# PP-YOLOE: An evolved version of YOLO

Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui,
Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, Baohua Lai
Baidu Inc.
{xushangliang, wangxinxin08, lvwenyu01, dengkaipeng, dangqingqing} @baidu.com

## Abstract

*In this report, we present PP-YOLOE, an industrial state-of-the-art object detector with high performance and friendly deployment. We optimize on the basis of the previous PP-YOLOv2, using anchor-free paradigm, more powerful backbone and neck equipped with CSPRepResStage, ET-head and dynamic label assignment algorithm TAL. We provide s/m/l/x models for different practice scenarios. As a result, PP-YOLOE-l achieves **51.4 mAP** on COCO test-dev and **78.1 FPS** on Tesla V100, yielding a remarkable improvement of (**+1.9 AP**, **+13.35% speed up**) and (**+1.3 AP**, **+24.96% speed up**), compared to the previous state-of-the-art industrial models PP-YOLOv2 and YOLOX respectively. Further, PP-YOLOE inference speed achieves **149.2 FPS** with TensorRT and FP16-precision. We also conduct extensive experiments to verify the effectiveness of our designs. Source code and pre-trained models are available at PaddleDetection*[1].

## 1. Introduction

One-stage object detector is popular in real-time applications due to excellent speed and accuracy trade-off. The most prominent architecture among one-stage detectors is the YOLO series[21, 22, 23, 2, 26, 14, 6, 18, 13]. Since YOLOv1[21], YOLO series object detectors have undergone tremendous changes in network structure, label assignment and so on. At present, YOLOX[6] achieves an optimal balance of speed and accuracy with 50.1 mAP at the speed of 68.9 FPS on Tesla V100.

YOLOX introduces advanced anchor-free method equipped with dynamic label assignment to improve the performance of detector, significantly outperforming YOLOv5[14] in terms of precision. Inspired by YOLOX, we further optimize our previous work PP-YOLOv2[13]. PP-YOLOv2 is a high-performance one-stage detector with 49.5 mAP at the speed of 68.9 FPS on Tesla V100. Based on PP-YOLOv2, we proposed an evolved version of YOLO named PP-YOLOE. PP-YOLOE avoids using operators like deformable convolution[3, 34] and Matrix NMS[28] to be well supported on various hardware. Moreover, PP-YOLOE can easily scale to a series of models for various hardware with different computing power. These characteristics further promote the application of PP-YOLOE in a wider range of practical scenarios.

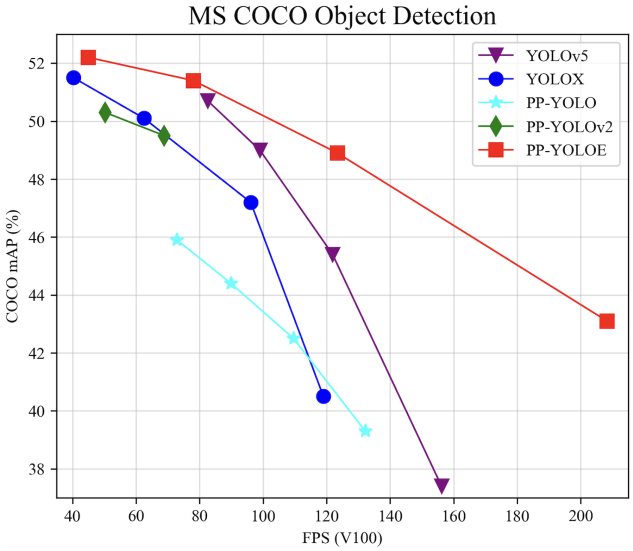



Figure 1: Comparison of the PP-YOLOE and other state-of-the-art models. PP-YOLOE-l achieves 51.4 mAP on COCO *test-dev* and 78.1 FPS on Tesla V100, obtains 1.9 AP and 9.2 FPS improvement compared with PP-YOLOv2[13].

