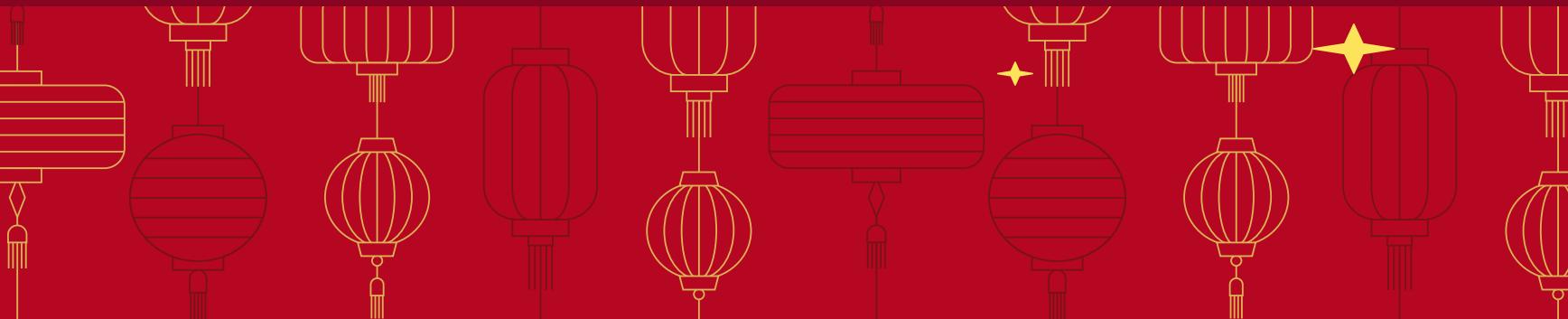




# 论文阅读2025.1

**TAKE IT EASY**



精度优先 113开会 需解决的问题

给出 OD / segment 的明确定义

对自己要做什么比较清楚

建议最应用目标检测完成  
更简单.

经典工作、技术路线 (Mine)

方法最新工作、

# YOLO - one-stage 单阶段 <晚于 Fast RCNN >

两阶段 = repurpose classifiers to perform detection.

YOLO = frameOD as a regression problem to spatially separated bounding boxes and associate class possibilities. 将目标检测视为回归问题

完整图像 → 单网络 → 预测边界框坐标、类别概率  
end2end

fps = frame per second

## Intro

2-stage = DPM 滑动窗口检测每处 + classifier

RCNN 区域候选框 + classifier

问题 = 速度慢，分离单独组件 → 难以优化 optimize

YOLO = 将目标检测视为单个回归问题

image pixels → 单网络 → bounding box coordinates  
class possibilities

Method:

将 img 分成  $S \times S$  grid 网格 - 若对象中心落入网格，则预测.

Each grid = 预测  $B$  个边界框 置信度分数 反映模型对 box 信心以及它认为预测这些 box 的准确度

$$\text{Confidence} = \Pr(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$$

因此预测 =

$$(x, y, w, h) + \text{confidence}$$



$$\text{类別概率} = \Pr(\text{class}_i | \text{Object})$$

"We only predict one set of class probabilities per grid cell,  
regardless of the number of boxes  $B$ "

$$\frac{\Pr(w_i | x) \Pr(x)}{\text{该类概率}} \times \text{IOU} = \Pr(w_i) \times \text{IOU}$$

单独 box 置信预测



$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)}$$

$$\therefore P(w_i|x)P(x) = P(x|w_i)P(w_i) \rightarrow P(w_i)$$

因为confidence 应该是  $\frac{\text{数据分布}}{\text{先验}}$ ?

$IoU \uparrow$  越置信、 $P(x|w_i) \rightarrow 1$  ?

