REX-MOE: KNOWLEDGE REUSE IN EXPANDABLE MIXTURE-OF-EXPERTS FOR CONTINUAL LEARNING OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) requires models to learn new tasks sequentially while preserving previously acquired knowledge. Expandable mixture-of-experts (MoE) provides a natural framework by introducing new experts for incoming tasks while preserving historical experts to retain knowledge. However, existing methods typically reuse historical experts only as whole modules, which limits the flexibility to reuse and recombine useful components of acquired knowledge when learning new tasks. By reusing these components while keeping historical experts fixed, models could better balance new-task learning and knowledge retention. In this paper, we propose knowledge Reuse in expandable MoE (Rex-MoE), which enables flexible knowledge reuse during expert learning while coordinating the contributions of new and historical experts. For expert learning, Rex-MoE factorizes each adaptation into expandable shared bases and a task-specific transformation coupling historical and new bases. This structure allows new tasks to reuse historical bases without rewriting the transformations among them. For expert integration, Rex-MoE learns the relative contributions of new and historical experts while constraining router updates to limit interference. Extensive experiments on CL benchmarks demonstrate state-of-the-art performance of Rex-MoE.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language tasks (Achiam et al., 2023; Touvron et al., 2023; Yang et al., 2025a). However, real-world applications often require LLMs to continuously adapt to new tasks over time. Continual learning (CL) aims to enable such sequential adaptation without repeatedly retraining the model on all historical data (Li & Hoiem, 2017; Van de Ven et al., 2022; Ge et al., 2025). A fundamental challenge in CL is catastrophic forgetting (McCloskey & Cohen, 1989; Parisi et al., 2019; Wang et al., 2024a), whereby learning new tasks interferes with previously acquired knowledge, resulting in substantial performance degradation on earlier tasks.

To mitigate forgetting while maintaining parameter efficiency, expandable mixture-of-experts (MoE) with low-rank adaptation (LoRA) (Hu et al., 2022) has become a promising paradigm for CL. LoRA introduces trainable low-rank matrices while keeping the pre-trained model frozen, substantially reducing the cost of task adaptation. By treating LoRA modules as experts, expandable MoE methods (Zhao et al., 2024; Liang et al., 2025) introduce new experts for incoming tasks while retaining historical experts. Such expansion provides additional capacity for new knowledge while protecting previous adaptations from direct overwriting. Moreover, routing over multiple experts keeps historical adaptations accessible to subsequent tasks, allowing their outputs to contribute without modifying the corresponding experts.

However, preserving historical experts alone does not provide a flexible mechanism for reusing their knowledge in new tasks. At the expert level, independently parameterized experts limit how new experts can build on historical knowledge. For example, Amazon product reviews and IMDB movie reviews are both sentiment-analysis tasks but come from different domains. A new expert for IMDB may benefit from knowledge components learned from Amazon. However, when the Amazon expert is reused only as a whole, these components remain tied to its task-specific adaptation module,

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.

limiting their flexible reuse within the new expert. Reusing these components while keeping historical experts fixed could better balance new-task learning and knowledge retention. At the integration level, expanding the expert pool requires coordinating the relative contributions of new and historical experts while limiting interference with previous tasks. Together, flexible knowledge reuse and controlled expert integration could help balance new-task learning and knowledge retention in CL.

To this end, we propose knowledge Reuse in expandable MoE (Rex-MoE), which considers continual expert expansion from two complementary aspects: expert learning and expert integration. For expert learning, Rex-MoE factorizes each adaptation into expandable shared bases and a task-specific transformation that couples these bases. When a new task arrives, the shared basis set is expanded by adding new bases, and the new expert learns transformations that connect historical and newly introduced bases. This enables the new expert to reuse historical input and output bases, while historical bases and previous experts' transformations remain frozen. For expert integration, Rex-MoE coordinates the relative contributions of new and historical experts, so that transformations preserved in historical experts remain directly reusable through routing. Since the new expert is optimized for the current task, Rex-MoE encourages a higher routing preference for it while constraining router updates to limit cross-task interference. In this way, Rex-MoE enables new tasks to reuse accumulated knowledge while limiting interference with previous tasks.

In summary, our main contributions are as follows:

- We introduce an expandable expert factorization that separates shared bases from task-specific transformations, enabling new tasks to reuse historical knowledge without overwriting previous-task adaptations.
- We develop a relative routing mechanism that learns the relative contributions of new and historical experts while constraining router updates to limit interference during expansion.
- We conduct extensive experiments on CL benchmarks, demonstrating that Rex-MoE achieves state-of-the-art performance compared with existing CL methods.

2 Related Work

2.1 Low-Rank Adaptation for Large Language Models

Low-rank adaptation (LoRA) (Hu et al., 2022) represents task-specific weight updates with low-rank matrices, providing a parameter-efficient alternative to full fine-tuning (Lialin et al., 2023; Han et al., 2024). Recent LoRA variants have explored initialization strategies (Meng et al., 2024; Wang et al., 2024b; Yang et al., 2024), optimization strategies (Hayou et al., 2024; Wang et al., 2025), adaptive rank allocation (Zhang et al., 2023; Mao et al., 2024), and structured parameterizations of low-rank updates (Liu et al., 2024; Kopiczko et al., 2024; Bini et al., 2025). In particular, StelLA (Li et al., 2025) parameterizes the low-rank update with three factors, where two orthonormal matrices characterize the input and output subspaces and a compact core matrix captures the transformation between them. While these methods primarily target individual-task adaptation, continual learning further requires reusing accumulated knowledge for new tasks while preserving prior capabilities.

2.2 Continual Learning with Pre-trained Models

Continual learning (CL) focuses on enabling models to acquire knowledge from a sequence of tasks while retaining prior capabilities (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017; Luo et al., 2025). Traditional CL methods were primarily developed for models trained from scratch (Lopez-Paz & Ranzato, 2017; Zenke et al., 2017). With the emergence of large pre-trained models, recent research has increasingly shifted toward continually adapting pre-trained representations while leveraging their strong generalizability for downstream tasks (Zhou et al., 2024; Shi et al., 2025; Yang et al., 2025c; Qiao & Mahdavi, 2026). For large language models (LLMs), a common strategy is to freeze the pre-trained backbone and progressively introduce new lightweight components for each task, such as prompts (Zhu et al., 2022; Razdaibiedina et al., 2023) or low-rank modules (Zhao et al., 2024; Qian et al., 2025). This expandable formulation provides additional adaptation capacity while reducing direct modification of previously learned parameters.

Building on the expandable formulation, recent LoRA-based methods explore different ways to organize, constrain, and integrate task-specific adaptations across tasks. KIFLoRA (Feng et al., 2024a)

consolidates important low-rank knowledge across tasks, while O-LoRA (Wang et al., 2023) reduces interference by encouraging new LoRA branches to be orthogonal to previous ones. InfLoRA (Liang & Li, 2024) further constrains new adaptations using historical task-gradient subspaces. More recently, GainLoRA (Liang et al., 2025) learns input-dependent routers to integrate old and new LoRA branches. However, maintaining prior adaptations as separately parameterized branches limits the flexibility of reusing accumulated knowledge. Rex-MoE addresses this limitation by decomposing expert adaptations into reusable components while coordinating the contributions of experts.

3 PRELIMINARIES

3.1 PROBLEM DEFINITION

Following prior CL studies (Wang et al., 2023; Liang et al., 2025), we formulate the CL setting for LLMs as follows. Specifically, a pre-trained language model is sequentially trained on T tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$. The t-th task $\mathcal{T}_t$ provides a training dataset $\mathcal{D}_t = \{(x_{t,i}, y_{t,i})\}_{i=1}^{N_t}$ of N_t instances, where $x_{t,i}$ and $y_{t,i}$ denote the input and the corresponding output, respectively. When learning task $\mathcal{T}_t$, only $\mathcal{D}_t$ is accessible, while data from previous tasks cannot be revisited. Under the task-agnostic CL setting for LLMs, task identities are unavailable at inference time. Let $\boldsymbol{\theta}_t$ denote the model parameters after learning task $\mathcal{T}_t$. The ideal objective is to sequentially learn new tasks while maintaining performance on previously learned ones, which can be formulated as:

$$\min_{\boldsymbol{\theta}_t} \sum_{k=1}^{t} \mathbb{E}_{(x,y)\sim\mathcal{D}_k} \left[-\log p_{\boldsymbol{\theta}_t}(y \mid x) \right], \tag{1}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and $p_{\boldsymbol{\theta}_t}(y \mid x)$ denotes the conditional probability of generating the output sequence y given the input x under the model parameters $\boldsymbol{\theta}_t$.

