DPC: Dual-Prompt Collaboration for Tuning Vision-Language Models

Supplementary Material

A. More Implementation Details

Herein, we provide additional detailed setup of DPC to enhance the reproducibility of our model.

A.1. Experimental Setup

Datasets. As described in the main text, for most datasets, we restrict the size of mini-batch sampled by the Dynamic Hard Negative Optimizer to $L \leq 32$ when executing DPC fine-tuning. The seed of DPC is consistent with backbone.

However, it is important to note that for the DTD [2] and OxfordPets [25] datasets, after the base and new splitting, there are only 24 and 19 sub-classes that involved in the base tasks, respectively, which are fewer than 32. Since the Dynamic Hard Negative Optimizer requires maintaining the size of the mini-batch smaller than the quantity of base classes (otherwise, the effectiveness of hard negative selection would be compromised), we reduce the parameters to $b = 2$ and $K = 8$ for these two datasets, ensuring $L \leq 16$. Furthermore, since EuroSAT [9] possesses only 5 base classes, we set $b = 2$ and $K = 2$ during optimization on this dataset.

Apart from these, the data sampling strategy for the inference process and fine-tuning on the backbone remains consistent with the baselines.

Hyperparemeters. Following the setup of the backbones, we utilize the ViT-B/16-based CLIP as the foundation model for prompt learners. For a fair comparison, all backbones and DPC are fine-tuned for epochs $ep = 20$ with learning rate $lr = 0.002$ for base-to-new tasks to avoid gradient explosion, and a 16-shot setting is applied to all models except PromptKD backbone. For cross-dataset tasks, we follow the PromptSRC [15] settings, fine-tuning on all categories of ImageNet with $ep = 5$ and a learning rate $lr = 0.0035$, while reducing the depth of visual and text prompts to 3 (except for CoOp).

Detailed information of the text and visual prompt settings is enumerated in Tab. 6. For the initialization process, text prompt in CoOp is randomly initialized adhering a zero-mean Gaussian distribution, while the other 3 backbones apply the encoded "A photo of a" tokens as the initialization template. Additionally, PromptSRC and PromptKD follow the Independent Vision-Language Prompt (IVLP) [28] setting, where prompts related to the two modalities are independently initialized. We use 1 Tesla A40 GPU to perform 3 runs on each dataset.

Algorithm. In Fig. 8, we demonstrate the DPC procedure as pseudo-code. For clarity, although the mini-batch sampled by DPC through the Dynamic Hard Negative Optimizer has a variable size L, we annotate the tensor dimensions in the comments with the assumption $L = b \cdot K$, meaning that all the hard negative objects sampled in this mini-batch are non-repetitive. This hypothesis does not affect the actual process of the model.

```
# T: mini-batch of text annotations
# I: mini-batch of images
# W: collaboration weight (W_b = base class, W_n = new class)
# HNS: Hard Negative Sampler
# FF: Feature Filter
# CE: Cross Entropy loss

### Dual prompts initialization
tuned_prompt = backbone_prompt_learner(T, I) # frozen
parallel_prompt = nn.Parameter(tuned_prompt) # learnable
mixed_prompt = W_b * parallel_prompt + (1-W_b) * tuned_prompt

### STAGE 1: Training stage on base class
# obtain features of image and text
T_feat = text_encoder(parallel _prompt) # [n_cls, dim]
I_feat = image_encoder(I) # [batch, dim]

# apply negative sampler to get hard negative features
# size of T_feat and I_feat after sampling: both are [batch*TopK, dim]
for (text, image) in batch:
   hn_t, hn_i = HNS ((text, image), tuned_prompt) # [TopK - 1]
   text_feat = FF (text_feat, [text_id, hn_t_id]) # [TopK, dim]
   img_feat = torch.cat([img_feat, image_encoder(hn_i)]) # [TopK, dim]

# Hard Negative Optimizer
logits_img = logit_scale * hn_I_feat @ filtered_T_feat.t()
logits_txt = logits_img.t() # [batch*TopK, TopK*batch]
ids = torch.arange(batch*TopK)
loss = (CE(logits_img, ids) + CE(logits_txt, ids)) / 2

### STAGE 2: Inference on base & new class
# inference on base
logit = similarity_head(text_encoder(mixed_prompt), I_feat)

# inference on new
mixed_prompt = W_n * parallel_prompt + (1-W_n) * tuned_prompt
logit = similarity_head(text_encoder(mixed_prompt), I_feat)
```

Figure 8. Pseudo-code of DPC in PyTorch. The size of dynamic hard negative mini-batch is considered as $L = b \cdot K$ for easier understanding.

