Pattern Recognition 138 (2023) 109365

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

AFI-GAN: Improving feature interpolation of feature pyramid networks via adversarial training for object detection

Seong-Ho Lee, Seung-Hwan Bae*

Vision & Learning Laboratory, Inha University, South Korea

ARTICLE INFO

Article history:
Received 15 June 2022
Revised 20 November 2022
Accepted 25 January 2023
Available online 27 January 2023

Keywords:
Multi-scale feature representation
Object detection
Adversarial training
Feature up-sampling

ABSTRACT

Recent convolutional detectors learn strong semantic features by generating and combining multi-scale features via feature interpolation. However, simple interpolation incurs often noisy and blurred features. To resolve this, we propose a novel adversarially-trained interpolator which can substitute for the traditional interpolation effortlessly. In specific, we design AFI-GAN consisting of an AF interpolator and a feature patch discriminator. In addition, we present a progressive adversarial learning and AFI-GAN losses to generate multi-scale features for downstream detection tasks. However, we can also finetune the proposed AFI-GAN with the recent multi-scale detectors without the adversarial learning once a pre-trained AF interpolator is provided. We prove the effectiveness and flexibility of our AF interpolator, and achieve the better box and mask APs by 2.2% and 1.6% on average compared to using other interpolation. Moreover, we achieve an impressive detection score of 57.3% mAP on the MSCOCO dataset. Code is available at https://github.com/inhavl-shlee/AFI-GAN.

1. Introduction

Due to the advances in deep convolutional neural networks (CNNs), the convolutional object detectors [1,2] have shown the remarkable accuracy improvement. To improve the robustness over the scale variations of objects, the state-of-the-art detectors are constructed based on the multi-scale feature representation. For multi-scale object detection, some architectures [3–6] are designed and used for base networks (i.e. backbone) of detectors. Among them, a feature pyramid network (FPN) [3] develops top-down feature propagation and provides the way to use multi-scale features across all scale levels. For boosting lower layer features, path aggregation FPN (PAFPN) [4] designs the extra bottom-up pathway following the top-down pathway.

Inspired by these works, many multi-scale feature methods [5–7] for object detection have been also presented. In specific, [5–7] design additional feature propagation pathway for better feature representation. Also, detection methods [8,9] are developed for using multi-scale features effectively for better detection.

In those works based on multi-scale feature representation, the main process is to resize feature maps before propagating feature maps to the next scale level. In general, the bottom-up and top-down feature maps are downsampled and upsampled, respectively. As a result, the feature resolution at the previous level can be fitted to it at the next level on the same pathway, but also combined with features forwarded from the different pathway. However, simple interpolation methods (e.g. nearest neighbor and bilinear) are still exploited when increasing the feature resolution. As shown in Wang et al. [10], these interpolations cause noisy and blurred feature maps. Using these features as an input of a detector also degrades the detection accuracy.

To resolve this problem, we aim at developing a novel feature interpolator which can produce up-sampled features used for multi-scale feature learning. In order to learn this interpolator via adversarial learning, we propose a new generative adversarial network (AFI-GAN) consisting of an AF interpolator and a feature patch discriminator. Furthermore, we present a new integral loss which can make our AFI-GAN appropriate more for multi-task learning to multi-scale object detection and segmentation.

For learning AFI-GAN, we perform adversarial training between the AF interpolator and the feature patch discriminator with an adversarial feature up-sampling loss. As a result, the AFI-GAN can learn a generic up-sampled feature representation from an input feature. Subsequently, we incorporate the AFI-GAN with a multi-scale feature extractor by replacing the interpolation module with the AF interpolator. Then, for learning the multi-scale AF extractor, the adversarial training between the multi-scale AF extractor and the feature patch discriminator is followed by using the integral loss including object detection and adversarial feature up-sampling losses.

* Corresponding author.
E-mail address: shbae@inha.ac.kr (S.-H. Bae).

https://doi.org/10.1016/j.patcog.2023.109365
0031-3203/© 2023 Elsevier Ltd. All rights reserved.

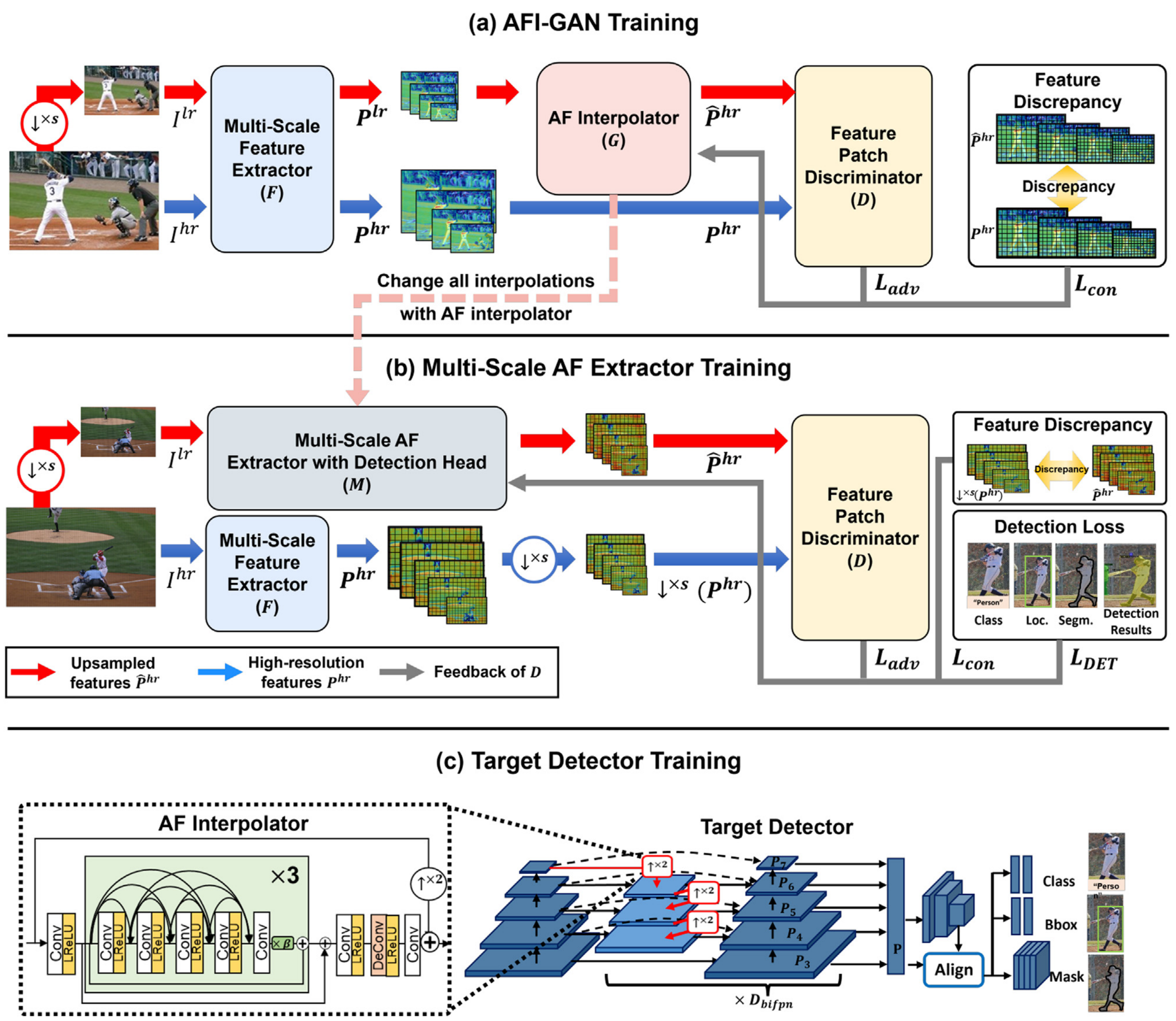

Fig. 1. Proposed AFI-GAN training procedures which consisting of (a) AFI-GAN training detailed in Section 4.1, (b) multi-scale AF extractor training described in Section 4.2, and (c) target detector training described in Section 4.3. Note that the multi-scale network and head network of a target detector can be replaced with the other one as shown in Fig. 3 and Section 4.3.

Note that learning the interpolator that upsamples a feature map is still challenging because up-sampled outputs should enhance or preserve object locality and discriminability after the interpolation. To address this, our core idea is to leverage the multi-scale feature network (e.g. FPN [3], PAFPN [4], and BiFPN [6]) as a target feature generator during these AFI-GAN training. To this end, we feed original and downsampled images to the feature network, and then extract feature maps at each pyramid level for both images. Then, multi-scale features from the downsampled image are forwarded to the AF interpolator, and the up-sampled ones are then compared with the corresponding features extracted from the original image by the feature patch discriminator. Finally, the AF interpolator is trained adversarially with the feedback of the feature patch discriminator as in Fig. 1.

Once the AFI-GAN is trained, we can use it for interpolation directly. However, the joint training with a target detector allows our AF interpolator to be more suitable for the specific detection task. In practical, we emphasize that the pre-training of AFI-GAN shown in Fig. 1 can be omitted because the reuse of the AFI-GAN trained with other backbones is also possible. This indicates that our AFI-GAN can learn generalized up-sampled features and have high flexibility over different backbones. To prove our AFI-GAN, we have implemented several different versions of target detectors by using RetinaNet [11], Mask R-CNN [12], FCOS [13], and CenterMask [14] representing anchor-based one-stage or two-stage detectors, and anchor-free detectors. We have achieved improvements of box and masks APs by 2.2% and 1.6% on average compared to the same detectors using other interpolation. We have also shown the extensive ablation study with different backbones and detectors.

