ADAPTER SHARING VIA TASK-CONDITIONED FEATURE TRANSFORMATIONS FOR CLASS INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exemplar-free class-incremental learning requires a model to incrementally learn tasks that contain new classes without accessing earlier training data while preserving the ability to recognize all seen classes. Recently, pretrained-model-based approaches have become prevalent by adapting a frozen backbone with additional lightweight trainable modules. Existing methods, however, exhibit limitations. Task-specific adapters learn explicit per-task representations but are parameter- and computation-inefficient, while LoRA-based merging methods combine per-task LoRA updates into a single static model. Because the merged update must serve all tasks with the same parameters, the resulting static update may reduce the separability of task-specific feature distributions during inference. To address these problems, we present **FACET**: task-conditioned **F**e**A**ture transformation with **C**ondition**E**d feature consis**T**ency, achieving excellent parameter efficiency while producing highly discriminative features during inference. When continually trained on a task sequence, FACET shares core adapter weights across tasks and employs task-conditioned feature transformations to shape the overall feature distribution of the adapter into a mixture of task-specific components with reduced overlap. This design supports task-dependent features without allocating a complete adapter to every task. Additionally, we propose an efficient replay-free task-conditioned feature consistency loss to mitigate drift in previously learned task-conditioned features as the shared parameters evolve during continual training. At inference, lightweight auxiliary classifiers estimate the task context without requiring an oracle task identity. We demonstrate scalability to task sequences of up to 200 tasks. Across short and long task sequences, FACET achieves leading final accuracy compared to strong baselines while reducing inference cost. On OmniBenchmark-1K with 200 tasks, FACET achieves 36.64 ms latency and 40.23 GFLOPs per image, corresponding to a 3.85× speedup and a 62.69% reduction in GFLOPs compared with MIN. The code and models will be made open source upon acceptance.

1 INTRODUCTION

In real-world applications, training data may arrive sequentially from continuously changing environments, which requires a learning system to acquire new concepts while retaining previous knowledge (Lesort et al., 2020; Meng et al., 2025). However, standard deep neural networks generally suffer from catastrophic forgetting: when trained on new data, their performance on previously learned tasks can severely degrade (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Rusu et al., 2016). Continual learning (CL) aims to address this problem by enabling models to learn from continuous data streams without forgetting past knowledge. Among the many CL settings, class-incremental learning (CIL) is one of the most challenging because the model must learn new classes over time and, at test time, discriminate among all classes seen so far (Masana et al., 2022).

Traditional continual learning methods address catastrophic forgetting through regularization (Kirkpatrick et al., 2017; Li & Hoiem, 2017), replay (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017),architectural expansion (Rusu et al., 2016; Yoon et al., 2017), and parameter isolation/subnetworks (Aljundi et al., 2017; Mallya & Lazebnik, 2018). More recently, pretrained-model (PTM)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

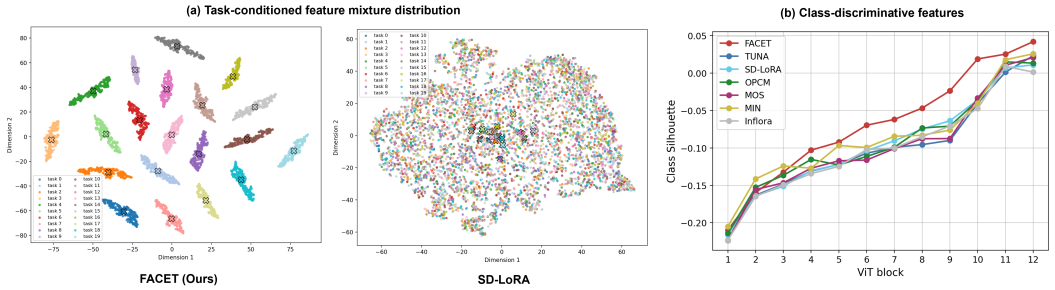

Figure 1: Comparison of feature distributions. (a) t-SNE of block 1 features, colored based on **task**. FACET shows well-separated task-specific distributions, while SD-LoRA's are mixed. (b) **Class**-averaged silhouette score (higher values mean better intra-class compactness and inter-class separation) across blocks. FACET achieves higher scores than representative LoRA-based (SD-LoRA) and per-task adapter based (TUNA) methods, producing more class-discriminative features that are beneficial for classification.

based approaches have become prevalent for CIL, as large pretrained backbones provide general-purpose, powerful representations, and require relatively few additional parameters when adapted to new tasks (Wang et al., 2022b; Smith et al., 2023; Zhou et al., 2024; 2025; Wang et al., 2025b; Sun et al., 2025b; Jiang et al., 2025). A common strategy attaches task-specific learnable adapters to a frozen vision backbone (Zhou et al., 2024; Wang et al., 2025b; Jiang et al., 2025; Wang et al., 2025a), which produces explicit task-specific feature perturbations but requires a separate adapter per task, resulting in parameter and computation inefficiency. Another popular solution trains separate LoRA modules (Hu et al., 2022) by task and merges all learned modules before inference, exemplified by InfLoRA (Liang & Li, 2024) and SD-LoRA (Wu et al., 2025). Such merging can introduce representation interference, a common issue observed in model-merging methods, where the resulting static parameters may compromise task-specific feature representations and reduce their separability during inference (Tang et al., 2025; Marczak et al., 2024).

To address these issues, we present **FACET**: task-conditioned **F**e**A**ture transformation with **C**ondition**E**d feature consis**T**ency. FACET adopts a shared adapter design in which the adapter modules inserted into the transformer blocks reuse their core projection weights across tasks. Rather than choosing between task-specific adapters and a static merged model, FACET uses the task context to dynamically reshape the latent feature space of adapter modules shared across tasks. The conditioned transformation rescales and reorients the shared latent representation, enabling different tasks to occupy distinct feature geometries while reusing the same core adapter weights. This design retains task-dependent representational capacity while avoiding the storage and inference cost of replicating a full adapter for every task.

Sharing the adapter weights also creates a continual-learning challenge: updating them for a new task can distort feature mappings conditioned under earlier task contexts. We propose a novel replay-free **task-conditioned feature consistency** loss to address this issue. Specifically, during training on the current task, the updated adapter and a frozen copy of the previous adapter process the same current-task images under a sampled historical task context. Matching their conditioned features constrains changes to the shared adapter that would drift features conditioned on historical tasks, thereby reducing drift in earlier task-conditioned feature mappings without storing historical examples.

We compare the representation quality of FACET with representative methods in Fig. 1. In early adapter blocks, FACET's task-specific feature distributions are widely separated, while those of SD-LoRA are completely mixed (Fig. 1 (a)). As CIL ultimately requires discrimination among classes, Fig. 1 (b) compares the block-wise class-averaged Silhouette score [1]. A higher score suggests more class-discriminative features. Driven by dynamic task conditioning, FACET achieves a substantially higher score in the last block, which indicates that features have transitioned from task-scaffolded to class-discriminative as depth increases.

[1]The Silhouette score of a class is $(b-a)/\max(a,b)$, where a is the mean pairwise feature distance within the class and b is the mean pairwise distance between any feature in the class and any feature in the nearest different class.

We evaluate FACET on multiple class-incremental learning benchmarks. FACET achieves the highest final accuracy on ImageNet-R, ImageNet-A, CIFAR-100, and ObjectNet. Compared with SD-LoRA (Wu et al., 2025), FACET improves the final accuracy from 75.26% to 79.97% on ImageNet-R with 20 tasks. Surprisingly, on OmniBenchmark-1K (Lou et al., 2026) with 200 tasks, FACET significantly improves over MIN (Jiang et al., 2025) and TUNA (Wang et al., 2025b) by 3.45% and 6.81% in final accuracy, respectively. FACET is also $3.85\times$ faster and reduces GFLOPs by 62.69% relative to MIN (Table 3). Note that MIN and TUNA are strong baselines that learn a task-specific adapter for every task.

Our main contributions are as follows:

Shared adapter with task-conditioned transformations. We introduce a shared adapter design with dynamically generated task-specific transformations, enabling task-specific features without replicating a complete adapter for every task.

