TOWARDS TRANSFERABLE PARAMETER UPDATES FOR CONTINUAL LEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) for large language models (LLMs) aims to enable sequential knowledge acquisition while mitigating catastrophic forgetting. However, most existing LLM CL methods primarily regulate parameter modifications, such as determining which parameters should be updated or preserved, while rarely exploring how effective task-specific update behaviors can be learned and generated. This limitation hinders their ability to facilitate knowledge transfer across tasks. To address this challenge, we rethink parameter updates in CL from the perspective of learning adaptive update strategies. Specifically, we introduce Meta-HA, a meta-learning framework that learns transferable update strategies through Hyper-Adapter modules. Hyper-Adapter module consists of a meta-learner and a controller, where the meta-learner captures transferable update priors across tasks through outer-loop optimization, and the controller instantiates these priors to generate task-specific updates and consolidate new knowledge into memory during inner-loop adaptation. Extensive experiments on multiple CL benchmarks demonstrate that our method consistently outperforms state-of-the-art baselines in both average performance and backward transfer.

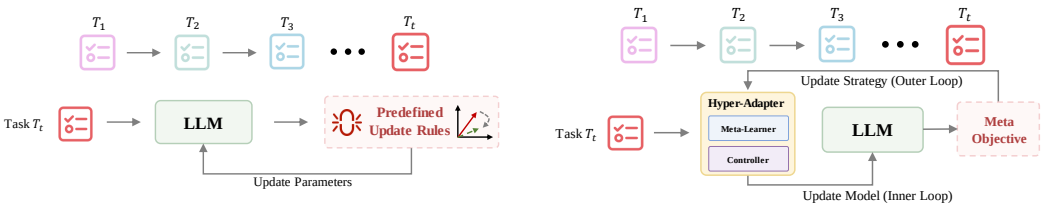
1 INTRODUCTION

Large language models (LLMs) have revolutionized natural language processing with their remarkable capabilities across diverse tasks (Huang et al., 2025; Zhu et al., 2024; Kiciman et al., 2024). However, these models are typically trained in a static, offline manner on fixed datasets, which limits their ability to adapt to new domains, tasks, or knowledge over time (Yu et al., 2026). This ability is particularly critical for LLMs (McCloskey & Cohen, 1989), which are increasingly deployed in dynamic and evolving environments where retraining from scratch is computationally prohibitive and often infeasible (Wang et al., 2024). Continual learning (CL) addresses this fundamental limitation by enabling models to acquire knowledge sequentially from a stream of tasks while retaining previously acquired knowledge. Despite these advances, effective CL for LLMs remains challenging due to the inherent trade-off between stability and plasticity (Elsayed & Mahmood, 2024).

Existing approaches mainly address this challenge by controlling how model parameters are modified during sequential adaptation. Replay-based methods revisit past samples or rely on pseudo-data to preserve previously acquired knowledge (He et al., 2024; Feng et al., 2026). Regularization-based methods restrict parameter updates to reduce interference between tasks (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Hu et al., 2020). Parameter isolation methods assign separate modules to different tasks to avoid conflicts (Zhao et al., 2024; Cheng et al., 2025). In practice, many works combine these strategies with parameter-efficient fine-tuning (PEFT) techniques, such as LoRA (Hu et al., 2021), to improve efficiency while maintaining performance (Bhat et al., 2025; Wang et al., 2023). From a unified perspective, these methods regulate parameter updates through past information or architectural designs, balancing new knowledge acquisition and prior knowledge retention.

Although these approaches have achieved promising results, mainstream LLM CL methods primarily regulate parameter updates through replay, optimization constraints, or architectural designs, while learned task-adaptive update generation remains relatively underexplored. Meanwhile, prior studies in meta-learning and hypernetwork-based adaptation have demonstrated the potential of

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



(a) Typical LLM continual learning.

(b) Meta-HA learns transferable update strategies.

Figure 1: Comparison between conventional continual learning and our method. Typical LLM continual learning methods directly optimize task-specific parameters under predefined objectives or constraints, whereas Meta-HA learns transferable update strategies through Hyper-Adapters.

learning adaptation mechanisms in other settings (Phang et al., 2023; Javed & White, 2019). This motivates a more specific question for LLM continual learning: **Can a model learn transferable update strategies that determine how new knowledge should be integrated when encountering new tasks, and use them to guide task-adaptive parameter updates during continual learning?** Addressing this question requires explicitly modeling how transferable adaptation behaviors can be learned across tasks. This suggests learning transferable update priors that guide how new knowledge is incorporated while accounting for previously acquired knowledge. This naturally motivates the use of meta-learning, which aims to capture reusable knowledge about the learning process across tasks.

To this end, we propose Meta-HA, a meta-learning framework that learns transferable update strategies for continual learning of LLMs. Building on this perspective, Meta-HA introduces a Hyper-Adapter to realize transferable update strategies through meta-learned update priors and task-adaptive update generation, as illustrated in Figure 1. Specifically, Meta-HA consists of a meta-learner, controllers, and task memories. The meta-learner is optimized through an outer-loop objective over CL episodes to capture transferable update priors across tasks. During inner-loop adaptation, controllers instantiate these priors to generate task-adaptive updates, which are subsequently consolidated into task memories. Through this bi-level optimization process, Meta-HA enables models to acquire new knowledge while preserving previously learned abilities.

Our contributions are as follows: (1) We formulate LLM CL from the perspective of learning transferable update strategies, complementing conventional approaches that primarily regulate task-specific parameter updates. (2) We design a Hyper-Adapter module that can be integrated with the PEFT framework, where the meta-learner captures transferable update priors and controllers instantiate these priors to generate task-adaptive parameter updates for continual knowledge consolidation. (3) We conduct extensive experiments across multiple CL benchmarks, demonstrating consistent improvements in both average performance and backward transfer over existing approaches.

2 Related Work

Continual Learning for LLMs: Continual Learning (CL) aims to enable models to acquire knowledge from sequential tasks while preserving previously learned capabilities (Hadsell et al., 2020). Existing CL methods can be broadly categorized into three paradigms. **Replay-based methods** preserve historical knowledge by revisiting previous samples or constructing auxiliary training signals. Recent studies improve replay efficiency through adaptive memory management and selective sample retrieval mechanisms (Feng et al., 2026; He et al., 2024). **Regularization-based methods** reduce task interference by constraining parameter changes according to the importance of previously learned knowledge. Representative approaches, such as EWC (Kirkpatrick et al., 2017) and SI (Zenke et al., 2017), estimate parameter importance through Fisher information or optimization dynamics, while recent works further introduce orthogonality constraints or skill localization mechanisms (Wang et al., 2023; Feng et al., 2024). **Parameter isolation methods** mitigate forgetting by allocating separate parameter spaces or task-specific modules for different tasks (Wang et al., 2025). Recent studies further exploit parameter-efficient fine-tuning techniques, such as LoRA adapters (Hu et al., 2021), to enable scalable continual adaptation of large language models (Cheng et al., 2025; Zhao et al., 2024). Despite their effectiveness, these approaches mainly focus on regulating

parameter updates through data replay, optimization constraints, or architectural designs. They determine where and how much parameters should be modified, while the update behaviors themselves are typically governed by predefined adaptation procedures. In contrast, our work investigates how transferable update strategies can be learned to guide continual adaptation and facilitate knowledge transfer across tasks.

