Mixture of Noise for Pre-Trained Model-Based Class-Incremental Learning

Kai Jiang[1,2], **Zhengyan Shi**[2,3], **Dell Zhang**[2], **Hongyuan Zhang**[2,4*] **and Xuelong Li**[2*]
[1]School of Artificial Intelligence, OPtics and ElectroNics, Northwestern Polytechnical University
[2]Institute of Artificial Intelligence (TeleAI) of China Telecom
[3]School of Computer Science & School of Artificial Intelligence, Shanghai Jiao Tong University
[4]University of Hong Kong
jk@mail.nwpu.edu.cn, macho2021@sjtu.edu.cn, dell.z@ieee.org,
hyzhang98@gmail.com, xuelong_li@ieee.org

Abstract

Class Incremental Learning (CIL) aims to continuously learn new categories while retaining the knowledge of old ones. Pre-trained models (PTMs) show promising capabilities in CIL. However, existing approaches that apply lightweight fine-tuning to backbones still induce parameter drift, thereby compromising the generalization capability of pre-trained models. Parameter drift can be conceptualized as a form of noise that obscures critical patterns learned for previous tasks. However, recent researches have shown that noise is not always harmful. For example, the large number of visual patterns learned from pre-training can be easily abused by a single task, and introducing appropriate noise can suppress some low-correlation features, thus leaving a margin for future tasks. To this end, we propose learning beneficial noise for CIL guided by information theory and propose **Mi**xture of **N**oise (MIN), aiming to mitigate the degradation of backbone generalization from adapting new tasks. Specifically, task-specific noise is learned from high-dimension features of new tasks. Then, a set of weights is adjusted dynamically for optimal mixture of different task noise. Finally, MIN embeds the beneficial noise into the intermediate features to mask the response of inefficient patterns. Extensive experiments on six benchmark datasets demonstrate that MIN achieves state-of-the-art performance in most incremental settings, with particularly outstanding results in 50-steps incremental settings. This shows the significant potential for beneficial noise in continual learning. Code is available at https://github.com/ASCIIJK/MiN-NeurIPS2025.

1 Introduction

Unlike human beings can naturally learn new concepts without forgetting, existing AI systems lack the ability for continual learning [1–4]. To address this challenge, Class-Incremental Learning (CIL) is proposed to mitigate catastrophic forgetting [5] of old knowledge when acquiring new concepts. Most traditional CIL works focus on learning-from-scratch [6–12], i.e., learning a model without pre-training. However, the pre-training technique has been widely used in real-world applications [13, 14]. Fine-tuning with pre-trained model (PTM) on downstream tasks has become a consensus within the AI community. This has sparked significant research interest in the continual fine-tuning of PTMs for downstream tasks.

Due to pre-training on large-scale datasets, PTMs appear strong generalization. Therefore, preserving the generalization ability of PTMs across all incremental phases becomes a key factor to maintain

*Corresponding authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

the performance in downstream tasks. A critical challenge arises from persistent cross-task feature interference. Pre-trained models inherently exhibit task-specific redundancy, with only limited discriminative features being task-relevant. During continual adaptation, models inadvertently assimilate non-essential features into the decision boundaries of the current task. These features induce catastrophic forgetting through two mechanisms: 1) when the features are inherited from previous tasks, they compromise established decision boundaries of preceding models, 2) adoption of these features by subsequent tasks disrupts current task boundaries. This issue stems from the parameter sensitivity of PTMs. While such models demonstrate remarkable adaptability to new tasks, this flexibility conversely renders them vulnerable to interference from subsequent tasks. Rather than pursuing enhanced feature utilization efficiency during task adaptation, it is advisable to mitigate individual tasks' dependence on redundant features.

Positive-incentive noise (Pi-Noise) has been proven effective in diverse tasks [1]. It works by masking the confusing part among different categories with noise and highlighting the recognizable part. Catastrophic forgetting stems from parameter drift for old tasks, it can be conceptualized as the intrusion of noise during learning a new task, which undermining the original pattern of the old framework. Joint training based on old tasks also changes the original parameters, but the noise introduced by joint training suppresses confusing inter-task patterns, consequently producing a positive impact. Therefore, we aim to **directly learn this kind of positive noise from new tasks** and embed it into intermediate features. The main contributions of our work are three-fold:

- We rethink class incremental learning from the perspective of noise, interpreting parameter drift as introducing noise that is destructive to old tasks. Based on this assumption, we propose to learn beneficial noise for new tasks to suppress the response of the confusing pattern among tasks.
- We design the *Noise Expansion* strategy to learn a new noise generation module for each task. To avoid multiple inference through the backbone, we mix the beneficial noise from different tasks by training a set of learnable weights using an auxiliary classifier and residual loss, dubbed the *Noise Mixture* strategy. The noise mixture from multiple tasks is embedded with intermediate features to suppress the confusing patterns among tasks.
- We validate MIN on six widely used benchmark datasets. Experimental results show that MIN only needs few learnable parameters to learn beneficial noise adapted to the new task and achieve state-of-the-art performance. Additionally, the visualization with Grad-CAM indicates that MIN effectively suppresses the response of the irrelevant region and strengthens the representation of key pattern.

2 Related Work

Class-Incremental Learning (CIL): aims to learn a unified model from a data stream without forgetting. Conventional CIL approaches can be roughly divided into several types. Rehearsal-based methods [8, 15–18] construct a fixed exemplar-set to preserve a fraction of samples from old tasks for future training. Regularization-based methods add parameter regularization terms [6, 7] or function regularization terms [9, 10, 19, 20] to transfer knowledge from the old model to the new one. Architecture-based methods [11, 21–24, 12, 25] often add new trainable modules to expand the feature space for new tasks. Analytic learning-based methods [26–29] reformulate the CIL procedure into a recursive analytic learning process.

Pre-Trained Model-Based CIL: aims to fine-tune the PTMs in sequential downstream tasks without forgetting while keeping the generalization ability of PTMs. L2P [30] and DualPrompt [31] construct a prompt pool to select the suitable prompts for samples during model inference. CODA-Prompt [32] develops the prompt selection process with attention-based weighting. SLCA [33] slows the update of the backbone and designs a classifier alignment strategy using Gaussian modeling for previous classes. APER [34] shows that the prototypical classifier is a strong baseline. FeCAM [35] explores the prototype network and shows that modeling the feature covariance relations is better than previous attempts at sampling features from normal distributions and training a linear classifier. RanPAC [36] adopts random projection to enhance the linear separability for prototype-based CIL. EASE [37] adds task-specific branches for feature expansion. COFiMA [38] selectively weighs each parameter in the weights ensemble by leveraging the Fisher information of the weights of the model. MOS [39] rectifies the model with a training-free self-refined adapter retrieval mechanism during inference.

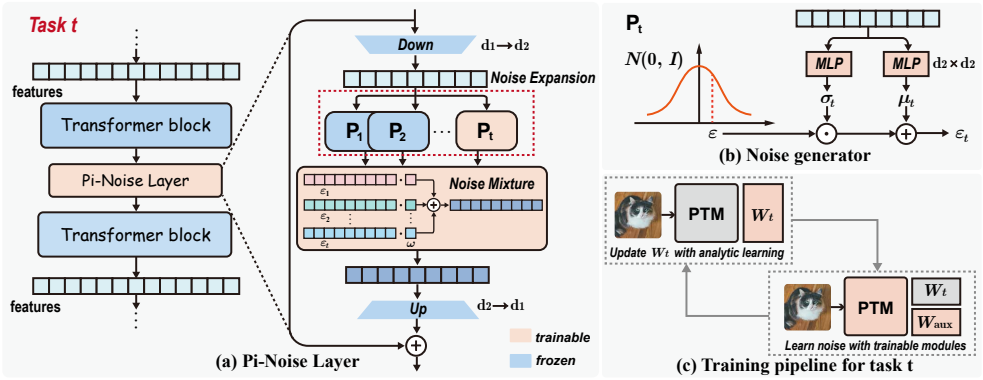

Figure 1: Illustration of MIN. **(a) Pi-Noise Layer:** is inserted between transformer blocks of PTM for learning positive noise. *Noise Expansion* adds a generator for each task and *Noise Mixture* learns to combine positive noise from different tasks. **(b) Noise generator:** samples a signal from a standard normal distribution and then transfers it into noise using the mean and variance vectors generated by MLP layers. **(c) Training pipeline for task t:** consists of 3 steps. It firstly updates classifier with analytic learning and then learns noise with trainable modules. Finally, it updates classifier again with analytic learning.

Positive-incentive Noise (Pi-Noise): is the random signal that can simplify the target task which is ubiquitous in diverse fields [1, 40–42]. Following the framework of Pi-Noise, [43] proposes an approximate method to learn Pi-noise via variational inference, which validates the efficacy of the idea about generating random noise to enhance the classifier. [44] designs an auxiliary Gaussian distribution related to contrastive loss to define task entropy and find that the developed Pi-Noise generator successfully learns augmentations on vision datasets. [45] adopts Pi-Noise to design a fine-tuning method towards vision-language alignment. In addition, [46] apply the Pi-Noise to graph contrastive learning, thus adding edges adaptively with low time and memory burden.

3 Preliminary

Class Incremental Learning: learns a unified model from a data stream. Assume that the data steam $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2 \cdots, \mathcal{D}_T\}$ consists of T tasks, with each task containing several disjoint categories. For task t, $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$ denotes the current training set with n_t instances. $x_i^t \in \mathcal{X}_t$ is the input image and $y_i^t \in \mathcal{Y}_t$ is the corresponding label, where $\mathcal{X}_t$ and $\mathcal{Y}_t$ denote the image set and the label set, respectively. Specially, we denote the label set of a new task and old tasks as $\mathcal{Y}_{\text{new}} = \mathcal{Y}_t$ and $\mathcal{Y}_{\text{old}} = \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_{t-1}$, respectively, with $\mathcal{Y}_{\text{new}} \cap \mathcal{Y}_{\text{old}} = \emptyset$. $|\mathcal{Y}_{\text{new}}| = K$ is denoted as the number of new categories, while $|\mathcal{Y}_{\text{old}}| = M$ is denoted as the number of old categories. We follow the exemplar-free setting [30, 39], where no data from previous tasks is retained in subsequent tasks. Only $\mathcal{D}_t$ can be accessed in task t. After learning each task, the model will be tested on all seen categories. By leveraging the PTM for initialization, we decompose the model $\mathbb{F}_t = \{\mathcal{F}_t, W_t\}$into two distinct components, i.e., the pre-trained backbone $\mathcal{F}_t$ and classifier W_t. Following typical PTM-based CIL works [30, 39], the most parameters of pre-trained backbone are frozen except a little part for fine-tuning.

