



Introduction to Computer Vision

Lecture 12

Detection and Instance Segmentation

Prof. He Wang

Logistics

- Assignment 4 (Point Cloud Learning, Detection & RNN)
 - will release on 5/24
 - Due on 6/8 11:59PM

Object Detection

Some slides are borrowed from Stanford CS231N.

Computer Vision Tasks

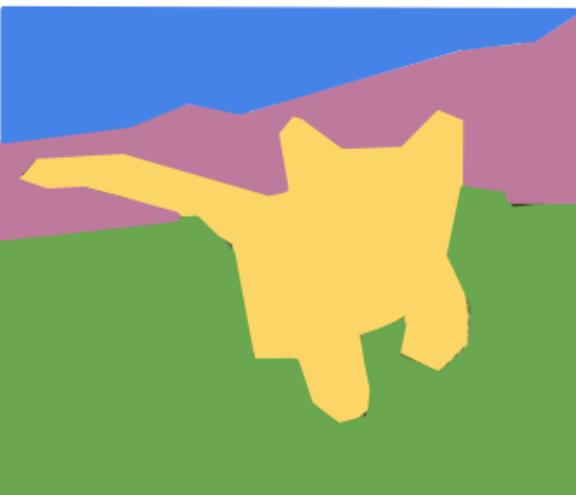
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

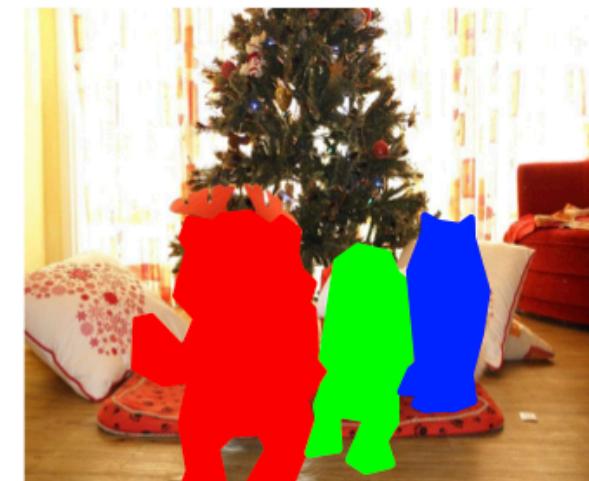
Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation

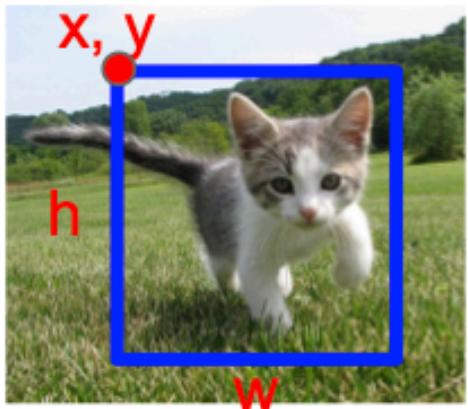


DOG, DOG, CAT

[This image](#) is CC0 public domain

Object Detection: Single Object

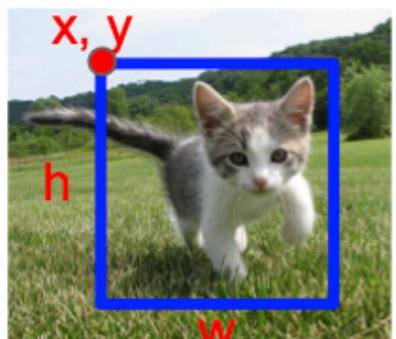
- Task: localization + classification
- Output: 2D (axis aligned) bounding box
 - How many degree-of-freedom?
 - 4 DoF
 - How to parameterize such a bounding box?
 - x, y, h, w



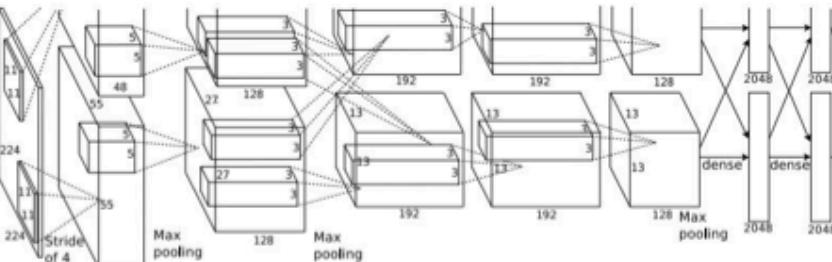
[This image is CC0 public domain](#)

Object Detection: Single Object

- Localization + Classification



This image is CC0 public domain



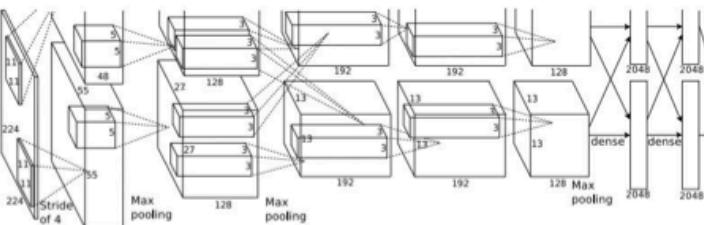
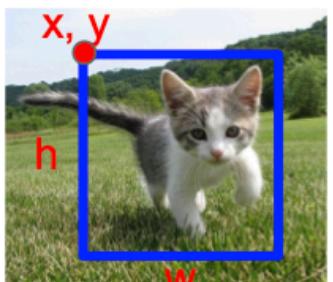
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...
Fully Connected:
4096 to 1000

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

Object Detection: Single Object



Fully
Connected:
4096 to 1000

Vector: Fully
Connected:
4096 to 4

Treat localization as a
regression problem!

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

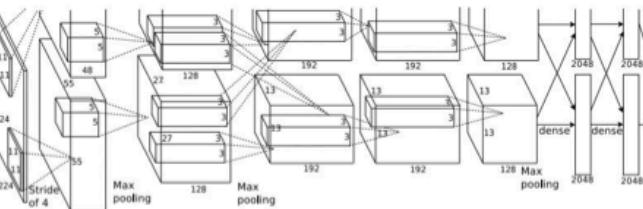
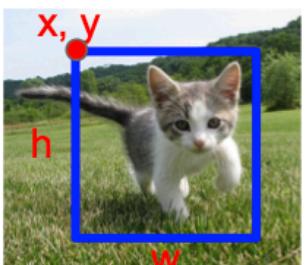
Box
Coordinates → L2 Loss
(x, y, w, h)

Correct label:
Cat

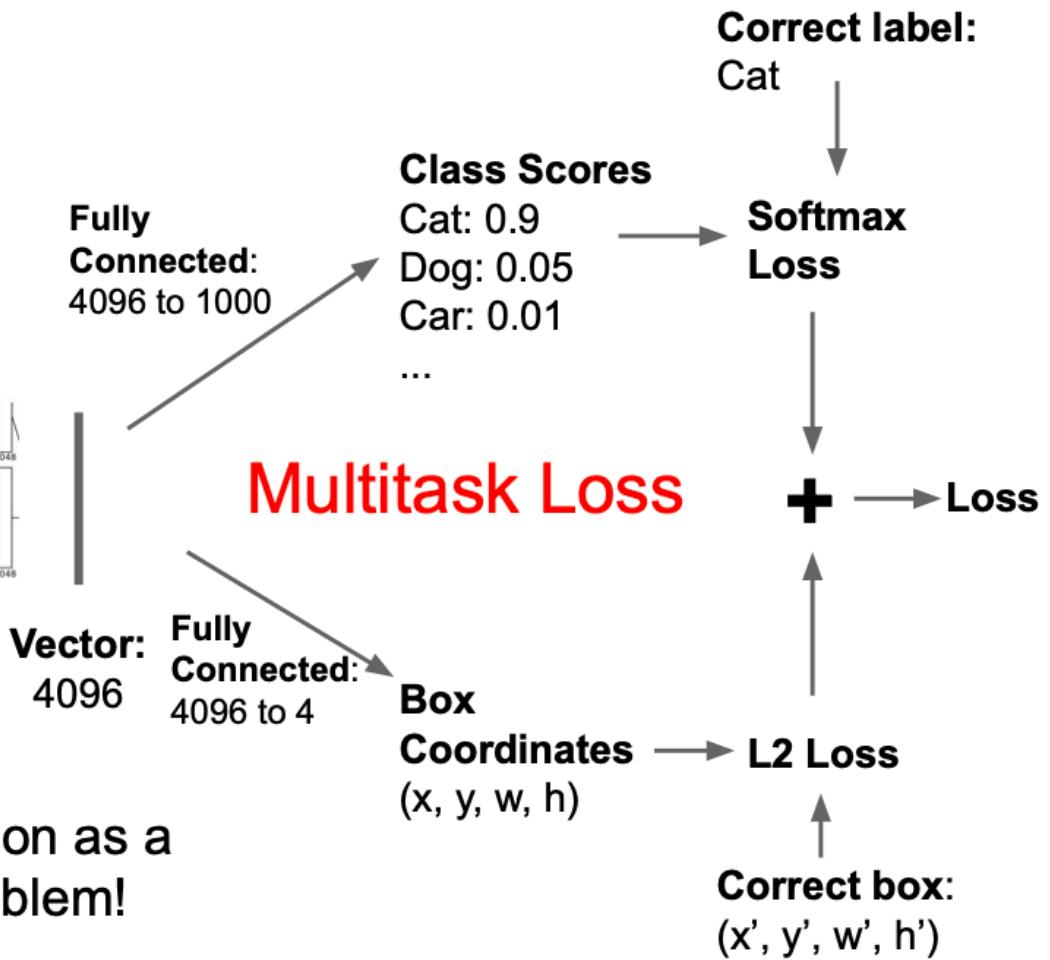
Softmax
Loss

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

Object Detection: Single Object



Treat localization as a
regression problem!



Regression Loss

- Error: $(\Delta x, \Delta y, \Delta w, \Delta h)$
- L1 loss: $\sum |\Delta_i|$ – robust, however not good at convergence
- L2 loss: $\sum \Delta_i^2$ (not the same to L2 norm) – not robust to a larger error, however good at convergence
- Rooted mean squared loss (RMSE): $\sqrt{\frac{1}{N} \sum \Delta_i^2}$ – the gradient of sqrt function is bad at 0

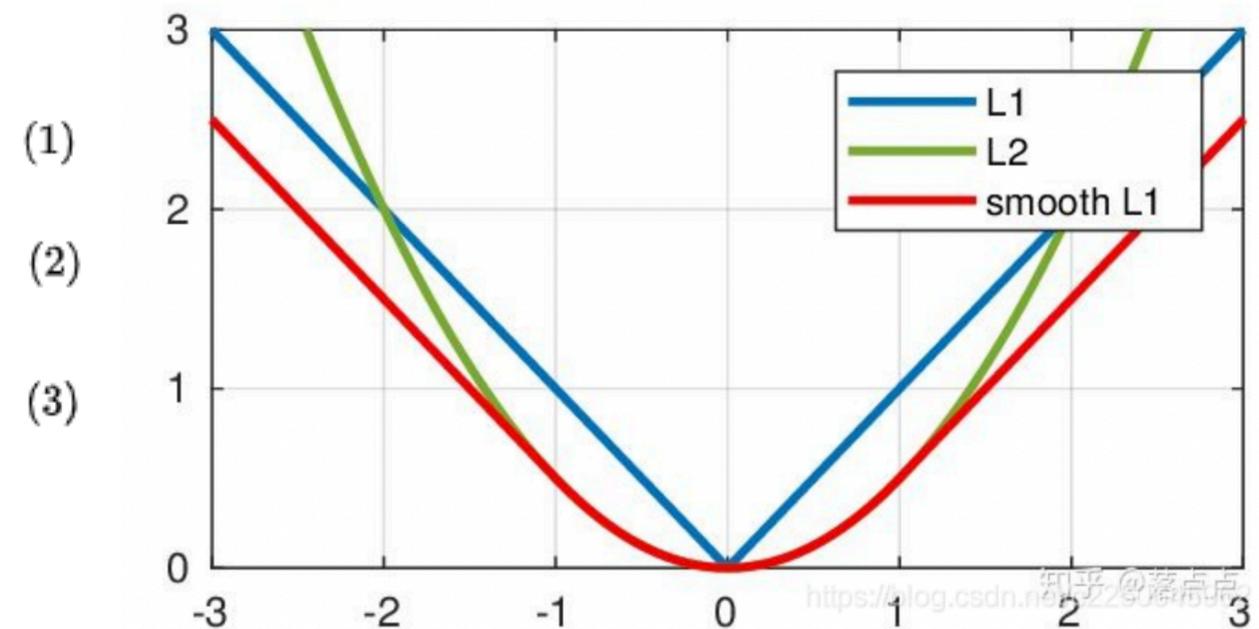
Regression Loss

- Smooth L1 loss (proposed by Fast RCNN, very similar to Huber loss widely used in robust optimization)

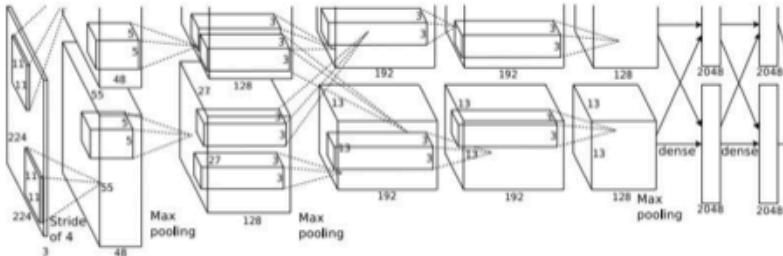
$$L_2(x) = x^2$$

$$L_1(x) = |x|$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

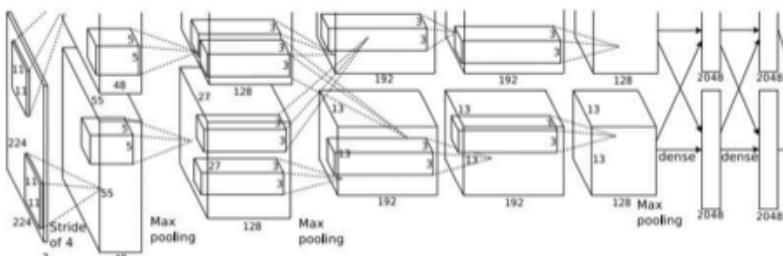


Object Detection: Multiple Objects



CAT: (x, y, w, h)

1 bounding box

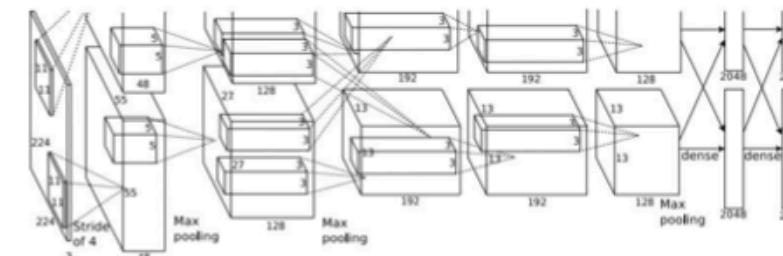


DOG: (x, y, w, h)

3 bounding boxes

DOG: (x, y, w, h)

CAT: (x, y, w, h)



DUCK: (x, y, w, h)

Many bounding boxes!

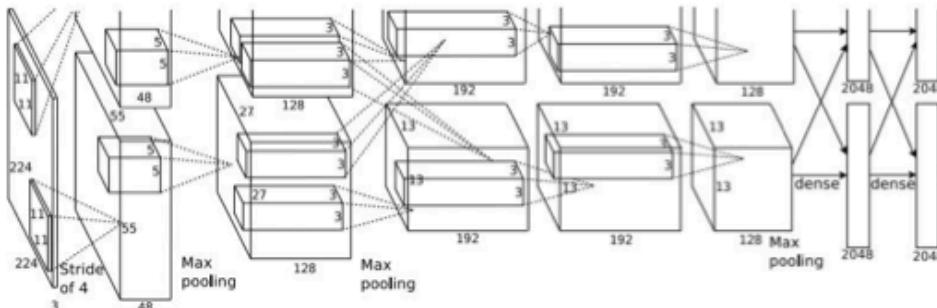
DUCK: (x, y, w, h)

....

Different images need different numbers of outputs!

Sliding-Window based Multi-Object Detection

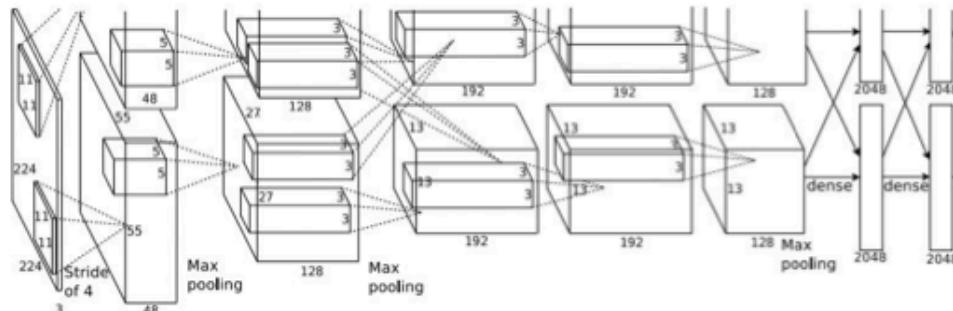
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Sliding-Window based Multi-Object Detection

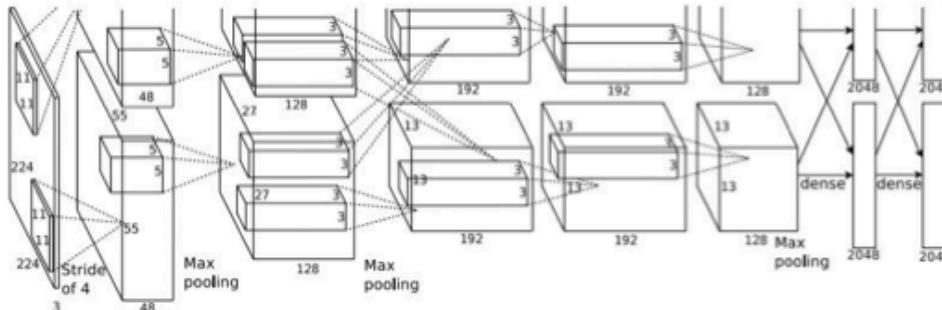
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Sliding-Window based Multi-Object Detection

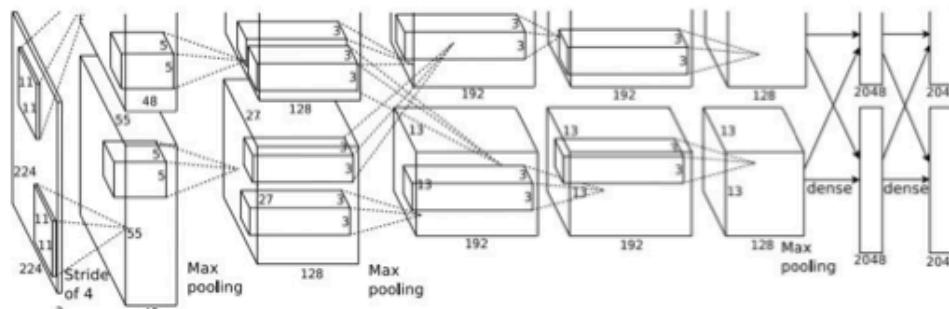
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Sliding-Window based Multi-Object Detection

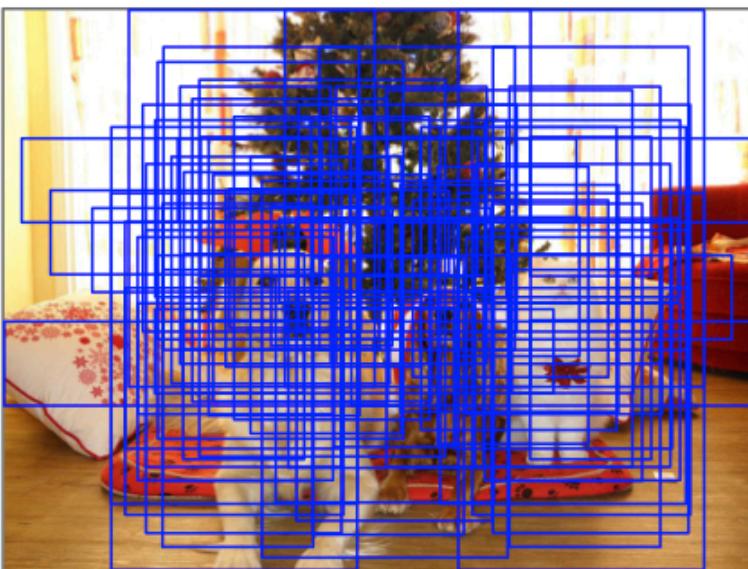
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



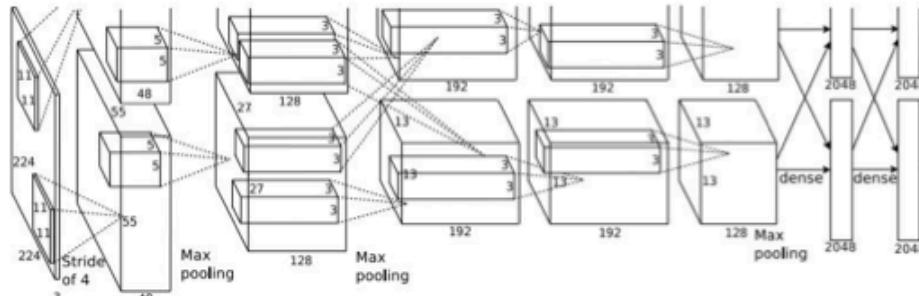
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Sliding-Window based Multi-Object Detection



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

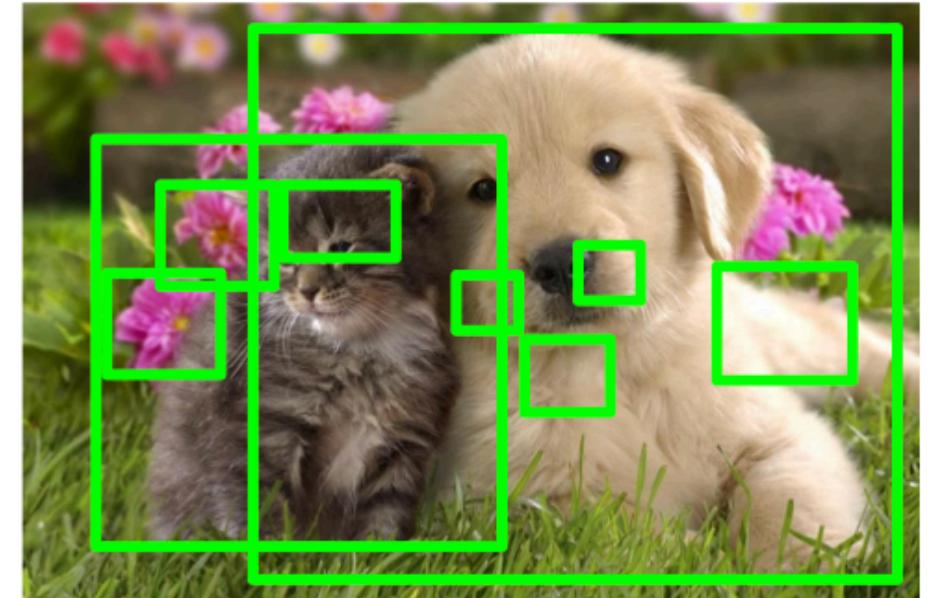


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, "Measuring the objectness of image windows", TPAMI 2012

Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014

Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

R-CNN

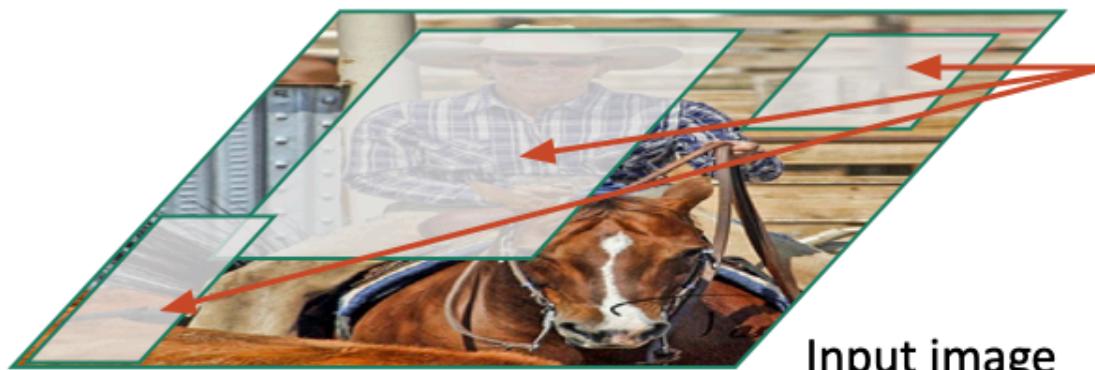


Input image

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

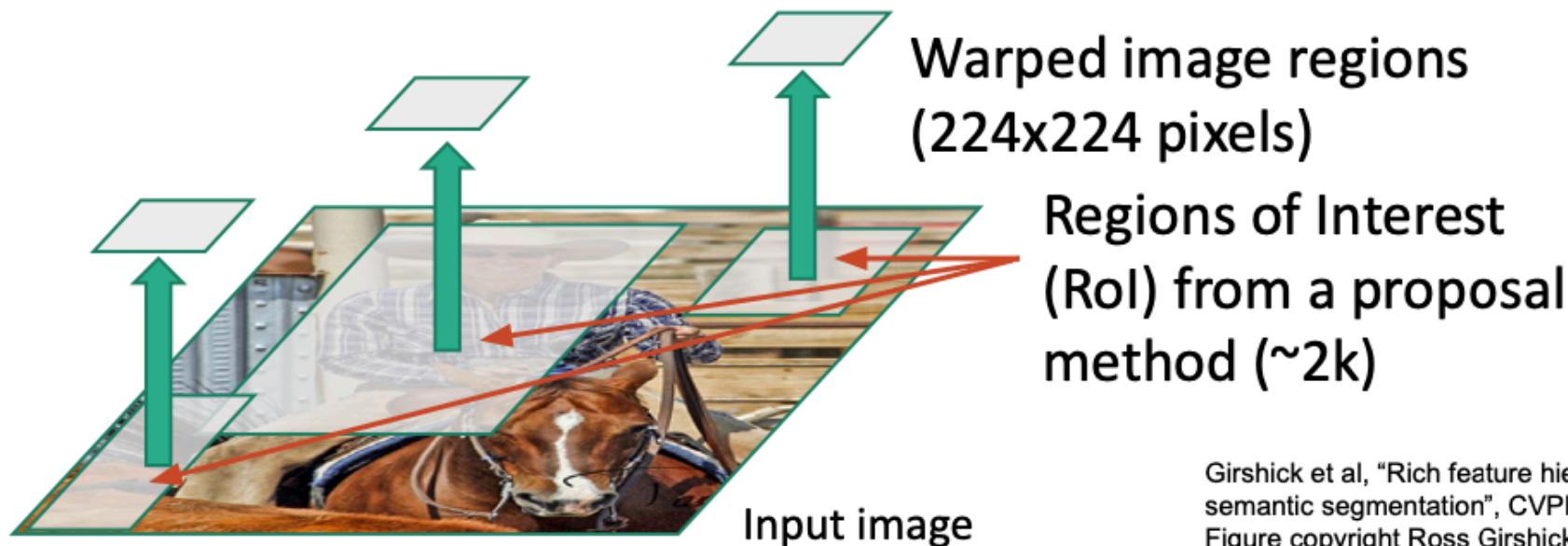


Input image

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

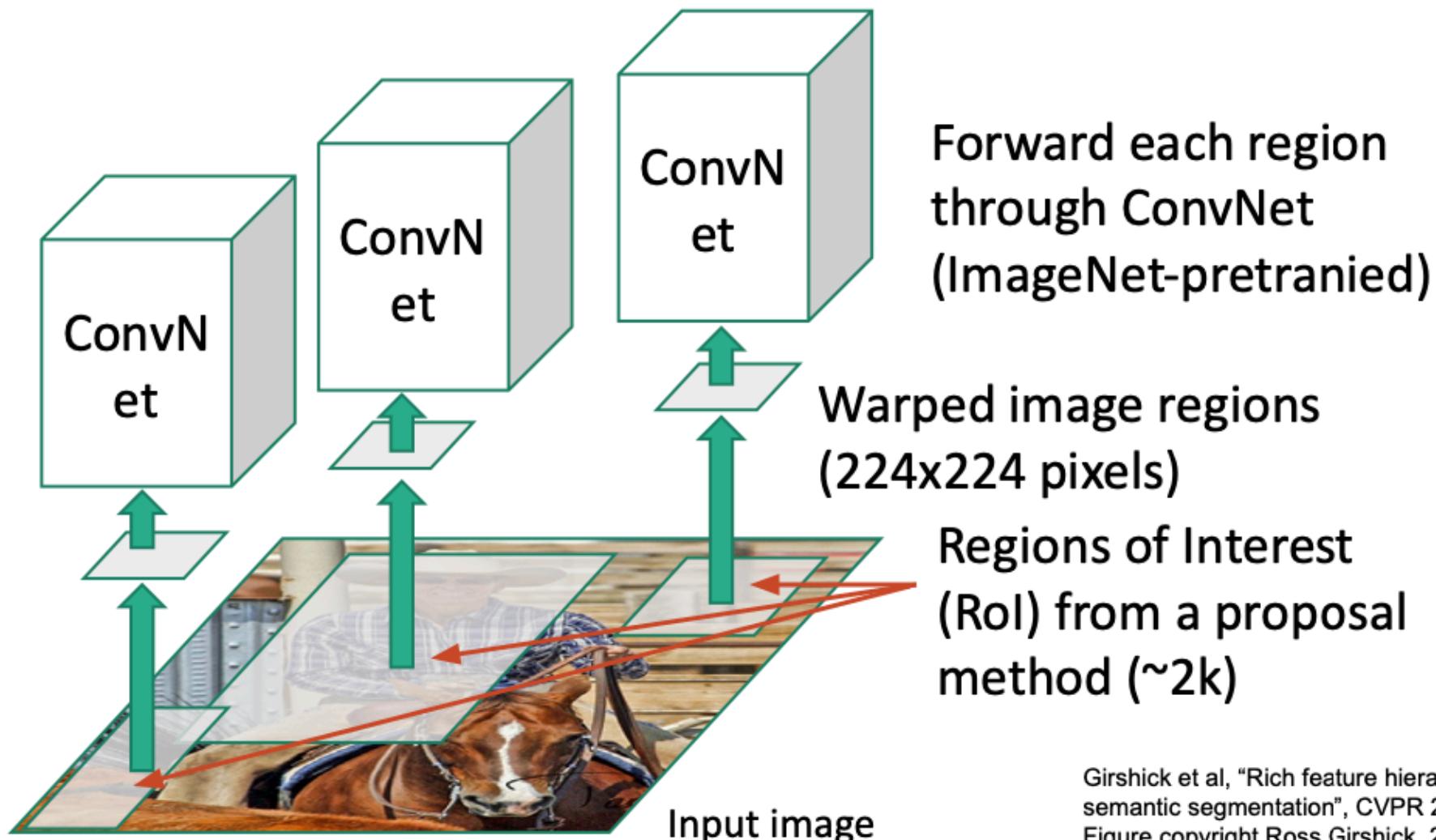
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN



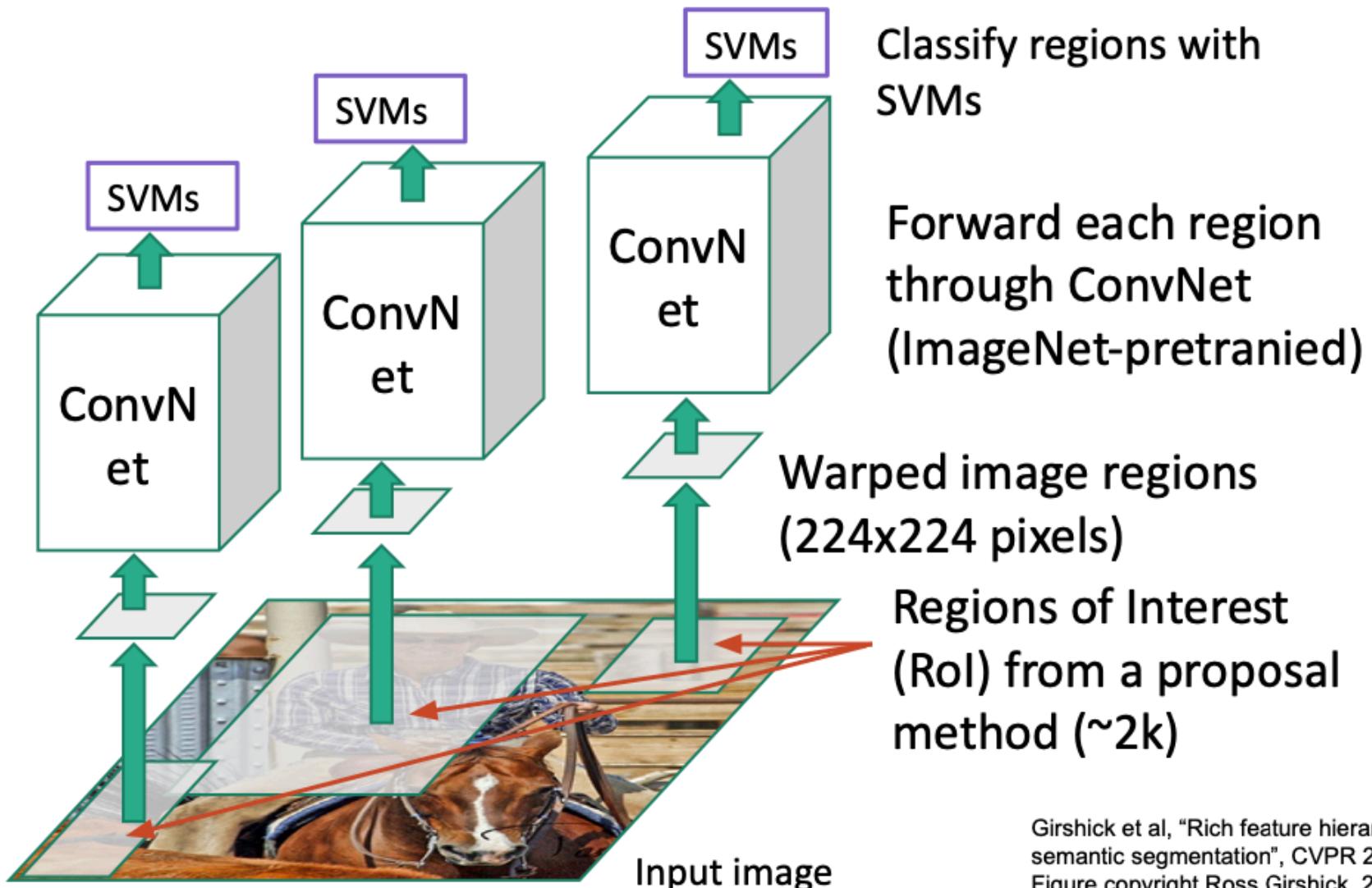
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

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.