Meta-Learning and Learned Update Mechanisms: Meta-learning aims to learn transferable knowledge about the learning process across tasks, enabling efficient adaptation to new tasks (Finn et al., 2017; Nichol et al., 2018). In continual learning, prior methods such as OML, ANML, and La-MAML meta-learn representations, modulation mechanisms, or optimization dynamics to improve sequential adaptation (Javed & White, 2019; Beaulieu et al., 2020; Gupta et al., 2020). Related studies have also explored learned parameter updates in other settings: MEND learns gradient transformations for model editing, while HyperTuning and HINT use hypernetworks to generate task-conditioned parameter-efficient modules (Mitchell et al., 2022; Phang et al., 2023; Ivison et al., 2023). These works establish the broader paradigm of learned adaptation, but focus on different forms or settings of adaptation. Meta-HA instead focuses on replay-free continual adaptation of LLMs, where transferable update priors are meta-learned over simulated CL episodes and instantiated by controllers as incremental updates for continual knowledge consolidation.

3 Problem Formulation and Setup

We define the CL task as learning a sequence of K tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$. Each task T_t is associated with a dataset $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$. In timestep t, the goal of CL is to optimize a model $\mathcal{F}$ to learn task-specific knowledge from only accessible $\mathcal{D}_t$. The objective is expressed as:

$$\max_{f_\Theta} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x^t, y^t)\sim\mathcal{D}_t} \log p(y^t|x^t, \mathcal{F}) \tag{1}$$

In general, a fundamental challenge of CL is how to update model parameters when acquiring new knowledge while preserving previously learned capabilities. Most LLM CL methods commonly rely on manually designed mechanisms to regulate parameter updates. In this paper, we investigate whether a model can learn a generalizable update strategy that determines how its parameters adapt to new tasks. Once learned, the update mechanism can be applied to sequential tasks without requiring access to historical samples or task-specific memories during downstream continual adaptation, enabling replay-free adaptation.

4 Methodology

4.1 Overview of the Framework

To address the aforementioned challenges, we propose Meta-HA, a meta-learning framework for continual adaptation of LLMs that equips them with Hyper-Adapter (HA) modules. Meta-HA explicitly models transferable update strategies that determine how task-specific knowledge should be incorporated during continual adaptation. By separating the acquisition of transferable update priors from task-specific knowledge consolidation, Meta-HA enables more effective knowledge transfer across tasks.

As illustrated in Figure 2, Meta-HA adopts a bi-level optimization framework consisting of two stages: (1) **Inner Optimization (Task-Specific Adaptation):** For each task, controllers adapt the transferable update priors provided by the frozen meta-learners to generate task-adaptive parameter updates according to the current task context. The acquired knowledge is subsequently consolidated into the shared task memory. This process is performed sequentially across tasks within each **Inner Loop**. (2) **Outer Optimization (Meta-Learning):** Beyond optimizing individual tasks, Meta-HA further optimizes the meta-learners through a meta-objective defined on held-out validation data within simulated CL episodes. This allows the meta-learners to capture transferable update priors that can be effectively adapted by controllers across different tasks.

Formally, we construct a series of simulated CL episodes, $\mathcal{E} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$, where each episode consists of a sequence of tasks arriving in a predefined order. The inner loop performs task-specific

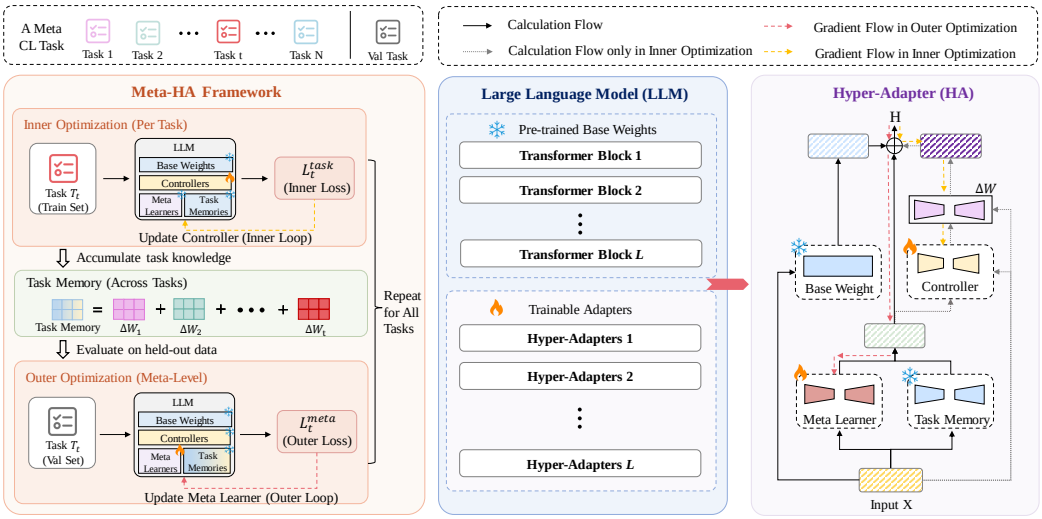


Figure 2: The overall framework of Meta-HA.

adaptation within each episode, while the outer loop optimizes the meta-learners across episodes to improve their ability to capture transferable update priors for continual adaptation.

4.2 Hyper-Adapter Architecture

We introduce a HA module to enable dynamic and transferable task adaptation in CL. Formally, an HA module is defined as $\mathcal{H}(\mathbf{W}_0, f_\theta, f_\phi, g_\psi)$, where $\mathbf{W}_0 \in \mathbb{R}^{d_{in} \times d_{out}}$ denotes the frozen base weights of the LLM, f_θ denotes the meta-learner that captures transferable update priors, g_ψ denotes the controller that adapts these priors to generate task-specific updates, and f_ϕ denotes the task memory that stores consolidated knowledge.

During the entire training process, the base weights $\mathbf{W}_0$ remain frozen. Moreover, the task memory f_ϕ is not optimized through standard gradient descent. Instead, it is updated through explicit write operations generated by the controller. This design separates knowledge storage from sequential parameter optimization, allowing the controller to selectively consolidate task-specific knowledge into memory.

