Construct Effective Geometry Aware Feature Pyramid Network for Multi-Scale Object Detection

Jinpeng Dong[1], **Yuhao Huang**[1], **Songyi Zhang**[1], **Shitao Chen**[1], **Nanning Zheng**[1]*
[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
{djp1235a,hyh950623,zhangsongyi,chenshitao}@stu.xjtu.edu.cn, nnzheng@xjtu.edu.cn

Abstract

Feature Pyramid Network (FPN) has been widely adopted to exploit multi-scale features for scale variation in object detection. However, intrinsic defects in most of the current methods with FPN make it difficult to adapt to the feature of different geometric objects. To address this issue, we introduce geometric prior into FPN to obtain more discriminative features. In this paper, we propose the Geometry-aware Feature Pyramid Network (GaFPN), which mainly consists of the novel Geometry-aware Mapping Module and Geometry-aware Predictor Head. The Geometry-aware Mapping Module is proposed to make full use of all pyramid features to obtain better proposal features by the weight-generation subnetwork. The weights generation subnetwork generates fusion weight for each layer proposal features by using the geometric information of the proposal. The Geometry-aware Predictor Head introduces geometric prior into predictor head by the embedding generation network to strengthen feature representation for classification and regression. Our GaFPN can be easily extended to other two-stage object detectors with feature pyramid and applied to instance segmentation task. The proposed GaFPN significantly improves detection performance compared to baseline detectors with ResNet-50-FPN: +1.9, +2.0, +1.7, +1.3, +0.8 points Average Precision (AP) on Faster-RCNN, Cascade R-CNN, Dynamic R-CNN, SABL, and AugFPN respectively on MS COCO dataset.

Introduction

As one of the most important computer vision tasks, object detection serves as a fundamental task for several high level applications such as panoptic segmentation (Kirillov et al. 2020) and instance segmentation (He et al. 2017). Following the advance of Convolutional Neural Networks (CNN) and benchmarks, the performance of CNN-based object detectors has been greatly improved. Among these detectors, Feature Pyramid Networks (FPN) (Lin et al. 2017a) is a classic and effective method for multi-scale object detection. It first constructs a feature pyramid and then selects one single level feature for each proposal by heuristic-guided mapping mechanism. Finally, the features of each proposal fed into the predictor head by the RoI Align layer (He et al. 2017).

*Corresponding author.

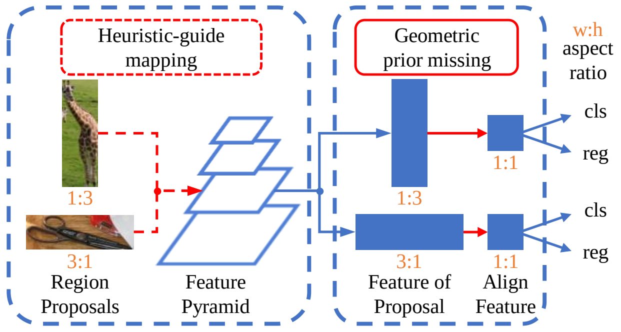

Figure 1: Defects in feature pyramid network.

Despite the performance of multi-scale object detection substantially improved by FPN, its potential has not been fully explored, as shown in Fig. 1. We summarize two potential promotions as follows:

1) Geometry-guided proposal mapping mechanism. Object proposals are assigned to a certain level of feature map by heuristic rules, such as the area of proposals. Although the areas of object proposals are similar, the aspect ratio of them may be significantly different. For example, a 1000×10 object and a 10×1000 object are assigned to the same feature map. The problem leads to a major flaw that the optimal feature may be hard to obtain based on heuristic-guided. Meanwhile, recent studies (Liu et al. 2018; Guo et al. 2020) report that selecting only single-layer feature may generate sub-optimal results, since this way ignores the valid information of unselected features from other layers. The geometry-guided proposal mapping mechanism by employing multi-layer features from the feature pyramid may alleviate these problems.

2) Geometric feature embedding. The geometric information of proposals is key prior knowledge for classification and location refinement. On the one hand, the extra geometric prior feature may facilitate the classification learning of two semantically similar proposals to distinguish them easily. On the other hand, the regression value may be dynamically changed according to the extra geometric prior feature. However, after feature mapping, each object proposal is extracted 7×7 features by RoI Align (He et al. 2017) layer. Then the extracted features are fed into the detector head consisting of fully-connected layers to be classified and regressed. Through the above processes,the geometric information of the proposal is largely ignored and damaged. So, how to embed geometric features into features of the proposal is an urgent problem to generate more accurate results.

These promotions motivate us to propose the Geometry-aware Feature Pyramid Network (GaFPN), a simple yet effective feature pyramid network, to improve feature representation for multi-scale object detection. First, the Geometry-aware Mapping Module makes full use of all pyramid features to obtain better proposal features and reduce the impact of heuristic-guided mapping by the weight-generation subnetwork. The weights generation subnetwork generates fusion weights for each proposal in different layers by using the geometric information of the proposal. Second, the Geometry-aware Predictor Head is utilized to introduce geometric features into the classification and regression branch by the embedding generation network. It deals with the geometric prior missing problem and generate more discriminative features. The embedding generation network generates geometric embedding values for each proposal by using the geometric information of the proposal, then the geometric embedding value embed into features of the proposal by multiplication operation.

Experimental results on MS COCO (Lin et al. 2014) dataset show that our GaFPN based Faster R-CNN (Ren et al. 2015) outperforms baseline detectors with ResNet-50-FPN: GaFPN improves the detection Average Precision (AP) by +1.9, +2.0, +1.7, +1.3, +0.8 points on Faster-RCNN (Ren et al. 2015), Cascade R-CNN (Cai and Vasconcelos 2018), Dynamic R-CNN (Zhang et al. 2020), SABL (Wang et al. 2020), and AugFPN (Guo et al. 2020), respectively.

