YOLOv4: Optimal Speed and Accuracy of Object Detection

YOLOv4：目标检测最优速度和精度

Alexey Bochkovskiy*	**Chien-Yao Wang***	**Hong-Yuan Mark Liao**
alexeyab84@gmail.com	Institute of Information Science Academia Sinica, Taiwan kinyiu@iis.sinica.edu.tw	Institute of Information Science Academia Sinica, Taiwan liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP_{50}) for the MS COCO dataset at a real-time speed of ~65 FPS on Tesla V100. Source code is at https://github.com/AlexeyAB/darknet.

摘要

有大量的技巧可以提高卷积神经网络（CNN）的精度。需要在大数据集下对这种技巧的组合进行实际测试，并需要对结果进行理论论证。某些技巧仅在某些模型上使用和专门针对某些问题，或只针对小规模的数据集；而一些技巧，如批处理归一化、残差连接等，适用于大多数的模型、任务和数据集。我们假设这种通用的技巧包括加权残差连接（Weighted-Residual-Connection，WRC）、跨小型批量连接(Cross-Stage-Partial-connection，CSP)、Cross mini-Batch Normalization（CmBN）、自对抗训练（Self-adversarial-training，SAT）和 Mish 激活函数。我们在本文中使用这些新的技巧：WRC、CSP、CmBN、SAT，Mish-activation，Mosaic data augmentation、CmBN、DropBlock 正则化和 CIoU 损失，以及这些技巧的组合，在 MS COCO 数据集达到目前最好的结果：43.5%的 AP（65.7% AP_{50}），在 Tesla V100 上速度达到约 65FPS。源码见：https://github.com/AlexeyAB/darknet.

1. Introduction

The majority of CNN-based object detectors are largely applicable only for recommendation systems. For example, searching for free parking spaces via urban video cameras is executed by slow accurate models, whereas car collision warning is related to fast inaccurate models. Improving the real-time object detector accuracy enables using them not only for hint generating recommendation systems, but also for stand-alone process management and human input reduction. Real-time object detector operation on conventional Graphics Processing Units (GPU) allows their mass usage at an affordable price. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size. We address such problems through creating a CNN that operates in real-time on a conventional GPU, and for which training requires only one conventional GPU.

1. 引言

大多数基于 CNN 的目标检测器基本上都仅适用于推荐系统。例如：通过城市摄像头寻找免费停车位，它由精确的慢速模型完成，而汽车碰撞警报需要由快速、低精度模型完成。改善实时目标检测器的精度，使其能够不仅可以用于提示生成推荐系统，也可以用于独立的流程管理和减少人力投入。传统 GPU 使得目标检测可以以实惠的价格运行。最准确的现代神经网络不是实时运行的，需要大量的训练的 GPU 与大的 mini bacth size。我们通过创建一个 CNN 来解决这样的问题，在传统的 GPU 上进行实时操作，而对于这些训练只需要一个传统的 GPU。

The main goal of this work is designing a fast operating speed of an object detector in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). We hope that the designed object can be easily trained and used. For example, anyone who uses a conventional GPU to train and test can achieve real-time, high quality, and convincing object detection results, as the YOLOv4 results shown in Figure 1. Our contributions are summarized as follows:

1. We develope an efficient and powerful object detection model. It makes everyone can use a 1080 Ti or 2080 Ti GPU to train a super fast and accurate object detector.

2. We verify the influence of state-of-the-art Bag-ofFreebies and Bag-of-Specials methods of object detection during the detector training.

3. We modify state-of-the-art methods and make them more effecient and suitable for single GPU training, including CBN [89], PAN [49], SAM [85], etc.

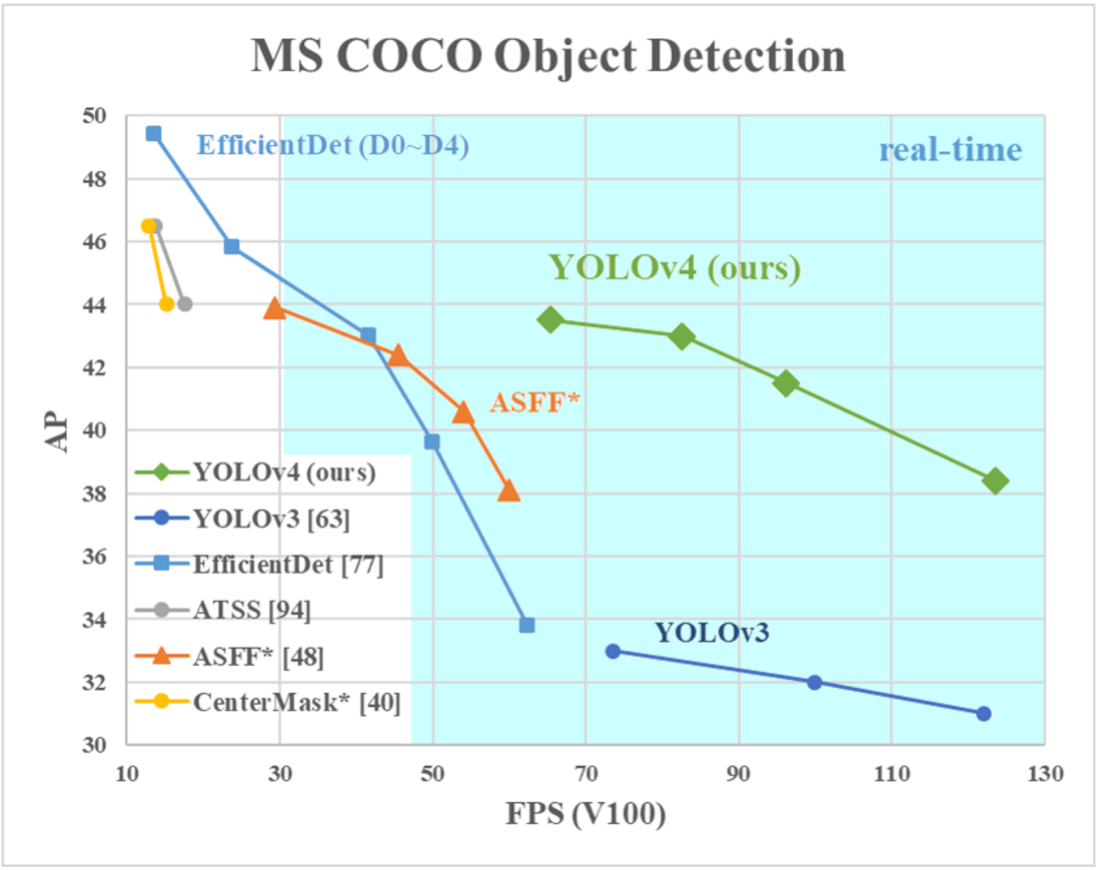


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

这研究的主要目的是设计一个可以在生产环境快速运行的目标检测器，并且进行并行计算优化，而不是较低的计算量理论指标（BFLOP）。我们希望所设计的目标易于训练和使用。例如，任何使用传统 GPU 进行训练和测试的人都可以实现实时、高质量、有说服力的目标检测结果，YOLOv4 的结果如图 1 所示。现将我们的成果总结如下：

1. 我们构建了一个快速、强大的模型，这使得大家都可以使用 1080 Ti 或 2080 Ti GPU 来训练一个超快、准确的目标检测器。

2. 我们验证了最先进的 Bag-of-Freebies 和 Bag-of-Specials 方法在目标检测训练期间的影响。

3. 我们修改了最先进的方法，使其变得更高效并且适合单 GPU 训练，包括 CBN[89]、PAN[49]、SAM[85]等。

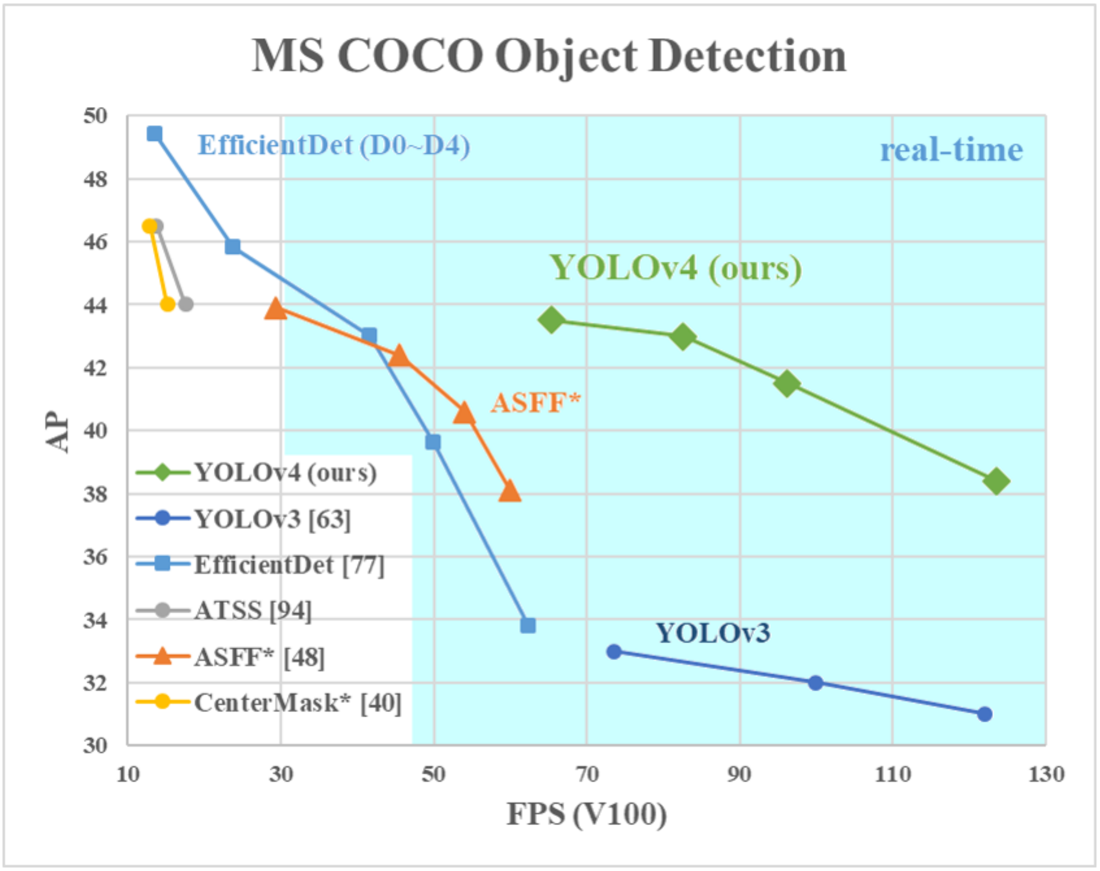


图 1：本文提出的 YOLOv4 和其他先进的目标检测器比较结果。YOLOv4 与 EfficientDet 相比精度差不多相同，但速度比其快两倍。YOLOv3 的 AP 值和 FPS 都分别提升了 10%和 12%。

2. Related work

2.1. Object detection models

A modern detector is usually composed of two parts, a backbone which is pre-trained on ImageNet and a head which is used to predict classes and bounding boxes of objects. For those detectors running on GPU platform, their backbone could be VGG [68], ResNet [26], ResNeXt [86], or DenseNet [30]. For those detectors running on CPU platform, their backbone could be SqueezeNet [31], MobileNet [28, 66, 27, 74], or ShuffleNet [97, 53]. As to the head part, it is usually categorized into two kinds, i.e., one-stage object detector and two-stage object detector. The most representative two-stage object detector is the R-CNN [19] series, including fast R-CNN [18], faster R-CNN [64], R-FCN [9], and Libra R-CNN [58]. It is also possible to make a two-stage object detector an anchor-free object detector, such as RepPoints [87]. As for one-stage object detector, the most representative models are YOLO [61, 62, 63], SSD [50], and RetinaNet [45]. In recent years, anchor-free one-stage object detectors are developed. The detectors of this sort are CenterNet [13], CornerNet [37, 38], FCOS [78], etc. Object detectors developed in recent years often insert some layers between backbone and head, and these layers are usually used to collect feature maps from different stages. We can call it the neck of an object detector. Usually, a neck is composed of several bottom-up paths and several top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN) [44], Path Aggregation Network (PAN) [49], BiFPN [77], and NAS-FPN [17]. In addition to the above models, some researchers put their emphasis on directly building a new backbone (DetNet [43], DetNAS [7]) or a new whole model (SpineNet [12], HitDetector [20]) for object detection.

2. 相关工作

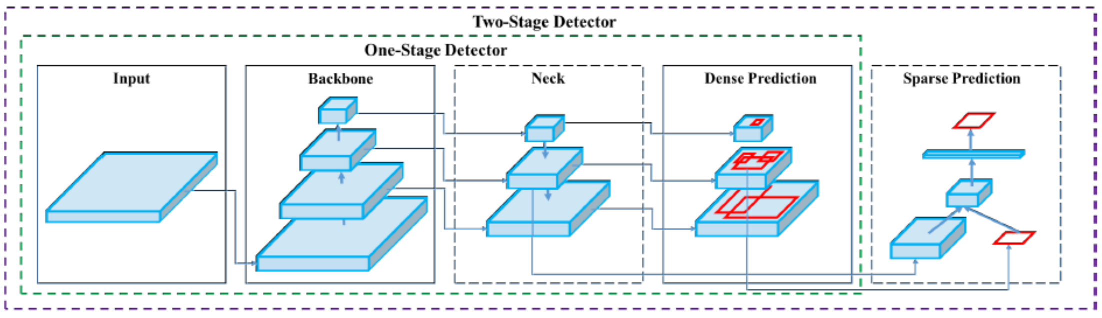
2.1. 目标检测模型

现代目标检测器通常由两部分组成：ImageNet 上预训练的 backbone 和用于预测类别和 BBOX 的检测器 head。对于那些在 GPU 平台上运行的探测器，其 backbone 可以是 VGG[68]，ResNet[26]、 ResNeXt[86]、或 DenseNet [30]。对于那些运行在 CPU 平台上的检测器形式，它们的 backbone 可以是 SqueezeNet[31]、MobileNet[28，66，27，74]，或 ShuffleNet[97，53]。至于 head 部分，它通常被分两类：即一阶段（one-stage）和两阶段（two-stage）的目标检测器。最有代表性的两阶段检测器是 R-CNN[19]系列模型，包括 Fast R-CNN[18]、Faster R-CNN[64]、R-FCN[9]和 Libra R-CNN[58]。也可以在两阶段目标检测器中不用 anchor 的目标检测器，如 RepPoints[87]。对于一阶段检测器来说，最代表性的有 YOLO[61、62、63]、SSD[50]和 RetinaNet[45]。近几年来，也开发了许多不使用 anchor 的一阶段目标检测器。这类检测器有 CenterNet[13]、CornerNet[37，38]、FCOS[78]等。近年来开发检测器往往会在 backbone 和 head 之间插入一些层，这些层用于收集不同阶段的特征图。我们可以称它为检测器的 neck。通常情况下 neck 是由几个自下而上或自上而下的通路（paths）组成。具有这种结构的网络包括 Feature Pyramid Network (FPN)[44]、Path Aggregation（PAN）[49]、BiFPN[77]和 NAS-FPN[17]。除上述模型外，有的研究者注重于直接重新构建 backbone（DetNet[43]、DetNAS[7]）或重新构建整个模型（SpineNet[12]、HitDetector[20])，并用于目标检测任务。

To sum up, an ordinary object detector is composed of several parts:

- 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 RCNN [23] (anchor based)
    - RepPoints [87] (anchor free)



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], … }

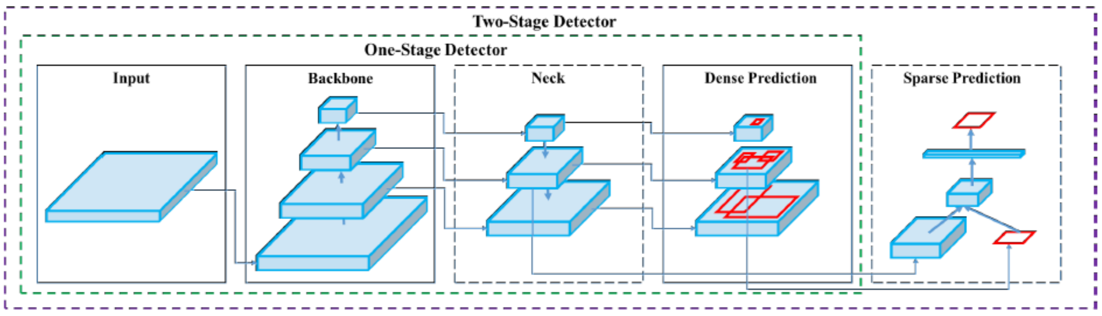
Figure 2: Object detector.

总结起来，通常目标检测模型由以下一些部分组成：

- 输入：图像、图像块、图像金字塔
- 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)



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], … }

图 2：目标检测器。

2.2. Bag of freebies

Usually, a conventional object detector is trained offline. Therefore, researchers always like to take this advantage and develop better training methods which can make the object detector receive better accuracy without increasing the inference cost. We call these methods that only change the training strategy or only increase the training cost as “bag of freebies.” What is often adopted by object detection methods and meets the definition of bag of freebies is data augmentation. The purpose of data augmentation is to increase the variability of the input images, so that the designed object detection model has higher robustness to the images obtained from different environments. For examples, photometric distortions and geometric distortions are two commonly used data augmentation method and they definitely benefit the object detection task. In dealing with photometric distortion, we adjust the brightness, contrast, hue, saturation, and noise of an image. For geometric distortion, we add random scaling, cropping, flipping, and rotating.

2.2. Bag of freebies

通常情况下，传统的目标检测器的训练都是在线下进行的。因此，研究者们总是喜欢利用纯下训练的好处而研究更好的训练方法，使得目标检测器在不增加测试成本的情况下达到更好的精度。我们将这些只需改变训练策略或只增加训练成本的方法称为 bag of freebies。目标检测经常采用并符合这个定义的就是数据增强。数据增强的目的是增加输入图像的多样性，从而使设计的目标检测模型对来自不同环境

的图片具有较高的鲁棒性。比如 photometric distortions 和 geometric distortions 是两种常用的数据增强方法，它们对检测任务肯定是有好处的。使用 photometric distortions 时，我们调整图像的亮度、对比度、色调、饱和度和噪声。使用 geometric distortions 时，我们对图像添加随机缩放、裁剪、翻转和旋转。

The data augmentation methods mentioned above are all pixel-wise adjustments, and all original pixel information in the adjusted area is retained. In addition, some researchers engaged in data augmentation put their emphasis on simulating object occlusion issues. They have achieved good results in image classification and object detection. For example, random erase [100] and CutOut [11] can randomly select the rectangle region in an image and fill in a random or complementary value of zero. As for hide-and-seek [69] and grid mask [6], they randomly or evenly select multiple rectangle regions in an image and replace them to all zeros. If similar concepts are applied to feature maps, there are DropOut [71], DropConnect [80], and DropBlock [16] methods. In addition, some researchers have proposed the methods of using multiple images together to perform data augmentation. For example, MixUp [92] uses two images to multiply and superimpose with different coefficient ratios, and then adjusts the label with these superimposed ratios. As for CutMix [91], it is to cover the cropped image to rectangle region of other images, and adjusts the label according to the size of the mix area. In addition to the above mentioned methods, style transfer GAN [15] is also used for data augmentation, and such usage can effectively reduce the texture bias learned by CNN.

