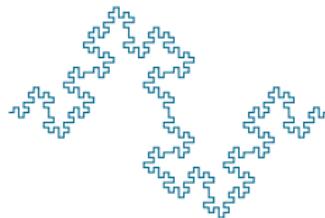


# 目标检测一唱三叹

更准，更快，更智能

邬书哲

中科院计算所 VIPL 研究组



2016 年 7 月 17 日

# 主要内容

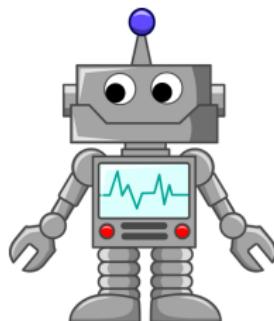
- ▶ 目标检测概述
  - ▶ 任务定义，性能评测
  - ▶ 代表性方法简介
- ▶ R-CNN 系列——更准
  - ▶ R-CNN/Fast R-CNN/Faster R-CNN
  - ▶ OHEM/R-FCN
- ▶ 级联结构目标检测器——更快
  - ▶ Cascade CNN/MT-CNN
  - ▶ 级联结构的联合训练
- ▶ 案例研习：人脸检测
- ▶ 目标检测与弱监督学习——更智能

# 目标检测概述

计算机看世界！



这是在室内还是街上呢?  
前面有没有人在过马路?



这是刘大大吗?  
有哪些店卖刘大大同款的T恤?

这些问题的解决都离不开目标检测！

## 目标检测任务



- ▶ 给定待检测的目标类别
  - ▶ **输入**: 待处理的图像
  - ▶ **输出**:
    - ▶ 图像是否包含给定类别的目标
    - ▶ 目标所处的位置, 以及目标的大小 (通常用矩形边框来表示)

## 目标检测任务



### (a) Object Categorization



## (b) Object Class Detection

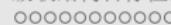
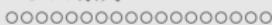
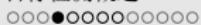
Car



### (c) Figure-Ground Segmentation



#### (d) Semantic Segmentation



## 什么样的目标检测器更好？

- **速度**: 处理一张图像要多长时间?

- ▶ 图像大小
  - ▶ 目标数目

- ## ► 精度：

- ▶ 漏检：没有检测到的目标
  - ▶ 误检：错误的检测

- 其它

- ▶ 训练时间
  - ▶ 模型存储所需空间
  - ▶ 算法空间复杂度
  - ▶ 不同场景下的迁移能力
  - ▶ 从单类到多类的推广能力

## 精度评价指标

评测：检测框 vs 标注框

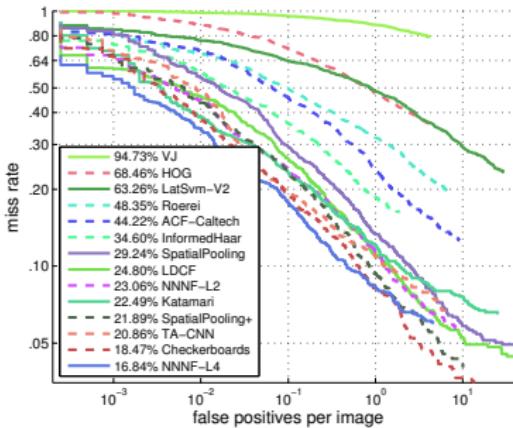
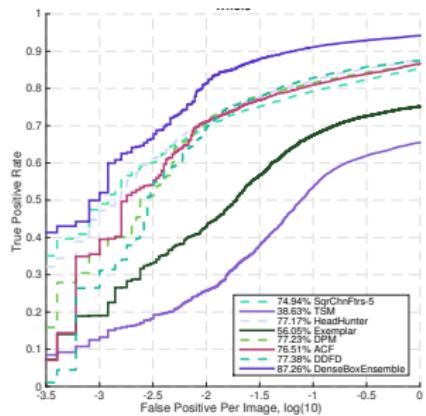
## ► 框的匹配

- ▶ 交并比 INTERSECTION-OVER-UNION, IOU
  - ▶ 边框的一致性问题

## ► 精度指标

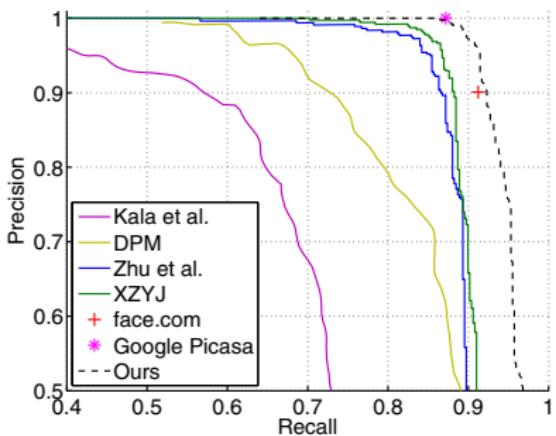
- ▶ 召回率 RECALL
  - ▶ 漏检率 MISS RATE
  - ▶ 误检率 FALSE POSITIVE PER IMAGE, FPPI
  - ▶ 精确率 PRECISION
  - ▶ 平均精确率 AVERAGE PRECISION

## 精度评价曲线：ROC



- ▶  $x$  轴: 误检率
  - ▶  $y$  轴: 召回率, 漏检率

## 精度评价曲线：PR

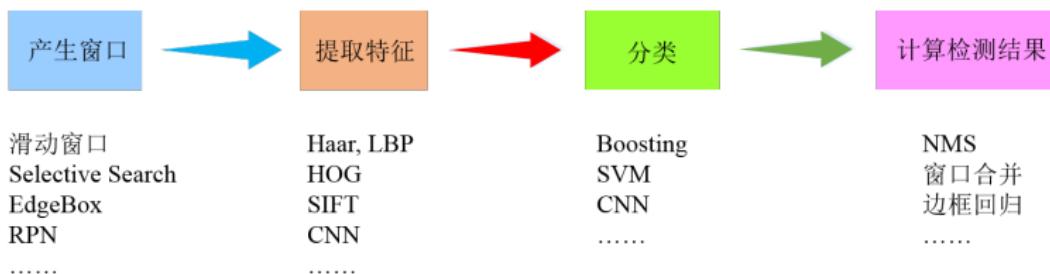


- ▶  $x$  轴: 召回率
  - ▶  $y$  轴: 精确率

评测数据集

- ▶ Pascal VOC
    - ▶ 2007, 2010, 2012
    - ▶ 20 类物体
  - ▶ ImageNet
    - ▶ ILSVRC 2013, 2014, 2015, 2016
    - ▶ 200 类物体
  - ▶ MS COCO
    - ▶ 2015
    - ▶ 80 类物体
  - ▶ 其它
    - ▶ 人脸检测: AFW, FDDB, MALF, IJB-A, WIDER FACE
    - ▶ 行人检测: INRIA, Caltech Pedestrian, KITTI
    - ▶ 车辆检测: KITTI

# 基本的检测思路



## ▶ 主要的几类检测方法

- ▶ 级联结构检测器：早在上世纪 80 年代就出现，从 VJ 人脸检测器开始风靡
- ▶ 部件模型：其思想在上世纪 70 年代已有体现，DPM 的出现使检测精度得到明显提升
- ▶ 神经网络：早期有不少关于人脸检测的工作，R-CNN 之后已成为主流的目标检测器
- ▶ 其它：基于范本的方法，概率模型

# BOOSTING 方法 + 级联结构

- ▶ 经典的 Viola-Jones (人脸) 检测器

- ▶ 快速计算的特征: Haar
  - ▶ 高效的分类器: boosting 方法
  - ▶ 级联结构 ATTENTIONAL CASCADE

- ▶ 特征表示和模型上的改进

- ▶ LBP, LAB, SURF, ICF/ACF
  - ▶ 链式 Boosting, 嵌套式 Boosting, 特征继承
  - ▶ 多任务: Joint Cascade

- ▶ 结构上的改进

- ▶ 金字塔形级联结构, 树形级联结构
  - ▶ 软级联 SOFT CASCADE

- ▶ 功能上的改进

- ▶ 串话级联 CROSSTALK CASCADE

# 可变形模板 & 部件模型

- ▶ 可变形部件模型 **DEFORMABLE PART MODEL, DPM**
  - ▶ HOG 特征
  - ▶ 支持向量机 **SUPPORT VECTOR MACHINE, SVM**
  - ▶ 部件模板，物体模板
- ▶ 多层结构
  - ▶ 对部件做进一步细分
- ▶ 加速
  - ▶ Cascade DPM
  - ▶ Fastest DPM: 低秩分解，查表法提取 HOG 特征