Meta-Learner and Task Memory. The meta-learner aims to capture transferable update priors across tasks, which characterize effective adaptation behaviors for CL. During task adaptation, these learned priors are instantiated by the controller to generate task-specific updates. In contrast, the task memory serves as a persistent knowledge repository that incrementally accumulates task-specific information throughout the CL process. In practice, we use the same architectural LoRA adapter to parameterize $f_\theta = \{\mathbf{A}_m, \mathbf{B}_m\}$ and $f_\phi = \{\mathbf{A}_t, \mathbf{B}_t\}$ for reducing computational overhead and system complexity. Given an input-label pair (x, y), we represent the combined hidden state as $\mathbf{h} \in \mathbb{R}^{d_{in} \times seq}$, f_θ and f_ϕ are combined to form a joint representation:

$$\mathbf{h}' = f_{\theta,\phi}(\mathbf{h}) = \frac{\alpha}{r}(\mathbf{B}_m + \mathbf{B}_t)(\mathbf{A}_m + \mathbf{A}_t)\mathbf{h} \tag{2}$$
$$\mathbf{B}_m, \mathbf{B}_t \in \mathbb{R}^{d_{out} \times r}, \mathbf{A}_m, \mathbf{A}_t \in \mathbb{R}^{r \times d_{in}}, \mathbf{h}' \in \mathbb{R}^{d_{out} \times seq}$$

where r and α represent the LoRA rank and scaling factor, respectively.

Controller. The controller is designed to determine *what knowledge should be written into the task memory* conditioned on $\mathbf{h}$ and $\mathbf{h}'$. Similarly, the controller is parameterized as $g_\psi = \{\mathbf{A}_c, \mathbf{B}_c\}$. As a result, conditioned on the hidden states from the task memory and meta-learner, the controller

generates a task-adaptive update:

$$\Delta \mathbf{W} = g_\psi(\mathbf{h}, \mathbf{h}') = \frac{\alpha_c}{r_c} \mathbf{B}_c \mathbf{A}_c (\mathbf{h} \oplus \mathbf{h}') \tag{3}$$
$$\mathbf{A}_c \in \mathbb{R}^{r_c \times (d_{in}+d_{out})}, \mathbf{B}_c \in \mathbb{R}^{r(d_{in}+d_{out}) \times r_c}$$

where $\oplus$ represents the row-wise concatenation operation between $\mathbf{h}$ and $\mathbf{h}'$, r_c and α_c are the LoRA rank and scaling factor of g_ψ, respectively.

In particular, we employ a binary mask $\mathbf{m} = [m_1, m_2, ..., m_{seq}]^\top \in \{0,1\}^{seq}$ to identify tokens such that only the positions corresponding to target outputs contribute to adaptive generation. This design emulates the paradigm of gradient-based LLM training, where optimization signals are derived solely from target tokens:

$$\Delta \hat{\mathbf{W}} = \frac{1}{|\mathbf{m}|} \sum_{i=1}^{seq} \mathbf{m}_i \Delta \mathbf{W}_{:,i} \tag{4}$$

where $\Delta \mathbf{W} \in \mathbb{R}^{r(d_{in}+d_{out}) \times seq}$ and $\Delta \hat{\mathbf{W}} \in \mathbb{R}^{r(d_{in}+d_{out})}$.

The generated $\Delta \hat{\mathbf{W}}$ can be written into the task memory. Specifically, we reshape $\Delta \hat{\mathbf{W}}$ into two matrices $\Delta \mathbf{A} \in \mathbb{R}^{r \times d_{in}}$ and $\Delta \mathbf{B}^\top \in \mathbb{R}^{r \times d_{out}}$, followed by a merging operation:

$$f_{\phi'} = \{\mathbf{A}_t + \Delta \mathbf{A}, \mathbf{B}_t + \Delta \mathbf{B}\} \tag{5}$$

Finally, the forward computation of HA is expressed as:

$$\hat{\mathbf{h}} = \mathbf{W}_0 \mathbf{h} + f_{\theta,\phi'}(\mathbf{h}) \tag{6}$$

4.3 Bi-level Meta Optimization

Although the HA provides a mechanism for generating task-adaptive updates, optimizing only current-task objectives still leads to short-sighted adaptation behaviors that fail to generalize across sequential tasks. To address this issue, Meta-HA adopts a bi-level meta-learning optimization framework that jointly optimizes task-specific adaptation and transferable update strategies.

For a CL episode $\mathcal{T}_k$, each task T_t is equipped with a meta-train set $\mathcal{D}_t^{tr}$ for inner-loop adaptation and a meta-validation set $\mathcal{D}_t^{val}$ for outer-loop optimization.

Inner Optimization. The goal of inner optimization is to adapt the transferable update priors captured by the meta-learner to the current task. During this phase, the meta-learner remains frozen, and only the controller parameters ψ are optimized. Specifically, given the current task T_t, the controller leverages the learned update priors and generates task-specific updates:

$$g_{\psi^*} = \arg\min_{g_\psi} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_t^{tr}} - \log(y|x, \mathcal{F}_\Theta) \tag{7}$$

where $\mathcal{F}_\Theta = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_L\}$ and L is the number of HA modules. After optimization, the learned controller g_{ψ^*} yields the final adaptive update for f_ϕ via Eq. 3, 4 and 5. Consequently, we obtain the task-adapted model $\mathcal{F}_{\Theta'}$.

Outer Optimization. After task adaptation, the outer optimization optimizes the meta-learner based on whether the update behaviors induced by the Hyper-Adapter can improve current-task performance while promoting continual transfer across tasks. To this end, we construct $\mathcal{D}_{\text{val}}^t$ using held-out validation sets from all tasks within the current CL episode. During meta-evaluation, the controller is not involved in prediction, since its role is to generate task-specific memory updates during adaptation. Instead, $\mathcal{F}_{\Theta'}$ makes predictions based on the consolidated memory after adaptation, as defined in Eq. 6. The meta-loss is expressed as:

$$\mathcal{L}_t^{meta}(\mathcal{F}_{\Theta'}) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_t^{val}} - \log(y|x, \mathcal{F}_{\Theta'}) \tag{8}$$

To promote stability and balanced transfer, we additionally introduce a relative loss that captures performance changes before and after learning the current task:

$$\mathcal{L}_t^{rel} = \max(\mathcal{L}_t^{meta}(\mathcal{F}_{\Theta'}) - \mathcal{L}_t^{meta}(\mathcal{F}_\Theta), 0) \tag{9}$$

The final outer objective is expressed as:

$$\mathcal{L}_t^{outer} = \mathcal{L}_t^{meta} + \lambda \mathcal{L}_t^{rel} \tag{10}$$

where λ balances absolute performance and relative continual stability.

The meta-learner is then updated via gradients at the end of $\mathcal{T}_k$:

$$f_\theta \leftarrow f_\theta - \frac{\eta_{out}}{|\mathcal{T}_k|} \sum_{t=1}^{|\mathcal{T}_k|} \nabla_{f_\theta} \mathcal{L}_t^{outer} \tag{11}$$

where η_{out} is the learning rate.

The overall process is repeated across multiple simulated CL episodes in the outer loop, allowing the meta-learner to capture transferable update priors that can be adapted by controllers for CL. The complete bi-level meta-learning optimization procedure of Meta-HA is summarized in Algorithm 1. Once this meta-training stage is completed, the meta-learners remain frozen throughout downstream CL evaluation, and only the controllers are optimized on the target task streams.

