ATTENTION-BASED FEATURE AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Capturing object instances in different scales is a long-standing problem in the tasks of visual recognition, e.g., object detection and instance segmentation. The conventional way is to learn scale-invariant features, e.g., by summing up the feature maps output by different layers in the backbone. In this paper, we propose a novel and adaptive feature aggregation module based on attention where the attention parameters can be learned to handle different situations, e.g., adding shallow layers is learned to be conservative to mitigate the effect of noisy pixels, while for deep layers, it tends to be audacious to incorporate high-level semantics. To implement this module, we define two variants of attention: self-attention on the summed-up feature map, and cross-attention between two feature maps before summed up. The former uses the aggregated pixel values to capture global attention (to improve the feature for the next layer of aggregation), while the latter allows attention-based interactions between two features before aggregation. In addition, we apply multi-scale pooling in our attention module to reduce computational costs, and thus call the two variants Multi-Scale Self-Attention (MSSA) and Multi-Scale Cross-Attention (MSCA), respectively. We incorporate each variant into multiple baselines, e.g., the state-of-the-art object recognizer Cascade Mask-RCNN, and evaluate them on MSCOCO and LVIS datasets. Results show our significant improvements over baselines, e.g., boosting Cascade Mask-RCNN by 2.2% for AP^{box} and 2.7% for AP^{mask} on the MSCOCO dataset.

1 INTRODUCTION

Visual object detection and segmentation using deep learning models have achieved remarkable success in the past few years (Ren et al., 2015; Tian et al., 2019; Liu et al., 2021; Zhu et al., 2021), e.g., the average precision of detecting thousands of objects on the MSCOCO challenge has been boosted from 30% (Lin et al., 2014) to 61% (Xu et al., 2021) since 2015. However, there is still the long-standing problem about how to capture the object instances of different scales that limits the model performance. The common solution is to learn scale-invariant features such as by aggregating the feature maps output by different layers that builds a pyramidal structure (Lin et al., 2017a) as shown in Figure 1 (a). Existing methods implement this aggregation via brute-force stacking operations such as summing up (Lin et al., 2017a; He et al., 2017; Huang et al., 2021) and concatenation (Zhang et al., 2019; Ren et al., 2017). We argue that there are two issues. First, brute-force operations are sub-optimal for feature extraction. For example, summing up is on two pixels at the same position and does not consider any contextual relationships between distant pixels. Its aggregated result thus fail to capture the global view on the features from different layers. Second, brute-force operations can never be adaptive to different aggregation situations as they do not have learnable parameters.

In this paper, we tackle these issues by proposing an attention-based feature aggregation method. Given a feature map, we let each pixel to attend to every others based on either the pixel values from another-scale feature map (from the adjacent layer) or the summed-up values of two feature maps (from the current and adjacent layers, respectively). Computing pixel-level attention is costly. We thus apply a few multi-scale pooling layers after each layer and use only pooling features in the subsequent computation. As shown in Figure 1 (b), our attention-based aggregation has two variants: one is called Multi-Scale Self-Attention (MSSA) that computes the global self-attention on the sum of pooling features, and the other one is called Multi-Scale Cross-Attention (MSCS) that yields the global attention between two pooling features, i.e., one feature used as query and the other as key (and value), and then applies the sum-up operation to produce the output feature. MSSA learns the global

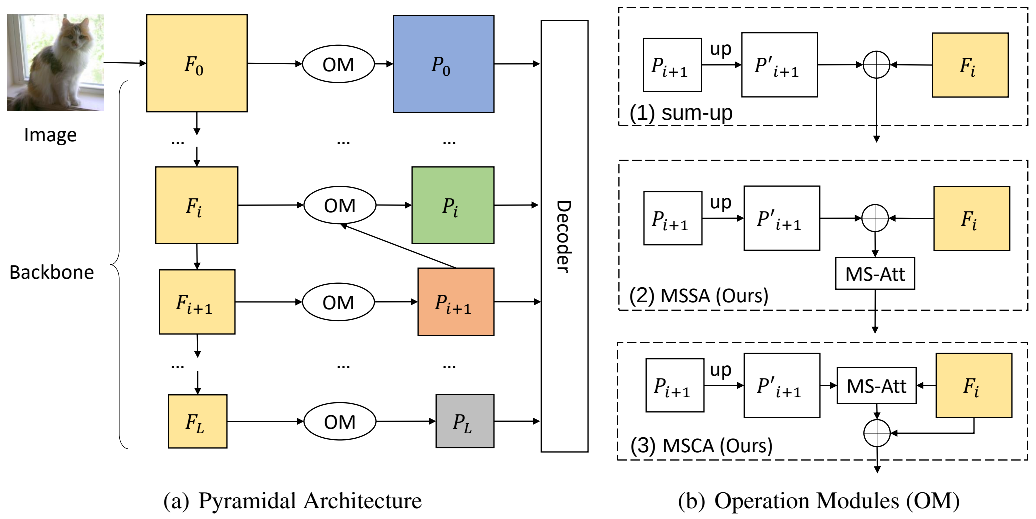


Figure 1: (a) is the network architecture that stacks the feature maps output by different layers using an aggregation operation represented by OM. (b) compares the conventional operation sum-up (used in a classical feature pyramid method FPN (Lin et al., 2017a)) with the two attention-based operations proposed in this paper: MSSA and MSCA. Please note that the input feature maps F_{i-1} and P_i are included in the illustration of OM in (b), while they are independently shown in (a) for a better visualization of the structure.

