深蓝学院
shenlanxueyuan.com

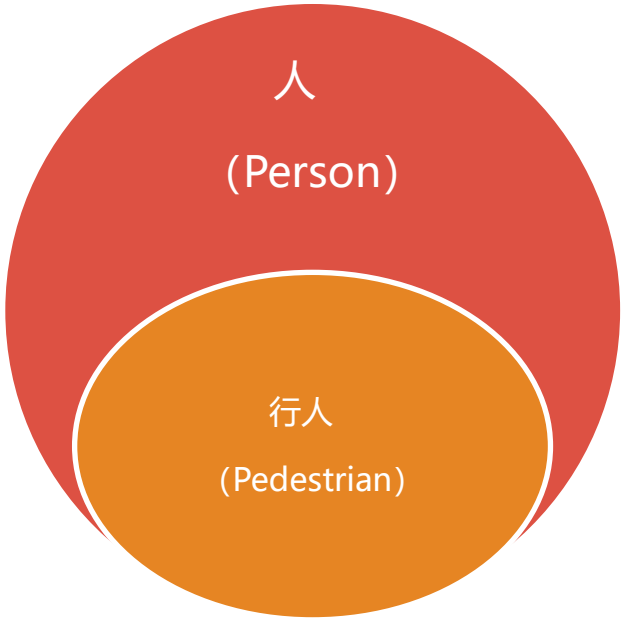
行人检测

主讲人 张士峰

中国科学院自动化研究所
模式识别国家重点实验室

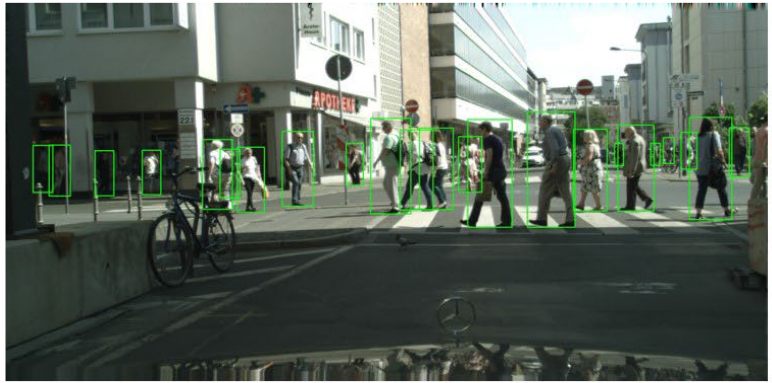
内容回顾：人 (Person) 和行人 (Pedestrian)

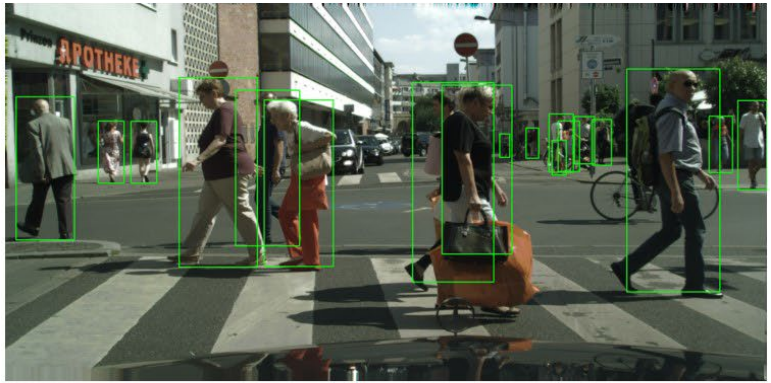
- **人 (Person):** 各种各样的人
- **行人 (Pedestrian):** 步行（走着或跑着）的人

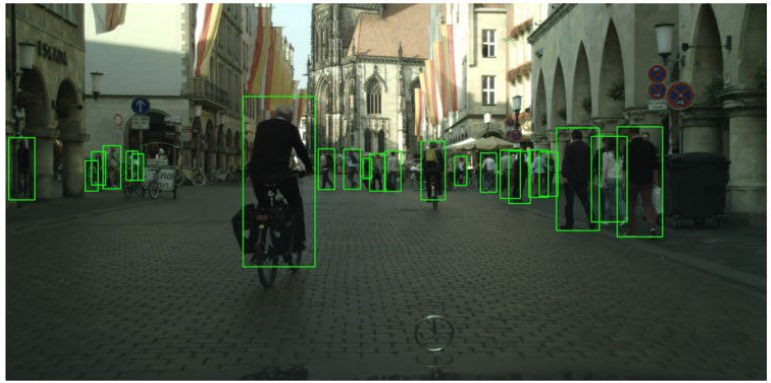


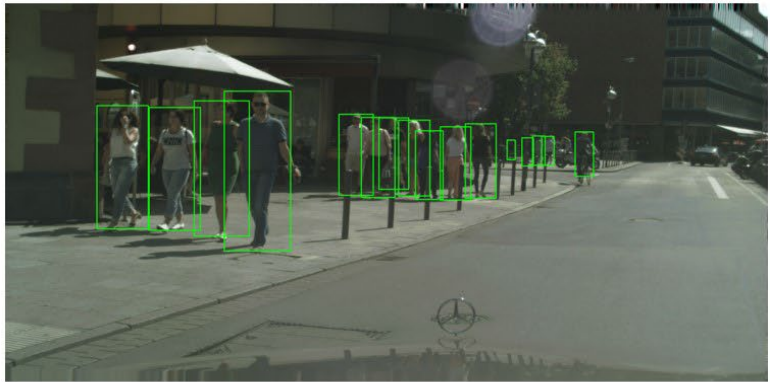
内容回顾：行人检测的定义

- 行人检测：判断一副图像上是否存在行人，如果存在，就给出所有行人的位置







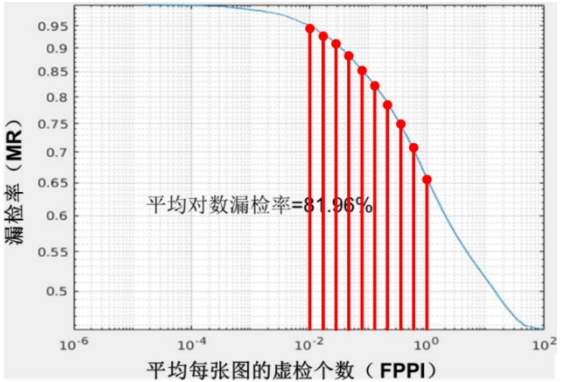


内容回顾：行人检测数据集

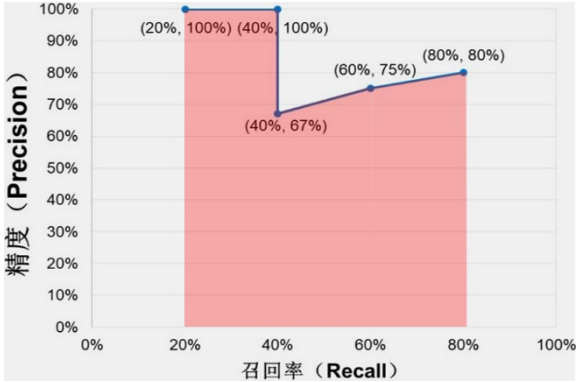
数据库	图片数量				标注类型				难度
	总共	训练集	测试集	验证集	可见行人	完整行人	人头	固定比例	
Caltech-USA	46806	42782	4024	0	×	√	×	√	★★
CityPersons	5000	2975	1525	500	√	√	×	√	★★★
CrowdHuman	24370	15000	5000	4370	√	√	√	x	★★★★★
WiderPerson	13382	8000	4382	1000	×	√	×	√	★★★★
EuroCityPersons	47325	28114	14175	5036	√	√	×	x	★★★★

内容回顾：行人检测评价指标

检测精度

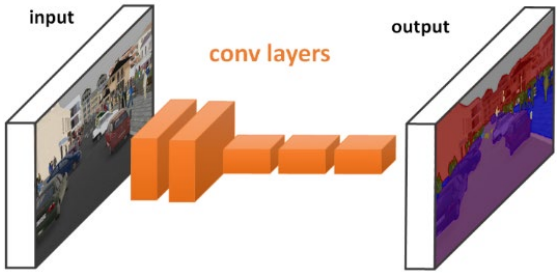


平均对数漏检率（MR^{-2}）

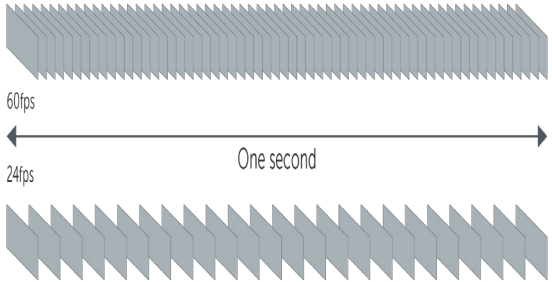


平均精度（AP）

检测速度



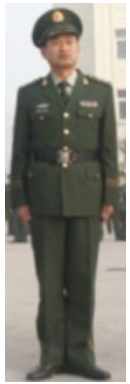
前传耗时 (ms)



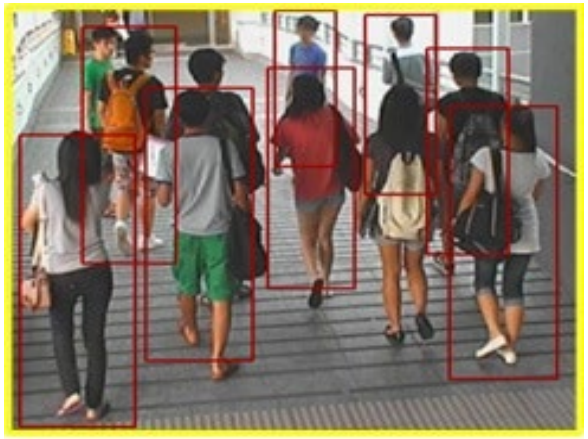
每秒帧数 (FPS)

内容回顾：行人检测的难点

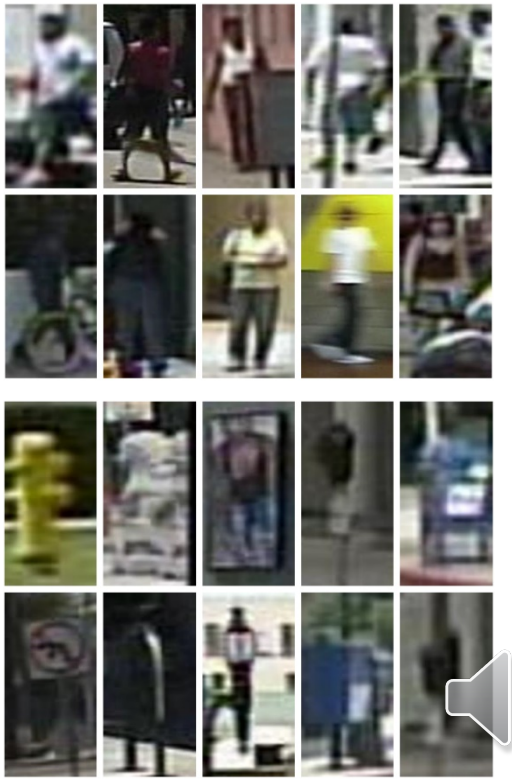
- 行人是可形变的



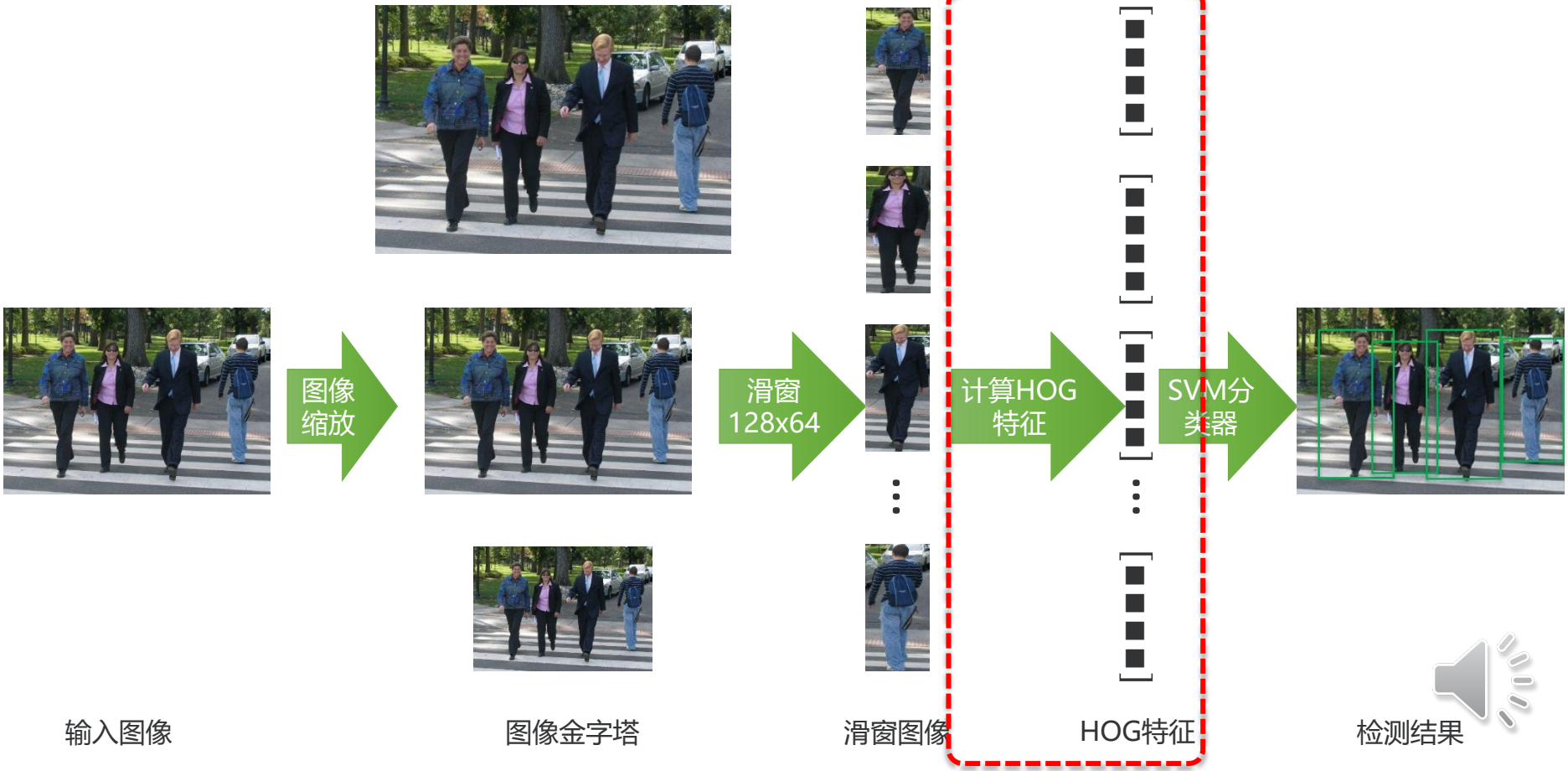

- 行人存在着各种各样的遮挡

- 行人检测中存在很多难样本



内容回顾：传统行人检测Dalal算法

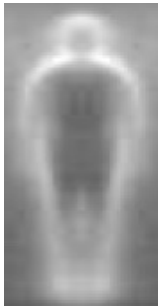
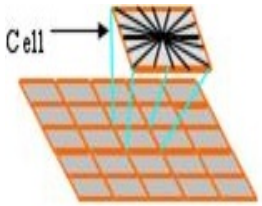


