YOLOX-PAI: An Improved YOLOX, Stronger and Faster than YOLOv6

Xinyi Zou[1], Ziheng Wu[1],Wenmeng Zhou[1], Jun Huang[1]

[1] Platform of AI (PAI), Alibaba Group

`{zhoulou.wzh,wenmeng.zwm,huangjun.hj}@alibaba-inc.com`

Abstract

We develop an all-in-one computer vision toolbox named EasyCV to facilitate the use of various SOTA computer vision methods. Recently, we add YOLOX-PAI, an improved version of YOLOX, into EasyCV. We conduct ablation studies to investigate the influence of some detection methods on YOLOX. We also provide an easy use for PAI-Blade which is used to accelerate the inference process based on BladeDISC and TensorRT. Finally, we receive 42.8 mAP on COCO dateset within 1.0 ms on a single NVIDIA V100 GPU, which is a bit faster than YOLOv6. A simple but efficient predictor api is also designed in EasyCV to conduct end2end object detection. Codes and models are now available at: https://github.com/alibaba/EasyCV.

1 Introduction

YOLOX (Ge et al., 2021) is one of the most famous one-stage object detection methods, and has been widely used in a various field, such as automatic driving, defect inspection, *etc*. It introduces the decoupled head and the anchor-free manner into the YOLO series, and receives state-of-the-art results among 40 mAP to 50 mAP.

Considering its flexibility and efficiency, we intend to integrate YOLOX into our EasyCV, an all-in-one computer vision methods that helps even a beginner easily use a computer vision algorithm. In addition, we investigate the improvement upon YOLOX by using different enhancement of the detection backbone, neck, and head. Users can simply set different configs to obtain a suitable object detection model according to their own requirements. Also, based on PAI-Blade [1](an inference optimization framework by PAI), we further speed up the inference process and provide an easy api to use PAI-Blade in our EasyCV. Finally, we design an efficient predictor api to use our YOLOX-PAI in

an end2end manner, which accelerate the original YOLOX by a large margin. The comparisons between YOLOX-PAI and the state-of-the-art object detection methods have been shown in Fig. 1.

In brief, our main contributions are as follows:

- We release YOLOX-PAI in EasyCV as a simple yet efficient object detection tool (containing the docker image, the process of model training, model evaluation and model deployment). We hope that even a beginner can use our YOLOX-PAI to accomplish his object detection tasks.

- We conduct ablation studies of existing object detection methods based on YOLOX, where only a config file is used to construct self-designed YOLOX model. With the improvement of the architecture and the efficiency of PAI-Blade, we obtain state-of-the-art object detection results among 40 mAP and 50 mAP within 1ms for model inference on a single NVIDIA Tesla V100 GPU.

- We provide a flexible predictor api in EasyCV that accelerates both the preprocess, inference and postprocess procedure, respectively. In this way, user can better use YOLOX-PAI for end2end objection detection task.

2 Methods

In this section, we will take a brief review of the used methods in YOLOX-PAI. We conduct several improvements on both the detection backbone, neck, head. We also use PAI-Blade to accelerate the inference process.

2.1 Backbone

Recently, YOLOv6 and PP-YOLOE (Xu et al., 2022) have replaced the backbone of CSPNet (Wang et al., 2020) to RepVGG (Ding et al., 2021).

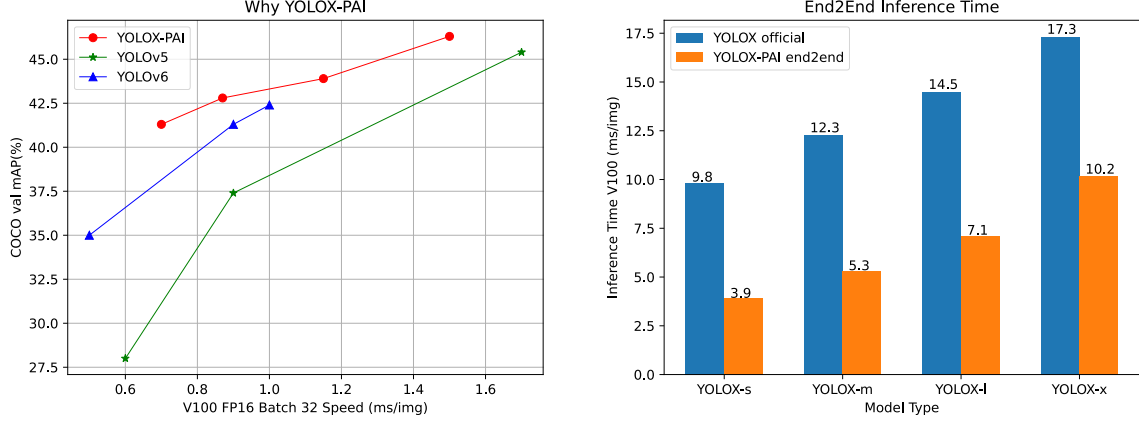[1] https://github.com/alibaba/BladeDISC

Figure 1: The comparisons between YOLOX-PAI and the existing methods.