attention on the summed feature map where each of the values is equally from two feature maps (of different scales) and thus more convincing than that on any individual map. MSCA also learns the global attention but it allows the features of different scales to interactively derive the attention weights and then sum the weighted feature up to the shallow layer (as in FPN). Intuitively, MSSA produces the shared attention based on commonly interested pixels of two feature maps, while MSCA generates the specific attention for one feature map (from the deep layer) based on the global contexts found in the other one (from the shallow layer) and then sum it up to the feature map of the shallow layer. Both of them aim to capture global contexts across feature scales but define different sets of query, key and value. Through either of them, the result features will be fed into the aggregation module in the next layer (i.e., the shallower layer in FPN).

For evaluation, we incorporate our method into the state-of-the-art object recognizers, Cascade Mask-RCNN (Cai & Vasconcelos, 2018) and Mask-RCNN (He et al., 2017) using ResNet (He et al., 2016) as backbones, and conduct experiments on two large-scale benchmarks, MSCOCO (Lin et al., 2014) and LVIS (Gupta et al., 2019). Our results show that both of MSSA and MSCA significantly outperform the conventional aggregation operations—summing up or concatenation. For example, our MSCA-based models achieve 2.2% (1.9%) improvement for AP^{box}, and 2.7% (2.7%) for AP^{mask} on the MSCOCO (LVIS) dataset. Our main contributions in this paper thus can be summarized into two points: 1) two adaptive feature aggregation operations based on attention—MSSA and MSCA that can be learned to handle different situations of feature aggregation in the pyramidal networks, and 2) extensive experiments on two large-scale benchmarks and in-depth performance evaluations using multiple state-of-the-art object recognizers as baselines.

2 Related Works

Object Detection and Instance Segmentation. Object detection and segmentation are two fundamental computer vision tasks. Object detection aims to identify the category of object and localize each instance using a bounding box. Instance segmentation can be viewed as a special case of object detection, where the location of the instance is through a segmentation mask. The mainstream of object detection and instance segmentation can be broadly categorized into two groups: 1) proposal-based methods and 2) proposal-free methods. Proposal-based methods contain two stages: generating region proposals, and classifying the regions and predicting the location coordinates.

Faster R-CNN (Ren et al., 2015) is one of the most widely used method in this group. It first generates a sparse set of region proposals by region proposal network (RPN), and then encodes each region proposal into a fixed length vector via ROI Pooling, followed by region classifiers and bounding box regressors. Based on Faster RCNN, He et al. (2017) proposed Mask-RCNN by appending a mask predictor in parallel to the detection branch to do instance segmentation. The success of Faster R-CNN and Mask-RCNN also lead to many other excellent works for object detection (Lin et al., 2017a; Dai et al., 2017; Xizhou Zhu & Dai, 2019) and segmentation (Vu et al., 2021; Cai & Vasconcelos, 2018). In contrast, proposal-free methods do not need to generate proposals but use the whole image as input to identify the object instances. Popular methods include Yolo (Redmon et al., 2016), SSD (Li & Zhou, 2017), FCOS (Tian et al., 2019) and RetinaNet (Lin et al., 2017b) for object detection, and PolarMask (Xie et al., 2020), SOLO (Wang et al., 2020) and CondInst (Tian et al., 2020) for instance segmentation. Compared to proposal-based methods, proposal-free methods are usually less accurate but have faster inference speed—more suitable in real-time applications.

Feature Aggregation Methods. Deep CNN learns hierarchical features through multiple layers which capture the instance feature in different scales, e.g., deeper-layer features capture more semantics and are more helpful for recognizing larger instances. Lin et al. (2017a) proposed Feature Pyramid Network (FPN) to learn scale-invariant features by subsequently rescaling and summing up each layer features to its adjacent layer (the shallower layer). Based on FPN, there are many improved variants (Ren et al., 2017; Wu et al., 2020; Jeong et al., 2017; Woo et al., 2018; Zhang et al., 2018). Fu et al. (2017) and Cui (2018) applied element-wise production to aggregate two-layer features. Li & Zhou (2017), Zhang et al. (2019) and Ren et al. (2017) proposed to use concatenation operation instead (for aggregation). Liu et al. (2018) and Wu et al. (2020) improved FPN by using a dual feature pyramid that reverses the conventional FPN based on both sum-up and concatenation operations. Zhou et al. (2018) propose a dot-production aggregation block which generates high resolution feature map based on the inter-scale consistency nature across different layer features. All these pioneer works use brute-force feature aggregation operations (sum-up, concatenation, production, etc.) thus fail to capture or utilize any global contexts from another feature scale. In contrast, our attention-based operation provide a general and learnable way to achieve this.

