INCREMENTAL UNIFIED PARAMETER ADDITIONAL TUNING WITH BASIC MEMORY REPLAYING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class incremental learning (CIL) aims to develop an open intelligence system that can continuously learn new concepts from new tasks while retaining the knowledge to distinguish between new and old concepts. Recently, parameter-additional-tuning methods (PAT) have successfully alleviated catastrophic forgetting by starting from a well-pre-trained model and only allowing a few additional parameters to be trained. However, the contradiction between stability and plasticity and the lack of inter-task features still challenge PAT-based CIL methods. To address these, we propose unified PAT and basic memory replaying (BMR). On the one hand, unified PAT transfer the model to sequential arrival downstream tasks based on a fixed pre-trained vision transformer by unifying the prompt-based and the adapter-based methods, offering more diversified plastic structures to efficiently capture more useful features without large-scale parameters. On the other hand, BMR synthesizes on-call virtual old samples with a fixed-size basic memory to create a global task that covers up all the sub-tasks, which makes inter-task features more learnable without a large memory budget. Abundant experiments prove the effectiveness of our method.

1 INTRODUCTION

Deep learning models are specialized in individual tasks but lack the open learning ability to deal with other new tasks, which is very different from human beings. To address this problem, class incremental learning (CIL) (Rebuffi et al. (2017)) attempts to develop an open artificial intelligence system that can continuously learn new concepts from new tasks, while retaining the knowledge learned from previous tasks and the ability to distinguish all the categories seen so far. However, class incremental learning is much more difficult due to catastrophic forgetting (McCloskey & Cohen (1989)), meaning deep learning models' performance of previous tasks degrades sharply when transferred to new tasks.

Most existing CIL works would like to fine-tune the entire model over all learning phases, which inevitably results in much forgetting since the decisive parameters for earlier tasks are very likely to be overwritten. Though this forgetting can be resisted by replaying the seen data (Rebuffi et al. (2017); Liu et al. (2020); Xin et al. (2021)), the performance of these approaches is still limited. At the same time, as the number of learned categories increases, the memory overhead required for replaying will become unacceptable for practical application.

In transfer leaning, instead of training the entire backbone, parameters-additional-tuning (PAT) methods (Ding et al. (2022)) focus on tuning additional lightweight parameters to adapt one large-scale pre-trained model to downstream tasks while keeping the pre-trained backbone frozen. There are two main proven PAT methods for tuning on downstream tasks based on Vit (Dosovitskiy et al. (2021)): prompt and adapter. As shown in the left picture of Figure 1, prompt-based PAT (Gao et al. (2021)) prepends additional tunable prefix tokens to the input or hidden layer and trains these soft prompts when tuning on downstream tasks. As shown in the middle picture of Figure 1, adapter-based PAT (Houlsby et al. (2019)) inserts small neural modules called adapters to each layer of the pre-trained Vit and only the adapters are trained at tuning time.

PAT naturally achieves the balance of stability and plasticity by making full use of the stable pre-trained model and effectively transferring the additional parameters to downstream tasks, which is inherently suitable for incremental learning. Recent studies (Wang et al. (2021; 2022)) have shown

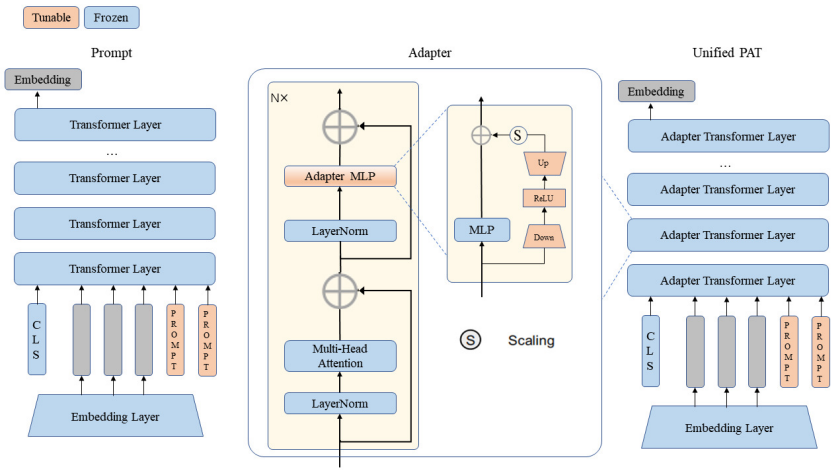

Figure 1: Illustration of the transformer architecture and state-of-the-art parameter-additional-tuning methods and UPAT. **Left**: Prompt-based methods only tune the additional prompt tokens concatenated with class embedding tokens (CLS) and image tokens from the embedding layer. **Middle**: Adapter-based methods only tune an additional small module which consists of two linear layers for down sampling and up sampling, a non-linear function Relu and a scaling factor. **Right**:UPAT combines prompt-based structures and adapter-based structures.

