A^2-FPN: Attention Aggregation based Feature Pyramid Network for Instance Segmentation

Miao Hu Yali Li* Lu Fang Shengjin Wang
Department of Electronic Engineering, Tsinghua University
hum19@mails.tsinghua.edu.cn, {liyali13, fanglu, wgsgj}@tsinghua.edu.cn

Abstract

Learning pyramidal feature representations is crucial for recognizing object instances at different scales. Feature Pyramid Network (FPN) is the classic architecture to build a feature pyramid with high-level semantics throughout. However, intrinsic defects in feature extraction and fusion inhibit FPN from further aggregating more discriminative features. In this work, we propose Attention Aggregation based Feature Pyramid Network (A^2-FPN), to improve multi-scale feature learning through attention-guided feature aggregation. In feature extraction, it extracts discriminative features by collecting-distributing multi-level global context features, and mitigates the semantic information loss due to drastically reduced channels. In feature fusion, it aggregates complementary information from adjacent features to generate location-wise reassembly kernels for content-aware sampling, and employs channel-wise reweighting to enhance the semantic consistency before element-wise addition. A^2-FPN shows consistent gains on different instance segmentation frameworks. By replacing FPN with A^2-FPN in Mask R-CNN, our model boosts the performance by 2.1% and 1.6% mask AP when using ResNet-50 and ResNet-101 as backbone, respectively. Moreover, A^2-FPN achieves an improvement of 2.0% and 1.4% mask AP when integrated into the strong baselines such as Cascade Mask R-CNN and Hybrid Task Cascade.

1. Introduction

Instance segmentation is one of the most challenging tasks in computer vision. It aims to categorize and localize individual objects with pixel-wise instance masks. Accurate instance segmentation has wide applications in real scenarios including automatic driving and video surveillance. Driven by the rapid advances in deep convolutional networks (ConvNets), the development of instance segmentation frameworks, *e.g.*, Mask R-CNN [14], PANet [26], and HTC [5], has substantially pushed forward the state-of-the-art. Learning multi-scale feature representations is of great significance because high-performance instance segmentation needs to recognize varying numbers of instances across a broad range of scales and locations.

*Corresponding author.

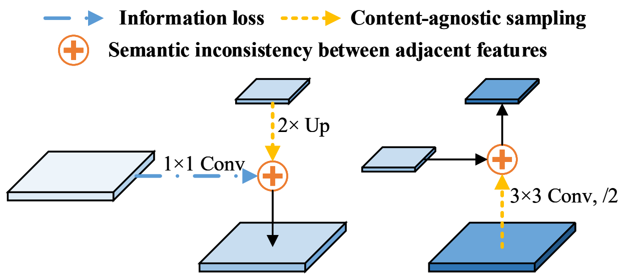
Information loss
Content-agnostic sampling
Semantic inconsistency between adjacent features
2× Up
1×1 Conv
3×3 Conv, /2

Figure 1. **Defects in the construction of feature pyramid:** (1) information loss, (2) content-agnostic sampling, and (3) semantic inconsistency between adjacent features.

To address the issue of multi-scale processing, Feature Pyramid Network (FPN) [22] is widely adopted in existing frameworks. FPN leverages the inherent feature hierarchy and constructs a feature pyramid that has strong semantics at all scales by fusing adjacent features through lateral connections and a top-down pathway. PAFPN in PANet [26] shortens the information path from the low level to top ones by adding an extra bottom-up pathway, further improving the localization capability of the feature pyramid.

Although FPN and PAFPN are effective in learning multi-scale feature representations, the simple designs inhibit feature pyramids from further aggregating more discriminative features. We decompose the construction of feature pyramid into feature extraction and fusion, and find each step has some intrinsic defects, as shown in Figure 1.

In feature extraction, lateral connections using 1×1 convolutional layers are employed to generate features of the same channel dimension. However, the extracted feature maps, especially the high levels, suffer from serious information loss because of drastic dimension reduction. In the first step of feature fusion, feature maps are upsampled using interpolation in the top-down pathway or downsampled through strided convolution in the bottom-up pathway. However, interpolation executes the upsampling process in a sub-pixel neighborhood according to the relative positions of pixels, failing to capture rich semantic information. Strided convolution applies the content-agnostic downsampling kernel across the entire image, neglecting the underlying content of features. In the second step of feature fusion, two adjacent features are merged by element-wise addition, which ignores the semantic gap between feature maps caused by different depths.

In this work, we propose Attention Aggregation based Feature Pyramid Network (A^2-FPN), to improve multi-scale feature learning through attention-guided feature aggregation. Compared to existing frameworks, A^2-FPN is distinctive in three significant aspects: (1) It extracts discriminative features by collecting global context features from the whole feature hierarchy and distributing them to each level. (2) It aggregates complementary information from adjacent features to produce location-wise reassembly kernels for content-aware upsampling and downsampling. (3) It applies channel-wise reweighting to enhance the semantic consistency before element-wise addition.

Without bells and whistles, A^2-FPN in Mask R-CNN framework leads to an improvement of 2.1% and 1.6% mask AP compared with the FPN based counterpart when using ResNet-50 and ResNet-101 as backbone, respectively. Moreover, when integrated into the state-of-the-art instance segmentation methods such as Cascade Mask RCNN and HTC [5], it achieves 2.0% and 1.4% higher mask AP than baseline models on the MS COCO dataset [24].

Our main contributions are summarized as follows: (1) We propose Attention Aggregation based Feature Pyramid Network (A^2-FPN), which effectively aggregates pyramidal feature representations through attention-guided feature extraction and fusion. (2)We demonstrate that not only cross-scale connections are important, but the node operations to aggregate features are also crucial to the construction of feature pyramid. (3) We evaluate A^2-FPN on the challenging COCO dataset [24] through comprehensive experiments, and it can bring consistent and substantial improvements upon various frameworks and backbone networks.

2. Related Work

Instance Segmentation. Current instance segmentation methods can be roughly divided into two categories, detection-based and segmentation-based. Detection-based methods employ object detectors [32, 23, 33, 3] to generate region proposals or bounding boxes, and then produce a pixel-wise mask for each instance. DeepMask [30], SharpMask [31] and MultiPathNet [41] predict object segments using discriminative ConvNets and improve progressively. MNC [8] decomposes instance segmentation into three sub-tasks: instance differentiation, mask estimation, and object categorization. FCIS [20] is proposed to predict instance masks fully convolutionally based on Instance-FCN [8]. Mask R-CNN [14] extends Faster R-CNN [33] by adding a mask prediction branch in parallel with the existing branch for classification and bounding box regression. PANet [26] shortens the information path in FPN [22] by adding an extra bottom-up pathway and aggregates features from all levels through adaptive feature pooling. Mask Scoring R-CNN [17] calibrates the misalignment between mask quality and mask score by learning a maskIoU for each mask instead of using its classification score. HTC [5] integrates cascade into instance segmentation by interweaving bounding box regression and mask prediction in a multi-stage cascade manner, and incorporates contextual information by adding a semantic segmentation branch.

