Small Object Detection via Coarse-to-fine Proposal Generation and Imitation Learning: Supplementary Materials

Xiang Yuan Gong Cheng * Kebing Yan Qinghua Zeng Junwei Han
School of Automation, Northwestern Polytechnical University, Xi'an, China
{shaunyuan, kebingyan, zengqinghua}@mail.nwpu.edu.cn, {gcheng, jhan}@nwpu.edu.cn

1. Overview

This supplementary material is intended to improve the clarity and comprehensibility of our research. It primarily provides in-depth information about the training procedure and the construction of the exemplar set within the Feature Imitation branch. Finally, we describe the empirical limitations about our approach.

2. Details of Feature Imitation Branch

This part we elucidate the detailed settings of training the proposed Feature Imitation (FI) branch, including the policy of producing augmentations for high-quality samples and further discussions about non-high-quality samples, as well as the details about constructing and updating the exemplar feature set.

The Augmentation for High-quality Instances. In the **Training** part of Sec. 3.2 of our main paper, we refer to that the imitation process for a high-quality instance is performed between the feature of itself and its transformed features. In self-supervised contrastive learning paradigm, the only single *positive* sample for an image is generated by the transformation (*e.g.*, AutoAugment [2], RandAugment [3] and SimAugment [1]). Inspired by this setting, a function $\mathbf{\Gamma}$ is employed in our FI head to augment the features for high-quality instances who have an $IQ \geq T_{\mathrm{hq}}$. Specifically, we use random translation and zoom-in/out operation to augment the target features, and the corresponding functions are defined as $\mathbf{R}(s_{\mathrm{w}}, s_{\mathrm{h}})$ and $\mathbf{Z}(s_{\min}, s_{\max})$, where s_{w} and s_{h} represent the translation factors along the width-axis and height-axis of the ground-truth box respectively, while $s_{\min}$ and $s_{\max}$ indicate the minimum and maximum factors during zoom-in/out operation. Finally, the overall transformation function $\mathbf{\Gamma}$ is formulated as:

$$\mathbf{\Gamma}(x, y, w, h) = \{\mathbf{R}(x, y, w, h), \mathbf{Z}(x, y, w, h)\}, \qquad (1)$$

where (x, y, w, h) determines the region of proposal. In our practices, we use 8 pairs of $(s_{\mathrm{w}}, s_{\mathrm{h}})$ and 8 pairs of $(s_{\min}, s_{\max})$ to obtain 16 *positive* samples for a high-quality instance. The other transformations may bring better performance while we leave the future work to explore the optimal transformation functions, since the designed simple $\mathbf{\Gamma}$ could fulfill the imitation learning procedure for high-quality instances.

$\beta_1, \beta_2, \beta_3$	AP	AP_{eS}	AP_{rS}	AP_{gS}
Baseline	28.9	13.8	25.7	34.5
0.5, 0, 0	29.0	13.7	25.7	34.7
0.5, 0.1, 0.05	**29.5**	**14.4**	**26.3**	**35.1**
0.5, 0.2, 0.1	29.2	14.3	26.1	34.8

Table 1. The effect of different weights of non-high-quality instances to the performance, where β_1, β_2 and β_3 represent the loss weights of low-quality, mid-quality, and high-quality instances, respectively. 'Baseline' denotes Faster RCNN [4].

Discussions about Non-high-quality Instances. We use IQ to indicate the quality of an instance and its competence to be an exemplar by setting a threshold T_{hq} (practically set to 0.65), which implies that instances whose IQ scores are below the predefined T_{hq} are marked as **low-quality**. Is this reasonable? Two instances with the scores of 0.64 and 0.14 will be regarded equally as low-quality samples and conduct the imitation, however our core idea of introducing IQ is to mine the exemplars to guide the representation learning of samples with uncertain predictions. In other words, these two instances both will be marked as *uncertain/ambiguous*, and this is not rigorous because the prediction (classification scores and localization) of the former one ($IQ = 0.64$) is actually not bad. Hence, to mitigate this issue, we experimentally involve a low-quality threshold T_{lq} to discover those instances with high demand to be amended. Noting the introduction of T_{lq} will not change the overall training procedure depicted in Alg. 1 of our main paper, and the only difference lies in that we highlight the feature leaning of low-quality instances by assigning different loss weight