Analytic Learning: develops a new branch for class-incremental learning [26, 27]. It iteratively solves the classifier weights through linear regression. For each task, only the current task data, the classifier weights from the previous task, and a fixed-size autocorrelation matrix are required to determine the new task classifier weights. Specifically, the classification problem can be defined in the form of linear regression, as shown in Eq. (1),

$$\arg\min_{W} \|\mathcal{Y} - \mathcal{F}(\mathcal{X})W\|_2^2 + \lambda \|W\|_2^2, \tag{1}$$

where $\mathcal{X}$ denotes the image set, $\mathcal{Y}$ the corresponding label set, $\mathcal{F}$ the feature extractor and W the classifier weight to be solved. The optimal solution to Eq. (1) can be found in

$$W = \left(\mathcal{F}(\mathcal{X})^{\mathsf{T}}\mathcal{F}(\mathcal{X}) + \lambda I\right)^{-1}\mathcal{F}(\mathcal{X})^{\mathsf{T}}\mathcal{Y}. \tag{2}$$

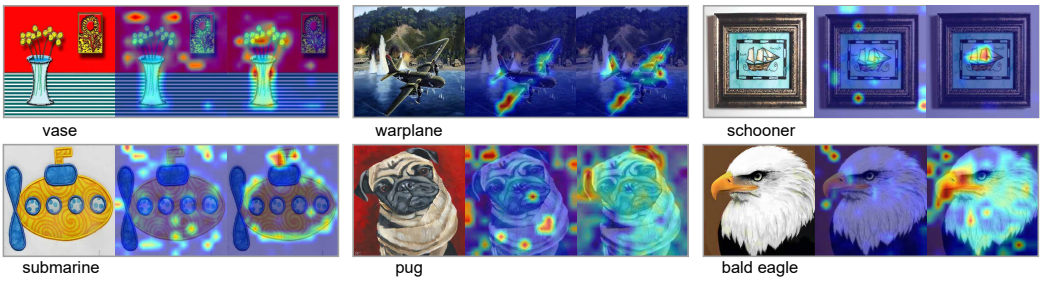

Figure 2: Grad-CAM visualization of the baseline method and MIN. The first image of each group is the original image, the second is the visualization of the baseline method, and the third image is the visualization result of the MIN. In addition, the first row shows the visualization where both methods achieve correct classification. In the second row, the baseline method is misclassified, while the MIN maintains correct.

According to [26], Eq. (2) can be rewritten in the iterative form shown in Eq. (3) for solving the classifier weight,

$$W_t = \left[W_{t-1} - R_t \mathcal{F}(\mathcal{X}_t)^\mathsf{T} \mathcal{F}\left(\mathcal{X}_t\right) W_{t-1}, R_t \mathcal{F}(\mathcal{X}_t)^\mathsf{T} \mathcal{Y}_t\right], \tag{3}$$

where W_{t-1} denotes the classifier weight at task $t-1$, $\mathcal{X}_t$ denotes the image set at task t, $\mathcal{Y}_t$ the corresponding label set. R_t is the autocorrelation matrix, which is found according to Eq. (4),

$$\begin{aligned} B_t &= \left(I + \mathcal{F}\left(\mathcal{X}_t\right) R_{t-1} \mathcal{F}(\mathcal{X}_t)^\mathsf{T}\right)^{-1}, \\ R_t &= R_{t-1} - R_{t-1} \mathcal{F}(\mathcal{X}_t)^\mathsf{T} B_t \mathcal{F}\left(\mathcal{X}_t\right) R_{t-1}. \end{aligned} \tag{4}$$

This method establishes a strong baseline [26], and the classifier derived from it serves as the foundation of our work.

4 The proposed approach: MIN

Although PTMs exhibit strong generalization capabilities, continuously fine-tuning in downstream tasks leads to rapid performance degradation. The degradation of generalization capability stems from parameter drift, which introduced a destructive noise to obscure the recognition pattern of the previous task. Therefore, we aim to **learn positive noise to adapt a new task while freezing the original parameters**. Continually learning tasks in sequence results in multiple noises from different tasks during inference. To avoid multiple inference through the backbone, we need to **integrate these noises from different tasks within each module**. In this section, we first introduce *Noise Expansion* to add a trainable noise learner for each task. Then, we adopt *Noise Mixture* to integrate these noises and embed the mixture of noise to intermediate features. As illustrated in Fig. 2, the visualization shows the impact of the proposed methods for subsequent tasks.

4.1 Noise Expansion

Due to the absence of guidance from the information among tasks, fine-tuning PTM is prone to absorb some irrelevant features to determine decision boundaries. It is equivalent to introducing noise for the current task. Since the label information within the task is available, this noise cannot affect the current task. But it may constitute the decision boundary of previous tasks and thus result in parameter drift to affect the classification pattern of old categories. To this end, we model the parameter drift as a kind of noise. Following [1], **noise is not always harmful**. For example, the noise introduced by joint training can play a positive role in classification. Therefore, we aim to **learn positive noise to suppress the confusing pattern across tasks** through a large interference to mask irrelevant activation in the features.

Suppose that the pre-trained backbone $\mathcal{F} = \{f_1, \cdots, f_L\}$ consists of L blocks. Its forward propagation process can be represented by Eq. (5) and (6),

$$r_1 = f_1\left(x\right), \tag{5}$$

$$r_l = f_l\left(r_{l-1}\right), \tag{6}$$

where x denotes the input images and f_l the l^{th} block. As illustrated in Fig. 1(a), we design Pi-Noise layers between these blocks, thereby modifying the forward inference process of the backbone network as Eq. (7),

$$r_l = \mathcal{P}_l\left(f_l\left(r_{l-1}\right)\right), \tag{7}$$

where $\mathcal{P}_l$ denotes the l^{th} Pi-Noise layer. Due to the absence of inter-task information guidance, employing a single Pi-Noise proves inadequate for effectively simplifying all tasks, thereby causing beneficial noise to degenerate into negative noise. To address this limitation, we suggest learning distinct noise for each task, with the parameters in these task-specific generators remaining mutually independent.

Algorithm 1: Training pipeline for MIN

Input: Incremental Datasets: $\{\mathcal{D}_1, \cdots, \mathcal{D}_T\}$, Pre-trained backbone: $\mathcal{F} = \{f_1, \cdots, f_L\}$
Output: Incrementally trained model

```
for t = 1 to T do
    Update the classifier W_t via Eq. (3)
    Expand the new noise generator P_t
    Initialize the weight ω via Eq. (12)
    Construct an auxiliary classifier W_aux
    Optimize the ω, P_t and W_aux via Eq. (14)
    Update the W_t via Eq. (3)
    Test the model
end
return the updated model
```

To improve efficiency of parameter utilization, we first reduce the dimensionality of the input features to minimize the scale of each noise generator. Specifically, we employ a down-projection layer $W_{\text{down}} \in \mathbb{R}^{d_1 \times d_2}$ to reduce the feature dimension from d_1 to d_2. After generating the noise, we use an up-projection layer $W_{\text{up}} \in \mathbb{R}^{d_2 \times d_1}$ to remap the dimension of noise back to d_1. Notably, both the down-projection layer W_{down} and the up-projection layer W_{up} are shared across all tasks, with **each of their elements sampled from a standard normal distribution**. For task t, P_t is employed to generate the corresponding beneficial noise, which is composed of two MLP layers denoted as ϕ_t^{μ} and ϕ_t^{σ}, responsible for producing the mean vector μ_t and variance vector σ_t, respectively. We sample a random signal ε from a normal distribution and transform it into beneficial noise as described in Eq. (8). Using this reparameterization technique, we can train P_t via backpropagation.

$$\varepsilon_t = \varepsilon \cdot \sigma_t + \mu_t. \tag{8}$$

In summary, the process of deriving beneficial noise from the input feature r_l is illustrated in Eq. (9):

$$\varepsilon_t = \varepsilon \cdot \phi_t^{\sigma}\left(r_l W_{\text{down}}\right) + \phi_t^{\mu}\left(r_l W_{\text{down}}\right). \tag{9}$$

For an intermediate feature r_l, we obtain a set of noise denoted as $\{\varepsilon_1, \cdots, \varepsilon_t\}$. By combining these noise instances, we generate a noise mixture, which is then mapped back to the d_1-dimensional space via the up-projection layer W_{up} to adjust the intermediate feature r_l,

$$\hat{r}_l = r_l + \varphi\left(\{\varepsilon_1, \cdots, \varepsilon_t\}\right) W_{\text{up}}, \tag{10}$$

where $\varphi\left(\cdot\right)$ denotes the noise mixing operation, which is detailed in the next section. For the current task, only the noise generator P_t is trained, while all noise generators remain frozen in subsequent tasks. The structure of P_t is simple, containing only two MLP layers. Since $d_2 \ll d_1$, the number of parameters trained for each task is approximately equivalent to two $d_2 \times d_2$ square matrices.

4.2 Noise Mixture

Through the *Noise Expansion*, we obtain a set of noise denoted as $\{\varepsilon_1, \ldots, \varepsilon_t\}$. If each noise is used separately for inference, it causes the test complexity to grow linearly with tasks like [37] and [39]. The simplest approach is to compute their average. However, substantial variations exist among different tasks. For task t, it may benefit from certain similar previous tasks, and thus the beneficial noise corresponding to those tasks should be assigned higher weights. Conversely, some tasks might conflict with the current task, making the noise derived from them detrimental to the current task. Therefore, when learning a new task, **it is critical to holistically consider the complexity of all tasks, ensuring that the noise mixture achieves an optimal trade-off between old and new tasksduring task adaptation**. To this end, we adjust all noise instances using a set of learnable weights ω. First,

Table 1: Average and last performance comparison on CIFAR-100 and CUB-200 datasets with **ViT-B/16-IN21K** as the backbone. We report all compared methods with their source code. The best performance is shown in bold, and the second-best result is underlined. All methods are implemented without using exemplars.

