Decoupled Multi-Predictor Optimization for Inference-Efficient Model Tuning

Supplementary Material

In the supplementary material, we conduct more experiments to further investigate the effectiveness of our DMPO. Specifically, we first provide a detailed explanation of the experiments shown in Fig. 4a and Fig. 4b, and conduct the same experiments on five FGVC datasets. Subsequently, we evaluate the generalization ability of DMPO using different fine-tuning modules. Additionally, we perform experiments on a larger dataset (*i.e.*, ImageNet-1K [6]). Next, we conduct ablation studies on various components of DMPO and present an analysis of the DMPO training process and overhead. Finally, we visualize the classified images at different stages.

S1. Decoupling Challenge in Early Stages

To investigate why early exiting networks cannot maintain the same high performance in both early and deep stages simultaneously, we design an experiment on CIFAR-100 [20] dataset to calculate the cosine similarity between two variants of Dyn-Adapter [47] and Original ViT. In this experiment, to better investigate the influence of early stages on deep stages, both Dyn-Adapter and its variant are configured with $S = 2$ stages, each consisting of $L = 6$ blocks.

As shown in Fig. 4a and Fig. 4b, Dyn-Adapter adopts the loss weight allocation strategy proposed in [47], with the output feature of Stage 1 denoted as $\mathbf{X}_{\text{Dyn}}$. The output feature of Original ViT in the 6-th block is denoted as $\mathbf{X}_{\text{Ori}}$, and we consider $\mathbf{X}_{\text{Ori}}$ to be the low-level representative feature required by deeper stage. Dyn-R indicates that based on Dyn-Adapter, the variation in $\mathbf{X}_{\text{Dyn}}$ is restricted using $\|\mathbf{X}_{\text{Ori}} - \mathbf{X}_{\text{Dyn}}\|_2$, resulting in a new feature representation, $\mathbf{X}_{\text{R}}$. By calculating the cosine similarity of $(\mathbf{X}_{\text{Ori}}, \mathbf{X}_{\text{Dyn}})$ and $(\mathbf{X}_{\text{Ori}}, \mathbf{X}_{\text{R}})$, we obtain the results shown in Fig. 4a. As illustrated in Fig. 4b, "Stage 1" indicates that all samples exit at Stage 1, whereas "Stage 2" indicates that all samples exit at Stage 2.

As training progresses, $\mathbf{X}_{\text{Dyn}}$, which is directly involved in classification learning, will contain more discriminative information than $\mathbf{X}_{\text{Ori}}$, while $\mathbf{X}_{\text{R}}$ will consistently maintain a fundamental representation highly similar to $\mathbf{X}_{\text{Ori}}$ due to $\|\mathbf{X}_{\text{Ori}} - \mathbf{X}_{\text{Dyn}}\|_2$. From Fig. 4a and Fig. 4b, the performance of early stage declines when providing representative features instead of discriminative features. At the same time, the performance of deep stage shows the opposite trend. This indicates a conflict where early stage struggles to learn representative and discriminative features simultaneously. In light of the preceding analysis, we introduce our DMPO method to address this conflict. Under similar experiment settings, as shown in Fig. 4a and Fig. 4b, the output features of early stages in our DMPO achieve a trade-off between fundamental and discriminative information, achieving the best results at both Stage 1 and Stage 2.

To thoroughly validate the unresolved decoupling conflict in early stages, we conduct the same experiments on five FGVC datasets. The results are presented in Fig. S1.

S2. Theoretical Analysis on Decoupled Optimization

Analogous to Eq.(3) in DSN [21], total loss of a model with multi-stage predictors $F(w)$ can be expressed as $F(w) = P(w) + Q(w)$, where $P(w)$ denotes final classification loss, and $Q(w)$ indicates sum loss of all preceding predictors. Based on Lemma 1 in DSN [21], we know $F(w)$ and $P(w)$ share the same optimal weights w^*, while $Q(w)$ may attain a different optimal w^+. For our DMPO, we require both w^* and w^+. Therefore, our two-phase optimization obtains the optimal solution w^* (representation) for $F(w)$ at first phase, and then optimize $Q(w)$ to achieve w^+ (discrimination) in second phase while keeping w^* as much as possible, constrained by α_i and $\sigma(\ell_{i-1})$. The detailed theoretical analysis will be added in revision.

