



清华大学
Tsinghua University

Advanced Computer Vision
THU×SENSETIME – 80231202



Chapter 2 - Section 7

Object Detection

Dr.Li Hongyang

Tuesday, April 2, 2021

Acknowledge : Wang Cheng



Outline

Part 1 **Introduction to Object Detection**

Part 2 **Anchor-based Solutions**

Part 3 **Anchor-free Solutions**

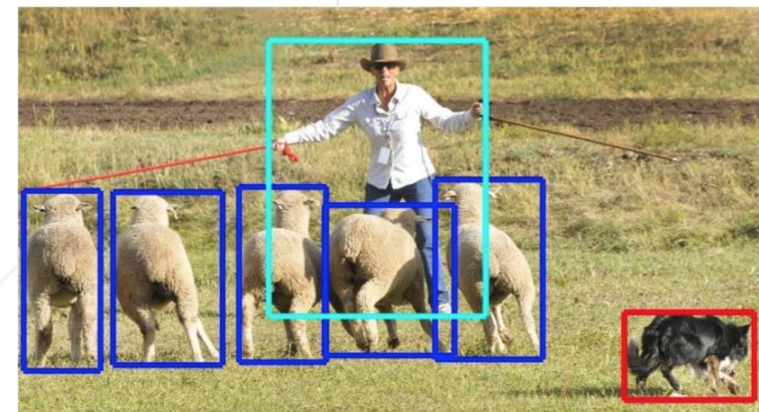
Part 4 **Future Research Topics**

Part 5 **Detection in Action**

- Type of computer vision tasks



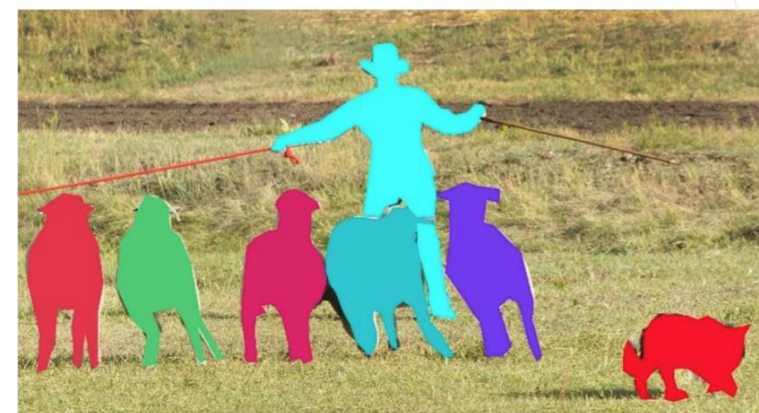
Image classification



Object detection

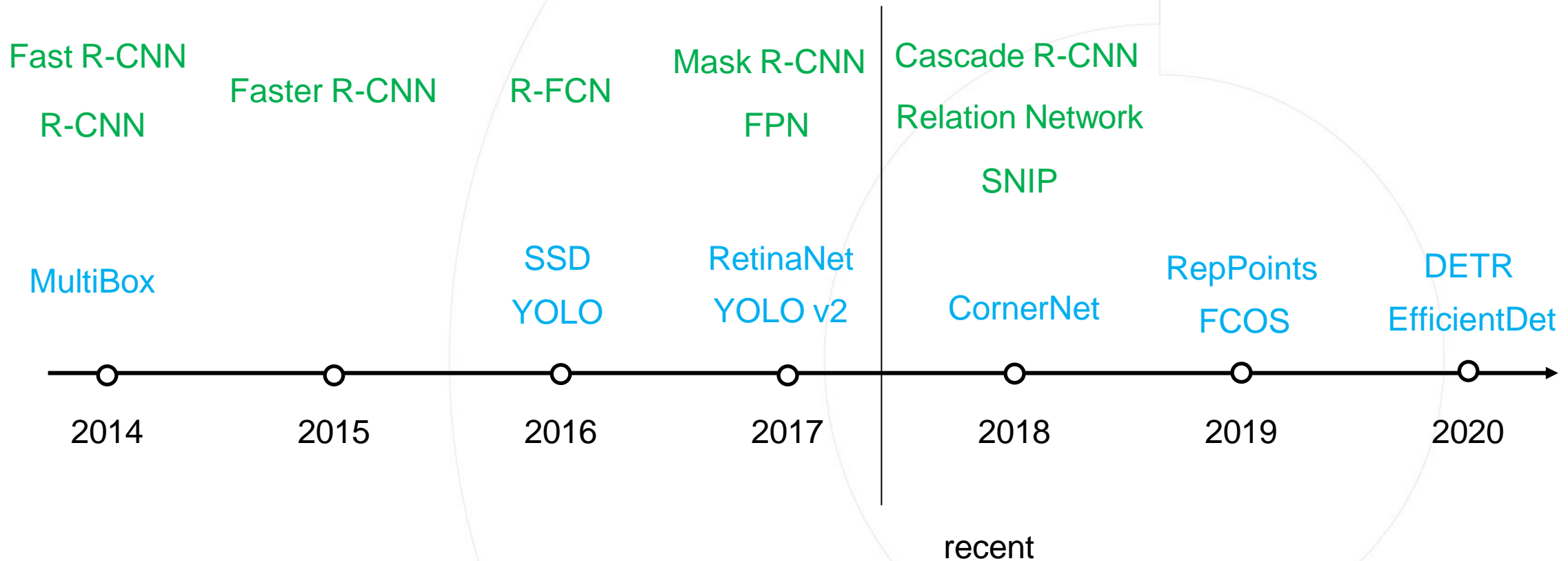


Semantic segmentation

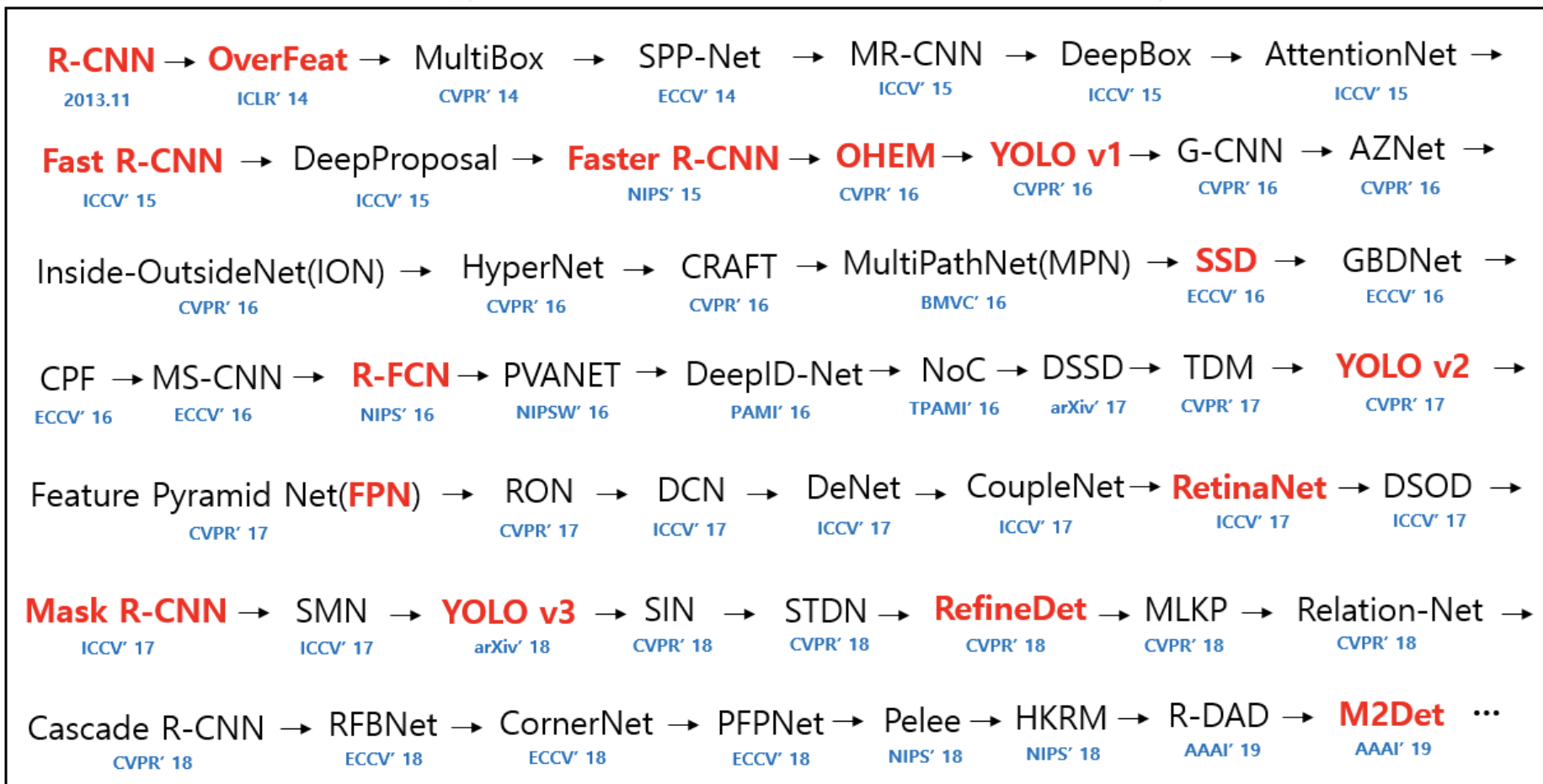


Instance segmentation

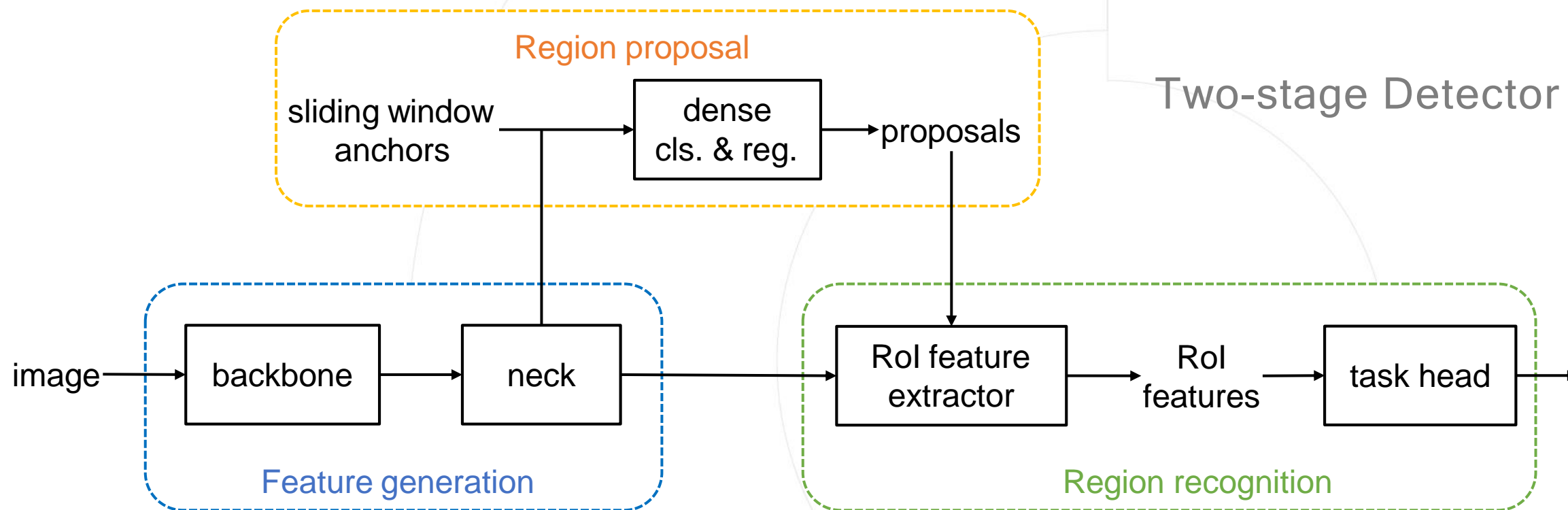
- Progress



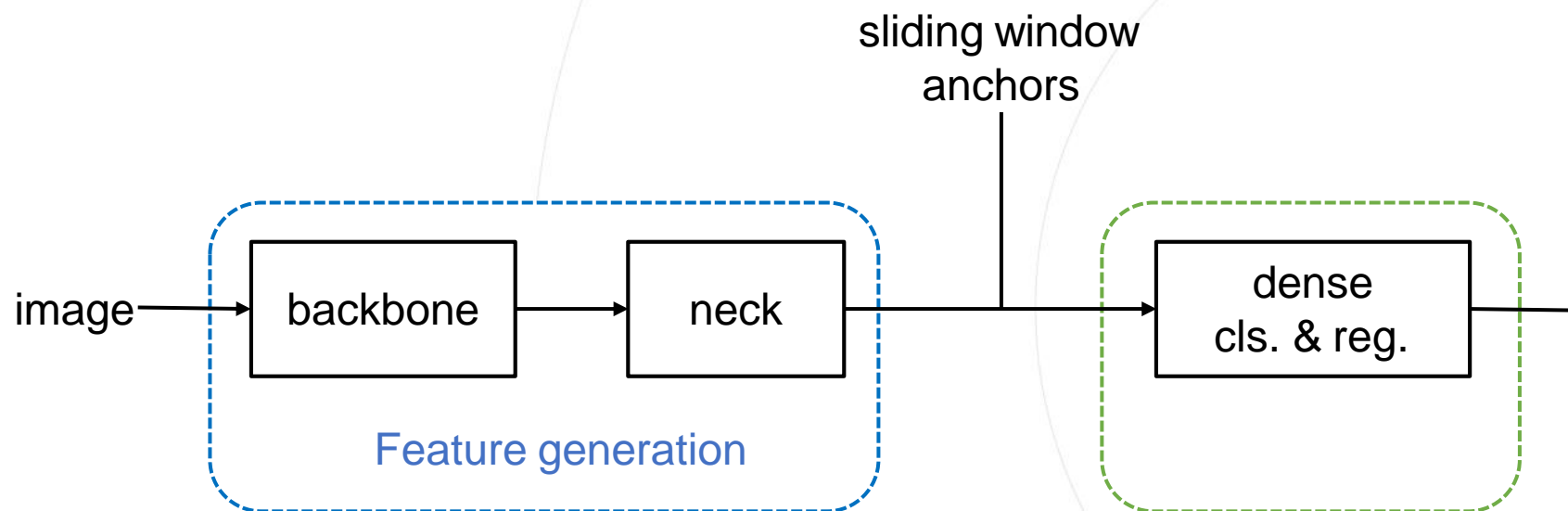
- History: a complete version



- General pipeline – two-stage pipeline



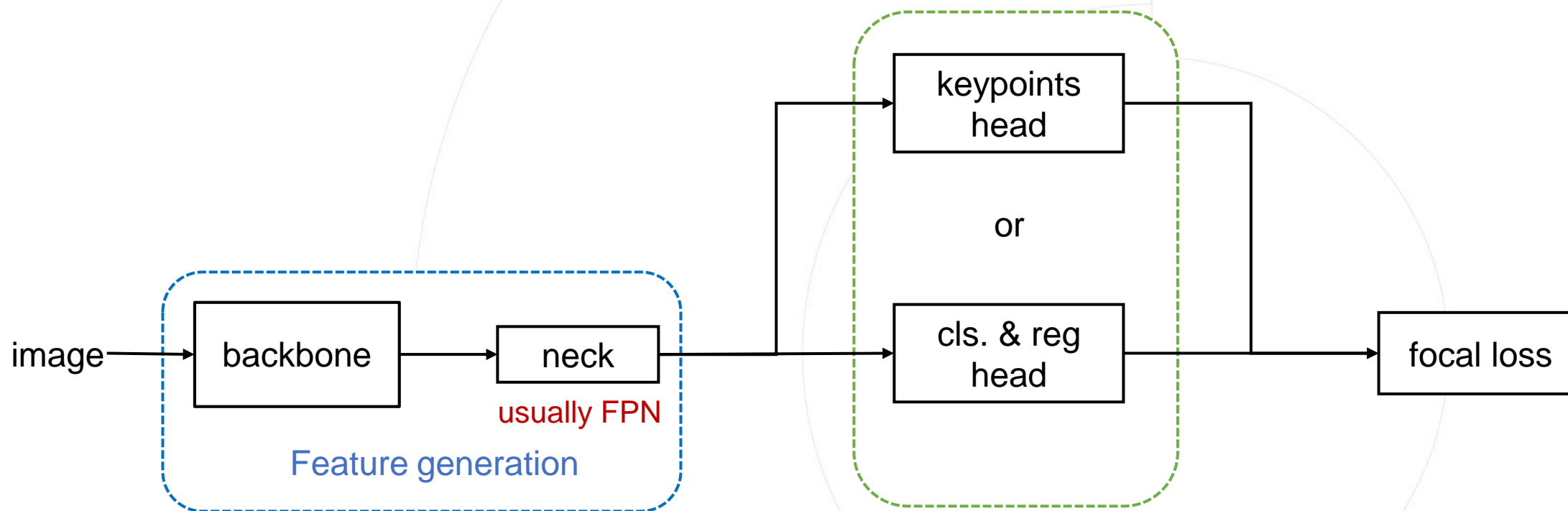
- General pipeline – one-stage pipeline



Single-stage Detector

- General pipeline – anchor-free pipeline

Anchor-free Detector



- Datasets and evaluation metrics
 - PASCAL VOC
 - MSCOCO
 - KITTI
 - Objects365
 - WIDER FACE,
 - mAP, FPPI
 - AP_{50} , AP_{75} , $AP@[.5:.95]$

- PASCAL VOC Detection Competition

- Homepage: <http://host.robots.ox.ac.uk/pascal/VOC/>

- 每一张图对应一个 xml 文件:

- bndbox 即 bounding box, 目标在照片中可见部分的包围框

- truncated 表明这个目标因为某种原因被截断

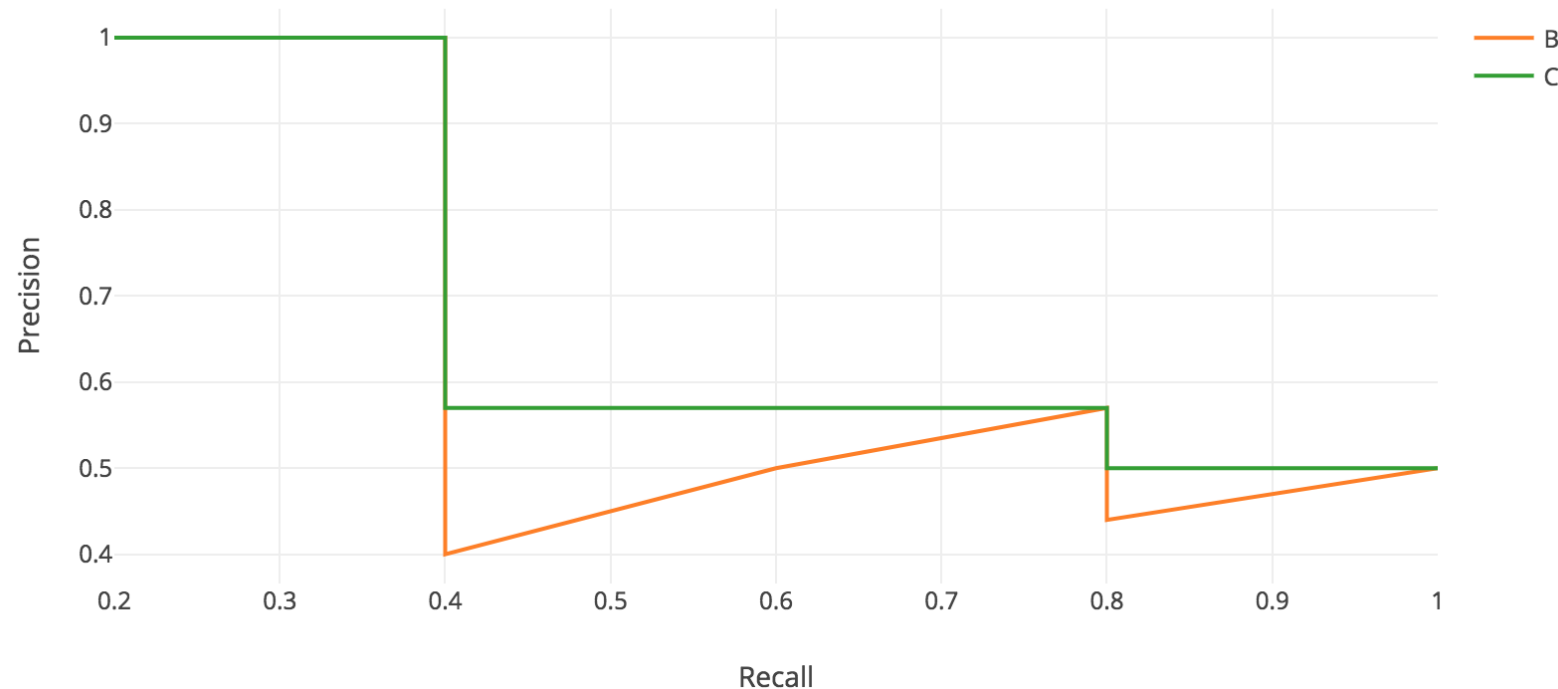
- occluded 表明这个目标的重要部分被遮挡

- difficult 表明这个目标识别难度较大

Year	Statistics
2007	20 classes: Person(1), Animal(6), Vehicle(7), Indoor(6) Train/validation/test: 9,963 images containing 24,640 annotated objects.
2012	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

- PASCAL VOC Detection Competition
 - interpolated average precision (before 2010)

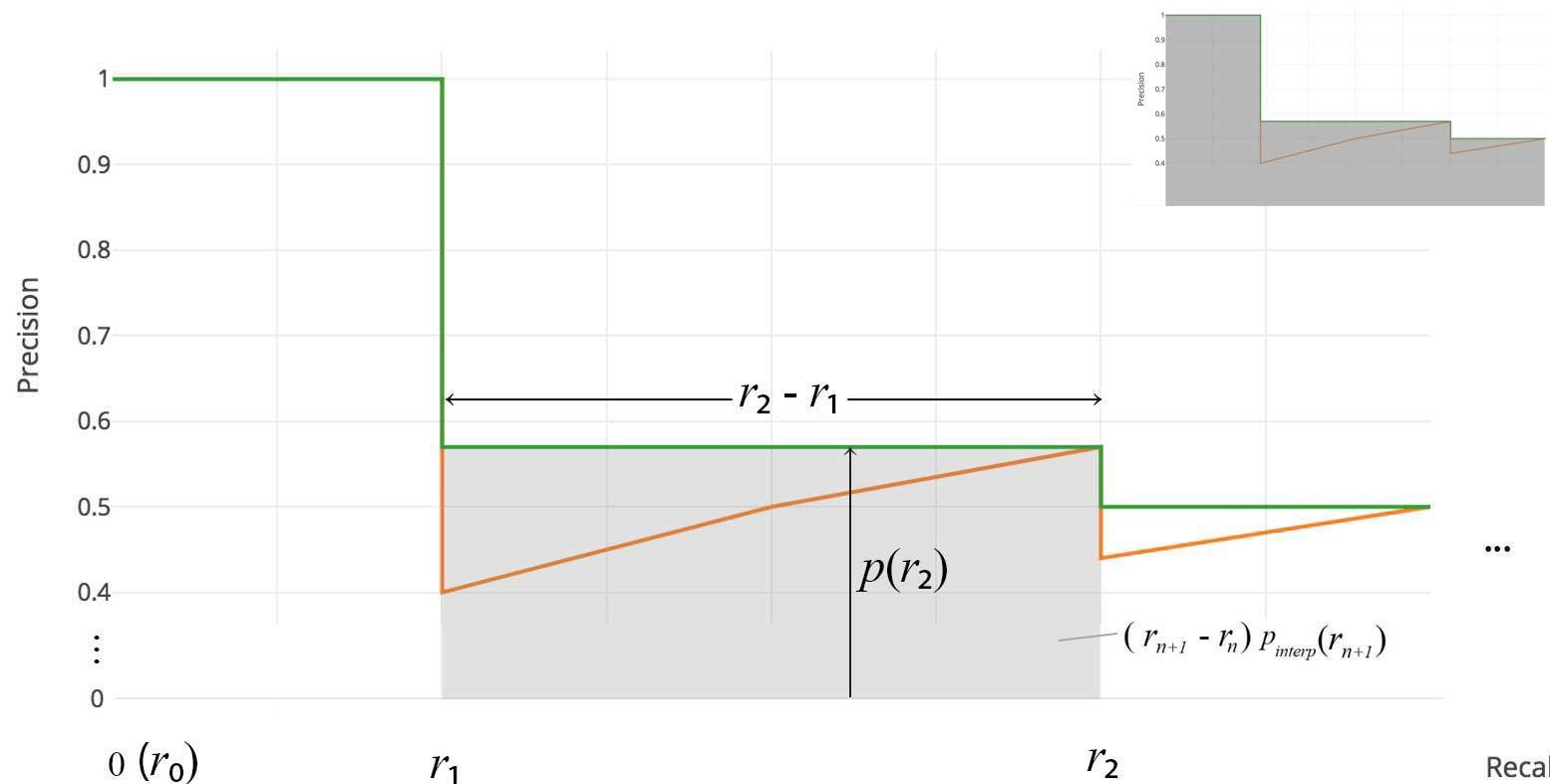
$$AP = \frac{1}{11} \sum_{0,0.1\dots 1.0} P_{smooth}(i)$$



- PASCAL VOC Detection Competition
 - 新 ap 指标 Area under curve AUC (after 2010)

$$AP = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

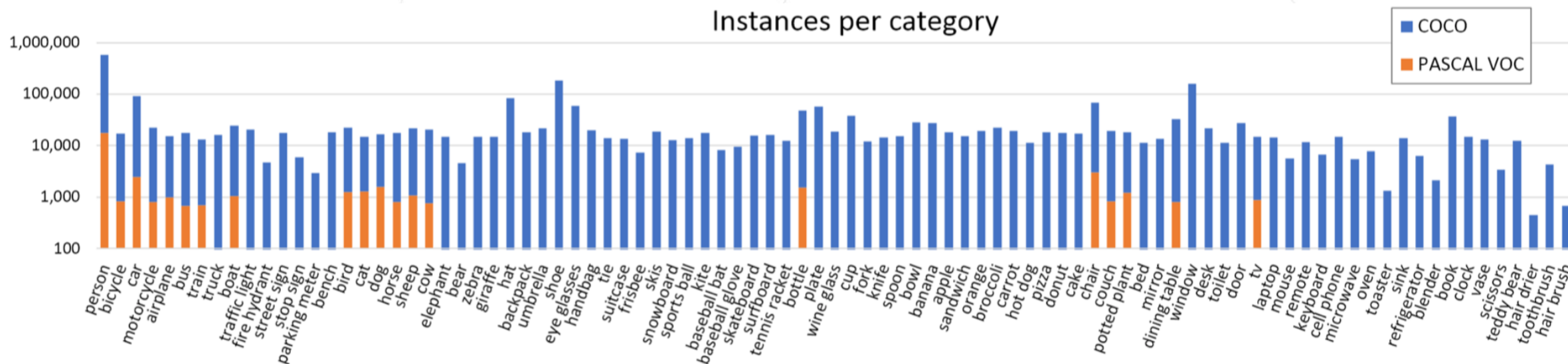


- Common Objects in Context (COCO)
 - Homepage: cocodataset.org
 - 使用 JSON 文件保存标注
 - images: 图片基本信息。“file_name” 为图片名称, “id” 为图片序号, “height” 和 “width” 为图像的高宽
 - annotations: 标注信息。“image_id” 为图片序号, “bbox” 为矩形框坐标 [x, y, w, h]。 “category_id” 为类别
 - categories: 对应类别信息。“id” 为类别序号, “name” 为类别名称。

- Common Objects in COntext

- The 2014 release contains 82,783 training, 40,504 validation, and 40,775 testing images
- 80 categories

Instances per category



- Common Objects in Context

注意与 PASCAL
VOC 中 AP 的区别

Matrix	Details
AP	AP at IoU=.50:.05:.95 (primary challenge metric)
AP ₅₀	AP at IoU=.50 (PASCAL VOC metric)
AP ₇₅	AP at IoU=.75
AP _{small}	AP for small objects: area < 32 ²
AP _{medium}	AP for medium objects: 32 ² < area < 96 ²
AP _{large}	AP for large objects: area > 96 ²

- Objects365
 - 11 个大类: human and related accessories, living room, clothes, kitchen, instrument, transportation, bathroom, electronics, food (vegetables), office supplies, and animal
 - 365 个小类, 覆盖 PASCAL VOC 和 COCO

Dataset	Images	Boxes	Categories	Boxes/img	Fully Annotated
Pascal VOC	11.5k	27k	20	2.4	Yes
ImageNet All	477k	534k	200	1.1	Yes
ImageNet Dense	80k	186k	200	2.3	Yes
COCO	123k	896k	80	7.3	Yes
OpenImages	1,515k	14,815k	600	9.8	Partial
Objects365	638k	10,101k	365	15.8	Yes

- KITTI

- 自动驾驶领域的数据集
- Object Detection Evaluation 方向包括 2D、3D和鸟瞰视角的 benchmark
- 包括汽车, 行人, 自行车分类
- calib.txt 保存标定参数
- label.txt 保存标注结果

```
Truck 0.00 0 -1.57 599.41 156.40 629.75 189.25 2.85 2.63 12.34 0.47 1.49 69.44 -1.56
Car 0.00 0 1.85 387.63 181.54 423.81 203.12 1.67 1.87 3.69 -16.53 2.39 58.49 1.57
Cyclist 0.00 3 -1.65 676.60 163.95 688.98 193.93 1.86 0.60 2.02 4.59 1.32 45.84 -1.55
DontCare -1 -1 -10 503.89 169.71 590.61 190.13 -1 -1 -1 -1000 -1000 -1000 -10
```