Methods	CIFAR-100						CUB-200					
	10 steps		20 steps		50 steps		10 steps		20 steps		50 steps	
	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T
L2P	85.92$_{\pm 0.78}$	79.19$_{\pm 0.57}$	81.90$_{\pm 0.54}$	73.93$_{\pm 0.33}$	74.29$_{\pm 0.41}$	61.83$_{\pm 0.26}$	84.29$_{\pm 0.32}$	74.51$_{\pm 0.36}$	81.75$_{\pm 0.18}$	70.31$_{\pm 0.25}$	78.51$_{\pm 0.47}$	64.59$_{\pm 0.32}$
DualPrompt	89.65$_{\pm 0.55}$	84.89$_{\pm 0.79}$	85.57$_{\pm 0.62}$	79.27$_{\pm 0.85}$	73.66$_{\pm 0.35}$	63.97$_{\pm 0.41}$	84.39$_{\pm 0.54}$	73.45$_{\pm 0.75}$	83.79$_{\pm 0.65}$	72.48$_{\pm 1.05}$	78.06$_{\pm 0.45}$	64.80$_{\pm 0.82}$
CODA-Prompt	91.05$_{\pm 0.32}$	86.44$_{\pm 0.87}$	87.51$_{\pm 0.70}$	80.87$_{\pm 0.25}$	69.54$_{\pm 0.58}$	55.66$_{\pm 0.48}$	84.15$_{\pm 0.87}$	74.00$_{\pm 1.05}$	83.89$_{\pm 0.37}$	71.33$_{\pm 0.55}$	75.66$_{\pm 0.25}$	61.83$_{\pm 0.32}$
ACIL	91.21$_{\pm 0.03}$	86.74$_{\pm 0.05}$	91.34$_{\pm 0.03}$	86.78$_{\pm 0.02}$	91.51$_{\pm 0.07}$	86.84$_{\pm 0.15}$	91.74$_{\pm 0.21}$	87.22$_{\pm 0.16}$	91.83$_{\pm 0.18}$	87.09$_{\pm 0.17}$	91.73$_{\pm 0.33}$	86.87$_{\pm 0.19}$
SLCA	92.67$_{\pm 0.89}$	89.30$_{\pm 0.44}$	93.32$_{\pm 0.76}$	88.21$_{\pm 0.55}$	90.76$_{\pm 0.30}$	84.52$_{\pm 0.57}$	86.83$_{\pm 0.44}$	79.47$_{\pm 0.57}$	83.38$_{\pm 0.34}$	74.77$_{\pm 0.28}$	76.60$_{\pm 0.47}$	62.51$_{\pm 0.62}$
FeCAM	93.23$_{\pm 0.15}$	89.05$_{\pm 0.17}$	91.86$_{\pm 0.11}$	87.04$_{\pm 0.26}$	90.92$_{\pm 0.17}$	85.34$_{\pm 0.39}$	92.73$_{\pm 0.27}$	88.38$_{\pm 0.33}$	92.89$_{\pm 0.25}$	88.42$_{\pm 0.20}$	91.16$_{\pm 0.34}$	85.88$_{\pm 0.43}$
RanPAC	93.25$_{\pm 0.07}$	89.55$_{\pm 0.11}$	91.81$_{\pm 0.15}$	88.69$_{\pm 0.12}$	91.75$_{\pm 0.10}$	87.19$_{\pm 0.08}$	93.30$_{\pm 0.14}$	89.78$_{\pm 0.15}$	93.30$_{\pm 0.17}$	89.27$_{\pm 0.20}$	–	–
APER	92.22$_{\pm 0.05}$	87.45$_{\pm 0.14}$	90.57$_{\pm 0.09}$	85.03$_{\pm 0.11}$	88.75$_{\pm 0.09}$	82.37$_{\pm 0.08}$	88.32$_{\pm 0.15}$	85.75$_{\pm 0.10}$	88.34$_{\pm 0.08}$	85.42$_{\pm 0.05}$	88.42$_{\pm 0.13}$	85.29$_{\pm 0.15}$
EASE	92.11$_{\pm 0.27}$	87.72$_{\pm 0.88}$	91.51$_{\pm 0.33}$	85.80$_{\pm 0.52}$	84.31$_{\pm 0.25}$	74.47$_{\pm 0.68}$	90.12$_{\pm 0.87}$	83.76$_{\pm 1.02}$	90.62$_{\pm 0.65}$	83.72$_{\pm 0.62}$	91.54$_{\pm 0.44}$	86.51$_{\pm 0.32}$
COFiMA	93.87$_{\pm 0.16}$	89.77$_{\pm 0.08}$	92.86$_{\pm 0.27}$	88.09$_{\pm 0.11}$	–	–	88.17$_{\pm 0.35}$	79.64$_{\pm 0.88}$	83.52$_{\pm 1.22}$	74.77$_{\pm 1.05}$	–	–
MOS	93.83$_{\pm 0.12}$	90.19$_{\pm 0.21}$	93.10$_{\pm 0.06}$	89.10$_{\pm 0.25}$	92.36$_{\pm 0.04}$	87.39$_{\pm 0.18}$	92.08$_{\pm 0.30}$	88.17$_{\pm 0.46}$	92.62$_{\pm 0.35}$	88.51$_{\pm 0.18}$	91.99$_{\pm 0.12}$	87.59$_{\pm 0.20}$
MIN (Ours)	**95.12**$_{\pm 0.05}$	**92.12**$_{\pm 0.16}$	**94.31**$_{\pm 0.04}$	**91.03**$_{\pm 0.29}$	**93.63**$_{\pm 0.20}$	**89.82**$_{\pm 0.36}$	**94.00**$_{\pm 0.15}$	**91.22**$_{\pm 0.18}$	**93.84**$_{\pm 0.12}$	**90.54**$_{\pm 0.14}$	**93.21**$_{\pm 0.14}$	**89.95**$_{\pm 0.22}$

initialize the ω as Eq. 12.

$$s_{t,i} = \frac{\mu_t \cdot \mu_i}{\|\mu_t\| \, \|\mu_i\|}, \tag{11}$$

$$w_i = \frac{\exp\left(s_{t,i}/\tau\right)}{\sum_{j=1}^{t} \exp\left(s_{t,j}/\tau\right)}, \tag{12}$$

where μ_i denotes the task prototype saved from the task i, τ is the temperature coefficient and is set to 2 by default. The noise mixing operation is expressed as Eq. (13),

$$\varphi\left(\{\varepsilon_1, \cdots, \varepsilon_t\}\right) = \sum\nolimits_{i=1}^{t} \varepsilon_i \omega_i. \tag{13}$$

Then, the output feature r_L of the backbone is given by Eq. (5) and (7).

$$\mathcal{L}_{cls} = \ell\left(z_L W_{\text{aux}}, y - z_L W_t\right). \tag{14}$$

For the training pipeline, we first update the classifier W_t according to Eq. (3) and construct an auxiliary classifier W_{aux} with all elements initialized to zero. Then, we update P_t and W_{aux} using the cross-entropy loss $\ell(\cdot)$ to fit the residual between the output logits $z_L W_t$ and the ground truth according to Eq. (14). It worth noting that the classifier W_t does not participate in gradient updates during training. After training, we update W_t once more with the updated P_t according to Eq. (3) while W_{aux} is discarded. We summarize the training pipeline of MIN in Algorithm 1.

5 Experiments

In this section, we evaluate MIN on several benchmark datasets and compare it with other SOTA methods to demonstrate its superiority. In addition, we provide an ablation study and a visualized analysis to validate the effectiveness of MIN. More experimental results can be found in the supplementary.

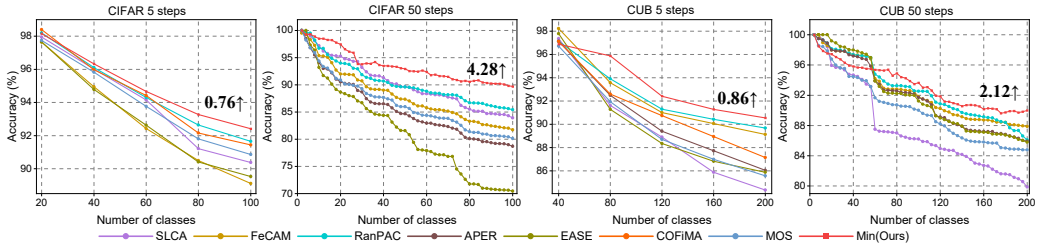

Figure 3: The performance of each learning session under different settings of CIFAR-100 and CUB-200. All methods are initialized with **ViT-B/16-IN1K**.

Table 2: Average and last performance comparison on ImageNet-A and ImageNet-R datasets with **ViT-B/16-IN21K** as the backbone. We report all compared methods with their source code. The best performance is shown in bold, and the second-best result is underlined. All methods are implemented without using exemplars.