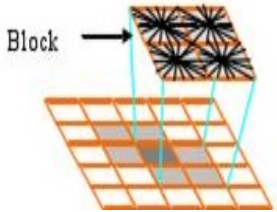
内容回顾：传统行人检测Dalal算法

- HOG特征计算流程

滑窗图象 → 灰度化 → 伽马矫正 → 计算梯度 → 统计梯度 → 归一化 → 特征向量



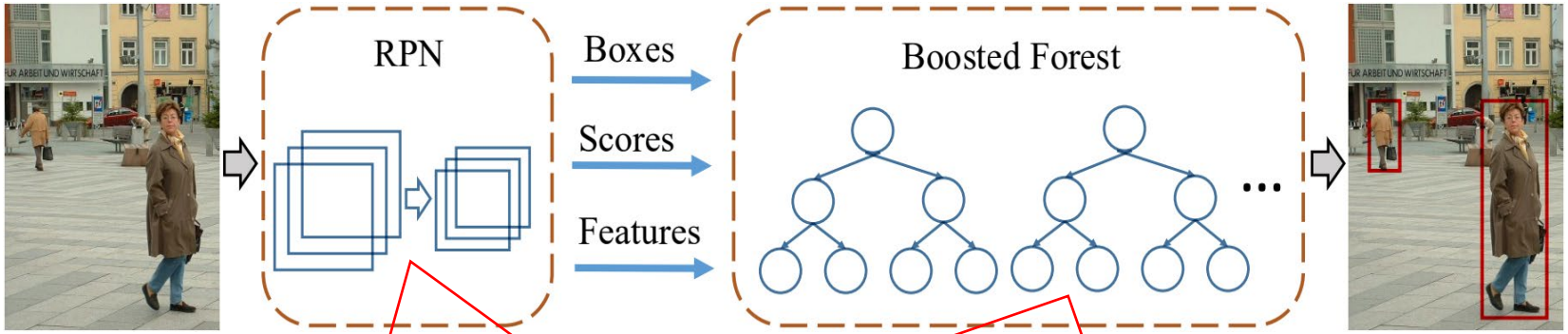


3780维

特征向量

$f = [1, \ldots, N]$

内容回顾：深度学习早期行人检测算法RPN+BF



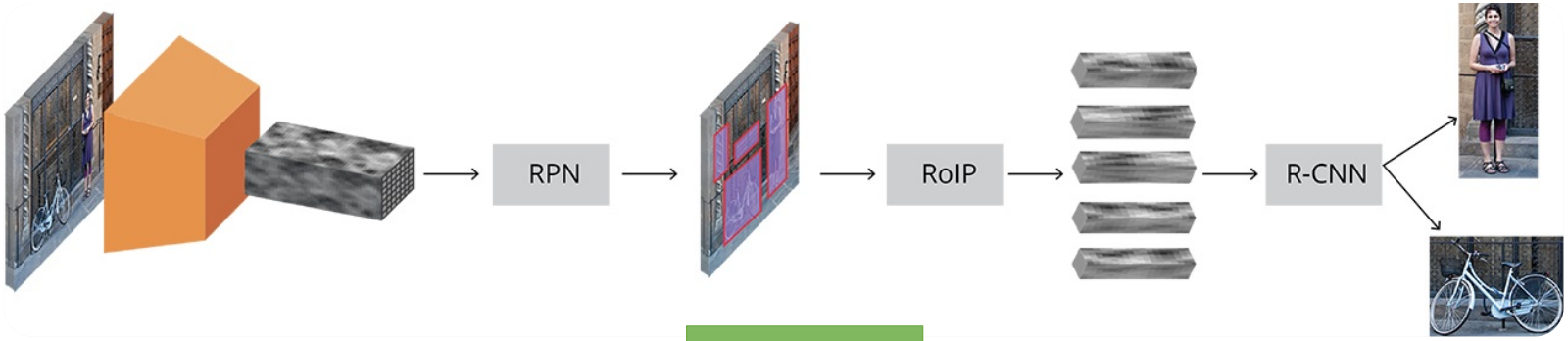
- **问题**：小尺度的人可利用的特征太少，不利于后续的分类和回归
- **方案**：从更浅的、分辨率更高的特征层上来进行特征扣取，从不同分辨率的特征层上扣取特征并进行融合，去掉下采样 + 空洞卷积，来增加特征层的分辨率

- **问题**：行人检测中，分类错误主要是难负样本的混淆，即把背景分为行人，而Faster R-CNN中第二阶段的Fast R-CNN不能很好的处理这些难负样本
- **方案**：采用级联的Boosted Forest (BF) 来替换Fast R-CNN，级联BF输入RoIPooling后的特征，并挖掘困难负样本，CNN特征比手工特征更加高效，更具表现力

目录

深度学习后期行人检测算法

- 深度学习后期行人检测算法：对**通用物体检测算法Faster R-CNN**进行**相应改进**，应用于**行人检测领域**

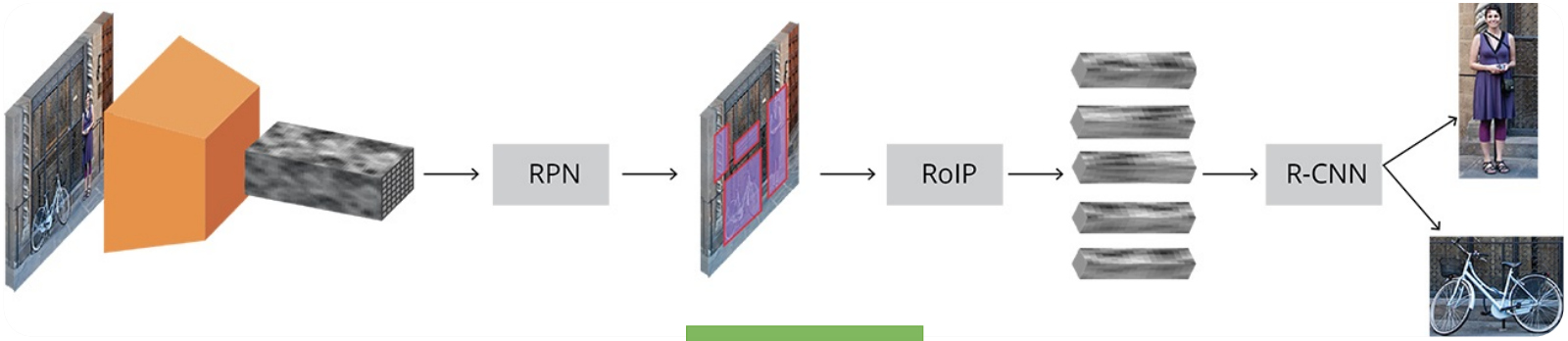


相应改进

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

深度学习后期行人检测算法

- 深度学习后期行人检测算法：对**通用物体检测算法Faster R-CNN**进行**相应改进**，应用于**行人检测领域**



相应改进

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

深度学习后期行人检测算法：Adapted FasterRCNN

- 2017年CVPR上提出的CityPersons数据集中，作者所构建的baseline算法
- 对通用物体检测算法Faster R-CNN进行简单的调整改进应用于行人检测领域
- 改进点：①重新设计锚框 ②放大输入图像 ③优化器SGD->Adam ④忽略区域不采负样本 ⑤增加特征的分辨率

CityPersons: A Diverse Dataset for Pedestrian Detection

Shanshan Zhang, Rodrigo Benenson and Bernt Schiele
Max Planck Institute for Informatics
Saarbrücken, Germany
firstname.lastname@mpi-inf.mpg.de

Adapted FasterRCNN行人检测算法

■ **原始Faster R-CNN算法**

- 基于原始Faster R-CNN算法
- 用CalTech训练集进行训练
- 训练图像大小为640x480
- 修改锚框比例为0.41
- 其他的超参数保持不变
- 在CalTech测试集进行评测
- 平均对数漏检率为20.98%

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

Adapted FasterRCNN行人检测算法

- **原始Faster R-CNN算法**
- **重新设计锚框**
  - RPN有3个锚框尺度：128、256、512
  - 这些尺度不能很好的适合行人这一目标
  - 统计CalTech数据集上行人尺度的分布
  - 根据这一统计信息重新设计锚框的尺度
  - 使得锚框尺度更加适应行人这一目标
  - 11个锚框尺度：8 * [2.0, 2.7, 3.64, 4.92, 6.64, 8.97, 12.11, 16.34, 22.06, 29.79, 40.21]
  - 平均对数漏检率提升2.87个点

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

Adapted FasterRCNN行人检测算法

- **原始Faster R-CNN算法**
- **重新设计锚框**
- **放大输入图像**
  - 原始输入图像大小为640x480
  - 放大2倍输入图像变为1280x960
  - 训练时图像被放大2倍
  - 测试时也被放大2倍
  - 输入图像放大后，行人尺度变大2倍，小尺度行人减少
  - 平均对数漏检率提升3.74个点

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

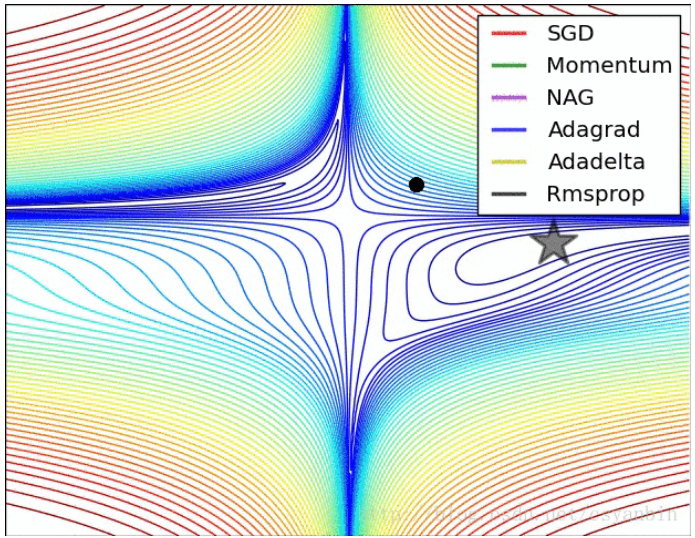
Adapted FasterRCNN行人检测算法

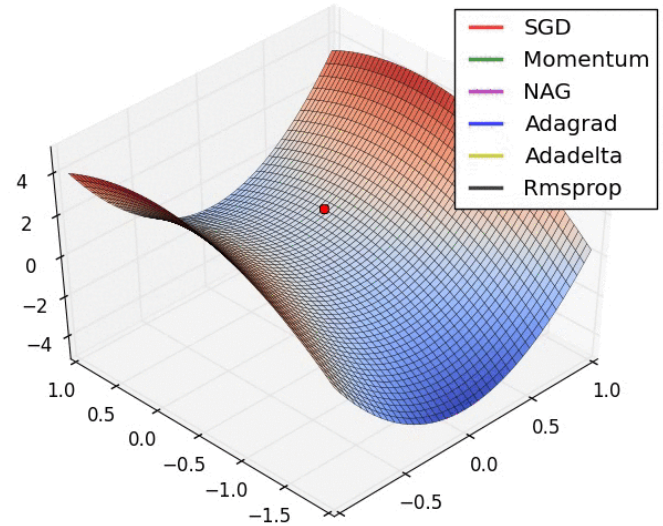
- **原始Faster R-CNN算法**
- **重新设计锚框**
- **放大输入图像**
- **优化器SGD->Adam**
  - 原始Faster R-CNN使用SGD优化器
  - 改用Adam优化器
  - 平均对数漏检率提升1.67个点

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

Adapted FasterRCNN行人检测算法：Adam优化器

- SGD是深度学习中用的最多的优化器，但并不是唯一的优化器，还有很多其他优化器
- Adam优化器对梯度的均值和梯度的未中心化的方差进行综合考虑，计算出更新步长
- 在某些任务中，Adam优化器的效果会好一些，某些任务中则没有提升，需要亲自尝试





Adapted FasterRCNN行人检测算法

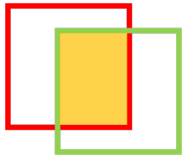
- **原始Faster R-CNN算法**
- **重新设计锚框**
- **放大输入图像**
- **优化器SGD->Adam**
- **忽略区域不采负样本**

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

- 行人检测数据集中有忽略区域的标注
- 忽略区域一般为密集的人群或其他非行人的人
- 它们与行人类似，不能当做负样本训练，故在忽略区域中不采负样本
- 平均对数漏检率提升1.33个点

Adapted FasterRCNN行人检测算法：忽略区域的处理

- 红色框是忽略区域标注
- 绿色框是一个为负样本的锚框
- 如果该负样本，它在忽略区域的面积与自身面积的比值大于0.5，则忽略该负样本



$$\frac{\text{交集的面积}}{\text{负样本的面积}} > 0.5$$

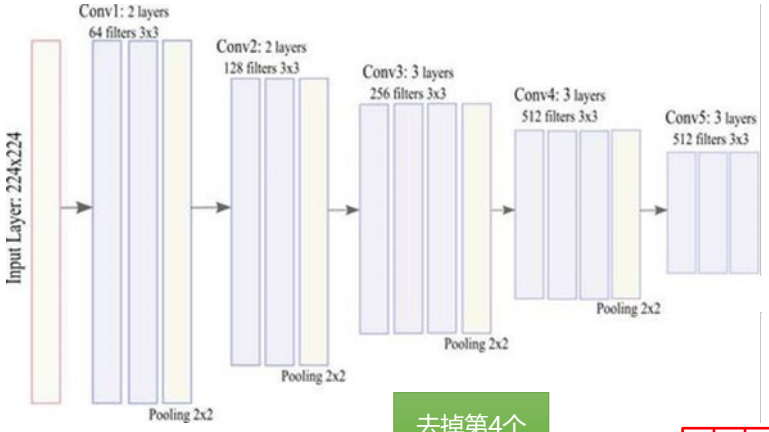
Adapted FasterRCNN行人检测算法

- **原始Faster R-CNN算法**
- **重新设计锚框**
- **放大输入图像**
- **优化器SGD->Adam**
- **忽略区域不采负样本**
- **增加特征的分辨率**

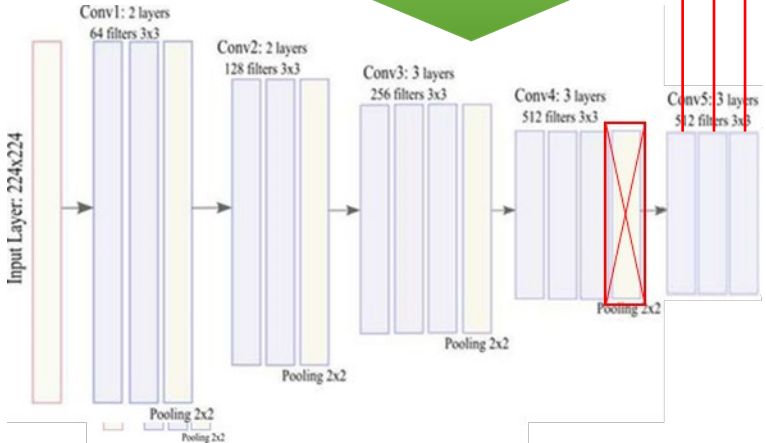
Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

- 去掉第4个Max-Pooling层，把检测层的分辨率增大一倍，使用空洞卷积弥补感受野
- 特征分辨率更高，能够利用的信息变多，对小尺度行人效果提升明显
- 平均对数漏检率提升1.10个点

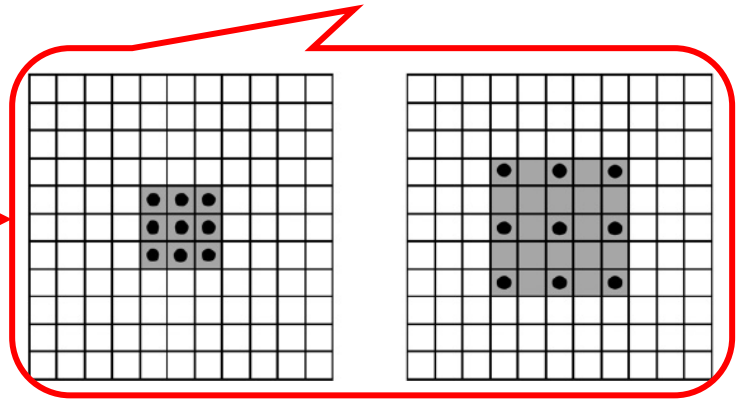
Adapted FasterRCNN行人检测算法：特征强化



去掉第4个
Pooling



空洞卷积：卷积核之间有间隙，以增大感受野



Adapted FasterRCNN行人检测算法

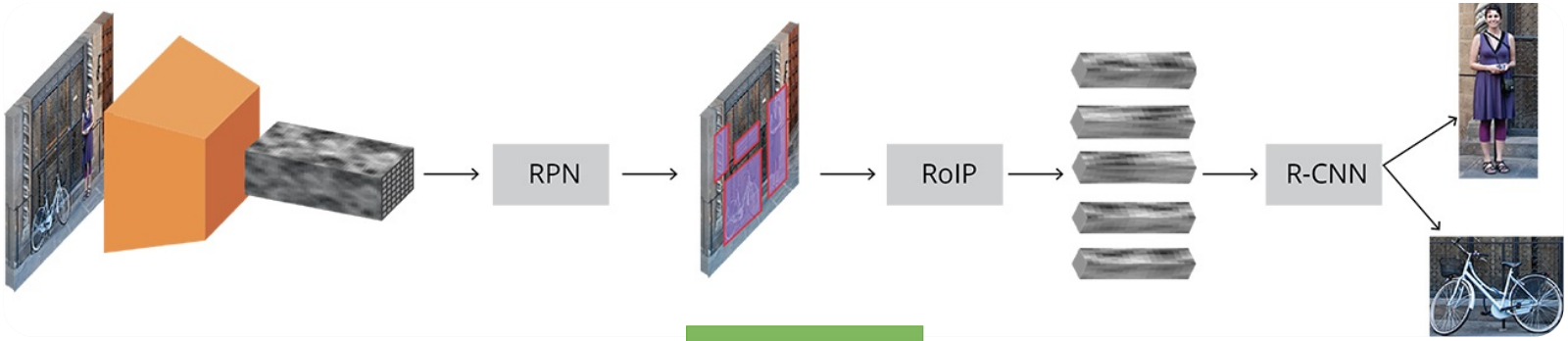
- 原始Faster R-CNN算法
- 重新设计锚框
- 放大输入图像
- 优化器SGD->Adam
- 忽略区域不采负样本
- 增加特征的分辨率

Detector aspect	MR^O	ΔMR
FasterRCNN-vanilla	20.98	-
+ quantized rpn scales	18.11	+ 2.87
+ input up-scaling	14.37	+ 3.74
+ Adam solver	12.70	+ 1.67
+ ignore region handling	11.37	+ 1.33
+ finer feature stride	10.27	+ 1.10
FasterRCNN-ours	10.27	+ 10.71

5个简单的调整改进，使得Faster R-CNN算法提升10.71%

深度学习后期行人检测算法

- 深度学习后期行人检测算法：对**通用物体检测算法Faster R-CNN**进行**相应改进**，应用于**行人检测领域**



相应改进

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

深度学习后期行人检测算法：Repulsion Loss

- 2018年CVPR上提出的行人检测算法，在Adapted FasterRCNN基础上，对回归损失函数进行改进
- 在原有的SmoothL1回归损失函数基础上，额外添加了一个排斥损失项