Vision Transformers. CNN has limited receptive fields—can not capture the long-range contexts in the image. In contract, Vision Transformer (Dosovitskiy et al., 2021) (ViT) treats the input image as a global sequence of patch tokens, learns the global attention, and thus has the potential to solve the limitation of CNN. Based on ViT, there are many variants focused on the global design of the backbone (Wang et al., 2021a;b; Yuan et al., 2021; Yang et al., 2021). DETR (Carion et al., 2020) was proposed for object detection using a transformer decoder. It re-formulated the detection problem as an end-to-end dictionary lookup problem via learnable query embeddings. The problem of DETR is the high computational costs, e.g., it needs 500 epochs to converge to a satisfactory model (ours takes only 12 epochs on the same training data). Deformable-DETR is an improved version to accelerate the training by taking only a sparse set of feature maps. Its training epochs are reduced to 50. Compared to these DETR-based methods, our work is focused on the feature aggregation operations used in the encoder (not decoder). We take it as our future work to incorporate our encoder with DETR-based decoders.

3 Attention-based Feature Aggregation

As shown in Figure 1, our proposed attention-based module (either MSSA or MSCA) is used in the pyramidal network to aggregate the features output by two adjacent network layers. Here is the pipeline of the data flow. First, we input an image to the network to produce a set of feature maps F_i after each Conv layer (e.g., Residual module) where we use i to denote the location of the layer in the backbone. Second, we aggregate the deeper layer features P_{i+1} to F_i via the aggregation operation defined in OM. In Figure 1 (b), we show the difference between ours and the conventional operation, i.e., the sum-up used in FPN (Lin et al., 2017a). Finally, we feed aggregated features to the decoder to predict the class label and location (either bounding box or mask) for each instance in the image.

In the following subsections, we elaborate the implementation of MSSA and MSCA, including the details of computing attention and applying multi-scale pooling layers.

3.1 Multi-Scale Self-Attention (MSSA)

The operation of MSSA is shown in the second block of Figure 1 (b). The input features are respectively F_i from the shallow layer and P_{i+1} from the deep layer (right after the shallow layer). Please note that P_{i+1} is upsampled (i.e., using bilinear interpolation denoted as "Up") to the same size of F_i before fed into the attention computation module.

$$R_i = F_i + \text{Up}(P_{i+1}), \tag{1}$$

and the result R_i is then fed into the attention computation module as follows,

$$P_i = \text{MS-Att}(R_i), \tag{2}$$

where the details of MS-Att and Multi-Scale (MS) pooling are presented in the Section 3.3.

In MSSA, the input to the attention module is the summed-up features (a single set of feature maps), therefore, it is to learn the "global self-attention" that highlights the shared high value pixels activated in two layers (scales).

3.2 Multi-Scale Cross-Attention (MSCA)

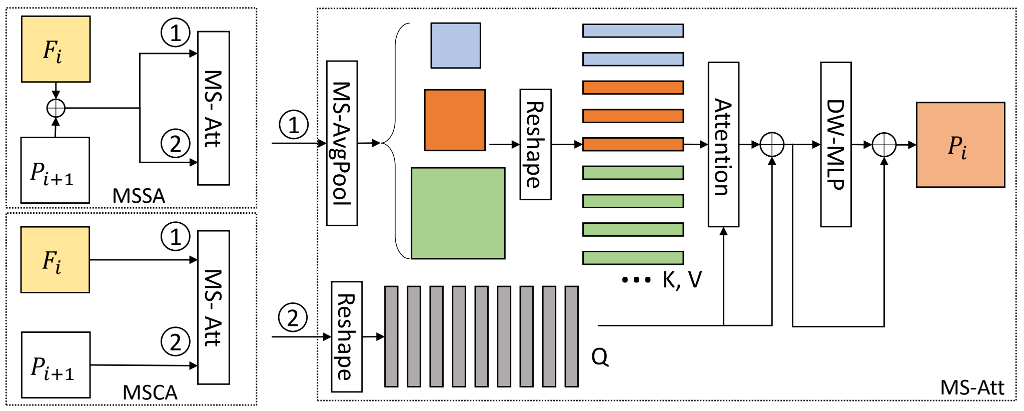
The operation of MSCA is presented in the third block of Figure 1 (b). The input features are the same with MSSA. The only different regarding the operation is that MSCA computes the cross attention between F_i and the upsampled P_{i+1} before the operation of sum-up. The detailed formulation is as follows,

$$S_i = \text{MS-Att}(F_i, \text{Up}(P_{i+1})), \tag{3}$$

where the results S_i is summed-up to the shallow layer feature as in follows,

$$P_i = F_i + S_i, \tag{4}$$

where the MS-Att operation between two features allows the deep feature $\text{Up}(P_{i+1})$ to highlight its valuable pixels based on the query of shallow feature F_i, and the sum-up operation yields the aggregated results. Compared to MSSA, MSCA does not generate a common attention but a specific attention for the deep layer based on the global contexts found in the shallow layer.



(a) Inputs (b) MS-Attention (MS-Att)

Figure 2: The framework of computing MS-Attention (MS-Att). Two kinds of inputs respectively correspond to the computations of MSSA and MSCA. The attention computation module is the same as in the Vision Transformer (Dosovitskiy et al., 2021). Specifically, two input features are transformed into Q, K and V first and then fed into a multi-head attention module followed by an MLP. Notably, we adopt a set of average pooling layers (each with a different stride) to generated multi-scale features before computing attention. This helps to reduce the computational costs and capture multi-scale contexts from each single feature map.