3.2 LOW-RANK ADAPTATION

LoRA (Hu et al., 2022) adapts a pre-trained weight matrix $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}$ through a low-rank update:

$$\boldsymbol{W} = \boldsymbol{W}_0 + \Delta\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}, \tag{2}$$

where $\boldsymbol{B} \in \mathbb{R}^{m\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times n}$, and $r \ll \min(m, n)$ denotes the LoRA rank. During fine-tuning, the pre-trained weights remain frozen while the low-rank factors are optimized. A low-rank update can also be expressed using a three-factor decomposition (Li et al., 2025):

$$\Delta\boldsymbol{W} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top, \tag{3}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ define orthonormal output and input bases, respectively, and $\boldsymbol{S} \in \mathbb{R}^{r\times r}$ models their transformation. Specifically, $\boldsymbol{U}$ and $\boldsymbol{V}$ are optimized on Stiefel manifolds $\mathrm{St}(r, d) = \{\boldsymbol{Y} \in \mathbb{R}^{d\times r} : \boldsymbol{Y}^\top\boldsymbol{Y} = \boldsymbol{I}_r\}$. Optimization details are provided in Appendix A.

3.3 EXPANDABLE MIXTURE-OF-EXPERTS

A mixture-of-experts (MoE) model combines multiple experts through input-dependent routing weights. In an expandable MoE for CL, a new expert is introduced for each incoming task, while historical experts are retained. For an input sample x, let $\boldsymbol{h} \in \mathbb{R}^n$ denote the input vector to an adapted layer. After learning task $\mathcal{T}_t$, the layer output is given by:

$$\boldsymbol{f}^{(t)}(\boldsymbol{h}; x) = \boldsymbol{W}_0\boldsymbol{h} + \sum_{i=1}^{t} \alpha_i^{(t)}(x) E_i(\boldsymbol{h}), \tag{4}$$

where $\boldsymbol{W}_0$ denotes the frozen pre-trained weights, $E_i(\boldsymbol{h})$ denotes the output of the i-th expert, and $\alpha_i^{(t)}(x)$ is its routing weight for sample x at task stage t. Accordingly, continual adaptation depends on both how new experts are learned and how all experts are coordinated as the expert pool expands.

4 METHODOLOGY

In this section, we present Rex-MoE in detail. As illustrated in Figure 1, Rex-MoE consists of two components: knowledge-reusing expert (KRE) for expert learning and constrained relative routing (CRR) for expert integration. We introduce these two components in turn.

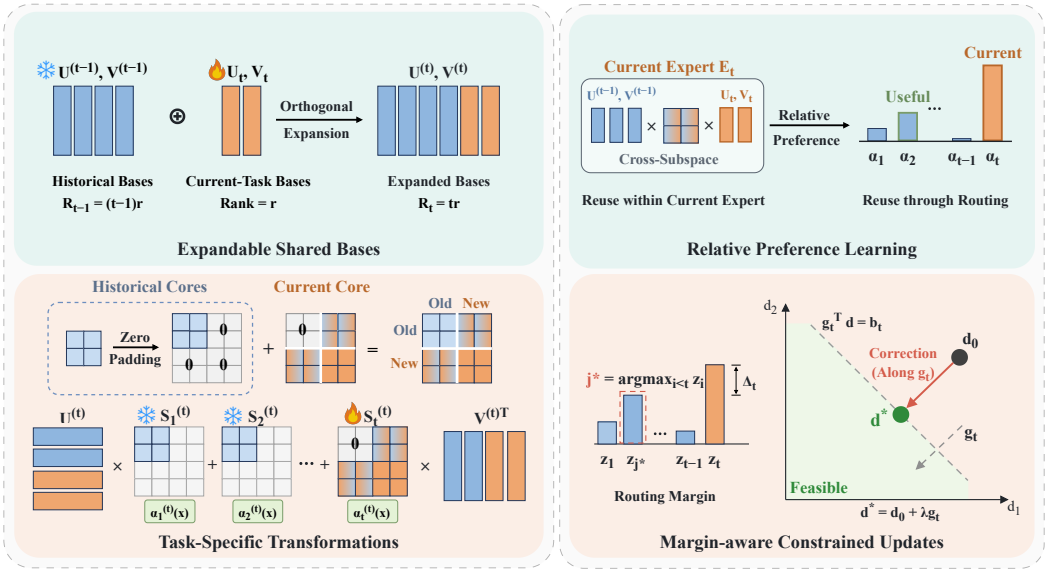


Figure 1: Overview of the Rex-MoE framework. **Left:** Historical expert parameters are preserved, while each new expert reuses accumulated bases through a task-specific core and expands them with new directions. **Right:** Relative routing further reuses historical experts in the mixture, with constrained updates regulating the contributions of new and historical experts to limit interference.

4.1 Knowledge-Reusing Expert

A common design in expandable LoRA-based CL is to maintain a separate low-rank expert for each task and freeze historical experts after training. Using the Stiefel-constrained LoRA parameterization in StelLA (Li et al., 2025), the expert associated with task $\mathcal{T}_i$ can be written as:

$$E_i(\boldsymbol{h}) = \boldsymbol{U}_i \boldsymbol{S}_i \boldsymbol{V}_i^\top \boldsymbol{h}, \tag{5}$$

where $\boldsymbol{U}_i \in \mathrm{St}(r, m)$ and $\boldsymbol{V}_i \in \mathrm{St}(r, n)$ define the output and input bases, respectively, and $\boldsymbol{S}_i \in \mathbb{R}^{r\times r}$ models the transformation between them. While freezing independently parameterized experts preserves their task-specific adaptations, the bases learned by previous tasks remain confined to their corresponding experts and are not directly incorporated into the construction of a new expert.

To enable new tasks to build on previously acquired knowledge, we introduce the knowledge-reusing expert (KRE), which separates the bases accumulated across tasks from the task-specific transformations defined over them. As new tasks arrive, KRE incrementally expands the shared bases while preserving previously learned expert transformations, allowing the current expert to reuse historical bases through task-specific cross-subspace transformations.

4.1.1 Expandable Shared Bases

For task $\mathcal{T}_t$, KRE maintains cumulative output and input bases by augmenting the historical bases with the basis components introduced for the current task:

$$\boldsymbol{U}^{(t)} = [\boldsymbol{U}^{(t-1)}, \boldsymbol{U}_t] \in \mathbb{R}^{m\times R_t}, \qquad \boldsymbol{V}^{(t)} = [\boldsymbol{V}^{(t-1)}, \boldsymbol{V}_t] \in \mathbb{R}^{n\times R_t}, \tag{6}$$

where $\boldsymbol{U}_t \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V}_t \in \mathbb{R}^{n\times r}$ denote the new bases introduced for the current task, and $R_t = tr \le \min(m, n)$ denotes the cumulative basis dimension after learning task $\mathcal{T}_t$.

When learning $\mathcal{T}_t$, the historical bases $\boldsymbol{U}^{(t-1)}$ and $\boldsymbol{V}^{(t-1)}$ are kept fixed, while the newly added basis directions are optimized subject to orthogonality with the historical bases and orthonormality within themselves:

$$\begin{aligned} \boldsymbol{U}^{(t-1)\top}\boldsymbol{U}_t = \boldsymbol{0}, \quad \boldsymbol{U}_t^\top \boldsymbol{U}_t = \boldsymbol{I}_r, \\ \boldsymbol{V}^{(t-1)\top}\boldsymbol{V}_t = \boldsymbol{0}, \quad \boldsymbol{V}_t^\top \boldsymbol{V}_t = \boldsymbol{I}_r. \end{aligned} \tag{7}$$

Together with the orthonormality of the historical bases, the constraints in Eq. (7) ensure that the expanded bases remain orthonormal:

$$\boldsymbol{U}^{(t)\top}\boldsymbol{U}^{(t)} = \boldsymbol{I}_{R_t}, \qquad \boldsymbol{V}^{(t)\top}\boldsymbol{V}^{(t)} = \boldsymbol{I}_{R_t}. \tag{8}$$

To maintain these constraints throughout training, after each optimization step the new basis directions are projected onto the orthogonal complements of the historical bases and re-orthonormalized. The complete optimization procedure is provided in Appendix A.

The cumulative bases therefore provide a shared coordinate system that incorporates both historical and newly introduced adaptation components. In contrast to independently parameterized experts, the historical input and output bases are explicitly retained within this shared parameterization and are directly available to subsequent experts.

4.1.2 Task-Specific Transformations

Given the cumulative bases, each expert is represented by a task-specific core matrix that parameterizes the transformation between the available input and output basis components:

$$E_i(\boldsymbol{h}) = \boldsymbol{U}^{(t)}\boldsymbol{S}_i^{(t)}\boldsymbol{V}^{(t)\top}\boldsymbol{h}, \qquad \boldsymbol{S}_i^{(t)} \in \mathbb{R}^{R_t \times R_t}. \tag{9}$$

The shared bases capture adaptation components accumulated across tasks, while $\boldsymbol{S}_i^{(t)}$ specifies the task-specific transformation over these bases.