Conditioned feature consistency. We propose a task-conditioned consistency loss that helps maintain historical task-conditioned features using only current-task inputs.

Scalable accuracy and efficiency. Across benchmarks with up to 200 tasks, FACET achieves leading final accuracy while reducing inference cost. On OmniBenchmark-1K with 200 tasks, it is $3.85\times$ faster and uses 62.69% fewer GFLOPs than MIN.

2 Related Work

Pre-trained model-based Continual Learning.

Early continual learning methods combat catastrophic forgetting through regularization Kirkpatrick et al. (2017); Zenke et al. (2017), knowledge distillation Li & Hoiem (2017), replay Rebuffi et al. (2017); Lopez-Paz & Ranzato (2017), parameter isolation or subnetworks Aljundi et al. (2017); Mallya & Lazebnik (2018), and architectural expansion Rusu et al. (2016); Yoon et al. (2017); Yan et al. (2021). More recently, pretrained-model-based approaches have become prevalent for CIL, as large pretrained backbones provide powerful representations while requiring relatively few additional parameters Wang et al. (2022b); Smith et al. (2023); Zhou et al. (2024; 2025); Wang et al. (2025b); Sun et al. (2025b); Jiang et al. (2025). Prompt-tuning methods such as L2P Wang et al. (2022b), DualPrompt Wang et al. (2022a), and CODA-Prompt Smith et al. (2023) store task-specific prompts, while adapter-based methods attach task-specific modules to a frozen backbone Zhou et al. (2024); Wang et al. (2025b); Jiang et al. (2025); Wang et al. (2025a). EASE Zhou et al. (2024) builds task-specific subspaces, while TUNA Wang et al. (2025b), MOS Sun et al. (2025b), and MIN Jiang et al. (2025) assign a dedicated adapter to each task. These methods provide explicit task-dependent representations but require increasing task-specific storage and computation. In contrast, FACET shares core adapter weights across tasks and uses the task context to generate task-conditioned transformations in the adapter's latent space.

Parameter Efficient CIL. Some methods improve parameter efficiency through classifier adaptation or model merging. SLCA (Zhang et al., 2023) and FeCAM (Goswami et al., 2023) focus on classifier alignment and covariance-aware classifiers, respectively. LoRA-based and model-merging methods, including InfLoRA Liang & Li (2024), SD-LoRA Wu et al. (2025), MagMax Marczak et al. (2024), and OPCM Tang et al. (2025), first learn task-specific LoRA/Adapter updates and then combine them into a single model for inference. Since all tasks use the same merged parameters, such methods may compromise task-specific feature representations and reduce their separability during inference (Tang et al., 2025; Marczak et al., 2024). FACET keeps task conditioning active during feature computation, allowing different tasks to use different transformations of the shared representation without storing a complete adapter copy for every task.

3 Method

3.1 Problem Formulation

In class-incremental learning, a model learns from a data stream $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$ consisting of T tasks arriving sequentially. Each task t provides a training set $\mathcal{D}_t = \{(x_i^t,\ y_i^t)\}_{i=1}^{n_t}$, where x_i^t is

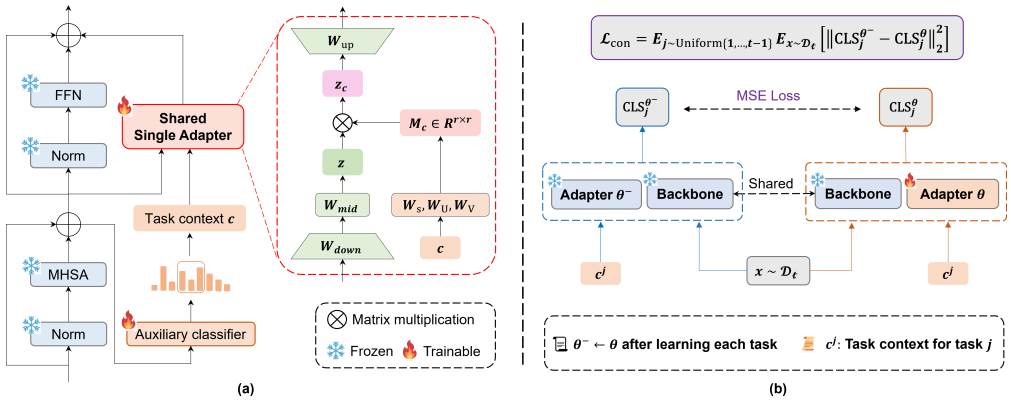

Figure 2: Illustration of FACET. **(a)** Task-conditioned feature transformation within a shared adapter. We predict the task context c and use it to generate a transformation matrix M_c to obtain task-dependent features. Activation functions are not shown for simplicity. **(b)** Task-conditioned feature consistency objective when training on a new task t with dataset $\mathcal{D}_t$.

an image, $y_i^t \in \mathcal{Y}_t$ is its class label, and n_t is the number of training samples. The label sets of different tasks are disjoint: $\mathcal{Y}_s \cap \mathcal{Y}_t = \emptyset$ for all $s \neq t$. We follow the exemplar-free setting (Zhou et al., 2024; Wang et al., 2025b). When task t is learned, only the latest training data $\mathcal{D}_t$ are accessible. We denote $\mathcal{Y}_{\text{new}} = \mathcal{Y}_t$ as the set of new classes and $\mathcal{Y}_{\text{old}} = \bigcup_{s=1}^{t-1} \mathcal{Y}_s$ as the set of previously seen classes. After being trained on task t, the model is evaluated on all classes seen so far, $\mathcal{Y}_{\text{old}} \cup \mathcal{Y}_{\text{new}}$.

We decompose the model into a pretrained ViT backbone $\mathcal{F}$, an adapter with parameters θ (Chen et al., 2022) inserted in parallel to the FFN in every transformer block, and a classifier $\mathbf{W}_t$ that grows by appending a new class vector for each task. The backbone $\mathcal{F}$ is frozen throughout training. Only θ and $\mathbf{W}_t$ are updated. The goal is to learn all T tasks from the data stream and correctly classify any test sample into one of the classes in $\bigcup_{t=1}^{T} \mathcal{Y}_t$ without access to past data.

3.2 Shared Adapter with Dynamic Task-Conditioned Transformations

As shown in Fig. 2 (a), a frozen pretrained ViT (Dosovitskiy et al., 2020) is equipped with adapter modules placed in parallel to the feed-forward network (FFN) in every transformer block. For an input x to the FFN, the output of the block is

$$o = \text{FFN}(x) + \text{Adapter}(x, c) + x, \tag{1}$$

where c denotes a context vector that reflects the task-related information of x. By conditioning the adapter's output explicitly on c, the shared adapter core produces task-dependent features during inference through context-conditioned latent transformations

We increase adapter capacity using an additional nonlinear hidden layer to compute the latent representation $z \in \mathbb{R}^r$:

$$z = \sigma\big(\sigma(xW_{\text{down}})\, W_{\text{mid}}\big), \qquad W_{\text{down}} \in \mathbb{R}^{d\times r},\ W_{\text{mid}} \in \mathbb{R}^{r\times r}, \tag{2}$$

where σ is the GELU (Hendrycks & Gimpel, 2016) activation, d is the ViT's feature dimensionality, and $r < d$ is the adapter's latent dimensionality. Next, before projecting the latent representation back to the ViT feature space, we apply a task-conditioned linear transformation to z,

$$z_c = zM_c, \tag{3}$$

where $M_c \in \mathbb{R}^{r\times r}$ is dynamically generated from c and z_c is the resulting task-conditioned representation. Different contexts therefore induce different transformations of the shared latent features, yielding distinct task-conditioned components in the overall feature distribution. The adapter output is obtained by projecting z_c up to the ViT feature space through the shared up-projection $W_{up} \in \mathbb{R}^{r\times d}$:

$$\text{Adapter}(x, c) = (z_c)W_{up}. \tag{4}$$

Because the up-projection is linear, transforming the latent representation z is sufficient to produce task-dependent adapter outputs.