that prompt-based CIL methods can obtain state-of-the-art results without replaying old samples. However, only prompt tokens are not plastic enough for their too few parameters and single structures, which limits the performance of prompt-bassed PAT. Since adapter-based methods remain unproven for CIL, we introduce adapter to CIL and propose UPAT (unified parameters-additional-tuning) as shown in the right picture of Figure 1 to further enhance the plasticity of PAT-based CIL methods which enrich the structures of PAT by unifying prompt and adapter to learn more useful features. In the meantime, to maintain the stability of UPAT, a feature distillation loss term is also injected into the total learning loss which minimizes the distance between the representation output from the new learning model and the representation output from the pre-trained model. Experiments show that UPAT can effectively increase the upper limit of PAT-Based CIL with basic memory replaying that we introduce next.

Although memory is no longer a necessary element of PAT-based CIL methods to prevent forgetting, when the model sequentially learns individual subtasks without any memory, the lack of inter-task features becomes a serious problem that limits PAT methods. Instead, when memory is preserved, inter-task supervision can be easily and implicitly introduced into the global classification tasks. Therefore, we believe that memory is still necessary for PAT-based CIL to learn inter-task knowledge. The key question is how to efficiently offer memory to introduce inter-task supervision while limiting the memory budget simultaneously. Some works (Zhai et al. (2019)) try to generate old samples by a high-capacity auxiliary network. They can somehow be regarded as data-free memory replaying methods, but these generative networks suffer from high complexity in both space and time. Different from these works, we design a novel basic memory replaying (BMR) manner, which only remembers a fixed-size basic memory and plays them by a simple linear combination with trainable synthesis coefficients. BMR only relies on only one main backbone and some coefficients to produce pictures which saves a lot of space budget. BMR distributes the virtual pictures in the representation space by maximizing the similarity between their embeddings and the example prototypes that are optimized by maximum similarity loss, which strengthens the supervision for inter-task representation learning. The experimental results show that BMR can further improve UPAT by effectively offering inter-task supervision.

In summary, we make the following contributions: (1) We propose UPAT, a unified tuning network that combines previous prompt-tuning and novel adapter-tuning for CIL, which enhances the plasticity by enriching the feature structures and maintaining stability by a simple feature distillation. (2) We emphasize that data-free PAT-based CIL lacks the inter-task features, and adopt BMR to offer

inter-task supervision without a large memory budget. (3) Combining UPAT and BMR, our method significantly outperforms other state-of-the-art PAT-based methods in the CIL scenario.

2 Related Work

2.1 Class Incremental Learning

Class incremental learning (CIL) (Rebuffi et al. (2017)) is a difficult scenario of incremental leaning. It not only requires that the model should not forget the knowledge of learned tasks when learning new tasks but also does not provide task ID when predicting, that is, the new and old categories should be put together for classification. CIL is dominated by three categories of methods:replaying-based, distillation-based, and parameter-isolation-based. Replaying-based methods replay original or synthesized samples to preserve knowledge learned from old tasks (Rebuffi et al. (2017); Liu et al. (2020); Xin et al. (2021); Iscen et al. (2020); Welling (2009); PourKeshavarzi et al. (2022)). Distillation-based methods distill the current model with the previous model as the teacher to maintain stability (Li & Hoiem (2017); Hou et al. (2019); Douillard et al. (2020)). Parameter-isolation-based methods are dedicated to each task to protect the model from any possible forgetting (Mallya & Lazebnik (2018); Serra et al. (2018); Rusu et al. (2016); Aljundi et al. (2017)).