The summarization of the main contributions of this paper is (i) proposition of a novel AFI-GAN to generate up-sampled features for multi-scale feature learning and be applicable easily for other convolutional detectors; (ii) proposition of the AFI-GAN losses for generating the high quality of up-sampled features and improving detection accuracy together; (iii) proposition of a progressive adversarial learning to improve the interpolation ability of AFI-GAN step-by-step for a specific detection task as shown in Algorithm 1; (iv) extensive evaluation and consistent mAP improvement by implementing various versions of AFI-based detectors with the recent multi-scale feature extractors (i.e. FPN [3], PAFPN [4], and BiFPN

Algorithm 1: Proposed AFI-GAN training.

```
Input  : Mini-batch of image set $I = \{I_i^{hr}, I_i^{lr}\}_{i=1}^N$ and
         pre-trained $F$.
Output: Trained AF interpolator $G$ and detector $T$
1  // Sec 4.1: Learning $G$ and $D$ while freezing $F$
2  for $k_1 = 1, \ldots, K_1$ do
3      Extract $(\hat{\mathbf{P}}_{k_1}^{hr}, \mathbf{P}_{k_1}^{hr}) \leftarrow (G(F(I_{k_1}^{lr})), F(I_{k_1}^{hr}))$
4      Train $D$ by maximizing $L_{adv_D}(\hat{\mathbf{P}}_{k_1}^{hr}, \mathbf{P}_{k_1}^{hr})$ in Eq. (1)
5      Train $G$ by minimizing $L_{AF}(\hat{\mathbf{P}}_{k_1}^{hr}, \mathbf{P}_{k_1}^{hr})$ in Eq. (2)
6  // Sec 4.2: Learning $M$ embedded $G$ and $D$ while freezing $F$
7  for $k_2 = 1, \ldots, K_2$ do
8      Extract $\hat{\mathbf{P}}_{k_2}^{hr}, \mathbf{P}_{k_2}^{hr} \leftarrow M(G(I_{k_2}^{lr})), F(I_{k_2}^{hr})$
9      Train $D$ by maximizing $L_{adv_D}$ $(\hat{\mathbf{P}}_{k_2}^{hr}, \downarrow^{\times s}(\mathbf{P}_{k_2}^{hr}))$ by Eq. (1)
10     Train $M$ by minimizing $L_{INT}(\hat{\mathbf{P}}_{k_2}^{hr}, \downarrow^{\times s}(\mathbf{P}_{k_2}^{hr}))$ by Eq. (3)
11 // Sec 4.3: Learning $T$ with $G$
12 for $k_3 = 1, \ldots, K_3$ do
13     Train $T$ by minimizing $L_{DET}(T(G(I_{k_3}^{hr})))$ as in Sec. 4.3.
```

[6]) and recent detection networks (i.e. Faster-RCNN [3], RetinaNet [11], Mask R-CNN [12], Cascade R-CNN [15], FCOS [13], and CenterMask [14]).

2. Related work

We discuss previous works on deep object detection, multi-scale representation, and interpolation methods, which are related to our work.

Deep object detection There are two main approaches in the recent deep object detection, which are anchor-based and anchor-free object detection. The anchor-based object detection has been flourishing since deep convolutional detectors using anchors [16] showed significant improvement on several detection benchmarks [17]. In addition, the anchor-based detection can be divided into two-stage and one-stage methods. The two-stage detection methods first generate regions of interest (RoIs) with the region proposal network, and then refine RoIs with the followed R-CNN. Mask R-CNN [12] attaches a mask head to the two-stage detector [16] for accurate pixel-wise segmentation. Multi-stage detection methods [15] can refine RoIs iteratively in a cascade manner. HON [18] exploits a hierarchical objectness network to improve the localization quality of the region proposal network. Peng et al. [19] attach a context-aware module into the detection head network in order to improve detection accuracies using high-level semantic information. On the other hand, the one-stage detectors predict detections directly without the region proposal. SSD [20] produces predictions of different scales from feature maps of different scales using the predefined anchors. RetinaNet [11] addresses the class imbalance problem by introducing a focal loss.

Recently, the anchor-free object detection methods reduce the computational complexity and hyper-parameters for anchor generation. FCOS [13] introduces the centerness branch to refine center areas of a box. CenterMask [14] adds a spatial attention-guided mask branch to FCOS. ATSS [2] introduces the automatic positive and negative sample selection method based on a statistical approach for objects in order to improve accuracies of both FCOS [13] and RetinaNet [11].

Furthermore, there are some efforts [21] to apply adversarial learning for improving object detection. However, they exploit an adversarial loss in order to improve the RoI feature representation for small objects. Compared to these works, our works more focus on enhancing the whole image feature maps.

Multi-scale representation In order to achieve the robustness to object scale variation and the better detection, feature pyramids (*or* multi-scale features) are employed for the multi-scale feature representation. SSD [20] combines multi-scale features extracted from a bottom-up pathway. MDFN [9] concatenates multi-scale features to use the semantic and contextual information by deep features. Gated CNN [8] integrates multi-scale features to provide robust features for accurate object detection. On the other hand, recent works [3–6] learn multi-scale features from several different pathways. FPN [3] first shows that using bottom-up and top-down features is effective for scale-invariant detection. PANet [4] further introduces an extra bottom-up pathway. Inspired by these works, many network architectures using cross-scale [5,6] have been presented. NAS-FPN [5] discovers a suitable architecture for feature pyramid by using the Neural Architecture Search algorithm [22]. BiFPN [6] applies top-down and bottom-up feature fusion repeatedly with bi-directional features. mSODANet [23] introduces the bi-directional feature aggregation module to refine the multi-scale feature from BiFPN [6] and to improve object detection in aerial images. Still, all these methods exploit a naïve interpolation method when increasing feature resolution. Therefore, we focus on developing a feature scaling-up method for learning feature pyramid more accurately. Remarkably, our method can be applicable easily for all these previous methods by replacing the interpolation method with ours.

Interpolation methods Interpolation methods can be categorized into traditional interpolation and learning-based interpolation. Traditional interpolation methods upsample an input image based on its own value without learning an additional model. Therefore, these methods are intuitive and easy to implement. For instance, a nearest-neighbor interpolation copies the value from the nearest pixel without consideration of other pixels. Bilinear and bicubic interpolations compute the value by evaluating the values of nearby pixels. However, they tend to occur noisy and blurred results as shown in Wang et al. [10]. Learning-based interpolation exploits additional weight and bias parameters in order to upscale an input resolution. Transposed convolution, also called deconvolution [24], upsamples an input by calculating the weighted value from the learned parameters and pixel values. A sub-pixel layer [25] performs upsampling by expanding the channels of the output features and then rearranging these points. Due to the usage of additional parameters for upsampling, the learning-based interpolation methods show better results compared to traditional interpolations. Compared to the existing interpolation methods, our works more focus on improving a detection accuracy by generating upsampled feature maps for multi-scale feature representation. To this end, we design the AF interpolator with both bilinear interpolation and deconvolution to improve the quality of the upsampled feature map. Moreover, we propose an adversarial learning method with a detection loss in order to increase feature locality and discriminability after enlarging the feature map.

3. Adversarial feature interpolation GAN

For generating up-sampled features at any scale which can be applicable for multi-scale feature learning, we first design AFI-GAN consisting of an AF interpolator and a feature patch discriminator as discussed in Section 3.1. For training the AFI-GAN from scratch, we exploit the existing multi-scale feature network as a target feature generator, and match up-sampled features from an AF interpolator with the corresponding features of the same resolution from the target generator as mentioned in Section 3.2. Furthermore, we present an adversarial progressive learning to avoid it overfitted as shown in Section 4. However, note again that we can train a

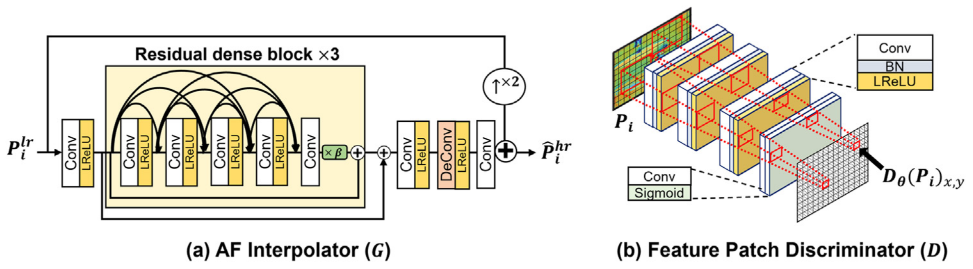

Fig. 2. Proposed AFI-GAN architecture consisting of (a) an AF interpolator and (b) a feature patch discriminator.

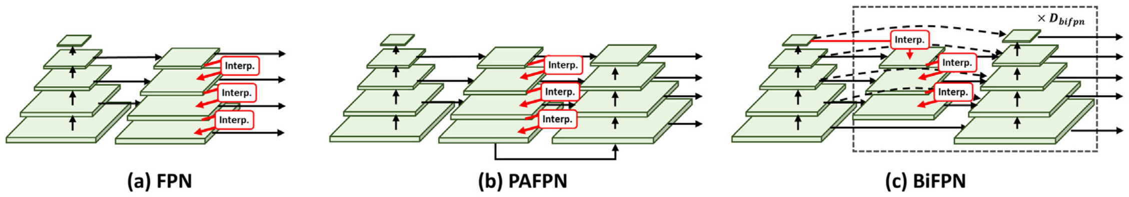

Fig. 3. Multi-scale feature extractors for multi-scale feature learning. Figure 3 shows the structures of (a) a feature pyramid network (FPN) [3], (b) a path aggregation network (PAFPN) [4], and (c) a bi-directional feature pyramid network (BiFPN) [6], respectively. In the interpolation module (Interp.) of each network, we replace a nearest neighbor interpolation with our AF interpolator. In our implementation, we set the number of BiFPN layers D_{bifpn} to 3 and 7 for ResNet [30]-BiFPN and Swin Transformer [31]-BiFPN, respectively.

target detector embedded with the AF interpolator at once as in Section 4.3 if a pre-trained AF interpolator by using any multi-scale feature network is provided. Thus, some pre-training phases to warm-up the AFI-GAN can be omitted in practice.

3.1. Overall architecture

As shown in Fig. 2, the AF interpolator G generates an up-sampled feature map $\hat{P}^{hr}$ for an input feature map of lower resolution feature P^{lr}. On the other hand, the feature patch discriminator D identifies between patches extracted within the up-sampled feature $\hat{P}^{hr}$ and high-resolution feature P^{hr} (For more details of P^{hr}, refer to Section 3.2).

For the interpolator G, we feed P^{lr} of any resolution to a 3×3 convolution and a Leaky ReLU activation (LReLU) layers ($\alpha = 0.2$). After them, we add 3 consecutive residual dense blocks [26] and a shortcut connection which bypasses them in order to exploit the more informative representation for up-sampling.[1] Concretely, each residual dense block contains 5 densely connected 3×3 convolution layers with growth rate of 32, 4 Leaky ReLU layers, and one shortcut connection. For stabilizing the training, residual scaling [27] ($\beta = 0.2$) is applied before the identity mapping. Then, one convolution and one deconvolution blocks are followed to scale-up the feature resolution by a factor of 2. Lastly, we add a 3×3 convolution layer and an additional shortcut connection between the deconvolved feature and upsampled input feature by the bilinear interpolation. We exploit the deconvolved feature in order to up-sample features. However, we additionally upsample the feature by using the bilinear interpolation in order to exploit an input feature representation fully with a low computational cost. As a result, we can generate the high quality of upsampled feature maps by using both features.

The discriminator D, which is a modified version of a patch discriminator [28], convolves $\hat{P}^{hr}$ or P^{hr} by using three convolution blocks with 512, 1024, and 1024 channels. Here, each block contains a 3×3 convolution, a batch normalization, and a Leaky ReLU activation layer ($\alpha = 0.2$). Then, the class per feature map pixel is predicted by a 3×3 convolution and a sigmoid activation function. As an activation function of AFI-GAN, we choose LeakyReLU due to its robustness of optimization [29]. In addition, it is often used for training GAN methods [28].

[1] A comparison of different G architectures is given in Table 2.