Repulsion Loss: Detecting Pedestrians in a Crowd

Xinlong Wang[1*] Tete Xiao[2*] Yuning Jiang[3] Shuai Shao[3] Jian Sun[3] Chunhua Shen[4]

[1]Tongji University
1452405wxl@tongji.edu.cn

[2]Peking University
jasonhsiao97@pku.edu.cn

[3]Megvii, Inc.
jyn, shaoshuai, sunjian@megvii.com

[4]The University of Adelaide
chunhua.shen@adelaide.edu.au

Repulsion Loss: Detecting Pedestrians in a Crowd, CVPR 2018

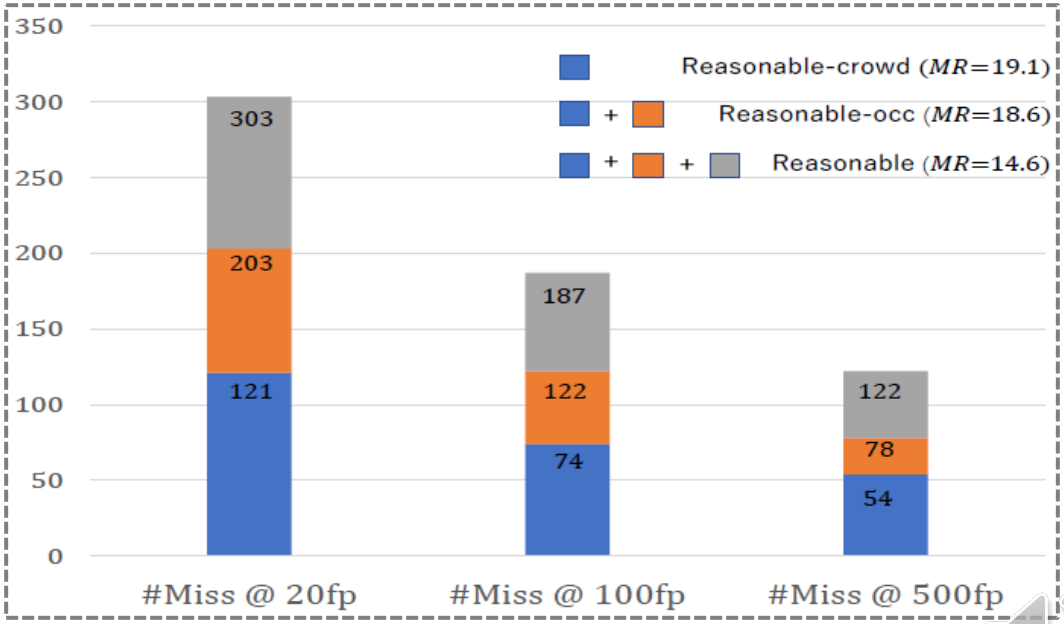
Repulsion Loss行人检测算法

- 行人检测中的遮挡问题

两种遮挡
- 背景物体遮住行人
- 行人之间的遮挡

Repulsion Loss行人检测算法

- 下图为不同虚检个数时，漏检个数的分布情况，其中蓝色为被其他行人遮挡的行人，红色为被背景物体遮挡的行人，灰色为其他



行人之间的遮挡带来的漏检最为明显！！！

Repulsion Loss行人检测算法

Target Groundtruth T

Surrounding Groundtruth B

Predicted Box (for the target)

- 行人检测中行人之间的遮挡所带来的影响
  - 使得算法在定位上不准确

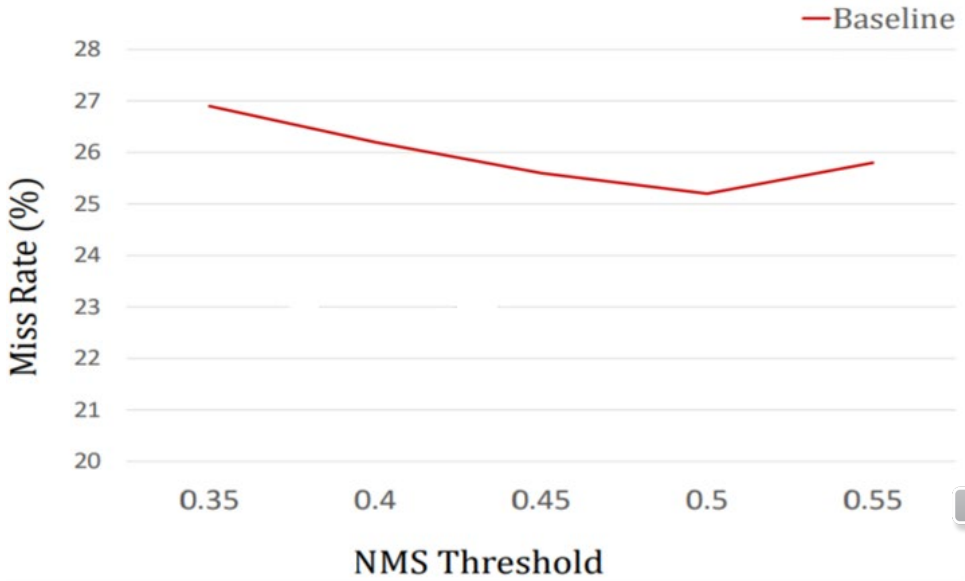
Repulsion Loss行人检测算法

Target Groundtruth T

Surrounding Groundtruth B

Predicted Box (for the target)

- 行人检测中行人之间的遮挡所带来的影响
  - 使得算法在定位上不准确
  - 检测精度对NMS的阈值非常敏感



Repulsion Loss行人检测算法

- 行人检测中行人之间的遮挡所带来的影响
  - 使得算法在定位上不准确
  - 检测精度对NMS的阈值非常敏感
- 行人检测中存在的挑战
  - 如何在密集行人中，精确定位到行人
  - 不漏检、不虚检

Repulsion Loss行人检测算法

- 行人检测中行人之间的遮挡所带来的影响
  - 使得算法在定位上不准确
  - 检测精度对NMS的阈值非常敏感
- 行人检测中存在的挑战
  - 如何在密集行人中，精确定位到行人
  - 不漏检、不虚检
- 可行的解决方案
  - 让行人检测框回归的更好更合理

Repulsion Loss行人检测算法

Target Groundtruth T

Surrounding Groundtruth B

Predicted Box (for the target)

- SmoothL1回归损失函数

$$SmoothL1\ Loss = \underbrace{Dist_{attr}(\square, \square)}_{\text{Attraction Term}}$$

- 预测候选区域（红色虚线框）和对应的真实标注（棕色实线框）之间的偏差，从而让红色虚线框更加接近棕色实线框
- 故可称SmoothL1损失函数这一项为吸引项（Attraction Term）

Repulsion Loss行人检测算法

Target Groundtruth T

Surrounding Groundtruth B

Predicted Box (for the target)

- SmoothL1回归损失函数

$$SmoothL1\ Loss = \underbrace{Dist_{attr}(\square, \square)}_{\text{Attraction Term}}$$

- Repulsion回归损失函数

$$Repulsion\ Loss = \underbrace{Dist_{attr}(\square, \square)}_{\text{Attraction Term}} - \underbrace{Dist_{rep}(\square, \square)}_{\text{Repulsion Term}}$$

- 吸引项 (Attraction term): 让正样本接近目标物体（T）
- 排斥项 (Repulsion term): 让正样本远离非目标物体（B）

Repulsion Loss行人检测算法

- Repulsion回归损失函数的公式

$$L = L_{\text{Attr}} + \alpha * L_{\text{RepGT}} + \beta * L_{\text{RepBox}}$$

Repulsion Loss行人检测算法

- Repulsion回归损失函数的公式

$$L = L_{\text{Attr}} + \alpha * L_{\text{RepGT}} + \beta * L_{\text{RepBox}}$$

$$L_{\text{Attr}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{L1}(P, G^P_{Attr})}{|\mathcal{P}_+|}$$

- P_+为正样本集合， P是一个正样本，G^P_{Attr}是该正样本对应的真实标注
- L_{Attr}是SmoothL1回归损失函数，让正样本更加接近其对应的真实标注

Repulsion Loss行人检测算法

- Repulsion回归损失函数的公式

$$L = L_{\text{Attr}} + \alpha * L_{\text{RepGT}} + \beta * L_{\text{RepBox}}$$

$$L_{\text{Attr}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{L1}(P, G^P_{Attr})}{|\mathcal{P}_+|}$$

$$L_{\text{RepGT}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{ln}\left(IoG(P, G^P_{Rep})\right)}{|\mathcal{P}_+|}$$

$$IoG = \frac{\text{交集的面积}}{\text{真实标注的面积}}$$

- P_+为正样本集合， P是一个正样本，G^P_{Rep}是该正样本对应的次优的真实标注
- SmoothIn损失函数公式为：$Smooth_{ln} = -ln(1-x)$
- L_{RepGT}是让正样本P远离它的次优的真实标注

Repulsion Loss行人检测算法

- Repulsion回归损失函数的公式

$$L = L_{\text{Attr}} + \alpha * L_{\text{RepGT}} + \beta * L_{\text{RepBox}}$$

$$L_{\text{Attr}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{L1}(P, G^P_{Attr})}{|\mathcal{P}_+|}$$

$$L_{\text{RepGT}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{ln}\left(IoG(P, G^P_{Rep})\right)}{|\mathcal{P}_+|}$$

$$L_{\text{RepBox}} = \frac{\sum_{i \neq j} \text{Smooth}_{ln}\left(IoU(P_i, P_j)\right)}{\sum_{i \neq j} \mathbb{1}[IoU(P_i, P_j) > 0] + \epsilon}$$

- P_i是一个正样本， P_j另外一个跟P_i有交集的属于其他真实标注的正样本
- $Smooth_{ln} = x$
- L_{RepBox}是让正样本P远离属于其他真实标注的正样本

Repulsion Loss行人检测算法

- Repulsion回归损失函数的公式

$$L = L_{\text{Attr}} + \alpha * L_{\text{RepGT}} + \beta * L_{\text{RepBox}}$$

$$L_{\text{Attr}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{L1}(P, G^P_{Attr})}{|\mathcal{P}_+|}$$

$$L_{\text{RepGT}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{ln}\left(IoG(P, G^P_{Rep})\right)}{|\mathcal{P}_+|}$$

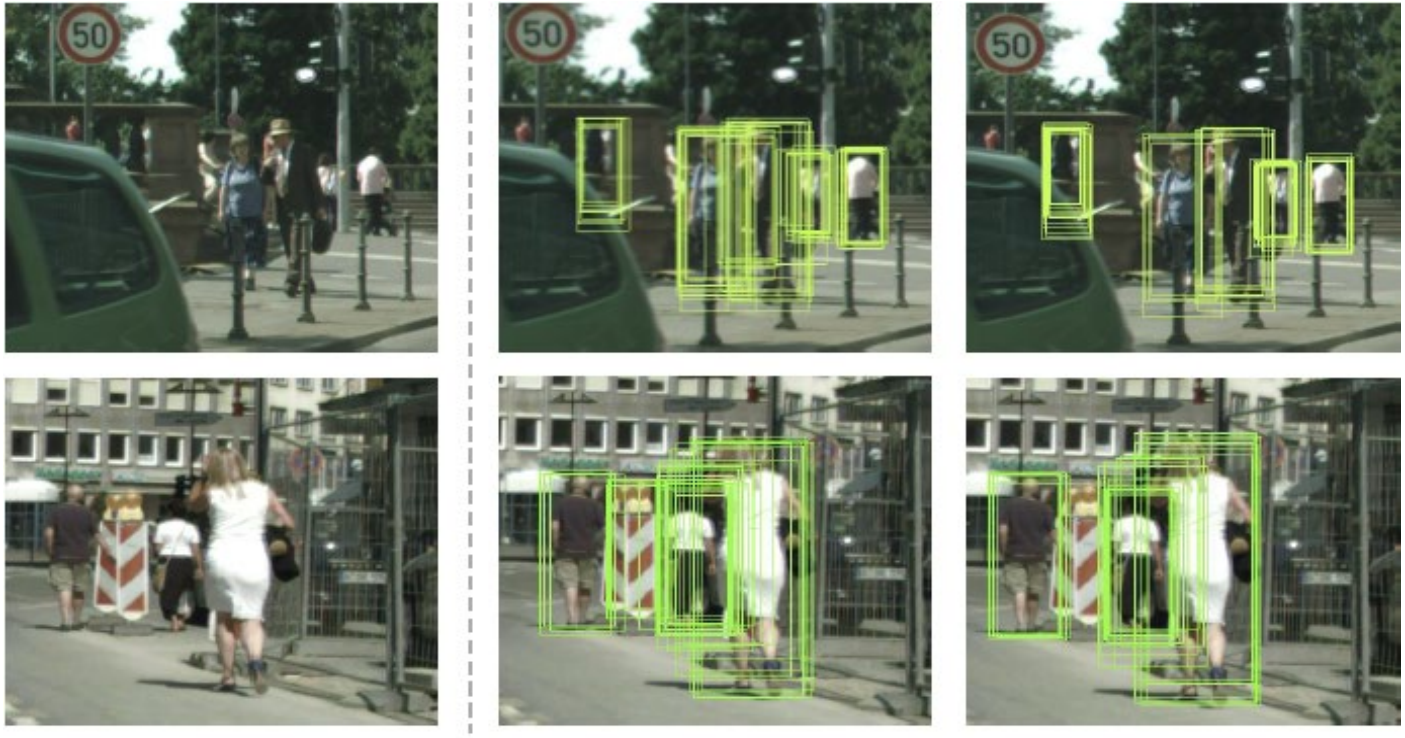
$$L_{\text{RepBox}} = \frac{\sum_{i \neq j} \text{Smooth}_{ln}\left(IoU(P_i, P_j)\right)}{\sum_{i \neq j} \mathbb{1}[IoU(P_i, P_j) > 0] + \epsilon}$$

- α和β是两个超参数，同为0.5时效果最好

α (RepGT)	0.3	0.4	0.5	0.6	0.7
β (RepBox)	0.7	0.6	0.5	0.4	0.3
MR^{-2}	13.9	13.9	**13.2**	13.3	14.1

Repulsion Loss行人检测算法：作用示意图

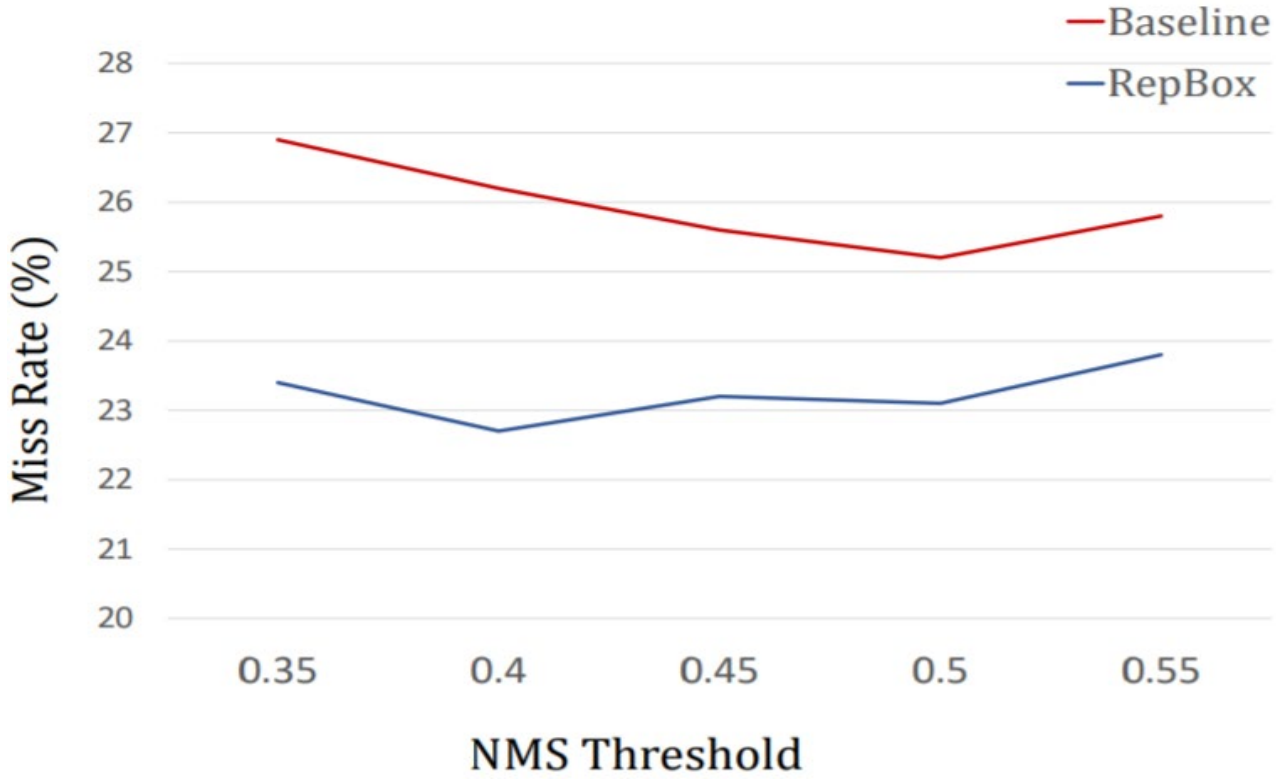
- Repulsion损失函数能让属于同一个行人的检测结果更加紧凑，不属于同一个行人的检测结果分得更开



