MIITA: MEMORY-INDUCED INFERENCE-TIME ADAPTATION FOR CONTINUAL LEARNING WITH SMALL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) is essential for small language models (SLMs) to adapt to evolving real-world needs in resource-constrained deployments. However, directly updating their limited parameter space causes catastrophic forgetting. While memory-based methods naturally address this by decoupling knowledge retention from parameters, existing approaches designed for large language models (LLMs) rely on abundant storage and strong in-context reasoning that SLMs lack. To address these challenges, we propose MIITA, a Memory-Induced Inference-Time Adaptation framework for supervised CL under constrained storage. MIITA stores supervised experiences as compact correction-direction prototypes with semantic anchors, and retrieves them at inference time using semantic and uncertainty-based cues. The retrieved directions are applied through gated temporary hidden-state adaptation, enabling non-destructive reuse of past supervision without backbone updates, prompt extensions, or test-time backpropagation. A local theoretical analysis links this design to first-order loss reduction, uncertainty-guided retrieval, and directional coverage for retaining old-stage knowledge. Extensive experiments across diverse supervised CL settings show that MIITA consistently improves final performance and mitigates forgetting under fixed memory budgets.

1 INTRODUCTION

Small language models (SLMs) are increasingly deployed in localized, low-latency, privacy-sensitive, and resource-constrained settings that require continual adaptation (Lu et al., 2024; Van Nguyen et al., 2025). Yet their limited capacity and high representational redundancy make repeated updates to the backbone prone to severe forgetting and unstable updates (Diera et al., 2025; Chen et al., 2026; K et al., 2026). Memory-based continual learning (CL) keeps the backbone stable by storing experiences in auxiliary memory (Shi et al., 2025; Mok et al., 2023; Gutiérrez et al., 2024). However, existing LLM-centric methods transfer poorly to SLMs: their textual examples, summaries, and demonstrations consume storage on surface content and resist consolidation of functionally redundant experiences under fixed budgets (Qiu et al., 2025; Das et al., 2024; Luo et al., 2025). Using these memories also requires in-context reasoning that is limited in smaller models (Wang et al., 2025). As Figure 1 shows, representative methods degrade as the base model shrinks, especially under fixed storage budgets. This exposes a dual bottleneck: retaining large textual memories within tight storage limits and using them effectively during inference.

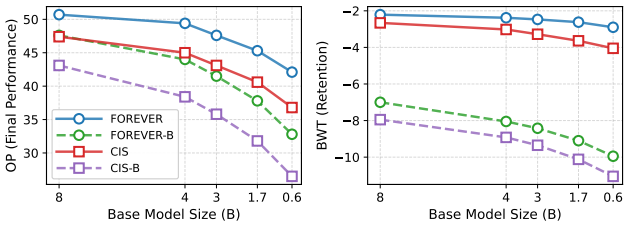


Figure 1: Model-size scaling of CIS (Qiu et al., 2025) and FOREVER (Feng et al., 2026), two LLM-oriented memory-based CL baselines, on `SuperNI` (Wang et al., 2022) benchmark. The "-B" suffix indicates variants restricted to a fixed memory budget, contrasting with their unconstrained counterparts. We evaluate Overall Performance (OP) and Backward Transfer (BWT); higher values indicate better accuracy and less forgetting, respectively.

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.

Overcoming this dual bottleneck requires rethinking what a memory item should preserve. Inspired by Von Oswald et al. (2023), who characterize in-context examples as inducing gradient-descent-like local corrections during the forward pass, the value of a memory item lies not in its raw semantic content alone, but in the functional correction it imposes on the model's hidden representation. This perspective reveals a *divergence between semantic and functional redundancy*: semantically disparate examples may induce identical corrections and can thus be merged, whereas semantically similar examples might require distinct corrections. Consequently, rather than preserving historical text and expecting the SLM to infer how to use it, an alternative paradigm is to organize memory directly around reusable adaptation directions. Storing these functional signals in a compact form addresses both tight storage limits and the reliance on in-context reasoning.

Motivated by this paradigm, we propose **MIITA**, a **Memory-Induced Inference-Time Adaptation** framework for supervised CL with SLMs under constrained persistent storage. MIITA converts each supervised experience into a compact correction direction within the model's internal hidden space, consolidating these functional signals into budget-aware memory prototypes that utilize semantic anchors purely as lightweight retrieval keys. At inference time, the framework retrieves relevant historical directions through both semantic relevance and an uncertainty-based directional proxy. These directions are aggregated into a query-specific correction and applied via a gated, temporary hidden-state update to directly guide the prediction. This adaptation is disposable, requiring no permanent parameter updates, test-time backpropagation, or lengthy prompt extensions. Our main contributions are summarized as follows:

- We propose a direction-oriented memory representation that stores supervised experiences as correction-direction prototypes with lightweight semantic anchors, overcoming the high storage and reasoning costs of traditional textual memory in SLMs.
- We introduce a memory-induced inference-time adaptation mechanism that applies retrieved historical directions via gated temporary hidden-state updates, safely reusing past supervision without test-time backpropagation or destructive backbone updates.
- We provide a local theoretical analysis showing that correction directions are first-order optimal for decreasing the supervised loss, that uncertainty directions identify the steepest direction for decreasing local predictive entropy and serve as label-free retrieval cues, and that a local bound relates old-stage supervised token loss to the directional coverage of retrieved memory.
- We provide comprehensive empirical evaluation across diverse CL settings, showing that MIITA improves final performance and reduces forgetting under fixed memory budgets, with further analyses demonstrating its robustness across model scales and memory budgets, as well as the effectiveness of functional correction memory and uncertainty-guided retrieval.

2 Related Work

CL for SLMs. CL for SLMs is crucial for resource-constrained deployment but remains underexplored. Existing methods address catastrophic forgetting through various mechanisms: DKVB (Diera et al., 2025) uses localized updates via a discrete key-value bottleneck; DA-GRPO (Chen et al., 2026) optimizes continual post-training while learning to invoke a cloud LLM under a cloud-assistance budget; and K et al. (2026) use POS-guided code-switching to mitigate cross-lingual forgetting through replay adapters. However, these approaches still rely on training-time parameter updates, cloud assistance, or sample replay. In contrast, MIITA addresses these limitations by bypassing persistent backbone modifications and training-time replay, thereby overcoming the inherent unreliability of SLMs in continual fine-tuning and text-based prompting.

Memory-Based CL for LLMs. Existing memory-based CL methods fall into two categories based on when memory is used. Training-time replay methods reuse stored examples during optimization; for example, Continual-T0 (CT0) (Scialom et al., 2022) employs a small rehearsal buffer to retain previous abilities, while FOREVER (Feng et al., 2026) schedules memory reviews based on update dynamics. Alternatively, inference-time methods like CIS (Qiu et al., 2025) and InCA (Momeni et al., 2025) utilize textual summaries or in-context examples to avoid forgetting without updating the LLM. Other inference-time methods (e.g., MAC (Tack et al., 2024), CMT (Li et al., 2025), HippoRAG 2 (Gutiérrez et al., 2025), MBC (Katraouras & Rafailidis, 2026)) retrieve external documents for continual knowledge updates. However, these methods focus on knowledge augmentation rather than directly addressing task-level forgetting in supervised CL, which differs from our target problem. Furthermore,

the remaining LLM-centric approaches are ill-suited for SLMs, where limited capacity and weaker in-context abilities make replay and textual memories unreliable. In contrast, MIITA stores supervised experiences as compact functional correction directions, invoking them via lightweight, optimization-free hidden-state adaptation at inference time.

3 METHODOLOGY

3.1 PROBLEM SETTING

We study **supervised continual learning (CL) for small language models (SLMs) under a constrained persistent auxiliary storage budget**. We consider a fixed base SLM throughout each supervised stream; CL methods update system state on top of it, and different base SLMs are evaluated in separate runs. An SLM receives a sequence of data stages $\mathcal{D}_{\text{stream}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, where each stage $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$ contains supervised input-output examples. After each stage, a CL algorithm updates its system state, which may include model parameters and/or auxiliary persistent information such as replay samples, summaries, prompts, adapters, statistics, or memory items. To reflect the resource constraints of SLM deployment, the total size of this auxiliary persistent information is bounded by a fixed storage budget C_{mem}. The goal is to learn from each new stage while preserving performance on all previously seen stages, i.e., maximizing final average performance and minimizing forgetting.

3.2 MIITA: MEMORY-INDUCED INFERENCE-TIME ADAPTATION

To address budget-constrained supervised CL for SLMs, we introduce **MIITA**, a memory-induced inference-time adaptation framework that decouples long-term experience retention from the frozen model parameter space by preserving historical supervision as compact correction directions rather than raw examples or textual summaries. Tailored to SLMs' limited capacity and weaker in-context abilities, MIITA models continual memory as reusable adaptation signals: during CL, supervised experiences are converted into pre-LM-head correction directions, which the memory manager then organizes into budget-aware prototypes with semantic keys for retrieval. At inference, MIITA retrieves relevant prototypes using semantic and uncertainty cues, applying them via a gated temporary hidden update. We provide local theoretical analyses for correction-direction extraction, uncertainty-based retrieval, temporary hidden adaptation, and direction-memory coverage; formal statements and proofs are given in Appendix A. The overall framework of MIITA is shown in Algorithm 1 in Appendix B.2

3.2.1 DIRECTION-ORIENTED MEMORY REPRESENTATION

MIITA maintains a memory bank $\mathcal{B} = \{u_1, \ldots, u_J\}$, where each unit is a direction prototype:

$$u_j = (d_j, \mathcal{A}_j, n_j).$$

This design reflects the need to separate functional retention from semantic access in SLMs, which cannot reliably depend on large textual memories due to tighter context budgets and weaker in-context abilities. The prototype direction $d_j \in \mathbb{R}^H$ is a normalized correction direction in the pre-LM-head hidden space and serves as the primary stored signal, preserving the functional adaptation effect of historical supervised experiences. The anchor set $\mathcal{A}_j$ contains lightweight semantic keys that are used only to retrieve this functional signal from different query contexts, and n_j records how many experiences have been merged into the prototype. The memory bank $\mathcal{B}$ is maintained under the global storage budget C_{mem}, which is enforced during memory writing through memory repair when new directions or anchors exceed the budget, as detailed in Section 3.2.3.

3.2.2 EXTRACTING DIRECTION SIGNAL FROM AN EXPERIENCE

Given a supervised experience (x_m, y_m), MIITA extracts two signals: a semantic key k_m for memory access and a correction direction d_m for memory writing.

Semantic key. Let $\mathcal{X}_m$ be the input token positions and $h_r^{(\ell_k)}$ the hidden state at layer ℓ_k. We define

$$k_m = \text{Norm}\left(\frac{1}{|\mathcal{X}_m|}\sum_{r\in\mathcal{X}_m} h_r^{(\ell_k)}\right). \tag{1}$$

where $\mathrm{Norm}(v) = v/(\|v\|_2 + \epsilon)$. The key k_m is used only for later retrieval.

