

3D Point Clouds

Lecture 6 – 3D Object Detection

主讲人 黎嘉信

Aptiv 自动驾驶
新加坡国立大学 博士
清华大学 本科





1. Image Based Object Detector



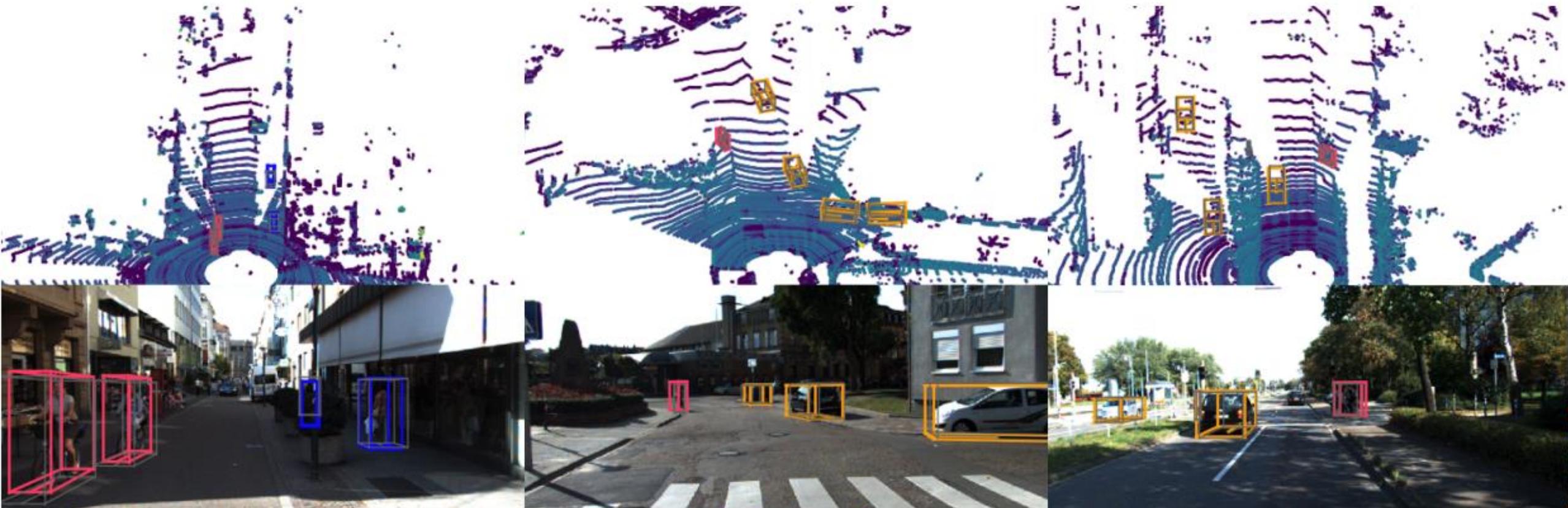
2. Point Cloud – VoxelNet & PointPillars



3. Point Cloud – PointRCNN



What is 3D Object Detection?

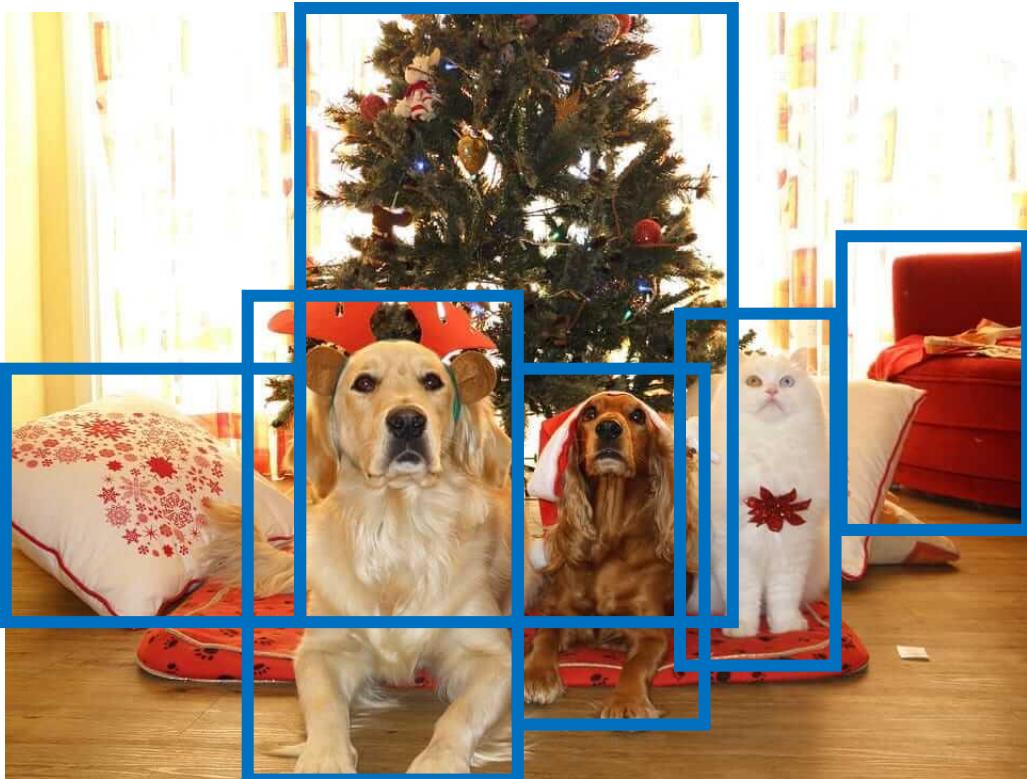


- ❖ Input: Point cloud and/or images
- ❖ Output: 3D bounding boxes – $[x, y, z, \text{length}, \text{width}, \text{height}, \text{heading}, \text{category}]$



How to Perform Object Detection?

Object Localization



Object Classification



Dog? Yes
Cat? No
Tree? No
Pillow? No

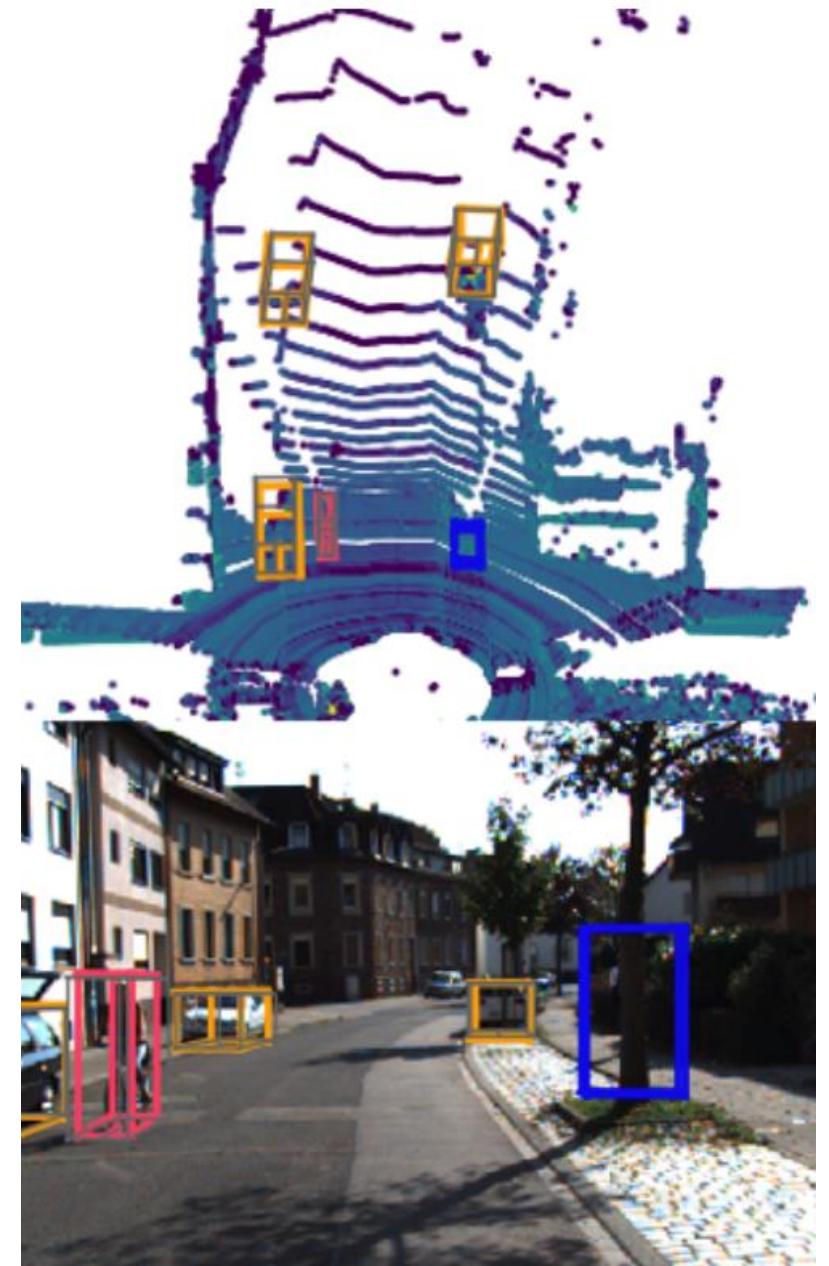


Image

- Occlusion
- Scale
- Class imbalance
- Rotation
- Generalization
- Illumination
-

Point Cloud

- Occlusion
- Scale
- Class imbalance
- Rotation
- Generalization
- Illumination
- **Lack of texture**
- **Sparsity**





Results Response JSON

< >

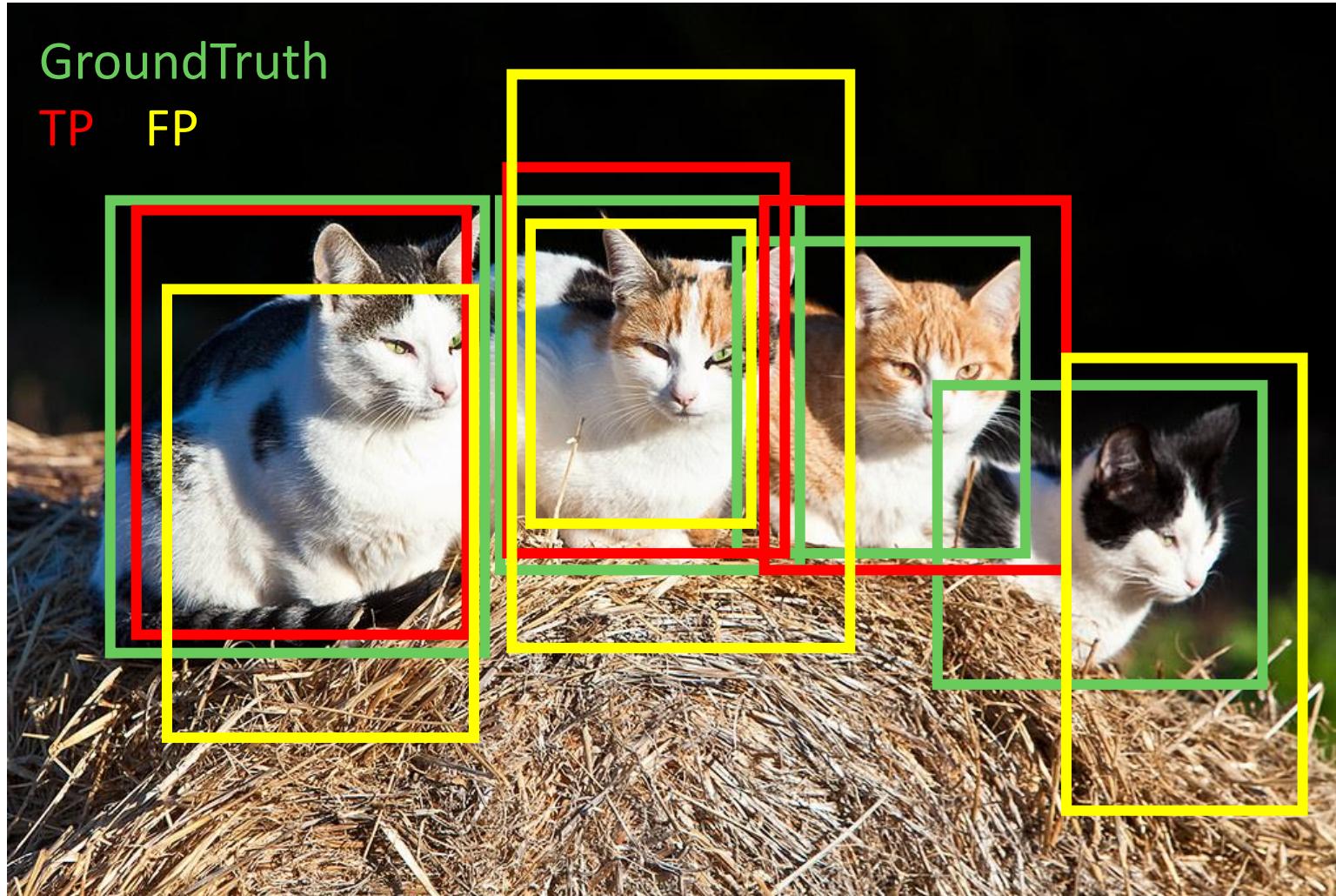
Gender Male

Upper Clothing Gray Color Family

Lower Clothing Gray Color Family



How good is this detection result?

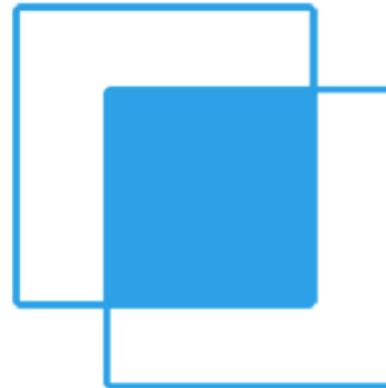




Intersection over Union (IoU)

- Determine whether a **predicted box** is matched with the **ground truth box**

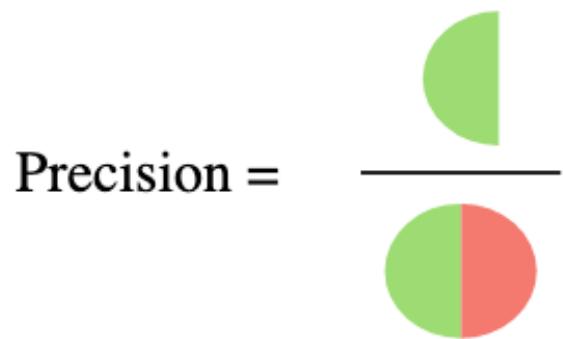
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$





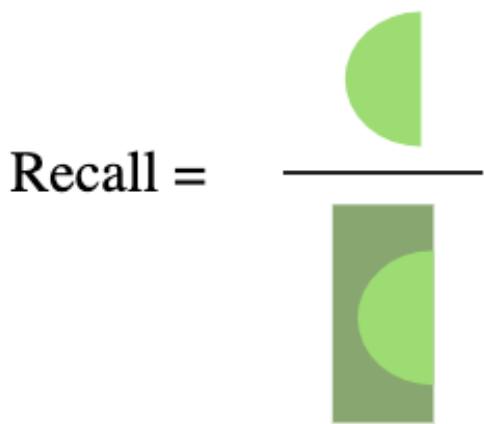
Precision & Recall

How many selected items are relevant?

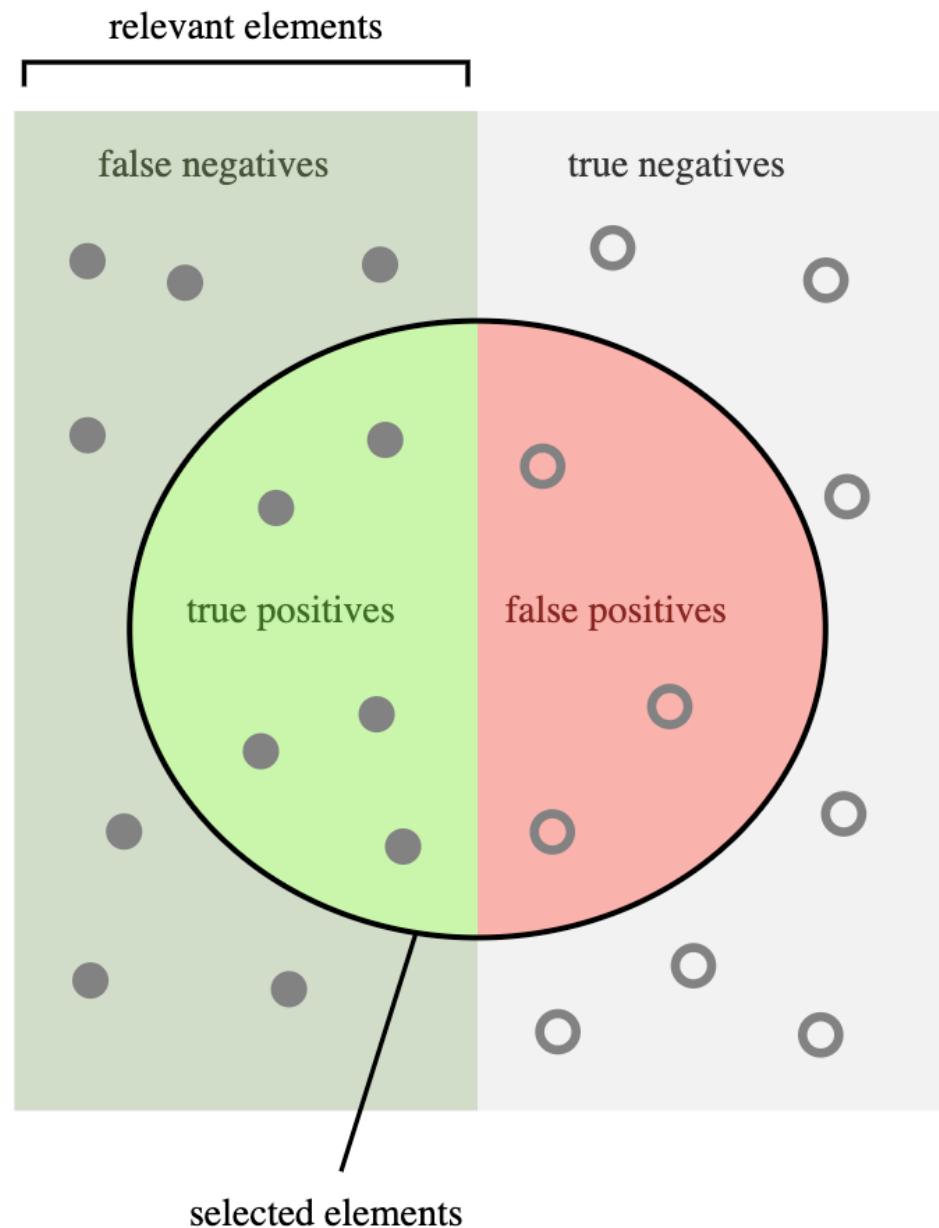


$$\text{Precision} = \frac{TP}{TP+FP} = \frac{TP}{P}$$

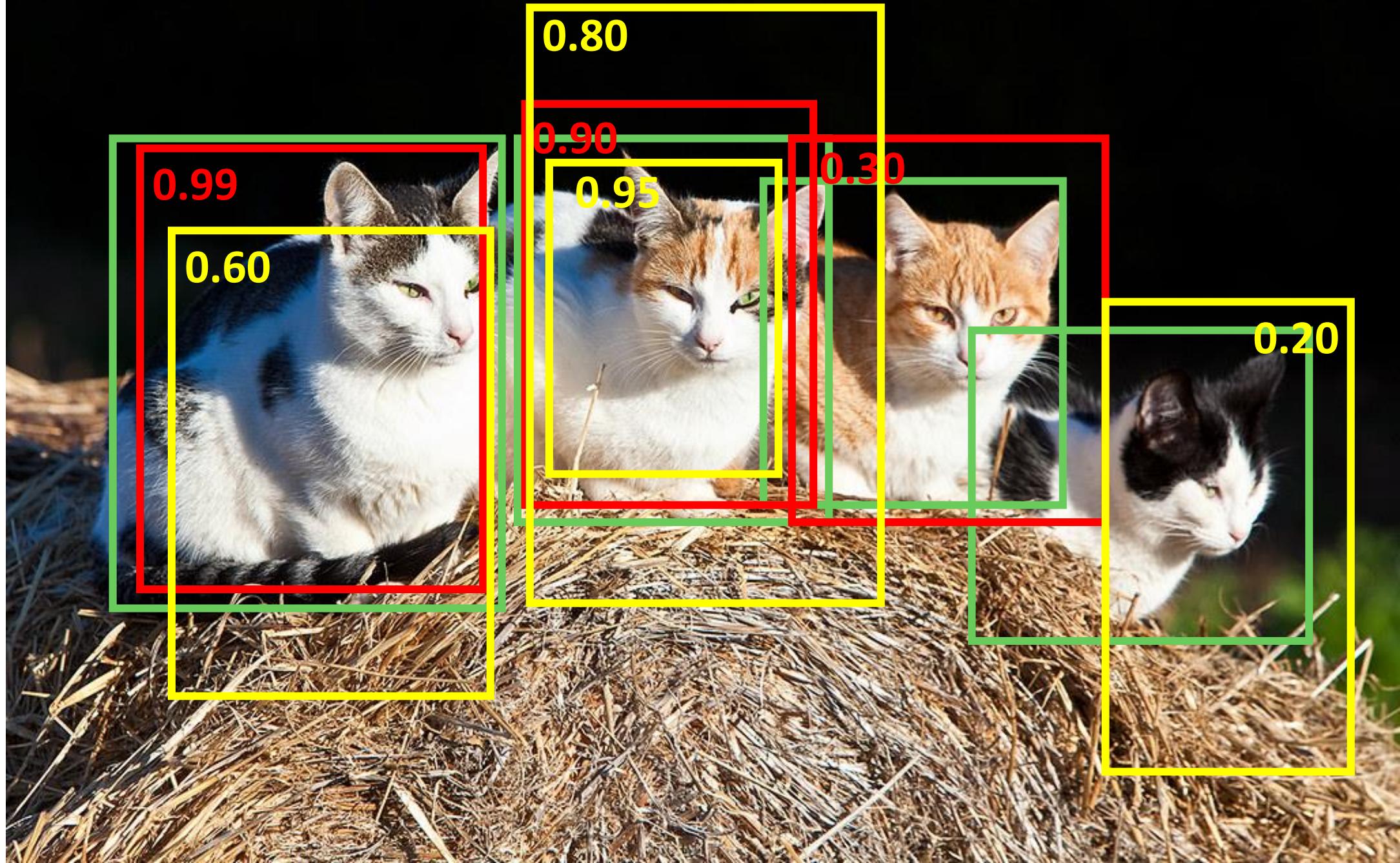
How many relevant items are selected?



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



GroundTruth TP FP





Average Precision (AP)

For one category

1. Choose IoU threshold
2. Choose confidence threshold
 - Only choose boxes with confidence > threshold, of that category
3. Determine each selected box is TP / FP
 - TP: IoU > threshold & Highest IoU – **One TP per GT Box!**
 - FP: IoU < threshold / Duplicated box
 - Duplicated box: **each ground truth box can be associated with one TP only**, others are FP.
4. Compute Precision / Recall
5. Get P-R curve by repeating 2-4 with decreasing confidence threshold
6. Get AP by calculating the Area Under Curve (AUC)

Repeat above for other categories

Mean over all categories → mAP



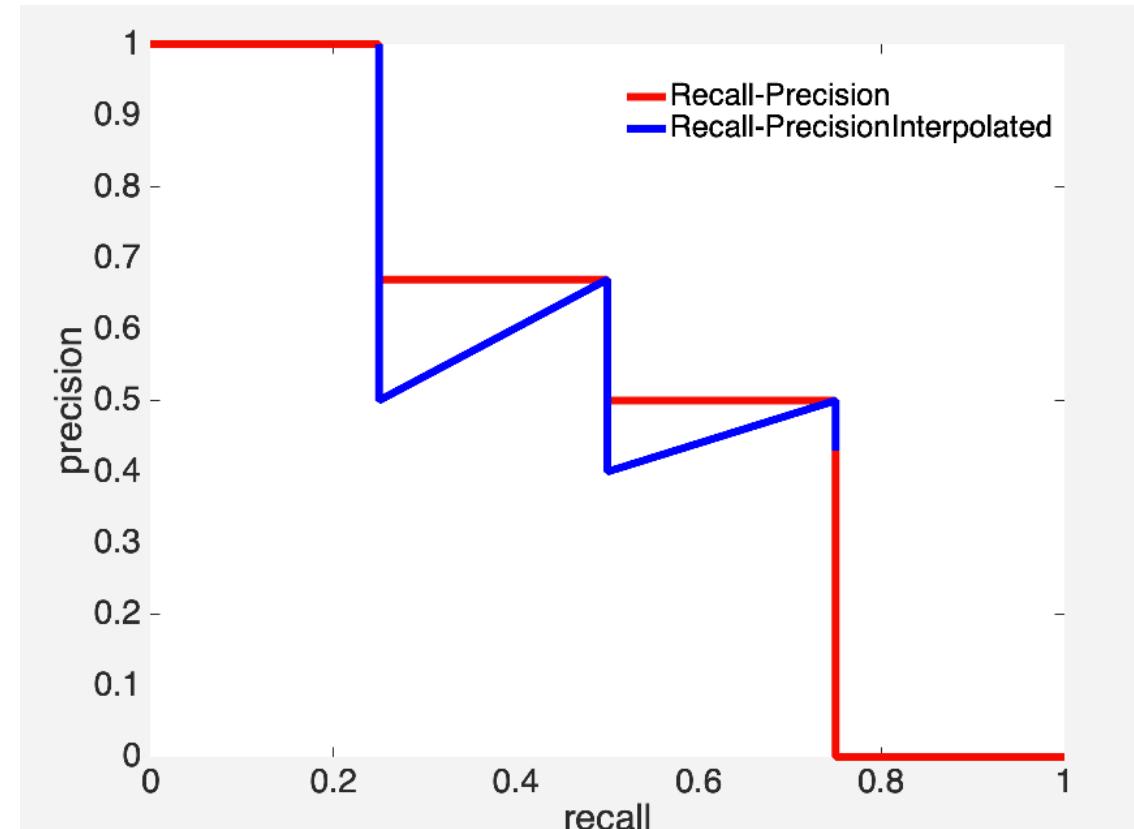
Precision Interpolated:

$$P_{interp}(r) = \max_{r' \geq r} p(r')$$

	Confidence	TP/FP	Precision	Recall	Precision Interpolated
1	0.99	TP	1/1=1	1/4=0.25	1
2	0.95	FP	1/2=0.5	1/4=0.25	1
3	0.9	TP	2/3=0.67	2/4=0.5	0.67
4	0.8	FP	2/4=0.5	2/4=0.5	0.67
5	0.6	FP	2/5=0.4	2/4=0.5	0.67
6	0.3	TP	3/6=0.5	3/4=0.75	0.5
7	0.2	FP	3/7=0.43	3/4=0.75	0.5