As shown in Fig. 1, PP-YOLOE outperforms YOLOv5 and YOLOX in terms of speed and accuracy trade-off. Specifically, PP-YOLOE-l achieves 51.4 mAP on COCO with 640 $\times$ 640 resolution at the speed of 78.1 FPS, surpassing PP-YOLOv2 by 1.9% AP and YOLOX-l by 1.3% AP. Moreover, PP-YOLOE has a series of models, which can be simply configured through width multiplier and depth multiplier like YOLOv5. Our code has released on

[1] https://github.com/PaddlePaddle/PaddleDetection

PaddleDetection[1], with TensorRT and ONNX supported.

## 2. Method

In this section, we will first review our baseline model and then introduce the design of PP-YOLOE (Fig. 2) in detail from the aspects of network structure, label assignment strategy, head structure and loss function.

### 2.1. A Brief Review of PP-YOLOv2

The overall architecture of PP-YOLOv2 contains the backbone of ResNet50-vd[10] with deformable convolution[34], the neck of PAN with SPP layer and DropBlock[7] and the lightweight IoU aware head. In PP-YOLOv2, ReLU activation function is used in backbone while mish activation function is used in neck. Following YOLOv3, PP-YOLOv2 only assigns one anchor box for each ground truth object. In addition to classification loss, regression loss and objectness loss, PP-YOLOv2 also uses IoU loss and IoU aware loss to boost the performance. For more details, please refer to [13].

### 2.2. Improvement of PP-YOLOE

**Anchor-free.** As mentioned above, PP-YOLOv2[13] assigns ground truths in an anchor-based manner. However, anchor mechanism introduces a number of hyper-parameters and depends on hand-crafted design which may not generalize well on other datasets. For the above reason, we introduce anchor-free method in PP-YOLOv2. Following FCOS[25], which tiles one anchor point on each pixel, we set upper and lower bounds for three detection heads to assign ground truths to corresponding feature map. Then, the center of bounding box is calculated to select the closest pixel as positive samples. Following YOLO series, a 4D vector (x, y, w, h) is predicted for regression. This modification makes the model a little faster with the loss of 0.3 AP as shown in Table 2. Although upper and lower bounds are carefully set according to the anchor sizes of PP-YOLOv2, there are still some minor inconsistencies in the assignment results between anchor-based and anchor-free manner, which may lead to little precision drop.

**Backbone and Neck.** Residual connections[9, 29, 11] and dense connections[12, 15, 20] have been widely used in modern convolutional neural network. Residual connections introduce shortcut to relieve gradient vanishing problem and can be also regarded as a model ensemble approach. Dense connections aggregate intermediate features with diverse receptive fields, showing good performance on the object detection task. CSPNet[27] utilizes cross stage dense connections to lower computation burden without the loss of precision, which is popular among effective object detectors such as YOLOv5[14], YOLOX[6]. VoVNet[15] and subsequent TreeNet[20] also show superior performance in object detection and instance segmentation. Inspired by these works, we propose a novel RepResBlock by combining the residual connections and dense connections, which is used in our backbone and neck.

Originating from TreeBlock[20], our RepResBlock is shown in Fig. 3(b) during the training phase and Fig. 3(c) during the inference phase. Firstly, we simplify the original TreeBlock (Fig. 3(a)). Then, we replace the concatenation operation with element-wise add operation (Fig. 3(b)), because of the approximation of these two operations to some extent shown in RMNet [19]. Thus, during the inference phase, we can re-parameterizes RepResBlock to a basic residual block (Fig. 3(c)) used by ResNet-34 in a RepVGG[4] style.

We use proposed RepResBlock to build backbone and neck. Similar to ResNet, our backbone, named CSPRepResNet, contains one stem composed of three convolution layer and four subsequent stages stacked by our RepResBlock as shown in Fig. 3(d). In each stage, cross stage partial connections are used to avoid numerous parameters and computation burden brought by lots of $3 \times 3$ convolution layers. ESE (Effective Squeeze and Extraction) layer is also used to impose channel attention in each CSPRepResStage while building backbone. We build neck with proposed RepResBlock and CSPRepResStage following PP-YOLOv2[13]. Different from backbone, shortcut in RepResBlock and ESE layer in CSPRepResStage are removed in neck.