Correction vector. We define the correction signal in the pre-LM-head hidden space, i.e., the space of final hidden states before they are projected to vocabulary logits. For an experience (x_m, y_m), its unnormalized correction vector R_m is

$$R_m = \frac{1}{|\mathcal{Y}_m|} \sum_{t \in \mathcal{Y}_m} \left(-\frac{\partial \mathcal{L}_t}{\partial h_t^{\mathrm{out}}} \right) = \frac{1}{|\mathcal{Y}_m|} \sum_{t \in \mathcal{Y}_m} W_{\mathrm{LM}}^{\top}(e_{y_t} - p_t). \tag{2}$$

Here, $\mathcal{Y}_m$ denotes the target-output token positions, $h_t^{\mathrm{out}} \in \mathbb{R}^H$ is the final hidden state before the LM head at position t, W_{LM} is the LM-head projection matrix, $p_t = \mathrm{softmax}(W_{\mathrm{LM}} h_t^{\mathrm{out}})$ is the predicted token distribution, y_t is the ground-truth token, e_{y_t} is its one-hot vector, and $\mathcal{L}_t = -\log p_t[y_t]$ is the token-level cross-entropy loss. The vector R_m represents the supervised correction signal exposed by this experience: it specifies the local hidden-space direction that would increase the likelihood of the target output under the current model. Unlike a text summary or a semantic embedding, this signal captures the functional training value of the experience. We therefore store its normalized form $d_m = \mathrm{Norm}(R_m)$ as a reusable adaptation direction in memory. This direction is first-order optimal for locally decreasing the supervised loss of the experience in the pre-LM-head hidden space (Appendix A, Proposition A.1). At this layer, Eq. (2) computes the supervised correction directly from the LM head without backpropagating through the Transformer blocks; computing a corresponding correction at an earlier layer requires backpropagation through the subsequent blocks. A layer-matched ablation in Appendix D.4 also favors the pre-LM-head location.

3.2.3 Budget-Aware Memory Management

Given the semantic key k_m and correction direction d_m extracted from a new experience, MIITA writes it into the memory bank by first updating the direction prototypes and then repairing the memory if the storage budget is exceeded.

Prototype assignment and update. If $|\mathcal{B}| = 0$, MIITA directly creates a new prototype $u_{\mathrm{new}} = (d_m, \{k_m\}, 1)$. Otherwise, it finds the closest existing prototype in the correction-direction space:

$$j^\star = \arg\max_{1 \le j \le J} \cos(d_m, d_j).$$

Let τ_d be the direction-merge threshold. If $\cos(d_m, d_{j^\star}) < \tau_d$, the experience is not sufficiently aligned with any existing prototype, so it is treated as a new direction and stored as $u_{\mathrm{new}} = (d_m, \{k_m\}, 1)$. Otherwise, it is merged into $u_{j^\star}$ by updating the prototype direction with a running average:

$$d_{j^\star} \leftarrow \mathrm{Norm}\left(n_{j^\star} d_{j^\star} + d_m\right), \qquad n_{j^\star} \leftarrow n_{j^\star} + 1. \tag{3}$$

After merging into $u_{j^\star}$, MIITA uses k_m only to expand the semantic access points of the assigned prototype. With semantic-anchor novelty threshold τ_k, we update

$$\mathcal{A}_{j^\star} \leftarrow \mathcal{A}_{j^\star} \cup \{k_m\} \quad \text{if } \max_{a \in \mathcal{A}_{j^\star}} \cos(k_m, a) < \tau_k. \tag{4}$$

Otherwise, k_m is discarded as a redundant anchor. This enables one correction direction to be retrieved from diverse contexts without storing redundant anchors or raw examples.

Memory repair. When the memory footprint exceeds C_{mem}, MIITA repairs memory in a direction-preserving order: it first prunes redundant semantic anchors, then merges the most similar prototypes with minimal distortion, and finally evicts the prototype with the lowest value:

$$\mathrm{value}(u_j) = \frac{\log(1 + n_j)\,(1 - \max_{r \neq j} \cos(d_j, d_r))}{\mathrm{Size}(u_j)}. \tag{5}$$

This value favors prototypes that are well supported, directionally distinctive, and storage-efficient. As formalized by the directional coverage view at the end of Section 3.2.4, the benefit of memory depends on how well stored directions cover historical corrections; thus, the limited budget is allocated to diverse reusable correction directions rather than raw examples.

3.2.4 INFERENCE-TIME RETRIEVAL AND ADAPTATION

After continual training, given a query q at inference time, MIITA first constructs query-side retrieval signals, then retrieves relevant historical correction directions, and finally applies the retrieved direction through temporary hidden-state adaptation to locally improve the current prediction without permanently modifying the model.

Query-side retrieval signals. MIITA computes the query semantic key k_q by applying the same pooling function in Eq. (1) to the query token positions. This key provides the semantic signal for accessing memory. Since the ground-truth output is unavailable at inference time, MIITA also constructs an unsupervised directional proxy from local prediction uncertainty. Let h_q be the pre-LM-head hidden state at the answer-start or first probe decoding position, and let $p_q = \mathrm{softmax}(W_{\mathrm{LM}} h_q)$. We keep the top-$K_v$ token set $\mathcal{V}_q$ under p_q and renormalize their probabilities as $\hat{p}_q(v) = p_q(v) / \sum_{v' \in \mathcal{V}_q} p_q(v')$ for $v \in \mathcal{V}_q$. The restricted entropy is then $\hat{H}_q = -\sum_{v \in \mathcal{V}_q} \hat{p}_q(v) \log \hat{p}_q(v)$. The gradient of $\hat{H}_q$ with respect to the restricted logits is

$$g_q = -\hat{p}_q \odot \left(\log \hat{p}_q + \hat{H}_q \mathbf{1} \right). \tag{6}$$

Projecting the restricted entropy gradient back to the hidden space gives

$$\tilde{d}_q = \mathrm{Norm}(-r_q), \qquad \text{where } r_q = W_{\mathcal{V}_q}^{\top} g_q. \tag{7}$$

If the norm of r_q exceeds a stability threshold ϵ_r, MIITA uses $\tilde{d}_q$ as the entropy-decreasing query direction. This direction is the steepest first-order direction for decreasing the restricted predictive entropy $\hat{H}_q$ at the current query state (Appendix A, Proposition A.2), and therefore serves as a label-free geometric cue for retrieving historical directions that may affect the current prediction.

Memory retrieval and direction aggregation. Given k_q and, when available, $\tilde{d}_q$, MIITA retrieves memory prototypes from both semantic and directional channels. For each prototype u_j, its semantic relevance is measured by the best matching anchor, $S_k(q, j) = \max_{a \in \mathcal{A}_j} \cos(k_q, a)$. The candidate set is defined as

$$\mathcal{C}_q = \mathrm{Top}_{R_k}(S_k(q, j)) \cup \begin{cases} \mathrm{Top}_{R_d}\left(\cos(\tilde{d}_q, d_j) \right), & \text{if } \tilde{d}_q \text{ is available,} \\ \varnothing, & \text{otherwise.} \end{cases} \tag{8}$$

Here, Top_{R_k} and Top_{R_d} return the prototype indices with the highest semantic and directional scores, respectively. For each candidate u_j with $j \in \mathcal{C}_q$, MIITA computes

$$\mathrm{score}(q, j) = \lambda_k S_k(q, j) + \lambda_u U(q, j), \qquad \text{where } U(q, j) = \max\left(0, -g_q^{\top} \epsilon W_{\mathcal{V}_q} d_j\right). \tag{9}$$

Here, $U(q, j)$ estimates whether moving along d_j would locally reduce the query uncertainty, ϵ is a small probe scale, and $\lambda_k, \lambda_u \geq 0$ balance semantic relevance and directional utility. If the directional proxy is unavailable, we set $U(q, j) = 0$. The top-M candidates under $\mathrm{score}(q, j)$ are selected as S_q and aggregated into a query-specific correction direction $\bar{d}_q$:

$$\bar{d}_q = \mathrm{Norm}\left(\sum_{j \in S_q} w_j d_j \right), \qquad \text{where } w_j = \frac{\exp(\mathrm{score}(q, j)/T_{\mathrm{agg}})}{\sum_{r \in S_q} \exp(\mathrm{score}(q, r)/T_{\mathrm{agg}})}. \tag{10}$$

Here, T_{agg} is the softmax temperature. The resulting $\bar{d}_q$ summarizes the historical correction directions retrieved for the current query.

Temporary hidden adaptation. The aggregated direction $\bar{d}_q$ is applied during generation through a query-specific hidden update. Rather than applying the same correction uniformly to all decoding states, MIITA uses an activation gate to make the correction stronger when the current generation state is close to the query state that triggered retrieval:

$$p'_t = \mathrm{softmax}(W_{\mathrm{LM}}(h_t^{\mathrm{out}} + \eta \alpha_t \bar{d}_q)), \qquad \text{where } \alpha_t = \max(0, \cos(\mathrm{Norm}(h_q), \mathrm{Norm}(h_t^{\mathrm{out}}))). \tag{11}$$

Here, η is the adaptation scale. Both the direction aggregation in Eq. (10) and the activation gate in Eq. (11) admit a local least-squares interpretation (Appendix A, Proposition A.3). The ablation and adversarial retrieval experiment in Appendix D.5 show that gating yields higher OP and BWT under both standard retrieval and adversarial replacement of selected prototypes. Thus, the inference-time update can be viewed as a low-cost rank-one correction in hidden space, modulated by local state similarity.

Theorem (Directional coverage bound, informal). *Suppose the supervised loss is β-smooth in the pre-LM-head hidden space. For an old-stage example z, suppose $G_z^\alpha \neq 0$ and the direction $\bar{d}_z$ returned by retrieval satisfies $\langle b_z, \bar{d}_z \rangle > 0$. For sufficiently small $\eta > 0$, adaptation decreases the local supervised token loss by at least*

$$\eta \|G_z^\alpha\|_2 \langle b_z, \bar{d}_z \rangle - \frac{\beta \eta^2}{2},$$

where G_z^α is the gated old-task correction vector and $b_z = G_z^\alpha / \|G_z^\alpha\|_2$. Thus, the benefit of memory is controlled by its directional coverage *of historical corrections, rather than by raw example coverage. The full statement and proof are given in Appendix A, Theorem A.4.*

This bound clarifies the role of memory in MIITA: stored prototypes should cover diverse historical correction directions that can reduce old-stage risk after retrieval and adaptation. This provides a unified theoretical view of the design choices, including direction prototypes in Section 3.2.1, direction-preserving memory repair in Section 3.2.3, and temporary hidden adaptation. The update is discarded after generation, enabling local adaptation without modifying the backbone, updating the LM head, extending the prompt with retrieved examples, or running test-time backpropagation. This is especially suitable for SLM CL, where repeatedly writing new experiences into a capacity-limited backbone can overwrite prior capabilities. An inference-time complexity analysis is in Appendix B.1.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate our method on four supervised CL benchmarks covering classification, question answering (QA), and generation tasks: (i) **class-incremental learning (CIL)** on BANKING77 (Casanueva et al., 2020), an intent classification dataset with 77 classes, split into 7 sequential tasks with 11 new intents per task; (ii) **domain-incremental learning (DIL)** on two benchmarks: DOCUMENT SENTIMENT CLASSIFICATION (DSC) (Ke et al., 2021), which contains 10 product-review domains with a fixed positive/negative label space, and PAXQA (Li & Callison-Burch, 2023), a multilingual QA dataset where 4 languages are treated as sequential domains; and (iii) **task-incremental learning (TIL)** on the SUPERNI BENCHMARK (Wang et al., 2022), a 15-task instruction-following benchmark covering dialogue generation, information extraction, QA, summarization, and sentiment classification.