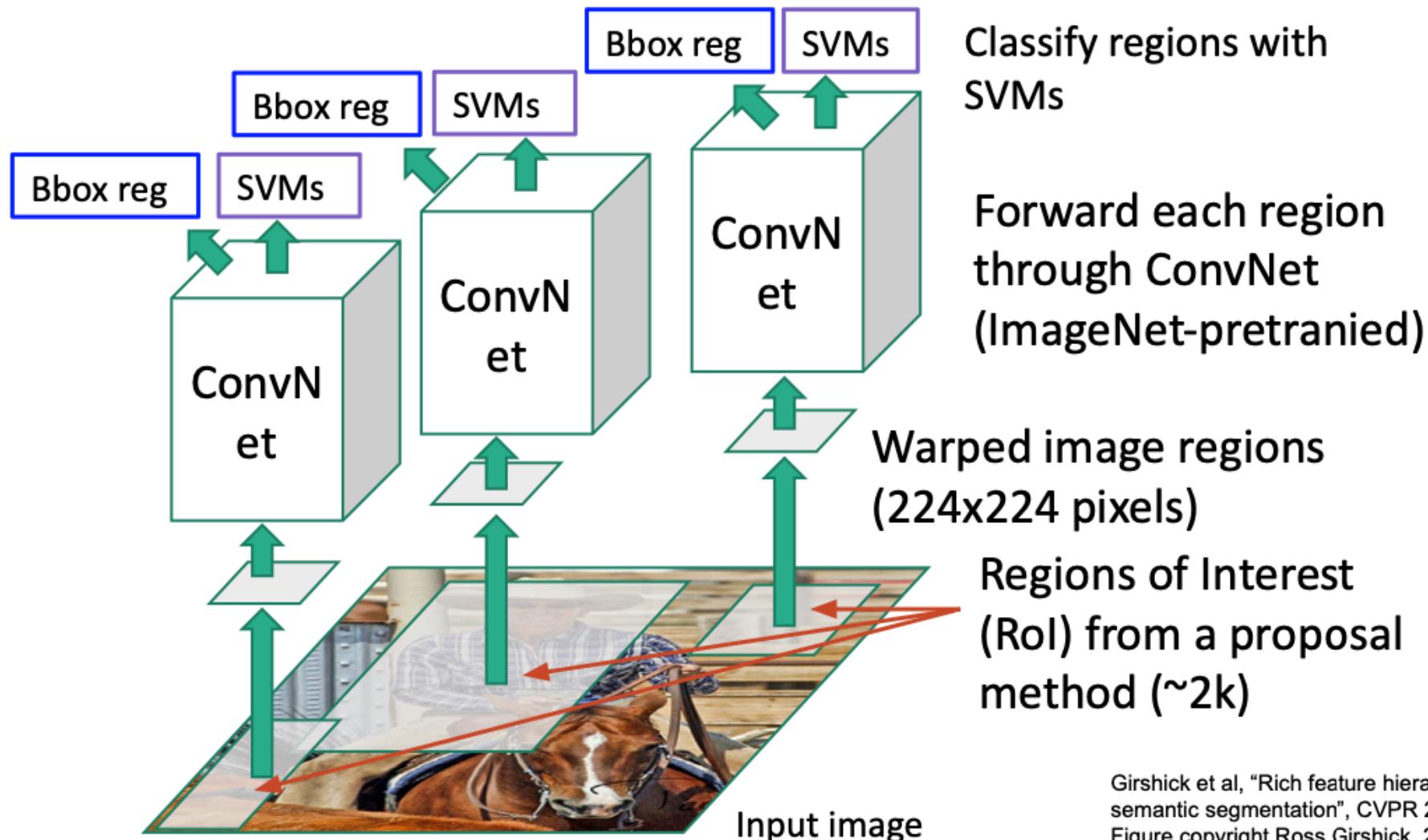
R-CNN



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

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

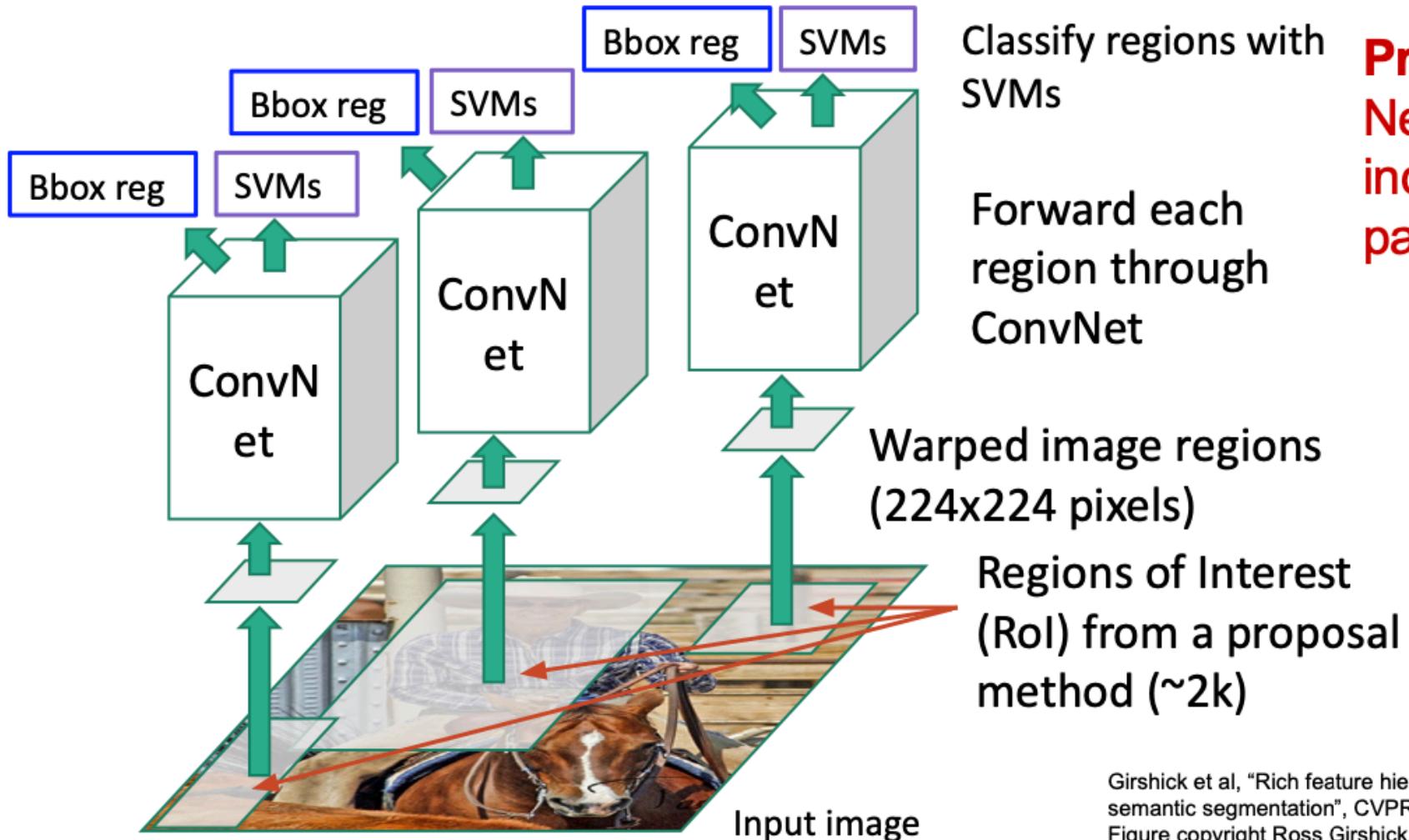


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission. 23

R-CNN

Predict “corrections” to the Roi: 4 numbers: (dx, dy, dw, dh)

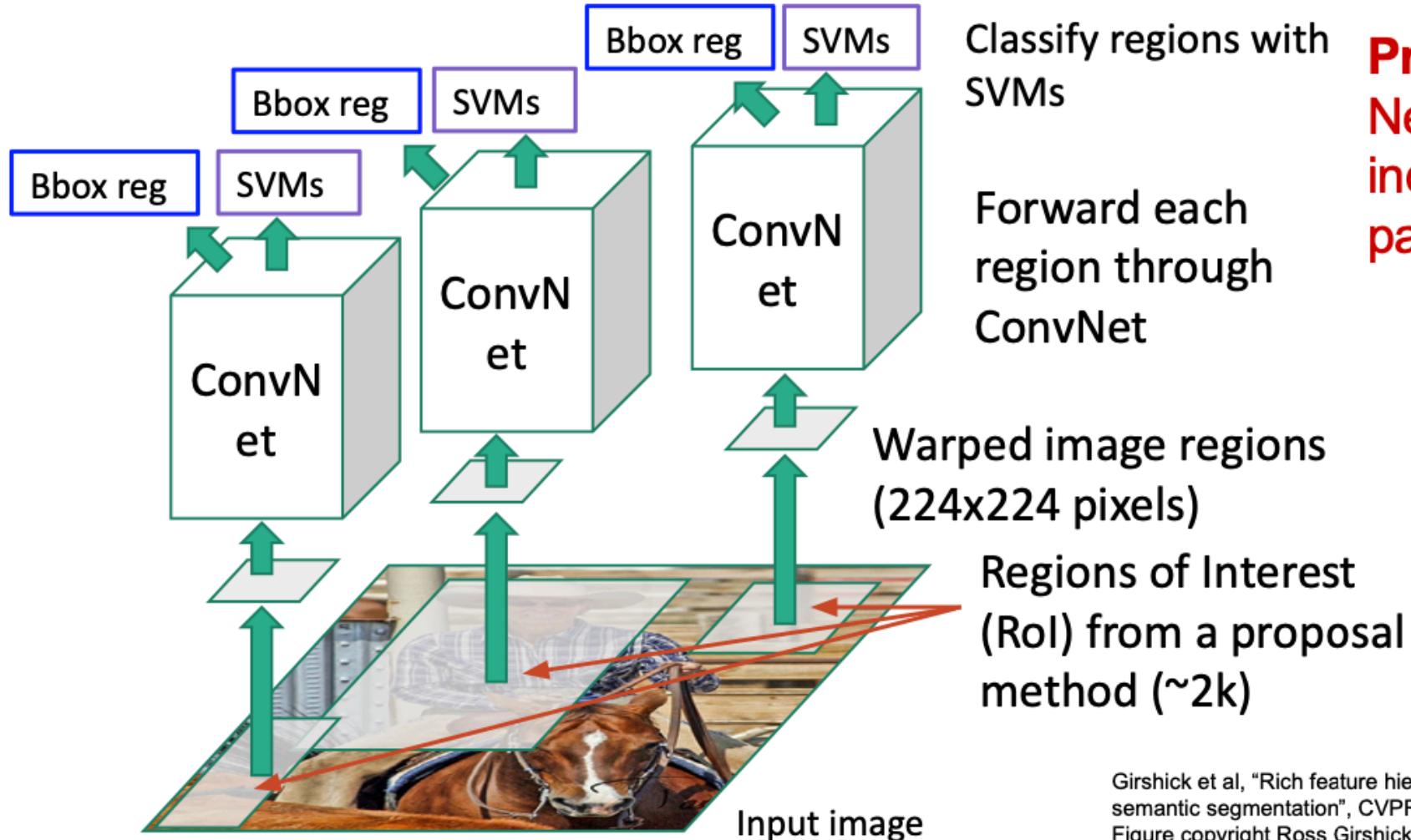


Problem: Very slow!
Need to do ~2k independent forward passes for each image!

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

R-CNN

Predict “corrections” to the Roi: 4 numbers: (dx, dy, dw, dh)



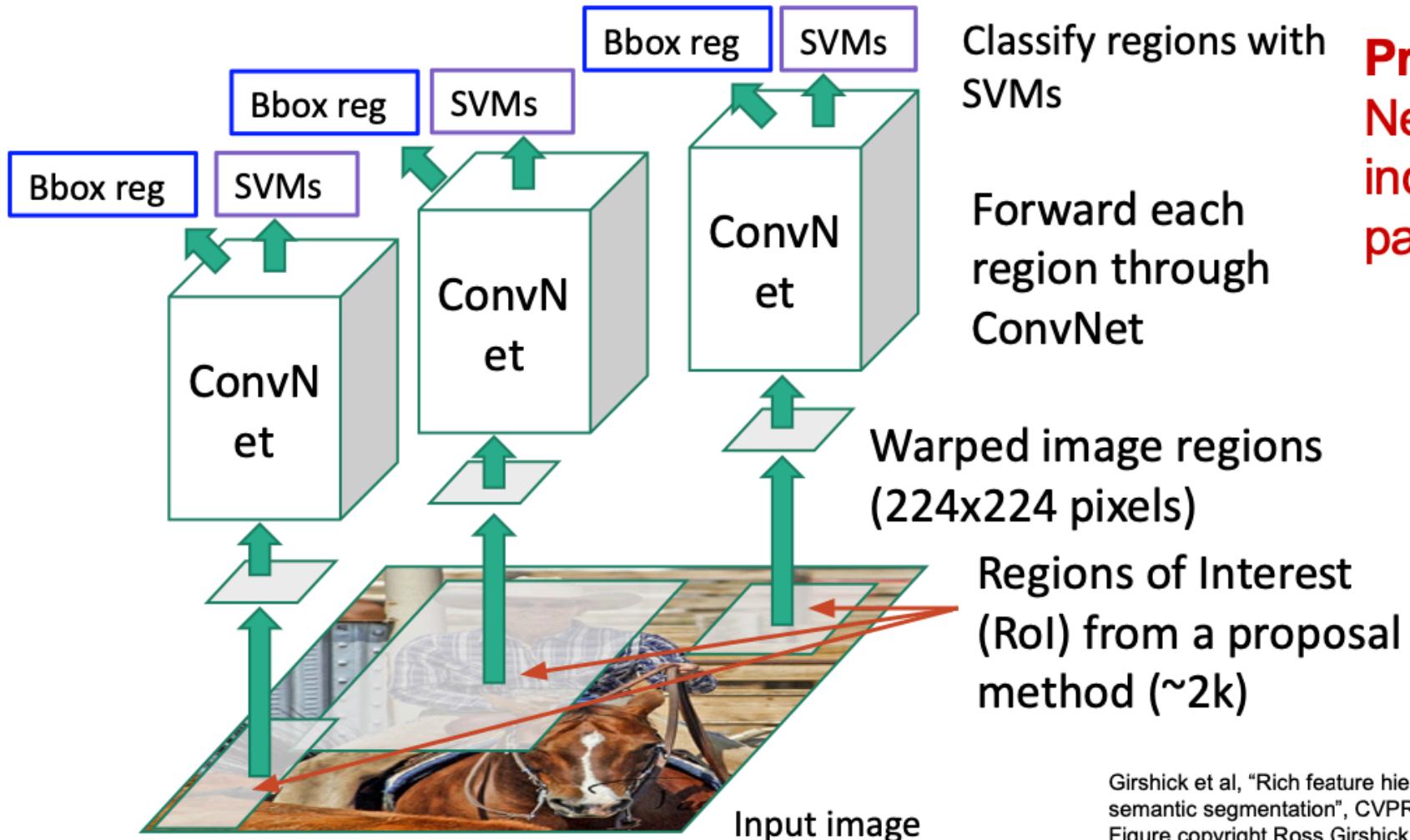
Problem: Very slow!
Need to do ~2k independent forward passes for each image!

Problem 2: The cropped region doesn't contain sufficient information to regress bounding box refinements.

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

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

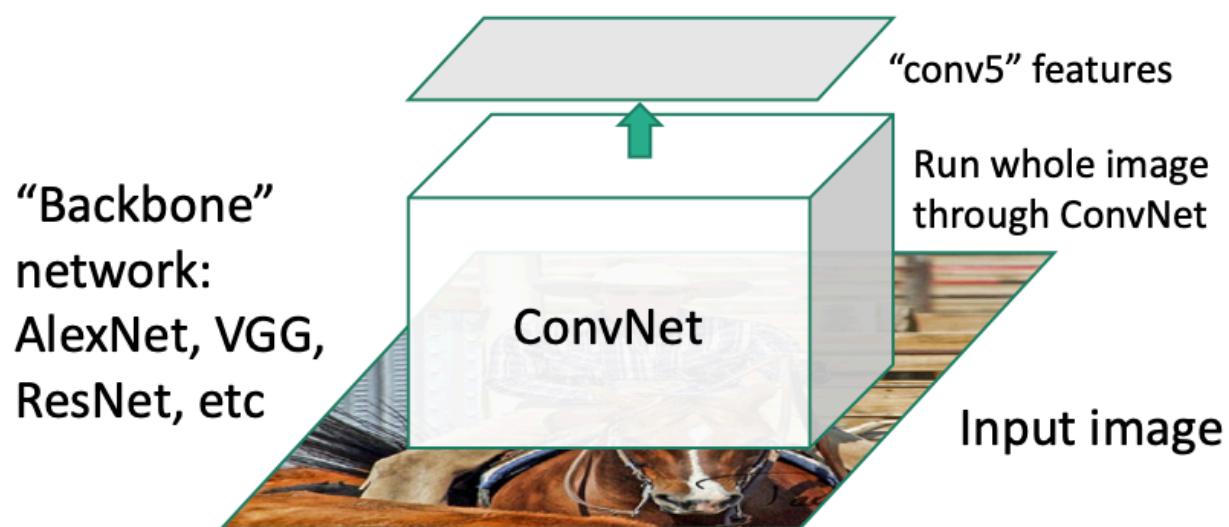


Problem: Very slow!
Need to do ~2k independent forward passes for each image!

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

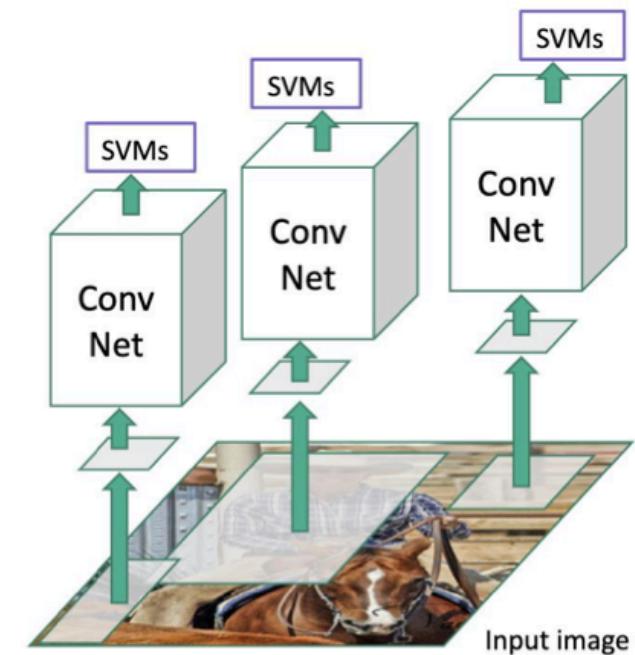
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN



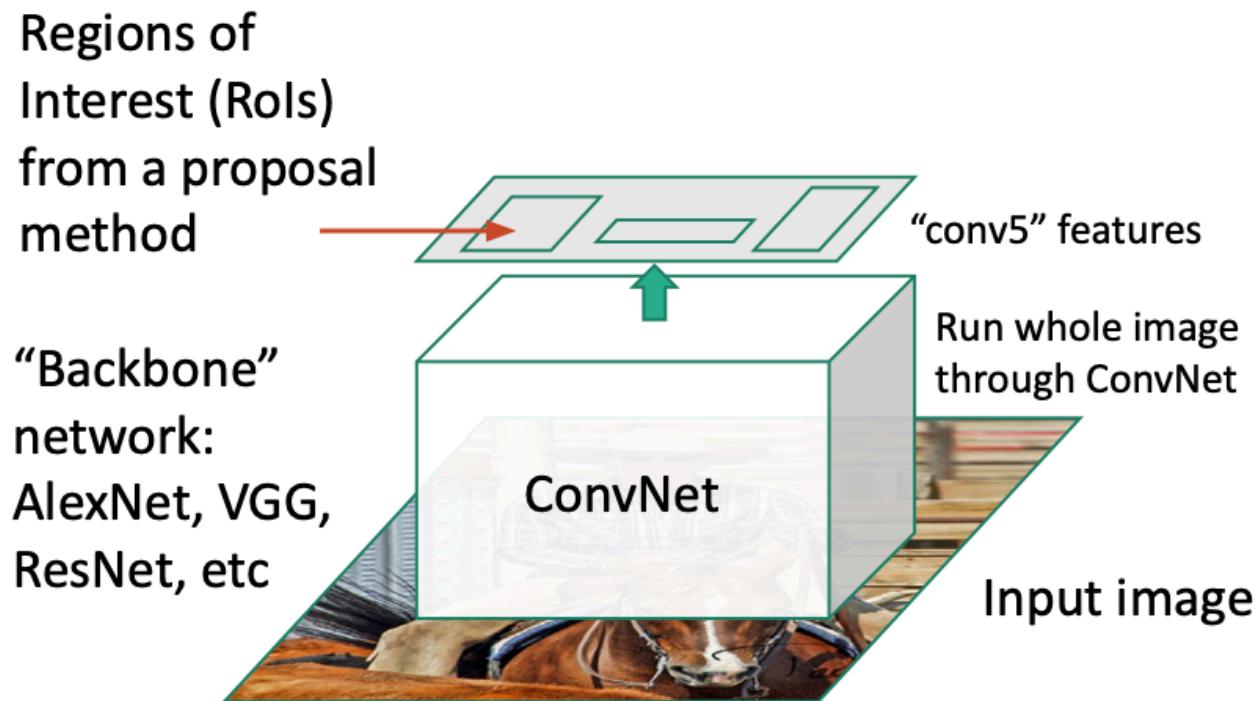
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast RCNN



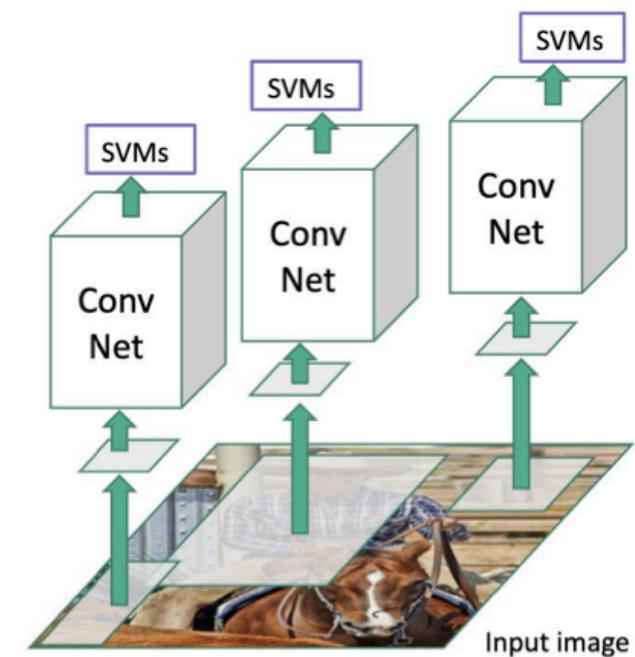
R-CNN

Fast R-CNN



Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast RCNN

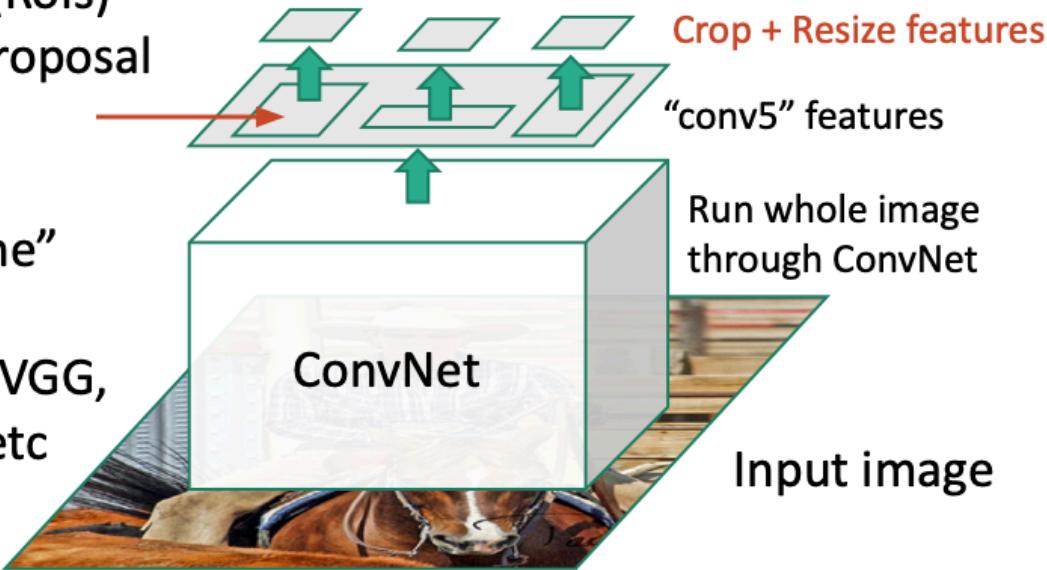


R-CNN

Fast R-CNN

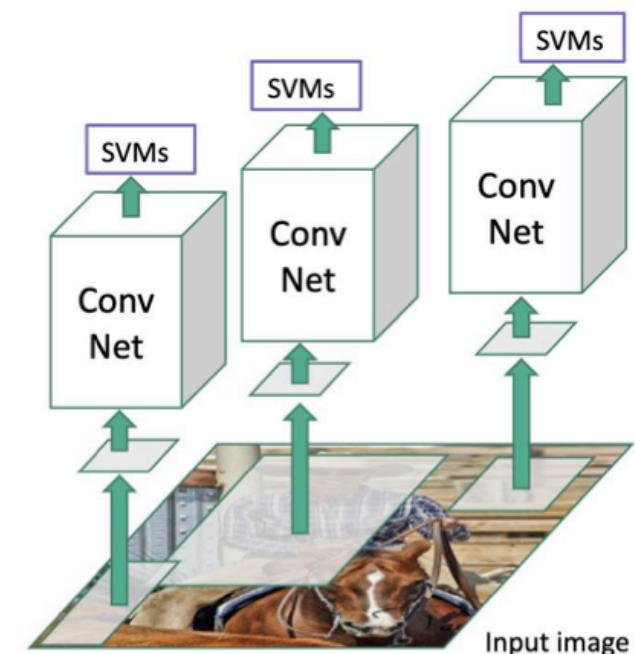
Regions of Interest (RoIs)
from a proposal
method

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



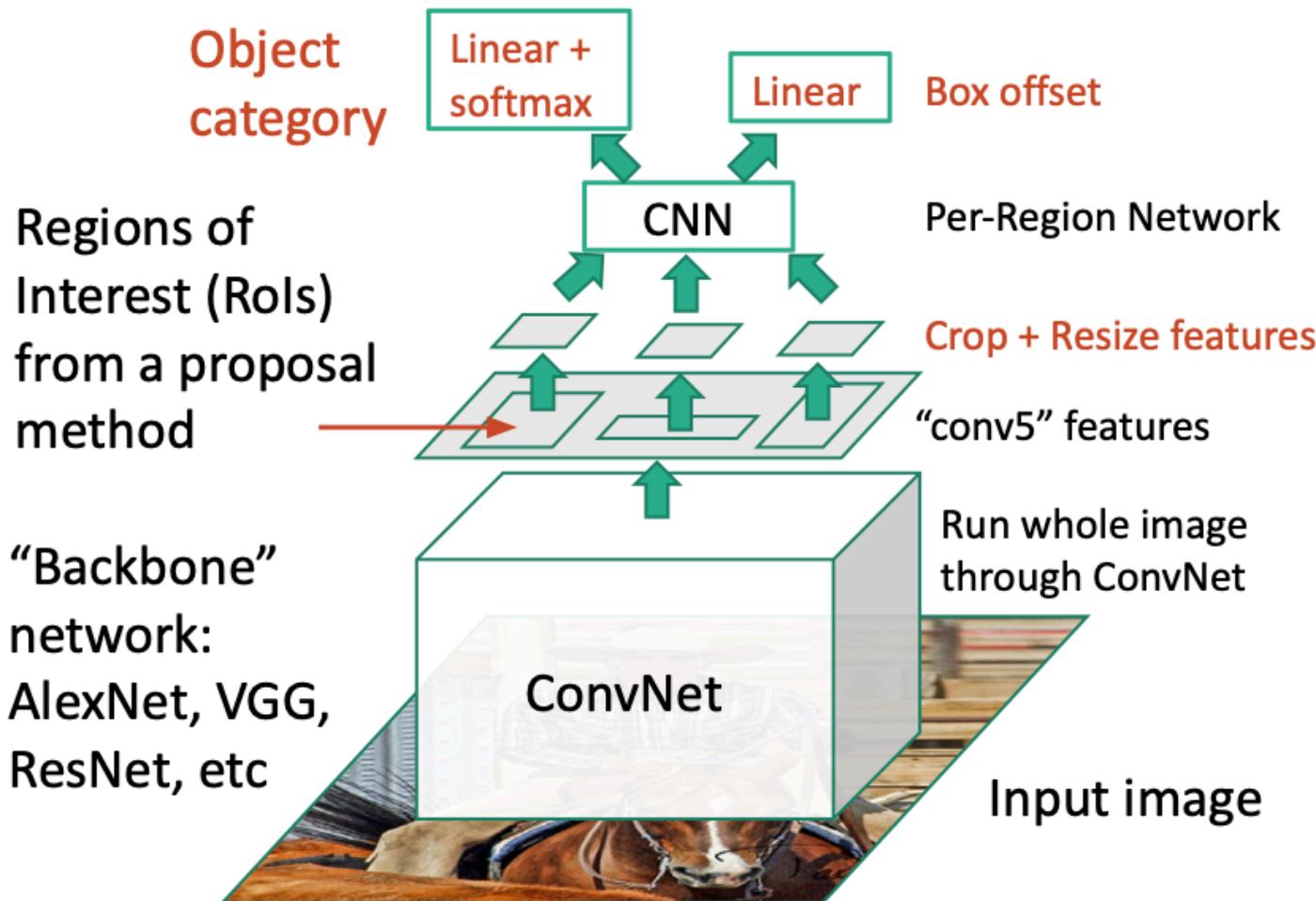
Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast RCNN