Our contributions are three-fold: (1) We systematically revisit the FPN detectors. Our study reveals two defects that limit the detection performance. (2) We propose a new feature pyramid network named GaFPN to address these problems by combining new components: Geometry-aware Mapping Module, Geometry-aware Predictor Head. (3) We verify the proposed GaFPN equipped with various detectors, backbones and tasks on MS COCO, and it consistently obtains significant improvements over FPN-based detectors.

Related Work

Object Detectors

Benefit from CNN, object detectors have achieved dramatic improvements in recent years. CNN-based detectors can be divided into two types: two-stage and one-stage. R-CNN (Girshick et al. 2014) was first employed as a two-stage detector. To achieve end-to-end detector, SPP (He et al. 2015), Fast R-CNN (Girshick 2015) and Faster R-CNN (Ren et al. 2015) were gradually proposed. Faster R-CNN introduces Region Proposal Network (RPN), a novel proposal generator, to replace traditional methods, then develop an end-to-end detector. FPN (Lin et al. 2017a) introduces feature pyramid architecture to tackle the scale variation. Cascade R-CNN (Cai and Vasconcelos 2018) introduces a classic cascade architecture and becomes a multi-stage detector. DCN (Dai et al. 2017; Zhu et al. 2019) introduces 2d offsets into standard sampling for accurate object detection. Dynamic RCNN (Zhang et al. 2020) proposed the dynamic training strategy based on the statistics of proposals.

In addition, one-stage detectors such as YOLO (Redmon et al. 2016; Redmon and Farhadi 2017, 2018) and SSD (Liu et al. 2016) be widely used for their high speed. To improve the accuracy of one-stage detectors, RetinaNet (Lin et al. 2017b) utilizes a novel focal loss which is a flexible manner to solve the extreme class imbalance problem and introduces the feature pyramid architecture into the backbone.

Mapping Strategy

Selecting the appropriate feature in FPN for each proposal is a key problem. FASF (Zhu, He, and Savvides 2019) makes the assignment by dynamically selecting the pyramid feature based on the minimal instance loss level during training. PANet (Liu et al. 2018) makes the mapping strategy which selects features of all pyramid levels for each proposal to fed into fully-connected layers independently and fuses them by the element-wise maximize operation. To better-exploited features from different levels, AugFPN (Guo et al. 2020) fuses features from all levels according to the learned weights for the two-stage detector. There is a distinct difference that we propose a geometry-aware strategy to obtain weights according to the external abstract geometric information of the proposal rather than the convolution features of the proposal itself between these methods and our work.

Predictor Heads

Many predictor heads to improve classification and localization accuracy have been proposed in recent years. Cascade R-CNN (Cai and Vasconcelos 2018) employs multi-stage R-CNN heads with different Intersection over Union (IoU) thresholds stage-by-stage to obtain more accurate results. Fitness NMS (Tychsen-Smith and Petersson 2018) designs subnetworks to predict the probabilities of localization. IoU-Net (Jiang et al. 2018) propose an IoU prediction module to predict the IoU for each proposal. Both of the latter two methods want to optimize the classification confidence according to the location quality (IoU), but the prediction of IoU is difficult and needs to introduce a complex structure. Another line of effort aims to decouple the classification and localization. Double-Head R-CNN (Wu et al. 2020) employs a fully-connected head for classification and a convolution head for regression. TSD (Song, Liu, and Wang 2020) decouples the classification and localization from proposals and feature extractors. Different from them, our method embeds geometric features into the predictor head to compensate for the lack of geometric information caused by the RoI Align layer (He et al. 2017).

Method

In this section, we introduces the Geometry-aware Feature Pyramid Network (GaFPN). Our framework is shown in Figure 2. GaFPN consists of three components: Geometry-aware Mapping Module (GMM), Geometry-aware Predictor Head (GPH) and Feature Augmentation Pyramid (FAP).

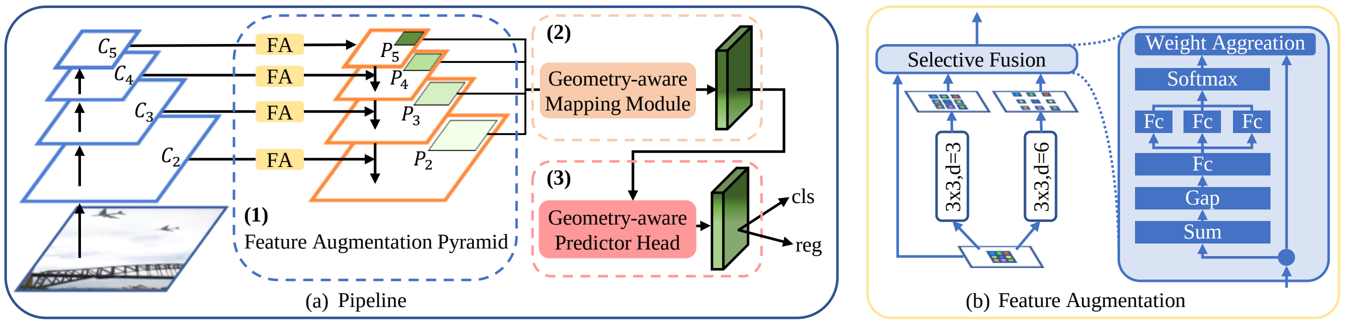

Figure 2: (a) is overall pipeline of GaFPN based detector. (1)-(3) are three main components of GaFPN: Feature Augmentation Pyramid, Geometry-aware Mapping Module, and Geometry-aware Predictor Head. (b) is the details of Feature Augmentation.

