Prompt-Free Universal Region Proposal Network

Supplementary Material

1. Ablation Study on *k*

In Sec. 3.2, we introduce sparsity to adaptively select the top-k informative feature maps for updating the learnable embedding. In this section, we ablate the choice of k in the Sparse Image-Aware Adapter module on the CD-FSOD benchmark to examine the effect of sparsity.

As shown in Tab. 7, PF-RPN achieves the best overall performance when $k = 2$, thereby choosing $k = 2$ as the default setting in our framework. Increasing k introduces more redundant feature maps and slightly degrades performance, while too small a k limits the available contextual information. This demonstrates that moderate sparsity offers the best trade-off.

Table 7. Ablation study of parameter k in the Sparse Image-Aware Adapter module on the CD-FSOD benchmark. Best results are highlighted in **bold**.

k	AR_{100}	AR_{300}	AR_{900}	AR_s	AR_m	AR_l
1	60.3	64.7	67.4	**44.2**	60.0	78.8
2	**60.7**	**65.3**	**68.2**	38.5	**61.9**	**80.3**
3	59.9	64.4	67.3	41.5	59.0	79.6
4	59.7	64.4	67.3	41.5	59.0	79.6

2. Ablation Study on Objective Loss

In our objective loss function, we introduce a hyperparameter λ to control the contribution of the centerness loss to the overall loss. To determine an appropriate setting, we conduct an ablation study on λ. As shown in Table 8, when λ is too small, the model does not learn to select queries located in the center regions. In contrast, when λ is too large, the centerness loss dominates optimization and negatively affects regression performance. The best performance is achieved when $\lambda = 5$.

Table 8. Ablation study of parameter λ in the Objective Loss on CD-FSOD benchmark. The best results are highlighted in **bold**.

k	AR_{100}	AR_{300}	AR_{900}	AR_s	AR_m	AR_l
1	60.5	65.1	68.1	39.6	59.8	79.4
3	60.6	65.1	68.1	39.6	60.3	79.5
5	**60.7**	**65.3**	**68.2**	38.5	**61.9**	**80.3**
7	58.5	63.1	66.3	37.1	58.3	79.2
9	59.6	64.3	67.4	39.7	59.6	78.9

3. Efficacy of Self-Prompt

As shown in Fig. 8, the object-internal feature in Fig. 8a focuses on the object region with high semantic consistency, whereas the learnable embedding in Fig. 8b yields a more diffused response.

These observations indicate that object-internal features exhibit stronger localization capability than the learnable embedding. Therefore, in Sec. 3.3, we leverage multi-level feature maps to update the learnable embedding, further enhancing its ability to localize objects based on internal visual cues.

(a) Cosine similarity heatmap between the 4th-level feature at the red point and the 4th-level visual embedding.

(b) Cosine similarity heatmap between the learnable embedding $\tilde{F}^T$ and the 4th-level visual embedding.

Figure 8. Comparison of feature localization between object-internal features and the learnable embedding $\tilde{F}^T$. The similarity map in Fig. 8a shows that the 4th-level feature at the red point focuses on the object region with high semantic consistency, while the learnable embedding $\tilde{F}^T$ in Fig. 8b produces a more diffused response, indicating weaker object correspondence.

4. Latency and Efficiency Analysis

The proposed iterative *Cascade Self-Prompt (CSP)* strategy introduces negligible latency overhead. As shown in Tab. 9, an increase in the number of CSP iterations from 1 to 3 yields consistent performance gains, accompanied by only a marginal rise in inference time (∼4.6 ms).

Table 9. Latency and performance analysis of different CSP iterations on the CD-FSOD benchmark.

Iteration	AR_{100}	AR_{300}	AR_{900}	AR_s	AR_m	AR_l	ms/img
Iter 1	59.6	63.1	65.4	35.7	57.8	77.7	**214.3**
Iter 2	59.9	63.7	65.9	36.7	57.9	78.2	216.7
Iter 3	**60.7**	**65.3**	**68.2**	**38.5**	**61.9**	**80.3**	218.9

Furthermore, the proposed method exhibits high flexibility and can be seamlessly integrated with lightweight detec-

tors to function as a real-time, high-performance region proposal network (RPN [34]). As demonstrated in Tab. 10, the integration of the proposed approach with YOLO-World [6] achieves competitive performance while preserving inference speeds comparable to those of conventional RPNs.

Table 10. Efficiency comparison of PF-RPN integrated with different detectors.

