多传感器融合感知技术（选修）

岑汝平（数据谷C609）

cenruping@cqupt.edu.cn

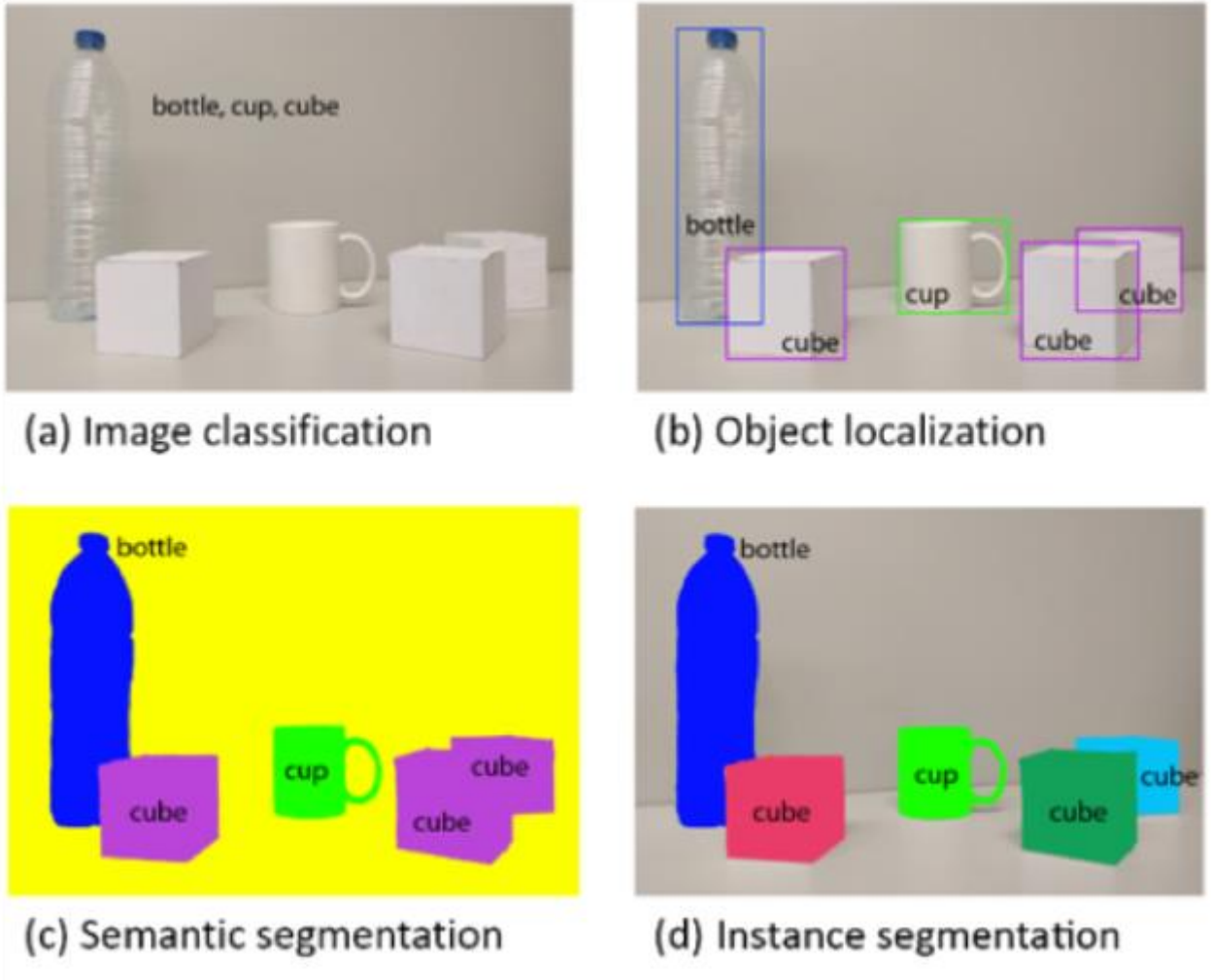
第4章 视觉目标检测

- **4.1 图像目标检测的基本原理**
- 4.2 图像目标检测
- 4.3 图像识别的原理

四、视觉目标检测 -- 4.1 图像目标检测的基本原理

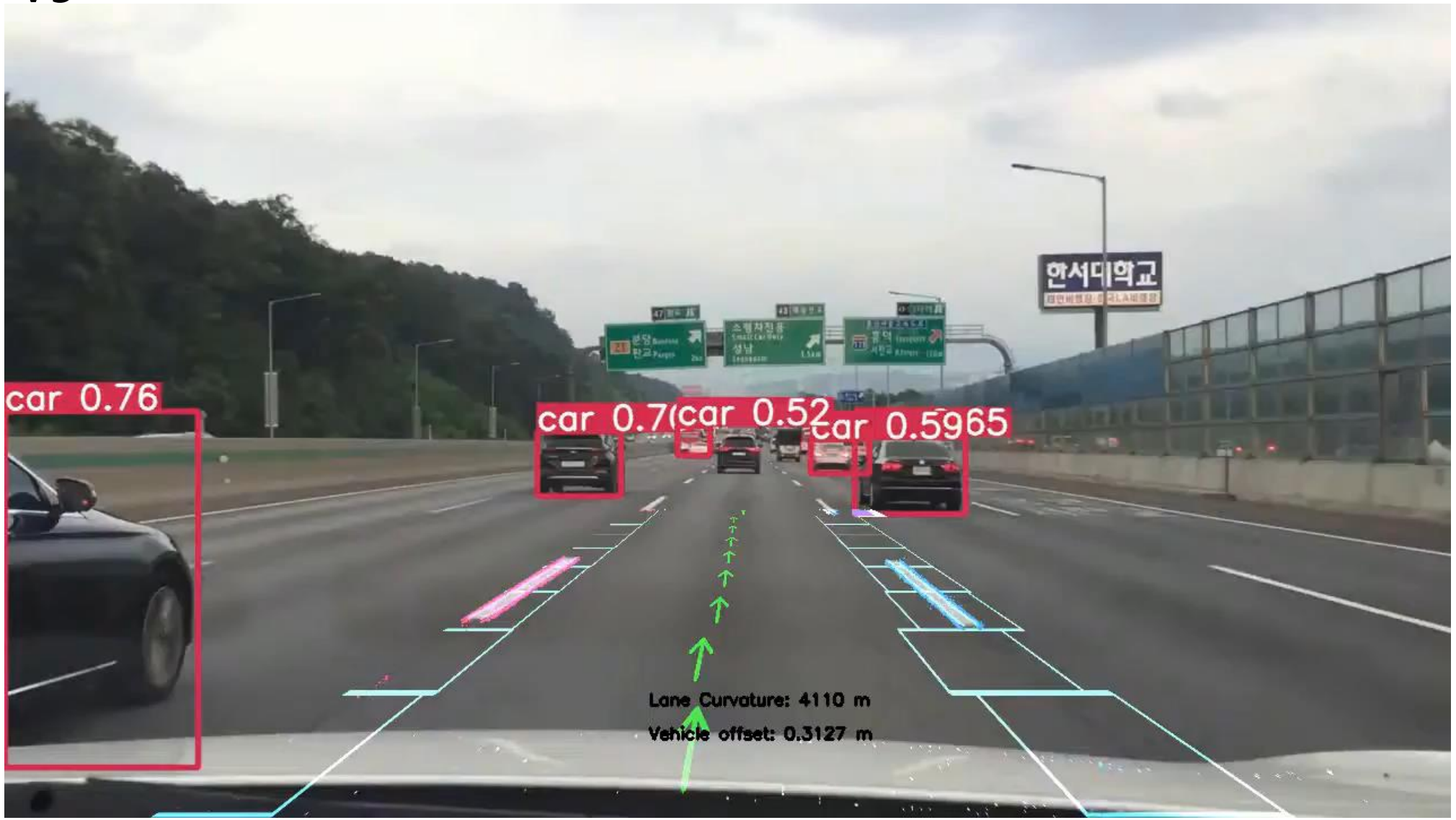
计算机视觉任务

- **(a) Image Classification**：图像分类，用于识别图像中物体的类别（如：bottle、cup、cube）。
- **(b) Object Localization**：目标检测，用于检测图像中每个物体的类别，并准确标出它们的位置。
- **© Semantic Segmentation**：图像语义分割，用于标出图像中每个像素点所属的类别，属于同一类别的像素点用一个颜色标识。
- **(d) Instance Segmentation**：实例分割，值得注意的是，（b）中的目标检测任务只需要标注出物体位置，而（d）中的实例分割任务不仅要标注出物体位置，还需要标注出物体的外形轮廓。