Segmentation-based methods first exploit a pixel-wise segmentation map over the entire image and then group the pixels of different instances. InstanceCut [18] combines semantic segmentation and boundary detection for instance partition. SGN [25] employs a sequence of neural networks to handle progressive sub-grouping problems. Deep learning and watershed transform are integrated in [2] to produce pixel-level energy values for instance derivation. Recent methods [9, 29, 10] use metric learning to learn per-pixel embedding and group pixels to form the instance masks.

Feature Pyramid. Pyramidal feature representations form the basis of solutions to multi-scale problems. SSD [27] firstly attempts to perform object detection on the pyramidal features. FPN [22] builds a feature pyramid of strong semantics through lateral connections and a top-down pathway. Based on FPN, PAFPN in PANet [26] introduces bottom-up augmentation to facilitate the information flow. EfficientDet [35] repeats bidirectional path multiple times for more high-level feature fusion. NAS-FPN [12] takes advantage of neural architecture search to seek a more powerful feature pyramid structure. Unlike the previous works that focus on the topological structure constructed by different cross-scale connections, node operations to aggregate features are explored in this work.

Attention Mechanism. Self-attention is first proposed in [36] for machine translation, where scaled dot-product attention is adopted. The effectiveness of Non-local operation to computer vision tasks is explored later in [38]. Graph reasoning is employed to model semantic nodes in [19, 21, 7, 42]. However, multi-level global context modeling is rarely explored in detectors. Channel attention is used to explicitly model interdependencies between channels in [16]. DANet [11] and GCNet [4] combine self-attention and channel attention to capture rich contextual dependencies.

3. Methodology

The overall framework of A^2-FPN is illustrated in Figure 2. We take ResNet [15] as backbone like FPN [22] and

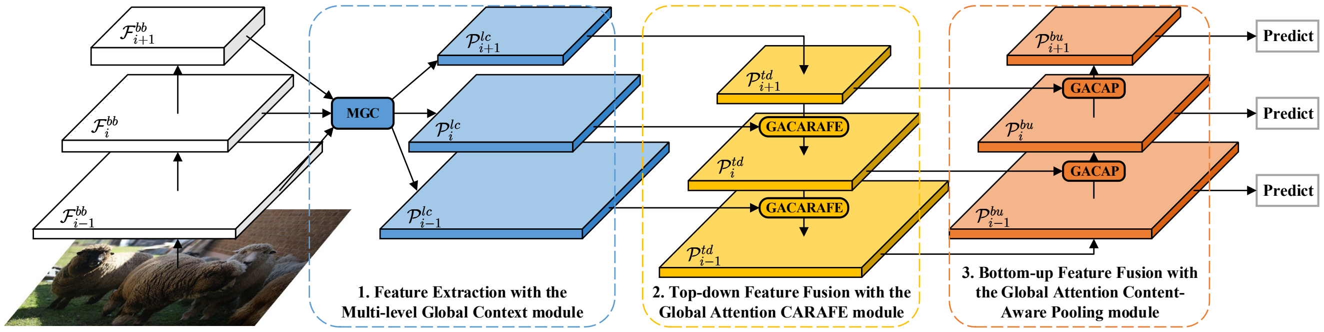

Figure 2. **Overall pipeline of A^2-FPN.** A^2-FPN extracts and fuses the pyramidal features progressively through three proposed modules, the MGC module, the GACARAFE module, and the GACAP module. For brevity, only three feature levels are plotted here.

utilize $\{\mathcal{F}^{bb}_2, \mathcal{F}^{bb}_3, \mathcal{F}^{bb}_4, \mathcal{F}^{bb}_5\}$ to denote the feature hierarchy. In addition, an extra feature $\mathcal{F}^{bb}_6$ is produced through a 3×3 convolutional layer with stride 2 from the feature $\mathcal{F}^{bb}_5$. Note that the pyramidal features have strides of $\{4, 8, 16, 32, 64\}$ pixels w.r.t. the input image. $\{\mathcal{P}^{lc}_2, \mathcal{P}^{lc}_3, \mathcal{P}^{lc}_4, \mathcal{P}^{lc}_5, \mathcal{P}^{lc}_6\}$ are the context-rich features extracted from the feature hierarchy in the Multi-level Global Context (MGC) module (Section 3.1). $\{\mathcal{P}^{td}_2, \mathcal{P}^{td}_3, \mathcal{P}^{td}_4, \mathcal{P}^{td}_5, \mathcal{P}^{td}_6\}$ are the features after top-down path augmentation with the Global Attention CARAFE (GACARAFE) module (Section 3.2), and $\{\mathcal{P}^{bu}_2, \mathcal{P}^{bu}_3, \mathcal{P}^{bu}_4, \mathcal{P}^{bu}_5, \mathcal{P}^{bu}_6\}$ are the features after bottom-up path augmentation with the Global Attention Content-Aware Pooling (GACAP) module (Section 3.3).

3.1. Multi-level Global Context

In feature extraction, pyramidal features suffer from information loss due to channel reduction. ParseNet [28] distributes the global context feature to all locations for information supplement. However, a single context feature collected by global average pooling ignores different needs across scales and locations. Inspired by this, we propose the Multi-level Global Context (MGC) module to extract more discriminative features by aggregating multi-level global context features and mitigate semantic information loss.

MGC consists of three steps to adaptively aggregate global context features from the feature hierarchy, as shown in Figure 3. First, the Context Collector collects global context features from all feature levels through attention pooling. Second, the Graph Convolutional Networks (GCNs) are employed to reason contextual relations. Third, the Context Distributor distributes the context features to each level. Each step will be discussed in detail as follows.

Context Collector. Given the i-th feature $\mathcal{F}^{bb}_i \in \mathbb{R}^{c_i \times h \times w}$ from the backbone, where c_i denotes channel dimension and h, w are spatial size, Context Collector generates a new feature $\mathcal{G}_i \in \mathbb{R}^{c \times n_i}$ comprising n_i different context features of dimension c. Inspired by AN [39], we assume that the features of different stages differ in semantic richness and intensity, and are composed of diverse semantic entities. Here the feature $\mathcal{F}^{bb}_i$ is supposed to consist of n_i semantic entities, and the feature points are gathered into different context features according to their cosine similarity to the semantic entities.

In Context Collector, one 1×1 convolutional layer ψ with n_i filters is adopted as semantic entities, and another 1×1 convolutional layer ϕ is used to embed the input feature. In particular, we formulate this procedure as Eqn. 1.

$$\mathcal{G}_i = W_\phi \mathcal{F}^{bb}_i \text{Softmax}(\sqrt{c_i} \cdot W_\psi \text{Norm}(\mathcal{F}^{bb}_i))^T, \quad (1)$$

$$L_o = \lambda_o \|W_\psi W_\psi^T - I\|_F^2, \quad (2)$$

where $W_\psi \in \mathbb{R}^{n_i \times c_i}$, $W_\phi \in \mathbb{R}^{c \times c_i}$, and Norm represents L_2 normalization. To normalize the semantic entities for computing cosine similarity and learn more diverse patterns, orthogonal regularization is applied to the weights W_ψ as shown in Eqn. 2. Different from the scaled dot-product attention in [36] which divides each dot product by $\sqrt{c_i}$, we multiply the cosine similarity by $\sqrt{c_i}$ to maintain the variance at different values of c_i. Then we apply a softmax function spatially to generate the attention masks, and collect context features using attention pooling accordingly.