Params	CoOp	MaPLe	ProSRC	ProKD
Text prompt depth	1	9	9	9
Visual prompt depth	-	9	9	9
Context length	(4,0)	(2,4)	(4,4)	(4,4)
Prompt layer	1	12	12	12
Optimizer	SGD	SGD	SGD	SGD

Table 6. Training settings of backbones for base-to-new tasks.

A.2. DPC Optimization for Backbones

As a robust plug-and-play module, the DPC Optimizer performs targeted modifications to various backbones based on separate forms of prompts and model architectures to achieve complete model adaptation. In this section, we provide a brief introduction to the frameworks of the 4 selected backbones and declare the specific strategies for introducing and fine-tuning the DPC module.

CoOp [49]. CoOp briefly introduces a randomly initialized text prompt to replace the original fixed template "A photo of a [CLASS]". Obeying the introduction of the DPC framework in the main text, we first fine-tune the original CoOp backbone to obtain the tuned text prompt $\boldsymbol{P}$. Subsequently, for the DPC optimizer, we append the parallel prompt $\boldsymbol{P}'$ into the text modality for dual-prompt collaboration, while replacing the cross-entropy loss of CoOp with the contrastive learning loss in hard-negatives $\mathcal{L}_{\text{CL}}$ of DPC for subsequent incremental fine-tuning.

MaPLe [14]. MaPLe integrates visual and text prompts by establishing a set of activated feature mapping layers, which derive corresponding visual prompts from learnable text prompts. Within the DPC framework, after fine-tuning the original backbone, we obtain sets of visual and text prompts $(\boldsymbol{P}_{vi}, \boldsymbol{P}_{ti})$ as initial values for the parallel prompts $(\boldsymbol{P}_{vi}{}', \boldsymbol{P}_{ti}{}')$ and load the weight parameters of the feature mapping layers to initialize the DPC optimizer. Since only text prompts $\boldsymbol{P}_{ti}$ are learnable in MaPLe, similar to CoOp, we upgrade the cross-entropy loss of MaPLe to DPC contrastive learning loss $\mathcal{L}_{\text{CL}}$ in subsequent stages, while continuously optimizing the text-based parallel prompts $\boldsymbol{P}_{ti}{}'$ while keeping the mapping layers for visual prompts activated within DPC. In the Weighting-Decoupling weight accumulation module during the inference process, we apply the same base-class weights ω_b or new-class weights ω_n for prompts of both modalities.

PromptSRC [15]. PromptSRC employs independent visual and text prompts for fine-tuning, following the IVLP setting. It introduces more robust loss functions as constraints to mitigate the negative impact of the BNT problem. Specifically, in addition to the cross-entropy loss $\mathcal{L}_{\text{CE}}$ adopted by typical prompt learners, PromptSRC also appends consistency constraints between the prompts and their corresponding modality features, $\mathcal{L}_{\text{SCL-image}}$ and $\mathcal{L}_{\text{SCL-text}}$, as well as a further constraint between the logits after modality interaction, $\mathcal{L}_{\text{SCL-logits}}$, to balance the base-new performance.

Therefore, in the DPC framework, after obtaining the visual and text tuned prompts $(\boldsymbol{P}_{vi}, \boldsymbol{P}_{ti})$ optimized by the backbone, we construct parallel prompts $(\boldsymbol{P}_{vi}{}', \boldsymbol{P}_{ti}{}')$ based on both modalities, keeping them activated to sustain learnability. During DPC optimization, we replace the original $\mathcal{L}_{\text{CE}}$ with $\mathcal{L}_{\text{CL}}$ that corresponding to DPC, while the other 3 loss functions are directly inherited by the DPC optimizer, collectively contributing to the continuous fine-tuning of the parallel prompts. Similarly, during inference stage, the visual and text parallel prompts $(\boldsymbol{P}_{vi}{}', \boldsymbol{P}_{ti}{}')$ are still weight-accumulated with the corresponding tuned prompts in relevant modalities of PromptSRC backbone to achieve intra-modality dual-prompt collaboration.

