InduceKV: Fixed-Footprint Continual Adaptation of Multimodal LLMs via Inducing KV Memories

Qianyu Chen[1] **Ziteng Feng**[2] **Runxuan Tang**[1] **Canran Xiao**[3*]
[1]Nanyang Technological University, Singapore
[2]University of Science and Technology of China
[3]Shenzhen Campus of Sun Yat-sen University, China
chen1981@e.ntu.edu.sg zitengfeng@mail.ustc.edu.cn
TANG0609@e.ntu.edu.sg xiaocr3@mail.sysu.edu.cn

Abstract

Multimodal large language models face evolving tasks and domains. Repeated parameter updates and growing replay stores can increase the deployed adaptation state over time. We study continual adaptation with a frozen backbone and a fixed memory budget for external task-specific updates. We propose INDUCEKV, a retrieval-based method that stores each selected training prefix as an attention-ready memory entry, consisting of a frozen retrieval key and compact layerwise key–value (KV) payloads that can be appended to the model's self-attention cache. A bilevel optimizer fits a lightweight retrieval calibration and selects a compact inducing set that balances current-task likelihood, anchor-based retention, and coverage in the frozen retrieval space under the memory budget. Across task-incremental instruction tuning, continual VQA, domain-incremental adaptation, and lifelong multimodal instruction tuning, INDUCEKV consistently improves over PEFT, MoE, replay, and prompt-retrieval baselines under matched memory budgets. The gains persist in backbone-matched, stage-1 CoIN, and compute-matched comparisons, including comparisons with replay, and under candidate shortlisting.

1 Introduction

Multimodal large language models (MLLMs) Yin et al. [2024] support visual understanding and instruction following. Applications include embodied reference grounding Li et al. [2023], continual VQA Jian et al. [2024], Jia et al. [2025], domain-specialized assistants Bian et al. [2025], and tool-augmented reasoning Hu et al. [2025] [see also Xiao et al., 2026c,b, for schema-grounded text agents]. New tasks, domains, and datasets arrive over time [see Zheng et al., 2026, for shifts across financial text sources]. Compute, privacy, and development constraints often make repeated training on all historical data impractical Huo and Tang [2025]. Continual adaptation requires retaining prior capabilities as new tasks arrive. We use the term "fixed footprint" for a bounded external adaptation state after each task.

Continual adaptation methods often update PEFT modules, route experts, or expand the architecture to balance stability and plasticity in sequential instruction tuning and continual VQA Guo et al. [2025], Huai et al. [2025], Zhang et al. [2025b], Ge et al. [2025], Zhang et al. [2025a], Wang et al. [2024b], Wei et al. [2025]. Growing expert or router state can complicate budget control, and parameter updates can affect the pretrained cross-modal interface that supports generalization. Replay reduces forgetting through synthetic or selective rehearsal, including dynamic data selection and memory-efficient replay or distillation for lifelong instruction tuning Maharana et al. [2025], Marouf et al. [2025], Wu et al. [2025]. Replay also helps threat attribution models retain prior knowledge

*Corresponding author: xiaocr3@mail.sysu.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

as new intelligence arrives Wang et al. [2026]. Replay depends on training-time data management. A large, redundant candidate pool makes it harder to preserve diverse historical behaviors within a small fixed budget. Studies of CLIP-style VLMs highlight the importance of preserving pretrained zero-shot capabilities in new domains Liu et al. [2025a], motivating care in choosing where and how to inject new information at inference.

We formulate continual adaptation as online construction of a compact, diverse inducing set under a fixed budget. The set improves current-task likelihood while retaining prior capabilities. Lightweight retrieval calibration controls how task-specific evidence enters the frozen backbone through its native attention pathway (Fig. 1).

Our contributions are as follows: (***i***) We formulate continual MLLM adaptation as budgeted online inducing-set selection, expressing the stability–plasticity trade-off as an allocation of retrieval memory. (***ii***) We introduce a retrieval-driven adaptation mechanism that stores task increments as attention-compatible external KV memories and uses a lightweight calibration interface to control retrieval strength, and accounts for the overhead of prefill and KV injection. (***iii***) Our approach achieves consistent state-of-the-art results across diverse continual instruction-tuning and continual VQA settings.

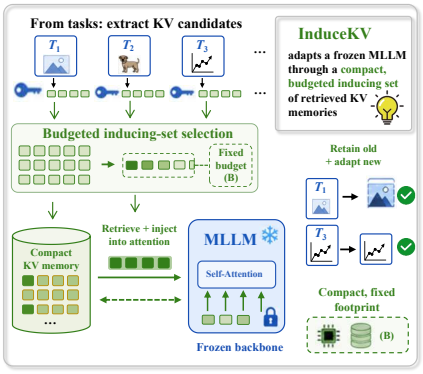


Figure 1: **INDUCEKV for budgeted continual MLLM adaptation.**

2 Related Work

Continual adaptation of instruction-following MLLMs. Most continual multimodal instruction tuning and VQA methods adapt models by *updating parameters* via PEFT and modularization. Representative directions include dynamic data selection/pruning for lifelong instruction tuning Maharana et al. [2025], PEFT-based isolation and routing such as hierarchical layer decoupling Guo et al. [2025], mixtures of LoRA experts Ge et al. [2025], Wang et al. [2024b], Zhang et al. [2025a], and projector-centric adaptation Jin et al. [2025]. For continual VQA, MoE routing and replay/distillation (e.g., dual-router momentum MoE and question-only replay with attention distillation) achieve strong stability–plasticity trade-offs Huai et al. [2025], Marouf et al. [2025], while domain-incremental settings often use domain-specific experts and routing Zhang et al. [2025b]. For continual video–language understanding, AFD pairs stable affordance tokens with query-routed adapter updates Xu et al. [2026]. These methods require ongoing gradient updates and often rely on growing expert or router state or on rehearsal. Parameter-space constraints on retention can limit plasticity under tight budgets. INDUCEKV injects retrieved KV payloads into the masked self-attention cache of a frozen backbone and learns a lightweight per-stream calibration.

Continual learning in CLIP-style VLMs. Pretrained vision–language models can supply language anchors for unsupervised road anomaly detection Tian et al. [2023]. Recent CLIP-style continual learning methods combine adapters/regularization with consolidation objectives and new benchmarks for domain/class shifts Liu et al. [2025a,b], and mitigate forgetting via synthetic replay/distillation Wu et al. [2025], structured adapters or memory units Luo et al. [2025], modality-gap constraints Huang et al. [2025b], external knowledge injection Zhou et al. [2025], and feature-geometry preserving regularization Huang et al. [2025a]. COMPO-REALIGN uses reversible alignment and a spectral trust region to preserve compositional structure Xiao et al. [2026a]. In autoregressive MLLMs, layerwise attention and long-range cache interactions govern how stored information affects decoding. We store layerwise KV payloads from the frozen generator for direct injection into self-attention.

Retrieval-augmented memory and budgeted subset selection. Retrieval-augmented LMs use non-parametric memory via passage retrieval Lewis et al. [2020], Borgeaud et al. [2022] or kNN retrieval over hidden states/KV pairs Khandelwal et al. [2019], Wu et al. [2022]. In continual web personalization, ProtoFed-SP retrieves shared prototypes to compose soft prompts for a frozen backbone Xiao and Hou [2026]. Coreset and diversity selection study coverage under budgets, often via log-det/DPP-style objectives Kulesza et al. [2012], Wang et al. [2024a], Zhang et al. [2025c]. FastMAC provides an example of spectral subset selection in point-cloud registration Zhang et al.

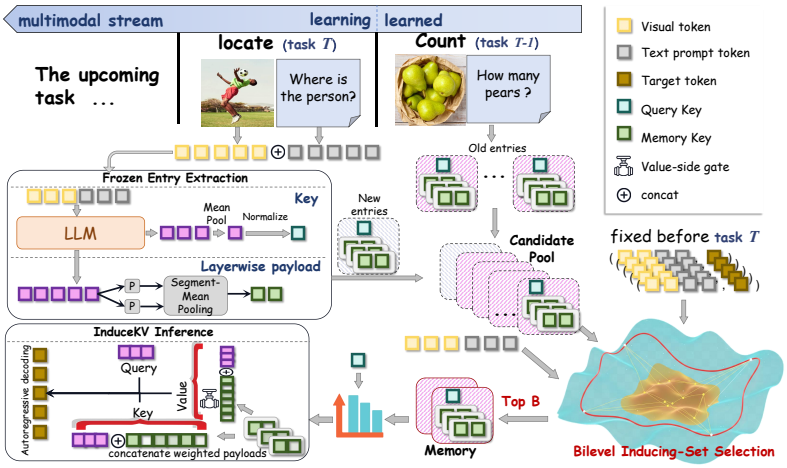


Figure 2: **INDUCEKV pipeline for continual adaptation.** For each incoming task t, the frozen MLLM extracts from each prefix x a unit-norm retrieval key $r(x)$ and compressed layerwise KV payloads $\{(\bar{K}^\ell(x), \bar{V}^\ell(x))\}_{\ell=1}^{L}$, forming new entries that are merged with the previous memory into a candidate pool $\mathcal{U}_t$. Under a fixed budget B, a bilevel optimizer constructs a compact inducing set: the inner level fits a minimal retrieval calibration $\phi^\star(w)$ on current-task data, while the outer level updates selection weights w to jointly optimize current-task likelihood, retention on historical anchors $\mathcal{A}_{<t}$, and a redundancy-aware spectral coverage regularizer in the frozen retrieval space. The Top-B entries are committed as the external KV memory $\mathcal{M}_t$. At inference, the query key retrieves weighted entries, whose KV payloads are concatenated and injected as additional cache tokens into masked self-attention, enabling task adaptation without updating θ.

[2024]. Budgeted subset selection also serves as a tool for stress-testing maximum-likelihood pairwise rankings Yao et al. [2026]. Most retrieval systems optimize query-time evidence from a static corpus. Fixed-footprint continual adaptation requires updating the corpus, and existing subset-selection methods rarely combine retrieval calibration with direct attention injection. INDUCEKV fills this gap by formulating memory updates as online inducing-set selection: a lightweight inner calibration adapts retrieval/gating, while the outer loop selects a budgeted subset that balances current fit, anchor-based retention, and log-det spectral coverage.

3 Method

INDUCEKV performs continual adaptation of a multimodal large language model (MLLM) by freezing all backbone parameters and externalizing task-specific increments into an attention-compatible key–value (KV) memory. Each memory entry stores (i) a frozen-model retrieval key in a fixed representation space and (ii) compact, layerwise KV payloads that can be injected into the Transformer self-attention cache. Under a strict memory budget, we construct a compact inducing set via bilevel optimization: an inner problem fits a minimal retrieval calibration parameter on the current task, while an outer problem selects a budgeted subset that jointly optimizes current-task likelihood, retention on historical anchors, and a redundancy-aware spectral coverage regularizer in the frozen retrieval space. The pipeline is shown in Fig. 2.

3.1 Problem Setup

We consider tasks $t = 1, \ldots, T$ arriving sequentially. At task t, we observe $\mathcal{D}_t = \{(x_n, y_n)\}_{n=1}^{N_t}$, where x is the *conditioning prefix* (multimodal input and prompt) and y is the target output sequence. Let θ denote the MLLM parameters. We freeze θ throughout. We maintain a small historical anchor set $\mathcal{A}_{<t}$ (fixed before training task t) used to measure retention.

Table 1: Adaptation-state accounting. The frozen backbone and ordinary query cache are shared inference costs.

State	Used at inference?	Bound / lifetime
KV memory and calibration	Yes	Fixed deployed budget
Historical anchors	No	64 per past task by default
Candidates and optimizer workspace	No	Transient during updates

Autoregressive objective. For a pair (x, y), the negative log-likelihood is

$$\mathcal{L}_{\mathrm{NLL}}(\theta;\, x, y) = -\sum_{m=1}^{|y|} \log p_\theta(y_m \mid y_{<m}, x)\,. \tag{1}$$

In (1), $p_\theta(\cdot)$ is the frozen model distribution, and x excludes all answer tokens to prevent label leakage in memory construction.