We use width multiplier $\alpha$ and depth multiplier $\beta$ to scale the basic backbone and neck jointly like YOLOv5[14]. Thus, we can get a series of detection network with different parameters and computation cost. The width setting of basic backbone is [64, 128, 256, 512, 1024]. Except for the stem, the depth setting of basic backbone is [3, 6, 6, 3]. The width setting and depth setting of basic neck are [192, 384, 768] and 3 respectively. Table 1 shows the specification of width multiplier $\alpha$ and depth multiplier $\beta$ for different model. Such modifications obtains 0.7% AP performance improvements – 49.5% AP as shown in Table 2.

| | width multiplier $\alpha$ | depth multiplier $\beta$ |
|---|---|---|
| s | 0.50 | 0.33 |
| m | 0.75 | 0.67 |
| l | 1.00 | 1.00 |
| x | 1.25 | 1.33 |

Table 1: Width multiplier $\alpha$ and depth multiplier $\beta$ specification for a series of networks

**Task Alignment Learning (TAL).** To further improve the

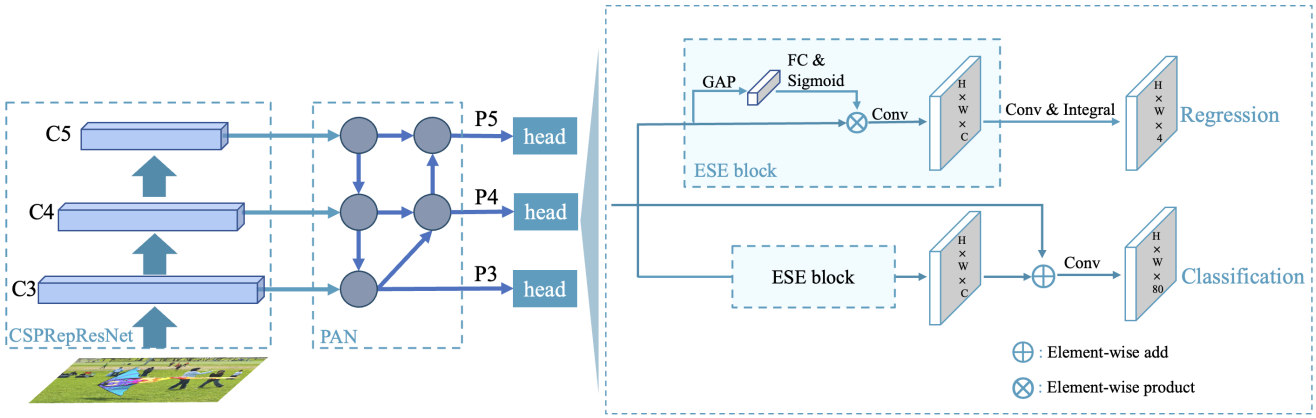
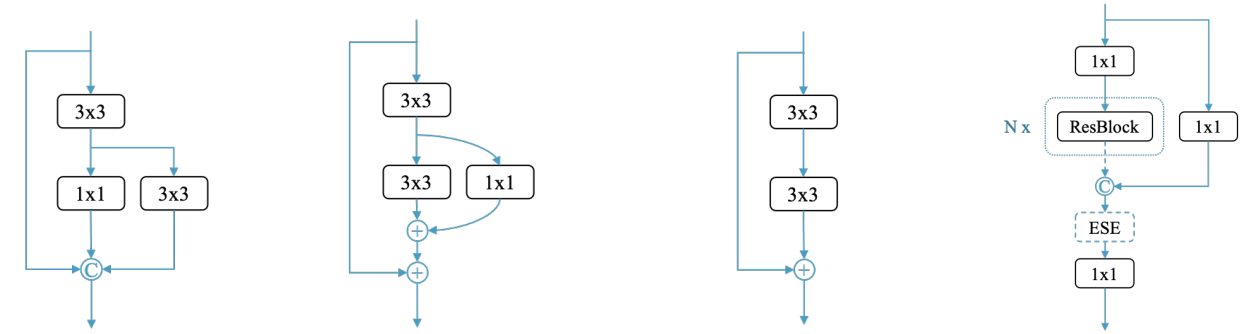



Figure 2: The model architecture of our PP-YOLOE. The backbone is CSPRepResNet, the neck is Path Aggregation Network (PAN), and the head is Efficient Task-aligned Head (ET-head).




