YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

Chien-Yao Wang[1], Alexey Bochkovskiy, and Hong-Yuan Mark Liao[1]
[1]Institute of Information Science, Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw, alexeyab84@gmail.com, and liao@iis.sinica.edu.tw

Abstract

YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWIN-L Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutional-based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. Moreover, we train YOLOv7 only on MS COCO dataset from scratch without using any other datasets or pre-trained weights. Source code is released in https://github.com/WongKinYiu/yolov7.

1. Introduction

Real-time object detection is a very important topic in computer vision, as it is often a necessary component in computer vision systems. For example, multi-object tracking [94, 93], autonomous driving [40, 18], robotics [35, 58], medical image analysis [34, 46], etc. The computing devices that execute real-time object detection is usually some mobile CPU or GPU, as well as various neural processing units (NPU) developed by major manufacturers. For example, the Apple neural engine (Apple), the neural compute stick (Intel), Jetson AI edge devices (Nvidia), the edge TPU (Google), the neural processing engine (Qualcomm), the AI processing unit (MediaTek), and the AI SoCs (Kneron), are all NPUs. Some of the above mentioned edge devices focus on speeding up different operations such as vanilla convolution, depth-wise convolution, or MLP operations. In this paper, the real-time object detector we proposed mainly hopes that it can support both mobile GPU and GPU devices from the edge to the cloud.

In recent years, the real-time object detector is still developed for different edge device. For example, the development of MCUNet [49, 48] and NanoDet [54] focused on producing low-power single-chip and improving the inference speed on edge CPU. As for methods such as YOLOX [21] and YOLOR [81], they focus on improving the inference speed of various GPUs. More recently, the development of real-time object detector has focused on the design of efficient architecture. As for real-time object detectors that can be used on CPU [54, 88, 84, 83], their design is mostly based on MobileNet [28, 66, 27], ShuffleNet [92, 55], or GhostNet [25]. Another mainstream real-time object detectors are developed for GPU [81, 21, 97], they mostly use ResNet [26], DarkNet [63], or DLA [87], and then use the CSPNet [80] strategy to optimize the architecture. The development direction of the proposed methods in this paper are different from that of the current mainstream real-time object detectors. In addition to architecture optimization, our proposed methods will focus on the optimization of the training process. Our focus will be on some optimized modules and optimization methods which may strengthen the training cost for improving the accuracy of object detection, but without increasing the inference cost. We call the proposed modules and optimization methods trainable bag-of-freebies.

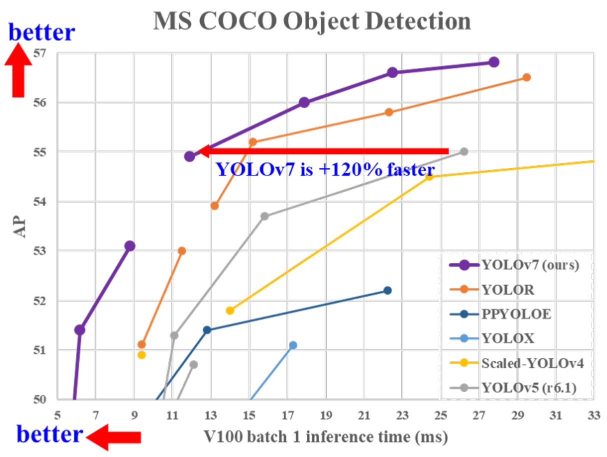


Figure 1: Comparison with other real-time object detectors, our proposed methods achieve state-of-the-arts performance.

Recently, model re-parameterization [13, 12, 29] and dynamic label assignment [20, 17, 42] have become important topics in network training and object detection. Mainly after the above new concepts are proposed, the training of object detector evolves many new issues. In this paper, we will present some of the new issues we have discovered and devise effective methods to address them. For model re-parameterization, we analyze the model re-parameterization strategies applicable to layers in different networks with the concept of gradient propagation path, and propose planned re-parameterized model. In addition, when we discover that with dynamic label assignment technology, the training of model with multiple output layers will generate new issues. That is: "How to assign dynamic targets for the outputs of different branches?" For this problem, we propose a new label assignment method called coarse-to-fine lead guided label assignment.

The contributions of this paper are summarized as follows: (1) we design several trainable bag-of-freebies methods, so that real-time object detection can greatly improve the detection accuracy without increasing the inference cost; (2) for the evolution of object detection methods, we found two new issues, namely how re-parameterized module replaces original module, and how dynamic label assignment strategy deals with assignment to different output layers. In addition, we also propose methods to address the difficulties arising from these issues; (3) we propose "extend" and "compound scaling" methods for the real-time object detector that can effectively utilize parameters and computation; and (4) the method we proposed can effectively reduce about 40% parameters and 50% computation of state-of-the-art real-time object detector, and has faster inference speed and higher detection accuracy.

2. Related work

2.1. Real-time object detectors

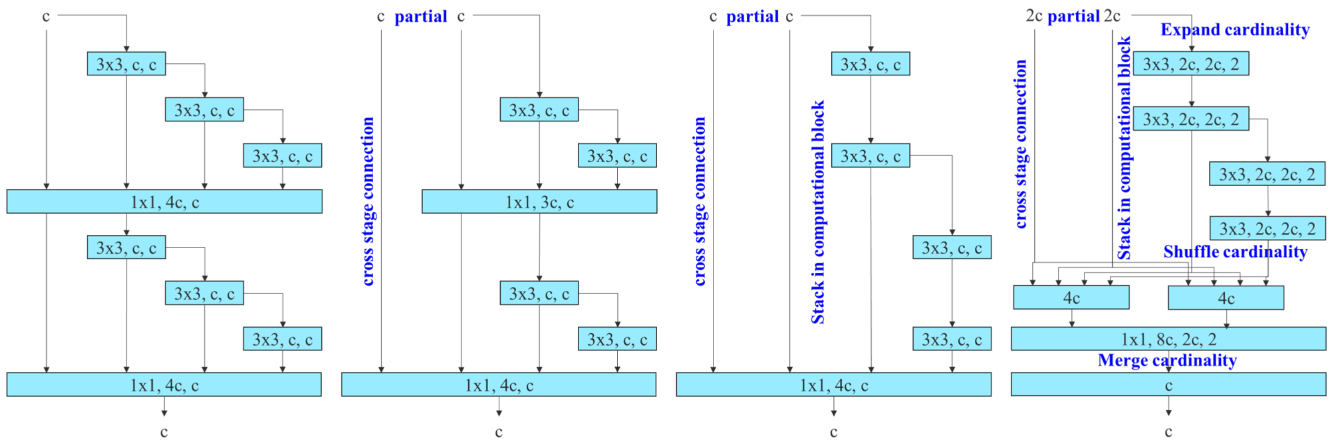
Currently state-of-the-art real-time object detectors are mainly based on YOLO [61, 62, 63] and FCOS [76, 77], which are [3, 79, 81, 21, 54, 85, 23]. Being able to become a state-of-the-art real-time object detector usually requires the following characteristics: (1) a faster and stronger network architecture; (2) a more effective feature integration method [22, 97, 37, 74, 59, 30, 9, 45]; (3) a more accurate detection method [76, 77, 69]; (4) a more robust loss function [96, 64, 6, 56, 95, 57]; (5) a more efficient label assignment method [99, 20, 17, 82, 42]; and (6) a more efficient training method. In this paper, we do not intend to explore self-supervised learning or knowledge distillation methods that require additional data or large model. Instead, we will design new trainable bag-of-freebies method for the issues derived from the state-of-the-art methods associated with (4), (5), and (6) mentioned above.

2.2. Model re-parameterization

Model re-parametrization techniques [71, 31, 75, 19, 33, 11, 4, 24, 13, 12, 10, 29, 14, 78] merge multiple computational modules into one at inference stage. The model re-parameterization technique can be regarded as an ensemble technique, and we can divide it into two categories, i.e., module-level ensemble and model-level ensemble. There are two common practices for model-level re-parameterization to obtain the final inference model. One is to train multiple identical models with different training data, and then average the weights of multiple trained models. The other is to perform a weighted average of the weights of models at different iteration number. Module-level re-parameterization is a more popular research issue recently. This type of method splits a module into multiple identical or different module branches during training and integrates multiple branched modules into a completely equivalent module during inference. However, not all proposed re-parameterized module can be perfectly applied to different architectures. With this in mind, we have developed new re-parameterization module and designed related application strategies for various architectures.

2.3. Model scaling

Model scaling [72, 60, 74, 73, 15, 16, 2, 51] is a way to scale up or down an already designed model and make it fit in different computing devices. The model scaling method usually uses different scaling factors, such as resolution (size of input image), depth (number of layer), width (number of channel), and stage (number of feature pyramid), so as to achieve a good trade-off for the amount of network parameters, computation, inference speed, and accuracy. Network architecture search (NAS) is one of the commonly used model scaling methods. NAS can automatically search for suitable scaling factors from search space without defining too complicated rules. The disadvantage of NAS is that it requires very expensive computation to complete the search for model scaling factors. In [15], the researcher analyzes the relationship between scaling factors and the amount of parameters and operations, trying to directly estimate some rules, and thereby obtain the scaling factors required by model scaling. Checking the literature, we found that almost all model scaling methods analyze individual scaling factor independently, and even the methods in the compound scaling category also optimized scaling factor independently. The reason for this is because most popular NAS architectures deal with scaling factors that are not very correlated. We observed that all concatenation-based models, such as DenseNet [32] or VoVNet [39], will change the input width of some layers when the depth of such models is scaled. Since the proposed architecture is concatenation-based, we have to design a new compound scaling method for this model.