S3. Performances on CNN Architecture

We experiment with the backbone of ConvNeXt-Base to validate effect of Dyn-Adapter and our DMPO on CNN by tuning on CUB-200-2011, where full inference with LoRA tuning achieves a top-1 accuracy of 88.66%. As illustrated in Tab. S1, for 70% FLOPs, DMPO clearly outperforms Dyn-Adapter. For 30% FLOPs, DMPO significantly surpasses Dyn-Adapter. These results verify effectiveness of DMPO for CNNs.

Method	$\text{Acc}_{0.3}$	$\text{Acc}_{0.7}$	$\text{Acc}_{1.0}$
Dyn-Adapter	31.83	74.65	88.76
DMPO	82.26	86.97	88.21

Table S1. ConvNeXt-Base as backbone on CUB-200-2011.

S4. Different Fine-tuning Modules

In Sec. 4, we use LoRA [16] as the fine-tuning module. To further validate the generalization ability of our DMPO, we conduct experiments using other fine-tuning modules (*i.e.*, Adapter [15] and Repadapter [28]). The results on VTAB-1K are presented in Tab. S2, while those on CIFAR-100 and five FGVC datasets are shown in Tab. S3. As shown in Tab. S2, although our DMPO exhibits a slight performance

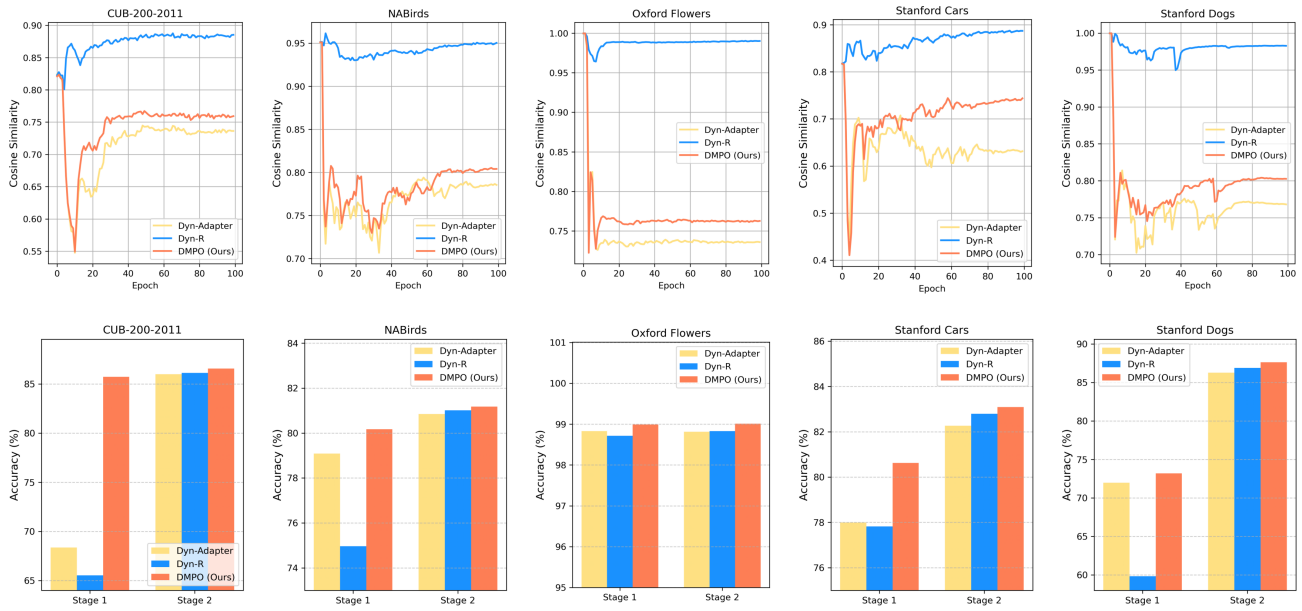


Figure S1. Comparison of cosine similarities (top row) among Dyn-Adapter, Dyn-R, DMPO, and Original ViT, and classification accuracies (bottom row) of Dyn-Adapter, Dyn-R, and DMPO at Stage 1 and Stage 2 on five FGVC datasets.