# 神经网络 → 深度学习

- ▶ 早期基于神经网络的目标检测器
  - ▶ 主要是人脸检测器
  - ▶ 采用多层感知机 **MLP** 或者卷积神经网络 **CNN**
  - ▶ 基于滑动窗口范式进行检测
- ▶ 伴随深度学习应运而生的检测器
  - ▶ R-CNN, Fast R-CNN, Faster R-CNN, R-FCN
  - ▶ SPP-Net, YOLO, G-CNN, HyperNet, SSD
  - ▶ DenseBox
  - ▶ 基于候选窗口生成方法进行检测，或者回归响应热图

# 各类检测器的比较

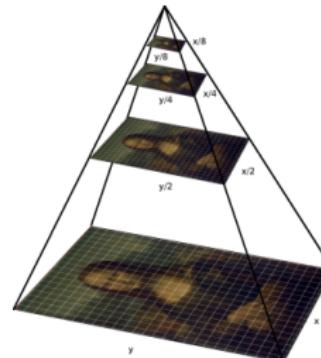
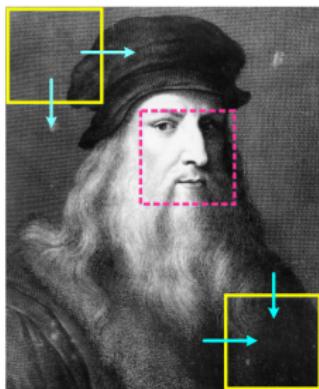
检测器	特点	速度	精度
级联结构	模型简单	快	一般
部件模型	同时建模局部和整体，以及几何特性	较慢	较好
深度网络	很强的非线性建模能力，自动学习特征表示	慢	state-of-the-art

## R-CNN 系列

# “新”的检测框架

## ▶ 基本思路

- ▶ 面对复杂多样的物体，需要容量更大、非线性建模能力更强的分类器
- ▶ 滑动窗口范式的输入规模过大，限制了分类器的复杂程度
- ▶ 能不能在保证召回率的条件下，尽可能生成少量窗口？



- ▶ **滑动窗口范式：**在二维空间和尺度空间进行暴力枚举，窗口数量： $10^5$
- ▶ **候选窗口生成方法：**找出最有可能包含目标的边框，窗口数量： $10^2 \sim 10^3$

# 候选窗口生成方法

- ▶ 选择性搜索 **SELECTIVE SEARCH**

- ▶ 无监督方法
- ▶ 采用图像分割的思路，基于多种特征进行超像素的合并
- ▶ 速度：2s/图（CPU）

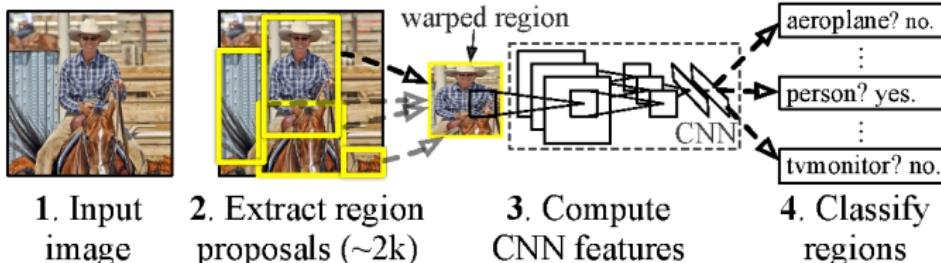
- ▶ EdgeBox

- ▶ 以滑动窗口作为初始化
- ▶ 基于边缘不断对窗口进行调整

- ▶ 其它方法

- ▶ MCG, CPMC, BING, Objectness, .....

# R-CNN

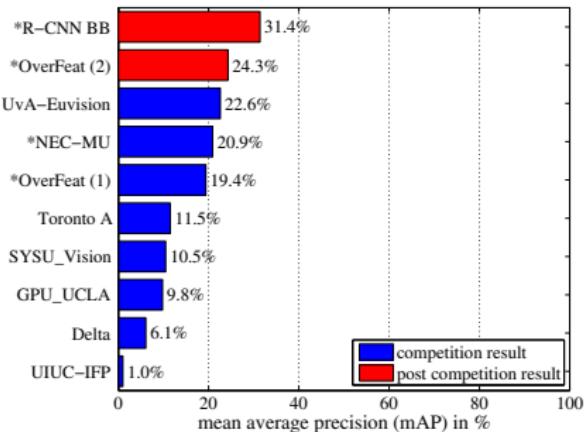


- ▶ 裁剪候选窗口区域:  $227 \times 227$ , 周围添补 6 个像素宽的上下文区域 CONTEXT
- ▶ 学习 CNN (特征表示)
  - ▶ 监督预训练 PRE-TRAINING : ILSVRC 2012, 图像分类任务
  - ▶ 精调 FINE-TUNING : 初始学习率降低为原来的  $\frac{1}{10}$
- ▶ 学习 SVM (二元分类器):
  - ▶ 正例: 标注框; 反例: 交并比阈值 = 0.3
- ▶ 边框回归 BOUNDING BOX REGRESSION
  - ▶ 岭回归 RIDGE REGRESSION ; 训练样本: 交并比阈值 = 0.6; 不同类别分别学习

R-CNN

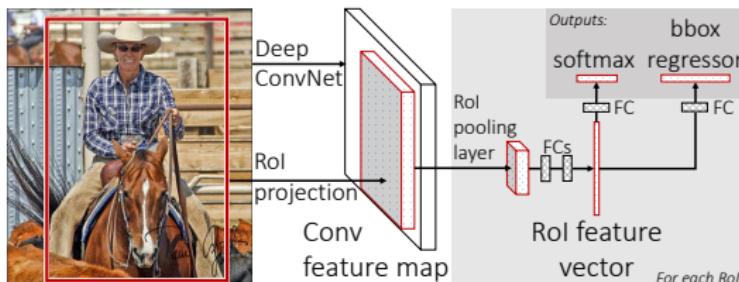
VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20] <sup>†</sup>	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18] <sup>†</sup>	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	<b>71.8</b>	<b>65.8</b>	<b>53.0</b>	<b>36.8</b>	<b>35.9</b>	<b>59.7</b>	<b>60.0</b>	<b>69.9</b>	<b>27.9</b>	<b>50.6</b>	<b>41.4</b>	<b>70.0</b>	<b>62.0</b>	<b>69.0</b>	<b>58.1</b>	<b>29.5</b>	<b>59.4</b>	<b>39.3</b>	<b>61.2</b>	<b>52.4</b>	<b>53.7</b>

IJSVRC2013 detection test set mAP



生成候选窗口 + 提取特征:  
GPU 13s/图, CPU 53s/图

# FAST R-CNN



## ▶ 全图卷积 → ROI 投影 → ROI Pooling

- ▶ 借鉴 SPPnet 的做法，避免提取候选窗口特征过程中的重复计算
- ▶ 采用单尺度的空间金字塔池化 **SPATIAL PYRAMID POOLING** 操作得到定长的特征向量

