You Only Look One-level Feature

Qiang Chen[1,2*], Yingming Wang[4], Tong Yang[4], Xiangyu Zhang[4], Jian Cheng[1,2,3†], Jian Sun[4]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
[4]MEGVII Technology
{qiang.chen, jcheng}@nlpr.ia.ac.cn, {wangyingming, yangtong, zhangxiangyu, sunjian}@megvii.com

Abstract

This paper revisits feature pyramids networks (FPN) for one-stage detectors and points out that the success of FPN is due to its divide-and-conquer solution to the optimization problem in object detection rather than multi-scale feature fusion. From the perspective of optimization, we introduce an alternative way to address the problem instead of adopting the complex feature pyramids - utilizing only one-level feature for detection. *Based on the simple and efficient solution, we present You Only Look One-level Feature (YOLOF). In our method, two key components, Dilated Encoder and Uniform Matching, are proposed and bring considerable improvements. Extensive experiments on the COCO benchmark prove the effectiveness of the proposed model. Our YOLOF achieves comparable results with its feature pyramids counterpart RetinaNet while being* $2.5\times$ *faster. Without transformer layers, YOLOF can match the performance of DETR in a single-level feature manner with* $7\times$ *less training epochs. Code is available at* https://github.com/megvii-model/YOLOF.

1. Introduction

In state-of-the-art two-stage detectors [19, 10, 1] and one-stage detectors [20, 35], feature pyramids become an essential component. The most popular way to build feature pyramids is the feature pyramid networks (FPN) [19], which mainly brings two benefits: (1) *multi-scale feature fusion*: fusing multiple low-resolution and high-resolution feature inputs to obtain better representations; (2) *divide-and-conquer*: detecting objects on different levels regarding objects' scales. A common belief for FPN is that its success relies on the fusion of multiple level features, inducing a line of studies of designing complex fusion methods manually [22, 14, 25], or via Neural Architecture Search (NAS) algorithms [7, 34]. However, the belief ignores the function of the divide-and-conquer in FPN. It leads to fewer studies on how these two benefits contribute to FPN's success and may hinder new advances.

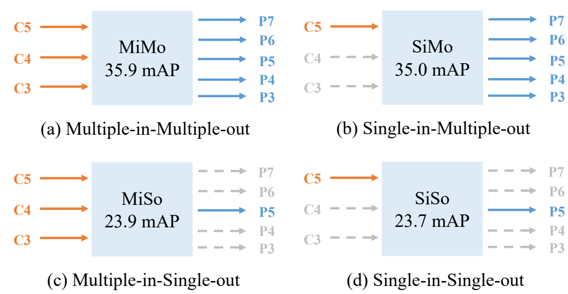

Figure 1. Comparison of box AP among the Multiple-in-Multiple-out (MiMo), Single-in-Multiple-out (SiMo), Multiple-in-Single-out (MiSo), and Single-in-Single-out (SiSo) encoders on COCO validation set. Here, we adopt the original RetinaNet [20] as our baseline model, where C3, C4, and C5 denote output features of the backbone with a downsample rate of $\{8, 16, 32\}$ and P3 to P7 represent the feature levels used for final detection. All results reported in the figure use the same backbone, ResNet-50 [11]. The structure of MiMo is same as the FPN in RetinaNet [20].

This paper studies the influence of FPN's two benefits in one-stage detectors. We design experiments by decoupling the *multi-scale feature fusion* and the *divide-and-conquer* functionalities with RetinaNet [20]. In detail, we consider FPN as a *Multiple-in-Multiple-out* (MiMo) encoder, which encodes multi-scale features from the backbone and provides feature representations for the decoder (the detection heads). We conduct controlled comparisons among *Multiple-in-Multiple-out* (MiMo), *Single-in-Multiple-out* (SiMo), *Multiple-in-Single-out* (MiSo), and *Single-in-Single-out* (SiSo) encoders in Figure 1. Surprisingly, the SiMo encoder, which only has one input feature C5 and does not perform feature fusion, can achieve com-

*This work is done during Qiang Chen's internship at MEGVII Technology.
†Corresponding author.

parable performance with the MiMo encoder (i.e., FPN). The performance gap is less than 1 mAP. In contrast, the performance drops dramatically ($\geq$ 12 mAP) in MiSo and SiSo encoders. These phenomenons suggest two facts: (1) the C5 feature carries sufficient context for detecting objects on various scales, which enables the SiMo encoder to achieve comparable results; (2) the *multi-scale feature fusion* benefit is far away less critical than the *divide-and-conquer* benefit, thus *multi-scale feature fusion might not be the most significant benefit of FPN*, which is also demonstrated by ExFuse [44] in semantic segmentation. Thinking one step deeper, *divide-and-conquer* is related to the optimization problem in object detection. It divides the complex detection problem into several sub-problems by object scales, facilitating the optimization process.

The above analysis suggests that the essential factor for the success of FPN is its solution to the optimization problem in object detection. The *divide-and-conquer* solution is a good way. But it brings memory burdens, slows down the detectors, and make detectors' structure complex in one-stage detectors like RetinaNet [20]. Given that the C5 feature carries sufficient context for detection, we show a simple way to address the optimization problem.

