JUST ENOUGH EXPERTS FOR CONTINUAL LEARNING OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies on continual learning (CL) have shown that data replay is a simple yet effective strategy for mitigating catastrophic forgetting in large language models (LLMs). However, its effectiveness depends heavily on the quantity and quality of rehearsal data, and substantial forgetting can persist when either is inadequate. Moreover, updating shared model parameters across tasks can introduce interference and further degrade performance. To address these limitations, we propose **Adaptive Expansion and Identification (AEI)**, a framework that combines minimal yet sufficient expert expansion with adaptive expert identification. Specifically, AEI encourages newly introduced experts to capture feature directions complementary to those represented by existing experts. As the represented feature subspace grows across tasks, the remaining complementary space becomes increasingly constrained, encouraging compact and effective expert expansion with reduced redundancy and cross-task interference. Meanwhile, AEI employs a context-aware gating module trained with task-identity supervision to adaptively identify and combine the independent experts relevant to each input. Together, these mechanisms support efficient adaptation to new tasks while preserving previously acquired knowledge. Extensive experiments across multiple LLM backbones demonstrate that AEI outperforms competing continual learning methods using only 1% rehearsal data.

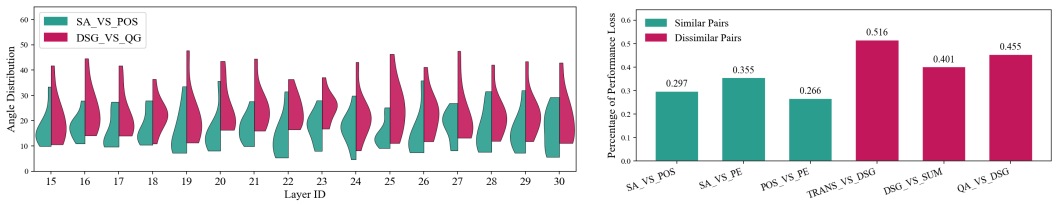
1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities across a broad range of applications (Brown et al., 2020; Touvron et al., 2023; Bai et al., 2023; Guo et al., 2025). Despite these advances, adapting general-purpose LLMs to specialized domains remains challenging, particularly in healthcare (Singhal et al., 2023; Chen et al., 2024), finance (Son et al., 2023; Wang et al., 2023a), and scientific research (Taylor et al., 2022; Haman & Školník, 2024), where reliable performance depends on domain-specific knowledge and capabilities. As knowledge evolves and new tasks emerge, LLMs must continue to learn beyond their initial pretraining. Continual learning (CL) seeks to meet this need by enabling sequential adaptation while preserving previously acquired knowledge. However, fine-tuning on new tasks can substantially degrade performance on earlier tasks, even after only a few gradient updates (Wang et al., 2023c; Zheng et al., 2025). This phenomenon, known as catastrophic forgetting, poses a central challenge to the continual adaptation of LLMs.

Data replay offers a simple yet effective strategy for mitigating forgetting. It supplements current-task training with stored samples from previous tasks (Wang et al., 2024a; Feng et al., 2025) or synthetic samples that approximate historical data distributions (Huang et al., 2024; Wang et al., 2025b). Its effectiveness, however, depends heavily on the quantity and quality of rehearsal data. Insufficient coverage or inaccurate synthetic samples can compromise knowledge retention, motivating careful sample selection (Wang et al., 2024a) and refinement of generated data (Huang et al., 2024).

Moreover, many replay-based approaches update shared model parameters across tasks. When tasks favor different update directions, these shared updates can introduce cross-task interference (Yu et al., 2020) that replay alone may not adequately resolve. Figure 1 illustrates the potential interference between task-specific updates. On SuperNI, certain task pairs exhibit large angles between their task vectors, particularly in higher model layers. Merging their task vectors produces greater performance degradation than merging those of small-angle task pairs.

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.



(a) Angle distribution (b) Performance loss

Figure 1: **(a): Angles between task vectors.** We visualize the angles between the delta weights $\Delta\mathbf{W}_{\text{task}_A}$ and $\Delta\mathbf{W}_{\text{task}_B}$ (i.e., task vectors (Ilharco et al., 2023)) for two representative task pairs from the SuperNI benchmark: SA vs. POS and DSG vs. QG (Wang et al., 2022). **(b): Performance degradation after model merging.** Intuitively, when the task vectors of two tasks form a large angle, merging them by updating the base model with $\Delta\mathbf{W}_{\text{task}_A} + \Delta\mathbf{W}_{\text{task}_B}$ (task-vector addition (Ilharco et al., 2023)) is more likely to cause substantial performance degradation on both tasks. For three similar (small-angle) and three dissimilar (large-angle) task pairs, we report the average performance loss after merging. Further details are provided in Appendix A.

Expert expansion offers a natural way to mitigate such interference by introducing separate expert branches for new tasks (Chen et al., 2023; Zhao et al., 2024; Han et al., 2025; Liang et al., 2025; Zhang et al., 2026). However, existing approaches face two key limitations. First, they often allocate experts with a predetermined capacity, without sufficiently accounting for knowledge already represented by previously learned experts. When a new task shares substantial structure with previous tasks, this fixed allocation can introduce redundant expert parameters and increase statistical error (Bartlett et al., 2019; Jiao et al., 2023; Tan et al., 2024). Second, preserving expert parameters does not ensure accurate expert identification. Inaccurate routing can activate unsuitable experts and degrade performance on previously learned tasks.

To address these limitations, we propose **Adaptive Expansion and Identification (AEI)**, a framework that combines minimal yet sufficient expert expansion with adaptive expert identification. Specifically, AEI formulates continual adaptation as a sequence of task-specific functional increments. By encouraging newly introduced experts to capture feature directions complementary to those represented by previously learned experts, AEI avoids redundant representations of existing knowledge and reduces interference among experts. As the represented feature subspace grows across tasks, AEI progressively reduces redundant expert expansion, thereby helping to reduce statistical error. Building on these independent features, AEI learns a context-aware gating module with task-identity supervision to adaptively identify and combine the experts relevant to each input. Together, these mechanisms enable efficient adaptation to new tasks while mitigating catastrophic forgetting and reducing interference among experts.

We evaluate AEI on the SuperNI (Wang et al., 2022) benchmark using LLaMA-2 (Touvron et al., 2023), LLaMA-3.1 (Grattafiori et al., 2024), and Qwen3 (Team, 2025) models. AEI consistently outperforms the evaluated CL baselines with only 1% rehearsal data. Our contributions are threefold: **i)** We propose an adaptive expert expansion mechanism that encourages the learning of complementary feature directions to limit parameter redundancy and reduce interference among experts, enabling minimal yet sufficient expert expansion with improved robustness; **ii)** We develop a context-aware gating module that adaptively identifies and combines relevant experts, enabling effective reuse of accumulated experts while reducing cross-task interference; **iii)** Extensive experiments across multiple LLM backbones and continual learning settings demonstrate that AEI achieves strong knowledge retention and effective adaptation to new tasks with only 1% rehearsal data.

2 Related Work

Continual Learning (CL). CL aims to adapt models to new tasks and domains while preserving previously acquired knowledge. **Parameter importance-based** methods seek to freeze or regularize parameters that are important for previously learned tasks, directing updates primarily toward less critical parameters when learning new tasks (Kirkpatrick et al., 2017; Feng et al., 2025; Chen et al., 2025; Wang et al., 2025c). For instance, Wang et al. (2025c) freeze selected rows in the down-projection matrix of the Feed-Forward Network (FFN) layer, while Chen et al. (2025) regularize

large-magnitude entries in the delta-weight matrices. A key limitation is that parameters important for previous tasks may also be crucial for new ones. Excessive constraints can therefore impede the acquisition of new knowledge. **Gradient orthogonality-based** methods constrain current-task weight updates to a subspace orthogonal to the feature space of previous tasks (Wang et al., 2021; Saha et al., 2021; Liang & Li, 2023; Wang et al., 2023b; Liang & Li, 2024; Liang et al., 2025; Zhu et al., 2025). To approximate this feature space, Wang et al. (2021) store a covariance matrix for each layer, while Saha et al. (2021) maintain feature banks across all layers. These approaches, however, incur substantial storage overhead, which becomes prohibitive for modern LLMs with large hidden dimensions. **Data replay-based** methods incorporate a small subset of data from previous tasks (Wang et al., 2024a; Feng et al., 2025) or generate synthetic samples to approximate historical data distributions (Huang et al., 2024; Wang et al., 2025b). As discussed in Sec. 1, their effectiveness depends heavily on the quantity and quality of rehearsal data. **Model growth-based** methods dynamically expand model capacity to accommodate new tasks (Qin et al., 2022; Chen et al., 2023; Wang et al., 2025a; Han et al., 2025). For example, Chen et al. (2023) and Wang et al. (2025a) introduce additional FFN branches, while Han et al. (2025) also increase the number of attention heads.