For the current task, previously learned transformations are preserved in historical experts and can be directly reused through routing. We therefore focus the current expert on reusing historical bases through cross-subspace transformations, while learning task-specific transformations within the newly introduced subspaces. Accordingly, we structure the current core as:

$$\boldsymbol{S}_t^{(t)} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{S}_t^{\text{cross},1} \\ \boldsymbol{S}_t^{\text{cross},2} & \boldsymbol{S}_t^{\text{new}} \end{bmatrix}, \tag{10}$$

where $\boldsymbol{S}_t^{\text{cross},1} \in \mathbb{R}^{R_{t-1}\times r}$, $\boldsymbol{S}_t^{\text{cross},2} \in \mathbb{R}^{r\times R_{t-1}}$, and $\boldsymbol{S}_t^{\text{new}} \in \mathbb{R}^{r\times r}$. The historical–historical block is therefore fixed to zero, while the two cross-subspace blocks and the new–new block are optimized for the current task. Substituting this block structure into Eq. (9) gives:

$$E_t(\boldsymbol{h}) = \boldsymbol{U}^{(t-1)}\boldsymbol{S}_t^{\text{cross},1}\boldsymbol{V}_t^{\top}\boldsymbol{h} + \boldsymbol{U}_t\boldsymbol{S}_t^{\text{cross},2}\boldsymbol{V}^{(t-1)\top}\boldsymbol{h} + \boldsymbol{U}_t\boldsymbol{S}_t^{\text{new}}\boldsymbol{V}_t^{\top}\boldsymbol{h}. \tag{11}$$

The three blocks play distinct roles in constructing the current expert. $\boldsymbol{S}_t^{\text{cross},1}$ and $\boldsymbol{S}_t^{\text{cross},2}$ define cross-subspace transformations between historical and newly introduced bases, enabling the current expert to reuse historical output and input bases, respectively, while $\boldsymbol{S}_t^{\text{new}}$ models transformations between the newly introduced input and output bases, providing additional task-specific capacity.

For each historical expert $i < t$, its previously learned transformation should remain unchanged as the shared bases expand. We therefore extend its core by zero padding:

$$\boldsymbol{S}_i^{(t)} = \begin{bmatrix} \boldsymbol{S}_i^{(t-1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{12}$$

Since both the historical bases and the previously learned core remain fixed, the corresponding expert transformation is exactly preserved:

$$\boldsymbol{U}^{(t)}\boldsymbol{S}_i^{(t)}\boldsymbol{V}^{(t)\top} = \boldsymbol{U}^{(t-1)}\boldsymbol{S}_i^{(t-1)}\boldsymbol{V}^{(t-1)\top}, \qquad i < t. \tag{13}$$

During task $\mathcal{T}_t$, KRE optimizes the newly introduced bases $\boldsymbol{U}_t$ and $\boldsymbol{V}_t$ together with $\boldsymbol{S}_t^{\text{cross},1}$, $\boldsymbol{S}_t^{\text{cross},2}$, and $\boldsymbol{S}_t^{\text{new}}$, while all historical bases and historical expert cores remain frozen.

4.2 Constrained Relative Routing

KRE enables task-specific adaptation by reusing historical bases within the current expert, while historical experts retain previously learned transformations that can be directly reused through routing. Since the current expert is optimized for the current task, we encourage a higher routing preference for it while allowing historical experts to provide complementary contributions. We therefore introduce constrained relative routing (CRR), which coordinates these contributions and limits cross-task interference through relative preference learning and margin-aware constrained updates.

4.2.1 RELATIVE PREFERENCE LEARNING

Given an input sample x, the routing score of expert i is computed as:

$$z_i(x) = \cos\left(\boldsymbol{q}_i(x), \boldsymbol{k}_i\right), \tag{14}$$

where $\boldsymbol{q}_i(x) \in \mathbb{R}^{d_{\text{router}}}$ is the routing query generated by a lightweight two-layer MLP and $\boldsymbol{k}_i \in \mathbb{R}^{d_{\text{router}}}$ is the corresponding key. We normalize the scores over all available experts to obtain their relative routing weights:

$$\alpha_i^{(t)}(x) = \frac{\exp\left(z_i(x)/\tau\right)}{\sum_{j=1}^{t} \exp\left(z_j(x)/\tau\right)}, \tag{15}$$

where τ is the temperature. This yields a normalized relative preference over the current and historical experts within the expanding expert pool. For samples from the current task, we optimize the following relative preference objective:

$$\mathcal{L}_{\text{pref}} = -\log \frac{\exp\left(z_t(x)/\tau\right)}{\sum_{i=1}^{t} \exp\left(z_i(x)/\tau\right)} = -\log \alpha_t^{(t)}(x). \tag{16}$$

Let $\mathcal{L}_{\text{task}}$ denote the negative log-likelihood loss on the current task data. The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{pref}} \mathcal{L}_{\text{pref}}, \tag{17}$$

where λ_{pref} controls the strength of the preference objective. This encourages the current expert to receive a higher routing preference without excluding useful contributions from historical experts.

4.2.2 MARGIN-AWARE CONSTRAINED UPDATES

Relative preference learning favors the current expert, but subsequent router updates may reduce its score separation from historical experts and increase routing interference. CRR therefore constrains how much each update can reduce this separation. Let $\boldsymbol{\theta}_r$ denote the trainable parameters of the current router and $\boldsymbol{d}_0$ the candidate update before applying the margin constraint:

$$\widetilde{\boldsymbol{\theta}}_r = \boldsymbol{\theta}_r + \boldsymbol{d}_0. \tag{18}$$

For task $\mathcal{T}_t$ with $t > 1$, we define the routing margin as:

$$\Delta_t(x) = z_t(x) - \max_{i<t} z_i(x). \tag{19}$$

Given a target margin $\gamma > 0$, we define the allowable first-order margin change as:

$$b_t(x) = \min\left(0, \gamma - \Delta_t(x)\right). \tag{20}$$

When $\Delta_t(x) < \gamma$, the update is constrained from further reducing the margin; once $\Delta_t(x) \geq \gamma$, it allows part of the excess margin to be reduced. Thus, the preference objective promotes a higher relative preference for the current expert, while the margin constraint limits reductions in the routing margin.

In practice, we aggregate the per-example bounds $b_t(x)$ over the samples contributing to an optimizer step to obtain the step-level bound b_t. We similarly obtain the step-level margin gradient $\boldsymbol{g}_t$ from the corresponding per-example gradients $\nabla_{\boldsymbol{\theta}_r} \Delta_t(x)$ within the update subspace. Under a first-order approximation, $\boldsymbol{g}_t^\top \boldsymbol{d}$ represents the margin change induced by an update $\boldsymbol{d}$. Starting from $\boldsymbol{d}_0$, CRR applies the minimum correction required to satisfy the margin condition:

$$\boldsymbol{d}^* = \boldsymbol{d}_0 + \frac{\left[b_t - \boldsymbol{g}_t^\top \boldsymbol{d}_0\right]_+}{\|\boldsymbol{g}_t\|_2^2} \boldsymbol{g}_t, \tag{21}$$

where $[a]_+ = \max(0, a)$; when $\boldsymbol{g}_t = \boldsymbol{0}$, the correction term is defined as zero. If $\boldsymbol{g}_t^\top \boldsymbol{d}_0 \geq b_t$, the candidate update remains unchanged; otherwise, CRR applies the minimum correction along $\boldsymbol{g}_t$ required to satisfy the constraint. The final router parameters are then updated as:

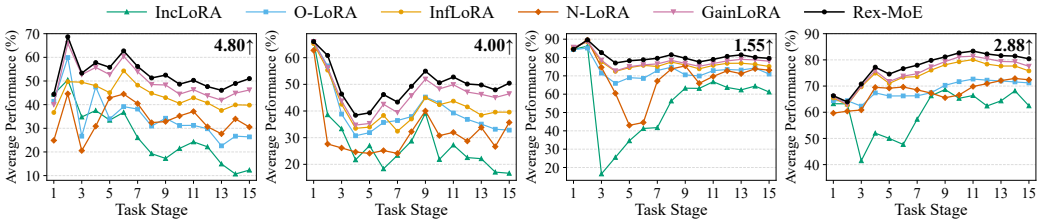
$$\boldsymbol{\theta}_r^* = \boldsymbol{\theta}_r + \boldsymbol{d}^*. \tag{22}$$

Further details on the constrained update are provided in Appendix B.

Summary of Rex-MoE. For each new task, Rex-MoE expands the shared bases and introduces a new expert core and router while keeping historical parameters fixed. The new parameters are jointly optimized using Eq. (17), with the margin-aware router correction applied during optimization. After training, these parameters are frozen for subsequent tasks. At inference, the model combines the outputs of all available experts using input-dependent routing weights.

Table 1: Results on different task sequences with T5-Large. The best results are highlighted in **bold**.

