SRF-GAN: Super-Resolved Feature GAN for Multi-Scale Representation

Seong-Ho Lee and Seung-Hwan Bae*
Computer Vision & Learning Laboratory
Inha University, Korea

22201366@inha.edu and shbae@inha.ac.kr

Abstract

Recent convolutional object detectors exploit multi-scale feature representations added with top-down pathway in order to detect objects at different scales and learn stronger semantic feature responses. In general, during the top-down feature propagation, the coarser feature maps are up-sampled to be combined with the features forwarded from bottom-up pathway, and the combined stronger semantic features are inputs of detector's headers. However, simple interpolation methods (e.g. nearest neighbor and bilinear) are still used for increasing feature resolutions although they cause noisy and blurred features.

In this paper, we propose a novel generator for super-resolving features of the convolutional object detectors. To achieve this, we first design super-resolved feature GAN (SRF-GAN) consisting of a detection-based generator and a feature patch discriminator. In addition, we present SRF-GAN losses for generating the high quality of super-resolved features and improving detection accuracy together. Our SRF generator can substitute for the traditional interpolation methods, and easily fine-tuned combined with other conventional detectors. To prove this, we have implemented our SRF-GAN by using the several recent one-stage and two-stage detectors, and improved detection accuracy over those detectors. Code is available at https://github.com/SHLee-cv/SRF-GAN.

1. Introduction

Due to the advances in deep convolutional neural networks (CNNs), the convolutional object detectors [1, 6, 14, 21, 29] 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 the multi-scale feature representation. For multi-scale object detection, some architectures [25, 28, 41] are designed and used for base networks (*i.e.* backbone) of detectors.

Among them, a feature pyramid network (FPN) [25] 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 network (PANet) [28] designs the extra bottom-up pathway following the top-down pathway.

In spired by these works, many multi-scale feature methods [9, 12, 24, 33, 34, 41, 46] for object detection have been also presented. In specific, [9, 34, 41, 46] design additional feature propagation pathway for better feature representation. Also, detection methods [12, 24, 33] 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 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 [43], 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 generator which can produce super-resolved features used for multi-scale feature learning. In order to learn this generator, we propose a super-resolved feature generative adversarial network (SRF-GAN) consisting of a SRF generator and a feature patch discriminator. Furthermore, we present a new integral loss which can make our SRF-GAN appropriate more for multi-task learning to multi-scale object detection and segmentation.

For learning SRF-GAN, we perform adversarial training between the SRF generator and the feature patch discriminator with a super-resolved feature GAN loss. As a result, the SRF-GAN can learn a generic super-resolved feature representation from an input feature. Subsequently, we incorporate the SRF-GAN with a multi-scale feature extractor

---
*Corresponding author: Seung-Hwan Bae (e-mail: shbae@inha.ac.kr)

by replacing the interpolation module with the SRF generator. Then, for learning the multi-scale SRF extractor, the adversarial training between the multi-scale SRF extractor and the feature patch discriminator is followed using the integral loss including object detection and super-resolved feature GAN losses.

Note that another difficulty of SRF-GAN training comes from that the ground truth for super-resolved features is not available. To address this, our core idea is that we exploit the multi-scale feature network (*e.g.* FPN) as a target feature generator during these SRF-GAN training. To this end, we feed original and downsampled images to the feature network, and then extract features at each level. Then, multi-scale features from a downsampled image are forwarded to the SRF generator, and the super-resolved ones are then compared with the corresponding features extracted from the original image by the feature patch discriminator as shown in Fig. 1.

Once the SRF-GAN is trained, we can use it for interpolation directly. However, the training of a target detector with the SRF generator shows the better detection because parameters can be tuned together for the specific task. In practical, we emphasize that the pre-training of SRF-GAN shown in Fig. 1 can be omitted because the reuse of the SRF-GAN trained with other backbones is available. This indicates that our SRF-GAN can learn generalized super-resolved features and have high flexibility over different backbones. To prove our SRF-GAN, we have implemented several different versions of target detectors by using RetinaNet [26], CenterMask [21], and Mask R-CNN [14] representing one-stage or two-stage detectors. We have shown the significant improved detection accuracy compared with the recent detectors on COCO dataset. We have also made the extensive ablation study with different backbones and detectors.

The summarization of the main contributions of this paper is (i) proposition of a novel SRF-GAN to generate super-resolved features for multi-scale feature learning and be applicable easily for other convolutional detectors; (ii) proposition of the SRF-GAN losses for generating the high quality of super-resolved features and improving detection accuracy together; (iii) proposition of a multi-scale feature learning scheme for stable SRF-GAN training without the ground truth of super-resolved features.

2. Related Work

We discuss previous works on deep object detection, multi-scale representation, and super-resolution for detection, 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 de-

tectors using anchors [10, 11, 15, 37] show the significant improvement on several detection benchmarks [7, 38, 27]. In addition, the anchor-based detection can be divided into two-stage and one-stage methods. The two-stage detection methods first generate region of interest (RoI) with the region proposal network, and then refine RoI with the followed R-CNN. Mask R-CNN [14] attaches a mask head to the two-stage detector [37] for accurate pixel-wise segmentation. Multi-stage detection methods [5, 6] can refine RoIs iteratively in a cascade manner. On the other hand, the one-stage detectors predict detections directly without the region proposal. SSD [29] produces predictions of different scales from feature maps of different scales using the predefined anchors. RetinaNet [26] addresses the class imbalance problem by introducing a focal loss. YOLACT [4] and RetinaMask [8] integrate an instance segmentation branch to [36] and [26], respectively.