(a) Input (b) Baseline (c) +RepBox

Repulsion Loss行人检测算法：作用示意图

- 与Baseline相比，Repulsion损失函数使得检测精度对非极大值抑制（NMS）的阈值更加鲁棒



Repulsion Loss行人检测算法：有效性验证

- Zhang et al. [31]就是Adapted FasterRCNN，但是没有开源训练代码
- Baseline是作者根据论文描述自己实现的Adapted FasterRCNN
- 所有结果都是在CityPersons的训练集上训练，验证集上测试得到的

Method	+RepGT	+RepBox	+Segmentation	Scale	Reasonable
Zhang *et al.* [31]				×1	15.4
			✓	×1	14.8
				×1.3	12.8
Baseline				×1	14.6
RepLoss	✓			×1	13.7
		✓		×1	13.7
	✓	✓		×1	13.2
	✓	✓		×1.3	11.6
	✓	✓		×1.5	10.9

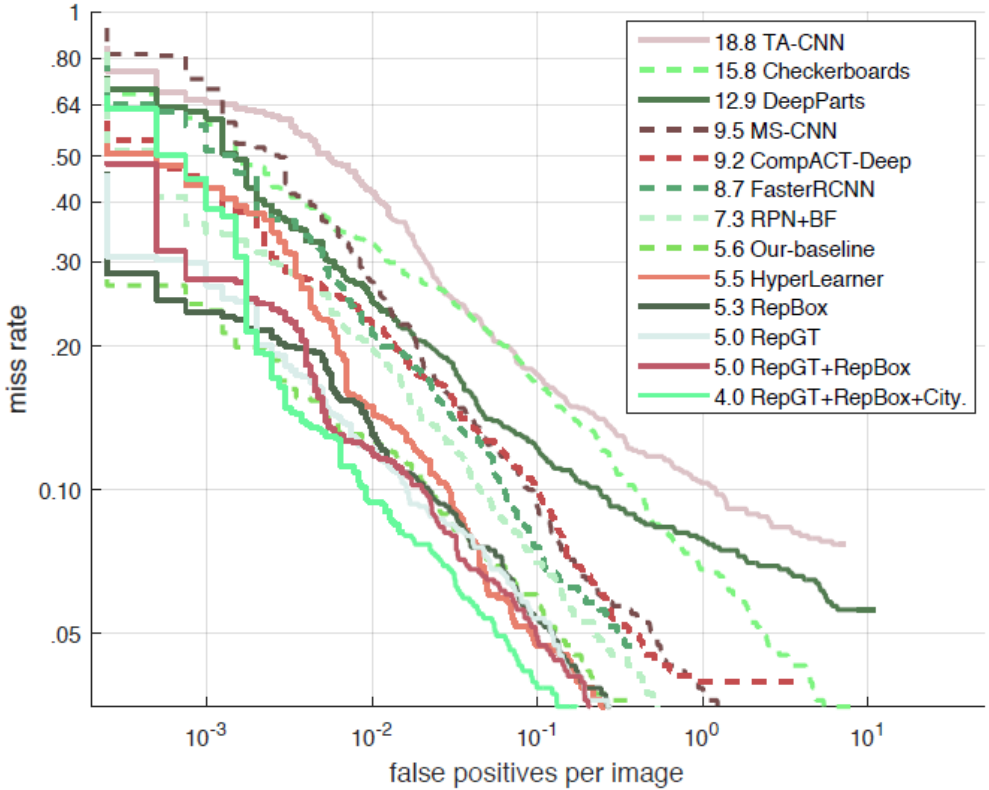
Repulsion Loss行人检测算法：有效性验证

Method	mAP	mAP on Crowd
Faster R-CNN [12]	76.4	-
Faster R-CNN (*ReIm*)	79.5	38.7
+ RepGT	**79.8**	**40.8**

- 在通用物体检测算法上验证Repulsion Loss的有效性
- 上表是在PASCAL VOC 2007上进行训练和测试得到的检测精度
- Faster R-CNN [12]是官方初始版本的检测精度
- Faster R-CNN (ReIm)是复现改进版本的检测精度
- 使用RepGT后能够在整个测试集上带来0.3%的提升
- 但在密集子集上能够带来2.1%的提升

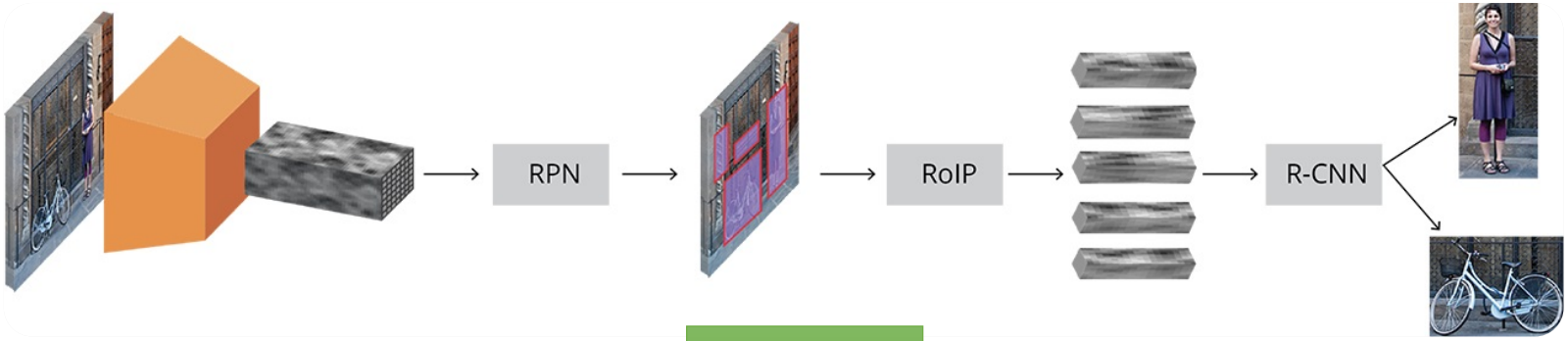
Repulsion Loss行人检测算法：检测精度

- 在CalTech数据集上的平均对数漏检率为4.0%



深度学习后期行人检测算法

- 深度学习后期行人检测算法：对**通用物体检测算法Faster R-CNN**进行**相应改进**，应用于**行人检测领域**



相应改进

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

深度学习后期行人检测算法：OR-CNN

- 2018年ECCV上发表的行人检测算法，基于Adapted FasterRCNN行人检测算法
- 为了解决遮挡问题，提出分块聚合的行人检测算法OR-CNN

Occlusion-aware R-CNN: Detecting Pedestrians in a Crowd

Shifeng Zhang[1,2][0000−0003−3109−5770], Longyin Wen[3][0000−0001−5525−492X], Xiao Bian[3][0000−0001−5447−6045], Zhen Lei[1,2⋆][0000−0002−0791−189X], and Stan Z. Li[4,1,2][0000−0002−2961−8096]

[1] Center for Biometrics and Security Research, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] GE Global Research, Niskayuna, NY
[4] Macau University of Science and Technology, Macau, China
{shifeng.zhang,zlei,szli}@nlpr.ia.ac.cn, {longyin.wen,xiao.bian}@ge.com

Occlusion-aware R-CNN: Detecting Pedestrians in a Crowd, ECCV 2018

OR-CNN行人检测算法：遮挡问题

- **类间互遮挡**：行人被其他非行人物体遮住，导致行人的部分特征消失，并引入了噪声特征

类间互遮挡

OR-CNN行人检测算法：遮挡问题

- **类间互遮挡**：行人被其他非行人物体遮住，导致行人的部分特征消失，并引入了噪声特征

类间互遮挡

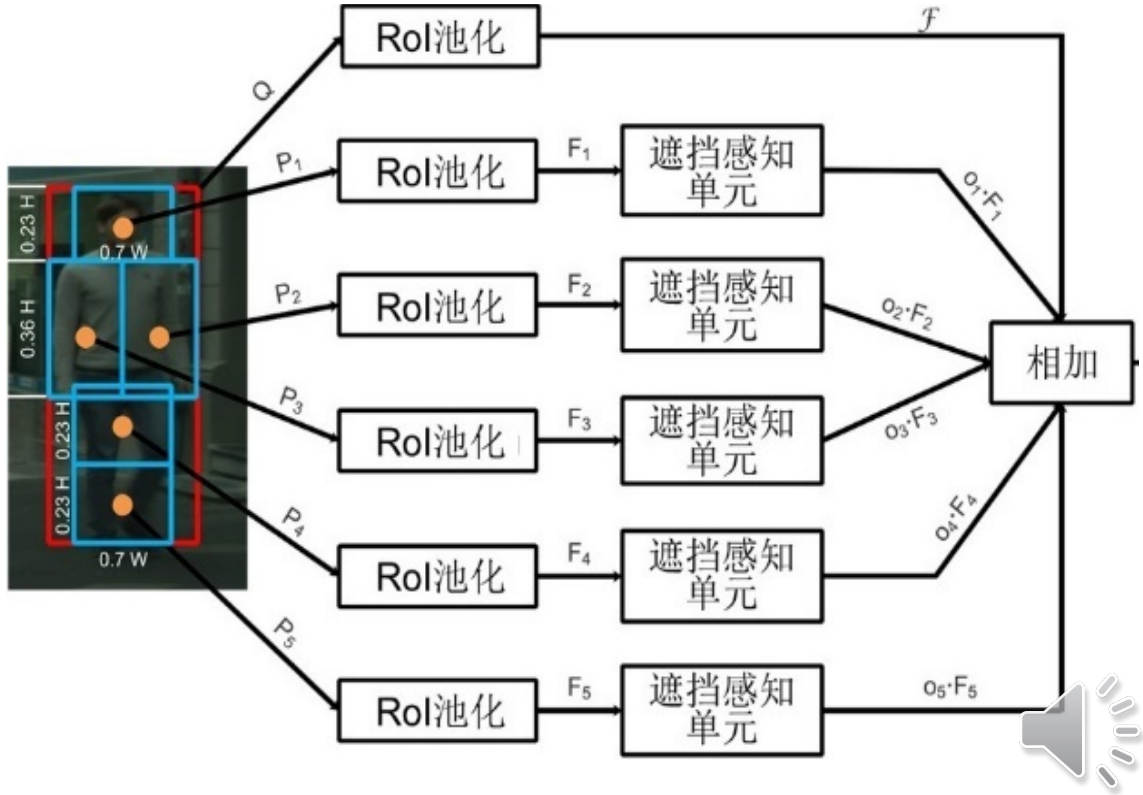
- **类内自遮挡**：行人之间相互遮挡，使得行人之间混在一块不易分开导致漏检或虚检