Methods	ImageNet-A						ImageNet-R					
	10 steps		20 steps		50 steps		10 steps		20 steps		50 steps	
	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T
L2P	54.90$_{\pm 1.22}$	43.12$_{\pm 1.06}$	54.25$_{\pm 1.35}$	41.41$_{\pm 2.08}$	49.89$_{\pm 0.88}$	36.41$_{\pm 0.62}$	66.66$_{\pm 0.36}$	59.78$_{\pm 0.56}$	63.75$_{\pm 0.36}$	55.78$_{\pm 0.45}$	57.86$_{\pm 0.22}$	48.33$_{\pm 0.18}$
DualPrompt	52.89$_{\pm 0.85}$	39.89$_{\pm 0.65}$	50.36$_{\pm 1.02}$	36.87$_{\pm 0.84}$	43.85$_{\pm 0.77}$	29.95$_{\pm 1.06}$	70.29$_{\pm 0.25}$	66.50$_{\pm 0.65}$	66.52$_{\pm 0.33}$	61.77$_{\pm 0.40}$	55.38$_{\pm 0.28}$	47.40$_{\pm 0.22}$
CODA-Prompt	50.16$_{\pm 1.35}$	40.42$_{\pm 1.27}$	49.19$_{\pm 2.08}$	38.05$_{\pm 1.55}$	38.24$_{\pm 0.66}$	26.60$_{\pm 0.82}$	76.76$_{\pm 0.59}$	72.98$_{\pm 0.47}$	70.45$_{\pm 0.56}$	64.68$_{\pm 0.51}$	55.82$_{\pm 0.36}$	45.43$_{\pm 0.12}$
ACIL	63.30$_{\pm 0.42}$	53.26$_{\pm 0.66}$	62.45$_{\pm 0.57}$	52.28$_{\pm 0.55}$	62.48$_{\pm 0.36}$	50.92$_{\pm 0.21}$	75.17$_{\pm 0.05}$	68.40$_{\pm 0.22}$	75.42$_{\pm 0.08}$	68.19$_{\pm 0.16}$	75.37$_{\pm 0.09}$	67.82$_{\pm 0.16}$
SLCA	–	–	–	–	–	–	82.26$_{\pm 0.26}$	76.82$_{\pm 0.34}$	81.85$_{\pm 0.22}$	76.63$_{\pm 0.28}$	79.35$_{\pm 0.31}$	72.87$_{\pm 0.18}$
FeCAM	56.04$_{\pm 1.05}$	46.41$_{\pm 0.65}$	55.41$_{\pm 1.15}$	45.29$_{\pm 0.94}$	55.36$_{\pm 0.85}$	45.16$_{\pm 0.65}$	79.02$_{\pm 0.22}$	72.53$_{\pm 0.18}$	77.84$_{\pm 0.15}$	71.05$_{\pm 0.26}$	63.69$_{\pm 0.17}$	55.93$_{\pm 0.11}$
RanPAC	64.61$_{\pm 1.35}$	54.05$_{\pm 0.45}$	62.37$_{\pm 1.07}$	46.61$_{\pm 0.65}$	–	–	82.12$_{\pm 0.31}$	77.55$_{\pm 0.16}$	77.88$_{\pm 0.09}$	72.27$_{\pm 0.15}$	75.62$_{\pm 0.27}$	69.28$_{\pm 0.16}$
APER	60.26$_{\pm 0.72}$	48.91$_{\pm 0.66}$	60.84$_{\pm 1.05}$	48.78$_{\pm 0.36}$	61.29$_{\pm 0.82}$	48.58$_{\pm 0.26}$	75.45$_{\pm 0.11}$	67.32$_{\pm 0.04}$	74.93$_{\pm 0.15}$	67.30$_{\pm 0.02}$	69.29$_{\pm 0.10}$	61.12$_{\pm 0.04}$
EASE	57.99$_{\pm 1.15}$	45.36$_{\pm 1.34}$	58.18$_{\pm 0.98}$	46.21$_{\pm 1.33}$	59.86$_{\pm 0.78}$	47.53$_{\pm 0.92}$	81.75$_{\pm 0.32}$	76.20$_{\pm 0.19}$	81.18$_{\pm 0.20}$	74.62$_{\pm 0.09}$	76.51$_{\pm 0.31}$	68.67$_{\pm 0.24}$
COFiMA	57.70$_{\pm 0.86}$	47.60$_{\pm 0.78}$	58.43$_{\pm 0.75}$	48.12$_{\pm 1.04}$	–	–	82.05$_{\pm 0.15}$	76.43$_{\pm 0.12}$	81.54$_{\pm 0.16}$	75.15$_{\pm 0.14}$	–	–
MOS	67.71$_{\pm 0.62}$	57.14$_{\pm 0.92}$	65.10$_{\pm 0.44}$	54.25$_{\pm 0.32}$	63.72$_{\pm 0.65}$	51.22$_{\pm 0.45}$	82.59$_{\pm 0.34}$	77.80$_{\pm 0.20}$	81.32$_{\pm 0.22}$	75.62$_{\pm 0.12}$	75.22$_{\pm 0.15}$	67.18$_{\pm 0.08}$
MIN (Ours)	**72.89**$_{\pm 0.45}$	**64.32**$_{\pm 0.89}$	**72.37**$_{\pm 0.60}$	**63.66**$_{\pm 0.26}$	**66.15**$_{\pm 0.35}$	**57.08**$_{\pm 0.27}$	**85.18**$_{\pm 0.19}$	**79.75**$_{\pm 0.07}$	**83.69**$_{\pm 0.17}$	**78.08**$_{\pm 0.15}$	**82.26**$_{\pm 0.16}$	**75.72**$_{\pm 0.19}$

5.1 Experimental settings

Datasets. We conduct experiments on six benchmark datasets, including CIFAR100 [47], CUB200 [48], ImageNet-A [49], ImageNet-R [50], FOOD101 [51] and Omnibenchmark [52]. There are 100 categories in CIFAR100, 101 categories in FOOD101, 200 categories in CUB200, ImageNet-A, ImageNet-R, and 300 categories in Omnibenchmark. Detailed information for the datasets is shown in the supplementary.

Protocols. Following recent CIL works, we split a dataset into T tasks, denoted as T steps. Each task learns the same number of categories. To further investigate the performance of all methods across various step sizes, we set the range of T from 5 to 50, i.e., $T = 5$, 10, 20, and 50. Following [37, 34, 39], the learning order is determined by the random seed 1993. We run each experiment 3 times and report the average result. Additionally, we also report the results using other random seeds in the supplementary.

Evaluation metric. After learning the task t, we test the model on all known categories. Assuming that A_t represents the accuracy of the model after t tasks. We use two metrics, the average accuracy across T tasks $\bar{A} = \frac{1}{T}\sum_{t=1}^{T} A_t$ and the last task accuracy A_T, to measure the CIL performance.

Comparison methods. We choose state-of-the-art PTM-based CIL methods for comparison. To be specific, we choose L2P [30], DualPrompt [31], CODA-Prompt [32], SLCA [33], FeCAM [35], RanPAC [36], APER [34], EASE [37], COFiMA [38] and MOS [39]. In addition, we also choose ACIL [26] as the baseline of our work.

Implementation details. Since the wide range of PTMs are publicly accessible, we choose two representative models following [31, 37, 34, 39], denoted as **ViT-B/16-IN21K** [53] and **ViT-B/16-IN1K**. They are both initially pre-trained on ImageNet-21K, while the latter is further finetuned on ImageNet-1K. In MIN, we set the batch size to 128 and train for 10 epochs using SGD optimizer with momentum. The learning rate is initially set to 0.001 and decays to 0 following a cosine annealing

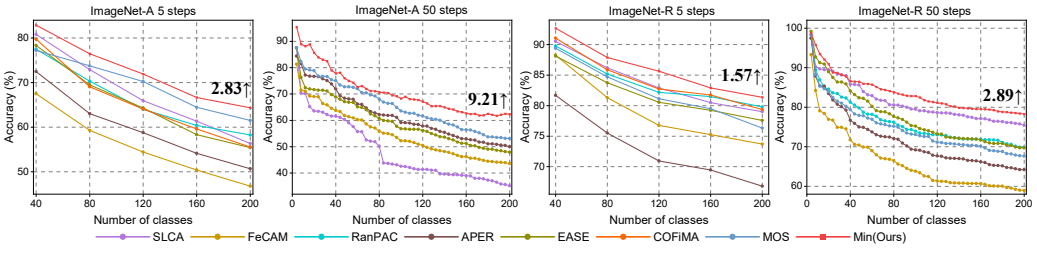

Figure 4: The performance of each learning session under different settings of ImageNet-A and ImageNet-R. All methods are initialized with **ViT-B/16-IN1K**.

Table 3: Average and last performance comparison on FOOD-101 and Omnibenchmark datasets with **ViT-B/16-IN21K** as the backbone. We report all compared methods with their source code. The best performance is shown in bold, and the second-best result is underlined. All methods are implemented without using exemplars.

Methods	FOOD-101						Omnibenchmark					
	10 steps		20 steps		50 steps		10 steps		20 steps		50 steps	
	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T
L2P	82.92$_{\pm 0.65}$	74.93$_{\pm 0.85}$	78.74$_{\pm 0.54}$	70.66$_{\pm 0.52}$	59.60$_{\pm 0.38}$	50.26$_{\pm 0.26}$	73.01$_{\pm 0.33}$	63.29$_{\pm 0.26}$	71.65$_{\pm 0.34}$	61.60$_{\pm 0.16}$	69.54$_{\pm 0.47}$	59.05$_{\pm 0.22}$
DualPrompt	86.25$_{\pm 0.45}$	80.02$_{\pm 0.40}$	80.48$_{\pm 0.37}$	72.78$_{\pm 0.26}$	67.23$_{\pm 0.58}$	58.80$_{\pm 0.67}$	77.68$_{\pm 0.18}$	67.45$_{\pm 0.45}$	74.13$_{\pm 0.35}$	64.88$_{\pm 0.27}$	71.28$_{\pm 0.56}$	59.93$_{\pm 0.38}$
CODA-Prompt	88.91$_{\pm 0.33}$	83.14$_{\pm 0.15}$	83.19$_{\pm 0.30}$	76.53$_{\pm 0.26}$	62.84$_{\pm 1.77}$	51.57$_{\pm 0.57}$	79.18$_{\pm 0.42}$	70.01$_{\pm 0.65}$	74.85$_{\pm 0.18}$	65.78$_{\pm 0.22}$	70.50$_{\pm 0.14}$	60.50$_{\pm 0.30}$
ACIL	90.68$_{\pm 0.03}$	86.03$_{\pm 0.07}$	90.66$_{\pm 0.08}$	85.81$_{\pm 0.07}$	90.34$_{\pm 0.04}$	85.38$_{\pm 0.05}$	84.26$_{\pm 0.35}$	76.11$_{\pm 0.52}$	84.76$_{\pm 0.26}$	75.96$_{\pm 0.47}$	84.28$_{\pm 0.18}$	75.31$_{\pm 0.44}$
SLCA	91.10$_{\pm 0.37}$	86.40$_{\pm 0.17}$	89.61$_{\pm 0.27}$	83.42$_{\pm 0.08}$	86.08$_{\pm 0.16}$	78.69$_{\pm 0.12}$	82.46$_{\pm 0.26}$	74.24$_{\pm 0.19}$	80.55$_{\pm 0.44}$	70.77$_{\pm 0.75}$	78.48$_{\pm 0.21}$	69.14$_{\pm 0.15}$
FeCAM	90.87$_{\pm 0.50}$	86.56$_{\pm 0.42}$	90.75$_{\pm 0.32}$	86.05$_{\pm 0.31}$	90.08$_{\pm 0.24}$	85.21$_{\pm 0.11}$	83.74$_{\pm 0.16}$	77.18$_{\pm 0.22}$	83.02$_{\pm 0.18}$	76.19$_{\pm 0.16}$	83.01$_{\pm 0.26}$	75.58$_{\pm 0.31}$
RanPAC	92.13$_{\pm 0.32}$	88.90$_{\pm 0.15}$	91.61$_{\pm 0.20}$	87.16$_{\pm 0.15}$	91.07$_{\pm 0.16}$	86.64$_{\pm 0.09}$	85.04$_{\pm 0.12}$	78.83$_{\pm 0.18}$	84.14$_{\pm 0.08}$	76.77$_{\pm 0.05}$	84.16$_{\pm 0.10}$	76.33$_{\pm 0.05}$
APER	88.49$_{\pm 0.19}$	83.60$_{\pm 0.05}$	88.49$_{\pm 0.18}$	83.19$_{\pm 0.05}$	89.03$_{\pm 0.08}$	83.38$_{\pm 0.04}$	80.72$_{\pm 0.20}$	74.34$_{\pm 0.14}$	80.26$_{\pm 0.21}$	73.36$_{\pm 0.16}$	80.27$_{\pm 0.08}$	73.00$_{\pm 0.10}$
EASE	88.63$_{\pm 0.25}$	83.26$_{\pm 0.22}$	85.27$_{\pm 0.28}$	79.30$_{\pm 0.35}$	85.61$_{\pm 0.30}$	78.75$_{\pm 0.33}$	75.71$_{\pm 0.45}$	68.59$_{\pm 0.40}$	74.99$_{\pm 0.37}$	65.72$_{\pm 0.25}$	73.56$_{\pm 0.48}$	65.52$_{\pm 0.37}$
COFiMA	90.83$_{\pm 0.06}$	85.84$_{\pm 0.08}$	88.12$_{\pm 0.15}$	81.82$_{\pm 0.16}$	–	–	81.69$_{\pm 0.40}$	73.38$_{\pm 0.28}$	80.30 $_{\pm 0.42}$	71.18$_{\pm 0.25}$	–	–
MOS	92.54$_{\pm 0.11}$	89.27$_{\pm 0.08}$	92.02$_{\pm 0.18}$	88.29$_{\pm 0.22}$	91.57$_{\pm 0.10}$	87.08$_{\pm 0.08}$	86.10$_{\pm 0.11}$	80.12$_{\pm 0.15}$	85.22$_{\pm 0.08}$	78.58$_{\pm 0.12}$	85.06$_{\pm 0.12}$	77.49$_{\pm 0.15}$
MIN (Ours)	**93.36**$_{\pm 0.07}$	**90.04**$_{\pm 0.06}$	**93.20**$_{\pm 0.12}$	**89.55**$_{\pm 0.16}$	**93.60**$_{\pm 0.07}$	**89.47**$_{\pm 0.12}$	**87.36**$_{\pm 0.04}$	**80.55**$_{\pm 0.10}$	**87.79**$_{\pm 0.03}$	**79.90**$_{\pm 0.11}$	**87.58**$_{\pm 0.15}$	**79.35**$_{\pm 0.19}$

