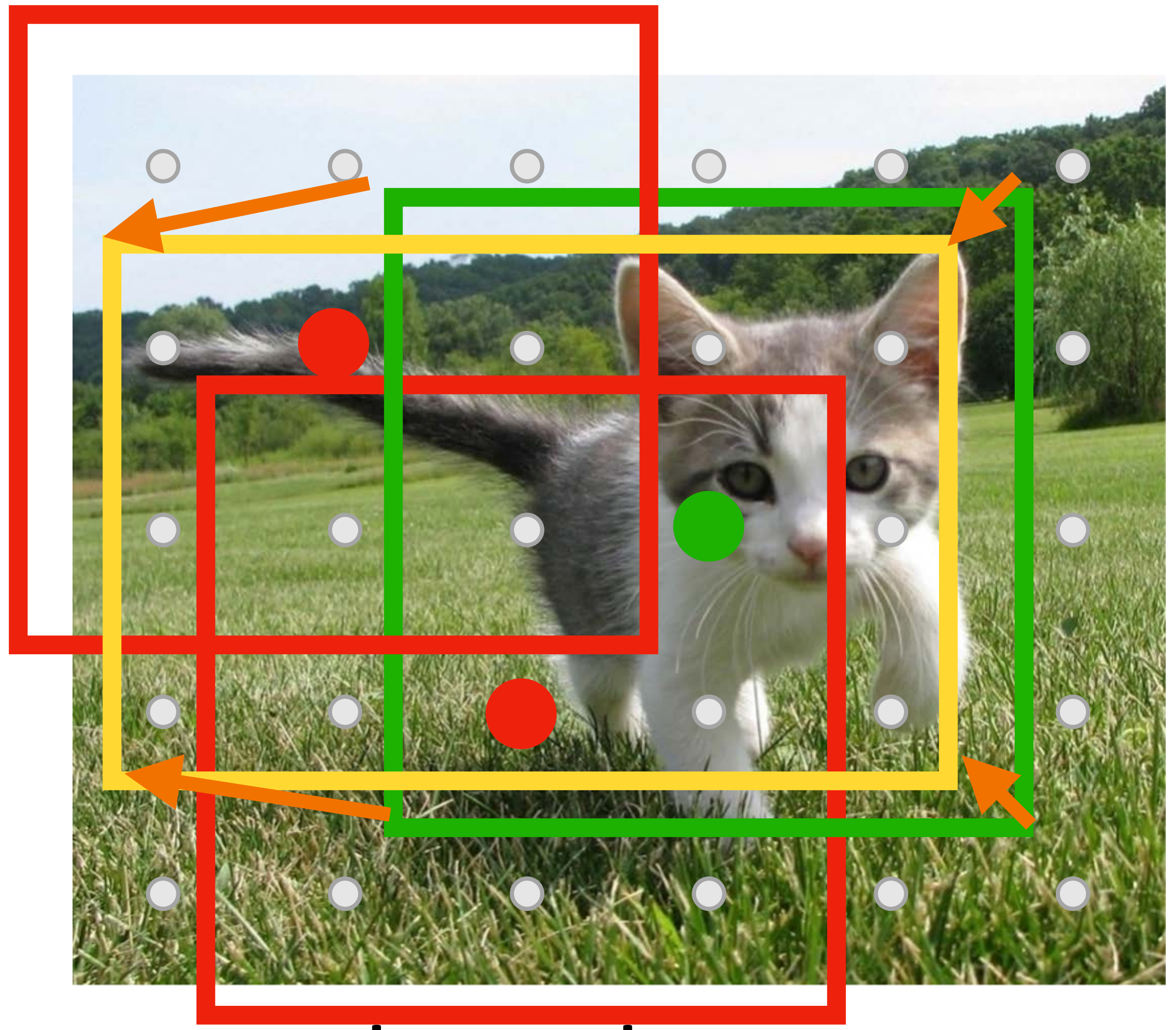
Anchor is object?
2 x 5 x 6

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

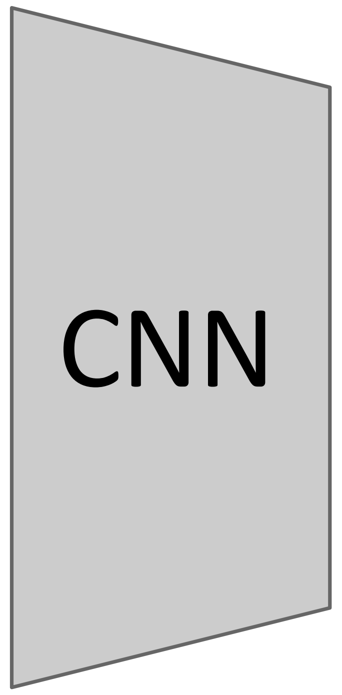
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



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



Each feature corresponds to a point in the input

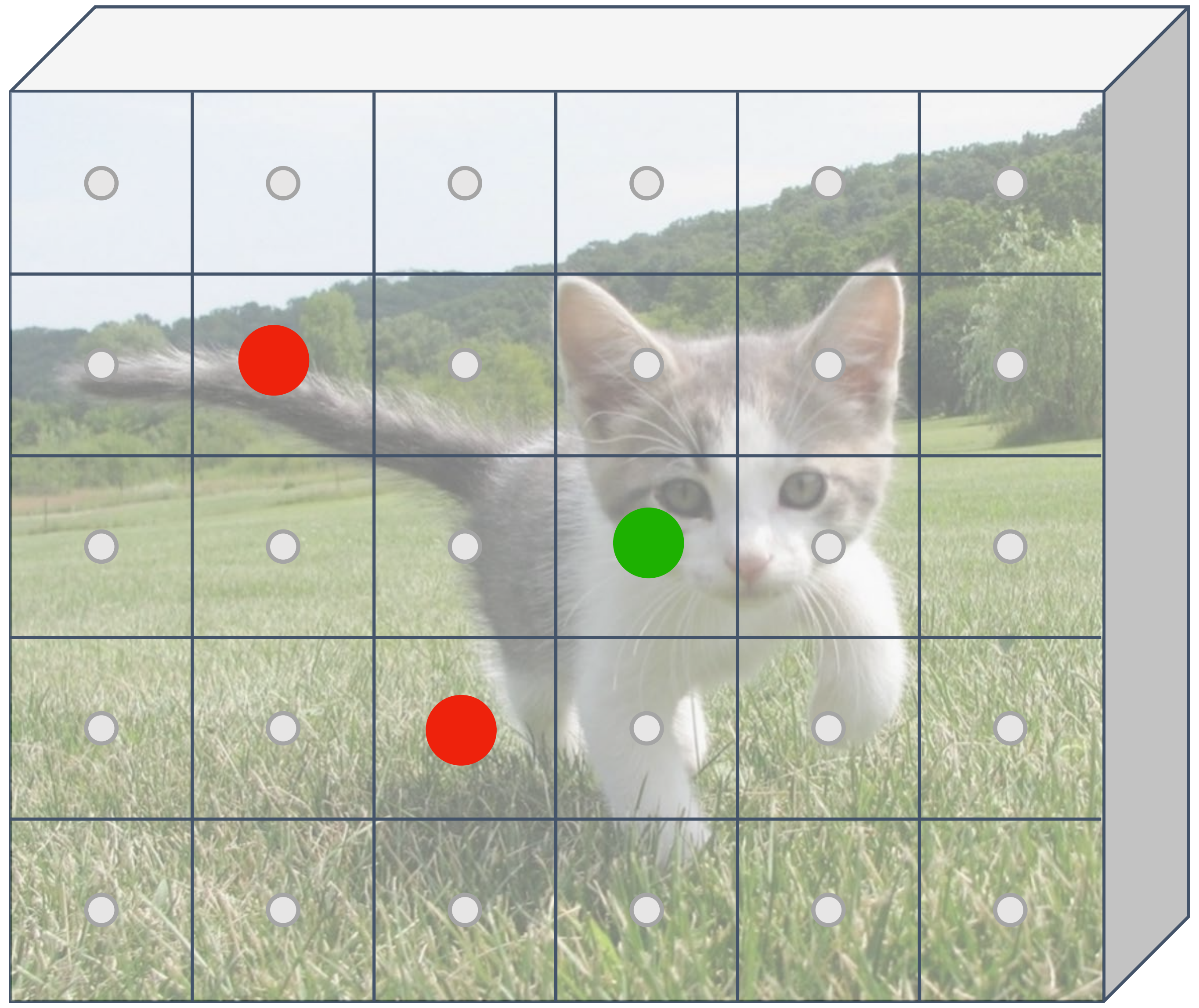
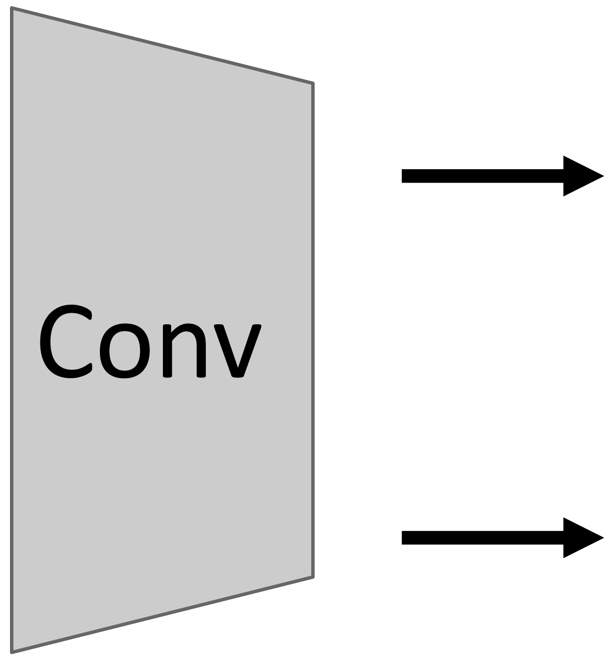