Method	Order 1		Order 2		Order 3		Order 4	
	AP↑	FT↓	AP↑	FT↓	AP↑	FT↓	AP↑	FT↓
SeqLoRA	7.30	47.60	7.03	47.97	49.46	27.60	33.81	45.53
LFPT5	39.03	10.87	29.70	20.72	66.62	14.57	67.40	13.20
EWC	15.32	26.78	18.19	30.28	43.24	23.66	46.25	32.90
TaSL	27.51	18.53	28.05	17.39	71.37	6.20	73.11	6.52
KIFLoRA	28.33	16.44	30.31	16.27	72.19	3.10	73.72	4.75
IncLoRA	12.33	41.93	16.65	36.56	61.19	13.63	62.46	15.92
O-LoRA	26.37	19.15	32.83	11.99	70.98	3.69	71.21	4.03
InfLoRA	39.78	7.64	39.57	8.93	75.15	4.19	75.79	3.47
N-LoRA	30.53	17.22	35.71	11.12	73.10	8.29	72.39	6.64
GainLoRA (InfLoRA)	46.21	2.40	46.44	2.61	78.01	0.77	77.54	1.25
Rex-MoE	**51.01**	**0.34**	**50.44**	**0.90**	**79.56**	**0.48**	**80.42**	**0.84**



(a) Order 1 (b) Order 2 (c) Order 3 (d) Order 4

Figure 2: Performance curves of different methods on four orders. We annotate Rex-MoE's final-stage AP improvement over the second-best method in the top-right corner of each subfigure.

5 Experiments

5.1 Experimental Settings

Datasets. Following previous works (Wang et al., 2023; Zhao et al., 2024; Liang et al., 2025), we evaluate Rex-MoE on two CL benchmarks for language models: SuperNI (Wang et al., 2022) and Long Sequence (Razdaibiedina et al., 2023). SuperNI covers diverse NLP task types, including dialogue generation, information extraction, question answering, summarization, and sentiment analysis. Following existing CL protocols (Zhao et al., 2024; Liang et al., 2025), we select three tasks from each type, resulting in 15 tasks, and organize them into two different task orders, denoted as Order 1 and Order 2. Long Sequence consists of 15 diverse classification tasks collected from multiple NLP benchmarks and is similarly organized into two task orders, denoted as Order 3 and Order 4. The detailed task sequences are provided in Appendix C.1.

Evaluation metrics. We use accuracy for classification tasks and ROUGE-L (Lin, 2004) for other types of tasks. Following previous CL studies (Chaudhry et al., 2020; Zhao et al., 2024; Liang et al., 2025), we employ average performance (AP) and forgetting (FT) for evaluation. Let $A_{j,i}$ denote the performance on task $\mathcal{T}_i$ after learning task $\mathcal{T}_j$. AP measures the final average performance across all tasks after the entire task sequence and is defined as $\mathrm{AP} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}$. FT quantifies the performance degradation on previously learned tasks and is defined as $\mathrm{FT} = \frac{1}{T-1}\sum_{i=1}^{T-1}\left(\max_{j\in\{i,i+1,\ldots,T-1\}} A_{j,i} - A_{T,i}\right)$, reflecting the model's ability to retain acquired knowledge throughout the CL process.

Baselines and implementation details. We conduct experiments on NVIDIA A6000 GPUs with three model backbones of different architectures and scales, including T5-Large (Raffel et al., 2020), Llama-2-7B (Touvron et al., 2023), and Qwen3-1.7B (Yang et al., 2025a). We compare Rex-MoE with several LLM CL methods, including SeqLoRA and IncLoRA (Wang et al., 2023) as LoRA baselines; LFPT5 (Qin & Joty, 2022) as a replay-based method; EWC (Kirkpatrick et al., 2017), TaSL (Feng et al., 2024b), and KIFLoRA (Feng et al., 2024a) as importance-aware methods; and

Table 2: Results on different task sequences with Llama-2-7B and Qwen3-1.7B. The best results are highlighted in **bold**.

Model	Method	Order 1		Order 2	
		AP↑	FT↓	AP↑	FT↓
Llama-2-7B	O-LoRA	39.37	15.84	37.55	20.23
	InfLoRA	42.93	11.23	39.94	15.00
	N-LoRA	39.74	8.24	41.79	7.87
	GainLoRA (InfLoRA)	51.27	2.84	50.17	4.71
	Rex-MoE	**55.41**	**0.65**	**54.71**	**1.27**
Qwen3-1.7B	O-LoRA	39.53	14.00	40.71	13.23
	InfLoRA	42.78	4.23	47.98	3.05
	N-LoRA	38.34	11.58	41.30	7.65
	GainLoRA (InfLoRA)	47.53	2.82	50.52	2.08
	Rex-MoE	**54.98**	**1.31**	**54.50**	**1.37**

Table 3: Ablation study of Rex-MoE on T5-Large. The best results are highlighted in **bold**.

Variant	Components			SuperNI (Order 1)	
	KRE	Preference	Margin	AP↑	FT↓
w/o KRE	×	✓	✓	46.55	3.75
w/o Preference	✓	×	✓	47.94	2.94
w/o Margin	✓	✓	×	48.84	0.88
Rex-MoE	✓	✓	✓	**51.01**	**0.34**

O-LoRA (Wang et al., 2023), InfLoRA (Liang & Li, 2024), N-LoRA (Yang et al., 2025b), and GainLoRA (Liang et al., 2025) as isolation-based expandable LoRA methods. For GainLoRA, we use its InfLoRA-based variant, denoted as GainLoRA (InfLoRA). For baseline methods, we follow official implementations and recommended hyperparameters whenever available. The main benchmark and ablation experiments are repeated three times with different random seeds, and the average results are reported. More implementation details are provided in Appendix C.

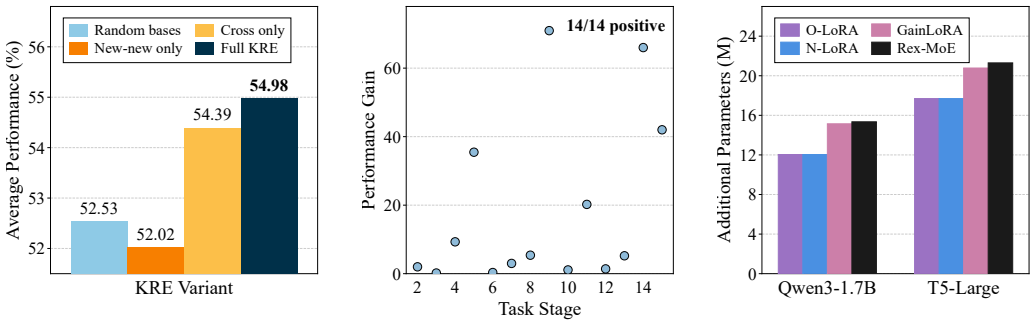
5.2 Benchmark Comparison

Table 1 and Figure 2 compare performance on T5-Large. Rex-MoE achieves the best AP across all four task sequences. On SuperNI, Rex-MoE obtains AP scores of 51.01 and 50.44 on Orders 1 and 2, outperforming GainLoRA (InfLoRA) by 4.80 and 4.00 points, respectively. Meanwhile, Rex-MoE substantially reduces forgetting, decreasing FT from 2.40 to 0.34 on Order 1 and from 2.61 to 0.90 on Order 2. Similar improvements are observed on the classification-oriented Long Sequence benchmark, where Rex-MoE improves AP by 1.55 and 2.88 points on Orders 3 and 4, respectively, while also achieving lower forgetting. These results demonstrate that Rex-MoE improves knowledge acquisition while effectively preserving previously learned knowledge across diverse CL scenarios.

Table 2 further evaluates Rex-MoE on decoder-only LLMs. On Llama-2-7B, Rex-MoE outperforms GainLoRA (InfLoRA) by 4.14 and 4.54 AP points on Orders 1 and 2, respectively, while substantially reducing forgetting. On the smaller Qwen3-1.7B decoder-only backbone, Rex-MoE also maintains strong performance, suggesting that its effectiveness is not tied to a particular backbone.

5.3 Ablation Studies

Table 3 evaluates three variants: w/o KRE replaces KRE with vanilla LoRA, w/o Preference removes the relative preference objective, and w/o Margin removes the margin-aware constrained update. For expert learning, removing KRE leads to the largest performance degradation, reducing AP from 51.01 to 46.55 and increasing FT from 0.34 to 3.75. This highlights the importance of KRE for effective expert learning. For expert integration, removing the preference mechanism decreases AP by 3.07 points and increases FT to 2.94, supporting the importance of learning the relative contributions of current and historical experts. Removing the margin constraint reduces AP to 48.84 and



(a) KRE analysis. (b) CRR analysis. (c) Additional parameters.