We next describe the context representation and details of the task-conditioned transformation. The context vector reflects information on the task identity of the given input. A simple option is to set it to be the task identity itself, i.e., $c \in \{0,1\}^{K_{\max}}$[2]. During training, the task identity is known, while lightweight task-wise classifiers predict the context vector during inference (Sec. 3.4).

Given a context vector, we construct M_c using the following structured form to maintain parameter efficiency.

$$M_c = \mathrm{Diag}(\mathbf{s}) + UV^\top, \tag{5}$$

where $\mathbf{s} \in \mathbb{R}^r$ provides task-conditioned channel scaling, and $U, V \in \mathbb{R}^{r\times k}$ with $k \ll r$ implement cross-dimension mixing. These factors are generated from c through learnable linear projections.

$$\mathbf{s} = W_s c, \qquad U = \mathrm{reshape}(W_U c), \qquad V = \mathrm{reshape}(W_V c),$$

where $W_s \in \mathbb{R}^{r\times K_{\max}}$ and $W_U, W_V \in \mathbb{R}^{rk\times K_{\max}}$ are learnable projection matrices, and $\mathrm{reshape}(\cdot)$ converts a vector of length rk into a $r \times k$ matrix. Thus, the shared adapter weights produce task-dependent representations through transformations generated from the task context. Since $k \ll r < d$, M_c require substantially fewer parameters than duplicating the full adapter for every task. Unlike model merging, task conditioning remains active during inference, allowing different tasks to use different transformations of the shared latent representation.

To improve training stability, we initialize W_s, W_U, W_V to zero so that the adapter output is zero during the first training step. We also add small Gaussian noise to the context vector during training, aiming to improve robustness against imperfect task identity predictions during inference.

Remark Because the core adapter projection weights $W_{down}, W_{mid}, W_{up}$ are shared across tasks while M_c is task-conditioned, the adapter's output distribution can be viewed as a mixture of task-specific components. The task-conditioned transformation can reduce overlap among these components by changing their locations, as supported by Fig. 1. We also considered adding explicit losses to improve the separability of the mixture components, such as maximizing their cosine distance or maximizing the orthogonality of task-conditioned transformations. These additions reduce the adapter's learning capacity and lead to lower performance (see ablation studies Table 5).

Note that subnetwork methods use task identity to gate units or select computational paths (Aljundi et al., 2017; Abati et al., 2020; Masse et al., 2018). In contrast, FACET uses context to generate channel-wise scaling and low-rank cross-channel mixing within any shared adapter, enabling task-dependent transformations of its latent representation.

3.3 Task-Conditioned Feature Consistency

Continual updates when learning new tasks can shift the adapter's latent feature distribution z, on which the learned task-conditioned components depend, causing drift in the historical task-conditioned mixture components. We therefore introduce a task-conditioned feature consistency objective that stabilizes these feature mappings against parameter drift without storing historical data. This allows our adapter to produce reliable task-dependent features in a CL setting. Specifically, after learning task $t-1$, we save the adapter as a frozen copy $\theta^- \leftarrow \theta$, which remains fixed when learning task t. For each mini-batch, we uniformly sample the index of a historical task $j \sim \mathrm{Uniform}\{1,\dots,t-1\}$ and construct the corresponding context vector $c^j \in \mathbb{R}^{K_{\max}}$. Then the same input $x \in \mathcal{D}_t$ and context c^j are passed through two paths:

Main path: The pair (x, c^j) is processed by the frozen ViT backbone with the current adapter parameters θ. We denote the [CLS] token from the last transformer block by $\mathrm{CLS}_\theta(x, c^j)$.

Reference path: The same pair (x, c^j) is processed by the frozen ViT backbone with the adapter snapshot θ^-. We denote the resulting [CLS] token by $\mathrm{CLS}_{\theta^-}(x, c^j)$.

Evaluating both paths using the same (x,c^j) enables us to directly measure how much the current adapter has shifted the task-conditioned feature component associated with task j. This does not require x to be a sample from task j as the context vector c^j drives the adapter to apply the transformation associated with task j. Our experiments show that this simple strategy retains the learned

[2] $K_{\max}$ is a predefined sufficiently large number that serves as an upper bound on the number of tasks. Note that most prior works study CIL up to 20 tasks, while we set $K_{\max}$ to 200.

feature distributions of historical tasks without their original data (Appendix D) and substantially mitigates forgetting (Table 4).

The training loss is the squared Euclidean distance between conditioned [CLS] tokens averaged uniformly over all previous tasks:

$$\mathcal{L}_{\text{con}} = \mathbb{E}_{j\sim\text{Uniform}\{1,\ldots,t-1\}}\, \mathbb{E}_{x\sim\mathcal{D}_t}\left[\left\|\text{CLS}_j^{\theta^-} - \text{CLS}_j^{\theta}\right\|_2^2\right]. \tag{6}$$

In practice, each mini-batch samples one j and one current-task data batch, giving an unbiased stochastic estimate of this expectation and providing uniform coverage of all earlier tasks over the course of training. We sample only one previous task per iteration rather than considering all previous tasks because the latter would over-constrain the adapter and limit learning on task t. For the first task, $\mathcal{L}_{\text{con}}$ is set to zero.

The overall training loss is

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{con}}\, \mathcal{L}_{\text{con}}, \tag{7}$$

where $\mathcal{L}_{\text{cls}}$ is the weighted classification and classifier-alignment loss used in prior work (Zhang et al., 2023; Wang et al., 2025b; Sun et al., 2025b), and $\lambda_{\text{con}} = 2$ in our experiments. Unlike LwF (Li & Hoiem, 2017) and related distillation methods (Heo et al., 2019; Li et al., 2024), which typically align predictions or unconditional features, our objective performs feature distillation under each historical context and directly regularizes the context-conditioned feature mappings used by FACET.

3.4 Oracle-Free Inference

Generating task-dependent features in each block requires a task context vector c. We therefore introduce a straightforward strategy to estimate c during inference. Specifically, we attach a linear classifier to the output of the attention layer within each transformer block (Fig. 2). This auxiliary classifier receives the same supervision as the final classification head during the continual training phase. During inference, this classifier outputs a probability distribution p_t for all classes $\mathcal{Y}_t$ for all learned tasks $t \in \{1, ..., T\}$. We compute the entropy $H_t = -\sum_{y\in\mathcal{Y}_t} p_t(y) \log p_t(y)$ for each task and select the task context with the highest prediction certainty (lowest entropy) $\hat{t} = \arg\min_t H_t$. The final classifier predicts from the last block's [CLS] feature using the estimated context. Notably, the auxiliary classifier in each block contains task-specific parameters with a total of $(d \times \sum_t |\mathcal{Y}_t|)$ parameters, where $\sum_t |\mathcal{Y}_t|$ is the total number of classes in the dataset. Table 3 shows that FACET significantly reduces task-specific costs compared to existing task-specific adapter methods. The appendix provides ablations on alternative context-prediction strategies.

4 Experiments

We evaluate FACET on standard CIL benchmarks covering short to very long task sequences and show that FACET outperforms strong baselines, especially on long task sequences and the final accuracy metric. Ablation studies validate the effectiveness of all components.

4.1 Experimental Setting

Datasets. We benchmark on four widely used datasets, including CIFAR-100 (Krizhevsky et al., 2009), ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al., 2021b), ObjectNet (Barbu et al., 2019), and OmniBenchmark-1K (Lou et al., 2026). On OmniBenchmark-1K, we evaluate task sequences of up to 200 tasks. Together, these benchmarks cover short, medium-length, and long task sequences in standard and distribution-shifted visual domains.

Baselines. We compare FACET against state-of-the-art prompt-based and adapter-based PTM CIL methods with task-specific parameters (L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODA-Prompt (Smith et al., 2023), EASE (Zhou et al., 2024), APER (Zhou et al., 2025), SEMA (Wang et al., 2025a), TUNA (Wang et al., 2025b), MOS (Sun et al., 2025b), and MIN (Jiang et al., 2025)), classifier-based methods (SLCA (Zhang et al., 2023), FeCAM (Goswami et al., 2023)), and LoRA-based weight-merging methods (InfLoRA (Liang & Li, 2024), SD-LoRA (Wu et al., 2025), MagMax-LoRA (Marczak et al., 2024), and OPCM-LoRA (Tang et al., 2025)).