We call our particular attention mechanism "Scaled Cosine-similarity Attention", which computes the compatibility function based on scaled cosine similarity. It focuses on semantic content instead of intensity and can avoid strongly activated keys surpassing other keys. For brevity, we reformulate the compatibility function out of Eqn. 1:

$$\boldsymbol{CF}(W_\psi^T, \mathcal{F}^{bb}_i, c_i) = \text{Softmax}(\sqrt{c_i} \cdot W_\psi \text{Norm}(\mathcal{F}^{bb}_i))^T. \quad (3)$$

GCN. Once we obtain context features $\mathcal{G}_i \in \mathbb{R}^{c \times n_i}$, we utilize GCNs to model and reason the contextual relations between them first in single level. The formula of the GCN with a residual connection [15] is interpreted as Eqn. 4, where $W^{gcn}_1 \in \mathbb{R}^{\frac{c}{4} \times c}, W^{gcn}_2 \in \mathbb{R}^{\frac{c}{4} \times c}, W^{gcn}_3 \in \mathbb{R}^{c \times c}$ are weight matrices of 1D convolutional layers. The convolutional weight of the GCN is W^{gcn}_3 and the adjacent matrix is dynamically generated through self-attention described in Eqn. 3. After contextual relations are reasoned in each level independently, all context features are concatenated and reasoned together in multi-level by the GCN as Eqn.

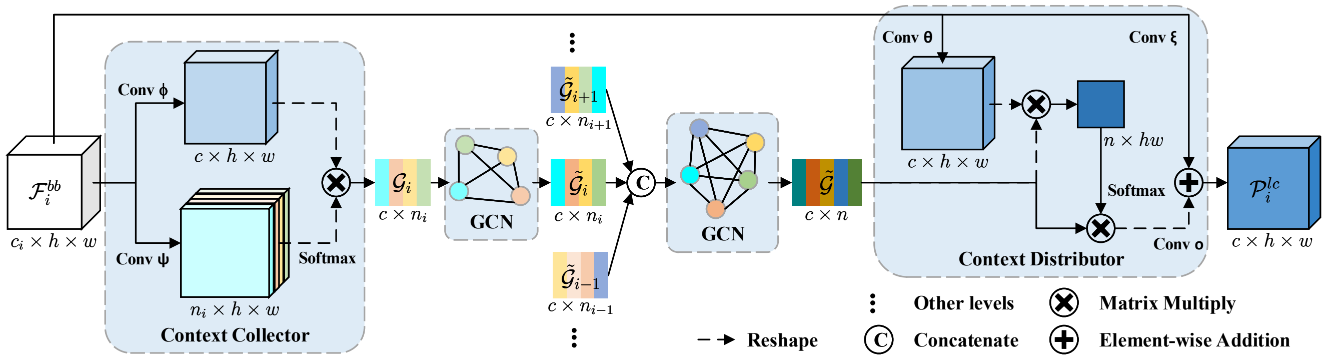

Figure 3. **Detailed illustration of the Multi-level Global Context (MGC) module.** MGC collects, reasons, and distributes multi-level global context features through three children modules, the Context Collector, the GCNs, and the Context Distributor, respectively.

5. In this work, we collect $\{n_2, n_3, n_4, n_5\}$ context features from the feature maps $\{\mathcal{F}_2^{bb}, \mathcal{F}_3^{bb}, \mathcal{F}_4^{bb}, \mathcal{F}_5^{bb}\}$, respectively.

$$\tilde{\mathcal{G}}_i = W_3^{gcn}\mathcal{G}_i\boldsymbol{CF}(W_1^{gcn}\mathcal{G}_i, W_2^{gcn}\mathcal{G}_i, \frac{c}{4}) + \mathcal{G}_i, \quad (4)$$

$$\tilde{\mathcal{G}} = \boldsymbol{GCN}([\cdots, \tilde{\mathcal{G}}_{i-1}, \tilde{\mathcal{G}}_i, \tilde{\mathcal{G}}_{i+1}, \cdots]). \quad (5)$$

Context Distributor. Context Distributor distributes the multi-level global context features $\tilde{\mathcal{G}} \in \mathbb{R}^{c\times n}$ to each level using the proposed scaled cosine-similarity attention. As shown in Eqn. 6, $W_o \in \mathbb{R}^{c\times c}$, $W_\theta \in \mathbb{R}^{c\times c_i}$ and $W_\xi \in \mathbb{R}^{c\times c_i}$ are weights of 1×1 convolutional layers. The feature maps $\mathcal{F}_i^{bb}$ are queries and the global context features $\tilde{\mathcal{G}}$ are shared as keys and values.

$$\mathcal{P}_i^{lc} = W_o\tilde{\mathcal{G}}\boldsymbol{CF}(W_\theta\mathcal{F}_i^{bb}, \tilde{\mathcal{G}}, c) + W_\xi\mathcal{F}_i^{bb}. \quad (6)$$

3.2. Global Attention CARAFE

In the top-down pathway of feature fusion, interpolation depends only on the relative positions and direct element-wise addition ignores the semantic gap between adjacent features. Recently, CARAFE [37] is proposed to upsample features through content-aware reassembly. However, CARAFE only leverages the information of low-resolution feature and fails to capture complementary semantics from the high-resolution level. As shown in Figure 4 (left), we propose the Global Attention CARAFE (GACARAFE) module for feature fusion, where the complementary information is aggregated to guide content-aware upsampling and channel attention [16] is employed to enhance the semantic consistency before merging adjacent features.