对 each box 有：特定类别的置信得分为：① 碰碰 box 内出现各类别概率  
 ② 预测框 box 与 object fits 多好

$$\text{Output} = \text{Tensor} = S \times S \times (B \times 5 + C)$$

$\xrightarrow{S \times S}$  grid cell       $\xrightarrow{5}$   $B$  个边界框预测  $(x, y, w, h)$  + confidence + 每类类条件  $P_r$

网络结构 = 24 层卷积 + 2 层全连接

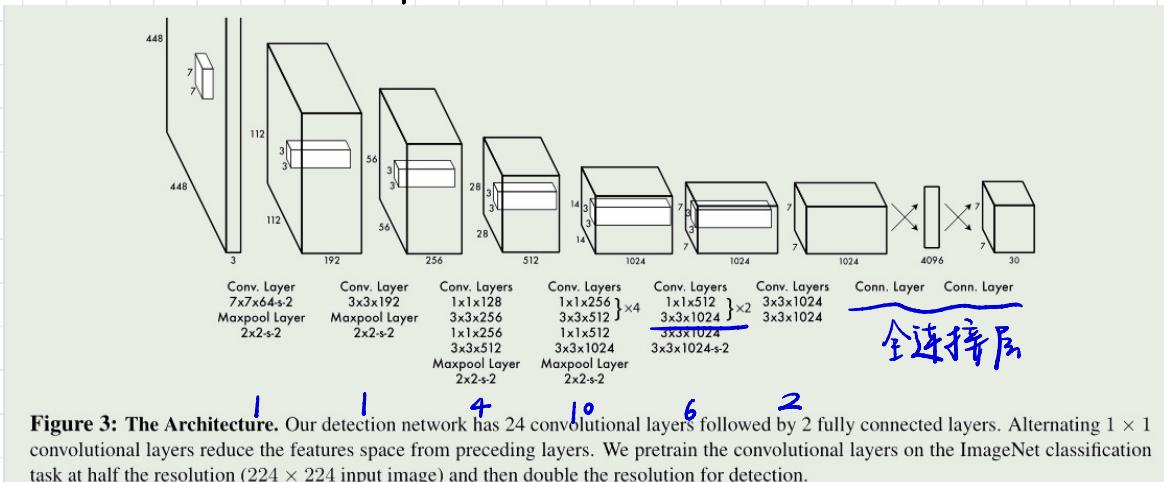


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

Fast YOLO = 9 层卷积 + 4 filters

Training = 对前 20 层卷积预训练 (2 magnet)

从 PASCAL VOC 看似 =

$$s \times s \times (B \times 5 + C) = 7 \times 7 \times 30$$

$\uparrow$        $\uparrow$        $\uparrow$   
 7      7      30

网络输出为  $7 \times 7 \times 30$  (全部归一化 0~1)

存在问题：① sum square error  
平方和误差

↓  
对齐不好 = 不 fit 优化目标 <最大化平均精度>

② 大部分 grid cells 中不包含 object

↓ 置信度  $\rightarrow 0$   
压制包含 object 的 cells 的精度

↓ 模型不稳定

↓ 早期出现分歧

Solution:

赋予权重

明示 = YOLO 在每个 grid cell 都会预测两个 box

在训练阶段 = (没太理解问题)

We only want one bounding box predictor to be responsible for each object.

YOLO 预测每个网格单元的多个边界框。在训练时，我们只希望一个边界框预测器负责每个对象。我们令每个预测器“负责”来预测一个对象，该对象根据该对象预测与基本事实的当前 IOU 最高。这导致边界框预测器之间的专业化。每个预测器在预测对象的某些大小、纵横比或类别方面做得更好，提高了整体召回率。

loss function = sum square error 平方和误差

cell i 的第 j 个 box 预测器 = "responsible" for that pred.

