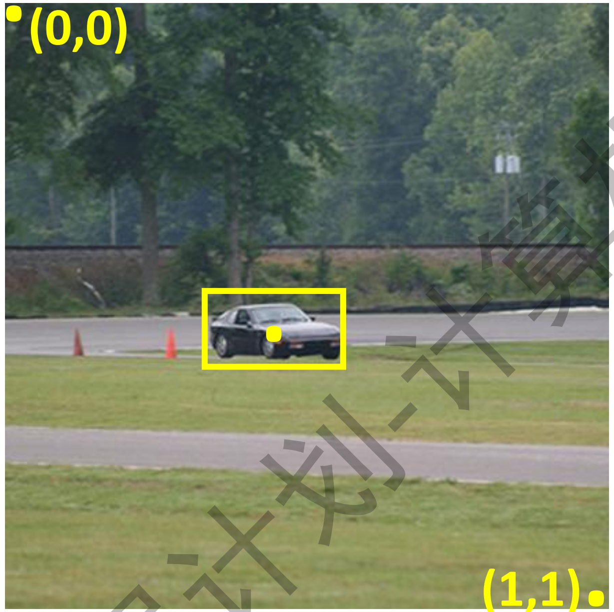
视觉探测任务

探测任务简介

初识探测任务

分类（Classification ）：
对图像中仅有的主体分类
Category：car



定位（Localization）：
对图像中仅有的主体分类，并给出位置
Category: car

Bounding box: (b_x, b_y, b_w, b_h)
(0.43, 0.57, 0.24, 0.15)

探测（Detection）：
多个Category与Bounding box组合

Car-1:(0.42, 0.67, 0.5, 0.13)

Car-2:(0.55, 0.43, 0.22,0.09)

Person-1:(0.45, 0.66, 0.07, 0.07)

silding windows & IoU

是否能将
未知转化为已知？

我们已经
掌握了分类问题

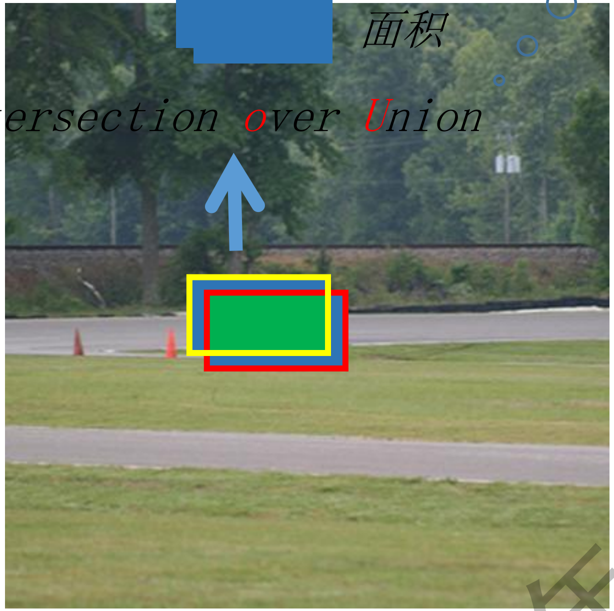
能否将探测问题
转化为分类问题？

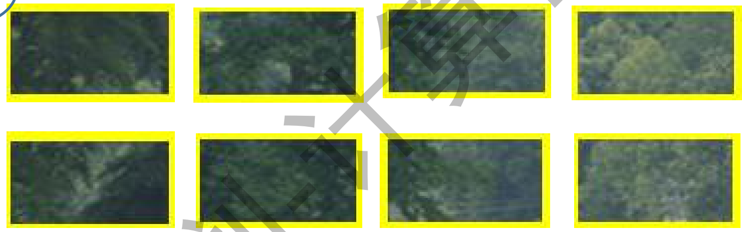
滑窗（sliding windows）

$IoU = \frac{\text{面积}}{\text{面积}}$

重叠率越高
IoU越接近1

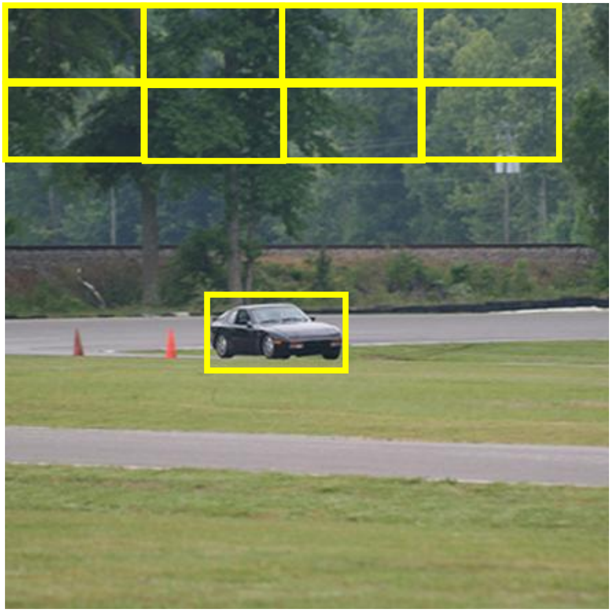
Intersection over Union





红色BB代表正确答案（ground truth）

黄色BB代表预测结果



除非步长以及bounding box尺寸选的十分合适
否则很难恰巧将目标“框住”，怎样评价BB的好坏？
重合率越高越好

滑窗：背景类丢弃，目标类输出

滑窗：把原图裁成许多小块
送入分类网络

Region Proposal

滑窗方法的本质是提出一些
候选区域（Region Proposal）用于分类

早期的深度探测
网络如**Over Feat**
便采用滑窗方法

这种方法
有啥问题？

选多大
的窗口？

选择几种
窗口？

步长是多
少？

……

为了能遍历图像
上的所有目标，
可能要生成数以
万计的候选框

除了滑窗
还有什么方法？

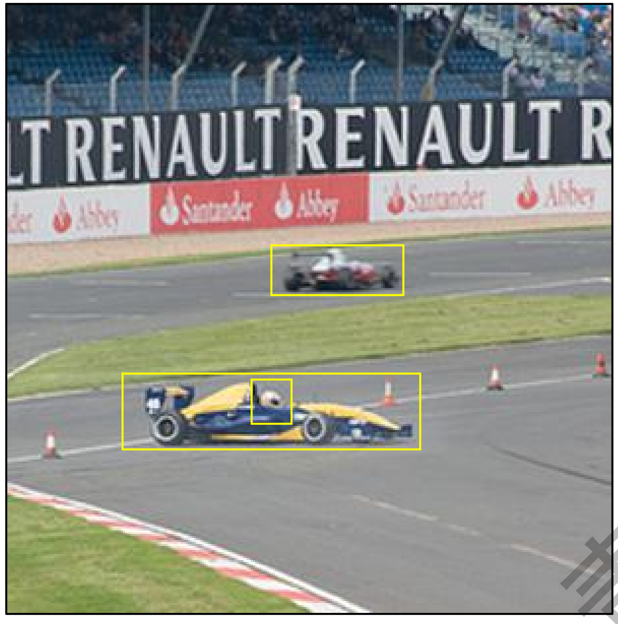
Selective Search
(SS)

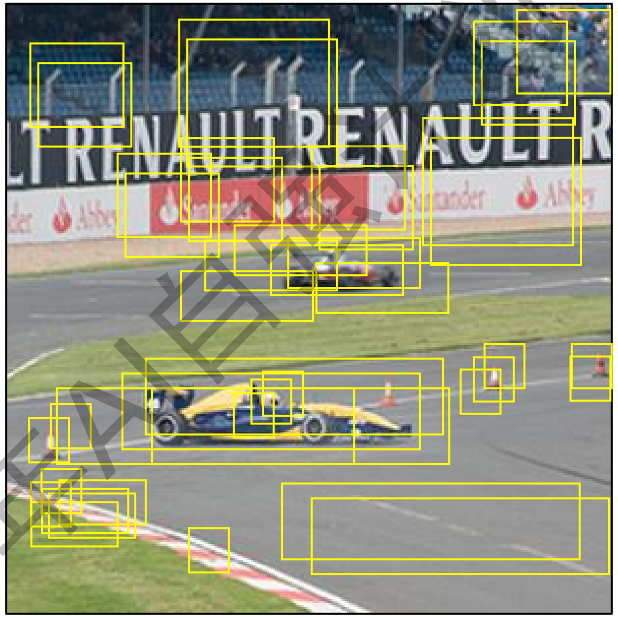
这是一个传统机器视觉算法
(按照特定规则提取候选框)

