AIMMerging: Adaptive Model Merging Using Training Trajectories for Language Model Continual Learning

Yujie Feng[1,2*], **Jian Li**[1*], **Xiaoyu Dong**[2], **Pengfei Xu**[1], **Xiaohui Zhou**[1], **Yujia Zhang**[1]
Zexin Lu[2], **Yasha Wang**[3], **Alan Zhao**[1†], **Xu Chu**[3†], **Xiao-Ming Wu**[2†]
[1]AI Technology Center of OVB, Tencent, China
[2]The Hong Kong Polytechnic University, Hong Kong S.A.R. [3]Peking University, China
yuujiefeng@tencent.com, xiao-ming.wu@polyu.edu.hk

Abstract

Continual learning (CL) is essential for deploying large language models (LLMs) in dynamic real-world environments without the need for costly retraining. Recent model merging-based methods have attracted significant attention, but they still struggle to effectively manage the trade-off between learning new knowledge and preventing forgetting, a challenge largely stemming from suboptimal number of merges and merging frequency. In this paper, we introduce Adaptive Iterative Model Merging (AimMerging), a novel CL framework that utilizes learning and forgetting signals from the training trajectory to dynamically monitor the model's training status. Guided by dynamic monitoring, the training trajectory-guided merge controller adaptively determines the timing and frequency of iterative fusion, while the rehearsal-based knowledge fusion module computes the merging weights and executes the fusion. Comprehensive experiments on three CL benchmarks with various model sizes (from 770M to 13B) demonstrate that AimMerging achieves significant performance improvements over existing state-of-the-art methods, with an average relative improvement of 80% and 59% on FWT and BWT, respectively. The source code[1] is provided for reproducibility.

1 Introduction

Continual learning (CL) is vital for the effective deployment of large language models (LLMs) in evolving environments, allowing them to sequentially acquire new knowledge and circumventing the necessity of costly retraining (Liao et al., 2025; Eskandar et al., 2025; Wang et al., 2024c; Jiang et al., 2024; Yu et al., 2024; Chang et al., 2024). However, the core challenge in CL lies in effectively balancing the retention of previously learned knowledge (mitigating catastrophic forgetting, CF (McCloskey and Cohen, 1989)) with the acquisition of new knowledge (facilitating knowledge transfer, KT (Ke et al., 2021)). Successfully managing this inherent trade-off is vital for practical deployment.

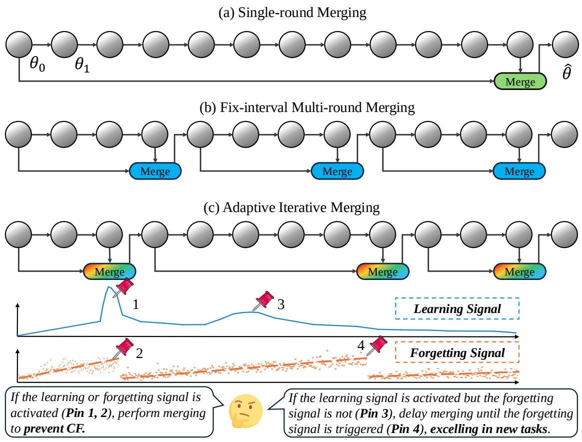

Figure 1: Illustration of three different model merging strategies. Guided by the learning and forgetting signals extracted from the training trajectory, our AimMerging adaptively adjusts the merging intervals and frequency, thereby enhancing CL performance.

Recent model merging methods (Dou et al., 2024; Wan et al., 2024; Yadav et al., 2024) have gained prominence for CL, largely due to their capacity for KT. Traditional approaches typically involve a single-round merge, commonly applied between pre- and post-training models, using global or fine-grained strategies (Figure 1(a)). Departing from single-round methods, Feng et al. (2025) proposed a recurrent framework that merges models iteratively after fixed training steps (Figure 1(b)), showing that leveraging intermediate training states through multiple merges can enhance performance.

This multi-round merging paradigm reveals significant potential and highlights the importance of optimizing the merging process. Inspired by these promising results, a critical question emerges:

How can we determine the optimal timing and frequency of merging during training to further

*Equal contribution.
†Corresponding author.
[1]https://github.com/WoodScene/AimMerging

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 13420–13437
November 4-9, 2025 ©2025 Association for Computational Linguistics

enhance performance?

To this end, we propose a novel CL framework called **A**daptive **I**terative **M**odel **Merging** (**AimMerging**). It achieves dynamic monitoring of the training status by innovatively employing ***learning*** and ***forgetting signals*** extracted from the training trajectory. Based on these signals, AimMerging consists of two key modules: the **Training Trajectory-guided Merge Controller**, responsible for adaptively scheduling the timing and frequency of model merging, and the **Rehearsal-based Knowledge Fusion Module**, which performs the global merging operation.

More specifically, the ***learning signal*** is quantified by the change in model parameters across training steps, reflecting the model's acquisition progress for new knowledge. Analysis of its trend via a sliding window identifies periods of rapid learning (peak in Figure 2) or slow convergence (downward trend in Figure 2). The ***forgetting signal***, on the other hand, is derived from the loss on historical data, offering real-time insight into the extent of CF. It is triggered when the historical loss exceeds a predefined threshold or shows a notable rise, signifying potential knowledge loss.

These signals guide the merge controller with distinct functions. The learning signal, typically measured after a merge, helps determine the next merging interval, thereby influencing the overall merging frequency. In contrast, the forgetting signal is continuously monitored during training. It serves as a critical, real-time trigger, prompting an immediate merge when significant forgetting of historical knowledge occurs.

Leveraging the dynamic monitoring from the learning and forgetting signals, the **training trajectory-guided merge controller** adaptively determines the merging schedule by interpreting their interplay. Based on the findings from our preliminary study (see Section 2), the controller increases the merging frequency to proactively mitigate CF when the learning signal indicates a rapid learning phase or the forgetting signal is activated. Conversely, when the learning signal indicates a slow convergence phase or the forgetting signal remains inactive, the controller reduces the merging frequency, allowing the model to focus more on learning new knowledge. Through this interaction, our method strikes a better balance between retaining previous knowledge and excelling in new tasks. The **rehearsal-based knowledge fusion module** executes the global merge operation, utilizing the relative importance weights derived from the learning and forgetting signals to merge new and historical knowledge effectively. Extensive experiments demonstrate the strong performance of our method in addressing CL challenges.

Our main contributions are summarized as:

- We propose a novel adaptive iterative model merging **framework** (AimMerging) for CL. To the best of our knowledge, AimMerging is the first to leverage the training trajectory by extracting learning and forgetting signals to dynamically monitor the model's state and guiding adaptive scheduling of iterative model merging.
- We introduce two novel **techniques**: the training trajectory-guided merge controller and the rehearsal-based knowledge fusion module.
- Extensive **evaluation** on three CL benchmarks utilizing four backbones (from 770M to 13B) demonstrates that AimMerging significantly enhances knowledge transfer capabilities, achieving an average relative improvement of 80% (from -2.5% to -0.5%) in FWT and 59% (from -4.9% to -2.0%) in BWT , surpassing previous state-of-the-art methods.

2 Preliminary Study

In this section, we conduct two key analysis: (i) investigating the dynamic changes in the model's training states regarding new knowledge acquisition and historical knowledge forgetting, and (ii) examining the impact of model merging during training on both new and historical knowledge. These analyses provide valuable insights for optimizing the adaptive merging strategy.

We first define the problem and introduce the relevant concepts for better clarity. All experiments in this section are conducted on long sequence benchmarks using T5-large.

Problem Formulation Continual learning aims to progressively accumulate knowledge from a sequence of tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$. Each task $\mathcal{T}_k$ includes a distinct dataset $\mathcal{D}_k = \left\{\left(x_i^k, y_i^k\right)\right\}_{i=1}^{N_k}$ of size N_k, where $x_i^k \in \mathcal{X}_k$ and $y_i^k \in \mathcal{Y}_k$. The model, parameterized by Θ, is trained sequentially on these tasks to minimize the following objective:

$$\mathcal{L} = \mathbb{E}_{(x,y)\sim\bigcup_{k=1}^{K}\mathcal{D}_k}\left[-\log p_\Theta(y \mid x)\right] \quad (1)$$

In this work, we consider a practical scenario where a small portion of data from previous tasks

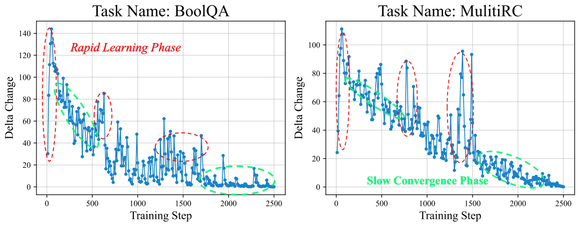

Figure 2: Parameter change for new knowledge acquisition during training.