Baselines. We compare MIITA with three groups of baselines: (i) general language model CL baselines, including LoRA (Hu et al., 2022) and MBPA++ (de Masson D'Autume et al., 2019); (ii) memory-based LLM CL baselines, including CIS (Qiu et al., 2025), InCA (Momeni et al., 2025), CT0 (Scialom et al., 2022), and FOREVER (Feng et al., 2026); and (iii) SLM-oriented CL baselines, including DKVB (Diera et al., 2025) and K et al. (2026). Implementation details for MIITA and the adaptation of each baseline to our supervised SLM CL setting are provided in Appendix C.

Evaluation Metrics. We evaluate CL performance using a performance matrix $\mathbf{A} \in \mathbb{R}^{T \times T}$, where $A_{t,\tau}$ denotes the task-specific score on task τ after the model has learned task t. For classification tasks, $A_{t,\tau}$ is accuracy; for QA tasks, $A_{t,\tau}$ is exact match; and for generation tasks in SUPERNI, $A_{t,\tau}$ is Rouge-L. All scores are scaled to $[0, 100]$. We report **Overall Performance (OP)** as $\mathrm{OP} = \frac{1}{T} \sum_{\tau=1}^{T} A_{T,\tau}$, which measures the final average task score over all tasks, and **Backward Transfer (BWT)** as $\mathrm{BWT} = \frac{1}{T-1} \sum_{\tau=1}^{T-1} (A_{T,\tau} - A_{\tau,\tau})$, which measures how learning later tasks affects earlier tasks. Higher OP and larger BWT are better.

4.2 Overall Performance

Table 1 reports the main results for Qwen3-0.6B and Qwen3-4B under a 1% persistent memory budget. MIITA consistently achieves the best OP and BWT across all benchmarks. This simultaneous improvement confirms that MIITA genuinely enhances the stability-plasticity trade-off, successfully incorporating new experiences without simply overfitting later stages at the expense of earlier ones. Additional byte-matched comparisons with parameter-efficient tuning and kNN-LM baselines on BANKING77 and SUPERNI with Qwen3-0.6B are reported in Appendix D.1. Crucially, MIITA overcomes the fundamental limitations of both parameter-update and textual-memory paradigms.

Table 1: Overall continual learning performance on four datasets, reported as mean±standard deviation over three independent runs. The standard deviation appears at the lower right of each mean. **Bold** indicates the best mean, underline indicates the second-best mean, and ↑ indicates that higher values are better.

Base Model	Method	Memory-Based	Designed For	BANKING77		DSC		PAXQA		SUPERNI	
				OP↑	BWT↑	OP↑	BWT↑	OP↑	BWT↑	OP↑	BWT↑
Qwen3-0.6B	LoRA	✗	–	$65.31_{\pm 0.31}$	$-11.44_{\pm 0.16}$	$66.04_{\pm 0.34}$	$-11.53_{\pm 0.18}$	$40.75_{\pm 0.25}$	$-5.57_{\pm 0.13}$	$31.24_{\pm 0.28}$	$-8.91_{\pm 0.14}$
	MBPA++	✓	LM	$67.82_{\pm 0.36}$	$-10.22_{\pm 0.19}$	$67.22_{\pm 0.35}$	$-9.12_{\pm 0.17}$	$40.48_{\pm 0.28}$	$-6.12_{\pm 0.15}$	$31.58_{\pm 0.29}$	$-10.21_{\pm 0.16}$
	CIS	✓	LLM	$66.78_{\pm 0.33}$	$-10.60_{\pm 0.18}$	$65.47_{\pm 0.37}$	$-10.66_{\pm 0.20}$	$38.96_{\pm 0.31}$	$-6.74_{\pm 0.14}$	$26.51_{\pm 0.24}$	$-11.05_{\pm 0.17}$
	InCA	✓	LLM	$71.20_{\pm 0.41}$	$-8.72_{\pm 0.16}$	$66.79_{\pm 0.38}$	$-10.25_{\pm 0.19}$	$39.47_{\pm 0.26}$	$-6.17_{\pm 0.15}$	$28.58_{\pm 0.32}$	$-11.17_{\pm 0.21}$
	CT0	✓	LLM	$68.79_{\pm 0.39}$	$-9.78_{\pm 0.15}$	$67.71_{\pm 0.35}$	$-9.75_{\pm 0.18}$	$41.46_{\pm 0.30}$	$-5.75_{\pm 0.14}$	$30.43_{\pm 0.27}$	$-10.82_{\pm 0.19}$
	FOREVER	✓	LLM	$70.58_{\pm 0.34}$	$-9.14_{\pm 0.16}$	$68.12_{\pm 0.31}$	$-9.63_{\pm 0.17}$	$42.54_{\pm 0.29}$	$-5.24_{\pm 0.13}$	$32.82_{\pm 0.25}$	$-9.95_{\pm 0.15}$
	DKVB	✗	SLM	$\underline{73.12}_{\pm 0.29}$	$\underline{-7.43}_{\pm 0.14}$	$\underline{75.47}_{\pm 0.27}$	$\underline{-7.23}_{\pm 0.12}$	$53.45_{\pm 0.22}$	$-4.64_{\pm 0.11}$	$\underline{37.89}_{\pm 0.24}$	$\underline{-6.72}_{\pm 0.13}$
	Srinath K *et al.*	✓	SLM	$70.44_{\pm 0.35}$	$-7.92_{\pm 0.17}$	$69.42_{\pm 0.32}$	$-9.24_{\pm 0.19}$	$\underline{55.64}_{\pm 0.28}$	$\underline{-4.43}_{\pm 0.14}$	$35.54_{\pm 0.31}$	$-7.44_{\pm 0.16}$
	MIITA (ours)	✓	SLM	$\mathbf{77.67}_{\pm 0.26}$	$\mathbf{-6.17}_{\pm 0.12}$	$\mathbf{78.29}_{\pm 0.24}$	$\mathbf{-6.14}_{\pm 0.11}$	$\mathbf{60.01}_{\pm 0.21}$	$\mathbf{-3.12}_{\pm 0.09}$	$\mathbf{40.58}_{\pm 0.27}$	$\mathbf{-4.89}_{\pm 0.12}$
Qwen3-4B	LoRA	✗	–	$68.42_{\pm 0.32}$	$-9.85_{\pm 0.15}$	$69.38_{\pm 0.35}$	$-9.74_{\pm 0.17}$	$44.92_{\pm 0.27}$	$-4.82_{\pm 0.12}$	$34.17_{\pm 0.29}$	$-7.96_{\pm 0.14}$
	MBPA++	✓	LM	$71.36_{\pm 0.38}$	$-8.71_{\pm 0.18}$	$70.54_{\pm 0.33}$	$-7.82_{\pm 0.16}$	$44.15_{\pm 0.29}$	$-5.05_{\pm 0.14}$	$36.35_{\pm 0.31}$	$-8.41_{\pm 0.15}$
	CIS	✓	LLM	$71.84_{\pm 0.35}$	$-8.88_{\pm 0.17}$	$69.75_{\pm 0.36}$	$-8.92_{\pm 0.19}$	$43.05_{\pm 0.28}$	$-5.73_{\pm 0.13}$	$38.40_{\pm 0.26}$	$-8.92_{\pm 0.16}$
	InCA	✓	LLM	$\underline{76.48}_{\pm 0.39}$	$-6.95_{\pm 0.15}$	$70.95_{\pm 0.37}$	$-8.76_{\pm 0.18}$	$43.74_{\pm 0.25}$	$-5.28_{\pm 0.14}$	$40.12_{\pm 0.30}$	$-8.64_{\pm 0.20}$
	CT0	✓	LLM	$73.92_{\pm 0.37}$	$-7.78_{\pm 0.14}$	$71.86_{\pm 0.34}$	$-8.15_{\pm 0.17}$	$46.18_{\pm 0.29}$	$-4.52_{\pm 0.13}$	$42.68_{\pm 0.28}$	$-8.37_{\pm 0.18}$
	FOREVER	✓	LLM	$75.10_{\pm 0.32}$	$-7.21_{\pm 0.15}$	$72.64_{\pm 0.30}$	$-7.65_{\pm 0.16}$	$47.32_{\pm 0.27}$	$-4.12_{\pm 0.12}$	$44.00_{\pm 0.26}$	$-8.05_{\pm 0.14}$
	DKVB	✗	SLM	$76.22_{\pm 0.28}$	$\underline{-6.31}_{\pm 0.13}$	$\underline{78.10}_{\pm 0.26}$	$\underline{-6.05}_{\pm 0.11}$	$57.82_{\pm 0.21}$	$-3.96_{\pm 0.10}$	$\underline{44.76}_{\pm 0.23}$	$\underline{-6.08}_{\pm 0.12}$
	Srinath K *et al.*	✓	SLM	$73.18_{\pm 0.33}$	$-6.94_{\pm 0.16}$	$72.68_{\pm 0.31}$	$-7.86_{\pm 0.18}$	$\underline{61.47}_{\pm 0.27}$	$\underline{-3.62}_{\pm 0.13}$	$41.85_{\pm 0.30}$	$-6.91_{\pm 0.15}$
	MIITA (ours)	✓	SLM	$\mathbf{81.05}_{\pm 0.22}$	$\mathbf{-5.02}_{\pm 0.10}$	$\mathbf{81.66}_{\pm 0.25}$	$\mathbf{-4.95}_{\pm 0.11}$	$\mathbf{65.21}_{\pm 0.19}$	$\mathbf{-2.38}_{\pm 0.08}$	$\mathbf{48.21}_{\pm 0.20}$	$\mathbf{-4.27}_{\pm 0.10}$

Parameter-updating baselines (e.g., LoRA) suffer from severe forgetting, while SLM-oriented methods show only task-dependent gains, confirming that repeatedly modifying limited SLM capacity is unreliable. Similarly, memory-based LLM methods struggle under strict storage budgets, especially on the smaller 0.6B backbone and open-ended tasks. This validates our intuition that textual memory is inefficient for SLMs, as it consumes surface-form storage and demands strong in-context reasoning. Overall, MIITA's consistent superiority validates its core design. Rather than destructively overwriting parameters or relying on surface textual prompts, MIITA preserves compact functional correction directions. Applying these directions via disposable inference-time hidden adaptation provides a more direct, reusable, and non-destructive mechanism for continual learning. Query-time costs are reported in Appendix B.1: on SUPERNI with Qwen3-0.6B, MIITA attains 40.58 OP at 721 ± 26 ms/query, a 17% increase over the frozen backbone (615 ± 21 ms/query).