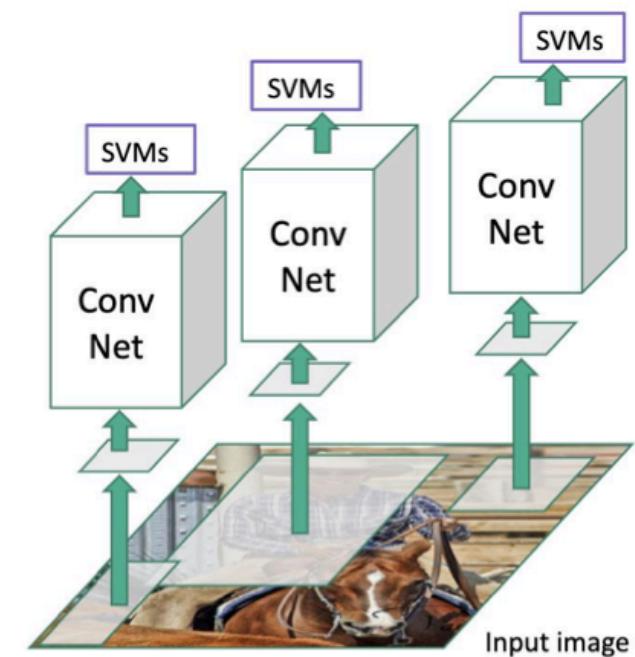
R-CNN

Fast R-CNN



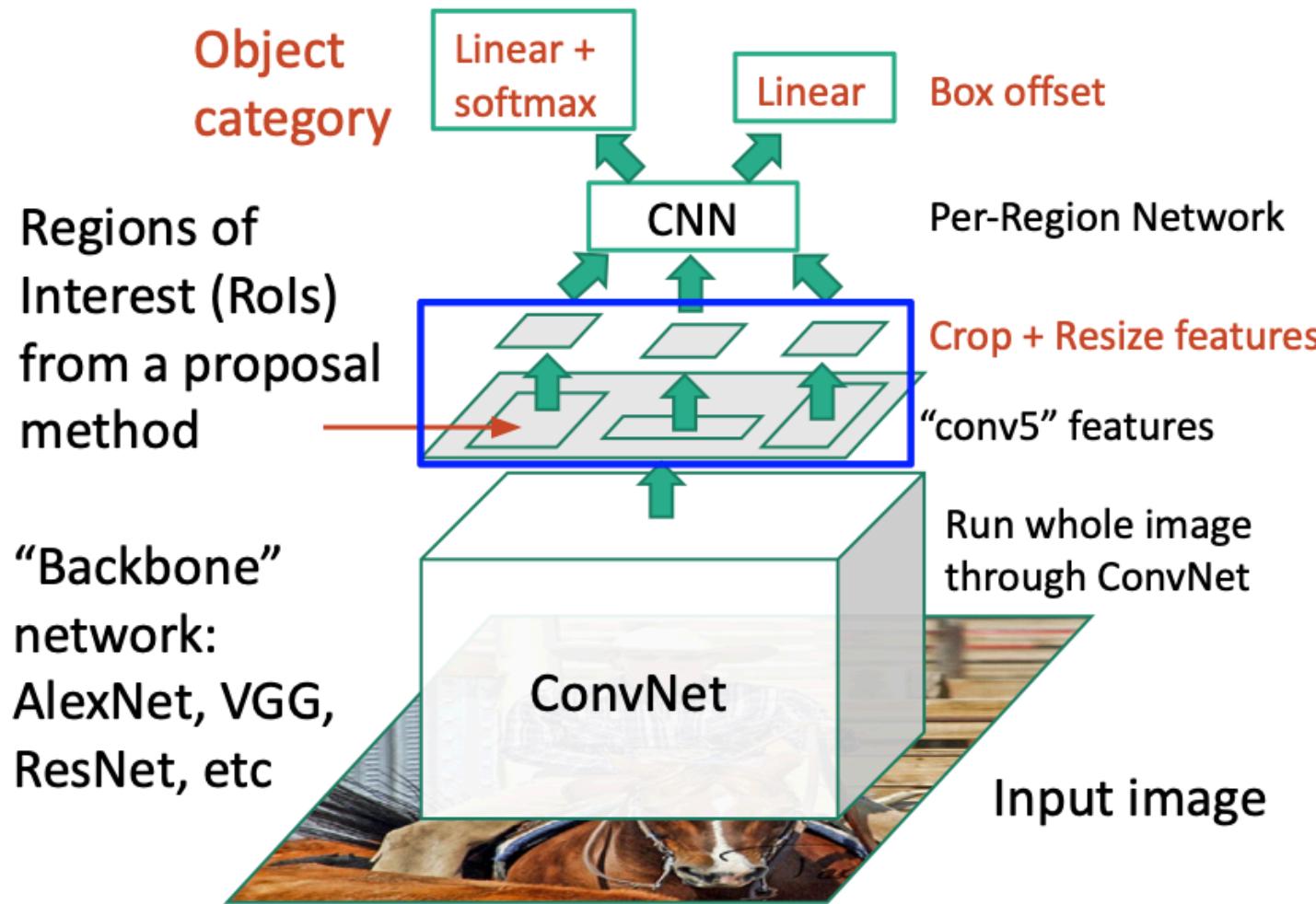
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast RCNN

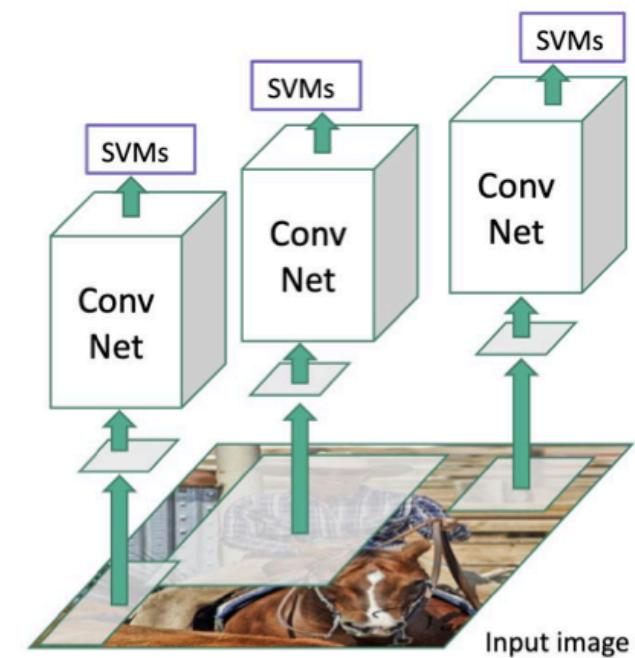


R-CNN

Fast R-CNN

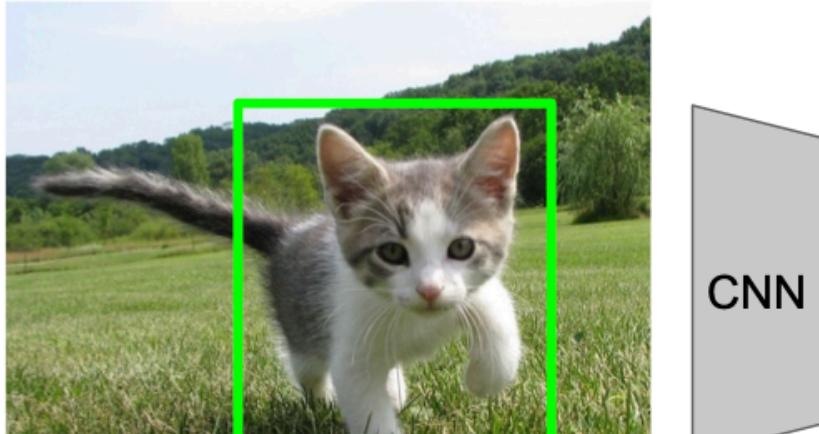


Fast RCNN



R-CNN

Cropping Features: RoI Pool



Input Image
(e.g. $3 \times 640 \times 480$)

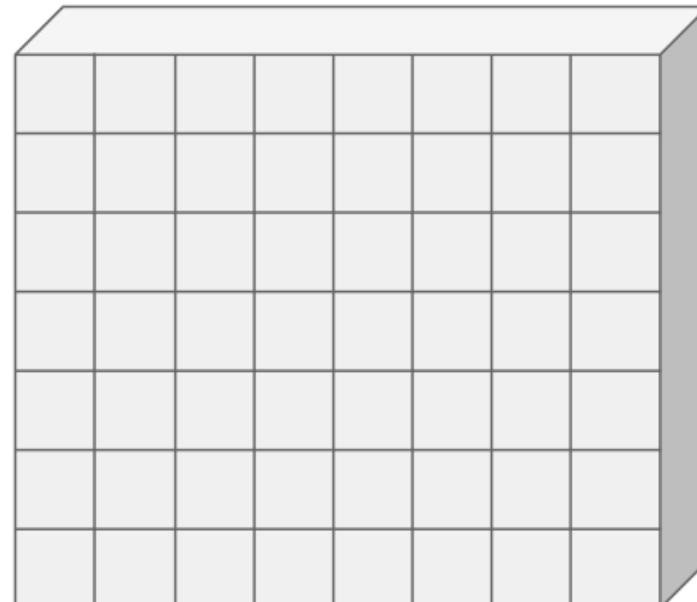
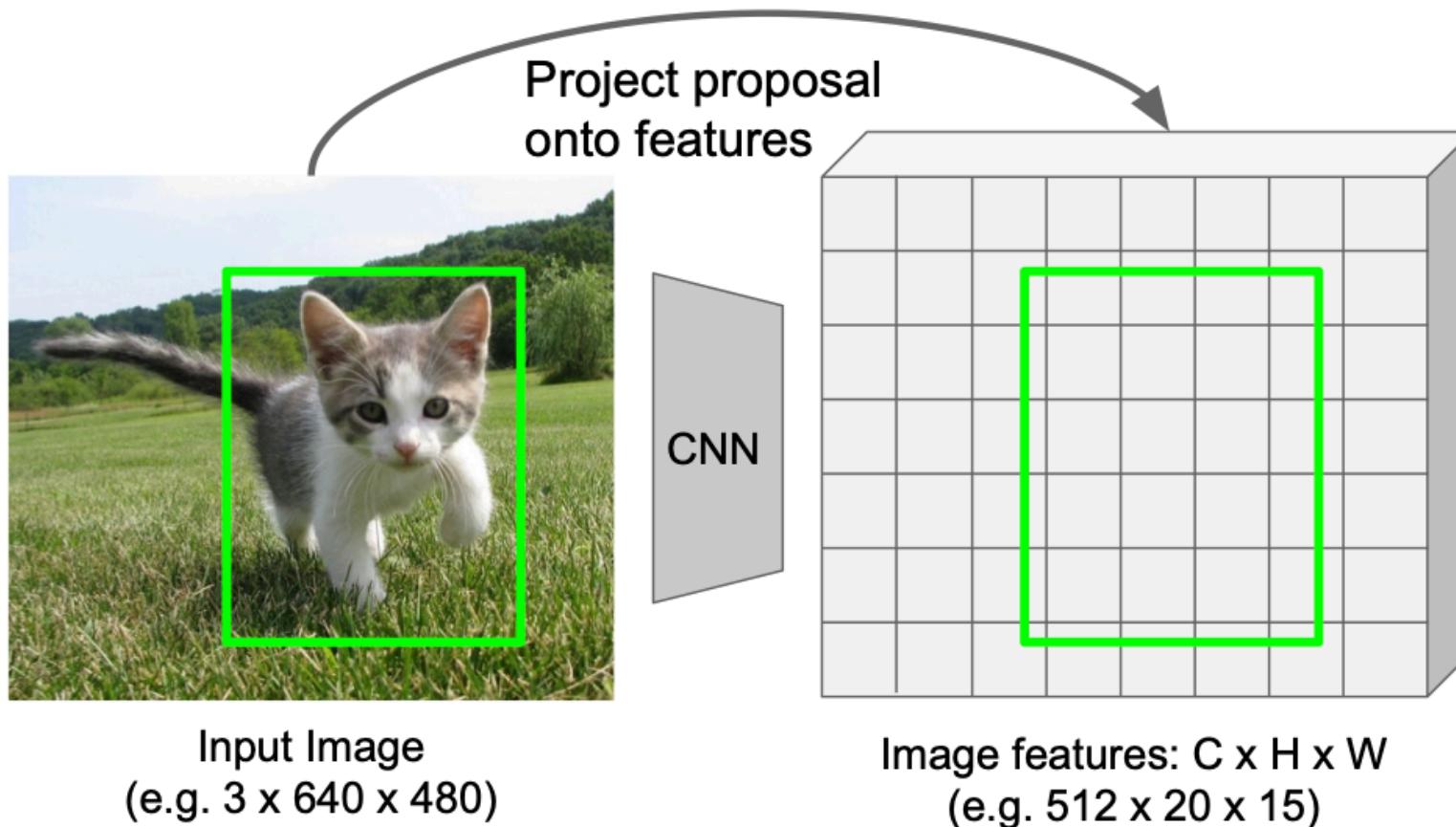
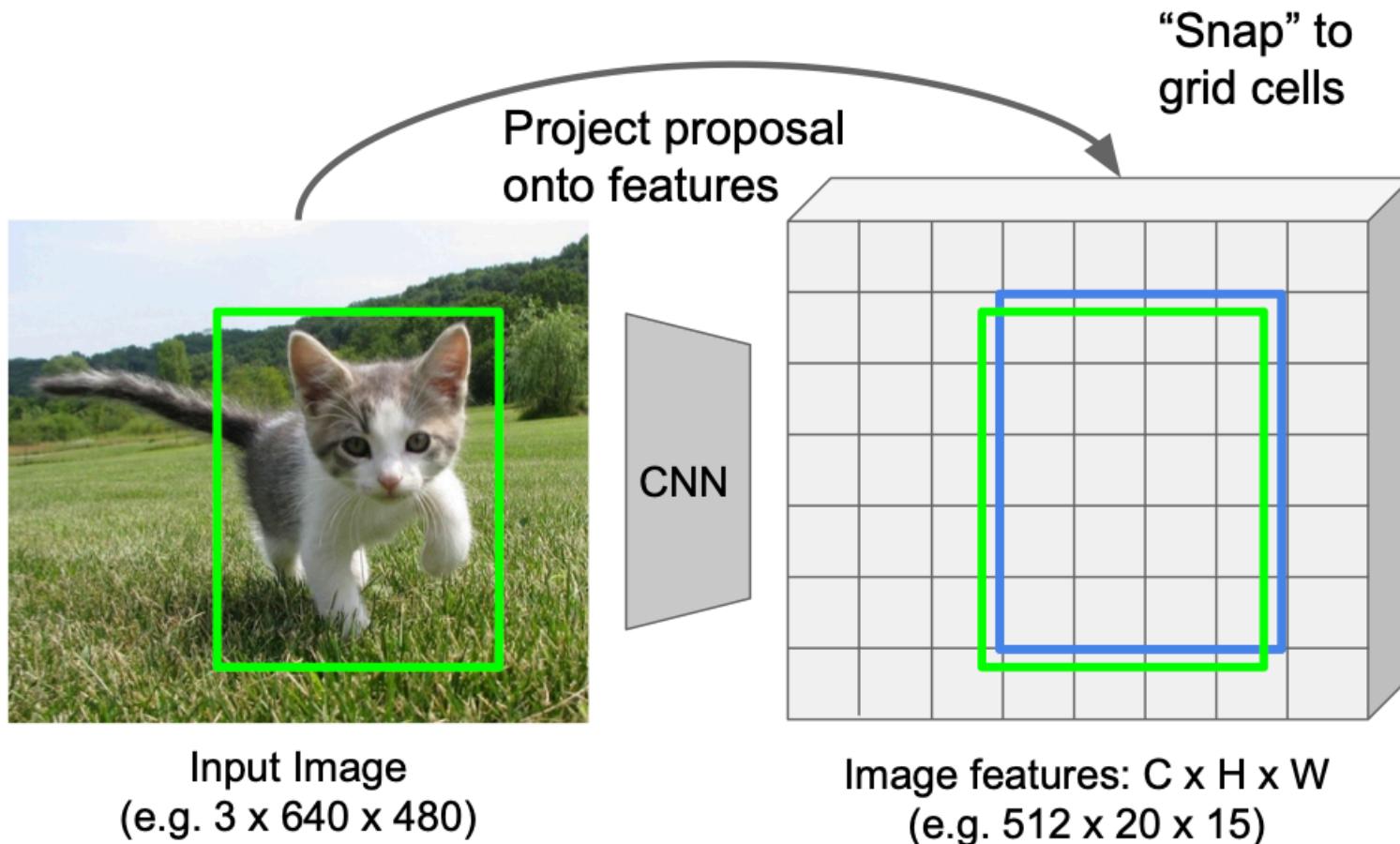


Image features: $C \times H \times W$
(e.g. $512 \times 20 \times 15$)

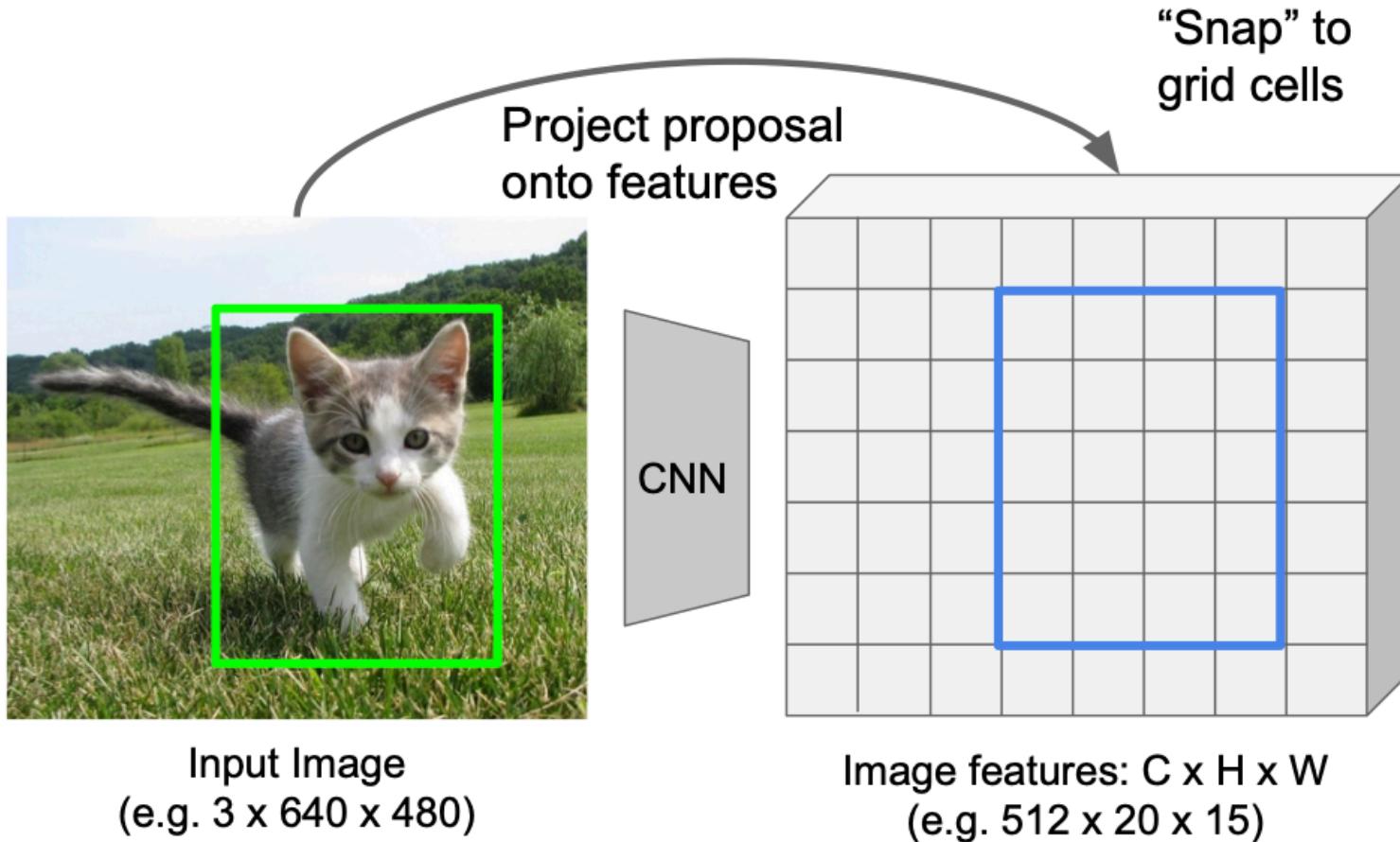
Cropping Features: RoI Pool



Cropping Features: RoI Pool

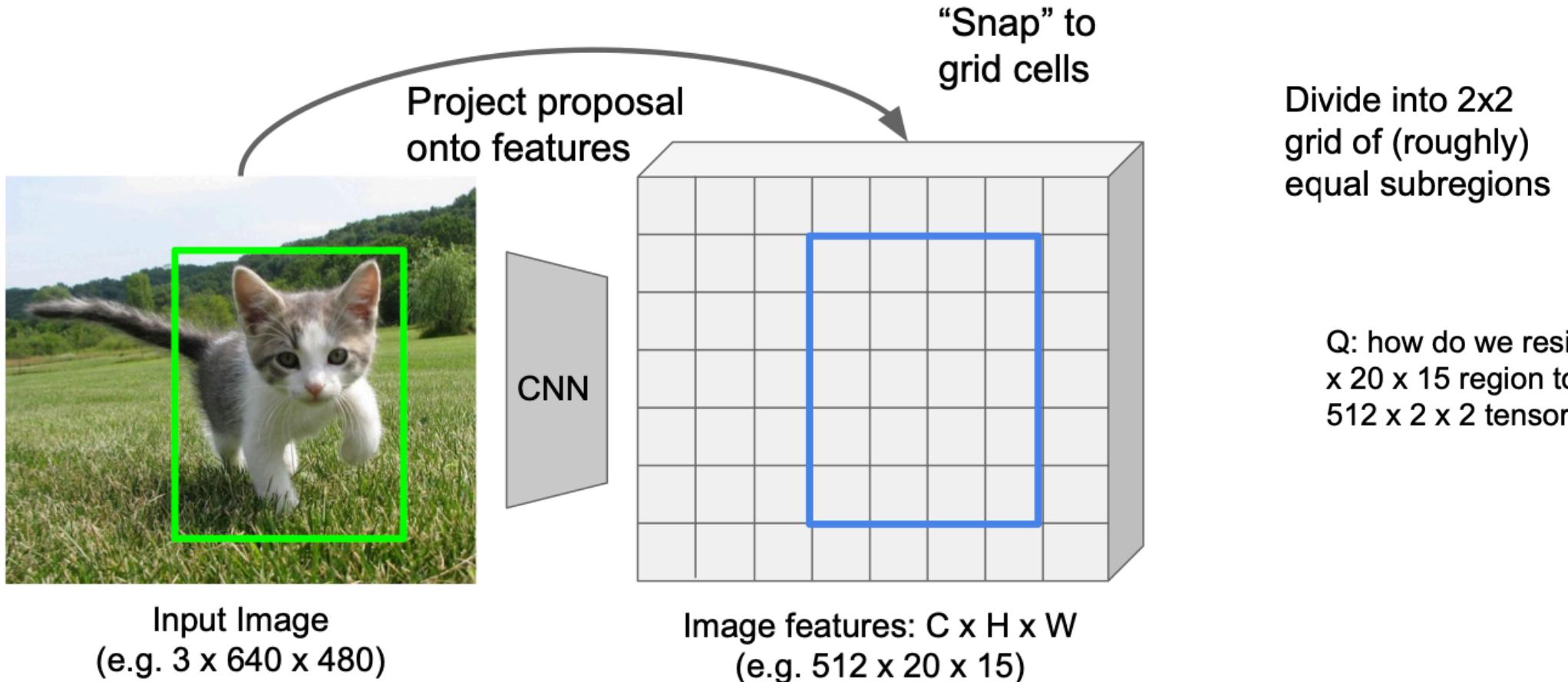


Cropping Features: RoI Pool



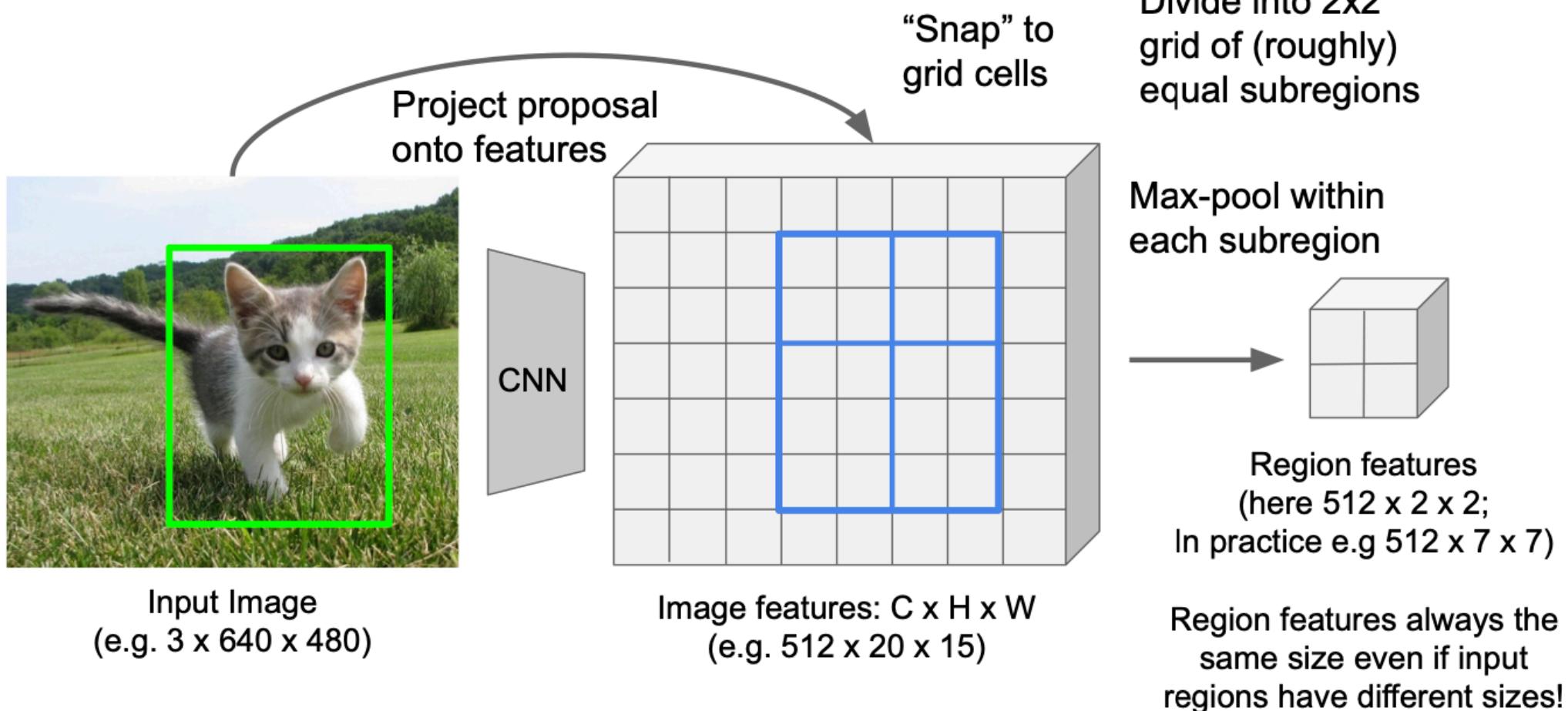
Q: how do we resize the 512 x 20 x 15 region to, e.g., a 512 x 2 x 2 tensor?

