STABLE ROUTING FOR MIXTURE-OF-EXPERTS IN CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class-incremental learning (CIL) requires models to learn new classes sequentially while preserving prior knowledge. Recently, approaches that combine pre-trained models with mixture-of-experts (MoE) have received increasing attention in CIL: they typically expand experts during learning and employ a router to assign weights across experts. However, existing MoE methods often overlook *routing drift* induced by expert expansion. Once new experts are introduced, the router may reassign samples from earlier classes to newly added experts, thereby perturbing previously established expert compositions and causing interference even when old experts remain frozen. We argue that expandable MoE in CIL requires two complementary properties: stable old-class routing for knowledge preservation and sufficient capacity utilization for new-class adaptation. To this end, we propose **Sta**ble **R**outing for **MoE** (StaR-MoE), a routing-level framework for expandable MoE in CIL. By incorporating sensitivity-aware routing alignment, StaR-MoE aligns current old-class routing behavior with historical routing distributions through sensitivity-guided constraints. Complementarily, StaR-MoE introduces asymmetric capacity regularization to encourage effective utilization of the expanded expert pool without compromising class-specific routing specialization. Extensive experiments across four standard CIL benchmarks demonstrate that StaR-MoE consistently improves both average and last accuracy over state-of-the-art methods, highlighting the importance of stable routing.

1 INTRODUCTION

Deep neural networks have demonstrated remarkable capabilities when training data are independent and identically distributed (He et al., 2016; Dosovitskiy et al., 2021). However, real-world environments are dynamic, requiring the model to learn from a continuous stream of tasks with shifting distributions, known as continual learning (CL) (Wang et al., 2024; Zhou et al., 2024b). The major challenge in CL is catastrophic forgetting, where learning new tasks often interferes with previously acquired knowledge, leading to performance degradation on old tasks (McCloskey & Cohen, 1989; McClelland et al., 1995; French, 1999).

Among various CL settings, class-incremental learning (CIL) is particularly challenging, as it requires the model to discriminate among all learned classes without access to task identities during inference (Masana et al., 2022; Van de Ven et al., 2022). Recent studies such as SD-LoRA (Wu et al., 2025) and SEMA (Wang et al., 2025) adopt expandable mixture-of-experts (MoE) architectures for CIL, where lightweight task-specific modules are progressively introduced and their outputs are aggregated by learnable routing mechanisms. However, these methods do not explicitly preserve the routing logic learned for previous classes. As a result, expert expansion can shift old-class routing by normalizing over an expanded expert set, even when historical expert parameters are frozen. As shown in Figure 1, newly added experts (e.g., Expert 6) are activated for earlier tasks (e.g., Task 1) despite being introduced later.

We refer to this structural interference problem as *routing drift*, where expert expansion alters the computational pathways of learned classes. Routing drift therefore provides a structural explanation for forgetting in expandable MoE: even if historical experts are retained, old-class predictions can still be disrupted when old samples are routed through altered expert compositions, especially when they are assigned to newly added experts that were not optimized for old classes. To mitigate routing

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.

drift in expandable MoE-based CIL, the router should satisfy two complementary properties. First, the router should maintain the routing behavior of previously learned classes, so that old samples remain unaffected by subsequently added experts. Second, the router should retain sufficient capacity plasticity for new classes, allowing newly added experts to be effectively utilized without disrupting the established routing paths of old-class samples.

To address routing drift, we propose **Sta**ble **R**outing for **MoE** (StaR-MoE), a framework that couples historical routing preservation with capacity utilization during expert expansion. StaR-MoE introduces sensitivity-aware routing alignment (SARA), which uses class-wise statistics to align current routing for old classes with historical targets. These targets assign zero probability to subsequently introduced experts, allowing the alignment to constrain changes in historical expert preferences while suppressing routing to newly added experts for old classes. SARA further weights the alignment across layers according to router sensitivity estimated from gradients of the classification loss. To complement historical routing preservation, asymmetric capacity regularization (ACR) penalizes only overloaded experts on samples from the current task, discouraging excessive load concentration in the expanded expert pool. Our routing-level analysis bounds dense routing drift under the modeled distributions of router inputs, supporting the SARA objective. Correlation and controlled intervention analyses provide empirical evidence that routing drift contributes to forgetting of previously learned classes.

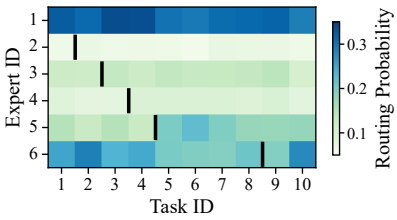


Figure 1: Illustration of *routing drift* in SEMA. Black lines mark expert expansion; non-zero routing probabilities to the left of each marker indicate later experts activated for earlier tasks.

In summary, our main contributions are as follows:

- We identify *routing drift* as a structural interference problem in expandable MoE-based CIL, where newly added experts can alter the computational pathways of previously learned classes even when historical experts are frozen.
- We propose StaR-MoE, a routing-level framework for mitigating routing drift by coupling historical routing preservation with controlled capacity utilization. It introduces sensitivity-aware routing alignment for preserving old-class routing and asymmetric capacity regularization to encourage expert utilization.
- We conduct extensive experiments on multiple CIL benchmarks, demonstrating the effectiveness of StaR-MoE in maintaining stable routing and accommodating new tasks.

2 Related Work

2.1 Continual Learning

Continual Learning (CL) aims to learn a sequence of tasks from non-stationary data streams without catastrophic forgetting (McCloskey & Cohen, 1989; Masana et al., 2022; Wang et al., 2024; Zhou et al., 2024b). Existing CL methods can be broadly classified into three primary categories: regularization-based, replay-based, and architecture-based approaches. Regularization-based methods (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Zenke et al., 2017; Aljundi et al., 2018) incorporate penalty terms into the loss function to constrain the updates of important parameters for previous tasks. Replay-based methods (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Buzzega et al., 2020; Choi et al., 2021; Sun et al., 2022) mitigate forgetting by retaining a buffer of raw data or training a generative model. More recently, several approaches (Zhang et al., 2023; Wang et al., 2023a; Li et al., 2025) alleviate memory overhead by modeling feature representations with class-wise Gaussian distributions. Architecture-based methods (Yoon et al., 2018; Li et al., 2019; Sokar et al., 2021; Liang & Li, 2023) dynamically expand the model capacity to accommodate new tasks. By isolating task-specific parameters, these approaches can retain acquired knowledge.

2.2 Continual Learning with Pre-trained Models

While traditional approaches primarily focus on models trained from scratch, recent strategies that pair a frozen pre-trained backbone with trainable lightweight components have demonstrated effec-

tiveness in CL (Wang et al., 2022b; Smith et al., 2023; Huang et al., 2024). Specifically, methods such as L2P (Wang et al., 2022b) and DualPrompt (Wang et al., 2022a) integrate pre-trained models with prompt-tuning strategies (Jia et al., 2022), learning to retrieve relevant prompts to adapt the frozen representations for different tasks. However, the absence of task identities in CIL makes such retrieval prone to errors. Mismatched prompts may activate inappropriate task-specific adaptation paths and degrade old-class predictions (Sun et al., 2025). CODA-Prompt (Smith et al., 2023) alleviates rigid prompt selection by dynamically combining prompt components, further enhancing adaptation flexibility. Moving beyond prompt-tuning, recent works such as InfLoRA (Liang & Li, 2024) incorporate specialized Low-Rank Adaptation (LoRA) (Hu et al., 2022) components and expand them during training. Similarly, approaches such as EASE (Zhou et al., 2024a) and APER (Zhou et al., 2025) employ adapter modules (Chen et al., 2022) for parameter-efficient fine-tuning.

2.3 Mixture of Experts in Continual Learning

To further scale model capacity while reducing the reliance on hard task-specific retrieval, recent research has adopted the MoE architecture, where multiple experts are softly composed through routing. Typically, an MoE comprises multiple experts and a router that assigns routing weights to aggregate their outputs (Shazeer et al., 2017; Fedus et al., 2022; Huai et al., 2025; Le et al., 2026), making the router a critical component for controlling expert utilization. For instance, MoE-Adapters (Yu et al., 2024) introduces a dedicated router for each task over a fixed set of experts, and selects the most relevant router during inference. SD-LoRA (Wu et al., 2025) utilizes task-specific LoRA components with decoupled direction and magnitude, and employs learnable scalar weights to combine these components. To address linear expert growth, SEMA (Wang et al., 2025) detects distribution shifts to adaptively decide when to expand. However, existing MoE-based CL methods mainly focus on expert construction or expansion, which leaves routing drift induced by continual expert expansion largely unaddressed. In contrast, StaR-MoE explicitly mitigates routing drift by preserving historical routing behavior, retaining stable routing for old classes while enabling the model to effectively learn new classes.

3 Preliminaries

We first introduce the CIL setting and the expandable MoE architecture used throughout the paper.