四、视觉目标检测 -- 4.1 图像目标检测的基本原理

- YOLO-V5



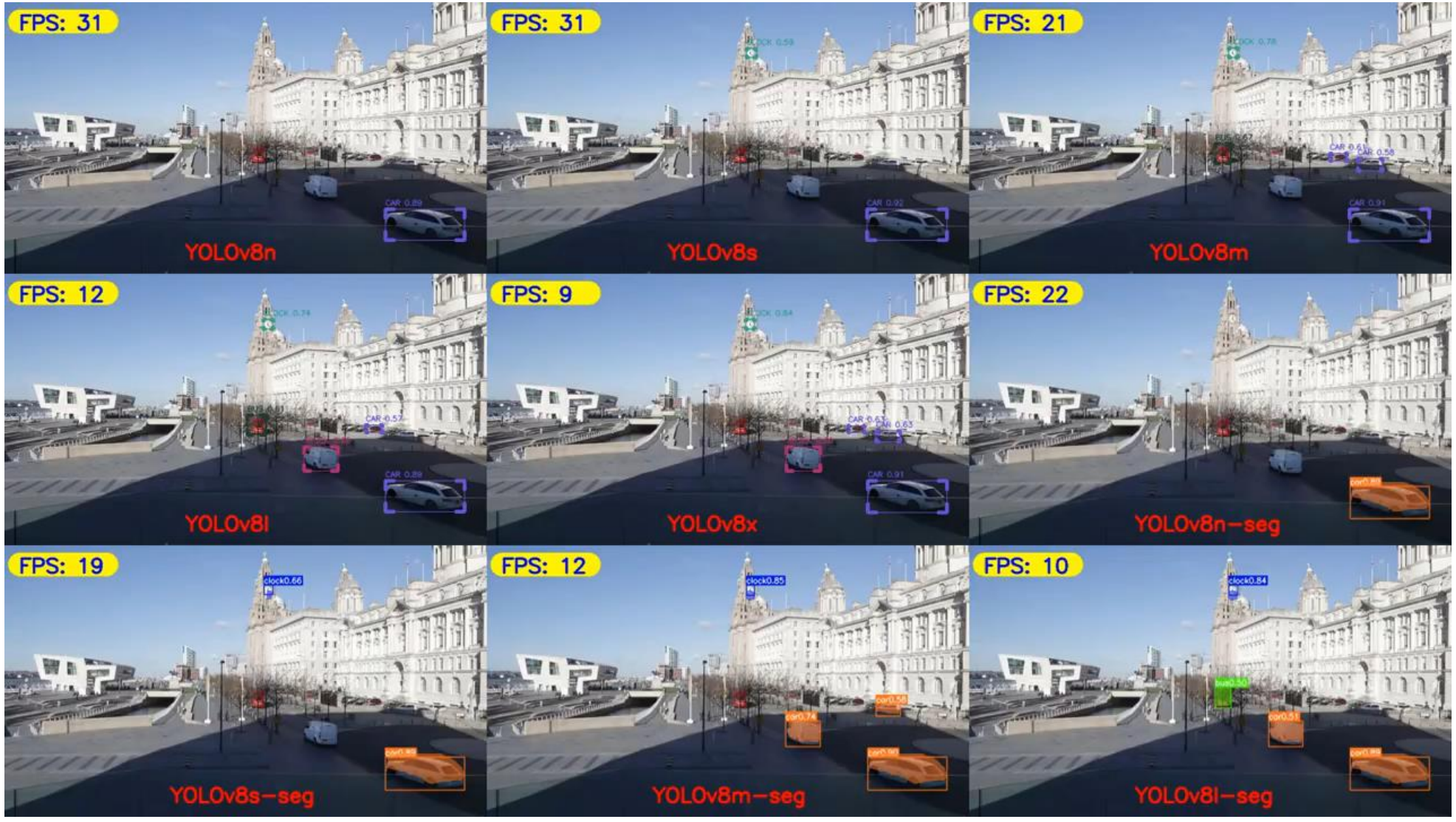
四、视觉目标检测 -- 4.1 图像目标检测的基本原理

YOLO-V5

https://github.com/nicedaddy/Yolov5_DeepSort_Pytorch_lane_detection

四、视觉目标检测 -- 4.1 图像目标检测的基本原理

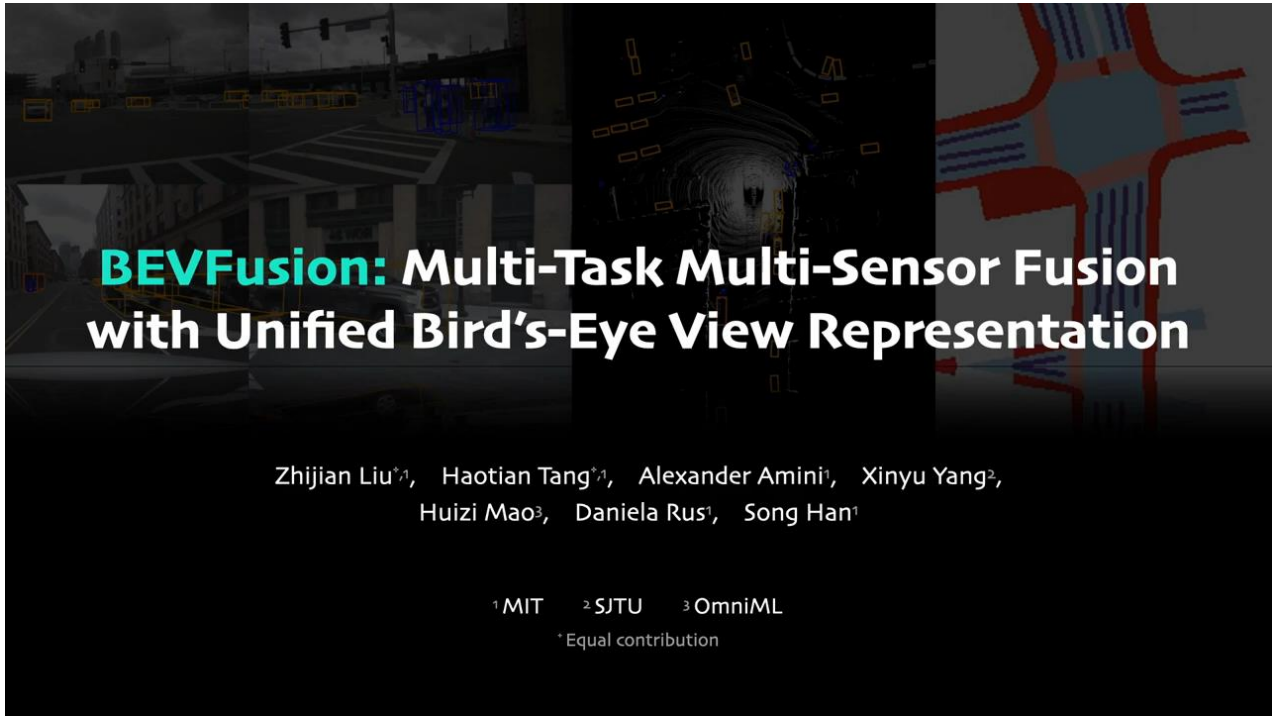
YOLO-V8



四、视觉目标检测 -- 4.1 图像目标检测的基本原理

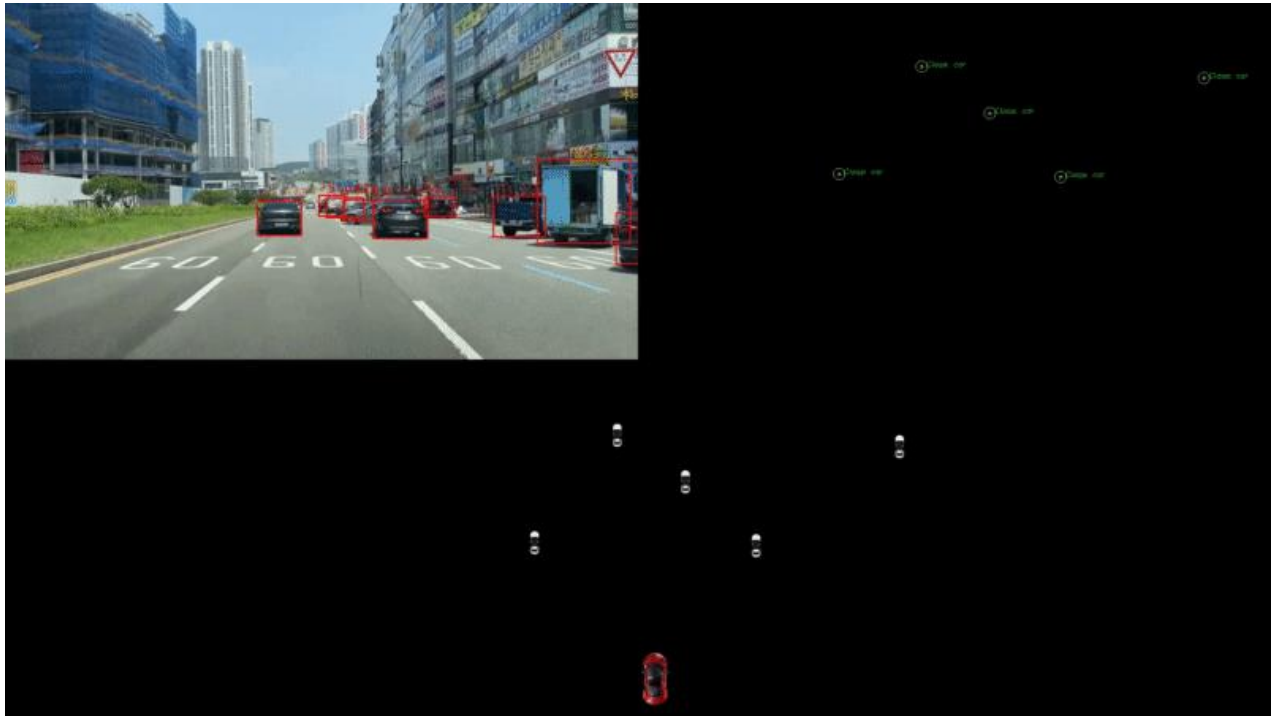
Bird ‘s Eye View (BEV)目标检测

BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation



https://bevfusion.mit.edu/

车辆、行人、骑自行车的人和交通标志。



https://github.com/bharath5673/yolov5_BEV

四、视觉目标检测 -- 4.1 图像目标检测的基本原理

边界框（bounding box）

检测任务需要同时预测物体的**类别**和**位置**，因此需要引入一些跟位置相关的概念。通常使用边界框（bounding box，bbox）来表示物体的位置，边界框是正好能包含物体的矩形框。

