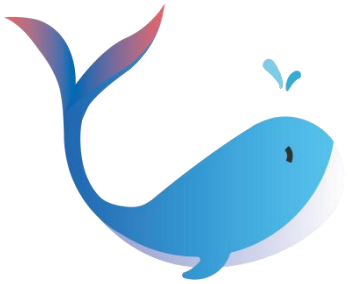


YOLO_v4 论文分享

论文： *https://arxiv.org/pdf/2004.10934.pdf*

代码： https://github.com/Tianxiaomo/pytorch-YOLOv4

以若

YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

目录

1. YOLO V4论文介绍
2. 创新点
3. 改进点代码解读

YOLOV4: Introduction

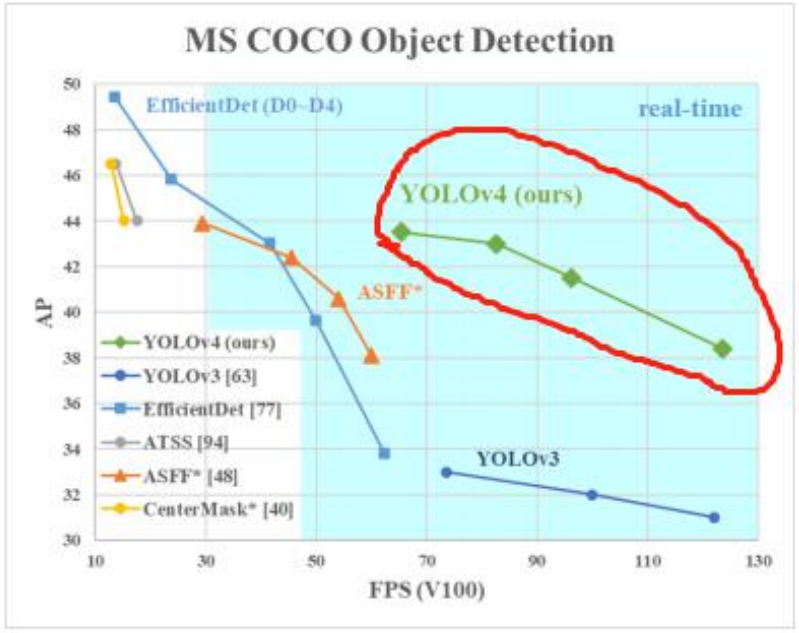


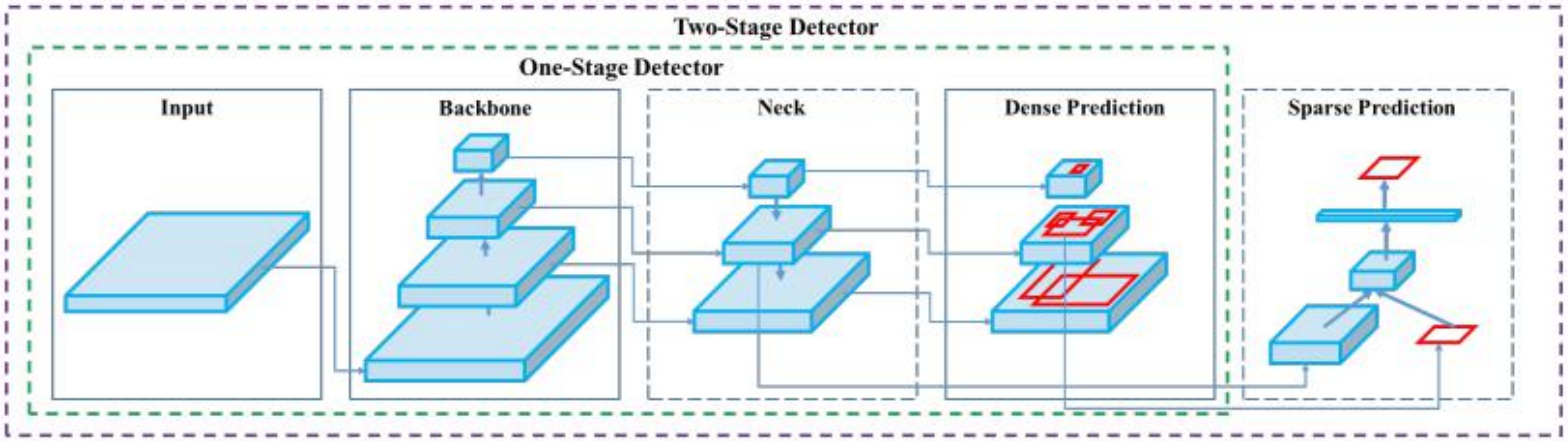
Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