Table 1: Performance comparison with state-of-the-art methods on standard benchmarks.

Method	ImageNet-R 10 Tasks (Inc20)		ImageNet-R 20 Tasks (Inc10)		CIFAR-100 10 Tasks (Inc10)		CIFAR-100 20 Tasks (Inc5)		ImageNet-A 10 Tasks (Inc20)		ObjectNet 10 Tasks (Inc20)	
	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T
L2P$_{(CVPR'22)}$	75.46	69.77	63.75	55.78	85.92	79.19	85.94	79.93	49.39	41.71	66.77	55.16
DualPrompt$_{(ECCV'22)}$	73.10	67.18	66.52	61.77	89.65	84.89	87.87	81.15	53.71	41.67	64.31	52.99
CODA-Prompt$_{(CVPR'23)}$	77.97	72.27	70.45	64.68	91.05	86.44	89.11	81.96	53.54	42.73	66.53	56.80
EASE$_{(CVPR'24)}$	81.74	76.17	81.18	74.62	92.11	87.72	91.51	85.80	65.34	55.04	71.04	59.37
MOAL$_{(CVPR'25)}$	84.45	79.33	82.94	76.85	94.22	90.49	93.27	88.36	**74.29**	64.06	75.37	65.33
SEMA$_{(CVPR'25)}$	81.39	77.84	77.84	69.60	91.60	86.75	92.23	87.84	63.83	52.21	67.95	54.92
MOS$_{(AAAI'25)}$	81.74	76.17	82.96	77.93	93.83	90.19	93.30	89.25	69.13	59.12	74.75	65.10
TUNA$_{(ICCV'25)}$	84.22	79.42	82.54	77.45	94.85	91.71	94.44	90.74	72.96	64.38	76.46	66.32
MIN$_{(NeurIPS'25)}$	85.18	79.75	83.69	78.08	95.12	92.12	94.31	91.03	72.89	64.32	72.56	61.36
SLCA$_{(ICCV'23)}$	81.17	77.00	81.85	76.63	92.67	89.30	93.52	90.07	68.66	58.74	74.12	63.23
FeCAM$_{(NeurIPS'23)}$	79.02	72.53	77.84	71.05	93.23	89.05	91.86	87.04	56.04	46.41	68.68	57.39
APER-Adapter$_{(IJCV'25)}$	75.82	67.95	72.35	64.33	92.22	87.45	90.65	85.15	60.53	49.57	69.24	57.41
MagMax-LoRA$_{(ECCV'24)}$	84.59	77.68	81.8	73.40	94.76	91.41	94.31	90.94	66.64	54.64	70.85	59.15
OPCM-LoRA$_{(NeurIPS'25)}$	84.37	79.38	84.08	77.62	94.90	91.92	94.60	91.18	69.67	59.78	71.99	60.28
InfLoRA$_{(CVPR'24)}$	80.82	75.65	77.28	71.01	91.70	86.51	89.13	81.46	58.50	46.28	70.67	58.04
SD-LoRA$_{(ICLR'25)}$	82.04	77.34	80.22	75.26	92.54	88.01	90.90	85.18	64.95	55.96	70.37	58.54
FACET (Ours)	**85.41**	**80.66**	**84.73**	**79.97**	**95.18**	**92.27**	**94.62**	**91.31**	73.31	**64.87**	**77.13**	**67.38**

Table 2: Performance comparison on benchmarks with long task sequences. InfLoRA and SD-LoRA failed to converge on the 200 task setting.

Method	ImageNet-R 50 Tasks (Inc4)		ImageNet-A 50 Tasks (Inc4)		OmniBenchmark-1K 100 Tasks (Inc10)		OmniBenchmark-1K 200 Tasks (Inc5)	
	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T	$\bar{A}$	A_T
L2P$_{(CVPR'22)}$	69.16	63.45	49.89	36.41	60.91	48.87	57.53	45.25
DualPrompt$_{(ECCV'22)}$	64.00	56.33	43.85	29.95	62.18	49.45	59.13	45.62
CODA-Prompt$_{(CVPR'23)}$	62.43	57.57	38.24	26.60	64.16	51.75	60.22	47.56
EASE$_{(CVPR'24)}$	78.11	70.63	59.86	47.53	65.00	53.54	67.23	55.13
SEMA$_{(CVPR'25)}$	67.80	59.32	52.99	40.68	56.55	33.96	45.06	19.95
MoAL$_{(CVPR'25)}$	68.81	52.32	54.46	33.57	63.29	41.55	53.72	27.03
MOS$_{(AAAI'25)}$	75.15	66.80	63.72	51.22	76.80	64.27	75.92	63.51
TUNA$_{(ICCV'25)}$	80.93	75.48	67.82	58.72	71.93	60.04	71.45	59.14
MIN$_{(NeurIPS'25)}$	82.26	75.72	66.15	57.08	75.46	63.60	74.94	62.50
APER-Adapter$_{(IJCV'25)}$	72.43	64.83	61.29	48.58	73.23	62.24	72.49	61.53
MagMax-LoRA$_{(ECCV'24)}$	81.87	74.97	65.40	52.58	69.93	41.85	61.30	30.27
OPCM-LoRA$_{(NeurIPS'25)}$	82.49	73.70	66.58	54.02	60.19	33.73	58.32	28.43
InfLoRa$_{(CVPR'24)}$	71.68	62.62	50.17	38.70	51.53	27.01	-	-
SD-LoRA$_{(ICLR'25)}$	68.40	63.28	54.72	41.28	53.97	28.15	-	-
FACET (Ours)	**82.70**	**76.63**	**69.16**	**59.58**	**76.88**	**66.17**	**77.30**	**65.95**

Implementation Details.

We run all our experiments on a single NVIDIA L40S GPU. Following prior work (Zhou et al., 2024; 2025; Wang et al., 2025b; Jiang et al., 2025; Sun et al., 2025b), we employ the representative pre-trained vision backbone ViT-B/16-IN21K (Dosovitskiy et al., 2020). FACET adds a shared adapter per transformer block (latent dimension 384, hidden depth 2) with $k = 8$ and $K_{\max} = 200$ for the task-conditioned transformation matrix M_c. For training, we use SGD (momentum 0.9, weight decay 5×10^{-4}), cosine learning-rate decay with base learning rate 0.02 and batch size 16, and conditional feature consistency weight $\lambda_{\text{con}} = 2.0$. The classification head follows (Zhou et al., 2024; Wang et al., 2025b). More details about datasets and settings are in the Appendix.

Evaluation Protocol and Performance Metrics. We follow standard settings (Wang et al., 2022b; 2025b; Jiang et al., 2025) to partition a dataset into T tasks with the learning order determined by the random seed 1993. We denote Inc n as each task having n classes. For FACET, we report the mean performance over 3 runs. We report the average incremental accuracy $\bar{A} = \frac{1}{T}\sum_{t=1}^{T} A_t$ and final accuracy A_T, where A_t is the average accuracy over all classes seen so far after the adapter has been trained on task t. Due to space limit, results on the ViT-B/16-IN1k backbone and other random seeds are provided in the Appendix B.2.

4.2 Main Results

Tables 1 and 2 compare FACET with representative PTM-based CIL baselines. FACET achieves superior final accuracy on standard benchmarks compared to computationally heavy methods based on task-specific adapters. Additionally, FACET surprisingly achieves leading performance on long task sequences of up to 200 tasks when compared to other strong baselines.

Table 3: Comparison of parameter, computation, and GFLOPs among SOTA methods. Latency is the median latency at batch size 32.