5 Experiments

5.1 Experimental Setup

Datasets. Following previous CL studies on LLMs (Du et al., 2024), we evaluate Meta-HA on three widely adopted CL benchmarks: The **Standard CL Benchmark (Std. CL)** consists of multiple text classification tasks (Zhang et al., 2015). The **SuperNI Benchmark (SuperNI)** (Wang et al., 2022) contains diverse natural language understanding and generation tasks, including dialogue generation, information extraction, question answering, summarization, and sentiment analysis. Following prior CL settings (Zhao et al., 2024), we select three tasks from each category, resulting in 15 sequential tasks. For each task, we randomly sample 1,000 training instances and use additional 100 instances as validation and testing sets. The **Long Sequence Benchmark (Long Seq.)** (Razdaibiedina et al., 2023) contains 15 long-context classification tasks designed to evaluate continuous adaptation under extended input sequences. Following previous settings (Feng et al., 2026), we use 1,000 training samples per task and reserve 500 samples held-out as validation and testing sets.

Evaluation Metrics. Following standard CL evaluation protocols, we denote the testing performance on task T_j after completing training on task T_i as $a_{i,j}$. We use Accuracy for classification tasks and Rouge-L (Lin, 2004) for other tasks. Based on the task-wise performance matrix, we report widely adopted continual learning metrics: **Average Performance (AP)** measures the overall performance after the model finishes learning the entire task stream, defined as $\mathrm{AP} = \frac{1}{K}\sum_{j=1}^{K} a_{K,j}$, where K denotes the total number of tasks. **Backward Transfer (BWT)** measures whether learning new tasks improves the performance of previously learned tasks, defined as $\mathrm{BWT} = \frac{1}{K-1}\sum_{j=1}^{K-1}(a_{K,j} - a_{j,j})$.

Baselines. We compare Meta-HA with a comprehensive set of representative continual learning baselines for LLMs, including replay-based methods, parameter-isolation approaches, regularization-based and meta-learning-based methods. For fairness, all methods are implemented under the same LoRA-based continual learning framework and trained using the same task stream and optimization settings. The evaluated baselines include SeqLoRA, MixReplay, Fixed-interval Replay, ANML (Beaulieu et al., 2020), O-LoRA (Wang et al., 2023), MoELoRA (Luo et al., 2024), EWC (Kirkpatrick et al., 2017), SAPT (Zhao et al., 2024), MIGU (Du et al., 2024), SSR (Huang et al., 2024), Recurrent-KIF (Feng et al., 2025b), AIMMerging (Feng et al., 2025a), VBM (Kang et al., 2025) and FOREVER (Feng et al., 2026). We additionally report the performance of multi-task joint training (MTL Upper Bound), which serves as the non-continual upper performance bound. More details are provided in Appendix B.

Implementation Details. To perform meta-training, we construct a diverse meta-task pool using Super-NaturalInstructions (Wang et al., 2022), where all tasks overlapping with the above CL benchmarks are removed to avoid information leakage. For each CL episode, we randomly sample a batch of tasks from the pool to construct a task sequence for bi-level meta optimization. During the outer optimization stage, we adopt the first-order MAML (FOMAML) algorithm (Nichol et al., 2018) to

Table 1: The overall results on the three CL benchmarks using the Qwen3-0.6B model. **Information Separation (IS)** indicates whether a method explicitly accesses or reuses information from previous tasks during training. **Fixed Parameters (FP)** denotes whether the number of trainable parameters remains constant as the number of tasks increases. We report the average performance across different task orders after training on the final task of each benchmark. Methods marked with $*$ are reproduced from previous work. The last row presents the upper bound under the multi-task learning (MTL) setting.

Method	IS	FP	Std. CL		Long Seq.		SuperNI	
			AP↑	BWT↑	AP↑	BWT↑	AP↑	BWT↑
SeqLoRA	✓	✓	47.2	-12.6	36.0	-17.5	8.2	-27.4
MoELoRA$*$ (Luo et al., 2024)	✓	✓	55.3	-8.2	35.3	-13.6	25.7	-11.3
EWC$*$ (Kirkpatrick et al., 2017)	✓	✓	51.0	-10.3	44.8	-13.8	32.9	-18.6
O-LoRA$*$ (Wang et al., 2023)	✓	✗	59.4	-7.9	54.1	-12.4	23.7	-17.5
MixReplay	✗	✓	65.8	-8.0	65.1	-11.4	34.6	-14.1
Fixed-interval Replay	✗	✓	65.1	-9.2	64.5	-10.9	34.7	-14.5
SAPT$*$ (Zhao et al., 2024)	✗	✗	68.8	-6.9	67.2	-8.8	38.5	-6.2
MIGU$*$ (Du et al., 2024)	✓	✓	69.9	-7.5	66.3	-9.2	35.0	-8.1
SSR$*$ (Huang et al., 2024)	✗	✓	68.4	-7.1	67.5	-9.0	40.1	-5.4
Recurrent-KIF$*$ (Feng et al., 2025b)	✗	✓	70.6	-6.5	67.7	-8.4	39.7	-5.0
AIMMerging$*$ (Feng et al., 2025a)	✗	✓	71.9	-5.0	67.9	-6.3	41.0	-3.4
VBM$*$ (Kang et al., 2025)	✗	✓	71.5	-5.2	68.1	-6.1	41.3	-3.7
FOREVER$*$ (Feng et al., 2026)	✗	✓	72.9	-4.7	69.4	-5.0	42.1	-2.9
ANML (Beaulieu et al., 2020)	✓	✓	72.1	-4.9	67.1	-6.9	41.8	-5.7
Meta-HA (ours)	✓	✓	**75.3**	**-0.2**	**71.7**	**-0.3**	**47.4**	**-0.8**
MTL	–	–	77.4	–	77.8	–	48.2	–

optimize the meta-learners efficiently, where the outer gradient is computed through the generated memory updates while ignoring the higher-order gradients from the inner-loop controller adaptation. Note that, for a fair comparison, we equip ANML with the same meta-training stage as Meta-HA. After completing meta-training, the meta-learners are frozen, and CL evaluation is conducted by optimizing only the controllers on downstream CL benchmarks. We evaluate Meta-HA across multiple backbone LLMs with different model scales, including Qwen3-0.6B (Yang et al., 2025), LLaMA-3.2-1B (Grattafiori et al., 2024), and LLaMA-2-7B (Touvron et al., 2023). All experiments are conducted on four NVIDIA RTX 4090 GPUs. Additional implementation details are provided in Appendix C.