We propose *You Only Look One-level Feature (YOLOF)*, which only uses one single C5 feature (with a downsample rate of 32) for detection. To bridge the performance gap between the SiSo encoder and the MiMo encoder, we first design the structure of the encoder properly to extract the multi-scale contexts for objects on various scales, compensating for the lack of multiple-level features; then, we apply a uniform matching mechanism to solve the imbalance problem of positive anchors raised by the sparse anchors in the single feature.

Without bells and whistles, YOLOF achieves comparable results with its feature pyramids counterpart RetinaNet [20] but $2.5\times$ faster. In a single feature manner, YOLOF matches the performance of the recent proposed DETR [2] while converging much faster ($7\times$). In a nutshell, the contributions of this paper are:

- We show that the most significant benefits of FPN is its *divide-and-conquer* solution to the optimization problem in dense object detection rather than the *multi-scale feature fusion*.

- We present YOLOF, which is a simple and efficient baseline without using FPN. In YOLOF, we propose two key components, *Dilated Encoder* and *Uniform Matching*, bridging the performance gap between the SiSo encoder and the MiMo encoder.

- Extensive experiments on COCO benchmark indicates the importance of each component. Moreover, we conduct comparisons with RetinaNet [20] and DETR [2]. We can achieve comparable results with a faster speed on GPUs.

2. Related Works

Multiple-level feature detectors. It is a conventional technique to employ multiple features for object detection. Typical approaches to construct multiple features can be categorized into image pyramid methods and feature pyramid methods. Image pyramids based detector such as DPM [6] dominates the detection in the pre-deep learning era. In CNN-based detectors, the image pyramids method also wins some researchers' [31, 32] praise as it can achieve higher performance out of the box. However, the image pyramids method is not the only way to obtain multiple features; it is more efficient and natural to exploit feature pyramids' power in CNN models. SSD [23] first utilizes multiple-scale features and performs object detection on each scale for different scales objects. FPN [19] follows SSD [23] and UNet [30] and constructs semantic-riched feature pyramids by combining shallow features and deep features. After that, several works [14, 22, 7, 34] follow FPN and focus on how to obtain better representations. FPN becomes an essential component and dominates modern detectors. It is also applied to popular one-stage detectors, such as RetinaNet [20], FCOS [35], and their variants [42]. Another line of method to get feature pyramids is to use multi-branch and dilation convolution [17]. Different from the above works, our method is a single-level feature detector.

Single-level feature detectors. In early times, the R-CNN series [9, 8, 28] and R-FCN [4] only extract RoI features on a single feature, while their performances lag behind their multiple feature counterparts [19]. Also, in one-stage detectors, YOLO [26] and YOLOv2 [27] only use the last output feature of the backbone. They can be super fast but have to bear a performance decline in detection. CornerNet [16] and CenterNet [45, 5] follow this fashion and achieve competitive results while using a single feature with a downsample rate of 4 to detect all the objects. Using a high-resolution feature map for detection brings enormous memory cost and is not friendly to deployment. Recently, DETR [2] introduces the transformer [36] to detection and shows that it could achieve state-of-the-art results only use a single C5 feature. Due to the totally anchor-free mechanism and transformer learning phase, DETR needs a long training schedule for its convergence. The long training schedule characteristic is cumbersome for further improvements. Unlike these papers, we investigate the working mechanism of multiple-level detection. From the perspective of optimization, we provide an alternative solution to the widely used FPN. Moreover, YOLOF converges faster

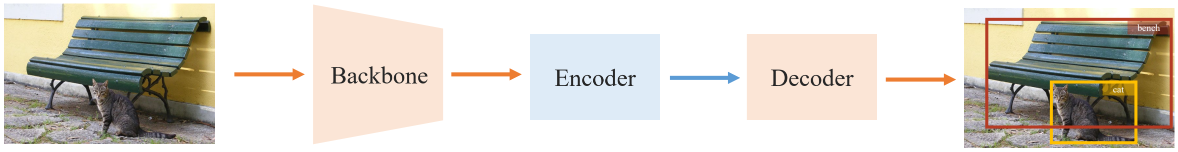

Figure 2. An illustration of the detection pipeline. In this paper, we format the detection pipeline into three parts: (1) the backbone; (2) the encoder, which receives inputs from the backbone and distributes representations for detection; (3) the decoder, which performs classification and regression tasks and generate final prediction boxes. The color for the encoder is corresponding to the one in Figure 1.

and achieves promising performance; thus, YOLOF can serve as a simple baseline for fast and accurate detectors.

3. Cost Analysis of MiMo Encoders

As mentioned in Section 1, the success of FPN in dense object detection is due to its solution to the optimization problem. However, the multi-level feature paradigm is inevitable to make detectors complex, brings memory burdens, and slows down the detector. In this section, we provide a quantitative study on the cost of MiMo encoders.