ImageNet-R (50 tasks)

Method	Params(M)	Latency(ms)	GFLOPs	$\bar{A}$	A_T
Ours	**10.20**	**28.61**	**37.37**	**82.70**	**76.63**
MoS	15.37	1431.30	1798.57	75.15	66.80
TUNA	15.37	1308.28	1798.05	80.93	75.48
MiN	135.15	44.02	55.39	82.26	75.72

OmniBenchmark-1K (200 tasks)

Method	Params(M)	Latency(ms)	GFLOPs	$\bar{A}$	A_T
Ours	**22.88**	**36.64**	**40.23**	**77.30**	**65.95**
MoS	61.63	5717.77	7088.81	75.92	63.51
TUNA	61.63	6306.79	7086.94	71.45	59.14
MiN	1831.59	141.42	107.84	74.94	62.50

Comparison with State-of-the-Art Methods. Table 1 shows that FACET achieves the highest final accuracy A_T in every reported standard setting. For instance, on ImageNet-R with 20 tasks, FACET improves A_T from 78.08% to 79.97% compared to MIN, one of the strongest task-specific expansion baselines. On the ImageNet-A benchmark, where adversarially filtered images make class discrimination more difficult, TUNA and MIN substantially outperform the strongest LoRA-based merging method, OPCM-LoRA, by 4.60% and 4.54% in absolute A_T, respectively, supporting the importance of retaining task-dependent representations under distribution shift. Nevertheless, FACET achieves the highest A_T among all methods (64.87%) and a competitive $\bar{A}$ of 73.31%. We note that A_T is more important than $\bar{A}$ in CIL (Jiang et al., 2025) because it evaluates the final model on all learned classes, whereas $\bar{A}$ also averages intermediate evaluations over smaller class sets. On ImageNet-R with 20 tasks, FACET also improves A_T over OPCM-LoRA by 2.35%. Together, these results show that the shared adapter retains strong task-dependent representations while avoiding inefficient use of task-specific adapter.

Scaling to Long Task Sequences. Table 2 evaluates whether FACET remains effective as the number of tasks in a sequence increases to 100 or even 200, when storing, searching over, or merging many task-specific modules becomes increasingly difficult. On ImageNet-R with 50 tasks, FACET achieves the highest $\bar{A}$ and A_T, improving A_T from 75.72% to 76.63% over MIN and exceeding MagMax-LoRA, the strongest merging baseline in A_T, by 1.66%. On ImageNet-A with 50 tasks, FACET again achieves the highest $\bar{A}$ and A_T, improving over TUNA by 1.34% in $\bar{A}$ and 0.86% in A_T. Its advantage over OPCM-LoRA reaches 5.56% in A_T, demonstrating strong performance under distribution shift while retaining shared adapter weights.

The scaling advantage becomes clearer on OmniBenchmark-1K. With 100 tasks, FACET improves over the strongest baseline by 1.90% in A_T, while achieving the highest $\bar{A}$ of 76.88. With 200 tasks, FACET achieves the highest reported accuracy, improving $\bar{A}$ from 75.92% to 77.30% and A_T from 63.51% to 65.95% over MOS. Doubling the task count reduces FACET's A_T by only 0.22%, while its lead over MOS widens from 1.9% to 2.44%. We note that task-specific adapters can mitigate forgetting, but routing errors increase with the number of tasks. FACET avoids complete per-task adapter allocation and remains robust to context-prediction errors, as shown in the AppendixC.

4.3 Parameter, Computation and GFLOPs

Table 3 compares our method with the best-performing baselines (i.e., MIN, MOS, and TUNA) in trainable parameter count, inference latency, and GFLOPs per image on long task sequences. Our method is among the most efficient and achieves a $156\times$ speedup in inference latency and only use 0.57% GFLOPs compared to MoS when there are 200 tasks. Furthermore, FACET requires only 1.25% of the cumulative trainable parameters of MIN. These findings demonstrate that our proposed method maintains leading performance while delivering better parameter efficiency and substantially lower latency than per-task adapter methods such as TUNA/MOS.

4.4 Ablation Studies

The baseline is our full FACET method presented in Table 1.

We extensively verify the effectiveness of all components in FACET with ablation studies

Table 4: Incremental component ablation on ImageNet-R (Inc10).

Variant	M_c cond	Consistency	Other designs	$\bar{\mathbf{A}}$	$\mathbf{A}_T$
Plain Single Adapter	×	×	×	57.75	48.75
+ M_c Transformation	✓	×	×	70.06	60.02
+ Cond. Feat. Consis.	✓	✓	×	83.90	78.23
FACET (Full)	✓	✓	✓	84.70	79.97
Feat. Consis. Only	×	✓	✓	72.94	65.39

conducted on the ImageNet-R Inc10 (20 tasks) dataset, which provides a challenging setting. We note that additional ablation and analysis on task context prediction and robustness of FACET to task context prediction errors are given in the Appendix.

Individual contributions of components. Table 4 decomposes the gains of FACET. A plain single adapter reaches only 48.75% final accuracy, confirming that simply sharing weights across tasks is insufficient. Adding task-conditioned feature transformations (M_c transformation) improves final accuracy to 60.02%, demonstrating that explicit task conditioning and conditioned transformations yield a substantial benefit by making task-specific feature distributions more separable. The performance jump to 78.23% after adding task-conditioned feature consistency, showing that once FACET creates separable mixture components, they are fragile to forgetting and must be explicitly preserved. Note that non-conditioned feature consistency alone achieves an A_T of 65.39%, indicating that the interplay between conditioned feature transformation and consistency is crucial. Finally, other designs like adding gaussian noise during training together further improve A_T to 79.97%.

Variants of conditioned feature transformation. Table 5 studies alternative forms of task-conditioned transformation and the effect of an explicit separability loss (Sec. 3.2), reporting $\bar{A}$, A_T, and the silhouette scores computed from last-block features as a measure of class discriminability. Transforming the adapter input instead of its output causes a sharp drop of A_T to 69.22% and a negative silhouette score, suggesting that intra-class distances exceed inter-class distances. This confirms that conditioning the adapter output via Eq. equation 3 is essential. Diagonal-only and low-rank-only variants of M_c, already raise performance clearly above the plain single adapter, showing that the primary gain comes from explicit task conditioning. Finally, adding explicit losses to further separate task-specific feature distributions hurts performance since it reduces the learning plasticity for new tasks.

Table 5: Ablation on conditioned feature transformation and separability loss.

Form	$\bar{\mathbf{A}}$	$\mathbf{A}_T$	Silhouette
No transformation	72.49	65.39	-0.11
Concat c to adapter input	69.46	66.42	-0.086
M_c at adapter input	72.27	69.22	-0.057
$M_c = UV^\top$	84.50	78.58	0.026
$M_c = \text{Diag}(s)$	84.46	79.60	0.040
$M_c = \text{Diag}(s) + UV^\top$	84.70	79.97	0.042
+Orthogonal M_c loss	82.19	78.63	0.035
$+z_c$ cos distance loss	81.83	78.02	0.014

Design of conditioned feature consistency. Table 6 studies how to best preserve previously learned task-specific feature distributions. Only enforcing consistency for the most recent task under-protects earlier task-specific distributions ($A_T = 77.03\%$), while considering all previous tasks simultaneously over-constrains the adapter and limits learning plasticity ($A_T = 78.37\%$). Uniformly sampling one previous task per mini-batch achieves the best trade-off, covering all prior task-specific distributions without hindering new task learning. A hard adapter snapshot outperforms an EMA teacher, indicating that the exact post-training adapter state is a more reliable reference. Alternative designs in computing the loss lead to suboptimal results, such as using CLS tokens from every block (rather than just the last block), all tokens from the last block, and using cosine distance (1-cosine similarity) in Eq. 6 instead of MSE.

Table 6: Ablation on conditioned feature consistency.

Strategy	$\bar{\mathbf{A}}$	$\mathbf{A}_T$
Most recent ($t-1$)	83.31	77.03
All tasks simultaneous	84.01	78.37
Uniform, EMA teacher	83.96	78.85
Uniform, snapshot (ours)	84.70	79.97
All features (Last Block)	80.56	70.70
CLS in every block	84.35	79.80
Cosine Distance	84.53	79.46

5 Conclusion

This work has introduced FACET, a shared adapter approach for pretrained-model-based class-incremental learning. By transforming the adapter's latent representation in a task-conditioned manner, FACET makes continual learning more efficient and scalable. However, although we achieve strong performance up to 200 tasks, longer task sequences or highly heterogeneous tasks may eventually exceed what one shared adapter can represent. In future work, we aim to extend our method to true lifelong continual learning.