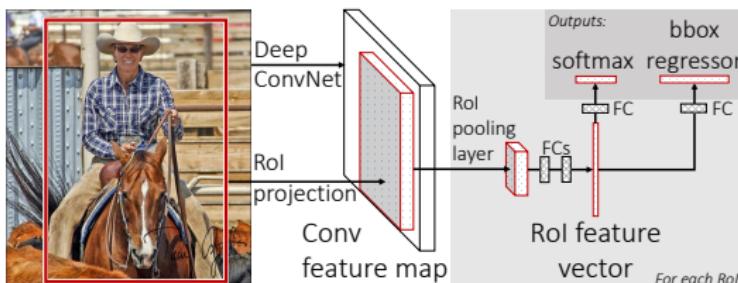
特征图大小:  $a \times a$

ROI Pooling 之后的大小:  $n \times n$

池化窗口大小:  $\lceil \frac{a}{n} \rceil \times \lceil \frac{a}{n} \rceil$

池化窗口的滑动步长:  $\lfloor \frac{a}{n} \rfloor$

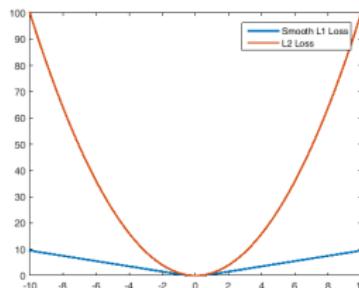
# FAST R-CNN



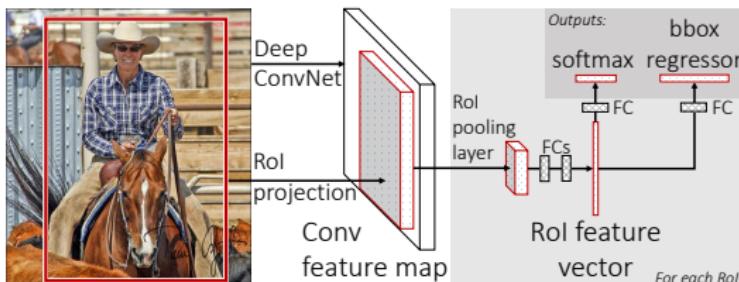
## ► 多任务损失函数

- 分类：交叉熵损失函数
- 边框回归：光滑  $L_1$  损失函数

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



# FAST R-CNN

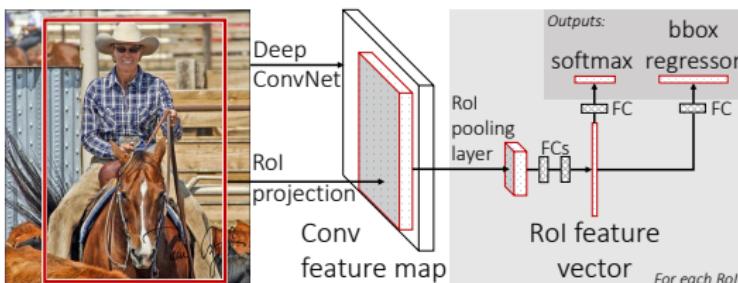


## ► 压缩全连接层：矩阵的奇异值分解 SVD

- 分解之前:  $uv$
- 分解之后:  $t(u + v)$
- 在参数数量减少的同时，计算量也减小了，尤其是当  $t \ll \min\{u, v\}$  时
- 一个全连接层  $\rightarrow$  两个全连接层（中间没有激活函数）

$$W \approx U\Sigma_t V^T$$

# FAST R-CNN



## ▶ 训练样本

- ▶ 以图像为中心的采样方式
- ▶ 正例：交并比阈值 = 0.5；反例：交并比  $\in [0.1, 0.5)$
- ▶ 正反例比例：1 : 3
- ▶ 样本扰动：每个样本以 0.5 的概率进行翻转

# FAST R-CNN

method	data	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN.c2000	?	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3	63.8
R-CNN BB [8]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours] <sup>†</sup>	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours] <sup>‡</sup>	07++12	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	68.4

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS\_NIN.c2000 use unspecified networks. All other methods use VGG16. Training data key: see Table 2, “?” unknown. Results: <sup>†</sup><http://goo.gl/weNq2Z>, <sup>‡</sup><http://goo.gl/o1tz10>

	Fast R-CNN			R-CNN			SPPnet
	S	M	L	S	M	L	<sup>†</sup> L
train time (h)	1.2	2.0	9.5	22	28	84	25
train speedup	18.3×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	0.06	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	213×	-	-	-	-
VOC07 mAP	57.1	59.2	66.9	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

还是不够快？

生成候选窗口占用了大量时间！

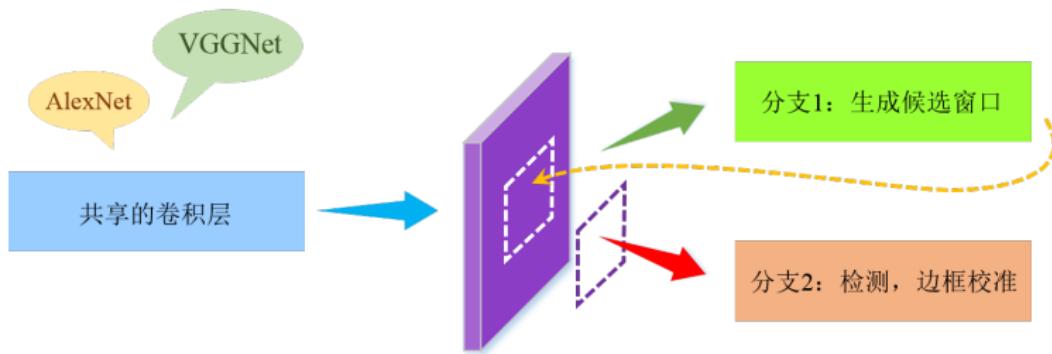
# FAST R-CNN

## ▶ 速度瓶颈：单独的候选区域生成方法

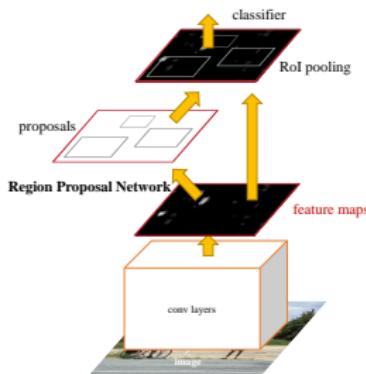
- ▶ Selective Search: CPU, 2s/图
- ▶ Edge Boxes: GPU, 0.2s/图, 和 CNN 检测的时间相当

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea- tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	***	*	-
CPMC [19]	Grouping	✓	✓	✓	250	-	**	*
EdgeBoxes [20]	Window scoring		✓	✓	0.3	**	***	****
Endres [21]	Grouping	✓	✓	✓	100	-	***	**
Geodesic [22]	Grouping	✓		✓	1	*	***	**
MCG [23]	Grouping	✓	✓	✓	30	*	***	****
Objectness [24]	Window scoring		✓	✓	3	-	*	-
Rahtu [25]	Window scoring		✓	✓	3	-	-	*
RandomizedPrim's [26]	Grouping	✓		✓	1	*	*	**
Rantalankila [27]	Grouping	✓		✓	10	**	-	**
Rigor [28]	Grouping	✓		✓	10	*	**	**
SelectiveSearch [29]	Grouping	✓	✓	✓	10	**	***	****
Gaussian				✓	0	-	-	*
SlidingWindow				✓	0	***	-	-
Superpixels		✓			1	*	-	-
Uniform				✓	0	-	-	-

# FASTER R-CNN



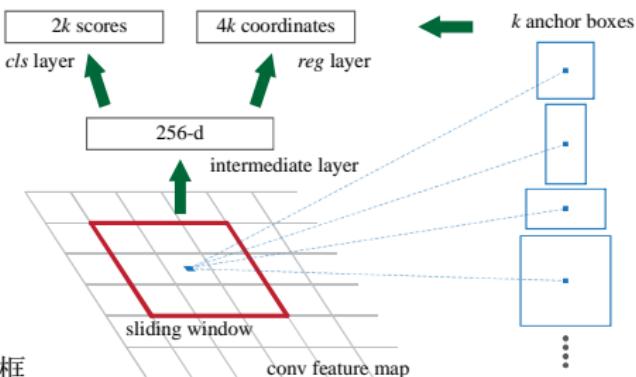
- ▶ 两个 CNN 共享卷积层
  - ▶ 生成候选窗口：RPN
  - ▶ 检测：Fast R-CNN



# FASTER R-CNN: RPN

## ▶ 生成候选窗口

- ▶ 滑动窗口范式：以  $3 \times 3$  的窗口在卷积特征图上滑动
- ▶ 特征表示：每个通道聚合成 1 维特征，256 个特征图  $\rightarrow$  256 维特征向量
- ▶ 确定候选窗口：分类 + 边框回归
- ▶ 用  $1 \times 1$  的卷积来替代全连接层



## ▶ 参考边框的设计 ANCHOR Box

- ▶ 同一个位置可以有  $k = 9$  种不同的边框
- ▶ 参考边框具有不同的尺度和长宽比

anchor	$128^2, 2:1$	$128^2, 1:1$	$128^2, 1:2$	$256^2, 2:1$	$256^2, 1:1$	$256^2, 1:2$	$512^2, 2:1$	$512^2, 1:1$	$512^2, 1:2$
proposal	$188 \times 111$	$113 \times 114$	$70 \times 92$	$416 \times 229$	$261 \times 284$	$174 \times 332$	$768 \times 437$	$499 \times 501$	$355 \times 715$