上面提到的数据增强方法都是逐像素的调整，以及调整区域的所有原始像素信息会被保留下来。此外，一些从事数据增强工作的研究者把重点放在了模拟目标遮挡问题上。他们在图像分类和目标检测取得了好的结果。例如，随机擦除[100]和 CutOut[11]可以随机的选取图像中的矩形区域，并填充随机值或零的互补值。至于 hide-and-seek [69]和 grid mask [6]，他们随机或均匀地选择图像中的多个矩形区域，并将其全部像素值替换为零值。如果将类似的概念应用到特征图中，就是 DropOut[71]、DropConnect[80]和 DropBlock[16]方法。此外，有研究者提出了将多张图像放在一起从而实现数据增强的方法。例如，MixUp[92]将两张图像以不同系数的进行相乘和叠加，并根据叠加比例调整标签。对于 CutMix[91]，它通过覆盖裁剪后的图像到其他图像的矩形区域，并根据混合区的大小调整标签。除了以上提到的方法，网络迁移 GAN[15]也常常用于数据增强，这种方法可以有效地减少 CNN 学习到的纹理偏差。

Different from the various approaches proposed above, some other bag of freebies methods are dedicated to solving the problem that the semantic distribution in the dataset may have bias. In dealing with the problem of semantic distribution bias, a very important issue is that there is a problem of data imbalance between different classes, and this problem is often solved by hard negative example mining [72] or online hard example mining [67] in two-stage object detector. But the example mining method is not applicable to one-stage object detector, because this kind of detector belongs to the dense prediction architecture. Therefore Lin et al. [45] proposed focal loss to deal with the problem of data imbalance existing between various classes. Another very important issue is that it is difficult to express the relationship of the degree of association between different categories with the one-hot hard representation. This representation scheme is often used when executing labeling. The label smoothing proposed in [73] is to convert hard label into soft label for training, which can make model more robust. In order to obtain a better soft label, Islam et al. [33] introduced the concept of knowledge distillation to design the label refinement network.

与上面提出的各种方法不同，其他的一些 Bag of freebies 方法是专门解决可能有偏差的数据集中语义分布问题。在处理语义分布偏差的问题上，有一个很重要的问题是不同类别之间的数据不平衡，而两阶段检测器处理这个问题通常是通过 hard negative example mining [72] 或 online hard example mining [67]。但 example mining method 不适用于一阶段的目标检测器，因为这种检测器属于密集预测架构。因此，Linet al.[45]提出了 focal loss 解决数据不平衡问题。另一个很重要的问题是，one-hot 编码很难表达出类与类之间关联程度。这种表示方法（one-hot）通常在打标签的时候使用。在[73]中提出的 label smoothing 方案是将硬标签转化为软标签进行训练，可以使模型更具有鲁棒性。为了获得更好的软标签，Islam 等[33]引入**知识蒸馏**的概念并用于设计标签细化网络。

The last bag of freebies is the objective function of Bounding Box (BBox) regression. The traditional object detector usually uses Mean Square Error (MSE) to directly perform regression on the center point coordinates and height and width of the BBox, i.e., {x_{center}, y_{center}, w, h} , or the upper left point and the lower right point, i.e., {x_{top_left}, y_{top_left}, x_{bottom_right}, y_{bottom_right}}. As for anchor-based method, it is to estimate the corresponding offset, for example { x center offset , y center offset , w offset , h offset } and { x top left offset , y top left offset , x bottom right offset , y bottom right offset } . However, to directly estimate the coordinate values of each point of the BBox is to treat these points as independent variables, but in fact does not consider the integrity of the object itself. In order to make this issue processed better, some researchers recently proposed IoU loss [90], which puts the coverage of predicted BBox area and ground truth BBox area into consideration. The IoU loss computing process will trigger the calculation of the four coordinate points of the BBox by executing IoU with the ground truth, and then connecting the generated results into a whole code. Because IoU is a scale invariant representation, it can solve the problem that when traditional methods calculate the l1 or l2 loss of { x, y, w, h } , the loss will increase with the scale. Recently, some researchers have continued to improve IoU loss. For example, GIoU loss [65] is to include the shape and orientation of object in addition to the coverage area. They proposed to find the smallest area BBox that can simultaneously cover the predicted BBox and ground truth BBox, and use this BBox as the denominator to replace the denominator originally used in IoU loss. As for DIoU loss [99], it additionally considers the distance of the center of an object, and CIoU loss [99], on the other hand simultaneously considers the overlapping area, the distance between center points, and the aspect ratio. CIoU can achieve better convergence speed and accuracy on the BBox regression problem.

最后一个 bag of freebies 是边界框(BBox)回归的目标函数。检测器通常使用 MSE 损失函数对 BBOX 的中心点和宽高进行回归，例如{x_{center}, y_{center}, w, h}，或者是回归预测左上角的点和右下角的点，例如{x_{top_left}，y_{top_left}，x_{bottom_right}，y_{bottom_right}}。对于基于 anchor 的方法，它会估算出对应的偏移量，例如{ x center offset , y center offset , w offset , h offset } and { x top left offset , y top left offset , x bottom right offset , y bottom right offset }。但是，如果要直接估计 BBOX 每个点的坐标值，就要将这些点作为独立变量，但实际上未考虑对象本身的完整性。为了使这一问题得到更好的解决，一些研究人员最近提出了 IoU 损失[90]，其考虑了预测 BBox 面积和 ground truth BBox 面积的覆盖度。IoU 损失计算过程将通过计算预测值与真实值的 IoU，然后将生成的结果连接成一个整体代码，最终通过计算获得 BBox 的四个坐标值。

因为 IOU 是一个与尺度无关的表示，它可以解决当传统方法计算{x，y，w，h}的 l1 或 l2 损失时，损失会随着尺度增加而增大的问题。最近，一些研究人员不断改善 IOU 损失。例如 GIoU 损失[65]除覆盖面积也考虑物体的形状和方向。他们建议找到能同时覆盖预测 BBOX 和真实值 BBox 的最小面积 BBOX，并使用这个 BBox 作为分母并取代原先 IoU 损失的分母。至于 DIoU 损失[99]，它另外还包括考虑物体中心的距离，另一方面 CIoU 损失[99]同时考虑到重叠面积和中心点之间的距离以及长宽比。CIoU 可以在 BBox 回归问题上获得了更好的收敛速度和精度。

2.3. Bag of specials

For those plugin modules and post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection, we call them “bag of specials”. Generally speaking, these plugin modules are for enhancing certain attributes in a model, such as enlarging receptive field, introducing attention mechanism, or strengthening feature integration capability, etc., and post-processing is a method for screening model prediction results.

2.3. Bag of specials

对于那些只会增加少量的推理成本的插入模块和后期处理方法，但可显著提高目标检测的准确性，我们称其为“Bag of specials”。一般来说,这些插入模块是用来增强模型的某些属性的，如扩大感受野、

引入注意力机制或增强特征整合能力等，而后处理是一种筛选模型预测结果方法。

Common modules that can be used to enhance receptive field are SPP [25], ASPP [5], and RFB [47]. The SPP module was originated from Spatial Pyramid Matching (SPM) [39], and SPMs original method was to split feature map into several d × d equal blocks, where d can be { 1, 2, 3, … } , thus forming spatial pyramid, and then extracting bag-of-word features. SPP integrates SPM into CNN and use max-pooling operation instead of bag-of-word operation. Since the SPP module proposed by He et al. [25] will output one dimensional feature vector, it is infeasible to be applied in Fully Convolutional Network (FCN). Thus in the design of YOLOv3 [63], Redmon and Farhadi improve SPP module to the concatenation of max-pooling outputs with kernel size k × k, where k = { 1, 5, 9, 13 } , and stride equals to 1. Under this design, a relatively large k × k maxpooling effectively increase the receptive field of backbone feature. After adding the improved version of SPP module, YOLOv3-608 upgrades AP 50 by 2.7% on the MS COCO object detection task at the cost of 0.5% extra computation. The difference in operation between ASPP [5] module and improved SPP module is mainly from the original k×k kernel size, max-pooling of stride equals to 1 to several 3 × 3 kernel size, dilated ratio equals to k, and stride equals to 1 in dilated convolution operation. RFB module is to use several dilated convolutions of k×k kernel, dilated ratio equals to k, and stride equals to 1 to obtain a more comprehensive spatial coverage than ASPP. RFB [47] only costs 7% extra inference time to increase the AP 50 of SSD on MS COCO by 5.7%.

可用于扩大感受野的常用模块有 SPP[25]、ASPP[5]和 RFB[47]。SPP 模块源于 Spatial Pyramid Match（SPM）[39]，而 SPMs 的原始方法是将特征图分割成几个 d×d 相等大小的块，其中 d 可以是{1,2,3,…}，从而形成空间金字塔，然后提取 bag-of-word 特征。SPP 将 SPM 集成到 CNN 并使用 max-pooling 操作而不是 bag-of-word 运算。由于 He 等人提出的 SPP 模块[25]会输出一维特征向量，因此不可能应用于全卷积网络（FCN）中。因此，在 YOLOv3 的设计[63]中，Redmon 和 Farhadi 改进了 YOLOv3 的设计，将 SPP 模块修改为融合 k×k 池化核的最大池化输出，其中 k = {1,5,9,13}，步长等于 1。在这种设计下，一个相对较大的 k×k 有效地增加了 backbone 的感受野。增加了改进版的 SPP 模块后，YOLOv3-608 在 MS COCO 上 AP_{50} 提升了 2.7%，但要付出 0.5%的额外计算成本。ASPP[5]模块和改进后的 SPP 模块在操作上的区别是主要由原来的步长 1、核大小为 k×k 的最大池化到几个 3×3 核的最大池化，缩放比例为 k，步长 1 的空洞卷积。RFB 模块是使用几个 k×k 核的空洞卷积，空洞率为 k，步长为 1 以得到比 ASPP 更全面的空间覆盖率。RFB[47]只需额外增加 7%推理时间却在 MS COCO 上将 SSD 的 AP_{50} 提升 5.7%。

The attention module that is often used in object detection is mainly divided into channel-wise attention and point-wise attention, and the representatives of these two attention models are Squeeze-and-Excitation (SE) [29] and Spatial Attention Module (SAM) [85], respectively. Although SE module can improve the power of ResNet50 in the ImageNet image classification task 1% top-1 accuracy at the cost of only increasing the computational effort by 2%, but on a GPU usually it will increase the inference time by about 10%, so it is more appropriate to be used in mobile devices. But for SAM, it only needs to pay 0.1% extra calculation and it can improve ResNet50-SE 0.5% top-1 accuracy on the ImageNet image classification task. Best of all, it does not affect the speed of inference on the GPU at all.

在目标检测中经常使用的注意力模块，通常分为 channel-wise 注意力和 point-wise 注意力，具有代表性的两个模型分别是 Squeeze-and-Excitation (SE) [29]和 Spatial Attention Module (SAM) [85]。虽然 SE 模块可以将 ResNet50 在 ImageNet 图像分类任务上的 top-1 准确率提升 1%，而计算量仅仅增加 2%，但是在 GPU 上推理时间通常会增加 10%左右，所以更适合用于移动端设备。但对于 SAM，它只需要额外 0.1%的计算量就可以将 ResNet50-SE 在 ImageNet 图像分类任务上的 Top-1 精度提高 0.5%。最重要的是，它完全不会影响 GPU 上推理的速度。

In terms of feature integration, the early practice is to use skip connection [51] or hyper-column [22] to integrate low-level physical feature to high-level semantic feature. Since multi-scale prediction methods such as FPN have become popular, many lightweight modules that integrate different feature pyramid have been proposed. The modules of this sort include SFAM [98], ASFF [48], and BiFPN [77]. The main idea of SFAM is to use SE module to execute channel-wise level re-weighting on multi-scale concatenated feature maps. As for ASFF, it uses softmax as point-wise level re-weighting and then adds feature maps of different scales. In BiFPN, the multi-input weighted residual connections is proposed to execute scale-wise level re-weighting, and then add feature maps of different scales.

在特征融合方面，早期的做法是使用快捷连接（skip connection）[51]或超列（hyper-column）[22]将低级物理特征融合成高级语义特征。由于 FPN 等多尺度预测方法越来越流行，许多集成了不同的特征金字塔特征的轻量级模块被提了出来。这类模块包括 SFAM[98]、ASFF[48]和 BiFPN[77]。SFAM 的主要思想是利用 SE 模块对多尺度特征图在通道方向上重新加权拼接特征图。至于 ASFF，它用 softmax 进行 point-wise 水平的重新加权，然后在不同尺度添加特征图。在 BiFPN 中，多输入加权残差连接进行多尺度水平的重新加权，然后在不同尺度上添加特征图。

In the research of deep learning, some people put their focus on searching for good activation function. A good activation function can make the gradient more efficiently propagated, and at the same time it will not cause too much extra computational cost. In 2010, Nair and Hinton [56] propose ReLU to substantially solve the gradient vanish problem which is frequently encountered in traditional tanh and sigmoid activation function. Subsequently, LReLU [54], PReLU [24], ReLU6 [28], Scaled Exponential Linear Unit (SELU) [35], Swish [59], hard-Swish [27], and Mish [55], etc., which are also used to solve the gradient vanish problem, have been proposed. The main purpose of LReLU and PReLU is to solve the problem that the gradient of ReLU is zero when the output is less than zero. As for ReLU6 and hard-Swish, they are specially designed for quantization networks. For self-normalizing a neural network, the SELU activation function is proposed to satisfy the goal. One thing to be noted is that both Swish and Mish are continuously differentiable activation function.

在深度学习的研究中，有人注重于寻找好的激活函数。一个好的激活函数可以使梯度更有效地传播，同时也不会造成太多的额外计算成本。2010 年，Nair 和 Hinton [56]提出了 ReLU 来实质性地解决梯度消失的问题，这也是 tanh 和 sigmoid 激活函数经常遇到的问题。随后便提出了 LReLU[54]、PReLU[24]、ReLU6[28]、Scaled Exponential Linear Unit (SELU)[35]、Swish[59]、hard-Swish[27]和 Mish[55]等，这些激活函数也用来解决梯度消失问题的。LReLU 和 PReLU 的主要目的是为了解决当 ReLU 输出小于零时梯度为零的问题。至于 ReLU6 和 Hard-Swish，它们是专为量化网络（quantization networks）设计。对于自归一化的神经网络，SELU 激活函数的提出满足了这一目的。需要注意的是，Swish 和 Mish 都是连续可微的激活函数。

The post-processing method commonly used in deep-learning-based object detection is NMS, which can be used to filter those BBoxes that badly predict the same object, and only retain the candidate BBoxes with higher response. The way NMS tries to improve is consistent with the method of optimizing an objective function. The original method proposed by NMS does not consider the context information, so Girshick et al. [19] added classification confidence score in R-CNN as a reference, and according to the order of confidence score, greedy NMS was performed in the order of high score to low score. As for soft NMS [1], it considers the problem that the occlusion of an object may cause the degradation of confidence score in greedy NMS with IoU score. The DIoU NMS [99] developers way of thinking is to add the information of the center point distance to the BBox screening process on the basis of soft NMS. It is worth mentioning that, since none of above post-processing methods directly refer to the captured image features, post-processing is no longer required in the subsequent development of an anchor-free method.

基于深度学习的目标检测中常用的后处理方法是 NMS（非极大值抑制），它可以用于过滤那些对相同目标预测较差的边界框，并且只保留响应较高的候选边界框。NMS 努力改进的方式与目标函数的优化方法一致。NMS 原始方法没有考虑背景信息，所以 Girshick 等人[19]在 R-CNN 中增加了分类置信度分数作为参考，并根据信任分数的顺序，从高分到低分的顺序执行贪婪 NMS。至于 soft NMS[1]，

它关注了这样一个问题，即目标遮挡可能会导致基于 IoU 分数的贪婪 NMS 的置信度得分降低。基于 DIoU 的 NMS[99]的开发者思路是在 soft NMS 基础上将中心点距离信息增加到 BBox 筛选的过程中。值得注意的是，由于上述后处理方法都没有直接涉及指提取特征图，所以在不使用 anchor 的方法后续开发中不再需要进行后处理。

3. Methodology

The basic aim is fast operating speed of neural network, in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). We present two options of real-time neural networks:

- For GPU we use a small number of groups (1 – 8) in convolutional layers: CSPResNeXt50 / CSPDarknet53
- For VPU – we use grouped-convolution, but we refrain from using Squeeze-and-excitement (SE) blocks – specifically this includes the following models: EfficientNet-lite / MixNet [76] / GhostNet [21] / MobileNetV3

3. 方法

基本目的是生产系统中神经网络的快速运行速度和并行计算的优化，而不是低计算量理论指标（BFLOP）。我们提出了两种实时神经网络：

- 对于 GPU，我们在卷积层使用少量组（1-8）：CSPResNeXt50 / CSPDarknet53

●对于 VPU，我们使用分组卷积，但避免使用 Squeeze-and-excitement (SE) blocks。具体包括以下模型：EfficientNet-lite / MixNet [76] / GhostNet[21] / MobileNetV3

3.1 Selection of architecture

Our objective is to find the optimal balance among the input network resolution, the convolutional layer number, the parameter number (filter size^2 * filters * channel / groups), and the number of layer outputs (filters). For instance, our numerous studies demonstrate that the CSPResNext50 is considerably better compared to CSPDarknet53 in terms of object classification on the ILSVRC2012 (ImageNet) dataset [10]. However, conversely, the CSPDarknet53 is better compared to CSPResNext50 in terms of detecting objects on the MS COCO dataset [46].

3.1 架构选择

我们的目的是在输入网络分辨率、卷积层数目、参数数量（卷积核^2* 卷积核个数 * 通道数/组数）和每层输出个数（过滤器）之间找到最佳平衡。例如我们的许多研究表明 CSPResNext50 在 ILSVRC2012(ImageNet)数据集上的目标分类效果比 CSPDarknet53 好很多。然而，CSPDarknet53 在 MS COCO 数据集上的目标检测效果比 CSPResNext50 更好。

The next objective is to select additional blocks for increasing the receptive field and the best method of parameter aggregation from different backbone levels for different detector levels: e.g. FPN, PAN, ASFF, BiFPN.

下一个目标是选择额外的 blocks 以扩大感受野，从不同级别的 backbone 中选择最佳的参数组合方法以达到不同水平的检测效果，例如 FPN、PAN、ASFF、BiFPN。

A reference model which is optimal for classification is not always optimal for a detector. In contrast to the classifier, the detector requires the following:

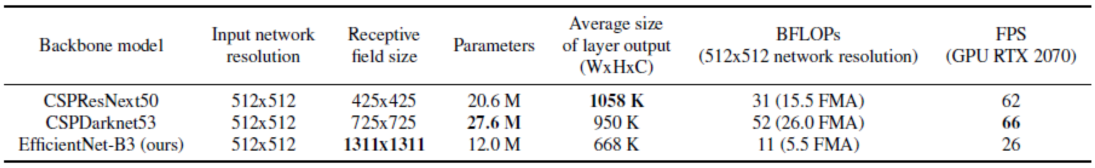
- Higher input network size (resolution) – for detecting multiple small-sized objects
- More layers – for a higher receptive field to cover the increased size of input network
- More parameters – for greater capacity of a model to detect multiple objects of different sizes in a single image

一个最佳的分类参考模型并不总是最佳的检测器。与分类器相比，检测器需要满足以下几点：

- 更大的输入网络尺寸（分辨率）——用于检测多个小尺寸目标
- 更多的层数——获得更大的感受野以便能适应网络输入尺寸的增加
- 更多参数——获得更大的模型容量以便在单个图像中检测多个大小不同的物体。

Hypothetically speaking, we can assume that a model with a larger receptive field size (with a larger number of convolutional layers 3 ×3) and a larger number of parameters should be selected as the backbone. Table 1 shows the information of CSPResNeXt50, CSPDarknet53, and EfficientNet B3. The CSPResNext50 contains only 16 convolutional layers 3 × 3, a 425 × 425 receptive field and 20.6 M parameters, while CSPDarknet53 contains 29 convolutional layers 3 × 3, a 725 × 725 receptive field and 27.6 M parameters. This theoretical justification, together with our numerous experiments, show that CSPDarknet53 neural network is the optimal model of the two as the backbone for a detector.

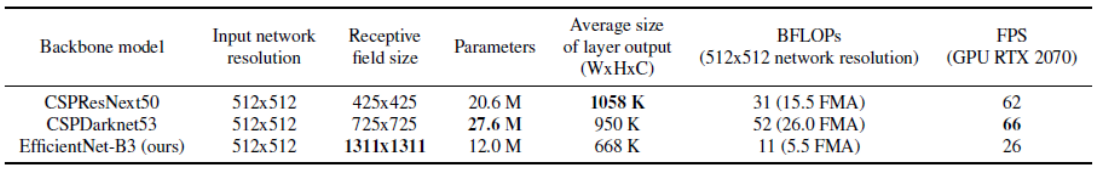
Table 1: Parameters of neural networks for image classification.



Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

我们可以假设选择的 backbone 模型具有较大的感受野（具有很多 3×3 卷积层）和大量的参数。表 1 显示了 CSPResNeXt50，CSPDarknet53 和 EfficientNet B3 的相关信息。CSPResNext50 仅包含 16 个 3×3 卷积层、425×425 感受野大小和 20.6M 个参数，而 CSPDarknet53 包含 29 个 3×3 卷积、725×725 感受野大小和 27.6M 个参数。这种理论上的证明，加上我们大量的实验表明：CSPDarknet53 是这两个神经网络中最佳的检测器 backbone 模型。

表 1：图像分类神经网络的参数。



Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

The influence of the receptive field with different sizes is summarized as follows:

- Up to the object size – allows viewing the entire object
- Up to network size – allows viewing the context around the object
- Exceeding the network size – increases the number of connections between the image point and the final activation

不同大小的感受野的影响总结如下：

- 最大目标尺寸——允许观察到整个目标
- 最大网络尺寸——允许观察到目标周围的上下文
- 超出网络尺寸——增加图像像素点与最终激活值之间的连接数

We add the SPP block over the CSPDarknet53, since it significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed. We use PANet as the method of parameter aggregation from different backbone levels for different detector levels, instead of the FPN used in YOLOv3.

我们将 SPP 模块添加到 CSPDarknet53 上，因为它大大增加了感受野，分离出最重要的 context 特征，然而几乎不会导致网络运行速

度降低。我们使用 PANet 作为来自不同检测器水平不同 backbone 的参数组合方法而不是 YOLOv3 中使用的 FPN。

Finally, we choose CSPDarknet53 backbone, SPP additional module, PANet path-aggregation neck, and YOLOv3 (anchor based) head as the architecture of YOLOv4.

最后，对于 YOLOv4 架构，我们选择 CSPDarknet53 为 backbone、SPP 额外添加模块、PANet path-aggregation 为 neck、YOLOv3（基于 anchor 的）为 head。

In the future we plan to expand significantly the content of Bag of Freebies (BoF) for the detector, which theoretically can address some problems and increase the detector accuracy, and sequentially check the influence of each feature in an experimental fashion.

之后，我们将计划大幅扩展 Bag of Freebies（BoF）的内容到检测器架构中，这些扩展的模块理论上应该可以解决一些问题并增加检测器准确性，并通过实验的方式按顺序检查每个功能的影响。

We do not use Cross-GPU Batch Normalization (CGBN or SyncBN) or expensive specialized devices. This allows anyone to reproduce our state-of-the-art outcomes on a conventional graphic processor e.g. GTX 1080Ti or RTX 2080Ti.

我们没有使用跨 GPU 的批标准化（CGBN 或 SyncBN）或昂贵的专用设备。这使得任何人都可用常规图形处理器，例如 GTX 1080Ti 或 RTX2080Ti，复现我们最新的成果。

3.2. Selection of BoF and BoS

For improving the object detection training, a CNN usually uses the following:

- Activations: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish
- Bounding box regression loss: MSE, IoU, GIoU, CIoU, DIoU
- Data augmentation: CutOut, MixUp, CutMix
- Regularization method: DropOut, DropPath [36], Spatial DropOut [79], or DropBlock
- Normalization of the network activations by their mean and variance: Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
- Skip-connections: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

3.2. BoF 和 BoS 的选择

为了改善目标检测的训练，CNN 通常会使用如下方法或结构：

- 激活函数: ReLU、leaky-ReLU、parametric-ReLU、ReLU6、SELU、Swish、Mish
- 边界框损失回归：MSE、IoU、GIoU、CIoU、DIoU

- 数据增强：CutOut、MixUp、CutMix
- 正则化方法：DropOut、DropPath[36]、Spatial DropOut [79]、DropBlock
- 通过均值和方差标准化网络激活函数输出值：Batch Normalization (BN) [32]、Cross-GPU Batch Normalization (CGBN or SyncBN)[93]、Filter Response Normalization (FRN) [70]、Cross-Iteration Batch Normalization (CBN) [89]
- 快捷连接（Skip-connections）：残差连接、加权残差连接、多输入加权残差连接、Cross stage partial connections (CSP)

As for training activation function, since PReLU and SELU are more difficult to train, and ReLU6 is specifically designed for quantization network, we therefore remove the above activation functions from the candidate list. In the method of reqularization, the people who published DropBlock have compared their method with other methods in detail, and their regularization method has won a lot. Therefore, we did not hesitate to choose DropBlock as our regularization method. As for the selection of normalization method, since we focus on a training strategy that uses only one GPU, syncBN is not considered.

至于训练激活函数，由于 PRELU 和 SELU 的训练难度较大，而 ReLU6 是专门为量化网络而设计的，因此我们从候选列表中删除这几个激活函数。至于正则化方法，发表了 DropBlock 的人将他们的方法与其他方法进行了细致的比较，他们的正则化方法完胜。因此，我

们毫不犹豫地选择了 DropBlock 作为我们正则化方法。至于归一化（或标准化）方法的选择，由于我们只关注在仅使用一个 GPU 上的训练策略，因此不会考虑使用 syncBN。

3.3. Additional improvements

In order to make the designed detector more suitable for training on single GPU, we made additional design and improvement as follows:

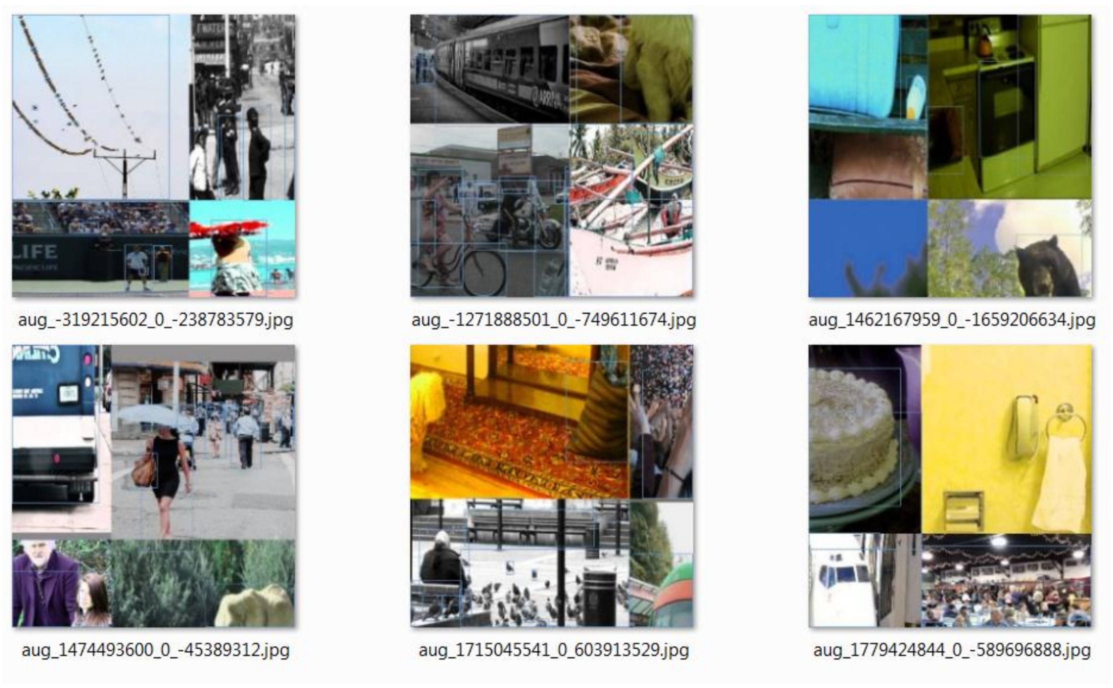
- We introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)
- We select optimal hyper-parameters while applying genetic algorithms
- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)

3.3 额外的改进

为了使所设计的检测器更适合于在单 GPU 上进行训练，我们做了如下额外的设计和改进：

- 我们引入了一种新的数据增强方法 Mosaic 和自对抗训练方法（Self-Adversarial Training，SAT）
- 我们使用遗传算法选择最优超参数
- 我们对现在方法做了一些修改，使得我们的设计更适合于高效的训练和检测——修改的 SAM、修改的 PAN 和 Cross mini-Batch Normalization (CmBN).

Mosaic represents a new data augmentation method that mixes 4 training images. Thus 4 different contexts are mixed, while CutMix mixes only 2 input images. This allows detection of objects outside their normal context. In addition, batch normalization calculates activation statistics from 4 different images on each layer. This significantly reduces the need for a large mini-batch size.



aug_-319215602_0_-238783579.jpg
aug_-1271888501_0_-749611674.jpg
aug_1462167959_0_-1659206634.jpg
aug_1474493600_0_-45389312.jpg
aug_1715045541_0_603913529.jpg
aug_1779424844_0_-589696888.jpg

Figure 3: Mosaic represents a new method of data augmentation.

Mosaic 是一个混合了 4 个训练图像的新的数据增强方法。由于混合了 4 个不同的 contexts，而 CutMix 只混合了 2 个输入图像。这使得可以检测到目标正常 contexts 之外的目标。此外，批标准化从每层上 4 个不同的图像计算激活值统计数据。这显著地减少了对大 batch size 的需要。

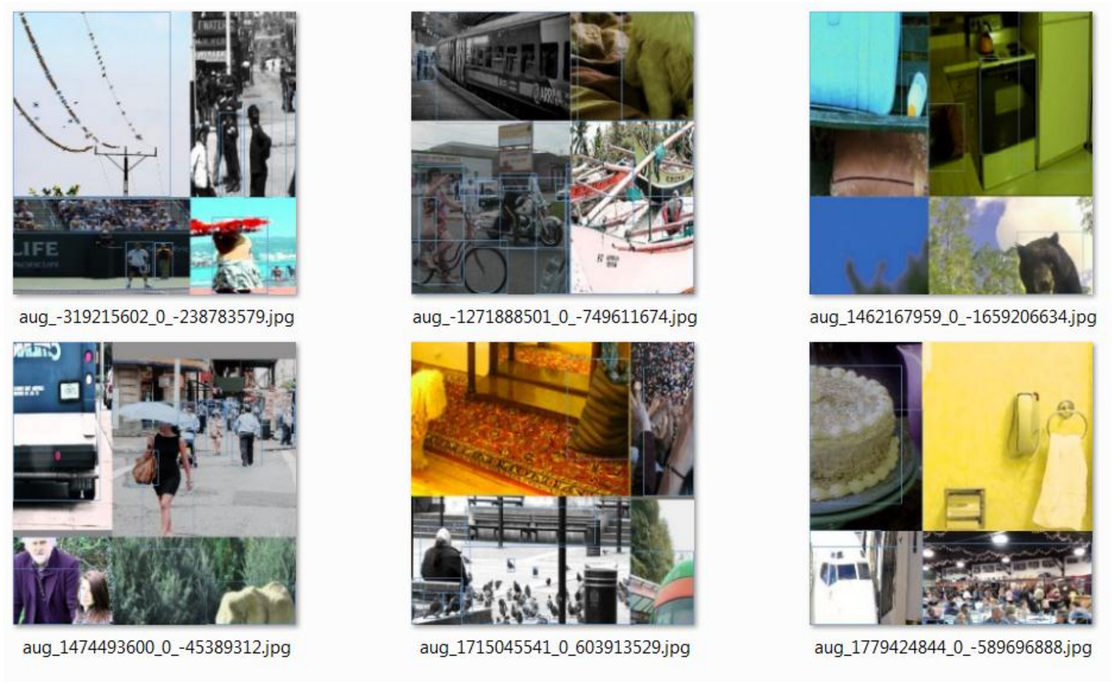
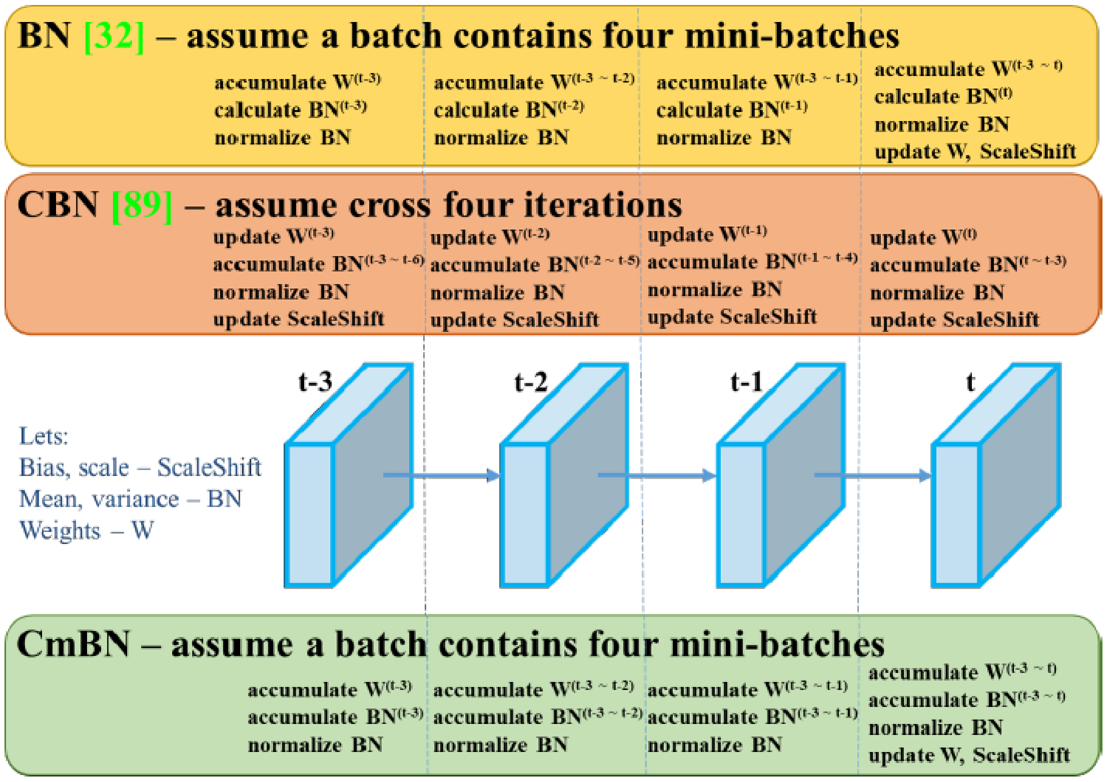

aug_-319215602_0_-238783579.jpg
aug_-1271888501_0_-749611674.jpg
aug_1462167959_0_-1659206634.jpg
aug_1474493600_0_-45389312.jpg
aug_1715045541_0_603913529.jpg
aug_1779424844_0_-589696888.jpg

图 3：数据增强的一个新方法马赛克（Mosaic）。

Self-Adversarial Training (SAT) also represents a new data augmentation technique that operates in 2 forward backward stages. In the 1st stage the neural network alters the original image instead of the network weights. In this way the neural network executes an adversarial attack on itself, altering the original image to create the deception that there is no desired object on the image. In the 2nd stage, the neural network is trained to detect an object on this modified image in the normal way.

自对抗训练（Self-Adversarial Training，SAT）也是一种新的数据增强技术，以 2 个前向反向阶段的方式进行操作。在第一个阶段，神经网络改变的是原始图像而不是的网络权重。这样神经网络对其自身进行对抗性攻击，改变原始图像并创造出图像上没有目标的假象。在第 2 个阶段中，通过正常方式在修改的图像上进行目标检测对神经网络进行训练。

CmBN represents a CBN modified version, as shown in Figure 4, defined as Cross mini-Batch Normalization (CmBN). This collects statistics only between mini-batches within a single batch.



BN [32] – assume a batch contains four mini-batches
accumulate $W^{(t-3)}$ calculate $BN^{(t-3)}$ normalize BN
accumulate $W^{(t-3 \sim t-2)}$ calculate $BN^{(t-2)}$ normalize BN
accumulate $W^{(t-3 \sim t-1)}$ calculate $BN^{(t-1)}$ normalize BN
accumulate $W^{(t-3 \sim t)}$ calculate $BN^{(t)}$ normalize BN update W, ScaleShift
CBN [89] – assume cross four iterations
update $W^{(t-3)}$ accumulate $BN^{(t-3 \sim t-6)}$ normalize BN update ScaleShift
update $W^{(t-2)}$ accumulate $BN^{(t-2 \sim t-5)}$ normalize BN update ScaleShift
update $W^{(t-1)}$ accumulate $BN^{(t-1 \sim t-4)}$ normalize BN update ScaleShift
update $W^{(t)}$ accumulate $BN^{(t \sim t-3)}$ normalize BN update ScaleShift
t-3
t-2
t-1
t
Lets:
Bias, scale – ScaleShift
Mean, variance – BN
Weights – W
CmBN – assume a batch contains four mini-batches
accumulate $W^{(t-3)}$ accumulate $BN^{(t-3)}$ normalize BN
accumulate $W^{(t-3 \sim t-2)}$ accumulate $BN^{(t-3 \sim t-2)}$ normalize BN
accumulate $W^{(t-3 \sim t-1)}$ accumulate $BN^{(t-3 \sim t-1)}$ normalize BN
accumulate $W^{(t-3 \sim t)}$ accumulate $BN^{(t-3 \sim t)}$ normalize BN update W, ScaleShift

Figure 4: Cross mini-Batch Normalization.

如图 4 所示，CmBN 是 CBN 的修改版，定义为 Cross mini-Batch Normalization(CmBN)。其只收集单个批次内 mini-batches 之间的统计数据。

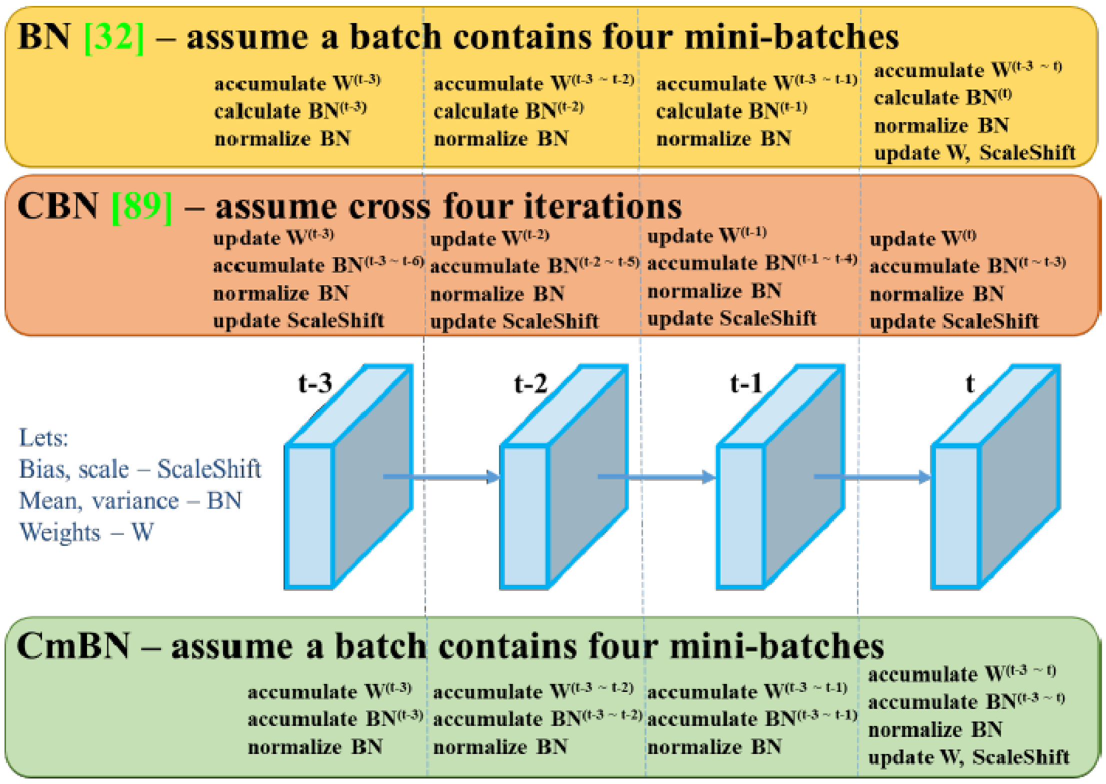


图 4：Cross mini-Batch Normalization。

We modify SAM from spatial-wise attention to pointwise attention, and replace shortcut connection of PAN to concatenation, as shown in Figure 5 and Figure 6, respectively.

