深蓝学院
shenlanxueyuan.com

行人检测

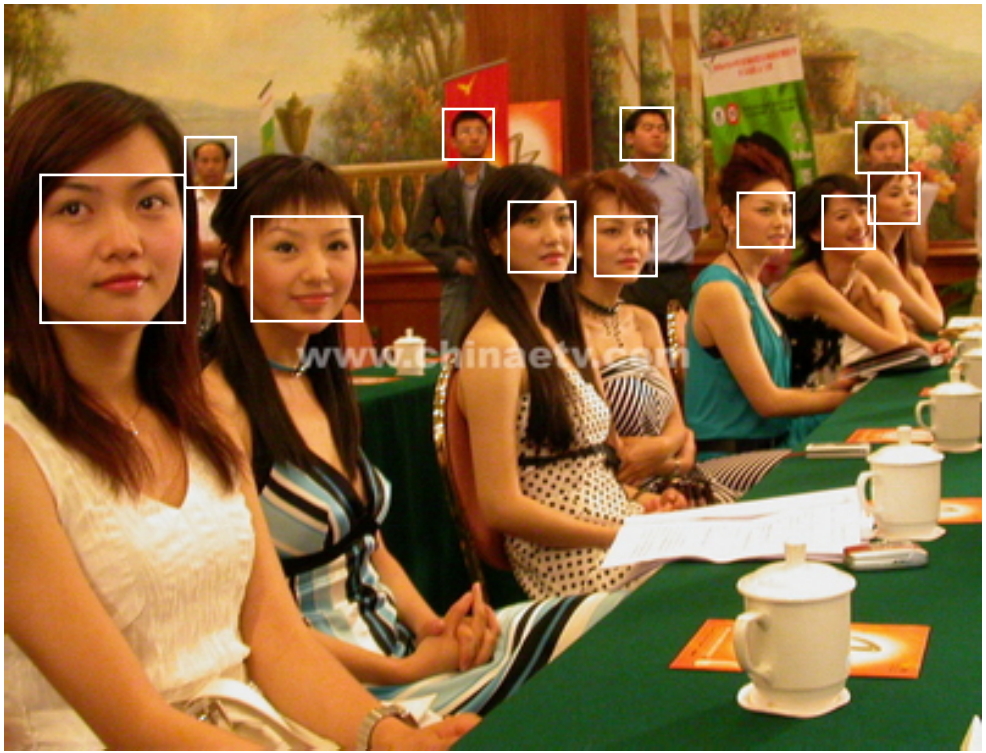
主讲人 张士峰

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：传统人脸检测算法

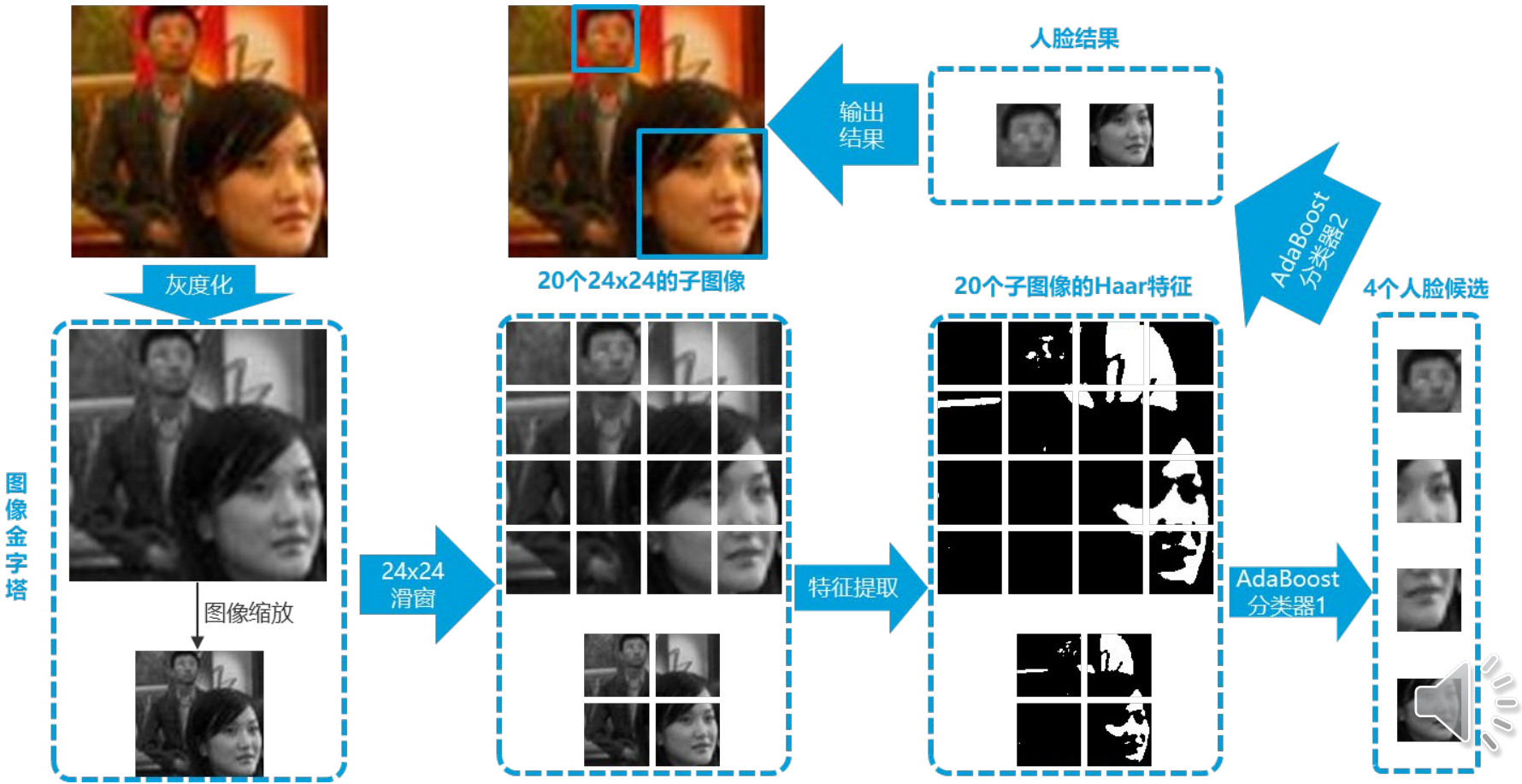
- 利用手工特征+分类器，以滑窗方式在图像金字塔上遍历所有位置和大小，进行人脸检测

遍历所有位置
滑窗方式



遍历不同大小
图像金字塔

内容回顾：传统人脸检测算法Viola-Jones



内容回顾：深度学习早期人脸检测算法

CascadeCNN → +关键点分支 → MTCNN → +角度预测分支 → PCN

滑窗	滑窗操作遍历所有的位置
金字塔	图像金字塔遍历所有的大小
级联	3个级联的阶段，人脸数量从多到少，人脸难度从易到难，网络结构从简单到复杂
深度学习	利用深度学习中的神经网络进行特征提取+分类器+矫正器+其他
优点	有着满足实际需求的精度，具备CPU实时的速度
缺点	局部最优：三个独立的阶段，容易取得局部最优而非全局最优 训练繁琐：训练不是端到端的，每个阶段单独处理，非常繁琐 检测速度：检测速度与图像上人脸的数量高度相关 检测精度：在复杂的场景下，检测精度不能够满足性能的需求
总结	深度学习早期人脸检测算法的代表，开创了深度学习时代下，人脸检测的一个派系，很多实际场景中都在使用该类型的算法

内容回顾：深度学习早期人脸检测算法

CascadeCNN → +关键点分支 → MTCNN → +角度预测分支 → PCN

滑窗	滑窗操作遍历所有的位置
金字塔	图像金字塔遍历所有的大小
级联	3个级联的阶段，人脸数量从多到少，人脸难度从易到难，网络结构从简单到复杂
深度学习	利用深度学习中的神经网络进行特征提取+分类器+矫正器+其他
优点	有着满足实际需求的精度，具备CPU实时的速度
缺点	局部最优：三个独立的阶段，容易取得局部最优而非全局最优 训练繁琐：训练不是端到端的，每个阶段单独处理，非常繁琐 检测速度：检测速度与图像上人脸的数量高度相关 检测精度：在复杂的场景下，检测精度不能够满足性能的需求
总结	深度学习早期人脸检测算法的代表，开创了深度学习时代下，人脸检测的一个派系，很多实际场景中都在使用该类型的算法

内容回顾：深度学习后期人脸检测算法

- 深度学习后期人脸检测算法：对**通用物体检测算法**进行**相应改进**，应用于**人脸检测领域**

高效率的人脸检测算法

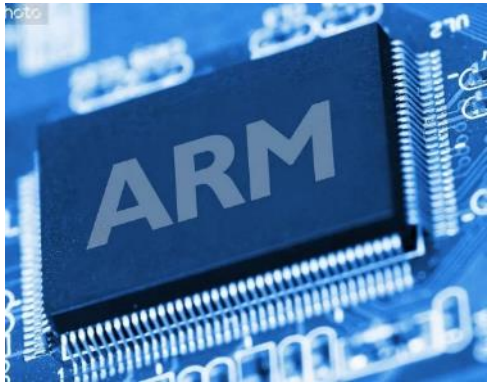
- 基础网络为专门设计的轻量级的网络结构
- 在实际场景中，检测大于30个像素的人脸，有着满足需求的检测精度
- 能够在资源受限的前端设备（CPU、ARM、FPGA等）上实时的运行
- 追求检测速度和检测精度的平衡，满足实用性

高精度的人脸检测算法

- 基础网络为重量级的VGG16或ResNet-50/101/152等
- 在复杂场景下，有着非常高的检测精度，非常小的人脸也能检测
- 可以在高性能的GPU设备上实时的运行
- 追求极致的检测精度，检测速度可以不考虑

内容回顾：深度学习后期人脸检测算法

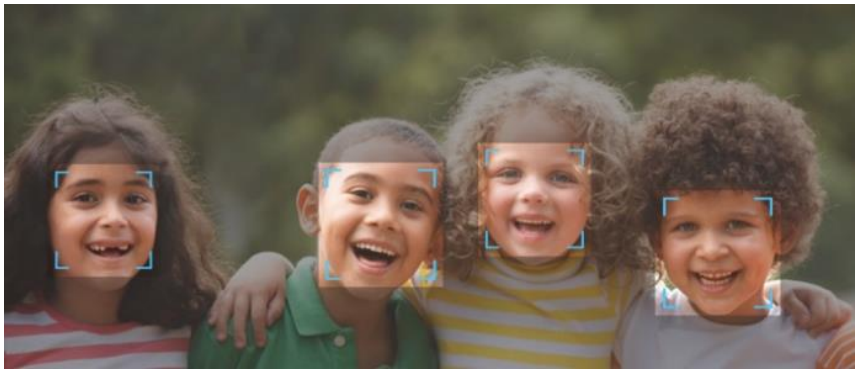
- 高效率算法：运行在**计算资源受限**的前端设备，如CPU、ARM、FPGA



- 高精度算法：运行在**计算资源充足**的后端设备，如英伟达显卡2080Ti、RTX

内容回顾：深度学习后期人脸检测算法

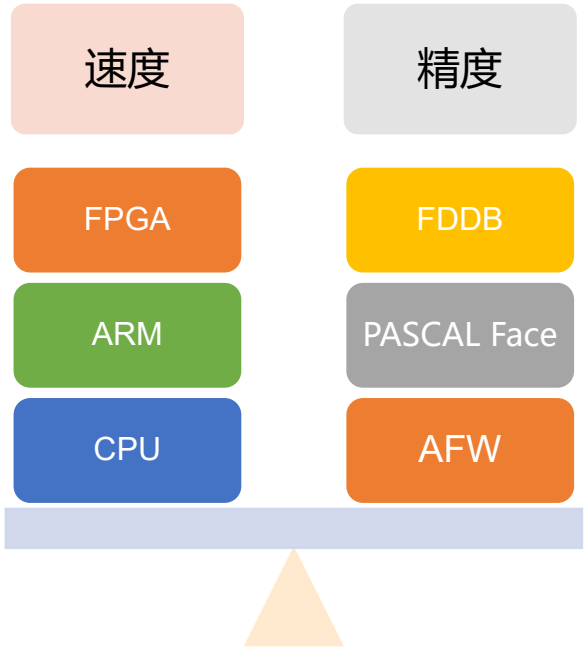
- 高效率算法：由于资源受限，只需要在**正常场景**下满足精度需求（人脸 > 30个像素，背景比较简单）



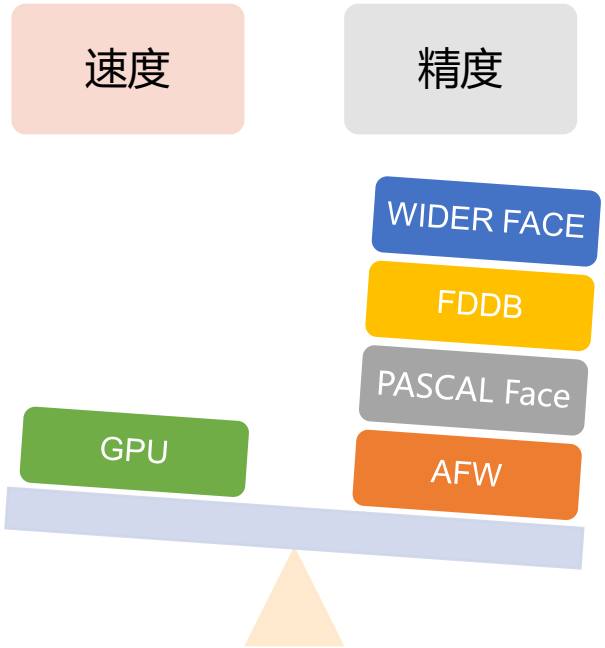
- 高精度算法：由于资源充足，需要在**所有场景**下都满足精度需求（任何尺度人脸，背景非常复杂）

内容回顾：深度学习后期人脸检测算法

高效率算法：实用性



高精度算法：竞争性

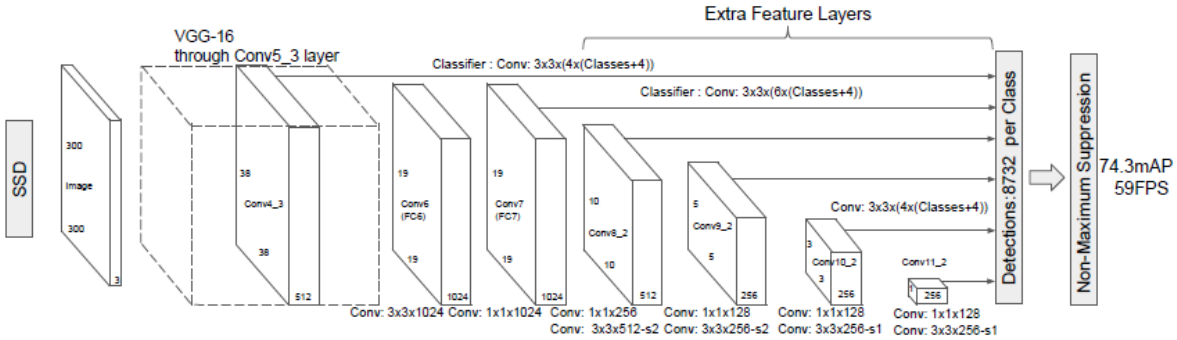


内容回顾：高效率的深度学习后期人脸检测算法

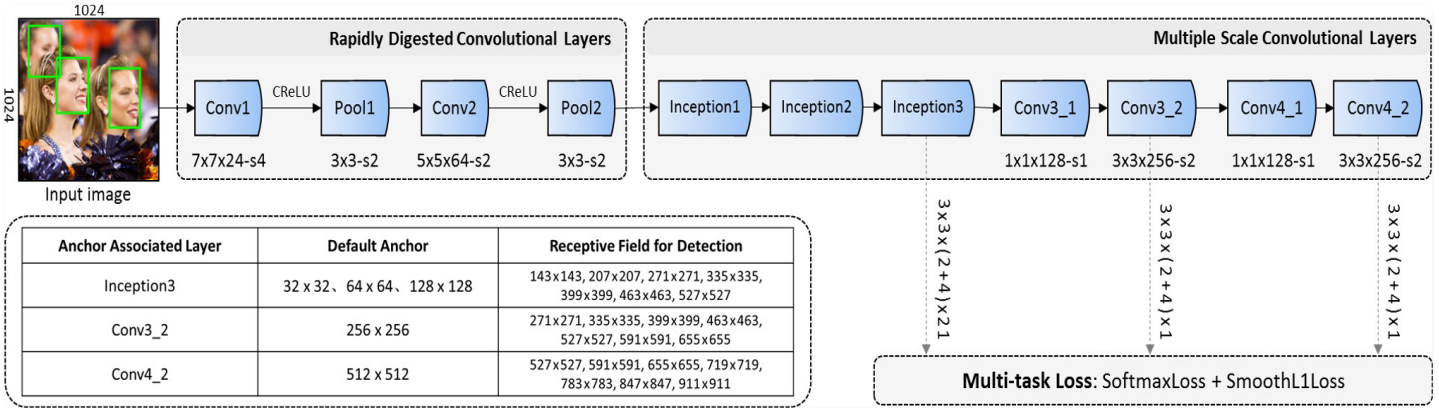
- 高效率的深度学习后期人脸检测算法几乎都是基于通用物体检测算法SSD进行的改进