5.2 Main Results

Does Meta-HA achieve superior CL performance compared with existing CL methods across models of different scales? Table 1 presents the overall performance of Meta-HA and recent baselines on three CL benchmarks. Meta-HA consistently achieves the best performance across all benchmarks. Among meta-learning-based CL approaches, ANML achieves competitive results with conventional CL methods, demonstrating the potential of meta-learning for continual adaptation. In contrast, Meta-HA further learns transferable update strategies for task-specific adaptation. Specifically, Meta-HA achieves 75.3% and 47.4% on Standard CL and SuperNI, respectively, approaching the MTL upper bounds of 77.4% and 48.2%. On the Long benchmark, Meta-HA reaches 71.7%, outperforming the strongest baseline FOREVER by 2.3%. Moreover, Meta-HA achieves consistently better BWT, indicating improved forgetting mitigation during sequential learning. We further evaluate Meta-HA across backbone models with different scales, ranging from 0.6B to 7B parameters, as shown in Figure 3. Meta-HA consistently outperforms all baselines across different model sizes. Its AP improves from 47.4% on Qwen3-0.6B to 58.8% on LLaMA2-7B, demonstrating the scalability of the proposed framework. The detailed experiments under different task orders can be found in the Appendix D.1.

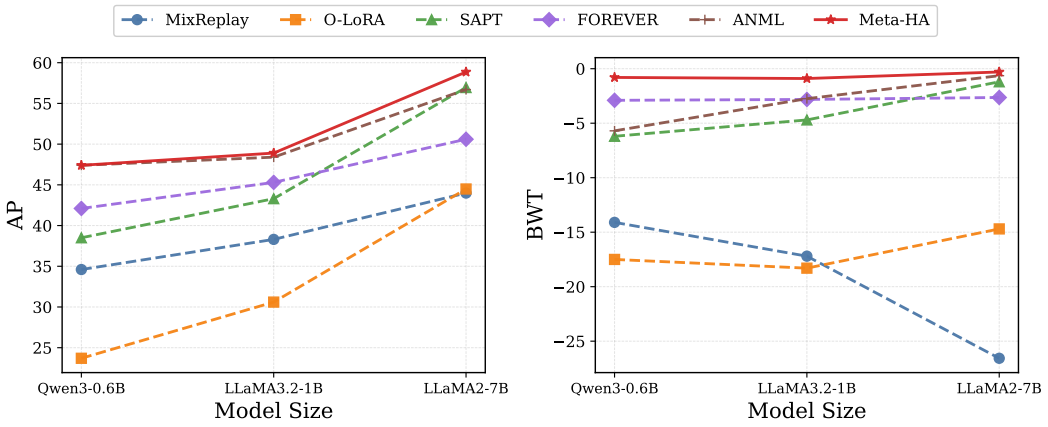


Figure 3: Performance of Meta-HA with different base models on the SuperNI Benchmark.

5.3 Ablation Study

Q2: How effective are the proposed adaptive update mechanism and relative loss for CL performance? We conduct ablation studies to investigate the effectiveness of the controller-based update mechanism and the relative loss L_{rel}. The results are reported in Table 2. Unless otherwise specified, all subsequent experiments are conducted on LLaMA-3.2-1B. We first replace the controller with conventional gradient-based optimization, where the meta-learner directly learns a shared initialization across the CL sequence. Removing the controller leads to consistent performance drops, reducing AP from 69.5% to 66.7% on Long Sequence and from 49.8% to 47.1% on SuperNI. This demonstrates that a fixed meta-initialization is insufficient to capture diverse adaptation dynamics across CL tasks. In contrast, Meta-HA delegates task-specific learning to the controller and performs only one final update to the task memory, substantially reducing optimization complexity over long CL streams. We then remove the relative loss L_{rel} during outer optimization. The results show a consistent degradation in BWT, with the most pronounced drop on SuperNI, where BWT decreases from -0.9% to -2.7%. It indicates that L_{rel} provides additional guidance for stable cross-task adaptation by considering relative performance changes during inner optimization. Moreover, due to the substantial diversity of tasks in CL settings, optimizing only the meta-objective L_{meta} encourages the meta-learner to over-focus on tasks with larger optimization losses, which further amplifies meta-training instability.

5.4 Further Analysis

Q3: Are the performance gains mainly attributed to meta-training rather than multi-task pretraining? We investigate the effectiveness of different meta-learner training strategies, including random initialization and multi-task pretraining. For the MTL setting, the meta-learner is first trained on sampled task batches using standard multi-task optimization before CL. The results are reported in Table 3. We observe that random initialization yields unstable continual adaptation behavior, leading to substantially negative BWT scores on both benchmarks. Surprisingly, MTL pretraining performs even worse, reducing AP from 69.5% to 43.3% on Long Sequence and from 49.8% to 23.3% on SuperNI. This result demonstrates that replacing meta-training with conventional multi-task learning leads to substantial degradation. When heterogeneous tasks are jointly optimized through the same fixed-rank LoRA modules, their task-specific signals compete for limited shared capacity, inducing cross-task interference (Khan, 2026). This can overly constrain the low-rank space and reduce its plasticity for acquiring new knowledge during subsequent CL tests.

Q4: How do the λ and controller rank influence the performance of Meta-HA? We analyze the effect of the balance coefficient λ, and the results are reported in Figure 5. Overall, increasing λ consistently improves BWT on both benchmarks, indicating that stronger relative optimization encourages the meta-learner to preserve previously acquired knowledge more effectively during continual adaptation. However, this improvement in stability is accompanied by a trade-off in plasticity. On Long Sequence, AP gradually decreases from 70.5% to

Table 2: Single component ablations of Meta-HA on the SuperNI and Long Sequence benchmarks.

Method	Long Seq.		SuperNI	
	AP↑	BWT↑	AP↑	BWT↑
Meta-HA	**69.5**	**-0.7**	**49.8**	**-0.9**
w/o controller	66.7	-0.3	47.1	-1.3
w/o $\mathcal{L}_{rel}$	70.5	-1.3	46.0	-2.7

Table 3: Analysis of the meta-learner with different initialization strategies on the SuperNI and Long Sequence benchmarks.

Method	Long Seq.		SuperNI	
	AP↑	BWT↑	AP↑	BWT↑
Meta-HA	**69.5**	**-0.7**	**49.8**	**-0.9**
w/ multi-task	43.3	-3.1	23.3	-9.7
w/ random	55.7	-6.4	36.7	-10.5

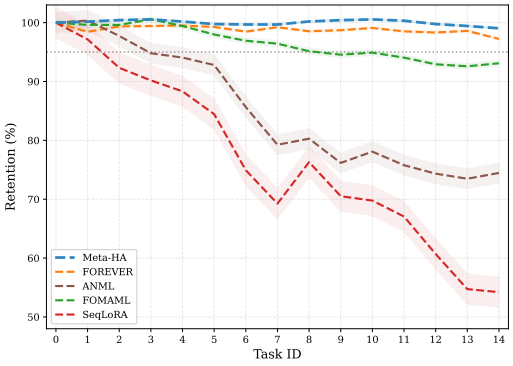


Figure 4: Retention dynamics of Task 1 across the CL sequence on the SuperNI benchmark.

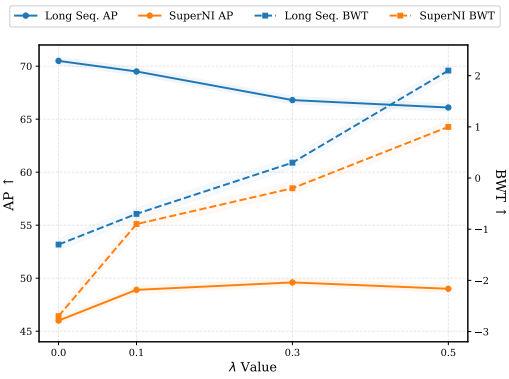


Figure 5: Comparison of different balance coefficient λ.