有 obj  
区域

$$\text{入 coord} \sum_{i=0}^S \sum_{j=0}^B \underbrace{\mathbb{1}_{ij}^{obj}}_{\text{cell i 的第 j 个 box 预测器}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \text{入 coord}$$

(x,y) 误差

$$\sum_{i=0}^S \sum_{j=0}^B \underbrace{\mathbb{1}_{ij}^{obj}}_{\text{cell i 的第 j 个 box 预测器}} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

(w,h) 误差

$$+ \sum_{i=0}^S \sum_{j=0}^B \underbrace{\mathbb{1}_{ij}^{obj}}_{\text{分类误差}} (c_i - \hat{c}_i)^2 + \text{入 noobj} \sum_{i=0}^S \sum_{j=0}^B \underbrace{\mathbb{1}_{ij}^{noobj}}_{\text{obj 是否出现在 cell i 中}} (c_i - \hat{c}_i)^2 + \sum_{i=0}^S \underbrace{\mathbb{1}_i^{obj}}_{\text{classes}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

S 如果该网格单元中存在对象，损失函数只会惩罚分类错误（因此前面讨论的条件类概率）。如果该预测器对地面真值框“负责”（即该网格单元中任何预测器的IOU最高），它也只执行边界框坐标误差。

ground truth box (IOU max)

Expr = dataset metric

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	<b>155</b>
YOLO	2007+2012	<b>63.4</b>	45

Less Than Real-Time	Train	mAP	FPS
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

**Table 1: Real-Time Systems on PASCAL VOC 2007.** Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

对比实验结果

### 4.3. Combining Fast R-CNN and YOLO

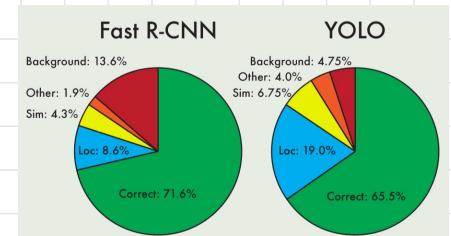
	mAP	Combined	Gain
Fast R-CNN	-	71.8	-
Fast R-CNN (2007 data)	<b>66.9</b>	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	<b>75.0</b>	<b>3.2</b>

**Table 2: Model combination experiments on VOC 2007.** We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Error 分析 = 按 IOU 值分类 =

- Correct: correct class and  $\text{IOU} > .5$
- Localization: correct class,  $.1 < \text{IOU} < .5$
- Similar: class is similar,  $\text{IOU} > .1$
- Other: class is wrong,  $\text{IOU} > .1$
- Background:  $\text{IOU} < .1$  for any object

在 20 个类别中对每种错误的  
细分、训练图分析。



**Figure 4: Error Analysis: Fast R-CNN vs. YOLO** These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

VOC 2012 上结果 (其实在 2017)