Cropping Features: RoI Pool



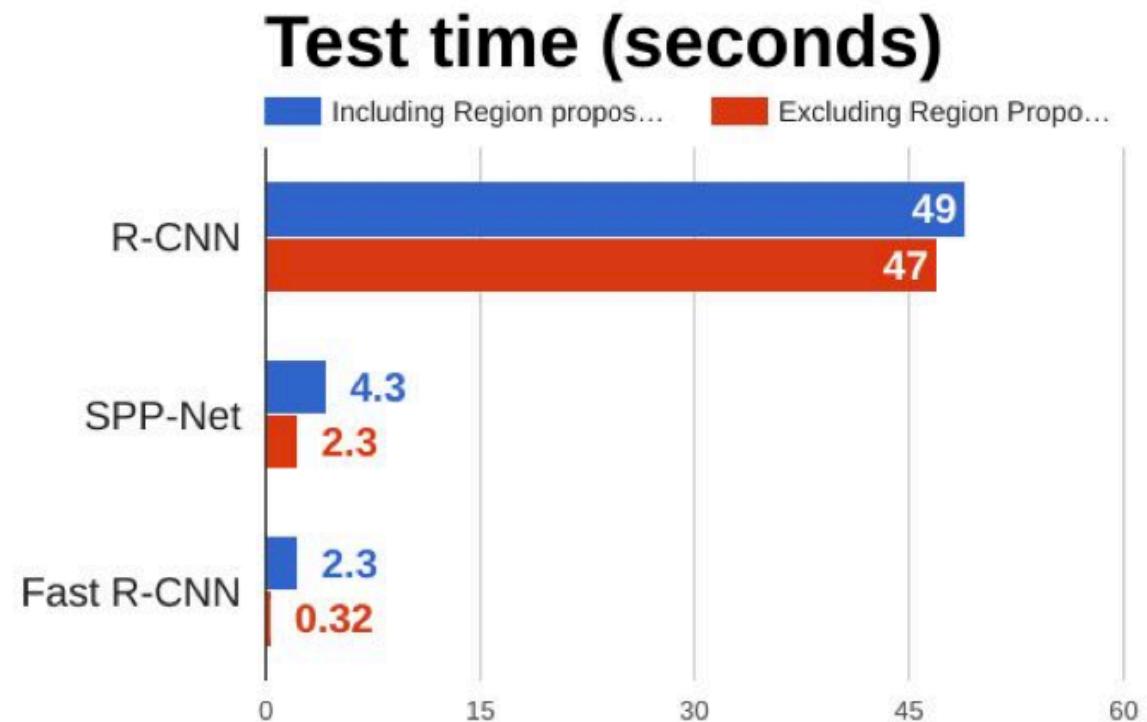
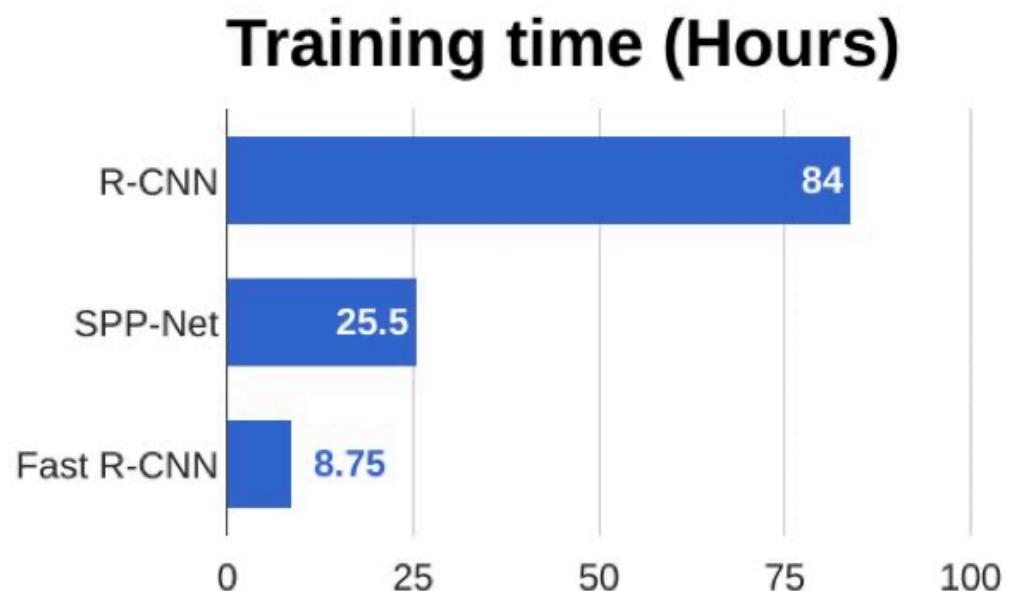
Q: how do we resize the 512 x 20 x 15 region to, e.g., a 512 x 2 x 2 tensor?.

Cropping Features: RoI Pool



Girshick, “Fast R-CNN”, ICCV 2015.

R-CNN vs. Fast R-CNN

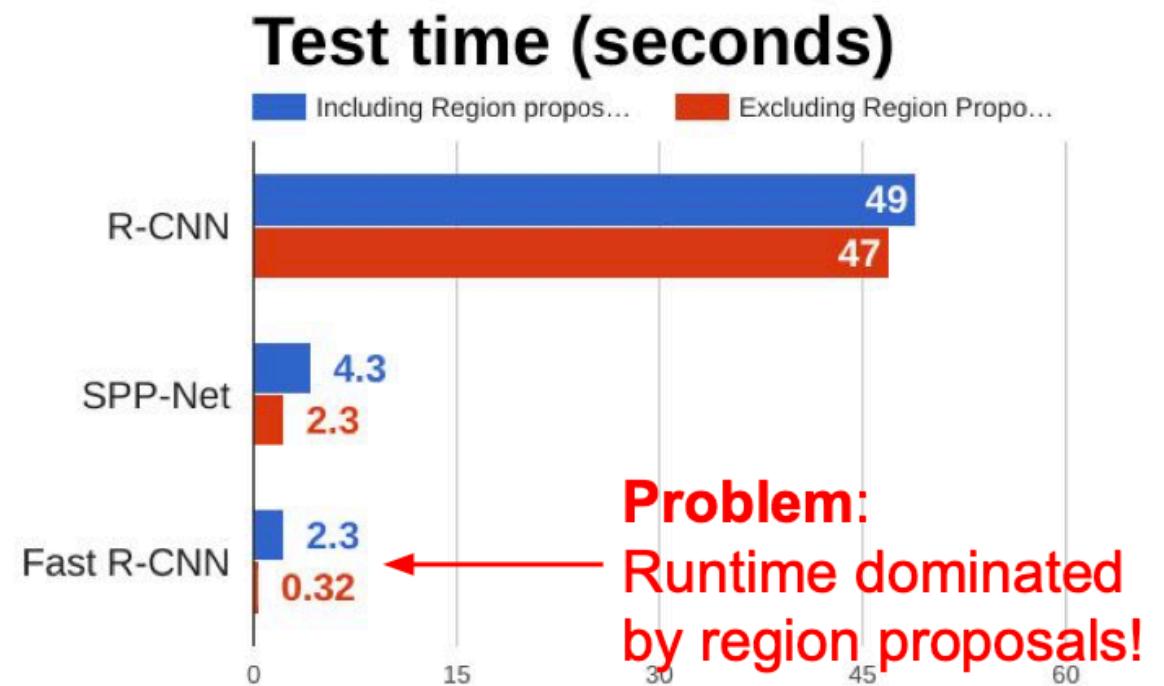
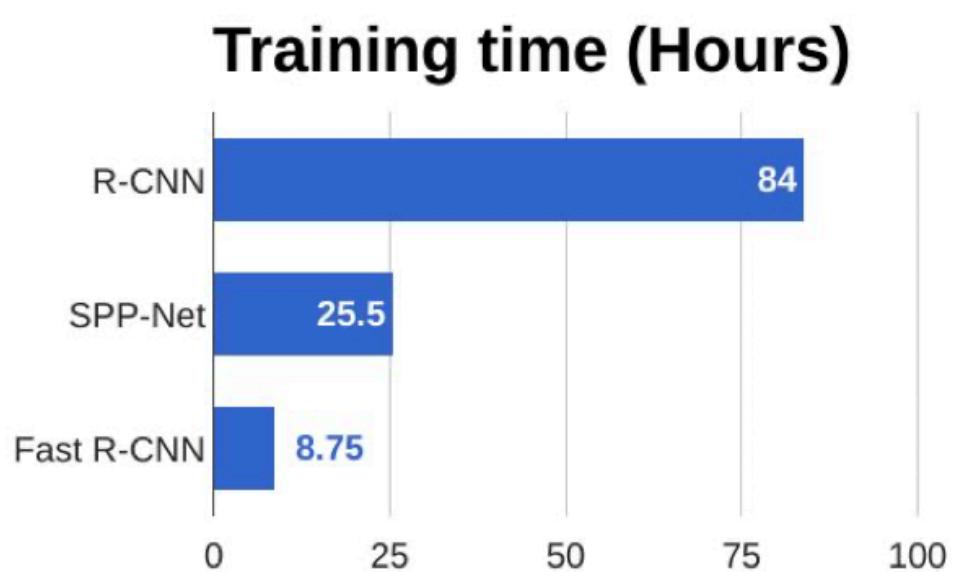


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014

Girshick, "Fast R-CNN", ICCV 2015

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

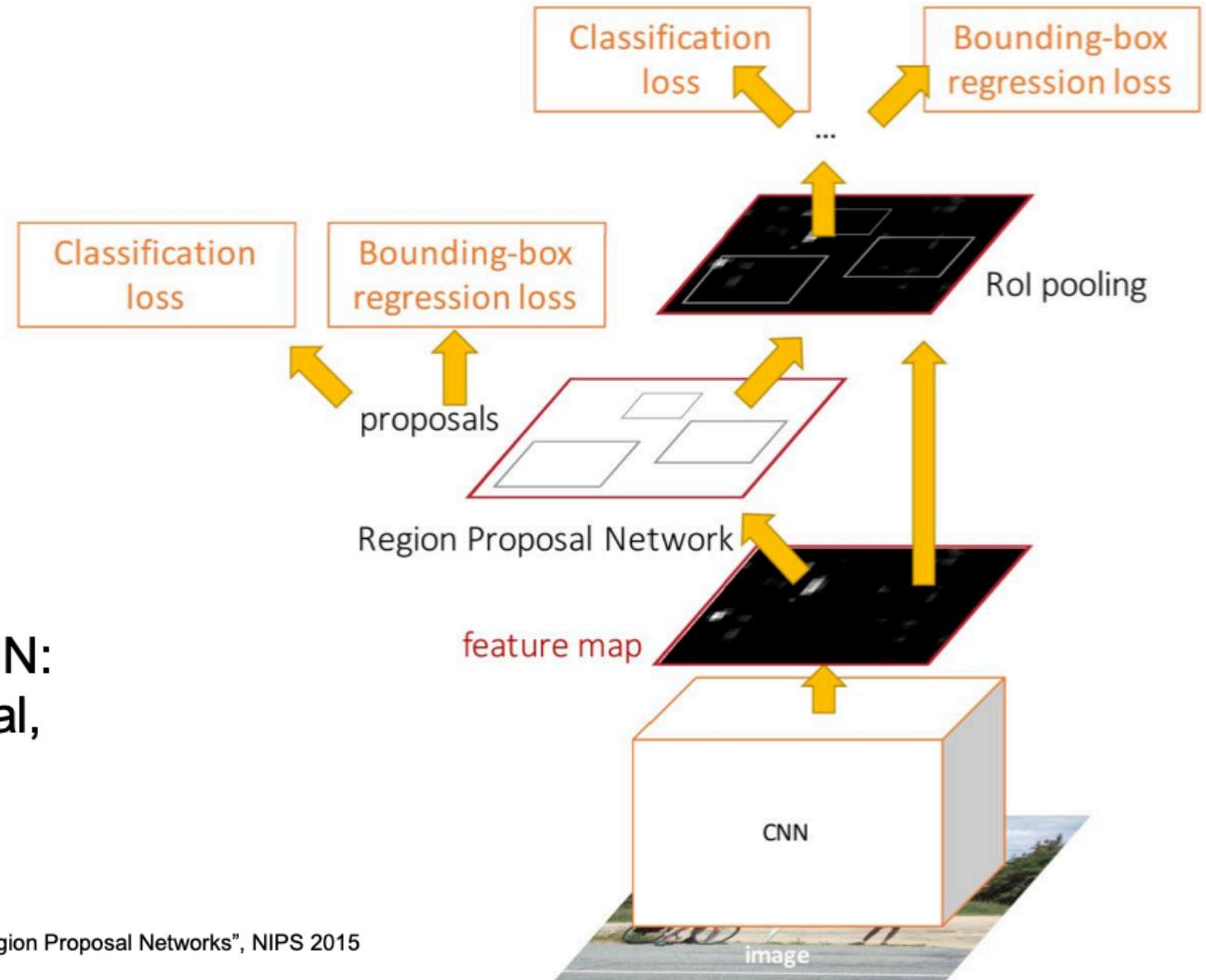
Girshick, "Fast R-CNN", ICCV 2015

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. $3 \times 640 \times 480$)

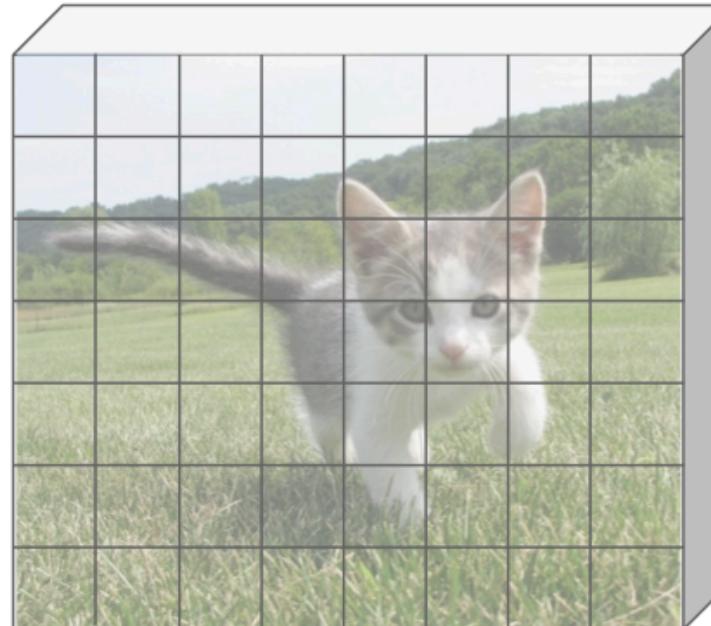
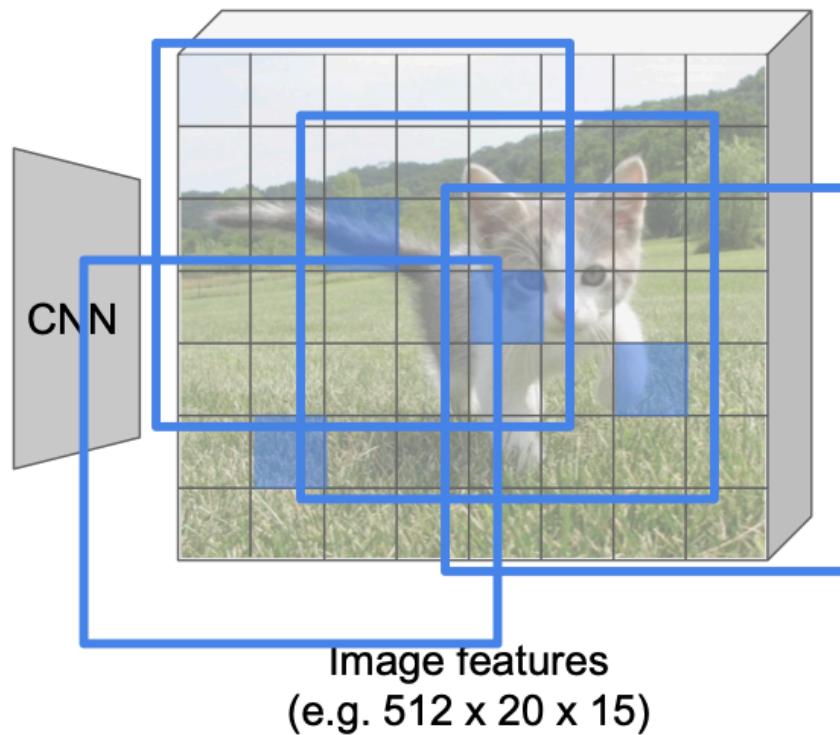


Image features
(e.g. $512 \times 20 \times 15$)

Region Proposal Network



Input Image
(e.g. $3 \times 640 \times 480$)

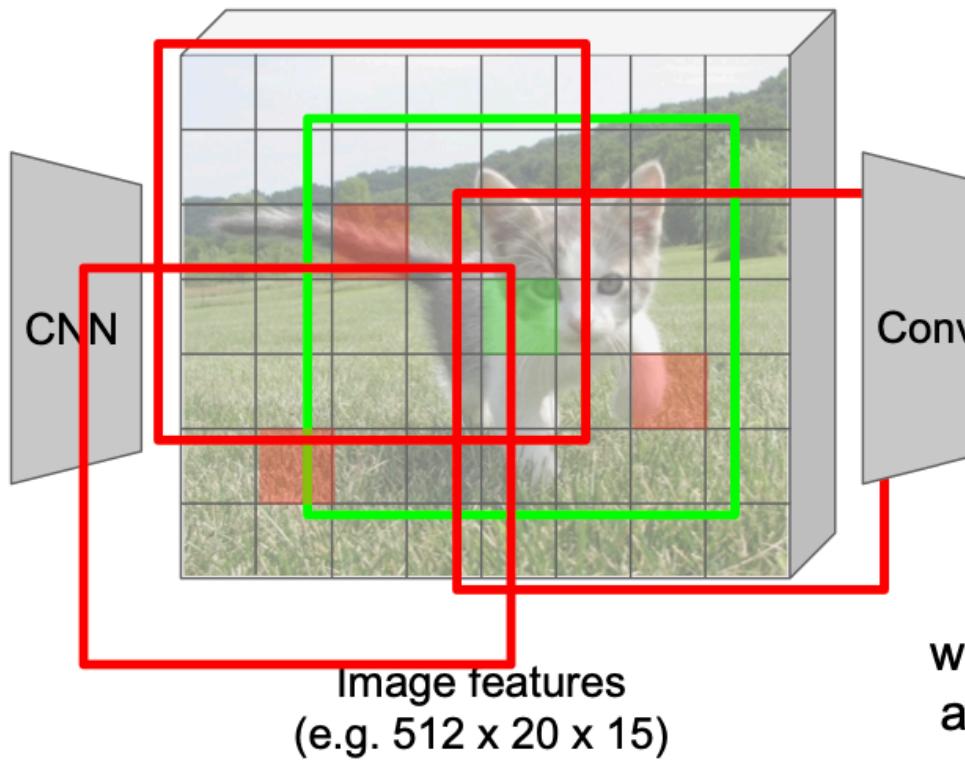


Imagine an **anchor box**
of fixed size at each
point in the feature map

Region Proposal Network



Input Image
(e.g. $3 \times 640 \times 480$)



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

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

Region Proposal Network



Input Image
(e.g. $3 \times 640 \times 480$)

CNN

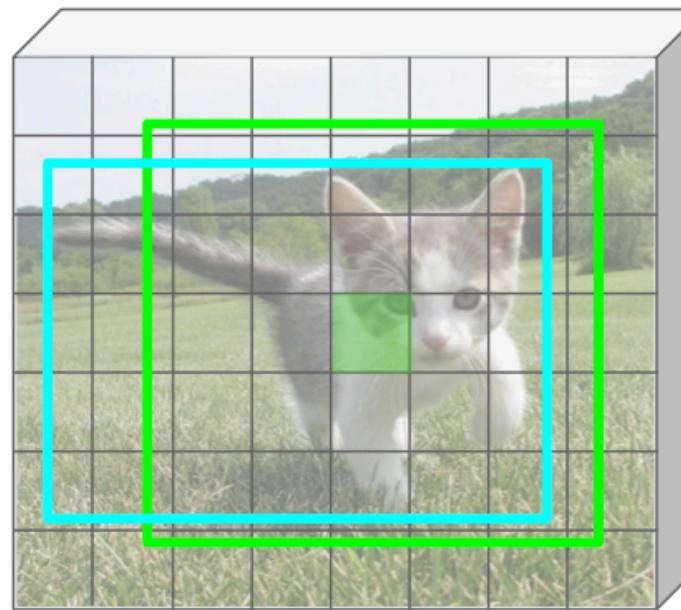


Image features
(e.g. $512 \times 20 \times 15$)

Conv

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

Anchor is an object?
 $1 \times 20 \times 15$

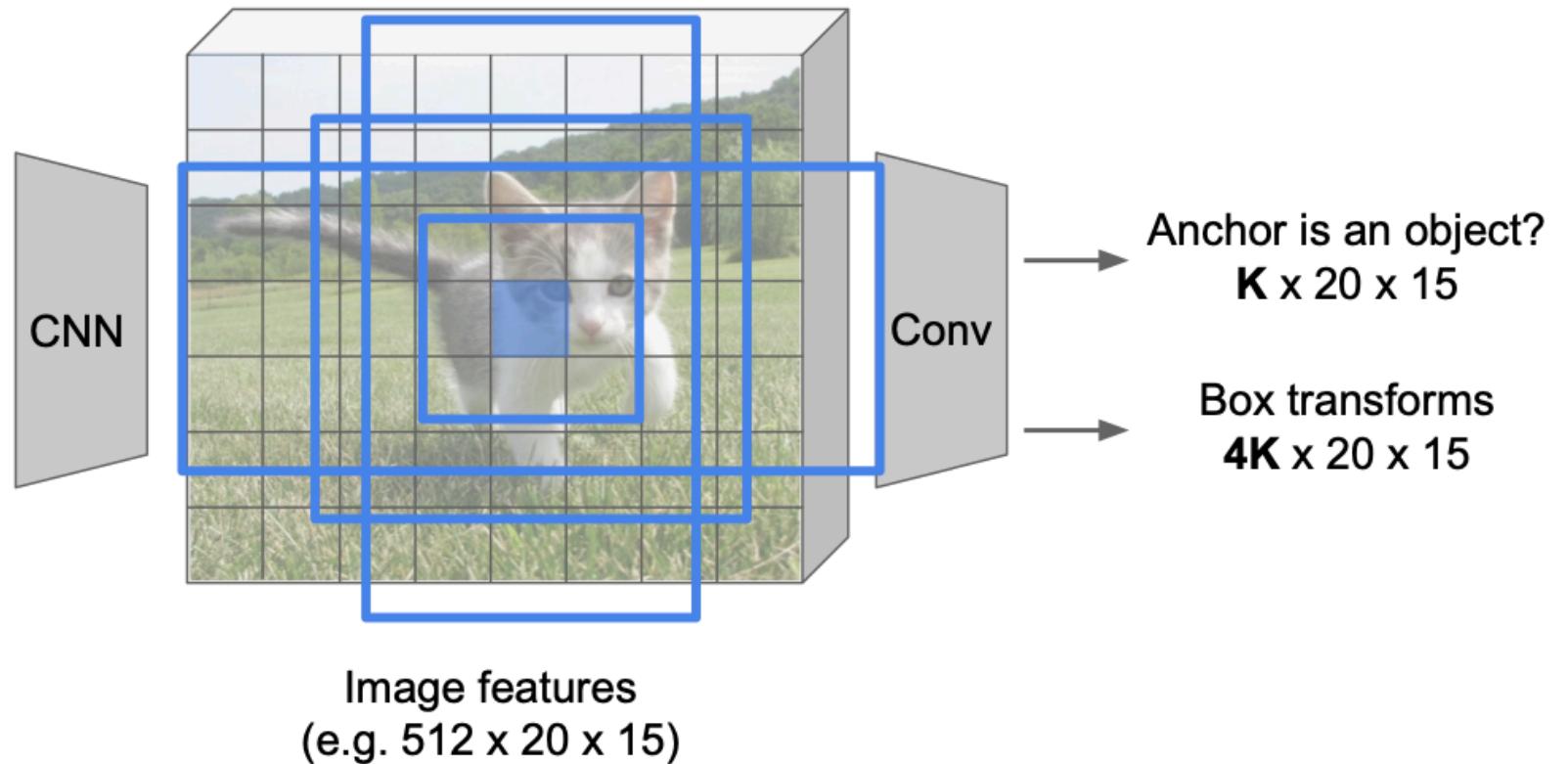
Box corrections
 $4 \times 20 \times 15$

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

Region Proposal Network



Input Image
(e.g. $3 \times 640 \times 480$)



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

Region Proposal Network



Input Image
(e.g. $3 \times 640 \times 480$)

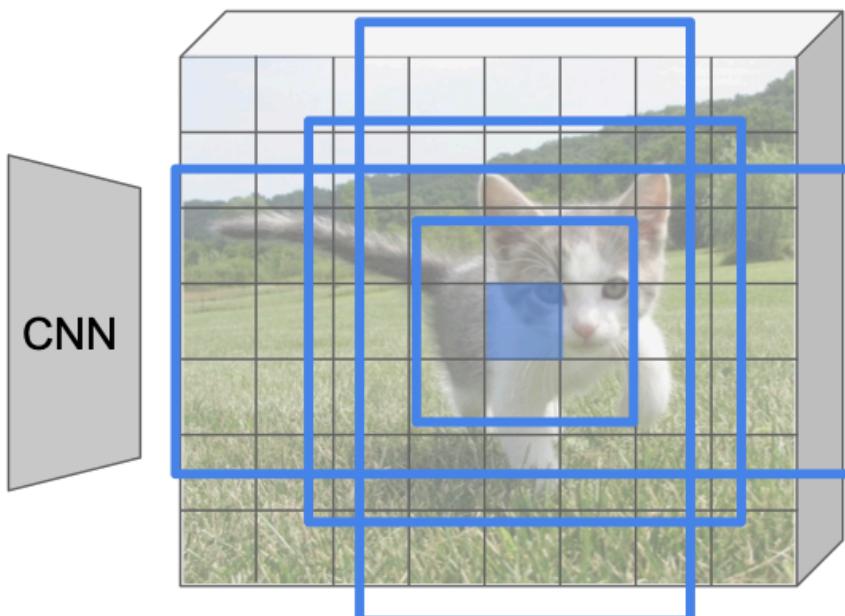


Image features
(e.g. $512 \times 20 \times 15$)

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

Anchor is an object?
 $K \times 20 \times 15$

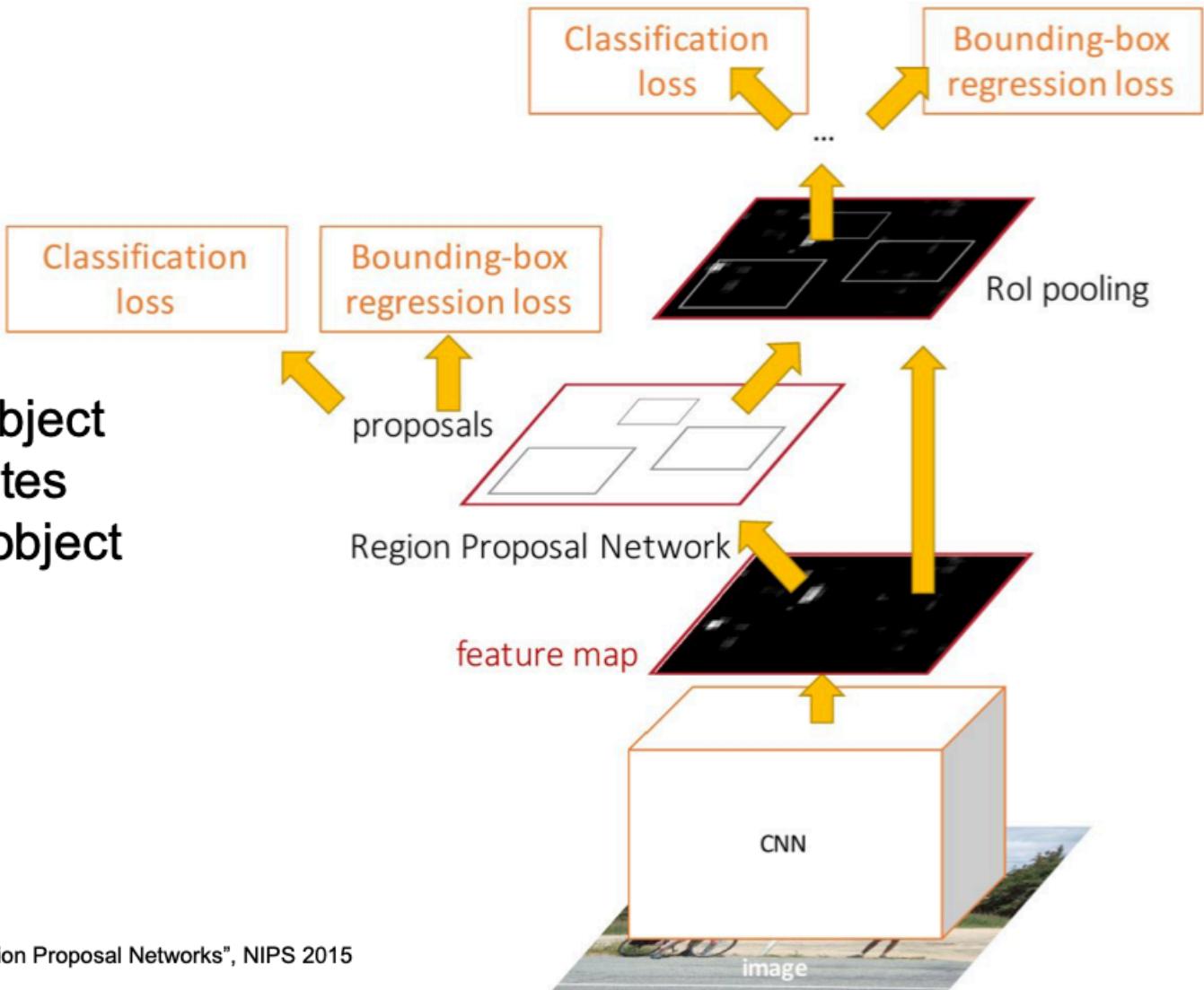
Box transforms
 $4K \times 20 \times 15$

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

Training Faster RCNN

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

Inference Time: Two-Stage Detector

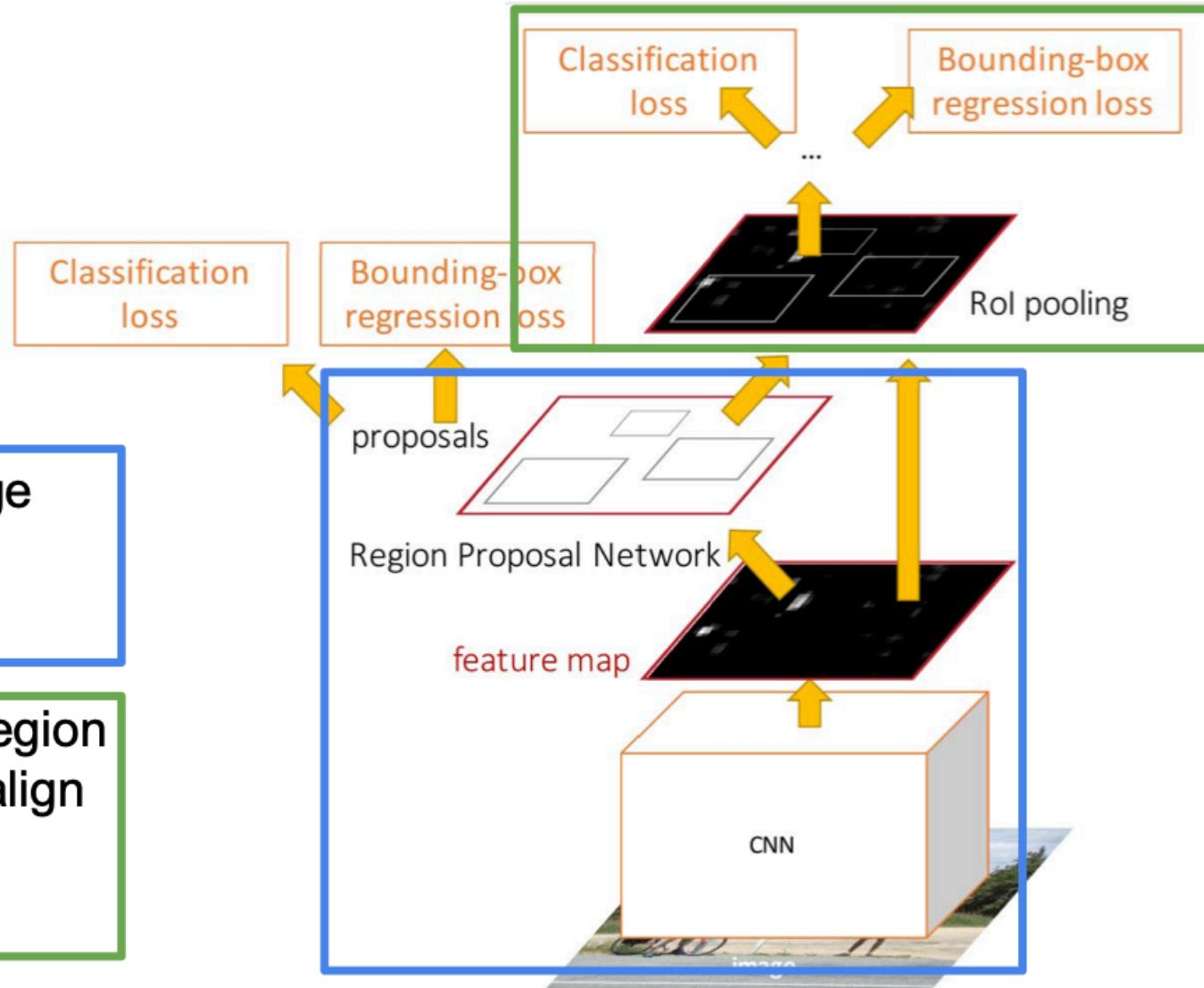
Faster R-CNN is a
Two-stage object detector

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



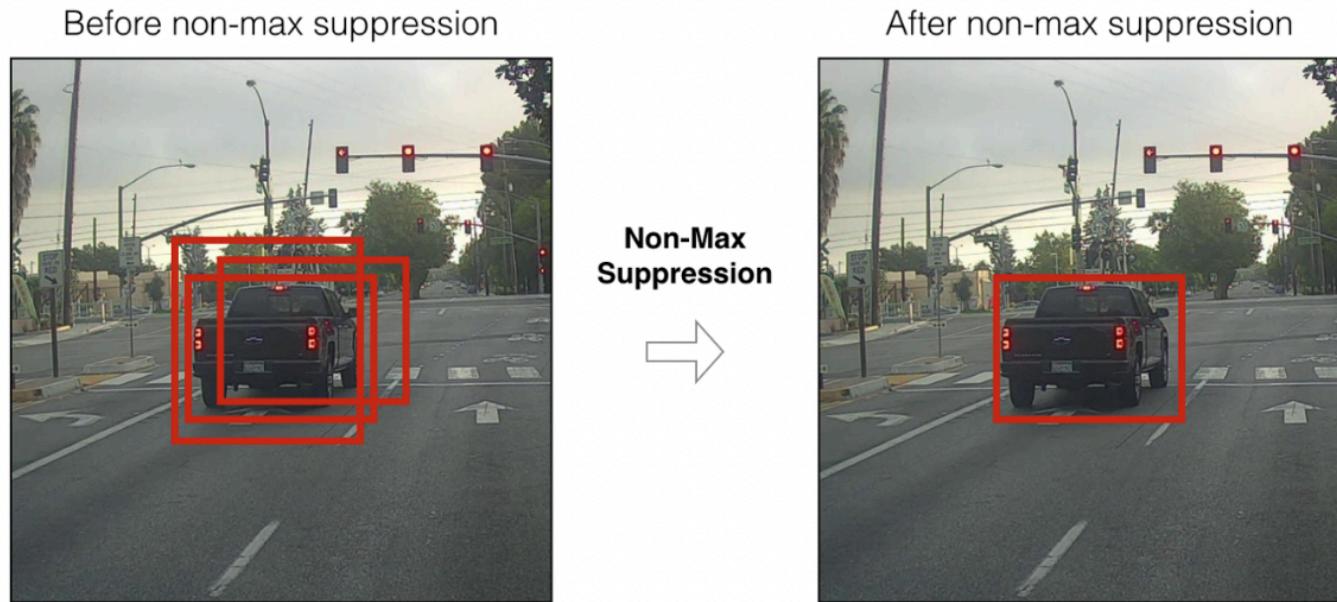
Inference Time

- First stage:
 - Use backbone to extract features
 - Use RPN to generate ~ 300 proposals
- Second stage:
 - For each proposal, predict class label and bbox refinement
 - Perform confidence thresholding to remove low-confidence bbox predictions
 - Perform non-maximal suppression (NMS) for deduplication

Non-Maximal Suppression (NMS)

Input: A list of Proposal boxes B , corresponding confidence scores S (in Faster RCNN, simply the classification score) and IoU threshold τ .