① 轻量级网络结构

② 锚框密集化操作

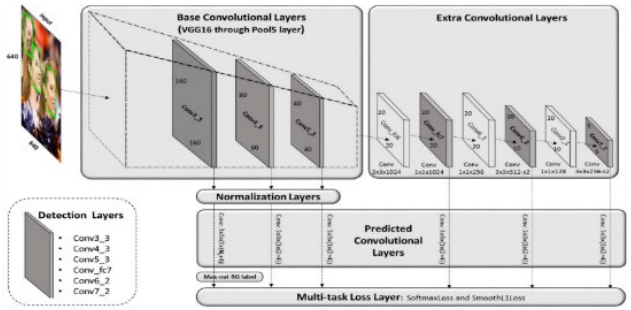


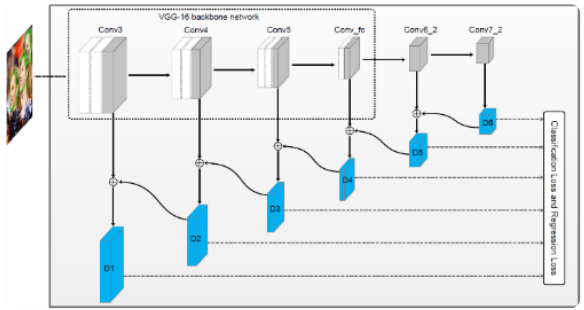
FaceBoxes



Anchor Associated Layer	Default Anchor	Receptive Field for Detection
Inception3	32 x 32、64 x 64、128 x 128	143x143, 207x207, 271x271, 335x335, 399x399, 463x463, 527x527
Conv3_2	256 x 256	271x271, 335x335, 399x399, 463x463, 527x527, 591x591, 655x655
Conv4_2	512 x 512	527x527, 591x591, 655x655, 719x719, 783x783, 847x847, 911x911

内容回顾：高精度的深度学习后期人脸检测算法



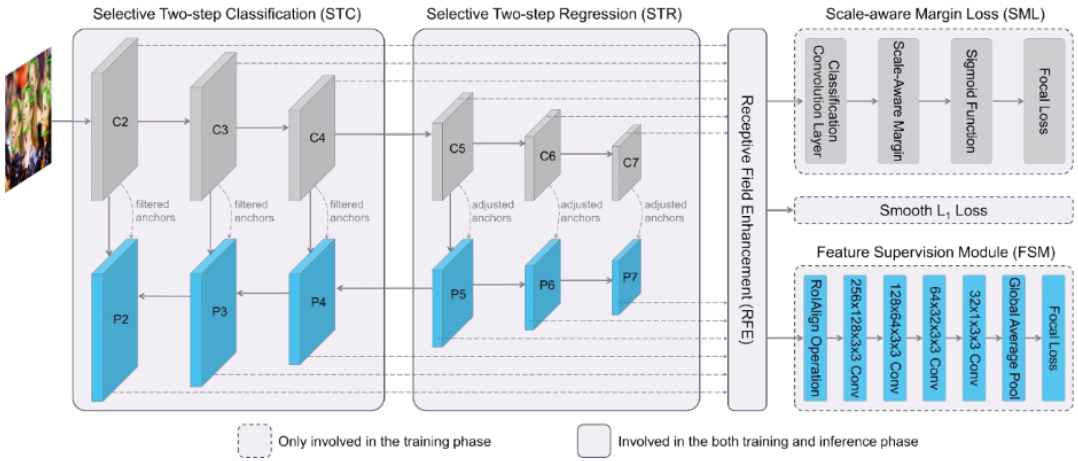


S³FD & SFDet

① 尺度上公平的检测框架

② 尺度补偿的锚框匹配策略

③ 背景标签输出最大化操作



SRN & RefineFace

① 选择性二阶段分类

② 选择性二阶段回归

③ 感受野增强模块

④ 尺度敏感的margin损失函数

⑤ 特征监督模块

目录

目录

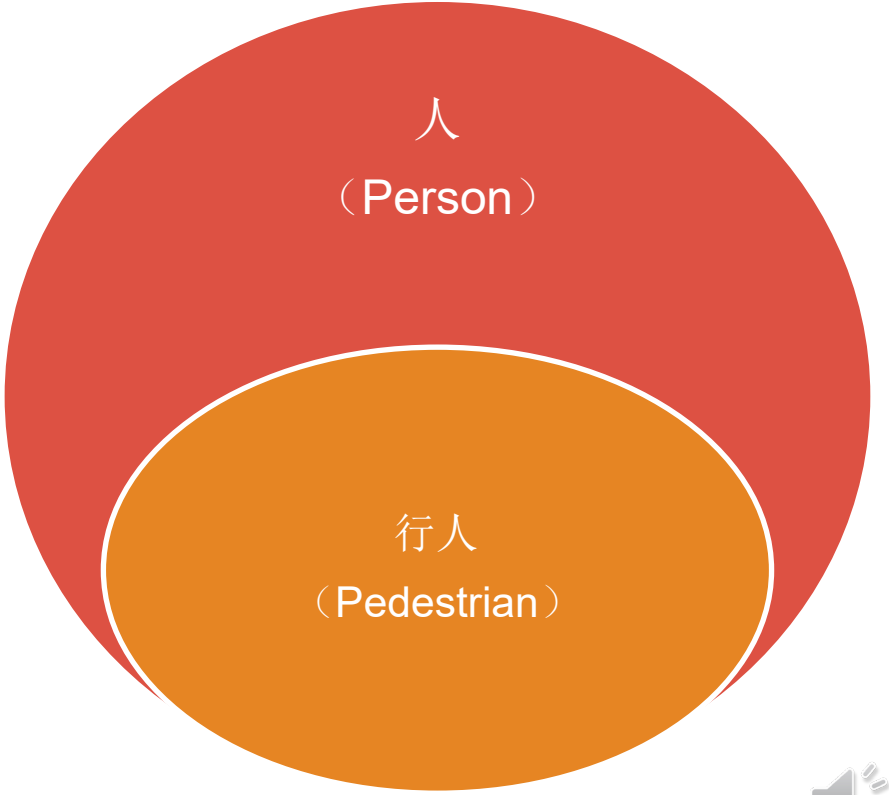
人 (Person) 和行人 (Pedestrian)

■ 人（Person）

- A person is an individual, usually a human being
- 各种各样的人

■ 行人（Pedestrian）

- A pedestrian is a person travelling on foot, whether walking or running
- 行人是步行的人，无论是走着还是跑着



行人 (Pedestrian)

