SET: Spectral Enhancement for Tiny Object Detection

Supplementary Material

6. API: Perturbation Derivation

The adversarial perturbations discussed in Sec. 3.2 are injected to adversarially change the model output, thereby enhancing feature saliency in critical regions. This section details the derivation of the closed-form solution in Eqn. 6 following [12]. The optimization process in Eqn. 5 involves solving a min-max problem. First, it maximizes the perturbed losses with respect to $\epsilon_{i,\text{cls}}$ within the constraint $\|\epsilon_{i,\text{cls}}\| \leq \rho_i$. This is achieved by finding $\epsilon^*_{i,\text{cls}}$ using a Taylor expansion of the loss functions with respect to $\mathbf{P}_i$ around 0:

$$\mathcal{L}_{\text{cls}}(\mathbf{P}_i + \epsilon_{i,\text{cls}}) \approx \mathcal{L}_{\text{cls}}(\mathbf{P}_i + \nabla_{\mathbf{P}_i}\mathcal{L}_{\text{cls}}(\mathbf{P}_i) \cdot \epsilon_{i,\text{cls}}. \quad (10)$$

Introducing a dual norm problem, $\epsilon^*_{i,\text{cls}}$ is approximated as follows with $\frac{1}{p} + \frac{1}{q} = 1$:

$$\epsilon^*_{i,\text{cls}} \approx \rho \cdot \text{sign}(\nabla_{\mathbf{P}_i}\mathcal{L}_{\text{cls}}(\mathbf{P}_i) \cdot \frac{|\nabla_{\mathbf{P}_i}\mathcal{L}_{\text{cls}}(\mathbf{P}_i)|^{q-1}}{(\|\nabla_{\mathbf{P}_i}\mathcal{L}_{\text{cls}}(\mathbf{P}_i)\|_q^q)^{1/p}}. \quad (11)$$

With the inner maximization problem solved using $\epsilon^*_{i,\text{cls}}$, the minimization problem is addressed by calculating the gradient while excluding the Hessian term, as shown below:

$$\nabla_{\mathbf{P}_i} \max_{\|\epsilon_{i,\text{cls}}\| \leq \rho} \mathcal{L}_{\text{cls}}(\mathbf{P}_i + \epsilon_{i,\text{cls}}) \approx \nabla_{\mathbf{P}_i}\mathcal{L}_{\text{cls}}(\mathbf{P}_i + \epsilon^*_{i,\text{cls}}) \quad (12)$$

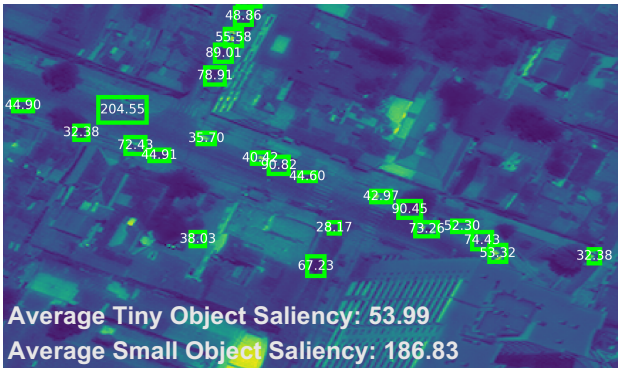
7. API: Facilitating Feature Robustness

In this section, we delve into how the Adversarial Perturbation Injection (API) module fosters robust feature representations through adversarial training. The optimization objective in Eqn. 5 can be decomposed into two components:

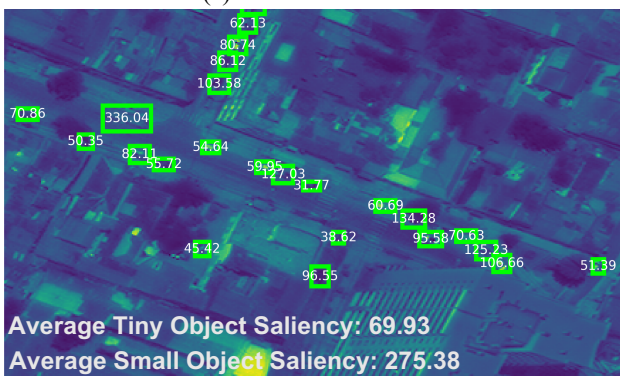
$$\begin{aligned} &\max_{\|\epsilon_{i,\text{cls}}\| \leq \rho_i} \mathcal{L}_{\text{cls}}(\mathbf{P}_i + \epsilon_{i,\text{cls}}) \\ &= \left[\max_{\|\epsilon_{i,\text{cls}}\| \leq \rho_i} \mathcal{L}_{\text{cls}}(\mathbf{P}_i + \epsilon_{i,\text{cls}}) - \mathcal{L}_{\text{cls}}(\mathbf{P}_i) \right] + \mathcal{L}_{\text{cls}}(\mathbf{P}_i). \end{aligned} \quad (13)$$