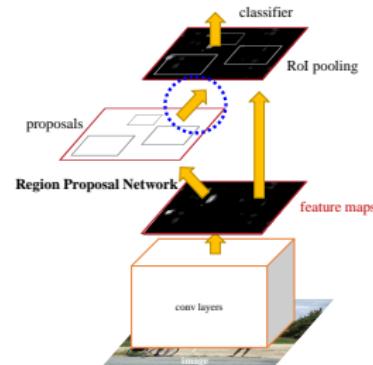
# FASTER R-CNN：训练方法

## ► 4 步法：交替式优化

- ▶ 基于 ImageNet 预训练模型，训练 RPN
- ▶ 基于 ImageNet 预训练模型，以及上一步训练的 RPN，训练 Fast R-CNN
- ▶ 固定共享的卷积层，训练 RPN
- ▶ 固定共享的卷积层，基于上一步训练的 RPN，训练 Fast R-CNN

## ► 1 步法：端到端优化

- ▶ 近似：忽略候选窗口参数上的梯度（可获得和交替式优化相当的性能，但训练时间更短）
- ▶ 非近似：要求 RoIPooling 层对候选窗口的参数可导



# 案例研习： PY-FASTER-RCNN

## ▶ 环境搭建

- ▶ git clone --recursive <url> <local\_path>
- ▶ 编译 caffe-fast-rcnn, WITH\_PYTHON\_LAYER := 1
- ▶ 编译 lib
- ▶ 依赖的 Python 包: python-opencv, easydict

## ▶ 用新的数据集训练

- ▶ factory.py
- ▶ 新的数据集类: my\_dataset.py

## ▶ 采用新的网络结构

- ▶ 用近似的端到端优化来训练
- ▶ ResNet

## ▶ 修改训练和测试参数

- ▶ faster\_rcnn\_end2end.yml
- ▶ 增加命令行接口参数

# 案例研习： PY-FASTER-RCNN

## ▶ 使用新的数据集

- ▶ 自行编写数据集类（参考： pascal\_voc.py）
- ▶ 数据组织方式： \_load\_image\_set\_index, image\_path\_from\_index
- ▶ 标注文件格式： \_load\_pascal\_annotation

## ▶ 常见错误

- ▶ 忘记修改 factory.py
- ▶ 忘记 import 新增的类

```
return {'boxes' : boxes,
        'gt_classes': gt_classes,
        'gt_overlaps' : overlaps,
        'flipped' : False}
```

```
for year in ['2007', '2012']:
    for split in ['train', 'val', 'trainval', 'test']:
        name = 'voc_{}_{}'.format(year, split)
        _sets[name] = (lambda split=split, year=year:
                       datasets.pascal_voc(split, year))
```

```
self._classes = ('__background__', # always index 0
                 'aeroplane', 'bicycle', 'bird', 'boat',
                 'bottle', 'bus', 'car', 'cat', 'chair',
                 'cow', 'diningtable', 'dog', 'horse',
                 'motorbike', 'person', 'pottedplant',
                 'sheep', 'sofa', 'train', 'tvmonitor')
```

## ▶ 相关资料

- ▶ <https://github.com/deboc/py-faster-rcnn/tree/master/help>
- ▶ <https://github.com/rbgirshick/py-faster-rcnn/issues/243>

## 案例研习： PY-FASTER-RCNN

- ▶ 采用新的网络结构： ResNet-50

- ▶ 下载模型配置文件：<https://github.com/KaimingHe/deep-residual-networks>
- ▶ 编写 train.prototxt
- ▶ 编写 test.prototxt

# 案例研习： PY-FASTER-RCNN

## ▶ 采用新的网络结构：ResNet-50

- ▶ 下载模型配置文件：<https://github.com/KaimingHe/deep-residual-networks>
- ▶ 编写 train.prototxt
- ▶ 编写 test.prototxt

## ▶ 训练

- ▶ 修改类别数目：input-data 层
- ▶ 增加 Fast R-CNN 层：替换原来的分类层 fc1000 和 prob
- ▶ 插入 RPN 层：res4f\_relu 之后

```
layer {
    name: "rpn_conv/3x3"
    type: "Convolution"
    bottom: "res4f"
    top: "rpn/output"
    param { lr_mult: 1.0 }
    param { lr_mult: 2.0 }
    convolution_param {
        num_output: 512
        kernel_size: 3 pad: 1 stride: 1
        weight_filler { type: "gaussian" std: 0.01 }
        bias_filler { type: "constant" value: 0 }
    }
}
```

```
...
name: "cls_score"
...
name: "bbox_pred"
...
name: "loss_cls"
...
name: "loss_bbox"
...
...
name: "rpn_relu/3x3"
...
name: "rpn_cls_score"
...
name: "rpn_bbox_pred"
...
name: "rpn_cls_score_reshape"
...
name: 'rpn-data'
...
name: "rpn_loss_cls"
...
name: "rpn_loss_bbox"
...
...
```

## 案例研习： PY-FASTER-RCNN

- ▶ 采用新的网络结构： ResNet-50

- ▶ 下载模型配置文件：<https://github.com/KaimingHe/deep-residual-networks>
- ▶ 编写 train.prototxt
- ▶ 编写 test.prototxt

- ▶ 测试

- ▶ 和训练类似
- ▶ 在 RPN 和 Fast R-CNN 中不需要损失函数层

# 案例研习： PY-FASTER-RCNN

## ▶ 采用新的网络结构：ResNet-50

- ▶ 下载模型配置文件：<https://github.com/KaimingHe/deep-residual-networks>
- ▶ 编写 train.prototxt
- ▶ 编写 test.prototxt

## ▶ 常见错误

- ▶ 忘记修改类别数目和边框回归的输出维数
- ▶ 修改了 RPN 的边框回归输出维数（应该保持  $4 \times 9 = 36$  不变）

## ▶ 其它

- ▶ 锁定前面几个卷积层？
- ▶ 设置 use\_global\_stats？

# 案例研习： PY-FASTER-RCNN

## ▶ 关于增加的几个 Python 层

- ▶ RoIDataLayer: 准备每个 Batch 的训练数据，对数据进行随机化
- ▶ AnchorTargetLayer: 生成参考边框的分类标签和回归目标
- ▶ ProposalLayer: 生成候选窗口（根据预测的“偏移量”计算窗口的参数）
- ▶ ProposalTargetLayer: 生成候选窗口的分类标签和回归目标

## ▶ 部分可定制代码

- ▶ 参考边框的生成: lib/rpn/generate\_anchors.py
- ▶ 模型训练接口: tools/train\_net.py
- ▶ 模型测试接口: tools/test\_net.py
- ▶ 不使用 IMDB 进行检测: tools/demo.py

## ▶ 部分可定制参数: lib/fast\_rcnn/config.py

- ▶ 输入图像大小相关: TRAIN.SCALES, TRAIN.MAX\_SIZE
- ▶ 检测分支样本 Batch 的大小: TRAIN.BATCH\_SIZE
- ▶ 训练时模型保存间隔: TRAIN.SNAPSHOT\_ITERS
- ▶ 测试时候选窗口的数目: TEST.RPN\_POST\_NMS\_TOP\_N

# 方法改进：难例挖掘 ONLINE HARD EXAMPLE MINING

## ▶ 基本思路

- ▶ 训练时所有的候选窗口都进行前向计算，以损失函数值作为难度指标，损失越大表示越难，仅挑选较难的样例进行反向传播
- ▶ 基于损失函数值对候选窗口进行非极大值抑制 NMS 操作，增大样例之间的差异性

## ▶ 实现比较

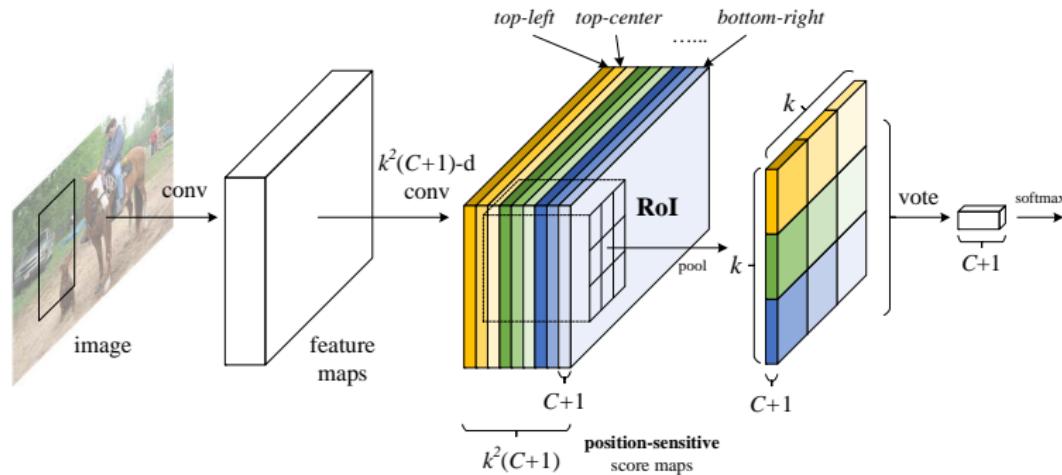
- ▶ 在损失函数层进行难例挖掘：空间复杂度高
- ▶ 两个检测分支：只读（正向），可写（反向），RoI 采样模块

Table 4: VOC 2012 test detection average precision (%). All methods use VGG16. Training set key: **12**: VOC12 trainval, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval. Legend: **M**: using multi-scale for training and testing, **B**: iterative bbox regression.