4.3 MODEL-SIZE SCALING UNDER A FIXED MEMORY BUDGET

To investigate whether existing methods remain effective as the base model becomes smaller under the same memory budget, we conduct model scaling experiments, as shown in Figure 2. The results reveal a severe degradation in memory methods designed for LLMs. Specifically, replay-based methods (CT0, FOREVER) drop substantially, highlighting their reliance on large parameter capacity to absorb and retain knowledge. Similarly, prompting-based methods (CIS, InCA) falter because SLMs lack the strong in-context reasoning required to utilize textual memories. While SLM-oriented baselines (DKVB, Srinath K *et al.*) exhibit better relative robustness, their performance still declines, indicating that purely architectural or replay-adapter solutions are insufficient. In contrast, MIITA consistently achieves the best OP and BWT across all scales. By preserving compact functional correction directions rather than raw text or examples, MIITA ensures that historical memory remains highly effective despite limited model capacity and tight storage budgets.

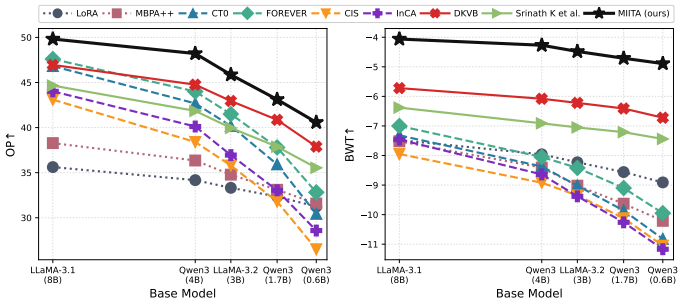


Figure 2: Model-size scaling on SUPERNI under 1% budget.

4.4 Memory Budget Sensitivity

Figure 3 evaluates memory-based CL methods under increasing storage budgets, focusing on those that scale with external memory and omitting fixed-storage baselines such as LoRA and DKVB. The results show that existing memory-based LLM CL methods are highly sensitive to available storage: replay-based methods like CT0 and FOREVER improve substantially as the budget increases, with the latter becoming strongest in the practically unlimited-memory setting by leveraging large buffers to retain extensive raw supervision from previous tasks. However, their performance drops sharply under tight budgets, indicating that replaying old examples is inefficient when memory is minimal. In contrast, MIITA remains consistently strong in the low-budget regime (0.1% to 2%), where the gap between our approach and memory methods designed for LLMs is most pronounced. This supports our central motivation: for SLM continual learning, the key challenge is not merely whether memory is used, but whether the stored representation is compact and functionally useful. While methods storing raw examples, summaries, or prompt contexts require significantly larger budgets to be competitive, MIITA stores supervised correction directions that directly encode reusable update signals. For a storage measurement on BANKING77 with Qwen3-0.6B, the final memory bank occupies 0.205 MB under the default 1% budget, compared with 20.60 MB for the full supervised stream; Appendix D reports the prototype and storage trajectory under the 5% budget.

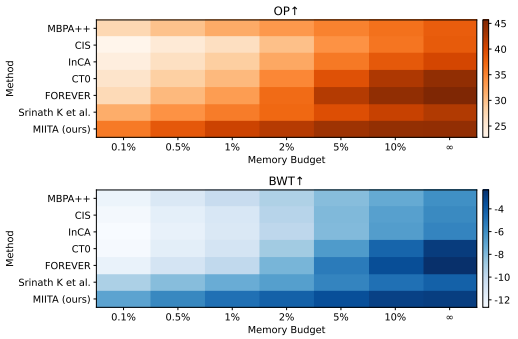
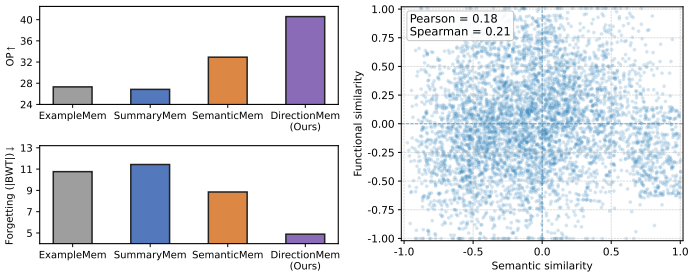


Figure 3: Budget sensitivity on SuperNI with Qwen3-0.6B for memory-based methods. Each column denotes a fixed auxiliary memory budget; ∞ denotes the unlimited-memory setting.

4.5 Effect of Functional Correction Directions

A key question in MIITA is whether supervised experiences should be stored as textual or semantic content, or as functional correction directions that encode their effect on the model prediction.

Memory variant comparison. To first isolate the effect of what is stored in memory, we compare four variants differing exclusively in their memory content: **ExampleMem** stores raw supervised examples (x, y), **SummaryMem** stores textual summaries of historical examples, **SemanticMem** stores semantic embedding prototypes, and **DirectionMem** stores correction-direction prototypes induced by supervised examples. Implementation details are provided in Appendix C.3. As shown in Figure 4(a), ExampleMem and SummaryMem are limited by the weak in-context utilization of small models, since they require the SLM to reuse retrieved textual contexts. SemanticMem avoids long textual memories and is more compact, but its semantic-redundancy-based compression does not reliably preserve the functional update signals needed under a tight storage budget. In contrast, DirectionMem directly stores correction-direction prototypes, preserving how past supervised examples should change the model's prediction. Its superior OP and forgetting results show that MIITA benefits from capturing functional redundancy, rather than merely storing human-readable text or semantic similarity. Under a non-binding 5% storage cap on BANKING77, 9,993 supervised examples form 87 prototypes, with mean inter-prototype cosine ranging from 0.557 to 0.610 across five checkpoints, as shown in Appendix D.3; the bank



(a) (b)

Figure 4: Analysis of why MIITA stores functional correction directions on SuperNI. **(a)** Comparison of memory variants with Qwen3-0.6B under a fixed 1% memory budget. **(b)** Pairwise comparison between semantic and functional similarity, where each point represents a pair of training examples.

consolidates similar corrections while continuing to add directions as the stream grows. An offline diagnostic on held-out SUPERNI queries finds that the oracle-matched stored prototype has 0.81 cosine similarity to the uncompressed label-derived direction, retains 82.6% of its supervised loss reduction, and reduces loss on 76.7% of queries, as reported in Appendix D.2.

Semantic vs. functional redundancy. To further validate the divergence between semantic and functional redundancy, we compare pairwise cosine similarity among training examples, using semantic keys for semantic similarity and correction directions for functional similarity. The two measures are only weakly correlated, with Pearson correlation 0.18 and Spearman correlation 0.21, as visualized in Figure 4(b). The Pearson correlation corresponds to only about $R^2 \approx 0.03$, meaning that semantic similarity explains less than 5% of the variance in functional similarity. Consistent with this, the points are widely scattered rather than concentrated along an increasing diagonal trend: semantically similar examples can still require very different correction directions, while semantically distant examples may share useful functional updates. This suggests that purely semantic compression can miss many functionally important directions, whereas direction-level prototypes preserve supervised update signals according to their functional effects.

4.6 VALIDATING UNCERTAINTY-GUIDED RETRIEVAL

To evaluate MIITA's retrieval design by ablating inference-time query signals, we compare variants differing exclusively in how prototypes are retrieved: **RandomRet** uniformly samples prototypes; **SemanticRet** relies solely on semantic-key similarity $S_k(q, j)$; **UncertaintyRet** scores stored directions d_j using only the uncertainty-induced direction; **SemUncRet** employs the full MIITA strategy; and **OracleRet** provides an upper bound using a label-derived oracle correction direction. To quantify consistency with supervised corrections, we report **Oracle Alignment (OA)**, the average cosine alignment between the retrieved and oracle directions. Implementation details of the retrieval variants and OA are provided in Appendix C.4. Table 2 confirms uncertainty is an effective query-side proxy. UncertaintyRet outperforms RandomRet and achieves higher OA than SemanticRet, proving it provides a meaningful label-free geometric cue. However, UncertaintyRet yields lower OP and BWT than SemanticRet because uncertainty captures local geometry but lacks semantic relevance. The optimal variant, SemUncRet, resolves this: semantic recall ensures relevance, while uncertainty prioritizes correction utility. This validates MIITA's design: uncertainty is a powerful label-free proxy, maximizing effectiveness when combined with semantic recall for reranking.

Table 2: Validation of uncertainty-guided retrieval on SUPERNI with Qwen3-0.6B under 1% memory budget. OracleRet is the label-informed upper bound.

Retrieval Variant	OP↑	BWT↑	OA↑
RandomRet	28.85	-12.63	0.03
SemanticRet	36.21	-6.98	0.35
UncertaintyRet	34.74	-8.92	0.42
SemUncRet (Ours)	**40.58**	**-4.89**	**0.52**
OracleRet	47.29	-2.42	–

5 CONCLUSION

We introduced MIITA, a memory-induced inference-time adaptation framework for supervised continual learning (CL) in small language models (SLMs) under constrained persistent storage. Unlike replay-based or prompting-based memory methods, MIITA preserves historical experiences as compact functional correction directions rather than raw examples, textual summaries, or long contexts. These directions are organized into budget-aware prototypes with lightweight semantic anchors and are invoked at inference time through semantic and uncertainty-guided retrieval followed by gated temporary hidden-state adaptation. This enables non-destructive reuse of past supervision without backbone updates, prompt extension, or test-time backpropagation. Experiments across diverse CL settings show that MIITA improves final performance and mitigates forgetting under fixed memory budgets, while further analyses demonstrate its robustness across model scales and memory budgets and validate the roles of functional correction memory and uncertainty-guided retrieval. Overall, this work suggests that SLM CL can benefit from shifting memory design away from storing textual content toward storing reusable functional adaptation signals.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing portions of the manuscript and to provide feedback on the presentation of the methodology and experimental comparisons during revision. The authors reviewed and revised all AI-assisted text and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

An anonymous implementation of MIITA is available at `https://anonymous.4open.science/r/MIITA-7AA0/`. Section 3 describes the method, and Section 4 specifies the evaluation protocol and experimental settings. Additional algorithmic details, baseline descriptions, and theoretical proofs are provided in Appendices B.2, C.2, and A, respectively.