这种方法
无需训练

1.运行传统分割算法
2.提取很多初始候选框
3.按照相似度合并候选框
4.最终留下固定数量的候选框

并不严谨的
示例



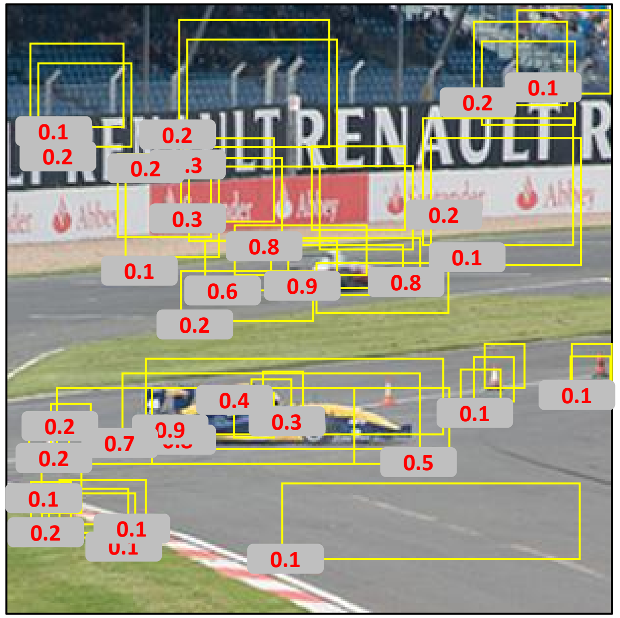


最终留下候选框的具体数量是超参
一般为2000个

Non-Max Suppression （NMS）

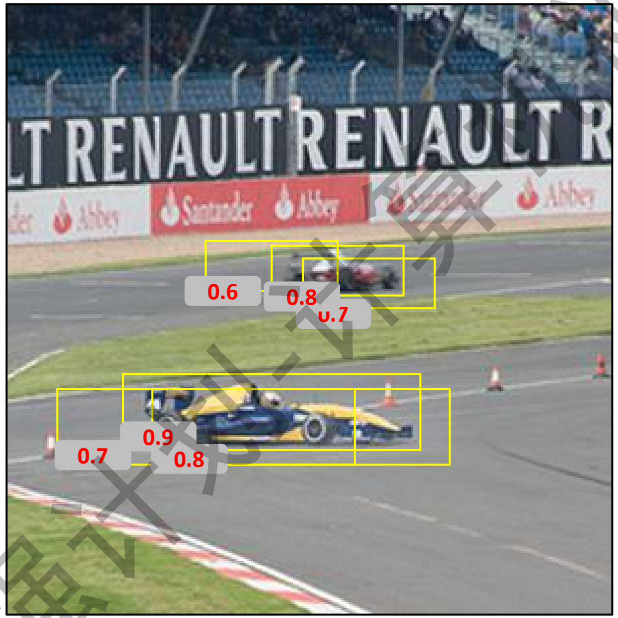
清華大學 数据科学研究院 Tsinghua University Institute for Data Science

候选框还是太多了怎么办？
非极大值抑制
（除了最大的都不要）



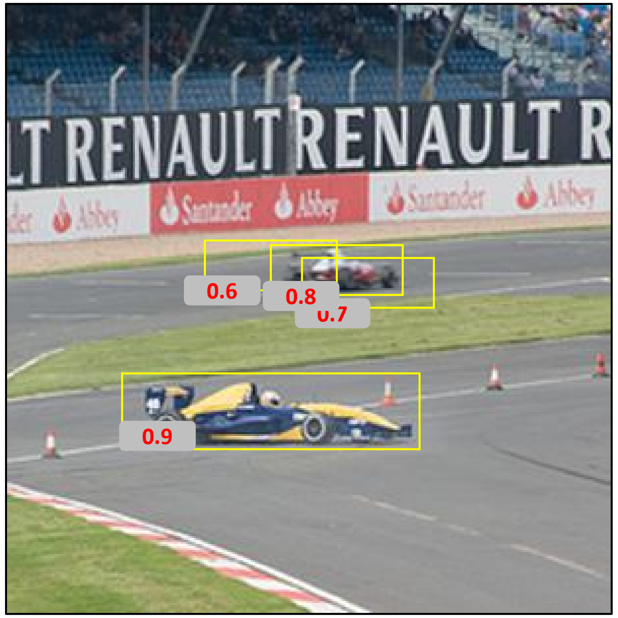
逐类粗筛

不够大的都不要



细筛

除了最大的都不要



图上仅标出了40余个候选框作为示意图
实际情况(2000个)要比示意图复杂得多

NMS需要逐类进行，此处先对类别“汽车”进行NMS
将含汽车概率小于一定阈值（如小于0.5）的候选框丢弃

再次挑出概率最大的候选框循环往复
直至所有粗筛后的候选框都被处理过一遍

我们可以将每个候选框送去分类
不同的类别逐个处理
此处仅标出了候选框内含有“汽车”的概率

挑出概率最大的候选框
将所有与之IoU大于一定阈值（如大于0.7）的候选框都丢弃
（最大值周围的都不要）

此类别处理完成，转而处理下一个类别

探测任务–网络发展概览

R-CNN (2013.11) → **OverFeat** (ICLR' 14) → MultiBox (CVPR' 14) → SPP-Net (ECCV' 14) → MR-CNN (ICCV' 15) → DeepBox (ICCV' 15) → AttentionNet (ICCV' 15) →

Fast R-CNN (ICCV' 15) → DeepProposal (ICCV' 15) → **RPN** (NIPS' 15) → **Faster R-CNN** (NIPS' 15) → **YOLO v1** (CVPR' 16) → G-CNN (CVPR' 16) → AZNet (CVPR' 16) →

Inside-OutsideNet(ION) (CVPR' 16) → HyperNet (CVPR' 16) → OHEM (CVPR' 16) → CRAFT (CVPR' 16) → MultiPathNet(MPN) (BMVC' 16) → **SSD** (ECCV' 16) →

GBDNet (ECCV' 16) → CPF (ECCV' 16) → MS-CNN (ECCV' 16) → R-FCN (NIPS' 16) → PVANET (NIPSW' 16) → DeepID-Net (PAMI' 16) → NoC (TPAMI' 16) → DSSD (Arxiv' 17) → TDM (CVPR' 17) →

Feature Pyramid Net(**FPN**) (CVPR' 17) → **YOLO v2** (CVPR' 17) → RON (CVPR' 17) → DCN (ICCV' 17) → DeNet (ICCV' 17) → CoupleNet (ICCV' 17) → **RetinaNet** (ICCV' 17) →

Mask R-CNN (ICCV' 17) → DSOD (ICCV' 17) → SMN (ICCV' 17) → **YOLO v3** (Arxiv' 18) → SIN (CVPR' 18) → STDN (CVPR' 18) → RefineDet (CVPR' 18) → RFBNet (ECCV' 18) → ···