Multi-head attention convention. We write self-attention for one head, with the head index implicit. Let d_h denote the per-head key dimension; the same construction is applied independently to each head and each layer.

The footprint budget applies to deployed adaptation state (Table 1). Default update-time anchors grow with task count; Appendix E.6 also evaluates globally capped anchors.

3.2 Frozen Key and KV-Payload Extraction

Each stored entry contains a retrieval key in a fixed representation space and compact KV payloads that can be appended to the self-attention cache while keeping θ fixed. Freezing the backbone avoids potential changes to cross-modal interactions from parameter-efficient fine-tuning.

Given a prefix input x, let $H^{(L)}(x) \in \mathbb{R}^{n_p(x)\times d}$ be the final-layer hidden states corresponding to the prefix tokens (length $n_p(x)$, hidden size d). We define the pooled prefix representation and normalized retrieval key as

$$\bar{h}(x) = \frac{1}{n_p(x)} \sum_{j=1}^{n_p(x)} H_j^{(L)}(x) \in \mathbb{R}^d, \qquad r(x) = \frac{\bar{h}(x)}{\|\bar{h}(x)\|_2} \in \mathbb{R}^d. \tag{2}$$

In (2), $H_j^{(L)}(x)$ is the j-th prefix token vector, and $r(x)$ is unit-norm, enabling cosine-similarity retrieval.

Layerwise KV payload. For each Transformer layer $\ell \in \{1, \ldots, L\}$, let $H^{(\ell-1)}(x) \in \mathbb{R}^{n_p(x)\times d}$ be the layer input over prefix tokens. Using the frozen per-head projections $W_K^\ell \in \mathbb{R}^{d\times d_h}$ and $W_V^\ell \in \mathbb{R}^{d\times d_h}$, we compute prefix keys/values and compress them to a fixed payload length m:

$$\begin{aligned} K^\ell(x) &= \mathrm{Pos}_x^\ell\Big(H^{(\ell-1)}(x)W_K^\ell\Big) \in \mathbb{R}^{n_p(x)\times d_h}, & V^\ell(x) &= H^{(\ell-1)}(x)W_V^\ell \in \mathbb{R}^{n_p(x)\times d_h}, \\ \bar{K}^\ell(x) &= P_m(x)K^\ell(x), & \bar{V}^\ell(x) &= P_m(x)V^\ell(x). \end{aligned} \tag{3}$$

Here, Pos_x^ℓ denotes the backbone's native cache-key positional transformation. For RoPE backbones, we extract source-position post-RoPE keys and pool them after rotation; injected keys are reused without a second rotation, while current-token positions remain native and unshifted. The position controls are reported in Appendix E.4. In (3), $P_m(x) \in \mathbb{R}^{m\times n_p(x)}$ is a fixed pooling matrix that partitions the prefix indices $\{1, \ldots, n_p(x)\}$ into m consecutive segments of equal size (the last segment absorbs any remainder) and averages within each segment; therefore $\bar{K}^\ell(x), \bar{V}^\ell(x) \in \mathbb{R}^{m\times d_h}$ have a task-independent token length m.

Memory entry. We represent the extracted entry from x as $e(x) = \big(r(x), \{(\bar{K}^\ell(x), \bar{V}^\ell(x))\}_{\ell=1}^L\big)$, and store only such entries in the external memory.

3.3 Retrieval-Weighted KV Induction and Masked Attention Injection

At inference, INDUCEKV appends retrieved KV payloads as cache tokens in each layer's masked self-attention. Let $\mathcal{M} = \{e_i\}_{i=1}^{|\mathcal{M}|}$ be the current memory, with stored keys $r_i \in \mathbb{R}^d$. We learn a minimal calibration parameter $\phi = \{\phi_\tau, \phi_1, \ldots, \phi_L\}$, where the retrieval temperature is $\tau = \mathrm{softplus}(\phi_\tau) > 0$ and the per-layer value gate is $\lambda_\ell = \sigma(\phi_\ell) \in (0, 1)$. Given a prefix x, we compute $r(x)$ by (2) and define the retrieval weight for entry i as

$$\alpha_i(x; \phi) = \frac{\exp\big(\langle r(x), r_i\rangle/\tau\big)}{\sum_{j=1}^{|\mathcal{M}|} \exp\big(\langle r(x), r_j\rangle/\tau\big)}. \tag{4}$$

where $\langle r(x), r_i\rangle$ is cosine similarity since both keys are unit-norm, and $\sum_i \alpha_i(x; \phi) = 1$.

For layer ℓ, we assemble memory KV tokens by concatenating payload blocks weighted by $\sqrt{\alpha_i(x;\phi)}$: $K^\ell_{\mathrm{mem}}(x) = \mathrm{Concat}\big(\sqrt{\alpha_i(x;\phi)}\,\bar{K}^\ell_i\big)_{i=1}^{|\mathcal{M}|} \in \mathbb{R}^{(|\mathcal{M}|m)\times d_h}$ and $V^\ell_{\mathrm{mem}}(x) = \mathrm{Concat}\big(\sqrt{\alpha_i(x;\phi)}\,\bar{V}^\ell_i\big)_{i=1}^{|\mathcal{M}|} \in \mathbb{R}^{(|\mathcal{M}|m)\times d_h}$, where $\mathrm{Concat}(\cdot)$ concatenates along the token dimension. Let $Q^\ell(x) \in \mathbb{R}^{n_p(x)\times d_h}$, $K^\ell_{\mathrm{self}}(x) \in \mathbb{R}^{n_p(x)\times d_h}$, and $V^\ell_{\mathrm{self}}(x) \in \mathbb{R}^{n_p(x)\times d_h}$ be the standard frozen per-head tensors produced at layer ℓ for the prefix tokens. We inject memory KV as additional cache tokens and compute masked self-attention as

$$\begin{aligned} A^\ell(x) &= \frac{Q^\ell(x)\big[\,K^\ell_{\mathrm{self}}(x);\; K^\ell_{\mathrm{mem}}(x)\,\big]^\top + M^\ell(x)}{\sqrt{d_h}} \\ \mathrm{Attn}^\ell(x) &= \mathrm{softmax}\big(A^\ell(x)\big)\,\big[\,V^\ell_{\mathrm{self}}(x);\; \lambda_\ell V^\ell_{\mathrm{mem}}(x)\,\big]. \end{aligned} \tag{5}$$

In (5), $[\cdot;\cdot]$ denotes token-wise concatenation and $M^\ell(x)$ is the extended causal mask that permits attention to all memory tokens while enforcing standard causal visibility among prefix tokens. The gate λ_ℓ scales only memory values, providing an interpretable per-layer control of memory strength.

Online inference path and cost. At test time, INDUCEKV uses a two-stage inference path. First, it runs a prefix-only forward pass with the frozen backbone to compute the query retrieval key $r(x)$. Second, after retrieving entries from the fixed memory $\mathcal{M}$, it performs the generation forward pass with the selected precomputed KV payloads appended to the self-attention cache. The stored memory payloads $\{(\bar{K}^\ell_i, \bar{V}^\ell_i)\}$ are extracted offline at task-update time and are never recomputed for each test query. The extra prefix pass for the retrieval key and the larger attention matrices during prefill account for the overhead relative to replay-free PEFT. The efficiency advantage is clearest against prompt-level retrieval, which processes retrieved exemplars through embeddings, projections, attention, and FFN layers at inference.

3.4 Bilevel Inducing-Set Selection Under a Fixed Budget

Storing all extracted entries is redundant under a fixed budget. INDUCEKV therefore builds a compact inducing set that fits the current task via minimal calibration while preserving performance on historical anchors, and enforces spectral coverage in the frozen retrieval space to avoid near-duplicate keys.

Candidate pool and selection variables. At task t, we build a candidate pool $\mathcal{U}_t = \mathcal{C}_t \cup \mathcal{M}_{t-1}$, where $\mathcal{C}_t$ contains entries extracted from prefixes in $\mathcal{D}_t$ and $\mathcal{M}_{t-1}$ is the previous memory. Let $N = |\mathcal{U}_t|$ and index candidates by $i \in \{1, \ldots, N\}$. We introduce continuous selection weights $w \in \mathbb{R}^N_{\geq 0}$ constrained by $\sum_{i=1}^N w_i = B$, where B is the fixed entry budget. We interpret $\pi_i(w) = w_i/B$ as a fractional inclusion mass with $\sum_i \pi_i(w) = 1$.

Retrieval during selection. While optimizing w, we define retrieval weights over candidates by incorporating $\pi_i(w)$ as a multiplicative prior: $\alpha_i(x; w, \phi) = \frac{\pi_i(w)\exp(\langle r(x), r_i\rangle/\tau)}{\sum_{j=1}^N \pi_j(w)\exp(\langle r(x), r_j\rangle/\tau)}$. This yields a differentiable dependence of induced KV on w.

Algorithm 1 INDUCEKV update at task t

Input: $\mathcal{D}_t$, $\mathcal{A}_{<t}$, $\mathcal{M}_{t-1}$, budget B
Build candidates $\mathcal{U}_t = \mathcal{C}_t \cup \mathcal{M}_{t-1}$; extract entries via (2)–(3)
Initialize $w \geq 0$ with $\sum_i w_i = B$ and initialize ϕ
for iter $= 1$ to I **do**
  for $j = 1$ to J **do**
    $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \Big(\frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\mathrm{NLL}}(\theta;\, x, y \mid \mathcal{U}_t, w, \phi) + \eta_{\mathrm{reg}} \|\phi\|_2^2 \Big)$
  end for
  $w \leftarrow w - \eta_{\mathrm{pgd}} \nabla_w \Big(\mathcal{L}_{\mathrm{cur}}(w, \phi) + \beta \mathcal{L}_{\mathrm{anc}}(w, \phi) + \gamma \Omega_{\mathrm{spec}}(w) \Big)$
  Project w onto $\{w \geq 0 : \sum_i w_i = B\}$
end for
$\mathcal{S}_t \leftarrow \mathrm{TopB}(w), \quad \mathcal{M}_t \leftarrow \{e_i\}_{i\in\mathcal{S}_t}, \quad \phi_t \leftarrow \phi$
Output: memory $\mathcal{M}_t$, calibration ϕ_t

Inner problem (fit minimal calibration on current task). Given w, the inner level fits ϕ by minimizing current-task likelihood with ℓ_2 regularization:

$$\phi^\star(w) = \arg\min_{\phi} \; \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\mathrm{NLL}}\big(\theta;\, x, y \mid \mathcal{U}_t, w, \phi\big) + \eta_{\mathrm{reg}} \|\phi\|_2^2. \tag{6}$$

In (6), the conditional notation $\mathcal{L}_{\mathrm{NLL}}(\cdot \mid \mathcal{U}_t, w, \phi)$ means that the model uses candidate-based retrieval weights $\alpha_i(x; w, \phi)$ and injects the induced KV via (5). The ℓ_2 regularization coefficient $\eta_{\mathrm{reg}} > 0$ is fixed.

Outer problem (select inducing set with retention and coverage). The outer level selects w by balancing current fit, anchor retention, and spectral coverage:

$$w^\star = \arg\min_{w\in\mathcal{W}} \; J(w), \tag{7}$$

where $\mathcal{W} = \{w \in \mathbb{R}^N_{\geq 0} : \sum_i w_i = B\}$, and the objective is

$$J(w) = \underbrace{\mathcal{L}_{\mathrm{cur}}\big(w, \phi^\star(w)\big)}_{\text{current fit}} + \beta \underbrace{\mathcal{L}_{\mathrm{anc}}\big(w, \phi^\star(w)\big)}_{\text{retention}} + \gamma \underbrace{\Omega_{\mathrm{spec}}(w)}_{\text{coverage}}. \tag{8}$$

where $\beta \geq 0$ and $\gamma \geq 0$ are fixed coefficients, $\mathcal{L}_{\mathrm{cur}}(w, \phi) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\mathrm{NLL}}(\theta;\, x, y \mid \mathcal{U}_t, w, \phi)$, and $\mathcal{L}_{\mathrm{anc}}(w, \phi) = \frac{1}{|\mathcal{A}_{<t}|} \sum_{(x,y)\in\mathcal{A}_{<t}} \mathcal{L}_{\mathrm{NLL}}(\theta;\, x, y \mid \mathcal{U}_t, w, \phi)$.