Merging Strategy	OP	FWT	BWT
Fix Interval	78.3	-3.4	-2.7
Slow Convergence Phase^{+}	77.9	-3.8	-3.0
Rapid Learning Phase^{+}	**78.5**	**-2.7**	**-1.9**

Table 1: The impact of different merging strategies on performance. "+" indicates increased merging frequency during the corresponding phases.

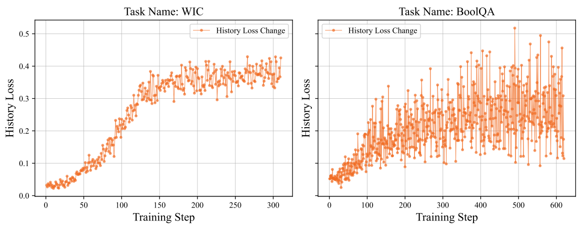

Figure 3: Changes in historical loss during training.

is stored in a memory buffer to facilitate the CL process. Specifically, we randomly store $|\mathcal{M}|$ samples from each task $\mathcal{T}_i$ in memory $\mathcal{M}_i$. During training, the model is jointly optimized on the new task data $\mathcal{D}_k$ and the memory buffer $\mathcal{M}_{<k}$.

Notation We consider a pre-trained model $\theta \in \mathbb{R}^n$ with n parameters. After training on task $\mathcal{T}_{k-1}$, the model are denoted as θ^{k-1}. Fine-tuning on a new task $\mathcal{T}_k$ produces updated parameters θ^k. The difference $\tau^k = \theta^k - \theta^{k-1}$, referred to as the *task vector* or *training residual* (Ilharco et al., 2023), represents task-specific parameter updates.

For traditional single-round merging methods, a merging function f_{merge} is typically used to combine the model θ^{k-1} and the fine-tuned model θ^k to obtain the final model: $\hat{\theta}^k = f_{\text{merge}}(\theta^{k-1}, \theta^k)$. In contrast, multi-round model merging methods perform merges during the training process. Specifically, assuming the total number of training iterations is J, θ_j^{k-1} represents the model's parameters at the j-th iteration. For example, if the interval between two consecutive merges is S (e.g., 100 training iterations), the merged model is represented as: $\hat{\theta}_{j+S}^{k-1} = f'_{\text{merge}}(\theta_j^{k-1}, \theta_{j+S}^{k-1})$. The model is then further trained based on $\hat{\theta}_{j+S}^{k-1}$.

2.1 Analysis of Knowledge Acquisition and Forgetting

New Knowledge Acquisition We measure the model's learning state for new knowledge by summing the absolute values of parameter changes within a fixed training interval, such as every 10 steps (Fig. 2). In the early stages of training, the parameter changes are large and show an upward trend, indicating the rapid learning phase. As training progresses, these changes decrease, signaling a slow convergence phase. Interestingly, peaks may reappear, suggesting the model revisits unlearned or challenging knowledge.

Based on two learning scenarios, we conducted two comparative experiments: (1) increasing merging frequency during the rapid learning phase, and (2) increasing merging frequency during the slow convergence phase. As shown in Table 1, the results reveal that increasing merging frequency during the rapid learning phase improves performance by preventing excessive accumulation of new knowledge. However, merging during the convergence phase leads to a decline in performance, likely due to redundancy in the stable model. This insight is valuable for refining the learning signal strategy in our method.

Forgetting of Historical Knowledge During training, we sample a batch of memory data from the buffer, feeding it into the model for loss calculation without gradient updates. This allows us to monitor historical knowledge loss, as shown in Fig. 3. The loss for historical knowledge increases as new knowledge is learned, aligning with expectations. When the loss exceeds a predefined threshold, the forgetting signal is triggered, prompting merging to mitigate forgetting.

2.2 Impact of Model Merging on New and Historical Knowledge

We perform one or two merges during training to observe the changes in loss for both new and historical knowledge (Figure 4). The results show that after each merge, historical knowledge loss decreases significantly, demonstrating effective mitigation of CF. However, the loss for new tasks increases, indicating that merging may interfere with learning new knowledge. Thus, selecting the appropriate merging timing is key to balancing new knowledge

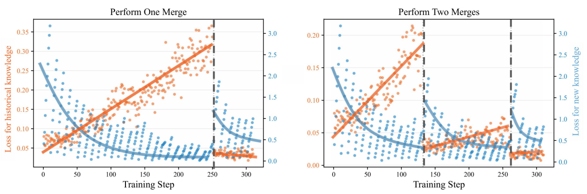

Figure 4: Loss change for new and historical knowledge after model merges during training.

acquisition and historical knowledge retention.

3 Proposed Method: AimMerging

Overview AimMerging employs two signals, the ***learning signal*** and the ***forgetting signal***, to monitor the model's training state. Based on the real-time fluctuations of these two signals, our approach reconfigures the training process into multiple iterative model merging cycles, guided by two core components: (i) ***Training Trajectory-guided Merge Controller***: adaptively selects the timing of model merges and dynamically adjusts the intervals between subsequent merges. (ii) ***Rehearsal-based Knowledge Fusion***: responsible for generating merging weights and applying memory-replay techniques to integrate new and historical knowledge.

3.1 The Design of Merge Controller

Assume the current task is $\mathcal{T}_k$, and θ_j^{k-1} represents the model's state after j training iterations [2].

Merge Controller with Learning Signal The core function of the learning signal is to dynamically adjust the merging interval S based on the model's current state for new knowledge. Assume the b-th merge is scheduled at the j-th training iteration, and the interval since the $(b-1)$-th merge is S_b. The task vector capturing the parameter update between these two successive merges is defined as:

$$\tau_b = \theta_j - \theta_{j-S_b} \tag{2}$$

By summing the absolute values of the task vector elements, we obtain a measure of the model's learning state for the new task, expressed as:

$$\Lambda_b = \sum_{i=1}^{n} |\tau_b^i| \,/\, S_b \tag{3}$$

where τ_b^i is the i-th element of the task vector. Dividing by S_b normalizes the value, allowing for fair comparison across intervals of varying lengths.

[2]For simplicity, we omit the superscript $k-1$ in the following descriptions.

We use Λ to assess the model's learning state for new knowledge. By comparing the current value Λ_b with the previous one Λ_{b-1}, we observe the parameter change trend and adjust the merging interval from S_b to S_{b+1}.

However, considering only the trend between two consecutive values may cause the learning signal to be overly sensitive to short-term fluctuations. To address this, we adopt a sliding window approach to analyze parameter change trends across multiple historical points and capture a more reliable overall trajectory. Specifically, we maintain a list to record the historical values of Λ, denoted as $\mathcal{H} = [\Lambda_1, \Lambda_2, \ldots, \Lambda_{b-1}]$. Given a sliding window length L_w, we compare the trends between consecutive entries, i.e., between Λ_b and Λ_{b-1}, Λ_{b-1} and $\Lambda_{b-2}, \ldots,$ up to Λ_{b-L_w+1} and Λ_{b-L_w}.

If upward trends dominate, indicating rapid learning phrase for new knowledge, we reduce the merging interval based on the magnitude of parameter changes (Case 1). If downward trends dominate, indicating slow convergence phase, we increase the merging interval (Case 3). If they are balanced, we keep the current interval (Case 2). The adjustment strategy is defined as:

$$S_{b+1} = \begin{cases} \max(S_{\min}, S_b/\gamma_{\text{learn}}^{-}), & \text{(Case 1)} \\ S_b, & \text{(Case 2)} \\ \min(S_{\max}, S_b \cdot \gamma_{\text{learn}}^{+}), & \text{(Case 3)} \end{cases} \tag{4}$$

where $S_{\min}$ and $S_{\max}$ denote the minimum and maximum allowed merging intervals, and γ_{learn} is a step-size adjustment factor. A cold-start phase is introduced at the beginning of training, during which no adjustments are made, lasting the length of the sliding window with an initial interval S_{init}.