The feature $\mathcal{P}_i^{lc} \in \mathbb{R}^{c\times h\times w}$ is extracted through the MGC module and the feature $\mathcal{P}_{i+1}^{td} \in \mathbb{R}^{c\times\frac{h}{s}\times\frac{w}{s}}$ is the upper level in the top-down pathway, where $s = 2$ is the scale factor. We downsample the feature $\mathcal{P}_i^{lc}$ using max-pooling and concatenate $\mathcal{P}_{i+1}^{td}$ and $\tilde{\mathcal{P}}_i^{lc}$ together. As shown in Figure 4 (left), the feature $\mathcal{P}_{i+1}^{td}$ is upsampled as $\tilde{\mathcal{P}}_{i+1}^{td}$ through two steps. GACARAFE predicts a location-wise kernel $\mathcal{K}_{i+1}^{up}(x,y)$ based on the $k_{en}^{up} \times k_{en}^{up}$ neighbor of feature points $\mathcal{P}_{i+1}^{td}(\frac{x}{s}, \frac{y}{s})$ and $\tilde{\mathcal{P}}_i^{lc}(\frac{x}{s}, \frac{y}{s})$, as shown in Eqn. 7. And then it reassembles the $k_{up} \times k_{up}$ neighbor of $\mathcal{P}_{i+1}^{td}(\frac{x}{s}, \frac{y}{s})$ with the kernel $\mathcal{K}_{i+1}^{up}(x,y)$, as presented in Eqn. 8.

$$\mathcal{K}_{i+1}^{up}(x,y) = \boldsymbol{K}(\mathcal{N}([\mathcal{P}_{i+1}^{td}, \tilde{\mathcal{P}}_i^{lc}](\frac{x}{s}, \frac{y}{s}), k_{en}^{up})), \quad (7)$$

$$\tilde{\mathcal{P}}_{i+1}^{td}(x,y) = \boldsymbol{R}(\mathcal{N}(\mathcal{P}_{i+1}^{td}(\frac{x}{s}, \frac{y}{s}), k_{up}), \mathcal{K}_{i+1}^{up}(x,y)), \quad (8)$$

where $\boldsymbol{K}$ and $\boldsymbol{R}$ are the kernel prediction module and content-aware reassembly module altered from CARAFE [37], respectively. The kernel prediction module contains three convolutions: the first one of 1×1 kernel for reducing channel dimension from $2c$ to c_m^{up}, the second one of 3×3 kernel with ReLU activation for blending features, and the third one of $k_{en}^{up} \times k_{en}^{up}$ kernel for predicting upsampling kernels. The kernels $\mathcal{K}_{i+1}^{up}$ are deformed using pixel shuffle [34] and normalized with a softmax function.

Besides, we apply channel attention to learn two vectors $\hat{\mathcal{S}}_i^{up}$ and $\check{\mathcal{S}}_i^{up}$ to recalibrate the adjacent features. As shown in Eqn. 9, we squeeze the global spatial information into a channel descriptor using attention pooling. Specifically, a 1×1 convolutional layer is employed to predict the attention mask $\mathcal{M}_i^{up}$ and channel-wise statistics $\mathcal{Z}_i^{up}$ is generated by shrinking the concatenated feature accordingly. And then a simple gating mechanism with $2\cdot\mathrm{sigmoid}$ activation is adopted to model channel-wise interdependencies.

$$\begin{aligned} \mathcal{M}_i^{up} &= \mathrm{Softmax}(W_1^{up}[\mathcal{P}_{i+1}^{td}, \tilde{\mathcal{P}}_i^{lc}])^T, \\ \mathcal{Z}_i^{up} &= [\mathcal{P}_{i+1}^{td}, \tilde{\mathcal{P}}_i^{lc}]\mathcal{M}_i^{up}, \\ [\hat{\mathcal{S}}_i^{up}, \check{\mathcal{S}}_i^{up}] &= 2\sigma(W_3^{up}\mathrm{ReLU}(\mathrm{LN}(W_2^{up}\mathcal{Z}_i^{up}))). \end{aligned} \quad (9)$$

Here LN stands for LayerNorm [1], $W_1^{up} \in \mathbb{R}^{1\times 2c}$, $W_2^{up} \in \mathbb{R}^{\frac{c}{2}\times 2c}$ and $W_3^{up} \in \mathbb{R}^{2c\times\frac{c}{2}}$. 2σ indicates the activation function $2\cdot\mathrm{sigmoid}$, which can keep the mean of channel-wise weights being 1 after successive multiplications and excite or restrain features selectively. Finally, the adjacent features $\tilde{\mathcal{P}}_{i+1}^{td}$ and $\mathcal{P}_i^{lc}$ are reweighted and merged by element-wise addition as shown in Eqn. 10.

$$\mathcal{P}_i^{td} = \hat{\mathcal{S}}_i^{up} \odot \tilde{\mathcal{P}}_{i+1}^{td} + \check{\mathcal{S}}_i^{up} \odot \mathcal{P}_i^{lc}. \quad (10)$$

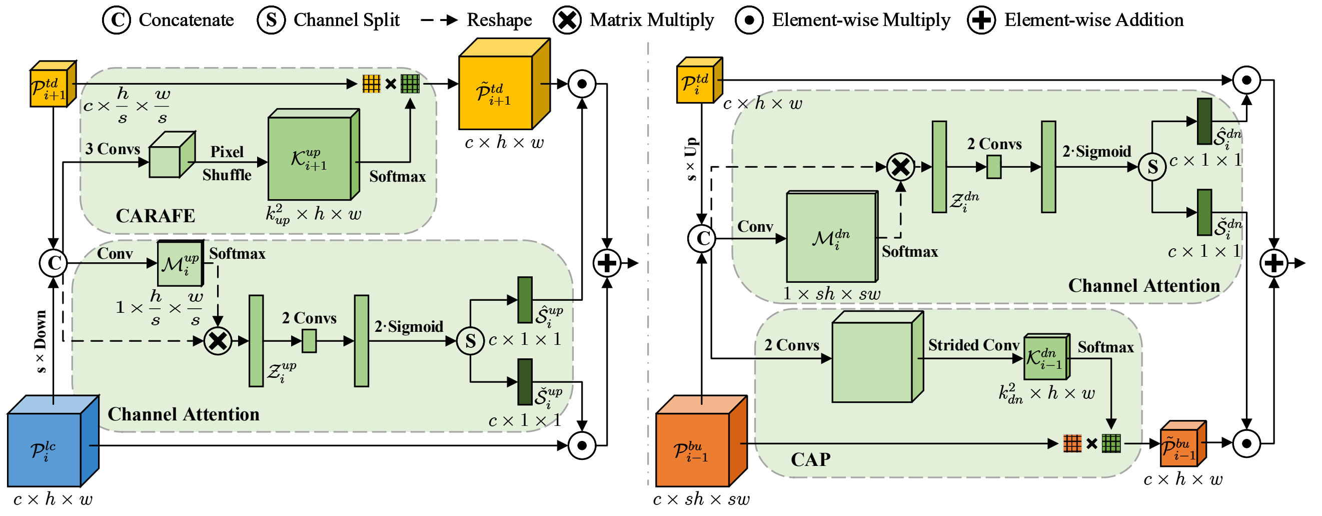

Figure 4. **The details of the Global Attention CARAFE (GACARAFE) module (left) and the Global Attention Content-Aware Pooling (GACAP) module (right).** Noting that the fused features $\mathcal{P}_i^{td}$ and $\mathcal{P}_i^{bu}$ after element-wise addition are omitted for simplification.

3.3. Global Attention Content-Aware Pooling

In the bottom-up pathway, strided convolution applies the same kernel across the entire image and ignores the underlying content. To aggregate more discriminative features, we propose the Global Attention Content-Aware Pooling (GACAP) module to execute the bottom-up fusion. As shown in Figure 4 (right), the Content-Aware Pooling (CAP) operator extends the idea of CARAFE [37] to feature pooling and the GACAP module differs from the GACARAFE module in several aspects.