We design experiments based on RetinaNet [20] with ResNet-50 [11]. In detail, we format the pipeline for the detection task as a combination of three key parts: the backbone, the encoder, and the decoder (Figure 2). In this view, we show the FLOPs of each component in Figure 3. Compared with SiSo encoders, the MiMo encoder brings enormous memory burdens to the encoder and the decoder(134G vs. 6G) (Figure 3). Moreover, the detector with MiMo encoder runs much slower than the ones with SiSo encoders (13 FPS vs. 34 FPS) (Figure 3). The slow speed is caused by detecting objects on high-resolution feature maps in the detector with MiMo encoder, such as the C3 feature (with a downsample rate of 8). Given the above drawbacks of the MiMo encoder, we aim to find an alternative way to solve the optimization problem while keeping the detector simple, accurate, and fast simultaneously.

4. Method

Motivated by the above purpose and the finding that the C5 feature contains enough context for detecting numerous objects, we try to replace the complex MiMo encoder with the simple SiSo encoder in this section. But this replacement is **nontrivial** as the performance drops extensively when applying SiSo encoders according to the results in Figure 3. Given the situation, we carefully analyze the obstacles preventing SiSo encoders from getting a comparable performance with MiMo encoders. We find that two problems brought by SiSo encoders are responsible for the performance drop. The first problem is that *the range of scales matching to the C5 feature's receptive field is limited*, which impedes the detection performance for objects across various scales. The second one is *the imbalance problem on positive anchors* raised by sparse anchors in the single-level feature. Next, we discuss these two problems in detail and provide our solutions.

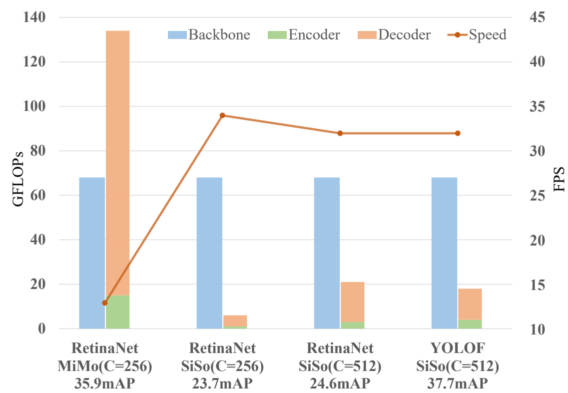

Figure 3. FLOPs, accuracy, and speed comparison between the models that adopt MiMo and SiSo encoders on COCO. As the FLOPs of the decoder is affected by the encoder's outputs, we stack the FLOPs of the encoder and the decoder in the figure to better understanding the effects of encoders on the FLOPs. All models use the same backbone, ResNet-50. All FLOPs are measured with a shorter edge size 800 over the first 100 images of COCO val2017. The FPS is calculated with batch size 1 on 2080Ti from the total inference pure compute time reported in the Detectron2 [38]. In the figure, C represents the number of channels used in the model's encoder and decoder.

4.1. Limited Scale Range

Recognizing objects at vastly different scales is a fundamental challenge in object detection. One feasible solution to this challenge is to leverage multiple-level features. In detectors with MiMo or SiMo encoders, they construct multiple-level features with different receptive fields (P3-P7) and detect objects on the level with receptive field matching to their scales. However, the single-level feature setting changes the game. There is only one output feature in SiSo encoders, whose receptive field is a constant. As shown in Figure 4(a), the C5 feature's receptive field can only cover a limited scale range, resulting in poor performance if the objects' scales mismatches with the receptive field. To achieve the goal of detecting all objects with SiSo encoders, we have to find a way to generate an output feature with various receptive fields, compensating for the lack of multiple-level features.

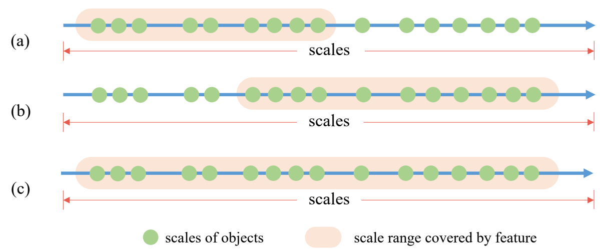

Figure 4. A toy example to illustrate the relation between the object scales and the scale range covered by the single feature. The axis in this figure denotes the scales. (a) indicates that the feature's receptive field can only cover a limited scale range; (b) shows that the enlarged scale ranges enable the feature to cover large objects while miss covering small ones; (c) represents that all scales can be covered the feature with multiple receptive fields.

We begin with enlarging the receptive field of the C5 feature by stacking standard and dilated convolutions [40]. Although the covered scale range is enlarged to some extent, it still can not cover all object scales as the enlarging process multiplies a factor greater than 1 to all originally covered scales. We illustrate the situation in Figure 4(b), where the whole scale range shifts to larger scales compare with the one in Figure 4(a). Then, we combine the original scale range and the enlarged scale range by adding the corresponding features, resulting in an output feature with multiple receptive fields covering all object scales (Figure 4(c)). The above operations can be easily achieved by constructing residual blocks [11] with dilations on the middle 3×3 convolution layer.

