# Few-Shot Class Incremental Learning with Attention-Aware Self-Adaptive Prompt

Chenxi Liu, Zhenyi Wang, Tianyi Xiong, Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang

University of Maryland, College Park, MD 20742, USA
{cxliu539, heng}@umd.edu

**Abstract.** Few-Shot Class-Incremental Learning (FSCIL) models aim to incrementally learn new classes with scarce samples while preserving knowledge of old ones. Existing FSCIL methods usually fine-tune the entire backbone, leading to overfitting and hindering the potential to learn new classes. On the other hand, recent prompt-based CIL approaches alleviate forgetting by training prompts with sufficient data in each task. In this work, we propose a novel framework named **A**ttention-aware **S**elf-adaptive **P**rompt (`ASP`). `ASP` encourages task-invariant prompts to capture shared knowledge by reducing specific information from the attention aspect. Additionally, self-adaptive task-specific prompts in `ASP` provide specific information and transfer knowledge from old classes to new classes with an *Information Bottleneck* learning objective. In summary, `ASP` prevents overfitting on base task and does not require enormous data in few-shot incremental tasks. Extensive experiments on three benchmark datasets validate that `ASP` consistently outperforms state-of-the-art FSCIL and prompt-based CIL methods in terms of both learning new classes and mitigating forgetting. Source code is available at `https://github.com/DawnLIU35/FSCIL-ASP`.

## 1 Introduction

As the world keeps changing over time, the data in the world also continually changes. Thus, there is a need for machine learning models to follow the data change and continually learn new classes while preserving the knowledge learned from previous data, which is called as Class-Incremental Learning (CIL) [26, 42, 51, 56]. The main challenge of CIL is the catastrophic forgetting problem [11, 52]: a model forgets the previous knowledge after training on new tasks, as the data from old tasks are not fully available due to reasons like limited storage space or privacy issues [8, 45]. While many CIL methods assume a model can continually train on new class data with sufficient samples [12, 22, 24], this assumption does not hold in many real-world applications. For instance, in the scenario of an intelligent medical decision system tracking physiological signals, new patients with limited data must be learned without discarding knowledge from existing patient data [38]. This task of continually learning new classes with limited data is called Few-Shot Class-Incremental Learning (FSCIL) [39, 41]. FSCIL typically

involves training a base model using a set of base classes with sufficient samples, subsequently utilizing the knowledge learned from base classes to facilitate the incremental learning of new classes with limited samples. Beyond the challenge of catastrophic forgetting, FSCIL also triggers overfitting on limited training samples, making it harder for a machine model to learn new classes.

Various works [41] have been proposed to address the FSCIL scenario. Some of them focus on enhancing base models' ability to generalize across newly encountered few-shot classes [6, 33, 63], while others aim to find a better strategy to incrementally train on new tasks with limited data [5, 9, 20, 50]. However, most existing works fine-tune all the parameters in the base model, which leads to overfitting on base classes and hinders the transferability to new classes. On the other hand, recent prompt-based CIL methods [34, 53, 54] leverage the inherent generalization ability of pre-trained Vision Transformer (ViT) [10] by fixing the backbone parameters and only training a few new parameters called prompts [21,55]. They usually learn task-specific prompts via a key-query mechanism and store the knowledge of seen tasks in a specialized prompt pool. In this way, they preserve the knowledge of old tasks without necessitating a rehearsal buffer to store old data samples. Nonetheless, to train the task-specific prompts, prompt-based approaches require sufficient data samples from new tasks, which are not available in few-shot incremental tasks.

In this paper, we propose a novel Attention-Aware Self-Adaptive Prompt (`ASP`) framework to overcome the shortcomings of existing FSCIL and prompt-based CIL methods under the FSCIL setting. Aiming to facilitate the continual learning of new classes with limited data, `ASP` leverages the inherent generalization capability of the pre-trained ViT and the knowledge learned from sufficient base classes. Specifically, `ASP` fixes the ViT backbone and introduces prompts between attention blocks for adapting to FSCIL tasks, where prompts are decomposed into attention-aware task-invariant prompts (`TIP`) and self-adaptive task-specific prompts (`TSP`). The attention blocks pay the same attention to each `TIP` regardless of the task, encouraging `TIP` to only contain task-invariant information that can be universally used for both base classes and new classes. Unlike the previous key-query mechanics, `ASP` uses a prompt encoder to convert input images to prompt features. Inspired by the *Information Bottleneck* (IB) theory [2], `ASP` guides the prompt encoder to generate prompt features that have a strong correlation with semantic information and a weak correlation with extraneous information contained in images. To further improve the generalization ability, `ASP` aggregates prompt features over the training set to prevent overfitting on a single input image. For a specific input image, the corresponding `TSP` is composed of the average prompt features and its own prompt features. Consequently, `ASP` avoids fine-tuning the entire backbone to alleviate overfitting and circumvents the need of sufficient data to train new `TSP` for new classes. Lastly, to further enhance model discrimination, a similarity-based loss is used to cluster feature vectors to their class centers, where class centers are estimated by the anchor samples during training.

Overall, **our contributions** can be summarized as follows:

- We propose ASP, an innovative prompt-based methodology to address both the overfitting challenge in existing FSCIL methods and the data-hungry drawback in existing prompt-based methods under the FSCIL scenario.
- We design attention-aware TIP and self-adaptive TSP to transfer knowledge from base classes to new classes and alleviate forgetting on old classes.
- Extensive experiments on three benchmark datasets demonstrate that ASP substantially outperforms SOTA FSCIL and prompt-based CIL methods in both learning new classes and preserving performance on old classes.

## 2 Related Work

### 2.1 Class-Incremental Learning

**Non-prompt-based approaches:** In general, there are three different settings of incremental learning: task-, domain, and class-incremental learning(TIL, DIL, and CIL) [42]. Among all, CIL is considered to be the most challenging scenario [41], which is required to learn new classes without forgetting old ones. In current CIL works, there are three main directions. The most effective one is rehearsal methods [4, 30, 31, 48, 49], which build a rehearsal buffer to store samples from previous tasks. The second direction is to find out important parameters for the current task and prevent them from changing over incremental tasks [3, 18, 58]. In addition, a lot of works utilize knowledge distillation to preserve the knowledge of previous tasks to overcome forgetting [14,22,30]. Recently, some rehearsal-free methods [25,53] have caught people's attention, as rehearsal samples are not always allowed to be stored in real-world scenarios [54]. Notably, ASP also doesn't need a rehearsal buffer to store any data samples. Typically, CIL methods require enough training data in each incremental task to learn new classes, which is not available under the FSCIL scenario.