Merge Controller with Both Learning and Forgetting Signals Relying only on the learning signal, the model adjusts the merging interval S based on the learning state for new knowledge, but this neglects the forgetting of historical knowledge, leading to a suboptimal merging strategy.

To address this, we further integrate the forgetting signal $\mathcal{F}$ to assist the controller adjust the merging strategy by considering both new and historical knowledge. The strategy triggers earlier or delayed merges depending on the forgetting signal, optimizing the merging interval.

We define the forgetting signal based on the loss change of historical data during the training of new tasks. In each iteration, a batch of historical data is sampled from the memory buffer and combined with the current task's batch. The historical data is used only for loss computation, excluding it from gradient updates. The forgetting signal is activated if the loss on historical tasks exceeds a predefined threshold, which is calculated using the average loss over the first $2/3 \times S_{b+1}$ steps, scaled by an adjustment factor γ_{forget} to produce the threshold δ_{b+1}. If the historical loss exceeds this threshold during subsequent training, the forgetting signal is triggered and the activation count is incremented as: $\mathcal{F}(b+1) = \mathcal{F}(b+1) + 1$.

Overall Workflow of the Merge Controller If the forgetting signal is activated multiple times (e.g., $\mathcal{F}(b+1) \geq \mathcal{F}_{max}$) before the scheduled merging interval S_{b+1}, an early merge is triggered to prevent further forgetting, i.e., the actual merging interval $S'_{b+1} < S_{b+1}$. Conversely, if the model reaches the predefined merging interval S_{b+1} without the forgetting signal being activated, the merge can be deferred to allow the model to continue focusing on learning new knowledge. The merge will be triggered either when the forgetting signal is activated ($S'_{b+1} > S_{b+1}$) or when the iteration count reaches the upper limit ($S'_{b+1} = 2 * S_{b+1}$).

In summary, our controller dynamically balances both learning and forgetting signals to optimize new knowledge acquisition while minimizing forgetting, resulting in an adaptive merging strategy.

3.2 Rehearsal-based Knowledge Fusion

When the merge controller initiates a merge, the knowledge fusion module performs the actual task knowledge fusion. Assume the b-th merge occurs at the j-th training iteration. The parameter change representing new knowledge is defined as:

$$\tau_{\text{new}_b} = \theta_j - \theta_{j-S'_b} \tag{5}$$

Next, we fine-tune θ_j on memory data for $S'_b/2$ steps, resulting in an updated model state $\theta_{j(M)}$. The task vector for historical knowledge is then:

$$\tau_{\text{past}_b} = \theta_{j(M)} - \theta_j \tag{6}$$

The final model parameters are updated by fusing both task vectors with learnable weights:

$$\hat{\theta}_j = \theta_{j-S'_b} + \alpha_1 \cdot \tau_{\text{new}_b} + \alpha_2 \cdot \tau_{\text{past}_b} \tag{7}$$

where α_1 and α_2 are the fusion weights, computed as follows:

- For τ_{new}, we assess the proportion of the upward trend in the learning signal's sliding window, $\mathcal{P}_{new} = L_{up}/L_w$, indicating the model's active learning of new knowledge.
- For τ_{past}, we compute the ratio of the forgetting signal's activation count $\mathcal{F}(b)$ to the maximum threshold $\mathcal{F}_{\text{max}}$, $\mathcal{P}_{past} = \mathcal{F}(b)/\mathcal{F}_{\text{max}}$, suggesting the extent of historical knowledge forgetting.

The fusion weights are then normalized as:

$$\alpha_1 = \frac{\mathcal{P}_{new}}{\mathcal{P}_{new} + \mathcal{P}_{past}}, \quad \alpha_2 = \frac{\mathcal{P}_{past}}{\mathcal{P}_{new} + \mathcal{P}_{past}} \tag{8}$$

After the fusion is completed, training continues from the updated model state $\hat{\theta}_j$.

4 Experiments and Analysis

Dataset We adopt the experimental setup from Du et al. (2024), using three CL benchmark datasets: (i) **Standard CL Benchmark**, which consists of five text classification tasks from Zhang et al. (2015). (ii) **Long Sequence Benchmark**, a more challenging evaluation scenario comprising 15 tasks (Razdaibiedina et al., 2023). (iii) **SuperNI Benchmark** (Wang et al., 2022a), a comprehensive benchmark for text generation, designed to evaluate 15 NLP tasks. Following Wang et al. (2023), we sample 1000 instances for training on each task and reserve 500 per class for testing. Different task sequences are evaluated for each benchmark, with detailed descriptions provided in Appendix C.

Metrics Let $a_{i,j}$ denote the testing performance on task $\mathcal{T}_i$ after training on task $\mathcal{T}_j$, and $a_{0,t}$ refers to the performance of training task t individually. We evaluate the overall performance (OP) (Chaudhry et al., 2018), backward transfer (BWT) (Ke and Liu, 2022), and forward transfer (FWT) (Lopez-Paz and Ranzato, 2017) after training on the final task:

$$\textbf{OP} = \frac{1}{K}\sum_{i=1}^{K} a_{i,K} \tag{9}$$

$$\textbf{BWT} = \frac{1}{K-1}\sum_{i=1}^{K-1} (a_{i,K} - a_{i,i}) \tag{10}$$

$$\text{FWT} = \frac{1}{K}\sum_{i=1}^{K} (a_{i,i} - a_{0,i}), \tag{11}$$

Baselines We compare AimMerging against various advanced methods, as well as both single-round and multi-round model merging methods. All methods are implemented using the LoRA framework

	Standard CL			Long Sequence			SuperNI		
	OP↑	FWT↑	BWT↑	OP↑	FWT↑	BWT↑	OP↑	FWT↑	BWT↑
SeqLoRA	43.7	-9.1	-50.4	11.6	-10.8	-73.4	6.4	-13.6	-31.0
IncLoRA	66.4	-8.7	-20.0	61.2	-11.1	-26.7	8.2	-15.1	-27.4
LoRAReplay	68.8	-9.0	-11.7	70.9	-11.3	-15.4	35.4	-12.4	-15.8
EWC* (Kirkpatrick et al., 2017)	50.3	-	-	45.1	-	-	35.7	-	-
L2P* (Wang et al., 2022b)	60.7	-	-	56.1	1.36	-16.3	12.7	-19.1	-8.0
LFPT5* (Qin and Joty, 2021)	72.7	-	-	69.2	-2.5	-12.8	34.4	-0.5	-14.5
MoELoRA* (Luo et al., 2024)	54.1	-6.2	-7.7	27.6	-8.6	-13.2	21.8	-7.2	-19.0
O-LoRA* (Wang et al., 2023)	75.8	-5.9	-3.8	69.6	-8.2	-4.1	25.9	-0.1	-24.6
TaSL (Feng et al., 2024b)	76.3	-5.4	-4.0	74.4	-7.9	-5.3	38.9	-1.2	-10.8
MIGU* (Du et al., 2024)	76.6	-	-	76.5	-	-	-	-	-
VR-MCL (Wu et al., 2024)	76.0	-4.6	-3.7	74.8	-6.0	-4.9	41.1	0.2	-9.3
SAPT-LoRA (Zhao et al., 2024)	-	-	-	76.6	-5.1	-3.7	41.7	1.9	-6.7
Recurrent-KIF* (Feng et al., 2025)	**78.4**	-3.1	-2.8	77.8	-4.6	-3.6	43.3	0.4	-8.4
AimMerging (ours)	78.1	**-1.5**	**-0.4**	**77.9**	**-2.3**	**-1.8**	**44.3**	**2.2**	**-4.0**
MTL	80.3	-	-	81.8	-	-	50.7	-	-

Table 2: Overall results on three CL benchmarks using the T5-large model. We report Overall Performance (OP), Forward Transfer (FWT), and Backward Transfer (BWT) after training on the final task. All results are averaged over different task orders. Methods marked with $*$ are copied from previous papers. The last row represents upper bound performance.

for fairness. (1) ***SeqLoRA***: LoRA parameters are trained on a task sequence without regularization or sample replay. (2) ***IncLoRA***: incremental learning of LoRA parameters without regularization or sample replay. (3) ***LoRAReplay***: LoRA fine-tuning with a memory buffer. (4) ***EWC (Kirkpatrick et al., 2017)***: finetune LoRA with a regularization loss to prevent interference with previous tasks. (5) ***L2P (Wang et al., 2022b)***: dynamically selects and updates prompts from a pool on an instance-by-instance basis. (6) ***LFPT5 (Qin and Joty, 2021)***: learns a soft prompt that solves tasks and generates training samples for replay. (7) ***O-LoRA (Wang et al., 2023)***: extends IncLoRA to learn different LoRAs in orthogonal subspaces. (8) ***MoELoRA (Luo et al., 2024)***: a vanilla MoE with LoRA number equals to the task number. (9) ***SAPT (Zhao et al., 2024)***: uses pseudo samples and a shared attention framework to align PEFT block learning and selection. (10) ***MIGU (Du et al., 2024)***: updates important parameters based on gradient magnitude. (11) ***TaSL (Feng et al., 2024b)***: a single-round model merging method based on parameter importance. (12) ***VR-MCL (Wu et al., 2024)***: dynamically updates the distribution of parameter importance through memory replay. (13) ***Recurrent-KIF (Feng et al., 2025)***: a multi-round model merging method based on fixed merging intervals. Additionally, multi-task learning, referred to as ***MTL***, serves as the upper bound.