通常有两种格式来表示边界框的位置：

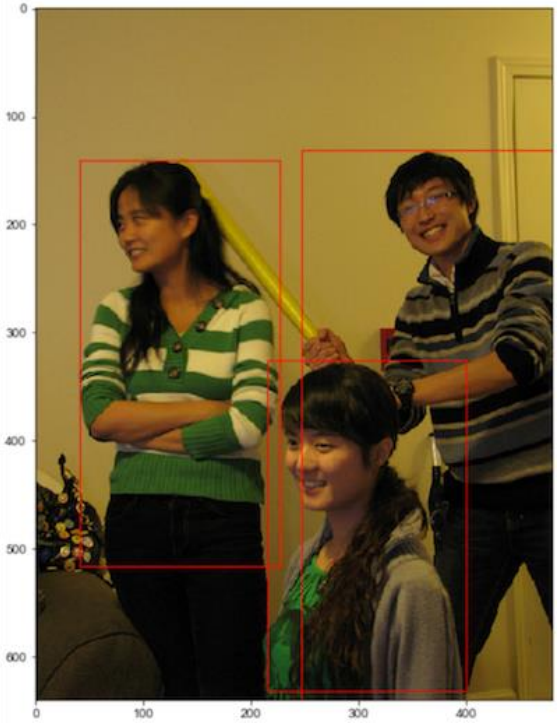
第一种方式$(x1,y1,x2,y2)$，其中$(x1,y1)$是矩形框左上角的坐标，$(x2,y2)$是矩形框右下角的坐标。

左：(40.93,141.1,226.99,515.73)(40.93,141.1,226.99,515.73)。

中：(214.29,325.03,399.82,631.37)(214.29,325.03,399.82,631.37)。

右：(247.2,131.62,480.0,639.32)(247.2,131.62,480.0,639.32)。

第二种方式(x,y,w,h)，其中(x,y)是矩形框中心点的坐标，w是矩形框的宽度，h是矩形框的高度。



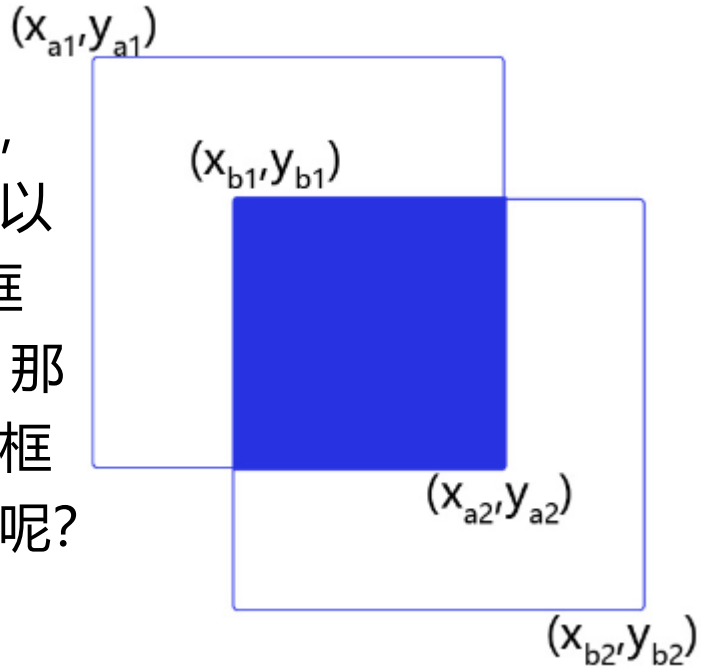
四、视觉目标检测 -- 4.1 图像目标检测的基本原理

交并比（Intersection of Union，IoU）

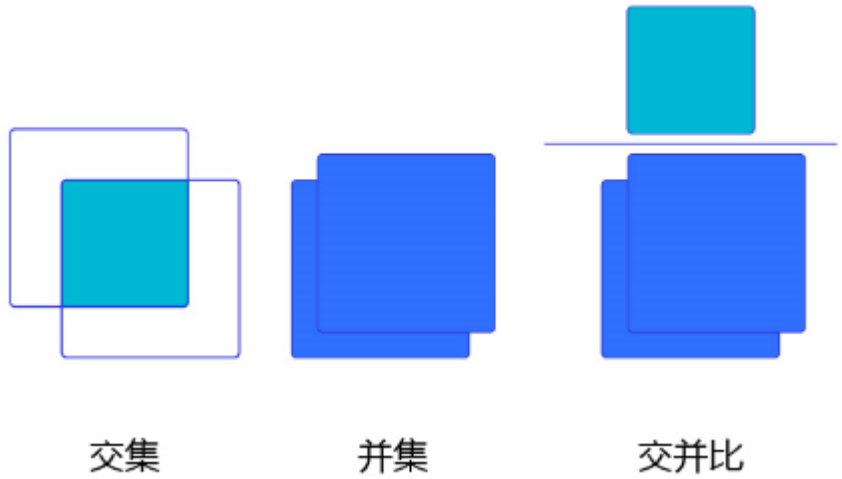
交并比描述两个框之间的重合度。两个框可以看成是两个像素的集合，它们的交并比等于两个框重合部分的面积除以它们**合并起来的面积**。

$$IoU = \frac{A \cap B}{A \cup B}$$

以点(300,500)为中心，生成的三个锚框，可以看到锚框A1 与真实框G1的重合度比较好。那么如何衡量这三个锚框跟真实框之间的关系呢?



