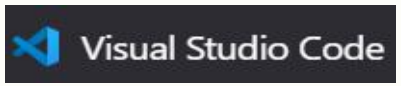
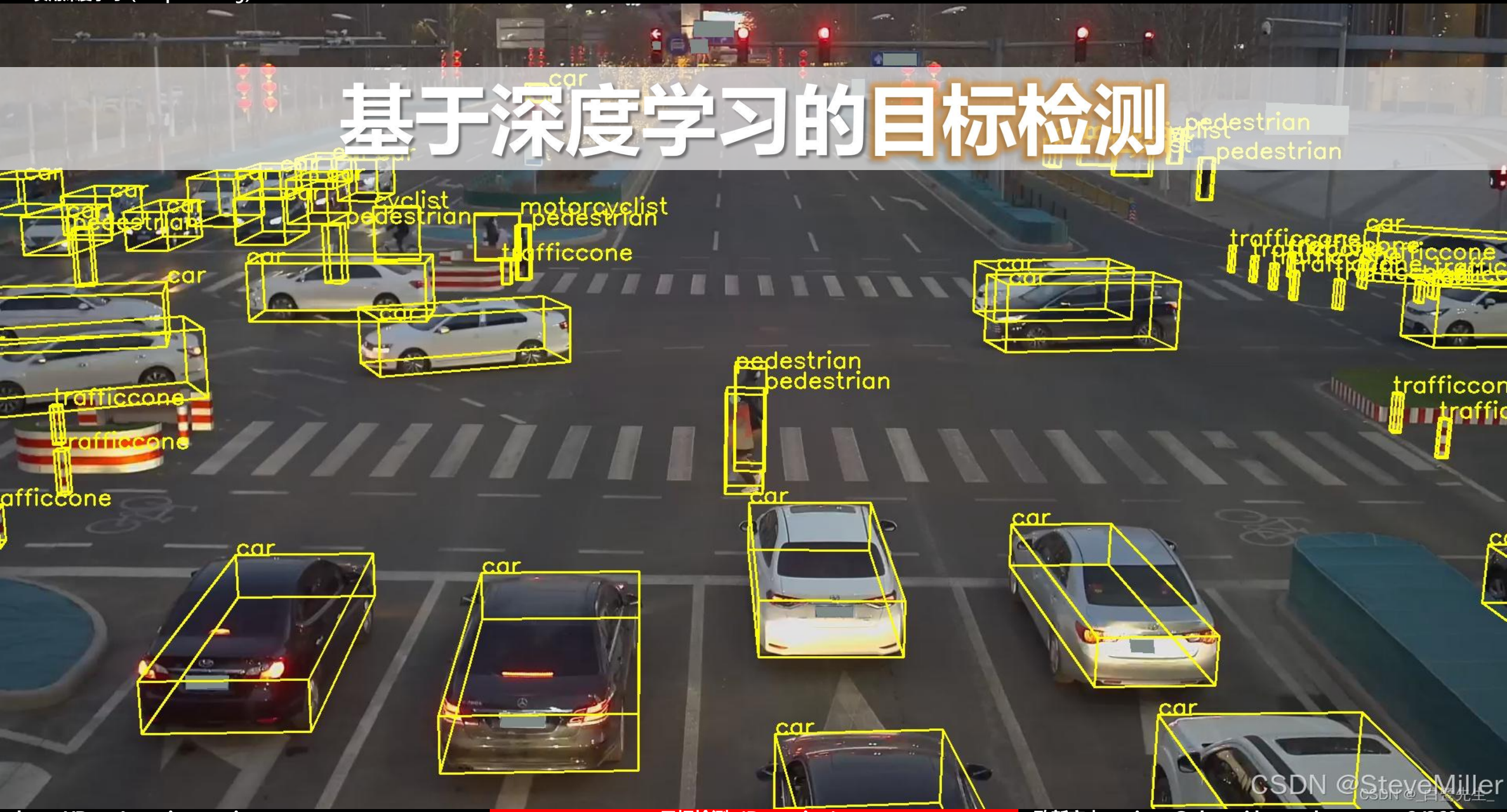
第11讲 目标检测

信息学院（智能应用研究院）
欧 新 宇



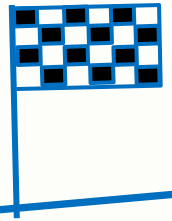
基于深度学习的目标检测



CSDN @SteveMiller

本章概要

- 目标检测概述
- 传统目标检测
- 目标检测关键技术
- 基于两阶段方法的目标检测
- 基于单阶段方法的目标检测
- 基于Anchor-Free的目标检测
- Beyond 2D



第09讲 基于深度学习的目标检测

计算机视觉四大任务

分 类
Classification

CAT

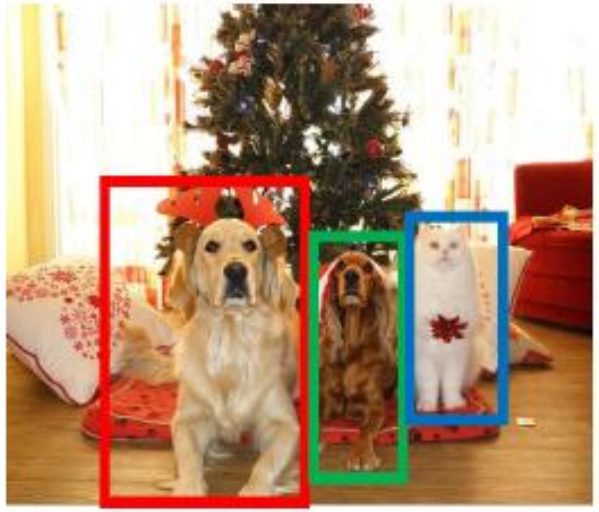
无空间概念

语义分割
Semantic
Segmentation

GRASS, CAT,
TREE, SKY

没有对象，只有像素

目标检测
Object Detection



DOG, DOG, CAT

多个目标对象

实例分割
Instance
Segmentation

DOG, DOG, CAT

This image is CC0 public domain

Part
01

目标检测概述

1.1 目标检测的定义

What is Object Detection?

目标检测（Object Detection）又称为物体检测，通常包含两个子任务，**一是**识别出图像中存在的物体，**二是**给出物体在图像中的位置（坐标）。

- 任务一，与图像分类基本一致，通常使用Softmax/SVM/二值交叉熵给出目标的类别；
- 任务二，给出目标的坐标位置，通常使用Smooth L1 Loss/L2 Loss去计算**边界框（Bouding-Box，BBox）**的置信度。边界框包含**四个参数(x, y, w, h)**，其中(w, h)是BBox的尺度，(x, y)是Bbox的位置，以图像的左上角为起点(0, 0)。

目标检测包括**单目标识别**和**多目标识别**两种，多目标识别需要同时给出所有目标的类别和坐标位置。人脸识别、行人检测、车辆识别、车辆计数、标识识别等都是目标检测的常见任务。

1.2 目标检测经典数据集和竞赛

重要竞赛和数据集简介

- **Pascal VOC**

  计算机视觉领域知名竞赛（2005-2012），包含人、车、猫、鸟等20类物体，Pascal VOC数据集目标较大，且每幅图片包含的目标数较少（<10）。http://host.robots.ox.ac.uk:8080/pascal/VOC/

- **MSCOCO**

  由微软和Facebook等企业联合发起，以场景理解为主要目标。包含91类，328,000张图片和2,500,000个标注。https://cocodataset.org/

- **自动驾驶数据集**

  KITTI、Carcraft、BDDV。

- **行人检测数据集**

  CUHK行人检测、INRIA行人数据集、DukeMTMC-reID、Market1501行人数据库、Caltech行人数据库、MIT行人数据库

1.2 目标检测经典数据集和竞赛

重要竞赛和数据集简介

数据集	类别数	train图片数，box数	Val图片数，box数	Boxes/Images
Pascal VOC 2012	20	5717, 1.3W+	5823, 1.3W+	2.4
COCO	80	118287, 4W+	5000, 3.6W+	7.3
Object365	356	600k, 9623k	38k, 47.9W+	16
OpenImages18	500	1643042, 86W+	10k, 69.6W+	7

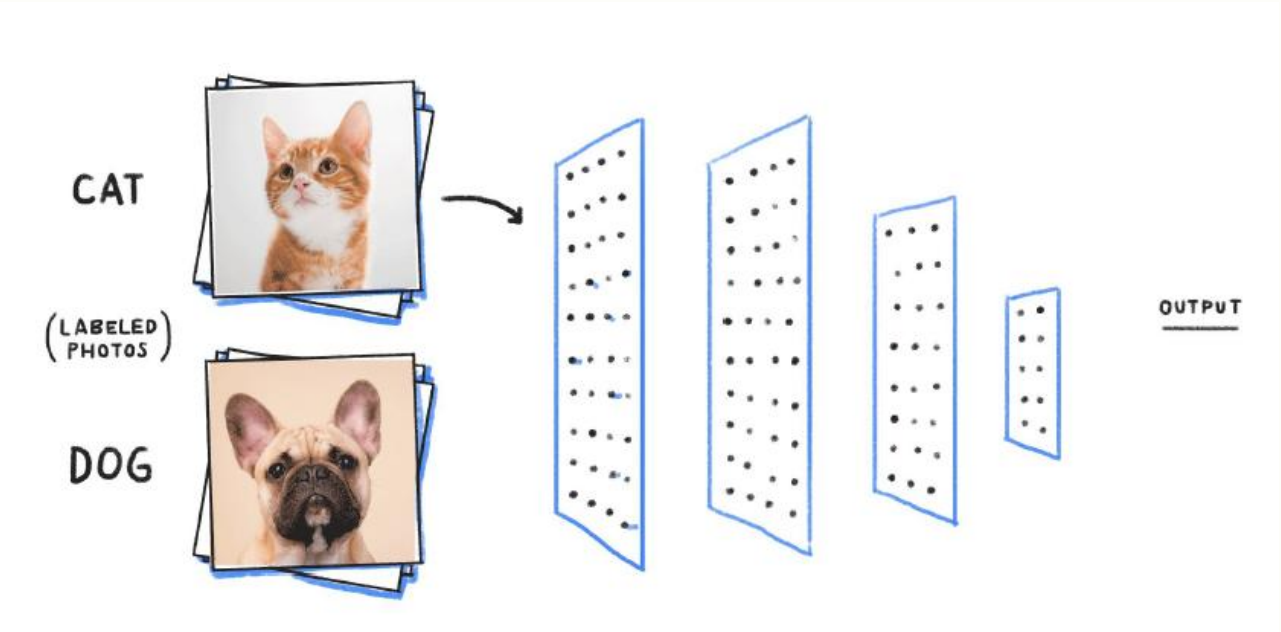
PaddleDetection提供了676类(Object365+OpenImages18)的预训练模型，包含大多数物体，可以直接使用或作为预训练模型提升业务精度。

1.3 目标检测的动机

图像分类

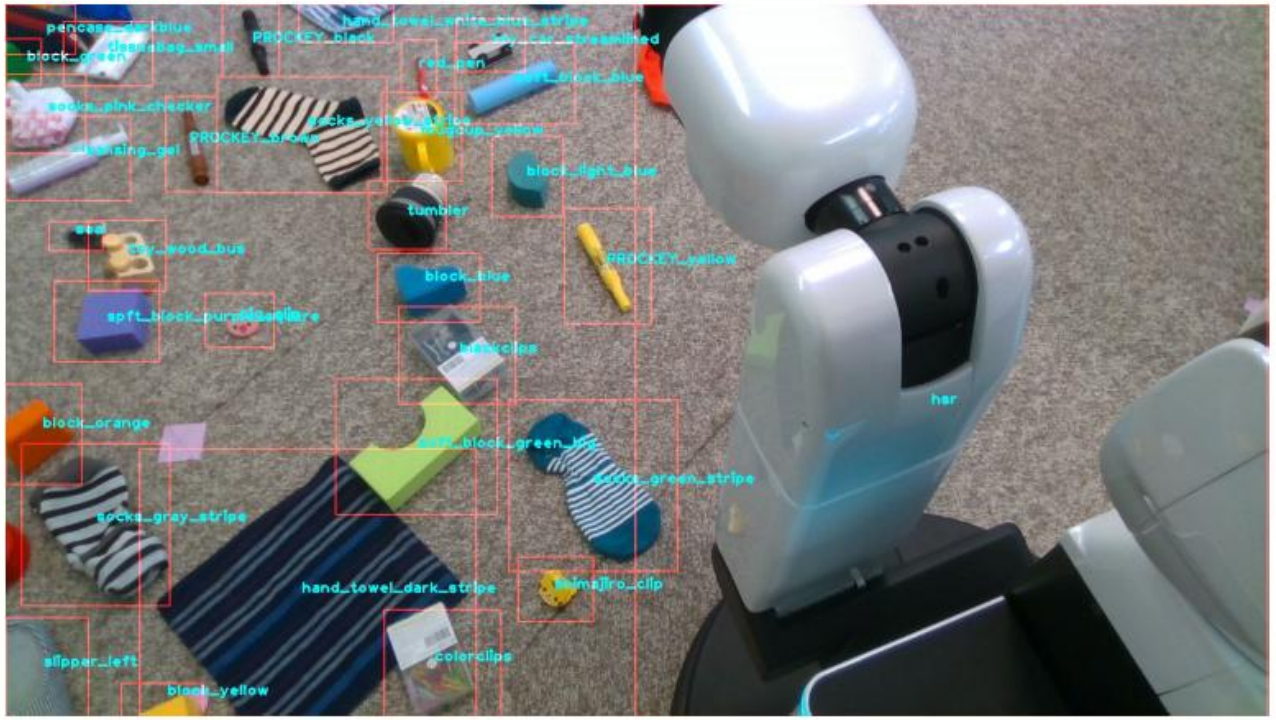
任务一：图像分类

- **输入**：图像
- **输出**：对象类别

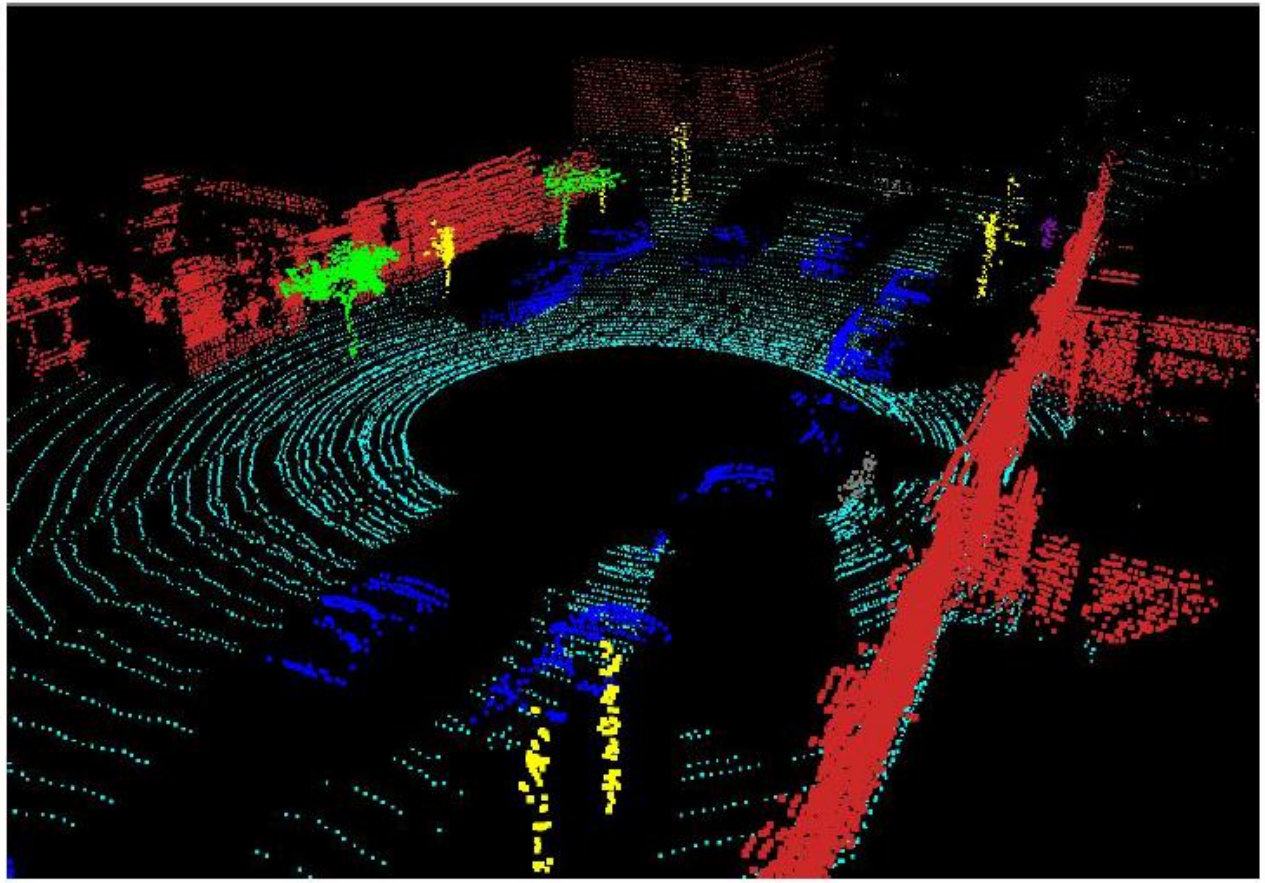


1.3 目标检测的动机

我们需要对复杂世界更高级的理解



目标检测：图像中所有对象的类别

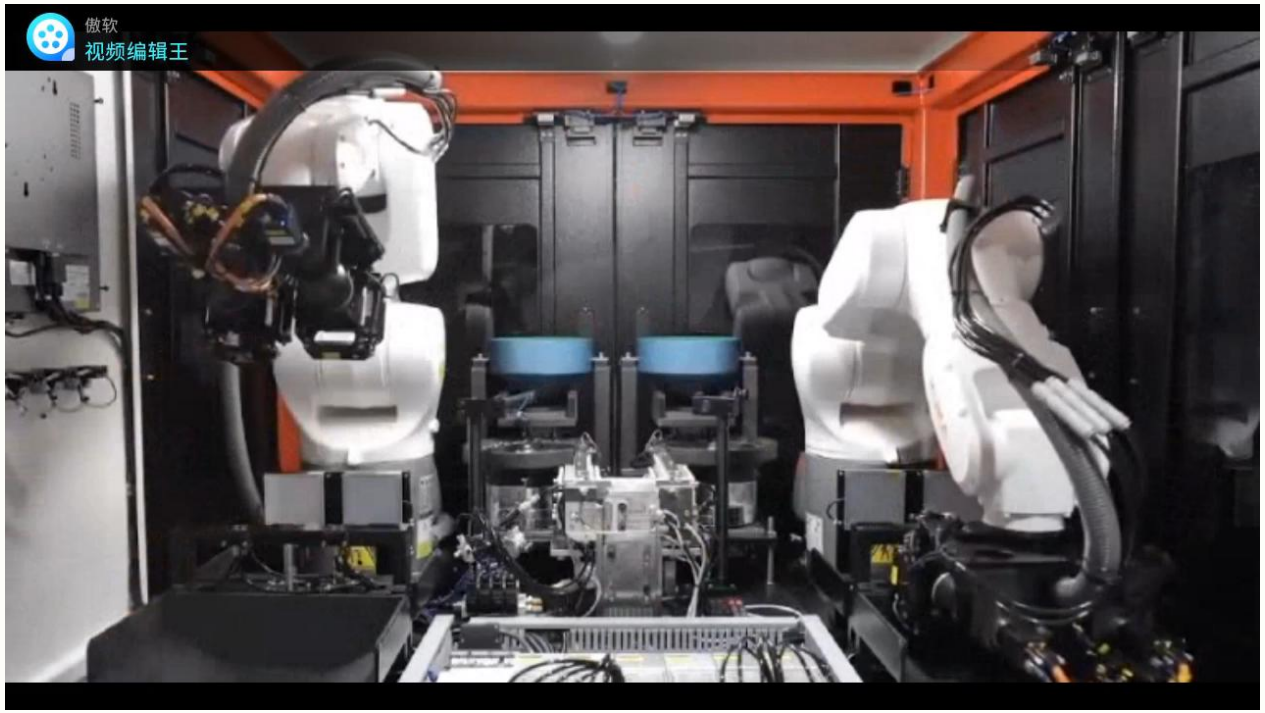


图像分割：所有像素的类别

1.3 目标检测的动机

我们也需要更多自动化的操作

商品检测



工业检测

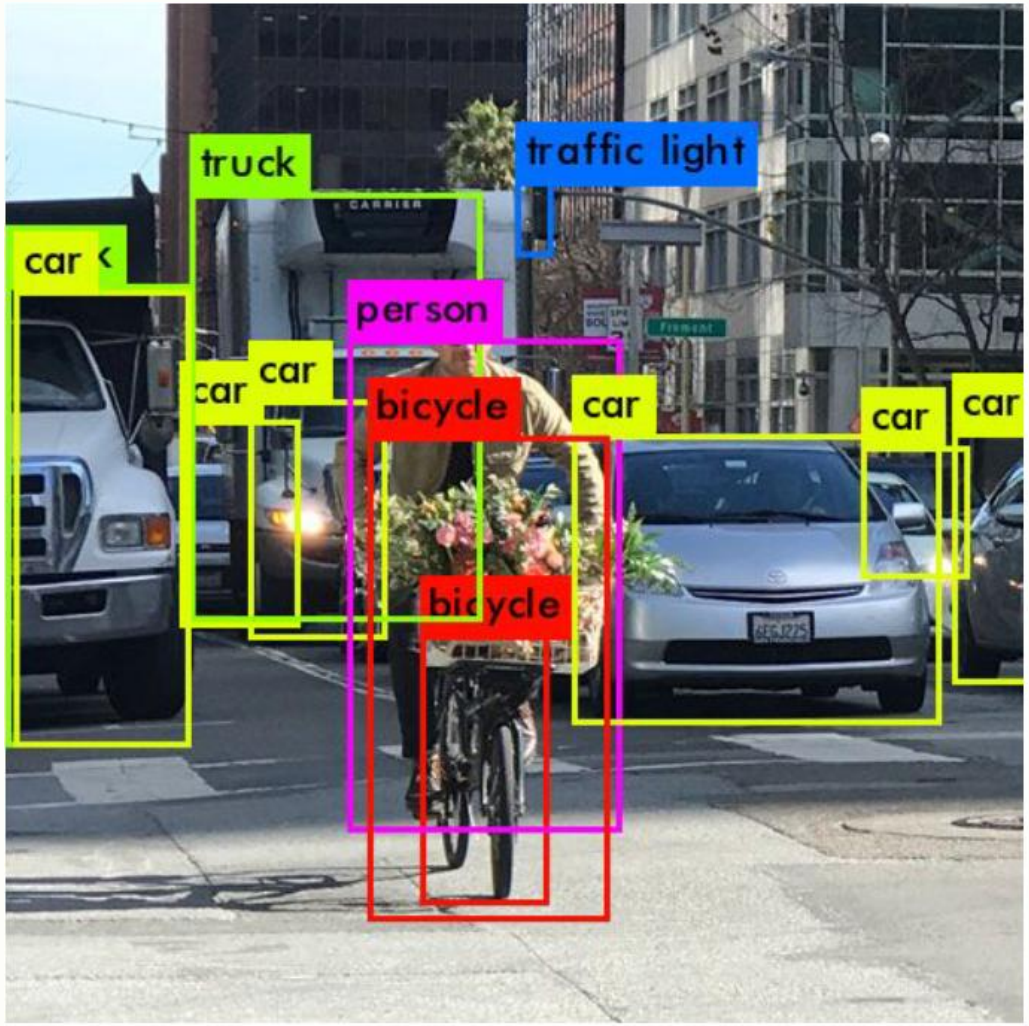
1.3 目标检测的动机

问题定义

任务二：有限制的图像分类

- **输入**：包含多个目标的图像
- **输出**：所有目标的信息
  - ✓ 目标对象的类别
  - ✓ 目标对象的定位信息（边界框）

=> 目标检测



1.3 目标检测的动机

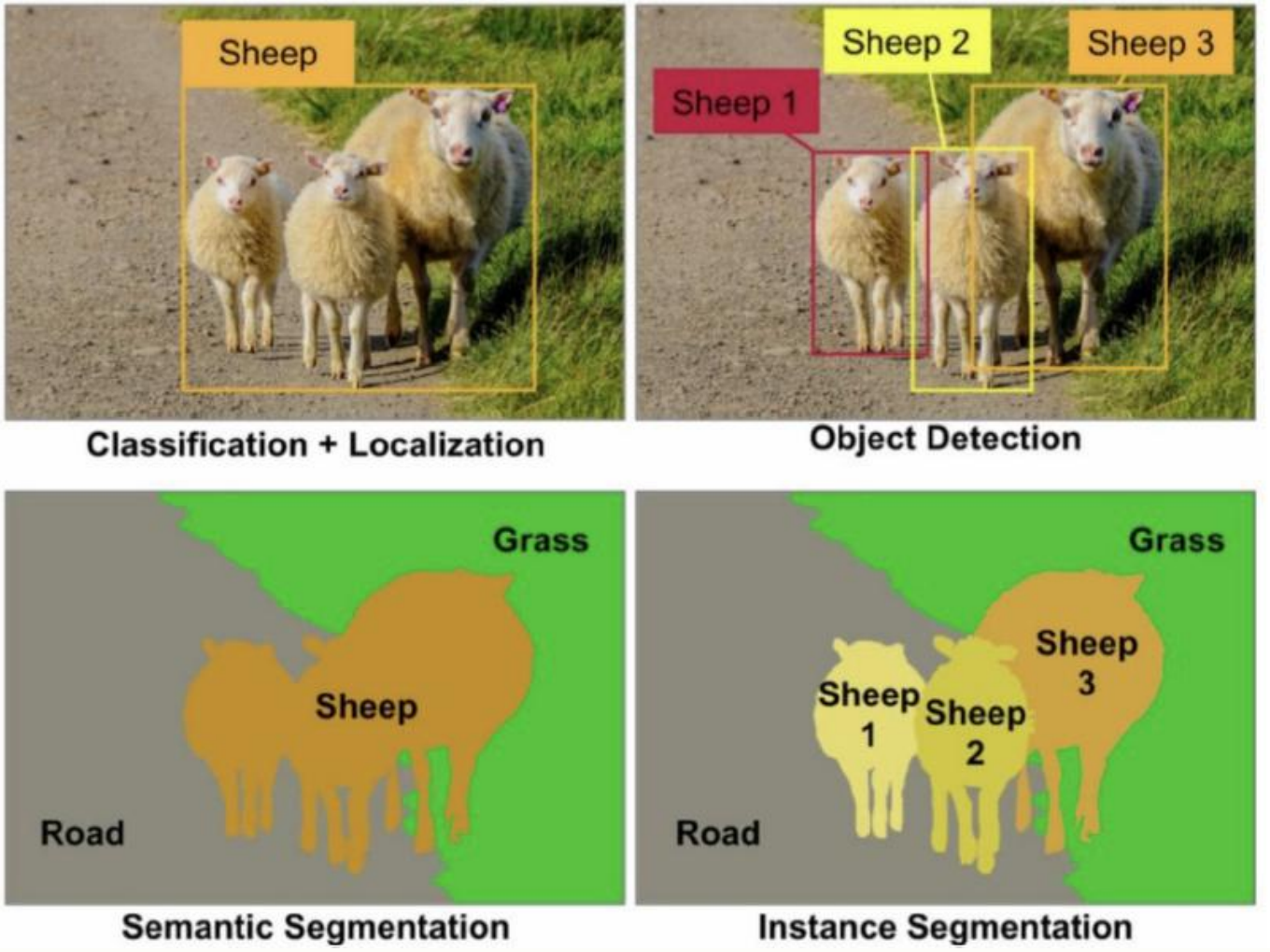
问题定义

任务三：目标检测

- **输入**：包含多个目标的图像
- **输出**：多个**实例**的信息
  - ✓ 对象类别
  - ✓ 对象的定位信息（边界框）

实例 (Instance)：

对应于单个目标对象，而不是将一类对象看作是同一个语义类别。

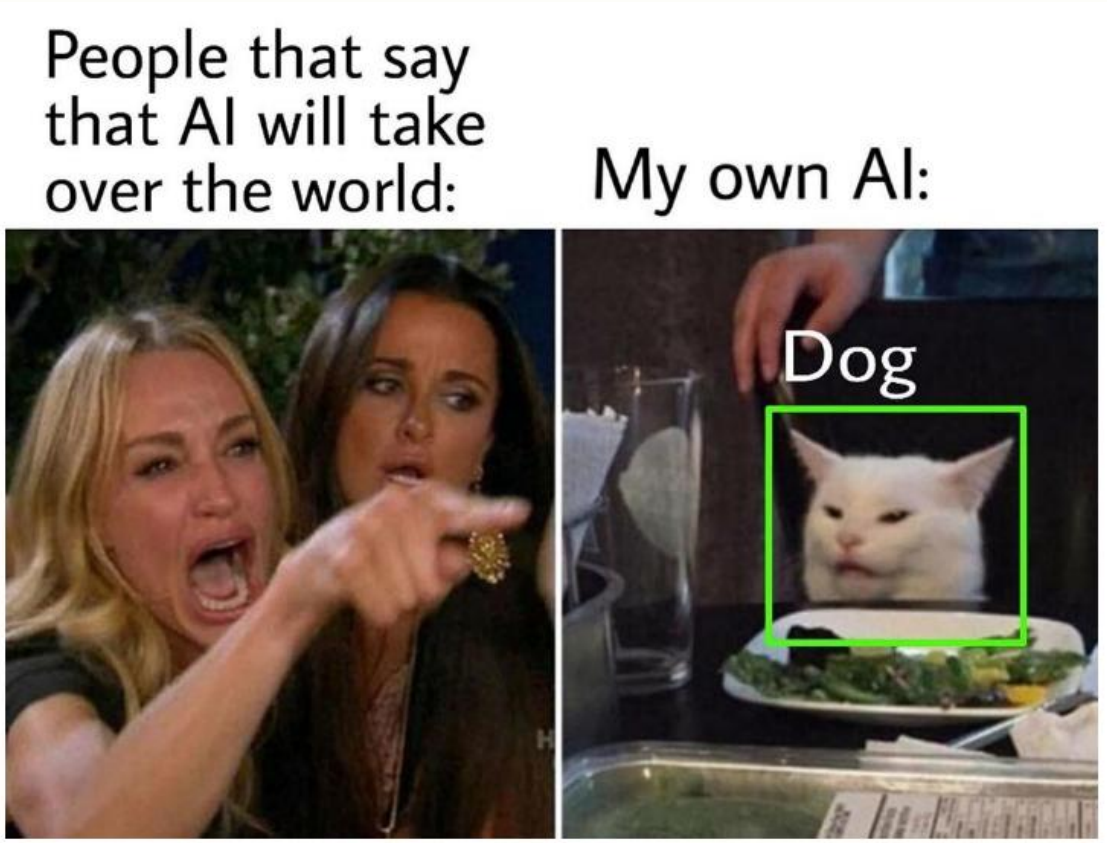


1.3 目标检测的动机

问题定义

任务三：目标检测

- **输入**：包含多个目标的图像
- **输出**：多个实例的信息
  - ✓ **对象类别**
  - ✓ 对象的定位信息（边界框）



对象类别（Object class）：

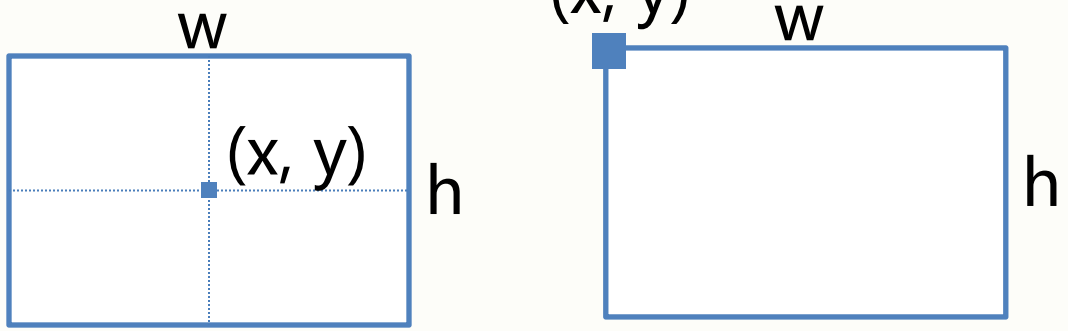
- **实例的语义类**
- **与图像分类相似，对象类别也会预测一个分数向量**

1.3 目标检测的动机

问题定义

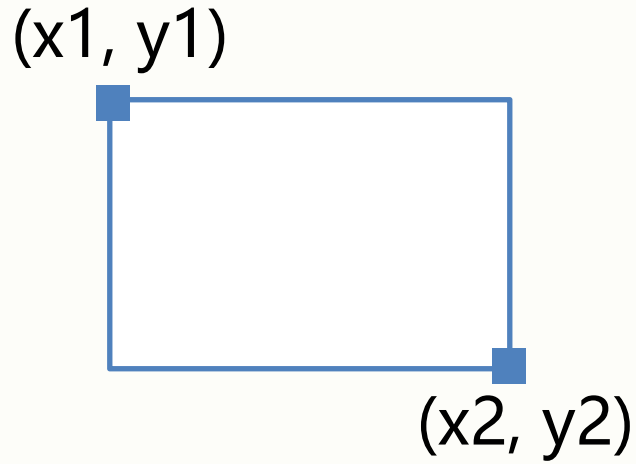
任务三：目标检测

- **输入**：包含多个目标的图像
- **输出**：多个实例的信息
  - ✓ 对象类别
  - ✓ **对象的定位信息（边界框）**



边界框（Bounding box, BBox）：

- **限制实例位置的刚性框**
- **多种参数定义**
  - ✓ **(center x, center y, width, heigh)**
  - ✓ **(x, y, width, heigh)**
  - ✓ **(x1, y1, x2, y2)**



1.4 现代目标检测的体系结构

经典目标检测体系结构

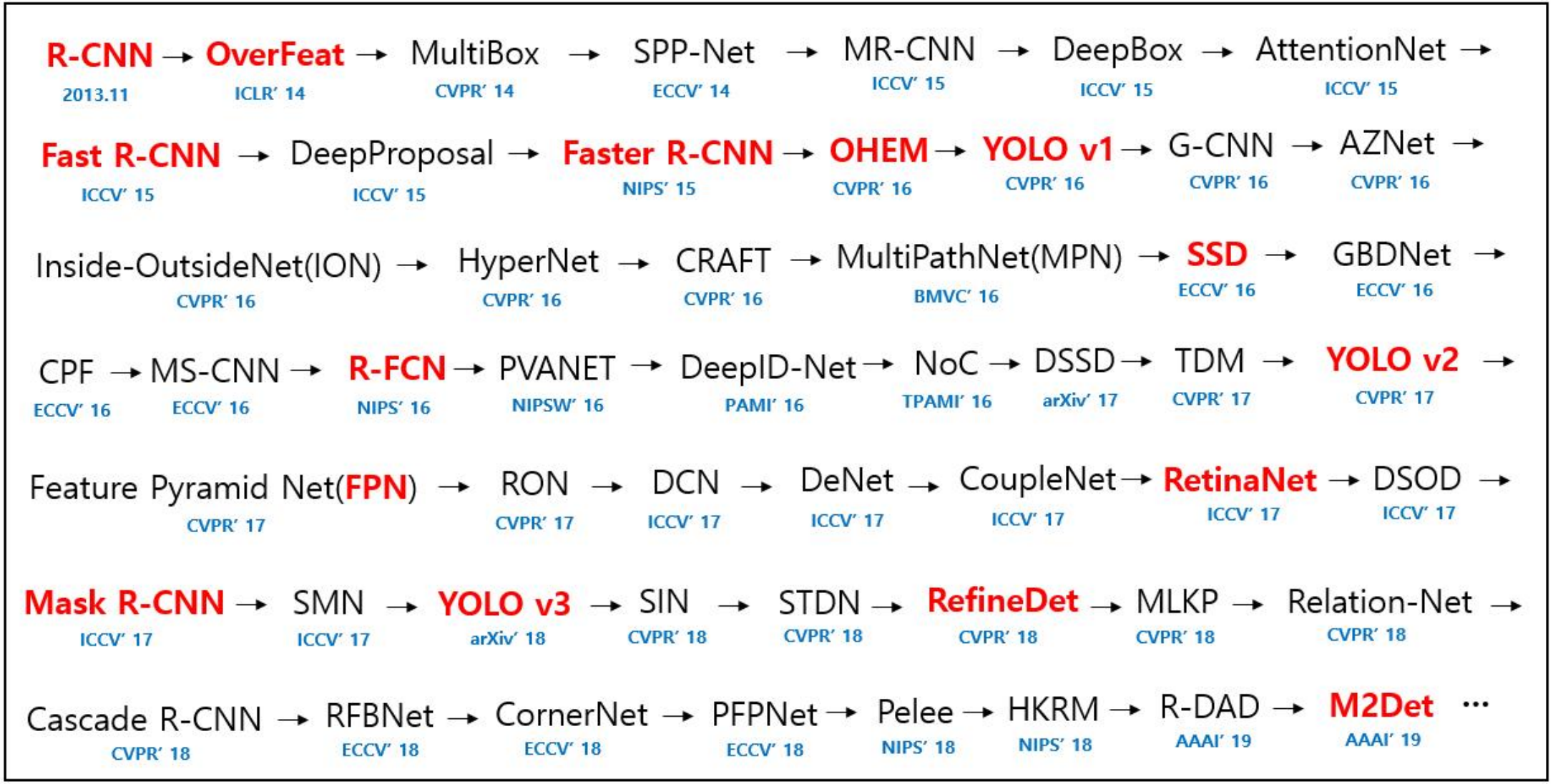
2014-1017年前后多项重要的工作奠定了现代目标检测的框架（基于卷积神经网络的目标检测）。

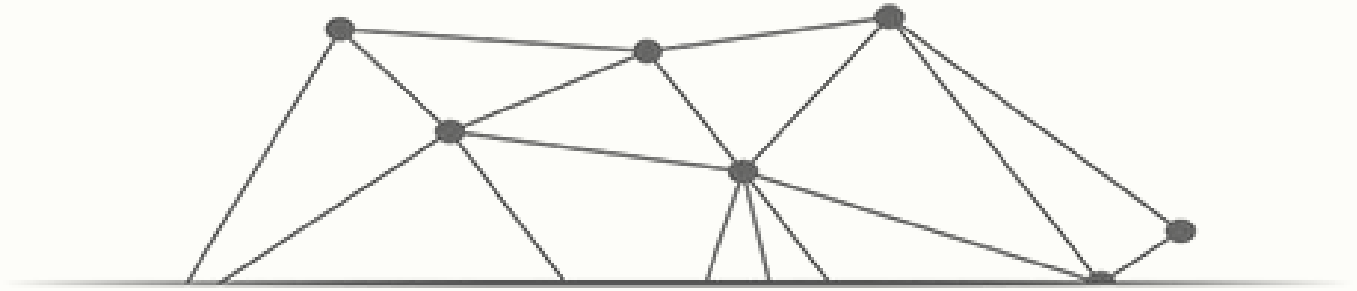
- R-CNN
- SPP-Net
- Fast RCNN
- Faster RCNN
- SSD
- YOLO(v2, v3)，PP-YOLO
- RetinaNet
- R-FCN

1.4 现代目标检测的体系结构

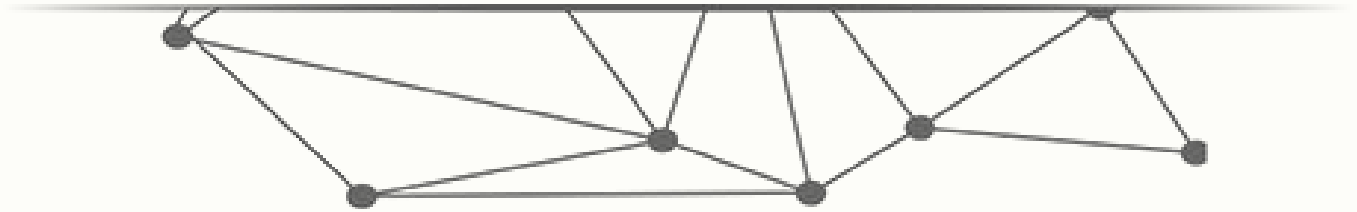
目标检测也是一个江湖

Detection History





课堂互动 13.2.1



1.5 评价指标 （Evaluation Metrics）

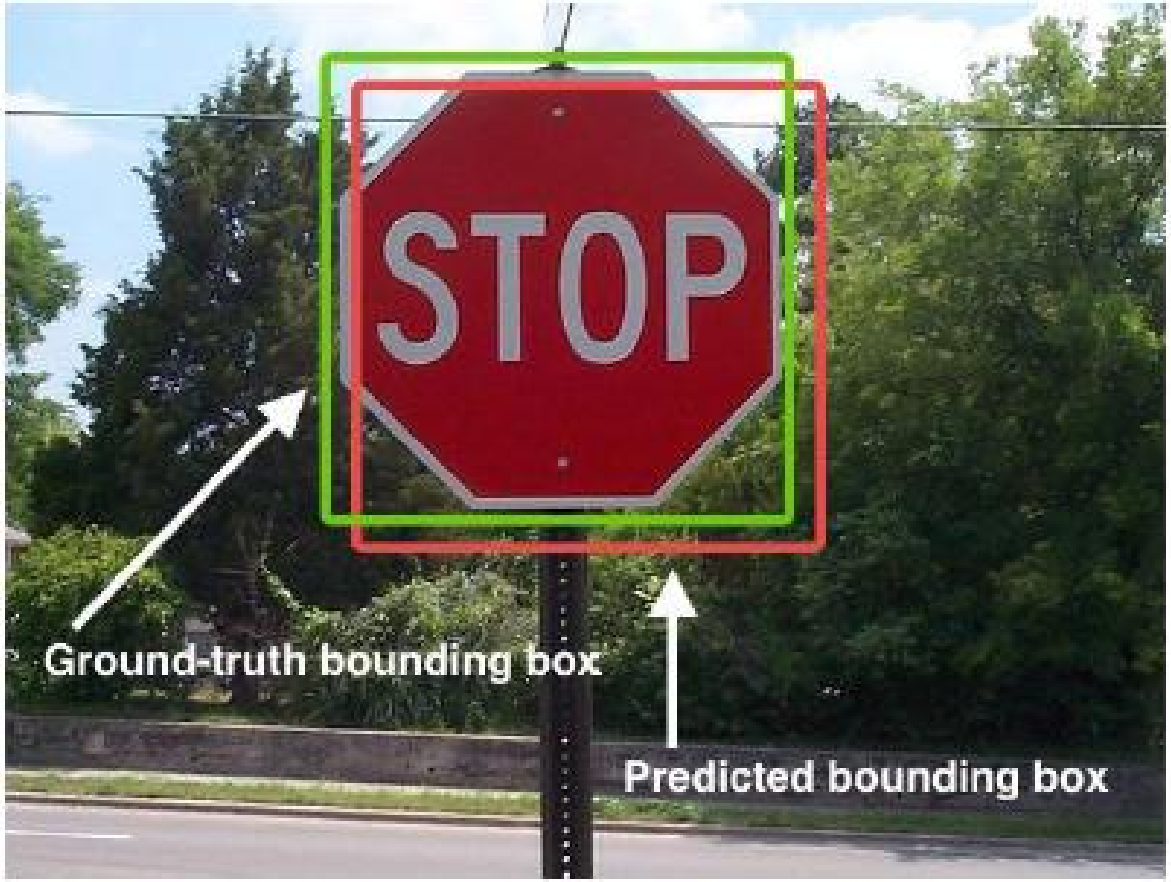
目标检测 – 单目标

给定：

- 单个Ground-truth边界框
- 单个预测边界框

输出：

- 评价结果
- **–>如何评价输出?**



1.5 评价指标 （Evaluation Metrics）

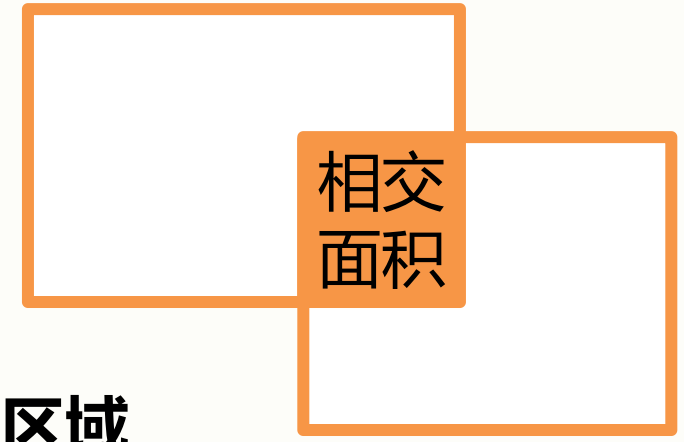
目标检测 – 单目标

给定：

- 单个Ground-Turth边界框
- 单个预测边界框

输出：

- 评价结果
- **–>如何评价输出?**



$$IoU = \frac{\text{重叠的区域}}{\text{相并的区域}}$$

交并比

(Intersection over Union，IoU)

1.5 评价指标 （Evaluation Metrics）

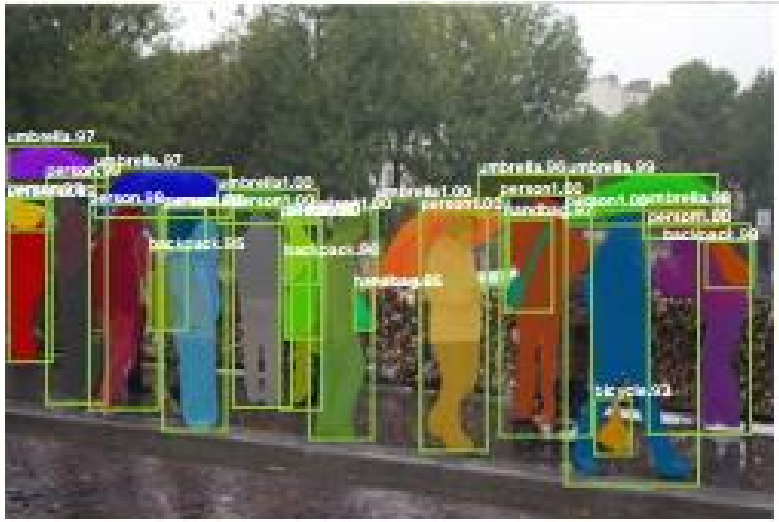
目标检测 – 多目标

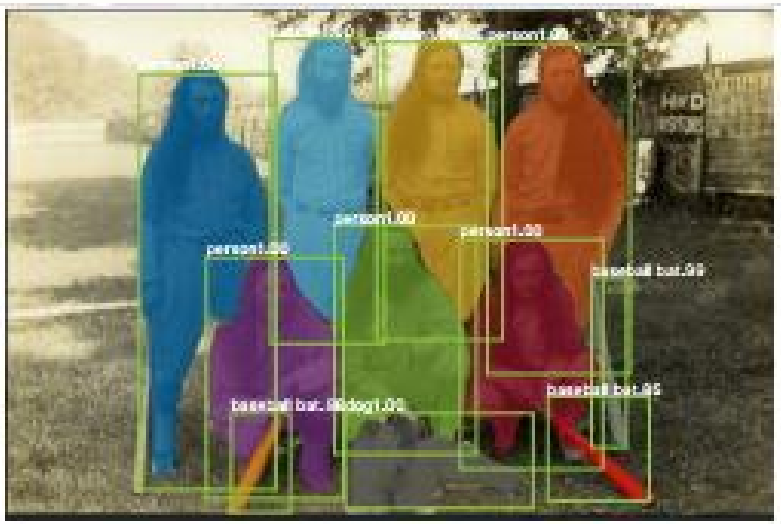
给定：

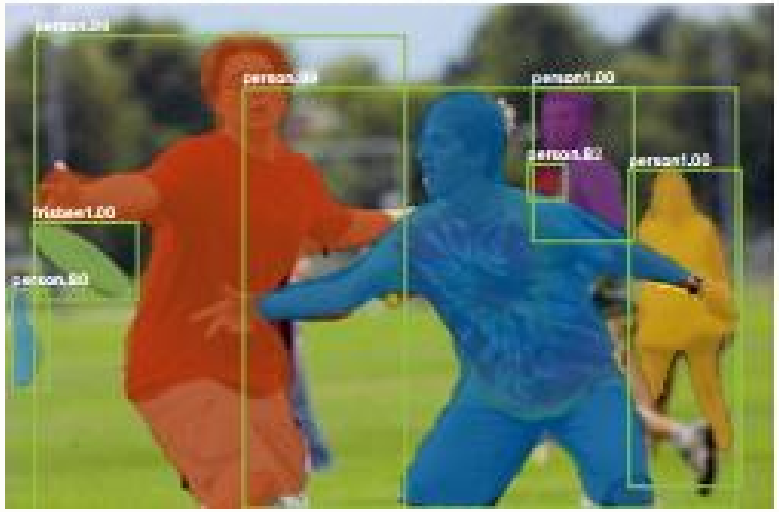
- 多个Ground-Turth边界框
- 多个预测边界框

输出：

- **每个目标都输出一个IoU**
- **平均IoU（mIoU）**







1.5 评价指标 （Evaluation Metrics）

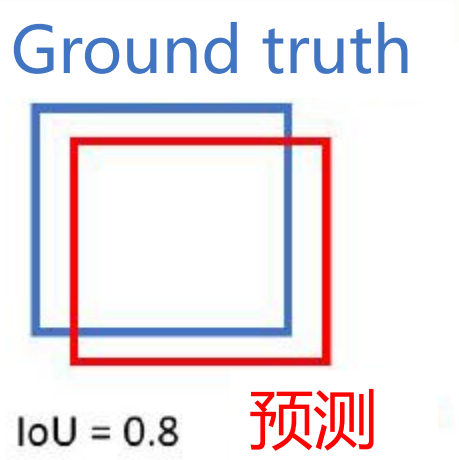
IoU的计算方法

匹配：以下条件都满足时为**真（True）**，记为**正确项**

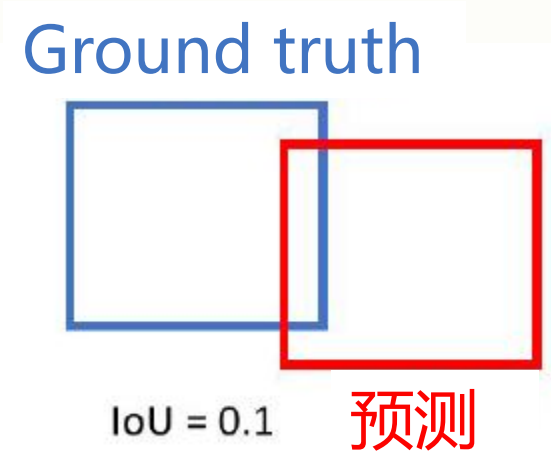
1. 基于Ground-truth box和预测box的IoU高于一个确定的阈值(threshold)
2. Ground-truth类别和预测的类别是相同的
3. 仅考虑1对1的匹配

当匹配结果为假（False）时，记为错误项。

Example
Threshold=0.5



真 (正确)



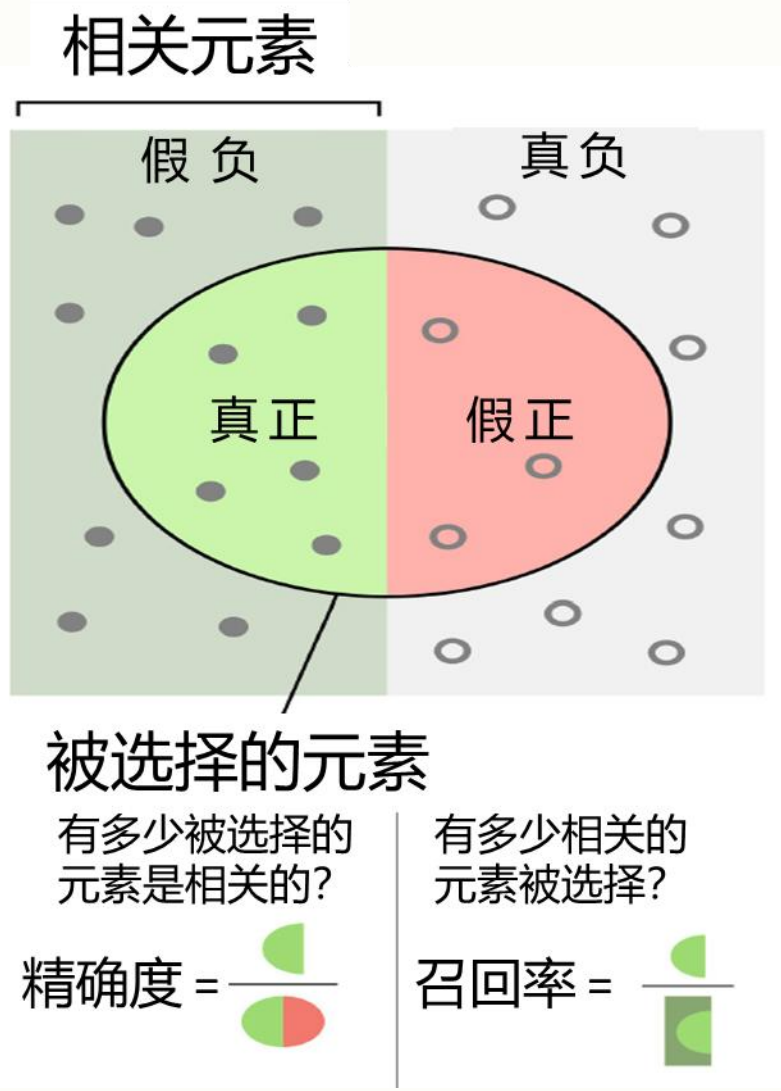
假 (错误)

1.5 评价指标 （Evaluation Metrics）

评价指标 – Precision和Recall

以狗识别为例，描述四个指标：

- **真正（True positive, TP）**：预测为狗，实际标注也是狗
- **假负（False negative, FN）**：预测不是狗，但实际标注是狗
- **假正（False positive, FP）**：预测为狗，但实际标注不是狗
- **真负（True negative，FN）**：预测不是狗，实际也不是狗



- 精确度(Precision) = TP / （TP+FP）

预测为正的样本中有多少是真正的正样本，例：预测为狗且真的为狗的样本数(TP)占预测为狗的样本数(TP+FP)的比例。

- 召回率(Recall) = TP / （TP+FN）

预测为正且真的为正的样本占所有正确样本的比例，例：预测为狗且真的为狗的样本数(TP)占所有标注为狗(GroundTruth)的样本的比例。

1.5 评价指标 （Evaluation Metrics）

评价指标 – Precision和Recall

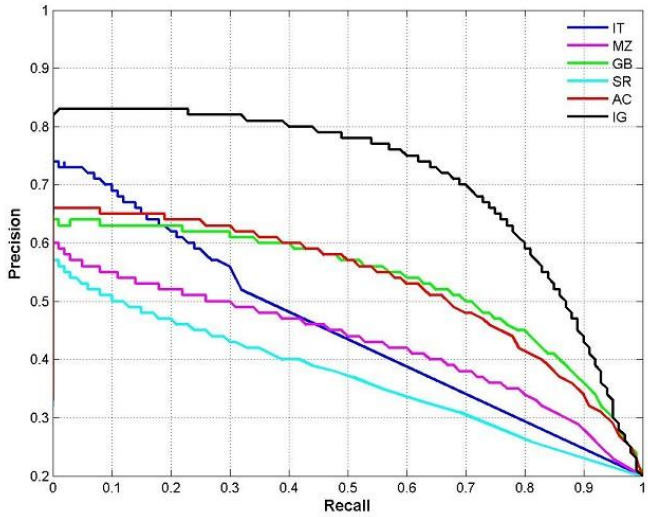
例：小刘使用某个猫狗识别数据集训练了一个模型A，该数据集包含300只狗和200只猫。假设使用模型A进行目标检测，总共检测出150只狗，其中100只的真实标签是狗，50只的真实标签是猫；同时检测出350只猫，其中200只的真实标签是狗，150只的真实标签是猫。试问，模型A的关于狗的精确度、召回率、准确率分别是多少?

- ✓ 精确度 = TP/(TP+FP) = 100/150 = 66.7%
- ✓ 召回率 = TP/(TP+FN) = 100/300 = 33.3%
- ✓ 准确率 = 预测对的/所有= (TP+TN)/Total = (100+150)/500 = 50%

- ✓ **模型A的综合准确率 = (关于狗的准确率+关于猫的准确率)/2**

1.5 评价指标 （Evaluation Metrics）

评价指标 – AP平均精度、mAP平均精度均值、F1-Score



Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666

- **P-R曲线：**以Precision，Recall为纵坐标和横坐标的曲线
- **平均精度(Average Precision, AP)：某一类P-R曲线下的面积**

$$\text{平均精度} = \frac{\text{类别C中所有样本精确度}}{\text{类别C的总样本数}}$$

注意：每个数据集会产生n个平均精度（n=验证样本的数量）

- **平均精度均值(Mean Average Precision，mAP)**：所有类别AP的平均

$$\text{平均精度均值} = \frac{\text{所有类别平均精确度的总和}}{\text{类别数}}$$

- **F1分数（F1-Score）**

$$F_1 = 2\frac{\text{精确度}\cdot\text{召回率}}{\text{精确度} + \text{召回率}}$$

特别注意：F1-Score是精确度和召回率的平衡指标

1.5 评价指标 （Evaluation Metrics）

Pascal VOC mAP 和 MS COCO mAP

不同的数据集可能会给出不同mAP计算方法，这其中最典型的就是Pascal VOC mAP 和 MS COCO mAP。

- **Pascal VOC**

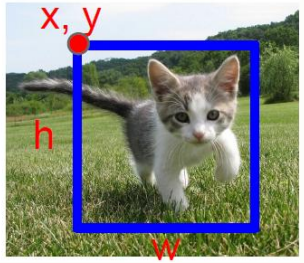
  Pascal VOC 2008中的IoU阈值为0.5，即预测框和真实框的IoU>0.5为正样本。

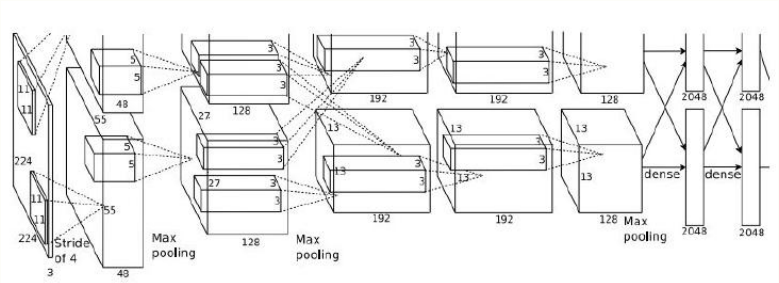
- **MS COCO**

  MS COCO采用区间阈值计算mAP，即设置IoU从[0.5-0.95]的区间上每隔0.05计算一次mAP的值，并取所有结果的平均值作为最终的结果。

1.6 基于卷积神经网络的目标检测

目标检测 – 单目标（分类+定位）





全连接网络
4096 ->1000维

类别分数
猫: 0.9
狗: 0.05
汽车: 0.01
...

Softmax Loss

正确标签：猫

多任务 Loss

Loss

特征向量
4096维

全连接网络
4096 -> 4维

边界框BBox
(x, y, w, h)

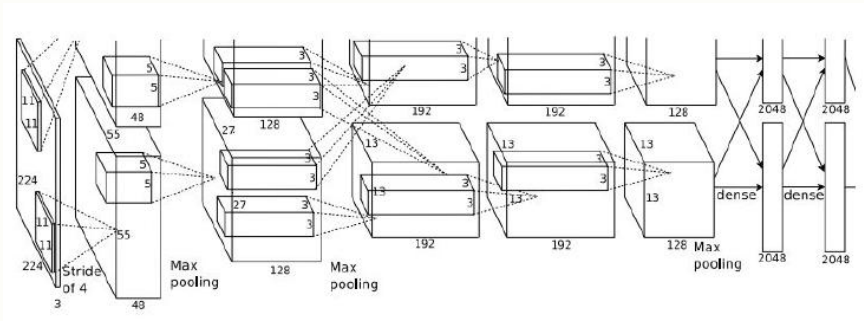
L2 Loss

正确边界框：
(x_0, y_0, w_0, h_0)

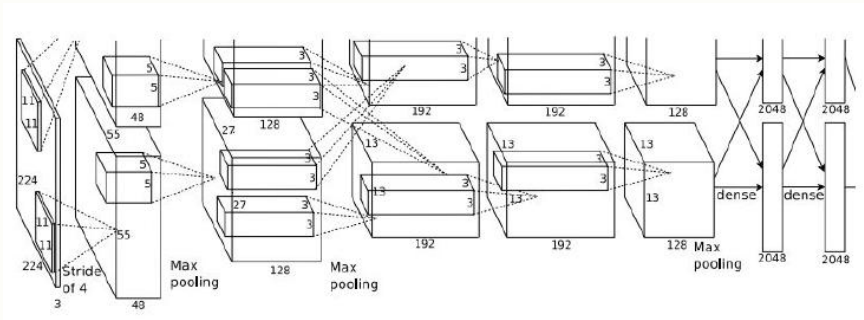
1.6 基于卷积神经网络的目标检测

目标检测 – 多目标

每幅图像都会产生不同数量的输出

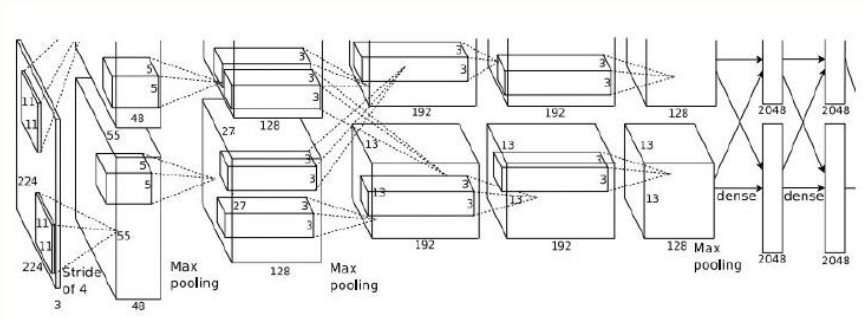


猫：(x, y, w, h)　5个输出（4+1）



狗：(x, y, w, h)
狗：(x, y, w, h)
猫：(x, y, w, h)

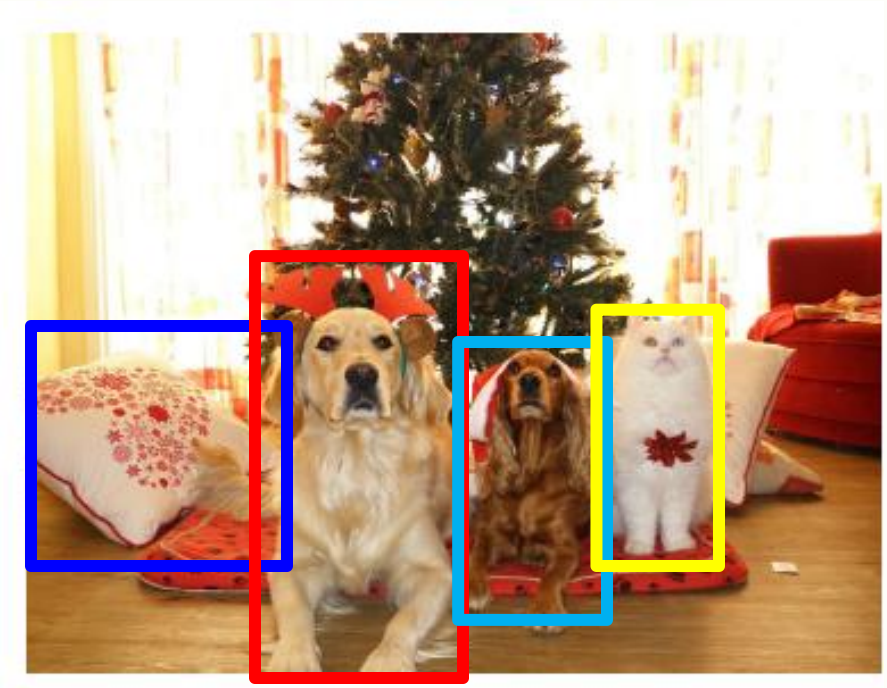
15个输出
3*(4+1)



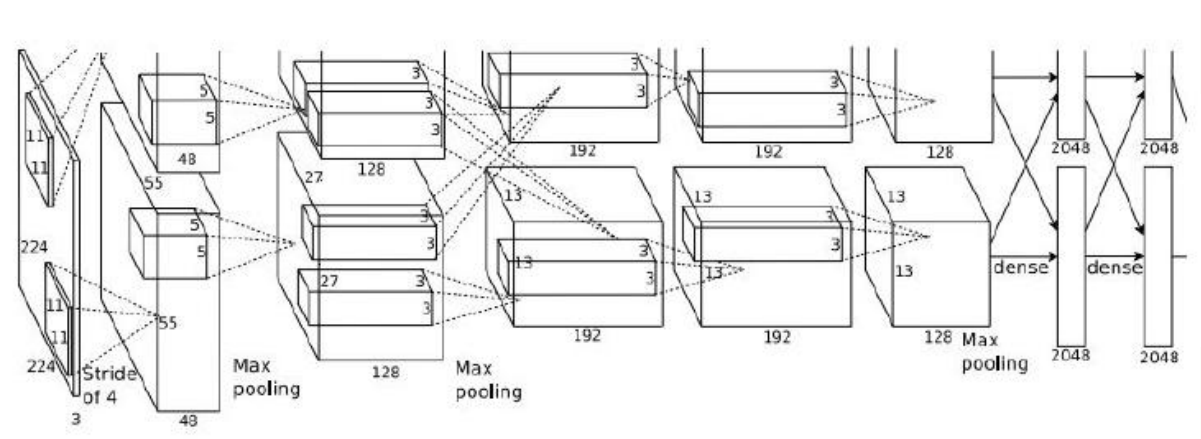
鸭子：(x, y, w, h)
鸭子：(x, y, w, h)　许多输出
鸭子：(x, y, w, h)
…

1.6 基于卷积神经网络的目标检测

目标检测 – 多目标



将CNN应用到图片的不同切片上，并判断它是否是某个类的样本(或者是背景).

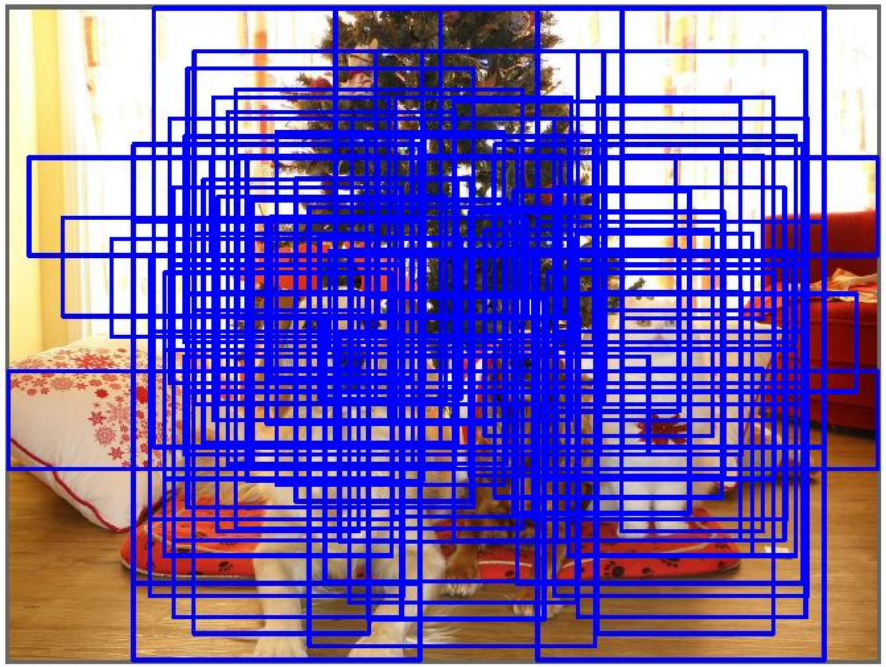


狗？ No
猫？ Yes
背景？ No

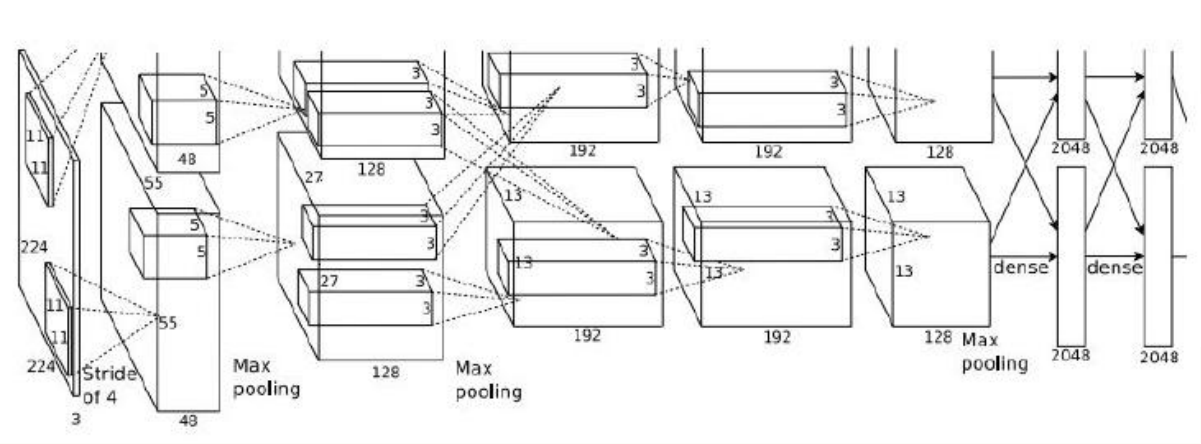
Q: 该方法是否存在什么问题?

1.6 基于卷积神经网络的目标检测

目标检测 – 多目标

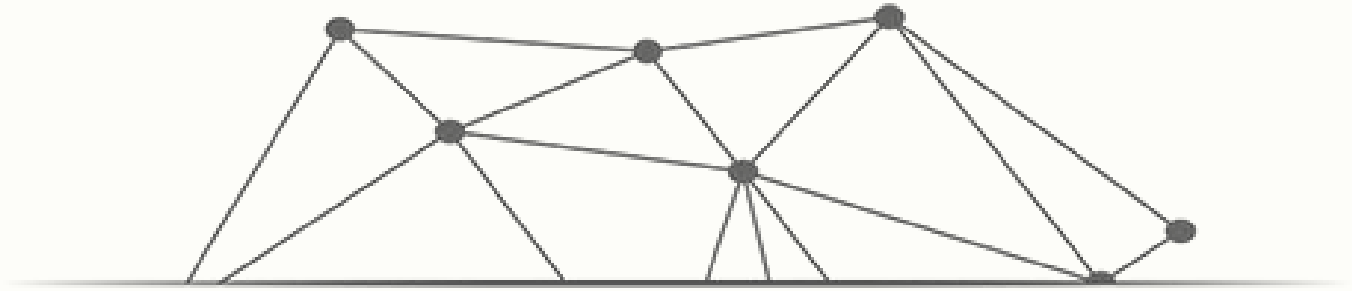


将CNN应用到图片的不同切片上，并判断它是否是某个类的样本(或者是背景).