Training Details We evaluate AimMerging using different backbone models, including T5-large (Raffel et al., 2020), Qwen3 1.7B (Yang et al., 2025), LLaMA2-7B (Touvron et al., 2023), and LLaMA2-13B. For the learning signal, the initial merging interval S_{init} is set to 8, and the sliding window size L_w is set to 3. The minimum and maximum merging intervals, S_{min} and S_{max}, are set to 2 and 128, respectively. The adjustment factors γ^+_{learn} and γ^-_{learn} are selected based on the current merging interval: if $S > 64$, we set $\gamma^+_{\text{learn}} = 1.5$ and $\gamma^-_{\text{learn}} = 2$; otherwise, we set $\gamma^+_{\text{learn}} = 2$ and $\gamma^-_{\text{learn}} = 1.5$. For the forgetting signal, the threshold scaling factor γ_{forget} is set to 2, and the maximum number of allowed activations before forcing an early merge, $\mathcal{F}_{\max}$, is set to 3. Following Feng et al. (2025), 2% of the original training set is used for replay samples. All experiments are averaged over 3 runs. More details are in Appendix D.

4.1 Main Results

The overall CL results using the same T5-large backbone are summarized in Table 2.

Our Training Trajectory-based AimMerging Method Effectively Addresses Both CF and KT Challenges. Compared to traditional CL methods (LoRAReplay, EWC) and model merging approaches (O-LoRA, MoELoRA, TaSL), AimMerging outperforms them in both CF (increasing OP from 59.5% to 66.8% compared to O-LoRA) and

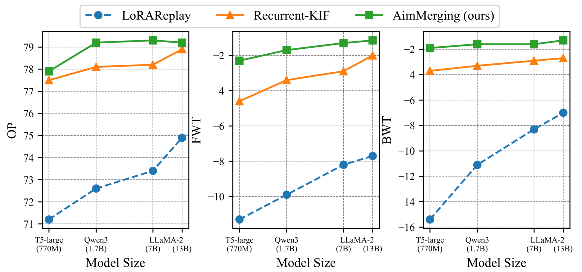

Figure 5: Performance of AimMerging with different backbones on the Long Sequence Benchmark.

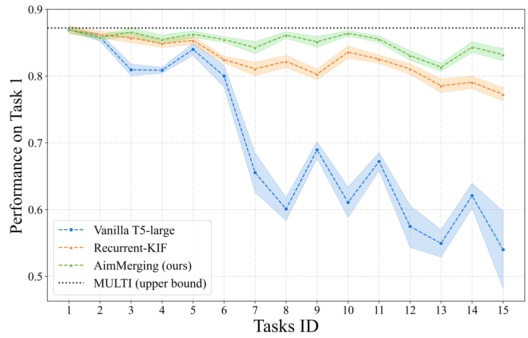

Figure 6: Performance trajectory of Task 1 on the longsequence benchmark during the CL process.

KT (improving BWT from -6.0% to -2.1% compared to VR-MCL). Moreover, AimMerging outperforms the state-of-the-art Recurrent-KIF, also based on multi-round merging, with significant improvements in both FWT (2.0%, from -2.5% to -0.5%) and BWT (2.9%, from -4.9% to -2.0%). These results show that AimMerging effectively balances preserving prior knowledge and excelling in new tasks.

AimMerging Demonstrates Consistent Superiority and Generalization Across Various Backbones. We validated the robustness of AimMerging using backbones ranging from 770M to 13B parameters, as shown in Figure 5. Across all sizes, AimMerging consistently outperforms baseline models. Notably, with the LLaMA2-7B backbone, AimMerging improves FWT from 78.2% to 79.3% and BWT from -2.9% to -1.6% compared to Recurrent-KIF, demonstrating its strong generalization ability across different model scales.

The Adaptive Iterative Merging Framework Enables Effective Knowledge Retention. Figure 6 illustrates the performance of the initial task after training on subsequent tasks. AimMerging significantly reduces catastrophic forgetting, with only a 4% performance drop after training on the final task. In contrast, vanilla replay shows a 32% drop, and Recurrent-KIF shows a 10% decline. These results underscore our model's strong backward knowledge transfer capability.

4.2 Ablation Study

We perform ablation studies to evaluate the effectiveness of the two key techniques in AimMerging. Results for task order 1 on the SuperNI Benchmark are shown in Table 3. Additional experiments, such as time complexity analysis and memory size impact, are provided in Appendix B.

Method	OP	FWT	BWT
AimMerging	**45.1**	**1.3**	-2.2
- LS	43.9	0.5	-3.4
- FS	44.3	0.8	-3.9
+ MGM	44.2	0.7	-3.7
+ IFM	44.9	1.2	**-2.1**

Table 3: Ablation study. "- LS", "- FS" refer to the removal of the learning signal and forgetting signal in our merge controller, respectively. "+ MGM" and "+ IFM" represent replacing the merging weights with manually set global merging weights and parameter importance-based fine-grained merging weights, respectively.

Effect of Training Trajectory-guided Merge Controller. To validate the contribution of the learning signal and forgetting signal to the merge controller's decision-making, we remove the learning signal ("- LS") and forgetting signal ("- FS") individually. When only the learning signal is used, merging occurs whenever the model's iteration reaches the pre-defined interval S. When only the forgetting signal is used, merging occurs when the loss of historical tasks exceeds the threshold. The performance decline in Table 3 highlights the necessity of both signals. Using only one signal leads to focusing on either new knowledge learning or historical knowledge retention, while both signals allow better balance.

Effect of Rehearsal-based Knowledge Fusion Module. We replace the weight calculation method in our fusion mechanism with two alternatives: (i) Manually set global merging weights (via grid search). (ii) Parameter importance-based fine-grained merging weights (following Feng et al. (2025)). Our results show that weights based on the learning and forgetting signals outperform manually set weights, improving three evaluation metrics

LoRA Target Modules	OP	FWT	BWT
Attention Q V	45.1	1.3	-2.2
Attention Q K V O	45.3	1.4	-2.3
FFN	49.8	1.0	-1.9
Attention All + FFN	45.7	2.1	-2.0

Table 4: Ablation study on LoRA target modules, using T5-large as the backbone.

by 0.9%, 0.6%, and 1.5%. Compared to parameter importance-based weights, our method shows slightly better performance, demonstrating that the learning and forgetting signals effectively capture the relevance of task vectors for knowledge updating and retention. Moreover, our approach is more efficient, avoiding the computational overhead of calculating and storing parameter importance.

4.3 Effect of Adding LoRA at Different Positions in the Model

We further investigate the impact of adding LoRA to different positions within the Transformer block. A typical Transformer block consists of the query, key, and value (QKV) linear layers, the output linear layer (O) in the multi-head attention module, and the two linear layers in the feedforward network (FFN). Our analysis, presented in Table 4, demonstrates that applying LoRA to all of these linear layers results in the best overall performance.

4.4 Visualization

We visualize two key aspects of our method's effectiveness. Full results for all tasks are provided in Appendix A (Figure 9 - 13).