Figure 3: Further analysis of Rex-MoE. (a) Effects of historical-basis reuse and core blocks in KRE on Qwen3-1.7B under SuperNI Order 1. (b) Task-wise gains from historical-expert contributions through CRR. (c) Additional parameters beyond the frozen backbone over 15 tasks.

increases FT to 0.88, supporting its empirical benefit for both overall performance and knowledge retention. Together, these results show that both preference learning and constrained router updates contribute to effective expert integration.

5.4 Further Analysis

Further analysis of KRE. Figure 3a compares retrained KRE variants on Qwen3-1.7B under SuperNI Order 1. Replacing the reused historical bases with random orthogonal bases under the same trainable parameter budget reduces AP from 54.98 to 52.53, suggesting that bases learned from previous tasks contain reusable knowledge. We further examine the contributions of the core blocks. Retaining only the two cross-subspace blocks achieves 54.39 AP, close to the full KRE at 54.98, whereas retaining only the new–new block yields 52.02 AP. These results suggest the importance of reusing historical bases through cross-subspace transformations. Adding the new–new block on top of the cross-subspace blocks further improves AP by 0.59 points, providing additional task-specific capacity that complements the reuse of historical bases.

Further analysis of CRR. Figure 3b examines knowledge reuse through routing on Qwen3-1.7B across stages 2–15 of SuperNI Order 1. Using the same trained checkpoints, we remove historical-expert outputs at inference while preserving the learned routing weight of the current expert. This reduces current-task performance on all 14 tasks, with an average drop of 18.75 points. Averaged across tasks, the current expert receives the highest individual routing weight on 88.0% of test samples. These results indicate that CRR generally prioritizes the current expert while still leveraging complementary contributions from historical experts.

Additional parameter analysis. Figure 3c compares the additional parameters introduced beyond the frozen backbone over 15 tasks. On both Qwen3-1.7B and T5-Large, Rex-MoE has a parameter count close to GainLoRA while consistently achieving better performance. We further provide results in Appendix D.3, where Rex-MoE continues to outperform GainLoRA even with fewer additional parameters. Together, these results indicate that the improvements of Rex-MoE are not simply due to a larger parameter budget and support the parameter efficiency of the proposed design.

6 Conclusion

In this paper, we presented Rex-MoE, an expandable mixture-of-experts framework for continual learning of large language models that reuses accumulated knowledge while preserving previously acquired capabilities. KRE enables new experts to reuse historical bases through cross-subspace transformations while leaving historical experts unchanged. CRR coordinates the relative contributions of new and historical experts, allowing preserved transformations to be reused through routing while constraining router updates to limit interference. Experiments across two CL benchmarks and three language model backbones demonstrate consistent gains in average performance and reduced forgetting. These results highlight the value of combining knowledge reuse within new experts with controlled integration of historical experts for continual learning.

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 sequences, evaluation metrics, model backbones, baselines, experimental protocol, and implementation details in Section 5.1 and Appendix C. The formulation of Rex-MoE is presented in Sections 4.1 and 4.2, with the optimization of the Stiefel-constrained basis factors and the constrained router updates detailed in Appendices A and B. All datasets used in our experiments are publicly available.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Massimo Bini, Leander Girrbach, and Zeynep Akata. Decoupling angles and strength in low-rank adaptation. In *International Conference on Learning Representations*, 2025.

Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. In *Advances in Neural Information Processing Systems 33*, pp. 9900–9911, 2020.

Yujie Feng, Xu Chu, Yongxin Xu, Zexin Lu, Bo Liu, Philip S. Yu, and Xiao-Ming Wu. KIF: Knowledge identification and fusion for language model continual learning. *arXiv preprint arXiv:2408.05200*, 2024a.

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. TaSL: Continual dialog state tracking via task skill localization and consolidation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1266–1279, 2024b.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic mixture of curriculum LoRA experts for continual multimodal instruction tuning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 19011–19033, 2025.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 17783–17806, 2024.

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.

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.

Dawid Kopiczko, Tijmen Blankevoort, and Yuki Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

Zhizhong Li, Sina Sajadmanesh, Jingtao Li, and Lingjuan Lyu. StelLA: Subspace learning in low-rank adaptation using Stiefel manifold. In *Advances in Neural Information Processing Systems 38*, pp. 75066–75092, 2025.

Vladislav Lialin, Vijeta Deshpande, Xiaowei Yao, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 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, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. In *Advances in Neural Information Processing Systems 38*, pp. 68659–68689, 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32100–32121, 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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. DoRA: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 11662–11675, 2024.

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

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 37*, pp. 121038–121072, 2024.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Yu-Yang Qian, Yuan-Ze Xu, Zhen-Yu Zhang, Peng Zhao, and Zhi-Hua Zhou. TreeLoRA: Efficient continual learning via layer-wise LoRAs guided by a hierarchical gradient-similarity tree. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 50066–50085, 2025.

Fuli Qiao and Mehrdad Mahdavi. Merge before forget: A single LoRA continual learning via continual merging. In *International Conference on Learning Representations*, 2026.

Chengwei Qin and Shafiq Joty. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of T5. In *International Conference on Learning Representations*, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *International Conference on Learning Representations*, 2023.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

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.

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

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, 2024a.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems 37*, pp. 54905–54931, 2024b.

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 Conference on Empirical Methods in Natural Language Processing*, pp. 10658–10671, 2023.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025a.

Shuo Yang, Kun-Peng Ning, Yu-Yang Liu, Jia-Yu Yao, Yong-Hong Tian, Yi-Bing Song, and Li Yuan. Is parameter collision hindering continual learning in LLMs? In *Proceedings of the International Conference on Computational Linguistics*, pp. 4243–4259, 2025b.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen L Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In *Advances in Neural Information Processing Systems 37*, pp. 71768–71791, 2024.

Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. Recent advances of foundation language models-based continual learning: A survey. *ACM Computing Surveys*, 57(5):1–38, 2025c.

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.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

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 Annual Meeting of the Association for Computational Linguistics*, pp. 11641–11661, 2024.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 8363–8371, 2024.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. Continual prompt tuning for dialog state tracking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1124–1137, 2022.

A Details of Stiefel Optimization

Following Li et al. (2025), we employ geometric optimization for the Stiefel-constrained parameters in our method. For clarity, we first describe the optimization of a generic matrix $\boldsymbol{Y} \in \mathrm{St}(R_t, d)$ and then specify the additional cross-task orthogonality operation used by KRE.

Riemannian Gradient. Let $\mathcal{L}$ denote the training objective and $\nabla_{\boldsymbol{Y}}\mathcal{L} \in \mathbb{R}^{d\times R_t}$ be the Euclidean gradient obtained by standard back-propagation. Under the canonical Riemannian metric on the Stiefel manifold, the corresponding Riemannian gradient is given by:

$$\mathrm{grad}_{\boldsymbol{Y}}\,\mathcal{L} = \nabla_{\boldsymbol{Y}}\mathcal{L} - \boldsymbol{Y}\left(\nabla_{\boldsymbol{Y}}\mathcal{L}\right)^{\top}\boldsymbol{Y}. \tag{23}$$

The resulting gradient lies in the tangent space at $\boldsymbol{Y}$ and provides a valid update direction for the Stiefel-constrained parameter.

Tangent-Space Projection. We then pass the Riemannian gradient to a standard Euclidean optimizer, such as AdamW (Loshchilov & Hutter, 2019). Let $\widetilde{\boldsymbol{Y}}$ denote the tentative parameter obtained after one optimizer step. The optimizer-induced displacement is defined as:

$$\boldsymbol{\Delta} = \widetilde{\boldsymbol{Y}} - \boldsymbol{Y}. \tag{24}$$

Although the optimizer receives a Riemannian gradient, its internal operations, such as momentum accumulation and coordinate-wise adaptive scaling, may perturb the resulting displacement away from the tangent space. We therefore project the displacement back onto $\mathrm{T}_{\boldsymbol{Y}}\mathrm{St}(R_t, d)$. The Euclidean orthogonal projection onto this tangent space is given by:

$$\Pi_{\boldsymbol{Y}}(\boldsymbol{\Delta}) = \boldsymbol{\Delta} - \boldsymbol{Y}\,\mathrm{sym}\left(\boldsymbol{Y}^{\top}\boldsymbol{\Delta}\right), \tag{25}$$

where the symmetrization operator is defined as:

$$\mathrm{sym}(\boldsymbol{A}) = \frac{1}{2}\left(\boldsymbol{A} + \boldsymbol{A}^{\top}\right). \tag{26}$$

By construction, $\Pi_{\boldsymbol{Y}}(\boldsymbol{\Delta}) \in \mathrm{T}_{\boldsymbol{Y}}\mathrm{St}(R_t, d)$.