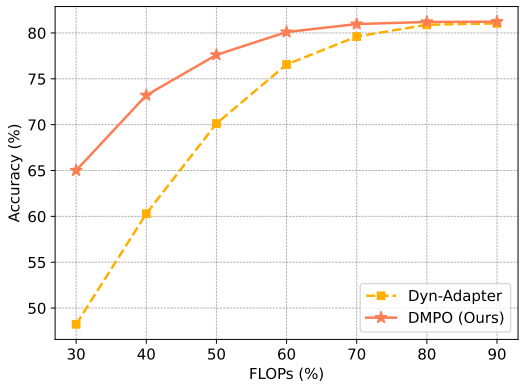


Figure S2. Results on ImageNet-1K at varying levels of inference FLOPs. Note that we use LoRA [16] as the fine-tuning module and the results are obtained from training over 40 epochs.

drop compared to Dyn-Adapter when using Repadapter as the fine-tuning module, it demonstrates better performance at low FLOPs and maintains greater stability across varying FLOPs. Furthermore, Tab. S3 indicates that when utilizing more downstream data, our DMPO consistently achieves the best performance at both low and high FLOPs.

S5. Evaluation by Utilizing ImageNet-1K as Downstream Data

The results of Dyn-Adapter and our DMPO on ImageNet-1K after 40 epochs of training are shown in Fig. S2. DMPO consistently outperforms Dyn-Adapter across all inference FLOPs levels. Specifically, at 30% FLOPs, DMPO achieves an accuracy of 63.76%, significantly surpassing Dyn-Adapter's 48.23% by a margin of 15.53%. Overall, the accuracy curve of Dyn-Adapter declines sharply as FLOPs decrease, whereas DMPO demonstrates a much smoother decline. The accuracy gap between DMPO at 30% and 70% FLOPs is attributed to the complexity of the dataset and the limited number of training epochs.

S6. Ablation Studies

To further validate the effectiveness of our DMPO, we conduct ablation studies on CIFAR-100 and five FGVC datasets. We use LoRA [16] as the fine-tuning module.

Effect of Architecture Component To further investigate the impact of different components of architecture, *i.e.*, bypass module and high-order statistics-based predictor, we conduct ablation studies on both. As shown in Tab. S4, ✓ indicates that the corresponding component is used, while ✗ indicates that it is not used. From Tab. S4, we can see that both architecture components contribute to improving performance.

Insertion Positions of High-order Statistics-based Predictor We further conduct experiments to evaluate the impact of inserting high-order statistics-based predictors at different stages. The results are presented in Tab. S5, where ✓ indicates the presence of a high-order statistics-based