3.2. Adversarial training with multi-scale features

Given an image I, we denote multi-scale features $F(I) = \{P_i | N_s \le i \le N_e\}$, where F is a multi-scale feature extractor, P_i is a feature map at level i, and N_s and N_e are the first and last scale levels of top-down feature maps from finer to coarser resolution. Here, a feature map is denoted as $P_i = \{P_i^c\}_{c=1}^{C}$, where C is the channel of the feature map. Given a high-resolution image I^{hr} and its low-resolution counterpart I^{lr}, we define the problem of learning G and D with F using the adversarial min-max loss L_{adv}:

$$\begin{aligned}\min_{\theta_G}\max_{\theta_D}\ & \mathbb{E}_{I^{hr}\sim p_{train}(I^{hr})}\left[\sum_{i=N_s}^{N_e}\frac{1}{W_iH_i}\sum_{x=1}^{W_i}\sum_{y=1}^{H_i}\log\left(D_{\theta_D}\left(P_i^{hr}\right)_{x,y}\right)\right] \\ & +\mathbb{E}_{I^{lr}\sim p_G(I^{lr})}\left[\sum_{i=N_s}^{N_e}\frac{1}{W_iH_i}\sum_{x=1}^{W_i}\sum_{y=1}^{H_i}\log\left(1-D_{\theta_D}\left(G_{\theta_G}\left(P_i^{lr}\right)\right)_{x,y}\right)\right]\end{aligned} \tag{1}$$

To solve this, we scale-down I^{hr} to $I^{lr} = \downarrow^{\times s}\left(I^{hr}\right)$, where $\downarrow^{\times s}$ means the down-sampling by a downscaling factor $s (< 1)$. We then extract multi-scale features $\mathbf{P}^{hr} = \left\{P_{N_s}^{hr}, \ldots, P_i^{hr}, \ldots, P_{N_e}^{hr}\right\}$ and $\mathbf{P}^{lr} = \left\{P_{N_s}^{lr}, \ldots, P_i^{lr}, \ldots, P_{N_e}^{lr}\right\}$ by feeding I^{hr} and I^{lr} to F, respectively. Thus, our main idea behind this formulation is that we make G learn the feature distribution of the high-resolution image at each scale by fooling a D that is trained to discriminate up-sampled feature patches from high-resolution feature patches. For multi-scale features, W_i and H_i are the width and height of muti-scale features along the scale level i, respectively. x and y are indexes of the feature pixel coordinates. θ_D and θ_G are parameters of the discriminator and interpolator, respectively. Also, the resolutions of P_i^{hr} and $D\left(P_i^{hr}\right)$ are same but their channel dimensions are C and 1, respectively, according to Section 3.1. For multi-scale feature learning, we exploit BiFPN [6] as F. However, it could be replaced with other multi-scale feature extractor (e.g. PAFPN [4] and FPN [3]).[2] Also, we set s to 0.5 since the resolution of P_{i-1} in BiFPN is higher than

[2] In our implementation, we use BiFPN [6], PAFPN [4], and FPN [3] for F as shown in Tables 1 and 5.

Table 1
Comparison with other detectors on COCO *test – dev*. 'R', 'S', 'X', and 'Swin' denote ResNet [30], ResNeSt [40], ResNeXt [44], and Swin-Transformer [31], respectively. '∗', '‡', and '◇' represent our re-implementation, multi-scale testing results, and training with COCO unlabeled set via self-training [45], respectively.

Interpol.	Backbone	Detector	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}
NN	X-101-64x4d-FPN	MAL‡ [35]	47.0	66.1	51.2	30.2	50.1	58.9	–	–	–	–	–	–
NN	VoVNetV2-99-FPN	CenterMask [14]	46.5	–	–	28.7	48.9	57.2	41.8	–	–	24.4	44.4	54.3
NN	X-101-64x4d-FPN-DCN	ATSS‡ [2]	50.7	68.9	56.3	33.2	52.9	62.4	–	–	–	–	–	–
NN	X-101-64x4d-FPN	HTC [1]	47.1	–	–	–	–	–	41.2	63.9	44.7	22.8	43.9	54.6
NN	R-101-FPN-DCN	TSP-RCNN [36]	47.4	66.7	51.9	29.0	49.7	59.1	–	–	–	–	–	–
NN	X-101-64x4d-FPN-DCN	Dynamic DETR [37]	49.3	68.4	53.6	30.3	51.6	62.5	–	–	–	–	–	–
NN	SENet-154-FPN-DCN	TSD‡ [38]	51.2	71.9	56.0	33.8	54.8	64.2	–	–	–	–	–	–
NN	X-101-64x4d-FPN-DCN	OTA‡ [39]	51.5	68.6	57.1	34.1	53.7	64.1	–	–	–	–	–	–
NN	S-200-FPN-DCN [40]	Cascade R-CNN‡	53.3	72.0	58.0	35.1	56.2	66.8	47.1	–	–	–	–	–
NN	Res2Net-101-FPN-DCN	GFLV2‡ [41]	53.3	70.9	59.2	35.7	56.1	65.6	–	–	–	–	–	–
NN	X-152-32x8d-FPN-DCN	PAA‡ [42]	53.5	71.6	59.1	36.0	56.3	66.9	–	–	–	–	–	–
NN	X-101-64x4d-RFP	DetectoRS‡ [7]	55.7	74.2	61.1	37.7	58.4	68.1	**48.5**	**72.0**	**53.3**	**31.6**	**50.9**	**61.5**
NN	Res2Net-101-BiFPN-DCN	CenterNet2$^{◇‡}$ [43]	56.4	74.0	61.6	38.7	59.7	68.6	–	–	–	–	–	–
NN	R-50-FPN	RetinaNet*	37.6	57.3	40.2	21.7	40.8	46.6	–	–	–	–	–	–
NN	R-50-FPN	FCOS*	39.7	58.9	43.2	23.6	42.7	48.6	–	–	–	–	–	–
NN	R-50-BiFPN	FCOS*	40.6	59.4	43.8	24.1	43.3	49.6	–	–	–	–	–	–
NN	R-50-FPN	Faster R-CNN*	38.3	59.5	41.4	22.3	40.7	47.9	–	–	–	–	–	–
NN	R-50-FPN	Mask R-CNN*	39.0	60.0	42.5	22.6	41.4	48.7	35.5	57.0	37.8	19.5	37.6	46.0
NN	R-50-PAFPN	Mask R-CNN*	39.0	59.8	42.5	22.8	41.3	48.8	35.6	56.9	38.1	19.8	37.6	46.3
NN	R-50-FPN	CenterMask*	39.7	58.1	43.2	23.0	42.3	49.7	35.2	55.7	37.8	19.1	37.6	45.8
NN	R-50-BiFPN	CenterMask*	40.6	58.7	44.2	23.5	43.2	50.1	35.8	56.3	38.5	19.5	38.1	46.1
NN	S-101-FPN	Cascade R-CNN*	48.5	67.1	52.7	30.1	51.3	61.3	41.8	64.6	45.3	24.8	44.4	54.4
NN	S-101-PAFPN	Cascade R-CNN*	48.6	67.2	52.7	29.8	51.6	61.2	41.9	64.7	45.3	24.5	44.5	54.4
NN	Swin-T-BiFPN	Cascade R-CNN*	46.3	64.0	50.0	28.6	48.7	58.6	–	–	–	–	–	–
NN	Swin-T-BiFPN	Cascade R-CNN$^{*◇}$	48.3	65.9	52.3	30.5	51.3	60.6	–	–	–	–	–	–
AFI (ours)	R-50-FPN	RetinaNet*	40.1	59.4	43.2	24.2	43.5	48.3	–	–	–	–	–	–
AFI (ours)	R-50-FPN	FCOS*	42.6	61.4	46.4	26.3	45.5	51.3	–	–	–	–	–	–
AFI (ours)	R-50-BiFPN	FCOS*	43.9	62.4	47.6	27.2	46.7	53.0	–	–	–	–	–	–
AFI (ours)	R-50-FPN	Faster R-CNN*	39.8	60.4	43.4	24.0	43.0	48.0	–	–	–	–	–	–
AFI (ours)	R-50-FPN	Mask R-CNN*	41.5	62.0	45.7	25.6	44.9	49.9	37.4	59.1	40.2	21.8	40.1	46.8
AFI (ours)	R-50-PAFPN	Mask R-CNN*	40.9	61.3	44.6	23.8	43.8	51.4	36.9	58.5	39.5	20.3	39.2	48.0
AFI (ours)	R-50-FPN	CenterMask*	42.4	60.5	46.2	25.8	45.7	51.6	37.5	58.1	40.5	21.3	40.5	47.5
AFI (ours)	R-50-BiFPN	CenterMask*	43.8	61.7	47.6	26.9	46.6	53.4	38.2	59.3	41.3	22.3	40.7	48.4
AFI (ours)	S-101-FPN	Cascade R-CNN*	49.3	67.7	53.4	31.1	52.3	61.5	42.5	65.3	45.9	25.7	45.1	54.4
AFI (ours)	S-101-FPN	Cascade R-CNN*‡	51.6	70.1	56.0	34.1	54.5	64.1	44.8	67.7	48.8	28.3	47.4	57.4
AFI (ours)	S-101-PAFPN	Cascade R-CNN*	49.4	67.8	53.4	30.9	52.2	61.9	42.6	65.3	46.0	25.7	45.0	54.8
AFI (ours)	S-101-PAFPN	Cascade R-CNN*‡	51.6	70.1	56.0	34.0	54.4	64.6	44.7	67.7	48.7	28.2	47.0	57.5
AFI (ours)	Swin-T-BiFPN	Cascade R-CNN*	47.8	66.1	51.5	29.4	50.5	60.3	–	–	–	–	–	–
AFI (ours)	Swin-T-BiFPN	Cascade R-CNN$^{*◇}$	51.7	70.2	55.9	33.0	55.0	65.0	–	–	–	–	–	–
AFI (ours)	Swin-T-BiFPN	Cascade R-CNN$^{*◇‡}$	53.8	72.4	58.3	36.3	56.6	67.1	–	–	–	–	–	–
AFI (ours)	Swin-L-BiFPN	Cascade R-CNN$^{*◇}$	54.9	73.0	59.6	36.0	58.3	69.0	–	–	–	–	–	–
AFI (ours)	Swin-L-BiFPN	Cascade R-CNN$^{*◇‡}$	**57.3**	**76.0**	**62.2**	**39.8**	**60.6**	**71.2**	–	–	–	–	–	–

Table 2
Comparison of detectors using different interpolation methods and multi-scale feature extractors on COCO *val*2017. All detectors are trained based on R-50-FPN for about 12 COCO epochs. All times are reported per image on a same Titan Xp GPU.