3.3 Multi-Scale Attention (MS-Att)

Computing the Attention. We follow the same attention module used in the Vision Transformer (Dosovitskiy et al., 2021) to implement our computation of attention. As illustrated in Figure 2 (b), the input into the "Attention" module is consisted of Query (Q), Key (K) and Value (V). The Query and Key share the same dimension d_k. The Key and Value have the same number of tokens. Figure 2 (a) shows the two sets of resource data to derive different values of Query, Key and Value, respectively, for MSSA and MSCA. To get the Query, we flatten the input feature on the lower branch (denoted by a circled 2) into a set of vectors f_q across the spatial axis.

To get the Key and Value, we conduct the similar flattening using the input feature on the upper branch and denote the resulted vectors as $f_{k,v}$. Please note that we introduce the multi-scale pooling in the end of this section. Then, we feed these vectors through their respective transformation layers (FC layers) and get:

$$Q = W_q * f_q, K = W_k * f_{k,v}, V = W_v * f_{k,v} \tag{5}$$

where W_q, W_k and W_v denote the parameters of three FC layers.

We compute the dot-product between Key and Query and normalize the results using the square root of the dimension of Key and Query (denoted as d_k). Then, we apply the Softmax function σ to obtain the weights on the Value. We multiply the Value by these weights to derive the attention (denoted as Att).

$$\text{Att} = \frac{\sigma(Q * K^T)}{\sqrt{d_K}} * V. \tag{6}$$

Typical transformer module require a fix-length learnable embedding for positional encoding, which is not suitable for dense prediction since the length of input is not fixed. Inspired by Chu et al. (2021) and Wang et al. (2021b), we learn position embedding directly from the input feature map by a depth-wise convolution layer. We sum the learned attention Att with the Query and feed the result into a Depth-Wise MultiLayer Perceptron encoder (DW-MLP) as follows,

$$p_q = f_q + \text{DW-MLP}(f_q + \text{Att}). \tag{7}$$

Typical transformer module require a fix-length learnable embedding for positional encoding, which is not suitable for dense prediction since the length of input is not fixed. Inspired by Chu et al. (2021) and Wang et al. (2021b), we learn position embedding directly from the input feature map by a depth-wise convolution layer. We follow the same setting of PVTv2 (Wang et al., 2021b) to implement the DW-MLP, where the input vectors go through a module consisted of a fully-connected layer, a depth-wise convolution layer, an activation layer and a fully-connected layer. Finally, we reshape p_q into 3D output feature maps P_i, and feed P_i into the next layer of feature aggregation (as illustrated in Figure 1 (a)).

Multi-Scale Pooling. The computation of attention is costly if input Query, Key and Value are of high dimensions. To control the cost, we apply a set of pooling layers before generating the vectors of Key and Value. Each layer has a different value of stride. Specifically in the i-th aggregation module, we apply 2 average pooling layers with strides as r_i and $\frac{r_i}{2}$, respectively, and then concatenate their flattened pooling results as the output.

3.4 Visual Recognizers

In this paper, we incorporate our modules into the state-of-the-art visual recognizers: Cascade Mask-RCNN (Cai & Vasconcelos, 2018) and Mask-RCNN (He et al., 2017) whose backbones are based on the FPN architecture. Cascade Mask-RCNN and Mask-RCNN are extended from Cascade-RCNN and Faster-RCNN (Ren et al., 2015), respectively, by appending an additional mask predictor in parallel to the detection branches. They have the shared structure in encoders: taking an image as input; passing it through multiple convolutional modules each of which produces a different scale of features; and stacking all features to produce the scale-invariant features. Their decoders have the similar structure. A region proposal network (RPN) is learned to generate region proposals that are then fed into object classifiers, bounding box regressors and mask predictors. Their difference lies in whether the prediction heads are trained sequentially in multi-stages, using the output to refine the next stage training. For both recognizers, we follow the same training pipelines where the model is optimized by the classification and localization losses. In the inference stage, we select the top-k regions with highest confidence scores as output, and then pass the predicted boxes to Non-Maximum Suppression (NMS) to drive the final evaluation results.

4 Experiments

4.1 Implementation Details

Datasets. We conduct experiments on MSCOCO-2017 (Lin et al., 2014) and LVIS-1.0 (Gupta et al., 2019) datasets. MSCOCO contains 80 categories with three splits, train set (115k images), validation set (5k images) and test set (20k images), and LVIS contains 1,203 categories with 100k images for training, 20k images for validation and 20k images for testing. For MSCOCO, we use train set to train the model, validation set in the ablation study and test set for the final evaluation and comparison. For LVIS, we train our models based on train set and evaluate on val set using the same values of hyper-parameters chosen in MSCOCO experiments. We evaluate the models using two metrics: bounding box AP (denoted as AP^{box}) and mask AP (denoted as AP^{mask}) where AP indicates the average precision.