decay pattern. The dimension d_2 of the latent vector within the Pi-Noise layer is set to 192, with more configurations of this hyperparameter detailed in the supplementary materials.

5.2 Comparison with State-of-the-arts

In this section, we compare the proposed MIN with other state-of-the-art methods on six benchmark datasets and different pre-trained weights. Complete comparison experiments are provided in the supplementary.

CIFAR & CUB. Tab. 1 presents the performance comparison of different methods using ViT-B/16-IN21K on the CIFAR100 and CUB200 datasets. We report the results under three different settings for each dataset: 10 steps, 20 steps, and 50 steps. As the number of tasks increases, the incremental learning difficulty progressively intensifies. Due to the lack of inter-task information guidance, all methods exhibit a significant performance decline in the 50-steps compared to the 10-steps scenario. The proposed method achieves the best performance across all three settings, with its advantage over the runner-up method becoming notably amplified under the 50-steps setting. Additionally, Fig. 3 presents the incremental performance trends of different methods using ViT-B/16-IN1K. For each dataset, we report results under both 5-steps and 50-steps settings to demonstrate the impact of the number of tasks on incremental performance. As shown in Fig. 3, the proposed method exhibits a more gradual decline rate in the 50-steps setting compared to other approaches. When contrasting the 5-step and 50-step results on CIFAR100, the performance advantage of our method over the runner-up approach increases from 0.76% in the 10-steps setting to 4.28% under 50-steps settings. Similarly, for CUB200, the performance gap expands from 0.86% to 2.12% between these two settings.

ImageNet-A/R. Tab. 2 presents the performance comparison of different methods using ViT-B/16-IN21K on the ImageNet-A/R datasets, with results reported under three different settings. The proposed method demonstrates superior performance across all settings, achieving a 6%–9% improvement in final accuracy over the runner-up approach on ImageNet-A, and a 2%–3% enhancement on ImageNet-R. Furthermore, Fig. 4 illustrates the incremental trends of various methods with

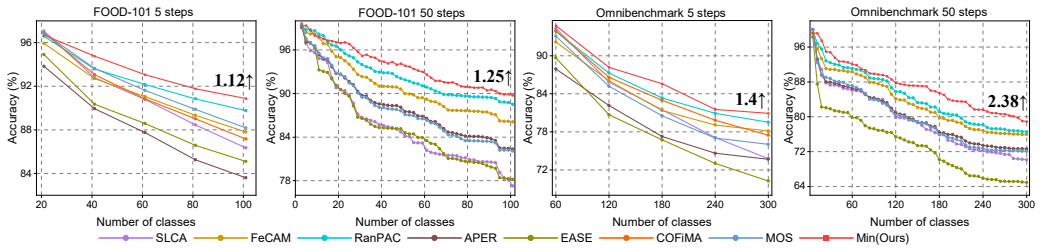

Figure 5: The performance of each learning session under different settings of FOOD-101 and Omnibenchmark. All methods are initialized with **ViT-B/16-IN1K**.

Table 4: Results of ablation study. NE+$\bar{\varepsilon}$ calculates the average noise of all tasks. NE+ε_μ only uses the μ of each task noise. NE+ε_σ only uses the σ of each task noise. NE+ε_t uses the last task noise and NE+ε_i uses a random task noise.

Methods	CIFAR		CUB		IN-A		IN-R		FOOD		Omni.	
	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T	$\overline{A}$	A_T
Baseline	91.18$_{\pm 0.03}$	86.82$_{\pm 0.07}$	91.55$_{\pm 0.16}$	87.08$_{\pm 0.17}$	62.23$_{\pm 0.45}$	53.65$_{\pm 0.57}$	75.14$_{\pm 0.05}$	68.45$_{\pm 0.27}$	90.26$_{\pm 0.03}$	85.57$_{\pm 0.10}$	84.55$_{\pm 0.33}$	76.82$_{\pm 0.44}$
w/ NE+$\bar{\varepsilon}$	94.07$_{\pm 0.04}$	90.22$_{\pm 0.10}$	92.89$_{\pm 0.09}$	90.77$_{\pm 0.15}$	71.43$_{\pm 0.55}$	62.59$_{\pm 0.70}$	83.86$_{\pm 0.24}$	78.65$_{\pm 0.11}$	92.67$_{\pm 0.05}$	89.34$_{\pm 0.09}$	86.77$_{\pm 0.15}$	80.12$_{\pm 0.18}$
w/ NE+ε_μ	89.27$_{\pm 0.24}$	85.63$_{\pm 0.33}$	88.97$_{\pm 0.46}$	85.45$_{\pm 0.45}$	65.87$_{\pm 1.27}$	55.36$_{\pm 1.02}$	76.27$_{\pm 0.44}$	67.15$_{\pm 0.34}$	88.45$_{\pm 0.09}$	83.16$_{\pm 0.12}$	82.15$_{\pm 0.68}$	73.95$_{\pm 0.70}$
w/ NE+ε_σ	94.37$_{\pm 0.05}$	91.32$_{\pm 0.12}$	93.15$_{\pm 0.17}$	90.47$_{\pm 0.09}$	70.45$_{\pm 0.25}$	61.32$_{\pm 0.33}$	84.15$_{\pm 0.20}$	79.05$_{\pm 0.12}$	92.90$_{\pm 0.05}$	89.25$_{\pm 0.04}$	87.06$_{\pm 0.07}$	80.12$_{\pm 0.05}$
w/ NE+ε_t	92.36$_{\pm 0.18}$	87.92$_{\pm 0.27}$	92.37$_{\pm 0.08}$	87.25$_{\pm 0.12}$	63.23$_{\pm 0.75}$	54.15$_{\pm 0.92}$	79.15$_{\pm 0.16}$	71.71$_{\pm 0.28}$	90.62$_{\pm 0.12}$	86.12$_{\pm 0.05}$	85.02$_{\pm 0.06}$	77.47$_{\pm 0.08}$
w/ NE+ε_i	92.65$_{\pm 0.21}$	88.10$_{\pm 0.15}$	92.15$_{\pm 0.23}$	88.76$_{\pm 0.26}$	64.39$_{\pm 1.08}$	54.88$_{\pm 1.35}$	81.39$_{\pm 0.65}$	73.15$_{\pm 0.81}$	91.08$_{\pm 0.30}$	86.57$_{\pm 0.19}$	84.77$_{\pm 0.24}$	77.96$_{\pm 0.16}$
MIN	**95.12**$_{\pm 0.05}$	**92.12**$_{\pm 0.16}$	**94.00**$_{\pm 0.15}$	**91.22**$_{\pm 0.18}$	**72.89**$_{\pm 0.45}$	**64.32**$_{\pm 0.89}$	**85.18**$_{\pm 0.19}$	**79.75**$_{\pm 0.07}$	**93.36**$_{\pm 0.07}$	**90.04**$_{\pm 0.06}$	**87.36**$_{\pm 0.04}$	**80.55**$_{\pm 0.10}$

ViT-B/16-IN1K, where our approach shows amplified advantages against the suboptimal method as task number increases.

FOOD101 & Omnibenchmark Tab. 3 demonstrates the performance comparison of various methods using ViT-B/16-IN21K on the FOOD101 and Omnibenchmark datasets. The proposed method maintains superior performance in all three settings. Fig. 5 illustrates the incremental trends with ViT-B/16-IN1K, where the findings further validate the progressive widening of the performance gaps between our method and the runner-up approach as the number of tasks increases.

5.3 Ablation Study and Visualizations

Ablation study. To verify the effectiveness of each component in MIN, we use the ViT-B/16-IN21K to conduct an ablation study on all six datasets with the 10-steps setting. **Baseline** denotes the approach using analytic classifier with the pre-trained backbone [26](see Sec. 3). It should be noted that in the absence of *Noise Mixture*, the mixing module operates by computing the average of all noises. Additionally, instead of employing an auxiliary classifier for training, it directly utilizes W_t to calculate the cross-entropy loss to update P_t. The results are presented in Tab. 4 which validate the effectiveness of each strategy.

Visualizations. We use Grad-CAM visualization to further demonstrate how MIN applies beneficial noise to improve the performance of baseline methods in Fig. 2. The comparative results demonstrate that while the baseline method also focuses on the primary subject, its activation region exhibits greater dispersion with pronounced responses in irrelevant regions. For example, in the first set of images labeled vase, the baseline method generates strong activation for the background wall and the hanging painting. But the proposed method pays more attention to the main part of the vase, and suppresses the irrelevant background information. This indicates that the proposed method works on improving the accuracy by suppressing invalid features. We provide more visualizations in the supplementary.