下图“交集”中青色区域是两个框的重合面积，“并集”为蓝色区域是两个框的相并面积。用这两个面积相除即可得到它们之间的**交并比**。

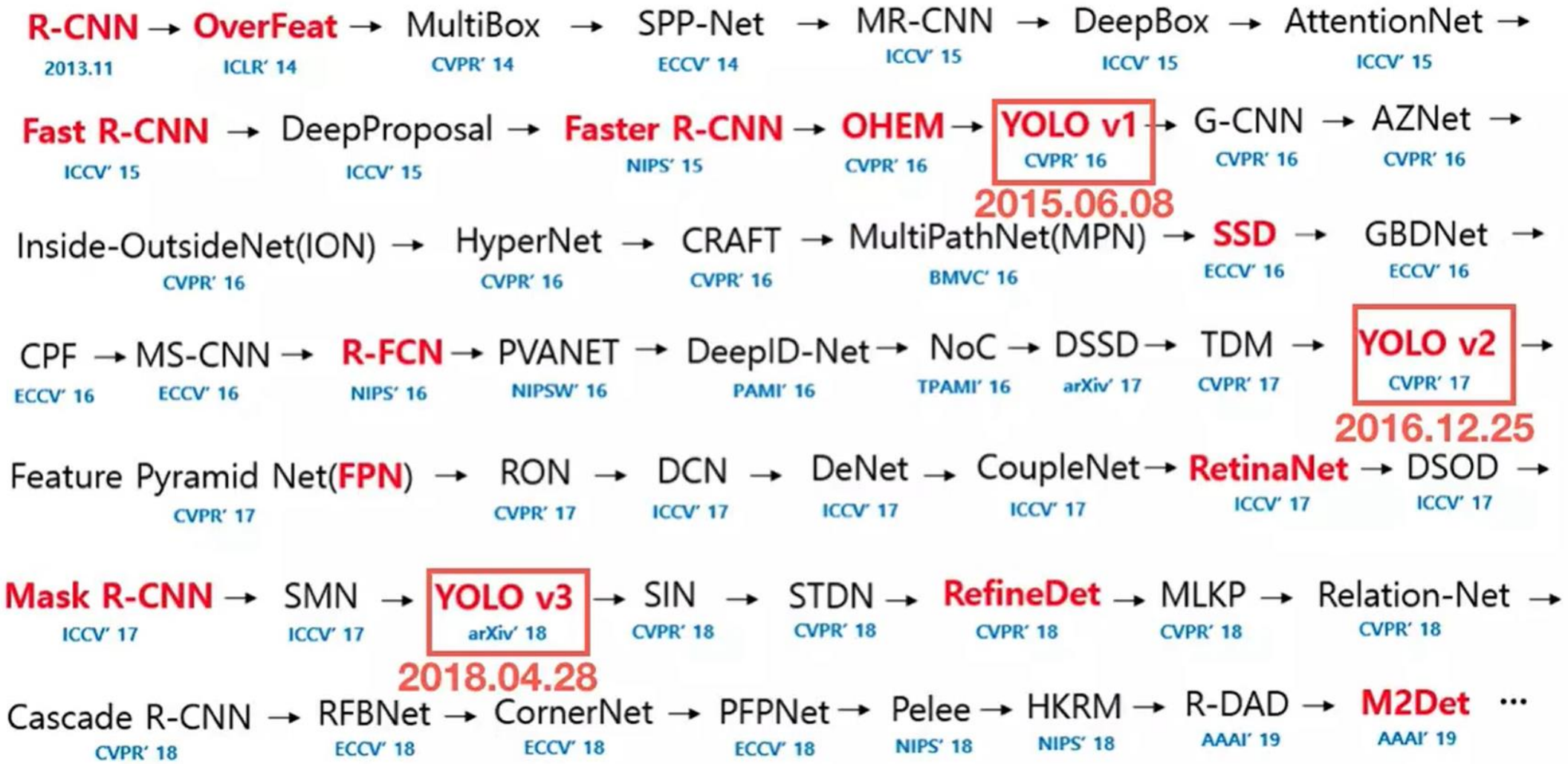


第4章 视觉目标检测

- 4.1 图像目标检测的基本原理
- **4.2 图像目标检测**
- 4.3 图像识别的原理

四、视觉目标检测 -- 4.2 图像目标检测

目标检测算法的历程



四、视觉目标检测 -- 4.2 图像目标检测

目标检测算法分类

基于深度学习的目标检测算法主要分为两类：

1.Two stage目标检测算法

先进行区域生成（region proposal，RP）（一个有可能包含待检物体的预选框），再通过卷积神经网络进行样本分类。

任务：特征提取—>生成RP—>分类/定位回归。

常见的two stage目标检测算法有：R-CNN、SPP-Net、Fast R-CNN和Faster R-CNN等。

2.One stage目标检测算法

不用RP，直接在网络中提取特征来预测物体分类和位置。

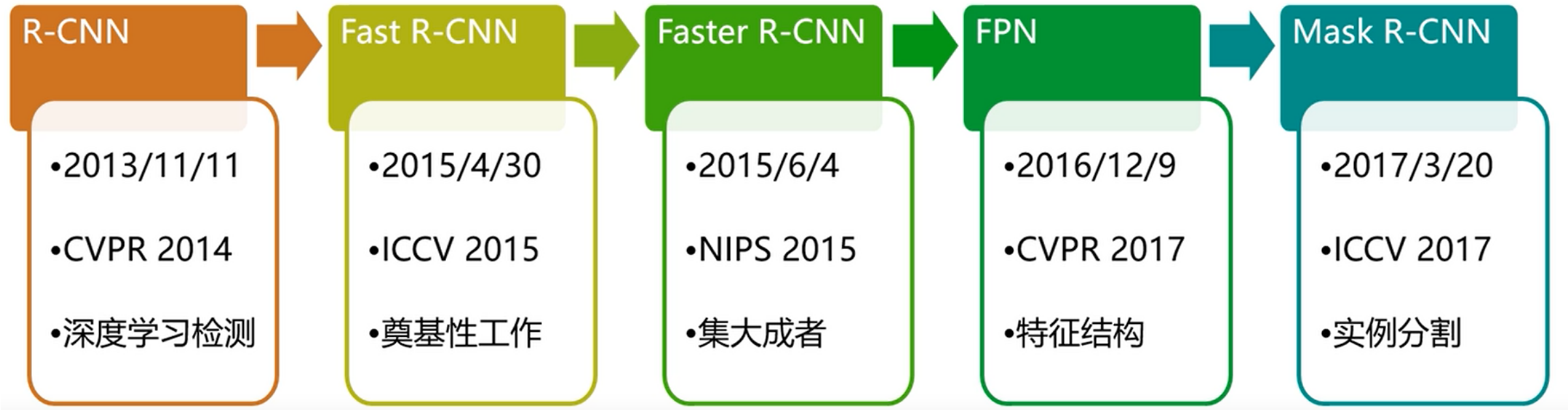
任务：特征提取—>分类/定位回归。

常见的one stage目标检测算法有：OverFeat、YOLO、SSD和RetinaNet等。

四、视觉目标检测

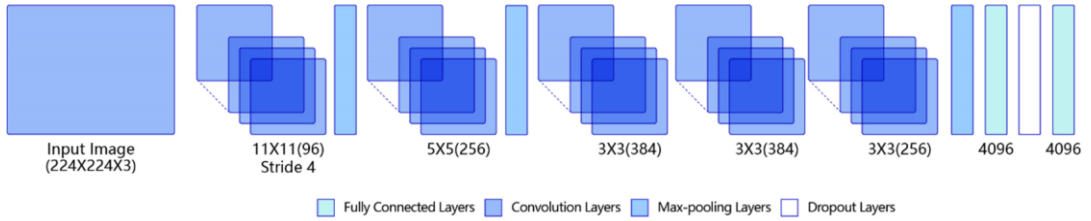
Two stage目标检测算法（基于锚框）

常见的two stage目标检测算法有：R-CNN、SPP-Net、Fast R-CNN和Faster R-CNN等。



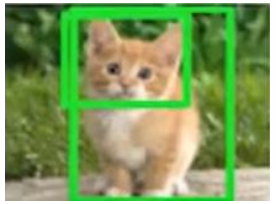
四、视觉目标检测 -- 4.2 图像目标检测

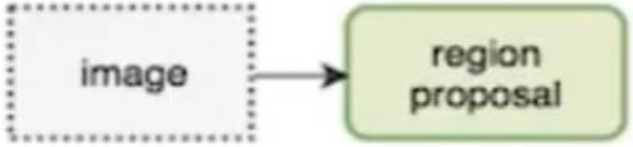
R-CNN算法流程



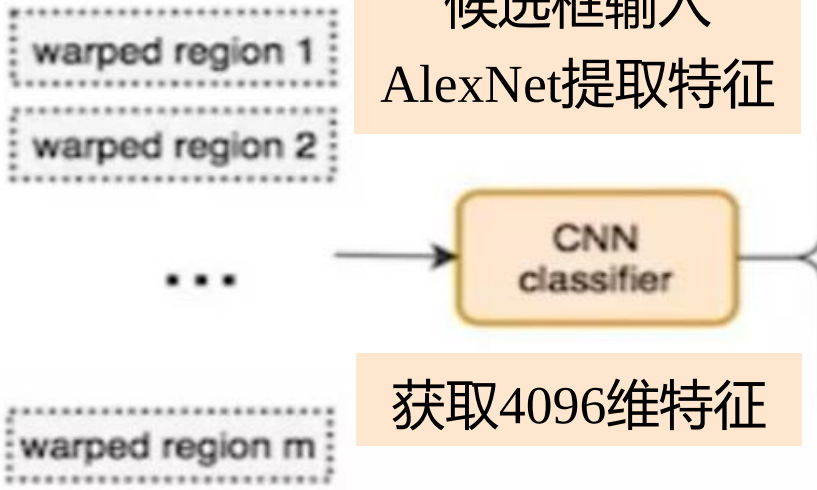
Step1: 输入图片

Step2: 生成1-2K个候选框





Step3: 将2000个候选框输入AlexNet提取特征



获取4096维特征

Step4: 假设输出类别是20个类，则创建20个SVM二分类器，分别对2000个候选框输出特征进行打分

Step5：修正bbox，对bbox做回归微调

Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. IEEE conference on computer vision and pattern recognition. 2014: 580-587.

R-CNN算法流程

1) 候选区域生成

SS即Selective Search（选择性搜索）方法

使用SS算法对一张图像生成约1000-2000个候选区域，基本思路如下：

（1）使用一种过分割手段，将图像分割成小区域

（2）查看现有小区域，合并可能性最高的两个区域，重复直到整张图像合并成一个区域位置。优先合并以下区域：

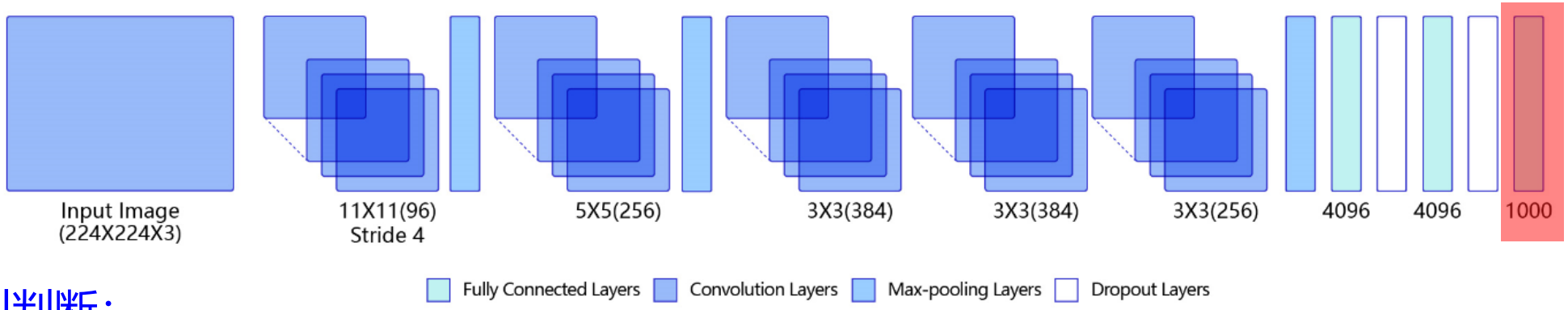
- 颜色（颜色直方图）相近的
- 纹理（梯度直方图）相近的
- 合并后总面积小的
- 合并后，总面积在其BBOX中所占比例大的在合并时须保证合并操作的尺度较为均匀，避免一个大区域陆续“吃掉”其它小区域，保证合并后形状规则。

（3）输出所有曾经存在过的区域，即所谓候选区域

R-CNN算法流程

2) 特征提取：

R-CNN选用Alexnet作为特征提取网络，获得候选区域的4096维特征。



3)类别判断：

对每一类目标，使用一个线性SVM二类分类器进行判别。输入为深度网络（如上图的AlexNet）输出的4096维特征，输出是否属于此类

将2000x4096维特征经过SVM分类器(20种分类，SVM是二分类器，则有20个SVM)，获得2000x20种类别得分矩阵（**每个类别的分类器都对2000个特征向量判断**）

四、视觉目标检测 -- 4.2 图像目标检测

R-CNN

4） 非极大值抑制

- 分别对2000x20维矩阵中进行**非极大值抑制**(NMS:non-maximum suppression)剔除重叠建议框，抑制冗余候选框（一个物体只保留一个最优框）

step1先利用得分剔除小于0.5的得分。

step2 假设：剩余的候选框假设图片真实物体个数为2(N)绿色表示，上一步筛选之后候选框为5(P)蓝色表示。

step3 计算N中每个物体位置与所有P的交并比(IoU)值，得到P中每个候选框对应IoU最高的N中一个，A、C候选框对应左边车辆，B、D、E对应右边车辆。（**为候选框归属类别**）