研究目的： 设计一个速度快，易部署，且训练简单的 目标检测算法

主要贡献：

1. 开发了一个简单且高效的目标检测算法（YOLO- V4），该算法可通过普通的GPU（1080TI或者2080Ti） 来训练。
2. 作者验证了在目标检测算法训练过程中不同的技巧 tricks对实验性能的影响，这些tricks主要包括Bag-of- Freebies和 Bag-of-Specials。
3. 作者修改了一些state-of-the-art的算法，使得这些算 法适用于单GPU上训练，这些算法包括了CBN，PAN 和SAM等等。

YOLOV4: Related Work



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

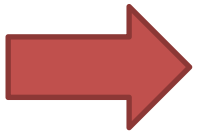
Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

Figure 2: Object detector.

Object detection = Backbone + Neck + Head

YOLOV4: Related Work

- **Input**: Image, Patches, Image Pyramid
- **Backbones**: VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]
- **Neck**:
  - **Additional blocks**: SPP [25], ASPP [5], RFB [47], SAM [85]
  - **Path-aggregation blocks**: FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]
- **Heads:**:
  - **Dense Prediction (one-stage)**:
    - RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based)
    - CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
  - **Sparse Prediction (two-stage)**:
    - Faster R-CNN [64], R-FCN [9], Mask R-CNN [23] (anchor based)
    - RepPoints [87] (anchor free)



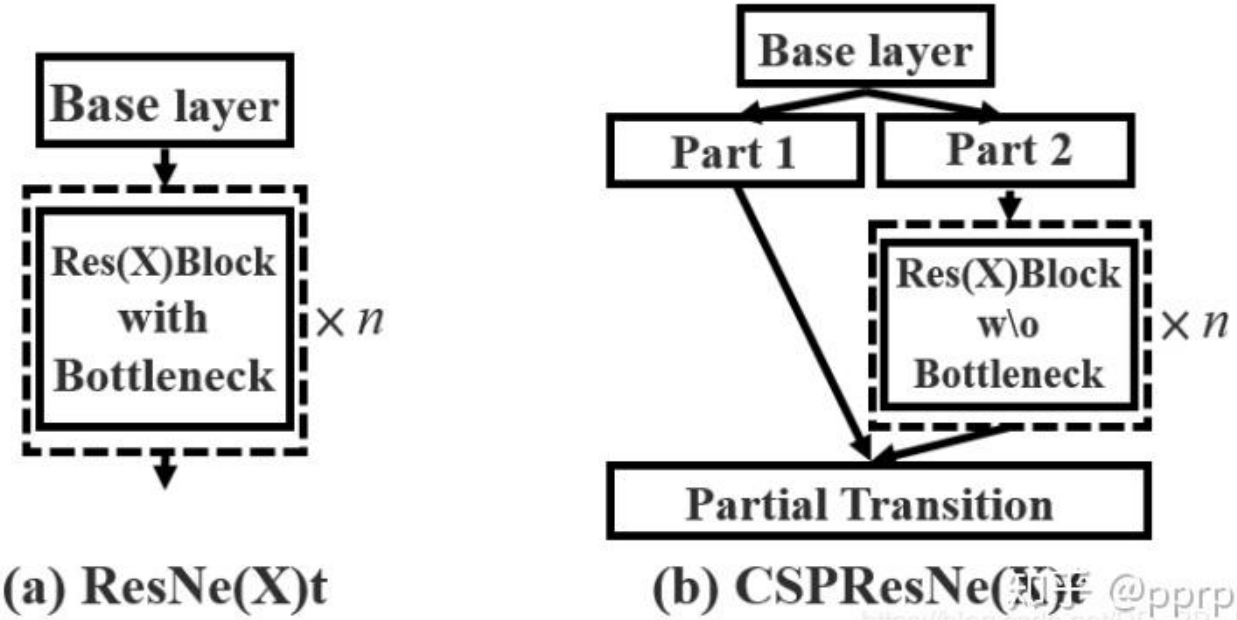
YOLOv4 consists of:

- Backbone: CSPDarknet53 [81]
- Neck: SPP [25], PAN [49]
- Head: YOLOv3 [63]

YOLOv4 = CSPDarknet53+SPP+PAN+Y

YOLOV4: Related Work

CSPNet



CSPNet

- CSP： Cross stage partial
- 增强CNN的学习能力，能够在轻量化的同时保持准确性。
- 降低计算瓶颈
- 降低内存成本

YOLOV4: Related Work

SPP： **Spatial Pyramid Pooling** 空间金字塔池化

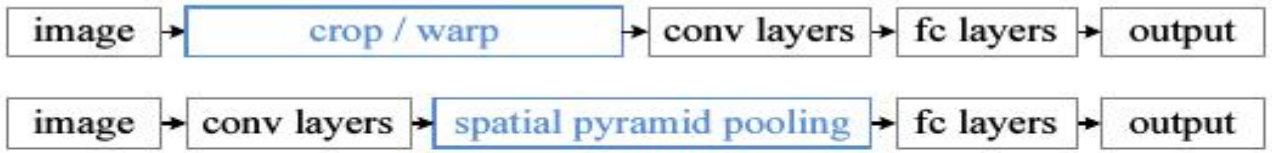
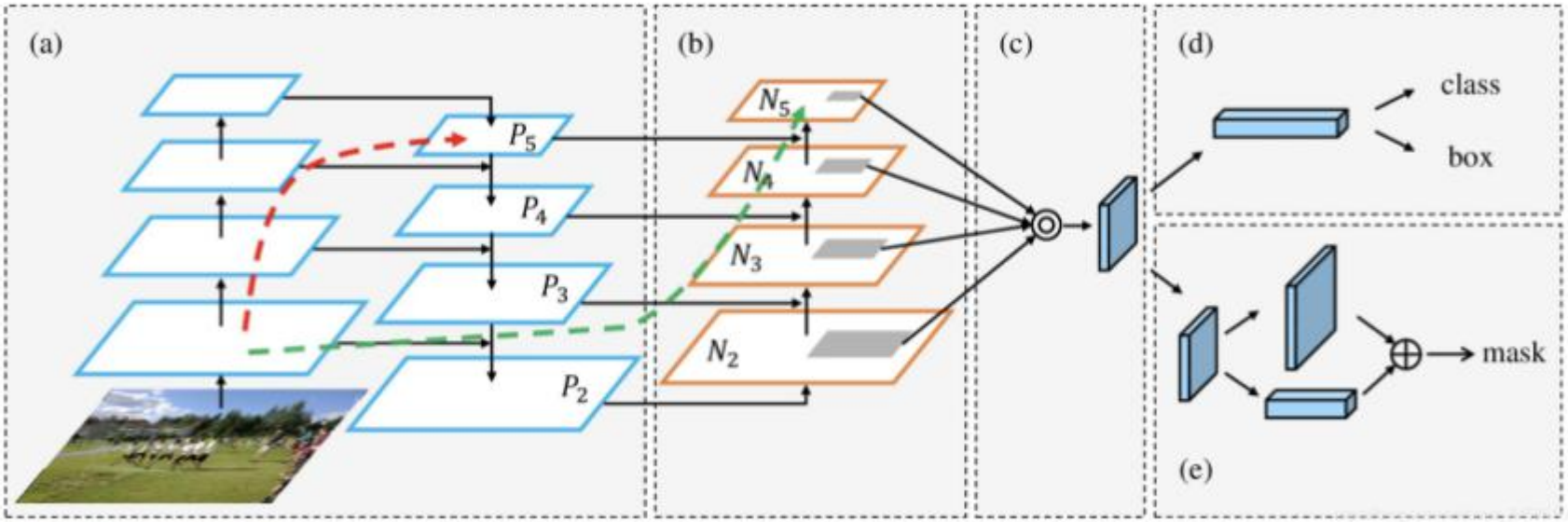