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [14]			12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
<b>Ours<sup>1</sup></b>			12	<b>69.8</b>	81.5	78.9	69.6	52.3	46.5	77.4	72.1	88.2	48.8	73.8	58.3	86.9	79.7	81.4	75.0	43.0	69.5	64.8	78.5	68.9
MR-CNN [13]	✓	✓	12	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
<b>Ours<sup>2</sup></b>	✓	✓	12	<b>72.9</b>	85.8	82.3	74.1	55.8	55.1	79.5	77.7	90.4	52.1	75.5	58.4	88.6	82.4	83.1	78.3	47.0	77.2	65.1	79.3	70.4
FRCN [14]			07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
<b>Ours<sup>3</sup></b>			07++12	<b>71.9</b>	83.0	81.3	72.5	55.6	49.0	78.9	74.7	89.5	52.3	75.0	61.0	87.9	80.9	82.4	76.3	47.1	72.5	67.3	80.6	71.2
MR-CNN [13]	✓	✓	07++12	73.9	85.5	82.9	76.6	57.8	<b>62.7</b>	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
<b>Ours<sup>4</sup></b>	✓	✓	07++12	<b>76.3</b>	<b>86.3</b>	<b>85.0</b>	<b>77.0</b>	<b>60.9</b>	59.3	<b>81.9</b>	<b>81.1</b>	<b>91.9</b>	<b>55.8</b>	<b>80.6</b>	<b>63.0</b>	<b>90.8</b>	<b>85.1</b>	<b>85.3</b>	<b>80.7</b>	<b>54.9</b>	<b>78.3</b>	<b>70.8</b>	<b>82.8</b>	<b>74.9</b>

# 方法改进：全卷积网络 R-FCN

- ▶ 动机：在 Faster R-CNN 的基础上进一步加速
- ▶ 基本思路
  - ▶ 去掉检测分支各个 RoI 之间的重复计算：将 RoIPooling 后移，继续在全图上卷积
  - ▶ 位置敏感得分图 POSITION SENSITIVE SCORE MAP



R-FCN: Object Detection via Region-based Fully Convolutional Networks. arXiv 1605.06409.

# 其它思路：利用不同层的特征

## ▶ 动机

- ▶ 浅层特征：保留了局部细节，但是抽象程度不够，不利于分类
- ▶ 深层特征：神经元感受野大，特征比较粗略，不利于边框的准确定位
- ▶ 物体存在尺度变化，不同尺度的物体不应该用相同尺度上的特征

## ▶ 基本思路

- ▶ 将浅层和深层特征组合使用：Hypercolumns, HyperNet
- ▶ 不同大小的 ROI 用不同层的特征：尺度相关池化 **SCALE-DEPENDENT POOLING**

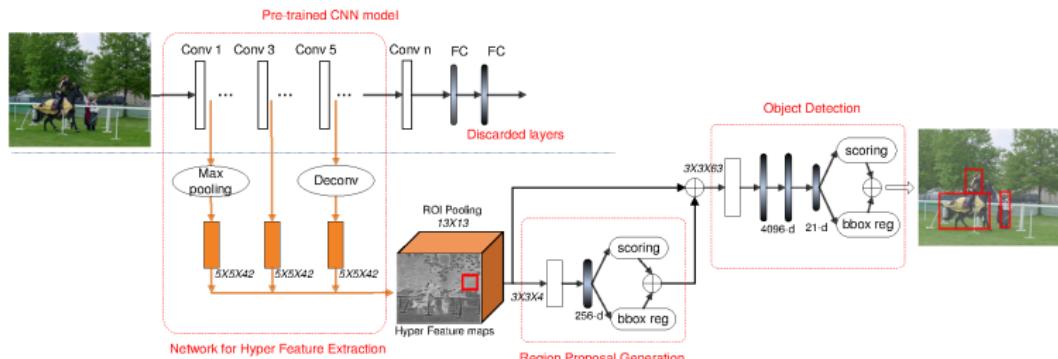
# 其它思路：利用不同层的特征

## ► 动机

- ▶ 浅层特征：保留了局部细节，但是抽象程度不够，不利于分类
- ▶ 深层特征：神经元感受野大，特征比较粗略，不利于边框的准确定位
- ▶ 物体存在尺度变化，不同尺度的物体不应该用相同尺度上的特征

## ► 基本思路

- ▶ 将浅层和深层特征组合使用：Hypercolumns, **HYPERNET**
- ▶ 不同大小的 ROI 用不同层的特征：尺度相关池化 **SCALE-DEPENDENT POOLING**



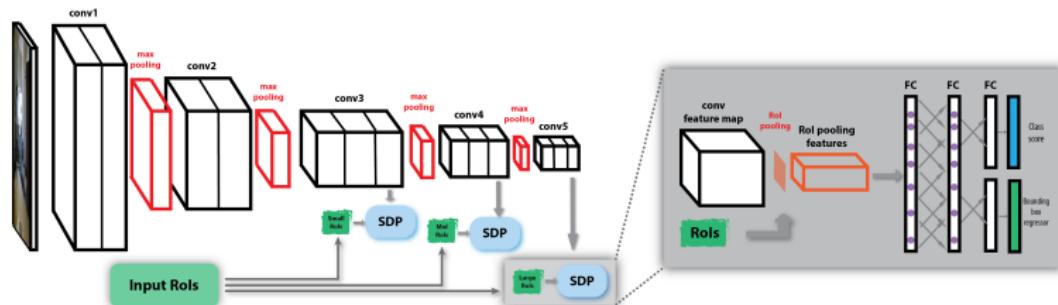
# 其它思路：利用不同层的特征

## ▶ 动机

- ▶ 浅层特征：保留了局部细节，但是抽象程度不够，不利于分类
- ▶ 深层特征：神经元感受野大，特征比较粗略，不利于边框的准确定位
- ▶ 物体存在尺度变化，不同尺度的物体不应该用相同尺度上的特征

## ▶ 基本思路

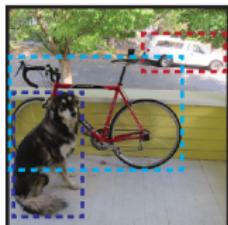
- ▶ 将浅层和深层特征组合使用：Hypercolumns, HyperNet
- ▶ 不同大小的 RoI 用不同层的特征：**尺度相关池化 SCALE-DEPENDENT POOLING**



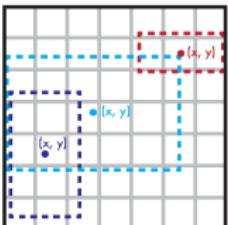
Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers. CVPR 2016.

# 其它思路

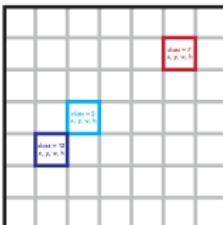
- ▶ 引入上下文信息
  - ▶ 与图像分割任务相结合: Multi-region CNN, segDeepM
  - ▶ Inside-Outside Net
- ▶ 网格式搜索
  - ▶ YOLO (同时也考虑了全图上下文信息)
  - ▶ G-CNN
- ▶ 回归目标热图
  - ▶ DenseBox



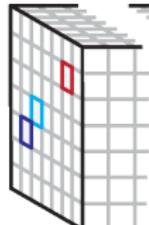
**Resize The Image**  
And bounding boxes to 448 x 448.



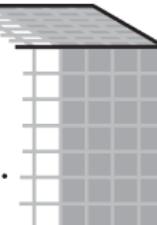
**Divide The Image**  
Into a  $7 \times 7$  grid. Assign detections to grid cells based on their centers.