First, intrinsic feature hierarchy in the backbone are denoted as $\{C_2, C_3, C_4, C_5\}$. Then these features reduced to the same channel by a 1×1 conv are denoted as $\{M_2, M_3, M_4, M_5\}$. The same channel features employ Feature Augmentation inspired by ASPP (Chen et al. 2017) and attention mechanism (Hu, Shen, and Sun 2018; Woo et al. 2018; Li et al. 2019) to strength feature and construct the Feature Augmentation Pyramid. The new generated features are denoted as $\{P_2, P_3, P_4, P_5\}$. Finally, the new features are fed into the GMM and the GPH to be classified and regressed. The GMM fuses multi-layer features of the proposal adaptively according to geometric information. The GPH embeds geometric information into features of the proposal. Two components of GaFPN will be described in detail below.

Geometry-aware Mapping Module

The feature level for each proposal is mapped based on the area of proposals in the conventional FPN. It may produce sub-optimal results. Meanwhile, features from other levels may be beneficial for object classification or regression. These findings motivates us to explore a geometry-guided proposal mapping mechanism and utilize multi-level features to obtain better proposal features.

We propose the Geometry-aware Mapping Module (GMM), which adaptively utilizes each pyramid level of features to enrich feature representation instead of using only one level of features guided by heuristics. The adaptive mechanism is realized by employing the weights generation subnetwork to generate the weight for each level. The subnetwork directly takes the geometric information of proposals as input.

Geometric feature generating: We first normalize the coordinates (x_1, y_1, x_2, y_2) of each proposal, which is computed as follows:

$$x_1' = \frac{x_1}{W},\ y_1' = \frac{y_1}{H},\ x_2' = \frac{x_2}{W},\ y_2' = \frac{y_2}{H}. \tag{1}$$

where W, H denote the size of train image, which is used as a normalization term.

Then we use the normalized coordinates to generate additional geometric features such as width, height, aspect ratio (r), and area (a), which is computed as follows:

$$w = x_2' - x_1',\ h = y_2' - y_1',\ r = \frac{w}{h},\ a = w \times h. \tag{2}$$

Finally, these features are concatenated together as the input $\mathbf{X} \in \mathbb{R}^8$ for the weight generation subnetwork, which can be formulated as follows:

$$\mathbf{X} = \mathrm{CAT}(w, h, r, a, x_1', y_1', x_2', y_2') \tag{3}$$

Weight calculation: After the geometric feature generating stage, we calculate weights for each proposal in each layer feature pyramid. We first use the weight generation subnetwork which is only consists of fully connected layers and ReLU to transform the geometric feature. The geometric feature representation is enhanced through transformation. Then the transformed feature is used to generate different weights for different levels by a sigmoid operator, which is defined as follows:

$$\begin{aligned}\mathbf{w} &= \sigma(\mathrm{MLP}_{\mathrm{WG}}(\mathbf{X})) \\ &= \sigma(\delta\mathbf{W_4}(\delta\mathbf{W_3}(\delta\mathbf{W_2}(\delta\mathbf{W_1}(\mathbf{X}))))) \end{aligned} \tag{4}$$

where δ denotes the Rectified Linear Unit (ReLU), σ is the Sigmoid function. The $\mathrm{MLP}_{\mathrm{WG}}(\cdot)$ is the weight generation subnetwork. The $\mathbf{W_1}$ is a dimension increasing layer, and the $\mathbf{W_2}, \mathbf{W_3}, \mathbf{W_4}$ is a dimension reduction layer. The output weights $\mathbf{w} = [w_1, w_2, w_3, w_4] \in \mathbb{R}^4$. The detailed structure of the weight generation subnetwork is illustrated in Figure 3 (a).

Feature aggregation: Finally, we obtain an integrated feature $\mathbf{V}$ by the adaptive multi-level weighted aggregation for each proposal. The multi-level proposal feature extracted by the RoI Align layer are denoted as $\{P_2', P_3', P_4', P_5'\}$, and the integrated feature is computed as follows:

$$\mathbf{V} = \sum_{i=2}^{5} w_{i-1} \cdot P_i' \tag{5}$$

where $\mathbf{V} \in \mathbb{R}^{7\times7\times256}$, $P_i' \in \mathbb{R}^{7\times7\times256}$.

In this way, we make full use of pyramid features to enrich feature representation instead of using only one level of features guided by heuristics. The GMM softens the rigid heuristic mapping strategy into an adaptive strategy that can be jointly trained with the detector by back-propagation.

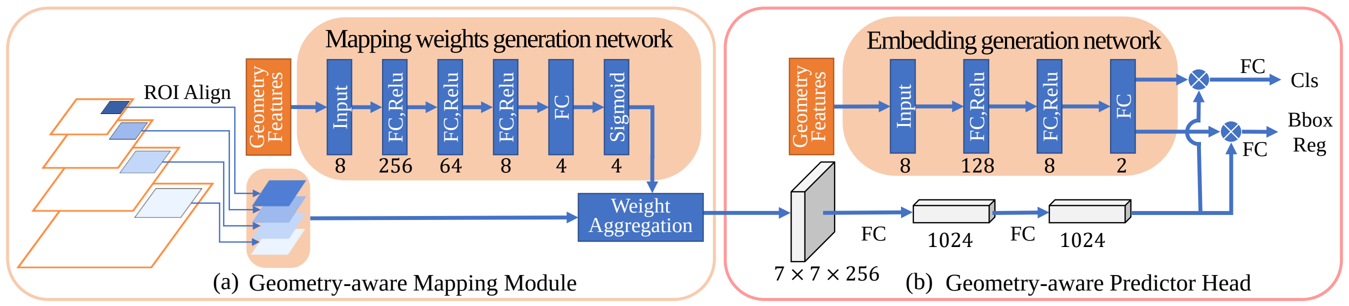

Figure 3: (a) is the process of fusing different levels proposal features for Geometry-aware Mapping Module. (b) is the process of Geometry-aware Predictor Head.