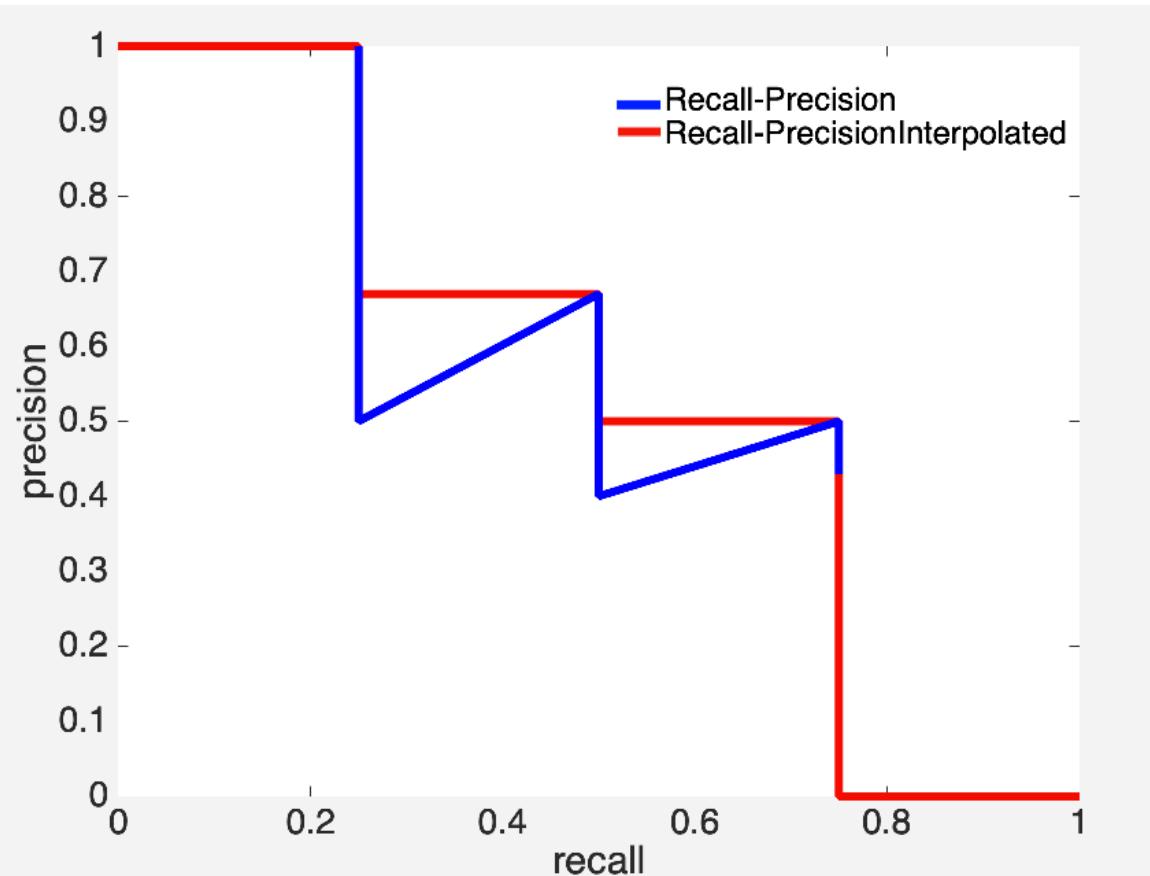
$$\text{Precision} = \frac{TP}{TP+FP} = \frac{TP}{P}$$

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

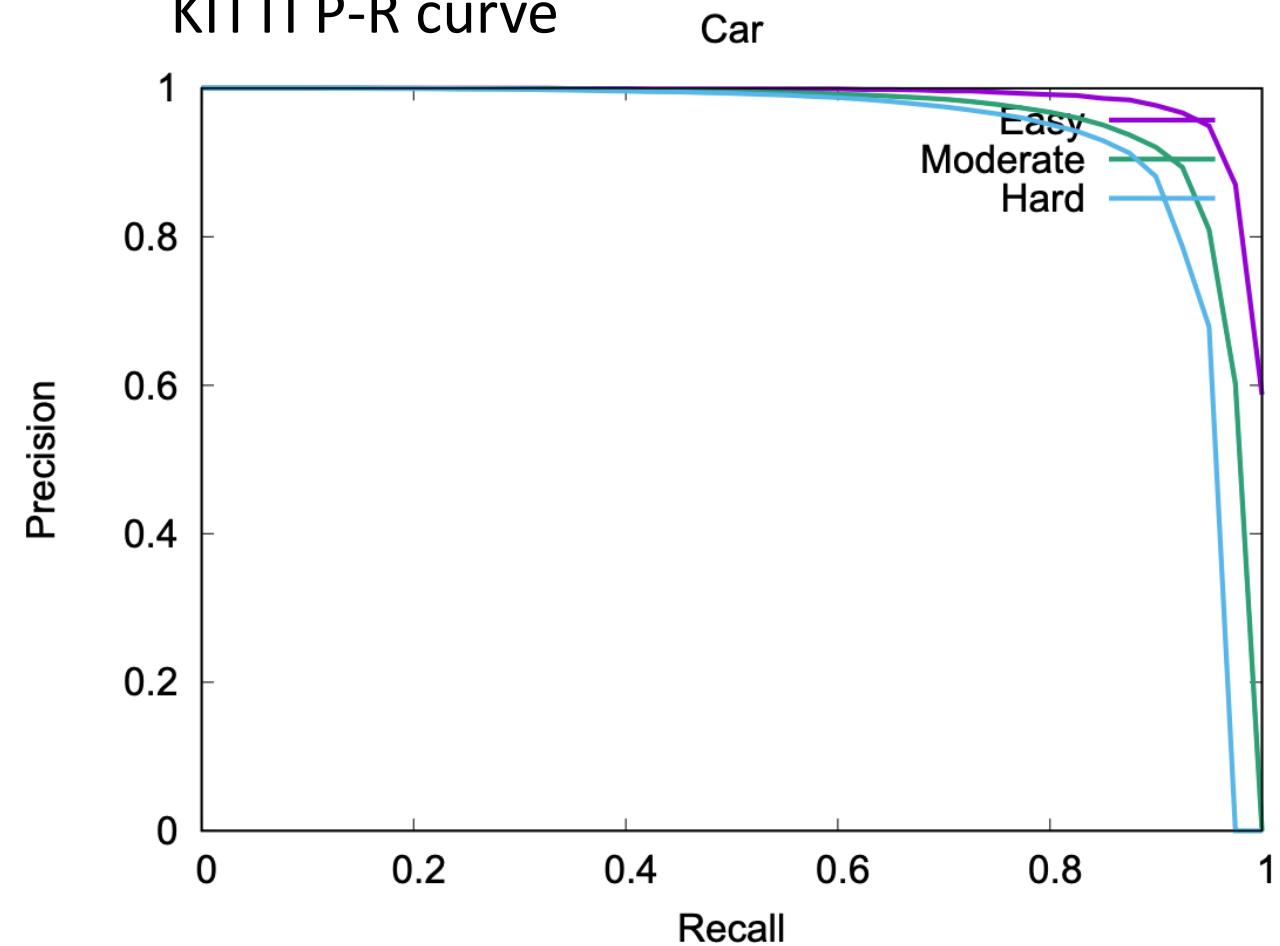




KITTI 3D Object Detection P-R Curve



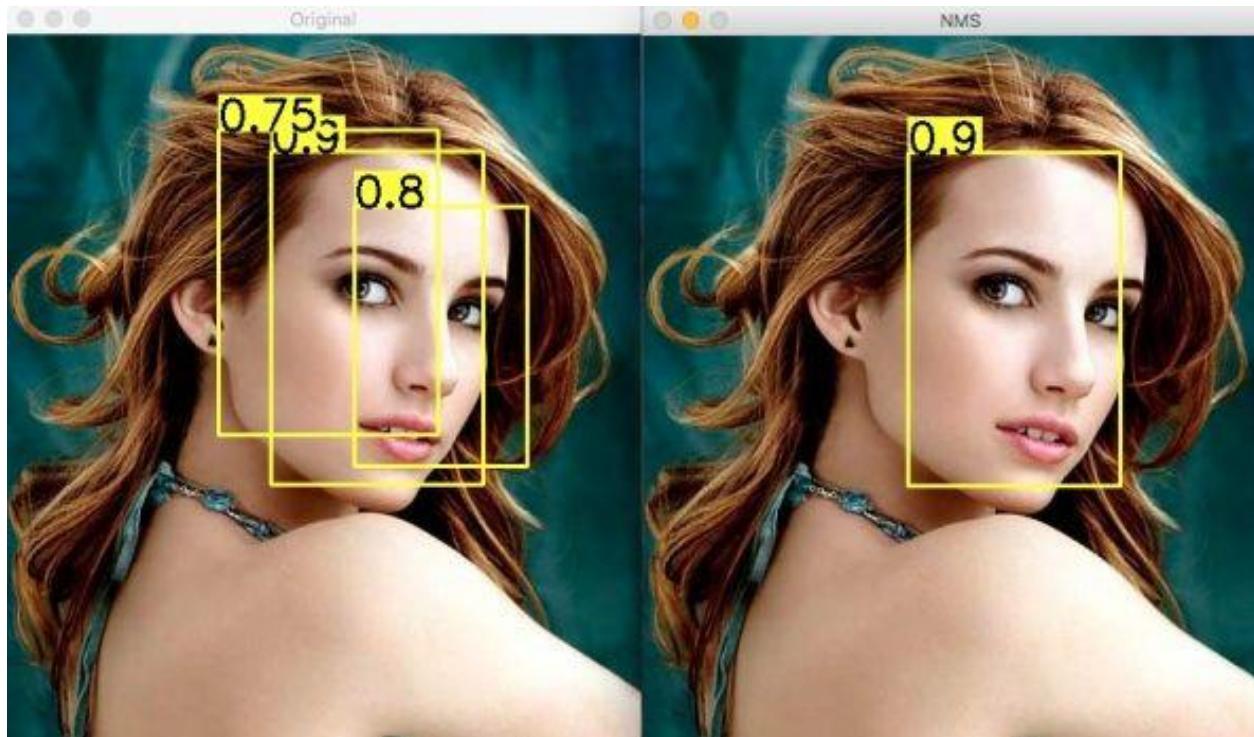
KITTI P-R curve





Non-Maximum Suppression (NMS)

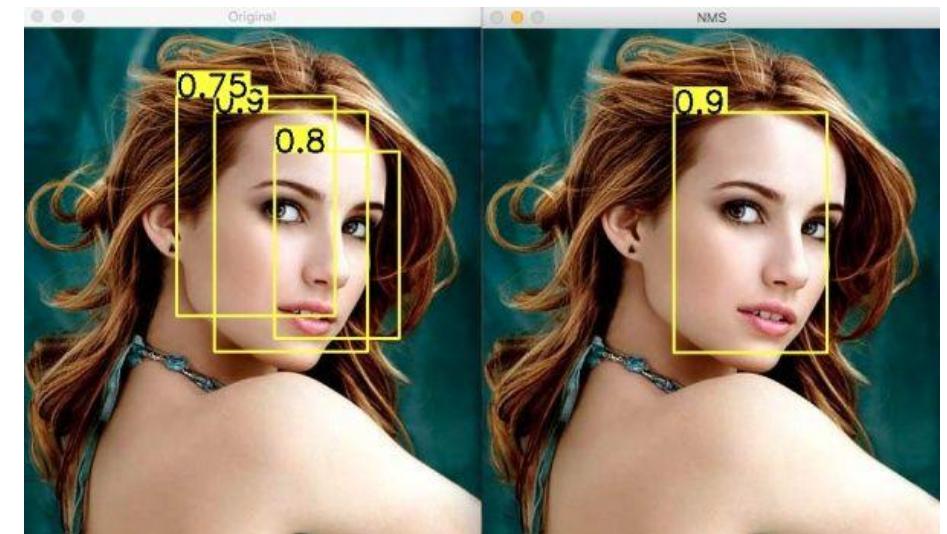
- Usually there are multiple predicted boxes on one object
- Select the one with highest predicted score/confidence





Steps for NMS

1. Keep all predicted boxes in the input list
2. Sort the predicted boxes by confidence
3. Pick the highest one, add it to the output list, remove it from the input list
4. Iterate all boxes in the input list
 1. Compute IoU between the picked box
 2. If $\text{IoU} > \text{threshold } \tau \rightarrow$ remove the box from the input list
5. Repeat step 3, 4 until the input list is empty
6. Output list is the result.

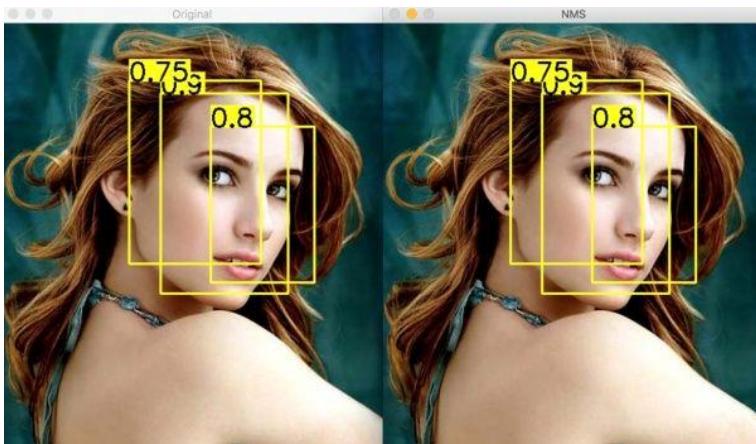




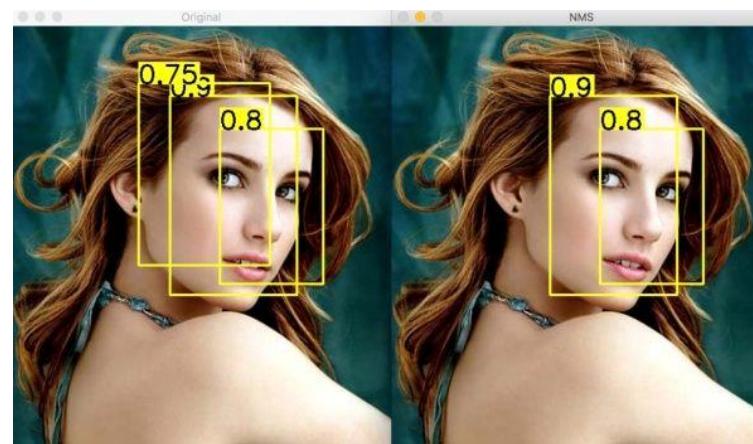
Non-Maximum Suppression (NMS)

- The larger is threshold τ , the more boxes NMS keeps

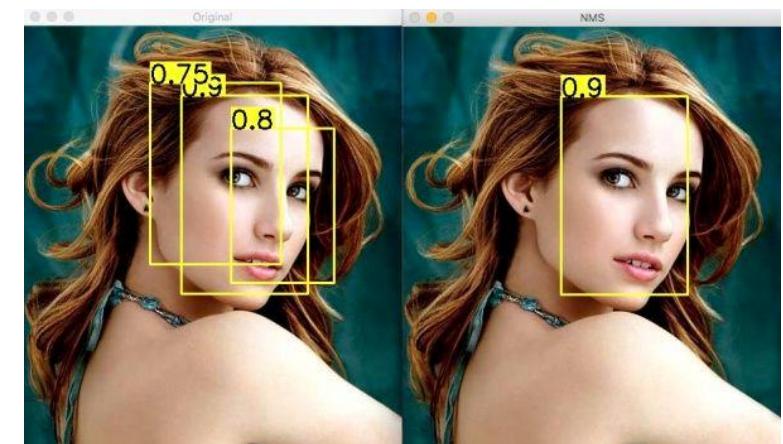
$\tau = 0.6$



$\tau = 0.5$

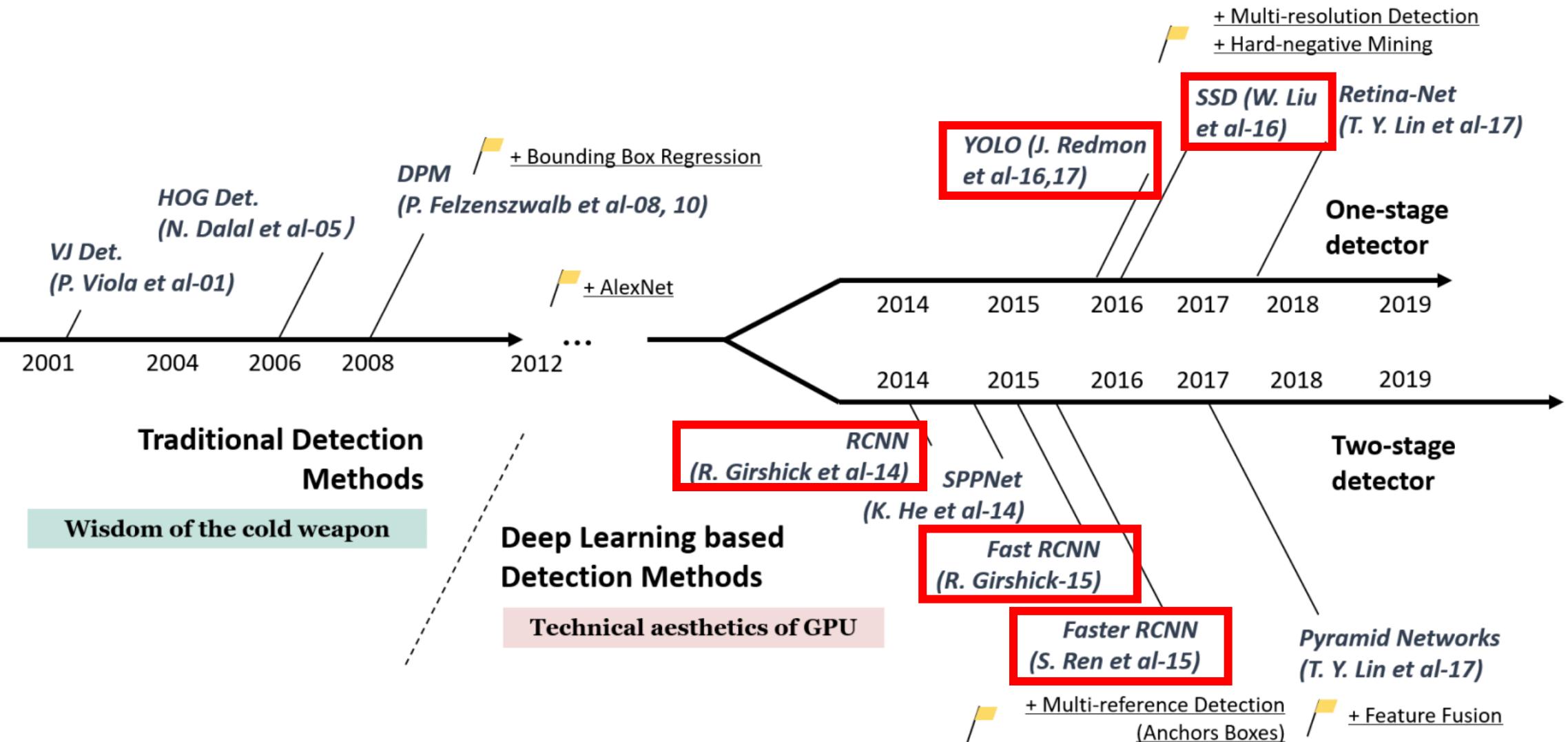


$\tau = 0.4$





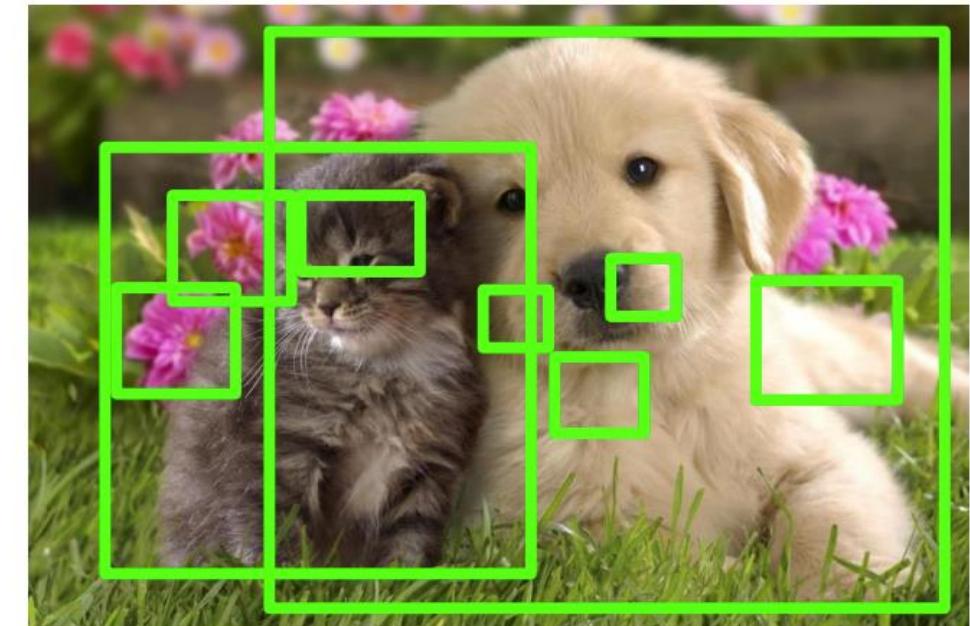
Object Detection Milestones

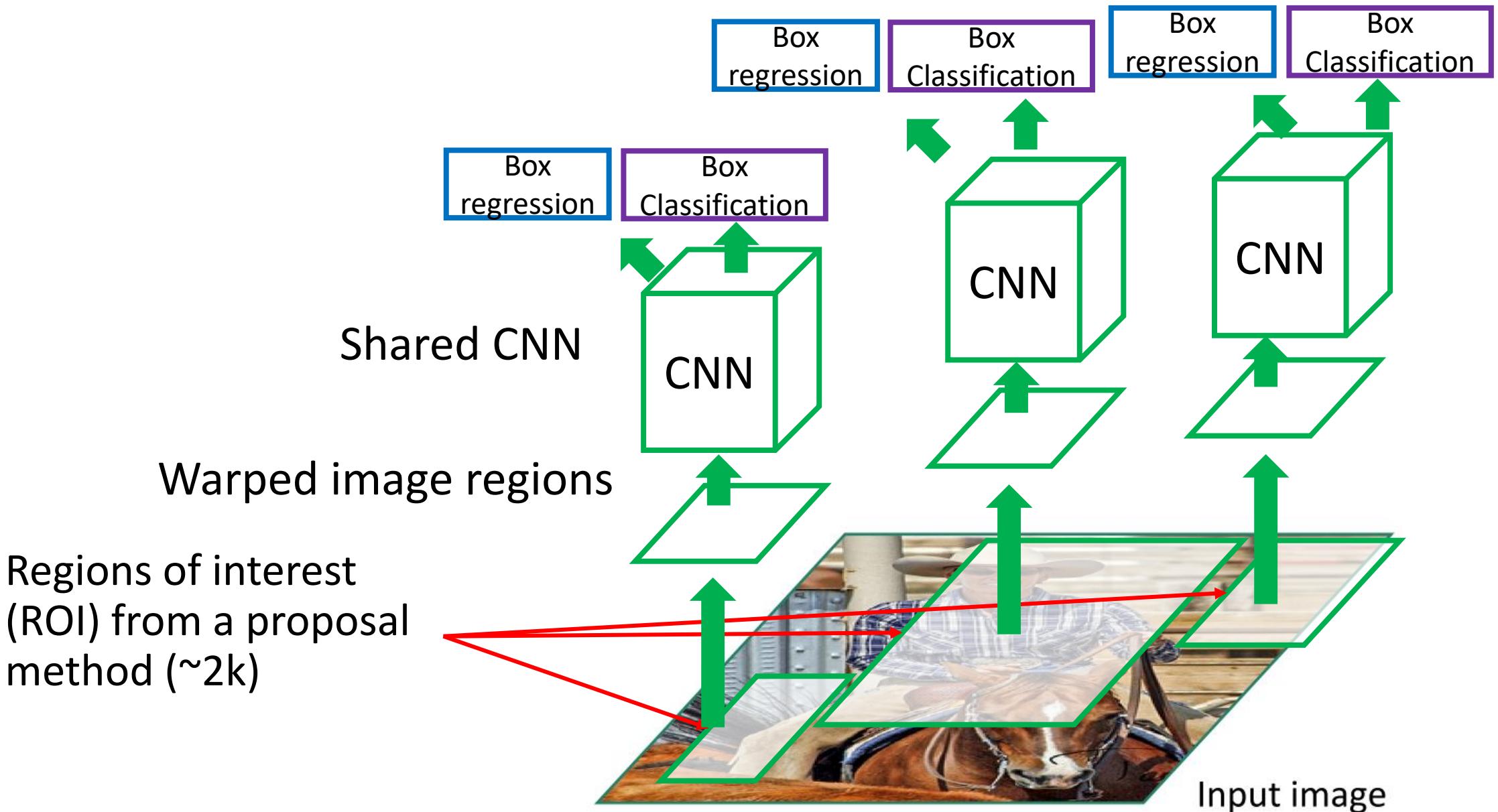




❖ Detection = Localization + Classification

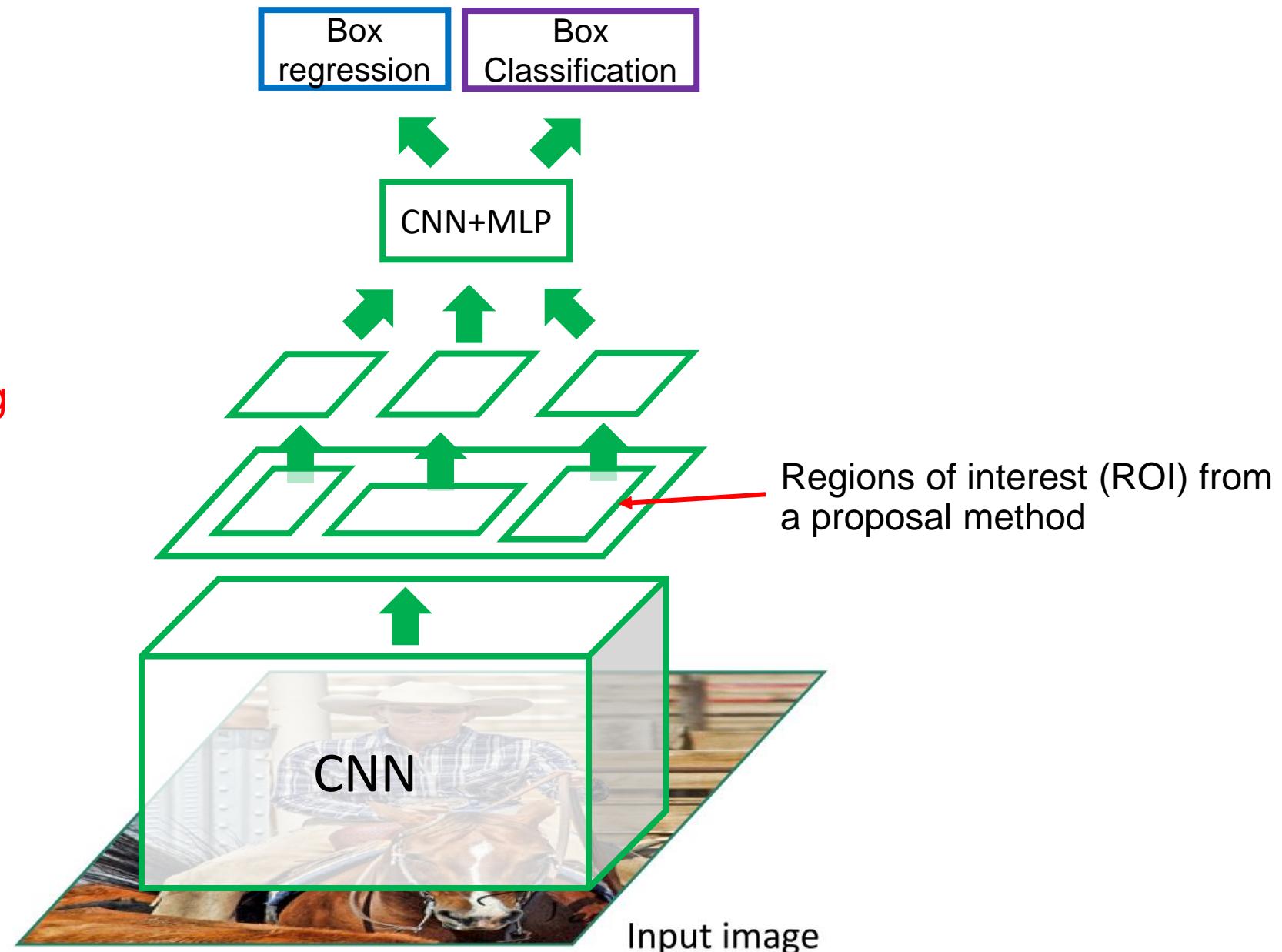
- Classification is well known
- What if we can get **region proposals** somewhere else?
 - E.g., color based clustering