Image features
(e.g. 512 x 5 x 6)

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



Anchor is object?
2 x 5 x 6

Anchor transforms
4 x 5 x 6

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

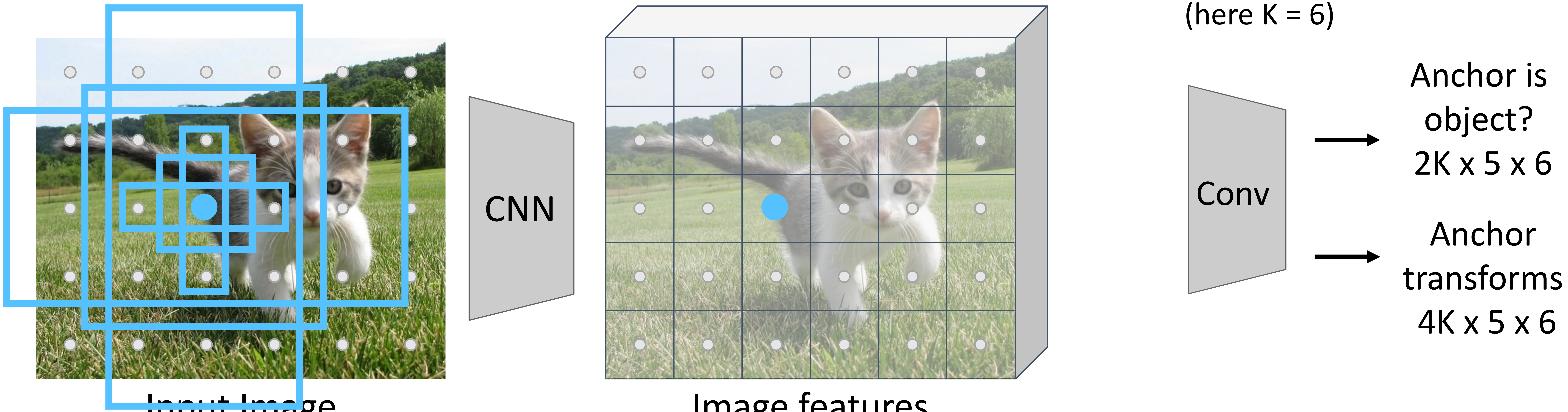
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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

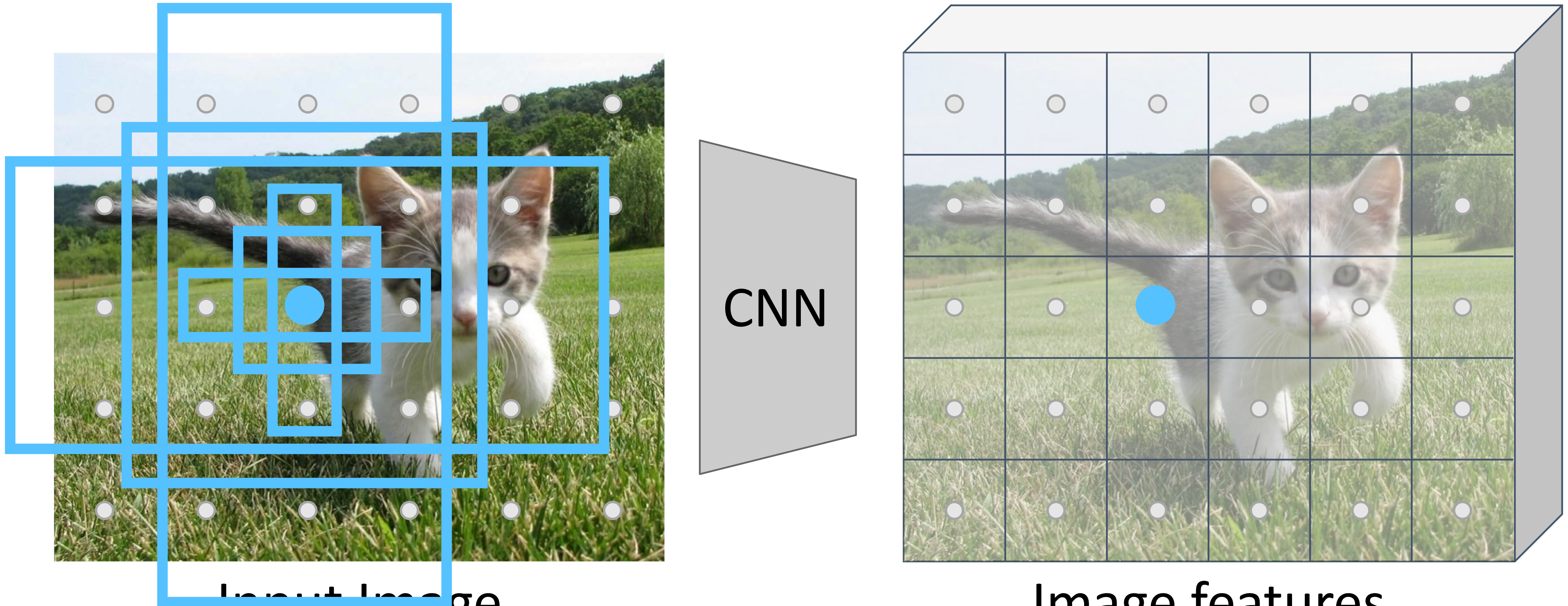
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

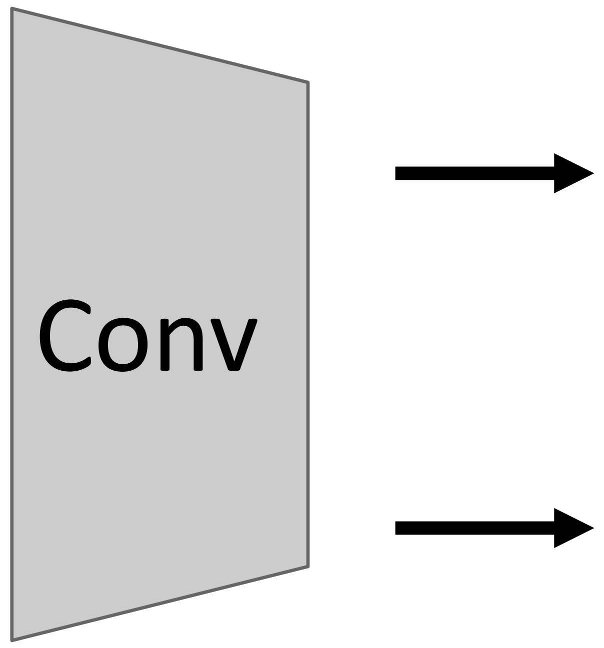
Each feature corresponds to a point in the input