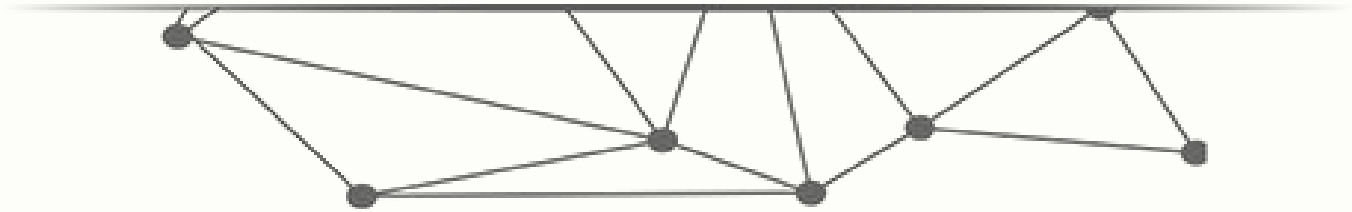


狗？ No
猫？ Yes
背景？ No

A：需要将CNN应用于大量的位置、尺度和长宽比。这种算法计算复杂度非常高！



课堂互动 13.2.2



Part
02

传统目标检测技术

2. 传统目标检测技术

传统目标检测基本步骤

候选区域生成

给出物体可能出现的建议区域。通常使用不同尺寸的滑动窗口，在给定图像的不同位置处选取候选区域。

滑动窗口
EdgeBox，Bing，
Selective Search

特征提取

对候选区域进行特征提取。包括颜色、纹理、空间特征、梯度等。

SIFT, DPM, LBP
HoG, Harr小波

分类器分类

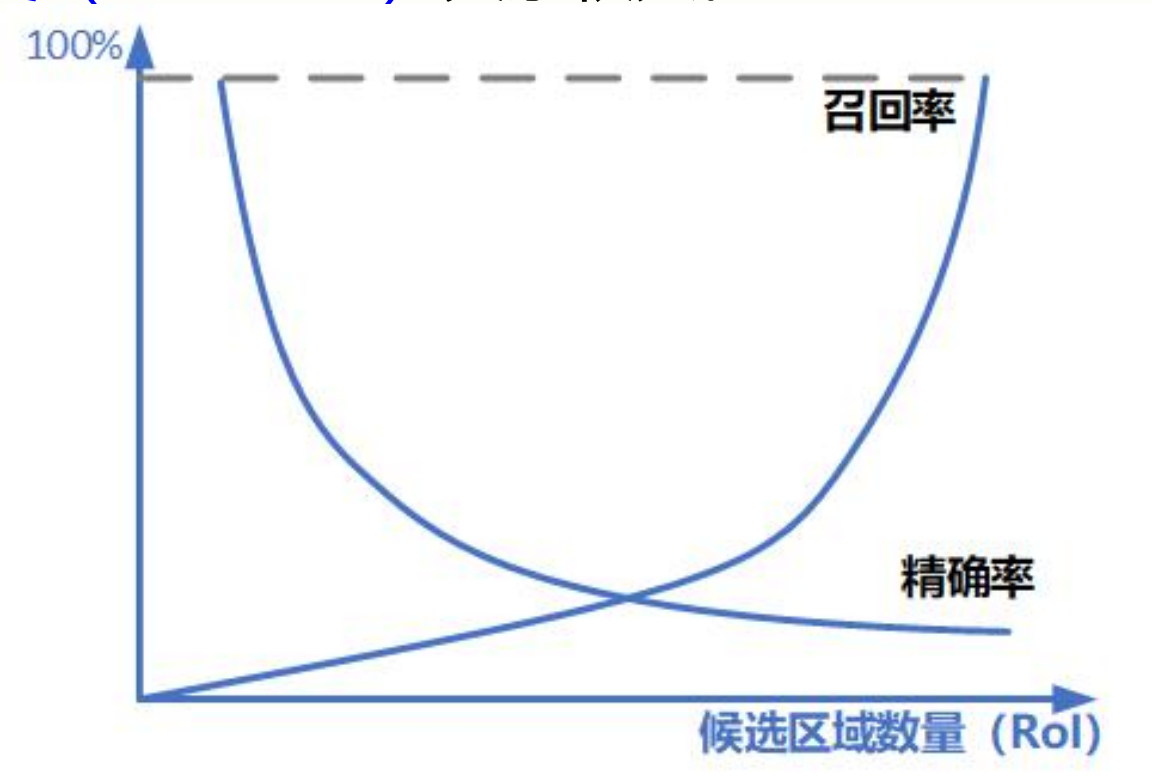
根据提取的特征使用一种分类方法对候选区域中的内容进行分类。

SVM, 随机森林
Adaboost, 决策树

2.1 候选区域生成

召回率与准确率的博弈

候选区域的生成也称为**区域提名（Region Proposal）**或区域建议，目标是尽可能准确地给出物品的可能位置**（Region of Interest, RoI）**，其结果与最终检测结果的召回率（Recall）和精确度（Precision）关系很大。



2.1 候选区域生成

常见区域建议算法

滑动窗口算法

逐像素进行扫描，产生大量RoI，召回率较高，精确度较低。

Bing

使用8×8的二进制计算梯度幅值实现封闭区域（对象）检测。该方法速度非常快，但性能相对较差。

Selective Search

综合多种颜色空间、纹理等信息，通过区域的逐渐聚合来获得检测目标。该方法准确率和精确度都较高

EdgeBox

利用边缘信息确定box内的轮廓数与box边缘重叠的边缘数实现目标检测。该方法速度较快，且召回率高和精确度较高。

2.2 特征提取

用于目标检测的常见的特征

在传统目标检测算法框架中，特征提取的好坏直接关系到后续分类结果的准确性。类似支持向量机（SVM）、随机森林、Adaboost等分类器的性能都比较优秀，因此目标检测的关键是特征的选择和提取。然而传统方法中提取的特征的专用性较强，且不容易被迁移。以下是常见特征：

- **人脸识别**：Harr小波
- **行人识别**：多尺度形变部件模型DPM，梯度方向直方图HoG，局部二进制模式LBP
- **通用识别**：尺度不变特征变换SIFT，Harr小波，边缘，颜色，纹理，角点等

2.3 传统目标检测存在的问题

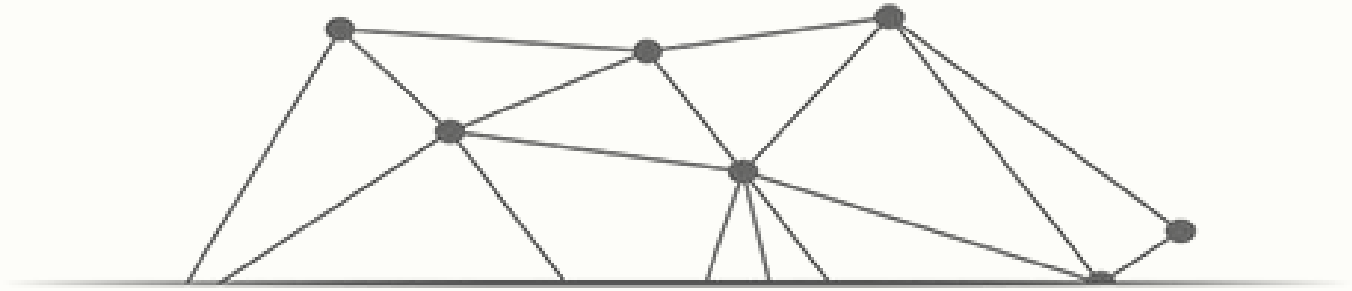
传统目标检测存在的问题

- **特征工程**

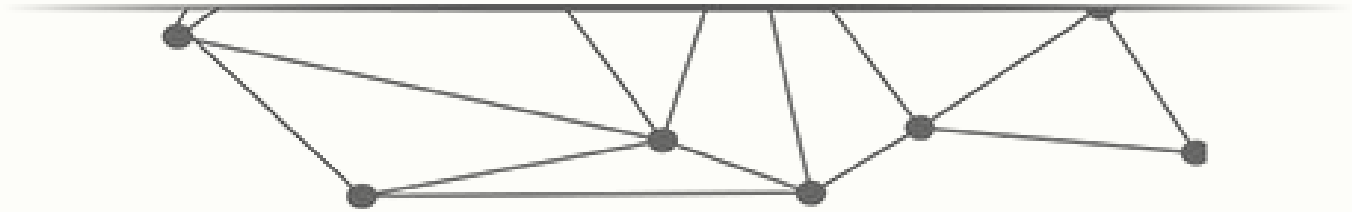
传统机器学习的通病。需要人工从图像中获取目标的特征信息，对于不同的特定检测任务，不同的数据集都需要使用不同的特征，且相互之间差异较大。**系统的鲁棒性和可移植性都较差。**

- **复杂度高**

传统方法多采用滑动窗口对图像进行遍历，把图像分成各种尺度和大小的区域，然后再进行识别筛选。该方法没有针对性，遍历操作的**时间和空间复杂性都很高**。



课堂互动 13.2.3



Part 03

关键技术简介

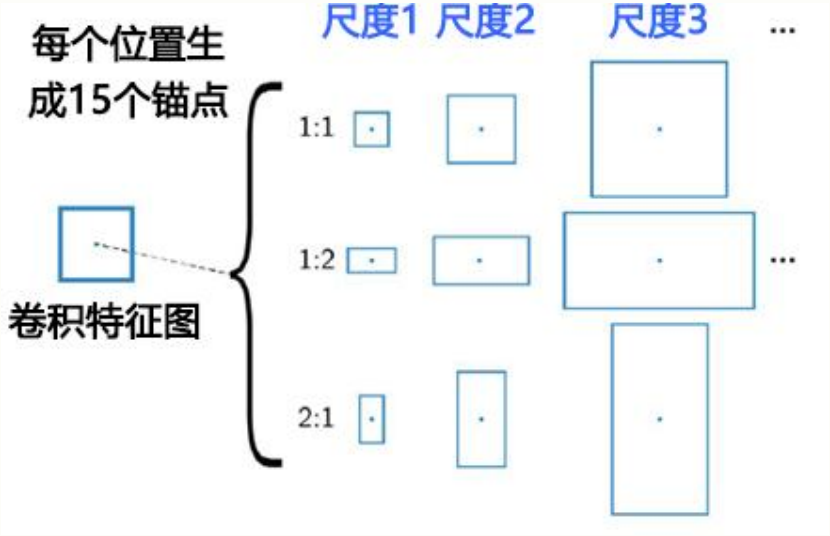
3.1 锚点框（Anchor）

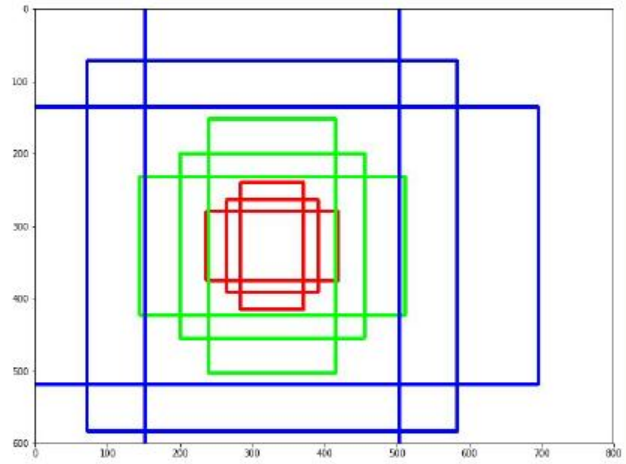
锚点框

Anchor Boxes

锚点框/锚点（Anchor Boxes/Anchor），被选定出来作为基准位置的像素区域。该像素区域可以存在于输入图像，也可以存在于卷积特征图。

实际应用中，特征图（输入图）上的每个点都会作为基准点，并用来生成多个大小（3-5种：32, 64, 128, 256, 512）、比例（1:1, 1:2, 2:1）不同的边界框，这些框就是Anchor。

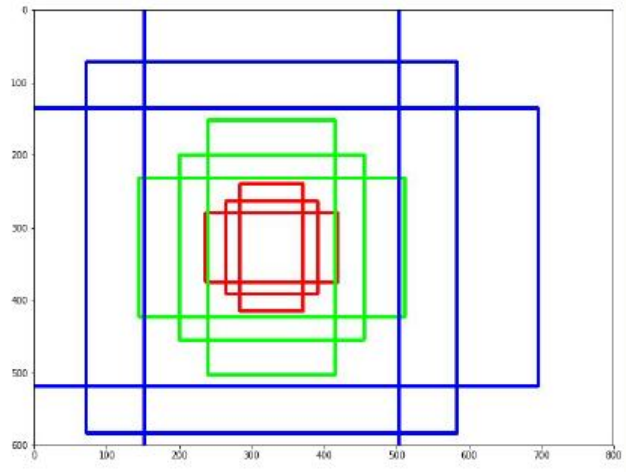


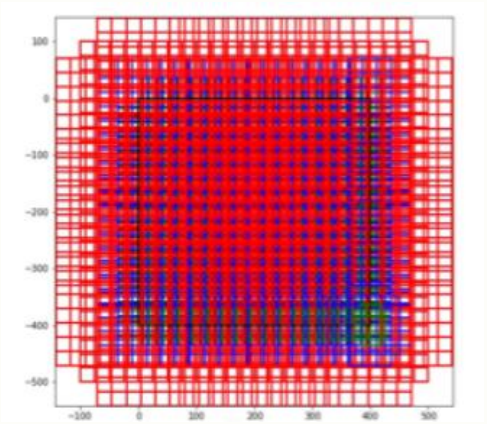


3.1 锚点框（Anchor）

锚点框

Anchor Boxes





图中，红色、蓝色、绿色代表三种Anchor，它们的大小不同。每种Anchor又包括三种不同比例的Anchor。

每个位置共生成9个Anchor。

卷积神经网络对于处理离散的预测要优于连续的回归。

=> **预选择**一些边界框模板，并在此基础上进行回归训练，用于缓解从头回归困难的问题

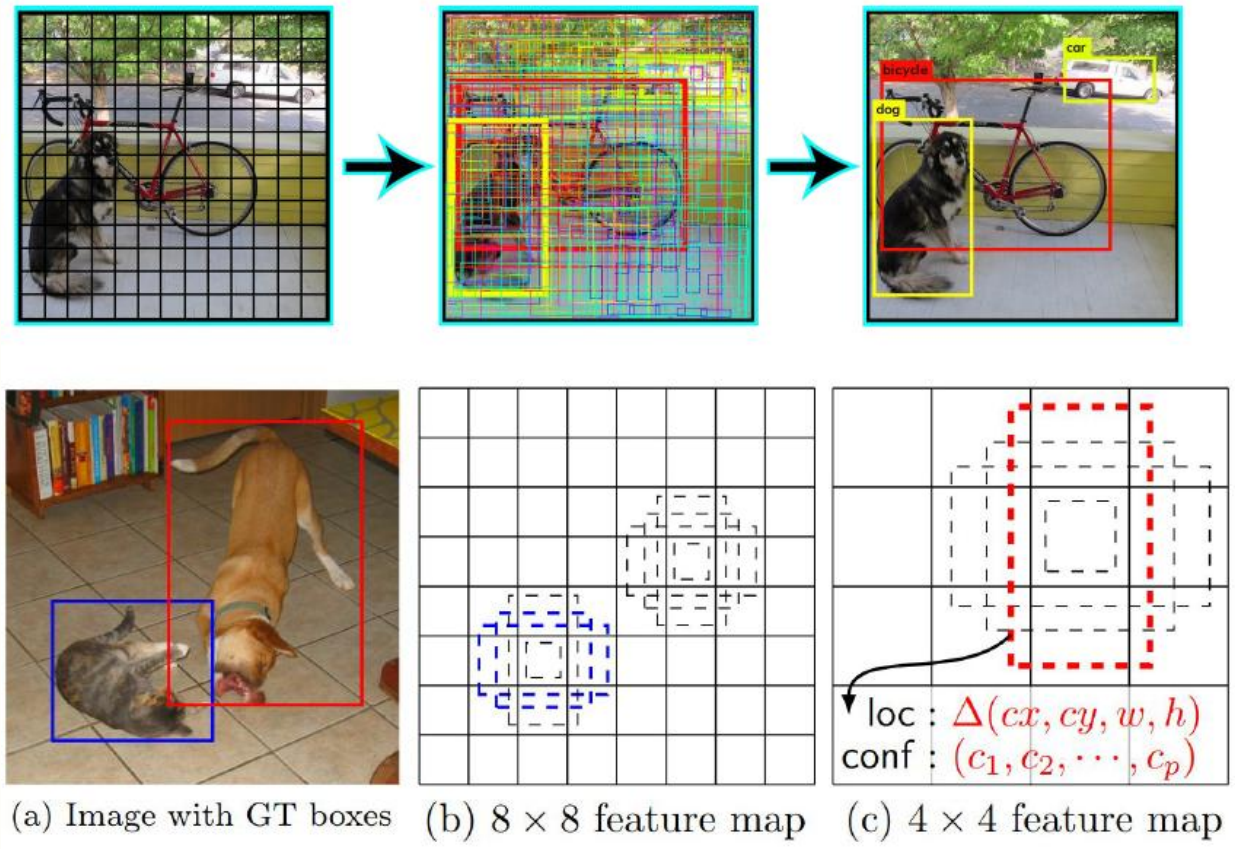
=> 使用神经网络对锚点进行分类和边界框微调

3.1 锚点框（Anchor）

边界框类别

Bounding Box Classification

- **对于每个边界框**
  - 使用二值交叉熵损失预测其置信度，并将置信度最大值作为语义类别(Semantic)进行输出
  - 除区域建议网络(RPN)，每个分类器对于每个类别（包括背景）都包含一个置信度，即预测结果为Yes or No
  - RPN网络输出其对象性(Objectness)置信度，即是否是对象
  - 后期，部分网络使用Softmax损失替代二值交叉熵，直接输出类别



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

3.1 锚点框（Anchor）

边界框微调

Bounding Box Refinement

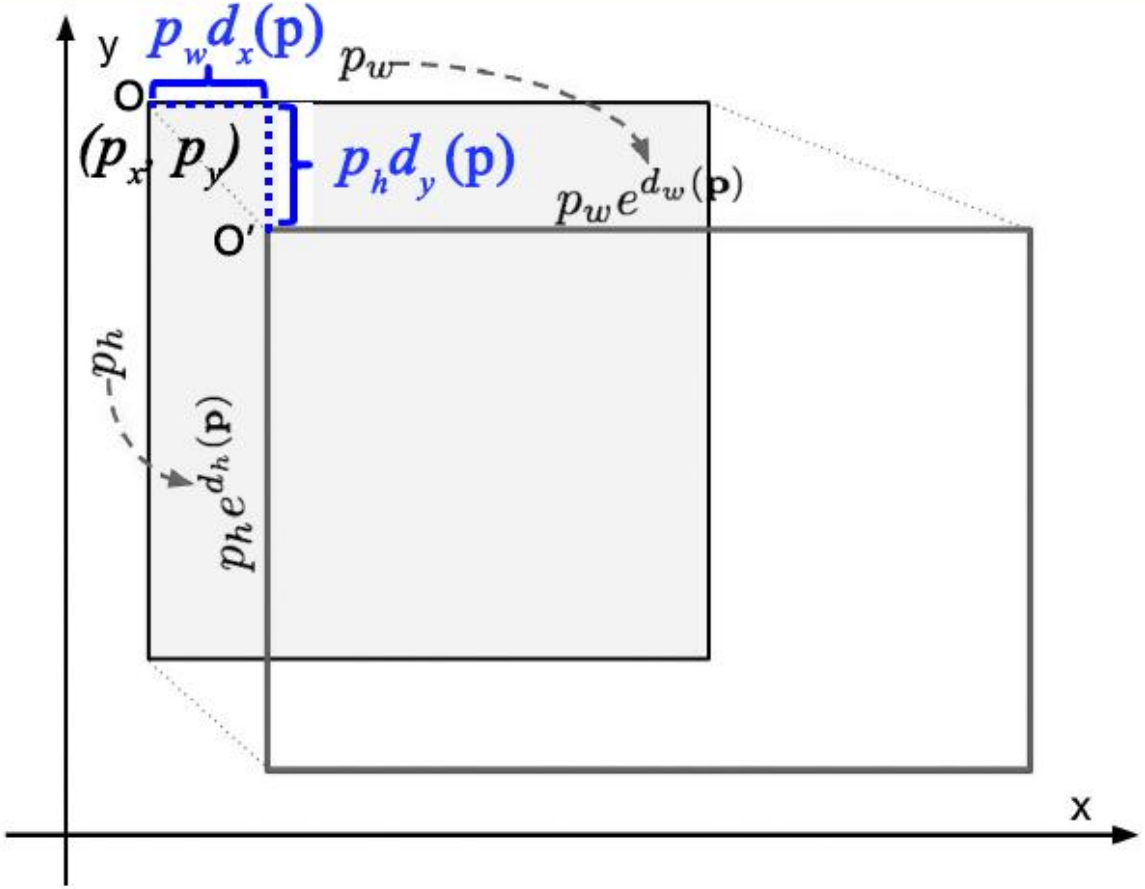
- **给定**
  - 左上角的坐标O(px, py)
  - 锚点框尺度（pw, ph）

- **输出：边界框微调的变化值**
  - 位置偏移

$$d_x = (b_x - p_x)/p_x,\ \ d_y = (b_y - p_y)/p_h$$

  - 对数比例

$$d_w = log(b_w/p_w)\ ,\ \ d_h = log(b_y/p_y)$$



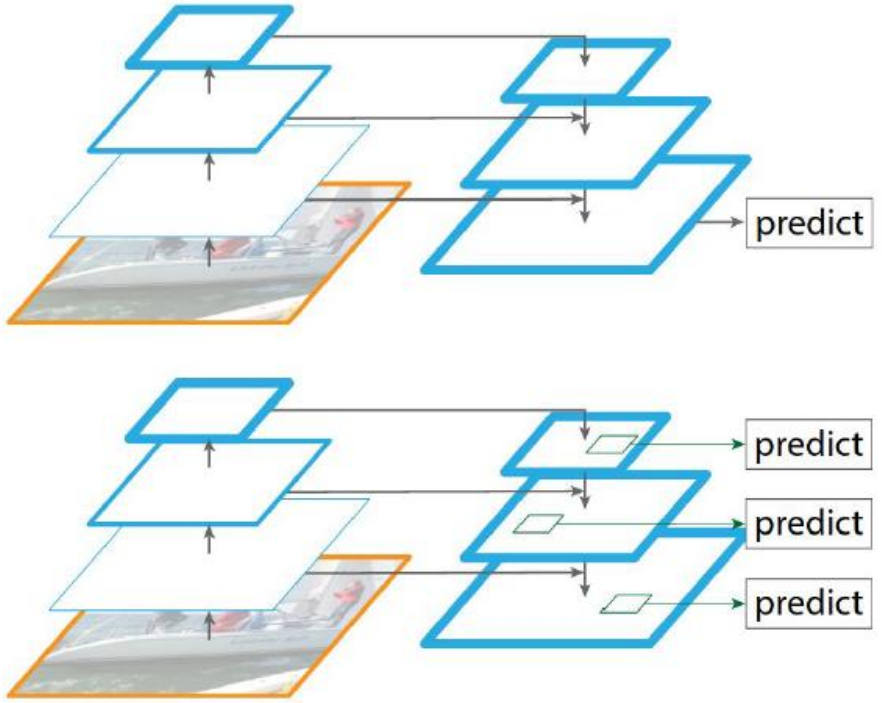
3.2 特征金字塔网络（FPN）

特征金字塔网络

Feature Pyramid Networks

特征金字塔网络（Feature Pyramid Networks, FPN）给标准的卷积神经网络增加了自顶向下的侧向连接，实现多尺度的特征提取。

- 较深的层语义信息丰富，分辨率小，感受野大，适合获取较大目标；较浅的层分辨率大，感受野小，适合获取较小目标。
- FPN可以嵌入到任意骨干网络，且由于是旁路机制，几乎不增加计算负担



上：自上至下结构，输出较好特征的预测；
下：特征金字塔结构，输出不同层次的预测，适合同时进行不同尺度的对象检测

3.3 非极大抑制（NMS）

非极大抑制

Non-maximum suppression

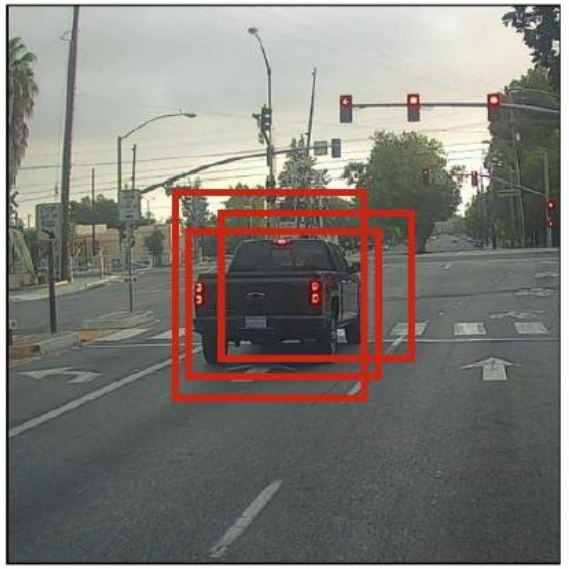
预测结果中可能包含多个对同一实例的预测。

=> 启发式算法，可以移除冗余的预测。

基本方法：

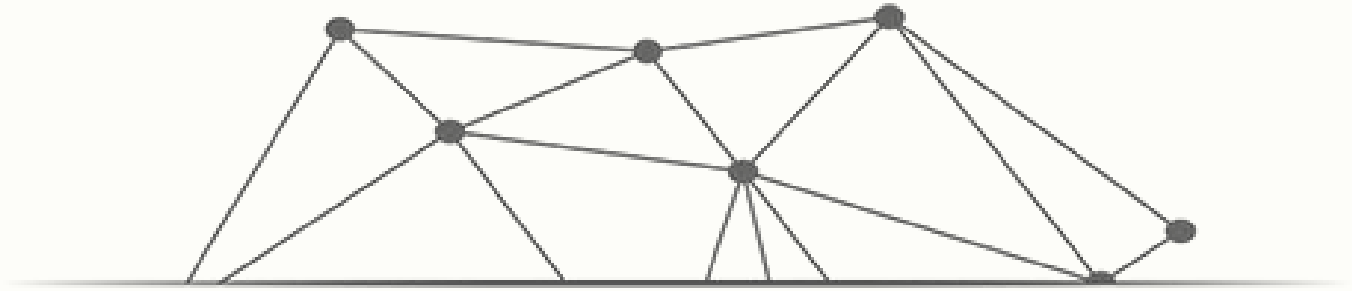
1. 对所有预测结果，按**置信度**进行**降序排序**
2. 将第一个样本**加入确认序列**
3. 若当前预测与**确认序列**严重重叠，则**丢弃**当前预测
4. 否则，将当前预测**加入确认序列**

Before 非极大抑制

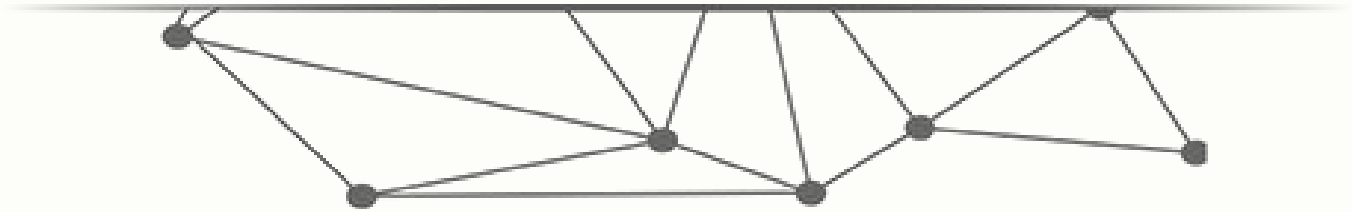


非极大抑制

After 非极大抑制



课堂互动 13.2.4



Part 04

基于两阶段方法的目标检测

/ R-CNN

/ SPP-Net

/ Fast RCNN

/ Faster RCNN

/ RCNN系列算法的进阶优化

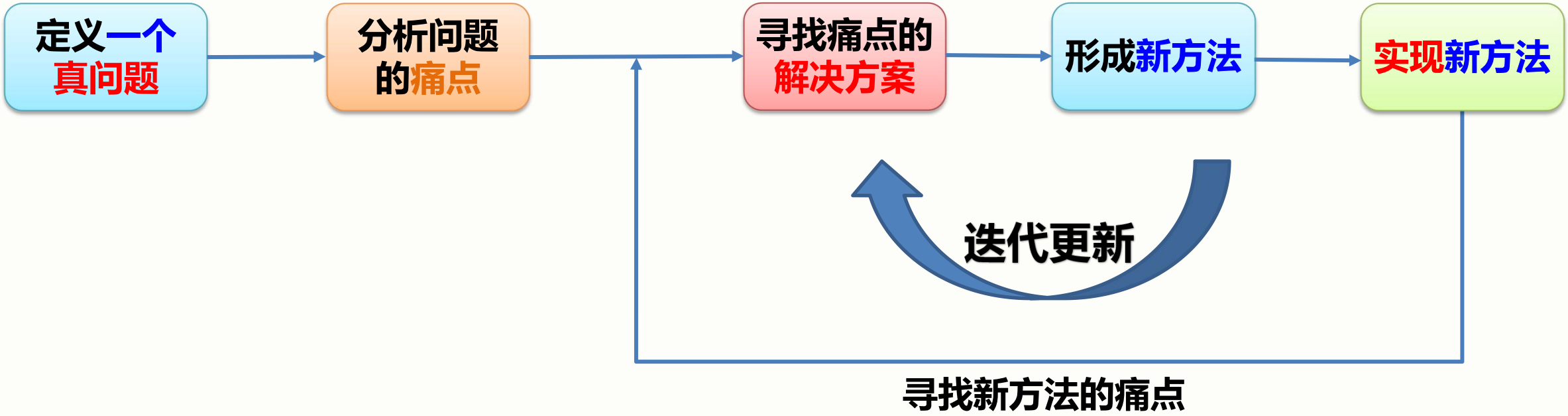
4.0 基于两阶段方法的目标检测

如何培养创造性思维？

如何开始进行创造性的行动？

4.0 基于两阶段方法的目标检测

如何开始进行创造性的行动？



4.0 基于两阶段方法的目标检测（区域选择）

如何开始进行创造性的行动？

Case1: 目标检测

ETC系统
行人检测
特定目标识别
犯罪份子检测
违章检测
行为识别
缺陷检测
森林防火检测
未带口罩识别
…

定义真问题

目标检测

分析痛点

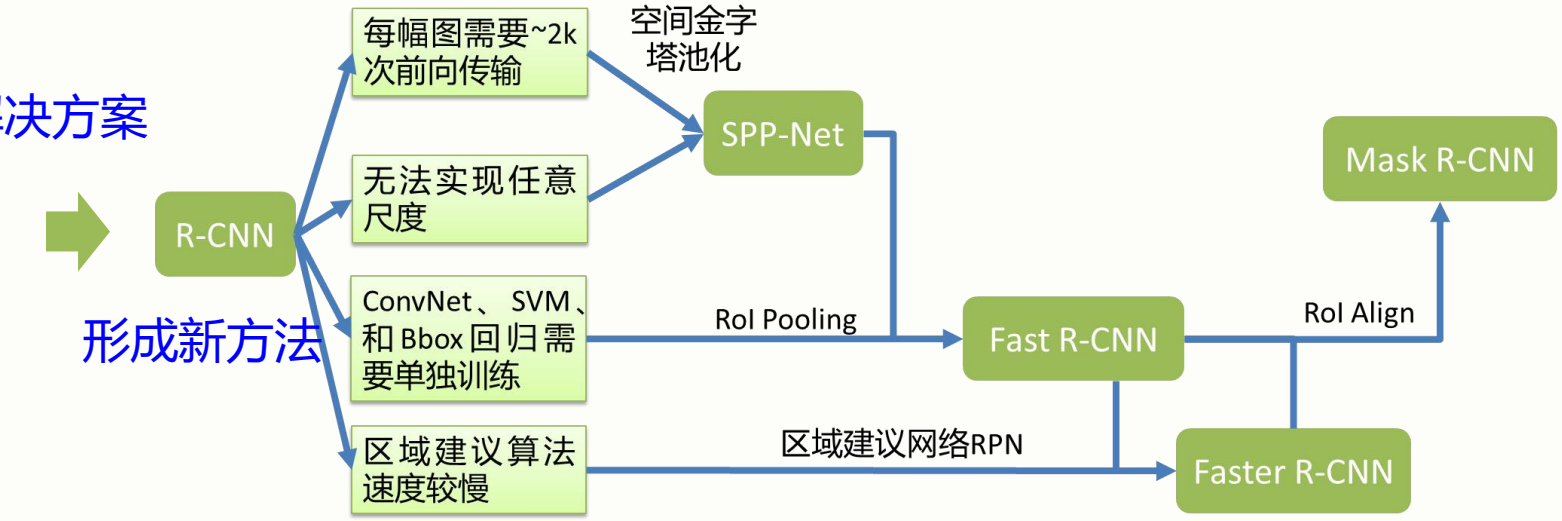
研究传统目标检测的问题

寻找解决方案

基于深度学习的目标检测

形成新方法

实现新方法，并迭代更新



4.0 基于两阶段方法的目标检测（区域选择）

如何开始进行创造性的行动？

1. 赚大钱
2. 找个好工作
3. 专升本、考研
4. 拥有一定的社会地位
5. 拥有一定的学术声誉
6.……

修炼自身 提高等级

定义真问题	分析痛点	寻找解决方案
	社交能力差	加入学生会
	编程能力差	做项目
	看不懂英语	新东方
	不会做数学题	刷题

Case2: 修炼自身

项目一 → 方法一

项目二 → 方法二

项目三 → 方法三

形成新方法

实现新方法，并迭代更新

4.0 基于两阶段方法的目标检测（区域选择）

Ross Girshick

Kaiming He

Shaoqing Ren

4.2 **SPP-Net (2014)**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
Microsoft Research

4.1 **R-CNN (2014)**

Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik
UC Berkeley

4.3 **Fast R-CNN (2014)**

Ross Girshick
Microsoft Research

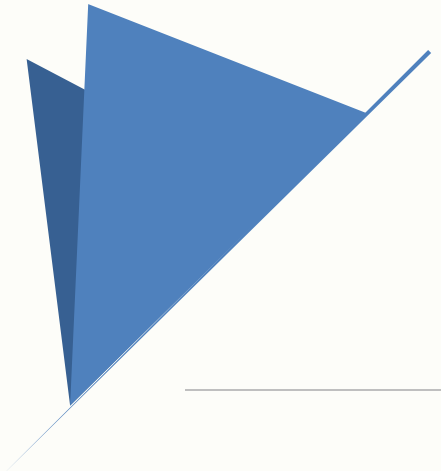
4.4 **Faster R-CNN (2014)**

Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun
Microsoft Research

4.5 RCNN系列算法的进阶优化

Mask R-CNN (2018)

Kaiming He, Georgina Gkioxari, Piotr Dollar, Ross Girshick
Facebook AI Research

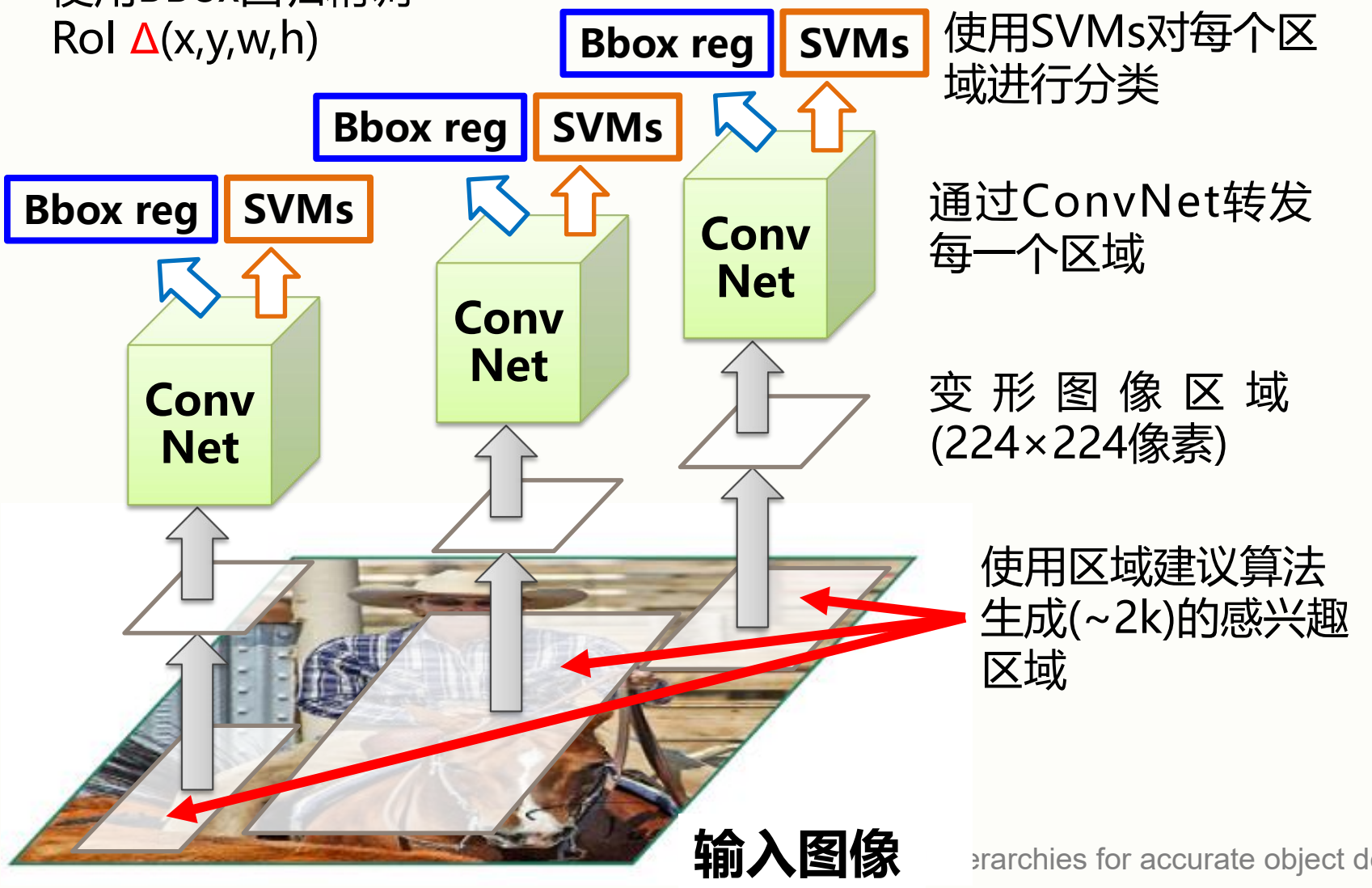


R-CNN

4.1 R-CNN

R-CNN的基本原理框架图

使用BBox回归精调
RoI Δ(x,y,w,h)



使用SVMs对每个区域进行分类

通过ConvNet转发每一个区域

变形图像区域(224×224像素)

使用区域建议算法生成(~2k)的感兴趣区域

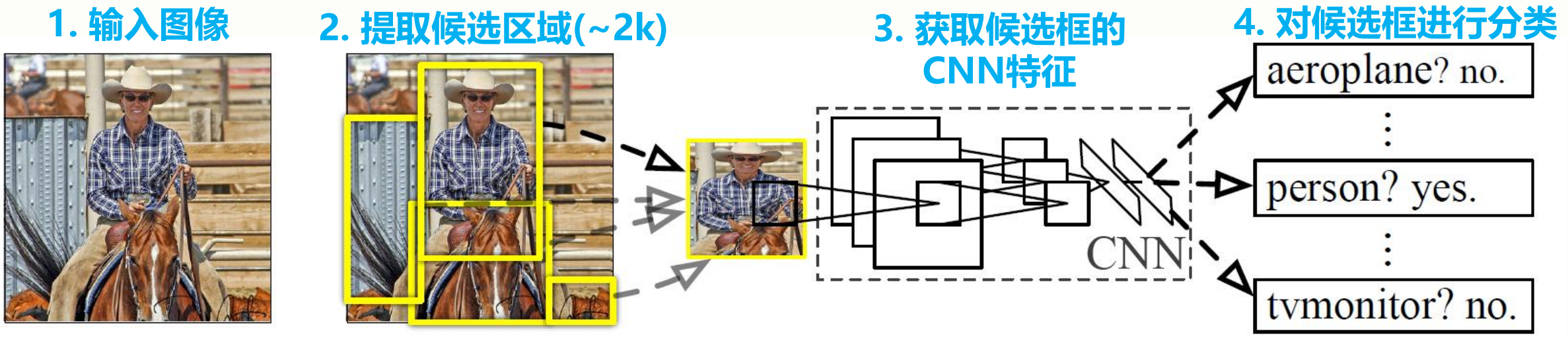
问题：

1. 预测非常慢！每幅图像需要独立地进行~2k次前向传输。每幅图像大约13秒。
2. 所有区域都需要变形为224×224，容易产生失真。
3. ConvNet、SVM和BBox回归需要单独训练，无法统一回传梯度，影响性能。
4. 区域建议算法速度较慢。

erarchies for accurate object detection and semantic segmentation", CVPR 2014.

4.1 R-CNN

R-CNN体系结构图

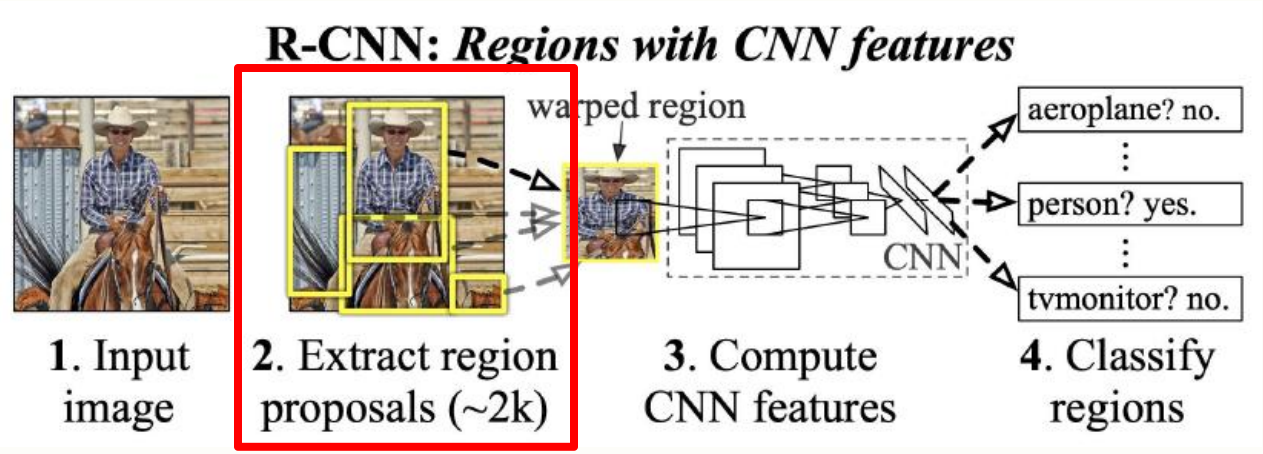


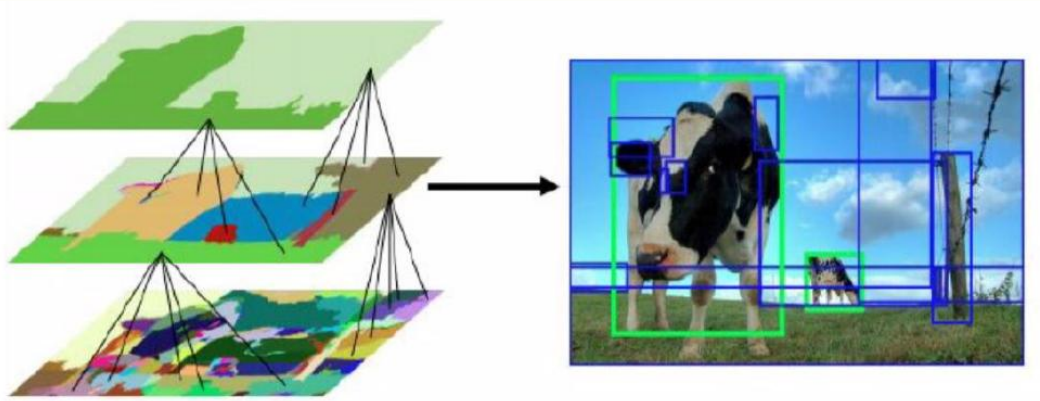
1. **生成候选区域**：输入原始图像，使用Selective Search生成~2k的RoIs；
2. **候选区域缩放**：将RoIs缩放(Warp)到固定大小224×224；
3. **特征提取**：使用卷积神经网络获取每个RoI的4096维特征(FC7)；
4. **候选区域分类**：使用训练好的SVM对这~2k的特征向量进行分类
5. **边界精调**：使用线性回归器(Bbox Reg)来精调边框，每个类别都需要一个线性回归器。
6. **非极大抑制**：对每个类进行NMS去除冗余

4.1 R-CNN

R-CNN的基本过程

1. 提取候选区域



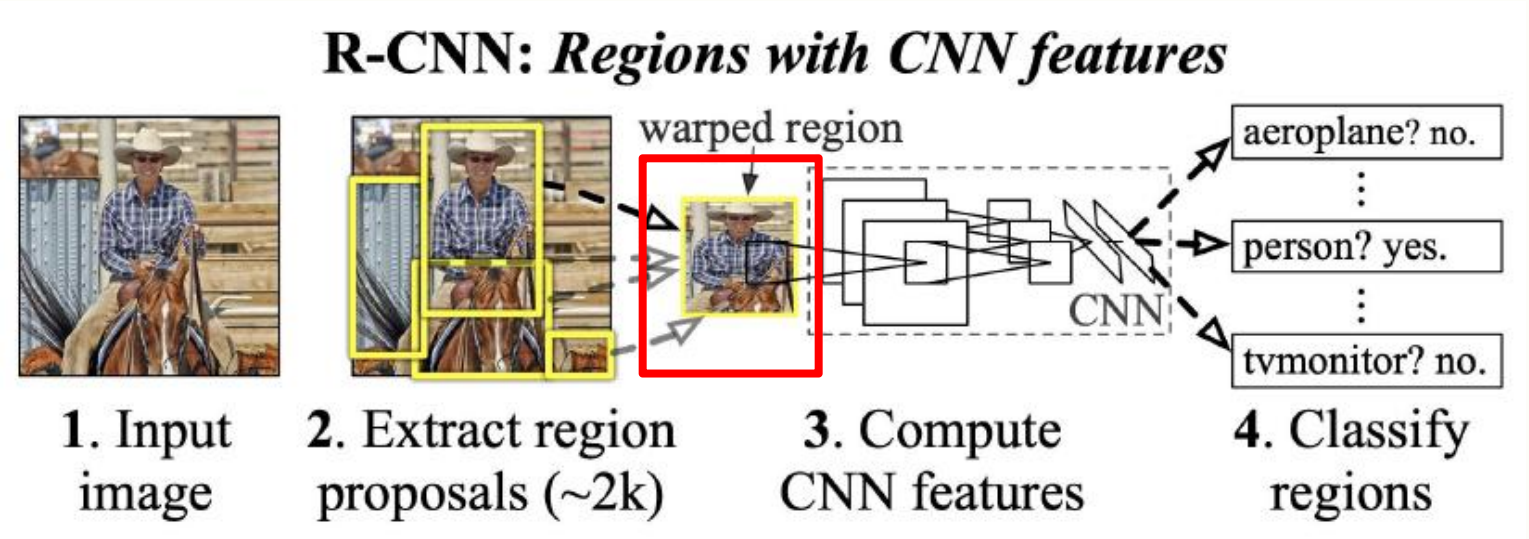


- **选择性搜索(Selective Search, SS)**：根据颜色、纹理、尺寸和空间较低相似度提取大约2000个候选区域 (Region Proposal, RoI)。
- **存在的问题**
  - ✓ 对于每幅图像，提取Region Proposal需要额外的步骤
  - ✓ 存储和重复提取Region Proposal的特征需要花费大量的存储和计算资源

4.1 R-CNN

R-CNN的基本过程

2. 统一尺寸



- **区域拉伸(Warped region)**：

通过Selective Search生成的候选区域尺度不同，与CNN(AlexNet)不兼容，需要统一压缩为相同的尺寸（227×227）。

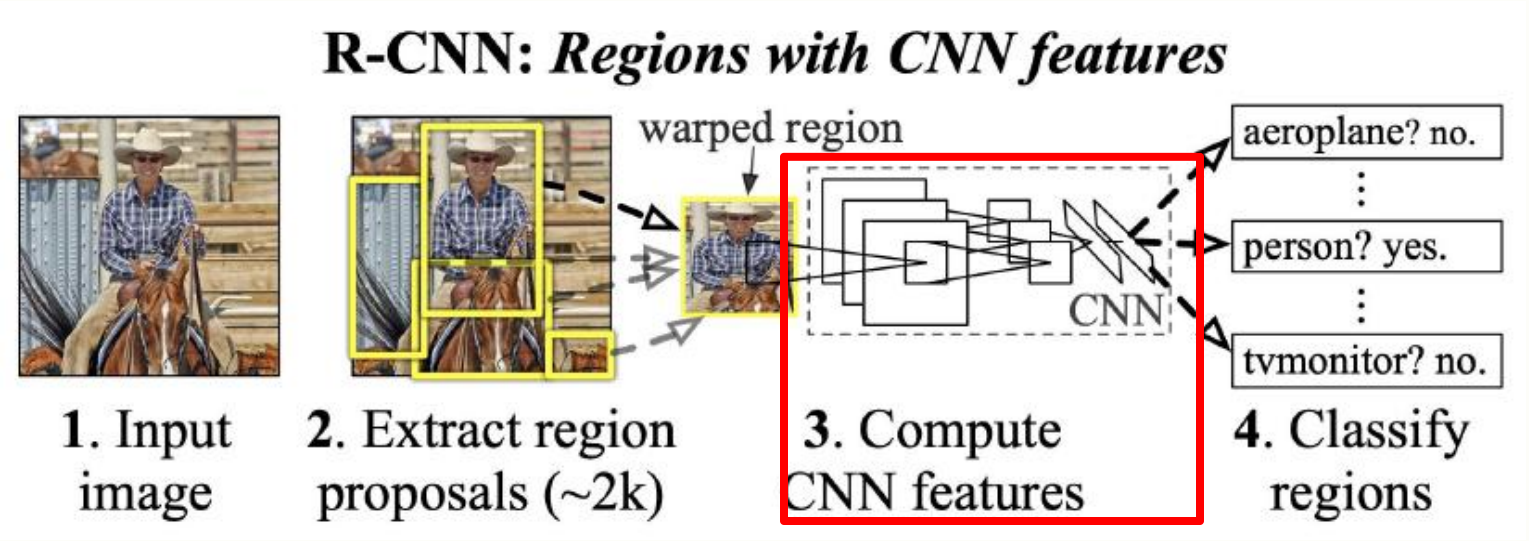
- **存在的问题**

将不同比例的Region Proposal压缩为统一尺寸，将严重影响CNN提取特征的质量。

4.1 R-CNN

R-CNN的基本过程

3. 特征提取



- **特征提取(Feature Extraction)**：

  使用卷积神经网络的最后一个全连接层FC7作为建议区域的特征向量。

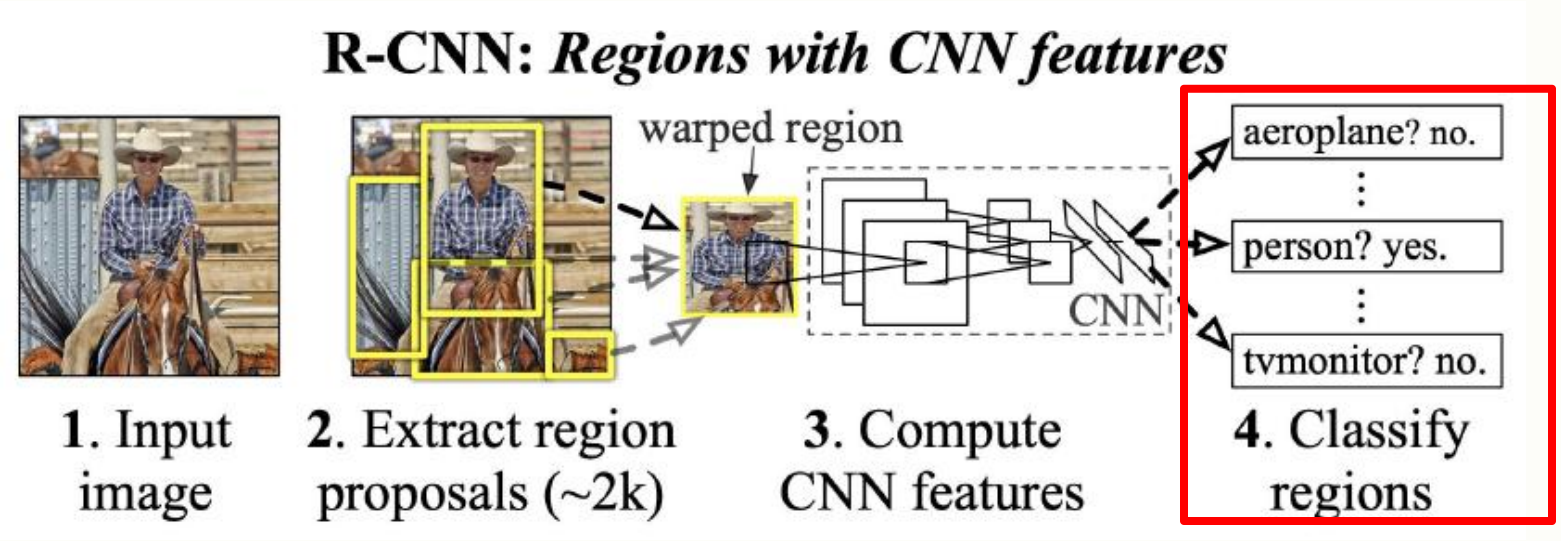
- **存在的问题**

  保存所有的目标区域的特征大约需要200G的存储空间。

4.1 R-CNN

R-CNN的基本过程

4. 区域分类



- **区域分类(Classify Regions)**： 为每个类（包括背景类）训练一个二分类的支持向量机。
- **存在的问题**
  - ✓ SVM需要单独进行训练，训练复杂性较大；
  - ✓ 每个类别需要单独训练，当类别数增加，复杂性大幅上升；且逐个执行SVM分类消耗较多时间

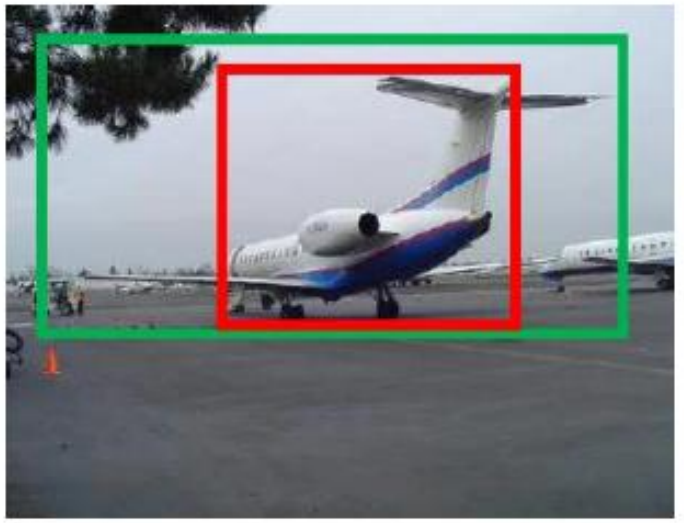
4.1 R-CNN

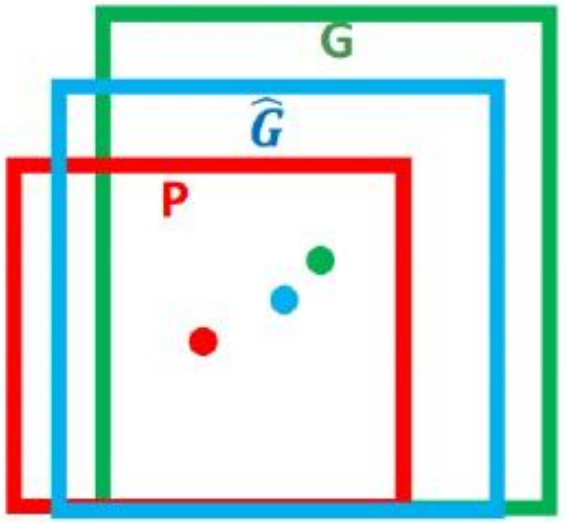
R-CNN的基本过程

5. 边界框回归

通过学习一种**映射关系**，对候选目标区域的位置进行**精化(Refine)**.

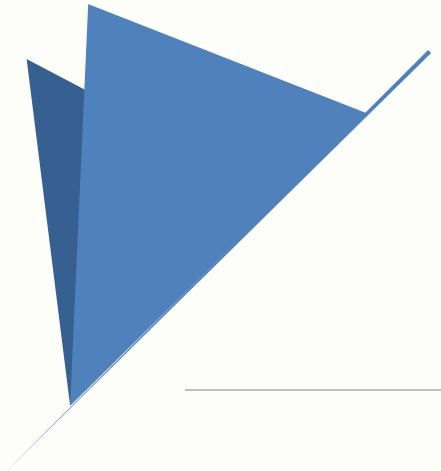
- G：真实坐标
- P：RoI坐标（Selective Search获得）
- $\hat{G}$：修正后的坐标





学习一种映射关系f，可以**最小化**
RoI (p_x, p_x, p_x, p_x)和
GroundTruth (G_x, G_y, G_w, G_h)
之间的差异。

$$f(p_x, p_x, p_x, p_x) = (\hat{G}_x, \hat{G}_y, \hat{G}_w, \hat{G}_h)$$
$$\approx (G_x, G_y, G_w, G_h)$$

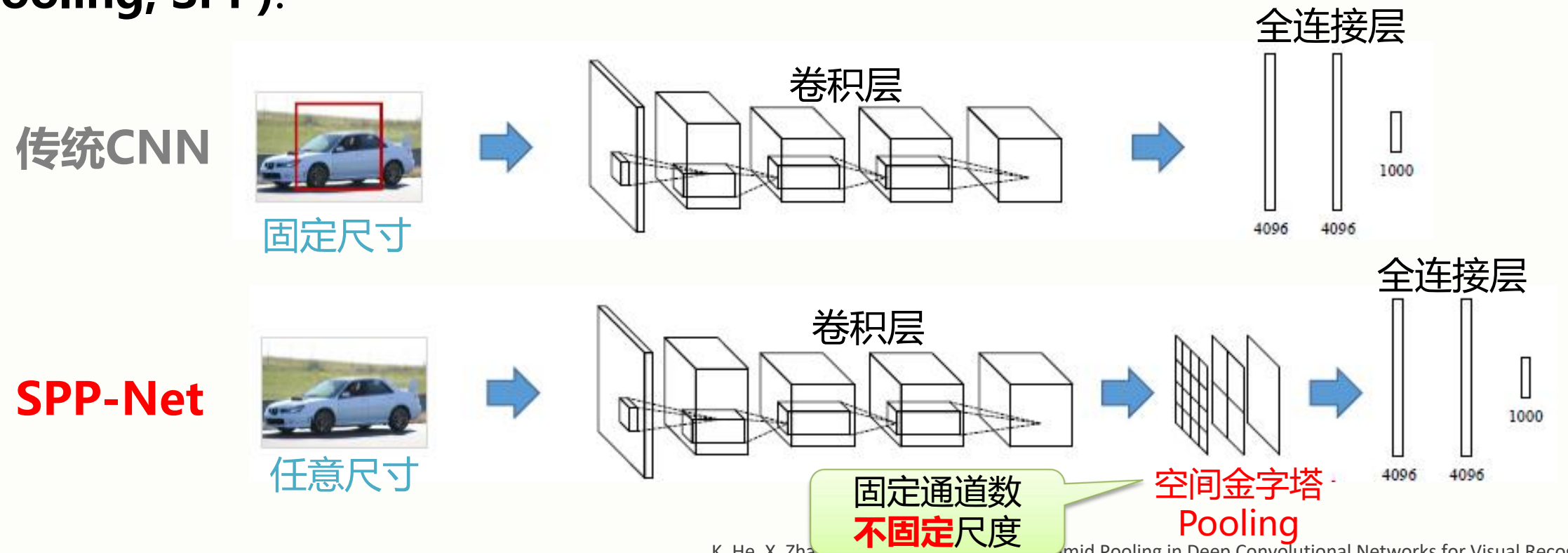


SPP-Net

4.2 SPP-Net

SPP-Net 体系结构图

针对R-CNN的两个问题：1). 每幅图像都需要进行~2k次的前向传输；2). 所有区域都需要裁剪为固定尺寸的问题。何凯明等人提出了**空间金字塔池化(Spatial Pyramid Pooling, SPP)**.

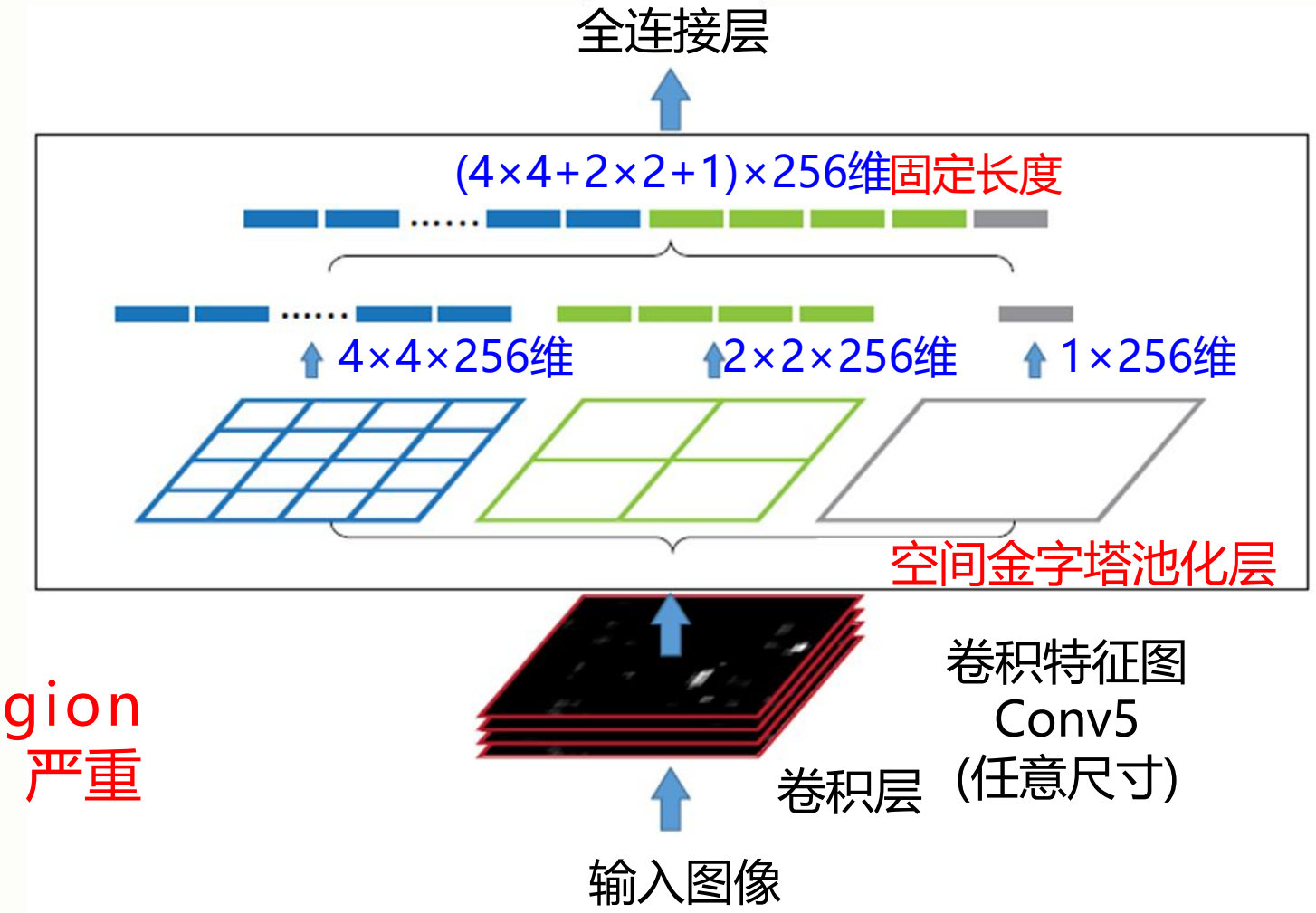


K. He, X. Zhang, S. Ren, J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **ECCV 2014**

4.2 SPP-Net

空间金字塔的基本原理

- **变量输入尺度/规模**
  - 多尺度训练
  - 多尺度测试
  - 全图像视野
- **多级池化**
  - 对形变比较鲁棒



缺点：使用Warp将每个Region Proposal统一成相同的尺寸，严重影响了CNN特征提取的质量。

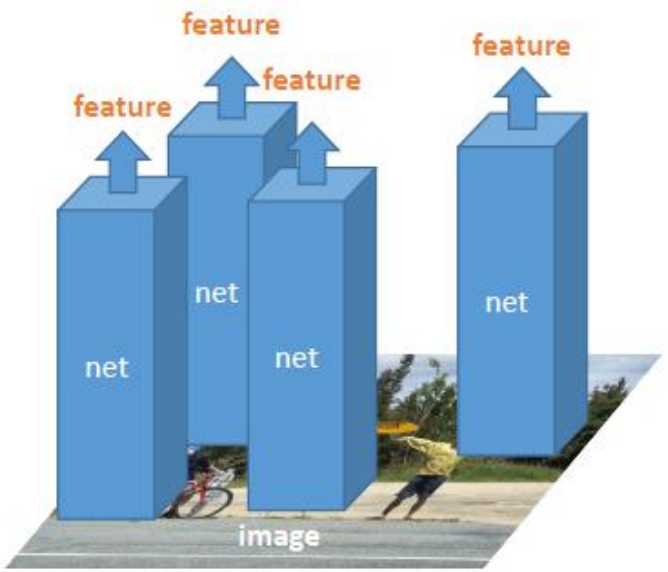
Why？？？

K. He, X. Zhang, S. Ren, J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **ECCV 2014**

4.2 SPP-Net

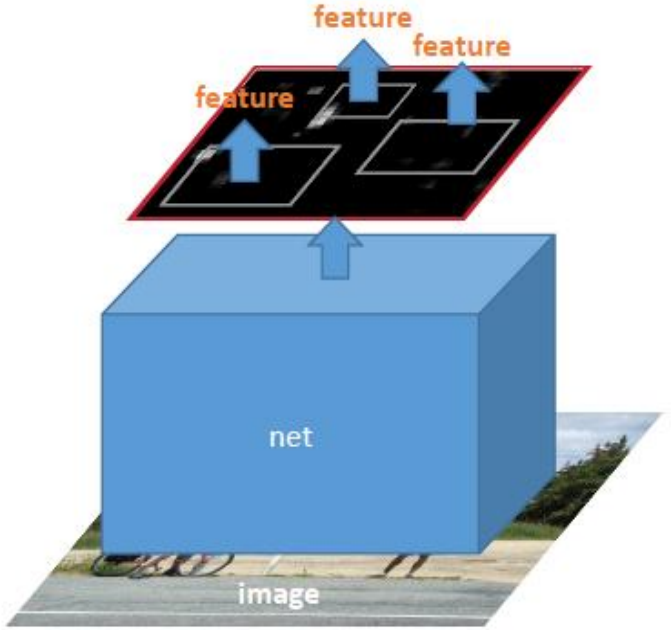
RCNN vs. SPP

图像区域 *vs.* 特征图区域



R-CNN

每幅图像大约包含~2000个网络



SPP-net

整幅图像只需要一个网络

R-CNN的缺点：为每个Region Proposal提取特征花费大量的计算时间和存储空间。

SPP-Net的改进：一次性提取整个图像的特征，然后在特征图上提取不同的RoI区域

减少了提取特征的时间
节省了用于存储特征的空间

K. He, X. Zhang, S. Ren, J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **ECCV 2014**

4.2 SPP-Net

SPP-Net的优缺点分析

问题：

1. 预测非常慢！每幅图像需要独立地进行~2k次前向传输。每幅图像大约13秒。

   利用**空间金字塔技术池化(SPP)**实现了特征提取只需要一次前向传输的改进，极大地加快了目标检测的速度。

2. 所有区域都需要变形为224×224，容易产生失真。

   空间金字塔池化也解决了任意尺寸输入的问题。

3. ConvNet、SVM和BBox回归需要单独训练，无法统一回传梯度，影响性能。

   三个网络仍然需要独立训练，且任然需要大量的磁盘保存中间特征。

4. 区域建议算法速度较慢。

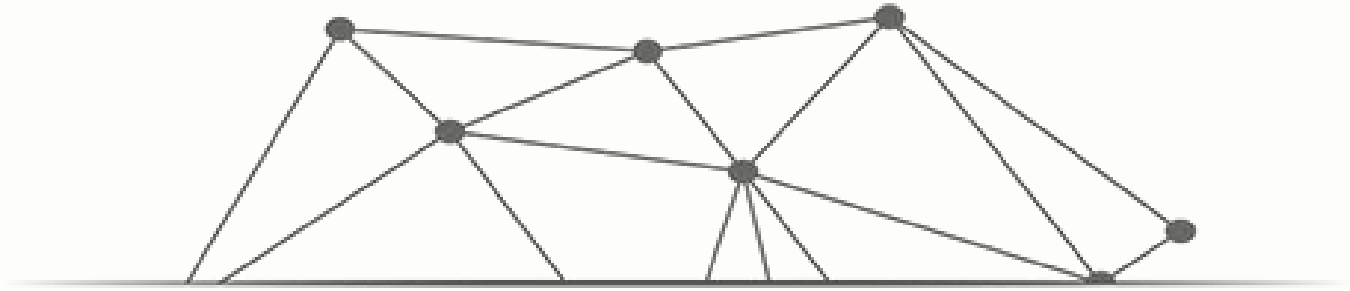
K. He, X. Zhang, S. Ren, J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **ECCV 2014**

4.2 SPP-Net

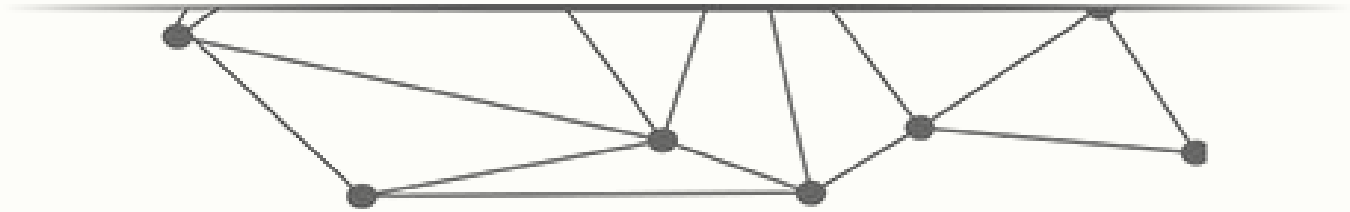
从R-CNN到SPP-Net预测过程的变化

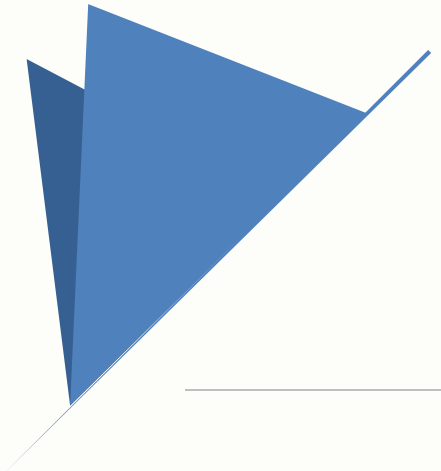
1. **生成候选区域**：输入原始图像，使用SS生成~2k的RoIs；
2. **候选区域缩放**：将RoIs缩放到固定大小224×224；
3. **特征提取**：使用卷积神经网络获取每个RoI的4096维特征；
4. **候选区域分类**：使用训练好的SVM对这~2k的特征向量进行分类
5. **边界精调**：使用线性回归器(Bbox Reg)来精调边框，每个类别都需要一个线性回归器。
6. **非极大抑制**：对每个类进行NMS去除冗余

1. **生成候选区域**：输入原始图像，使用SS生成~2k的RoIs；
2. **特征提取**：使用卷积神经网络提取整个图像的特征图
3. **空间池化采样**：将候选区域映射到特征图上，对每个RoI进行SPP操作，得到固定长度的特征并输出到全连接层，生成4096维特征
4. **候选区域分类**：使用训练好的SVM对这~2k的特征向量进行分类
5. **边界精调**：使用线性回归器(Bbox Reg)来精调边框，每个类别都需要一个线性回归器。
6. **非极大抑制**：对每个类进行NMS去除冗余



课堂互动 13.2.5





Fast R-CNN

4.3 Fast R-CNN

“Fast” R-CNN的基本原理框架图

使用Bounding Box 精调RoI

Bbox regressor

Softmax

对象分类器

CNN

全连接层

“RoI Pooling”层

“conv5”特征

从区域建议算法中获取感兴趣区域(RoIs)

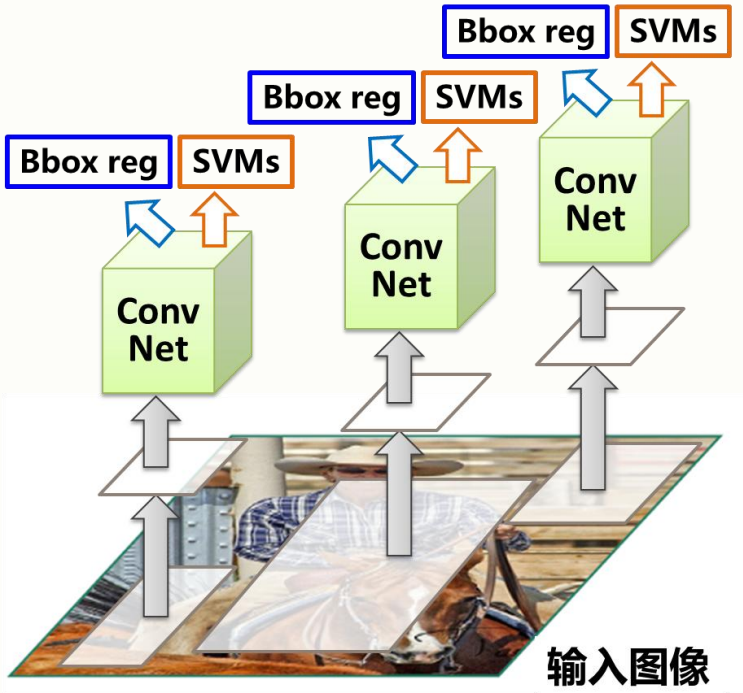
在ConvNet中运行整幅图片

主干网络：
AlexNet, VGG, ResNet, etc

ConvNet

输入图像

“Slow” R-CNN

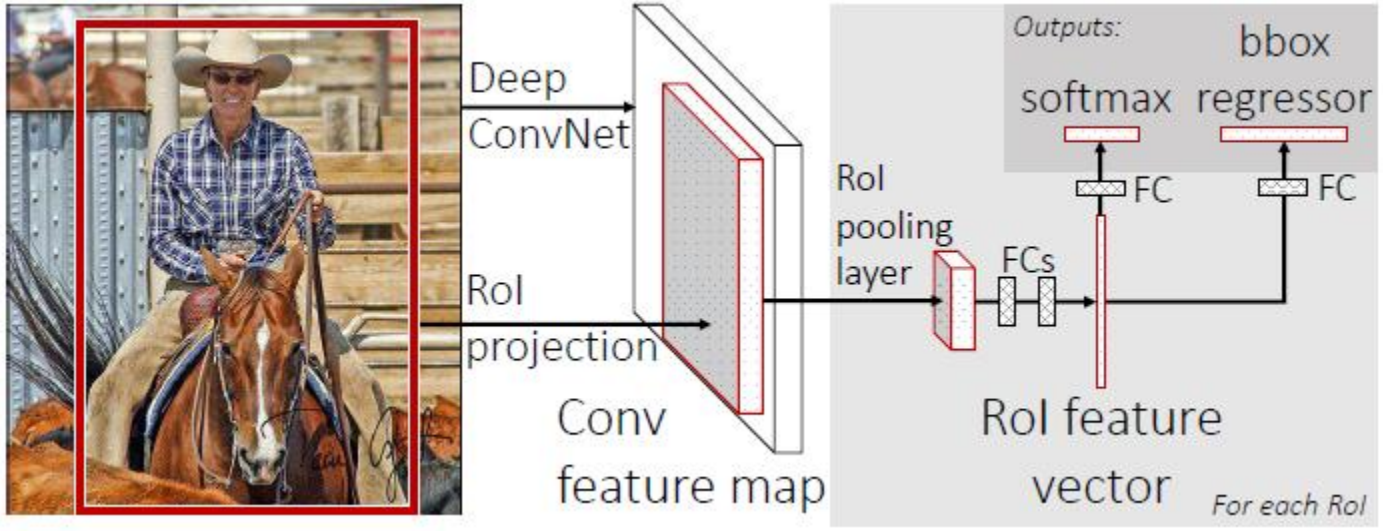


Girshick, “Fast R-CNN”, ICCV 2015

4.3 Fast R-CNN

Fast R-CNN 体系结构图

将分类损失和回归损失统一到同一个框架中



图像Warp → 特征Warp

SVM regressor → Softmax regressor

1. **生成候选区域**：输入原始图像，使用Selective Search生成~2k的RoIs；
2. **特征提取**：使用卷积神经网络提取整个图像的特征图
3. **RoI Pooling**：将候选区域映射到特征图，并经过RoI Pooling得到固定长度的特征，输出到全连接层
4. **分类及边界精调**：经过**2个全连接层**后，特征被分别送入Softmax分类器(21类)和Bbox回归器(84类)
5. **非极大抑制**：对每个类进行NMS去除冗余

4.3 Fast R-CNN

从SPP-Net到Fast R-CNN预测过程的变化

1. **生成候选区域**：输入原始图像，使用Selective Search生成~2k的RoIs；
2. **特征提取**：使用卷积神经网络提取整个图像的特征图
3. **空间池化采样** 将候选区域映射到特征图上，对每个RoI进行SPP操作，得到固定长度的特征并输出到全连接层，生成4096维特征
4. **候选区域分类**：使用训练好的SVM对这~2k的特征向量进行分类
5. **边界精调**：使用线性回归器(Bbox Reg)来精调边框，每个类别都需要一个线性回归器。
6. **非极大抑制**：对每个类进行NMS去除冗余

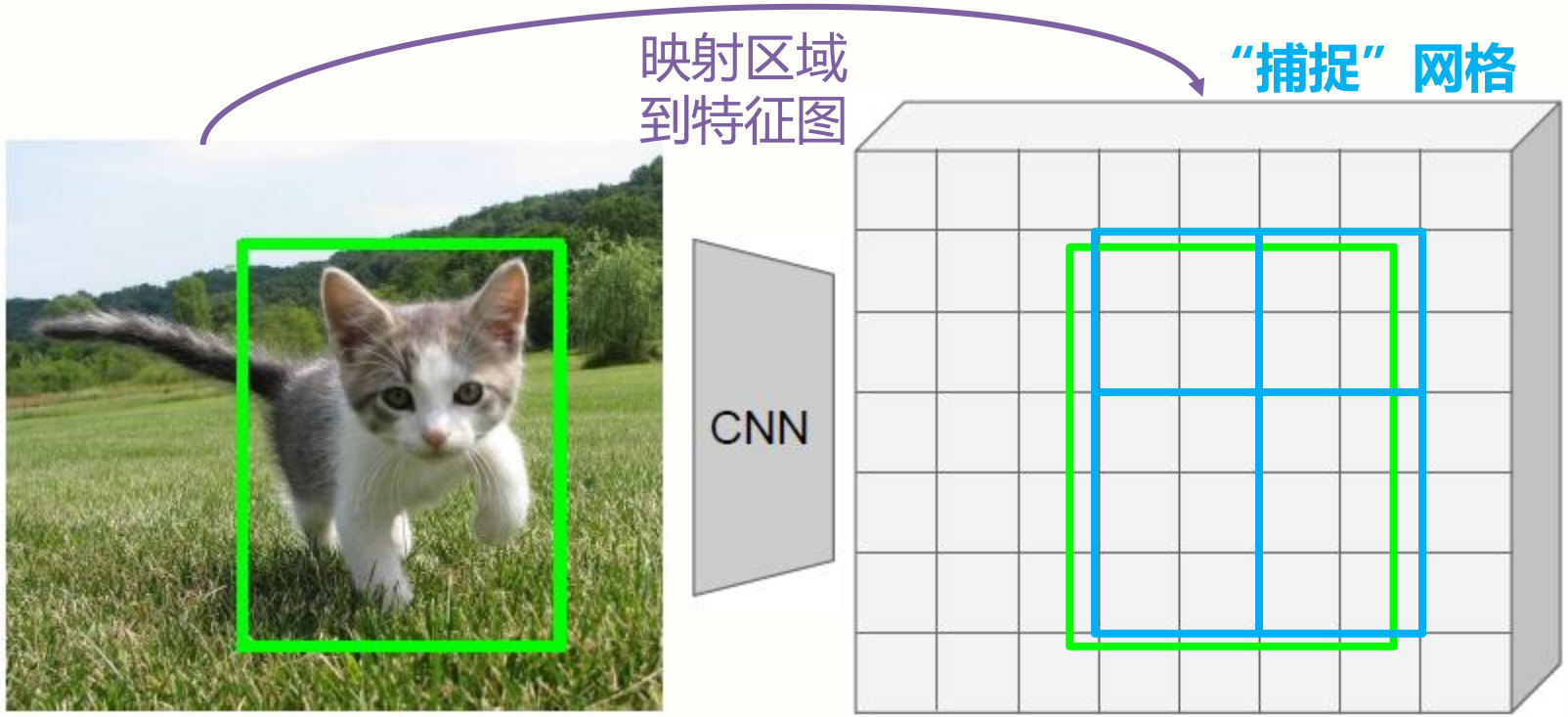
相似但不同

1. **生成候选区域**：输入原始图像，使用Selective Search生成~2k的RoIs；
2. **特征提取**：使用卷积神经网络提取整个图像的特征图
3. **RoI Pooling**：将候选区域映射到特征图上，并经过RoI Pooling得到固定长度的特征，输出到全连接层
4. **分类及边界精调**：经过**2+1个全连接层**后，特征被分别送入Softmax分类器和Bbox回归器
5. **非极大抑制**：对每个类进行NMS去除冗余

Girshick, “Fast R-CNN”, ICCV 2015

4.3 Fast R-CNN

特征裁剪：RoI Pooling



Q：如何将512×20×15的尺度缩放到固定长度(512×2×2)?

Answer：

1. 捕捉网格，将RoI对齐到卷积特征图的网格
2. 将网格划分为（大致）相等的2×2的子区域，并确保每个区域具有相同的激活值。

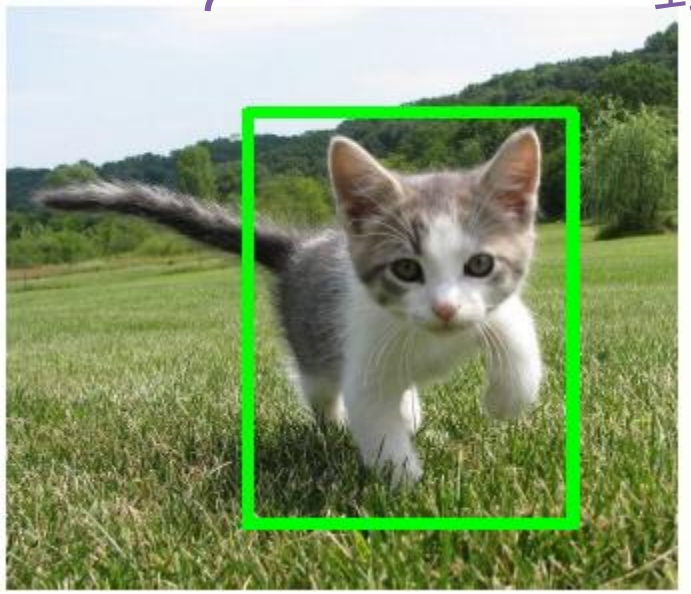
How to ensure?