Visualizing How the Merge Controller Adjusts Merging Timing and Step Size Based on the Learning and Forgetting Signals. As shown on the left of Figure 7, for simpler training scenarios, the merge controller gradually increases the merging interval. In the early stages of training, when there is significant new knowledge update, increasing the merging frequency helps prevent forgetting of historical knowledge. While in later stages, reducing it avoids redundant merges. In contrast, for more complex scenarios shown on the right, our method dynamically adjusts the merging frequency based on changes in the training state. For example, the model enters multiple rapid learning phases again during the middle of training (indicated by the peaks in the figure), prompting an increase in merging frequency.

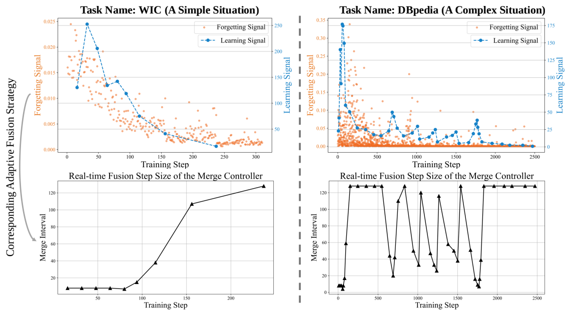

Figure 7: Visualizing merge controller behavior based on dynamic changes in learning and forgetting signals.

Visualizing Adaptive Iterative Merging's Effect on Catastrophic Forgetting. Figure 8 demonstrates the impact of our multi-round merging approach on historical knowledge forgetting. With vanilla LoRAReplay, the loss for historical tasks increases progressively, reflecting an escalating degree of forgetting. In contrast, our method effectively mitigates forgetting by selecting optimal merging points, enabling timely suppression before significant forgetting occurs. This results in a more stable or even decreasing overall loss trend, demonstrating the effectiveness of our approach in alleviating catastrophic forgetting.

5 Related Work

Continual learning (CL) (Zhou et al., 2024) focuses on the development of algorithms that enable models to accumulate knowledge from non-stationary data. In the era of LLMs, model mixture-based methods that employ parameter-efficient fine-tuning (PEFT) have become the dominant approach (Huang et al., 2024; Shi et al., 2024; Zhong et al., 2025), typically falling into two categories: model ensemble and model merging techniques.

Model ensemble methods allocate independent PEFT blocks to each task, effectively isolating task-specific parameters (Feng et al., 2023a; Pham et al., 2023; Ke et al., 2023; Xiang et al., 2025; Li et al., 2024; He et al., 2024; Wang et al., 2024a). For instance, O-LoRA (Wang et al., 2023) enforces orthogonality among LoRA adapters, while SAPT (Zhao et al., 2024) uses a selection module to combine task-specific blocks via task correlations. Though effective for knowledge preservation, they hinder inter-task transfer and scale poorly due to growing memory overhead (Zhang et al., 2025).

In contrast, model merging techniques combine

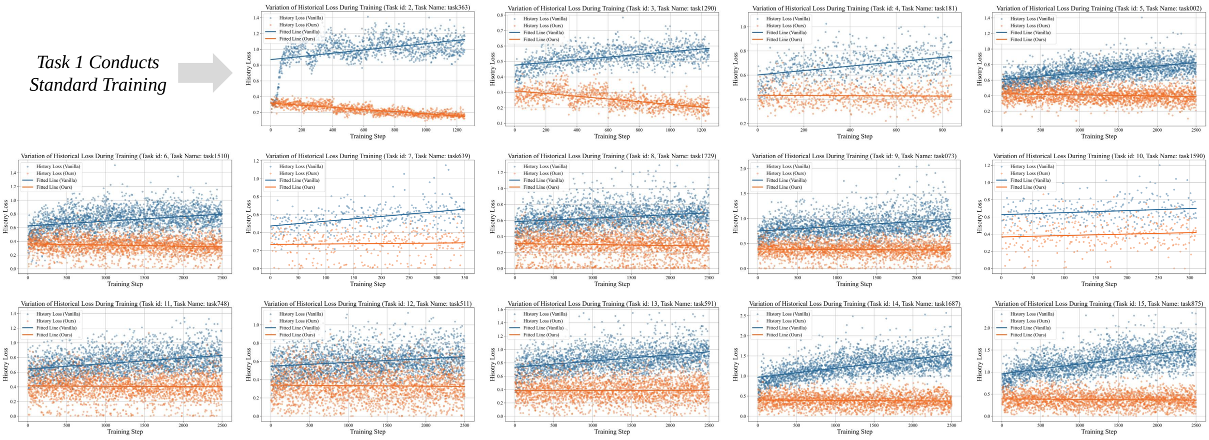

Figure 8: Visualization of the effect of AimMerging in alleviating catastrophic forgetting in the SuperNI benchmark.

multiple models into a single unified model (Cheng et al., 2024; Alexandrov et al., 2024; Ren et al., 2024), addressing memory constraints. For example, global model merging approaches (Wortsman et al., 2022; Ilharco et al., 2023) perform a weighted fusion of models before and after training, often assuming that all model parameters contribute equally to each task. Fine-grained approaches like Feng et al. (2024a) leverage parameter importance masks to enable neuron- or matrix-level fusion. Recently, Feng et al. (2025) introduced a multi-round merging paradigm for CL, demonstrating that integrating merges during model iterations can significantly enhance model performance. Yet key challenges persist: the optimal number, timing, and frequency of merges remain underexplored. To address this, we propose AimMerging, a novel adaptive iterative framework that leverages learning and forgetting signals to dynamically monitor the model's state. By analyzing training trajectory, AimMerging optimizes merging strategies, advancing the efficiency and effectiveness of CL.

6 Conclusion

In this paper, we introduce Adaptive Iterative Model Merging (AimMerging), a novel CL framework that enables dynamic monitoring of the training status by leveraging learning and forgetting signals extracted from the training trajectory. The framework consists of two key modules: the training trajectory-guided merge controller, which adaptively schedules the timing and frequency of model merging, and the rehearsal-based knowledge fusion module, which performs the global merging operation based on these signals. Extensive experiments validate the effectiveness of AimMerging in addressing the key challenges of continual learning.

Limitations

We acknowledge two limitations in our work. First, while our approach selectively determines model merging timing by monitoring parameter changes and historical task loss, it remains an open question whether alternative metrics such as gradient information or other indicators could more effectively capture learning states and forgetting phenomena. Second, current merging strategies involve semi-heuristic design choices regarding intervals and thresholds. Future research could focus on developing fully automated optimization methods that minimize the need for manual parameter tuning.

References

Anton Alexandrov, Veselin Raychev, Mark Mueller, Ce Zhang, Martin Vechev, and Kristina Toutanova. 2024. Mitigating catastrophic forgetting in language transfer via model merging. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17167–17186.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547.

Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, and Xiao Huang. 2024. Entity alignment with noisy annotations from large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. 2025. You don't need pre-built graphs for rag: Retrieval augmented generation with adaptive reasoning structures.

Xi Chen and Min Zeng. 2025. Prototype conditioned generative replay for continual learning in nlp. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12754–12770.

Feng Cheng, Ziyang Wang, Yi-Lin Sung, Yan-Bo Lin, Mohit Bansal, and Gedas Bertasius. 2024. Dam: Dynamic adapter merging for continual video qa learning. *arXiv preprint arXiv:2403.08755*.

Xiaoyu Dong, Yujie Feng, Zexin Lu, Guangyuan Shi, and Xiao-Ming Wu. 2024. Zero-shot cross-domain dialogue state tracking via context-aware auto-prompting and instruction-following contrastive decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8527–8540.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Wenyu Du, Shuang Cheng, Tongxu Luo, Zihan Qiu, Zeyu Huang, Ka Chun Cheung, Reynold Cheng, and Jie Fu. 2024. Unlocking continual learning abilities in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6503–6522.

Masih Eskandar, Tooba Imtiaz, Davin Hill, Zifeng Wang, and Jennifer Dy. 2025. Star: Stability-inducing weight perturbation for continual learning. *arXiv preprint arXiv:2503.01595*.

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

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. 2024b. Tasl: Continual dialog state tracking via task skill localization and consolidation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1266–1279.

Yujie Feng, Bo Liu, Xiaoyu Dong, Zexin Lu, Li-Ming Zhan, Xiao-Ming Wu, and Albert Lam. 2024c. Continual dialogue state tracking via reason-of-select distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7075–7087.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023a. Towards llm-driven dialogue state tracking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 739–755.