In RepVGG, a 3x3 convolution block is used to replace a multi-branch structure during the inference process, which is beneficial to both save the inference time and improve the object detection results. Following YOLOv6,we also use a RepVGG-based backbone as a choice in YOLOX-PAI.

2.2 Neck

We use two methods to improve the performance of YOLOX in the neck of YOLOX-PAI, that is 1) Adaptively Spatial Feature Fusion (ASFF) (Liu et al., 2019) and its variance (denoted as ASFF_Sim) for feature augmentation 2) GSConv (Li et al., 2022), a lightweight convolution block to reduce the compute cost.

The original ASFF method uses several vanilla convolution blocks to first unify the dimension of different feature maps. Inspired by the Focus layer in YOLOv5, we replace the convolution blocks by using the non-parameter slice operation and mean operation to obtain the unified feature maps (denoted as ASFF_Sim). To be specific, the operation for each feature map of the output of YOLOX is defined in Fig. 2.

We also use two kinds of GSConv-based neck to optimize YOLOX. The used neck architectures are shown in Fig 3 and Fig 4. The differences of the two architectures are whether to replace all the blocks with GSConv. As proved by the authors, GSconv is specially designed for the neck where the channel reaches the maximum and the the size reaches the minimum.

2.3 Head

We enhance the YOLOX-Head with the attention mechanism as (Feng et al., 2021) to align the task

of object detection and classification (denoted as TOOD-Head). The architecture is shown in Fig. 5. A stem layer is first used to reduce the channel, following by a group of inter convolution layers to obtain the inter feature maps. Finally, the adaptive weight is computed according to different tasks. We test the result of using the vanilla convolution or the repvgg-based convolution in the TOOD-Head, respectively.

2.4 PAI-Blade

PAI-Blade is an easy and robust inference optimization framework for model acceleration. It is based on many optimization techniques, such as Blade Graph Optimizer, TensorRT, PAI-TAO (Tensor Accelerator and Optimizer), and so on. PAI-Blade will automatically search for the best method to optimize the input model. Therefore, people without the professional knowledge of model deployment can also use PAI-Blade to optimize the inference process. We integrate the use of PAI-Blade in EasyCV so that users are allowed to obtain an efficient model by simply change the export config.

2.5 EasyCV Predictor

Along with the model inference, the preprocess function and the postprocess function are also important in an end2end object detection task, which are often ignored by the existing object detection toolbox. In EasyCV, we allow user to choose whether to export the model with the preprocess/-postprocess procedure flexibly. Then, a predictor api is provided to conduct efficient end2end object detection task with only a few lines.
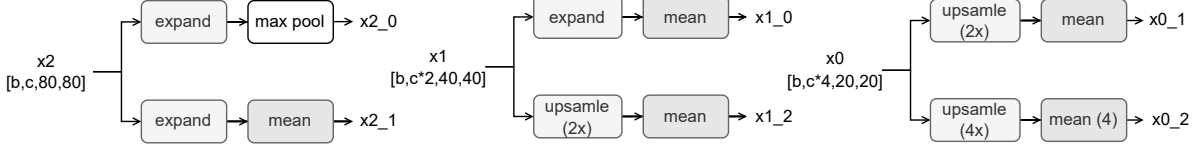
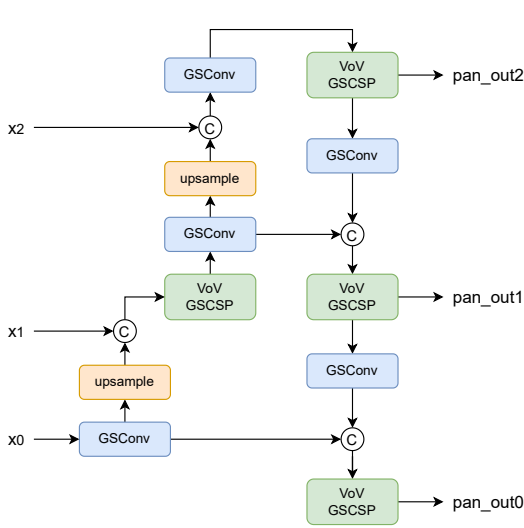Figure 2: Operations in ASFF_Sim. The expand operation is a slice operation based on the Focus layer.