3.1 Class-Incremental Learning

In the CIL setting, the model learns from a continuous data stream where task identities are unavailable at inference time (Wang et al., 2022a; Wu et al., 2025). Formally, the training process consists of T sequential tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$. Each task $\mathcal{T}_t$ provides a dataset $\mathcal{D}_t = \{(x_n, y_n)\}_{n=1}^{N_t}$ with N_t instances, where x_n is the input image and $y_n \in \mathcal{C}_t$ is the corresponding label. The label sets are mutually disjoint, i.e., $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$. At step t, only data from $\mathcal{D}_t$ are accessible for training, and the model f_{θ_t} parameterized by θ_t is optimized on the current dataset via the classification loss:

$$\mathcal{L}_{\text{cur}}(\theta_t; \mathcal{D}_t) = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(f_{\theta_t}(x), y)], \tag{1}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and $\ell(\cdot, \cdot)$ is a per-sample loss function such as cross-entropy. The overall objective is to minimize the expected risk over all classes observed up to step t, i.e., $\mathcal{C}_{1:t} = \bigcup_{i=1}^{t} \mathcal{C}_i$, which requires the model to learn new tasks while mitigating forgetting on previous tasks whose data are unavailable.

3.2 Parallel Adapter as Expert

A Vision Transformer (ViT) (Dosovitskiy et al., 2021) consists of multiple transformer layers, each containing a multi-head self-attention module and a multi-layer perceptron (MLP) block. Following prior work (Chen et al., 2022; He et al., 2022), we freeze the ViT backbone and insert lightweight adapters as experts in parallel with the MLP blocks. Let $\mathbf{x}^l \in \mathbb{R}^d$ denote the input feature to the MLP block at the l-th layer. The adapter function $A^l(\mathbf{x}^l)$ comprises a down-projection $\mathbf{W}^l_{\text{down}} \in \mathbb{R}^{d\times r}$ that maps features to a bottleneck dimension r, followed by a ReLU activation (Nair & Hinton, 2010) and an up-projection $\mathbf{W}^l_{\text{up}} \in \mathbb{R}^{r\times d}$. For brevity, we omit the layer index l in the following

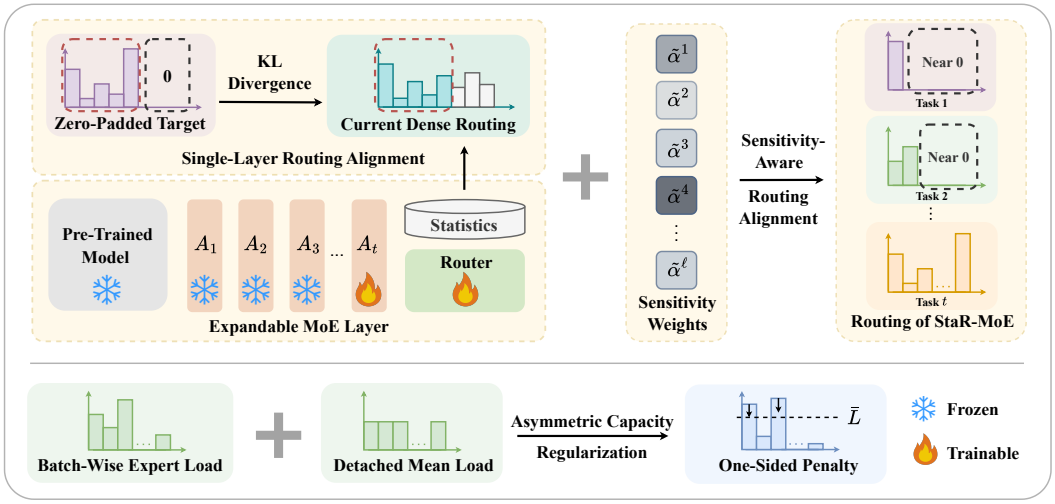


Figure 2: Overview of the StaR-MoE framework.

formulations. The adapter structure is then defined as:

$$A(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}\mathbf{W}_{\mathrm{down}})\mathbf{W}_{\mathrm{up}}. \tag{2}$$

3.3 Mixture-of-Experts with Top-k Routing

In each expandable MoE layer, a new adapter-based expert is added for each new task, and the router is expanded accordingly. Let E_{total} denote the number of available experts at the current task. Given the `[CLS]` token $\boldsymbol{z} \in \mathbb{R}^d$ as the router input, the routing logits are computed as:

$$\boldsymbol{h} = \boldsymbol{z}\mathbf{W}_{\mathrm{router}}, \tag{3}$$

where $\mathbf{W}_{\mathrm{router}} \in \mathbb{R}^{d \times E_{\mathrm{total}}}$ is the learnable routing matrix. Following the standard Top-k MoE routing (Shazeer et al., 2017), we define $\mathcal{K} = \text{Top-}k(\boldsymbol{h})$ as the selected expert set, with $|\mathcal{K}| = \min(k, E_{\mathrm{total}})$. The sparse routing probability is defined as:

$$G(\boldsymbol{z})_j = \begin{cases} \dfrac{\exp(h_j)}{\sum_{m \in \mathcal{K}} \exp(h_m)}, & j \in \mathcal{K}, \\ 0, & j \notin \mathcal{K}. \end{cases} \tag{4}$$

The MoE output is computed as:

$$\mathbf{x}_{\mathrm{out}} = \mathrm{MLP}(\mathbf{x}) + \sum_{j \in \mathcal{K}} G(\boldsymbol{z})_j A_j(\mathbf{x}), \tag{5}$$

where $A_j(\cdot)$ denotes the j-th adapter expert. In CIL, the expert pool and routing matrix expand over tasks. Without explicit constraints, expert expansion may change the routing distribution of earlier classes and alter their computational pathways.

4 Methodology

In this section, we present StaR-MoE in detail. As illustrated in Figure 2, StaR-MoE consists of two complementary components: sensitivity-aware routing alignment (SARA), which preserves historical routing for old classes, and asymmetric capacity regularization (ACR), which encourages effective capacity utilization under dynamic expert expansion. We first introduce SARA and then describe ACR.

4.1 Sensitivity-Aware Routing Alignment

To stabilize the routing for learned classes, we propose sensitivity-aware routing alignment (SARA), which maintains the historical routing behavior of old classes while focusing the alignment constraint on routers that are more sensitive to the current task optimization.

Single-layer routing alignment. We first describe the routing alignment objective for a single MoE layer. Given the router input $\boldsymbol{z}$ and routing matrix $\mathbf{W}_{\text{router}}$, the dense routing distribution vector is defined as:

$$\boldsymbol{P}(\boldsymbol{z}) = \text{Softmax}(\boldsymbol{z}\mathbf{W}_{\text{router}}). \tag{6}$$

We use the dense routing distribution for alignment instead of the sparse Top-k routing used in the forward pass. Dense routing provides gradients to all expert logits and directly suppresses newly added experts for old classes. Because dense alignment and Top-k selection share the same routing logits, dense alignment also constrains the scores underlying sparse expert selection.

Once class c has been learned, we derive its class-wise routing anchor to summarize its historical expert preference. Specifically, given the router inputs $\{\boldsymbol{z}_{c,i}\}_{i=1}^{N_c}$ of class c, we compute the average routing logits as $\bar{\boldsymbol{l}}_c = \frac{1}{N_c}\sum_{i=1}^{N_c} \boldsymbol{z}_{c,i}\mathbf{W}_{\text{router}}$, where N_c is the number of training samples in class c. The corresponding historical routing distribution is $\bar{\boldsymbol{P}}_c = \text{Softmax}(\bar{\boldsymbol{l}}_c)$. When new experts are added in later tasks, we construct a zero-padded target distribution for class c:

$$\hat{\boldsymbol{P}}_c = \text{Concat}\left(\bar{\boldsymbol{P}}_c, \mathbf{0}_{\text{new}}\right), \tag{7}$$

where $\mathbf{0}_{\text{new}}$ denotes zero probabilities assigned to newly added experts. This target distribution explicitly encodes the temporal constraint of expert expansion: experts introduced after class c was learned should not be activated for class c.