Backbones and Recognizers. In our experiments, we use ResNet-50 and ResNet-101 (He et al., 2016) as backbone architectures for experiments, which are pre-trained on ImageNet classification task. We incorporate our modules into the state-of-the-art detectors: Cascade Mask-RCNN and Mask-RCNN. We use 5-level features for prediction (denoted as $\{P_0, P_1, P_2, P_3, P_4\}$), each of which is encoded by the feature aggregation module, with strides of 4, 8, 16, 32 and 64 compared with the original input image. For each level of the feature map, a region proposal network (RPN) is attached to generate proposals, The anchors used in RPN is set with single scale (8 pixels) and 3 aspect ratios (0.5, 1.0, 2.0). During inference, top-100 predictions with highest confidence scores are used for output. We set the NMS threshold as 0.7 in all experiments. For other training details, we follow the default settings as Cai & Vasconcelos (2018).

MSCA and MSSA Details. There are totally 4 MSSA/MSCA operation modules. For each input of MSCA or MSSA module, we first apply a 1x1 convolution layer to reduce the channels of input features into 256 before feature aggregation. In MS-Attention module, we apply 2 average pooling layers with different strides for multi-scale representation. Specifically, we set the strides r_0 to r_3 as (16, 8, 4, 2). For attention computation, we set the number of head as 4 and for DW-MLP block, we use GELU (Hendrycks & Gimpel, 2016) as the activation function and set expansion ratio in MLP as 2. We use layer normalization (Vaswani et al., 2017) to normalize the feature map before computing attention and set drop rates as 0 for all dropout layers.

Optimization Settings. We follow the same data augmentation strategies as (Cai & Vasconcelos, 2018), where each image is resized into $800 \times 1,333$ pixels, and a random size image patch will be cropped for training. The whole framework is implemented on the mmdetection platform (Chen et al., 2019). All the models are trained with the adamW optimization method (Hendrycks & Gimpel, 2016) for 12 epochs with 8 images per mini-batch. The initial learning rate is set as 1e-4, and it is reduced 10 times in 8-th and 11-th epoch. The value of betas in adamW is set as (0.9 and 0.99) with weight decay as 0.5. We conduct all the experiments on 8 A-100 cards.

4.2 Experiment Results

Cascade Mask-RCNN. Table 1 shows the results on the COCO val set by comparing our methods with baseline aggregation operations: sum-up and concatenation. It is clear that our proposed aggregation operations MSSA and MSCA consistently and significantly outperform the baselines on both backbones. For example, MSSA achieves 2.1% and 2.2% improvement for AP$_{box}$, and 2.4% and 2.5% for AP$_{box}$. MSCA achieves 2.2% and 2.5% improvement for AP$_{box}$ and 2.7% and 2.8% for AP$_{box}$ using ResNet-50. More encouragingly, 1) our best ResNet-50 model outperform the baseline ResNet-101 model regarding the performance, and moreover, its inference is faster than using ResNet-101; and 2) our overall performance improvements on the more challenging mask prediction are higher than those for box prediction (e.g., AP$_{mask}$ vs. AP$_{box}$).

Table 2 shows the comparison to state-of-the-art object detectors using ResNet-50 on the COCO test-dev set. We divide these methods into three groups due to the different encoders. Detectors with convolution-based decoders are mainly proposal-free algorithms and their decoders contains

Modules	Cascade Mask-RCNN							
	Backbone	FPS	AP^{box}	AP$^{box}_{50}$	AP$^{box}_{75}$	AP^{mask}	AP$^{mask}_{50}$	AP$^{mask}_{75}$
Sum-up	ResNet-50	13.0	41.2	59.4	44.8	35.9	56.6	38.4
Concat.		12.8	40.9	59.1	44,3	35.8	56.2	38.4
MSSA		11.9	43.3 +2.1	62.4 +3.0	**47.2** +2.4	38.3 +2.4	59.9 +3.3	41.2 +2.8
MSCA		11.9	**43.4** +2.2	**62.8** +3.4	47.1 +2.3	**38.6** +2.7	**60.1** +3.5	**41.6** +3.2
Sum-up	ResNet-101	10.9	42.9	61.0	44.6	37.3	58.2	40.1
Concat.		10.0	42.4	60.3	46.2	37.0	57.7	40.1
MSSA		9.6	**44.4** +1.5	**63.3** +2.3	**48.5** +3.9	**39.2** +1.9	**60.8** +2.6	**42.2** +1.9
MSCA		9.6	44.3 +1.4	63.3 +2.3	48.2 +2.8	39.0 +1.7	60.7 +2.5	42.0 +1.9

Table 1: Our experiments results compared to those of baseline operations. FPN uses summing-up and FPN-cat replaces summing up with concatenation, for feature aggregation. All models are trained on the MSCOCO train set for 12 epochs, and evaluated on MSCOCO val set with 8 A-100 cards.

a few Conv layer before the prediction layers (Tian et al., 2019; Lin et al., 2017b; Li et al., 2021). Detectors with transformer-based decoders generate a set of query vectors using a sequence of attention modules (Zhu et al., 2021; Carion et al., 2020). Our recognizers (He et al., 2017; Cai & Vasconcelos, 2018) belongs to the region-based group which makes the prediction without any additional layers or modules—the simplest decoder. We can see that our models outperform all the others using convolution or region decoders, and achieves comparable results as those with transformer-based decoders. We highlight that our method is applied to the encoder, and is orthogonal to the decoder. We leave the combination with transformer-based decoders as one of our future work, due to the expensive computation (e.g., DETR (Carion et al., 2020) using 500 epochs vs. ours using 12 epochs to achieve the model convergence on the same dataset).