The first term of Eqn. 13 captures the sharpness of $\mathcal{L}_{\text{cls}}$ at $\mathbf{P}_i$ by measuring how quickly the training loss can be increased by moving from $\mathbf{P}_i$ to a nearby activation value, as described in Sharpness-Aware Minimization (SAM) [12]. This term imposes a penalty on sharp regions in the loss landscape, where predictions are highly sensitive to minor changes in activations of $\mathbf{P}_i$. The second term is the classification loss of original samples.

The regularization in Eqn. 13 encourages the model to prioritize robust attributes, thereby enhancing its ability to discriminate subtle characteristics of tiny objects despite the inherent low resolution.



(a) Vanilla FCOS



(b) FCOS w/ API

Figure 8. We provide a zoom-in example of the average saliency visualization. Note that the saliency maps are first computed based on the ℓ_2-norm of the gradient [13] at neck feature P_3. The saliency of each object is then summed and annotated within their respective bounding box. The average saliency of each object scale is calculated as the mean saliency across objects in the corresponding size category, displayed in the bottom-left corner of each image.

8. HBS: Variants of Smoothing Kernels

We supplement our analysis of the smoothing kernels in the HBS module. In Tab. 5, we compare convolutional smoothing (Eqn. 3) with adaptively calculated and fixed kernels of sizes 3×3, 5×5, and 7×7. Results reveal that all smoothing strategies surpass the FCOS baseline in model performance (AP), demonstrating the effectiveness of background smoothing. Convolutional smoothing with adaptively calculated kernels achieves a notable AP increase of 1.9% (round down) and 1.4% (round up). Based on the comparisons, HBS is fixed with the convolutional smoothing with adaptively calculated kernels as delineated in Eqn. 4 for the experiments within this paper.