(a) VoVNet [39] (b) CSPVoVNet [79] (c) ELAN [1] (d) E-ELAN

Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

3. Architecture

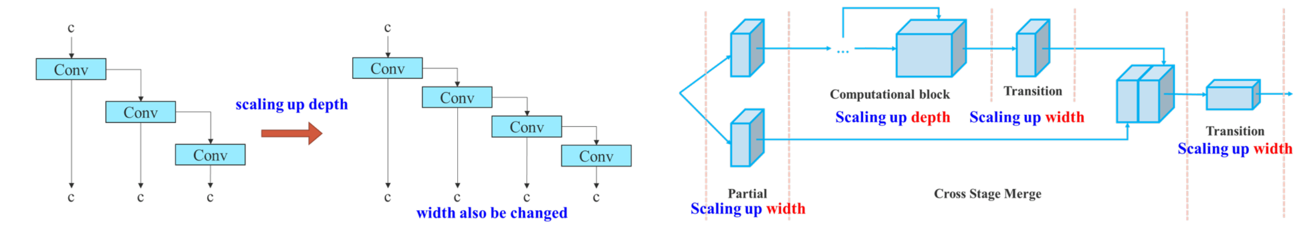
3.1. Extended efficient layer aggregation networks

In most of the literature on designing the efficient architectures, the main considerations are no more than the number of parameters, the amount of computation, and the computational density. Starting from the characteristics of memory access cost, Ma *et al.* [55] also analyzed the influence of the input/output channel ratio, the number of branches of the architecture, and the element-wise operation on the network inference speed. Dollár *et al.* [15] additionally considered activation when performing model scaling, that is, to put more consideration on the number of elements in the output tensors of convolutional layers. The design of CSPVoVNet [79] in Figure 2 (b) is a variation of VoVNet [39]. In addition to considering the aforementioned basic designing concerns, the architecture of CSPVoVNet [79] also analyzes the gradient path, in order to enable the weights of different layers to learn more diverse features. The gradient analysis approach described above makes inferences faster and more accurate. ELAN [1] in Figure 2 (c) considers the following design strategy – "How to design an efficient network?." They came out with a conclusion: By controlling the longest shortest gradient path, a deeper network can learn and converge effectively. In this paper, we propose Extended-ELAN (E-ELAN) based on ELAN and its main architecture is shown in Figure 2 (d).

Regardless of the gradient path length and the stacking number of computational blocks in large-scale ELAN, it has reached a stable state. If more computational blocks are stacked unlimitedly, this stable state may be destroyed, and the parameter utilization rate will decrease. The proposed E-ELAN uses expand, shuffle, merge cardinality to achieve the ability to continuously enhance the learning ability of the network without destroying the original gradient path. In terms of architecture, E-ELAN only changes the architecture in computational block, while the architecture of transition layer is completely unchanged. Our strategy is to use group convolution to expand the channel and cardinality of computational blocks. We will apply the same group parameter and channel multiplier to all the computational blocks of a computational layer. Then, the feature map calculated by each computational block will be shuffled into g groups according to the set group parameter g, and then concatenate them together. At this time, the number of channels in each group of feature map will be the same as the number of channels in the original architecture. Finally, we add g groups of feature maps to perform merge cardinality. In addition to maintaining the original ELAN design architecture, E-ELAN can also guide different groups of computational blocks to learn more diverse features.

3.2. Model scaling for concatenation-based models

The main purpose of model scaling is to adjust some attributes of the model and generate models of different scales to meet the needs of different inference speeds. For example the scaling model of EfficientNet [72] considers the width, depth, and resolution. As for the scaled-YOLOv4 [79], its scaling model is to adjust the number of stages. In [15], Dollár *et al.* analyzed the influence of vanilla convolution and group convolution on the amount of parameter and computation when performing width and depth scaling, and used this to design the corresponding model scaling method.



(a) concatenation-based model (b) scaled-up concatenation-based model (c) compound scaling up depth and width for concatenation-based model

Figure 3: Model scaling for concatenation-based models. From (a) to (b), we observe that when depth scaling is performed on concatenation-based models, the output width of a computational block also increases. This phenomenon will cause the input width of the subsequent transmission layer to increase. Therefore, we propose (c), that is, when performing model scaling on concatenation-based models, only the depth in a computational block needs to be scaled, and the remaining of transmission layer is performed with corresponding width scaling.

The above methods are mainly used in architectures such as PlainNet or ResNet. When these architectures are in executing scaling up or scaling down, the in-degree and out-degree of each layer will not change, so we can independently analyze the impact of each scaling factor on the amount of parameters and computation. However, if these methods are applied to the concatenation-based architecture, we will find that when scaling up or scaling down is performed on depth, the in-degree of a translation layer which is immediately after a concatenation-based computational block will decrease or increase, as shown in Figure 3 (a) and (b).

It can be inferred from the above phenomenon that we cannot analyze different scaling factors separately for a concatenation-based model but must be considered together. Take scaling-up depth as an example, such an action will cause a ratio change between the input channel and output channel of a transition layer, which may lead to a decrease in the hardware usage of the model. Therefore, we must propose the corresponding compound model scaling method for a concatenation-based model. When we scale the depth factor of a computational block, we must also calculate the change of the output channel of that block. Then, we will perform width factor scaling with the same amount of change on the transition layers, and the result is shown in Figure 3 (c). Our proposed compound scaling method can maintain the properties that the model had at the initial design and maintains the optimal structure.

4. Trainable bag-of-freebies

4.1. Planned re-parameterized convolution

Although RepConv [13] has achieved excellent performance on the VGG [68], when we directly apply it to ResNet [26] and DenseNet [32] and other architectures, its accuracy will be significantly reduced. We use gradient flow propagation paths to analyze how re-parameterized convolution should be combined with different network. We also designed planned re-parameterized convolution accordingly.

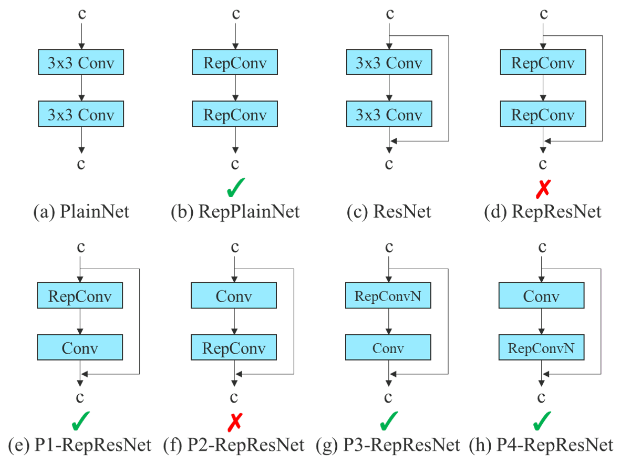


Figure 4: Planned re-parameterized model. In the proposed planned re-parameterized model, we found that a layer with residual or concatenation connections, its RepConv should not have identity connection. Under these circumstances, it can be replaced by RepConvN that contains no identity connections.

RepConv actually combines 3×3 convolution, 1×1 convolution, and identity connection in one convolutional layer. After analyzing the combination and corresponding performance of RepConv and different architectures, we find that the identity connection in RepConv destroys the residual in ResNet and the concatenation in DenseNet, which provides more diversity of gradients for different feature maps. For the above reasons, we use RepConv without identity connection (RepConvN) to design the architecture of planned re-parameterized convolution. In our thinking, when a convolutional layer with residual or concatenation is replaced by re-parameterized convolution, there should be no identity connection. Figure 4 shows an example of our designed "planned re-parameterized convolution" used in PlainNet and ResNet. As for the complete planned re-parameterized convolution experiment in residual-based model and concatenation-based model, it will be presented in the ablation study session.

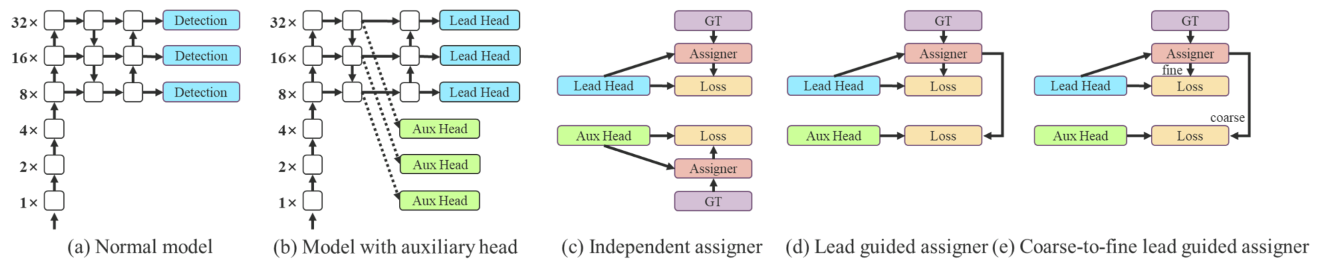


Figure 5: Coarse for auxiliary and fine for lead head label assigner. Compare with normal model (a), the schema in (b) has auxiliary head. Different from the usual independent label assigner (c), we propose (d) lead head guided label assigner and (e) coarse-to-fine lead head guided label assigner. The proposed label assigner is optimized by lead head prediction and the ground truth to get the labels of training lead head and auxiliary head at the same time. The detailed coarse-to-fine implementation method and constraint design details will be elaborated in Apendix.