Spectral coverage regularizer. Let $r_i \in \mathbb{R}^d$ be the unit-norm retrieval key of candidate i. We fix a projection matrix $P \in \mathbb{R}^{d\times d'}$ with orthonormal columns ($P^\top P = I_{d'}$) and set $d' = 256$. Define projected keys $z_i = P^\top r_i \in \mathbb{R}^{d'}$ and the weighted covariance $C(w) = \sum_{i=1}^N \pi_i(w)\, z_i z_i^\top \in \mathbb{R}^{d'\times d'}$. We use a log-determinant coverage penalty

$$\Omega_{\mathrm{spec}}(w) = -\log\det\big(C(w) + \epsilon I_{d'}\big), \tag{9}$$

where $\epsilon > 0$ is a fixed stabilizer and $I_{d'}$ is the $d' \times d'$ identity. Minimizing (9) discourages redundant keys by penalizing low-rank, spectrally concentrated $C(w)$ in the frozen retrieval space.

We solve (6)–(7) by unrolling a fixed number of inner steps for ϕ and applying projected gradient updates to w onto $\{w \geq 0 : \sum_i w_i = B\}$. After convergence, we take $\mathcal{S}_t = \mathrm{TopB}(w^\star)$ and set $\mathcal{M}_t = \{e_i\}_{i\in\mathcal{S}_t}$. For later tasks, $\mathcal{M}_t$ is fixed, and we optimize only the new task's (w, ϕ).

4 Theory: Budgeted Continual Learning as Online Inducing-Set Selection

Once the inner calibration is solved, the outer memory update in INDUCEKV is an online budgeted subset-selection problem over relaxed inducing-set weights. Let $f_t(w)$ denote the inner-solved

outer surrogate at task t, defined on the budget simplex $\Delta_B^N = \{w \geq 0 : \sum_i w_i = B\}$, and let $D = \max_{u,v\in\Delta_B^N} \|u - v\|_2 = B\sqrt{2}$. With step size $\eta_{\mathrm{pgd}} > 0$, the projected outer update is $w_{t+1} = \Pi_{\Delta_B^N}(w_t - \eta_{\mathrm{pgd}}\nabla f_t(w_t))$, followed by the TopB discretization used in Alg. 1. Full definitions, assumptions, and proofs are provided in Appendix A. Under the stated convexity and bounded-gradient assumptions, the guarantees apply to the inner-solved relaxed surrogate before discrete Top-B deployment. To compare candidate pools across tasks, we embed them in a common candidate universe and assign zero weight to absent entries. Appendix C.15 measures the empirical deployment gap.

Budgeted selection has sublinear regret. The comparator is the best fixed inducing-set allocation in hindsight under the same budget.

Theorem 4.1 (Static regret under fixed budget)**.** *Assume each inner-solved surrogate f_t is convex and has bounded gradients $\|\nabla f_t(w)\|_2 \leq G$ on Δ_B^N. Then for any comparator $u \in \Delta_B^N$, the projected outer updates satisfy*

$$\mathrm{Reg}_T(u) \triangleq \sum_{t=1}^{T} \big(f_t(w_t) - f_t(u)\big) \leq \frac{D^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}} G^2 T}{2}, \tag{10}$$

$$\mathrm{Reg}_T(u) \leq DG\sqrt{T} = B\sqrt{2}\,G\sqrt{T} \qquad \textit{when } \eta_{\mathrm{pgd}} = \frac{D}{G\sqrt{T}}. \tag{11}$$

The bound adapts to nonstationary task streams. If the best inducing-set allocation changes over time, the regret increases only with the path-length variation of the changing comparator.

Theorem 4.2 (Dynamic regret with task variation)**.** *Let $\{u_t\}_{t=1}^T \subset \Delta_B^N$ be any time-varying comparator sequence and $V_T = \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2$. Under the same bounded-gradient condition,*

$$\mathrm{DReg}_T(\{u_t\}) \triangleq \sum_{t=1}^{T} \big(f_t(w_t) - f_t(u_t)\big) \lesssim \frac{D^2 + DV_T}{\eta_{\mathrm{pgd}}} + \eta_{\mathrm{pgd}} G^2 T, \tag{12}$$

$$\mathrm{DReg}_T(\{u_t\}) \lesssim G\sqrt{T}\sqrt{D^2 + DV_T} \quad \textit{with the optimized step size.} \tag{13}$$

Anchors control historical retention. The anchor term links the online surrogate to retention: if anchors approximate the historical risk, then controlling the anchor loss controls forgetting up to the anchor approximation error.

Theorem 4.3 (Anchor-controlled historical risk)**.** *Assume the anchor loss $\mathcal{A}_t(w)$ uniformly approximates the true historical risk $\widetilde{\mathcal{R}}_{<t}(w)$ within $\epsilon_t^{\mathrm{rep}}$. Then for every task t,*

$$\widetilde{\mathcal{R}}_{<t}(w_t) \leq \mathcal{A}_t(w_t) + \epsilon_t^{\mathrm{rep}}, \tag{14}$$

$$\frac{1}{T}\sum_{t=1}^{T} \widetilde{\mathcal{R}}_{<t}(w_t) \leq \frac{1}{\beta T}\sum_{t=1}^{T} f_t(u) + \frac{\mathrm{Reg}_T(u)}{\beta T} + \frac{1}{T}\sum_{t=1}^{T} \epsilon_t^{\mathrm{rep}}, \tag{15}$$

for any $u \in \Delta_B^N$ and $\beta > 0$.

The relaxed selector achieves sublinear regret against a fixed comparator. Its dynamic regret bound scales with comparator variation, and the anchor term links the surrogate objective to historical retention. The final TopB step is a rounding operation from the relaxed selector to the actual memory; its additional gap is controlled when the surrogate is Lipschitz, as discussed in Appendix A.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate INDUCEKV on (i) task-incremental continual instruction tuning benchmarks COIN and UCIT Tang et al. [2019], Guo et al. [2025], (ii) continual VQA streams on VQAV2 (10 question-type tasks) and VQACL Huai et al. [2025], Marouf et al. [2025], (iii) domain-incremental

Table 2: **Task-incremental continual instruction tuning on UCIT and CoIN.** Baseline results follow the HiDe-LLaVA protocol Guo et al. [2025].

Method	Backbone / Protocol	UCIT Avg↑	UCIT Last↑	CoIN Avg↑	CoIN Last↑
Zero-shot (lower bound)	LLaVA-v1.5-7B	31.68	–	54.21	–
Multi-task (upper bound)	LLaVA-v1.5-7B	74.78	–	67.40	–
FineTune	LLaVA-v1.5-7B	57.52	48.12	52.86	47.31
LwF	LLaVA-v1.5-7B	59.40	49.52	53.22	48.40
EWC	LLaVA-v1.5-7B	59.34	50.20	53.30	48.95
L2P	LLaVA-v1.5-7B	53.99	48.51	53.96	52.83
O-LoRA	LLaVA-v1.5-7B	64.54	58.36	62.60	60.77
MoELoRA	LLaVA-v1.5-7B	61.33	52.06	55.24	50.58
HiDe-LLaVA	LLaVA-v1.5-7B	68.94	64.19	64.70	63.95
INDUCEKV	LLaVA-v1.5-7B	**69.82**	**65.31**	**66.05**	**65.38**
INDUCEKV	LLaVA-OV-4B	**70.25**	**65.84**	**66.38**	**65.72**

Table 3: **Continual VQA under LV and T5 protocols. LV**: LLaVA/Vicuna-7B protocol for the VQAv2 10-task stream; **T5**: T5-based VQACL protocol for VQACL; **OV**: our default LLaVA-OneVision-1.5-4B-Instruct protocol. Rows marked **LV/T5** use LV for VQAv2 10-task and T5 for VQACL columns.

		VQAv2 10-task		**VQACL: VQAv2 (Std.)**		**VQACL: NExT-QA (Std.)**	
Method	Prot.	AP↑	AF↓	AP↑	Forget↓	AP↑	Forget↓
Vanilla	LV/T5	32.51	20.69	14.92	30.80	12.68	25.94
EWC	LV/T5	37.28	15.27	15.77	30.62	13.01	24.06
MAS	LV/T5	37.71	14.91	20.56	11.16	18.04	10.07
ER	LV/T5	41.95	10.20	36.99	5.99	30.55	4.91
DER	LV/T5	41.16	11.28	35.35	8.62	26.17	5.12
VS	LV/T5	39.79	12.70	34.03	8.79	28.13	4.45
VQACL	LV/T5	43.49	9.10	37.46	6.96	30.86	4.12
CL-MoE	LV	51.34	-0.02	–	–	–	–
QUAD	T5	–	–	39.25	4.91	31.70	2.91
INDUCEKV-LV	LV	**52.64**	**1.70**	–	–	–	–
INDUCEKV-T5	T5	–	–	**39.86**	**4.54**	**32.12**	**2.68**
INDUCEKV-OV	OV	**53.12**	**1.85**	**40.73**	**3.64**	**33.25**	**2.18**

continual instruction tuning across MEDICINE/CHART/MATH Zhang et al. [2025b], and (iv) lifelong multimodal instruction tuning streams (LiIT) under dynamic dataset arrival Maharana et al. [2025]. Full dataset composition, task orders, and preprocessing follow the official protocols and are summarized in Appendix B.1.

Metrics. We follow the official evaluation of each suite: UCIT reports *Avg* and *Last* Guo et al. [2025]; CoIN reports task scores with overall *MAA* and *BWT* (for both instruction-following and reasoning-style evaluation) Tang et al. [2019]; continual VQA reports *AP* and *AF/Forget* Huai et al. [2025], Marouf et al. [2025]; domain-incremental CIT uses domain/task-specific metrics (e.g., RelaxAcc/RMSF1/BLEU4/Acc) Zhang et al. [2025b]; LiIT reports *AvgAcc*, *Relative Gain*, and *Forgetting Rate* Maharana et al. [2025]. We additionally report memory footprint and inference overhead (Appendix B.2).

Compared Methods. We compare against recent continual MLLM adaptation methods spanning replay/regularization, PEFT isolation, and routing-based architectures, including CL-MoE Huai et al. [2025], QUAD Marouf et al. [2025], SMoE Zhang et al. [2025b], HiDe-LLaVA Guo et al. [2025], Adapt-∞ Maharana et al. [2025], and others Ge et al. [2025], Chen et al. [2025], Zhang et al. [2025a], Wang et al. [2024b], Wu et al. [2025]. We also include standard continual learning baselines. We match the *total extra footprint* (external memory, replay buffers, or added parameters) and keep inference-time compute comparable. Appendix B.3 details the budget matching.

Table 4: **Single-factor ablation of INDUCEKV.** Each row removes exactly one component from the full model under the same memory budget (B=256, m=8). Numbers in blue denote the drop relative to the full INDUCEKV model.