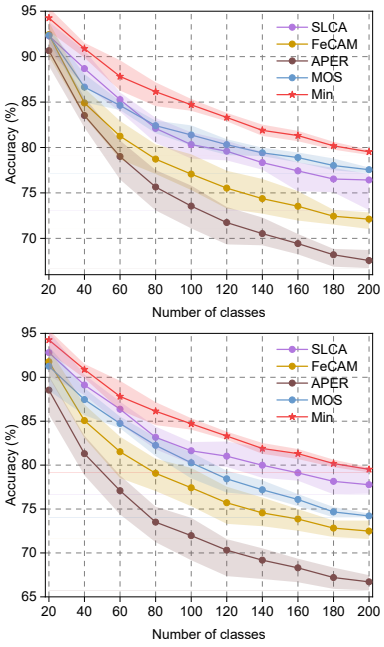

Figure 6: Incremental trends with random orders on the ImageNet-R dataset using ViT-B/16-IN21K (Top) and ViT-B/16-IN1K (Bottom).

Random seed experiments. We shuffle the learning order with the random seeds, i.e., {1993, 1994, 1995, 1996, 1997, 1998}. Therefore, we obtain six results of different learning order for each method on ImageNet-R using ViT-B/16-IN21K and ViT-B/16-IN1K. The results are shown in Fig. 6. For each method, the solid line indicates the mean of the five outcomes and the shading indicates the fluctuation range. The results demonstrate that MIN consistently outperforms other methods by a substantial margin.

Robustness of hyperparameters. There are two hyperparameters in the baseline method, i.e., the coefficient γ of regularization terms and the buffer size. We investigate the robustness by changing these hyperparameters. Specifically, we select γ from {10, 50, 100, 500, 1000} and the buffer size

from {1024, 2048, 4096, 8192, 16384}. We report the average accuracy in Fig. 7(a). The results indicate that selecting γ as 100 or 500, along with a buffer size of 8196 or 16384, yields better performance. In addition, we change the dimension of the hidden layer P_t to investigate its impact on the model performance. Specifically, we select from {96, 192, 256, 384, 512, 768} as the dimension that W_{down} reduce the input feature into. And we report the incremental trends in Fig. 7(b). Then, the incremental trends of the number of trainable parameters are shown in Fig. 7(c). As shown in the figure, the model achieves the best performance when the dimension is set to 192 (dim=192), and further increasing this dimension results in performance degradation. Hence, we adopt dim=192 as the dimension of the hidden layer in P_t. In addition, Tab. 5 shows the results of changing τ in Eq. (12). It indicates that the proposed method is not sensitive for τ.

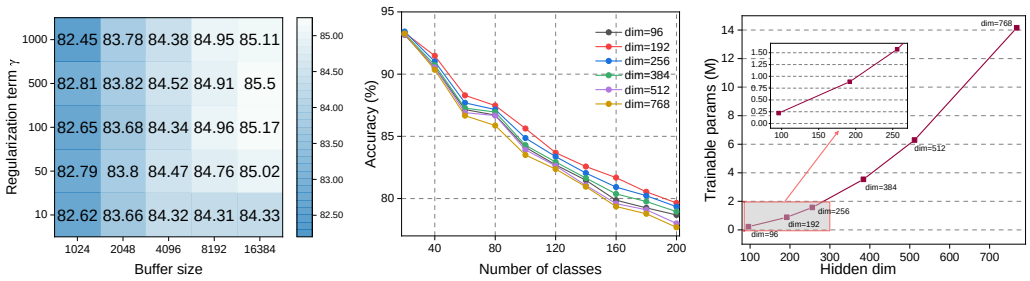

Figure 7: Further analysis on parameter robustness.

Additional experiments. We set the hyperparameter d_2 to 192, which is the same as in the comparative experiments, to analyze the number of model parameters. From Fig. 8(a) and (b), it can be seen that the total number of model parameters of MIN is the same as most of the methods, while the number of learnable parameters for new tasks is less than most of the methods. Fig. 8(c) shows the impact of Pi-Noise compared to alternatives including Adapter [54], VPT-shallow [55] and VPT-deep [55]. Following ACIL [26], we use these alternative approaches to train the model in the first task and then learn subsequent tasks by using analytic classifier. The comparison demonstrates that Pi-Noise shows superiority over other methods.

Table 5: Robustness of τ in the setting of ImageNet-R 10 steps.

τ	$\overline{A}$	A_T
0.50	84.92$_{\pm 0.21}$	79.30$_{\pm 0.23}$
1.00	84.79$_{\pm 0.22}$	79.38$_{\pm 0.10}$
1.50	84.91$_{\pm 0.19}$	79.59$_{\pm 0.12}$
2.00	85.18$_{\pm 0.19}$	79.75$_{\pm 0.07}$

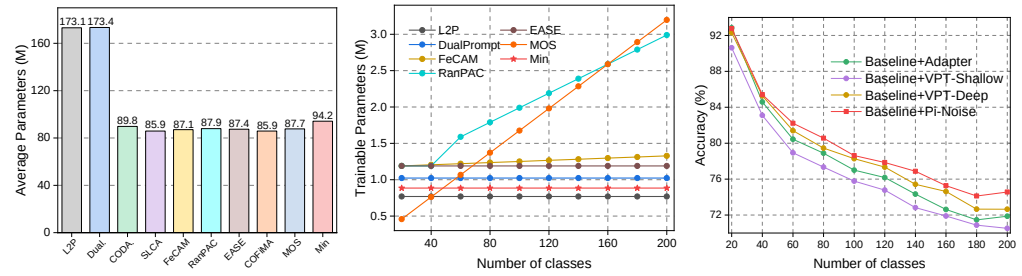

Figure 8: More experiments results. (a) presents the average number of model parameters for different methods across 10 steps. (b) presents the number of trainable parameters for different methods across 10 steps. (c) Impact of Pi-Noise compared to alternatives under the setting of ImagenNet-R 10 steps.

6 Conclusion

Incremental learning is a key step to achieve a continual AI system. This paper propose mixture of positive-incentive noise (MIN) for class incremental learning based on pre-trained models. Specifically, the proposed method expands a Pi-Noise generator for each task to adapt new tasks without the degradation of backbone generalization. Considering the cross-task collaboration among different Pi-Noises, we guide the model via an auxiliary classifier to learn a set of adaptive weights for adjusting their mixing ratios. Extensive experiments validate the effectiveness of MIN.

References

[1] Xuelong Li. Positive-incentive noise. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):8708–8714, 2024.

[2] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

[3] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[4] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[5] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

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

[7] 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*, pages 3366–3375, 2017.

[8] 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*, pages 2001–2010, 2017.

[9] 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*, pages 831–839, 2019.

[10] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 86–102. Springer, 2020.

[11] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[12] Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. Beef: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[13] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[15] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[16] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217, 2020.

[17] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems*, 34:3478–3490, 2021.

[18] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11371–11380, 2023.

[19] Arjun Ashok, KJ Joseph, and Vineeth N Balasubramanian. Class-incremental learning with cross-space clustering and controlled transfer. In *European Conference on Computer Vision*, pages 105–122. Springer, 2022.

[20] Haitao Wen, Lili Pan, Yu Dai, Heqian Qiu, Lanxiao Wang, Qingbo Wu, and Hongliang Li. Class incremental learning with multi-teacher distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28443–28452, 2024.

[21] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pages 398–414. Springer, 2022.

[22] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022.

[24] Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11858–11867, 2023.

[25] Kai Jiang, Xueru Bai, and Feng Zhou. Recurrent network expansion for class incremental learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2025.

[26] Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. *Advances in Neural Information Processing Systems*, 35:11602–11614, 2022.

[27] Huiping Zhuang, Run He, Kai Tong, Ziqian Zeng, Cen Chen, and Zhiping Lin. Ds-al: A dual-stream analytic learning for exemplar-free class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17237–17244, 2024.

[28] Huiping Zhuang, Zhenyu Weng, Run He, Zhiping Lin, and Ziqian Zeng. GKEAL: Gaussian kernel embedded analytic learning for few-shot class incremental task. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7746–7755, June 2023.

[29] Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. GACL: Exemplar-free generalized analytic continual learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., December 2024.

[30] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149, 2022.

[31] 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. In *European conference on computer vision*, pages 631–648. Springer, 2022.

[32] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 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*, pages 11909–11919, 2023.

[33] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158, 2023.

[34] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, pages 1–21, 2024.

[35] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

[36] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

[37] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23554–23564, 2024.

[38] Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024.

[39] Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[40] Siqi Huang, Yanchen Xu, Hongyuan Zhang, and Xuelong Li. Learn beneficial noise as graph augmentation. In *Forty-second International Conference on Machine Learning*, 2025.

[41] Ziheng Jiao, Hongyuan Zhang, and Xuelong Li. Cnn2gnn: How to bridge cnn with gnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[42] Jiquan Shan, Junxiao Wang, Lifeng Zhao, Liang Cai, Hongyuan Zhang, and Ioannis Liritzis. Anchorformer: Differentiable anchor attention for efficient vision transformer. *arXiv preprint arXiv:2505.16463*, 2025.

[43] Hongyuan Zhang, Sida Huang, and Xuelong Li. Variational positive-incentive noise: How noise benefits models. *arXiv preprint arXiv:2306.07651*, 2023.

[44] Hongyuan Zhang, Yanchen Xu, Sida Huang, and Xuelong Li. Data augmentation of contrastive learning is estimating positive-incentive noise. *arXiv preprint arXiv:2408.09929*, 2024.

[45] Sida Huang, Hongyuan Zhang, and Xuelong Li. Enhance vision-language alignment with noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[46] Yanchen Xu, Siqi Huang, Hongyuan Zhang, and Xuelong Li. Why does dropping edges usually outperform adding edges in graph contrastive learning? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[47] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

[48] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Technical Report CNS-TR-2011-001*, 2011.

[49] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[50] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

[51] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 446–461. Springer, 2014.

[52] Yuanhan Zhang, Zhenfei Yin, Jing Shao, and Ziwei Liu. Benchmarking omni-vision representation through the lens of visual realms. In *European Conference on Computer Vision*, pages 594–611. Springer, 2022.

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

[54] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

[55] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pages 709–727. Springer, 2022.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[57] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: See Abstraction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See supplementary materials

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes] .

Justification: see Sec. 3

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] .

   Justification: See pseudo-code and implementation details in supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes] .

Justification: We have provided the code link in the abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: See supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes] .

   Justification: We have provided the standard deviations for all the repeated experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes] .

   Justification: See the hardware information in supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes] .

    Justification: See Sec. 6

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes] .

    Justification: CC-BY 4.0

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA] .

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA] .