Dilated Encoder: Based on the above designs, we propose our SiSo encoder in Figure 5, named as *Dilated Encoder*. It contains two main components: the *Projector* and the *Residual Blocks*. The projection layer first applies one 1×1 convolution layer to reduce the channel dimension, then add one 3×3 convolution layer to refine semantic contexts, which is the same as in the FPN [19]. After that, we stack four successive dilated residual blocks with different dilation rates in the 3×3 convolution layers to generate output features with multiple receptive fields, covering all objects' scales.

Discussion: Dilated convolution [40] is a common strategy to enlarge the features' receptive field in object detection. As reviewed in the Section 2, TridentNet [17] use dilated convolution to generate multi-scale features. It deals with the scale variation problem in object detection via multi-branch structure and weight sharing mechanism, which is different from our single-level feature setting. Moreover, *Dilated Encoder* stack dilated residual blocks one by one without weight sharing. Although DetNet [18] also successively applies dilated residual blocks, its purpose is to maintain the spatial resolution of the features and keep more details in the backbone's outputs, while ours is to generate a feature with multiple receptive fields out of the backbone. The design of *Dilated Encoder* enables us to detecting all objects on single-level feature instead of on multiple-level features like TridentNet [17] and DetNet [18].

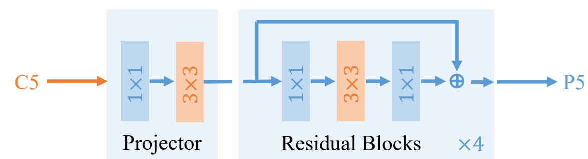

Figure 5. An illustration of the structure of *Dilated Encoder*. In the figure, 1×1 and 3×3 denotes 1×1 and 3×3 convolution layers and $\times 4$ means four successive residual blocks. All convolution layers in Residual Blocks are followed by a batchnorm layer [12] and a ReLU layer [24], while in Projector, we only use convolution layers and batchnorm layers [12].

4.2. Imbalance Problem on Positive Anchors

The definition of positive anchors is crucial for the optimization problem in object detection. In anchor-based detectors, strategies to define positive are dominated by measuring the IoUs between anchors and ground-truth boxes. In RetinaNet [20], if the max IoU of the anchor and ground-truth boxes is greater than a threshold 0.5, this anchor will be set as positive. We call it Max-IoU matching.

In MiMo encoders, the anchors are pre-defined on multiple levels in a dense paved fashion, and the ground-truth boxes generate positive anchors in feature levels corresponding to their scales. Given the divide-and-conquer mechanism, Max-IoU matching enables ground-truth boxes in each scale to generate a sufficient number of positive anchors. However, when we adopt the SiSo encoder, the number of anchors diminish extensively compare to the one in the MiMo encoder, from $100k$ to $5k$, resulting in sparse anchors[1]. Sparse anchors raise a matching problem for detectors when applying Max-IoU matching, as shown in Figure 6. Large ground-truth boxes induce more positive anchors than small ground-truth boxes in natural, which cause an imbalance problem for positive anchors. This imbalance makes detectors pay attention to large ground-truth boxes while ignoring the small ones when training.

Uniform Matching: To solve this imbalance problem in positive anchors, we propose an *Uniform Matching* strategy: marking the indexes of the *k nearest* anchor and the *k nearest* predicted boxes as positives for each ground-truth box, which makes sure that all ground-truth boxes can be matched with the same number of positive anchors uniformly regardless of their sizes (Figure 6). Balance in positive samples makes sure that all ground-truth boxes partic-

[1] In SiSo encoders, we simply collapse multiple anchors on multiple-level features to single-level, e.g., we construct 5 anchors with different anchor sizes of $\{32, 64, 128, 256, 512\}$ on each position of the C5 feature.

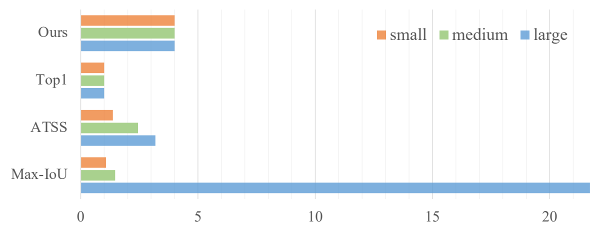

Figure 6. Distribution of the generated positive anchors in various matching methods with single feature. This figure aims to show the balancedness of the generated positive anchors. The positive anchors in the Max-IoU are dominated by large ground-truth boxes, causing huge imbalance across object scales. ATSS alleviates the imbalance problem by adaptively sampling positive anchors when training. The Top1 and Ours adopt a uniform matching, generating positive anchors in a balanced manner regardless of small, medium, and large objects.

ipate in training and contribute equally. Besides, following Max-IoU matching [20], we set IoU thresholds in Uniform Matching to ignore large IoU ($>$0.7) negative anchors and small IoU ($<$0.15) positive anchors.