Geometry-aware Predictor Head

In FPN, each proposal is extracted as a fixed size 7×7×256 feature by the RoI Align (He et al. 2017) layer and fed into the predictor head. Through the above processes, the geometric information is largely ignored and damaged.

To address this issue, we proposed the Geometry-aware Predictor Head (GPH), which learns to generate geometric aware features by the embedding generation subnetwork. The geometric aware features compensates for the lack of geometric information caused by the RoI Align layer. Furthermore, to alleviate the side effects of classification and regression sharing features, different embedding features are generated for different tasks to decouple classification and regression.

Specifically, the embedding generation subnetwork is similar to the mapping weights generation subnetwork of the GMM. The geometric features in Eq. (5) are used as the input $\mathbf{X}$ to obtain embedded features.

Embedding value generating: After obtaining the geometric feature, we first use the embedding generation subnetwork to generate embedding value for each proposal. The embedding generation subnetwork is only consists of fully connected layers and ReLU. The embedding value is defined as follows:

$$\begin{aligned}\mathbf{E} &= \mathrm{MLP}_{\mathrm{EG}}(\mathbf{X}) \\ &= \mathbf{W_3}(\delta \mathbf{W_2}(\delta \mathbf{W_1}(\mathbf{X}))) \end{aligned} \quad (6)$$

where δ denotes the Rectified Linear Unit (ReLU), $\mathrm{MLP}_{\mathrm{EG}}(\cdot)$ is the embedding generation subnetwork. $\mathbf{W_1}$ is a dimension increasing layer, and $\mathbf{W_2}, \mathbf{W_3}$ is a dimension reduction layer. The output embedding value $\mathbf{E} = [E_1, E_2] \in \mathbb{R}^2$.

Feature embedding: Finally, the embedded features $\mathbf{U}'_{\mathbf{cls}}, \mathbf{U}'_{\mathbf{reg}}$ are generated by combining embedding value and features from the classification and regression branch $(\mathbf{U_{cls}}, \mathbf{U_{reg}})$, which is computed as follows:

$$\mathbf{U}'_{\mathbf{cls}} = E_1 \cdot \mathbf{U_{cls}} \quad (7)$$

$$\mathbf{U}'_{\mathbf{reg}} = E_2 \cdot \mathbf{U_{reg}} \quad (8)$$

where $\mathbf{U_{cls}}, \mathbf{U_{reg}}, \mathbf{U}'_{\mathbf{cls}}, \mathbf{U}'_{\mathbf{reg}} \in \mathbb{R}^{1024}$.

The detailed structure of the embedding generation subnetwork is illustrated in Figure 3 (b). It is worth noting that the embedding generation subnetwork is not activated by the sigmoid at the end, which is different from the weighted generation subnetwork. The GPH is jointly trained with the detector by back-propagation. In this way, geometric features are embedded into proposal features, which enriches feature representation and improves classification and regression performance.

Experiments

Dataset and Evaluation Metrics

All experiments are performed on the MS COCO (Lin et al. 2014) 2017 dataset with 80 object categories. It consists of 115k training images (train-2017), 5k validation images (val-2017) and 20k testing images (test-dev). The model training is conducted on the train-2017 sets, and the evaluation is performed on the val-2017 or test-dev sets. All reported results follow standard COCO-style Average Precision (AP) metrics that include AP (averaged over different IoUs), AP_{50} (AP for IoU 0.5), AP_{75} (AP for IoU 0.75). We also include AP_S, AP_M and AP_L, which correspond to the results on small, medium, and large area respectively.

Implementation Details

For fair comparisons, we conduct all experiments based on PyTorch, and mmdetection (Chen et al. 2019). ResNet (He et al. 2016) which is pretrained on the ImageNet (Russakovsky et al. 2015) is used as backbone. The shorter side of input images is resized to 800 pixels, and the maximum size is less than 1333 pixels. By default, the models are trained on 4 NVIDIA RTX 2080Ti GPUs (2 images per GPU) for 12 epochs, known as 1× schedule. The 2× means doubling the total training epochs and learning rate schedules accordingly. We initialize the learning rate as 0.02 and decreased by a factor of 0.1 at 8th-epoch and 11th-epoch. We use Stochastic Gradient Descent (SGD) (Krizhevsky, Sutskever, and Hinton 2012) as an optimizer. Only horizontal flipping augmentation is used for training. All other hyper-parameters in this paper follow the settings in mmdetection (Chen et al. 2019) if not specifically noted.