- 行人 (Pedestrian)：步行的人，无论是走着还是跑着

其他类型的人

- 骑行者

其他类型的人

- 坐着或躺着的人

其他类型的人

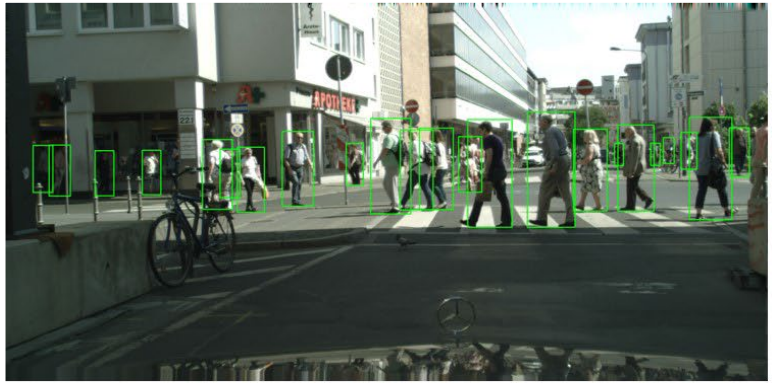
- 姿态诡异的人

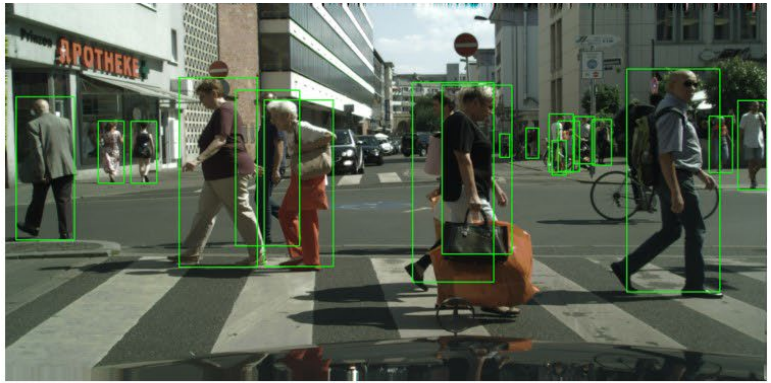
其他类型的人

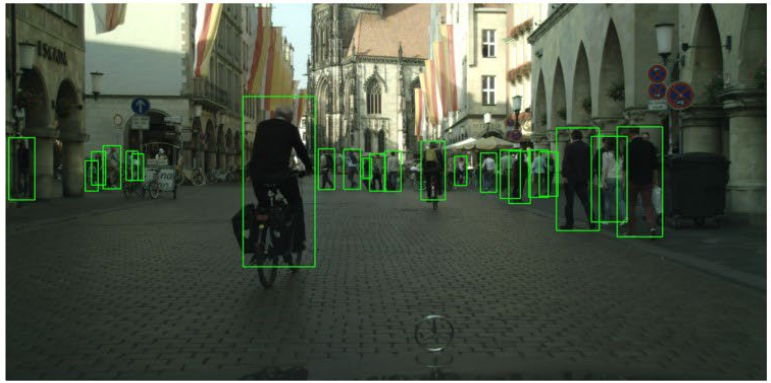
- 人群

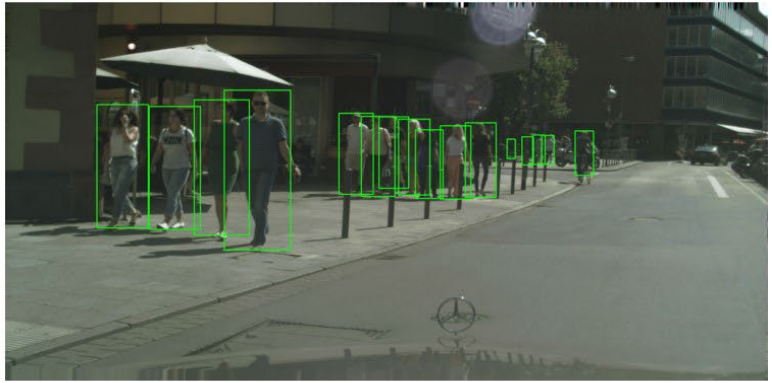
行人检测的定义

- 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





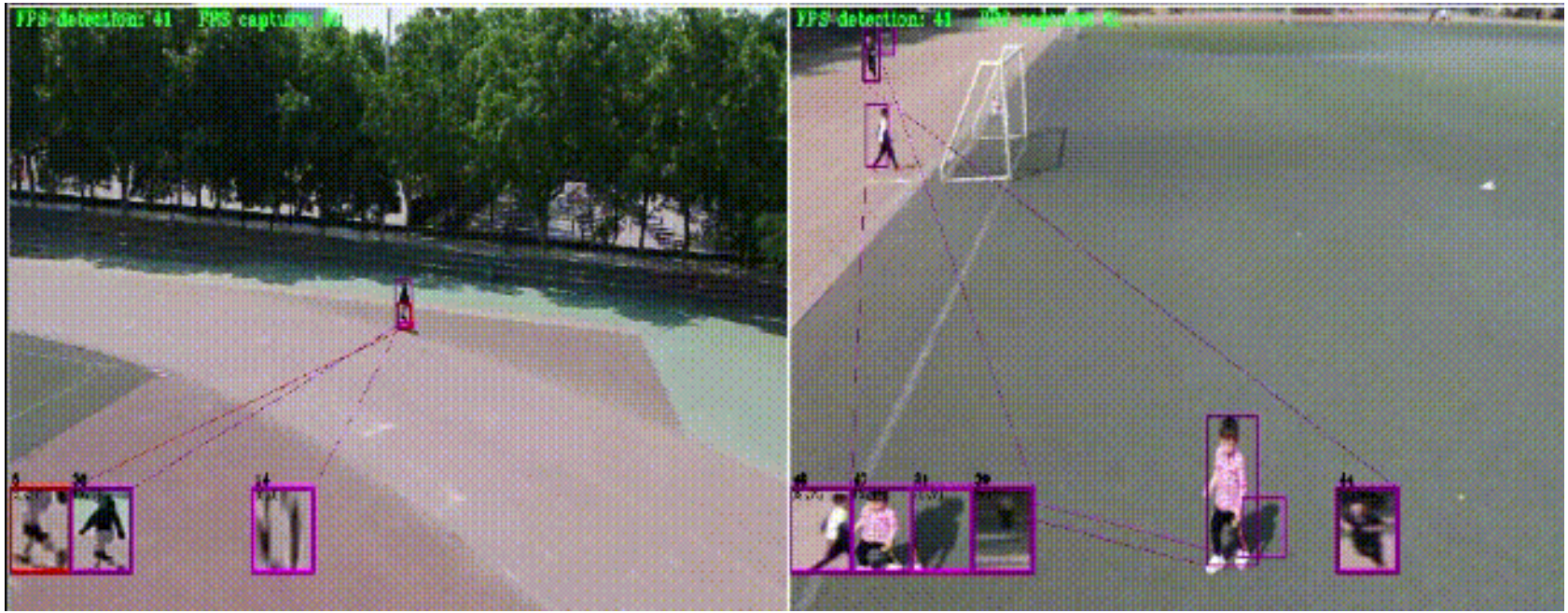




行人检测的应用

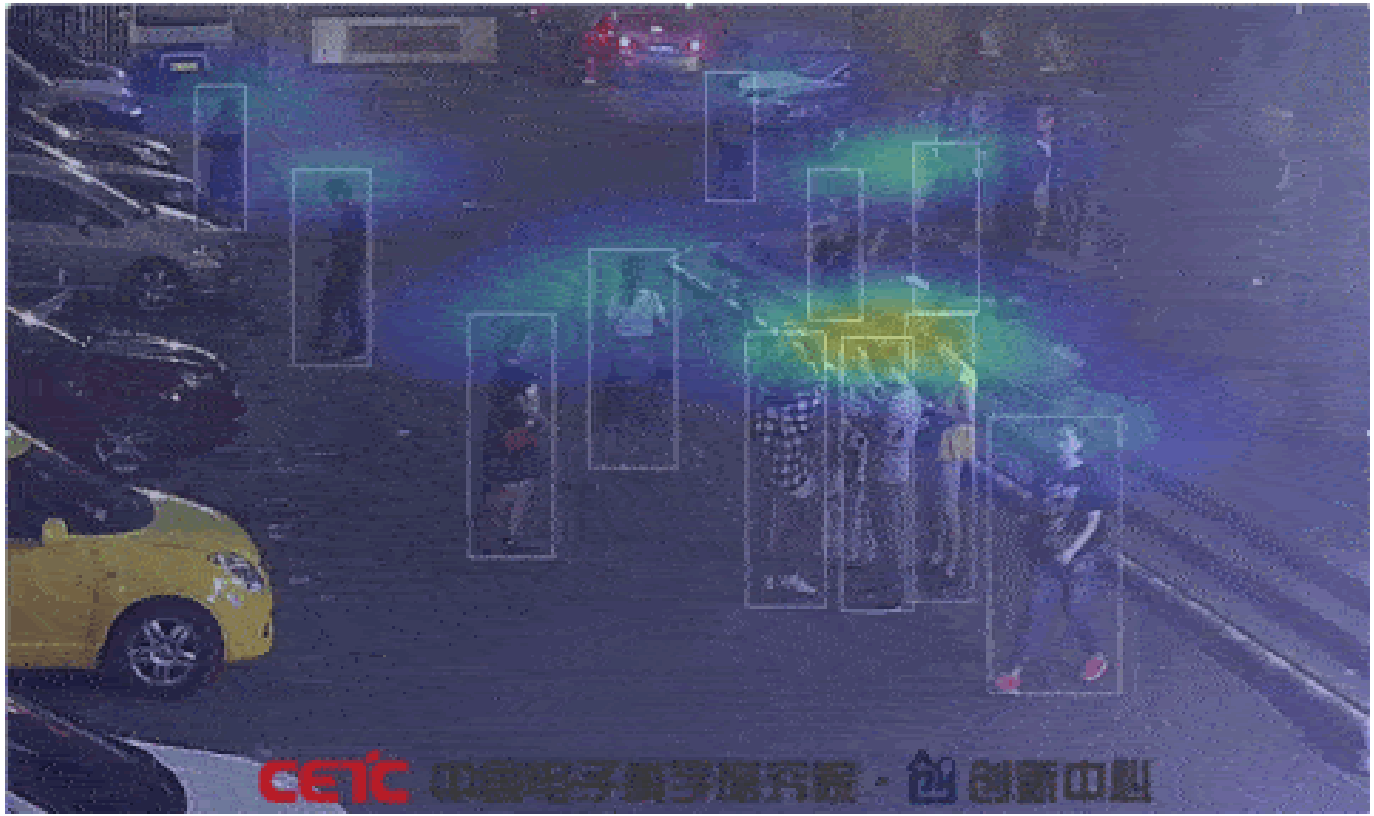
交通路口监控系统

行人检测的应用



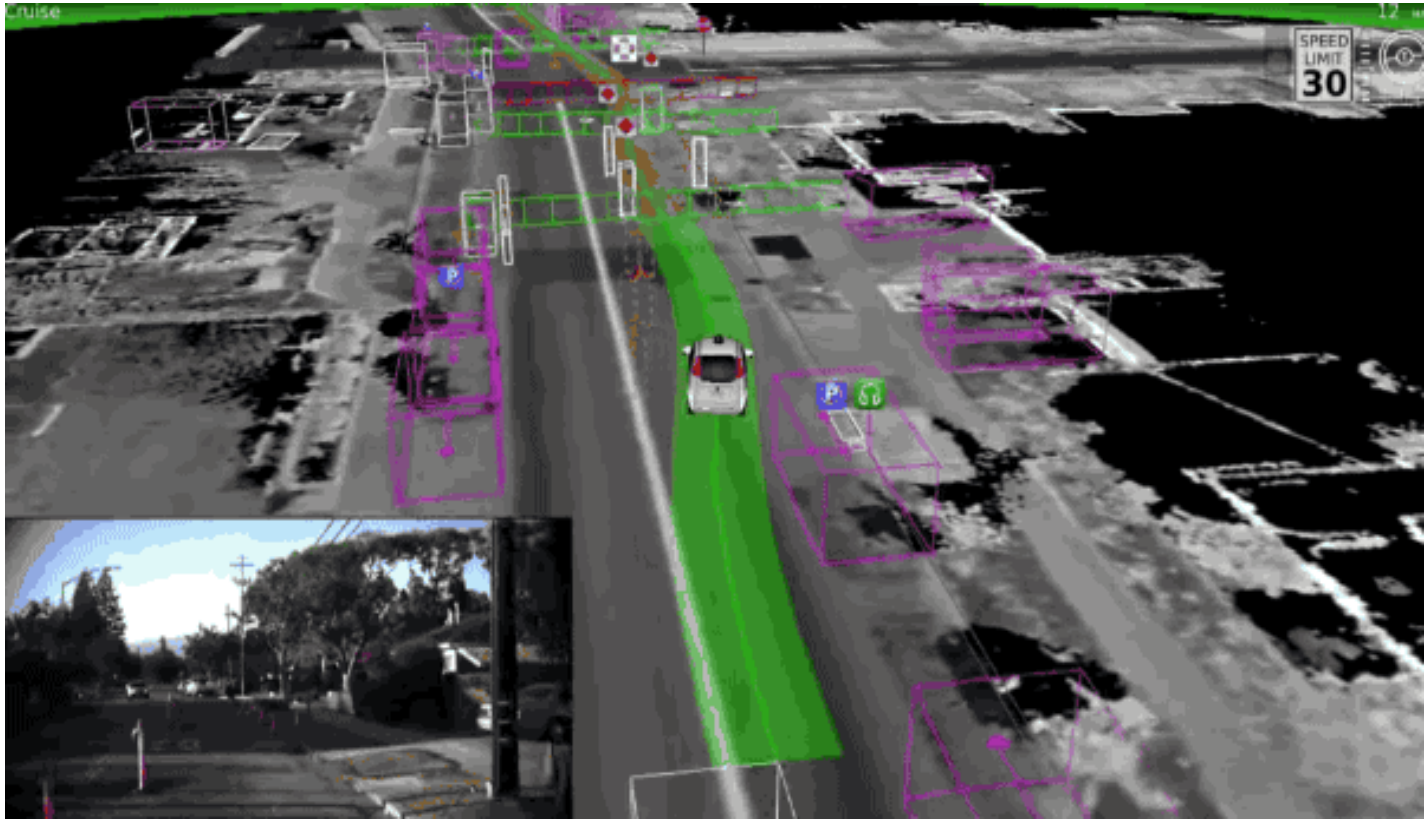
校园监控系统

行人检测的应用



视频监控系统

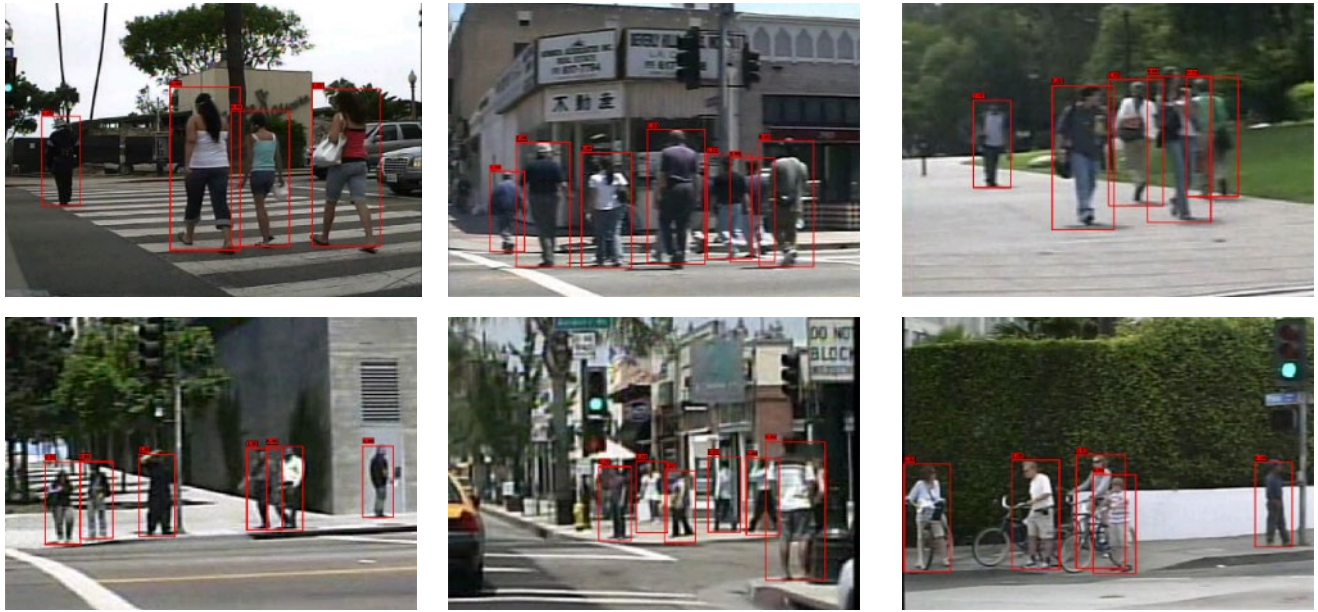
行人检测的应用



自动驾驶

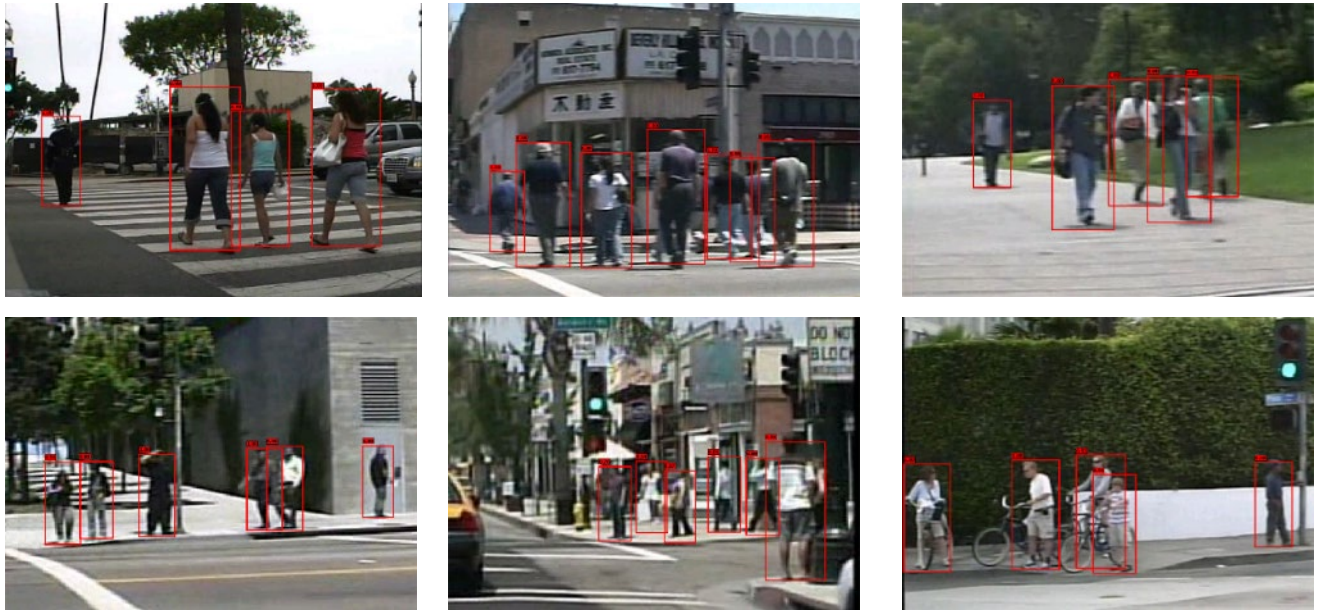
行人检测数据集：Caltech-USA

- 10小时30帧的分辨率为640x480的VGA视频
- **42782**张训练集，**4024**张测试集



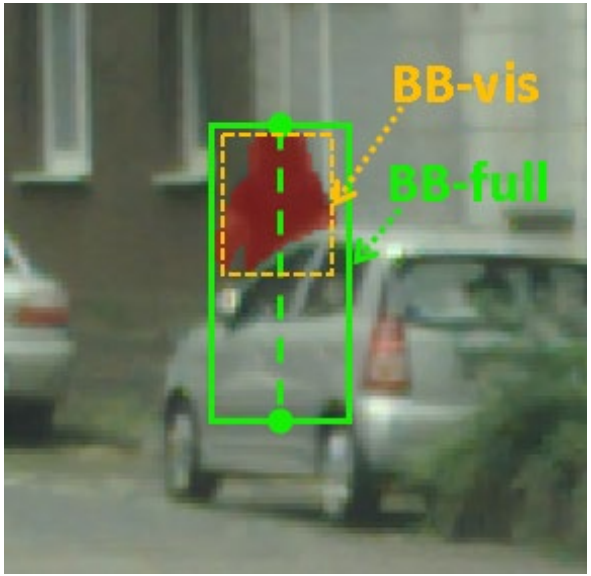
行人检测数据集：Caltech-USA

- 10小时30帧的分辨率为640x480的VGA视频
- **42782**张训练集，**4024**张测试集



行人检测数据集：CityPersons

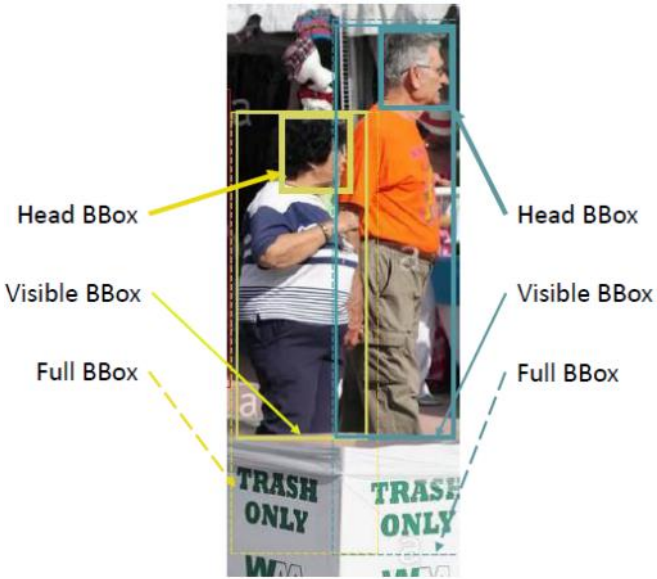
- **5000**张图：2975张训练，500张验证，1525张测试
- **35000**+行人（完整、可见区域），13000+忽略区域



- Pedestrian (walking, running or standing up)
- rider (riding bicycles or motorbikes)
- sitting person
- other person (with unusual postures)
- Ignore region

行人检测数据集：CrowdHuman

- **24370**张图：训练集15000，验证集4370，测试集5000
- **47万**行人标注，平均每张图23个行人
- Head、human visible-region、human full-body

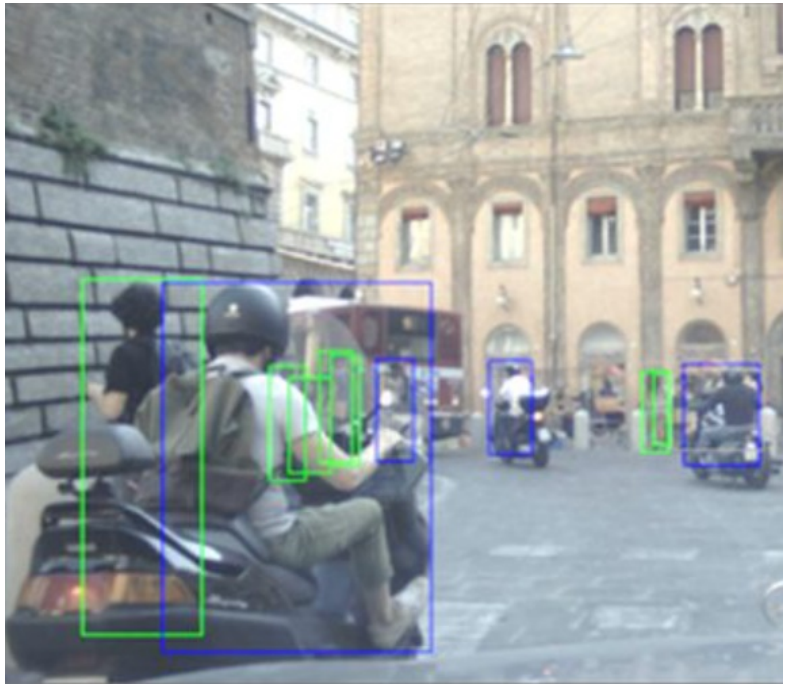


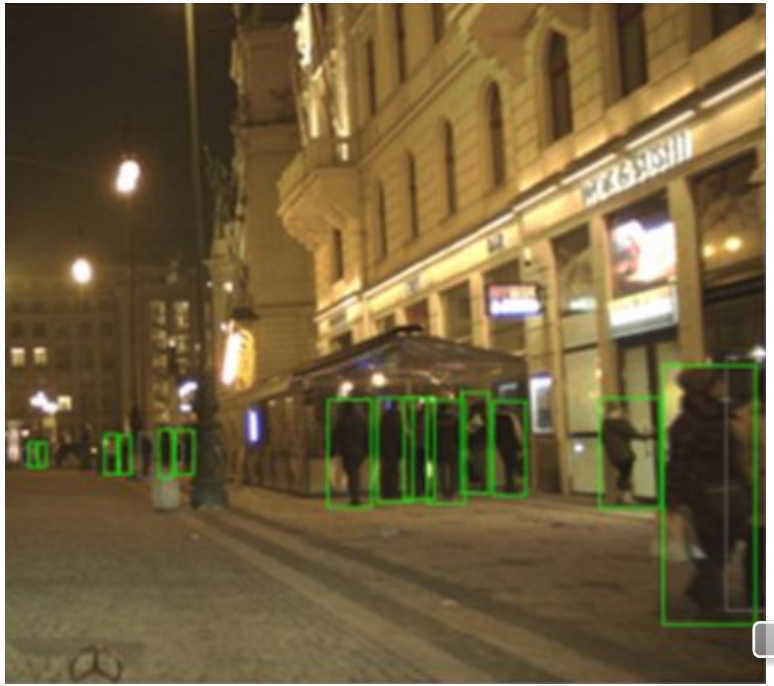
行人检测数据集：WiderPerson

- **13382**张图，约**40万**个标注，丰富的场景
- 训练集8000 张、验证集1000 张、测试集4382 张

行人检测数据集：EuroCityPersons

- CityPersons 强化版：**47325** 张图像，**23.8万**+个标注（行人、骑行者以及忽略区域）
- 该数据不仅包含白天的图像，还有夜晚的图像





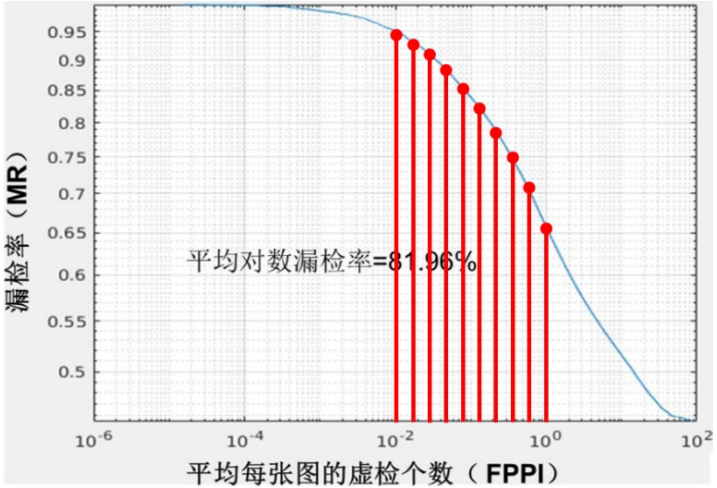
行人检测数据集

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	★★★★

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	★★★★

行人检测评价指标：精度

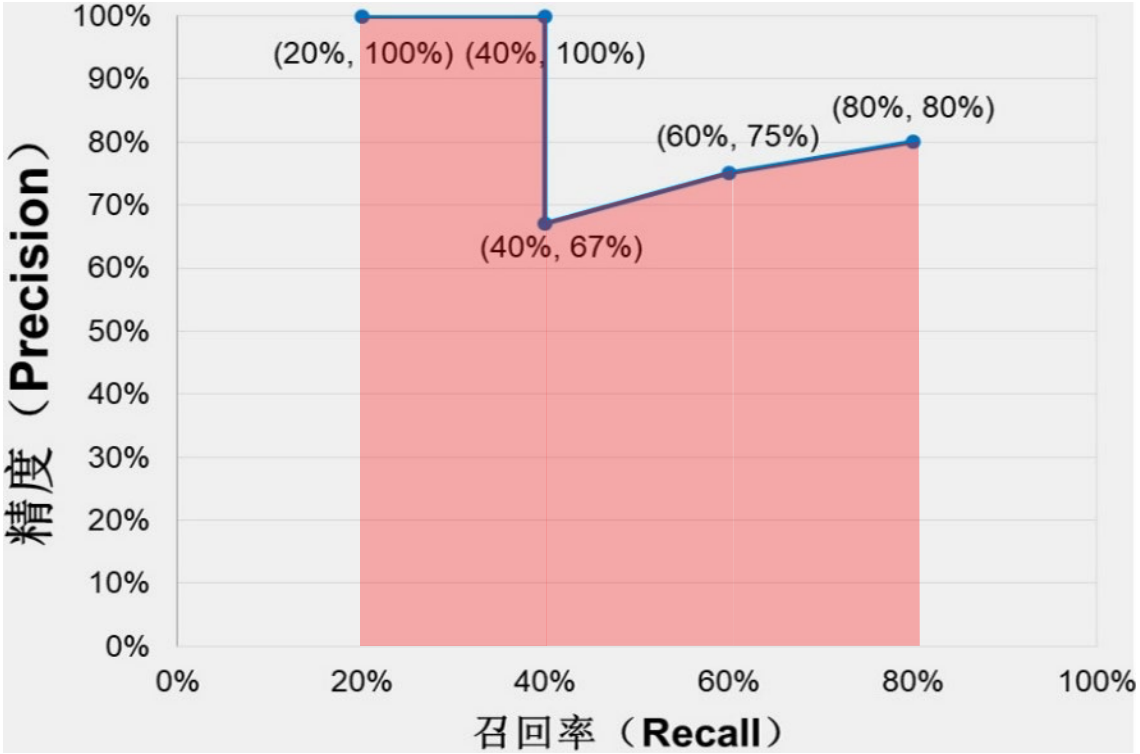


平均对数漏检率（MR^{-2}）

$$= e^{\left(\frac{1}{9}\sum_{i=1}^{9} ln(MR_i)\right)}$$

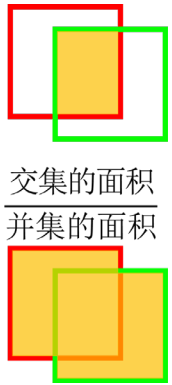
FPPI	10^{-2}	$10^{-1.75}$	$10^{-1.5}$	$10^{-1.25}$	10^{-1}	$10^{-0.75}$	$10^{-0.5}$	$10^{-0.25}$	10^{0}
MR	94.8%	93.1%	91.8%	88.2%	85.6%	78.2%	74.9%	71.1%	66.0%
ln (MR)	-0.0534	-0.0715	-0.0856	-0.1256	-0.1555	-0.2459	-0.2890	-0.3411	-0.4155
平均值	-0.1981								
MR^{-2}	**81.96%**								