Object Detectors	Decoder	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
RetinaNet (Lin et al., 2017b)	Convolution	36.5	55.4	39.1	20.4	40.3	48.1
FCOS (Tian et al., 2019)		37.1	55.9	39.8	21.3	41.0	47.8
FCOS-impr (Tian et al., 2019)		38.5	57.4	41.4	22.3	42.5	49.8
FSAF (Zhu et al., 2019)		37.4	56.8	39.8	20.4	41.1	48.8
Reppoints (Yang et al., 2019)		39.1	60.0	42.1	22.1	41.9	48.4
ATSS (Zhang et al., 2020)		39.6	58.2	42.9	23.3	42.4	48.5
GFLv1 (Li et al., 2020)		40.2	58.6	43.4	23.0	44.3	53.0
Reppoints v2 (Chen et al., 2020)		41.3	59.8	44.0	24.0	44.6	54.7
PolarNet (Wu et al., 2021)		39.6	57.9	42.9	23.2	43.5	51.8
GFLv2 (Li et al., 2021)		41.1	58.8	44.9	23.5	44.9	53.3
DETR*(Carion et al., 2020)	Transformer	36.2	57.0	37.4	16.3	39.2	53.9
DETR-full†(Carion et al., 2020)		43.3	63.1	45.9	22.5	47.3	61.1
Deform-DETR*(Zhu et al., 2021)		43.8	62.6	47.7	26.4	47.1	58.0
CenterMask (Lee & Park, 2020)	Region	41.0	-	-	24.8	45.1	54.5
TSD (Song et al., 2020)		42.0	59.9	45.4	24.3	45.5	57.1
CasMRCN (Cai & Vasconcelos, 2018)		41.4	59.7	45.1	23.3	44.1	52.7
CasMRCN-MSSA (ours)		43.5	63.1	47.3	25.4	46.0	55.4
CasMRCN-MSCA (ours)		43.5	63.1	47.3	25.1	46.0	55.9

* This entry reports the model is trained with 50 epochs.
† This entry reports the model is trained with 500 epochs.

Table 2: Comparing to state-of-the-art models for the object detection task. All models 1) use ResNet-50 backbone; 2) are trained on the MSCOCO train set; and 3) are evaluated on the MSCOCO test-dev set. *Please note that "CasMRCN" is the abbreviation of "Cascade Mask-RCNN".*

Mask-RCNN. Table 3 shows the results of MSSA and MSCA in Mask-RCNN compared to the baseline Mask-RCNN (using sum-up operation). We evaluate all models on the validation set of MSCOCO and LVIS. On the MSCOCO dataset, the MSSA and MSCA outperform baseline by 0.7% and 0.5% on AP$_{box}$, and 1.1% and 0.9% on AP$_{mask}$. On the LVIS dataset, they outperform baseline by 1.6% and 1.9% on AP$_{box}$, and 2.4% and 2.7% on AP$_{mask}$. In terms of datasets, our method obtains higher improvement on the LVIS dataset. In terms of tasks, ours improves more for the segmentation task. These verify that the cross-scale global contexts captured by our attention modules are more helpful for tackling more challenging problems.

Modules	Mask-RCNN							
	Benchmarks	FPS	AP^{box}	AP$^{box}_{50}$	AP$^{box}_{75}$	AP^{mask}	AP$^{mask}_{50}$	AP$^{mask}_{75}$
Sum-up	COCO	16.1	38.0	58.6	41.4	34.4	55.1	36.7
MSSA		14.8	**38.7** +0.7	**60.4** +0.8	**41.8** +0.4	**35.5** +1.1	**57.3** +2.2	**37.8** +1.1
MSCA		14.8	38.5 +0.5	60.0 +0.4	41.5 +0.1	35.3 +0.9	56.8 +1.7	37.6 +0.9
Sum-up	LVIS	14.0	22.5	36.9	23.8	21.7	34.3	23.0
MSSA		12.3	24.1 +1.6	**40.1** +3.2	25.3 +1.5	24.1 +2.4	37.8 +2.5	25.4 +1.6
MSCA		12.3	**24.4** +1.9	40.0 +3.1	**25.9** +2.1	**24.4** +2.7	**37.9** +3.6	**25.9** +2.9

Table 3: Experiments results of proposed modules based on Mask-RCNN frameworks. All the models are trained with train set of the corresponding datasets for 12 epochs and evaluated on val set. ResNet-50 is used as backbone which is pre-trained on ImageNet.

4.3 Ablation Study

Here we conduct an ablation study to evaluate the proposed method regarding the following factors: aggregating how many layers, using different pooling layers and using different types of backbones in the encoder. We use MSSA plugged in Cascade Mask-RCNN for all experiments. We train the models on the MSCOCO train set and evaluate them on the val set.