66.1% as λ increases from 0.0 to 0.5, suggesting that overly strong regularization can restrict the model's ability to adapt to new tasks. We further investigate how different controller ranks $r_c \in \{4, 8, 16\}$ affect the learning of task-specific update strategies. As shown in Table 4, increasing the controller rank does not consistently improve CL performance, while leading to worse backward transfer, especially under $r_c = 16$, suggesting that overly expressive controllers may encourage excessively task-specific updates and amplify inter-task interference. In contrast, a moderate rank ($r_c = 8$) achieves a better balance between plasticity and stability.

Backward Transfer Dynamics during Continual Learning. Q5: How does Meta-HA preserve previously learned knowledge during the CL process? We visualize the retention dynamics of the initial task (Task 1) across all subsequent tasks in Figure 4 as a representative case study. As shown in the figure, Meta-HA consistently maintains the highest retention rate for Task 1 and exhibits less forgetting compared with other baselines.

Table 4: Analysis of controller rank on the Qwen3-0.6B.

Rank	Long Seq.		SuperNI	
	AP↑	BWT↑	AP↑	BWT↑
$r_c = 4$	70.3	-0.7	46.1	-1.4
$r_c = 8$	71.7	-0.3	47.4	-0.8
$r_c = 16$	70.6	-1.5	47.8	-5.5

6 Conclusion

In this paper, we proposed Meta-HA, a meta-learning framework for continual learning of large language models that formulates adaptation from the perspective of learning transferable update strategies. By introducing a Hyper-Adapter consisting of a meta-learner, task memory, and controllers, Meta-HA enables adaptive task-specific updates and continual knowledge consolidation without accessing previous task samples. Extensive experiments across three CL benchmarks and different models demonstrate that Meta-HA achieves strong average performance and substantially improves backward transfer. These results highlight the potential of learning transferable update strategies for continual adaptation in large language models.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation of parts of the manuscript. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to edit the research paper to improve readability, including grammar correction, wording refinement, and sentence-level polishing. We have reviewed all AI-assisted work. All AI-assisted translations and edits were checked by the authors to ensure that they accurately reflected the intended technical meaning and did not introduce inaccurate or unsupported statements. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn, 2020. URL https://arxiv.org/abs/2002.09571.

Prashant Shivaram Bhat, Shakib Yazdani, Elahe Arani, and Bahram Zonooz. Parameter efficient continual learning with dynamic low-rank adaptation, 2025. URL https://arxiv.org/abs/2505.11998.

Xin Cheng, Jiabo Ye, Haiyang Xu, Ming Yan, Ji Zhang, Feng Liu, Fei Huang, and Lei Feng. Exploiting presentative feature distributions for parameter-efficient continual learning of large language models. In *Forty-second International Conference on Machine Learning*, pp. 10159–10181, Vancouver, BC, Canada, 2025. PMLR / OpenReview.net.

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

Mohamed Elsayed and A Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024. OpenReview.net.

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 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1266–1279, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Yujie Feng, Jian Li, Xiaoyu Dong, Pengfei Xu, Xiaohui Zhou, Yujia Zhang, Zexin Lu, Yasha Wang, Alan Zhao, Xu Chu, et al. Aimmerging: Adaptive iterative model merging using training trajectories for language model continual learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13431–13448, Suzhou, China, 2025a. The Association for Computer Linguistics.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S Yu, Xu Chu, and Xiao-Ming Wu. Recurrent knowledge identification and fusion for language model continual learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27396–27413, Vienna, Austria, 2025b. The Association for Computer Linguistics.

Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S Yu, and Xiao-Ming Wu. Forever: Forgetting curve-inspired memory replay for language model continual learning, 2026. URL https://arxiv.org/abs/2601.03938.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135, Sydney, NSW, Australia, 2017. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning, 2020. URL https://arxiv.org/abs/2007.13904.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266, Miami, FL, USA, 2024. Association for Computational Linguistics.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Guannan Hu, Wu Zhang, Hu Ding, and Wenhao Zhu. Gradient episodic memory with a soft constraint for continual learning, 2020. URL https://arxiv.org/abs/2011.07801.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, Bangkok, Thailand, 2024. The Association for Computer Linguistics.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models, 2025. URL https://arxiv.org/abs/2503.06749.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew E Peters. Hint: hypernetwork instruction tuning for efficient zero-and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11272–11288, Toronto, Canada, 2023. Association for Computational Linguistics.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32:1818–1828, 2019.

Hankyul Kang, Gregor Seifer, Donghyun Lee, and Jongbin Ryu. Do your best and get enough rest for continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10077–10086, Nashville, TN, USA, 2025. Computer Vision Foundation / IEEE.

Asif Khan. Capacity and redundancy trade-offs in multi-task learning. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, pp. 2934–2957. PMLR, 2026.

Emre Kiciman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Transactions on Machine Learning Research*, 2024, 2024.

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

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models, 2024. URL https://arxiv.org/abs/2402.12851.

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

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

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, Dublin, Ireland, 2022. The Association for Computer Linguistics.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, Virtual Event, 2022. OpenReview.net.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018. URL https://arxiv.org/abs/1803.02999.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*, pp. 27854–27875, Honolulu, Hawaii, 2023. pmlr.

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

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604, Stockholm, Sweden, 2018. PMLR.

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, 2023. URL https://arxiv.org/abs/2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, New Orleans, LA, USA, 2018. OpenReview.net.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32:3261–3275, 2019.

Lingxiang Wang, Hainan Zhang, and Zhiming Zheng. Parameter importance-driven continual learning for foundation models, 2025. URL https://arxiv.org/abs/2511.15375.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024.

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

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 2022 conference on empirical methods in natural language processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

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, 2025. URL https://arxiv.org/abs/2505.09388.

Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S Yu, and Irwin King. Recent advances of multimodal continual learning: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.

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

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. Personality alignment of large language models, 2024. URL https://arxiv.org/abs/2408.11779.

A Details of Dataset

We evaluate Meta-HA on three widely used CL benchmarks: the Standard CL Benchmark, the Long Sequence Benchmark, and the SuperNI Benchmark.

The **Standard CL Benchmark** (Zhang et al., 2015) consists of multiple text classification tasks: AG News, Amazon Reviews, DBpedia, and Yahoo Answers.

The **Long Sequence Benchmark** (Razdaibiedina et al., 2023) provides a more challenging CL setting with 15 tasks. It includes all tasks from the Standard CL Benchmark, four tasks from GLUE (Wang et al., 2018), five tasks from SuperGLUE (Wang et al., 2019), Yelp review dataset and the IMDB movie review dataset (Maas et al., 2011).