**Train The Network**  
To predict this grid of class probabilities  
and bounding box coordinates.



**1st - 20th Channels:**  
Class probabilities  
 $\Pr(\text{Airplane}), \Pr(\text{Bike})\dots$



**Last 4 Channels:**  
Box coordinates  
 $x, y, w, h$

## 级联结构目标检测器

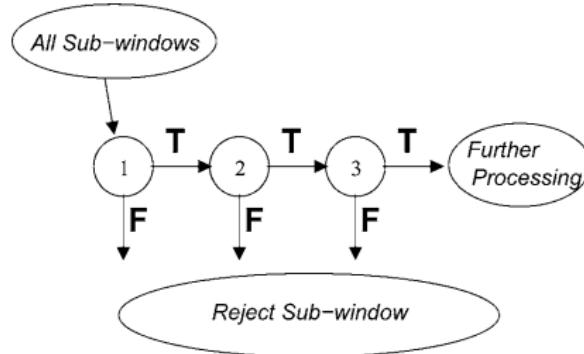
# 两条路线

## ▶ 从更准到更快！

- ▶ 基于深度学习的 R-CNN 系列检测器带来了检测精度的巨大提升
- ▶ R-CNN → **FAST** R-CNN → **FASTER** R-CNN → R-FCN
- ▶ 不断思考速度怎么怎样才能更快！

## ▶ 从更快到更准？

- ▶ 传统的 AdaBoost 级联结构检测器
- ▶ 能不能将深度模型和传统架构相结合？



# CASCADE CNN

## ▶ 特定类别目标的检测：人脸

- ▶ 人脸/人体/车辆检测相比通用目标检测在速度上具有更高的要求
- ▶ 考虑特定目标本身的特点
- ▶ 从单个类别扩展到多个类别通常比较繁琐

## ▶ 基本思路

- ▶ 人脸复杂的表观变化要求使用容量更大的模型 → CNN
- ▶ 实际应用场景具有实时性要求 →
  - ▶ 减小单个 CNN 的复杂度，多个级联
  - ▶ 加大滑动步长，减小输入窗口规模 → 窗口不够准？增加校准模块

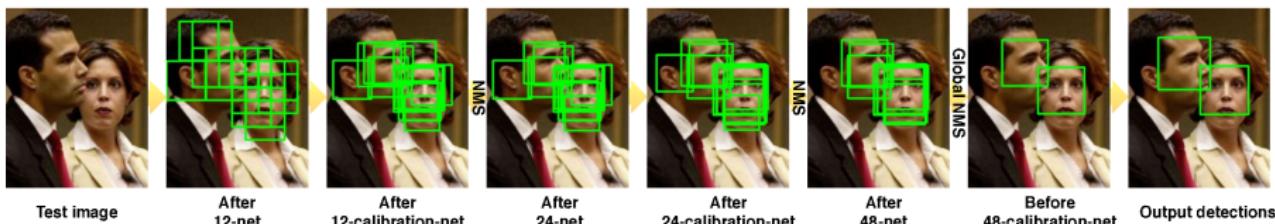
# CASCADE CNN

## ▶ 特定类别目标的检测：人脸

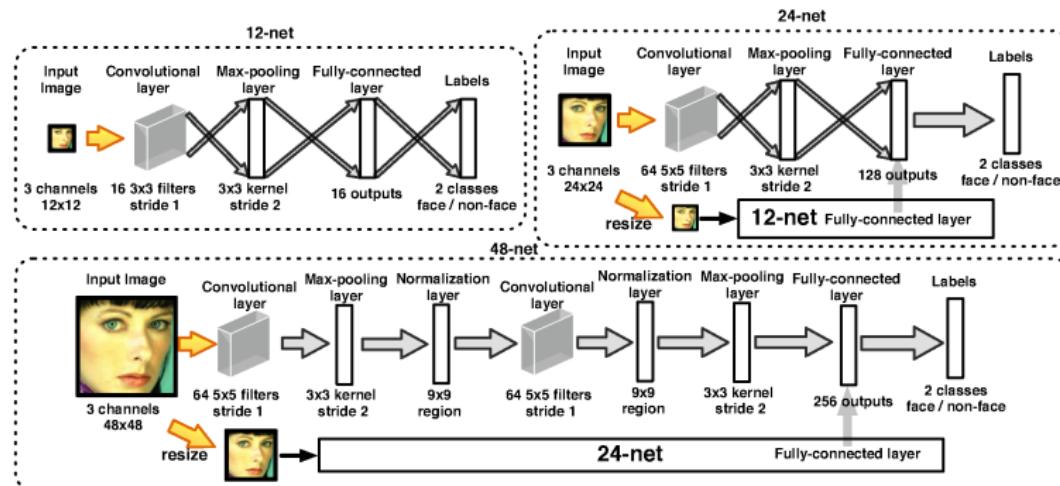
- ▶ 人脸/人体/车辆检测相比通用目标检测在速度上具有更高的要求
- ▶ 考虑特定目标本身的特点
- ▶ 从单个类别扩展到多个类别通常比较繁琐

## ▶ 基本思路

- ▶ 人脸复杂的表观变化要求使用容量更大的模型 → CNN
- ▶ 实际应用场景具有实时性要求 →
  - ▶ 减小单个 CNN 的复杂度，多个级联
  - ▶ 加大滑动步长，减小输入窗口规模 → 窗口不够准？增加校准模块



# CASCADE CNN: 窗口分类网络



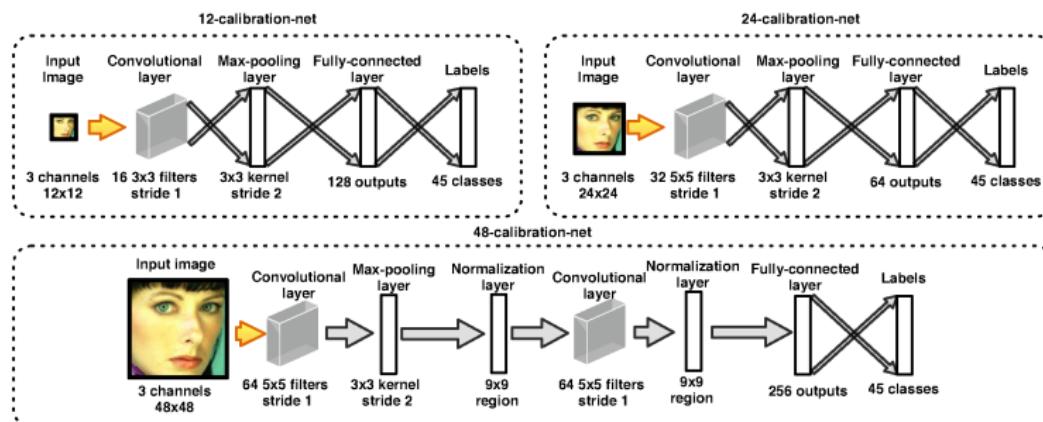
## ▶ 12-net

- ▶ 滑动窗口，步长 = 4
- ▶ 输入图像  $800 \times 600$ ，最小人脸  $40 \times 40$ ，共产生 2494 个候选窗口，CPU 36ms/图

## ▶ 24-net 和 48-net

- ▶ 输入逐步增大，网络逐步变复杂
- ▶ 拼接相邻两级分类网络的全连接层

# CASCADE CNN：边框校准网络



## ► 边框校准

- ▶ 预测偏移量:  $\left( x - \frac{x_n w}{s_n}, y - \frac{y_n h}{s_n}, \frac{w}{s_n}, \frac{h}{s_n} \right)$
- ▶ 设计为 45 类的（**多类**）分类任务
  - ▶  $s_n \in \{0.83, 0.91, 1.0, 1.10, 1.21\}$
  - ▶  $x_n \in \{-0.17, 0, 0.17\}$
  - ▶  $y_n \in \{-0.17, 0, 0.17\}$