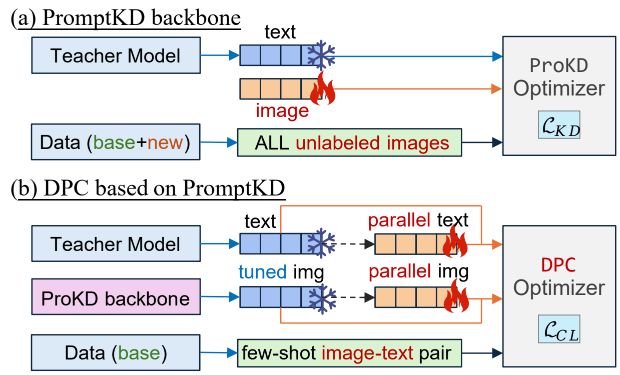

Figure 9. Initialization of text & image prompts and optimizers in PromptKD backbone and DPC.

Model	Avg. accuracy		
	Base	New	H
CLIP [27]	69.34	74.22	71.70
CoCoOp [48]	80.47	71.69	75.83
KgCoOp [42]	80.73	73.60	77.00
TCP [43]	84.13	**75.36**	79.51
DPC-SRC	**86.10**	74.78	**80.04**
KDPL [23]	77.11	71.61	74.26
CoPrompt [30]	84.00	77.23	80.48
DPC-PK	**87.55**	**80.55**	**83.91**

Table 7. Comparison with additional prompt tuning baselines fine-tuned based on internal constraints or external knowledge for base-to-new tasks. DPC-SRC denotes a combination of DPC and PromptSRC, while DPC-PK is a binding of DPC and PromptKD.

PromptKD [20]. As a knowledge distillation-driven model, PromptKD introduces PromptSRC fine-tuned on larger ViT-L/14 as the teacher model. Unlike other backbones, PromptKD processes unlabeled images from the entire dataset and applies the teacher model to infer and optimize the student model across all base and new classes during fine-tuning. In this procedure, the text prompts $\boldsymbol{P}_{ti}$ extracted from the teacher model are frozen, while only the prompts in the visual branch $\boldsymbol{P}_{vi}$ and a projection layer $h(\cdot)$ for aligning the student model with the teacher model are updated.

To integrate the DPC optimizer into PromptKD, we devise a targeted framework, as shown in Fig. 9. Initially, we fine-tune the original PromptKD backbone to obtain the visual prompts $\boldsymbol{P}_{vi}$ and the parameters of the projection layer.

Subsequently, we fully modify the Dataloader of PromptKD, altering it from loading unlabeled images across all categories to sampling few-shot image-text pairs from base classes, aligning it with other prompt tuning backbones.

Corresponding to the data input modification, fine-tuning strategy of PromptKD is also momentously updated to accommodate our DPC optimizer. Specifically, we construct parallel prompts ($\boldsymbol{P}_{vi}{}'$, $\boldsymbol{P}_{ti}{}'$) based on the frozen text prompts from the teacher model $\boldsymbol{P}_{ti}$ and the visual prompts fine-tuned by PromptKD $\boldsymbol{P}_{vi}$, then set both of them activated. During DPC fine-tuning, all original loss functions of PromptKD are disabled, while only the DPC image-text contrastive loss $\mathcal{L}_{\text{CL}}$ is applied for further optimization. It is worth noticing that to maintain the original generalization performance of PromptKD, the contrastive loss is applied under the ViT-L/14 setting of teacher model, transmitting the text parallel prompts $\boldsymbol{P}_{ti}{}'$ and the visual parallel prompts upscaled by the activated projection layer $h(\boldsymbol{P}_{vi}{}')$ as inputs to the feature encoders. The weight accumulation procedure is consistent with DPC in PromptSRC.