**Prompt-based approaches:** Prompt-based methods [21,55] are first proposed for Natural Language Processing tasks, which can better utilize the pre-trained knowledge for downstream tasks. The basic idea is fixing the backbone parameters and only fine-tuning a few new parameters (prompts) prepend to input text or images [16]. Recently, prompt-based CIL approaches using ViT backbones achieve significant performance in both learning new classes and preventing catastrophic forgetting. L2P [54] first proposes to use a key-query mechanism to select task-specific prompts from a prompt pool. DualP [53] adds task-invariant prompts to capture the shared information among tasks. But their task-invariant prompts still provide much task-specific information and their task-specific prompts require sufficient training data in incremental tasks. This would result in significant overfitting to new few-shot incremental tasks because of the limited availability of new task data, ultimately causing a decline in performance for FSCIL. Subsequently, CodaP [34] proposes to train the prompt pool and selection mechanism in an end-to-end manner. The most recent HideP [46] decomposes CIL into hierarchical components and optimizes them respectively. However, all existing prompt-based CIL methods are not suitable under the FSCIL scenario, as they all need sufficient data samples in incremental tasks to

capture task-specific knowledge and store them in prompts. Reversely, ASP does not need to train new task-specific prompts for new tasks and thus works well in few-shot incremental tasks.

### 2.2 Few-Shot Class-Incremental Learning

Few-Shot Class-Incremental Learning (FSCIL) is even more challenging than normal CIL, as it needs to incrementally learn new classes with limited labeled data [41]. TOPIC [40] first proposed the FSCIL task, which is required to first obtain a base model trained on sufficient base classes, then incrementally learn some new tasks with limited new class data. Current FSCIL methods can be roughly divided into two groups. The first group aims to utilize base classes to train a generalized backbone that can be transferred to few-shot incremental tasks [6,33,63]. The trained backbone is usually kept frozen during the following few-shot incremental tasks [28, 36, 59]. The second group focuses on the strategy of incrementally learning few-shot new classes without overfitting [5, 9, 20]. These FSCIL methods usually fine-tune all parameters in the backbone when training on base classes, which leads to overfitting on base classes and hinders the transferability to new classes. In contrast, ASP fixes the pre-trained backbone and stores task-invariant knowledge in prompts to overcome this issue. On the other hand, some recent studies [7, 15, 57] utilize OpenAI CLIP [29] to handle multi-modal inputs, requiring alignment of features between image and text. Conversely, ASP employs a ViT backbone with single-modal inputs, simplifying the training process and enhancing the model's focus on visual features alone.

## 3 Preliminaries

**Few-shot Class-incremental Learning** aims to learn a sequence of tasks $t$ using their respective data $\mathcal{D}_0, ..., \mathcal{D}_T$. When learning on task $t$, data from previous tasks $0, ..., t-1$ are totally unavailable or only partially available, and the model is required to perform well on all seen tasks $0, ..., t$. The training data in task $t$ is represented as $\mathcal{D}_t = \{(\boldsymbol{x}_{t,i}, y_{t,i})\}_{i=1}^{N_t}$, where $N_t = |\mathcal{D}_t|$ denotes the size of $\mathcal{D}_t$, $\boldsymbol{x}_{t,i} \in \mathcal{X}_t$ and $y_{t,i} \in \mathcal{Y}_t$ represent the sample and label, respectively. The training label spaces between different tasks are disjoint, i.e., for any task $t, t' \in [0, T]$ and $t \neq t'$, $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \varnothing$. The first task has sufficient training data $\mathcal{D}_0$ and is called the base task, while the following incremental tasks can be denoted as $N$-way $K$-shot classification tasks, i.e., $N$ classes for each task and $K$ samples for each class. A FSCIL model can be decoupled into a backbone $f_\theta$ parameterized by $\theta$, and a linear classifier $h_\psi$ parameterized by $\psi$. For an input test data $\boldsymbol{x}$ drawn from all seen tasks, the model tries to predict $y = h_\psi(f_\theta(\boldsymbol{x}))$ which matches the class label.

**Prompt-based Approaches** for vision tasks typically use a pre-trained vision transformer (ViT) [10] as the backbone $f_\theta$, and the parameter $\theta$ are typically frozen during training to maintain the generalization ability obtained from the pre-training. A ViT model contains multiple multi-head self-attention (MSA)

layers, and we denote the input of the $l^{th}$ MSA layer as $\boldsymbol{h}^l \in \mathbb{R}^{L_{\boldsymbol{h}} \times D}$, then the output of this layer is given as:

$$\text{MSA}(\boldsymbol{h}^l) = \text{Concat}(h_1^l, ..., h_m^l)W^O, \text{ where } h_i^l = \text{Attention}(\boldsymbol{h}^l W_i^Q, \boldsymbol{h}^l W_i^K, \boldsymbol{h}^l W_i^V) \quad (1)$$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

where $W^O, W_i^Q, W_i^K, W_i^V$ are projection matrices, $m$ is the number of attention heads and $d_k$ is a scaling factor. Among prompt-based approaches, Prompt Tuning (ProT) [16,21] is one of the most commonly used techniques which introduce a few trainable parameters $\boldsymbol{p}^l \in \mathbb{R}^{L_{\boldsymbol{p}} \times D}$ as prompts for the $l^{th}$ layer, and these prompts are prepend to $\boldsymbol{h}^l$:

$$f_{\text{ProT}}(\boldsymbol{p}^l, \boldsymbol{h}^l) = \text{MSA}([\boldsymbol{p}^l; \boldsymbol{h}^l]) \quad (3)$$

where $[\cdot\,;\cdot]$ denotes the concatenation operation along the dimension of sequence length. Before the first layer of ViT, an input image is first split as a few patches and transformed into a sequence-like representation $\boldsymbol{x}^e \in \mathbb{R}^{L_{\boldsymbol{x}} \times D}$. For image classification tasks [16], a class token $\boldsymbol{cls} \in \mathbb{R}^{1 \times D}$ is prepend and the visual prompts are prepend to form the input of ViT blocks:

$$\boldsymbol{x}^p = [\boldsymbol{cls}; \boldsymbol{p}^0; \boldsymbol{x}^e] \quad (4)$$

**Prototypical Network** [35] is a widely used approach for few-shot learning problems. It calculates the mean features $c_k$ of a class $k$ and uses it as the class prototype, i.e., $\boldsymbol{c}_k = \frac{1}{N_k} \sum_{y_i=k} f(\boldsymbol{x}_i)$; where $N_k$ is the number of samples in class $k$, and the output feature of the $\boldsymbol{cls}$ token is used as the image embedding. For a classification task with $K$ classes, $W = [\boldsymbol{c}_0, \boldsymbol{c}_1, ..., \boldsymbol{c}_K]$ is used as the linear classifier, and an input sample is classified via the softmax probability with class prototypes: $P(y = k|\boldsymbol{x}) \propto \boldsymbol{c}_k^T f_{\theta,\boldsymbol{p}}(\boldsymbol{x})$. Following works [47, 61], new class prototypes are append to $W$ to perform classification over all seen classes.