行人检测评价指标：精度



平均精度（AP）

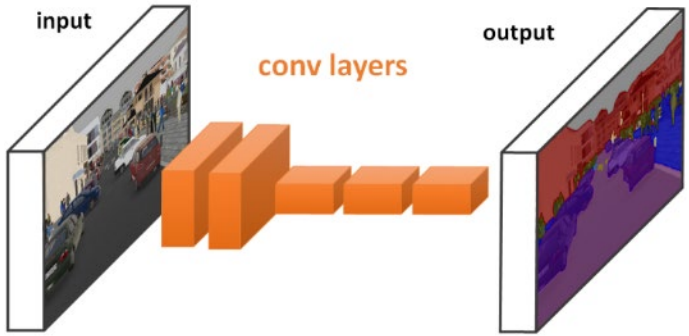
精度-召回率曲线下的面积



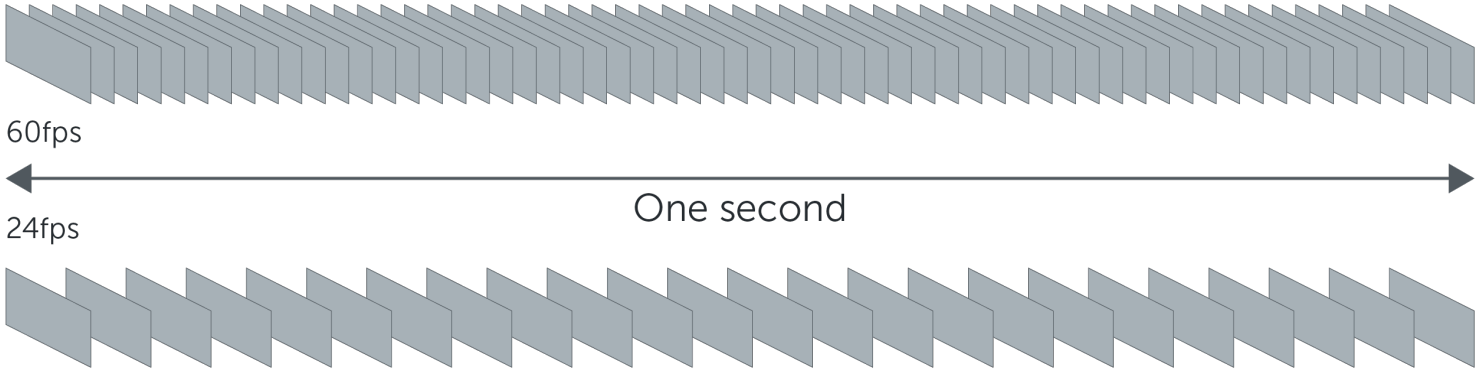
$$\text{交除并重叠比（}IoU\text{）} = \frac{\text{交集的面积}}{\text{并集的面积}} \geq 0.5$$

行人检测评价指标：速度

- **前传耗时 (ms)**：从输入一张图像到输出最终结果所消耗的时间，这包括前处理耗时 (如图像归一化)、网络前传耗时、后处理耗时(如非极大值抑制)



- **每秒帧数 (FPS)**：每秒钟能处理的图像数量

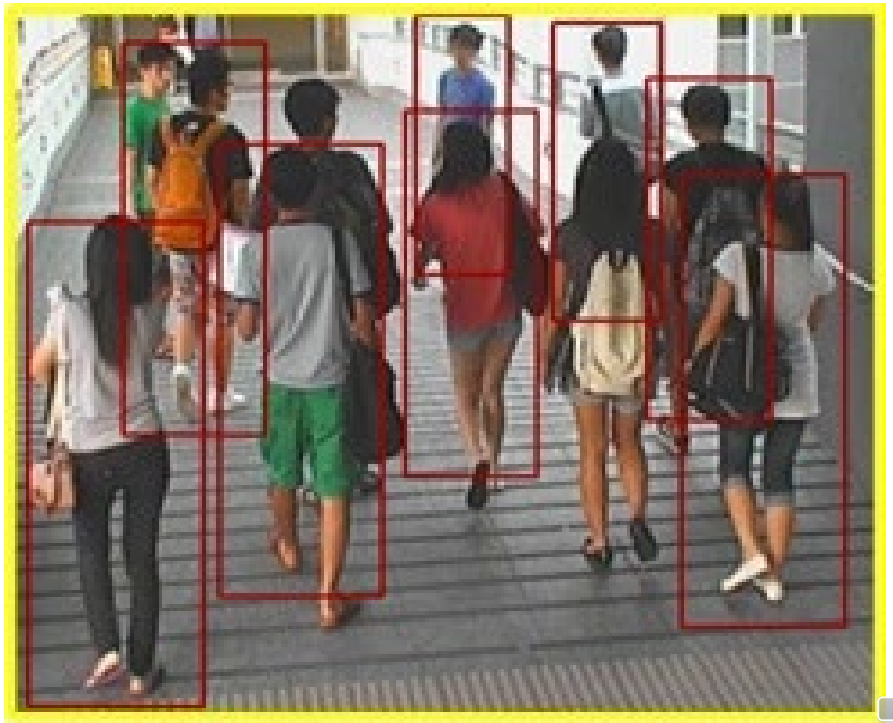


行人检测的难点

- 行人是可形变的（Deformable）

行人检测的难点

- 行人存在着各种各样的遮挡（occlusion）



行人检测的难点

- 行人检测中存在很多难正样本和难负样本

难正样本　　难负样本

行人检测的发展脉络

传统方法
- Dalal行人检测算法

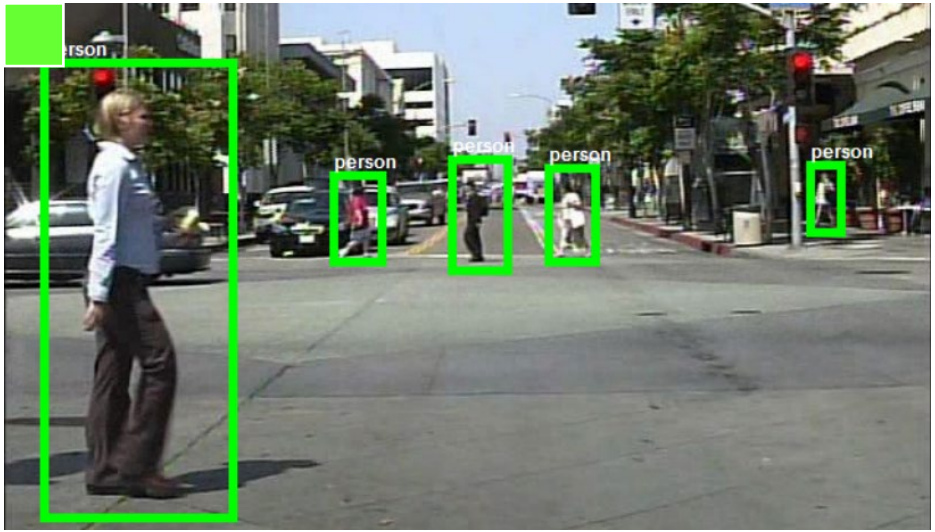
深度学习早期
- RPN+BF

深度学习后期
- Faster R-CNN, SSD

目录

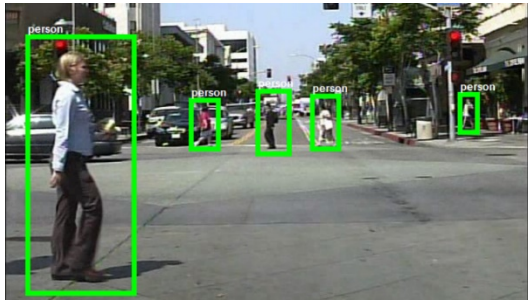
传统行人检测算法

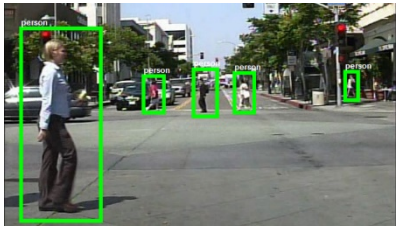
- 利用手工特征+分类器，以滑窗方式在图像金字塔上遍历所有位置和大小，进行行人检测

