Dynamic Adapter Tuning for Long-Tailed Class-Incremental Learning Supplementary Materials

Yanan Gu[1], Muli Yang[3], Xu Yang[2], Kun Wei[2], Hongyuan Zhu[3*], Gabriel James Goenawan[3], Cheng Deng[2*]
[1]Norinco Group Testing and Research Institute, Xi'an, China
[2]School of Electronic Engineering, Xidian University, Xi'an, China
[3]Institute for Infocomm Research (I^{2}R), A*STAR, Singapore
{yanangu.xd, xuyang.xd, weikunsk, chdeng.xd}@gmail.com,
{yangml, zhuh, gabriel_james_goenawan}@i2r.a-star.edu.sg

Algorithm 1: DAT

Input: Data D^t, Threshold τ
Initialize: tasks T, Pre-trained M_0, Adapted model M_a^t with hyper-adapter H^t,Classification head P_b^t and P_{imb}^t, learnable key k^t, Adapter Cache $\mathcal{M}$
for $t \sim T$ **do**
  for $i \sim len(\mathcal{M})$ **do**
    $Acc_i \leftarrow$ [Cluster($M_a^i(D^t)$); λ]
  end
  if $len(\mathcal{M}) = 0$ *or* $max(Acc) \leq \tau$ **then**
    // Creation
    $\theta \leftarrow \mathcal{L}_{imb,b,bd,m}\{M_0, M_a^t, P_b^t, P_{imb}^t, k^t; D^t\}$
    $\mathcal{M}.append(k^t, H^t, P^t)$
  else
    // Selection
    initialize H^t with H^*
    $\theta \leftarrow$
    $\mathcal{L}_{imb,b,bd,m,td}\{M_0, M_a^t, M_a^*, P_b^t, P_{imb}^t, k^t; D^t\}$
    Replace H^* with H^t
  end
end

A. Algorithm

Algorithm 1 shows the overall training process of the proposed algorithm.

B. Dataset Setting

As depicted in Fig. A, Ordered LT-CIL considers a scenario in which all classes are arranged based on the number of samples per class and subsequently allocated into distinct tasks. Conversely, Shuffled LT-CIL assumes that the appearance of classes across different tasks is random, potentially resulting in varying degrees of imbalance in class distributions within each task.

*Corresponding authors.

Table A. Comparison of the number of parameters between our method and other methods under different settings.

Methods	Number of Parameters		AIA
	Trainable	Stored	
DualPrompt/E-Prompt=5	0.33M	0.33M	70.39
DualPrompt/E-Prompt=15	0.79M	0.79M	71.11
DualPrompt/E-Prompt=20	1.02M	1.02M	71.05
CODA-Prompt/Prompt=8	3.9M	3.9M	67.61
DAT	0.32M	0.70M	**82.77**

C. Comparison of Parameters

As shown in Tab. A, for DualPrompt, as the length of E-Prompt gradually increases, both the trainable parameters and the stored parameters increase. However, the performance improvement remains marginal. In contrast, our approach exhibits higher performance despite having fewer trainable parameters compared to DualPrompt. Moreover, upon completing incremental task learning across the 10-task scenario, the parameters stored in our method are nearly equivalent to those of DualPrompt with E-prompt 15. However, our method outperforms DualPrompt by a substantial margin, demonstrating the effectiveness of our adapter tuning. And CODA-Prompt obtains poorer performance with more parameters.

D. More Ablation Experiments

Tab. B presents detailed experimental results for different threshold settings. We explored two extremes: a threshold of 0.0, where the model consistently utilizes the initial task's hyper-adapter without creating new ones, and a threshold of 1.0, where a new hyper-adapter is generated for each task. Results indicate that as the threshold increases, more hyper-adapters are accumulated, leading to

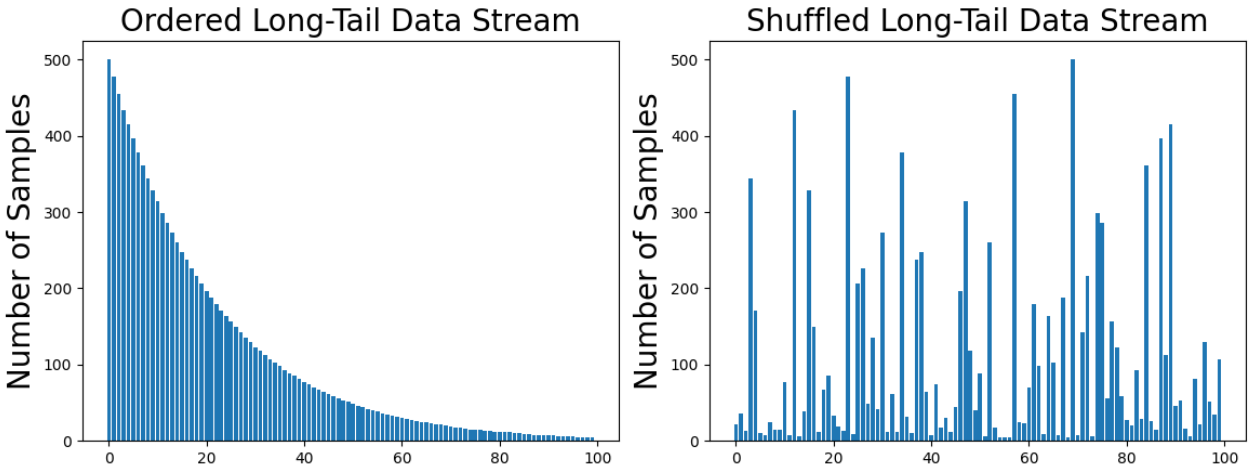

Figure A. An illustration of Order LT-CIL and Shuffled LT-CIL scenarios.