The **SuperNI Benchmark** is constructed based on the Natural-Instructions project (Mishra et al., 2022; Wang et al., 2022), a large-scale benchmark designed to evaluate the ability of language models to generalize to unseen tasks through natural language instructions. Following previous works (Zhao et al., 2024), we select 15 tasks from five categories, including dialogue generation, information extraction (IE), question answering (QA), summarization, and sentiment analysis (SA), with three tasks per category.

In addition to the CL evaluation benchmark, we further construct a large-scale **meta task pool** for meta-continual training based on the Natural-Instructions dataset. To avoid potential data leakage, all tasks used in the final CL evaluation are removed from the meta-training pool. Moreover, we perform additional filtering over the original 1,613 NLP tasks. Specifically, we remove multilingual translation tasks and tasks whose Rouge-L scores are consistently close to zero during preliminary experiments. We observe that LLaMA-series models generally perform poorly on these translation-oriented tasks, which may negatively affect the stability and effectiveness of meta-training. After filtering, the final meta task pool contains 1,187 diverse NLP tasks for meta-training.

Details of the 15 tasks used in the SuperNI Benchmark are presented in Table 9, while the task details of the Long Sequence Benchmark are shown in Table 8. We further report eight different task orders in Table 10.

B Details of Baselines

We compare Meta-HA against a diverse set of CL baselines, with particular emphasis on replay-based, regularization-based, parameter-isolation-based and meta-learning-based methods. For fairness, all baselines are implemented under the LoRA framework.

Algorithm 1 Bi-level Meta Optimization for Meta-HA

Require: A meta-task pool $\mathcal{D}_{\mathcal{T}}$, the number of episode N, a frozen LLM model $\mathcal{F}_\Theta$ with multiple trainable HA module $\mathcal{H}(\mathbf{W}_0, f_{\theta_0}, f_{\phi_0}, g_{\psi_0})$, inner epoch E, learning rates $\eta_{\text{in}}, \eta_{\text{out}}$, trade-off coefficient λ.

Ensure: Meta-learner parameters f_{θ^*}

for episode $n = 1$ to N **do**
  Sample a batch of tasks $\mathcal{T}_n \sim \mathcal{D}_{\mathcal{T}}$
  Construct $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$ based on $\mathcal{T}_n$
  Initialize frozen task memory $f_\phi \leftarrow f_{\phi_0}$
  Initialize meta gradient $g \leftarrow 0$
  for each task $T_t \in \mathcal{T}_n$ **do**
Inner optimization (Task Adaptation)
    Initialize controller $g_\psi \leftarrow g_{\psi_0}$
    Enable controller forward path
    for epoch $i = 1$ to E **do**
      $\mathcal{L}_t^{\text{task}} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_t^{tr}} -\log(y|x, \mathcal{F}_\Theta)$
      $g_\psi \leftarrow g_\psi - \eta_{\text{in}} \nabla_\psi \mathcal{L}_t^{\text{task}}$
    end for
    Write task knowledge into f_ϕ by g_ψ
    Obtain task-adapted model $\mathcal{F}_{\Theta'}$
Outer Optimization (Meta-Learning)
    Disable controller forward path
    $\mathcal{L}_t^{meta} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_t^{val}} -\log(y|x, \mathcal{F}_{\Theta'})$
    $\mathcal{L}_t^{rel} = \max[\mathcal{L}_t^{meta}(\mathcal{F}_{\Theta'}) - \mathcal{L}_t^{meta}(\mathcal{F}_\Theta), 0]$
    $\mathcal{L}_t^{outer} = \mathcal{L}_t^{meta} + \lambda \mathcal{L}_t^{rel}$
    $g \leftarrow g + \eta_{out} \nabla_{f_\theta} \mathcal{L}_t^{outer}$
    $\mathcal{F}_\Theta \leftarrow \mathcal{F}_{\Theta'}$
  end for
  Update meta-learner: $f_\theta \leftarrow f_\theta - \frac{1}{|\mathcal{T}_n|} g$
end for
return f_{θ^*}

MixReplay jointly optimizes the current-task data together with replay samples drawn from the memory buffer at every step. **Fixed-interval Replay** performs rehearsal periodically according to predefined training intervals. **EWC** Kirkpatrick et al. (2017) introduces parameter-level regularization to preserve knowledge acquired from previous tasks. **O-LoRA** Wang et al. (2023) constructs task-specific LoRA adapters in mutually orthogonal subspaces to reduce task interference. **MoELoRA** Luo et al. (2024) incorporates a mixture-of-experts mechanism into continual adaptation. **SAPT** Zhao et al. (2024) leverages a shared-attention strategy to coordinate LoRA block selection across different tasks. **MIGU** Du et al. (2024) selectively updates parameters according to gradient importance in order to alleviate catastrophic forgetting. **SSR** Huang et al. (2024) synthesizes rehearsal samples using the LLM itself for replay-based CL. **Recurrent-KIF** Feng et al. (2025b) performs recurrent model merging with dynamically estimated parameter importance scores. **AIM-Merging** Feng et al. (2025a) adopts an adaptive iterative merging strategy guided by optimization trajectories. **VBM** Kang et al. (2025) schedules replay according to training dynamics inspired by the Ebbinghaus forgetting curve. **FOREVER** Feng et al. (2026) introduces a forgetting curve-inspired replay mechanism that schedules rehearsal according to the model's internal optimization dynamics. **ANML** Beaulieu et al. (2020) is a representative meta-learning-based method that introduces a neuromodulatory network to adaptively regulate the activation of the task network, enabling task-specific modulation during CL.

Table 5: Detailed results under different task orders on the Qwen3-0.6B model.

Benchmark	Order	AP	BWT	FWT
Std. CL	1	74.2	0.5	0.3
	2	75.9	-0.6	2.9
	3	75.8	-0.5	2.5
Long Seq.	4	71.7	-0.5	5.2
	5	72.1	-0.4	4.9
	6	71.3	0.0	3.9
SuperNI	7	47.3	-0.6	3.7
	8	47.6	-0.9	2.5

C Implementation Details

We implement Meta-HA based on the Hugging Face Transformers[1] and PEFT[2] libraries, allowing the proposed framework to be seamlessly integrated into arbitrary LLMs supported by the PEFT framework. To improve training efficiency during meta-training, we adopt several engineering tricks: (1) Instead of using the entire training set to optimize the controller, we only sample a subset of the current task data for controller training, while the trained controller subsequently performs knowledge writing into the task memory using the full training set. This benefit stems from the Meta-HA design, which decouples knowledge storage from gradient optimization. (2) We let the task memory and meta-learner share the same parameter space, which not only reduces the overall parameter overhead but also improves the optimization stability of FOMAML during long CL episodes. (3) Although LLaMA-2-7B can be trained on a single NVIDIA RTX 4090 48GB GPU, we use four GPUs in our experiments to improve training efficiency. (4) For parameter optimization, we utilize the Adafactor optimizer (Shazeer & Stern, 2018) to reduce memory costs.

Finally, Algorithm 1 summarizes the bi-level meta-learning optimization procedure of Meta-HA, and the training hyperparameters are listed in Table 11.