Mixture-of-Experts (MoE). Introduced by Jacobs et al. (1991), MoE is a powerful statistical machine learning framework that extends conventional mixture models (Lindsay, 1995) via a divide-and-conquer strategy. Since its integration into language modeling and machine translation by Shazeer et al. (2017), MoE has substantially reduced training energy consumption and computational cost while improving overall zero-, one-, and few-shot performance across diverse NLP tasks and domains (Kudugunta et al., 2021; Du et al., 2022). For LLMs, MoE enables efficient model scaling, achieving stronger performance while maintaining manageable computational costs, making it a mainstream architecture for modern LLMs (Lepikhin et al., 2021; Fedus et al., 2022; Liu et al., 2024; Dubey et al., 2024; Team et al., 2024; Guo et al., 2025). However, controlling expert redundancy and identifying the most relevant experts for each input remain challenging.

In this work, we propose an expandable expert framework for continual learning of LLMs with just enough experts. Unlike conventional MoE architectures with a predefined set of experts, our framework progressively introduces and trains task-specific experts as new tasks arrive. A context-aware gating module trained with task-identity supervision adaptively identifies and combines experts relevant to each input. Together, these mechanisms enable efficient adaptation to new tasks while mitigating catastrophic forgetting and interference among experts.

3 METHOD

3.1 PRELIMINARIES

Continual Learning (CL). Let $\mathcal{M}_0$ denote a pretrained large language model (LLM). We aim to sequentially adapt $\mathcal{M}_0$ to a stream of downstream instruction-tuning tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$, while preserving the knowledge acquired from previously learned tasks. For each task $\mathcal{T}_t$, we are given a training set $\hat{\mathcal{D}}_t = \{(\boldsymbol{q}_i, \boldsymbol{a}_i)\}_{i=1}^{n_t}$, where $\boldsymbol{q}_i$ denotes an input prompt and $\boldsymbol{a}_i$ denotes its corresponding answer. To mitigate catastrophic forgetting, we adopt a lightweight replay strategy. At stage t, the training data consist of the current task data together with a small set of rehearsal samples from previously learned tasks:

$$\mathcal{D}_t = \hat{\mathcal{D}}_t \cup \bigcup_{i=1}^{t-1} \left(r \cdot \hat{\mathcal{D}}_i \right), \tag{1}$$

where r denotes the replay ratio and $r \cdot \hat{\mathcal{D}}_i$ denotes a subset sampled from $\hat{\mathcal{D}}_i$ at ratio r. In AEI, replay samples are used primarily to supervise expert identification and routing rather than to relearn previously acquired task knowledge. Thus, AEI requires only a small replay ratio in practice (e.g., $r = 1\%$).

Transformer. A standard Transformer consists primarily of Multi-Head Self-Attention (MHSA) layers and Feed-Forward Network (FFN) layers. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the hidden-state matrix at a given layer, where n is the sequence length and d is the hidden dimension. The forward computations

for an MHSA layer $\mathcal{A}(\cdot)$ and an FFN layer $\mathcal{F}(\cdot)$ are given by

$$\mathcal{A}(\mathbf{X}) = \sum_{i=1}^{h} \sigma\left(\frac{\mathbf{X}\mathbf{W}_{q_i}\mathbf{W}_{k_i}^{\top}\mathbf{X}^{\top}}{\sqrt{d_h}}\right)\mathbf{X}\mathbf{W}_{v_i}\mathbf{W}_{o_i}^{\top}, \tag{2}$$

and

$$\mathcal{F}(\mathbf{X}) = f\left(\mathbf{X}\mathbf{W}_{\text{up}}\right)\mathbf{W}_{\text{down}}^{\top}, \tag{3}$$

where $\sigma(\cdot)$ denotes the softmax function. For the i-th attention head, $\mathbf{W}_{q_i}, \mathbf{W}_{k_i}, \mathbf{W}_{v_i}, \mathbf{W}_{o_i} \in \mathbb{R}^{d\times d_h}$ denote the query, key, value, and output projection matrices, respectively. Here, h denotes the number of attention heads and d_h denotes the dimension of each head. For the FFN, $\mathbf{W}_{\text{up}}, \mathbf{W}_{\text{down}} \in \mathbb{R}^{d\times d_m}$, where d_m is the FFN hidden dimension, and $f(\cdot)$ denotes an element-wise activation function, such as GeLU or Gated Linear Units (GLU) (Shazeer, 2020).

A Transformer typically contains L stacked blocks. Each block $\mathcal{B}(\cdot)$ consists of an MHSA layer followed by an FFN layer, with residual connections:

$$\begin{aligned}\mathbf{X}^{\text{MHSA}} &= \mathbf{X} + \mathcal{A}(\mathbf{X}),\\ \mathcal{B}(\mathbf{X}) &= \mathbf{X}^{\text{MHSA}} + \mathcal{F}(\mathbf{X}^{\text{MHSA}}).\end{aligned} \tag{4}$$

We denote the MHSA and FFN layers in the l-th block of the pretrained model $\mathcal{M}_0$ by $\mathcal{A}_0^l$ and $\mathcal{F}_0^l$, respectively. After learning task $\mathcal{T}_t$, their counterparts in the updated model $\mathcal{M}_t$ are denoted by $\mathcal{A}_t^l$ and $\mathcal{F}_t^l$. The corresponding Transformer blocks are denoted by $\mathcal{B}_0^l$ and $\mathcal{B}_t^l$.

3.2 Overall Framework

AEI performs continual learning through an expandable framework of FFN experts. We formulate the adaptation of each FFN layer as a sequence of task-specific functional increments. For clarity, we omit the layer index l in this subsection. The FFN after learning task $\mathcal{T}_t$ can be written as

$$\begin{aligned}\mathcal{F}_t(\boldsymbol{x}) &= \mathcal{F}_0(\boldsymbol{x}) + \sum_{i=1}^{t}\left(\mathcal{F}_i(\boldsymbol{x}) - \mathcal{F}_{i-1}(\boldsymbol{x})\right)\\ &= \mathcal{F}_0(\boldsymbol{x}) + \sum_{i=1}^{t-1}\Delta\mathcal{F}_i(\boldsymbol{x}) + \Delta\mathcal{F}_t(\boldsymbol{x}),\end{aligned} \tag{5}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ denotes a token representation and $\Delta\mathcal{F}_i(\cdot)$ represents the functional increment introduced when learning task $\mathcal{T}_i$.

We parameterize each increment as an independent gated expert branch:

$$\begin{aligned}\Delta\mathcal{F}_i(\boldsymbol{x}) &= \boldsymbol{\alpha}_i(\boldsymbol{x})\cdot\mathcal{F}_i^e(\boldsymbol{x})\\ &= \boldsymbol{\alpha}_i(\boldsymbol{x})\cdot f(\boldsymbol{x}\mathbf{W}_{\text{up}}^i){\mathbf{W}_{\text{down}}^i}^{\top}.\end{aligned} \tag{6}$$

Here, $\mathbf{W}_{\text{up}}^i, \mathbf{W}_{\text{down}}^i \in \mathbb{R}^{d\times d_{m'}}$ are the learnable parameters of the i-th expert, $d_{m'}$ denotes its hidden dimension, and $\boldsymbol{\alpha}_i(\boldsymbol{x}) \in \mathbb{R}$ is an adaptive gating weight that controls the contribution of expert $\mathcal{F}_i^e(\boldsymbol{x})$ to input $\boldsymbol{x}$. When learning a new task, we train the newly introduced expert while keeping the pretrained backbone and all previously learned FFN expert $\mathcal{F}_i^e(\boldsymbol{x})$ frozen. This isolates newly acquired knowledge in a separate parameter space, reducing interference with previously learned tasks. This separation makes previously learned experts individually accessible, allowing the gating module to identify and combine relevant experts for each input.

To balance stability and plasticity, AEI follows two principles: complementary expert expansion and adaptive expert identification. First, the expert learned for task $\mathcal{T}_t$ should capture feature directions complementary to those represented by previously learned experts. Specifically, for any $j < t$, we encourage

$$\mathcal{F}_t^e(\boldsymbol{x})\mathcal{F}_j^e(\boldsymbol{x})^{\top} \approx \mathbf{0}. \tag{7}$$

Intuitively, this encourages the new expert to focus on feature directions that remain insufficiently represented by previously learned experts. As the represented feature subspace grows across tasks, the complementary space shrinks, progressively limiting redundant expert parameters. This is consistent with learning-theoretic results showing that appropriate structural constraints can reduce statistical error (Bartlett et al., 2019; Jiao et al., 2023; Tan et al., 2024).