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	car	cat	chair	cow	table	dog	horse	mbike	personplant	sheep	sofa	train	tv		
MR-CNN-MORE-DATA [11]	<b>73.0</b>	<b>85.5</b>	<b>82.9</b>	<b>76.6</b>	<b>57.8</b>	<b>62.7</b>	<b>79.4</b>	<b>77.2</b>	<b>86.6</b>	<b>79.1</b>	<b>68.2</b>	<b>87.0</b>	<b>83.4</b>	<b>84.7</b>	<b>78.9</b>	<b>45.3</b>	<b>72.4</b>	<b>65.8</b>	<b>89.3</b>		
HyperNet-VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	<b>79.8</b>	87.7	49.6	74.9	52.1	86.0	81.7	<b>81.8</b>	<b>48.6</b>	<b>73.5</b>	<b>59.4</b>	79.9	65.7	
HyperNet-S	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	84.4	73.2	59.3	79.7	61.2	77.7	
<b>Fast R-CNN + YOLO</b>	<b>70.7</b>	83.4	78.5	73.5	55.8	43.4	79.1	73.1	<b>89.4</b>	49.4	75.5	57.0	<b>87.5</b>	80.9	81.0	74.7	41.8	71.5	68.5	<b>82.1</b>	
MR-CNN-LCNN [11]	70.7	85.0	79.6	71.5	55.3	57.2	76.0	73.9	88.5	45.6	77.1	55.3	89.8	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Faster R-CNN [11]	70.4	84.9	79.8	74.3	53.9	47.9	77.5	75.9	88.5	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
DEEP-ENS.COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	<b>68.8</b>	75.9	71.4
NoC [28]	68.8	82.0	79.0	71.6	53.7	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	76.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	<b>87.5</b>	80.8	82.0	72.0	35.1	68.3	65.7	80.4	64.2
UMICH-FGS-STRUCT	66.4	82.9	76.1	64.1	44.6	49.1	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS-NIC_2000 [7]	63.8	80.2	73.8	61.9	43.7	43.6	70.3	67.6	80.7	41.1	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.4	72.4	61.3	45.7	42.1	68.2	66.8	80.2	40.4	70.0	49.8	79.6	77.9	74.0	65.7	35.2	68.7	62.6		
NUS-NIC	62.4	77.6	73.1	62.6	39.5	43.6	69.1	66.4	79.1	39.1	68.5	50.4	77.2	71.3	76.7	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN-VGG BB [13]	62.4	79.4	72.7	64.0	41.2	42.6	70.0	69.0	80.9	41.2	70.2	57.2	82.0	74.8	76.0	65.6	35.6	64.2	60.7	60.3	
R-CNN-VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	79.6	74.6	76.0	64.0	31.6	63.4	54.2	63.5	50.7
<b>YOLO</b>	<b>57.9</b>	70.0	67.2	57.7	33.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	73.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	80.8	30.8	65.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN-BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.8	73.6	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	53.3	49.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7	
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

**Table 3: PASCAL VOC 2012 Leaderboard.** YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

消融实验(?)

还有这个能力测试：在其他 dataset 中 run 取 VOC子集 train，比拼结果。

Combined & Gain何意？

如何结合？

结合不同模型？

结合不同模型何意？

# 定性 vs 定量

## ► 定性分析 (Qualitative Research)

定性分析也称质化研究，是社会科学领域的一种基本研究范式<sup>+</sup>，也是科学的重要步骤和方法之一。

定性分析通常是在观察和解释的基础上深入分析。

## ► 定量分析 (Quantitative Research)

定量分析是与定性分析相对的概念，要考察和研究事物的量，就需用数学工具<sup>+</sup>对事物进行数量的分析，这就叫定量的研究，也称量化研究<sup>+</sup>，是社会科学领域的一种基本研究范式，也是科学的重要步骤和方法之一。

定量分析是指确定事物某方面量的规定性的科学研究，就是将问题与现象用数量来表示，进而去分析、考验、解释，从而获得意义的研究方法和过程。如果没有数字，那肯定不是定量分析。

YOLO V1 → V11

### YOLOv1

- 发布日期: 2016年6月
- 作者: Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
- 论文: "YOLO: You Only Look Once: Unified, Real-Time Object Detection"
- 主要优化点:
  - 将目标检测任务转化为单次前向传播问题, 显著提升检测速度
  - 能够以45 FPS的速度处理图像, 有一个更快的版本可以达到155 FPS
  - 限制: 在小物体检测上的精度较差, 且定位误差较高

### YOLOv2 (YOLO9000)

- 发布日期: 2017年12月
- 作者: Joseph Redmon, Ali Farhadi
- 论文: "YOLO9000: Better, Faster, Stronger"
- 主要优化点:
  - 能够检测9000种类别物体
  - 多尺度训练增强模型鲁棒性
  - 引入anchor boxes改进对小物体的检测能力

### YOLOv3

- 发布日期: 2018年4月
- 作者: Joseph Redmon, Ali Farhadi
- 论文: "YOLOv3: An Incremental Improvement"
- 主要优化点:
  - 引入Darknet-53作为主干网络, 结合残差网络提高检测精度
  - 多尺度预测改善对小物体的检测
  - 取消软分类器, 使用独立的二元分类器提高性能