Discussion: relation to other matching methods. Applying topk in the matching process is not new. ATSS [42] first select topk anchors for each ground-truth box on $\mathcal{L}$ feature levels, then samples positive anchors among $k \times \mathcal{L}$ candidates by dynamic IoU thresholds. However, ATSS focuses on defining positives and negatives adaptively, while our uniform matching focuses on achieving **balance on positive samples with sparse anchors**. Although several previous methods achieve balance on positive samples, their matching processes are **not** designed for this imbalance problem. For example, YOLO [26] and YOLOv2 [27] match the ground-truth boxes with the best matching cell or anchor; DETR [2] and [33] apply Hungarian algorithm [15] for matching. These matching methods can be view as top1 matching, which is a specific case of our uniform matching. More importantly, the difference between the uniform matching and the learning-to-match methods is that: the learning-to-match methods, such as FreeAnchor [43] and PAA [13], adaptively separate anchors into positives and negatives according to the learning status, while uniform matching is **fixed** and does not evolve with training. The uniform matching is proposed to address the specific imbalance problem on positive anchors under the SiSo design. The comparison in Figure 6 and the results in Table 4e demonstrate the significance of the balance in positives in SiSo encoders.

4.3. YOLOF

Based on the solutions above, we propose a fast and straightforward framework with single-level feature, denoted as YOLOF. We format YOLOF into three parts: the backbone, the encoder, and the decoder. The sketch of YOLOF is shown in Figure 2. In this section, we give a brief introduction to the main components of YOLOF.

Backbone. In all models, we simply adopt the ResNet [11] and ResNeXt [39] series as our backbone. All models are pre-trained on ImageNet. The output of the backbone is the C5 feature map which has 2048 channels and with a downsample rate of 32. To make a fair comparison with other detectors, all batchnorm layers in the backbone are frozen by default.

Encoder. For the encoder (Figure 5), we first follow FPN by adding two projection layers (one 1×1 and one 3×3 convolution) after the backbone, resulting in a feature map with 512 channels. Then, to enable the encoder's output feature to cover all objects on various scales, we propose to add residual blocks, which consist of three consecutive convolutions: the first 1×1 convolution apply channel reduction with a reduction rate of 4, then a 3×3 convolution with dilation is used to enlarge the receptive field, at last, a 1×1 convolution to recover the number of channels.

Decoder. For the decoder, we adopt the main design of RetinaNet, which consists of two parallel task-specific heads: the classification head and the regression head. We only add two minor modifications. The first one is that we follow the design of FFN in DETR [2] and make the number of convolution layers in two heads different. There are four convolutions followed by batch normalization layers and ReLU layers on the regression head while only have two on the classification head. The second is that we follow Autoassign [46] and add an implicit objectness prediction (without direct supervision) for each anchor on the regression head. The final classification scores for all predictions are generated by multiplying the classification output with the corresponding implicit objectness.

Other Details. As mentioned in the previous section, the pre-defined anchors in YOLOF are sparse, decreasing the match quality between anchors and ground-truth boxes. We add a random shift operation on the image to circumvent this problem. The operation shifts the image randomly with a maximum of 32 pixels in left, right, top, and bottom directions and aims to inject noises into the object's position in the image, increasing the probability of ground-truth boxes matching with high-quality anchors. Moreover, we found that a restriction on the anchors' center's shift is also helpful to the final classification when using a single-level feature. We add a restriction that the centers' shift for all anchors should smaller than 32 pixels.

5. Experiments

We evaluate our YOLOF on the MS COCO [21] benchmark and conduct comparisons with RetinaNet [20] and DETR [2]. Then, we provide a detailed ablation study

Model	schedule	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#params	GFLOPs	FPS
RetinaNet [20]	1x	35.9	55.7	38.5	19.4	39.5	48.2	38M	201	13
RetinaNet-R101 [20]	1x	38.3	58.5	41.3	21.7	42.5	51.2	57M	266	11
RetinaNet+	1x	37.7	58.1	40.2	22.2	41.7	49.9	38M	201	13
RetinaNet-R101+	1x	40.0	60.4	42.7	23.2	44.1	53.3	57M	266	10
YOLOF	1x	37.7	56.9	40.6	19.1	42.5	53.2	44M	86	32
YOLOF-R101	1x	39.8	59.4	42.9	20.5	44.5	54.9	63M	151	21
YOLOF-X101	1x	42.2	62.1	45.7	23.2	47.0	57.7	102M	289	10
YOLOF-X101†	3x	44.7	64.1	48.6	25.1	49.2	**60.9**	102M	289	10
YOLOF-X101†‡	3x	**47.1**	**66.4**	**51.2**	**31.8**	**50.9**	60.6	102M	-	-

Table 1. Comparison with RetinaNet on the COCO2017 validation set. The top section shows the results of RetinaNet. The middle section gives the results of an improved RetinaNet (with a "+"), which is RetinaNet with GIoU [29], GN [37], and implicit objectness. The last section shows the results of various YOLOF models. In the table, the model with a suffix of R101 or X101 means it use ResNet-101 [11] or RetNeXt-101-64×4d [39] as backbone. For those not marked with suffix, they adopt ResNet-50 [11] by default. In the last two rows, we use multi-scale training and testing techniques († indicates multi-scale training and ‡ means multi-scale testing), whose settings follow HTC [3]. In the last three columns, we show models' number of parameters (#params), GFLOPs, and inference speed. All FLOPs are measured with a shorter edge size 800 over the first 100 images of COCO val2017. Moreover, the FPS in the table is calculated with batch size 1 on 2080Ti from the total inference pure compute time reported in the Detectron2 [38].

of each component's design with quantitative results and analysis. Finally, to give insights to further research on single-level detection, we provide error analysis and show the weaknesses of YOLOF compared with DETR [2]. The details are as follows.