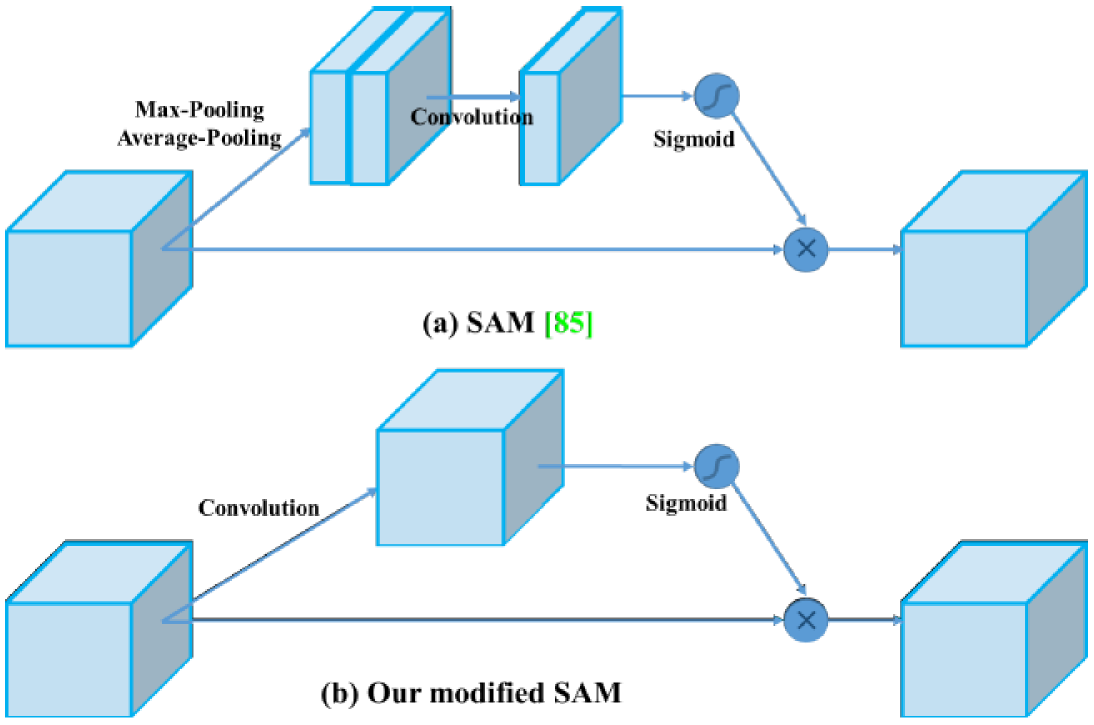
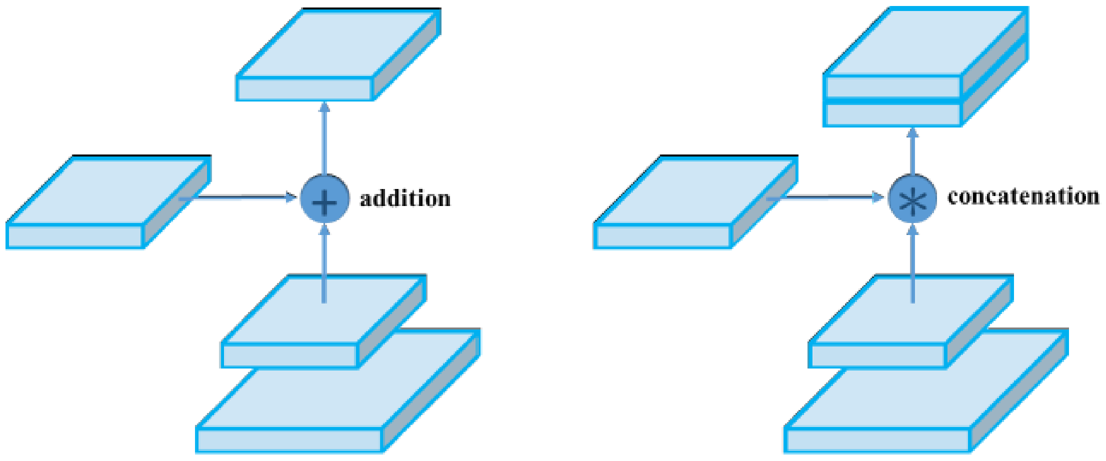


Figure 5: Modified SAM.



(a) PAN [49]

(a) Our modified PAN

Figure 6: Modified PAN.

如图 5 所示，我们将 SAM 从 spatial-wise attention 修改为 point-wise attention，如图 6 所示，我们将 PAN 的快捷连接改为拼接。

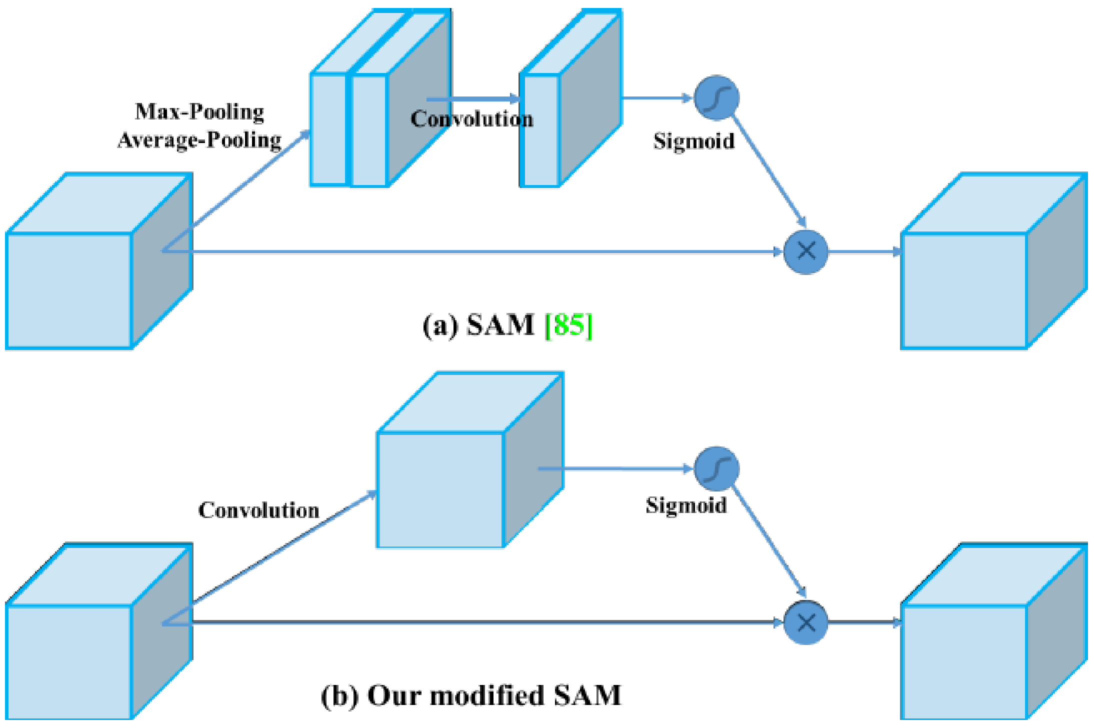


图 5：修改的 SAM。

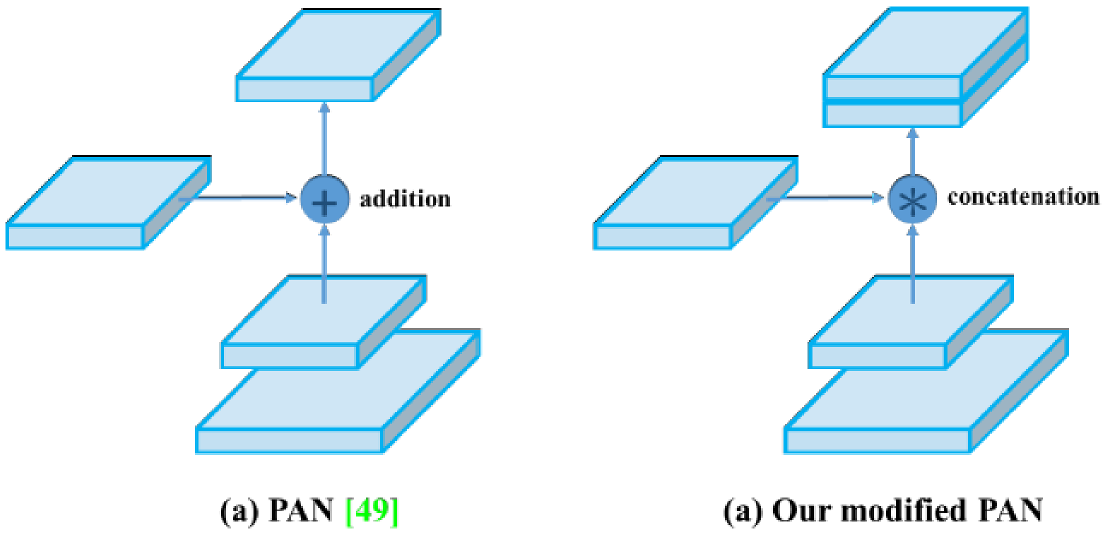


图 6：修改的 PAN。

3.4. YOLOv4

In this section, we shall elaborate the details of YOLOv4.

YOLOv4 consists of:

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

YOLO v4 uses:

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
- Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC)
- Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple

anchors for a single ground truth, Cosine annealing scheduler [52], Optimal hyperparameters, Random training shapes

- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

3.4. YOLOv4

本节，我们将详细介绍 YOLOv4。

YOLOv4 包括：

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

YOLO v4 使用：

- Bag of Freebies (BoF) for backbone：CutMix and Mosaic 数据增强、DropBlock 正则化、Class label smoothing
- Bag of Specials (BoS) for backbone：Mish 激活函数、Cross-stage partial connections (CSP)、多输入加权残差连接 (MiWRC)
- Bag of Freebies (BoF) for detector: CIoU 损失、CmBN、DropBlock 正则化、Mosaic 数据增强、自对抗训练、Eliminate grid sensitivity、Using multiple anchors for a single ground truth、Cosine annealing scheduler [52]、优化超参数、Random training shapes

- Bag of Specials (BoS) for detector：Mish 激活函数、SPP-block、SAM-block、PAN path-aggregation block、DIoU-NMS

4 Experiments

We test the influence of different training improvement techniques on accuracy of the classifier on ImageNet (ILSVRC 2012 val) dataset, and then on the accuracy of the detector on MS COCO (test-dev 2017) dataset.

4. 实验

我们测试了不同的训练改进在 ImageNet 数据集分类任务（ILSVRC 2012 年 val）和 MS COCO（test-dev 2017）数据集检测上的准确性。

4.1. Experimental setup

In ImageNet image classification experiments, the default hyper-parameters are as follows: the training steps is 8,000,000; the batch size and the mini-batch size are 128 and 32, respectively; the polynomial decay learning rate scheduling strategy is adopted with initial learning rate 0.1; the warm-up steps is 1000; the momentum and weight decay are respectively set as 0.9 and 0.005. All of our BoS experiments use the same hyper-parameter as the default setting, and in the BoF experiments, we add an additional 50% training steps. In the BoF experiments, we verify MixUp, CutMix, Mosaic, Bluring data augmentation, and label smoothing regularization methods. In the BoS experiments, we compared the effects of LReLU, Swish, and Mish activation function. All experiments are trained with a 1080 Ti or 2080 Ti GPU.

4.1 实验设置

在 ImageNet 的图像分类实验中，默认超参数如下：训练步数为 8 百万次；批大小和 mini 批大小分别为 128 和 32；polynomial decay learning rate scheduling strategy 初始学习率为 0.1 的多项式衰减调度策略；warm-up 步数为 1000；动量和衰减权重分别设定为 0.9 和 0.005。我们所有的 BoS 实验都使用的默认超参数设置，而在 BoF 实验中，我们增加了额外 50%的训练步数。在 BoF 实验中，我们验证了 MixUp、CutMix、Mosaic、Bluring 数据增强和标签 smoothing 正则化方法。在 BoS 实验中我们比较了 LReLU、Swish 和 Mish 激活函数的效果。所有实验都使用用 1080Ti 或 2080 Ti GPU 进行了训练。

In MS COCO object detection experiments, the default hyper-parameters are as follows: the training steps is 500,500; the step decay learning rate scheduling strategy is adopted with initial learning rate 0.01 and multiply with a factor 0.1 at the 400,000 steps and the 450,000 steps, respectively; The momentum and weight decay are respectively set as 0.9 and 0.0005. All architectures use a single GPU to execute multi-scale training in the batch size of 64 while mini-batch size is 8 or 4 depend on the architectures and GPU memory limitation. Except for using genetic algorithm for hyper-parameter search experiments, all other experiments use default setting. Genetic algorithm used YOLOv3-SPP to train with

GIoU loss and search 300 epochs for min-val 5k sets. We adopt searched learning rate 0.00261, momentum 0.949, IoU threshold for assigning ground truth 0.213, and loss normalizer 0.07 for genetic algorithm experiments. We have verified a large number of BoF, including grid sensitivity elimination, mosaic data augmentation, IoU threshold, genetic algorithm, class label smoothing, cross mini-batch normalization, selfadversarial training, cosine annealing scheduler, dynamic mini-batch size, DropBlock, Optimized Anchors, different kind of IoU losses. We also conduct experiments on various BoS, including Mish, SPP, SAM, RFB, BiFPN, and Gaussian YOLO [8]. For all experiments, we only use one GPU for training, so techniques such as syncBN that optimizes multiple GPUs are not used.

在 MS COCO 目标检测实验中，默认参数如下：训练步数为 500500；采用初始学习率为 0.01 的学习率衰减策略，并分别在 40 万步和 45 万步时乘以系数 0.1。动量和权重衰减分别设置为 0.9 和 0.0005。所有的架构使用单个 GPU 进行了多尺度训练，批大小为 64，mini 批大小为 8 或 4，具体取决于模型架构和 GPU 显存容量限制。除了使用使用遗传算法进行超参数搜索外，所有其他实验均使用默认设置。YOLOv3-SPP 使用的遗传算法实验使用 GIoU 损失进行训练，对 min-val 5k 数据集进行了 300 轮的搜索。遗传算法采用搜索学习率为 0.00261、动量为 0.949，真实值的 IoU 阈值设置为 0.213，损失正则化为 0.07。我们也经验证了大量的 BoF，包括 grid sensitivity elimination、

马赛克数据增强、IoU 阈值、遗传算法、类别标签 smoothing、跨小批量标准化、自对抗训练、cosine annealing scheduler、dynamic mini-batch size、DropBlock、Optimized Anchors、不同类型的 IoU 损失。我们也对各种 BoS 进行了实验，包括 Mish、SPP、SAM、RFB、BiFPN、BiFPN 和 Gaussian YOLO[8]。对于所有的实验，我们只使用一个 GPU 进行了训练，因此诸如 syncBN 可以优化多 GPU 训练的技术并没有使用。

4.2. Influence of different features on Classifier training

First, we study the influence of different features on classifier training; specifically, the influence of Class label smoothing, the influence of different data augmentation techniques, bilateral blurring, MixUp, CutMix and Mosaic, as shown in Fugure 7, and the influence of different activations, such as Leaky-ReLU (by default), Swish, and Mish.

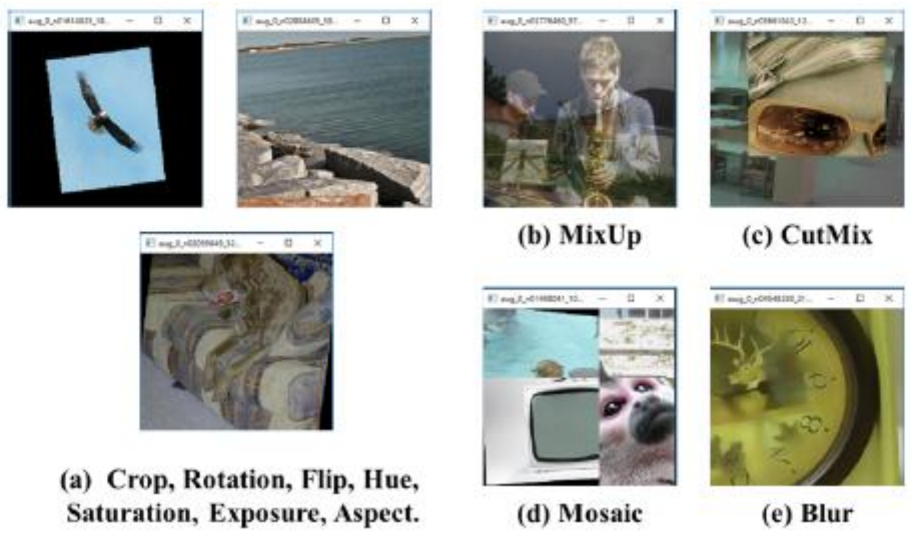
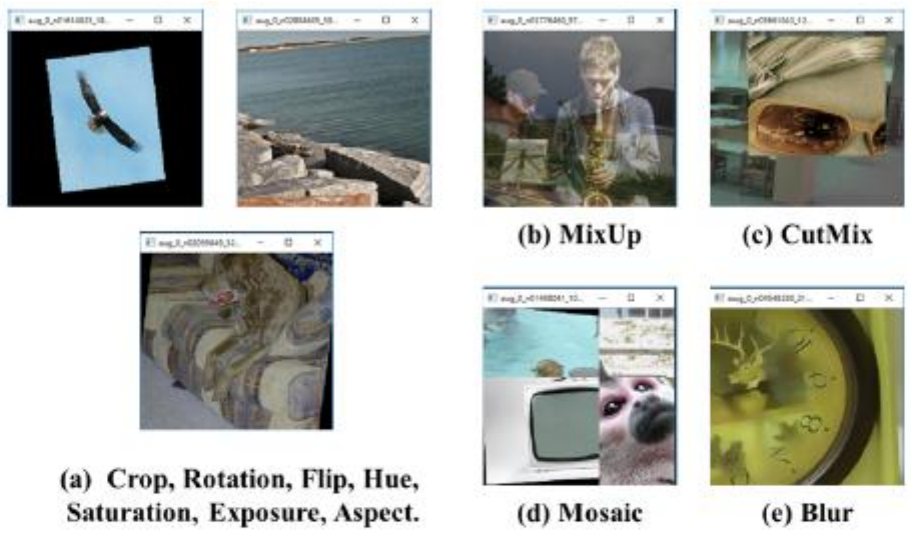

(b) MixUp
(c) CutMix
(a) Crop, Rotation, Flip, Hue, Saturation, Exposure, Aspect.
(d) Mosaic
(e) Blur

Figure 7: Various method of data augmentation.