D Additional Results

D.1 Complete Results on Meta-HA

To further examine the robustness of Meta-HA under varying task sequences, we provide the detailed results for each task order across different backbone models in Table 5. The results demonstrate that Meta-HA achieves consistently robust performance under different task orders. In addition, Meta-HA consistently achieves positive **Forward Transfer (FWT)** scores across all benchmarks, demonstrating that previously acquired knowledge can effectively facilitate adaptation to future tasks in continual learning. Following prior work, FWT is defined as $\mathrm{FWT} = \frac{1}{K}\sum_{j=1}^{K}(a_{j,j} - a_j^0)$, where a_j^0 denotes the performance of learning task T_j independently from scratch.

D.2 Effect of Label Masking

In Meta-HA, a binary label mask is introduced to eliminate the interference from non-target tokens during memory updates. To investigate its effectiveness, we conduct an ablation study by removing the label mask and evaluating the resulting model on the SuperNI and Long Sequence benchmarks. To isolate the effect of label masking from the learned meta-knowledge, we use Meta-HA without the meta-training phase for this analysis. Specifically, we evaluate two settings: (1) **SFT**, where each task is individually fine-tuned and the average performance across tasks is reported, and (2) **CL**, where tasks are learned sequentially following the CL setting. The SFT setting examines whether la-

[1]Transformers: https://github.com/huggingface/transformers

[2]PEFT: https://github.com/huggingface/peft

Table 6: Ablation study of label masking in Meta-HA on different CL benchmarks.

Benchmark	Method	AP_{SFT} ↑	AP_{CL} ↑	BWT↑
Long Seq.	w mask	68.4	63.8	**-4.8**
	w/o mask	**69.5**	**64.7**	-5.6
SuperNI	w mask	47.5	**39.1**	**-13.4**
	w/o mask	**48.0**	32.8	-20.5

bel masking affects general task learning ability, while the CL setting evaluates its role in mitigating task interference during sequential adaptation.

As shown in Table 6, removing the label mask has little impact on single-task learning, indicating that it does not affect the model's basic adaptation ability. However, label masking consistently improves BWT performance by reducing interference from task-irrelevant tokens during memory updates. The benefit is more pronounced on SuperNI, where tasks exhibit greater diversity, while the smaller improvement on Long Sequence may be attributed to its relatively homogeneous task distribution. These results demonstrate that label masking is important for achieving more robust memory consolidation in CL.

D.3 Effect of Meta Task Pool Construction

We investigate how different meta task pool construction strategies affect the transferable update strategies learned by Meta-HA. Specifically, we categorize the tasks in the Natural-Instruction into seven groups: *others*, *classification*, *summarization*, *generation*, *reasoning*, *QA*, and *commonsense*. Based on these task categories, we construct four types of meta task pools for meta-training: (1) **Default**: The original Natural-Instruction task pool is used, where CL episodes are constructed by uniformly sampling tasks from all categories. (2) **Hybrid**: The default task pool is augmented with category-specific task sampling, enabling Meta-HA to encounter more structured task sequences (e.g., episodes containing multiple classification tasks). (3) **Classification (Cls Only)**: Only tasks from the classification category are used for meta-training. (4) **Generation (Gen Only)**: Only tasks from the generation category are used for meta-training.

For the last two settings, we select classification and generation tasks because both categories contain more than 300 tasks, providing sufficient task diversity for Meta-HA to learn transferable update strategies.

As shown in Table 7, the **Default** meta task pool achieves the best overall performance on both benchmarks, indicating that diverse task compositions enable Meta-HA to learn more transferable update strategies. In contrast, task-specific meta pools show clear distribution-dependent behaviors. For example, **Cls Only** achieves better BWT on Long Sequence, where tasks are predominantly classification-based and thus share a similar distribution with the meta-training pool, but performs worse on SuperNI with more diverse task types. Similarly, **Gen Only** benefits SuperNI while showing limited generalization to Long Sequence. These results suggest that a narrow meta task distribution may lead to specialized update strategies that are less transferable to different CL scenarios. Interestingly, **Hybrid** does not outperform **Default**, possibly because the imbalanced task distribution among categories causes certain task types to be over-sampled during episode simulation, reducing the diversity of learned adaptation strategies. Overall, these findings highlight the importance of a diverse and balanced meta task pool for learning robust update strategies.

Table 7: Performance comparison under different meta task pool constructions.

Pool Type	Long Seq.		SuperNI	
	AP↑	BWT↑	AP↑	BWT↑
Default	**71.7**	-0.3	**47.4**	-0.8
Hybrid	71.2	-1.8	46.1	-1.9
Cls Only	71.5	**0.2**	44.3	-1.7
Gen Only	71.1	-1.4	46.8	**-0.6**

Table 8: The details of 15 classification datasets in the Long Sequence Benchmark

Dataset name	Category	Task	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 and answering	accuracy
BoolQ	SuperGLUE	boolean question and answering	accuracy
MultiRC	SuperGLUE	question and answering	accuracy
IMDB	SuperGLUE	sentiment analysis	accuracy

Table 9: The details of 15 datasets in the SuperNI Benchmark

Dataset name	Task	Metric
task639_multi_woz_user_utterance_generation	dialogue generation	Rouge-L
task1590_diplomacy_text_generation		Rouge-L
task1729_personachat_generate_next		Rouge-L
task181_outcome_extraction	information extraction	Rouge-L
task748_glucose_reverse_cause_event_detection		Rouge-L
task1510_evalution_relation_extraction		Rouge-L
task002_quoref_answer_generation	question answering	Rouge-L
task073_commonsenseqa_answer_generation		Rouge-L
task591_sciq_answer_generation		Rouge-L
task511_reddit.tifu_long_text_summarization	summarization	Rouge-L
task1290_xsum_summarization		Rouge-L
task1572_samsum_summary		Rouge-L
task363_sst2_polarity_classification	sentiment analysis	accuracy
task875_emotion_classification		accuracy
task1687_sentiment140_classification		accuracy

Table 10: Eight different task orders used for continual learning experiments

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

Table 11: Hyperparameters of Meta-HA

Symbol	Description	Value or Range
-	Batch size for inner optimization	16
-	Step for outer optimization	1
N	Number of CL episodes	200
E	Number of adaptation epochs per task	3
η_{out}	Learning rate for outer optimization	5×10^{-4}
η_{in}	Learning rate for inner optimization	5×10^{-5}
K	Number of tasks per CL episode	$[15, 20]$
n_t^{tr}	Subset size of $\mathcal{D}_t^{\text{tr}}$ for controller training	$\{128, 256, 512, 768, 1024\}$
n_t^{val}	Size of $\mathcal{D}_t^{\text{val}}$ for meta-learner training	256
r	LoRA rank for meta-learner and task memory	8
α	LoRA scaling factor for meta-learner and task memory	16
r_c	LoRA rank for controller	8
α_c	LoRA scaling factor for controller	8
λ	Balance coefficient for relative loss	0.1