## 4 Methodology

A base model with good generalization ability is beneficial for adapting to few-shot new classes [36, 61]. To prevent overfitting on base classes after sufficient training, and to leverage the generalization ability of pre-trained ViT for learning new classes with limited data, `ASP` fixes the pre-trained backbone and learns prompts that can transfer the knowledge learned from base classes to new classes. Inspired by DualP [53], we decompose the prompts into attention-aware task-invariant prompts and self-adaptive task-specific prompts. In Sec. 4.1, `ASP` maintains consistent attention across all task-invariant prompts for any given task, thereby containing minimal task-specific information. In Sec. 4.2, `ASP` employs a prompt encoder to map an input image to `TSP` leveraging the *Information Bottleneck* theory, which has been proven to enhance generalization ability [23]. Thus,

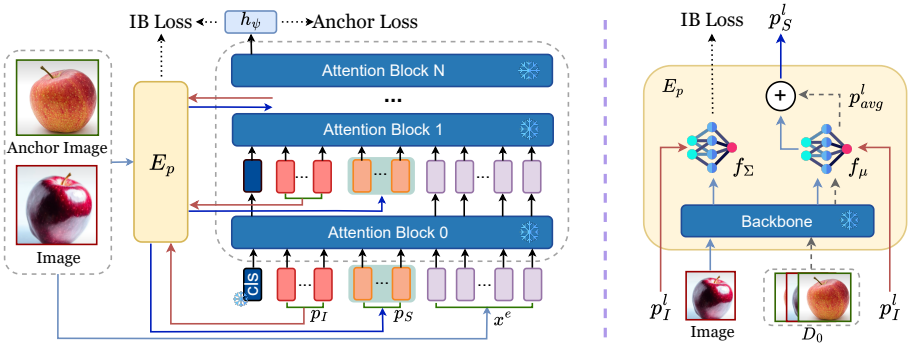


**Fig. 1:** Overall training scheme of ASP for the base task. For incremental tasks where $t > 0$, ASP updates only $\boldsymbol{p}_{avg}$ using Eq. (11). **Left:** The pre-trained backbone remains frozen during training and the prompts are inserted into layers between attention blocks. The TIP $\boldsymbol{p}_I$ are initialized from an attention-aware aspect, while the TSP $\boldsymbol{p}_S$ are derived from the prompt encoder $E_{\boldsymbol{p}}$. At the beginning of each training epoch, anchor images are selected using Eq. (16). Throughout the training, the prompts and $E_{\boldsymbol{p}}$ are optimized using IB loss and Anchor loss as specified in Eq. (17). **Right:** Details of the prompt encoder $E_{\boldsymbol{p}}$. Image features extracted by the frozen pre-trained backbone are fed to two tiny networks $f_\mu$ and $f_\Sigma$. At the start of each training epoch, $p_{avg}$ is calculated via Eq. (10) using $\boldsymbol{p}_I$ and all data in the base task. Within a training epoch, the output of $f_\Sigma$ contributes to IB loss, while $\boldsymbol{p}_S$ results from blending $p_{avg}$ and the output of $f_\mu$, as outlined in Eq. (9).

the prompt encoder can also be used for new class data without further training. Finally, Sec. 4.3 introduces an Anchor Loss, which enlarges class margins by pulling class features toward their class centers, thereby further improving model discrimination ability. ASP only trains the model on the base task using sufficient data from base classes. Subsequently, the prompts are updated using Eq. (11) for few-shot incremental tasks. Overall, our training scheme is shown in Fig. 1.

### 4.1 Attention-Aware Task-Invariant Prompts

Similar to DualP [53], task-invariant prompts are fixed after training on base classes and used for subsequent few-shot incremental tasks. Despite DualP employing the same task-invariant prompts for all tasks, it is not true that they convey identical information across different tasks due to the difference of attention on each prompt token. Consequently, these task-invariant prompts continue to offer task-specific information. To diminish the task-specific information stored in prompts, ASP promotes consistent attention weight on each prompt token. The attention on different tokens can be measured by the attention matrix $A = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$. Denote the $i^{th}$ token in layer $\boldsymbol{h}^l$ as $\boldsymbol{t}_i \in \mathbb{R}^{1\times D}$, the attention from the $i^{th}$ token to $j^{th}$ token is:

$$A_{ij} = \frac{exp(\boldsymbol{t}_i W^Q \cdot \boldsymbol{t}_j W^K)}{\sum_{m=1}^{1+L_{\boldsymbol{p}}+L_x} exp(\boldsymbol{t}_i W^Q \cdot \boldsymbol{t}_m W^K)} \tag{5}$$

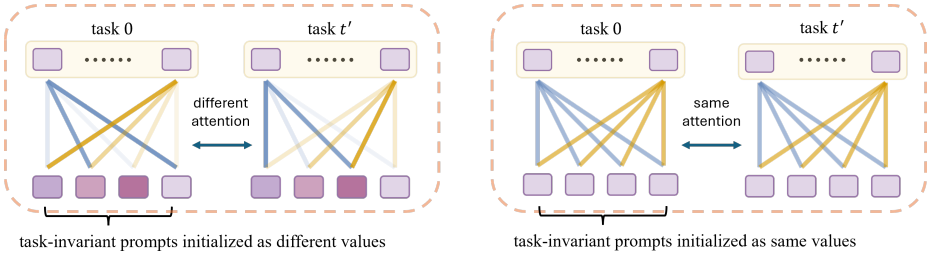


**Fig. 2:** Attention on task-invariant prompts between different tasks. A deeper line color indicates greater attention. **Left:** When initialized with different values, attention on each prompt differs across tasks, thereby providing inconsistent information. **Right:** `ASP` initializes `TIP` with the same values, ensuring consistent attention across tasks and providing uniform information.

The attention is conditioned on the value of the prompt token, and the same attention can be guaranteed if two prompt tokens have the same values. The simplest way is initializing each prompt token with the same value before training, and the values will keep the sample during the model update using gradient decent algorithms [32]. In the base class training task, we use prompts $\boldsymbol{p}_I^l \in \mathbb{R}^{L_{\boldsymbol{p}_I} \times D}$ where each token is initialized using the same values as the task-invariant prompt. The comparison between our attention-aware `TIP` and widely used randomly initialized prompt is shown in Fig. 2. As $\boldsymbol{p}_I^l$ always contribute the same to the model output regardless of task and image class, it is encouraged to contain knowledge shared among all base classes. This class-invariant knowledge can also be used for new classes in incremental tasks, thus it is also task-invariant knowledge. After training on base classes, the `TIP` are fixed during few-shot incremental tasks.

### 4.2 Self-Adaptive Task-Specific Prompts

However, relying solely on `TIP` may lead to underfitting, as it only considers the shared attributes across different tasks and ignores the unique attributes. To incorporate task-specific information into prompts, prior studies [53, 54] introduce a key-query mechanism to generate task-specific prompts based on input images. However, these methods require a large amount of training data to capture the task-specific information and store it in prompts. In contrast, `ASP` utilizes a compact neural network as a prompt encoder $E_{\boldsymbol{p}}$ to convert input images to task-specific prompts. This prompt encoder is initially trained with sufficient base class data to acquire encoding capabilities. To ensure these capabilities are extendable to new classes, `ASP` enhances the generalization ability of the prompt encoder by adopting principles from the *Information Bottleneck* theory [2].