In summary, being constructed as an independent task, DPC is introduced into PromptKD based on few-shot image-text data as a plug-and-play module. Aforementioned design successfully integrates DPC optimization while preserving the original performance of PromptKD.

B. More Experimental Results

Herein, we supplement the main text with more elaborated experiments. Performance comparisons with additional prompt tuning baselines (§B.1), the specific impact of collaboration weights (ω_b, ω_n) on each dataset (§B.2), similarity measurements of samples from the Hard Negative Sampler (§B.3), the effects of the DPC optimizer on the visual or text branches (§B.4), more ablation studies on DPC components (§B.5), and assessments of computational cost (§B.6) are contained in this section.

B.1. Compare with More Baselines

To further highlight the comprehensive performance advantages of DPC, more baselines are brought in for comparison on base-to-new generalization tasks. As illustrated in Tab. 7, we compare DPC based on PromptSRC (DPC-SRC) with the initial CLIP and other models optimized by internal constraints, containing CoCoOp [48], KgCoOp [42], and TCP [43]. For knowledge distillation-based models, KDPL [23] and CoPrompt [30] are utilized for contrasting with the combination of DPC and PromptKD (DPC-PK). It is apparent that models reinforced by DPC surpass the current baselines, achieving the latest State-Of-The-Art performance.

B.2. Detailed Ablation on Collaboration Weights

Impact of Different Values. In Tab. 8 and Tab. 10, we comprehensively compare the performance of various col-

Dataset	weight for base class (ω_b)					
	0	0.1	0.2	0.3	0.5	1
ImageNet	76.41	77.72	77.72	**77.95**	77.92	77.58
Caltech101	97.55	98.32	**98.58**	98.39	98.39	98.00
StanfordCars	75.69	81.21	81.13	81.33	**81.41**	81.36
SUN397	80.99	82.58	**82.81**	82.54	82.67	82.33
Food101	90.49	91.09	91.15	**91.18**	91.12	91.08
DTD	80.09	84.95	84.61	**85.76**	85.53	83.22
EuroSAT	87.60	93.32	93.40	**93.79**	92.29	91.50
Flowers102	96.96	98.10	98.86	98.96	**98.77**	98.67
OxfordPets	95.06	95.11	**95.80**	95.48	95.27	94.90
UCF101	83.66	86.76	**87.02**	85.52	85.83	86.19
FGVCAircraft	37.33	42.38	**45.56**	45.26	44.00	42.20
Avg.	81.98	84.69	**85.15**	85.11	84.84	84.28
Δ	+0.00	+2.71	+3.17	+3.13	+2.86	+2.30

Table 8. Ablation study on the impact of collaboration weight for base (ω_b) of DPC. Benefiting from Weighting-Decoupling structure, weights for base or new can be different.

Method	weight	Base	New	H	Δ
MaPLe		83.52	73.31	78.08	
+DPC	0.2	85.07	73.31	78.75	+0.67
+DPC	1.0	**85.93**	**73.31**	**79.12**	+1.04

Table 9. Impact of collaboration weight for base (ω_b) on DPC based on MaPLe [14] backbone. Analysis of this phenomenon is exhibited in Appendix B.2.

Dataset	weight for new class (ω_n)					
	0	0.01	0.02	0.05	0.1	0.2
ImageNet	**68.85**	68.75	68.77	68.62	68.26	67.53
Caltech101	94.65	94.98	**95.09**	94.98	94.87	94.76
StanfordCars	**70.14**	69.79	69.42	68.61	66.92	63.04
SUN397	**74.10**	74.05	74.03	73.74	73.17	71.40
Food101	91.47	91.47	91.54	91.53	**91.57**	91.49
DTD	49.88	50.00	**50.00**	49.76	47.95	45.77
EuroSAT	**51.62**	51.41	51.08	49.31	46.05	39.79
Flowers102	**68.37**	68.30	68.23	67.66	66.67	65.11
OxfordPets	97.60	97.54	97.54	**97.60**	97.43	97.43
UCF101	**66.31**	65.66	65.33	64.09	63.66	62.52
FGVCAircraft	24.24	23.94	24.06	24.12	24.25	**25.85**
Avg.	**68.84**	68.72	68.64	68.18	67.35	65.88
Δ	+0.00	-0.12	-0.20	-0.66	-1.49	-2.96

Table 10. Ablation study on the impact of collaboration weight for new (ω_n) of DPC. Benefiting from Weighting-Decoupling structure, weights for base or new can be different.