Detector	Interpolation	Reuse	Arch.	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}	# Params	Time (ms)
Baseline detectors																	
RetinaNet	NN	-	-	37.4	56.7	40.3	23.1	41.6	48.3	-	-	-	-	-	-	37 M	88
Mask R-CNN	NN	-	-	38.6	59.5	42.1	22.5	42.0	49.9	35.2	56.3	37.5	17.2	37.7	50.3	44 M	72
CenterMask	NN	-	-	39.8	57.9	43.5	23.5	43.2	51.3	35.5	53.9	38.9	17.0	38.3	51.1	51 M	73
RetinaNet (ours)	AFI	*G*	*RB*	37.8	58.0	40.5	22.2	41.9	48.2	-	-	-	-	-	-	47 M	101
Mask R-CNN (ours)	AFI	*G*	*RB*	39.5	60.0	43.5	24.2	44.2	49.3	35.8	57.0	38.6	17.5	38.7	50.1	54 M	118
CenterMask (ours)	AFI	*G*	*RB*	39.9	58.1	43.5	23.8	43.9	50.0	35.6	55.6	38.2	16.4	39.0	50.5	61 M	84
RetinaNet (ours)	AFI	*G*	*RDB*	37.5	56.9	40.1	23.4	41.8	47.7	-	-	-	-	-	-	45 M	101
Mask R-CNN (ours)	AFI	*G*	*RDB*	39.7	60.5	43.4	23.8	43.4	50.2	36.0	57.4	38.5	17.8	38.7	51.0	52 M	116
CenterMask (ours)	AFI	*G*	*RDB*	39.6	57.8	43.0	23.8	43.5	50.2	35.2	55.3	37.8	16.7	38.2	50.4	59 M	85
RetinaNet (ours)	AFI	*M*	*RB*	39.6	59.0	42.4	24.7	44.0	49.6	-	-	-	-	-	-	47 M	101
Mask R-CNN (ours)	AFI	*M*	*RB*	41.2	61.4	45.4	25.2	45.0	51.4	37.0	58.3	40.0	**18.8**	39.5	52.2	54 M	118
CenterMask (ours)	AFI	*M*	*RB*	**42.1**	60.0	**46.0**	25.8	**46.2**	**53.4**	**37.2**	57.4	**40.4**	18.5	**40.4**	**53.5**	61 M	84
RetinaNet (ours)	AFI	*M*	*RDB*	39.6	58.8	42.6	24.4	43.8	49.5	-	-	-	-	-	-	45 M	101
Mask R-CNN (ours)	AFI	*M*	*RDB*	41.4	**61.8**	45.3	25.4	45.0	51.9	**37.2**	**58.7**	40.0	**18.8**	39.8	52.3	52 M	116
CenterMask (ours)	AFI	*M*	*RDB*	42.0	59.8	45.6	**26.1**	45.8	52.9	36.9	57.0	39.9	18.6	40.0	52.1	59 M	85

it of P_i by a factor of 2 ($N_s < i \le N_e$). For BiFPN and PAFPN, we exploit the final multi-scale features as a target feature when training AFI-GAN, respectively. This is because they are the final output features after repeated bottom-up and top-down propagations in multi-scale feature extractors as shown in Fig. 3. In the next Section 4, we use BiFPN as a base feature extractor F, and denote P_i as an i-th level multi-scale feature map for simplicity.

4. Training

The goal of the AFI-GAN training is to generate a multi-scale AF extractor for a target detector. We first train a generalized AFI-GAN which can scale-up any lower-resolution features by a factor of 2. To train it, we perform adversarial training between G and D by exploiting multi-scale features of F as target features. We then build a multi-scale AF extractor by changing all the interpolation modules of the feature extractor with the pre-trained AF interpolator. The multi-scale AF extractor and D can be trained adversarially in the alternative manner. Basically, we can train them by solving Eq. (1). However, we add additional content (pixel-wise L1) and detection losses. As a result, we can improve the quality of up-sampled features per scale and multi-scale representation for object detection. Finally, we can train several FPN-based Mask R-CNN, RetinaNet, and CenterMask detectors with the trained AF interpolater by minimizing its detection loss without adversarial training. For more clarity, we present the AFI-GAN training as shown in Algorithm 1.

4.1. AFI-GAN

For generating upsampled features, we perform adversarial training between an AF interpolator G and a feature patch discriminator D as shown in Fig. 1(a). We first define an adversarial feature up-sampling loss $L_{AF}(G, D, F) = L_{cont}(G, F) + \lambda L_{adv}(G, D, F)$ composed of the content loss and adversarial loss of Eq. (1). L_{cont} evaluates the discrepancy between upsampled ones of low-resolution features and its counterpart high-resolution features at each scale level i using a pixel-wise L1 distance. On the other hand, L_{adv_G} encourages G to produce upsampled features by fooling D. By minimizing L_{AF} with respect to θ_G, we can train G as:

$$\min_{\theta_G} \sum_{i=N_s}^{N_e} \frac{1}{CW_iH_i} \sum_{c=1}^{C} \sum_{x=1}^{W_i} \sum_{y=1}^{H_i} \left| \left(P_i^{hr}\right)^c_{x,y} - \left(G_{\theta_G}\left(P_i^{lr}\right)\right)^c_{x,y} \right| + \lambda \sum_{i=N_s}^{N_e} \frac{1}{W_iH_i} \sum_{x=1}^{W_i} \sum_{y=1}^{H_i} - \log\left(D_{\theta_D}\left(G_{\theta_G}\left(P_i^{lr}\right)\right)_{x,y}\right), \tag{2}$$

where λ is a hyper parameter for controlling the feedback of D and tuned to 0.001. C is the channel of the feature map, and is set to 256.[3]

On the other hand, when training D, we exploit a generic GAN loss described in Eq. (1). D tries to maximize the probabilities of identifying the correct labels for the given target and up-sampled feature patches from G. From this adversarial training, AFI-GAN can learn a generalized up-sampled feature representation for an input feature.

4.2. Multi-scale AF extractor

We design a multi-scale AF extractor by embedding the trained AF interpolator into BiFPN. Simply, we change all the interpolation modules of BiFPN with the AF interpolator.[4] As shown in Fig. 1(b),

[3] The number of output channels of the feature map is different from that of the original BiFPN [6]. Also, we use the same $C = 256$ for PAFPN [4] and FPN [3].

[4] The comparison of different interpolation methods are provided in Table 6 and Fig. 7.

we attach box and mask heads on the extractor, and we denote this AFI-based detection architecture as M for simplicity. For adversarial training, we also use the feature patch discriminator D and the trained parameters of D are re-used. We define an integral loss in consideration of detection accuracy and the quality of upsampled features as $L_{INT}(M,D,F) = L_{AF}(M,D,F) + L_{DET}(M)$. Here, $L_{DET}(M)$ is the overall detection loss which is slightly different according to the detection heads. In our case, we use losses of Mask R-CNN [12], RetinaNet [11], FCOS [13], CenterMask [14], and Cascade R-CNN [15] when attaching their heads to our AF extractor. Similar to Eq. (2), $L_{AF}(M,D,F)$ is the adversarial feature upsampling loss evaluating the discrepancy between upsampled features and target features as well as encouraging G to generate upsampled features by deceiving D. While training M by minimizing $L_{INT}(M,D,F)$, F is not trained, but it just provides target features to D. Because L_{INT} contains the detection loss L_{DET}, optimizing our network with respect to L_{INT} enhances the localization and discrimination powers of our AF interpolator.

Note that for L_{AF} evaluation we first scale-down I^{hr} to resample $I^{lr} = \downarrow^{\times 0.5}\left(I^{hr}\right)$, and then provide I^{lr} and I^{hr} to M and F to extract $\mathbf{P}^{lr} = \left\{P_3^{lr},\ldots,P_7^{lr}\right\}$ and $\mathbf{P}^{hr} = \left\{P_3^{hr},\ldots,P_7^{hr}\right\}$, respectively, as shown in Fig. 1(b). We then extract a set of up-sampled features $\hat{\mathbf{P}}^{hr} = \left\{G_{\theta_G}\left(P_3^{lr}\right),\ldots,G_{\theta_G}\left(P_7^{lr}\right)\right\}$, but scale-down $\mathbf{P}^{hr}$ to $\downarrow^{\times 0.5}\left(\mathbf{P}^{hr}\right)$. This is because of the following reasons: (1) To compare up-sampled and high-resolution features at the same scale (*or* pyramid) level i since semantic information levels are different across feature pyramid levels as also discussed in Lin et al. [3]. For instance, we can feed the same I^{hr} to M and F to compare $\left\{\hat{P}_4^{hr},\hat{P}_5^{hr},\hat{P}_6^{hr},\hat{P}_7^{hr}\right\}$ and $\left\{P_3^{hr},P_4^{hr},P_5^{hr},P_6^{hr}\right\}$, respectively. However, when evaluating L_{AF}, the mismatch of feature semantic levels degrades mAP to about 1.6% shown in Table 7. (2) To reduce GPU usage. Alternatively, we can feed the original I^{hr} and $\uparrow^{\times 2}\left(I^{hr}\right)$ to M and F, and make the level-wise feature comparison between $\left\{G_{\theta_G}\left(P_3^{hr}\right),\ldots,G_{\theta_G}\left(P_7^{hr}\right)\right\}$ and $F\left(\uparrow^{\times 2}\left(I^{hr}\right)\right)$ without the downsampling. However, it is very costly for GPU memory. In return, for evaluating $L_{DET}(M)$ with the input of $\downarrow^{\times 0.5}\left(I^{hr}\right)$, we need to fit the ground truth of box locations and mask regions to $\downarrow^{\times 0.5}\left(I^{hr}\right)$ of the resolution. In order to train parameters θ_M of the multi-scale AF extractor, we minimize the following $L_{AF}(M,D,F)$:

$$\min_{\theta_M} \sum_{i=N_s}^{N_e} \frac{1}{C\lfloor sW_i\rfloor\lfloor sH_i\rfloor} \times \sum_{c=1}^{C}\sum_{x'=1}^{\lfloor sW_i\rfloor}\sum_{y'=1}^{\lfloor sH_i\rfloor}\left|\left(\downarrow^{\times s}\left(P_i^{hr}\right)\right)^c_{x',y'} - \left(\hat{P}_i^{hr}\right)^c_{x',y'}\right| + \lambda\sum_{i=N_s}^{N_e}\frac{1}{\lfloor sW_i\rfloor\lfloor sH_i\rfloor}\sum_{x'=1}^{\lfloor sW_i\rfloor}\sum_{y'=1}^{\lfloor sH_i\rfloor} -\log\left(D_{\theta_D}\left(\hat{P}_i^{hr}\right)_{x',y'}\right), \tag{3}$$

where $\lfloor sW_i\rfloor$ and $\lfloor sH_i\rfloor$ are width and height of downsampled target feature by a factor $s(=0.5)$ at level i. The same λ of Eq. (2) is used. Compared to Eq. (2), an up-sampled feature at previous level is used as an input of the next scale level, where an up-sampled feature is extracted as $\hat{P}_i^{hr} = M_{\theta_M}(\hat{P}_{i+1}^{hr})$, $N_s \le i < N_e$. Therefore, $L_{AF}(M,D,F)$ makes M suitable more for multi-scale representation.