Second, based on these independent features, the gating module should adaptively identify and combine experts relevant to each input. By assigning input-dependent weights $\boldsymbol{\alpha}_i(\boldsymbol{x})$, it combines knowledge from the accumulated experts to meet the needs of each input. In particular, when $\boldsymbol{\alpha}_i(\boldsymbol{x}) \approx 0$, the contribution of the i-th expert is negligible.

Based on the training data 1, the overall learning objective at stage t comprises three components:

$$\mathcal{L}_t = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{gate}}, \tag{8}$$

where $\mathcal{L}_{\text{task}}$ is the next-token prediction loss on the training data $\mathcal{D}_t$, $\mathcal{L}_{\text{reg}}$ encourages complementarity between newly introduced and previously learned experts, and $\mathcal{L}_{\text{gate}}$ is the expert identification loss. The following two subsections detail adaptive expert expansion and context-aware expert identification, respectively.

3.3 Adaptive Expert Expansion

Eq. (7) encourages each newly introduced expert to capture feature directions complementary to those represented by previously learned experts. Intuitively, when existing feature directions largely capture a new task, only a small number of additional parameters are needed; otherwise, the new expert should explore directions that remain insufficiently represented. Because the complete training data of previously learned tasks are unavailable, directly enforcing Eq. (7) in feature space is impractical. We therefore apply structured regularization to the up-projection matrix of each newly introduced FFN expert, encouraging exploration of new feature space.

At stage t, let $\mathcal{W}_t = \text{Span}(\mathbf{W}_{\text{up}}^0, \mathbf{W}_{\text{up}}^1, \ldots, \mathbf{W}_{\text{up}}^{t-1})$ denote the subspace spanned by the up-projection matrices of the pretrained backbone and all previously learned experts, and let $\mathbf{U}_t = [\mathbf{u}_1, \ldots, \mathbf{u}_{m_t}] \in \mathbb{R}^{d \times m_t}$ contain an orthonormal basis of $\mathcal{W}_t$. For the newly introduced expert, we write $\mathbf{W}_{\text{up}}^t = [\mathbf{w}_1, \ldots, \mathbf{w}_{d_{m'}}] \in \mathbb{R}^{d \times d_{m'}}$. We decompose it into components parallel and orthogonal to $\mathcal{W}_t$:

$$\mathbf{W}^{\parallel} = \mathbf{U}_t \mathbf{U}_t^{\top} \mathbf{W}_{\text{up}}^t, \qquad \mathbf{W}^{\perp} = \mathbf{W}_{\text{up}}^t - \mathbf{W}^{\parallel}. \tag{9}$$

Here, $\mathbf{W}^{\parallel} = [\mathbf{w}_1^{\parallel}, \ldots, \mathbf{w}_{d_{m'}}^{\parallel}]$ captures directions within the existing feature subspace, whereas $\mathbf{W}^{\perp} = [\mathbf{w}_1^{\perp}, \ldots, \mathbf{w}_{d_{m'}}^{\perp}]$ captures complementary directions outside the existing feature subspace.

We then impose a group-sparsity regularizer on both components:

$$\mathcal{L}_{\text{reg}} = \frac{\lambda}{d_{m'}} \sum_{i=1}^{d_{m'}} \left(\|\mathbf{w}_i^{\parallel}\|_2 + \|\mathbf{w}_i^{\perp}\|_2 \right), \tag{10}$$

where λ controls the regularization strength. We apply this regularizer to all expert-augmented FFN layers. Before learning the t-th expert, $\mathbf{U}_t$ is computed once by applying singular value decomposition (SVD) to the concatenated up-projection matrices spanning $\mathcal{W}_t$ and remains fixed throughout training on task $\mathcal{T}_t$. This makes the computational overhead of Eq. (10) small; see Appendix C for details.

Lemma 1 (Orthogonality of the regularization gradients)**.** *For every i such that $\mathbf{w}_i^{\parallel} \neq \mathbf{0}$ and $\mathbf{w}_i^{\perp} \neq \mathbf{0}$, the gradients of $\mathcal{L}_{\text{reg}}$ with respect to the parallel and orthogonal components are orthogonal:*

$$\left\langle \nabla_{\mathbf{w}_i^{\parallel}} \mathcal{L}_{\text{reg}}, \nabla_{\mathbf{w}_i^{\perp}} \mathcal{L}_{\text{reg}} \right\rangle = 0. \tag{11}$$

Lemma 1 shows that shrinking one component does not directly interfere with the optimization of the other, because the two components lie in orthogonal subspaces. Consequently, Eq. (10) naturally induces three sparsity patterns. Specifically, (i) $\|\mathbf{w}_i^{\parallel}\|_2 > 0$ and $\|\mathbf{w}_i^{\perp}\|_2 = 0$ indicate that neuron i reuses or reinforces previously represented features; (ii) $\|\mathbf{w}_i^{\parallel}\|_2 = 0$ and $\|\mathbf{w}_i^{\perp}\|_2 > 0$ indicate that

neuron i captures new feature directions; and (iii) $\|\mathbf{w}_i^{\parallel}\|_2 = \|\mathbf{w}_i^{\perp}\|_2 = 0$ indicates that neuron i is redundant and can be removed. The proof is provided in Appendix D. In practice, after training with AdamW using the regularizer in Eq. (10), we prune hidden neurons whose up-projection weights satisfy $\|\mathbf{w}_i\|_2 < \epsilon$, where ϵ is a pruning threshold (e.g., $\epsilon = 10^{-1}$), following common practice in FFN pruning (Li et al., 2024; He & Lin, 2025).

3.4 Context-Aware Gating Module

To identify the experts relevant to each input, we use the task identity of each replay token as auxiliary supervision for the gating module. This task-identity information is naturally available in rehearsal data but is typically overlooked by existing replay-based CL methods (Zhao et al., 2024; Huang et al., 2024; Wang et al., 2025b). Existing gating approaches typically route tokens independently using autoencoders (Aljundi et al., 2017; Yu et al., 2024), linear models (Chen et al., 2023; Wang et al., 2025a), or MLPs (Zhao et al., 2024; Liang et al., 2025). However, token generation in LLMs is inherently contextual: an incorrect routing decision early in generation may affect subsequent tokens and ultimately lead to an incorrect response. To incorporate context into expert routing, we introduce an attention-based gating module that aggregates information up to the current token to determine expert weights.

Since representations in higher Transformer blocks tend to be more task-specific (see Appendix E), we build the gating module on the representations from an intermediate block l_0 of the pretrained model $\mathcal{M}_0$. The resulting gating weights are shared by all expert-augmented FFN layers above l_0. For any $l > l_0$, Eq. (5) becomes

$$\mathcal{F}_t^l(\boldsymbol{x}^{l-1}) = \mathcal{F}_0^l(\boldsymbol{x}^{l-1}) + \sum_{i=1}^{t} \boldsymbol{\alpha}_i(\boldsymbol{x}^{l_0}) \cdot \mathcal{F}_i^{e^l}(\boldsymbol{x}^{l-1}). \tag{12}$$

Here, $\boldsymbol{x}^{l_0}$ and $\boldsymbol{x}^{l-1}$ correspond to the same token and are produced by blocks $\mathcal{B}_0^{l_0}$ and $\mathcal{B}_t^{l-1}$, respectively. For layers $l \le l_0$, no experts are introduced, so the lower blocks remain identical to those of $\mathcal{M}_0$.

For the k-th token $\boldsymbol{x}_k^{l_0} \in \mathbb{R}^d$, let $\mathbf{X}_{1:k} = [\boldsymbol{x}_1^{l_0}, \ldots, \boldsymbol{x}_k^{l_0}]^\top \in \mathbb{R}^{k\times d}$ denote the sequence of token representations up to and including token k. The gating vector is computed as

$$\boldsymbol{\alpha}(\boldsymbol{x}_k^{l_0}) = \sigma\Big(\sum_{i=1}^{h} \sigma\left(\frac{\boldsymbol{x}_k^{l_0}\mathbf{W}_{q_i}^t \mathbf{W}_{k_i}^{t\ \top} \mathbf{X}_{1:k}^\top}{\sqrt{d_h}}\right) \mathbf{X}_{1:k}\mathbf{W}_{v_i}^t \mathbf{W}_{o_i}^{t\ \top}\Big). \tag{13}$$

Here, $\mathbf{W}_{q_i}^t, \mathbf{W}_{k_i}^t, \mathbf{W}_{v_i}^t \in \mathbb{R}^{d\times d_h}$ and $\mathbf{W}_{o_i}^t \in \mathbb{R}^{d_h\times t}$ are trainable gating parameters. When a new task $\mathcal{T}_{t+1}$ arrives, we expand only the output projection by one column, yielding $\mathbf{W}_{o_i}^{t+1} \in \mathbb{R}^{d_h\times(t+1)}$. The parameters learned at stage t are used to initialize the gating module at stage $t+1$, and the entire module is then updated for the new task.