Since old training data are unavailable during the learning process, SARA approximates router inputs of old classes effectively through class-wise statistics. While prior works (Zhu et al., 2021; Zhang et al., 2023) typically use class-wise Gaussian statistics at the final layer for classifier retraining, SARA instead models router-input distributions across MoE layers, with empirical visualizations provided in Appendix E.2.2. For each old class c, we model the router-input distribution as $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, where $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the mean and diagonal covariance of the router inputs from class c. During later tasks, we sample synthetic router inputs $\tilde{\boldsymbol{z}}_c \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ and feed them only into the router to obtain the current dense routing distribution $\boldsymbol{P}(\tilde{\boldsymbol{z}}_c)$. The single-layer routing alignment loss is then defined using the Kullback-Leibler (KL) divergence as:

$$\begin{aligned}\mathcal{L}_{\text{align}} &= \sum_{c\in\mathcal{C}_{\text{old}}} \mathbb{E}_{\tilde{\boldsymbol{z}}_c\sim\mathcal{N}(\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)}\left[D_{\text{KL}}\left(\hat{\boldsymbol{P}}_c \parallel \boldsymbol{P}(\tilde{\boldsymbol{z}}_c)\right)\right] \\ &= \sum_{c\in\mathcal{C}_{\text{old}}} \mathbb{E}_{\tilde{\boldsymbol{z}}_c}\left[\sum_{j=1}^{E_{\text{total}}} \hat{\boldsymbol{P}}_{c,j}\log\left(\frac{\hat{\boldsymbol{P}}_{c,j}}{\boldsymbol{P}(\tilde{\boldsymbol{z}}_c)_j}\right)\right],\end{aligned} \tag{8}$$

where $\mathcal{C}_{\text{old}}$ denotes the set of previously learned classes, and E_{total} is the total number of experts. By matching $\hat{\boldsymbol{P}}_c$ and $\boldsymbol{P}(\tilde{\boldsymbol{z}}_c)$, the alignment loss preserves the relative expert preference learned for each old class and suppresses the routing probability assigned to newly introduced experts.

Sensitivity-aware weighting. While Eq. (8) stabilizes the router in a single MoE layer, StaR-MoE introduces routers across multiple transformer layers. Applying the same alignment strength to all routers can be suboptimal, as routing drift in different layers may affect the final prediction to different extents. Over-constraining less sensitive routers may impair new knowledge acquisition, while under-constraining highly sensitive routers may induce severe forgetting of previously acquired knowledge. Therefore, SARA adaptively weights the alignment objective according to router sensitivity.

For the ℓ-th MoE layer, we denote its routing matrix by $\mathbf{W}^{\ell}_{\text{router}}$ and define its alignment loss $\mathcal{L}^{\ell}_{\text{align}}$ following Eq. (8). Before training on the current task, SARA estimates the sensitivity of each router with a lightweight one-batch gradient proxy. Specifically, SARA uses only a single mini-batch $\mathcal{B}_t$ from the current task, without storing raw samples from previous tasks. SARA then performs one forward and one backward pass using the classification loss $\mathcal{L}_{\text{cur}}$, without updating any parameters. The sensitivity score of the ℓ-th router is defined as the gradient norm with respect to its routing matrix:

$$s^{\ell} = \left\| \nabla_{\mathbf{W}^{\ell}_{\text{router}}} \mathcal{L}_{\text{cur}}(\theta_t; \mathcal{B}_t) \right\|_F. \tag{9}$$

The sensitivity scores are normalized across all MoE layers as:

$$\alpha^{\ell} = \frac{\exp(s^{\ell})}{\sum_{\ell'\in\mathcal{L}_{\text{moe}}}\exp(s^{\ell'})}, \tag{10}$$

where $\mathcal{L}_{\text{moe}}$ denotes the set of MoE layers. To avoid an overly concentrated constraint on a single layer, we interpolate the sensitivity weights with uniform weights:

$$\tilde{\alpha}^{\ell} = \gamma \alpha^{\ell} + (1-\gamma)\frac{1}{|\mathcal{L}_{\text{moe}}|}, \tag{11}$$

where $\gamma \in [0,1]$ controls the strength of sensitivity-aware weighting. The final SARA objective is:

$$\mathcal{L}_{\text{SARA}} = \sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^{\ell} \mathcal{L}^{\ell}_{\text{align}}. \tag{12}$$

Routing drift bound. We further relate the alignment loss to routing drift. Consider the set of old classes $\mathcal{C}_{\text{old}}$ with cardinality $N = |\mathcal{C}_{\text{old}}|$. For the ℓ-th MoE layer and each class $c \in \mathcal{C}_{\text{old}}$, let $\hat{\boldsymbol{P}}^{\ell}_{c}$ denote the zero-padded target distribution, and let $\boldsymbol{P}^{\ell}(\tilde{\boldsymbol{z}}^{\ell}_{c})$ denote the current dense routing distribution on the synthesized router input $\tilde{\boldsymbol{z}}^{\ell}_{c}$. We define the layer-wise routing drift as:

$$\Delta^{\ell} = \sum_{c \in \mathcal{C}_{\text{old}}} \mathbb{E}_{\tilde{\boldsymbol{z}}^{\ell}_{c}} \left[\left\| \hat{\boldsymbol{P}}^{\ell}_{c} - \boldsymbol{P}^{\ell}(\tilde{\boldsymbol{z}}^{\ell}_{c}) \right\|_1 \right]. \tag{13}$$

Since SARA assigns different alignment strengths to different layers, we define the sensitivity-weighted routing drift across all MoE layers as:

$$\Delta_{\text{total}} = \sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^{\ell} \Delta^{\ell}. \tag{14}$$

Proposition 1 (Bound on Routing Drift)**.** *If the SARA objective satisfies $\mathcal{L}_{\text{SARA}} \leq \xi$, then the sensitivity-weighted routing drift is bounded by:*

$$\Delta_{\text{total}} \leq \sqrt{2N\xi}. \tag{15}$$

Remark. *Proposition 1 bounds dense routing drift under the modeled router-input distributions, providing a routing-level justification for SARA. The connection between routing drift and old-class forgetting is supported empirically in Section 5.4. The proof is provided in Appendix A.*

4.2 Asymmetric Capacity Regularization

While SARA stabilizes routing for old classes, relying on routing alignment alone may make the router overly biased toward historical experts. Since the zero-padded targets in SARA assign zero probability to newly added experts for old classes, the router may become biased toward historical experts and fail to fully utilize the expanded capacity for the current task. Thus, we introduce asymmetric capacity regularization (ACR), which encourages effective capacity utilization under dynamic expert expansion.

Let $\mathcal{B}$ denote a batch of router inputs derived from the current task, and let $\mathcal{K}$ be the Top-k expert set selected by the router for input $\boldsymbol{z}$. For each expert $j \in \{1, \ldots, E_{\text{total}}\}$, we compute its batch-level load as:

$$L_j = \sum_{\boldsymbol{z} \in \mathcal{B}} \mathcal{S}\left(j \in \mathcal{K} \mid \boldsymbol{z}\right), \tag{16}$$

where $\mathcal{S}(j \in \mathcal{K} \mid \boldsymbol{z})$ denotes the smooth probability that expert j is selected into the Top-k set for router input $\boldsymbol{z}$. The detailed formulation of $\mathcal{S}(\cdot)$ is provided in Appendix B. The load L_j measures the expected selection frequency of expert j on current router inputs.

Conventional load-balancing regularization in MoE (Shazeer et al., 2017; Fedus et al., 2022) encourages all experts to be used uniformly. However, such a symmetric objective is designed for static expert sets and does not account for dynamic expert expansion in CIL. In expandable MoE, previously learned experts should store old knowledge, while newly added experts should provide capacity for new-task adaptation. Forcing old and new experts toward uniform usage may therefore conflict with established expert specialization and amplify routing drift.

To address this issue, ACR complements SARA with an asymmetric capacity penalty. We define the detached mean load as:

$$\bar{L} = \text{sg}\left(\frac{1}{E_{\text{total}}} \sum_{j=1}^{E_{\text{total}}} L_j \right), \tag{17}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation. Instead of penalizing all experts uniformly, ACR only penalizes experts whose loads exceed the mean:

$$\mathcal{L}_{\mathrm{ACR}} = \frac{\frac{1}{E_{\mathrm{total}}} \sum_{j=1}^{E_{\mathrm{total}}} \left[\max(L_j - \bar{L}, 0)\right]^2}{\bar{L}^2 + \epsilon}. \tag{18}$$

Here, ϵ is a small constant for numerical stability. This one-sided formulation prevents a small subset of experts from dominating the routing allocation on the current task, while allowing non-uniform expert usage to preserve expert specialization. ACR improves the utilization of newly expanded capacity by discouraging excessive load concentration, while preserving routing specialization.

Total Objective. The final training objective of StaR-MoE is given by:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{cur}} + \lambda_{\mathrm{SARA}} \mathcal{L}_{\mathrm{SARA}} + \lambda_{\mathrm{ACR}} \mathcal{L}_{\mathrm{ACR}}, \tag{19}$$

where λ_{SARA} and λ_{ACR} control the strength of routing alignment and capacity regularization, respectively. The complete StaR-MoE algorithm is presented in Appendix C.

5 Experiments

In this section, we evaluate StaR-MoE on multiple class-incremental learning benchmarks. We first describe the experimental settings, then present benchmark comparisons, ablation studies, and further analysis on routing drift.