REFERENCES

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In Tsung-Hsien Wen, Asli Celikyilmaz, Zhou Yu, Alexandros Papangelis, Mihail Eric, Anuj Kumar, Iñigo Casanueva, and Rushin Shah (eds.), *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL `https://aclanthology.org/2020.nlp4convai-1.5/`.

Evan Chen, Wenzhi Fang, Shiqiang Wang, and Christopher Brinton. Joint continual learning of local language models and cloud offloading decisions with budget constraints. *arXiv preprint arXiv:2602.00166*, 2026.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Soham Dan, et al. Larimar: Large language models with episodic memory control. *arXiv preprint arXiv:2403.11901*, 2024.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Andor Diera, Lukas Galke, Fabian Karl, and Ansgar Scherp. Efficient continual learning for small language models with a discrete key-value bottleneck. In *Proceedings of the 8th International Conference on Natural Language and Speech Processing (ICNLSP-2025)*, pp. 162–176, 2025.

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. *arXiv preprint arXiv:2601.03938*, 2026.

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

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Santosh Srinath K, Mudit Somani, Varun Reddy Padala, Prajna Devi Upadhyay, and Abhijit Das. Continual-learning for modelling low-resource languages from large language models. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pp. 6258–6275, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.293. URL https://aclanthology.org/2026.eacl-long.293/.

Thomas Katraouras and Dimitrios Rafailidis. Memory bank compression for continual adaptation of large language models. *arXiv preprint arXiv:2601.00756*, 2026.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34: 22443–22456, 2021.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Bryan Li and Chris Callison-Burch. Paxqa: Generating cross-lingual question answering examples at training scale. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 439–454, 2023.

Dongfang Li, Zetian Sun, Xinshuo Hu, Baotian Hu, and Min Zhang. Cmt: A memory compression method for continual knowledge learning of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24413–24421, 2025.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*, 2024.

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

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12573–12589, 2023.

Saleh Momeni, Sahisnu Mazumder, Zixuan Ke, and Bing Liu. In-context continual learning assisted by an external continual learner. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7292–7306, 2025.

Jiabao Qiu, Zixuan Ke, and Bing Liu. Continual learning using only large language model prompting. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6014–6023, 2025.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

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

Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. *Advances in Neural Information Processing Systems*, 37:130109–130135, 2024.

Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, et al. A survey on small language models. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing-Natural Language Processing in the Generative AI Era*, pp. 807–821, 2025.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, et al. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *ACM Transactions on Intelligent Systems and Technology*, 16(6):1–87, 2025.

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, 2022.

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

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

A THEORETICAL ANALYSIS

We provide a local theoretical analysis of MIITA to justify the main components used in Section 3.2: correction-direction extraction, uncertainty-guided retrieval, temporary hidden adaptation, and direction-memory coverage. The analysis does not assume global convexity or task stationarity; it only uses differentiability of the LM-head loss and first-order local approximations.

Notation. For a target token y with pre-LM-head hidden state $h \in \mathbb{R}^H$, define

$$\ell_y(h) = -\log \mathrm{softmax}(W_{\mathrm{LM}}h)[y].$$

Let $p = \mathrm{softmax}(W_{\mathrm{LM}}h)$. Then

$$\nabla_h \ell_y(h) = W_{\mathrm{LM}}^\top (p - e_y), \qquad -\nabla_h \ell_y(h) = W_{\mathrm{LM}}^\top (e_y - p).$$

For an experience m, define its hidden-perturbed loss

$$\mathcal{L}_m(\delta) = \frac{1}{|\mathcal{Y}_m|} \sum_{t \in \mathcal{Y}_m} \ell_{y_t}(h_t^{\mathrm{out}} + \delta).$$

Then the correction vector used by MIITA is exactly

$$R_m = -\nabla_\delta \mathcal{L}_m(0).$$

Proposition A.1 (First-order optimality of correction directions)**.** *Assume $R_m \neq 0$ and let $d_m = R_m/\|R_m\|_2$. For any unit direction d and sufficiently small $\eta > 0$,*

$$\mathcal{L}_m(\eta d) = \mathcal{L}_m(0) - \eta \langle R_m, d \rangle + o(\eta).$$

Therefore, among all unit-norm hidden-space directions, d_m maximizes the first-order decrease of $\mathcal{L}_m$:

$$d_m = \arg\max_{\|d\|_2 = 1} \langle R_m, d \rangle.$$

Proof. By the first-order Taylor expansion of $\mathcal{L}_m$ around $\delta = 0$,

$$\mathcal{L}_m(\eta d) = \mathcal{L}_m(0) + \eta \langle \nabla_\delta \mathcal{L}_m(0), d \rangle + o(\eta).$$

Since $R_m = -\nabla_\delta \mathcal{L}_m(0)$, we obtain

$$\mathcal{L}_m(\eta d) = \mathcal{L}_m(0) - \eta \langle R_m, d \rangle + o(\eta).$$

The first-order loss decrease is therefore proportional to $\langle R_m, d \rangle$. By Cauchy–Schwarz,

$$\langle R_m, d \rangle \leq \|R_m\|_2 \|d\|_2 = \|R_m\|_2,$$

with equality if and only if $d = R_m/\|R_m\|_2 = d_m$. Hence d_m is the steepest first-order correction direction in the pre-LM-head hidden space. □

Proposition A.2 (Uncertainty direction as a local geometric proxy)**.** *Fix the restricted vocabulary set $\mathcal{V}_q$ at query state h_q. Let*

$$\hat{p}_q(v) = \frac{p_q(v)}{\sum_{v' \in \mathcal{V}_q} p_q(v')}, \qquad v \in \mathcal{V}_q,$$

and

$$\hat{H}_q = -\sum_{v \in \mathcal{V}_q} \hat{p}_q(v) \log \hat{p}_q(v).$$

Let $\hat{z}_q = W_{\mathcal{V}_q} h_q$ be the restricted logits. Then

$$g_q = \frac{\partial \hat{H}_q}{\partial \hat{z}_q} = -\hat{p}_q \odot \left(\log \hat{p}_q + \hat{H}_q \mathbf{1} \right),$$

and

$$r_q = W_{\mathcal{V}_q}^\top g_q = \nabla_{h_q} \hat{H}_q.$$

Therefore, if $r_q \neq 0$, the direction

$$\tilde{d}_q = -\frac{r_q}{\|r_q\|_2}$$

is the steepest first-order direction for decreasing the restricted entropy.

Proof. For restricted softmax probabilities $\hat{p}$, the entropy is

$$\hat{H} = -\sum_v \hat{p}_v \log \hat{p}_v.$$

Using the softmax Jacobian

$$\frac{\partial \hat{p}_i}{\partial \hat{z}_j} = \hat{p}_i(\mathbf{1}[i=j] - \hat{p}_j),$$

we have

$$\frac{\partial \hat{H}}{\partial \hat{z}_j} = \sum_i -(\log \hat{p}_i + 1)\hat{p}_i(\mathbf{1}[i=j] - \hat{p}_j).$$

Rearranging gives

$$\frac{\partial \hat{H}}{\partial \hat{z}_j} = -\hat{p}_j(\log \hat{p}_j + \hat{H}).$$

Thus

$$g_q = -\hat{p}_q \odot (\log \hat{p}_q + \hat{H}_q \mathbf{1}).$$

Since $\hat{z}_q = W_{\mathcal{V}_q} h_q$, the chain rule gives

$$\nabla_{h_q} \hat{H}_q = W_{\mathcal{V}_q}^\top g_q = r_q.$$

For any unit direction d,

$$\hat{H}_q(h_q + \eta d) = \hat{H}_q(h_q) + \eta \langle r_q, d \rangle + o(\eta).$$

The first-order entropy decrease is maximized by minimizing $\langle r_q, d \rangle$, whose unit-norm solution is $d = -r_q / \|r_q\|_2$. □

Moreover,

$$U(q,j) = \max\left(0, -g_q^\top \epsilon W_{\mathcal{V}_q} d_j\right) = \epsilon \max\left(0, -\langle r_q, d_j \rangle\right).$$

Therefore $U(q,j)$ is exactly the positive part of the first-order restricted-entropy decrease induced by moving along historical direction d_j. This proxy does not assume that the entropy-decreasing direction equals the ground-truth correction direction. Rather, it exposes the local geometry of the current predictive distribution and is used as a label-free auxiliary retrieval cue for comparing the query state with stored historical directions.

Proposition A.3 (Local least-squares derivation of the hidden adapter)**.** *Given selected memory directions $\{d_j\}_{j \in S_q}$ and weights $w_j \geq 0$, $\sum_j w_j = 1$, consider the problem of representing them by a single unit correction direction:*

$$\min_{\|d\|_2 = 1} \sum_{j \in S_q} w_j \|d - d_j\|_2^2.$$

If $\sum_j w_j d_j \neq 0$, the solution is

$$\bar{d}_q = \frac{\sum_{j \in S_q} w_j d_j}{\left\| \sum_{j \in S_q} w_j d_j \right\|_2}.$$

Furthermore, with

$$\hat{h}_q = \mathrm{Norm}(h_q), \qquad \hat{h}_t = \mathrm{Norm}(h_t^{\mathrm{out}}),$$

the activation gate

$$\alpha_t = \max(0, \cos(\hat{h}_t, \hat{h}_q))$$

is the nonnegative least-squares coefficient for projecting the current decoding state onto the query state.

Proof. Expanding the direction-aggregation objective,

$$\sum_j w_j \|d - d_j\|_2^2 = \sum_j w_j (\|d\|_2^2 + \|d_j\|_2^2 - 2\langle d, d_j \rangle).$$

Since $\|d\|_2 = 1$, $\|d_j\|_2 = 1$, and $\sum_j w_j = 1$, minimizing the objective is equivalent to maximizing

$$\left\langle d, \sum_j w_j d_j \right\rangle.$$

By Cauchy–Schwarz, the maximizer is the normalized weighted mean direction.

For the gate, consider the nonnegative least-squares projection

$$\alpha_t = \arg\min_{\alpha \in [0,1]} \|\hat{h}_t - \alpha \hat{h}_q\|_2^2.$$

Since $\|\hat{h}_q\|_2 = \|\hat{h}_t\|_2 = 1$, the unconstrained minimizer is

$$\alpha_t^\star = \langle \hat{h}_t, \hat{h}_q \rangle.$$

Projecting it to $[0, 1]$ gives

$$\alpha_t = \max(0, \langle \hat{h}_t, \hat{h}_q \rangle) = \max(0, \cos(\hat{h}_t, \hat{h}_q)).$$

Thus, the gate is the least-squares coefficient measuring how much the current decoding state lies in the local query direction. □

Combining the least-squares prototype estimator $\bar{d}_q$ with the least-squares local activation coefficient α_t, the lowest-cost rank-one hidden adapter takes the form

$$h_t' = h_t^{\text{out}} + \eta \alpha_t \bar{d}_q.$$

This update realizes the retrieved historical correction as a single vector addition in hidden space, modulated by local state similarity, and is discarded after generation. Therefore, the temporary hidden adapter can be interpreted as the local least-squares projection of multiple retrieved historical correction signals onto a single low-cost inference-time correction field.

