arXiv:2004.10934v1 [cs.CV] 23 Apr 2020

YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

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

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

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.

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.

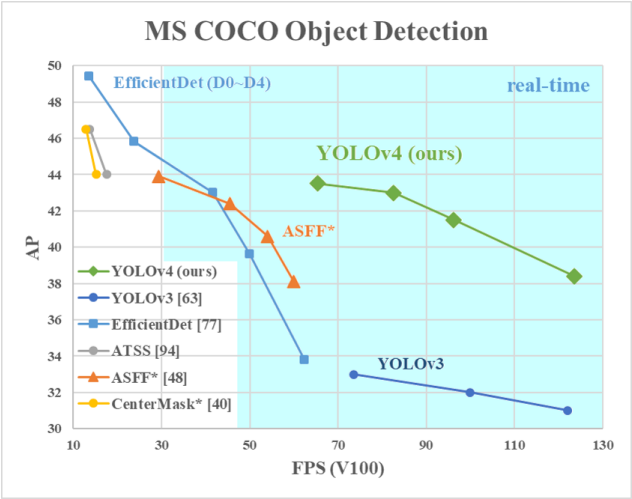


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

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-of-Freebies 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.

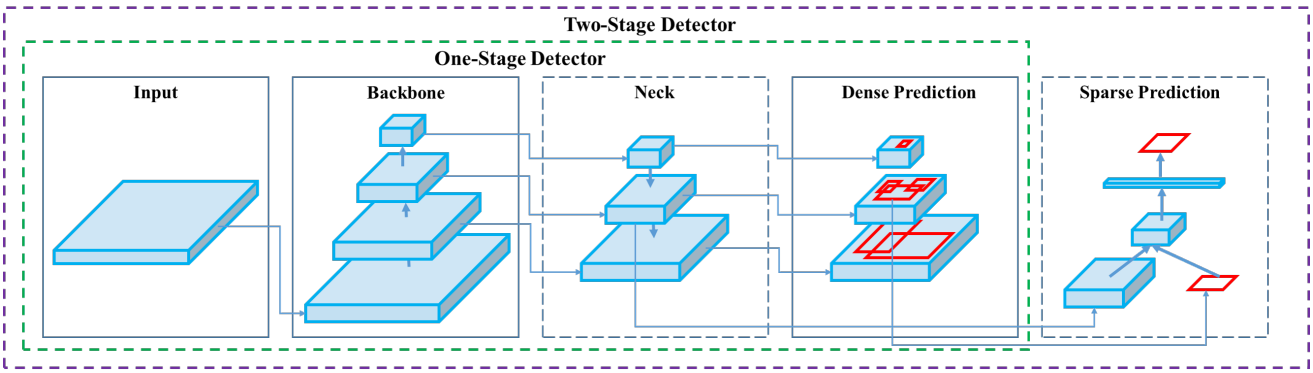


Figure 2: Object detector.

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.

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 R-CNN [23] (anchor based)
    - RepPoints [87] (anchor free)

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.

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.

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.

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 l_1 or l_2 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.

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.

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 $d \times d$ 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 $k \times k$, where $k = \{1, 5, 9, 13\}$, and stride equals to 1. Under this design, a relatively large $k \times k$ max-pooling 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 $k \times k$ 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 $k \times k$ 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%.

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.

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.

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.

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.

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

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

3.1. 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].

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.

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

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.

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

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.

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

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.

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.

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)

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 requlariza tion, 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.

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)

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.

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

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.