The anchor-free object detection can reduce the computational complexity and hyper-parameters for anchor generation. CornerNet [19] exploits paired keypoints. Grid R-CNN [30] changes the Faster R-CNN head [37] to regress boxes using the grid points. FCOS [42] introduces the centerness branch to refine center areas of a box. CenterMask [21] adds a spatial attention-guided mask branch to FCOS.

Multi-scale representation: In order to achieve the robustness to object scale variation and the better detection, the intuitive approach is to use features at different scales. [40] presents a scale normalization for image pyramid scheme for reducing object scale variation during training. Autofocus [31] can determine regions to be focused more at each scale during multi-scale inference. These works also provide a good intuition to train a scale-invariant detector.

Instead of using the multi-scale image representation, feature pyramids (*or* multi-scale features) can be also employed for scale-invariant detection. Previous works [18, 3, 13, 29] combine multi-scale features extracted from a bottom-up pathway. On the other hand, recent works [25, 28, 9, 34, 41, 46] learn multi-scale features from several different pathways. FPN [25] first shows that using bottom-up and top-down features is effective for scale-invariant detection. PANet [28] further introduces an extra bottom-up pathway. Inspired by these works, many network architectures using cross-scale [9, 34, 41, 46] and multi-scale feature fusion [12] have been presented. NAS-FPN [9] discovers a suitable architecture for feature pyramid by using the Neural Architecture Search algorithm [49]. AugFPN [12] further improves FPN [25]. EfficientDet [41] applies top-down and bottom-up feature fusion repeatedly with bidirectional features. 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.

Super-resolution for detection: Many super-resolution (SR) methods using a generative adversarial network (GAN) have been presented and the extensive survey can be found in [20, 39]. There are some efforts [35, 2, 23, 32] to apply SR for improving object detection. EESRGAN [35] exploits super-resolved images directly to detect objects at low scale. SOD-MTGAN [2] designs a multi-task loss with a SR loss for object proposals. Super-resolved RoI features [23, 32] are learned for improving small object detection. Compared to these works, our work can upsample a feature map itself.

3. Super-Resolved Feature GAN (SRF-GAN)

For generating super-resolved features at any scale which can be applicable for multi-scale feature learning, we first design SRF-GAN consisting of a SRF generator and a feature patch discriminator as discussed in Sec. 3.1. However, direct supervision is challenging since the ground truth of super-resolved features is unavailable in general. Therefore, our idea is to use the existing multi-scale feature network as a target feature generator, and match super-resolved features from a SRF generator with the corresponding features of the same resolution from the target generator as mentioned in Sec. 3.2. To train SRF-GAN from a scratch, we present progressive learning to avoid it overfitted as shown in Sec. 4. However, note again that we can train a target detector embedded with the SRF generator at once as in Sec. 4.3 if a pre-trained SRF generator by using any multi-scale feature network is provided. Thus, some pre-training phases to warm-up the SRF-GAN can be omitted in practice.

3.1. Overall Architecture

As shown in Fig. 1 (1)-(2), the SRF generator G generates a super-resolved feature map P^{sr} 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 super-resolved feature P^{sr} and target feature P^{tr} (For more details of P^{tr}, refer to Sec. 3.2).

For the generator G, we feed P^{lr} 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 super-resolution. Then, one convolution and one deconvolution blocks are followed to scale-up the feature resolution by a factor of 2. In order to make the channel dimensionality equal to the input, we attach a 1×1 convolution layer. For residual learning, a shortcut connection is added between the deconvolved fea-

ture and upsampled input feature by the bilinear interpolation.

The discriminator D, which is a modified version of a patch discriminator [22, 17], convolves P^{sr} or P^{tr} 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 1×1 convolution and sigmoid activation function.

3.2. Multi-Scale Feature Learning Formulation

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. Given a target image I^{tr} and its low-resolution counterpart I^{lr}, we define the problem of learning G and D with F as the adversarial min-max problem:

$$
\min_{\theta_G} \max_{\theta_D} \; \mathbb{E}_{I^{tr} \sim p_{train}(I^{tr})} \left[\sum_{i=n_s}^{n_e} \frac{1}{W_i H_i} \sum_{x=1}^{W_i} \sum_{y=1}^{H_i} \log \left(D_{\theta_D} \left(P_i^{tr} \right)_{x,y} \right) \right]
$$
$$
+ \; \mathbb{E}_{I^{lr} \sim p_G(I^{lr})} \left[\sum_{i=n_s}^{n_e} \frac{1}{W_i H_i} \sum_{x=1}^{W_i} \sum_{y=1}^{H_i} \log \left(1 - D_{\theta_D} \left(G_{\theta_G} \left(P_i^{lr} \right) \right)_{x,y} \right) \right],
$$
$$(1)$$

To solve this, we scale-down I^{tr} to $I^{lr} = \downarrow^{\times s}(I^{tr})$, where $\downarrow^{\times s}$ means the down-sampling by a downscaling factor $s(< 1)$. We then extract multi-scale features $\mathbf{P}^{tr} = \left\{ P_{n_s}^{tr}, ..., P_i^{tr}, ..., P_{n_e}^{tr} \right\}$ and $\mathbf{P}^{lr} = \left\{ P_{n_s}^{lr}, ..., P_i^{lr}, ..., P_{n_e}^{lr} \right\}$ by feeding I^{tr} and I^{lr} to F, respectively. Thus, our main idea behind this formulation is that we make G learn the feature distribution of the target image at each scale by fooling a D that is trained to discriminate super-resolved feature patches from target 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 generator, respectively. Also, the resolutions of P_i^{tr} and $D(P_i^{tr})$ are same. In our implementation, we use FPN [25] as F, but it could be replaced with other multi-scale feature extractors (*e.g.* PANet [28] and BiFPN [41]). Also, we set s to 0.5 since the resolution of P_{i-1} in FPN is higher than it of P_i by a factor of 2 ($n_s < i \le n_e$).