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



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

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



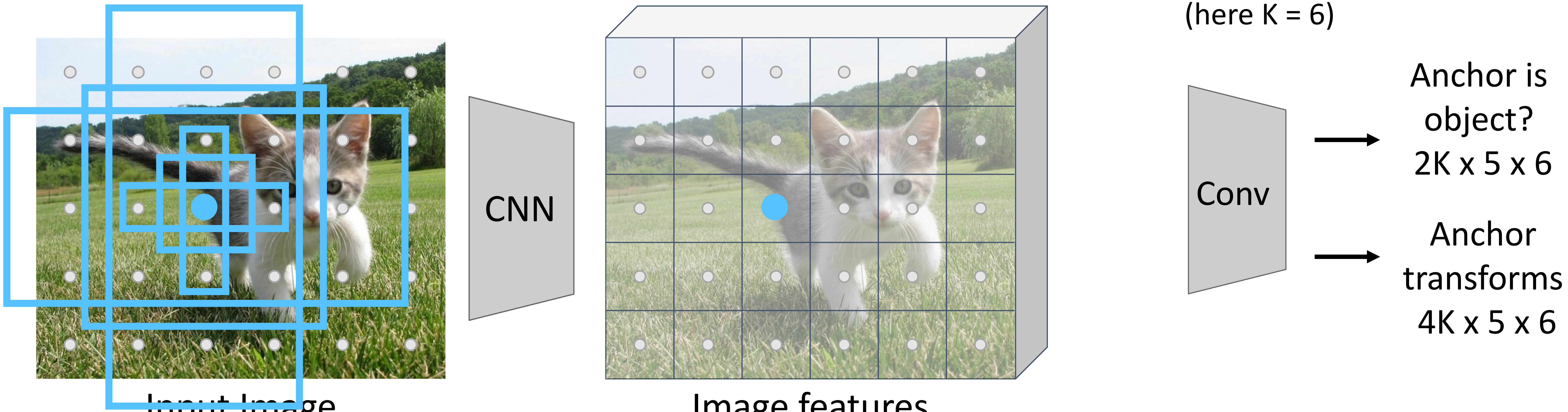
Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

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

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

Region Proposal Network (RPN)



Run backbone CNN to get features aligned to input image

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

Each feature corresponds to a point in the input

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

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

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

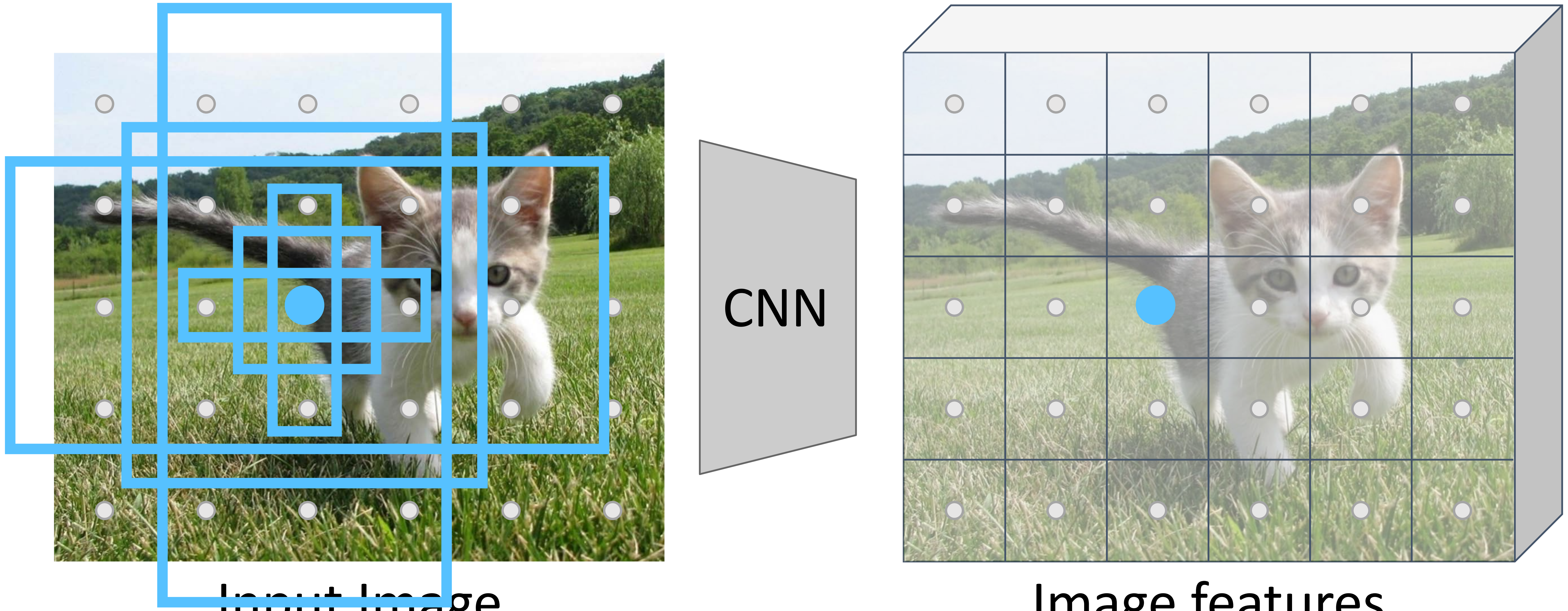
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

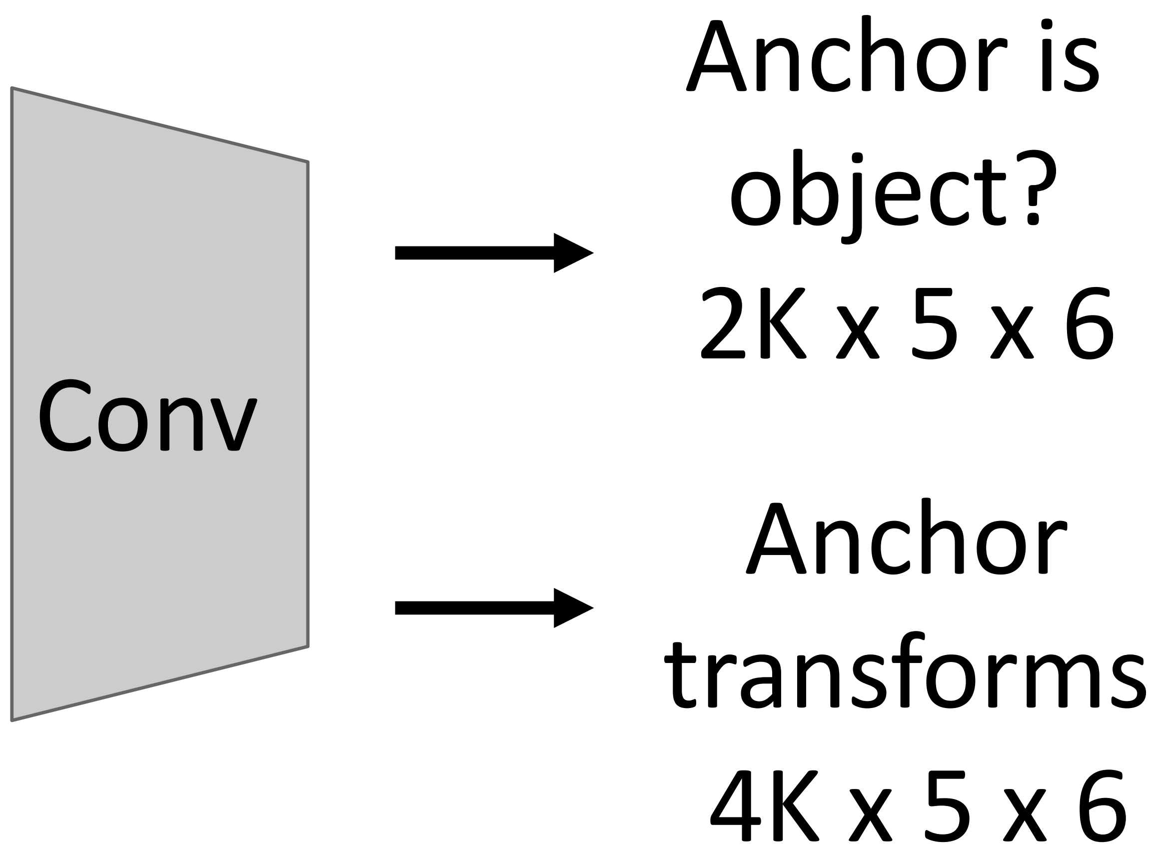
Each feature corresponds to a point in the input