For adversarial training of D, we denote L_{adv_D} in Eq. (1) as the adversarial loss of D. We use $\downarrow^{\times 0.5}\left(\mathbf{P}^{hr}\right)$ and $\hat{\mathbf{P}}^{hr}$ as real and fake input features. In the similar manner, by maximizing L_{adv_D}, we can train D, and leverage its predictions for the generated $\hat{\mathbf{P}}^{hr}$ for training M.

4.3. Target detector

As shown in Fig. 1(c), we apply our trained AF interpolator for training a target detector T which exploits a multi-scale feature extractor as a backbone. More concretely, we change all the interpolation modules of T with the AF interpolator only, but do not reuse other trained parameters of the feature extractor. In order to train T, we minimize the overall detection loss L_{DET} defined by the head type of T as discussed in Section 4.2.

Note that the main difference from the previous training on the multi-scale AF extractor feeds the original image itself to T without downsampling. Therefore, the AF interpolator can be fine-tuned to be suitable more for the detection in high resolution images through this training.

In addition, we fine-tune the parameters of the AF interpolator while training T.[5] In practice, once a pre-trained AF interpolator model is given, we can train T directly without the training of AFI-GAN and AF extractor. This indicates that the training complexity of T using the AF interpolator can be significantly reduced. We prove the effectiveness of reusing the pre-trained models in Table 3. Furthermore, it is also feasible to reuse the whole multi-scale AF feature extractor for T instead of using the AF interpolator only. In this case, the mAP of T can be improved further as shown in Table 2.

As shown in Fig. 4, we combine various recent detection heads into our architecture, and implement several versions of AFI-based detectors with a different detection head T. We present the details of each detection head as follows:

4.3.1. Mask R-CNN

This anchor-based two-stage object detection method [12] has two-stages for region proposal and refinement. At the first stage, the detector generates region proposals by feeding multi-scale features and reference boxes (anchors) into RPN [16]. After applying non-maximum suppression (NMS) on proposals with foreground classification scores and IoU scores, features of proposals are cropped by RoI align operation. At the second stage features of proposals are forwarded through 3 parallel head branches (box regression, classification, and class-agnostic mask prediction heads) in order to refine the box locations, and predict a class confidence and object mask regions for each proposal. Then, per-class NMS is applied to candidate boxes in order to yield final results. For training Mask R-CNN, the loss function of Mask R-CNN L_{DET}^M is defined as:

$$L_{DET}^M = L_{cls}^M + L_{loc}^M + L_{mask}^M + L_{rpn}^M, \tag{4}$$

where L_{cls}^M and L_{loc}^M are log (i.e. cross entropy) and smooth L1 losses, respectively, as described in Girshick [32]. L_{mask}^M is the average binary cross-entropy loss. The loss L_{rpn}^M of the region proposal network is composed of the binary cross entropy and smooth L1 losses for classification and box regression.

4.3.2. RetinaNet

This anchor-based one-stage object detection method [11] presents the focal loss to solve the class imbalance problem. The detector consists of a backbone network and two subnets for classification and box regression. For a backbone network, it extracts multi-scale features. Then, the classification subnet predicts the classification probability at each spatial position for each of the predefined 9 anchors and 80 object classes. The box regression subnet infers the 4×9 linear outputs for predicting spatial location for each object. Then, RetinaNet decodes box predictions from top 1k candidates per multi-scale feature extractor level, after thresholding detector confidence 0.05. After merging top predictions from all levels, per-class NMS is applied to get

[5] When freezing the learned parameters of the AF interpolator during T training, the mAP of T is degraded as shown in Table 4

Table 3
Comparison results of target detectors trained by different AF extractors on COCO *val*2017. All detectors are trained based on R-50-FPN for about 12 COCO epochs. All times are reported per image on a same Titan Xp GPU.

Target detector	Interpol	Head for multi-scale AF extractor	AP^{box}	AP^{mask}	# Params	Time (ms)
Mask R-CNN (Baseline)	NN	–	38.6	35.2	44 M	72
Mask R-CNN	AFI	Mask R-CNN	**41.2**	**37.0**	52 M	118
Mask R-CNN	AFI	RetinaNet	40.0	36.2	52 M	118
Mask R-CNN	AFI	Centermask	40.5	36.5	52 M	117
RetinaNet (Baseline)	NN	–	37.4	–	37 M	88
RetinaNet	AFI	Mask R-CNN	39.6	–	45 M	101
RetinaNet	AFI	RetinaNet	38.2	–	45 M	102
RetinaNet	AFI	Centermask	**39.7**	–	45 M	107
Centermask (Baseline)	NN	–	39.8	35.1	51 M	73
Centermask	AFI	Mask R-CNN	**42.1**	**37.2**	59 M	84
Centermask	AFI	RetinaNet	40.9	35.9	59 M	86
Centermask	AFI	Centermask	41.7	36.8	59 M	85

Table 4
Effects of progressive learning. 'FR' means freezing the pre-trained parameters of the AF interpolator during target detector training.

Name	Interpol.	AFI-GAN	Multi-scale AF extractor	Target detector	FR	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
A1	NN					38.6	59.4	42.1	35.2	56.3	37.5
A2	AFI (Ours)	✓		✓		39.4	60.3	43.2	35.8	57.0	38.4
A3	AFI (Ours)	✓	✓			32.1	52.1	33.7	30.0	50.2	31.5
A4	AFI (Ours)	✓	✓	✓	✓	39.9	60.2	43.7	36.2	57.1	39.1
A5	AFI (Ours)	✓	✓	✓		**41.2**	**61.4**	**45.4**	**37.0**	**58.3**	**40.0**
A6	AFI (Ours)			✓		38.8	59.6	42.3	35.4	56.6	37.7

Table 5
Comparisons with different multi-scale feature methods and the proposed method on COCO *val*2017. All detectors are trained for about 12 COCO epochs.

Interpolation	Backbone	Multi-scale feature method	Detector	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}
NN	R-50	FPN	RetinaNet	37.4	56.7	40.3	23.1	41.6	48.3
NN	R-50	PAFPN	RetinaNet	37.7	56.7	40.4	23.0	42.2	48.2
NN	R-50	BiFPN	RetinaNet	38.2	57.4	40.8	22.4	42.4	48.8
AFI (ours)	R-50	FPN	RetinaNet	39.6	58.8	42.6	24.4	43.8	49.5
AFI (ours)	R-50	PAFPN	RetinaNet	39.7	58.8	43.0	25.7	44.0	48.8
AFI (ours)	R-50	BiFPN	RetinaNet	**41.9**	**60.8**	**44.9**	**26.8**	**45.7**	**52.6**

Table 6
Comparison of interpolation methods for accuracy and network capacity on the COCO *val*2017 set. All detectors are trained with Mask R-CNN R-50-FPN for about 12 COCO epochs.

Interpolation method for FPN	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}	# Params
Nearest Neighbor (NN)	38.6	59.5	42.1	22.5	42.0	49.9	35.2	56.3	37.5	17.2	37.7	50.3	44 M
Bilinear (BL)	38.6	59.4	42.2	22.5	41.9	50.0	35.2	56.4	37.5	16.9	37.7	50.7	44 M
Bicubic (BC)	38.5	59.4	42.3	23.1	41.8	49.2	35.1	56.4	37.6	17.3	37.5	50.2	44 M
NN-5conv	37.4	57.0	40.6	21.3	40.4	48.5	34.0	54.0	36.6	16.0	36.0	49.0	53 M
BL-5conv	37.4	57.0	40.7	21.4	40.5	48.1	34.0	54.1	36.4	15.9	36.0	48.8	53 M
BC-5conv	37.0	56.6	40.3	21.4	39.8	48.1	33.8	53.8	36.3	15.7	36.0	49.0	53 M
Deconv	40.2	61.0	44.0	23.9	43.5	**52.1**	36.7	58.0	39.4	17.9	38.9	**52.9**	53 M
AF interpolator (Ours)	**41.2**	**61.4**	**45.4**	**25.2**	**45.0**	51.4	**37.0**	**58.3**	**40.0**	**18.8**	**39.5**	52.2	54 M

Table 7
Effect of semantic level matching for L_{AF}.

$L_{AF}(M, D, F)$	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}
Different feature levels	39.6	60.1	43.2	23.6	42.9	51.1	35.9	57.1	38.3	17.6	38.5	51.6
Same feature levels (Ours)	**41.2**	**61.4**	**45.4**	**25.2**	**45.0**	**51.4**	**37.0**	**58.3**	**40.0**	**18.8**	**39.5**	**52.2**

final results. For training this RetinaNet, the loss function L^R_{DET} is defined as:

$$L^R_{DET} = L^R_{cls} + L^R_{loc}, \tag{5}$$

where L^R_{cls} and L^R_{loc} are the focal and smooth L1 losses, respectively.

4.3.3. *FCOS*

This anchor-free one-stage object detection method [13] introduces the centerness branch, which refines center areas of a box. The detector consists of a backbone and 3 head branches (classification, regression, and centerness). The detector is based on a multi-scale feature extractor. In order to determine a box location,

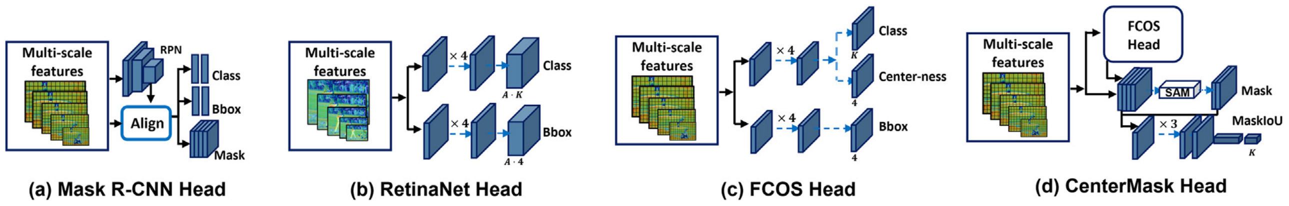

Fig. 4. Various recent detection network heads. For multi-scale AF extractor and target detector training as in Sections 4.2 and 4.3, we minimize the overall detection loss L_{DET} which is different according to the mounted detection head.

the regression branch predicts 4-dimensional normalized offsets (left, right, top, and bottom) to adjust box lengths at the center point. The classification branch infers a 80-dimensional vector per box when classifying 80 different object categories. The centerness branch measures how close the predicted center point is to the corresponding GT point. For predicting this score, a single layer branch is added. By multiplying the classification score of a box with its centerness, the box score can be re-weighted. As a result, during NMS, the low-quality boxes can be suppressed better. The loss of FCOS L_{DET}^{F} is

$$L_{DET}^{F} = L_{cls}^{F} + L_{reg}^{F} + L_{ctr}^{F}, \tag{6}$$

where L_{cls}^{F} is the focal loss as in Lin et al. [11] and L_{reg}^{F} is the IoU loss as in Yu et al. [33]. L_{ctr}^{F} is the binary cross entropy loss for predicting a centerness score ranging from 0 to 1.