	Natural							Specialized				Structured										
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Elev	Param. (M)	FLOPs (G)	Average
LoRA [16]																						
Dyn-Adapter$_{0.7}$ [47]	67.9	90.5	70.4	99.1	89.8	86.4	53.6	**86.3**	95.7	84.3	75.1	81.6	67.8	50.2	**82.1**	79.1	47.0	31.6	39.6	0.46	11.6	75.0
DMPO$_{0.7}$ (Ours)	**69.2**	**92.5**	**71.3**	**99.4**	**90.9**	**89.4**	**54.3**	85.1	**96.2**	**86.0**	**76.0**	**83.6**	**70.2**	**51.3**	80.9	**81.7**	**48.0**	**34.6**	**42.8**	0.87	11.6	**76.2**
Dyn-Adapter$_{0.3}$ [47]	66.8	90.1	69.3	99.0	89.4	84.6	53.7	81.9	95.2	84.4	**75.2**	72.4	64.9	44.0	77.0	63.4	38.4	28.2	33.6	0.46	5.2	72.0
DMPO$_{0.3}$ (Ours)	**68.6**	**92.4**	**71.2**	**99.4**	**90.2**	**89.4**	**54.3**	**85.0**	**96.2**	**86.1**	**75.2**	**83.2**	**70.2**	**51.0**	**80.8**	**81.6**	**47.6**	**34.8**	**44.1**	0.87	5.2	**76.0**
Adapter [15]																						
Dyn-Adapter$_{0.7}$ [47]	68.3	91.1	67.6	98.9	89.5	85.7	54.3	83.0	95.8	84.6	75.8	80.4	64.8	**48.5**	78.5	76.0	49.4	30.0	40.0	0.28	11.6	74.2
DMPO$_{0.7}$ (Ours)	**68.9**	**92.6**	**70.4**	**99.1**	**90.4**	**88.0**	**54.5**	**84.5**	**96.0**	**85.0**	**76.0**	**81.8**	**65.1**	48.4	**79.8**	**76.6**	**50.6**	**30.8**	**41.9**	0.78	11.6	**75.1**
Dyn-Adapter$_{0.3}$ [47]	62.9	90.8	62.9	98.4	81.0	81.0	53.6	80.0	94.8	82.1	75.4	65.4	60.9	39.2	69.2	37.7	32.4	19.7	30.9	0.28	5.2	67.8
DMPO$_{0.3}$ (Ours)	**68.8**	**92.5**	**70.2**	**99.1**	**90.1**	**88.1**	**54.2**	**84.6**	**96.0**	**85.2**	**75.5**	**81.6**	**65.6**	**48.0**	**78.3**	**78.3**	**51.3**	**31.1**	**44.6**	0.78	5.2	**75.2**
Repadapter [28]																						
Dyn-Adapter$_{0.7}$ [47]	**71.8**	92.6	**71.7**	**99.1**	**90.6**	90.8	**54.3**	**85.8**	95.9	**86.4**	**76.1**	80.3	**68.9**	49.9	**81.9**	**82.3**	**50.3**	**36.8**	41.1	0.38	11.6	**76.4**
DMPO$_{0.7}$ (Ours)	70.1	**93.4**	**71.7**	**99.1**	90.5	**91.1**	53.6	85.3	**96.1**	**86.4**	75.9	**81.8**	**68.9**	**51.0**	81.0	80.7	49.6	34.4	**43.0**	0.78	11.6	76.2
Dyn-Adapter$_{0.3}$ [47]	69.0	92.5	68.0	98.9	88.1	86.4	**53.7**	81.9	95.5	85.2	74.8	69.9	64.3	44.1	77.4	70.4	40.6	28.3	33.3	0.38	5.2	72.4
DMPO$_{0.3}$ (Ours)	**69.8**	**93.4**	**71.4**	**99.1**	**90.0**	**91.1**	53.6	**85.4**	**96.2**	**86.4**	**75.5**	**81.7**	**68.9**	**50.9**	**81.5**	**81.3**	**49.8**	**34.3**	**44.9**	0.78	5.2	**76.2**

Table S2. Results of different fine-tuning modules on VTAB-1K benchmark.

Method	Param. (M)	FLOPs (G)	CIFAR-100	CUB-200 -2011	NABirds	Oxford Flowers	Stanford Cars	Stanford Dogs	Average
LoRA [16]									
Dyn-Adapter$_{0.7}$ [47]	0.95	11.61	91.6	82.4	80.4	98.7	82.2	85.5	86.80
DMPO$_{0.7}$ (Ours)	1.73	11.62	**92.6**	**87.0**	**81.8**	**99.1**	**83.6**	**87.3**	**88.57**
Dyn-Adapter$_{0.3}$ [47]	0.95	5.17	78.5	81.8	77.0	98.8	81.0	72.9	81.67
DMPO$_{0.3}$ (Ours)	1.73	5.19	**92.3**	**86.1**	**80.8**	**99.1**	**83.7**	**86.5**	**88.08**
Adapter [15]									
Dyn-Adapter$_{0.7}$ [47]	0.81	11.63	91.1	81.5	79.8	98.5	78.2	84.2	85.54
DMPO$_{0.7}$ (Ours)	1.60	11.64	**91.6**	**84.0**	**80.1**	**98.5**	**80.0**	**86.6**	**86.80**
Dyn-Adapter$_{0.3}$ [47]	0.81	5.18	72.6	77.3	66.3	98.4	60.8	54.8	71.68
DMPO$_{0.3}$ (Ours)	1.60	5.20	**90.5**	**84.0**	**79.2**	**98.5**	**80.3**	**85.7**	**86.37**
Repadapter [28]									
Dyn-Adapter$_{0.7}$ [47]	1.67	11.61	91.7	83.1	80.7	98.0	83.3	85.2	87.02
DMPO$_{0.7}$ (Ours)	0.89	11.62	**92.3**	**86.6**	**81.3**	**99.2**	**84.1**	**87.2**	**88.44**
Dyn-Adapter$_{0.3}$ [47]	1.67	5.17	78.4	82.3	77.4	98.0	81.5	69.3	81.14
DMPO$_{0.3}$ (Ours)	0.89	5.19	**91.7**	**86.5**	**80.7**	**99.2**	**84.1**	**86.3**	**88.08**