Implementation Details. YOLOF is trained with synchronized SGD over 8 GPUs with a total of 64 images per mini-batch (8 images per GPU). All models are trained with an initial learning rate of 0.12. Moreover, following DETR [2], we set a smaller learning rate for the backbone, which is $1/3$ of the base learning rate. To stabilize the training at the beginning, we extend the number of warmup iterations from 500 to 1500. For training schedules, as we increase the batch size, the '$1\times$' schedule setting in YOLOF is a total of $22.5k$ iterations and with base learning rate decreased by 10 in the $15k$ and the $20k$ iteration. Other schedules are adjusted according to the principles in Detectron2 [38]. For model inference, we employ NMS with a threshold of 0.6 to post-process the results. For other hyperparameters, we follow the settings of RetinaNet [20].

5.1. Comparison with previous works

Comparison with RetinaNet: To make a fair comparison, we align RetinaNet with YOLOF by employing generalized IoU [29] for the box loss, adding an implicit objectness prediction, and applying group normalization layers [37] in heads (as there are only two images per GPU and both BN [12] and SyncBN [41] give poor results in RetinaNet [2], we use GN [37] instead of BN [12] in the heads). The results are presented in Table 1. All '$1\times$' models are trained with a single scale that the shorter side is set as 800 pixels and the longer side is at most 1333 [20]. In the top section, we give RetinaNet baseline results trained with Detectron2 [38]. In the middle section, we present the results of the improved RetinaNet baseline (with a "+"), whose settings are aligned with YOLOF. In the last section, we show results from multiple YOLOF models. Thanks to the single-level feature, YOLOF achieves results *on par with* RetinaNet+ with a 57% flops reduction (flops for each component in YOLOF are shown in Figure 3) and a $2.5\times$ speed up. Due to the large stride (32) of the C5 feature, YOLOF has an inferior performance (-3.1) than RetinaNet+ on small objects. However, YOLOF achieves better performance on large objects (+3.3) as we add dilated residual blocks in the encoder. The comparison between RetinaNet+ and YOLOF with a ResNet-101 [11] show similar evidence as well. Although YOLOF is inferior to RetinaNet+ on small objects when applying the same backbone, it can match small objects' performance with a stronger backbone ResNeXt [39] while running at the same speed. Moreover, to prove that our method is compatible and complementary to current technologies in object detection, we show results that training with multi-scale images and a longer schedule in the last two rows of Table 1. Finally, with the help of multi-scale testing, we obtain our final result of 47.1 mAP and a competitive performance of 31.8 mAP on small objects.

Comparison with DETR. DETR [2] is a recent proposed detector which introduces transformer [36] to object detection. It achieves surprising results on the COCO bench-

[2] https://github.com/facebookresearch/detectron2/blob/master/detectron2/modeling/meta_arch/retinanet.py#L532

Model	Epochs	#params	GFLOPS/FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR [2]	500	41M	86/24*	42.0	62.4	44.2	20.5	45.8	61.1
DETR-R101 [2]	500	60M	152/17*	43.5	**63.8**	46.4	21.9	48.0	**61.8**
YOLOF	72	44M	86/32	41.6	60.5	45.0	22.4	46.2	57.6
YOLOF-R101	72	63M	151/21	**43.7**	62.7	**47.4**	**24.3**	**48.3**	58.9

Table 2. Comparison with DETR on the COCO2017 validation set. We conduct comparisons with backbone ResNet-50 (without suffix) and ResNet-101 (with a suffix R101). To make fair comparison, YOLOF adopts multi-scale training (same as in Table 1) with a '6×' schedule, which is roughly 72 epochs. For the FPS of DETR, $*$ means we follow the method in the original paper [2] and re-measure it on 2080Ti.

Dilated Encoder	*Uniform Matching*	AP	Δ	AP_S	AP_M	AP_L
		21.1	-16.6	8.6	31.1	34.5
✓		29.1	-8.6	9.5	32.2	50.6
	✓	33.8	-3.9	17.7	40.9	43.8
✓	✓	**37.7**	-	**19.1**	**42.5**	**53.2**

Table 3. Effect of *Dilated Encoder* and *Uniform Matching* with ResNet-50. These two components improve the original single-level detector by 16.6 mAP. Note that the result of 21.1 mAP in the table is not a bug. It perform slightly worse than the detectors with SiSo encoders in Figure 1 and Figure 3 due to the design of the decoder in YOLOF - only two convolution layers in the classification head.