Variant	UCIT Avg↑	CoIN Avg↑	VQAv2 AP↑	Domain Overall↑	LiIT AvgAcc↑
INDUCEKV (full)	70.25	66.38	53.12	51.14	53.8
w/o **bilevel coupling** (single-level update)	68.97 (-1.28)	64.95 (-1.43)	51.76 (-1.36)	49.91 (-1.23)	52.4 (-1.4)
w/o **anchor retention** (β=0)	68.52 (-1.73)	64.41 (-1.97)	52.64 (-0.48)	50.55 (-0.59)	53.2 (-0.6)
w/o **spectral coverage** (γ=0)	69.31 (-0.94)	65.02 (-1.36)	52.28 (-0.84)	50.12 (-1.02)	52.9 (-0.9)
w/o **inner calibration** (fix τ=0.07, λ_ℓ=0.5)	67.40 (-2.85)	63.88 (-2.50)	49.63 (-3.49)	48.27 (-2.87)	51.5 (-2.3)
w/o **layerwise value gates** (set $\lambda_\ell \equiv 1$)	69.88 (-0.37)	65.74 (-0.64)	52.61 (-0.51)	50.76 (-0.38)	53.1 (-0.7)
w/o **retrieval-weighted induction** (uniform α_i over selected entries)	66.95 (-3.30)	62.73 (-3.65)	48.92 (-4.20)	47.50 (-3.64)	50.9 (-2.9)

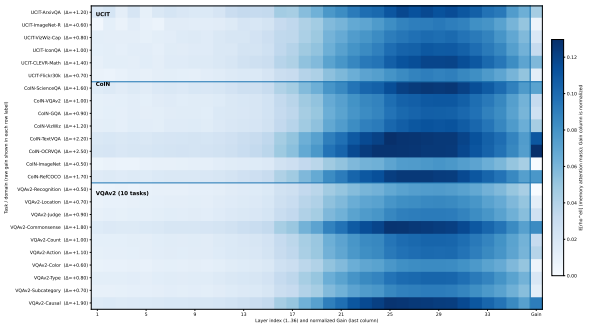


Figure 3: **Memory attention utilization.** Rows are tasks/domains and columns are layers; heatmap values show memory-attention mass, while the last column reports normalized gain over NO-MEM.

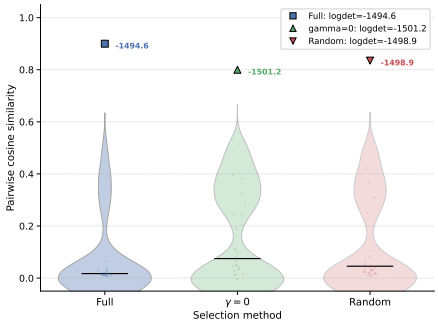


Figure 4: **Inducing-set diversity.** Violins show within-set cosine similarity. Diamonds mark scaled mean log-det coverage for each selection rule.

5.2 Main Results

Task-incremental continual instruction tuning. As shown in Table 2, INDUCEKV consistently improves over PEFT- and routing-based continual adaptation baselines on both UCIT and COIN. Under the backbone-matched LLaVA-v1.5-7B protocol, INDUCEKV improves over HiDe-LLaVA by 0.88 Avg and 1.12 Last on UCIT, and by 1.35 Avg and 1.43 Last on COIN. The default LLaVA-OneVision-4B variant further increases the gains to 1.31/1.65 on UCIT and 1.68/1.77 on COIN. These improvements indicate that storing task increments as budgeted, attention-compatible KV memories can improve both average retained performance and final-task adaptation without repeatedly updating the backbone.

Continual VQA. Table 3 shows that INDUCEKV achieves consistent gains across both continual VQA protocols. Under the LV protocol, INDUCEKV-LV improves AP over CL-MoE from 51.34 to 52.64 while maintaining low forgetting. Under the T5-VQACL protocol, INDUCEKV-T5 outperforms QUAD on both VQAv2 and NExT-QA, with higher AP and lower forgetting.

Additional results on domain-incremental continual instruction tuning and lifelong multimodal instruction tuning are provided in Appendix C.1.

5.3 Ablation Study and Mechanism Analysis

Single-factor ablation. We ablate INDUCEKV under the same backbone, decoding, and footprint. Table 4 shows that each component matters: removing retrieval-weighted induction hurts most (up to 4.20 AP on VQAv2), and disabling inner calibration also causes consistent drops, especially in generation-heavy settings. Anchor retention is most important for long-horizon task-incremental CIT (UCIT/CoIN), while spectral coverage improves robustness by reducing redundancy under a fixed budget. Removing bilevel coupling consistently degrades performance, underscoring the need to jointly optimize selection weights and retrieval calibration; additional sensitivity results are in §C.2.

Is the memory actually used? We measure the fraction of attention assigned to injected memory tokens across layers and tasks and compare it with task-level gains over NO-MEM. The metric and sampling protocol are defined in Appendix B.5, Memory-attention utilization protocol. As

shown in Fig. 3, memory attention concentrates in mid-to-late layers and increases from early layers (≈ 0.018) to late layers (≈ 0.094). Tasks with larger gains also exhibit stronger late-layer memory usage, suggesting that INDUCEKV improves performance when the model actually routes generation through retrieved KV payloads.

Does inducing-set selection reduce redundancy? We assess inducing-set diversity under a fixed budget by comparing full INDUCEKV, the no-coverage variant (γ=0), and random Top-B selection. The diagnostics are within-set pairwise cosine similarity and projected log-determinant coverage, defined in Appendix B.5, Inducing-set diversity protocol. Fig. 4 shows that the full objective yields lower pairwise similarity and higher log-det coverage than both alternatives. This pattern indicates that the spectral coverage term limits concentration in dense retrieval modes and promotes a higher-rank memory that better supports retention.

6 Conclusion

INDUCEKV adapts frozen MLLMs to sequential tasks by selecting attention-compatible KV entries for a fixed-budget external memory. Our empirical and theoretical results support online inducing-set selection as a principled alternative to parameter-updating pipelines.

Acknowledgments and Disclosure of Funding

Funding. The authors received no third-party funding or support for this work.

Competing interests. The authors declare no competing interests.

References

Zhipeng Bian, Jieming Zhu, Xuyang Xie, Quanyu Dai, Zhou Zhao, and Zhenhua Dong. Mira: Empowering one-touch ai services on smartphones with mllm-based instruction recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 1457–1465, 2025.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

Jinpeng Chen, Runmin Cong, Yuzhi Zhao, Hongzheng Yang, Guangneng Hu, Horace Ip, and Sam Kwong. Sefe: Superficial and essential forgetting eliminator for multimodal continual instruction tuning. In *Forty-second International Conference on Machine Learning*, 2025.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic mixture of curriculum lora experts for continual multimodal instruction tuning. In *Forty-second International Conference on Machine Learning*, 2025.

Haiyang Guo, Fanhu Zeng, Ziwei Xiang, Fei Zhu, Da-Han Wang, Xu-Yao Zhang, and Cheng-Lin Liu. Hide-llava: Hierarchical decoupling for continual instruction tuning of multimodal large language model. *arXiv preprint arXiv:2503.12941*, 2025.

Saisai Hu. Research on security enhancement methods for adversarial robust large language model intelligent agents for medical decision-making tasks. In *2026 7th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 599–603. IEEE, 2026.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, et al. Os agents: A survey on mllm-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465, 2025.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. Cl-moe: Enhancing multimodal large language model with dual momentum mixture-of-experts for continual visual question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19608–19617, 2025.

Chen Huang, Skyler Seto, Hadi Pouransari, Mehrdad Farajtabar, Raviteja Vemulapalli, Fartash Faghri, Oncel Tuzel, Barry-John Theobald, and Joshua M Susskind. Proxy-fda: Proxy-based feature distribution alignment for fine-tuning vision foundation models without forgetting. In *Forty-second International Conference on Machine Learning*, 2025a.

Linlan Huang, Xusheng Cao, Haori Lu, Yifan Meng, Fei Yang, and Xialei Liu. Mind the gap: Preserving and compensating for the modality gap in clip-based continual learning. *arXiv preprint arXiv:2507.09118*, 2025b.

Yukang Huo and Hao Tang. When continue learning meets multimodal large language model: A survey. *arXiv preprint arXiv:2503.01887*, 2025.

Ziheng Jia, Zicheng Zhang, Jiaying Qian, Haoning Wu, Wei Sun, Chunyi Li, Xiaohong Liu, Weisi Lin, Guangtao Zhai, and Xiongkuo Min. Vqa2: visual question answering for video quality assessment. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 6751–6760, 2025.

Pu Jian, Donglei Yu, and Jiajun Zhang. Large language models know what is key visual entity: An llm-assisted multimodal retrieval for vqa. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10939–10956, 2024.

Hyundong Jin, Hyung Jin Chang, and Eunwoo Kim. Instruction-grounded visual projectors for continual learning of generative vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3466–3476, 2025.

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

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Yang Li, Xiaoxue Chen, Hao Zhao, Jiangtao Gong, Guyue Zhou, Federico Rossano, and Yixin Zhu. Understanding embodied reference with Touch-Line Transformer. In *International Conference on Learning Representations*, 2023.

Wenzhuo Liu, Fei Zhu, Longhui Wei, and Qi Tian. C-clip: Multimodal continual learning for vision-language model. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Xiaohao Liu, Xiaobo Xia, See-Kiong Ng, and Tat-Seng Chua. Continual multimodal contrastive learning. *arXiv preprint arXiv:2503.14963*, 2025b.

Mao-Lin Luo, Zi-Hao Zhou, Tong Wei, and Min-Ling Zhang. Lada: Scalable label-specific clip adapter for continual learning. In *Forty-second International Conference on Machine Learning*, 2025.

Mingjun Ma, Saisai Hu, Tiantian Zhu, Yitian Lin, Jiachen Sun, Zhengqiu Weng, Dehao Li, and Yajie Zhang. MuSK: Multi-scale knowledge learning for provenance-graph anomaly detection. *Computer Networks*, 289:112728, 2026. doi: 10.1016/j.comnet.2026.112728.

Adyasha Maharana, Jaehong Yoon, Tianlong Chen, and Mohit Bansal. Adapt-∞: Scalable continual multimodal instruction tuning via dynamic data selection. In *The Thirteenth International Conference on Learning Representations*, 2025.

Imad Eddine Marouf, Enzo Tartaglione, Stéphane Lathuilière, and Joost van de Weijer. Ask and remember: A questions-only replay strategy for continual visual question answering. *arXiv preprint arXiv:2502.04469*, 2025. URL https://arxiv.org/pdf/2502.04469.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019.

Beiwen Tian, Mingdao Liu, Huan-ang Gao, Pengfei Li, Hao Zhao, and Guyue Zhou. Unsupervised road anomaly detection with language anchors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7778–7785, 2023. doi: 10.1109/ICRA48891.2023.10160470.

Mengxiao Wang and Nitesh Saxena. Trading agents or market crashers? dissecting robustness and security failures in academic financial LLM trading schemes. arXiv preprint, 2026. URL https://arxiv.org/abs/2609.19705.

Peiqi Wang, Yikang Shen, Zhen Guo, Matthew Stallone, Yoon Kim, Polina Golland, and Rameswar Panda. Diversity measurement and subset selection for instruction tuning datasets. *arXiv preprint arXiv:2402.02318*, 2024a.

Zijun Wang, Saisai Hu, Tiantian Zhu, Wenrui Cheng, Zhengqiu Weng, Haiting Chen, Xiangyang Zheng, and Suyu Zhang. Trident: Dual-stream APT attribution over heterogeneous threat knowledge graphs. *Computers & Security*, 171:105094, 2026. doi: 10.1016/j.cose.2026.105094.

Ziqi Wang, Chang Che, Qi Wang, Yangyang Li, Zenglin Shi, and Meng Wang. Separable mixture of low-rank adaptation for continual visual instruction tuning. *arXiv preprint arXiv:2411.13949*, 2024b.

Xiwen Wei, Mustafa Munir, and Radu Marculescu. Mitigating intra-and inter-modal forgetting in continual learning of unified multimodal models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Bin Wu, Wuxuan Shi, Jinqiao Wang, and Mang Ye. Synthetic data is an elegant gift for continual vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2813–2823, 2025.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Canran Xiao and Liwei Hou. Prototype-aligned federated soft-prompts for continual web personalization. In *Proceedings of the ACM Web Conference 2026*, pages 6743–6754, 2026.

Canran Xiao, Tianxiang Xu, Siyuan Ma, Yiyang Jiang, Haoyu Gao, and Yuhan Wu. Reversible primitive-composition alignment for continual vision-language learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=eiTy6AYeQi.

Linqi Xiao, Venkateswarlu Gudepu, and Andrea Fumagalli. CORA: A conversational AI-agent for intelligent automation of OpenROADM multi-vendor optical transport networks. In *Applications of Machine Learning 2026*, volume 14218 of *Proceedings of SPIE*, page 142180Y. SPIE, 2026b. doi: 10.1117/12.3113861.