First, GACAP unsamples the feature $\mathcal{P}_i^{td} \in \mathbb{R}^{c\times h\times w}$ to get $\tilde{\mathcal{P}}_i^{td}$ through bilinear interpolation and concatenates $\mathcal{P}_{i-1}^{bu} \in \mathbb{R}^{c\times sh\times sw}$ with it. Second, GACAP downsamples the feature $\mathcal{P}_{i-1}^{bu}$ as $\tilde{\mathcal{P}}_{i-1}^{bu}$, as shown in Eqn. 11 and 12.

$$\mathcal{K}_{i-1}^{dn}(x,y) = \boldsymbol{K}(\mathcal{N}([\mathcal{P}_{i-1}^{bu}, \tilde{\mathcal{P}}_i^{td}](sx,sy), k_{en}^{dn})), \quad (11)$$

$$\tilde{\mathcal{P}}_{i-1}^{bu}(x,y) = \boldsymbol{R}(\mathcal{N}(\mathcal{P}_{i-1}^{bu}(sx,sy), k_{dn}), \mathcal{K}_{i-1}^{dn}(x,y)), \quad (12)$$

where the kernel prediction module $\boldsymbol{K}$ applies a $k_{en}^{dn} \times k_{en}^{dn}$ convolutional layer with stride s to generate downsampling kernels $\mathcal{K}_{i-1}^{dn}$ directly. The other parts of GACAP is exactly the same with GACARAFE described in Section 3.2. After the channel attention vectors $\hat{\mathcal{S}}_i^{dn}$ and $\check{\mathcal{S}}_i^{dn}$ are obtained, the adjacent features are fused to generate feature $\mathcal{P}_i^{bu}$ as Eqn. 13. Notably, we finally append a 3×3 convolution on each merged feature map to reduce the aliasing effect, whether in the top-down pathway or the bottom-up pathway.

$$\mathcal{P}_i^{bu} = \hat{\mathcal{S}}_i^{dn} \odot \mathcal{P}_i^{td} + \check{\mathcal{S}}_i^{dn} \odot \tilde{\mathcal{P}}_{i-1}^{bu}. \quad (13)$$

4. Experiments

4.1. Datasets and Evaluation Metrics

Datasets. We conduct experiments on the challenging MS COCO dataset [24]. It contains 115k images for training (*train2017*), 5k images for validation (*val2017*), and 20k images for testing (*test-dev*). The labels of *test-dev* are not publicly available. We train our models on *train2017* subset and report results on *val2017* subset for ablation study. we also report results on *test-dev* for comparison.

Evaluation Metrics. We report the standard COCO-style Average Precision (AP) metric including AP (averaged over IoU thresholds), AP_{50}, AP_{75} (AP at different IoU thresholds), and AP_S, AP_M, AP_L (AP at different scales). Since our framework is general to both instance segmentation and object detection, both mask AP and box AP (superscripted as "*bb*") are evaluated.

4.2. Implementation Details

All experiments are implemented based on MMDetection [6]. We train detectors with a batchsize of 8 over 4 GPUs (2 images per GPU) for 12 epochs with an initial learning rate of 0.01, and decrease it by 0.1 after 8 and 11 epochs, respectively. Images are resized to a maximum scale of 1333×800 pixels without changing the aspect ratio. If not otherwise specified, A^2-FPN adopts a fixed set of hyper-parameters in experiments, where $c = 256$, $n_i = 64(6-i)$, and $\lambda_o = 0.0001$ in Eqn. 2 in the MGC module, $C_m^{up} = 64, k_{en}^{up} = 3$ and $k_{up} = 5$ in the GACARAFE module, $C_m^{dn} = 64, k_{en}^{dn} = 3$ and $k_{dn} = 5$ in the GACAP module. All other hyper-parameters in this work follow the default setting of MMDetection [6].

4.3. Benchmarking Results

We integrate A^2-FPN into the state-of-the-art detectors and evaluate the performance on the COCO *test-dev*. For a fair comparison, we re-implement the corresponding FPN based methods as baselines.

Instance Segmentation Results. A^2-FPN, combined

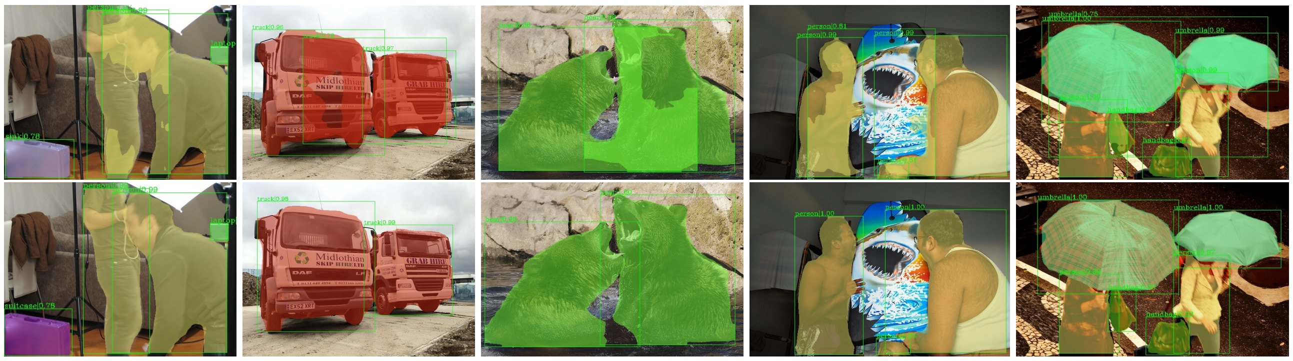
Figure 5. Comparison of instance segmentation results between **FPN** (top row) and ***A*^{2}-FPN** (bottom row) on COCO *val2017*.

Table 1. **Instance segmentation mask AP and object detection box AP**, vs. the state-of-the-art on COCO *test-dev*. "*ms*" in [] indicates multi-scale training and the symbol '*' denotes our re-implementation results. The letters 'R' and 'X' stand for the backbone networks ResNet [15] and ResNeXt [40], respectively. 'Sch.' is short for the training schedule, which follows the setting of MMdetection [6].