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



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

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



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

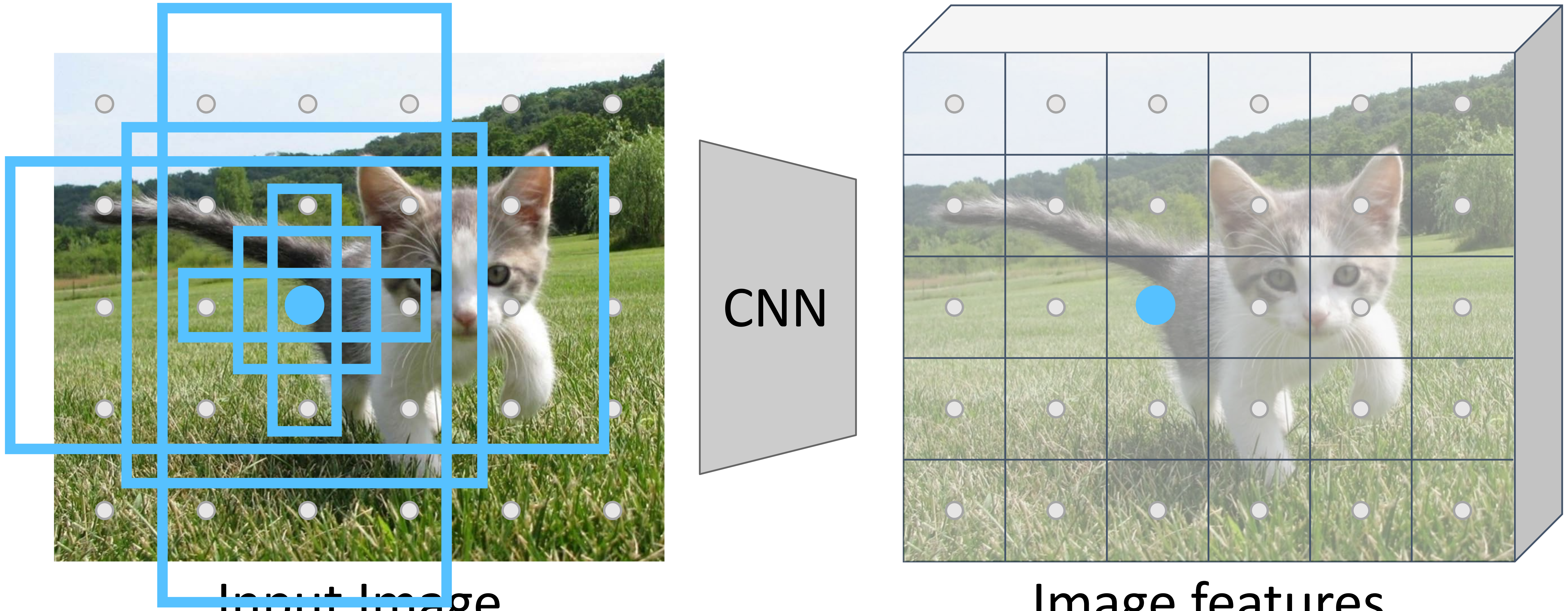
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

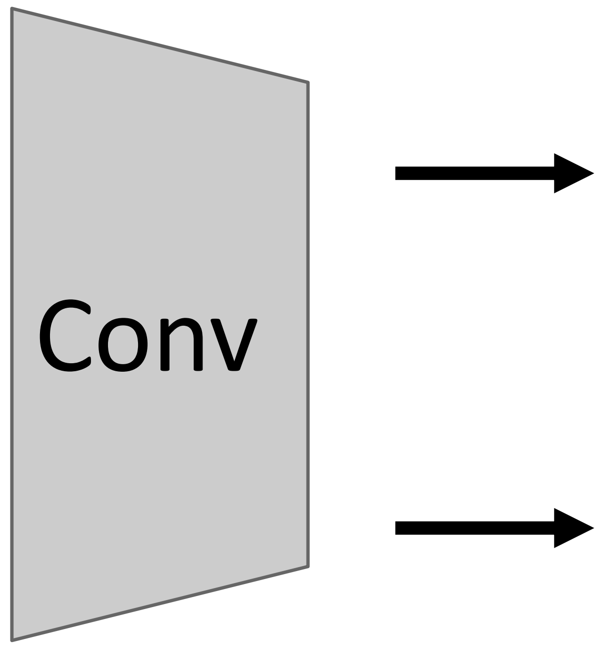
Each feature corresponds to a point in the input

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



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

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



Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

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

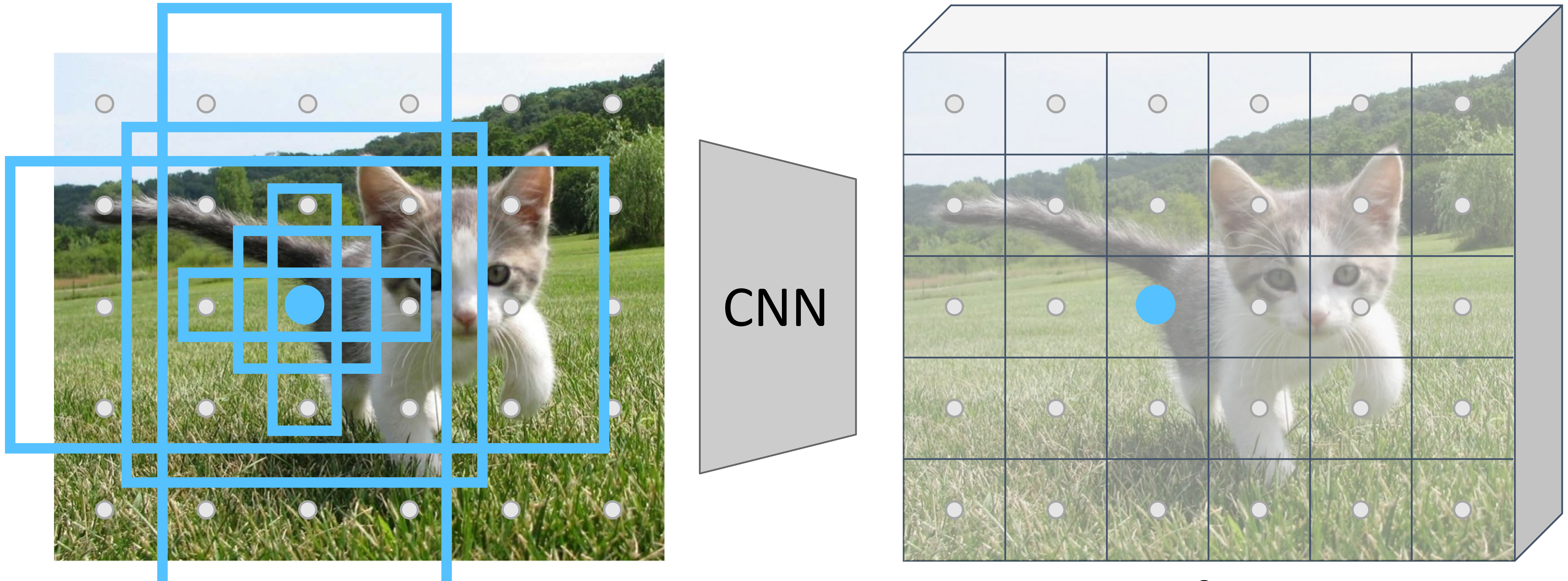
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