Linqi Xiao, Aparaajitha Gomathinayakam Latha, Venkateswarlu Gudepu, and Andrea Fumagalli. A YANG-grounded LLM agent supporting multi-vendor OpenROADM optical transport networks. In *Proceedings of the ACM SIGCOMM 2026 Conference*, pages 2109–2115. ACM, 2026c. doi: 10.1145/3789240.3822606. HotOptics workshop.

Mengzhu Xu, Hanzhi Liu, Ningkang Peng, Qianyu Chen, and Canran Xiao. Affordance-first decomposition for continual learning in video-language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3908–3919, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/xu_Affordance-First_Decomposition_for_Continual_Learning_in_Video-Language_Understanding_CVPR_2026_paper.html.

Dishu Yang, Jingjing Liu, and Jize Li. Balancing privacy, utility, and safety in LLM alignment through preference optimization. arXiv preprint arXiv:2608.30141, 2026a. URL https://arxiv.org/abs/2608.30141.

Dishu Yang, Qi Su, Hongbo Qin, and Hansong Zhang. Measurement-first auditing of agentic leaderboards: Contamination susceptibility, matched-control re-evaluation, and scorer validation. arXiv preprint arXiv:2610.05830, 2026b. URL https://arxiv.org/abs/2610.05830.

Junyi Yao, Zihao Zheng, and Jiayu Long. Perturbation sensitivity of maximum-likelihood pairwise ranking in computational decision systems. arXiv preprint, 2026. URL https://arxiv.org/abs/2604.17805.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

Duzhen Zhang, Yong Ren, Zhong-Zhi Li, Yahan Yu, Jiahua Dong, Chenxing Li, Zhilong Ji, and Jinfeng Bai. Enhancing multimodal continual instruction tuning with branchlora. *arXiv preprint arXiv:2506.02041*, 2025a.

Liang Zhang, Ziyao Lu, Fandong Meng, Hui Li, Jie Zhou, and Jinsong Su. Advancing smoe for continuous domain adaptation of mllms: Adaptive router and domain-specific loss. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26584–26602, 2025b.

Xiaoyu Zhang, Juan Zhai, Shiqing Ma, Chao Shen, Tianlin Li, Weipeng Jiang, and Yang Liu. Staff: Speculative coreset selection for task-specific fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Yifei Zhang, Hao Zhao, Hongyang Li, and Siheng Chen. FastMAC: Stochastic spectral sampling of correspondence graph. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17857–17867, 2024.

Zihao Zheng, Baichuan Li, Junyi Yao, and Jiayu Long. Reliable financial named entity recognition under domain shift: Confidence estimation and selective prediction. arXiv preprint, 2026. URL https://arxiv.org/abs/2608.19558.

Da-Wei Zhou, Kai-Wen Li, Jingyi Ning, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. External knowledge injection for clip-based class-incremental learning. *arXiv preprint arXiv:2503.08510*, 2025.

A Full Proofs for §4

A.1 Proof Setup and Definitions

We analyze the relaxed outer selector before the final TopB rounding step. At task t, the candidate pool is $\mathcal{U}_t = \mathcal{C}_t \cup \mathcal{M}_{t-1}$, where $\mathcal{C}_t$ contains entries extracted from the current task and $\mathcal{M}_{t-1}$ is the previous memory. Let $N = |\mathcal{U}_t|$, and let $w \in \mathbb{R}^N_{\geq 0}$ denote relaxed selection weights.

Budget simplex. The feasible set is the fixed-budget simplex

$$\begin{aligned} \Delta_B^N &\triangleq \Big\{ w \in \mathbb{R}^N_{\geq 0} : \sum_{i=1}^{N} w_i = B \Big\}, \\ D &\triangleq \max_{u,v \in \Delta_B^N} \|u - v\|_2 = B\sqrt{2}. \end{aligned} \tag{16}$$

We write $\pi_i(w) = w_i/B$, so that $\sum_i \pi_i(w) = 1$. During selection, $\pi_i(w)$ acts as a differentiable inclusion prior inside the retrieval weights.

Inner-solved outer surrogate. Given w, the inner calibration $\phi_t^\star(w)$ is obtained by solving the current-task calibration problem in Eq. (6). The outer analysis treats this calibration as solved and defines the per-task surrogate

$$f_t(w) \triangleq \mathcal{L}_t\big(w, \phi_t^\star(w)\big) + \beta \mathcal{A}_t\big(w, \phi_t^\star(w)\big) + \gamma \Omega(w), \tag{17}$$

where

$$\mathcal{L}_t(w, \phi) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\mathrm{NLL}}\big(\theta;\, x, y \mid \mathcal{U}_t, w, \phi\big), \tag{18}$$

$$\mathcal{A}_t(w, \phi) = \frac{1}{|\mathcal{A}_{<t}|} \sum_{(x,y)\in\mathcal{A}_{<t}} \mathcal{L}_{\mathrm{NLL}}\big(\theta;\, x, y \mid \mathcal{U}_t, w, \phi\big). \tag{19}$$

Here, $\mathcal{L}_t$ measures current-task fit, $\mathcal{A}_t$ measures anchor retention, and $\Omega(w)$ is the log-determinant coverage penalty in the frozen retrieval space.

Projected outer update. The relaxed selector is updated by projected online gradient descent:

$$w_{t+1} = \Pi_{\Delta_B^N}\Big(w_t - \eta_{\mathrm{pgd}} \nabla f_t(w_t)\Big), \tag{20}$$

where $\Pi_{\Delta_B^N}$ is Euclidean projection. After optimization at task t, INDUCEKV discretizes the relaxed selector by taking $\mathrm{TopB}(w_t)$ and commits the corresponding entries as $\mathcal{M}_t$.

Assumption A.1 (Convexity and bounded gradients)**.** *For each task t, the inner-solved surrogate $f_t : \Delta_B^N \to \mathbb{R}$ is convex and differentiable, and there exists $G > 0$ such that*

$$\|\nabla f_t(w)\|_2 \le G, \qquad \forall w \in \Delta_B^N. \tag{21}$$

This assumption is standard for online convex optimization analyses. In our setting, it corresponds to using a stable inner calibration solver, bounded retrieval keys, and a smooth soft-retrieval dependence on w. The log-determinant coverage term is convex and has a bounded gradient, as shown in Lemmas A.4–A.5.

Dynamic comparator. For nonstationary streams, we compare the learner to a changing sequence of relaxed selectors $\{u_t\}_{t=1}^T \subset \Delta_B^N$. Its path-length variation is

$$V_T \triangleq \sum_{t=1}^{T-1} \|u_{t+1} - u_t\|_2. \tag{22}$$

Small V_T corresponds to streams whose best memory allocation evolves gradually, while large V_T corresponds to abrupt task shifts.

Historical risk and anchor representativeness. Let $\widetilde{\mathcal{P}}_{<t}$ denote the unknown mixture distribution over previous tasks. For the inner-solved induced predictor, define the true historical risk

$$\widetilde{\mathcal{R}}_{<t}(w) \triangleq \mathbb{E}_{(x,y)\sim\widetilde{\mathcal{P}}_{<t}}\big[\ell_t(x, y; w)\big], \tag{23}$$

where $\ell_t(x, y; w)$ is the negative log-likelihood induced by candidate-based retrieval and calibrated KV injection. We use the following coreset-style anchor condition.

Assumption A.2 (ϵ-representative anchors)**.** *For each task t, the anchor loss uniformly approximates the true historical risk:*

$$\sup_{w\in\Delta_B^N} \Big|\mathcal{A}_t(w) - \widetilde{\mathcal{R}}_{<t}(w)\Big| \le \epsilon_t^{\mathrm{rep}}. \tag{24}$$

This condition is empirically checkable by measuring the gap between anchor loss and held-out past-task loss. The approximation error appears explicitly in the retention bound.

Rounding by TopB**.** The regret analysis is stated for the relaxed selector w_t. The actual memory update uses $\mathrm{TopB}(w_t)$, which is a deterministic rounding step from the relaxed solution to a discrete budget-B subset. If each f_t is Lipschitz around the relaxed solution, the additional rounding gap is controlled by the distance between w_t and its rounded selection vector. Empirically, the spectral coverage term reduces this gap by discouraging highly concentrated or near-duplicate selections.

A.2 Preliminaries

We use the following standard properties of Euclidean projection. For a closed convex set $\mathcal{K} \subset \mathbb{R}^N$, define $\Pi_{\mathcal{K}}(x) = \arg\min_{y \in \mathcal{K}} \|y - x\|_2$.

Lemma A.3 (Projection inequality)**.** *Let $\mathcal{K}$ be closed and convex, $w \in \mathcal{K}$, $g \in \mathbb{R}^N$, $\eta_{\text{pgd}} > 0$, and $w^+ \triangleq \Pi_{\mathcal{K}}(w - \eta_{\text{pgd}} g)$. Then for any $u \in \mathcal{K}$,*

$$\langle g, w - u \rangle \;\le\; \frac{\|u - w\|_2^2 - \|u - w^+\|_2^2}{2\eta_{\text{pgd}}} \;+\; \frac{\eta_{\text{pgd}}}{2}\|g\|_2^2. \tag{25}$$

Proof. Let $z \triangleq w - \eta_{\text{pgd}} g$ and $w^+ = \Pi_{\mathcal{K}}(z)$. The first-order optimality condition of Euclidean projection (a variational inequality) states that

$$\langle z - w^+, u - w^+ \rangle \le 0 \qquad \forall u \in \mathcal{K}. \tag{26}$$

Substituting $z = w - \eta_{\text{pgd}} g$ into (26) gives

$$\langle w - \eta_{\text{pgd}} g - w^+, u - w^+ \rangle \le 0 \;\;\Longrightarrow\;\; \langle g, u - w^+ \rangle \ge \frac{1}{\eta_{\text{pgd}}} \langle w - w^+, u - w^+ \rangle. \tag{27}$$

Now expand the squared norm difference:

$$\begin{aligned} \|u - w\|_2^2 - \|u - w^+\|_2^2 &= \langle u - w, u - w \rangle - \langle u - w^+, u - w^+ \rangle \\ &= \langle (u - w^+) + (w^+ - w), (u - w^+) + (w^+ - w) \rangle - \|u - w^+\|_2^2 \\ &= 2\langle u - w^+, w^+ - w \rangle + \|w^+ - w\|_2^2 \\ &= -2\langle w - w^+, u - w^+ \rangle + \|w^+ - w\|_2^2. \end{aligned} \tag{28}$$

Rearranging (28) yields

$$\langle w - w^+, u - w^+ \rangle = \frac{1}{2}\Big(\|u - w^+\|_2^2 - \|u - w\|_2^2 + \|w^+ - w\|_2^2 \Big). \tag{29}$$

Plug (29) into (27):

$$\langle g, u - w^+ \rangle \ge \frac{1}{2\eta_{\text{pgd}}}\Big(\|u - w^+\|_2^2 - \|u - w\|_2^2 + \|w^+ - w\|_2^2 \Big). \tag{30}$$

Finally decompose $\langle g, w - u \rangle = \langle g, w - w^+ \rangle + \langle g, w^+ - u \rangle$ and use (30):

$$\begin{aligned} \langle g, w - u \rangle &= \langle g, w - w^+ \rangle - \langle g, u - w^+ \rangle \\ &\le \langle g, w - w^+ \rangle - \frac{1}{2\eta_{\text{pgd}}}\Big(\|u - w^+\|_2^2 - \|u - w\|_2^2 + \|w^+ - w\|_2^2 \Big) \\ &= \frac{\|u - w\|_2^2 - \|u - w^+\|_2^2}{2\eta_{\text{pgd}}} + \Big(\langle g, w - w^+ \rangle - \frac{1}{2\eta_{\text{pgd}}}\|w^+ - w\|_2^2 \Big). \end{aligned} \tag{31}$$

Apply the inequality $a^\top b \le \frac{\eta_{\text{pgd}}}{2}\|a\|_2^2 + \frac{1}{2\eta_{\text{pgd}}}\|b\|_2^2$ with $a = g$ and $b = w - w^+$:

$$\langle g, w - w^+ \rangle \le \frac{\eta_{\text{pgd}}}{2}\|g\|_2^2 + \frac{1}{2\eta_{\text{pgd}}}\|w - w^+\|_2^2. \tag{32}$$