5.1 Experimental Settings

Datasets. Following previous work (Smith et al., 2023; Liang & Li, 2024; Wang et al., 2025), we evaluate StaR-MoE on four standard class-incremental learning benchmarks: ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al., 2021b), CIFAR-100 (Krizhevsky & Hinton, 2009), and VTAB (Zhai et al., 2019). We evaluate ImageNet-R under three different settings by splitting it into 5, 10, and 20 tasks, with each task containing 40, 20, and 10 classes, respectively. For ImageNet-A, we divide it into 10 tasks, each containing 20 classes. Similarly, CIFAR-100 is split into 10 tasks, each containing 10 classes. Finally, for VTAB, which comprises 50 classes across five domains, we organize it into 5 tasks with 10 classes each.

Evaluation metrics. Two standard continual learning metrics widely used in recent studies are employed for evaluation (Wang et al., 2022b; Wu et al., 2025). The first metric, last accuracy $\mathcal{A}_T$, measures the average classification accuracy across all tasks after the complete training sequence. The second metric, average accuracy $\bar{\mathcal{A}}$, evaluates the overall performance throughout the incremental process. It is defined as $\bar{\mathcal{A}} = \frac{1}{T} \sum_{t=1}^{T} \mathcal{A}_t$, where $\mathcal{A}_t$ is the accuracy on all learned classes after task t. These two metrics capture both the model's ability to learn new tasks and to retain knowledge of previously learned ones.

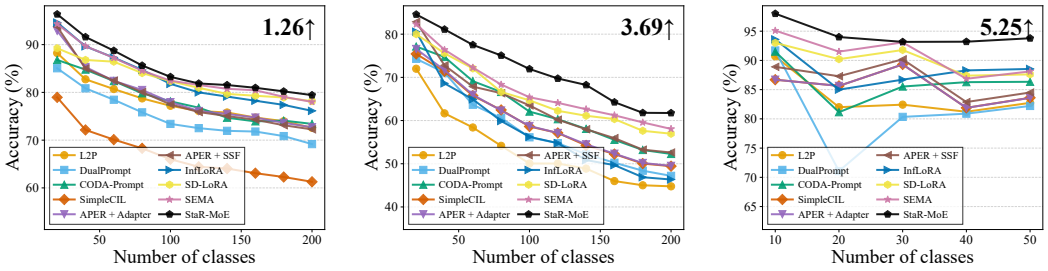
Baselines and implementation details. We compare StaR-MoE against state-of-the-art pre-trained ViT-based methods. Prompt-based methods include L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), and CODA-Prompt (Smith et al., 2023). Statistics-based methods include SimpleCIL (Zhou et al., 2025) and APER (Zhou et al., 2025). Dynamically expandable methods include InfLoRA (Liang & Li, 2024), SD-LoRA (Wu et al., 2025), and SEMA (Wang et al., 2025). Comparing against these diverse approaches allows us to comprehensively evaluate the effectiveness of StaR-MoE. In addition to the ViT-B/16 (Dosovitskiy et al., 2021) pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K using supervised learning, we also evaluate a self-supervised ViT-B/16 obtained with DINO (Caron et al., 2021). Details of the experiments are provided in Appendix D.

5.2 Benchmark Comparison

Table 1 reports the comparison with state-of-the-art ViT-based CIL methods. StaR-MoE consistently achieves the best performance across all benchmarks and task splits. On ImageNet-R, StaR-MoE obtains $\bar{\mathcal{A}} = 86.09\%$, 84.96%, and 82.90% under the 5-task, 10-task, and 20-task settings, respectively, showing robust performance as the task sequence becomes longer. The improvements are more pronounced on ImageNet-A and VTAB, which involve stronger distribution shifts. On

Table 1: Comparison with state-of-the-art ViT-based methods. All models adopt ViT-B/16-IN1K as the backbone. IN-R stands for ImageNet-R. $\bar{\mathcal{A}}$ denotes average accuracy, and $\mathcal{A}_T$ denotes last accuracy. The best results are highlighted in **bold**.

Method	CIFAR-100		5-Task IN-R		10-Task IN-R		20-Task IN-R		ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
L2P	87.18	81.85	77.40	73.59	66.97	62.72	70.67	62.90	51.40	42.94	81.19	80.83
DualPrompt	87.36	82.30	76.39	72.29	72.83	66.75	62.33	61.97	54.68	45.49	82.89	79.79
CODA-Prompt	91.50	86.16	81.63	76.98	78.48	73.40	75.00	70.02	62.90	52.27	86.16	86.34
SimpleCIL	82.31	76.21	65.84	61.28	67.06	61.28	67.58	61.28	59.67	49.44	85.42	83.61
APER + Adapter	90.94	85.75	79.43	74.02	79.17	72.70	77.66	70.98	59.89	49.51	85.42	83.59
APER + SSF	89.81	83.92	79.82	74.13	78.93	72.25	75.60	68.53	63.33	52.60	86.75	84.51
InfLoRA	90.51	85.05	81.81	76.95	81.39	75.32	78.87	72.60	60.92	49.20	88.43	88.54
SD-LoRA	92.05	87.26	83.01	79.15	82.59	78.17	79.80	75.13	64.95	55.96	90.00	87.61
SEMA	91.63	86.85	84.25	78.85	83.56	78.00	81.75	74.53	67.03	58.06	90.93	88.12
StaR-MoE	**92.69**	**88.78**	**86.09**	**81.62**	**84.96**	**79.43**	**82.90**	**76.28**	**71.60**	**61.75**	**94.43**	**93.79**



(a) ImageNet-R (10 tasks) (b) ImageNet-A (c) VTAB

Figure 3: Performance curves of different methods on ImageNet-R, ImageNet-A, and VTAB. We annotate StaR-MoE's final-stage accuracy gain over the runner-up method in the top-right corner of each subfigure.

ImageNet-A, StaR-MoE achieves $\bar{\mathcal{A}} = 71.60\%$ and $\mathcal{A}_T = 61.75\%$, improving over SEMA by 4.57 and 3.69 points, respectively. On VTAB, StaR-MoE obtains $\bar{\mathcal{A}} = 94.43\%$ and $\mathcal{A}_T = 93.79\%$, surpassing the best baseline in each metric by 3.50 and 5.25 points. These results indicate that stable routing is particularly beneficial under distribution-shifted incremental scenarios. Appendix E.8 further shows that StaR-MoE outperforms SEMA under matched capacity and expansion schedules.

Figure 3 further shows the task-wise accuracy curves on representative benchmarks. StaR-MoE maintains a consistent advantage throughout the incremental process, and the final-stage gains annotated in each subfigure confirm its superiority over the runner-up methods. This suggests that mitigating routing drift helps preserve previously learned knowledge while adapting to new classes.

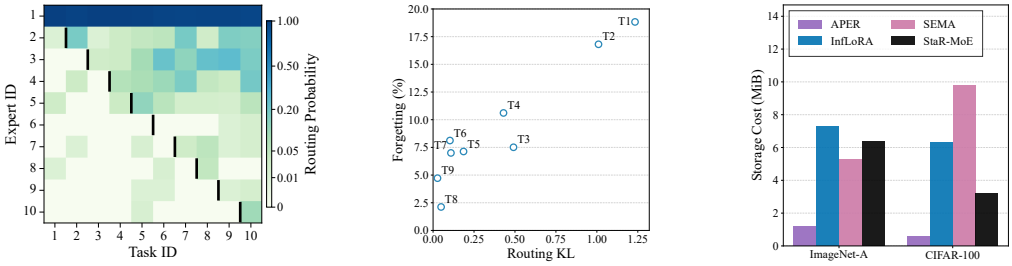
5.3 Ablation Studies

In this section, we conduct component-wise ablation studies on ImageNet-R (10 tasks) and ImageNet-A to evaluate SARA and ACR. We compare StaR-MoE with three variants: **Baseline**, which trains the newly added expert and router using only the classification loss; **w/o ACR**, which keeps only SARA; and **w/o SARA**, which keeps only ACR. The results are reported in Table 2.

As shown in Table 2, removing either SARA or ACR degrades the performance of StaR-MoE. The w/o ACR variant still substantially outperforms the baseline, confirming that SARA is crucial for preserving historical routing behavior and mitigating routing drift. The w/o SARA variant also improves over the baseline, indicating that ACR helps exploit the expanded expert capacity. However, ACR alone is consistently weaker than SARA alone, showing that capacity regularization cannot replace explicit routing alignment. The full StaR-MoE achieves the best results across all metrics, demonstrating the complementarity of SARA and ACR in expandable MoE.

Table 2: Ablation studies on ImageNet-R (10 tasks) and ImageNet-A. We report average accuracy $\bar{\mathcal{A}}$ and last accuracy $\mathcal{A}_T$.