4.2. Coarse for auxiliary and fine for lead loss

Deep supervision [38] is a technique that is often used in training deep networks. Its main concept is to add extra auxiliary head in the middle layers of the network, and the shallow network weights with assistant loss as the guide. Even for architectures such as ResNet [26] and DenseNet [32] which usually converge well, deep supervision [70, 98, 67, 47, 82, 65, 86, 50] can still significantly improve the performance of the model on many tasks. Figure 5 (a) and (b) show, respectively, the object detector architecture "without" and "with" deep supervision. In this paper, we call the head responsible for the final output as the lead head, and the head used to assist training is called auxiliary head.

Next we want to discuss the issue of label assignment. In the past, in the training of deep network, label assignment usually refers directly to the ground truth and generate hard label according to the given rules. However, in recent years, if we take object detection as an example, researchers often use the quality and distribution of prediction output by the network, and then consider together with the ground truth to use some calculation and optimization methods to generate a reliable soft label [61, 8, 36, 99, 91, 44, 43, 90, 20, 17, 42]. For example, YOLO [61] use IoU of prediction of bounding box regression and ground truth as the soft label of objectness. In this paper, we call the mechanism that considers the network prediction results together with the ground truth and then assigns soft labels as "label assigner."

Deep supervision needs to be trained on the target objectives regardless of the circumstances of auxiliary head or lead head. During the development of soft label assigner related techniques, we accidentally discovered a new derivative issue, i.e., "How to assign soft label to auxiliary head and lead head ?" To the best of our knowledge, the relevant literature has not explored this issue so far. The results of the most popular method at present is as shown in Figure 5 (c), which is to separate auxiliary head and lead head, and then use their own prediction results and the ground truth to execute label assignment. The method proposed in this paper is a new label assignment method that guides both auxiliary head and lead head by the lead head prediction. In other words, we use lead head prediction as guidance to generate coarse-to-fine hierarchical labels, which are used for auxiliary head and lead head learning, respectively. The two proposed deep supervision label assignment strategies are shown in Figure 5 (d) and (e), respectively.

Lead head guided label assigner is mainly calculated based on the prediction result of the lead head and the ground truth, and generate soft label through the optimization process. This set of soft labels will be used as the target training model for both auxiliary head and lead head. The reason to do this is because lead head has a relatively strong learning capability, so the soft label generated from it should be more representative of the distribution and correlation between the source data and the target. Furthermore, we can view such learning as a kind of generalized residual learning. By letting the shallower auxiliary head directly learn the information that lead head has learned, lead head will be more able to focus on learning residual information that has not yet been learned.

Coarse-to-fine lead head guided label assigner also used the predicted result of the lead head and the ground truth to generate soft label. However, in the process we generate two different sets of soft label, i.e., coarse label and fine label, where fine label is the same as the soft label generated by lead head guided label assigner, and coarse label is generated by allowing more grids to be treated as positive target by relaxing the constraints of the positive sample assignment process. The reason for this is that the learning ability of an auxiliary head is not as strong as that of a lead head, and in order to avoid losing the information that needs to be learned, we will focus on optimizing the recall of auxiliary head in the object detection task. As for the output of lead head, we can filter the high precision results from the high recall results as the final output. However, we must note that if the additional weight of coarse label is close to

Table 1: Comparison of baseline object detectors.

Model	#Param.	FLOPs	Size	AP^{val}	AP_{50}^{val}	AP_{75}^{val}	AP_{S}^{val}	AP_{M}^{val}	AP_{L}^{val}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

that of fine label, it may produce bad prior at final prediction. Therefore, in order to make those extra coarse positive grids have less impact, we put restrictions in the decoder, so that the extra coarse positive grids cannot produce soft label perfectly. The mechanism mentioned above allows the importance of fine label and coarse label to be dynamically adjusted during the learning process, and makes the optimizable upper bound of fine label always higher than coarse label.

4.3. Other trainable bag-of-freebies

In this section we will list some trainable bag-of-freebies. These freebies are some of the tricks we used in training, but the original concepts were not proposed by us. The training details of these freebies will be elaborated in the Appendix, including (1) Batch normalization in conv-bn-activation topology: This part mainly connects batch normalization layer directly to convolutional layer. The purpose of this is to integrate the mean and variance of batch normalization into the bias and weight of convolutional layer at the inference stage. (2) Implicit knowledge in YOLOR [81] combined with convolution feature map in addition and multiplication manner: Implicit knowledge in YOLOR can be simplified to a vector by pre-computing at the inference stage. This vector can be combined with the bias and weight of the previous or subsequent convolutional layer. (3) EMA model: EMA is a technique used in mean teacher [75], and in our system we use EMA model purely as the final inference model.

5. Experiments

5.1. Experimental setup

We use Microsoft COCO dataset to conduct experiments and validate our object detection method. All our experiments did not use pre-trained models. That is, all models were trained from scratch. During the development process, we used train 2017 set for training, and then used val 2017 set for verification and choosing hyperparameters. Finally, we show the performance of object detection on the test 2017 set and compare it with the state-of-the-art object detection algorithms. Detailed training parameter settings are described in Appendix.

We designed basic model for edge GPU, normal GPU, and cloud GPU, and they are respectively called YOLOv7-tiny, YOLOv7, and YOLOv7-W6. At the same time, we also use basic model for model scaling for different service requirements and get different types of models. For YOLOv7, we do stack scaling on neck, and use the proposed compound scaling method to perform scaling-up of the depth and width of the entire model, and use this to obtain YOLOv7-X. As for YOLOv7-W6, we use the newly proposed compound scaling method to obtain YOLOv7-E6 and YOLOv7-D6. In addition, we use the proposed E-ELAN for YOLOv7-E6, and thereby complete YOLOv7-E6E. Since YOLOv7-tiny is an edge GPU-oriented architecture, it will use leaky ReLU as activation function. As for other models we use SiLU as activation function. We will describe the scaling factor of each model in detail in Appendix.

Table 2: Comparison of state-of-the-art real-time object detectors.

Model	#Param.	FLOPs	Size	FPS	AP^{test} / AP^{val}	AP^{test}_{50}	AP^{test}_{75}	AP^{test}_{S}	AP^{test}_{M}	AP^{test}_{L}
YOLOX-S [21]	9.0M	26.8G	640	102	40.5% / 40.5%	-	-	-	-	-
YOLOX-M [21]	25.3M	73.8G	640	81	47.2% / 46.9%	-	-	-	-	-
YOLOX-L [21]	54.2M	155.6G	640	69	50.1% / 49.7%	-	-	-	-	-
YOLOX-X [21]	99.1M	281.9G	640	58	51.5% / 51.1%	-	-	-	-	-
PPYOLOE-S [85]	7.9M	17.4G	640	208	43.1% / 42.7%	60.5%	46.6%	23.2%	46.4%	56.9%
PPYOLOE-M [85]	23.4M	49.9G	640	123	48.9% / 48.6%	66.5%	53.0%	28.6%	52.9%	63.8%
PPYOLOE-L [85]	52.2M	110.1G	640	78	51.4% / 50.9%	68.9%	55.6%	31.4%	55.3%	66.1%
PPYOLOE-X [85]	98.4M	206.6G	640	45	52.2% / 51.9%	69.9%	56.5%	33.3%	56.3%	66.4%
YOLOv5-N (r6.1) [23]	1.9M	4.5G	640	159	- / 28.0%	-	-	-	-	-
YOLOv5-S (r6.1) [23]	7.2M	16.5G	640	156	- / 37.4%	-	-	-	-	-
YOLOv5-M (r6.1) [23]	21.2M	49.0G	640	122	- / 45.4%	-	-	-	-	-
YOLOv5-L (r6.1) [23]	46.5M	109.1G	640	99	- / 49.0%	-	-	-	-	-
YOLOv5-X (r6.1) [23]	86.7M	205.7G	640	83	- / 50.7%	-	-	-	-	-
YOLOR-CSP [81]	52.9M	120.4G	640	106	51.1% / 50.8%	69.6%	55.7%	31.7%	55.3%	64.7%
YOLOR-CSP-X [81]	96.9M	226.8G	640	87	53.0% / 52.7%	71.4%	57.9%	33.7%	57.1%	66.8%
YOLOv7-tiny-SiLU	6.2M	13.8G	640	286	38.7% / 38.7%	56.7%	41.7%	18.8%	42.4%	51.9%
YOLOv7	36.9M	104.7G	640	161	51.4% / 51.2%	69.7%	55.9%	31.8%	55.5%	65.0%
YOLOv7-X	71.3M	189.9G	640	114	53.1% / 52.9%	71.2%	57.8%	33.8%	57.1%	67.4%
YOLOv5-N6 (r6.1) [23]	3.2M	18.4G	1280	123	- / 36.0%	-	-	-	-	-
YOLOv5-S6 (r6.1) [23]	12.6M	67.2G	1280	122	- / 44.8%	-	-	-	-	-
YOLOv5-M6 (r6.1) [23]	35.7M	200.0G	1280	90	- / 51.3%	-	-	-	-	-
YOLOv5-L6 (r6.1) [23]	76.8M	445.6G	1280	63	- / 53.7%	-	-	-	-	-
YOLOv5-X6 (r6.1) [23]	140.7M	839.2G	1280	38	- / 55.0%	-	-	-	-	-
YOLOR-P6 [81]	37.2M	325.6G	1280	76	53.9% / 53.5%	71.4%	58.9%	36.1%	57.7%	65.6%
YOLOR-W6 [81]	79.8G	453.2G	1280	66	55.2% / 54.8%	72.7%	60.5%	37.7%	59.1%	67.1%
YOLOR-E6 [81]	115.8M	683.2G	1280	45	55.8% / 55.7%	73.4%	61.1%	38.4%	59.7%	67.7%
YOLOR-D6 [81]	151.7M	935.6G	1280	34	56.5% / 56.1%	74.1%	61.9%	38.9%	60.4%	68.7%
YOLOv7-W6	70.4M	360.0G	1280	84	54.9% / 54.6%	72.6%	60.1%	37.3%	58.7%	67.1%
YOLOv7-E6	97.2M	515.2G	1280	56	56.0% / 55.9%	73.5%	61.2%	38.0%	59.9%	68.4%
YOLOv7-D6	154.7M	806.8G	1280	44	56.6% / 56.3%	74.0%	61.8%	38.8%	60.1%	69.5%
YOLOv7-E6E	151.7M	843.2G	1280	36	56.8% / 56.8%	74.4%	62.1%	39.3%	60.5%	69.0%