**Prompt Learning Objective:** Inspired by IB theory [2], we propose the following learning objective for prompt learning:

$$\mathcal{L}_{IB} = I(\mathcal{P}; \mathcal{X}) - \gamma I(\mathcal{P}; \mathcal{Y}) \tag{6}$$

We utilize the random variable $\mathcal{P}$ to represent the latent prompt associated with input $\mathcal{X}$. The mutual information between the latent prompt $\mathcal{P}$ and data labels $\mathcal{Y}$ is denoted as $I(\mathcal{P};\mathcal{Y})$, while $I(\mathcal{P};\mathcal{X})$ denotes the mutual information between latent prompt $\mathcal{P}$ and input data $\mathcal{X}$. Here, $\gamma$ is a constant. In Eq. (6), the term $I(\mathcal{P};\mathcal{X}) - \gamma I(\mathcal{P};\mathcal{Y})$ constitutes the *Information Bottleneck* (IB) loss. Maximizing the mutual information between $\mathcal{P}$ and $\mathcal{Y}$ aims to strengthen the correlation between them, thereby capturing semantic knowledge. Conversely, minimizing the mutual information between $\mathcal{X}$ and $\mathcal{P}$ aims to mitigate the influence of extraneous information from input $\mathcal{X}$ on the prompt $\mathcal{P}$, thereby enhancing generalization. However, computing the mutual information is generally intractable since it often necessitates integration over the joint distribution of the involved variables. This integration poses computational or analytical challenges. Variational inference offers a framework for approximating these intractable integrals. We formulate the variational lower bound as follows:

$$\begin{aligned} I(\mathcal{P};\mathcal{Y}) - \eta I(\mathcal{P};\mathcal{X}) \geq & \int P(\boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{p}|\boldsymbol{x}) \log P(\boldsymbol{y}|\boldsymbol{p}) d\boldsymbol{x} d\boldsymbol{y} d\boldsymbol{p} \\ & - \eta \int P(\boldsymbol{x})P(\boldsymbol{p}|\boldsymbol{x}) \log \frac{P(\boldsymbol{p}|\boldsymbol{x})}{r(\boldsymbol{p})} d\boldsymbol{x} d\boldsymbol{p} \end{aligned} \tag{7}$$

Here, $r(\boldsymbol{p})$ serves as a variational marginal approximation of the intractable marginal $P(\boldsymbol{p})$, and it is chosen as $r(\boldsymbol{p}) = \mathcal{N}(0, I)$. The detailed derivation steps are provided in the Appendix.

We now delve into the calculation of Eq. (7). The term $P(\boldsymbol{y}, \boldsymbol{x}) = P(\boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x})$ is approximated by the empirical data distribution $P(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{N}\sum_{n=1}^{N} \delta_{\boldsymbol{x}_n}(\boldsymbol{x})\delta_{y_n}(\boldsymbol{y})$, where $\delta_{\boldsymbol{x}_n}$ and $\delta_{y_n}$ represent the Dirac delta function. Assuming the prompt encoder $E_{\boldsymbol{p}}$ follows the form: $P(\boldsymbol{p}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{p}|f_\mu(\boldsymbol{x}), f_\Sigma(\boldsymbol{x}))$ where two fully connected networks $f_\mu$ and $f_\Sigma$ outputs the $K$-dimensional mean $\mu$ of $\boldsymbol{p}$ and the $K \times K$ covariance matrix $\Sigma$. Then, employing the reparameterization trick [1,17], we rewrite $P(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{p}$ as $P(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$. By calculating the Kullback-Leibler (KL) divergence between $P(\boldsymbol{p}|\boldsymbol{x})$ and $r(\boldsymbol{p})$, and combining all components, we obtain the empirical *Information Bottleneck* loss function as described in Eq. (8), which we aim to minimize.

$$\mathcal{L}_{IB} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}[-\log P(y|(\boldsymbol{x}, \boldsymbol{\epsilon}))] + \mathbb{KL}(P(\boldsymbol{p}|\boldsymbol{x})|r(\boldsymbol{p})) \tag{8}$$

We use the prompt encoder $E_{\boldsymbol{p}}$ to convert an image to the corresponding `TSP`. Inspired by previous works [53, 54], $E_{\boldsymbol{p}}$ first uses the pre-trained backbone $f_\theta$ to extract the embedding features of input images and then fed to $f_\mu$ to obtain prompt features. Aiming to transfer the knowledge learned from base classes to new classes, the prompt encoder also takes `TIP` as input to generate `TSP`. To further improve the generalization ability of our `TSP` for new classes, we consider all prompt features obtained from $\mathcal{X}_0$ to provide general information together with the prompt features of an input image to construct the corresponding task-specific prompts $\boldsymbol{p}_S$. In this way, $\boldsymbol{p}_S$ can avoid overfitting to a single input image and become easier to generalize to unseen classes. During the base class training stage, the task-specific prompts for layer $l$ are given as:

$$\boldsymbol{p}_S^l = \alpha \boldsymbol{p}_{avg}^l + (1-\alpha) f_\mu([\boldsymbol{p}_I^l; f_\theta(\boldsymbol{x}, \boldsymbol{\epsilon})]) \tag{9}$$

$$\boldsymbol{p}_{avg}^{l} = \frac{1}{N_0} \sum_{N_0} f_{\mu}([\boldsymbol{p}_I^l; f_{\theta}(\boldsymbol{x}_0)]) \tag{10}$$

where $\alpha$ is a hyperparameter. $\boldsymbol{p}_{avg}^{l}$ is the average prompt features of data in the base task, which is recalculated at the beginning of each training epoch. For incremental task $t$, information of new classes is incorporated into $\boldsymbol{p}_{avg}^{l}$ using Exponential Moving Average (EMA) [60]:

$$\boldsymbol{p}_{avg}^{l} = \beta \boldsymbol{p}_{avg}^{l} + (1-\beta)\frac{1}{N_t} \sum_{N_t} f_{\mu}([\boldsymbol{p}_I^l; f_{\theta}(\boldsymbol{x}_t)]) \tag{11}$$

where $\beta$ is a hyperparameter to control the adapting speed. To balance the influence of task-invariant and task-specific prompts, their prompt length is set as the same, i.e. $L_{\boldsymbol{p}_S} = L_{\boldsymbol{p}_I}$. Finally, the prompts inserted into layer $l$ is:

$$\boldsymbol{p}^{l} = [\boldsymbol{p}_I^l; \boldsymbol{p}_S^l] \tag{12}$$

### 4.3 Anchor Loss

During base classes training, we aim to obtain a feature extractor that can (1) maximize the distance between inter-class feature embeddings, and (2) minimize the distance between intra-class feature embeddings. Furthermore, we also want to obtain a classifier head $h_{\psi}$ that can accurately classify these features into class predictions. Following [28,47], we use a fully connected layer without bias term as the classifier head $h_{\psi}$, and the prediction is obtained by measuring the cosine similarity between feature embeddings and weights $W$ in the classifier head:

$$y_k = \frac{W_k^T f_{\theta,\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{\epsilon})}{||W_k|| \cdot ||f_{\theta,\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{\epsilon})||} \tag{13}$$

where $||\cdot||$ is $l_2$ normalization. The weight $W_k$ can be seen as the prototype of class $k$ [24,28]. To separate class features and prototypes, the Cross-Entropy loss is used in Eq. (8):

$$\mathcal{L}_{IB} = -\frac{1}{N} \sum_{N} log \frac{exp(y_k)}{\sum_{k' \in K} exp(y_{k'})} + \mathbb{KL}(P(\boldsymbol{p}|\boldsymbol{x})|r(\boldsymbol{p})) \tag{14}$$