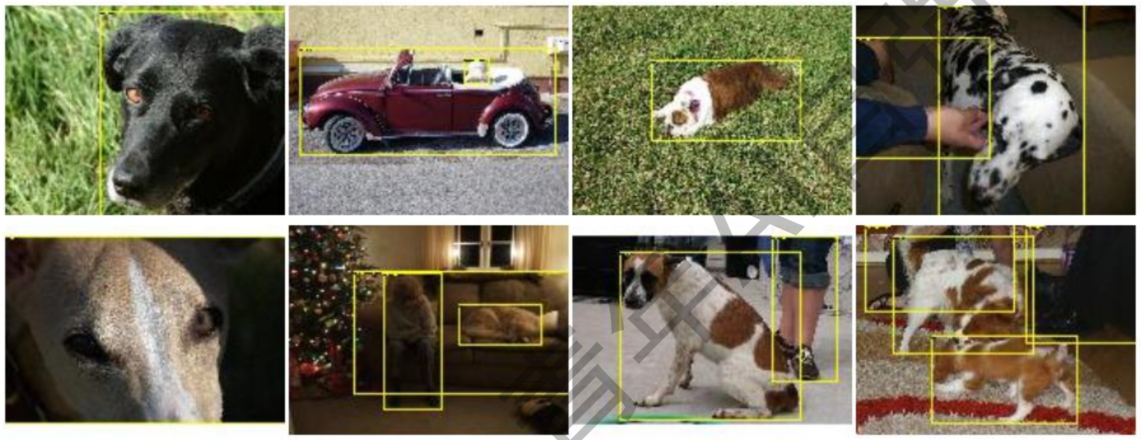
网络介绍思路

Two-stage VS end to end

PASCAL VOC 数据集简介

Birds - ***all images contain at least one bird.***

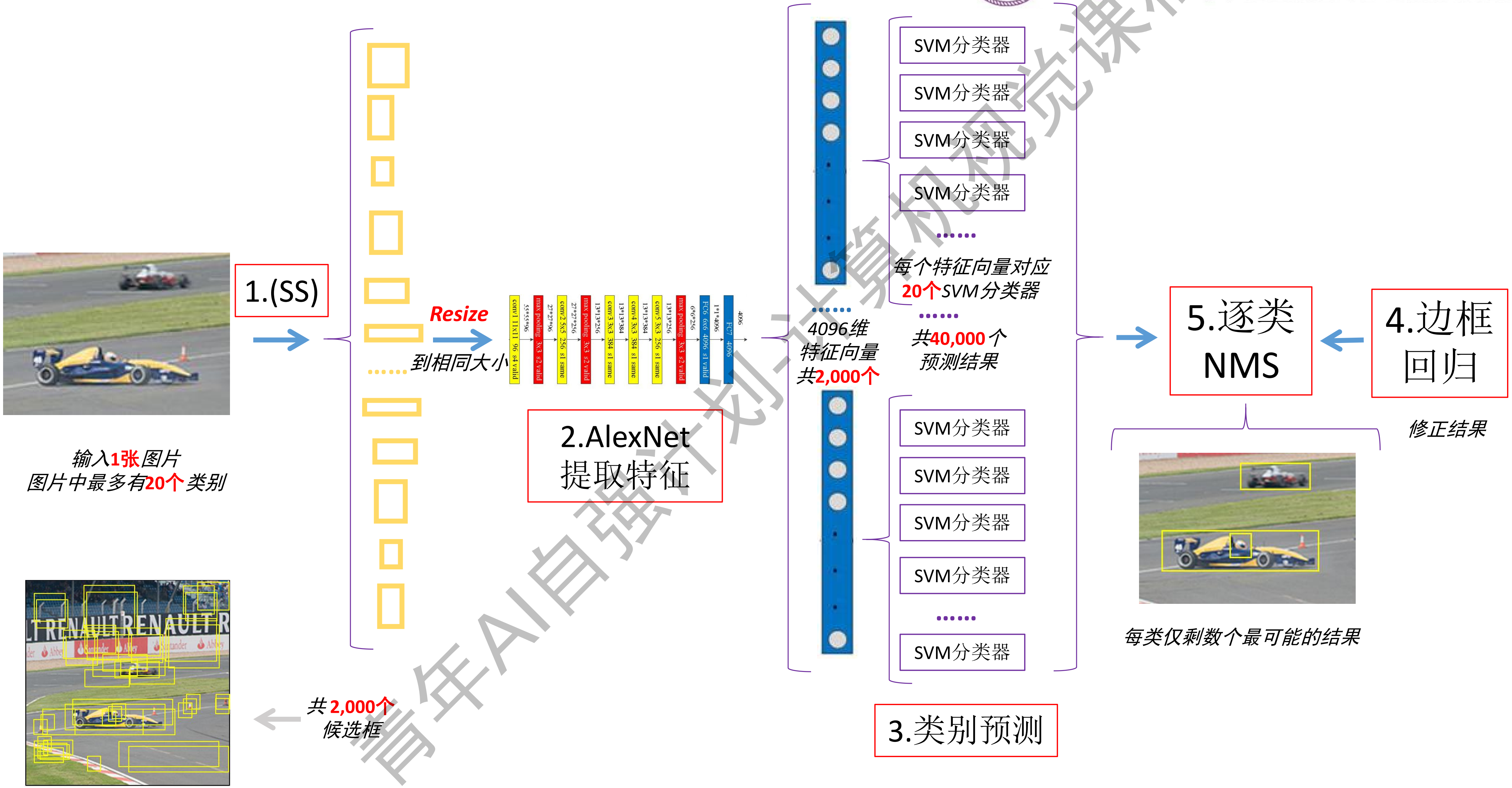
Dogs - ***all images contain at least one dog.***