类内自遮挡

OR-CNN行人检测算法：类间互遮挡

- 分块遮挡感知的特征融合操作

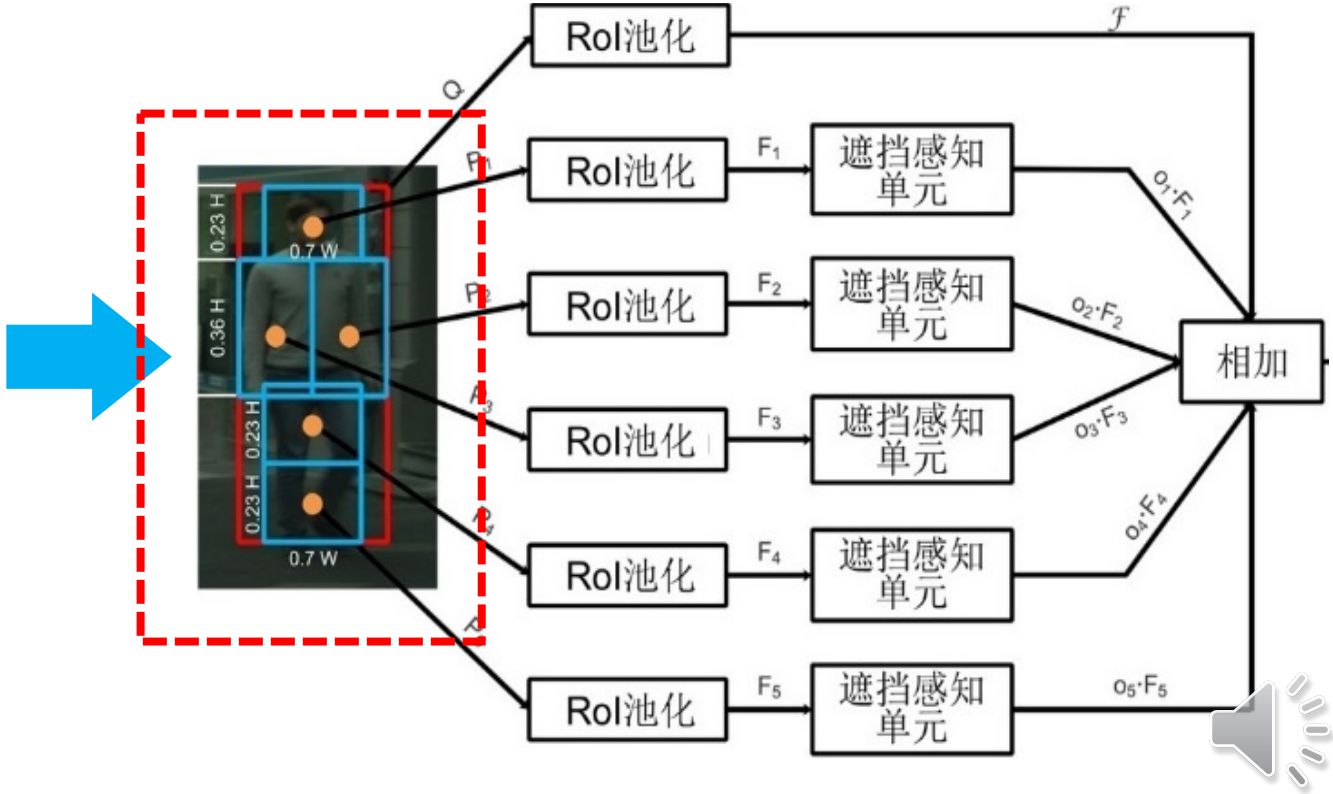
类间互遮挡



OR-CNN行人检测算法：类间互遮挡

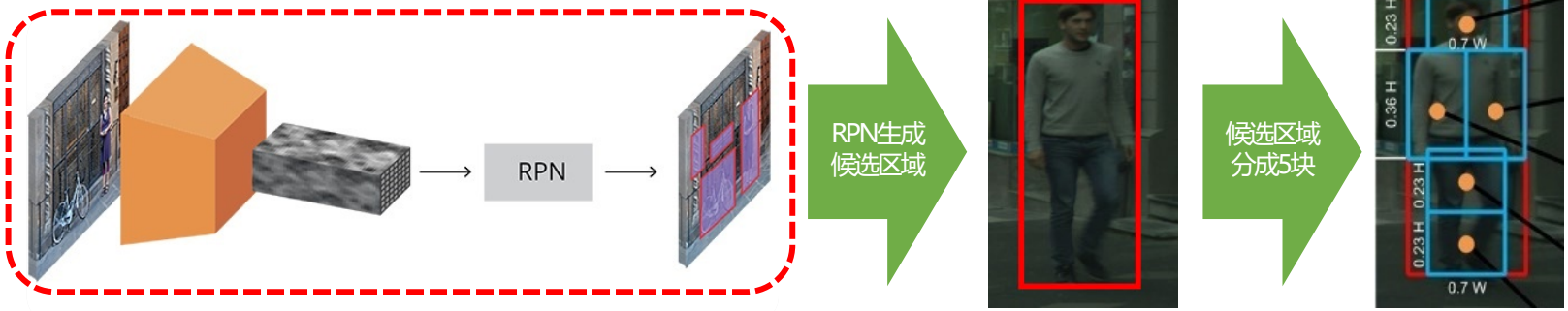
- 分块遮挡感知的特征融合操作

类间互遮挡



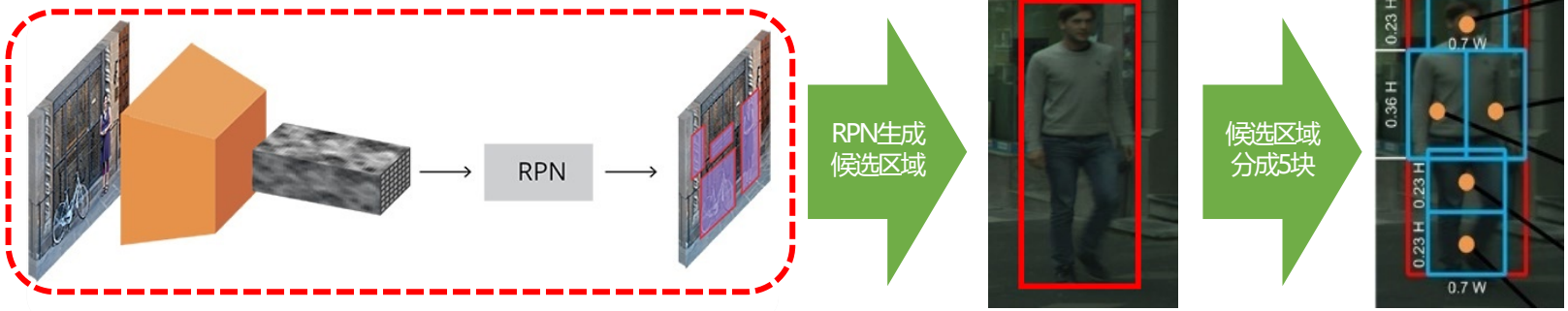
OR-CNN行人检测算法：类间互遮挡

- 候选区域分块：把每个候选区域分为5个部件

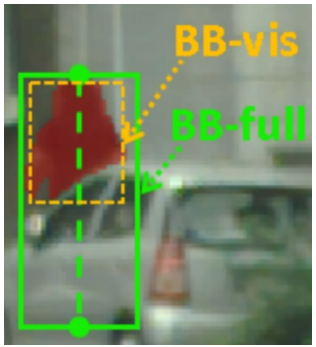


OR-CNN行人检测算法：类间互遮挡

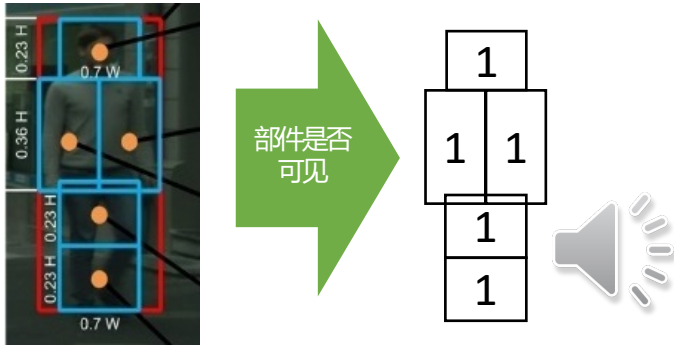
- 候选区域分块：把每个候选区域分为5个部件



- 部件可见性：判断每个部件是否可见，可见为1，不可见为0



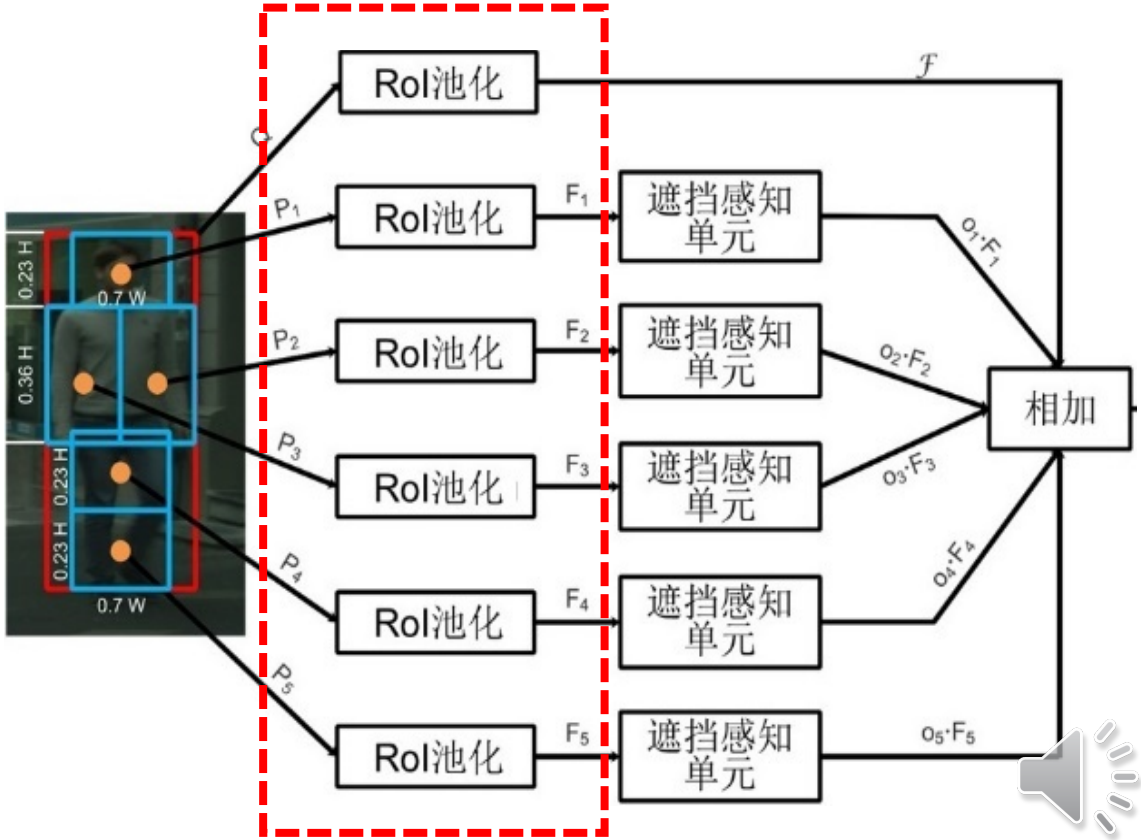
- 每个行人都有2个标注
- BB-full：完整行人的标注框
- BB-vis：可见区域的标注框
- 把BB-full分成5个部件，若部件有50%可见记为1，否则为0



OR-CNN行人检测算法：类间互遮挡

- 分块遮挡感知的特征融合操作

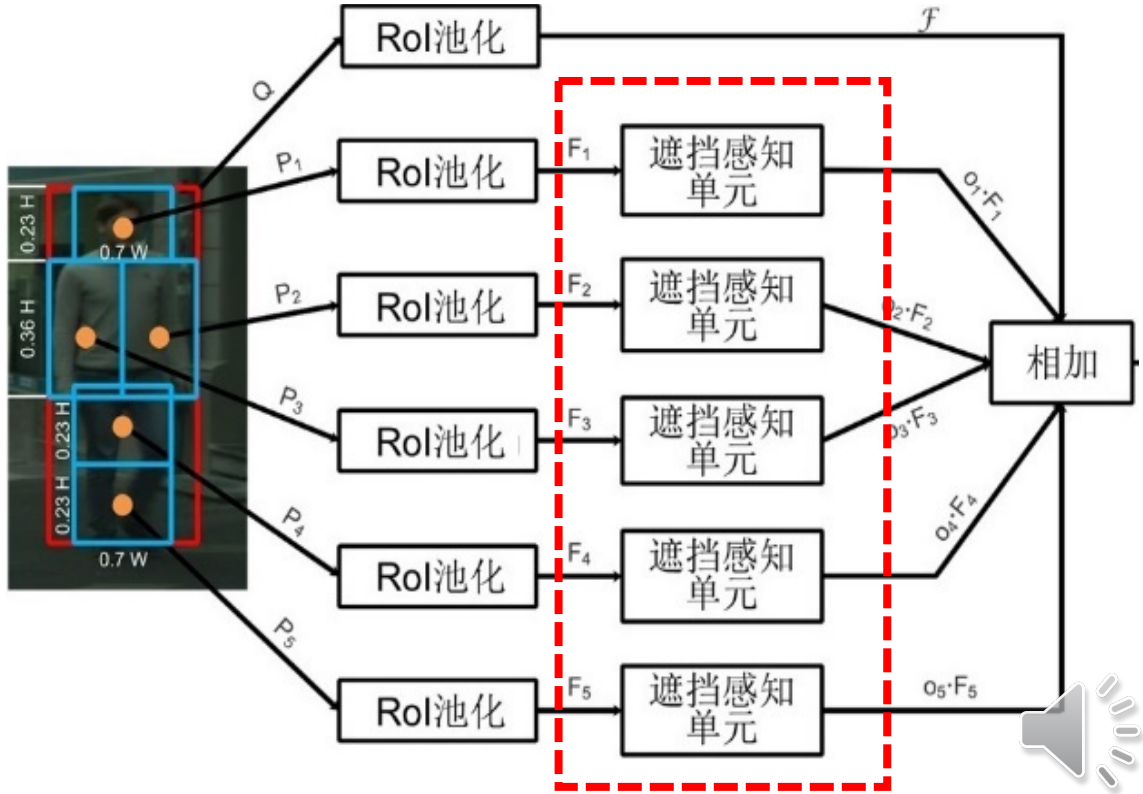
类间互遮挡



OR-CNN行人检测算法：类间互遮挡

- 分块遮挡感知的特征融合操作

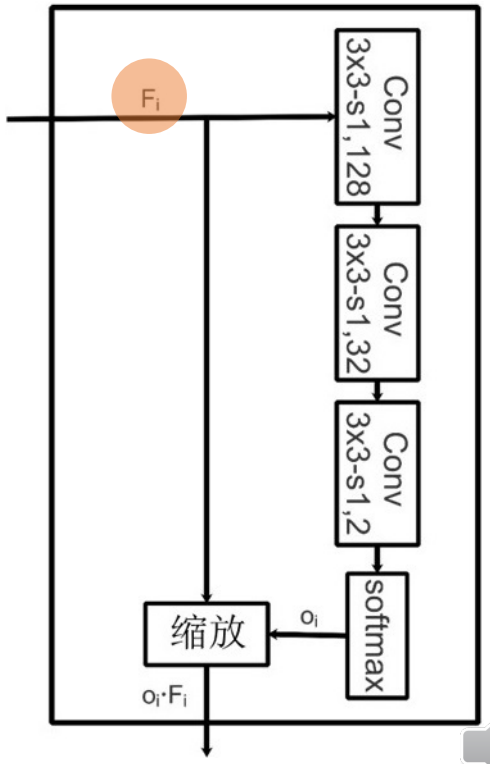
类间互遮挡



OR-CNN行人检测算法：类间互遮挡

■ **部件可见性的预测**

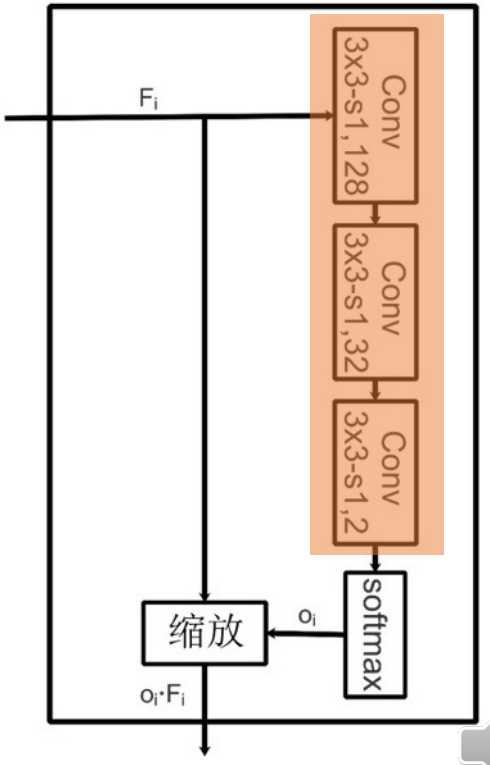
- 每个部件经过RoIPooling扣出7x7x512的特征F



OR-CNN行人检测算法：类间互遮挡

■ **部件可见性的预测**

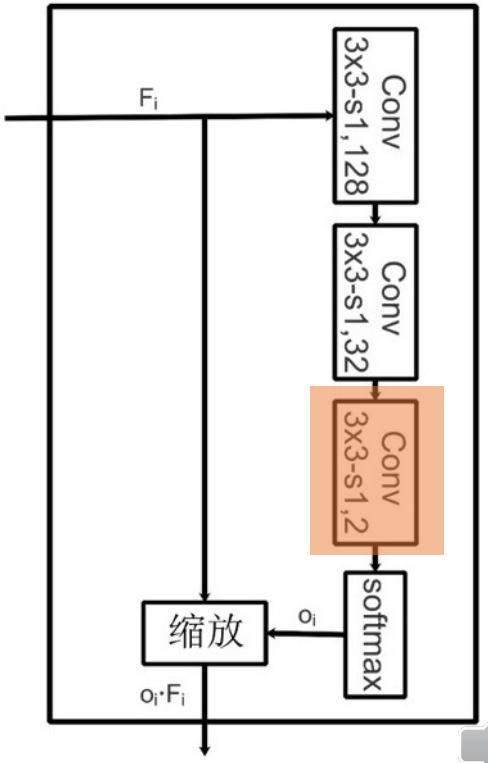
- 每个部件经过RoIPooling扣出7x7x512的特征F
- 经过3个不补零的卷积层3x3x128、3x3x32、3x3x2，输出一个1x1x2的特征来做二分类任务



OR-CNN行人检测算法：类间互遮挡

■ **部件可见性的预测**

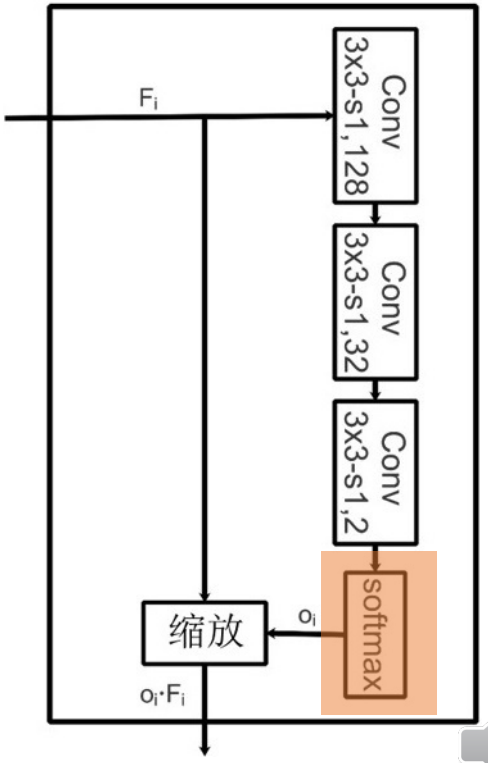
- 每个部件经过RoIPooling扣出7x7x512的特征F
- 经过3个不补零的卷积层3x3x128、3x3x32、3x3x2，输出一个1x1x2的特征来做二分类任务
- 1x1x2特征[o, i]代表着可见的预测值和不可见的预测值



OR-CNN行人检测算法：类间互遮挡

■ **部件可见性的预测**

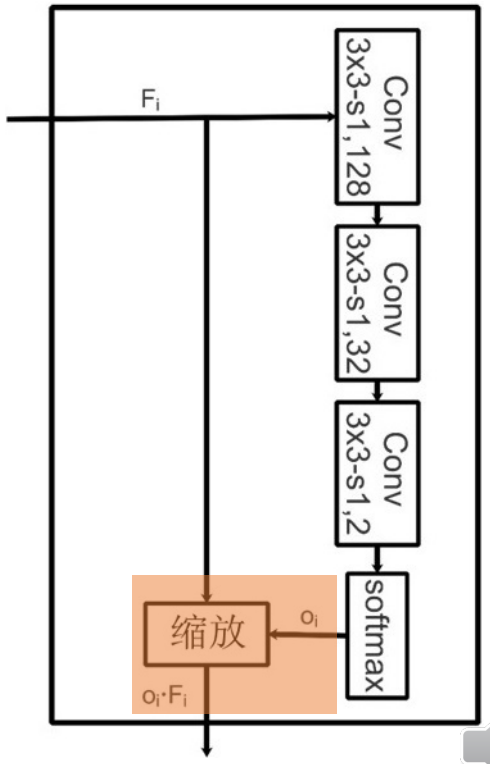
- 每个部件经过RoIPooling扣出7x7x512的特征F
- 经过3个不补零的卷积层3x3x128、3x3x32、3x3x2，输出一个1x1x2的特征来做二分类任务
- 1x1x2特征[o, i]代表着可见的预测值和不可见的预测值
- 把预测值和真实值送入softmax损失函数进行训练



OR-CNN行人检测算法：类间互遮挡

■ **部件可见性的预测**

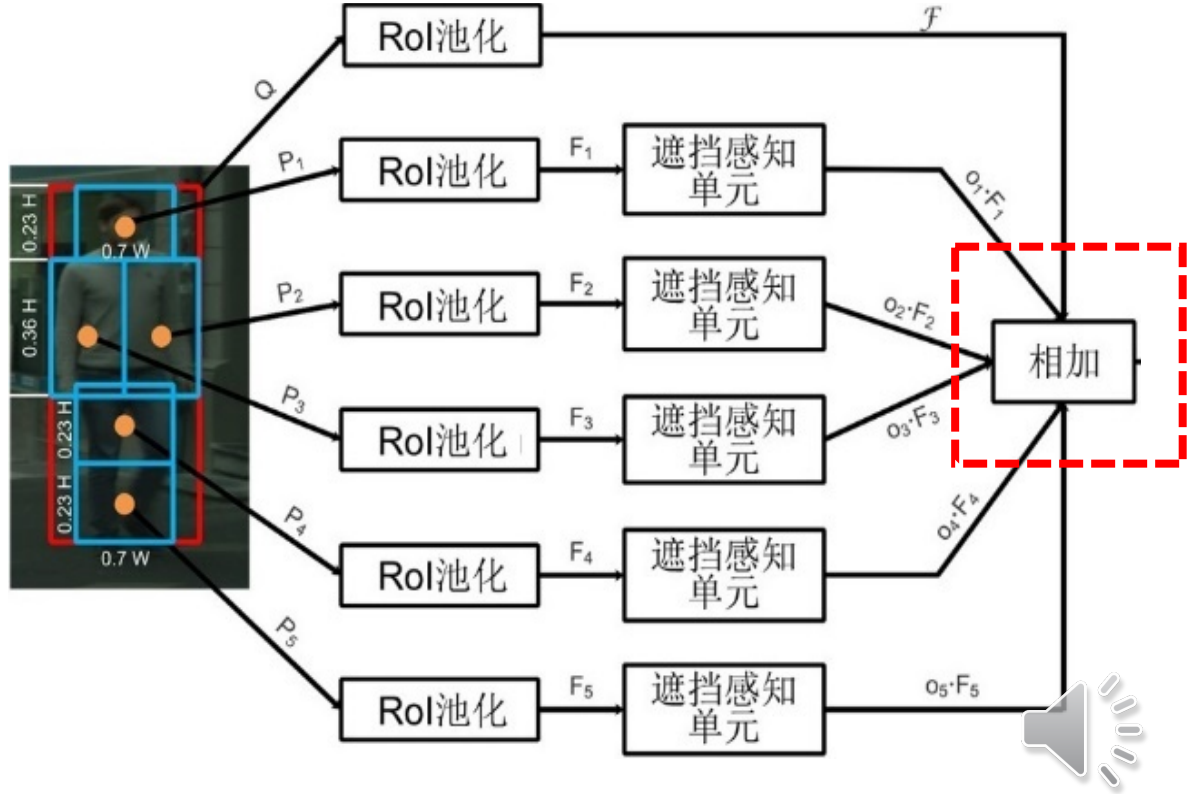
- 每个部件经过RoIPooling扣出7x7x512的特征F
- 经过3个不补零的卷积层3x3x128、3x3x32、3x3x2，输出一个1x1x2的特征来做二分类任务
- 1x1x2特征[o, i]代表着可见的预测值和不可见的预测值
- 把预测值和真实值送入softmax损失函数进行训练
- 取出预测值中的可见性得分o，乘以原始7x7x512特征F，输出经过可见性得分缩放后的特征o·F