### YOLOv4

- 发布日期: 2020年4月
- 作者: Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao
- 论文: "YOLOv4: Optimal Speed and Accuracy of Object Detection"
- 主要优化点:
  - 提出Bag of Freebies和Bag of Specials优化策略, 提高模型精度
  - CSPDarknet53更高效的主干网络, 提升网络推理速度和精度
  - 引入CIOU损失函数提高边界框回归性能

### YOLOv5

- V4 与 V5
- 发布日期: 2020年6月
  - 作者: Glenn Jocher
  - 无论文发表, 开源地址: [github.com/ultralytics/...](https://github.com/ultralytics/)
  - 主要优化点:
    - YOLOv5转向Pytorch框架, 便于开发者使用和扩展
    - 自适应的anchor box学习机制提高检测效率
    - 提供多种尺寸的预训练模型满足不同场景需求

### YOLOv6

V6 与 V7

- 发布日期: 2022年6月
- 作者: 美团技术团队
- 论文: "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications"
- 主要优化点:
  - 针对行业应用优化, 尤其注重推理速度
  - 引入EfficientRep带来更高效的网络架构
  - 优化模型部署性能, 适合工业环境中的大规模应用

### YOLOv7

- 发布日期: 2022年7月
- 作者: Wong Kin-Yiu, Alexey Bochkovskiy, Chien-Yao Wang
- 论文: "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors"
- 主要优化点:
  - 在COCO数据集上达到新的速度与精度平衡
  - 跨尺度特征融合提高对不同尺度物体的检测能力
  - 改进训练过程中的标签分配方式提高训练效率

### YOLOv8

- 发布日期: 2023年1月
- 作者: Ultralytics团队
- 无论文发表, 开源地址: [github.com/ultralytics/...](https://github.com/ultralytics/)
- 主要优化点:
  - 提供可定制的模块化设计方便用户根据需求进行扩展
  - 内置多种训练和超参数优化策略简化模型调优过程
  - 集成检测、分割和跟踪功能

### YOLOv9

- 发布日期: 2024年2月
- 作者/贡献者: WongKinYiu等
- 论文: "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information"
- 主要优化点:
  - 可编程梯度信息(PGI)+广义高效层聚合网络(GELAN)。
  - 与YOLOv8相比, 其出色的设计使深度模型的参数数量减少了49%, 计算量减少了43%, 但在MS COCO数据集上仍有0.6%的AP改进。

### YOLOv10

- 发布日期: 2024年5月
- 作者: 清华大学
- 论文: "YOLOv10: Real-Time End-to-End Object Detection"
- 主要优化点:
  - 实时端到端的对象检测, 主要在速度和性能方面的提升

### YOLOv11

- 发布日期: 2024年9月
- 作者: Ultralytics团队
- 无论文发表, 开源地址: [github.com/ultralytics/...](https://github.com/ultralytics/)
- 主要优化点:
  - YOLOv11继承自YOLOv8, 在YOLOv8基础上进行了改进, 使同等精度下参数量降低20%, 在速度和准确性方面具有无与伦比的性能。
  - 其流线型设计使其适用于各种应用, 并可轻松适应从边缘设备到云API等不同硬件平台。
  - 使其成为各种物体检测与跟踪、实例分割、图像分类和姿态估计任务的绝佳选择。

one-stage =

Fcos: anchor 基点

↓  
centerNet (Object as point) & Reppoint  
一个物体一个匹配样本

↓ Free Anchor = Recall & Precision 监督样本定义

image、backbone、head 出分类+回归、指标注直接监督训练

Two-stage =

Reppoint + AlignDet  $\text{input} = \text{ROI Pooling} / \text{ROI Align} = \text{Deformable conv}$

backbone + head 看成一体

NAS-DENAS-FPNNAS (FPN大物体掉豆小物体涨豆，整体涨豆)

TridentNet (大、中、小物体均涨豆)