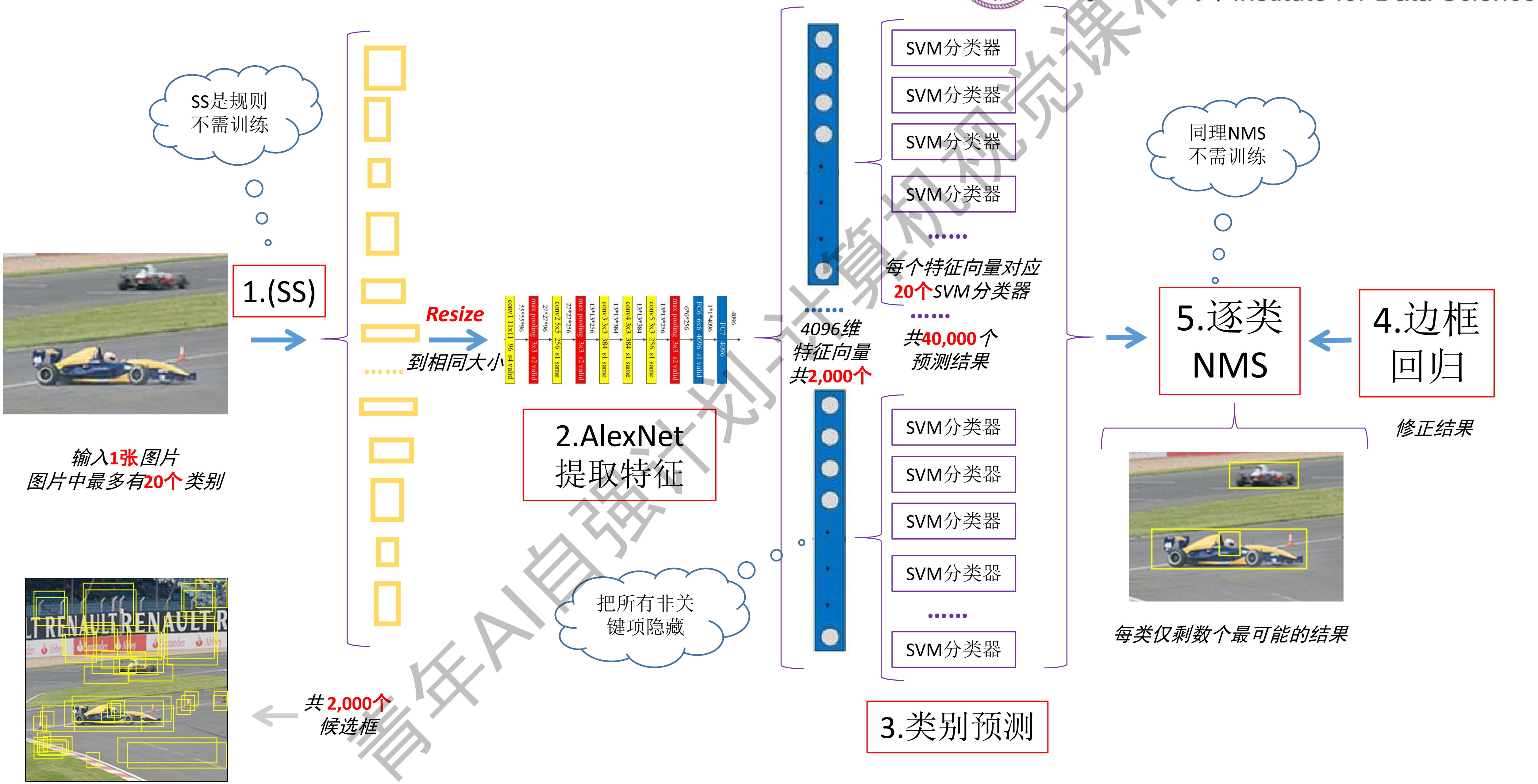
Comparative scale

		PASCAL VOC 2012	ILSVRC 2014
Number of object classes		20	**200**
Training	Num images	5717	456567
	Num objects	13609	478807
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

R-CNN(Region)-前馈工作流

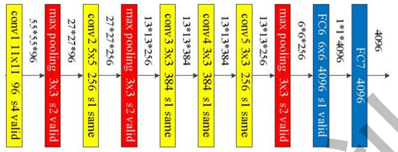


R-CNN(Region)-如何训练



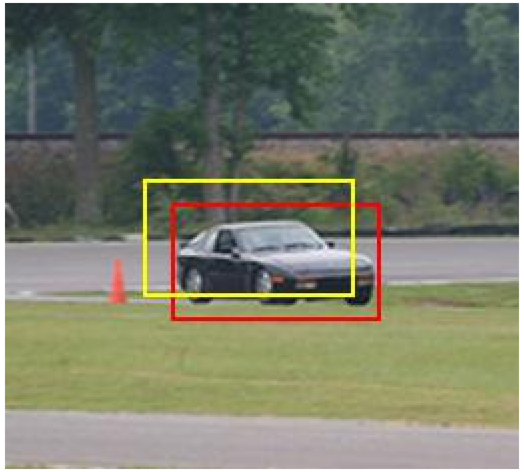
R-CNN(Region)-如何训练

1.此处可以替换为任何CNN分类网络
2.此处的CNN已经预训练完成
3. SS生成的所有候选框，挑出IoU大于0.7的，作为正样本训练集对网络微调



2.AlexNet
提取特征

1.采用线性回归模型
2.X为预测结果（图中黄色部分）
3.Y为ground truth（图中黄色部分）
4.训练的本质是找到一个线性关系，对X进行平移和缩放，以期能够更加接近Y



4.边框
回归

修正结果

1.此处SVM为二分类的分类器
2.此处的CNN已经预训练完成
3.将ground truth为正样本，SS生成的所有候选框，小于0.3的作为负样本对SVM训练

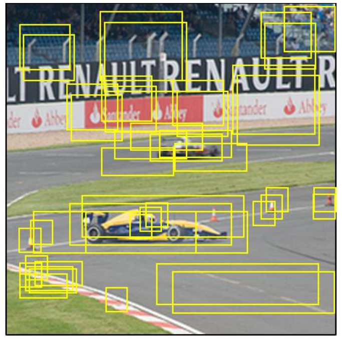
SVM分类器

3.类别预测

R-CNN(Region)-开山之作的不足

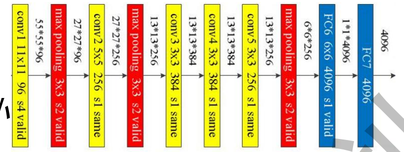
输入1张图片
图片中最多有20个类别

1.(SS)



共2,000个
候选框

Resize
到相同大小



2.AlexNet
提取特征

1个样本需要缓存
2,000个4,096维向量；
训练集1万个样本需要
缓存多少过程数据？

4096维
特征向量
共2,000个

SVM分类器
SVM分类器
SVM分类器
SVM分类器
......