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

- 在CNN的最后一个卷积层之后加入一个SPP层
- 对之前卷积得到的特征进行”整合”(aggregation)
- 得到一个固定长度的特征向量,再传到全连层

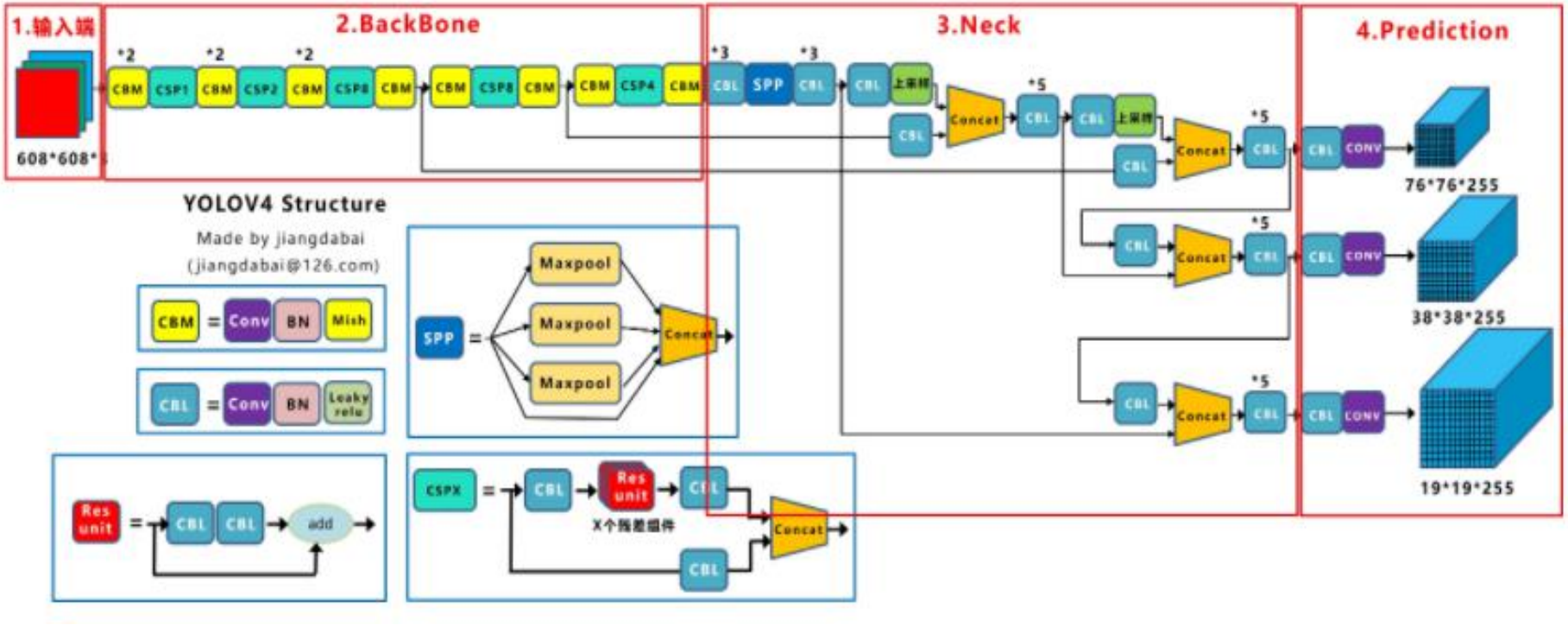
YOLOV4: Related Work

PAN： Path Aggregation Network



- 自底向上的路径增强，缩短信息传播路径,利用低层特征的精准定位信息
- 动态特征池化，每个proposal利用金字塔所有层的特征，为了避免proposal的随意分配，对分类及定位更加有利。
- 全连接层融合，为了给掩码预测增加信息来源
- Addition改成concat