2.2 Parameter Additional Tuning

Parameter-additional-tuning (PAT) (Ding et al. (2022)) is first proposed in NLP for parameter-efficient transfer learning. It aims to transfer the large-scale pre-trained model to downstream tasks by a few tunable parameters without adjusting the pre-trained model. The most popular PAT methods can be categorized into prompt-based methods and adapter-based methods. The adapter-based methods inject tiny modules to the feed-forward networks in each transformer layer and get competitive performance on downstream tasks by only training these tiny modules (Houlsby et al. (2019); Karimi Mahabadi et al. (2021); Rücklé et al. (2021)). These prompt-based methods do not inject neural modules into the Transformer model, but instead, learn new input tokens for the Transformer backbone, and have achieved good performance in various NLP tasks compared with fine-tuning methods (Gao et al. (2021); Li & Liang (2021); Hu et al. (2022)). L2P and DualPrompt introduce the prompt-based method to the CIL field and also produce remarkable results. Instead, IPT (Deng et al. (2022)) introduces prototype tuning to CIL, which is also a kind of PAT method that only tunes additional prototypes and keeps the pre-trained model frozen. Different from the above methods, our work first introduces an adapter to CIL and unifies the above methods to enrich the PAT structures for class incremental tuning.

3 Preliminary

3.1 Problem Defintion

Given two stream of dataset $\{X^1, X^2, \cdots, X^y\}$ and $\{D_0, D_1, \cdots, D_t\}$. $X^y = \{x_1^y, x_2^y, \cdots, x_{n_y}^y\}$ consists of all samples belonging to class y. $D_t = \{X^{s_t+1}, X^{s_t+2}, \cdots, X^{s_{t+1}}\}$, where $D_i \cap D_j = \emptyset$, for any $i \neq j$ and s_t represents the total number of categories learned before phase t. The classification requirements of class incremental learning in training and testing are different. During training at phase t, D_t is the only accessible original dataset. During testing at phase t, the trained model is required to classify totally s_{t+1} seen categories.

3.2 Tuing with Example Prototypes

BMR learns new example prototypes for each new class in the same way as IPT to alleviate semantic drift. To enrich the diversity of example prototypes, for each class i, its N_e example prototypes e_i are initialized by k-means clustering from sample embeddings and optimized by maximum similarity loss (MSL) L_{MS}:

$$L_{MS}^t(x) = 1 - \max_j(< \Phi(x, \theta), e_{g(x)}^j >), \tag{1}$$

where θ means the frozen pre-trained parameters, $< \Phi(x, \theta), e^j_{g(x)} >$ means the cosine similarity between the embedding of sample x and the j-th example prototype of class $g(x)$ that sample x truly belongs to and $x \in D_t$.

Example prototypes can effectively reduce high-level forgetting (semantic drift) by constraining the category prototypes for each class with the prototypes classification loss:

$$L^t_{CP}(e^j_i) = -\sum_{k=1}^{s_{t+1}} \hat{y_k}(e^j_i) \log y_k(e^j_i), \tag{2}$$

where $i \in [1, s_{t+1}] \cap N^+$,

$$[y_k(e^j_i)]^T = softmax([< e^j_i, c_k >]^T), \tag{3}$$

$k \in [1, s_{t+1}] \cap N^+$, c_k means the k-th class's category prototypes and $\hat{y_k}(e^j_i)$ means the label of e^j_i.

Besides high-level forgetting, BMR further applies example prototypes to reduce lower-level forgetting (features forgetting) by the way described in Section 4.

4 METHODOLOGY

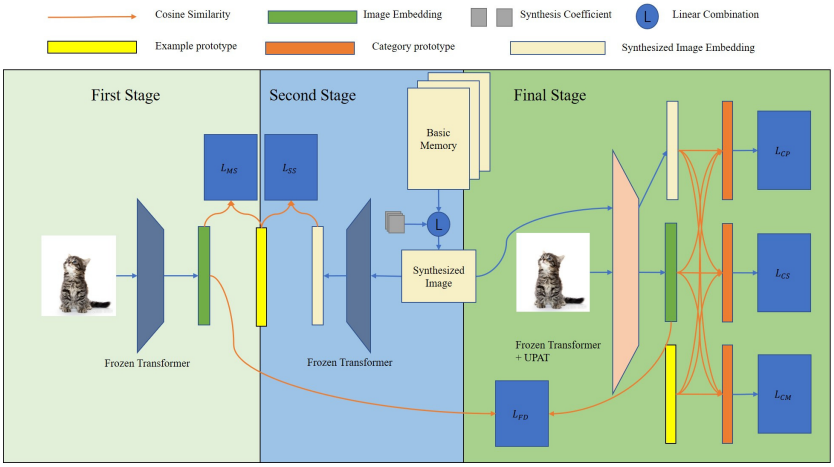

Figure 2: **Overview of the proposed framework.** There are three training stages for BMR. In the first stage, example prototypes are trained by L_{MS}. In the second stage, basic memory and synthesis coefficients are trained by L_{SS}. In the final stage, only the additional parameters and category prototypes are trainable and they are optimized by L_{FD}, L_{CP}, L_{CS}, and L_{CM}.