Figure 3: Architecture of the GSConv-based YOLOX neck.

Figure 4: Architecture of the GSConv-based (part) YOLOX neck.

3 Experiments

In this section, we report the ablation study results of the above methods on the COCO dataset (Lin et al., 2014) and the comparisons between YOLOX-PAI and the SOTA object detection methods.

3.1 Comparisons with the SOTA methods

We select the useful improvements in YOLOX-PAI and compare it with the SOTA YOLOv6 method in Table 1. It can be seen that YOLOX-PAI is much faster compared to the corresponding version of YOLOv6 with a better mAP (i.e., obtain +0.2 mAP and 22% speed up, +0.2 mAP and 13% speed up of the YOLOv6-tiny, and the YOLOv6-s model, respectively).

3.2 Ablation studies

Influence of Backbone. As shown in Table 1, YOLOX with a RepVGG-based backbone achieve better mAP with only a little sacrifice of speed. It indeed adds more parameters and flops that may need more computation resource, but does not require much inference time. Considering its effi-ciency, we make it as a flexible config setting in EasyCV.
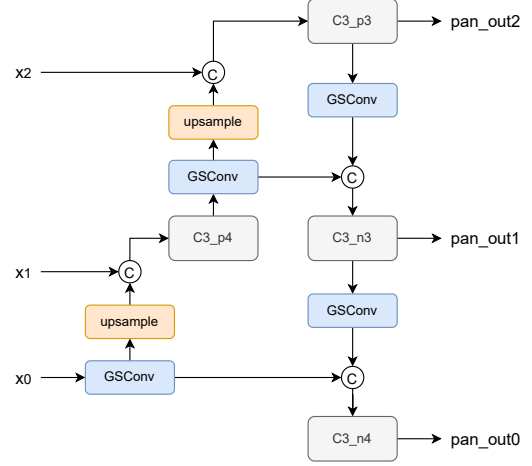
Influence of Neck. The influence of ASFF and ASFF_Sim are shown in Table 2. It shows that, compared with ASFF, ASFF_Sim is also benefical to improve the detection result with only a little gain of parameters and flops. However, the time cost is much larger and we will implement the CustomOP to optimize it in the future. The influence of GSConv is shown in Table 3. The result is that GSConv will bring 0.3 mAP and reduce 3% of speed on a single NVIDIA V100 GPU.

Influence of Head. The influence of the TOOD-Head is shown in Table 4. We investigate the influence of different number of inter convolution layers. We show that when adding additional inter convolution layers, the detection results can be better. It is a trade-off between the speed and accuracy to choose a suitable hyperparameter. We also show that when replacing the vanilla convolution with the repconv-based convolution in the inter convolution layers, the result become worse. It will slightly improve the result when using repconv-based cls_conv/reg_conv layer (in Fig. 5) when the stack number is small (i.e., 2, 3).
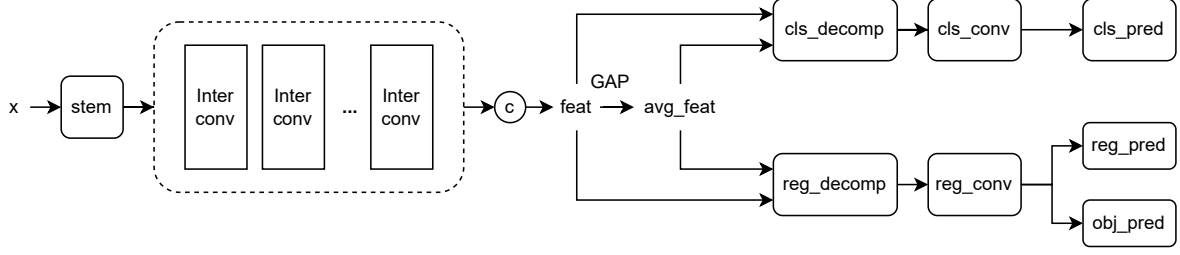
Figure 5: Architecture of the TOOD-Head.

Table 1: Comparisons between YOLOX-PAI and SOTA methods on the COCO dataset.