YOLOV4: Structure



YOLOV4: Tricks---- Bag of freebies

什么叫Bag of freebies？字面上的意思就是免费赠品。

YOLOV4: Tricks---- Bag of freebies

数据增强方法

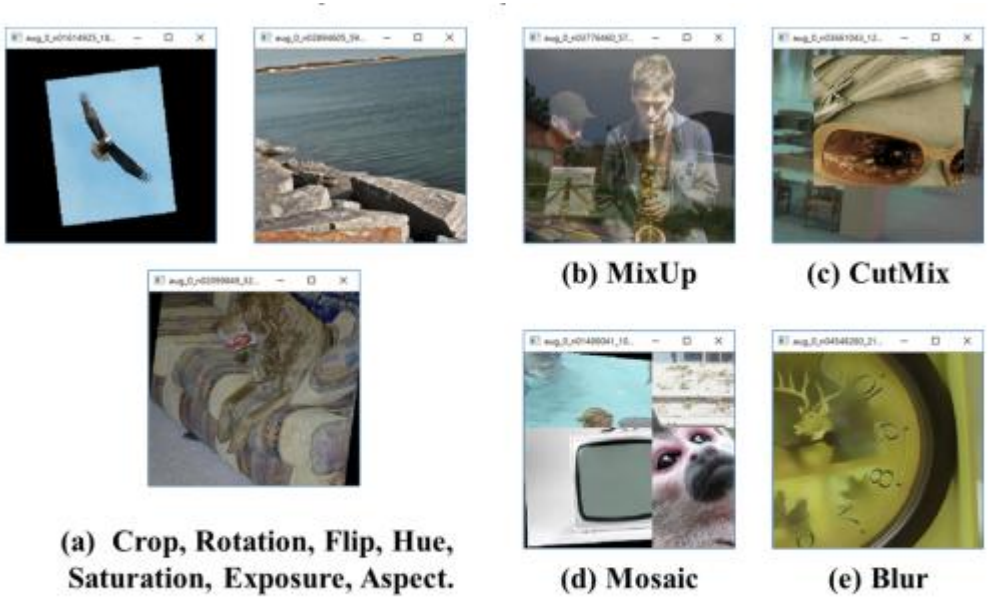


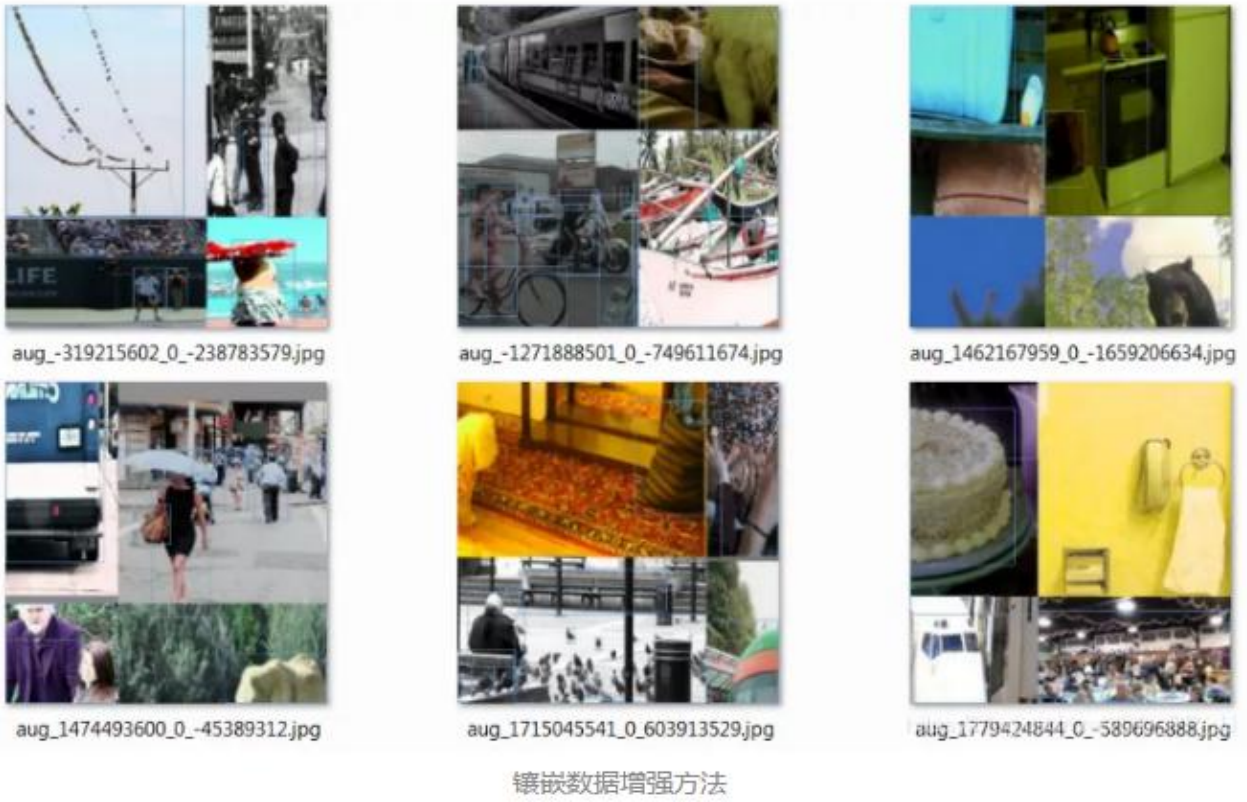
Figure 7: Various method of data augmentation.