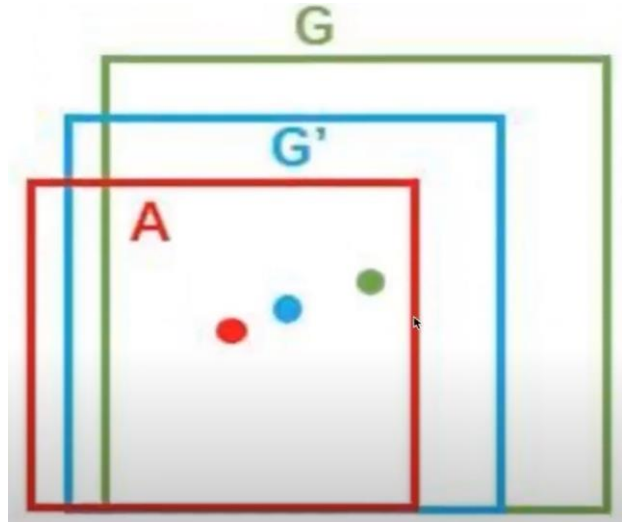
step4 循环迭代：对与每一个类别对应的候选框（以右边框为例），计算IOU得分最高的框B与其他几个候选框D、E的IOU，如果B与D/E的重叠很大(>0.5)，则删除DE，只保留B。（**为每个类别筛选出最优的候选框**）

四、视觉目标检测 -- 4.2 图像目标检测

R-CNN

5）修正bbox，对bbox做回归微调

假设：预测框与真实值之间存在线性关系



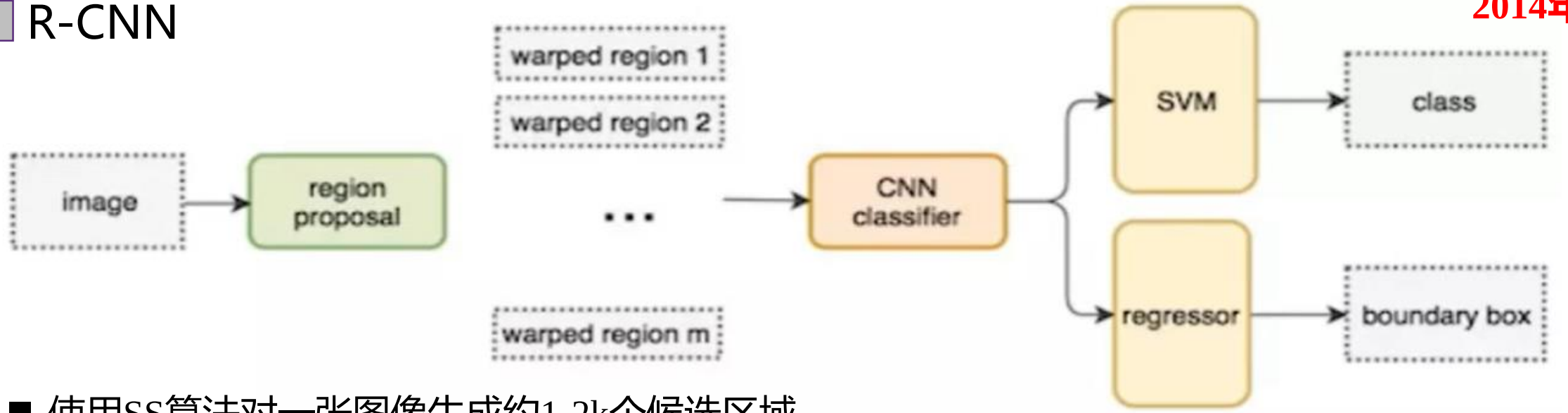
G为真实值

- 给定：anchor $A = (A_x, A_y, A_w, A_h)$ 和 $GT = [G_x, G_y, G_w, G_h]$
- 寻找一种变换F，使得：$F(A_x, A_y, A_w, A_h) = (G'_x, G'_y, G'_w, G'_h)$，$(G'_x, G'_y, G'_w, G'_h) \approx (G_x, G_y, G_w, G_h)$

四、视觉目标检测 -- 4.2 图像目标检测

R-CNN

2014年CVPR

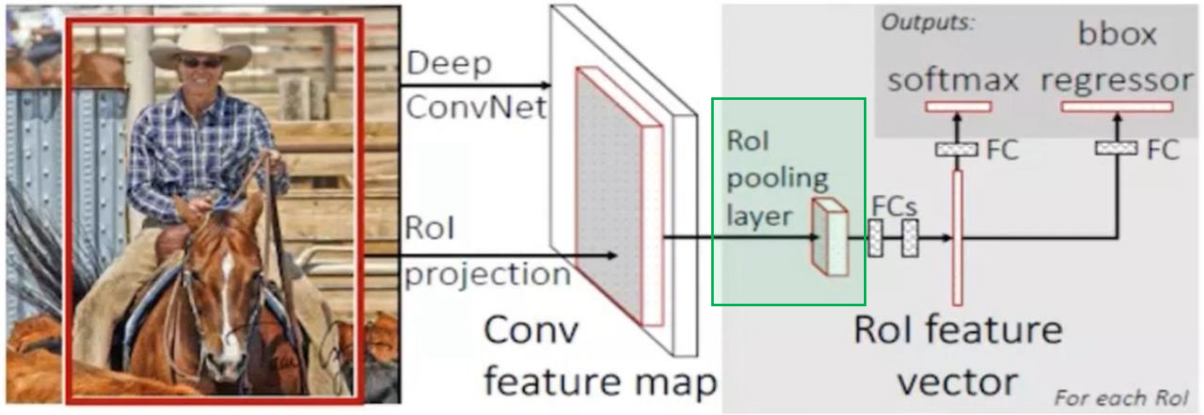


- 使用SS算法对一张图像生成约1-2k个候选区域
- 将候选区域图像尺寸调整为了适应AlexNet网络的输入图像的大小227x227，通过CNN对候选区域提取特征向量，2000个建议框的CNN特征组合成网络AlexNet最终输出: 2000x4096维矩阵
- 将2000x4096维特征经过SVM分类器(20种分类，SVM是二分类器，则有20个SVM)，获得2000x20种类别得分矩阵（**每个类别的分类器都对2000个特征向量判断**）
- 分别对2000x20维矩阵中进行非极大值抑制 (NMS:non-maximum suppression)剔除重叠建议框，抑制冗余候选框（**一个物体只保留一个最优框**）
- 修正bbox，对bbox做回归微调 （**与groundtruth框相比较，修正候选区域位置**）

四、视觉目标检测 -- 4.2 图像目标检测

Fast R-CNN

- 提出一个RoI Pooling层
- 整合整个模型，把CNN、Rol pooling、分类器、bbox回归几个模块整个一起训练



步骤

1.将整个图片输入到一个基础卷积网络，得到整张图的feature map

2.将选择性搜索算法的结果region proposal (Rol) 映射到feature map中

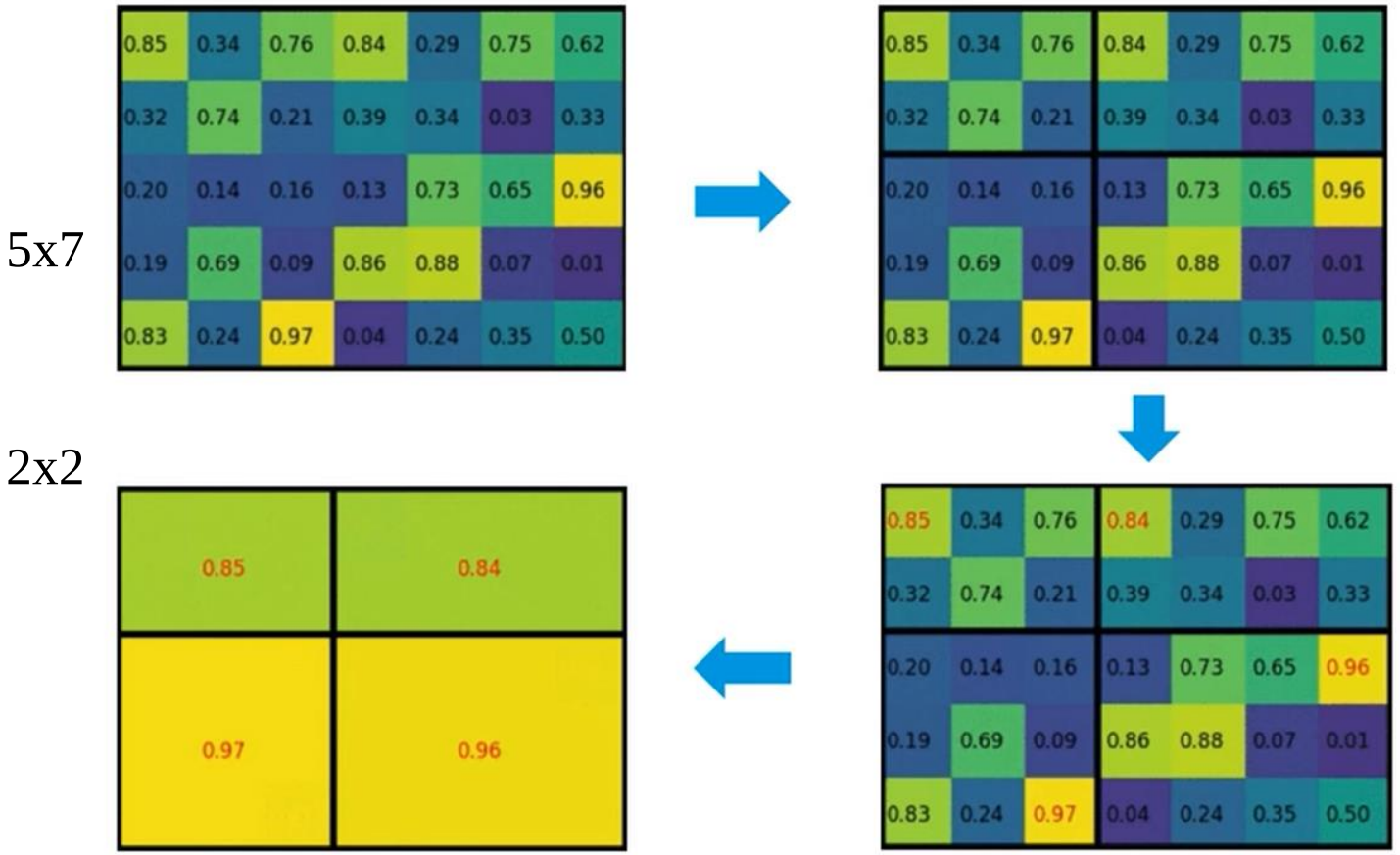
3.Rol pooling layer提取一个固定长度的特征向量，每个特征会输入到一系列全连接层，得到一个RoI特征向量(此步骤是对每一个候选区域都会进行同样的操作)