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

A Overview

In this supplementary material, we provide more details about MIN. Firstly, we give the implementation details in Sec. B. Then, we provide the details of six datasets in Sec. C. In Sec. D, we provide hardware information used for all experiments. Then, we provide descriptions of all comparison methods in Sec. E. In Sec. F, we provide more visualizations to further illustrate the effectiveness of MIN. In the end, we provide more comparison experimental results in Sec. G. Limitations of MIN is provided in Sec. H.

B Implementation details

We run all the experiments with PyTorch [56] and reproduce all other comparison methods with Pilot [57]. Following [30, 32, 36, 34, 39], we conduct experiments with the ViT-B/16-IN21K and ViT-B/16-IN1K. In MIN, we train the model using the SGD optimizer, with a batch size of 128 for 10 epochs. The learning rate decays from 0.001 with cosine annealing to zero. There are subtle differences in these hyper-parameter between different datasets due to scale of datatsets. We set the dimension of hidden layer as 192 according to Sec. 5.3. The hyperparameters of baseline method, i.e., regularization term γ and buffer size, are set to 100 and 16384 according to Sec. 5.3.

C Datasets

The details of six datasets are illustrated in Tab. 6, including CIFAR100, CUB, ImageNet-A/R, FOOD101 and Omnibenchmark.

Table 6: Details of six datasets.

Datasets	Classes	Train	Test	Avg size
CIFAR-100	100	50,000	10,000	32×32
CUB-200	200	9,430	2,358	467×386
ImageNet-A	200	5,960	1,515	443×427
ImageNet-R	200	24,000	6,000	443×427
Omnibenchmark	300	89,697	5,985	764×581
FOOD-101	101	75,750	25,250	496×475

D Hardware Information

The hardware information is as follows:

- **CPU**: Intel Xeon(R) Gold 6244 CPU
- **GPU**: 2×NVIDIA GeForce RTX 4090
- **Mem**: 8×DDR4 SAMSUNG-32GB

E Comparison methods

We totally choose the **10** approaches for comparison. A brief description of these methods is given below.

L2P. L2P [30] freezes the parameters of pre-trained weights and uses visual prompt tuning to learn the new tasks. It constructs a prompt pool and selects a suitable prompt for each sample with a key-value mapping strategy.

DualPrompt. DualPrompt [31] is the improvement work for L2P. It divides the prompt into two types, i.e., general prompts and expert prompts.

CODA-Prompt. CODA-Prompt [32] replaces prompt reweighting with an attention-based prompt recombination during the prompt selection process. Owning to the attention module, it needs more parameters and training time.

SLCA. SLCA [33] updates the backbone slowly and rectifies the classifier with pseudo features which are sampled from the Gaussian distribution modeled by the prototype of old categories.

FeCAM. FeCAM [35] applies the Mahalanobis distance to replace the Euclidean distance for classification without updating the backbone. In addition, it shows that modeling the feature covariance relations is better than previous attempts at sampling features from normal distributions and training a linear classifier.

RanPAC. RanPAC [36] introduces a frozen random projection layer between the feature output of the pre-trained model and the classifier, combines with a nonlinear activation function, expands the feature dimension and enhances linear separability, so as to alleviate the forgetting problem without updating the backbone.

APER. APER [34] builds the prototype-based classifier and uses a cosine classifier for classification. It adapts the new data only in the first task with additional modules. We choose the best version, i.e., APER-adapter for comparison.

EASE. EASE [37] is based on the idea of network expansion. It adds a new branch to each transformer block for each task. Although it trains only one branch in training process, it needs to infer multiple times through the backbone for test.

COFiMA. COFiMA [38] integrates the model parameters of the current task and the previous task by introducing the Fisher information weighting mechanism and balances the plasticity and stability of the model.

MOS. MOS [39] alleviates the parameter drift problem through adapter merging technology, and combines the self-optimization retrieval mechanism without training to dynamically correct the module matching error, which solves the catastrophic forgetting problem in the incremental learning of pre-trained models from two perspectives of parameters and retrieval.

F More Visualizations

We provide more visualizations in Fig. 9 to demonstrate the effectiveness of MIN. For instance, three bird species from the CUB dataset are shown, which require fine-grained recognition for accurate differentiation. The backbone network of the baseline method fails to learn cross-task fine-grained features, only achieving coarse-grained discrimination with limited activation values concentrated on the birds' main body regions. In contrast, MIN learns task-specific Pi-Noise that suppresses common features shared across bird species while generating stronger activation responses to discriminative fine-grained characteristics. This indicates that MIN can effectively inhibit the backbone network's activation of common features across similar categories in different tasks, thereby facilitating the learning of underlying target features and avoiding confusion of decision boundaries between multiple tasks.

G More Experimental Results with SOTAs

In this section, we show more experimental results of different methods. As demonstrated from Fig. 10 to Fig. 21, we report the incremental performance of different methods with ViT-B/16-IN21K and ViT-B/16-IN1K on six datasets. MIN consistently outperforms other methods on different datasets by a substantial margin.

H Limitations

MIN relies on an excellent feature extractor to provide a prior information to guide Pi-Noise generator for adaptation of subsequent downstream tasks. With the rapid development of pre-training technology, the weights of the backbone network trained by self-supervised learning on large-scale unlabeled data are easy to access, which reduces this limitations. In addition, MIN requires an additional downsampling layer in each Pi-Noise layer (although they are training-free). Introduces additional

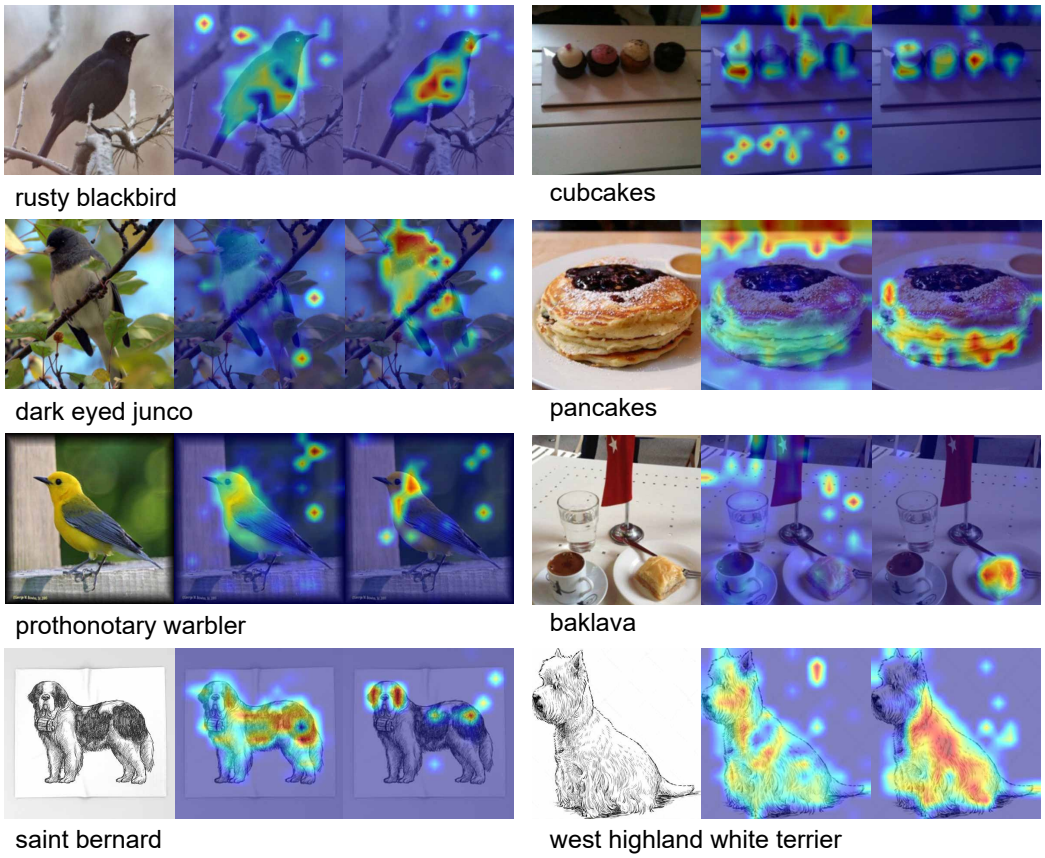

Figure 9: More visualizations

network parameters (equivalent to 4% of the number of parameters for ViT-B/16). However, the number of actual training parameters is much smaller (equivalent to 0.8M parameters). Therefore, it is worth trade-off for the simplicity of implementation and low-cost training.

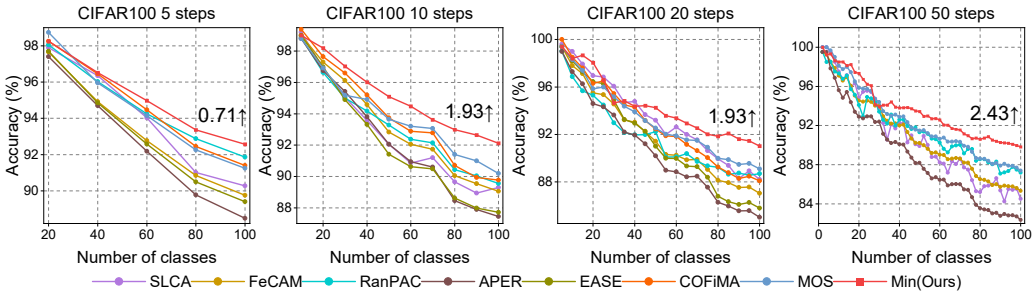

Figure 10: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the CIFAR100 dataset.

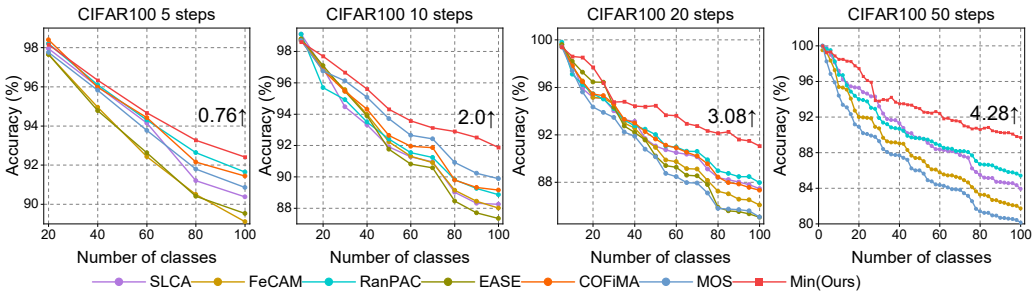

Figure 11: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the CIFAR100 dataset.