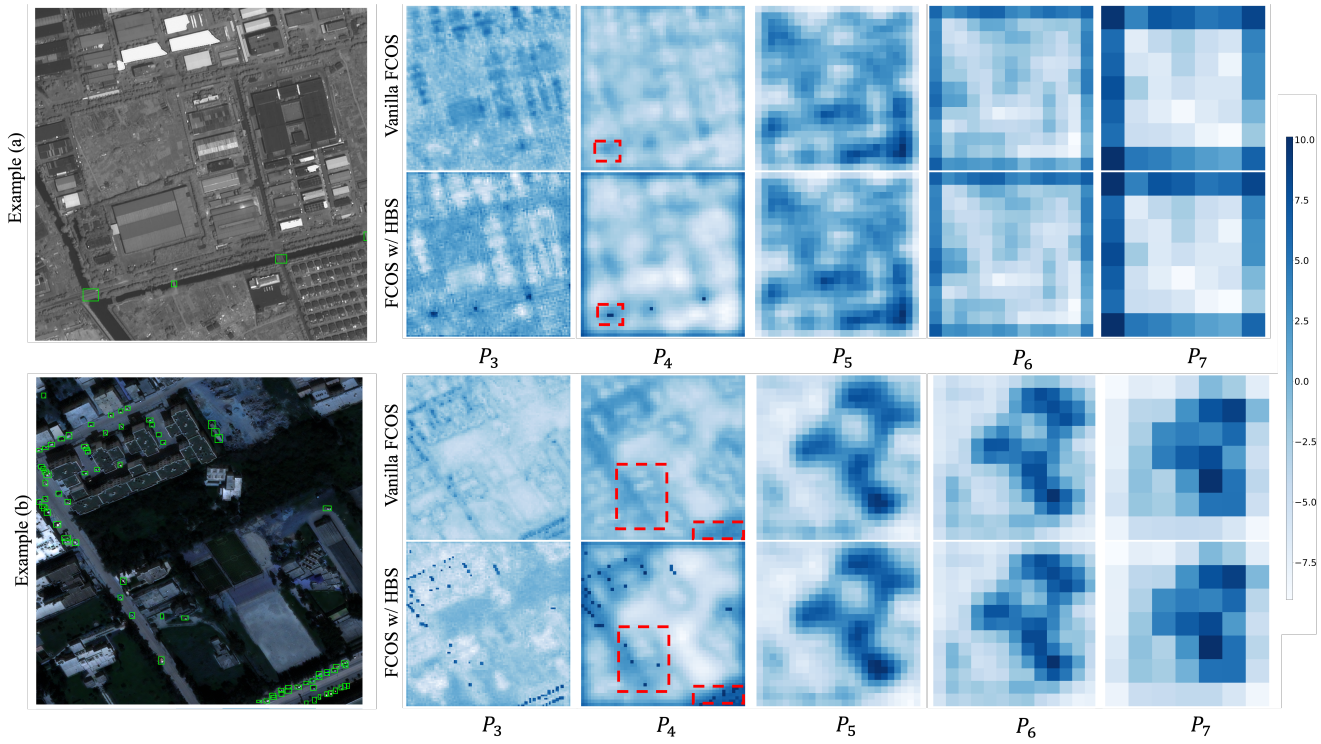


Figure 9. We visualize (a) the input image with ground-truths shown in green boxes, (b) P_3-P_7 feature in the vanilla FCOS after Principal Component Analysis (PCA) [46], and (c) P_3-P_7 feature in FCOS enhanced with HBS after PCA. Darker colors in the PCA maps indicate more critical components in the feature.

Table 6. Main results with various frameworks on TinyPerson. Note that models are trained on the TinyPerson `train` and validated on the TinyPerson `val`. We report APs (%) with different IoU threshold and APs (%) for objects in various sizes following the TinyPerson benchmark [56]. The **bold** denotes the best result.

Framework	Backbone	AP_{50}^{tiny}	A_{50}^{tiny1}	AP_{50}^{tiny2}	AP_{50}^{tiny3}	AP_{50}^{small}	AP_{25}^{tiny}	AP_{75}^{tiny}
FCOS [39]	R50	16.9	3.9	12.4	29.3	36.8.8	40.5	1.5
DSFD [39]	R50	31.1	14.0	35.1	46.3	51.6	59.6	2.0
Faster R-CNN [39]	R50	43.6	48.3	53.5	43.6	56.7	64.1	5.4
RetinaNet [24]	R50	15.5	3.0	14.4	29.1	46.8	48.4	1.3
RetinaNet w/ SET	R50	**17.1**$^{+1.6}$	**3.5**$^{+0.5}$	**15.7**$^{+1.3}$	**29.4**$^{+0.3}$	**47.0**$^{+0.2}$	**51.4**$^{+3.0}$	**1.4**$^{+0.1}$
AutoAssign [64]	R50	21.0	7.1	19.7	32.3	48.1	55.0	1.4
AutoAssign w/ SET	R50	**24.0**$^{+3.0}$	**7.3**$^{+0.2}$	**22.6**$^{+2.9}$	**37.0**$^{+4.7}$	**50.0**$^{+1.9}$	**58.0**$^{+3.0}$	**1.9**$^{+0.5}$

9. PCA Visualization of High-level Features

We present the PCA visualizations of P_3-P_7 features in Fig. 9 to further illustrate the effectiveness of HBS. The vanilla FCOS results for example (a) at the P_3 layer highlight the low discrimination challenge of tiny objects, where tiny objects (e.g., bridges) appear less distinct after feature encoding compared to background objects (e.g., houses). The **red** boxed areas in P_4 layer further demonstrate that tiny objects are overshadowed by dominant background information, particularly in high-level features. A comparison between vanilla FCOS and FCOS w/ SET reveals that HBS effectively smooths background regions and significantly enhances the contrast between foreground and background, which is critical for the overshadowed tiny objects. Moreover, the P_4 layer results in both examples clearly demonstrate the smoothing effect, where HBS reduces the intensity of patterns in multiple background regions, making them more vague.