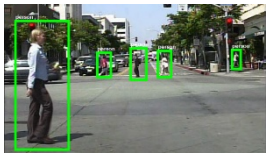


滑窗方式

遍历所有位置







图像金字塔遍历不同大小

传统行人检测算法

- 利用手工特征+分类器，以滑窗方式在图像金字塔上遍历所有位置和大小，进行行人检测

手工特征

手工分类器

传统行人检测算法

- 利用手工特征+分类器，以滑窗方式在图像金字塔上遍历所有位置和大小，进行行人检测

手工特征 → 手工分类器

- Dalal行人检测算法

检测机制	手工特征	手工分类器
• 滑窗 + 图像金字塔	• HOG特征	• SVM分类器

Histogram of oriented gradients for human detection, CVPR 2005

Dalal行人检测算法：整体检测流程

输入图像

Dalal行人检测算法：整体检测流程

图像
缩放

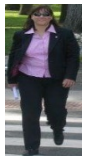
输入图像

图像金字塔

Dalal行人检测算法：整体检测流程

图像
缩放

滑窗
128x64

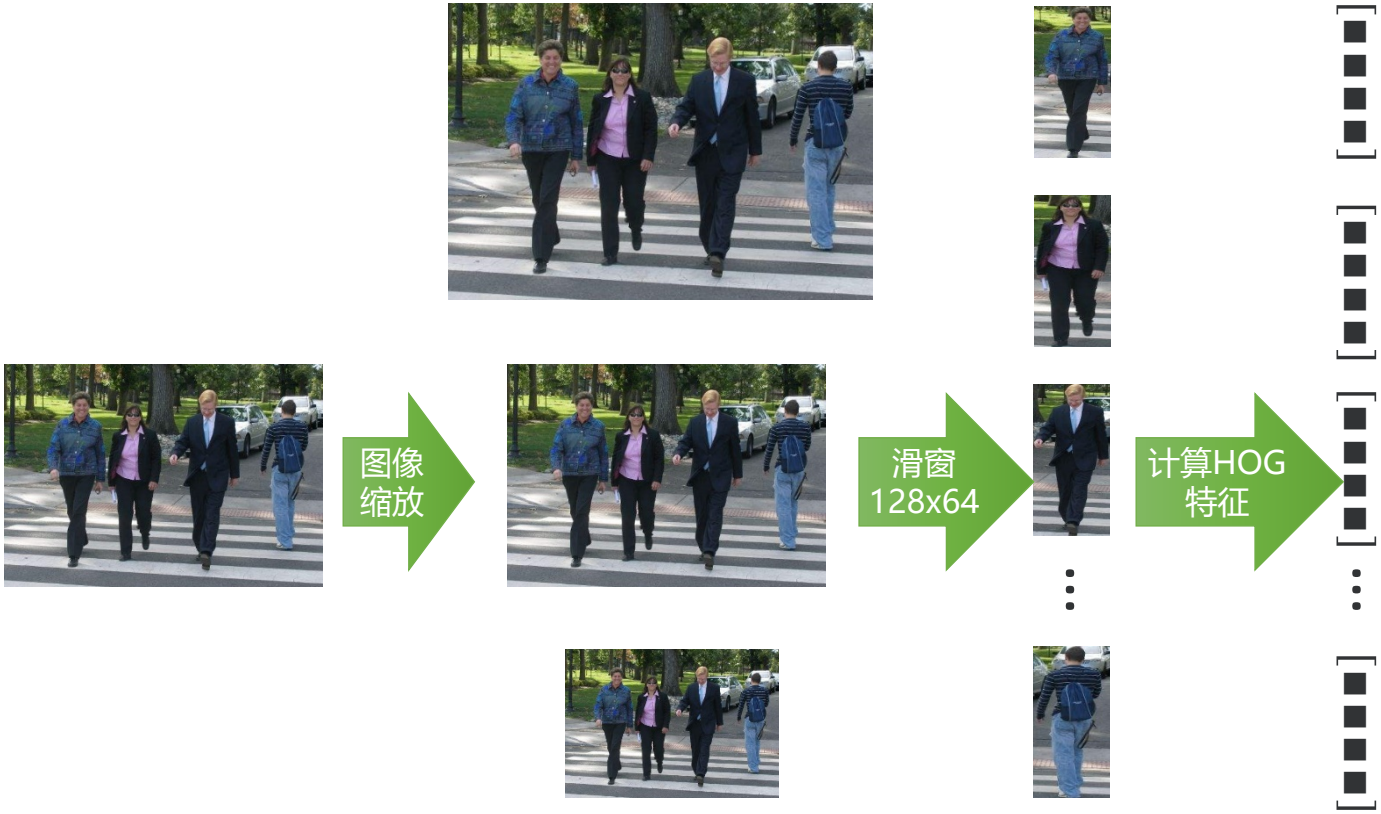


输入图像

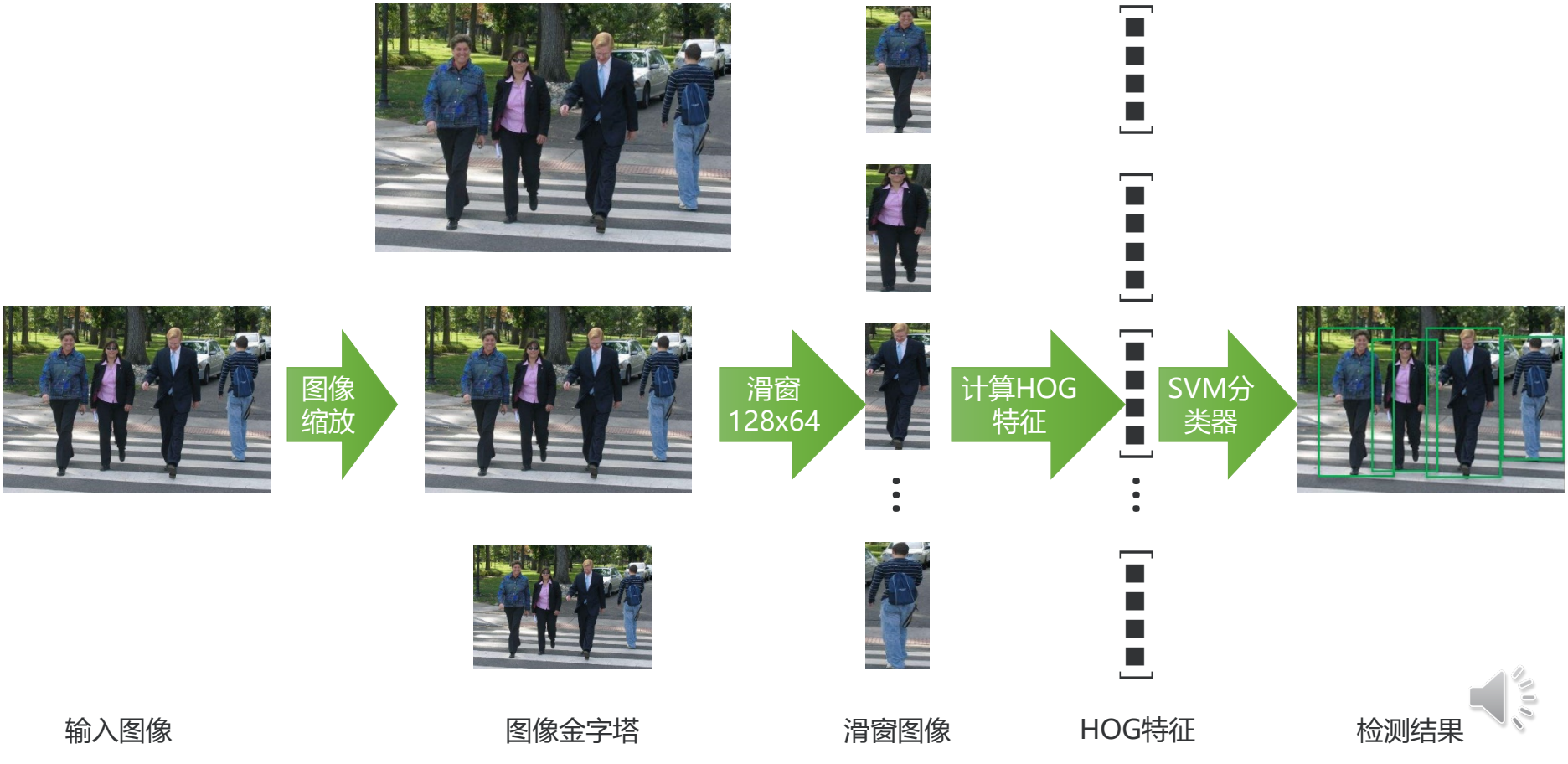
图像金字塔

滑窗图像

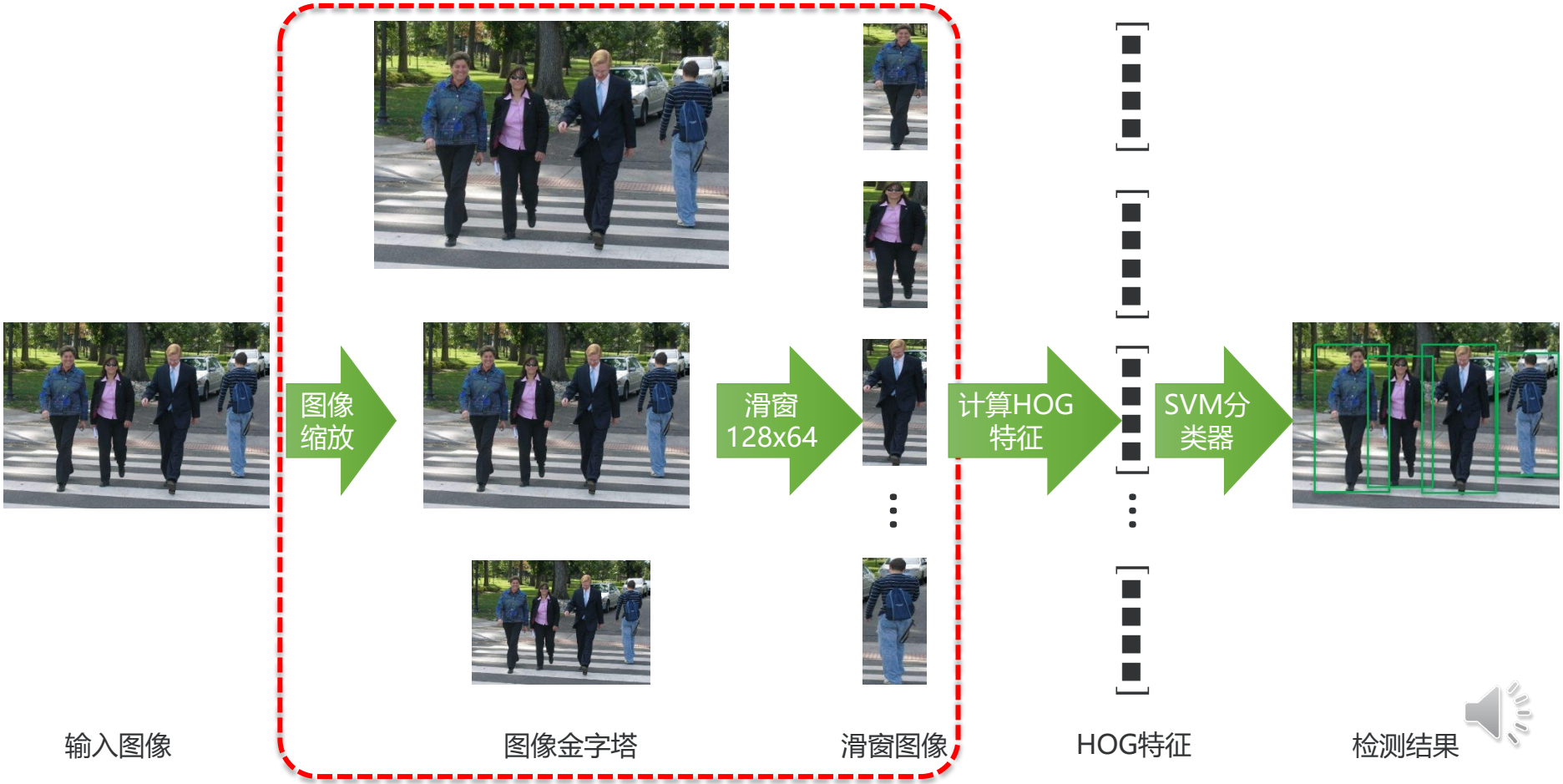
Dalal行人检测算法：整体检测流程



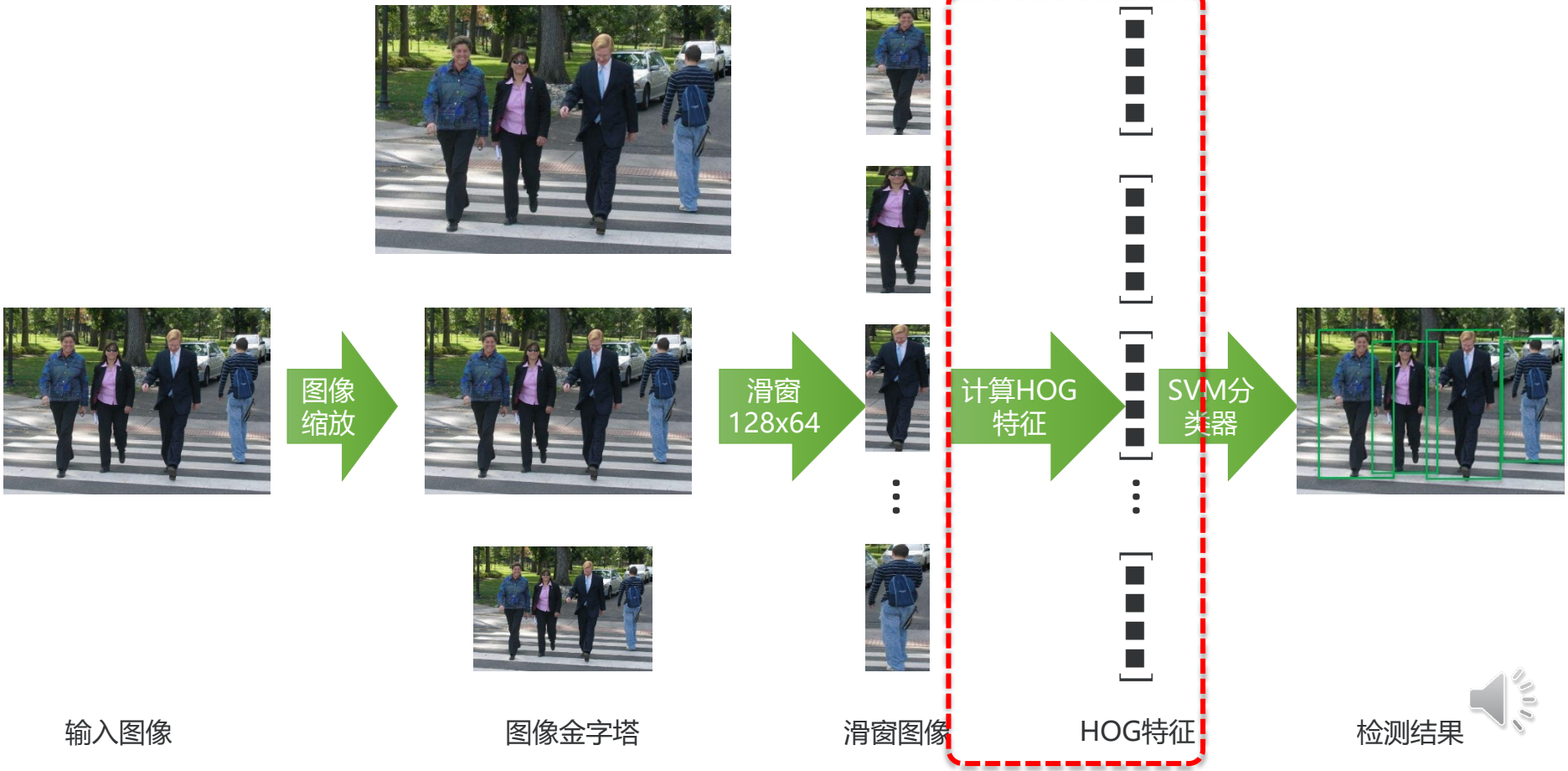
Dalal行人检测算法：整体检测流程



Dalal行人检测算法：整体检测流程



Dalal行人检测算法：整体检测流程



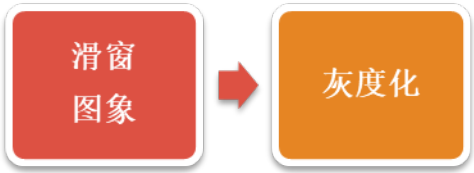
Dalal行人检测算法：HOG特征

- 滑窗图像：经过滑窗 + 图像金字塔得到的128x64子图像

滑窗
图象

Dalal行人检测算法：HOG特征

- 灰度化：RGB -> Gray的转化公式为：$Gray = 0.3 * R + 0.59 * G + 0.11 * B$



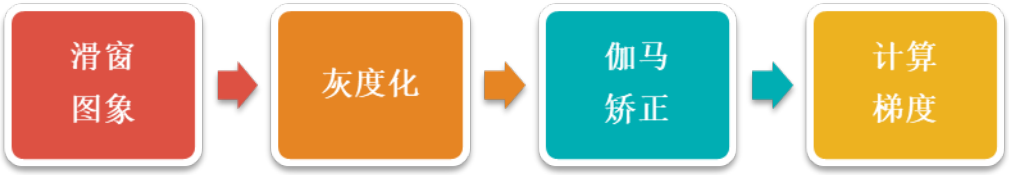
Dalal行人检测算法：HOG特征

- 伽马矫正：在图像亮度不均匀的情况下，将图像整体亮度提高或降低，公式如下（其中γ=0.5）：

$$Y(x,y) = I(x,y)^{\gamma}$$

Dalal行人检测算法：HOG特征

- 计算梯度：对经过颜色归一化的图像，首先计算水平方向和垂直方向的梯度，再计算最终梯度的幅值和方向

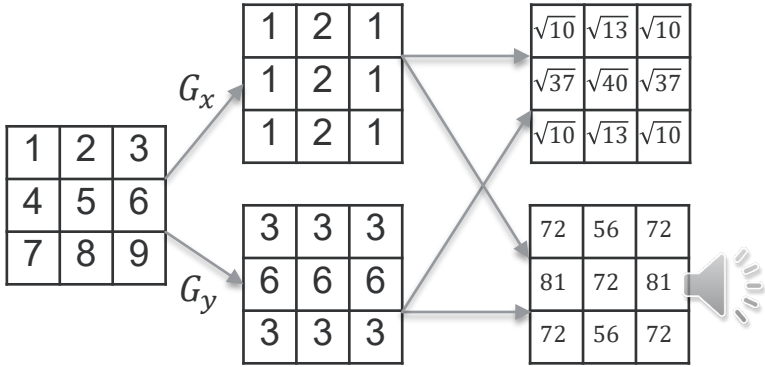


$$G_x(x,y) = I(x+1,y) - I(x-1,y)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1)$$

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

$$\theta(x,y) = arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right)$$

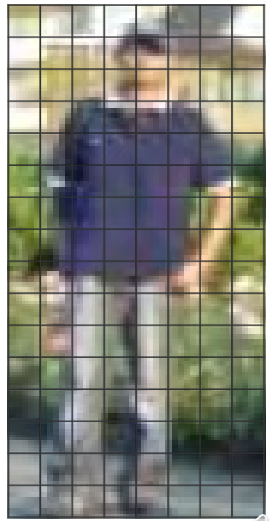


Dalal行人检测算法：HOG特征

- 统计梯度：根据梯度幅值和梯度方向，统计得到梯度直方图

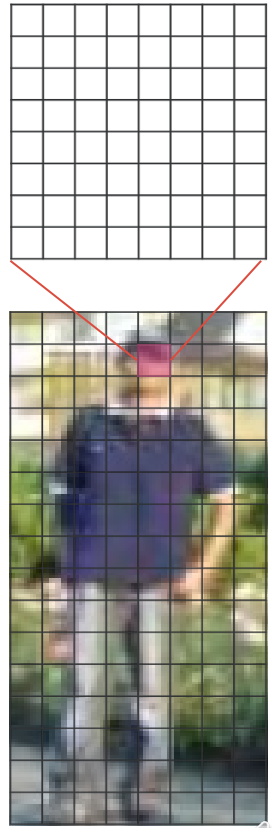
HOG特征：统计梯度

- 将128x64的图像划分成16x8个互不重叠的cell (单元)



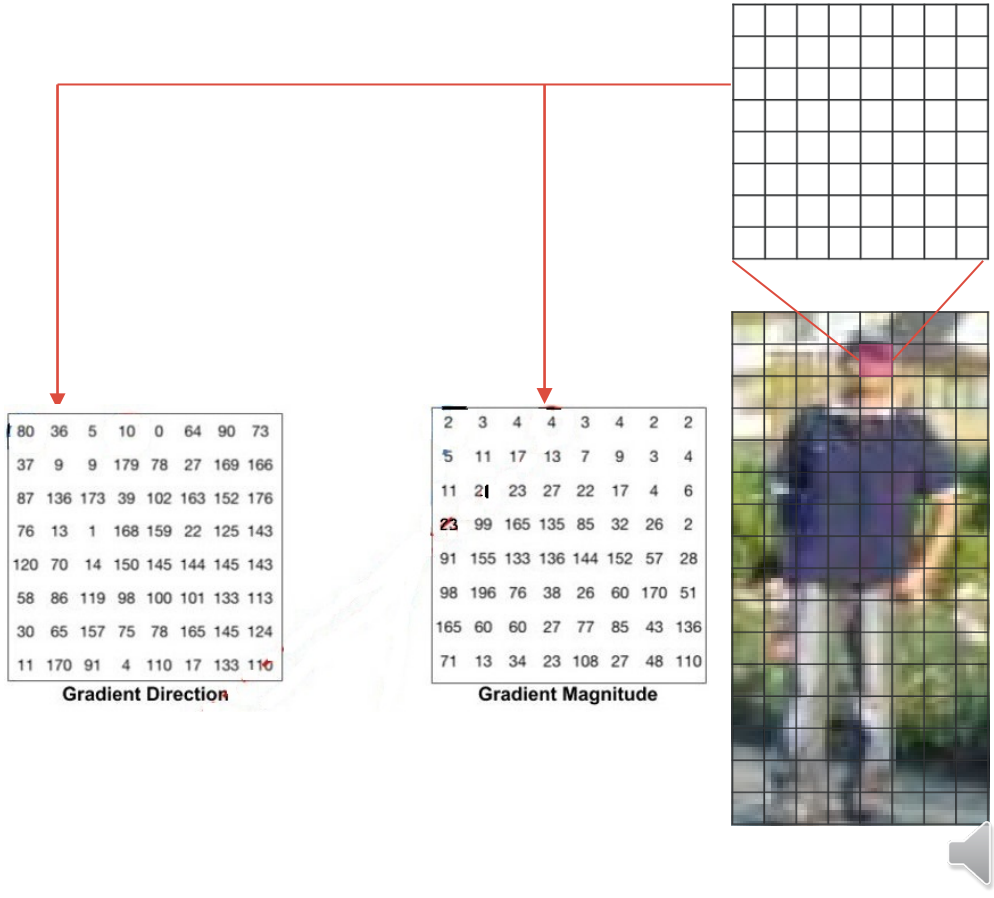
HOG特征：统计梯度

- 将128x64的图像划分成16x8个互不重叠的cell (单元)
- 每个cell大小为8x8，对应8x8的梯度幅值和8x8的梯度方向



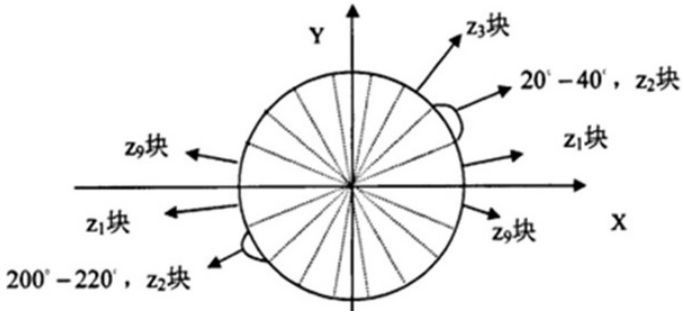
HOG特征：统计梯度

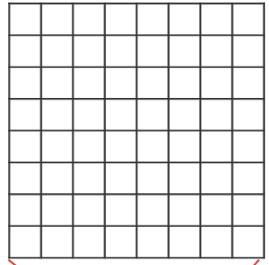
- 将128x64的图像划分成16x8个互不重叠的cell (单元)
- 每个cell大小为8x8，对应8x8的梯度幅值和8x8的梯度方向

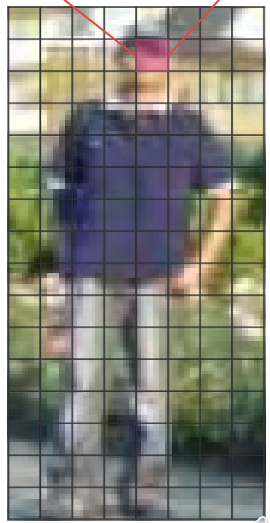


HOG特征：统计梯度

- 将128x64的图像划分成16x8个互不重叠的cell (单元)
- 每个cell大小为8x8，对应8x8的梯度幅值和8x8的梯度方向
- 梯度方向所在范围为0~360度，将0~360度划分为9块，每块的角度范围如图所示







80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

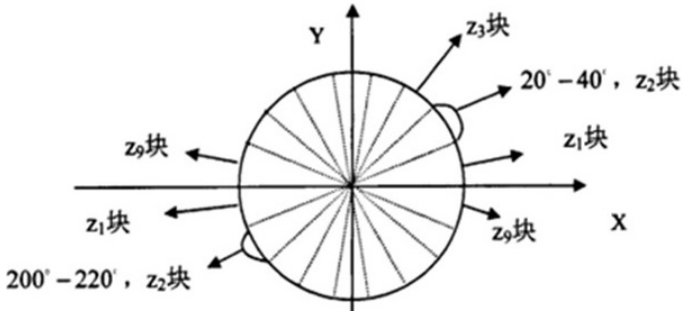
Gradient Direction

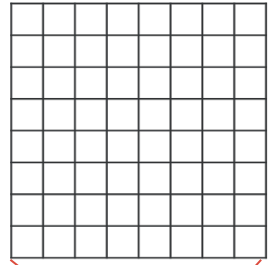
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

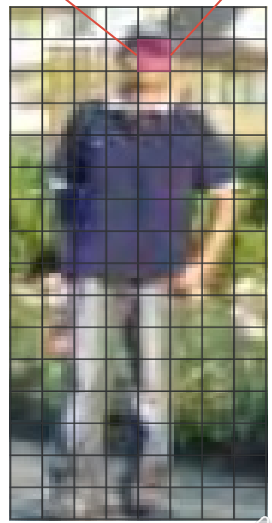
Gradient Magnitude

HOG特征：统计梯度

- 将128x64的图像划分成16x8个互不重叠的cell (单元)
- 每个cell大小为8x8，对应8x8的梯度幅值和8x8的梯度方向
- 梯度方向所在范围为0~360度，将0~360度划分为9块，每块的角度范围如图所示
- 每个cell内统计梯度得到梯度方向直方图，横轴为9块角度范围，纵轴为所对应的梯度值累加值