每个特征向量对应
20个SVM分类器
......
共40,000个
预测结果

SVM分类器
SVM分类器
SVM分类器
SVM分类器
......
SVM分类器

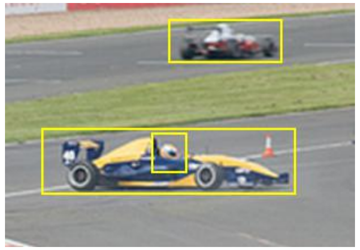
3.分类

类别预测与边框（bbox）
预测相互独立，效率较低？

5.逐类
NMS

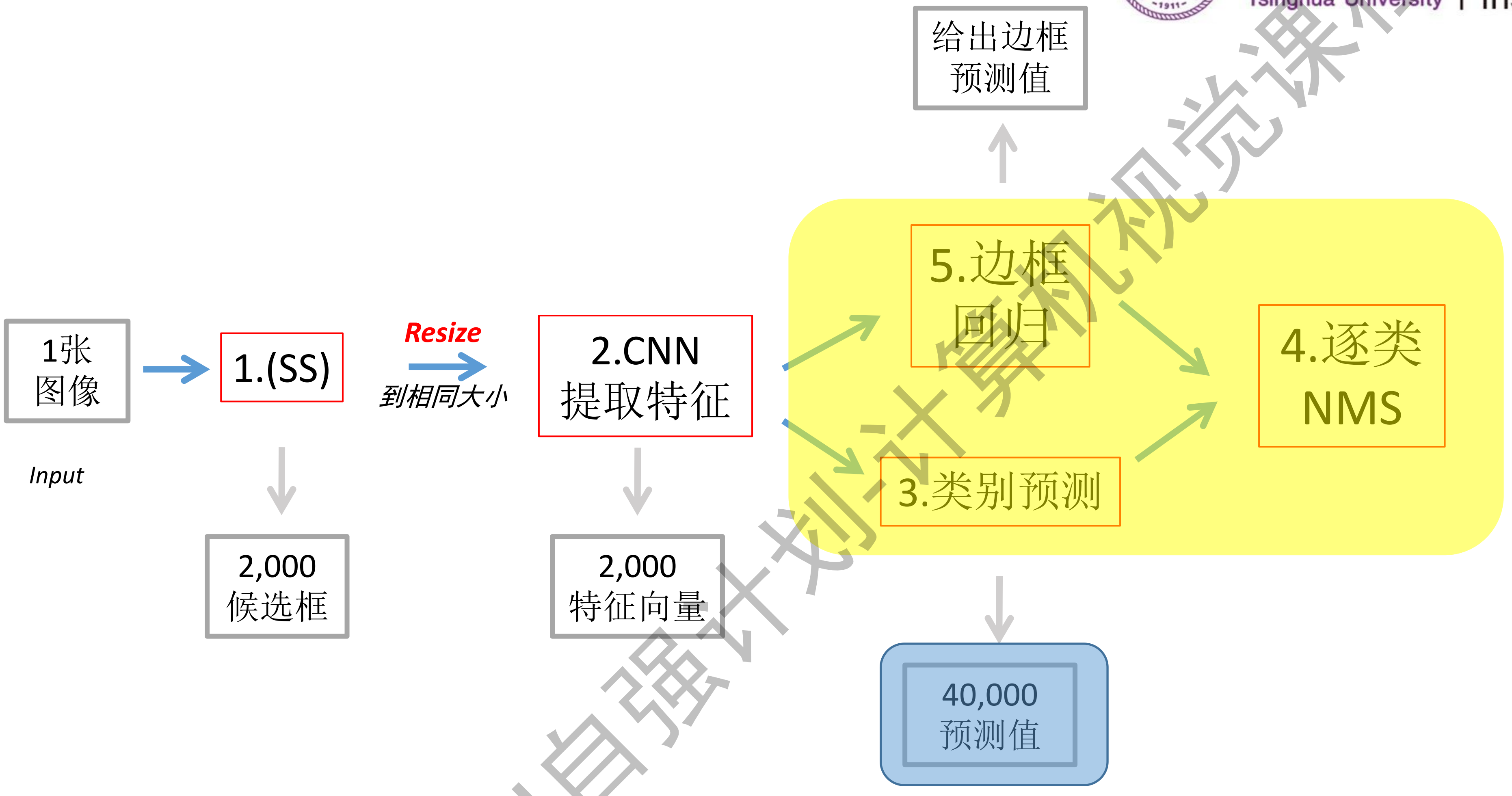
4.边框
回归

修正结果



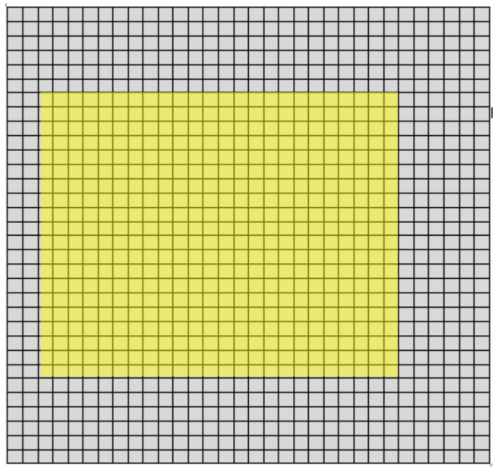
每类仅剩数个最可能的结果

R-CNN(Region)-开山之作的不足



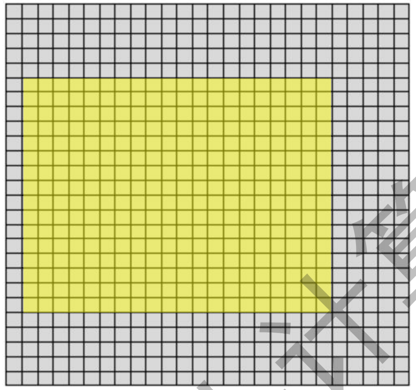
Fast R-CNN　区域映射

中间运算结果缓存太烦，怎么破？



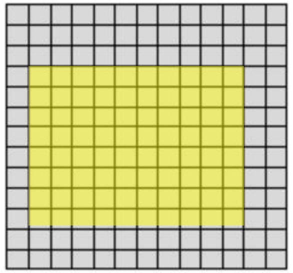
32 × 32 × 3

Conv*16 $\xrightarrow{\quad}$ 7 × 7



26 × 26 × 16

MAX POOL(s=2) $\xrightarrow{\quad}$ 2 × 2



13 × 13 × 16

输入图像

1.R-CNN在原图片上以候选框截取图像，送入网络

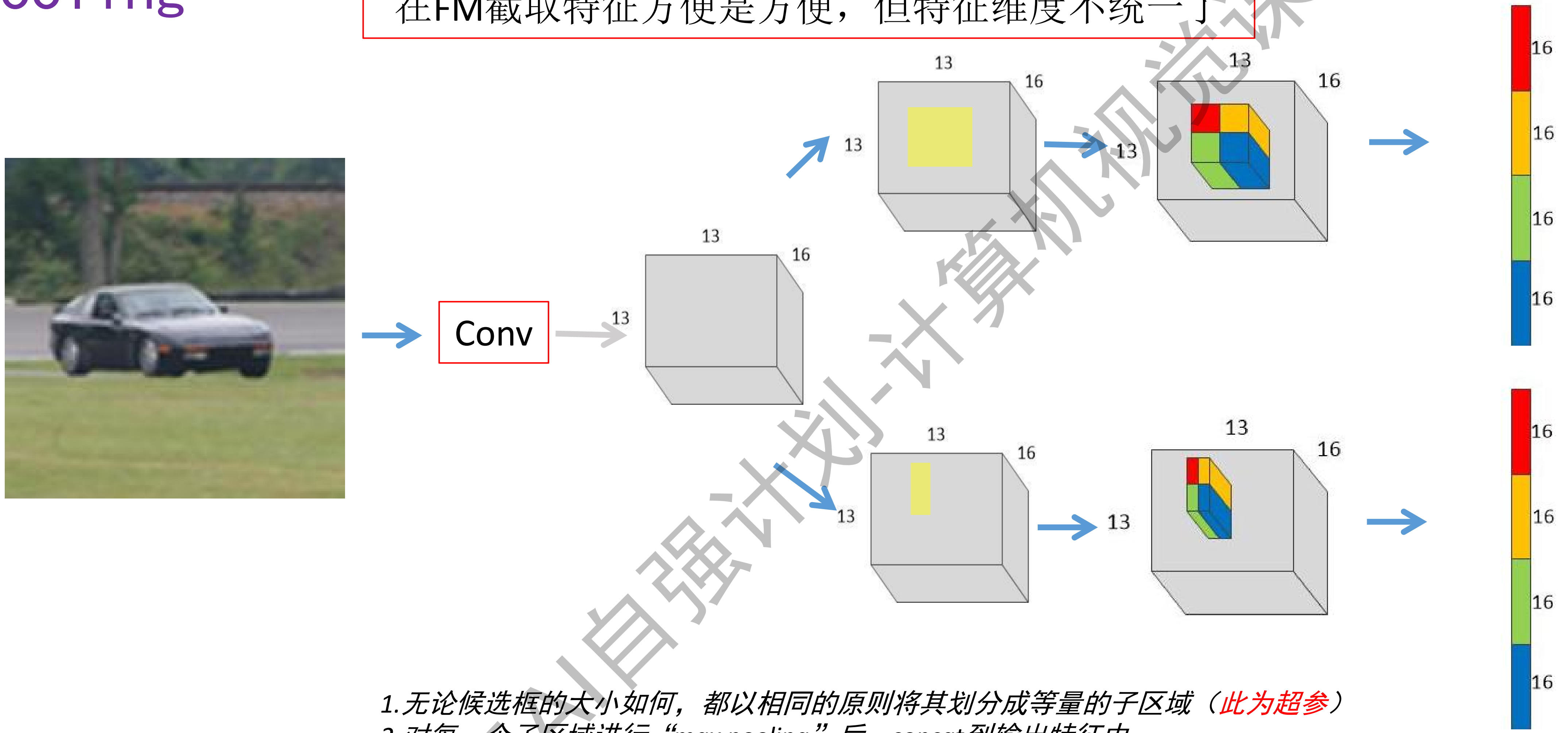
2.Fast R-CNN直接在分类网络输出的FM上以候选框截取目标特征

- *此方法节省了大量运算。*
- *无需resize,减少图像的信息损失。*

Feature map size	Center-x	Center-y	width	height
32*32	14/ 8	16/ 7	24/ 5	20/ 12
26*26	11/ 7	13/ 6	20/ 4	16/ 10
13*13	6/ 4	7/ 3	10/ 2	8/ 5