(a) Simplified TreeBlock (b) Our RepResBlock during training (c) Our RepResBlock during inference (d) Our CSPRepResStage

Figure 3: Structure of our RepResBlock and CSPRepResStage

accuracy, label assignment is another aspect to be considered. YOLOX uses SimOTA as the label assignment strategy to improve performance. However, to further overcome the misalignment of classification and localization, task alignment learning (TAL) is proposed in TOOD[5], which is composed of a dynamic label assignment and task aligned loss. Dynamic label assignment means prediction/loss aware. According to the prediction, it allocate dynamic number of positive anchors for each ground-truth. By explicitly aligning the two tasks, TAL can obtain the highest classification score and the most precise bounding box at the same time. TAL可以同时获得最高的分类分数和最精确的边界框

For task aligned loss, TOOD use a normalized $t$, namely $\hat{t}$, to replace the target in loss. It adopts the largest IoU within each instance as the normalization. The *Binary Cross Entropy* (BCE) for the classification can be rewritten as: TOOD采用每个实例内最大的IoU作为规范化

$$L_{cls-pos} = \sum_{i=1}^{N_{pos}} BCE\left(p_i, \hat{t}_i\right) \quad (1)$$

We investigate the performance using different label assignment strategy. We conduct this experiment on above modified model, which use CSPRepResNet as backbone. For get the verification results quickly, we only train 36 epochs on COCO *train2017* and verify it on COCO *val*. As shown in Table 3, TAL achieves the best 45.2% AP performance. We use TAL to replace label assignment like FCOS style and achieve 0.9% AP improvement – 50.4% AP as shown in Table 2.

**Efficient Task-aligned Head (ET-head).** In object detection, the task conflict between classification and localization is a well-known problem. Corresponding solutions are proposed in many papers[5, 32, 16, 30]. YOLOX's decoupled head draws lessons from most of the one-stage and two-stage detectors, and successfully apply to YOLO model to improve accuracy. However, the decoupled head may make the classification and localization tasks separate and independent, and lack of task specific learning. Based on

| Model | mAP(%) | Parameters(M) | GFLOPs | Latency(ms) | FPS |
|---|---|---|---|---|---|
| PP-YOLOv2 baseline model | 49.1 | 54.58 | 115.77 | 14.5 | 68.9 |
| +Anchor-free | 48.8 (−0.3) | 54.27 | 114.78 | 14.3 | 69.8 |
| +CSPRepResNet | 49.5 (+0.7) | 47.42 | 101.87 | 11.7 | 85.5 |
| +TAL | 50.4 (+0.9) | 48.32 | 104.75 | 11.9 | 84.0 |
| +ET-head | **50.9** (+0.5) | 52.20 | 110.07 | 12.8 | 78.1 |

Table 2: Ablation study of PP-YOLOE-l on COCO *val*. We use 640×640 resolution as input with FP32-precision, and test on Tesla V100 without post-processing.

| Method | mAP(0.5:0.95) |
|---|---|
| ATSS[33] | 43.1 |
| SimOTA[6] | 44.3 |
| TAL[5] | **45.2** |

Table 3: Different label assignment on base model. We use CSPRepResStage as backbone and neck, one 1×1 conv layer as head, and only train 36 epochs on COCO *train2017*.

TOOD[5], we improve the head and propose ET-head with the goal of both speed and accuracy. As shown in Fig. 2, we use ESE to replace the layer attention in TOOD, simplify the alignment of classification branches to shortcut, and replace the alignment of regression branches with distribution focal loss (DFL) layer[16]. Through the above changes, the ET-head brings an increase of 0.9ms on V100.