Method	Components		ImageNet-R		ImageNet-A	
	SARA	ACR	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Baseline	×	×	80.82	75.20	66.56	57.87
w/o ACR	✓	×	84.56	78.62	71.09	59.71
w/o SARA	×	✓	83.00	76.18	68.76	59.12
StaR-MoE	✓	✓	**84.96**	**79.43**	**71.60**	**61.75**



(a) Routing stability visualization. (b) Routing drift vs. forgetting. (c) Storage cost.

Figure 4: Further analysis. (a) Task-wise routing probabilities of StaR-MoE on ImageNet-A. Low probabilities in the lower-left region indicate that later experts are rarely activated for earlier tasks. (b) Relationship between final routing drift and forgetting across old tasks on ImageNet-R, showing a strong positive correlation. (c) Extra storage cost of different methods.

5.4 Further Analysis

Routing stability analysis. We visualize the task-wise routing probabilities of StaR-MoE on ImageNet-A, using the same dataset as in Figure 1. To better reveal small routing probabilities, we apply a non-uniform color scale to the heatmap. As shown in Figure 4a, the lower-left region has consistently low probabilities, indicating that later experts are rarely activated for earlier tasks. In contrast to the routing drift observed in Figure 1, this result shows that StaR-MoE effectively mitigates routing drift and preserves historical routing behavior.

Routing drift and forgetting. For the model after completing all tasks, we observe a strong positive correlation between routing KL and forgetting across old tasks, with Spearman's $\rho = 0.850$ and $p = 0.0037$. The results indicate that tasks with greater final routing drift generally suffer more forgetting. A controlled routing intervention using oracle task identities further recovers old-task accuracy while keeping all model parameters fixed. Detailed results are provided in Appendix E.1.

Storage cost. We further compare the extra storage cost of StaR-MoE with representative CIL methods. As shown in Figure 4c, StaR-MoE incurs moderate storage overhead on ImageNet-A and requires substantially less storage than SEMA and InfLoRA on CIFAR-100. These results show that StaR-MoE preserves routing stability without introducing excessive storage overhead.

6 Conclusion

In this paper, we have introduced Stable Routing for MoE (StaR-MoE), a routing-level framework for expandable MoE in CIL, motivated by the observation that continual expert expansion can induce routing drift. Specifically, newly added experts may alter old-class routing behavior and contribute to forgetting. To mitigate routing drift, StaR-MoE incorporates sensitivity-aware routing alignment to preserve old-class routing behavior by imposing stronger constraints on prediction-sensitive router layers. It further introduces asymmetric capacity regularization to encourage effective utilization of the expanded expert pool. Theoretical analysis connects SARA to an upper bound on aggregate routing drift, and extensive experiments validate StaR-MoE across multiple CIL benchmarks. In the future, extending StaR-MoE to multimodal continual learning settings remains a challenging direction, where stable routing must be maintained across diverse multimodal data streams.

AI USE STATEMENT

In this work, we used generative AI tools to edit the research paper to improve readability and to assist with literature retrieval and discovery. We have reviewed all AI-assisted work. All AI-assisted outputs were reviewed by the authors for correctness and relevance. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the datasets, task construction, evaluation protocols, experimental settings, and implementation details in Section 5 and Appendix D, with the training procedure detailed in Appendix C. All datasets used in our experiments are publicly available. The definitions and assumptions underlying Proposition 1 are presented in Section 4.1, and the complete proof is provided in Appendix A.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision*, pp. 139–154, 2018.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems 33*, pp. 15920–15930, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

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

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Dual-teacher class-incremental learning with data-free generative replay. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3543–3552, 2021.

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

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

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. 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.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. CL-MoE: Enhancing multimodal large language model with dual momentum Mixture-of-Experts for continual visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19608–19617, 2025.

Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with CLIP: Adaptive representation adjustment and parameter fusion. In *Proceedings of the European Conference on Computer Vision*, pp. 214–231, 2024.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of the European Conference on Computer Vision*, pp. 709–727, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Minh Le, Bao-Ngoc Dao, Huy Nguyen, Quyen Tran, Anh Nguyen, and Nhat Ho. One-prompt strikes back: Sparse mixture of experts for prompt-based continual learning. In *International Conference on Learning Representations*, 2026.

Lan Li, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Addressing imbalanced domain-incremental learning through dual-balance collaborative experts. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Xiao Li, Sheng Liu, Jinxin Zhou, Xinyu Lu, Carlos Fernandez-Granda, Zhihui Zhu, and Qing Qu. Understanding and improving transfer learning of deep models via neural collapse. *Transactions on Machine Learning Research*, 2024.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3925–3934, 2019.

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. Adaptive plasticity improvement for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7816–7825, 2023.

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. In *Advances in Neural Information Processing Systems 30*, pp. 6467–6476, 2017.

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.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve Restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.

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.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

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.

Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. SpaceNet: Make free space for continual learning. *Neurocomputing*, 439:1–11, 2021.

Hai-Long Sun, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. PILOT: A pre-trained model-based continual learning toolbox. *arXiv preprint arXiv:2309.07117*, 2023.

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*, pp. 20699–20707, 2025.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. In *Advances in Neural Information Processing Systems 35*, pp. 27075–27086, 2022.

Gido M. Van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10087–10098, 2025.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. In *Advances in Neural Information Processing Systems 36*, pp. 69054–69076, 2023a.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *Proceedings of the European Conference on Computer Vision*, pp. 631–648, 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.

Zijian Wang, Yadan Luo, Liang Zheng, Zi Huang, and Mahsa Baktashmotlagh. How far pre-trained models are from neural collapse on the target dataset informs their transferability. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5526–5535, 2023b.

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.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23219–23230, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3987–3995, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

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.

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, 2024a.

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*, 46(12):9851–9873, 2024b.

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.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5871–5880, 2021.

A Proof of Proposition 1

We first bound the routing drift for a single MoE layer ℓ. Following the notation in Sec. 4.1, let $\hat{\boldsymbol{P}}_c^\ell$ denote the zero-padded target distribution, and let $\boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell)$ denote the current dense routing distribution on the synthesized router input $\tilde{\boldsymbol{z}}_c^\ell$. For the set of old classes $\mathcal{C}_{\text{old}}$ with $N = |\mathcal{C}_{\text{old}}|$, the layer-wise routing drift is defined as:

$$\Delta^\ell = \sum_{c \in \mathcal{C}_{\text{old}}} \mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[\left\| \hat{\boldsymbol{P}}_c^\ell - \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right\|_1 \right]. \tag{20}$$

Since both distributions are defined over the same expanded expert set, their L_1 distance is twice the total variation (TV) distance:

$$\left\| \hat{\boldsymbol{P}}_c^\ell - \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right\|_1 = 2\delta_{\text{TV}} \left(\hat{\boldsymbol{P}}_c^\ell, \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right). \tag{21}$$

By Pinsker's inequality, for any two distributions P and Q, $\delta_{\text{TV}}(P, Q) \le \sqrt{\frac{1}{2} D_{\text{KL}}(P \parallel Q)}$. Applying this inequality to the zero-padded target distribution and the current dense routing distribution, and using Jensen's inequality $\mathbb{E}[\sqrt{X}] \le \sqrt{\mathbb{E}[X]}$, we obtain:

$$\begin{aligned} \Delta^\ell &= \sum_{c \in \mathcal{C}_{\text{old}}} 2\mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[\delta_{\text{TV}} \left(\hat{\boldsymbol{P}}_c^\ell, \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right) \right] \\ &\le \sum_{c \in \mathcal{C}_{\text{old}}} 2\mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[\sqrt{\frac{1}{2} D_{\text{KL}} \left(\hat{\boldsymbol{P}}_c^\ell \parallel \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right)} \right] \\ &\le \sqrt{2} \sum_{c \in \mathcal{C}_{\text{old}}} \sqrt{\mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[D_{\text{KL}} \left(\hat{\boldsymbol{P}}_c^\ell \parallel \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right) \right]}. \end{aligned} \tag{22}$$

To bound the summation over old classes, we use the Cauchy–Schwarz inequality. For non-negative values x_c, $\sum_{c=1}^N \sqrt{x_c} \le \sqrt{N \sum_{c=1}^N x_c}$. Letting x_c be given by:

$$x_c = \mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[D_{\text{KL}} \left(\hat{\boldsymbol{P}}_c^\ell \parallel \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right) \right], \tag{23}$$

we have:

$$\sum_{c \in \mathcal{C}_{\text{old}}} \sqrt{x_c} \le \sqrt{N} \sqrt{\sum_{c \in \mathcal{C}_{\text{old}}} \mathbb{E}_{\tilde{\boldsymbol{z}}_c^\ell} \left[D_{\text{KL}} \left(\hat{\boldsymbol{P}}_c^\ell \parallel \boldsymbol{P}^\ell(\tilde{\boldsymbol{z}}_c^\ell) \right) \right]}. \tag{24}$$