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

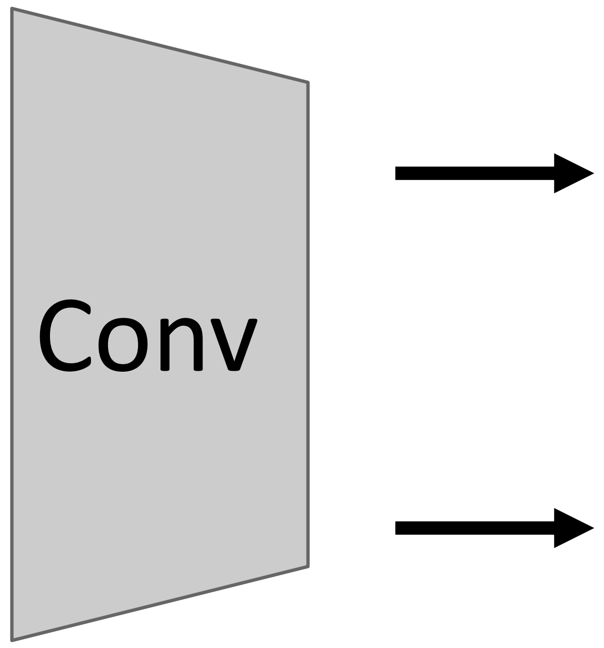
Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



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

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



Anchor is object?
2K x 5 x 6

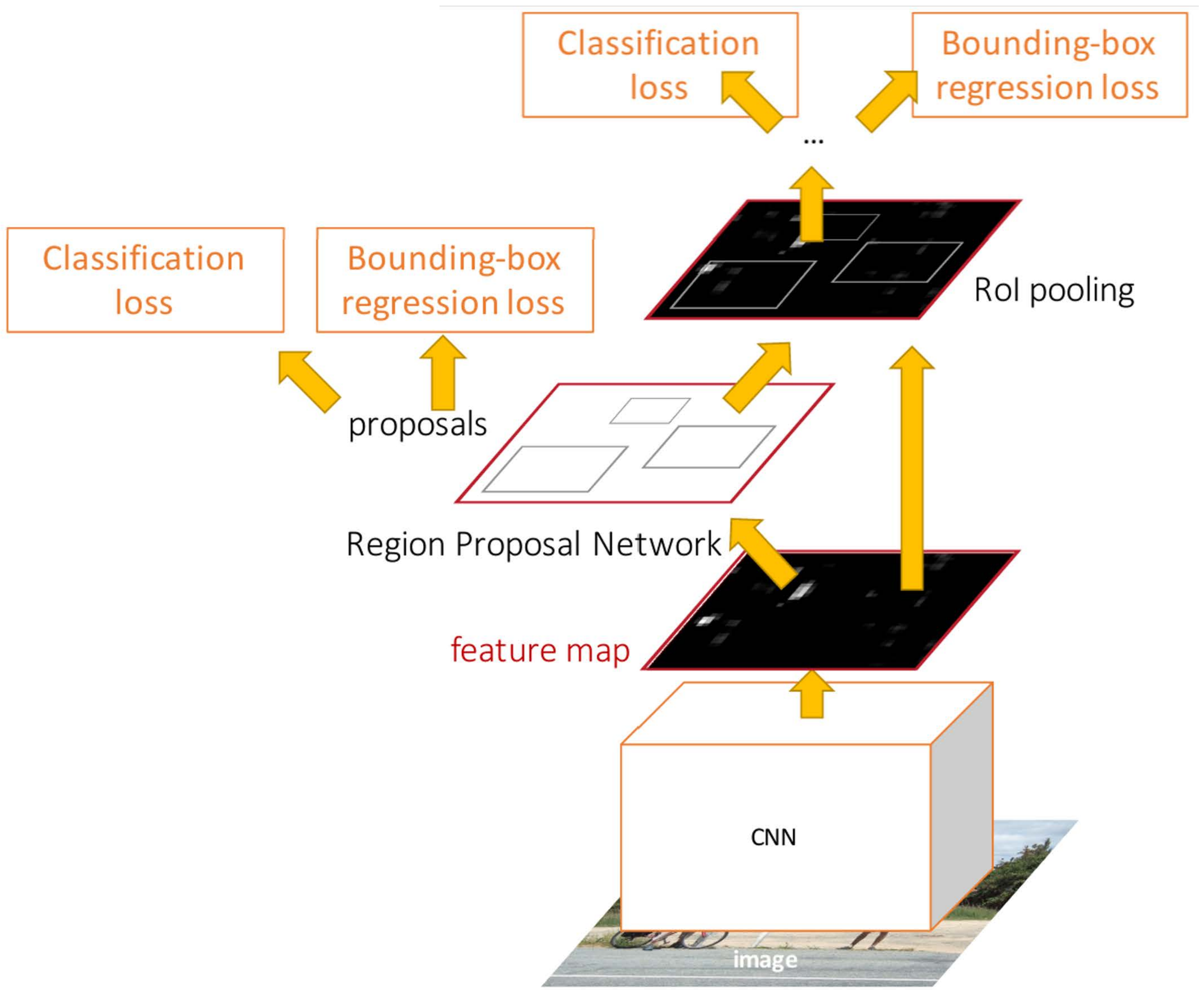
Anchor transforms
4K x 5 x 6

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

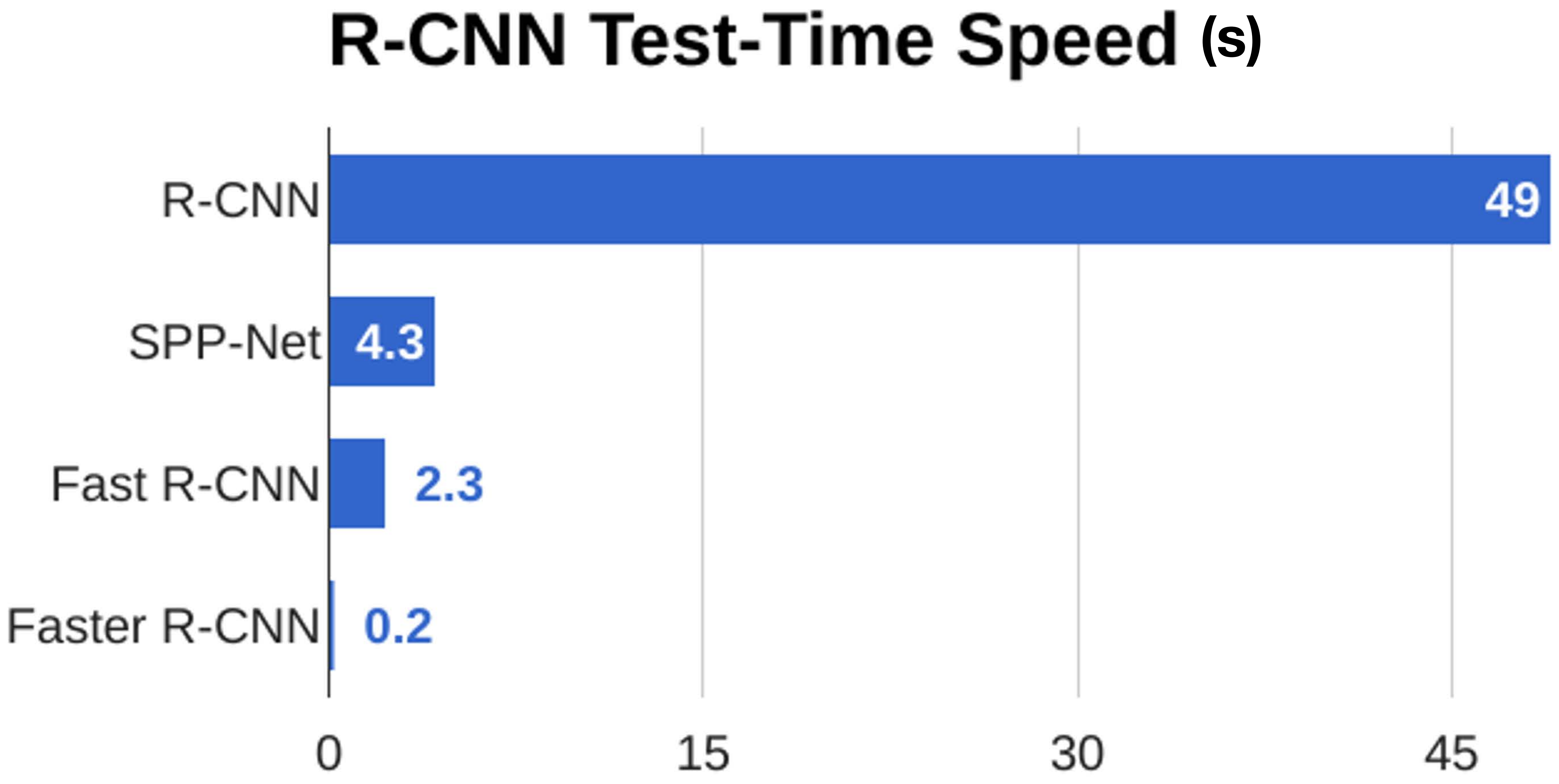
Jointly train four losses:

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



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

Faster R-CNN: Learnable Region Proposals



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

Classification

"Chocolate Pretzels"

No spatial extent

Semantic Segmentation

Chocolate Pretzels, Shelf

No objects, just pixels

Object Detection

Instance Segmentation

Flipz, Hershey's, Keese's

Multiple objects

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

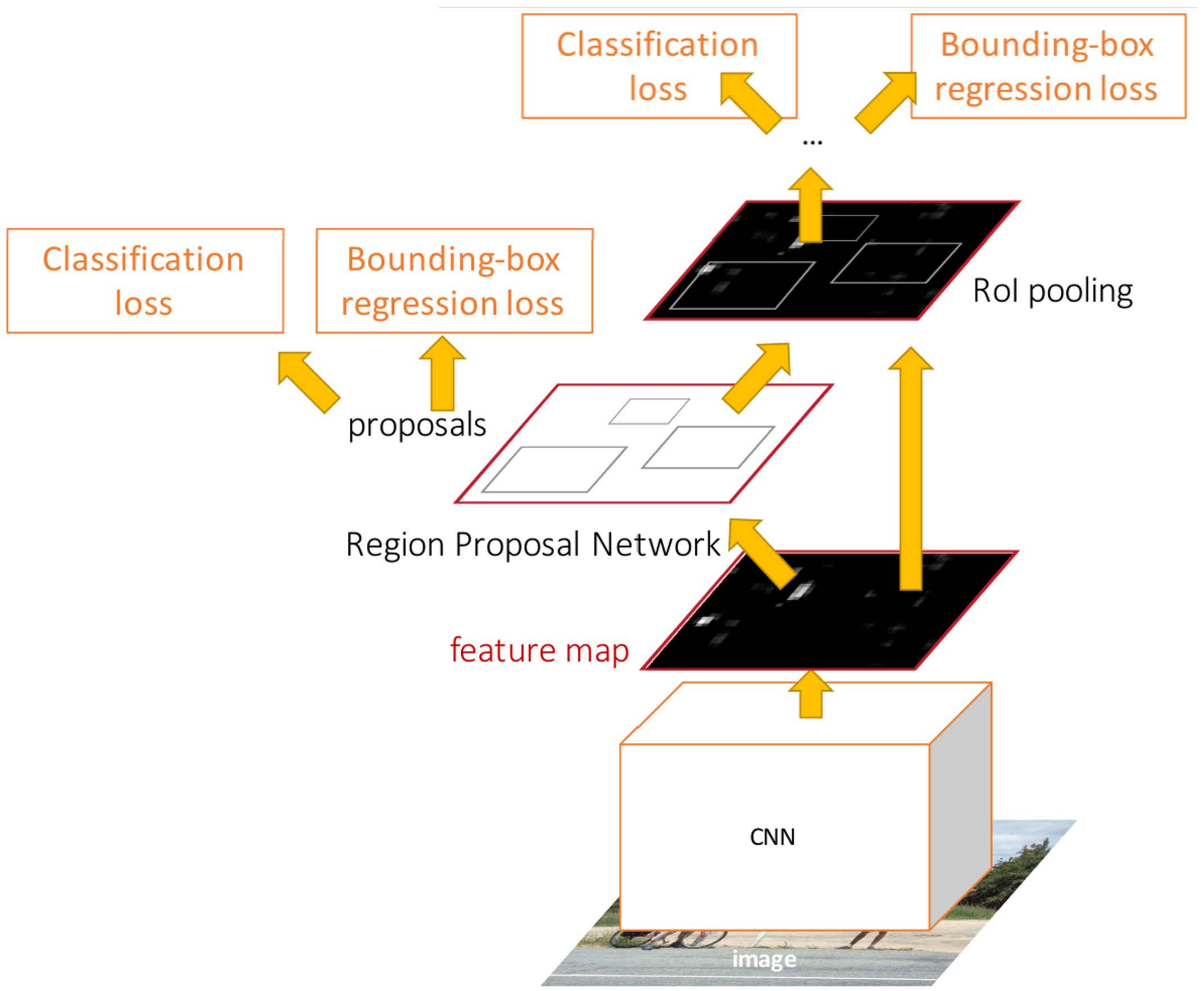
Instance Segmentation
Detect all objects in the image and identify the pixels that belong to each object (Only things!)

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

Extend Faster R-CNN into 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 transforn from proposal box to object box

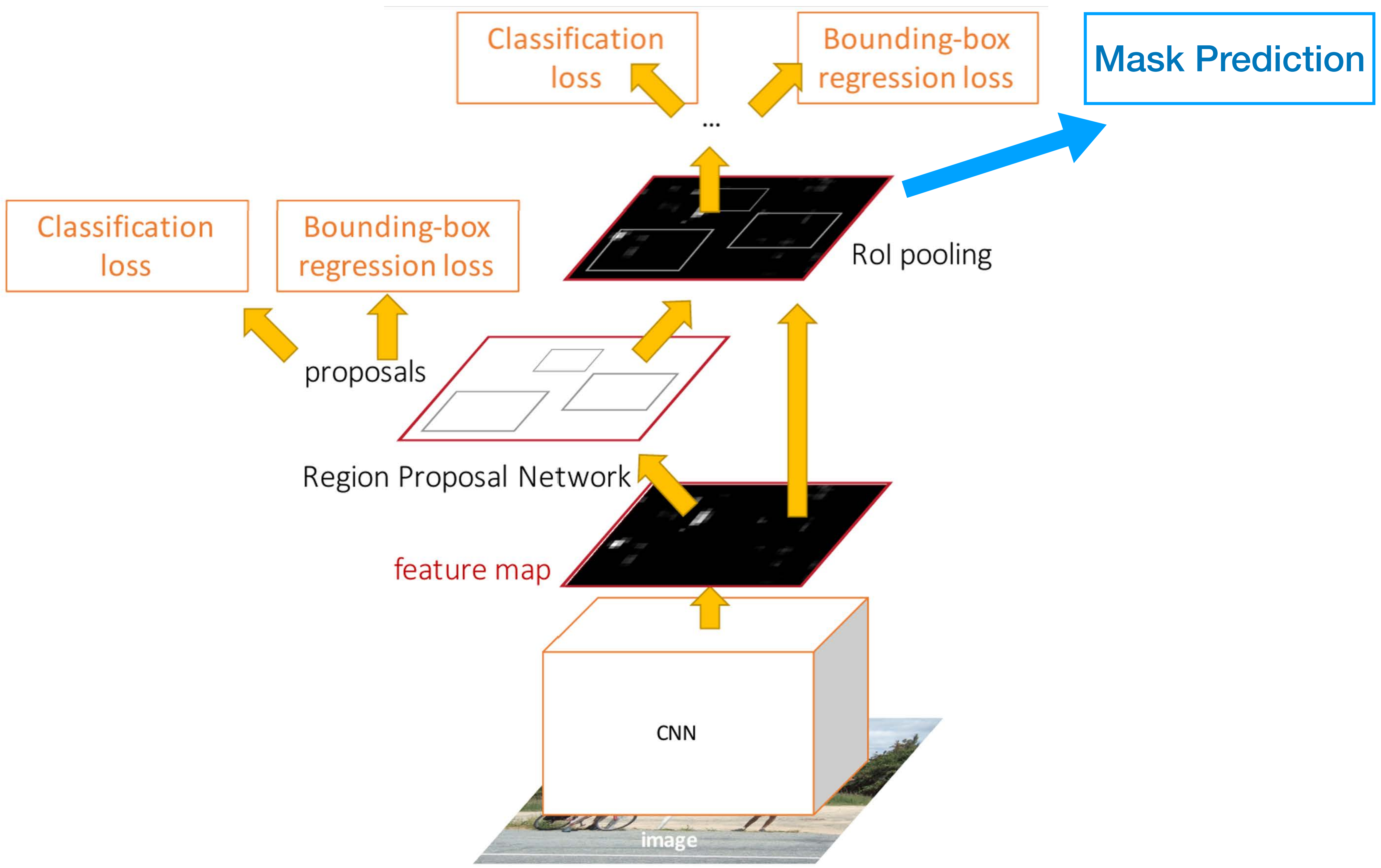


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick.