Aggregation Operations. Table 4 shows the results of gradually replacing the conventional sum-up operation with MSSA from deep to shallow layers in the backbone. Row 1 is the baseline of using only sum-up in all layers of aggregation. Comparing it with Row 2, we see that using a single MSSA module (replacing the sum-up on the top layer of aggregation) boosts 1.3% and 1.9% for AP$_{box}$ and AP$_{mask}$, respectively. These improvements are increased when using more MSSA modules.

Modules	Cascade Mask-RCNN						
	OMs	AP^{box}	AP$^{box}_{50}$	AP$^{box}_{75}$	AP^{mask}	AP$^{mask}_{50}$	AP$^{mask}_{75}$
Sum-up	-	41.2	59.4	44.8	35.9	56.6	38.4
MSSA	(P_3)	42.5	61.7	46.2	37.8	59.2	40.6
MSSA	(P_2, P_3)	42.7	61.9	46.3	38.0	59.5	40.9
MSSA	(P_1, P_2, P_3)	43.1	62.5	46.7	38.3	59.8	41.3
MSSA	(P_0, P_1, P_2, P_3)	**43.4**	**62.8**	**47.1**	**38.6**	**60.1**	**41.6**

Table 4: Using different numbers of MSSA modules. All models are trained on the train set of MSCOCO for 12 epochs and evaluated on the val set. We use ResNet-50 as the backbone.

Different Pooling Layers. In our MS-Att, we apply two multi-scale pooling layers to reduce the size of Key and Value as well as to extract multi-scale feature representation from each single feature map. In this ablation study, we try another two types of pooling layers, single-scale pooling and adaptive pooling, and show the results in Table 5. The single-scale pooling layer have the stride as (16,8,4,2) to compute attention. The adaptive pooling layer generates an $n \times n$ feature map with arbitrary input size, and we set $n = 7$ in our experiments. From the table, we can see that these two alternatives achieve comparable results to the multi-scale version, e.g., for AP^{box} (AP^{mask}) they are slightly lower, respectively, by 0.2% (0.5%) and 0.3% (0.4%). These two alternatives can be served as fast models due to the fast inference speed and good performance.

Modules	Cascade Mask-RCNN							
	Pooling Type	FPS	AP^{box}	AP$^{box}_{50}$	AP$^{box}_{75}$	AP^{mask}	AP$^{mask}_{50}$	AP$^{mask}_{75}$
MSSA	Single-Scale	13.0	43.2	62.7	47.0	38.1	60.1	41.0
MSSA	Adaptive	14.1	43.1	62.3	46.8	38.2	59.4	41.1
MSSA	Multi-Scale	11.9	**43.4**	**62.8**	47.1	**38.6**	**60.1**	**41.6**

Table 5: Using different types of pooling layers for MSSA. All models are trained on the train set of MSCOCO for 12 epochs and evaluated on the val set. We use ResNet-50 as the backbone.

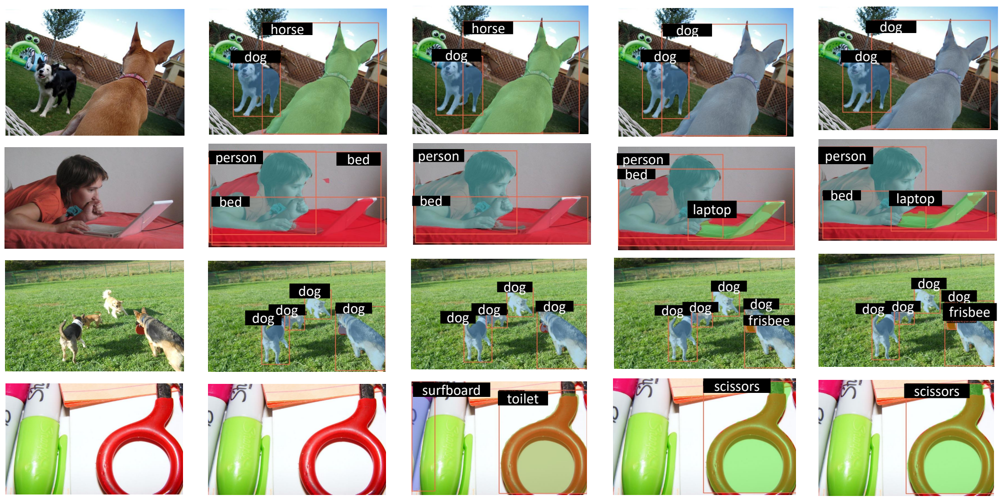
Transformer-Based Backbone. In this section, we replace the ResNet with a transformer-based backbone Swin-T (Liu et al., 2021) to evaluate the performance of our MSSA. We report the results in Table 6. On this stronger backbone, we still obtain clear and consistent improvement using MSSA over the baseline (sum-up), e.g., 0.8% for both AP^{box} and AP^{mask}.

Modules	Cascade Mask-RCNN							
	Backbones	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Sum-up	Swin-T	11.2	48.0	66.8	51.6	41.5	64.1	44.9
MSSA	Swin-T	9.8	**48.8**	**67.9**	**52.7**	**42.3**	**65.0**	**45.6**

Table 6: Using the vision-transformer-based backbone called Swin-T and plugging MSSA in. All models are trained on the train set of MSCOCO for 12 epochs and evaluated on the val set.