Table B. The performance implications of adjusting the threshold τ for DAT.

Methods	$\tau = 0.0$	$\tau = 0.1$	$\tau = 0.2$	$\tau = 0.3$	$\tau = 0.4$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$	$\tau = 1.0$
AIA	50.67	57.52	59.46	66.03	71.52	75.88	82.77	82.77	86.83	89.21	90.91
Added	+0	+1	+1	+2	+2	+3	+5	+5	+8	+8	+10

incremental performance enhancements. When the number of stored hyper-adapters is low, the performance gain from adding hyper-adapters is substantial. For instance, transitioning from 0 additional adapters to introducing 1 results in a 7.15% performance increase. However, when the number of hyper-adapters becomes substantial, the performance gains from adding more adapters diminish. For example, adding two additional hyper-adapters, from 8 to 10, only yields a 1.7% performance increase.

Additionally, we explored the effect of embedding hyper-adapters into pre-trained models and training them on balanced datasets. The experimental results in Tab. C demonstrate a considerable performance disparity compared to our method.

E. Hyper-Adapter

Fig. B shows the structure of the adapter and hyper-adapter. Specifically, for task t, we utilize hyper-adapter H^t to generate weights for each layer's adapter. H^t takes the layer identity as input to generate weights for the down-projection and up-projection layers in the adapter. Without loss of generality, the weight generation process of the adapter f in layer i can be formulated as follows:

$$\left[f_{\theta_i^D}, f_{\theta_i^U}\right] = H^t\left(i\right), \theta^H \in \mathcal{R}^{2\times d\times d_i}. \tag{1}$$

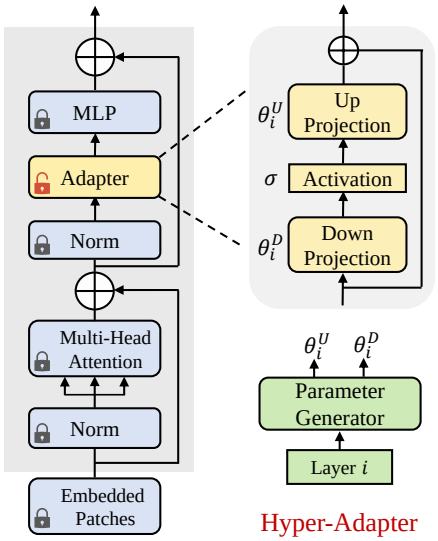

Figure B. The structure of adapter and hyper-adapter.

The hyper-adapter greatly reduces the number of parameters required to store. The number of parameters for a hyper-adapter is roughly equal to that of a single adapter, namely, $|H^t| \approx \left(\left|f_{\theta_i^D}\right| + \left|f_{\theta_i^U}\right|\right)$. $|\cdot|$ denotes the parameter count operation. That is, the hyper-adapter reduces

Table C. The performance of a pre-trained model trained with adapters on a balanced dataset.

Methods	Average Incremental Accuracy
Pre-trained + hyper-adapter on balanced	66.65
Ours	82.77

Table D. AIA of various methods on NINCO [1].

Methods	L2P	DualPrompt	DAP [3]	CODA-Prompt	DAT (Ours)
AIA	93.08	93.91	90.62	93.47	95.73

Table E. AIA of various methods with 20 tasks.

Methods	L2P	DualPrompt	CODA-Prompt	DAT (Ours)
AIA	43.56	60.77	56.23	70.36

about $(L-1)(|f_{\theta_i^D}| + |f_{\theta_i^U}|)$ parameters. L is the number of transformer layers in the pre-trained model. We use the traditional ViT [2] model which contains 12 transformer layers as the pre-trained model. Therefore, using the hyper-adapter reduces about 91.6% of the parameters compared to directly using the traditional adapter.

F. Experimental Results on NINCO

To mitigate data leakage, we conducted a 10-task shuffled-order experiment on the NINCO dataset Tab. D, where the classes do not overlap with those in the ViT pre-trained dataset ImageNet-1K. we also include the long-tail specific method DAP [3] as a comparison. As shown in Tab. D, even though the pre-trained model has never seen those classes, our method can still achieve excellent performance.

G. Results of 20 Tasks

We conducted incremental learning on CIFAR-100 with 20 tasks in a shuffled order. As shown in Tab. E, DAT demonstrates superior performance compared to others.

H. Details on Competitors

In the experiments results on CIFAR-100 and ImageNet-Subset, we reported the results of UCIR, PODNET, LWS, and GVAlign based on the values provided in the GVAlign paper. Therefore, the backbone network for UCIR is still ResNet. The results for L2P, DualPrompt, and CODA-Prompt were obtained from our experiments using their official code. For all experiments on DomainNet, the results were obtained based on the official code of these methods.

References

[1] Julian Bitterwolf, Maximilian Mueller, and Matthias Hein. In or out? fixing imagenet out-of-distribution detection evaluation. In *ICML*, 2023. 3

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[3] Chenxing Hong, Yan Jin, Zhiqi Kang, Yizhou Chen, Mengke Li, Yang Lu, and Hanzi Wang. Dynamically anchored prompting for task-imbalanced continual learning. *arXiv preprint arXiv:2404.14721*, 2024. 3