核心思想：候选区域共享特征图特征，并保持大小一致。

Girshick, “Fast R-CNN”, ICCV 2015

4.3 Fast R-CNN

特征裁剪：RoI Pooling

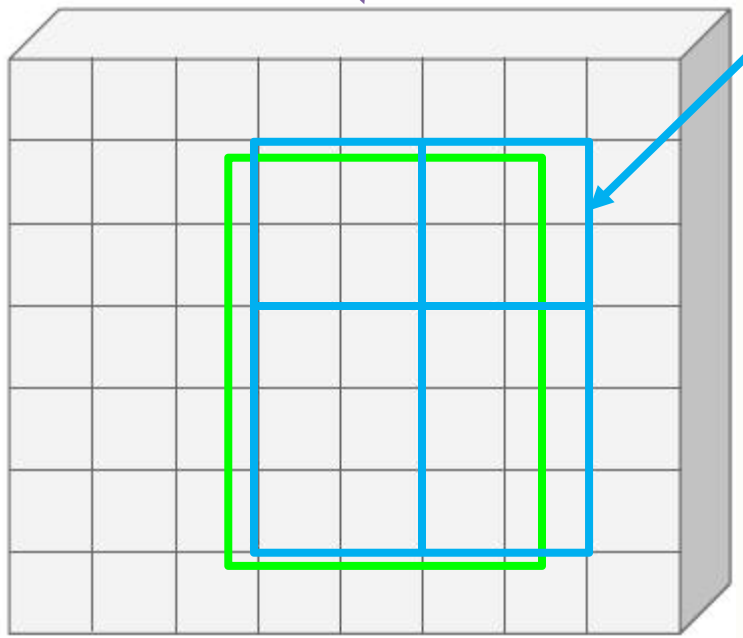


输入图像
(e.g. 3x640×480)

映射区域
到特征图

CNN

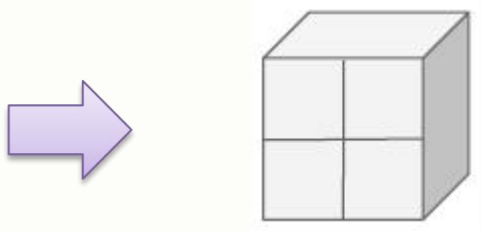
“捕捉”网格



图像特征
(e.g. 512x20×15)

划分为（大致）相等的
子区域2×2。

每个子区域都进行Max-Pool，
其输出作为该区域的输出。



区域特征
(here 512×2×2;
In practice e.g. 512×7×7)

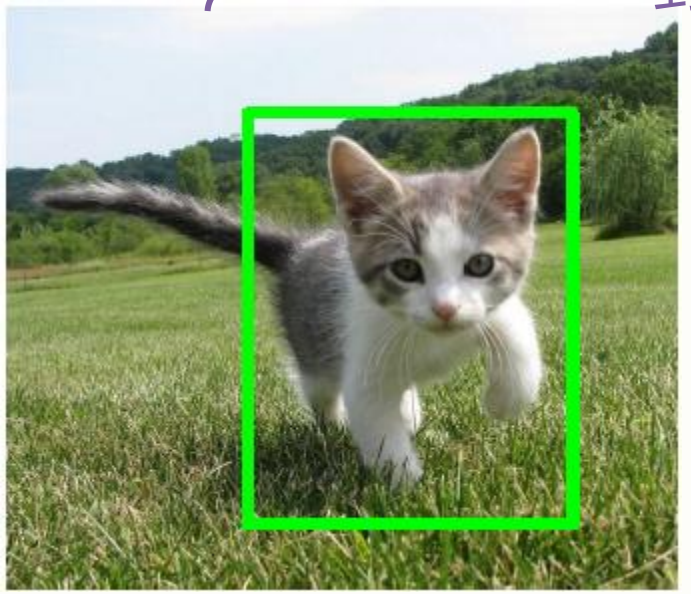
区域特征具有相同的尺度，
即使输入区域尺度不同

问题：区域特征会存在一定的错位误差。

Girshick, “Fast R-CNN”, ICCV 2015

4.3 Fast R-CNN

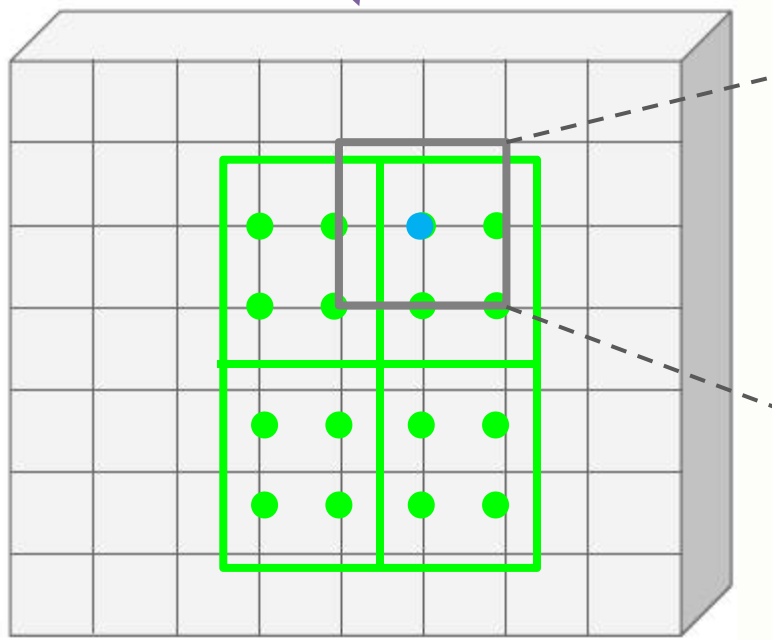
特征裁剪：RoI Align对齐 来自Mask R-CNN的lightspot



输入图像
(e.g. 3x640×480)

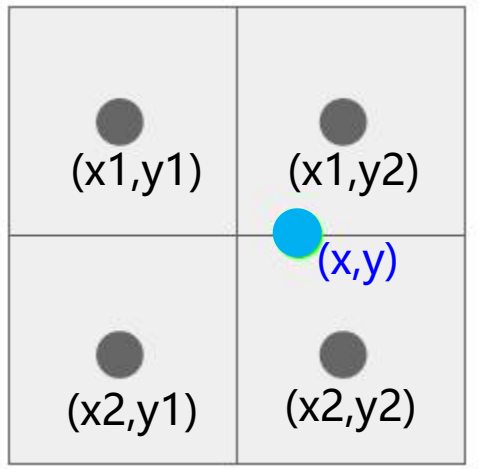
映射区域
到特征图

No“捕捉”！



图像特征
(e.g. 512x20×15)

1. 在每个子区域中均匀地取N个点，
2. 使用**双线性插值**获取每个点的值。



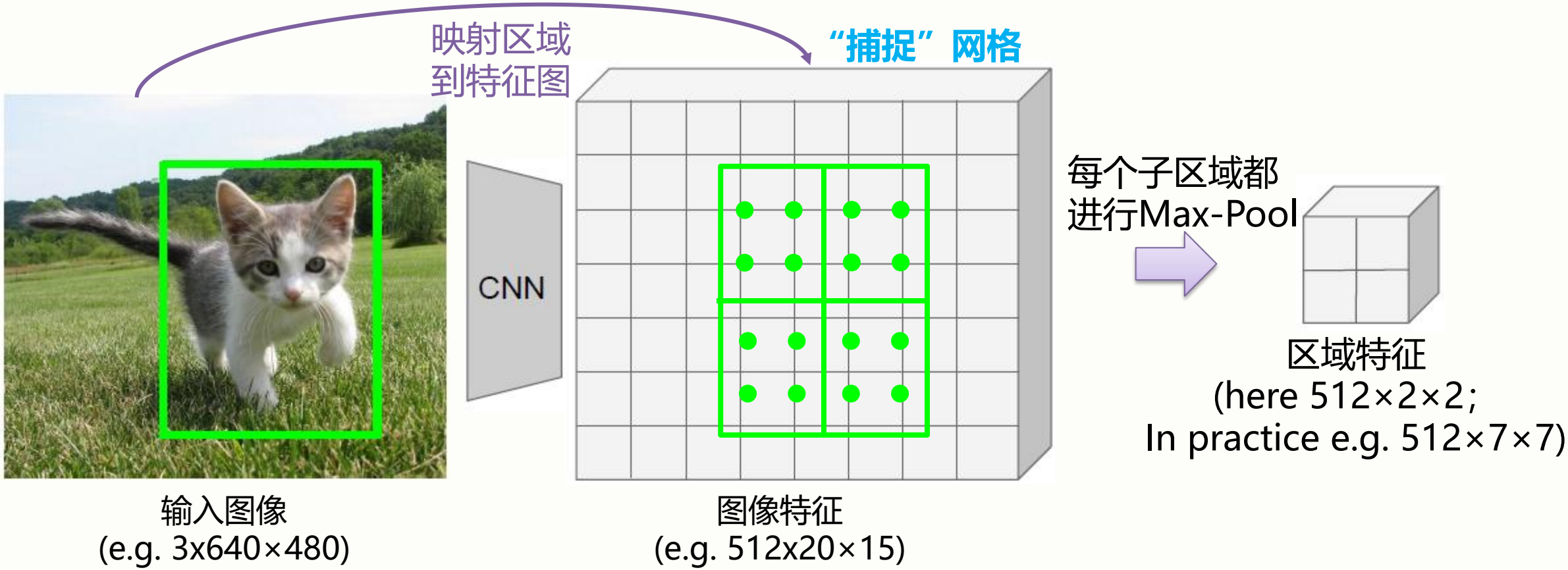
点(x,y)的特征f_{xy}由其四个近邻区域的特征值通过双线性组合获得。

$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0,1-|x-x_i|) \max(0,1-|y-y_j|)$$

He et al, “Mask R-CNN”, ICCV 2017

4.3 Fast R-CNN

特征裁剪：RoI Align对齐 来自Mask R-CNN的lightspot

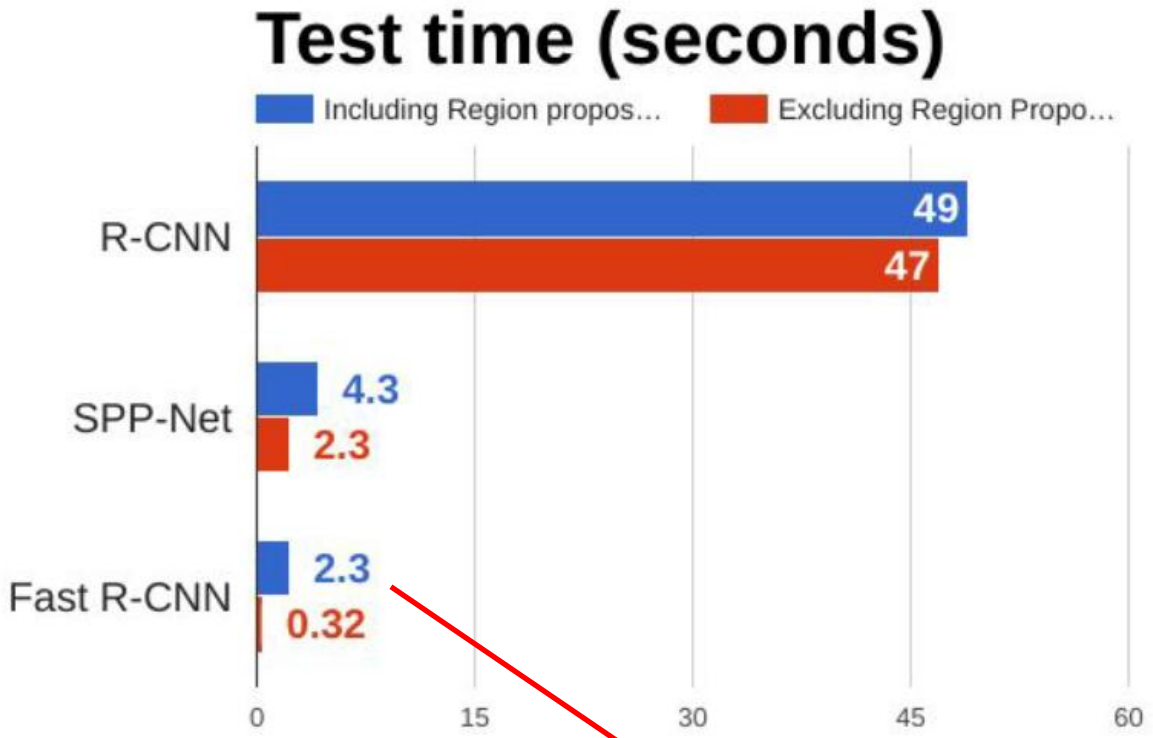


错位误差在RoI Align机制下被消除！

He et al, “Mask R-CNN”, ICCV 2017

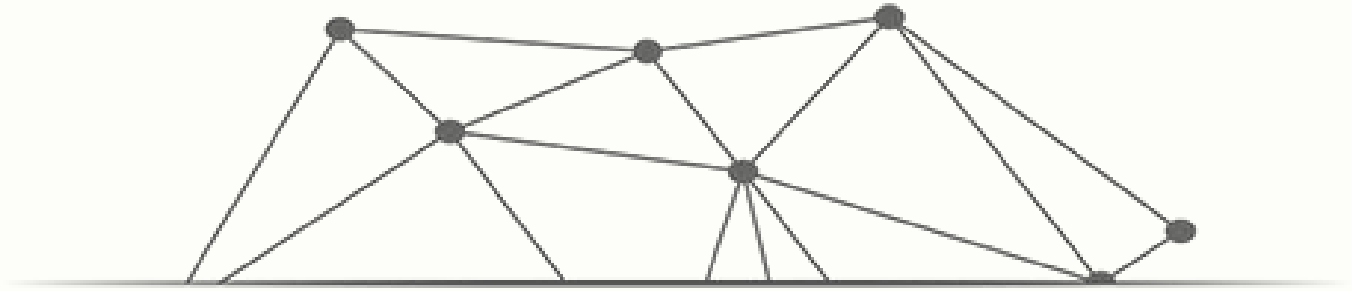
4.3 Fast R-CNN

R-CNN v.s SPP-Net v.s Fast R-CNN

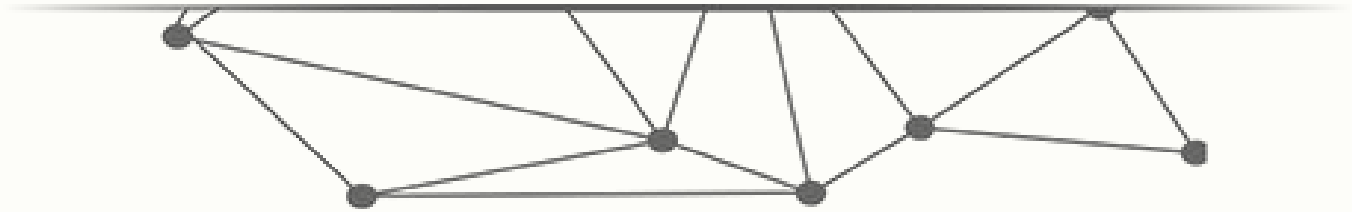


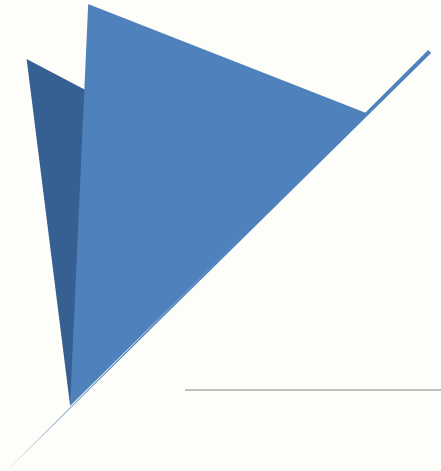
问题4：运行时间由区域建议主导

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015
He et al, “Mask R-CNN”, ICCV 2017



课堂互动 13.2.6





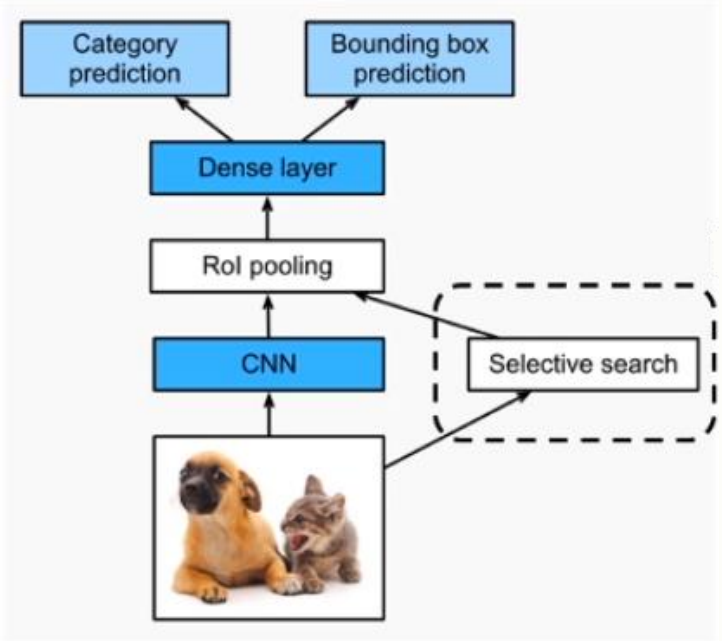
Faster R-CNN

4.4 Faster R-CNN

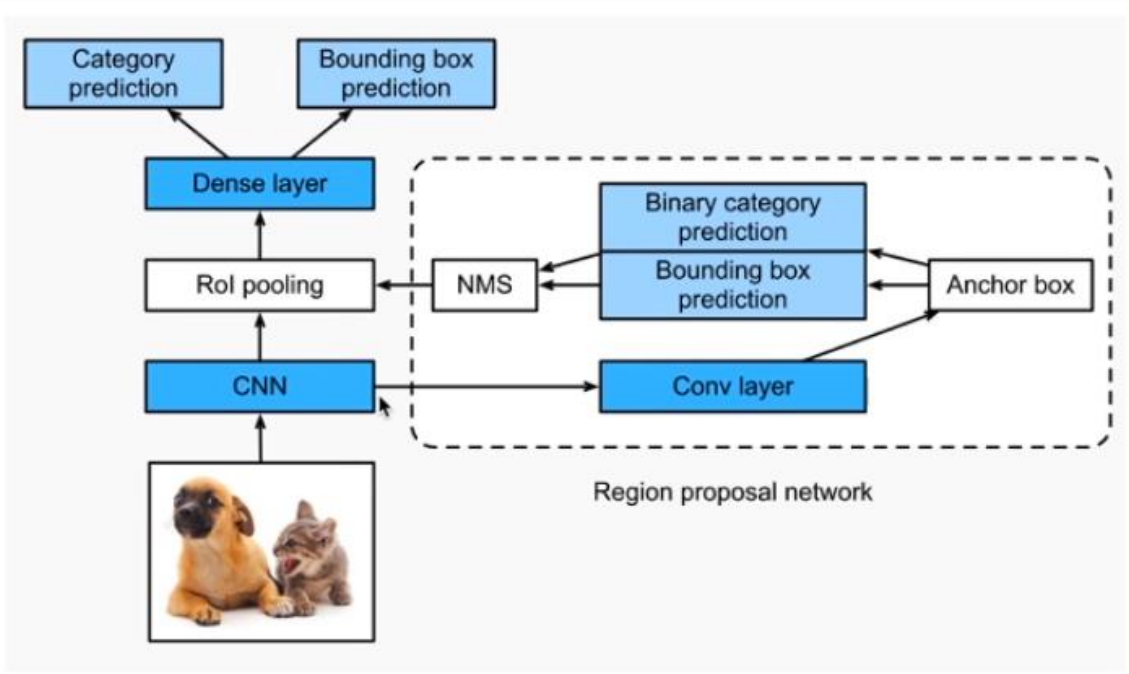
2.3
0.32

Question：Fast RCNN的运行时间主要由区域建议主导

Solustion: 使用RPN替代选择性搜索（Selective Search）算法做区域建议，生成RoIs



Fast R-CNN



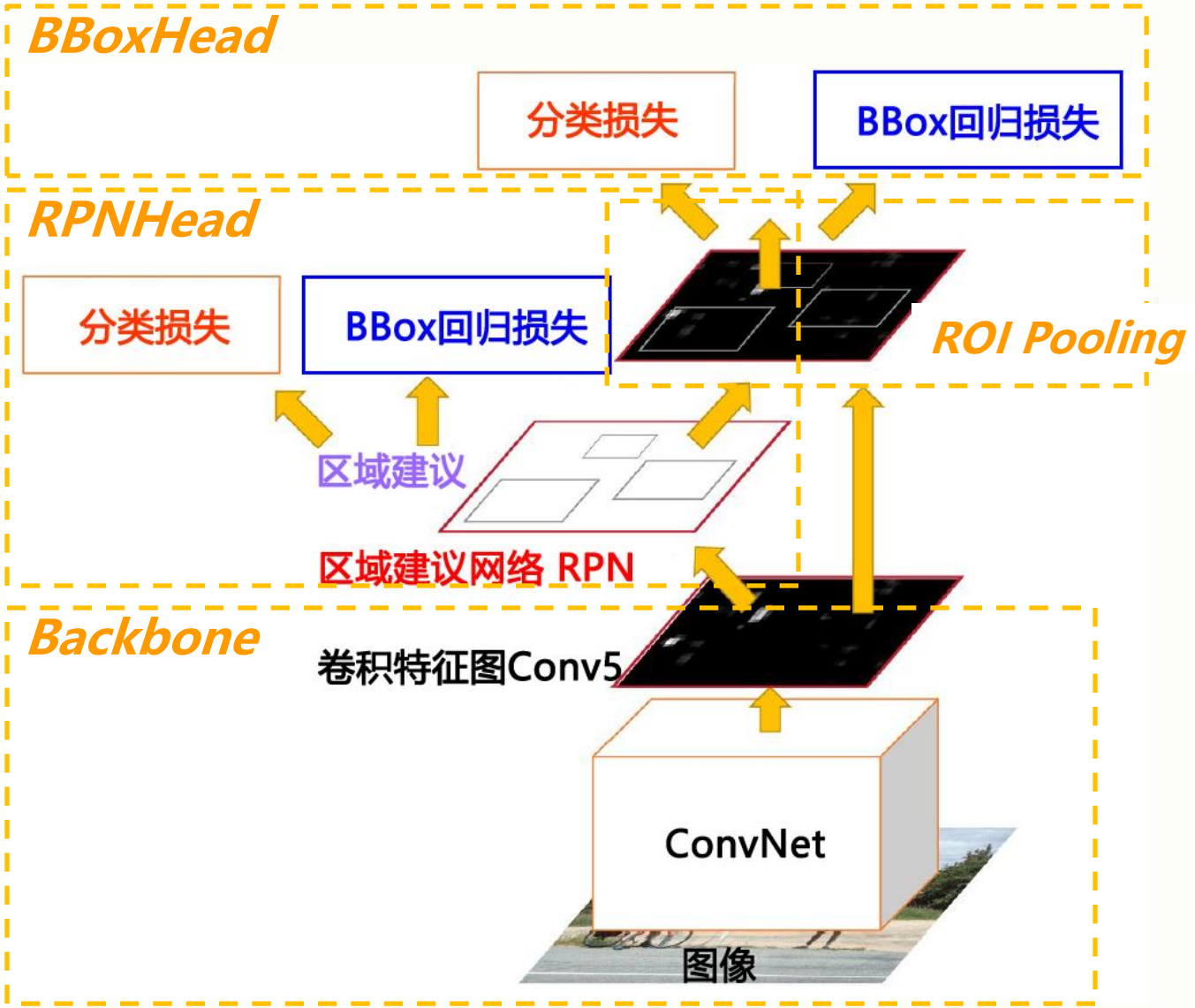
Faster R-CNN

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

使用RPN来做区域建议，以替代选择性搜索算法生成RoIs

1. 使用卷积神经网络生成样本的特征图（*Backbone*）
2. 使用**区域建议网络(Region Proposal Network, RPN)**替代Selective Search选取候选区域（*RPNHead*）
3. 使用RoI Pooling对包含物体的Anchor进行特征提取
4. 对候选区域进行分类并预测目标物体位置（*BBoxHead*）

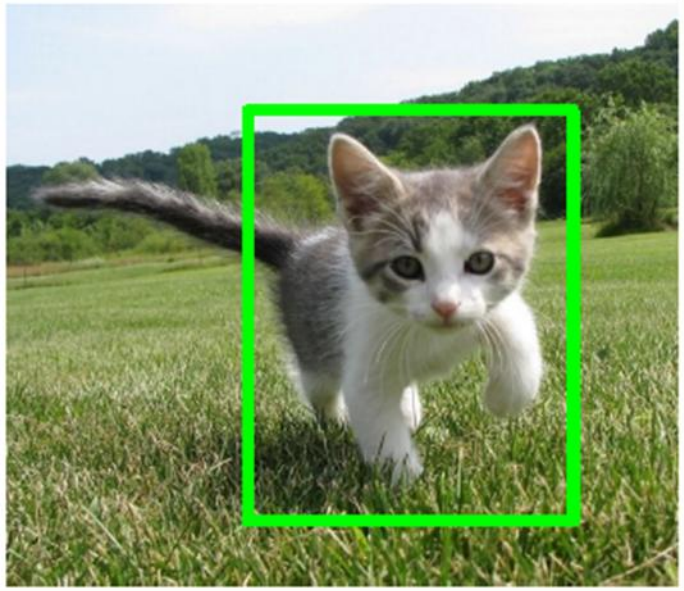


Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

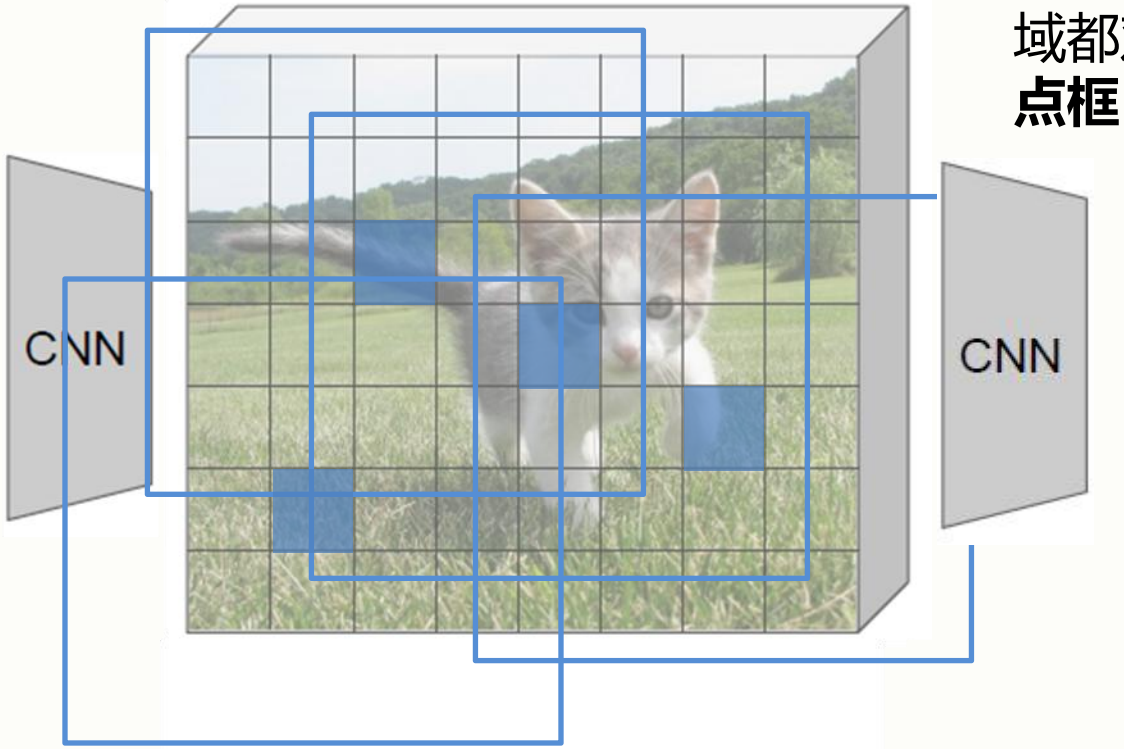
4.4 Faster R-CNN

区域建议网络 Region Proposal Network

想象在特征图的每个像素点区域都对应于一个固定尺寸的**锚点框(anchor)**。



输入图像
(e.g. 3x640×480)

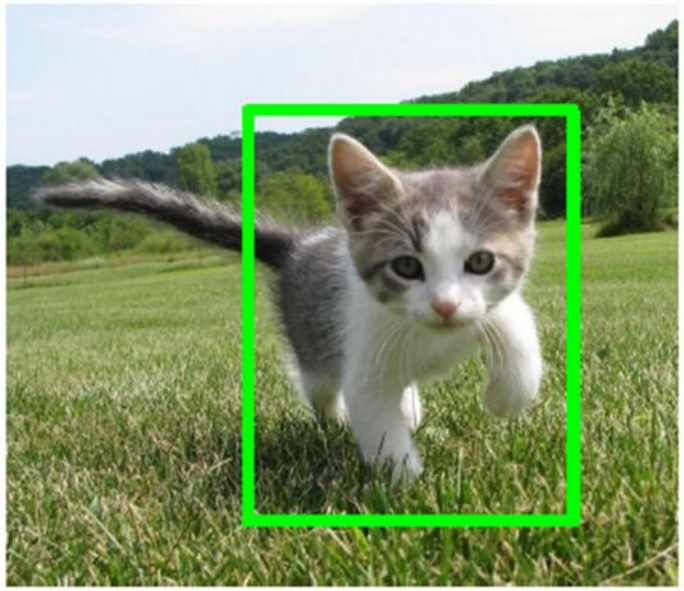


图像特征
(e.g. 512x20×15)

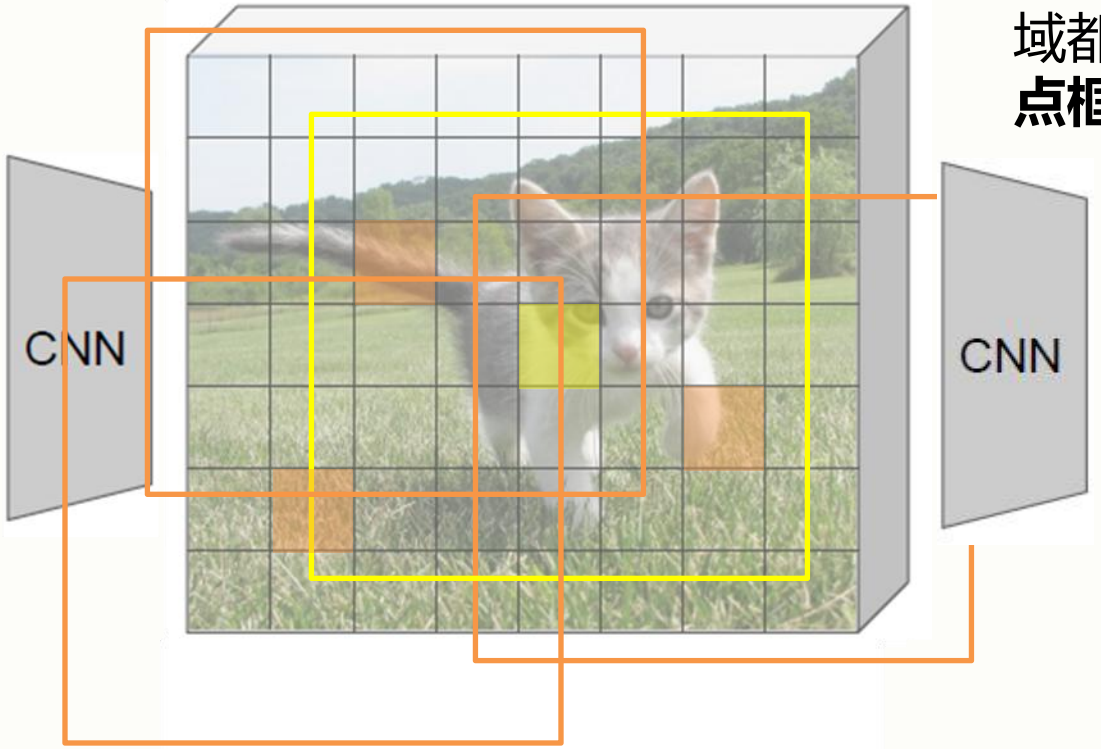
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

区域建议网络 Region Proposal Network



输入图像
(e.g. 3x640×480)



图像特征
(e.g. 512x20×15)

想象在特征图的每个像素点区域都对应于一个固定尺寸的**锚点框(anchor)**。

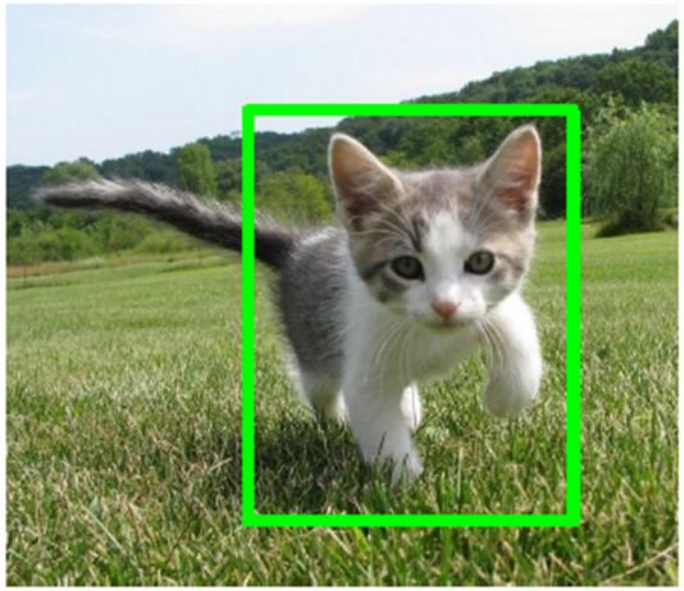
锚点是否是一个对象?
Yes or No
1×20×15

对于每个像素点，都预测锚点是否包含一个对象（二值分类）

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

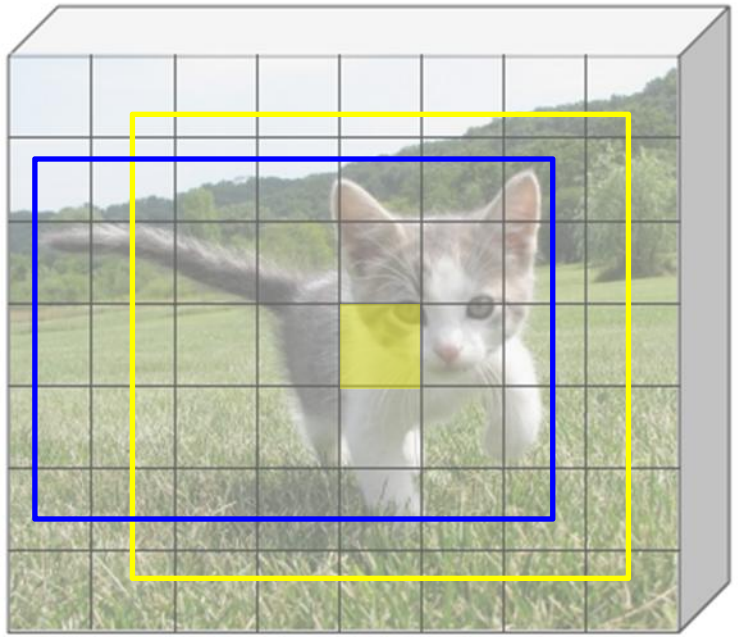
4.4 Faster R-CNN

区域建议网络 Region Proposal Network



输入图像
(e.g. 3x640×480)

CNN



图像特征
(e.g. 512x20×15)

CNN

想象在特征图的每个像素点上都有与一个固定尺寸的**锚点(anchor)**。

锚点是否是一个对象?
Yes or No
1×20×15

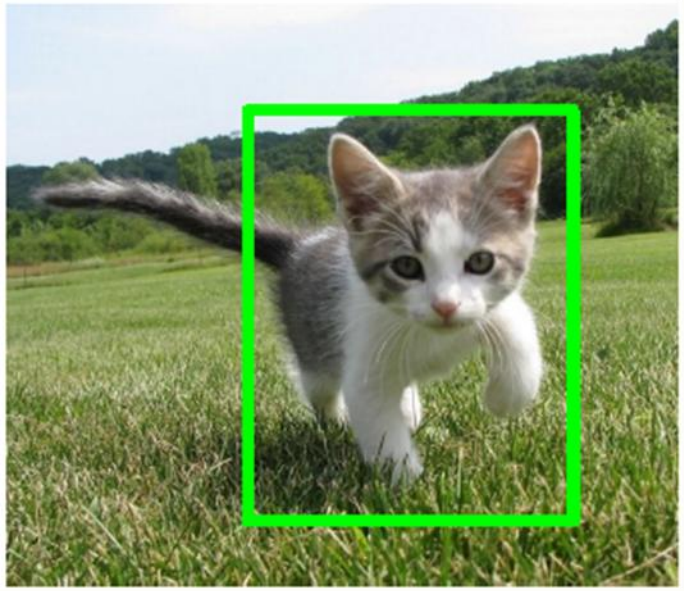
边界框精调
4×20×15

对于预测为真的边界框，需要对其进行Bbox位置修正

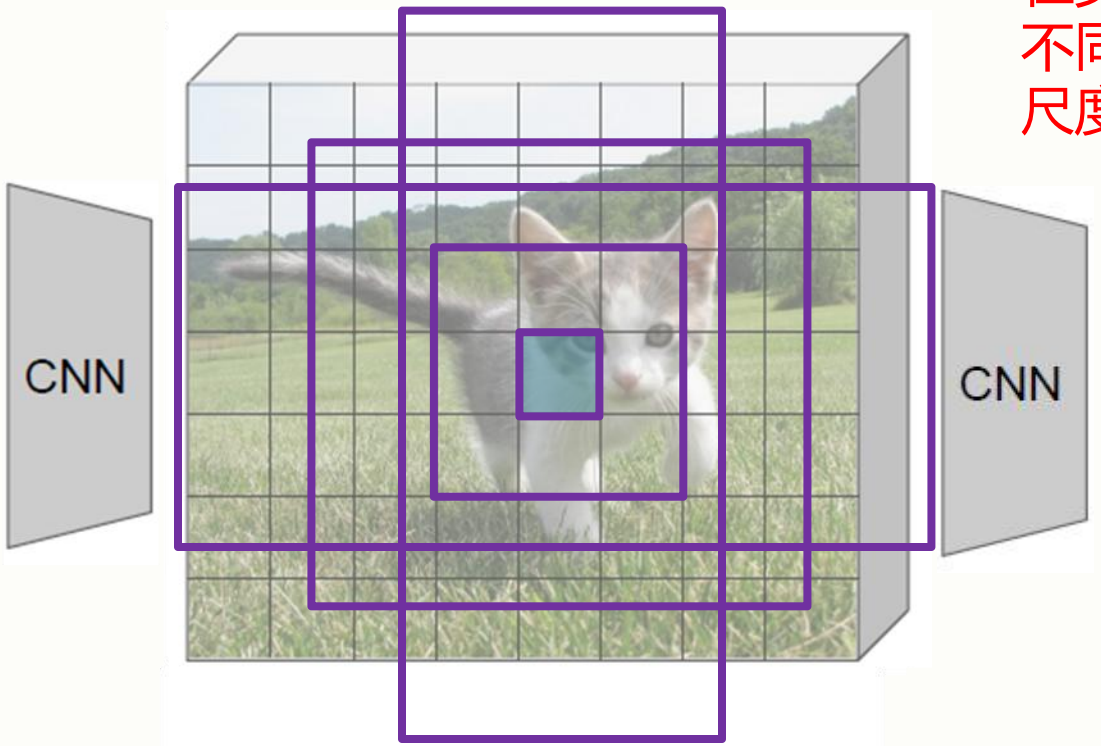
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

区域建议网络 Region Proposal Network



输入图像
(e.g. 3x640×480)



图像特征
(e.g. 512x20×15)

在实践中，每个锚点都会定义K个不同的候选框，它们包含不同的尺度和长宽比。

锚点是否是一个对象?
Yes or No
K×20×15

边界框精调
4K×20×15

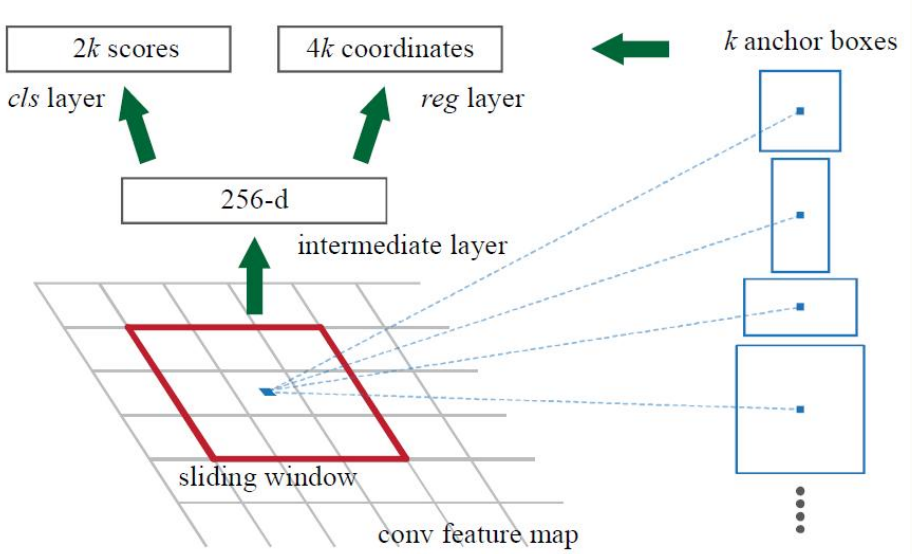
按照**对象性**分数，对这K×20×15个边界框进行排序，全图取大约~300个建议框作为RoIs.

对象性Objectness：
可能是一个对象的几率。

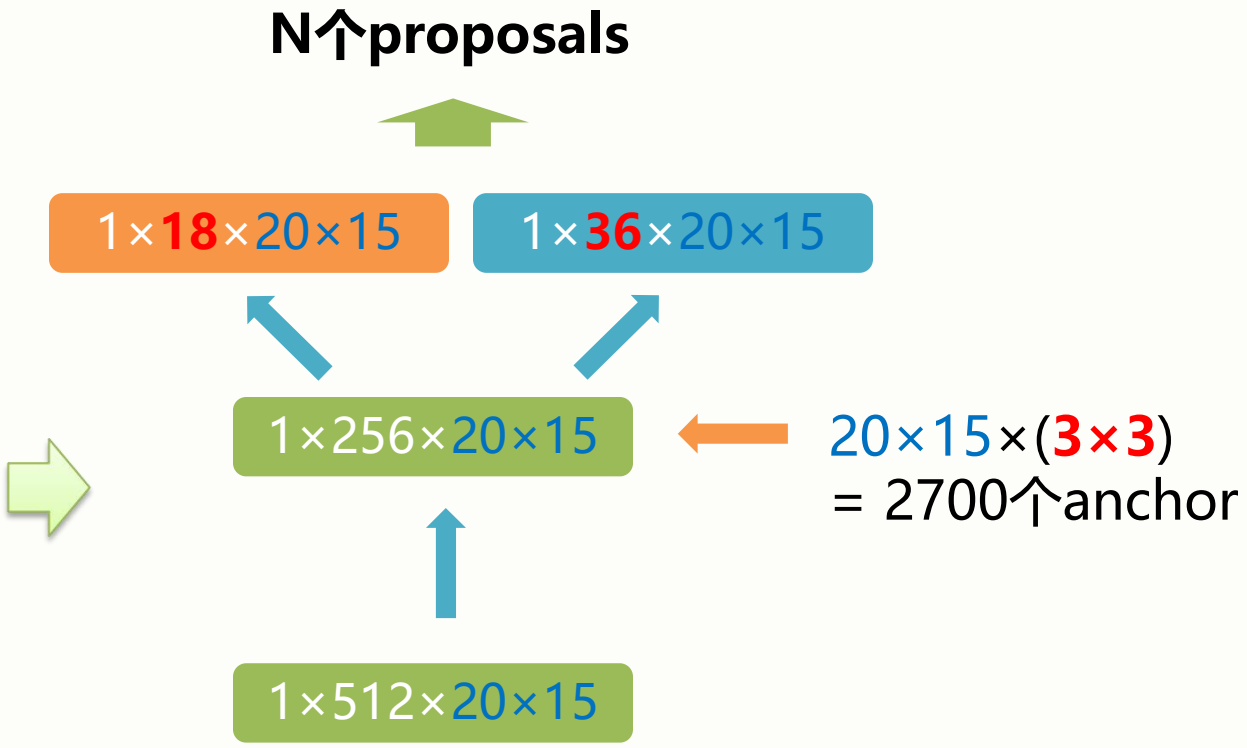
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

区域建议网络 Region Proposal Network



512个20×15的特征图

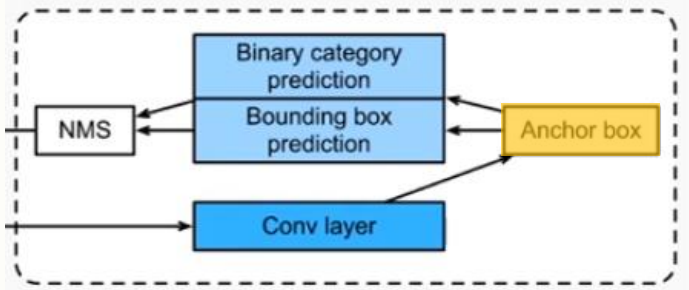


1. RPN网络怎么训练?
2. 如何使用RPN网络得到proposals?

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

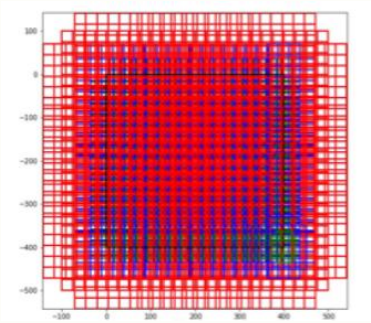
4.4 Faster R-CNN

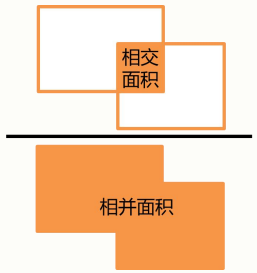
RPN网络的训练策略



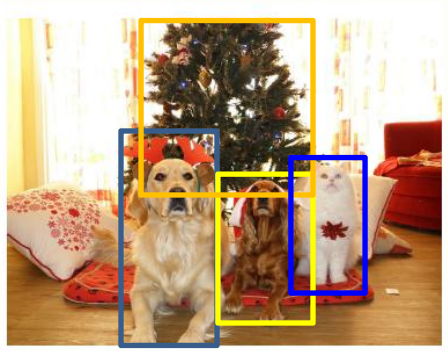
1. 向RPN网络输入一个监督信息，根据Anchor和真实框IoU的取值，判断Anchor是否包含物体，即判断Anchor是正样本or负样本。

- 包含物体的Anchor → 正样本
  - ✓ 正样本：与某一真实框IoU最大的Anchor
    与任意真实框的IoU>0.7的Anchor
- 不包含物体的Anchor → 负样本
  - ✓ 负样本：与所有真实框的IoU<0.3的Anchor





2. 采样规则：
   - ✓ 共采样256个样本
   - ✓ 从正样本中随机采样，采样个数不超过128个
   - ✓ 从负样本中随机采样，补齐256个样本

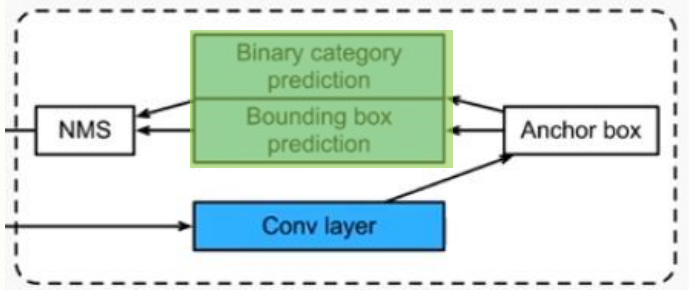


Anchor \ 真实框	1	2	3	4
1	0.8	0.3	0.1	0.4
2	0.2	0.9	0.3	0.2
3	0.1	0.2	0.1	0.2
4	0.4	0.1	0.6	0.2
5	0.3	0.6	0.4	0.9
6	0.3	0.3	0.3	0.8
7	0.5	0.2	0.1	0.3

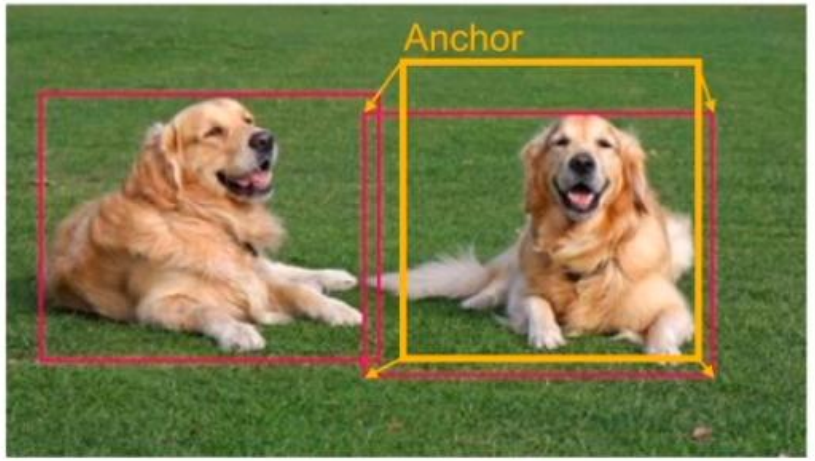
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

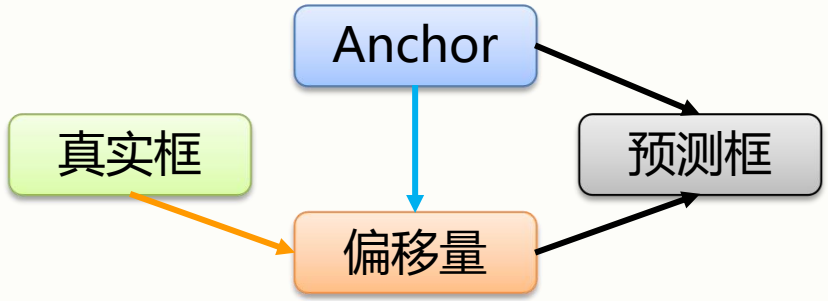
4.4 Faster R-CNN

RPN网络的监督信息



- **分类分支：候选框是否包含物体**
  - ✓ 正样本label = 1
  - ✓ 负样本label = 0
- **回归分支：Anchor到真实框的偏移量 t**
  - $t_x^* = (x^* - x_a)/w_a$， $t_y^* = (y^* - y_a)/h_a$
  - $t_w^* = log(w^*/w_a)$， $t_h^* = log(h^*/h_a)$
  - $t_x = (x - x_a)/w_a$， $t_y = (y - y_a)/h_a$
  - $t_w = log(w/w_a)$， $t_h = log(h/h_a)$
  - Anchor: x_a, y_a, w_a, h_a
  - 真实框: x^*, y^*, w^*, h^*
  - 预测框: x, y, w, h

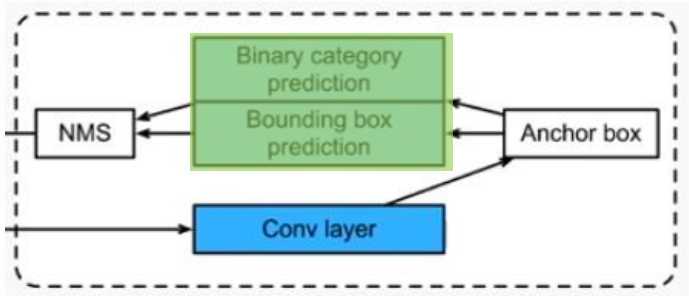




Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

RPN网络的损失函数



- **RPN网络的*Loss*计算公式**

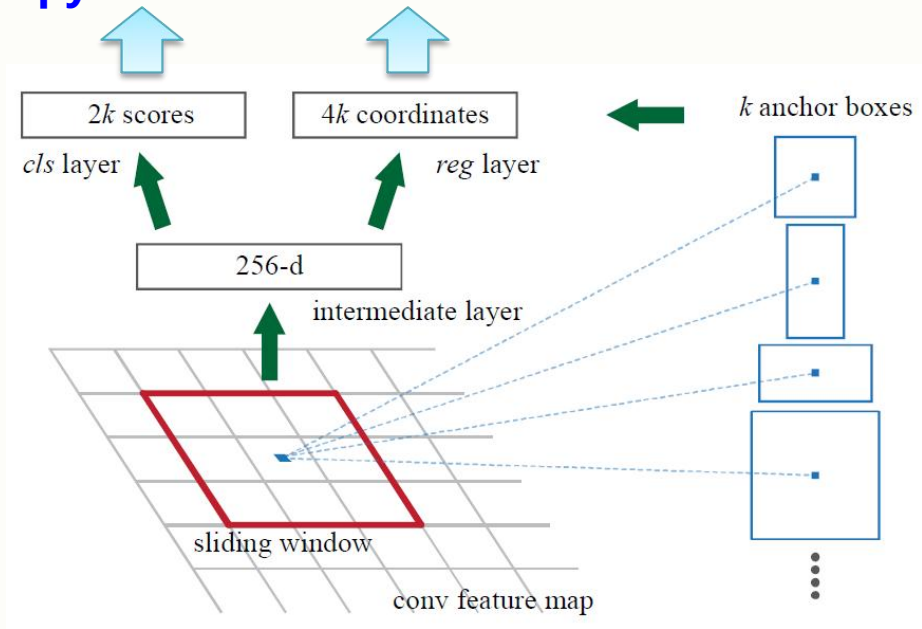
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

RPN网络的*Loss*由分类和回归两个*loss*相加获得

- p: 分类分支的预测值
- t: 回归分支的预测值
- p^*: 表示分类监督信息，取值为0或1
  - ✓ 1: 表示Anchor为正样本
  - ✓ 0: 表示Anchor为负样本
- t^*: 表示回归分支的监督信息

Binary Cross Entropy

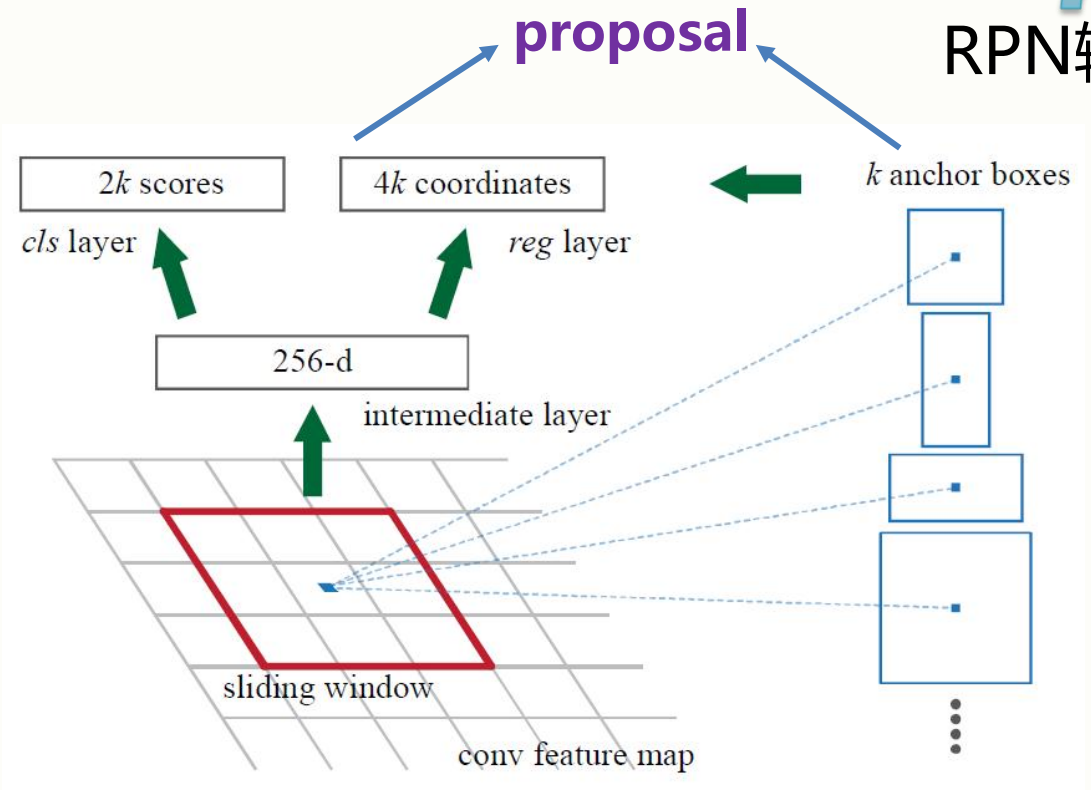
Smooth L1



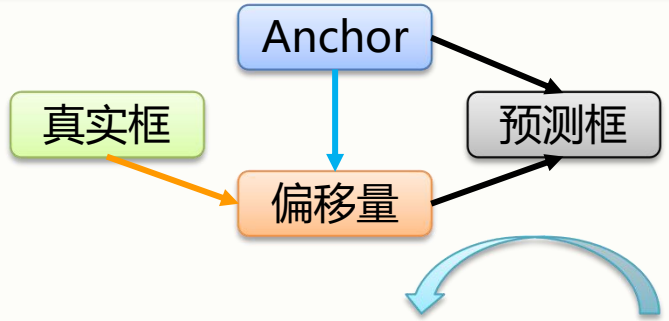
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

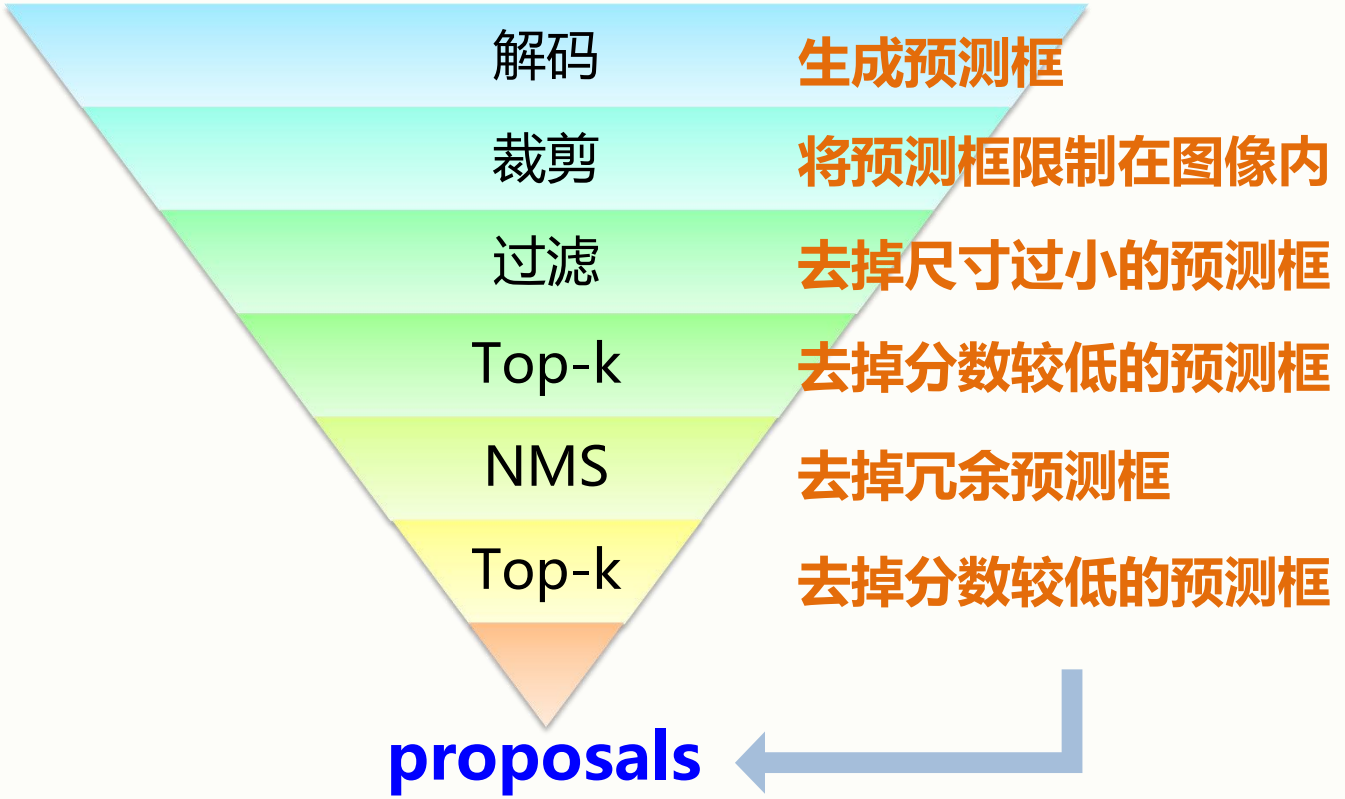
Proposals的生成



RPN输出



Anchor



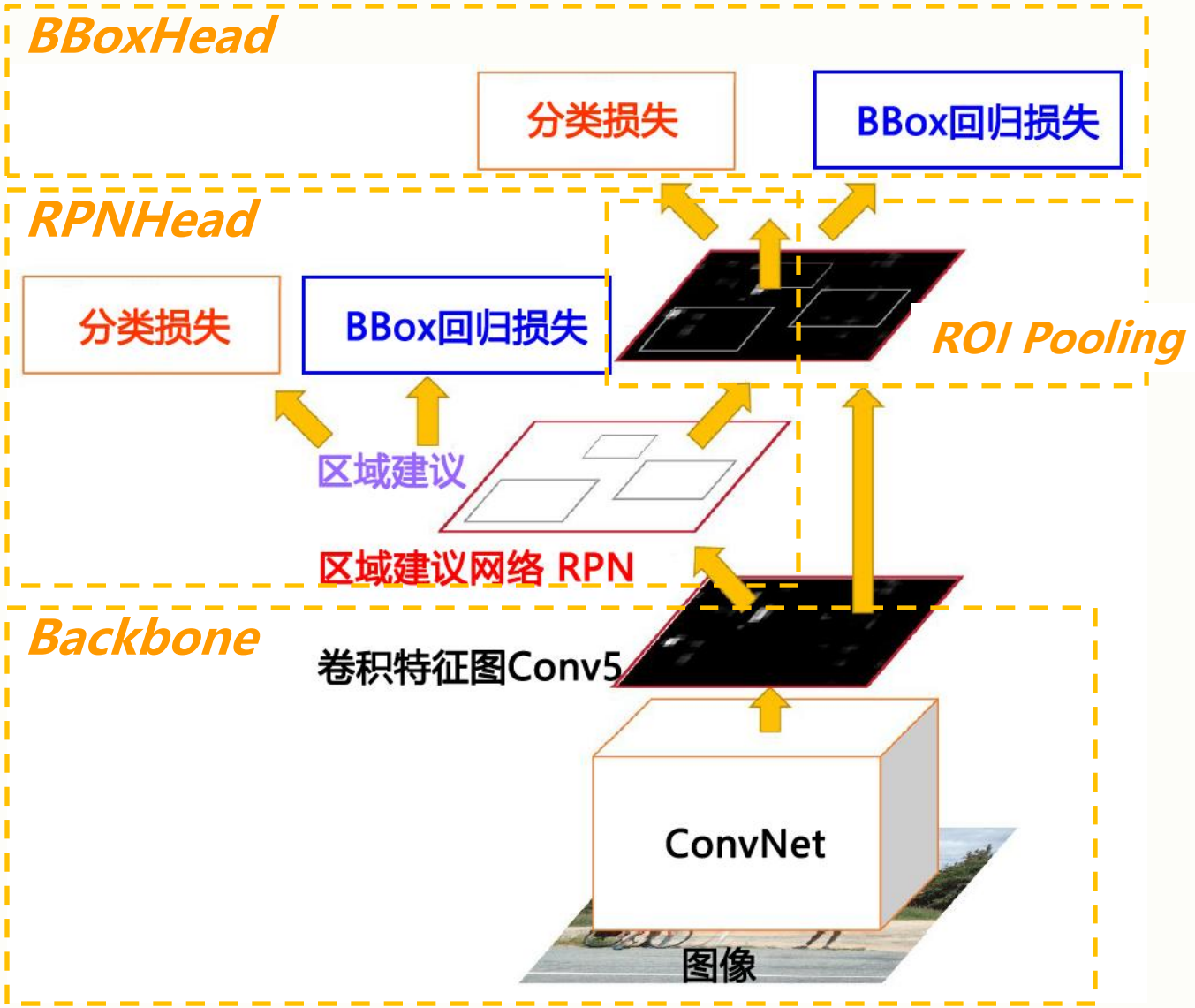
proposals

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

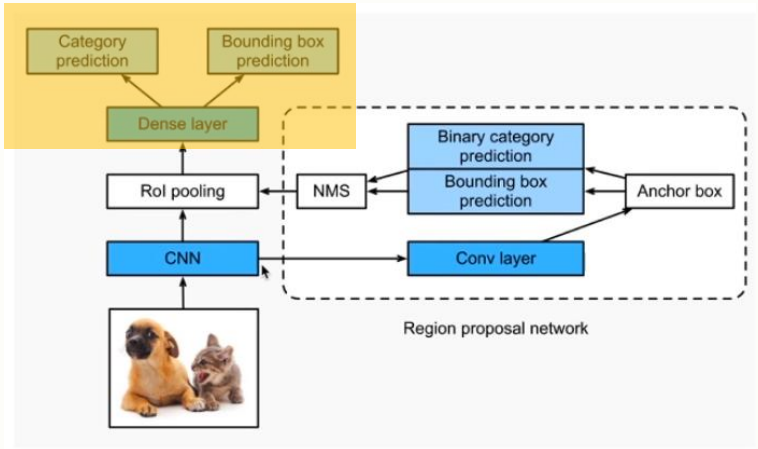
使用Bbox Head实现目标检测

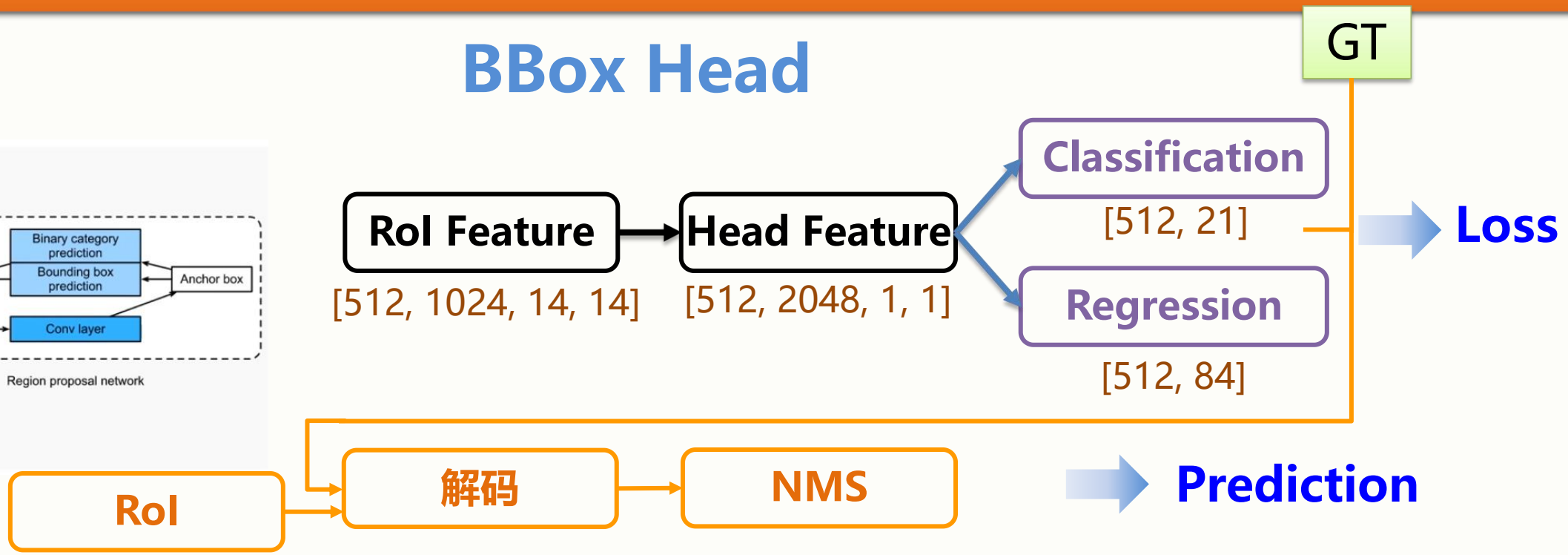
- ✓ 候选区域直接由RPN网络生成，不再依靠额外的区域建议算法（如SS）
- ✓ 根据RPN提供的RoIs使用RoI Pooling生成基于区域的特征，这些特征通过两个分支分别（*BBoxHead*）计算每个RoI的类别和边界框回归



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN





- **训练阶段**，RoI特征经过RoI Pooling从14×14压缩到1×1得到HeadFeature，然后被送入两个不同的FC层作为分类分支和回归分支进行预测。最后计算Loss，并进行反向传播。
- **预测阶段**和训练阶段生成proposals基本相似，BBoxHead和RPN的输出得到RoI解码的预测框，再通过NMS得到最终预测结果。

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

BBox Head

RPN网络训练阶段：

1．向RPN网络输入一个监督信息，根据Anchor和真实框IoU的取值，判断Anchor是否包含物体，即判断Anchor是正样本or负样本。

- ✓ 正样本：IoU>0.7
- ✓ 负样本：IoU<0.3

2．采样规则：

- ✓ 共采样256个样本
- ✓ 从正样本中随机采样，采样个数不超过128个
- ✓ 从负样本中随机采样，补齐256个样本

BBox Head训练阶段：

1．向Bbox Head输入一个监督信息，判断RPN网络生成的proposals是否包含物体，即判断RoI是正样本or负样本。

- ✓ 正样本：IoU>0.5
- ✓ 负样本：IoU<0.5

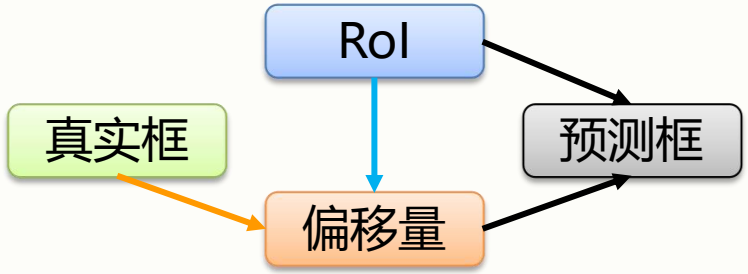
2．采样规则：

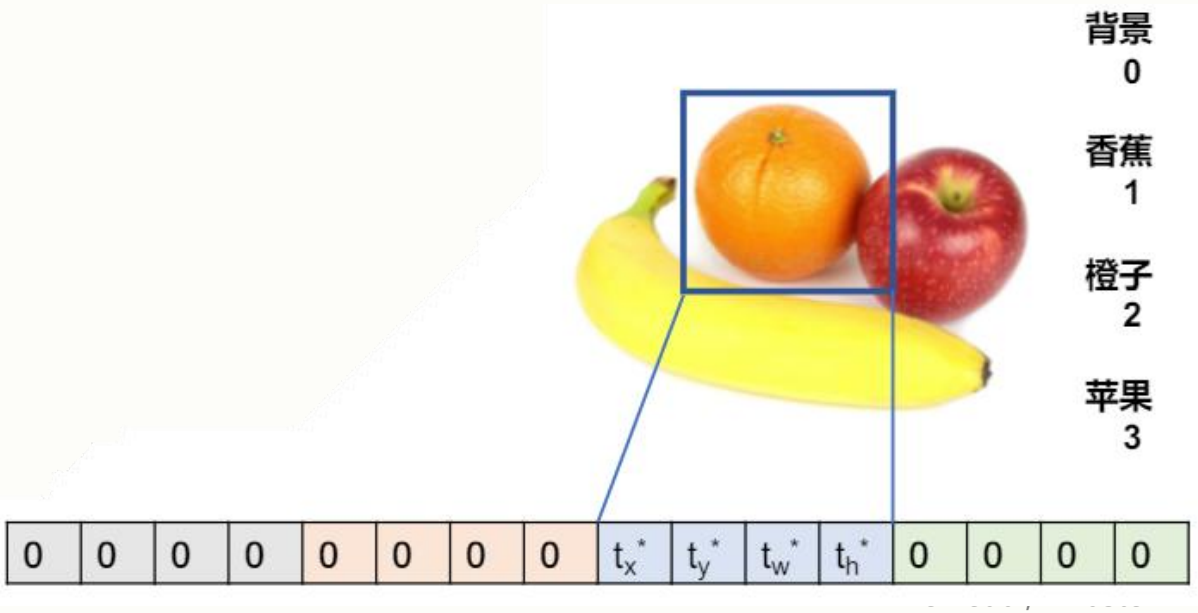
- ✓ 共采样512个样本
- ✓ 从正样本中随机采样，采样个数不超过128个
- ✓ 从负样本中随机采样，补齐512个样本

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

BBox Head 中的监督信息





- **分类分支：**学习每个预测框的类别
  - ✓ Label = [0, 1, 2, 3, ...]
- **回归分支：**学习每个RoI到真实框的偏移量
  - ✓ $t_x^* = (x^* - x_a)/w_a$， $t_y^* = (y^* - y_a)/h_a$
  - ✓ $t_w^* = log(w^*/w_a)$， $t_h^* = log(h^*/h_a)$
  - ✓ $t_x = (x - x_a)/w_a$， $t_y = (y - y_a)/h_a$
  - ✓ $t_w = log(w/w_a)$， $t_h = log(h/h_a)$
  - ◆ RoI: x_a, y_a, w_a, h_a
  - ◆ 真实框: x^*, y^*, w^*, h^*
  - ◆ 预测框: x, y, w, h

CNN: Towards Real-Time Object Detection with Region Proposal Networks”，NIPS 2015

4.4 Faster R-CNN

BBox Head的损失函数

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

多分类问题：Softmax Cross Entropy

回 归 问 题：Smooth L1

- p: 分类分支的预测值
- t: 回归分支的预测值
- p^*: 表示分类监督信息，取值为0或1
  - ✓ 1: 表示RoI为正样本
  - ✓ 0: 表示RoI为负样本
- t^*: 表示回归分支的监督信息

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

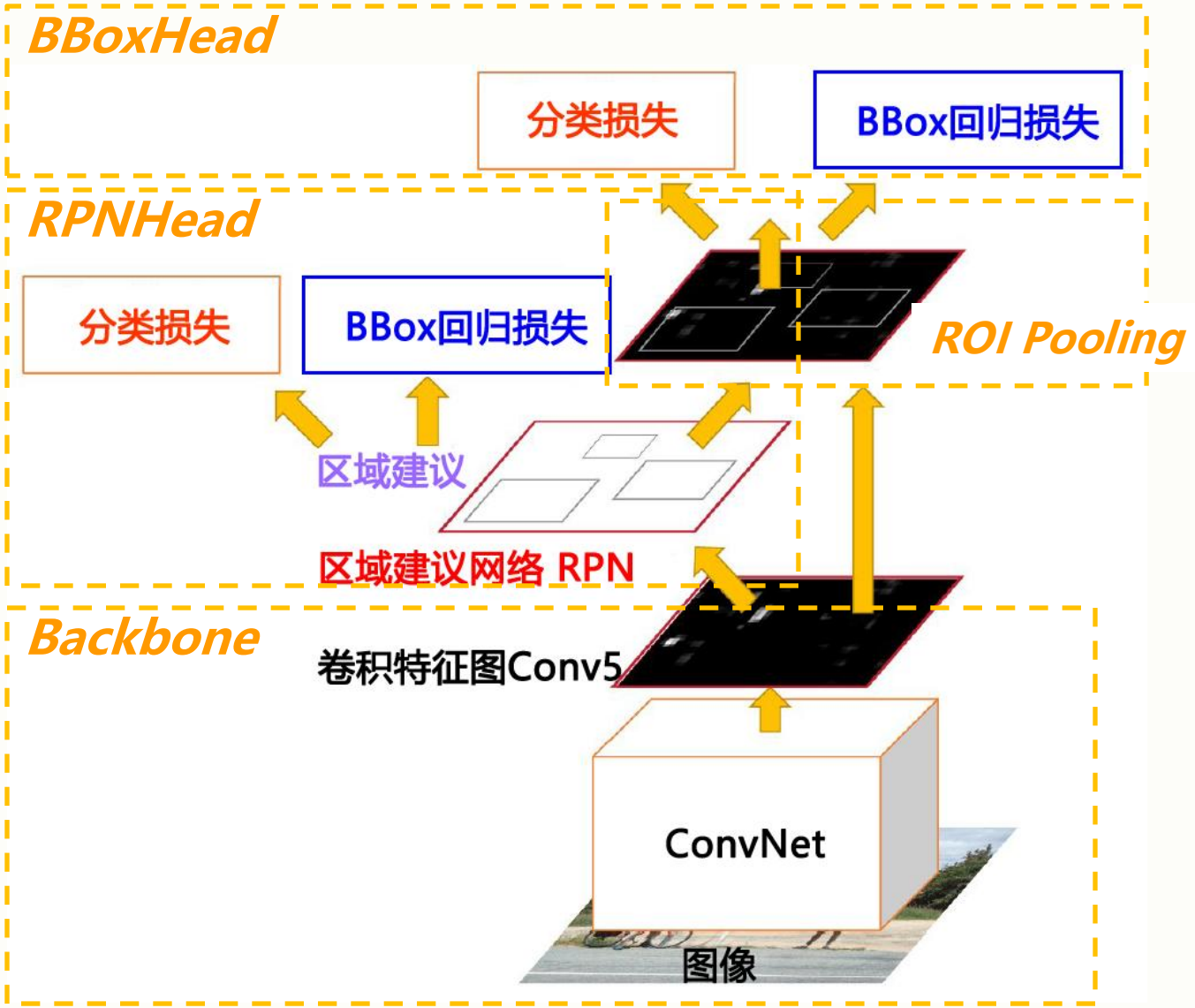
Faster R-CNN的训练

端到端的检测网络，极大地提升了训练和检测的速度

$$L_{total} = L_{RPN} + L_{BBoxHead}$$

训练时，4个损失函数进行联合训练:

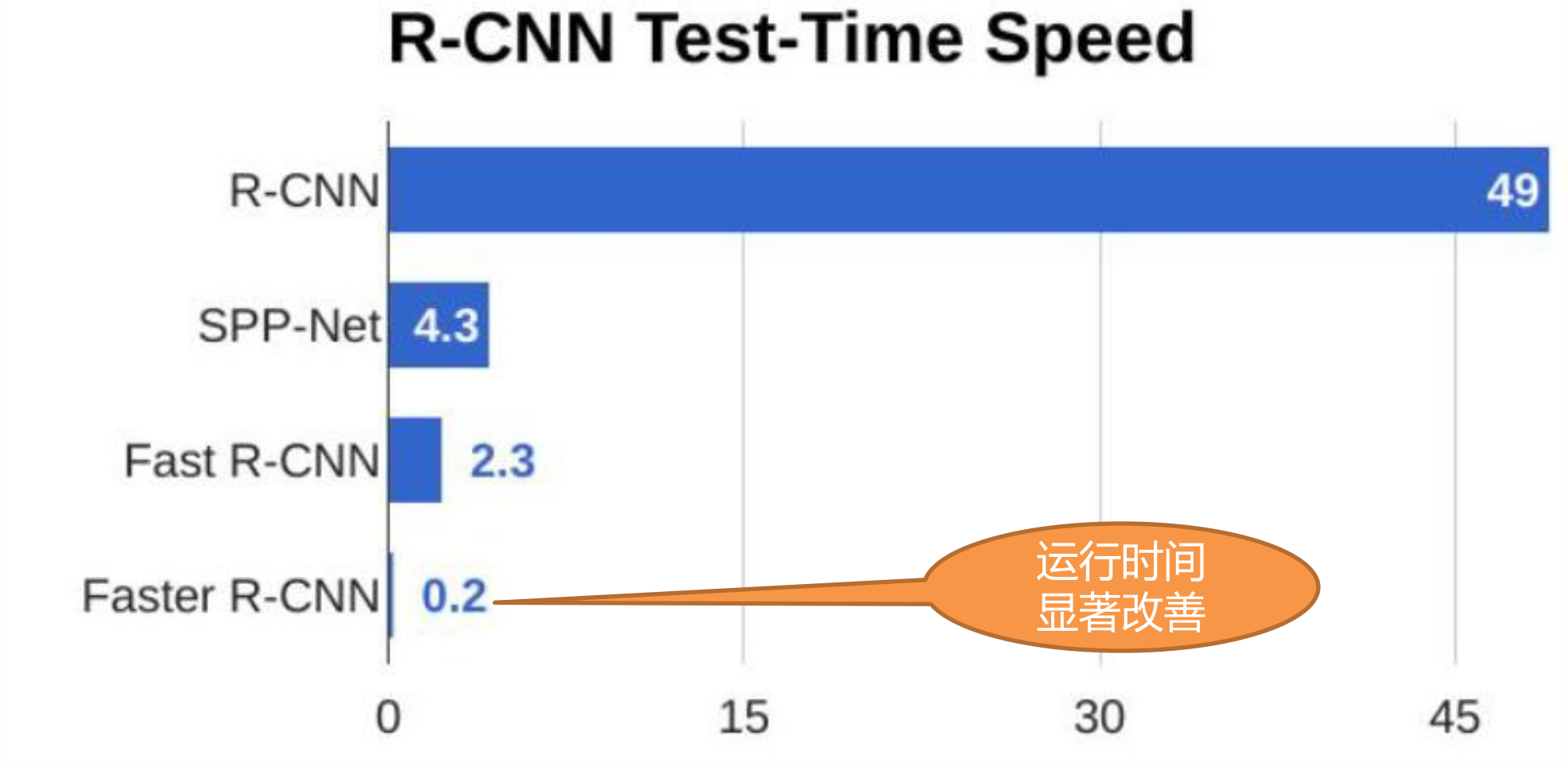
1. RPN分类是对象/不是对象
2. RPN回归边界框的坐标
3. BBoxHead的类别分类分数
4. BBoxHead的边界框的坐标



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

4.4 Faster R-CNN

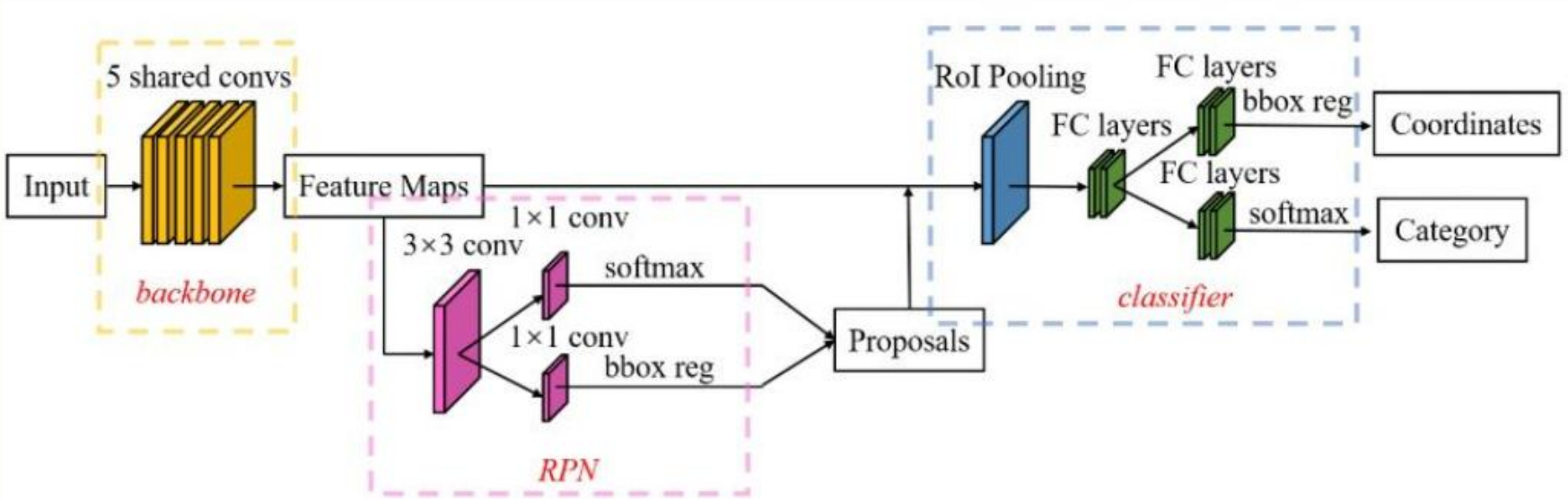
R-CNN系列检测模型测试时间对比



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

Faster R-CNN 体系结构图



1. **特征提取**：使用卷积神经网络提取整个图像的特征图，该特征图被RPN和目标分类回归网络共享；
2. **生成候选区域：**由RPN产生候选区域并通过对象性排序获得RoIs(训练选择2000个，测试选择300个)
3. **RoI Pooling**：将候选区域映射到特征图上，并经过RoI Pooling得到固定长度的特征，输出到全连接层
4. **分类及边界精调**：经过**2+1个全连接层**后，特征被分别送入Softmax分类器和Bbox回归器
5. **非极大抑制**：对每个类进行NMS去除冗余

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

从Fast R-CNN到Faster R-CNN预测过程的变化

1. **生成候选区域**：输入原始图像，使用SS生成~2k的RoIs；
2. **特征提取**：使用卷积神经网络提取整个图像的特征图
3. **RoI Pooling**：将候选区域映射到特征图上，并经过RoI Pooling得到固定长度的特征，输出到全连接层
4. **分类及边界精调**：经过**2+1个全连接层**后，特征被分别送入Softmax分类器和Bbox回归器
5. **非极大抑制**：对每个类进行NMS去除冗余

1. **特征提取**：使用卷积神经网络提取整个图像的特征图，该特征图被RPN和目标分类回归网络共享；
2. **生成候选区域：**由RPN产生的候选区域，通过对象性排序获得RoIs(训练时选择2000个，与测试选择300个)
3. **RoI Pooling**：将候选区域映射到特征图上，并经过RoI Pooling得到固定长度的特征，输出到全连接层
4. **分类及边界精调**：经过**2+1个全连接层**后，特征被分别送入Softmax分类器和Bbox回归器
5. **非极大抑制**：对每个类进行NMS去除冗余

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

4.4 Faster R-CNN

两级网络结构

我们是否真的需要两级结构?