OR-CNN行人检测算法：类间互遮挡

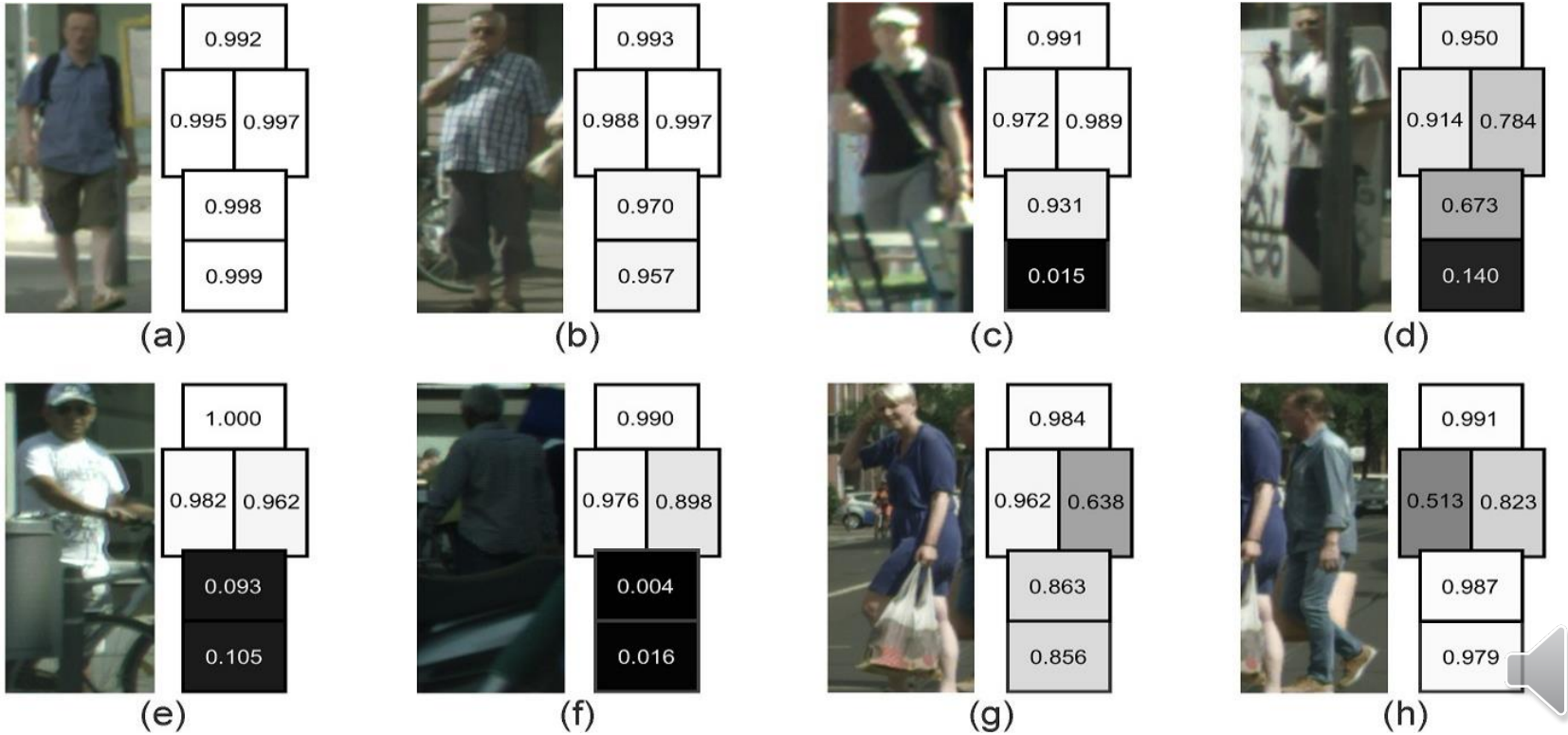
- 分块遮挡感知的特征融合操作: $F_{out} = F + o_1 \cdot F_1 + o_2 \cdot F_2 + o_3 \cdot F_3 + o_4 \cdot F_4 + o_5 \cdot F_5$

类间互遮挡



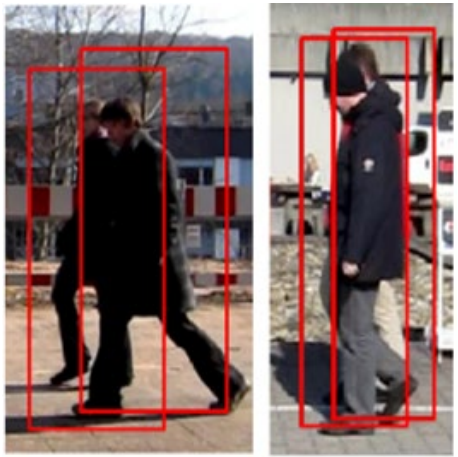
OR-CNN行人检测算法：类间互遮挡

- 分块遮挡感知的特征融合操作：加强可见区域的特征，弱化遮挡区域的特征，从而提取有效特征

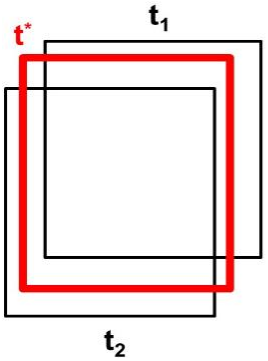


OR-CNN行人检测算法：类内自遮挡

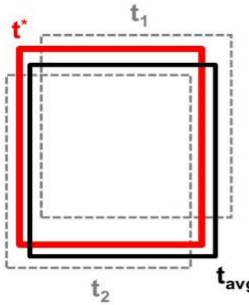
- 聚合损失函数



类内自遮挡



$$L_{reg}(\{p_i^*\},\{t_i\},\{t_i^*\}) = \frac{1}{N_{reg}}\sum_i p_i^* \triangle (t_i^* - t_i) = \frac{1}{2}[\triangle (t^* - t_1) + \triangle (t^* - t_2)]$$

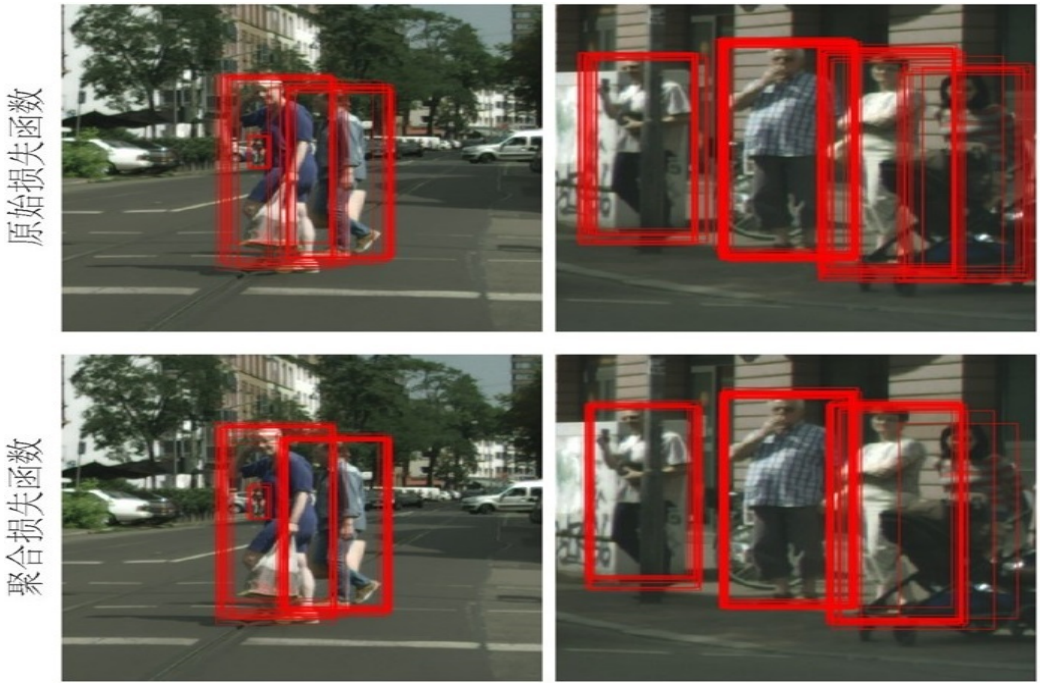


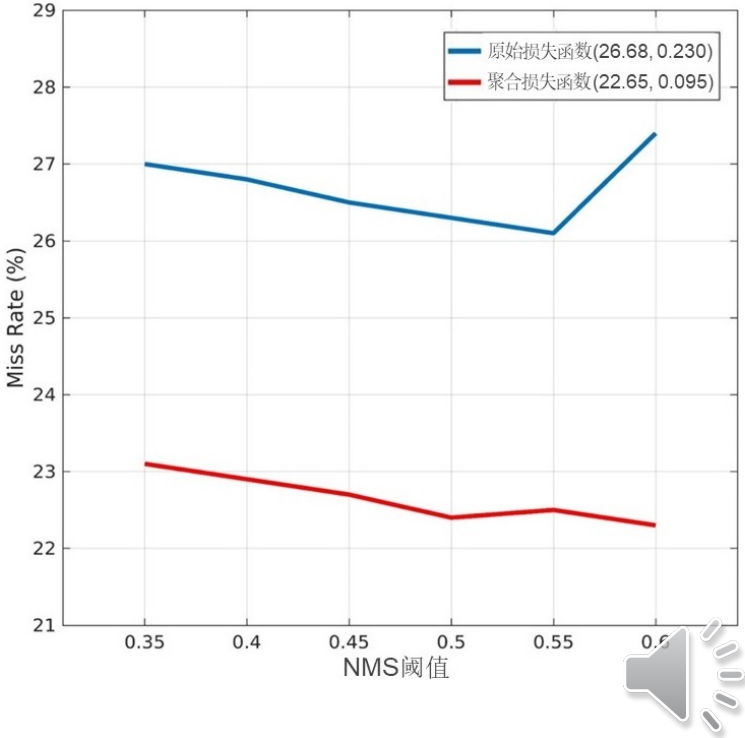
$$L_{com}(\{p_i^*\},\{t_i\},\{t_i^*\}) = \frac{1}{N_{com}}\sum_{i=1}^{p} \triangle \left(\tilde{t}_i^* - \frac{1}{|\phi_i|}\sum_{j\in\phi_i} t_j\right) = \triangle (t^* - t_{avg})$$

- t_1和t_2是两个检测结果框
- t*是一个真实标注框
- t_{avg}是t_1和t_2检测结果框的平均框

OR-CNN行人检测算法：类内自遮挡

- 聚合损失函数：每个行人对应的检测结果更加紧凑，对NMS阈值更加鲁棒





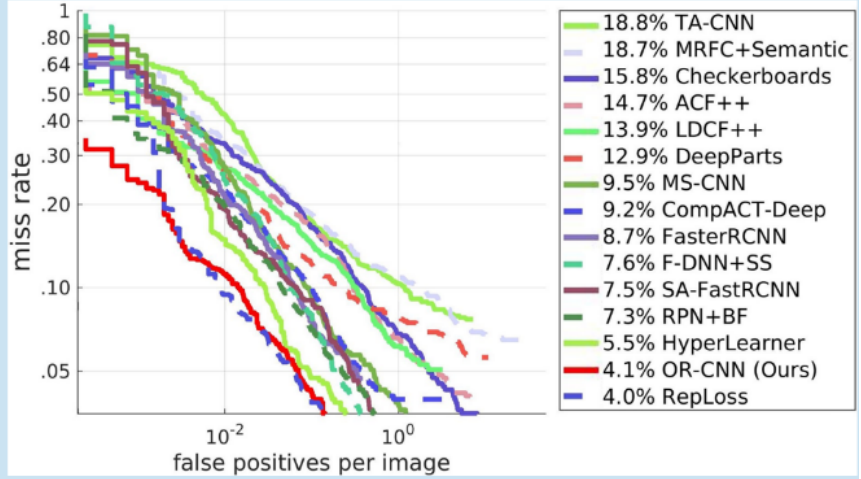
OR-CNN行人检测算法：检测精度

- 常用行人检测数据集上
- 取得最佳的性能
- CityPersons: 11.32%
- Caltech: 4.1%
- ETH: 24.5%
- INRIA: 6.4%

CityPersons

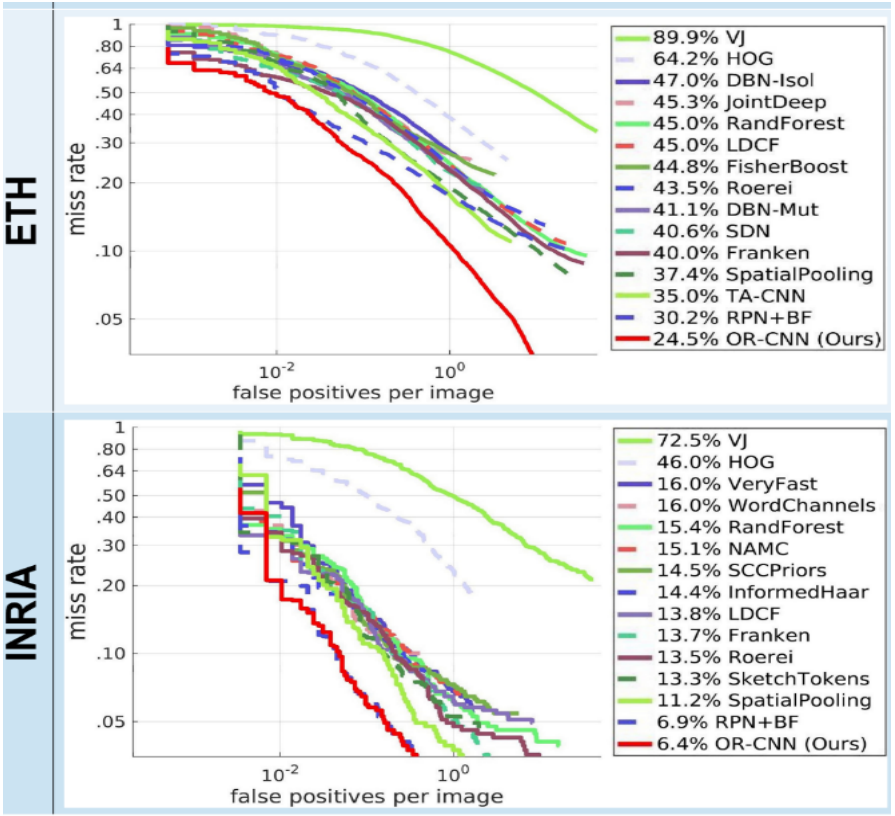
Method	Backbone	Scale	*Reasonable*	*Reasonable_Small*
Adapted Faster RCNN	VGG-16	× 1.3	12.97	37.24
Repulsion Loss	ResNet-50	× 1.5	11.48	15.67
OR-CNN	VGG-16	× 1.3	**11.32**	**14.19**

Caltech



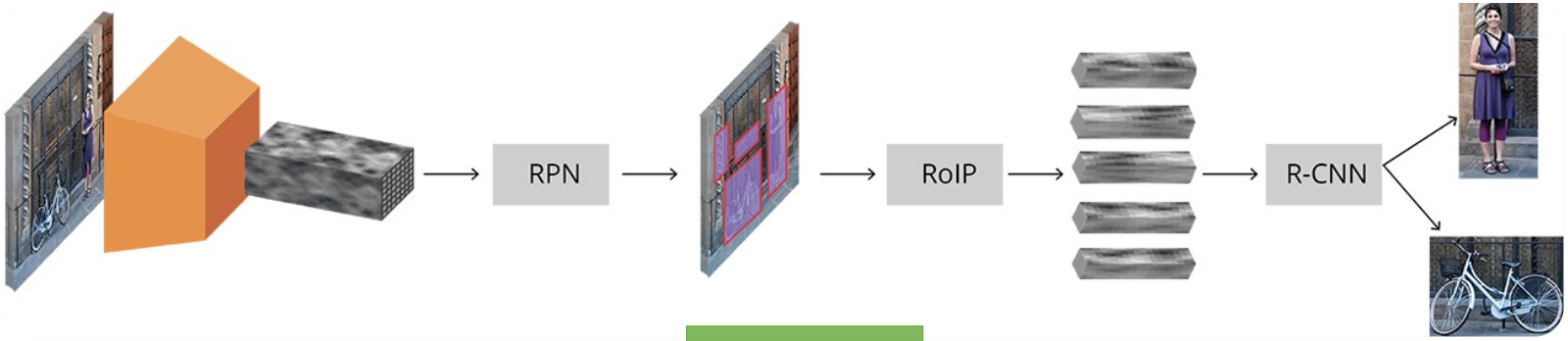
OR-CNN行人检测算法：检测精度

- 常用行人检测数据集上
- 取得最佳的性能
- CityPersons: 11.32%
- Caltech: 4.1%
- ETH: 24.5%
- INRIA: 6.4%