80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

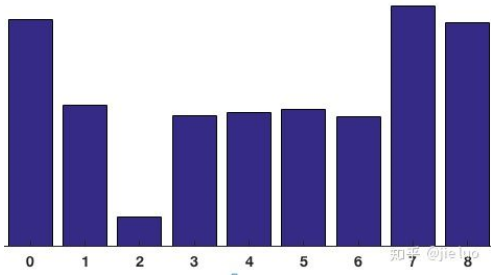
Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

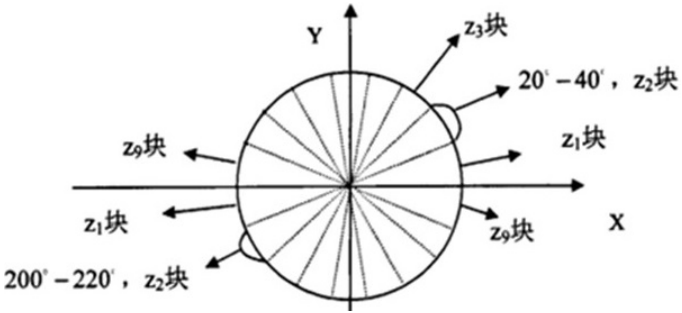
Gradient Magnitude

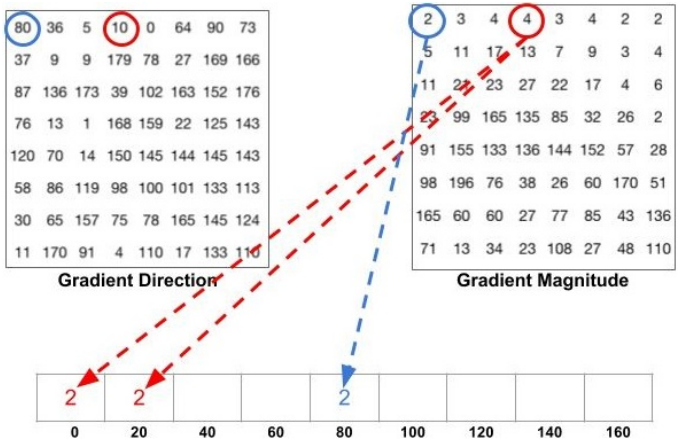
HOG特征：统计梯度

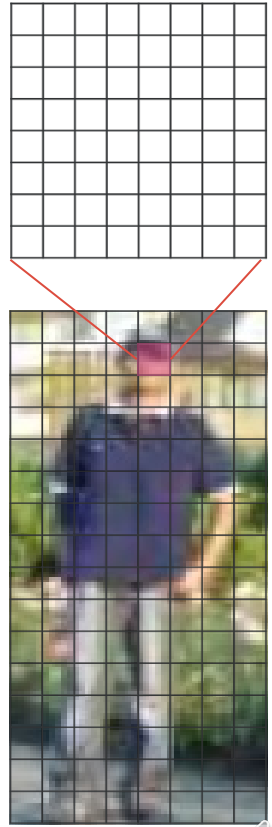
- 将128x64的图像划分成16x8个互不重叠的cell (单元)
- 每个cell大小为8x8，对应8x8的梯度幅值和8x8的梯度方向
- 梯度方向所在范围为0~360度，将0~360度划分为9块，每块的角度范围如图所示
- 每个cell内统计梯度得到梯度方向直方图，横轴为9块角度范围，纵轴为所对应的梯度值累加值



统计直方图

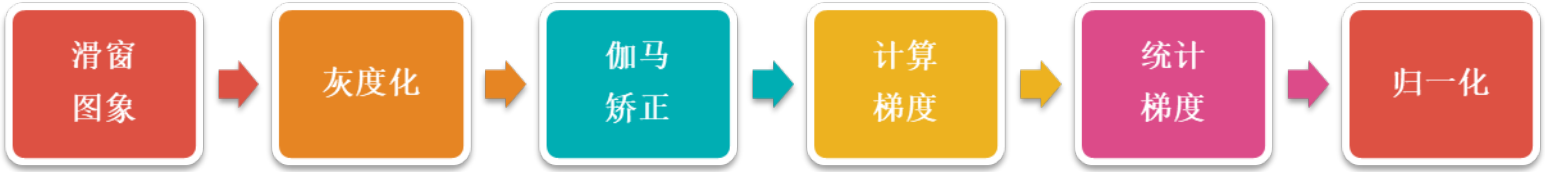


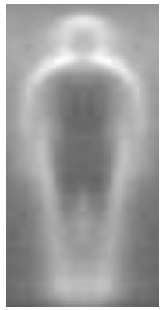


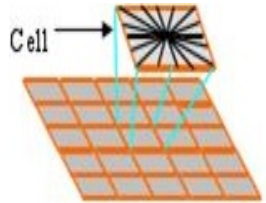


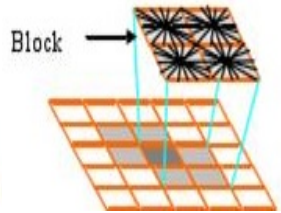
Dalal行人检测算法：HOG特征

- 特征归一化：对得到的特征进行归一化处理

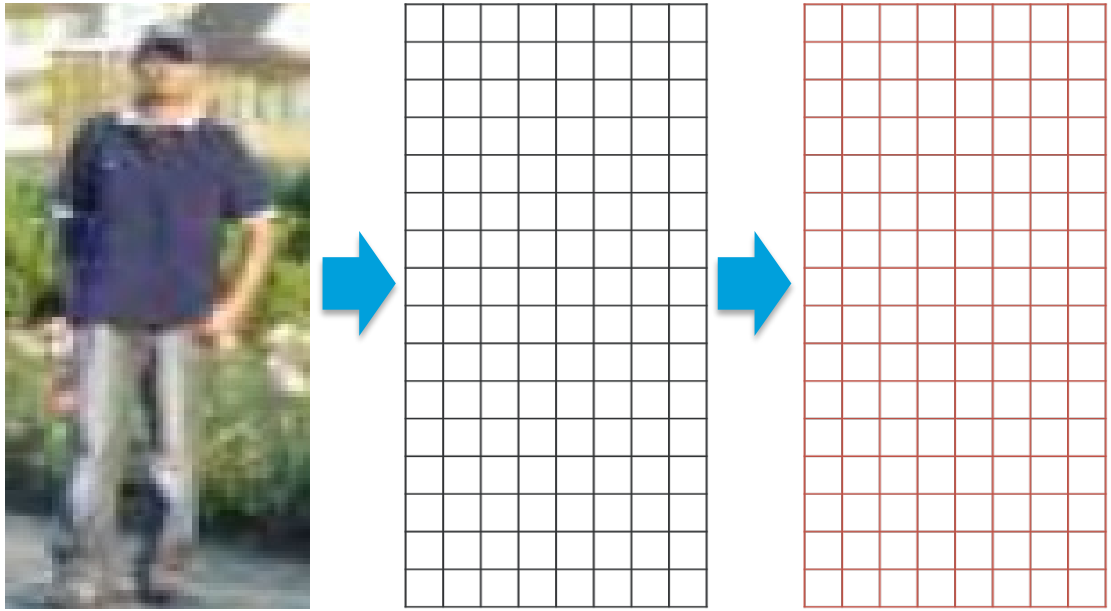








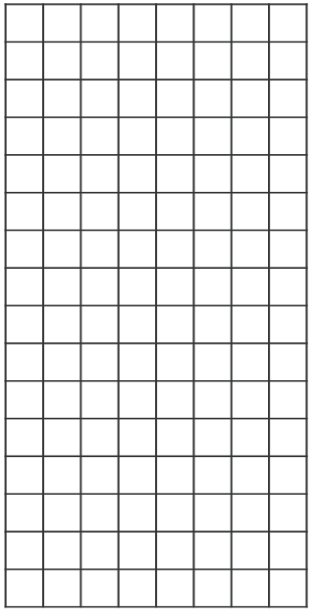
HOG特征：归一化



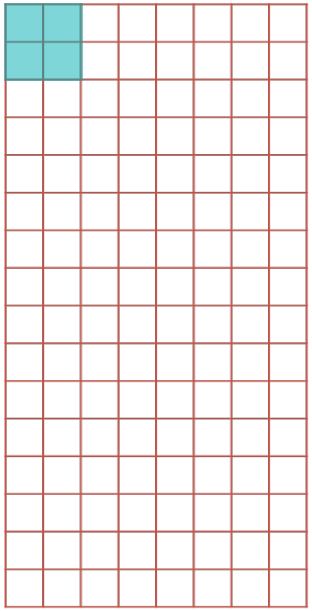
128x64个像素　　16x8个cell　　16x8个9维向量

HOG特征：归一化

128x64个像素



16x8个cell

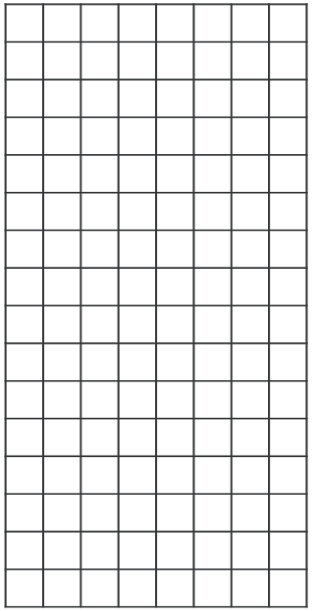


16x8个9维向量

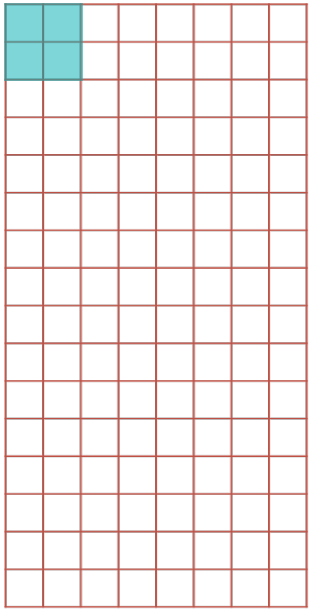
- 相邻的4个cell进行归一化
- 每个cell对应9维向量
- 4个cell共36维向量
- 36维向量进行归一化
- 得到归一化的36维向量

HOG特征：归一化

128x64个像素



16x8个cell

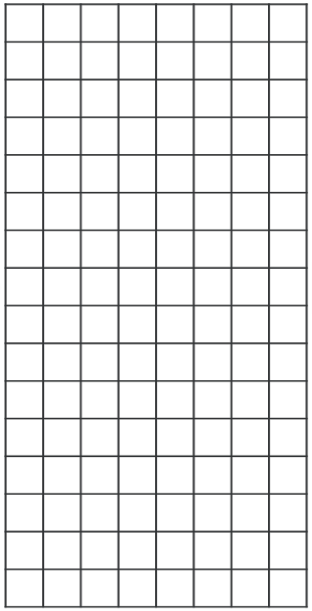


16x8个9维向量

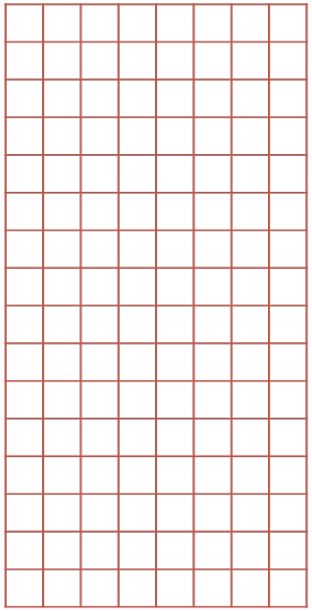
- 相邻的4个cell进行归一化
- 每个cell对应9维向量
- 4个cell共36维向量
- 36维向量进行归一化
- 得到归一化的36维向量
- 逐cell地从左到右、从上到下滑动
- 可以得到15x7个36维归一化向量

HOG特征：归一化

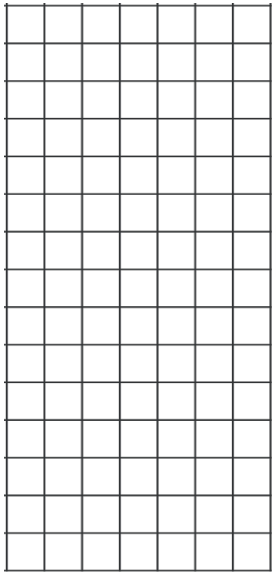
128x64个像素



16x8个cell



16x8个9维向量



15x7个36维向量

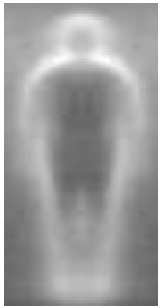
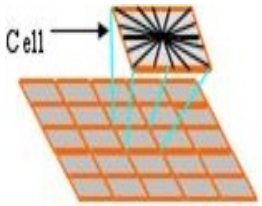
15x7x36=
3780维
特征向量

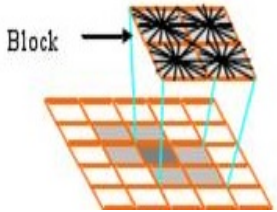
Dalal行人检测算法：HOG特征

- 特征向量：

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



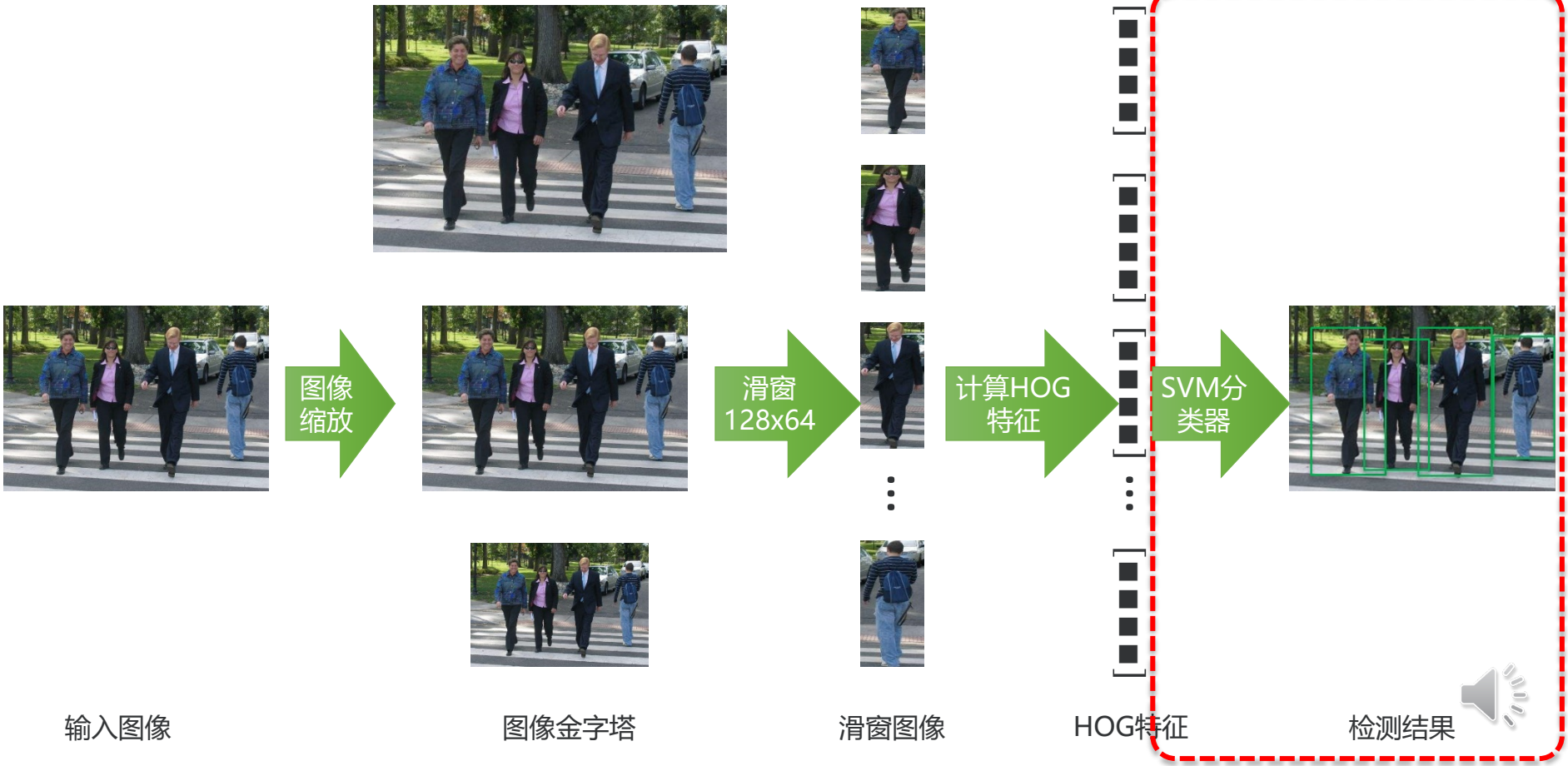


3780维

特征向量

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

Dalal行人检测算法：整体检测流程



Dalal行人检测算法：支持向量机（SVM）

什么是SVM？

支持向量机（SVM）是一个有监督的机器学习算法，它可用于分类和回归分析，最主要是用在分类问题中。

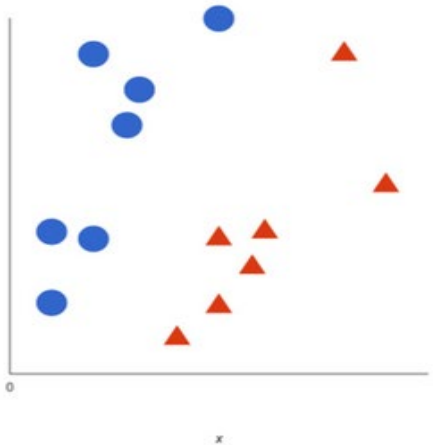
在这个算法中，根据特征值，构建一个n维空间（其中n即是特征数量），把每个数据点投影到此空间内。