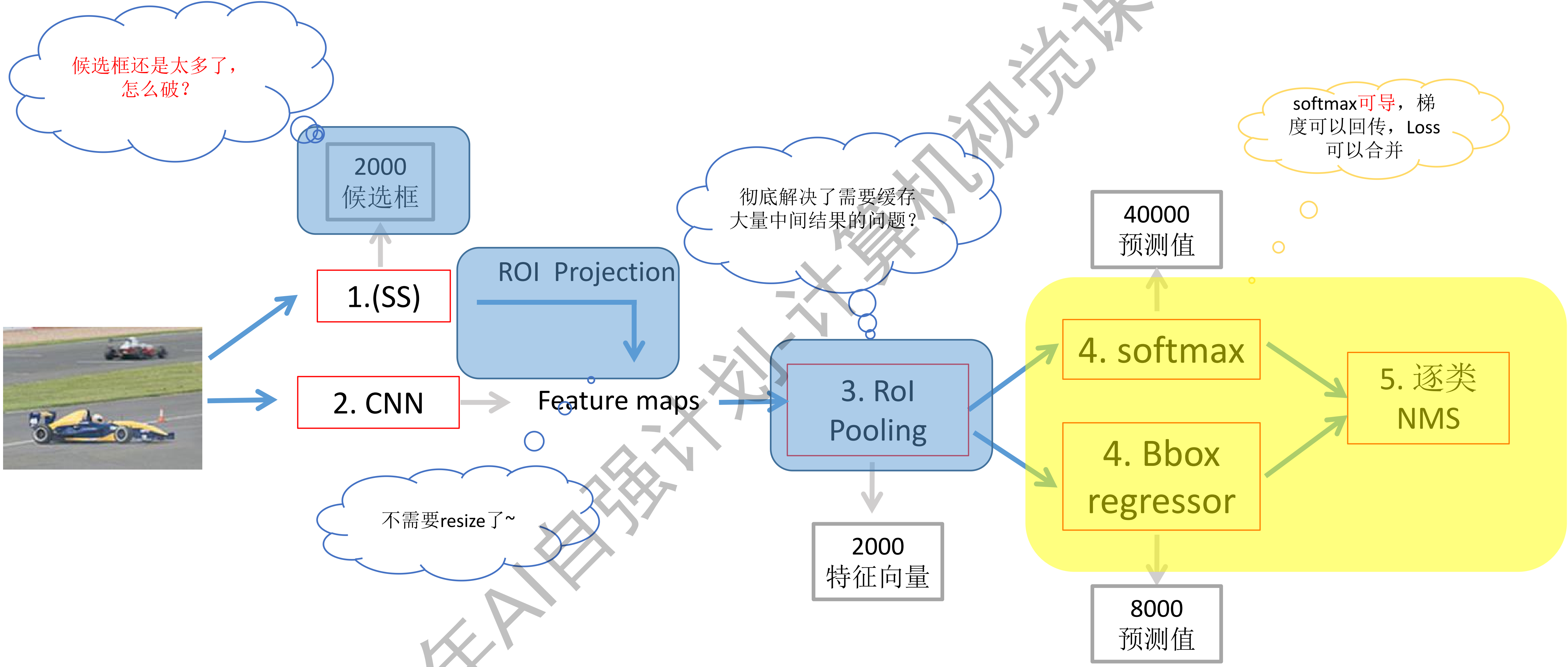
Fast R-CNN 统一输出特征维度RoI pooling

在FM截取特征方便是方便，但特征维度不统一了



1.无论候选框的大小如何，都以相同的原则将其划分成等量的子区域（此为超参）
2.对每一个子区域进行“max pooling”后，concat到输出特征中
3.此方法不需要resize即可输出统一维度的特征

Fast R-CNN 升级总结



1. *最后一层卷积层后加了一个ROI pooling layer*
2. *损失函数使用了多任务损失函数(multi-task loss)，将边框回归直接加入到CNN网 络中训练*

Faster R-CNN 创新点

不用ss，而用一个网络学习如何提取候选框

1. CNN

2.(RPN)

300 候选框

ROI Projection

Feature maps

3. RoI Pooling

300 特征向量

4. softmax

6000 预测值

4. Bbox regressor

1200 预测值

5. 逐类 NMS

RPN 略解

把RPN当做黑盒来理解

1. CNN → Feature maps → 2.(RPN) → 指定数量的候选框

训练集

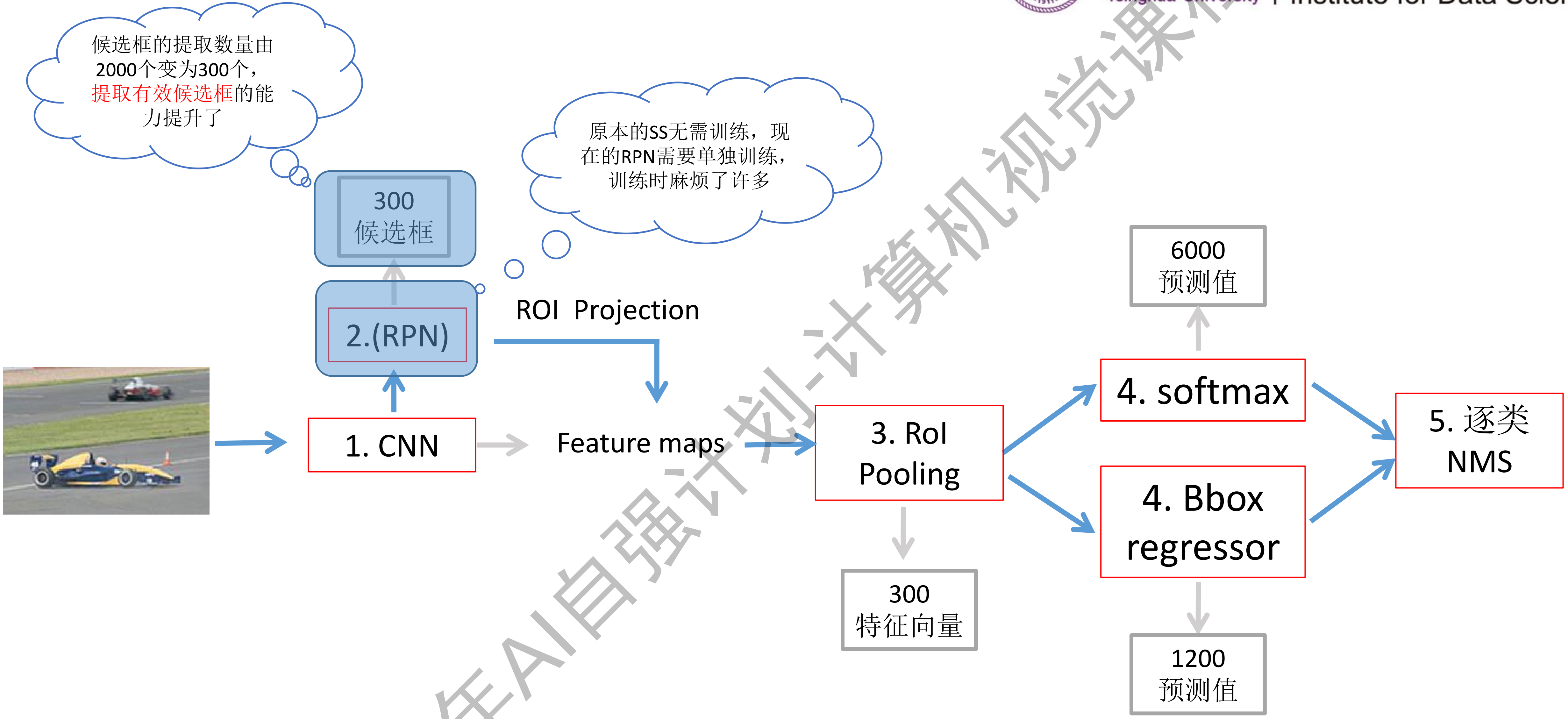
- 利用*anchor box*的思想（稍后介绍），在输入的*FM*上滑窗生成好多好多*bounding box*
- 选取与*ground truth*的IoU值最高的*bounding box*当作正样本。此外，如果一个*bounding box*和*ground truth*的IoU超过*0.7*,则也当成正样本
- 选取所有与*ground truth*的IoU低于*0.3*的*bounding*,作为负样本。
- 对于既不是正样本，也不是负样本的*bounding* box则直接丢弃。超出边界的*bounding* box也丢弃