Method	Backbone	Sch.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_S	AP^{bb}_M	AP^{bb}_L
Faster R-CNN [22]	R-101-FPN	-	-	-	-	-	-	-	36.2	59.1	-	18.2	39.0	48.2
Mask R-CNN [14]	R-101-FPN	-	35.7	58.0	37.8	15.5	38.1	52.4	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN [14]	X-101-FPN	-	37.1	60.0	39.4	16.9	39.9	53.5	39.8	62.3	43.4	22.1	43.2	51.2
Mask R-CNN [13]	R-101-AugFPN	1x	37.8	60.4	40.4	20.4	41.0	49.8	41.3	63.5	44.9	24.2	44.8	52.0
PANet [26]	R-50-PAFPN	-	36.6	58.0	39.3	16.3	38.1	53.1	41.2	60.4	44.4	22.7	44.0	54.6
PANet[*ms*] [26]	R-50-PAFPN	-	38.2	60.2	41.4	19.1	41.1	52.6	42.5	62.3	46.4	26.3	47.0	52.3
PANet [26]	X-101-PAFPN	-	40.0	62.8	43.1	18.8	42.3	57.2	45.0	65.0	48.6	25.4	48.6	59.1
PANet[*ms*] [26]	X-101-PAFPN	-	42.0	65.1	45.7	22.4	44.7	58.1	47.4	67.2	51.8	30.1	51.7	60.0
Cascade R-CNN [3]	R-50-FPN	-	-	-	-	-	-	-	40.6	59.9	44.0	22.6	42.7	52.1
Cascade R-CNN [3]	R-101-FPN	-	-	-	-	-	-	-	42.8	62.1	46.3	23.7	45.5	55.2
HTC [5]	R-50-FPN	20e	38.4	60.0	41.5	20.4	40.7	51.2	43.6	-	-	-	-	-
HTC [5]	R-101-FPN	20e	39.7	61.8	43.1	21.0	42.2	53.5	45.3	-	-	-	-	-
HTC [5]	X-101-FPN	20e	41.2	63.9	44.7	22.8	43.9	54.6	47.1	-	-	-	-	-
Mask R-CNN*	R-50-FPN	1x	34.5	56.3	36.7	18.6	37.3	44.7	37.6	59.5	40.6	21.8	40.8	46.4
Mask R-CNN(ours)	R-50-A^2-FPN-Lite	1x	**36.4**	**58.9**	**38.7**	**19.6**	**39.1**	**47.8**	**39.8**	**62.3**	**43.4**	**23.6**	**42.8**	**49.9**
Mask R-CNN(ours)	R-50-A^2-FPN	1x	**36.6**	**59.3**	**39.1**	**19.8**	**39.3**	**48.0**	**40.2**	**62.7**	**43.7**	**23.7**	**43.2**	**50.2**
Mask R-CNN[*ms*](ours)	R-50-A^2-FPN	2x	**38.8**	**62.0**	**41.6**	**22.1**	**41.7**	**50.4**	**42.8**	**65.2**	**47.0**	**26.5**	**45.9**	**53.3**
Mask R-CNN*	R-101-FPN	1x	36.3	58.5	38.9	19.4	39.3	47.8	39.9	61.6	43.6	23.1	43.2	50.0
Mask R-CNN(ours)	R-101-A^2-FPN	1x	**37.9**	**60.8**	**40.5**	**20.6**	**41.8**	**50.1**	**41.7**	**64.1**	**45.5**	**24.6**	**45.0**	**52.5**
Cascade Mask R-CNN*	R-50-FPN	1x	36.1	57.1	38.9	19.1	38.6	47.4	41.5	59.8	45.1	23.4	44.3	52.7
Cascade Mask R-CNN(ours)	R-50-A^2-FPN	1x	**38.1**	**60.1**	**41.0**	**20.6**	**40.5**	**50.4**	**43.9**	**62.8**	**47.7**	**25.4**	**46.8**	**56.0**
Cascade Mask R-CNN*	R-101-FPN	1x	37.4	58.9	40.5	19.5	40.1	49.6	43.3	61.6	47.2	24.1	46.2	55.3
Cascade Mask R-CNN(ours)	R-101-A^2-FPN	1x	**39.1**	**61.3**	**42.2**	**21.0**	**41.9**	**51.8**	**45.1**	**64.1**	**48.9**	**25.7**	**48.2**	**57.7**
HTC*	R-50-FPN	20e	38.4	60.0	41.4	20.3	40.6	51.2	43.5	62.6	47.3	24.5	45.9	55.9
HTC(ours)	R-50-A^2-FPN	20e	**39.8**	**62.3**	**43.0**	**21.6**	**42.4**	**52.8**	**45.4**	**64.9**	**49.1**	**26.3**	**48.2**	**57.7**
HTC*	R-101-FPN	20e	39.6	61.6	42.9	21.1	42.2	53.1	45.1	64.3	49.0	25.7	47.9	58.2
HTC(ours)	R-101-A^2-FPN	20e	**40.8**	**63.6**	**44.1**	**22.3**	**43.5**	**54.4**	**46.6**	**66.2**	**50.4**	**27.1**	**49.6**	**59.9**
HTC*	X-101-FPN	20e	41.3	64.0	44.8	22.7	43.9	54.8	47.2	66.6	51.4	27.7	50.0	60.2
HTC(ours)	X-101-A^2-FPN	20e	**42.1**	**65.3**	**45.7**	**23.6**	**44.8**	**56.0**	**48.3**	**68.0**	**52.4**	**28.9**	**51.3**	**61.7**
HTC[*ms*](ours)	X-101-A^2-FPN	2x	**44.0**	**67.5**	**47.9**	**25.7**	**47.1**	**57.6**	**50.4**	**70.1**	**54.9**	**31.6**	**53.7**	**63.7**

with different instance segmentation frameworks and backbone networks, achieves better performance compared with baseline models. As shown in Table 1, by replacing FPN with A^2-FPN, Mask R-CNN using ResNet-50 (denoted as R-50-A^2-FPN) achieves 36.6% mask AP, which is 2.1% higher than the counterpart. Moreover, when using ResNet-101 as the feature extractor, A^2-FPN still brings a gain of 1.6%, proving that A^2-FPN can consistently improve the performance even with a more powerful backbone.

A^2-FPN-Lite is the lightweight setting of A^2-FPN including three differences, halving the channel ($c = 128$) and number ($n_i = 32(6 - i)$) of context features, removing $\mathcal{F}_6^{bb}$ and generating $\mathcal{F}_6^{bu}$ from $\mathcal{F}_5^{bu}$ using max-pooling like FPN [22], and removing the 3×3 convolution after $\mathcal{F}_2^{bu}$ like PAFPN [26]. As shown in Table 1, A^2-FPN-Lite improves the performance of Mask R-CNN by 1.9% mask AP.

Besides, our proposed method works well with different frameworks. A^2-FPN combined with Cascade Mask R-CNN leads to a gain of 2.0% and 1.7% for ResNet-50 and ResNet-101, respectively. It is noteworthy that A^2-FPN

Table 2. **Effect of each module in our design.** MC: the MGC module, GE: the GACARAFE module, GP: the GACAP module. Results are reported on COCO *val2017*.