4. Training

The goal of the SRF-GAN training is to generate a multi-scale SRF extractor for a target detector. We first train a generalized SRF-GAN which can scale-up any lower-resolution features by a factor of 2. To train it, we perform
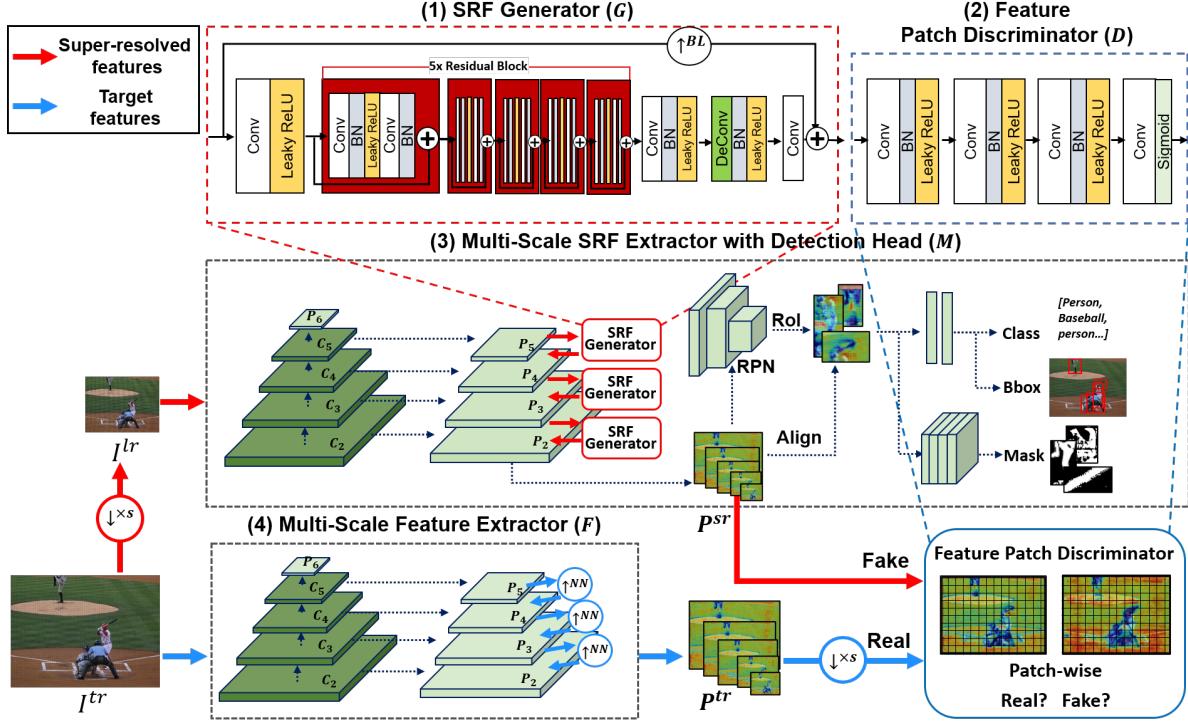
Figure 1. Proposed SRF-GAN architecture which performs adversarial training between the multi-scale SRF extractor with a detection head and the feature patch discriminator. In Fig. 1 (1)-(2), the SRF generator and discriminator are depicted. Figure 1 (3) shows the multi-scale SRF extractor with box and mask heads of Mask R-CNN [14]. Figure 1 (4) shows the multi-scale feature extractor to extract target features. $\uparrow^{NN}$ and $\uparrow^{BL}$ mean the up-sampling using the conventional nearest-neighbor interpolation (NN) and bilinear interpolation (BL), respectively. $\downarrow^{\times s}$ is the the down-sampling using the bilinear interpolation with a downscaling factor s.

adversarial training between G and D by exploiting multi-scale features of FPN as target features. We then build a multi-scale SRF extractor by changing all the interpolation modules of FPN with the pre-trained SRF generator. The multi-scale SRF extractor and D can be trained adversarially in the alternative manner. Basically, we can train them by solving Eq. (1). However, we add additional pixel-wise L1 and detection losses. As a result, we can improve the quality of super-resolved features per scale and multi-scale representation for object detection. Finally, we can train several FPN-based Mask R-CNN, RetinaNet, and Center-Mask detectors with the trained SRF generator by minimizing its detection loss without adversarial training.

4.1. SRF-GAN

For super-resolved feature generation, we perform adversarial training between a SRF generator G and feature patch discriminator D. We first define a super-resolved feature loss $L_{SRF}(G, D, F) = L_{L1}(G, F) + \lambda L_{adv}(G, D, F)$ composed of the pixel-wise L1 loss and adversarial loss of Eq. (1). L_{L1} evaluates the discrepancy between super-resolved ones of low-resolution features and its counterpart target features of high-resolution at each scale level i. On the other hand, L_{adv_G} encourages

G to produce super-resolved features by fooling D. By minimizing L_{SRF} with respect to θ_G, we can train G as:

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

λ is a hyper parameter for controlling the feedback of D and tuned to 0.001. C is the channel of the feature map i, and is set to 256 same as the FPN [25].

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 super-resolved feature patches from G. From this adversarial training, SRF-GAN can learn a generalized super-resolved feature representation for an input feature.