4.1 OVERVIEW OF THE FRAMEWORK

The framework of our method is shown in Figure 2. The entire model consists of a fixed pre-trained transformer, category prototypes for classification, example prototypes for semantic drift compensation, additional parameters for efficient tuning, and the basic memory with each example's synthesis coefficients for on-call replaying. On the whole, the framework loops with three sequential training stages. In the first training stage, only new example prototypes are optimized by L_{MS} which minimizes the cosine distance between example prototypes and image embeddings produced by the pre-trained transformer. In the second training stage, basic memory and each example's synthesis coefficients are trained by L_{SS} which minimizes the cosine distance between synthesized image embeddings and target example prototypes. In the final training stage, virtual old samples are synthesized by the BMR. Virtual samples and new task samples are separately sent to the network. The additional parameters and category prototypes are trained by the classification loss of virtual samples, new task samples, and prototypes samples as well as the feature distillation loss term.

4.2 TUNING WITH UNIFIED PAT STRUCTURES AND FEATURE DISTILLATION

PAT is a promising paradigm to make a good trade-off between stability and plasticity for incremental learning. For plasticity, PAT has been proved that can achieve similar performance to fine-tuning when it transfer pre-trained Vit to downstream tasks. For stability, PAT works with fewer additional parameters without tuning the entire model. In this section, to further improve the plasticity and stability of PAT for incremental learning, we introduce adapter to CIL and unify prompt-based structures and adapter-based structures to make the backbone more plastic. We also introduce a distillation loss to stabilize the representation.

Unified PAT structures. As shown in Figure 1, we combine two common structures of additional parameters for Vit including prompt-based and adapter-based. Let the input tokens to the i-th transformer layer be $x_i \in R^{LD}$. Original Transformer layer (Layer) is denoted by:

$$x_{i+1} = Layer(x_i), \tag{4}$$

$$Layer(x_i) = MLP(LN(h_i(x_i))) + h_i(x_i), \tag{5}$$

$$h_i(x_i) = MHSA(LN(x_i)) + x_i. \tag{6}$$

MHSA and LN respectively represent multi-head-self-attention and layer normalization. We first concatenate extra trainable tokens pt with image tokens x_1' and class embedding ct as the input to the first transformer layer in the same way as prompt-based methods:

$$x_1 = Concatenate(x_1', pt, ct), \tag{7}$$

$$where\ x_1' \in R^{(HW)D}, pt \in R^{PD}, ct \in R^{1\times D}. \tag{8}$$

Then, we modify the Equation 5 for each transformer layer in the same way as adapter-based methods:

$$TL(x_i) = MLP(LN(h_i(x_i))) + Adapter(LN(h_i(x_i))) + h_i(x_i) \tag{9}$$

The Adapter module is defined by followed formulation:

$$Adapter(x) = Linear_{up}(Relu(Linear_{down}(x))) \times scaling \tag{10}$$

where $Linear_{up}$ is an up-sample linear layer and $Linear_{down}$ is a down-sample linear layer.

Distilling from the pre-trained model. Adding the extra parameters to the pre-trained backbone risks the instability of the representation. To address this, feature distillation loss is introduced by distilling from the pre-trained model:

$$L_{FD}(x) = 1 - < \Phi(x, \theta), \Phi(x, \theta, \gamma) >, \tag{11}$$

where γ represents all the extra parameters including adapters and prompt tokens and θ means the frozen pre-trained parameters.

4.3 TUNING WITH BASIC MEMORY REPLAYING

As shown in the experiment results in Section 5.2, although unified PAT with distillation from the pre-trained model can significantly improve the performance, it still performs mediocrely for smaller split CIL settings compared to the IPT baseline. We believe that is caused by the lack of inter-task features. Since the model only learns new knowledge from the sequential arriving individual subtasks, it is difficult to get more useful features to distinguish inter-task concepts. The key to addressing this problem is to construct inter-task supervision. Replaying with buffer memory that consists of original samples is an effective method to construct inter-task supervision. However naive memory replaying (NMR) requires at least one example for each class, bringing a large uncontrollable memory budget, which may not meet the requirements of practical application (Naive memory replaying means selecting several original samples for each class from old data, the selective method is to choose the closest samples to the example prototypes). Therefore, we propose a novel replaying strategy basic memory replaying (BMR) to introduce inter-task supervision. BMR only stores a small, fixed-size basic memory and replays the same number of synthesized samples as example prototypes for each class in an on-call manner by a simple linear combination with synthesis coefficients. It is worth noting that synthesis coefficients do not need to be stored after eacn phase's training while basic memory will not grow with the increase of the number of categories. The size