Theorem A.4 (Direction coverage bound for old-stage risk)**.** *Consider an old-stage example z under teacher-forced evaluation with the final frozen backbone. We hold the pre-LM-head states h_t^{out} and activation gates α_t fixed when evaluating the local effect of a retrieved direction. Let*

$$\mathcal{L}_z^\alpha(\eta, d) = \frac{1}{|\mathcal{Y}_z|} \sum_{t \in \mathcal{Y}_z} \ell_{y_t} \left(h_t^{\text{out}} + \eta \alpha_t d \right),$$

where $\alpha_t \in [0, 1]$ is the inference-time activation gate. Define the gated old-task correction vector

$$G_z^\alpha = \frac{1}{|\mathcal{Y}_z|} \sum_{t \in \mathcal{Y}_z} \alpha_t W_{\text{LM}}^\top (e_{y_t} - p_t).$$

Let $b_z = G_z^\alpha / \|G_z^\alpha\|_2$ when $G_z^\alpha \neq 0$. Suppose the memory retrieval and aggregation procedure returns a unit direction $\bar{d}_z$. Define the directional coverage error

$$\varepsilon_{\mathcal{B}}(z) = 1 - \langle b_z, \bar{d}_z \rangle.$$

If the token loss is β-smooth in hidden space, then

$$\mathcal{L}_z^\alpha(\eta, \bar{d}_z) \leq \mathcal{L}_z^\alpha(0, \bar{d}_z) - \eta \|G_z^\alpha\|_2 \big(1 - \varepsilon_{\mathcal{B}}(z)\big) + \frac{\beta \eta^2}{2}.$$

Proof. For fixed z and $\bar{d}_z$, define

$$\phi_z(\eta) = \mathcal{L}_z^\alpha(\eta, \bar{d}_z).$$

Its derivative at $\eta = 0$ is

$$\phi_z'(0) = \frac{1}{|\mathcal{Y}_z|} \sum_{t \in \mathcal{Y}_z} \alpha_t \left\langle \nabla_h \ell_{y_t}(h_t^{\text{out}}), \bar{d}_z \right\rangle.$$

Using

$$\nabla_h \ell_{y_t}(h_t^{\text{out}}) = W_{\text{LM}}^\top (p_t - e_{y_t}),$$

we get

$$\phi_z'(0) = -\left\langle G_z^\alpha, \bar{d}_z \right\rangle = -\|G_z^\alpha\|_2 \langle b_z, \bar{d}_z \rangle.$$

Since $\langle b_z, \bar{d}_z \rangle = 1 - \varepsilon_{\mathcal{B}}(z)$, and by β-smoothness,

$$\phi_z(\eta) \le \phi_z(0) + \eta \phi_z'(0) + \frac{\beta \eta^2}{2},$$

which gives the desired result. □

Let $\mathcal{P}_{\text{old}}$ be the distribution over previously seen examples. Denote the frozen final old-task risk by

$$\mathcal{R}_{\text{old}}^0 = \mathbb{E}_{z \sim \mathcal{P}_{\text{old}}}[\mathcal{L}_z^\alpha(0, \bar{d}_z)],$$

and the adapted risk by

$$\mathcal{R}_{\text{old}}^{\mathcal{B}}(\eta) = \mathbb{E}_{z \sim \mathcal{P}_{\text{old}}}[\mathcal{L}_z^\alpha(\eta, \bar{d}_z)].$$

Then

$$\mathcal{R}_{\text{old}}^{\mathcal{B}}(\eta) \le \mathcal{R}_{\text{old}}^0 - \eta \mathbb{E}_z \left[\|G_z^\alpha\|_2 \langle b_z, \bar{d}_z \rangle \right] + \frac{\beta \eta^2}{2}.$$

Thus, the benefit of memory is controlled by the directional coverage term

$$\text{Cov}(\mathcal{B}) = \mathbb{E}_z \left[\|G_z^\alpha\|_2 \langle b_z, \bar{d}_z \rangle \right].$$

If $\text{Cov}(\mathcal{B}) > 0$, then the bound yields a decrease in the defined old-stage token-loss risk for sufficiently small $\eta > 0$. Relative to any reference risk $\mathcal{R}_{\text{old}}^{\text{ref}}$ evaluated with the same token-loss objective, the risk difference satisfies

$$\mathcal{R}_{\text{old}}^{\mathcal{B}}(\eta) - \mathcal{R}_{\text{old}}^{\text{ref}} \le \mathcal{R}_{\text{old}}^0 - \mathcal{R}_{\text{old}}^{\text{ref}} - \eta \text{Cov}(\mathcal{B}) + \frac{\beta \eta^2}{2}.$$

B Extended Methodology

B.1 Inference-Time Efficiency

At inference time, MIITA adds only retrieval, reranking, and temporary hidden-state adaptation on top of standard generation. Exact semantic and directional recall cost $O(AH)$ and $O(JH)$, respectively, where $A = \sum_j |\mathcal{A}_j|$ is the number of anchors and J is the number of prototypes. Both are bounded by the fixed memory budget and can be further accelerated with approximate indexing. The uncertainty proxy and reranking cost $O(K_v H)$ and $O(|\mathcal{C}_q| K_v H)$, respectively, where K_v is the restricted vocabulary size and $\mathcal{C}_q$ is the candidate set. With fixed retrieval hyperparameters and bounded memory, the additional query-level overhead is linear in the hidden dimension H. During generation, the adapter introduces only a cosine gate and a vector addition per token, which also cost $O(H)$. Therefore, MIITA avoids prompt extension, replay, parameter updates, and test-time backpropagation, while incurring the query-time overhead measured below.

Under the same model and GPU, MIITA processes 14.2 ± 0.7 supervised examples/s during memory writing, compared with 3.8 ± 0.3 examples/s for sequential LoRA updates. The query-time increase is paid on each inference request, so the overall runtime trade-off depends on the number of writes and queries in the deployment workload.

B.2 Detailed Implementation of MIITA

Algorithm 1 summarizes the full implementation of MIITA, including memory construction, budget-aware memory repair, inference-time retrieval, and temporary hidden-state adaptation.

Table 3: Continual-learning performance and end-to-end query latency on SUPERNI with Qwen3-0.6B under the 1% persistent-memory budget. OP and BWT are three-run means from Table 1. Latency is mean±standard deviation over three repetitions of 1,000 batch-1 queries after 100 warm-up queries on one A100-80GB GPU. Relative latency uses the frozen SLM (615 ± 21 ms/query) as the reference.

Method	OP↑	BWT↑	Latency (ms/query)↓	Relative latency
LoRA	31.24	-8.91	616 ± 20	$1.00\times$
MBPA++	31.58	-10.21	$4{,}220 \pm 173$	$6.86\times$
InCA	28.58	-11.17	865 ± 31	$1.41\times$
FOREVER	32.82	-9.95	620 ± 22	$1.01\times$
DKVB	37.89	-6.72	649 ± 24	$1.06\times$
MIITA (ours)	**40.58**	**-4.89**	721 ± 26	$1.17\times$

C EXPERIMENTAL DETAILS

C.1 IMPLEMENTATION DETAILS

We evaluate all methods on instruction-tuned base models, including Qwen3 (0.6B, 1.7B, 4B) (Yang et al., 2025) and LLaMA (3.2-3B, 3.1-8B) (Grattafiori et al., 2024). Unless otherwise specified, we use a fixed 1% memory budget, measured as 1% of the byte size of the full training stream. For MIITA, we set the direction-merge threshold to $\tau_d = 0.85$ and the semantic-anchor novelty threshold to $\tau_k = 0.80$. During inference, we retrieve $R_k = R_d = 8$ candidates and aggregate the top $M = 4$ prototypes after reranking with weights $\lambda_k = 1.0$ and $\lambda_u = 0.5$, using $T_{\text{agg}} = 0.2$ and $\eta = 0.5$. Results are reported as mean±standard deviation over three independent runs.

C.2 BASELINE DETAILS

General protocol. All baselines are evaluated under the same budget-constrained supervised SLM CL setting. Unless otherwise specified, each benchmark is converted into the unified instruction-style format (x, y), and all methods use the same SLM backbone. For methods that store auxiliary information, including replay samples, summaries, statistics, episodic memories, adapters, or bottleneck parameters, we constrain the total persistent storage to the same budget C_{bytes}. Methods designed only for classification are evaluated on the classification benchmarks and are marked as not applicable to open-ended QA.

Baseline descriptions and adaptations.

- **LoRA** (Hu et al., 2022). LoRA is a parameter-efficient sequential tuning baseline that updates low-rank adapters while keeping the SLM backbone frozen. We train one LoRA module sequentially on each incoming stage using only the current data $\mathcal{D}_t$, without replay or external memory. The LoRA parameters are carried across stages as the model's continual update state.
- **MBPA++** (de Masson D'Autume et al., 2019). MBPA++ is an episodic-memory language-model CL method that stores past examples and uses sparse replay together with inference-time local adaptation. We replace its original language model with the same SLM backbone used by our method, store serialized (x, y) examples in episodic memory under C_{bytes}, retrieve nearest neighbors using frozen SLM representations, and perform temporary local LoRA adaptation for each query before discarding the update.
- **CT0** (Scialom et al., 2022). CT0 is a training-time rehearsal baseline that continually learns new instruction-following tasks by replaying a small buffer of previous examples. We adapt it by maintaining a budgeted replay buffer of old supervised examples and mixing sampled replay data with the current stage data during LoRA-based continual tuning.
- **FOREVER** (Feng et al., 2026). FOREVER is a replay-based LLM CL method that schedules memory replay according to model-centric update dynamics. We use the same budgeted replay buffer as CT0, but trigger replay according to FOREVER's model-time schedule computed from accumulated LoRA update norms, and apply its replay-time regularization during continual tuning.