# CASCADE CNN

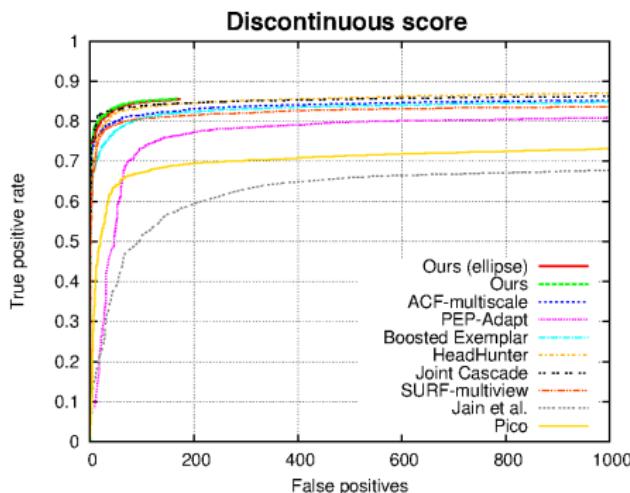
## ▶ 精度

- ▶ 评测集：FDDB，2845 张测试图像，5171 张人脸
- ▶ 在产生 100 个误检时，召回率达到 85%

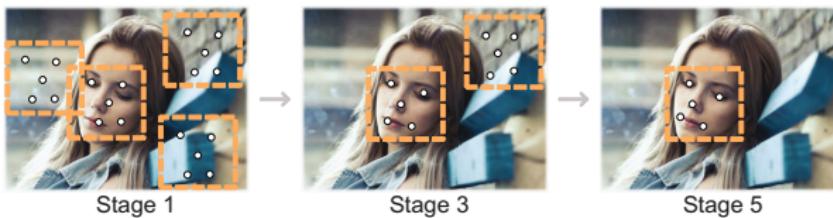
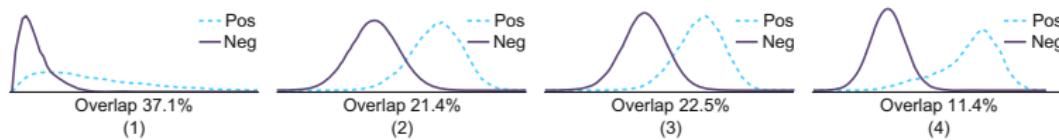
## ▶ 速度

- ▶ 测试条件：图像大小  $640 \times 480$ ，最小人脸  $80 \times 80$ ，图像金字塔缩放系数 1.414
- ▶ CPU 110ms/图，GPU 10ms/图

Stage	# windows	Recall
sliding window	5341.8	95.9%
12-net	426.9	93.9%
12-calibration-net	388.7	94.8%
24-net	60.5	88.8%
24-calibration-net	53.6	89.0%
48-net	33.3	85.8%
global NMS	3.6	82.1%
48-calibration-net	3.6	85.1%

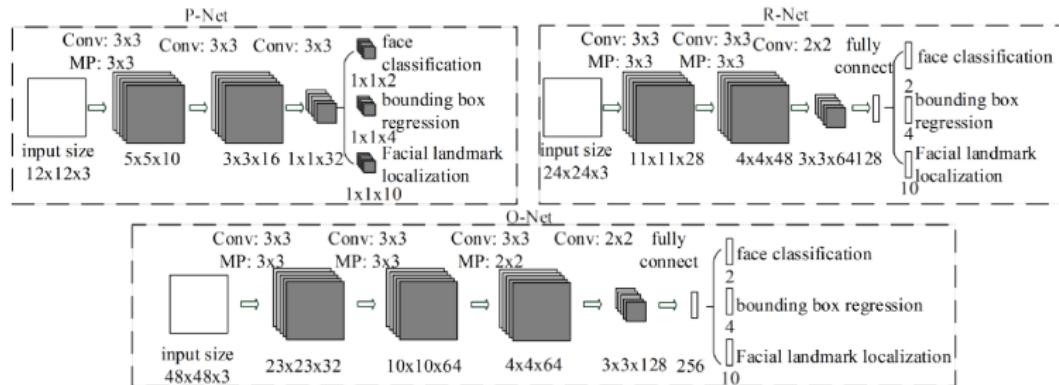


# 番外：JOINT CASCADE



- ▶ 特征的语义一致性
  - ▶ 不同姿态的人脸，相同的部位位于不同的位置
- ▶ 多任务学习
  - ▶ 相关任务之间可以相互促进

# MT-CNN



## ▶ 多任务学习

- ▶ 人脸/非人脸分类，边框回归，特征点定位（5 点）
- ▶ 没有单独的边框校准网络和特征点定位模型

## ▶ 第一级采用全卷积网络生成候选窗口

- ▶ 联想：RPN? YOLO?

# MT-CNN

## ▶ 精度

- ▶ 评测集：FDDB，2845 张测试图像，5171 张人脸
- ▶ 在产生 100 个误检时，召回率达到 90.83%，明显好于 Cascade CNN

## ▶ 速度

- ▶ 测试条件：图像大小  $640 \times 480$
- ▶ CPU 16fps，GPU 99fps，稍快于 Cascade CNN

Group	CNN	300 Times Forward	Accuracy
Group1	12-Net [19]	0.038s	94.4%
Group1	P-Net	0.031s	94.6%
Group2	24-Net [19]	0.738s	95.1%
Group2	R-Net	0.458s	95.4%
Group3	48-Net [19]	3.577s	93.2%
Group3	O-Net	1.347s	95.4%

## ▶ 窗口数量影响速度

- ▶ 窗口越多，速度越慢
- ▶ 人脸越多，窗口越多
- ▶ 同一个人脸周围窗口越少，分类器精度要求越高

# 级联结构的联合训练

## ► 级联结构的问题

- ▶ 训练和调优非常、非常、非常繁琐 -\_-#
- ▶ 关联性太强：牵一发而动全身，改变其中一级，理论上后面各级都要重新训练
- ▶ 关联性太弱：后一级训练时前一级固定不变，前一级的信息不能被后一级的信息所使用 → 例外：软级联，链式 Boosting，嵌套式 Boosting，特征继承

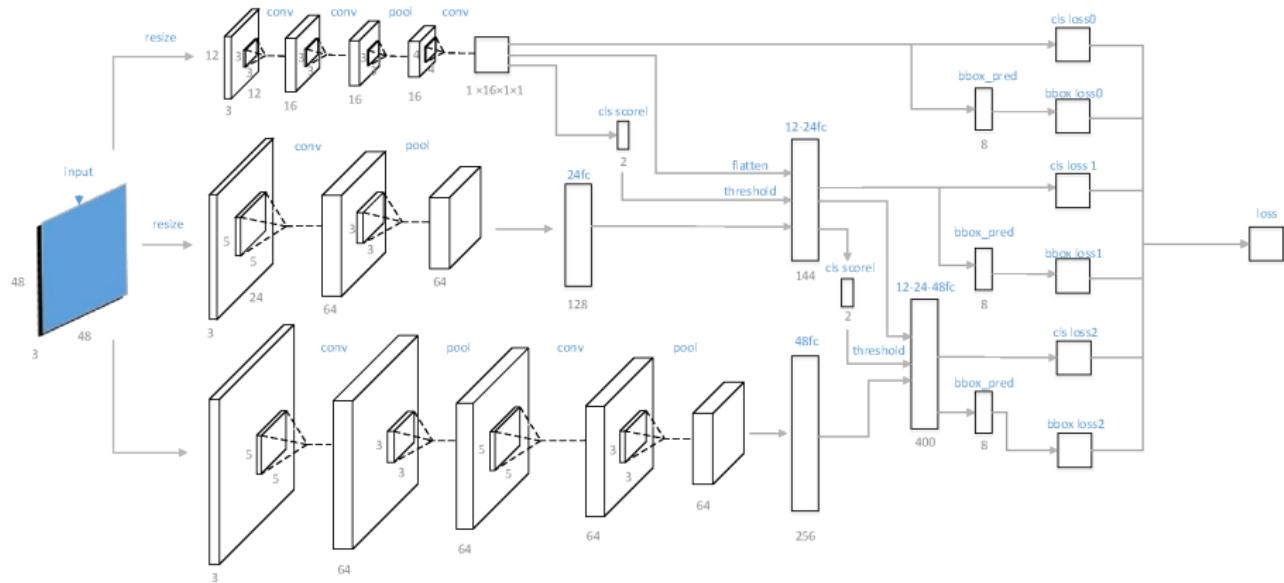
## ► 回顾 Cascade CNN

- ▶ 24-Net 在分类时使用了 12-Net 的全连接层输出，48-Net 也采用了类似的做法
- ▶ 不同级之间的关联性能否被利用起来？

## ► 回顾 Faster R-CNN

- ▶ RPN 和 Fast R-CNN 具有共享的卷积层（在不共享的情况下，两者也可以看成是构成一种级联式结构）
- ▶ 原文给出了一种近似的端到端优化方式

# 级联结构的联合训练



## ► 精度

- FDDB, 1000 个误检时, 召回率为 88.2%/87.3%/90.8%

## ► 速度

- 图像大小  $640 \times 480$ , 最小人脸  $24 \times 24$ , CPU 10fps

# 比较： R-CNN vs CASCADE CNN

- ▶ R-CNN 和 Cascade CNN 各有什么特点？优劣何在？
  - ▶ 我：这一页没时间做了，大家回家自己想想吧 =\_=
  - ▶ 台下：开什么玩笑？！我们大老远跑来你就把话说一半？ -\_-#
  - ▶ 我：看在刘大大昨天讲了这么多的份上今天就放过我吧 T\_T
  - ▶ 台下：刘大大呢？
  - ▶ 刘大大： .....