[1] Our FLOPs is calaculated by rectangle input resolution like 640 $\times$ 640 or 1280 $\times$ 1280.
[2] Our inference time is estimated by using letterbox resize input image to make its long side equals to 640 or 1280.

5.2. Baselines

We choose previous version of YOLO [3, 79] and state-of-the-art object detector YOLOR [81] as our baselines. Table 1 shows the comparison of our proposed YOLOv7 models and those baseline that are trained with the same settings.

From the results we see that if compared with YOLOv4, YOLOv7 has 75% less parameters, 36% less computation, and brings 1.5% higher AP. If compared with state-of-the-art YOLOR-CSP, YOLOv7 has 43% fewer parameters, 15% less computation, and 0.4% higher AP. In the performance of tiny model, compared with YOLOv4-tiny-31, YOLOv7-tiny reduces the number of parameters by 39% and the amount of computation by 49%, but maintains the same AP. On the cloud GPU model, our model can still have a higher AP while reducing the number of parameters by 19% and the amount of computation by 33%.

5.3. Comparison with state-of-the-arts

We compare the proposed method with state-of-the-art object detectors for general GPUs and Mobile GPUs, and the results are shown in Table 2. From the results in Table 2 we know that the proposed method has the best speed-accuracy trade-off comprehensively. If we compare YOLOv7-tiny-SiLU with YOLOv5-N (r6.1), our method is 127 fps faster and 10.7% more accurate on AP. In addition, YOLOv7 has 51.4% AP at frame rate of 161 fps, while PPYOLOE-L with the same AP has only 78 fps frame rate. In terms of parameter usage, YOLOv7 is 41% less than PPYOLOE-L. If we compare YOLOv7-X with 114 fps inference speed to YOLOv5-L (r6.1) with 99 fps inference speed, YOLOv7-X can improve AP by 3.9%. If YOLOv7-X is compared with YOLOv5-X (r6.1) of similar scale, the inference speed of YOLOv7-X is 31 fps faster. In addition, in terms of the amount of parameters and computation, YOLOv7-X reduces 22% of parameters and 8% of computation compared to YOLOv5-X (r6.1), but improves AP by 2.2%.

If we compare YOLOv7 with YOLOR using the input resolution 1280, the inference speed of YOLOv7-W6 is 8 fps faster than that of YOLOR-P6, and the detection rate is also increased by 1% AP. As for the comparison between YOLOv7-E6 and YOLOv5-X6 (r6.1), the former has 0.9% AP gain than the latter, 45% less parameters and 63% less computation, and the inference speed is increased by 47%. YOLOv7-D6 has close inference speed to YOLOR-E6, but improves AP by 0.8%. YOLOv7-E6E has close inference speed to YOLOR-D6, but improves AP by 0.3%.

5.4. Ablation study

5.4.1 Proposed compound scaling method

Table 3 shows the results obtained when using different model scaling strategies for scaling up. Among them, our proposed compound scaling method is to scale up the depth of computational block by 1.5 times and the width of transition block by 1.25 times. If our method is compared with the method that only scaled up the width, our method can improve the AP by 0.5% with less parameters and amount of computation. If our method is compared with the method that only scales up the depth, our method only needs to increase the number of parameters by 2.9% and the amount of computation by 1.2%, which can improve the AP by 0.2%. It can be seen from the results of Table 3 that our proposed compound scaling strategy can utilize parameters and computation more efficiently.

Table 3: Ablation study on proposed model scaling.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}
base (v7-X light)	47.0M	125.5G	640	51.7%	70.1%	56.0%
width only (1.25 w)	73.4M	195.5G	640	52.4%	70.9%	57.1%
depth only (2.0 d)	69.3M	187.6G	640	52.7%	70.8%	57.3%
compound (v7-X)	71.3M	189.9G	640	**52.9%**	**71.1%**	**57.5%**
improvement	-	-	-	+1.2	+1.0	+1.5

5.4.2 Proposed planned re-parameterized model

In order to verify the generality of our proposed planed re-parameterized model, we use it on concatenation-based model and residual-based model respectively for verification. The concatenation-based model and residual-based model we chose for verification are 3-stacked ELAN and CSPDarknet, respectively.

In the experiment of concatenation-based model, we replace the 3×3 convolutional layers in different positions in 3-stacked ELAN with RepConv, and the detailed configuration is shown in Figure 6. From the results shown in Table 4 we see that all higher AP values are present on our proposed planned re-parameterized model.

In the experiment dealing with residual-based model, since the original dark block does not have a 3×3 convolution block that conforms to our design strategy, we additionally design a reversed dark block for the experiment, whose architecture is shown in Figure 7. Since the CSPDarknet with dark block and reversed dark block has exactly the same amount of parameters and operations, it is fair to compare. The experiment results illustrated in Table 5 fully confirm that the proposed planned re-parameterized model is equally effective on residual-based model. We find that the design of RepCSPResNet [85] also fit our design pattern.

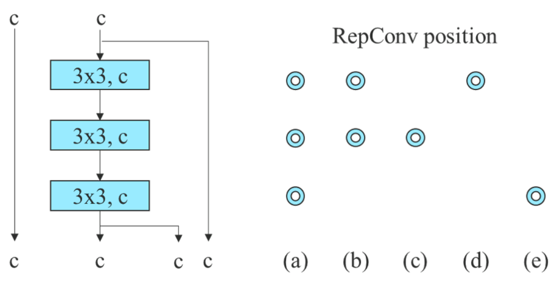


Figure 6: Planned RepConv 3-stacked ELAN. Blue circles are the position we replace Conv by RepConv.

Table 4: Ablation study on planned RepConcatenation model.

Model	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
base (3-S ELAN)	52.26%	70.41%	56.77%	35.81%	57.00%	67.59%
Figure 6 (a)	52.18%	70.34%	56.90%	35.71%	56.83%	67.51%
Figure 6 (b)	52.30%	70.30%	**56.92%**	35.76%	56.95%	67.74%
Figure 6 (c)	**52.33%**	**70.56%**	56.91%	**35.90%**	**57.06%**	67.50%
Figure 6 (d)	52.17%	70.32%	56.82%	35.33%	**57.06%**	**68.09%**
Figure 6 (e)	52.23%	70.20%	56.81%	35.34%	56.97%	66.88%

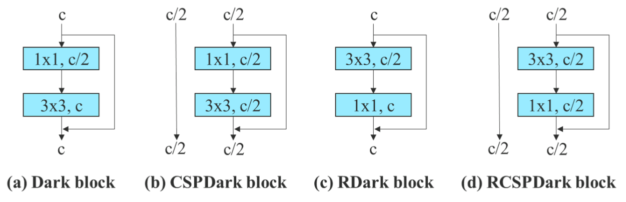


Figure 7: Reversed CSPDarknet. We reverse the position of 1×1 and 3×3 convolutional layer in dark block to fit our planned re-parameterized model design strategy.

Table 5: Ablation study on planned RepResidual model.

Model	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
base (YOLOR-W6)	54.82%	72.39%	59.95%	39.68%	59.38%	**68.30%**
RepCSP	54.67%	72.50%	59.58%	40.22%	**59.61%**	67.87%
RCSP	54.36%	71.95%	59.54%	40.15%	59.02%	67.44%
RepRCSP	**54.85%**	**72.51%**	**60.08%**	**40.53%**	59.52%	68.06%
base (YOLOR-CSP)	50.81%	69.47%	55.28%	33.74%	**56.01%**	65.38%
RepRCSP	**50.91%**	**69.54%**	**55.55%**	**34.44%**	55.74%	**65.46%**

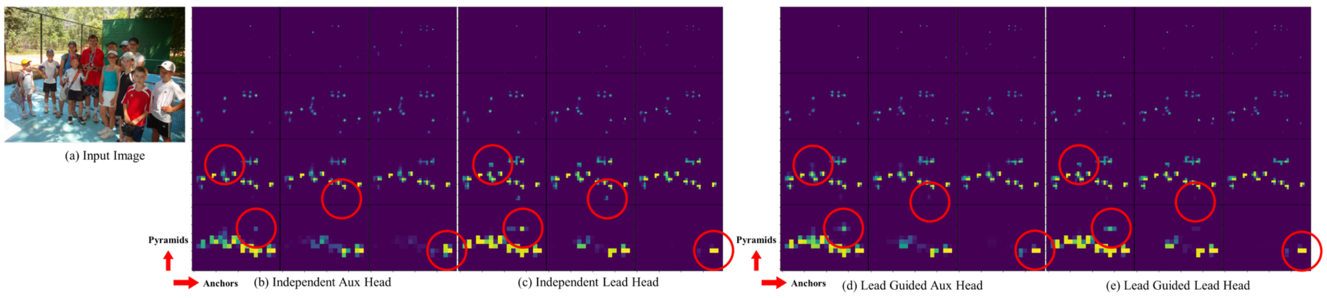


Figure 8: Objectness map predicted by different methods at auxiliary head and lead head.