4.2. 不同技巧对分类器训练的影响

首先，我们研究了不同技巧对分类器训练的影响；具体而言，研究了类别标签 smoothing 的影响，如图 7 所示双边模糊（bilateral blurring）、MixUp、CutMix 和马赛克等不同数据增强的影响，以及 Leaky-ReLU（默认值）、Swish 和 Mish 等不同激活函数的影响。

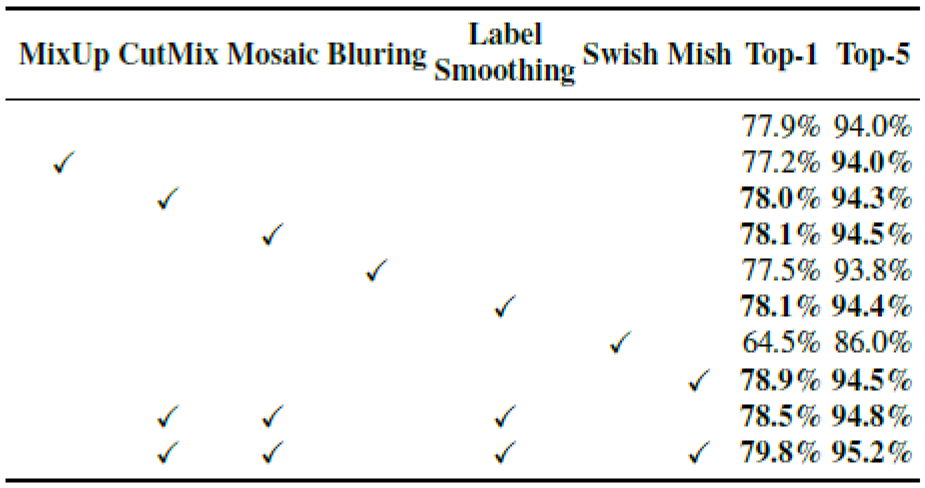


(a) Crop, Rotation, Flip, Hue, Saturation, Exposure, Aspect.
(b) MixUp
(c) CutMix
(d) Mosaic
(e) Blur

图 7：不同的数据增强方法。

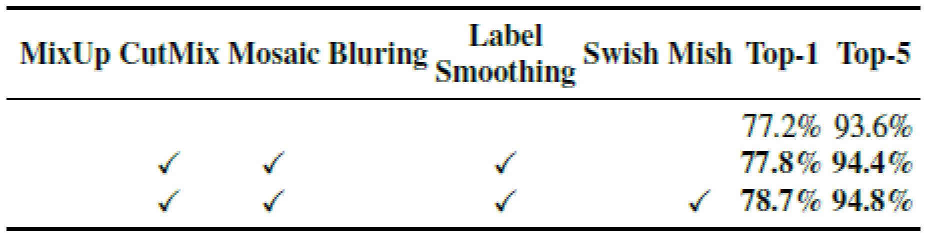
In our experiments, as illustrated in Table 2, the classifier’s accuracy is improved by introducing the features such as: CutMix and Mosaic data augmentation, Class label smoothing, and Mish activation. As a result, our BoF-backbone (Bag of Freebies) for classifier training includes the following: CutMix and Mosaic data augmentation and Class label smoothing. In addition we use Mish activation as a complementary option, as shown in Table 2 and Table 3.

Table 2: Influence of BoF and Mish on the CSPResNeXt-50 classifier accuracy.



MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	94.0%
	✓						78.0%	94.3%
		✓					78.1%	94.5%
			✓				77.5%	93.8%
				✓			78.1%	94.4%
					✓		64.5%	86.0%
						✓	78.9%	94.5%
	✓	✓		✓			78.5%	94.8%
	✓	✓		✓		✓	79.8%	95.2%

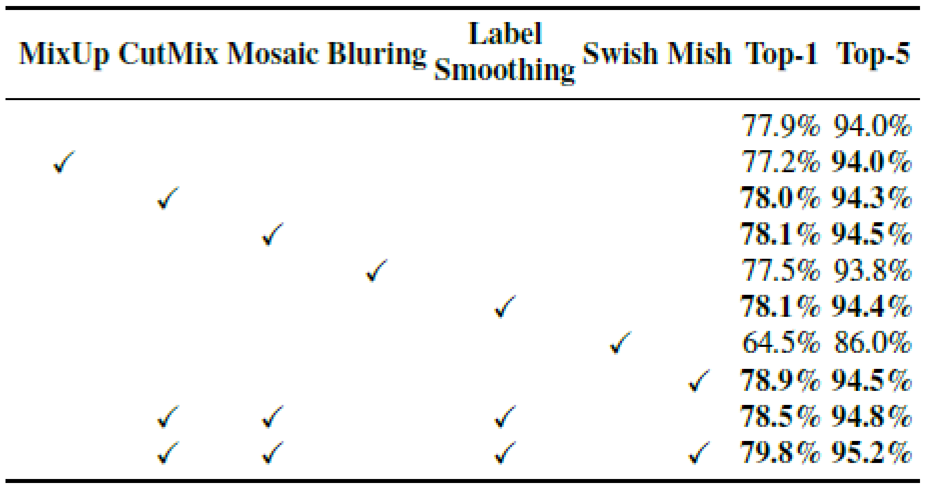
Table 3: Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy.



MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			77.8%	94.4%
	✓	✓		✓		✓	78.7%	94.8%

如表 2 所示，我们的实验引入了以下技巧从而提高了精度，如 CutMix 和马赛克数据增强、类别标签 smoothing 和 Mish 激活函数。因此，我们的分类器训练的 BoF-backbone (Bag of Freebies) 包括 CutMix 和 Mosaic 数据增强、类别标签 smoothing。除此之外，如表 2 和表 3 所示我们还使用了 Mish 激活函数作为互补选项。

表 2：BoF 和 Mish 对 CSPResNeXt-50 分类器精度的影响



MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	94.0%
	✓						78.0%	94.3%
		✓					78.1%	94.5%
			✓				77.5%	93.8%
				✓			78.1%	94.4%
					✓		64.5%	86.0%
						✓	78.9%	94.5%
	✓	✓		✓			78.5%	94.8%
	✓	✓		✓		✓	79.8%	95.2%

表 3：BoF 和 Mish 对 CSPDarknet-53 分类器精度的影响

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			77.8%	94.4%
	✓	✓		✓		✓	78.7%	94.8%

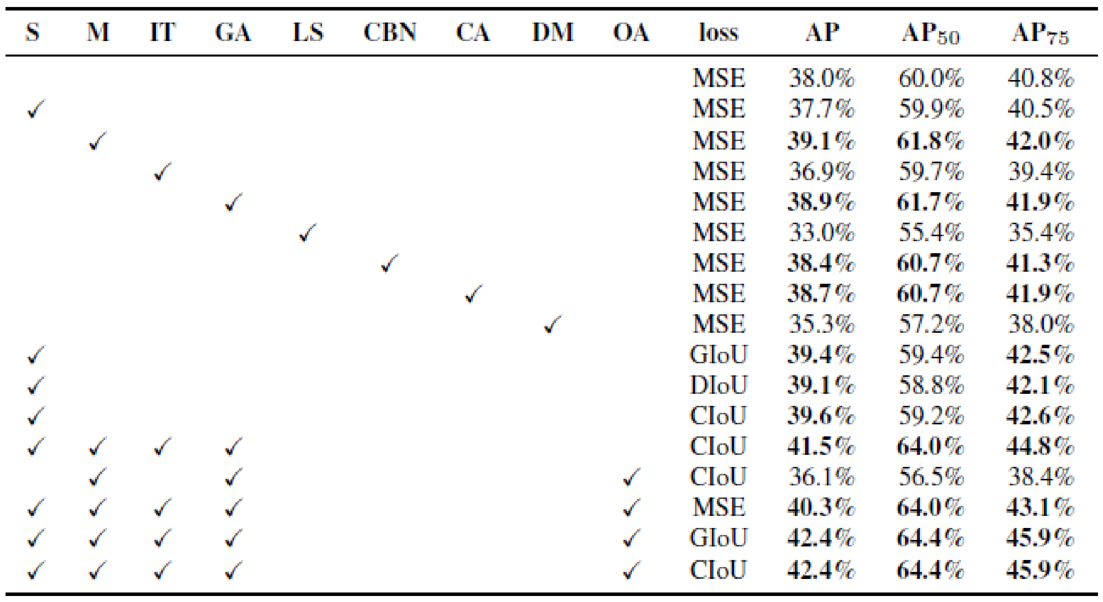
4.3 Influence of different features on Detector training

Further study concerns the influence of different Bag-of-Freebies (BoF-detector) on the detector training accuracy, as shown in Table 4. We significantly expand the BoF list through studying different features that increase the detector accuracy without affecting FPS:

- S: Eliminate grid sensitivity the equation bx=σ(tx)+cx, by=σ(ty)+cy, where cx and cy are always whole numbers, is used in YOLOv3 for evaluating the object coordinates, therefore, extremely high tx absolute values are required for the bx value approaching the cx or cx+1 values. We solve this problem through multiplying the sigmoid by a factor exceeding 1.0, so eliminating the effect of grid on which the object is undetectable.
- M: Mosaic data augmentation – using the 4-image mosaic during training instead of single image
- IT: IoU threshold – using multiple anchors for a single ground truth IoU (truth, anchor) > IoU threshold
- GA: Genetic algorithms – using genetic algorithms for selecting the optimal hyperparameters during network training on the first 10% of time periods

- LS: Class label smoothing – using class label smoothing for sigmoid activation
- CBN: CmBN – using Cross mini-Batch Normalization for collecting statistics inside the entire batch, instead of collecting statistics inside a single mini-batch
- CA: Cosine annealing scheduler – altering the learning rate during sinusoid training
- DM: Dynamic mini-batch size – automatic increase of mini-batch size during small resolution training by using Random training shapes
- OA: Optimized Anchors – using the optimized anchors for training with the 512×512 network resolution
- GIoU, CIoU, DIoU, MSE – using different loss algorithms for bounded box regression

Table 4: Ablation Studies of Bag-of-Freebies. (CSPResNeXt50-PANet-SPP, 512x512).



S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP_{50}	AP_{75}
									MSE	38.0%	60.0%	40.8%
✓									MSE	37.7%	59.9%	40.5%
	✓								MSE	**39.1%**	**61.8%**	**42.0%**
		✓							MSE	36.9%	59.7%	39.4%
			✓						MSE	**38.9%**	**61.7%**	**41.9%**
				✓					MSE	33.0%	55.4%	35.4%
					✓				MSE	**38.4%**	**60.7%**	**41.3%**
						✓			MSE	**38.7%**	**60.7%**	**41.9%**
							✓		MSE	35.3%	57.2%	38.0%
✓									GIoU	**39.4%**	59.4%	**42.5%**
✓									DIoU	**39.1%**	58.8%	**42.1%**
✓									CIoU	**39.6%**	59.2%	**42.6%**
✓	✓	✓	✓						CIoU	**41.5%**	**64.0%**	**44.8%**
	✓		✓					✓	CIoU	36.1%	56.5%	38.4%
✓	✓	✓	✓					✓	MSE	**40.3%**	**64.0%**	**43.1%**
✓	✓	✓	✓					✓	GIoU	**42.4%**	**64.4%**	**45.9%**
✓	✓	✓	✓					✓	CIoU	**42.4%**	**64.4%**	**45.9%**

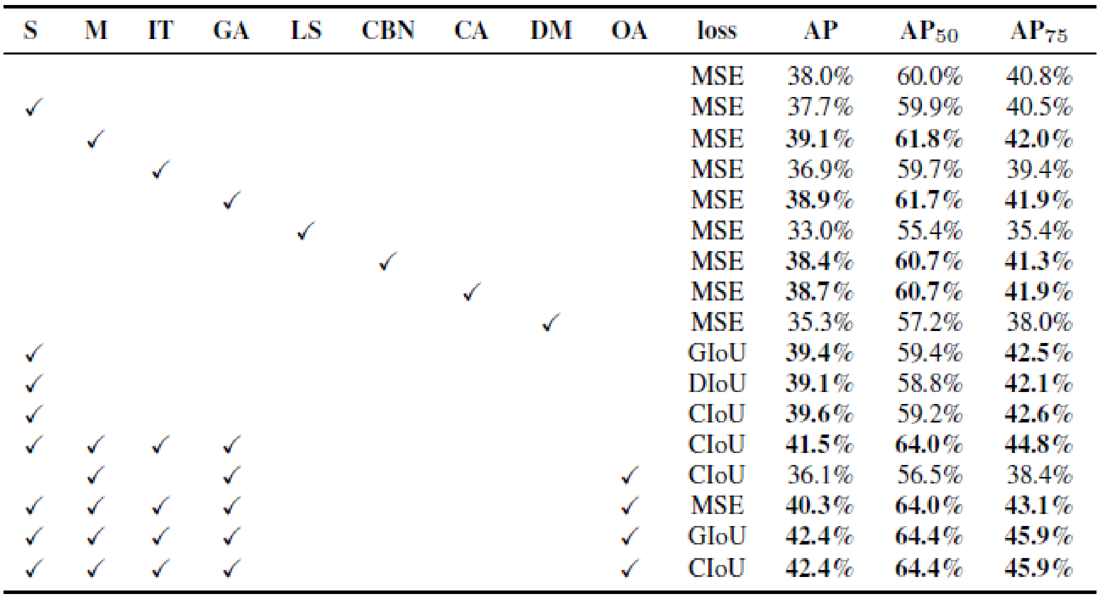
4.3 不同技巧对检测训练的影响

如表 4 所示，深入研究了不同 Bag-of-Freebies (BoF-detector)在检测器训练中的影响。我们通过研究不影响 FPS 的同时能提升精度的技巧，显著扩展了 BOF 列表的内容，具体如下：

- S：消除了格子灵敏度，在 YOLOv3 通过方程 bx=σ(tx)+cx, by=σ(ty)+cy 计算对象坐标，其中 cx 和 cy 始终为整数，因此，当 bx 值接近 cx 或 cx+1 时需要极高的 tx 绝对值。我们通过将 sigmoid 乘以超过 1.0 的因子来解决此问题，从而消除了没有检测到目标格子的影响。
- M：马赛克数据增强——训练时使用 4 张图像的马赛克结果而不是单张图像
- IT：IoU 阈值——针对一个真值边界框使用多个 anchor，Iou（真值，anchor）>IoU 阈值
- GA：遗传算法——在网络训练最初 10%的时间内使用遗传算法筛选最优超参数

- LS：类别标签 smoothing——对 sigmoid 激活函数结果使用类别标签 smoothing
- CBN：CmBN——使用 Cross mini-Batch Normalization 在整个小批量内收集统计数据，而不是在单个 mini 小批量收集统计数据
- CA：Cosine annealing scheduler——在正弦训练中改变学习率
- DM：Dynamic mini-batch size——采用随机训练形状时，对于小分辨率的输入自动增大 mini-batch 的大小
- OA：最优化 Anchors——使用最优化 anchor 对 512×512 的网络分辨率进行训练
- GIoU、CIoU、DIoU、MSE——边界框使用不同的损失算法

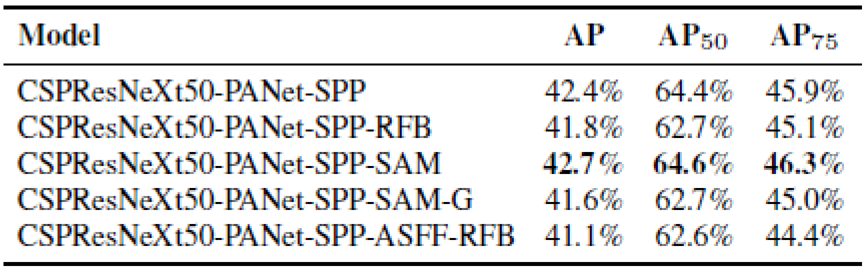
表 4：Bag-of-Freebies 的消融研究(CSPResNeXt50-PANet-SPP，512x512)。



S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP_{50}	AP_{75}
									MSE	38.0%	60.0%	40.8%
✓									MSE	37.7%	59.9%	40.5%
	✓								MSE	**39.1%**	**61.8%**	**42.0%**
		✓							MSE	36.9%	59.7%	39.4%
			✓						MSE	**38.9%**	**61.7%**	**41.9%**
				✓					MSE	33.0%	55.4%	35.4%
					✓				MSE	**38.4%**	**60.7%**	**41.3%**
						✓			MSE	**38.7%**	**60.7%**	**41.9%**
							✓		MSE	35.3%	57.2%	38.0%
✓									GIoU	**39.4%**	59.4%	**42.5%**
✓									DIoU	**39.1%**	58.8%	**42.1%**
✓									CIoU	**39.6%**	59.2%	**42.6%**
✓	✓	✓	✓						CIoU	**41.5%**	**64.0%**	**44.8%**
	✓		✓					✓	CIoU	36.1%	56.5%	38.4%
✓	✓	✓	✓					✓	MSE	**40.3%**	**64.0%**	**43.1%**
✓	✓	✓	✓					✓	GIoU	**42.4%**	**64.4%**	**45.9%**
✓	✓	✓	✓					✓	CIoU	**42.4%**	**64.4%**	**45.9%**

Further study concerns the influence of different Bagof-Specials (BoS-detector) on the detector training accuracy, including PAN, RFB, SAM, Gaussian YOLO (G), and ASFF, as shown in Table 5. In our experiments, the detector gets best performance when using SPP, PAN, and SAM.

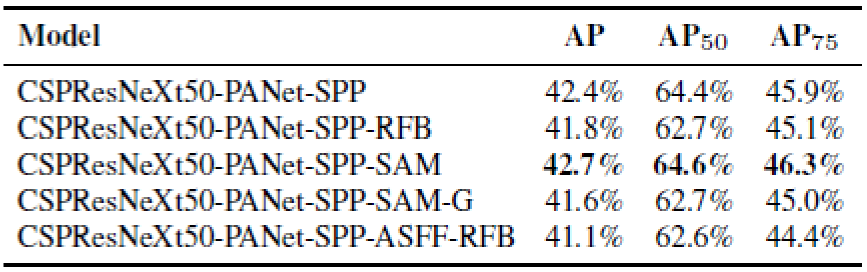
Table 5: Ablation Studies of Bag-of-Specials. (Size 512x512).



Model	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	42.4%	64.4%	45.9%
CSPResNeXt50-PANet-SPP-RFB	41.8%	62.7%	45.1%
CSPResNeXt50-PANet-SPP-SAM	**42.7%**	**64.6%**	**46.3%**
CSPResNeXt50-PANet-SPP-SAM-G	41.6%	62.7%	45.0%
CSPResNeXt50-PANet-SPP-ASFF-RFB	41.1%	62.6%	44.4%

如表 5 所示，进一步的研究涉及了不同的 Bag-of-Specials(BoS-detector)对检测器训练精度的影响，包括 PAN、RFB、SAM、Gaussian YOLO(G)和 ASFF。在我们的实验中，当使用 SPP、PAN 和 SAM 时，检测器获得最佳性能。

表 5：对 Bag-of-Specials 进行消融研究（尺寸 512x512）。

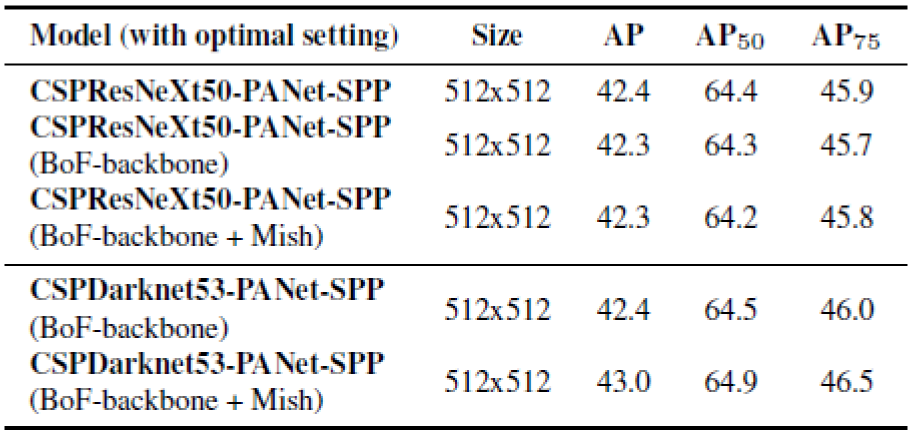


Model	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	42.4%	64.4%	45.9%
CSPResNeXt50-PANet-SPP-RFB	41.8%	62.7%	45.1%
CSPResNeXt50-PANet-SPP-SAM	**42.7%**	**64.6%**	**46.3%**
CSPResNeXt50-PANet-SPP-SAM-G	41.6%	62.7%	45.0%
CSPResNeXt50-PANet-SPP-ASFF-RFB	41.1%	62.6%	44.4%

4.4 Influence of different backbones and pre-trained weightings on Detector training

Further on we study the influence of different backbone models on the detector accuracy, as shown in Table 6. We notice that the model characterized with the best classification accuracy is not always the best in terms of the detector accuracy.

Table 6: Using different classifier pre-trained weightings for detector training (all other training parameters are similar in all models).