Algorithm 1 MIITA: Memory-Induced Inference-Time Adaptation Framework for CL

Require: Stream $\{\mathcal{D}_t\}_{t=1}^{T}$, base SLM f, budget C_{mem}, hyperparameters $\{\tau_d, \tau_k, R_k, R_d, M, \lambda_k, \lambda_u, T_{\mathrm{agg}}, \eta\}$
Ensure: Memory bank $\mathcal{B}$, prediction $\hat{y}$

$\mathcal{B} \leftarrow \varnothing$

// Phase 1: Build budgeted direction memory
for $t = 1$ to T **do**
  for each $(x_m, y_m) \in \mathcal{D}_t$ **do**
    $k_m \leftarrow$ Eq. (1), $d_m \leftarrow$ Eq. (2)
    if $\mathcal{B} = \varnothing$ **then**
      Add $u = (d_m, \{k_m\}, 1)$ to $\mathcal{B}$
    else
      $j^\star \leftarrow \arg\max_j \cos(d_m, d_j)$
      if $\cos(d_m, d_{j^\star}) < \tau_d$ **then**
        Add $u = (d_m, \{k_m\}, 1)$ to $\mathcal{B}$
      else
        Update $u_{j^\star}$ by Eqs. (3) and (4)
      end if
    end if
    while $\mathrm{Size}(\mathcal{B}) > C_{\mathrm{mem}}$ **do**
      Repair $\mathcal{B}$ by anchor pruning, prototype merging, or eviction via Eq. (5)
    end while
  end for
end for

// Phase 2: Retrieve and aggregate memory directions
Given query q, compute k_q by Eq. (1) and $\tilde{d}_q$ by Eq. (7)
Construct $\mathcal{C}_q$ by Eq. (8)
for each $j \in \mathcal{C}_q$ **do**
  Compute $\mathrm{score}(q, j)$ by Eq. (9)
end for
$S_q \leftarrow \mathrm{Top}_M\{\mathrm{score}(q, j) : j \in \mathcal{C}_q\}$
Aggregate $\bar{d}_q$ by Eq. (10)

// Phase 3: Apply temporary hidden adaptation
for each decoding step s **do**
  Apply gated update h'_s by Eq. (11)
  Generate next token from h'_s
end for
Discard the temporary update
return $\hat{y}$

- **CIS** (Qiu et al., 2025). CIS is the concrete method under the CLOB paradigm, which performs black-box CL by storing class knowledge as incrementally updated textual summaries. For classification benchmarks, we follow its original design by generating and updating class-level summaries under C_{bytes} and using them in inference-time prompts. For QA benchmarks, where the output space is open-ended rather than class-based, we adapt CIS by maintaining stage-level or domain-level summaries of previously seen QA examples. At inference time, the query is paired with the relevant stored summaries in the prompt, and the SLM generates the answer directly.
- **InCA** (Momeni et al., 2025). InCA is an inference-time ICL method that uses an external continual learner to select candidate classes before prompting the LLM. For classification benchmarks, we keep this mechanism and incrementally store class-level tag statistics and summaries under C_{bytes}. For QA benchmarks, we adapt the external learner from class selection to memory selection: the learner ranks stored stage/domain summaries or representative QA memories according to the semantic tags of the query, and the top-ranked memories are used to construct the inference prompt. The final prediction is generated by the SLM rather than selected from a finite class set.

- **DKVB** (Diera et al., 2025). DKVB is an SLM-oriented CL method that uses a discrete key-value bottleneck to enable localized continual updates. Since the original method is designed for encoder-only classification, we adapt it by inserting the bottleneck at the SLM hidden representation before the prediction head. For instruction-style tasks, labels are verbalized as output tokens, and the bottleneck capacity is selected to fit the same storage budget.
- **K et al. (2026)** propose an SLM-oriented multilingual CL method for mitigating cross-lingual forgetting with replay adapters. For multilingual QA, we treat each language as a domain and use its replay-adapter strategy in the original language-incremental form. For non-multilingual benchmarks, we adapt the method by retaining the shared replay-adapter mechanism but replacing POS-guided code-switching with task- or domain-preserving replay examples from the same benchmark. Thus, the baseline becomes a replay-adapter CL method that uses the same instruction-style input-output format while removing the language-specific code-switching component when it is not applicable.

C.3 Details of Memory Item Variants

We evaluate four memory item formats under the same retrieval backbone, retrieval budget, and total storage constraint.

- **ExampleMem.** This variant stores serialized supervised examples (x, y) as memory items. At inference time, retrieved examples are inserted into the prompt as demonstrations or contextual evidence. The storage cost is measured by the byte size of the stored text.
- **SummaryMem.** This variant compresses historical examples into natural-language summaries. Retrieved summaries are inserted into the prompt before generation. The storage cost includes the byte size of the stored summaries.
- **SemanticMem.** This variant stores semantic embedding prototypes obtained from the same retrieval backbone. At inference time, retrieved semantic prototypes are used through the same retrieval interface, but they do not contain supervised correction information. The storage cost includes the stored embedding vectors and associated prototype metadata.
- **DirectionMem.** This is the memory format used by MIITA. Each supervised example induces a correction direction in hidden space, and similar directions are merged into prototypes under the memory budget. At inference time, retrieved direction prototypes are used for temporary hidden-state adaptation without gradient-based test-time training.

C.4 Details of Retrieval Variants

We evaluate retrieval variants under the same DirectionMem memory bank, retrieval backbone, retrieval budget, and total storage constraint. All variants use the same supervised stream and Qwen3-0.6B backbone. They differ only in how memory prototypes are selected or reranked at inference time.

- **RandomRet.** This variant uniformly samples memory prototypes from the DirectionMem bank. It ignores both semantic relevance and uncertainty information and serves as a lower-bound retrieval baseline.
- **SemanticRet.** This variant retrieves memory prototypes using only semantic-key similarity $S_k(q, j)$ between the query and stored memory anchors. It tests whether semantic relevance alone is sufficient for selecting useful historical correction directions.
- **UncertaintyRet.** This variant retrieves memory prototypes using only the uncertainty-induced query direction $\tilde{d}_q$. Since $\tilde{d}_q$ and the stored correction directions d_j are both defined in the LM-head input hidden space, they can be compared by directional alignment. This variant tests whether uncertainty provides a label-free query-side signal for selecting historical directions that may locally affect the current prediction distribution.
- **SemUncRet.** This is the retrieval strategy used by MIITA. It first applies semantic recall to construct a query-relevant candidate memory set and then applies uncertainty-guided soft reranking within this set. Semantic recall keeps retrieved memories related to the query, while uncertainty prioritizes directions that are more likely to influence the current predictive state.
- **OracleRet.** This variant uses the ground-truth target to compute an oracle correction direction and retrieves memory prototypes according to their alignment with this direction. For a single target

Table 4: Comparison under the same dataset-specific 1% persistent-storage budget with Qwen3-0.6B. Results are mean±standard deviation over three runs. Per-task LoRA uses oracle task-ID routing.

	BANKING77		SUPERNI	
Method	OP↑	BWT↑	OP↑	BWT↑
Sequential LoRA	65.31 ± 0.31	-11.44 ± 0.16	31.24 ± 0.28	-8.91 ± 0.14
Per-task LoRA (oracle task ID)	64.72 ± 0.33	0.00 ± 0.00	30.30 ± 0.30	0.00 ± 0.00
SAPT-LoRA (Zhao et al., 2024)	74.02 ± 0.29	-7.08 ± 0.14	36.88 ± 0.30	-5.63 ± 0.16
kNN-LM (Khandelwal et al., 2019)	72.11 ± 0.32	-8.03 ± 0.17	34.92 ± 0.29	-7.20 ± 0.15
MIITA (ours)	77.67 ± 0.26	-6.17 ± 0.12	40.58 ± 0.27	-4.89 ± 0.12

token, the oracle direction is computed as $d_q^\star = W_{\mathrm{LM}}^\top(e_y - p)$, where p is the predicted distribution and e_y is the one-hot target distribution. For generation tasks, token-level oracle directions are averaged over the target sequence. OracleRet is used only as an upper bound and is not available at inference time.

Oracle Alignment. For retrieval analysis, we compute **Oracle Alignment (OA)** on a held-out evaluation set where labels are available. For each query q, we first compute the oracle correction direction $d_q^\star$ from the ground-truth target. Each retrieval variant then produces an aggregated retrieved direction $\bar{d}_q$ using the same direction aggregation rule as MIITA. OA is defined as the average cosine similarity between $\bar{d}_q$ and $d_q^\star$:

$$\mathrm{OA} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \cos(\bar{d}_q, d_q^\star).$$

A higher OA indicates closer alignment between the aggregated retrieved direction and the label-derived oracle correction direction. In Table 2, UncertaintyRet has higher OA than SemanticRet (0.42 versus 0.35), while SemanticRet achieves higher OP and BWT. SemUncRet achieves the highest OA, OP, and BWT among the label-free retrieval variants (0.52, 40.58, and -4.89, respectively). In this experiment, combining semantic recall with uncertainty-guided reranking improves both directional alignment and final performance.

D SUPPLEMENTARY EXPERIMENTS

D.1 BUDGET-MATCHED PEFT AND KNN-LM COMPARISONS

We compare additional parameter-efficient and datastore baselines with MIITA on BANKING77 and SUPERNI using Qwen3-0.6B and the dataset-specific 1% persistent-storage budget. All methods use the same supervised streams, storage precision, and decoding settings. The carried adapter parameters of sequential LoRA count toward the budget. For per-task LoRA, the global budget is divided uniformly across the seven BANKING77 stages or 15 SUPERNI tasks, and we use the largest common rank and target-layer configuration that fits the budget; inference assumes oracle task IDs. For SAPT-LoRA, we count all parameter-efficient tuning blocks and selection parameters. For kNN-LM, we count datastore entries and index metadata.

Under the matched storage budget, MIITA achieves the highest OP on both datasets and the least negative BWT among methods without oracle task-ID routing. Compared with SAPT-LoRA, the strongest non-oracle PEFT baseline, MIITA improves OP by 3.65 points on BANKING77 and 3.70 points on SUPERNI, while improving BWT by 0.91 and 0.74 points, respectively. Its OP gains over kNN-LM are 5.56 and 5.66 points, showing that the advantage also holds against a datastore-based baseline in this fixed-budget setting.

D.2 DIRECT DIAGNOSTICS OF DIRECTION MEMORY AND UNCERTAINTY GUIDANCE

We conduct offline diagnostics on held-out SUPERNI queries with Qwen3-0.6B under the main 1% persistent-memory budget. These diagnostics separately examine whether stored prototypes retain

Table 5: Offline diagnostics on held-out SUPERNI queries with Qwen3-0.6B under the 1% persistent-memory budget. D-Fid, D-Ret, and D-Help use label-derived nearest-prototype matching; U-Err and U-Gain use reference answers only to evaluate errors or loss changes.

Diagnostic	Value
D-Fid (direction cosine)	0.81
D-Ret (retained loss reduction)	82.6%
D-Help (queries with reduced loss)	76.7%
U-Err (error-detection AUROC)	0.72
U-Gain (within-query Spearman correlation)	0.53

Table 6: Growth of the direction-memory bank on BANKING77 with Qwen3-0.6B under a non-binding 5% storage cap.

Processed examples	Prototypes	Anchors	Memory (MB)	Mean cosine
20% (1,999)	16	16	0.131	0.610
40% (3,998)	35	35	0.287	0.591
60% (5,996)	57	57	0.468	0.572
80% (7,995)	74	74	0.607	0.557
100% (9,993)	87	87	0.714	0.566

supervised correction information and whether the uncertainty signal is associated with prediction errors and useful candidate directions.