Main Results

In this section, we compare GaFPN with other state-of-the-art object detection approaches on the COCO test-dev set. For fair comparisons, we report our re-implemented results

Method	Backbone	Schedule	AP	AP$_{50}$	AP$_{75}$	AP$_{S}$	AP$_{M}$	AP$_{L}$
Faster R-CNN*	ResNet-50	$1\times$	35.9	57.4	38.7	20.9	38.9	43.9
Faster R-CNN*	ResNet-101	$1\times$	37.9	59.3	41.3	21.9	41.0	46.9
Faster R-CNN*	ResNet-101	$2\times$	39.3	60.8	42.9	22.3	42.5	49.2
Faster R-CNN*	ResNext-101	$1\times$	39.8	61.5	43.3	23.8	43.2	48.8
Faster R-CNN†	ResNet-50	$1\times$	37.7	58.9	40.7	21.9	40.7	46.5
Cascade R-CNN*	ResNet-50	$1\times$	40.2	58.2	43.6	23.0	42.6	50.5
Cascade R-CNN*	ResNet-50	$2\times$	41.3	59.6	44.8	23.3	43.7	52.4
Dynamic R-CNN*	ResNet-50	$1\times$	38.3	56.8	42.1	21.4	40.7	48.4
SABL*	ResNet-50	$1\times$	39.3	57.3	42.2	22.4	42.3	48.5
AugFPN*	ResNet-50	$1\times$	38.3	60.5	41.6	23.0	41.4	47.3
Mask R-CNN*	ResNet-50	$1\times$	36.9{33.9}	58.2{55.0}	39.9{36.0}	21.6{18.3}	39.6{36.3}	45.4{43.9}
Mask R-CNN*	ResNet-101	$1\times$	39.0{35.6}	60.3{57.2}	42.6{38.1}	22.5{18.9}	42.1{38.4}	48.3{46.4}
Mask R-CNN*	ResNet-101	$2\times$	40.3{36.6}	61.5{58.5}	44.1{39.4}	22.9{19.3}	43.5{39.5}	50.6{47.9}
Faster R-CNN**	ResNet-50	$1\times$	37.8[**+1.9**]	60.3	40.7	22.5	40.7	46.5
Faster R-CNN**	ResNet-101	$1\times$	39.4[**+1.5**]	61.8	43.1	23.0	42.6	49.2
Faster R-CNN**	ResNet-101	$2\times$	40.6[**+1.3**]	63.0	44.1	23.3	43.8	51.0
Faster R-CNN**	ResNext-101	$1\times$	40.8[**+1.0**]	63.3	44.5	24.3	44.1	50.7
Faster R-CNN†**	ResNet-50	$1\times$	39.4[**+1.7**]	61.2	42.6	22.9	42.4	49.2
Cascade R-CNN**	ResNet-50	$1\times$	42.2[**+2.0**]	61.2	45.8	24.4	45.0	53.4
Cascade R-CNN**	ResNet-50	$2\times$	43.1[**+1.8**]	62.2	46.8	24.6	45.6	54.9
Dynamic R-CNN**	ResNet-50	$1\times$	40.0[**+1.7**]	59.2	43.8	22.8	42.5	50.9
SABL**	ResNet-50	$1\times$	40.6[**+1.3**]	58.9	43.7	23.4	43.5	50.6
AugFPN**	ResNet-50	$1\times$	39.1[**+0.8**]	61.5	42.5	23.4	42.3	48.6
Mask R-CNN**	ResNet-50	$1\times$	38.5[**+1.6**]{35.3[**+1.4**]}	60.7{57.3}	41.7{37.5}	22.6{18.9}	41.2{37.8}	47.8{46.1}
Mask R-CNN**	ResNet-101	$1\times$	40.3[**+1.3**]{36.8[**+1.2**]}	62.5{59.1}	43.9{39.3}	23.2{19.5}	43.5{39.7}	50.4{48.3}
Mask R-CNN**	ResNet-101	$2\times$	41.7[**+1.4**]{37.9[**+1.3**]}	63.9{60.6}	45.5{40.5}	24.2{20.2}	45.0{40.8}	52.6{50.3}

Table 1: Comparison with the single-model and single-scale state-of-the-art methods evaluated on COCO test-dev set. The symbol '*' means the results from our re-implementation. The symbol '**' means the results from our method. The symbol† means the results based on mmdet2.4. For Mask R-CNN, the results in { } stands for the corresponding mask results. The bold numbers means the relative improvement. The $1\times$, $2\times$ training schedule follows the setting as Detectron (Girshick et al. 2018).

of the corresponding baseline methods equipped with FPN. All results are shown in Table 1. Applying the GaFPN on Faster R-CNN with ResNet-50 achieves 37.8 AP, which is 1.9 points higher than Faster R-CNN based on ResNet50-FPN. Besides, the GaFPN can consistently improve performance even with more powerful backbone networks and longer iteration steps. For example, when using ResNet101 and ResNext101-32x4d as the backbone, our method still improves the performance by 1.5 and 1.0 AP, respectively. In the $2\times$ training scheme, our method with ResNet101 still improves the performance by 1.3 AP. These results show that GaFPN significantly improves the performance of the conventional FPN-based Faster R-CNN.

Next, we evaluate the effectiveness of GaFPN on several variants of Faster R-CNN (Ren et al. 2015) detector, i.e., Cascade R-CNN (Cai and Vasconcelos 2018), Dynamic R-CNN (Zhang et al. 2020), SABL (Wang et al. 2020), AugFPN (Guo et al. 2020). Considering a multi-stage predictor head in Cascade R-CNN, each predictor head is replaced by our Geometry-aware Predictor Head (GPH). As shown in Table 1, when using ResNet50 as the backbone, Cascade R-CNN can be improved by 2.0 AP and 1.8 AP in the $1\times$ or $2\times$ training scheme. Meanwhile, Dynamic R-CNN is boosted to 40.0 AP from 38.3 AP when replacing FPN with GaFPN. Since SABL (Wang et al. 2020) employ SABL Head which is very different from traditional FPN-based predictor head, the GPH is not included. SABL with GaFPN achieves 40.6 AP, which is 1.3 points higher than the SABL baseline. The result shows that the other component still strengthen the feature representation even without the GPH. Even though AugFPN is a very strong FPN variant, our method can still improve 0.8 AP when introducing GaFPN into AugFPN. The result shows that our GaFPN and AugFPN are complementary and solve different problems.