For the learning of classification and location tasks, we choose varifocal loss (VFL) and distribution focal loss (DFL) respectively. PP-Picodet[31] successfully applys VFL and DFL in object detectors, and obtains performance improvement. For VFL in [32], different from the quality focal loss (QFL) in [16], VFL uses target score to weight the loss of positive samples. This implementation makes the contribution of positive samples with high IoU to loss relatively large. This also makes the model pay more attention to high-quality samples rather than those low-quality ones at training time. The same is that both use IoU-aware classification score (IACS) as the target to predict. This can effectively learn a joint representation of classification score and localization quality estimation, which enables high consistency between training and inference. For DFL, in order to solve the problem of inflexible representation of bounding box, [16] proposes to use general distribution to predict bounding box. Our model is supervised by the loss function:

$$Loss = \frac{\alpha \cdot loss_{VFL} + \beta \cdot loss_{GIoU} + \gamma \cdot loss_{DFL}}{\sum_{i}^{N_{pos}} \hat{t}} \quad (2)$$

where $\hat{t}$ denote the normalized target score, see Eq. (1). And as shown in Table 2, the ET-head obtains 0.5% AP improvement – 50.9% AP.

VFL使用目标评分来衡量正样本的损失权重，这种实现使得具有高IoU的正样本对损失的贡献相对较大。这也使得模型在训练时更注重高质量的样本，而不是那些低质量的样本。并且两者都用IoU感知的分类评分(IACS)作为预测的目标，这可以有效地得到classification score和localization quality estimation的联合表示，使训练和推理之间具有高度的一致性。

## 3. Experiment

In this section, we present the experiments details and results. All experiments are trained on MS COCO-2017 training set with 80 classes and 118k images. For ablation study, we use the standard COCO AP metric with single scale on MS COCO-2017 validation set with 5000 images. And we report final results using MS COCO-2017 *test-dev*.

### 3.1. Implementation details

We use stochastic gradient descent (SGD) with momentum = 0.9 and weight decay = 5e-4. We use cosine learning rate schedule, total epochs are 300, warmup epochs are 5, and base learning rate is 0.01. The total batch size is 64 on 8 × 32 G V100 GPU devices by default, and we follow linear scaling rule[8] to adjust learning rate. The exponential moving average (EMA) strategy with decay = 0.9998 is also adopted during training process. We only use some basic data augmentations, including random crop, random horizontal flip, color distortion, and multi-scale. Specially, input size is evenly drawn from 320 to 768 with 32 stride.

### 3.2. Comparsion with Other SOTA Detectors

Table 4 and Figure 1 show comparison of the results on MS-COCO test split with other state-of-the-art object detectors. We re-evaluate YOLOv5[14] and YOLOX[6] using official codebase because they have non-scheduled updates. We compare model inference speed with batch size = 1 (without data preprocess and non-maximum suppression). However, PP-YOLOE series using paddle inference engine. Further, for fair comparison, we also test the FP16 precision speed based on tensorRT 6.0 in the same environment. It should be emphasized that PaddlePaddle[2] officially supports tensorRT for model deployment. Therefore, PP-YOLOE can use paddle inference with tensorRT directly, and other tests follow the official guidelines.