4.2. Multi-Scale SRF Extractor

We design multi-scale SRF extractor by embedding the trained SRF generator into FPN. Simply, we change all the interpolation modules of FPN with the SRF genera-

tor[1]. As shown in Fig. 1 (3), we attach box and mask heads on the extractor, and we denote this SRF-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 super-resolved features as $L_{INT}(M, D, F) = L_{SRF}(M, D, F) + L_{DET}(M)$. 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 [14], RetinaNet[26], CenterMask [21], and Cascade R-CNN [5] when attaching their heads to our SRF extractor. Similar to Eq. (2), $L_{SRF}(M, D, F)$ is the super-resolution feature loss evaluating the discrepancy between super-resolved and target features as well as encouraging G to generate super-resolved 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.

Note that for L_{SRF} evaluation we first scale-down I^{tr} to resample $I^{lr} = \downarrow^{\times 0.5}(I^{tr})$, and then provide I^{lr} and I^{tr} to M and F to extract $\mathbf{P}^{lr} = \{P_2^{lr}, ..., P_5^{lr}\}$ and $\mathbf{P}^{tr} = \{P_2^{tr}, ..., P_5^{tr}\}$, respectively, as shown in Fig. 1. We then extract a set of super-resolved features $\mathbf{P}^{sr} = \{G_{\theta_G}(P_2^{lr}), ..., G_{\theta_G}(P_5^{lr})\}$, but scale-down $\mathbf{P}^{tr}$ to $\downarrow^{\times 0.5}(\mathbf{P}^{tr})$. This is because of the following reasons: (1) To compare super-resolved and target features at the same scale (*or* pyramid) level i since semantic information levels are different across feature pyramid levels as also discussed in [25]. For instance, we can feed the same I^{tr} to M and F to compare $\{P_3^{sr}, P_4^{sr}, P_5^{sr}\}$ and $\{P_2^{tr}, P_3^{tr}, P_4^{tr}\}$, respectively. However, when evaluating L_{SRF}, 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^{tr} and $\uparrow^{\times 2}(I^{tr})$ to M and F, and make the level-wise feature comparison between $\{G_{\theta_G}(P_2^{tr}), ..., G_{\theta_G}(P_5^{tr})\}$ and $F(\uparrow^{\times 2}(I^{tr}))$ without the downsampling. However, it is very costly for GPU memory. In return, for evaluating $L_{DET}(M)$ with the input of $\downarrow^{\times 0.5}(I^{tr})$, we need to fit the ground truth of box locations and mask regions to $\downarrow^{\times 0.5}(I^{tr})$ of the resolution. In order to train parameters θ_M of the multi-scale SRF extractor, we

minimize the following $L_{SRF}(M, D, F)$:

$$
\begin{aligned}
\min_{\theta_M} &\sum_{i=n_s}^{n_e} \frac{1}{C\lfloor sW_i \rfloor \lfloor sH_i \rfloor} \\
&\times \sum_{c=1}^{C} \sum_{x'=1}^{\lfloor sW_i \rfloor} \sum_{y'=1}^{\lfloor sH_i \rfloor} \left| \left(\downarrow^{\times s}(P_i^{tr}) \right)_{x',y'}^c - G_{\theta_G}(P_i^{lr})_{x',y'}^c \right| \\
&+ \lambda \sum_{i=n_s}^{n_e} \frac{1}{\lfloor sW_i \rfloor \lfloor sH_i \rfloor} \sum_{x'=1}^{\lfloor sW_i \rfloor} \sum_{y'=1}^{\lfloor sH_i \rfloor} -\log \left(D_{\theta_D}(P_i^{sr})_{x',y'} \right),
\end{aligned}
$$
(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), a super-resolved feature at previous level is used as an input of the next scale level. Therefore, $L_{SRF}(M, D, F)$ makes M suitable more for multi-scale representation.

For adversarial training of D, we use $\downarrow^{\times 0.5}(\mathbf{P}^{tr})$ and $\mathbf{P}^{sr}$ as real and fake input features. In the similar manner, by maximizing Eq. (1), we can train D, and leverage its predictions for the generated $\mathbf{P}^{sr}$ for training M.

4.3. Target Detector

We apply our trained SRF generator 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 SRF generator 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 Sec 4.2. The main difference from the previous training on the multi-scale SRF extractor feeds the original image itself to T without downsampling. Therefore, the SRF generator can be fine-tuned to be suitable more for the detection in high resolution image through this training.

In addition, we fine-tune the parameters of the SRF generator while training T[2]. In practice, provided any trained SRF generator we can train T directly without the training of SRF-GAN and SRF extractor. This indicates that the training complexity of T using the SRF generator can be significantly reduced. We prove the effectiveness of reusing the pre-trained models in Table. 1 and 3. Furthermore, it is also feasible to reuse the whole multi-scale SRF feature extractor for T instead of using the SRF generator only. In this case, the mAP of T can be improved further as shown in Table. 1.