Model	Params (M)	Flops (G)	mAP^{val} 0.5:0.95 (672)	mAP^{val} 0.5:0.95 (640)	$Speed^{V100}$ fp16 bs32 (ms)
YOLOXs [‡]	9.0	26.8	40.2	40.1	0.68
YOLOv6-tiny	15.0	36.7	41.3	-	0.9
PAI-YOLOXs	15.9	36.8	41.5	41.4	0.70
YOLOv6-s	17.2	44.2	43.1	42.4	1.0
PAI-YOLOXs-ASFF	21.3	41.0	43.3	42.8	0.87
PAI-YOLOXs-ASFF-TOOD3	23.7	49.9	44.0	43.9	1.15
YOLOXm [‡]	25.3	73.8	46.3	46.3	1.50

[‡] denotes the re-implementation YOLOX result in EasyCV, which is optimized by PAI-Blade.

Table 2: Ablation studies of ASFF and its variance.

Model	Params (M)	Flops (G)	mAP^{val} 0.5:0.95 (640)	$Speed^{V100}$ fp16 bs32 (ms)
Baseline	15.9	36.8	41.4	0.70
+ASFF	21.3	41.0	42.8	0.87
+ASFF-Sim	16.3	37.1	42.6	1.13

* PAI-YOLOXs is the baseline.

Table 3: Ablation studies of GSConv block.

Model	Params (M)	Flops (G)	mAP^{val} 0.5:0.95 (640)	$Speed^{V100}$ fp16 bs32 (ms)
Baseline	2.83	2.67	41.4	0.70
GSConv	1.22	1.33	41.6	0.78
GSConv_part	2.39	2.39	41.7	0.72

* only the parameter and flops of neck is computed.
* PAI-YOLOXs is the baseline.

Table 4: Ablation studies of TOOD-Head.

Model	Params (M)	Flops (G)	mAP^{val} 0.5:0.95 (640)	$Speed^{V100}$ fp16 bs32 (ms)
Baseline	1.92	5.23	42.8	0.87
stack=2	2.22	7.71	43.5	1.09
stack=3	2.37	8.94	43.9	1.15
stack=4	2.53	10.18	44.1	1.24
stack=5	2.68	11.42	44.4	1.32
stack=6	2.83	12.66	44.7	1.40

* only the parameter and flops of head is computed.
* PAI-YOLOXs-ASFF is the baseline.

Table 5: End2end inference time of YOLOX-s with different export configs.

export_type	preprocess_jit	use_trt_efficientnms	Inference Time^{V100} bs1 end2end (ms)
raw	-	-	24.58
jit	×	×	18.30
jit	×	✓	18.38
jit	✓	×	13.44
jit	✓	✓	13.04
blade	×	×	8.72
blade	×	✓	9.39
blade	✓	×	3,93
blade	✓	✓	4.53

3.3 End2end results

Table 5 shows the end2end prediction results of YOLOXs model with different export configs. The keywords in the table are the same as in our EasyCV config file. It is evident that the blade optimization is useful to optimize the inference process. Also, the preprocess process can be greatly speed up by the exported jit model. As for the postprocess, we are still work on it to realize a better CustomOP that can be optimized by PAI-Blade for a better performance. In the right part of Fig 1, we show that with the optimization of PAI-Blade and EasyCV predictor, we can receive a satisfactory end2end inference time on YOLOX.

4 Conclusion

In this paper, we introduced YOLOX-PAI, an improved version of YOLOX based on EasyCV. It receives SOTA object detection results among 40 mAP and 50 mAP with the improvement of the model architecture and PAI-Blade. We also provide an easy and efficient predictor api to conduct end2end object detection in a flexible way. EasyCV is an all-in-one toolkit box that focuses on SOTA computer vision methods, especially in the field of self-supervised learning and vision transformer. We hope that users can conduct computer vision tasks immediately and enjoy CV by using EasyCV!

Acknowledgments

References

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

Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. 2021. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499. IEEE Computer Society.

Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. 2021. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*.

Hulin Li, Jun Li, Hanbing Wei, Zheng Liu, Zhenfei Zhan, and Qiliang Ren. 2022. Slim-neck by gsconv: A better design paradigm of detector architectures for autonomous vehicles. *arXiv preprint arXiv:2206.02424*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Songtao Liu, Di Huang, and Yunhong Wang. 2019. Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516*.

Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. 2020. 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.

Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. 2022. Pp-yoloe: An evolved version of yolo. *arXiv preprint arXiv:2203.16250*.