Outline

Part 1 **Introduction to Object Detection**

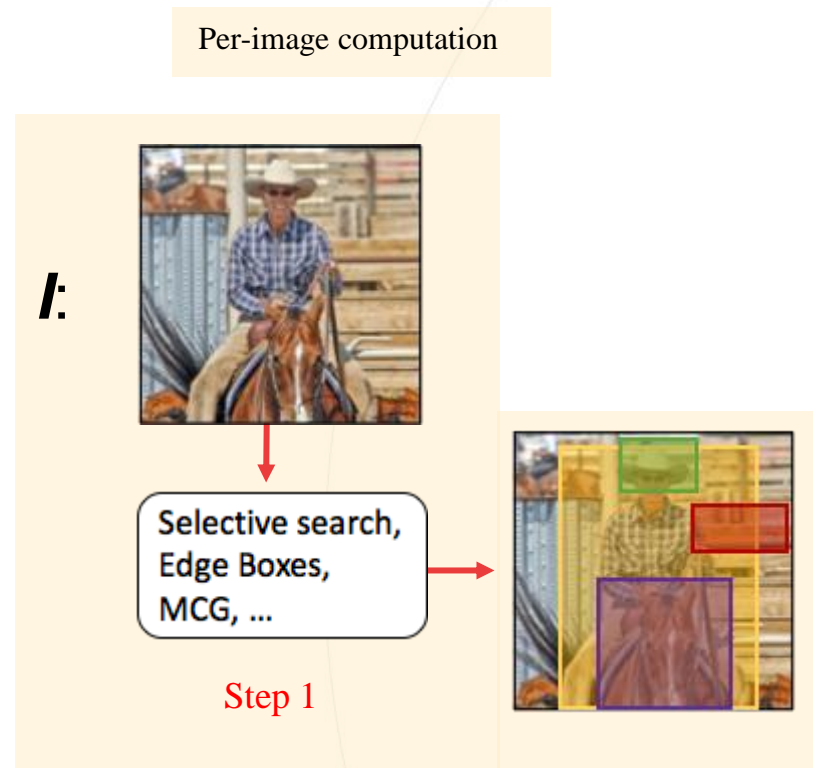
Part 2 **Anchor-based Solutions**

Part 3 **Anchor-free Solutions**

Part 4 **Future Research Topics**

Part 5 **Detection in Action**

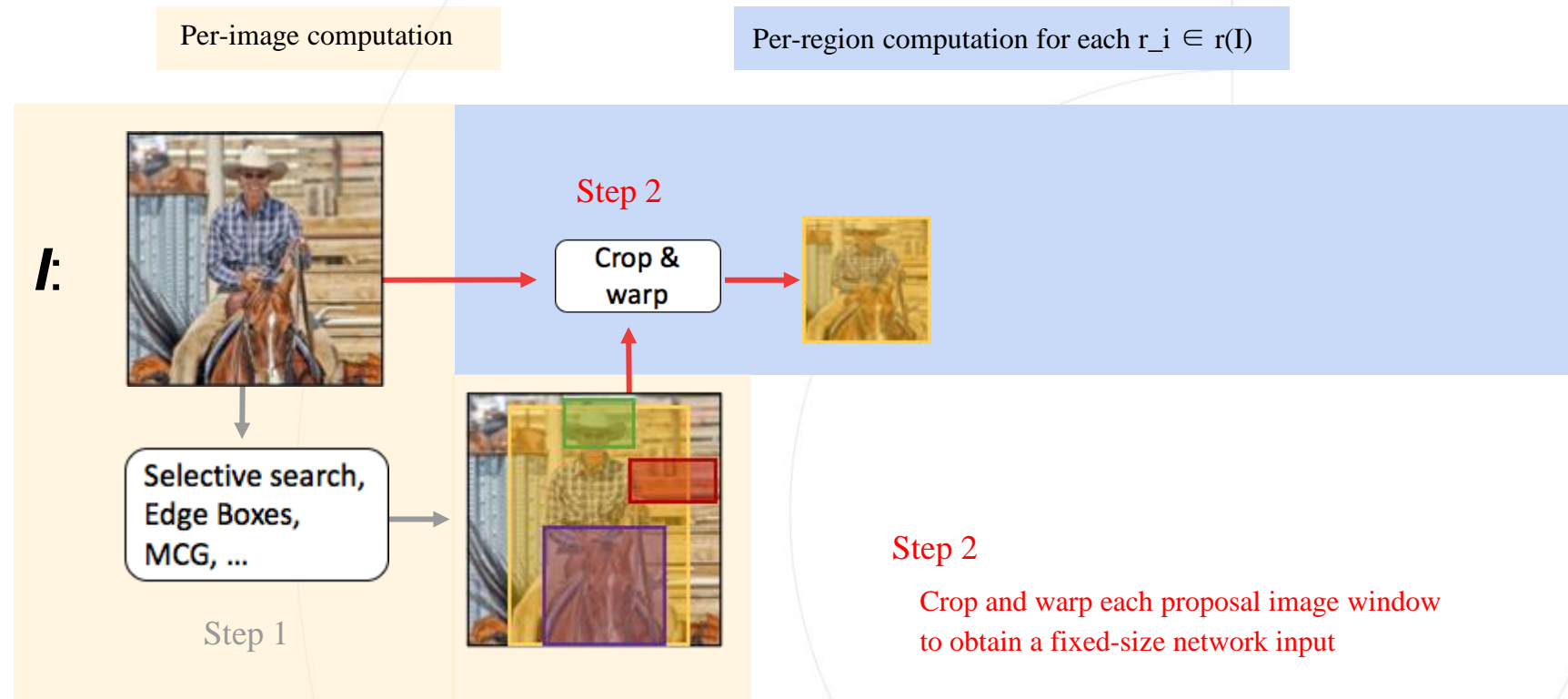
- RCNN method



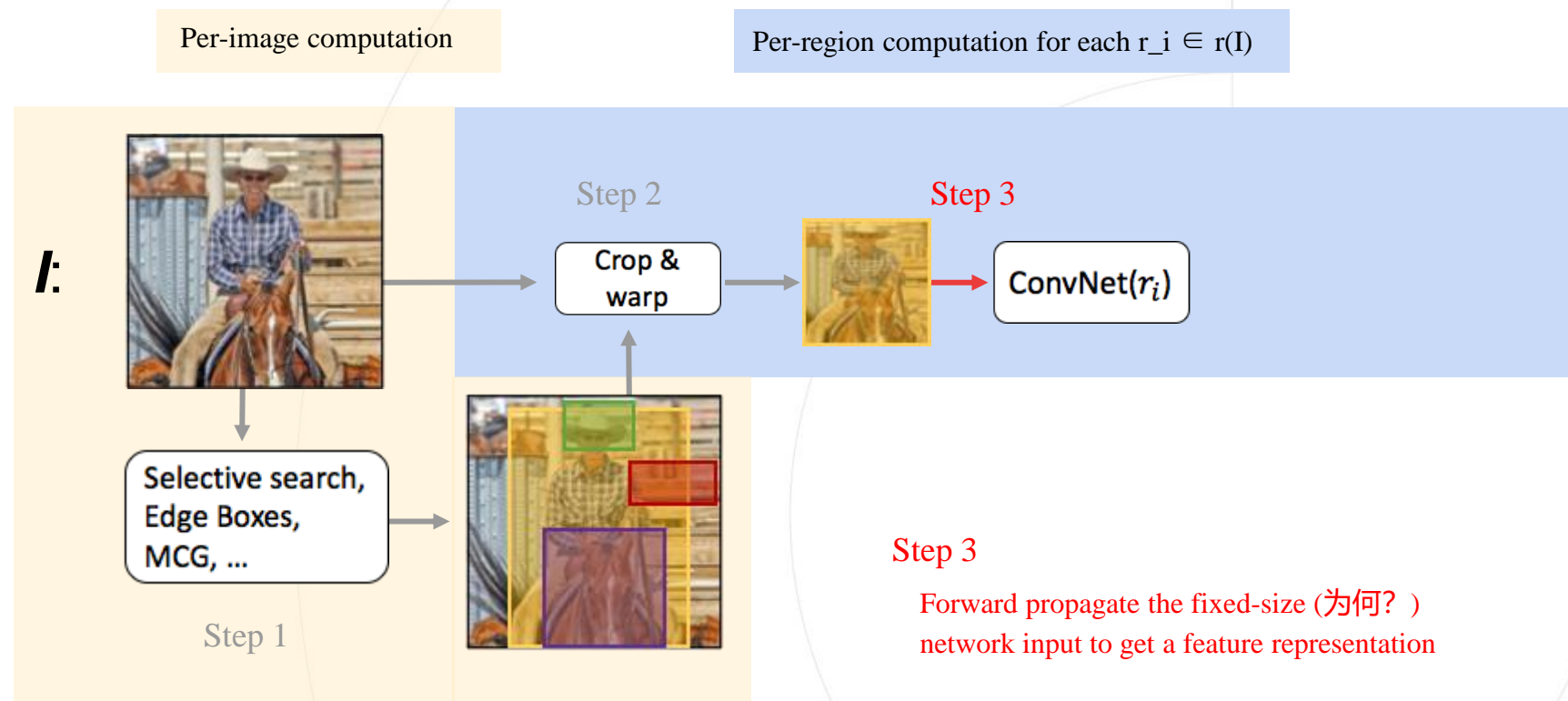
Step 1

Use an off-the-shelf region/object/detection proposal algorithm (~2k proposals per image)

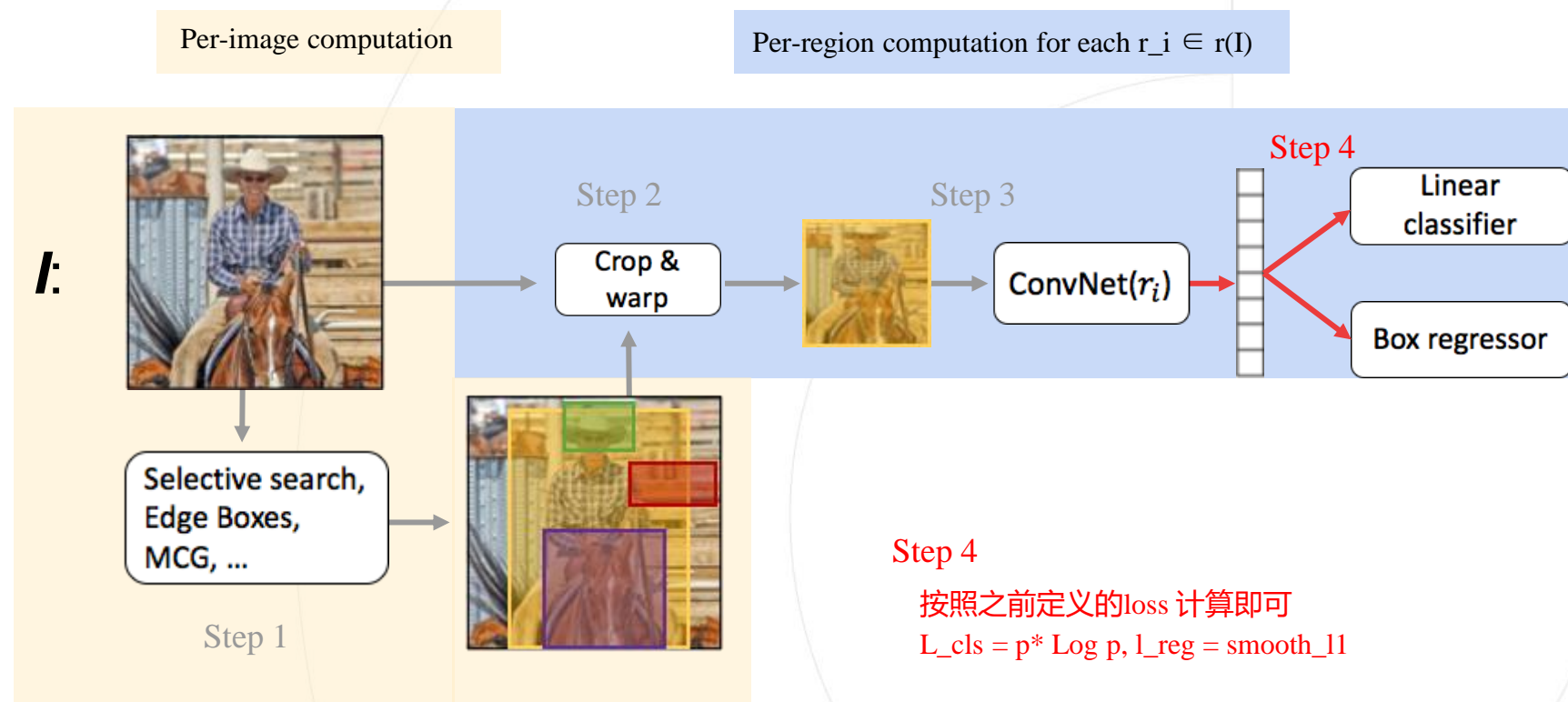
- RCNN method



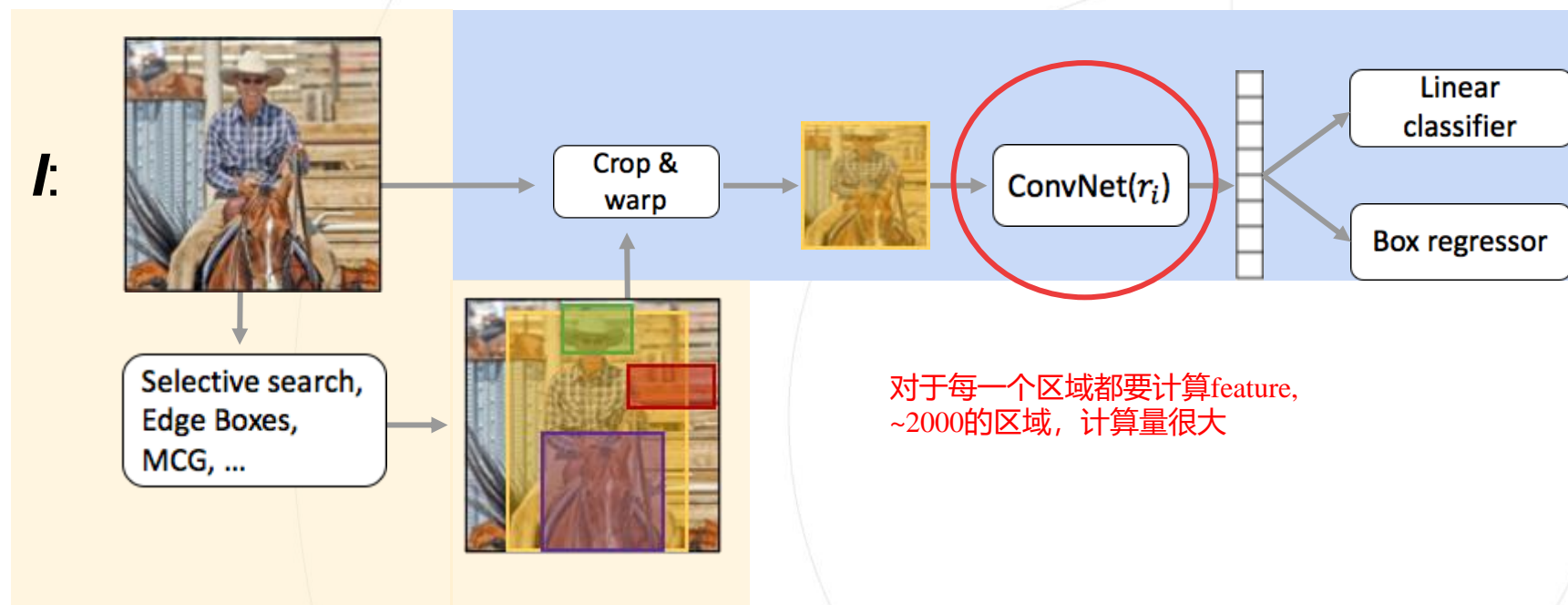
- RCNN method



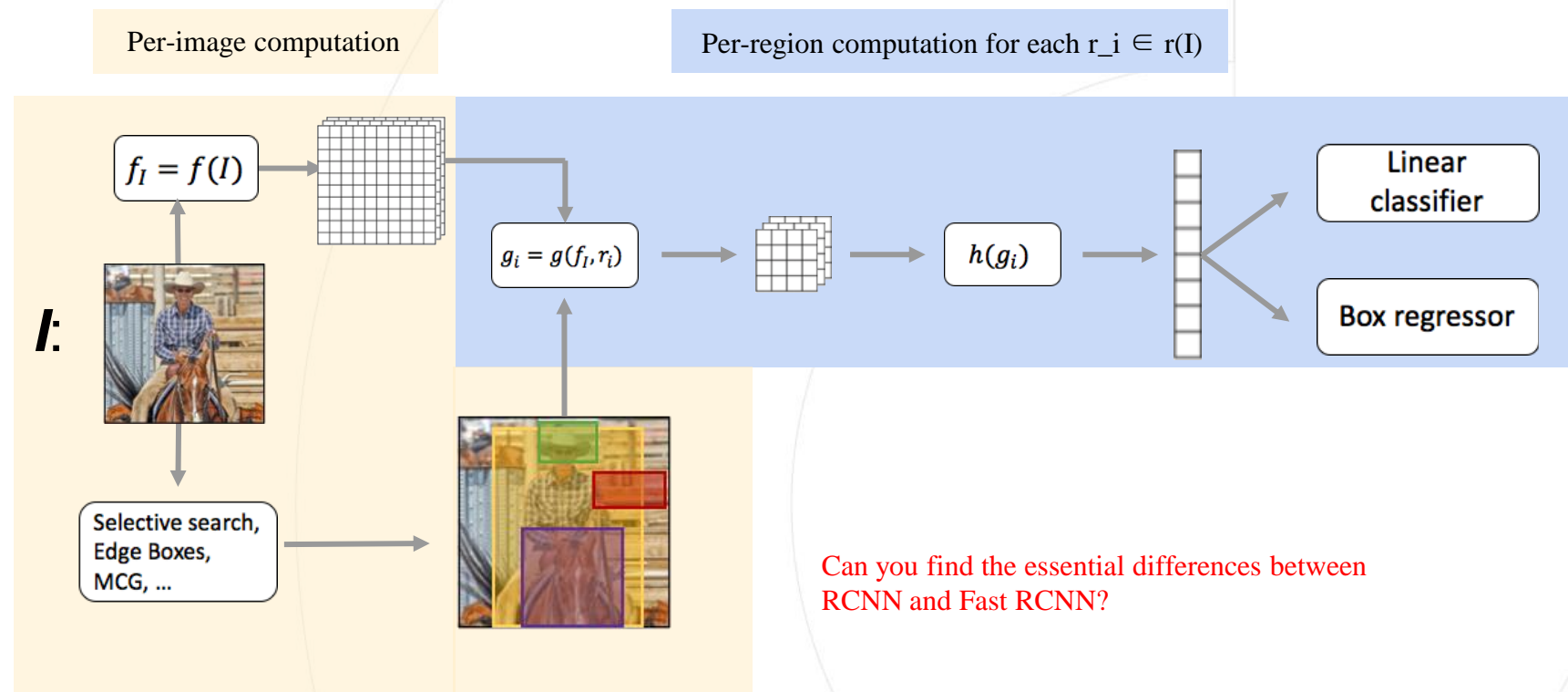
- RCNN method



- The bottleneck of RCNN

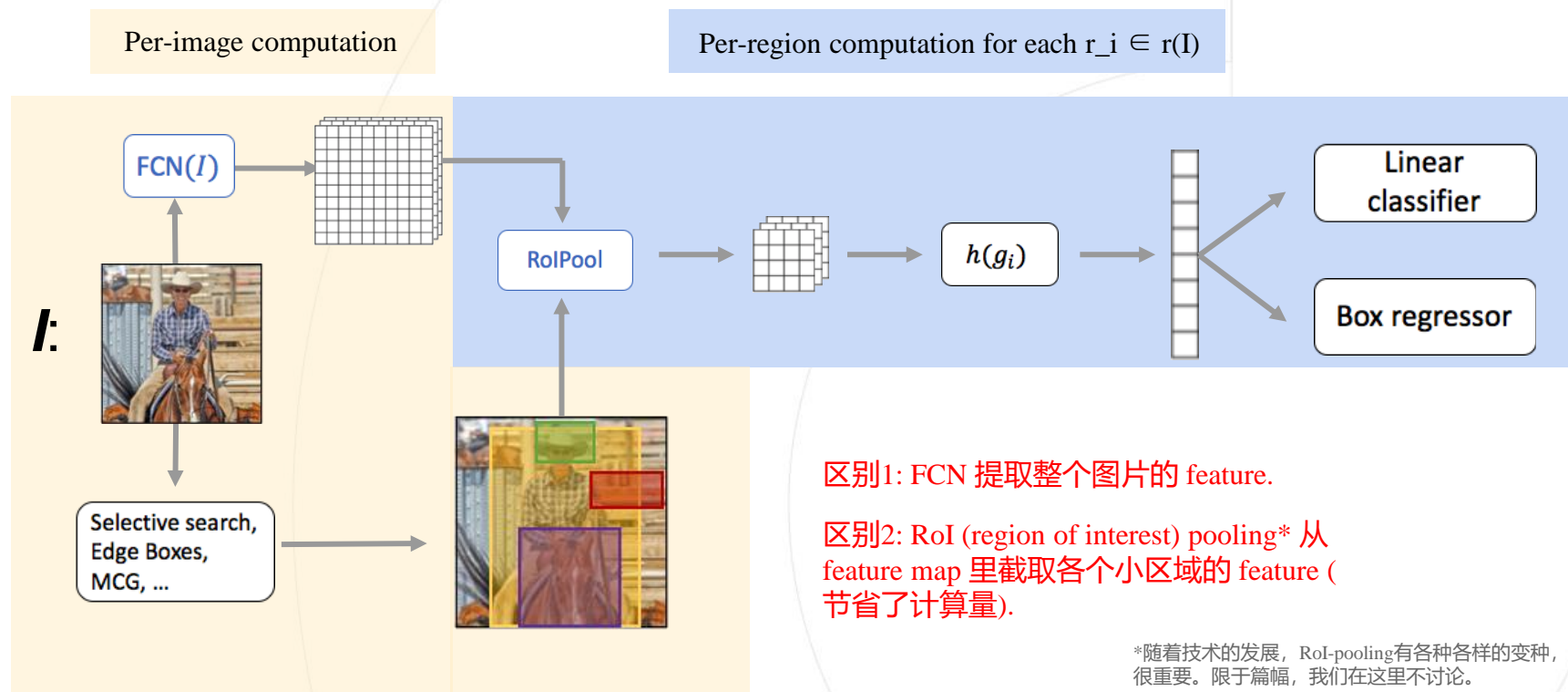


- Fast RCNN

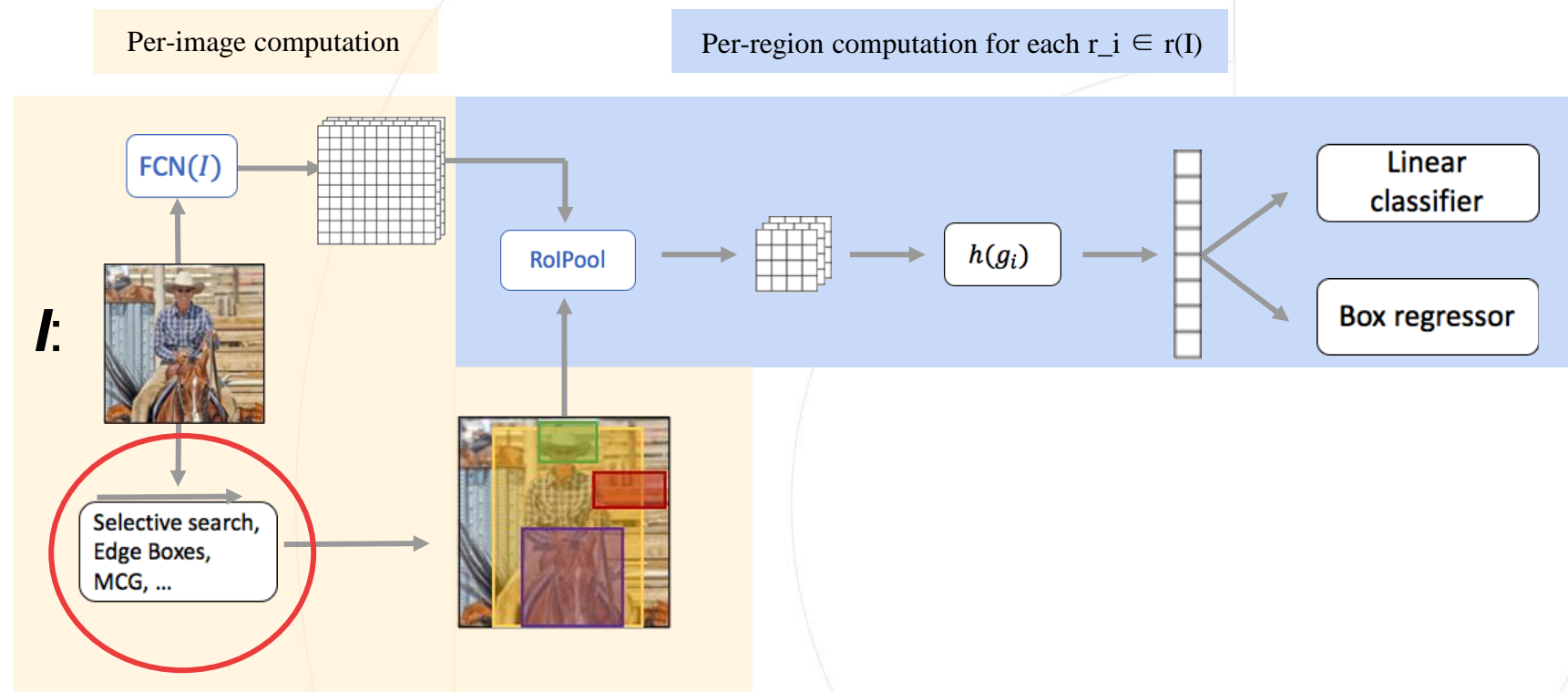


Can you find the essential differences between RCNN and Fast RCNN?