Correction information retained by prototypes. For each held-out query, we compute an uncompressed correction direction from its reference answer using Eq. (2) and match it to the closest final memory prototype in direction space. We apply the uncompressed direction and the matched prototype separately under teacher forcing, using the same adaptation scale, and measure the supervised token-level cross-entropy loss. D-Fid is the cosine similarity between the uncompressed direction and its matched prototype. D-Ret measures the proportion of the loss reduction obtained with the uncompressed direction that is retained by the matched prototype. D-Help is the percentage of queries for which applying the matched prototype reduces loss relative to no adaptation.

Reliability of uncertainty guidance. U-Err measures the AUROC of restricted entropy at the answer-start position for identifying whether the unadapted model's top-1 next-token prediction differs from the reference token. For U-Gain, we apply each prototype in a query's semantic candidate set separately through the temporary hidden-state update. We then compute the within-query Spearman correlation between its uncertainty-guided score $U(q, j)$ in Eq. (9) and its measured reduction in supervised loss.

The oracle-matched prototypes remain directionally close to the uncompressed corrections and retain much of their supervised loss-reduction effect. Restricted entropy is informative about answer-start errors, and the uncertainty-guided score is positively associated with the measured benefit of semantic candidate prototypes. End-to-end OP, BWT, and retrieved-direction OA for the retrieval variants are reported in Table 2.

D.3 PROTOTYPE GROWTH AND DIRECTIONAL SIMILARITY

We trace the direction-memory bank at five checkpoints on BANKING77 with Qwen3-0.6B under a 5% storage cap of 1.03 MB. The cap is not reached in this run. Mean inter-prototype cosine is computed over pairs of stored prototype directions.

Across the five checkpoints, the bank grows from 16 to 87 prototypes as the supervised stream grows from 1,999 to 9,993 examples. At the final checkpoint, 87 prototypes represent the directions induced by 9,993 examples, a 99.13% reduction in the number of stored directions relative to the number of incoming examples. Mean inter-prototype cosine remains between 0.557 and 0.610, while the increasing prototype count shows that the bank continues to accommodate new correction directions as new intents arrive.

Table 7: Layer-matched correction extraction and injection on SUPERNI with Qwen3-0.6B under the 1% persistent-memory budget. Results are mean±standard deviation.

Correction and injection layer	OP↑	BWT↑
$L/4$	37.48 ± 0.32	-6.43 ± 0.16
$L/2$	38.61 ± 0.30	-5.86 ± 0.15
$3L/4$	39.66 ± 0.28	-5.24 ± 0.17
Pre-LM-head (L)	$\mathbf{40.58 \pm 0.27}$	$\mathbf{-4.89 \pm 0.12}$

Table 8: Adversarial prototype replacement on SUPERNI with Qwen3-0.6B under the 1% memory budget. The replacement candidates are semantically relevant prototypes with the lowest label-informed oracle alignment. Higher OP and BWT are better.

Replaced fraction	Gated OP	Gated BWT	Ungated OP	Ungated BWT
0%	40.58	-4.89	38.72	-6.18
10%	39.96	-5.17	37.71	-6.76
25%	38.63	-5.94	35.62	-7.81
50%	35.74	-7.46	31.88	-10.02

The full BANKING77 supervised stream occupies 20.60 MB. For the same stream and backbone, the default 1% budget has a 0.206 MB cap, and the final memory bank occupies 0.205 MB, giving a stream-to-bank size ratio of 100.5×. In the 5% setting above, the final 0.714 MB bank gives a ratio of 28.9×.

D.4 EFFECT OF THE CORRECTION AND INJECTION LAYER

We compare four layer-matched variants on SUPERNI with Qwen3-0.6B under the 1% persistent-memory budget. Each variant extracts supervised correction directions and applies the retrieved directions at the same layer. The supervised stream and all other settings are held fixed.

Performance improves consistently with layer depth: OP rises from 37.48 at $L/4$ to 40.58 at the pre-LM-head layer, while BWT improves from -6.43 to -4.89. Relative to $3L/4$, the pre-LM-head variant gains 0.92 OP and 0.35 BWT. This performance trend, together with the direct correction computation in Eq. (2), supports extracting and injecting directions at the pre-LM-head layer.

D.5 EFFECT OF THE ACTIVATION GATE

We additionally evaluate the activation gate α_t used in the temporary hidden adaptation step. The experiment is conducted on SUPERNI with Qwen3-0.6B under the fixed 1% memory budget. The full MIITA uses the gated update $h'_t = h_t^{\text{out}} + \eta\alpha_t\bar{d}_q$, while the ablated variant removes the gate by setting $\alpha_t = 1$ for all decoding steps, so that the same correction direction is uniformly applied throughout generation.

Removing the gate reduces performance from 40.58 OP / -4.89 BWT to 38.72 OP / -6.18 BWT. This confirms that the gate is not a redundant component: uniformly injecting the retrieved direction can over-correct decoding states that are no longer close to the query state that triggered retrieval. In contrast, the full gated design applies stronger correction only when the current decoding state remains locally aligned with the query state. This is consistent with Proposition A.3, which interprets α_t as a nonnegative least-squares projection coefficient measuring how much the current decoding state lies in the local query direction.

Adversarial retrieval. We further test the gate by replacing a specified fraction of selected prototypes with semantically relevant candidates having the lowest label-informed oracle alignment. Table 8 compares gated and ungated adaptation under the same SUPERNI setting.

From 0% to 50% replacement, the gated model loses 4.84 OP and 2.57 BWT points, compared with 6.84 OP and 3.84 BWT points without gating. At 25% replacement, gating yields 3.01 higher OP and 1.87 higher BWT than the ungated variant. The widening performance gap as corruption increases

Table 9: Hyperparameter sensitivity on SUPERNI with Qwen3-0.6B under a fixed 1% memory budget. When sweeping one threshold, all other hyperparameters are fixed to the default configuration. **Bold** indicates the default setting used in the main experiments.

Hyperparameter	**Value**	OP↑	BWT↑
τ_d	0.75	39.82	-5.31
	0.80	40.21	-5.08
	0.85	**40.58**	**-4.89**
	0.90	40.34	-5.02
	0.95	39.76	-5.36
τ_k	0.70	40.05	-5.17
	0.75	40.31	-5.02
	0.80	**40.58**	**-4.89**
	0.85	40.39	-4.97
	0.90	40.12	-5.15

Table 10: Joint sensitivity of the retrieval weights on SUPERNI with Qwen3-0.6B under a fixed 1% memory budget. Each cell reports OP/BWT. Bold indicates the default setting.

$\lambda_k \backslash \lambda_u$	0.25	0.5	1.0
0.5	$39.78/-5.21$	$39.42/-5.45$	$38.51/-6.03$
1.0	$39.91/-5.16$	$\mathbf{40.58/-4.89}$	$39.88/-5.24$
2.0	$39.35/-5.43$	$39.96/-5.18$	$40.08/-5.11$

shows that state-dependent gating reduces the effect of adversarially replaced prototypes on final predictions.

D.6 HYPERPARAMETER SENSITIVITY

We further study the sensitivity of MIITA to the two memory-construction thresholds, τ_d and τ_k, on SUPERNI with Qwen3-0.6B under the fixed 1% memory budget. When sweeping one hyperparameter, all others are fixed to the default configuration. The default setting used in the main experiments is $\tau_d = 0.85$ and $\tau_k = 0.80$.

Table 9 shows that MIITA is robust across a reasonable range of threshold choices. For τ_d, overly small values merge directions too aggressively and may blur distinct functional corrections, while overly large values create more fragmented prototypes and reduce the benefit of direction sharing. The default value $\tau_d = 0.85$ provides the best balance between compression and functional diversity. For τ_k, lower values keep too few semantic anchors and may weaken retrieval access, while higher values preserve more anchors but consume budget with redundant semantic entry points. The default value $\tau_k = 0.80$ achieves the best overall trade-off. Across all tested values, performance remains close to the default setting, indicating that the gains of MIITA are not due to a fragile hyperparameter choice.

We also vary the retrieval weights λ_k and λ_u jointly on SUPERNI with Qwen3-0.6B under the fixed 1% memory budget, keeping the other hyperparameters at their default values. Table 10 shows that the default pair $(\lambda_k, \lambda_u) = (1.0, 0.5)$ achieves the highest OP and BWT in the evaluated grid. At $\lambda_k = 0.5$, increasing λ_u from 0.5 to 1.0 lowers OP from 39.42 to 38.51; at $\lambda_u = 0.5$, increasing λ_k from 1.0 to 2.0 lowers OP from 40.58 to 39.96. The two retrieval signals therefore perform best at an intermediate balance in this setting.

E LIMITATIONS

MIITA is designed for supervised continual learning with small language models under a fixed persistent-memory budget, and its conclusions should be interpreted within this setting. The method assumes that each incoming stage provides input-output supervision from which correction directions can be extracted, and it does not directly cover unsupervised, reinforcement-learning, multimodal, or

tool-use continual adaptation. Because the backbone remains frozen and adaptation is applied only as a temporary hidden-state perturbation, MIITA cannot recover capabilities that are absent from the base model or fully replace parameter updates when new tasks require substantial new reasoning skills or knowledge that is poorly represented in the existing hidden space. Like parameter-efficient CL states tied to model weights or hidden spaces, MIITA's correction directions and semantic anchors are model-specific. We study continual retention within a fixed deployment; migration to another backbone is outside this setting and would require rebuilding the bank from historical supervision or adding a representation-alignment or functional-distillation mechanism.

The effectiveness of MIITA also depends on the quality and coverage of the stored direction prototypes. If historical examples induce highly diverse or conflicting correction directions under a very small memory budget, memory repair may merge or evict directions that are useful for rare tasks. Similarly, retrieval relies on semantic anchors and an uncertainty-based label-free proxy, so out-of-domain queries, ambiguous prompts, poorly calibrated predictive uncertainty, or weak semantic representations may retrieve suboptimal directions and reduce the benefit of inference-time adaptation. The theoretical analysis is local: it relies on differentiability, first-order approximations around the current hidden state, fixed LM-head geometry, and smoothness assumptions, and therefore does not guarantee global performance improvements for large adaptation scales or across arbitrary distribution shifts.

Although MIITA avoids prompt extension, test-time backpropagation, and persistent backbone updates, it still introduces additional inference-time computation for memory retrieval, reranking, and per-token hidden-state gating. Exact retrieval scales with the number of stored anchors and prototypes, so very long streams or larger memory budgets may require approximate indexing or further compression to maintain low latency. Our experiments cover multiple supervised CL benchmarks, task types, model families, model scales, and memory budgets, but they do not exhaustively evaluate very long-horizon streams, safety-critical deployments, non-English domains beyond the tested multilingual QA setting, or all possible SLM architectures. Finally, storing compact correction directions reduces the amount of raw text retained but should not be interpreted as a formal privacy guarantee; if the training stream contains sensitive or biased supervision, additional auditing, privacy protection, and fairness evaluation would be needed before deployment.