Yujie Feng, Jiangtao Wang, Yasha Wang, and Xu Chu. 2023b. Towards sustainable compressive population health: A gan-based year-by-year imputation method. *ACM Transactions on Computing for Healthcare*, 4(1):1–18.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S Yu, Xu Chu, and Xiao-Ming Wu. 2025. Recurrent knowledge identification and fusion for language model continual learning. *arXiv preprint arXiv:2502.17510*.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. 2024. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. *arXiv preprint arXiv:2411.06171*.

Zhiyuan Hu, Yuliang Liu, Jinman Zhao, Suyuchen Wang, WangYan WangYan, Wei Shen, Qing Gu, Anh Tuan Luu, See-Kiong Ng, Zhiwei Jiang, and Bryan Hooi. 2025. LongRecipe: Recipe for efficient long context generalization in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11857–11870, Vienna, Austria. Association for Computational Linguistics.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. *arXiv preprint arXiv:2403.01244*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Gangwei Jiang, Caigao Jiang, Zhaoyi Li, Siqiao Xue, Jun Zhou, Linqi Song, Defu Lian, and Ying Wei. 2024. Interpretable catastrophic forgetting of large language model fine-tuning via instruction vector. *arXiv preprint arXiv:2406.12227*.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in*

Neural Information Processing Systems, 34:22443–22456.

Zixuan Ke, Bing Liu, Wenhan Xiong, Asli Celikyilmaz, and Haoran Li. 2023. Sub-network discovery and soft-masking for continual learning of mixed tasks. *arXiv preprint arXiv:2310.09436*.

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

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. 2024. Revisiting catastrophic forgetting in large language model tuning. *arXiv preprint arXiv:2406.04836*.

Huanxuan Liao, Shizhu He, Yupu Hao, Jun Zhao, and Kang Liu. 2025. Data: Decomposed attention-based task adaptation for rehearsal-free continual learning. *arXiv preprint arXiv:2502.11482*.

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

Zexin Lu, Jing Li, Yingyi Zhang, and Haisong Zhang. 2021. Getting your conversation on track: Estimation of residual life for conversations. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 1036–1043. IEEE.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Quang Pham, Chenghao Liu, and Steven CH Hoi. 2023. Continual learning, fast and slow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Anastasia Razdai, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2022. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. 2024. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*.

Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, and Xiao-Ming Wu. 2024. Understanding layer significance in llm alignment. *arXiv preprint arXiv:2410.17875*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. 2024a. Self-expansion of pre-trained models with mixture of adapters for continual learning. *arXiv preprint arXiv:2403.18886*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024b. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

Xujia Wang, Haiyan Zhao, Shuo Wang, Hanqing Wang, and Zhiyuan Liu. 2024c. Malora: Mixture of asymmetric low-rank adaptation for enhanced multi-task learning. *arXiv preprint arXiv:2410.22782*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022a. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Yichen Wu, Long-Kai Huang, Renzhen Wang, Deyu Meng, and Ying Wei. 2024. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *The Twelfth International Conference on Learning Representations*.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2025. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*.

Yongxin Xu, Xinke Jiang, Xu Chu, Rihong Qiu, Yujie Feng, Hongxin Ding, Junfeng Zhao, Yasha Wang, and Bing Xie. 2025. Dearllm: Enhancing personalized healthcare via large language models-deduced feature correlations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 941–949.

Yongxin Xu, Ruizhe Zhang, Xinke Jiang, Yujie Feng, Yuzhen Xiao, Xinyu Ma, Runchuan Zhu, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024. Parenting: Optimizing knowledge selection of retrieval-augmented language models with parameter decoupling and tailored tuning. *arXiv preprint arXiv:2410.10360*.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. 2024. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S Yu, and Irwin King. 2024. Recent advances of multimodal continual learning: A comprehensive survey. *arXiv preprint arXiv:2410.05352*.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Jinman Zhao and Xueyan Zhang. 2024. Large language model is not a (multilingual) compositional relation reasoner. In *First Conference on Language Modeling*.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. 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)*, pages 11641–11661.

Yibo Zhong, Jinman Zhao, and Yao Zhou. 2025. Low-rank interconnected adaptation across layers. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17005–17029, Vienna, Austria. Association for Computational Linguistics.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. 2024. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*.

A Visualization

Here, we present the performance of our method across all datasets and tasks. Figures 9, 10, and

11 illustrate how our merge controller adjusts the merging strategy for all tasks in the SuperNI, LongSequence, and Standard benchmarks, respectively. Figures 12 and 13 also demonstrate the effectiveness of our method in alleviating catastrophic forgetting across all tasks.

B Additional Results

B.1 Effect of the Memory Size

We examine the effect of varying memory size on the performance of LoRAReplay and AimMerging. By adjusting the memory size per task $|M|$ to 2%, 5%, 10%, 50%, the results are presented in Table 5. As anticipated, increasing the memory size generally enhances the performance of all methods. However, AimMerging utilizes iterative knowledge fusion mechanism to effectively retain historical knowledge, resulting in superior performance compared to LoRAReplay.

	Memory Size			
	2%	5%	10%	50%
LoRAReplay	71.2	72.4	73.8	76.1
AimMerging	77.9	78.9	78.3	80.9

Table 5: Ablation study on memory size, using T5-large as the backbone.

B.2 Time Complexity Analysis

In this section, we discuss the time complexity challenges introduced by multi-round merging. Generally, multi-round merging methods tend to have higher time complexity than traditional merging approaches. To mitigate this, we optimized the time complexity during the design of our framework. Our forgetting signal requires monitoring the loss changes of historical data. To reduce complexity, we insert historical data into the new data batch during implementation, performing only loss calculation without updating gradients, thus avoiding additional training costs.

Furthermore, in our merging method, we directly use the parameter change between two successive merges, rather than merging the entire model before and after training, which further improves the efficiency of merging. Quantitatively, we compare the training time of our method with that of LoRAReplay, the single-round merging method TaSL, and Recurrent-KIF, which also performs multiple merges. The results are shown in Tablee 6.

Training Time (Min/Epoch)	T5-large	Qwen3-1.7B	LLaMA2-7B	LLaMA2-13B
LoRAReplay	1.4	3.3	4.5	6.6
TaSL	1.4	3.4	4.6	6.7
Recurrent-KIF	1.4	4.9	5.5	9.1
AimMerging	1.4	4.4	5.9	8.5

Table 6: Training time comparison across backbones.

Method	OP ↑	FWT ↑	BWT ↑
Replay	37.7	-13.5	-21.8
VR-MCL	44.9	-6.1	-15.7
Recurrent-KIF	46.4	-5.9	-14.6
AimMerging (ours)	**48.3**	**-3.0**	**-9.1**
Multi-task Learning	57.2	-	-

Table 7: Cross-dataset evaluation on a 19-task sequence (4 Standard CL tasks + 15 SuperNI tasks). AimMerging consistently outperforms baselines.

The results indicate that although our method takes slightly more time than LoRA Replay and TaSL, with an average increase of approximately 1.3 times, it delivers significant performance improvements. Moreover, compared to Recurrent-KIF, which also uses multi-round merging, our method benefits from adaptive merging timing, filtering out many unnecessary merges, and achieves lower time complexity through model design optimizations.

B.3 Generalizability of Learning and Forgetting Signals

To validate the robustness of our method in highly imbalanced or severely shifting environments, we further conducted cross-dataset experiments, combining 4 tasks from the Standard CL benchmark and 15 from the SuperNI Benchmark, and tested on a 19-task sequence. The results are shown in the table 7.

As shown in the table, our method still outperforms other baseline methods, with an average improvement of 1.9% on OP, 2.9% on FWT, and 5.5% on BWT compared to Recurrent-KIF. These results will be included in the revised paper.