Generality to Instance Segmentation. Finally, we verify the performance of GaFPN for the instance segmentation task. By introducing GaFPN into Mask R-CNN (He et al. 2017) instead of FPN, Mask R-CNN using ResNet50 as the backbone improves the performance by 1.6 AP on the detection and 1.4 AP on instance segmentation. When using ResNet101 as the feature extractor, the performance of Mask R-CNN with GaFPN is boosted by 1.3 AP on the detection and 1.2 AP on instance segmentation, respectively. In the $2\times$ training scheme, our method based on Mask R-CNN with ResNet101 still improves the performance by 1.4 AP on the detection and 1.3 AP on instance segmentation, respectively. As shown in Table 1, experiments on different backbones, detectors, and even different tasks show that GaFPN can obtain consistent performance improvement. These results fully demonstrates the robustness and generalization ability of GaFPN. We believe that the proposed method can also be applied to other computer vision tasks.

GMM	GPH	FAP	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
			35.6	57.0	38.5	21.1	39.0	45.6
✓			36.2	57.9	39.1	20.7	40.0	46.4
	✓		36.1	57.6	39.0	20.9	39.8	46.5
✓	✓		36.9	58.7	40.0	21.6	40.4	47.9
✓	✓	✓	37.3	59.7	40.4	23.0	41.2	48.1

Table 2: Ablative experiments for our GaFPN on the COCO val2017. We study the effect of GMM: Geometry-aware Mapping Module, GPH: Geometry-aware Predictor Head, FAP: Feature Augmentation Pyramid.

Ablation Study

In this section, extensive ablation experiments are performed to demonstrate the effectiveness of each proposed component in our method.

Ablation studies on the importance of each component. To demonstrate the effects of each component in GaFPN, we gradually apply Geometry-aware Mapping (GMM), Geometry-aware Predictor Head (GPH), and Feature Augmentation Pyramid (FAP) on ResNet-50 FPN Faster R-CNN (baseline). Simultaneously, we also introduce the improvement brought by the combination of different components, which shows that these components are complementary to each other. All experiment results are reported in Table 2.

As shown in Table 2, the GMM obtains 0.6 AP absolute gains over the baseline. It can be seen that most of the final improvements are from AP_M (+1.0 AP) and AP_L (+0.8 AP), which means that the GMM makes the large proposal which originally mapped to single higher-level features can obtain spatial information by combined with lower-level features.

The GPH improves the detection performance from 35.6 to 36.1 AP. The benefits from that the GPH obtain more discriminative semantic features of the lager proposal by embedding the geometric information of proposals.

We also analyze the detection performance of combining two components. For example, the GMM and the GPH together can obtain 1.3 AP improvement. It is worthy to note that the two components introduce few extra parameters. Finally, When combining three components, the detection performance is boosted to 37.3 AP with 1.7 AP improvement from 35.6 AP, especially on the AP_{50}, we get a 2.7 AP improvement. To sum up, these results show that the three components are complementary and deal with different problems.

Ablation studies on Geometry-aware Mapping Module. The results of ablative experiments on Geometry-aware Mapping Module (GMM) are shown in Table 3. We first study the influence on the number of features in the weight generation subnetwork. As shown in Table 3, single-use of area or aspect ratio features can bring 0.4 AP performance improvement. Combining the features of area and aspect ratio can improve the performance by 0.6 AP. The combination of area, aspect ratio, and coordinate features can also improve the performance by 0.6 AP. Although the features of coordinates does not improve the performance for this dataset, considering that the overall feature dimension is not high and the features of coordinates may be beneficial to other datasets, our final model employs a combination of the area, aspect ratio, and coordinates features as the input features in the weight generation subnetwork. Next, we study different methods of fusing multi-level features. As shown in Table 3, we can find that sum fusion and max fusion both improve 0.2 AP than the baseline method. Sum fusion means that the fused features are all levels of features summation. Max fusion means that the fused features are the maximum of all levels of features. The idea of max fusion originated from adaptive pooling in PANet (Liu et al. 2018). Because our main purpose is to explore the fusion method with less resource cost. The difference between max fusion in this experiment and adaptive pooling is that we did not employ extra fully-connected layers to adapt the proposal feature before fusion. The GMM achieves 36.2 AP, which is 0.4 points higher than the two fusion methods. Our performance are comparable to those of the Adaptive Spatial Fusion (ASF) (Guo et al. 2020). However, the parameters and FLOPs of ASF are larger than those of our method, as can be seen from Table 6. These results indicate that the GMM can adaptively learn fusion weights and produce more powerful features of the proposals under low complexity.

Quantitative analysis of the weight. To analyze the fusion weights at different levels generated by the GMM, we map proposals on COCO val2017 into four levels based on heuristic-guided mapping strategy. For each proposal, we first obtain four weights corresponding to four feature levels. Then, we calculate the average weight of the proposals at each level. The results corresponding to four pyramid lev-

Setting	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	35.6	57.0	38.5	21.1	39.0	45.6
area	36.0	57.6	38.7	20.9	39.9	46.6
aspect ratio	36.0	57.6	38.6	20.8	40.0	46.1
area+aspect ratio	36.2	57.7	38.8	21.2	40.0	46.2
area+aspect ratio+coord.	36.2	57.9	39.1	20.7	40.0	46.4
sum	35.8	57.4	38.6	21.1	39.8	45.2
max	35.8	57.2	38.5	20.6	39.4	45.6
Adaptive Spatial Fusion	36.2	58.2	39.0	21.0	40.2	46.7

Table 3: Ablative experiments of Geometry-aware Mapping Module on the COCO val2017. area means the area of the proposal. aspect ratio means the width, length and aspect ratio of the proposal. coordinates means the coordinates of the proposal. sum and max means fusion type.