- Fast RCNN

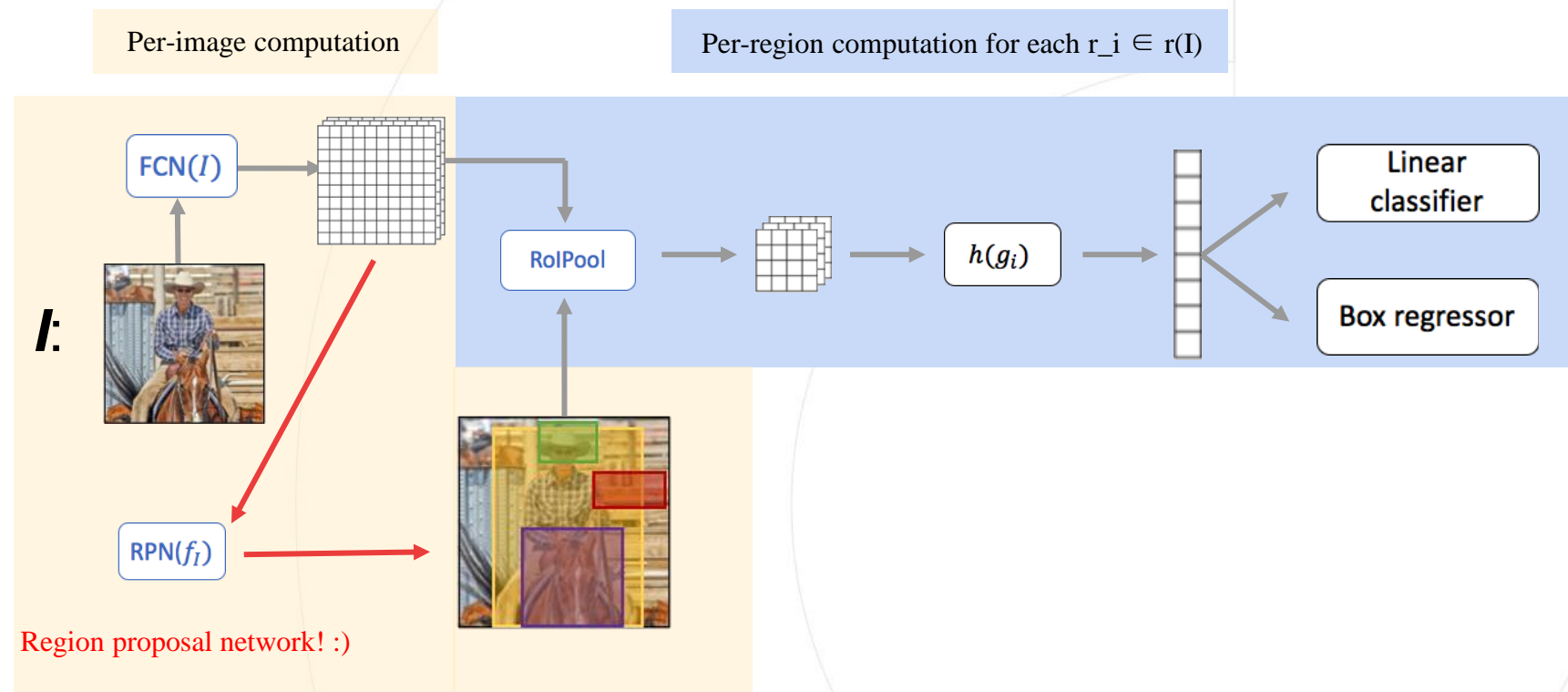


- The **bottleneck** of Fast RCNN



Region proposals have very poor recall
(ok for PASCAL VOC, major bottleneck
for COCO) Also, they can be slow

- The solution: Faster RCNN

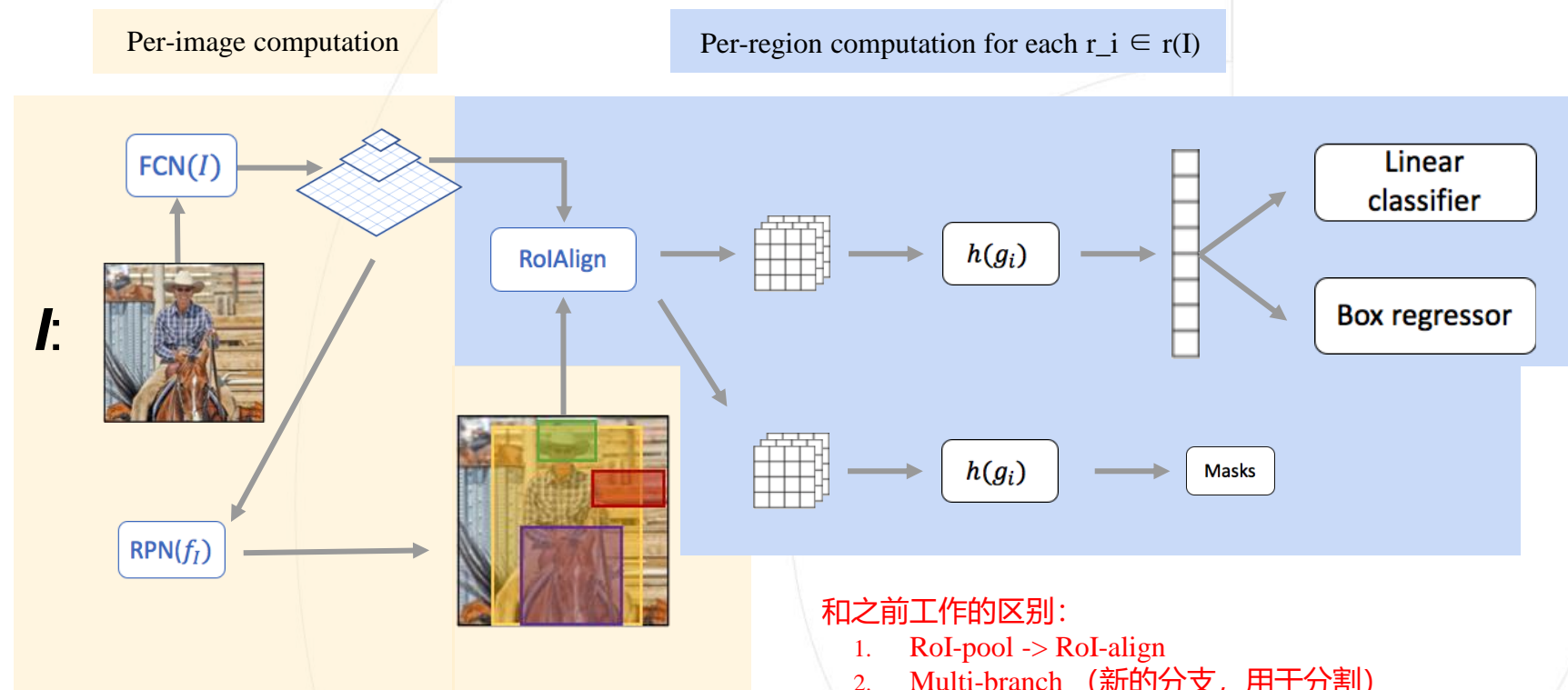


Region proposal network! :)

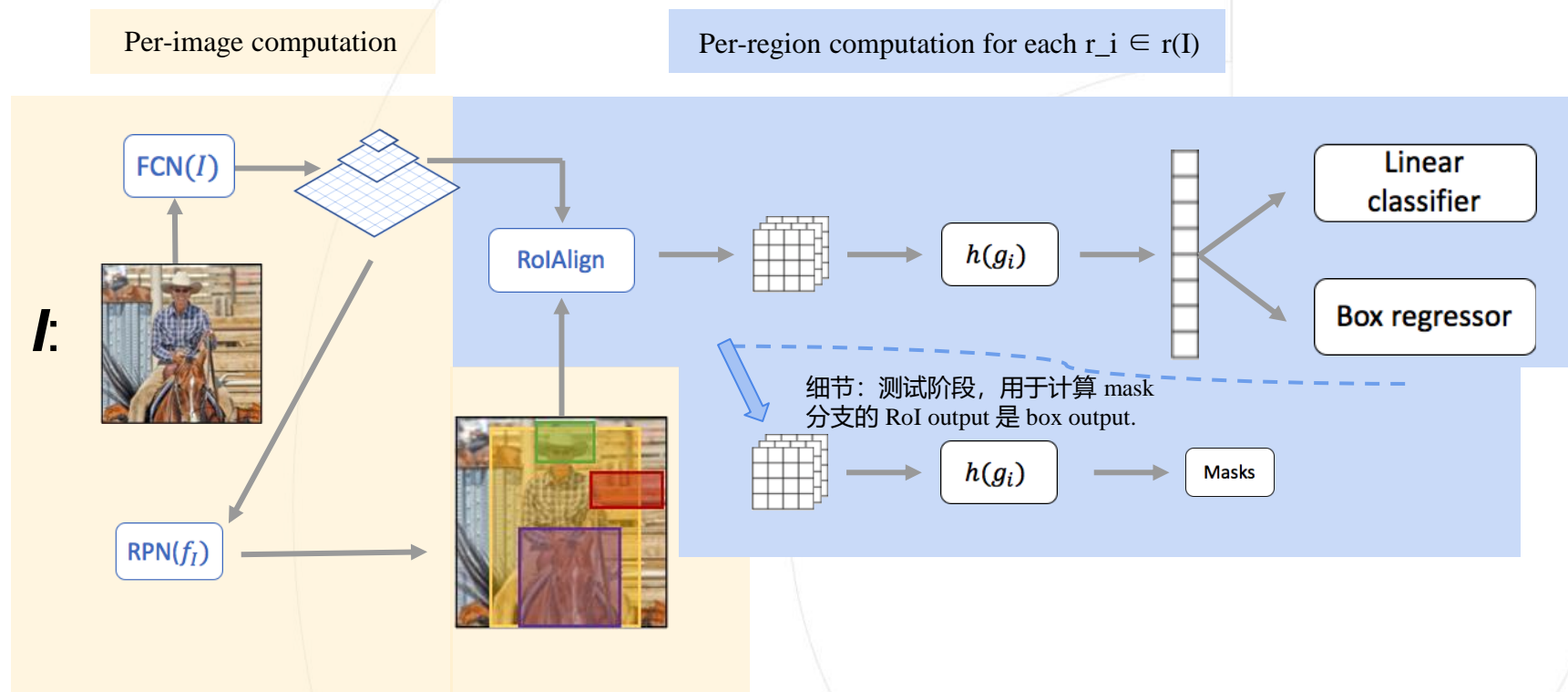
Learned proposals **shares computation** with whole-image network

Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. PAMI, 2016, 39(6): 1137-1149.

- Mask RCNN

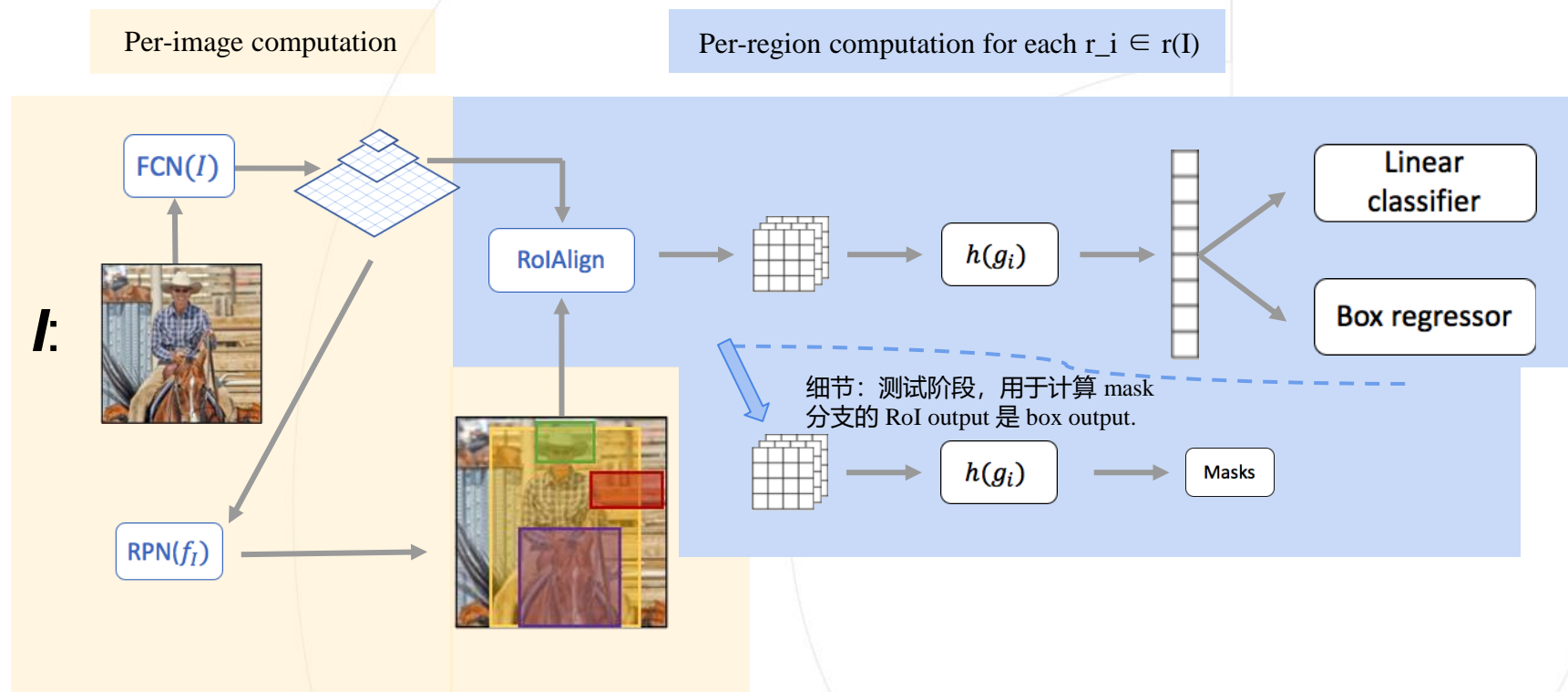


- Mask RCNN



细节：测试阶段，用于计算 mask 分支的 RoI output 是 box output.

- Mask RCNN



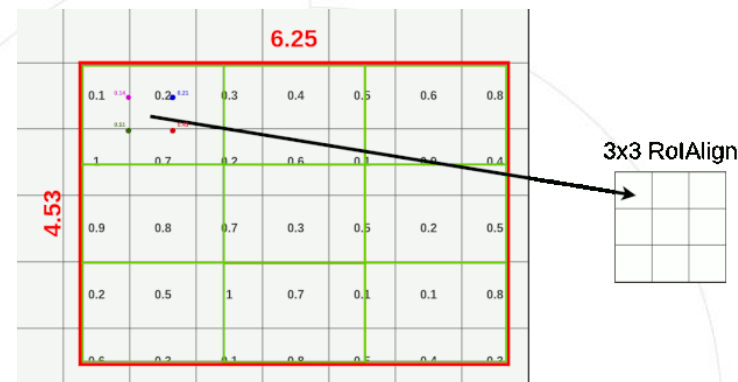
- Can you identify deficiencies of RoI Pooling?