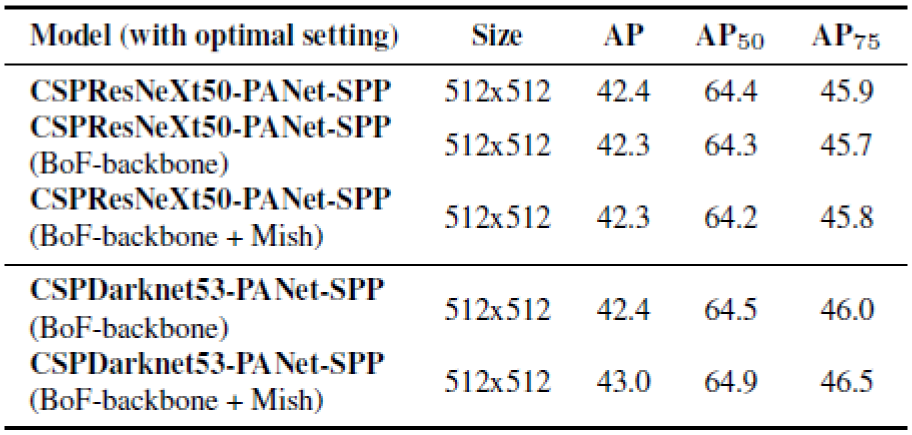


Model (with optimal setting)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	512x512	42.4	64.4	45.9
CSPResNeXt50-PANet-SPP (BoF-backbone)	512x512	42.3	64.3	45.7
CSPResNeXt50-PANet-SPP (BoF-backbone + Mish)	512x512	42.3	64.2	45.8
CSPDarknet53-PANet-SPP (BoF-backbone)	512x512	42.4	64.5	46.0
CSPDarknet53-PANet-SPP (BoF-backbone + Mish)	512x512	43.0	64.9	46.5

4.4 不同 backbone 和预训练权重对检测器训练的影响

如表 6 所示，我们进一步研究不同 backbone 对检测器精度的影响。我们注意到具有最佳的分类精度的模型架构并不总是具有最好的检测精度。

表 6：使用不同分类器预训练的权重进行检测器训练（所有模型中所有其它参数都是相同的）。



Model (with optimal setting)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	512x512	42.4	64.4	45.9
CSPResNeXt50-PANet-SPP (BoF-backbone)	512x512	42.3	64.3	45.7
CSPResNeXt50-PANet-SPP (BoF-backbone + Mish)	512x512	42.3	64.2	45.8
CSPDarknet53-PANet-SPP (BoF-backbone)	512x512	42.4	64.5	46.0
CSPDarknet53-PANet-SPP (BoF-backbone + Mish)	512x512	43.0	64.9	46.5

First, although classification accuracy of CSPResNeXt50 models trained with different features is higher compared to CSPDarknet53 models, the CSPDarknet53 model shows higher accuracy in terms of object detection.

首先，虽然使用不同特征的 CSPResNeXt50 模型的分类准确率高于 CSPDarknet53 模型，但是 CSPDarknet53 模型在目标检测方面具有更高的精度。

Second, using BoF and Mish for the CSPResNeXt50 classifier training increases its classification accuracy, but further application of these pre-trained weightings for detector training reduces the detector accuracy. However, using BoF and Mish for the CSPDarknet53 classifier training increases the accuracy of both the classifier and the detector which uses this classifier pre-trained weightings. The net result is that backbone CSPDarknet53 is more suitable for the detector than for CSPResNeXt50.

其次，CSPResNeXt50 分类器的训练使用 BoF 和 Mish 后提高了其分类精度，但将这些预先训练的权重应用到检测器训练时则降低了检测器的精度。然而，CSPDarknet53 分类器的训练时使用 BoF 和 Mish 均提高了分类器和检测器的精度，检测器使用了分类器预训练的权重。最终的结果是， CSPDarknet53 比 CSPResNeXt50 更适合于做检测器的 backbone。

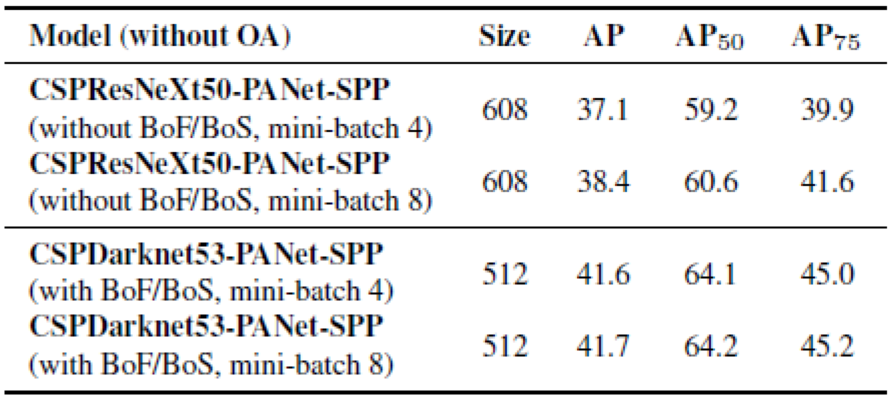
We observe that the CSPDarknet53 model demonstrates a greater ability to increase the detector accuracy owing to various improvements.

我们观察到，CSPDarknet53 模型由于各种改进体现出更大的能力来提高检测器的精度。

4.5 Influence of different mini-batch size on Detector training

Finally, we analyze the results obtained with models trained with different mini-batch sizes, and the results are shown in Table 7. From the results shown in Table 7, we found that after adding BoF and BoS training strategies, the mini-batch size has almost no effect on the detector's performance. This result shows that after the introduction of BoF and BoS, it is no longer necessary to use expensive GPUs for training. In other words, anyone can use only a conventional GPU to train an excellent detector.

Table 7: Using different mini-batch size for detector training.

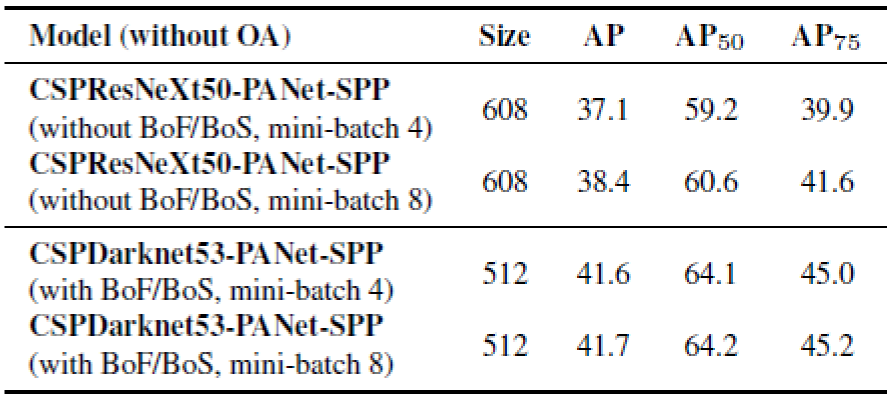


Model (without OA)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 4)	608	37.1	59.2	39.9
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 8)	608	38.4	60.6	41.6
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 4)	512	41.6	64.1	45.0
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 8)	512	41.7	64.2	45.2

4.5 不同的 mini-batch size 对检测器训练的影响

最后，我们分析了模型经过不同 mini-batch 大小的训练的结果，结果图表 7 所示。从表 7 所示的结果来看，我们发现训练时加入 BoF 和 BoS 后 mini-batch 大小几乎对检测器性能没有任何影响。这一结果表明，引入 BoF 和 BoS 后将不再需要使用昂贵的 GPU 来进行训练。换句话说，任何人都可以只使用一个传统的 GPU 来训练一个优秀的检测器。

表 7：使用不同 mini-batch 大小进行检测器训练。



Model (without OA)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 4)	608	37.1	59.2	39.9
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 8)	608	38.4	60.6	41.6
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 4)	512	41.6	64.1	45.0
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 8)	512	41.7	64.2	45.2

5 Results

Comparison of the results obtained with other state-of-the-art object detectors are shown in Figure 8. Our YOLOv4 are located on the Pareto optimality curve and are superior to the fastest and most accurate detectors in terms of both speed and accuracy.

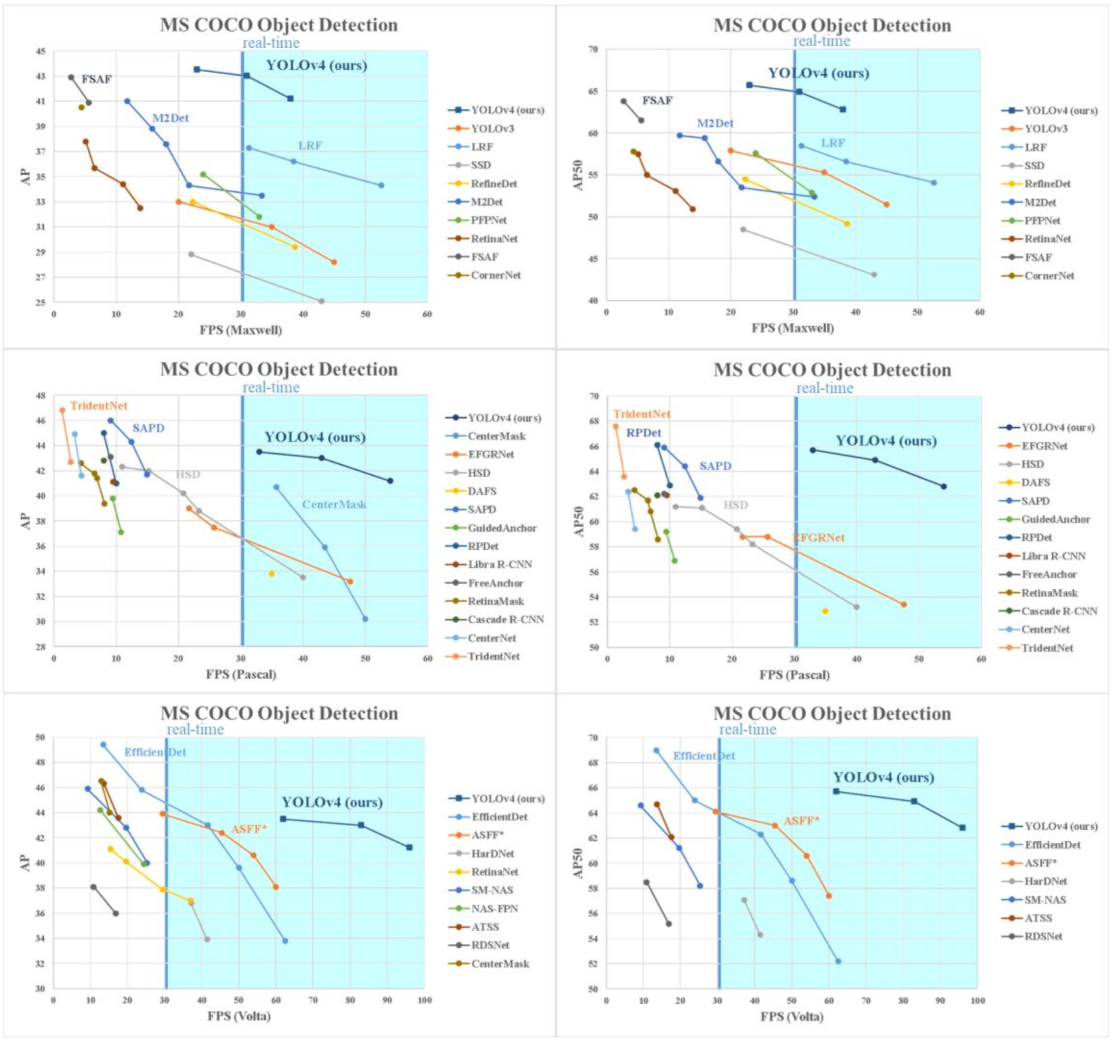


Figure 8: Comparison of the speed and accuracy of different object detectors. (Some articles stated the FPS of their detectors for only one of the GPUs: Maxwell/Pascal/Volta)

5. 结果

如图 8 所示为我们的模型与其他最先进的检测器的比较结果。我们的 YOLOv4 位于帕累托最优曲线上，并且在速度和精度方面都超过最快和最精确的检测器。

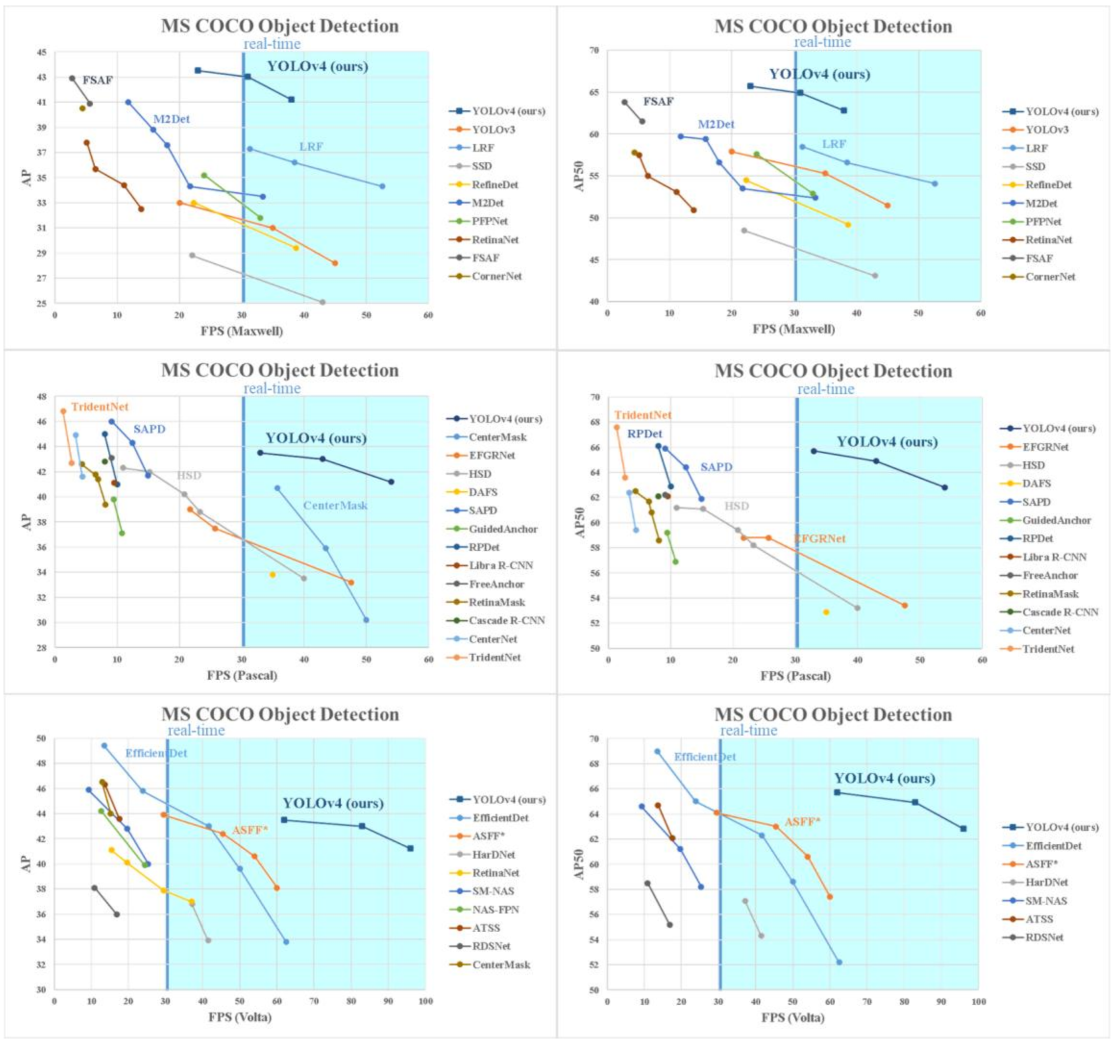


图 8: 不同目标检测器速度和精度的比较。（一些文章指出，它们的检测器的 **FPS** 仅基于某一种 **GPU：Maxwell/Pascal/Volta**）。

Since different methods use GPUs of different architectures for inference time verification, we operate YOLOv4 on commonly adopted GPUs of Maxwell, Pascal, and Volta architectures, and compare them with other state-of-the-art methods. Table 8 lists the frame rate comparison results of using Maxwell GPU, and it can be GTX Titan X (Maxwell) or Tesla M40 GPU. Table 9 lists the frame rate comparison results of using Pascal GPU, and it can be Titan X (Pascal), Titan Xp, GTX 1080 Ti, or Tesla P100 GPU. As for Table 10, it lists the frame rate comparison results of using Volta GPU, and it can be Titan Volta or Tesla V100 GPU.

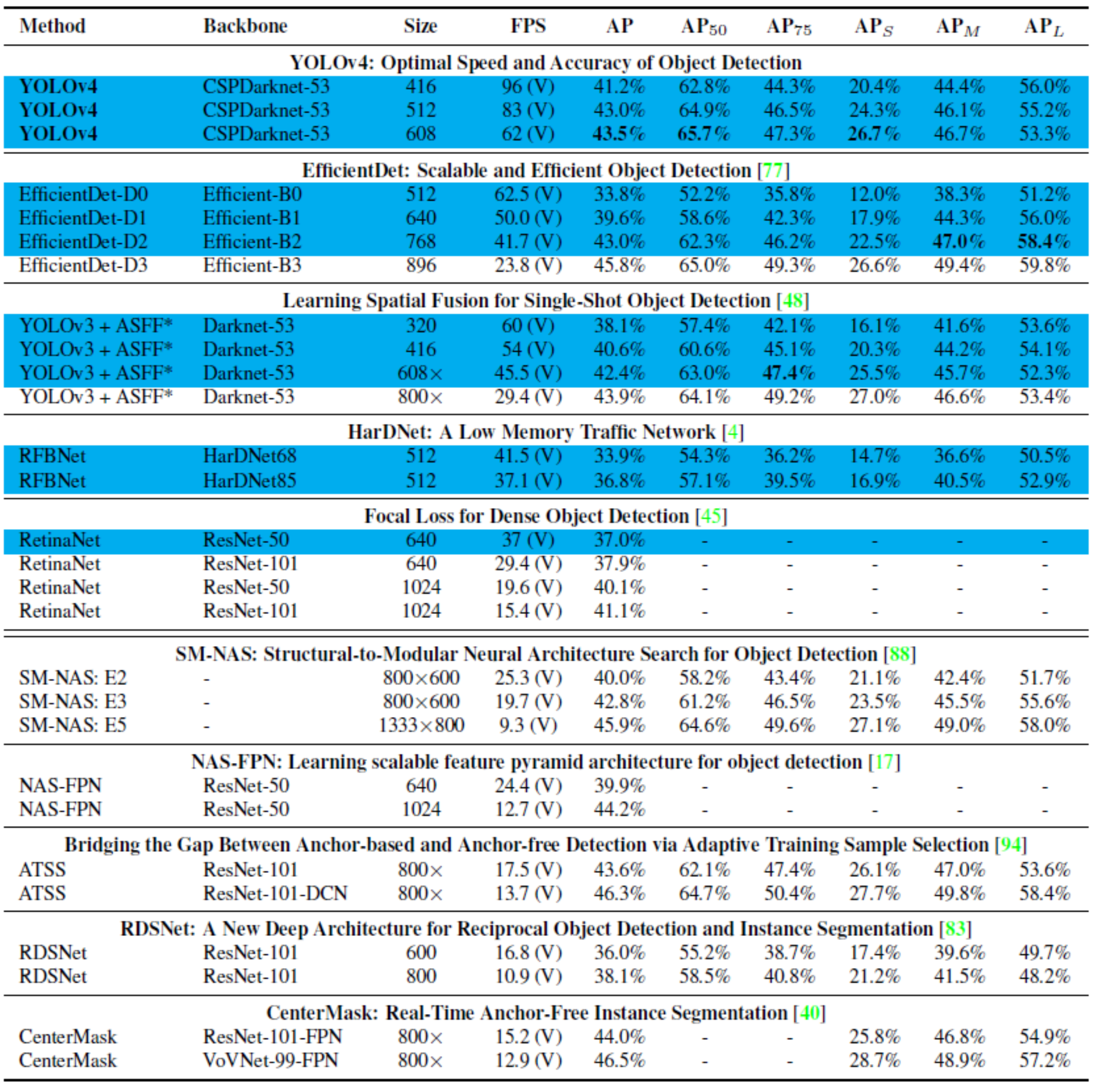
Table 8: Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (test-dev 2017). (Real-time detectors with FPS 30 or higher are highlighted here. We compare the results with batch=1 without using tensorRT.)