The term inside the square root is exactly the single-layer routing alignment loss $\mathcal{L}_{\text{align}}^\ell$. Therefore, we obtain:

$$\Delta^\ell \le \sqrt{2N \mathcal{L}_{\text{align}}^\ell}. \tag{25}$$

We now extend the result to the sensitivity-weighted routing drift across all MoE layers. By definition, we have:

$$\Delta_{\text{total}} = \sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^\ell \Delta^\ell. \tag{26}$$

Substituting the layer-wise bound gives:

$$\Delta_{\text{total}} \le \sqrt{2N} \sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^\ell \sqrt{\mathcal{L}_{\text{align}}^\ell}. \tag{27}$$

The interpolated sensitivity weights $\tilde{\alpha}^\ell$ are non-negative and satisfy $\sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^\ell = 1$. Applying Jensen's inequality to the concave square-root function yields:

$$\sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^\ell \sqrt{\mathcal{L}_{\text{align}}^\ell} \le \sqrt{\sum_{\ell \in \mathcal{L}_{\text{moe}}} \tilde{\alpha}^\ell \mathcal{L}_{\text{align}}^\ell} = \sqrt{\mathcal{L}_{\text{SARA}}}. \tag{28}$$

Thus, we obtain:

$$\Delta_{\text{total}} \le \sqrt{2N \mathcal{L}_{\text{SARA}}}. \tag{29}$$

If the SARA objective satisfies $\mathcal{L}_{\text{SARA}} \le \xi$, then we have:

$$\Delta_{\text{total}} \le \sqrt{2N\xi}. \tag{30}$$

This proves Proposition 1. Proposition 1 therefore provides an upper bound on the sensitivity-weighted dense routing drift under the modeled router-input distributions.

Algorithm 1 StaR-MoE for Continual Learning

Input: Sequential Tasks $\{1, \ldots, T\}$, Pre-trained ViT Backbone, Hyperparameters $\lambda_{\text{SARA}}, \lambda_{\text{ACR}}, \gamma$.
Initialize: Frozen ViT Backbone, Initial Experts $\mathcal{E}$, Routers $\mathcal{R}$, and Empty Statistics $\mathcal{M} \leftarrow \emptyset$.
for Task $t = 1$ **to** T **do**
  if $t > 1$ **then**
    Expand $\mathcal{E}$ with new experts and $\mathcal{R}$ with new router parameters.
    Compute layer-wise sensitivity weights $\tilde{\alpha}$ via a single mini-batch from $\mathcal{D}_t$.
  end if
  for each training epoch and mini-batch (x, y) in $\mathcal{D}_t$ **do**
    $\mathcal{L}_{\text{cur}} \leftarrow$ Compute Cross-Entropy Loss via Eq. (1)
    if $t > 1$ **then**
      $\mathcal{L}_{\text{SARA}} \leftarrow$ Routing Alignment using $\tilde{\boldsymbol{\alpha}}$ via Eq. (12)
      $\mathcal{L}_{\text{ACR}} \leftarrow$ Encourage Capacity Utilization via Eq. (18)
    else
      $\mathcal{L}_{\text{SARA}} \leftarrow 0$, $\mathcal{L}_{\text{ACR}} \leftarrow 0$
    end if
    Update by minimizing $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cur}} + \lambda_{\text{SARA}}\mathcal{L}_{\text{SARA}} + \lambda_{\text{ACR}}\mathcal{L}_{\text{ACR}}$
  end for
  if $t < T$ **then**
    Update statistics $\mathcal{M}$ using $\mathcal{D}_t$.
  end if
end for

B Smooth Load Estimator

In ACR, directly using the hard Top-k assignment indicator $\mathbb{I}[j \in \mathcal{K}]$ to compute expert load is non-differentiable. Therefore, we use a smooth estimator $\mathcal{S}(j \in \mathcal{K} \mid \boldsymbol{z})$ to approximate the probability that expert j is selected into the Top-k expert set (Shazeer et al., 2017).

Given the router input $\boldsymbol{z}$, the routing logit of expert j is:

$$g_j(\boldsymbol{z}) = \left(\boldsymbol{z}\mathbf{W}_{\text{router}}\right)_j . \tag{31}$$

Let $\tau_k(\boldsymbol{z}, j)$ denote the k-th highest logit value among all other experts excluding j for router input $\boldsymbol{z}$, and let σ_j denote the smoothing scale. The smooth selection probability is computed as:

$$\mathcal{S}\left(j \in \mathcal{K} \mid \boldsymbol{z}\right) = \Phi\left(\frac{g_j(\boldsymbol{z}) - \tau_k(\boldsymbol{z}, j)}{\sigma_j}\right), \tag{32}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. This differentiable estimator allows gradients from the load regularization term to be propagated to the router.

Using this estimator, the batch-level load of expert j is computed as:

$$L_j = \sum_{\boldsymbol{z} \in \mathcal{B}} \mathcal{S}\left(j \in \mathcal{K} \mid \boldsymbol{z}\right), \tag{33}$$

which measures the total selection strength of expert j over the current-task router inputs.

C StaR-MoE Algorithm

We present the detailed procedure in Algorithm 1.

D Experimental Setups and Implementation Details

Following established protocols (Smith et al., 2023; Wu et al., 2025), we adopt ViT-B/16 (Dosovitskiy et al., 2021) pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K as our backbone.

Table 3: Comparison of routing drift and forgetting under different configurations on ImageNet-R with 10 tasks.

Method	Routing KL	Forgetting
Baseline	3.329	13.21%
SARA only	**0.143**	9.58%
Full model	0.404	**9.21%**

For fair comparison, the backbone remains frozen across all methods. In addition to the supervised backbone, we evaluate a self-supervised ViT-B/16 pre-trained with DINO (Caron et al., 2021). Following SEMA (Wang et al., 2025), we apply expert expansion only to the last six transformer layers, while each of the first six layers uses a single adapter without expansion. In the MoE layers, experts learned from previous tasks are frozen during subsequent training, while newly added experts and all router parameters are updated.

For consistent and fair comparison, we use the same class order and data partitions for all methods. Following Rebuffi et al. (2017); Zhou et al. (2024b), we randomly shuffle the class order using the seed 1993 before partitioning classes into incremental tasks. The resulting task sequence and the corresponding training and testing splits are kept identical across all compared methods.

For baselines such as SimpleCIL, APER, InfLoRA, SD-LoRA, and SEMA, we utilize their official implementations. For prompting-based methods, namely L2P, DualPrompt, and CODA-Prompt, we adopt the open-source implementations from the PILOT toolbox (Sun et al., 2023). All baseline methods follow the hyperparameter configurations recommended in their original publications or official implementations.

StaR-MoE is optimized using Adam (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, where the learning rate decays with cosine annealing. The training epochs are set to 30 for ImageNet-R, 10 for CIFAR-100, 15 for ImageNet-A, and 30 for VTAB. The adapter projection dimension r is set to 8 for ImageNet-A and the 20-task ImageNet-R setting, and 16 for all other benchmarks. We consistently set the number of active experts $k = 2$ across all datasets. For the method-specific hyperparameters of StaR-MoE, we split the original training set into training and validation subsets with a ratio of $4{:}1$, and select hyperparameters according to the validation performance without using the official test results. We set $\lambda_{\mathrm{SARA}} = 0.6$, $\lambda_{\mathrm{ACR}} = 0.4$, and $\gamma = 0.5$ as the default configuration across datasets. Once the hyperparameters are fixed, the official test set is used exclusively for final evaluation. All experiments are conducted on NVIDIA GeForce RTX 4090 GPUs with 24GB of memory, using PyTorch 2.0.1.

E Additional Experiment Results

E.1 Relationship between Routing Drift and Forgetting

To examine the relationship between routing drift and forgetting more directly, we conduct three complementary analyses. Routing KL compares dense routing of the same test samples using $D_{\mathrm{KL}}(P_{\mathrm{hist}} \parallel P_{\mathrm{cur}})$, with zero-padded references recorded immediately after learning their task.

E.1.1 Configuration-Level Analysis

We compare routing drift and forgetting across three representative configurations on ImageNet-R with 10 tasks. For each previous task $j < T$, we define its forgetting as $f_{T,j} = \max_{k\in\{j,\ldots,T-1\}} a_{k,j} - a_{T,j}$, where $a_{k,j}$ denotes the accuracy on task j after training through task k. Overall forgetting is computed by averaging over all previous tasks as $\frac{1}{T-1}\sum_{j=1}^{T-1} f_{T,j}$. As shown in Table 3, both configurations with SARA substantially reduce routing drift and forgetting compared with the baseline. While SARA alone achieves the lowest routing KL, the full model further shows that achieving the best performance requires both low routing drift and sufficient expert utilization.

Table 4: Task-wise final routing drift and forgetting after completing Task 10 on ImageNet-R.

Metric	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
Final Routing KL	1.233	1.010	0.491	0.430	0.186	0.103	0.109	0.049	0.027
Final Forgetting	18.83%	16.81%	7.51%	10.61%	7.13%	8.12%	7.00%	2.12%	4.72%

Table 5: Comparison with alternative router-preservation strategies on ImageNet-A.