5.4.3 Proposed assistant loss for auxiliary head

In the assistant loss for auxiliary head experiments, we compare the general independent label assignment for lead head and auxiliary head methods, and we also compare the two proposed lead guided label assignment methods. We show all comparison results in Table 6. From the results listed in Table 6, it is clear that any model that increases assistant loss can significantly improve the overall performance. In addition, our proposed lead guided label assignment strategy receives better performance than the general independent label assignment strategy in AP, AP_{50}, and AP_{75}. As for our proposed coarse for assistant and fine for lead label assignment strategy, it results in best results in all cases. In Figure 8 we show the objectness map predicted by different methods at auxiliary head and lead head. From Figure 8 we find that if auxiliary head learns lead guided soft label, it will indeed help lead head to extract the residual information from the consistant targets.

Table 6: Ablation study on proposed auxiliary head.

Model	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}
base (v7-E6)	1280	55.6%	73.2%	60.7%
independent	1280	55.8%	73.4%	60.9%
lead guided	1280	55.9%	73.5%	61.0%
coarse-to-fine lead guided	1280	**55.9%**	**73.5%**	**61.1%**
improvement	-	+0.3	+0.3	+0.4

In Table 7 we further analyze the effect of the proposed coarse-to-fine lead guided label assignment method on the decoder of auxiliary head. That is, we compared the results of with/without the introduction of upper bound constraint. Judging from the numbers in the Table, the method of constraining the upper bound of objectness by the distance from the center of the object can achieve better performance.

Table 7: Ablation study on constrained auxiliary head.

Model	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}
base (v7-E6)	1280	55.6%	73.2%	60.7%
aux without constraint	1280	55.9%	73.5%	61.0%
aux with constraint	1280	**55.9%**	**73.5%**	**61.1%**
improvement	-	+0.3	+0.3	+0.4

Since the proposed YOLOv7 uses multiple pyramids to jointly predict object detection results, we can directly connect auxiliary head to the pyramid in the middle layer for training. This type of training can make up for information that may be lost in the next level pyramid prediction. For the above reasons, we designed partial auxiliary head in the proposed E-ELAN architecture. Our approach is to connect auxiliary head after one of the sets of feature map before merging cardinality, and this connection can make the weight of the newly generated set of feature map not directly updated by assistant loss. Our design allows each pyramid of lead head to still get information from objects with different sizes. Table 8 shows the results obtained using two different methods, i.e., coarse-to-fine lead guided and partial coarse-to-fine lead guided methods. Obviously, the partial coarse-to-fine lead guided method has a better auxiliary effect.

Table 8: Ablation study on partial auxiliary head.

Model	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}
base (v7-E6E)	1280	56.3%	74.0%	61.5%
aux	1280	56.5%	74.0%	61.6%
partial aux	1280	**56.8%**	**74.4%**	**62.1%**
improvement	-	+0.5	+0.4	+0.6

6. Conclusions

In this paper we propose a new architecture of real-time object detector and the corresponding model scaling method. Furthermore, we find that the evolving process of object detection methods generates new research topics. During the research process, we found the replacement problem of re-parameterized module and the allocation problem of dynamic label assignment. To solve the problem, we propose the trainable bag-of-freebies method to enhance the accuracy of object detection. Based on the above, we have developed the YOLOv7 series of object detection systems, which receives the state-of-the-art results.

7. Acknowledgements

The authors wish to thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

Table 9: More comparison (batch=1, no-TRT, without extra object detection training data)

Model	**#Param.**	**FLOPs**	**Size**	**FPS**^{V100}	**AP**^{test} / **AP**^{val}	**AP**$_{50}^{test}$	**AP**$_{75}^{test}$
YOLOv7-tiny-SiLU	6.2M	13.8G	640	286	**38.7% / 38.7%**	**56.7%**	**41.7%**
PPYOLOE-S [85]	7.9M	17.4G	640	208	**43.1% / 42.7%**	**60.5%**	**46.6%**
YOLOv7	36.9M	104.7G	640	161	**51.4% / 51.2%**	**69.7%**	**55.9%**
YOLOv5-N (r6.1) [23]	1.9M	4.5G	640	159	- / 28.0%	-	-
YOLOv5-S (r6.1) [23]	7.2M	16.5G	640	156	- / 37.4%	-	-
PPYOLOE-M [85]	23.4M	49.9G	640	123	48.9% / 48.6%	66.5%	53.0%
YOLOv5-N6 (r6.1) [23]	3.2M	18.4G	1280	123	- / 36.0%	-	-
YOLOv5-S6 (r6.1) [23]	12.6M	67.2G	1280	122	- / 44.8%	-	-
YOLOv5-M (r6.1) [23]	21.2M	49.0G	640	122	- / 45.4%	-	-
YOLOv7-X	71.3M	189.9G	640	114	**53.1% / 52.9%**	**71.2%**	**57.8%**
YOLOR-CSP [81]	52.9M	120.4G	640	106	51.1% / 50.8%	69.6%	55.7%
YOLOX-S [21]	9.0M	26.8G	640	102	40.5% / 40.5%	-	-
YOLOv5-L (r6.1) [23]	46.5M	109.1G	640	99	- / 49.0%	-	-
YOLOv5-M6 (r6.1) [23]	35.7M	200.0G	1280	90	- / 51.3%	-	-
YOLOR-CSP-X [81]	96.9M	226.8G	640	87	53.0% / 52.7%	**71.4%**	**57.9%**
YOLOv7-W6	70.4M	360.0G	1280	84	**54.9% / 54.6%**	**72.6%**	**60.1%**
YOLOv5-X (r6.1) [23]	86.7M	205.7G	640	83	- / 50.7%	-	-
YOLOX-M [21]	25.3M	73.8G	640	81	47.2% / 46.9%	-	-
PPYOLOE-L [85]	52.2M	110.1G	640	78	51.4% / 50.9%	68.9%	55.6%
YOLOR-P6 [81]	37.2M	325.6G	1280	76	53.9% / 53.5%	71.4%	58.9%
YOLOX-L [21]	54.2M	155.6G	640	69	50.1% / 49.7%	-	-
YOLOR-W6 [81]	79.8G	453.2G	1280	66	**55.2% / 54.8%**	**72.7%**	**60.5%**
YOLOv5-L6 (r6.1) [23]	76.8M	445.6G	1280	63	- / 53.7%	-	-
YOLOX-X [21]	99.1M	281.9G	640	58	51.5% / 51.1%	-	-
YOLOv7-E6	97.2M	515.2G	1280	56	**56.0% / 55.9%**	**73.5%**	**61.2%**
YOLOR-E6 [81]	115.8M	683.2G	1280	45	55.8% / 55.7%	73.4%	61.1%
PPYOLOE-X [85]	98.4M	206.6G	640	45	52.2% / 51.9%	69.9%	56.5%
YOLOv7-D6	154.7M	806.8G	1280	44	**56.6% / 56.3%**	**74.0%**	**61.8%**
YOLOv5-X6 (r6.1) [23]	140.7M	839.2G	1280	38	- / 55.0%	-	-
YOLOv7-E6E	151.7M	843.2G	1280	36	**56.8% / 56.8%**	**74.4%**	**62.1%**
YOLOR-D6 [81]	151.7M	935.6G	1280	34	56.5% / 56.1%	**74.1%**	**61.9%**
F-RCNN-R101-FPN+ [5]	60.0M	246.0G	1333	20	- / 44.0%	-	-
Deformable DETR [100]	40.0M	173.0G	-	19	- / 46.2%	-	-
Swin-B (C-M-RCNN) [52]	145.0M	982.0G	1333	11.6	- / 51.9%	-	-
DETR DC5-R101 [5]	60.0M	253.0G	1333	10	- / 44.9%	-	-
EfficientDet-D7x [74]	77.0M	410.0G	1536	6.5	55.1% / 54.4%	72.4%	58.4%
Dual-Swin-T (C-M-RCNN) [47]	113.8M	836.0G	1333	6.5	- / 53.6%	-	-
ViT-Adapter-B [7]	122.0M	997.0G	-	4.4	- / 50.8%	-	-
Dual-Swin-B (HTC) [47]	235.0M	-	1600	2.5	**58.7% / 58.4%**	-	-
Dual-Swin-L (HTC) [47]	453.0M	-	1600	1.5	**59.4% / 59.1%**	-	-

Model	**#Param.**	**FLOPs**	**Size**	**FPS**^{A100}	**AP**^{test} / **AP**^{val}	**AP**$_{50}^{test}$	**AP**$_{75}^{test}$
DN-Deformable-DETR [41]	48.0M	265.0G	1333	23.0	- / 48.6%	-	-
ConvNeXt-B (C-M-RCNN) [53]	-	964.0G	1280	11.5	- / 54.0%	73.1%	58.8%
Swin-B (C-M-RCNN) [52]	-	982.0G	1280	10.7	- / 53.0%	71.8%	57.5%
DINO-5scale (R50) [89]	47.0M	860.0G	1333	10.0	- / 51.0%	-	-
ConvNeXt-L (C-M-RCNN) [53]	-	1354.0G	1280	10.0	- / 54.8%	73.8%	59.8%
Swin-L (C-M-RCNN) [52]	-	1382.0G	1280	9.2	- / 53.9%	72.4%	58.8%
ConvNeXt-XL (C-M-RCNN) [53]	-	1898.0G	1280	8.6	- / 55.2%	74.2%	59.9%

8. More comparison

YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8% AP test-dev / 56.8% AP min-val among all known real-time object detectors with 30 FPS or higher on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWIN-L Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutional-based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. More over, we train YOLOv7 only on MS COCO dataset from scratch without using any other datasets or pre-trained weights.