We train the gating module using task-identity supervision. Let $c_k \in \{1, \ldots, t\}$ denote the task from which token $\boldsymbol{x}_k^{l_0}$ originates. The gating objective is

$$\mathcal{L}_{\text{gate}} = -\sum_{i=1}^{t} \boldsymbol{y}_i^k \log(\boldsymbol{\alpha}_i(\boldsymbol{x}_k^{l_0})). \tag{14}$$

Using one-hot task labels for $\boldsymbol{y}^k \in \mathbb{R}^t$ tends to produce overly confident routing decisions and discourage reuse of previously learned experts. Standard label smoothing is also unsuitable for CL because it assigns nonzero probability to all non-target experts, including experts introduced *after* the task from which a replay sample originates. These later experts should not contribute to previously learned tasks. We therefore introduce a temporally asymmetric label-smoothing strategy:

$$\boldsymbol{y}_i^k = \begin{cases} 1, & c_k = 1,\ i = 1, \\ 1-\varepsilon, & c_k > 1,\ i = c_k, \\ \varepsilon/(c_k - 1), & c_k > 1,\ i < c_k, \\ 0, & i > c_k, \end{cases} \tag{15}$$

Table 1: Results on the SuperNI-10 benchmark under different task orders (LLaMA-2-7B).

Method	Order 1		Order 2		Order 3		Avg.	
	AR ↑	BWT ↑	AR ↑	BWT ↑	AR ↑	BWT ↑	AR ↑	BWT ↑
SuperNI 5 Tasks								
EWC	49.25	-4.21	48.83	-5.47	49.02	-6.01	49.03	-5.23
KMSel	49.73	-5.22	50.14	-4.17	50.12	-3.61	50.00	-4.33
SAPT	50.47	-3.75	51.04	-2.98	50.22	-4.37	50.58	-3.70
OAdapter	50.71	-2.71	51.26	-2.17	50.89	-2.95	50.95	-2.61
OLoRA	51.32	-1.80	50.98	-1.93	51.14	-1.82	51.15	-1.85
GoDMoE	52.94	-0.77	52.71	-0.89	53.01	-0.73	52.89	-0.80
InfLoRA	52.39	-1.21	52.28	-1.77	52.51	-1.33	52.39	-1.44
SSR	51.33	-1.97	52.41	-1.18	52.02	-1.01	51.92	-1.39
SERS	52.90	-0.55	53.01	-0.27	52.84	-0.63	52.92	-0.48
Gain	53.15	-0.13	52.93	-0.41	53.04	-0.59	53.04	-0.38
AEI (Ours)	**53.46**	**+0.03**	**53.26**	**-0.02**	**53.26**	**-0.19**	**53.33**	**-0.06**
SuperNI 10 Tasks								
EWC	57.54	-3.92	57.46	-4.39	57.83	-4.36	57.61	-4.22
KMSel	59.13	-5.88	60.71	-5.39	60.44	-7.17	60.09	-6.15
SAPT	62.51	-2.06	61.90	-2.81	62.29	-2.30	62.23	-2.39
OAdapter	59.34	-5.25	58.72	-5.95	59.62	-4.96	59.23	-5.39
OLoRA	59.76	-3.76	59.36	-4.15	60.14	-3.89	59.75	-3.93
GoDMoE	61.57	-3.08	61.62	-2.84	61.34	-3.48	61.51	-3.13
InfLoRA	62.89	-3.04	62.21	-3.42	62.76	-3.35	62.62	-3.27
SSR	62.29	-1.84	62.64	-1.86	62.36	-3.95	62.43	-2.55
SERS	63.42	-1.72	63.45	-1.11	64.46	-2.27	63.78	-1.70
Gain	63.14	-1.07	63.69	-1.32	63.81	-1.87	63.55	-1.42
AEI (Ours)	**65.56**	**+0.04**	**65.32**	**+0.01**	**65.64**	**-0.10**	**65.51**	**-0.01**

where $\varepsilon \in [0, 1)$ controls knowledge reuse from previously learned experts. For a token originating from task $\mathcal{T}_{c_k}$, its corresponding expert receives target weight $1 - \varepsilon$ when $c_k > 1$, while all experts learned before $\mathcal{T}_{c_k}$ equally share the remaining mass ε. In contrast, every expert learned after $\mathcal{T}_{c_k}$ receives zero target weight. This design encourages reuse of previously acquired knowledge while explicitly preventing later experts from interfering with previously learned tasks.

Finally, rehearsal data are used primarily to learn expert identification rather than to relearn task-specific knowledge. Identifying the appropriate expert mixture is substantially easier than relearning an entire task, so replay samples need not be numerous or of particularly high quality. In practice, a replay ratio as small as $r = 1\%$ is sufficient to preserve task awareness with negligible storage and computational overhead.

4 Experiments

4.1 Setups

Datasets. We evaluate our method on the SuperNI benchmark (Wang et al., 2022). **i) SuperNI-10** contains a diverse set of instruction-tuning tasks. Following prior work (Huang et al., 2024; Wang et al., 2025b), we select ten tasks covering translation, question answering, and information extraction, and evaluate both 5-task and 10-task settings. Each task contains 2,000 training instances and 500 evaluation instances. We further consider three task orders, consistent with (Huang et al., 2024; Wang et al., 2025b). **ii) SuperNI-Long** extends SuperNI-10 to longer task streams. Based on the same task categories, we sample 2, 3, 4, and 5 tasks per category, yielding continual learning sequences of 20, 30, 40, and 50 tasks. This setting is designed to better reflect long task streams with less explicit task

Table 2: SuperNI-Long results using LLaMA-3.1-8B and Qwen3-4B as the initial LLM.

Method	20 Tasks		30 Tasks		40 Tasks		50 Tasks	
	AR ↑	BWT ↑	AR ↑	BWT ↑	AR ↑	BWT ↑	AR ↑	BWT ↑
LLaMA3.1-8B								
O-LoRA	59.76	-5.17	57.34	-5.70	55.72	-6.13	54.84	-5.92
InfLoRA	63.72	-3.19	61.10	-3.51	59.67	-4.13	57.31	-3.56
SERS	62.91	-4.72	60.97	-4.74	59.53	-5.04	57.81	-4.68
Gain	63.62	-0.85	61.92	-0.82	60.16	-1.14	58.36	-1.74
AEI (Ours)	**66.96**	**-0.01**	**65.38**	**+0.03**	**63.27**	**+0.06**	**61.57**	**+0.05**
Qwen3-4B								
O-LoRA	60.73	-3.36	58.29	-2.79	54.91	-4.27	53.28	-4.67
InfLoRA	63.25	-2.88	60.07	-3.11	58.06	-3.98	55.42	-4.16
SERS	63.01	-3.87	60.56	-4.03	58.87	-3.99	56.14	-4.24
Gain	63.59	-1.12	60.26	-1.56	59.23	-1.78	56.82	-1.84
AEI Ours	**66.19**	**-0.15**	**63.60**	**+0.12**	**62.33**	**+0.02**	**60.81**	**+0.03**

boundaries. For each task, we randomly sample 1,000 instances for training and 250 for evaluation, with an additional 100 training instances held out for validation. All tasks are arranged in a fixed order. Further details on task configurations and evaluation metrics are provided in Appendix F.

Implementation Details. For *SuperNI-10*, we use the pretrained LLaMA-2-7B model (Touvron et al., 2023) as the initial LLM and set the regularization coefficient in Eq. (10) to $\lambda = 0.1$ for all tasks. For *SuperNI-Long*, we use LLaMA-3.1-8B (Grattafiori et al., 2024) and Qwen3-4B (Team, 2025), and set $\lambda = 0.5$ because this setting involves substantially fewer training iterations per task. Additional implementation details are provided in Appendix F.

Baselines. We compare our method with a range of state-of-the-art baselines, including *EWC* (Kirkpatrick et al., 2017); *KMSel*, which selects real samples from previous tasks for replay using K-means clustering; *OLoRA* (Wang et al., 2023b); *InfLoRA* (Liang & Li, 2024); *SSR* (Huang et al., 2024); *SAPT* (Zhao et al., 2024); *OAdapter* (Wan et al., 2025); *Gain* (Liang et al., 2025) *SERS* (Wang et al., 2025b); and *GoDMoE* (Zhang et al., 2026). For a fair comparison, we augment each non-rehearsal baseline with replay using 1% of the real data from each previous task.

4.2 Results and Analysis

Main Results. As shown in Tables 1 and 2, AEI consistently achieves the best overall performance across different task orders, sequence lengths, and LLM backbones. On SuperNI-10, AEI obtains the highest average AR for both the 5-task and 10-task settings, while maintaining BWT close to zero, with its advantage becoming more pronounced as the task sequence grows. This trend remains consistent on SuperNI-Long: across 20–50 tasks, AEI substantially outperforms the strongest baselines on both LLaMA-3.1-8B and Qwen3-4B, while keeping BWT around zero even for the longest sequences. These results demonstrate that AEI effectively mitigates catastrophic forgetting while maintaining strong adaptation capability, and scales reliably to substantially longer continual learning sequences.