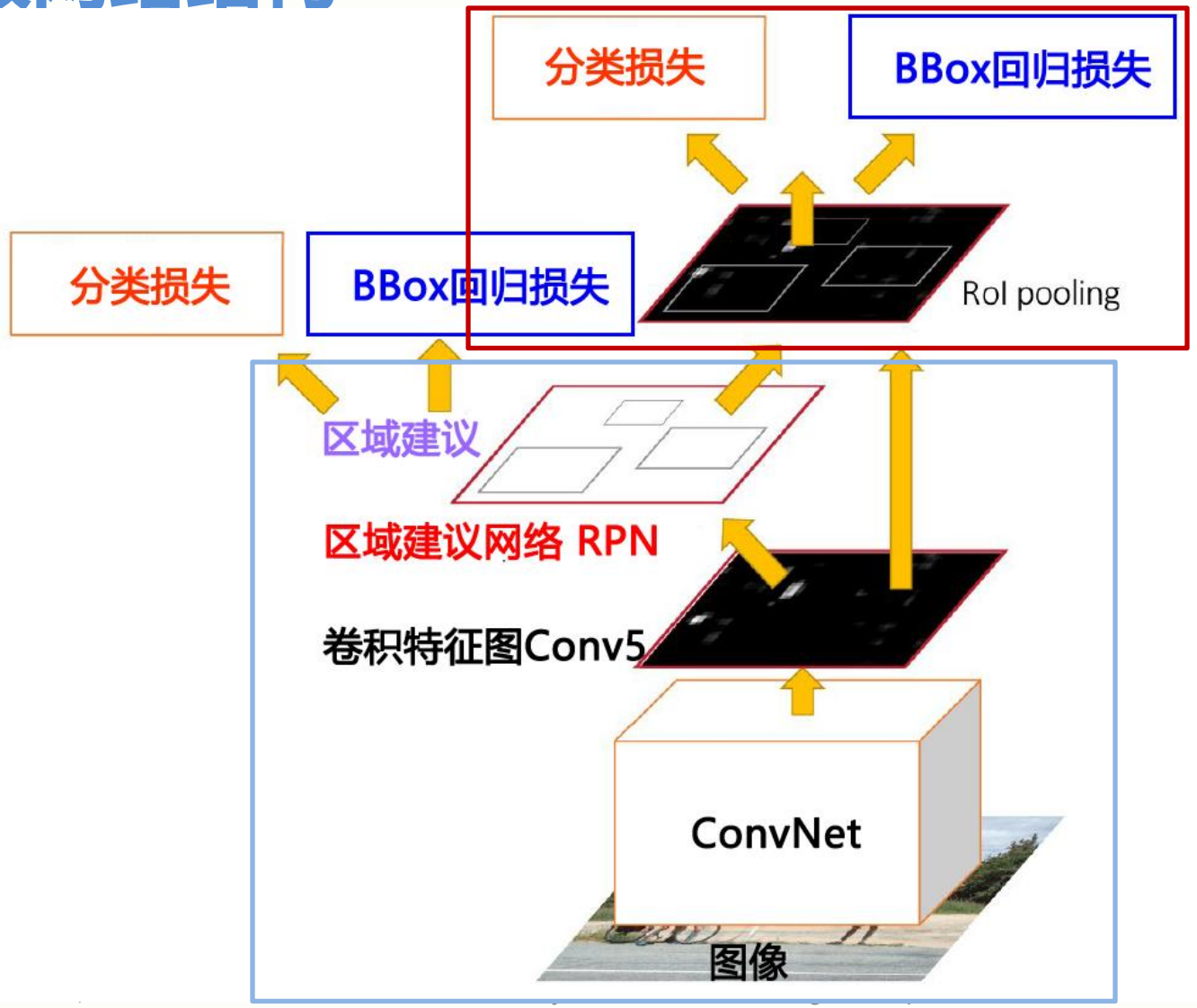
*Faster R-CNN 是一个**两级**对象检测器*

第一级：每个图像运行一次

- 主干网络 (AlexNet, VGGNet, ResNet)
- 区域建议网络

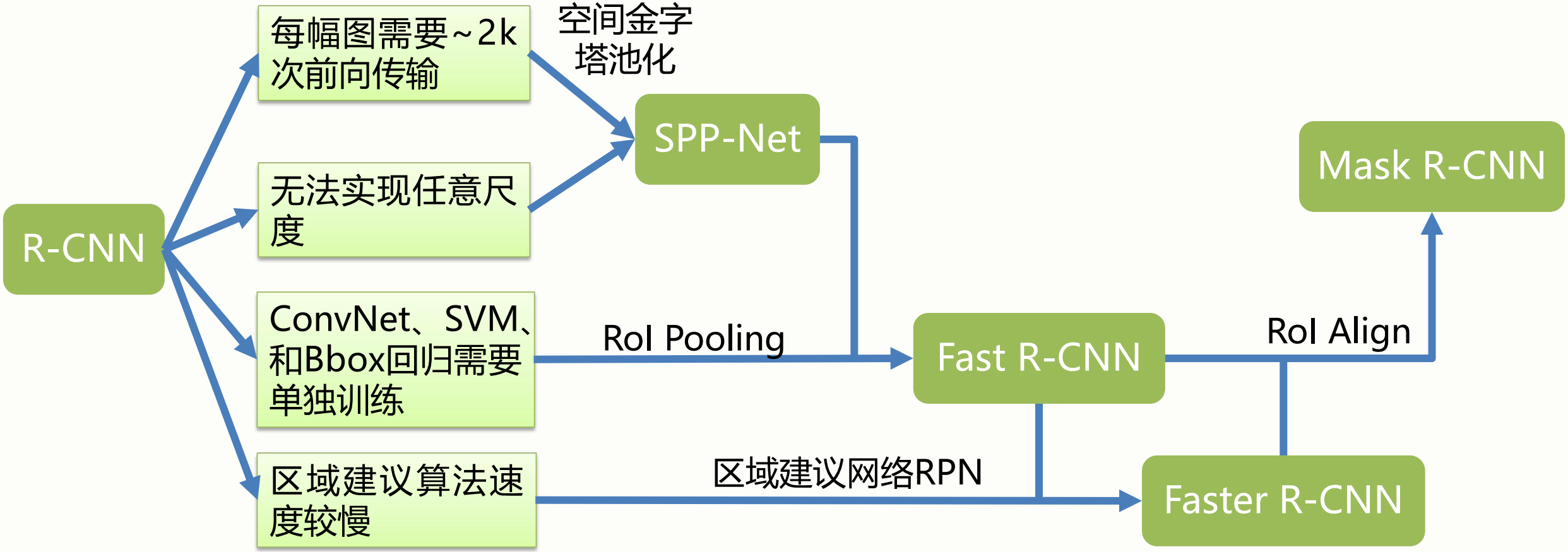
第二级：每个区域运行一次

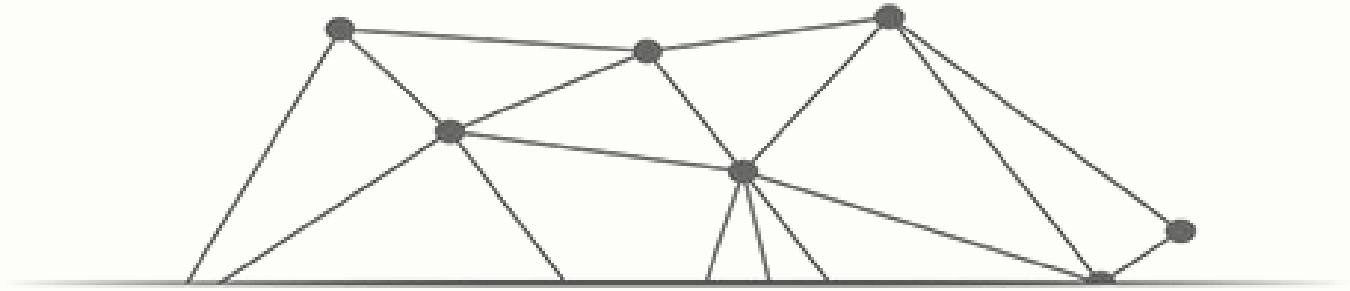
- 切割特征 (RoI Pooling, RoI Align)
- 预测对象类别
- 预测Bbox偏移量（Bbox 精调）



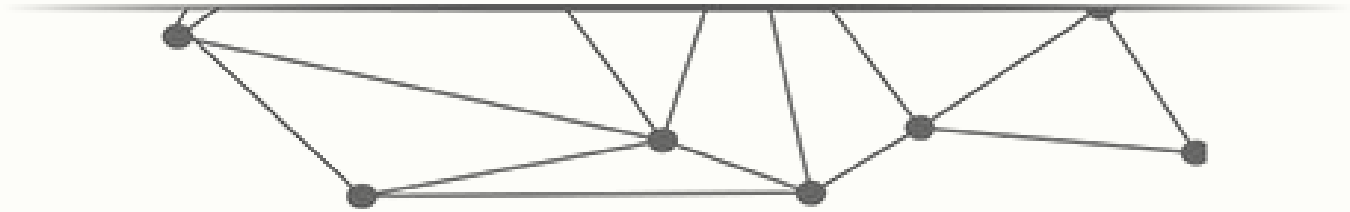
4. 基于两阶段方法的目标检测（区域选择）

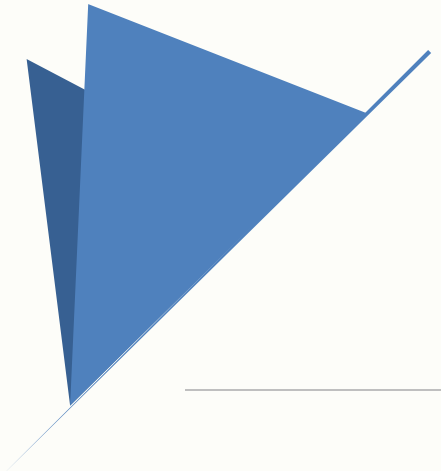
小　结





课堂互动 13.2.7

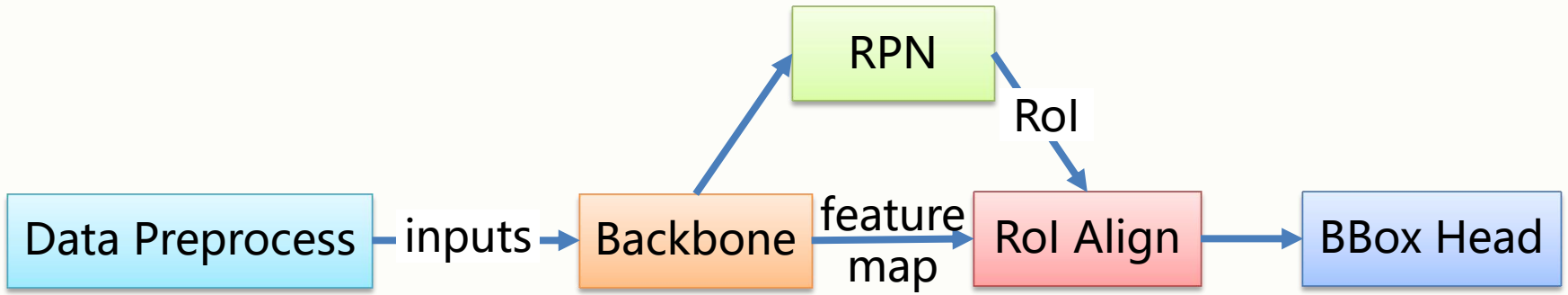




RCNN系列算法的进阶优化

4.5 RCNN系列算法的进阶优化

两阶段目标检测算法的进阶模型



改进的Faster RCNN

- FPN
- Cascade R-CNN
- Libra R-CNN
- **Mask R-CNN**
- **RFCN**
- **Light-Head R-CNN**
- **……**

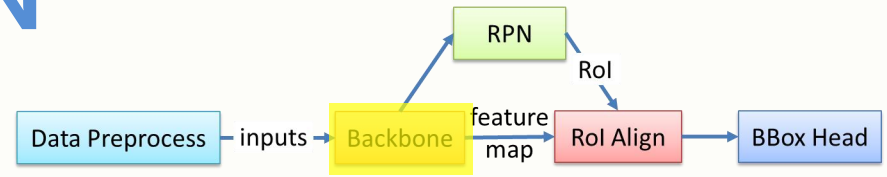
4.5 RCNN系列算法的进阶优化

特征金字塔网络 FPN

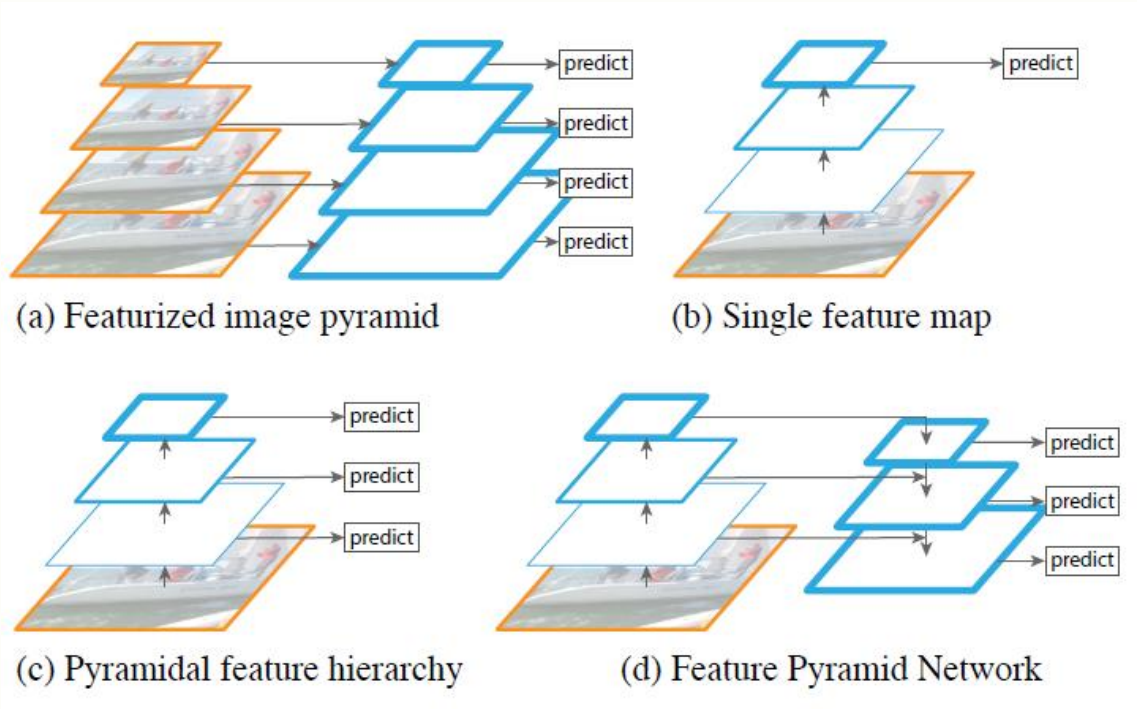
Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection. arXiv1612

4.5 RCNN系列算法的进阶优化

两阶段进阶模型FPN



多尺度金字塔使模型具备检测不同大小尺度物体的能力



(a) Featurized image pyramid: 将输入图像缩放到不同尺度，使用多个模型进行预测；

(b) Single feature map: 仅使用最后一层的特征作为检测模型后续部分的输入；

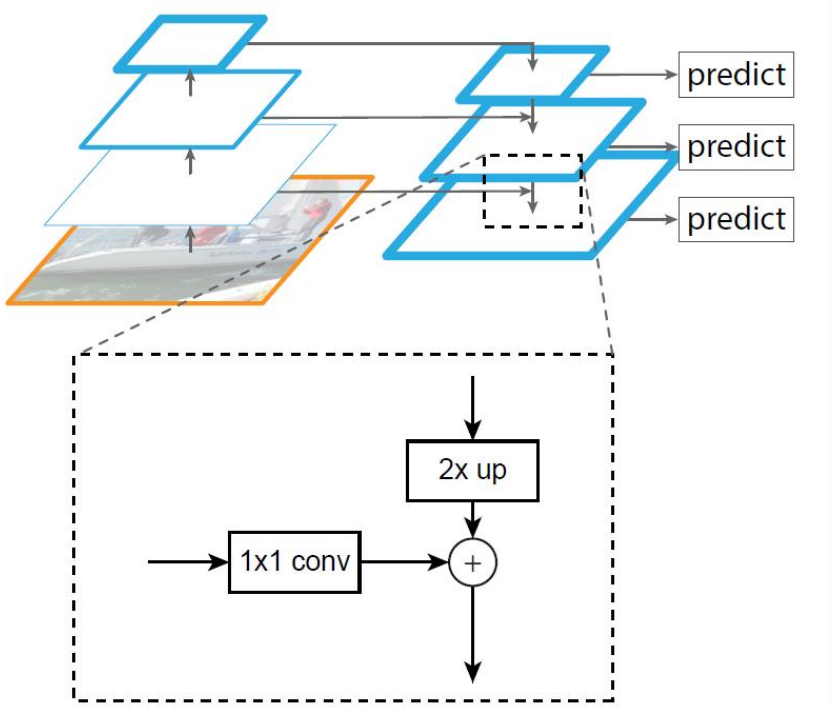
(c) Pyramidal feature hierarchy: 每个层级分别预测；

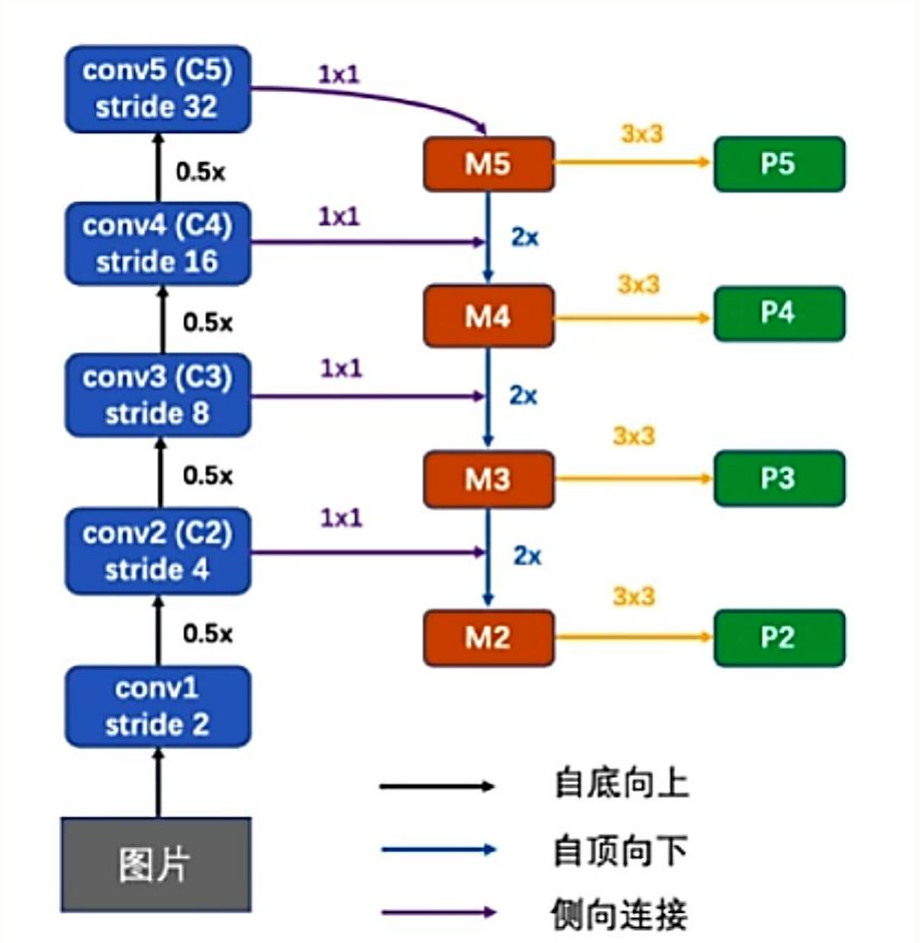
(d) Feature Pyramid Network(FPN): 将不同层的特征进行融合再分级预测。

Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection. arXiv1612

4.5 RCNN系列算法的进阶优化

自上而下的通路和侧部连接





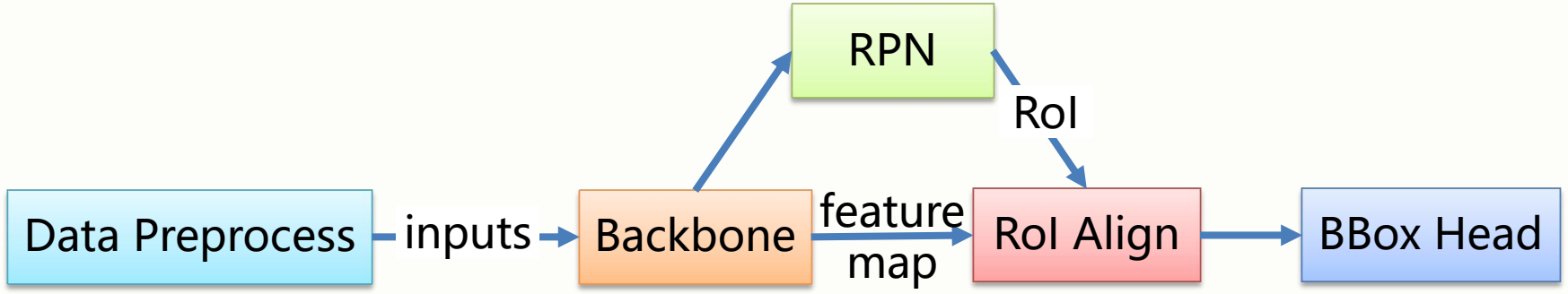
ResNet+FPN网络结构

FPN网络和骨干网络相互独立，其输入为骨干网络每层的输出；特征进行上采样后再与上一层特征相加得到FPN的输出。

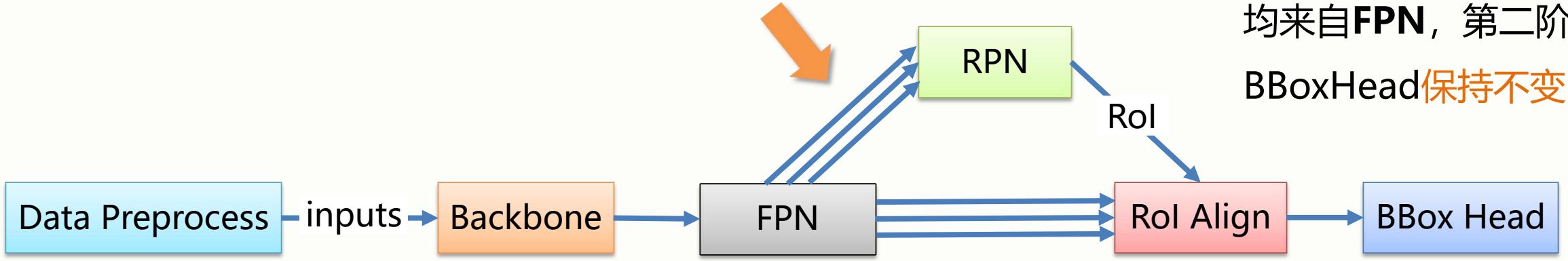
k, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection. arXiv:1612

4.5 RCNN系列算法的进阶优化

包含FPN的目标检测算法



改进的Faster RCNN



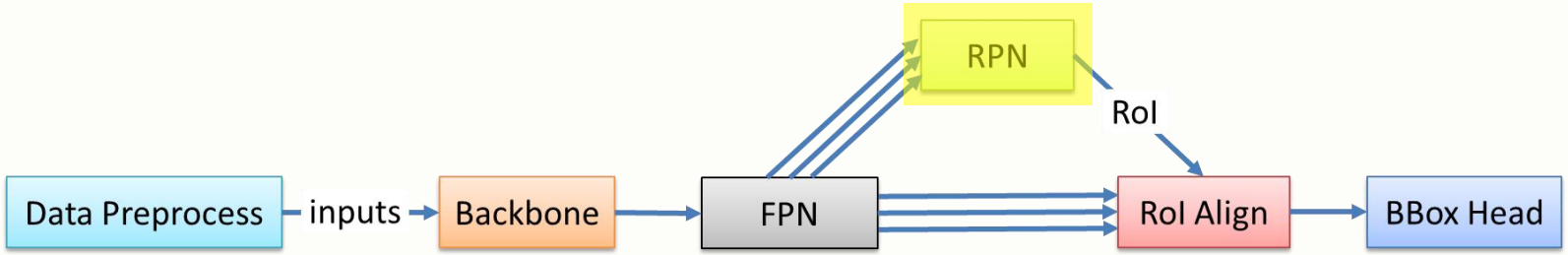
Faster RCNN + FPN

FPN是附加在骨干网络之后的独立网络结构，其输出为多个特征图。RPN模块和RoI Align模块的输入均来自**FPN**，第二阶段的BBoxHead保持不变。

Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection. arXiv:1612

4.5 RCNN系列算法的进阶优化

FPN结构下的RPN网络

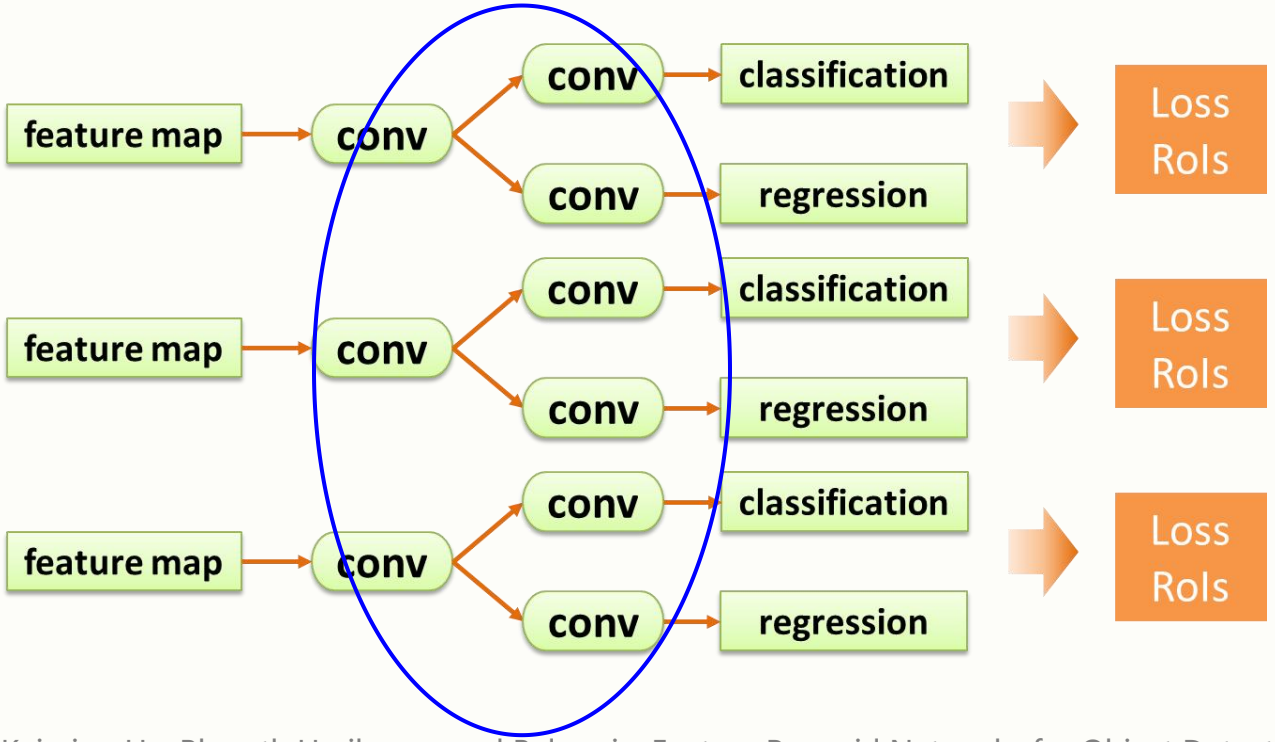


1. Anchor:

{32, 64, 128,256, 512} => $\{P_2, P_3, P_4, P_5, P_6\}$

2. RPN网络由多个Head构成，分别处理不同尺度的候选框

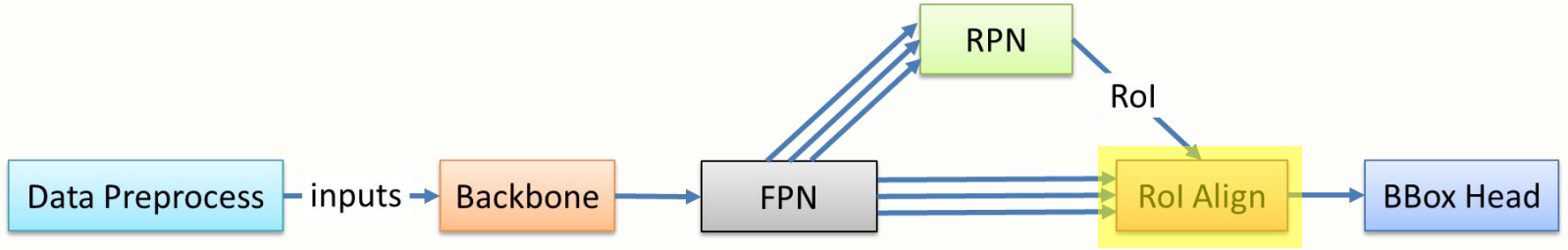
3. RPN网络的预测结果和Anchor解码得到的RoI会进行合并

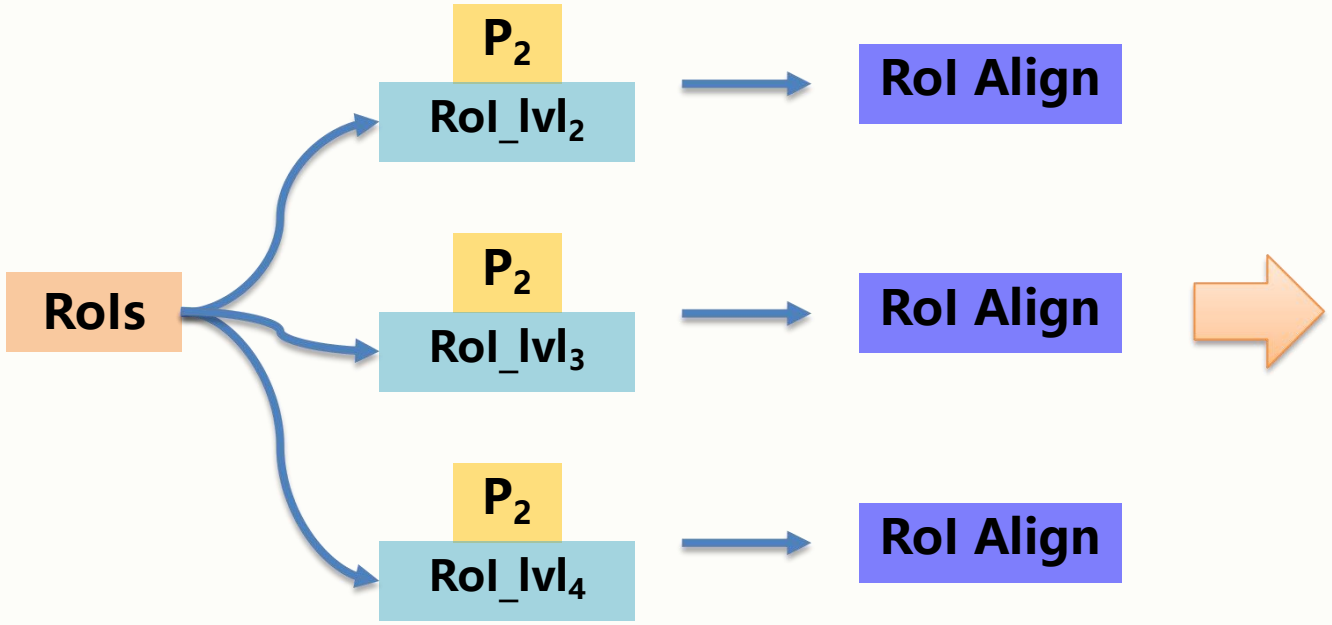


Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection.

4.5 RCNN系列算法的进阶优化

FPN结构下的RoI Align





如何将RoI分配到不同层级？

将FPN的特征金字塔类比为图像金字塔，根据面积来对候选框进行分配。

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$$

- k：分级的层级
- k_0：基准层
- wh：RoI的尺寸

Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection.

4.5 RCNN系列算法的进阶优化

FPN模块的效果

Backbone骨干网络	网络类型	学习率策略	mAP	FPS
ResNet50	Faster	1x	35.2	12.7
ResNet50	Faster	2x	37.1	12.7
ResNet50+FPN	Faster	1x	**37.2**	22.3
ResNet50+FPN	Faster	2x	**37.7**	22.3

包含FPN的检测系统对于小目标的预测要优于非FPN的检测系统

Tsung-YiLin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Belongie. Feature Pyramid Networks for Object Detection. arXiv1612

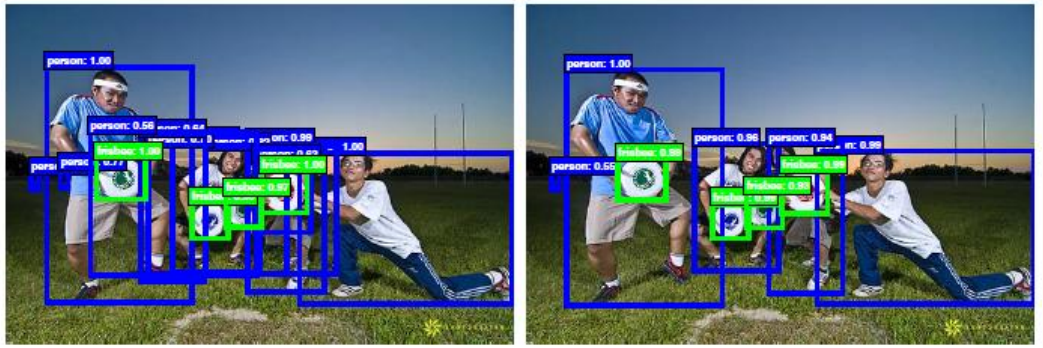
4.5 RCNN系列算法的进阶优化

Cascade R-CNN

4.5 RCNN系列算法的进阶优化

两阶段进阶模型Cascade R-CNN

在目标检测系统中，**IoU**表示两个检测框之间的重叠程度，IoU的取值与RoIs的精确率(Precision)和召回率(Recall)有关。



(a) Detection of $u = 0.5$ (b) Detection of $u = 0.7$

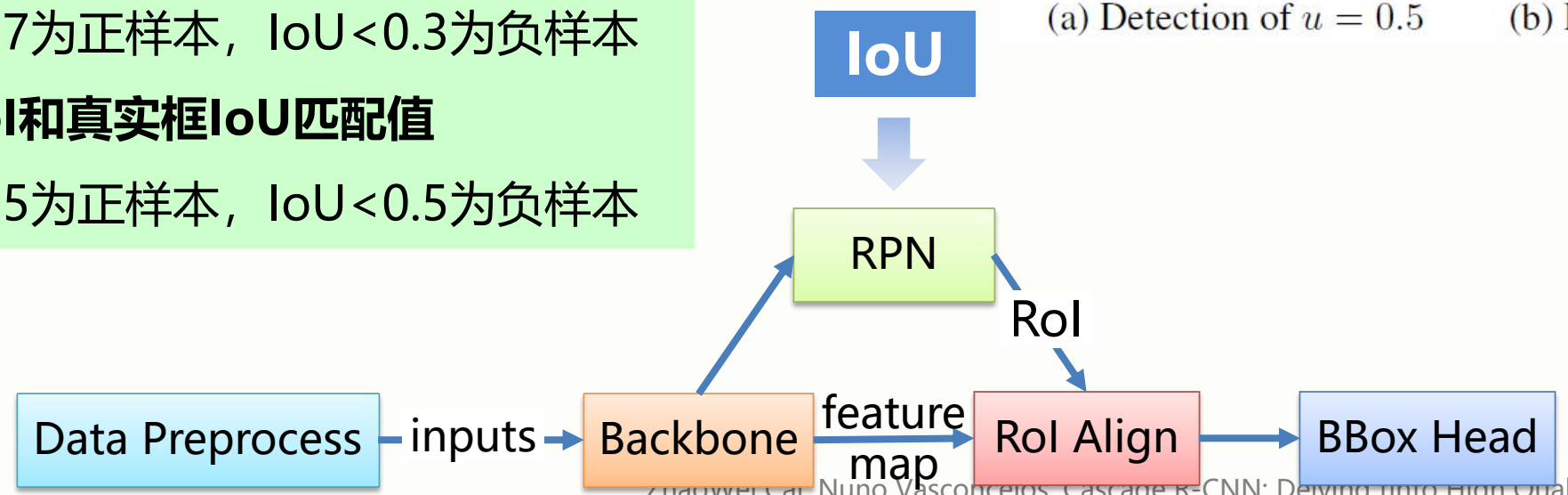
Faster RCNN的正负样本定义规则：

- **基于Anchor和真实框IoU匹配值**

  IoU>0.7为正样本，IoU<0.3为负样本

- **基于RoI和真实框IoU匹配值**

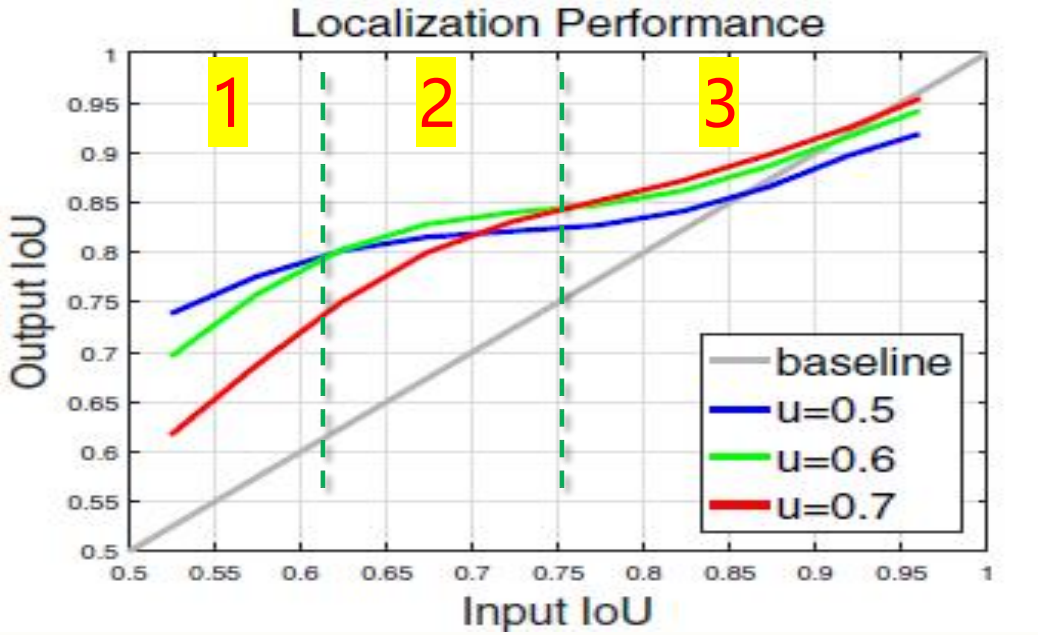
  IoU>0.5为正样本，IoU<0.5为负样本

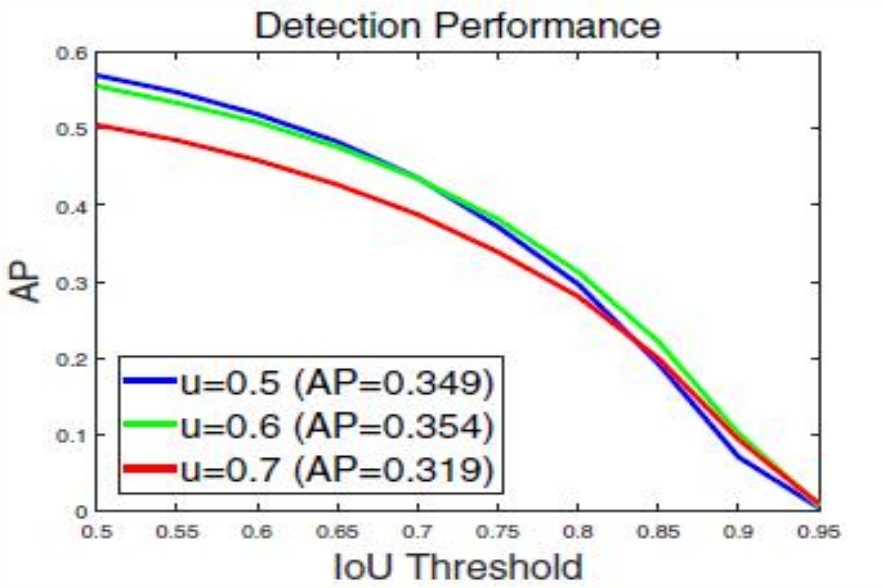


Zhaowei Cai, Nuno Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. CVPR2018

4.5 RCNN系列算法的进阶优化

Cascade R-CNN的IoU分析





单一阈值训练得到的检测器效果非常有限

- ✓ **横轴：** RPN生成的RoI和真值框的IoU
- ✓ **纵轴：** 经过第二阶段BBoxHead得到的新的BBox和真值框的IoU
- ✓ **曲线：** 使用不同IoU阈值筛选出候选框的模型

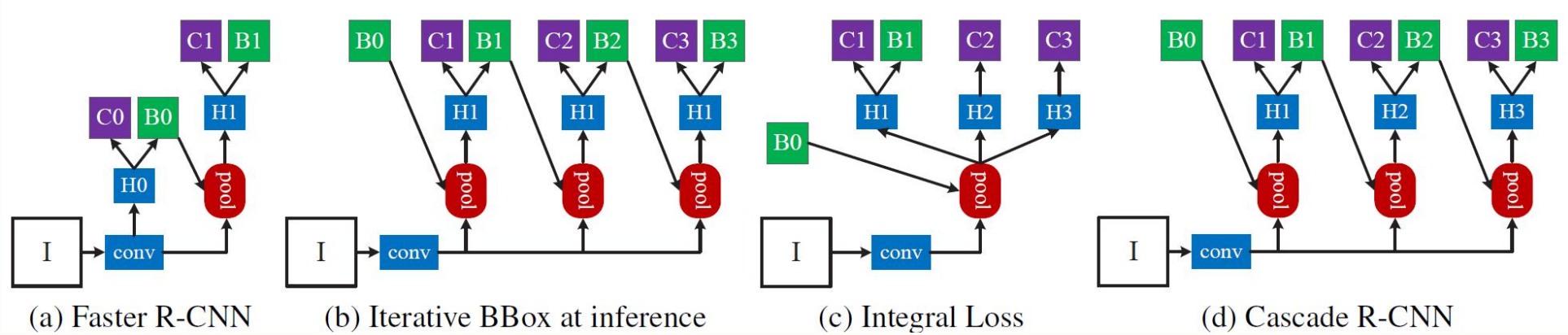
=> 结论：RoI的阈值和训练器训练用的阈值较为接近时，训练器的性能最优。

使用多个检测Head分别对不同阈值的RoI进行调整

ZhaoWei Cai, Nuno Vasconcelos. Cascade R-CNN: Delving tinto High Quality Object Detection. CVPR2018

4.5 RCNN系列算法的进阶优化

Cascade R-CNN的网络结构



(a) Faster R-CNN (b) Iterative BBox at inference (c) Integral Loss (d) Cascade R-CNN

不同的改进方式

- **Cascade R-CNN**

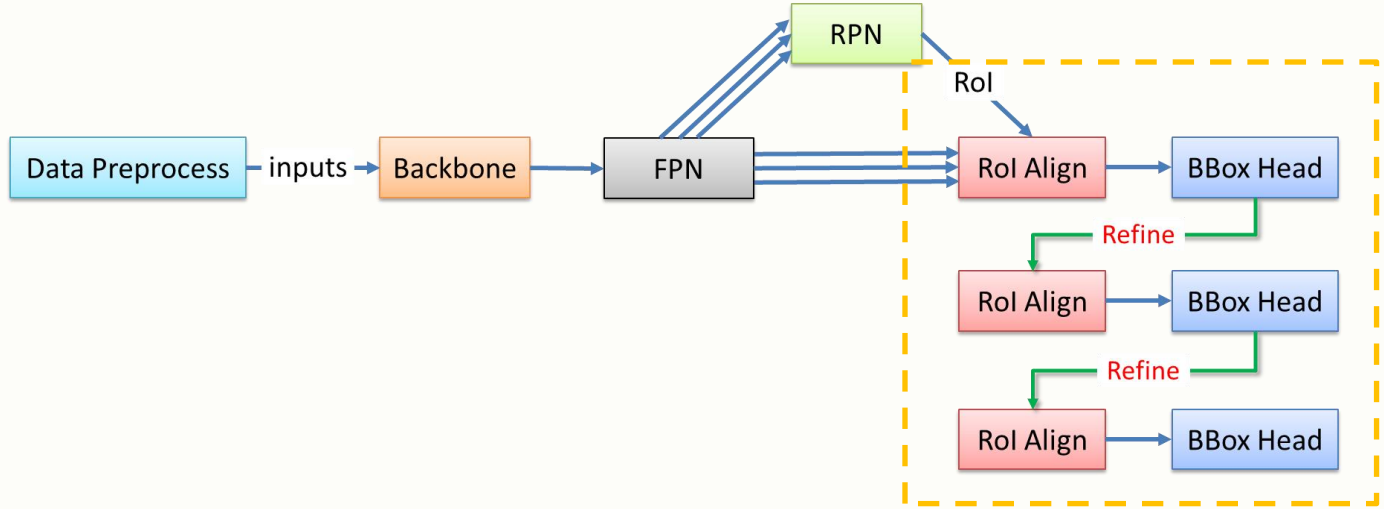
  对box进行三次微调，每次Bbox Head的偏移量和RoI解码被作为下个阶段的RoI输入

- **Iterative Bbox**

  三次微调时的检测头共享权重

- **Integral Loss**

  三次微调并联完成，共用同样的RoI



Cascade Faster RCNN + FPN

ZhaoWei Cai, Nuno Vasconcelos. Cascade R-CNN: Delving tinto High Quality Object Detection. CVPR2018

4.5 RCNN系列算法的进阶优化

Cascade RCNN模型的效果

Backbone骨干网络	网络类型	学习率策略	mAP	FPS
ResNet50	Faster	1x	35.2	12.7
ResNet50	Faster	2x	37.1	12.7
ResNet50+FPN	Faster	1x	37.2	22.3
ResNet50+FPN	Faster	2x	37.7	22.3
ResNet50+FPN	Cascade Faster	1x	**40.9**	17.5

ZhaoWei Cai, Nuno Vasconcelos. Cascade R-CNN: Delving tinto High Quality Object Detection. CVPR2018

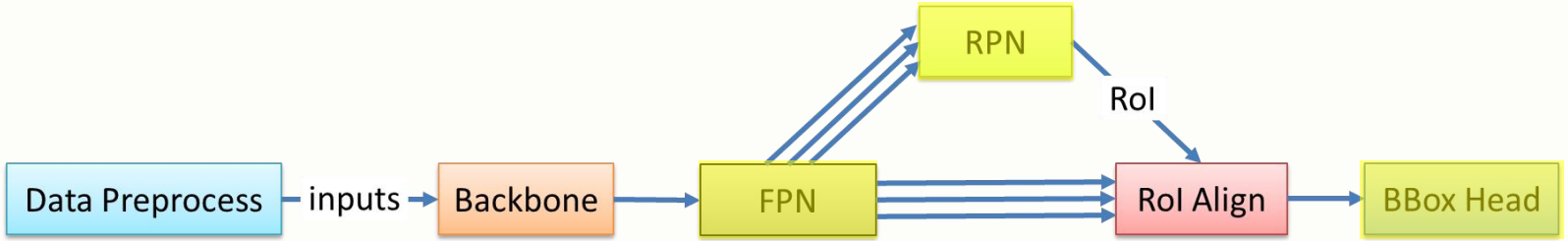
4.5 RCNN系列算法的进阶优化

Libra R-CNN

Jianmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN Towards Balanced Learning for Object Detection. CVPR2019

4.5 RCNN系列算法的进阶优化

Faster RCNN+FPN存在的不平衡问题

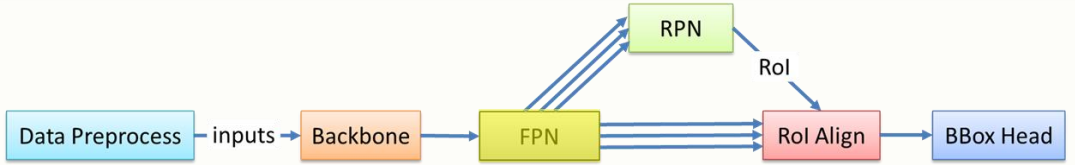


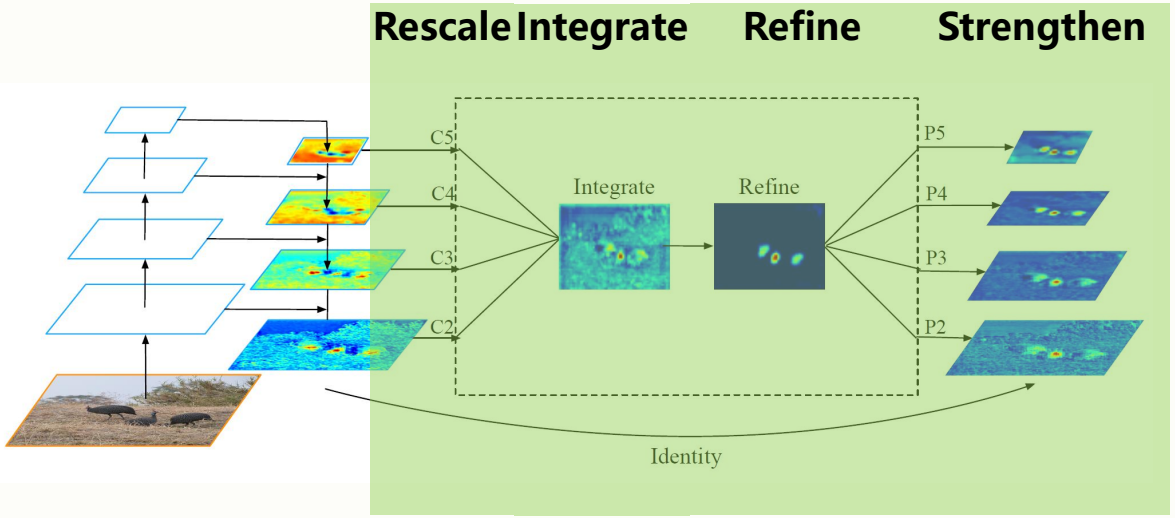
- ✓ **Feature level** => 提取出不同level的特征怎么才能真正地充分利用？ => FPN特征融合
- ✓ **Sample level** => 采样出来的候选区域是否具有代表性？ => RPN中的采样策略
- ✓ **Objective level** => 目前设计的损失函数能不能引导目标检测器更好地收敛 => Loss

Jianmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN Towards Balanced Learning for Object Detection. CVPR2019

4.5 RCNN系列算法的进阶优化

Libra R-CNN的特征融合





FPN特征融合实现了显著性特征的增强

1. **Rescale 尺度归一化**

   将不同层级的特征图通过插值或下采样的方法将特征图的尺度统一到中间层(C4层)的尺度

2. **Integrate 集成**

   将归一化的多个特征图进行融合：$C = \frac{1}{L}\sum_{l=l_{min}}^{l_{max}} C_t$

3. **Refine 精炼化**

   使用non-local结构对融合特征进一步加强

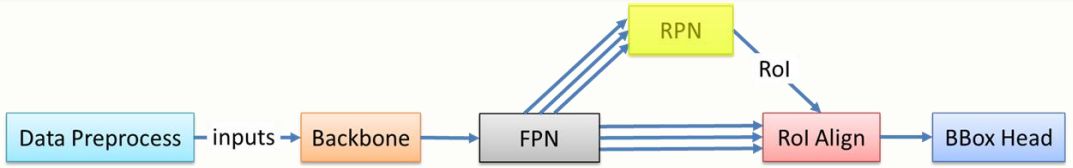
4. **Strengthen 特征加强**

   将优化后的特征与不同层上的原始特征进行加和

Jianmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN Towards Balanced Learning for Object Detection. CVPR2019

4.5 RCNN系列算法的进阶优化

Libra R-CNN的采样策略



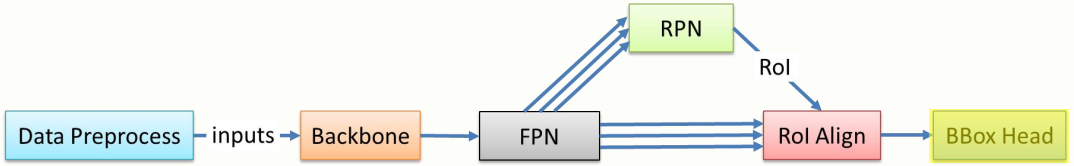
共采样256个样本

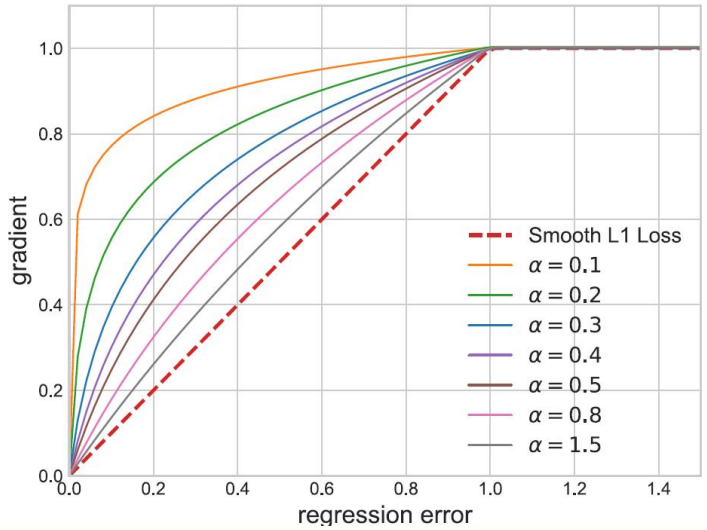
正样本采样	负样本采样
从正样本中随机采样128个	从负样本中随机采样补齐256个
类别不平衡	**IoU分配不平衡**
✓ 计算每个类别需要的采样个数 ✓ 对每个类分别进行采样 ✓ 例：正样本1000个，包含4个类，期望采样128个正样本；那么，每个类随机采样32个正样本	✓ 将负样本根据IoU阈值分为两部分 ✓ 对大于阈值的样本，根据IoU进行分桶，计算应该落在每个桶中的样本数量，并进行平均采样 ✓ 低于阈值的部分属于背景，使用随机采样的方法进行采样

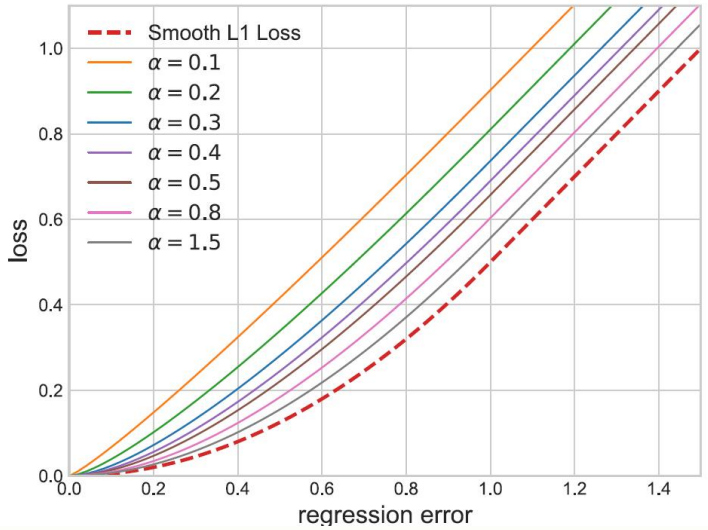
ang, 2019

4.5 RCNN系列算法的进阶优化

Libra R-CNN的回归损失函数







Smooth L1 Loss

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

Balanced L1 Loss

$$L_b(x) = \begin{cases} \frac{a}{b}(b|x| + 1)\ln(b|x| + 1) - \alpha|x| & if\ |x| < 1 \\ \gamma|x| + C & otherwise \end{cases}$$

- Loss较大的样本是困难样本
- Loss较小的样本是容易样本

- Smooth L1中，由于困难样本对应的梯度对于容易样本的梯度更大，导致不同样本学习能力的不平衡
- Balance L1中，困难样本和容易样本界限处的梯度更加平滑

Jianmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN Towards Balanced Learning for Object Detection. CVPR2019

4.5 RCNN系列算法的进阶优化

Libra RCNN模型的效果

Backbone骨干网络	网络类型	学习率策略	mAP	FPS
ResNet50	Faster	1x	35.2	12.7
ResNet50	Faster	2x	37.1	12.7
ResNet50+FPN	Faster	1x	37.2	22.3
ResNet50+FPN	Faster	2x	37.7	22.3
ResNet50+FPN	Cascade Faster	1x	40.9	17.5
ResNet50-vd-BFP	Faster	1x	**40.5**	**18.2**
ResNet101-vd-BFP	Faster	1x	**42.5**	**14.9**

Jianmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN Towards Balanced Learning for Object Detection. CVPR2019

4. 基于两阶段方法的目标检测（区域选择）

【项目032-期末项目】 PCB电路板的缺陷检测

目标：印刷电路板(PCB)的瑕疵检测

PCB数据集是一个公共的合成PCB数据集，由北京大学发布，用于检测、分类和配准任务。其中包含1368张图像以及6中缺陷（缺失孔，鼠标咬伤，开路，短路，杂散，伪铜）。本项目选择了适用于检测任务的693张图像，随机选择593张图像作为训练集，100张图像作为验证集。

作业描述：

基于PaddleDetection工具包中的算法，完成印刷电路板(PCB)瑕疵数据集的训练与评估任务，要求在验证集上$mAP_{0.5}$达到0.7以上。

Part 05

基于单阶段方法的目标检测

/ Yolo v1, v2, v3

/ SSD

/ RetinaNet

/ PP-Yolo

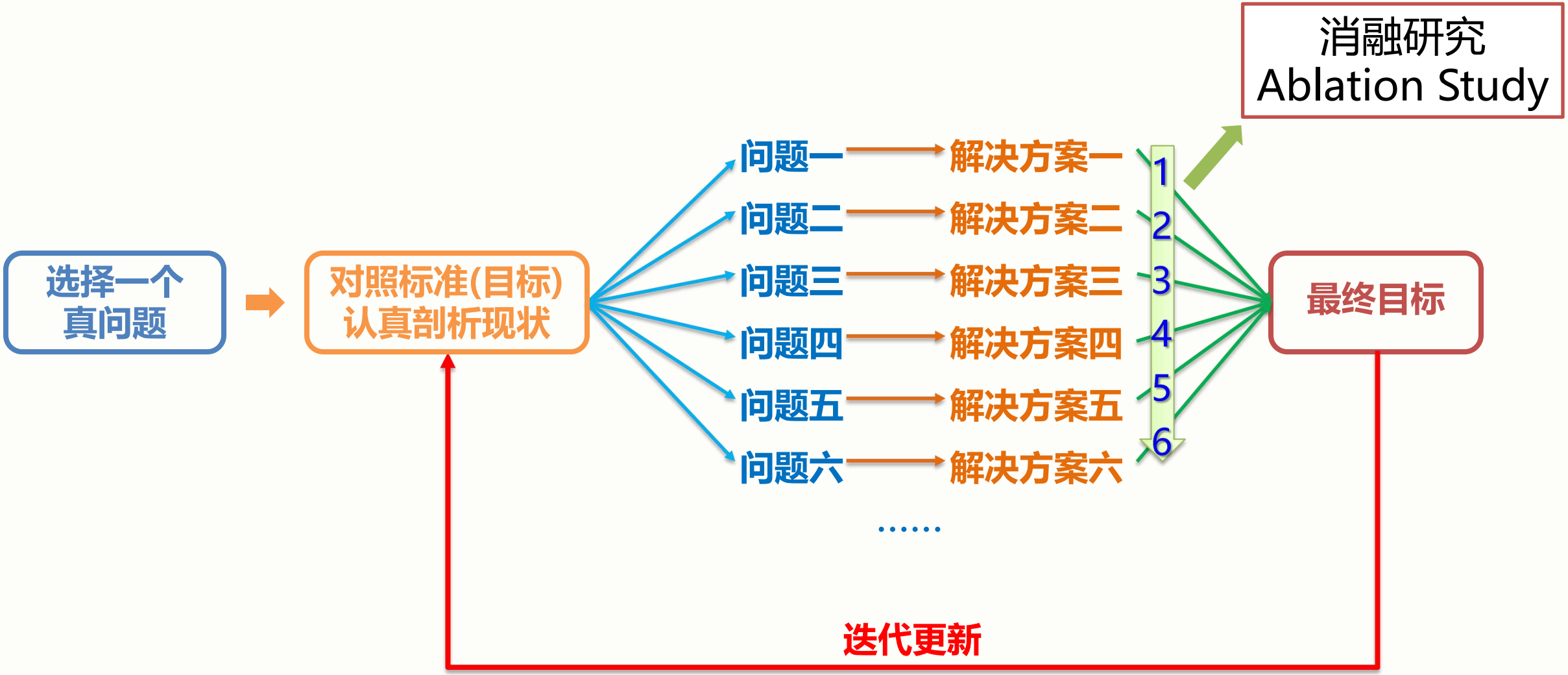
5. 基于单阶段方法的目标检测

如何持续地进行创新?

如何对自我/项目进行持续地提高?

5. 基于单阶段方法的目标检测

如何持续地进行创新?



5. 基于单阶段方法的目标检测

如何持续地进行创新?

Case1: 成神之路

成就神位

Who Am I

武魂殿长老

- 缺少神格？神位？ → 去海神岛参加海神九考
- 缺少感悟？ → 去各片大陆游历

→ 超脱入神界

新手村战五渣

- 等级低 → 去野外刷怪
- 缺装备 → 去刷各种副本
- 没宠物 → 去星斗大森林抓
- 没钱买丹药 → 发展第二职业赚钱
- 个人刷副本太难 → 加入史莱克学院
- 缺少神格？神位？ → 去海神岛参加海神九考
- 缺少感悟？ → 去各片大陆游历

→ 超脱入神界

定义真问题　剖析现状　提出问题，解决问题　实现最终目标

5. 基于单阶段方法的目标检测

如何持续地进行创新?

Case2: 晋升之路

定义真问题	剖析现状	提出问题	解决问题	实现最终目标
我要评教授 我要当工程师 我要进BATH ……	对照国家、省、单位的职称/晋升/招聘/评定文件	20万字专著1部	30万字，2部	横扫竞争者
		SCI论文3篇	SCI论文5篇	
		主持省级基金2项	国家基金，省级3项	
		省级教学成果奖1项	国家级，省级2项	
		申请知识产权3项	专利2个，软著 3个	
		担任辅导员1年	担任1届	
		企业挂职6个月	挂职1年	
		竞赛获省级一等奖1项	国家级，省级2项	
		项目转化1项（20万）	转化2项，40万	
		……	逐渐迭代完成	

5. 基于单阶段方法的目标检测

如何持续地进行创新?

Case3: 从YOLOv3到PP-YOLO

定义真问题

同时提升目标检测的性能和效率

剖析现状

YOLO v3

特征鲁棒性
Label过硬
难样本
遮挡
一个网格一个对象
后处理冗余评估
对象尺度多样化
网络初始化
收敛性

提出问题

解决问题

序号	模型	AP^{val}	AP^{test}	FPS (V100)
-	YOLOv3(darknet)	33.0	-	-
A	YOLOv3(Ours)	38.9	-	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	-	79.2
C	B + LB + EMA + DropBlock	41.4	-	79.2
D	C + IoU Loss	41.9	-	79.2
E	D + IoU Aware	42.5	-	74.9
F	E + Grid Sensitive	42.8	-	74.8
G	F + Matrix NMS	43.5	-	74.8
H	G + Coord Conv	44.0	-	74.1
I	H + SPP	44.3	45.2	72.9
J	I + Better ImageNet Pretrain	44.8	45.2	72.9
K	J + 2x Scheduler	45.3	45.9	72.9

消融研究

实现最终目标

PP-YOLO (2020)

迭代更新

PP-YOLOv2 (2021)

To Be Continued

5. 基于单阶段方法的目标检测

Joseph Redmon
YOLO之父

5.1 **YOLO (2015)**

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
University of Washington

5.2 **SSD (2016)**

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, ChengYang Fu, Alexander C.Bery

5.3 **YOLOv2 (2016)**

Joseph Redmon, Ali Farhadi
University of Washington

5.4 **RetinaNet (Focal Loss, 2018)**

TsungYi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar
Facebook AI Reserach

5.5 **YOLOv3 (2018)**

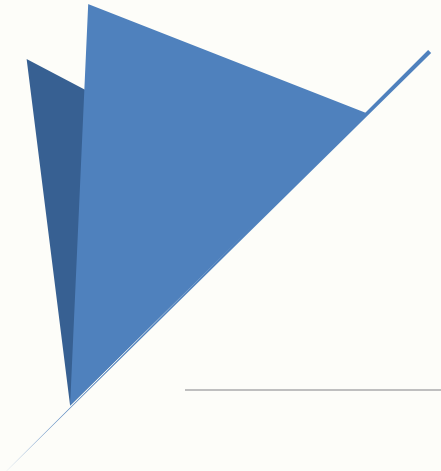
Joseph Redmon, Ali Farhadi
University of Washington

5.6 **YOLOv4 (2020)**

Alexey Bochkovskiy, ChienYTao Wang, HongYuan Mark Liao

5.7 **PP-YOLO (2020)**

Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, Shilei Wen



YOLO

5.1 YOLO

设计动机

- ***Motivation***

Faster R-CNN系列算法基于**区域建议算法**构建了一个两阶段（two-stage）的目标检测系统，此类算法准确度较高，但执行速度较慢，很难达到实时检测的要求。另一类系统基于**回归分析**生成预测，该类算法仅使用一个CNN网络直接预测不同目标的类别与位置，属于一阶段（one-stage）算法。该类算法速度快，但准确性相对低一些。典型的算法包括：YOLO系列、SSD、RetinaNet、RefineDet、RFB-Net等。

- ***YOLO***

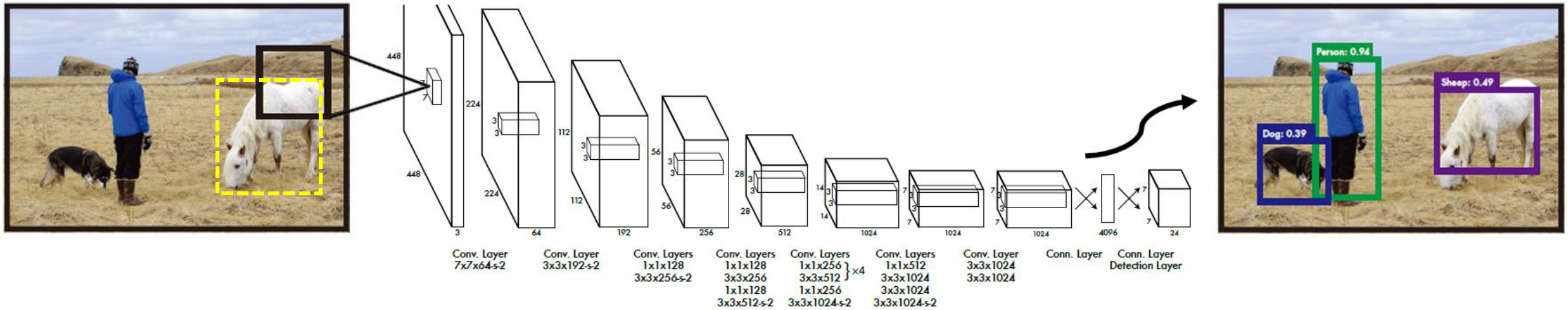
YOLO又称为You Only Look Once的简称，是第一个基于卷积神经网络的单阶段目标检测系统。它速度非常快，在TitanX上可以实现每秒45张图片的处理速度，极速版可以达到150FPS。YOLO结构简单，可以端到端实现类别识别和边界框回归预测。

Redmon et al, “You Only Look Once:Unified, Real-Time Object Detection”, CVPR 2016

5.1 YOLO

YOLO开山之作——YOLOv1

将目标检测当作一个单一的回归任务，直接预测目标框的坐标位置



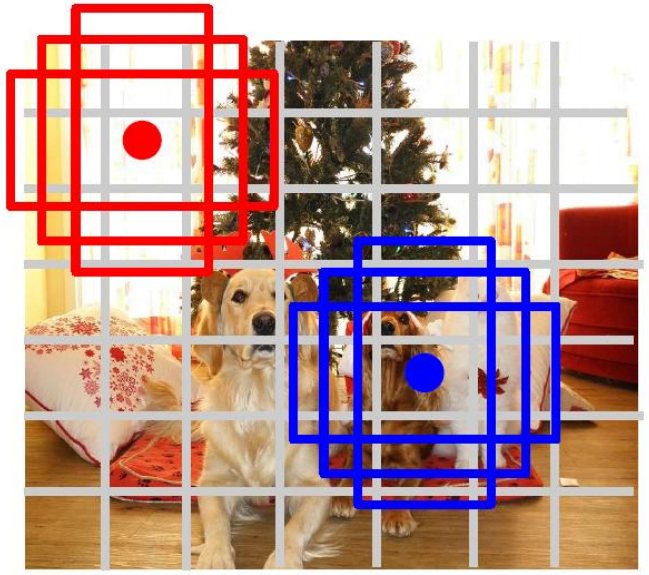
1. **缩放图像**：将图像缩放至448×448并送入CNN网络
2. **预测边界框**：将图像分割为S×S的网格，每个网格对应B个边界框，预测所有网格对应的边界框位置，并对网格中的内容进行分类
3. **非极大抑制**：执行NMS算法去除冗余边界框

Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016

5.1 YOLO

YOLO核心原理

输入图像
3×H×W



将图像划分为7×7的网格

想象一组以每个网格为中心的B个**基本区域**
此处，B = 3

如果一个目标的中心落在一个网格单元中，则该网格负责检测该目标。

对于每个网格单元：

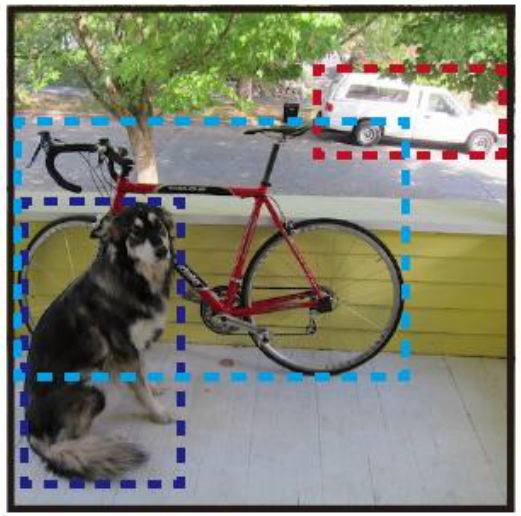
- 从每B个**基本区域**回归到最终的区域，每个区域包含5个参数(x, y, h, w, 置信度)
- 对每个区域都给出针对C+1个类的预测分数（包括背景）
- 类似RPN，但类别相关

输出：H×W×(5×B+C) 维
= 7×7×(5×3+21) = 1764 (Pascal VOC)
= 7×7×(5×3+81) = 4704 (MSCOCO)

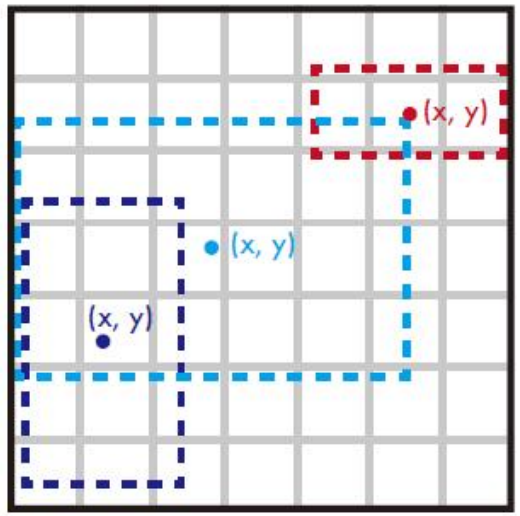
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

5.1 YOLO

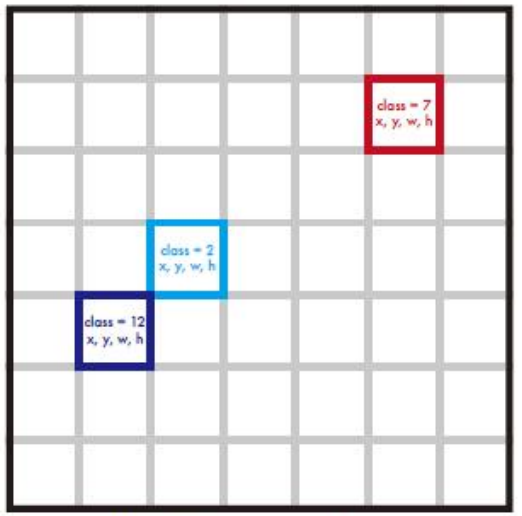
YOLO核心原理



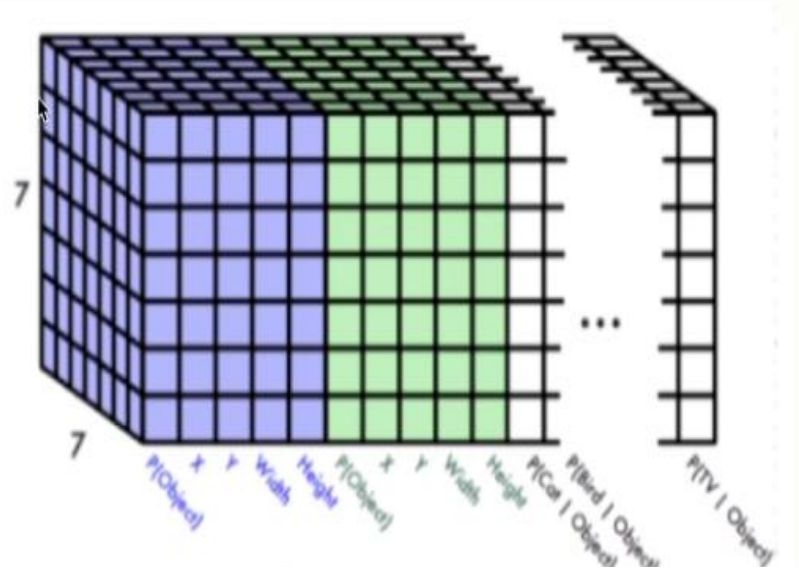
Resize The Image
And bounding boxes to 448 x 448.



Divide The Image
Into a 7 x 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.