数据如何分类？

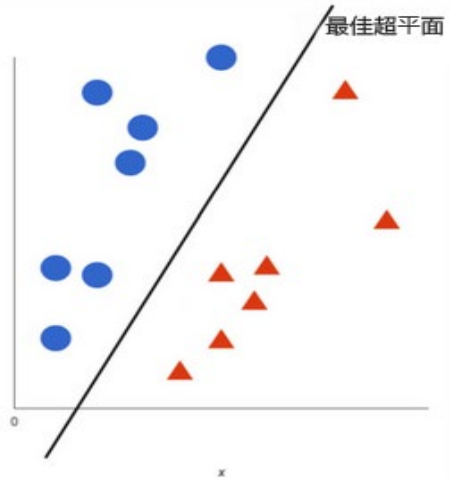
通过查找一个超平面，把数据区分成两类。换句话说，算法输出一个最佳超平面，用于数据分类。

什么是最佳超平面？

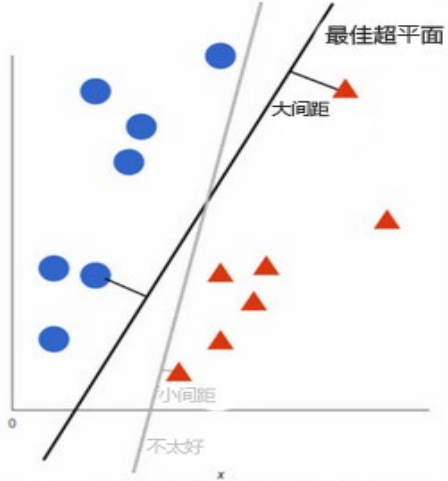
对SVM来说，它指的是距离两类数据最远的一个超平面。即是，此超平面到最近元素的距离最远。



标签数据



在2D空间中，最佳超平面就是一条直线。



不是所有超平面距离相等

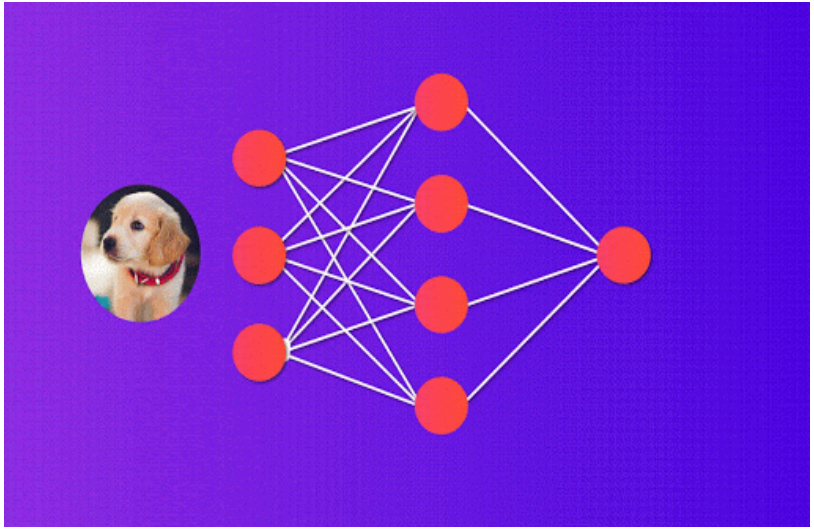
传统行人检测算法：总结

- 滑窗操作遍历所有的位置：在图像上进行密集地窗口滑动以遍历所有的位置
- 图像金字塔遍历所有的大小：对图像进行缩放来解决行人的尺度问题
- 手工特征提取：提取人为设计的手工特征，例如HOG
- 手工分类器：使用人为设计的分类器，例如SVM分类器
- 检测过程分为多个独立的阶段，训练过程比较繁琐
- 传统物体检测算法有较快的速度，但在复杂场景下精度不大理想
- 随着深度学习的出现，传统算法渐渐被取代，但仍有着重要的指导意义

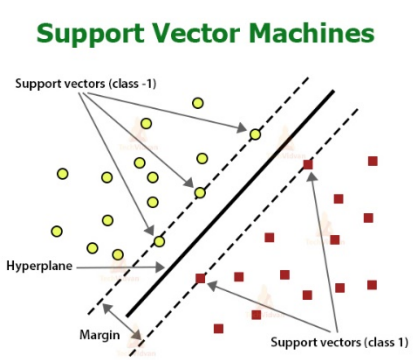
目录

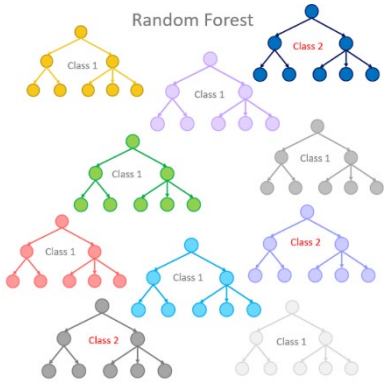
深度学习早期行人检测算法

- 深度学习早期行人检测算法：利用深度学习提取特征，再用传统分类器进行分类，传统方法与深度学习方法的相结合



利用深度学习提取特征

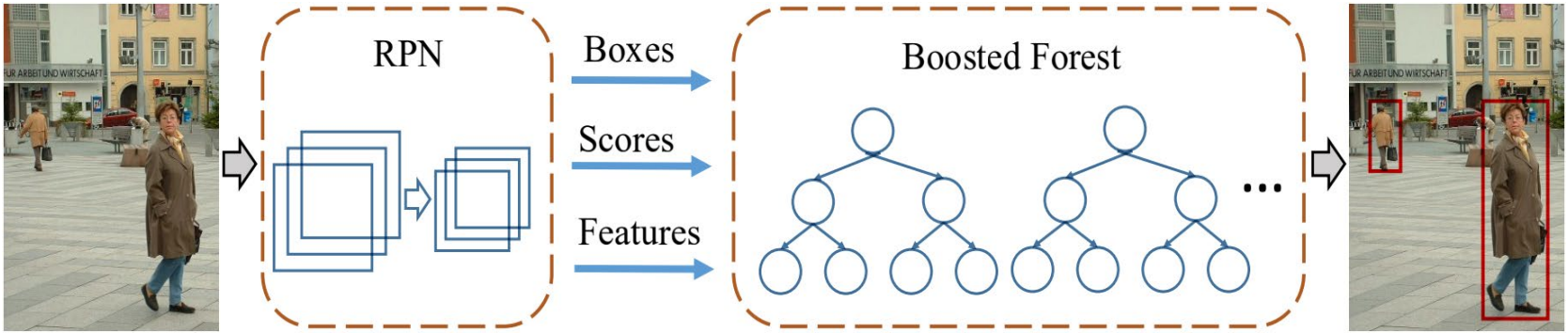




用传统分类器进行分类

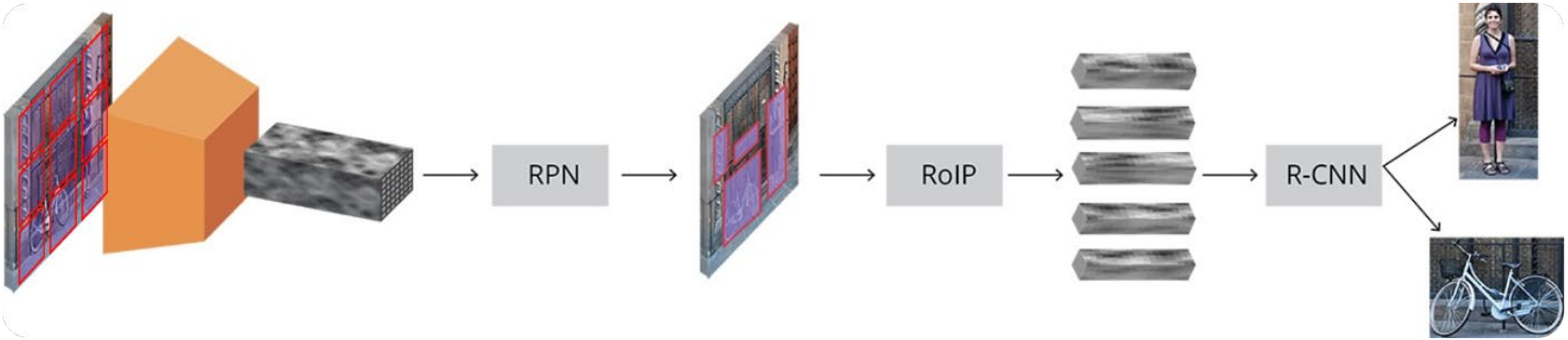
深度学习早期行人检测算法：RPN+BF

- 尝试利用通用物体检测算法Faster R-CNN来做行人检测这一特定任务
- 但是发现Faster R-CNN算法在行人检测任务中的效果不大理想
- 分析通用物体检测算法Faster R-CNN用于行人检测效果不好的原因
- 提出使用RPN + Boosted Forests (RPN + BF) 的行人检测算法
- 在多个benchmark上（Caltech，INRIA，ETH，KITTI）取得了不错的检测结果



Is Faster R-CNN Doing Well for Pedestrian Detection? ECCV 2016

行人检测算法RPN+BF：分析Faster R-CNN



- 把Faster R-CNN中的锚框比例改为0.41，锚框尺度改为9个（$40*1.3^{[0,1,2...8]}$）
- Faster R-CNN在CalTech数据集上只取得了20.2%的平均对数漏检率（MR^{-2}）
- Faster R-CNN做行人检测任务效果不好的原因有以下两点
- 原因1：缺少特殊的策略来处理困难样本
- 原因2：处理小尺度物体时，特征图的分辨率太小

行人检测算法RPN+BF：困难样本的分类策略

- 通用物体检测中，分类错误主要是多类间的混淆，即类别之间互相分错
- 行人检测问题上，分类错误主要是难负样本的混淆，即把背景分为行人
- 如右图所示，这些困难负样本跟行人非常相似，肉眼辨认都需要集中精力
- 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

行人检测算法RPN+BF：困难样本的分类策略

- 通用物体检测中，分类错误主要是多类间的混淆，即类别之间互相分错
- 行人检测问题上，分类错误主要是难负样本的混淆，即把背景分为行人
- 如右图所示，这些困难负样本跟行人非常相似，肉眼辨认都需要集中精力
- 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

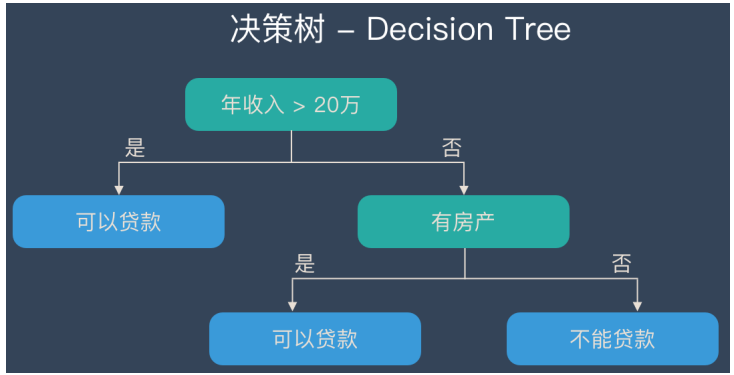
解决方案

- 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

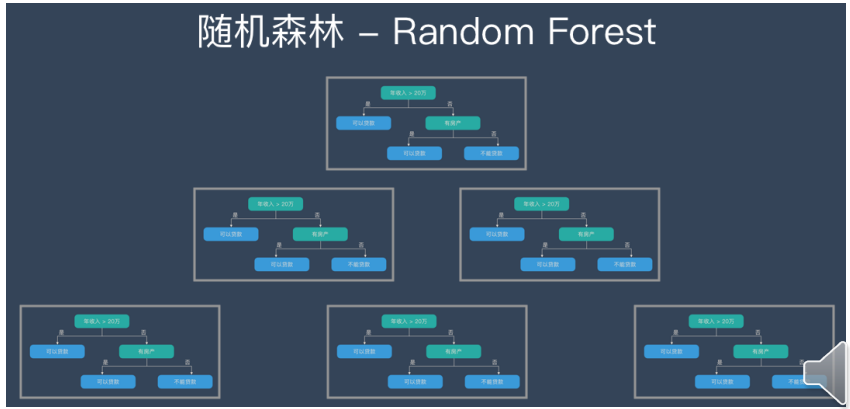
Method	MR^{-2} (%)
RPN + Fast R-CNN	20.2
RPN + BF	18.2

行人检测算法RPN+BF：Boosted Forest (BF)

- 决策树（Decision Tree）是一种逻辑简单的机器学习算法，它是一种树形结构
- https://easyai.tech/ai-definition/decision-tree

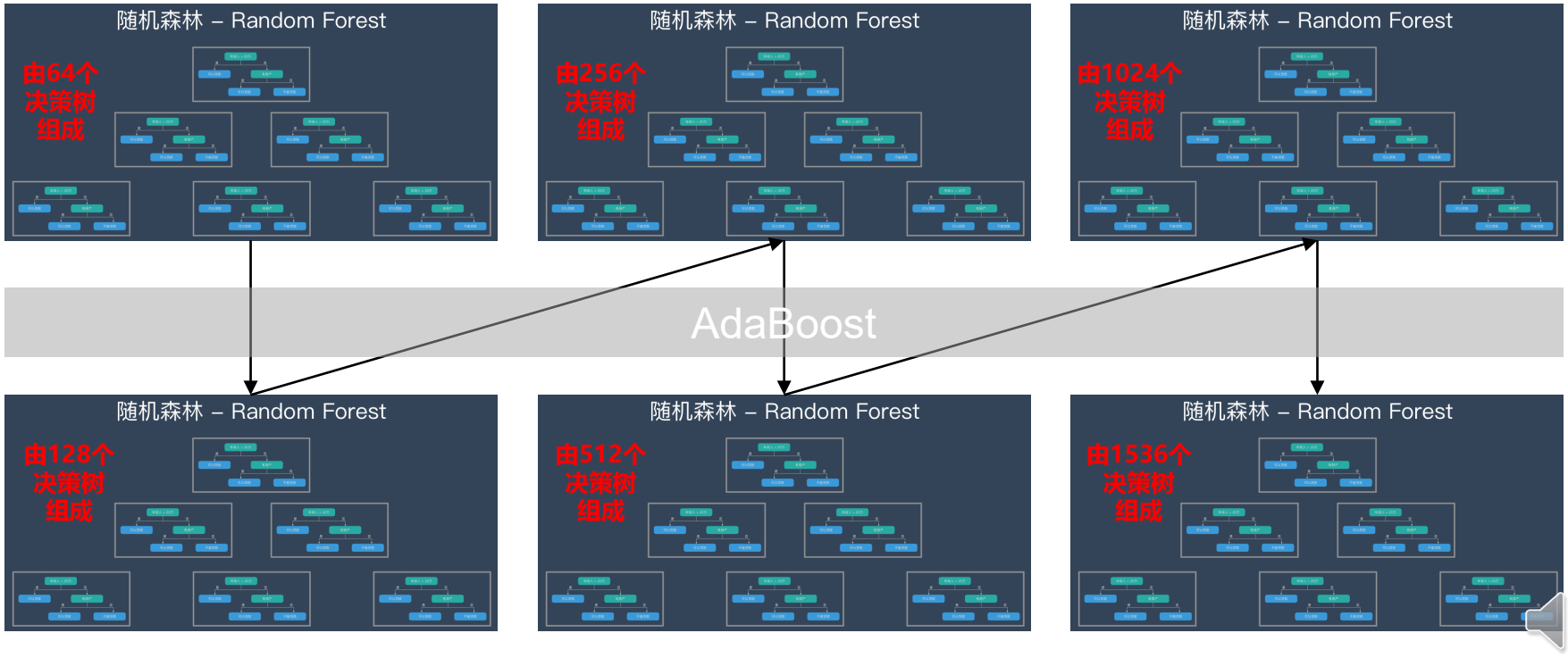


- 随机森林（Random Forest）是一种由决策树构成的集成算法
- https://easyai.tech/ai-definition/random-forest



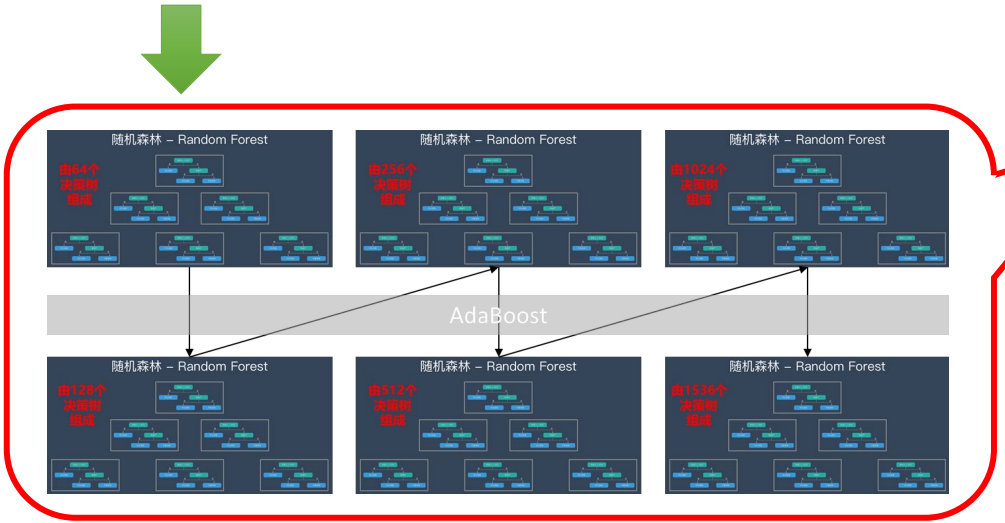
行人检测算法RPN+BF：Boosted Forest (BF)

- Boosted Forest是由几个随机森林，按照改进版的AdaBoost训练的分类器



行人检测算法RPN+BF：Boosted Forest (BF)