Substituting (32) into (31) cancels the $\|w - w^+\|_2^2/(2\eta_{\text{pgd}})$ terms, yielding (25). □

A.3 Convexity and Gradient Control for the Log-Det Coverage Regularizer

Recall the coverage regularizer (Eq. (9) in §3):

$$\Omega(w) = -\log\det\big(C(w) + \epsilon I_{d'}\big), \qquad C(w) = \sum_{i=1}^{N} \pi_i(w)\, z_i z_i^\top, \quad \pi_i(w) = \frac{w_i}{B}. \tag{33}$$

Lemma A.4 (Convexity of Ω)**.** *$\Omega(w)$ is convex over Δ_B^N.*

Proof. The map $w \mapsto C(w)$ is affine because $C(w) = \sum_i (w_i/B) z_i z_i^\top$. The function $X \mapsto -\log\det(X)$ is convex over the positive definite cone $X \succ 0$ (a standard result in matrix convex analysis). Since $C(w) + \epsilon I_{d'} \succ 0$ for any $\epsilon > 0$, the composition $w \mapsto -\log\det(C(w) + \epsilon I_{d'})$ preserves convexity under an affine map, hence Ω is convex. □

Lemma A.5 (Gradient formula and norm bound)**.** *Let $A(w) \triangleq C(w) + \epsilon I_{d'}$. Then for each coordinate i,*

$$\frac{\partial \Omega(w)}{\partial w_i} = -\frac{1}{B}\, z_i^\top A(w)^{-1} z_i. \tag{34}$$

If $\|z_i\|_2 \le 1$ for all i, then

$$\Big|\frac{\partial \Omega(w)}{\partial w_i}\Big| \le \frac{1}{B\epsilon}, \qquad \|\nabla \Omega(w)\|_2 \le \frac{\sqrt{N}}{B\epsilon}. \tag{35}$$

Proof. Differentiate $\Omega(w) = -\log\det(A(w))$. Using the matrix differential identity $\mathrm{d}\log\det(A) = \mathrm{tr}(A^{-1}\mathrm{d}A)$, we have $\mathrm{d}\Omega = -\mathrm{tr}(A^{-1}\mathrm{d}A)$. Moreover, $\partial A/\partial w_i = \partial C/\partial w_i = (1/B) z_i z_i^\top$. Thus

$$\frac{\partial \Omega}{\partial w_i} = -\mathrm{tr}\Big(A^{-1} \cdot \frac{1}{B} z_i z_i^\top\Big) = -\frac{1}{B}\mathrm{tr}\big(z_i^\top A^{-1} z_i\big) = -\frac{1}{B} z_i^\top A^{-1} z_i,$$

proving (34). For the bound, note $A(w) \succeq \epsilon I$ implies $A(w)^{-1} \preceq \epsilon^{-1} I$. Therefore $z_i^\top A^{-1} z_i \le \epsilon^{-1}\|z_i\|_2^2 \le \epsilon^{-1}$, giving $|\partial\Omega/\partial w_i| \le 1/(B\epsilon)$. Finally, $\|\nabla\Omega\|_2 \le \sqrt{N}\|\nabla\Omega\|_\infty \le \sqrt{N}/(B\epsilon)$. □

A.4 Proof of Theorem 4.1

Proof. Let $g_t \triangleq \nabla f_t(w_t)$. By convexity of f_t,

$$f_t(w_t) - f_t(u) \le \langle g_t, w_t - u\rangle. \tag{36}$$

Apply Lemma A.3 to the update $w_{t+1} = \Pi_{\Delta_B^N}(w_t - \eta_{\mathrm{pgd}} g_t)$ with $\mathcal{K} = \Delta_B^N$:

$$\langle g_t, w_t - u\rangle \le \frac{\|u - w_t\|_2^2 - \|u - w_{t+1}\|_2^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}}}{2}\|g_t\|_2^2. \tag{37}$$

Combining (36) and (37) yields

$$f_t(w_t) - f_t(u) \le \frac{\|u - w_t\|_2^2 - \|u - w_{t+1}\|_2^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}}}{2}\|g_t\|_2^2. \tag{38}$$

Summing (38) from $t = 1$ to T gives a telescoping series:

$$\begin{aligned}
\sum_{t=1}^{T}\big(f_t(w_t) - f_t(u)\big) &\le \frac{1}{2\eta_{\mathrm{pgd}}}\sum_{t=1}^{T}\big(\|u - w_t\|_2^2 - \|u - w_{t+1}\|_2^2\big) + \frac{\eta_{\mathrm{pgd}}}{2}\sum_{t=1}^{T}\|g_t\|_2^2 \\
&= \frac{\|u - w_1\|_2^2 - \|u - w_{T+1}\|_2^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}}}{2}\sum_{t=1}^{T}\|g_t\|_2^2 \\
&\le \frac{D^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}}}{2}\sum_{t=1}^{T}\|g_t\|_2^2,
\end{aligned} \tag{39}$$

where we used $\|u - w_1\|_2 \le D$ and $\|u - w_{T+1}\|_2^2 \ge 0$. Using Assumption A.1 gives $\|g_t\|_2 \le G$, hence

$$\mathrm{Reg}_T(u) \le \frac{D^2}{2\eta_{\mathrm{pgd}}} + \frac{\eta_{\mathrm{pgd}} G^2 T}{2}.$$

Choosing $\eta_{\mathrm{pgd}} = D/(G\sqrt{T})$ yields $\mathrm{Reg}_T(u) \le DG\sqrt{T}$. With $D = B\sqrt{2}$ from (16), we obtain (11). □

A.5 Proof of Theorem 4.2

Proof. Let $g_t \triangleq \nabla f_t(w_t)$ and $\{u_t\}_{t=1}^T \subset \Delta_B^N$. By convexity,

$$f_t(w_t) - f_t(u_t) \le \langle g_t, w_t - u_t \rangle. \tag{40}$$

Decompose

$$\langle g_t, w_t - u_t \rangle = \langle g_t, w_t - u_{t+1} \rangle + \langle g_t, u_{t+1} - u_t \rangle. \tag{41}$$

Apply Lemma A.3 to the first term with comparator $u = u_{t+1}$:

$$\langle g_t, w_t - u_{t+1} \rangle \le \frac{\|u_{t+1} - w_t\|_2^2 - \|u_{t+1} - w_{t+1}\|_2^2}{2\eta_{\text{pgd}}} + \frac{\eta_{\text{pgd}}}{2}\|g_t\|_2^2. \tag{42}$$

For the second term, Cauchy–Schwarz gives

$$\langle g_t, u_{t+1} - u_t \rangle \le \|g_t\|_2 \cdot \|u_{t+1} - u_t\|_2 \le G\,\|u_{t+1} - u_t\|_2. \tag{43}$$

Combining (40)–(43) yields

$$f_t(w_t) - f_t(u_t) \le \frac{\|u_{t+1} - w_t\|_2^2 - \|u_{t+1} - w_{t+1}\|_2^2}{2\eta_{\text{pgd}}} + \frac{\eta_{\text{pgd}}}{2}\|g_t\|_2^2 + G\,\|u_{t+1} - u_t\|_2. \tag{44}$$

We now relate $\|u_{t+1} - w_t\|_2^2$ to $\|u_t - w_t\|_2^2$. Using $(a+b)^2 \le a^2 + 2ab + b^2$ with $a = \|u_t - w_t\|_2$ and $b = \|u_{t+1} - u_t\|_2$, we obtain

$$\begin{aligned}
\|u_{t+1} - w_t\|_2^2 &= \|(u_t - w_t) + (u_{t+1} - u_t)\|_2^2 \\
&\le \|u_t - w_t\|_2^2 + 2\|u_t - w_t\|_2\|u_{t+1} - u_t\|_2 + \|u_{t+1} - u_t\|_2^2 \\
&\le \|u_t - w_t\|_2^2 + 2D\|u_{t+1} - u_t\|_2 + D\|u_{t+1} - u_t\|_2,
\end{aligned} \tag{45}$$

where we used $\|u_t - w_t\|_2 \le D$ and $\|u_{t+1} - u_t\|_2^2 \le D\|u_{t+1} - u_t\|_2$ (because $\|u_{t+1} - u_t\|_2 \le D$ on a diameter-D set). Thus

$$\|u_{t+1} - w_t\|_2^2 \le \|u_t - w_t\|_2^2 + 3D\|u_{t+1} - u_t\|_2. \tag{46}$$

Plug (46) into (44) and sum over $t = 1, \dots, T$:

$$\begin{aligned}
\sum_{t=1}^T \big(f_t(w_t) - f_t(u_t)\big) \le \frac{1}{2\eta_{\text{pgd}}} \sum_{t=1}^T \Big(\|u_t - w_t\|_2^2 - \|u_{t+1} - w_{t+1}\|_2^2\Big) + \frac{3D}{2\eta_{\text{pgd}}} \sum_{t=1}^T \|u_{t+1} - u_t\|_2 \\
+ \frac{\eta_{\text{pgd}}}{2} \sum_{t=1}^T \|g_t\|_2^2 + G \sum_{t=1}^T \|u_{t+1} - u_t\|_2.
\end{aligned} \tag{47}$$

The first summation telescopes:

$$\sum_{t=1}^T \Big(\|u_t - w_t\|_2^2 - \|u_{t+1} - w_{t+1}\|_2^2\Big) = \|u_1 - w_1\|_2^2 - \|u_{T+1} - w_{T+1}\|_2^2 \le D^2. \tag{48}$$

Also, $\sum_{t=1}^T \|u_{t+1} - u_t\|_2 = V_T$ if we set $u_{T+1} = u_T$. Using $\|g_t\|_2 \le G$ and combining terms gives

$$\text{DReg}_T(\{u_t\}) \le \frac{D^2}{2\eta_{\text{pgd}}} + \frac{3D}{2\eta_{\text{pgd}}} V_T + \frac{\eta_{\text{pgd}} G^2 T}{2} + G V_T. \tag{49}$$

To express the bound in the form of Theorem 4.2, we may upper bound $GV_T \le (D/\eta_{\text{pgd}})V_T$ by choosing $\eta_{\text{pgd}} \le D/G$ (always possible in our regime), yielding

$$\text{DReg}_T(\{u_t\}) \le \frac{D^2 + 5DV_T}{2\eta_{\text{pgd}}} + \frac{\eta_{\text{pgd}} G^2 T}{2}. \tag{50}$$

A slightly sharper constant can be obtained by a refined coupling that avoids the $\|u_{t+1} - u_t\|_2^2 \le D\|u_{t+1} - u_t\|_2$ relaxation. The resulting order is $O\big((D^2 + DV_T)/\eta_{\text{pgd}} + \eta_{\text{pgd}} G^2 T\big)$ form. Optimizing the right-hand side over η_{pgd} yields $\eta_{\text{pgd}}^\star = \sqrt{(D^2 + cDV_T)/(G^2 T)}$ and $\text{DReg}_T(\{u_t\}) \le G\sqrt{T}\sqrt{D^2 + cDV_T}$ for a small constant c. □

Comment on constants. The proof identifies the constants behind the dynamic-regret order in (12). A Bregman-divergence analysis on Δ_B^N (entropic mirror descent) and retaining the squared-variation term $\sum_t \|u_{t+1} - u_t\|_2^2$ before bounding it by DV_T can tighten these constants.

A.6 Proof of Theorem 4.3

Proof. The pointwise bound follows directly from Assumption A.2. For $w = w_t$,

$$\widetilde{\mathcal{R}}_{<t}(w_t) \le \mathcal{A}_t(w_t) + \epsilon_t^{\mathrm{rep}},$$

which gives Eq. (14).