Table S3. Results of different fine-tuning modules on CIFAR-100 and five FGVC datasets.

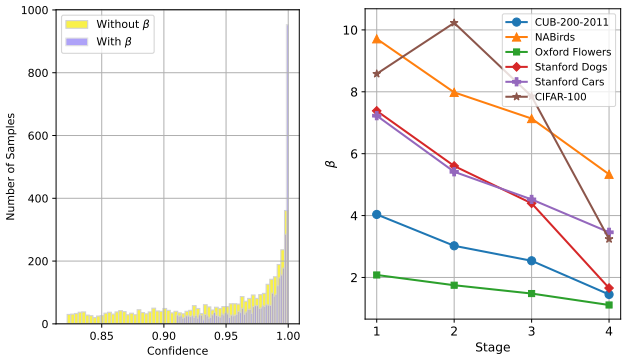
predictor at the corresponding position, while ✗ denotes its absence. As shown in Tab. S5, inserting a high-order statistics-based predictor at Stage 1 achieves better performance than at Stage 2, particularly at 30% FLOPs. This is because, at 30% FLOPs, most samples exit at Stage 1, where inserting a high-order statistics-based predictor allows to extract rich high-level discriminative features. However, at 70% FLOPs, the features at Stage 1 are more low-level and representative compared to Stage 2. Without the high-order statistics-based predictor, the original predictor disrupts these low-level representative features more significantly, negatively impacting accuracy. Furthermore, using only one high-order statistics-based predictor in early stage results in a significant accuracy gap ($> 1\%$) between 30% FLOPs and 70% FLOPs, but inserting high-order statistics-based predictors at both Stage 1 and Stage 2 effectively reduces this gap. Additionally, while inserting a high-order statistics-based predictor at Stage 3 additionally offers moderate performance improvements, the additional parameters and limited gains lead us to restrict insertions to Stages 1 and 2.

Component		Average Acc (%)	
BYP	HP	30% FLOPs	70% FLOPs
✗	✗	84.41	86.96
✓	✗	84.96	87.68
✗	✓	87.76	88.21
✓	✓	**88.08**	**88.57**

Table S4. Average results of DMPO with different architecture components. Note that gray represents our DMPO.

Stage				Param.	Average Acc (%)	
1	2	3	4	(M)	30% FLOPs	70% FLOPs
✗	✗	✗	✗	1.00	84.96	87.68
✓	✗	✗	✗	1.63	87.04	88.32
✗	✓	✗	✗	1.34	85.42	87.66
✓	✓	✗	✗	1.73	88.08	**88.57**
✓	✓	✓	✗	2.10	**88.17**	88.55

Table S5. Results of DMPO with different insertion positions of high-order statistics-based predictors. Note that gray represents our DMPO.



(a) Visualization of confidence (b) Final β values for different stages

Figure S3. (a) Confidence distribution at Stage 1 with and without β at 50% inference FLOPs on CIFAR-100. The total number of samples is 3,500. (b) Final β values across the four stages after training, with each stage's β initially setting to 1.