The Cross-Entropy loss simultaneously pulls class features $f_{\theta,\boldsymbol{p}}(\boldsymbol{x}_k)$ to its class prototype $W_k$, and pushes far away from other class prototypes $W_{i \neq k}$. However, only the pull action is accurate while the push action may lead the class prototype to diverge from the class mean and cause misclassification. Similar to previous works [47, 62], we replace $W$ with class mean $\boldsymbol{c}$ after training on base classes. Therefore, it is necessary to align class features with their class mean, which can also reserve places for new classes to improve new class accuracy [36, 61]. For any input sample, we maximize the cosine similarity between its features and the corresponding class mean:

$$\mathcal{L}_c = 1 - \frac{\boldsymbol{c}_k^T f_{\theta,\boldsymbol{p}}(\boldsymbol{x}_k)}{||\boldsymbol{c}_k|| \cdot ||f_{\theta,\boldsymbol{p}}(\boldsymbol{x}_k)||} \tag{15}$$

As the class mean keeps changing during the training process, it is resource-consuming to compute the accurate class mean after each mini-batch. Thus, we use an anchor sample to estimate the class mean. At the beginning of each epoch, the accurate class mean is computed, and the sample that has the largest similarity with the class mean is selected as the anchor sample for that class:

$$\hat{\boldsymbol{x}}_k = \arg\max_{\boldsymbol{x}\in\mathcal{X}_k} \frac{\boldsymbol{c}_k^T f_{\theta,\boldsymbol{p}}(\boldsymbol{x})}{||\boldsymbol{c}_k|| \cdot ||f_{\theta,\boldsymbol{p}}(\boldsymbol{x})||} \tag{16}$$

Then the estimated class mean for $L_c$ is $\hat{\boldsymbol{c}}_k = f_{\theta,\boldsymbol{p}}(\hat{\boldsymbol{x}}_k)$. The training loss is:

$$\mathcal{L} = \mathcal{L}_{IB} + \lambda \mathcal{L}_c \tag{17}$$

where $\lambda$ is a hyperparameter.

## 5 Experiments

In this section, we first introduce the experiment details of FSCIL, including datasets, evaluation protocol, training details, and baseline methods. Subsequently, we compare `ASP` with baselines on three benchmark datasets, which show the effectiveness of `ASP`. In addition, the ablation study verifies the effectiveness of different components in `ASP`. Lastly, we provide more experimental results for further analysis. We will release our code after the paper decision.

### 5.1 Implementation Details

**Datasets:** Following FSCIL works [47, 61] and prompt-based CIL works [37, 46, 53], we evaluate the performance on CIFAR100 [19], CUB200-2011 [43], and ImageNet-R [13] dataset. Similar to prior studies [47, 61], the datasets are split to form the FSCIL tasks. In detail, CIFAR100 is divided into 60 base classes and 40 new classes. The new classes are further divided into eight 5-way 5-shot incremental tasks. CUB200 and ImageNet-R are divided into 100 classes for the base task, and the left 100 classes are divided into ten 10-way 5-shot incremental tasks.

**Evaluation protocol:** Following previous works [6,39,61], we denote the Top-1 accuracy on all seen tasks $0, ..., t$ after the $t$-th task as $A_t$. The average accuracy $A_{avg} = \frac{\sum_{t=0}^{T} A_t}{T+1}$ measures the overall performance during all incremental tasks. And the forgetting phenomenon is measured by the performance dropping rate (PD), i.e., $PD = A_0 - A_T$, where 0 stands for the base task and $T$ stands for the last task. Furthermore, Harmonic Accuracy (HAcc) [28] is used to reflect the balanced performance across both base and new classes after task $T$: $A_h = \frac{2\times A_o \times A_n}{A_o + A_n}$, where $A_o$ is the accuracy of base class in task 0 and $A_n$ is the average accuracy of all classes in tasks $t > 0$.

**Training details:** All experiments are conducted with PyTorch [27] on NVIDIA RTX A6000. We follow works [34, 62] to choose ViT-B/16-1K [10], which pre-trained on ImageNet1K as the backbone $f_\theta$. For all datasets, the input images

are resized to $224 \times 224$ and trained 20 epochs using SGD optimizer. The learning rate is set as 0.01 for CIFAR100 and CUB200 while 0.03 is used for ImageNet-R. We use batch size of 48 for CIFAR100 and batch size of 24 for both CUB200 and ImageNet-R. The prompt token length $L_{p_g} = L_{p_d}$ of 10 is used for ImageNet-R, and 3 for CIFAR100 and CUB200. All experiments are run using three random seeds and the average results are reported.

**Baselines:** We first compare two widely used CIL methods iCaRL [30] and Foster [44]. Besides, we also compare the SOTA FSCIL algorithms: CEC [59], FACT [61] and TEEN [47]. Lastly, we compare the recent SOTA prompt-based CIL approaches: L2P [54], DualP [53], and CodaP [34].

### 5.2 Benchmark Comparisons

In this section, we report the performance of baselines and `ASP` under FSCIL setting. For CIFAR100 and ImageNet-R, the detailed accuracy in each task and the three evaluation metrics are shown in Tab. 1 and Tab. 2. The detailed performance of CUB200 can be found in the appendix. Furthermore, the performance curves of the Top-1 accuracy $A_t$ in each incremental task are shown in Fig. 3.

Based on the experimental results on CIFAR100, CUB200 and ImageNet-R, `ASP` achieves the best Top-1 accuracy $A_T$ of 86.7%, 83.5% and 69.7%, surpassing the second best by 2.7%, 2.9% and 7.2% respectively. Furthermore, `ASP` usually performs the best on $A_t$ before the last task, except the first task. In the first task, classical CIL and FSCIL methods fine-tune all the parameters using sufficient data, thus it is reasonable that they perform better than the prompt-based CIL approaches. Besides, `ASP` performs the best on the average accuracy $A_{avg}$, which achieves 89.0%, 83.8% and 75.3% and surpasses the second best by 1.7%, 0.7% and 9.2% on three benchmark datasets respectively. In addition, `ASP` achieves the lowest PD on CIFAR100 and CUB200 while achieving the second lowest on ImageNet-R. For the HAcc metric, `ASP` also achieves the best of 85.3%, 83.4% and 67.0% on three benchmark datasets, which can demonstrate the superiority of `ASP` in incrementally learning new classes while preserving performance on base classes at the same time.

Interestingly, we find that prompt-based CIL approaches learn nearly nothing about the new classes based on the HAcc metric. We think the main reason may be the severe overfitting of few-shot data in the classifier head, which is a fully connected layer. Thus, we replace the original classifier head with the class mean to form a prototypical network, which is denoted as L2P+, DualP+, and CodaP+. The HAcc results in Tabs. 1 and 2 show that this modification largely improves their ability to learn new tasks under the FSCIL setting.