1th-5th 1# Box　6th-10th 2# Box　11th-15th 3# Box　16th-35th 类别概率

由于每个网格都只有一个类别向量C，所以无法在同一个网格中检测出多个不同的目标。因此对于位置比较接近的目标，检测效果较差。

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

5.1 YOLO

YOLO的损失函数

- **边界框置信度**，表示边界框包含目标且边界框位置准确的概率，表示为：

$$C_i = P_{object} \times IoU_{pred}^{gt}$$

- **边界框质量：**对于每个网格，虽然包含多个边界框，但类别都是一样的，因此边界框的质量等于分类概率和边界框置信度的乘积：

$$P_b = P(Class_i|Object) \times P_{object} \times IoU_{pred}^{gt} = P(Class_i) \times IoU_{pred}^{gt}$$

- ***最终的损失函数：***

$$L_{YOLO} = \lambda_{coord} L_{coord} + L_{conf} + L_{cls}$$

边界框坐标损失　边界框置信度损失　分类损失

Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016

5.1 YOLO

YOLO优缺点分析

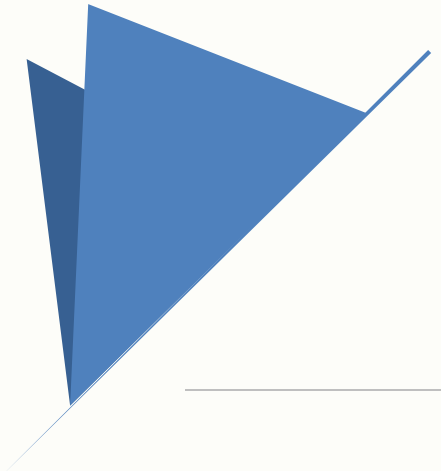
优点 *Advantage*

1. 首次将目标检测转化为端到端问题，检测速度很快（但仍未达到实时要求）；且单级系统训练方便，不需要分部训练；
2. 预测时直接输入图像，不需要前置的滑动窗空或区域候选方法；
3. 泛化能力强

缺点 *Disadvantage*

1. 精度低于两级检测系统（Faster R-CNN）
2. 容易产生定位误差
3. 对小目标检测效果不好
4. 一个网格只能检测出一个样本

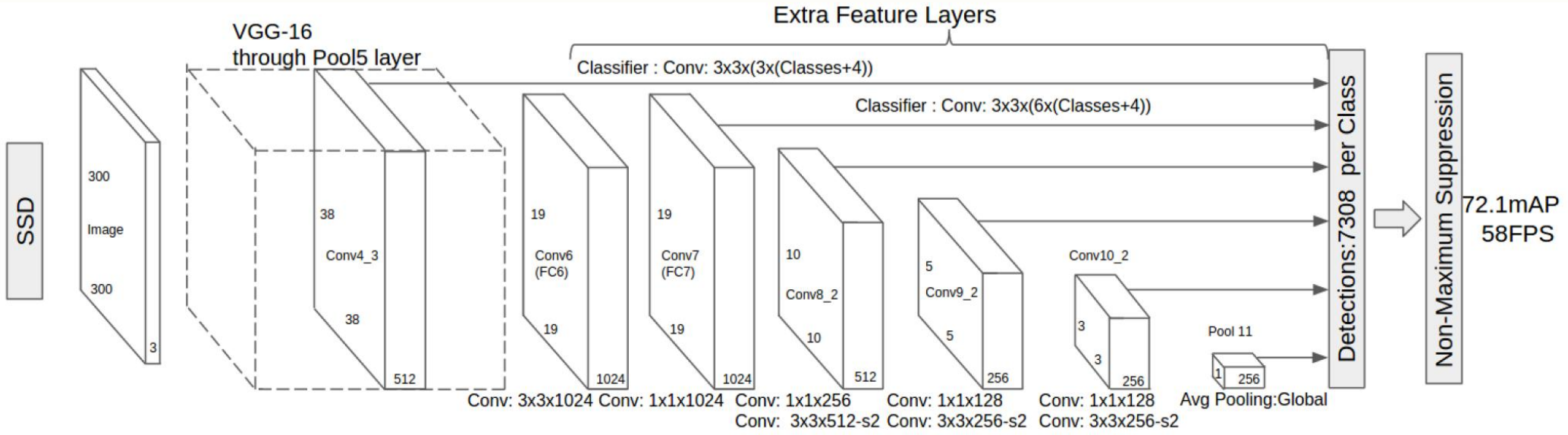
Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016



SSD

5.2 SSD

SSD模型体系结构



1. **缩放图像**：将图像缩放至 300×300(500×500) 并送入CNN网络
2. **特征提取**：使用改进版VGG16提取多尺度卷积特征 (conv4~pool11)
3. **预测边界框**：在不同尺度的特征图的每个像素上都设置一组不同形状的anchors，并对每个anchors进行分类和边界框回归
4. **非极大抑制**：执行NMS算法去除冗余边界框

5.2 SSD

SSD的损失函数

分类置信度

GroundTruth

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right)$$

预测框坐标

分类损失

定位损失

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} smooth_{L1}\left(l_i^m - \hat{g}_j^m\right)$$

第i个预测框与第j个真实框关于类别k是否匹配：1 or 0

$$L_{conf}(x, c) = \sum_{i \in Pos}^{N} x_{ij}^{p} \log \hat{c}_i^p - \sum_{i \in Neg} \log \hat{c}_i^p$$

当预测框不包含目标时，背景类概率越高，损失越小

概率通过Softmax生成

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

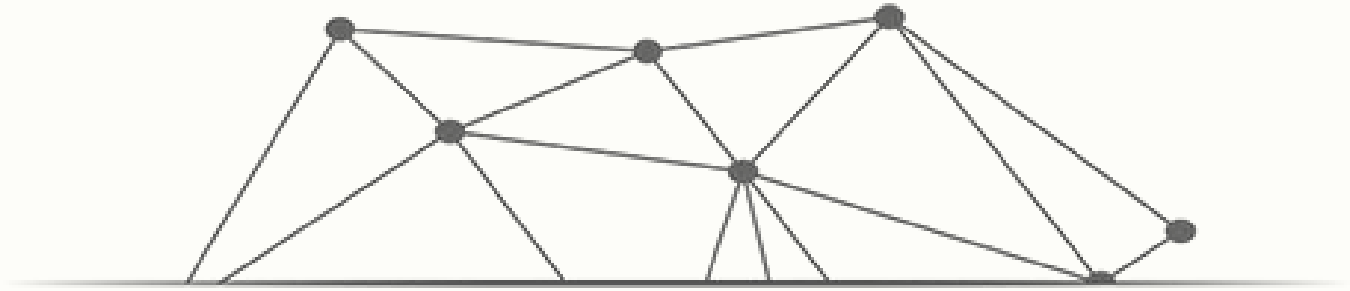
预测框i与真实框j关于类别p的匹配损失，预测p的概率越高，损失值越小

5.2 SSD

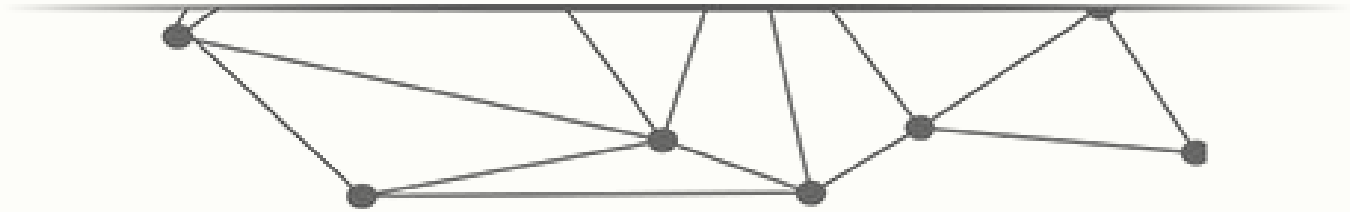
SSD的创新

1. 使用全卷积结构替代YOLOv1的全连接层，速度进一步提升；
2. 使用多尺度特征提取，使用浅层特征检测小物体，使用深层特征检测大物体，对不同尺寸的目标检测鲁棒性更强；
3. 学习RPN的anchor思想，在特征图上预设不同尺度的Anchor；学习R-CNN的思想，将回归坐标改为回归坐标的偏移。缓解YOLOv1的定位误差，从而提高边界框回归的效率；
4. 对负样本anchor进行**难样本挖掘 (Hard negative mining)**，缓解正副样本不均衡的问题。

Liu wei, et. al. SSD: Single shot multibox detector

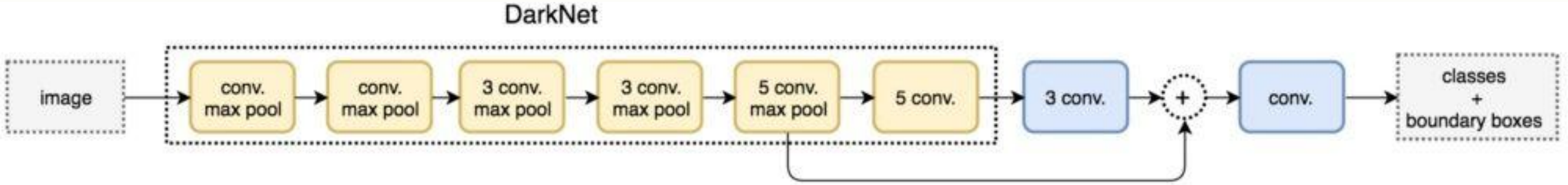


5.3 YOLO v2



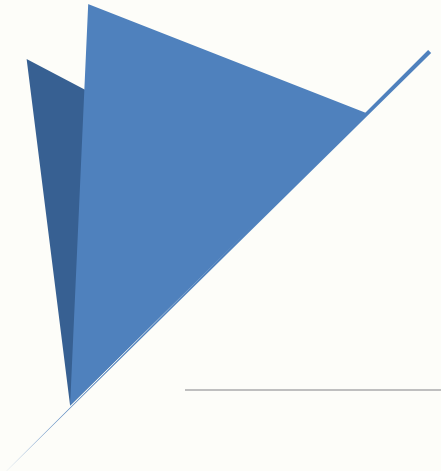
5.3 YOLOv2

YOLOv2 体系结构及创新点分析



1. **改进主干网络**，命名为Darknet19，包含19个卷积层，5个最大池化层，去掉全连接层，使用全卷积网络结构：Conv + Batch Normalization；
2. 学习RPN的anchor思想，在特征图上预设不同尺度的Anchor，但Anchor使用Kmeans聚类算法从coco，voc上生成（Faster 和SSD均是人为经验设置）；
3. 学习R-CNN的思想，将边界框约束参考从cell的中心改为cell的左上角；
4. 相同cell中的每个anchor都使用独立的类别（YOLOv1中一个cell共享同一个类别）；
5. **多尺度训练**，样本从scale={320,352,...,608}中随机选择不同尺度图片进行训练。

Redmon et al, "Yolo 9000 better faster stronger", CVPR 2016



RetinaNet

5.4 RetinaNet

目标与动机

小目标识别困难 → 特征金字塔网络

类别严重不平衡 → Focal Loss → 正负样本不均衡

Focal Loss → 难易样本不均衡

Tsung-Yi Lin, Priya Goyal, Ross Girshik, Kaiming He, Piotr Dollar. Focal Loss for Dense Object Detection. arXiv 2018

5.4 RetinaNet

Focal Loss

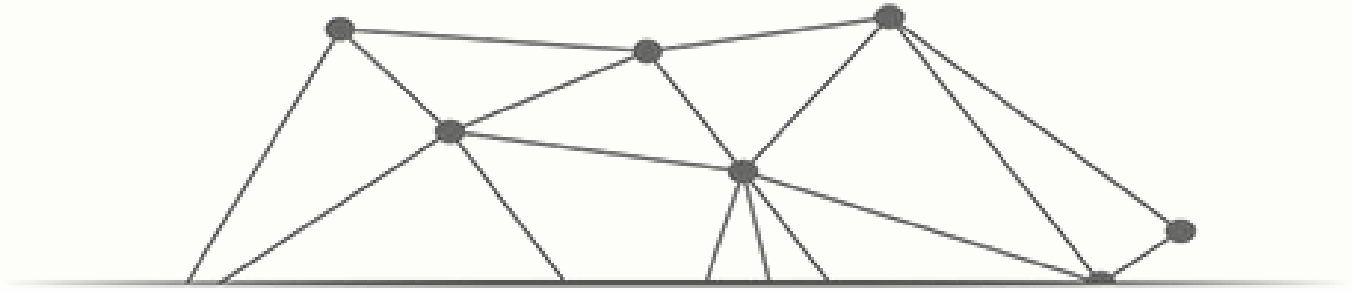
调节正负样本重要性

交叉熵

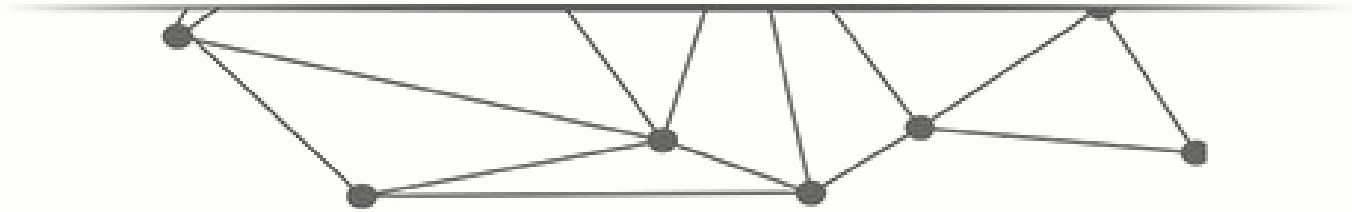
$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

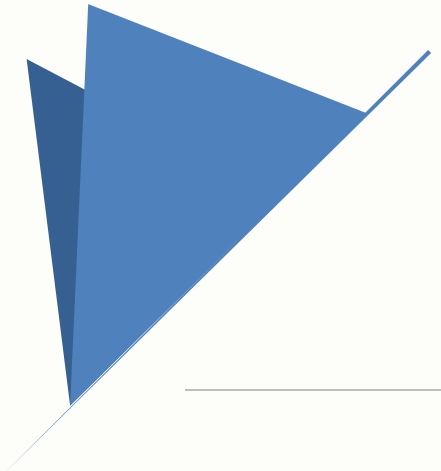
调整难易样本的重要性

Tsung-Yi Lin, Priya Goyal, Ross Girshik, Kaiming He, Piotr Dollar. Focal Loss for Dense Object Detection. arXiv 2018



课堂互动 13.2.8



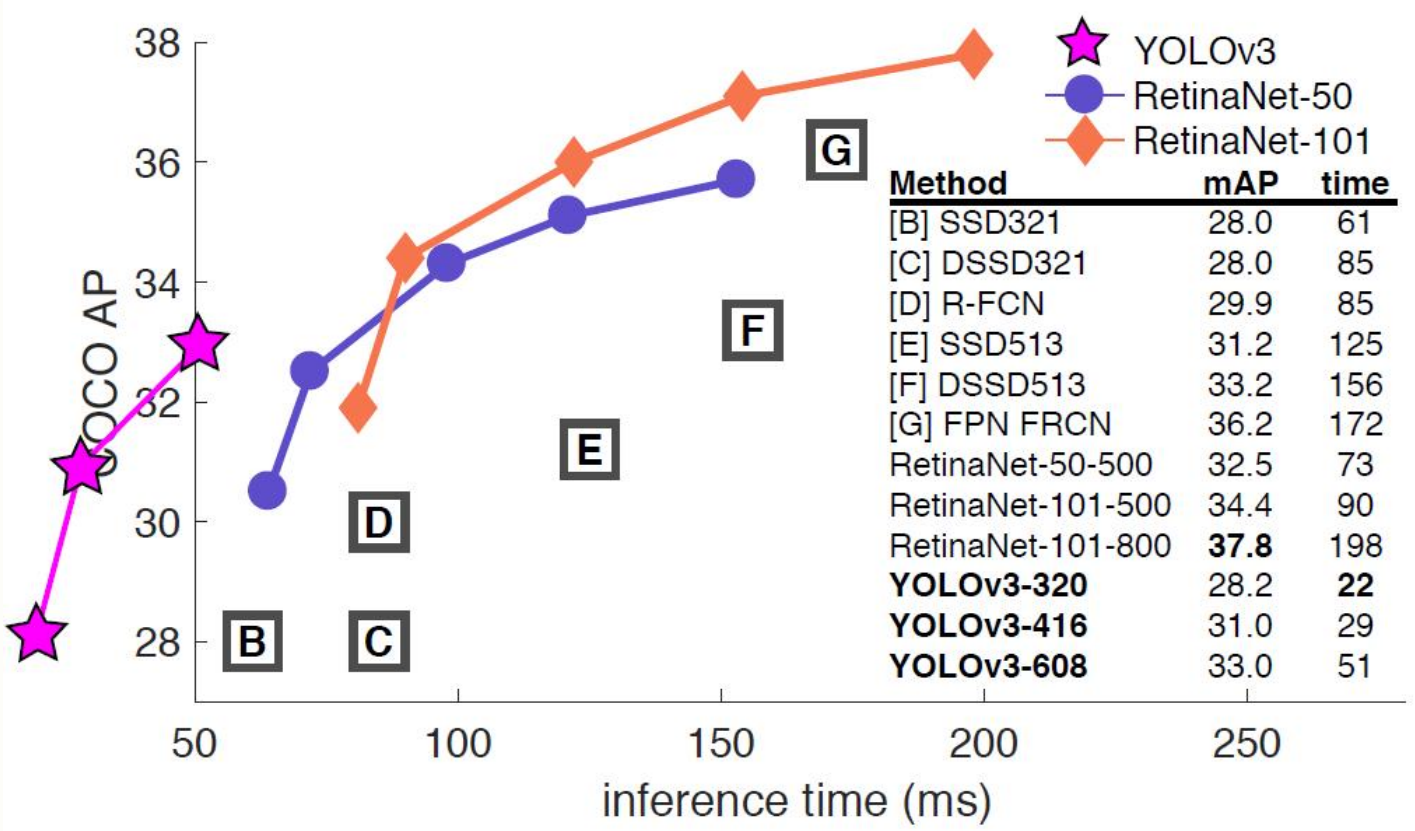


YOLO v3

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

YOLOv3的优缺点



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

优点

- ✓ 推理速度快，性价比高（推理速度快于SSD**3倍**，快于RetinaNet**3.8倍**）
- ✓ 背景误检率低
- ✓ 通用性强

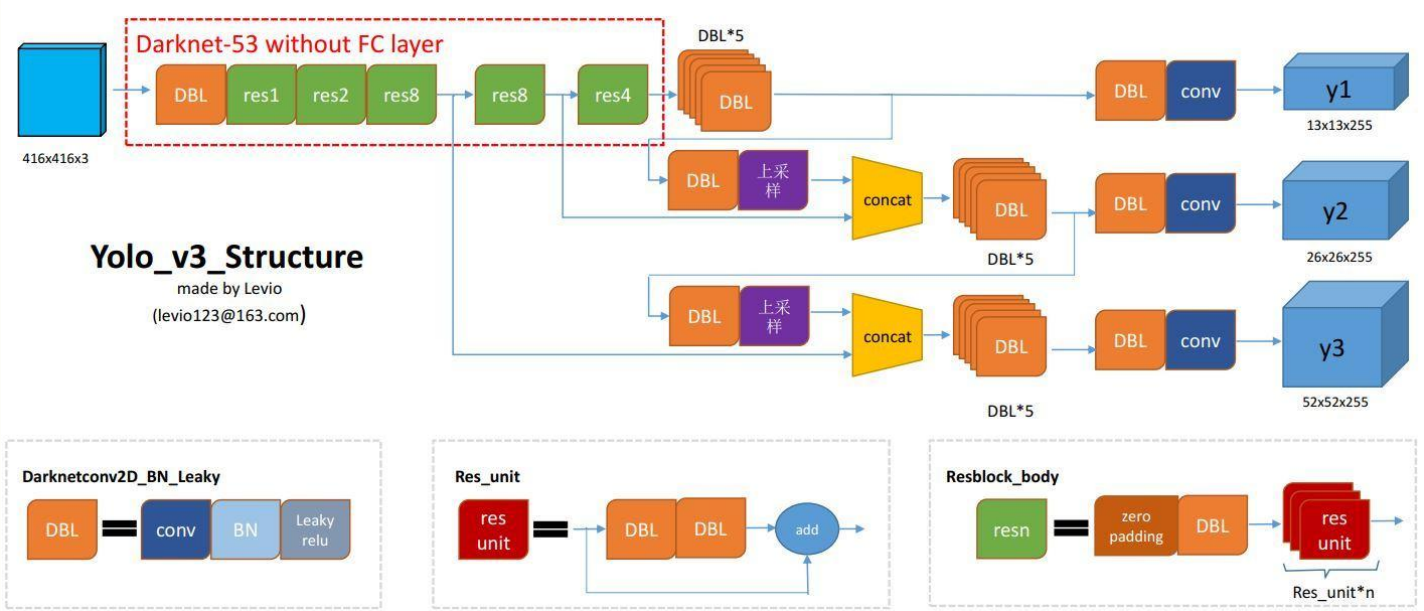
缺点

- ✓ 召回率依然较低
- ✓ 定位精度仍需改进
- ✓ 靠近/遮挡的群体、小物体检测效果差

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

YOLOv3的主要改进



YOLOv3网络体系结构

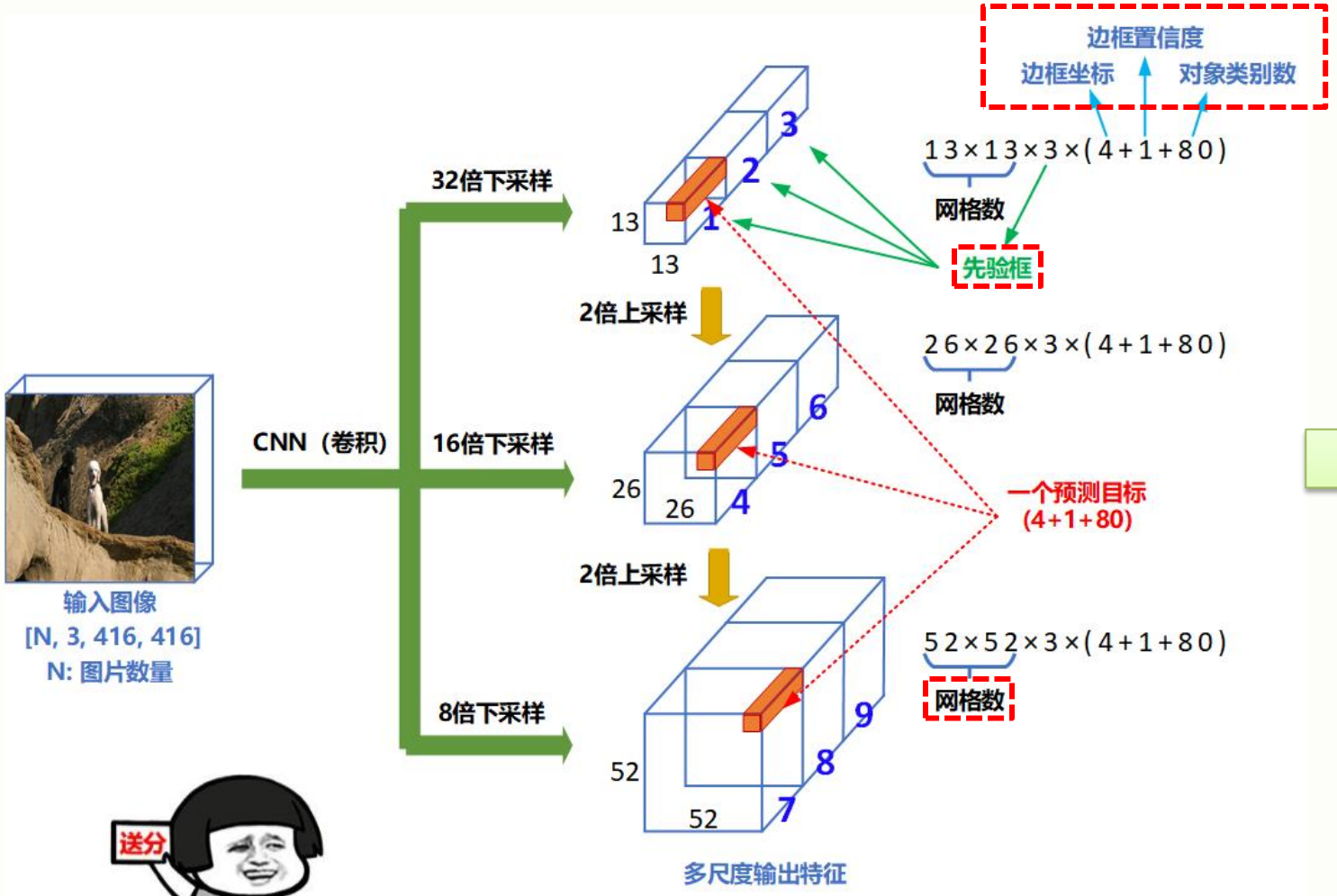
主要改进

- ✓ 精心设计的主干网络Darknet53
- ✓ 空间金字塔网络=>多尺度检测
- ✓ 跨尺度特征融合
- ✓ Coco数据集聚类9种不同尺度anchor
- ✓ Objectness对象性预测
- ✓ 分类使用sigmoid激活，支持单一目标的多分类体检测效果差

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

YOLOv3的输入输出



如果边框坐标和对象类别是目标检测的任务；那么，神马是**边框置信度**？

先验框是个啥？和锚框（Anchor）有啥关系？为神马要有先验框？

网络输出

[N, 255, 13, 13]
[N, 255, 26, 26]
[N, 255, 52, 52]

网格数又是什么？

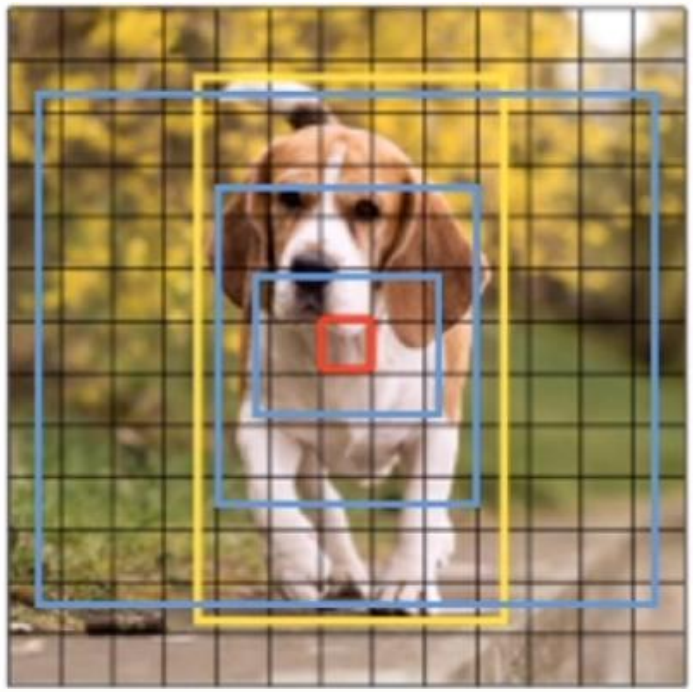
这是一道送分题

为什么要有三个输出分支？

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

基于先验的锚框



特征图	13×13			26×26			52×52		
感受野	大			中			小		
先验锚框	(373×326)	(156×198)	(116×90)	(59×119)	(62×45)	(30×61)	(33×23)	(16×30)	(10×13)

先验锚框由Kmeans算法在coco数据集的真实框上聚类获得，共生成9种anchor。

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

基于先验的锚框

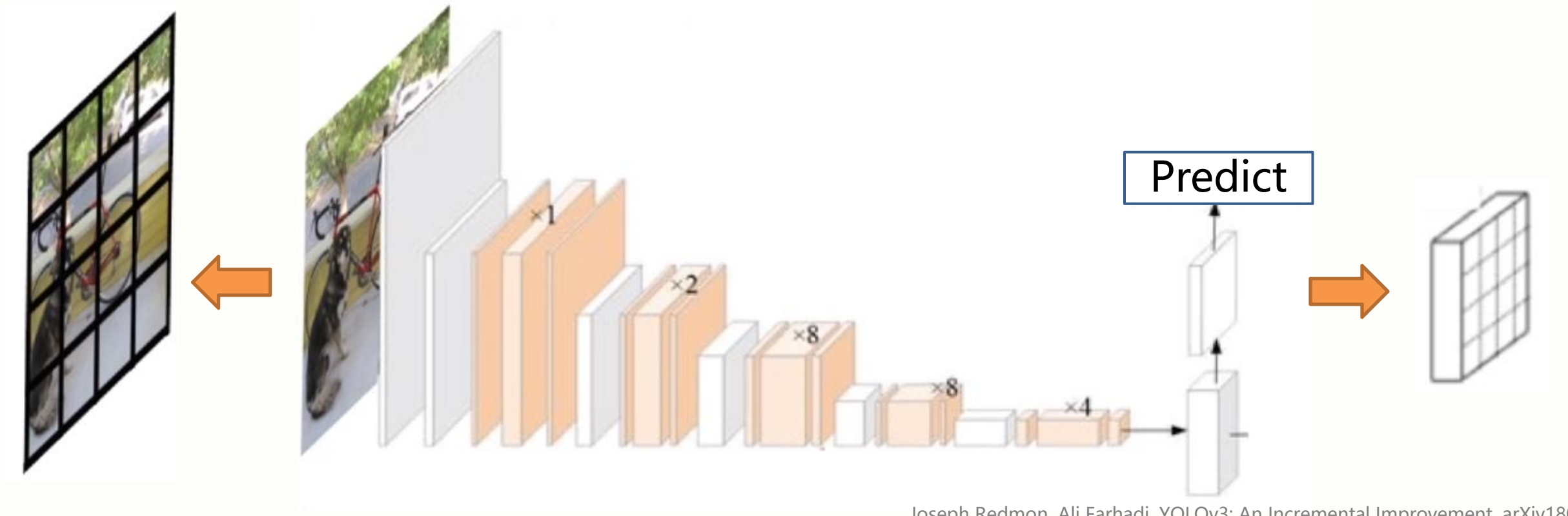
- **先验锚框**只给出了框的高度和宽度，那如何去表示图像中物体的真实框？
- 当样本中存在多个真实框（目标）时，如何对它们进行表示？
- 如何在图像中对真实框进行定位？
- **先验锚框**的高度和宽度是固定的，如何去适配物体各种不同尺度的高度和宽度？

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

从先验锚框到预测框

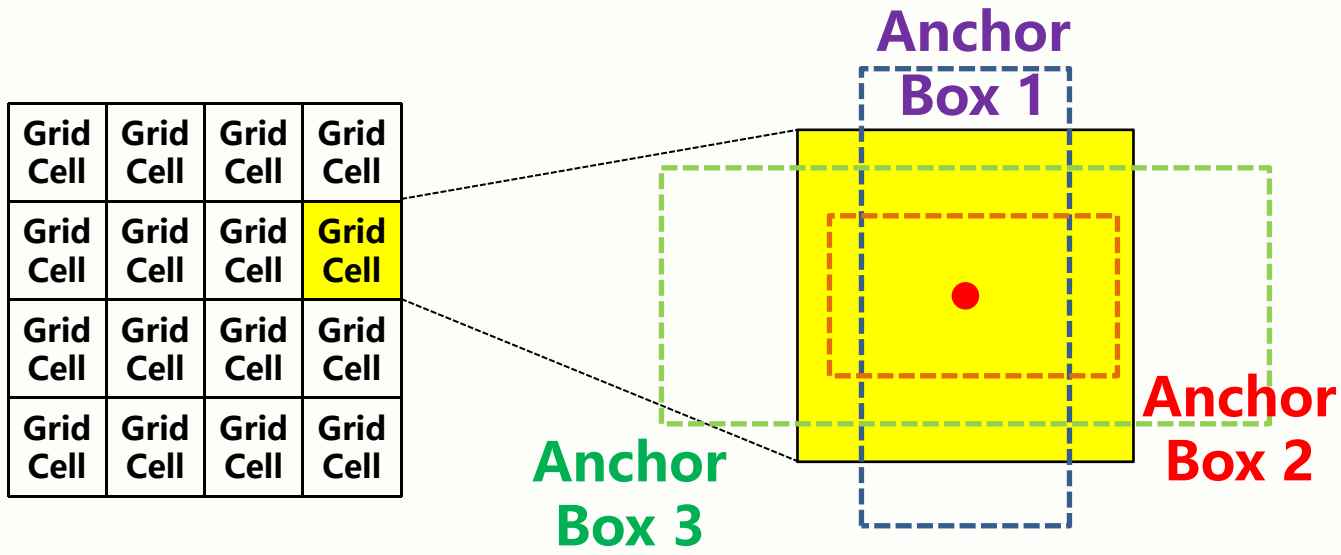
- **输出特征图**的宽度和高度分别为W、 H， 相当于将图像划分为W×H个网格
- 图像的每个网格对应于**输出特征图WH平面**上的一个像素点



Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

从先验锚框到预测框



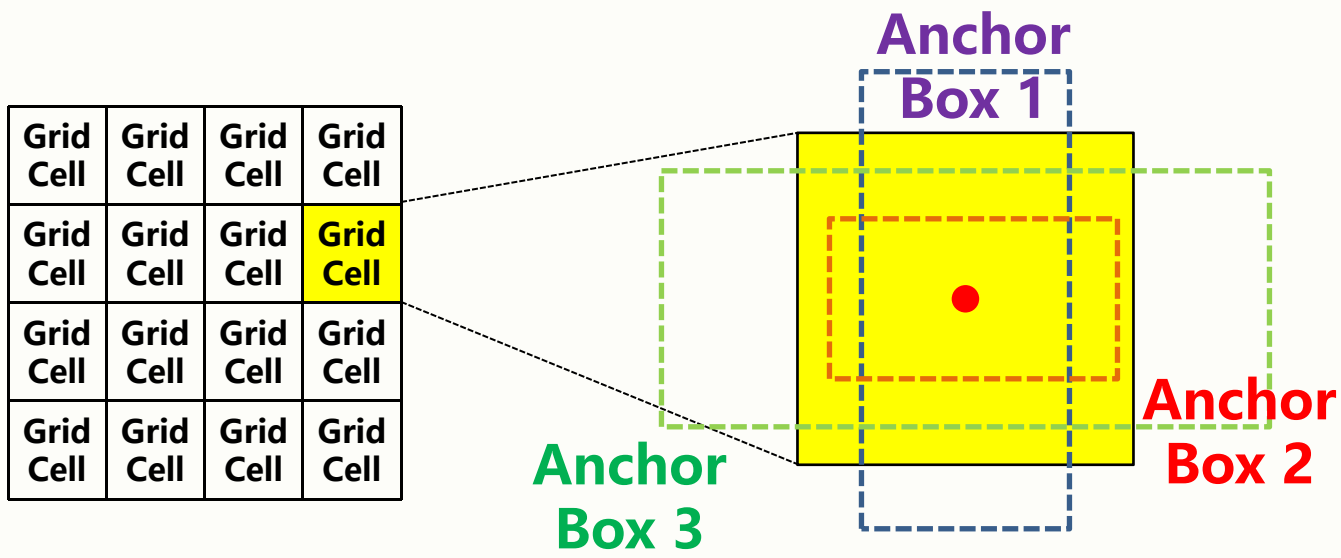
- 每个网格上都有一组先验锚框
- 每个先验锚框都对应一个预测框
- 预测框数 = 网格数×先验锚框数

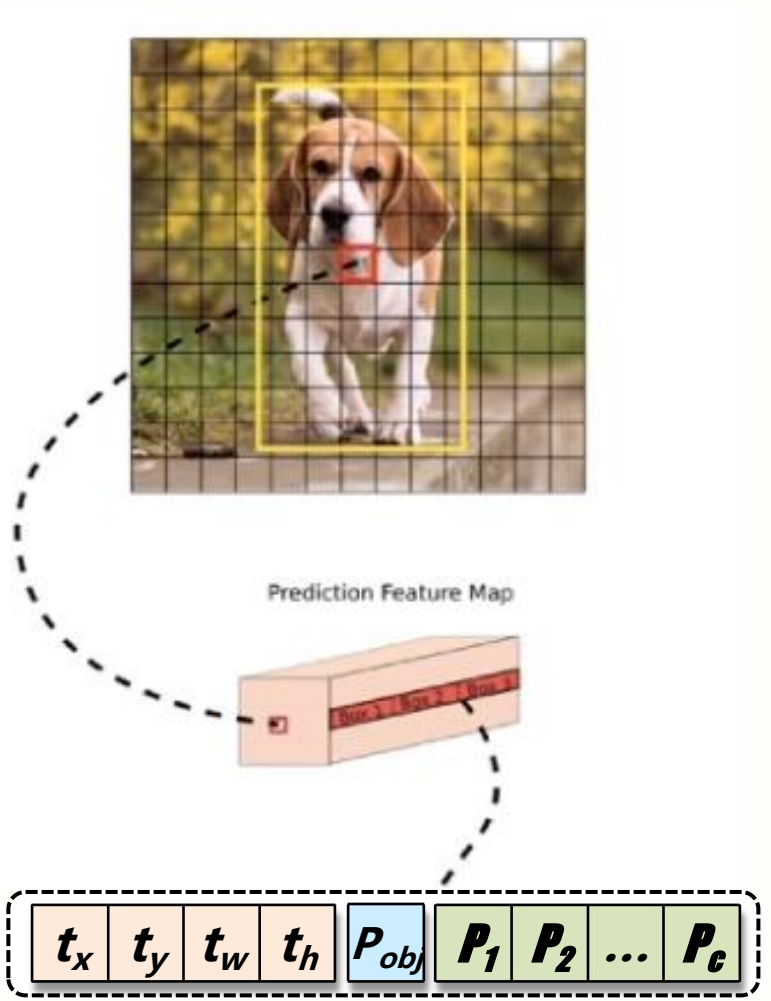
“框”海战术

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

从先验锚框到预测框





- 每个预测框都包含**三组**信息：
  - 坐标： t_x, t_y, t_w, t_h
  - 坐标置信度： p_{obj}
  - 各分类置信度： p_c
- 特征向量C = B×(5+class_num)，B为特征图上先验锚框的数量

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

从先验锚框到预测框

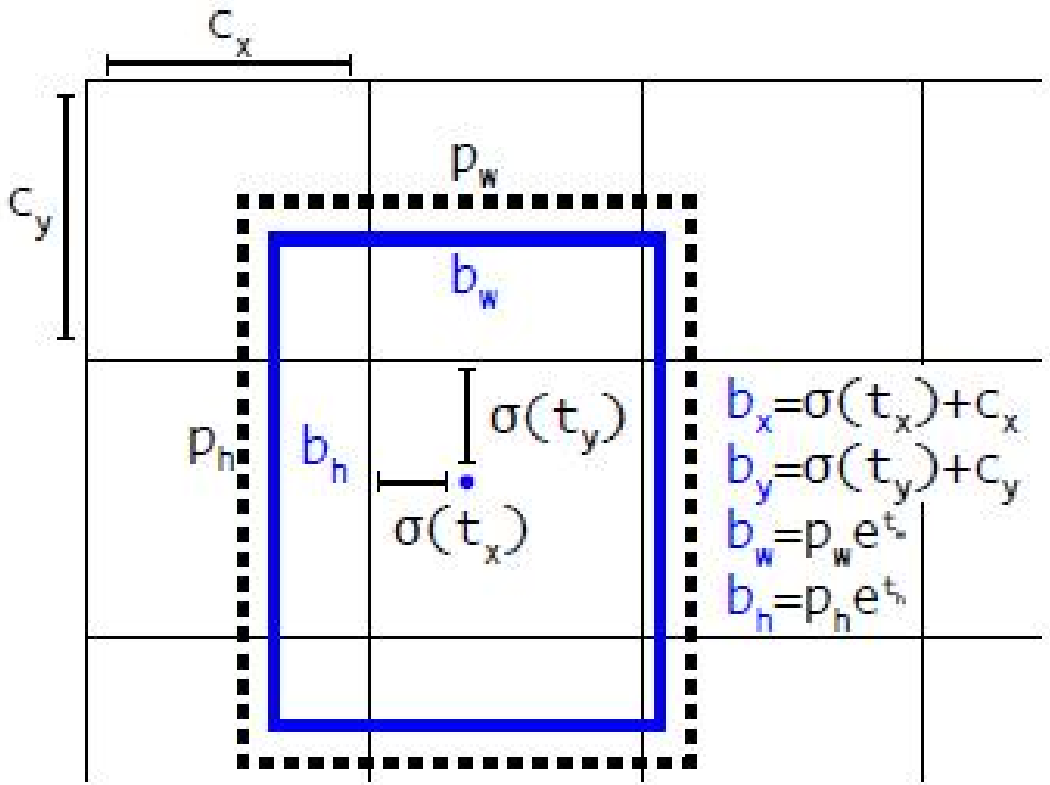


Figure 2. **Bounding boxes with dimension priors and location prediction.** We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

- **先验锚框的尺寸:** $[p_w, p_h]$
- **网络学到的目标坐标:** $[t_x, t_y, t_w, t_h]$
- **中心点偏移**:

$$b_{x=}c_x + \sigma(t_x)$$

$$b_{y=}c_y + \sigma(t_y)$$

- **宽高拉伸**:

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

- **坐标变换实现真实的绝对坐标:**

$$[t_x, t_y, t_w, t_h] \rightarrow [b_x, b_y, b_w, b_h]$$

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

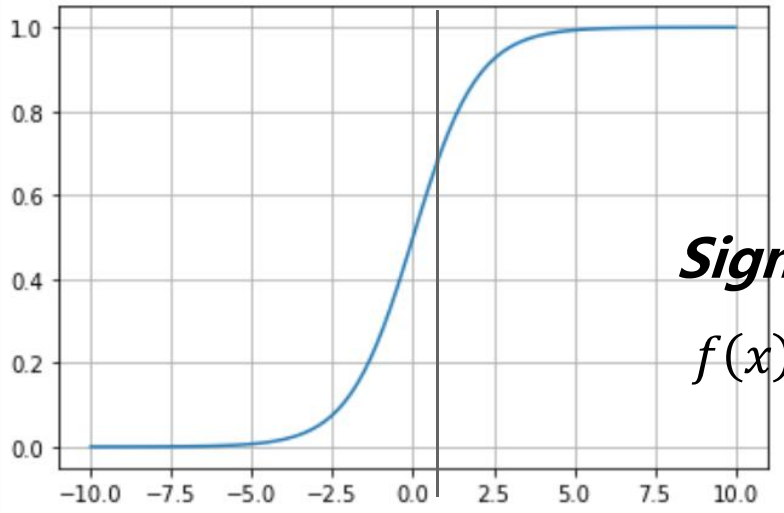
5.5 YOLOv3

从先验锚框到预测框

- 为什么在中心坐标(c_x, c_y)上取sigmoid，而宽度和高度上取exp?

中心点偏移：

$$b_{x=}c_x + \sigma(t_x)$$
$$b_{y=}c_y + \sigma(t_y)$$



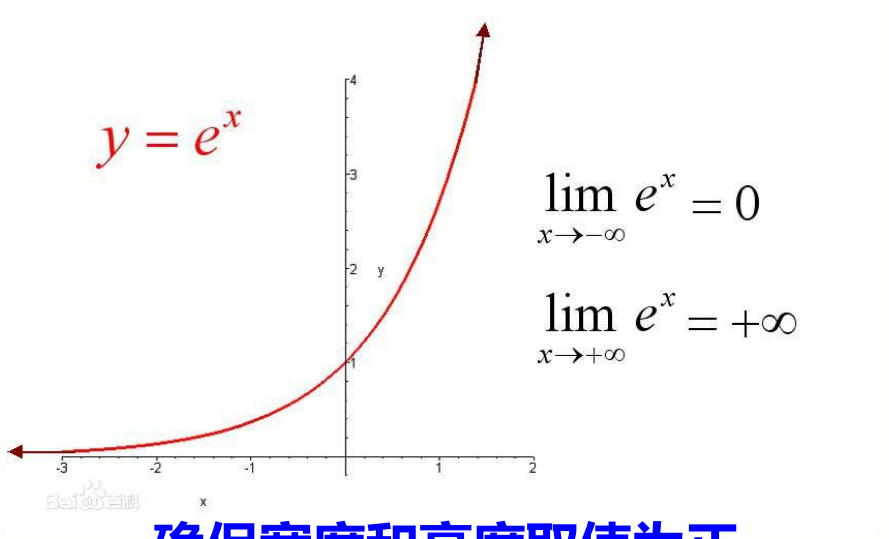
Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

确保中心点落在网格内

宽高拉伸：

$$b_h = p_h e^{t_h}$$
$$b_w = p_w e^{t_w}$$



确保宽度和高度取值为正

Joseph [illegible] YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

基于先验的锚框

- **先验锚框**只给出了框的高度和宽度，那如何去表示图像中物体的真实框？
  - ✓ 划分网格，并通过每个网格中的Anchor来匹配真实框
- **当样本中存在多个真实框（目标）时，如何对它们进行进行表示？**
  - ✓ 使用“框海战术”去匹配不同的真实框
- 如何在图像中对真实框进行定位？
  - ✓ 网格粗定位+中心点偏移精确定位
- **先验锚框**的高度和宽度时固定的，如何去适配物体各种不同尺度的高度和宽度？
  - ✓ 宽高拉伸

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

从先验锚框到预测框

- 为什么要使用先验锚框，而不是直接回归真实框的(x, y, w, h)呢？

训练过程可以看作是一个模型猜答案的过程！

一万年以后…

如果说我看得比别人更远些，
那是因为我站在巨人的肩膀上。
——牛顿

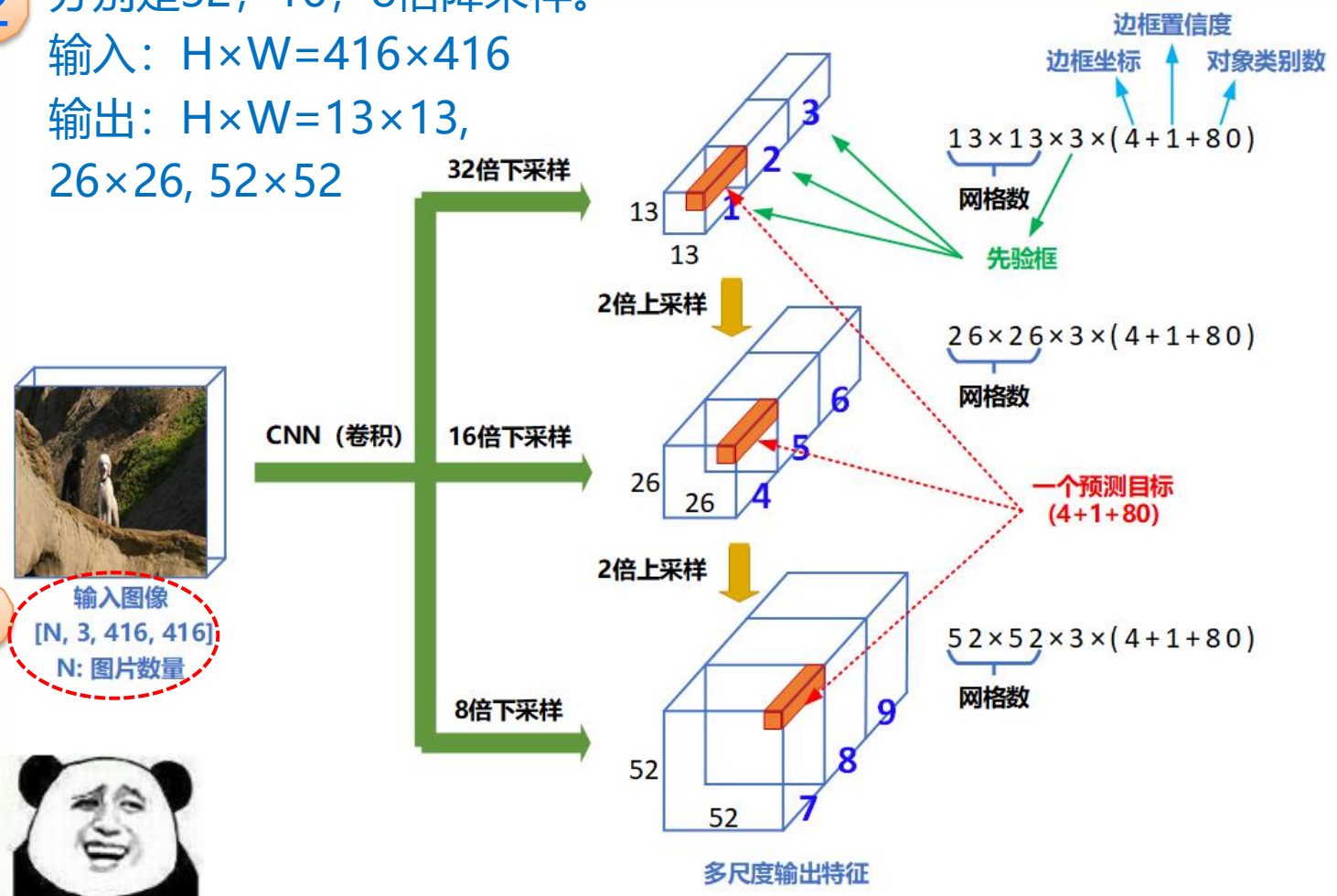
先验锚框就是YOLOv3的巨人。

- 在训练集上使用KMeans聚类获得先验锚框
- 生成的预测框仅需在先验锚框的基础上进行“微调”
- 收敛更快，效果更好
- PaddleDetection中提供tools/anchor_cluster.py

5.5 YOLOv3

YOLOv3的输入输出

2 模型网络有三个输出分支，分别是32，16，8倍降采样。
输入：H×W=416×416
输出：H×W=13×13, 26×26, 52×52



3 每个输出分支包含3个先验锚框Anchor

4 每个先验锚框对应的预测框包含85个信息：

- 坐标：t_x, t_y, t_h, t_w
- 目标概率：Objectness 对象性分数
- 分类：coco数据集包含80个类别（one-hot classification scores）

$4_{(坐标)} + 1_{(目标概率)} + 80_{(类别)} = 85$

$85 \times 3 = 255_{(锚框)}$

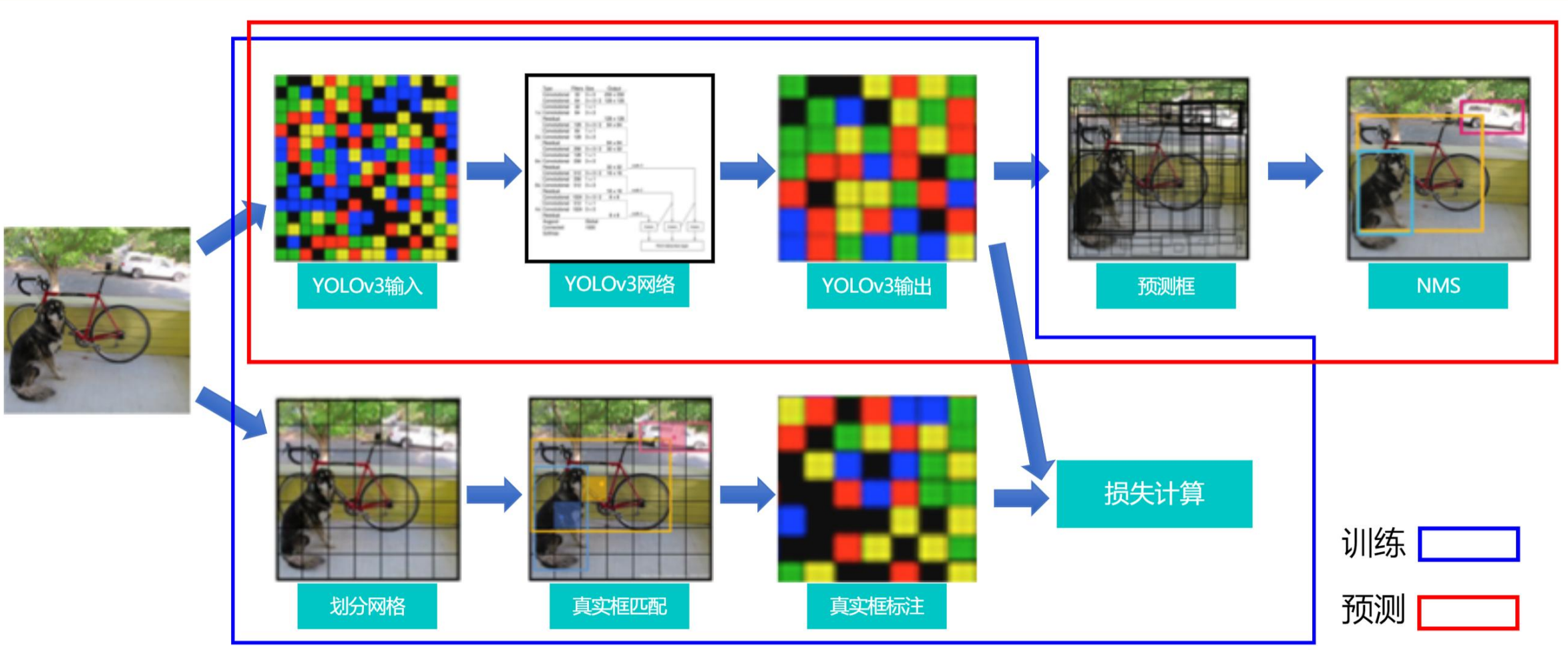
5 网络输出：

[N, 255, 13, 13]
[N, 255, 26, 26]
[N, 255, 52, 52]

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

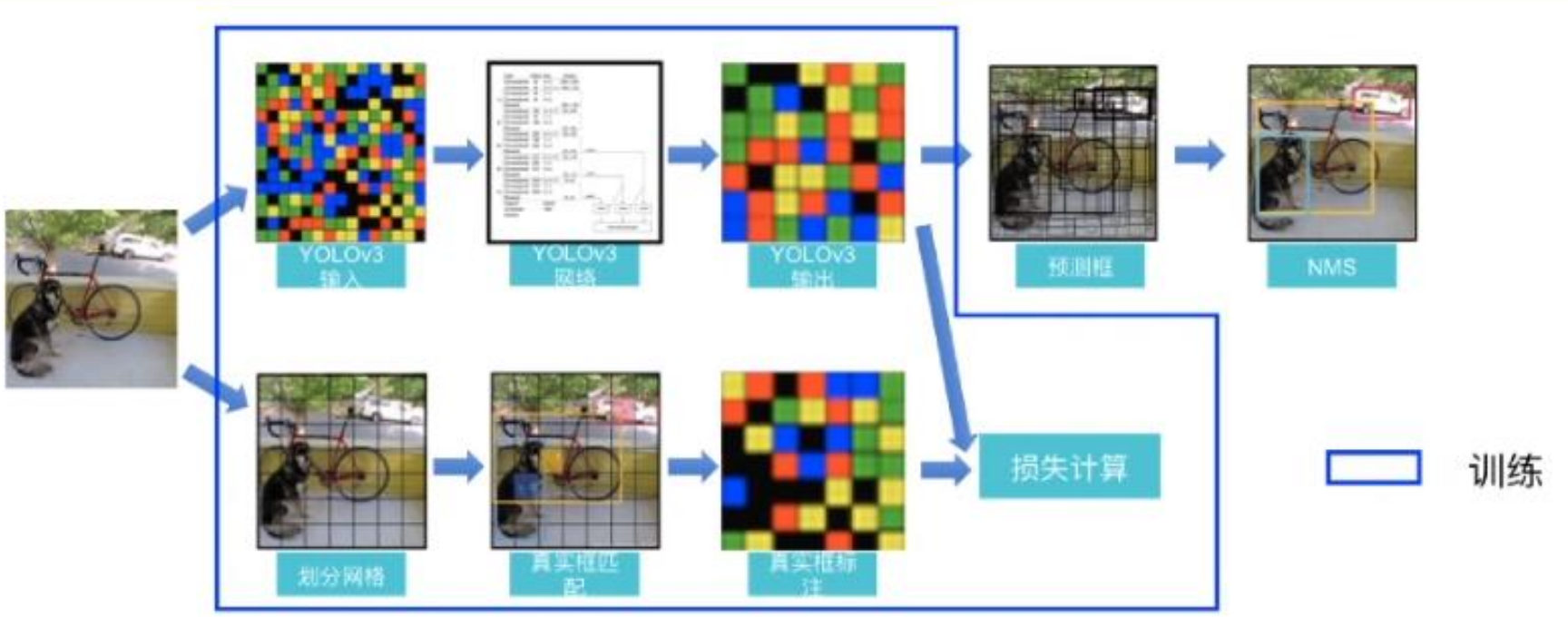
YOLOv3的检测流程



Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

YOLOv3的训练流程



输入 [[x, y, w, h, class]
[x, y, w, h, class]
…]

整容

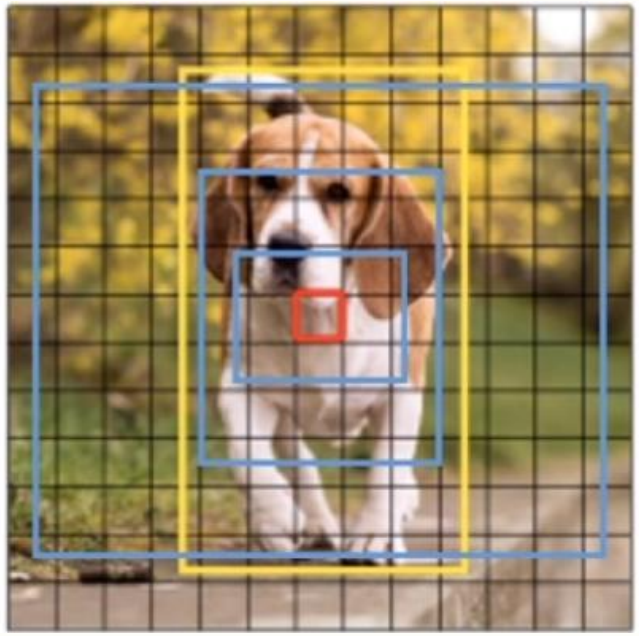
输出 [N, 255, 13, 13]
[N, 255, 26, 26]
[N, 255, 52, 52]

如何获得匹配算法的真实框标注?

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

真实框匹配



Q1：如何确定目标的位置?

A1：目标真实框的中心落在**哪个网格**，就由**哪个网格**负责检测该目标。

Q2：每个样本都存在三个特征图，那么哪个先验锚框负责检测当前目标?

A2：在所有的9个先验锚框中，与目标真实框IoU最大（最匹配）的**先验锚框**负责检测当前目标。

如果两个目标的真实框匹配到同一个网格的同一个先验锚框怎么办?

YOLOv3对此无能为力

5.5 YOLOv3

基于真实框的监督信息

- **先验锚框的尺寸:** $[p_w, p_h]$
- **网络学到的目标:** $[t_x, t_y, t_w, t_h]$
- **中心点偏移**:

$$b_{x=}c_x + \sigma(t_x)$$

$$b_{y=}c_y + \sigma(t_y)$$

- **宽高拉伸**:

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

- **坐标变换实现真实的绝对坐标:**

$$[t_x, t_y, t_w, t_h] \rightarrow [b_x, b_y, b_w, b_h]$$

- **锚框尺寸** $[p_w, p_h]$，**真实框** $[gt_x, gt_y, gt_w, gt_h]$
- **中心点偏移**:

$$b_{x=}c_x + \sigma(t_x) \quad \Rightarrow \quad \sigma(t_x^*) = gt_x - c_x$$

$$b_{y=}c_y + \sigma(t_y) \quad \Rightarrow \quad \sigma(t_y^*) = gt_y - c_x$$

- **宽高拉伸**:

$$b_w = p_w e^{t_w} \quad \Rightarrow \quad t_w^* = \log(\frac{gt_w}{p_w})$$

$$b_h = p_h e^{t_h} \quad \Rightarrow \quad t_h^* = \log(\frac{gt_h}{p_h})$$

- **坐标变换:**

$$[gt_x, gt_y, gt_w, gt_h] \rightarrow [t_x^*, t_y^*, t_w^*, t_h^*]$$

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

基于真实框的监督信息

- **目标对象性概率Objectness**

$$score_{obj} = \sigma(t_{obj})$$

存在　真实框：$t^*_{obj} = 1$

不存在真实框：$t^*_{obj} = 0$

- **分类分数：各类独立、支持多分类**

$$score_{c_k} = \sigma(t_{c_k})$$

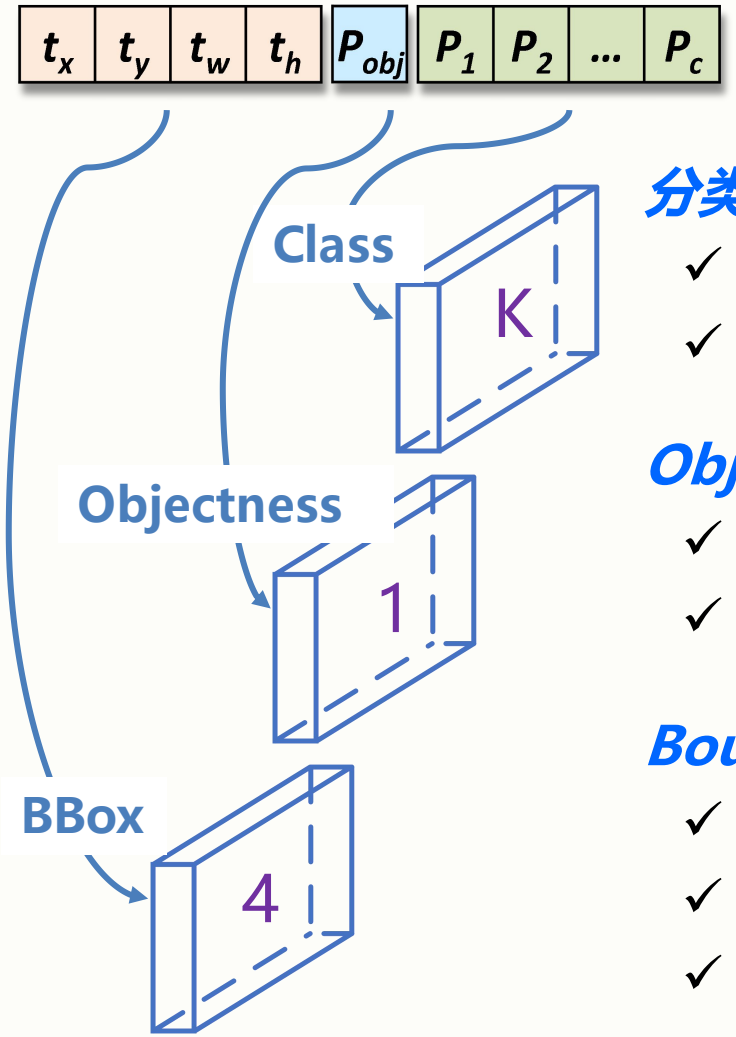
是真实类别：$t^*_{ck} = 1$

非真实类别：$t^*_{ck} = 0$

Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

损失函数



分类类别

- ✓ YOLOv3使用sigmoid激活函数
- ✓ 分类损失为Sigmoid Cross Entropy损失

Objectness对象性

- ✓ 一个0~1之间的目标性评分，使用sigmoid激活函数
- ✓ 使用Sigmoid Cross Entropy损失

Bounding Box 定位

- ✓ x, y使用sigmoid激活，并使用Sigmoid Cross Entropy损失
- ✓ w, h使用exp缩放，并使用L1损失
- ✓ 定位损失上乘以scale，平衡大小真实框的权重，增加小框损失敏感度

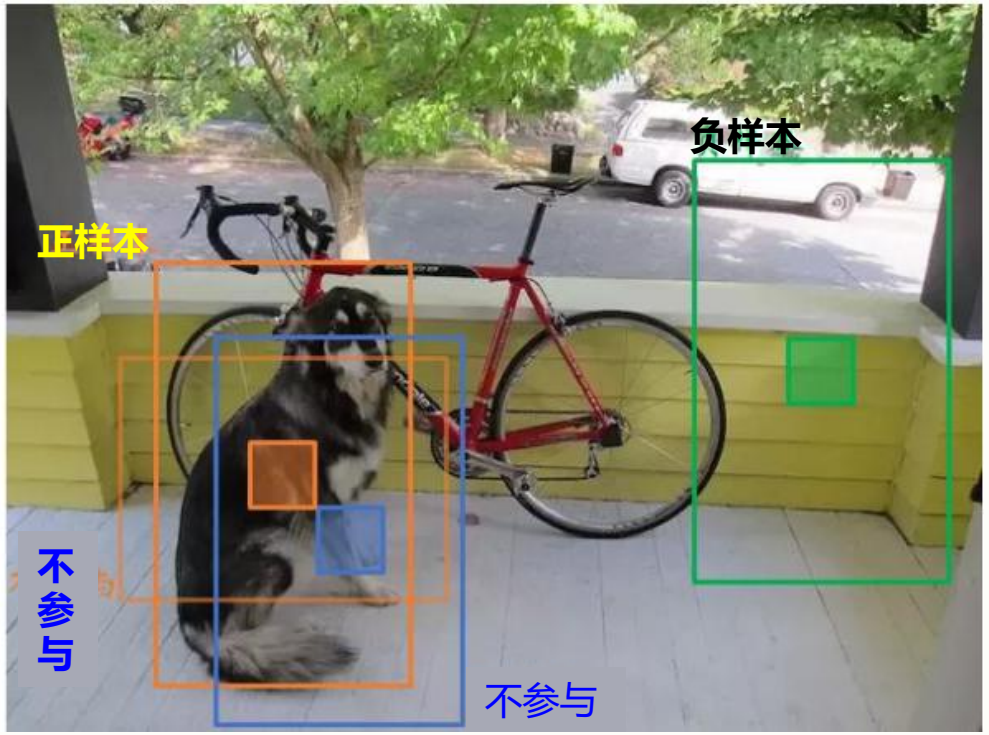
Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

损失函数

哪些数据需要计算损失?

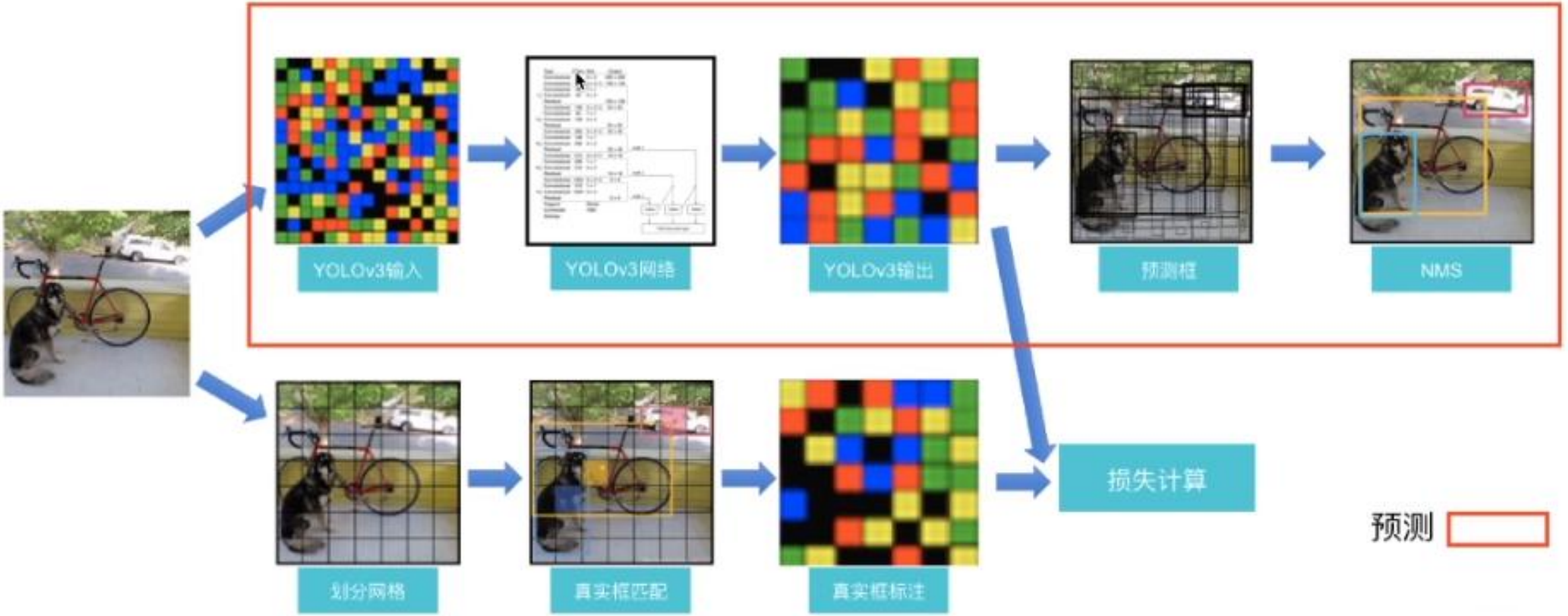
- **正例**：有真实框落在内的先验锚框
  - **BBox定位损失**：形变系数t_x, t_y, t_w, t_h向真实框的值回归
  - **Objectness对象性损失**：将预测框的objectness向1回归
  - **分类损失**：分类概率向真实框类别one-hot编码回归
- **负例**：没有真实框落在内的先验锚框(IoU<0.5)
  - **Objectness对象性损失**: 将预测框的objectness向0回归
  - 其他两项损失不做计算
- **非正非负**：与真实框的IoU较大(>0.7)，但不是匹配的先验锚框
  - 预测框与任一真实框IoU大于Ignore threshold，不计算损失



Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

YOLOv3的预测流程



Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

5.5 YOLOv3

计算预测框的输出

- **中心点偏移，宽高拉伸：**

$$b_{x=}c_x + \sigma(t_x) \qquad b_w = p_w e^{t_w}$$

$$b_{y=}c_y + \sigma(t_y) \qquad b_h = p_h e^{t_h}$$

- **目标对象性概率Objectness**

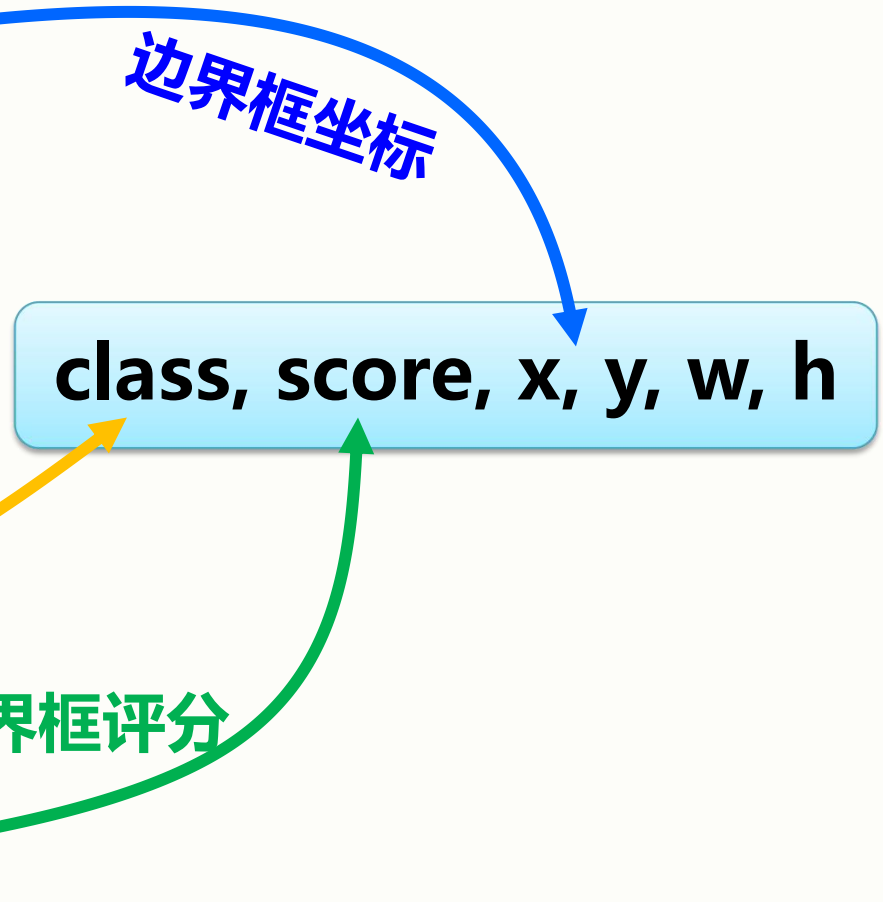
$$score_{obj} = \sigma(t_{obj})$$

- **分类分数，各类独立支持多分类**

$$score_{c_k} = \sigma(t_{c_k})$$

- **预测框评分：有物体且是该类别**

$$score_{bbox} = score_{obj} * score_{c_k}$$



Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804

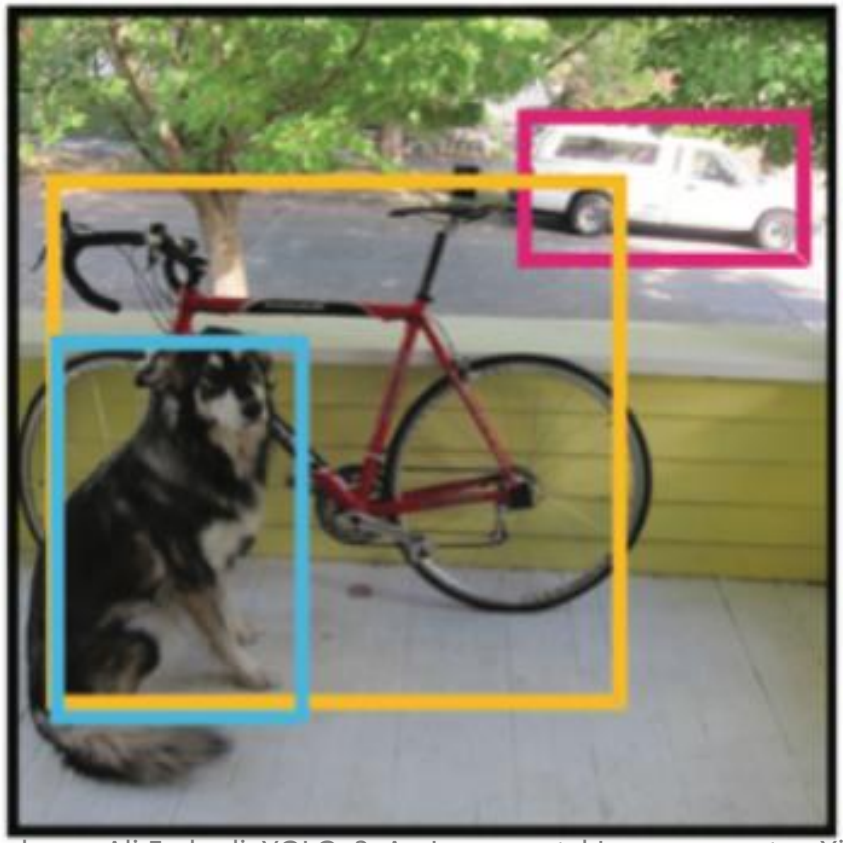
5.5 YOLOv3

NMS非极大抑制

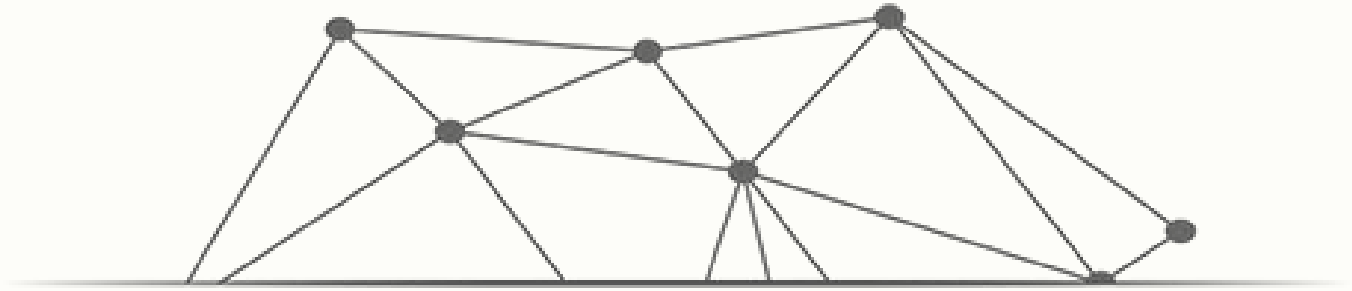
class, score, x, y, w, h

NMS

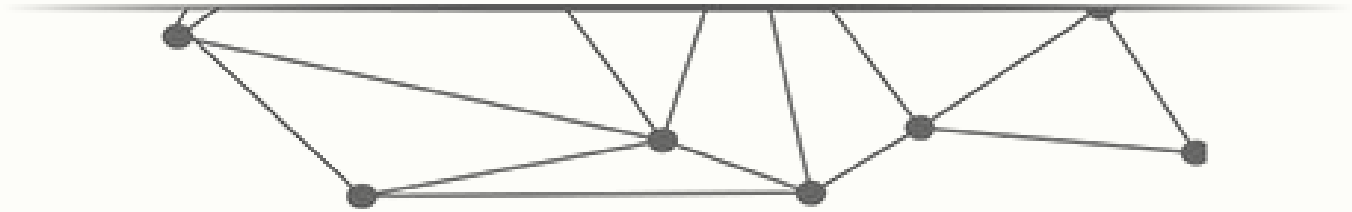
1. 按class分组
2. 按score排序
3. 按[x, y, w, h]计算IoU
4. 保留IoU最大的样本
5. 删除剩余样本

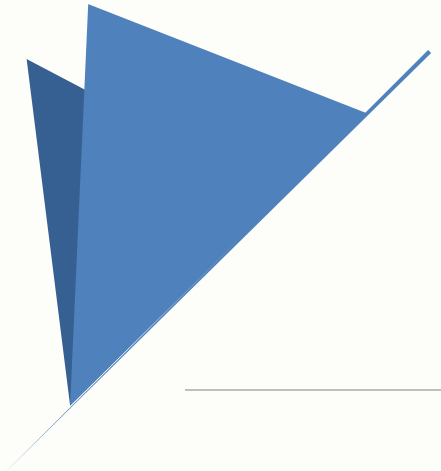


Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv1804



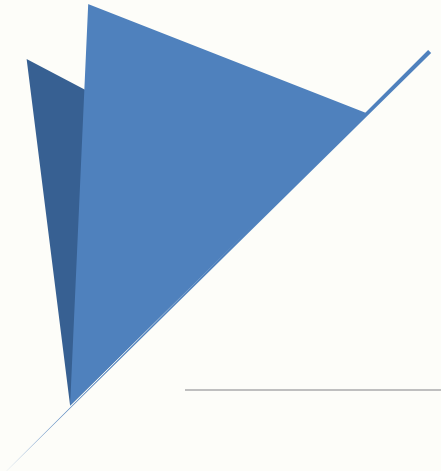
课堂互动 13.2.9





YOLO v4 v5

Come soon...
请自行学习



PP-YOLO

5.7 PP-YOLO

YOLO系列模型的发展史

YOLOv01 (2015)

- 首个单阶段目标检测深度学习模型
- 将目标检测当作一个单一的回归任务
- 基于锚框学习形变系数

YOLOv2 (2016)

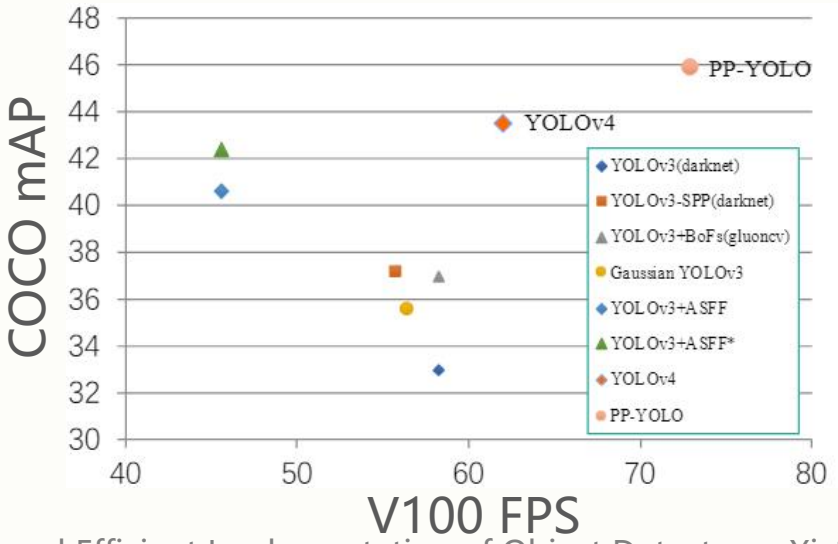
- 全卷积网络
- Kmeans聚类锚框
- 多尺度训练
- 多类别识别

YOLOv3 (2018)

- 新骨干网络DarkNet53
- Kmeans聚类9个锚框，并分为大中小三组
- 多尺度训练

YOLOE (2025)

模型	年份	COCO mAP	FPS (V100)
YOLOv3(darknet)	2018	33.0	58.2
YOLOv3-SPP(darknet)	2018	37.1	55.8
YOLOv3 + BoFs(GluonCV)	2019	37.0	58.2
Gaussin YOLOv3	2019	35.6	56.4
YOLOv3 + ASFF	2019	40.6	45.5
YOLOv3 + ASFF*	2019	42.4	45.5
YOLOv4	2020	43.5	62.0
PP-YOLO	2020	45.9	72.9



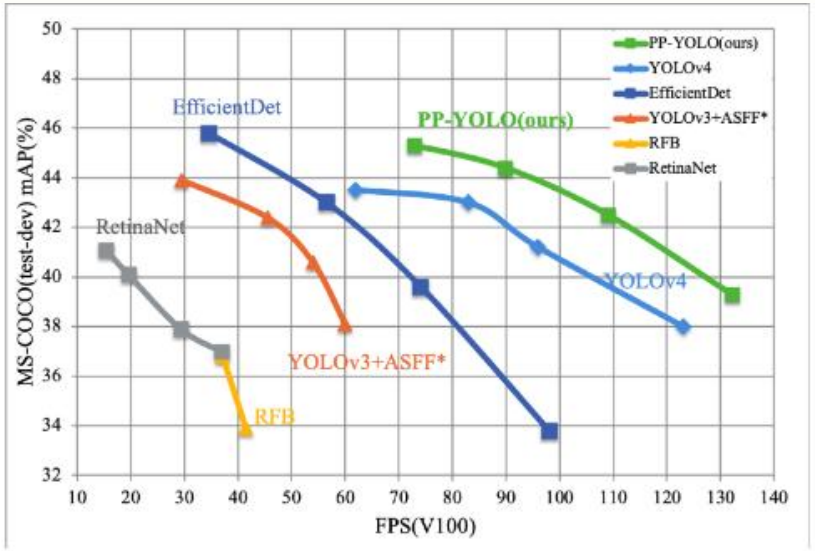
- ✓ 具有较高精度速度性价比的YOLO系列模型被广泛应用于实时监测的业务场景
- ✓ YOLOv3之后各种基于tricks的优化方法被提出，但综合性能并不是特别理想
- ✓ 2020年上半年的YOLOv4和v5得到了空前的成功，超越了大多数两阶段检测模型

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO简介

- 一个基于YOLOv3的优化模型
- 集成了大量tricks，但不增加额外计算量的优化模型
- 精度和速度都优于YOLOv3，和同时期的YOLOv4
- 2021年推出的PP-YOLO v2，性能全面超越YOLOv5

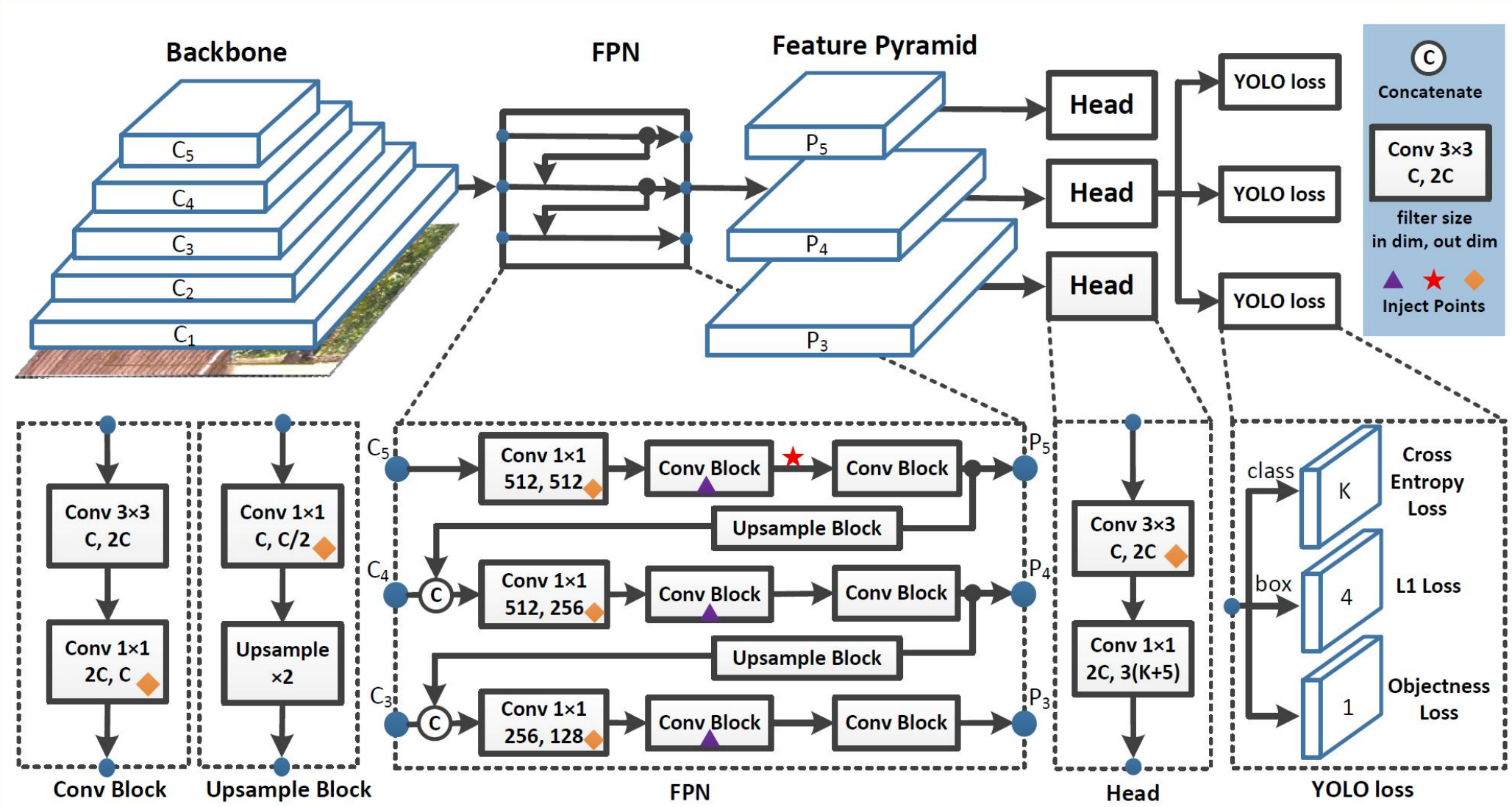


输入尺寸	COCO mAP		V100 FP32 FPS		V100 TRT-FP16 FPS	
	PP-YOLO	YOLOv4	PP-YOLO	YOLOv4	PP-YOLO	YOLOv4
608	45.9	43.5	72.9	62.0	155.6	105.5
512	45.0	43.0	89.9	83.0	188.4	138.4
416	43.2	41.2	109.1	96.0	215.4	164.0
320	40.1	38.0	132.2	123.0	242.2	199.0

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO模型结构



Detector. arXiv2007

5.7 PP-YOLO

目标检测常见的优化方法

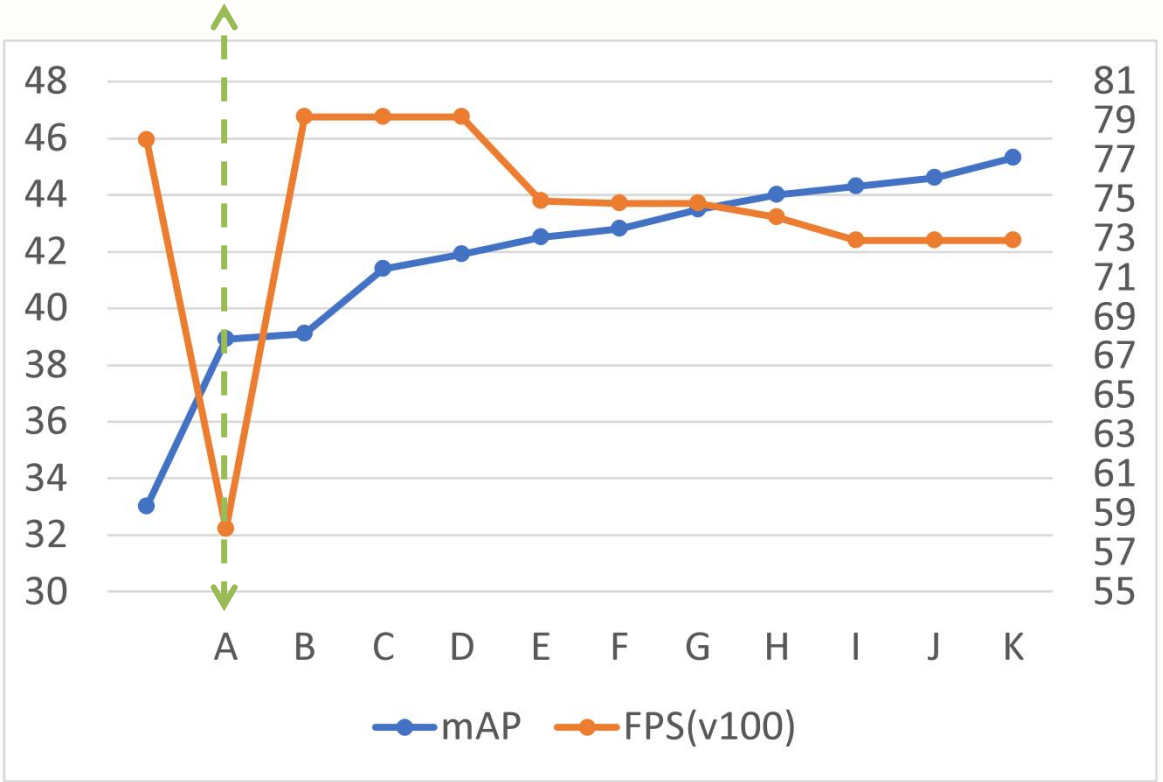
- **数据增强**
  - Image Mixup
  - Auto Augment
  - Image Mosaic
- **数据采样**
  - 难样本挖掘 OHEM
  - Class aware sampling
- **网络结构**
  - 更优的Backbone
  - Deformable Conv
  - Coord Conv
  - SPP
  - Drop Block
- **学习策略**
  - IoU/GIoU/DIoU /CIoU Loss
  - IoU Aware
  - Label smooth
  - 加权Loss
  - Grid Sensitive
- **训练策略**
  - 调整BatchSize/ 学习率/迭代次数
  - Synchronize batch norm
  - EMA
- **后处理**
  - Soft NMS
  - Matrix NMS

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

YOLOv3-DarkNet53的优化

- **Image Mixup**
  - ✓ 训练前250个epoch每幅图像都使用任意两张图象进行混合(Mixup)
  - ✓ 最后20个epoch不做混合
- **Label Smooth**
  - ✓ 对GroudTurth Label进行微调，优化目标的分类精度
  - ✓ 对One-Hot目标的正例的1减去一个小量；负例的0增加一个小量
- **Synchronized Batch Norm**
  - ✓ 多卡训练时，进行卡间均值和方差的同步

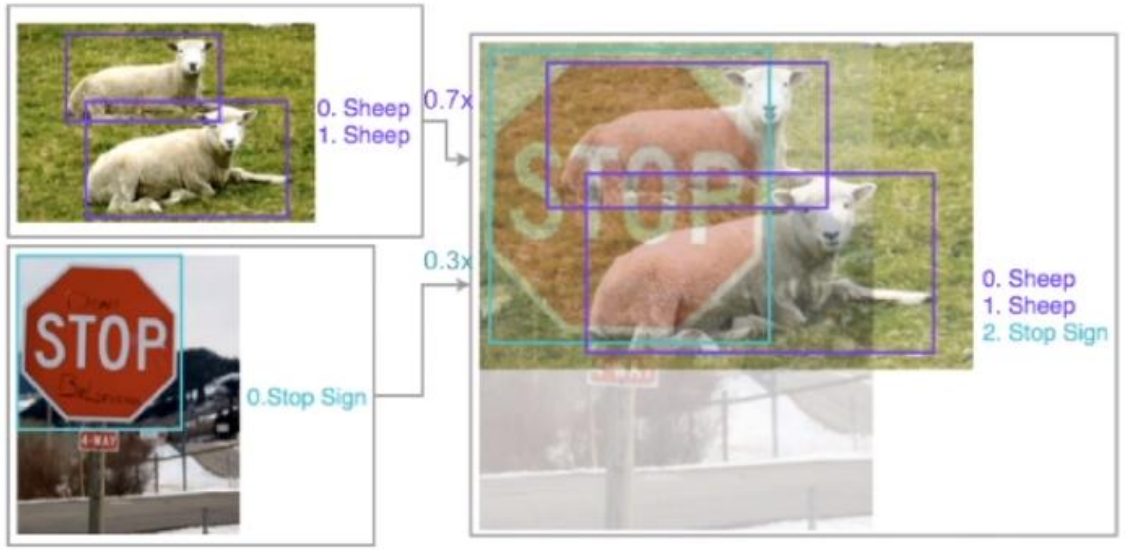
优化效果：COCO mAP 33.0 ->38.9 预测速度不变

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

Image Mixup

效果：COCO数据集mAP提升 ≈ 1%



具体实现方法

- ✓ 训练前250个epoch每幅图像都使用任意两张图象进行融合(Mixup)
- ✓ 最后20个epoch不做混合

优化点

- ✓ 提高网络对空间扰动的泛化能力
- ✓ 一定程度上增加正样本比例，提高召回率

计算方法

- ✓ Mixup的方法：Sheep像素值乘0.7与Stop Sign像素值乘0.3，左上角对齐叠加
- ✓ Loss的计算：Sheep的真实框(GT BBox)引入的Loss乘0.7+Stop Sign的真实框引入的Loss乘0.3

```
sample_transforms:
  - Decode: {}
  - Mixup: {alpha: 1.5, beta: 1.5}
  - RandomDistort: {}
  - RandomExpand: {fill_value: [123.675, 116.28, 103.53]}
```