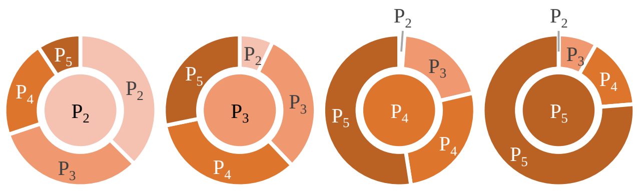

Figure 4: Multi-level aggregation weights for Geometry-aware Mapping Module. The figures from left to right correspond to the proposals originally assigned to $P_2 - P_5$.

Setting	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	35.6	57.0	38.5	21.1	39.0	45.6
cls	35.9	57.6	38.7	20.9	39.9	46.6
reg	35.5	56.9	38.5	20.8	39.0	45.1
cls+reg	36.0	57.5	38.7	21.1	39.5	45.8
cls_reg	36.1	57.6	39.0	20.9	39.8	46.5
cls+reg*	35.6	57.0	38.3	20.9	39.1	44.8

Table 4: Ablative experiments of Geometry-aware Predictor Head on the COCO val2017. cls: classification, reg: regression. cls+reg means that classification and regression branches share features. cls_reg means no share features. cls+reg* means that feature embedding by summation.

Setting	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
PAFPN*	35.7	57.2	38.5	20.8	39.3	45.9
PAFPN+GaFPN	37.3	59.4	40.2	23.0	41.0	47.4
BiFPN*	36.0	55.9	39.1	20.0	39.1	47.1
BiFPN+GaFPN	38.0	59.0	41.1	22.6	41.9	49.5

Table 5: Ablative experiments of generality to other variations of FPN. The '*' means our re-implementation.

els are shown in Figure 4. We can observe that the proposals originally mapped to pyramid level P_2 require more features from higher pyramid levels and the importance of demand diminishes as the pyramid level increases. Meanwhile, the proposals originally mapped to $P_3 - P_5$ require more features from lower and higher pyramid levels. In particular, the features of proposals in P_3 and P_4 are dominated by higher-level features. To sum up, features from multi-levels contribute together to generate more powerful features of each proposal. These results demonstrate that features from other levels are also beneficial to classification and regression for each proposal.

Ablation studies on Geometry-aware Predictor Head. The results of ablative experiments on Geometry-aware Predictor Head are shown in Table 5. Since there are two branches in the predictor head, we first explore the influence of feature embedding (product) on different branches. As shown in Table 5, feature embedding on the classification branch improves the baseline method by 0.3 AP. The result show that the GPH generate more discriminative features for classification. Feature embedding on regression branch gains no improvement. The improvement of shared feature embedding on both classification and regression branches reaches 0.4 AP. Then we explore the influence of with/without shared feature embedding for two branches. As shown in the fifth row in Table 5, the performance of unshared feature embedding in two branches is 0.1 AP higher than that of shared feature embedding. Finally, we verify the influence of different embedding methods. When feature embedding by summation is employed, there is no improvement. Through the above analysis, we adopt unshared feature embedding and the product embedding type for two branches in the GPH. These results demonstrate the GPH is beneficial for classification and regression branches to produce more discriminative features.

Setting	AP	Params(M)	FLOPs(G)
Adaptive Spatial Fusion	36.2	0.27	13.32
Geometry-aware Mapping	36.2	0.02	0.02
FPN	35.6	41.53	207.07
GaFPN* (without FAP)	36.9	41.55	207.09
GaFPN	37.3	41.1	194.6

Table 6: Ablative experiments of FLOPs and Params on the COCO val2017. All the reported FLOPs are calculated when input a 1280×800 image.

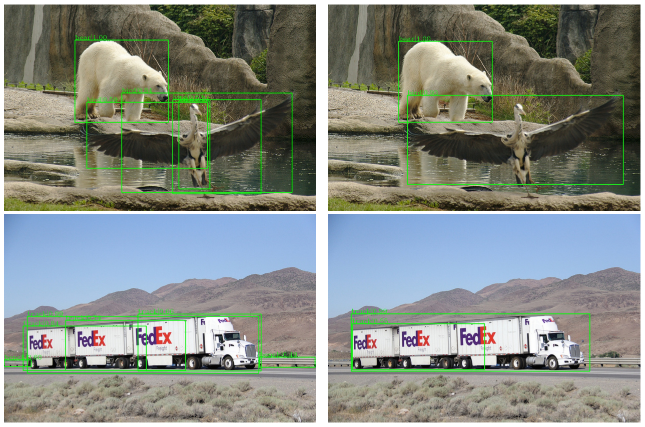

Figure 5: Comparison of object detection results between FPN (left) and GaFPN (right) on COCO val2017.

Generality to other variants of FPN. We find that our GaFPN can also be introduce into the other variants of FPN. Thus, we conduct experiments to evaluate the effectiveness of the GaFPN on PAFPN (Liu et al. 2018) and BiFPN (Tan, Pang, and Le 2020). As shown in Table 5, our method can consistently improve the performance of detection. Specifically, our GaFPN can achieve 1.6 AP increase on PAFPN and 2.0 AP increase on BiFPN. Especially under the AP_{50}, the improvement is more larger. We believe that the proposed method can be applied to other similar detectors. Finally, we show some examples of detection results in Figure 5, where GaFPN generates more accurate results compared to the FPN based baseline.