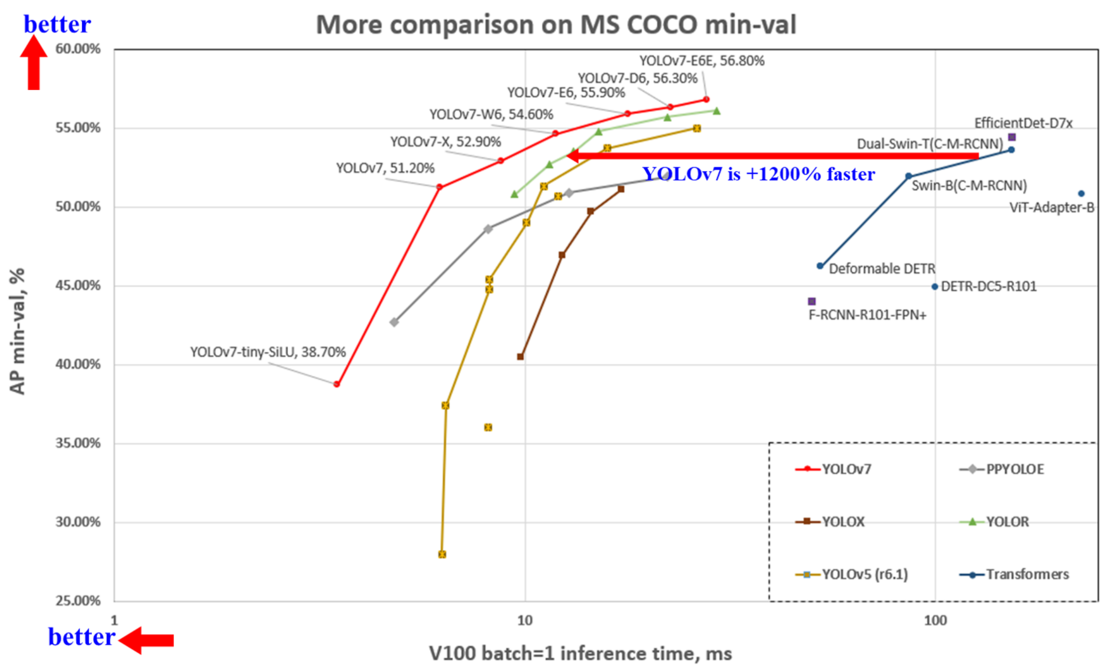


Figure 9: Comparison with other object detectors.

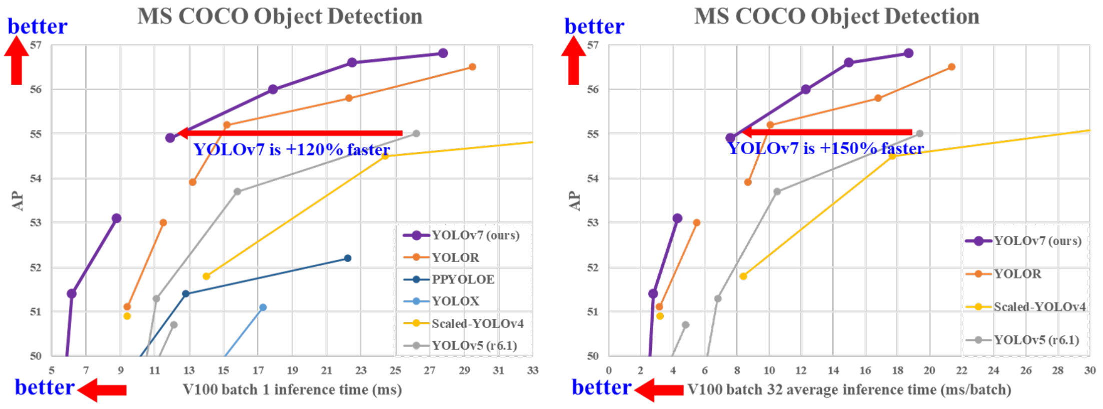


Figure 10: Comparison with other real-time object detectors.

Table 10: Comparison of different setting.

Model	Presicion	IoU threshold	AP^{val}
YOLOv7-X	FP16 (default)	0.65 (default)	**52.9%**
YOLOv7-X	FP32	0.65	**53.0%**
YOLOv7-X	FP16	0.70	**53.0%**
YOLOv7-X	FP32	0.70	**53.1%**
improvement	-	-	+0.2%

[*] Similar to meituan/YOLOv6 and PPYOLOE, our model could get higher AP when set higher IoU threshold.

The maximum accuracy of the YOLOv7-E6E (56.8% AP) real-time model is +13.7% AP higher than the current most accurate meituan/YOLOv6-s model (43.1% AP) on COCO dataset. Our YOLOv7-tiny (35.2% AP, 0.4 ms) model is +25% faster and +0.2% AP higher than meituan/YOLOv6-n (35.0% AP, 0.5 ms) under identical conditions on COCO dataset and V100 GPU with batch=32.

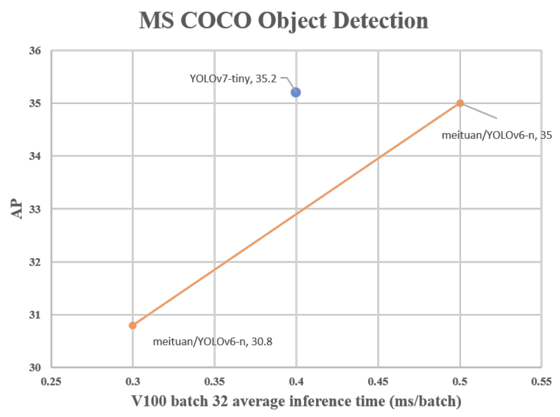


Figure 11: Comparison with other real-time object detectors.

References

[1] anonymous. Designing network design strategies. *anonymous submission*, 2022. 3

[2] Irwan Bello, William Fedus, Xianzhi Du, Ekin Dogus Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 2

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2, 6, 7

[4] Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang, and Pengyi Yang. Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*, 2(9):500–508, 2020. 2

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2020. 10

[6] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. AP-loss for accurate one-stage object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(11):3782–3798, 2020. 2

[7] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 10

[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/CVF International Conference on Computer Vision (ICCV)*, pages 502–511, 2019. 5

[9] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7373–7382, 2021. 2

[10] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. *arXiv preprint arXiv:2205.15242*, 2022. 2

[11] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. ACNet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1911–1920, 2019. 2

[12] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10886–10895, 2021. 2

[13] 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 (CVPR)*, pages 13733–13742, 2021. 2, 4

[14] Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[15] Piotr Dollár, Mannat Singh, and Ross Girshick. Fast and accurate model scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 924–932, 2021. 2, 3

[16] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. *arXiv preprint arXiv:2107.00057*, 2021. 2

[17] 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 (ICCV)*, pages 3490–3499, 2021. 2, 5

[18] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2020. 1

[19] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 2

[20] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. OTA: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 303–312, 2021. 2, 5

[21] 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, 7, 10

[22] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7036–7045, 2019. 2

[23] Jocher Glenn. YOLOv5 release v6.1. https://github.com/ultralytics/yolov5/releases/tag/v6.1, 2022. 2, 7, 10

[24] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. ExpandNets: Linear over-parameterization to train compact convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1298–1310, 2020. 2

[25] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. GhostNet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1580–1589, 2020. 1

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 1, 4, 5

[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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1314–1324, 2019. 1

[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. 1

[29] Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiaojin Gong, and Xiansheng Hua. Online convolutional re-parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[30] Miao Hu, Yali Li, Lu Fang, and Shengjin Wang. A^2-FPN: Attention aggregation based feature pyramid network for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15343–15352, 2021. 2

[31] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *International Conference on Learning Representations (ICLR)*, 2017. 2

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. 2, 4, 5

[33] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. 2

[34] Paul F Jaeger, Simon AA Kohl, Sebastian Bickelhaupt, Fabian Isensee, Tristan Anselm Kuder, Heinz-Peter Schlemmer, and Klaus H Maier-Hein. Retina U-Net: Embarrassingly simple exploitation of segmentation supervision for medical object detection. In *Machine Learning for Health Workshop*, pages 171–183, 2020. 1

[35] Hakan Karaoguz and Patric Jensfelt. Object detection approach for robot grasp detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4953–4959, 2019. 1