Ablation Study. Table 3 evaluates the contributions of the adaptive expert regularization term $\mathcal{L}_{\text{reg}}$ and the context-aware router. Removing both components leads to clear performance degradation, showing that expert expansion alone is insufficient. Adding $\mathcal{L}_{\text{reg}}$ consistently improves AR and BWT, while the router yields further gains by enabling more accurate expert identification. Combining both components achieves the best performance, confirming their complementary roles in balancing knowledge acquisition and retention.

Sample Efficiency. We investigate the rehearsal efficiency of AEI by varying the number of replay samples per task, with results reported in Table 4. AEI consistently maintains strong performance under different replay budgets, demonstrating its effectiveness with limited rehearsal data. In particular, reducing the number of replay samples from 9 to 4 only leads to marginal performance

Table 3: Ablation study (Order 1).

Router	$\mathcal{L}_{\text{reg}}$	5 Tasks		10 Tasks	
		AR ↑	BWT ↑	AR ↑	BWT ↑
		51.68	-2.36	62.72	-3.46
	✓	52.34	-1.53	64.41	-2.55
✓		53.13	-0.04	65.12	-0.01
✓	✓	**53.46**	**+0.03**	**65.56**	**+0.04**

Table 4: Rehearsal efficiency (Order 1).

# Replay	Method	5 Tasks		10 Tasks	
		AR ↑	BWT ↑	AR ↑	BWT ↑
9	AEI	53.37	-0.05	65.40	-0.11
	Gain	52.65	-0.15	61.60	-1.05
4	AEI	53.05	-0.14	65.12	-0.06
	Gain	51.82	-0.76	61.00	-0.46

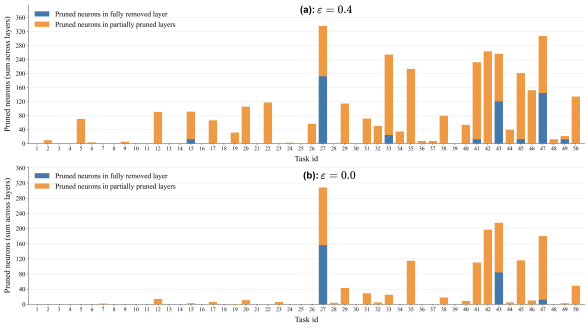


Figure 2: Pruned parameters using different ε values.

Table 5: Cost comparison using HuggingFace on a single NVIDIA RTX 6000 GPU (LLaMA3.1-8B). TT and IT refer to training and inference time across the 50 SuperNI tasks, respectively. Param. denotes the additional parameters introduced by continual learning methods.

Method	Param.	TT	IT
Gain	21.0M	∼5.41h	∼0.79h
SERS	-	∼25.6h	∼0.40h
AEI	23.5M	∼5.13h	∼0.44h

degradation, while AEI continues to outperform the baseline method, especially in the 10-task setting. These results indicate that AEI can effectively preserve historical knowledge without relying on large amounts of rehearsal data.

Adaptive Expert Expansion Analysis. Figure 2 illustrates the number of pruned neurons under different ε values, where fully removed layers and partially pruned layers indicate redundant and partially reusable expert components, respectively. The results show that AEI performs task-dependent parameter pruning rather than uniformly maintaining a fixed expert capacity. For tasks whose knowledge can be sufficiently represented by previously learned experts, a large proportion of newly introduced parameters are removed, while tasks requiring new knowledge retain more task-specific neurons. Moreover, varying ε changes the degree of knowledge reuse and correspondingly affects the pruning pattern, demonstrating that AEI can flexibly balance knowledge reuse and expert expansion.

Cost Analysis. Table 5 compares the computational and parameter costs of different continual learning methods on the 50-task SuperNI setting. AEI introduces 23.5M additional parameters, only slightly more than Gain (21.0M), while achieving lower training and inference costs. Specifically, AEI requires approximately 5.13 hours for training, comparable to Gain (5.41 hours) and substantially faster than SERS (25.6 hours). Its inference time is also reduced from 0.79 hours for Gain to 0.44 hours, and remains close to SERS (0.40 hours). These results indicate that AEI achieves strong continual learning performance with only modest additional parameter and computational overhead.

5 Conclusion

We propose **Adaptive Expansion and Identification (AEI)** for continual learning in large language models. AEI encourages newly introduced experts to capture feature directions complementary to those represented by existing experts, enabling compact expert expansion while reducing redundancy and cross-task interference. In addition, AEI leverages task-identity supervision to learn a context-aware gating module that adaptively identifies and combines the experts relevant to each input. Extensive experiments across multiple LLM backbones demonstrate that AEI effectively mitigates catastrophic forgetting, remains robust to task-order variations and long task sequences, and outperforms competing continual learning methods using only 1% rehearsal data.

AI USE STATEMENT

We used generative AI tools to assist with code development and manuscript writing, including improving clarity and readability. All AI-assisted code and text were reviewed and verified by the authors. We take full responsibility for the final content of this work, including the implementation, results, and claims.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7120–7129, 2017. doi: 10.1109/CVPR.2017.753.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019. URL http://jmlr.org/papers/v20/17-612.html.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jinpeng Chen, Runmin Cong, Yuzhi Zhao, Hongzheng Yang, Guangneng Hu, Horace Ip, and Sam Kwong. SEFE: Superficial and essential forgetting eliminator for multimodal continual instruction tuning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=teJdFzLnKh.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. Huatuogpt-o1, towards medical complex reasoning with llms. *arXiv preprint arXiv:2412.18925*, 2024.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pp. 5383–5395. PMLR, 2023.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S. Yu, Xu Chu, and Xiao-Ming Wu. Recurrent knowledge identification and fusion for language model continual learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27396–27413, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1328. URL https://aclanthology.org/2025.acl-long.1328/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Michael Haman and Milan Školník. Using chatgpt to conduct a literature review. *Accountability in research*, 31(8):1244–1246, 2024.

Xue Han, Yitong Wang, Junlan Feng, wenchun.gao, Qian Hu, and Chao Deng. LOIRE: Lifelong learning on incremental data via pre-trained language model growth efficiently. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=F5PlYMC5ik.

Jiujun He and Huazhen Lin. Olica: Efficient structured pruning of large language models without retraining. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hhhcwCgyM1.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.77. URL https://aclanthology.org/2024.acl-long.77/.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991.

Yuling Jiao, Guohao Shen, Yuanyuan Lin, and Jian Huang. Deep nonparametric regression on approximate manifolds: Nonasymptotic error bounds with polynomial prefactors. *The Annals of Statistics*, 51(2):691–716, 2023.

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.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3577–3599, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Guangyan Li, Yongqiang Tang, and Wensheng Zhang. LoRAP: Transformer sub-layers deserve differentiated structured compression for large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=mhI5nc5QwX.

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.

Yan-Shuo Liang, Jiarui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=lVV7F0piDK.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Bruce G Lindsay. Mixture models: Theory, geometry and applications. In *NSF-CBMS Regional Conference Series in Probability and Statistics*, pp. i–163. JSTOR, 1995.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. ELLE: Efficient lifelong pre-training for emerging data. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2789–2810, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.220. URL https://aclanthology.org/2022.findings-acl.220/.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3AOj0RCNC2.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Guijin Son, Hanearl Jung, Moonjeong Hahm, Keonju Na, and Sol Jin. Beyond classification: Financial reasoning in state-of-the-art language models. In *Proceedings of the Fifth Workshop on Financial Technology and Natural Language Processing and the Second Multimodal AI For Financial Forecasting*, pp. 34–44, 2023.

Zhiyao Tan, Ling Zhou, and Huazhen Lin. Generative adversarial learning with optimal input dimension and its adaptive generator architecture. *arXiv preprint arXiv:2405.03723*, 2024.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Zhiyi Wan, Wanrou Du, Liang Li, Miao Pan, and Xiaoqi Qin. Budget-adaptive adapter tuning in orthogonal subspaces for continual learning in llms. *arXiv e-prints*, pp. arXiv–2505, 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.

Jun Wang, Liang Ding, Shuai Wang, Hongyu Li, Yong Luo, Huangxuan Zhao, Han Hu, and Bo Du. Self-evolving pseudo-rehearsal for catastrophic forgetting with task similarity in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=jR1lvwexLt.

Neng Wang, Hongyang Yang, and Christina Wang. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023a.