MC	**GE**	**GP**	AP^{bb}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN [22]			37.1	34.1	55.4	36.2	18.4	37.3	46.0
PAFPN [26]			37.6	34.4	55.9	36.4	18.7	37.5	47.2
✓			38.6	35.4	57.4	37.5	19.5	38.6	48.3
	✓		39.4	35.7	57.9	37.8	19.1	39.0	48.6
		✓	38.3	35.0	56.6	37.2	18.0	38.4	48.3
✓	✓		39.8	36.1	**58.6**	**38.4**	**20.2**	**39.4**	49.4
✓		✓	38.9	35.5	57.7	37.6	19.0	39.2	48.8
	✓	✓	39.6	35.9	58.2	37.9	19.8	39.0	48.9
✓	✓	✓	**40.0**	**36.2**	58.4	38.1	20.1	39.2	**49.6**

further boosts the performance of HTC by 1.4% when using ResNet-50 as backbone, while PAFPN only shows a limited 0.1% gain in the extensive study in HTC [5]. Meanwhile, HTC achieves 40.8% and 42.1% mask AP when using R-101-A^2-FPN and X-101-A^2-FPN as backbone, pushing the powerful HTC by 1.2% and 0.8%, respectively. Particularly, A^2-FPN equipped with HTC achieves 44.0% mask AP under the setting of a '2x' schedule and multi-scale training. The improvements demonstrate that A^2-FPN further improves the multi-scale feature learning of feature pyramid through attention-guided aggregation. In Figure 5, we show some examples of instance segmentation results, where A^2-FPN generates more accurate instance masks compared to the FPN based baseline.

Object Detection Results. We also evaluate A^2-FPN on object detection task as shown in Table 1. By replacing FPN with A^2-FPN in Mask R-CNN, the performance is boosted by 2.6% and 1.8% for ResNet-50 and ResNet-101. A^2-FPN-Lite also boosts the performance of Mask R-CNN by 2.2%. Moreover, Cascade Mask R-CNN can be improved by 2.4% and 1.8% correspondingly when using ResNet-50 and ResNet-101 as backbone. Meanwhile, A^2-FPN equipped with HTC contributes a gain of 1.9%, 1.5%, and 1.1% for ResNet-50, ResNet-101, and ResNeXt-101, respectively. When adopting the setting of a '2x' schedule and multi-scale training, A^2-FPN achieves 50.4% box AP. In brief, A^2-FPN effectively improves the performance of state-of-the-art detectors not only on instance segmentation but also on object detection. As shown in Table 1, A^2-FPN brings consistent gains on various backbone networks, frameworks, and even different tasks, which verifies the effectiveness and generalization ability of A^2-FPN.

4.4. Ablation Study

In this section, we conduct extensive ablation experiments to analyze the effects of main components in A^2-FPN. Note that the baseline method for all ablation studies is Mask R-CNN with R-50-PAFPN.

Component-wise Analysis. Firstly, we investigate the importance of each component in A^2-FPN. The baseline PAFPN is extended from FPN, where the difference from the original implementation in PANet [26] is that we do not use Synchronized BatchNorm. And then MGC, GACARAFE, and GACAP are gradually applied to PAFPN by substituting the corresponding operations with them.

Table 3. **Ablation study of each module** on COCO *val2017*. "w/ ·" indicates A^2-FPN only equipped with the module ·.

Method	setting	AP^{bb}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
A^2-FPN	-	**40.0**	**36.2**	**58.4**	**38.1**	**20.1**	39.2	**49.6**
A^2-FPN	w/o GCN	39.6	35.9	**58.4**	38.0	19.2	**39.4**	49.0
w/ MGC	-	**38.6**	**35.4**	**57.4**	**37.5**	**19.5**	38.6	**48.3**
w/ MGC	w/o GCN	38.5	35.2	56.9	**37.5**	19.1	**38.7**	47.8
w/ GACARAFE	-	**39.4**	**35.7**	**57.9**	**37.8**	**19.1**	**39.0**	48.6
w/ CARAFE [37]	-	38.9	35.4	57.1	37.6	19.0	38.4	**48.7**
w/ GACAP	-	**38.3**	**35.0**	**56.6**	**37.2**	**18.0**	**38.4**	48.3
w/ CAP	-	38.0	34.8	56.2	37.0	17.7	38.0	**48.5**

As shown in Table 2, each module effectively improves the performance of baseline. Specifically, the MGC module boosts both mask AP and box AP by 1.0%. This benefits from that MGC sufficiently extracts discriminative features through cross-scale self-attention based on scaled cosine-similarity attention, and alleviates semantic information loss. The GACARAFE module contributes an improvement of 1.8% and 1.3% in terms of box AP and mask AP. Similarly, 0.7% and 0.6% for the GACAP module. These results indicate that content-aware sampling aggregates more semantic information and attention-guided fusion mitigates the inconsistency between adjacent levels. Besides, the GACARAFE module plays a key role in the construction of feature pyramid since there is only one top-down pathway but two bottom-up pathways.

When combining any two modules, the performance of baseline is further improved. For example, MGC and GACARAFE lead to a gain of 2.2% box AP and 1.7% mask AP. When integrating all three modules together, it achieves 40.0% box AP and 36.2% mask AP, with a gain of 2.4% and 1.8%. The improvements of AP_S, AP_M and AP_L are 1.4%, 1.7% and 2.4% respectively, suggesting that A^2-FPN is beneficial to various scales, especially the large scale.

Ablation study of each module. As shown in Table 3, the performance of MGC and A^2-FPN degrades by 0.2% and 0.3% respectively when removing the GCNs. These results indicate that GCN can model the contextual relations between context features and benefit the subsequent feature fusion. Compared with CARAFE [37], GACARAFE further leads to a gain of 0.5% box AP and 0.3% mask AP. Similarly, 0.3% box AP and 0.2% mask AP for GACAP. These improvements benefit from that GACARAFE and GACAP aggregate complementary information from adjacent features for content-aware sampling and employ channel attention to enhance the semantic consistency.

Impact of the number of context features. To investigate the impact of n_i, the number of context features collected from each level, we conduct contrast experiments

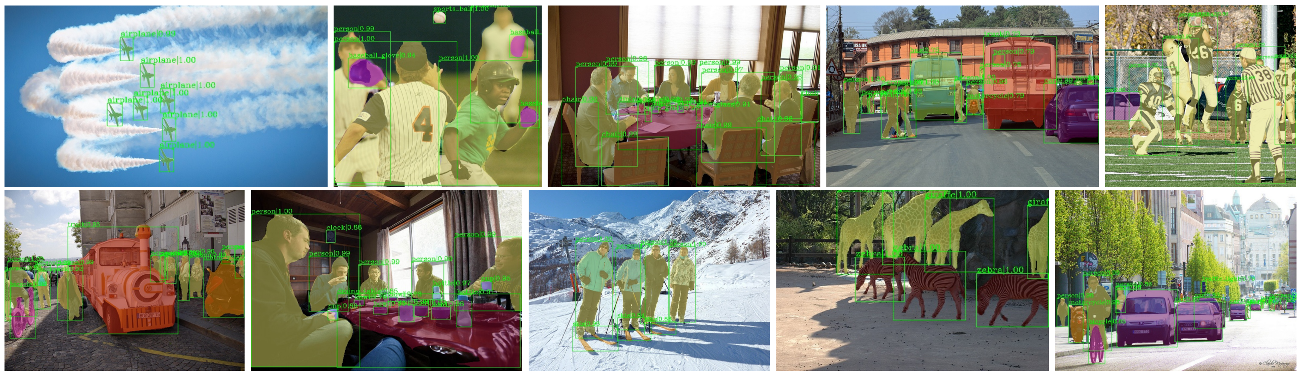
Figure 6. Instance segmentation results of $\boldsymbol{A^2}$**-FPN** equipped with HTC on COCO *val2017*.