B.4 Sensitivity Analysis of Hyperparameters

Our method involves several hyperparameters in the learning and forgetting signals. Specifically, in the learning signal we consider the initial merging interval S_{init}, the sliding window size L_w, and the range for merging intervals S_{min}, S_{max}; while in the forgetting signal, we consider the threshold scaling factor γ_{forget} and the maximum activa-

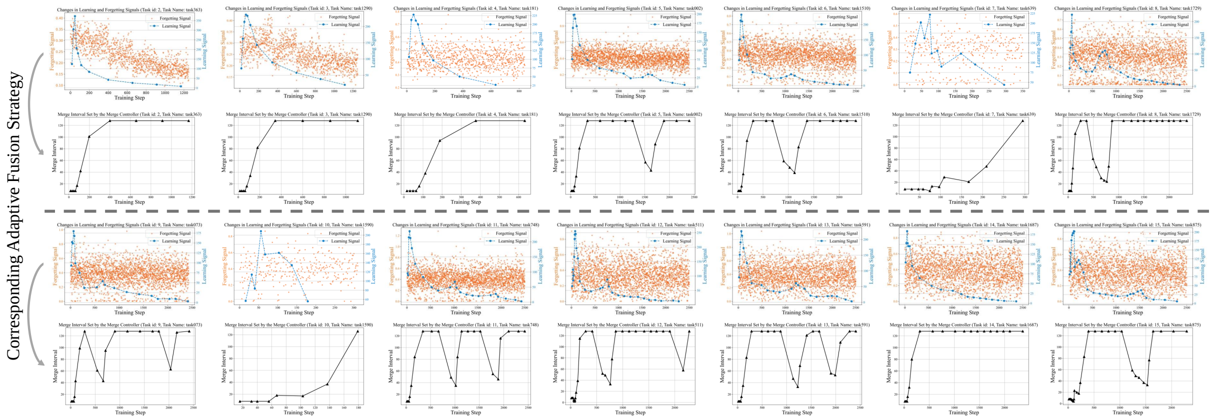

Figure 9: Visualizing the behavior of the merge controller based on dynamic changes in learning and forgetting signals in the SuperNI benchmark.

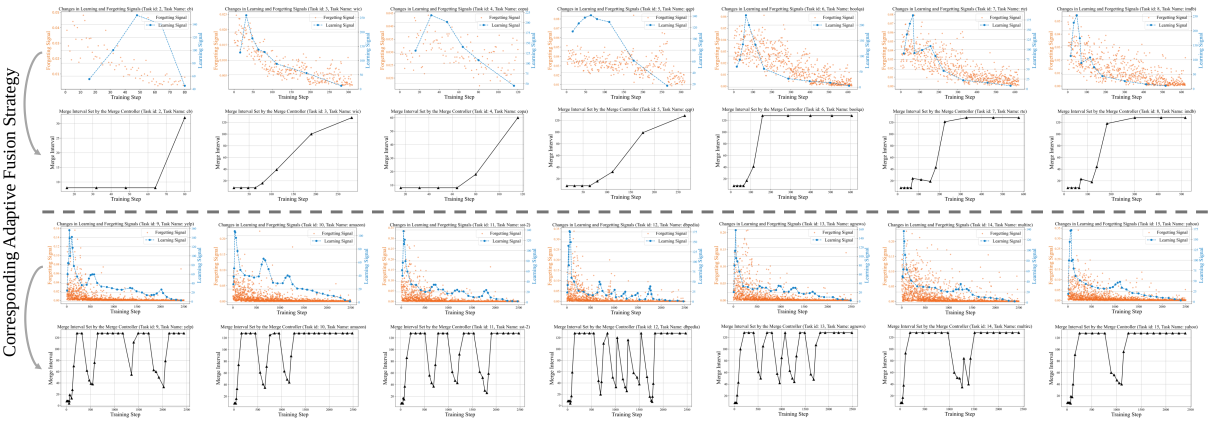

Figure 10: Visualizing the behavior of the merge controller based on dynamic changes in learning and forgetting signals in the Long Sequence benchmark.

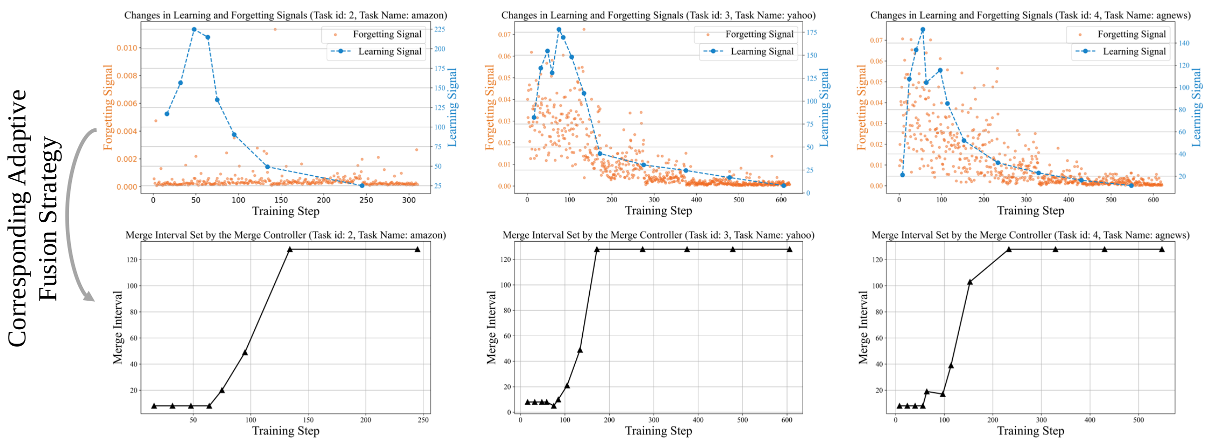

Figure 11: Visualizing the behavior of the merge controller based on dynamic changes in learning and forgetting signals in the Standard CL benchmark.

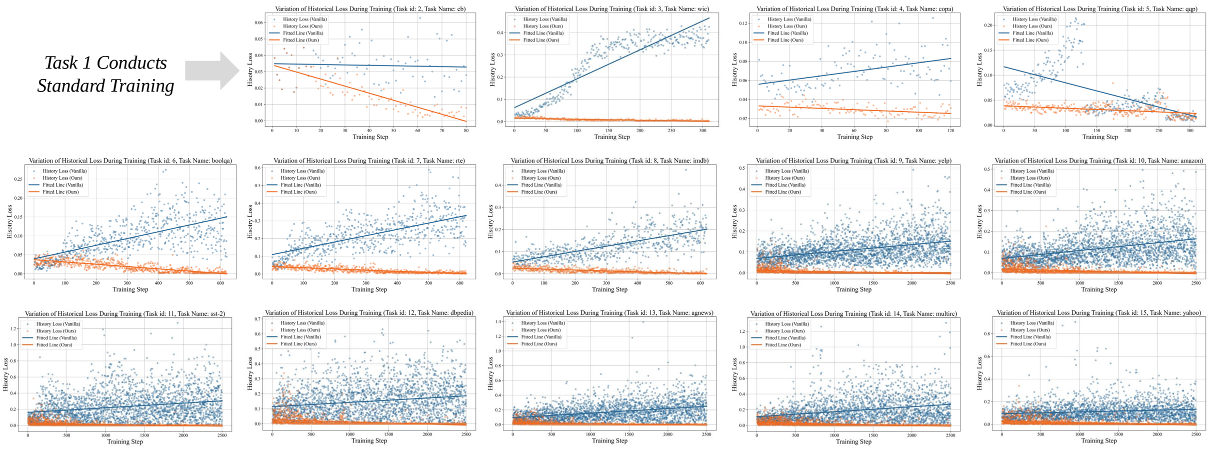

Figure 12: Visualization of the effect of AimMerging in alleviating catastrophic forgetting in the Longsequence benchmark.

Figure 13: Visualization of the effect of AimMerging in alleviating catastrophic forgetting in the Standard CL benchmark.

tion count F_{max}. To evaluate the sensitivity of our method to these hyperparameters, we conducted experiments on the Standard CL benchmark (task order 1) using the T5-large backbone model. The results are presented in Table 8.

As shown in the table, increasing S_{init} may lead to missing the optimal merging timing, resulting in more forgetting. Enlarging the sliding window size L_w improves the stability of the learning signal, but overly long windows can accumulate erroneous data, causing performance degradation. Overall, when hyperparameters are set within reasonable ranges, the model remains robust and is not highly sensitive. This demonstrates that the lack of automated hyperparameter tuning does not compromise the practicality or reproducibility of our method.

B.5 Comparison with Rehearsal-free Baselines

Our method relies on memory data to obtain the forgetting signal, and thus directly removing the memory buffer is not straightforward. To ensure fairness in comparison, we additionally equipped prior rehearsal-free baselines with the same memory buffer and re-evaluated them (i.e., fine-tuning for two epochs on memory data after standard training). The results on the SuperNI benchmark (task order 6) with the T5-large backbone are presented in Table 9.