- softmax层进行分类，输出类别有K个类别加上”背景”类
- bounding box regressor

四、视觉目标检测

Fast R-CNN

ROI Pooling: 利用特征采样，把不同空间大小的特征，变为空间大小一致的特征。

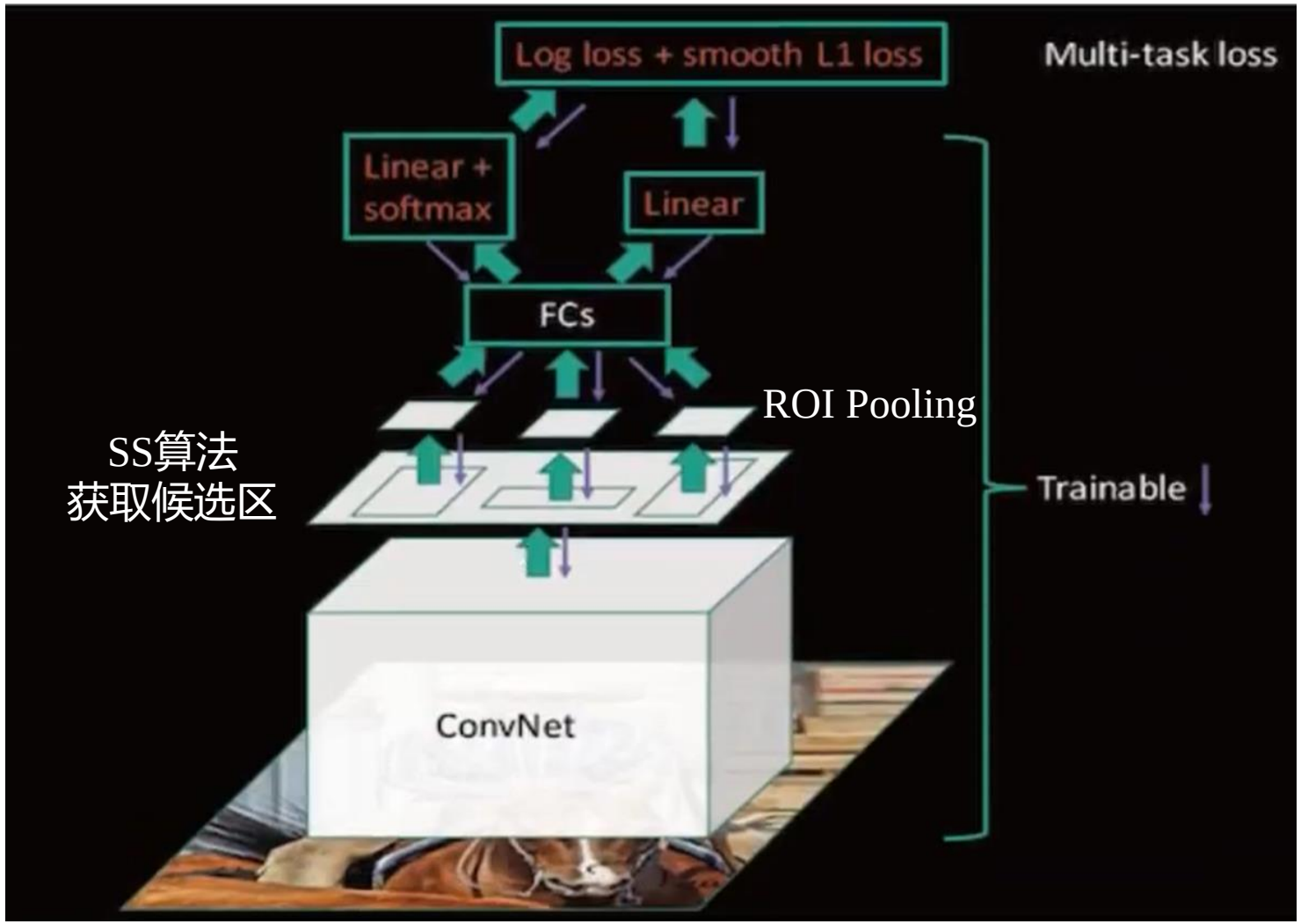


为什么要使用RolPooling把不同大小的特征变成固定大小?

网络后面是全连接层 (FC层)，要求输入有固定的维度各个候选区域的特征大小一致，可以组成batch进行处理

四、视觉目标检测 -- 4.2 图像目标检测

Fast R-CNN



优势

参数	R-CNN	Fast R-CNN
训练时间(h)	84	**9.5**
测试时间/图片	47.0s	**0.32s**
mAP	66.0	**66.9**

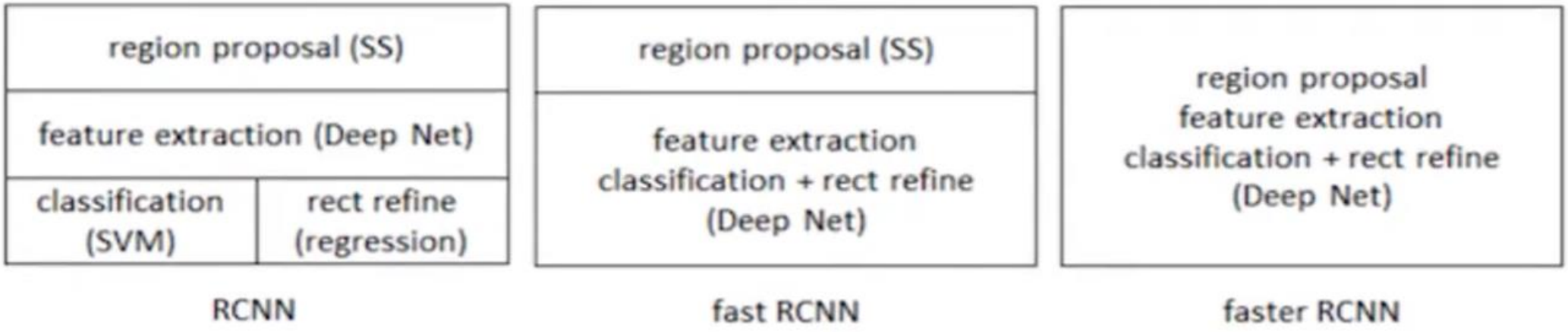
不足：使用Selective Search提取Region Proposals，没有实现真正意义上的**端对端**，操作也十分**耗时**

四、视觉目标检测 -- 4.2 图像目标检测

Faster R-CNN

- 候选区域筛选融合到网络当中

四个基本步骤(候选区域生成，特征提取，分类，位置精修) 被统一到一个深度网络框架



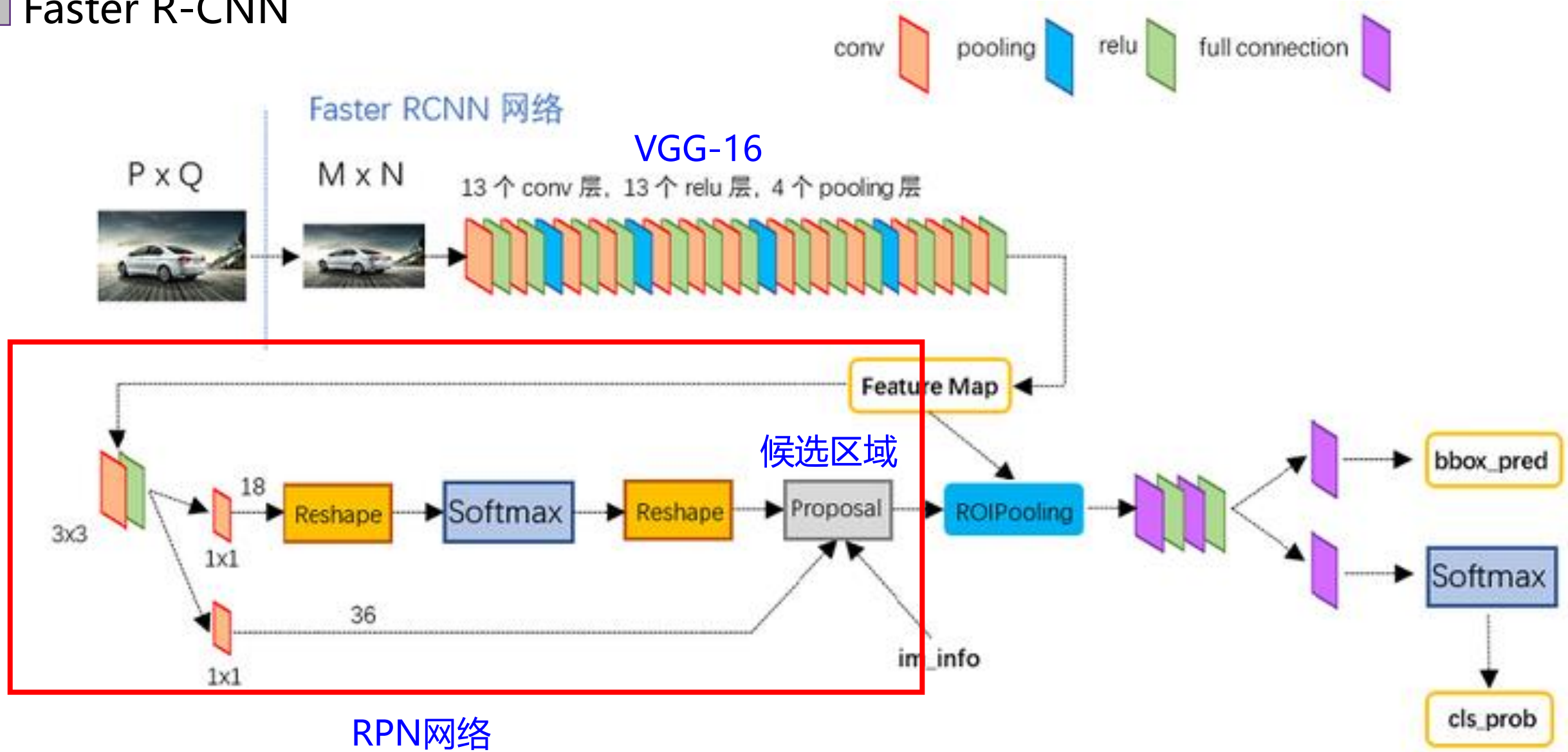
区域生成网络 (RPN) +Fast R-CNN　RPN网络替代了SS选择性搜索算法