Extend Faster R-CNN into Mask R-CNN

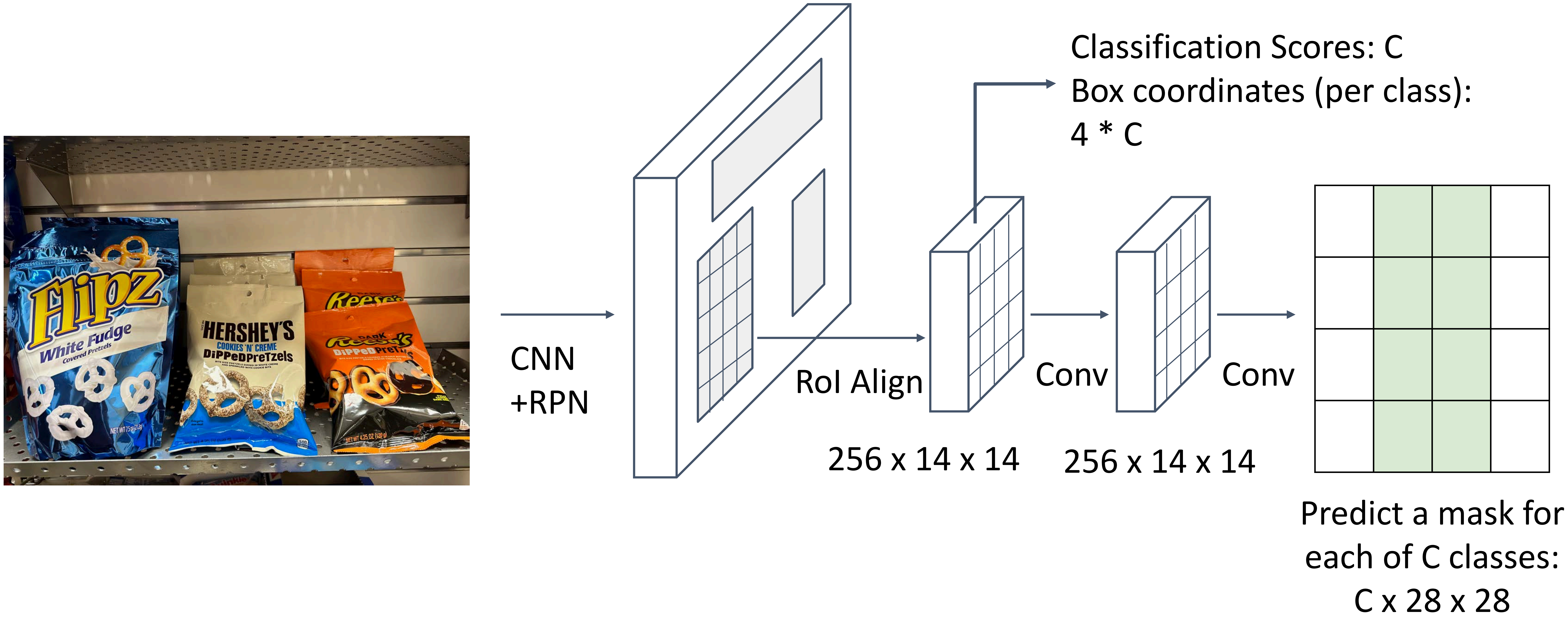
Mask R-CNN

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



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

Mask R-CNN


Flipz
White Fudge
HERSHEY'S
CNN
+RPN
RoI Align
256 x 14 x 14
Conv
256 x 14 x 14
Conv
Classification Scores: C
Box coordinates (per class):
4 * C
Predict a mask for
each of C classes:
C x 28 x 28

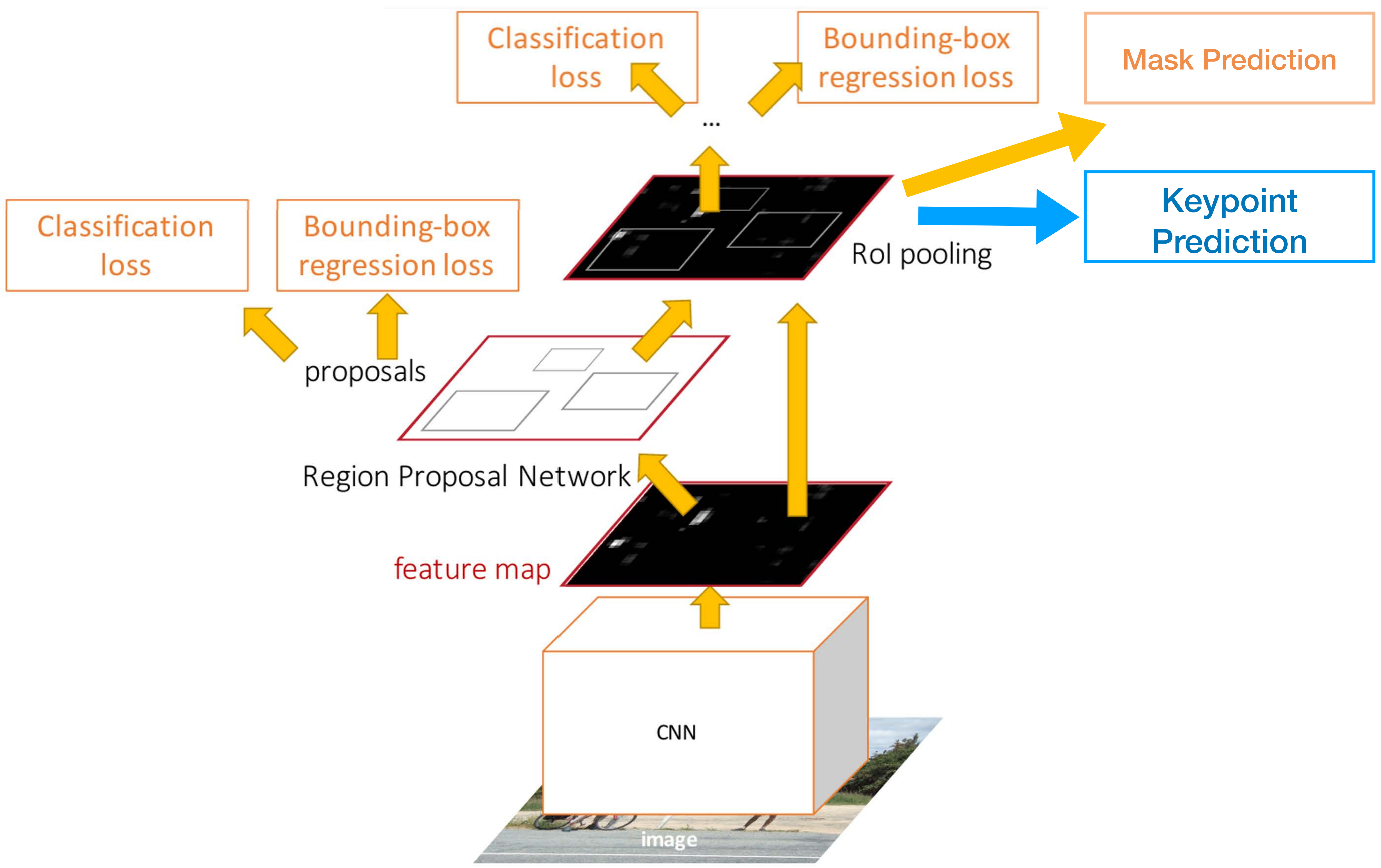
Mask R-CNN: Very Good Results!