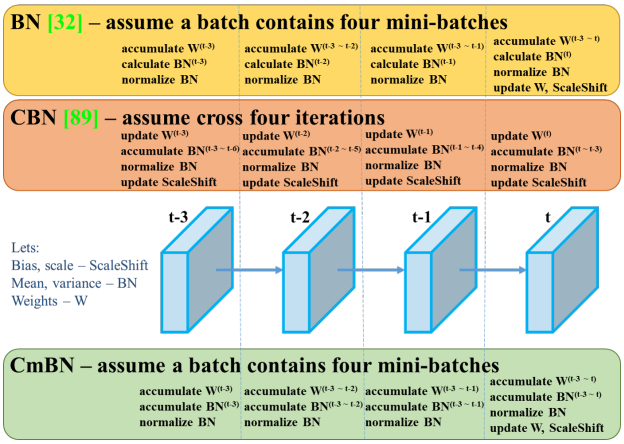


Figure 4: Cross mini-Batch Normalization.

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.

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

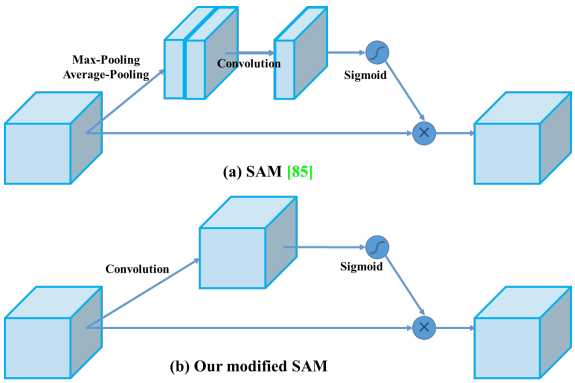


Figure 5: Modified SAM.

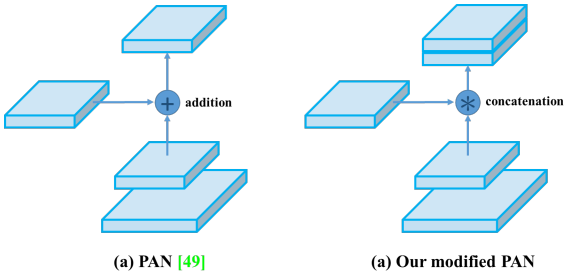


Figure 6: Modified PAN.

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), Multi-input 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 hyper-parameters, Random training shapes
- Bag of Specials (BoS) for detector: Mish activation, 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.

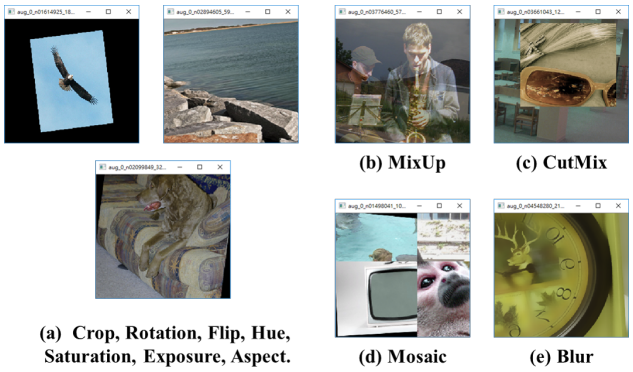
4.1. Experimental setup

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.

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, self-adversarial 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.

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.



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

Figure 7: Various method of data augmentation.

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%**

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 $b_x = \sigma(t_x) + c_x, b_y = \sigma(t_y) + c_y$, where c_x and c_y are always whole numbers, is used in YOLOv3 for evaluating the object coordinates, therefore, extremely high t_x absolute values are required for the b_x value approaching the c_x or $c_x + 1$ 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 512x512 network resolution

- GIoU, CIoU, DIoU, MSE - using different loss algorithms for bounded box regression

Further study concerns the influence of different Bag-of-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 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%**

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%

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.

First, although classification accuracy of CSPResNeXt-50 models trained with different features is higher compared to CSPDarknet53 models, the CSPDarknet53 model shows higher accuracy in terms of object detection.

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.

