Parameterized Prompt for Incremental Object Detection

Supplementary Material

A. Implementation Details

The pre-trained detectors use the original hyperparameter settings as those in their respective papers [47, 49]. We employ the Adam optimizer with a weight decay of 0.0001. The learning rate is set to 0.0001 for the class embeddings and the parametrized prompt structure, and a lower learning rate of 0.00001 is used for the bounding box embeddings, while freezing the remaining parameters. For training Deformable-DETR [47] on the PASCAL VOC2007 dataset [6], we train each incremental task for 100 epochs, with the learning rate dropped to 0.1 of its original value at the 80th epoch. For training Co-DETR [49] on the PASCAL VOC2007 dataset, we train each incremental task for 12 epochs, with the learning rate dropped to 0.1 of its original value at the 7th epoch. For training Co-DETR on the MS COCO dataset, we train each incremental task for 1 epoch. It is worth noting that in the 19+1 setting on PASCAL VOC2007, the second task contains only 279 images. Due to the limited data, Deformable-DETR tends to overfit during training. To mitigate this issue, we set the hyperparameters of the focal loss to $\alpha = 0.5$ and $\gamma = 3.0$. The reported accuracy is the average of three independent trials. All experiments are conducted on NVIDIA RTX 4090 GPUs.

Source of Pretrained Models. We use pretrained detectors available on HuggingFace. Specifically, we employ the Deformable-DETR pretrained on the MS COCO dataset, provided by SenseTime, which can be loaded via the transformers library. Additionally, we use the Co-DETR pretrained on the Objects365 dataset [30], provided by zongzhuofan, which can be loaded via the mmdetection library.

Algorithm 1 Parameterized Prompts Fusion for Incremental Task T_t

Require: θ_t, θ^f_{t-1}, θ_{init}, top-$k\%$, top-$l\%$
Ensure: θ^f_t

$\boldsymbol{v}_t \leftarrow \theta_t - \theta^f_{t-1}$
$\mu_t \leftarrow |\boldsymbol{v}_t|$, $\gamma_t \leftarrow \text{sgn}(\boldsymbol{v}_t)$
$\boldsymbol{v}^f_{t-1} \leftarrow \theta^f_{t-1} - \theta_{init}$
$\mu^f_{t-1} \leftarrow |\boldsymbol{v}^f_{t-1}|$, $\gamma^f_{t-1} \leftarrow \text{sgn}(\boldsymbol{v}^f_{t-1})$
$\mathcal{I}^f_{t-1} \leftarrow$ Top-$k\%$ indices of μ^f_{t-1}
$\mathcal{I}_t \leftarrow$ Top-$l\%$ indices of μ_t
for all parameter index i **do**
  if $i \in \mathcal{I}^f_{t-1}$ **then**
    $\theta^f_t[i] \leftarrow \theta^f_{t-1}[i]$ ▷ Retain important
  else if $i \in \mathcal{I}_t \setminus \mathcal{I}^f_{t-1}$ **then**
    $\theta^f_t[i] \leftarrow \theta_t[i]$ ▷ Retain important
  else if $\gamma_t[i] = \gamma^f_{t-1}[i]$ **then**
    $\theta^f_t[i] \leftarrow \frac{1}{2}(\theta_t[i] + \theta^f_{t-1}[i])$ ▷ Average consistent
  else
    $\theta^f_t[i] \leftarrow \theta^f_{t-1}[i]$
  end if
end for
return θ^f_t

B. Additional Experiment Results

B.1. Single-step Comparison on MS COCO dataset

We compare two single-step scenarios on the MS COCO dataset, namely the 40+40 and 70+10 settings. As shown in Tab. 5, P^{2}IOD performs excellently across all experimental settings. Compared to the prompt-pool-based MD-DETR, P^{2}IOD achieves AP_{50} accuracy improvements of 5.4% and 6.3% in the two scenarios, demonstrating that the proposed method effectively addresses the prompts pool confusion.

B.2. Impact of Parameterized Prompt Fusion Threshold

We quantitatively analyze the impact of the top-k and top-l in parameterized prompt fusion. The threshold determines the proportion of the previous and current parameterized prompt structures retained during fusion. To clarify the impact of one-step fusion on accuracy, we perform experiments in the PASCAL VOC2007 under 10+10 setting. As shown in Tab. 6, we observe that neither introducing model fusion (no fusion) nor using only the first task's parameterized prompts (top-$k = 1.0$) results in poor performance. When top-$k = 0.0$ and top-$l = 0.0$, the method averages the consistent parameters, improving the performance compared to the non-fused approach, demonstrating that averaging consistent parameters enhances generalization. Furthermore, by comparing different values of top-k and top-l, we find that moderately retaining parameters from both the current and previous tasks can further improve performance. The accuracy improvement indicates that retaining key parameters from each task helps preserve task-specific knowledge. We observe the best performance at top-$k = 0.7$ and top-$l = 0.3$, resulting in a 1.08% accuracy increase over the non-fused method.