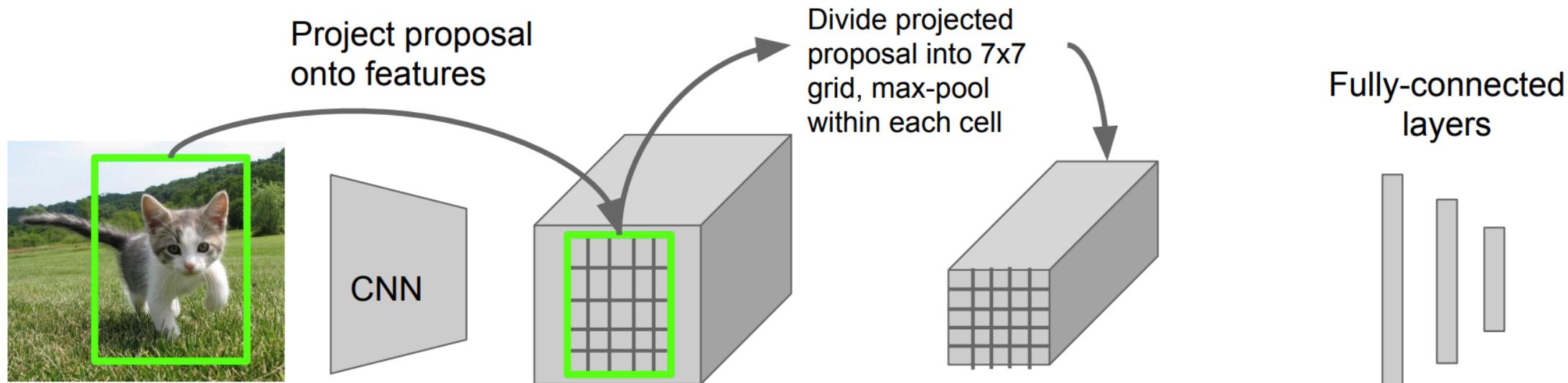


ROI Pooling
The feature map
Single CNN:
Input: image
Output: Feature map





Fast RCNN – ROI Pooling



Hi-res input image:
3 x 640 x 480
with region proposal

Hi-res conv features:
512 x 20 x 15;
Projected region proposal is e.g.
512 x 18 x 8
(varies per proposal)

RoI conv features:
512 x 7 x 7
for region proposal

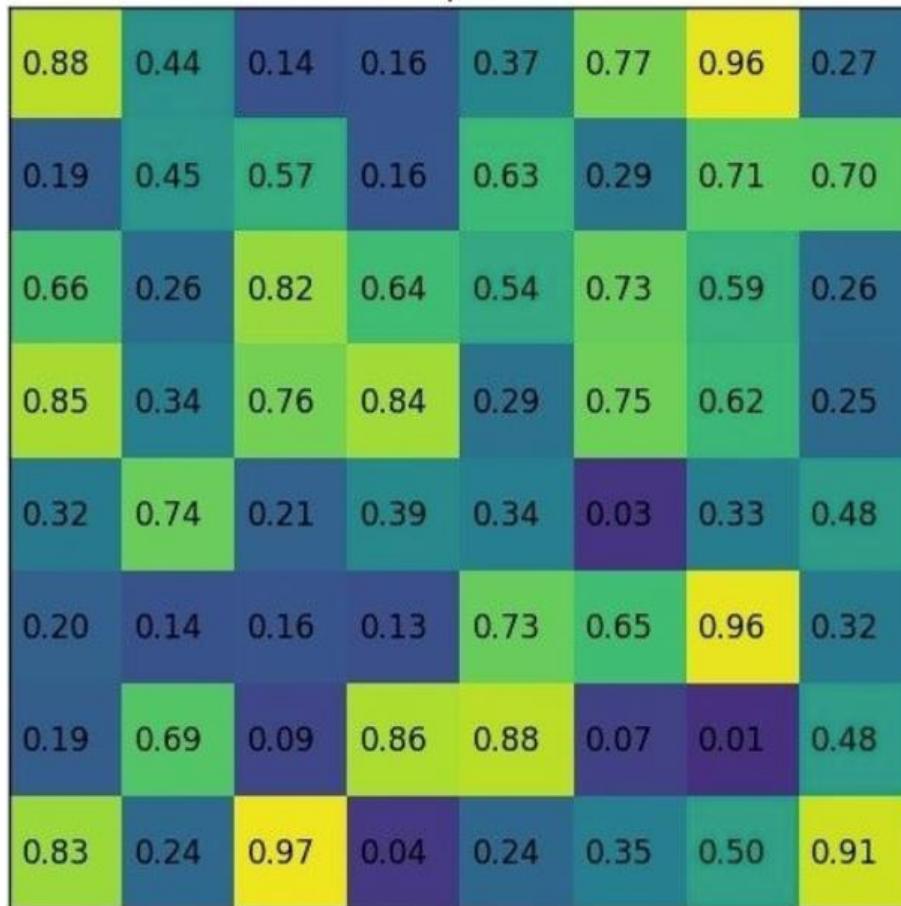
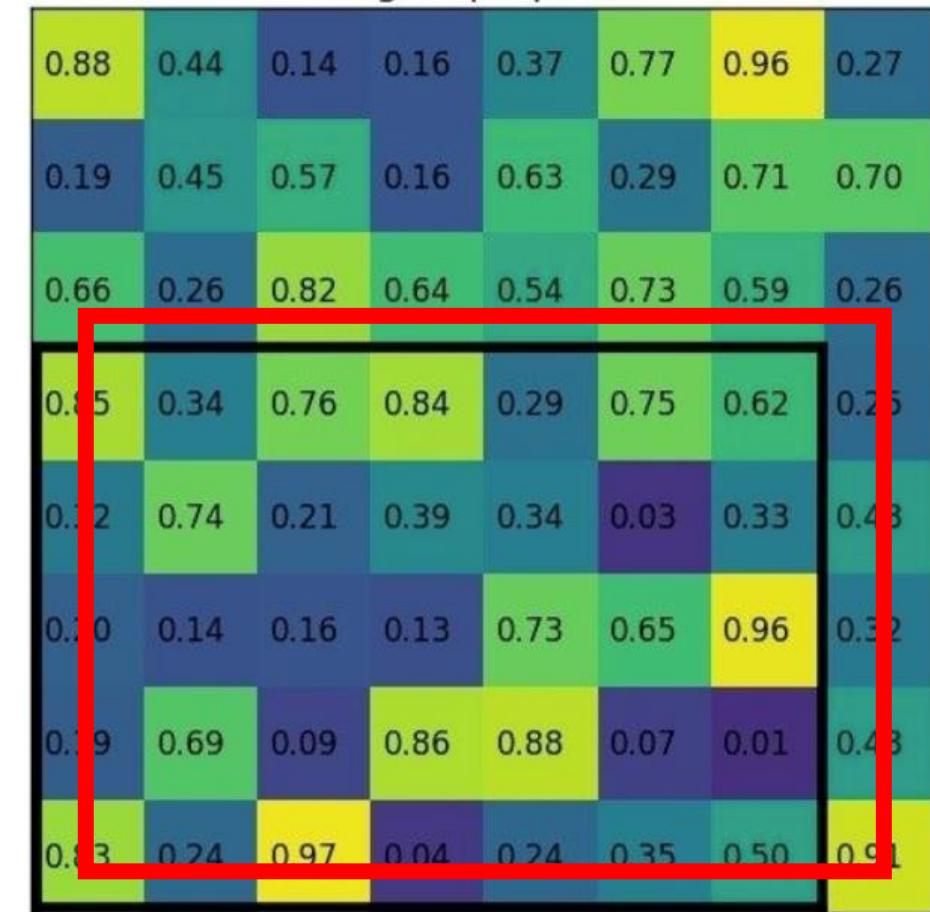
Fully-connected layers expect
low-res conv features:
512 x 7 x 7

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

Source: CS231n, Stanford 2018, Fei-Fei Li, et. al.
Fast R-CNN, R Girshick, et.al.

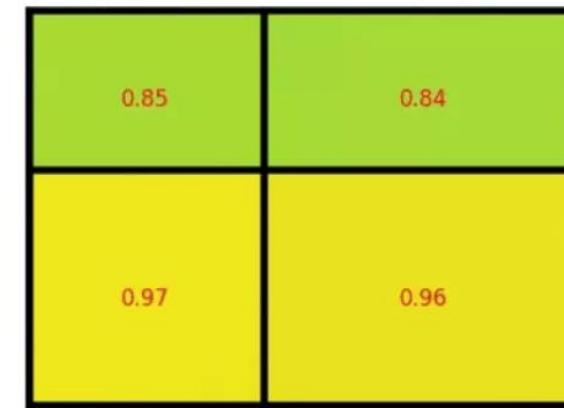
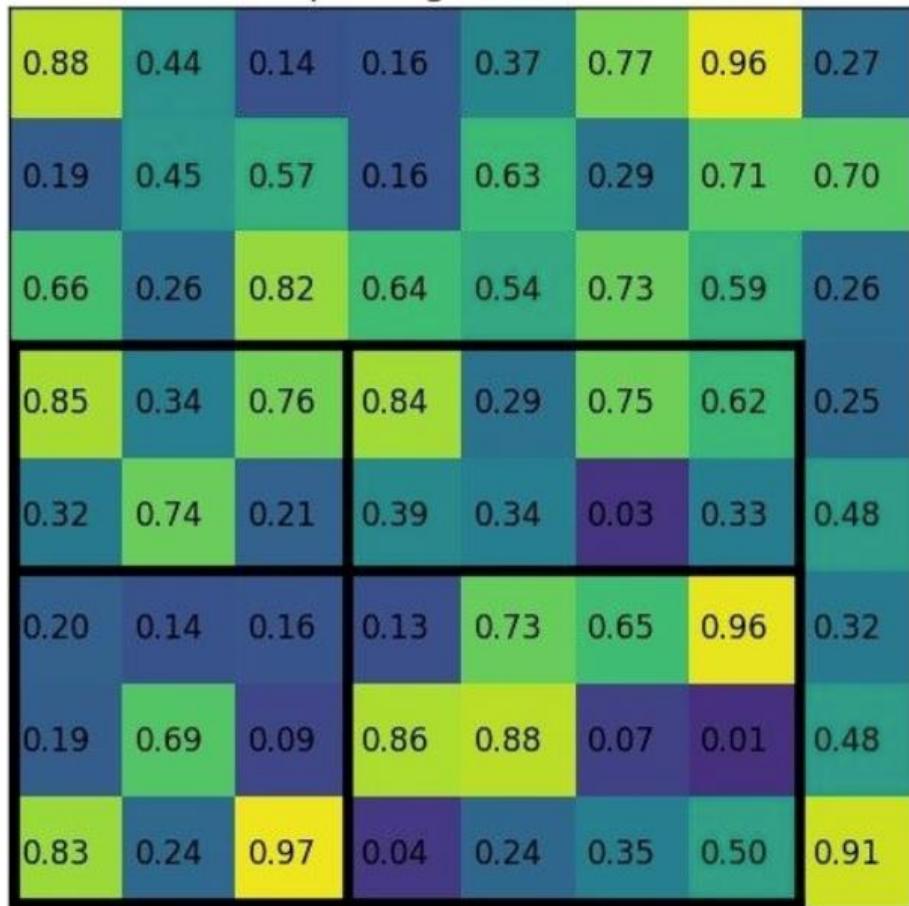


Feature Map

ROI at Feature Map (**Before** and **after** floor operation)



• The ROI pooling outputs size 2x2

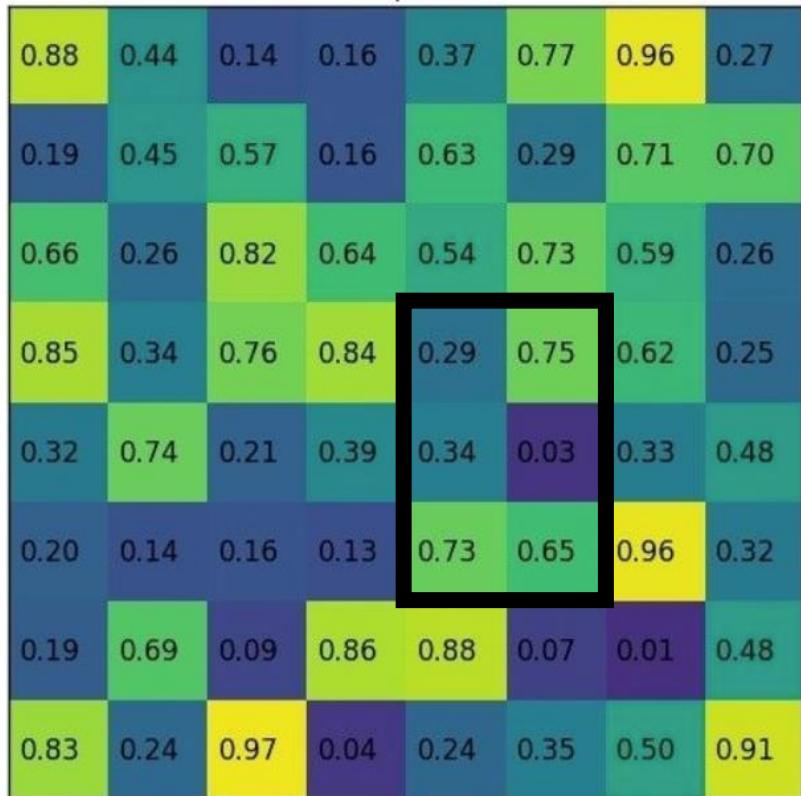




What if ROI is smaller than output?

• Similar to image resize

- Interpolation by nearest neighbor



$3 \times 2 \rightarrow 4 \times 4$

$$\text{Ratio: } \left(\frac{3}{4}, \frac{2}{4} \right) = (0.75, 0.5)$$

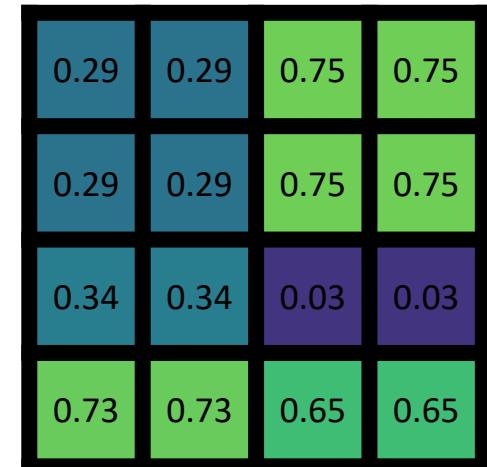
Coordinate (a, b) in output \rightarrow
coordinate range on feature map ROI:

$$\begin{aligned} i: & [0.75a, 0.75(a+1)] \\ j: & [0.50b, 0.50(b+1)] \end{aligned}$$

Floor operation

$$\begin{aligned} i: & [\lfloor 0.75a \rfloor, \lfloor 0.75(a+1) \rfloor] \\ j: & [\lfloor 0.50b \rfloor, \lfloor 0.50(b+1) \rfloor] \end{aligned}$$

Maxpool for the range





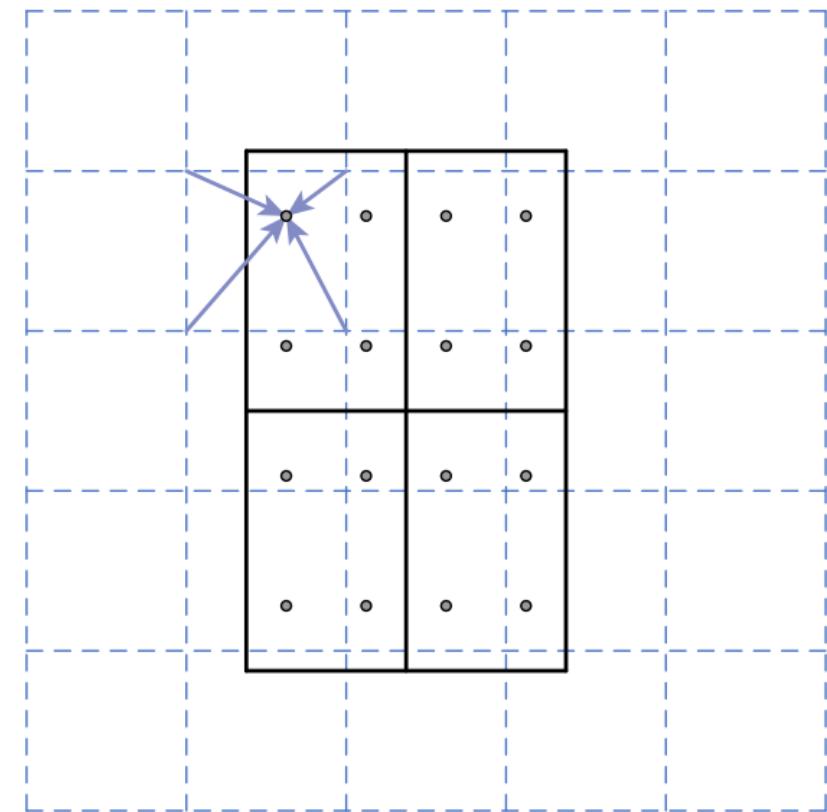
- Image resize: interpolation is better than nearest neighbor
- Same for feature map resize: ROI Pooling → ROI Align

Dash grid: 5×5 feature map

Solid Rectangle: $H \times W$ ROI at feature map

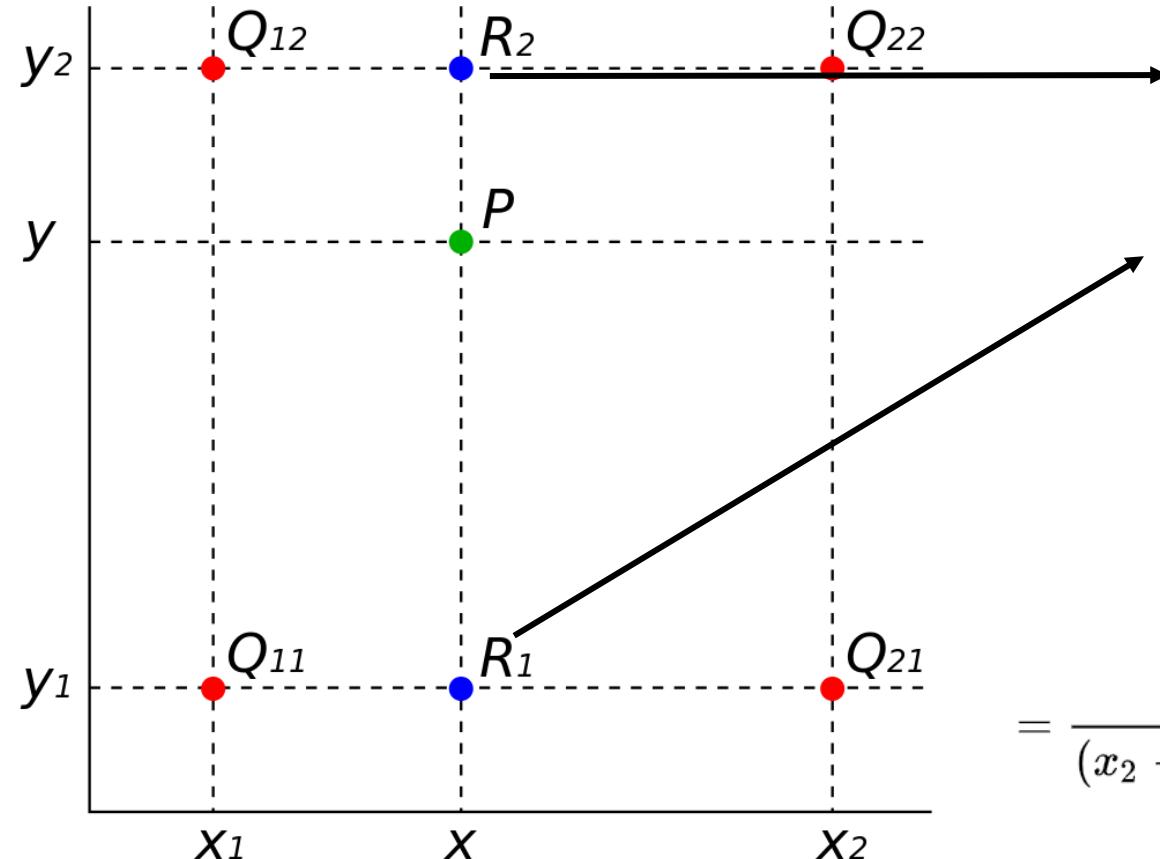
Output: 2×2 solid grid

1. Each cell in 2×2 output is sampled with 4 points
2. Each point is bilinear interpolation into dash grid
3. Each cell is max/avg pool of the 4 points





ROI Align – Bilinear Interpolation



$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

$$= \frac{1}{(x_2 - x_1)(y_2 - y_1)} [x_2 - x \quad x - x_1] \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}$$



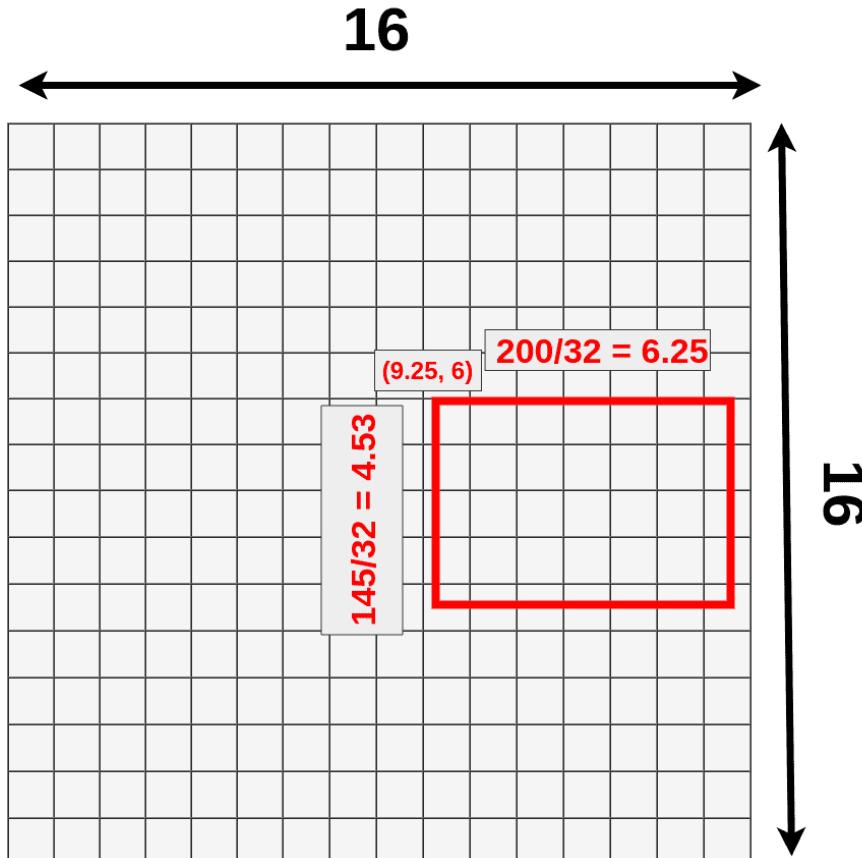
ROI Align Example



Image source: <https://towardsdatascience.com/understanding-region-of-interest-part-2-roi-align-and-roi-warp-f795196fc193>



ROI Align Example

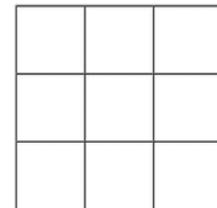


• ROI Align into 3x3

- Object size in image = 145x200
- Scale factor = 32 (ratio between feature map and image)



3x3 ROI Pooling





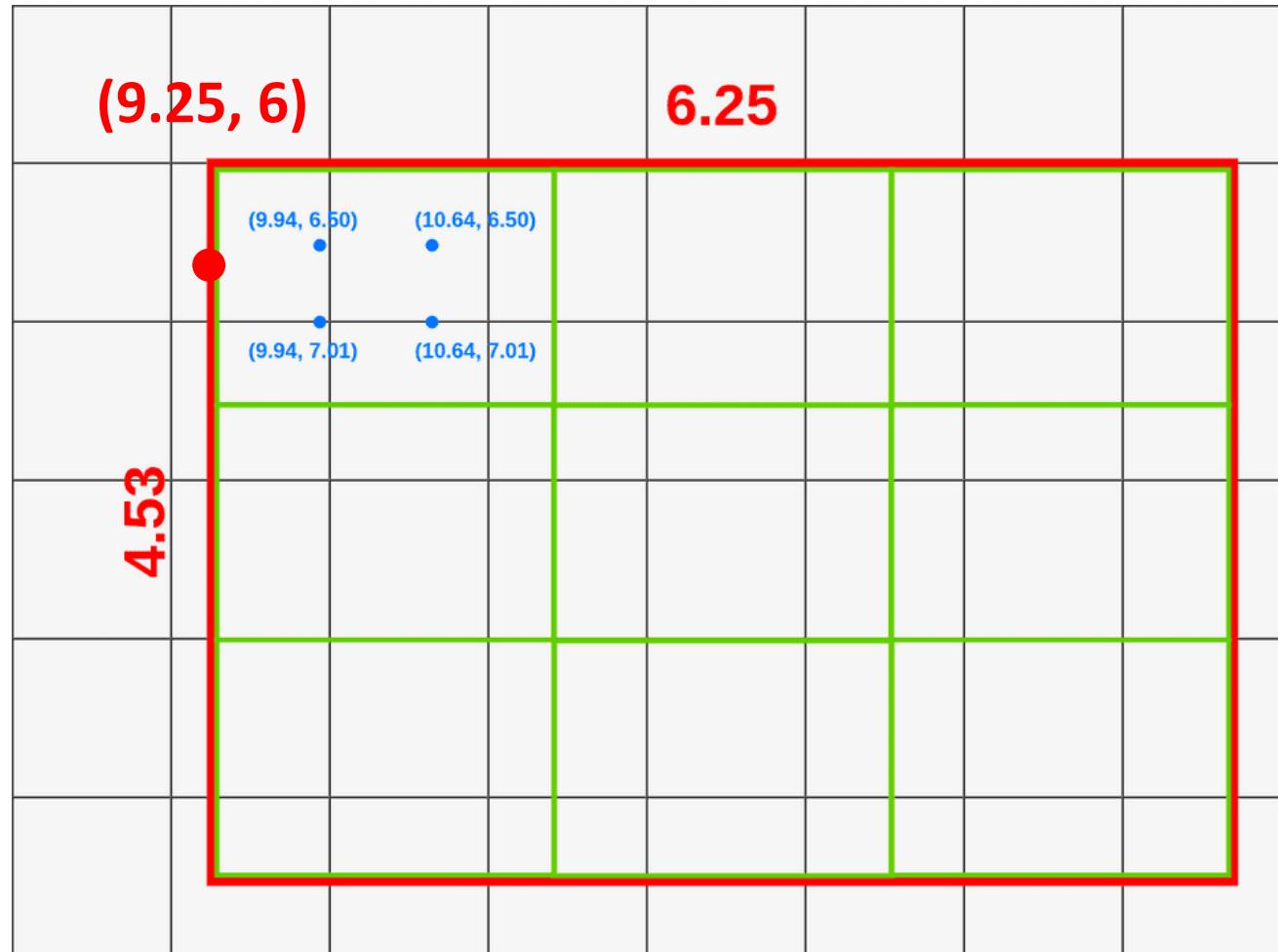
ROI Align Example

Each cell

- Width = $6.25 / 3 = 2.0833$
- Height = $4.53 / 3 = 1.51$

Sample 4 points in each cell.