As shown in the results, all baselines benefit from the memory buffer. However, AimMerging still consistently achieves the best performance across OP, FWT, and BWT. This demonstrates that our method retains its advantage even under comparable settings. We leave the development of a memory-free variant of AimMerging as an important direction for future work.

C Dataset Statistics

We adopt the experimental setup from Du et al. (2024), using three CL benchmark datasets: (i) **Standard CL Benchmark**, which consists of five text classification tasks from Zhang et al. (2015): AG News, Amazon Reviews, Yelp Reviews, DBpedia, and Yahoo Answers. (ii) **Long Sequence Benchmark**, a more challenging evaluation scenario comprising 15 tasks (Razdaibiedina et al., 2023): five from the Standard CL Benchmark, four from the GLUE benchmark (Wang, 2018), five from SuperGLUE (Wang et al., 2019), and the IMDB Movie Reviews dataset (Maas et al., 2011). (iii) **SuperNI Benchmark** (Wang et al., 2022a), a

Hyperparameter	Value	OP ↑	FWT ↑	BWT ↑
S_{init}	2	78.2	-1.5	-0.7
	8	78.3	-1.4	-0.6
	16	78.0	-1.7	-1.0
	32	77.7	-2.0	-1.5
L_w	2	78.3	-1.5	-0.6
	3	78.3	-1.4	-0.6
	4	78.4	-1.3	-0.7
	8	78.5	-1.5	-0.7
S_{min}, S_{max}	2, 128	78.3	-1.4	-0.6
	8, 128	78.0	-1.6	-0.9
	8, 64	78.2	-1.6	-0.7
	2, 64	78.5	-1.5	-0.7
γ_{forget}	2	78.3	-1.4	-0.6
	8	78.1	-1.5	-0.8
	16	78.0	-1.7	-0.9
	32	77.6	-1.9	-1.4
F_{max}	2	78.5	-1.3	-0.8
	3	78.3	-1.4	-0.6
	4	78.4	-1.5	-0.7
	8	78.0	-1.7	-1.0

Table 8: Sensitivity analysis of key hyperparameters on the Standard CL benchmark (task order 1) with T5-large. Results show that the method remains robust when hyperparameters are set within reasonable ranges.

comprehensive benchmark designed to evaluate a wide range of NLP tasks, includes tasks in dialogue generation (Xu et al., 2024), information extraction, question answering (Lu et al., 2021), summarization (Hu et al., 2025), and sentiment analysis (Xu et al., 2025; Chen et al., 2025).

Table 10 & 11 show details of the datasets we used for our experiments, along with their evaluation metrics. Overall, in SuperNI (Chen and Zeng, 2025), we choose 3 tasks from dialogue generation (Dialog) (Feng et al., 2024c; Dong et al., 2024), information extraction (IE), question answering (QA) (Zhao and Zhang, 2024), summarization (Sum) and sentiment analysis (SA), respectively.

For the Long Sequence benchmark (Wang et al., 2024b), this includes five tasks from the standard CL benchmark (AG News, Amazon reviews, Yelp reviews, DBpedia and Yahoo Answers), four from GLUE benchmark (MNLI, QQP, RTE, SST2), five from SuperGLUE benchmark (WiC, CB, COPA, MultiRC, BoolQ), and the IMDB movie reviews dataset (Feng et al., 2023b; Chen et al., 2024).

We report 7 different task orders used for our experiments in Table 12.

Method	OP ↑	FWT ↑	BWT ↑
Replay	35.6	-12.7	-15.4
O-LoRA	39.2	0.2	-9.4
MIGU	39.0	-1.1	-6.3
TaSL	41.9	-0.4	-5.9
MoCL	42.4	-0.3	-5.8
AimMerging (ours)	**44.1**	**2.3**	**-3.9**

Table 9: Comparison with rehearsal-free baselines on the SuperNI benchmark (task order 6) using T5-large. All methods benefit from memory buffer usage, but AimMerging achieves the best performance across all metrics.

D Implementation Details

Experiments are implemented using PyTorch and the Transformer library, running on 8 NVIDIA V100 GPUs with 32GB memory. The following hyperparameters are used: a learning rate of 3e-4, a batch sizes of 8, and training for 10 epochs. The LoRA settings are: $r = 8$, $\alpha = 32$, dropout = 0.05, targeting modules [q_proj,v_proj]. For testing: temperature = 0.02, top_p = 0, top_k = 1, num_beams = 1, max new tokens = 128.

It is worth noting that we used the same hyperparameters across different datasets and backbones, demonstrating the generalizability of our method without requiring extensive hyperparameter tuning for each specific setting.

Dataset name	Task	Metric
1. task639_multi_woz_user_utterance_generation	dialogue generation	Rouge-L
2. task1590_diplomacy_text_generation	dialogue generation	Rouge-L
3. task1729_personachat_generate_next	dialogue generation	Rouge-L
4. task181_outcome_extraction	information extraction	Rouge-L
5. task748_glucose_reverse_cause_event_detection	information extraction	Rouge-L
6. task1510_evalution_relation_extraction	information extraction	Rouge-L
7. task002_quoref_answer_generation	question answering	Rouge-L
8. task073_commonsenseqa_answer_generation	question answering	Rouge-L
9. task591_sciq_answer_generation	question answering	Rouge-L
10. task511_reddit_tifu_long_text_summarization	summarization	Rouge-L
11. task1290_xsum_summarization	summarization	Rouge-L
12. task1572_samsum_summary	summarization	Rouge-L
13. task363_sst2_polarity_classification	sentiment analysis	accuracy
14. task875_emotion_classification	sentiment analysis	accuracy
15. task1687_sentiment140_classification	sentiment analysis	accuracy

Table 10: The details of 15 datasets in the SuperNI Benchmark (Wang et al., 2022a).

Dataset name	Category	Task	Domain	Metric
1. Yelp	CL Benchmark	sentiment analysis	Yelp reviews	accuracy
2. Amazon	CL Benchmark	sentiment analysis	Amazon reviews	accuracy
3. DBpedia	CL Benchmark	topic classification	Wikipedia	accuracy
4. Yahoo	CL Benchmark	topic classification	Yahoo Q&A	accuracy
5. AG News	CL Benchmark	topic classification	news	accuracy
6. MNLI	GLUE	natural language inference	various	accuracy
7. QQP	GLUE	paragraph detection	Quora	accuracy
8. RTE	GLUE	natural language inference	news, Wikipedia	accuracy
9. SST-2	GLUE	sentiment analysis	movie reviews	accuracy
10. WiC	SuperGLUE	word sense disambiguation	lexical databases	accuracy
11. CB	SuperGLUE	natural language inference	various	accuracy
12. COPA	SuperGLUE	question and answering	blogs, encyclopedia	accuracy
13. BoolQA	SuperGLUE	boolean question and answering	Wikipedia	accuracy
14. MultiRC	SuperGLUE	question and answering	various	accuracy
15. IMDB	SuperGLUE	sentiment analysis	movie reviews	accuracy

Table 11: The details of 15 classification datasets in the Long Sequence Benchmark (Razdai et al., 2022). First five tasks correspond to the standard CL benchmark (Zhang et al., 2015).

Order	Benchmark	Task Sequence
1		dbpedia → amazon → yahoo → ag
2	Standard CL	dbpedia → amazon → ag → yahoo
3		yahoo → amazon → ag → dbpedia
4	Long Sequence	mnli → cb → wic → copa → qqp → boolqa → rte → imdb → yelp → amazon → sst-2 → dbpedia → ag → multirc → yahoo
5		yelp → amazon → mnli → cb → copa → qqp → rte → imdb → sst-2 → dbpedia → ag → yahoo → multirc → boolqa → wic
6	SuperNI	task1572 → task363 → task1290 → task181 → task002 → task1510 → task639 → task1729 → task073 → task1590 → task748 → task511 → task591 → task1687 → task875
7		task748 → task073 → task1590 → task639 → task1572 → task1687 → task591 → task363 → task1510 → task1729 → task181 → task511 → task002 → task1290 → task875

Table 12: Seven different orders of task sequences used for continual learning experiments. Orders 1-3 correspond to the standard CL becnhmark adopted by prior works. Orders 4-5 are long-sequence orders spanning 15 tasks, and orders 6-7 are superni spanning 15 tasks following (Razdaibiedina et al., 2023).