Runyu Wang, Peng Ping, Zhengyu Guo, Xiaoye Zhang, Quan Shi, Liting Zhou, and Tianbo Ji. Loki: Low-damage knowledge implanting of large language models. *arXiv preprint arXiv:2505.22120*, 2025c.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 184–193, 2021.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023b.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023c.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 663–677, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.37. URL https://aclanthology.org/2024.naacl-long.37/.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024b.

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.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Ying Zhang, Xingyue Guo, Yu Zhao, Xuhui Sui, Baohang Zhou, Xinying Qian, and Xiaojie Yuan. Grow-on-demand: Sparse and adaptive expert expansion for continual instruction tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 28474–28482, 2026.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11641–11661, 2024.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ScI7IlKGdI.

Hao Zhu, Yifei Zhang, Junhao Dong, and Piotr Koniusz. Bilora: Almost-orthogonal parameter spaces for continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25613–25622, 2025.

A TASK CONFLICT.

Delta Weights and Model Merging. Let $\mathbf{W}_0^{(l)}$ denote the pretrained weight of the l-th layer of an LLM, and let $\mathbf{W}_A^{(l)}$ be the corresponding weight after fine-tuning on task A. The *delta weight* (or *task vector*) for task A is defined as

$$\Delta\mathbf{W}_A^{(l)} = \mathbf{W}_A^{(l)} - \mathbf{W}_0^{(l)}.$$

Similarly, fine-tuning on task B yields

$$\Delta\mathbf{W}_B^{(l)} = \mathbf{W}_B^{(l)} - \mathbf{W}_0^{(l)}.$$

Prior work shows that task vectors support arithmetic operations (Ilharco et al., 2023). In particular, merging tasks A and B via addition,

$$\mathbf{W}_{AB} = \mathbf{W}_0 + \Delta\mathbf{W}_A + \Delta\mathbf{W}_B,$$

can endow the model with capabilities from both tasks. Intuitively, if $\Delta\mathbf{W}_A$ and $\Delta\mathbf{W}_B$ lie in largely orthogonal subspaces, their interference may degrade performance.

Formally, let $\mathcal{P}_A^X$ denote the performance on task X after fine-tuning on task A, and let $\mathcal{P}_{AB}^X$ denote the performance after merging $\Delta\mathbf{W}_A$ and $\Delta\mathbf{W}_B$. We say tasks A and B are **conflicting** if

$$\mathcal{P}_{AB}^A + \mathcal{P}_{AB}^B \ll \mathcal{P}_A^A + \mathcal{P}_B^B.$$

Accordingly, the performance loss ratio reported in Figure 1(b) is

$$1 - \frac{1}{2}\left(\frac{\mathcal{P}_{AB}^A}{\mathcal{P}_A^A} + \frac{\mathcal{P}_{AB}^B}{\mathcal{P}_B^B}\right).$$



Figure 3: Full results of angle distributions.

Angles Between Delta Weights. To quantify the geometric relationship between $\Delta\mathbf{W}_A$ and $\Delta\mathbf{W}_B$, we measure projection angles following (Zheng et al., 2025). Consider the singular value decompositions

$$\Delta\mathbf{W}_A = \mathbf{U}_{r_A}\mathbf{\Sigma}_{r_A}\mathbf{V}_{r_A}^\top, \qquad \Delta\mathbf{W}_B = \mathbf{U}_{r_B}\mathbf{\Sigma}_{r_B}\mathbf{V}_{r_B}^\top,$$

where r_A and r_B are the respective ranks. Assume $\Delta\mathbf{W}_A, \Delta\mathbf{W}_B \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$. Then $\mathbf{U}_{r_A} \in \mathbb{R}^{d_{\text{in}} \times r_A}$ and $\mathbf{U}_{r_B} \in \mathbb{R}^{d_{\text{in}} \times r_B}$ span the column spaces of the two delta weights.

Let $\mathbf{U}_{r_A} = [\mathbf{u}_1, \ldots, \mathbf{u}_{r_A}]$. For each basis vector $\mathbf{u}_i$, we compute its projection onto the subspace spanned by $\mathbf{U}_{r_B}$,

$$\hat{\mathbf{u}}_i = \mathbf{U}_{r_B}\mathbf{U}_{r_B}^\top \mathbf{u}_i,$$

and define the corresponding projection angle as

$$\theta_i = \arccos\left(\frac{\mathbf{u}_i^\top \hat{\mathbf{u}}_i}{\|\mathbf{u}_i\| \, \|\hat{\mathbf{u}}_i\|}\right).$$

For each layer l, this yields a set of angles $\{\theta_i^{(l)}\}_{i=1}^{r_A}$.

In Figure 1(a), we obtain layer-wise delta weights using LoRA (Hu et al., 2022) with rank 8 and report the angle distributions for the `q_proj` weights. Results for all layers are provided in Figure 3.

B Relations to LoRA-based Methods

Our proposed AEI differs fundamentally from LoRA-based expansion approaches (Wang et al., 2023b; Liang & Li, 2024), which encode task-specific knowledge as additive low-rank perturbations to a shared transformation, e.g., $(\mathbf{W}_0 + \sum_j \Delta\mathbf{W}_j)\boldsymbol{x}$. Although individual perturbations can be parameter-wise preserved, their functional contributions are generally not identifiable from the resulting additive transformation: multiple decompositions may realize the same effective mapping, and different updates may compensate for or cancel one another when composed. Therefore, retaining a historical update does not necessarily retain the function it originally contributed, giving rise to functional forgetting. AEI instead isolates task-induced functional increments into independent expert branches and uses explicit expert identification to control their activation. This makes previously acquired functions individually addressable and allows new knowledge to be incorporated without altering their functional roles.

Moreover, based on the discussions in Lemma 1, orthogonality-based methods such as OLoRA (Wang et al., 2023b) and InfLoRA (Liang & Li, 2024) enforce the form of ${\mathbf{W}_{\text{up}}^t}^\top \mathbf{W}_{\text{up}}^j = \mathbf{0}$ for $j < t$, which only suppresses the component of the new expert lying in the previously learned subspace. Such constraints favor novelty but limit the reuse of useful historical features and do not explicitly identify redundant neurons. Our regularization instead allows each neuron to adaptively choose among knowledge reuse, knowledge expansion, and removal.

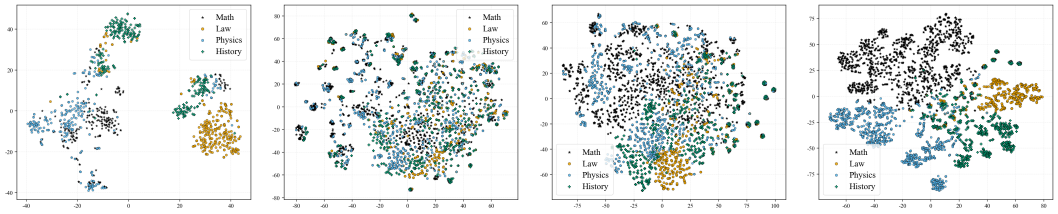
C Details of the Penalty

To obtain the orthonormal bases $\mathbf{U}_t$, we denote by $\mathbf{W}_t = [\mathbf{W}_{\text{up}}^0, \mathbf{W}_{\text{up}}^1, \cdots, \mathbf{W}_{\text{up}}^{t-1}]$ the concatenation of the up-projection matrices in the LLM. Then, we perform Singular Value Decomposition (SVD) on $\mathbf{W}_t$: $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \text{SVD}(\mathbf{W}_t)$. To approximate the space $\mathcal{W}_t$, we let $\mathbf{U}_t = \mathbf{U}[:, : r_{0.9}]$, where $r_{0.9}$ denotes the rank index that maintains 90% energy of the singular values in $\mathbf{\Sigma}$.

D Proofs of Lemma 1

Lemma (Orthogonality of the regularization gradients)**.** *For every i such that $\mathbf{w}_i^{\parallel} \neq \mathbf{0}$ and $\mathbf{w}_i^{\perp} \neq \mathbf{0}$, the gradients of $\mathcal{L}_{\text{reg}}$ with respect to the parallel and orthogonal components are orthogonal:*

$$\left\langle \nabla_{\mathbf{w}_i^{\parallel}} \mathcal{L}_{\text{reg}}, \nabla_{\mathbf{w}_i^{\perp}} \mathcal{L}_{\text{reg}} \right\rangle = 0.$$



(a) PATs (Block 1, $k = 1$) (b) Answer tokens (Block 1) (c) Answer tokens (Block 7) (d) Answer tokens (Block 10)

Figure 4: Token representations produced by different transformer blocks across various tasks (Qwen2.5-7B-Instruct). Each task (sampled from MMLU-Pro (Wang et al., 2024b)) is represented by a distinct color. We can see that, for answer tokens, the representations extracted by the higher block of LLMs are more domain-specific.