Output: A list of detected bounding boxes D .

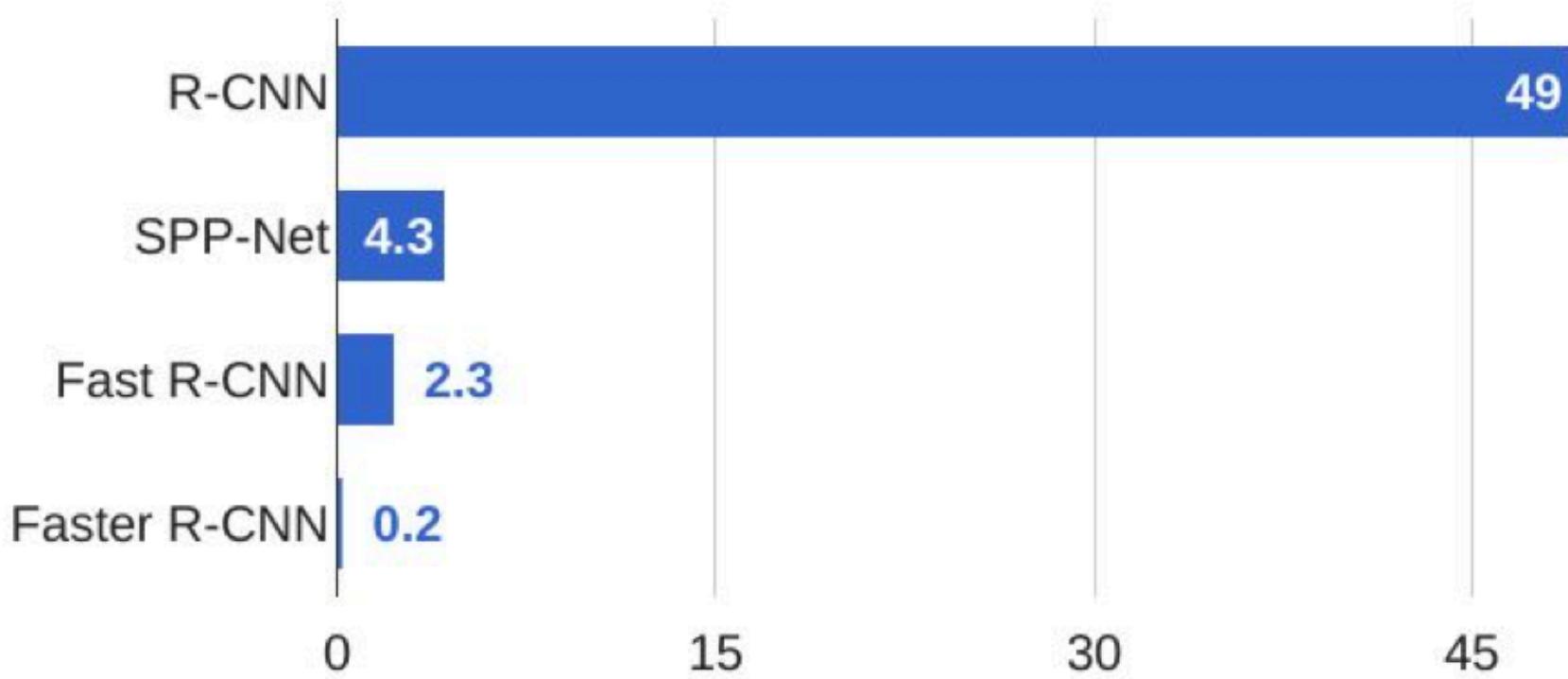


Algorithm:

- Initially D is empty
- Select the proposal with highest confidence score, remove it from B and add it to the final detection list D.
- Now compare this proposal with all the proposals — calculate the IoU of this proposal with every other proposal. If the IOU is greater than the threshold τ , remove that proposal from B.
- Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
- Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have a IoU higher than τ .
- This process is repeated until there are no more proposals left in B.

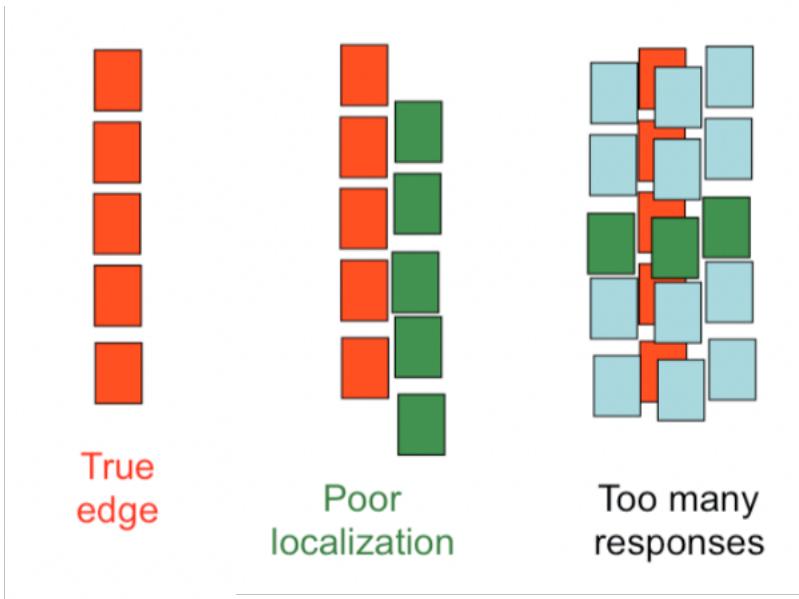
Speed Comparison

R-CNN Test-Time Speed



How to Evaluate Detection?

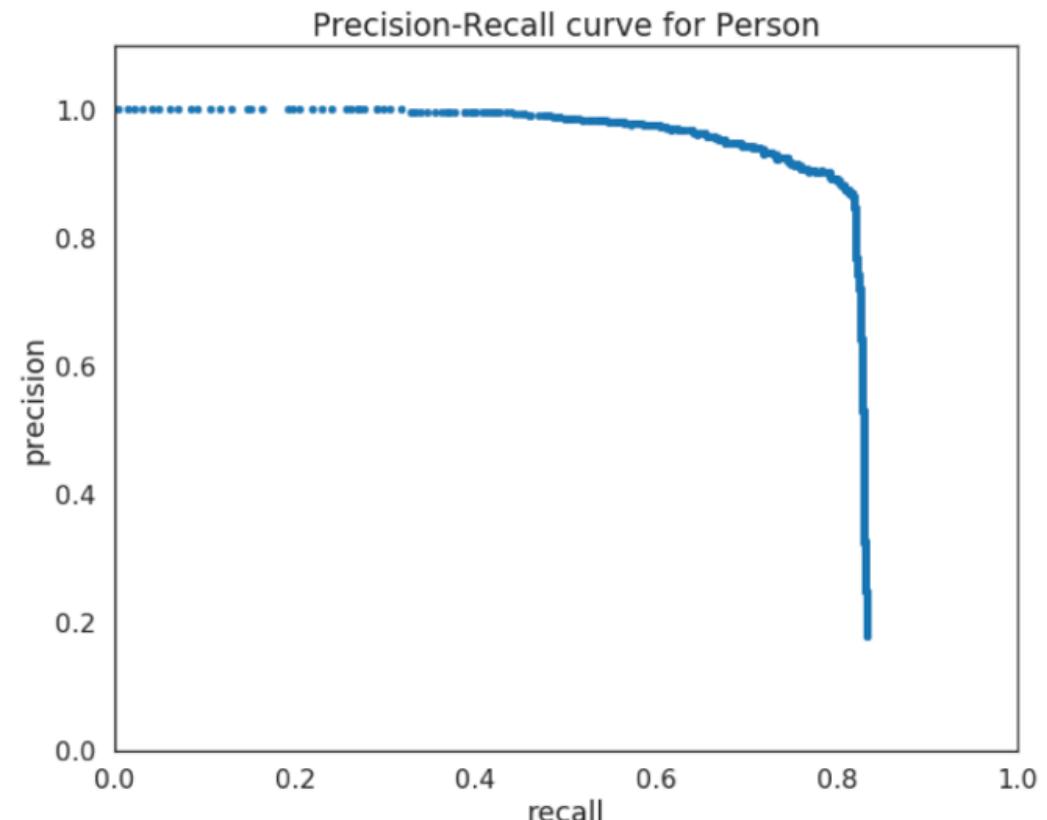
Recall from optimal edge detection, Lecture 02



- Good accuracy (precision): minimize false positive
- Good localization (precision): maximize IoU
- Single response constraint (precision): minimize redundant responses
- Good coverage (recall): make sure all edges are detected.

Evaluation Metric: AP (Average Precision)

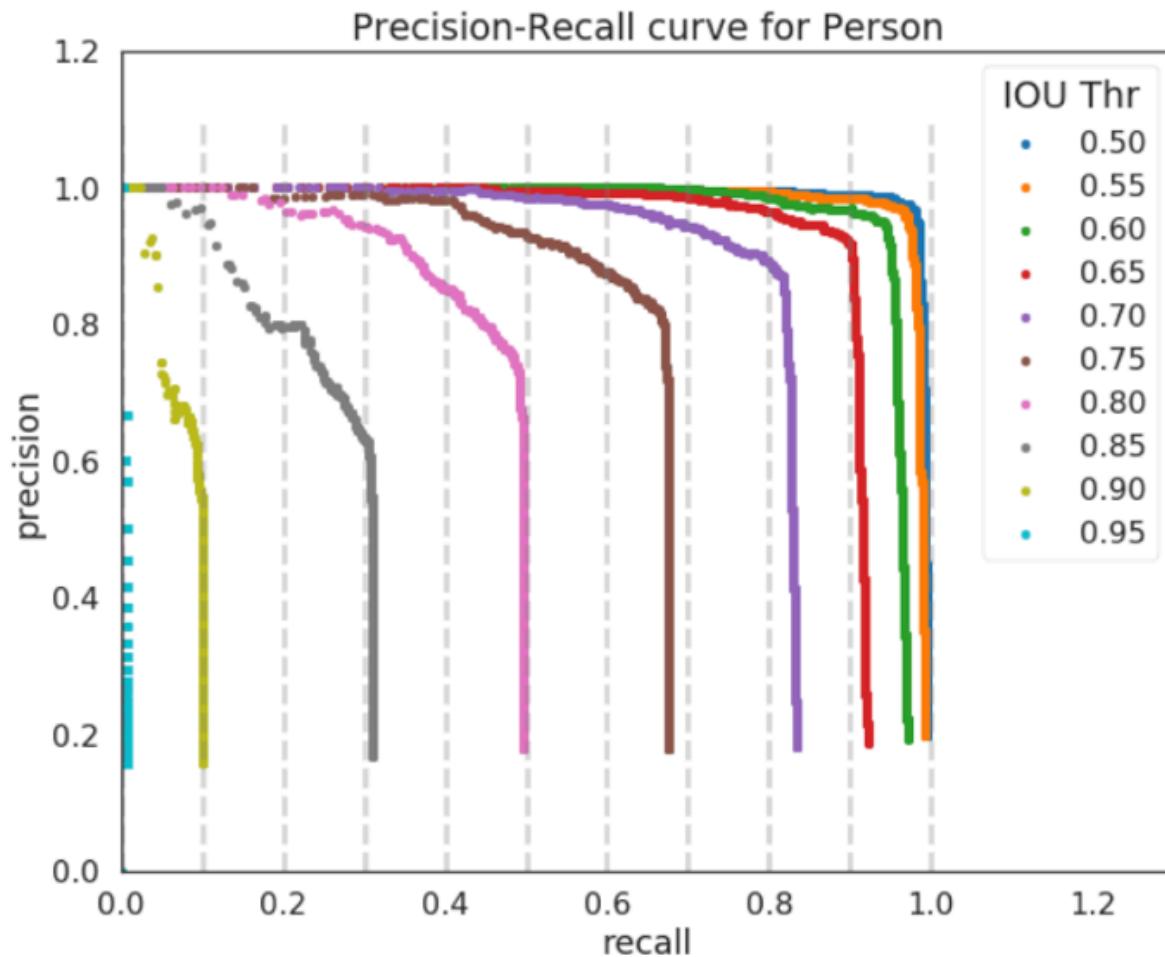
- Per category rank the output bounding boxes according to the confidence (classification score) in a descending order.
- Select top n outputs and compute recall.
- Precision: the ratio of bboxes that satisfy $\text{IoU} > x\%$ threshold
- Compute the area under precision-recall curve (approximate by 11 points).



$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$

$\text{Recall}_i = [0, 0.1, 0.2, \dots, 1.0]$.

Evaluation Metric: AP at Different IoU Thres.



Precision-Recall curves calculated at various IoU thresholds, according to the COCO challenge. Dashed lines correspond to equally spaced recall values where the AP is calculated.

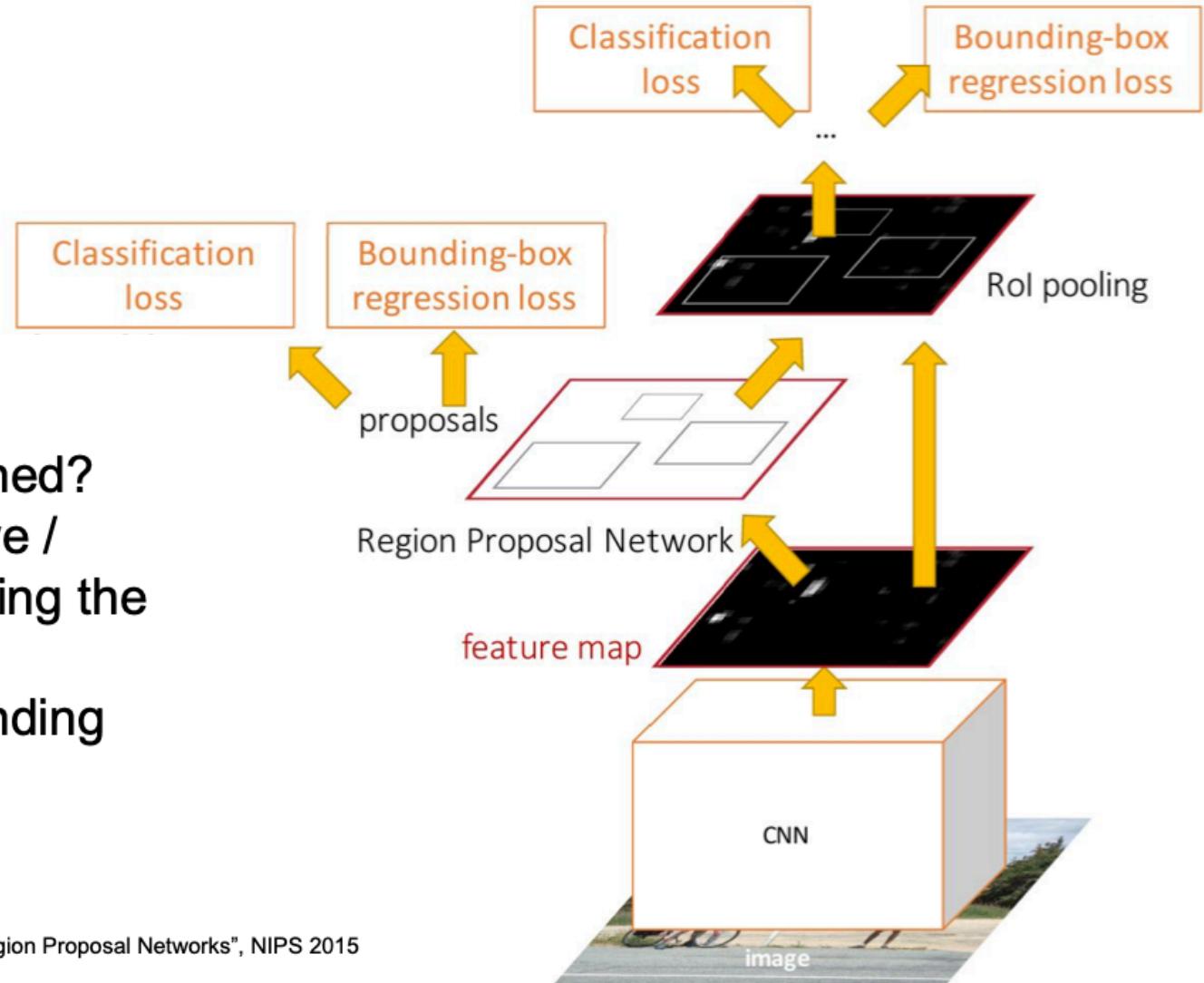
Evaluation Metric: mAP

- mAP is the mean of AP across different categories and/or IoU thresholds. Sometimes m is ignored.
- Examples when evaluating on MS COCO:
 - AP
 - AP₅₀

Faster RCNN

Glossing over many details:

- How are anchors determined?
- How do we sample positive / negative samples for training the RPN?
- How to parameterize bounding box regression?



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

Two-Stage Detector

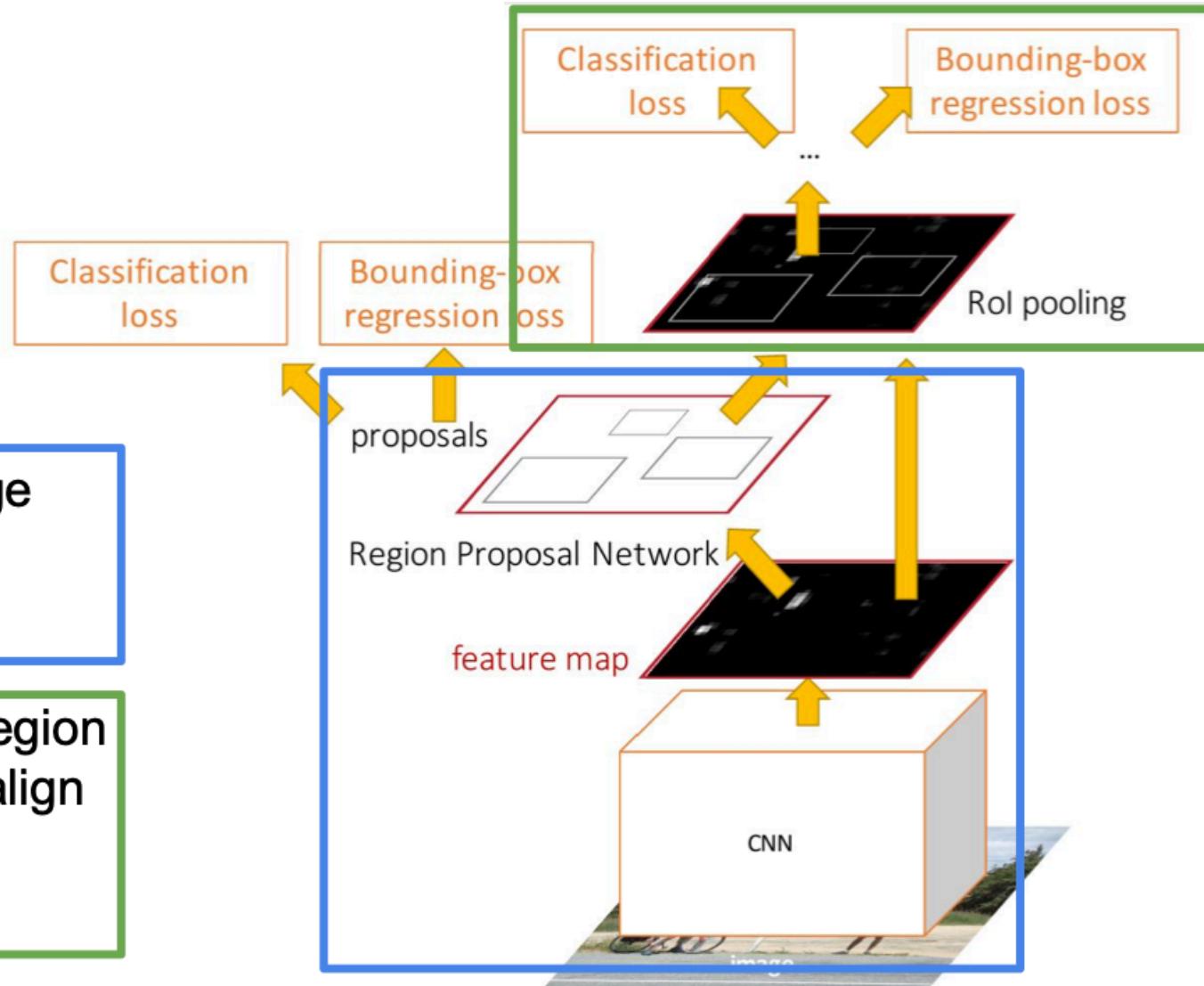
Faster R-CNN is a
Two-stage object detector

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 Detector

Do we really need the second stage?

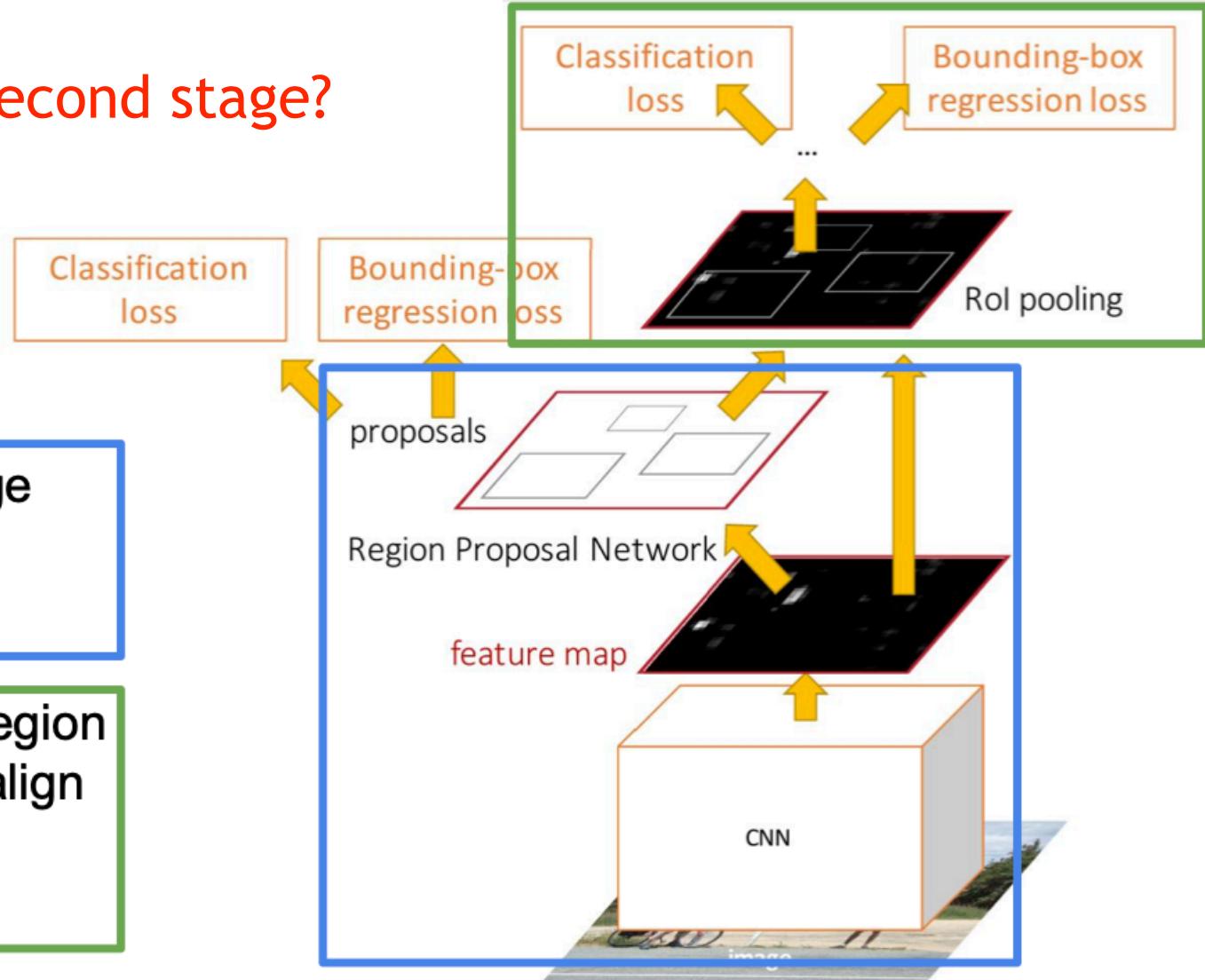
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

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

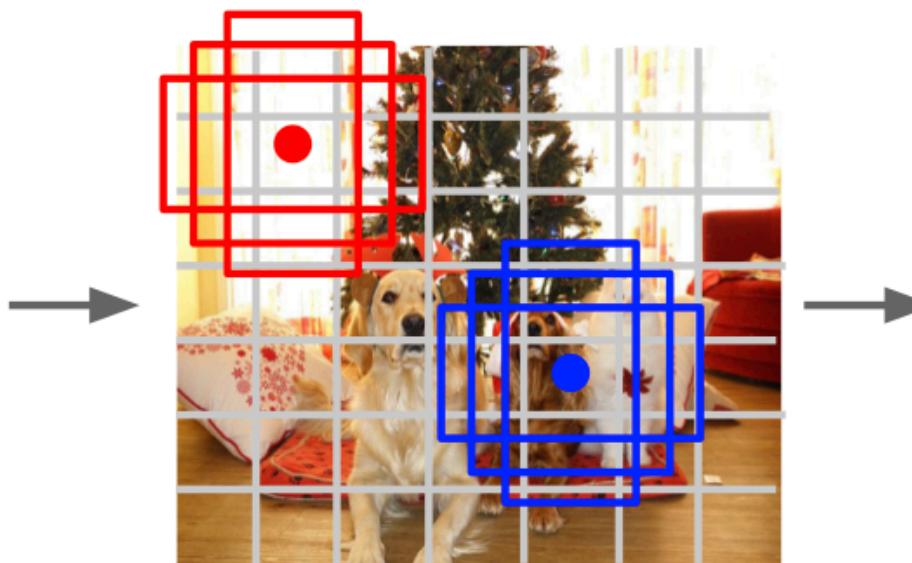


Single-Stage Detectors: YOLO/SSD/RetinaNet



Input image
 $3 \times H \times W$

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

- Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers:
(dx , dy , dh , dw , confidence)
 - Predict scores for each of C classes (including background as a class)
 - Looks a lot like RPN, but category-specific!

Output:
 $7 \times 7 \times (5 * B + C)$

Object Detection: Lots of Variables

Backbone Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

“Meta-Architecture”
Two-stage: Faster R-CNN
Single-stage: YOLO / SSD
Hybrid: R-FCN

Image Size
Region Proposals
...

Takeaways
Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Bigger / Deeper backbones work better

Huang et al, “Speed/accuracy trade-offs for modern convolutional object detectors”, CVPR 2017

Zou et al, “Object Detection in 20 Years: A Survey”, arXiv 2019

R-FCN: Dai et al, “R-FCN: Object Detection via Region-based Fully Convolutional Networks”, NIPS 2016

Inception-V2: Ioffe and Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, ICML 2015

Inception V3: Szegedy et al, “Rethinking the Inception Architecture for Computer Vision”, arXiv 2016

Inception ResNet: Szegedy et al, “Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning”, arXiv 2016

MobileNet: Howard et al, “Efficient Convolutional Neural Networks for Mobile Vision Applications”, arXiv 2017

Instance Segmentation

Some slides are borrowed from Stanford CS231N.