- $9.25 + (2.0833 / 3) * 1 = 9.94$
- $6 + (1.51 / 3) * 1 = 6.50$
-
- $9.25 + (2.0833 / 3) * 2 = 10.64$
- $6 + (1.51 / 3) * 3 = 7.01$
-



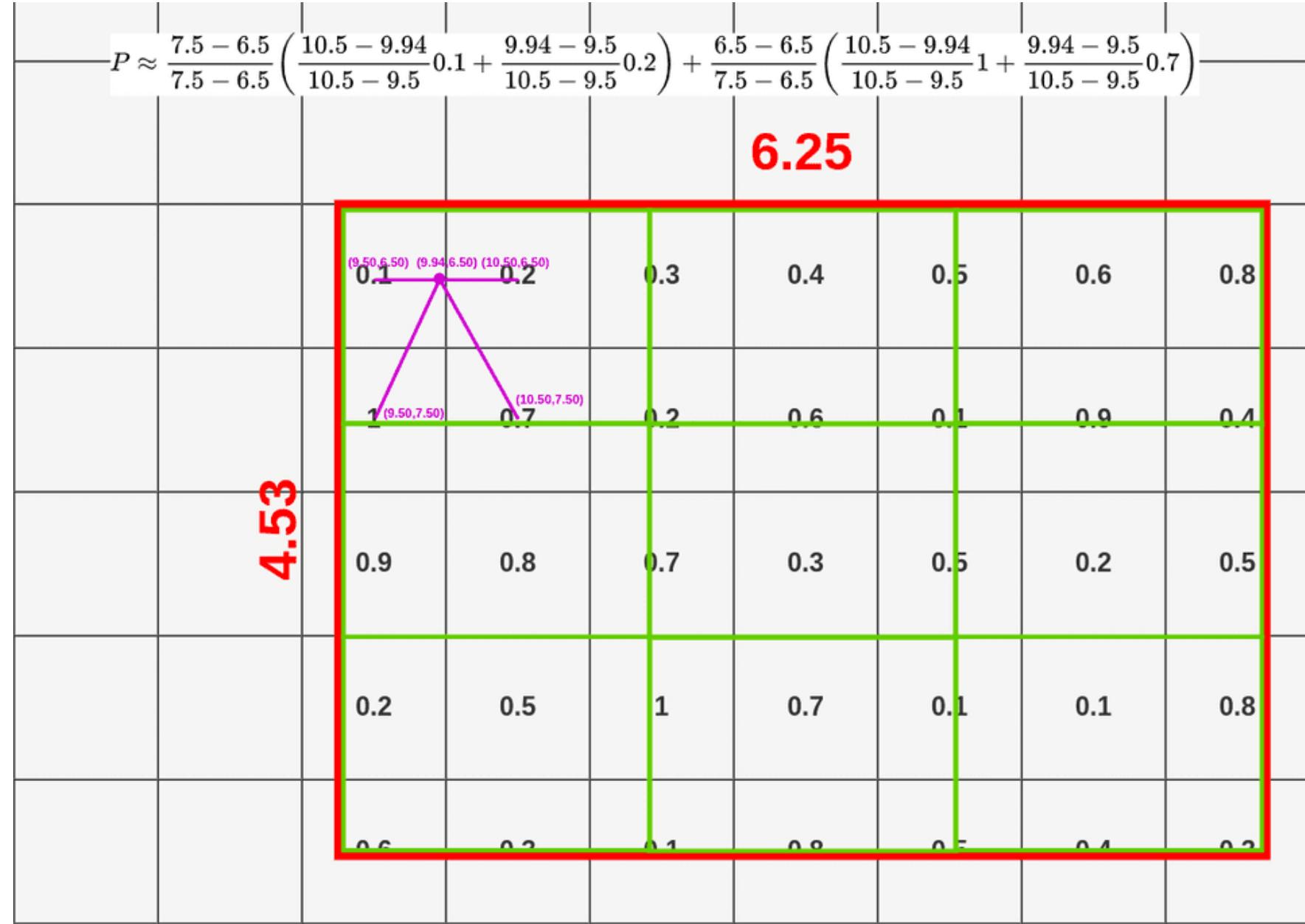


• Interpolation for each sampled point.

$$P \approx \frac{7.5 - 6.5}{7.5 - 6.5} \left(\frac{10.5 - 9.94}{10.5 - 9.5} 0.1 + \frac{9.94 - 9.5}{10.5 - 9.5} 0.2 \right) + \frac{6.5 - 6.5}{7.5 - 6.5} \left(\frac{10.5 - 9.94}{10.5 - 9.5} 1 + \frac{9.94 - 9.5}{10.5 - 9.5} 0.7 \right)$$

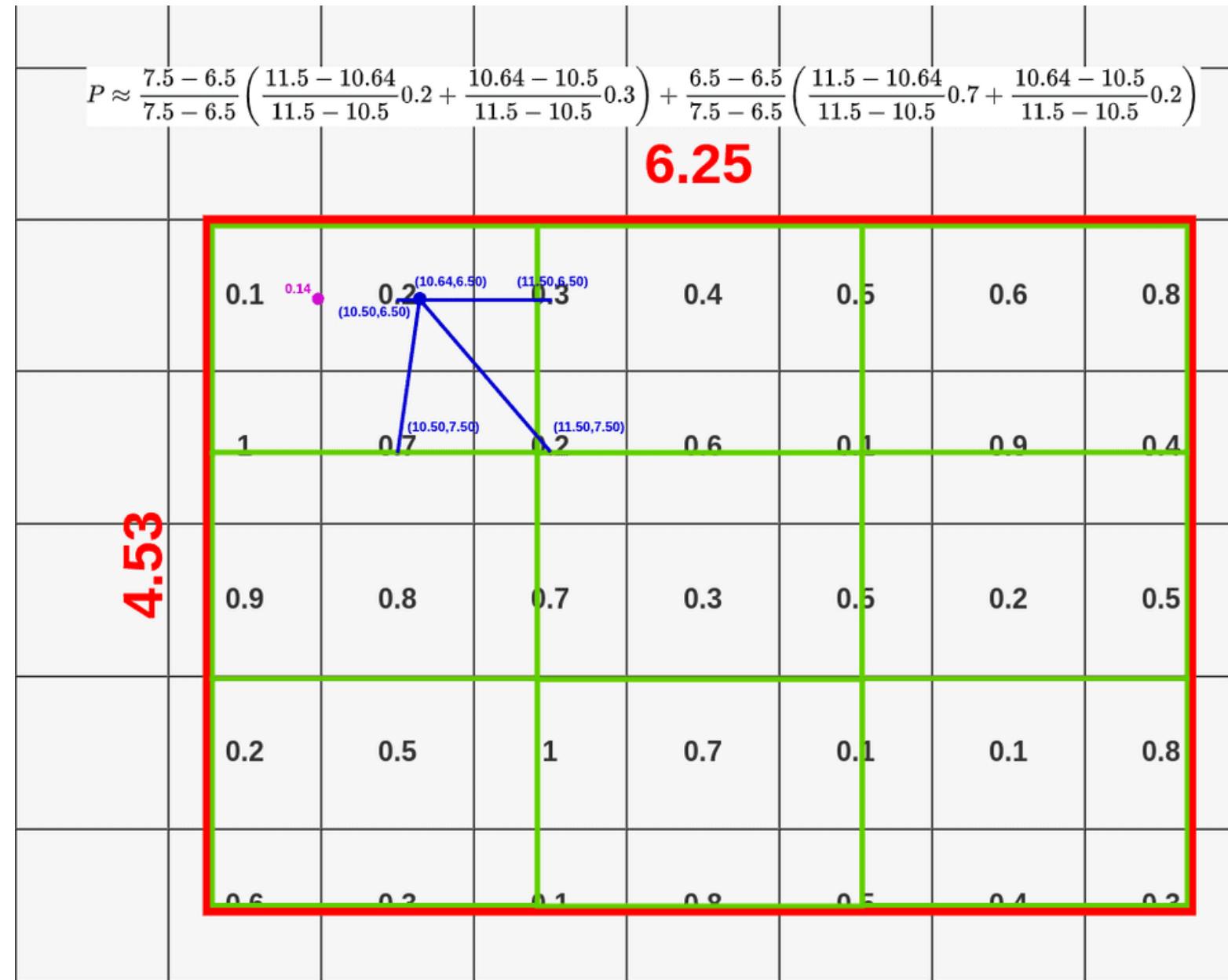
6.25

4.53





- Interpolation for each sampled point.



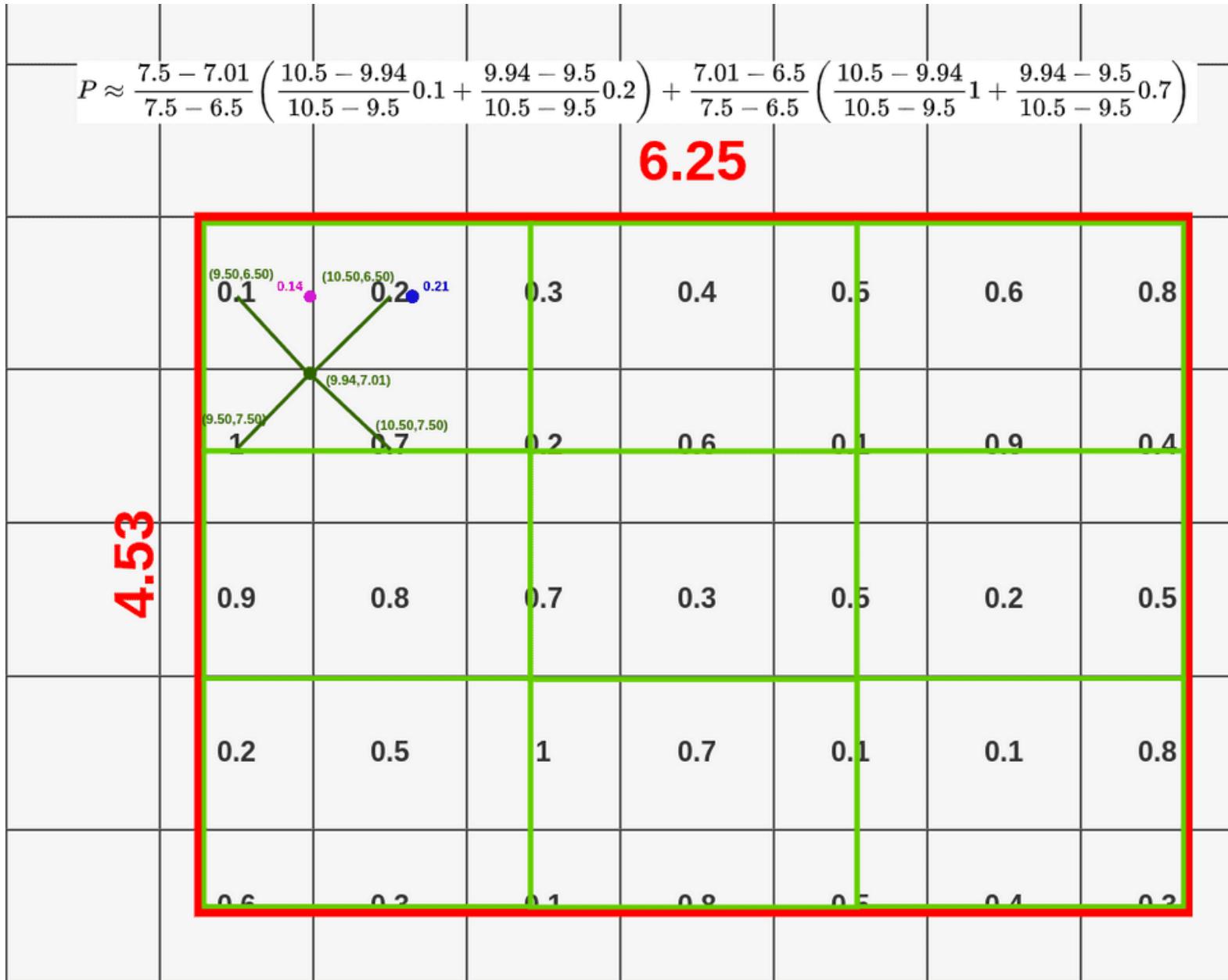


- Interpolation for each sampled point.

$$P \approx \frac{7.5 - 7.01}{7.5 - 6.5} \left(\frac{10.5 - 9.94}{10.5 - 9.5} 0.1 + \frac{9.94 - 9.5}{10.5 - 9.5} 0.2 \right) + \frac{7.01 - 6.5}{7.5 - 6.5} \left(\frac{10.5 - 9.94}{10.5 - 9.5} 1 + \frac{9.94 - 9.5}{10.5 - 9.5} 0.7 \right)$$

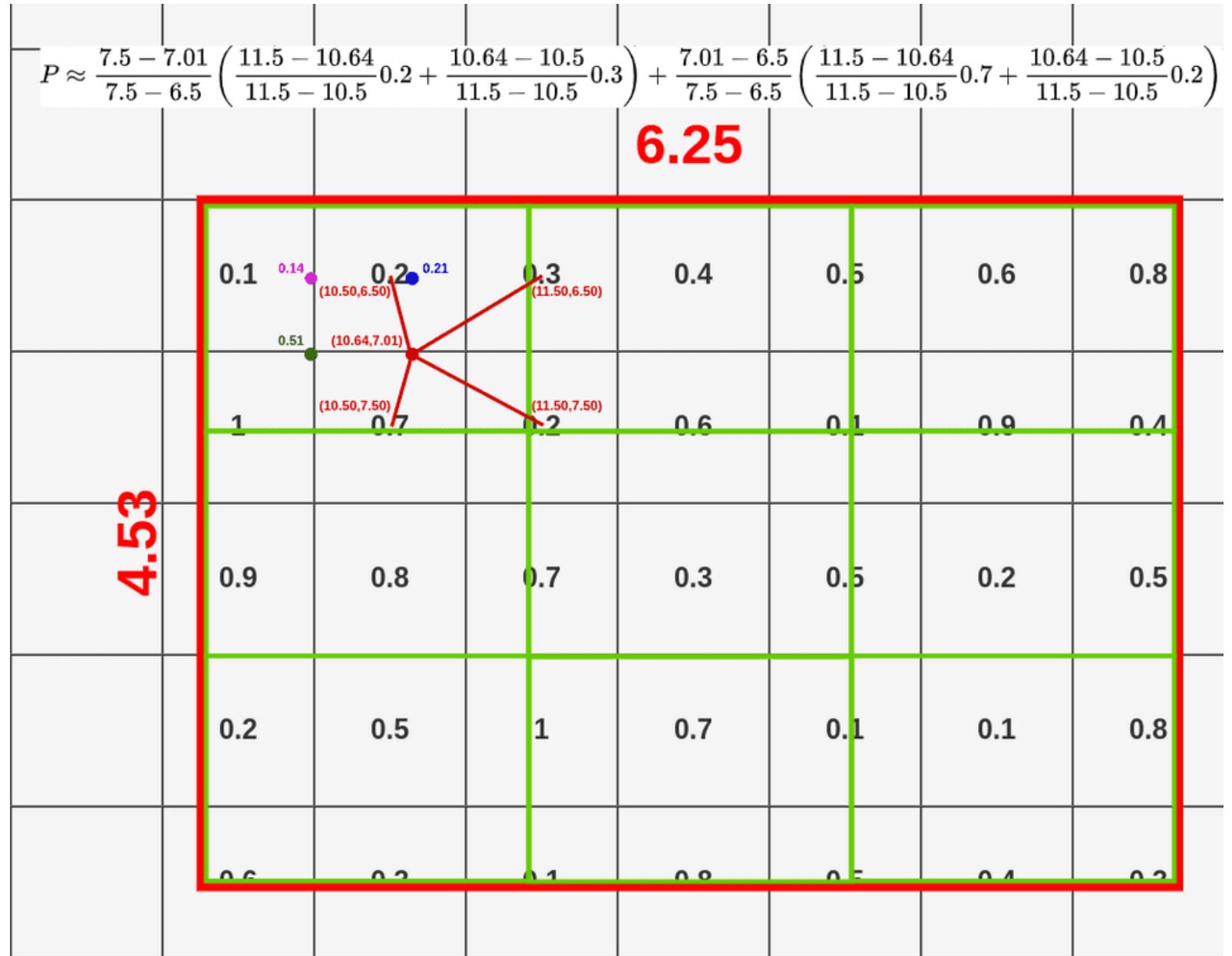
6.25

4.53



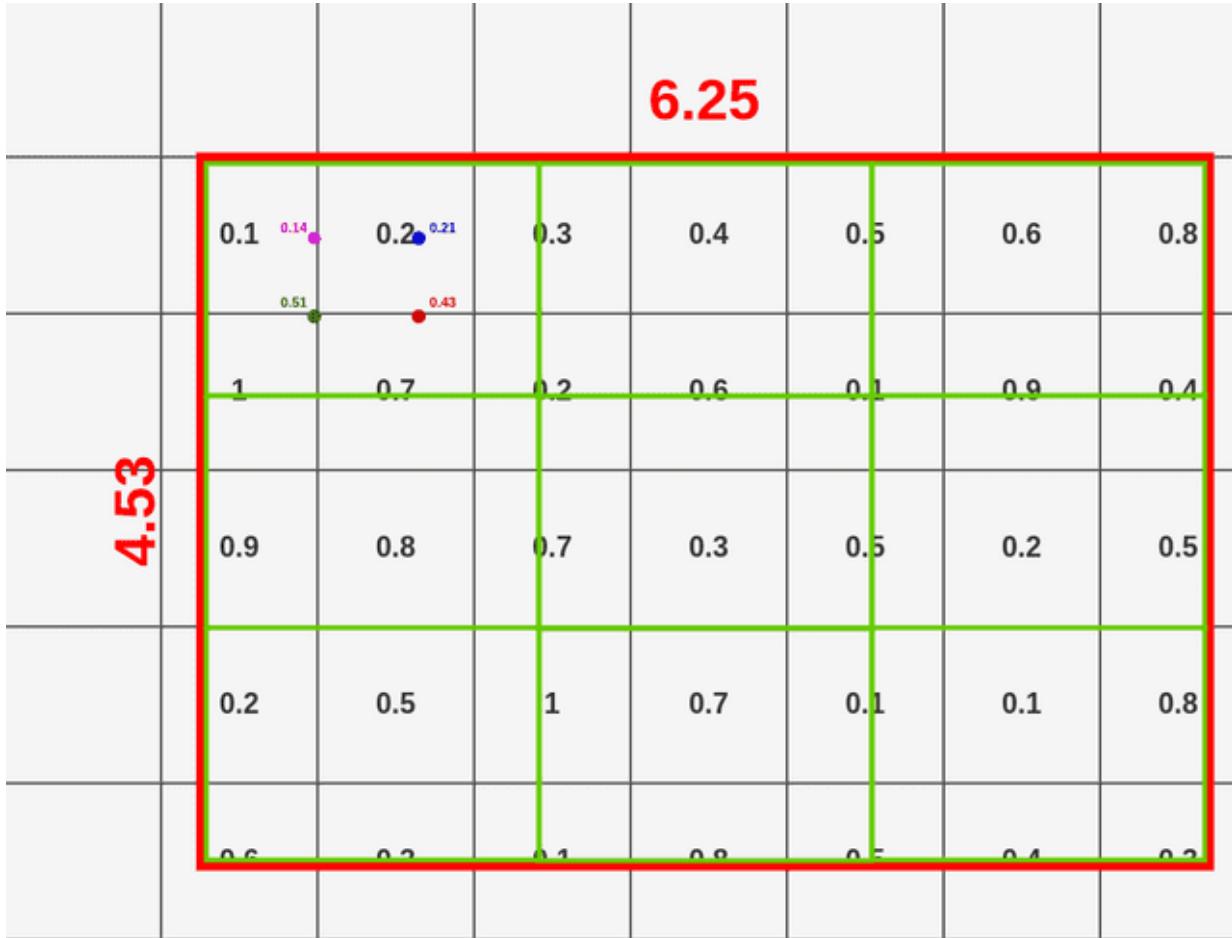


• Interpolation for each sampled point.





- Max/Average pooling of the 4 sampled points to get a cell



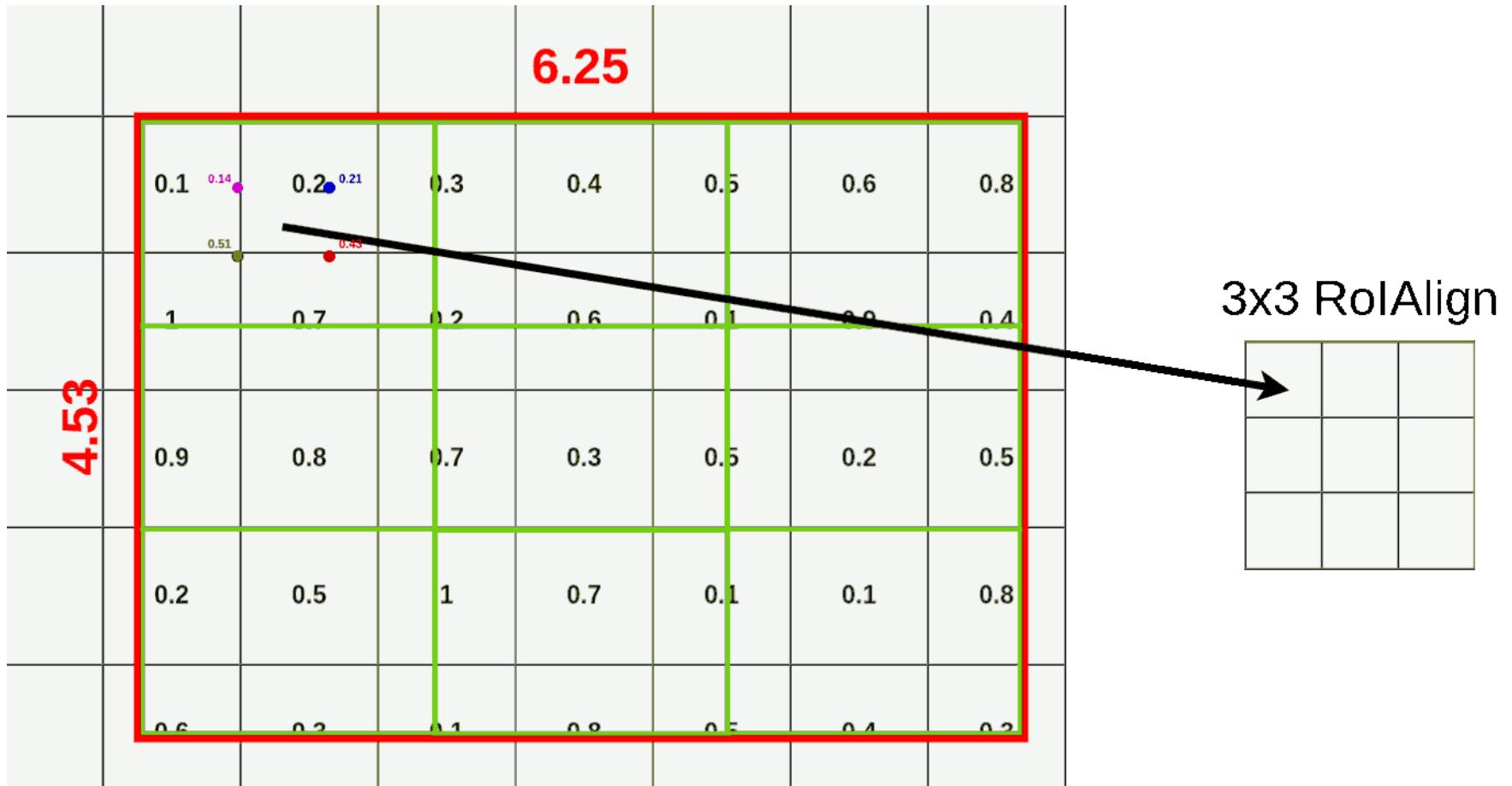
$1 \times 1 = \text{MAX}(0.14, 0.21, 0.51, 0.43) = 0.51$

3x3 RoIAlign

0.51		

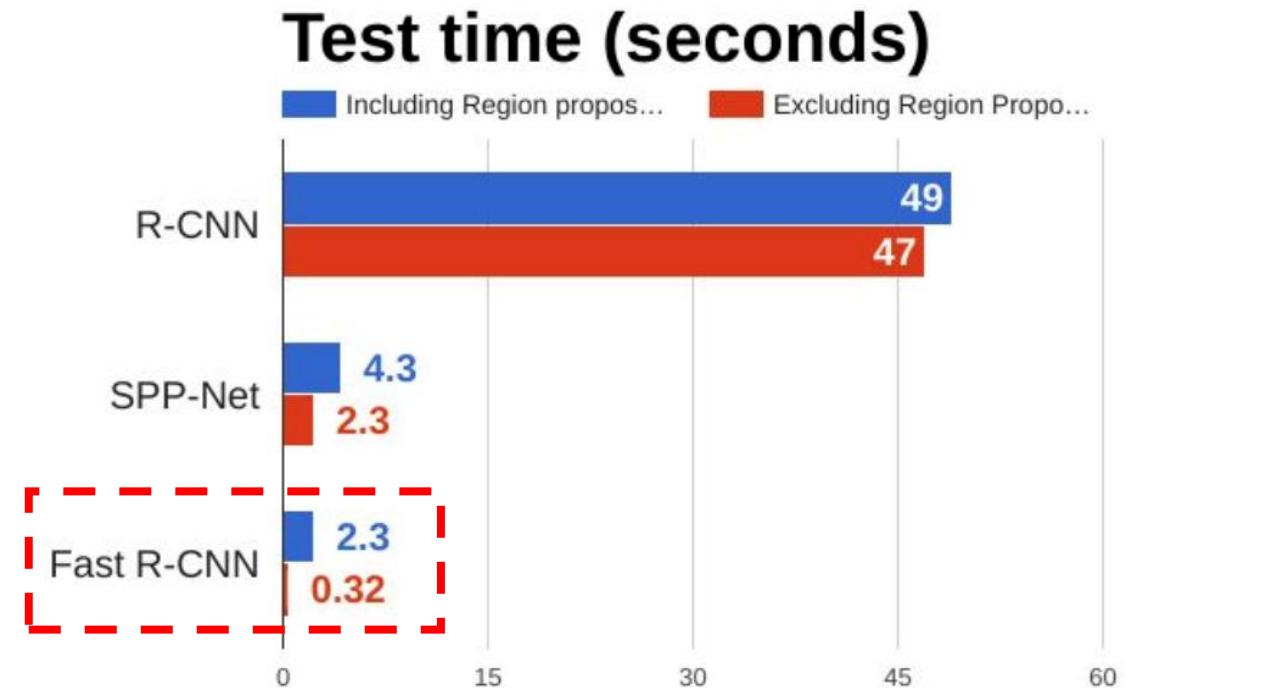
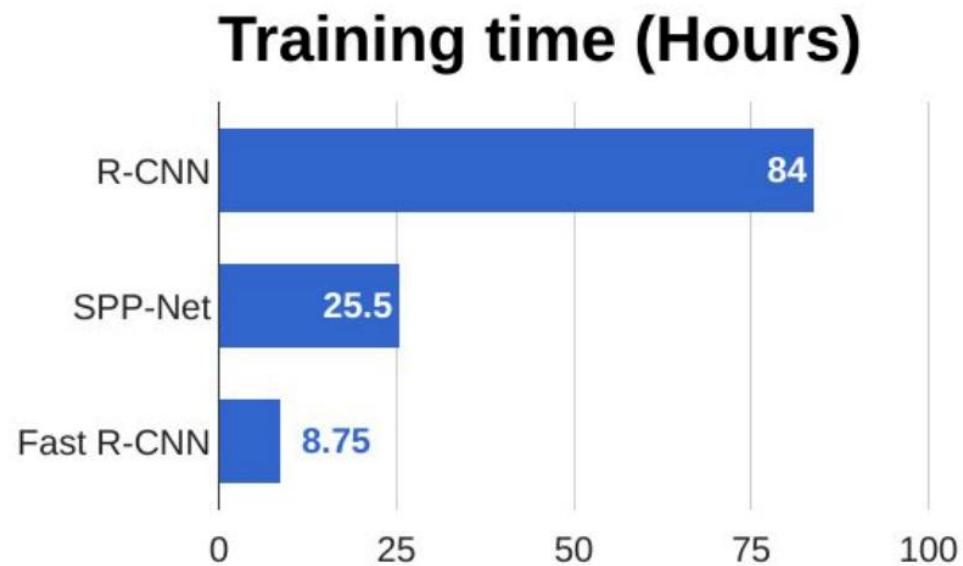


ROI Align Example





Fast RCNN is fast compared to RCNN, because of the shared feature extraction.