B.3. Impact of λ in Sparse Loss

We quantitatively analyze the impact of λ, which controls the sparsity of parameterized prompts structure. A larger

Table 5. Average precision is compared on the MS COCO dataset under single-step settings of 40+40 and 70+10.

Scenarios	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
40 + 40	RILOD [18]	29.9	45.0	32.0	15.8	33.0	40.5
	SID [29]	34.0	51.4	36.3	18.4	38.4	44.9
	ERD [7]	36.9	54.5	39.6	21.3	40.4	47.5
	CL-DETR [25]	42.0	60.1	45.9	24.0	45.3	55.6
	LEA [33]	41.2	59.8	44.8	25.7	45.1	54.8
	SDDGR [16]	43.0	62.1	47.1	24.9	46.9	57.0
	GCD [36]	45.7	62.9	49.7	28.4	49.3	60.0
	MD-DETR (Objects365)[2]	50.0	65.7	55.1	35.7	54.1	65.7
	P^{2}IOD (Objects365)	**54.7**	**71.1**	**60.3**	**39.2**	**59.4**	**69.7**
70 + 10	RILOD [18]	24.5	37.9	25.7	14.2	27.4	33.5
	SID [29]	32.8	49.0	35.0	17.1	36.9	44.5
	ERD [7]	34.9	51.9	37.4	18.7	38.8	45.5
	CL-DETR [25]	40.4	58.0	43.9	23.8	43.6	53.5
	LEA [33]	45.4	65.3	49.6	28.5	48.8	59.2
	SDDGR [16]	40.9	59.5	44.8	23.9	44.7	54.0
	GCD [36]	46.7	63.9	50.8	29.7	49.9	61.6
	MD-DETR (Objects365)[2]	50.8	65.6	55.8	34.9	55.8	66.0
	P^{2}IOD (Objects365)	**55.2**	**71.9**	**60.7**	**41.0**	**59.7**	**70.5**

Table 6. Results (AP_{50}, %) on different fusion thresholds in the parameterized prompt fusion on PASCAL VOC2007 under the 10+10 setting.

Fusion Threshold		**10+10**		
top-k	top-l	**1-10**	**11-20**	**1-20**
no fusion		80.72	79.54	80.13
0.0	0.0	81.57	79.92	80.74
0.0	0.3	81.16	80.06	80.61
0.0	0.7	80.75	79.62	80.19
0.3	0.3	81.62	80.61	81.11
0.3	0.7	80.46	80.55	81.00
0.7	0.3	**82.00**	**80.42**	**81.21**
0.7	0.7	81.96	80.44	81.20
1.0	-	81.64	79.78	80.71

Table 7. Results (AP_{50}, %) on different λ in the sparse loss on PASCAL VOC2007 under the 5+5+5+5 setting.

λ	**5+5+5+5**		
	1-5 (T_1)	**6-20** ($T_2+T_3+T_4$)	**1-20**
0	73.1	76.0	75.3
1×10^{-6}	73.1	76.5	75.7
3×10^{-6}	73.4	76.8	76.0
1×10^{-5}	**74.0**	**77.2**	**76.4**
3×10^{-5}	73.7	77.0	76.2
1×10^{-4}	73.9	76.8	76.1

λ enforces sparser weights. We conduct experiments in the PASCAL VOC2007 under the 5+5+5+5 setting. Tab. 7 presents the accuracy results across different λ values. We observe that when λ is too small (e.g., 1×10^{-6}), insufficient sparsity fails to concentrate critical knowledge in important parameters, leading to poor performance. Conversely, when λ is too large (e.g., 1×10^{-4}), excessive sparsity limits the capacity for preservation of task knowledge, leading to performance degradation. $\lambda = 1\times10^{-5}$ strikes the optimal balance, achieving the highest accuracy of 76.4% and providing 1.1% improvement over the baseline without sparse loss. The experiment demonstrates that appropriate sparsity in parameterized prompt structure can help preserve knowledge.

B.4. Proposal Compression

MD-DETR [2] proposes compressing object-related proposals and introduces a localized query retrieval (QL) method. QL uses a fully connected layer with an input dimension of $N\times D$ and an output dimension of N to generate weights for each proposal. At the same time, QL utilizes the assignment obtained by the Hungarian matching criterion as a supervision signal and constructs a cross-entropy regularization term to supervise the weights of each proposal. However, this design leads to a fully connected layer with a parameter size of $N^2\times D$, significantly larger than the prompts pool size. The excessive number of parameters not only reduces storage efficiency but also leads to training difficulties. In the code released by MD-DETR, we observe that although the loss in QL does not converge, QL still has a positive effect on accuracy. Inspired by this observation, we designed a Self-Attention Compression (SAC)

Table 8. Results (AP_{50}, %) on different structrues for proposal compression on PASCAL VOC2007 under the 5+5+5+5 setting.