Router Strategy	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Router freezing	69.62	58.33
LwF-style router distillation	70.01	59.38
EWC on router	69.48	59.18
Router entropy regularization	69.32	59.05
SARA	**71.60**	**61.75**

E.1.2 Task-Level Correlation Analysis

We further report the task-wise final routing drift and forgetting under the full model after completing Task 10 on ImageNet-R. The exact task-wise values corresponding to Figure 4b are summarized in Table 4. The two quantities are strongly positively correlated across old tasks, with Spearman's $\rho = 0.850$ and $p = 0.0037$, indicating that tasks with greater final routing drift generally suffer more forgetting.

E.1.3 Controlled Routing Intervention

We additionally conduct a controlled routing intervention on a fully trained ImageNet-A model. After training on all tasks, we use oracle task identities at inference time to mask experts introduced after the corresponding task, while keeping all model parameters unchanged. This intervention recovers 5.72 percentage points of accuracy on Task 1 and 1.04 points on average across old tasks. Since only the routing paths are changed while all model parameters remain fixed, the recovered performance provides more direct evidence that routing drift is an important contributing factor to forgetting.

E.2 Detailed Analysis of SARA

E.2.1 Comparison with Alternative Router-Preservation Strategies

We further compare SARA with several alternative router-preservation strategies on ImageNet-A. All variants use the same architecture and replace SARA with the corresponding strategy, while keeping ACR and all other settings fixed. Router freezing freezes existing router columns while training only newly added columns; LwF-style router distillation (Li & Hoiem, 2017) preserves the pre-expansion routing outputs using current-task inputs; EWC (Kirkpatrick et al., 2017) penalizes Fisher-weighted changes to router parameters; and router entropy regularization encourages confident routing on current-task samples.

SARA achieves the best performance among the evaluated strategies, outperforming the strongest alternative, LwF-style router distillation, by 1.59/2.37 percentage points in $\bar{\mathcal{A}}/\mathcal{A}_T$. With ACR fixed across all variants, these results support the effectiveness of SARA's historical routing alignment over the evaluated router-preservation alternatives in expandable MoE.

E.2.2 Visualization of Router-Input Distributions

We visualize the router-input distributions across different MoE layers using t-SNE (Van der Maaten & Hinton, 2008), as shown in Figure 5. Router inputs from the same class generally exhibit compact local structures, suggesting that router inputs retain class-related information across layers. Prior representation-learning studies have similarly observed compact class-wise structures in pre-trained models, including ViTs (Wang et al., 2023b; Li et al., 2024). This observation provides empiri-

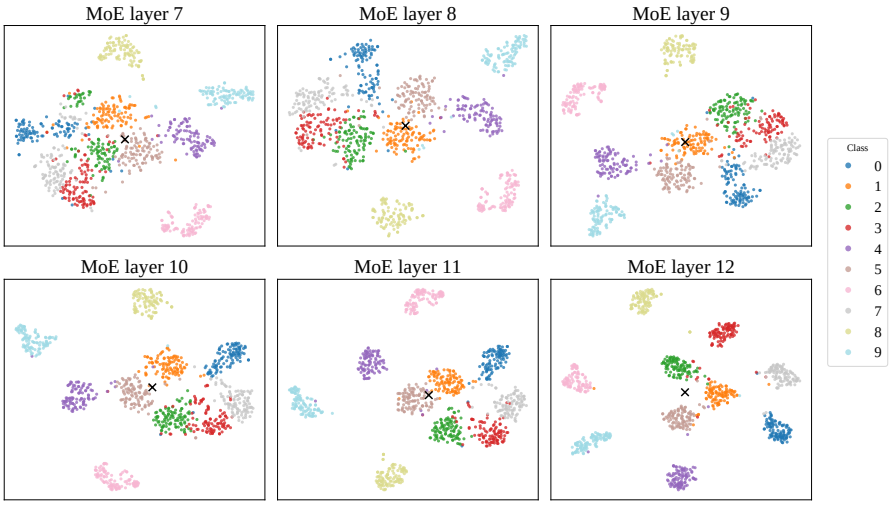


Figure 5: t-SNE visualization of router-input distributions across MoE layers on VTAB. Samples from different classes exhibit compact local structures in the router-input space, providing empirical motivation for class-wise statistics.

Table 6: Robustness to different modeling strategies for historical router inputs on ImageNet-A.

Historical Router-Input Modeling	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Mean only	70.44	59.78
Laplace	70.83	60.43
Student-t	70.94	60.11
Diagonal Gaussian	**71.60**	**61.75**

Table 7: Effect of sensitivity-aware weighting on ImageNet-A.

Weighting Strategy	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Uniform	71.22	61.16
Sensitivity-aware	**71.60**	**61.75**

cal motivation for using class-wise statistics to approximate old-class router-input distributions in SARA.

E.2.3 Robustness to Router-Input Modeling

To examine whether SARA depends on the specific Gaussian approximation of historical router inputs, we compare several alternative modeling strategies on ImageNet-A, including mean-only statistics, Laplace and Student-t distributions. As shown in Table 6, SARA remains effective across different modeling strategies, while the diagonal Gaussian achieves the best overall performance. These results support the use of class-wise statistical modeling and indicate that SARA is robust to the specific choice of router-input distribution modeling.

E.2.4 Effect and Robustness of Sensitivity-Aware Weighting

We further evaluate the effectiveness of the sensitivity-aware weighting strategy in SARA by comparing it with uniform alignment on ImageNet-A, while keeping all other settings unchanged. As shown in Table 7, sensitivity-aware weighting improves $\bar{\mathcal{A}}$ and $\mathcal{A}_T$ by 0.38 and 0.59 percentage points, respectively. These results support the benefit of assigning larger alignment weights to more sensitive router layers rather than treating all layers equally.

Table 8: Effect of sensitivity-estimation batch size on VTAB. B denotes the default batch size.

Estimation Batch Size	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
B	94.43	**93.79**
$2B$	94.37	93.33
$5B$	**94.55**	93.24

Table 9: Analysis of ACR on ImageNet-A.

Capacity regularization	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
SARA only (w/o ACR)	71.09	59.71
Standard load balancing	70.73	60.17
Full model with ACR	**71.60**	**61.75**

We next examine the effect of sensitivity-estimation batch size. On VTAB, we increase the number of samples used in a single sensitivity estimate to two and five times the default size, while keeping all other settings fixed. As shown in Table 8, increasing the estimation batch size does not yield consistent improvements. In particular, using five times as many samples improves $\bar{\mathcal{A}}$ by only 0.12 percentage points, while $\mathcal{A}_T$ decreases by 0.55 points. These results support the default estimation batch size as an effective and lightweight choice.

Finally, we assess performance sensitivity to the mini-batch used for sensitivity estimation. We fix the class order and all other training settings, and vary only the random seed used to sample the sensitivity-estimation batch. Across three batch-sampling seeds, StaR-MoE achieves $\bar{\mathcal{A}} = 94.25 \pm 0.16$ and $\mathcal{A}_T = 93.21 \pm 0.50$ on VTAB (mean $\pm$ standard deviation). These results suggest limited performance sensitivity to the sampled mini-batch.

E.3 Detailed Analysis of ACR

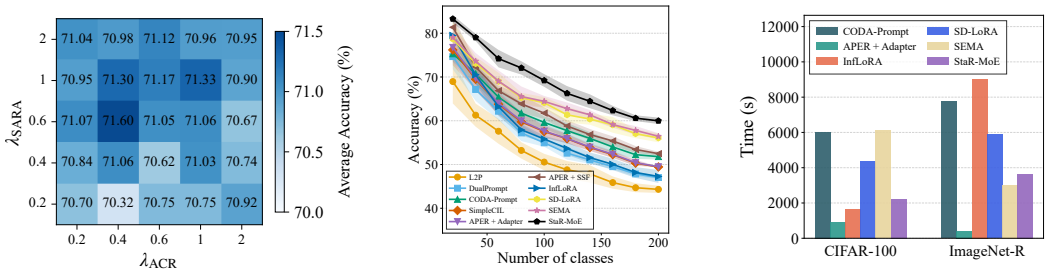
We further evaluate the contribution of ACR on ImageNet-A by comparing the full model with SARA-only and a conventional load-balancing regularizer (Shazeer et al., 2017). For each variant, we replace only the ACR component while keeping all other settings unchanged. As shown in Table 9, ACR improves $\bar{\mathcal{A}}/\mathcal{A}_T$ by 0.51/2.04 points over SARA-only and also outperforms conventional load balancing. Unlike symmetric load balancing, ACR penalizes only above-average expert loads on current-task samples, without directly penalizing underutilized experts. This design discourages excessive load concentration while allowing flexibility in how the remaining capacity is allocated. The improvements over SARA-only and standard load balancing support ACR as an effective complement to historical routing alignment.