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

---

[1]The comparison of different interpolation methods are provided in Table 5 and Fig. 2

[2]When freezing the learned parameters of the SRF generator during T training, the mAP of T is degraded as shown in Table. 4

Detector	Backbone	Interpolation	Reuse	Epoch	AP^{box}	AP^{box}_S	AP^{box}_M	AP^{box}_L	AP^{mask}	AP^{mask}_S	AP^{mask}_M	AP^{mask}_L	# Params	Time (ms)
Baseline Detectors														
RetinaNet	R-50-FPN	NN	-	12	37.4	23.1	41.6	48.3	-	-	-	-	37M	88
Faster R-CNN	R-50-FPN	NN	-	12	37.9	22.4	41.1	49.1	-	-	-	-	41M	64
Mask R-CNN	R-50-FPN	NN	-	12	38.6	22.5	42.0	49.9	35.2	17.2	37.7	50.3	44M	72
Mask R-CNN	R-50-FPN	NN	-	37	41.0	24.9	43.9	53.3	37.2	18.6	39.5	53.3	44M	72
RetinaNet (ours)	R-50-FPN	SRF	G	12	37.8[+0.4]	22.2	41.9	48.2	-	-	-	-	47M	101
Faster R-CNN (ours)	R-50-FPN	SRF	G	12	38.9[+1.0]	23.2	43.0	49.7	-	-	-	-	51M	109
Mask R-CNN (ours)	R-50-FPN	SRF	G	12	39.5[+0.9]	24.2	44.2	49.3	35.8[+0.6]	17.5	38.7	50.1	54M	118
RetinaNet (ours)	R-50-FPN	SRF	M	12	39.6[+2.2]	24.7	44.0	49.6	-	-	-	-	47M	101
Faster R-CNN (ours)	R-50-FPN	SRF	M	12	39.3[+1.4]	23.8	43.0	49.8	-	-	-	-	51M	109
Mask R-CNN (ours)	R-50-FPN	SRF	M	12	41.2[+2.6]	25.2	45.0	51.4	37.0 [+1.8]	18.8	39.5	52.2	54M	118
Mask R-CNN (ours)	R-50-FPN	SRF	M	37	41.6 [+0.6]	25.3	45.3	52.5	37.4 [+0.2]	19.1	39.6	52.8	54M	118

Table 1. Comparison with the state-of-the-art methods on COCO val2017. NN and SRF are nearest neighbor interpolation and our method, respectively. G and M mean that reusing a pre-trained SRF generator only and whole multi-scale SRF extractor when training the target detector, respectively. The scores in **[]** are the performance gain compared with scores of the baseline detectors. All times are reported per image on same Titan Xp GPU.

Detector	Backbone	Interpolation	Epoch	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
RetinaNet [26]	R-101-FPN	NN	-	39.1	59.1	42.3	21.8	42.7	50.2	-	-	-	-	-	-
Faster R-CNN [25]	R-101-FPN	NN	-	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Libra R-CNN [33]	R-50-FPN	NN	12	38.7	59.9	42.0	22.5	41.1	48.7	-	-	-	-	-	-
Libra R-CNN [33]	R-101-FPN	NN	12	40.3	61.3	43.9	22.9	43.1	51.0	-	-	-	-	-	-
Mask R-CNN [14]	R-101-FPN	NN	-	38.2	60.3	41.7	20.1	41.1	50.2	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN [12]	R-50-AugFPN [12]	NN	12	37.5	59.4	40.6	22.1	40.6	46.2	34.4	56.3	36.6	18.6	37.2	44.5
Mask R-CNN [12]	R-101-AugFPN [12]	NN	12	39.8	61.6	43.3	22.9	43.2	49.7	36.3	58.5	38.7	19.2	39.3	47.4
PANet [28]	R-50-FPN	NN	-	41.2	60.4	44.4	22.7	44.0	47.0	36.6	58.0	39.3	16.3	38.1	53.1
FCOS [42]	R-101-FPN	NN	-	41.5	60.7	45.0	24.4	44.8	51.6	-	-	-	-	-	-
FCOS [42]	X-101-64x4d-FPN	NN	-	43.2	62.8	46.6	26.5	46.2	53.3	-	-	-	-	-	-
CenterMask [21]	R-101-FPN	NN	37	44.0	-	-	25.8	46.8	54.9	39.8	-	-	21.7	42.5	52.0
CenterMask [21]	V-99-FPN	NN	37	46.5	-	-	28.7	48.9	57.2	41.8	-	-	24.4	44.4	54.3
TridentNet [24]	R-101	-	37	42.7	63.6	46.5	23.9	46.6	56.6	-	-	-	-	-	-
ATSS [48]	R-101-FPN	NN	25	43.6	62.1	47.4	26.1	47.0	53.6	-	-	-	-	-	-
RetinaNet*	R-50-FPN	NN	12	37.6	57.3	40.2	21.7	40.8	46.6	-	-	-	-	-	-
Faster R-CNN*	R-50-FPN	NN	12	38.3	59.5	41.4	22.3	40.7	47.9	-	-	-	-	-	-
Mask R-CNN*	R-50-FPN	NN	12	39.0	60.0	42.5	22.6	41.4	48.7	35.5	57.0	37.8	19.5	37.6	46.0
Mask R-CNN*	R-50-FPN	NN	37	41.3	62.2	44.9	24.2	43.6	51.7	37.5	59.3	40.2	21.1	39.6	48.3
CenterMask*	R-50-FPN	NN	12	39.7	58.1	43.2	23.0	42.3	49.7	35.2	55.7	37.8	19.1	37.6	45.8
Cascade R-CNN*	S-101-FPN	NN	12	48.5	67.1	52.7	30.1	51.3	61.3	41.8	64.6	45.3	24.8	44.4	54.4
RetinaNet* (ours)	R-50-FPN	SRF	12	40.1[+2.5]	59.4	43.2	24.2	43.5	48.3	-	-	-	-	-	-
Faster R-CNN* (ours)	R-50-FPN	SRF	12	39.8[+1.5]	60.4	43.4	24.0	43.0	48.0	-	-	-	-	-	-
Mask R-CNN* (ours)	R-50-FPN	SRF	12	41.5[+2.5]	62.0	45.7	25.6	44.9	49.9	37.4[+1.9]	59.1	40.2	21.8	40.1	46.8
Mask R-CNN* (ours)	R-50-FPN	SRF	37	42.1[+0.8]	62.4	46.4	25.9	45.5	51.1	37.9[+0.4]	59.6	40.9	22.3	40.6	47.5
CenterMask* (ours)	R-50-FPN	SRF	12	42.4[+2.7]	60.5	46.2	25.8	45.7	51.6	37.5[+2.3]	58.1	40.5	21.3	40.5	47.5
Cascade R-CNN* (ours)	S-101-FPN	SRF	12	48.7[+0.2]	67.2	53.0	30.0	51.8	60.8	42.0[+0.2]	64.8	45.4	24.9	44.8	53.8
Cascade R-CNN*† (ours)	S-101-FPN	SRF	12	50.9	69.7	55.3	33.4	54.0	63.6	44.2	67.1	48.2	27.7	47.0	56.8