~50k正样本 + ~50k负样本



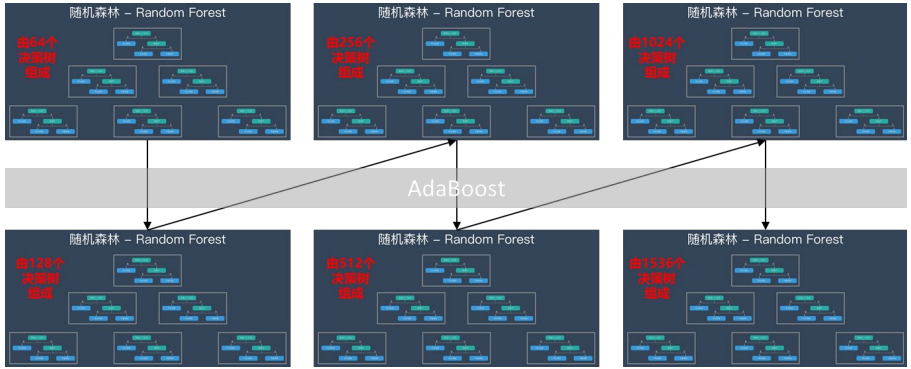
6个随机森林，对应着6个训练阶段

每个训练阶段后，挖掘~5k个困难样本

~30k难负样本

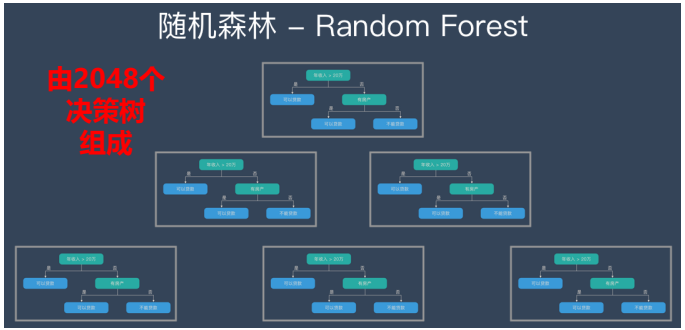
行人检测算法RPN+BF：Boosted Forest (BF)

~50k正样本 + ~50k负样本



~30k难负样本

~50k正样本 + ~50k负样本 + ~30k难负样本



最终的分类器

行人检测算法RPN+BF：小尺度物体的特征

- 在CalTech数据集上，有不少小尺度的行人，它们的尺寸约为28x70像素
- 而Faster R-CNN中RoIPooling扣特征的检测层，其下采样倍数是16
- 那么28x70大小的行人映射到该检测层上的大小仅为2x4
- RoIPooling操作会从2x4大小的区域重复采样得到7X7大小的特征
- 7x7大小的特征源自2x4的特征，可利用的特征太少，不利于后续的分类和回归

行人检测算法RPN+BF：小尺度物体的特征

- 在CalTech数据集上，有不少小尺度的行人，它们的尺寸约为28x70像素
- 而Faster R-CNN中RoIPooling扣特征的检测层，其下采样倍数是16
- 那么28x70大小的行人映射到该检测层上的大小仅为2x4
- RoIPooling操作会从2x4大小的区域重复采样得到7X7大小的特征
- 7x7大小的特征源自2x4的特征，可利用的特征太少，不利于后续的分类和回归

解决方案

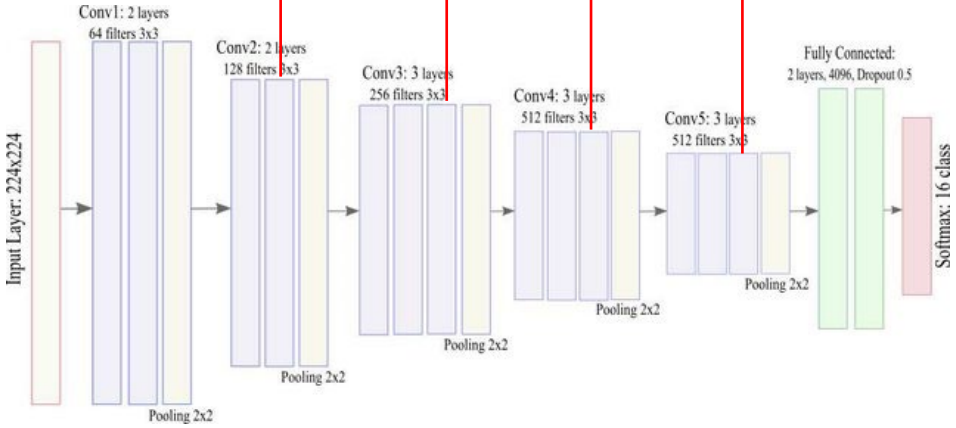
- 从更浅的、分辨率更高的特征层上来进行特征扣取
- 从不同分辨率的特征层上扣取特征并进行融合
- 去掉下采样 + 空洞卷积，来增加特征层的分辨率

RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

行人检测算法RPN+BF：特征增强

- 从更浅的、分辨率更高的特征层上来进行特征扣取

RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

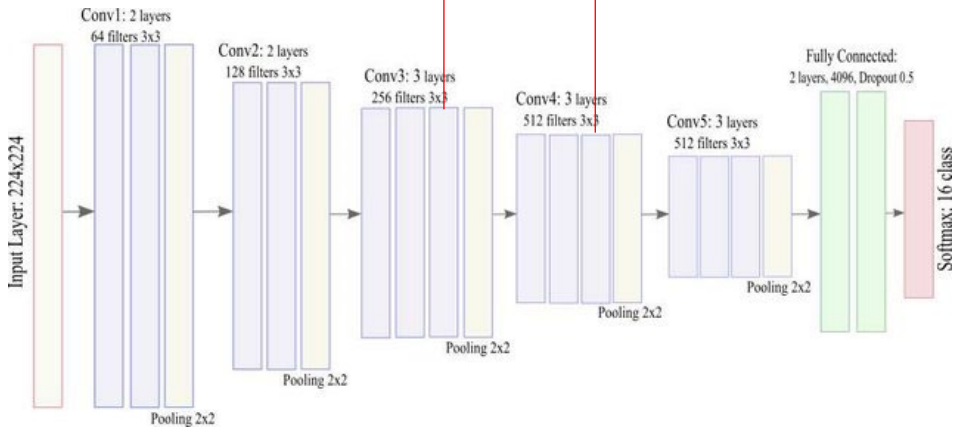


- **浅层的特征图分辨率更大，扣取的特征更丰富，检测效果更好**
- **太浅层的特征，其语义信息不够丰富，不利于分类**

行人检测算法RPN+BF：特征增强

- 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

RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

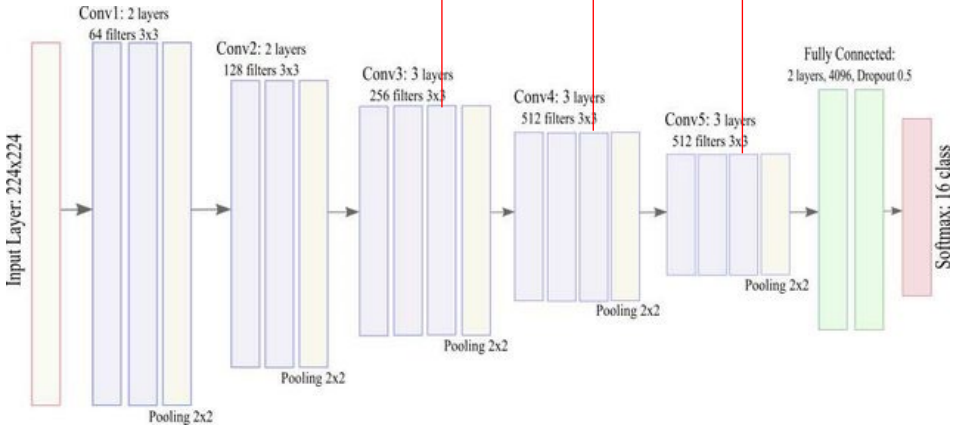


- **融合不同层的特征，特征更丰富，检测效果更好**

行人检测算法RPN+BF：特征增强

- 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

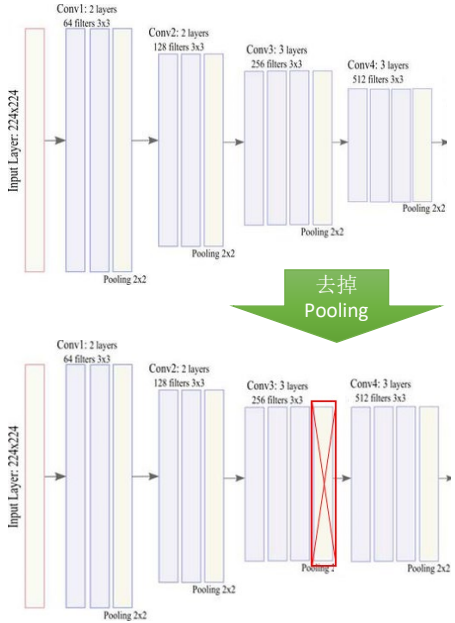
RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**



- **融合不同层的特征，特征更丰富，检测效果更好**
- **Conv5_3的特征分辨率太小，特征细节丢失，不利于检测**

行人检测算法RPN+BF：特征增强

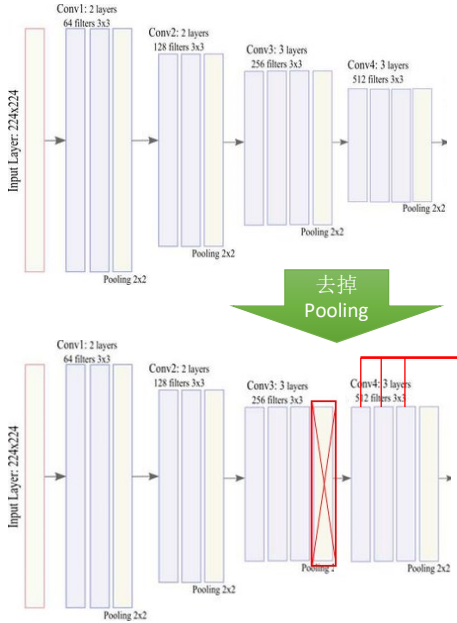
- 从更浅的、分辨率更高的特征层上来进行特征扣取
- 从不同分辨率的特征层上扣取特征并进行融合
- 去掉下采样 + 空洞卷积，来增加特征层的分辨率



RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

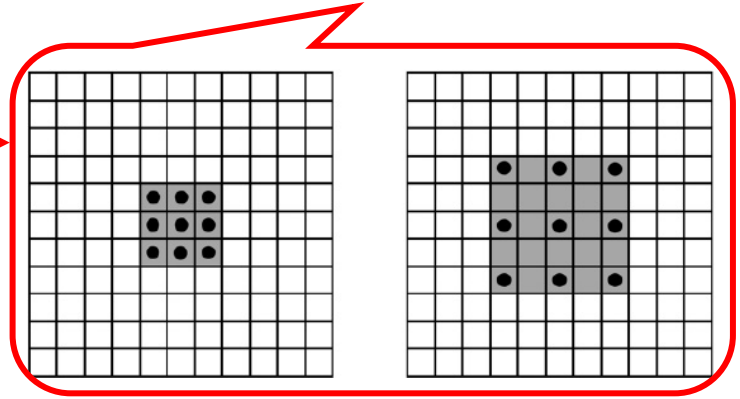
行人检测算法RPN+BF：特征增强

- 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

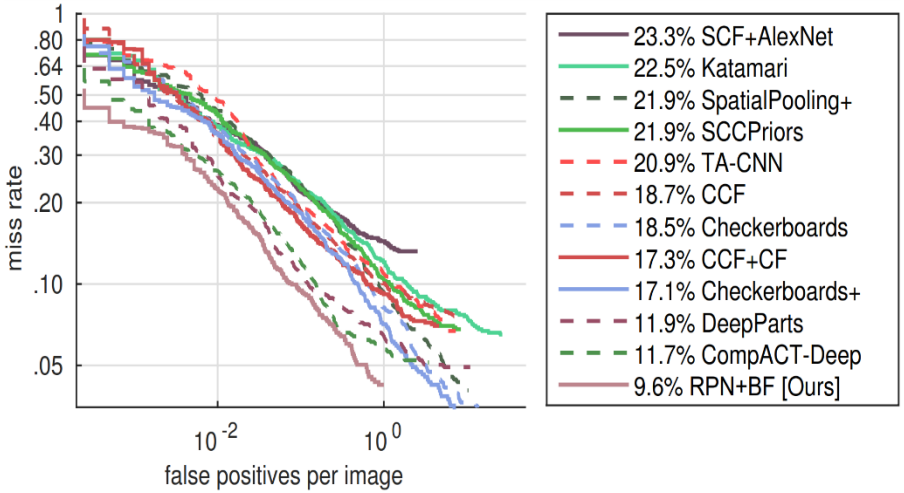


RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

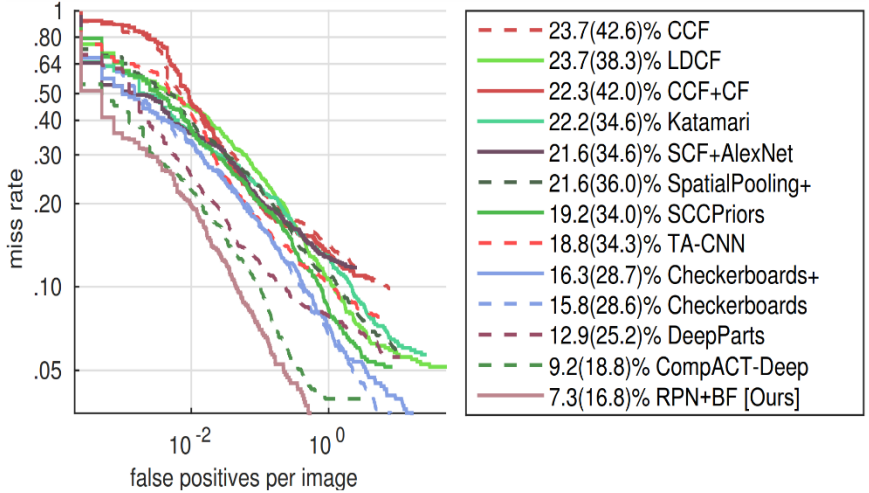
空洞卷积：卷积核之间有间隙，以增大感受野



行人检测算法RPN+BF：检测精度



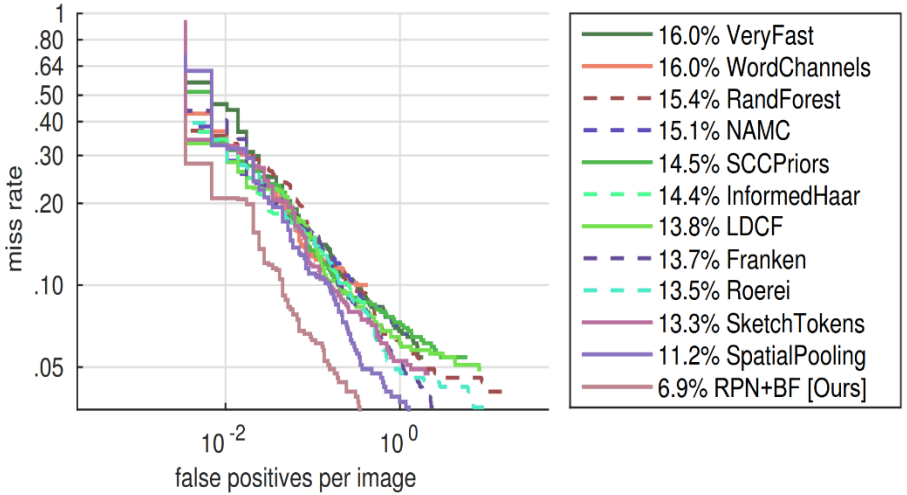
CalTech数据集（旧标注）



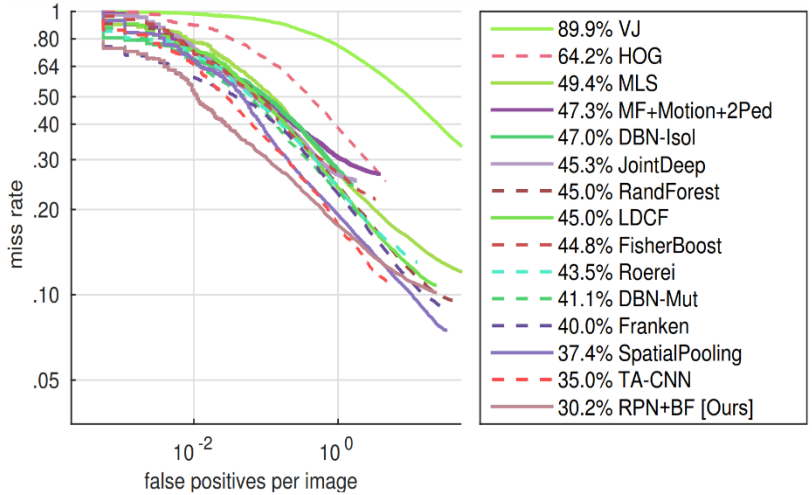
CalTech数据集（新标注）

How far are we from solving pedestrian detection? CVPR 2016

行人检测算法RPN+BF：检测精度



INRIA数据集



ETH数据集

行人检测算法RPN+BF：检测速度

- 硬件设备：Tesla K40 GPU
- 图像大小：CalTech x1.5，即960x720分辨率
- 检测速度：0.5秒/每张图，即2 FPS

method	hardware	time/img (s)	MR (%)
LDCF [24]	CPU	0.6	24.8
CCF [25]	Titan Z GPU	13	17.3
CompACT-Deep [6]	Tesla K40 GPU	**0.5**	11.7
RPN+BF [ours]	Tesla K40 GPU	**0.5**	**9.6**

深度学习早期行人检测算法：总结

- 特点：传统方法与深度学习方法相结合
- 深度学习：利用深度学习中的神经网络生成候选区域和相应特征
- 传统方法：利用传统的手工分类器对检测结果进行再次分类
- 缺点：整个检测过程分为多个独立的阶段，训练和测试都比较繁琐
- 随着通用物体检测算法的发展，它们在行人检测领域的精度越来越高
- 这类算法逐渐被完全基于通用物体检测算法的深度学习后期行人检测算法所取代

课程作业

- 测试RPN+BF行人检测算法

1. RPN+BF的官方代码链接（https://github.com/zhangliliang/RPN_BF）

2. 按照安装教程，配置好相关环境

3. 利用官方提供的模型，测试RPN+BF算法的性能

4. 撰写RPN+BF的测试报告，包括RPN+BF检测效果图，所存在的问题等

感谢各位聆听!

Thanks for Listening