*Corresponding author: gcheng@nwpu.edu.cn

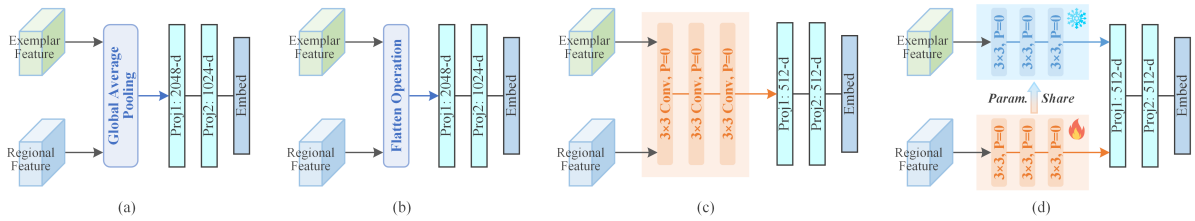

Figure 1. Four architectures for Feat2Embed module: (a) GAP-Embed, (b) Flatten-Embed, (c) Conv-Embed, and (d) SharedConv-Embed.

to instances with a quality score $T_{\text{lq}} \leq IQ < T_{\text{hq}}$ (noted as mid-quality instances) and that with $IQ < T_{\text{lq}}$ (noted as low-quality instances). Specifically, we conduct a series experiments to investigate the effect of such settings to the overall performance. As in Table 1, it is interesting that only focusing on the low-quality instances does not get the best results, and we conjecture this originates that the Feat2Embed module has not been optimized well with low-quality instances only, especially at early stage. Meanwhile, the undue concentration on those non-low-quality instances also poses negative impact to the learning of Feature Imitation branch. To sum up, the introduction of mid-quality instances can be regarded as a buffer area that is beneficial for stabilizing the training process and amending the representations of low-quality instances.

Details about the Exemplar Feature Set. The exemplar feature set is crucial in our method, and here we describe some details about its construction and updating rules. We empirically set the number of the samples for each ground-truth instance as 128, with half positive samples and half negative ones (except for high-quality instances). Moreover, the general rule of updating the exemplar set is resemble that of queue, namely *first in first out*. And the maximum size of the feature set for each category is 256 which is double to that of the sampling number for each instance. For the classes with limited high-quality ground truths, we halve the size of exemplar feature set and positive number to avoid that the feature set is unable to update for a long time.

Choices for Feat2Embed Module. In the Feature Imitation branch, we propose to measure the similarity between different RoI features in the embedding space with the help of the Feat2Embed module. Here, we explore the impact of different Feat2Embed designs on the performance of the FI branch. As demonstrated in Figure 1, we investigate four pipelines to perform the embedding process: (a) GAP-Embed, (b) Flatten-Embed, (c) Conv-Embed, and (d) SharedConv-Embed. These four architectures consist of two key components: dimensionality reduction and the embedding function. The primary difference among them lies in how they map the regional features to compact representations within the embedding space. We then utilize Faster RCNN as the baseline detector and conduct experiments to identify the optimal setting for the Feat2Embed module.

Feat2Embed	AP	AP_{eS}	AP_{rS}	AP_{gS}
Baseline	28.9	13.8	25.7	34.5
GAP-Embed	29.2	14.1	25.8	34.9
Flatten-Embed	29.4	**14.4**	26.1	**35.2**
Conv-Embed	**29.5**	14.2	**26.3**	**35.2**
SharedConv-Embed	**29.5**	**14.4**	**26.3**	35.1

Table 2. The effect of different Feat2Embed module designs to the performance of Feature Imitation branch, in which the term 'Baseline' denotes Faster RCNN [4].

Table 2 reveals that the proposed Feature Imitation branch demonstrates robustness to most designs except for GAP-Embed. We suspect that directly pooling the regional feature into a single vector results in significant information loss, thereby compromising the representation and similarity computation in the embedding space. Given that the number of parameters to be optimized in Flatten-Embed is approximately 60 times that of SharedConv-Embed, and the latter achieves a better average precision (AP_{eS}) performance compared to Conv-Embed, we choose SharedConv-Embed as our standard Feat2Embed module.

Empirical Limitations. Albeit facilitating the result of baseline detector on small objects especially on size-limited ones, the Feature Imitation branch may exhibit instability in performance. Empirically, the final performance of our feature imitation head significantly relies on the exemplars which dominate the imitation learning. However, the exemplar feature set constructed in each training procedure is distinct due to the dynamic of optimization. In other words, the exemplar features in current turn may fail to reach the bar of a high-quality teacher feature in next turn, and vice versa. Hence, a more flexible and general indicator of instance quality greatly contributes to a more elegant and effective method, and we leave this issue open to further research.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Cconference on Machine Learning*, pages 1597–1607, 2020.

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE Cconference*

on Computer Vision and Pattern Recognition, pages 113–123, 2019.

[3] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 2015.