Table 2. Comparison with other detectors on COCO $test-dev$. When implementing Cascade R-CNNs, we attach a mask branch to the cascade R-CNN head for instance segmentation. R, S, V, and X denote ResNet [16], ResNeSt [47], VoVNetV2 [21], and ResNeXt [45], respectively. '∗' and '†' represent our re-implementation and multi-scale testing results.

MS COCO dataset [27] containing 118k images for training ($train$2017), and 5k images for validation (val2017). For testing, 20k images without labels are included and results can be evaluated only on the challenge server. For training SRF-GAN, SRF extractor, and target detector, we use the $train$2017 set. When training SRF-GAN and SRF extractor, we downsample the training images by a factor of 2 for generating low-resolution images, and use original ones as target images. For ablation study and comparisons, we use val2017 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 [44] for implementing all detectors and networks. For learning SRF-GAN, we exploit FPNs with different backbones provided in Detectron2 as multi-scale feature extractors. We then adversarially train the SRF generator and the feature patch discriminator from scratch by minimizing Eq. (2) and maximizing Eq. (1). We use stochastic gradient descent (SGD) with 0.9 momentum and 0.0001 weight decay. We train them 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 multi-scale SRF extractors, we design multi-scale SRF extractor by substituting all interpolation modules of FPN with the SRF generator. For instance segmentation, we attach the Mask R-CNN head on the SRF extractor. For the SRF generator and feature patch dis-

Target detector	Head for multi-scale SRF extractor	AP^{box}	AP^{mask}	Time (ms)
Mask R-CNN (Baseline)	-	38.6	35.2	72
Mask R-CNN	Mask R-CNN	41.2	37.0	118
Mask R-CNN	RetinaNet	40.0	36.2	118
Mask R-CNN	Centermask	40.5	36.5	117
RetinaNet (Baseline)	-	37.4	-	88
RetinaNet	Mask R-CNN	39.6	-	101
RetinaNet	RetinaNet	38.2	-	102
RetinaNet	Centermask	39.7	-	107
Centermask (Baseline)	-	39.8	35.5	73
Centermask	Mask R-CNN	42.1	37.2	84
Centermask	RetinaNet	40.9	35.9	86
Centermask	Centermask	41.7	36.8	85

Table 3. Comparison results of target detectors trained by different SRF extractors on COCO val2017.

Name	SRF-GAN	Multi-Scale SRF Extractor	Target Detector	AP^{box}	AP^{mask}
A1				38.6	35.2
A2	✓		✓	39.4	35.8
A3	✓	✓		32.1	30.0
A4	✓	✓	✓(FR)	39.9	36.2
A5 (Ours)	✓	✓	✓	41.2	37.0

Table 4. Effects of progressive learning. 'FR' means freezing the learned parameters of the SRF generator during other training.

criminator, we reuse the learned parameters by the previous SRF-GAN training. However, other the parameters of the multi-scale SRF extractor are initialized. We then train the multi-scale SRF 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 FPN with the trained SRF generator, or replace the FPN itself with the trained multi-scale SRF extractor. However, for training and testing target detectors, we maintain the default setting parameters of the detectors. As target detectors, we select RetinaNet[26], CenterMask [21], Mask R-CNN [14] and Cascade R-CNN [5] since they can be good baselines as one-stage and two-stage detectors. We implement all the detectors by incorporating the SRF generator and multi-scale SRF extractor. We train these detectors using 1x or 3x schedules ($\sim$ 12 or $\sim$ 37 COCO epochs).

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

In this evaluation, we compare our proposed method with other methods on val2017 and $test-dev$ sets. As mentioned, we first train the SRF generator or the multi-scale SRF extractor, apply them for several one- and two-stage detectors. Because we can reuse the SRF generator or the whole multi-scale SRF extractor, we mark G and M as shown in Table. 1. For all the detectors shown in Table 1 and 2, we train its SRF generator and multi-scale SRF extractor with the same backbone.

Effects of SRF generator: We replace interpolation modules of FPN with the SRF generator only without using the SRF extractor. As shown in Table 1, it provides $0.4 \sim 1.0$ box AP and 0.6 mask AP gains although the improved APs are different according to the detectors. These results show

that using SRF generators shows the betters results than using the NN interpolation since it can generate the higher quality of feature maps as also shown in Fig. 2.

Effects of multi-scale SRF extractor: In this evaluation, we use the trained multi-scale SRF extractor as a backbone of a detector. As shown in Table 1, it provides better box and AP gains than using the SRF generator only. This is because the backbone is also trained better to be suitable for the SRF generator. In particular, for Mask R-CNN with ResNet-50-FPN we achieves 2.6% and 1.8% improvements for AP^{box} and AP^{mask} compared of using NN interpolation. As shown in Table 2, our SRF extractors provide more AP gains for the detectors with light backbones. However, it can still improve AP scores for heavy detectors.