Computer Vision Tasks

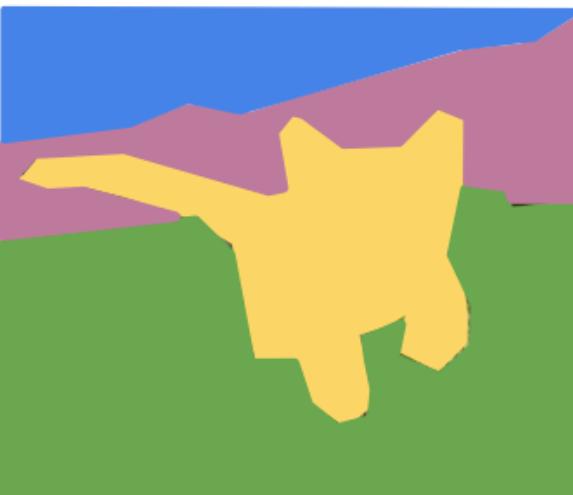
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation

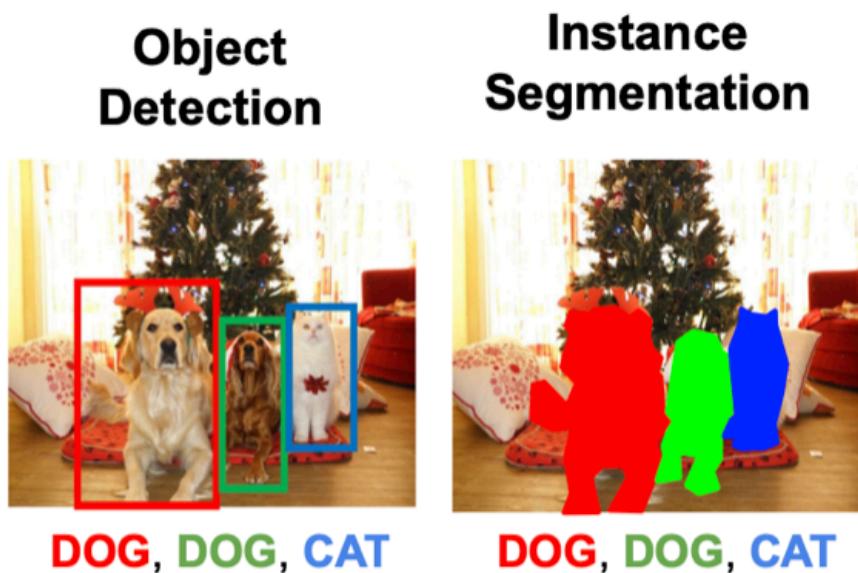


DOG, DOG, CAT

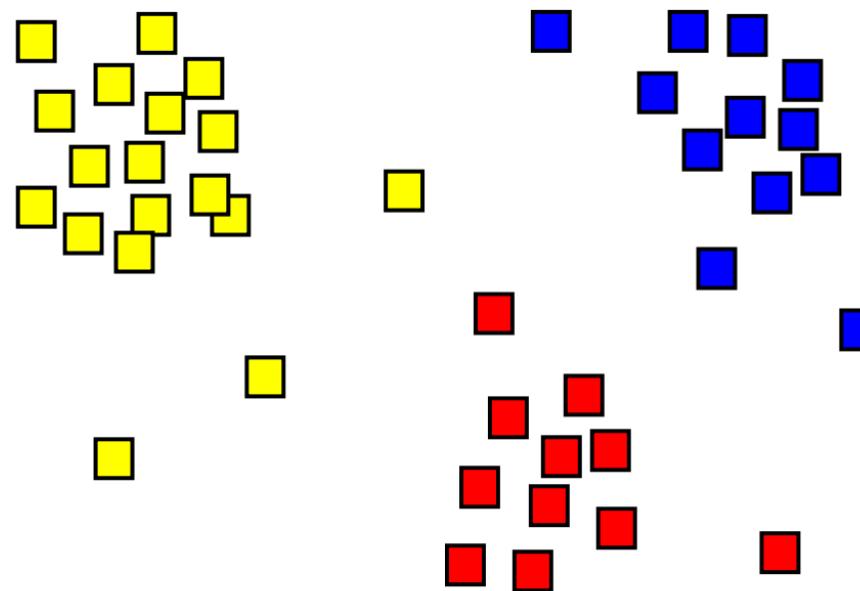
[This image](#) is CC0 public domain

Different Approaches for Instance Segmentation

- **Top-down approach:** object detection and then further find a binary mask inside the bounding box



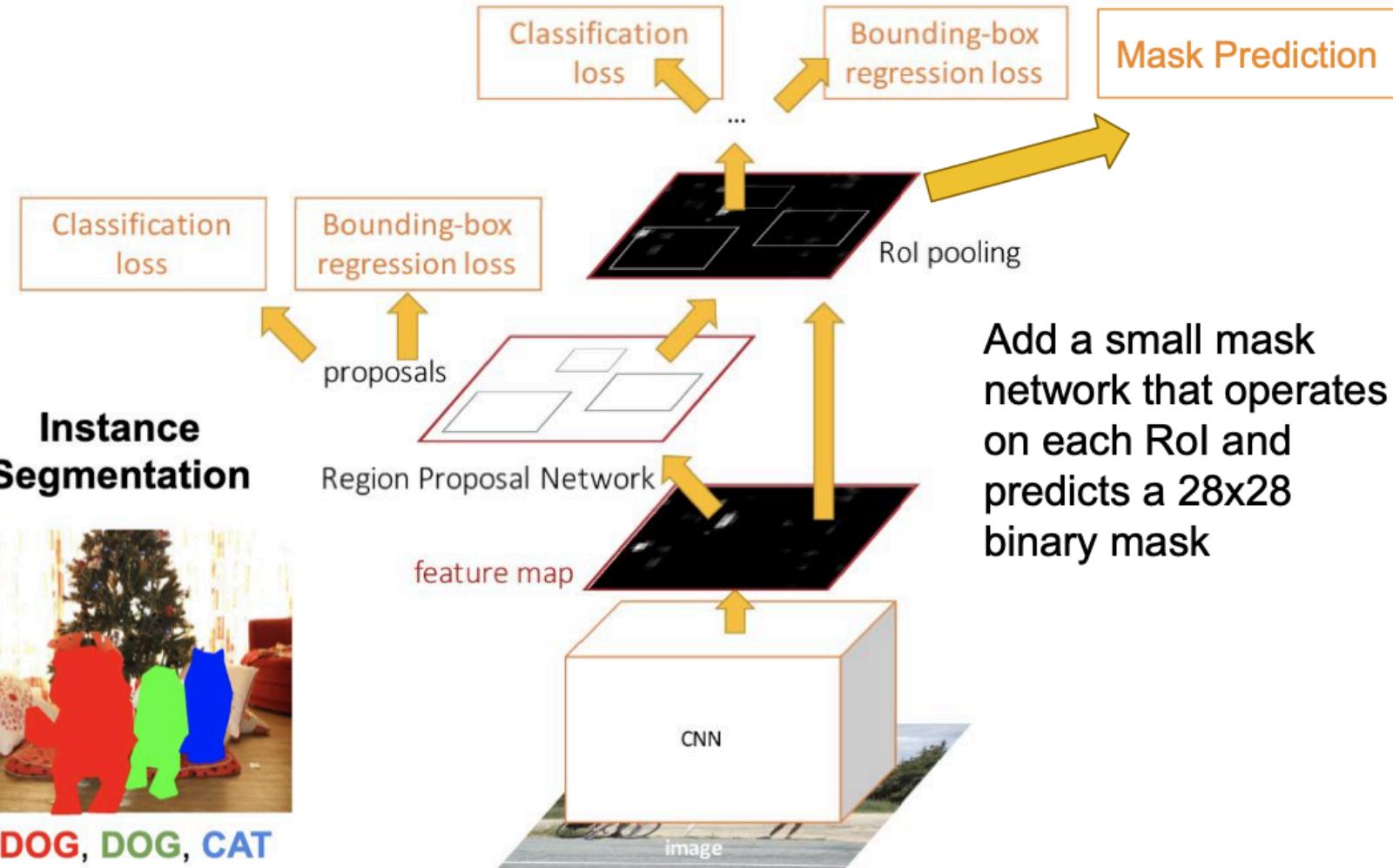
- **Bottom-up approach:** grouping and then classification
 - Grouping: group together similar data points and represents them with a single token



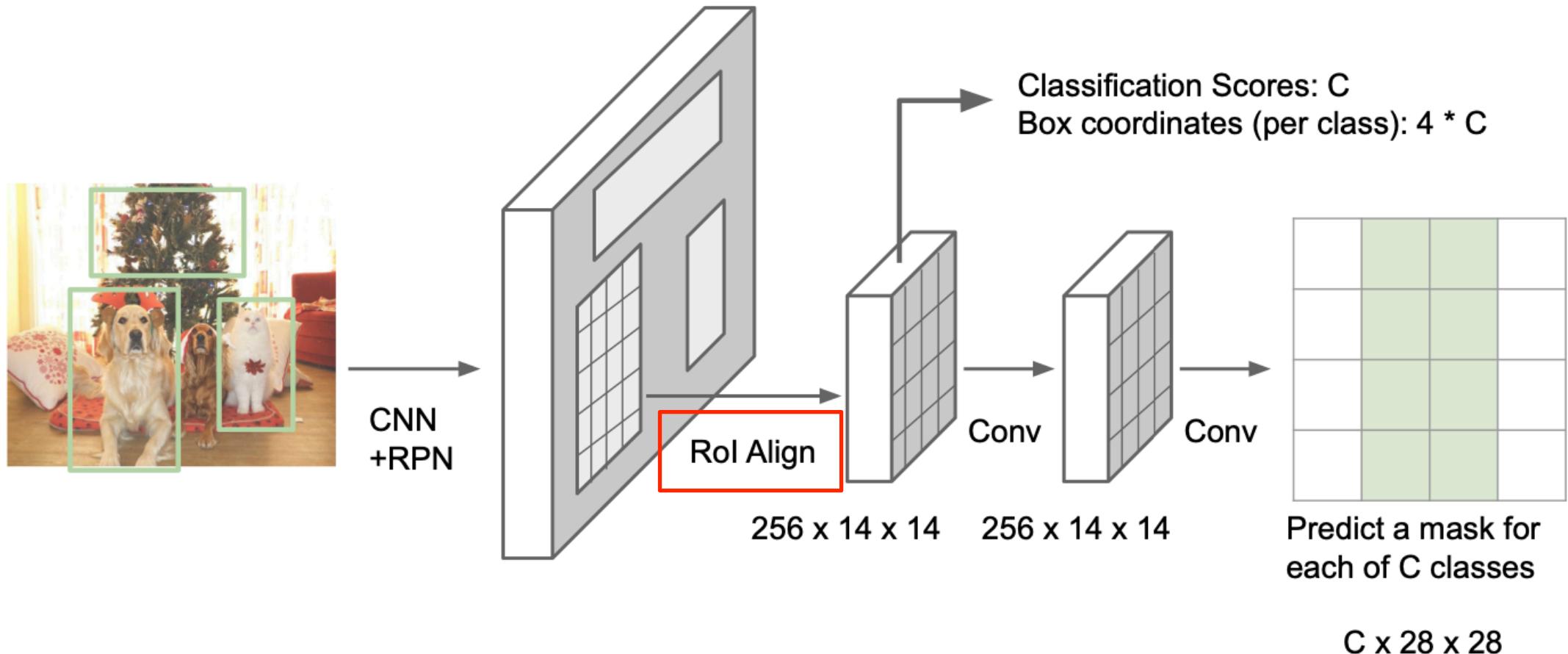
Top-Down Approach: Mask R-CNN



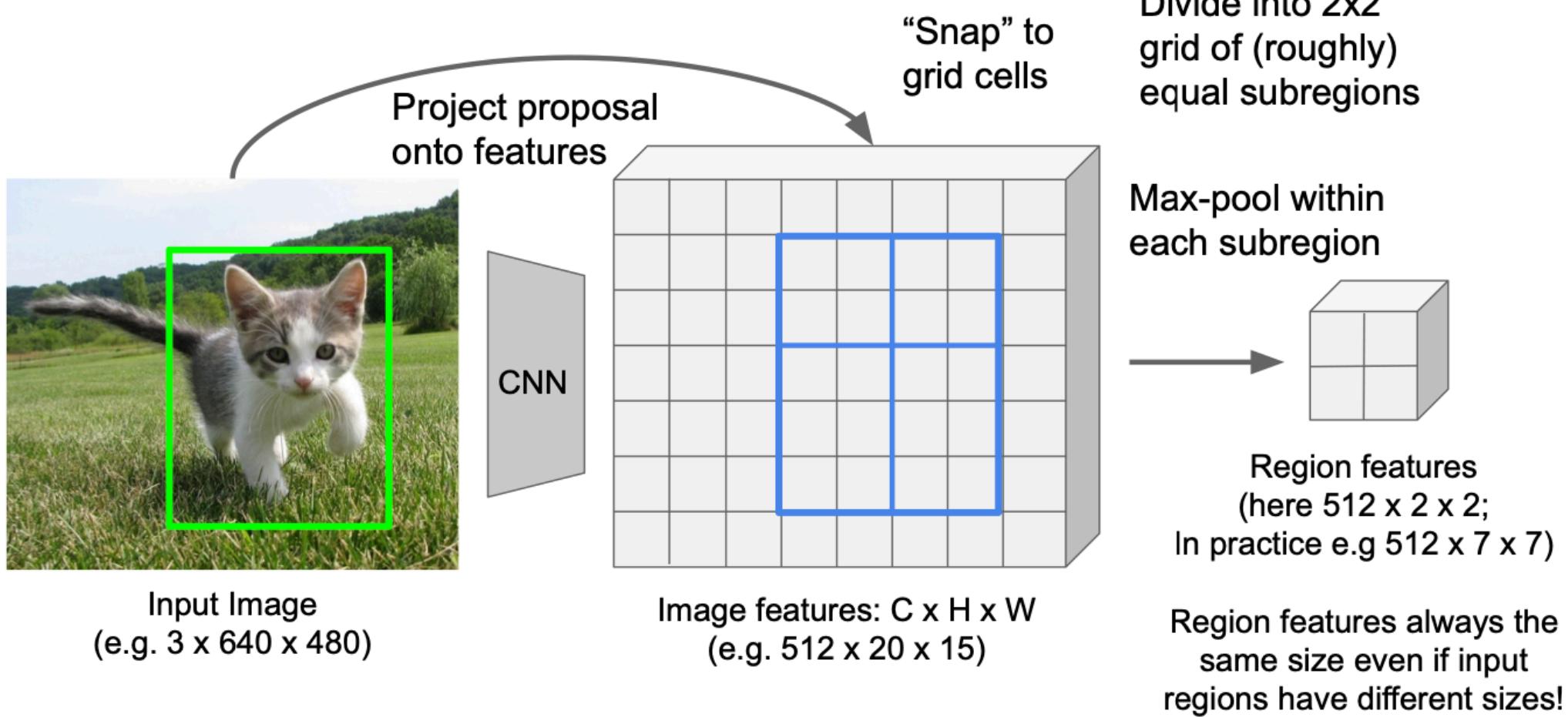
He et al, "Mask R-CNN", ICCV 2017



Mask R-CNN

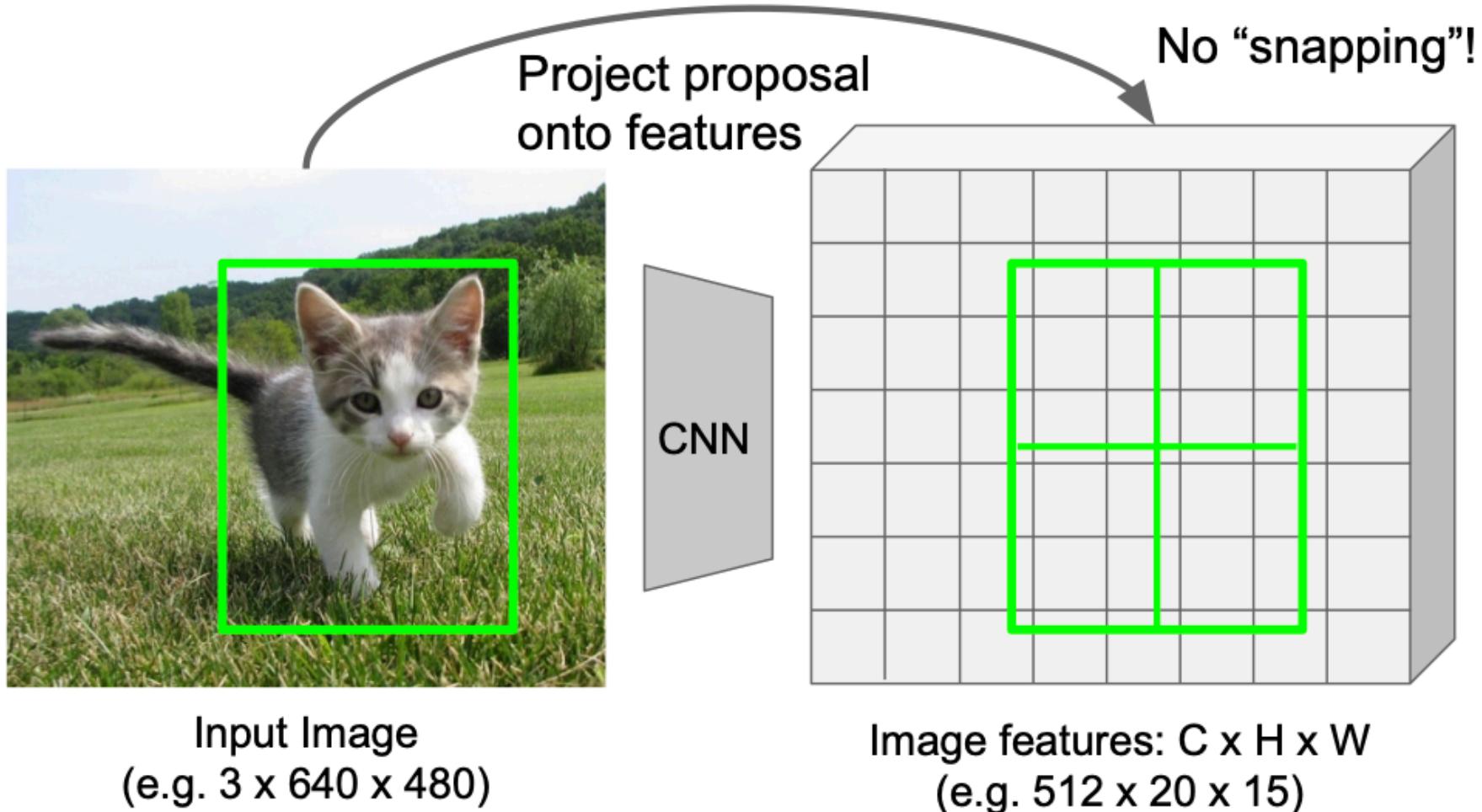


Problems with RoI Pool

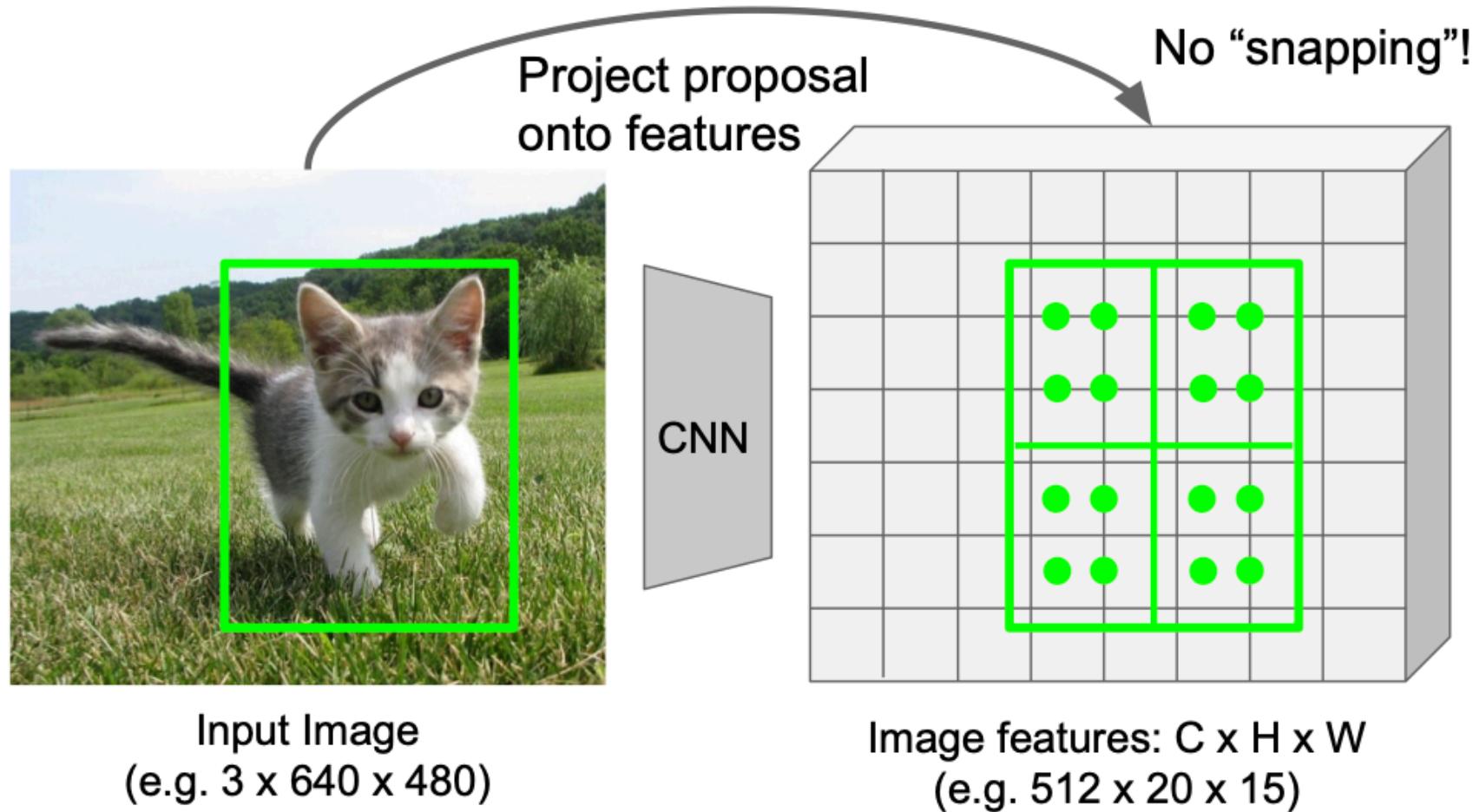


Girshick, “Fast R-CNN”, ICCV 2015.

RoI Align

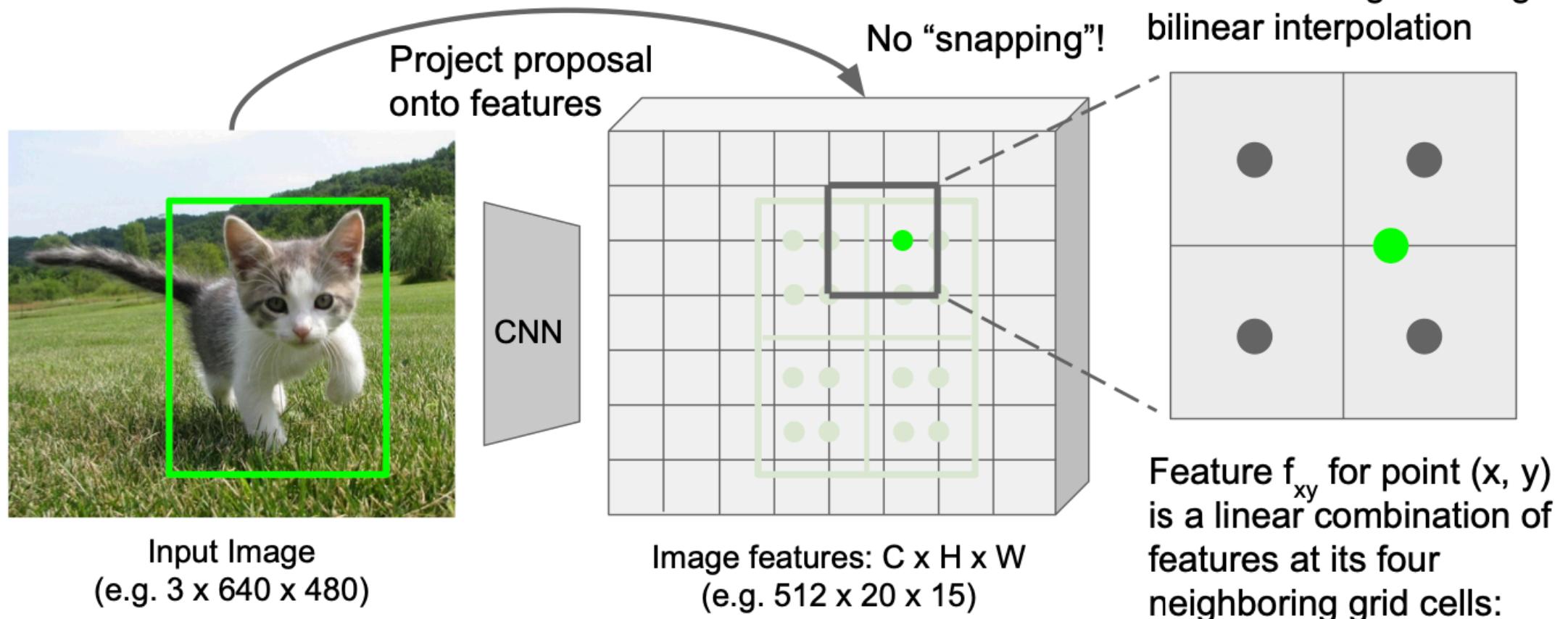


RoI Align

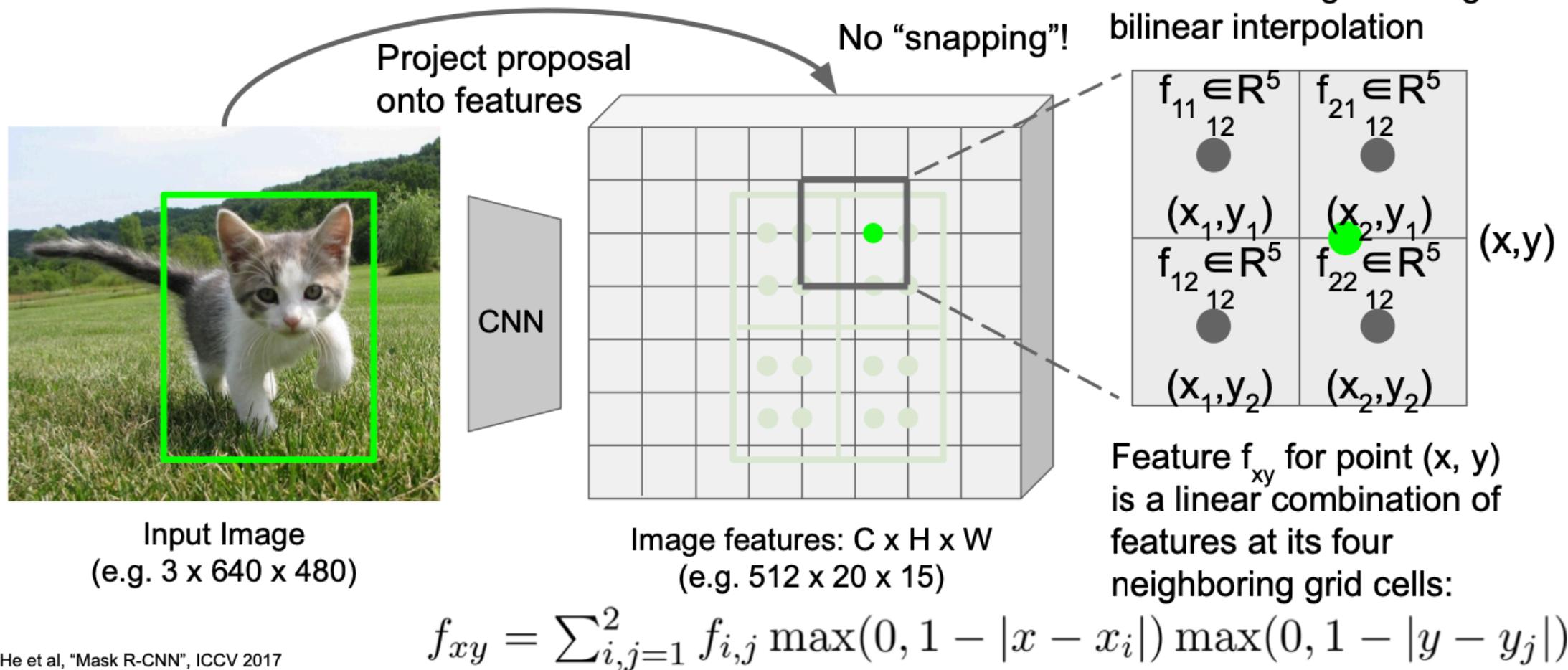


Sample at regular points
in each subregion using
bilinear interpolation

RoI Align

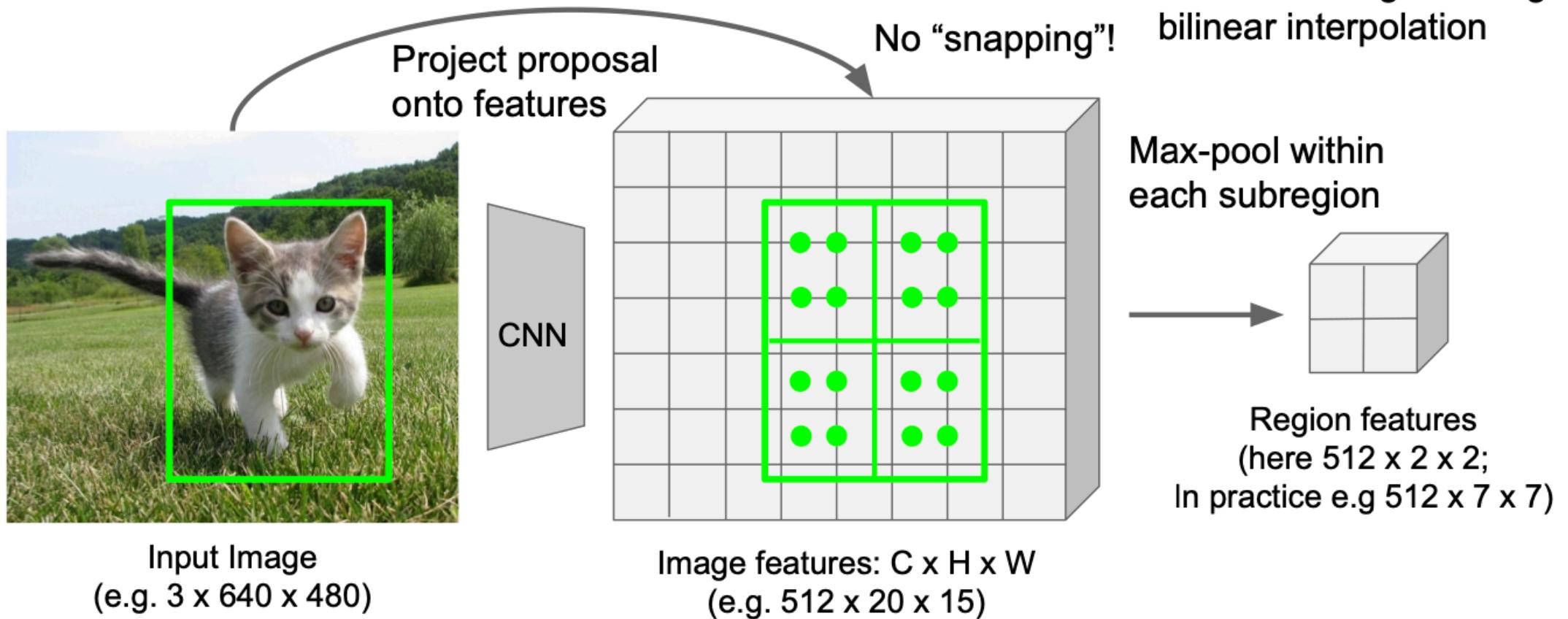


RoI Align



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

RoI Align



Ablation Study on RoI Align

| | AP | AP ₅₀ | AP ₇₅ | AP ^{bb} | AP ^{bb} ₅₀ | AP ^{bb} ₇₅ |
|-----------------|-------------|------------------|------------------|------------------|--------------------------------|--------------------------------|
| <i>RoIPool</i> | 23.6 | 46.5 | 21.6 | 28.2 | 52.7 | 26.9 |
| <i>RoIAlign</i> | 30.9 | 51.8 | 32.1 | 34.0 | 55.3 | 36.4 |
| | +7.3 | + 5.3 | +10.5 | +5.8 | +2.6 | +9.5 |

(d) **RoIAlign** (ResNet-50-**C5**, *stride* 32): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in big accuracy gaps.

Class-Specific vs. Class-Agnostic Masks

- Our default instantiation predicts class-specific masks, *i.e.*, one $m \times m$ mask per class.
- Mask R-CNN with class-agnostic masks (*i.e.*, predicting a single $m \times m$ output regardless of class) is nearly as effective.