AI USE

In this work, we used generative AI tools to improve the clarity of the paper by fixing grammar mistakes and typos. We did not use generative AI tools for any other tasks requiring disclosure. We reviewed all AI-assisted work. We take responsibility for the final content of this work, including all text, claims, code, and conclusions produced with the aid of generative AI.

REPRODUCIBILITY

We provide detailed implementation details required to reproduce our work in the main paper as well as Appendix Implementation Details. Code and model checkpoints will be open-sourced upon acceptance.

REFERENCES

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3931–3940, 2020.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3366–3375, 2017.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

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.

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.

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.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, 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, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021b.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1921–1930, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Kai Jiang, Zhengyan Shi, Dell Zhang, Hongyuan Zhang, and Xuelong Li. Mixture of noise for pre-trained model-based class-incremental learning. *Advances in neural information processing systems*, 39, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

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

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

Songze Li, Tonghua Su, Xu-Yao Zhang, and Zhongjie Wang. Continual learning with knowledge distillation: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6): 9798–9818, 2024.

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

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23638–23647, 2024.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Meng Lou, Yunxiang Fu, and Yizhou Yu. Scaling continual learning to 300+ tasks with bi-level routing mixture-of-experts. In *Forty-third International Conference on Machine Learning*, 2026.

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

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pp. 379–395. Springer, 2024.

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.

Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

Yuan Meng, Zhenshan Bing, Xiangtong Yao, Kejia Chen, Kai Huang, Yang Gao, Fuchun Sun, and Alois Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, 7(2):256–269, 2025.

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

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

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*, pp. 11909–11919, 2023.

Hai-Long Sun, Da-Wei Zhou, De-Chuan Zhan, and Han-Jia Ye. Pilot: A pre-trained model-based continual learning toolbox. *SCIENCE CHINA Information Sciences*, 68(4):147101, 2025a. doi: https://doi.org/10.1007/s11432-024-4276-4.

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*, volume 39, pp. 20699–20707, 2025b.

An Quang Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Merging models on the fly without retraining: A sequential approach to scalable continual model merging. *Advances in neural information processing systems*, 39, 2025.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10087–10098, 2025a.

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 806–816, 2025b.

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*, pp. 631–648. Springer, 2022a.

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*, pp. 139–149, 2022b.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, 2025. URL https://api.semanticscholar.org/CorpusID:275820710.

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*, pp. 3014–3023, 2021.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

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*, pp. 19148–19158, 2023.

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*, pp. 594–611. Springer, 2022.

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*, pp. 23554–23564, 2024.

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*, 133(3):1012–1032, 2025.

A Additional Dataset and Implementation Details

A.1 Datasets

We evaluate FACET on five datasets, including ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al., 2021b), CIFAR100 (Krizhevsky et al., 2009), ObjectNet (Barbu et al., 2019; Zhou et al., 2025) , and OmniBenchmark-1K (Zhang et al., 2022; Lou et al., 2026). The datasets used in the main comparison are summarized below:

ImageNet-R (Hendrycks et al., 2021a) has 200 classes with 30000 images. It consists of ImageNet-compatible classes rendered in artistic and non-standard visual styles.

ImageNet-A (Hendrycks et al., 2021b) contains 200 classes with 7475 images. The dataset consists of naturally adversarial images with ImageNet-compatible categories.

CIFAR100 (Krizhevsky et al., 2009) is a standard image classification benchmark that contains 100 classes and 60000 images.

Following (Wang et al., 2025b; Sun et al., 2025b; Zhou et al., 2025), the **ObjectNet** (Barbu et al., 2019; Zhou et al., 2025) dataset we refer to for CIL contains 200 classes with 33137 images. It a subset of the original ObjectNet (Barbu et al., 2019) dataset that contains 50000 images of 313 distinct objects with different backgrounds, rotations, and imaging viewpoints.

OmniBenchmark-1K (Zhang et al., 2022; Lou et al., 2026) has 188569 images and 1000 classes with non-overlapping data with ImageNet-1k. It is a subset of the clean version of OmniBenchmark (Zhang et al., 2022) with at least 100 images per class.

A.2 Implementation Details

We use the PILOT (Sun et al., 2025a) codebase to run our experiments. Following (Wang et al., 2022b;a; Zhou et al., 2024; Sun et al., 2025b; Zhou et al., 2025; Wang et al., 2025b; Jiang et al., 2025; Liang & Li, 2024; Wu et al., 2025), our main experiments use the ViT-B/16 backbone pre-trained on ImageNet-21k. FACET inserts one task-conditioned adapter shared across all tasks in each transformer block, with two 384-dimensional hidden layers. The linear projections for generating context vectors (W_s, W_U, W_V) are initialized to zeros for the first step of training. The auxiliary classifier is a linear layer that is trained to predict class labels directly. A zero-mean Gaussian noise with a standard deviation of 0.3 is added to the context vector during training to improve its robustness against imperfect context predictions during inference. Optimization uses SGD with a cosine learning-rate schedule, a weight decay of 5×10^{-4}, a batch size of 16, a learning rate of 0.02, and 20 training epochs. Classifier alignment following (Wang et al., 2025b) is included during training. For the short task sequence benchmarks with 10 to 20 tasks, we ensemble the predictions when using task context and when disabling the adapter by setting task context to zero. For long task sequences, predictions use only the FACET probabilities from a single backbone pass.

B Additional Experiments

B.1 Task Context Prediction.

Table 7 investigates alternative methods to predict task context during inference. Assembling the context vector using the highest logit in each task from the final classification head results in sub-optimal performance, while a context with the highest logits from auxiliary classifiers sees a 1.34% drop in A_T, demonstrating the advantage of entropy for task identity prediction. Directly filling the context using the entropy value for each task leads to a larger decline of A_T to 77.85%, confirming that the use of sparse context vectors is beneficial. Notably, using ground truth task context during inference improves final accuracy to 81.62%, implying that FACET can benefit from more accurate task context predictors for short task sequences.

Table 7: Ablation on context prediction.

Predictor	$\bar{\mathbf{A}}$	$\mathbf{A}_T$
Final classifier highest logits	84.07	79.33
Aux. classifier highest logits	84.19	78.63
Entropy as task context	84.23	77.85
Min entropy	84.70	79.97
Oracle	-	81.62

Table 8: Performance comparison on ImageNet-R (Inc10) using ViT-B/16-IN21k backbone and 5 random seeds. Metric is $\bar{A}$ / A_T.

Method \ Seed	49	127	982	1523	2026	Mean ± Std
SD-LoRA	78.50 / 70.58	77.27 / 70.73	77.82 / 71.85	76.58 / 69.95	77.22 / 70.77	77.48 ± 0.72 / 70.78 ± 0.68
TUNA	82.51 / 77.78	81.55 / 76.97	82.37 / 77.73	82.44 / 77.68	82.44 / 77.15	82.26 ± 0.40 / 77.46 ± 0.37
MIN	84.41 / 78.90	84.46 / 77.80	85.22 / 79.40	84.38 / 78.10	83.65 / 77.43	84.42 ± 0.56 / 78.33 ± 0.81
FACET	**85.63 / 80.80**	**85.14 / 79.32**	**85.89 / 80.98**	**84.66 / 79.52**	**84.77 / 79.28**	**85.22 ± 0.53 / 79.98 ± 0.84**

Table 9: Performance comparison on ImageNet-R (Inc10) using ViT-B/16-IN1k backbone and 5 random seeds. Metric is $\bar{A}$ / A_T.