4x6 RoI

0.1	0.2	0.3	0.4	0.5	0.6
1	0.7	0.2	0.6	0.1	0.9
0.9	0.8	0.7	0.3	0.5	0.2

(a) RoI Pooling

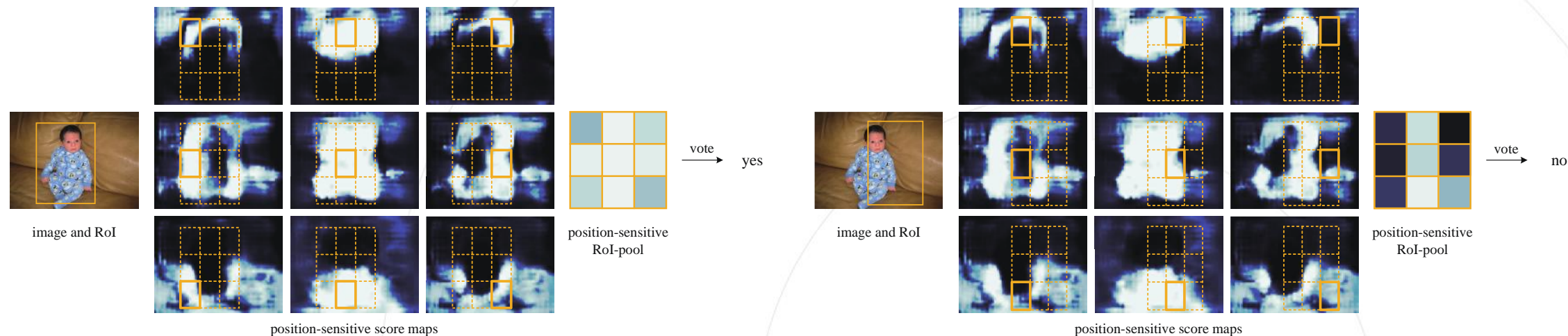
3x3 RoI Pooling



(b) RoI Align

RoI Pooling 需要在 feature map 上对坐标进行量化。Feature map 上 (x1: 9.25, y1: 6, x2: 15.5, y: 10.5) 的 RoI 将被量化成 4*6 执行 Pooling 时需要第二次量化。例如上图中由于 $4/3=1.33$ ，最后一行 feature 将被丢弃 两次量化后，实际使用的 feature 是 (x1: 9, y1: 6, x2: 15, y: 10) 中的

- Position-sensitive ROI Pooling



(a) Position-sensitive ROI Pooling

Problem: faster R-CNN 经过 RoI pooling 后, ConvNet 是不共享的, 导致冗余的计算。如果一副图片有 500 个 region proposals, 那么需要分别进行 500 次卷积操作

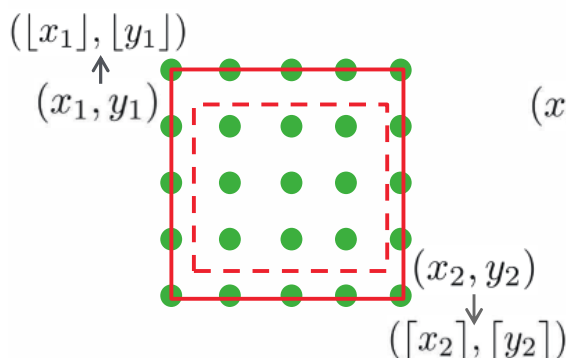
Motivation: 将 RoI pooling 后的 FC layer 换成 Conv layer 并且共享, 从而加速计算

基于 feature maps 提取 $(C+1)$ -dim position-sensitive score maps, 这个 score maps 对于所有 proposals 是共享的

Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[J]. arXiv preprint arXiv:1605.06409, 2016.

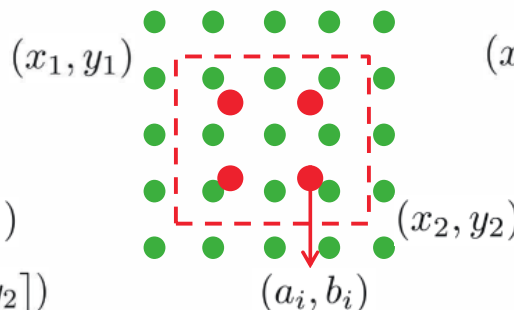
- Precise RoI Pooling

1. RoI Pooling



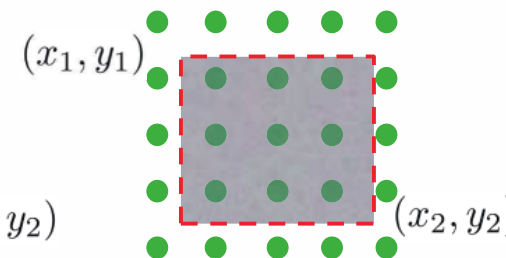
$$\frac{\sum_{i=\lfloor x_1 \rfloor}^{\lfloor x_2 \rfloor} \sum_{j=\lfloor y_1 \rfloor}^{\lfloor y_2 \rfloor} w_{i,j}}{(\lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1) \times (\lfloor y_2 \rfloor - \lfloor y_1 \rfloor + 1)}$$

2. RoI Align



$$\sum_{i=1}^N f(a_i, b_i) / N$$

3. PrRoI Pooling



$$\frac{\int_{y_1}^{y_2} \int_{x_1}^{x_2} f(x, y) dx dy}{(x_2 - x_1) \times (y_2 - y_1)}$$

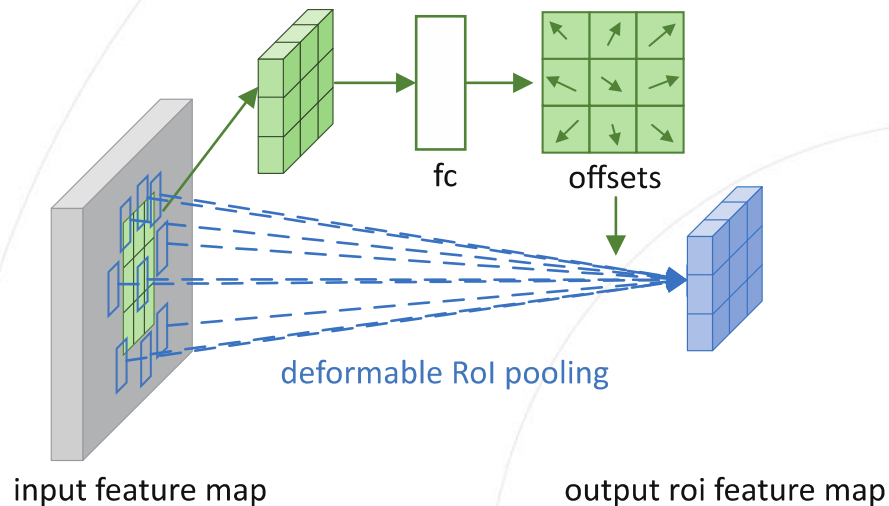
(b) PrRoI Pooling

Problem: RoI Pooling 的量化问题

Motivation: IoU-Net directly estimates $\text{IoU}(\text{box}_{\text{det}}, \text{box}_{\text{gt}})$. PrRoI Pooling **avoids any quantization** of coordinates and has a **continuous gradient** on bounding box coordin

PrRoI Pooling 将 feature maps 连续化, 通过**积分求和除以面积**得到 Pooling 结果

- Deformable RoI Pooling



(c) Deformable RoI Pooling

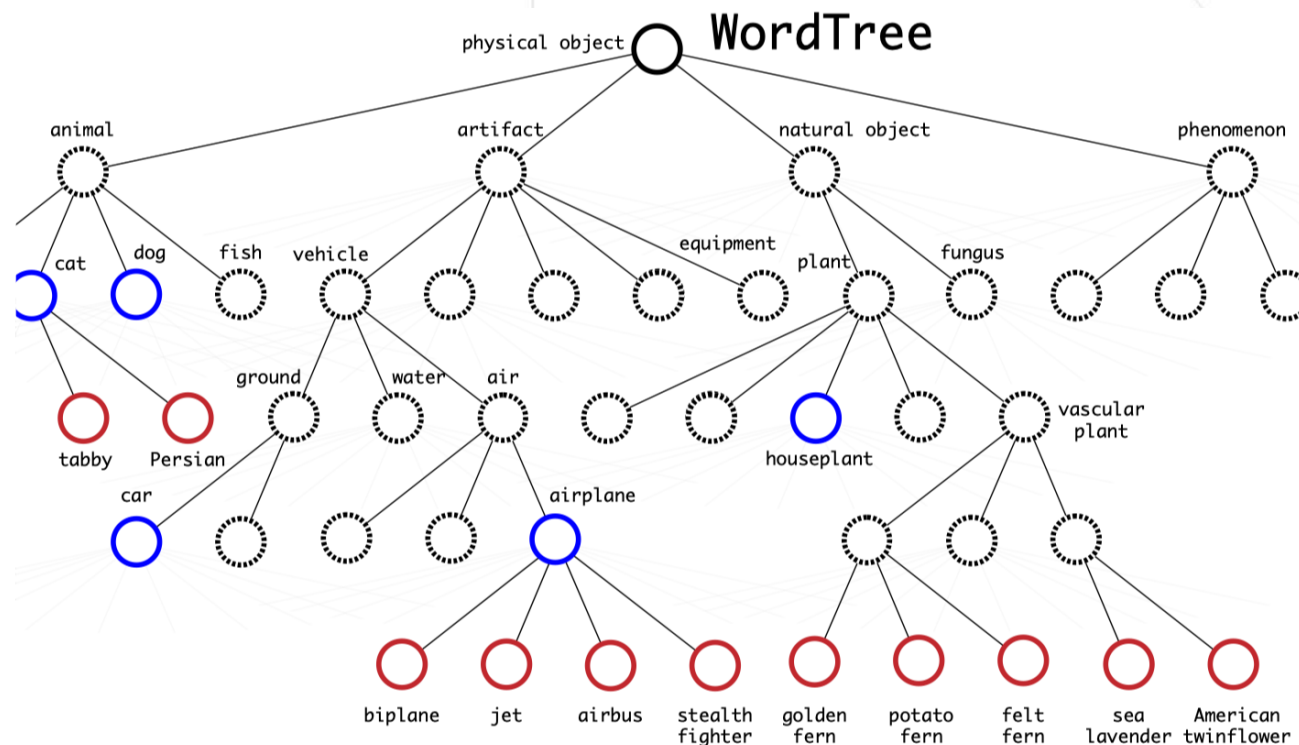
Problem: RoI Pooling 无法考虑不规则目标

Motivation: 给 RoI Pooling 中每一个 bin 加入一个 offset 来产生偏移

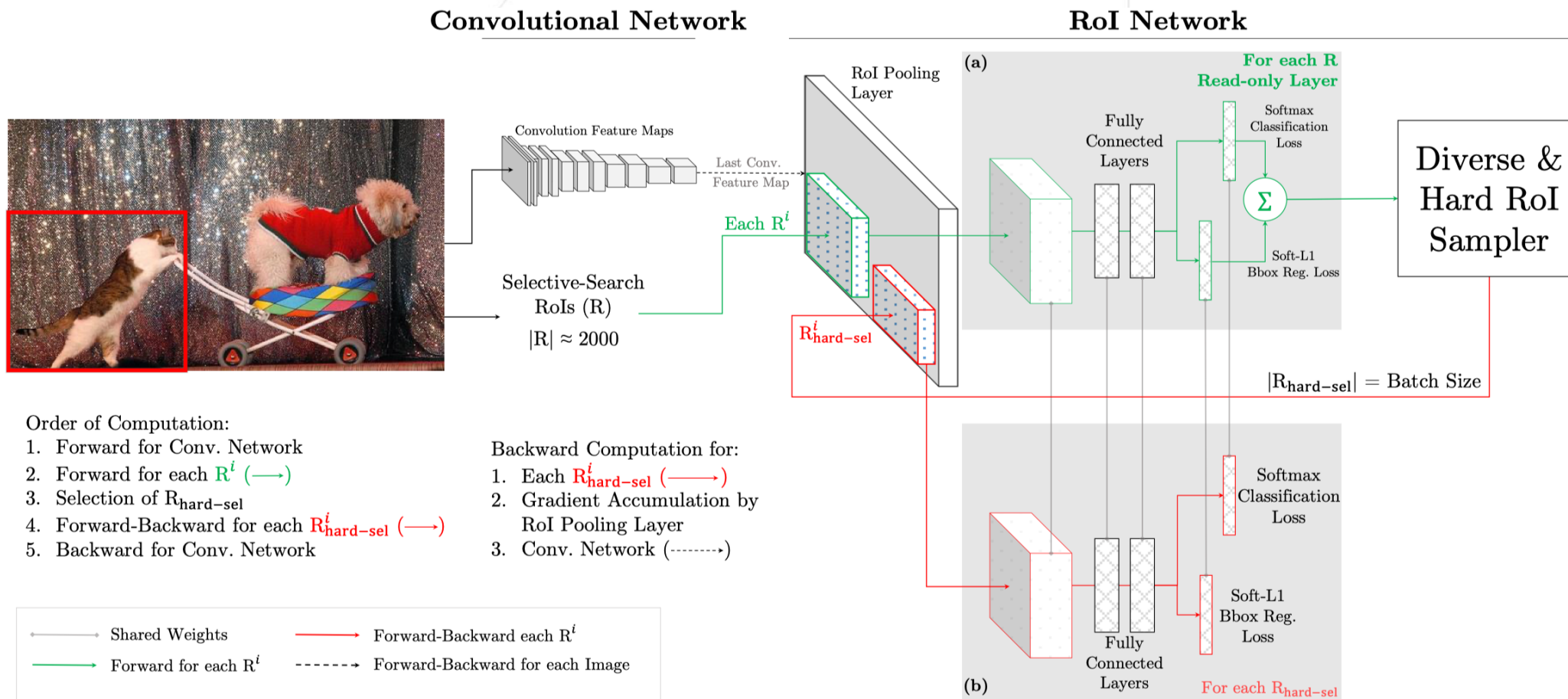
首先使用 RoI Pooling 生成 pooled feature maps; 一个 FC layer 生成 normalized offsets 给每一个 bin; 使用 offsets 即可在 input feature map 上做 deformable RoI Pooling

- 样本不均衡问题研究历史（在 two stage 方法中通过 two-stage cascade 和 sampling heuristics 避免
 - Yolo2 提出了 Hierarchical classification 方法，通过 Data Augmentation 达到局部平衡，并做局部 softmax。（不同类别样本均衡=性能一致？）
 - OHEM 提出对 classification loss 较大的样本做 back propagation。（loss 依然是类别 balanced）
 - 引出类别 imbalanced 的 Focal loss

- Hierarchical classification
 - WordTree 中每个节点的子节点属于同一子类，对同一子类进行 softmax 处理
 - 在给出某个类别的预测概率时，需要找到其所在的位置，遍历这个 path，然后计算 path 上各个节点的概率之积

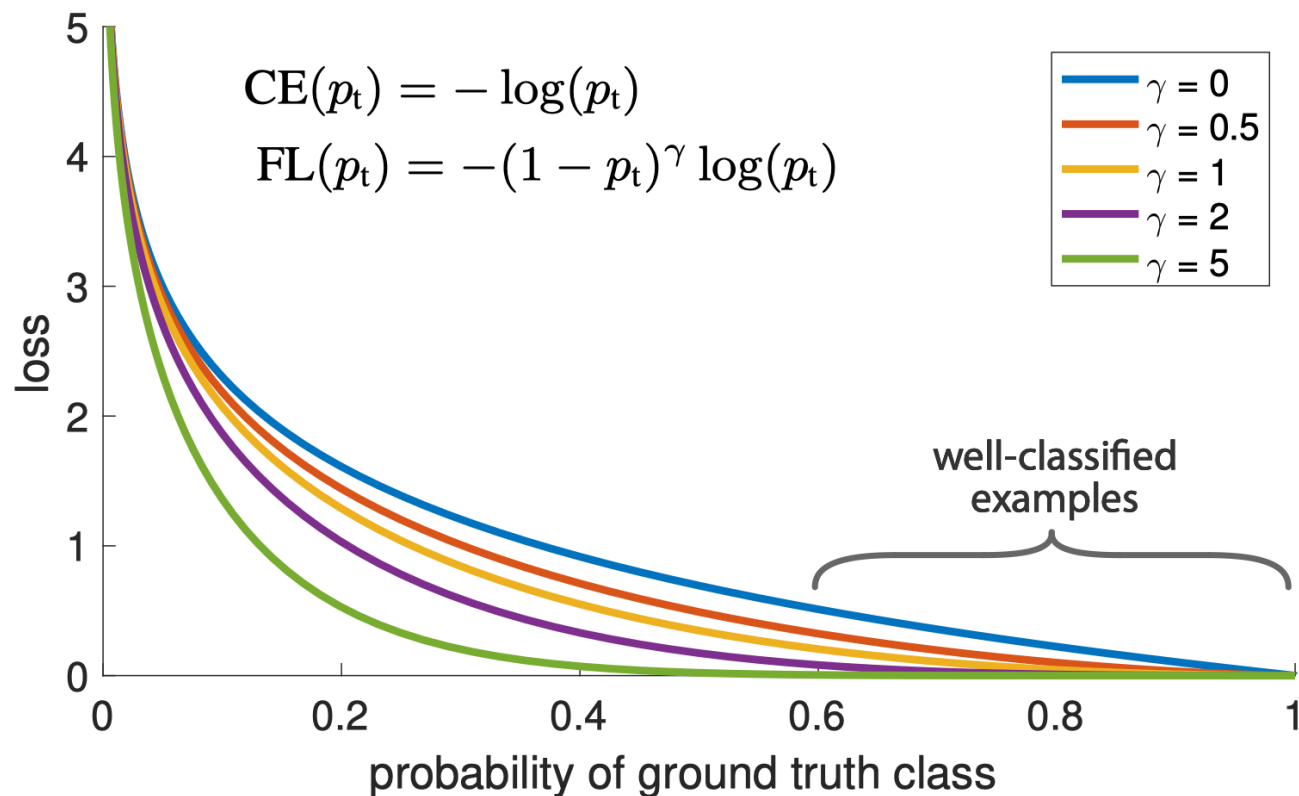


- OHEM



Shrivastava A, Gupta A, Girshick R. Training region-based object detectors with online hard example mining. CVPR2016

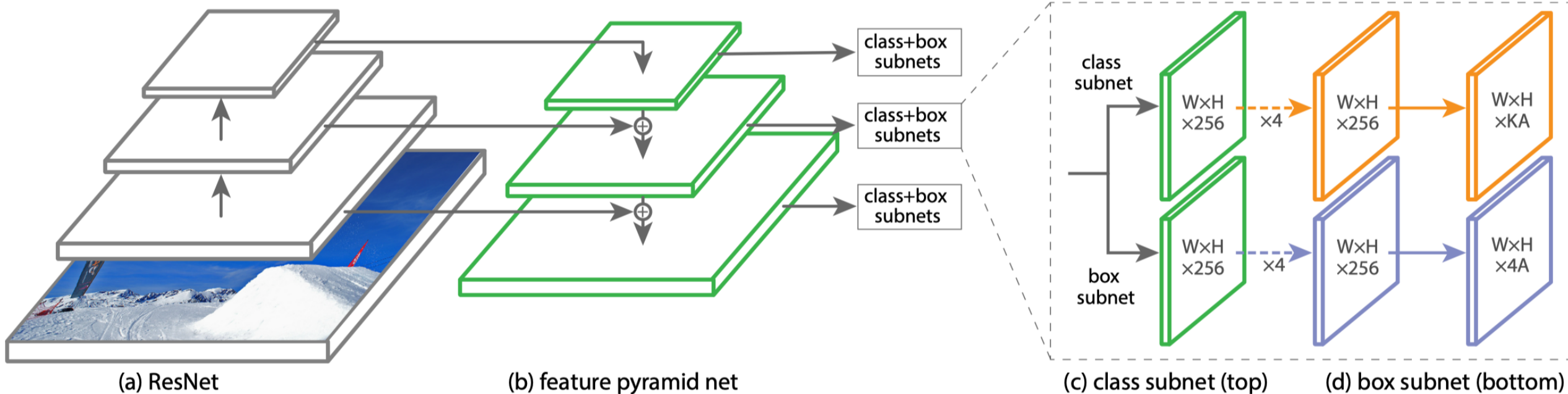
- Focal loss



- 将分类难易体现在 loss 上
- 当 negative examples 远比 positive examples 多时, 负类的 $(1 - p_t)^\gamma$ 较小, 模型倾向于正类样本