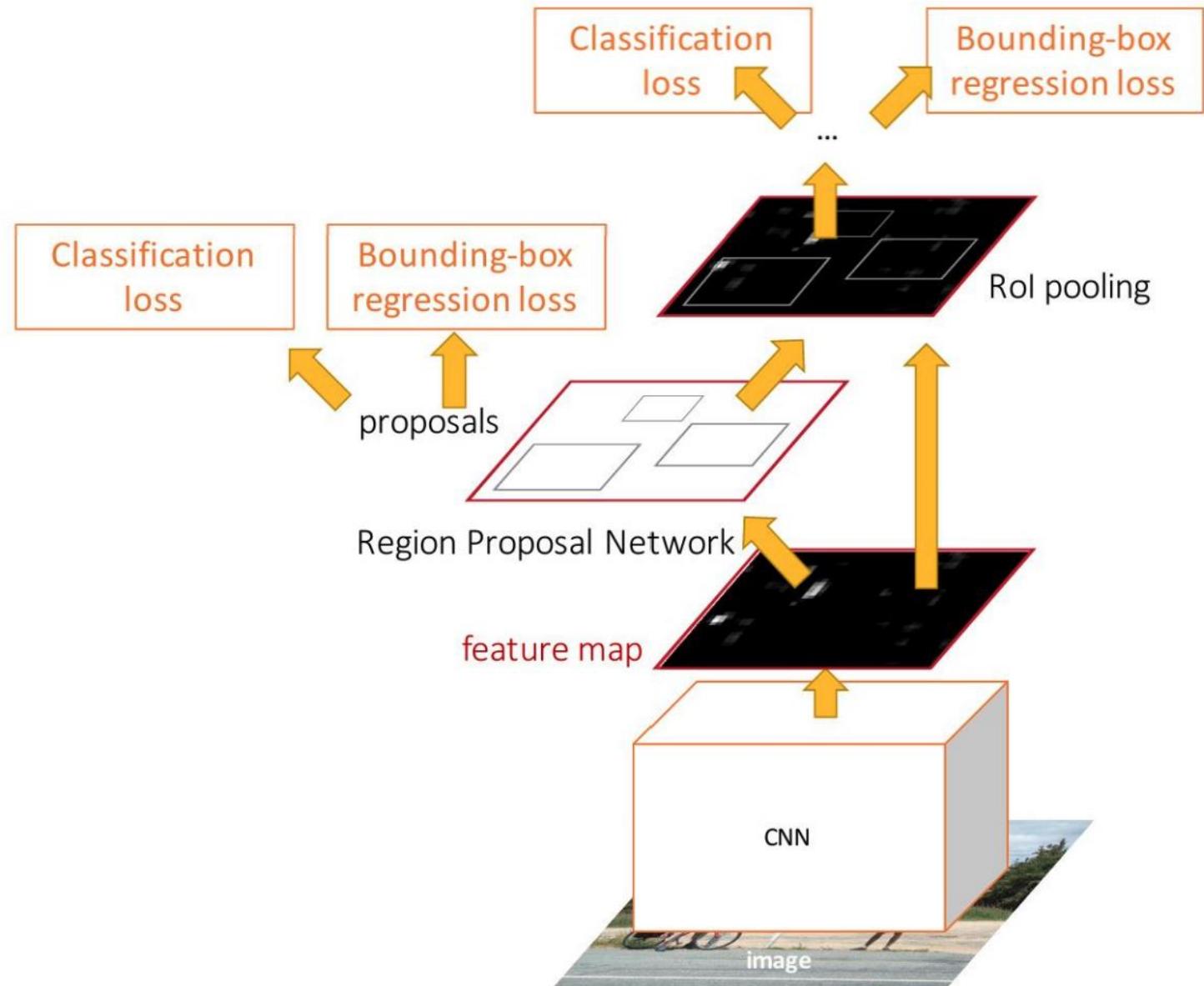
Runtime dominated by non-deep learning region proposal



Region Proposal Network

(RPN)

- Input:
 - Feature map
 - Anchors
- Output:
 - Object proposals (x, y, h, w)

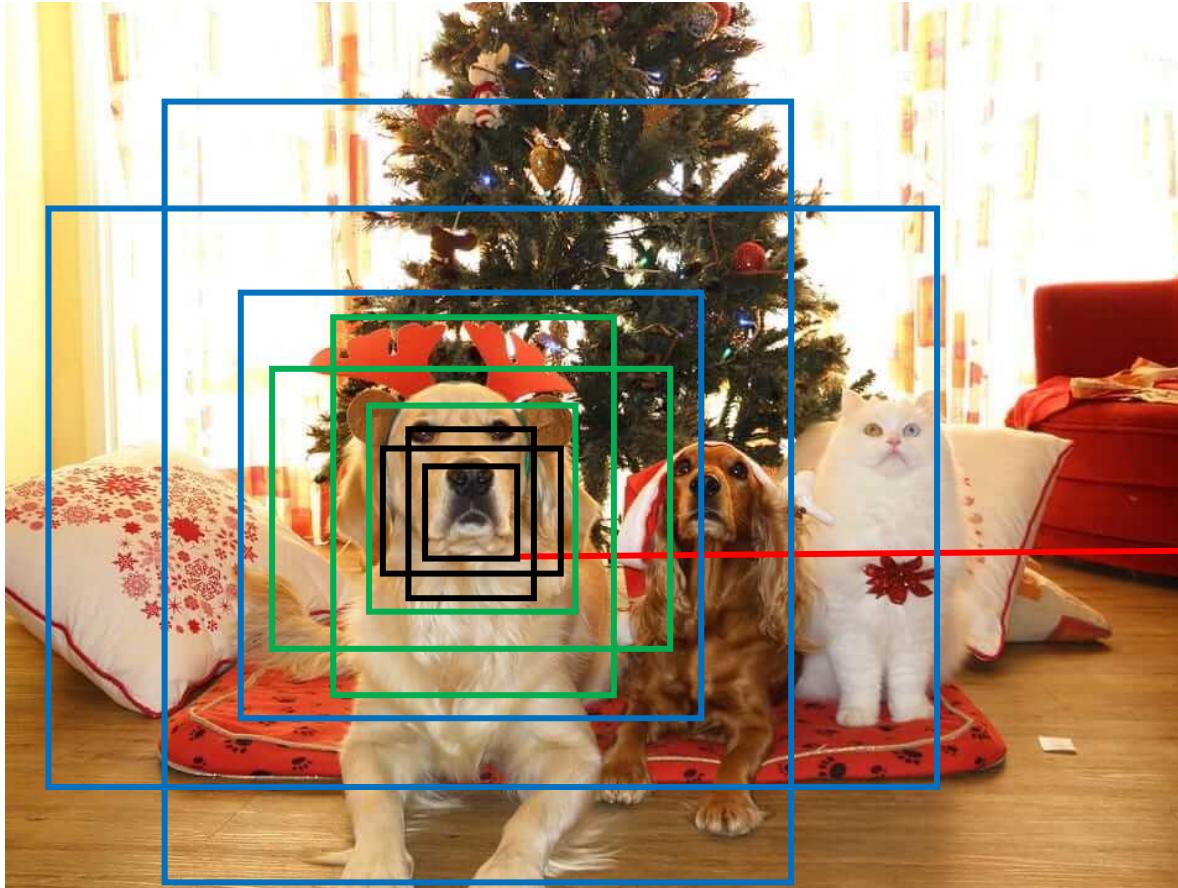


Source: CS231n, Stanford 2018, Fei-Fei Li, et. al.

Faster R-CNN: Towards real-time object detection with region proposal networks. Ren Shaoqing, et.al.



◆ Anchors → Proposals



- ◆ 1 position in feature map
- ◆ k anchors in image

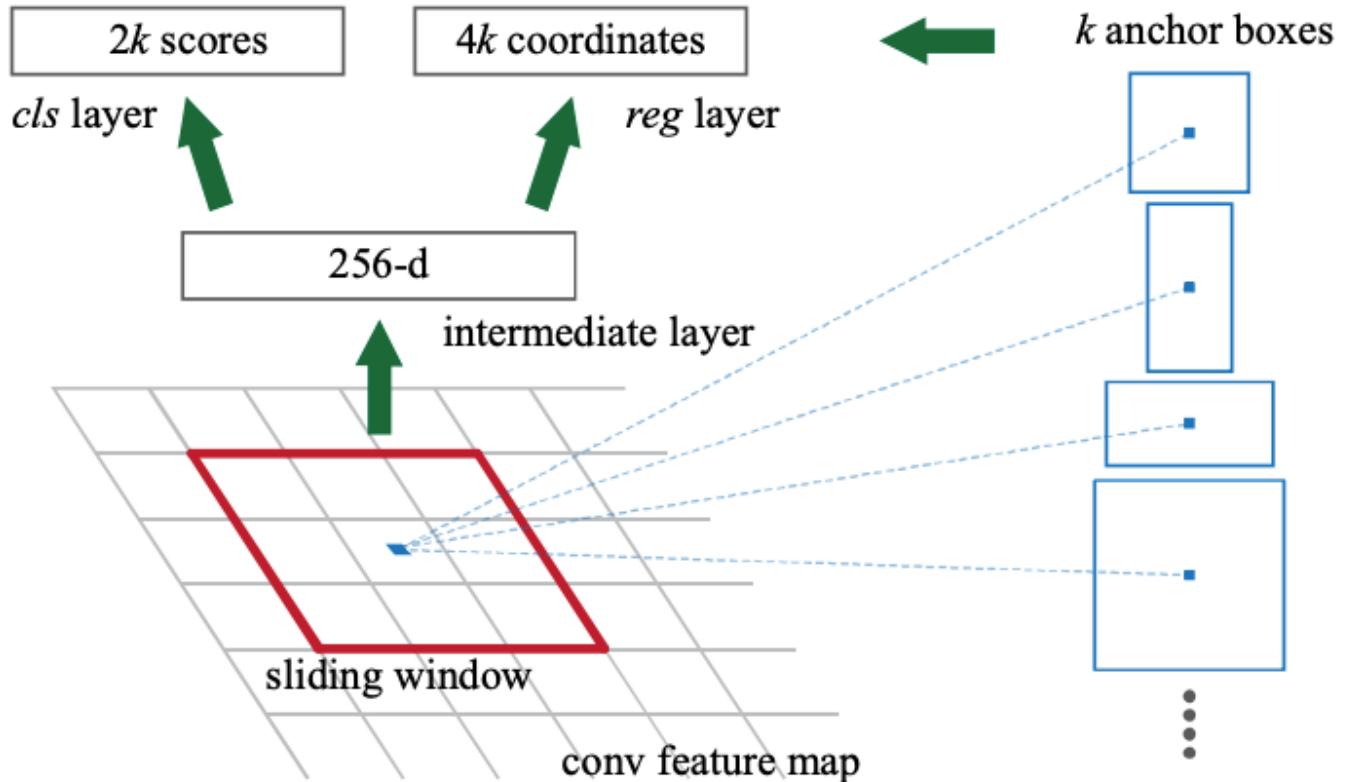
0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.22	0.7	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91



RPN determines

- Object or NOT-Object
 - 2k scores
- If yes, $[x, y, h, w]$ of the object box
 - 4k coordinates

Doesn't care about categories of the objects



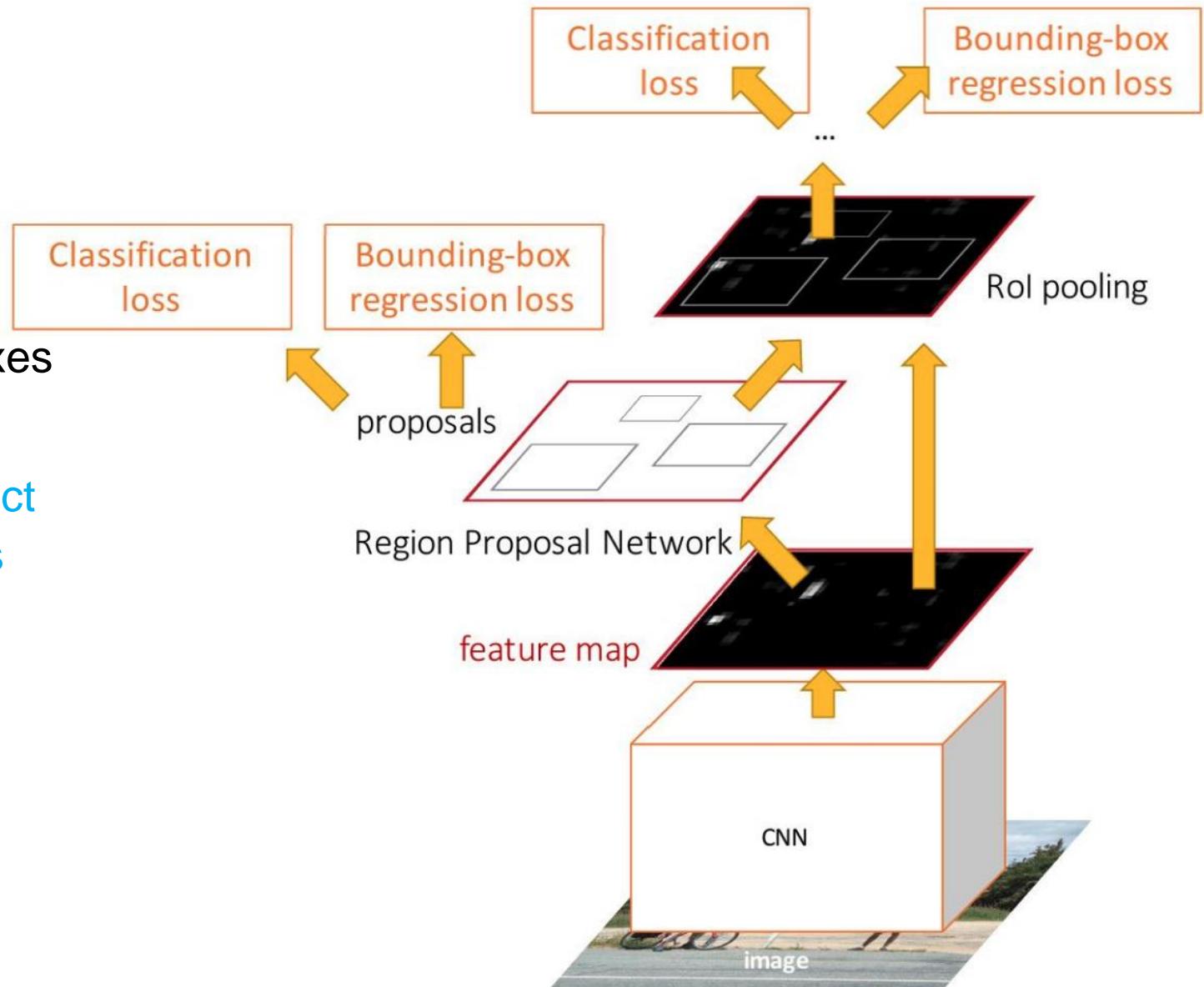


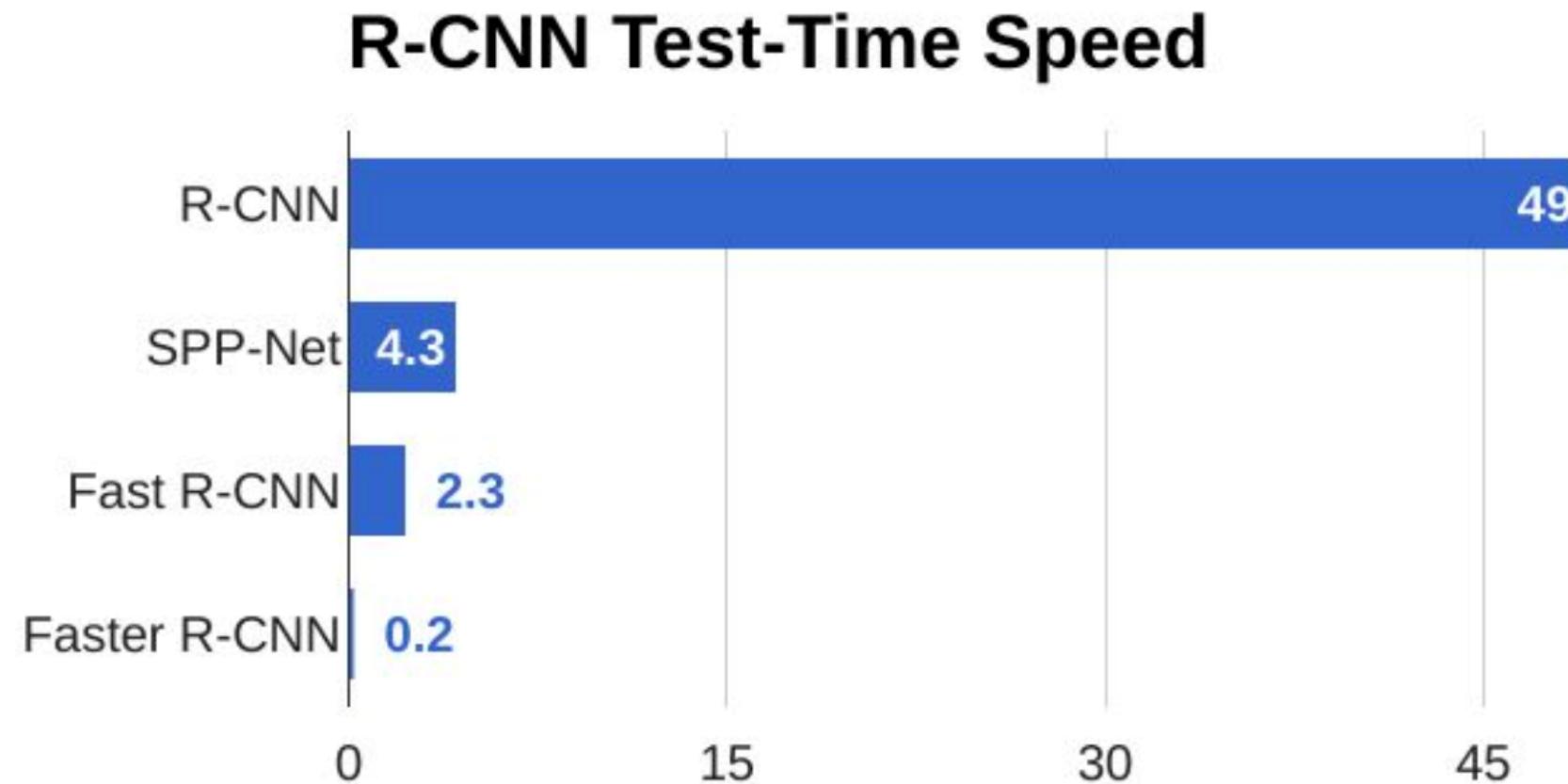
Steps

1. CNN → Feature Map
2. RPN → Proposals
3. ROI Pooling
4. CNN + MLP → bounding boxes

Losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final box classification
4. Final box coordinates

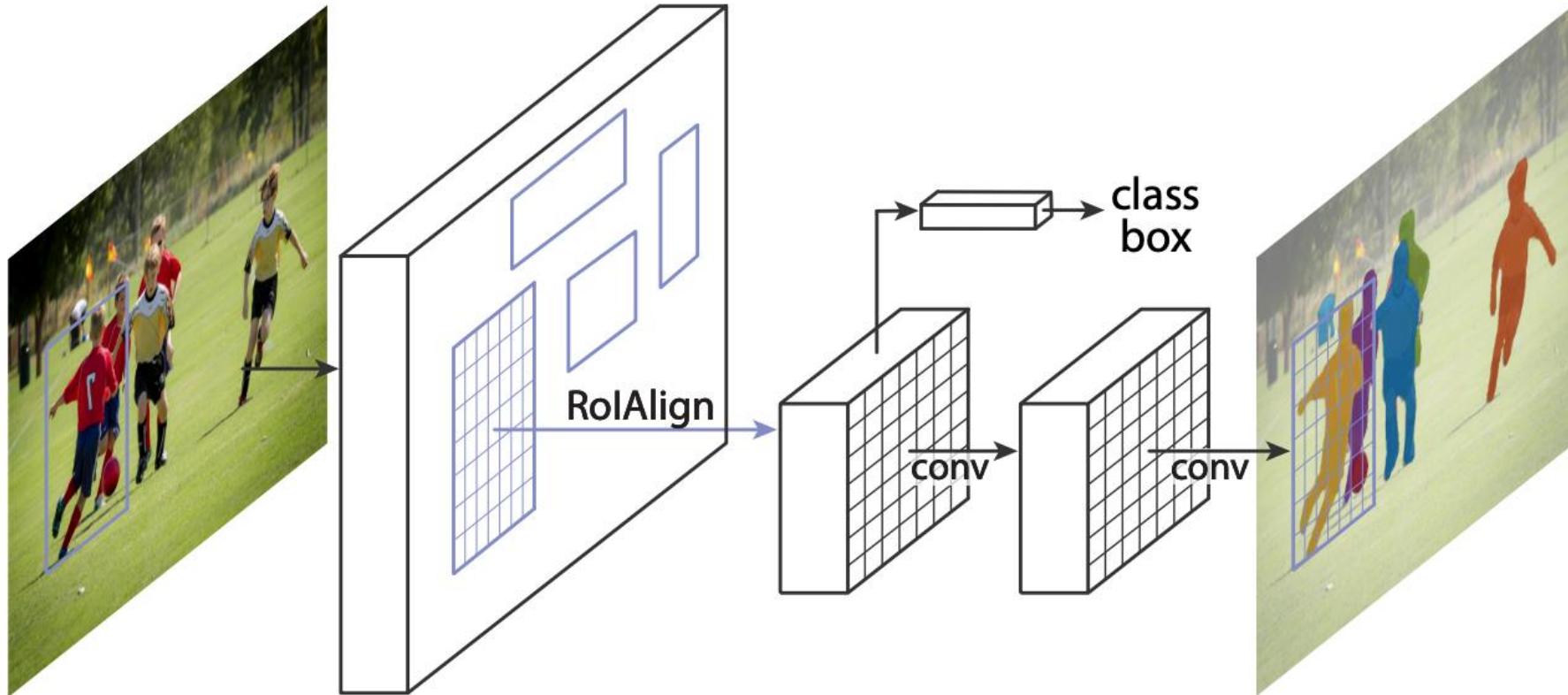






- Input: Image
- Output:
 - Object boxes
 - Per-pixel **mask**

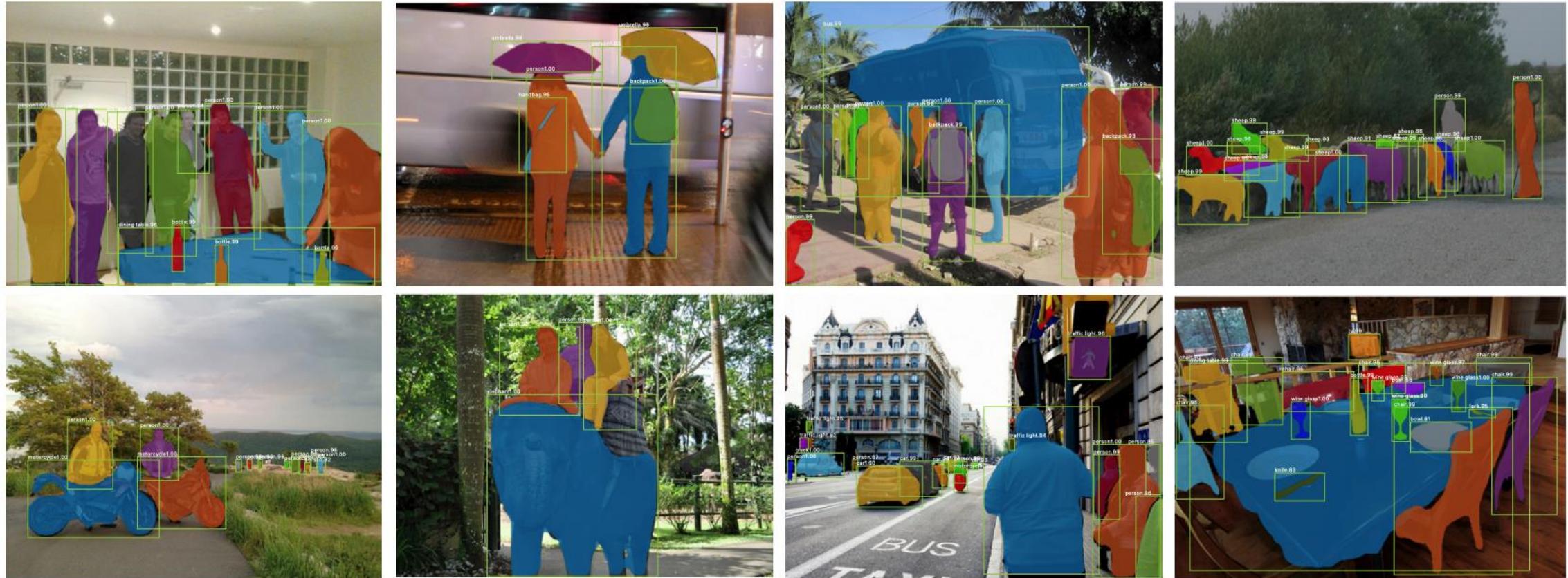
- Two of the improvements over Faster RCNN
 - ROI Align
 - Object detection + Instance Segmentation



Source: Mask R-CNN, Kaiming He, et. al.



Mask RCNN



Source: Mask R-CNN, Kaiming He, et. al.



RCNN Family Summary

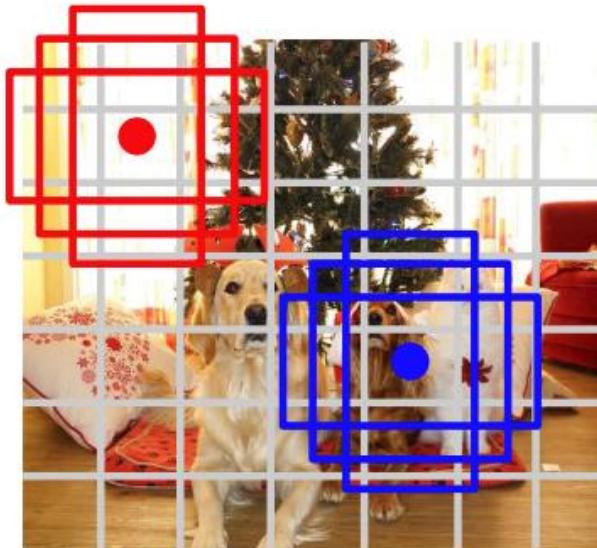
	RCNN	Fast RCNN	Faster RCNN	Mask RCNN
Region Proposal	External	External	RPN	RPN
Region processing	ROI crop on Image	ROI pooling on feature map	ROI pooling on feature map	ROI align on feature map
Multi-task	No	No	No	Instance Seg.