We now prove the averaged bound. Summing the pointwise inequality over $t = 1, \dots, T$ gives

$$\sum_{t=1}^{T} \widetilde{\mathcal{R}}_{<t}(w_t) \le \sum_{t=1}^{T} \mathcal{A}_t(w_t) + \sum_{t=1}^{T} \epsilon_t^{\mathrm{rep}}. \tag{51}$$

Since the surrogate is

$$f_t(w) = \mathcal{L}_t(w, \phi_t^\star(w)) + \beta \mathcal{A}_t(w, \phi_t^\star(w)) + \gamma \Omega(w),$$

and the likelihood and coverage terms are nonnegative, we have

$$\mathcal{A}_t(w_t) \le \frac{1}{\beta} f_t(w_t). \tag{52}$$

Therefore,

$$\sum_{t=1}^{T} \widetilde{\mathcal{R}}_{<t}(w_t) \le \frac{1}{\beta} \sum_{t=1}^{T} f_t(w_t) + \sum_{t=1}^{T} \epsilon_t^{\mathrm{rep}}. \tag{53}$$

For any comparator $u \in \Delta_B^N$, add and subtract $\sum_{t=1}^{T} f_t(u)$:

$$\sum_{t=1}^{T} f_t(w_t) = \sum_{t=1}^{T} f_t(u) + \underbrace{\sum_{t=1}^{T} \big(f_t(w_t) - f_t(u)\big)}_{\mathrm{Reg}_T(u)}. \tag{54}$$

Combining Eqs. (53) and (54), then dividing by T, yields

$$\frac{1}{T} \sum_{t=1}^{T} \widetilde{\mathcal{R}}_{<t}(w_t) \le \frac{1}{\beta T} \sum_{t=1}^{T} f_t(u) + \frac{\mathrm{Reg}_T(u)}{\beta T} + \frac{1}{T} \sum_{t=1}^{T} \epsilon_t^{\mathrm{rep}},$$

which is Eq. (15). □

B Additional Experimental Details

B.1 Benchmarks, Task Order, and Data Processing

UCIT (task-incremental CIT). We follow the UCIT benchmark introduced in Guo et al. [2025], consisting of six datasets: {ARXIVQA, IMAGENET-R, VIZWIZ-CAPTION, ICONQA, CLEVR-MATH, FLICKR30K}. Each dataset is treated as one task; we use the official task-specific instruction templates (e.g., option-letter answers for multiple-choice tasks and short captions for captioning tasks) Guo et al. [2025]. We use the default task order of Guo et al. [2025] unless otherwise stated.

CoIN (task-incremental CIT). We follow COIN Tang et al. [2019], which covers eight task categories and ten underlying datasets. The tasks include: SCIENCEQA, VQAV2, GQA, VIZWIZ, TEXTVQA, OCR-VQA, a CLASSIFICATION task (ImageNet subset), and a GROUNDING task built from REFCOCO/REFCOCO+/REFCOCOG Tang et al. [2019]. We use the benchmark's default random task order.

Continual VQA. For VQAV2 continual VQA, we use the 10-task question-type split: recognition, location, judge, commonsense, count, action, color, type, subcategory, causal Huai et al. [2025]. For

VQACL, we follow Marouf et al. [2025] on VQAv2 and NExT-QA, and report both the standard testing protocol and the novel-composition evaluation when available.

Domain-incremental CIT (Medicine/Chart/Math). We follow the benchmark in Zhang et al. [2025b]: **Medicine** uses VQA-RAD, PathVQA, SLAKE; **Chart** uses ChartQA, Chart-to-Table, Chart-to-Text; **Math** uses a combined MWP&GPS suite for multimodal math word problems and geometry problem solving Zhang et al. [2025b]. Each domain is treated as one continual step (domain-incremental).

LiIT stream (dynamic dataset arrival). We follow Adapt-∞ Maharana et al. [2025] and sequentially train over a dataset stream M3IT $\rightarrow$ MiniGPT4 $\rightarrow$ MANTIS $\rightarrow$ LAMM $\rightarrow$ VisionFLAN starting from an instruction-tuned LLaVA checkpoint (timestep t=0 in Maharana et al. [2025]). We evaluate on the same skill-suite grouping (e.g., (Knowledge) VQA, grounding, reasoning, language-only, multilingual) as in Maharana et al. [2025].

B.2 Metrics and Continual Learning Scores

UCIT (Avg/Last). We report *Avg* and *Last* as in Guo et al. [2025], computed from the final model's test scores on all seen tasks and on the most recent task, respectively. Captioning tasks follow standard caption metrics (BLEU-1/2/3/4, METEOR, ROUGE-L, CIDEr) aggregated per the benchmark Guo et al. [2025].

CoIN (task scores, MAA, BWT). We follow Tang et al. [2019]. Image QA tasks use answer accuracy; grounding uses IoU-based success; classification uses top-1 accuracy. We aggregate across tasks with mean average accuracy (MAA) and backward transfer (BWT):

$$\mathrm{MAA} = \frac{1}{T}\sum_{t=1}^{T} a_{T,t}, \tag{55}$$

$$\mathrm{BWT} = \frac{1}{T-1}\sum_{t=1}^{T-1}\big(a_{T,t} - a_{t,t}\big), \tag{56}$$

where $a_{i,t}$ is the task-t test score after finishing training on task i.

Continual VQA (AP, AF/Forget). We report the same metrics as Huai et al. [2025], Marouf et al. [2025]:

$$\mathrm{AP} = \frac{1}{T}\sum_{t=1}^{T} a_{T,t}, \tag{57}$$

$$\boxed{\mathrm{AF} = \frac{1}{T-1}\sum_{t=1}^{T-1}\left(\max_{t\le i<T} a_{i,t} - a_{T,t}\right)}, \tag{58}$$

with $a_{i,t}$ measured by VQA accuracy on VQAv2 and the protocol-specific score for NExT-QA Marouf et al. [2025].

Domain-incremental CIT (per-domain metrics). We follow Zhang et al. [2025b]: medicine reports closed-set accuracy and open-set recall; chart reports RelaxAcc (VQA), RMSF1 (table), and BLEU4 (text); math reports accuracy on MWP&GPS. We report per-domain results and their average.

LiIT (AvgAcc, Relative Gain, Forgetting Rate). We follow Maharana et al. [2025] and report AvgAcc at the final timestep, Relative Gain (normalized by per-skill upper bounds), and Forgetting Rate averaged across skills and timesteps.

System metrics. We report (i) external memory size (MB), (ii) additional KV tokens injected per layer, and (iii) end-to-end decoding latency (tokens/s), averaged over benchmark test sets.

B.3 Baselines and Budget Matching

Recent SOTA baselines. We include the most relevant recent methods per suite: **CIT:** HiDe-LLaVA Guo et al. [2025], SEFE Chen et al. [2025], BranchLoRA Zhang et al. [2025a], and related PEFT routing/merging baselines reported with UCIT/CoIN. **Domain-incremental CIT:** SMoE Zhang

et al. [2025b] and its compared PEFT baselines (e.g., M-LoRA/O-LoRA/MoELoRA/SMoELoRA/L-SMoE) under matched parameter budgets. **Continual VQA:** CL-MoE Huai et al. [2025] and QUAD Marouf et al. [2025], plus rehearsal and regularization baselines (ER/DER/VS/VQACL, EWC/MAS). **LiIT:** Adapt-∞ Maharana et al. [2025] and its suite baselines (random replay; score-based selection such as entropy/perplexity/EL2N; and recent pruning baselines).

Budget matching across methods. INDUCEKV freezes the backbone and learns only a tiny calibration ϕ while selecting an external KV memory. To compare against (i) parameter-growing PEFT methods and (ii) replay-based methods, we match *total extra footprint* and keep inference compute comparable:

- **Footprint matching.** We match either (a) total FP16 bytes of INDUCEKV memory vs. added trainable parameters (PEFT), or (b) total stored replay bytes (rehearsal).
- **Compute matching.** We report results at fixed additional KV-token budgets (controlled by (B, m) for INDUCEKV) and use the same max decoding length and batch sizes across methods.

We provide the full budget conversion and all matched settings in Table 5.

Tier	(B, m)	KV tokens (per layer)	Mem 4B (MiB)	Mem 8B (MiB)	PEFT 4B equiv. (FP16 params)	Replay text 4B (exemplars)
S	$(64, 8)$	512	72.31	72.50	37.91M	37,024
M	$(128, 8)$	1,024	144.62	145.00	75.83M	74,048
L (main)	$(256, 8)$	2,048	289.25	290.00	151.65M	148,096
XL	$(512, 8)$	4,096	578.50	580.00	303.30M	296,192

Table 5: **Budget conversion across adaptation methods.** INDUCEKV stores B FP16 retrieval keys and their layerwise KV payloads. We use the same configuration as Table 7: L=36, H_{KV}=8, d_h=128, and m=8, with d=2560 (4B) or d=4096 (8B). Total storage is $\boxed{\mathrm{MemBytes} = B(2d + 4LH_{\mathrm{KV}}md_h)}$, or $\mathrm{MemBytes}/2^{20}$ MiB. The 4B PEFT equivalent is $\lfloor \mathrm{MemBytes}/2 \rfloor$ FP16 parameters. For text replay, each exemplar uses 2048 bytes (512 int32 token IDs), yielding $\lfloor \mathrm{MemBytes}/2048 \rfloor$ exemplars. For image+text replay, replace the denominator with $2048 + \bar{s}_{\mathrm{img}}$, where $\bar{s}_{\mathrm{img}}$ is the mean JPEG payload size in bytes under the same preprocessing and compression.

B.4 Implementation Details of INDUCEKV

Implementation. The result tables identify the backbone and evaluation protocol for each method. We use LLaVA-OneVision-4B by default and include a backbone-matched INDUCEKV variant for suites whose strongest baselines use LLaVA-1.5/LLaVA-v1.5-7B. For VQACL on VQAv2 and NExT-QA, QUAD and VQACL use the T5 protocol. The default INDUCEKV row provides a separate MLLM-protocol reference. Additional results on `LLaVA-OneVision-1.5-8B-Instruct` and cross-family backbones are reported in §C.7.

Backbones, checkpoints, and precision. We use two LLaVA-OneVision-1.5 instruction-tuned checkpoints: `lmms-lab/LLaVA-OneVision-1.5-4B-Instruct` and `lmms-lab/LLaVA-OneVision-1.5-8B-Instruct`. Their text backbones use L=36 decoder layers and grouped-query attention with `num_attention_heads`=32, `num_key_value_heads`=8, and `head_dim`=128; the hidden size is d=2560 (4B) or d=4096 (8B). We additionally evaluate on `Qwen/Qwen3-VL-4B-Instruct` and `Qwen/Qwen3-VL-8B-Instruct`, which share the same $(L, \texttt{num_attention_heads}, \texttt{num_key_value_heads}, \texttt{head_dim})$ as above, with d=2560 (4B) or d=4096 (8B). For DeepSeek-VL2, we evaluate `deepseek-ai/deepseek-vl2-tiny` and `deepseek-ai/deepseek-vl2-small`, whose language backbones have $(L, d) = (12, 1280)$ and $(27, 2048)$, respectively, with `num_attention_heads` = `num_key_value_heads` (standard MHA). We also report results on `deepseek-ai/deepseek-vl2` (4.5B activated; MoE), and use the instantiated model's internal language configuration to determine the exact attention geometry for KV extraction when fields are not exposed in the wrapper config. Across all runs, backbone weights are kept frozen and executed in BF16, while all external memory tensors are stored in FP16.

Scope	Checkpoint	L	d	H	H_{KV}	d_h	MaxPos	torch_dtype
Main	lmms-lab/LLaVA-OneVision-1.5-4B-Instruct	36	2560	32	8	128	262144	BF16
Main	lmms-lab/LLaVA-OneVision-1.5-8B-Instruct	36	4096	32	8	128	32768	BF16
App.	Qwen/Qwen3-VL-4B-Instruct	36	2560	32	8	128	262144	BF16
App.	Qwen/Qwen3-VL-8B-Instruct	36	4096	32	8	128	262144	BF16
App.	deepseek-ai/deepseek-vl2-tiny	12	1280	10	10	128	4096	BF16
App.	deepseek-ai/deepseek-vl2-small	27	2048	16	16	128	4096	BF16
App.	deepseek-ai/deepseek-vl2	–	2560	–	–	–	4096	BF16

Table 6: **Backbone configurations used in this paper.** L is the number of decoder layers; d is the text hidden size; H and H_{KV} are the query-head and KV-head counts; d_h is the per-(KV)-head dimension. For DeepSeek-VL2 (base), the released wrapper config does not expose $(L, H, H_{\mathrm{KV}}, d_h)$, so we read them from the instantiated model at runtime to construct KV payloads.