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	**43.0%**	**64.9%**	**46.5%**	**24.3%**	**46.1%**	**55.2%**
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
Learning Rich Features at High-Speed for Single-Shot Object Detection [84]									
LRF	VGG-16	300	76.9 (M)	32.0%	51.5%	33.8%	12.6%	34.9%	47.0%
LRF	ResNet-101	300	52.6 (M)	34.3%	54.1%	36.6%	13.2%	38.2%	50.7%
LRF	VGG-16	512	38.5 (M)	36.2%	56.6%	38.7%	19.0%	39.9%	48.8%
LRF	ResNet-101	512	31.3 (M)	37.3%	58.5%	39.7%	19.7%	42.8%	50.1%
Receptive Field Block Net for Accurate and Fast Object Detection [47]									
RFBNet	VGG-16	300	66.7 (M)	30.3%	49.3%	31.8%	11.8%	31.9%	45.9%
RFBNet	VGG-16	512	33.3 (M)	33.8%	54.2%	35.9%	16.2%	37.1%	47.4%
RFBNet-E	VGG-16	512	30.3 (M)	34.4%	55.7%	36.4%	17.6%	37.0%	47.6%
YOLOv3: An incremental improvement [63]									
YOLOv3	Darknet-53	320	45 (M)	28.2%	51.5%	29.7%	11.9%	30.6%	43.4%
YOLOv3	Darknet-53	416	35 (M)	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv3	Darknet-53	608	20 (M)	33.0%	57.9%	34.4%	18.3%	35.4%	41.9%
YOLOv3-SPP	Darknet-53	608	20 (M)	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
SSD: Single shot multibox detector [50]									
SSD	VGG-16	300	43 (M)	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
SSD	VGG-16	512	22 (M)	28.8%	48.5%	30.3%	10.9%	31.8%	43.5%
Single-shot refinement neural network for object detection [95]									
RefineDet	VGG-16	320	38.7 (M)	29.4%	49.2%	31.3%	10.0%	32.0%	44.4%
RefineDet	VGG-16	512	22.3 (M)	33.0%	54.5%	35.5%	16.3%	36.3%	44.3%
M2det: A single-shot object detector based on multi-level feature pyramid network [98]									
M2det	VGG-16	320	33.4 (M)	33.5%	52.4%	35.6%	14.4%	37.6%	47.6%
M2det	ResNet-101	320	21.7 (M)	34.3%	53.5%	36.5%	14.8%	38.8%	47.9%
M2det	VGG-16	512	18 (M)	37.6%	56.6%	40.5%	18.4%	43.4%	51.2%
M2det	ResNet-101	512	15.8 (M)	38.8%	59.4%	41.7%	20.5%	43.9%	53.4%
M2det	VGG-16	800	11.8 (M)	41.0%	59.7%	45.0%	22.1%	46.5%	53.8%
Parallel Feature Pyramid Network for Object Detection [34]									
PFPNet-R	VGG-16	320	33 (M)	31.8%	52.9%	33.6%	12%	35.5%	46.1%
PFPNet-R	VGG-16	512	24 (M)	35.2%	57.6%	37.9%	18.7%	38.6%	45.9%
Focal Loss for Dense Object Detection [45]									
RetinaNet	ResNet-50	500	13.9 (M)	32.5%	50.9%	34.8%	13.9%	35.8%	46.7%
RetinaNet	ResNet-101	500	11.1 (M)	34.4%	53.1%	36.8%	14.7%	38.5%	49.1%
RetinaNet	ResNet-50	800	6.5 (M)	35.7%	55.0%	38.5%	18.9%	38.9%	46.3%
RetinaNet	ResNet-101	800	5.1 (M)	37.8%	57.5%	40.8%	20.2%	41.1%	49.2%
Feature Selective Anchor-Free Module for Single-Shot Object Detection [102]									
AB+FSAF	ResNet-101	800	5.6 (M)	40.9%	61.5%	44.0%	24.0%	44.2%	51.3%
AB+FSAF	ResNeXt-101	800	2.8 (M)	42.9%	63.8%	46.3%	26.6%	46.2%	52.7%
CornerNet: Detecting objects as paired keypoints [37]									
CornerNet	Hourglass	512	4.4 (M)	40.5%	57.8%	45.3%	20.8%	44.8%	56.7%

Table 9: Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (test-dev 2017). (Real-time detectors with FPS 30 or higher are highlighted here. We compare the results with batch=1 without using tensorRT.)

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	54 (P)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	43 (P)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	33 (P)	**43.5%**	**65.7%**	**47.3%**	**26.7%**	**46.7%**	53.3%
CenterMask: Real-Time Anchor-Free Instance Segmentation [40]									
CenterMask-Lite	MobileNetV2-FPN	600×	50.0 (P)	30.2%	-	-	14.2%	31.9%	40.9%
CenterMask-Lite	VoVNet-19-FPN	600×	43.5 (P)	35.9%	-	-	19.6%	38.0%	45.9%
CenterMask-Lite	VoVNet-39-FPN	600×	35.7 (P)	40.7%	-	-	22.4%	43.2%	**53.5%**
Enriched Feature Guided Refinement Network for Object Detection [57]									
EFGRNet	VGG-16	320	47.6 (P)	33.2%	53.4%	35.4%	13.4%	37.1%	47.9%
EFGRNet	VG-G16	512	25.7 (P)	37.5%	58.8%	40.4%	19.7%	41.6%	49.4%
EFGRNet	ResNet-101	512	21.7 (P)	39.0%	58.8%	42.3%	17.8%	43.6%	54.5%
Hierarchical Shot Detector [3]									
HSD	VGG-16	320	40 (P)	33.5%	53.2%	36.1%	15.0%	35.0%	47.8%
HSD	VGG-16	512	23.3 (P)	38.8%	58.2%	42.5%	21.8%	41.9%	50.2%
HSD	ResNet-101	512	20.8 (P)	40.2%	59.4%	44.0%	20.0%	44.4%	54.9%
HSD	ResNeXt-101	512	15.2 (P)	41.9%	61.1%	46.2%	21.8%	46.6%	57.0%
HSD	ResNet-101	768	10.9 (P)	42.3%	61.2%	46.9%	22.8%	47.3%	55.9%
Dynamic anchor feature selection for single-shot object detection [41]									
DAFS	VGG16	512	35 (P)	33.8%	52.9%	36.9%	14.6%	37.0%	47.7%
Soft Anchor-Point Object Detection [101]									
SAPD	ResNet-50	-	14.9 (P)	41.7%	61.9%	44.6%	24.1%	44.6%	51.6%
SAPD	ResNet-50-DCN	-	12.4 (P)	44.3%	64.4%	47.7%	25.5%	47.3%	57.0%
SAPD	ResNet-101-DCN	-	9.1 (P)	46.0%	65.9%	49.6%	26.3%	49.2%	59.6%
Region proposal by guided anchoring [82]									
RetinaNet	ResNet-50	-	10.8 (P)	37.1%	56.9%	40.0%	20.1%	40.1%	48.0%
Faster R-CNN	ResNet-50	-	9.4 (P)	39.8%	59.2%	43.5%	21.8%	42.6%	50.7%
RepPoints: Point set representation for object detection [87]									
RPDet	ResNet-101	-	10 (P)	41.0%	62.9%	44.3%	23.6%	44.1%	51.7%
RPDet	ResNet-101-DCN	-	8 (P)	45.0%	66.1%	49.0%	26.6%	48.6%	57.5%
Libra R-CNN: Towards balanced learning for object detection [58]									
Libra R-CNN	ResNet-101	-	9.5 (P)	41.1%	62.1%	44.7%	23.4%	43.7%	52.5%
FreeAnchor: Learning to match anchors for visual object detection [96]									
FreeAnchor	ResNet-101	-	9.1 (P)	43.1%	62.2%	46.4%	24.5%	46.1%	54.8%
RetinaMask: Learning to Predict Masks Improves State-of-The-Art Single-Shot Detection for Free [14]									
RetinaMask	ResNet-50-FPN	800×	8.1 (P)	39.4%	58.6%	42.3%	21.9%	42.0%	51.0%
RetinaMask	ResNet-101-FPN	800×	6.9 (P)	41.4%	60.8%	44.6%	23.0%	44.5%	53.5%
RetinaMask	ResNet-101-FPN-GN	800×	6.5 (P)	41.7%	61.7%	45.0%	23.5%	44.7%	52.8%
RetinaMask	ResNeXt-101-FPN-GN	800×	4.3 (P)	42.6%	62.5%	46.0%	24.8%	45.6%	53.8%
Cascade R-CNN: Delving into high quality object detection [2]									
Cascade R-CNN	ResNet-101	-	8 (P)	42.8%	62.1%	46.3%	23.7%	45.5%	55.2%
Centernet: Object detection with keypoint triplets [13]									
Centernet	Hourglass-52	-	4.4 (P)	41.6%	59.4%	44.2%	22.5%	43.1%	54.1%
Centernet	Hourglass-104	-	3.3 (P)	44.9%	62.4%	48.1%	25.6%	47.4%	57.4%
Scale-Aware Trident Networks for Object Detection [42]									
TridentNet	ResNet-101	-	2.7 (P)	42.7%	63.6%	46.5%	23.9%	46.6%	56.6%
TridentNet	ResNet-101-DCN	-	1.3 (P)	46.8%	67.6%	51.5%	28.0%	51.2%	60.5%

Table 10: Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (test-dev 2017). (Real-time detectors with FPS 30 or higher are highlighted here. We compare the results with batch=1 without using tensorRT.)



Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	96 (V)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	83 (V)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	62 (V)	**43.5%**	**65.7%**	47.3%	**26.7%**	46.7%	53.3%
EfficientDet: Scalable and Efficient Object Detection [77]									
EfficientDet-D0	Efficient-B0	512	62.5 (V)	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1	Efficient-B1	640	50.0 (V)	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2	Efficient-B2	768	41.7 (V)	43.0%	62.3%	46.2%	22.5%	**47.0%**	**58.4%**
EfficientDet-D3	Efficient-B3	896	23.8 (V)	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
Learning Spatial Fusion for Single-Shot Object Detection [48]									
YOLOv3 + ASFF*	Darknet-53	320	60 (V)	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF*	Darknet-53	416	54 (V)	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv3 + ASFF*	Darknet-53	608×	45.5 (V)	42.4%	63.0%	**47.4%**	25.5%	45.7%	52.3%
YOLOv3 + ASFF*	Darknet-53	800×	29.4 (V)	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%
HarDNet: A Low Memory Traffic Network [4]									
RFBNet	HarDNet68	512	41.5 (V)	33.9%	54.3%	36.2%	14.7%	36.6%	50.5%
RFBNet	HarDNet85	512	37.1 (V)	36.8%	57.1%	39.5%	16.9%	40.5%	52.9%
Focal Loss for Dense Object Detection [45]									
RetinaNet	ResNet-50	640	37 (V)	37.0%	-	-	-	-	-
RetinaNet	ResNet-101	640	29.4 (V)	37.9%	-	-	-	-	-
RetinaNet	ResNet-50	1024	19.6 (V)	40.1%	-	-	-	-	-
RetinaNet	ResNet-101	1024	15.4 (V)	41.1%	-	-	-	-	-
SM-NAS: Structural-to-Modular Neural Architecture Search for Object Detection [88]									
SM-NAS: E2	-	800×600	25.3 (V)	40.0%	58.2%	43.4%	21.1%	42.4%	51.7%
SM-NAS: E3	-	800×600	19.7 (V)	42.8%	61.2%	46.5%	23.5%	45.5%	55.6%
SM-NAS: E5	-	1333×800	9.3 (V)	45.9%	64.6%	49.6%	27.1%	49.0%	58.0%
NAS-FPN: Learning scalable feature pyramid architecture for object detection [17]									
NAS-FPN	ResNet-50	640	24.4 (V)	39.9%	-	-	-	-	-
NAS-FPN	ResNet-50	1024	12.7 (V)	44.2%	-	-	-	-	-
Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection [94]									
ATSS	ResNet-101	800×	17.5 (V)	43.6%	62.1%	47.4%	26.1%	47.0%	53.6%
ATSS	ResNet-101-DCN	800×	13.7 (V)	46.3%	64.7%	50.4%	27.7%	49.8%	58.4%
RDSNet: A New Deep Architecture for Reciprocal Object Detection and Instance Segmentation [83]									
RDSNet	ResNet-101	600	16.8 (V)	36.0%	55.2%	38.7%	17.4%	39.6%	49.7%
RDSNet	ResNet-101	800	10.9 (V)	38.1%	58.5%	40.8%	21.2%	41.5%	48.2%
CenterMask: Real-Time Anchor-Free Instance Segmentation [40]									
CenterMask	ResNet-101-FPN	800×	15.2 (V)	44.0%	-	-	25.8%	46.8%	54.9%
CenterMask	VoVNet-99-FPN	800×	12.9 (V)	46.5%	-	-	28.7%	48.9%	57.2%

由于不同的方法在进行推理时间验证的时候使用了不同架构的 GPU，我们让 YOLOv4 运行在 Maxwell、Pascal 和 Volta 等常用的 GPU 上，并与其他最新技术进行了比较。表 8 列出了使用 Maxwell GPU 时帧率的比较结果，具体型号可以是 GTX Titan X (Maxwell)或 Tesla M40 GPU）。表 9 列出了使用 Pascal GPU 时帧率的比较结果，具体型号可以是 Titan X（Pascal）、Titan Xp、GTX 1080 Ti 或 Tesla P100 GPU。表 10 列出使用 Volta GPU 时帧率的比较结果，具体型号可以是 Titan Volta 或 Tesla V100 GPU。

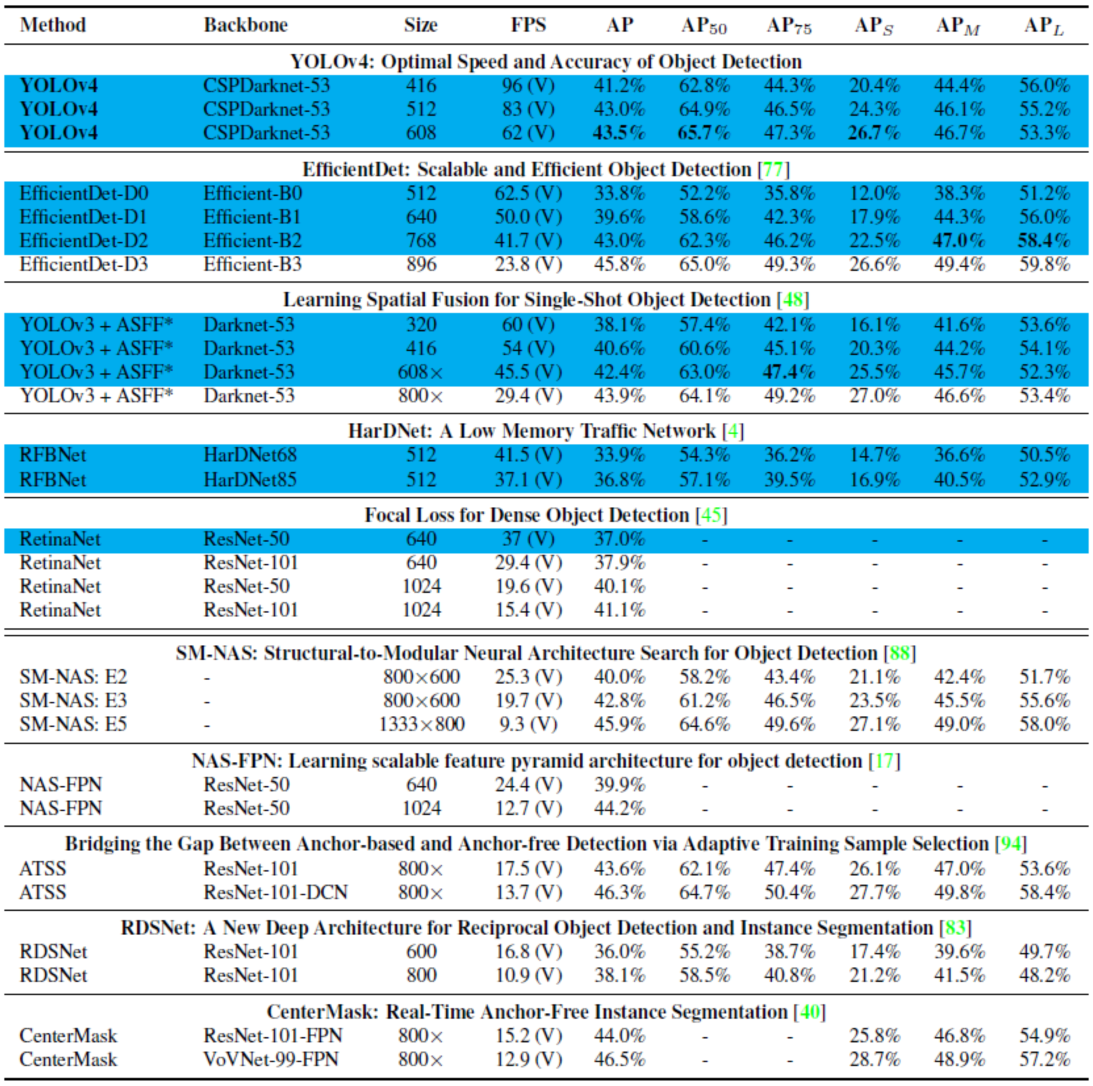
表 8：不同目标检测器在 MS COCO 数据集上的速度和准确性的比较（test-dev 2017）。（FPS 30 或更高的实时检测器突出显示。我们在 batch=1、不使用 tensorRT 的情况下对结果进行了比较。）（译者注：使用 Maxwell GPU）

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	**43.0%**	**64.9%**	**46.5%**	**24.3%**	**46.1%**	**55.2%**
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
Learning Rich Features at High-Speed for Single-Shot Object Detection [84]									
LRF	VGG-16	300	76.9 (M)	32.0%	51.5%	33.8%	12.6%	34.9%	47.0%
LRF	ResNet-101	300	52.6 (M)	34.3%	54.1%	36.6%	13.2%	38.2%	50.7%
LRF	VGG-16	512	38.5 (M)	36.2%	56.6%	38.7%	19.0%	39.9%	48.8%
LRF	ResNet-101	512	31.3 (M)	37.3%	58.5%	39.7%	19.7%	42.8%	50.1%
Receptive Field Block Net for Accurate and Fast Object Detection [47]									
RFBNet	VGG-16	300	66.7 (M)	30.3%	49.3%	31.8%	11.8%	31.9%	45.9%
RFBNet	VGG-16	512	33.3 (M)	33.8%	54.2%	35.9%	16.2%	37.1%	47.4%
RFBNet-E	VGG-16	512	30.3 (M)	34.4%	55.7%	36.4%	17.6%	37.0%	47.6%
YOLOv3: An incremental improvement [63]									
YOLOv3	Darknet-53	320	45 (M)	28.2%	51.5%	29.7%	11.9%	30.6%	43.4%
YOLOv3	Darknet-53	416	35 (M)	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv3	Darknet-53	608	20 (M)	33.0%	57.9%	34.4%	18.3%	35.4%	41.9%
YOLOv3-SPP	Darknet-53	608	20 (M)	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
SSD: Single shot multibox detector [50]									
SSD	VGG-16	300	43 (M)	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
SSD	VGG-16	512	22 (M)	28.8%	48.5%	30.3%	10.9%	31.8%	43.5%
Single-shot refinement neural network for object detection [95]									
RefineDet	VGG-16	320	38.7 (M)	29.4%	49.2%	31.3%	10.0%	32.0%	44.4%
RefineDet	VGG-16	512	22.3 (M)	33.0%	54.5%	35.5%	16.3%	36.3%	44.3%
M2det: A single-shot object detector based on multi-level feature pyramid network [98]									
M2det	VGG-16	320	33.4 (M)	33.5%	52.4%	35.6%	14.4%	37.6%	47.6%
M2det	ResNet-101	320	21.7 (M)	34.3%	53.5%	36.5%	14.8%	38.8%	47.9%
M2det	VGG-16	512	18 (M)	37.6%	56.6%	40.5%	18.4%	43.4%	51.2%
M2det	ResNet-101	512	15.8 (M)	38.8%	59.4%	41.7%	20.5%	43.9%	53.4%
M2det	VGG-16	800	11.8 (M)	41.0%	59.7%	45.0%	22.1%	46.5%	53.8%
Parallel Feature Pyramid Network for Object Detection [34]									
PFPNet-R	VGG-16	320	33 (M)	31.8%	52.9%	33.6%	12%	35.5%	46.1%
PFPNet-R	VGG-16	512	24 (M)	35.2%	57.6%	37.9%	18.7%	38.6%	45.9%
Focal Loss for Dense Object Detection [45]									
RetinaNet	ResNet-50	500	13.9 (M)	32.5%	50.9%	34.8%	13.9%	35.8%	46.7%
RetinaNet	ResNet-101	500	11.1 (M)	34.4%	53.1%	36.8%	14.7%	38.5%	49.1%
RetinaNet	ResNet-50	800	6.5 (M)	35.7%	55.0%	38.5%	18.9%	38.9%	46.3%
RetinaNet	ResNet-101	800	5.1 (M)	37.8%	57.5%	40.8%	20.2%	41.1%	49.2%
Feature Selective Anchor-Free Module for Single-Shot Object Detection [102]									
AB+FSAF	ResNet-101	800	5.6 (M)	40.9%	61.5%	44.0%	24.0%	44.2%	51.3%
AB+FSAF	ResNeXt-101	800	2.8 (M)	42.9%	63.8%	46.3%	26.6%	46.2%	52.7%
CornerNet: Detecting objects as paired keypoints [37]									
CornerNet	Hourglass	512	4.4 (M)	40.5%	57.8%	45.3%	20.8%	44.8%	56.7%