4.3.4. CenterMask

This [14] is an improved version of FCOS [13] by attaching a spatial attention-guided mask branch (SAG-Mask) for instance segmentation. Therefore, it has the similar architecture as FCOS described in Section 4.3.3. However, CenterMask adds a mask head named as a spatial attention-guided mask head in order to predict segmentation mask inside the cropped regions in a per-pixel manner. Also, a mask scoring head [34] is attached for recalibrating the classification score in consideration of predicted mask quality. For training this, the following detection loss L_{DET}^{C} is used:

$$L_{DET}^{C} = L_{cls}^{C} + L_{reg}^{C} + L_{ctr}^{C} + L_{mask}^{C} + L_{maskiou}^{C}, \tag{7}$$

where $L_{cls}^{C}, L_{reg}^{C}, L_{ctr}^{C}$ are the same as in FCOS as described in Section 4.3.3. L_{mask}^{C} is the same average binary cross-entropy loss as in Mask R-CNN as described in Section 4.3.1. $L_{maskiou}^{C}$ is the L2 loss for regressing MaskIoU as in Huang et al. [34].

5. Experiments

In this section, we prove the effects of our method via ablation studies and comparisons with state-of-the-arts (SOTA) methods. All experiments are conducted on the MS COCO dataset [17] containing 118k images for training (*train*2017), and 5k images for validation (*val*2017). For testing, 20k images without labels are included and results can be evaluated only on the challenge server. For training AFI-GAN, AF extractor, and target detector, we use the *train*2017 set. When training AFI-GAN and AF extractor, we downsample the training images by a factor of 2 for generating low-resolution images, and use original ones as high-resolution images. For ablation study and comparisons, we use *val*2017 and $test-dev$ sets for evaluating detectors. We use the standard COCO-style metrics. We evaluate box AP^{box} and mask AP^{mask} (average precision over IoU = 50:5:95). For boxes and masks, we also compute AP$_{50}$ (IoU = 50%), and AP$_{75}$ (IoU = 75%), AP$_S$, AP$_M$, and AP$_L$ (for different sizes of objects).

5.1. Implementation details

We use Detectron2 for implementing all detectors and networks. For learning AFI-GAN, we implement and train BiFPN, PAFPN, and FPN with different backbones provided in Detectron2, and use them as a F. We then adversarially train the AF interpolator and the feature patch discriminator from scratch with target F by optimizing Eqs. (1) and (2). We use stochastic gradient descent (SGD) with 0.9 momentum and 0.0001 weight decay. We train them by using 8 Titan Xp GPUs for 150k iterations. We set a learning rate of 0.001, and decay it by a factor of 0.1 at 120k iterations.

For learning the multi-scale AF extractor M, we design M by substituting all interpolation modules of multi-scale feature extractors (e.g. BiFPN, PAFPN, and FPN) with the AF interpolator. For instance segmentation, we attach the Mask R-CNN head on the AF extractor. For the AF interpolator and the feature patch discriminator, we reuse the learned parameters by the previous AFI-GAN training. However, other the parameters of the multi-scale AF extractor are initialized. We then train the multi-scale AF extractor and feature patch discriminator by minimizing Eq. (3) and maximizing Eq. (1). Here, we also use the same SGD optimizer, and train them for 270k iterations with a mini-batch including 16 target images. We set a learning rate to 0.02 and decay it by a factor of 0.1 at 210k and 250k iterations.

When training a target detector, we change interpolation modules of the multi-scale feature extractors with the trained AF interpolator. However, for training and testing target detectors, we maintain the default setting parameters of the detectors. As target detectors, we select RetinaNet [11], FCOS [13], Faster R-CNN [16], Mask R-CNN [12], CenterMask [14], and Cascade R-CNN [15] since they can be good baselines as one-stage and two-stage detectors. We implement all the detectors by incorporating the AF interpolator. We train these detectors using 1x schedules (∼12 COCO epochs). For applying the Swin transformer [31] for our detector, we replace the SGD optimizer with the AdamW optimizer and set a learning rate to 0.0001. We train them using 3x schedules (∼37 COCO epochs) for achieving better accuracy scores.

5.2. Comparison with state-of-the-arts methods

In this evaluation, we compare our proposed method with other methods on $test-dev$ and *val*2017 sets. As mentioned, we first train the AF interpolator or the multi-scale AF extractor and then apply them for several one- and two-stage detectors. Because we can reuse the AF interpolator or the whole multi-scale AF extractor, we mark G and M as shown in Table. 2. For all the detectors shown in Table 1 and 2, we train our AF interpolator with the same backbone.

Comparison on COCO test-dev Table 1 shows the comparison results on COCO $test-dev$. For this comparison, we implement a lot of detectors using different interpolation methods. Furthermore, we implement and evaluate our detectors with different backbones (e.g. R-50-FPN, R-50-PAFPN, R-50-BiFPN, S-101-FPN, S-101-PAFPN, Swin-T-BiFPN, and Swin-L-BiFPN). Compared to our 12 detectors

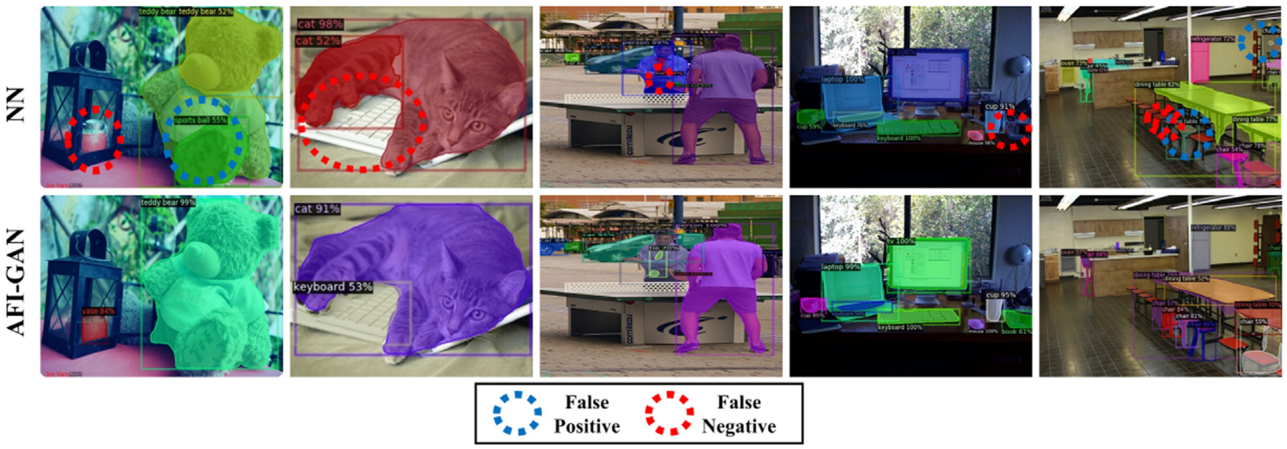

Fig. 5. Qualitative comparison of NN interpolation and the proposed AFI-GAN on detection results. Both are trained based on Mask R-CNN R-50-FPN for 12 COCO epochs. All images are from COCO *test* – *dev* set. Refer to our supplementary video for more detection results.

using the AF interpolator (AFI) with their counterparts using NN (shown in the 2nd row of Table 1), our detectors show the much better box and mask scores. More specifically, the box and mask mAP scores are improved by about 2.2% and 1.6% on average, respectively. Furthermore, we compare our method with the recent detectors. Our method reports the remarkable accuracy at 57.3% mAP on the COCO test-dev set by using a large transformer backbone [31] and a self-training method [45].

Additionally, some qualitative comparison results are shown in Fig. 5, and detection and segmentation results are shown in Fig. 8 on COCO *test* – *dev*. From these experimental results, we verify that our method is indeed beneficial of improving detection and segmentation results regardless of the types of backbones and detectors.

Effects of AF interpolator We replace interpolation modules of FPN with the AF interpolator only without using the AF extractor. As shown in Table 2, it provides 0.1% ∼ 1.1% box AP and 0.1% ∼ 0.8% mask AP gains although the improved APs are different according to the detectors. These results show that using the AF interpolator shows the better results than using the NN interpolation since it can generate the higher quality of feature maps for object detection as also shown in Fig. 7.

Effects of multi-scale AF extractor In this evaluation, we use the trained multi-scale AF extractor as a backbone of a detector. As shown in Table 2, it provides better box and mask AP gains than using the AF interpolator only. This is because the backbone is also trained better to be suitable for the AF interpolator. In particular, for Mask R-CNN with ResNet-50-FPN, we achieve 2.8% and 2.0% improvements for AP^{box} and AP^{mask} compared to their counterparts using the NN interpolation. As shown in Table 1, our AF extractors provide more AP gains for the detectors with light backbones. However, it can still improve AP scores for heavy detectors.

Comparison of G architecture To investigate the effects of *G* architectures, we implement different *G* with residual dense blocks (*RDB*) and residual blocks (*RB*). The details of *RDB* are described in Section 3.1. In *RB*, we use 5 residual blocks instead of the 3 residual dense blocks. However, for a fair comparison we maintain the number of parameters of both generators to be almost same.

For the details of the *RB* architecture in Fig. 6, we feed a feature map of any resolution to a 3 × 3 convolution and a Leaky ReLU activation layers ($\alpha = 0.2$). After them, we add 5 consecutive residual blocks consisting of two 3 × 3 convolution, two batch normalization, and one Leaky ReLU layers to learn the more informative representation for up-sampling. Then, one convolution and one deconvolution blocks are followed to scale-up the feature resolution by

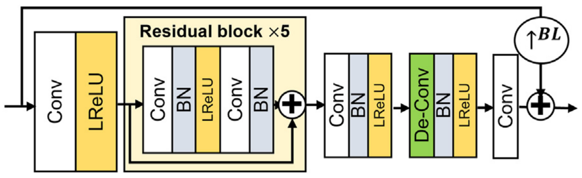

Fig. 6. AF interpolator *G* architecture with 5 residual blocks (*RB*).

a factor of 2. In order to make the channel dimensionality equal to the input, we attach a 3 × 3 convolution layer. For residual learning, a shortcut connection is added between the deconvolved feature and upsampled input feature by the bilinear interpolation.

In Table 2, we compare detectors with the different *G*. We found that the differences of box and mask AP scores are marginal in most cases. This means that our AF interpolator learning method is not sensitive to the architecture of an interpolator.

Speed and parameters In Table 3, we compare the inference time between detectors using NN and our method. Our method needs about additional 8M parameters and delays inference time by about 24ms in average. This is because convolving features iteratively in the AF interpolator. The speed can be improved by using the lighter AF interpolator.

5.3. Ablation study

Flexibility of AF interpolator To find the effects of using the AF interpolator trained by other detector's head, we first implement three AF interpolators with different heads of Mask R-CNN [12], RetinaNet [11], and CenterMask [14]. We use ResNet-50-FPN as the backbones of all extractors. We train the extractor with Mask R-CNN head using $\mathbf{P}^{hr} = \{P_2^{hr}, \ldots, P_6^{hr}\}$ and $\mathbf{P}^{lr} = \{P_2^{lr}, \ldots, P_6^{lr}\}$ from the extractors. Once the AF extractors are trained, we train each detector with different AF extractors for 12 epochs, and evaluate them on the COCO *val*2017 set.