One Stage Detector – No ROI Pooling



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

i.e. 7×7 feature map after CNN

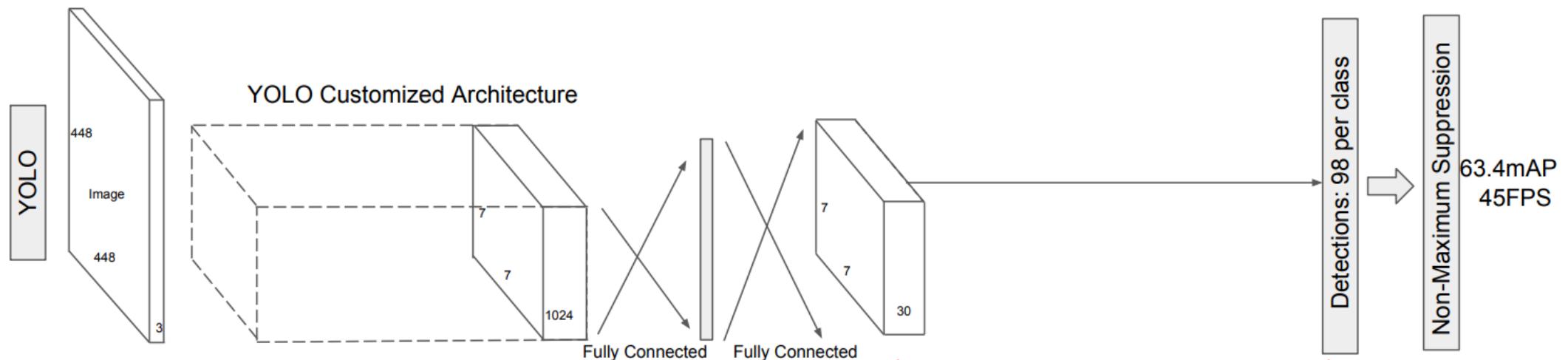
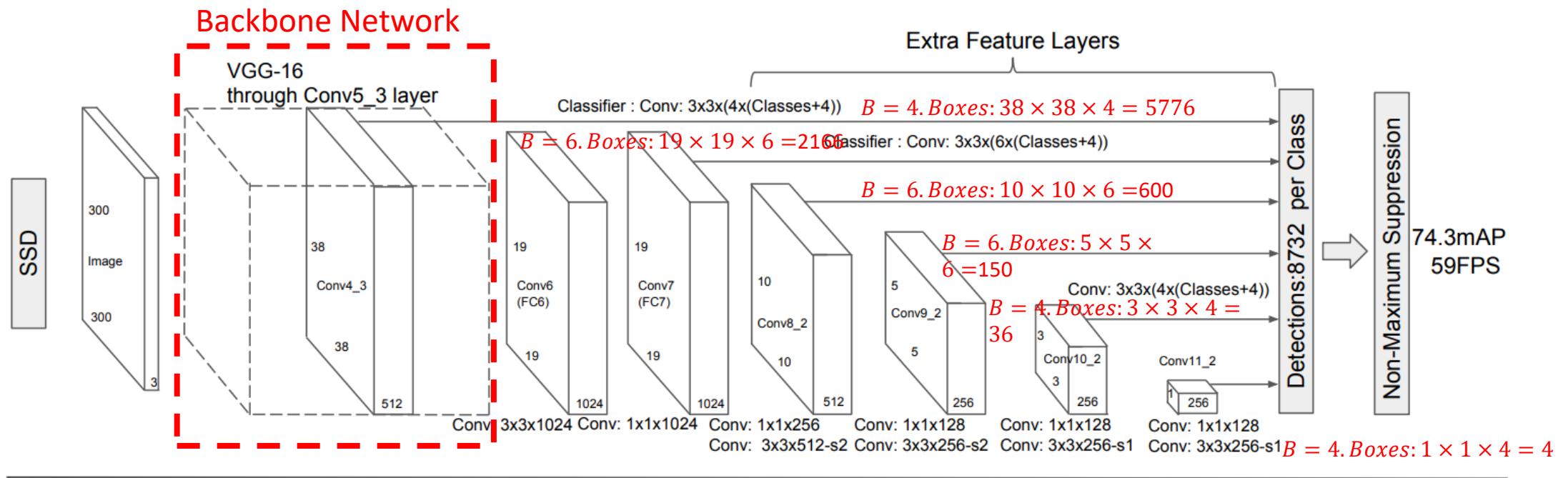
Each cell corresponds to B prior
bounding boxes, here $B = 3$

With each cell:

- Regression of B boxes
 - $dx, dy, dh, dw, \text{confidence}$
 - “d” means relative to prior box
- Classification of this cell: C classes
- Output:
 $7 \times 7 \times (B \times 5 + C)$
or
 $7 \times 7 \times (B \times (4 + C))$



SSD



This is YOLO v1. YOLO v3 is similar to SSD



- Ground truth boxes: $\{x^g, y^g, w^g, h^g, p\}$, p is the label
- Network output: a set of boxes $\{x, y, w, h, \{c_0, c_1, \dots, c_k\}\}$
 - Category 0 is background / not-object
- Loss function contains two parts
 - Classification for both **positive** and **negative** matches, CrossEntropyLoss

$$L_{cls} = \alpha L_{cls}^{pos} + \beta L_{cls}^{neg} = \alpha \sum -\log \frac{\exp c_p}{\sum \exp c_i} + \beta \sum -\log \frac{\exp c_0}{\sum \exp c_i}$$

- Regression for **positive** matches:

$$L_{reg}^{pos} = \sum \rho \left(\frac{x^g - x}{w} \right) + \rho \left(\frac{y^g - y}{h} \right) + \rho \left(\log \frac{w^g}{w} \right) + \rho \left(\log \frac{h^g}{h} \right)$$



❖ Training steps:

- Establish positive matching – which output boxes are associated with ground truth boxes
 - $\text{IoU} > \text{threshold}$ or the largest IoU with one of the ground truth boxes
- Establish negative matching – which output boxes are not matched to any ground truth box
 - $\text{IoU} < \text{threshold}$
- Compute $L = \alpha L_{cls}^{pos} + \beta L_{cls}^{neg} + L_{reg}^{pos}$

❖ Inference steps:

- Pick boxes with high classification scores
- Non-Maximum Suppression (NMS)



- ❖ Two stage methods are usually more accurate, but slower
- ❖ The same method's speed & accuracy can be very different, depending on
 - Network design
 - Training configuration
 - Parameter setting,
 - image/feature map resolution,
 - number of anchor/prior boxes,
 - how to select anchors/prior boxes
 -

	Train Time (s/iter)	Inference Time (fps)	Box AP
Fast RCNN	0.462	10.9	38.9
Faster RCNN	1.04	7.3	41.3
Mask RCNN	1.102	6.5	42.1
SSD	0.412	25.4	29.3



Multi-view Projection

Build image-like feature map with PointNet

- PointNet to convert point cloud detection into image detection + normal 2D CNN
- Two-stage / One-stage

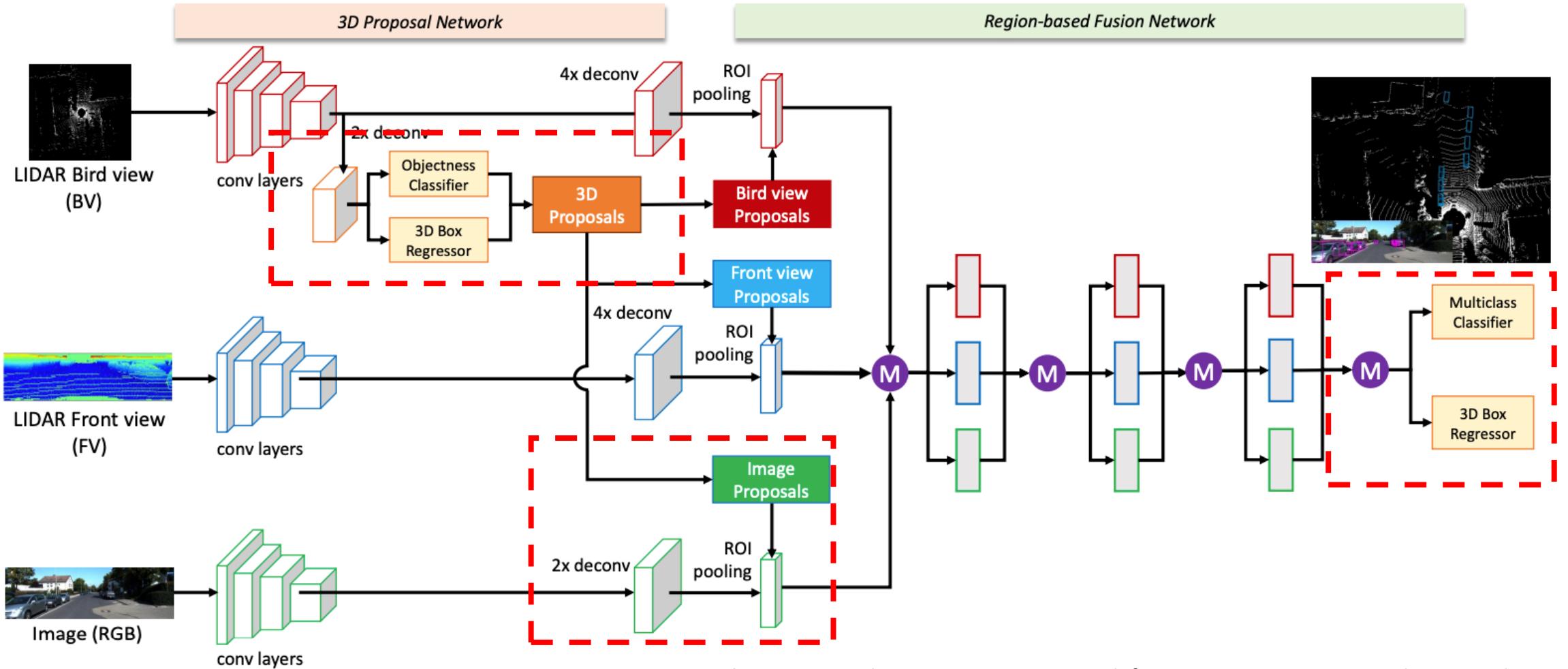
Point-wise operation

- Replace 2D CNN with point-wise operations like PointNet
- Two-stage / One-stage

Point cloud & image fusion



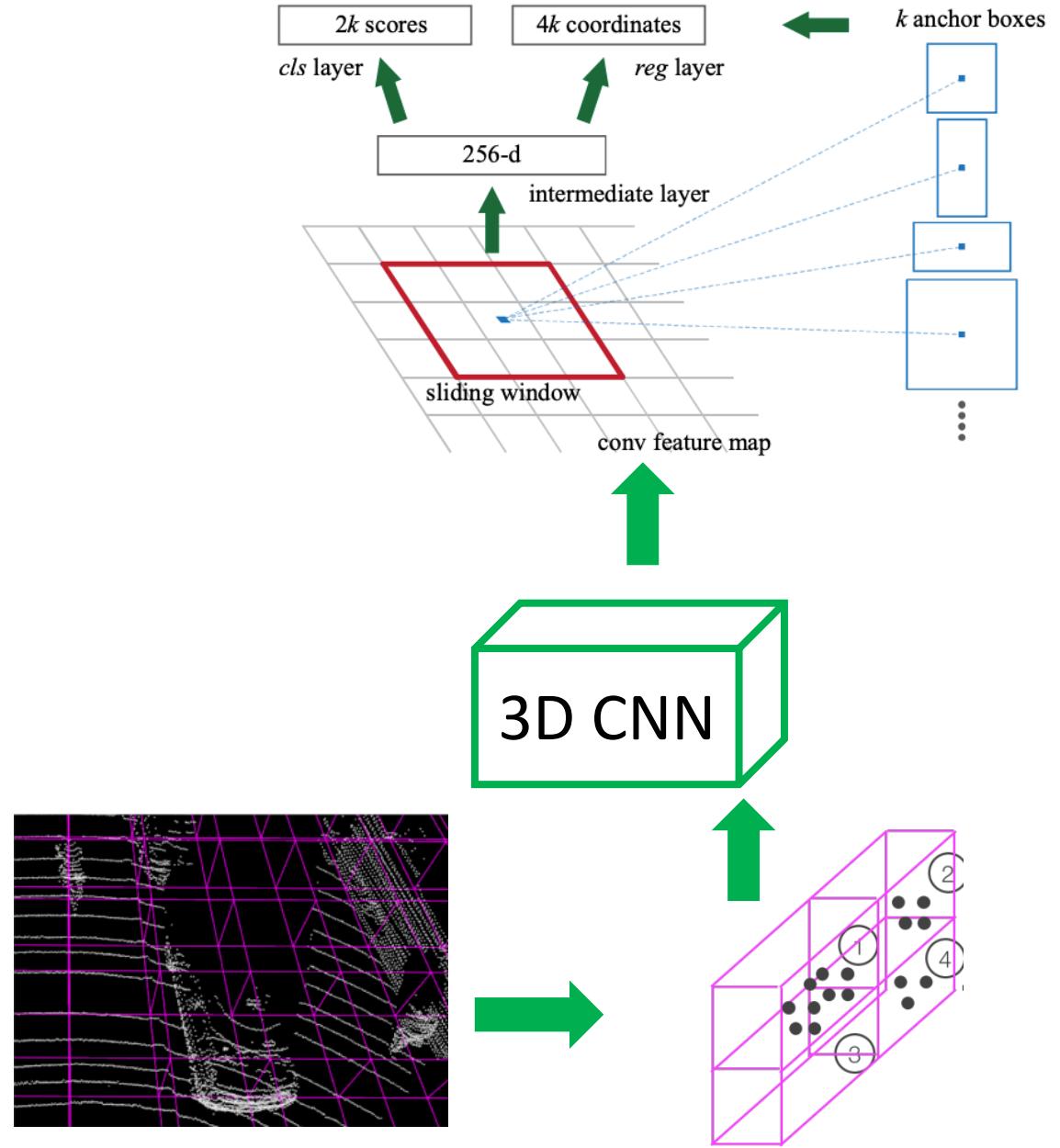
- ❖ Multi-view projection + Faster RCNN
 - 3D proposal from BV, project into other views
 - ROI pooling in each view, combined for Cls./Reg.
- ❖ Pioneer work but not frequently used nowadays

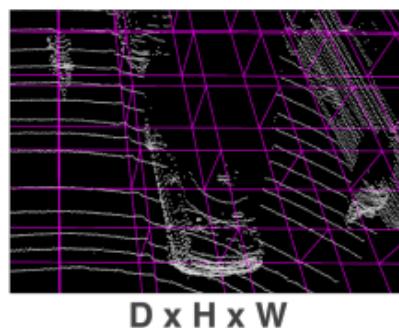




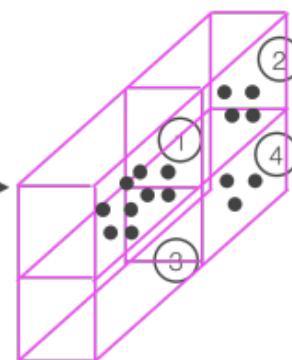
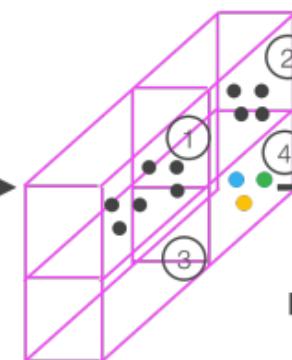
VoxelNet

1. Build voxel grid
2. Get feature per-cell
 - PointNet
3. 3D voxel \rightarrow 2D feature map
 - 3D convolution
4. Region Proposal Network (RPN)
 - One network for one category.
 - RPN is only predicting if the anchor box is an object

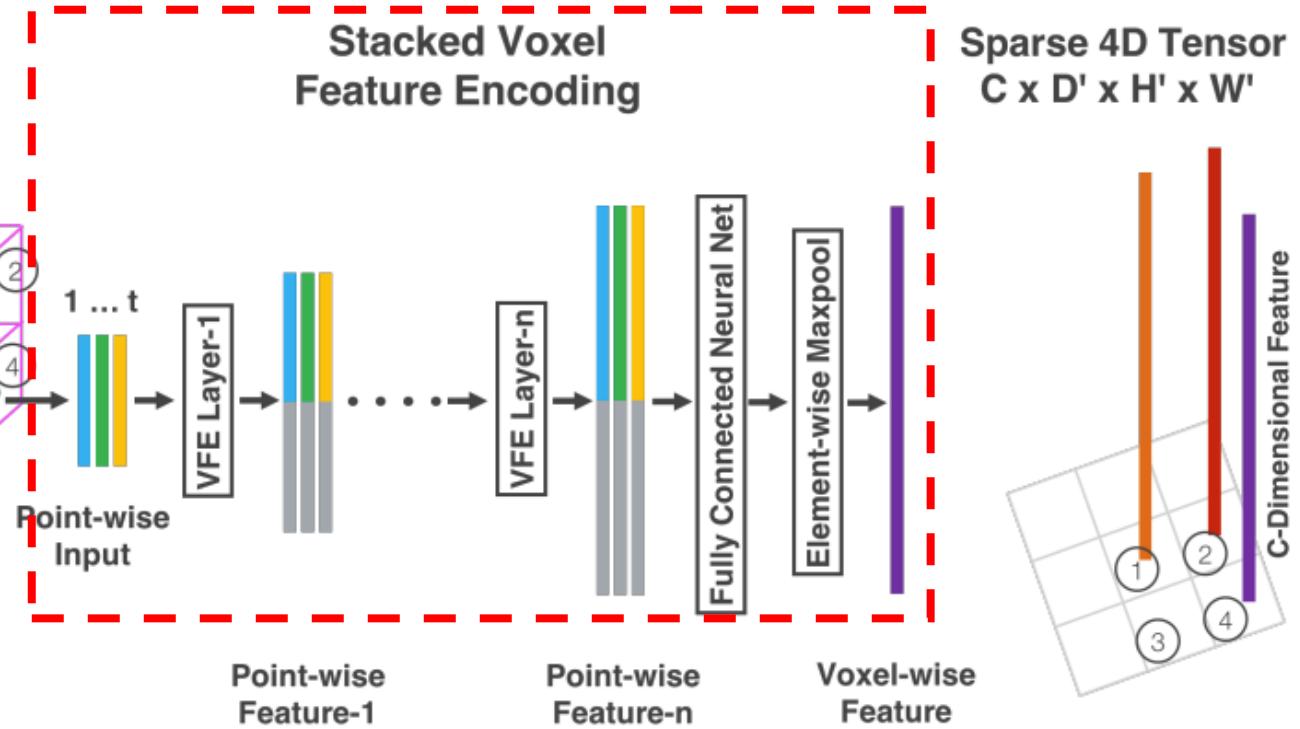
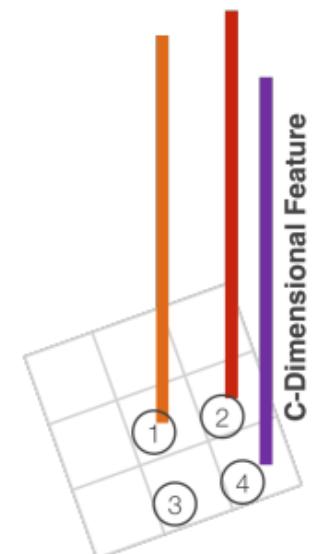


*Input:* $10 \times 400 \times 352$ voxelVoxel
PartitionMax T points per cell

Grouping

Random
Sampling

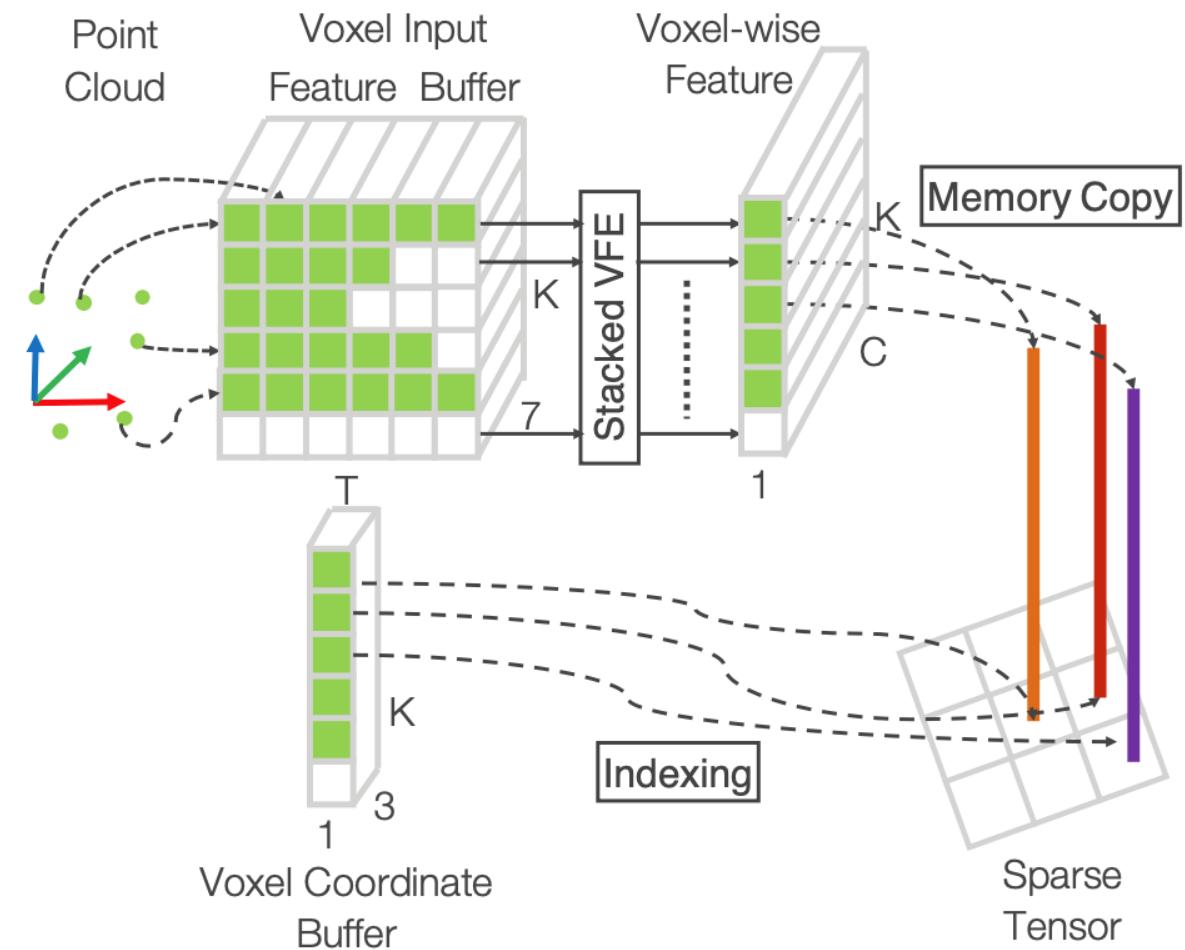
VFE Lecture 5

*Output:* $128 \times 10 \times 400 \times 352$ Sparse 4D Tensor
 $C \times D' \times H' \times W'$ 



VoxelNet – Voxel Feature Encoding

- 10 × 400 × 352 cells, Run VFE for all of them?
 - No, 90% cells are empty
- 1. Build two containers
 - a) K – maximum number of non-empty voxels
 - b) T – maximum number of points per voxel
 - c) $K \times T \times 7$ for points
 $(x_i, y_i, z_i, r_i, x_i - v_{ix}, y_i - v_{iy}, z - v_{iz})$
 - d) $K \times 3$ for voxel coordinates
- 2. Iterate each point
 - 1. Determine voxel coordinate (h_x, h_y, h_z)
 - Hash(h_x, h_y, h_z) → index k in container (c)(d)
 - 2. If k -th element exist in (c)/(d), append to (c)
 - 3. If not, create k -th element in (c) (d)
- 3. VFE / PointNet on (c) → $K \times 128$
- 4. Put (c) back to the voxel grid → $128 \times 10 \times 400 \times 352$





Input: $128 \times 10 \times 400 \times 352$

Conv3D

- Output channel # 64, kernel (3, 3, 3), stride (2, 1, 1), padding (1, 1, 1)
- Output channel # 64, kernel (3, 3, 3), stride (1, 1, 1), padding (0, 1, 1)
- Output channel # 64, kernel (3, 3, 3), stride (2, 1, 1), padding (1, 1, 1)

Output: $C' \times D' \times H' \times W'$ – Homework

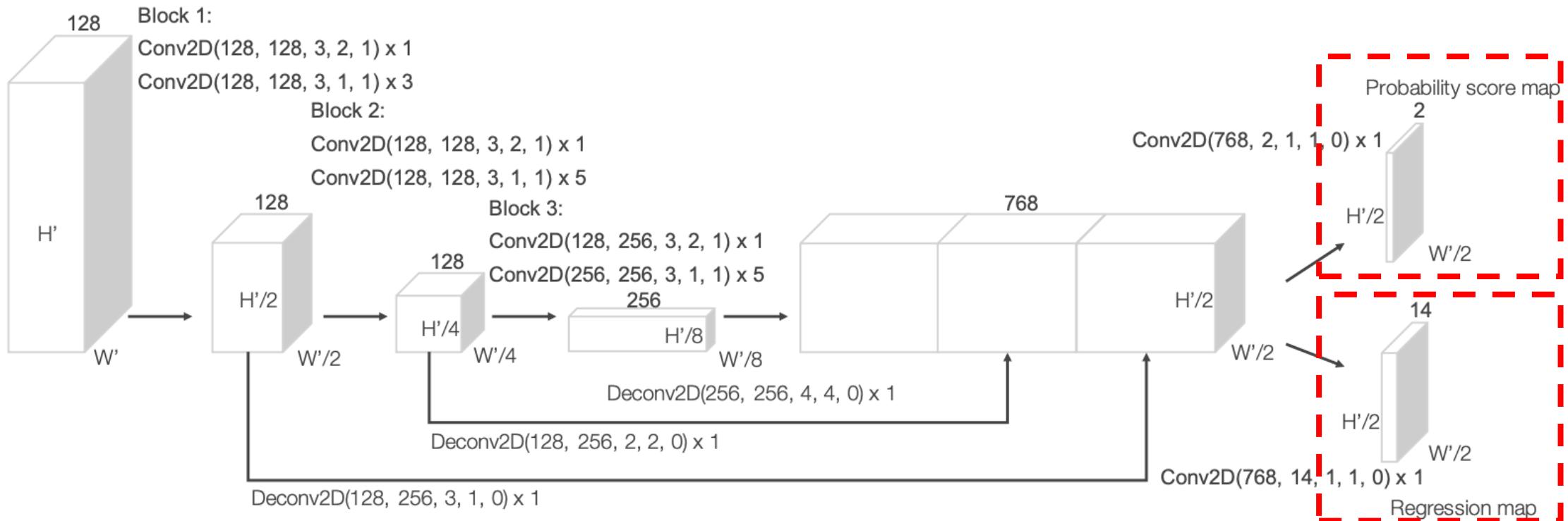
- Answer is $64 \times 2 \times 400 \times 352$

Reshape into 2D feature map $2C' \times H' \times W'$

- This is image-like feature map!



VoxelNet – Region Proposal Network (RPN)



Two anchor boxes

- $x, y, z, l, w, h, \theta = 0^\circ$
- $x, y, z, l, w, h, \theta = 90^\circ$



• Loss function for VoxelNet

$$L = \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) + \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*)$$

• L_{cls} is “Binary Cross Entropy Loss” – in fact not a standard one

- VoxelNet represents the “objectness confidence” with only 1 number, instead of 2.
- $L_{\text{cls}}(p_i^{\text{pos}}) = \log\left(\sigma\left(e^{p_i^{\text{pos}}}\right)\right)$
- $L_{\text{cls}}(p_i^{\text{neg}}) = \log\left(1 - \sigma\left(e^{p_i^{\text{neg}}}\right)\right)$
- Sigmoid $\sigma(x) = \frac{e^x}{e^x + 1} \in (0, 1)$



- ❖ What is positive / negative anchor box?
- ❖ Thousand readers have one thousand Hamlet.
- ❖ E.g.
 - Positive anchor box
 - $IoU > 0.5$ with **any** ground truth box
 - Has highest IoU with a ground truth box (each ground truth box will be associated with at least one anchor box).
 - Negative anchor box
 - $IoU < 0.35$ with **every** ground truth box
 - Don't care
 - $0.35 \leq IoU \leq 0.5$ with **every** ground truth box



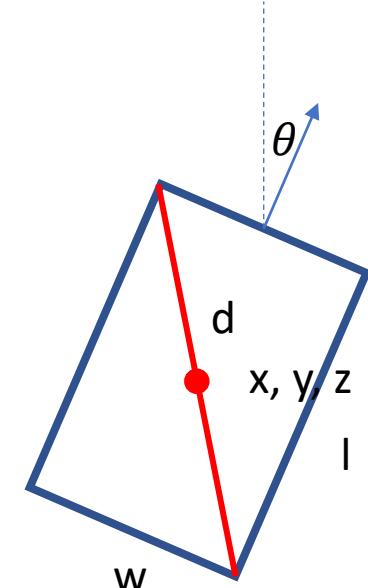
- Ground truth box: $\mathbf{u}^* = (x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g)$
- A matching anchor box (positive box): $\mathbf{u} = (x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$
- Loss between Ground truth and positive anchor box:

$$L_{reg}(\mathbf{u}, \mathbf{u}^*) = \rho(\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta \theta)$$

$$\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$$

$$\Delta l = \log\left(\frac{l^g}{l^a}\right), \Delta w = \log\left(\frac{w^g}{w^a}\right), \Delta h = \log\left(\frac{h^g}{h^a}\right),$$

$$\Delta \theta = \theta^g - \theta^a \quad d^a = \sqrt{(l^a)^2 + (w^a)^2}$$

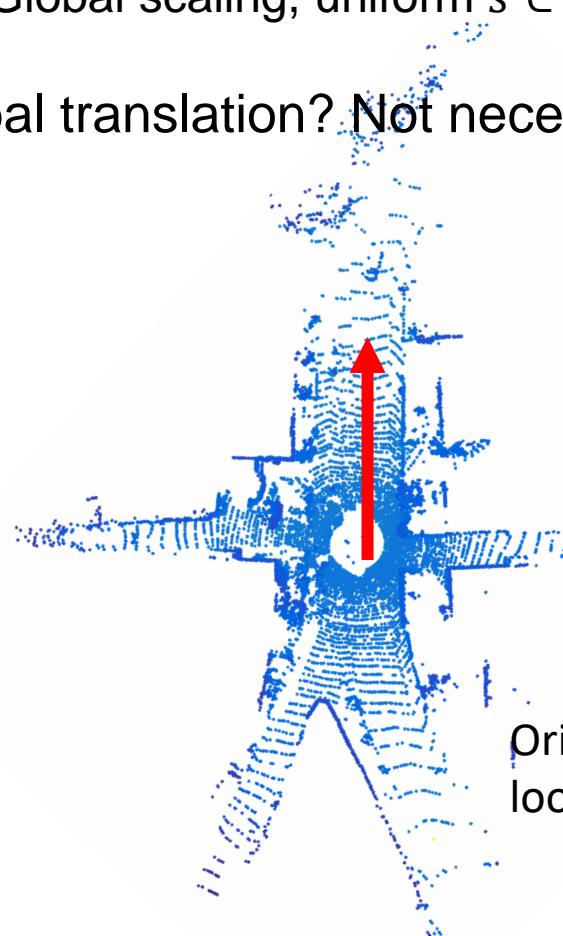




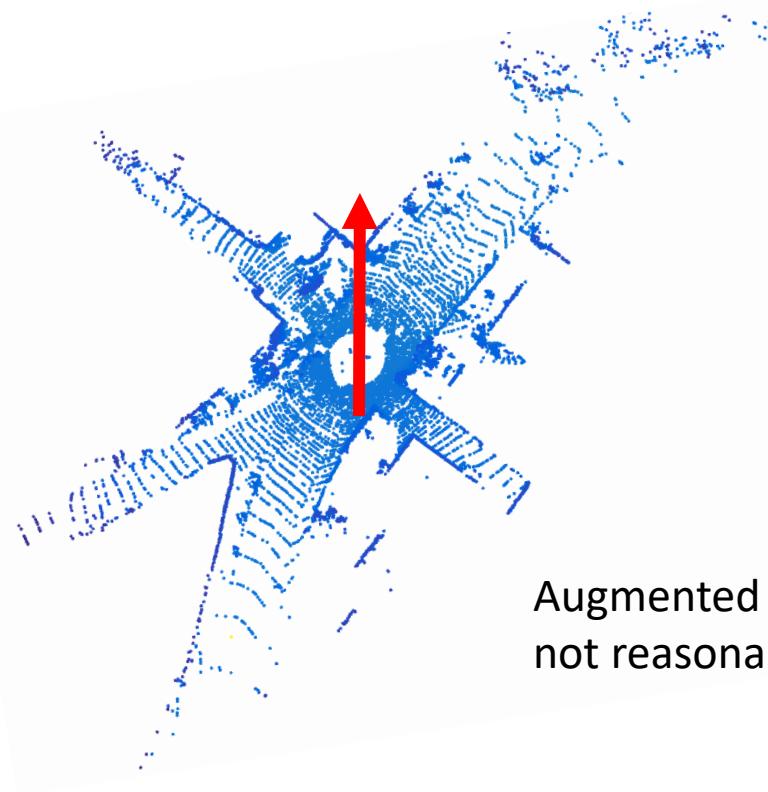
Global rotation around the up-axis

- Uniform $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, why not $[-\pi, \pi]$?
- Global scaling, uniform $s \in [0.95, 1.05]$

Global translation? Not necessary.