Retraction. Adding a tangent-space update to the current parameter does not in general yield a point on the Stiefel manifold. We therefore use a retraction to map the tentative update back onto the manifold. We denote the projected displacement by:

$$\widehat{\boldsymbol{\Delta}} = \Pi_{\boldsymbol{Y}}(\boldsymbol{\Delta}). \tag{27}$$

We employ the following polar retraction:

$$\mathcal{R}_{\boldsymbol{Y}}\left(\widehat{\boldsymbol{\Delta}}\right) = \mathrm{uf}\left(\boldsymbol{Y} + \widehat{\boldsymbol{\Delta}}\right), \tag{28}$$

where $\mathrm{uf}(\cdot)$ denotes the orthonormal factor of the polar decomposition. Specifically, the compact singular value decomposition of $\boldsymbol{Y} + \widehat{\boldsymbol{\Delta}}$ is given by:

$$\boldsymbol{Y} + \widehat{\boldsymbol{\Delta}} = \boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{Q}^{\top}. \tag{29}$$

The retracted parameter is then given by:

$$\boldsymbol{Y}^{\mathrm{ret}} = \boldsymbol{P}\boldsymbol{Q}^{\top}. \tag{30}$$

This guarantees $(\boldsymbol{Y}^{\mathrm{ret}})^{\top}\boldsymbol{Y}^{\mathrm{ret}} = \boldsymbol{I}_{R_t}$, thereby preserving the Stiefel orthonormality constraint.

Historical-Complement Re-orthogonalization. The Stiefel retraction in Eq. (28) preserves the orthonormality of the complete basis, but may alter the historical directions and therefore does not by itself preserve orthogonality with respect to their frozen values. KRE therefore applies an additional cross-task re-orthogonalization after the Stiefel retraction. Let $\boldsymbol{Y}^{(t-1)} \in \mathbb{R}^{d\times R_{t-1}}$ denote the frozen basis accumulated from previous tasks and $\boldsymbol{Y}_t \in \mathbb{R}^{d\times r}$ denote the newly introduced directions for the current task. For tasks after the first one, the complete basis can be written as $\boldsymbol{Y} = [\boldsymbol{Y}^{(t-1)}, \boldsymbol{Y}_t]$. The resulting orthogonality constraints are given by:

$$\boldsymbol{Y}^{(t-1)\top}\boldsymbol{Y}_t = \boldsymbol{0}, \qquad \boldsymbol{Y}_t^{\top}\boldsymbol{Y}_t = \boldsymbol{I}_r. \tag{31}$$

After the Stiefel retraction, we first restore the historical basis to its frozen value. Let $\widetilde{\boldsymbol{Y}}_t$ denote the provisional current-task directions extracted from $\boldsymbol{Y}^{\mathrm{ret}}$. We remove their components within the historical subspace using the orthogonal-complement projector:

$$\boldsymbol{P}_{\mathrm{hist}}^{\perp} = \boldsymbol{I}_d - \boldsymbol{Y}^{(t-1)}\boldsymbol{Y}^{(t-1)\top}. \tag{32}$$

The projected current-task directions are given by:

$$\boldsymbol{Z}_t = \boldsymbol{P}_{\mathrm{hist}}^{\perp}\widetilde{\boldsymbol{Y}}_t. \tag{33}$$

We then perform a reduced QR decomposition:

$$\boldsymbol{Z}_t = \boldsymbol{Q}_t\boldsymbol{T}_t. \tag{34}$$

The updated basis is set to:

$$\boldsymbol{Y}^{+} = \left[\boldsymbol{Y}^{(t-1)}, \boldsymbol{Q}_t\right]. \tag{35}$$

Provided that $\boldsymbol{Z}_t$ has full column rank, $\boldsymbol{Q}_t$ spans the same subspace as $\boldsymbol{Z}_t$ and therefore satisfies:

$$\boldsymbol{Y}^{(t-1)\top}\boldsymbol{Q}_t = \boldsymbol{0}, \qquad \boldsymbol{Q}_t^{\top}\boldsymbol{Q}_t = \boldsymbol{I}_r. \tag{36}$$

Thus, this additional operation preserves both the frozen historical directions and the cross-task orthogonality required by KRE. It is particularly important when using adaptive Euclidean optimizers, whose coordinate-wise transformations need not preserve the historical orthogonal complement.

Optimization Procedure. The optimization of each Stiefel-constrained parameter proceeds in four steps. First, we convert the Euclidean gradient obtained by back-propagation into the Riemannian gradient using Eq. (23). Second, we apply the underlying Euclidean optimizer and project the resulting optimizer-induced displacement back onto the tangent space using Eq. (25). Third, we retract the updated parameter onto the Stiefel manifold using Eq. (28). Fourth, for tasks after the first one, we restore the frozen historical directions and re-orthonormalize the newly introduced directions within their orthogonal complement to preserve cross-task orthogonality. This modular procedure preserves both the Stiefel constraint and the cross-task orthogonality required by KRE while retaining standard adaptive optimizers and their internal optimization states.

B Details of Margin-Aware Constrained Updates

This section provides the derivation of the margin-aware constrained update introduced in the main text. We first derive the first-order margin constraint and then obtain its closed-form solution.

First-Order Margin Constraint. Let $\boldsymbol{\theta}_r$ denote the trainable router parameters. To make the parameter dependence explicit, we write the current routing score as $z_t(x;\boldsymbol{\theta}_r)$. Since historical routers are frozen, their routing scores remain fixed with respect to $\boldsymbol{\theta}_r$. The routing margin is defined as:

$$\Delta_t(x;\boldsymbol{\theta}_r) = z_t(x;\boldsymbol{\theta}_r) - \max_{i<t} z_i(x). \tag{37}$$

Given a target margin $\gamma > 0$, the allowable first-order margin change is defined as:

$$b_t(x;\boldsymbol{\theta}_r) = \min\left(0, \gamma - \Delta_t(x;\boldsymbol{\theta}_r)\right). \tag{38}$$

For an update $\boldsymbol{d}$, the first-order approximation of the routing margin is given by:

$$\Delta_t(x;\boldsymbol{\theta}_r + \boldsymbol{d}) \approx \Delta_t(x;\boldsymbol{\theta}_r) + \nabla_{\boldsymbol{\theta}_r}\Delta_t(x;\boldsymbol{\theta}_r)^{\top}\boldsymbol{d}. \tag{39}$$

To constrain decreases of the linearized margin before reaching γ, while allowing any surplus above γ to be consumed, we impose the following condition:

$$\Delta_t(x;\boldsymbol{\theta}_r) + \nabla_{\boldsymbol{\theta}_r}\Delta_t(x;\boldsymbol{\theta}_r)^{\top}\boldsymbol{d} \geq \min\left(\Delta_t(x;\boldsymbol{\theta}_r), \gamma\right). \tag{40}$$

Equivalently, the constraint can be written as:

$$\nabla_{\boldsymbol{\theta}_r} \Delta_t(x; \boldsymbol{\theta}_r)^\top \boldsymbol{d} \geq \min\left(0, \gamma - \Delta_t(x; \boldsymbol{\theta}_r)\right) = b_t(x; \boldsymbol{\theta}_r). \tag{41}$$

When $\Delta_t(x; \boldsymbol{\theta}_r) < \gamma$, we have $b_t(x; \boldsymbol{\theta}_r) = 0$, so the update cannot further decrease the linearized margin. When $\Delta_t(x; \boldsymbol{\theta}_r) \geq \gamma$, we have $b_t(x; \boldsymbol{\theta}_r) \leq 0$, allowing the linearized update to consume part of the surplus. As described in the main text, we average the per-example margin gradients and bounds over the samples $\mathcal{B}_t$ contributing to an optimizer step:

$$\boldsymbol{g}_t = \frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}_r} \Delta_t(x; \boldsymbol{\theta}_r), \qquad b_t = \frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} b_t(x; \boldsymbol{\theta}_r). \tag{42}$$

The resulting constraint is given by:

$$\boldsymbol{g}_t^\top \boldsymbol{d} \geq b_t. \tag{43}$$

Closed-Form Correction. We first obtain a router update $\boldsymbol{d}_0 \in \mathcal{U}_t$. To require the corrected update to remain in the same subspace, the final update is obtained by solving:

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \frac{1}{2} \|\boldsymbol{d} - \boldsymbol{d}_0\|_2^2 \quad \text{s.t.} \quad \boldsymbol{d} \in \mathcal{U}_t, \qquad \boldsymbol{g}_t^\top \boldsymbol{d} \geq b_t. \tag{44}$$

We likewise restrict $\boldsymbol{g}_t$ to $\mathcal{U}_t$, such that the correction along $\boldsymbol{g}_t$ remains in the same update subspace. For $\boldsymbol{g}_t \neq \mathbf{0}$, the corresponding Lagrangian is defined as:

$$\mathcal{J}(\boldsymbol{d}, \lambda) = \frac{1}{2} \|\boldsymbol{d} - \boldsymbol{d}_0\|_2^2 + \lambda \left(b_t - \boldsymbol{g}_t^\top \boldsymbol{d}\right), \tag{45}$$

where $\lambda \geq 0$. The stationarity condition is given by:

$$\nabla_{\boldsymbol{d}} \mathcal{J} = \boldsymbol{d} - \boldsymbol{d}_0 - \lambda \boldsymbol{g}_t = \mathbf{0}. \tag{46}$$