Table 3 shows the comparison results. For all the detectors, AP scores are improved compared to the baseline using the NN interpolation. Interestingly, the most detectors show the better APs when using the AF extractors trained with the Mask R-CNN head. The ability of the AF interpolator might be improved more as generating up-sampled features for the finer feature map P_2 during training. This also means that we can improve the AF extractor by training it with finer feature maps than P_2. From these results, we could apply a pre-trained AF extractor for any detector in practice since our AF extractor has high flexibility.

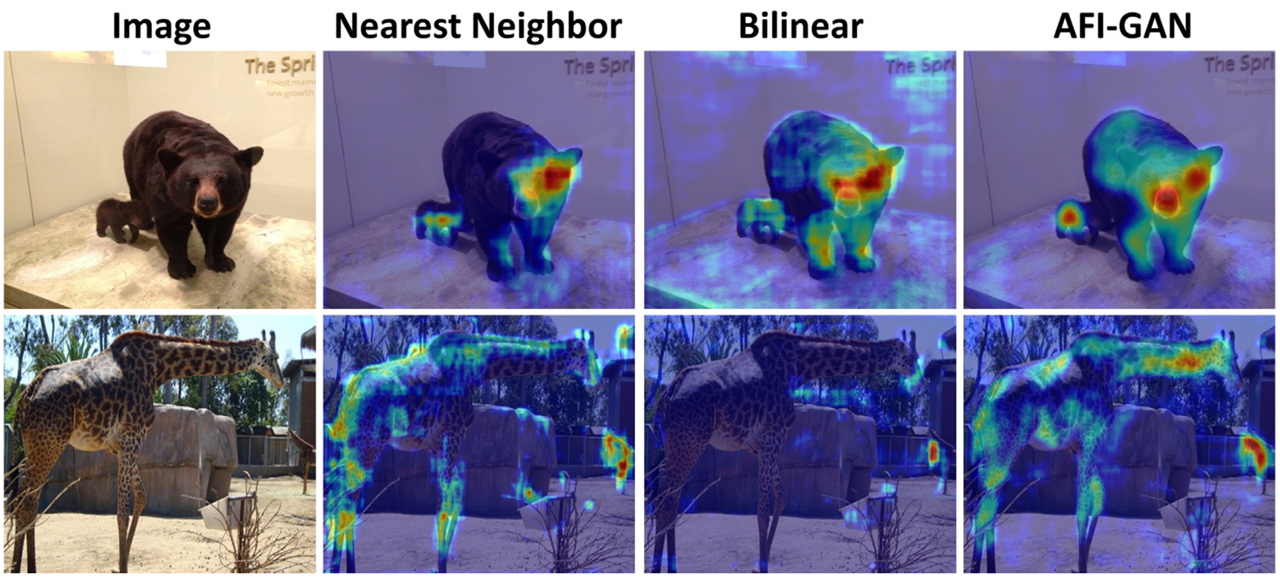

Fig. 7. Activation maps of P_3 with different interpolations.

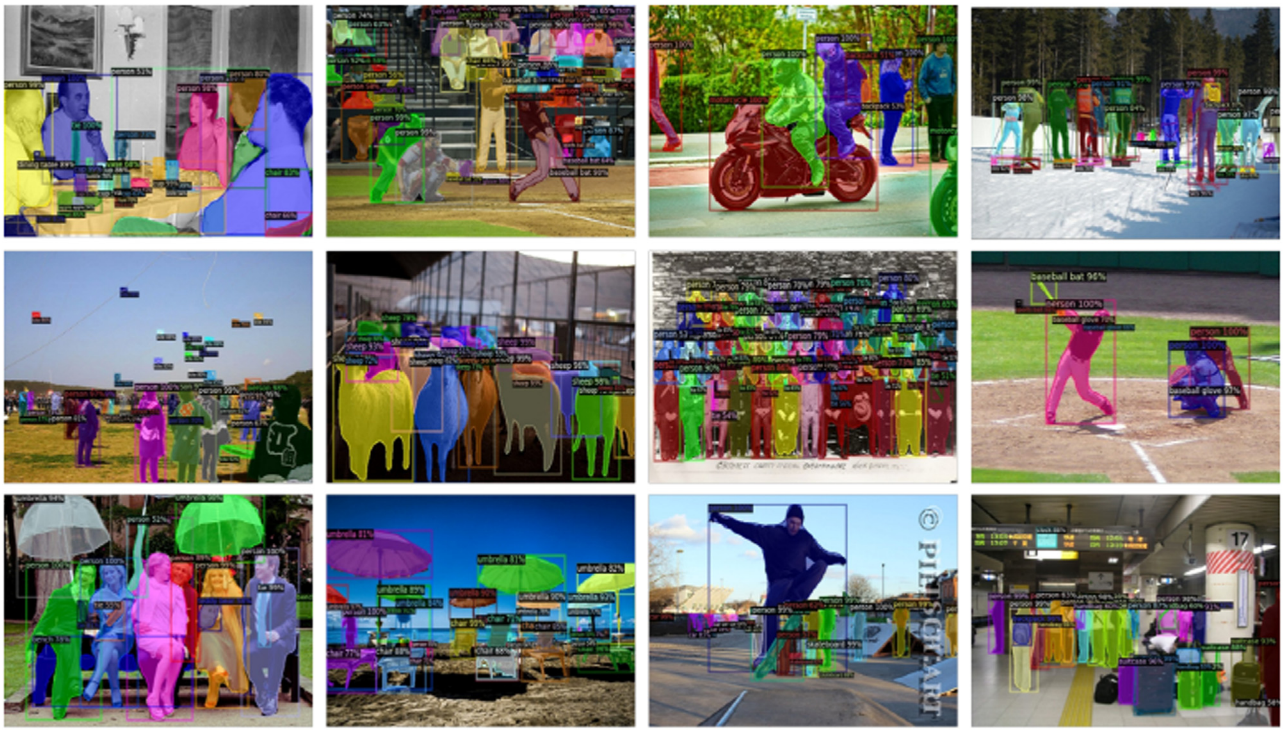

Fig. 8. Detection and segmentation results of our proposed AFI-GAN. All images are from COCO *test* – *dev* set. Refer to our supplementary video for more detection results.

Effects of learning methods To show the effects of our learning methods, based on ResNet-50-FPN, we train several Mask R-CNN detectors (A1–A6): (A1) is the baseline using the NN interpolation; (A2–A6) use the AF interpolator by substituting the NN interpolation of the baseline; (A2) is trained without the multi-scale AF extractor learning; (A3) is the adversarially trained detector during the training of multi-scale AF extractor; (A4) freezes the learned parameters of the AF interpolator during training of the detector; (A5) is trained by using all our learning methods; (A6) does not use our adversarial learning methods. Therefore, the entire network of (A6) is trained from end-to-end joint training without adversarial training.

Table 4 shows the AP scores of (A1)–(A6). For (A3), the performance is degraded severely because it is not trained with images of the original resolutions. Except for (A3), other detectors using our methods show the better results than (A1). When comparing (A4) and (A5), additional fine-tuning of the AF extractor is more effective at the stage of target detector training. Compared to (A1), (A5) achieves box and mask AP gains by 2.6% and 1.8%. These results indicate that our learning methods are beneficial of generating up-sampled features for detection and segmentation. When comparing (A1) and (A6), box and mask AP scores of (A6) slightly increase by 0.2% and 0.2%, respectively. However, AP gains are too marginal compared to that of (A5).[6] It implies that our proposed

[6] Note that we do not use a pre-trained AF interpolator in (A6). If AF interpolator is pre-trained via our adversarial learning, we can achieve more AP gains as shown in (A5).

Table 8
Comparisons of different degradation functions. All detectors are trained based on Mask R-CNN with R-50-FPN for about 12 COCO epochs.

Degradation function	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}
Nearest Neighbor	41.0	61.3	**45.4**	25.0	**45.5**	**51.7**	**37.1**	58.6	39.8	18.5	**39.7**	**52.5**
Bicubic	**41.2**	**61.6**	**45.4**	**25.9**	45.0	51.5	**37.1**	**58.8**	39.9	**19.0**	39.6	52.0
Bilinear (Ours)	**41.2**	61.4	**45.4**	25.2	45.0	51.4	37.0	58.3	**40.0**	18.8	39.5	52.2

Table 9
Effects of higher times upsampling for the proposed AF interpolator. All detectors are trained with Mask R-CNN R-50 for about 12 COCO epochs.

Name	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}
B1	36.9	57.7	39.6	19.8	41.3	49.5	32.2	53.5	33.7	12.5	35.6	49.7
B2 (Ours)	**41.2**	**61.4**	**45.4**	**25.2**	**45.0**	**51.4**	**37.0**	**58.3**	**40.0**	**18.8**	**39.5**	**52.2**

progressive learning method allows AF interpolator to generate the high quality of up-sampled features and to improve detection accuracy together.

Multi-scale feature networks As shown in Table 5, we provide comparison results among different multi-scale feature networks and interpolation methods. For a fair comparison, we fix the backbone to ResNet-50. Furthermore, we exploit RetinaNet as a detector because BiFPN works better usually for the one-stage object detector. As a result, the box AP scores are improved when applying the improved multi-scale feature representation methods and interpolation methods. In particular, when we exploit the proposed interpolation method, we show consistent improvements compared to NN. Therefore, it indicates that improving the interpolation is also important to improve the quality of the feature maps and the detection results. More importantly, our AF interpolation methods have indeed high flexibility over different multi-scale feature networks and can work well with the recent multi-scale feature representation methods.

Interpolation method As shown in Table 6, we train several Mask R-CNN detectors based on the ResNet-50-FPN by applying different interpolation methods. We exploit the nearest neighbor, bilinear, and bicubic interpolations and our AF interpolator. Furthermore, we consider a network capacity (i.e. the number of parameters) together for fair comparison. To this end, we implement NN-5conv, BL-5conv, and BC-5conv interpolation methods by adding 5 convolution layers[7] to make the number of parameters similar to that of the AF interpolator. We also implement a deconvolution-based upsampling method (Deconv) containing a 6×6 deconv layer and a 3×3 conv layer. It has almost the same number of parameters as our AF interpolator.