深度学习后期行人检测算法

- 深度学习后期行人检测算法：对**通用物体检测算法Faster R-CNN**进行**相应改进**，应用于**行人检测领域**



相应改进

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

深度学习后期行人检测算法：JointDet

- 2020年AAAI上发表的行人检测算法，基于通用物体检测算法Faster R-CNN
- 利用人头人体的结构关系进行联合检测，抑制人头虚检，召回人体漏检

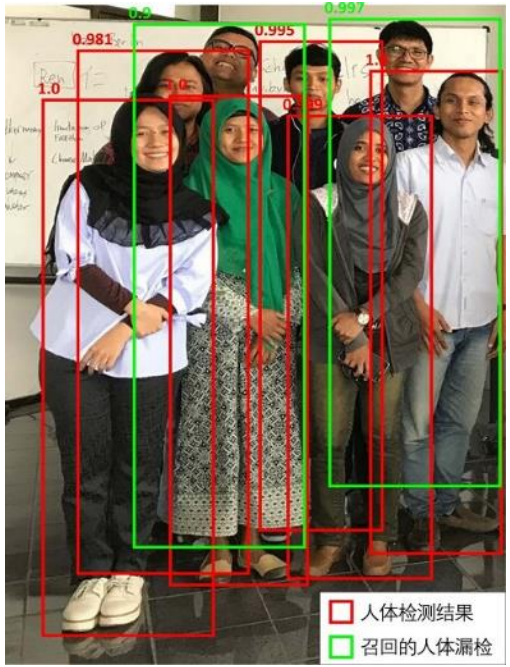
Relational Learning for Joint Head and Human Detection

Cheng Chi[1,3*], **Shifeng Zhang**[2,3*], **Junliang Xing**[2,3], **Zhen Lei**[2,3], **Stan Z. Li**[2,3], **Xudong Zou**[1,3]
[1] Institute of Electronics, Chinese Academy of Sciences, Beijing, China
[2]CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China

Relational Learning for Joint Head and Human Detection, AAAI 2020

JointDet行人检测算法：背后动机

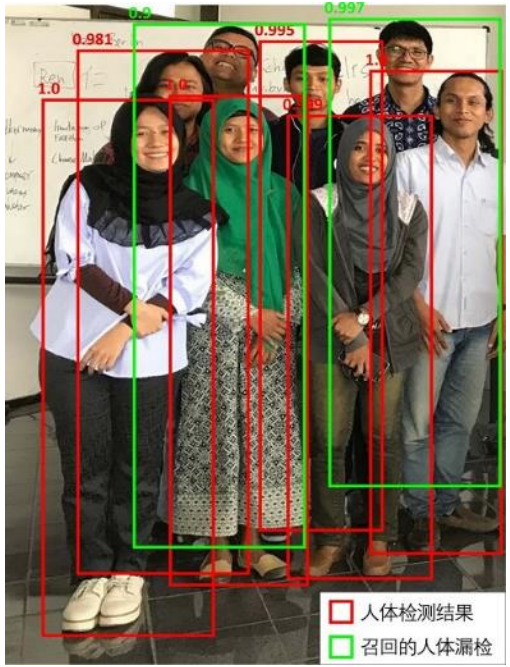
- **密集人群中的行人检测**：行人之间遮挡程度很严重，NMS会错误地抑制掉正确的检测结果