YOLOV4: Tricks---- Bag of freebies

Mosaic数据增强方法

•混合四张具有不同语义信息的图片，可以让检测器检测超出常规语境的目标，增强模型的鲁棒性。

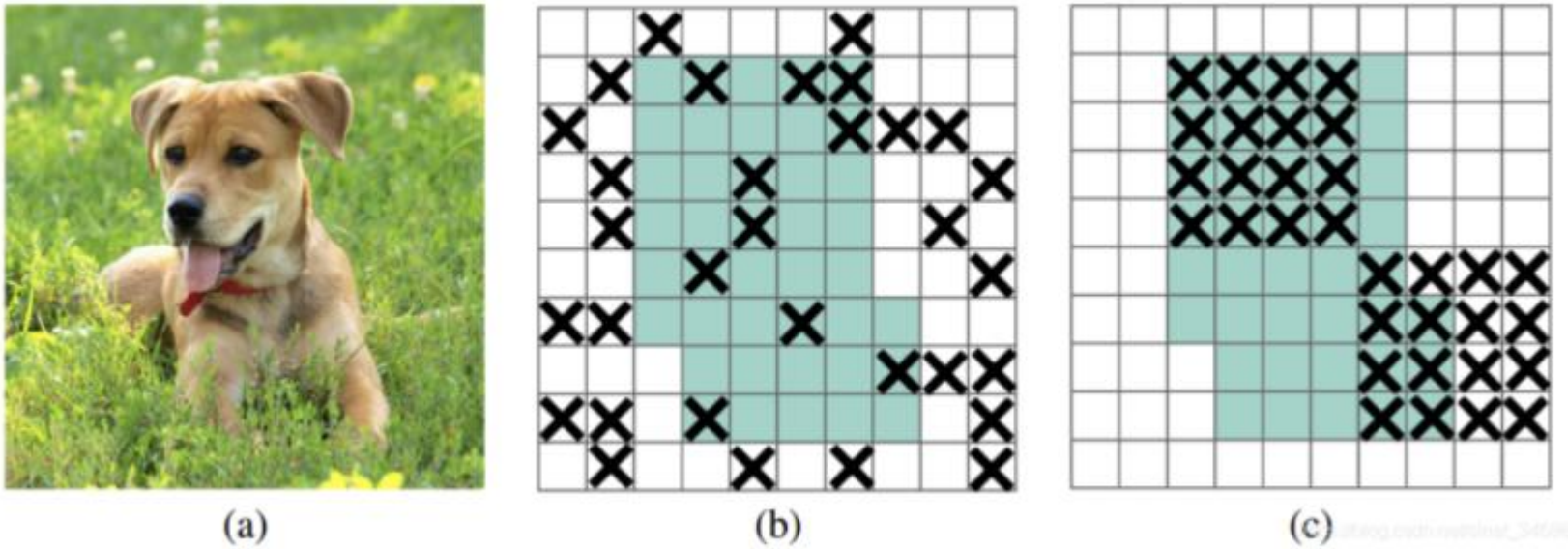
•由于BN是从四张图片计算得到的，所以可以减少对大的mini-batch的依赖。