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

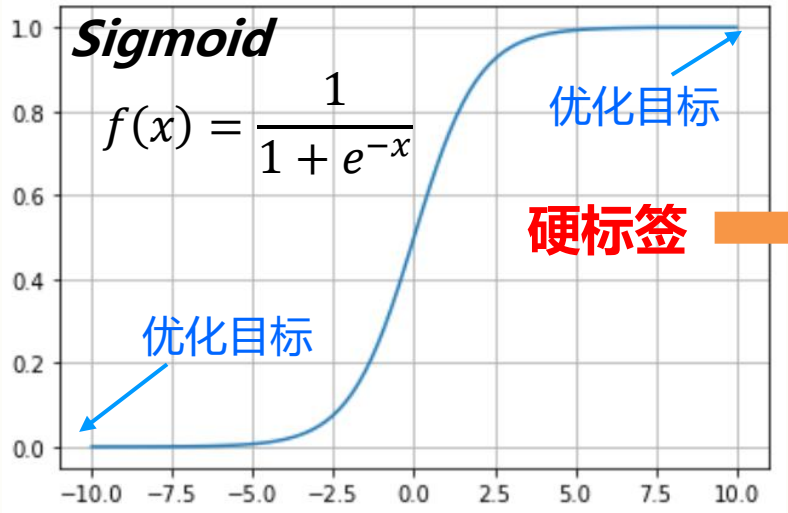
5.7 PP-YOLO

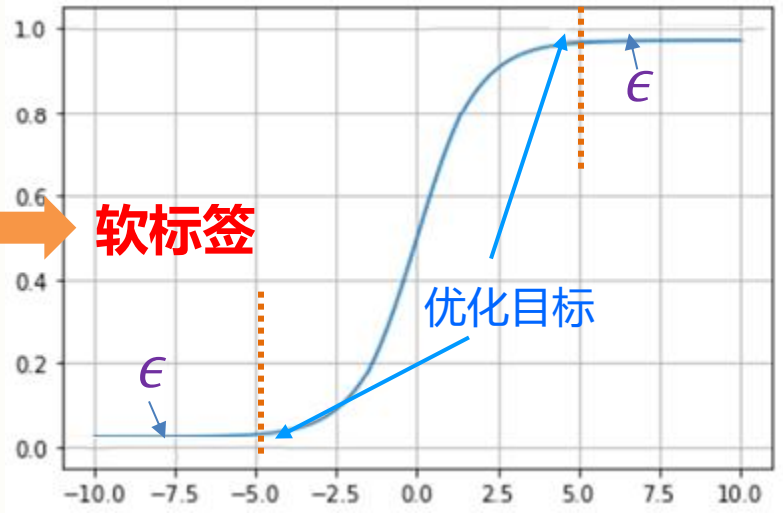
Label Smooth

效果：COCO数据集mAP提升 ≈ 0.2-0.3%

优化点

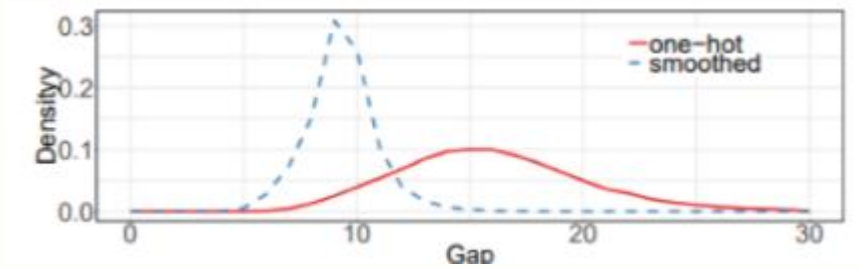
- ✓ one-hot的硬标签会将输出特征图向+∞获-∞方向优化，从而导致过分拟合，降低分类精度
- ✓ 通过小量扰动，将硬标签转变为软标签将缓解过拟合问题





计算方法

$$g_i \begin{cases} 1-\epsilon & if\ i=y \\ \epsilon & otherwise \end{cases}$$




```
YOLOv3Loss:
  ignore_thresh: 0.7
  downsample: [32, 16, 8]
  label_smooth: false
```

具体实现方法

- ✓ 对One-Hot目标的正例的1减去一个小量；负例的0增加一个小量

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

Synchronized Batch Norm

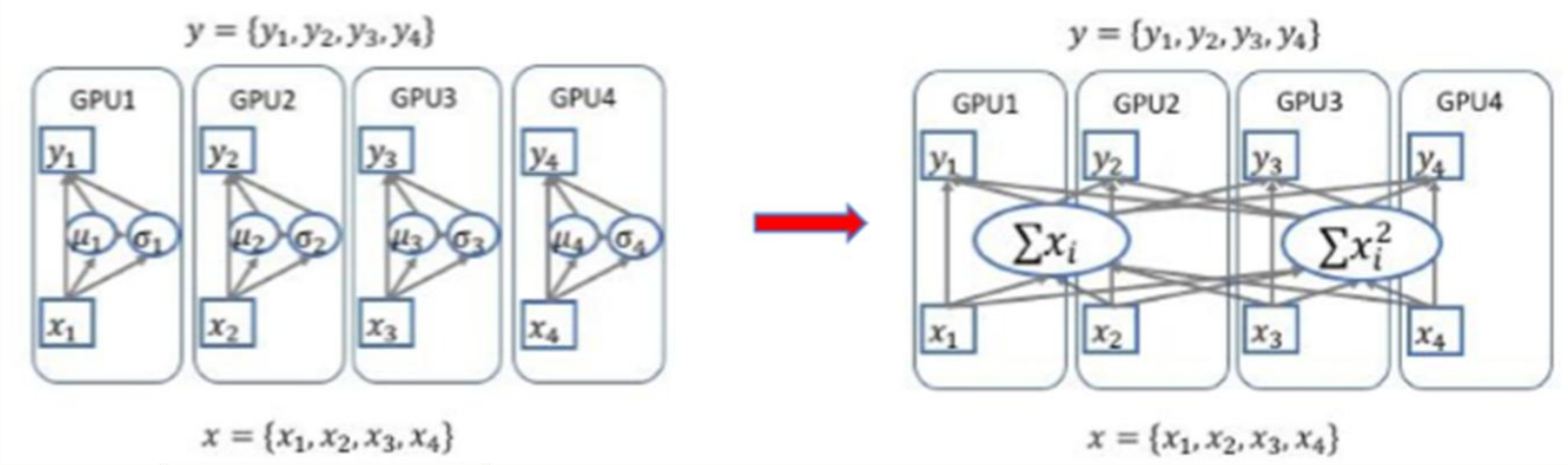
同步批正则化（Synchronized Batch Norm）

用于多卡并行的场景，实现在多GPU的机器上所有GPU上的均值和方差的同步。

卡间同步，无形间增大了BatchSize，使得均值方差更合理，收敛速度和收敛效果更好。

```
architecture: YOLOv3
pretrain_weights: https://p
norm_type: sync_bn
```

优化效果：COCO数据集mAP提升 ≈ 1.2%

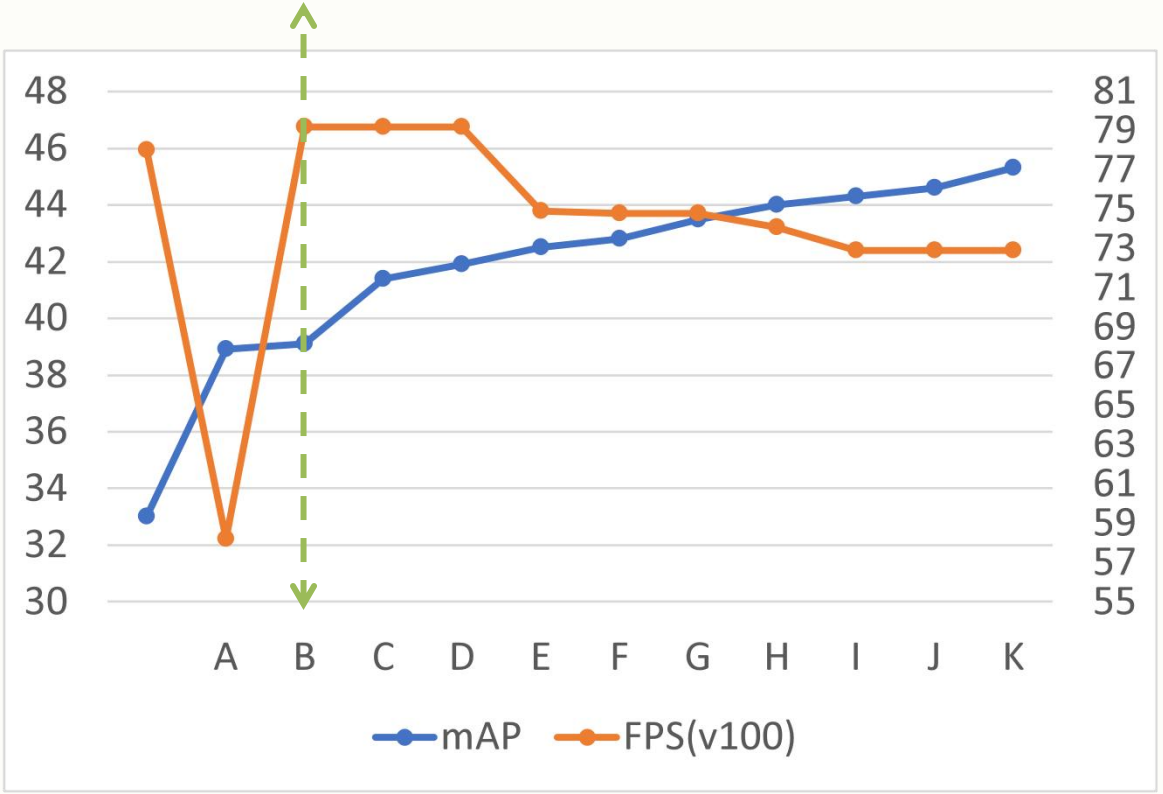


Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9

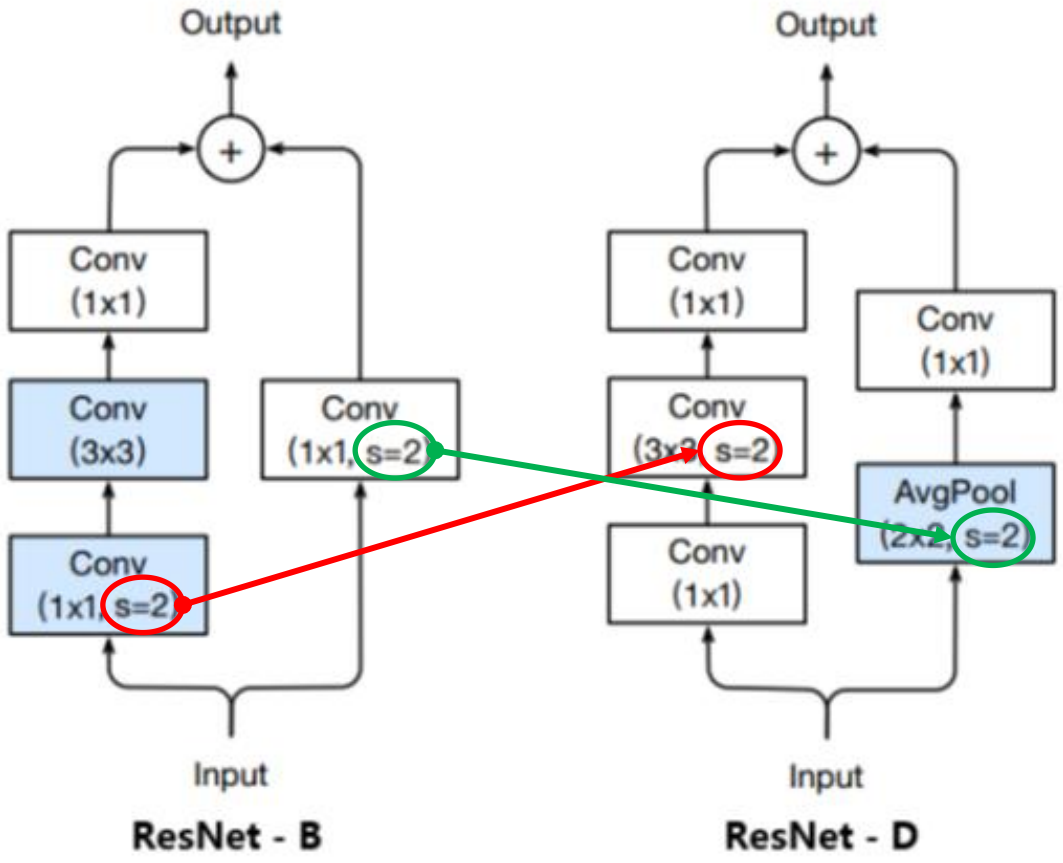


Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

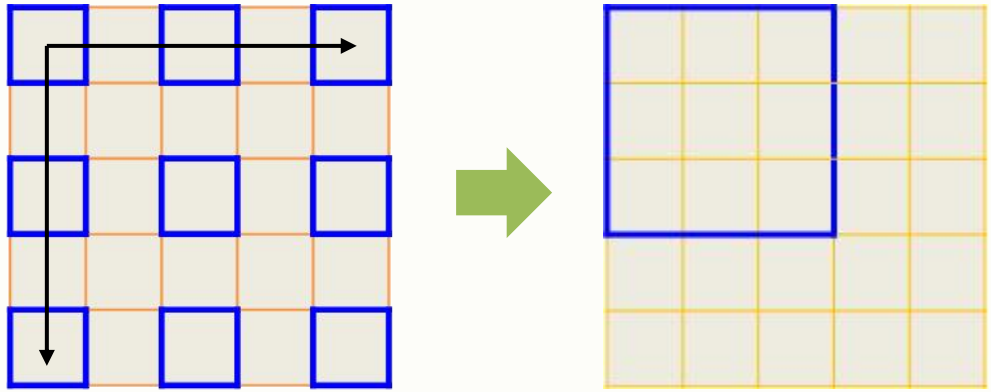
ResNet-D

效果：预测速度不变，模型精度提高 ≈ 1-2%



优化点

✓ 每个阶段的下采样都会丢失大约3/4的信息



优化方法

✓ 修改和调整stride=2的位置

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

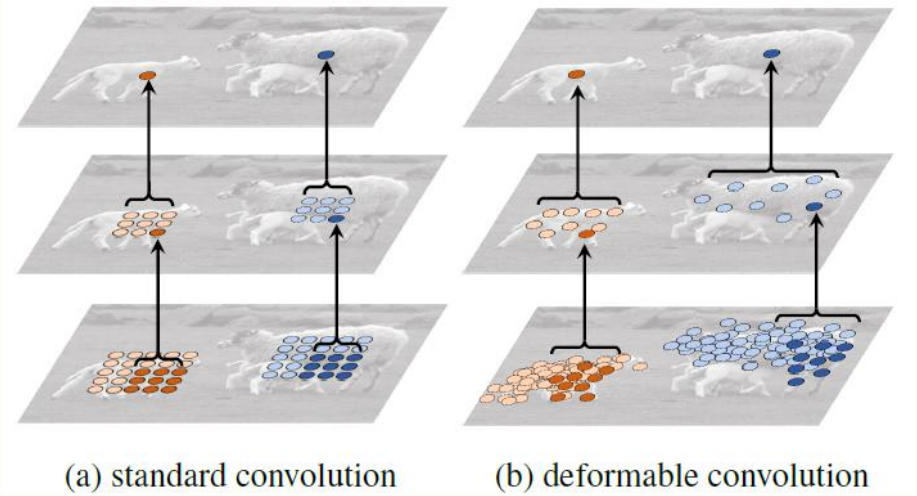
5.7 PP-YOLO

Deformable Convolutional Networks

Deformable Conv

优化点

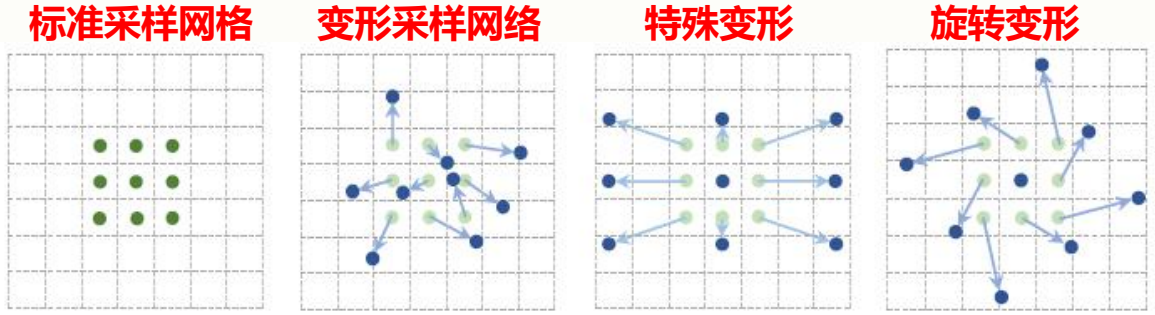
✓ 在学习权重的过程中学习卷积核的形状

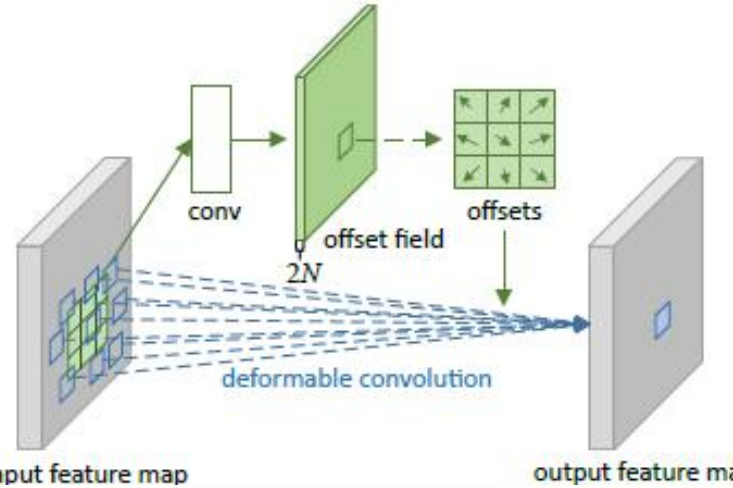


(a) standard convolution (b) deformable convolution

优化方法

✓ 学习一个**offset**表示卷积核的偏移量，再学习一个**mask**表示哪些点需要计算偏移量。






```
ResNet:
  depth: 50
  variant: d
  return_idx: [1, 2, 3]
  dcn_v2_stages: [3]
  freeze_at: -1
  freeze_norm: false
  norm_decay: 0.
```

效果：

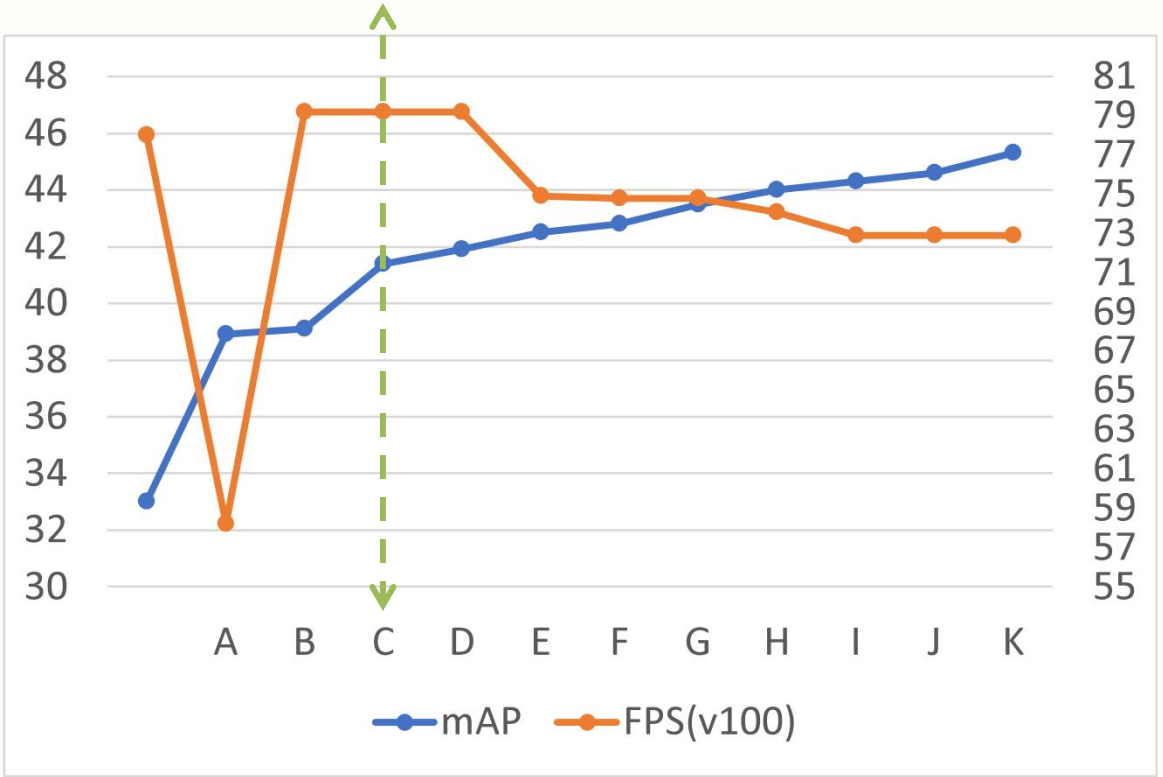
- ***预测速度不变，模型精度提高 ≈ 0.2%***
- ***预测速度提升 ≈ 20FPS***

Xiang Long, et. of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9

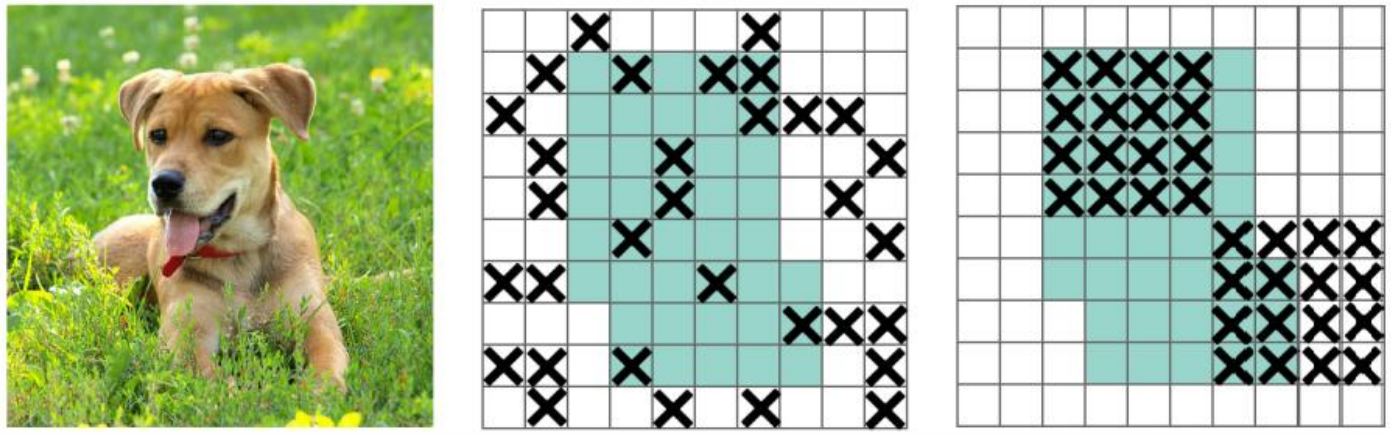


Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

DropBlock: A regularization method for convolutional networks

Drop Block



Input　　Dropout　　DropBlock

优化点

- ✓ Dropout是深度学习常用的降低过拟合、提高网络泛化能力的方法。但随机丢弃破坏了目标对象的针对性，稀释了目标的“浓度”
- ✓ 每次Drop一个区域，而非零散的点，更适用于检测“连通域”的目标检测任务

计算方法

- ✓ 使用随机Drop块(Block)，替代随机Drop离散点(Point)

```
PPYOLOFPN:
  coord_conv: true
  drop_block: true
  block_size: 3
  keep_prob: 0.9
  spp: true
```

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

Exponential Moving Average

指数平均数指标(Exponential Moving Average，EXPMA或EMA) 是一种趋向类指标，其构造原理是对价格收盘价进行算术平均，并根据计算结果来进行分析，用于判断价格未来走势的变动趋势。

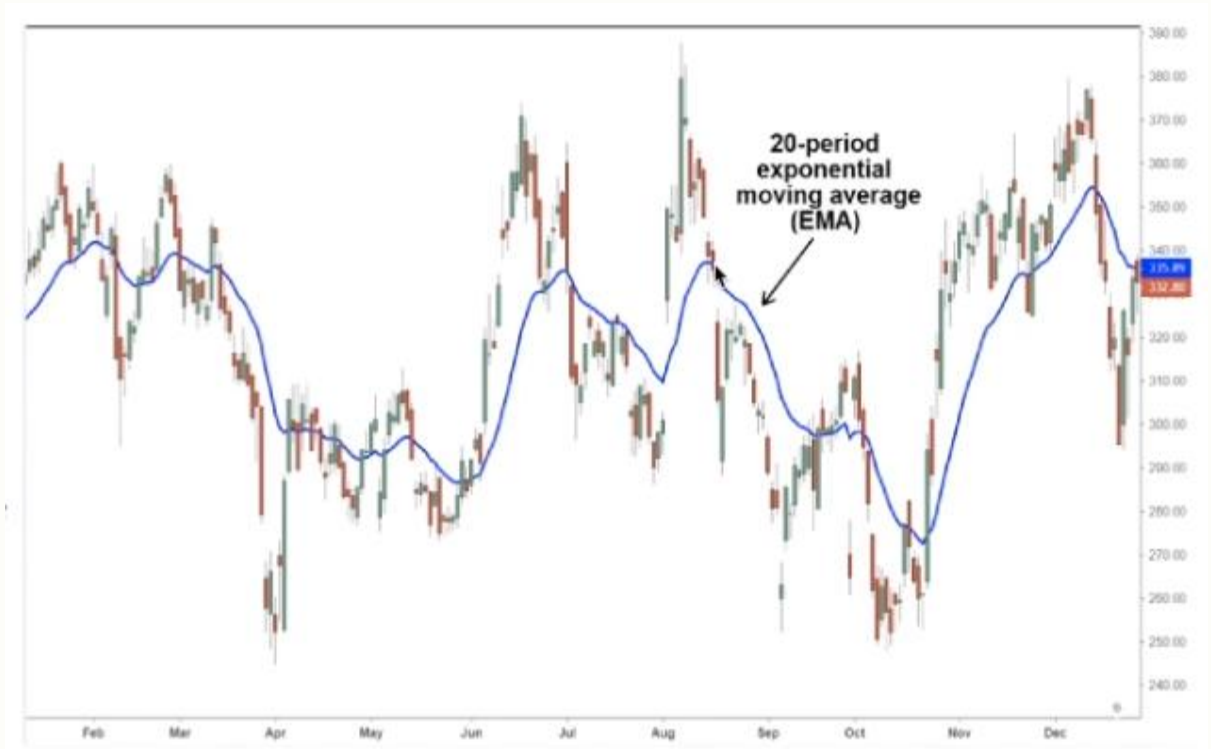
优化方法

EMA被广泛的应用在深度学习的BN层中和RMSprop，adadelta，adam等梯度下降方法。在模型训练过程中，通过对参数进行历史滑动平均，从而降低抖动来使训练更平缓稳定。

计算方法

$$\mathrm{EMA}_0 = 0$$

$$\mathrm{EMA}_t = \text{decay}^* \mathrm{EMA}_{t-1} + (1\text{-decay})^* \theta_t$$




```
3    norm_type: sync_bn
4    use_ema: true
5    ema_decay: 0.9998
```

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

Larger Batch Size

增大Batch Size

- ✓ 增大batch size后每次迭代计算的样本数增多，模型训练更加稳定
- ✓ 每卡mini-batch size由8增加到24（GPU可容纳范围内）
- ✓ Learning rate调整到0.01

优化效果

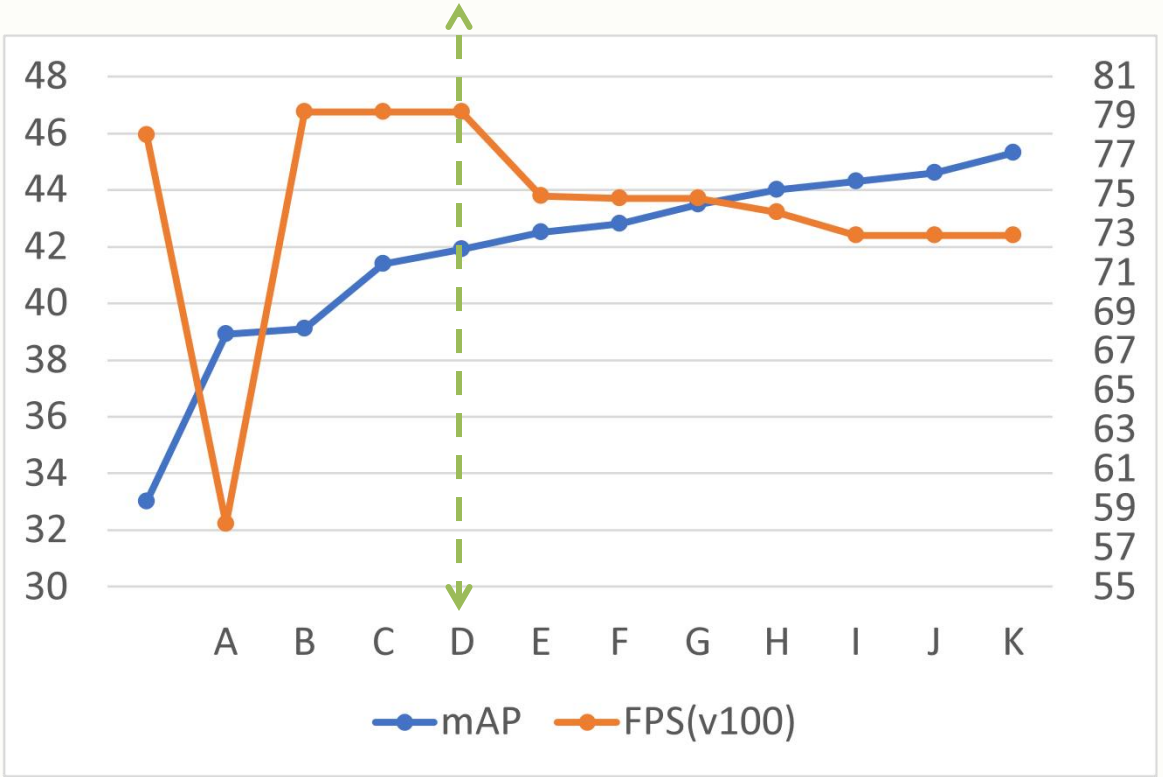
- ✓ EMA + DropBlock + Larger Batch Size：COCO数据集mAP提升 ≈ 2.3%
- ✓ 推理速度不变（79.2 FPS）

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regressio

IoU Loss

优化方法

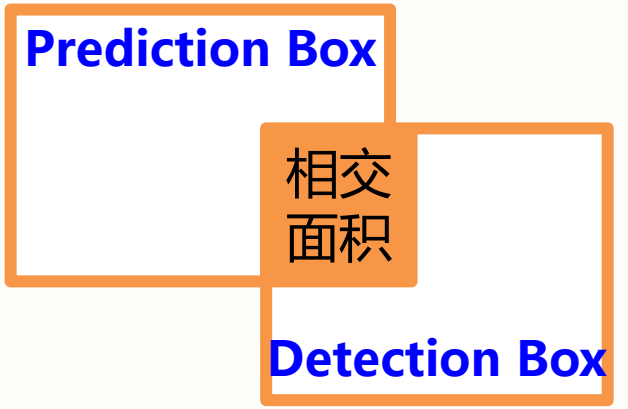
- ✓ IoU：检测框定位精度标准
- ✓ 所见即所得，将IoU加入到Loss全家桶中

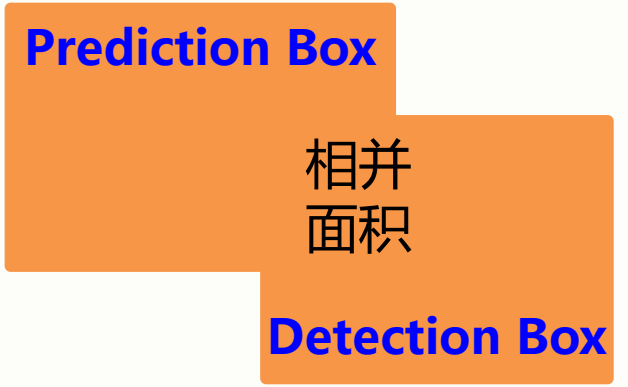
计算方法

- ✓ 预测框特征图与真实框特征图计算IoU
- ✓ IoU Loss = (1 – IoU*IoU) *$LossWeight_{IoU}$

```
IouLoss:
  loss_weight: 2.5
  loss_square: true
```

$$IoU = \frac{\text{重叠的区域}}{\text{相并的区域}}$$





优化效果

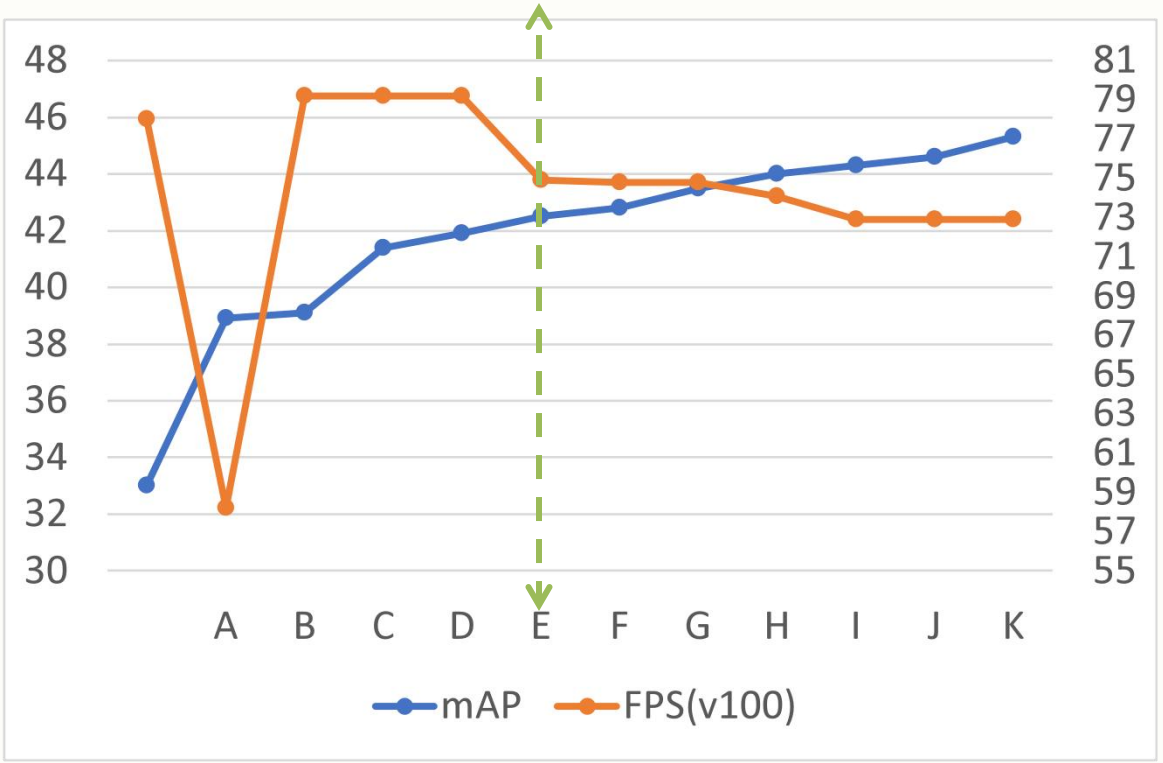
- ✓ ***COCO数据集mAP提升 ≈ 0.4%***
- ✓ ***推理速度不变***

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

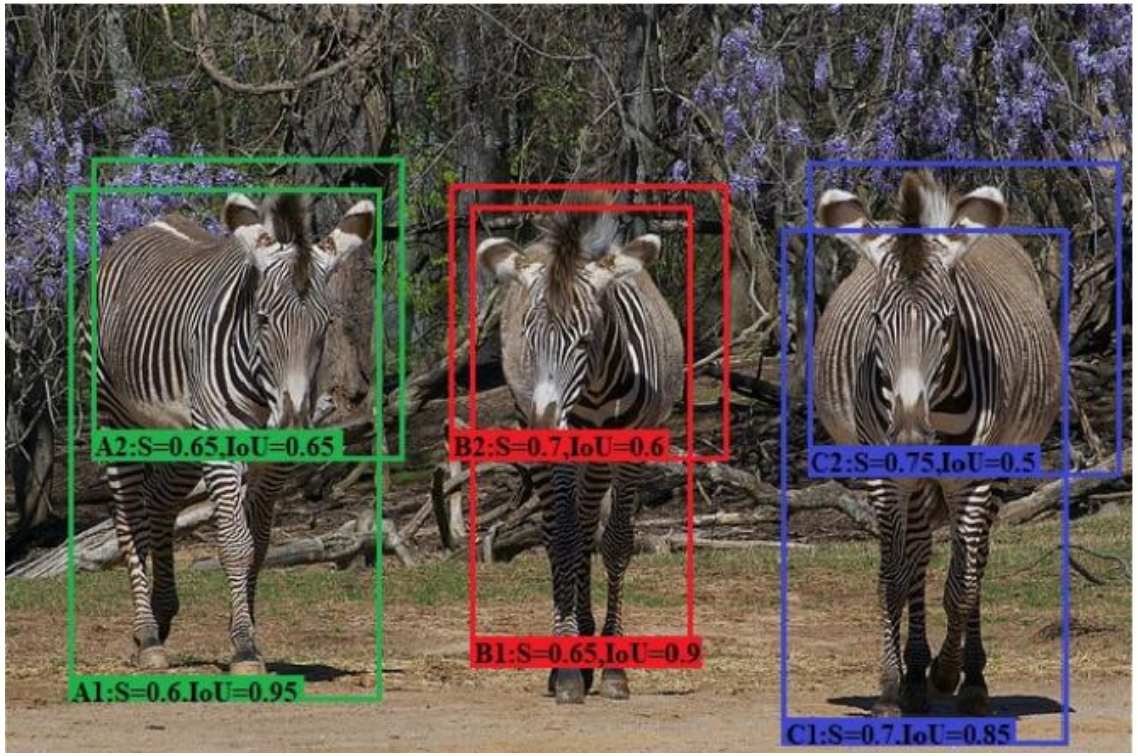
5.7 PP-YOLO

IoU-aware Single-stage Object Detector for Accurate Localization

IoU Aware

优化点

- ✓ 准确的检测框Score不高时，容易被NMS过滤
- ✓ 考虑将定位精度指标加入到Score



计算方法

- ✓ 增加一个Channel来学习IoU，即输出通道数由B*(5+C)变为B*(6+C)
- ✓ 预测时，Objectness的预测值更新为：

$$S_{det} = p_i^{\alpha} IoU_i^{(1-\alpha)}$$

```
YOLOv3Loss:
  ignore_thresh: 0.7
  downsample: [32, 16, 8]
  label_smooth: false
  scale_x_y: 1.05
  iou_loss: IouLoss
  iou_aware_loss: IouAwareLoss

IouAwareLoss:
  loss_weight: 1.0
```

优化效果

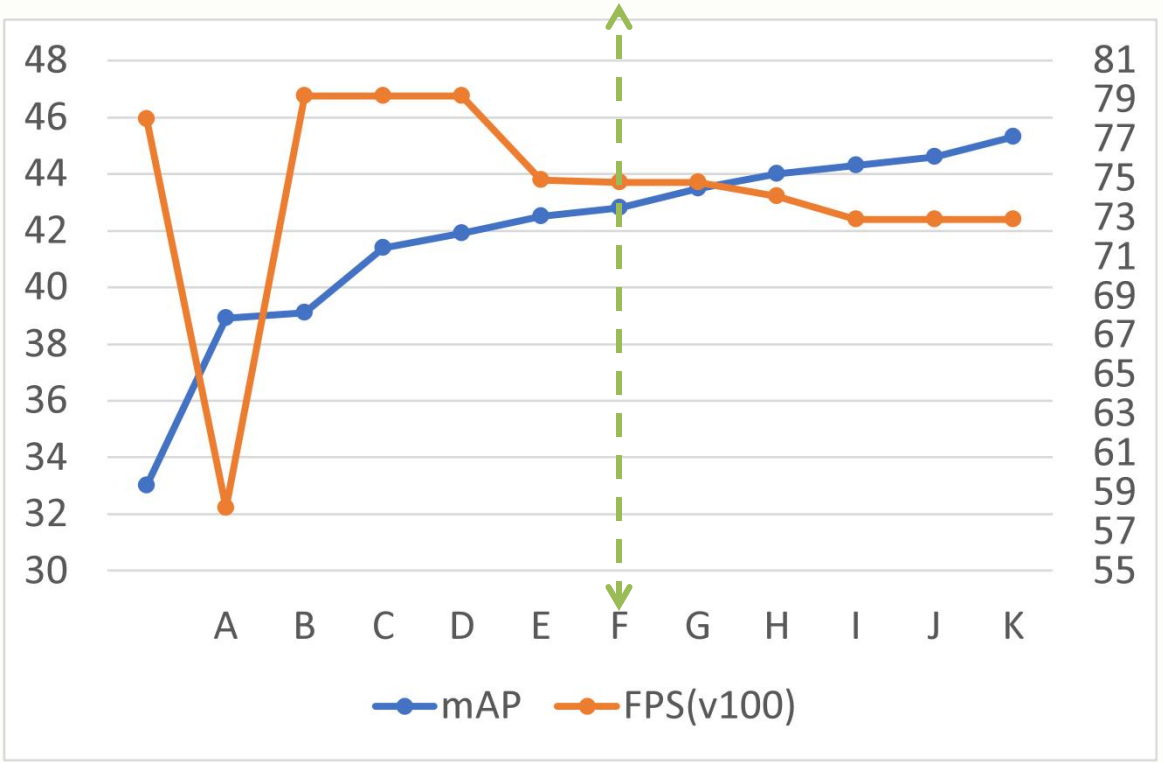
- ✓ ***COCO数据集mAP提升 ≈ 0.6%***
- ✓ ***推理速度略微降低：79.2FPS -> 74.9FPS***

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

YOLOv4: Optimal Speed
Accuracy of Object Dete

Grid Sensitive

优化点

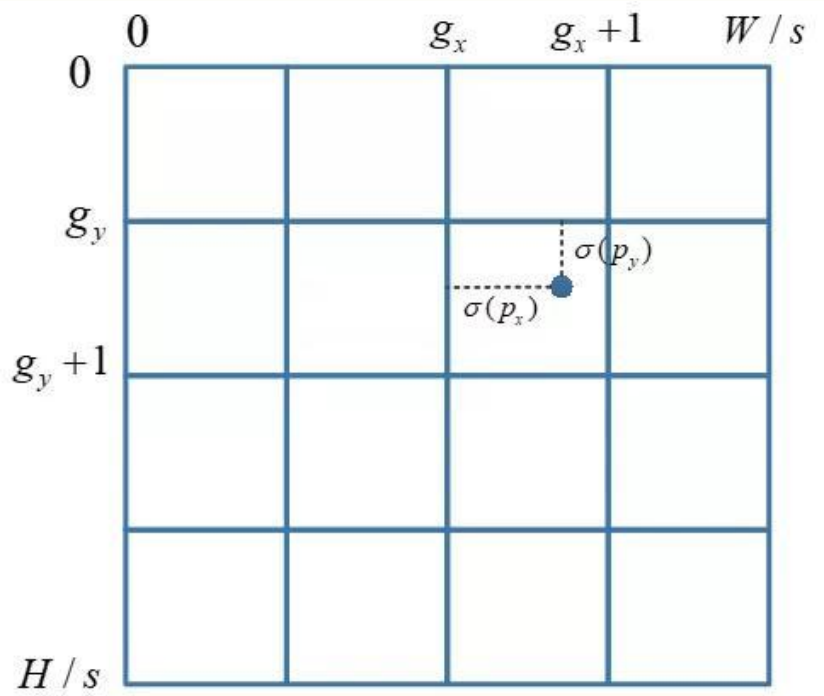
- ✓ 当真实框的中心坐标落在网格边缘时，会倾向于将logits向±∞学习(类似label smooth)，造成过拟合
- ✓ 通过引入Grid Sensitive (YOLOv4)，可以缓解这种过拟合

优化方法

- ✓ 计算中心坐标时，加上一个偏移和缩放

$$x = s \cdot \left(g_x + \sigma(p_x)\right),\ y = s \cdot (g_y + \sigma(p_y))$$

$$x = s \cdot (g_x + \alpha \cdot \sigma(p_x) - (\alpha - 1)/2)$$
$$y = s \cdot (g_y + \alpha \cdot \sigma(p_y) - (\alpha - 1)/2)$$




```
BBoxPostProcess:
  decode:
    name: YOLOBox
    conf_thresh: 0.01
    downsample_ratio: 32
    clip_bbox: true
    scale_x_y: 1.05
  nms:
```

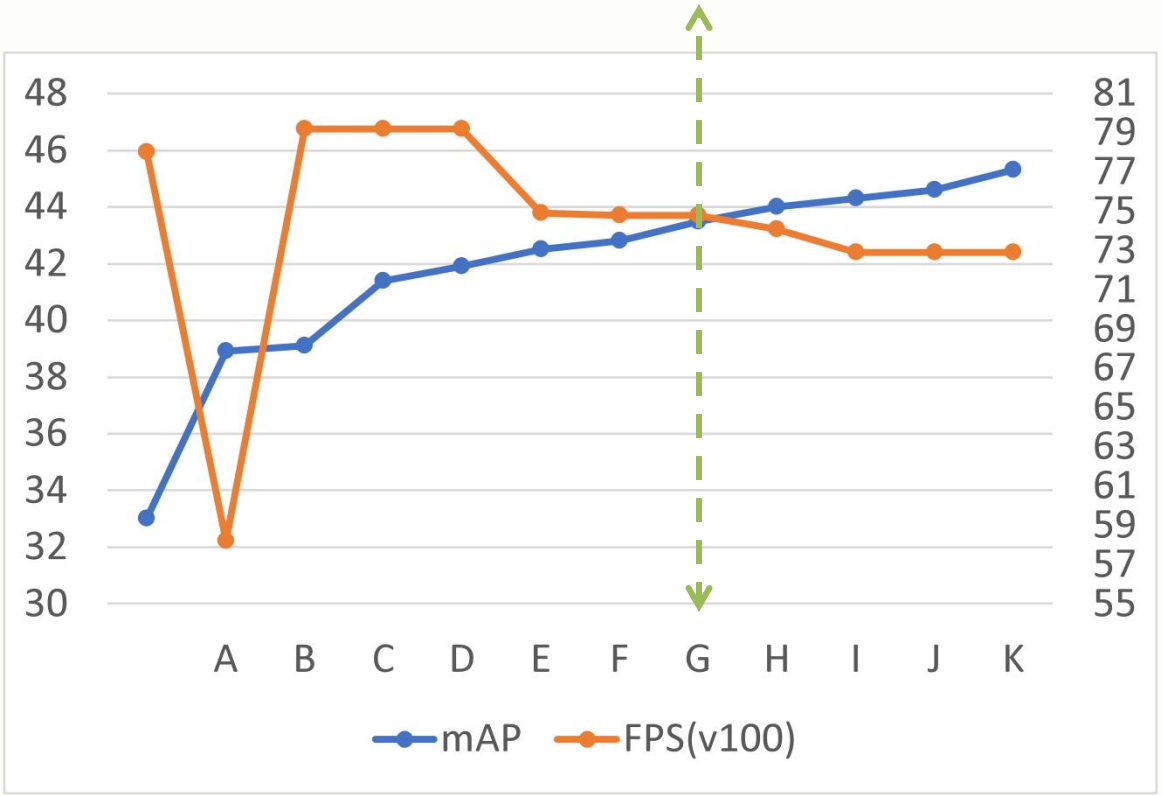
优化效果：COCO数据集mAP提升 ≈ 0.3%

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

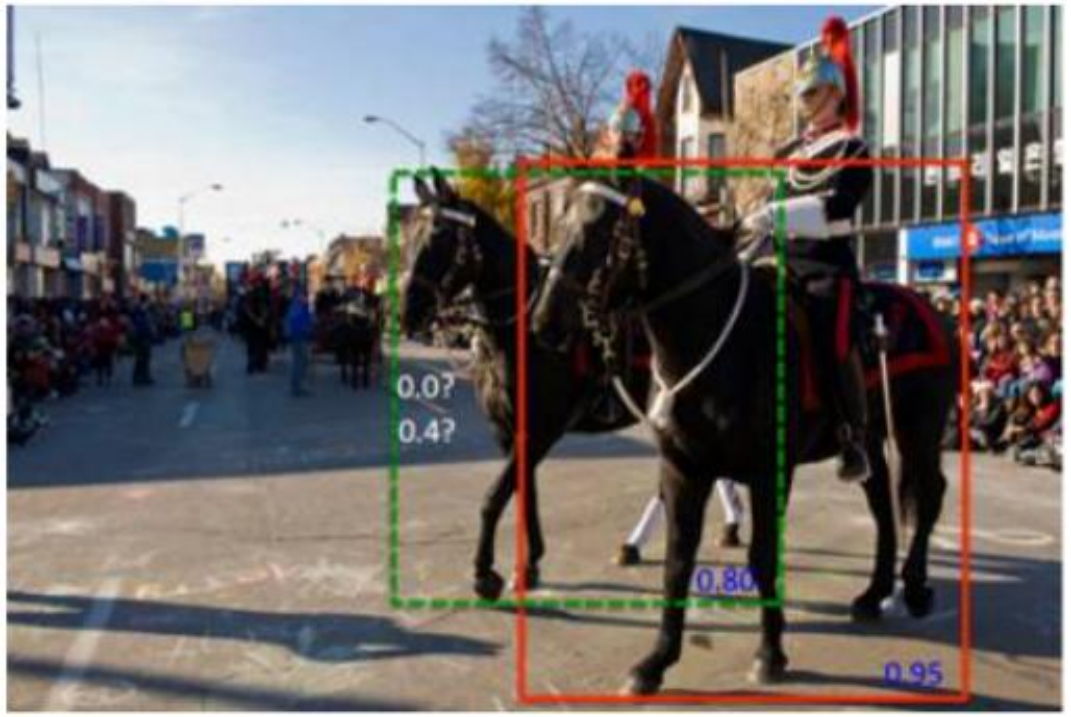
5.7 PP-YOLO

SOLOv2: Dynamic, Faster and Stronger

Matrix NMS

SoftNMS：直接抑制 -> 惩罚系数

- ✓ 传统NMS(Hard NMS)直接抑制了分数低的预测框，对于靠近的同类物体会出现误抑制的情况。对于比较接近的目标对象A和B，如果其IoU大于阈值，则第二个目标将会被丢弃。
- ✓ 通过分数惩罚系数，使RoI的排序下降，但不丢弃该目标。



优化方法

$$s_i = \begin{cases} s_i, & IoU(GT, b_i) \geq thres \\ 0, & IoU(GT, b_i) < thres \end{cases}$$

$$s_i = \begin{cases} s_i, & IoU(GT, b_i) \geq thres \\ s_i(1 - IoU(GT, b_i)), & IoU(GT, b_i) < thres \end{cases}$$

由于SoftNMS没有抑制输出，这导致了计算量的大大增大

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

SOLOv2: Dynamic and Fast Instance Segmentation

Matrix NMS

优化点

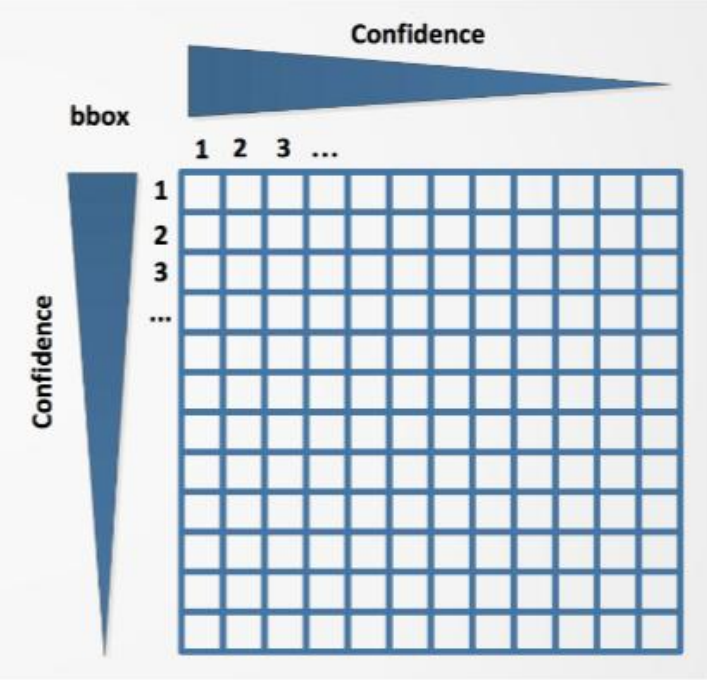
- ✓ 通过并行计算优化NMS的运行速度
- ✓ 设置抑制阈值，丢弃置信度极低的RoIs

优化方法

- ✓ 使用GPU并行计算任意两个框之间的IoU
- ✓ 对于预测框B，并行计算所有得分高于阈值的预测框与预测框B的IoU，然后根据这些IOU和得分高于阈值的预测框的被抑制概率做近似估算，估算出预测框B的抑制系数

优化效果

- ✓ ***COCO数据集mAP提升 ≈ 0.7%***
- ✓ ***推理速度不变，但后处理速度降低***



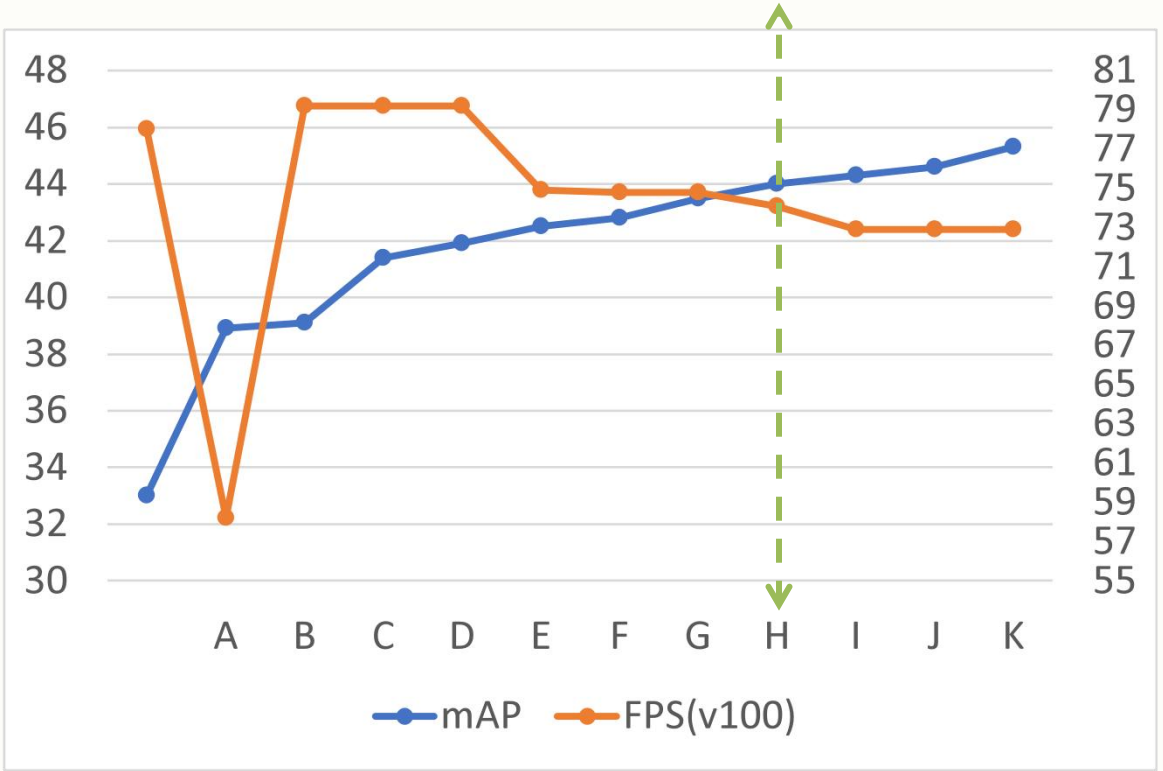

```
nms:
  name: MatrixNMS
  keep_top_k: 100
  score_threshold: 0.01
  post_threshold: 0.01
  nms_top_k: -1
  background_label: -1
```

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution

Coord Conv

优化点

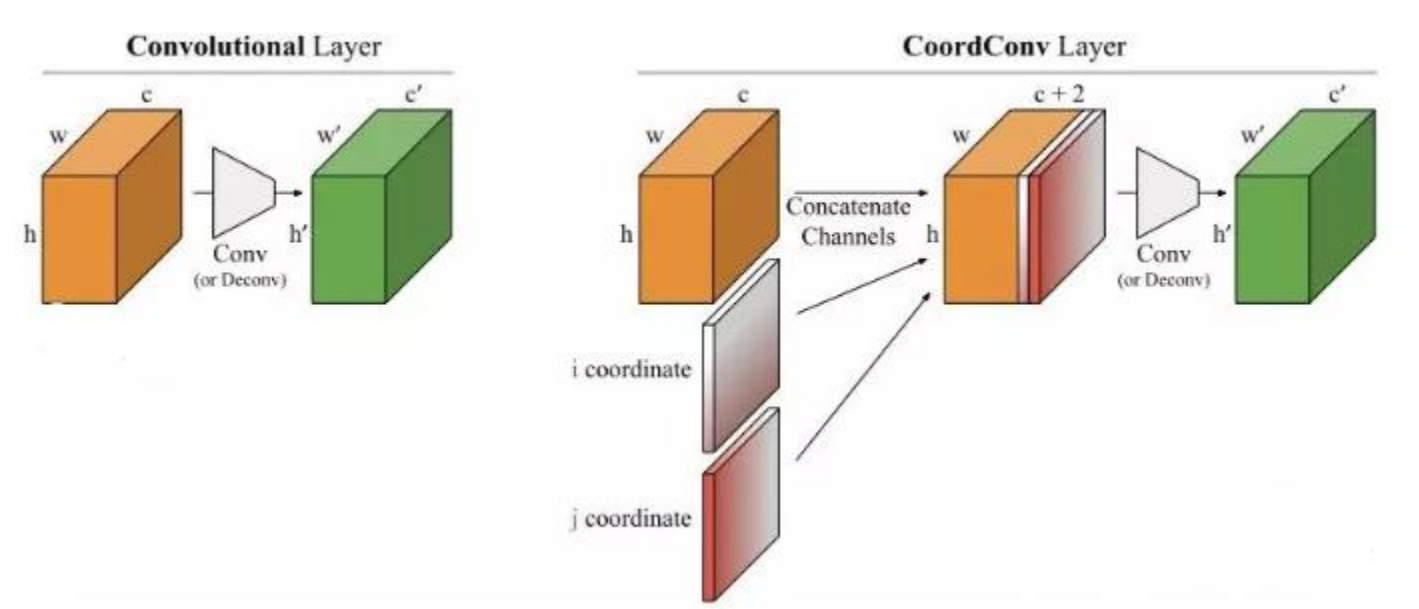
- ✓ 卷积只感知卷积核对应的感受野，与其所在的坐标没有关系
- ✓ 如果对象具有全局坐标信息，是否更具有辨识度？**灵魂提问**

优化方法

- ✓ 增加两个卷积通道，用于学习目标的坐标信息x, y

优化效果

- ✓ ***COCO数据集mAP提升 ≈ 0.5%***
- ✓ ***推理速度变化不大，降低了0.7FPS***



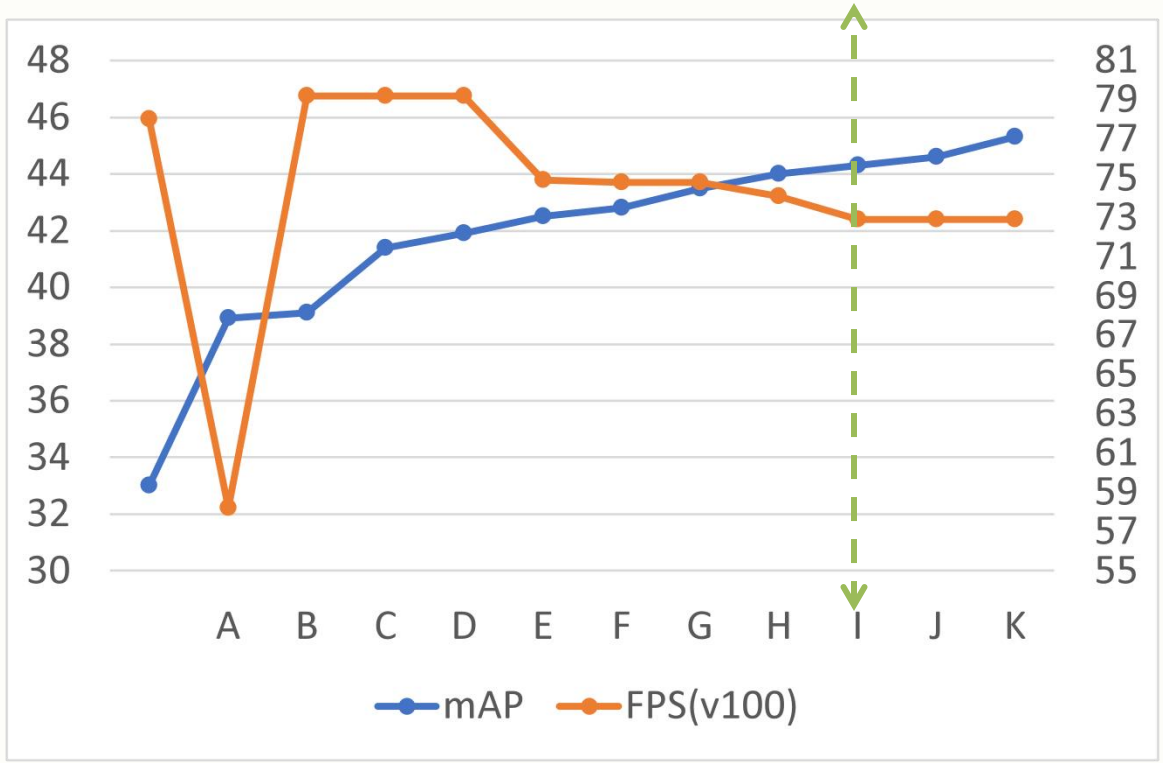

```
YOLOv3Head:
  anchor_masks: [[6, 7, 8], [3, 4, 5], [0, 1, 2]]
  anchors: [[10, 13], [16, 30], [33, 23],
            [30, 61], [62, 45], [59, 119],
            [116, 90], [156, 198], [373, 326]]
  norm_decay: 0.
  coord_conv: true
```

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

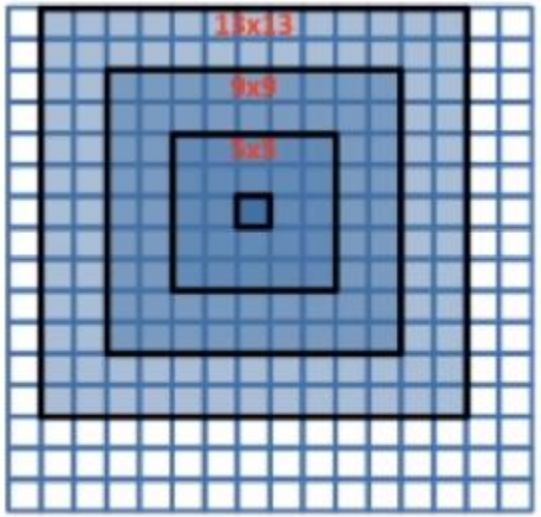
Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

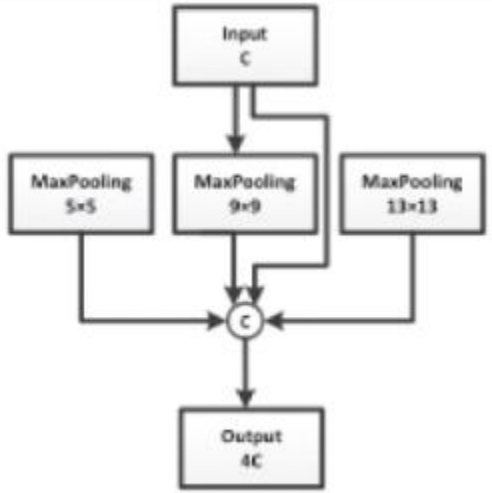
SPP

优化点

✓ 通过提取不同的池化窗口的特征，优化特征提取

优化方法






```
YOLOv3Head:
  anchor_masks: [[6, 7, 8], [3, 4, 5], [0, 1, 2]]
  anchors: [[10, 13], [16, 30], [33, 23],
            [30, 61], [62, 45], [59, 119],
            [116, 90], [156, 198], [373, 326]]
  norm_decay: 0.
  coord_conv: true
  iou_aware: true
  iou_aware_factor: 0.4
  scale_x_y: 1.05
  spp: true
  yolo_loss: YOLOv3Loss
  nms: MatrixNMS
  drop_block: true
```

优化效果

✓ *COCO数据集mAP提升 ≈ 0.5%*

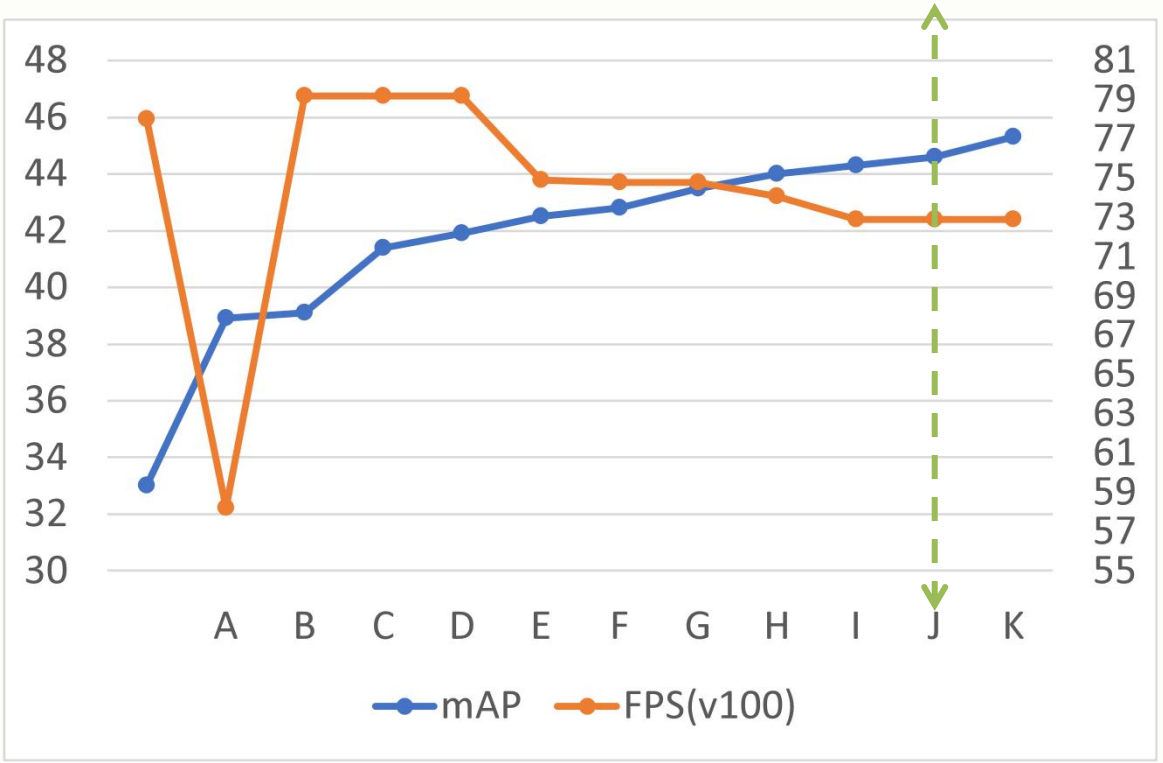
✓ *推理速度变化不大，降低了0.7FPS*

Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9

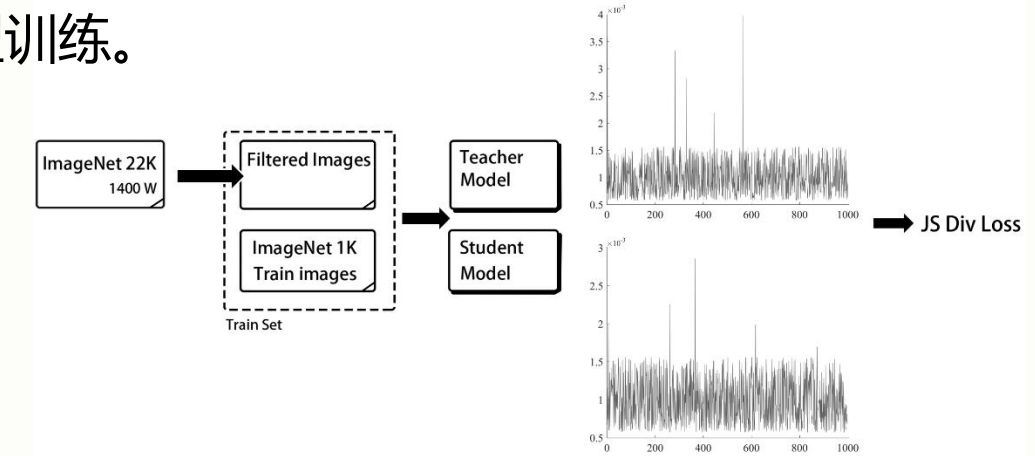


Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

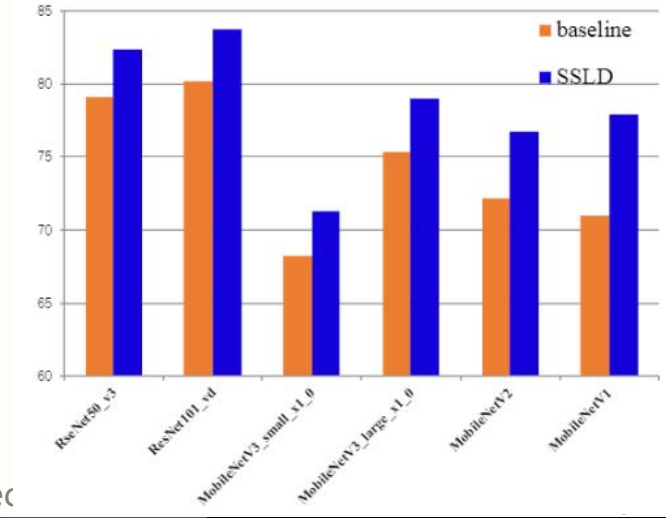
SSLD知识蒸馏

SSLD(Simple Semi-supervised Label Distillation)称为简单的半监督标签知识蒸馏，简称知识蒸馏，基本思想是利用一个精度更高的教师模型去指导目标模型训练。



蒸馏方法

- ✓ 在ImageNet22K数据集中挖掘了400W张无标签图片，与ImageNet1K组成了一个500W的蒸馏数据集
- ✓ 选择一个教师模型和学生模型组成一个新的网络，在蒸馏数据集上进行蒸馏训练。教师模型的特点是与学生模型差别不大，但精度更高。
- ✓ 将蒸馏好的学生模型在目标小数据集上进行finetune训练



优化效果

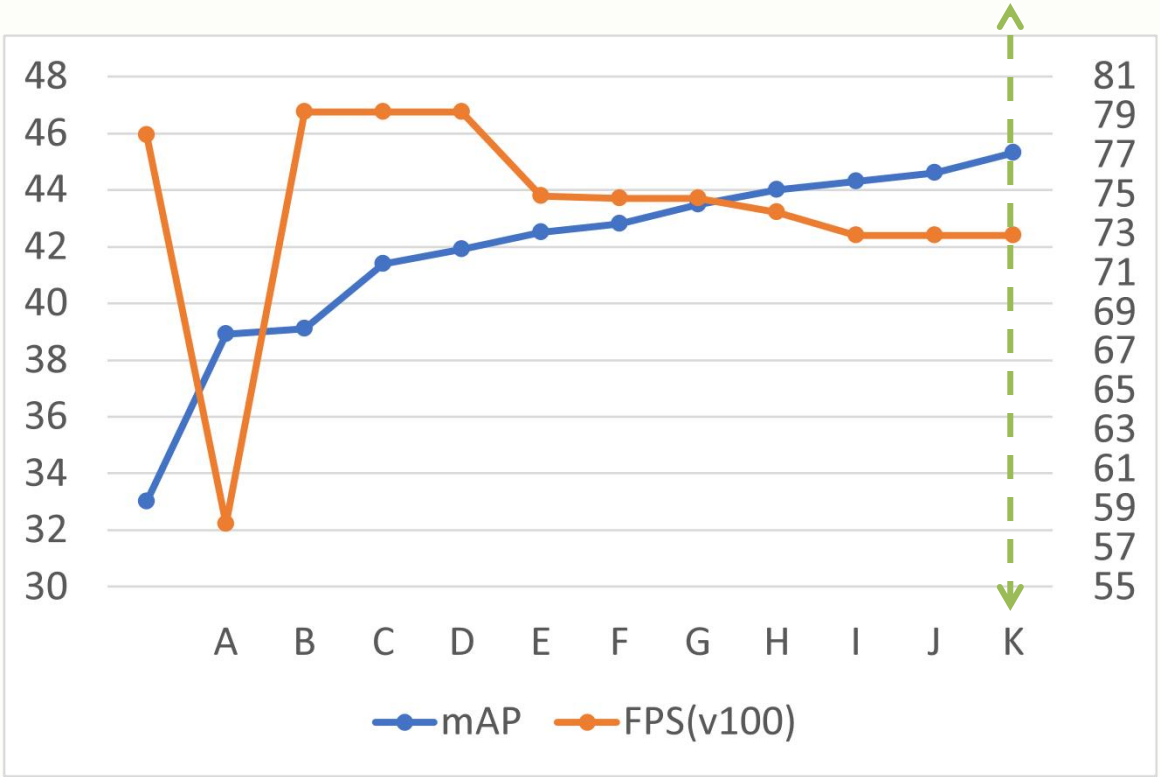
- ✓ ***COCO数据集mAP提升 ≈ 0.8%***
- ✓ ***推理速度不变***

Xiang Long, et. al. PP-YOLO: An Effec Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO提升历程（Ablation Study 消融研究）

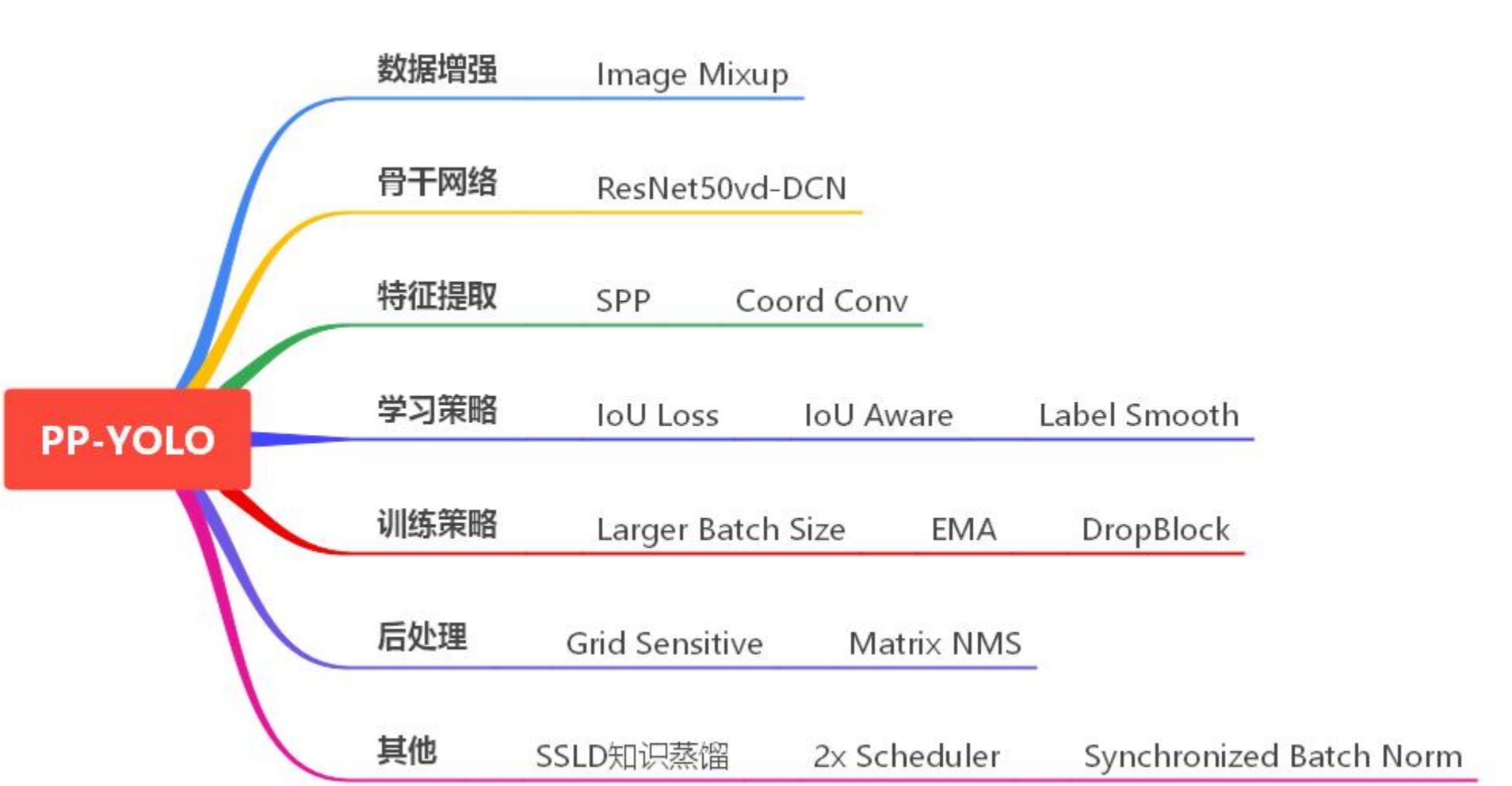
序号	模型	mAP	FPS(v100)
	YOLOv3(Darknet53)原版	33.0	78
A	YOLOv3(Darknet53)优化版	38.9	58.2
B	YOLOv3-ResNet50vd-DCN	39.1	79.2
C	B + LB + EMA + DropBlock	41.4	79.2
D	C + IoU Loss	41.9	79.2
E	D + IoU Aware	42.5	74.9
F	E + Grid Sensitive	42.8	74.8
G	F + Matrix NMS	435	74.8
H	G + Coord Conv	44.0	74.1
I	H + SPP	44.3	72.9
J	I + Better ImageNet Pretrain	44.6	72.9
K	J + 2x Scheduler	45.3	72.9



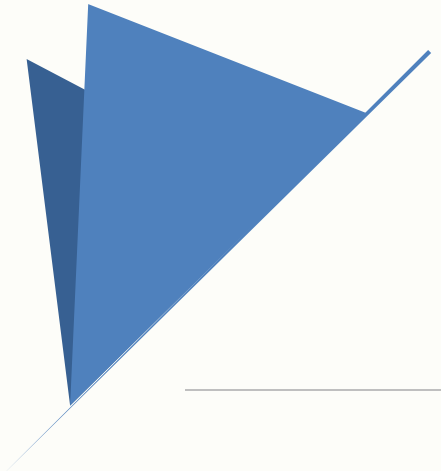
Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

5.7 PP-YOLO

PP-YOLO小结

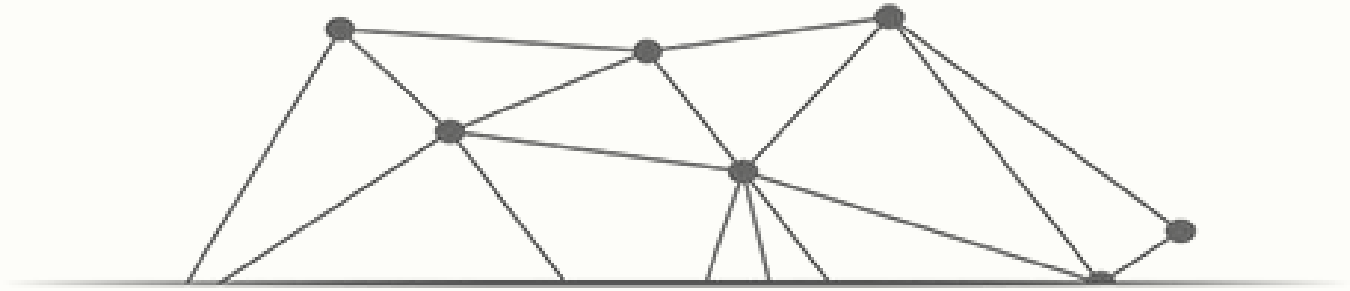


Xiang Long, et. al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. arXiv2007

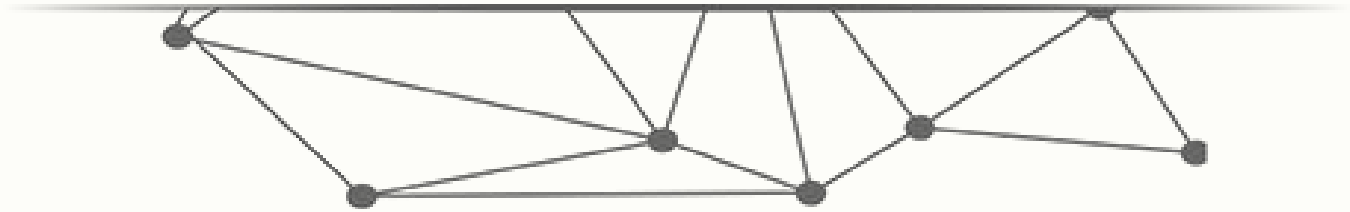


PP-YOLO v2

Come soon...
请自行学习



课堂互动 13.2.10



5. 基于单阶段方法的目标检测

作业07： 基于Yolo系列模型的电路板故障检测

目标：印刷电路板(PCB)的瑕疵检测

PCB数据集是一个公共的合成PCB数据集，由北京大学发布，用于检测、分类和配准任务。其中包含1368张图像以及6中缺陷（缺失孔，鼠标咬伤，开路，短路，杂散，伪铜）。本项目选择了适用于检测任务的693张图像，随机选择593张图像作为训练集，100张图像作为验证集。

作业描述：

基于PaddleDetection中的**Yolo系列**算法，完成印刷电路板(PCB)瑕疵数据集的训练与评估任务，要求在验证集上**$mAP_{0.5}$**达到**0.7**以上。

Part 06

基于Anchor-Free的目标检测

/ Anchor Free的基本概念

/ CornerNet

/ FCOS

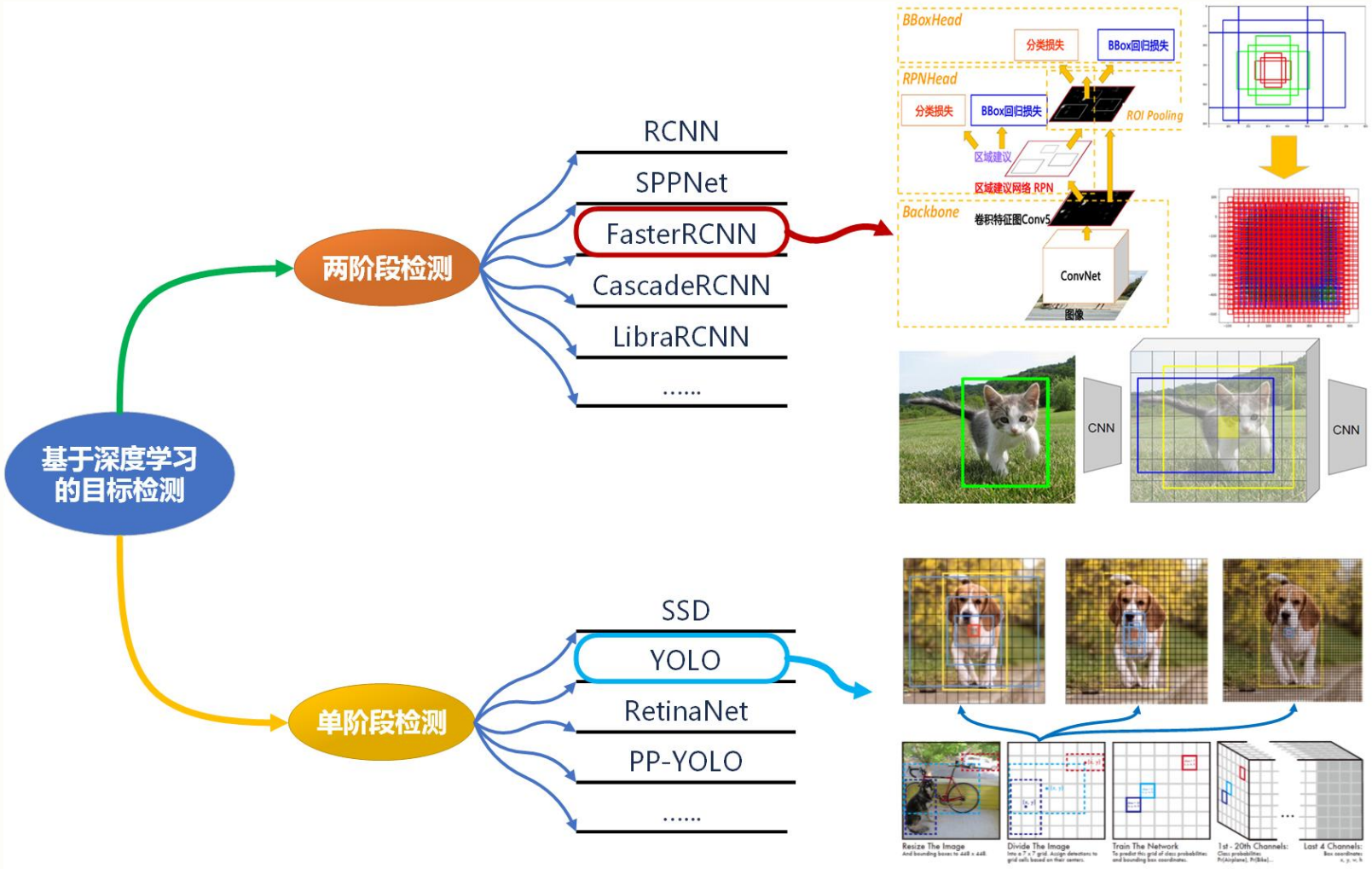
/ CenterNet

6. 基于Anchor-Free的目标检测

Anchor-Based方法回顾

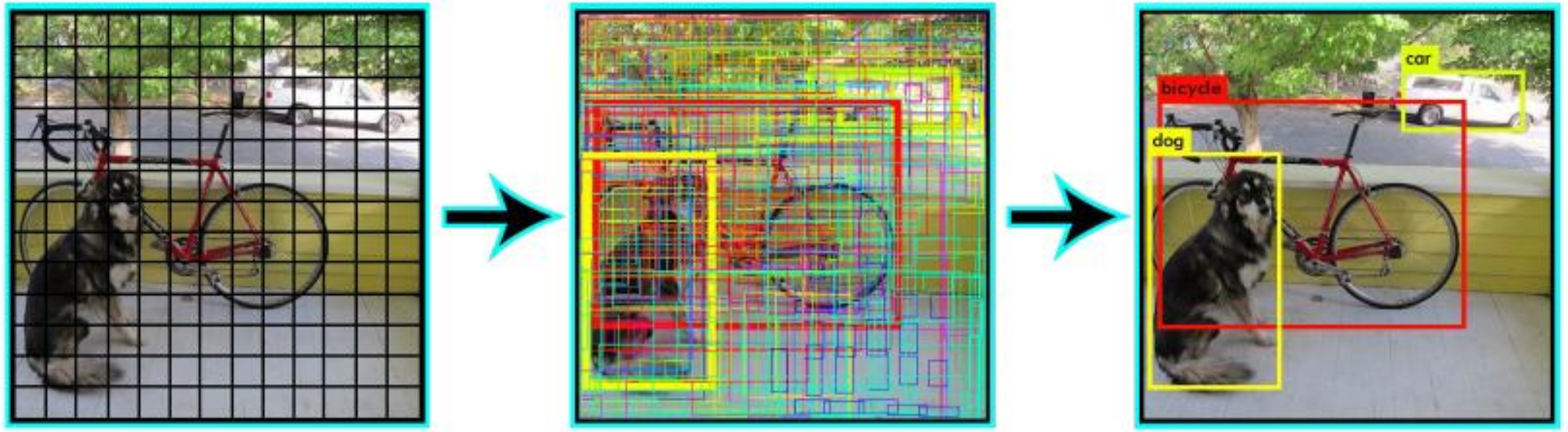
基于Anchor的目标检测

- **Anchor**是两阶段和单阶段检测算法的基础
- **两阶段(Two-stage)算法**使用RPN等方法生成anchor，是一种坐标和类别分离的anchor生成方法
- **单阶段(One-stage)算法**将坐标和类别都作为anchor的属性，进行统一预测



6. 基于Anchor-Free的目标检测

Anchor是否是必须的?