行人检测的问题

JointDet行人检测算法：背后动机

- **密集人群中的行人检测**：行人之间遮挡程度很严重，NMS会错误地抑制掉正确的检测结果



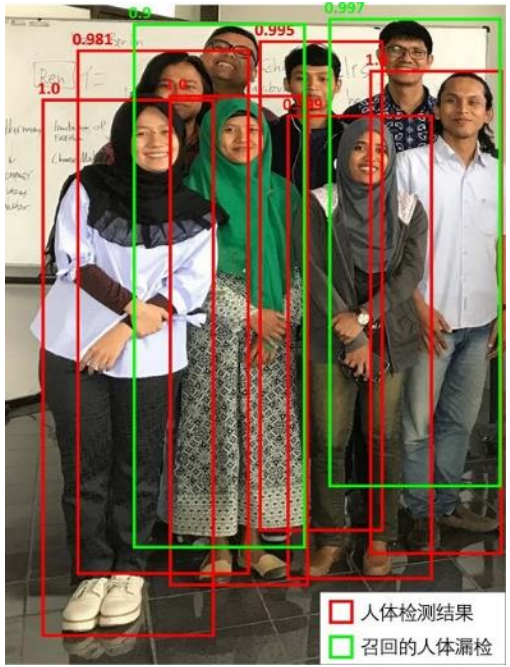
行人检测的问题

- **复杂场景下的人头检测**：缺少上下文信息，容易出现类似人头的虚检

人头检测的问题

JointDet行人检测算法：背后动机

- **密集人群中的行人检测**：行人之间遮挡程度很严重，NMS会错误地抑制掉正确的检测结果



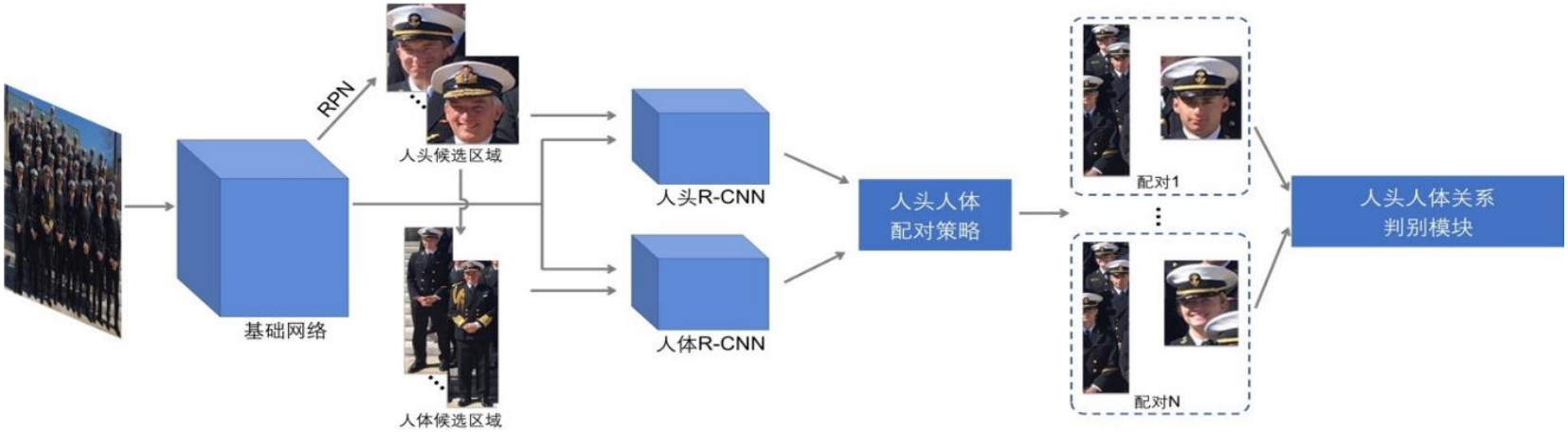
行人检测的问题

利用人头人体的结构关系进行联合检测，来同时解决这2个问题

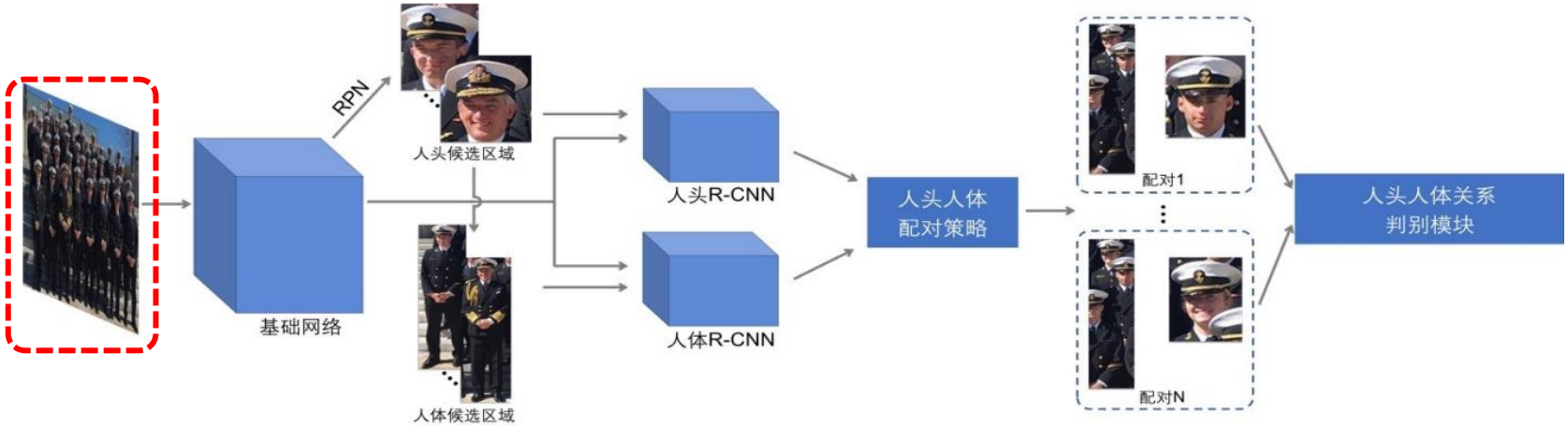
- **复杂场景下的人头检测**：缺少上下文信息，容易出现类似人头的虚检

人头检测的问题

JointDet行人检测算法：整体框架

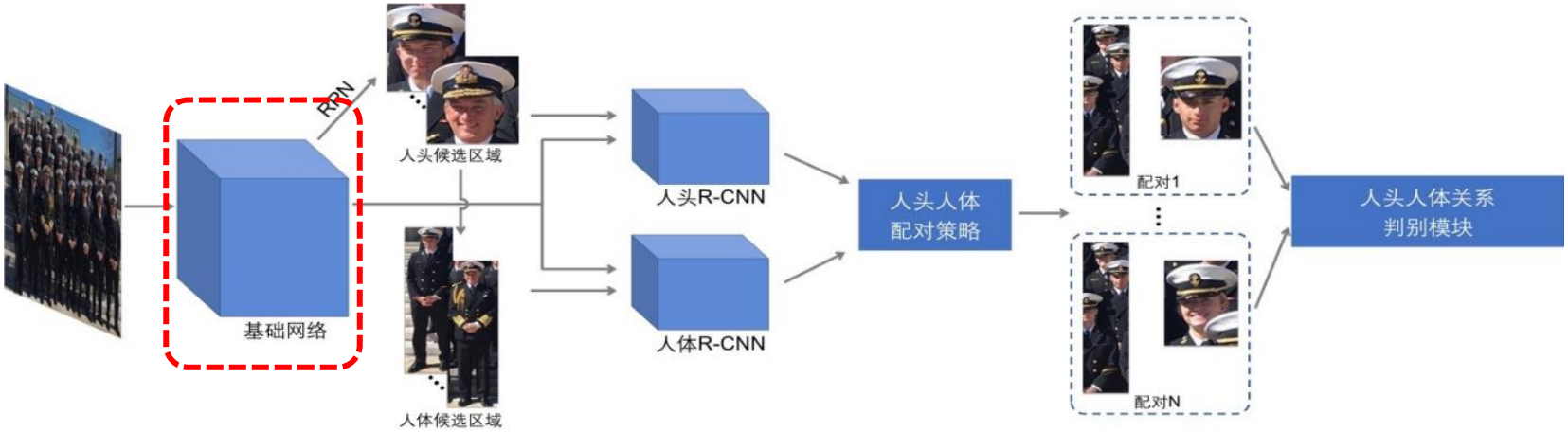


JointDet行人检测算法：整体框架



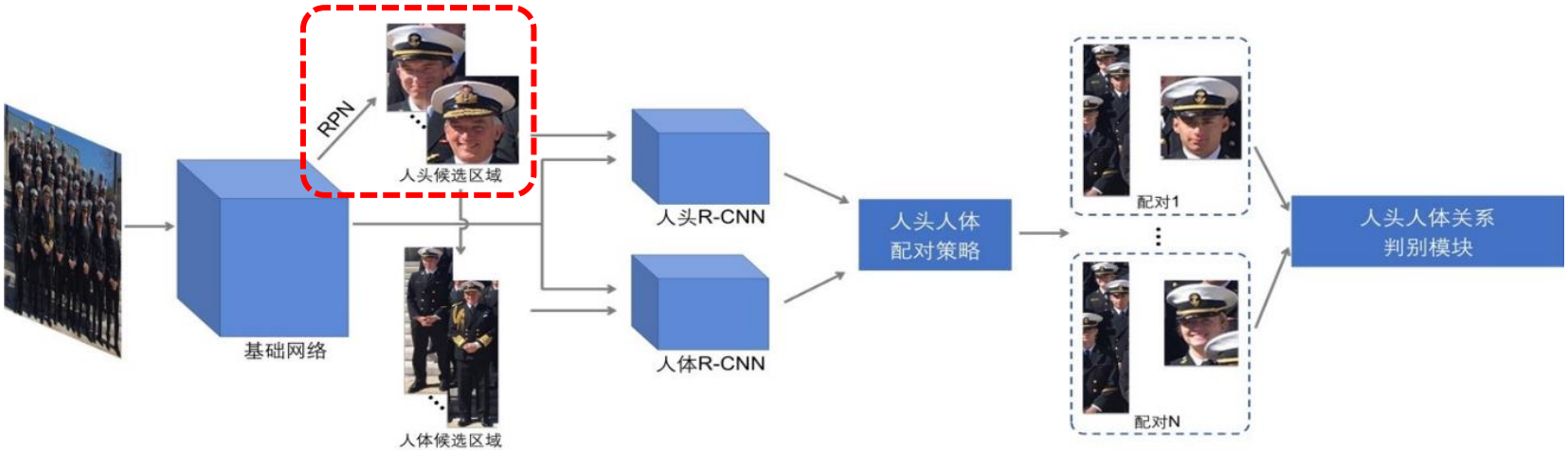
- 输入图像 + 人头标注 + 人体标注

JointDet行人检测算法：整体框架



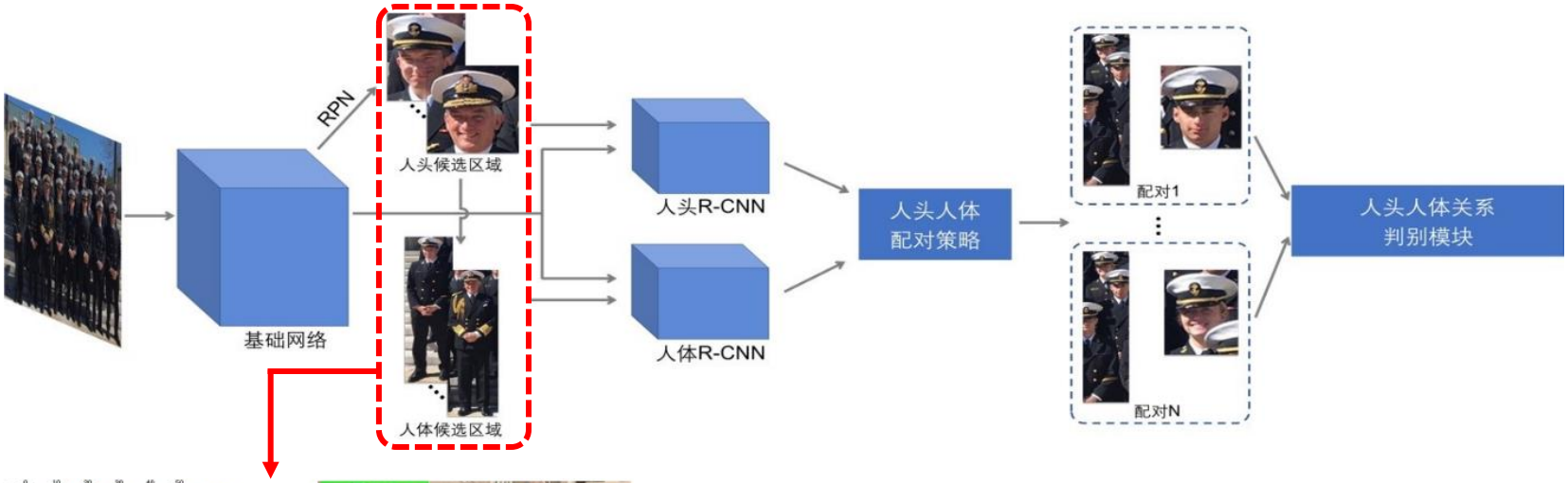
- 基础网络ResNet-50

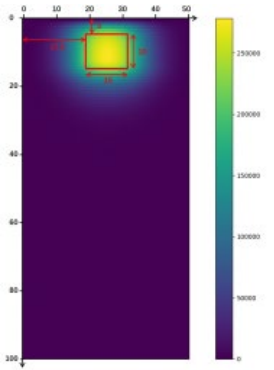
JointDet行人检测算法：整体框架

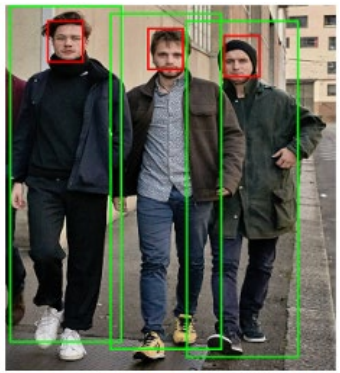


- 利用RPN仅生成人头的候选区域
- 人头的长宽比例固定为1:1，其尺度较集中
- 可以铺设较少锚框，高效生成人头的候选区域

JointDet行人检测算法：联合检测

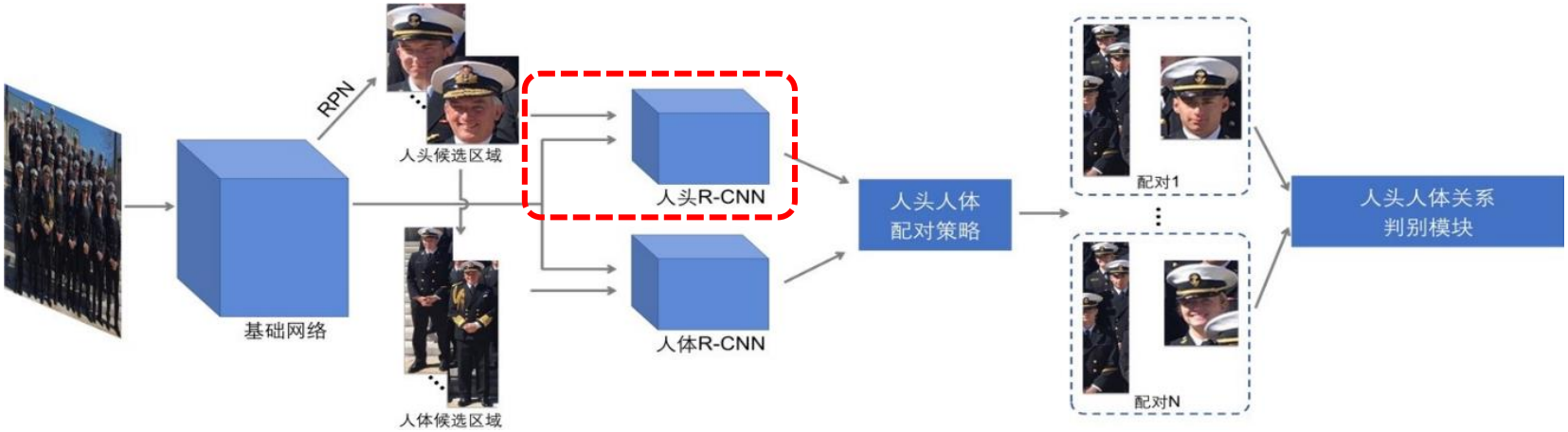






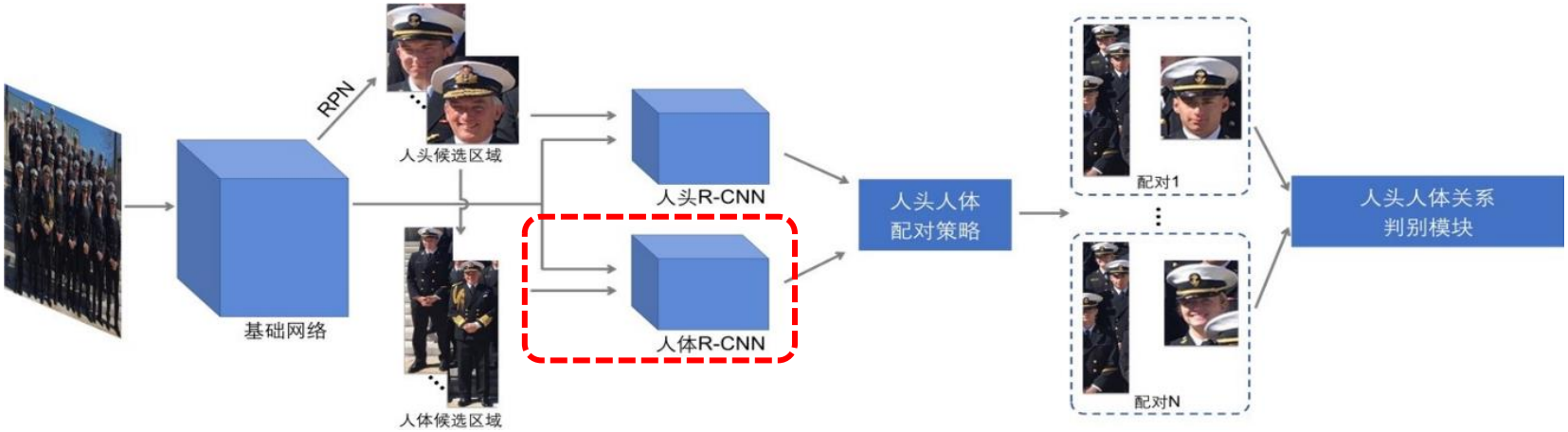
- 人体有各种比例，且尺度比较分散
- 需要铺设较多锚框才能生成人体候选区域
- 提出一种高效的人体候选区域的生成方式
- 利用统计得到的人头人体结构关系
- 根据人头的候选区域，直接生成人体的候选区域

JointDet行人检测算法：联合检测



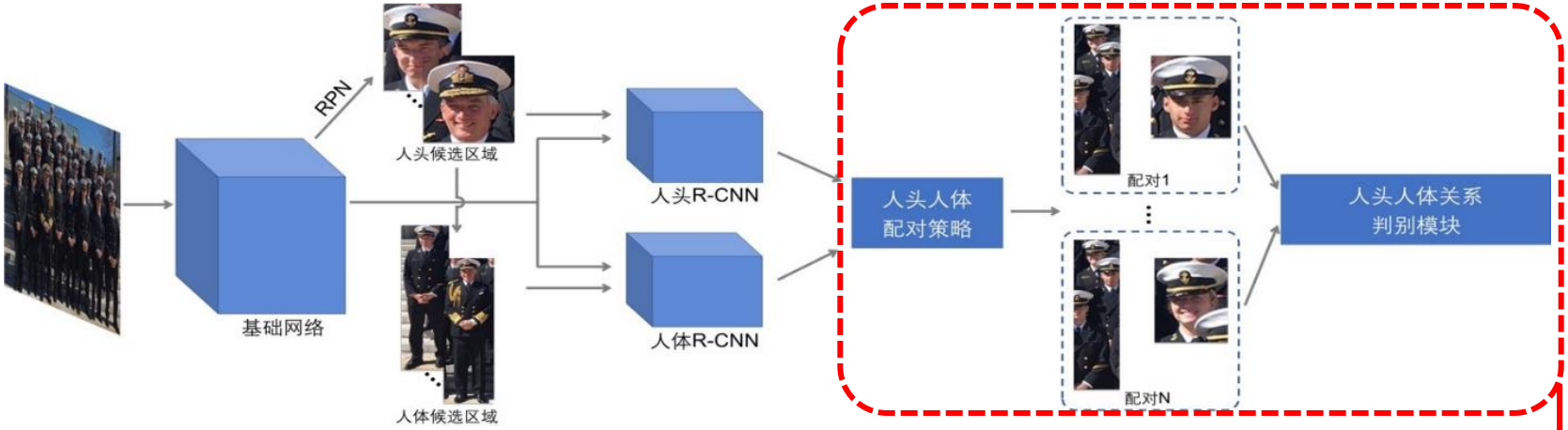
- 扣取人头候选区域的特征，送入人头R-CNN分支，进行下一步分类和回归

JointDet行人检测算法：联合检测



- 扣取人体候选区域的特征，送入人体R-CNN分支，进行下一步分类和回归

JointDet行人检测算法：关系学习



- 人体是由人头生成，所以存在一一对应关系
- 找出那些没有对应的人体检测结果的人头检测结果
- 如果它对应的人体检测结果得分很高，那么召回这个人体检测结果
- 如果它对应的人体检测结果得分很低，那么抑制该人头检测结果

JointDet行人检测算法：检测精度

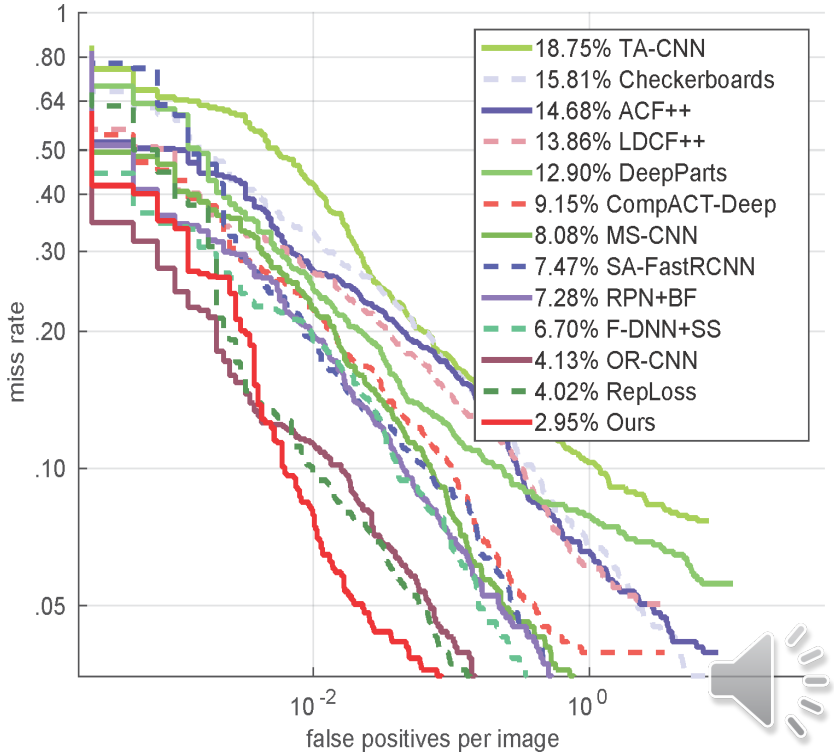
- CrowdHuman数据集

Source	Method	Head	Human
CrowdHuman	FPN-Head	52.1	-
	FPN-Human	-	50.4
Ours	FPN-Head	48.9	-
	FPN-Human	-	49.7
	FPN-Human-Cascade	-	49.2
	JointDet w/o RDM	48.7	47.0
	JointDet	**48.3**	**46.5**

- CityPersons数据集

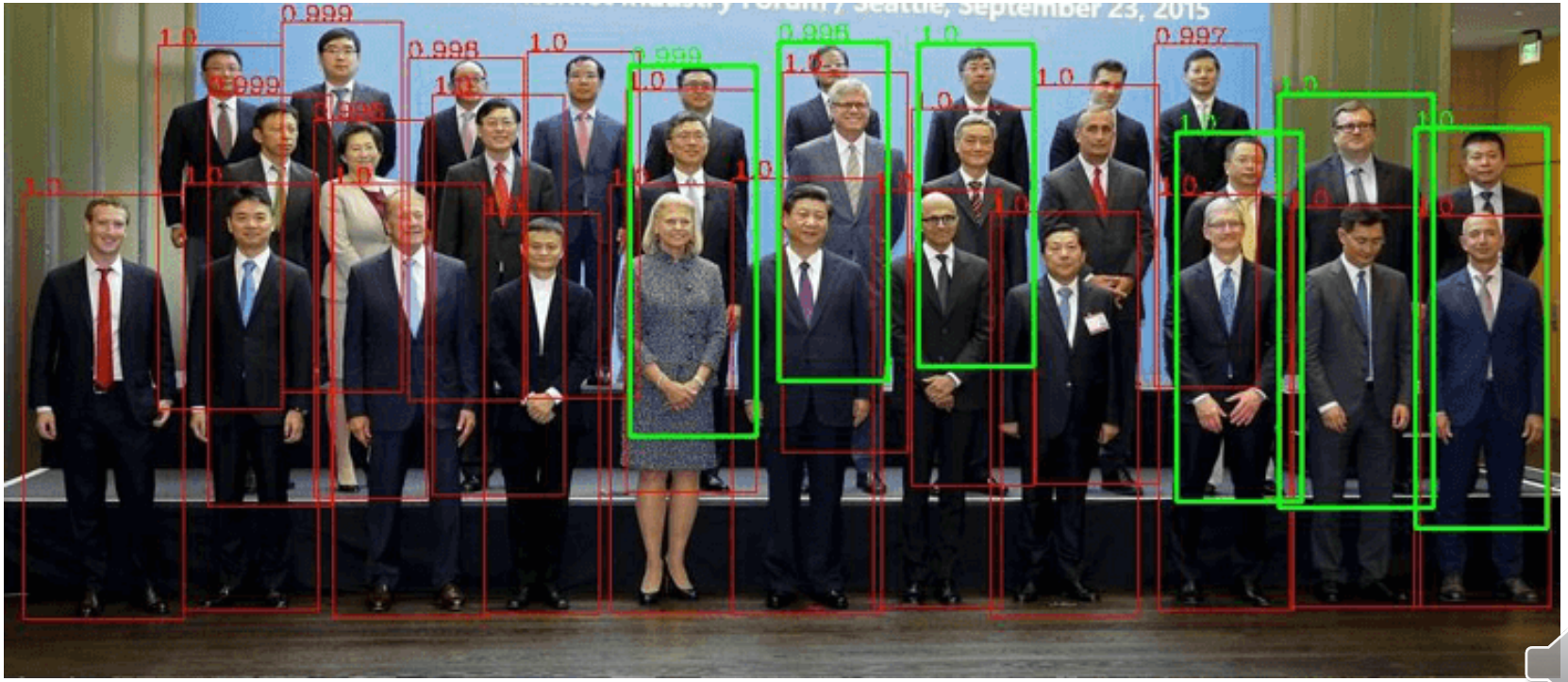
Method	Backbone	Scale	*Reasonable*
TLL(MRF)	ResNet-50	-	14.40
Adapted FasterRCNN	VGG-16	×1.3	12.97
ALFNet	VGG-16	×1	12.00
Repulsion Loss	ResNet-50	×1.3	11.60
PODE+RPN	VGG-16	-	11.24
OR-CNN	VGG-16	×1.3	11.00
JointDet (Ours)	ResNet-50	×**1.3**	**10.23**

- Caltech数据集



JointDet行人检测算法：召回漏检

- 召回被NMS错误抑制的人体检测结果

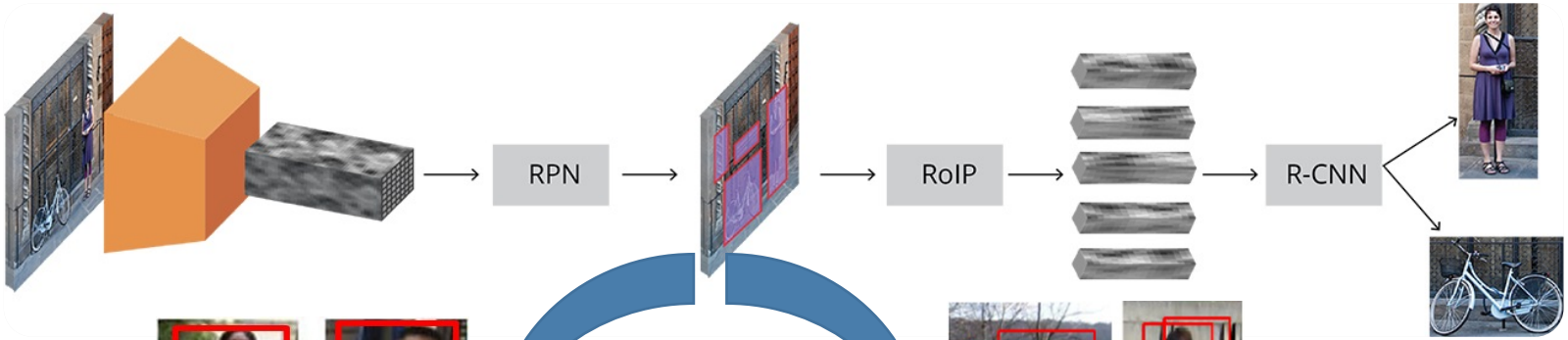


JointDet行人检测算法：抑制虚检

- 抑制错误的人头虚检

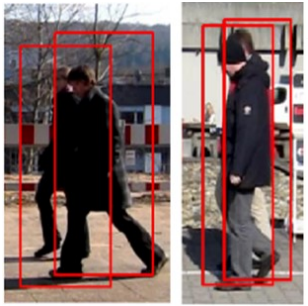
深度学习后期行人检测算法：总结

- 深度学习后期行人检测算法：基于**通用物体检测算法Faster R-CNN**，针对**遮挡问题**进行**相应改进**



- 回归损失函数
- 分块预测思想
- 人头人体结构

类间互遮挡



类内自遮挡

行人检测算法	Adapted FasterRCNN	Repulsion Loss	OR-CNN	JointDet

课程作业

- **熟悉Pedestron行人检测算法平台**

1. Pedestron行人检测算法平台链接（https://github.com/hasanirtiza/Pedestron）
2. 按照安装教程，利用Anaconda配置好相应的环境
3. 按照数据教程中的步骤，准备好行人检测数据集
4. 熟悉Pedestron行人检测算法平台中，实现了哪些行人检测算法
5. 下载各个行人检测算法训好的模型，分别输出demo中3张图片的检测结果图
6. （可选）测试或训练Pedestron行人检测算法平台的现有算法

感谢各位聆听!

Thanks for Listening