Dataset	weight for target domain (ω_n)					
	0	0.02	0.05	0.1	0.2	0.3
ImageNet-V2	**64.58**	64.53	64.52	64.57	64.53	64.52
ImageNet-S	**48.89**	48.83	48.83	48.77	48.72	48.61
ImageNet-A	**51.13**	51.11	51.03	50.97	50.71	50.49
ImageNet-R	**76.64**	76.56	76.47	76.36	76.25	76.29
Avg.	**60.31**	60.26	60.21	60.17	60.05	59.98
Δ	+0.00	-0.05	-0.10	-0.14	-0.26	-0.33

Table 11. Ablation study on the impact of collaboration weight for target domain (ω_n) of DPC. Benefiting from Weighting-Decoupling structure, weights for source or target can be different.

laboration weights (ω_b, ω_n) for base and new classes in DPC across 11 base-to-new tasks. For the base-class weight ω_b, we observe that: *(i)* Although the model achieves the best overall performance at $\omega_b = 0.2$, this weight value is not necessarily representative of the peak performance for individual datasets. We attribute this to the diverse data distributions across different datasets. *(ii)* When $\omega_b = 1$, implying that the entire parallel prompt $\boldsymbol{P}'$ is loaded for base class inference, the performance is still substantially better than the baseline. This corroborates that the Dynamic Hard Negative Optimizer in DPC effectually enhances the fitting of learnable prompts to the base classes.

In contrast, by observing the trend of the weight for new class ω_n, we quantitatively verify the existence of the BNT problem, i.e. the model achieves maximum performance at $\omega_n = 0$ (we add a $1e$-6 term to avoid gradient propagation errors), and as the collaboration weight increases, gradually introducing the parallel prompt optimized on the base classes to the mixed prompt $\widetilde{\boldsymbol{P}}_b$, the performance of the model declines. We also acquire similar results in the ablation study of cross-domain transfer tasks in Tab. 11. This confirms that the optimization directions for base and new classes during fine-tuning are opposite, leading to interference between them. Nevertheless, benefiting from the Weighting-Decoupling architecture of DPC, the collaboration weights are variable across different tasks. Therefore, we directly set $\omega_n = 10^{-6}$ to retain generalization of backbones on new classes, successfully avoiding BNT problem.

Special Phenomenon on MaPLe. For CoOp, PromptSRC, and PromptKD, we observe better performance at $\omega_b = 0.2$ and $\omega_n = 10^{-6}$. However, for MaPLe, we discover that DPC achieves the best results at $\omega_b = 1$, as exhibited in Tab. 9. Upon analysis, we consider that it may be due to the application of non-linear feature projection layer in MaPLe for generating visual prompts. Disrupting the linear consistency of latent feature channels between the visual and text prompt vectors (§4.4 in main text), this process leads to feature bias during the weighting of dual prompts.

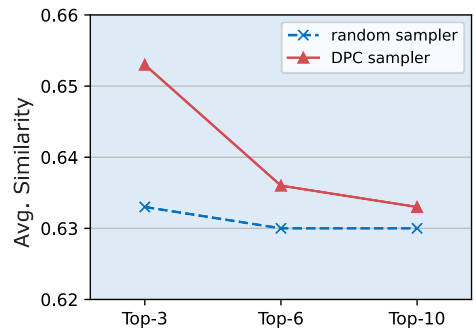

Figure 10. Cosine similarity between ground-truth and Top-K results in entire Caltech101 [6] dataset. We compare the similarity between random sampler in backbones and Negative Sampler in DPC. Higher score reveals stronger similarity.