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

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

4.5. 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

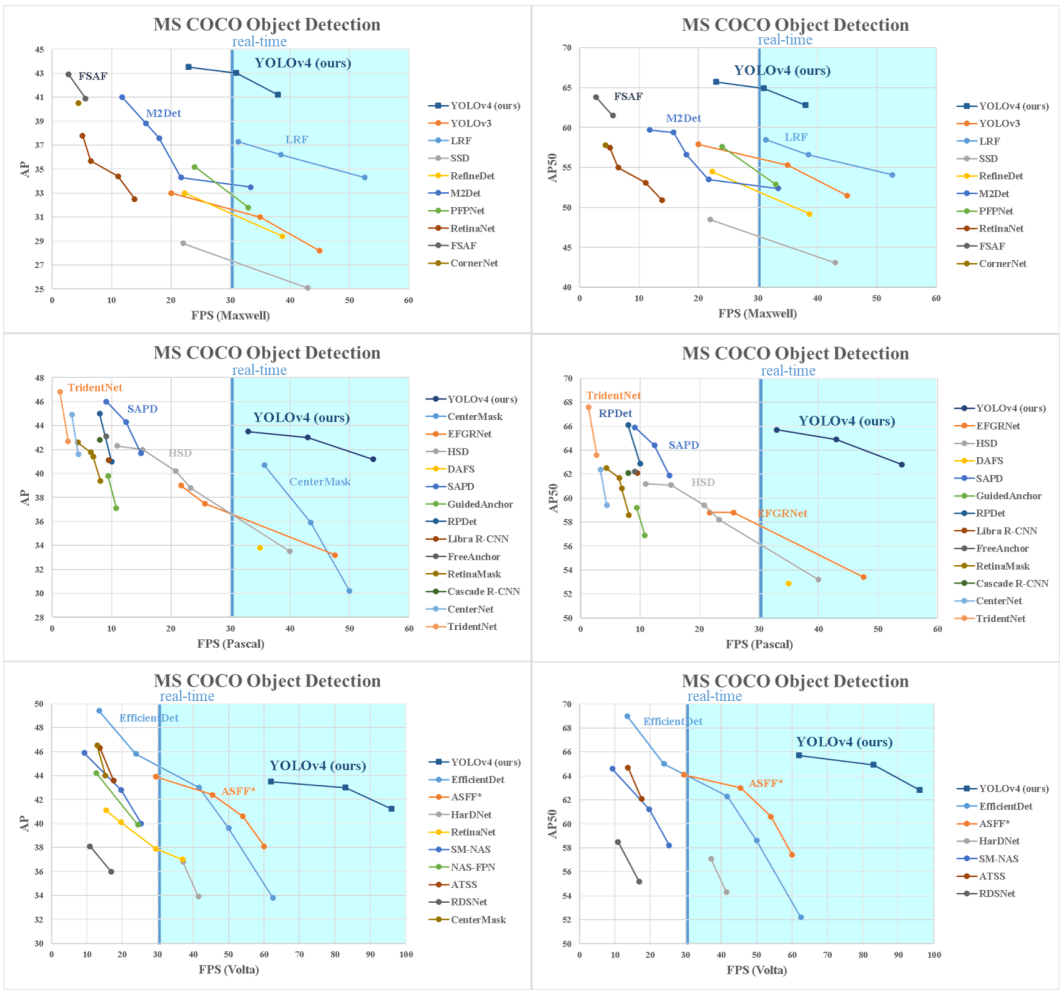


Figure 8: Comparison of the speed and accuracy of different object detectors. (Some articles stated the FPS of their detectors for only one of the GPUs: Maxwell/Pascal/Volta)

5. 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.

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.

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 8-16 GB-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.

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.

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%

References

[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 7036–7045, 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. Hit-Detector: 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 Arbeláez, 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 Dollár, 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 7132–7141, 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 4700–4708, 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] Günter 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 Dollár, 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 Dollár. 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 Dollár, 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 non-monotonic 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 779–788, 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 7263–7271, 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 Dollár, 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