Sensitivity of α Regarding α, results of different settings are summarized in Tab. S6. It can be observed that α is not sensitive to different start and end values, and the default configuration is kept for different tasks.

Start	End	$Acc_{0.3}$	$Acc_{0.7}$
$[0.01, 0.01, 1.0, 2.0]$	$[1.5, 1.0, 0.1, 0.1]$	88.08	88.57
$[0.01, 0.01, 2.0, 5.0]$	$[2.0, 1.0, 0.1, 0.1]$	88.04	88.45
$[0.01, 0.01, 0.5, 1.5]$	$[2.0, 1.0, 0.1, 0.1]$	88.08	88.45
$[0.01, 0.01, 2.0, 5.0]$	$[5.0, 2.0, 0.1, 0.1]$	88.05	88.40

Table S6. Results of various α settings. Gray is default setting.

Effect of β To validate the effectiveness of β on performance, we conduct an ablation study. As shown in Tab. S7, ✗ indicates that β is not used, while ✓ denotes its use. The results demonstrate that β can further enhance overall performance by improving the confidence of early stages in classification. Besides, Fig. S3a proves that learned β effectively enhances the classification confidence of early stages. Fig. S3b illustrates that the values of learned β generally decrease with increasing stage depth across most datasets. This trend reveals that, compared to deeper stages, early stages exhibit lower classification confidence due to limited discriminative ability. Consequently, early stages tend to learn larger β values to enhance confidence.

β	Average Acc (%)	
	30% FLOPs	70% FLOPs
✗	87.08	87.76
✓	**88.08**	**88.57**

Table S7. Average results with and without β. Note that gray represents our DMPO.

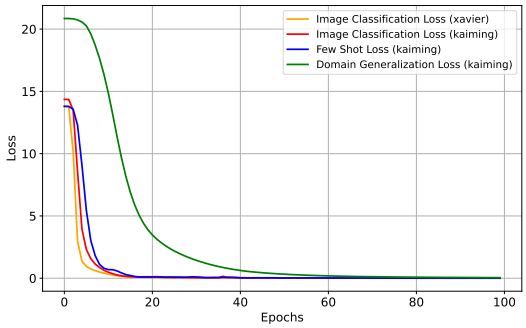


Figure S4. Training losses across tasks and initialization methods.

S7. Convergence of Decoupled Optimization

As illustrated by the loss curves in Fig. S4, our decoupled optimization algorithm demonstrates stable and rapid convergence across various tasks and parameter initialization methods.

S8. FLOPs of Different Module

For different model parts per stage, FLOPs is recorded as 0.0024G (1 Bypass), 0.048G (1 HP classifier), being much lower than 4.2G (3 ViT blocks) and leading negligible cost.

S9. Training Overhead

We evaluate the training time and memory overhead with a batch size of 32 on a single NVIDIA 3090 GPU. The

comparison of training samples per seconds is: 221 images/s (LoRA) *v.s.* 218 images (Dyn-Adapter) *v.s.* 194 images (DMPO). The memory consumption is: 5.55GB (Dyn-Adapter) *v.s.* 5.67GB (DMPO). Overall, these results indicate that DMPO introduces minimal additional training overhead.

S10. Visualization of Images at Different Stages

To intuitively demonstrate the feasibility of our DMPO, we visualize the classified images at different stages. As shown in Fig. S5, we train Dyn-Adapter [47] and our DMPO on the CUB-200-2011 dataset and select five correctly classified images from Stage 1 and Stage 4 at 70% FLOPs. We use LoRA [16] as the fine-tuning module. From Fig. S5, it can be observed that our DMPO outperforms Dyn-Adapter in correctly classifying more complex images at early stages, indirectly indicating that DMPO's early stages exhibit stronger discriminative ability compared to Dyn-Adapter.

Figure S5. Visualization of the classified images at different stages by our DMPO and Dyn-Adapter. The top row shows images correctly classified by Dyn-Adapter at Stage 1, the middle row shows images correctly classified by DMPO at Stage 1, and the bottom row presents images correctly classified by DMPO at Stage 1 but only correctly classified by Dyn-Adapter at Stage 4.