RPN网络用于生成region proposals

- 通过softmax判断anchors属于foreground或者background
- Bounding box regression修正anchors获得精确的proposals
- 得到默认300个候选区域给ROI Pooling,后续步骤与Fast-RCNN相同

四、视觉目标检测 -- 4.2 图像目标检测

Faster R-CNN



四、视觉目标检测

物体检测数据集

通用物体检测数据集	人脸检测数据集	行人检测数据集
• PASCAL VOC	• AFW	• Caltech-USA
• MS COCO	• PASCAL FACE	• CityPersons
• OpenImages	• MALF	• CrowdHuman
• LVIS	• MAFA	• WiderPerson
	• FDDB	• EuroCityPersons
	• WIDER FACE	

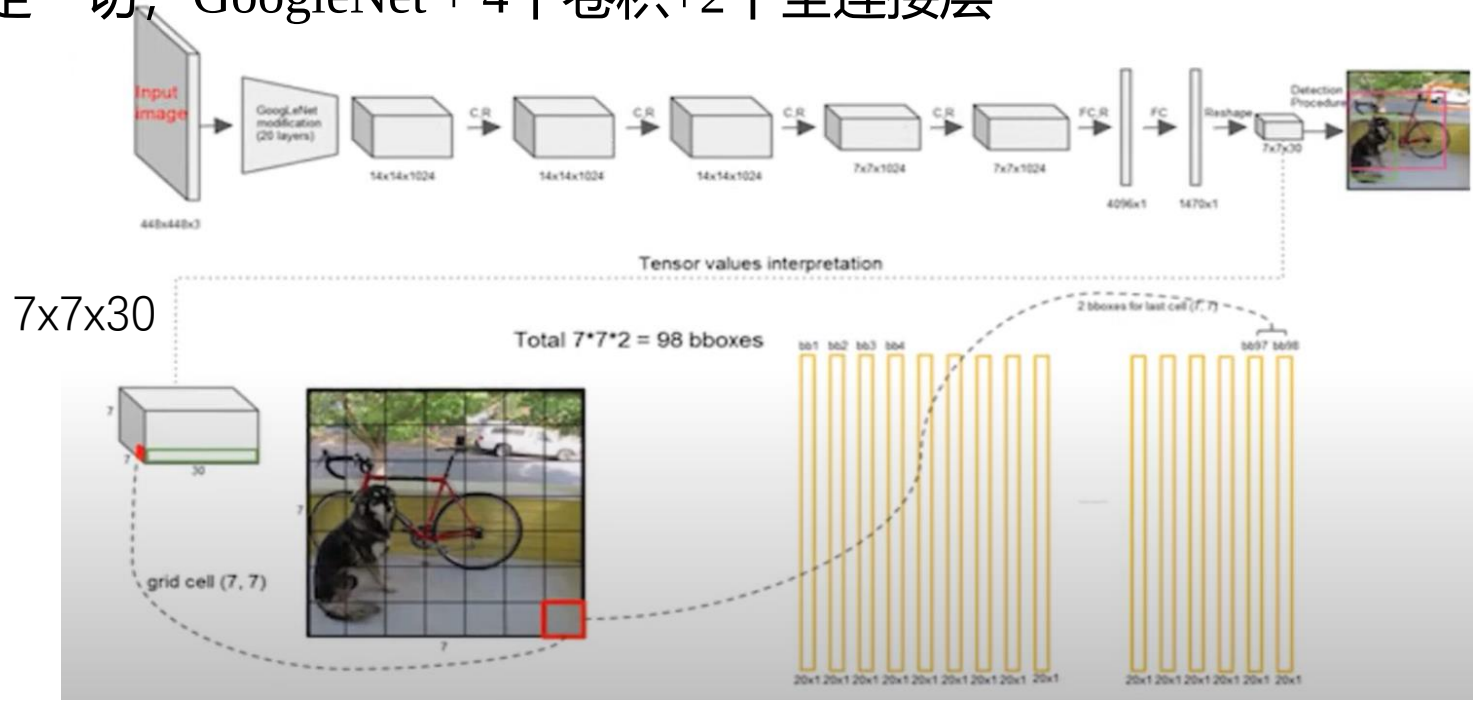
第5章 YOLO目标检测算法

- **5.1 yolo图像目标检测的基本原理**
- 5.2 yolo环境安装方法
- 5.3 行人目标检测数据集
- 5.4 yolo模型参数重载
- 5.5 基于摄像头数据实现行人检测

五、YOLO目标检测算法

YOLO网络结构

一个网络搞定一切，GoogleNet + 4个卷积+2个全连接层



Step1: 原始图片resize到448x448，经过前面卷积网络之后，将图片输出成了一个7x7x30的结构

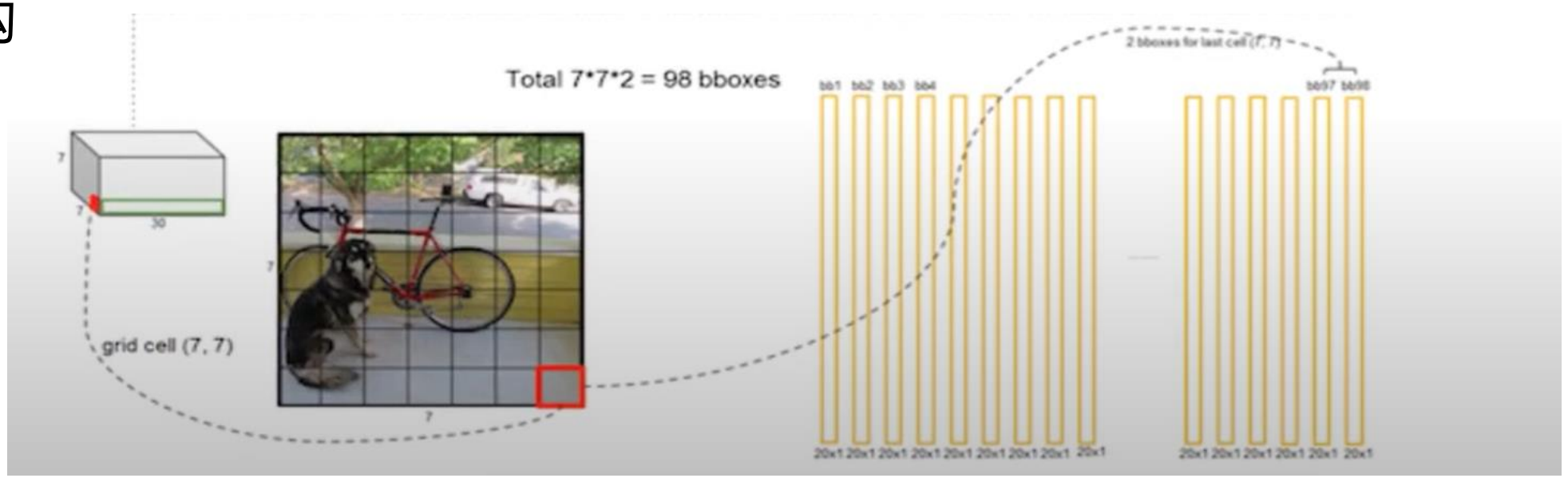
Step2 把图像分为7x7的**网格**，每个**网格**提取**2个候选框 (得到98个候选框)**

Step3 进行NMS筛选，筛选概率及IoU

五、YOLO目标检测算法

YOLO网络结构

单元格 grid cell



每一个7x7单元格的有30维的特征：

1.每个单元格负责预测一个物体类别，并且直接预测物体的概率值

2.每个单元格预测两个(默认bbox位置，两个bbox置信度(confidence)