The resulting update direction is therefore given by:

$$\boldsymbol{d} = \boldsymbol{d}_0 + \lambda \boldsymbol{g}_t. \tag{47}$$

If $\boldsymbol{g}_t^\top \boldsymbol{d}_0 \geq b_t$, the candidate update already satisfies the constraint and $\lambda^* = 0$. Otherwise, the constraint is active, and the corresponding multiplier is given by:

$$\lambda^* = \frac{b_t - \boldsymbol{g}_t^\top \boldsymbol{d}_0}{\|\boldsymbol{g}_t\|_2^2}. \tag{48}$$

Combining the two cases, the multiplier is given by:

$$\lambda^* = \frac{\left[b_t - \boldsymbol{g}_t^\top \boldsymbol{d}_0\right]_+}{\|\boldsymbol{g}_t\|_2^2}, \tag{49}$$

where $[a]_+ = \max(0, a)$. The final update is therefore given by:

$$\boldsymbol{d}^* = \boldsymbol{d}_0 + \frac{\left[b_t - \boldsymbol{g}_t^\top \boldsymbol{d}_0\right]_+}{\|\boldsymbol{g}_t\|_2^2} \boldsymbol{g}_t. \tag{50}$$

When $\boldsymbol{g}_t = \mathbf{0}$, the constraint is automatically satisfied because $b_t \leq 0$, and we set $\boldsymbol{d}^* = \boldsymbol{d}_0$.

Computation of $\boldsymbol{d}_0$. Let $\widetilde{\boldsymbol{d}}_{0,l}$ denote the unconstrained optimizer update for the l-th router parameter block, and let $\boldsymbol{M}_{t,l}$ be a matrix whose columns form an orthonormal basis of the corresponding historical subspace. We obtain the constrained update by projecting $\widetilde{\boldsymbol{d}}_{0,l}$ onto the orthogonal complement of this historical subspace (Liang et al., 2025). Accordingly, the update subspace is $\mathcal{U}_{t,l} = \mathrm{span}(\boldsymbol{M}_{t,l})^\perp$, and the projected update is given by:

$$\boldsymbol{d}_{0,l} = (\boldsymbol{I} - \boldsymbol{M}_{t,l} \boldsymbol{M}_{t,l}^\top) \widetilde{\boldsymbol{d}}_{0,l}. \tag{51}$$

The overall update subspace $\mathcal{U}_t$ is formed by the block-wise subspaces $\{\mathcal{U}_{t,l}\}$, and the candidate update $\boldsymbol{d}_0$ is obtained by concatenating the projected updates of all trainable router parameter blocks.

Table 4: Task sequences used in our experiments. Orders 1 and 2 are constructed from SuperNI, while Orders 3 and 4 correspond to the Long Sequence benchmark.

Benchmark	Order	Task Sequence
SuperNI	1	task1572 → task363 → task1290 → task181 → task002 → task1510 → task639 → task1729 → task073 → task1590 → task748 → task511 → task591 → task1687 → task875
	2	task748 → task073 → task1590 → task639 → task1572 → task1687 → task591 → task363 → task1510 → task1729 → task181 → task511 → task002 → task1290 → task875
Long Sequence	3	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
	4	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC

Table 5: Tasks used in the SuperNI benchmark.

Dataset	Task Type	Metric
task639_multi_woz_user_utterance_generation	Dialogue generation	ROUGE-L
task1590_diplomacy_text_generation	Dialogue generation	ROUGE-L
task1729_personachat_generate_next	Dialogue generation	ROUGE-L
task181_outcome_extraction	Information extraction	ROUGE-L
task748_glucose_reverse_cause_event_detection	Information extraction	ROUGE-L
task1510_evalution_relation_extraction	Information extraction	ROUGE-L
task002_quoref_answer_generation	Question answering	ROUGE-L
task073_commonsenseqa_answer_generation	Question answering	ROUGE-L
task591_sciq_answer_generation	Question answering	ROUGE-L
task511_reddit_tifu_long_text_summarization	Summarization	ROUGE-L
task1290_xsum_summarization	Summarization	ROUGE-L
task1572_samsum_summary	Summarization	ROUGE-L
task363_sst2_polarity_classification	Sentiment analysis	Accuracy
task875_emotion_classification	Sentiment analysis	Accuracy
task1687_sentiment140_classification	Sentiment analysis	Accuracy

C Additional Experimental Settings

C.1 Datasets and Task Sequences

We follow the experimental protocol of GainLoRA (Liang et al., 2025) and evaluate all methods on the same task sequences from the SuperNI (Wang et al., 2022) and Long Sequence (Razdaibiedina et al., 2023) benchmarks. SuperNI covers diverse NLP tasks, including dialogue generation, information extraction, question answering, summarization, and sentiment analysis. We use 15 tasks spanning these five categories. Long Sequence consists of 15 classification tasks drawn from several widely used NLP benchmarks. For both benchmarks, we adopt exactly the same task orders as GainLoRA. The four task sequences used in our experiments are summarized in Table 4. Detailed information about the individual tasks and their evaluation metrics is provided in Tables 5 and 6.

C.2 Implementation Details

All experiments are implemented in PyTorch and conducted on NVIDIA A6000 GPUs. We use AdamW for optimization with a constant learning-rate scheduler and no warmup. LoRA modules are applied to the query and value projections in each Transformer block. For LoRA-based baselines, the LoRA rank is set to 4 for Orders 1–2 and 8 for Orders 3–4, and Rex-MoE uses the same values

Table 6: Tasks used in the Long Sequence benchmark.

Dataset	Source	Task Type	Metric
Yelp	CL Benchmark	Sentiment analysis	Accuracy
Amazon	CL Benchmark	Sentiment analysis	Accuracy
DBpedia	CL Benchmark	Topic classification	Accuracy
Yahoo	CL Benchmark	Topic classification	Accuracy
AG News	CL Benchmark	Topic classification	Accuracy
MNLI	GLUE	Natural language inference	Accuracy
QQP	GLUE	Paraphrase detection	Accuracy
RTE	GLUE	Natural language inference	Accuracy
SST-2	GLUE	Sentiment analysis	Accuracy
WiC	SuperGLUE	Word sense disambiguation	Accuracy
CB	SuperGLUE	Natural language inference	Accuracy
COPA	SuperGLUE	Question answering	Accuracy
BoolQ	SuperGLUE	Question answering	Accuracy
MultiRC	SuperGLUE	Question answering	Accuracy
IMDB	Standalone	Sentiment analysis	Accuracy

for its per-task incremental basis rank r. We use an effective batch size of 32 for all backbones. For T5-Large, the learning rate is set to 3×10^{-4} with a maximum input length of 512 tokens; each task is trained for 100 epochs on SuperNI and 10 epochs on the Long Sequence benchmark. For Llama-2-7B and Qwen3-1.7B, the learning rates are set to 5×10^{-5} and 5×10^{-4}, respectively. Both models are trained for 50 epochs per task and use a maximum input length of 1024 tokens. The maximum generation length is set to 50 tokens for all backbones. For routing, each expert index is associated with a single two-layer query MLP and a corresponding key, both shared across all Transformer layers. The query MLP takes the average-pooled input token embeddings as input, with a hidden dimension of 100 for T5-Large and 50 for Llama-2-7B and Qwen3-1.7B. Its output is compared with the corresponding key using cosine similarity to obtain a routing score shared across layers. For the Stiefel-constrained basis factors, projection and retraction are performed at every optimization step.

For baselines, we follow their official implementations and recommended hyperparameters whenever available. When hyperparameter selection is required, we split each task's training set into training and validation subsets at a ratio of 4:1 and select hyperparameters based solely on validation performance. Unless explicitly varied in an ablation or sensitivity analysis, Rex-MoE uses $\lambda_{\text{pref}} = 0.1$, $\gamma = 0.2$, and $\tau = 0.1$. Once the hyperparameters are fixed, Rex-MoE and all compared baselines are trained on the full training set of each task, with test sets used only for evaluation after task training, never for hyperparameter selection. The main benchmark and ablation experiments are repeated three times with different random seeds, and the average results are reported.

C.3 Settings for Analysis Experiments

We provide implementation details for the analysis variants. Unless otherwise specified, all variants follow the same training protocol and routing mechanism as Rex-MoE and are retrained under their corresponding structural constraints.

Cross only and New-new only variants. The core matrix is partitioned into historical–historical, historical–new, new–historical, and new–new blocks according to historical and newly introduced basis directions. For Cross only, the first task trains the standard new–new block because no historical basis is available, while from the second task onward only the two cross-subspace blocks are retained. For New-new only, only the new–new block is retained for every task. All excluded core blocks are fixed to zero throughout training. Historical experts and the routing mechanism are preserved in both variants.