### 5.3 Ablation Study

We conduct ablation study on three benchmark datasets, and the implementation details are the same as the setting in Sec. 5.1. To verify the effectiveness of each component in `ASP`, we alternatively remove each component from the framework and measure the average accuracy $A_{avg}$. The results are shown

**Table 1:** Detailed Top-1 accuracy $A_t$ in each incremental task, average accuracy $A_{avg}$, performance dropping rate (PD) and Harmonic Accuracy (HAcc) on CIFAR100 dataset. ↑ means higher is better, and ↓ means lower is better.

| Method | Accuracy $A_t$ in each task (%) ↑ | | | | | | | | | $A_{avg}$ ↑ | PD↓ | HAcc↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | | |
| iCaRL | **94.2** | 88.9 | 84.7 | 80.0 | 74.9 | 75.6 | 71.8 | 68.2 | 67.1 | 78.4 | 27.2 | 57.5 |
| Foster | **94.2** | 88.3 | 81.6 | 77.0 | 72.8 | 67.9 | 64.4 | 60.9 | 58.3 | 73.9 | 35.9 | 11.0 |
| CEC | 91.6 | 88.1 | 85.3 | 81.7 | 80.2 | 78.0 | 76.5 | 74.8 | 72.6 | 81.0 | 19.0 | 64.1 |
| FACT | 91.0 | 87.2 | 83.5 | 79.7 | 77.2 | 74.8 | 73.1 | 71.6 | 69.4 | 78.6 | 21.7 | 55.5 |
| TEEN | 92.9 | 90.2 | 88.4 | 86.8 | 86.4 | 86.0 | 85.8 | 85.1 | 84.0 | 87.3 | 8.8 | 81.2 |
| L2P | 92.2 | 85.2 | 79.2 | 73.8 | 69.2 | 65.1 | 61.4 | 58.1 | 55.2 | 71.1 | 37.0 | 0.0 |
| DualP | 91.8 | 84.7 | 78.6 | 73.3 | 68.7 | 64.6 | 61.1 | 57.8 | 54.9 | 70.6 | 36.9 | 0.1 |
| CodaP | 93.4 | 86.2 | 80.1 | 74.7 | 70.1 | 66.0 | 62.3 | 59.0 | 56.0 | 72.0 | 37.4 | 0.0 |
| L2P+ | 84.7 | 82.3 | 80.1 | 77.5 | 77.0 | 76.0 | 75.6 | 74.1 | 72.3 | 77.7 | 12.4 | 68.0 |
| DualP+ | 86.0 | 83.6 | 82.9 | 80.2 | 80.6 | 80.2 | 80.5 | 79.0 | 77.4 | 81.1 | 8.5 | 75.3 |
| CodaP+ | 86.0 | 83.6 | 81.6 | 79.2 | 79.1 | 78.5 | 78.3 | 77.0 | 75.4 | 79.9 | 10.6 | 72.2 |
| **Ours** | 92.2 | **90.7** | **90.0** | **88.7** | **88.7** | **88.2** | **88.2** | **87.8** | **86.7** | **89.0** | **5.5** | **85.3** |

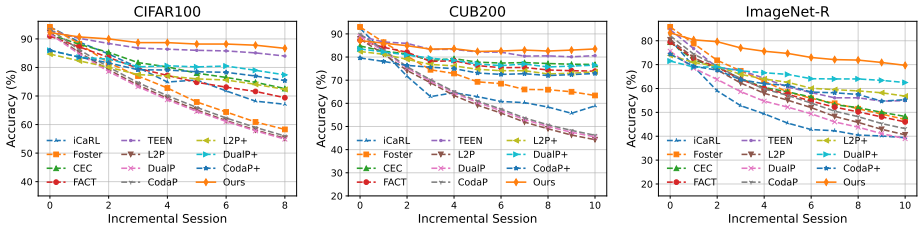


**Fig. 3:** Detailed Top-1 accuracy $A_t$ in each incremental task on three benchmark datasets. `ASP` outperforms baselines in most tasks.

in Tab. 3, where *ours w/o TIP* refers to removing the attention-aware task-invariant prompt $p_I$ from our framework, *ours w/o TSP* refers to removing the self-adaptive task-specific prompt $p_S$ from our framework, *ours w/o* $\mathcal{L}_c$ refers to removing the anchor loss $\mathcal{L}_c$ from our framework. Finally, *ours w/ Diff TIP* means we use a Gaussian distribution to initialize different values for each task-invariant prompt. The results show that removing any component leads to a performance drop in the average accuracy $A_{avg}$, which demonstrates the effectiveness of our design.

### 5.4 Further Analysis

**Hyper-Parameter:** We conduct sensitive analysis of $\alpha$, $\beta$, $\lambda$ and prompt length $L_g = L_d$. The same experimental setting of 10-way 5-shot is used on the ImageNet-R dataset and the results are shown in Fig. 4. To achieve the highest $A_{avg}$, we

**Table 2:** Detailed Top-1 accuracy $A_t$ in each incremental task, average accuracy $A_{avg}$, performance dropping rate (PD) and Harmonic Accuracy (HAcc) on ImageNet-R dataset. ↑ means higher is better, and ↓ means lower is better.

| Method | Accuracy $A_t$ in each task (%) ↑ | | | | | | | | | | | $A_{avg}$ ↑ | PD↓ | HAcc↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | |
| iCaRL | 80.6 | 69.2 | 59.0 | 52.8 | 49.4 | 45.5 | 42.8 | 42.3 | 40.5 | 40.1 | 39.4 | 51.0 | 41.2 | 36.1 |
| Foster | **85.8** | 78.8 | 71.8 | 67.4 | 63.1 | 58.5 | 55.9 | 53.8 | 51.5 | 49.3 | 47.0 | 62.1 | 38.7 | 36.8 |
| CEC | 79.4 | 71.9 | 69.0 | 64.1 | 60.4 | 58.6 | 56.4 | 53.2 | 52.0 | 50.0 | 48.3 | 60.3 | 31.1 | 32.6 |
| FACT | 79.4 | 72.5 | 69.0 | 63.8 | 60.1 | 57.6 | 54.7 | 52.2 | 50.2 | 48.1 | 46.0 | 59.4 | 33.4 | 22.3 |
| TEEN | 84.6 | 76.7 | 68.8 | 67.6 | 64.3 | 60.6 | 58.3 | 56.1 | 56.1 | 54.7 | 54.9 | 63.9 | 29.7 | 45.4 |
| L2P | 80.4 | 73.0 | 67.8 | 62.3 | 58.0 | 55.1 | 52.0 | 48.3 | 45.8 | 42.9 | 40.6 | 56.9 | 39.8 | 1.0 |
| DualP | 75.6 | 68.5 | 63.8 | 58.7 | 54.7 | 52.2 | 49.4 | 46.0 | 43.8 | 41.1 | 39.0 | 53.9 | 36.7 | 2.7 |
| CodaP | 82.1 | 74.4 | 69.4 | 64.1 | 59.8 | 57.1 | 53.9 | 50.5 | 48.2 | 45.3 | 43.2 | 58.9 | 38.9 | 6.5 |
| L2P+ | 73.9 | 70.9 | 69.3 | 65.9 | 64.0 | 62.6 | 60.1 | 59.5 | 59.0 | 58.2 | 56.8 | 63.7 | 17.2 | 52.0 |
| DualP+ | 71.5 | 69.0 | 69.0 | 67.4 | 66.6 | 65.9 | 64.1 | 64.0 | 64.0 | 63.4 | 62.5 | 66.1 | **9.0** | 61.6 |
| CodaP+ | 74.4 | 69.3 | 67.7 | 63.7 | 62.0 | 61.3 | 58.5 | 58.2 | 57.5 | 54.7 | 55.3 | 62.0 | 19.1 | 54.5 |
| **Ours** | 83.3 | **80.4** | **79.6** | **77.0** | **75.6** | **74.7** | **73.0** | **72.1** | **71.9** | **70.9** | **69.7** | **75.3** | 13.5 | **67.0** |