Method \ Seed	49	127	982	1523	2026	Mean ± Std
SD-LoRA	81.40 / 74.65	81.19 / 75.72	80.45 / 74.77	80.03 / 74.75	79.82 / 73.32	80.58 ± 0.70 / 74.64 ± 0.86
TUNA	83.98 / 78.53	83.37 / 77.97	84.08 / 78.78	83.36 / 78.31	83.45 / 78.22	83.65 ± 0.35 / 78.36 ± 0.31
MIN	85.01 / 79.35	84.26 / 77.95	85.75 / 80.03	83.73 / 78.47	83.86 / 78.28	84.52 ± 0.85 / 78.82 ± 0.85
FACET	**86.05 / 80.87**	**84.82 / 79.90**	**86.65 / 80.88**	**86.68 / 80.67**	**85.50 / 80.07**	**85.94 ± 0.79 / 80.48 ± 0.46**

B.2 Robustness to Task Order and Pretrained Backbone

To evaluate the robustness and generalizability of our approach, we shuffle the task order with 5 random seeds and experiment with two widely used pretrained backbones, ViT-B/16-IN21k and ViT-B/16-IN1k (Dosovitskiy et al., 2020). Tables 8-9 present our results. FACET outperforms strong baselines in all task orders on both backbones. This indicates that the advantage of FACET is consistent rather than an artifact of a particular task order.

B.3 Detailed Ablations on Single Adapter Design

Random initialization starts each new task with a random transformation of the adapter feature space and reduces A_T to 76.68%. Zero initialization avoids this unstable start by making a newly activated task-conditioned mixture component initially fall back to the pretrained representation, improving A_T to 78.80%. Adding noise to the context vector further improves A_T to 79.97%, supporting its role in making the model robust to imperfect task identity prediction during inference.

Table 10: Impact of initialization and noise on ImageNet-R (Inc10).

Config.	$\bar{\mathbf{A}}$	$\mathbf{A}_T$
Random, no noise	82.74	76.68
Zero, no noise	83.95	78.80
Zero, +noise	84.70	79.97

Finally, Table 12 varies the bottleneck dimension and adapter depth. The results are stable across a range of capacities, and deeper adapters do not consistently improve performance, indicating that FACET's gains are not simply due to adding more adapter parameters.

As shown in Table 11, FACET is robust to different values of $K_{\max}$, with smaller values achieving better performance for ImageNet-R (Inc10) with 20 tasks. Since most prior works study 20 tasks, we select 200 to be a sufficiently large upper bound on the number of tasks in CIL.

Table 11: Effect of $K_{\max}$ on ImageNet-R (Inc10).

Metric \ Dim	50	100	200	400
$\mathbf{A_T}$	80.20	80.14	79.97	79.64
$\bar{\mathbf{A}}$	84.77	84.73	84.70	84.44

Table 12: Ablation of adapter capacity on ImageNet-R (Inc10). Metric is ($\mathbf{A}_T$).

Hidden depth \ Dim	16	64	256	384
1	79.08	79.10	79.38	79.49
2	79.39	79.30	79.57	**79.97**
3	79.37	78.97	79.62	79.15

Table 13: Impact of the consistency weight λ_{con} on ImageNet-R (Inc10).

Metric	0.5	1.0	2.0	4.0	8.0
$\bar{\mathbf{A}}$	83.94	84.75	84.70	84.34	84.13
$\mathbf{A}_T$	77.12	79.28	79.97	78.72	78.50

B.4 Impact of the Coefficient for Task-Dependent Feature Consistency Loss

Table 13 shows that $\lambda_{\text{con}} = 2.0$ achieves the highest final accuracy, achieving an optimal balance between mitigating forgetting of task-specific feature distributions and learning new task.

C Analysis of Task Prediction Accuracy

We further evaluate the robustness of FACET to wrong task-context predictions during inference. Without retraining, we compare with a randomly assigned context and a deliberately incorrect context on the same checkpoint. Random and forced-wrong results are mean $\pm$ sample standard deviation over five tries.

Table 14: Robustness of FACET to wrong task-context predictions at inference (same checkpoint, no retraining). Random and forced-wrong results are mean $\pm$ sample standard deviation over five tries.

Setting	Task-context pred. acc. (random choice)	Final acc. w/ predicted ctx.	Final acc. w/ oracle ctx.	Final acc. w/ random ctx.	Final acc. w/ forced-wrong ctx.
ImageNet-R Inc20	53.83% (10%)	80.66	86.38	78.07 ± 0.19	77.54 ± 0.15
ImageNet-R Inc10	49.05% (5%)	79.97	81.62	75.80 ± 0.24	75.49 ± 0.28
ImageNet-R Inc4	15.21% (2%)	76.63	78.72	76.59 ± 0.13	76.47 ± 0.09

Task prediction accuracy is 5.3 to 9.8 times above random guesses. More importantly, using an incorrect context reduces final accuracy by at most 4.48 (79.97 to 75.49 for ImageNet-R 20 tasks). We note that 75.49 is comparable to the performance of the strong baseline SD-LoRA (75.26). This shows that FACET is robust to wrong task-context predictions during inference. We note that providing ground-truth task context during inference is beneficial (ablation Table 7), and can improve final accuracy from 79.97 to 81.62 on ImageNet-R 20 tasks. Therefore, the performance of FACET is not only positively correlated with task-context prediction accuracy, but also robust against wrong task-context predictions. More advanced mechanisms to predict task context can further improve performance.

D Additional Analysis of Task-Conditioned Feature Consistency

Here we evaluate how well our conditioned feature consistency loss preserves the feature distributions of historical tasks. Specifically, we measure how well the sample features from the first task are preserved after learning new tasks. The CLS token from the last transformer block is selected as the feature representation since it is passed to the classifier for prediction. Note that the task-conditioned feature consistency loss is computed using data from task t ($x \in \mathcal{D}_t$), which is the latest task the adapter is being trained on.

Results are shown in Fig. 3, where the solid line denotes the mean distance and the light shaded area illustrates the standard deviation. It can be seen that our conditioned feature consistency loss (blue line) effectively preserves the original features from the first task, even after the adapter has been trained on 99 subsequent tasks from the OmniBenchmark-1K dataset. Without our conditioned feature consistency loss (orange line), the MSE and Cosine distances are much larger, which indicates forgetting of the task-specific distribution. For instance, after task 100, the mean cosine distance and mean squared error when using the consistency loss are 0.0588 and 0.0866, respectively. Without the consistency loss, cosine distance and MSE increases to 0.249 (+323%) and 0.427 (+393%), respectively. These results confirm the effectiveness of the conditioned feature consistency loss in preserving task-specific feature distributions in continual learning even though it relies on data from the latest task instead of those from the original tasks.

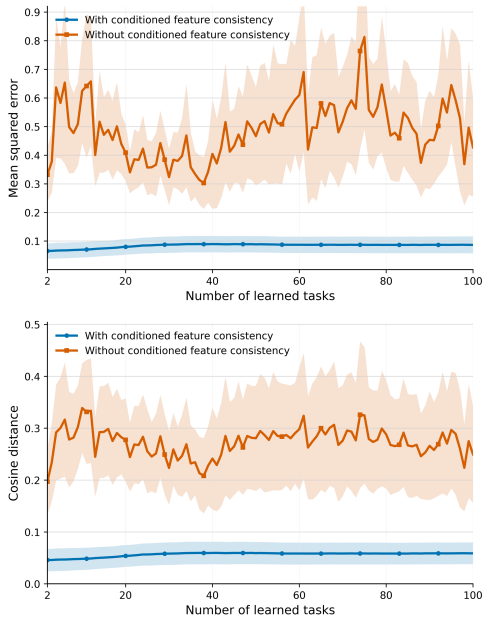

Figure 3: How well our single adapter preserves the feature distribution of the first task as it is sequentially trained on new tasks from task 2 to 100. **Top**: Mean square error as the distance metric. **Bottom**: Cosine distance as the distance metric. We compute the distance between the CLS feature produced by the model after it has learned the first task with the corresponding CLS feature after the model has sequentially learned $t \in \{2, ..., 100\}$ tasks. The original training data and context for the first task are used for a direct comparison.

We provide more details on how cosine distance and MSE are computed below: We first extract the CLS features of the data samples from the first task using the adapter trained on its dataset $\mathcal{D}_1$. This defines the original feature distribution of the first task. Then, after learning t tasks, we extract CLS features using the same data samples from $\mathcal{D}_1$ and the same context vector for the first task, but using the adapter sequentially trained on all t tasks. We evaluate $t \in \{2, ..., 100\}$ using the OmniBenchmark-1K dataset. Because the input images and context vectors are the same, any difference between corresponding CLS features directly measures how much an original feature from the first task has drifted after learning later tasks.