Original direction,
looks reasonable

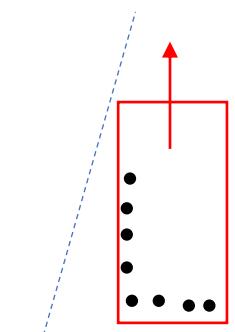


Augmented direction,
not reasonable

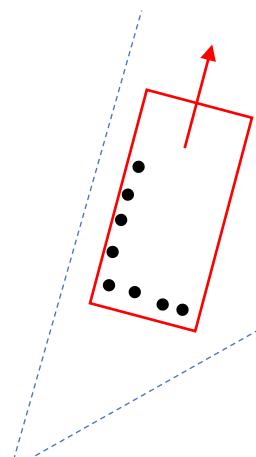


For each ground truth box:

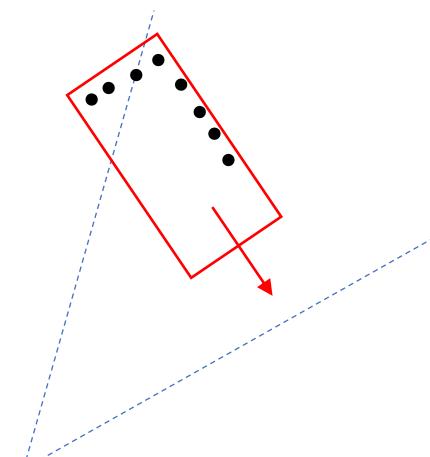
- Random rotate it and points inside it by uniform $\theta \in \left[-\frac{\pi}{10}, \frac{\pi}{10}\right]$
- Random translate $[\Delta x, \Delta y, \Delta z], \Delta * \sim \mathcal{N}(0, 1)$
- If there collision for two boxes after augmentation, give up augmentation.



Original data



Proper augmentation

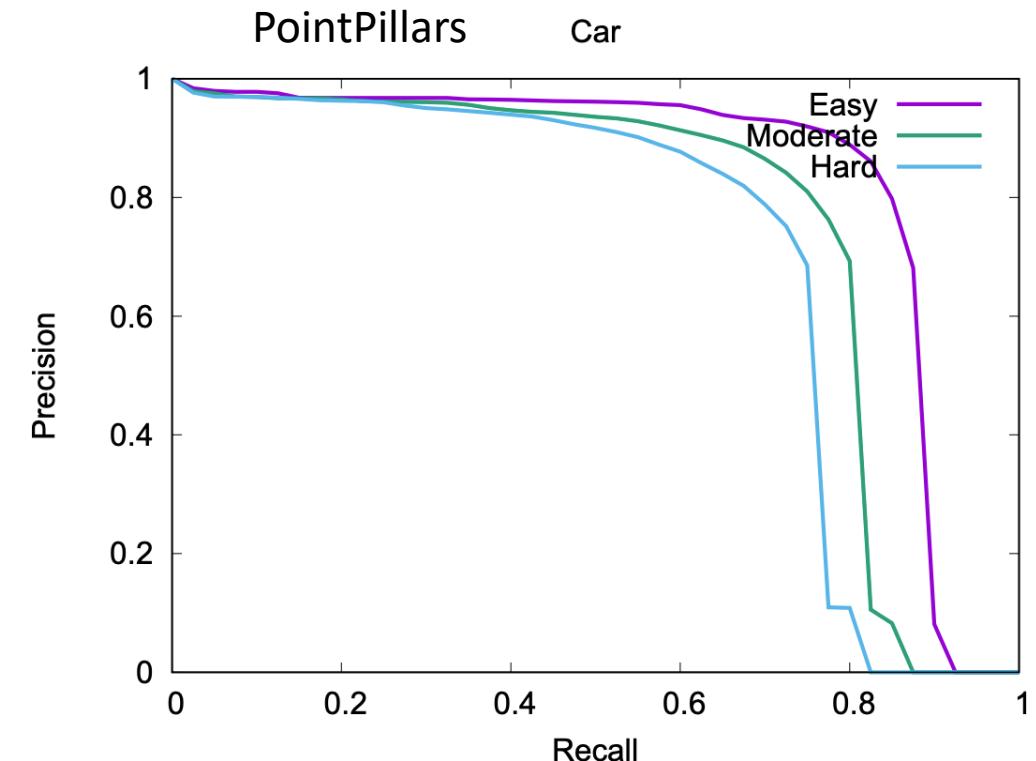
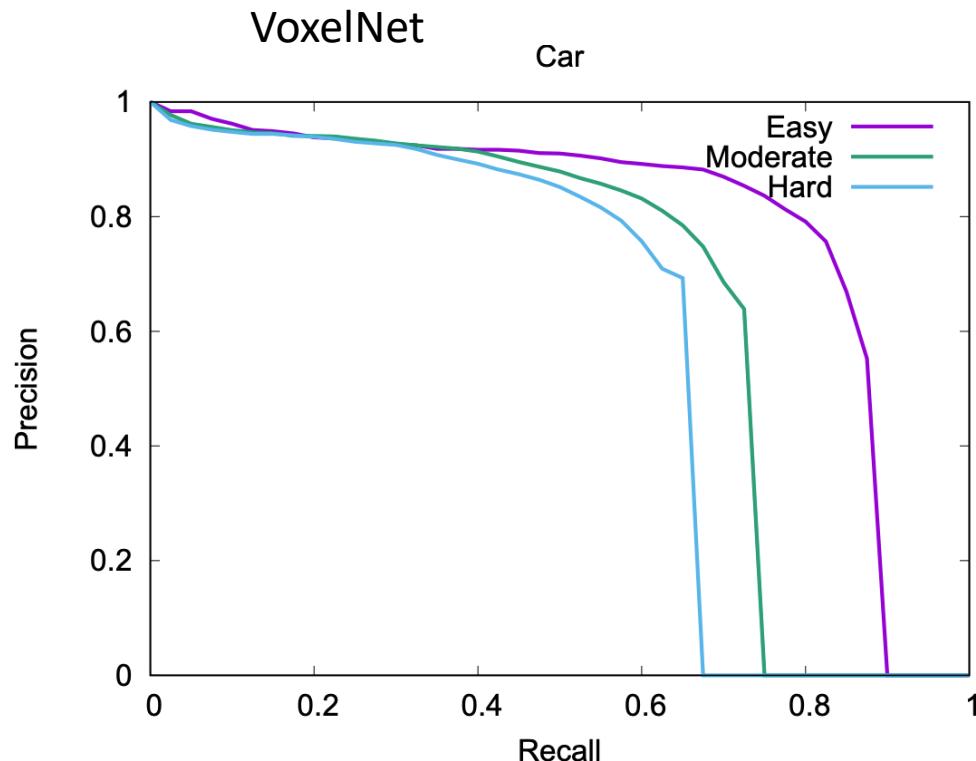


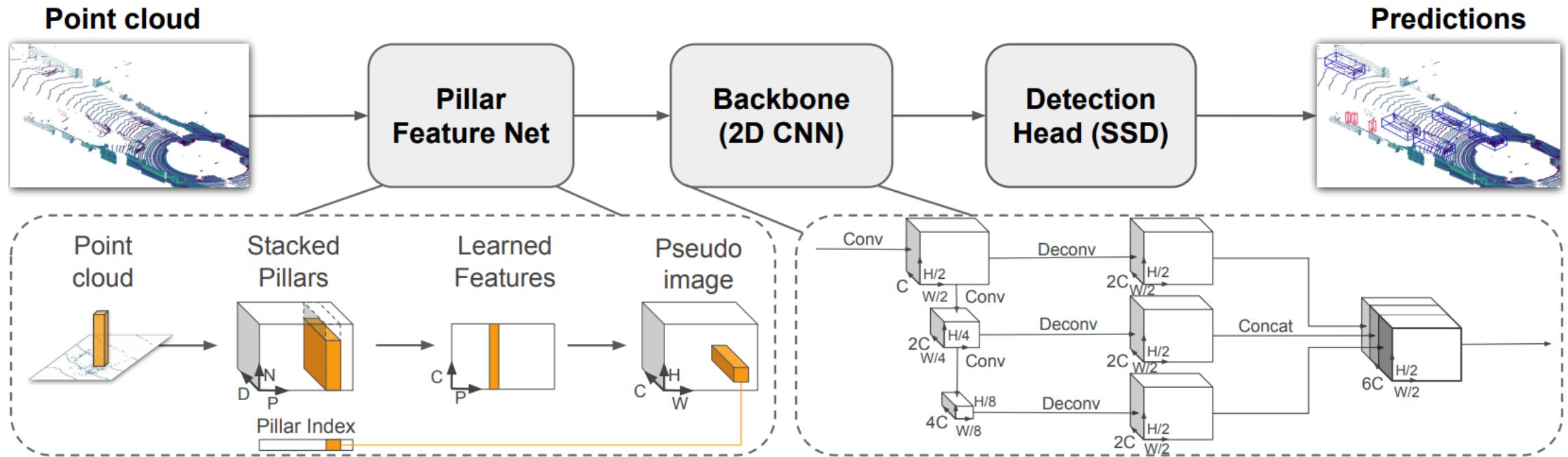
Too much augmentation



VoxelNet – Performance on KITTI

	Method	Setting	Code	Moderate	Easy	Hard	Runtime
125	<u>VoxelNet(Unofficial)</u>			64.17 %	77.82 %	57.51 %	0.5 s
77	<u>PointPillars</u>	++	code	74.31 %	82.58 %	68.99 %	16 ms



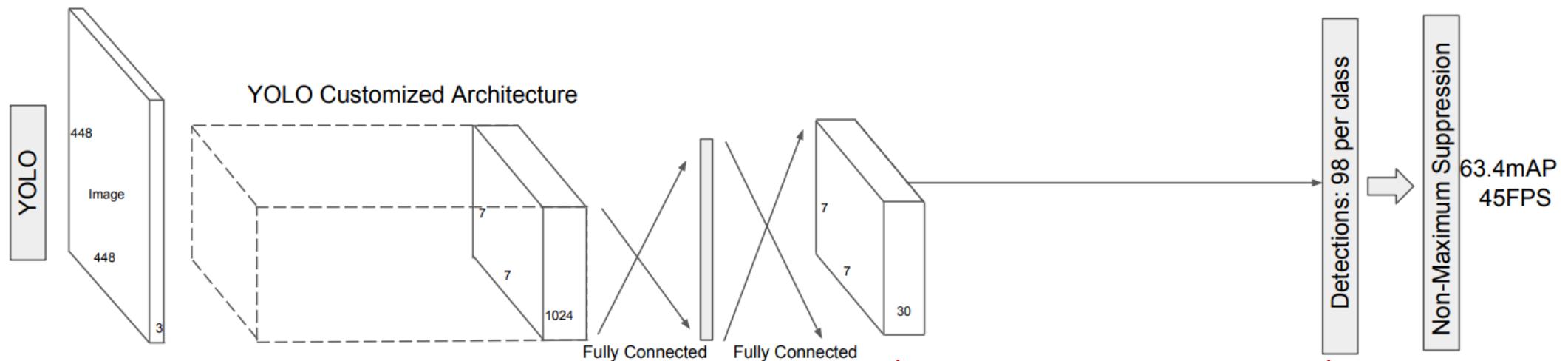
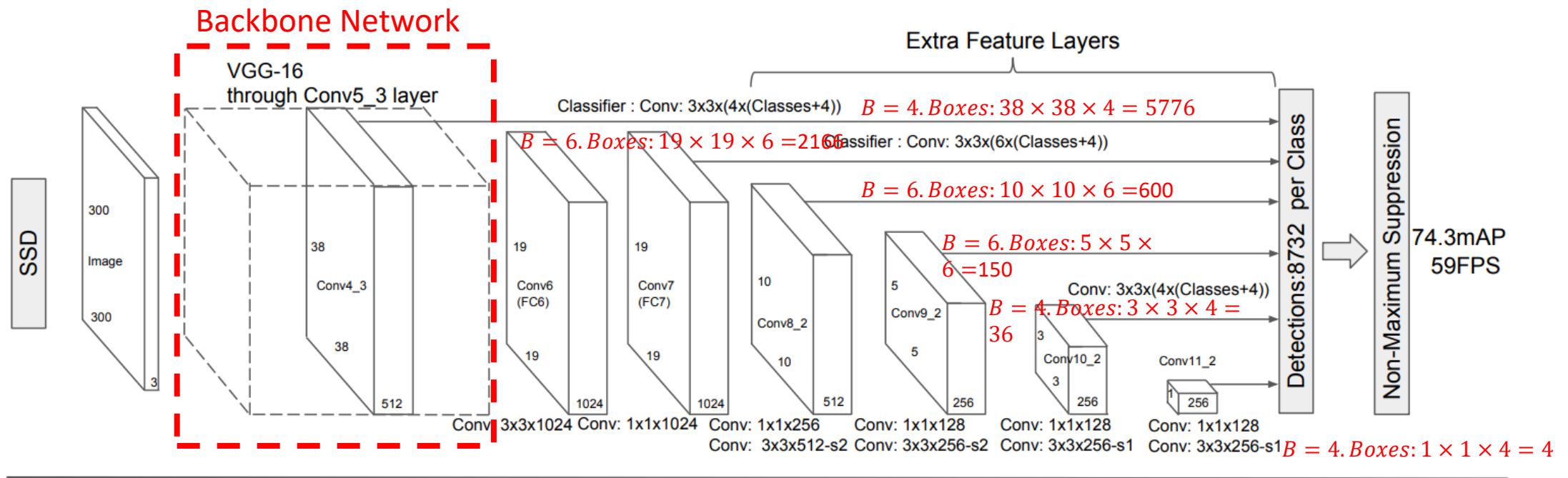


❖ Why 3D voxel grid?

❖ Directly compress point cloud into $C \times H \times W$ feature map!



SSD



This is YOLO v1. YOLO v3 is similar to SSD



Overall loss function

$$\mathcal{L} = \frac{1}{N_{pos}} (\beta_{loc} \mathcal{L}_{loc} + \beta_{cls} \mathcal{L}_{cls} + \beta_{dir} \mathcal{L}_{dir})$$

L_{loc} : regression loss, almost same as VoxelNet, except

$$\Delta\theta = \sin(\theta^g - \theta^a)$$

L_{dir} : the sin in L_{loc} can't distinguish 180° flip

- Add a softmax classification loss on the discretized directions
- E.g., divide the rotation into 4 bins, [0, 90], [90, 180], [180, 270], [270, 360]

L_{cls} : similar to VoxelNet, but use **FocalLoss** instead of CrossEntropyLoss



Consider an example,

- 1 ground truth “human” box
- Network predicts 100 boxes
 - $p_{background} = 0.6$ for the first 99 boxes
 - $p_{human} = 0.2$ for the 100th box

Cross Entropy Loss

$$L_{CE} = \sum_t -\log p_t = -99 \log 0.6 - \log 0.2 = 50.57 + 1.61$$

The loss caused by wrong box classification (the 100th box) is 1.61

The loss caused by correct box classification (the 99 boxes) is 50.57

Easy, network just predict everything as background!



Focal Loss

• Cross Entropy Loss $L_{CE}(p_t) = -\log p_t$

• Focal Loss $L_{FL} = -(1 - p_t)^\gamma \log p_t$

• Come back to our example, with $\gamma = 2$.

$$L_{FL} = -99(1 - 0.6)^2 \log 0.6 - (1 - 0.2)^2 \log 0.2 = 8.09 + 1.03$$

• The loss caused by wrong classification is taking a larger percentage now.

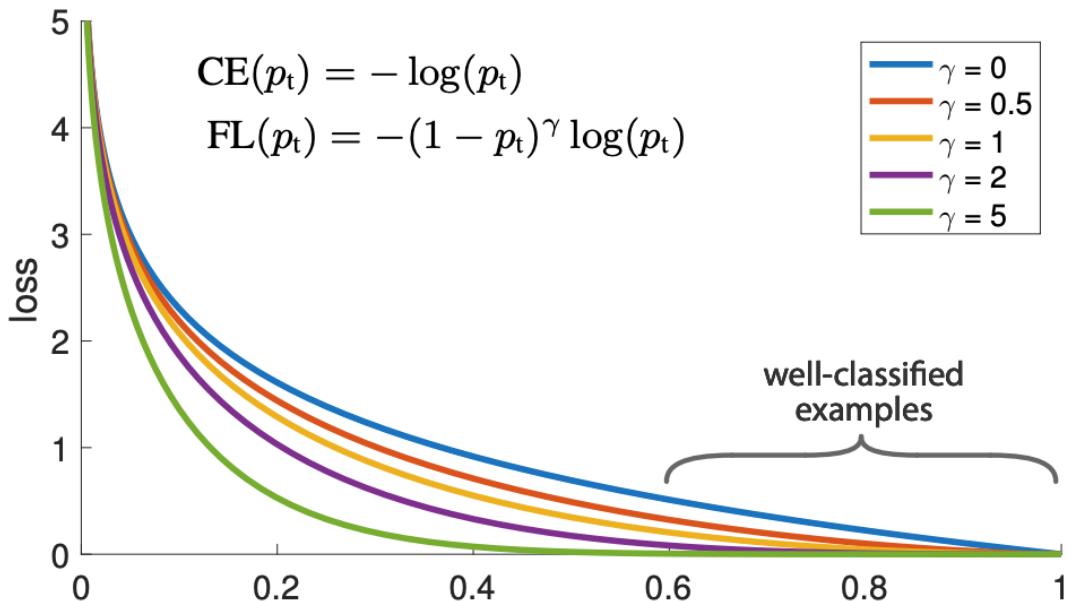


Figure: Focal Loss with various γ

Original Paper: Focal Loss for Dense Object Detection

- Focal Loss is designed for unbalanced classification
- Reduce the relative loss for well-classified examples ($p_t > 0.5$)
- Focus on hard, mis-classified examples ($p_t \leq 0.5$)



Multi-view Projection

Build image-like feature map with PointNet

- PointNet to convert point cloud detection into image detection + normal 2D CNN
- Two-stage / One-stage

Point-wise operation

- Replace 2D CNN with point-wise operations like PointNet
- Two-stage / One-stage

Point cloud & image fusion

- Core idea only, not in detail.



❖ VoxelNet / PointPillar

- Convert point cloud into image-like feature map
- Single stage detector – RPN / SSD, no ROI pooling

❖ How about native method on point cloud?

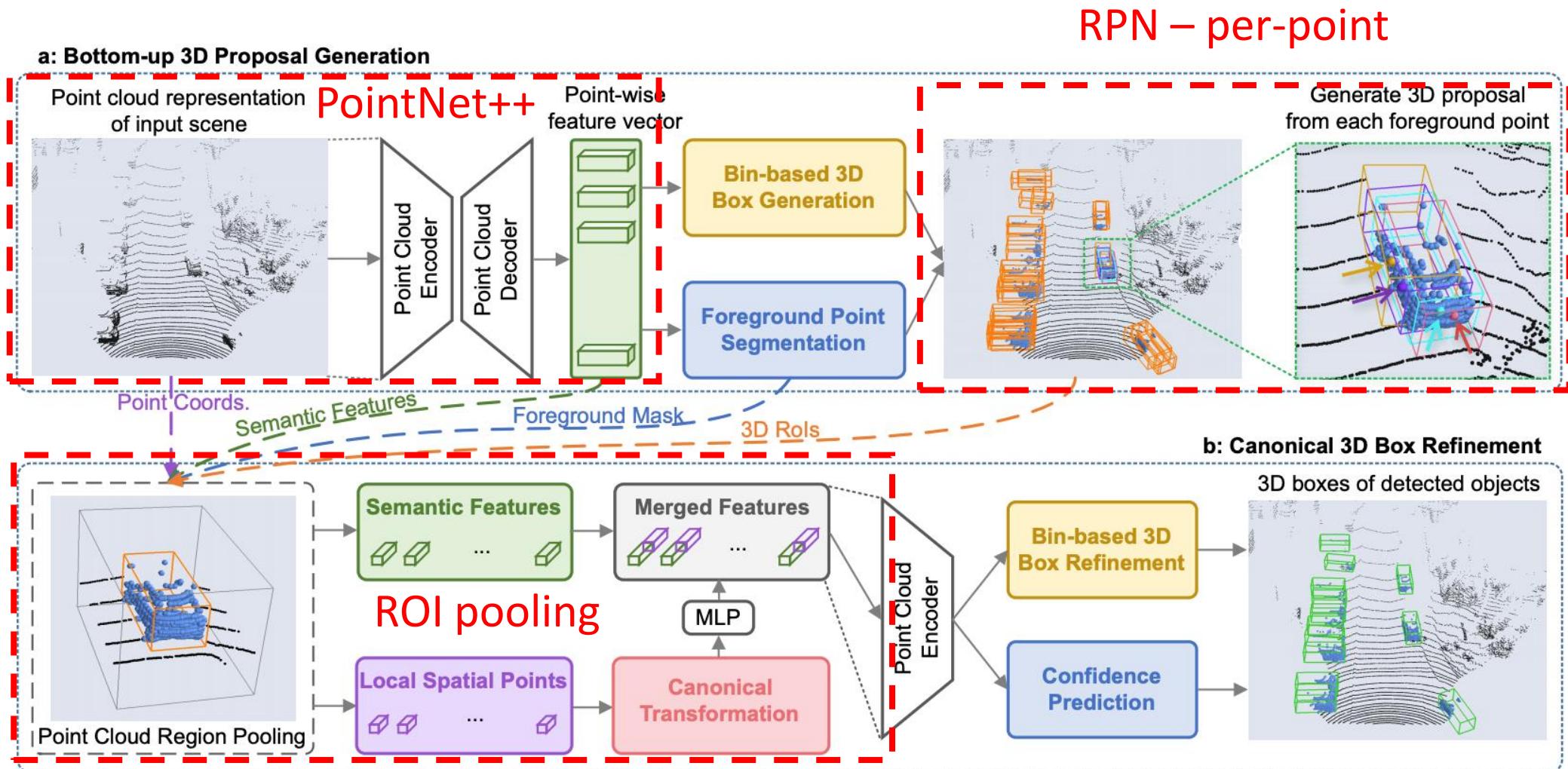
❖ How about two stage method?

❖ PointRCNN

	Method	Setting	Code	Moderate	Easy	Hard	Runtime
125	VoxelNet(Unofficial)			64.17 %	77.82 %	57.51 %	0.5 s
77	PointPillars		code	74.31 %	82.58 %	68.99 %	16 ms
61	MMLab-PointRCNN		code	75.64 %	86.96 %	70.70 %	0.1 s



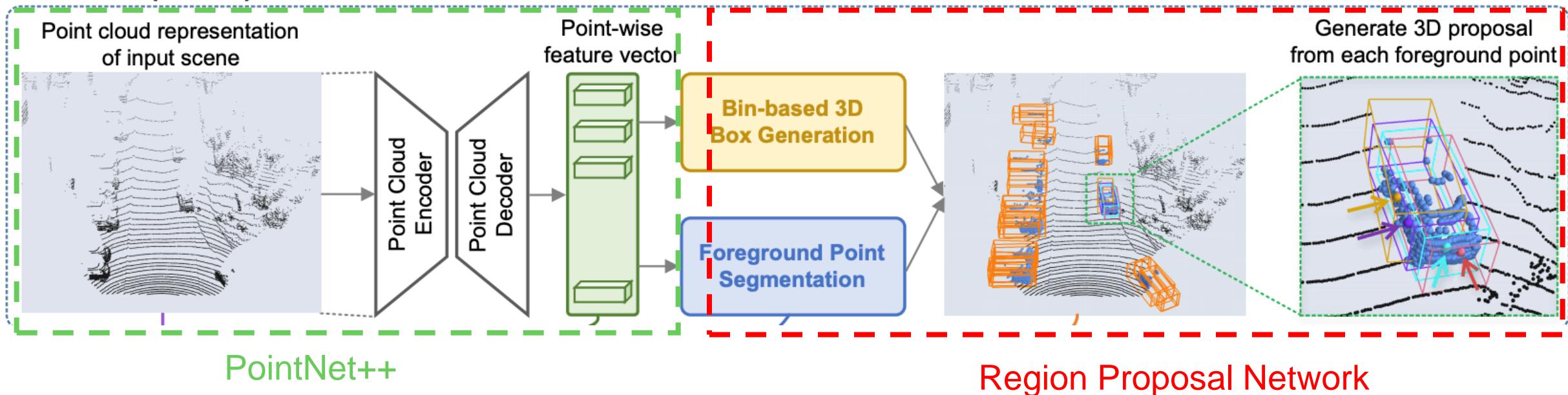
PointRCNN - Architecture





PointRCNN – Stage 1

a: Bottom-up 3D Proposal Generation



Backbone – PointNet++

- Input: point cloud $4 \times N$ (x, y, z , intensity)
- Output: per-point feature $C \times N$

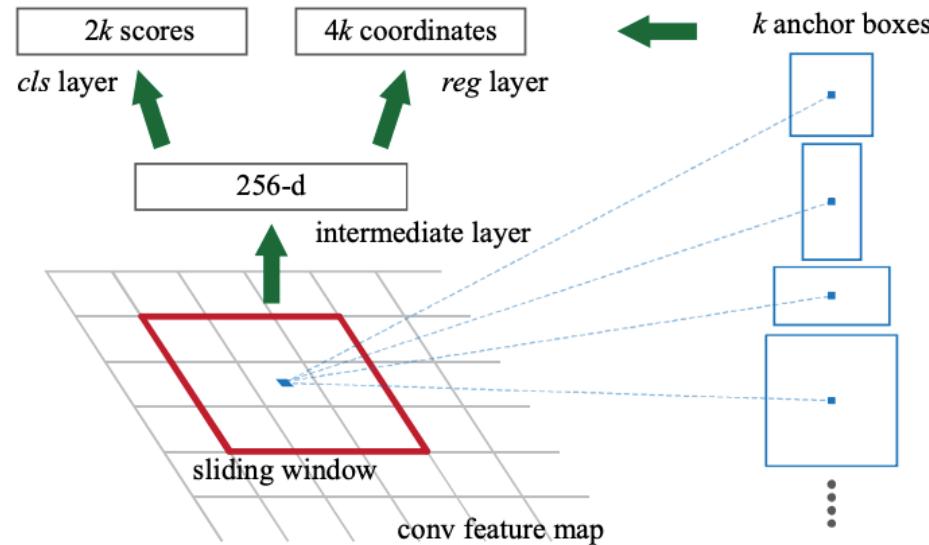
RPN

- Each foreground point generate a proposal



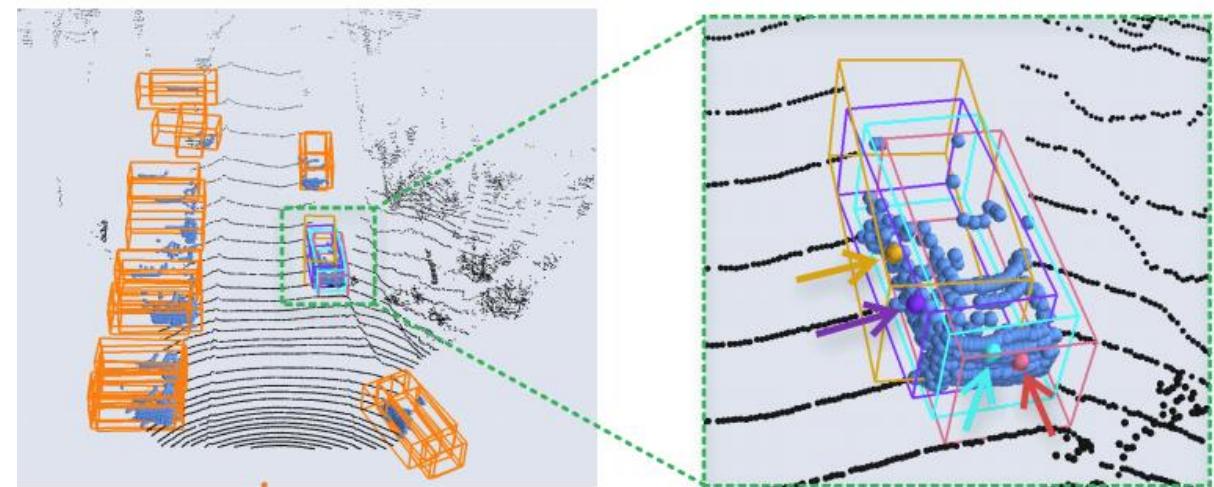
What is image based RPN?