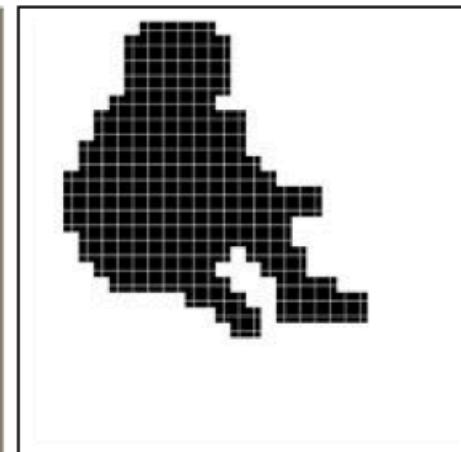
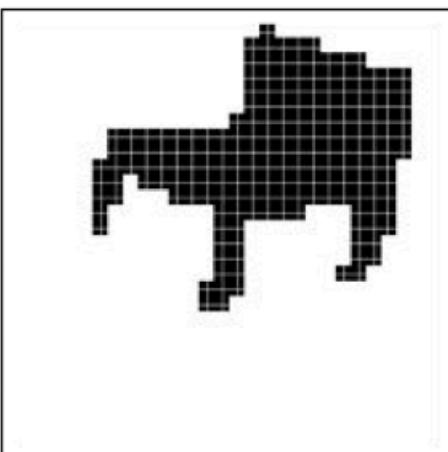
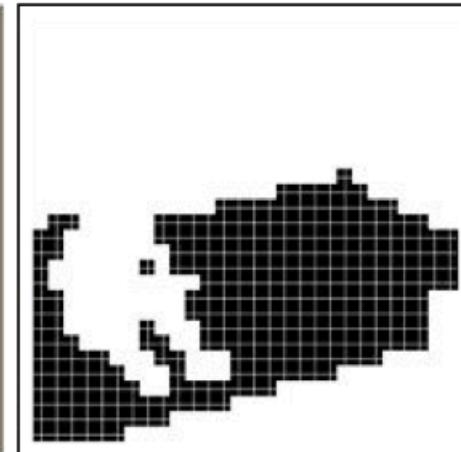
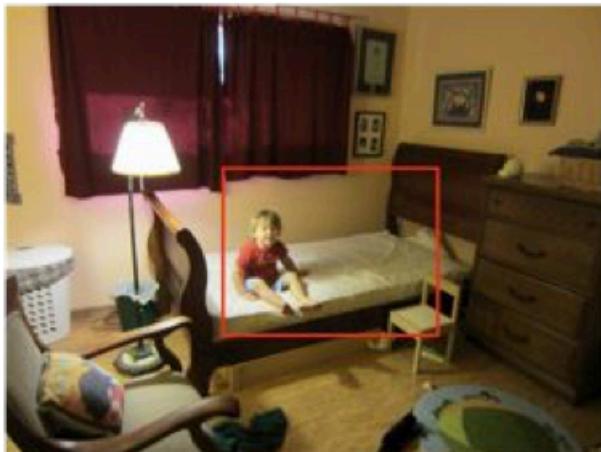
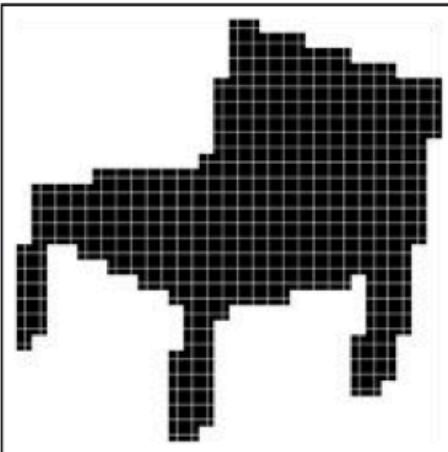
Multinomial vs. Independent Masks

- Decouples mask and class prediction
- Generate a mask for each class without competition among classes (by a per-pixel *sigmoid* and a *binary loss*).

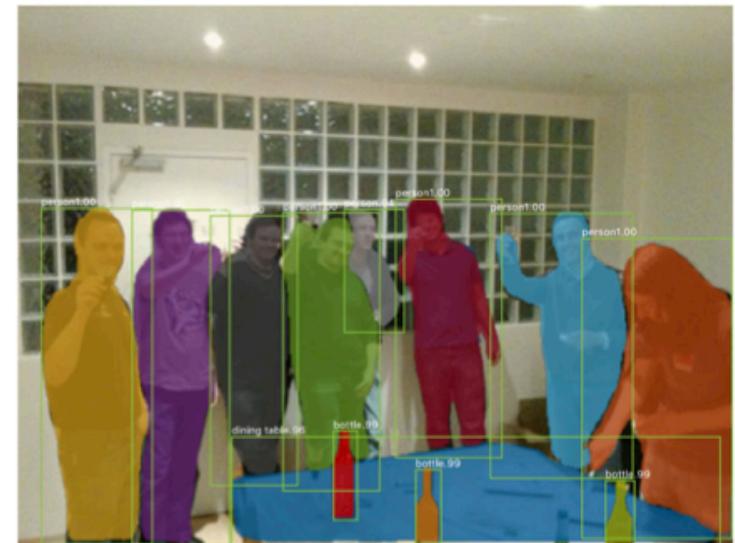
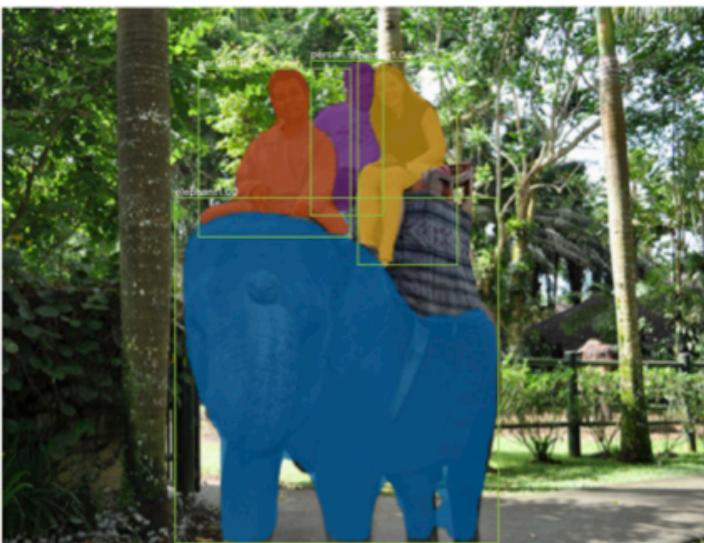
| | AP | AP ₅₀ | AP ₇₅ |
|----------------|-------------|------------------|------------------|
| <i>softmax</i> | 24.8 | 44.1 | 25.1 |
| <i>sigmoid</i> | 30.3 | 51.2 | 31.5 |
| | +5.5 | +7.1 | +6.4 |

(b) **Multinomial vs. Independent Masks**
(ResNet-50-C4): *Decoupling* via per-class binary masks (*sigmoid*) gives large gains over multinomial masks (*softmax*).

Mask RCNN: Example Mask Training Target



Result Visualization



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

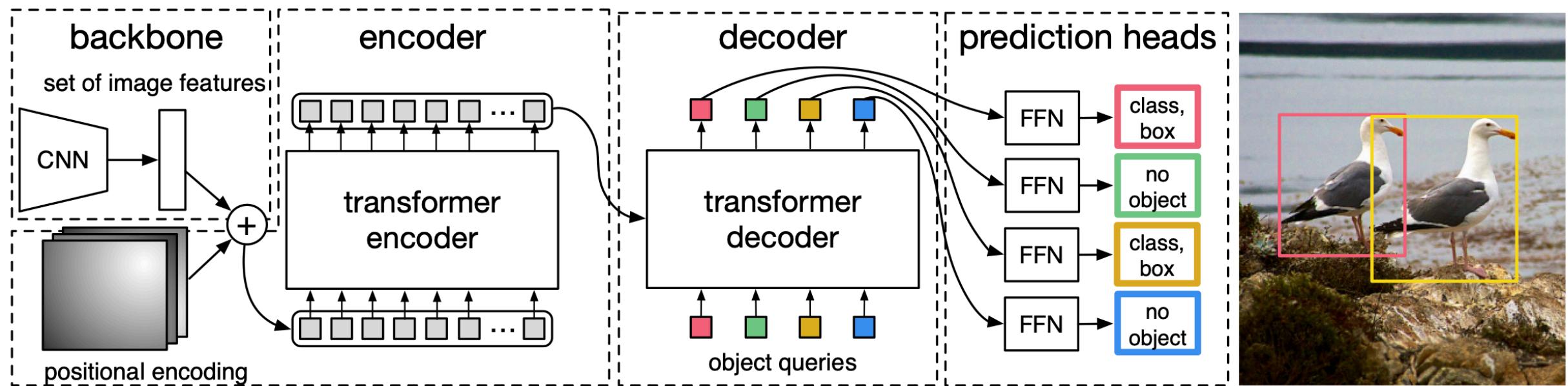
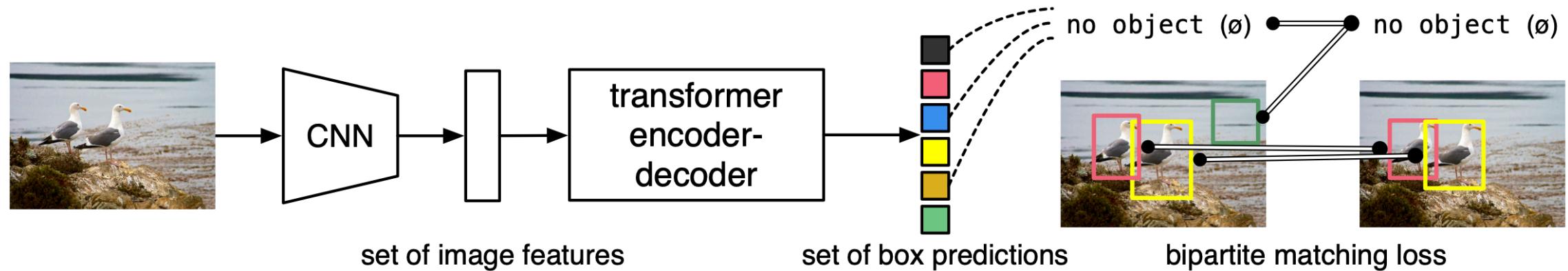
Human Pose Visualization



Open Source Framework

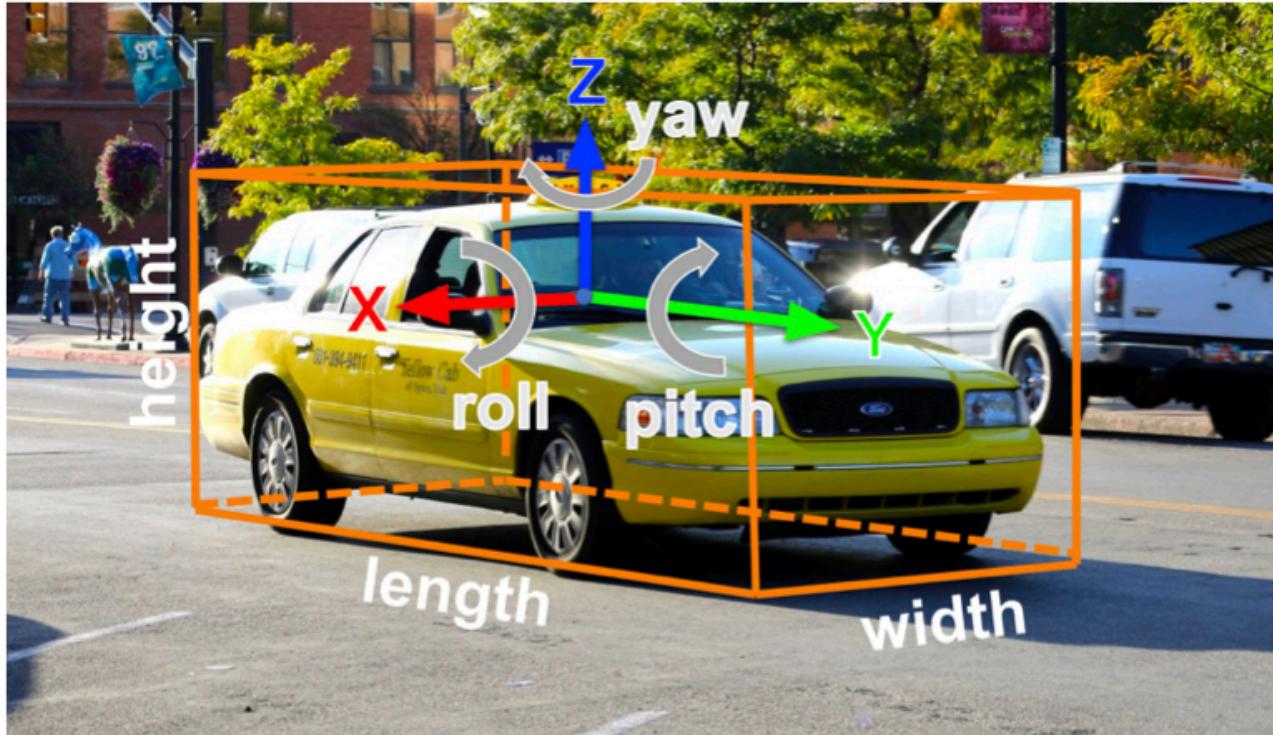
- Lots of good implementations on GitHub!
- TensorFlow Detection API:
 - https://github.com/tensorflow/models/tree/master/research/object_detection
 - Faster RCNN, SSD, RFCN, Mask R-CNN, ...
- Detectron2 (PyTorch) :
 - <https://github.com/facebookresearch/detectron2>
 - Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN, ... Finetune on your own dataset with pre-trained models

End-to-End Object Detection with Transformers (DETR)



3D Object Detection and Instance Segmentation

3D Object Detection



2D Object Detection:
2D bounding box
 (x, y, w, h)

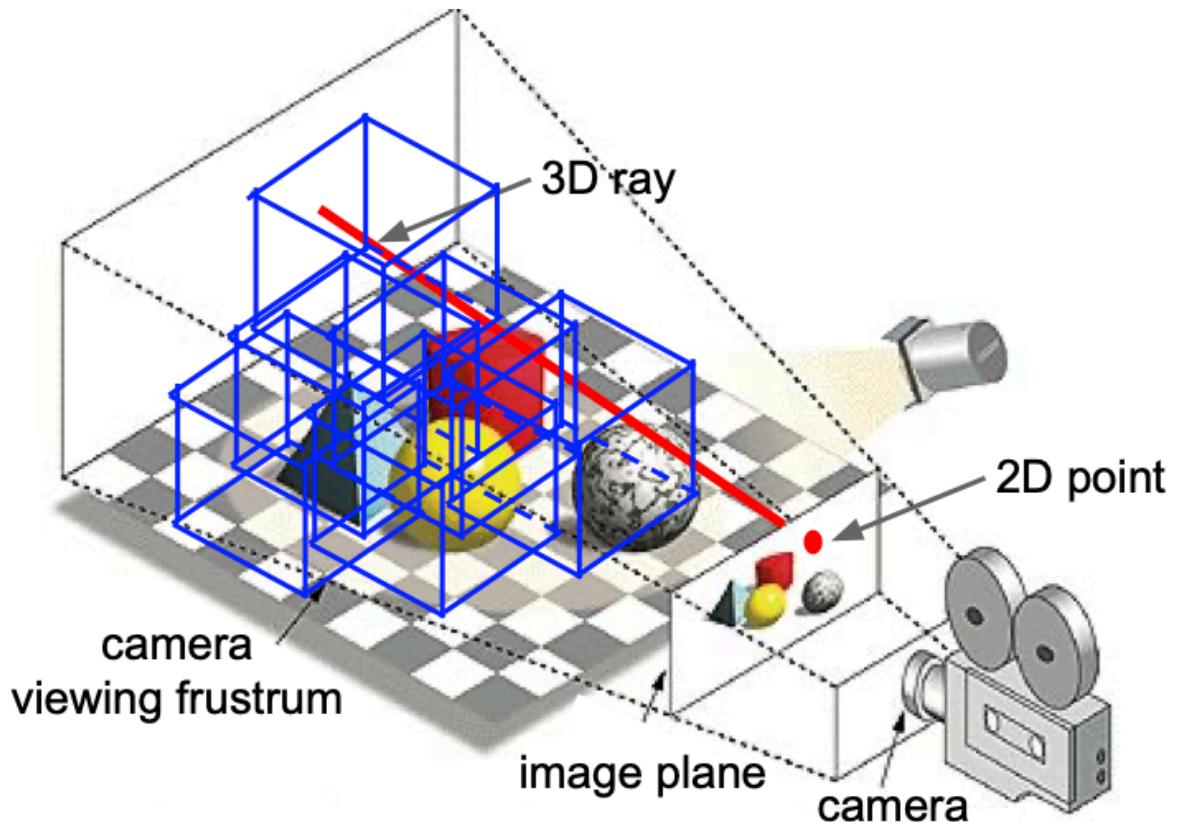
3D Object Detection:
3D oriented bounding box
 $(x, y, z, w, h, l, r, p, y)$

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

[This image](#) is CC0 public domain

3D Object Detection



A point on the image plane corresponds to a **ray** in the 3D space

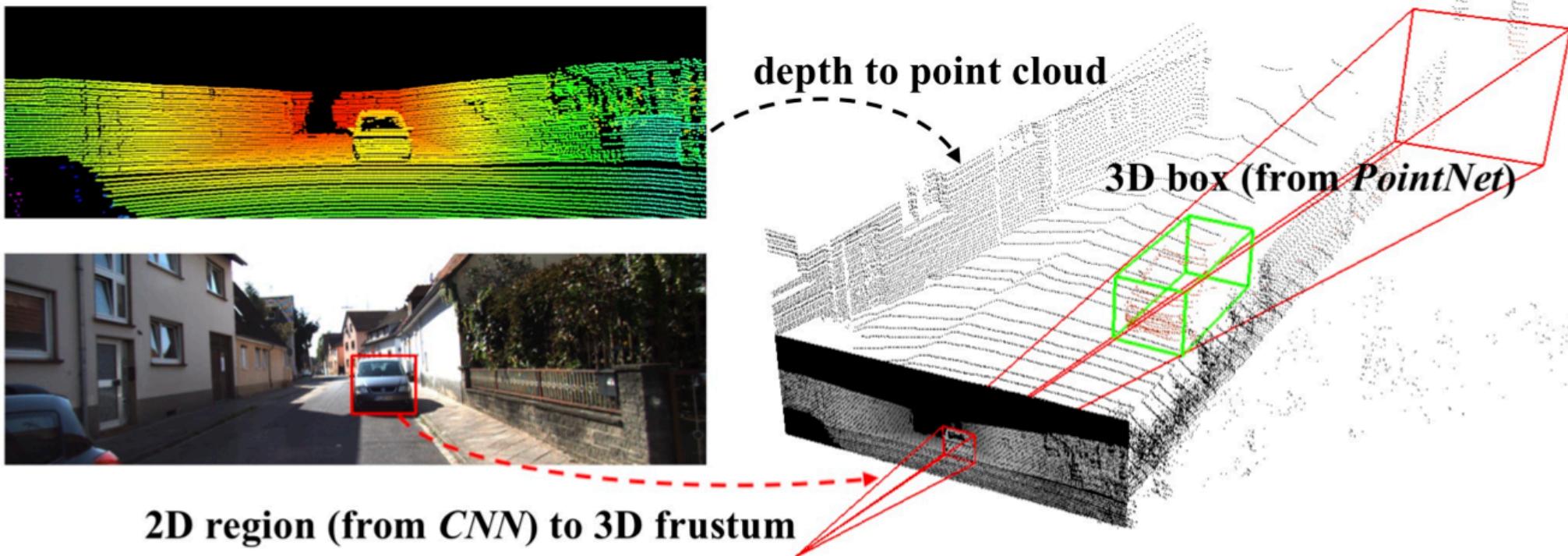
A 2D bounding box on an image is a **frustrum** in the 3D space

Localize an object in 3D:
The object can be anywhere in the **camera viewing frustum!**

Image source: https://www.pcmag.com/encyclopedia_images/_FRUSTUM.GIF

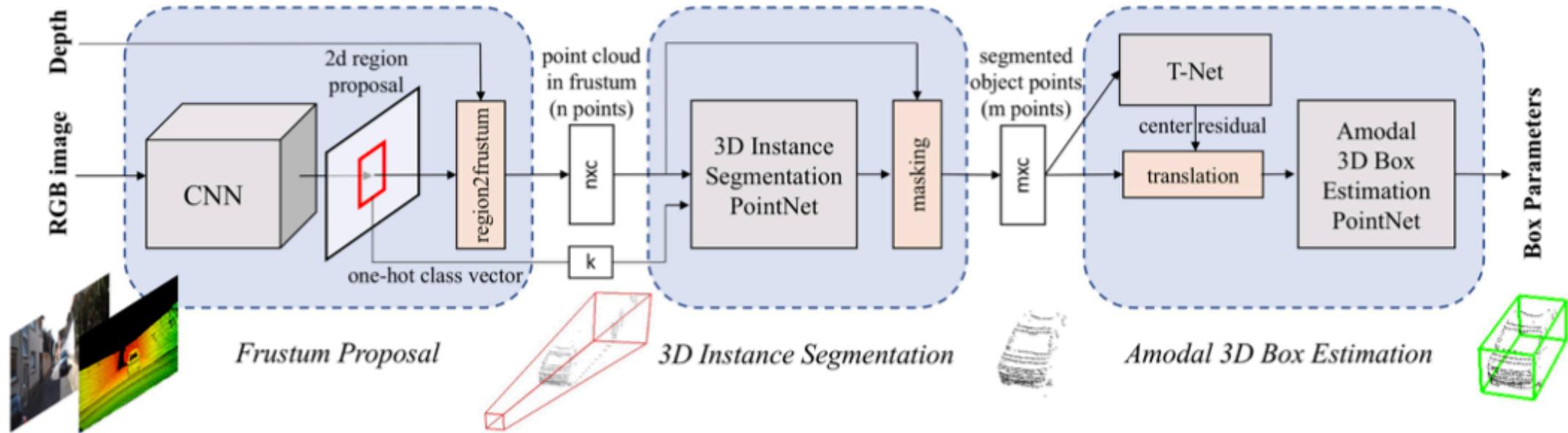
3D Object Detection from RGB-D

Frustum PointNet



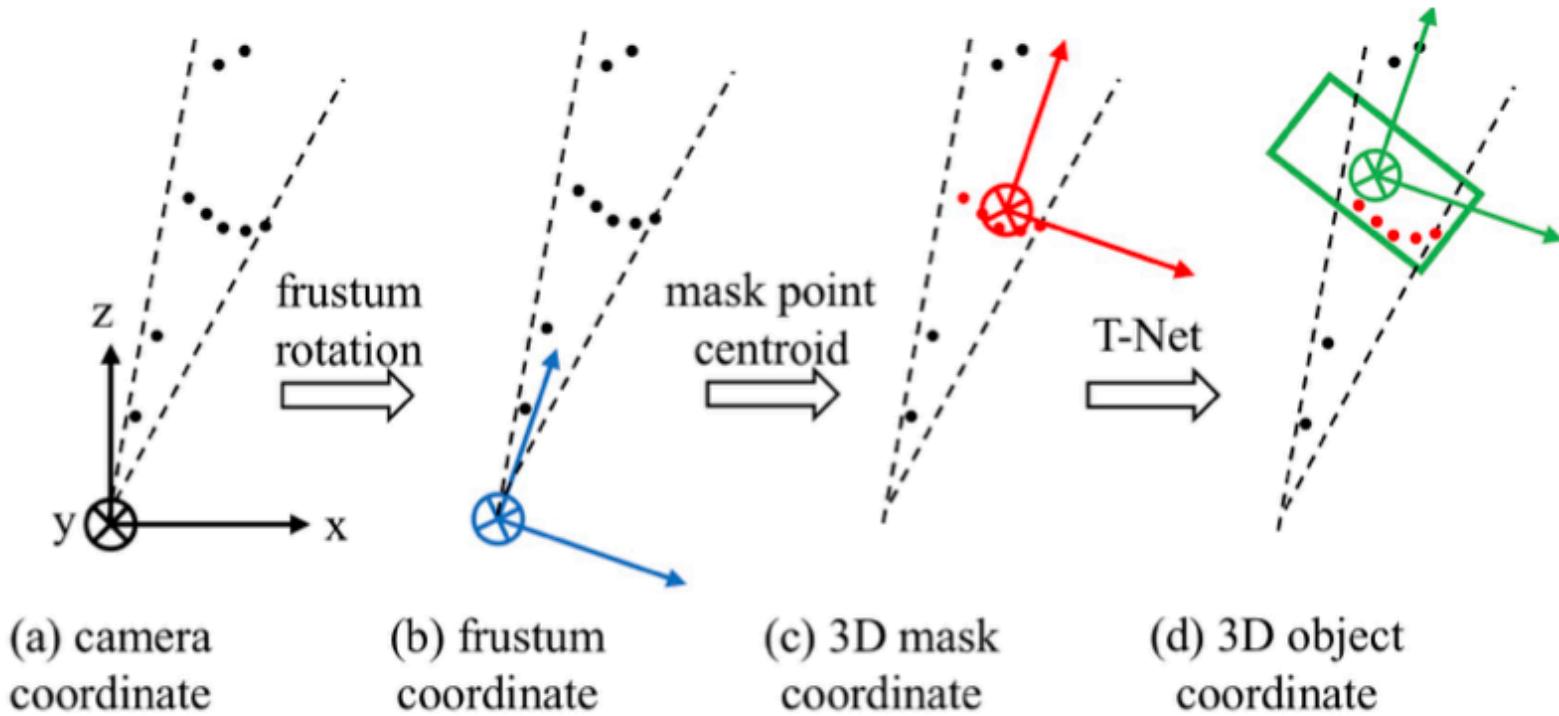
Qi, Charles R., et al. "Frustum pointnets for 3d object detection from rgb-d data." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Pipeline of Frustum PointNet



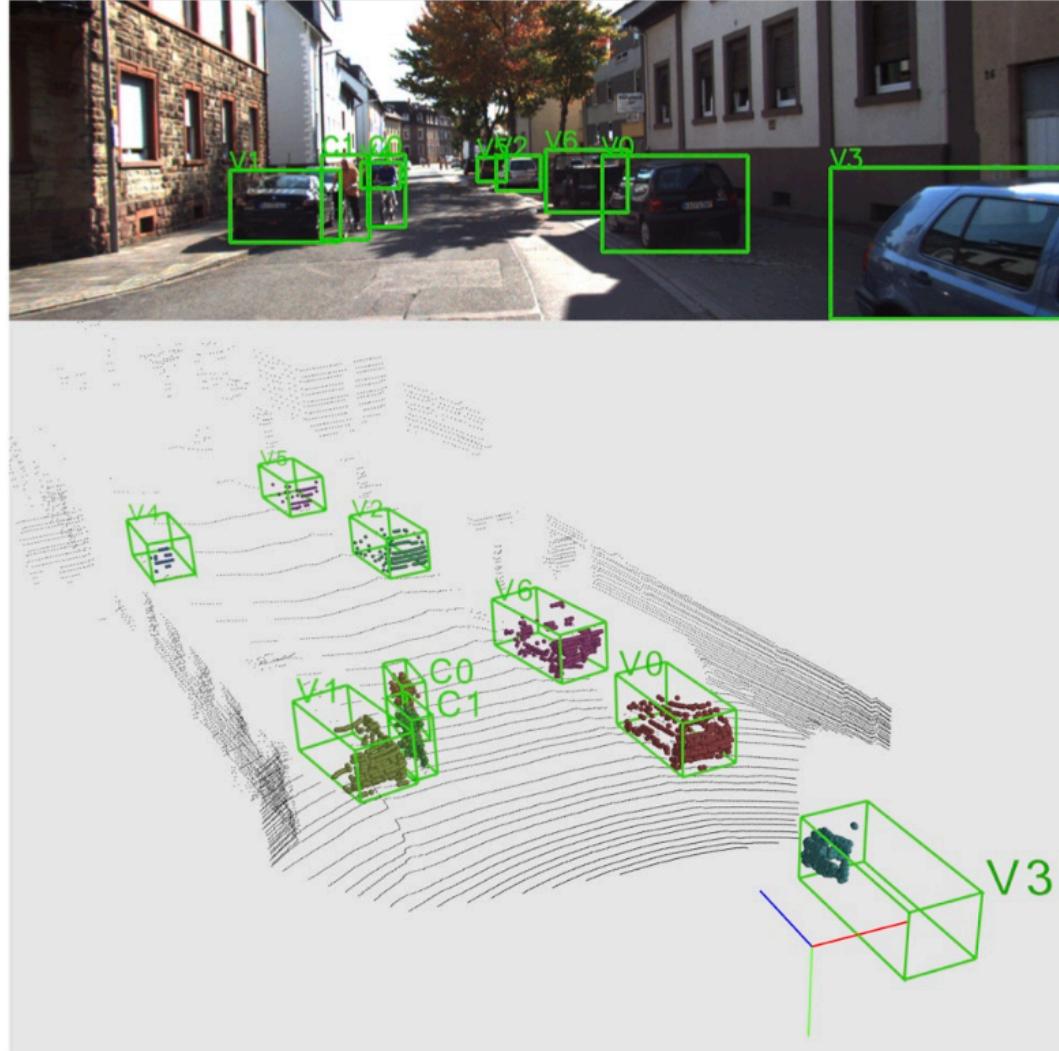
Qi, Charles R., et al. "Frustum pointnets for 3d object detection from rgbd data." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Coordinate Systems for Point Cloud

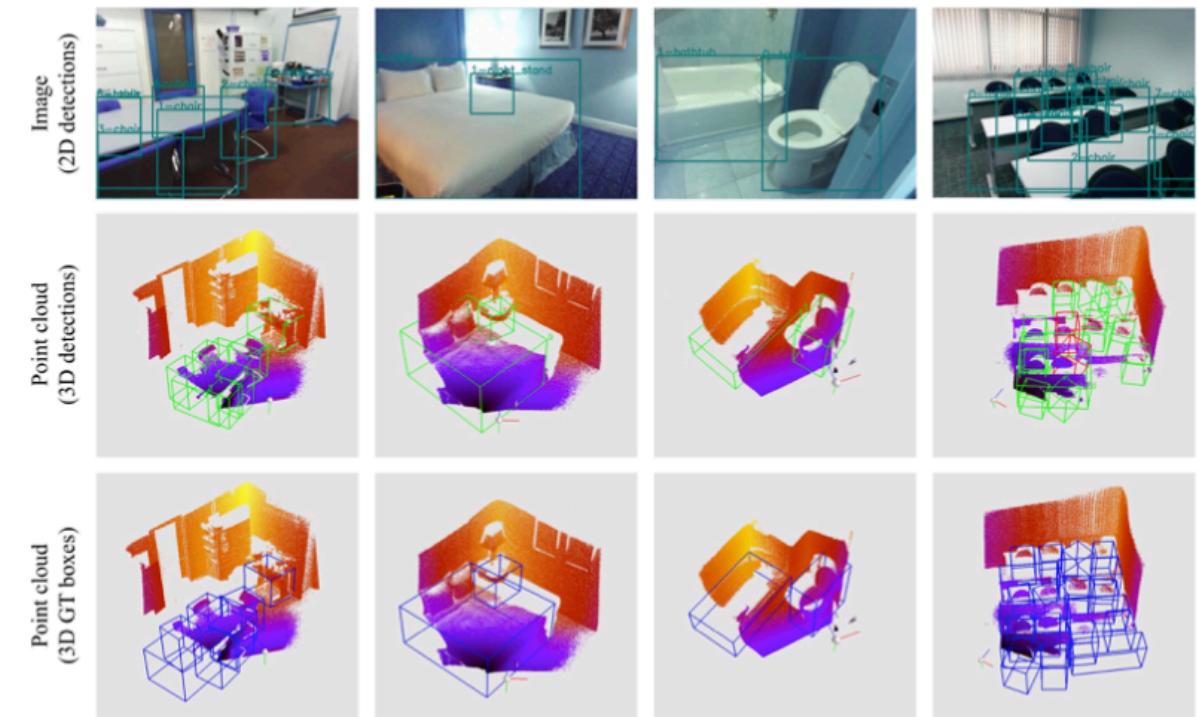


Qi, Charles R., et al. "Frustum pointnets for 3d object detection from rgb-d data." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Result Visualization