Speed and parameters: In Table 1, we compare the inference time between detectors using NN and our method. Our method needs about additional 10M parameters and delay inference time by about 34ms in average. This is because convolving features iteratively in the SRF generator. The speed can be improved by using the lighter SRF generator.

Comparison on COCO test-dev: Table 2 shows the comparison results on on COCO $test-dev$. For this comparison, we implement a lot of detectors using different interpolation methods. Compared to the detectors with NN interpolation, our detectors achieve the better box and mask scores. 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.

5.3. Ablation study

Flexibility of SRF extractor: To find the effects of using the multi-scale SRF extractor trained by other detector's head, we first implement three SRF extractors with different heads of Mask R-CNN [14], RetinaNet [26], and CenterMask [21]. We use ResNet-50-FPN as the backbones of all the extractors. When training the extractors with RetinaNet and CenterMask heads, we use $\mathbf{P}^{tr} = \{P_3^{tr}, ..., P_5^{tr}\}$ and $\mathbf{P}^{lr} = \{P_3^{lr}, ..., P_5^{lr}\}$ from the extractors because they do not feed P_2 to the heads. Once the SRF extractors are trained, we train each detector with different SRF extractors for 12 epochs, and evaluate them on the COCO val2017 set.

Table 3 shows the comparison results. For all the detectors, AP scores are improved compared to the baseline using NN interpolation. Interestingly, the most detectors show the
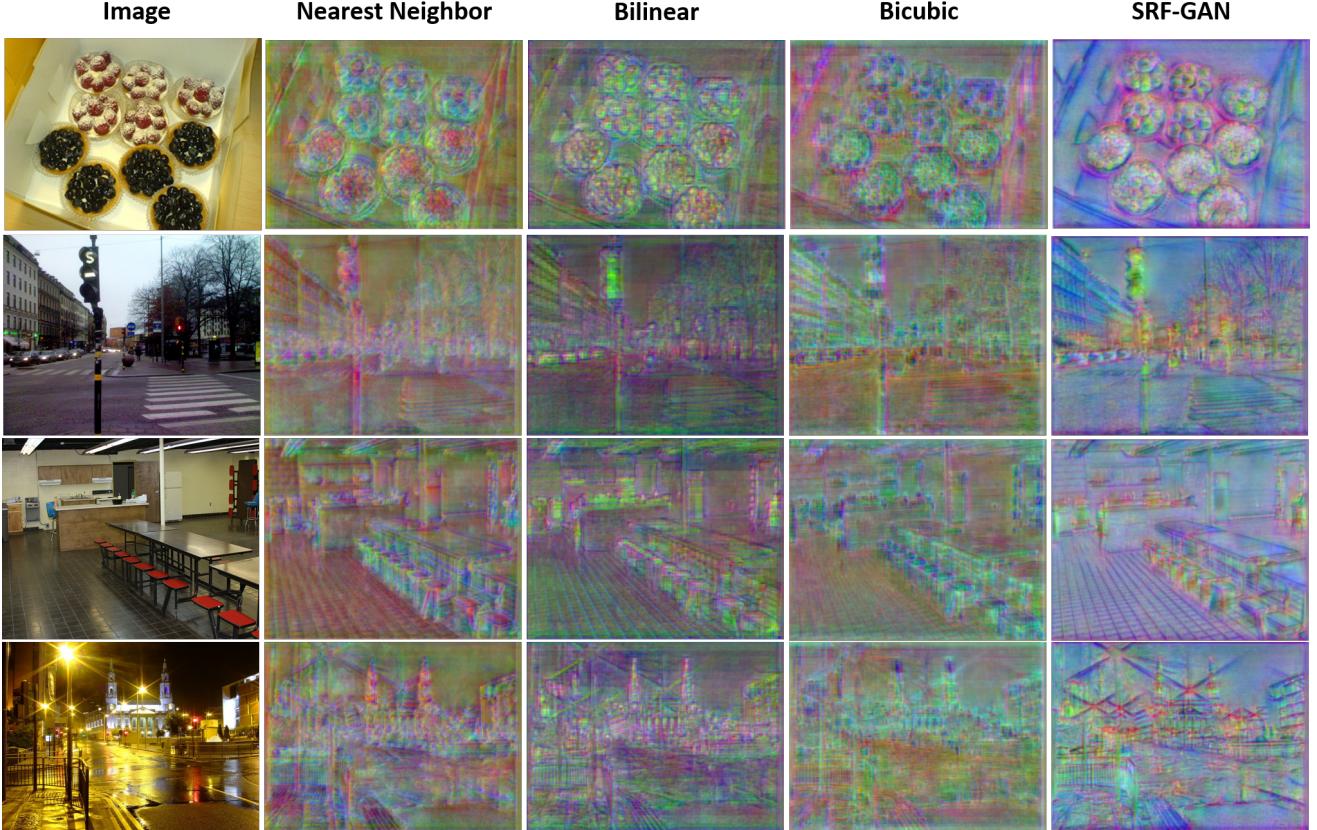
| Image | Nearest Neighbor | Bilinear | Bicubic | SRF-GAN |

Figure 2. Comparison of different interpolations and our SRF-GAN. We visualize a feature map P_2 of FPN by applying each method.

Interpolation method for FPN	AP^{box}	AP^{mask}
Nearest Neighbor	38.6	35.2
Bilinear	38.6	35.2
Bicubic	38.5	35.1
SRF generator (Ours)	41.2	37.0

Table 5. Comparison of different interpolation methods and our SRF generator on the COCO $val2017$ set.