- RetinaNet

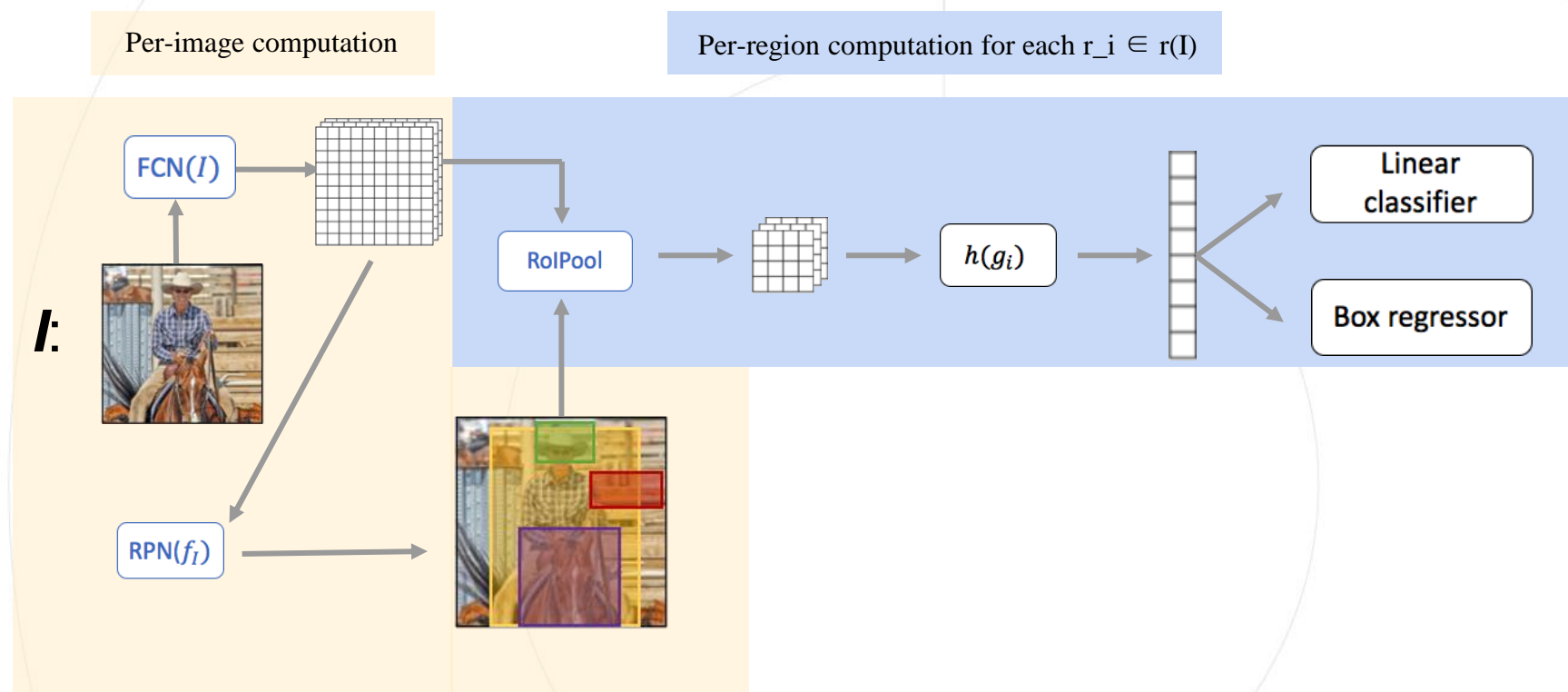
- 基于 FPN 和 Focal loss, 提出了 RetinaNet, 该网络对 anchor-free 方法有很大影响



- The problem in anchor-based methods

1) Huge **imbalance** between positive and negative anchor boxes and **slows** down training. Anchor-based methods need a very large set of anchor boxes, while only a tiny fraction of anchor boxes will overlap with ground truth

2) The use of anchor boxes introduces **many hyperparameters** and design choices, such as how many boxes, what sizes, and what aspect ratios.





Outline

Part 1 **Introduction to Object Detection**

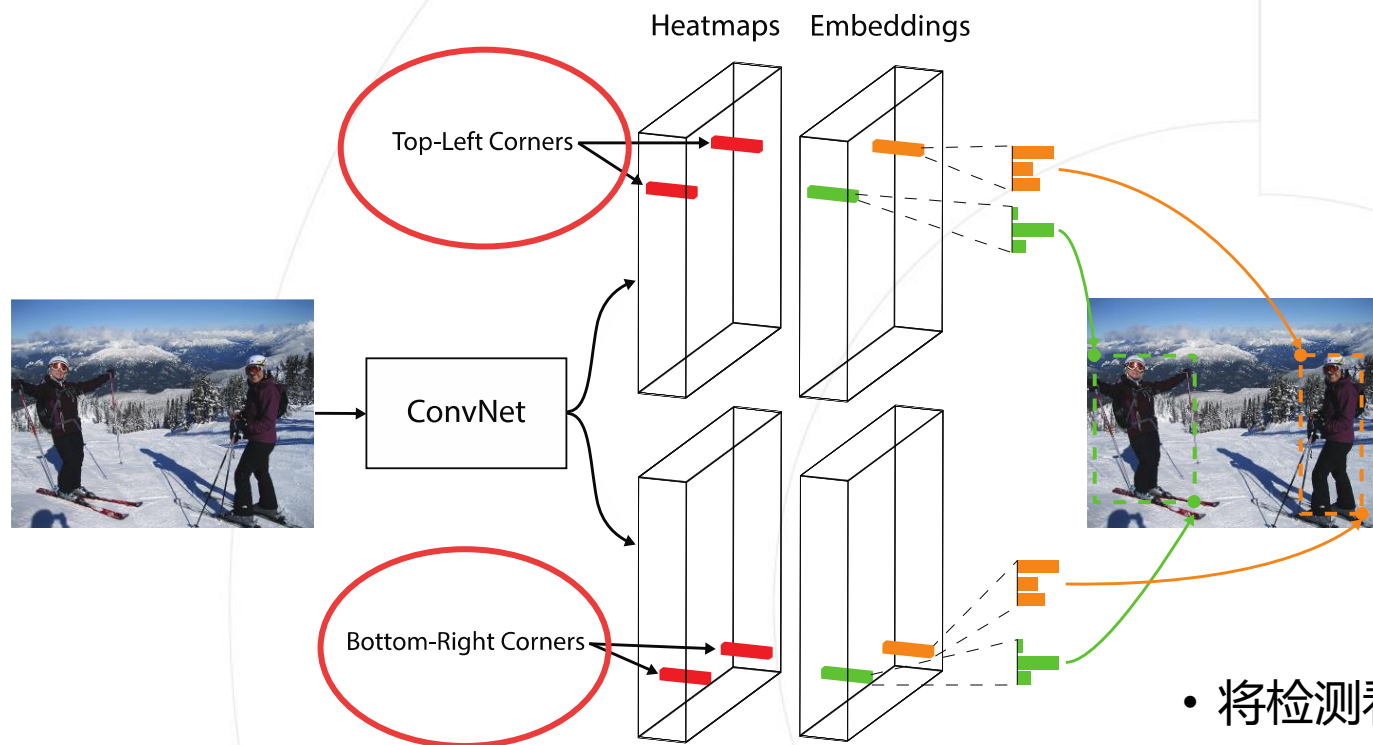
Part 2 **Anchor-based Solutions**

Part 3 **Anchor-free Solutions**

Part 4 **Future Research Topics**

Part 5 **Detection in Action**

- CornerNet: Detecting Objects as Paired Keypoints



Discussion

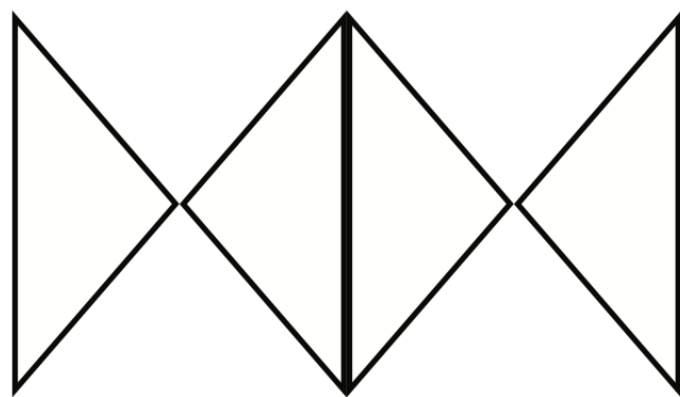
How does this idea differentiate with anchor-based methods?

- 将检测看作预测物体的 **Top-left Corner** 和 **Bottom-Right Corner**;
- 为了寻找 Top-left Corner, 提出 **corner pooling** 来定位点与物体间的关系

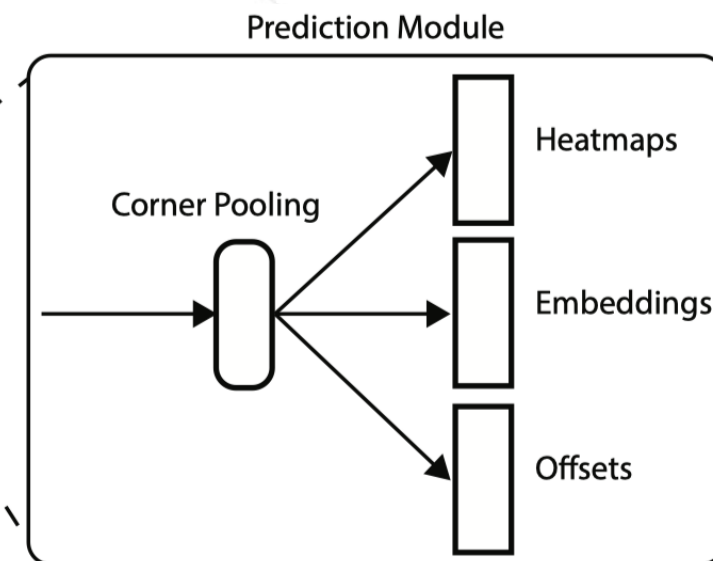
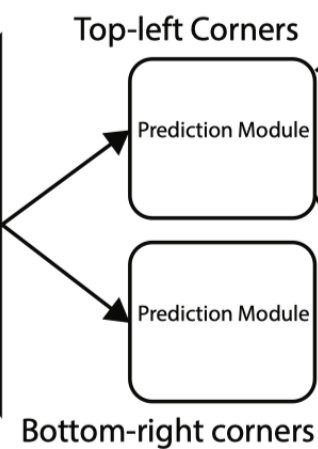
- CornerNet Network structure

- Hourglass Network, Corner Pooling, A variant of focal loss

- 为 top-left corners 和 bottom-right corners 预测两个 heatmaps, channel 数等于类别数 C , 每一维为一个 binary mask



Hourglass Network



- A variant of focal loss
 - Let p_{cij} be the score at location (i, j) for class c in the predicted heatmaps
 - Let y_{cij} be the GT heatmap augmented with the unnormalized Gaussians.

$$L_{det} = \frac{-1}{N} \sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W \begin{cases} (1 - p_{cij})^\alpha \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^\beta (p_{cij})^\alpha \log(1 - p_{cij}) & \text{otherwise} \end{cases}$$

- N is the number of objects in an image
- α and β are the hyper-parameters which control the contribution of each point

- Offset loss

- 由于下采样后热点图比原图分辨率低，因此引入 offset loss 对 corners 进行微调

$$L_{off} = \frac{1}{N} \sum_{k=1}^N \text{SmoothL1Loss}(\mathbf{o}_k, \hat{\mathbf{o}}_k)$$

- Grouping Corners

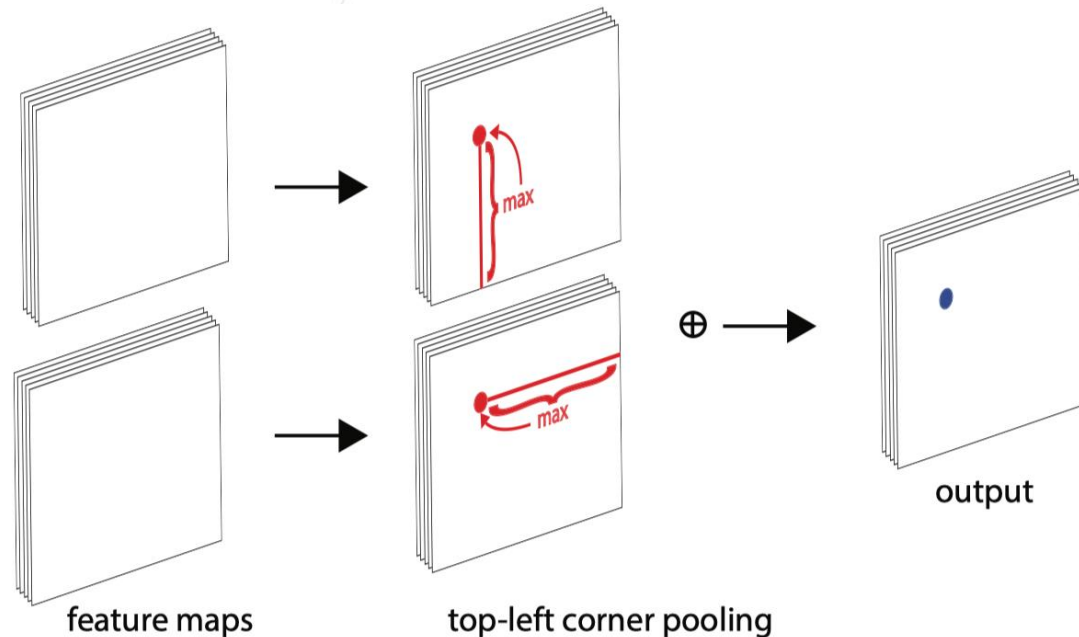
- 同一张图中可能会有多个目标，模型生成多组 top-left corners 和 bottom-right corners
- 使用 pull loss 拉近同一目标的 corners

$$L_{pull} = \frac{1}{N} \sum_{k=1}^N \left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2 \right],$$

- 使用 push loss 推远不同目标的 corners

$$L_{push} = \frac{1}{N(N-1)} \sum_{k=1}^N \sum_{\substack{j=1 \\ j \neq k}}^N \max(0, \Delta - |e_k - e_j|),$$

- Corner Pooling



$$t_{ij} = \begin{cases} \max(f_{t_{ij}}, t_{(i+1)j}) & \text{if } i < H \\ f_{t_{Hj}} & \text{otherwise} \end{cases}$$

$$l_{ij} = \begin{cases} \max(f_{l_{ij}}, l_{i(j+1)}) & \text{if } j < W \\ f_{l_{iW}} & \text{otherwise} \end{cases}$$

Fig. 3 Corner pooling: for each channel, we take the maximum values (*red dots*) in two directions (*red lines*), each from a separate feature map, and add the two maximums together (*blue dot*).

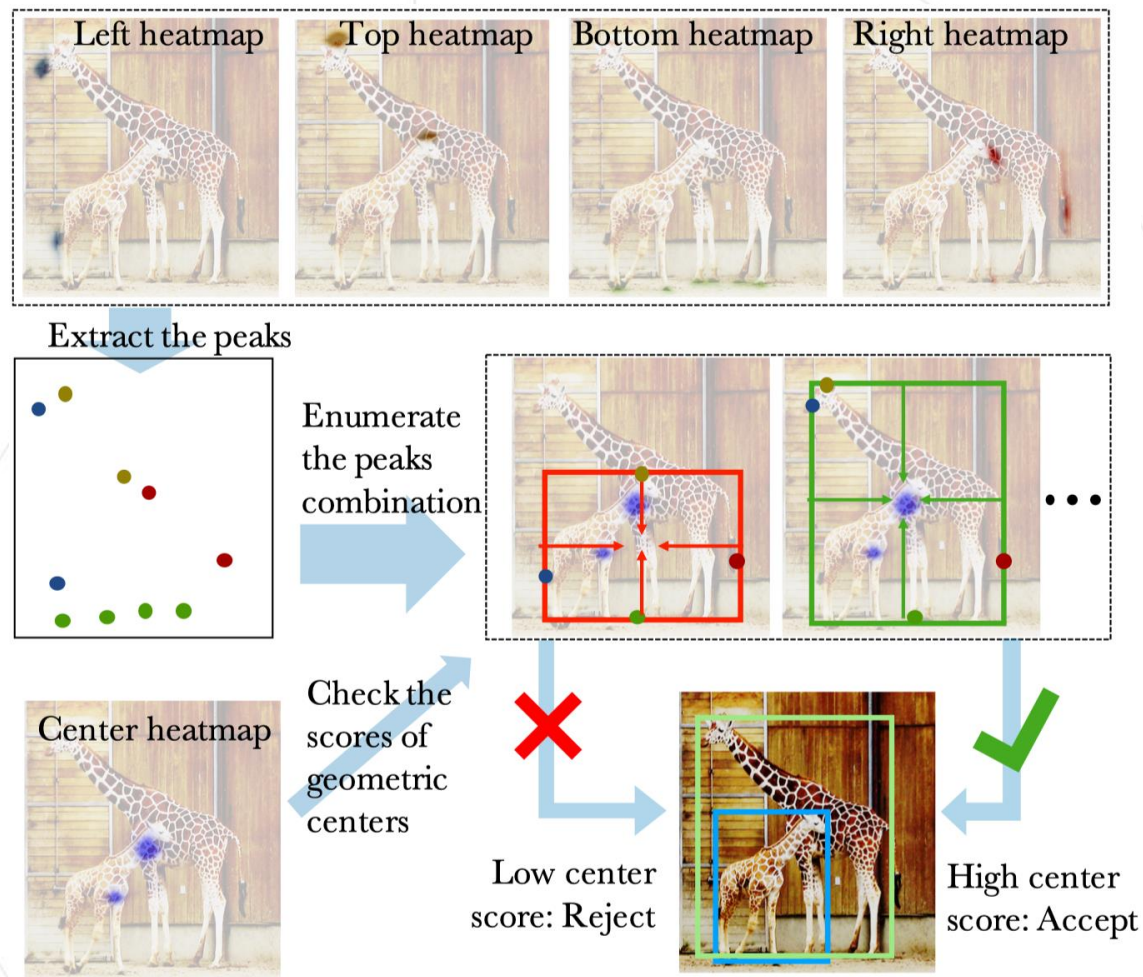
- Anchor-free 中的 Bottom-up 思路

area regression -> **keypoints location**.