loss

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

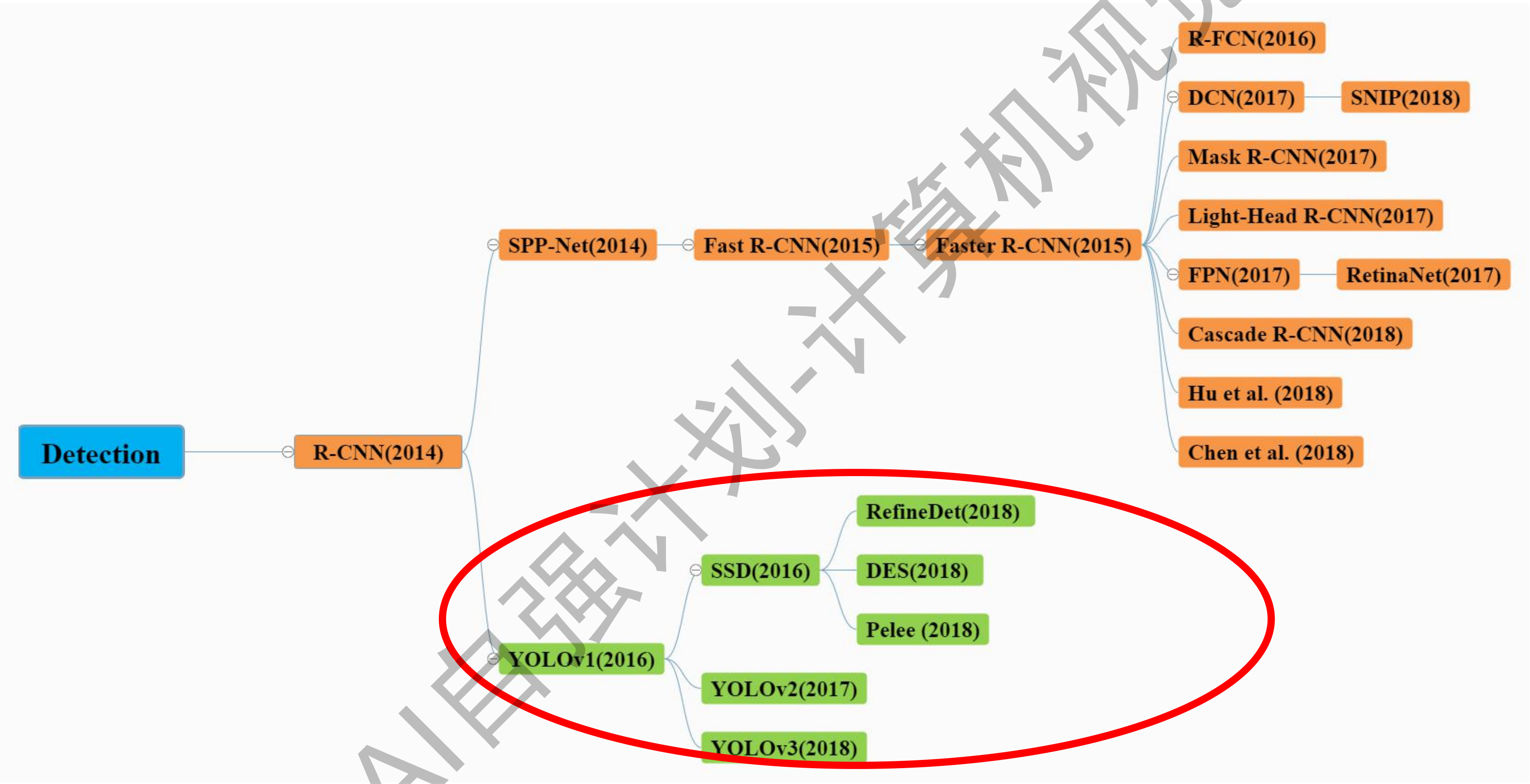
第一项表示分类损失
第二项表示回归损失

Faster R-CNN 小结

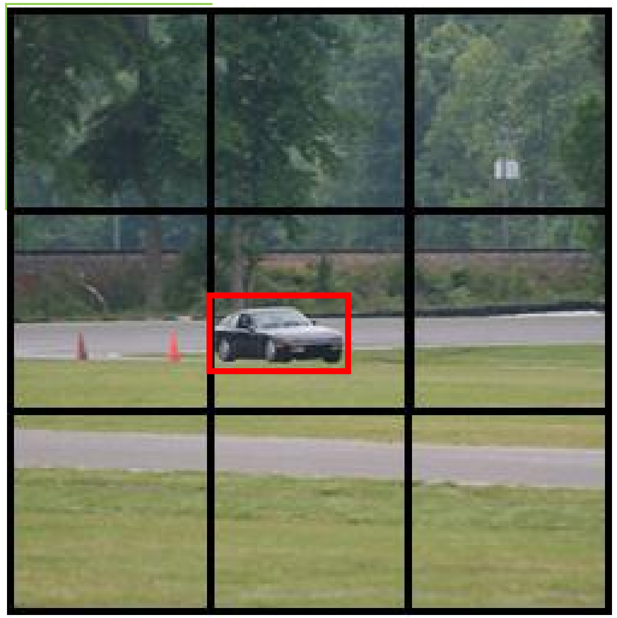


回顾

Two-stage 思想的罪魁祸首是过多的候选框，能否有更高效的思路？



YOLO 分而治之



将图像划分为若干(超参)个girds(网格)
并针对每一个网格单独做出标注
此处我们将图像划分为3*3共9个网格

假设我们的数据集共有2 个类别
Category-1：汽车；Category-2：人

$$Y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \end{bmatrix}$$

有目标的概率

bbox 是多少

目标属于类别1的概率

目标属于类别2的概率

每一个网格都有一个 Y 值
一幅图像有多少网格，就有多少个 Y 值

$$Y_{label} = \begin{bmatrix} P_c = 0 \\ b_x = ? \\ b_y = ? \\ b_w = ? \\ b_h = ? \\ C_1 = ? \\ C_2 = ? \end{bmatrix}$$

当 P_c 的取值为 0 时
Y 中其他所有取值
我们并不关心

此处的取值是相对于所在网格的

$$Y_{label} = \begin{bmatrix} P_c = 1 \\ b_x = 0.35 \\ b_y = 0.6 \\ b_w = 0.68 \\ b_h = 0.36 \\ C_1 = 1 \\ C_2 = 0 \end{bmatrix}$$

当 P_c 的取值为1时
Y 给出了正确的bbox
以及所属的类别

YOLO = You Only Look Once

清華大學 Tsinghua University | 数据科学研究院 Institute for Data Science

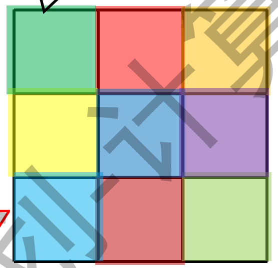
每个网格内，
图像的尺寸为
75 ×75 ×3

$225 \times 225 \times 3$

CNN

通过选取合适的CNN
将输出FM的尺寸调整为：3 ×3 ×7

FM中每个元素都是
原图中相应**图像块**
的**映射**



$\boxed{3 \times 3} \times \boxed{7}$

与grid数
量对应

与Y的位
数一致

这个就是对应grid的
输出$Y_{predict}$值啦

取出
一条元素

$\boxed{1 \times 1 \times 7}$

与“正确答案”Y_{lable}
对应做差得到Loss

$(\widetilde{P_c} - P_c)^2 + (\widetilde{b_x} - b_x)^2 + \ldots + (\widetilde{C_1} - C_2)^2$