Degradation function for low-resolution	AP^{box}	AP^{mask}
Nearest Neighbor	41.0	37.1
Bicubic	41.2	37.1
Bilinear (Ours)	41.2	37.0

Table 6. Comparisons of different degradation functions.

betters APs when using the SRF extractors trained with the Mask R-CNN head. The ability of the SRF generator might be improved more as generating super-resolved features for the finer feature map P_2 during training. This also means that we can improve the SRF extractor further by training it with finer feature maps than P_2. From these results, we could apply a pre-trained SRF extractor for any detector in practice since our SRF extractor has high flexibility.

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-A5): (A1) is the baseline using the NN interpolation; (A2) uses the SRF generator

for interpolation; (A3) is the adversarially trained detector during the training of multi-scale SRF extractor; (A4) maintains trained parameters of the SRF generator during training of the detector; (A5) is trained by using all our methods[3].

Table 4 shows the AP scores of (A1) - (A5). 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 the SRF extractor is more effective while training a target detector. 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 super-resolved features for detection and segmentation.

Interpolation method: As shown in Table 5, 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 SRF generator. As mentioned, these interpolation methods are used for upsampling feature maps before fusing them with other directional features in FPN. The differences

---

[3]In our implementation, adversarial training a SRF extractor directly without training SRF-GAN beforehand incurs the loss divergence.

$L_{SRF}(M, D, F)$	AP^{box}	AP^{mask}
Different feature levels	39.6	35.9
Same feature levels (Ours)	41.2	37.0

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

of AP scores are marginal between other different interpolation methods. However, our SRF generator provides the better results. Compared to others, our SRF generator improves box and mask APs by 2.6% and 1.8%. We also provide more qualitative comparisons of these methods in Fig. 2. Our SRF generator can capture finer details of objects over other interpolation methods. From these quantitative and qualitative results, we confirm that our method is more appropriate as an interpolation method for object detectors.

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 6, all the methods produce almost similar scores. It means that our learning methods are not sensitive to the image degradation functions.

Importance of semantic level matching: As discussed in Sec. 4.2, a multi-scale SRF 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 $\{P_3^{sr}, P_4^{sr}, P_5^{sr}\}$ and $\{P_2^{tr}, P_3^{tr}, P_4^{tr}\}$ 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 SRF extractor.

6. Conclusion

In this paper, we have proposed a novel super-resolved feature (SRF) generator for multi-scale feature representation. We have presented a SRF-GAN architecture and learning methods to train it effectively. From the extensive ablation study and comparison with SOTA detectors, our method indeed is beneficial to enhance detection and segmentation accuracy. In addition, it can be easily applied for the existing detectors. We believe that our method can substitute the conventional interpolation methods.

References

[1] Seung-Hwan Bae. Object detection based on region decomposition and assembly. In *AAAI*, volume 33, pages 8094–8101, 2019. 1

[2] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Sod-mtgan: Small object detection via multi-task generative adversarial network. In *ECCV*, pages 206–221, 2018. 3

[3] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, pages 2874–2883, 2016. 2

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, pages 9157–9166, 2019. 2

[5] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 2, 5, 7

[6] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, pages 4974–4983, 2019. 1, 2

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 2

[8] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv preprint arXiv:1901.03353*, 2019. 2

[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019. 1, 2

[10] Ross Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015. 2

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 2

[12] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. Augfpn: Improving multi-scale feature learning for object detection. In *CVPR*, pages 12595–12604, 2020. 1, 2, 6

[13] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Object instance segmentation and fine-grained localization using hypercolumns. *IEEE TPAMI*, 39(4):627–639, 2016. 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 1, 2, 4, 5, 6, 7

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI*, 37(9):1904–1916, 2015. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 3

[18] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, pages 845–853, 2016. 2

[19] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 734–750, 2018. 2

[20] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 4681–4690, 2017. 3

[21] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *CVPR*, pages 13906–13915, 2020. 1, 2, 5, 6, 7

[22] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, pages 702–716. Springer, 2016. 3

[23] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *CVPR*, pages 1222–1230, 2017. 3

[24] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *ICCV*, pages 6054–6063, 2019. 1, 6

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 1, 2, 3, 4, 5, 6

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 2, 5, 6, 7

[27] 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*, pages 740–755. Springer, 2014. 2, 6

[28] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 1, 2, 3, 6

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016. 1, 2

[30] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *CVPR*, pages 7363–7372, 2019. 2

[31] Mahyar Najibi, Bharat Singh, and Larry S Davis. Autofocus: Efficient multi-scale inference. In *ICCV*, pages 9745–9755, 2019. 2

[32] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: towards precise supervision of feature super-resolution for small object detection. In *ICCV*, pages 9725–9734, 2019. 3

[33] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, pages 821–830, 2019. 1, 6

[34] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020. 1, 2

[35] Jakaria Rabbi, Nilanjan Ray, Matthias Schubert, Subir Chowdhury, and Dennis Chao. Small-object detection in remote sensing images with end-to-end edge-enhanced gan and object detector network. *Remote Sensing*, 12(9):1432, 2020. 3

[36] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, pages 7263–7271, 2017. 2

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 2

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2

[39] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019. 3

[40] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *CVPR*, pages 3578–3587, 2018. 2

[41] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, pages 10781–10790, 2020. 1, 2, 3

[42] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, pages 9627–9636, 2019. 2, 6

[43] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE TPAMI*, 2020. 1

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

[45] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 6

[46] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, pages 6649–6658, 2019. 1, 2

[47] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 6

[48] 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*, pages 9759–9768, 2020. 6

[49] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2