**Table 3:** Ablation study of removing each component from `ASP` respectively. The average accuracy $A_{avg}$ is reported on three benchmark datasets.

| Methods | $A_{avg}$ | | |
|---|---|---|---|
| | CIFAR100 | CUB200 | ImageNet-R |
| Ours w/o TIP | 88.5 | 83.4 | 72.2 |
| Ours w/o TSP | 88.1 | 83.0 | 73.9 |
| Ours w/o $\mathcal{L}_c$ | 88.6 | 83.2 | 74.5 |
| Ours w/ Diff TIP | 87.6 | 82.6 | 73.3 |
| Ours | **89.0** | **83.8** | **75.3** |

**Table 4:** Influence of the Incremental Shot on average accuracy $A_{avg}$ on three benchmark datasets.

| Methods | $A_{avg}$ | | |
|---|---|---|---|
| | CIFAR100 | CUB200 | ImageNet-R |
| 1-shot | 82.9 | 78.6 | 69.3 |
| 5-shot | 89.0 | 83.8 | 75.3 |
| 10-shot | 90.0 | 85.3 | 76.6 |
| 20-shot | 90.3 | 85.8 | 77.4 |

choose $\alpha = 0.8$, $\beta = 0.99$ and $\lambda = 0.1$ for all benchmark datasets. The prompt length is set as 3 for CIFAR100 and CUB200 while is set as 10 for ImageNet-R.

**Incremental Shot:** `ASP` learns new classes using $K$-shot data, and we change the shot to find out the data influence on the average accuracy $A_{avg}$. The experimental setting is the same with Sec. 5.2 and 1, 10, 20-shot of new classes are provided in incremental tasks on three datasets. The results in Tab. 4 show that more available samples per class in incremental tasks can help improve the model performance. However, we also find that the performance improvement becomes smaller as we increase shot $K$.

**Task-Specific Prompts:** The effectiveness of the self-adaptive task-specific prompts module is validated in Sec. 5.3, and we provide further analysis to evaluate the importance of the average prompt features $p_{avg}$ in Eq. (10) and the EMA in Eq. (11). We remove $p_{avg}$ by setting $\alpha = 0$ and don't update $p_{avg}$ for new tasks by setting the EMA parameter $\beta$ as 1. Based on the results in Fig. 4,

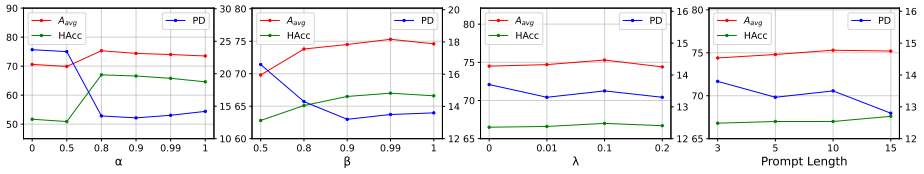


**Fig. 4:** Sensitive analysis of $\alpha$, $\beta$, $\lambda$ and prompt length $L_g = L_d$. The average accuracy $A_{avg}$ is reported on ImageNet-R datasets.

both cases lead to performance drop on $A_{avg}$, PD and HACC, which validates the effectiveness of $p_{avg}$ and EMA.

**Base&New Class Accuracy:** Except HAcc metrics, we provide the detailed accuracy of the base classes from task 0 and the new classes from task $t > 0$ after the last task $T$. The results are shown in Fig. 5. `ASP` outperforms all baselines in terms of learning new classes using few-shot data while achieving competitive performance in maintaining performance on base classes.

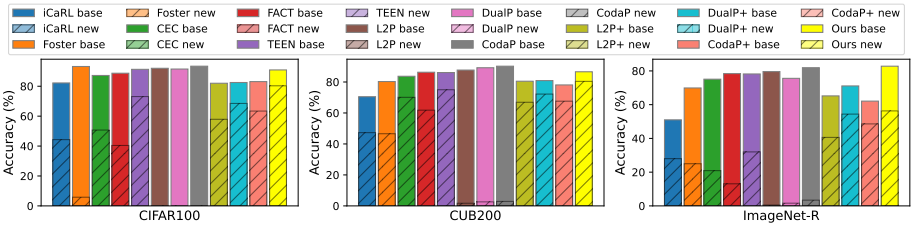


**Fig. 5:** Comparison of baselines and `ASP` on detailed accuracy of base and new classes after the last task. `ASP` outperforms all baselines in terms of learning new classes while achieving competitive performance in maintaining performance on base classes.

## 6 Conclusion

In this work, we first point out the limitations of applying existing FSCIL methods and prompt-based CIL methods on FSCIL scenarios with large vision models. We further propose a new framework called `ASP` to learn generalized prompts to leverage the generalization of pre-trained ViT for incrementally learning new classes with limited data. Under the FSCIL setting, `ASP` outperforms baseline methods from classical CIL, FSCIL, and prompt-based CIL on three benchmark datasets.

## Acknowledgments

This work was partially supported by NSF IIS 2347592, 2347604, 2348159, 2348169, DBI 2405416, CCF 2348306, CNS 2347617.