[36] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 355–371, 2020. 5

[37] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6399–6408, 2019. 2

[38] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015. 5

[39] 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 (CVPRW)*, pages 0–0, 2019. 2, 3

[40] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. GS3D: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1019–1028, 2019. 1

[41] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. DN-DETR: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13619–13627, 2022. 10

[42] Shuai Li, Chenhang He, Ruihuang Li, and Lei Zhang. A dual weighting label assignment scheme for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9387–9396, 2022. 2, 5

[43] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11632–11641, 2021. 5

[44] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21002–21012, 2020. 5

[45] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 2

[46] Zhuoling Li, Minghui Dong, Shiping Wen, Xiang Hu, Pan Zhou, and Zhigang Zeng. CLU-CNNs: Object detection for medical images. *Neurocomputing*, 350:53–59, 2019. 1

[47] Tingting Liang, Xiaojie Chu, Yudong Liu, Yongtao Wang, Zhi Tang, Wei Chu, Jingdong Chen, and Haibin Ling. CBNetV2: A composite backbone network architecture for object detection. *arXiv preprint arXiv:2107.00420*, 2021. 5, 10

[48] Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. Memory-efficient patch-based inference for tiny deep learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:2346–2358, 2021. 1

[49] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. MCUNet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:11711–11722, 2020. 1

[50] Yuxuan Liu, Lujia Wang, and Ming Liu. YOLOStereo3D: A step back to 2D for efficient stereo 3D detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 13018–13024, 2021. 5

[51] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong,

et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[52] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 10

[53] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022. 10

[54] Rangi Lyu. NanoDet-Plus. https://github.com/RangiLyu/nanodet/releases/tag/v1.0.0-alpha-1, 2021. 1, 2

[55] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 1, 3

[56] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:15534–15545, 2020. 2

[57] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Rank & sort loss for object detection and instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3009–3018, 2021. 2

[58] Shuvo Kumar Paul, Muhammed Tawfiq Chowdhury, Mircea Nicolescu, Monica Nicolescu, and David Feil-Seifer. Object detection and pose estimation from rgb and depth data for real-time, adaptive robotic grasping. In *Advances in Computer Vision and Computational Biology*, pages 121–142. 2021. 1

[59] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10213–10224, 2021. 2

[60] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10428–10436, 2020. 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 2, 5

[62] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE/CVF 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. 1, 2

[64] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019. 2

[65] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse DETR: Efficient end-to-end object detection with learnable sparsity. *arXiv preprint arXiv:2111.14330*, 2021. 5

[66] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 1

[67] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Object detection from scratch with deep supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(2):398–412, 2019. 5

[68] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[69] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse R-CNN: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14454–14463, 2021. 2

[70] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 5

[71] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 2

[72] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019. 2, 3

[73] Mingxing Tan and Quoc Le. EfficientNetv2: Smaller models and faster training. In *International Conference on Machine Learning (ICML)*, pages 10096–10106, 2021. 2

[74] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10781–10790, 2020. 2, 10

[75] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 2, 6

[76] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proceed-*

ings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 9627–9636, 2019. 2

[77] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(4):1922–1933, 2022. 2

[78] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. An improved one millisecond mobile backbone. *arXiv preprint arXiv:2206.04040*, 2022. 2

[79] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, 2021. 2, 3, 6, 7

[80] 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 (CVPRW)*, pages 390–391, 2020. 1

[81] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021. 1, 2, 6, 7, 10

[82] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15849–15858, 2021. 2, 5

[83] Bichen Wu, Chaojian Li, Hang Zhang, Xiaoliang Dai, Peizhao Zhang, Matthew Yu, Jialiang Wang, Yingyan Lin, and Peter Vajda. FBNetv5: Neural architecture search for multiple tasks in one run. *arXiv preprint arXiv:2111.10007*, 2021. 1

[84] Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh, and Bo Chen. MobileDets: Searching for object detection architectures for mobile accelerators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3825–3834, 2021. 1

[85] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. PP-YOLOE: An evolved version of YOLO. *arXiv preprint arXiv:2203.16250*, 2022. 2, 7, 8, 10

[86] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3D-MAN: 3D multi-frame attention network for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1863–1872, 2021. 5

[87] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2018. 1

[88] Guanghua Yu, Qinyao Chang, Wenyu Lv, Chang Xu, Cheng Cui, Wei Ji, Qingqing Dang, Kaipeng Deng, Guanzhong Wang, Yuning Du, et al. PP-PicoDet: A better real-time object detector on mobile devices. *arXiv preprint arXiv:2111.00902*, 2021. 1

[89] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. DINO: DETR with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022. 10

[90] 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 (CVPR)*, pages 8514–8523, 2021. 5

[91] 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 (CVPR)*, pages 9759–9768, 2020. 5

[92] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018. 1

[93] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. BYTETrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864*, 2021. 1

[94] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FAIRMOT: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021. 1

[95] 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)*, volume 34, pages 12993–13000, 2020. 2

[96] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. IoU loss for 2D/3D object detection. In *International Conference on 3D Vision (3DV)*, pages 85–94, 2019. 2

[97] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1, 2

[98] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A nested U-Net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2018. 5

[99] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. AutoAssign: Differentiable label assignment for dense object detection. *arXiv preprint arXiv:2007.03496*, 2020. 2, 5

[100] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 10

A. Appendix

A.1. Implementation details

A.1.1 Architectures

Figure A1 and Figure A2 show the architectures of the proposed YOLOv7 P5 and YOLOv7 P6, respectively. Please note that Down, Up, CSPSPP, and the compression rate of a transition layer of ELAN modules in different models have a little bit different implementation. The details can be found at https://github.com/WongKinYiu/yolov7/tree/main/cfg/deploy.

	YOLOv7	YOLOv7x	YOLOv7-tiny
Stage 0	3x3/1 Conv, 32	3x3/1 Conv, 40	
Stage 1	3x3/2 Conv, 64 3x3/1 Conv, 64	3x3/2 Conv, 80 3x3/1 Conv, 80	3x3/2 Conv, 32
Stage 2	3x3/2 Conv, 128 2-4 ELAN, 256	3x3/2 Conv, 160 2-6 ELAN, 320	3x3/2 Conv, 64 1-2 ELAN, 64
Stage 3	/2 Down, 256 2-4 ELAN, 512	/2 Down, 320 2-6 ELAN, 640	/2 MaxPool 1-2 ELAN, 128
Stage 4	/2 Down, 512 2-4 ELAN, 1024	/2 Down, 640 2-6 ELAN, 1280	/2 MaxPool 1-2 ELAN, 256
Stage 5	/2 Down, 1024 2-4 ELAN, 1024	/2 Down, 1280 2-6 ELAN, 1280	/2 MaxPool 1-2 ELAN, 512
Stage 5	CSPSPP, 512	CSPSPP, 640	CSPSPP, 256
Stage 4	*2 Up, 512 1-4 ELAN, 256	*2 Up, 640 2-6 ELAN, 320	*2 Up, 256 1-2 ELAN, 128
Stage 3	*2 Up, 256 1-4 ELAN, 128	*2 Up, 320 2-6 ELAN, 160	*2 Up, 128 1-2 ELAN, 64
Stage 4	/2 Down, 512 1-4 ELAN, 256	/2 Down, 640 2-6 ELAN, 320	/2 Down, 256 1-2 ELAN, 128
Stage 5	/2 Down, 1024 1-4 ELAN, 512	/2 Down, 1280 2-6 ELAN, 640	/2 Down, 512 1-2 ELAN, 256

Figure A1: Architectures of YOLOv7 P5 models.

	YOLOv7-W6	YOLOv7-E6	YOLOv7-D6	YOLOv7-E6E
Stage 1	/2 ReOrg 3x3/1 Conv, 64	/2 ReOrg 3x3/1 Conv, 80	/2 ReOrg 3x3/1 Conv, 96	/2 ReOrg 3x3/1 Conv, 80
Stage 2	3x3/2 Conv, 128 2-4 ELAN, 128	/2 Down, 160 2-6 ELAN, 160	/2 Down, 192 2-8 ELAN, 192	/2 Down, 160 2-6 E-ELAN, 160
Stage 3	3x3/2 Conv, 256 2-4 ELAN, 256	/2 Down, 320 2-6 ELAN, 320	/2 Down, 384 2-8 ELAN, 384	/2 Down, 320 2-6 E-ELAN, 320
Stage 4	3x3/2 Conv, 512 2-4 ELAN, 512	/2 Down, 640 2-6 ELAN, 640	/2 Down, 768 2-8 ELAN, 768	/2 Down, 640 2-6 E-ELAN, 640
Stage 5	3x3/2 Conv, 768 2-4 ELAN, 768	/2 Down, 960 2-6 ELAN, 960	/2 Down, 1152 2-8 ELAN, 1152	/2 Down, 960 2-6 E-ELAN, 960
Stage 6	3x3/2 Conv, 1024 2-4 ELAN, 1024	/2 Down, 1280 2-6 ELAN, 1280	/2 Down, 1536 2-8 ELAN, 1536	/2 Down, 1280 2-6 E-ELAN, 1280
Stage 6	CSPSPP, 512	CSPSPP, 640	CSPSPP, 768	CSPSPP, 640
Stage 5	*2 Up, 768 1-4 ELAN, 384	*2 Up, 960 1-6 ELAN, 480	*2 Up, 1152 1-8 ELAN, 576	*2 Up, 960 1-6 E-ELAN, 480
Stage 4	*2 Up, 512 1-4 ELAN, 256	*2 Up, 640 1-6 ELAN, 320	*2 Up, 768 1-8 ELAN, 384	*2 Up, 640 1-6 E-ELAN, 320
Stage 3	*2 Up, 256 1-4 ELAN, 128	*2 Up, 320 1-6 ELAN, 160	*2 Up, 384 1-8 ELAN, 192	*2 Up, 320 1-6 E-ELAN, 160
Stage 4	/2 Down, 512 1-4 ELAN, 256	/2 Down, 640 1-6 ELAN, 320	/2 Down, 768 1-8 ELAN, 384	/2 Down, 640 1-6 E-ELAN, 320
Stage 5	/2 Down, 768 1-4 ELAN, 384	/2 Down, 960 1-6 ELAN, 480	/2 Down, 1152 1-8 ELAN, 576	/2 Down, 960 1-6 E-ELAN, 480
Stage 6	/2 Down, 1024 1-4 ELAN, 512	/2 Down, 1280 1-6 ELAN, 640	/2 Down, 1536 1-8 ELAN, 768	/2 Down, 1280 1-6 E-ELAN, 640

Figure A2: Architectures of YOLOv7 P6 models.