Proof. For a fixed index i, only the i-th term in L_{reg} depends on $\boldsymbol{w}_i^{\parallel}$ and $\boldsymbol{w}_i^{\perp}$. Moreover,

$$\frac{\partial \|\boldsymbol{w}_i^{\parallel}\|_2}{\partial \boldsymbol{w}_i^{\perp}} = \mathbf{0}, \qquad \frac{\partial \|\boldsymbol{w}_i^{\perp}\|_2}{\partial \boldsymbol{w}_i^{\parallel}} = \mathbf{0}.$$

Hence,

$$\nabla_{\boldsymbol{w}_i^{\parallel}} L_{\text{reg}} = \frac{\lambda}{d_{m'}} \frac{\boldsymbol{w}_i^{\parallel}}{\|\boldsymbol{w}_i^{\parallel}\|_2}, \qquad \nabla_{\boldsymbol{w}_i^{\perp}} L_{\text{reg}} = \frac{\lambda}{d_{m'}} \frac{\boldsymbol{w}_i^{\perp}}{\|\boldsymbol{w}_i^{\perp}\|_2}.$$

Therefore,

$$\left\langle \nabla_{\boldsymbol{w}_i^{\parallel}} L_{\text{reg}}, \nabla_{\boldsymbol{w}_i^{\perp}} L_{\text{reg}} \right\rangle = \frac{\lambda^2}{d_{m'}^2} \frac{\langle \boldsymbol{w}_i^{\parallel}, \boldsymbol{w}_i^{\perp} \rangle}{\|\boldsymbol{w}_i^{\parallel}\|_2 \|\boldsymbol{w}_i^{\perp}\|_2}.$$

It remains to show that $\langle \boldsymbol{w}_i^{\parallel}, \boldsymbol{w}_i^{\perp} \rangle = 0$. By definition,

$$\boldsymbol{w}_i^{\parallel} = \mathbf{U}_t \mathbf{U}_t^{\top} \boldsymbol{w}_i \in \text{span}(\mathbf{U}_t),$$

and

$$\boldsymbol{w}_i^{\perp} = (\mathbf{I} - \mathbf{U}_t \mathbf{U}_t^{\top}) \boldsymbol{w}_i.$$

Moreover,

$$\begin{aligned} \mathbf{U}_t^{\top} \boldsymbol{w}_i^{\perp} &= \mathbf{U}_t^{\top} (\mathbf{I} - \mathbf{U}_t \mathbf{U}_t^{\top}) \boldsymbol{w}_i \\ &= (\mathbf{U}_t^{\top} - \mathbf{U}_t^{\top} \mathbf{U}_t \mathbf{U}_t^{\top}) \boldsymbol{w}_i \\ &= (\mathbf{U}_t^{\top} - \mathbf{I}_m \mathbf{U}_t^{\top}) \boldsymbol{w}_i \\ &= \mathbf{0}. \end{aligned}$$

Thus $\boldsymbol{w}_i^{\perp} \in \text{span}(\mathbf{U}_t)^{\perp}$. Since $\boldsymbol{w}_i^{\parallel} \in \text{span}(\mathbf{U}_t)$, we obtain

$$\langle \boldsymbol{w}_i^{\parallel}, \boldsymbol{w}_i^{\perp} \rangle = 0.$$

Substituting this into the previous display yields

$$\left\langle \nabla_{\boldsymbol{w}_i^{\parallel}} L_{\text{reg}}, \nabla_{\boldsymbol{w}_i^{\perp}} L_{\text{reg}} \right\rangle = 0.$$

This completes the proof. □

E VISUALIZATION OF HIDDEN STATES IN LLMS.

Visualization Experiments. We visualize the LLM's intermediate features to assess their domain separability. Let $\mathbf{Q}_t^l = \{\boldsymbol{q}_{t_i}^l\}_{i=1}^{n_q}$ be a set of prompt token features produced by the lth block of $\mathcal{M}_0$, and $\mathbf{A}_t^l = \{\boldsymbol{a}_{t_i}^l\}_{i=1}^{n_a}$ be the corresponding answer token features, where t is the task index and $\boldsymbol{q}_{t_i}^l, \boldsymbol{a}_{t_j}^l \in \mathbb{R}^d$. Due to the causal attention nature of LLMs, the former prompt tokens can not see the information of the later tokens, thus encoding limited domain-specific information. To evaluate the

extent to which domain-identity information is encoded in the prompt tokens, we define the last k tokens of $\mathbf{Q}_t^l = \{\boldsymbol{q}_{t_i}^l\}_{i=1}^{n_q}$ as Prompt Aggregation Tokens (PATs) which aggregate the information from the entire prompt. Then, we visualize PATs and answer tokens $\mathbf{A}_t^l$ for all $t = 1, \ldots, T$ and $l = 1, \ldots, L$ using t-SNE (Maaten & Hinton, 2008).

Prompt Aggregation Tokens. For the base LLMs (without instruction-finetuning), we use a "<s>user\n{prompt}<s>assistant\n{answer}" chat template, in which the tokens that correspond to <s>assistant\n are PATs.

Observations. As shown in Figure 4(a), the features of PATs are nearly domain-separable, showing that the prompt tokens encode rich domain-identity information. In contrast, Figure 4(b) reveals that the answer tokens possess less distinguishable domain identities. This discrepancy can be attributed to two primary factors. First, the features extracted by the lower blocks of $\mathcal{M}_0$ inherently contain limited domain-specific information, as is visually apparent in Figures 4(c) and 4(d). Second, the answer tokens, which have a broader contextual view in the MHSA layer, may assimilate excessive general information, thereby diluting or overshadowing the salient domain-identity signals. These observations motivate two key design principles for the gating function: (1) operating on higher intermediate layers of LLMs, where the domain-identity information is more pronounced; and (2) allowing the later tokens to selectively aggregate information from prompt tokens that encode domain-specific instructions, thereby emphasizing domain-identity information. The later can be achieved via an attention-based gating function.

Table 6: Details of selected tasks of SuperNI-10.

Abbreviation	Task Name
QA	task024_cosmosqa_answer_generation
QG	task074_squad1.1_question_generation
SA	task1312_amazonreview_polarity_classification
SUM	task511_reddit_tifu_long_text_summarization
TRANS	task1219_ted_translation_en_es
DSG	task574_air_dialogue_sentence_generation
EXPL	task192_hotpotqa_sentence_generation
PARA	task177_para-nmt_paraphrasing
POS	task346_hybridqa_classification
PE	task064_all_elements_except_first_i

Table 7: Details of task orders

Settings	Task Chain
5 Tasks Order 1	TRANS → SA → QA → SUM → QG
5 Tasks Order 2	QA → QG → SA → SUM → TRANS
5 Tasks Order 3	SUM → QG → TRANS → QA → SA
10 Tasks Order 1	TRANS → SA → QA → SUM → QG → PE → PARA → POS → DSG → EXPL
10 Tasks Order 2	QA → QG → SA → SUM → TRANS → DSG → EXPL → PARA → PE → POS
10 Tasks Order 3	SUM → QG → TRANS → QA → SA → PARA → DSG → POS → EXPL → PE

F Further Details

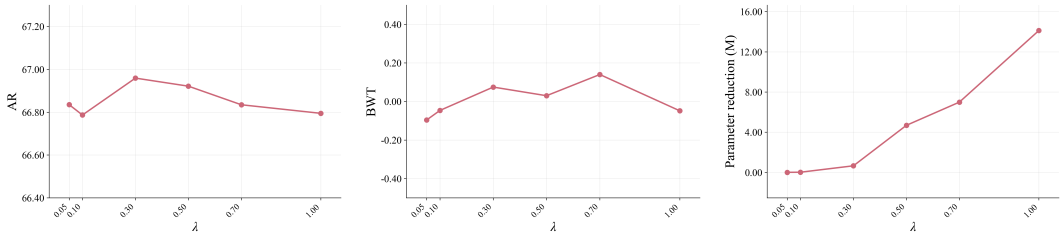
Details of SuperNI-10. Following previous continual learning studies (Huang et al., 2024; Wang et al., 2025b), we select a sequence of 10 tasks from the SuperNI benchmark. The selected tasks and their training order are detailed in Table 6 and Table 7, respectively.

Details of SuperNI-Long. There specific tasks and orders are summarized in Table 8.

Evaluation Metrics. For all tasks, we evaluate model performance using the ROUGE-L metric (Lin, 2004). We denote by $\mathcal{P}_j^i$ the performance on task j after completing training stage i, and by $\mathcal{P}_t^{\text{ind}}$ the performance on task t when the model is trained exclusively on that task from the same initialization.