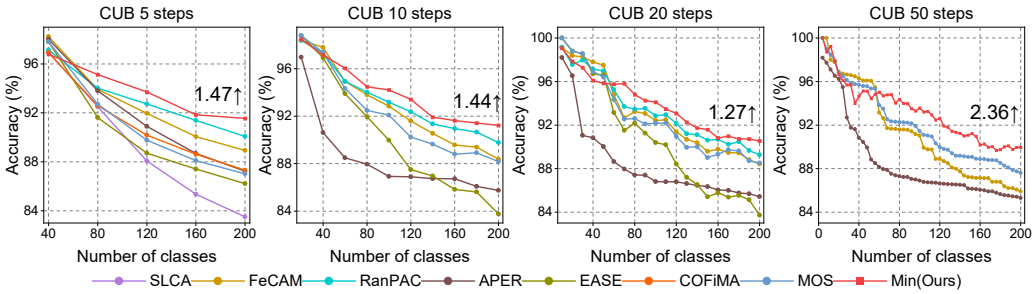

Figure 12: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the CUB dataset.

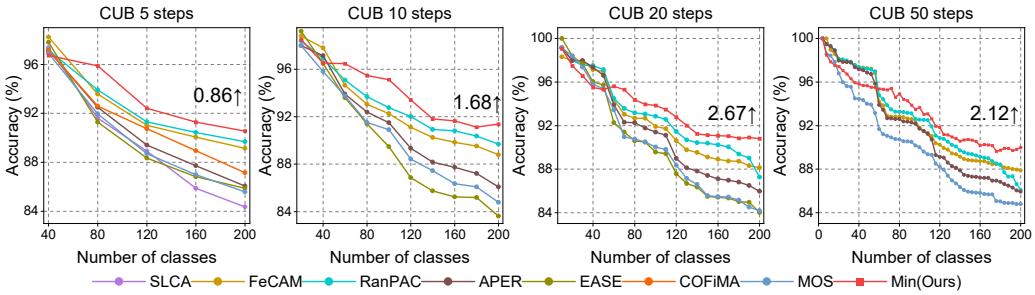

Figure 13: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the CUB dataset.

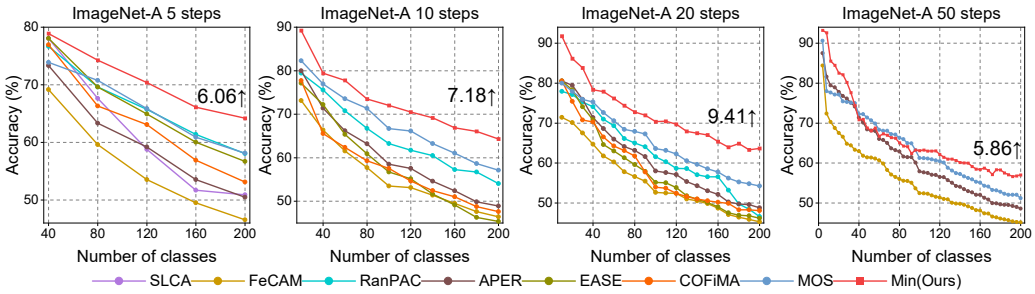

Figure 14: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the ImageNet-A dataset.

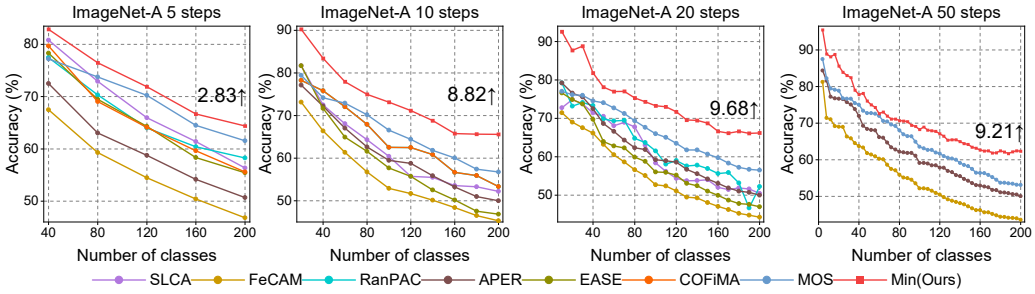

Figure 15: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the ImageNet-A dataset.

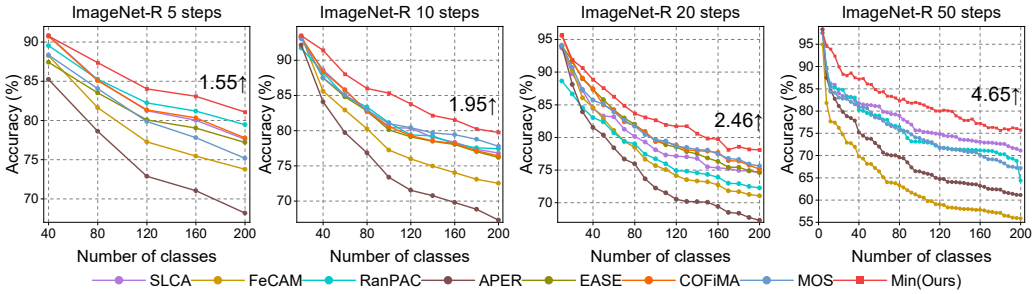

Figure 16: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the ImageNet-R dataset.

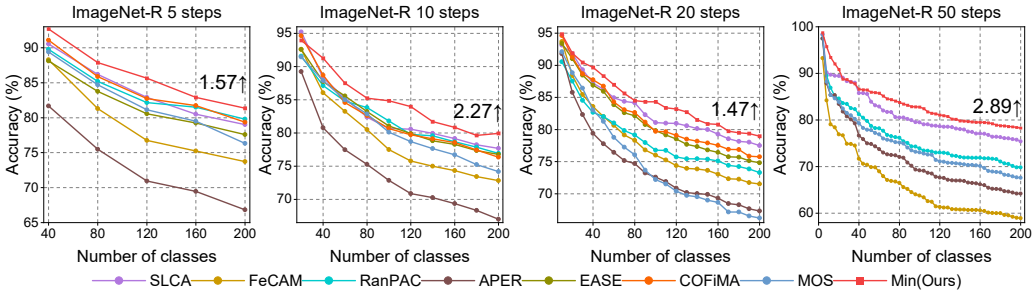

Figure 17: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the ImageNet-R dataset.

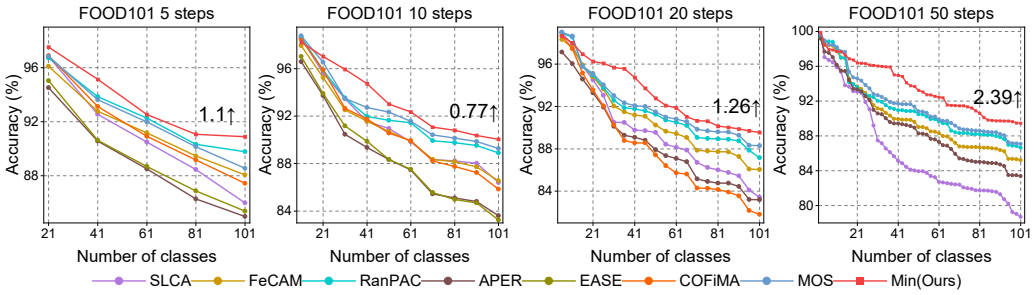

Figure 18: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the FOOD101 dataset.

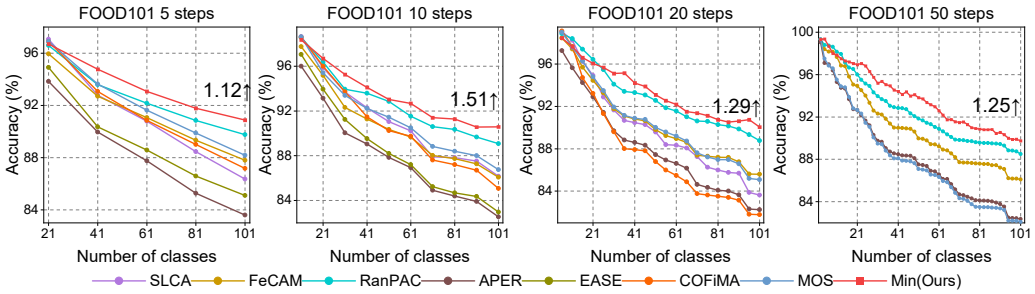

Figure 19: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the FOOD101 dataset.

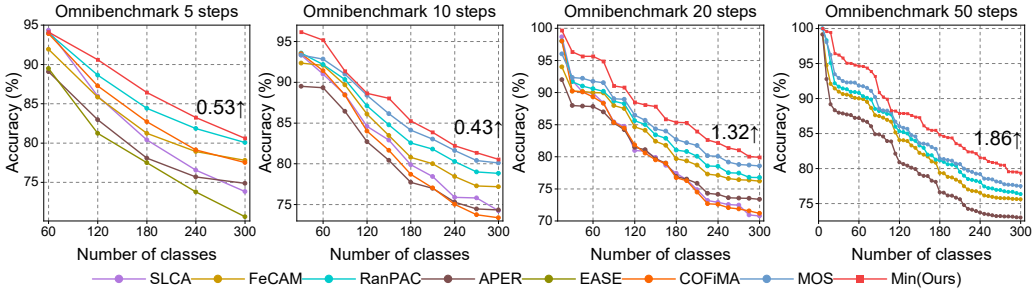

Figure 20: Incremnetal trends of different approaches using **ViT-B/16-IN21K** on the Omnibenchmark dataset.

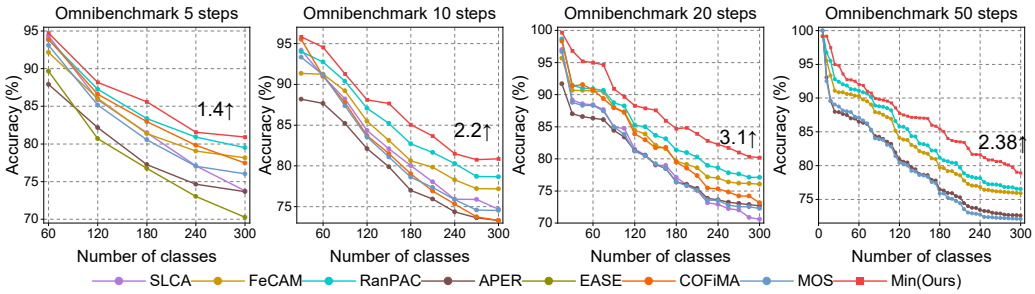

Figure 21: Incremnetal trends of different approaches using **ViT-B/16-IN1K** on the Omnibenchmark dataset.