[2] https://github.com/PaddlePaddle/Paddle

| Method | Backbone | Size | FPS (v100) | | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | w/o TRT | with TRT | | | | | | |
| YOLOv3 + ASFF* [17] | Darknet-53 | 320 | 60 | - | 38.1% | 57.4% | 42.1% | 16.1% | 41.6% | 53.6% |
| YOLOv3 + ASFF* [17] | Darknet-53 | 416 | 54 | - | 40.6% | 60.6% | 45.1% | 20.3% | 44.2% | 54.1% |
| YOLOv4 [2] | CSPDarknet-53 | 416 | 96 | - | 41.2% | 62.8% | 44.3% | 20.4% | 44.4% | 56.0% |
| YOLOv4 [2] | CSPDarknet-53 | 512 | 83 | - | 43.0% | 64.9% | 46.5% | 24.3% | 46.1% | 55.2% |
| YOLOv4-CSP [26] | Modified CSPDarknet-53 | 512 | 97 | - | 46.2% | 64.8% | 50.2% | 24.6% | 50.4% | 61.9% |
| YOLOv4-CSP [26] | Modified CSPDarknet-53 | 640 | 73 | - | 47.5% | 66.2% | 51.7% | 28.2% | 51.2% | 59.8% |
| EfficientDet-D0 [24] | Efficient-B0 | 512 | 98.0 | - | 33.8% | 52.2% | 35.8% | 12.0% | 38.3% | 51.2% |
| EfficientDet-D1 [24] | Efficient-B1 | 640 | 74.1 | - | 39.6% | 58.6% | 42.3% | 17.9% | 44.3% | 56.0% |
| EfficientDet-D2 [24] | Efficient-B2 | 768 | 56.5 | - | 43.0% | 62.3% | 46.2% | 22.5% | 47.0% | 58.4% |
| EfficientDet-D2 [24] | Efficient-B3 | 896 | 34.5 | - | 45.8% | 65.0% | 49.3% | 26.6% | 49.4% | 59.8% |
| PP-YOLO [18] | ResNet50-vd-dcn | 320 | $132.2^+$ | $242.2^+$ | 39.3% | 59.3% | 42.7% | 16.7% | 41.4% | 57.8% |
| PP-YOLO [18] | ResNet50-vd-dcn | 416 | $109.6^+$ | $215.4^+$ | 42.5% | 62.8% | 46.5% | 21.2% | 45.2% | 58.2% |
| PP-YOLO [18] | ResNet50-vd-dcn | 512 | $89.9^+$ | $188.4^+$ | 44.4% | 64.6% | 48.8% | 24.4% | 47.1% | 58.2% |
| PP-YOLO [18] | ResNet50-vd-dcn | 608 | $72.9^+$ | $155.6^+$ | 45.9% | 65.2% | 49.9% | 26.3% | 47.8% | 57.2% |
| PP-YOLOv2 [13] | ResNet50-vd-dcn | 320 | 123.3 | 152.9 | 43.1% | 61.7% | 46.5% | 19.7% | 46.3% | 61.8% |
| PP-YOLOv2 [13] | ResNet50-vd-dcn | 416 | $102^+$ | $145.1^+$ | 46.3% | 65.1% | 50.3% | 23.9% | 50.2% | 62.2% |
| PP-YOLOv2 [13] | ResNet50-vd-dcn | 512 | $93.4^+$ | $141.2^+$ | 48.2% | 67.1% | 52.7% | 27.7% | 52.1% | 62.1% |
| PP-YOLOv2 [13] | ResNet50-vd-dcn | 640 | $68.9^+$ | $106.5^+$ | 49.5% | 68.2% | 54.4% | 30.7% | 52.9% | 61.2% |
| PP-YOLOv2 [13] | ResNet101-vd-dcn | 640 | $50.3^+$ | $87.0^+$ | 50.3% | 69.0% | 55.3% | 31.6% | 53.9% | 62.4% |
| YOLOv5-s [14] | Modified CSP v6 | 640 | $156.2^+$ | $454.5^*$ | 37.4% | 56.8% | - | - | - | - |
| YOLOv5-m [14] | Modified CSP v6 | 640 | $121.9^+$ | $263.1^*$ | 45.4% | 64.1% | - | - | - | - |
| YOLOv5-l [14] | Modified CSP v6 | 640 | $99.0^+$ | $172.4^*$ | 49.0% | 67.3% | - | - | - | - |
| YOLOv5-x [14] | Modified CSP v6 | 640 | $82.6^+$ | $117.6^*$ | 50.7% | 68.9% | - | - | - | - |
| YOLOX-s [6] | Modified CSP v5 | 640 | $119.0^*$ \| $102.0^+$ | $246.9^*$ | 40.5% | - | - | - | - | - |
| YOLOX-m [6] | Modified CSP v5 | 640 | $96.1^*$ \| $81.3^+$ | $177.3^*$ | 47.2% | - | - | - | - | - |
| YOLOX-l [6] | Modified CSP v5 | 640 | $62.5^*$ \| $68.9^+$ | $120.1^*$ | 50.1% | - | - | - | - | - |
| YOLOX-x [6] | Modified CSP v5 | 640 | $40.3^*$ \| $57.8^+$ | $87.4^*$ | 51.5% | - | - | - | - | - |
| **PP-YOLOE-s** | CSPRepResNet | 640 | 208.3 | 333.3 | **43.1%** | **60.5%** | **46.6%** | **23.2%** | **46.4%** | **56.9%** |
| **PP-YOLOE-m** | CSPRepResNet | 640 | 123.4 | 208.3 | **48.9%** | **66.5%** | **53.0%** | **28.6%** | **52.9%** | **63.8%** |
| **PP-YOLOE-l** | CSPRepResNet | 640 | 78.1 | 149.2 | **51.4%** | **68.9%** | **55.6%** | **31.4%** | **55.3%** | **66.1%** |
| **PP-YOLOE-x** | CSPRepResNet | 640 | 45.0 | 95.2 | **52.2%** | **69.9%** | **56.5%** | **33.3%** | **56.3%** | **66.4%** |