of synthesis coefficients is $s_{t+1} \times N_e \times N_b$ at phase t and the total size parameters trained in the second training stage is $N_b \times C \times H \times W + s_{t+1} \times N_e \times N_b$, which is much smaller than the size of total synthesized images $s_{t+1} \times N_e \times C \times H \times W$ when $s_(t+1)$ increases to a large number. This help saves the memory budget and speeds up the second training stage.

Basic memory is the set of N_b basic samples $B_m \in R^{CHW}$ that is the same shape as the original sample $x \in R^{CHW}$ inputted to the network. It synthesizes virtual samples $v_k^j \in R^{CHW}$ by a linear combination with synthesis coefficients $s_k^{jm} \in R$:

$$v_k^j = \sum_{m=1}^{N_b} s_c^{jm} B_m, \tag{12}$$

where v_k^j means the j-th virtual samples of the k-th class, $k \in [1, s_{t+1}] \cap N^+$ and $j \in [1, N_e] \cap N^+$. In the second stage, BMR attempts to optimize basic memory and synthesis coefficients by the following synthesis similarity loss:

$$L_{SS}(e_k^j) = 1 - < \Phi(v_k^j, \theta), e_k^j > . \tag{13}$$

L_{SS} aims to maximize the similarity between synthesized samples and example prototypes in the representation space. To learn useful inter-task features, the distribution of the synthesized samples neads to be similar to the real distribution of original samples. Since example prototypes have been diversified by the k-means initialization and maximum similarity loss (MSL) to cover the distribution of original samples in the representation space, L_{SS} can ensure the representativeness of the virtual samples for simulating the real distribution. Note that the θ here is frozen.

Tuning with on-call basic memory replaying. After basic memory and synthesis coefficients of each prototype are trained into a steady state, BMR uses Equation 12 to generate $N_e \times S_{t+1}$ virtual samples v_k^j for each class to construct a super-task that covers all the class seen so far at phase t. In the final training stage, the total tuning loss is formulated:

$$L_{tuning} = \lambda_1 L_{CS} + \lambda_2 L_{CP} + \lambda_3 L_{CV} + \lambda_4 L_{FD}, \tag{14}$$

where L_{CS} is the classification loss of new samples, L_{CV} is the classification loss of synthesized virtual samples, and $\lambda_p, p \in [1, 4] \cap N^+$ are the adjustable hyperparameters to balance sub-loss items. At each phase t, we formulate L_{CS}^t and L_{CV}^t as:

$$L_{CS}^t(x) = -\sum_{i=s_t+1}^{s_{t+1}} \hat{y}_i(x) \log y_i(x), \tag{15}$$

$$L_{CV}^t(v_k^j) = -\sum_{i=1}^{s_{t+1}} \hat{y}_i(v_k^j) \log y_i(v_k^j), \tag{16}$$

$$where\ [y_k(x)]^T = softmax([< \Phi(x, \theta, \gamma), c_k >]^T), \tag{17}$$

$x \in D_t$, $k \in [1, s_{t+1}] \cap N^+$ and $j \in [1, N_e] \cap N^+$.

5 EXPERIMENTS

5.1 EXPERIMENTS SETUP

Datasets. We perform our experiments on CIFAR-100 (Krizhevsky et al. (2009)), Stanford-Cars (Krause et al. (2013)) and Oxford-Flower(Nilsback & Zisserman (2008)). CIFAR-100 consists of 60000 32×32 color images that belong to 100 classes. There are 500 training images and 100 testing images per class. The Stanford Cars dataset consists of 196 classes of cars with a total of 16,185 images that are divided into 8,144 training images and 8,041 testing images which are in the shape of 360×240. Oxford-Flower is an image classification dataset consisting of 102 flower categories. Each class consists of between 40 and 258 images.