Table 8: Task order of SuperNI-Long.

task591 (Question Answering)
→ task600 (Program Execution) → task568 (Question Generation)
→ task132 (Paraphrasing) → task593 (Explanation) → task155 (Pos Tagging)
→ task618 (Summarization) → task565 (Dialogue Generation) → task1329 (Translation)
→ task517 (Sentiment Analysis) → task1412 (Question Answering)
→ task079 (Program Execution)
→ task821 (Question Generation) → task177 (Paraphrasing) → task295 (Explanation)
→ task1543 (Pos Tagging) → task1309 (Summarization) → task1603 (Dialogue Generation)
→ task1330 (Translation) → task195 (Sentiment Analysis) → task1293 (Question Answering)
→ task376 (Program Execution) → task897 (Question Generation) → task223 (Explanation)
→ task345 (Pos Tagging) → task589 (Summarization) → task576 (Dialogue Generation)
→ task659 (Translation) → task923 (Sentiment Analysis) → task898 (Question Answering)
→ task113 (Program Execution) → task1602 (Question Generation) → task192 (Explanation)
→ task382 (Pos Tagging) → task668 (Summarization) → task1729 (Dialogue Generation)
→ task658 (Translation) → task512 (Sentiment Analysis) → task1726 (Question Answering)
→ task365 (Program Execution) → task1552 (Question Generation) → task1369 (Explanation)
→ task346 (Pos Tagging) → task511 (Summarization) → task360 (Dialogue Generation)
→ task650 (Translation) → task397 (Sentiment Analysis) → task615 (Question Answering)
→ task499 (Program Execution) → task581 (Question Generation)



(a) AR (b) BWT (c) Parameter reduction

(d) AR (e) BWT (f) Parameter reduction

Figure 5: Hyperparameter analysis of the regularization coefficient λ and ε on AR, BWT, and parameter reduction under the 20-task setting. The top row varies the regularization coefficient λ, while the bottom row varies ε.

We report four metrics: Average ROUGE-L (**AR**), which measures the final average performance over all T tasks; and Backward Transfer (**BWT**), which quantifies the impact of learning subsequent tasks on previously learned ones. These metrics are defined as follows:

$$\mathrm{AR} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{P}_t^T,$$

$$\mathrm{BWT} = \frac{1}{T-1}\sum_{t=1}^{T-1} \left(\mathcal{P}_t^T - \mathcal{P}_t^t\right).$$

Higher AR and BWT values indicate better performance.

Table 9: Ablation study on different gating functions.

Gating Type	SuperNI 5 Tasks		SuperNI 10 Tasks	
	AR ↑	BWT ↑	AR ↑	BWT ↑
Linear	48.87	-7.63	43.04	-21.94
AE	50.29	-3.58	58.43	-6.15
MLP	51.73	-1.60	62.88	-2.15
Ours	**53.46**	**+0.03**	**65.56**	**+0.04**

Table 10: Results with standard deviations on SuperNI-10. Standard deviations are shown in parentheses. * indicates a statistically significant difference from Gain at $p < 0.05$.

Method	Metric	Order 1	Order 2	Order 3	Avg.
SuperNI 5 Tasks					
Gain	AR	53.15 (0.19)	52.93 (0.18)	53.04 (0.18)	53.04 (0.21)
	BWT	-0.13 (0.06)	-0.41 (0.16)	-0.59 (0.12)	-0.38 (0.09)
AEI (Ours)	AR	**53.46 (0.24)**	**53.26 (0.12)**	**53.26 (0.11)**	**53.33 (0.11)**
	BWT	**+0.03 (0.05)**	**-0.02 (0.14)**	**-0.19 (0.03)**	**-0.06 (0.12)**
SuperNI 10 Tasks					
Gain	AR	63.14 (0.31)	63.69 (0.46)	63.81 (0.51)	63.55 (0.43)
	BWT	-1.07 (0.25)	-1.32 (0.24)	-1.87 (0.30)	-1.42 (0.18)
AEI (Ours)	AR	**65.56 (0.25)***	**65.32 (0.27)***	**65.64 (0.45)***	**65.51 (0.16)***
	BWT	**+0.04 (0.04)***	**+0.01 (0.05)***	**-0.10 (0.09)***	**-0.01 (0.08)***

Implementation Details. We set the intermediate dimension of each lightweight expert to $d_{m'} = 12$, keeping the number of trainable parameters per task comparable to that of the baselines. On both benchmarks, we train each task for 3 epochs and retain 1% of the data from each previous task for replay. We use the AdamW optimizer with an initial learning rate of 2×10^{-4} and a batch size of 16 for expert training. The pruning threshold ϵ is fixed as 10^{-1} for all experiments.

The gating function takes the outputs of the first block as input, i.e., $l_0 = 1$ in Eq. (12). For gating training, we randomly reserve 25% of the available training data for validation and use the Adam optimizer with an initial learning rate of 1×10^{-4} and a batch size of 2. We train the gating function for up to 5,000 steps with early stopping.

Each training stage follows a two-phase procedure: we first train the gating function by minimizing $\mathcal{L}_{\text{gate}}$, and then freeze it while training the experts by minimizing $\mathcal{L}_{\text{task}} + \mathcal{L}_{\text{reg}}$. All reported results are averaged over three random seeds.

G Additional Experiments

Significance Tests. For each experimental setting, we conduct three independent runs with different random seeds and report the mean and standard deviation of AR and BWT. We use a t-test to compare AEI with Gain, with statistically significant differences at $p < 0.05$ marked by an asterisk ($*$). As shown in Tables 10 and 11, the improvements of AEI on the 5-task SuperNI-10 setting are not statistically significant, although AEI consistently achieves better mean AR and BWT. In contrast, for the more challenging 10-task setting, AEI significantly outperforms Gain in both AR and BWT across all three task orders and their averages. This pattern becomes even more pronounced on SuperNI-Long: all comparisons across 20–50 tasks are statistically significant for both LLaMA-3.1-8B and Qwen3-4B. These results indicate that the advantage of AEI becomes increasingly robust as the continual learning sequence grows, providing statistically reliable improvements in both adaptation performance and knowledge retention.

Table 11: SuperNI-Long results with standard deviations using LLaMA-3.1-8B and Qwen3-4B. Standard deviations are shown in parentheses. * indicates a statistically significant difference from Gain at $p < 0.05$.

Method	Metric	20 Tasks	30 Tasks	40 Tasks	50 Tasks
LLaMA-3.1-8B					
Gain	AR	63.62 (0.51)	61.92 (0.48)	60.16 (0.38)	58.36 (0.67)
	BWT	-0.85 (0.09)	-0.82 (0.08)	-1.14 (0.12)	-1.74 (0.12)
AEI (Ours)	AR	**66.96 (0.28)***	**65.38 (0.38)***	**63.27 (0.45)***	**61.57 (0.41)***
	BWT	**-0.01 (0.09)***	**+0.03 (0.04)***	**+0.06 (0.10)***	**+0.05 (0.06)***
Qwen3-4B					
Gain	AR	63.59 (0.51)	60.26 (0.47)	59.23 (0.62)	56.82 (0.73)
	BWT	-1.12 (0.11)	-1.56 (0.13)	-1.78 (0.19)	-1.84 (0.18)
AEI (Ours)	AR	**66.19 (0.32)***	**63.60 (0.39)***	**62.33 (0.42)***	**60.81 (0.39)***
	BWT	**-0.15 (0.10)***	**+0.12 (0.11)***	**+0.02 (0.08)***	**+0.03 (0.12)***

Gating Ablation. Table 9 compares different gating functions. The performance improves consistently from Linear to AE and MLP, indicating that more expressive gating mechanisms are better able to identify task-relevant experts. Our context-aware gating achieves the best results in both 5-task and 10-task settings, with substantial gains in both AR and BWT. In particular, the improvement becomes more pronounced as the task sequence grows, suggesting that incorporating contextual information is especially important for accurate expert identification and mitigating cross-task interference in longer continual learning sequences.

Hyperparameter Analysis. Figure 5 investigates the effects of the regularization coefficient λ and the gating smoothing parameter ε. For λ, both AR and BWT remain highly stable over a wide range of values, with only minor fluctuations in performance. Moderate values of λ can improve both metrics, suggesting that an appropriate regularization strength helps the newly introduced experts focus on complementary feature directions without impairing knowledge acquisition. In contrast, the effect of λ on model size is much more pronounced: as λ increases, the number of reduced parameters grows substantially, indicating that stronger regularization effectively suppresses redundant expert capacity. A similar pattern is observed for ε. Varying ε causes only small changes in AR and BWT, while moderate settings can slightly improve performance by encouraging more effective reuse of previously learned experts. Meanwhile, larger ε leads to a clear increase in parameter reduction, showing that stronger knowledge reuse reduces the need to retain newly introduced expert parameters. Overall, AEI is insensitive to both hyperparameters in terms of predictive performance, while they provide effective control over model expansion. These observations support our central motivation: as the represented feature subspace grows across tasks, the remaining complementary space becomes increasingly constrained, allowing AEI to maintain compact expert expansion with reduced redundancy and cross-task interference.