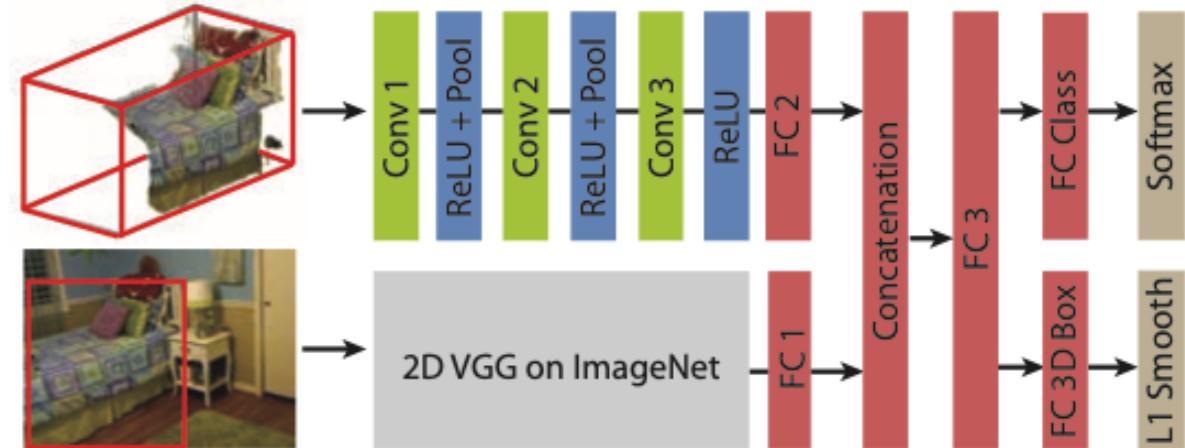
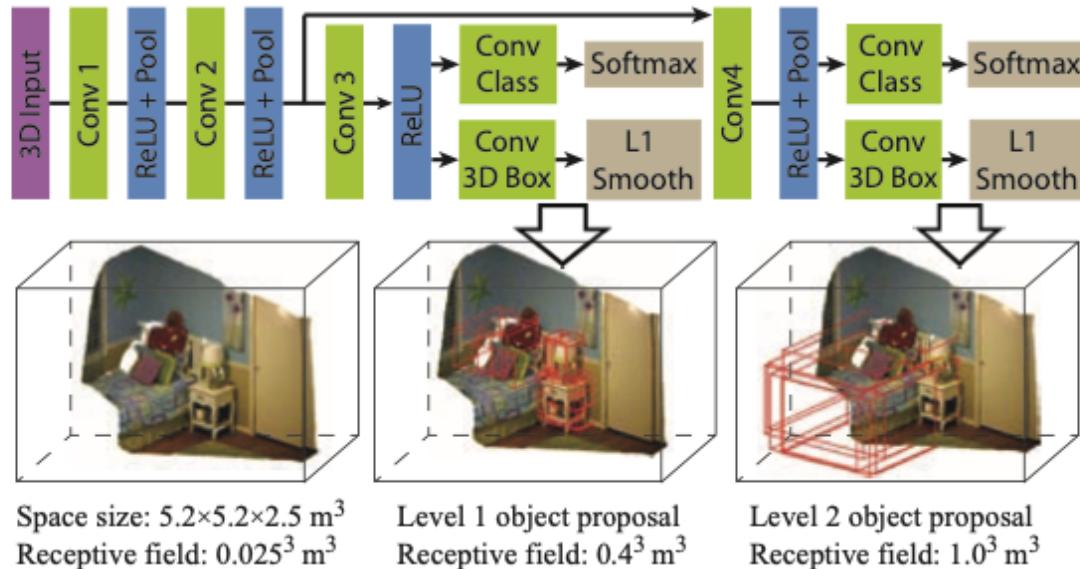


KITTI



SUN-RGBD

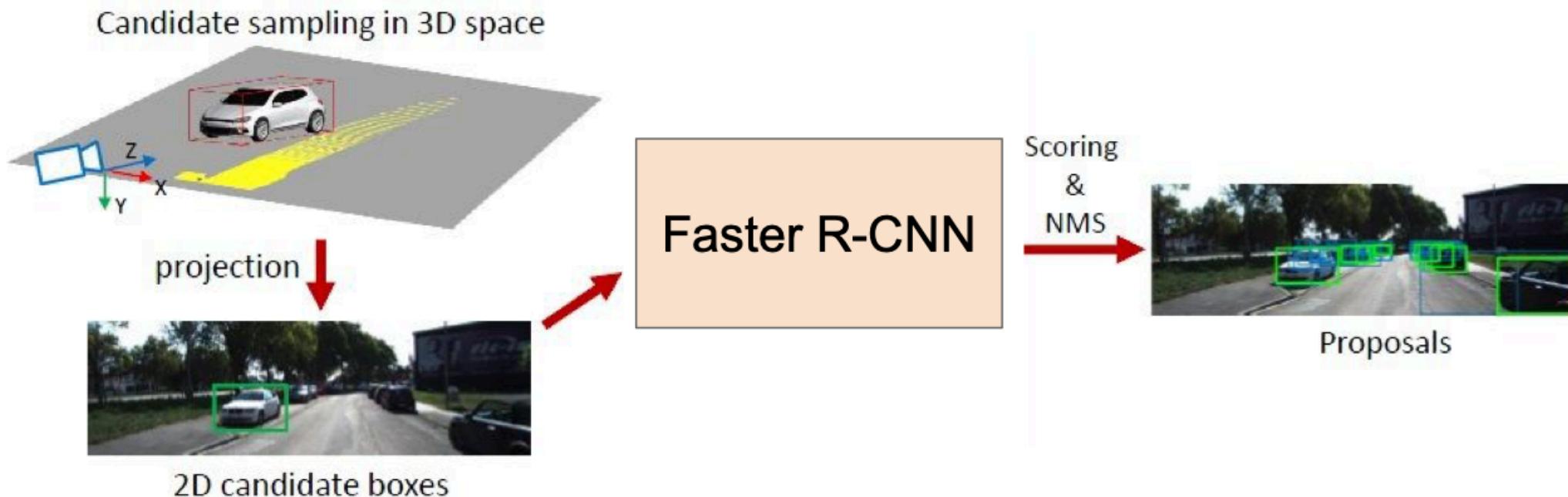
Deep Sliding Shape



Very expensive to perform sliding windows in 3D!

Song, Shuran, and Jianxiong Xiao. "Deep sliding shapes for amodal 3d object detection in rgb-d images." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

3D Object Detection: Monocular Camera

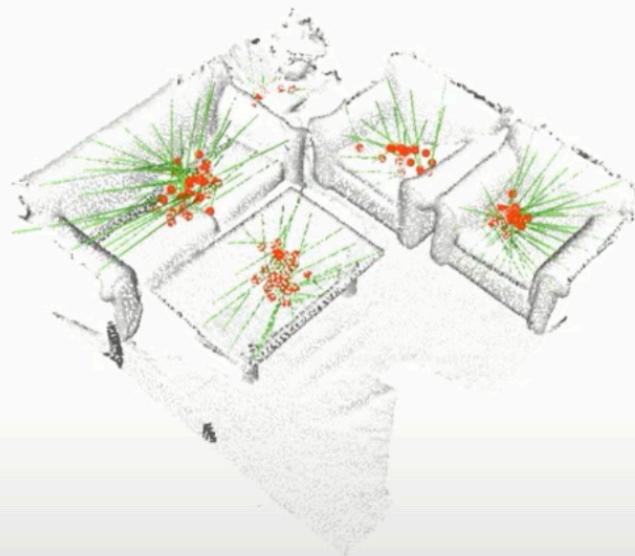


- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

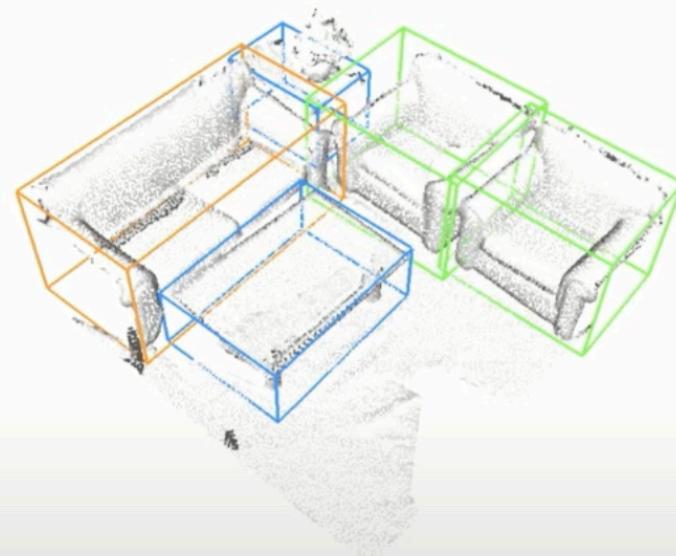
Chen, Xiaozhi, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "Monocular 3d object detection for autonomous driving." CVPR 2016.

VoteNet

Our idea: “ask” the surface points to
vote for object centers



Voting from surface points



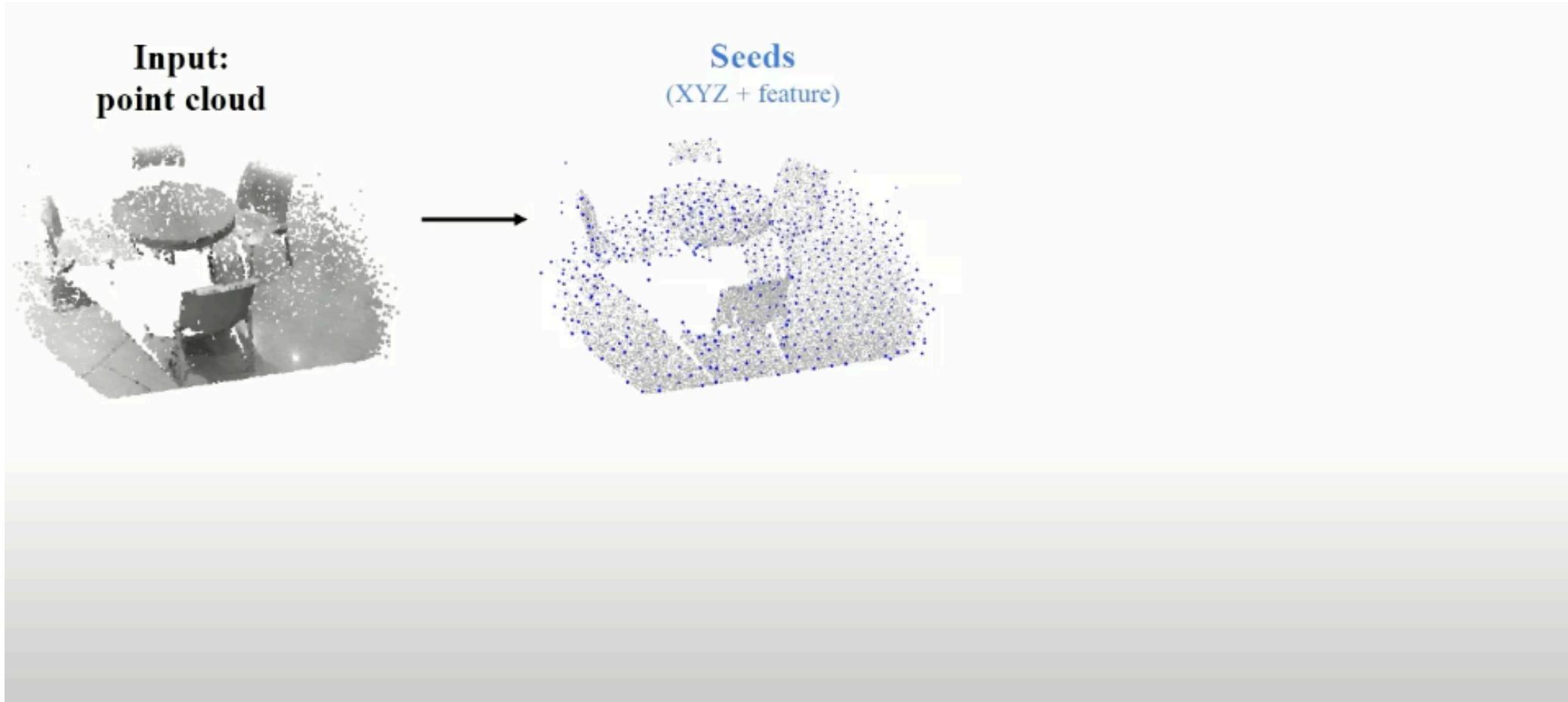
Detected 3D bounding boxes

Deep Hough Voting: Detection Pipeline



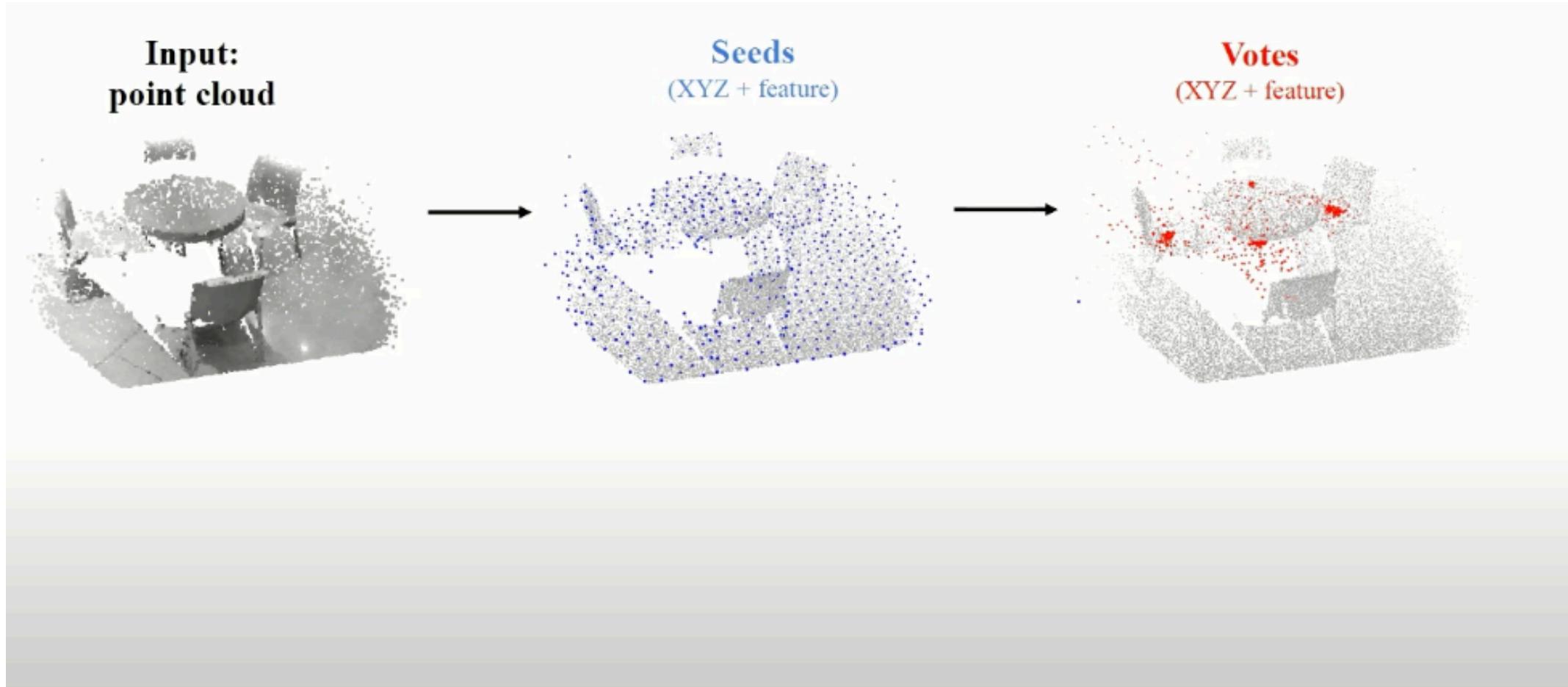
Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Deep Hough Voting: Detection Pipeline



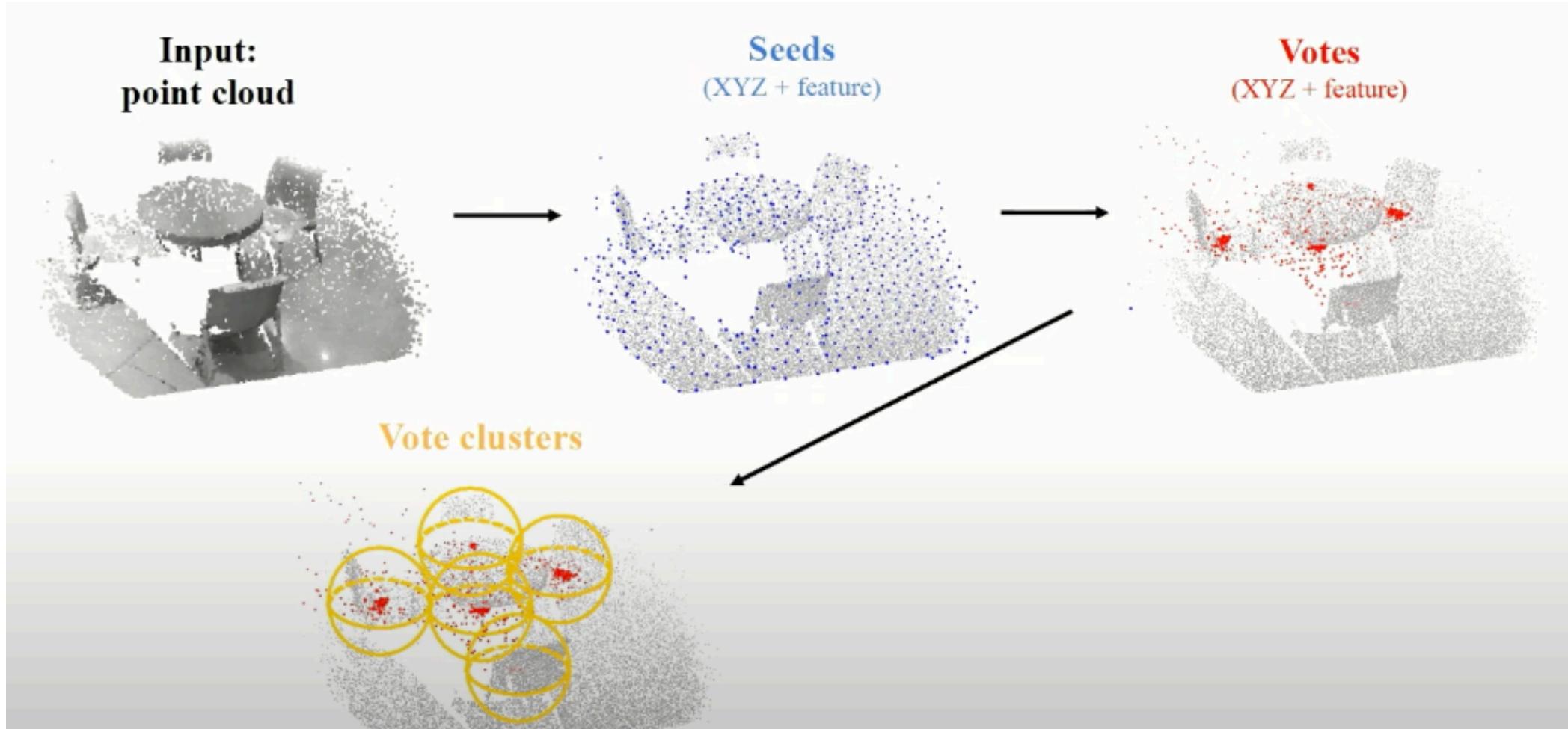
Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Deep Hough Voting: Detection Pipeline



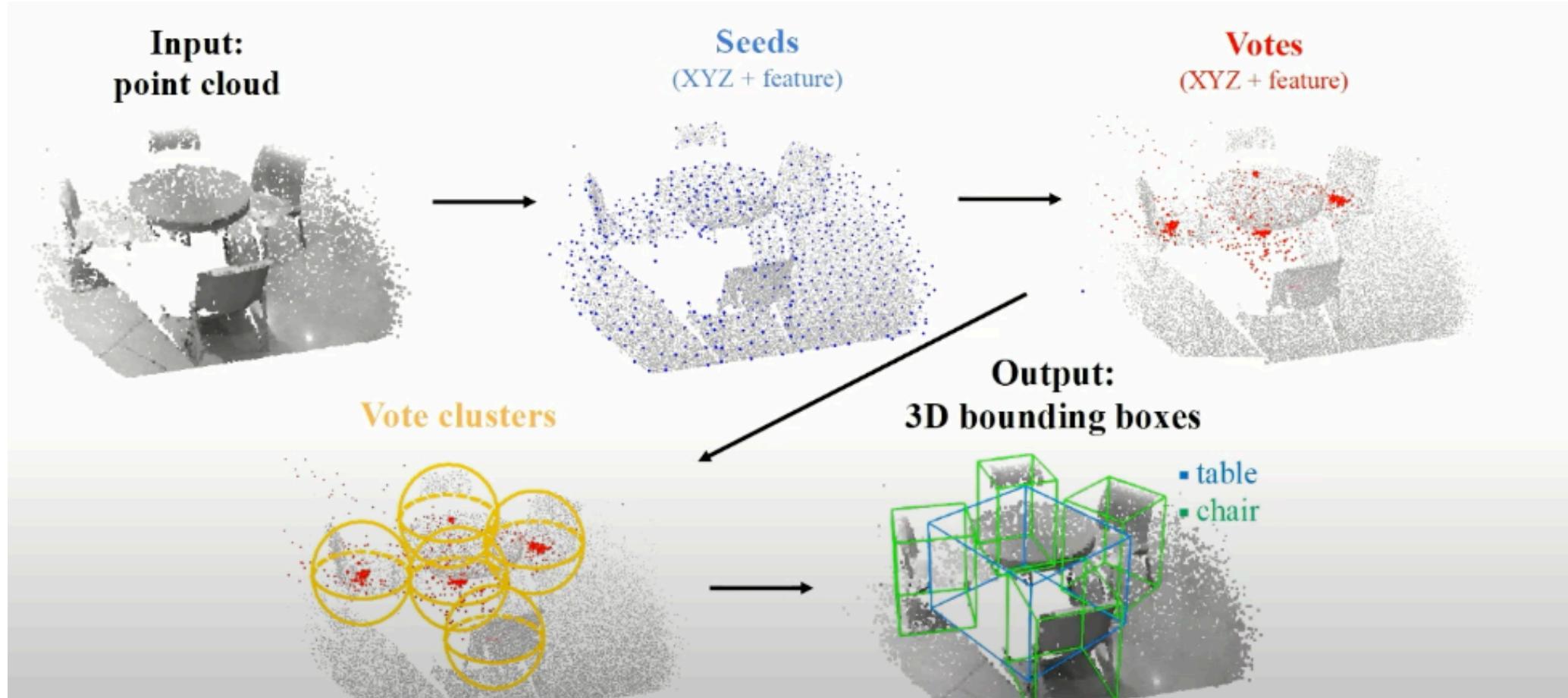
Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Deep Hough Voting: Detection Pipeline



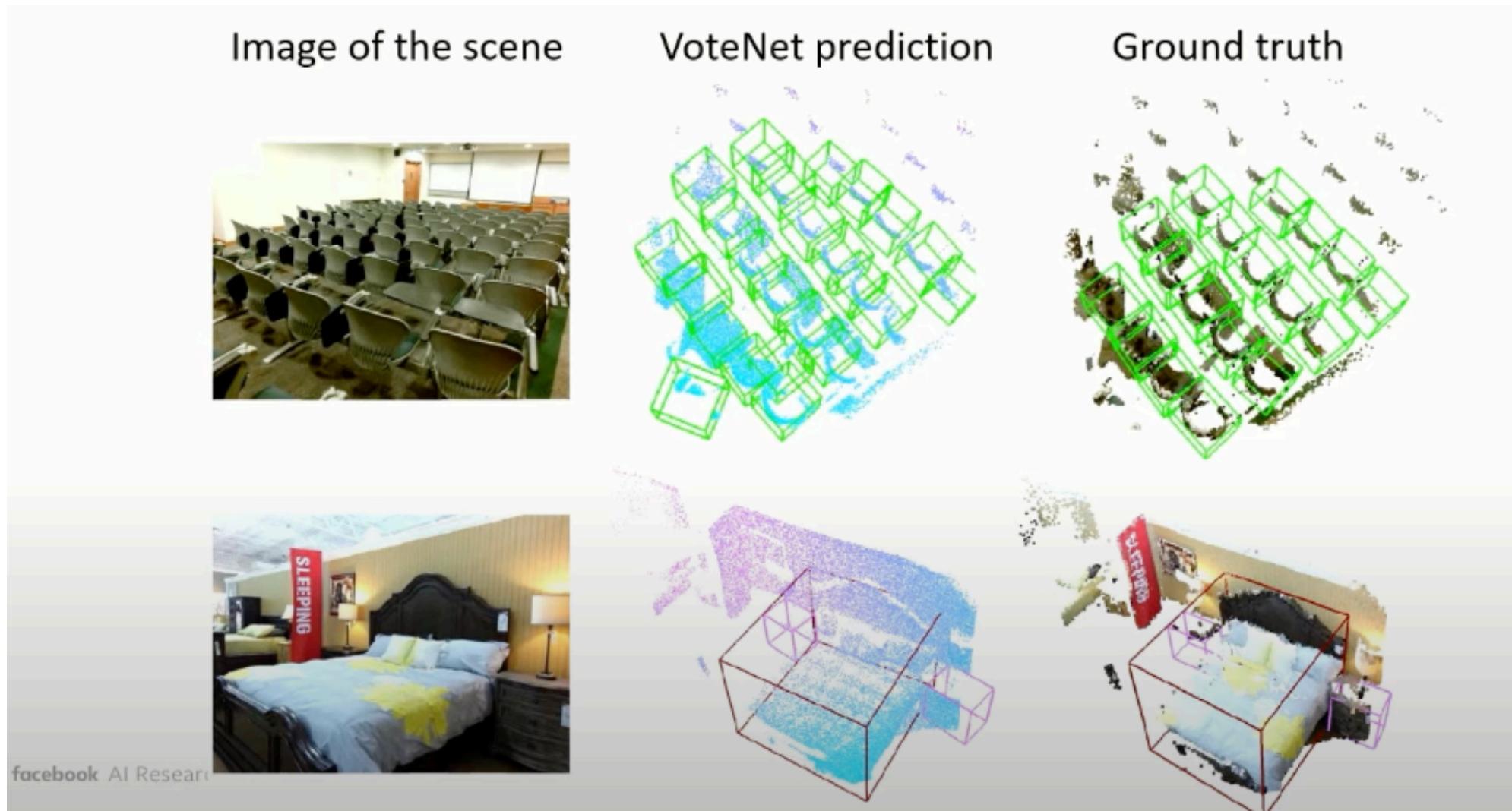
Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Deep Hough Voting: Detection Pipeline



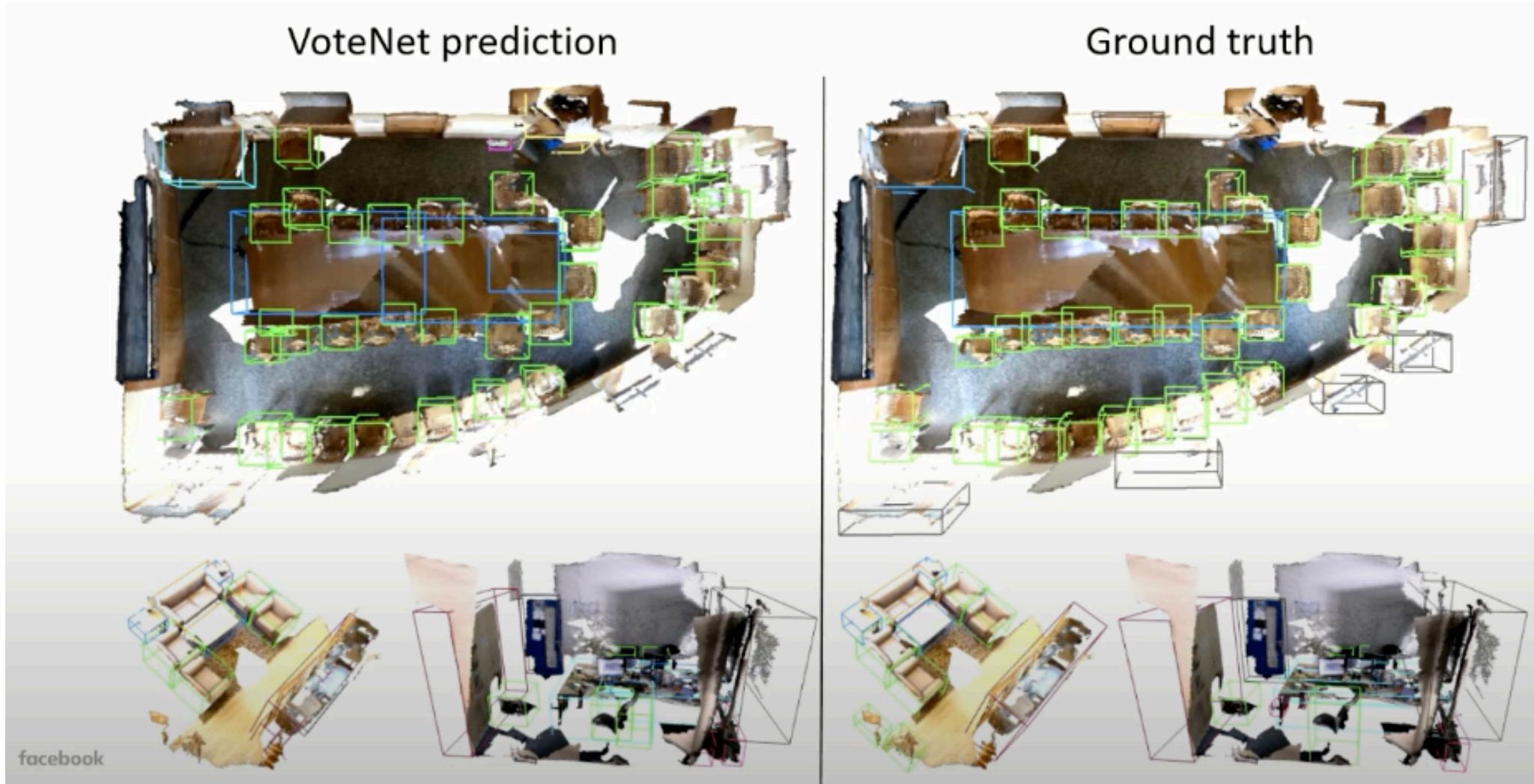
Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Results: SUN RGB-D (single depths)



Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Results: ScanNet (3D Reconstruction)



Qi, Charles R., et al. "Deep hough voting for 3d object detection in point clouds." *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

3D Instance Segmentation

- Top-Down
 - GSPN
- Bottom-Up
 - SGPN
 - PointGroup



Introduction to Computer Vision

Next week: Lecture 13,
Generative Model