Benchmark protocol. We evaluate our method under different phase settings. For CIFAR-100, we evaluate the proposed methods under 5, 10, and 25 phases settings. For Stanford-Car, the settings are 7, 14, and 49 phases. For Oxford-Flowers, the settings are 6 and 17 phases. Because PAT-based

CIL methods do not train the entire model, the fine-tuning upper bound is not suitable for PAT-based methods. So we adopt the 1 phase setting as the upper bound for all the PAT-based methods.

Metrics. We report the two standard metrics to measure the quality of CIL: final accuracy (Final Acc) and the average of forgetting (AF). Supposed the accuracy of task i at phase t is A_t^i and the accuracy of all the classes that have been learned at phase t is $A_t^{1:t}$. Final accuracy is just defined as the accuracy at the last phase T: A_T. Average forgetting is to estimate the forgetting of previous tasks which is defined as:

$$\frac{1}{T}\sum_{i=1}^{T}(A_i^i - A_T^i) \tag{18}$$

Implementation details. We implement our method in PyTorch with RTX 3090 GPUs. All the results are based on VitB/16 or VitL/14 pre-trained in CLIP (Radford et al. (2021)). SGD is used for optimization with a base learning rate of 0.01. All the gradients are clipped into the range of 5 to 100 to accelerate training. The model is trained for 20 epochs in the first stage, 10 epochs in the second stage, and 10 epochs in the final stage on CIFAR-100. On Stanford-Cars, the numbers of epochs at sequential stages are 40, 20, and 20. On Oxford-Flowers, the numbers are also 40, 20, 20. Our code will be open source after review.

5.2 Ablation Study

Table 1: Main ablation results (final accuracy) on CIFAR-100, Stanford-Cars, and Oxford-Flowers. We compare different memory management methods (Mem Management) including BMR, NMR(naive memory replaying), and no memory (NO). We report the results of NMR that use 1 and 10 samples per class. Results are reported under different phase settings. Note that we regard the result of the 1 phase setting as the different upper bounds of different tuning structures. "Fixed" structure means keeping the backbone frozen with no additional parameters. All results are based on the same VitB/16 pre-trained on CLIP400M by CLIP with 10 example prototypes.

Dataset	Metric	Tuning Structure Mem Management	Fixed No	UPAT No	UPAT BMR	UPAT NMR	UPAT NMR
CIFAR-100	Final Acc	phase=1	81.52	88.66	88.6	88.67	88.64
		phase=5	77.67	80.37	81.77	80.72	82.61
		phase=10	76.13	77.98	80.06	78.99	81.56
		phase=25	76.11	76.46	78.01	77.14	80.88
	buffer size		0	0	10	100	1000
Stanford-Cars	Final Acc	phase=1	85.94	87.84	86.81	87.78	87.82
		phase=7	81.18	81.74	82.26	82.88	83.74
		phase=14	79.99	81.67	82.12	82.72	83.65
		phase=49	79.62	79.92	81.80	80.34	83.45
	buffer size		0	0	10	198	1980
Oxford-Flowers	Final Acc	phase=1	97.35	97.72	97.63	97.69	97.74
		phase=6	92.80	93.54	94.44	94.75	95.89
		phase=17	93.01	93.30	95.53	94.50	95.85
	buffer size		0	0	10	102	1020

Main Ablation The proposed method is comprised of three components: example prototypes, UPAT(unified-parameter-additional-tuning), and BMR(basic memory replaying). Here we analyze the effect of these components. Note that we regard the result of the 1 phase setting as the different upper bounds of different tuning structures. From the results of Table 1 and and Figure 3, we make the following observations: (1) UPAT significantly improves the IPT baseline (fixed backbone + example prototypes) under larger split setting such as 5 phases and 10 phases settings of CIFAR-100 by capturing more useful features. (2) UPAT performs mediocrely compared with the IPT baseline under small task setting such as 25 phases settings of CIFAR-100 and 49 phases settings of Stanford Car because the positive impact of the new features learned from small tasks is not enough to cover up the negative impact of forgetting. (3) BMR successfully constructs inter-task supervision for UPAT to learn more inter-task features while reducing intra-task forgetting, which is reflected by the improvement of final accuracy and higher representation separability as shown in Figure 3.

(4) Compared with NMR, BMR achieves competitive performance with a fixed-size buffer memory and is even better than the NMR uses about ten times bigger buffer size.