- Box classification (is it an object or not)
- Box regression



What is PointRCNN Stage 1?

- Foreground / Background classification
- Bin-based 3D box generation





- Foreground – points inside ground truth boxes
- Background – other points
- Loss function: Focal Loss

$$\mathcal{L}_{\text{focal}}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t),$$

where $p_t = \begin{cases} p & \text{for foreground point} \\ 1 - p & \text{otherwise} \end{cases}$

- E.g., network predicts the background/foreground score $[c_0, c_1]$ for each point

- $p = p_{\text{foreground}} = \frac{e^{c_1}}{e^{c_0} + e^{c_1}}, \quad p_{\text{background}} = 1 - p = \frac{e^{c_0}}{e^{c_0} + e^{c_1}}$



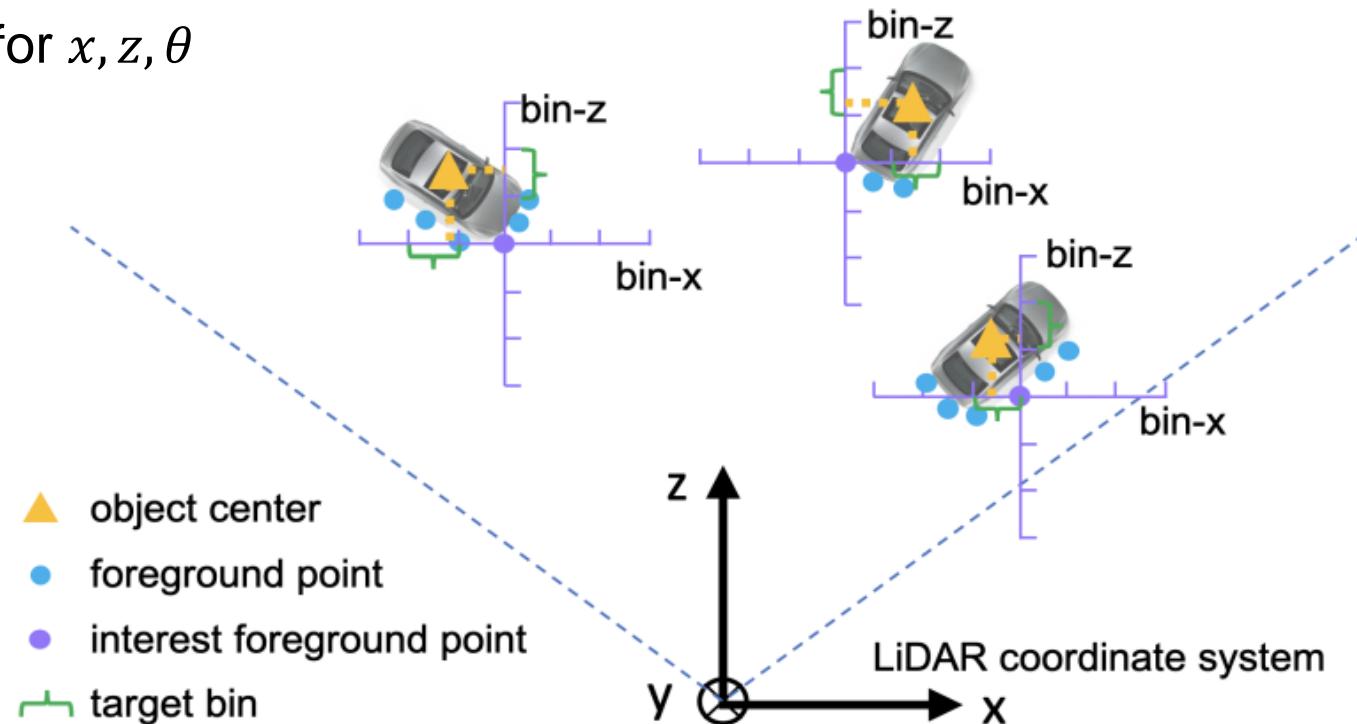
❖ Objective: generate one box / foreground point

❖ Regression for x, y, z, l, w, h, θ ?

- Classification + Finetuning residual for x, z, θ
- Direct regression for y, l, w, h

❖ For a foreground point,
how to determine box center x ?

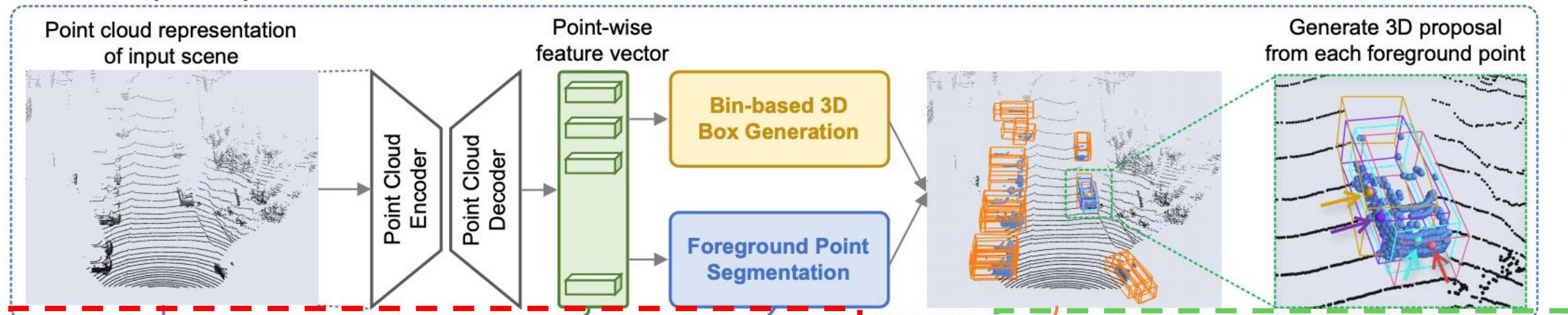
1. Classify which bin
2. Predict the residual to bin center
3. $x = bin_{center} + res_x$



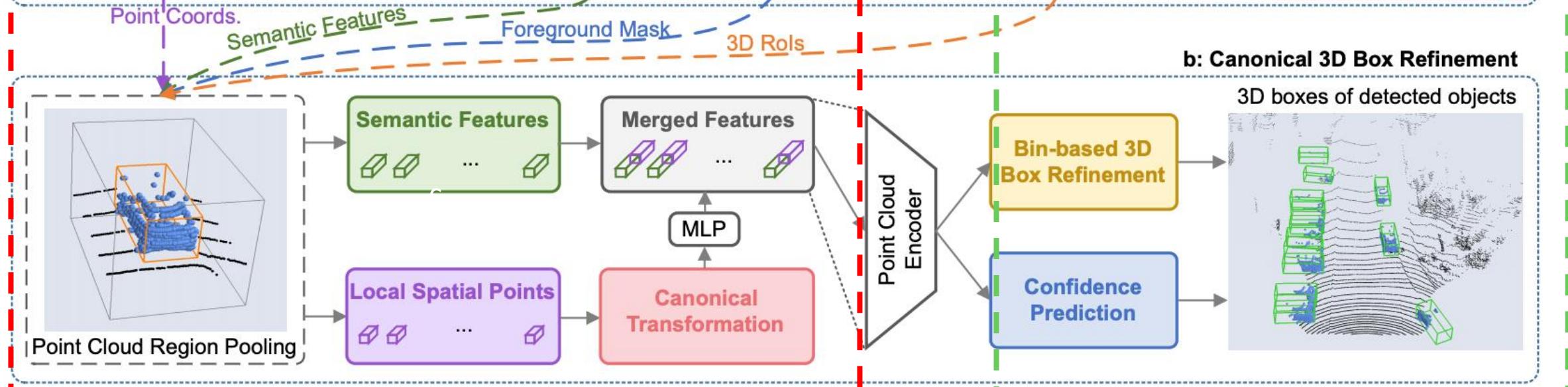


PointRCNN – Stage 2

a: Bottom-up 3D Proposal Generation



b: Canonical 3D Box Refinement

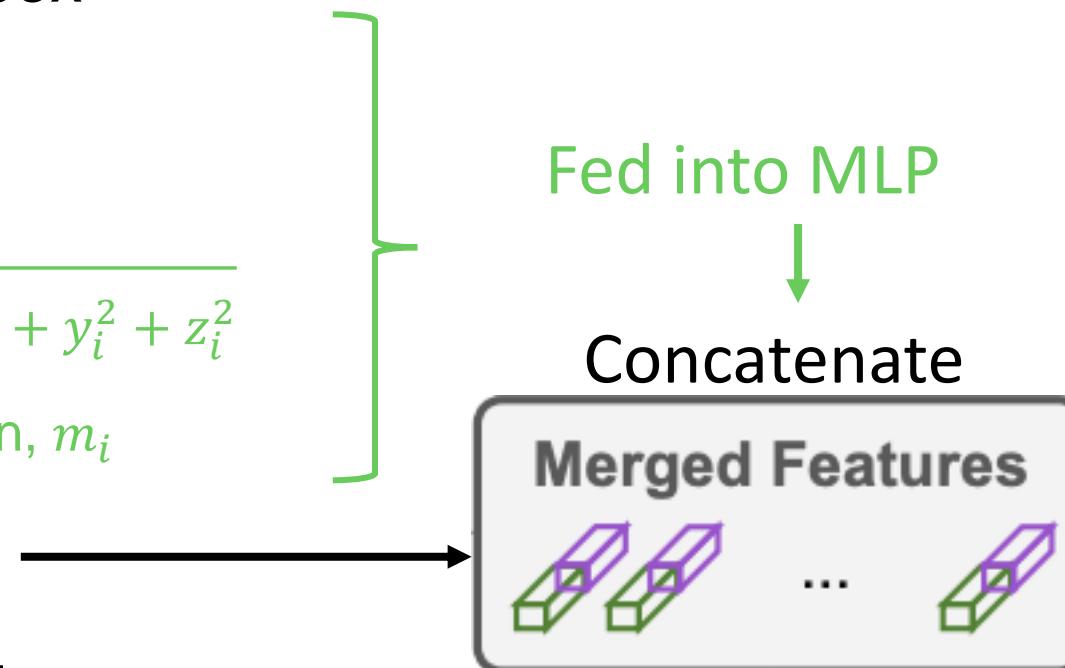


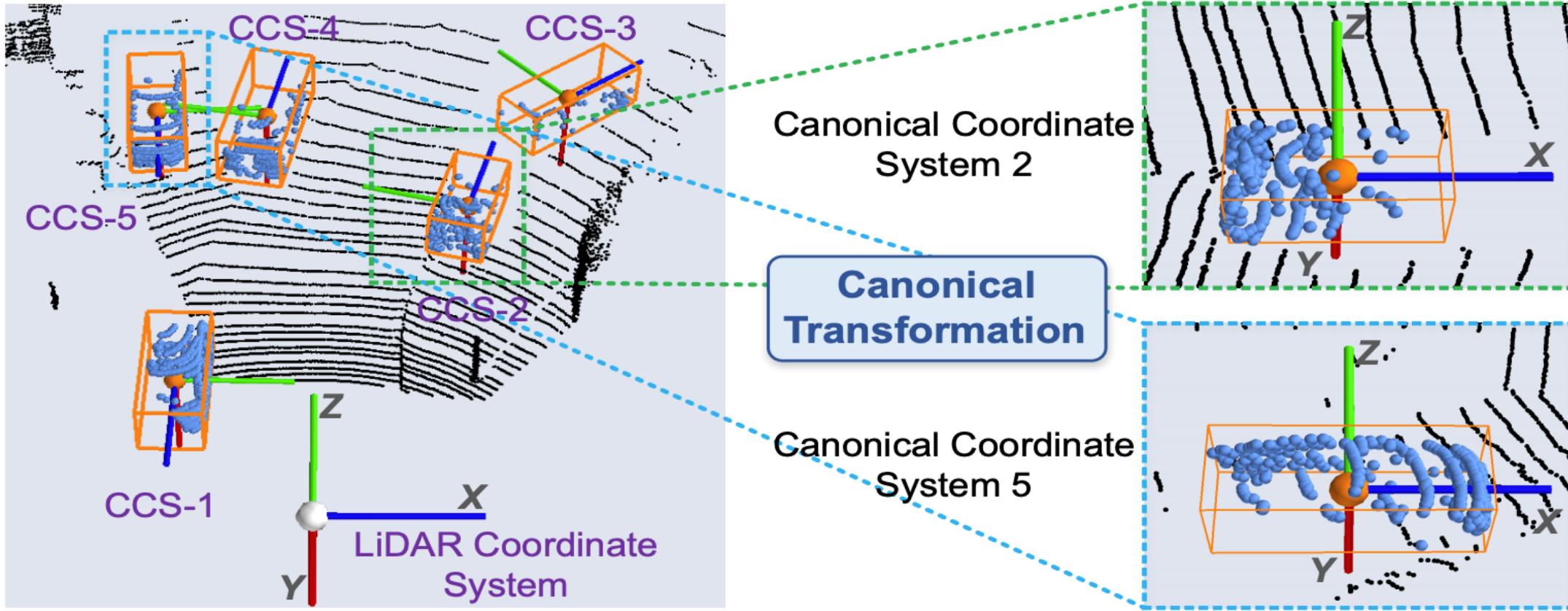
ROI pooling

Box refinement



- ❖ NMS on the Stage 1 proposals. (300 for training, 100 for inference)
- ❖ Enlarge the Stage 1 proposals
- ❖ Get information in the enlarged box
 - Point coordinate, x_i, y_i, z_i
 - Canonical Transformation
 - Point intensity, r_i
 - Point's distance to origin, $d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$
 - Foreground / background prediction, m_i
 - Per-point features from PointNet++





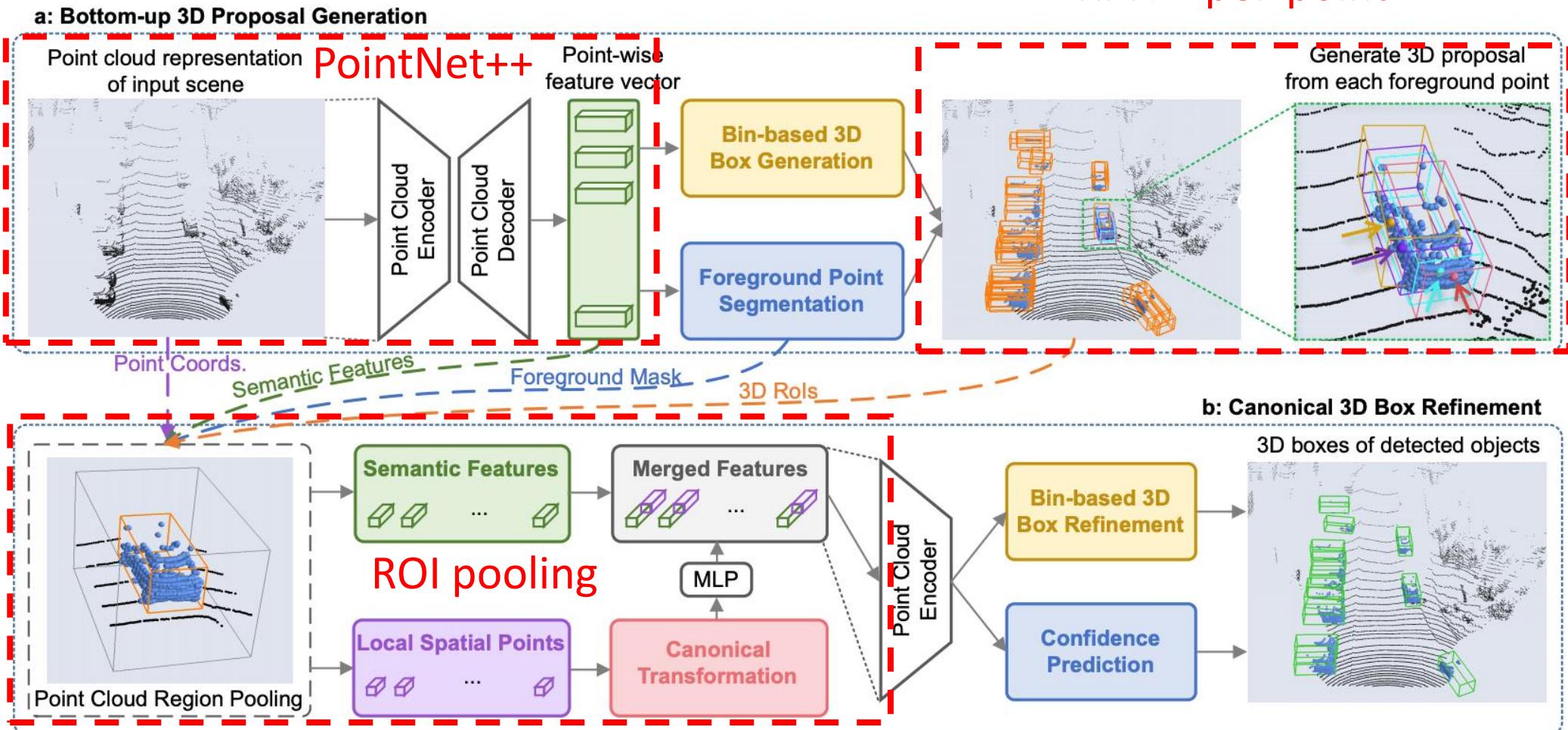
canonical transformation

- Normalized points so that their center is box proposal center
- Rotate the points box proposal heading is x-axis



PointRCNN - Summary

RPN – per-point





Multi-view Projection

Build image-like feature map with PointNet

- PointNet to convert point cloud detection into image detection + normal 2D CNN
- Two-stage / One-stage

Point-wise operation

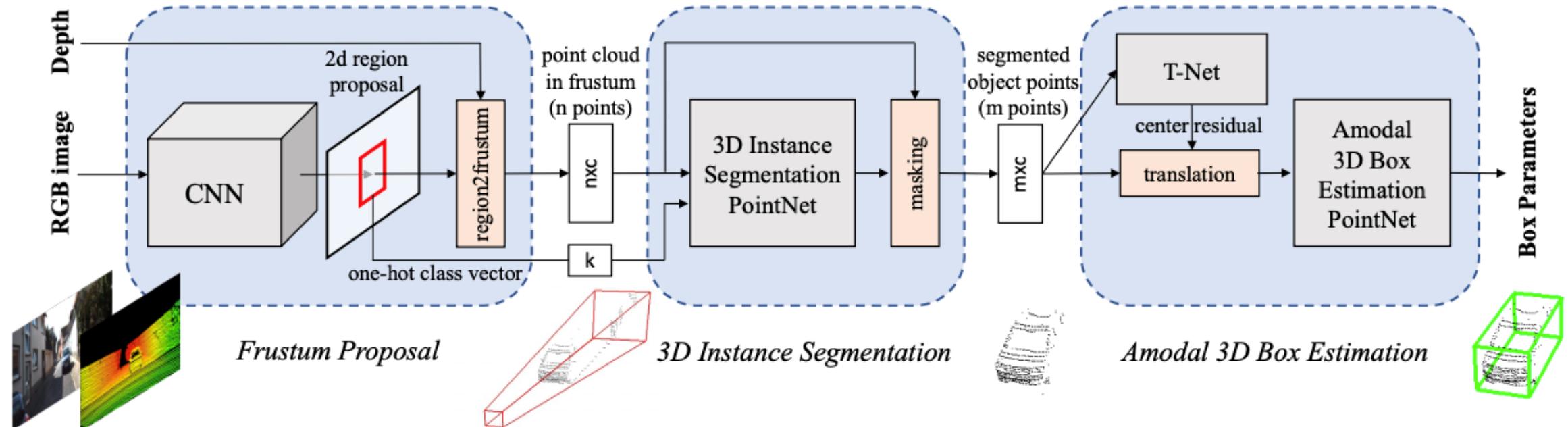
- Replace 2D CNN with point-wise operations like PointNet
- Two-stage / One-stage

Point cloud & image fusion

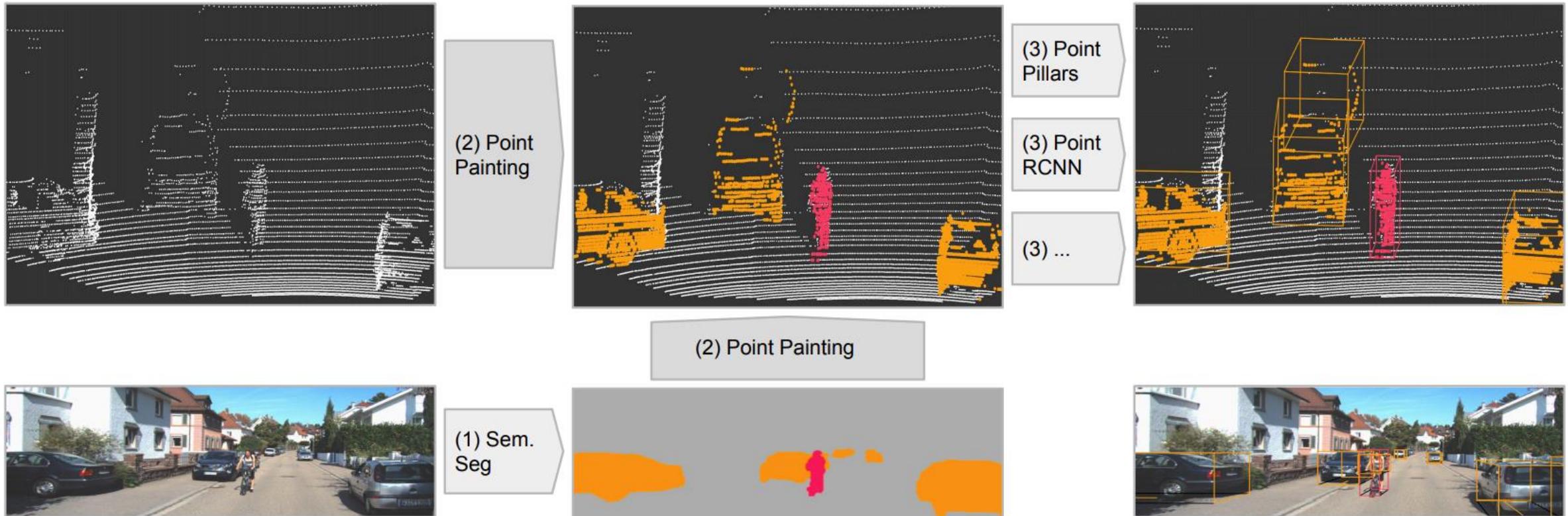
- Core idea only, not in detail.



Frustum PointNet



- Object proposal from 2D image based detector.
- ROI pooling is actually the frustum extraction.



- ➄ Augment each point with the semantic segmentation label from an image based semantic segmentation network.



Multi-view Projection

Build image-like feature map with PointNet

- PointNet to convert point cloud detection into image detection + normal 2D CNN
- Two-stage / One-stage

Point-wise operation

- Replace 2D CNN with point-wise operations like PointNet
- Two-stage / One-stage

Point cloud & image fusion

- Core idea only, not in detail.



Homework

- ① 1. Setup the KITTI object detection evaluation environment
 1. git clone https://github.com/prclibo/kitti_eval.git
 2. g++ -O3 -DNDEBUG -o evaluate_object_3d_offline evaluate_object_3d_offline.cpp
 3. sudo apt-get install gnuplot
 4. sudo apt-get install texlive-extra-utils
- ② 2. Download and read the KITTI Object Detection dataset "devkit" readme.
- ③ 3. Divide the KITTI Object Detection into training set and validation set.
 - [KITTI train/val split used in 3DOP/Mono3D/MV3D](#)
 - “train.txt” for training, “val.txt” for testing, ignore the “test.txt/trainval.txt”
- ④ 4. Generate object detection results on KITTI validation set
 1. Option 1: find any open-source 3d object detector, run it.
 2. Option 2: copy the ground truth as the result, but you need to process it into the correct format.
- ⑤ 5. Write a report.



What does KITTI Label / Result look like?

#Values	Name	Description
1	type	Describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'
1	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries
1	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded 2 = largely occluded, 3 = unknown
1	alpha	Observation angle of object, ranging [-pi..pi]
4	bbox	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates
3	dimensions	3D object dimensions: height, width, length (in meters)
3	location	3D object location x,y,z in camera coordinates (in meters)
1	rotation_y	Rotation ry around Y-axis in camera coordinates [-pi..pi]
1	score	Only for results: Float, indicating confidence in detection, needed for p/r curves, higher is better.



What does KITTI Label / Result look like?

Label example:

- 000000.txt
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01

Result example:

- data/000000.txt
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 **10.0**
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 **8.0**
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 **6.0**



Homework

- 5. ./evaluate_object_3d_offline gt_dir result_dir
- 6. Submit your result folder as a zip file.
- 7. Build your own object classification dataset from KITTI
 - 1. Extract points in car/pedestrian/cyclist as point cloud
 - 2. The category label, of course, is the label of the box
- 8. Design and train your own classification network on this dataset

This is part of the final project.



❖ Object detection pipeline for lidar

- Use KITTI 3D object detection dataset
- Step 1. Remove the ground from the lidar points
 - Any method you want – LSQ, Hough, RANSAC
- Step 2. Clustering over the remaining points
 - Any method you want
- Step 3. Classification over the clusters
- Step 4. Report the detection precision-recall for three categories: vehicle, pedestrian, cyclist



◆ Step 3. Classification over the clusters

- vehicle / pedestrian / cyclist / other
 - Step 3.1. Build dataset from KITTI 3D object detection dataset
 - Extract objects in the box for vehicle / pedestrian / cyclist
 - Other objects? Clustering!
 - Step 3.2. Build deep network to classify them.
 - Step 3.3. Report classification accuracy

◆ Step 4. Report the detection Precision-Recall for three categories: vehicle, pedestrian, cyclist

- Fit a cuboid over object detected as vehicle / pedestrian / cyclist
- Generate object detection results for KITTI
 - IoU threshold set to 0.2 (This is very low because fitting cuboid is not easy)
 - Modify evaluate_object_3d_offline.cpp, line 55 to everything 0.2. Then re-compile.