Method	5+5+5+5		
	1-5	**6-20**	**1-20**
MD-DETR - QL	55.2	63.6	61.5
MD-DETR - SAC	68.9	62.3	64.0
MD-DETR - average	39.0	45.0	43.5
P^{2}IOD - SAC	40.9	72.6	64.7
P^{2}IOD - w/o compression	45.9	70.5	64.3
P^{2}IOD - average	**74.0**	**77.2**	**76.4**

module to more effectively compress object-related proposals. We first aggregate the feature information of each proposal by applying global average and max pooling operations along the embedding dimension D of the proposal P, generating two different proposal descriptors: F_{avg} and F_{max}, which represent the average-pooled and max-pooled proposals, respectively. We then apply the shared fully connected layer to both descriptors, fuse the resulting feature vectors through element-wise summation, and apply a sigmoid activation function to obtain the final attention map $M \in \mathbb{R}^N$. In short, the attention map is computed as follows:

$$M = \sigma(FC(AvgPool(P)) + FC(MaxPool(P))) \quad (10)$$

where σ denotes the sigmoid activation function, and FC represents the shared fully connected layer. Compared to the $N^2 \times D$ parameter size required by QL, the fully connected layer in our method contains only N^2 parameters, significantly reducing the required number of parameters. To verify the effectiveness of SAC, we first integrate it into MD-DETR [2]. We conduct experiments on the PASCAL VOC2007 dataset using the MS COCO pretrained detector. As shown in Tab. 8, replacing QL with our proposed SAC module improves performance, demonstrating that our method achieves more effective proposal compression with fewer parameters.

We integrate the proposed SAC module into P^{2}IOD, and an inconsistency arises compared to its integration into MD-DETR. As shown in Tab. 8, adding the SAC module does not improve accuracy as in MD-DETR but instead reduces performance. Averaging all background and object proposals achieves the best performance. We attribute this discrepancy to differences in prompt structure. In MD-DETR, the prompt matching mechanism in the prompts pool requires query features to be task-discriminative, so removing background information enables better matching. Additionally, the prompts pool stores a limited number of prompts and linearly combines them. The limited representational capacity forces the prompts pool to retain only the most distinctive information, namely, object-related information. In contrast, the parameterized prompts in P^{2}IOD act as a mapping from the proposal domain to the prompt domain, offering a larger representational space that allows for the inclusion of background information. In detection tasks, including background information helps the detector distinguish between foreground and background. Therefore, removing background information via the SAC module results in suboptimal prompts. The above experiments suggest that in prompt-based IOD, background-related knowledge should be incorporated into the prompts. The prompts pool is not suitable for IOD due to its structural limitations in providing background knowledge. Furthermore, as shown in Tab. 8, removing the averaging operation leads to performance degradation, indicating that proposal compression is necessary. Without proposal compression, the number of parameters in the parameterized prompts increases dramatically. Although this setup preserves complete proposal information, the excessive number of trainable parameters hinders its convergence. The averaging operation maintains a reasonable parameter size while simultaneously retaining both background and object information. Therefore, P^{2}IOD

Table 9. Lookup table for variable definition in the paper.

Variable	Definition
T_t	task t
θ^*	frozen parameters
θ	parametrized prompt
θ_t	parameterized prompt after task t training
θ_t^f	parameterized prompt after task t fusion
$\boldsymbol{v}_t$	task vector
μ_t	magnitude of task vector
$\mathcal{I}_t$	top indices of μ_t
γ_t	direction of task vector
x	input image
P	proposals
N	number of proposals
D	dimension of embedding
Q	query function
p	prompts
L_p	length of prompts
d	hidden layer dimension
W	weight of MLP layer
L_s	sparse loss
λ	sparse loss hyperparameter
$\hat{y}_i$	detector prediction
$\hat{s}_i$	score for prediction
$\hat{b}_i$	bounding box coordinates for prediction
$\tilde{y}_i$	pseudo label
$\tilde{c}_i$	category for pseudo label
$\tilde{b}_i$	bounding box coordinates for pseudo label
τ	threshold of Pseudo Labeling

adopts averaging as its proposal compression strategy.

C. More Explanations

This section provides more details about the parameterized prompt fusion algorithm and variable definition.

C.1. Parameterized Prompt Fusion

The details of the parameterized prompt fusion algorithm are presented in Alg. 1.

C.2. Variable Definitions

All variable definitions used in our method are listed in Tab. 9.