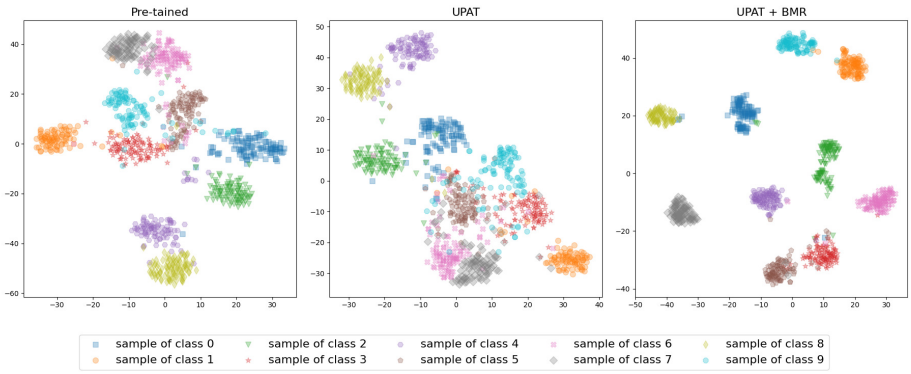

Figure 3: T-SNE(Van der Maaten & Hinton (2008)) of 10 classes that belong to different phases on CIFAR-100. Compared to pre-trained and UPTA, BMR significantly improves the inter-phase performance of UPAT.

Table 2: Ablation results of UPAT on CIFAR-100, Stanford-Cars, and Oxford-Flowers. We study the prompt structure and adapter structure's effect for UPAT as well as feature distillation (FD). We study their effect with BMR and no memory (No). All results are based on the same VitB/16 pre-trained on CLIP400M by CLIP with 10 example prototypes.

	Phase			10	14	17
	Dataset			CIFAR-100	Stanford-Cars	Oxford-Flowers
Memory Management	Prompt	Adapter	FD	Final Acc	Final Acc	Final Acc
No	✓	✗	✓	77.02	81.52	93.59
No	✗	✓	✓	77.54	79.97	93.51
No	✓	✓	✗	77.61	25.79	90.76
No	✓	✓	✓	77.98	79.92	93.30
BMR	✓	✓	✗	**80.13**	66.53	95.09
BMR	✓	✗	✓	76.09	80.76	94.45
BMR	✗	✓	✓	78.59	81.84	95.52
BMR	✓	✓	✓	80.06	**82.12**	**95.53**

Ablation of UPAT. UPAT consists of prompt tokens, adapters, and feature distillation (FD). Here we study the effect of these elements. From the results in Table 2, we make the following conlusions: (1) Prompt-tuning has more single structures and fewer parameters than adapter-tuning, which brings more stability but less plasticity. When there is no memory used, the positive impact from the strong stability of prompt-tuning covers up the negative impact from the weak plasticity, so prompt-tuning may even be better than adapter-tuning or UPAT such as the results of 14 phases setting on Stanford Car. However, when some memory makes up for the weaker stability of the adapter and UPAT, prompt-tuning will be no longer better than the adapter and UPAT. Therefore, the upper limit of prompt-tuning is lower than that of the adapter and UPAT. (2) Feature distillation effectively improves performance when there is no memory used. Feature distillation also helps the training stability of UPAT. There may be problems with training stability without feature distillation, taking the results of 14 phases on Stanford-Cars for example. (3) UPAT is consistently better than prompt and adapter on different datasets with BMR memory management. UPAT offering richer structures to strengthen the plasticity for incremental representation learning. (4) For the low plasticity of the structure of prompt-tuning, the synthetic sample brought by BMR is more like a noise than effective inter-task supervision to some extent, which reduces the performance of prompt-tuning.

Table 3: Comparison with several state-of-the-art PAT-based CIL methods on CIFAR-100 and Stanford-Cars. We report the results with BMR or no memory (No) on CIFAR-100. Note that we regard the result of the 1 phase setting as the different upper bounds of different PAT methods. All results are based on the same VitB/16 pre-trained on CLIP400M by CLIP with 10 example prototypes. "AF" means average forgetting.