$\|$

Loss of central grid

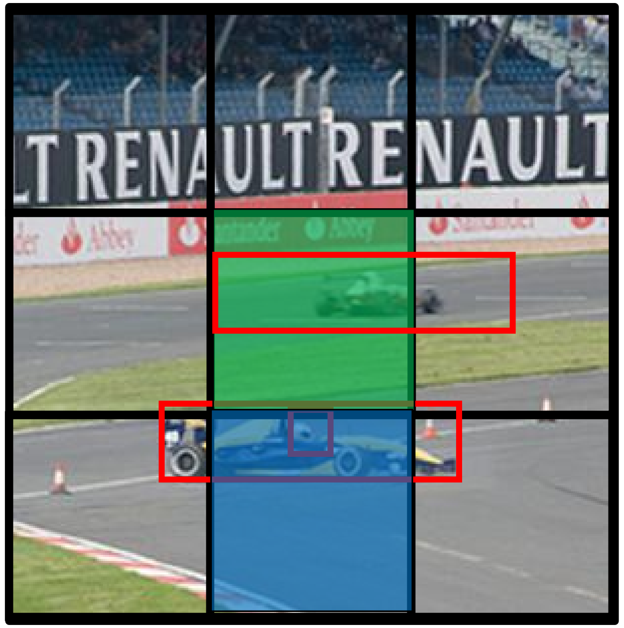
我们把原始图片分为 3 ×3 个网格
每个网格对应的标签Y 为 7 位

根据区域映射关系，FM中左上角的元素，
是左上角grid的映射，其他同理。

$$Y_{predict} = \begin{bmatrix} \widetilde{P_c} = 1 \\ \widetilde{b_x} = 0.3 \\ \widetilde{b_y} = 0.5 \\ \widetilde{b_w} = 0.68 \\ \widetilde{b_h} = 0.36 \\ \widetilde{C_1} = 1 \\ \widetilde{C_2} = 0 \end{bmatrix} \quad Y_{label} = \begin{bmatrix} P_c = 1 \\ b_x = 0.35 \\ b_y = 0.6 \\ b_w = 0.68 \\ b_h = 0.36 \\ C_1 = 1 \\ C_2 = 0 \end{bmatrix}$$

YOLO + anchor box

如果遇到一个网格中有多个目标物体怎么办?



按照同样的网格划分方法划分
中下方的grid 有2个分属不同类别的目标物体
这可怎么办?

显然这样的label设置
已经不能满足需要

$$Y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \end{bmatrix}$$

引入 anchor box →

anchor box的数量以及宽、高值都是预设好的

Anchor box-1=(b_x, b_y, 0.2, 0.2)

Anchor box-2=(b_x, b_y, 1.5, 0.4)

这里我们预设了两个anchor

升级 label →

Abox-1

Abox-2

$$Y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \end{bmatrix} \quad Y = \begin{bmatrix} P_c = 0 \\ b_x = ? \\ b_y = ? \\ b_w = ? \\ b_h = ? \\ C_1 = ? \\ C_2 = ? \\ P_c = 1 \\ b_x = 0.8 \\ b_y = 0.4 \\ b_w = 1.5 \\ b_h = 0.4 \\ C_1 = 1 \\ C_2 = 0 \end{bmatrix} \quad Y = \begin{bmatrix} P_c = 1 \\ b_x = 0.5 \\ b_y = 0.1 \\ b_w = 0.2 \\ b_h = 0.2 \\ C_1 = 0 \\ C_2 = 1 \\ P_c = 1 \\ b_x = 0.5 \\ b_y = 0.1 \\ b_w = 1.5 \\ b_h = 0.4 \\ C_1 = 1 \\ C_2 = 0 \end{bmatrix}$$

横向对比arXiv:1809.02165v1 [cs.CV] 6 Sep 2018

清華大学 Tsinghua University | 数据科学研究院 Institute for Data Science

	Detector Name	RP	Backbone DCNN	Input ImgSize	VOC07 Results	VOC12 Results	Speed (FPS)	Published In	Source Code	Highlights and Disadvantages
Region based (Section 3.1)	RCNN [65]	SS	AlexNet	Fixed	58.5 (07)	53.3 (12)	< 0.1	CVPR14	Caffe Matlab	**Highlights:** First to integrate CNN with RP methods; Dramatic performance improvement over previous state of the art; ILSVRC2013 detection result 31.4% mAP. **Disadvantages:** Multistage pipeline of sequentially-trained (External RP computation, CNN finetuning, Each warped RP passing through CNN, SVM and BBR training); Training is expensive in space and time; Testing is slow.
	SPPNet [77]	SS	ZFNet	Arbitrary	60.9 (07)	–	< 1	ECCV14	Caffe Matlab	**Highlights:** First to introduce SPP into CNN architecture; Enable convolutional feature sharing; Accelerate RCNN evaluation by orders of magnitude without sacrificing performance; Faster than OverFeat; ILSVRC2013 detection result 35.1% mAP. **Disadvantages:** Inherit disadvantages of RCNN except the speedup; Does not result in much speedup of training; Finetuning not able to update the CONV layers before SPP layer.
	Fast RCNN [64]	SS	AlexNet VGGM VGG16	Arbitrary	70.0 (VGG) (07+12)	68.4 (VGG) (07++12)	< 1	ICCV15	Caffe Python	**Highlights:** First to enable end to end detector training (when ignoring the process of RP generation); Design a RoI pooling layer (a special case of SPP layer); Much faster and more accurate than SPPNet; No disk storage required for feature caching; **Disadvantages:** External RP computation is exposed as the new bottleneck; Still too slow for real time applications.
	Faster RCNN [175]	RPN	ZFnet VGG	Arbitrary	73.2 (VGG) (07+12)	70.4 (VGG) (07++12)	< 5	NIPS15	Caffe Matlab Python	**Highlights:** Propose RPN for generating nearly cost free and high quality RPs instead of selective search; Introduce translation invariant and multiscale anchor boxes as references in RPN; Unify RPN and Fast RCNN into a single network by sharing CONV layers; An order of magnitude faster than Fast RCNN without performance loss; Can run testing at 5 FPS with VGG16. **Disadvantages:** Training is complex, not a streamlined process; Still fall short of real time.
	RCNN$\ominus$R [117]	New	ZFNet +SPP	Arbitrary	59.7 (07)	–	< 5	BMVC15	–	**Highlights:** Replace selective search with static RPs; Prove the possibility of building integrated, simpler and faster detectors that rely exclusively on CNN. **Disadvantages:** Fall short of real time; Decreased accuracy from not having good RPs.
	RFCN [40]	RPN	ResNet101	Arbitrary	80.5 (07+12) 83.6 (07+12+CO)	77.6 (07++12) 82.0 (07++12+CO)	< 10	NIPS16	Caffe Matlab	**Highlights:** Fully convolutional detection network; Minimize the amount of regionwise computation; Design a set of position sensitive score maps using a bank of specialized CONV layers; Faster than Faster RCNN without sacrificing much accuracy. **Disadvantages:** Training is not a streamlined process; Still fall short of real time.
Unified (Section 3.2)	YOLO [174]	–	GoogLeNet like	Fixed	66.4 (07+12)	57.9 (07++12)	< 25 (VGG)	CVPR16	DarkNet	**Highlights:** First efficient unified detector; Drop RP process completely; Elegant and efficient detection framework; Significantly faster than previous detectors; YOLO runs at 45 FPS and Fast YOLO at 155 FPS; **Disadvantages:** Accuracy falls far behind state of the art detectors; Struggle to localize small objects.
	YOLOv2[173]	–	DarkNet	Fixed	78.6 (07+12)	73.5 (07++12)	< 50	CVPR17	DarkNet	**Highlights:** Propose a faster DarkNet19; Use a number of existing strategies to improve both speed and accuracy; Achieve high accuracy and high speed; YOLO9000 can detect over 9000 object categories in real time. **Disadvantages:** Not good at detecting small objects.
	SSD [136]	–	VGG16	Fixed	76.8 (07+12) 81.5 (07+12+CO)	74.9 (07++12) 80.0 (07++12+CO)	< 60	ECCV16	Caffe Python	**Highlights:** First accurate and efficient unified detector; Effectively combine ideas from RPN and YOLO to perform detection at multiscale CONV layers; Faster and significantly more accurate than YOLO; Can run at 59 FPS; **Disadvantages:** Not good at detecting small objects.

从论文数量上看，R-CNN是主流，并且其精度很高，但训练难，速度慢

YOLO作为后起之秀主要特点是好用，易于训练，速度快但缺点是检测小目标效果不佳

尾声

下堂课将

1、明确获取证书的具体办法；

2、明确所有的作业、课件、视频、比赛的节奏

3、给大家现场组队参加“转化任务”的机会

4、由算法资源赞助商提供精美的小礼品