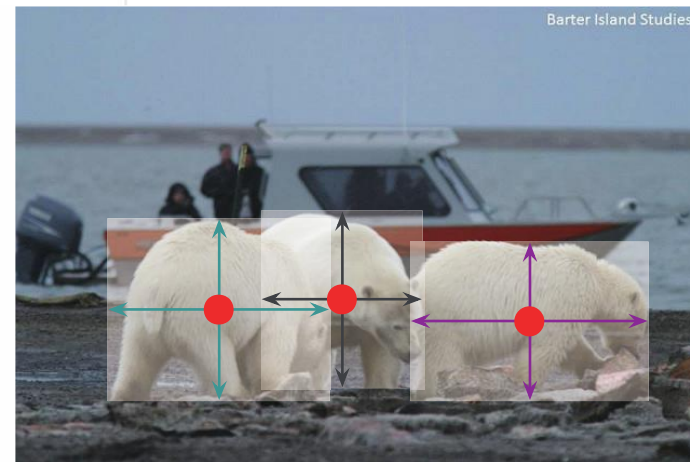
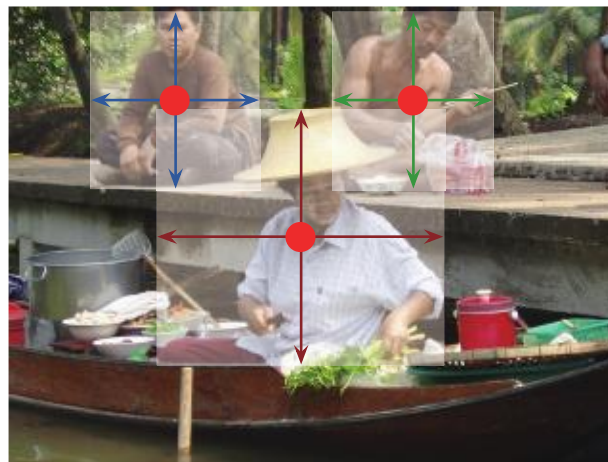
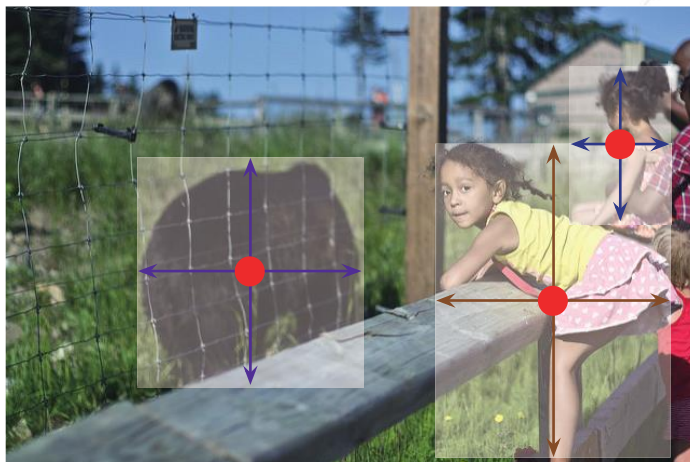
即优化的目标 GT 不再是矩形框

相似的工作还有 ExtremeNet (右图)。

ExtremeNet 的关键点为**四边的极值点**和**框中心点**



- CenterNet: Objects as Points



- **Objects as Points.** 物体的中心点即代表这个物体，物体的 size 需要回归
- 基于 Hourglass backbone
- 三个分支，分别输出：HeatMap，中心点位置；Offset，refine Heatmap；Height&Width，检测框的宽高

Zhou X, Wang D, Krähenbühl P. Objects as points. arXiv preprint arXiv:1904.07850, 2019.

- CenterNet: Objects as Points

Produce GT keypoint heatmap

- Keypoint types 标注为物体种类
- 使用 Gaussian kernel 从 point 生成 heatmap

$$Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$$

中心点预测 loss function (focal loss)

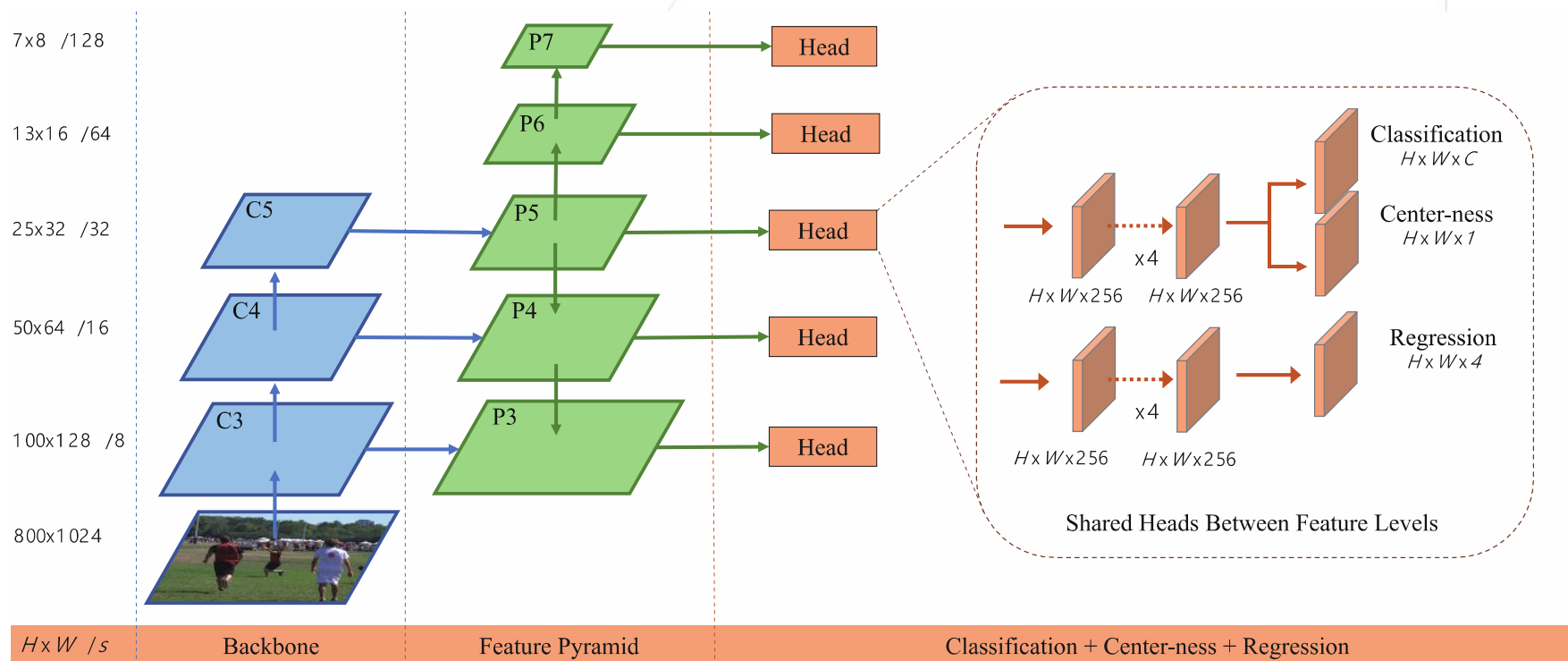
$$L_k = \frac{-1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^\alpha \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1 - Y_{xyc})^\beta (\hat{Y}_{xyc})^\alpha & \text{otherwise} \\ \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases}$$

Offset loss function

$$L_{off} = \frac{1}{N} \sum_p \left| \hat{O}_{\tilde{p}} - \left(\frac{p}{R} - \tilde{p}\right) \right|$$

- CenterNet: Objects as Points
 - One problem: center point collision
 - CenterNet 中网络下采样 stride 为 4, 在 COCO 中存在 614 对物体出现 center point collision
 - CenterNet is unable to predict $< 0.1\%$ of objects. This is much fewer than anchor-based methods miss due to insufficient anchor placement (20.0% for Faster-RCNN with 15 anchors at 0.5 IOU threshold).

• FCOS: Fully Convolutional One-Stage Object Detection



- FCOS 将 feature maps 每个位置都作为训练样本;
- 使用 Feature Pyramid, 多级框预测解决 GT box 重叠问题
- 训练一个分支预测 Center-ness, 测试时用于抑制远离中心的框

- GT box 重叠问题
 - 两个物体中心相同，例如运动员手持球拍，那么如何解决这个问题呢（理论分析、常识、统计数据）
 - 大多数情况下，这种重叠区域的 box 尺寸差异明显，因此可以使用基于 FPN 的结构将其分到不同层

- FCOS: Fully Convolutional One-Stage Object Detection

回归目标: (l, t, r, b)

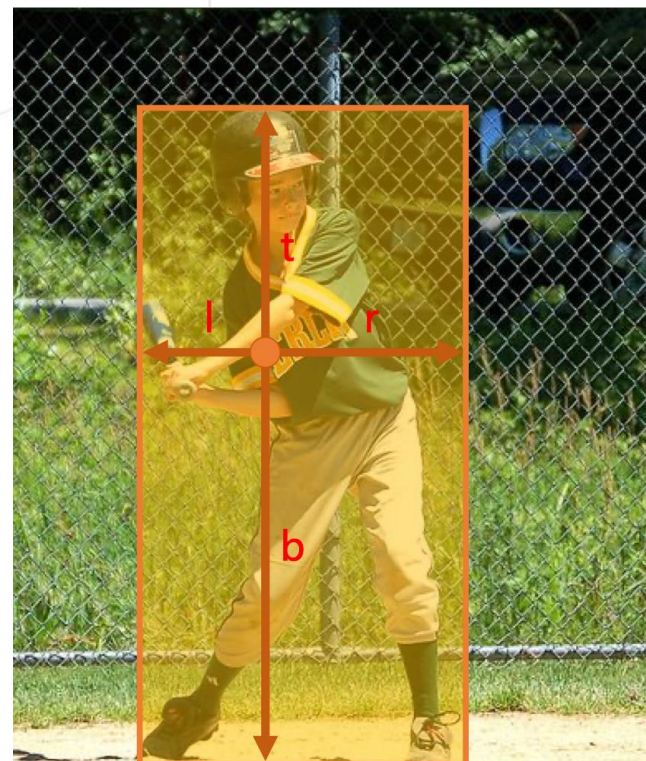
$$l^* = x - x_0^{(i)}, \quad t^* = y - y_0^{(i)},$$

$$r^* = x_1^{(i)} - x, \quad b^* = y_1^{(i)} - y.$$

对 feature map 中, (映射回原图后) 属于 GT box 的点做回归

$$L(\{\mathbf{p}_{x,y}\}, \{\mathbf{t}_{x,y}\}) = \frac{1}{N_{\text{pos}}} \sum_{x,y} L_{\text{cls}}(\mathbf{p}_{x,y}, c_{x,y}^*) \\ + \frac{\lambda}{N_{\text{pos}}} \sum_{x,y} \mathbb{1}_{\{c_{x,y}^* > 0\}} L_{\text{reg}}(\mathbf{t}_{x,y}, \mathbf{t}_{x,y}^*),$$

L_{reg} 为 IoU Loss



- Anchor-free 中的 Point 先验思路
 - anchor 先验 -> **point 先验**。不再提取 anchor，而是将 feature map 上的点作为中心点优化。PS：依然使用矩形区域
 - 该类型的工作一般使用 FPN neck + Focal loss
 - 以上介绍的以及 FoveaBox、FSAF 等工作都可以在 MMDetection 看到复现代码

```
master mmdetection / configs / foveabox / fovea_r50_fpn_4x4_1x_coco.py / <> Jump to v
yhcao6 clean op (#3232) ... ✓
4 contributors
52 lines (52 sloc) | 1.51 KB
1 _base_ = [
2     '../_base_/datasets/coco_detection.py',
3     '../_base_/schedules/schedule_1x.py', '../_base_/default_runtime.py'
4 ]
5 # model settings
6 model = dict(
7     type='FOVEA',
8     pretrained='torchvision://resnet50',
9     backbone=dict(
10        type='ResNet',
11        depth=50,
12        num_stages=4,
13        out_indices=(0, 1, 2, 3),
14        frozen_stages=1,
15        norm_cfg=dict(type='BN', requires_grad=True),
16        norm_eval=True,
17        style='pytorch'),
18    neck=dict(
19        type='FPN',
20        in_channels=[256, 512, 1024, 2048],
21        out_channels=256,
22        start_level=1,
23        num_outs=5,
24        add_extra_convs='on_input'),
25    bbox_head=dict(
26        type='FoveaHead',
27        num_classes=80,
28        in_channels=256,
29        stacked_convs=4,
30        feat_channels=256,
31        strides=[8, 16, 32, 64, 128],
32        base_edge_list=[16, 32, 64, 128, 256],
33        scale_ranges=((1, 64), (32, 128), (64, 256), (128, 512), (256, 2048)),
34        sigma=0.4,
35        with_deform=False,
36        loss_cls=dict(
37            type='FocalLoss',
38            use_sigmoid=True,
39            gamma=1.5,
40            alpha=0.4,
41            loss_weight=1.0),
42        loss_bbox=dict(type='SmoothL1Loss', beta=0.11, loss_weight=1.0))
43 # training and testing settings
```

Chen K, Wang J, Pang J, et al. Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155, 2019.

- Experimental results and discussion

Table 6: Comparison of RepPoints v2 to state-of-the-art detectors on COCO test-dev. * denote that the number is obtained by multi-scale testing.

method	backbone	epoch	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet [16]	ResNet-101	18	39.1	59.1	42.3	21.8	42.7	50.2
FCOS [27]	ResNeXt-101	24	43.2	62.8	46.6	26.5	46.2	53.3
DCN V2* [36]	ResNet-101+DCN	18	46.0	67.9	50.8	27.8	49.1	59.5
RepPoints* [30]	ResNet-101+DCN	24	46.5	67.4	50.9	30.3	49.7	57.1
MAL* [11]	ResNeXt-101	24	47.0	66.1	51.2	30.2	50.1	58.9
FreeAnchor* [33]	ResNeXt-101	24	47.3	66.3	51.5	30.6	50.4	59.0
ATSS* [32]	ResNeXt-101+DCN	24	50.7	68.9	56.3	33.2	52.9	62.4
TSD* [25]	SENet154+DCN	24	51.2	71.9	56.0	33.8	54.8	64.2
CornerNet [13]	HG-104	100	40.5	56.5	43.1	19.4	42.7	53.9
ExtremeNet [35]	HG-104	100	40.2	55.5	43.2	20.4	43.2	53.1
CenterNet [4]	HG-104	100	44.9	62.4	48.1	25.6	47.4	57.4
RepPoints v2	ResNet-50	24	44.4	63.5	47.7	26.6	47	54.6
RepPoints v2	ResNet-101	24	46.0	65.3	49.5	27.4	48.9	57.3
RepPoints v2	ResNeXt-101	24	47.8	67.3	51.7	29.3	50.7	59.5
RepPoints v2	ResNet-101+DCN	24	48.1	67.5	51.8	28.7	50.9	60.8
RepPoints v2	ResNeXt-101+DCN	24	49.4	68.9	53.4	30.3	52.1	62.3
RepPoints v2*	ResNeXt-101+DCN	24	52.1	70.1	57.5	34.5	54.6	63.6

Discussion
Anchor-based or anchor-free, what's your opinion?

- Anchor-free Paper list

<https://github.com/XinZhangNLPR/awesome-anchor-free-object-detection>

awesome-anchor-free-object-detection

for anyone who wants to do research about anchor free object detection.

If you find the awesome paper/code/dataset or have some suggestions, please contact xin.zhang2018@nlpr.ia.ac.cn, nuo.xu@nlpr.ia.ac.cn and xswang@wayne.edu. Thanks for your valuable contribution to the research community 😊

- Recent papers (from 2015)

Statistics: 🔥 code is available & stars >= 100

2020

- [\[arXiv\]](#) OneNet: End-to-End One-Stage Object Detection by Classification Cost. [\[pytorch\]](#) 🔥
- [\[arXiv\]](#) End-to-End Object Detection with Fully Convolutional Network
- [\[arXiv\]](#) Sparse R-CNN: End-to-End Object Detection with Learnable Proposals. [\[pytorch\]](#) 🔥
- [\[arXiv\]](#) End-to-End Object Detection with Transformers. [\[pytorch\]](#) 🔥
- [\[arXiv\]](#) AutoAssign: Differentiable Label Assignment for Dense Object Detection.