镶嵌数据增强方法

YOLOV4: Tricks---- Bag of freebies

DropBlock



- 在featuremap上去一块一块的找，进行归零操作，类似于dropout

- 参数:
- block_size，用来控制进行归零的block大小;
- γ，用来控制每个卷积结果中，到底有多少个channel要进行dropblock;
- keep_prob，作用和dropout里的参数一样;

YOLOV4: Tricks---- Bag of freebies

Label Smoothing

对标签平滑化处理以防止过拟合

$$p = (1 - \epsilon)y + \frac{\epsilon}{K}$$

```
def label_smoothing(inputs, epsilon=0.1):
    K = inputs.get_shape().as_list()[-1]    # number
    return ((1-epsilon) * inputs) + (epsilon / K)
```

YOLOV4: Tricks---- Bag of freebies

Ciou-Loss

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v$$

- 引入一个box长宽比的惩罚项
- 考虑了box的长宽比

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2$$

$$\alpha = \frac{v}{(1 - IoU) + v}$$

CmBN

- **Cross mini-Batch Normalization**
- 收集单个批中的小批之间的统计信息

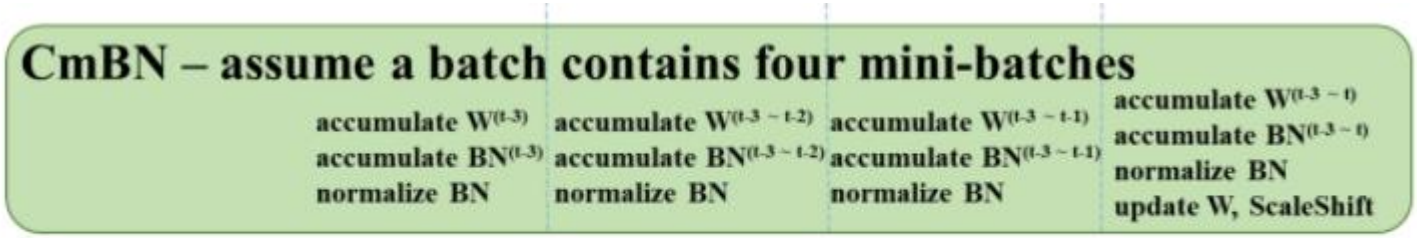


Figure 4: Cross mini-Batch Normalization.

- `IOU Loss:`

$$\mathcal{L}_{IoU} = 1 - \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$$

- `GIoU Loss:` 引入了一个惩罚项,其中C表示能覆盖候选框和真实框的最小box.由于惩罚项的引入，在不重叠的情况下，预测框会向目标框移动。

$$\mathcal{L}_{GIoU} = 1 - IoU + \frac{|C - B \cup B^{gt}|}{|C|}$$

- `DIoU Loss:` 出现下面情况时，GIoU Loss完全降级成IoU Loss，因此引入DIoU Loss，DIoU Loss是在IoU Loss基础上引入一个惩罚项。b,b^{gt}是B，B^{gt}的中心点，$\rho(\cdot)$为欧氏距离，c为覆盖B，B^{gt}最小box的对角线长度。惩罚使中心点距离最小。

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$

$$\mathcal{R}_{CIoU} = \frac{\rho^2\left(\mathbf{b}, \mathbf{b}^{gt}\right)}{c^2} + \alpha v$$ 其中 α 是权重函数，