E.4 Hyperparameter Robustness

We analyze the sensitivity of StaR-MoE to λ_{SARA} and λ_{ACR}, which control the strengths of routing alignment and capacity regularization, respectively. Specifically, we vary both coefficients within $\{0.2, 0.4, 0.6, 1.0, 2.0\}$. As shown in Figure 6a, StaR-MoE maintains stable performance across a wide range of values, indicating that the method is robust to these coefficients.

E.5 Experimental Results with Different Seeds

Following Rebuffi et al. (2017), the experiments in the main text use the random seed 1993 to randomize the class order. To further evaluate the statistical stability of StaR-MoE, we repeat the experiments on ImageNet-A with multiple random seeds, namely $\{1993, 1994, 1995, 1996, 1997\}$. These repeated trials yield five sets of incremental results for each method, from which we compute the mean and standard deviation reported in Figure 6b. As shown in the figure, StaR-MoE maintains higher mean accuracy across incremental stages with limited variation across seeds, supporting its robustness to different class orders.



(a) Hyperparameter robustness. (b) Multi-seed evaluation. (c) Running time comparison.

Figure 6: Additional experimental analysis. (a) Sensitivity to λ_{SARA} and λ_{ACR} on ImageNet-A. StaR-MoE maintains stable performance across a wide range of values. (b) Multi-seed evaluation on ImageNet-A. StaR-MoE maintains higher mean accuracy across incremental stages with limited variation across seeds. (c) Running time comparison on CIFAR-100 and ImageNet-R. StaR-MoE maintains competitive training efficiency while achieving the best overall performance.

Table 10: Compatibility with selective expert expansion. Expert Reduction is computed relative to the default expansion of StaR-MoE.

Dataset	Method	Expansion	Expert Reduction	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
VTAB	SEMA	selective	–	90.93	88.12
	StaR-MoE	task-wise	–	94.43	93.79
	StaR-MoE-SE	selective	30.0%	93.38	92.25
ImageNet-A	SEMA	selective	–	67.03	58.06
	StaR-MoE	task-wise	–	71.60	61.75
	StaR-MoE-SE	selective	25.0%	69.33	58.92

E.6 Running Time Comparison

To further evaluate the practical efficiency of StaR-MoE, we compare the training time of different methods on CIFAR-100 and ImageNet-R (10 tasks). All experiments are conducted under the same hardware environment using NVIDIA GeForce RTX 4090 GPUs. As shown in Figure 6c, StaR-MoE maintains competitive training efficiency on both benchmarks while achieving the best overall performance in the main experiments, indicating that the proposed routing stabilization strategy does not introduce excessive computational overhead.

E.7 Compatibility with Selective Expert Expansion

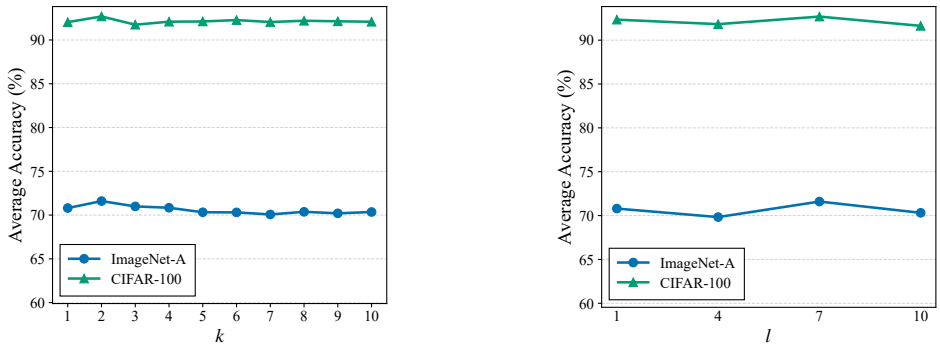
In the main paper, we adopt task-wise expert expansion to isolate the routing-drift problem without confounding it with a separate expansion policy, while each expert is implemented as a lightweight adapter. We further evaluate StaR-MoE with a selective expansion policy inspired by SEMA (Wang et al., 2025). In this variant, denoted as StaR-MoE-SE, new experts are added according to the selective expansion criterion of SEMA, while SARA and ACR remain unchanged. Expansion decisions are made dynamically during StaR-MoE's own training. As shown in Table 10, StaR-MoE-SE reduces the total number of expanded experts compared with the default task-wise expansion strategy, with only a moderate drop in accuracy. Importantly, StaR-MoE-SE still outperforms SEMA, especially on VTAB, suggesting that routing stabilization remains beneficial under selective expert growth.

E.8 Capacity-Matched Comparisons

To control for the effect of model capacity, we conduct a separate comparison with SEMA on ImageNet-A. Unlike StaR-MoE-SE, which determines expansion decisions during its own training, this variant directly reuses the layer-wise expansion decisions recorded from SEMA at each task. We denote this variant as StaR-MoE (SEMA-matched). As shown in Table 11, under the matched

Table 11: Capacity-matched comparison with SEMA on ImageNet-A.

Method	Expansion Schedule	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
SEMA	Selective	67.03	58.06
StaR-MoE (SEMA-matched)	Same as SEMA	**71.12**	**60.96**



(a) Number of active experts.

(b) MoE layer placement.

Figure 7: Impact of MoE design choices. (a) Effect of the number of active experts k. (b) Effect of the MoE starting layer l, where expert expansion is applied from layer l to the last transformer layer.

capacity and expansion schedule, StaR-MoE (SEMA-matched) improves $\bar{\mathcal{A}}/\mathcal{A}_T$ by 4.09/2.90 percentage points over SEMA. These results indicate that its advantage over SEMA cannot be explained solely by additional experts or a larger parameter budget.

E.9 Impact of MoE Design Choices

We further analyze two MoE design choices in StaR-MoE: the number of active experts k and the placement of MoE layers. Figure 7 presents representative results on CIFAR-100 and ImageNet-A. For the number of active experts, StaR-MoE achieves relatively stable performance across different choices of k, and $k = 2$ provides a favorable trade-off between routing specialization and capacity utilization. For MoE layer placement, we vary the starting layer l of expert expansion, where expansion is applied from layer l to the last transformer layer. The setting $l = 7$ corresponds to expanding the last six transformer layers and achieves a practical trade-off between performance and expansion scope. We therefore use this setting as the default configuration.

E.10 Varying the Pre-trained Model

To assess the generality of StaR-MoE, we perform experiments using a ViT-B/16 backbone pre-trained with the self-supervised method DINO (Caron et al., 2021) on ImageNet-R (10 tasks). As shown in Table 12, StaR-MoE demonstrates improved performance over the baselines, indicating its effectiveness across different backbones.

F Storage and Parameter Overhead

F.1 Storage Cost Discussion

We report the method-specific extra storage cost in Figure 4c. The frozen backbone and common components shared by all methods are excluded from this comparison. For APER, the extra storage comes from the prototypes maintained for its prototype-based classifier. For InfLoRA, it comes from the gradient-subspace bases maintained to keep LoRA updates orthogonal to old-task gradient subspaces. For SEMA, it corresponds to the additional expansion-related storage required by its dynamic model expansion. For StaR-MoE, it comes from the class-wise statistics used by SARA.

Table 12: Performance comparison on ImageNet-R (10 tasks) with DINO. We report average accuracy $\bar{\mathcal{A}}$ and last accuracy $\mathcal{A}_T$.

Method	ImageNet-R	
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
L2P	68.77	61.94
DualPrompt	67.65	60.40
CODA-Prompt	72.20	64.63
InfLoRA	76.40	67.91
SD-LoRA	64.61	69.12
SEMA	72.55	64.10
StaR-MoE	**77.46**	**69.72**

Table 13: Learnable parameter comparison on ImageNet-R (10 tasks). Learnable Param. denotes the accumulated learnable parameters after the final task, excluding the frozen backbone.

Method	Learnable Param. (M)
L2P	0.05
DualPrompt	0.25
CODA-Prompt	3.84
APER + Adapter	1.48
InfLoRA	3.69
SD-LoRA	3.69
SEMA	1.35
StaR-MoE	1.88

F.2 Learnable Parameter Overhead

We report the number of accumulated learnable parameters after the final task in Table 13. The frozen ViT backbone is excluded for all methods, since it is shared across compared methods. Among the compared baselines, DualPrompt and CODA-Prompt introduce task-dependent prompt parameters, while SD-LoRA introduces task-dependent LoRA components; these parameters grow linearly with the number of tasks. SEMA does not expand at every task, but its adapter modules can still increase when selective expansion is triggered. Therefore, we report the accumulated method-specific learnable parameters after completing all tasks. StaR-MoE introduces a moderate amount of learnable parameters among representative CIL methods. In particular, its parameter overhead remains much smaller than CODA-Prompt and SD-LoRA.