Method	branch		HM Acc.	Δ
	Text	Image		
PromptKD		✓	83.59	
+DPC (w/o img)	✓		83.04	-0.55
+DPC (w/ img)	✓	✓	**83.91**	+0.32

Table 12. Effect of freezing visual or text branches of DPC on base-to-new tasks, utilizing PromptKD [20] as backbone model.

B.3. Quantification of Negative Sampler

In the Dynamic Hard Negative Optimizer module of DPC, the Negative Sampler is introduced for autonomously sampling hard negative objects (§3.3 in main text). To validate the effectiveness of this module, we quantify the discrepancy between the mini-batches sampled by DPC and the prompt tuning backbone using semantic similarity measurement. As demonstrated in Fig. 10, we apply a pre-trained bert-base-uncased [4] model to calculate the average cosine similarity between ground-truth objects and other samples in the mini-batch obtained using either the Negative Sampler of DPC or the random sampling strategy of the backbone. Observations indicate that the samples obtained by DPC possess higher similarity, providing effective data-level gains for the Dynamic Hard Negative Optimizer.

B.4. Effect of Visual or Text Branches on DPC

To examine the impact of the DPC optimizer on the prompts in respective modality, we conduct ablation experiments on the visual and text branches based on DPC-PK. Specifically, after obtaining the tuned visual prompts through the PromptKD backbone, we attempt to freeze them and activate only the text branch during DPC fine-tuning, then compare this with the original DPC that activates both modality branches.

We notice in Tab. 12 that freezing the visual branch results in the performance of DPC being even weaker than

	Negative Sampling	Hard Negative Optimizing	HM Acc.	Δ
(a)		Cross Entropy	74.84	
(b)	✓	Cross Entropy	75.06	+0.22
(c)	✓	DPC Contrastive	**76.13**	+1.29

Table 13. Additional ablation study on components in the Dynamic Hard Negative Optimizer. Experiments are conducted on the base-to-new tasks.

Method	Learnable Params	Memory Cost (MB)			Inference FPS
		ImgNet	Caltech	Cars	
CoOp	8K	8126	1071	1813	767.7
+DePT	(10+N/2) K	8128	1195	1906	773.2
+DPC	16K	8390	1321	2067	758.5

Table 14. Computational cost comparison between CoOp backbone, DePT [45] and our DPC. N is the quantity of base classes.

the backbone. We believe this is caused by the image-text contrastive loss of DPC, which enhances modality interaction and affects the feature channels of both branches simultaneously. Therefore, the operation that freezing single modality may lead to a deviation of text and image features. This indicates that the DPC optimizer simultaneously tunes both visual and text prompts, benefiting from the contrastive learning loss introduced by the Dynamic Hard Negative Optimizer.

B.5. Ablation on Components in DHNO

To demonstrate the necessity of each sub-module in the Dynamic Hard Negative Optimizer (DHNO) proposed in §3.3 of the main text, we conduct more ablation studies on the components of DHNO. The results are exhibited in Tab. 13. Since the Negative Sampler and Feature Filtering module are bound together in the process of reconstructing hard negatives, the Negative Sampling section in the table represents the combination of the two.

Compared with (a) CoOp backbone model, although (b) introducing only the Negative Sampler reveals a performance improvement, the gain is not distinct. We attribute this to the relatively weak effectiveness of the cross-entropy loss in the prompt learner backbones. Although the Negative Sampler effectively constructs mini-batches containing hard negatives, the standard cross-entropy loss, due to its lack of cross-modal interaction ability, fails to achieve deep alignment between visual and textual features. In contrast, significant enhancement in HM performance is observed in (c) introducing the symmetric image-text contrastive loss of DPC. The above results indicate a strong dependency among the Negative Sampler, Feature Filtering, and Hard Negative Optimizing components in DHNO. The combination of these 3 sub-modules leads to a remarkable improvement in base class performance.