Outline

课后阅读内容

Click to jump to Part 5

Part 1 **Introduction to Object Detection**

Part 2 **Anchor-based Solutions**

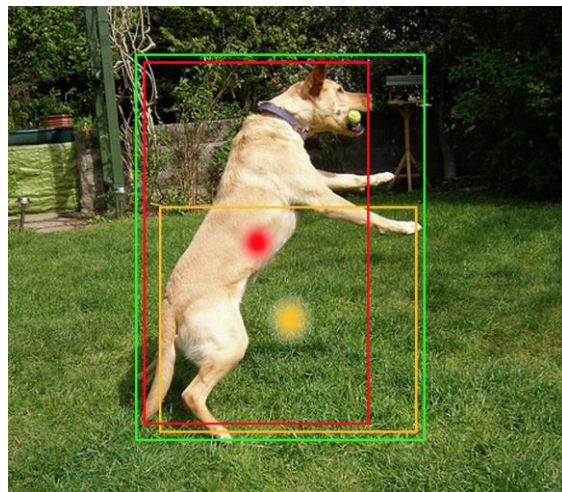
Part 3 **Anchor-free Solutions**

Part 4 **Future Research Topics**

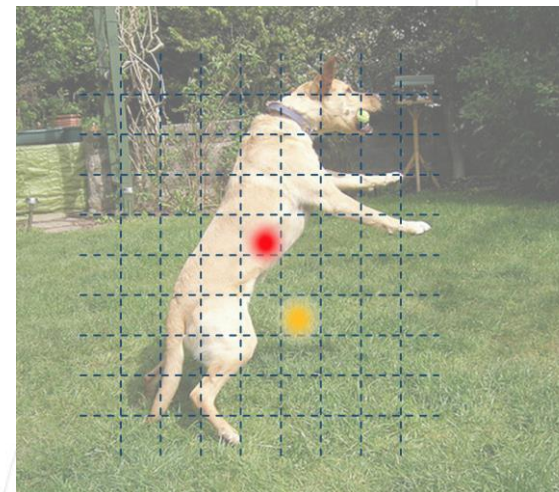
Part 5 **Detection in Action**

- Discover the problem

课后阅读内容



(a) Original image



(b) Feature grid

(a) **green**: ground truth, **orange**: original anchor, **red**: refined anchor

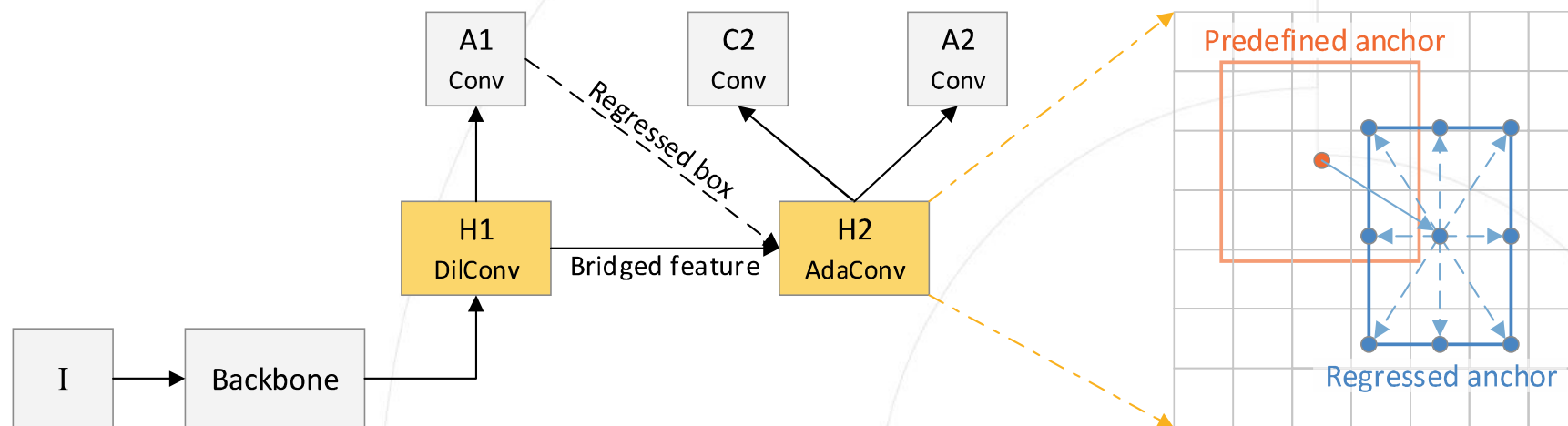
(b) Location of center points for original and refined anchors in the feature grid

Do you think simply extracting features from the previous location is reasonable?

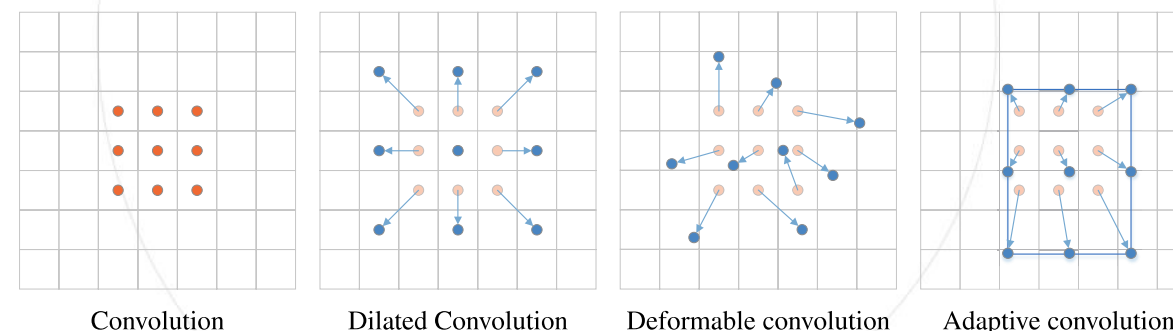
This problem appears in heuristic sampling (RPN) and any one-stage detector

- A solution: Cascade RPN

课后阅读内容



Cascade RPN uses **adaptive convolution** to perform sampling within the anchors to ensure alignment between the **anchors** and **features**



Convolution

Dilated Convolution

Deformable convolution

Adaptive convolution

Vu T, Jang H, Pham T X, et al. Cascade rpn: Delving into high-quality region proposal network with adaptive convolution. NIPS2019

- Recommended reading

课后阅读内容

Cascade RPN: Delving into High-Quality Region Proposal Network with Adaptive Convolution

- 引入 adaptive convolution 来对齐 anchor 和 feature

Revisiting Feature Alignment for One-stage Object Detection

- 提出 RoI Convolution, 根据偏移在 feature 上做 RoI Convolution 得到对齐的特征, 然后再做 regression

Cascade RetinaNet: Maintaining Consistency for Single-Stage Object Detection

- 提出 FCM 来将当前阶段的特征修正到新的位置 (而不是上一 stage 的位置)

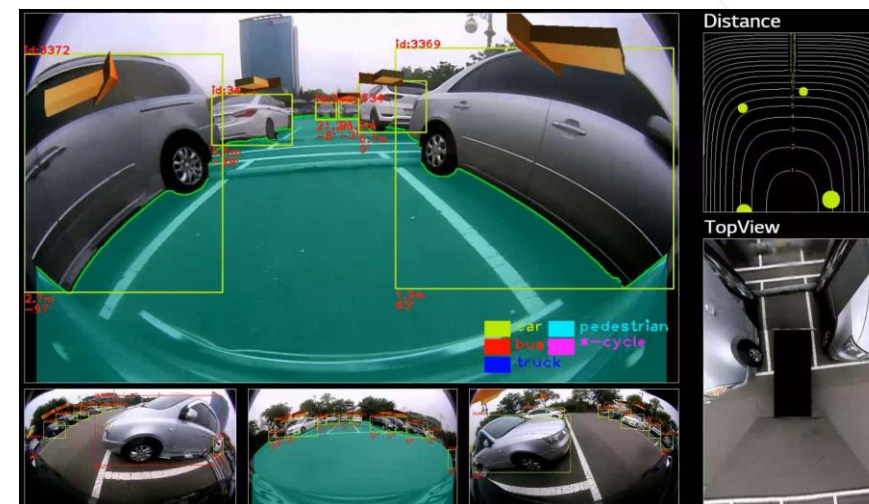
- Problem Formulation

- What do the omnidirectional images look like?
- Why we need to make research on this problem?
- Introduction to some existing methods.

课后阅读内容



(a) Example of omnidirectional image in VR application



(b) Example of omnidirectional image in autonomous driving application

- Why we need to make research on this problem?

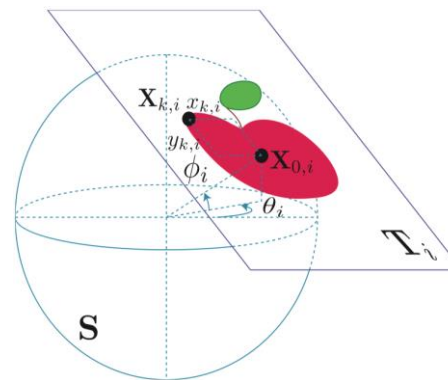
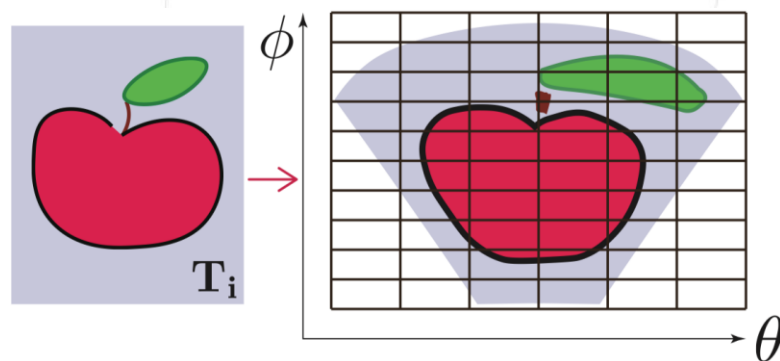
课后阅读内容

针对普通平面图像的方法不适用于 Omnidirectional Cameras

下图为 tangent plane 中的一张苹果的图片，和对应点在 sphere 上的点。

a) 普通的卷积能否融合 lens geometry knowledge;

b) 当图片与球面的切点不同时，我们还能得到相同的卷积结果吗?



Example of the equirectangular representation of the image (left), and the gnomonic projection (right)

- Scene Understanding Networks for Autonomous Driving based on Around View Monitoring System

课后阅读内容

Task: Around View Monitoring (AVM) system, free drivable area

Method: 网络接受 **columns** 形式的输入, 每一 column 经过 conv layers 输出该列障碍物对应 bottom pixel 的位置。例如原图高为 720, 则是一个 720 类分类任务。结合每一列障碍物 bottom pixel, 即可得到 free drivable area
该网络每个 column 之间有大量 overlap, 作者进而设计了一个端到端的网络结构, 此处不再细说



A columnwise prediction for a corresponding pixel augmented with the adjacent 24 pixels area

- Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360° Panoramic Imagery

课后阅读内容

Method: 全景图像使用 equirectangular projection 表示。

调整 rectilinear imagery 上的网络来适应 equirectangular

panoramic imagery。通过 GAN 网络生成全景图像

基于 faster R-CNN 算法，输出类和象限 (quadrant) 分

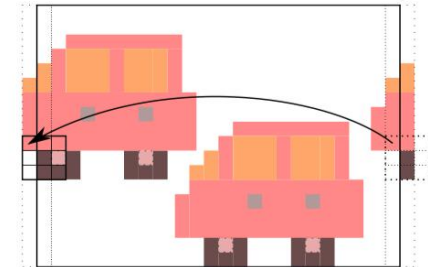
类，边框的位置、目标中心、距离和朝向

当一个物体在展开图边界时，作者提出在边缘进行填充

计算 conv



(a) A 360° equirectangular image can be folded over itself until the ends meet.



(b) A 3×3 convolution kernel, a column of padding copied from the other side is added at each extremity



Outline

Part 1 **Introduction to Object Detection**

Part 2 **Anchor-based Solutions**

Part 3 **Anchor-free Solutions**

Part 4 **Future Research Topics**

Part 5 **Detection in Action**



清华大学
Tsinghua University



Detection in Action

2D物体检测实战

对于初学者:

mmdetection/[code link](#)

detectron2/[code link](#)

如果你已经很了解物体检测:

通过 [paperswithcode](#) 掌握 Object Detection 的新进展与性能榜单

在 [Kaggle](#) 尝试参与竞赛



Search for papers, code and tasks

Browse State-of-the-Art Methods Reproducibility Portals About

Log In/Register



Object Detection

Edit Task

Computer Vision • 3D Object Detection

1056 papers with code 27 benchmarks

About

Edit

Object detection is the task of detecting instances of objects of a certain class within an image. The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN and Cascade R-CNN.

The most popular benchmark is the MSCOCO dataset. Models are typically evaluated according to a Mean Average Precision metric.

(Image credit: Detectron)

Benchmarks

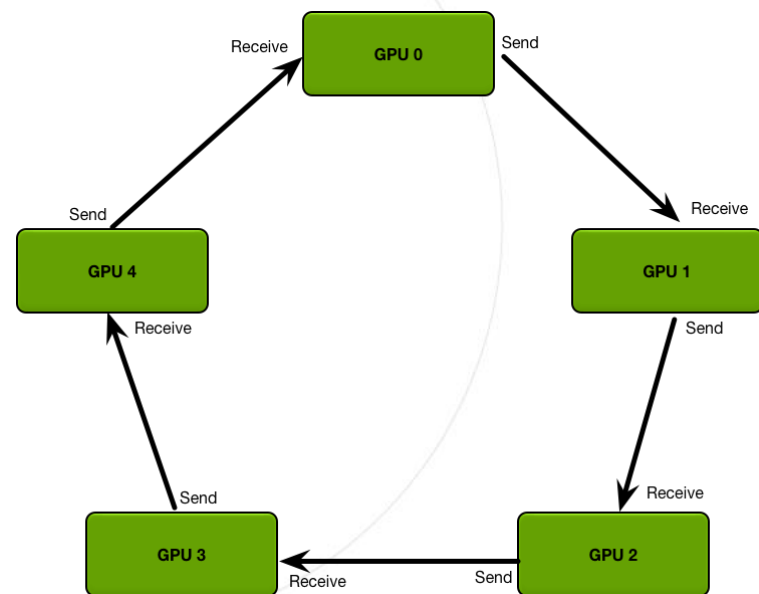
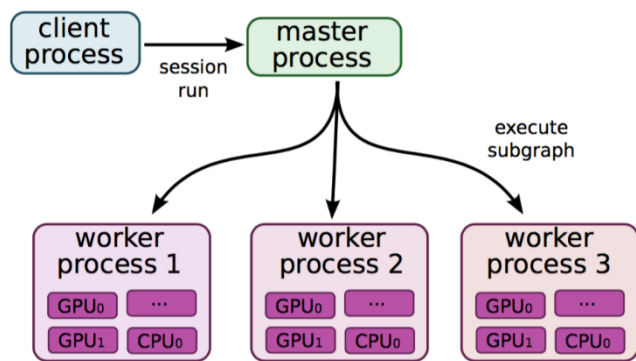
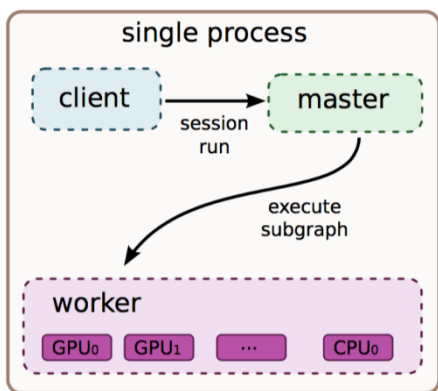
Add a Result

TREND	DATASET	BEST METHOD	PAPER TITLE	PAPER	CODE	COMPARE
	COCO test-dev	🏆 Cascade Eff-B7 NAS-FPN (1280, self-training Copy Paste, single-scale)	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation			See all
	COCO minival	🏆 Cascade Eff-B7 NAS-FPN (1280, self-training Copy Paste, single-scale)	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation			See all
	PASCAL VOC 2007	🏆 Cascade Eff-B7 NAS-FPN (Copy Paste pre-training, single-scale)	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation			See all
	KITTI Cars Easy	🏆 Patches	Patch Refinement -- Localized 3D Object Detection			See all
	CrowdHuman (full body)	🏆 IterDet (Faster RCNN, ResNet50, 2 iterations)	IterDet: Iterative Scheme for ObjectDetection in Crowded Environments			See all
	KITTI Cars Moderate	🏆 Patches	Patch Refinement -- Localized 3D Object Detection			See all

- Pytorch 中的 `nn.DataParallel` 与 `torch.distributed`
 - `torch.distributed` 基于 MPI 和 NCCL, 解决 `DataParallel` 速度慢, GPU 负载不均衡的问题
- Apex 混合精度训练时的多卡
 - `apex.parallel.DistributedDataParallel`
 - `amp.scale_loss`
- BN 层统计量问题
 - `nn.DataParallel` 不支持 BN 同步, `torch.distributed` 支持
- Learning rate 的调整

Parameter Server 与 Ring AllReduce 算法

- Parameter Server: GPU 0 将数据分成 N 份分到各个卡上, 每张卡负责自己的那一份 mini-batch 的训练, 得到 gradient 后, 返回给 GPU 0 上做累积, 得到更新的权重参数后, 再分发给各个卡。
- Ring AllReduce: N 张 GPU 以环形相连, 每张卡都有左手卡和右手卡, 一个负责接收, 一个负责发送, 循环 4 次完成梯度累积, 再循环 $N-1$ 次做参数同步。分为 Scatter Reduce 和 All Gather 两个环节。



- Pytorch 中的 `nn.DataParallel` 与 `torch.distributed`
 - DataParallel (DP)
 - Parameter Server 模式
 - 模型较大时, GPU 0 负载较严重
 - 不支持同步 BN。每次计算 BN 时, DP 会把 GPU 0 上的数据计算的 mean 和 variance 广播给其他卡
 - 使用方法

```
model = nn.DataParallel(model)
```

- Pytorch 中的 `nn.DataParallel` 与 `torch.distributed`
 - 推荐: `DistributedDataParallel (DDP)`
 - All-Reduce 模式, 既可以分布式训练, 也可用于单机多卡
 - 支持同步 BN。前传在各 GPU 上计算各自的小 batch mean 和小 batch variance。各 GPU 对小 batch mean 和小 batch variance 进行 `all_gather` 操作, 每个进程都得到全局量。接下来, 延续正常的 BN 计算

```
# DDP init
dist.init_process_group(backend='nccl')
#使用普通 BN 方式定义模型
model = MyModel()
# 引入SyncBN
model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model).to(device)
# 构造DDP模型
model = DDP(model, device_ids=[local_rank], output_device=local_rank)
```




清华大学
Tsinghua University



END