30=((4+1)*2+20),4个坐标信息，1个置信度(confidence) +20类别的概率

方框1的（x,y,w,h,confidence）+方框2的（x,y,w,h,confidence）+20个类别

五、YOLO目标检测算法

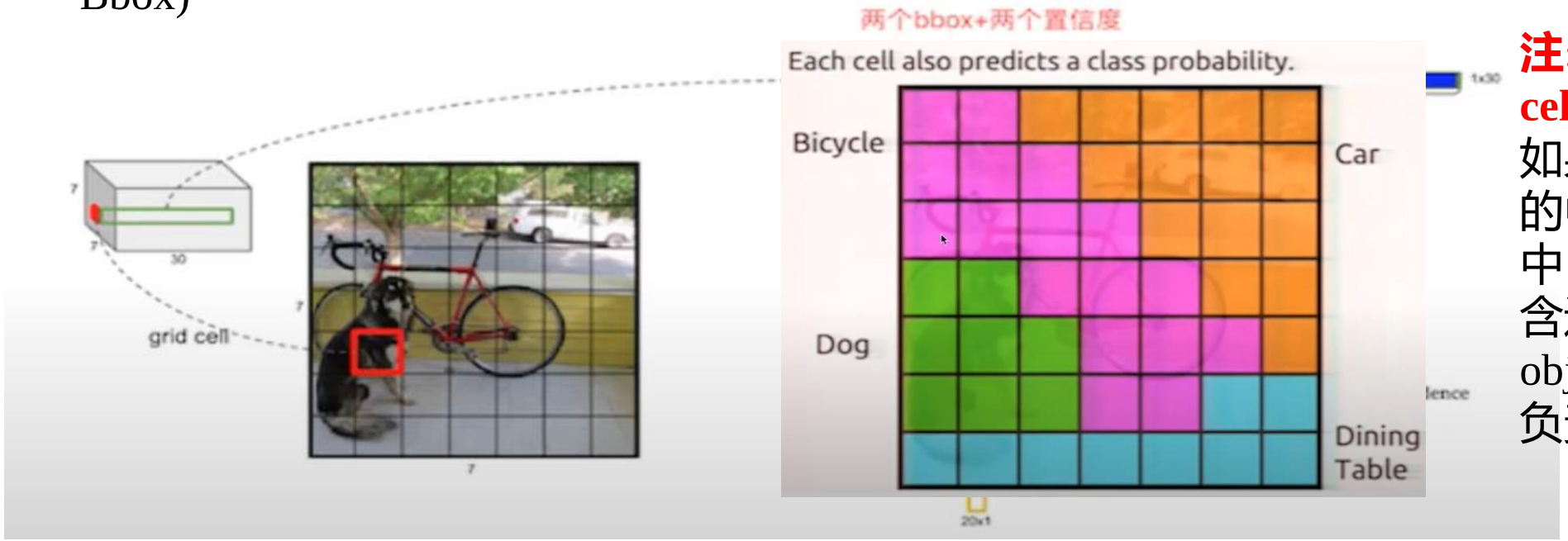
YOLO网络结构

一个网格会预测两个Bbox，在训练时我们只有一个Bbox专门负责(一个object一个Bbox)

判断准则：通过置信度大小比较

每个bounding box都对应一个confidence score

- 如果grid cell里面没有object，confidence就是0
- 如果有，则confidence score等于预测的box和ground truth的IOU乘积



注:所以如何判断一个grid cell中是否包含object呢?

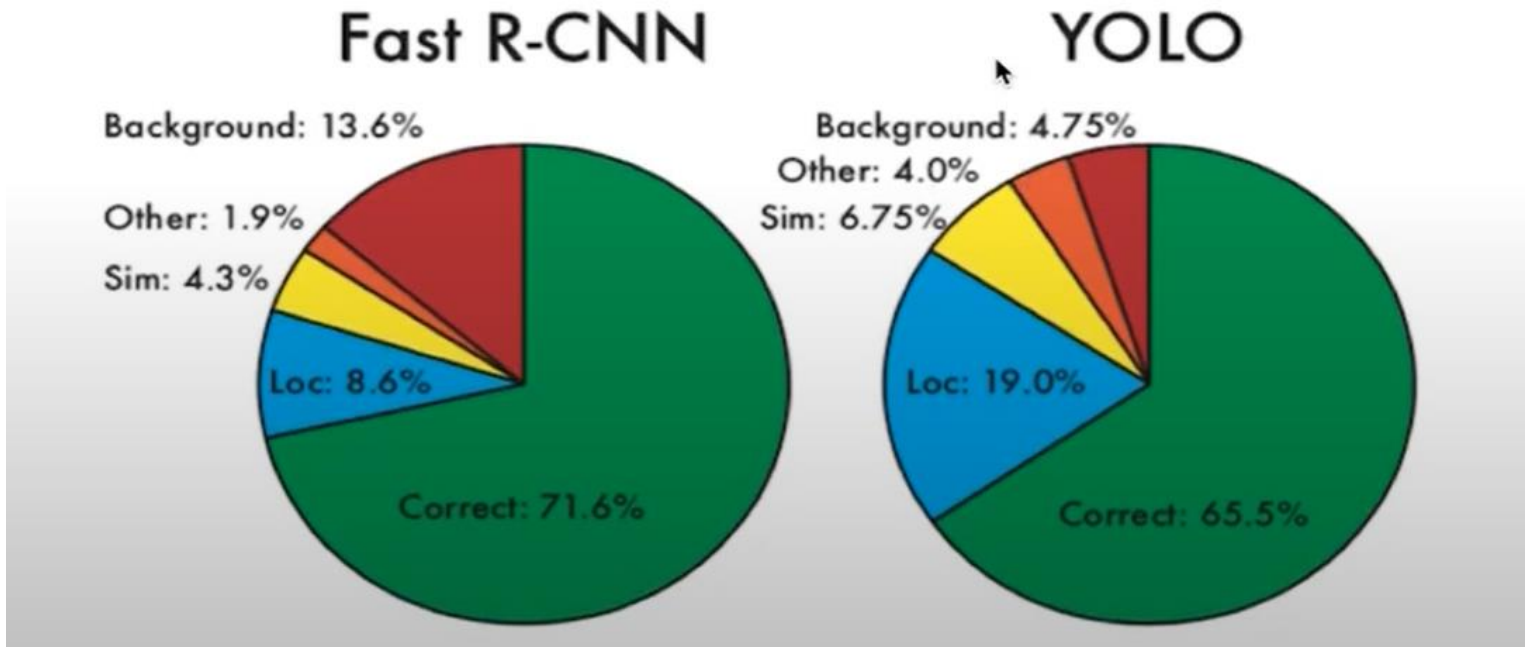
如果一个object的ground truth的中心点坐标在一个grid cell中，那么这个grid cell就是包含这个object，也就是说这个object的预测就由该grid cell负责。

这个概率可以理解为不属于任何一个bbox，而是属于这个单元格所预测的类别。

五、YOLO目标检测算法

YOLO网络结构

Faster R-CNN利用RPN网络与真实值调整了候选区域，然后再进行候选区域和卷积特征结果映射的特征向录的处理来通过与真实值优化网络预测结果。
而这两步在YOLO当中合并成了一个步骤，直接网络输出预测结果进行优化。所以经常也会称之为YOLO算法为直接回归法代表。YOLO的特点就是快



优点：速度快

不足：对相互靠的很近的物体(挨在一起且中点都落在同一个格子上的情况)，：还有很小的群体检0测效果不好，这是因为一个网格中只预测了两个框。

五、YOLO目标检测算法

YOLO-V5 安装

虚拟机环境 Ubuntu 20.04.1 LTS
python3.8
pytorch 1.6 ~ 1.7

step1: 下载 pytorch 1.7 和 torchvision 0.8.0

```
git clone -b 1.7 --recursive https://github.com/pytorch/pytorch pytorch_1.7
cd pyorch_1.7 && git submodule update --init --recursive && cd -
git clone -b release/0.8.0  https://github.com/pytorch/vision.git vision_0.8.0
```

step2: 编译代码

```
cd pytorch_1.7
pip3 install -r requirements.txt
NO_CUDA=1 python3 setup.py install
cd ../vision_0.8.0
python3 setup.py install
```

step3: 验证环境是否成功

```
python3 test_torch.py
```

```
ubuntu@ubuntu:~/yolo$ python3 test_torch.py
2.1.0+cu121
0.16.0+cu121
12.1
True
False
```

五、YOLO目标检测算法

YOLO-V5 安装

step4: 下载yolov5 代码

git clone https://github.com/ultralytics/yolov5

pip install -U -r requirements.txt

step5: 运行检测

python3 detect.py --source data/images/ --weights yolov5s.pt --conf 0.4 --device cpu --img-size 320

```
ubuntu@ubuntu:~/yolo/yolov5$ python3 detect.py --source data/images/ --weights yolov5s.pt --conf 0
.4 --device cpu --img-size 320
detect: weights=['yolov5s.pt'], source=data/images/, data=data/coco128.yaml, imgsz=[320, 320], con
f_thres=0.4, iou_thres=0.45, max_det=1000, device=cpu, view_img=False, save_txt=False, save_csv=Fa
lse, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=Fal
se, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3
, hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5 🚀 2023-10-19 Python-3.8.10 torch-2.1.0+cu121 CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
image 1/2 /home/ubuntu/yolo/yolov5/data/images/bus.jpg: 320x256 3 persons, 1 bus, 39.0ms
image 2/2 /home/ubuntu/yolo/yolov5/data/images/zidane.jpg: 192x320 2 persons, 37.3ms
Speed: 0.2ms pre-process, 38.1ms inference, 0.6ms NMS per image at shape (1, 3, 320, 320)
Results saved to runs/detect/exp
```

五、YOLO目标检测算法

Jetsion Tx2 安装

1.环境安装Yolov5：使用deepstream加速，视觉算法Jetson端优化部署。

环境安装教程:https://github.com/guojianyang/cv-detect-robot/tree/main/yolov5-ros-deepstream

2.模型转换: pt文件→wts文件→engine文件。

模型转换教程:https://github.com/wang-xinyu/tensorrtx/tree/yolov5-v5.0/yolov5

3.使用方式
在 /opt/nvidia/deepstream/deepstream-5.0/sources/yolov5-ros/ 目录下
1)开始识别
deepstream -app -c source1_usb_dec_infer_yolov5.txt
2)将数据传入ros
python Client-ros

4.相机属性查询
教程： https://www.xpstem.com/tutorials/lesson/10516
source1_usb_dec_infer_yolov5.txt中source0属性与其保持一致

五、YOLO目标检测算法

实验报告

除封面外，正文双面打印

实验2：手写数字识别

1. Alexnet网络原理
2. 百度平台配置步骤
3. 数据集运行结果
4. 实际数据运行结果
5. 总结（遇到的问题+心得体会）

实验3：基于YOLO的目标识别

1. YOLOV5算法原理
2. 系统环境
3. 安装步骤
4. 数据集运行结果
5. 实际数据运行结果
6. 总结（遇到的问题+心得体会）

谢谢！