Metric	GLIP [27]	CasRPN [39]	RPN [34]	**PF-RPN(GDINO)**	**PF-RPN(YWorld)**
AR_{100}	47.6	45.8	32.0	**60.7**	52.3
FPS	5.5	24.8	27.8	4.6	**25.1**

5. Analysis of False Positives

The RPN is designed to detect all potential objects, a process that inevitably leads to the proposal of task-irrelevant regions, thereby generating false positives (FPs). In comparison to existing RPNs, the proposed PF-RPN assigns higher confidence scores to true positive object candidates while effectively suppressing irrelevant regions. As indicated in Tab. 11, the proposed method achieves more substantial improvements in AP and a lower number of false positives when restricted to 100 proposals compared to the 300-proposal setting. This result demonstrates the capacity of the proposed approach to prioritize high-quality candidates and mitigate redundant false positives.

Table 11. Analysis of false positives on different baselines under varying top proposal settings.

Metric	DeViT [49]		**DeViT + Ours**		CD-ViTO [12]		**CD-ViTO + Ours**	
	Top 100	Top 300	Top 100	Top 300	Top 100	Top 300	Top 100	Top 300
FP (↓)	18.2	21.7	17.1	18.3	15.3	20.2	**14.6**	17.9
AP (↑)	32.3	33.4	**37.3 (+5.0)**	37.1 (+3.7)	26.8	29.0	34.2 (+7.4)	35.1 (+6.1)

6. Dependence on Base Detectors

The proposed method exhibits strong extensibility and can be effectively integrated with various base detectors. As presented in Tab. 12, the proposed approach derives direct benefits from more powerful base detectors, demonstrating steady performance improvements as the capacity of the base model increases.

Table 12. Performance comparison of integrating PF-RPN with stronger base models (MMGrounding DINO [58]).

Method	AR_{100}	AR_{300}	AR_{900}	AR_s	AR_m	AR_l
MMGDINO-B	72.4	73.5	73.9	**46.4**	66.7	78.7
+ Ours	**76.3**	**78.9**	**79.8**	45.7	**75.2**	**86.3**
MMGDINO-L	73.6	74.2	74.5	48.3	69.8	79.4
+ Ours	**77.4**	**79.1**	**80.1**	**48.5**	**78.4**	**86.5**

7. Comparison with Previous Prompt-Free Methods

We compare PF-RPN with representative open-source prompt-free methods, GenerateU [22] and Open-Det [3]. We do not include CapDet [29] and DetCLIPv3 [52] due to the unavailability of their official code. As presented in Tab. 13, PF-RPN surpasses GenerateU by **+13.0** AR_{100} on CD-FSOD, while reducing VRAM usage by **95%** and accelerating inference by nearly **20**×. Note that PF-RPN is also faster than the baseline GDINO [27], primarily due to the removal of the computationally expensive text encoder.

Table 13. Comparison with open-source prompt-free methods regarding performance and efficiency.

Benchmark	Method	Average Recall (AR)			Scale (AR)			Efficiency	
		AR_{100}	AR_{300}	AR_{900}	S	M	L	FPS	VRAM
CD-FSOD	GDINO [27]	54.7	57.8	61.6	34.1	49.3	67.0	3.3	0.9G
	GenerateU [22]	47.7	54.1	55.7	28.1	48.3	69.4	0.22	12.2G
	Open-Det [3]	36.6	46.3	54.3	28.2	45.3	67.7	0.15	30.7G
	PF-RPN (Ours)	**60.7**	**65.3**	**68.2**	**38.5**	**61.9**	**80.3**	**4.6**	**0.5G**
ODinW13	GDINO [27]	69.1	70.9	72.4	40.8	64.6	78.4	3.3	0.9G
	GenerateU [22]	67.3	71.5	72.2	32.8	63.1	80.0	0.22	12.2G
	Open-Det [3]	53.9	62.9	69.1	27.7	59.8	76.6	0.15	30.7G
	PF-RPN (Ours)	**76.5**	**78.6**	**79.8**	**45.4**	**71.9**	**85.8**	**4.6**	**0.5G**

8. Detailed Experimental Results on All 19 Datasets

Comprehensive performance metrics of the proposed method are reported on various datasets across both the CD-FSOD benchmark and the ODinW13 benchmark. Additionally, Fig. 9 presents a visual line-chart comparison of these results against those of existing baseline methods.

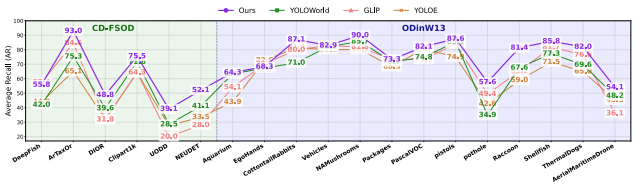

Figure 9. Detailed performance (AR) trends on all 19 target datasets compared to alternative methods.