mark [21] and proves that by only adopting a single C5 feature, it can achieve comparable results with a multi-level feature detector (Faster R-CNN w/ FPN [19]) for the first time. Given this, one might expect that layers capture global dependencies such as transformer layers [36] are required to achieve promising results in single-level feature detection. *However, we show that a conventional network with local convolution layers can also achieve this goal.* We compare DETR with global layers and YOLOF with local convolution layers in Table 2. The results show that YOLOF matches the DETR's performance, and YOLOF gets more benefits from deeper networks than DETR (w/ ResNet-50 (-0.4) *vs.* w/ ResNet-101 (+0.2)). Interestingly, we find that YOLOF outperforms DETR on small objects (+1.9 and +2.4) while lags behind DETR on large objects (-3.5 and -2.9). The finding is consistent with the local and global discussion above. More importantly, compared with DETR, YOLOF converge much faster ($\sim 7\times$), making it more suitable than DETR to serve as a simple baseline for single-level detectors.

5.2. Ablation Experiments

We run a number of ablations to analyze YOLOF. We first provide an overall analysis of the two proposed components. Then, we show the ablation experiments on detailed designs of each component. Results are shown in Table 3,4 and discussed in detail next.

Dilated Encoder and Uniform Matching: Table 3 shows that both *Dilated Encoder* and *Uniform Matching* are necessary to YOLOF and bring considerable improvements. Specifically, Dilated Encoder has a significant impact on large objects (43.8 *vs.* 53.2) and slightly improves the results of small and medium objects. The results indicate that the *limited scale range* is a severe problem in the C5 feature (Section 4.1). Our Dilated Encoder provides a simple but effective solution to this problem. On the other side, the performance of small and medium objects drops significantly ($\sim 10AP$) without uniform matching, while the large objects' performance is only lightly affected. The finding is consistent with the *imbalance problem on positive anchors* analyzed in Section 4.2. The positive anchors are dominated by large objects, resulting in poor results on small and medium objects. Finally, when we remove both Dilated Encoder and Uniform Matching, a single-level feature detector's performance drops back to $\sim$ 20 mAP like the results in Figure 1 and Figure 3.

Number of ResBlock: YOLOF stacks residual blocks in the SiSo encoder. The results in Table 4a shows that stacking more blocks gives extensive improvements on large objects, which is due to the increment of the feature scale range. Although we observe continuous improvements with more blocks, we choose to add four residual blocks to keep YOLOF simple and neat.

Different dilations: Following the analysis in Section 4.1, to enable the C5 feature to cover large scales, we replace the standard 3×3 convolution layer in the residual blocks with its dilated counterpart. We show the results with different dilations in the residual blocks in Table 4b. Applying dilations to residual blocks bring improvements to YOLOF, while the improvements are saturated when using too large dilations. We conjecture that the reason for this phenomenon is that dilations of $2, 4, 6, 8$ are enough to match object scales in all images.

Add shortcut or not: Table 4c shows that shortcuts play an essential role in Dilated Encoder. The performance of all objects will drop significantly if we remove the shortcuts in residual blocks. According to Section 4.1, shortcuts combine different scale ranges. A largely and densely paved

N	AP	AP_s	AP_m	AP_l
0	33.8	17.7	40.9	43.8
2	34.9	17.8	41.3	46.8
4	**35.5**	**17.6**	**41.4**	**48.4**
6	36.0	17.7	41.9	49.5
8	36.6	18.5	42.0	50.7
10	36.9	18.3	42.4	50.4

(a) **Number of ResBlocks** (ResNet-50): More residual blocks bring more gains. *N* represent the number of ResBlocks. To keep YOLOF simple and neat, we add 4 blocks in the encoder by default.

Dilations	AP	AP_s	AP_m	AP_l
1,1,1,1	35.5	17.6	41.4	48.4
2,2,2,2	36.4	18.1	41.8	50.2
3,3,3,3	36.9	18.4	42.1	51.0
1,2,3,4	37.4	18.6	42.6	51.8
2,4,6,8	**37.7**	**19.1**	**42.5**	**53.2**
3,6,9,12	37.3	18.7	42.1	52.6

(b) **Different dilations** (ResNet-50-N4): 'N4' means we add 4 ResBlocks in the encoder. Dilation in the residual block gives large gains on large objects and slightly improve the performance of small and medium objects.

Dilations & Shortcut	AP	AP_s	AP_m	AP_l
2,4,6,8 ✓	**37.7**	**19.1**	**42.5**	**53.2**
2,4,6,8 -	34.1	16.2	38.4	47.5
1,1,1,1 ✓	35.5	17.6	41.4	48.4
1,1,1,1 -	32.6	15.0	38.4	44.2

(c) **Add shortcut or not** (ResNet-50): YOLOF results with shortcuts or not on various dilation settings. Shortcut brings considerable gains on all object scales and becomes more important when the dilations are adopted (+3.6 AP with dilations 2,4,6,8 vs. +2.9 AP when dilations are all ones).

topk	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
top1	35.9	55.6	38.4	17.5	40.3	50.2
top2	37.2	56.7	39.9	18.9	41.6	52.0
top3	37.5	**57.1**	40.2	18.6	41.9	52.5
top4	**37.7**	56.9	**40.6**	**19.1**	**42.5**	**53.2**
top5	37.5	56.7	40.3	18.1	42	**53.2**

(d) **Number of positives** (ResNet-50-N4): Number of positive anchors in *Uniform Matching*. Increase the positive anchor for each ground-truth box can improve the performance while it saturates when too many positive anchors. We choose the top4 anchors in YOLOF which achieves best results.