Random historical basis control. We replace the basis directions used by the current expert with randomly generated orthogonal directions while preserving the same basis dimensions and parameterization. The current expert uses a separate set of randomly initialized orthonormal bases, while

Table 7: Average performance (AP) of Rex-MoE on T5-Large across three runs, reported as mean $\pm$ standard deviation.

Method	Order 1	Order 2	Order 3	Order 4
Rex-MoE	51.01 ± 0.06	50.44 ± 0.50	79.56 ± 0.11	80.42 ± 0.30

Table 8: Comparison between full routing and current-only interventions on Qwen3-1.7B under SuperNI Order 1. Mean performance is computed as the unweighted mean of the current-task scores across stages 2–15. "Current-only (w/o renorm)" keeps the learned current-expert weight and sets all historical-expert weights to zero, while "Current-only (renorm)" additionally sets the current-expert weight to one.

Setting	Mean performance ↑	Full − Setting	Tasks improved
Full routing	56.53	–	–
Current-only (w/o renorm)	37.78	18.75	14/14
Current-only (renorm)	32.19	24.34	14/14

historical experts retain their original bases and cores. The random directions replacing the historical bases are kept fixed, while the newly introduced directions remain trainable. This preserves the trainable parameter budget while removing reuse of learned historical directions.

D Additional Experimental Results

D.1 Stability Across Runs

To assess the robustness of Rex-MoE on T5-Large, we report the mean and standard deviation of AP across three runs. As shown in Table 7, Rex-MoE exhibits consistently low variation across all four task orders, with standard deviations ranging from 0.06 to 0.50. These results indicate that the performance of Rex-MoE is stable across the evaluated runs.

D.2 Additional Analysis of CRR

In the main text, we assess historical-expert contributions by removing their outputs while preserving the learned routing weight of the current expert. We further consider a renormalized current-only intervention, where all historical-expert weights are set to zero and the current-expert weight is set to one. Both interventions are applied to the same trained checkpoints at evaluation time without retraining or updating model parameters. As shown in Table 8, the renormalized intervention reduces current-task performance on all 14 tasks, with an average drop of 24.34 points relative to full routing. Thus, restoring the total routing weight to one does not recover the performance of the full mixture. These results support the contribution of historical experts to current-task performance under the learned routing.

D.3 Performance under Reduced Expansion

To further examine whether the gains of Rex-MoE primarily arise from a larger additional parameter budget, we evaluate more compact expansion schedules. Parameter counts include all additionally introduced basis, learned core, and router parameters accumulated across tasks. The frozen backbone and structural zero padding are excluded. In addition to the default 4–4 setting, we reduce the incremental rank to 3 while keeping the first-task rank at 4, denoted as 4–3. To further examine different allocations of adaptation capacity between the first and subsequent tasks, we consider an initial rank of 16 with incremental ranks of 3, 2, or 1, denoted as 16–3, 16–2, and 16–1, respectively. These settings result in cumulative ranks of 46, 58, 44, and 30 after 15 tasks. As shown in Table 9, Rex-MoE maintains strong performance under reduced expansion. With the first-task rank unchanged, the 4–3 setting reduces the additional parameter budget from 21.30M to 16.96M while achieving 50.60 AP, only 0.41 points below the default setting. It still outperforms GainLoRA in

Table 9: Performance and parameter comparison under different expansion schedules on SuperNI Order 1 with T5-Large. For Rex-MoE, "a–b" denotes an initial rank of a for the first task and an incremental rank of b for each subsequent task. R denotes the cumulative rank after 15 tasks, and Params. reports the total number of additionally introduced parameters.

Method	R	Params.	AP↑	FT↓
GainLoRA (InfLoRA) ($r = 4$)	–	20.78M	46.21	2.40
Rex-MoE 4–4	60	21.30M	**51.01**	**0.34**
Rex-MoE 4–3	46	16.96M	50.60	0.71
Rex-MoE 16–3	58	20.68M	50.78	0.88
Rex-MoE 16–2	44	16.34M	50.09	0.63
Rex-MoE 16–1	30	12.06M	49.77	0.84

Table 10: Hyperparameter sensitivity of Rex-MoE on SuperNI Order 1.

λ_{pref}	0.005	0.1	0.2	0.3	0.5
AP↑	50.28	51.01	50.65	50.76	50.31
FT↓	0.81	0.34	0.86	0.89	0.81
γ	0.1	0.2	0.3	0.4	0.5
AP↑	50.98	51.01	50.55	50.41	50.41
FT↓	0.64	0.34	0.85	0.68	0.68
τ	0.05	0.1	0.15	0.2	0.3
AP↑	50.71	51.01	50.30	49.61	49.50
FT↓	0.87	0.34	1.40	1.68	2.47

both AP and FT with fewer additional parameters. The 16–3 setting also improves over GainLoRA with a slightly smaller parameter budget, while the 16–2 and 16–1 settings retain strong performance with further reductions in additional parameters. These results indicate that the gains of Rex-MoE cannot be explained solely by a larger additional parameter budget. They are also consistent with the benefit of knowledge reuse: accumulated bases provide reusable adaptation directions for subsequent tasks, supporting effective adaptation under reduced expansion.

D.4 Hyperparameter Sensitivity

We analyze the sensitivity of Rex-MoE to three key hyperparameters: the preference weight λ_{pref}, the target routing margin γ, and the routing temperature τ. As shown in Table 10, varying λ_{pref} yields AP ranging from 50.28 to 51.01 and FT from 0.34 to 0.89, while varying γ yields AP ranging from 50.41 to 51.01 and FT from 0.34 to 0.85. These results show limited performance variation over the tested ranges of preference weights and target margins. For τ, moderate values maintain comparable AP. Larger temperatures are associated with lower AP and greater forgetting, consistent with less concentrated routing. The default setting, $\lambda_{\text{pref}} = 0.1$, $\gamma = 0.2$, and $\tau = 0.1$, provides a favorable balance between average performance and knowledge retention.

D.5 Routing Weight Analysis

Figure 4 visualizes the average routing weights of T5-Large on SuperNI Order 1 using the checkpoint obtained after learning all 15 tasks. Each row reports the routing weights averaged over the test samples of the corresponding task, with all 15 experts available during evaluation. The routing matrix exhibits a clear diagonal structure, with the corresponding task expert generally receiving a relatively large weight. Historical experts also receive appreciable routing weight for many tasks, illustrating how the router combines task-specific and historical experts. Notably, most tasks assign negligible routing weights to experts introduced later in the sequence, indicating that subsequently introduced experts contribute little to earlier tasks.

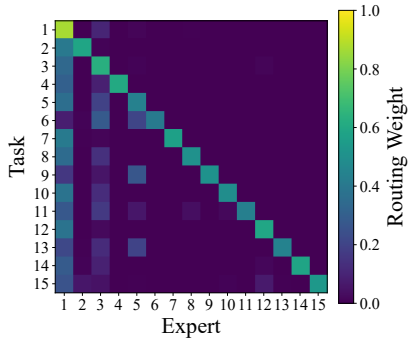


Figure 4: Average routing weights of T5-Large on SuperNI Order 1 after learning all 15 tasks.

Table 11: Inference FLOPs and MACs on T5-Large with an input shape of $(1, 128)$.

Method	FLOPs (G)	MACs (G)
Original	194.25	97.10
GainLoRA	198.79	99.37
Rex-MoE	198.92	99.43

D.6 Computational Cost

Since all experts share the same cumulative input and output bases, their routed outputs can be computed by first aggregating the expert cores:

$$\sum_{i=1}^{t} \alpha_i^{(t)}(x) E_i(\boldsymbol{h}) = \boldsymbol{U}^{(t)} \left(\sum_{i=1}^{t} \alpha_i^{(t)}(x) \boldsymbol{S}_i^{(t)} \right) \boldsymbol{V}^{(t)\top} \boldsymbol{h}. \tag{52}$$

Thus, the shared basis projections are performed only once rather than separately for each expert, avoiding repeated computation as the expert pool expands. The remaining additional computation is confined to the compact core space and lightweight routing.

Table 11 compares the floating-point operations (FLOPs) and multiply-add operations (MACs) during inference on T5-Large after learning all 15 tasks. We compute FLOPs and MACs with an input shape of $(1, 128)$. "Original" denotes the T5-Large backbone without adaptation. Although Rex-MoE maintains multiple experts, all experts share the same cumulative input and output bases, allowing their routed core matrices to be aggregated before the basis projections. Consequently, the shared basis projections are performed once rather than separately for each expert, avoiding repeated computation across experts. As a result, Rex-MoE introduces only limited additional computation and remains comparable to GainLoRA in overall inference cost.