基于Anchor算法的基本过程:

- 在特征图的每个像素上生成多个不同大小和比例的候选框
- 对候选框进行初步筛选生成RoIs
- 对RoIs进行分类和回归，再根据置信度等实现最终的检测

6. 基于Anchor-Free的目标检测

Anchor-Based方法的缺点

弊端?

Anchor-Based

01 海量的锚框使系统具有较高的计算量和内存占用，同时影响了系统的效率，无法实现实时推理

02 检测性能对于锚框的尺寸、长宽比、数量非常敏感，这些超参数调节困难，过于依赖经验

03 锚框固定的尺寸、长宽比会导致处理形变较大的候选对象时比较困难，尤其是小目标

04 为了提高召回率，需要在图像上放置密集的锚框；然而过多的锚框会导致正负样本严重不平衡

05 预先定义的锚框限制了检测器的泛化能力，针对不同对象类型的样本需要进行专门设计

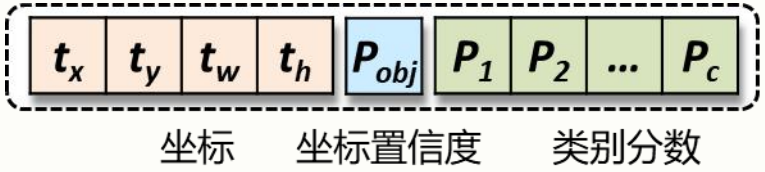
Anchor

抛弃Anchor?

6. 基于Anchor-Free的目标检测

如果去掉Anchor后该如何表示检测框?

Question: 在Anchor-Based的方法中，是如何表示检测框的?



Answer： Anchor和其对应的编码信息

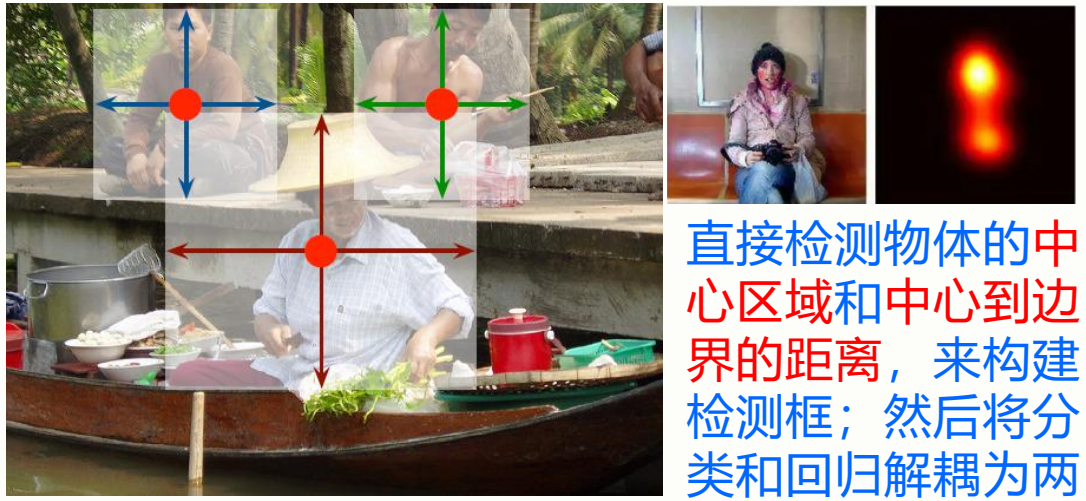
Question：在Anchor-Free中，没有Anchor怎么办呢?

基于关键点的检测算法

先检测目标左上角和右下角的**角点**，再通过角点组合成检测框。

VS

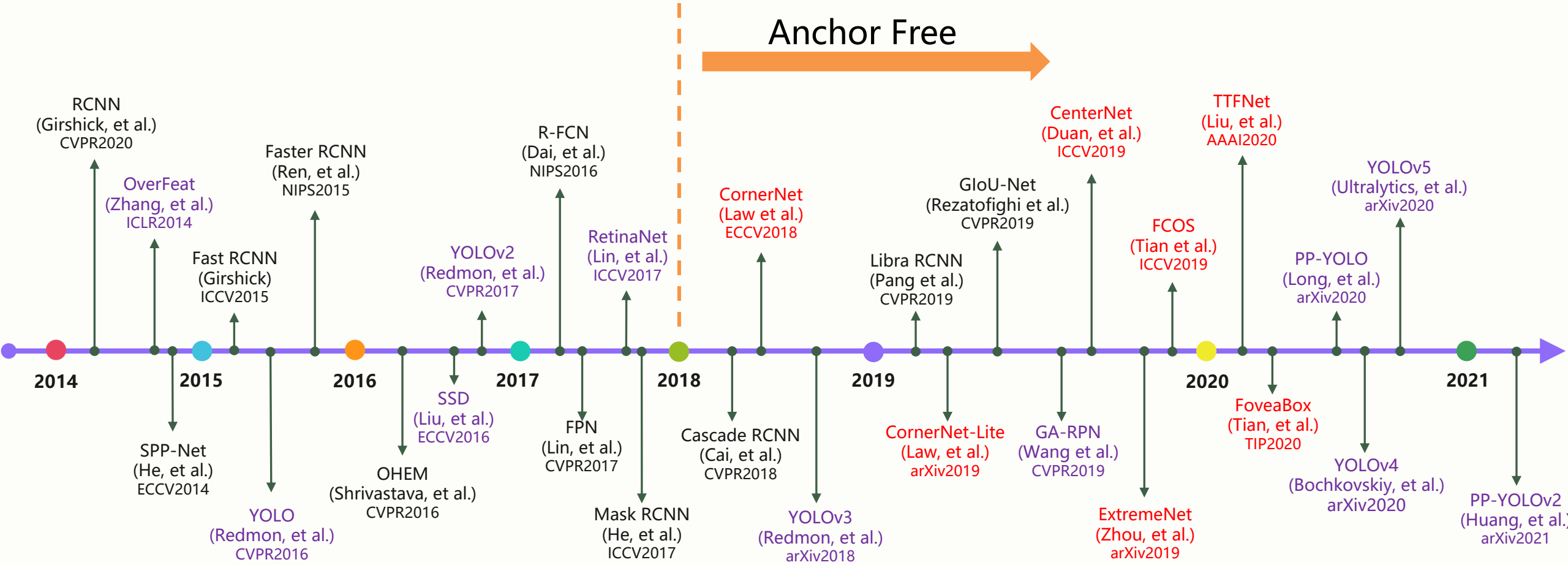
基于中心点的检测算法



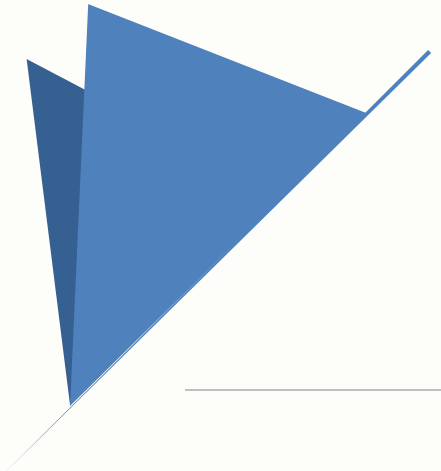
直接检测物体的中心区域和中心到边界的距离，来构建检测框；然后将分类和回归解耦为两个子网络。

6. 基于Anchor-Free的目标检测

Anchor Free系列算法的历史



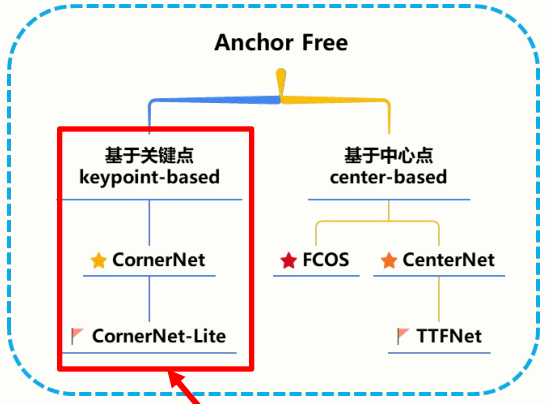
两阶段目标检测算法
单阶段目标检测算法
AnchorFree目标检测算法



CornerNet

6.1 CornerNet

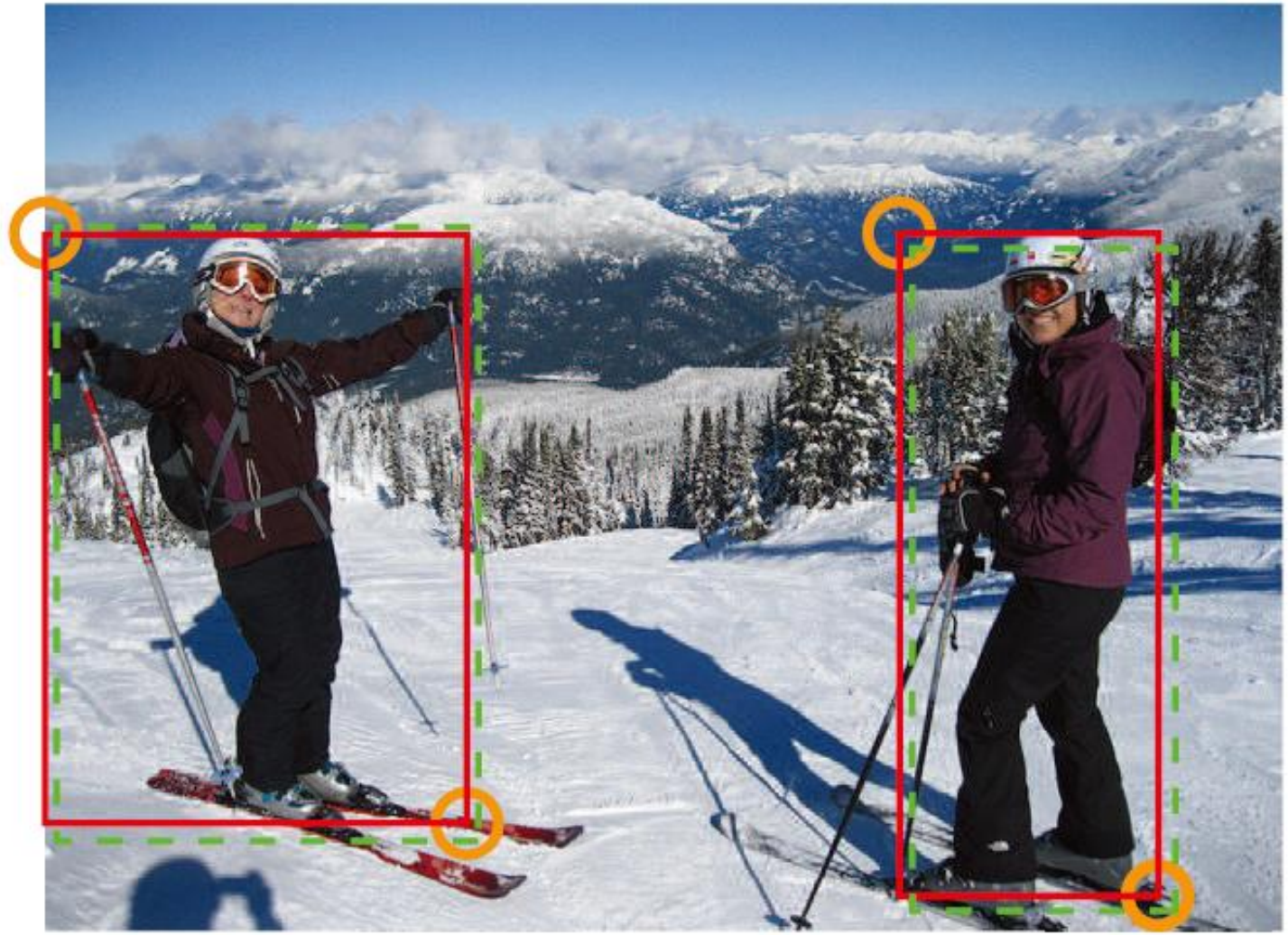
基于关键点的Anchor Free检测算法



CornerNet: Detecting Objects as Paired Keypoints

Hei Law[0000−0003−1009−164X], Jia Deng[0000−0001−9594−4554]

University of Michigan, Ann Arbor
{heilaw,jiadeng}@umich.edu

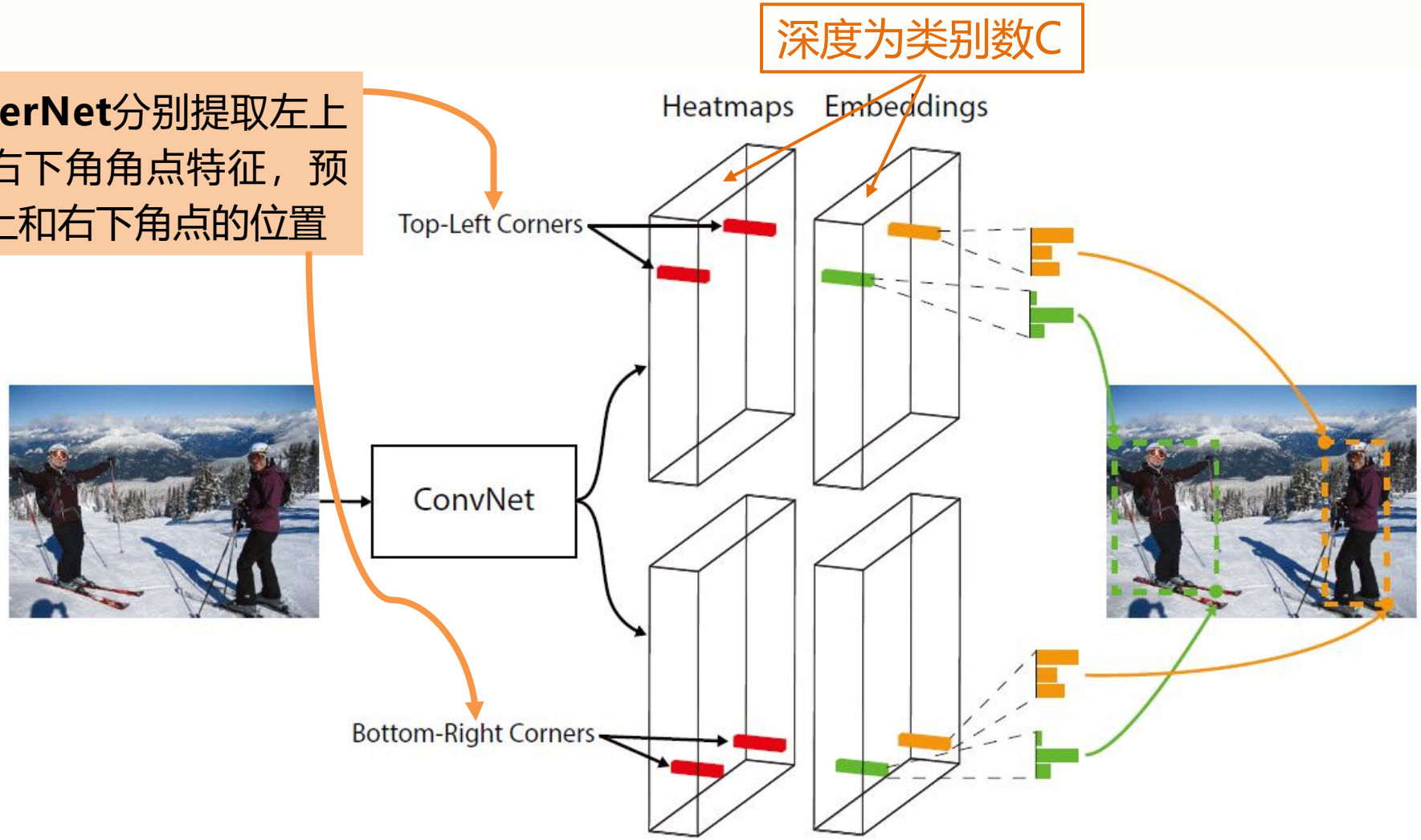


Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

CornerNet的核心思想

CornerNet分别提取左上角和右下角角点特征，预测左上和右下角点的位置



Heatmaps

预测哪些点最有可能是角点

Embeddings

表征属于相同对象的角点的相似度

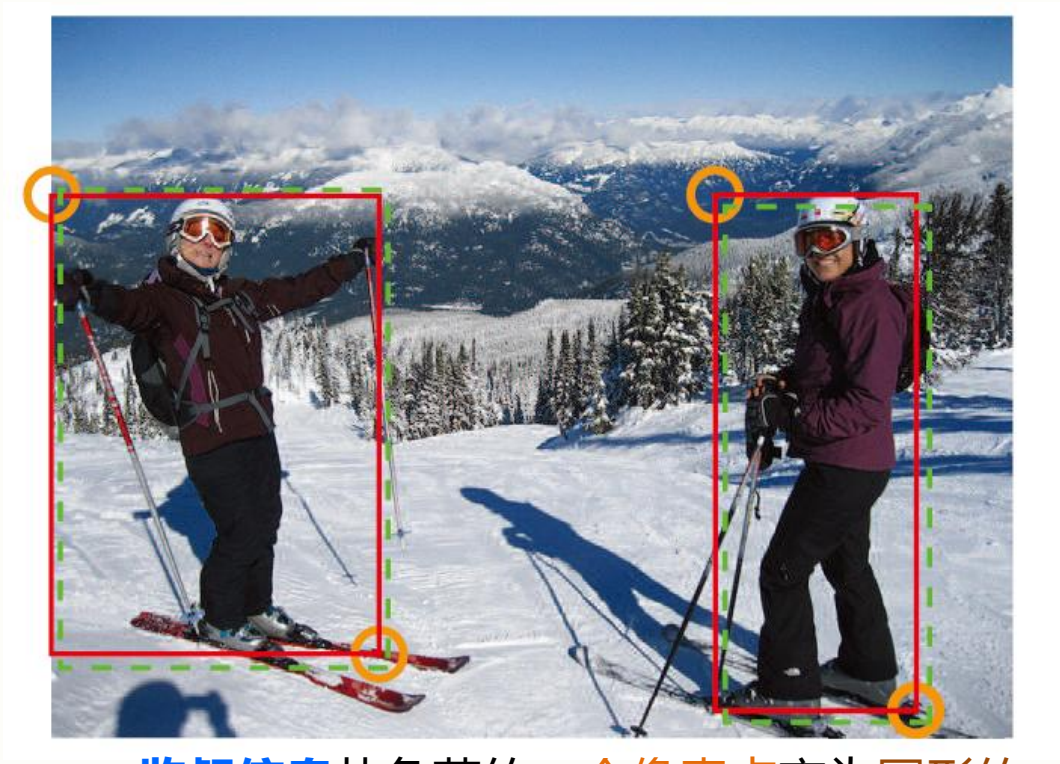
- 相同物体，embeddings具有较高相似度
- 不同物体，embeddings具有较大的距离

Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

使用Heatmaps寻找角点Corner

降低惩罚 (Reduce penalty)



监督信息从角落的一个像素点变为圆形的高斯区域，越接近圆形中心值越接近1，越接近圆形边缘值越接近0。

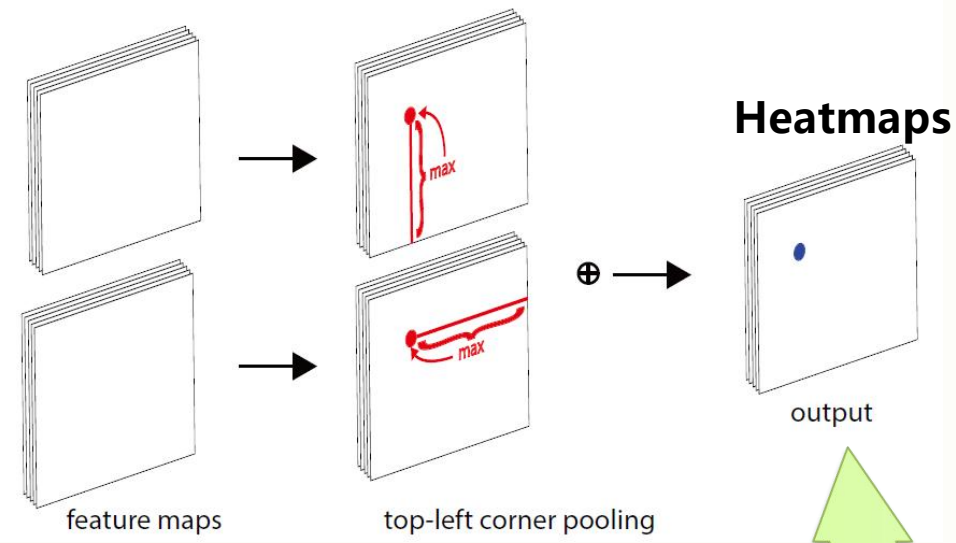
角点池化 (Corner Pooling)

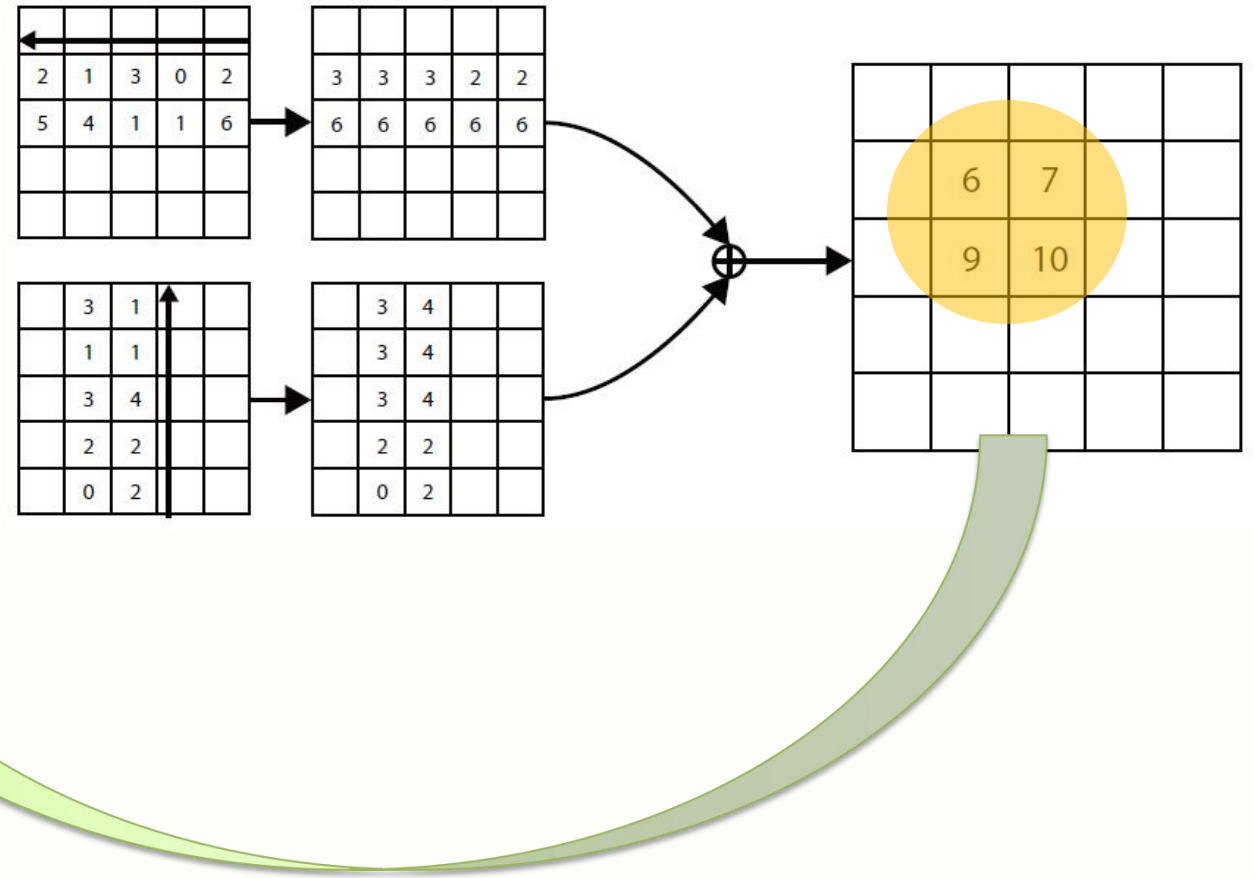
从左下角和右上角开始往左上角的角点进行池化，两个方向的池化结果进行叠加后，可获得左上角的角点。

Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

Corner pooling算法

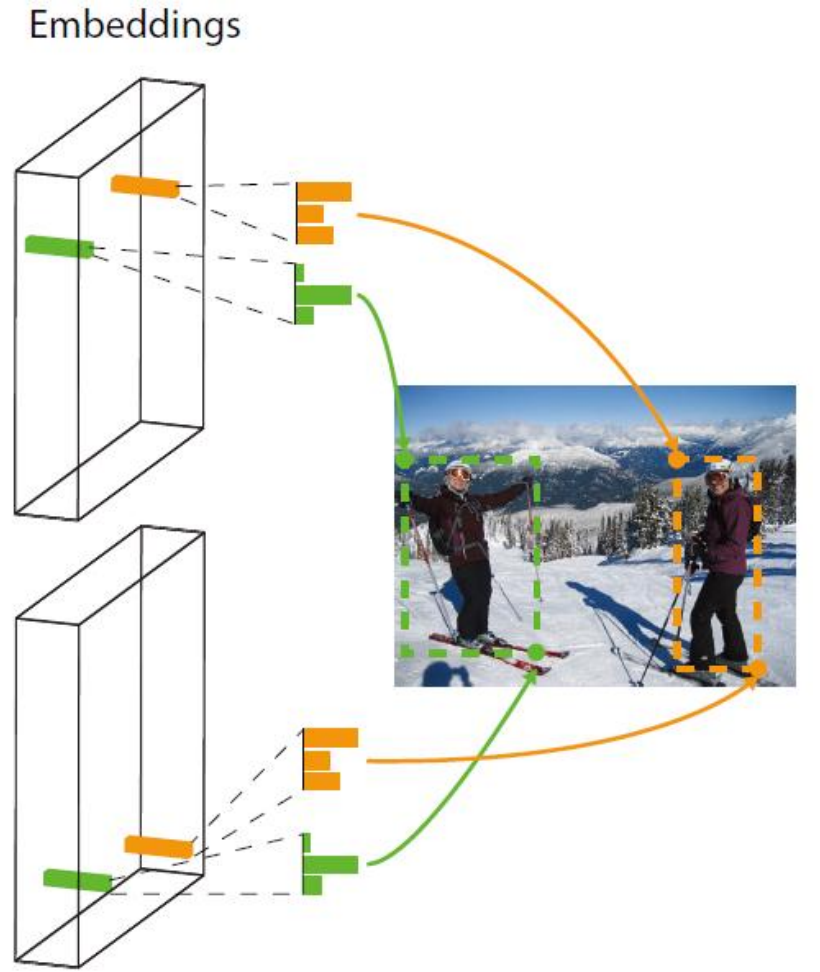




Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

使用Embeddings来组合同一目标的角点



期望的结果

✓ 同一目标的一对角点相似度较高（距离较小）

✓ 不同目标的一对角点相似度较低（距离较大）

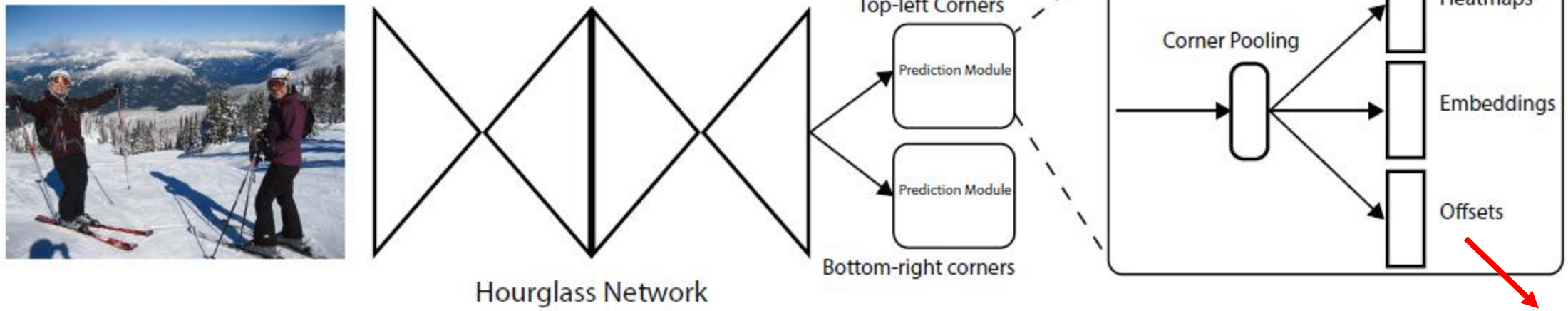
Embeddings

1. 对特征图上的每个点进行编码，得到的一维编码向量即为embedding特征，表示该点的位置信息，维度为 [N, 1, H, W]
2. 左上角点和右下角点的embeddings距离即为两个角点的相似度。

Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

CornerNet的网络结构



角点经过降采样后的坐标与原始坐标之间的误差

Loss

- **Heatmaps**

$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1-p_{cij})^{\alpha} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1-y_{cij})^{\beta} (p_{cij})^{\alpha} \log(1-p_{cij}) & \text{otherwise} \end{cases}$$

- **Embeddings**

$$L_{pull} = \frac{1}{N} \sum_{k=1}^{N} \left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2 \right],$$

$$L_{push} = \frac{1}{N(N-1)} \sum_{k=1}^{N} \sum_{\substack{j=1 \\ j \neq k}} \max(0, \Delta - |e_k - e_j|),$$

距离较大时，Loss_{push}=0

- **Offsets**

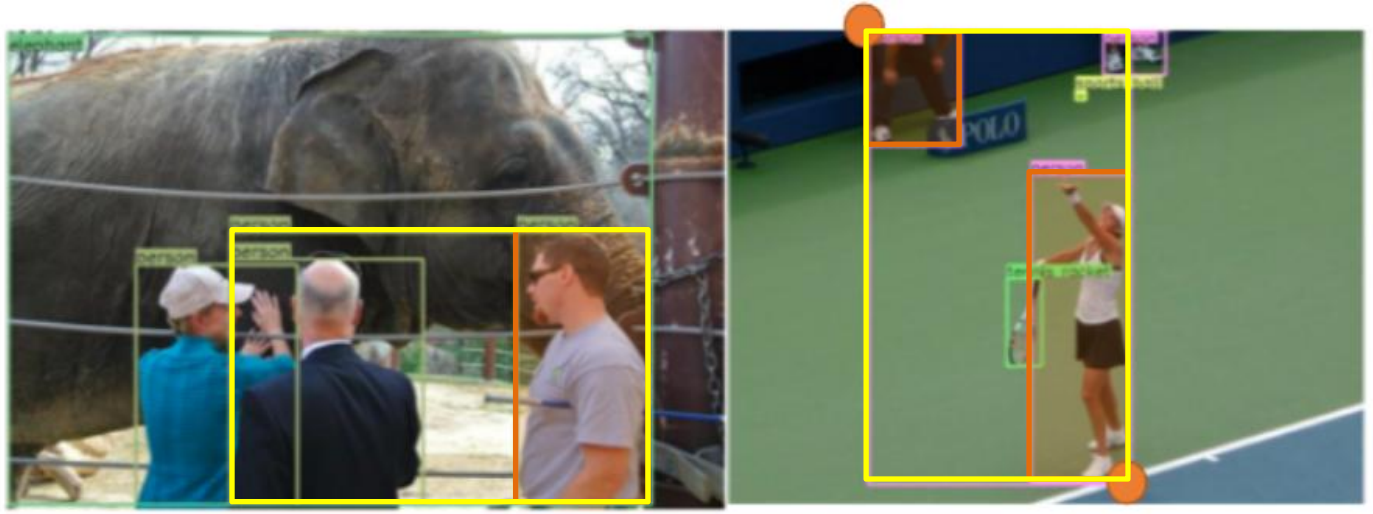
$$L_{off} = \frac{1}{N} \sum_{k=1}^{N} \text{SmoothL1Loss}(o_k, \hat{o}_k)$$

Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

6.1 CornerNet

CornerNet的检测结果

Method	Backbone	AP	AP^{50}	AP^{75}
Two-stage detectors				
DeNet [39]	ResNet-101	33.8	53.4	36.1
CoupleNet [46]	ResNet-101	34.4	54.8	37.2
Faster R-CNN by G-RMI [16]	Inception-ResNet-v2 [38]	34.7	55.5	36.7
Faster R-CNN+++ [15]	ResNet-101	34.9	55.7	37.4
Faster R-CNN w/ FPN [22]	ResNet-101	36.2	59.1	39.0
Faster R-CNN w/ TDM [35]	Inception-ResNet-v2	36.8	57.7	39.2
D-FCN [7]	Aligned-Inception-ResNet	37.5	58.0	-
Regionlets [43]	ResNet-101	39.3	59.8	-
Mask R-CNN [13]	ResNeXt-101	39.8	62.3	43.4
Soft-NMS [2]	Aligned-Inception-ResNet	40.9	62.8	-
LH R-CNN [21]	ResNet-101	41.5	-	-
Fitness-NMS [40]	ResNet-101	41.8	60.9	44.9
Cascade R-CNN [4]	ResNet-101	42.8	62.1	46.3
D-RFCN + SNIP [37]	DPN-98 [5]	45.7	67.3	51.1
One-stage detectors				
YOLOv2 [31]	DarkNet-19	21.6	44.0	19.2
DSOD300 [33]	DS/64-192-48-1	29.3	47.3	30.6
GRP-DSOD320 [34]	DS/64-192-48-1	30.0	47.9	31.8
SSD513 [25]	ResNet-101	31.2	50.4	33.3
DSSD513 [10]	ResNet-101	33.2	53.3	35.2
RefineDet512 (single scale) [45]	ResNet-101	36.4	57.5	39.5
RetinaNet800 [23]	ResNet-101	39.1	59.1	42.3
RefineDet512 (multi scale) [45]	ResNet-101	41.8	62.9	45.7
CornerNet511 (single scale)	Hourglass-104	40.5	56.5	43.1
CornerNet511 (multi scale)	Hourglass-104	42.1	57.8	45.3



效果解读

1. 精度媲美Anchor-based方法
2. 相同类别的不同对象容易形成误检框
3. 基于Hourglass104的Backbone在检测速度方面有较大的提升空间

Hei Law, Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV2018.

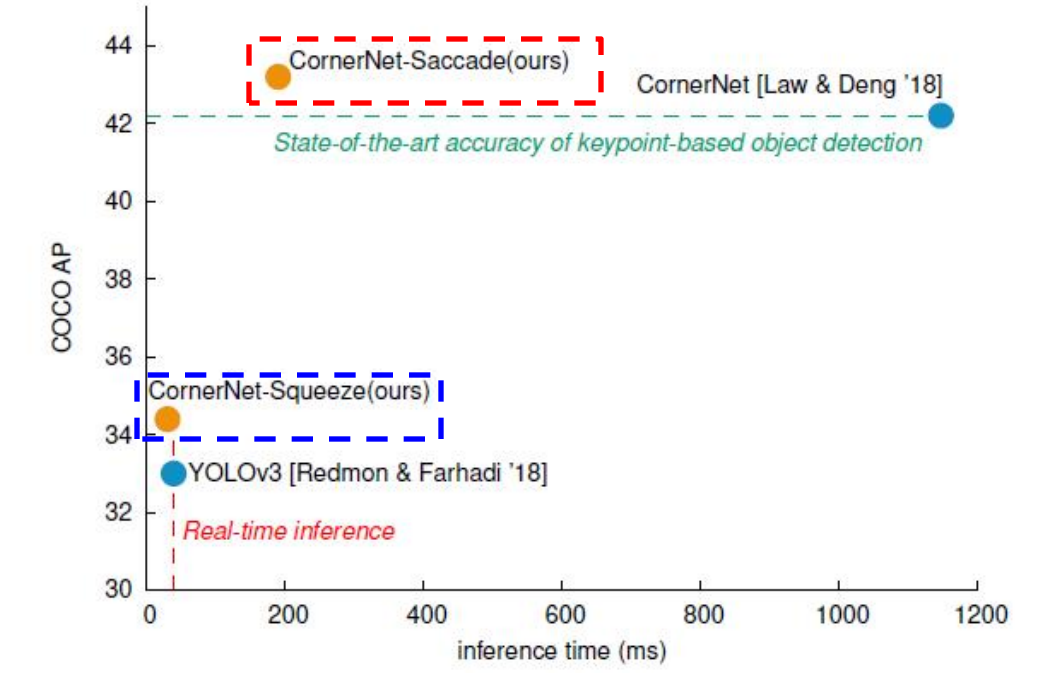
6.1 CornerNet

CornerNet 进化版

CornerNet-Lite: Efficient Keypoint-Based Object Detection

Hei Law
heilaw@cs.princeton.edu
Yun Tang

Department of Computer Science
Princeton University
Princeton, NJ USA



	Time	AP	AP^s	AP^m	AP^l
YOLOv3	39ms	33.0	18.3	35.4	41.9
CornerNet-Squeeze	30ms	34.4	13.7	36.5	47.4
CornerNet (single)	211ms	40.6	19.1	42.8	54.3
CornerNet (multi)	1147ms	42.2	20.7	44.8	56.6
CornerNet-Saccade	190ms	43.2	24.4	44.6	57.3

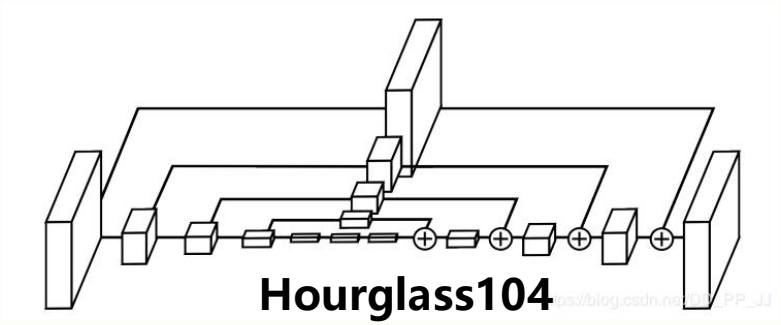
Table 6: CornerNet-Lite versus CornerNet and YOLOv3 on COCO test set.

Comparisons with YOLO and CornerNet. We also compare CornerNet-Lite with CornerNet and YOLOv3 on COCO test set in Tab. 6. CornerNet-Squeeze is faster and more accurate than YOLOv3. CornerNet-Saccade is more accurate than CornerNet at multi-scales and 6 times faster.

H Law，Y Teng，O Russakovsky，J Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV 2018.

6.1 CornerNet

CornerNet Squeeze



- 减少每个像素的处理过程
- 结合SqueezeNet及MobileNet的设计思想

Input	Operator	Output
Residual block in CornerNet		
$h \times w \times k$	3×3 Conv, ReLU	$h \times w \times k'$
$h \times w \times k'$	3×3 Conv, ReLU	$h \times w \times k'$
Fire module in CornerNet-Squeeze		
$h \times w \times k$	1×1 Conv	$h \times w \times \frac{k'}{2}$
$h \times w \times \frac{k'}{2}$	1×1 Conv + 3×3 Dwise, ReLU	$h \times w \times k'$

Table 1: Comparison between the residual block and the new fire module.

可分离卷积

```
def fire_block(x, out_dim, sr=2, stride=1, name=None):
    conv1 = _conv_norm(x, 1, out_dim // sr, ind=1, name=name)
    conv_1x1 = fluid.layers.conv2d(
        conv1,
        filter_size=1,
        num_filters=out_dim // 2,
        stride=stride,
        param_attr=ParamAttr(
            name=name + "_conv_1x1_weight", initializer=kaiming_init(conv1, 1)),
        bias_attr=False,
        name=name + '_conv_1x1')
    conv_3x3 = fluid.layers.conv2d(
        conv1,
        filter_size=3,
        num_filters=out_dim // 2,
        stride=stride,
        padding=1,
        groups=out_dim // sr,
        param_attr=ParamAttr(
            name=name + "_conv_3x3_weight", initializer=kaiming_init(conv1, 3)),
        bias_attr=False,
        name=name + '_conv_3x3',
        use_cudnn=False)
    conv2 = fluid.layers.concat(
        [conv_1x1, conv_3x3], axis=1, name=name + '_conv2')
    pattr = ParamAttr(name=name + '_bn2_weight')
    battr = ParamAttr(name=name + '_bn2_bias')

    bn2 = fluid.layers.batch_norm(
        input=conv2,
        name=name + '_bn2',
        param_attr=pattr,
        bias_attr=battr,
        moving_mean_name=name + '_bn2_running_mean',
        moving_variance_name=name + '_bn2_running_var')
```

Y Teng, CornerNet: Detecting Objects as Paired Keypoints, ECCV 2018.

6.1 CornerNet

CornerNet Squeeze 实验结果

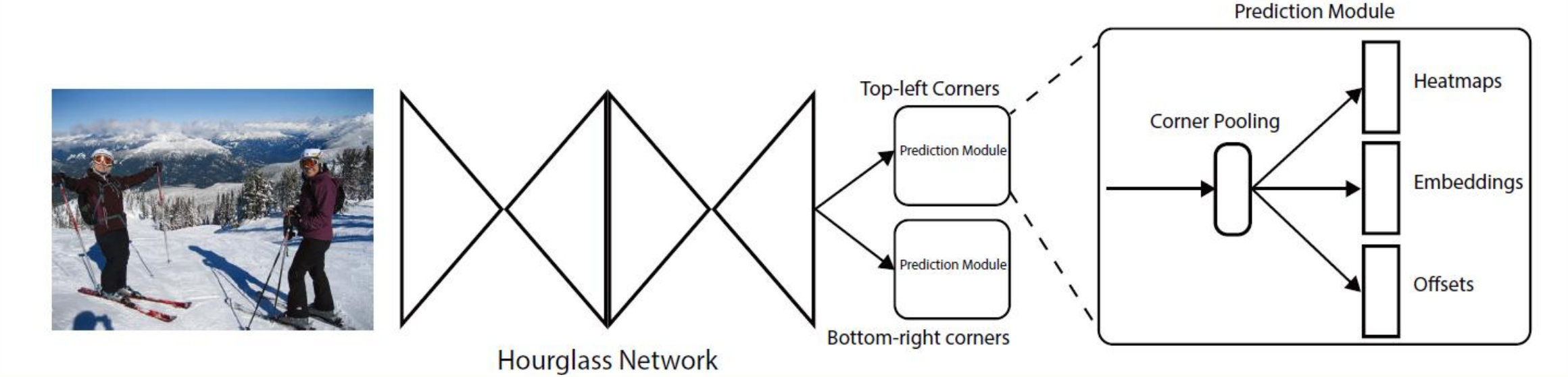
网络结构	骨干网络	图片数量/GPU	mAP	FPS*
CornerNet-Squeeze(paper)	ResNet50	14	34.8	35.5
CornerNet-Squeeze(paddle)	ResNet50-vd	14	32.7	47.01
CornerNet-Squeeze-dcn(paddle)	ResNet50-vd	14	**34.9**	**40.43**
CornerNet-Squeeze-dcn-mixup-cosine*(paddle)	ResNet50-vd	14	**38.2**	**39.70**

*CornerNet-Squeeze-dcn-mixup-consine是基于原版CornerNet-Squeeze优化效果最好的模型，在ResNet50-vd骨干网络基础上增加了mixup预处理、可变形卷积和cosine_decay的学习率策略。

H Law，Y Teng，O Russakovsky，J Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV 2018.

6.1 CornerNet

CornerNet 算法小结



Heatmaps

- ✓ 解决如何找到cornet的问题
- ✓ 核心算法：Reduce penalty, Corner Pooling

Embeddings

- ✓ 解决如何组合角点为检测框的问题
- ✓ 同一目标的一对角点相似度较高
- ✓ 不同目标的一对角点距离较大

核心思想

通过一对角点实现目标检测

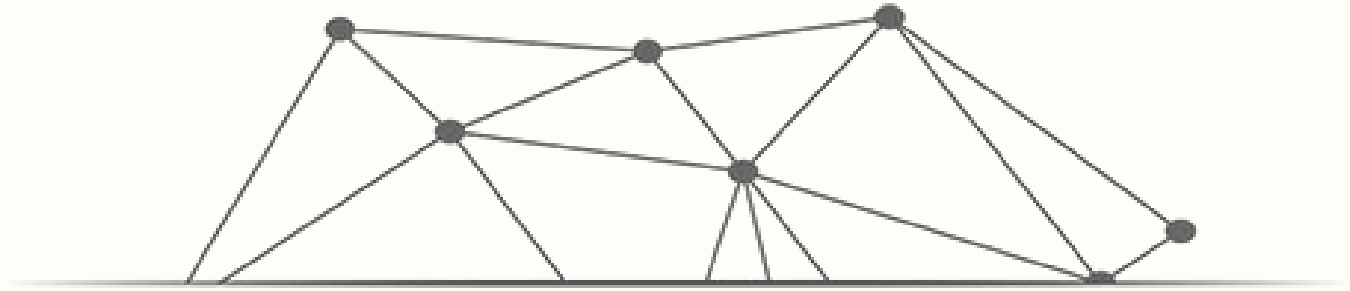
损失函数

Heatmaps+Embeddings+offsets

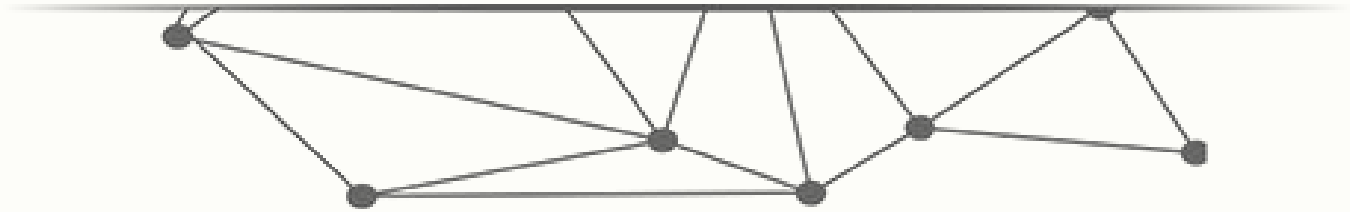
CornerNet进化版：

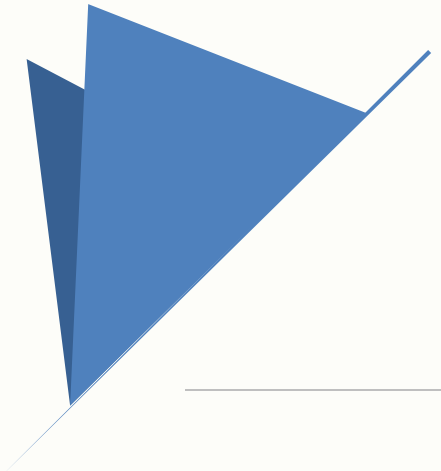
CornetNet Squeeze减少每个像素点的处理过程

H Law，Y Teng，O Russakovsky，J Deng. CornerNet: Detecting Objects as Paired Keypoints. ECCV 2018.



课堂互动 13.2.11



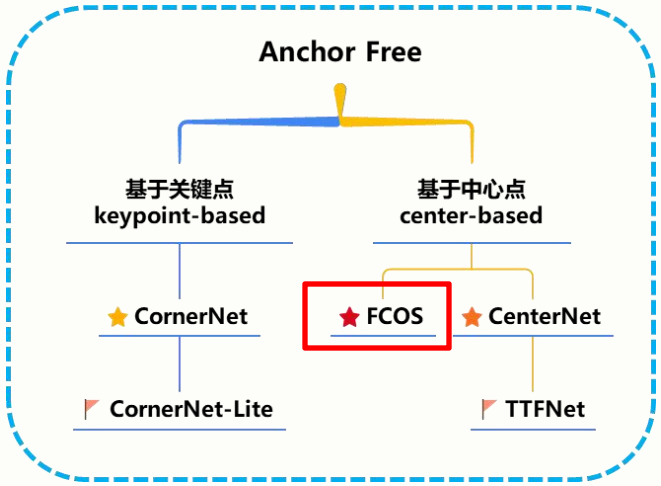


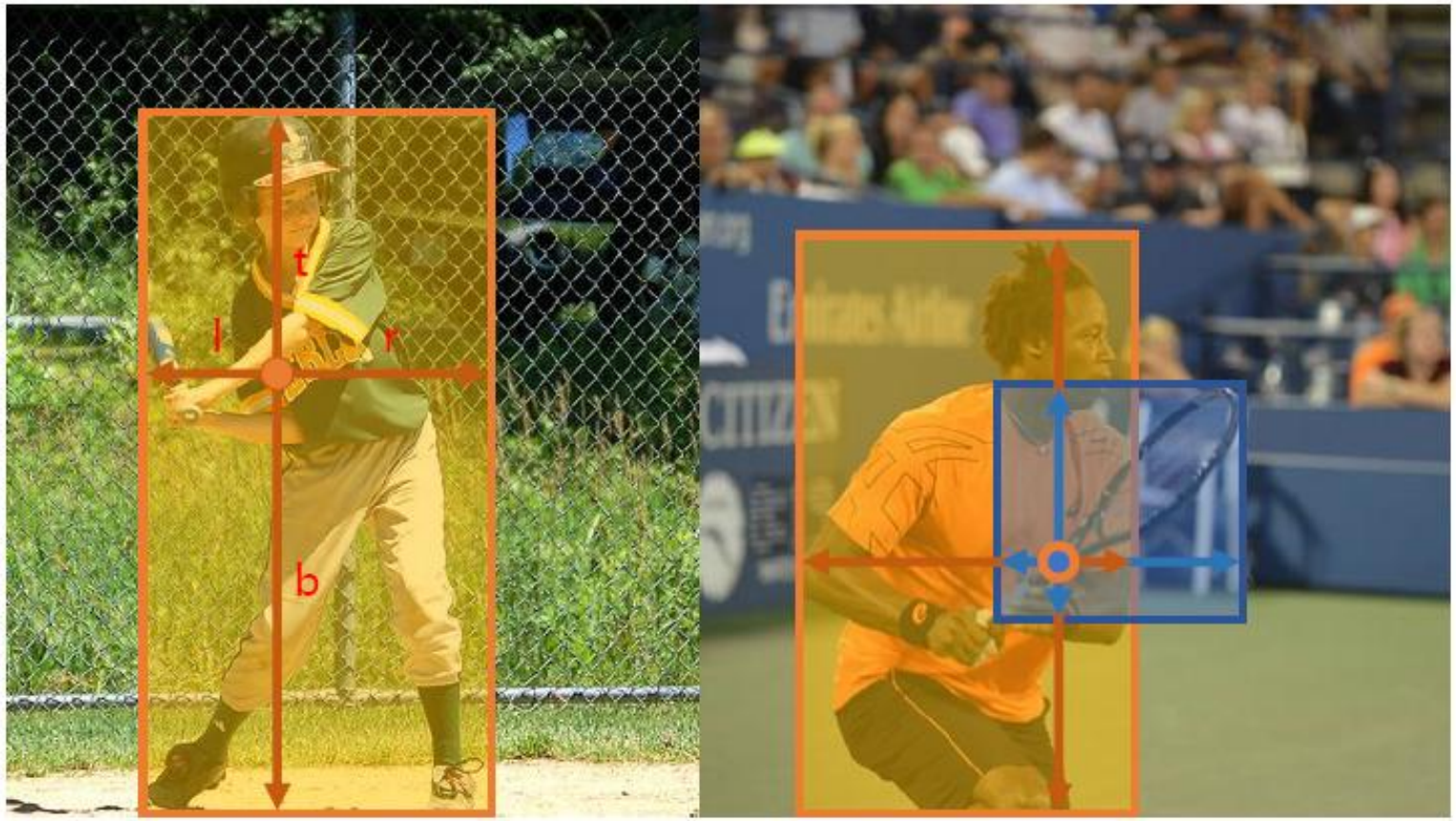
FCOS

6.2 FCOS

FCOS: Fully Convolutional One-Stage Object Detection

FCOS是一种基于FCN的逐像素目标检测算法，实现了**无锚点**(anchor-free)、**无建议框**(proposal free)的解决方案，同时提出**中心度**(centerness)的思想。

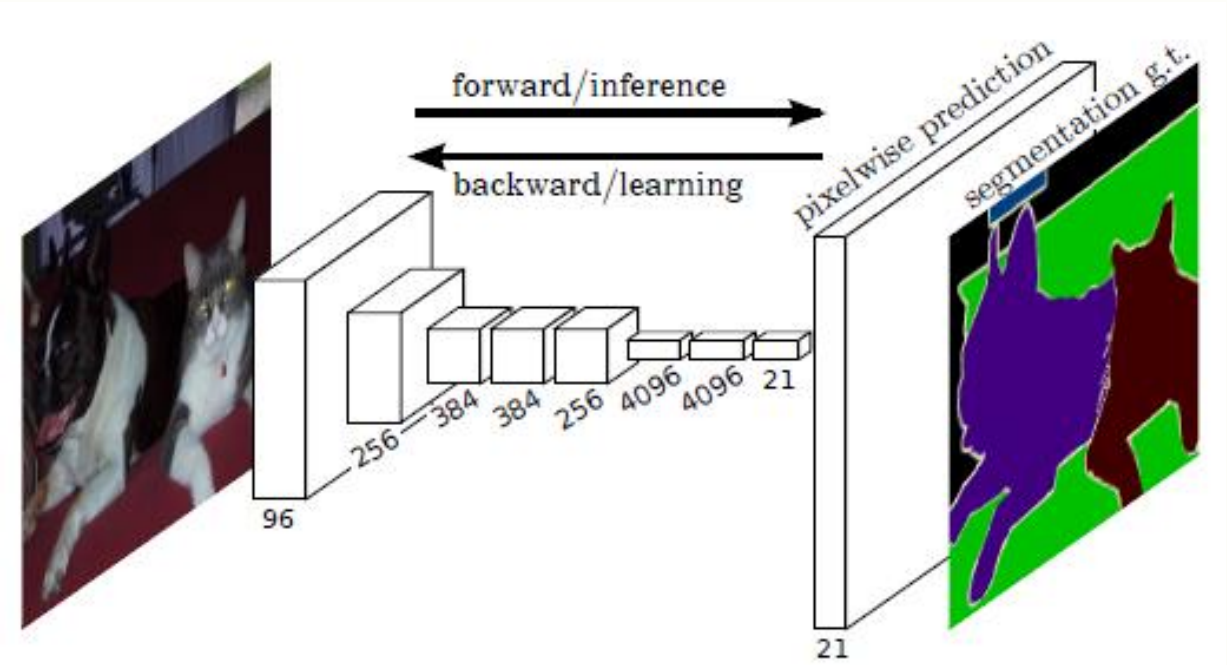




Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV 2019

6.2 FCOS

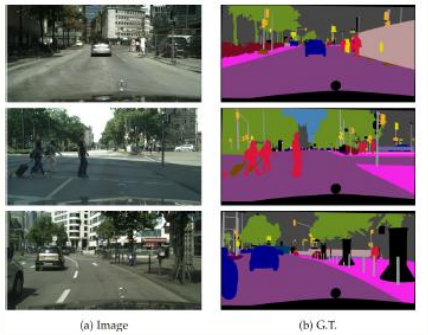
FCOS的核心思想



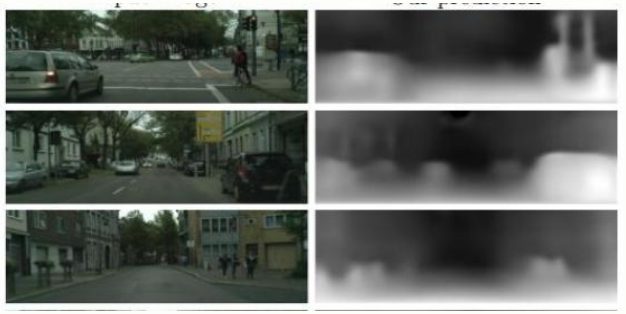
Fully Convolutional Networks for Semantic Segmentation

FCN是一种端到端的，像素到像素的语义分割方法，它实现利用监督预训练的方法实现像素级的预测，并且可以对任意尺寸的输入进行等尺度输出。

语义分割



深度估计



姿态估计

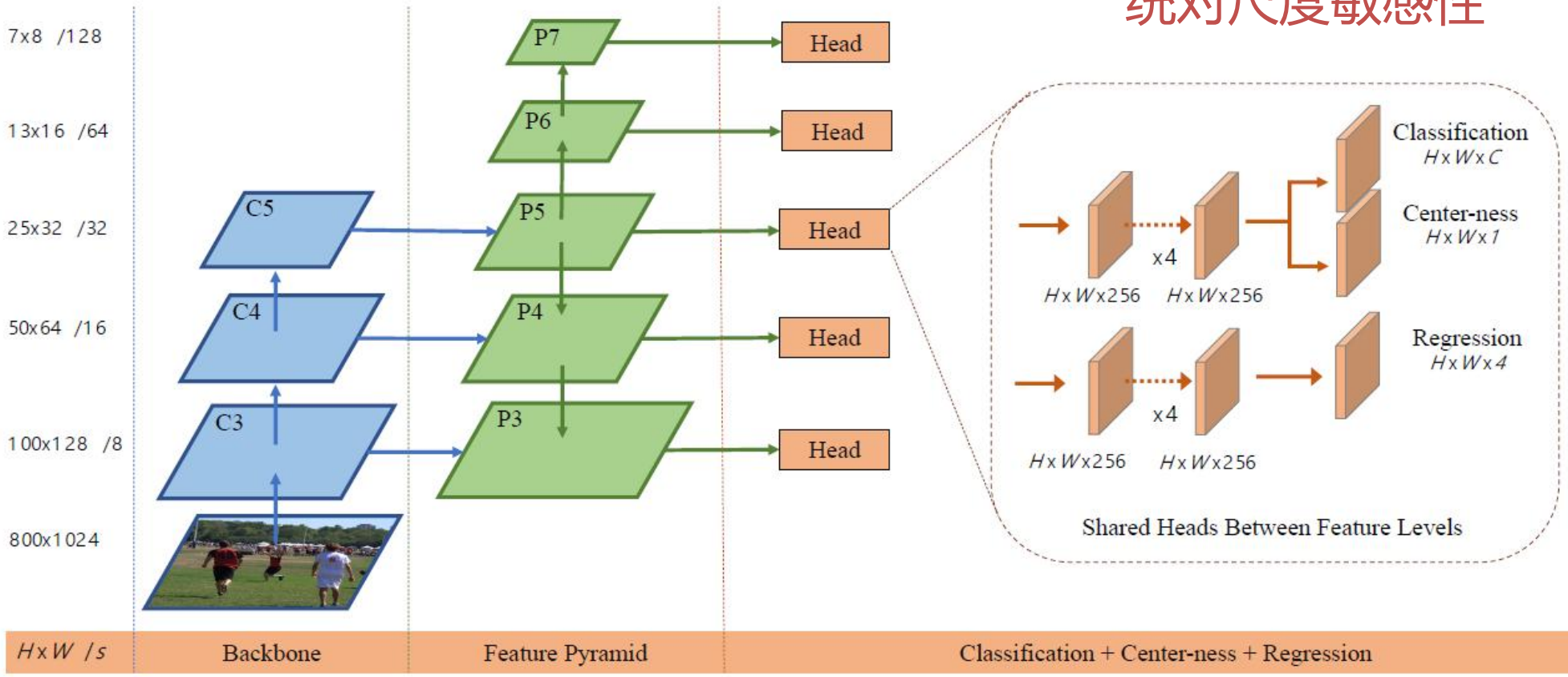
Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV 2019

6.2 FCOS

Is FCN can be used in Detection?

基于FPN的多级预测

实现对不同尺度目标的区分预测，从而提高检测系统对尺度敏感性



分类: Focal Loss

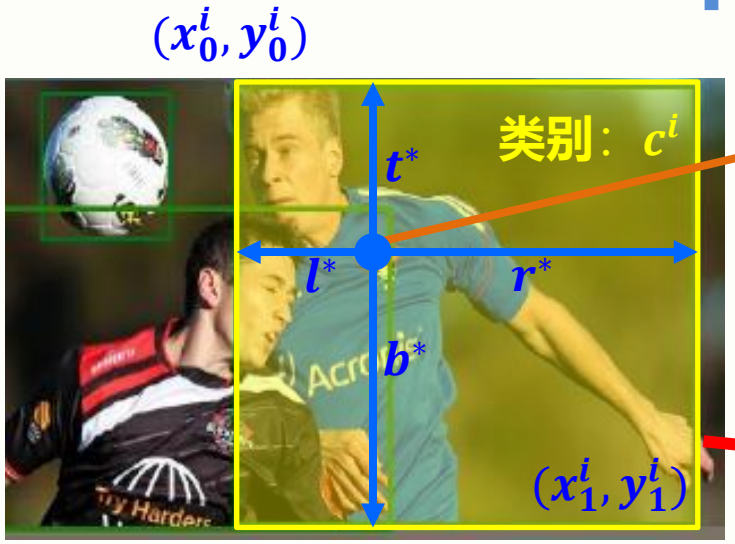
中心度: BCE Loss

回归: IoU Loss

Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV2019.

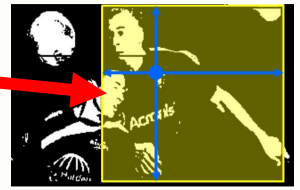
6.2 FCOS

FCOS的分类分支和回归分支



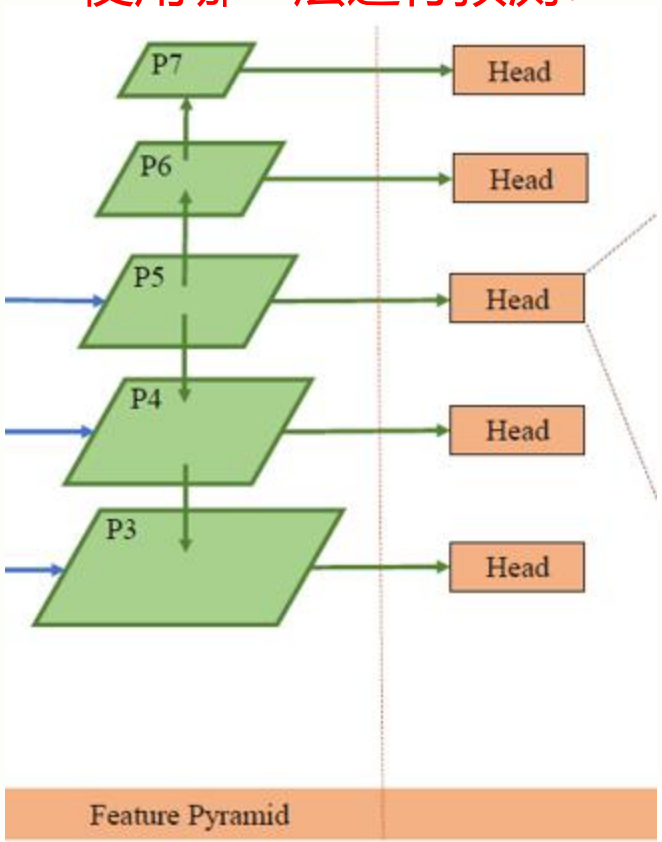
真实目标框B_i^*

分类分支以像素点为训练样本，而不是RoIs(IoU大于阈值的Anchor)。像素点在某类i目标框内，则为正样本，对应的类别i的标签$c_i = 1$，否则为负样本，标签$c_i = 0$



第i层的特征图F_i

使用哪一层进行预测？



多级预测

- **分类分支**：以像素点为训练样本，每个像素被标上类别或背景，进行分类训练
- **回归分支**：以目标框中每个像素到目标边框四条边的距离为目标值进行回归训练

➡ Focal Loss

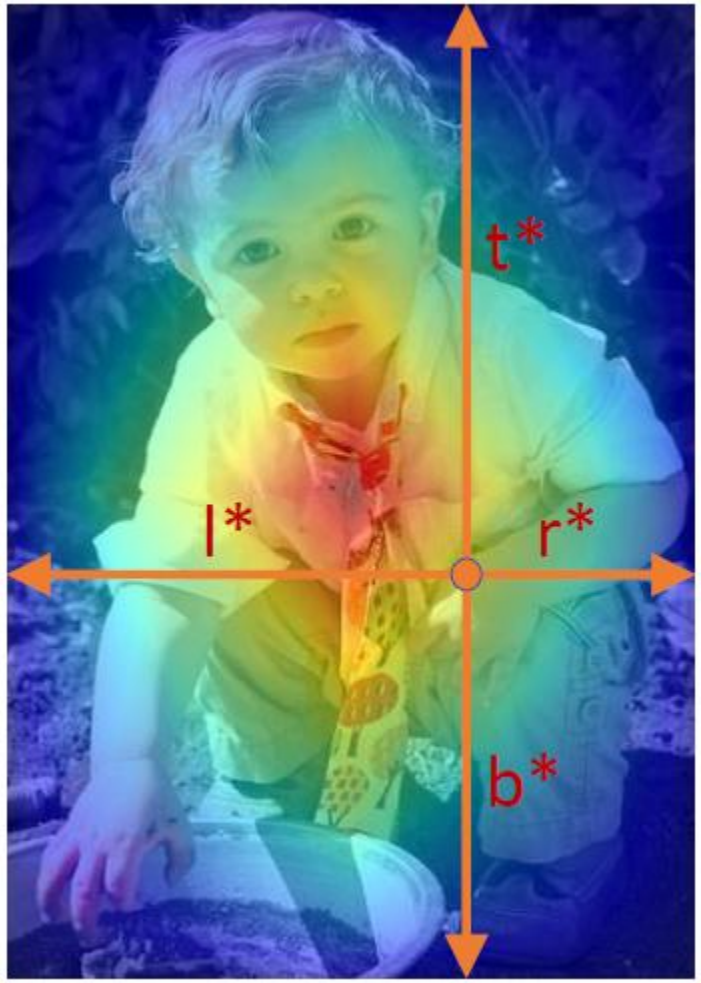
➡ IoU Loss

按照每一层的阈值，根据(l^*, r^*, t^*, b^*)的最大值进行分配

Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV2019

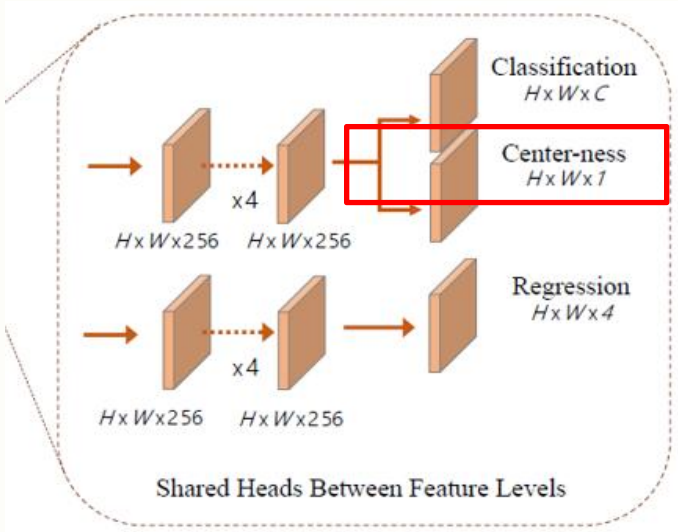
6.2 FCOS

FCOS的中心度(Center-ness)分支



- ✓ 越接近中心，Centerness->1
- ✓ 越偏离中心，Centerness->0

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$



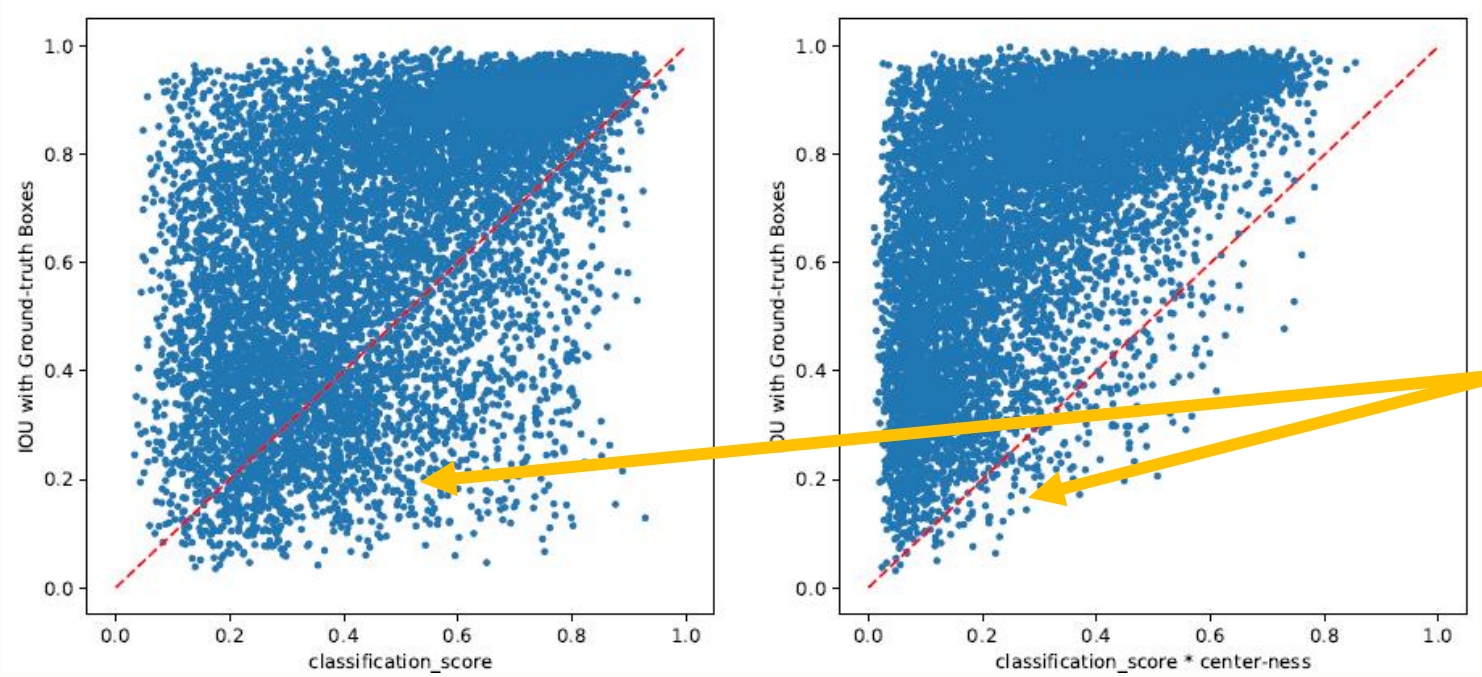
Binary Cross Entropy Loss

Score=Score*Centerness

Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV2019.

6.2 FCOS

FCOS中心度的作用



Center-ness提高了RoIs的质量，即减少了IoU值较低但置信度较高的BBox，也即大量偏离中心的目标。

RoIs质量的提高促进了检测性的提高。

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
None	33.5	52.6	35.2	20.8	38.5	42.6
center-ness†	33.5	52.4	35.1	20.8	37.8	42.8
center-ness	**37.1**	**55.9**	**39.8**	**21.3**	**41.0**	**47.8**

基于回归向量计算中心度

基于单独的分支计算中心度

Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV 2019

6.2 FCOS

Group Normalizaiton, Yuxin Wu, Kaiming He, ECCV2018

FCOS的进化版

Table 3: FCOS vs. RetinaNet on the minival split with ResNet50-FPN backbone

Method	C_5/P_5	w/ GN	nms thr.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
RetinaNet	C_5		.50	35.9	56.0	38.2	20.0	39.8	47.4	31.0	49.4	52.5
FCOS	C_5		.50	36.3	54.8	38.7	20.5	39.8	47.8	31.5	50.6	53.5
FCOS	P_5		.50	36.4	54.9	38.8	19.7	39.7	48.8	31.4	50.6	53.4
FCOS	P_5		.60	36.5	54.5	39.2	19.8	40.0	48.9	31.3	51.2	54.5
FCOS	P_5	✓	.60	37.1	55.9	39.8	21.3	41.0	47.8	31.4	51.4	54.9
Improvements												
1 + ctr. on reg.	P_5	✓	.60	37.4	56.1	40.3	21.8	41.2	48.8	31.5	51.7	55.2
2 + ctr. sampling [1]	P_5	✓	.60	38.1	56.7	**41.4**	**22.6**	41.6	**50.4**	32.1	52.8	56.3
3 + GIoU [1]	P_5	✓	.60	38.3	57.1	41.0	21.9	42.4	49.5	32.0	52.9	56.5
4 + Normalization	P_5	✓	.60	**38.6**	**57.4**	**41.4**	22.3	**42.5**	49.8	**32.3**	**53.4**	**57.1**

1. Centerness分支与回归分支共享特征
2. 在GT的中心部分只采样正样本
3. 使用GIoU Loss去惩罚IoU外的非重叠部分
4. 使用FPN的步长对回归目标进行正则化

Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV2019.