We quantify this drift using cosine distance and mean squared error (MSE). For each training image $x \in \mathcal{D}_1$, cosine distance is computed between the original CLS feature and the corresponding CLS feature extracted after learning t tasks:

$$d_{\cos} = 1 - \frac{\langle \mathrm{CLS}_1^{\theta_1}, \mathrm{CLS}_1^{\theta_t} \rangle}{\|\mathrm{CLS}_1^{\theta_1}\|_2 \|\mathrm{CLS}_1^{\theta_t}\|_2},$$

where $\mathrm{CLS}_1^{\theta_1}$ is the original CLS feature from the model trained on the first task, and $\mathrm{CLS}_1^{\theta_t}$ is the CLS feature from the model after learning t tasks using the same context. We average the cosine distance over all data samples. We also compute the MSE distance:

$$d_{\mathrm{MSE}} = \frac{1}{N} \left\| \mathrm{CLS}_1^{\theta_1} - \mathrm{CLS}_1^{\theta_t} \right\|_2^2,$$

where N is the number of images in the first task. These two metrics are complementary: cosine distance captures directional drift, while MSE captures overall feature-space displacement.

We also directly test whether feature drift is related to old-task classification forgetting. On ImageNet-R Inc10, we train six models that differ only in λ_{con}. After task 20, we compare the CLS features of the task-1 images with their features immediately after task 1, always using the task-1 context, and measure task-1 forgetting.

Across the six settings, cosine distance and MSE have Pearson correlations of 0.752 and 0.711 with task-1 forgetting. Greater drift is strongly associated with greater old-task classification forgetting.

Table 15: Task-1 forgetting versus CLS feature drift under different λ_{con} on ImageNet-R Inc10.

λ_{con}	Task-1 forgetting	Cosine distance	MSE
0	35.96	0.2376	0.5072
0.5	36.59	0.0888	0.1578
1	31.23	0.0538	0.0961
2	18.61	0.0285	0.0502
4	15.46	0.0167	0.0296
8	13.32	0.0113	0.0206

E JUSTIFICATION OF USING CURRENT-TASK DATA FOR CONDITIONED FEATURE CONSISTENCY

In Equation 6, we use the data from the current task D_t to preserve the old feature distributions. Here we provide a justification on why it works.

Current-task images are probe inputs used to detect whether new-task training changes the shared adapter under a previous context. They are not treated as approximations of old-task images in the consistency loss.

After task $t-1$, we freeze a copy of the adapter. While learning task t, a current image is processed by the frozen and updated adapters under the same sampled old context c_j. Because the input and context are identical, their feature difference isolates the effect of the adapter update. Minimizing this difference therefore constrains new-task training from changing the adapter's behavior under c_j.

The backbone is frozen and all tasks use the same low-dimensional adapter. Consequently, current probes can protect old features when the same adapter weights affect both current and old features. In that case, a weight change that damages old features also changes the current probe features and is penalized by the consistency loss. This does not require the tasks to have identical images, labels, or feature distributions.

Local Justification. Let $\Delta\theta = \theta - \theta^-$ be one adapter update and let $h_\theta(x, c_j)$ be the conditioned feature. For a small update,

$$h_\theta(x, c_j) - h_{\theta^-}(x, c_j) \approx J_j(x)\Delta\theta, \tag{8}$$

where $J_j(x)$ is the feature Jacobian with respect to the shared-adapter parameters. The current-input consistency loss and old-input feature drift are locally

$$\mathcal{L}^j_{\text{con}} \approx \Delta\theta^\top G^j_t \Delta\theta, \qquad D_j \approx \Delta\theta^\top G^j_j \Delta\theta, \tag{9}$$

where

$$G^j_a = \mathbb{E}_{x\sim\mathcal{D}_a}\left[J_j(x)^\top J_j(x)\right]. \tag{10}$$

If every small adapter change that affects old features is also detected by the current probes, up to a constant κ,

$$G^j_j \preceq \kappa G^j_t, \tag{11}$$

then

$$D_j \le \kappa \mathcal{L}^j_{\text{con}}. \tag{12}$$

Thus, minimizing consistency on current inputs also limits old-task feature drift when current and old inputs overlap and respond to the same adapter changes.

Empirical justification for sufficient overlap. Overlap means that changing the same adapter weights affects both sets of conditioned features in similar ways. It does not require current and old images, or even their frozen-backbone features, to be similar one by one. $J_j(x)$ records the effect of small adapter-weight changes for one input, and G^j_t and G^j_j summarize these effects over current and old inputs.

We test this overlap directly with feature-Jacobian products over all shared-adapter parameters. We use ImageNet-R (Inc10) and analyze a checkpoint obtained while training the second task. At this fixed checkpoint, we construct two input sets. The first contains old inputs, namely images from the

Table 16: Feature-Jacobian overlap between current probes and old inputs under the old-task context c_1 on ImageNet-R (Inc10).

Measurement	Result	Interpretation
Similarity between adapter changes detected by current and old inputs	0.809 ± 0.041	Normalized similarity between G_1^0 and G_2^0. A value of 1 means identical sensitivity structure up to scale, and 0 means orthogonal structure. The result indicates substantial overlap.
Rank-16 old-gradient energy captured by current directions	$85.2 \pm 3.5\%$	We learn the 16 strongest adapter directions from current probes and test them on held-out old inputs. They capture 85.2% as much old-input Jacobian variance as directions learned directly from old inputs.
Sampled old-active directions satisfying $v^\top G_1^0 v \leq 2\, v^\top G_2^0 v$	96.88%	For 96.88% of sampled directions that affect old features, the old-input effect is no more than twice the effect detected by current probes.
Sampled old-active directions satisfying $v^\top G_1^0 v \leq 5\, v^\top G_2^0 v$	100%	Every sampled old-active direction is detected by current probes within a factor of 5.

first task. The second contains current probes, namely images from the second task. Both sets are processed using the old-task context c_1. Therefore, the only difference between G_1^0 and G_2^0 is the input distribution,

$$G_1^0 = \mathbb{E}_{x \sim \mathcal{D}_1}\big[J_1(x)^\top J_1(x)\big], \qquad G_2^0 = \mathbb{E}_{x \sim \mathcal{D}_2}\big[J_2(x)^\top J_2(x)\big]. \tag{13}$$

For each image set, we measure how the 768-dimensional conditioned feature responds to all shared-adapter parameters. The auxiliary classification heads are excluded because they do not produce the conditioned feature matched by the consistency loss.

These measurements support the required coverage interpretation. Current-task images, when evaluated under the old-task context, activate most adapter directions that are important for first-task features. The experiment does not claim that old- and current-task images are semantically similar.

We verified that this behavior remains after the full 20-task sequence. At the final checkpoint, we use images from the last task as current probes and compare them with images from old tasks 0, 4, 8, 12, and 16. For each old task j, both image sets are evaluated under that task's context c_j. Across these five comparisons, the average similarity between current-task directions and old-task directions is 0.869 ± 0.049, and 99.58% of sampled old-active directions satisfy the directional relation with $\kappa = 5$.

F Limitations

FACET is designed for exemplar-free class-incremental learning with a frozen pretrained vision backbone and disjoint task label sets during training. These assumptions make the method directly comparable to PTM-based CIL methods, but they also leave several directions open. While FACET substantially reduces the task-specific costs compared to state of the art methods for long task sequences with 200 tasks, task-specific cost is kept small in the auxiliary classifiers but does not fully disappear. Moreover, the single adapter capacity may saturate. Although we achieve strong performance up to 200 tasks, longer task sequences or highly heterogeneous tasks may eventually exceed what one shared adapter can represent. Extending our method to true lifelong continual learning over thousands of tasks is non-trivial. In addition, the replay-free objective preserves old task-specific feature distributions by matching a frozen copy of the model on current-task inputs, which is efficient and avoids storing old data. However, it is not equivalent to matching the full old data distribution and may be weaker when the data distribution of the new task is highly different from earlier tasks. Promising directions include extending the method to large language models and studying alternative task context prediction methods.