## 案例研习：人脸检测

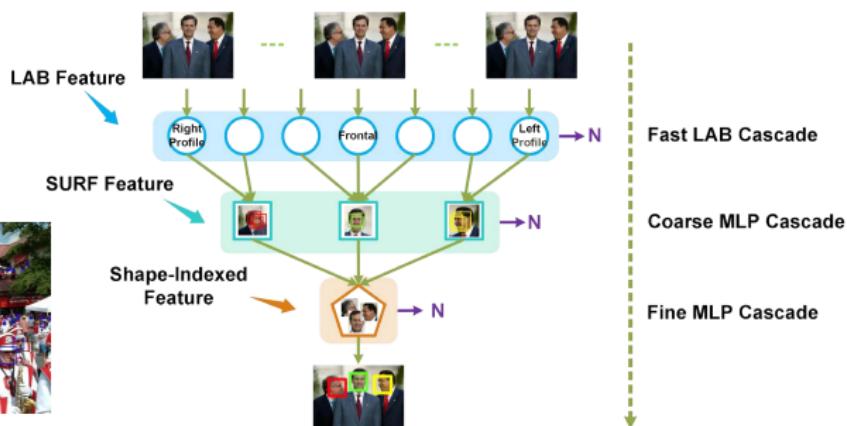
# 案例研习：人脸检测

## ▶ 两类检测器

- ▶ 漏斗型级联结构多姿态人脸检测器 **FuSt DETECTOR** ——讲者的微小工作
- ▶ Faster R-CNN + ResNet-50

## ▶ 数据集

- ▶ 训练集：WIDER FACE (好可怕！)
- ▶ 测试集：FDDB

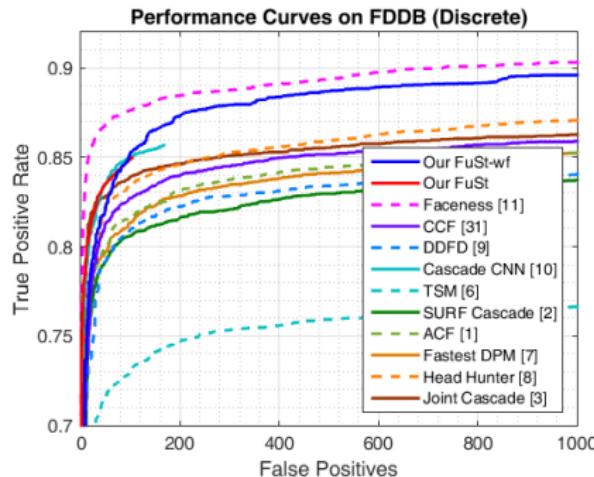


Funnel-Structured Cascade for Multi-View Face Detection with Alignment Awareness. 2016.  
[\(Available soon\)](#)

# 漏斗型级联结构的检测器

## ▶ 基本想法

- ▶ 特征和模型从简单到复杂
- ▶ 采用汇聚式的窗口处理流程
- ▶ 考虑特征的语义一致性



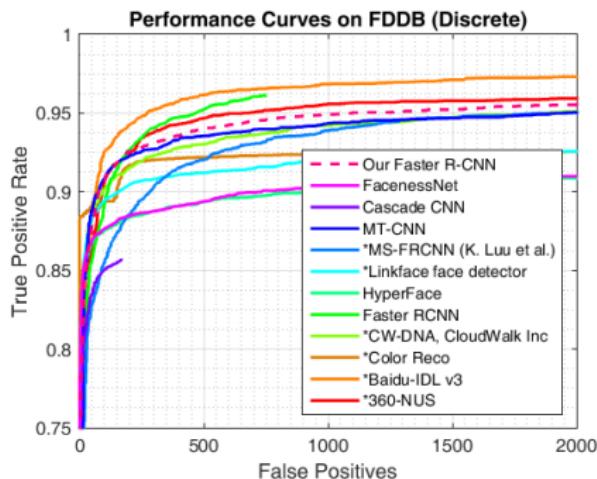
## ▶ 性能

- ▶ 精度：目前最好的结果是 FDDB 上产生 100 个误检时召回率 85.3%，1000 个误检时召回率 89.6%
- ▶ 速度：相比 Cascade CNN 汇报的结果，在相同测试条件下快一倍
- ▶ 讲者还在继续努力！

# 基于 FASTER R-CNN 的人脸检测器

## ▶ 实验设置

- ▶ 网络结构: ResNet-50
- ▶ 训练集: WIDER FACE (train), AFLW
- ▶ 默认参数: 单尺度, 300 个候选窗口/图



## ▶ 性能

- ▶ 精度: FDDB 上产生 100 个误检时召回率 90.6%, 1000 个误检时召回率 94.9%
- ▶ 速度: GPU 4 ~ 5fps
- ▶ 讲者还在继续努力!

# 目标检测与弱监督学习

# 现实困境

## ▶ 对大量数据的需求

- ▶ 数据太少深度网络学不好，容易过拟合
- ▶ 小模型也能从大数据中受益
- ▶ 模型训练大都采用监督学习方法，需要人工标注

## ▶ 数据收集和标注的困难

- ▶ 人力，物力，财力
- ▶ 难以贴近真实应用场景，或者存在隐私和版权问题
- ▶ 不同人标注的不一致性，多次检查增加工作量
- ▶ 增加新的类别需要增加新的标注
- ▶ .....

## ▶ 研究者们的思考

- ▶ 无监督，半监督，**弱监督**
- ▶ 主动学习 **ACTIVE LEARNING**，自步学习 **SELF-PACED LEARNING**
- ▶ Zero-Shot, One-Shot, Low-Shot

# 目标检测与弱监督学习

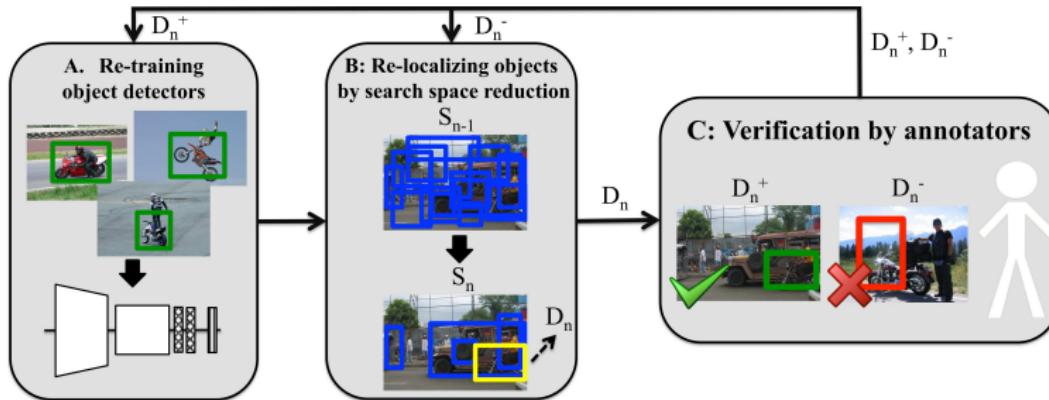
## ▶ 基本思路

- ▶ 用容易获得的标注替代较难获得的标注：图像类别 vs 物体边框
- ▶ 选择最需要做精细标注的样例
- ▶ 模型训练和自动标注交替进行

## ▶ 问题

- ▶ 相比于用完全的监督信息训练的检测器，基于弱监督信息训练的检测器往往精度只到其一半
- ▶ 仍然需要标注部分物体边框

# 目标检测与弱监督学习



## ▶ 基于验证信号进行学习

- ▶ 通过候选窗口生成方法产生窗口
- ▶ 仅把验证信号确定加入训练集的新样例
- ▶ 基于负样例验证信号缩小搜索空间



# 谢谢！

邬书哲

微信：ptsntwsz

邮箱：shuzhe.wu@vipl.ict.ac.cn