表 9：不同目标检测器在 MS COCO 数据集上的速度和准确性的比较（test-dev 2017）。（FPS 30 或更高的实时检测器突出显示。我们在 batch=1、不使用 tensorRT 的情况下对结果进行了比较。）（译者注：使用 Pascal GPU）

Method	Backbone	Size	FPS	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
			YOLOv4: Optimal Speed and Accuracy of Object Detection						
YOLOv4	CSPDarknet-53	416	54 (P)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	43 (P)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	33 (P)	**43.5%**	**65.7%**	**47.3%**	**26.7%**	**46.7%**	53.3%
			CenterMask: Real-Time Anchor-Free Instance Segmentation [40]						
CenterMask-Lite	MobileNetV2-FPN	600×	50.0 (P)	30.2%	-	-	14.2%	31.9%	40.9%
CenterMask-Lite	VoVNet-19-FPN	600×	43.5 (P)	35.9%	-	-	19.6%	38.0%	45.9%
CenterMask-Lite	VoVNet-39-FPN	600×	35.7 (P)	40.7%	-	-	22.4%	43.2%	**53.5%**
			Enriched Feature Guided Refinement Network for Object Detection [57]						
EFGRNet	VGG-16	320	47.6 (P)	33.2%	53.4%	35.4%	13.4%	37.1%	47.9%
EFGRNet	VG-G16	512	25.7 (P)	37.5%	58.8%	40.4%	19.7%	41.6%	49.4%
EFGRNet	ResNet-101	512	21.7 (P)	39.0%	58.8%	42.3%	17.8%	43.6%	54.5%
			Hierarchical Shot Detector [3]						
HSD	VGG-16	320	40 (P)	33.5%	53.2%	36.1%	15.0%	35.0%	47.8%
HSD	VGG-16	512	23.3 (P)	38.8%	58.2%	42.5%	21.8%	41.9%	50.2%
HSD	ResNet-101	512	20.8 (P)	40.2%	59.4%	44.0%	20.0%	44.4%	54.9%
HSD	ResNeXt-101	512	15.2 (P)	41.9%	61.1%	46.2%	21.8%	46.6%	57.0%
HSD	ResNet-101	768	10.9 (P)	42.3%	61.2%	46.9%	22.8%	47.3%	55.9%
			Dynamic anchor feature selection for single-shot object detection [41]						
DAFS	VGG16	512	35 (P)	33.8%	52.9%	36.9%	14.6%	37.0%	47.7%
			Soft Anchor-Point Object Detection [101]						
SAPD	ResNet-50	-	14.9 (P)	41.7%	61.9%	44.6%	24.1%	44.6%	51.6%
SAPD	ResNet-50-DCN	-	12.4 (P)	44.3%	64.4%	47.7%	25.5%	47.3%	57.0%
SAPD	ResNet-101-DCN	-	9.1 (P)	46.0%	65.9%	49.6%	26.3%	49.2%	59.6%
			Region proposal by guided anchoring [82]						
RetinaNet	ResNet-50	-	10.8 (P)	37.1%	56.9%	40.0%	20.1%	40.1%	48.0%
Faster R-CNN	ResNet-50	-	9.4 (P)	39.8%	59.2%	43.5%	21.8%	42.6%	50.7%
			RepPoints: Point set representation for object detection [87]						
RPDet	ResNet-101	-	10 (P)	41.0%	62.9%	44.3%	23.6%	44.1%	51.7%
RPDet	ResNet-101-DCN	-	8 (P)	45.0%	66.1%	49.0%	26.6%	48.6%	57.5%
			Libra R-CNN: Towards balanced learning for object detection [58]						
Libra R-CNN	ResNet-101	-	9.5 (P)	41.1%	62.1%	44.7%	23.4%	43.7%	52.5%
			FreeAnchor: Learning to match anchors for visual object detection [96]						
FreeAnchor	ResNet-101	-	9.1 (P)	43.1%	62.2%	46.4%	24.5%	46.1%	54.8%
			RetinaMask: Learning to Predict Masks Improves State-of-The-Art Single-Shot Detection for Free [14]						
RetinaMask	ResNet-50-FPN	800×	8.1 (P)	39.4%	58.6%	42.3%	21.9%	42.0%	51.0%
RetinaMask	ResNet-101-FPN	800×	6.9 (P)	41.4%	60.8%	44.6%	23.0%	44.5%	53.5%
RetinaMask	ResNet-101-FPN-GN	800×	6.5 (P)	41.7%	61.7%	45.0%	23.5%	44.7%	52.8%
RetinaMask	ResNeXt-101-FPN-GN	800×	4.3 (P)	42.6%	62.5%	46.0%	24.8%	45.6%	53.8%
			Cascade R-CNN: Delving into high quality object detection [2]						
Cascade R-CNN	ResNet-101	-	8 (P)	42.8%	62.1%	46.3%	23.7%	45.5%	55.2%
			Centernet: Object detection with keypoint triplets [13]						
Centernet	Hourglass-52	-	4.4 (P)	41.6%	59.4%	44.2%	22.5%	43.1%	54.1%
Centernet	Hourglass-104	-	3.3 (P)	44.9%	62.4%	48.1%	25.6%	47.4%	57.4%
			Scale-Aware Trident Networks for Object Detection [42]						
TridentNet	ResNet-101	-	2.7 (P)	42.7%	63.6%	46.5%	23.9%	46.6%	56.6%
TridentNet	ResNet-101-DCN	-	1.3 (P)	46.8%	67.6%	51.5%	28.0%	51.2%	60.5%

表 10：不同目标检测器在 MS COCO 数据集上的速度和准确性的比较（test-dev 2017）。（FPS 30 或更高的实时检测器突出显示。我们在 batch=1、不使用 tensorRT 的情况下对结果进行了比较。）（译者注：使用 Volta GPU）



Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	96 (V)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	83 (V)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	62 (V)	**43.5%**	**65.7%**	47.3%	**26.7%**	46.7%	53.3%
EfficientDet: Scalable and Efficient Object Detection [77]									
EfficientDet-D0	Efficient-B0	512	62.5 (V)	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1	Efficient-B1	640	50.0 (V)	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2	Efficient-B2	768	41.7 (V)	43.0%	62.3%	46.2%	22.5%	**47.0%**	**58.4%**
EfficientDet-D3	Efficient-B3	896	23.8 (V)	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
Learning Spatial Fusion for Single-Shot Object Detection [48]									
YOLOv3 + ASFF*	Darknet-53	320	60 (V)	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF*	Darknet-53	416	54 (V)	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv3 + ASFF*	Darknet-53	608×	45.5 (V)	42.4%	63.0%	**47.4%**	25.5%	45.7%	52.3%
YOLOv3 + ASFF*	Darknet-53	800×	29.4 (V)	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%
HarDNet: A Low Memory Traffic Network [4]									
RFBNet	HarDNet68	512	41.5 (V)	33.9%	54.3%	36.2%	14.7%	36.6%	50.5%
RFBNet	HarDNet85	512	37.1 (V)	36.8%	57.1%	39.5%	16.9%	40.5%	52.9%
Focal Loss for Dense Object Detection [45]									
RetinaNet	ResNet-50	640	37 (V)	37.0%	-	-	-	-	-
RetinaNet	ResNet-101	640	29.4 (V)	37.9%	-	-	-	-	-
RetinaNet	ResNet-50	1024	19.6 (V)	40.1%	-	-	-	-	-
RetinaNet	ResNet-101	1024	15.4 (V)	41.1%	-	-	-	-	-
SM-NAS: Structural-to-Modular Neural Architecture Search for Object Detection [88]									
SM-NAS: E2	-	800×600	25.3 (V)	40.0%	58.2%	43.4%	21.1%	42.4%	51.7%
SM-NAS: E3	-	800×600	19.7 (V)	42.8%	61.2%	46.5%	23.5%	45.5%	55.6%
SM-NAS: E5	-	1333×800	9.3 (V)	45.9%	64.6%	49.6%	27.1%	49.0%	58.0%
NAS-FPN: Learning scalable feature pyramid architecture for object detection [17]									
NAS-FPN	ResNet-50	640	24.4 (V)	39.9%	-	-	-	-	-
NAS-FPN	ResNet-50	1024	12.7 (V)	44.2%	-	-	-	-	-
Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection [94]									
ATSS	ResNet-101	800×	17.5 (V)	43.6%	62.1%	47.4%	26.1%	47.0%	53.6%
ATSS	ResNet-101-DCN	800×	13.7 (V)	46.3%	64.7%	50.4%	27.7%	49.8%	58.4%
RDSNet: A New Deep Architecture for Reciprocal Object Detection and Instance Segmentation [83]									
RDSNet	ResNet-101	600	16.8 (V)	36.0%	55.2%	38.7%	17.4%	39.6%	49.7%
RDSNet	ResNet-101	800	10.9 (V)	38.1%	58.5%	40.8%	21.2%	41.5%	48.2%
CenterMask: Real-Time Anchor-Free Instance Segmentation [40]									
CenterMask	ResNet-101-FPN	800×	15.2 (V)	44.0%	-	-	25.8%	46.8%	54.9%
CenterMask	VoVNet-99-FPN	800×	12.9 (V)	46.5%	-	-	28.7%	48.9%	57.2%

6. Conclusions

We offer a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO AP 50⋯95 and AP 50) than all available alternative detectors. The detector described can be trained and used on a conventional GPU with 8-16GB-VRAM this makes its broad use possible. The original concept of one-stage anchor-based detectors has proven its viability. We have verified a large number of features, and selected for use such of them for improving the accuracy of both the classifier and the detector. These features can be used as best-practice for future studies and developments.

6. 结论

我们提供了一个最先进的检测器，相比于其它所有可用、可替代的检测器其速度更快（FPS）、更准确（MS COCO $AP_{50...95}$ 和 AP_{50}）。该检测器可以在 8-16GB-VRAM 的传统 GPU 上训练和使用，这使得它能够被广泛使用。基于一阶段 anchor 原始概念的检测测器已经被证实是可行的。我们已经验证了大量方法，并选择使用其中一些方法以提高分类器和检测器的准确性。这些方法可以用作未来研究和开发的最佳实践。

7. Acknowledgements

The authors wish to thank Glenn Jocher for the ideas of Mosaic data augmentation, the selection of hyper-parameters by using genetic algorithms and solving the grid sensitivity problem https://github.com/ultralytics/yolov3.

7. 致谢

作者们感谢 Glenn Jocher 关于马赛克数据增强、使用遗传算法进行超参数筛选和解决格子敏感问题的想法，https://github.com/ultralytics/yolov3。

References

参考文献

[1] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-NMS–improving object detection with one line of code. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 5561–5569, 2017. 4

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6154–6162, 2018. 12

[3] Jiale Cao, Yanwei Pang, Jungong Han, and Xuelong Li. Hierarchical shot detector. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9705–9714, 2019. 12
[4] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. HarDNet: A low memory traffic network. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019. 13
[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 40(4):834–848, 2017. 2, 4
[6] Pengguang Chen. GridMask data augmentation. arXiv preprint arXiv:2001.04086, 2020. 3
[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. DetNAS: Backbone search for object detection. In Advances in Neural Information Processing Systems (NeurIPS), pages 6638–6648, 2019. 2
[8] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 502–511, 2019. 7
[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In Advances in Neural Information Processing Systems (NIPS), pages 379–387, 2016. 2
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009. 5
[11] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with CutOut. arXiv preprint arXiv:1708.04552, 2017. 3
[12] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. SpineNet: Learning scale-permuted backbone for recognition and localization. arXiv preprint arXiv:1912.05027, 2019. 2
[13] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6569–6578, 2019. 2, 12
[14] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. arXiv preprint arXiv:1901.03353, 2019. 12
[15] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In International Conference on Learning Representations (ICLR), 2019. 3
[16] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. DropBlock: A regularization method for convolutional networks. In Advances in Neural Information Processing Systems (NIPS), pages 10727–10737, 2018. 3
[17] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. NAS-FPN: Learning scalable

feature pyramid architecture for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 70367045, 2019. 2, 13
[18] Ross Girshick. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 1440–1448, 2015. 2
[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 580–587, 2014. 2, 4
[20] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. HitDetector: Hierarchical trinity architecture search for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2
[21] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. GhostNet: More features from cheap operations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 5
[22] Bharath Hariharan, Pablo Arbel´aez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 447–456, 2015. 4
[23] Kaiming He, Georgia Gkioxari, Piotr Doll´ar, and Ross Girshick. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2961–2969, 2017. 2
[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 1026–1034, 2015. 4
[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 37(9):1904–1916, 2015. 2, 4, 7
[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceed-ings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 2
[27] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019. 2, 4
[28] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017. 2, 4
[29] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 71327141, 2018. 4

[30] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 47004708, 2017. 2
[31] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016. 2
[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015. 6
[33] Md Amirul Islam, Shujon Naha, Mrigank Rochan, Neil Bruce, and Yang Wang. Label refinement network for coarse-to-fine semantic segmentation. arXiv preprint arXiv:1703.00551, 2017. 3
[34] Seung-Wook Kim, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. Parallel feature pyramid network for object detection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 234–250, 2018. 11
[35] G¨unter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 971–980, 2017. 4
[36] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648, 2016. 6
[37] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), pages 734–750, 2018. 2, 11
[38] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. CornerNet-Lite: Efficient keypoint based object detection. arXiv preprint arXiv:1904.08900, 2019. 2
[39] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 2169–2178. IEEE, 2006. 4
[40] Youngwan Lee and Jongyoul Park. CenterMask: Real-time anchor-free instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 12, 13
[41] Shuai Li, Lingxiao Yang, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Dynamic anchor feature selection for single-shot object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6609–6618, 2019. 12
[42] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6054–6063, 2019. 12
[43] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. DetNet: Design backbone for object detection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 334–350, 2018. 2
[44] Tsung-Yi Lin, Piotr Doll´ar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2117–2125, 2017. 2

[45] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Doll´ar. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2980–2988, 2017. 2, 3, 11, 13

[46] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll´ar, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In Proceedings of the European Conference on Computer Vision (ECCV), pages 740–755, 2014. 5

[47] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 385–400, 2018. 2, 4, 11

[48] Songtao Liu, Di Huang, and Yunhong Wang. Learning spatial fusion for single-shot object detection. arXiv preprint arXiv:1911.09516, 2019. 2, 4, 13

[49] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8759–8768, 2018. 1, 2, 7

[50] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV), pages 21–37, 2016. 2, 11

[51] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, 2015. 4

[52] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016. 7

[53] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNetV2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), pages 116–131, 2018. 2

[54] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of International Conference on Machine Learning (ICML), volume 30, page 3, 2013. 4

[55] Diganta Misra. Mish: A self regularized nonmonotonic neural activation function. arXiv preprint arXiv:1908.08681, 2019. 4

[56] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of International Conference on Machine Learning (ICML), pages 807–814, 2010. 4

[57] Jing Nie, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Enriched feature guided refinement network for object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9537–9546, 2019. 12

[58] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: Towards balanced learning for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages

821–830, 2019. 2, 12
[59] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2017. 4
[60] Abdullah Rashwan, Agastya Kalra, and Pascal Poupart. Matrix Nets: A new deep architecture for object detection. In Proceedings of the IEEE International Conference on Computer Vision Workshop (ICCV Workshop), pages 0–0, 2019. 2
[61] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779788, 2016. 2
[62] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 72637271, 2017. 2
[63] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018. 2, 4, 7, 11
[64] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems (NIPS), pages 91–99, 2015. 2
[65] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 658–666, 2019. 3
[66] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4510–4520, 2018. 2
[67] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 761–769, 2016. 3
[68] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 2
[69] Krishna Kumar Singh, Hao Yu, Aron Sarmasi, Gautam Pradeep, and Yong Jae Lee. Hide-and-Seek: A data augmentation technique for weakly-supervised localization and beyond. arXiv preprint arXiv:1811.02545, 2018. 3
[70] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. arXiv preprint arXiv:1911.09737, 2019. 6
[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. DropOut: A simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014. 3
[72] K-K Sung and Tomaso Poggio. Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 20(1):39–51, 1998. 3
[73] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew

Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818–2826, 2016. 3

[74] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MNASnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2820–2828, 2019. 2

[75] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of International Conference on Machine Learning (ICML), 2019. 2

[76] Mingxing Tan and Quoc V Le. MixNet: Mixed depthwise convolutional kernels. In Proceedings of the British Machine Vision Conference (BMVC), 2019. 5

[77] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and efficient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2, 4, 13

[78] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9627–9636, 2019. 2

[79] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 648–656, 2015. 6

[80] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using DropConnect. In Proceedings of International Conference on Machine Learning (ICML), pages 1058–1066, 2013. 3

[81] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of cnn. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop), 2020. 2, 7

[82] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2965–2974, 2019. 12

[83] Shaoru Wang, Yongchao Gong, Junliang Xing, Lichao Huang, Chang Huang, and Weiming Hu. RDSNet: A new deep architecture for reciprocal object detection and instance segmentation. arXiv preprint arXiv:1912.05070, 2019. 13

[84] Tiancai Wang, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Learning rich features at high-speed for single-shot object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 1971–1980, 2019. 11

[85] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), pages 3–19, 2018. 1, 2, 4

[86] Saining Xie, Ross Girshick, Piotr Doll´ar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1492–1500, 2017. 2
[87] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. RepPoints: Point set representation for object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9657–9666, 2019. 2, 12
[88] Lewei Yao, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. SM-NAS: Structural-to-modular neural architecture search for object detection. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020. 13
[89] Zhuliang Yao, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. Cross-iteration batch normalization. arXiv preprint arXiv:2002.05712, 2020. 1, 6
[90] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. UnitBox: An advanced object detection network. In Proceedings of the 24th ACM international conference on Multimedia, pages 516–520, 2016. 3
[91] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6023–6032, 2019. 3
[92] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. MixUp: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017. 3
[93] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7151–7160, 2018. 6
[94] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 13
[95] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4203–4212, 2018. 11
[96] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. FreeAnchor: Learning to match anchors for visual object detection. In Advances in Neural Information Processing Systems (NeurIPS), 2019. 12
[97] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6848–6856, 2018. 2
[98] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), volume 33, pages 9259–9266, 2019. 2, 4, 11
[99] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU Loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020. 3, 4

[100] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. arXiv preprint arXiv:1708.04896, 2017. 3
[101] Chenchen Zhu, Fangyi Chen, Zhiqiang Shen, and Marios Savvides. Soft anchor-point object detection. arXiv preprint arXiv:1911.12448, 2019. 12
[102] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 840–849, 2019. 11