The accuracy difference between NN, BL, and BC interpolation methods is so marginal. In addition, NN/BL/BC-5Conv degrade box and mask APs by about 1.3% and 1.2% compared to their counterparts without 5conv. We expect that forwarding the upsampled features to the several Conv layers leads to the aggregation effect within the vicinity of each pixel. It means that semantic information could be learned but the localized details could be reduced in return. On the other hand, our AF interpolator and the Deconv method provide the better results with the similar number of parameters compared to NN/BL/BC-5conv. It is because deconvolution contains learnable parameters to upsample the features. We conjecture that learnable parameters of the deconvolution affect the quality enhancement of upsampled features. Furthermore, in our implementation, deconvolution is followed by one conv layer in order to maintain the same number of parameters as NN/BL/BC-5conv and the AF interpolator. Therefore, the loss of the feature locality information would be less than that of NN/BL/BC-5conv. Also, our AF interpolator shows 1.0% and 0.3% better scores for box and mask APs, respectively, compared to the deconvolution-based method. It reflects that adding more layers after the simple interpolation method for generating up-sampled features better is not effective, but degrades the detection accuracies. However, our AF interpolator and learning method can resolve this problem. We also provide qualitative comparisons of these methods in Fig. 7. Our AF interpolator can allocate higher weights within the object than other interpolations. From these results, we confirm that our interpolation method is more appropriate for object detection.

Importance of semantic level matching As discussed in Section 4.2, a multi-scale AF extractor can also be trained by comparing features between different semantic levels. More concretely, we feed training images of the same resolution to M and F, and compare $\left\{\hat{P}^{hr}_3, \hat{P}^{hr}_4, \hat{P}^{hr}_5, \hat{P}^{hr}_6\right\}$ and $\left\{P^{hr}_2, P^{hr}_3, P^{hr}_4, P^{hr}_5\right\}$ when evaluating the loss Eq. (3). Table 7 shows the results. The mismatch between the feature semantic levels degrades box and mask APs by 1.6% and 1.1%. Thus, it is crucial to compare features at the same semantic level when training the multi-scale AF extractor.

Degradation function For generating low-resolution images, we use bilinear interpolation as a degradation function as shown in Fig. 1. We also evaluate box and mask APs when applying nearest neighbor and bicubic interpolation methods. As shown in Table 8, all the methods produce almost similar scores. It means that our learning methods are not sensitive to the image degradation functions.

Higher times upsampling In Table 9, we perform higher times upsampling by using the AF interpolator. To this end, (B1) iteratively uses the AF interpolator to scale-up the feature P_5 by 2x, 4x, and 8x. Therefore, we generate $\hat{P}_4$, $\hat{P}_3$, and $\hat{P}_2$ features by using the P_5 feature only. On the other hand, (B2) uses the AF interpolator to upsample multi-scale features by 2 times as mentioned in Section 4.3. Thus, compared to (B2), (B1) does not exploit lateral connections [3] since it requires P_5 only. As a result, (B1) achieves much lower accuracies than (B2). We conjecture that the localization power of generated features is reduced because of the absence of the lateral connections which can add a more accurately localized feature from a fewer subsampled feature map.

6. Conclusion

In this paper, we have proposed a novel adversarial feature interpolator for multi-scale feature representation. We have presented an AFI-GAN architecture and learning methods to train it effectively. From the extensive ablation study and comparison with

[7] Concretely, we add more 5 convolution and 4 batch normalization layers (between feature pyramid levels) after each interpolation method.

state-of-the-art detectors, our method indeed is beneficial to enhance detection and segmentation accuracies. Another important benefit of our method is its high flexibility. Indeed, we have shown our AF interpolator can be applicable for the many recent detectors, and multi-scale feature extractors without much effort. Albeit the end-to-end joint learning between our AF interpolator and a target detector is possible, the detection accuracy improvement is rather marginal. Once the pre-trained AF interpolator from our adversarial learning is provided, this problem can be addressed as proved in our experiment. Because we open our code and pre-trained AF models to the public, we believe that our method can be a solid feature interpolator for convolutional detectors. For future works, we focus on generating high quality of multi-scale feature maps for object recognition tasks. To this end, a recent Denoising Diffusion Probabilistic Model (DDPM) can be adopted since it generates better upsampled features compared to GAN methods via iterative refinement steps for log-likelihood-based objectives. We will extend our methods by applying other vision tasks (e.g. pose estimation and scene understanding) which use multi-scale feature representation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have provided our code link at GitHub to the public, and this link has been attached to the manuscript.

Acknowledgments

This work was supported by the INHA UNIVERSITY Research Grant and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1C1C1009208) and funded by the Ministry of Education (No. 2022R1A6A1A03051705), and funded by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00448: Deep Total Recall, 10%, No. RS-2022-00155915: Artificial Intelligence Convergence Innovation Human Resources Development (Inha University)).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2023.109365.

References

[1] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, et al., Hybrid task cascade for instance segmentation, in: CVPR, 2019, pp. 4974–4983.
[2] S. Zhang, C. Chi, Y. Yao, Z. Lei, S.Z. Li, Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, in: CVPR, 2020, pp. 9759–9768.
[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: CVPR, 2017, pp. 2117–2125.
[4] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: CVPR, 2018, pp. 8759–8768.
[5] G. Ghiasi, T.-Y. Lin, Q.V. Le, NAS-FPN: learning scalable feature pyramid architecture for object detection, in: CVPR, 2019, pp. 7036–7045.
[6] M. Tan, R. Pang, Q.V. Le, EfficientDet: scalable and efficient object detection, in: CVPR, 2020, pp. 10781–10790.
[7] S. Qiao, L.-C. Chen, A. Yuille, Detectors: detecting objects with recursive feature pyramid and switchable atrous convolution, in: CVPR, 2021, pp. 10213–10224.
[8] J. Yuan, H.-C. Xiong, Y. Xiao, W. Guan, M. Wang, R. Hong, Z.-Y. Li, Gated CNN: integrating multi-scale feature layers for object detection, Pattern Recognit. 105 (2020) 107131.
[9] W. Ma, Y. Wu, F. Cen, G. Wang, MDFN: multi-scale deep feature learning network for object detection, Pattern Recognit. 100 (2020) 107149.
[10] Z. Wang, J. Chen, S.C.H. Hoi, Deep learning for image super-resolution: a survey, IEEE TPAMI 43 (10) (2020) 3365–3387.
[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: ICCV, 2017, pp. 2980–2988.
[12] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: ICCV, 2017, pp. 2961–2969.
[13] Z. Tian, C. Shen, H. Chen, T. He, FCOS: fully convolutional one-stage object detection, in: ICCV, 2019, pp. 9627–9636.
[14] Y. Lee, J. Park, Centermask: real-time anchor-free instance segmentation, in: CVPR, 2020, pp. 13906–13915.
[15] Z. Cai, N. Vasconcelos, Cascade R-CNN: delving into high quality object detection, in: CVPR, 2018, pp. 6154–6162.
[16] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: NIPS, 2015, pp. 91–99.
[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: common objects in context, in: ECCV, Springer, 2014, pp. 740–755.
[18] J. Wang, X. Tao, M. Xu, Y. Duan, J. Lu, Hierarchical objectness network for region proposal generation and object detection, Pattern Recognit. 83 (2018) 260–272.
[19] J. Peng, H. Wang, S. Yue, Z. Zhang, Context-aware co-supervision for accurate object detection, Pattern Recognit. 121 (2022) 108199.
[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in: ECCV, Springer, 2016, pp. 21–37.
[21] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, S. Yan, Perceptual generative adversarial networks for small object detection, in: CVPR, 2017, pp. 1222–1230.
[22] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, ICLR, 2017.
[23] V. Chalavadi, P. Jeripothula, R. Datla, C. Sobhan Babu, C. Krishna Mohan, mSODANet: a network for multi-scale object detection in aerial images using hierarchical dilated convolutions, Pattern Recognit. 126 (2022) 108548.
[24] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: CVPR, IEEE, 2010, pp. 2528–2535.
[25] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: CVPR, 2016, pp. 1874–1883.
[26] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu, Residual dense network for image super-resolution, in: CVPR, 2018, pp. 2472–2481.
[27] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, AAAI, vol. 31, 2017.
[28] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: CVPR, 2017, pp. 1125–1134.
[29] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: ICML, vol. 30, 2013, p. 3.
[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: ICCV, 2021, pp. 10012–10022.
[32] R. Girshick, Fast R-CNN, in: ICCV, 2015, pp. 1440–1448.
[33] J. Yu, Y. Jiang, Z. Wang, Z. Cao, T. Huang, Unitbox: an advanced object detection network, in: Proceedings of the 24th ACM International Conference on Multimedia, 2016, pp. 516–520.
[34] Z. Huang, L. Huang, Y. Gong, C. Huang, X. Wang, Mask scoring R-CNN, in: CVPR, 2019, pp. 6409–6418.
[35] W. Ke, T. Zhang, Z. Huang, Q. Ye, J. Liu, D. Huang, Multiple anchor learning for visual object detection, in: CVPR, 2020, pp. 10206–10215.
[36] Z. Sun, S. Cao, Y. Yang, K.M. Kitani, Rethinking transformer-based set prediction for object detection, in: ICCV, 2021, pp. 3611–3620.
[37] X. Dai, Y. Chen, J. Yang, P. Zhang, L. Yuan, L. Zhang, Dynamic DETR: end-to-end object detection with dynamic attention, in: ICCV, 2021, pp. 2988–2997.
[38] G. Song, Y. Liu, X. Wang, Revisiting the sibling head in object detector, in: CVPR, 2020, pp. 11563–11572.
[39] Z. Ge, S. Liu, Z. Li, O. Yoshie, J. Sun, OTA: optimal transport assignment for object detection, in: CVPR, 2021, pp. 303–312.
[40] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, et al., Resnest: split-attention networks, arXiv preprint arXiv:2004.08955(2020).
[41] X. Li, W. Wang, X. Hu, J. Li, J. Tang, J. Yang, Generalized focal loss v2: learning reliable localization quality estimation for dense object detection, in: CVPR, 2021, pp. 11632–11641.
[42] K. Kim, H.S. Lee, Probabilistic anchor assignment with iou prediction for object detection, in: ECCV, Springer, 2020, pp. 355–371.
[43] X. Zhou, V. Koltun, P. Krähenbühl, Probabilistic two-stage detection, arXiv preprint arXiv:2103.07461(2021).
[44] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: CVPR, 2017, pp. 1492–1500.
[45] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E.D. Cubuk, Q. Le, Rethinking pre-training and self-training, NIPS 33 (2020) 3833–3845.

Seong-Ho Lee received the B.S. degree in Computer Science and Engineering from Incheon National University in 2019, and received the M.S. degree with the Department of Electronic Computer Engineering at Inha University, Korea. He was a full-time researcher at Inha University in 2022. He is curruntly working at SK Hynix Inc. His research interests include object detection, multi-object tracking, multi-scale representation and generative adversarial networks.

Seung-Hwan Bae received the B.S. degree in information and communication engineering from Chungbuk National University, in 2009 and the M.S. and Ph.D. degrees in information and communications from the Gwangju Institute of Science and Technology (GIST), in 2010 and 2015, respectively. He was a senior researcher at Electronics and Telecommunications Research Institute (ETRI) in Korea from 2015 to 2017. He was an assistant professor in the Department of Computer Science and Engineering at Incheon National University, Korea from 2017 to 2020. He is currently an Associate Professor with the Department of Computer Engineering at Inha University. His research interests include multi-object tracking, object detection, deep learning, feature learning, medical image analysis, generative adversarial networks, face forensic, etc.