6.2 FCOS

Paddle对FCOS的改进

网络结构	骨干网络	图片数量/GPU	mAP	FPS*
FCOS(paper)	ResNet50	2	38.7	--
FCOS(paddle)	ResNet50	2	39.8	18.85
FCOS+multiscale_train(paddle)	ResNet50	2	**42.0**	**19.05**
FCOS+DCN(paddle)	ResNet50	2	**44.4**	**13.66**

*FPS为单卡V100 GPU，使用eval.py测评

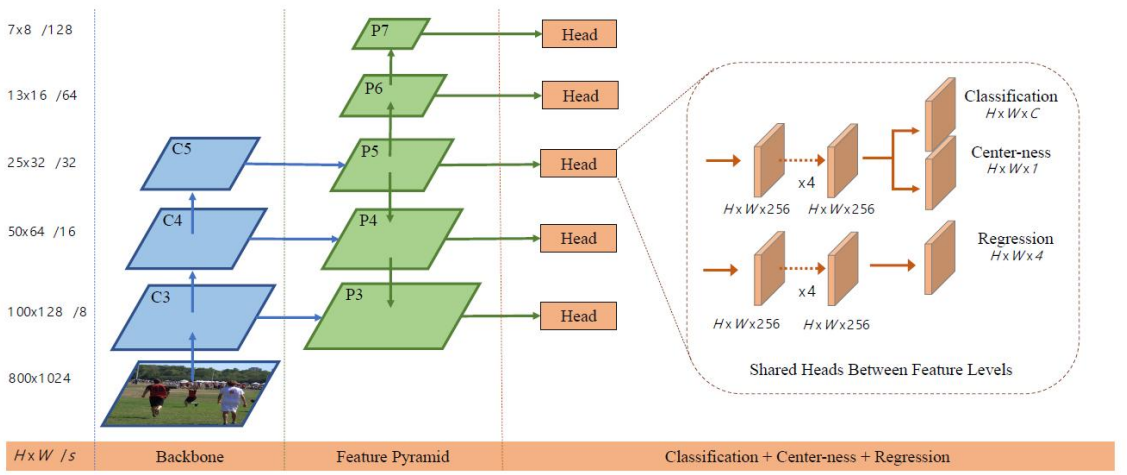
Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV2019

6.2 FCOS

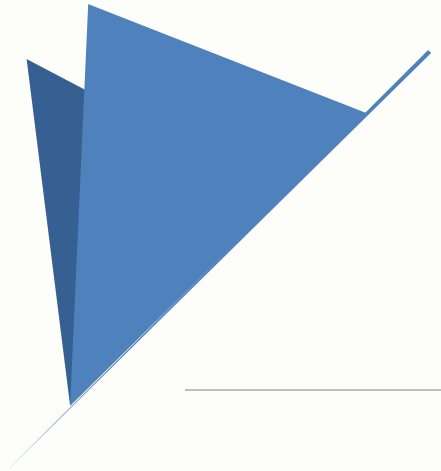
FCOS的算法小结

核心思想

- 检测每个像素点的类别和该点到目标框(GroundTruth)的距离
- 真实框内的点均为正样本
- 通过像素点到四边距离的最大值决定检测层（尺度选择）
- 去除距离中心点较远的低质量预测框（特别是置信度较高但IoU较低的Anchor）
- 像素点距离中心点越远，中心度越小



Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: Fully Convolutional One-Stage Object Detection. ICCV 2019



CenterNet

6.3 CenterNet

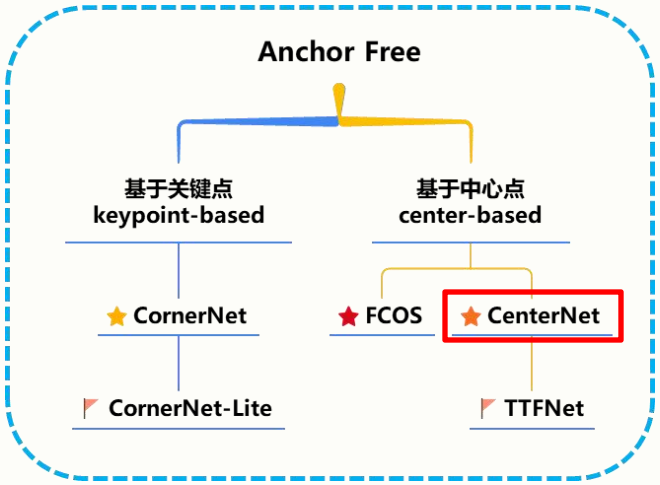
CenterNet: Objects as Points

Objects as Points

Xingyi Zhou
UT Austin
zhouxy@cs.utexas.edu

Dequan Wang
UC Berkeley
dqwang@cs.berkeley.edu

Philipp Krähenbühl
UT Austin
philkr@cs.utexas.edu



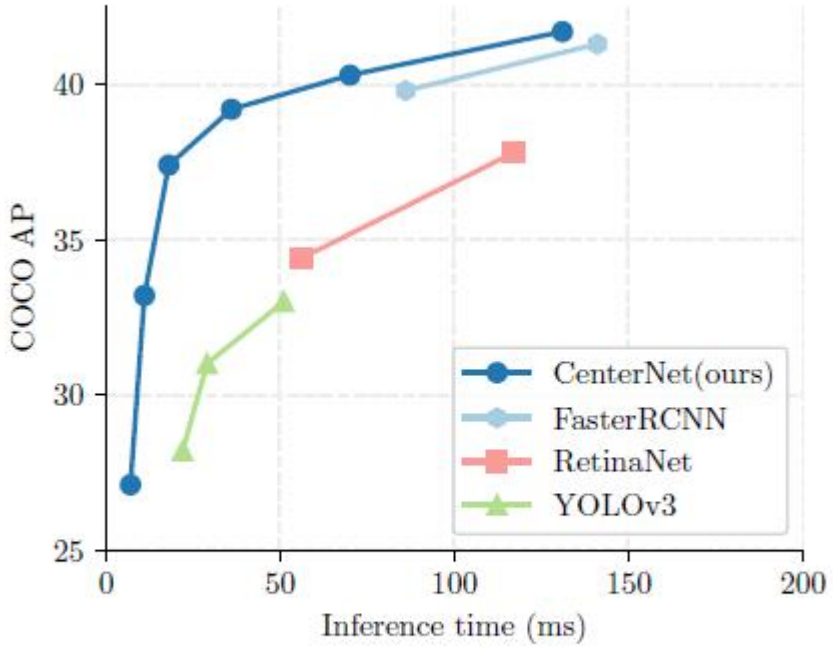
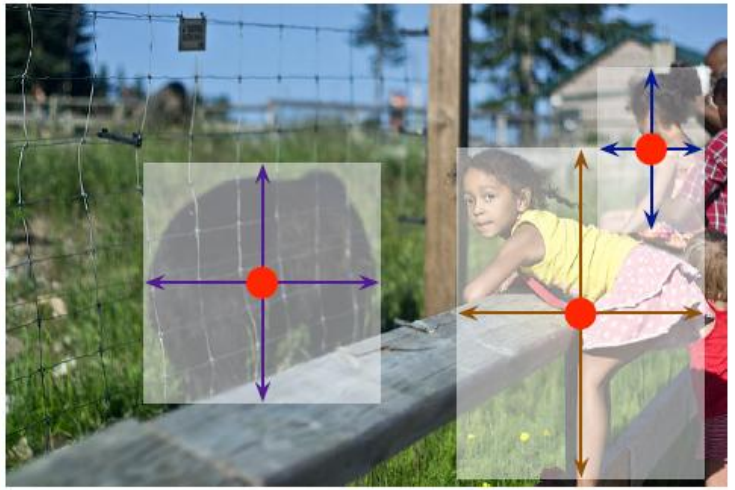


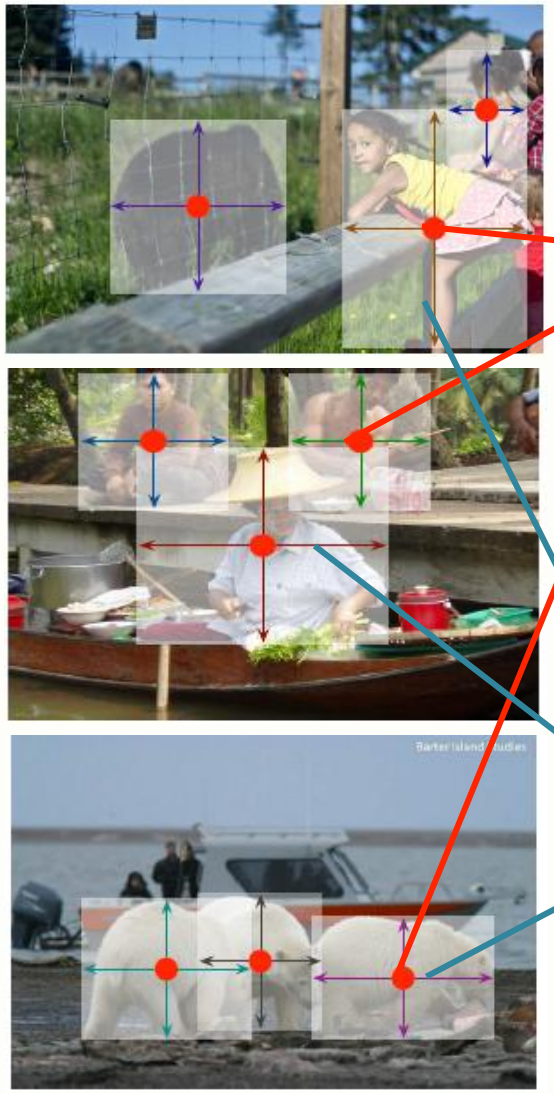
Figure 1: Speed-accuracy trade-off on COCO validation for real-time detectors. The proposed CenterNet outperforms a range of state-of-the-art algorithms.



Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points. 2019

6.3 CenterNet

CenterNet的核心思想



中心点

- ✓ 检测问题转化为求关键点，不依赖anchor
- ✓ 每个位置仅有一个正样本，不再使用NMS

物体类别

- ✓ 根据关键点预测物体的类别

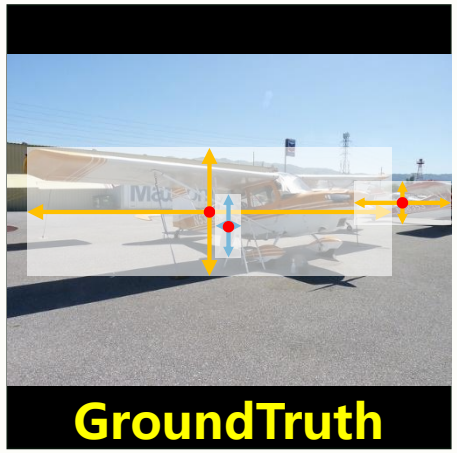
物体尺寸

- ✓ 直接预测目标框大小
- ✓ 能扩展到其他任务

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points.

6.3 CenterNet

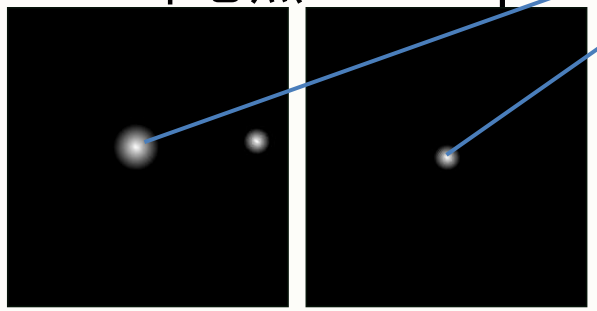
中心点（heatmap）的确定



(1x3x512x512)

1.计算中心点
2.生成heatmap

GT中心点heatmap



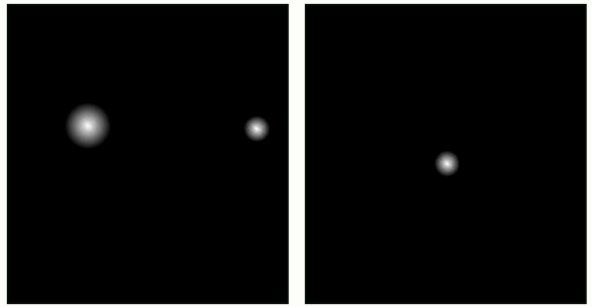
(1xCx128x128)

类别数，每个类存在一张heatmap

- 根据目标box大小计算高斯圆的半径R
- 以Point为圆心，半径R填充高斯函数
- Point点处为最大值，沿着半径向外按高斯函数递减

(1x3x512x512)

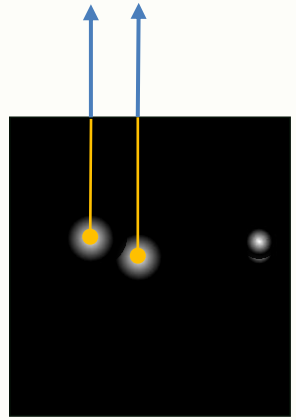
预测中心点heatmap



(1xCx128x128)

偏移优化

offset Loss



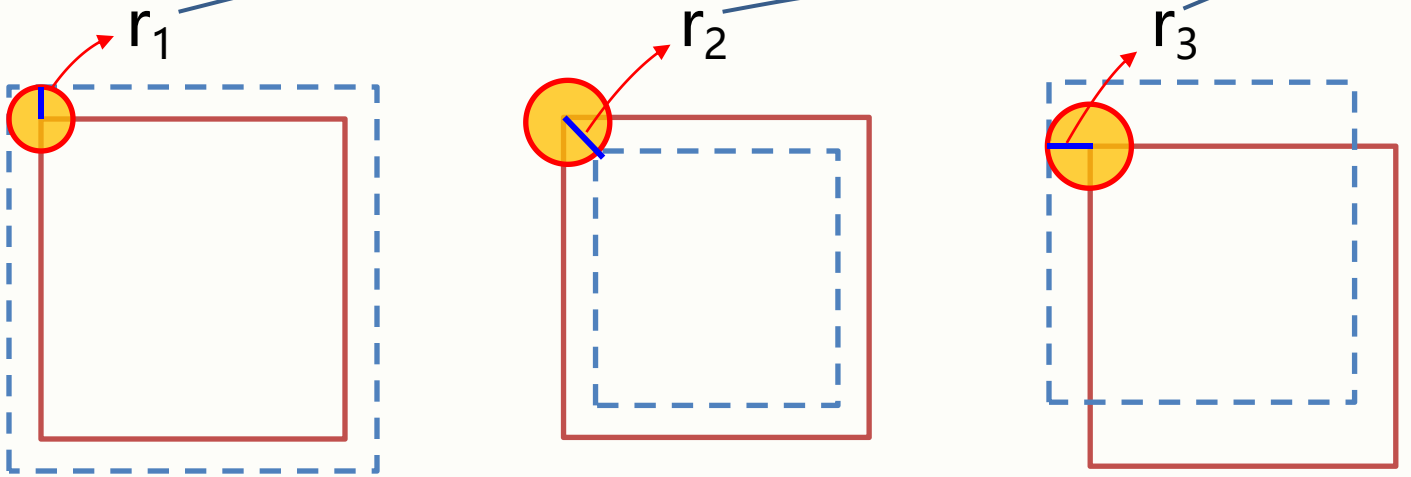
Heatmap的中心点如何确定半径?

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points. ICCV2019

6.3 CenterNet

中心点高斯半径的确定

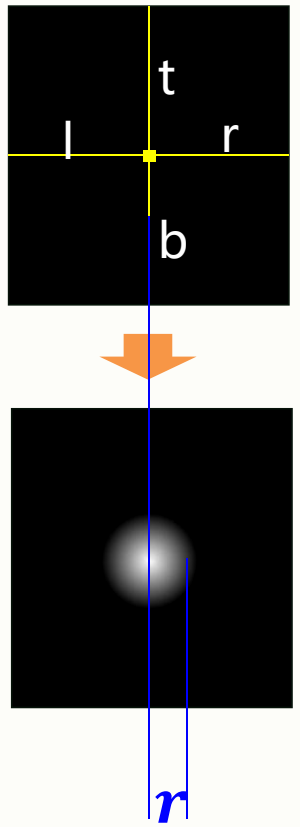
$$r = min(r_1, r_2, r_2)$$



预测框包围GT_BBox

GT_BBox包围预测框

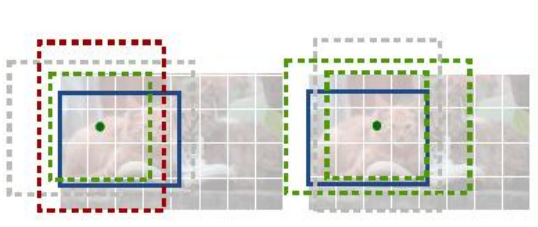
GT_Bbox和预测框重叠



6.3 CenterNet

预测目标的中心点

类比Anchor-based方法



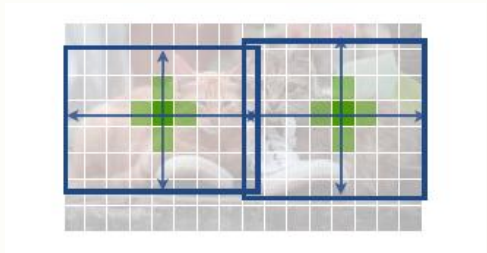
相似点

- ✓ 每个中心点可以看作是一个不含shape信息的anchor。

不同点

- ✓ 每个物体仅有一个正样本，不需要NMS，仅需提取heatmap上的最大值的位置即可
- ✓ 因为没有FPN，因此采用更大的分辨率进行解析
- ✓ 分配正样本时依赖于位置，而不依赖于Overlap

类别CornetNet方法



相似点

- ✓ 采用相似的关键点检测思路
- ✓ 使用heatmap和focal loss进行训练

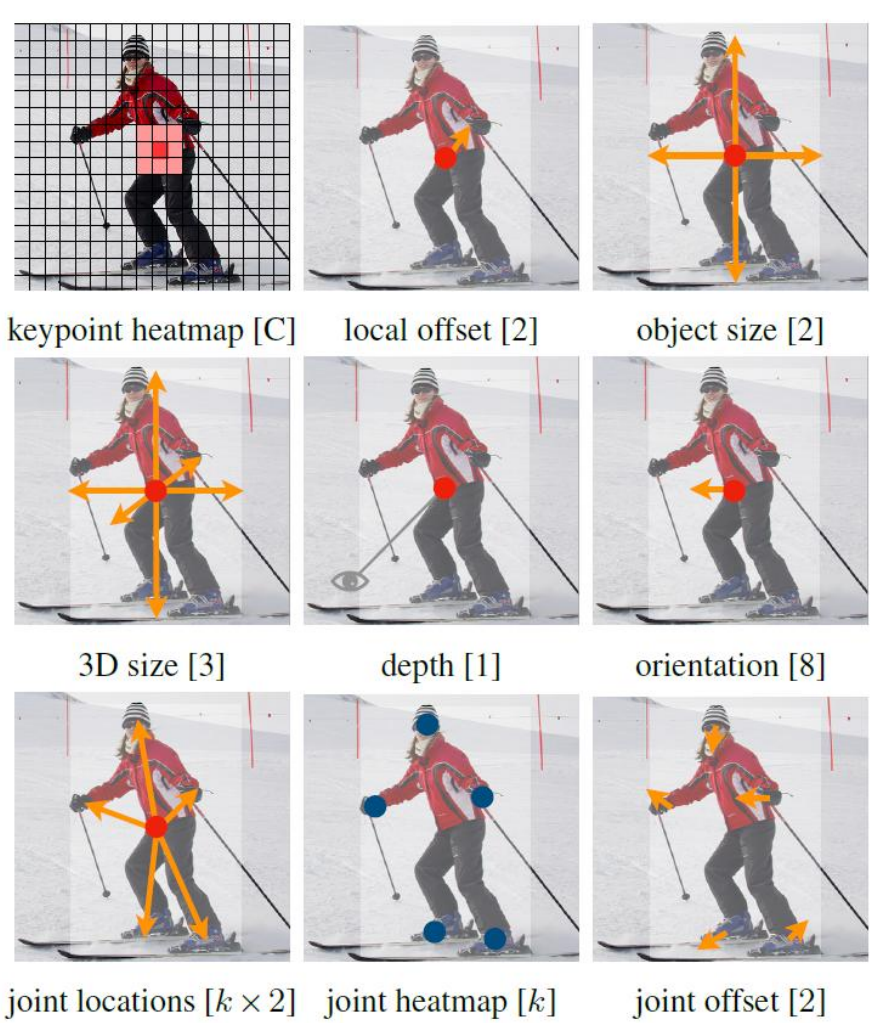
不同点

- ✓ 采用相似的offset分支， 弥补下采样的位置误差

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points. ICCV2019

6.3 CenterNet

多任务的支持



目标检测

- 使用 Focal loss 直接对中心点进行回归；
- 基于中心点，使用L1 Loss计算目标框的尺度

3D目标检测

在2D的基础上增加深度(Depth)和方向(Orientation)的回归目标

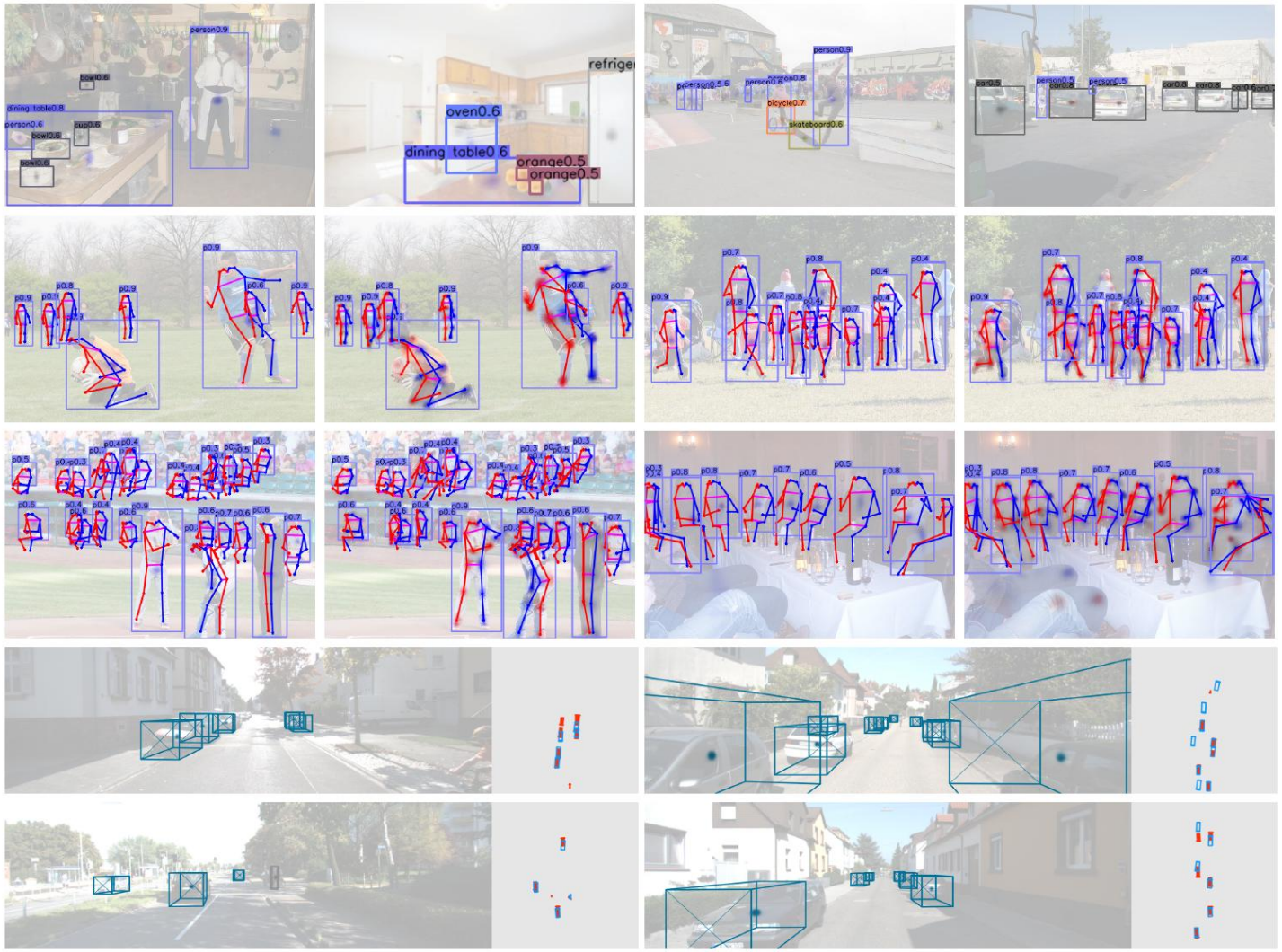
姿态估计

在coco上总共包含17个关键点，直接通过中心点预测这17个关键点的offset，然后再预测一个heatmap做match（回归offset比直接回归关键点更准确）

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points.

6.3 CenterNet

多任务的支持



coco目标检测

coco 姿态估计

KITTI 3D目标检测

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points.

6.3 CenterNet

CenterNet的对比实验

相似性能下，具有更快的检测速度

	Backbone	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MaskRCNN [21]	ResNeXt-101	**11**	39.8	62.3	43.4	22.1	43.2	51.2
Deform-v2 [63]	ResNet-101	-	46.0	67.9	50.8	27.8	49.1	59.5
SNIPER [48]	DPN-98	*2.5*	46.1	67.0	51.6	29.6	48.9	58.1
PANet [35]	ResNeXt-101	-	47.4	67.2	51.8	30.1	**51.7**	60.0
TridentNet [31]	ResNet-101-DCN	0.7	**48.4**	**69.7**	**53.5**	**31.8**	51.3	**60.3**
YOLOv3 [45]	DarkNet-53	20	33.0	57.9	34.4	18.3	25.4	41.9
RetinaNet [33]	ResNeXt-101-FPN	5.4	40.8	61.1	44.1	24.1	44.2	51.2
RefineDet [59]	ResNet-101	-	36.4 / 41.8	57.5 / 62.9	39.5 / 45.7	16.6 / 25.6	39.9 / 45.1	51.4 / 54.1
CornerNet [30]	Hourglass-104	4.1	40.5 / 42.1	56.5 / 57.8	43.1 / 45.3	19.4 / 20.8	42.7 / 44.8	**53.9** / 56.7
ExtremeNet [61]	Hourglass-104	3.1	40.2 / 43.7	55.5 / 60.5	43.2 / 47.0	20.4 / 24.1	43.2 / 46.9	53.1 / 57.6
FSAF [62]	ResNeXt-101	*2.7*	**42.9** / 44.6	**63.8** / **65.2**	**46.3** / 48.6	**26.6** / **29.7**	**46.2** / **47.1**	52.7 / 54.6
CenterNet-DLA	DLA-34	**28**	39.2 / 41.6	57.1 / 60.3	42.8 / 45.1	19.9 / 21.5	43.0 / 43.9	51.4 / 56.0
CenterNet-HG	Hourglass-104	7.8	42.1 / **45.1**	61.1 / 63.9	45.9 / **49.3**	24.1 / 26.6	45.5 / **47.1**	52.8 / **57.7**

同等速度下，具有更好的AP

整体性能有一定的提升，具有更好的性价比

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points.

6.3 CenterNet

CenterNet算法小结

核心思想

✓ 直接检测物体的中心和尺寸

Advantage

✓ 模型结构设计简单（DLA-34 and Hourglass-104）

✓ 对其他视觉任务具有较好的扩展性

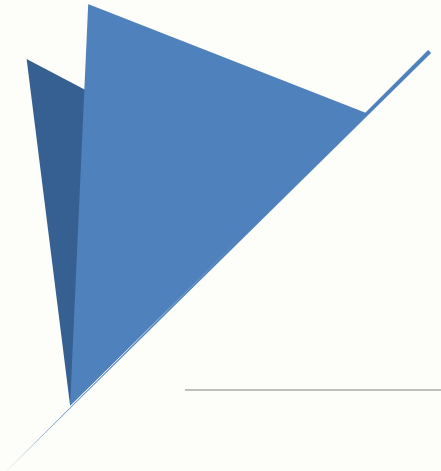
✓ 去除NMS后处理过程有利于加速模型预测

Disadvantage

✓ 训练时间较长，coco数据集需要大约140epochs

✓ 回归的监督信息仅通过中心位置产生

Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. CenterNet: Objects as Points. 2019



TTFNet

6.4 TTFNet

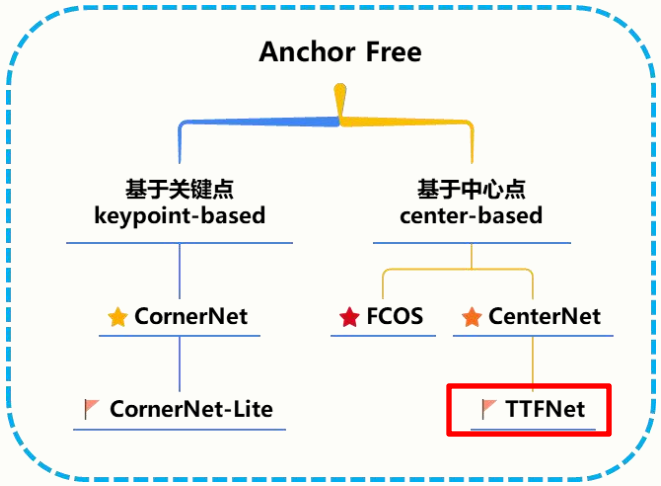
TTFNet的核心思想

减少数据预处理

提高学习率

提高监督信号质量

保持性能的基础上提高模型的收敛速度



Zili Liu, Tu Zheng, et al. TTFNet: Training-Time-Friendly Network for Real-Time Object Detection. 2020

6.4 TTFNet

TTFNet对CenterNet的改进

分支	CenterNet	TTFNet
中心点Focal loss检测分支	在中心点周围采用半径为r的圆形高斯核产生正样本监督信号	在中心点周围采用椭圆高斯核产生**正样本**监督信号，椭圆的长短半径与真实框大小成比例
尺寸回归分支	仅在中心点产生真实框大小的监督信息	中心点周围采用椭圆高斯核产生采样区域，区域内的点都产生回归分支**正样本**监督信息
Offset分支	弥补降采样带来的坐标误差	回归分支预测了点到真实框四边的距离，因此可以去除offset分支

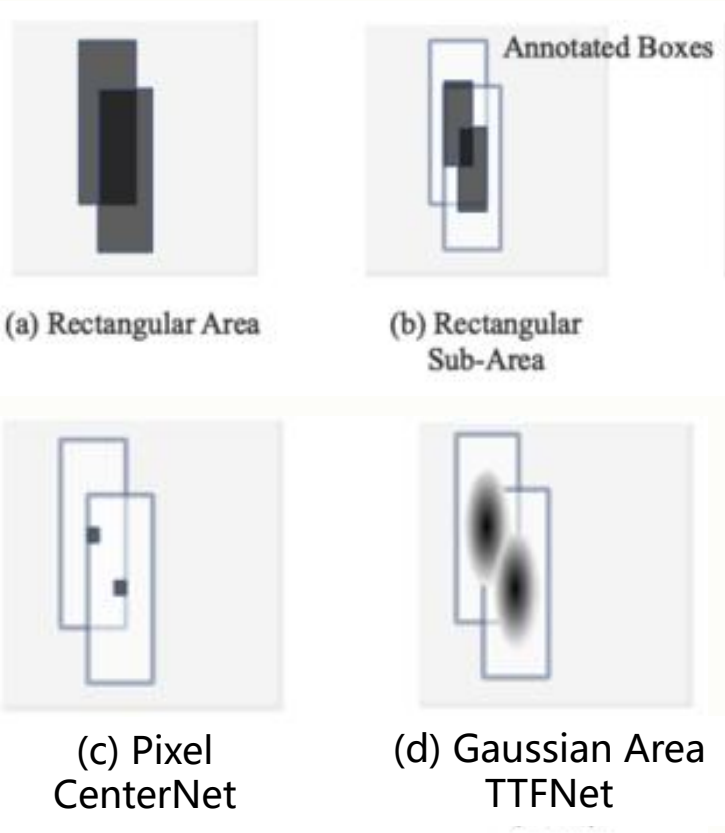


图3 不同的采样策略。黑色为正例，高斯定义权重。

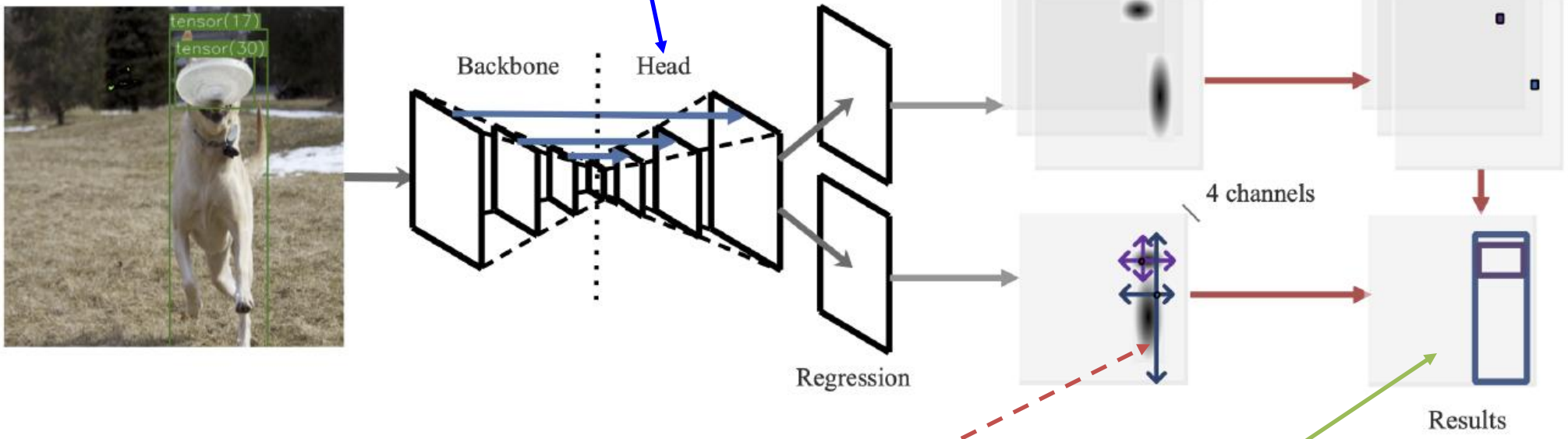
Zili Liu, Tu Zheng, et al. TTFNet: Training-Time-Friendly Network for Real-Time Object Detection. AAAI2020

6.4 TTFNet

TTFNet的网络结构

上采样：引入可变形卷积提升采样质量

定位分支：使用Focal Loss平衡难易样本



直接预测距离，不需要offset分支

回归分支：使用GIoU Loss，配合采样权重，平衡不同尺度的对象。（降低大尺寸目标的权重，提高小尺寸目标对总损失的贡献）

Zili Liu, Tu Zheng, et al. TTFNet: Training-Time-Friendly Network for Real-Time Object Detection. AAAI 2020

6.4 TTFNet

TTFNet的效果总结

Method	Backbone	Schedule	w/Augmentation	AP
CenterNet	R18	2x	✓	20.0
CenterNet	R18	2x		20.8
TTFNet	R18	2x		28.1
CenterNet	R18	11.67x	✓	28.1
TTFNet	R18	10x	✓	31.8
CenterNet	DLA34	2x	✓	26.2
CenterNet	DLA34	2x		31.6
TTFNet	DLA34	2x		34.9
CenterNet	DLA34	11.67x	✓	37.4
TTFNet	DLA34	10x	✓	38.2

Table 8: TTFNet vs. CenterNet.

网络结构	骨干网络	BatchSize	mAP	FPS
TTFNet	DarkNet53	96	32.9	85.92

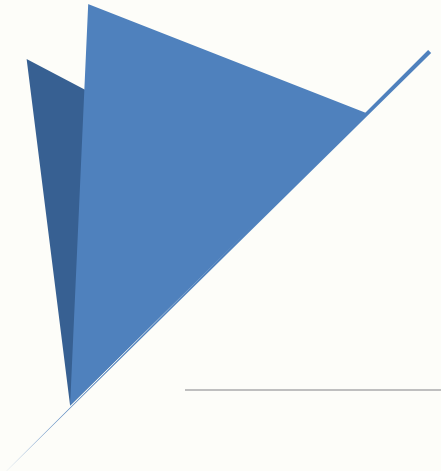
学习率：0.0005 -> 0.015

训练轮数：140epoch -> 12epoch

=>PaddleDetection – configs – anchor_free

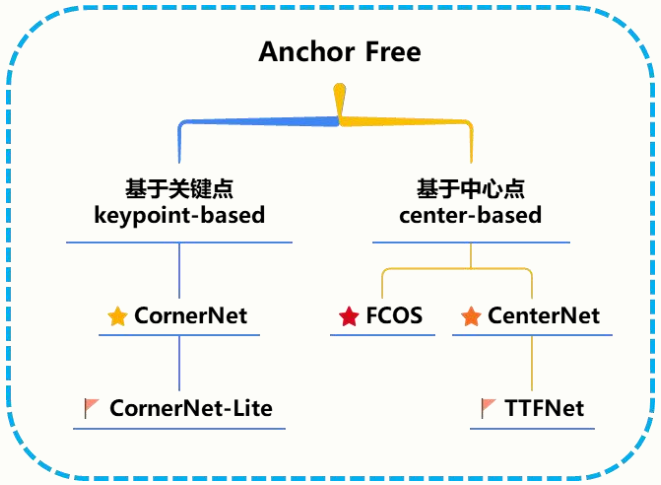
- 讨论并验证了batch size和真实框(使用椭圆高斯产生大量的BBox)产生编码信息的关联
- 提出使用椭圆高斯核应用到产生中心点定位和尺寸回归监督信息的新方法
- 在保持高性能的同时，极大的降低了检测器训练时间
- 模型结构有利于搜索实验

Zili Liu, Tu Zheng, et al. TTFNet: Training-Time-Friendly Network for Real-Time Object Detection. 2019/2020



Anchor Free系列算法小结

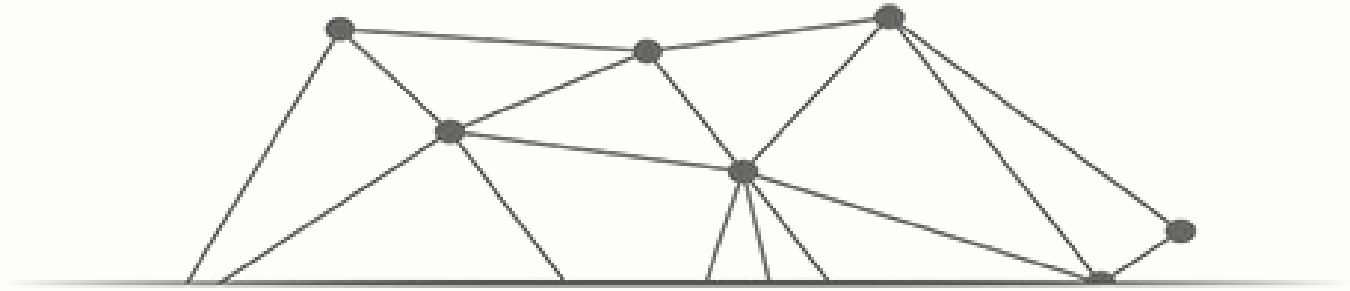
6.5 Anchor Free系列算法小结



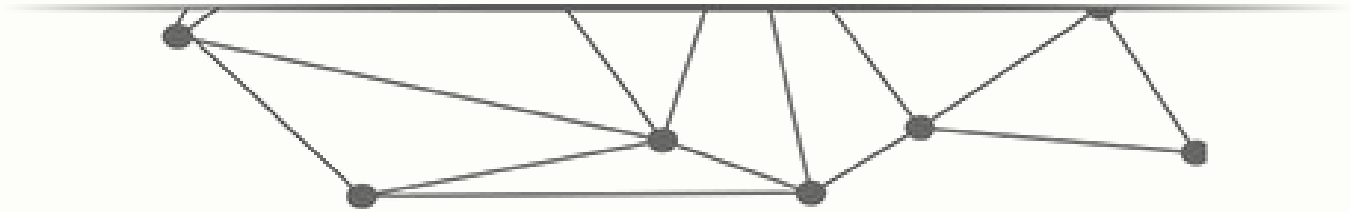
ExtremeNet
CenterNet
RepPoints
CentripetalNet
…

FoveaBox
ATSS
BorderDet
AutoAssign
…

模型	检测机制	优点	缺点	使用范围
Corner Net	一对角点	corner, corner pooling, embeddings提高模型精度	角点分组难度和计算量较大，对边缘敏感，忽略内部信息	多目标检测
FCOS	点+点到四边的距离	多尺度预测解决目标模糊性，设计复杂度低	多尺度监督信息需要增强，中心度可解释性需要增强	可用于实力分割
Center Net	中心点+长宽	简单高效，没有nms后处理	只使用中心点回归，可获得监督信息信息较少	可用于2D、3D目标检测及人体姿态识别
TTFNet	高斯椭圆+像素到四边的距离	使用高斯核编码训练样本，网络收敛更快；能够降低模糊及低质量样本；不需要offset预测；资源消耗较低		多目标检测



课堂互动 13.2.12

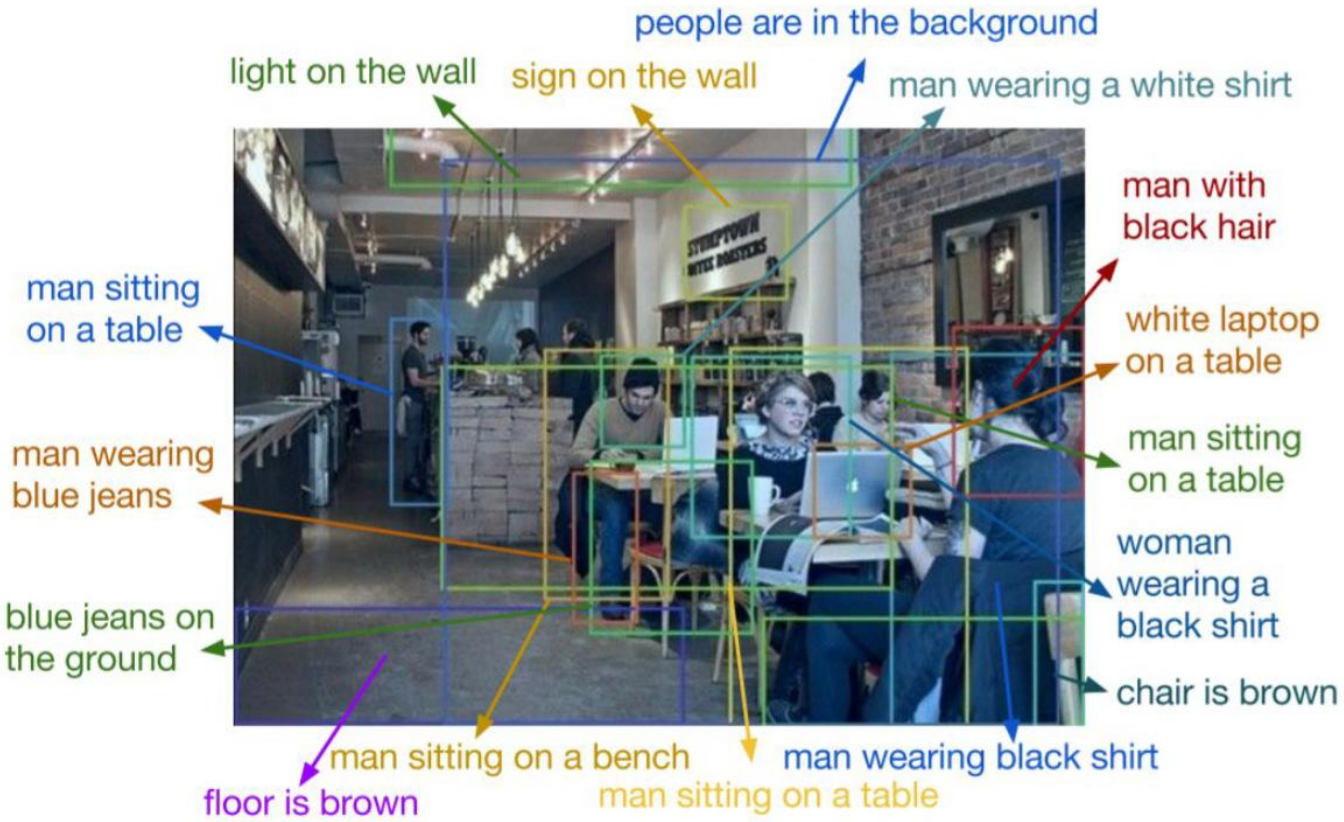


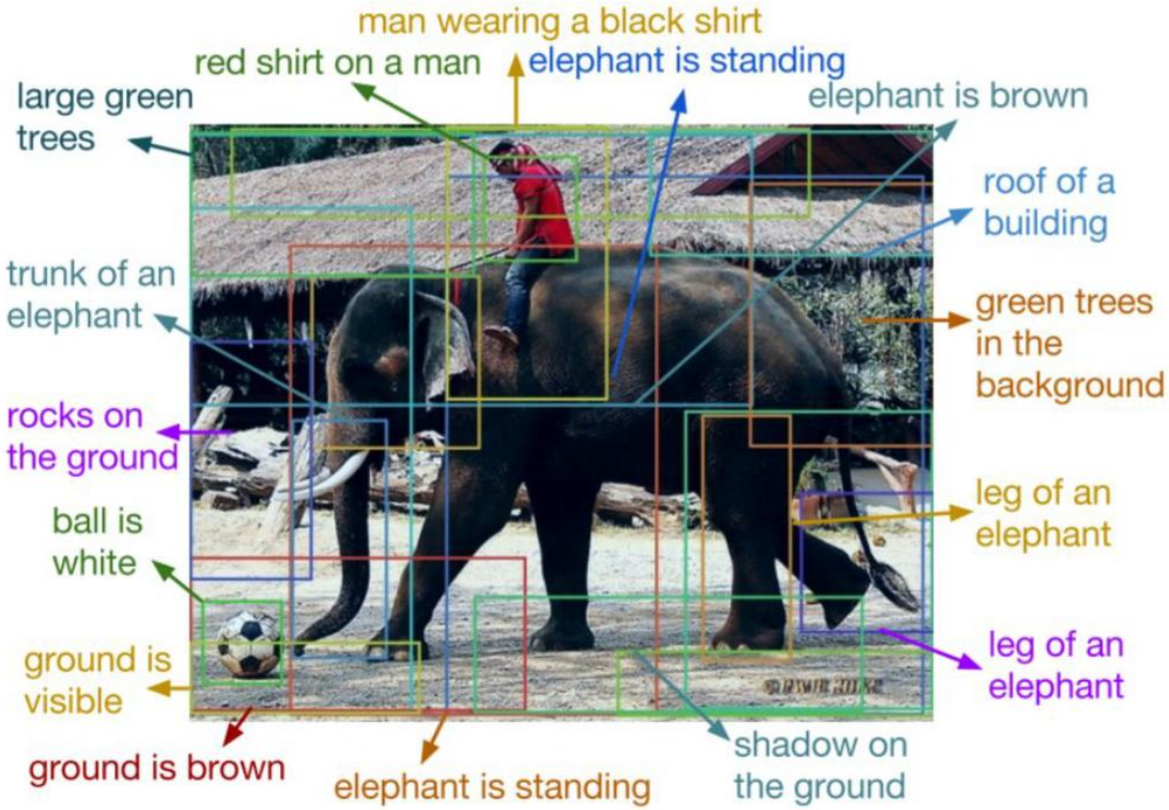
Part 07

Beyond 2D

7. Beyongd 2D

目标检测+图像字幕 = 密集图像字幕

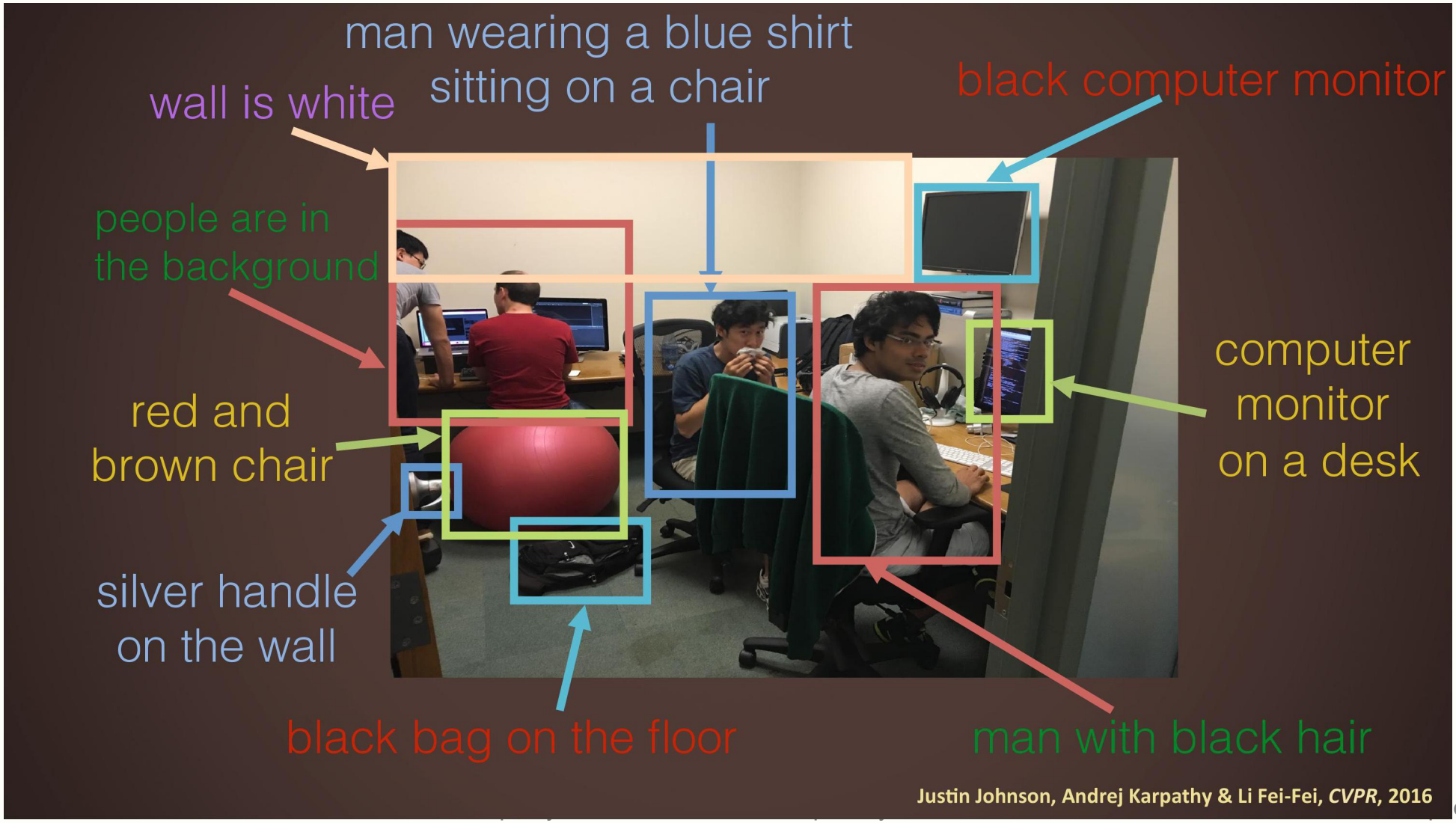




Johnson, Karpathy, and Fei-Fei, “DenseCap: Fully Convolutional Localization Networks for Dense Captioning”, CVPR 2016

6. Beyongd 2D

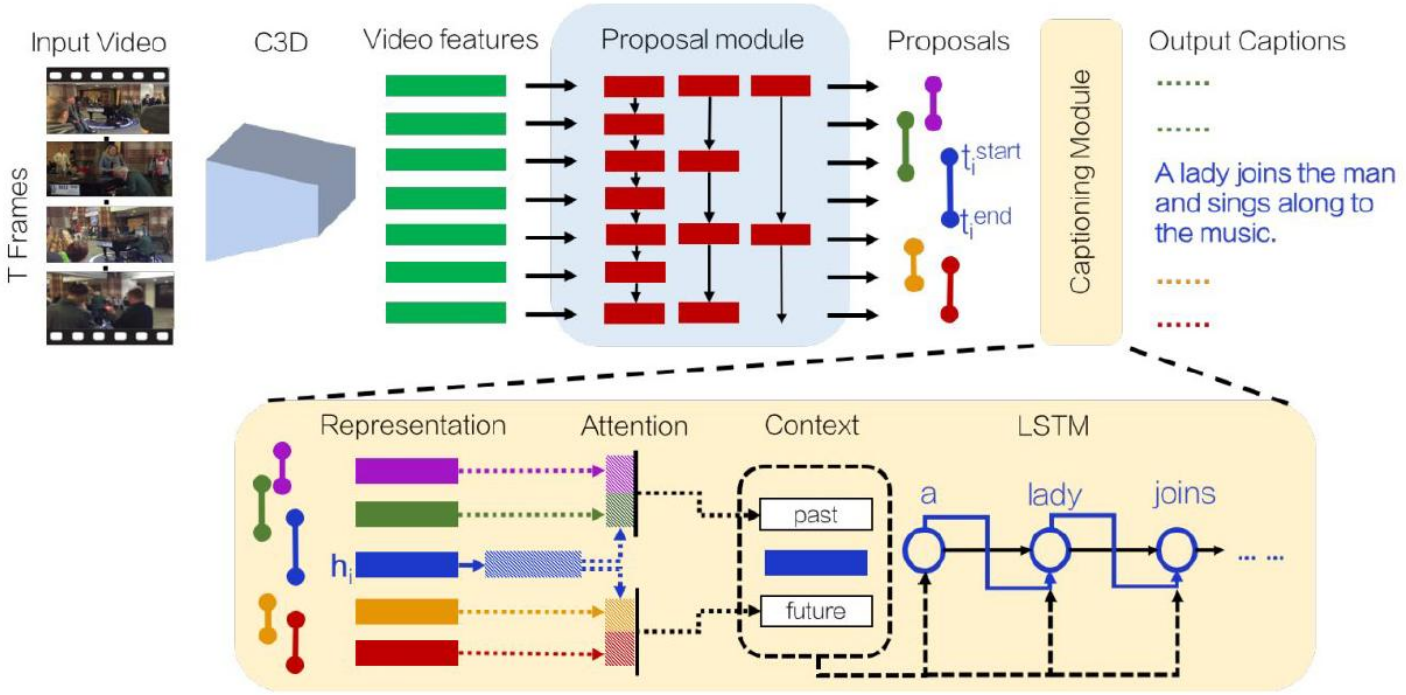
目标检测+图像字幕 = 密集图像字幕

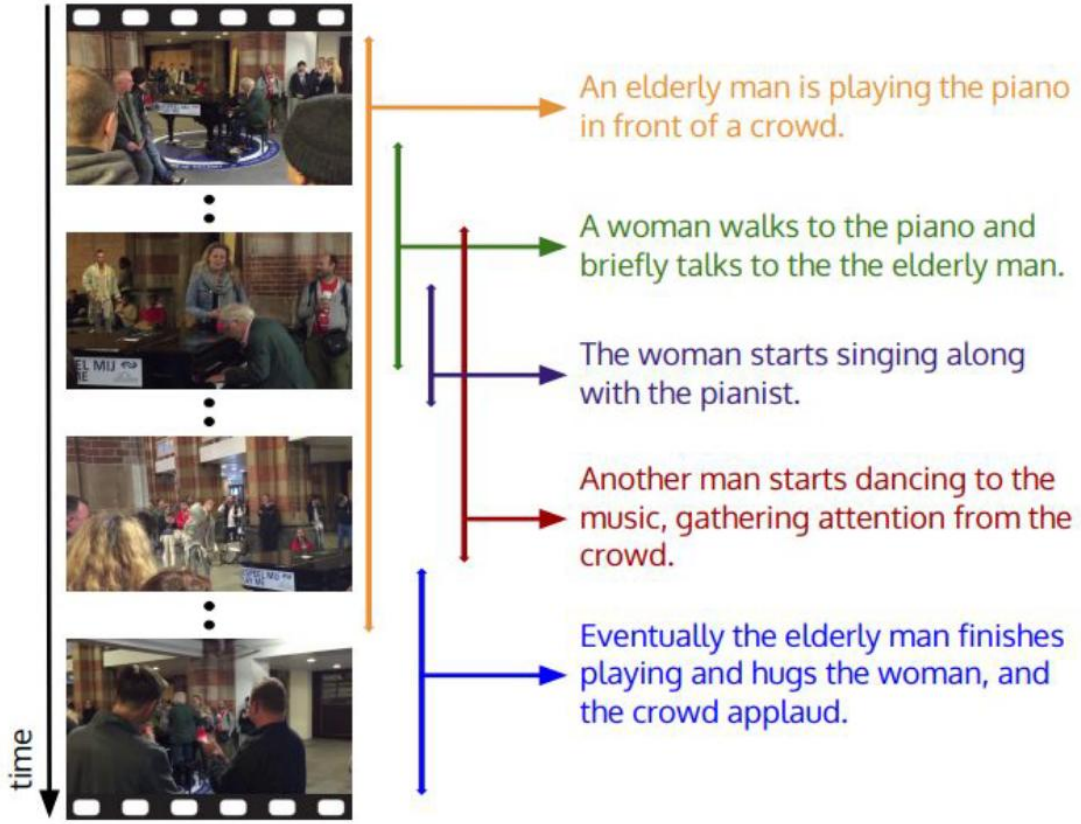


oning", CVPR 2016

6. Beyongd 2D

密集视频字幕

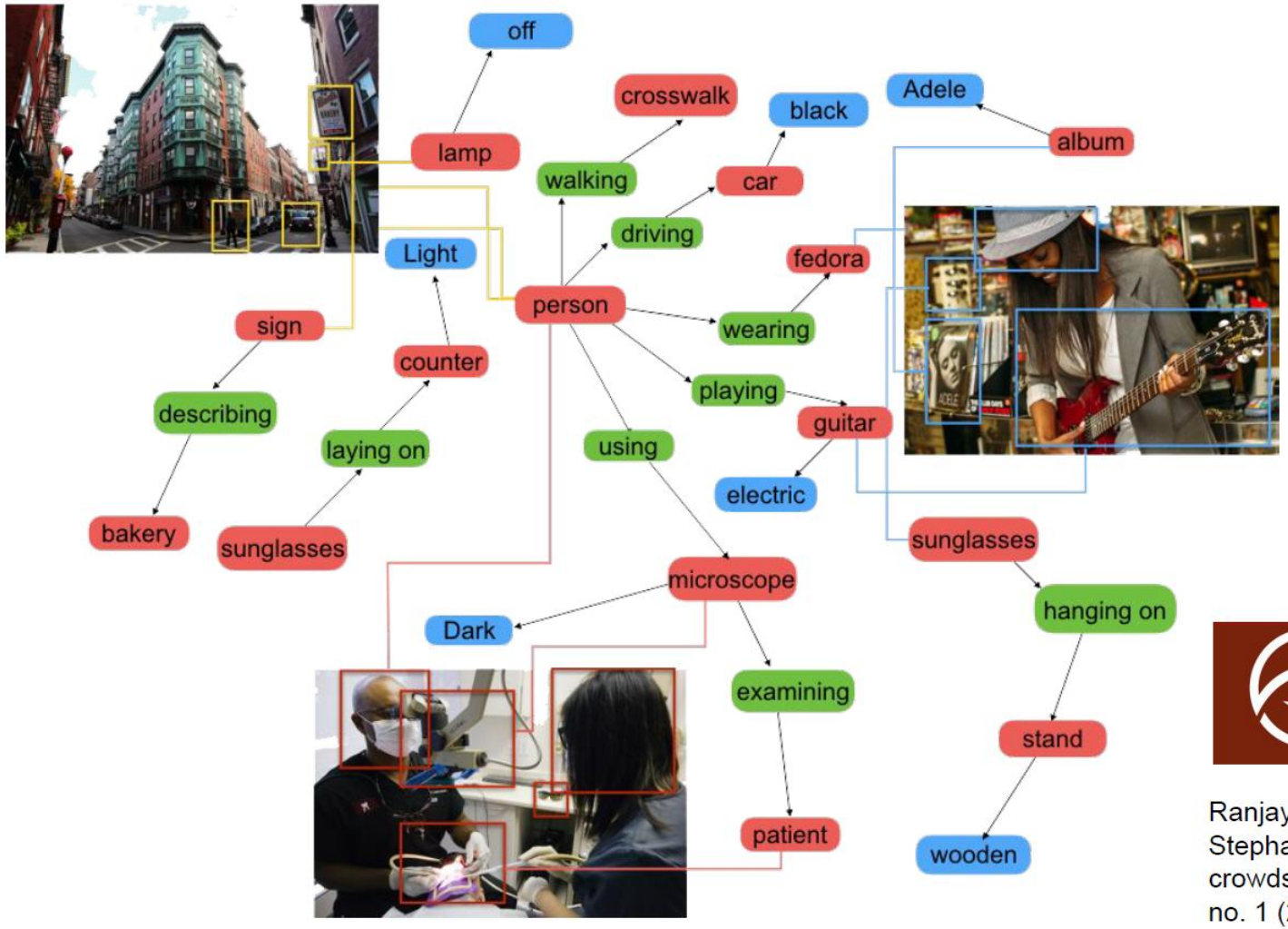




Ranjay Krishna et al., “Dense-Captioning Events in Videos”, ICCV 2017

6. Beyongd 2D

目标检测+关系 = 场景图



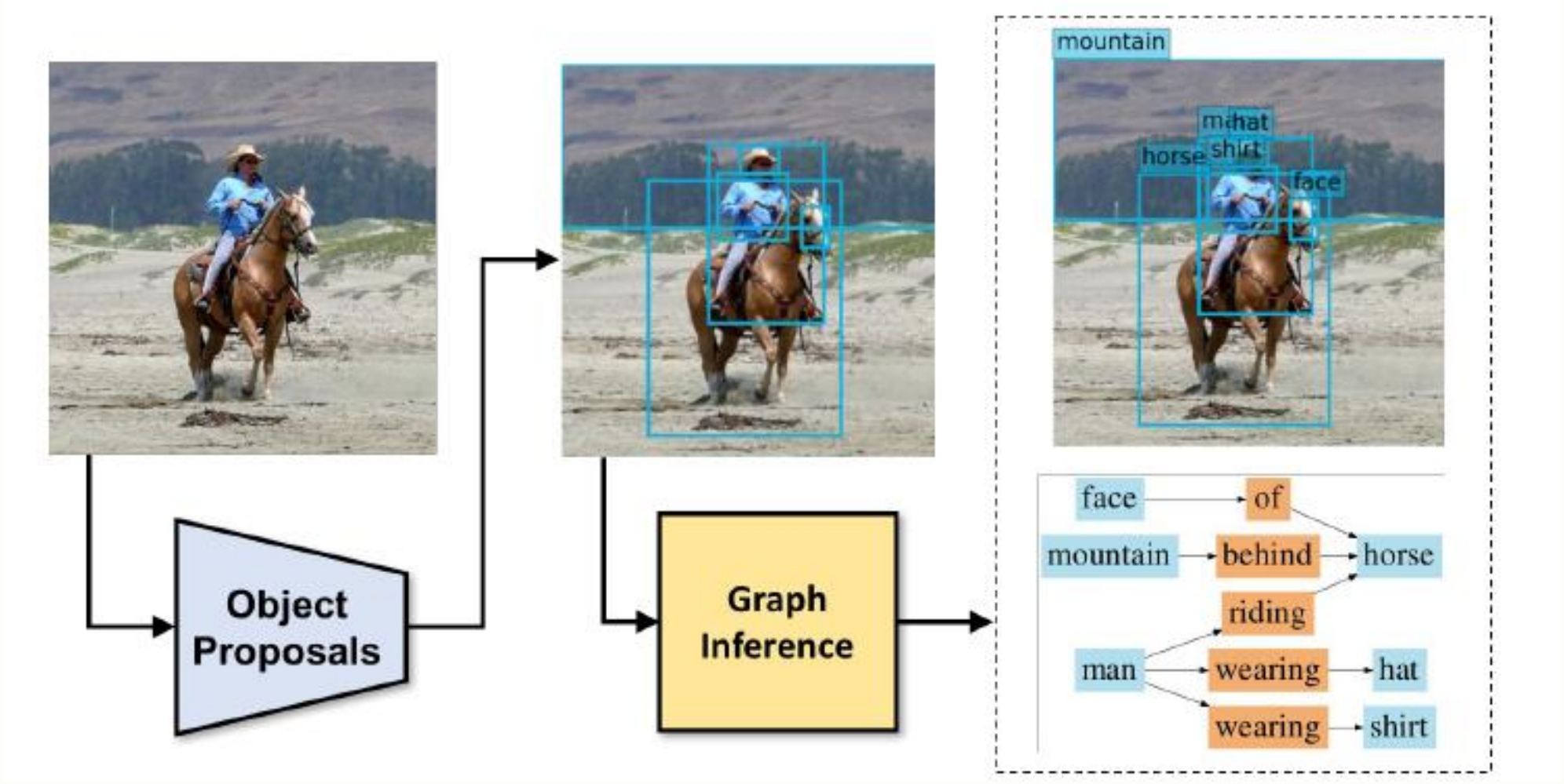
108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets

VISUALGENOME

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." International Journal of Computer Vision 123, no. 1 (2017): 32-73.

6. Beyongd 2D

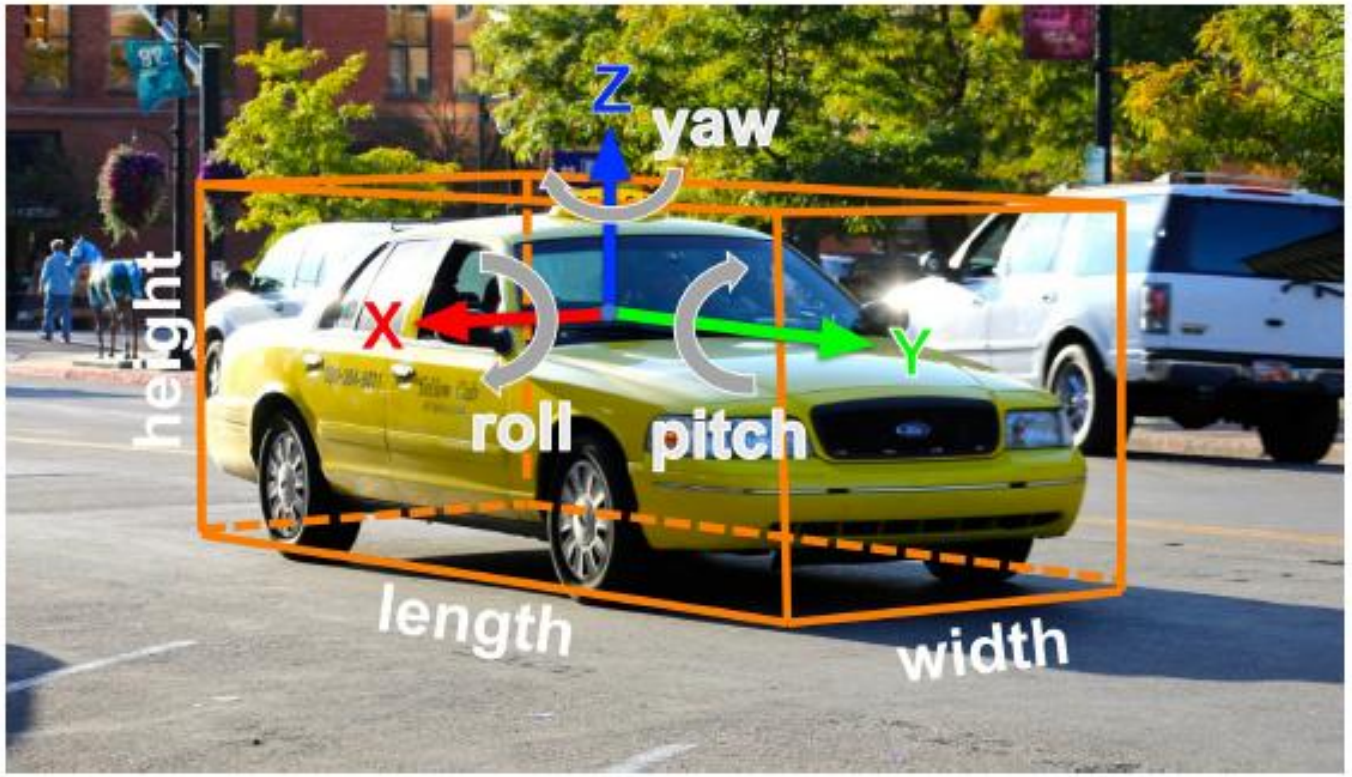
目标检测+关系 = 场景图



Xu, Zhu, Choy, and Fei-Fei, “Scene Graph Generation by Iterative Message Passing”, CVPR 2017

6. Beyongd 2D

3D目标检测



- **2D目标检测**
  - 2D边界框

    (x, y, w, h)

- **3D目标检测**
  - 3D方向边界框

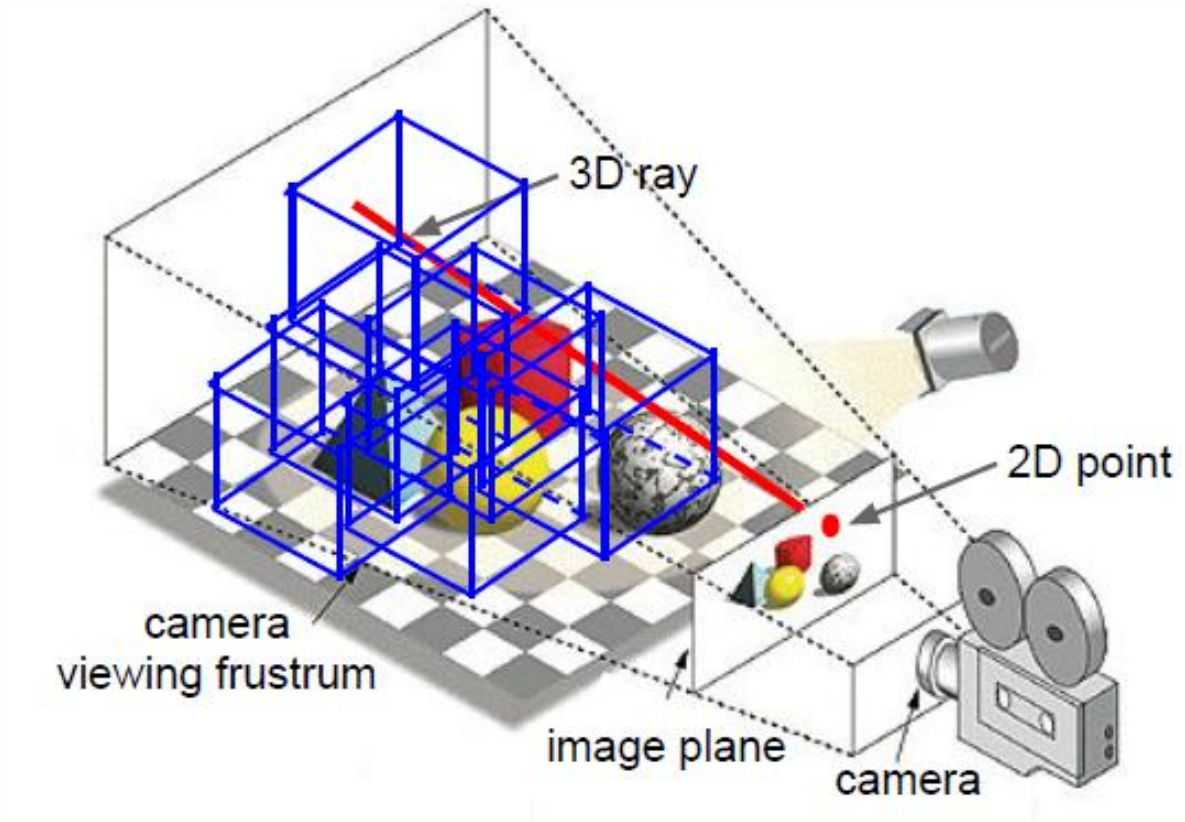
    (x, y, z, w, h, l, r, p, y)
  - 简化版bbox：

    (x, y, z, w, h, l, y) 没有roll和pitch

相比2D检测，3D检测要难得多

6. Beyongd 2D

3D目标检测 – 简单的摄像机模型

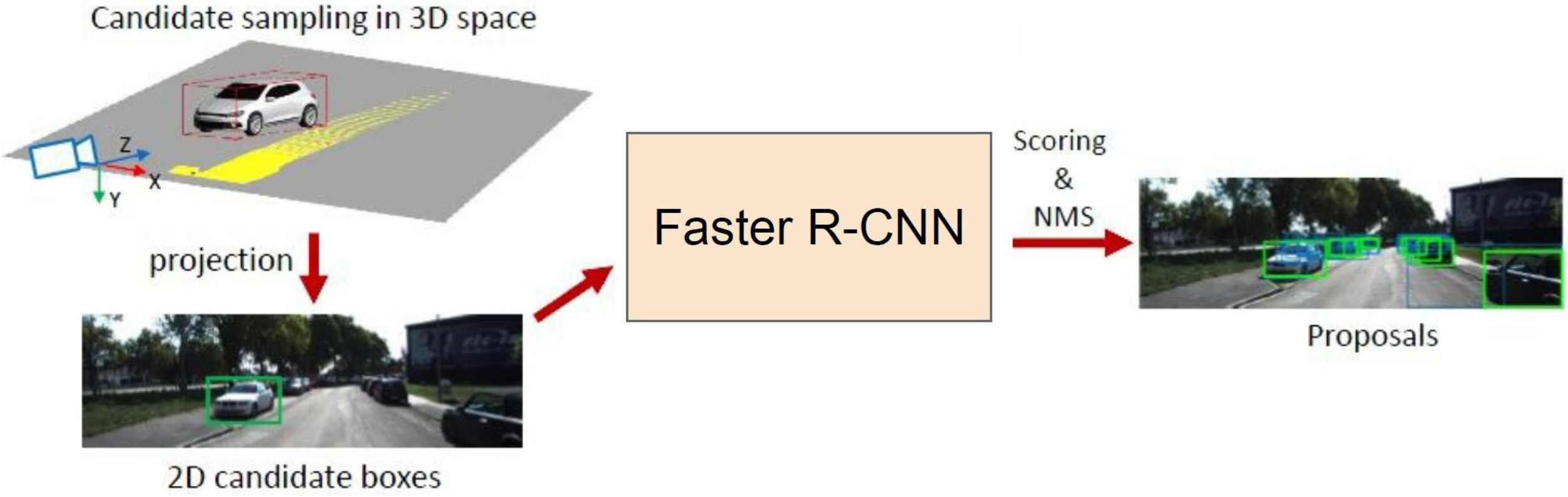


- 图像平面(image plane)上的一个2D point对应于3D空间中的一条射线(3D ray)
- 图像平面中的2D边界框对应于3D空间的一个截头锥体
- 在3D空间中定位一个对象：

  对象可以出现在摄像机视角上的截头锥体的任意位置。

6. Beyongd 2D

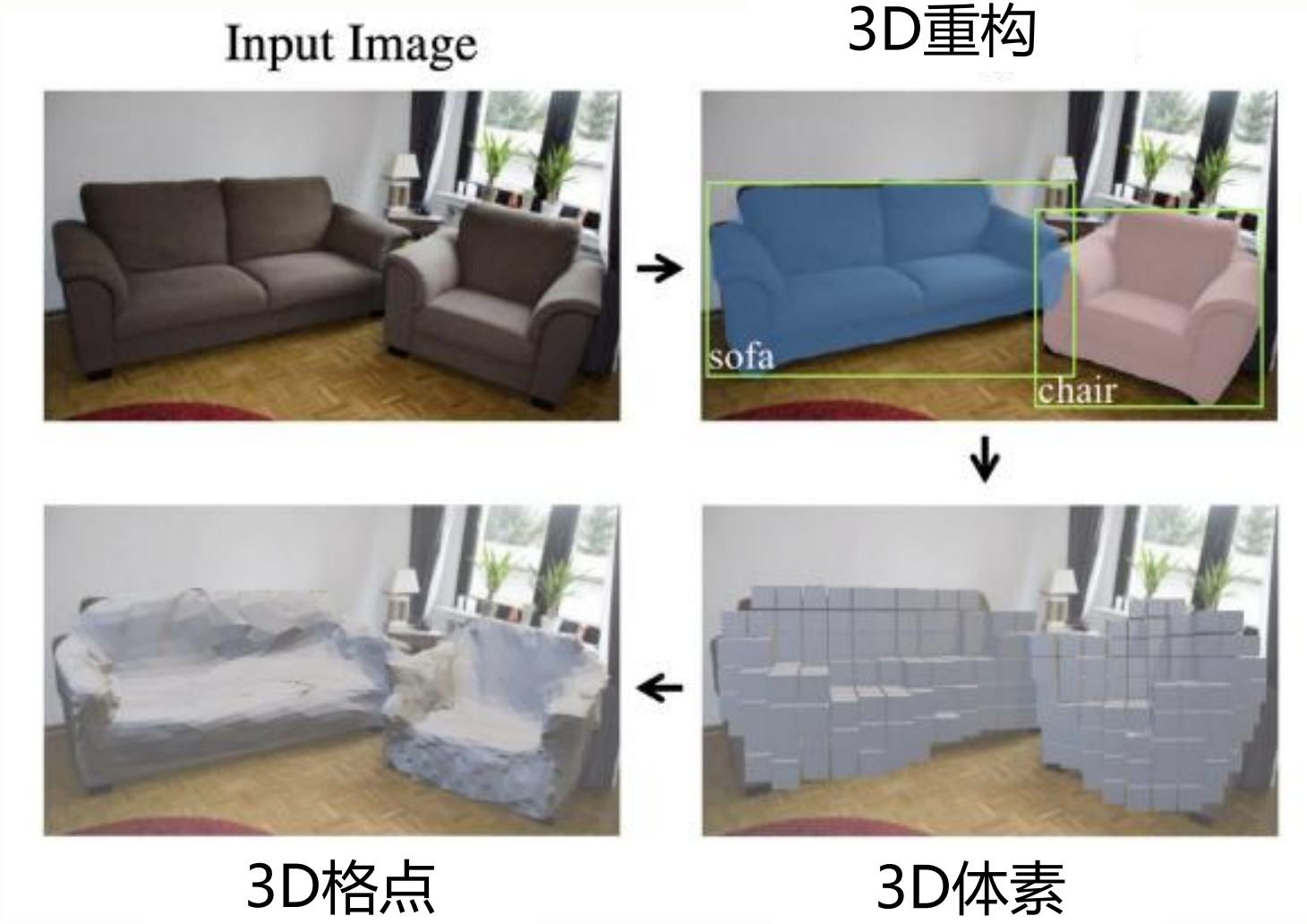
3D目标检测 – 简单的摄像机模型



- 思路与Faster RCNN类似，但使用3D边界框
- 3D区域建议+类别分数

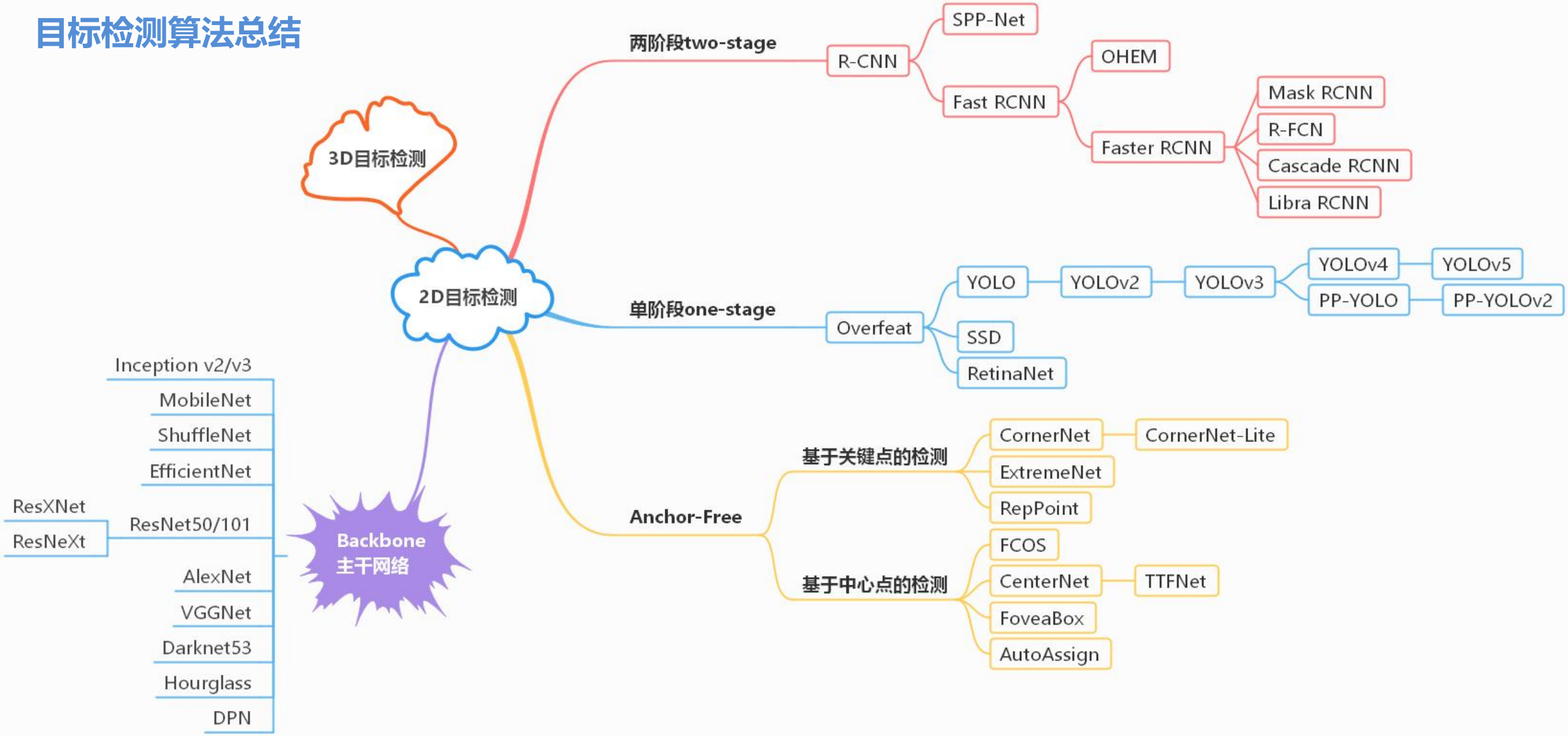
6. Beyongd 2D

3D模型预测



第09讲 基于深度学习的目标检测

目标检测算法总结



第09讲 基于深度学习的目标检测

【竞赛02】目标检测综合竞赛

目标：基于VOC数据集实现目标检测

Pascal VOC数据集是最常见的目标检测和图像分割基准数据集，常见的Pascal VOC数据集包括VOC2007和VOC2012. 它有四大类，二十个类别，四大类别。

竞赛描述：

基于PaddleDetection中实现任意一种目标检测算法，在测试集上获得最有的mAP值。

分数评定：

- ✓ 参与竞赛，期末总分奖励2分
- ✓ 高于Baseline的成绩，期末总分奖励3分
- ✓ 额外奖励：第一名5分，第二名3分，第三名2分
- ✓ 应用到学科（互联网+）竞赛，获奖队伍所有成员奖励10分

欧老师的联系方式

读万卷书 行万里路 只为最好的修炼

QQ：14777591 (宇宙骑士)

Email：ouxinyu@alumni.hust.edu.cn

Website: http://ouxinyu.cn

Tel：18687840023

地址：安宁校区 诚远楼201

南院 智能应用研究院A306-2