4.4 QUALITATIVE RESULTS

We make the qualitative comparison between our methods (MSSA and MSCA) and the baseline methods (sum-up and concatenation) and show the results in Figure 3. The models are trained on the MSCOCO train set using ResNet-101 as the backbone. From the examples, we can see that the improvements by MSSA and MSCA are mainly due to the corrected recognition of small or confusing objects in the image. For example on Row 3, "frisbee" has a very limited visibility in the image—small, occluded and having a confusing color. Both baseline models fail to detect it. While our proposed MSCA and MSSA capture it. We believe that this is because MSCA and MSSA use attention to capture different scales of information in surrounding contexts (such as from "dog" and "grass") and can achieve a better "understanding" of the objects even in small scales, e.g., the "frisbee" can be predicted if the model attends to the pixels close to the "mouth" region of the "dog" but different with the pixels of "grass".



(a) Source Image (b) Concatenation (c) Sum-up (d) MSCA (e) MSSA

Figure 3: The qualitative comparison between the proposed feature aggregation modules (MSCA and MSSA) and baselines (sum-up and concatenation). All models are based on Cascade Mask-RCNN uniformly trained on the MSCOCO train set and evaluated on the val set. We use ResNet-101 as the backbone. We set the confidence threshold for visualization as 0.6 (zoom-in for a better view).

5 CONCLUSIONS

In this paper, we present two adaptive attention-based operations named MSSA and MSCA, for improving the feature aggregation in pyramidal models. We aim to not only capture global contexts in different scales of feature maps and but also aggregate them based on the extracted global attention that helps to mitigate noisy feature pixels from the shallow layer. We validate that object detectors and segmenters with MSSA or MSCA plugged in can achieve significant and consistent improvements over the baselines on both MSCOCO and LVIS datasets. In the future work, we plan to explore more in incorporating our modules into stronger backbones and detectors, such as the transformer-based DETR and Deformable DETR, meanwhile not incurring additional computational costs.

6 Ethics Statement

Ethics Statement

Computational costs. Our methods are based on deep learning models and require intensive usage of computation resource, it is not climate-friendly. It calls for future research into proposing more effective training strategies that can train deep learning models without GPU.

Privacy issues. In our experiments, all the datasets and platform are publicly available, and thus we do not have privacy issues.

Licenses. We use the open-source platform mmdetection (Chen et al., 2019) which is under Apache License 2.0.

Datasets. We use two public datasets in our paper:LVIS-1.0 (Gupta et al., 2019) and MSCOCO (Lin et al., 2014). Both datasets are available from their official websites, and allowed to use for non-commercial usage.

Reproducibility Statement

We have included all the implementation details in Sec 4. We will definitely release all the code and models if the paper is accepted.

References

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

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.

Yihong Chen, Zheng Zhang, Yue Cao, Liwei Wang, Stephen Lin, and Han Hu. Reppoints v2: Verification meets regression for object detection. In *NeurIPS*, 2020.

Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

Lisha Cui. Mdssd: Multi-scale deconvolutional single shot detector for small objects. In *arXiv preprint arXiv:1805.07009*, 2018.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. In *arXiv preprint arXiv:1701.06659*, 2017.

Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. FaPN: Feature-aligned pyramid network for dense image prediction. In *ICCV*, 2021.

Jisoo Jeong, Hyojin Park, and Nojun Kwak. Enhancement of ssd by concatenating feature maps for object detection. In *arXiv preprint arXiv:1705.09587*, 2017.

Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *CVPR*, 2020.

Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In *NeurIPS*, 2020.

Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In *CVPR*, 2021.

Zuoxin Li and Fuqiang Zhou. Fssd: Feature fusion single shot multibox detector. In *arXiv preprint arXiv:1712.00960*, 2017.

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 *ECCV*, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017a.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017b.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *CVPR*, 2020.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.

Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Thang Vu, Kang Haeyong, and Chang D Yoo. Scnet: Training inference sample consistency for instance segmentation. In *AAAI*, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021a.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021b.

Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *ECCV*, 2020.

Sanghyun Woo, Soonmin Hwang, and In So Kweon. Stairnet: Top-down semantic aggregation for accurate one shot detection. In *WACV*, 2018.

Xiongwei Wu, Daoxin Zhang, Jianke Zhu, and Steven CH Hoi. Single-shot bidirectional pyramid networks for high-quality object detection. *Neurocomputing*, 2020.

Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Polarnet: Learning to optimize polar keypoints for keypoint based object detection. In *ICLR*, 2021.

Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020.

Stephen Lin Xizhou Zhu, Han Hu and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.

Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *ICCV*, 2021.

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021.

Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *ICCV*, 2019.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

Jialiang Zhang, Xiongwei Wu, Jianke Zhu, and Steven CH Hoi. Feature agglomeration networks for single stage face detection. *Neurocomputing*, 2019.

Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020.

Peng Zhou, Bingbing Ni, Cong Geng, Jianguo Hu, and Yi Xu. Scale-transferrable object detection. In *CVPR*, 2018.

Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *CVPR*, 2019.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.