He et al., “Mask R-CNN”, ICCV 2017

Mask R-CNN for Human Pose Estimation

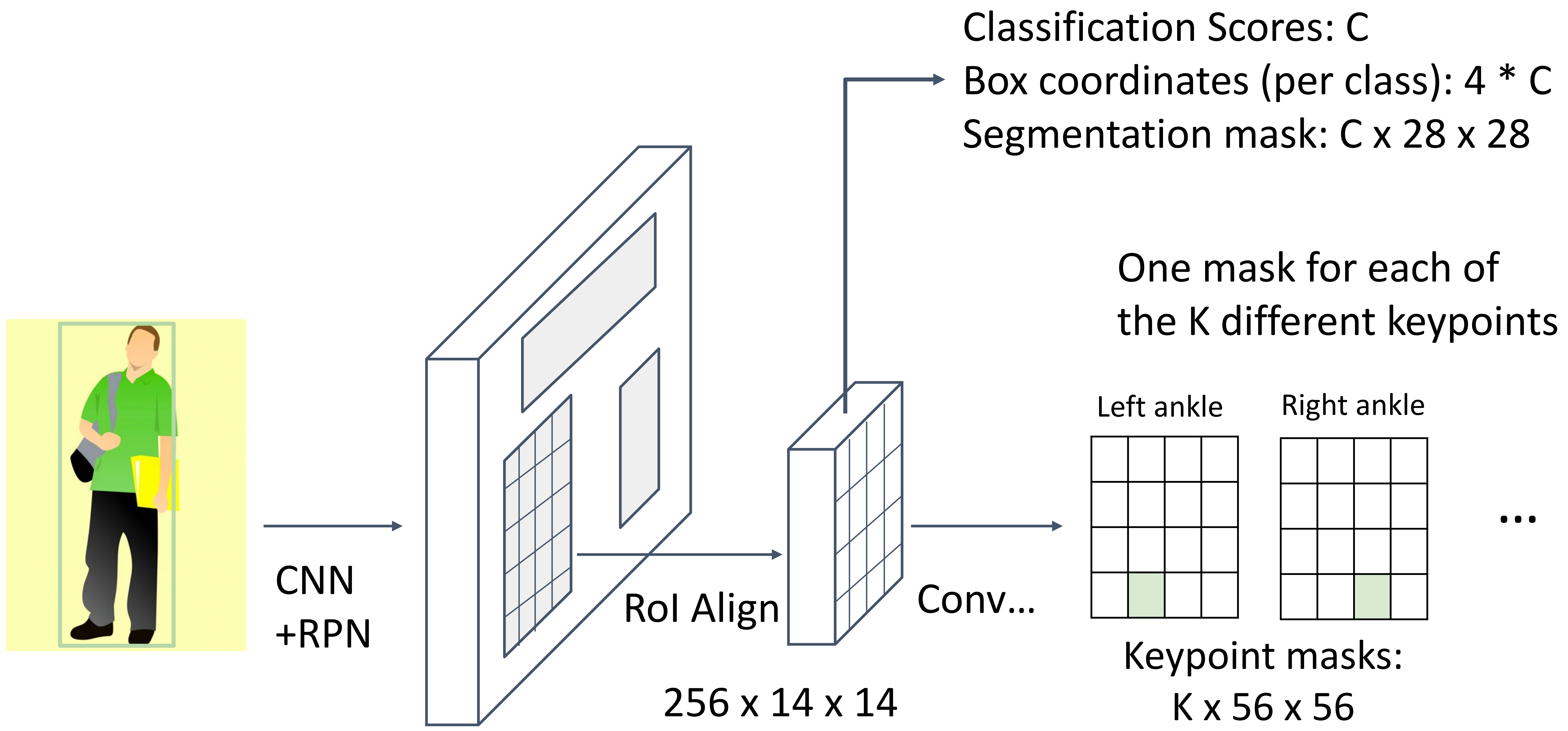
Mask R-CNN

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



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

Mask R-CNN for Human Pose Estimation



Ground-truth has one “pixel” turned on per keypoint. Train with softmax loss

He et al., “Mask R-CNN”, ICCV 2017

Mask R-CNN for Human Pose Estimation

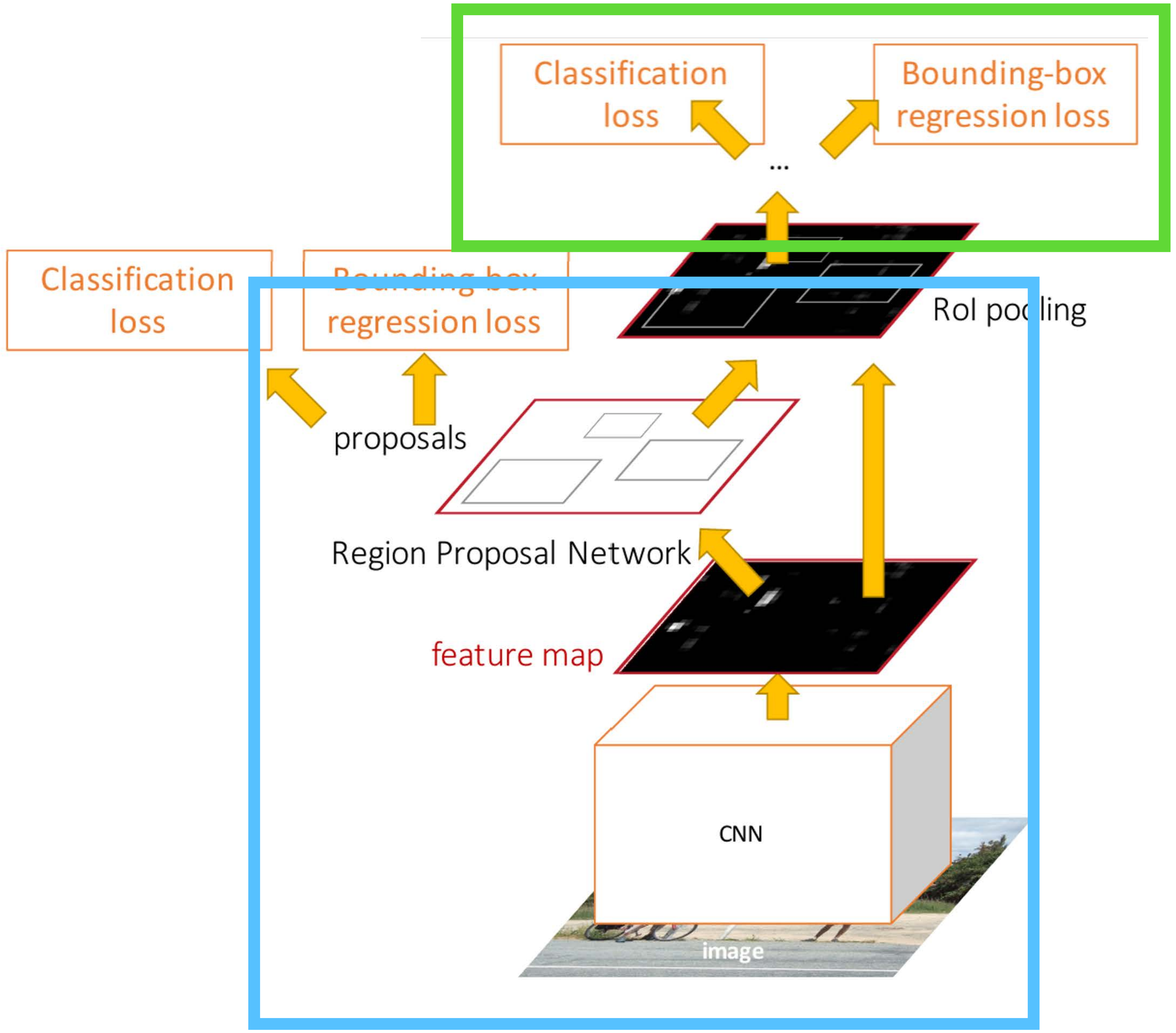
He et al., “Mask R-CNN”, ICCV 2017

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



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

Next Time:

Robot Grasp Learning

DeepRob

Lecture 12
Object Detectors and Segmentation
University of Minnesota