dataset	Method	Memory	Buffer size	phase=1	5		10		25	
				Final Acc	Final Acc	AF	Final Acc	AF	Final Acc	AF
CIFAR-100	L2P	No	0	84.52	75.58	0.79	70.07	0.70	65.17	0.51
	DualPrompt	No	0	85.19	77.02	0.26	76.40	0.37	75.08	0.45
	Fixed-ITP	No	0	81.52	77.67	0.23	76.13	0.27	76.11	**0.01**
	UPAT	No	0	**88.66**	80.37	0.69	77.98	0.51	76.46	0.23
	L2P	BMR	10	84.35	75.64	2.60	75.90	1.37	74.15	0.69
	DualPrompt	BMR	10	85.03	79.46	0.95	76.78	0.34	75.43	0.29
	Fixed-ITP	BMR	10	81.32	77.87	**0.21**	76.21	**0.24**	76.09	**0.01**
	UPAT	BMR	10	88.60	**81.77**	0.93	**80.06**	0.59	**78.01**	0.60
	Method	Memory	Buffer size	phase=1	7		14		49	
				Final Acc	Final Acc	AF	Final Acc	AF	Final Acc	AF
Stanford-Cars	L2P	BMR	10	86.71	79.95	0.61	80.26	0.60	77.91	0.40
	DualPrompt	BMR	10	86.41	81.13	0.30	81.32	0.38	80.78	0.11
	Fixed-ITP	BMR	10	85.78	81.20	**0.04**	80.01	**0.19**	79.78	**0.03**
	UPAT	BMR	10	**86.81**	**82.26**	0.22	**82.12**	0.46	**81.80**	0.35

5.3 Comparative Study

Comparison Approaches. We compare our method UPAT with the recent state-of-the-art PAT-based methods including L2P (Wang et al. (2021)), DualPrompt (Wang et al. (2022)) and IPT (Deng et al. (2022)) with no memory as well as BMR memory management. For fair comparison, all the results are based on the same VitB/16 pre-trained on CLIP400M (Radford et al. (2021)) by CLIP.

Comparative rsults. Results are shown in Table 3. For the metric final accuracy, UPAT significantly outperforms all other methods under different phase settings with it's higher upper bound that comes from richer tuning structures while Fixed-ITP has the lowest average forgetting because it only tunes the prototypes and remains representation unchanged. In additional, we can see that UPAT can benefit more from BMR than other methods, which is also related to the rich structures of UPAT.

5.4 Why Training basic memory from Example Prototypes

Example prototypes are initialized by k-means clustering and optimized by maximum similarity loss (MSL), which makes them widely distributed in the representation space. Training basic memory from example prototypes helps to avoid the homogenization of synthetic samples. Here we compare two methods: training basic memory from the example prototypes (ours) or by the way MRP (PourKeshavarzi et al. (2022)). Table 4 shows that training basic memory from example prototypes is much better than the way proposed in MRP.

Table 4: Final accuracy of CIFAR-100 under 5 phase setting and 10 phase setting.

	phase=5	10	25
Ours	81.77	80.06	78.01
MRP	81.04	79.02	76.73

6 Conclusion

In this work, we propose UPAT and BMR. UPAT first introduces adapter-based methods and unifies prompt-based structures and adapter structures, which help UPAT to get higher plasticity potential for CIL. Meanwhile, the stability of UPAT is also guaranteed by feature distillation. BMR effectively constructs inter-task supervision to promote UPAT learning better inter-task features by synthesizing virtual samples via a simple linear combination of trainable fixed-size basic memory and small-size synthesis coefficients. Our method UPAT-BMR achieves a better balance of stability and plasticity with limited acceptable overhead and outperforms other state-of-the-art PAT-based approaches under different settings as demonstrated by extensive experiments.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

Jieren Deng, Jianhua Hu, Haojian Zhang, and Yunkuan Wang. Incremental prototype tuning for class incremental learning, 2022. URL https://arxiv.org/abs/2204.03410.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models, 2022. URL https://arxiv.org/abs/2203.06904.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pp. 86–102. Springer, 2020.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3816–3830, 2021.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2225–2240, 2022.

Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *European Conference on Computer Vision*, pp. 699–715. Springer, 2020.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1022–1035. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/081be9fdff07f3bc808f935906ef70c0-Paper.pdf.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, 2020.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Mozhgan PourKeshavarzi, Guoying Zhao, and Mohammad Sabokrou. Looking back on learned experiences for class/task incremental learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RxplU3vmBx.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *EMNLP (1)*, 2021.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning, 2021. URL https://arxiv.org/abs/2112.08654.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *European Conference on Computer Vision*, 2022.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121–1128, 2009.

Xiaomeng Xin, Yiran Zhong, Yunzhong Hou, Jinjun Wang, and Liang Zheng. Memory-free generative replay for class-incremental learning. *arXiv preprint arXiv:2109.00328*, 2021.

Mengyao Zhai, Lei Chen, Fred Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation, 2019. URL https://arxiv.org/abs/1907.10107.