Table 4. **Impact of the number of context features** on COCO *val2017*. n_i means the number of context features collected from the i-th feature level $\mathcal{F}_i^{bb}$.

Method	n_i	AP^{bb}	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
FPN [22]	-	37.1	34.1	55.4	36.2	18.4	37.3	46.0
PAFPN [26]	-	37.6	34.4	55.9	36.4	18.7	37.5	47.2
w/ MGC	$64(i-1)$	38.7	**35.4**	**57.8**	37.4	19.2	**38.8**	48.1
w/ MGC	160	**38.8**	35.3	57.4	**37.6**	19.2	38.7	48.2
w/ MGC	$64(6-i)$	38.6	**35.4**	57.4	37.5	**19.5**	38.6	**48.3**

Table 5. **Effectiveness of activation function 2 · sigmoid** on COCO *val2017*. "Act." is short for activation function, and σ means the sigmoid function.

Method	Act.	AP^{bb}	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
FPN [22]	-	37.1	34.1	55.4	36.2	18.4	37.3	46.0
PAFPN [26]	-	37.6	34.4	55.9	36.4	18.7	37.5	47.2
w/ GACARAFE	σ	38.9	35.5	57.5	37.7	**19.2**	38.8	48.3
w/ GACARAFE	2σ	**39.4**	**35.7**	**57.9**	**37.8**	19.1	**39.0**	**48.6**
w/ GACAP	σ	38.2	34.8	56.3	37.0	**18.3**	38.3	47.7
w/ GACAP	2σ	**38.3**	**35.0**	**56.6**	**37.2**	18.0	**38.4**	**48.3**

with different settings. As shown in Table 4, the MGC module is robust to the setting of number n_i and achieves similar performance where the difference is only 0.1% at most. Considering that the more context features from the high levels, the more parameters are introduced because of high dimension, we set the hyper-parameters as $n_i = 64(6-i)$, i.e., $n_i = \{256, 192, 128, 64\}$ and $n = 640$.

Effectiveness of activation function 2 · sigmoid. To verify the effectiveness of activation function $2 \cdot \mathrm{sigmoid}$, we compare it with the conventional function $\mathrm{sigmoid}$ in both GACARAFE and GACAP. As shown in Table 5, the GACARAFE module with function $2 \cdot \mathrm{sigmoid}$ achieves 39.4% box AP and 35.7% mask AP respectively, which is 0.5% and 0.2% higher than the counterpart with function $\mathrm{sigmoid}$. Meanwhile, function $2\cdot\mathrm{sigmoid}$ contributes a gain of 0.1% box AP and 0.2% mask AP in the GACAP module. The pyramidal features are iteratively reweighted in the feature fusion, and the final calibration weights are the product of several activated weights. When using $\mathrm{sigmoid}$ as the activation function, the mean of channel-wise weights would be always less than 1 and decay exponentially after successively reweighting, so that pyramidal features would be suppressed in the iterative fusion. On the contrary, the activation function $2\cdot\mathrm{sigmoid}$ can keep the mean of channel-wise weights being 1 and excite or restrain features selectively.

Table 6. **Complexity analysis of** $\boldsymbol{A^2}$**-FPN** on COCO *val2017*.

Method	Image Size	#FLOPs	#Params	FPS	AP^{bb}	AP
FPN [22]	1280×832	**283.28G**	**44.18M**	**13.1**	37.1	34.1
PAFPN [26]	1280×832	309.05G	47.72M	12.8	37.6	34.4
A^2-FPN	1280×832	375.39G	57.49M	9.5	**40.0**	**36.2**
A^2-FPN-Lite	1280×832	319.91G	48.83M	11.1	39.6	36.0

4.5. Complexity Analysis

As shown in Table 6, we analyze the complexity of A^2-FPN. A^2-FPN adds some computation cost, but A^2-FPN-Lite can reduce it greatly at a small performance sacrifice. Compared to PAFPN, A^2-FPN-Lite improves the performance significantly while increases the computation complexity acceptably, proving that the improvement mostly comes from our attention-guided feature aggregation.

5. Conclusion

We propose Attention Aggregation based Feature Pyramid Network (A^2-FPN), which improves multi-scale feature learning through attention-guided feature aggregation. A^2-FPN extracts discriminative features by collecting-distributing multi-level global context features, and fuses adjacent levels using content-aware sampling and channel-wise reweighting before element-wise addition. Upon various backbone networks and instance segmentation frameworks, A^2-FPN consistently and substantially improves the performance of baseline methods.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant No. 61771288, Cross-Media Intelligent Technology Project of Beijing National Research Center for Information Science and Technology (BNRist) under Grant No. BNR2019TD01022 and the research fund under Grant No. 2019GQG0001 from the Institute for Guo Qiang, Tsinghua University. We also thank for the funding support of Huawei Technologies Co. Ltd.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[2] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017. 2

[3] 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. 2, 6

[4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[5] 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. 1, 2, 6, 7

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5, 6

[7] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019. 2

[8] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 2

[9] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017. 2

[10] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017. 2

[11] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2

[12] 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. 2

[13] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. Augfpn: Improving multi-scale feature learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12595–12604, 2020. 6

[14] 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, 2, 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3, 6

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2, 4

[17] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6409–6418, 2019. 2

[18] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017. 2

[19] Yin Li and Abhinav Gupta. Beyond grids: Learning graph representations for visual recognition. In *Advances in Neural Information Processing Systems*, pages 9225–9235, 2018. 2

[20] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017. 2

[21] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems*, pages 1853–1863, 2018. 2

[22] 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, 6, 7, 8

[23] 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. 2

[24] 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. 2, 5

[25] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017. 2

[26] 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, 6, 7, 8

[27] 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

[28] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 3

[29] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in neural information processing systems*, pages 2277–2287, 2017. 2

[30] Pedro OO Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 2

[31] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016. 2

[32] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[33] 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

[34] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 4

[35] 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. 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, 3

[37] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3007–3016, 2019. 4, 5, 7

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[39] Yi Wang, Ying-Cong Chen, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Attentive normalization for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5094–5103, 2020. 3

[40] 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. 6

[41] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. A multipath network for object detection. *arXiv preprint arXiv:1604.02135*, 2016. 2

[42] Songyang Zhang, Xuming He, and Shipeng Yan. Latentgnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pages 7374–7383, 2019. 2