Runtime Analysis

We analyze the increased FLOPs and Params of two components and the total FLOPs and Params of the GaFPN, as can be seen from Table 6. Specifically, our method does not introduce large Params and FLOPs compared with the baseline method. We also measure the inference time with and without GaFPN. When inputting a shorter size of 800 pixels image, GaFPN*, GaFPN, and FPN can run at 11.8 fps, 10.8 fps, and 13.4 fps, respectively. The inference time is the average inference time over COCO val5000 split, including the time of data loading, network forwarding, and post-processing. All the runtimes are tested on NVIDIA RTX 2080Ti GPU.

Conclusion

In this paper, we revisit the training process of FPN-based detectors and present some promotions in the model architecture. Based on the observation, we propose a new feature pyramid network called GaFPN to further enhance feature representation for multi-scale object detection. The GaFPN consists of three components: Geometry-aware Mapping Module, Geometry-aware Predictor Head and Feature Augmentation Pyramid. By equipping with these simple but effective components, GaFPN brings a large margin improvement compared with various detectors and tasks on the challenging MS COCO dataset.

Acknowledgments

This work was supported by by the National Natural Science Foundation of China under Grant 61790562 and Grant 61773312.

References

Cai, Z.; and Vasconcelos, N. 2018. Cascade R-CNN: Delving Into High Quality Object Detection. In *CVPR*, 6154–6162.

Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; Zhang, Z.; Cheng, D.; Zhu, C.; Cheng, T.; Zhao, Q.; Li, B.; Lu, X.; Zhu, R.; Wu, Y.; Dai, J.; Wang, J.; Shi, J.; Ouyang, W.; Loy, C. C.; and Lin, D. 2019. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*.

Chen, L.-C.; Papandreou, G.; Schroff, F.; and Adam, H. 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv e-prints*, arXiv:1706.05587.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable Convolutional Networks. In *ICCV*, 764–773.

Girshick, R. 2015. Fast R-CNN. In *ICCV*, 1440–1448.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 580–587.

Girshick, R.; Radosavovic, I.; Gkioxari, G.; Dollár, P.; and He, K. 2018. Detectron. https://github.com/facebookresearch/detectron.

Guo, C.; Fan, B.; Zhang, Q.; Xiang, S.; and Pan, C. 2020. AugFPN: Improving Multi-Scale Feature Learning for Object Detection. In *CVPR*, 12595–12604.

He, K.; Gkioxari, G.; Dollar, P.; and Girshick, R. 2017. Mask R-CNN. In *ICCV*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9): 1904–1916.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In *CVPR*, 7132–7141.

Jiang, B.; Luo, R.; Mao, J.; Xiao, T.; and Jiang, Y. 2018. Acquisition of Localization Confidence for Accurate Object Detection. In *ECCV*, 784–799.

Kirillov, A.; He, K.; Girshick, R.; Rother, C.; and Dollár, P. 2020. Panoptic Segmentation. In *CVPR*, 9404–9413.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Li, X.; Wang, W.; Hu, X.; and Yang, J. 2019. Selective Kernel Networks. In *CVPR*, 510–519.

Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017a. Feature Pyramid Networks for Object Detection. In *CVPR*, 2117–2125.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017b. Focal loss for dense object detection. In *ICCV*, 2980–2988.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollar, P.; and Zitnick, L. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*, 740–755.

Liu, S.; Qi, L.; Qin, H.; Shi, J.; and Jia, J. 2018. Path Aggregation Network for Instance Segmentation. In *CVPR*, 8759–8768.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. SSD: Single Shot MultiBox Detector. In *ECCV*, 21–37.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*, 779–788.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: Better, Faster, Stronger. In *CVPR*, 7263–7271.

Redmon, J.; and Farhadi, A. 2018. YOLOv3: An Incremental Improvement. *arXiv e-prints*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 91–99.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; and Bernstein, M. a. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252.

Song, G.; Liu, Y.; and Wang, X. 2020. Revisiting the Sibling Head in Object Detector. In *CVPR*, 11563–11572.

Tan, M.; Pang, R.; and Le, Q. V. 2020. Efficientdet: Scalable and efficient object detection. In *CVPR*, 10781–10790.

Tychsen-Smith, L.; and Petersson, L. 2018. Improving Object Localization with Fitness NMS and Bounded IoU Loss. In *CVPR*, 6877–6885.

Wang, J.; Zhang, W.; Cao, Y.; Chen, K.; Pang, J.; Gong, T.; Shi, J.; Loy, C. C.; and Lin, D. 2020. Side-Aware Boundary Localization for More Precise Object Detection. In *ECCV*, 403–419.

Woo, S.; Park, J.; Lee, J.-Y.; and So Kweon, I. 2018. CBAM: Convolutional Block Attention Module. In *ECCV*, 3–19.

Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; and Fu, Y. 2020. Rethinking Classification and Localization for Object Detection. In *CVPR*, 10186–10195.

Zhang, H.; Chang, H.; Ma, B.; Wang, N.; and Chen, X. 2020. Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training. In *ECCV*, 260–275.

Zhu, C.; He, Y.; and Savvides, M. 2019. Feature Selective Anchor-Free Module for Single-Shot Object Detection. In *CVPR*, 840–849.

Zhu, X.; Hu, H.; Lin, S.; and Dai, J. 2019. Deformable ConvNets V2: More Deformable, Better Results. In *CVPR*, 9308–9316.