For E-ELAN architecture, since our edge device do not support group convolution and shuffle operation, we are forced to implement it as an equivalence architecture, which is shown in Figure A3.

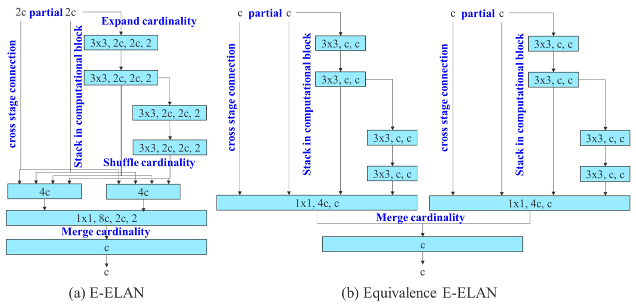


Figure A3: Equivalence E-ELAN.

The designed equivalence E-ELAN makes it easier for us to implement partial auxiliary head. E-ELAN with normal auxiliary head and partial auxilary head are shown in Figure A4.

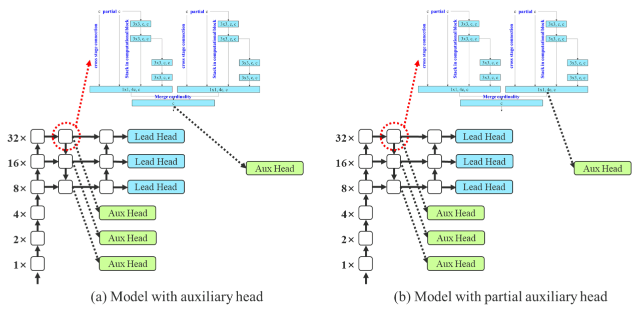

32×
16×
8×
4×
2×
1×
Lead Head
Aux Head
(a) Model with auxiliary head
(b) Model with partial auxiliary head

Figure A4: Partial auxilary head on E-ELAN.

Here we show how we make coarse-to-fine constraint lead head guided label assigner in Figure A5. We make a dynamic constraint by limiting the decoder of two additional candidate positive grids (yellow grids in the figure). Theoretically, yellow grids need to predict the range in [-1, 2] to fit ground truth bounding box, and we make the decoder can only predict in the range of [-0.5, 1.5]. This constraint makes the model can automatically learn the pink grids and yellow grid at different levels.

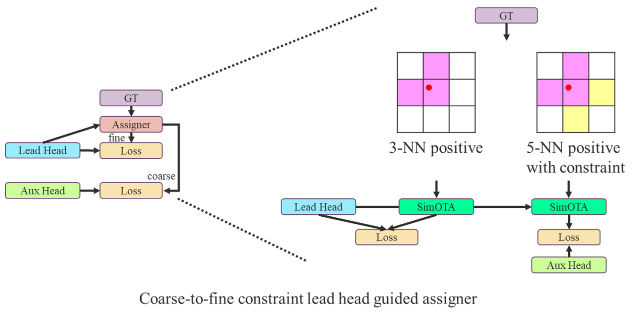

GT
Assigner
fine
coarse
Loss
Lead Head
Aux Head
3-NN positive
5-NN positive with constraint
SimOTA
Coarse-to-fine constraint lead head guided assigner

Figure A5: Coarse-to-fine lead head guided label assigner.

A.1.2 Hyper-parameters

We have three different training hyper-parameters settings. One is for YOLOv7-tiny, one is for YOLOv7 and YOLOv7x, and the last one is for YOLOv7-W6, YOLOv7-E6, YOLOv7-D6, and YOLOv7-E6E. The hyper-parameter setting are follow updated YOLOR code, which is described in https://github.com/WongKinYiu/yolor#yolor. These three hyper-parameter settings can be found at https://github.com/WongKinYiu/yolov7/tree/main/data, and they are respectively named by "hyp.scratch.tiny.yaml", "hyp.scratch.p5.yaml", and "hyp.scratch.p6.yaml".

An additional training hyper-parameter is top k of simOTA. To train 640×640 models, we follow YOLOX to use $k = 10$. For 1280×1280 models, as described in previous section, 3-NN candidate postive grids are used in our training. That is to say, when input resolution grows, the candidate positive grids will grow along two directions, not four directions. Therefore, we set $k = 20$ on simOTA to train 1280×1280 models.

A.1.3 Re-parameterization

Merging Convolution-BatchNorm-Activation into Convolution-Activation at inference time is a well-known technique, and its cprresponding formulas are shown in Figure A6. RepConv has become a popular re-parameterization method in recent years. Here we show how the YOLOR implicit knowledge can be merged into convolutional layer when addition or multiplication is used to combine implicit knowledge and representation. Figure A7 shows the formula to re-parameterize the implicit knowledge of YOLOR and the convolutional layer.

convolution → batch normalization → activation function

$$\begin{aligned} &(((wx + b) - m\ ) / v\ ) \\ &= (wx + (b-m)) / v \\ &= (w/v)x + (b-m)/v \\ &= w'x + b' \end{aligned}$$

Figure A6: Batch normalization as trainable BoF.

YOLOR+ → convolution → YOLOR+

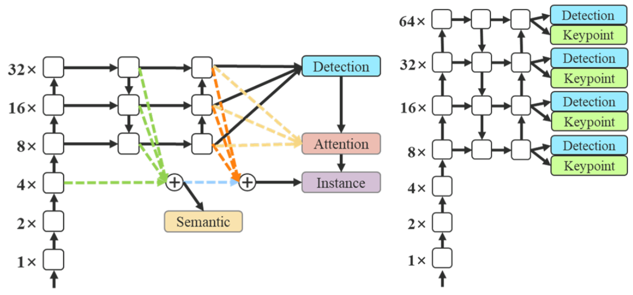
$$\begin{aligned} &w(x + g_1(z_1)) + b + g_2(z_2) \\ &= wx + (b + wg_1(z_1) + g_2(z_2)) \\ &= wx + b' \end{aligned}$$

YOLOR* → convolution → YOLOR*

$$\begin{aligned} &(w\ (g_1(z_1)x) + b)g_2(z_2) \\ &= g_1(z_1)g_2(z_2)wx + bg_2(z_2) \\ &= w'x + b' \end{aligned}$$

Figure A7: YOLOR implicit knowledge as trainable BoF.

A.2. More results



(a) YOLOv7-mask (b) YOLOv7-pose

Figure A8: Architectures of YOLOv7-mask and YOLOv7-pose.

A.2.1 YOLOv7-mask

We integrate YOLOv7 with BlendMask [1] to do instance segmentation. We simply fine-tune YOLOv7 object detection model on MS COCO instance segmentation dataset and trained for 30 epochs. It achieves state-of-the-art real-time instance segmentation result. The architecture of YOLOv7-mask and the corresponding results are shown in Figure A8 (a) and Figure A9, respectively.

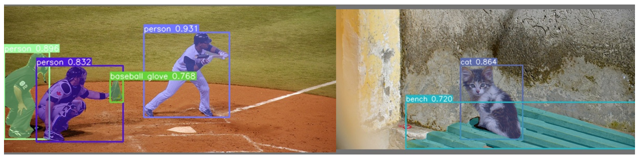


Figure A9: Sample results after applying YOLOv7-mask.

A.2.2 YOLOv7-pose

We integrate YOLOv7 with YOLO-Pose [2] to do keypoint detection. We follow the same setting as [2] to fine-tune YOLOv7-W6 people detection model on MS COCO keypoint detection dataset. YOLOv7-W6-pose achieves state-of-the-art real-time pose estimation result. The architecture and sample results are shown in Figure A8 (b) and Figure A10, respectively.

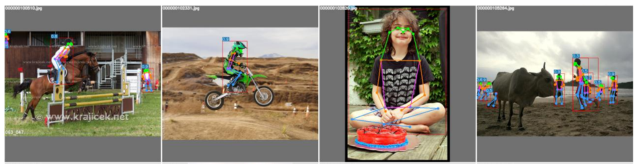


Figure A10: Sample results after applying YOLOv7-pose.

References

[1] Chen *et al.* BlendMask: Top-down meets bottom-up for instance segmentation. *CVPR*, 2020. 17

[2] Maji *et al.* YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss. *CVPRW*, 2022. 17