## References

1. Alemi, A.A., Fischer, I., Dillon, J.V., Murphy, K.: Deep variational information bottleneck. arXiv preprint arXiv:1612.00410 (2016)
2. Alemi, A.A., Fischer, I., Dillon, J.V., Murphy, K.: Deep variational information bottleneck. In: International Conference on Learning Representations (2017)
3. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV). pp. 139–154 (2018)
4. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. Advances in neural information processing systems **32** (2019)
5. Cheraghian, A., Rahman, S., Fang, P., Roy, S.K., Petersson, L., Harandi, M.: Semantic-aware knowledge distillation for few-shot class-incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2534–2543 (2021)
6. Chi, Z., Gu, L., Liu, H., Wang, Y., Yu, Y., Tang, J.: Metafscil: A meta-learning approach for few-shot class incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14166–14175 (2022)
7. D'Alessandro, M., Alonso, A., Calabrés, E., Galar, M.: Multimodal parameter-efficient few-shot class incremental learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3393–3403 (2023)
8. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence **44**(7), 3366–3385 (2021)
9. Dong, S., Hong, X., Tao, X., Chang, X., Wei, X., Gong, Y.: Few-shot class-incremental learning via relation knowledge distillation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1255–1263 (2021)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
11. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
12. Goswami, D., Liu, Y., Twardowski, B., van de Weijer, J.: Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. arXiv preprint arXiv:2309.14062 (2023)
13. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al.: The many faces of robustness: A critical analysis of out-of-distribution generalization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8340–8349 (2021)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)

15. Huang, Z., Chen, Z., Chen, Z., Zhou, E., Xu, X., Goh, R.S.M., Liu, Y., Feng, C., Zuo, W.: Learning prompt with distribution-based feature replay for few-shot class-incremental learning. arXiv preprint arXiv:2401.01598 (2024)
16. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: European Conference on Computer Vision. pp. 709–727. Springer (2022)
17. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
19. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
20. Kukleva, A., Kuehne, H., Schiele, B.: Generalized and incremental few-shot learning by explicit learning and calibration without forgetting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9020–9029 (October 2021)
21. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. In: Moens, M.F., Huang, X., Specia, L., Yih, S.W.t. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 3045–3059. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (Nov 2021). https://doi.org/10.18653/v1/2021.emnlp-main.243, https://aclanthology.org/2021.emnlp-main.243
22. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
23. Lin, Y., Dong, H., Wang, H., Zhang, T.: Bayesian invariant risk minimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16021–16030 (2022)
24. Liu, C., Wang, L., Lyu, L., Sun, C., Wang, X., Zhu, Q.: Deja vu: Continual model generalization for unseen domains. In: The Eleventh International Conference on Learning Representations (2023)
25. Lomonaco, V., Maltoni, D., Pellegrini, L., et al.: Rehearsal-free continual learning over small non-iid batches. In: CVPR Workshops. vol. 1, p. 3 (2020)
26. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., Van De Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(5), 5513–5533 (2022)
27. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
28. Peng, C., Zhao, K., Wang, T., Li, M., Lovell, B.C.: Few-shot class-incremental learning from an open-set perspective. In: European Conference on Computer Vision. pp. 382–397. Springer (2022)
29. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)

30. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
31. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. Advances in Neural Information Processing Systems **32** (2019)
32. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
33. Shi, G., Chen, J., Zhang, W., Zhan, L.M., Wu, X.M.: Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. Advances in neural information processing systems **34**, 6747–6761 (2021)
34. Smith, J.S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R., Kira, Z.: Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11909–11919 (2023)
35. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems **30** (2017)
36. Song, Z., Zhao, Y., Shi, Y., Peng, P., Yuan, L., Tian, Y.: Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24183–24192 (2023)
37. Sun, H.L., Zhou, D.W., Ye, H.J., Zhan, D.C.: Pilot: A pre-trained model-based continual learning toolbox. arXiv preprint arXiv:2309.07117 (2023)
38. Sun, L., Zhang, M., Wang, B., Tiwari, P.: Few-shot class-incremental learning for medical time series classification. IEEE Journal of Biomedical and Health Informatics (2023)
39. Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., Gong, Y.: Few-shot class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
40. Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., Gong, Y.: Few-shot class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12183–12192 (2020)
41. Tian, S., Li, L., Li, W., Ran, H., Ning, X., Tiwari, P.: A survey on few-shot class-incremental learning. Neural Networks **169**, 307–324 (2024)
42. Van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019)
43. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
44. Wang, F.Y., Zhou, D.W., Ye, H.J., Zhan, D.C.: Foster: Feature boosting and compression for class-incremental learning. In: European conference on computer vision. pp. 398–414. Springer (2022)
45. Wang, L., Liu, C., Guo, J., Dong, J., Wang, X., Huang, H., Zhu, Q.: Federated continual novel class learning. arXiv preprint arXiv:2312.13500 (2023)
46. Wang, L., Xie, J., Zhang, X., Huang, M., Su, H., Zhu, J.: Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. arXiv preprint arXiv:2310.07234 (2023)
47. Wang, Q.W., Zhou, D.W., Zhang, Y.K., Zhan, D.C., Ye, H.J.: Few-shot class-incremental learning via training-free prototype calibration. arXiv preprint arXiv:2312.05229 (2023)

48. Wang, Z., Li, Y., Shen, L., Huang, H.: A unified and general framework for continual learning. In: The Twelfth International Conference on Learning Representations (2024)
49. Wang, Z., Shen, L., Duan, T., Suo, Q., Fang, L., Liu, W., Gao, M.: Distributionally robust memory evolution with generalized divergence for continual learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
50. Wang, Z., Shen, L., Fang, L., Suo, Q., Zhan, D., Duan, T., Gao, M.: Meta-learning with less forgetting on large-scale non-stationary task distributions. In: European Conference on Computer Vision. pp. 221–238. Springer (2022)
51. Wang, Z., Shen, L., Zhan, D., Suo, Q., Zhu, Y., Duan, T., Gao, M.: Metamix: Towards corruption-robust continual learning with temporally self-adaptive data transformation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24521–24531 (2023)
52. Wang, Z., Yang, E., Shen, L., Huang, H.: A comprehensive survey of forgetting in deep learning beyond continual learning. arXiv preprint arXiv:2307.09218 (2023)
53. Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.Y., Ren, X., Su, G., Perot, V., Dy, J., et al.: Dualprompt: Complementary prompting for rehearsal-free continual learning. In: European Conference on Computer Vision. pp. 631–648. Springer (2022)
54. Wang, Z., Zhang, Z., Lee, C.Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., Pfister, T.: Learning to prompt for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 139–149 (2022)
55. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
56. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
57. Yoon, I.U., Choi, T.M., Lee, S.K., Kim, Y.M., Kim, J.H.: Image-object-specific prompt learning for few-shot class-incremental learning. arXiv preprint arXiv:2309.02833 (2023)
58. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International conference on machine learning. pp. 3987–3995. PMLR (2017)
59. Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., Xu, Y.: Few-shot incremental learning with continually evolved classifiers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12455–12464 (June 2021)
60. Zhang, X., Jiang, J., Feng, Y., Wu, Z.F., Zhao, X., Wan, H., Tang, M., Jin, R., Gao, Y.: Grow and merge: A unified framework for continuous categories discovery. Advances in Neural Information Processing Systems **35**, 27455–27468 (2022)
61. Zhou, D.W., Wang, F.Y., Ye, H.J., Ma, L., Pu, S., Zhan, D.C.: Forward compatible few-shot class-incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9046–9056 (2022)
62. Zhou, D.W., Ye, H.J., Zhan, D.C., Liu, Z.: Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need (2023)
63. Zhu, K., Cao, Y., Zhai, W., Cheng, J., Zha, Z.J.: Self-promoted prototype refinement for few-shot class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6801–6810 (June 2021)