Datasets		ProSRC	+DePT	TCP	+DPC
Avg. over 11 datasets	Base	83.45	84.08	84.13	**86.10**
	New	74.78	75.03	**75.36**	74.78
	H	78.87	79.29	79.51	**80.04**
ImageNet	Base	77.28	77.91	77.27	**78.48**
	New	70.72	**70.77**	69.87	70.72
	H	73.85	74.17	73.38	**74.40**
Caltech101	Base	97.93	98.37	98.23	**98.90**
	New	94.21	94.14	**94.67**	94.21
	H	96.03	96.21	96.42	**96.50**
OxfordPets	Base	95.41	94.83	94.67	**96.13**
	New	97.30	97.21	97.20	**97.30**
	H	96.34	96.00	95.92	**96.71**
StanfordCars	Base	76.34	78.26	80.80	**82.28**
	New	74.98	74.73	74.13	**74.98**
	H	75.65	76.46	77.32	**78.46**
Flowers102	Base	97.06	97.44	**97.73**	97.44
	New	73.19	74.89	**75.57**	73.19
	H	83.45	84.69	**85.23**	83.59
Food101	Base	90.83	90.61	90.57	**91.40**
	New	91.58	**91.63**	91.37	91.58
	H	91.20	91.12	90.97	**91.49**
Aircraft	Base	39.20	41.18	41.97	**46.74**
	New	35.33	**35.63**	34.43	35.33
	H	37.16	38.20	37.83	**40.24**
SUN397	Base	82.28	82.60	82.63	**83.63**
	New	78.08	**78.82**	78.20	78.08
	H	80.13	80.67	80.35	**80.76**
DTD	Base	83.45	83.64	82.77	**86.88**
	New	54.31	**59.18**	58.07	54.31
	H	65.80	**69.32**	68.25	66.84
EuroSAT	Base	92.84	94.46	91.63	**96.25**
	New	74.73	71.01	74.73	**74.73**
	H	82.80	81.07	82.32	**84.13**
UCF101	Base	85.28	85.54	87.13	**88.99**
	New	78.13	77.29	**80.77**	78.13
	H	81.55	81.20	**83.83**	83.21

Table 15. Detailed comparison between plug-and-play methods.

B.6. Computational Cost

Tab. 14 summarizes the variations of learnable parameters, GPU memory overhead and inference time efficiency (evaluated by Frames Per Second, FPS) for the CoOp backbone, as well as two plug-and-play models, DePT and our `DPC`, across 3 example datasets. Due to the dual-prompt framework of `DPC`, the amount of learnable parameters in `DPC` is doubled relative to the initial model. However, profiting from the two-step fine-tuning strategy of `DPC`, the backbone prompt and parallel prompt are activated in separate stages, meaning that the computational overhead does not significantly increase. Experiments indicate that the memory cost of `DPC` slightly raises compared with the backbone (~ 0.25 GB), which we believe is mainly due to the increased computation required for the contrastive learning loss. As a PEFT method, the computational cost of introducing `DPC` to enhance prompt learners is completely acceptable.

B.7. Detailed Comparison: Plug-and-Play

In Tab. 15, we provide a more detailed supplement to the data presented in Fig. 5 of the main text. Applying PromptSRC as the backbone model, we report the base-to-new performance of DePT and our `DPC` across 11 datasets, and introduce another plug-and-play model, TCP [43], for comparison. It is clear that `DPC` achieves superior base-class performance on most datasets, leading to the highest HM score among all baseline models.

C. Limitation and Future Work

Although our `DPC` effectively conquers the BNT problem in prompt tuning through prompt-level decoupling, we believe that this framework still has the room for optimization. Firstly, while we inherit the settings of the original backbone to obtain the tuned prompt, these configurations may not represent globally optimal points for generalization. How to adaptively acquire the top new-class performance through the backbone, thereby further leveraging the decoupled structure of `DPC`, remains a research-worthy question. Secondly, `DPC` demands learnable text prompts and image features (as well as optional visual prompts) for contrastive learning. For the research based on pure visual prompts (such as VPT [13]) or feature extraction layers (such as CLIP-Adapter [7]), it is challenging for `DPC` to integrally adapt as a plug-and-play approach.

In future work, beyond the directions outlined in Sec. 5 of the main text, we will continue to explore strategies for enhancing the performance of base and new tasks, and investigate the feasibility of matching other forms of backbone models.