10. Experiments on TinyPerson

We further conduct experiments on the tiny object detection dataset Tinyperson [56]. The TinyPerson dataset represents

Table 7. Results with various frameworks using the Swin-T backbone on AI-TOD. Note that models are trained on the AI-TOD `trainval` and validated on the AI-TOD `test`. We report APs (%) with different IoU threshold and APs (%) for objects in various sizes based on the AI-TOD criterion. The **bold** denotes the best result.

Framework	Backbone	AP	$AP_{0.5}$	$AP_{0.75}$	AP_{vt}	AP_t	AP_s	AP_m
FCOS [39]	Swin-T	11.9	30.2	7.2	2.7	11.6	16.4	23.7
FCOS w/ SET	Swin-T	**14.1**	**37.2**	**9.4**	**3.1**	**13.2**	**21.4**	**31.1**
ATSS [61]	Swin-T	12.7	32.2	7.6	**3.5**	11.7	17.2	26.1
ATSS w/ SET	Swin-T	**15.0**	**36.5**	**9.8**	3.1	**13.0**	**22.2**	**33.8**
AutoAssign [64]	Swin-T	10.5	28.1	5.3	4.0	12.1	12.9	15.6
AutoAssign w/ SET	Swin-T	**16.8**	**43.3**	**9.8**	**5.4**	**17.4**	**20.9**	**26.5**

Table 8. Results on SeaPerson [57]. Models are trained on SeaPerson `trainval` and tested on SeaPerson `test`.

Framework	AP_{50}	AP_{50}^{tiny}
Faster R-CNN	73.44	60.78
Faster R-CNN w/ SET	**79.65**	**65.40**

persons in extremely low resolutions, mainly less than 20 pixels. Notably, it includes a large number of tiny objects (smaller than 16×16 pixels). It serve as a valuable supplement to existing datasets, particularly in terms of the diversity it brings to the table in relation to poses and views.

Results in Tab. 6 show that our proposed method achieves 1.6% and 3.0% improvement regarding AP_{50}^{tiny} respectively, which are substantial gains. Moreover, SET consistently improves performance across various IoU thresholds and object sizes. For example, SET boosts the AP_{50}^{tiny3} by 4.7% for AutoAssign detector, a significant margin.

11. Experiments on SeaPerson

Moreover, we include results on SeaPerson [57], where SET achieves 6.21% AP_{50} improvement, highlighting its effectiveness in dense scenes.

12. Experiments using the Swin backbone

Unlike previous TOD methods [51, 52] that rely on receptive field computations, our method can be easily combined with other advanced backbone networks. To demonstrate the generalizability of SET, we conducted additional experiments using various detectors paired with the Swin-T backbone [27] on AI-TOD. Similar to the results obtained using the ResNet-50 backbone, the SET method improves the performance of both anchor-based detectors (ATSS [61] and RetinaNet [24]) and anchor-free detectors (FCOS [39] and AutoAssign [64]) by margins ranging from 3% to 6%, which is significant. As shown in Tab. 7, AutoAssign[64] with SET achieves an AP of 16.8%, deriving a significant improvement of 6.3% over the vanilla baseline. In addition, FCOS [39] trained with our SET shows a 2.2% AP increase.

Table 9. Detection performance on MS COCO [21]. Note that models are trained on COCO `train2017` and validated on COCO `val2017`.

Framework	AP	AP_{vt}	AP_t	AP_s	AP_m
FCOS [39]	36.4	7.9	19.6	27.2	43.6
FCOS w/ HBS	36.9	8.0	20.2	27.5	43.9
FCOS w/ API	37.0	8.1	20.3	27.6	44.0
FCOS w/ SET	**37.4**	**8.3**	**23.3**	**27.6**	**44.8**

13. Ablation Study on COCO

We further conduct ablation experiments on the general object detection benchmark MS COCO [21] to evaluate the effectiveness of each module. As shown in Tab. 9, SET achieves a 1.0% AP improvement over the FCOS baseline, with a significant 1.7% improvement in AP_t. The HBS module improves the baseline AP by 0.5%, with a 0.6% increase in AP_t and a 0.3% increase in AP_s, while the API module achieves a 0.6% improvement in AP, with a 0.7% increase in AP_t and a 0.4% improvement in AP_s. These results demonstrate the effectiveness of both modules in enhancing tiny object detection within general object detection tasks. Furthermore, while improving overall detection performance, the greater gains in AP_t validate our design is TOD specific.