而 ν 用来度量长宽比的相似性，定义为 $v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2$

YOLOV4: Tricks---- Bag of Specials

Mish：新的深度学习激活函数

$$f(x) = x \cdot tanh(ln(1 + e^x))$$

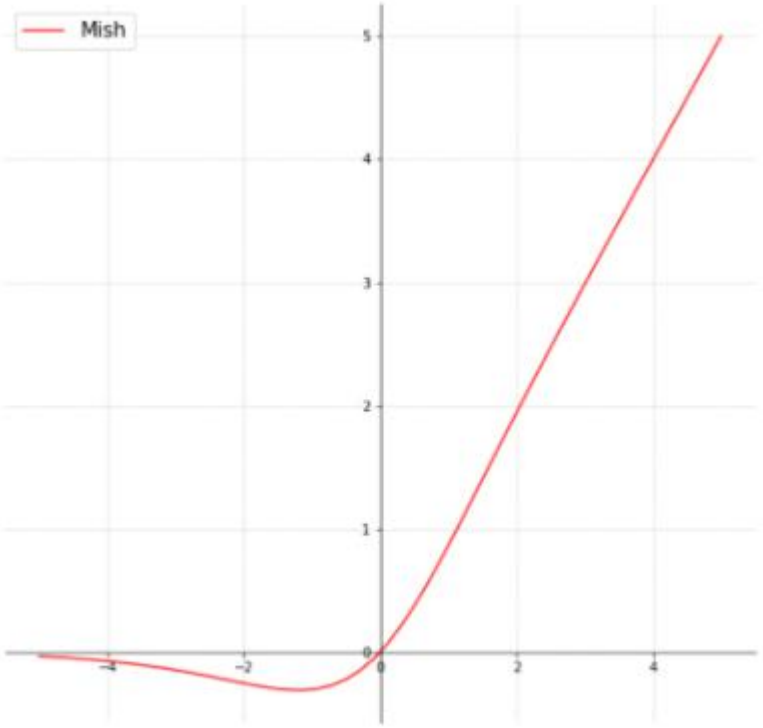


Figure 1. Mish Activation Function

MiWRC:

- Weighted-Residual-Connections
- 加权残差连接

YOLOV4: Tricks---- Bag of Specials

SAM

- Spatial attention module
- 从空间上的attention修改为点上的attention

DIoU-NMS

- 用DIoU替换IoU

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$

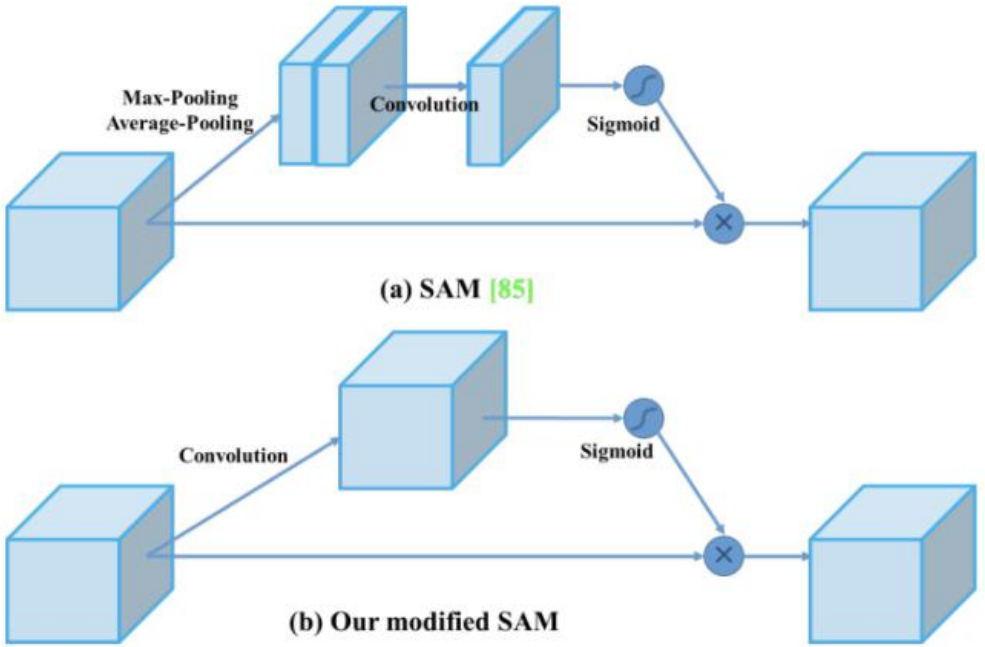


Figure 5: Modified SAM.