Matching Methods	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Max-IoU Matching [20]	29.1	45.9	29.6	9.5	32.2	50.6
ATSS(topk=9) [42]	34.6	54.3	37.1	17.7	40.6	46.9
ATSS(topk=15) [42]*	36.5	55.9	38.6	18.1	41.4	50.8
Hungarian Matching [2]	35.8	55.5	38.3	18.2	39.9	50.2
Uniform Matching	**37.7**	**56.9**	**40.6**	**19.1**	**42.5**	**53.2**

(e) **Uniform matching *vs*. other matchings** (ResNet-50-N4): Comparison with other matching methods. Uniform Matching achieve balance in positive anchors and get the best results among other matching methods, which is consistent with the comparison in Figure 6. Note that '*' represents that we get the best result for ATSS [42] when setting topk as 15.

Table 4. **Ablations**. We show ablation experiments for *Dilation Encoder* and *Uniform Matching* on COCO2017 val set with ResNet-50.

scale range covered by the feature is the critical factor for detecting all objects in a single-level feature manner.

Number of positives: A comparison among the number of induced positive anchors by ground-truth boxes is conducted in Table 4d. Intuitively, more positive anchors can achieve better performance as the learning will be easier when given more samples. Thus, in our uniform matching manner, we empirically increase the number of positive anchors induced by each ground-truth box. As shown in Table 4d, the hyper-parameter k is very robust for the performance when k is larger than 1, which may suggest that the most important is the uniform matching manner in YOLOF. We set $top4$ for our uniform matching as it is the best choice according to the results.

Uniform matching *vs*. other matchings: We compare the uniform matching with other matching strategies for YOLOF and show results in Table 4e. The proposed uniform matching strategy can achieve the best results, compatible with the imbalance analysis in Figure 6. It worth noting that the Hungarian matching strategy can be roughly treated as Top1 matching (Table 4d) so that they get similar performance. The difference between them is that an anchor will only match one object in Hungarian matching while the Top1 matching does not have this constraint, and the experiments show that this is not important. The original ATSS find that top9 anchors are the best choice, while we find top15 anchors are much better in the single-level feature detector. By using top15 anchors, ATSS achieves a good result of 36.5 mAP while still lags behind our uniform matching by a 1.2 mAP gap.

6. Conclusion

In this work, we identify that the success of FPN is due to its divide-and-conquer solution to the optimization problem in dense object detection. Given that FPN makes network structure complex, brings memory burdens, and slows down the detectors, we propose a simple but highly efficient method without using FPN to address the optimization problem differently, denoted as YOLOF. We prove its efficacy by making fair comparisons with RetinaNet and DETR. We hope our YOLOF can serve as a solid baseline and provide insight for designing single-level feature detectors in future research.

Acknowledgements

This work is supported by The National Key Research and Development Program of China (No. 2017YFA0700800), Beijing Academy of Artificial Intelligence (BAAI), National Natural Science Foundation of China (No.61972396, 61876182, 61906193), National Key Research and Development Program of China (No. 2020AAA0103402), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDB32050200), and The NSFC-General Technology Collaborative Fund for Basic Research (Grant No. U1936204).

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 1

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020. 2, 5, 6, 7, 8

[3] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4974–4983, 2019. 6

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2

[5] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019. 2

[6] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009. 2

[7] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 1, 2

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 4, 5, 6

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4, 6

[13] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. In *ECCV*, 2020. 5

[14] Tao Kong, Fuchun Sun, Chuanqi Tan, Huaping Liu, and Wenbing Huang. Deep feature pyramid reconfiguration for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 169–185, 2018. 1, 2

[15] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 5

[16] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 2

[17] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 6054–6063, 2019. 2, 4

[18] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: Design backbone for object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 334–350, 2018. 4

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 2, 4, 7

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2, 3, 4, 5, 6, 8

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5, 7

[22] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. 1, 2

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 4

[25] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 821–830, 2019. 1

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2, 5

[27] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2, 5

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[29] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box

regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019. 6

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[31] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018. 2

[32] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in neural information processing systems*, pages 9310–9320, 2018. 2

[33] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016. 5

[34] 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*, pages 10781–10790, 2020. 1, 2

[35] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 9627–9636, 2019. 1, 2

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 6, 7

[37] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 6

[38] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 3, 6

[39] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5, 6

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 4

[41] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018. 6

[42] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020. 2, 5, 8

[43] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. FreeAnchor: Learning to match anchors for visual object detection. In *Neural Information Processing Systems*, 2019. 5

[44] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–284, 2018. 2

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

[46] 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. 5