Table 4: Comparison of the speed and accuracy of different object detectors on COCO 2017 *test-dev*. Results marked by "+" are updated results from the corresponding official release. Results marked by "*" are tested in our environment using official codebase and model. The input size of YOLOv5 is not exactly square of $640 \times 640$ in validation and speed test, so we skip it in the table. The default precision of speed is FP32 for *w/o trt* and FP16 for *with trt*. Moreover, we provide both FP32 and FP16 for YOLOX *w/o trt* scene, the FP32 speed on the left side of split line and FP16 speed on the right.

## 4. Conclusion

In this report, we present several updates to PP-YOLOv2, including scalable backbone-neck architecture, efficient task aligned head, advanced label assignment strategy and refined objective loss function, which forms a series high-performance object detectors called PP-YOLOE. Meanwhile, we present s/m/l/x models which can cover different scenarios in practice. Moreover, these models can smoothly transition to deployment, with PaddlePaddle official support. We hope these designs with encouraging results can provide inspirations for developers and researchers.

## References


[1] PaddlePaddle Authors. PaddleDetection, object detection and instance segmentation toolkit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleDetection, 2021. 2

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1, 5

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 1

[4] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021. 2

[5] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3510–3519, 2021. 3, 4

[6] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 1, 2, 4, 5

[7] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018. 2

[8] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. 4

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[10] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 2

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[12] 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*, pages 4700–4708, 2017. 2

[13] Xin Huang, Xinxin Wang, Wenyu Lv, Xiaying Bai, Xiang Long, Kaipeng Deng, Qingqing Dang, Shumin Han, Qiwen Liu, Xiaoguang Hu, Dianhai Yu, Yanjun Ma, and Osamu Yoshie. Pp-yolov2: A practical object detector, 2021. 1, 2, 5

[14] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, Feb. 2022. 1, 2, 4, 5

[15] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[16] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv preprint arXiv:2006.04388*, 2020. 3, 4

[17] Songtao Liu, Di Huang, and Yunhong Wang. Learning spatial fusion for single-shot object detection, 2019. 5

[18] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, and Shilei Wen. Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*, 2020. 1, 5

[19] Fanxu Meng, Hao Cheng, Jiaxin Zhuang, Ke Li, and Xing Sun. Rmnet: Equivalently removing residual connection from networks. *arXiv preprint arXiv:2111.00687*, 2021. 2

[20] Lu Rao. Treenet: A lightweight one-shot aggregation convolutional network. *arXiv preprint arXiv:2109.12342*, 2021. 2

[21] 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*, pages 779–788, 2016. 1

[22] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1

[24] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020. 5

[25] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 2

[26] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network, 2021. 1, 5

[27] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020. 2

[28] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020. 1

[29] 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*, pages 1492–1500, 2017. 2

[30] Yang Yang, Min Li, Bo Meng, Junxing Ren, Degang Sun, and Zihao Huang. Rethinking the aligned and misaligned features in one-stage object detection. 2021. 3

[31] Guanghua Yu, Qinyao Chang, Wenyu Lv, Chang Xu, Cheng Cui, Wei Ji, Qingqing Dang, Kaipeng Deng, Guanzhong Wang, Yuning Du, Baohua Lai, Qiwen Liu, Xiaoguang Hu, Dianhai Yu, and Yanjun Ma. Pp-picodet: A better real-time object detector on mobile devices. *CoRR*, abs/2111.00902, 2021. 4

[32] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8514–8523, 2021. 3, 4

[33] 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/CVF conference on computer vision and pattern recognition*, pages 9759–9768, 2020. 4

[34] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable. *Better Results*, 2018. 1, 2