Tier	B	m	Bm (KV tok/layer)	LLaVA-OV-4B (MiB)	LLaVA-OV-8B (MiB)	Qwen3-VL-4B (MiB)	Qwen3-VL-8B (MiB)	DS-VL2-Tiny / Small (MiB)
S	64	8	512	72.31	72.50	72.31	72.50	30.16 / 108.25
M	128	8	1024	144.62	145.00	144.62	145.00	60.31 / 216.50
L	256	8	2048	289.25	290.00	289.25	290.00	120.62 / 433.00
XL	512	8	4096	578.50	580.00	578.50	580.00	241.25 / 866.00

Table 7: **System overhead of INDUCEKV.** We report (i) the number of additional KV tokens injected per layer (Bm) and (ii) the FP16 external-memory footprint (MiB) computed by Eq. (59). DeepSeek-VL2 (base) is omitted here because $(L, H, H_{\mathrm{KV}}, d_h)$ is not fully specified in the released wrapper config; we report its exact footprint by directly summing the stored tensor sizes in our code.

External memory tensors and footprint. Each memory entry stores a normalized retrieval key $r(x) \in \mathbb{R}^d$ and layerwise KV payloads $\{(\bar{K}^\ell(x), \bar{V}^\ell(x))\}_{\ell=1}^L$ extracted from the frozen backbone (Sec. 3). We store $r(x)$ and all KV payload tensors in FP16. For backbones with grouped-query attention, we store KV payloads for KV heads only (H_{KV}), matching the shape of the frozen W_K^ℓ, W_V^ℓ. With payload length m=8, the per-entry FP16 footprint is

$$\mathrm{BytesPerEntry} \;=\; 2d \;+\; 4\,L\,H_{\mathrm{KV}}\,m\,d_h, \tag{59}$$

where $2d$ accounts for FP16 retrieval keys and the second term accounts for FP16 (K, V) payloads. We fix the spectral projection dimension d'=256 and stabilizer ϵ=10^{-3} in Eq. (9). We use a fixed random orthonormal projector $P \in \mathbb{R}^{d \times d'}$ generated once by drawing a Gaussian matrix and applying QR decomposition (seed 0), and reuse P for all tasks.

Memory budgets and system overhead. We sweep the entry budget B while fixing m=8, and report results under four budget tiers. Table 7 reports the induced KV-token overhead (Bm additional cache tokens per layer) and the corresponding FP16 memory footprint computed from Eq. (59) using the backbone hyperparameters in Table 6.

Bilevel optimization hyperparameters. We optimize the inner calibration $\phi = \{\phi_\tau, \phi_1, \ldots, \phi_L\}$ with AdamW: learning rate α_ϕ=1×10^{-2}, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0, for J=10 inner steps per outer iteration. We initialize the retrieval temperature to τ_0=0.07 (by setting $\phi_\tau = \log(\exp(\tau_0) - 1)$) and initialize all layer gates to λ_ℓ=0.5 (by setting ϕ_ℓ=0). For the outer selection weights, we run I=100 projected gradient steps per task with step size η_{pgd}=0.1 and Euclidean projection onto $\Delta_B^N = \{w \geq 0 : \sum_i w_i = B\}$. We use η_{reg}=10^{-4} for the ℓ_2 penalty in Eq. (6). For (β, γ) in Eq. (8), we perform a task-1 validation grid search over $\beta \in \{0.1, 0.3, 0.5, 1.0\}$ and $\gamma \in \{0, 0.05, 0.1, 0.3\}$, and then fix the selected values for the full stream.

Hardware and measurement protocol. All runs are executed on NVIDIA A100 80GB GPUs. Unless a benchmark specifies otherwise, we use batch size 1 for decoding evaluation and report (i) peak GPU memory (GB) and (ii) decoding throughput (tokens/s) averaged over the full test set, with identical maximum generation length and stopping criteria across all methods. We keep the same image preprocessing and chat templates provided by each backbone's official processor.

B.5 Other Experiment Details

Memory-attention utilization protocol. To quantify whether external KV memory is used during generation, we measure the attention mass assigned to injected memory tokens. For each task/domain in UCIT, CoIN, and continual VQA, we sample 500–1,000 evaluation instances and compute the layer-wise memory-attention ratio

$$\rho^{\ell} \triangleq \mathbb{E}_{(x,y)}\left[\frac{1}{H}\sum_{h=1}^{H}\frac{\sum_{q}\sum_{k\in\mathcal{K}_{\mathrm{mem}}}\mathrm{Attn}_{qk}^{\ell,h}}{\sum_{q}\sum_{k}\mathrm{Attn}_{qk}^{\ell,h}}\right], \tag{60}$$

where H is the number of attention heads, $\mathcal{K}_{\mathrm{mem}}$ indexes injected KV-memory tokens, q ranges over query positions, and the denominator sums over both self-attention and memory tokens. We compute ρ^{ℓ} after memory injection using the same decoding settings as the main evaluation. For each task t, we also record the raw performance gain

$$\Delta_t = \mathrm{Score}_t(\textsc{InduceKV}) - \mathrm{Score}_t(\textsc{no-mem}), \tag{61}$$

where NO-MEM disables KV-memory injection while keeping the frozen backbone, prompt, decoding, and evaluation metric unchanged. Because tasks use heterogeneous metrics, the final Gain column in Fig. 3 reports a min–max normalized version of Δ_t:

$$\widetilde{\Delta}_t = \frac{\Delta_t - \min_{t'}\Delta_{t'}}{\max_{t'}\Delta_{t'} - \min_{t'}\Delta_{t'} + \delta}, \qquad \delta = 10^{-8}. \tag{62}$$

Raw gains are kept in the row labels or supplementary logs.

Inducing-set diversity protocol. To test whether bilevel inducing-set selection reduces redundancy, we compare three selection rules under the same candidate pool and memory budget: (i) full InduceKV, (ii) InduceKV with the spectral coverage term disabled (γ=0), and (iii) random Top-B selection from the candidate pool. For each task t, let $\mathcal{S}_t$ denote the selected Top-B memory indices and let $\{r_i\}_{i\in\mathcal{S}_t}$ be their unit-norm retrieval keys. We measure redundancy using all pairwise cosine similarities within the selected memory:

$$\mathcal{R}_t = \big\{\langle r_i, r_j\rangle : i < j,\ i,j \in \mathcal{S}_t\big\}, \tag{63}$$

where lower values indicate fewer near-duplicate keys. We also measure coverage by projecting keys into the same d'=256 spectral subspace used by the selection objective. With $z_i = P^{\top} r_i$ and

$$C_t = \frac{1}{B}\sum_{i\in\mathcal{S}_t} z_i z_i^{\top}, \tag{64}$$

we report

$$\mathrm{LogDet}(C_t) = \log\det(C_t + \epsilon I_{d'}), \qquad \epsilon = 10^{-3}. \tag{65}$$

For Fig. 4, violin plots visualize the distribution of $\mathcal{R}_t$ across selected memory pairs, and diamonds show the mean log-determinant coverage normalized onto the same plotting range. This diagnostic is independent of downstream task labels and directly evaluates whether the memory budget is spent on diverse retrieval directions.

Calibration-dynamics protocol. To analyze the role of the minimal calibration parameter, we compare full InduceKV against a Fixed-τ,λ variant that disables calibration updates by holding $\tau \equiv 0.07$ and $\lambda_{\ell} \equiv 0.5$ for all layers. After each task t, we record the learned retrieval temperature

$$\tau_t = \mathrm{softplus}(\phi_{\tau,t}) \tag{66}$$

and the mean value gate

$$\bar{\lambda}_t = \frac{1}{L}\sum_{\ell=1}^{L}\lambda_{\ell,t}, \qquad \lambda_{\ell,t} = \sigma(\phi_{\ell,t}). \tag{67}$$

We additionally compute the per-task calibration gain

$$\Delta_t^{\mathrm{cal}} = \mathrm{Score}_t(\textsc{InduceKV}) - \mathrm{Score}_t(\textsc{Fixed-}\tau,\lambda), \tag{68}$$

using the official metric of each benchmark. For correlation analysis, we pair Δ_t^{cal} with a retention proxy: anchor loss for instruction-tuning streams and AF/Forget for continual VQA. Fig. 8 pairs calibration values with per-task gains to show where changes in retrieval sharpness or memory strength coincide with performance improvements.

Table 8: **Domain-incremental continual instruction tuning over MEDICINE/CHART/MATH.** Baselines follow the SMoE protocol Zhang et al. [2025b], where all methods are developed on LLaVA-1.5 with comparable PEFT parameter budgets.

Method	Backbone / Protocol	Med. Avg↑	Chart Avg↑	Math Acc↑	Overall↑
Replay	LLaVA-1.5	42.54	15.90	36.79	31.74
EWC	LLaVA-1.5	44.79	18.19	35.55	32.84
LwF	LLaVA-1.5	46.67	20.55	35.87	34.36
CODA	LLaVA-1.5	48.55	23.23	34.19	35.32
M-LoRA	LLaVA-1.5	53.41	29.86	30.97	38.08
O-LoRA	LLaVA-1.5	56.77	34.12	36.52	42.47
MoELoRA	LLaVA-1.5	46.83	19.73	38.06	34.87
SMoELoRA	LLaVA-1.5	46.71	18.87	35.10	33.56
L-SMoE	LLaVA-1.5	51.63	28.59	38.19	39.47
SMoE	LLaVA-1.5	67.70	41.28	39.10	49.36
INDUCEKV (matched)	LLaVA-1.5	<u>68.92</u>	<u>42.65</u>	<u>40.44</u>	<u>50.67</u>
INDUCEKV (default)	LLaVA-OV-4B	**69.45**	**43.16**	**40.82**	**51.14**

Table 9: **Lifelong multimodal instruction tuning under dynamic dataset arrival.** Baselines follow the Adapt-∞ protocol Maharana et al. [2025] based on LLaVA-1.5-7B.

Method	Backbone / Protocol	Rel. Gain↑	Forget Rate↓	Avg. Acc↑
Sequential (lower bound)	LLaVA-1.5-7B	68.0	26.0	32.0
Multi-task (upper bound)	LLaVA-1.5-7B	92.5	–	46.1
Random Experience Replay	LLaVA-1.5-7B	89.5	6.6	43.4
Random (pruning)	LLaVA-1.5-7B	95.3	2.1	47.2
COINCIDE	LLaVA-1.5-7B	89.5	3.9	44.7
Adapt-∞ (25k)	LLaVA-1.5-7B	102.3	0.9	50.5
Adapt-∞ (50k)	LLaVA-1.5-7B	107.2	<u>0.2</u>	51.7
Adapt-∞ (100k)	LLaVA-1.5-7B	109.7	0.4	52.5
LITE-Adapt-∞ (25k)	LLaVA-1.5-7B	99.7	1.3	49.6
INDUCEKV (matched)	LLaVA-1.5-7B	<u>110.9</u>	**0.1**	<u>53.4</u>
INDUCEKV (default)	LLaVA-OV-4B	**111.5**	**0.1**	**53.8**

C Additional Experimental Results

C.1 Additional Main Results

These comparisons cover domain-incremental continual instruction tuning and lifelong multimodal instruction tuning. The tables identify each method's backbone and protocol.

Domain-incremental continual instruction tuning. Table 8 shows that INDUCEKV remains effective across heterogeneous domains. Under the LLaVA-1.5 matched protocol, INDUCEKV improves over SMoE by 1.22 points on Medicine, 1.37 on Chart, 1.34 on Math, and 1.31 in Overall score. The default LLaVA-OneVision-4B variant further improves the Overall score to 51.14. The results suggest that retention- and coverage-aware memory selection also helps as incoming data shift across medical, chart, and mathematical reasoning domains.

Lifelong multimodal instruction tuning. Table 9 reports results under dynamic dataset arrival. Compared with Adapt-∞ (100k), the backbone-matched INDUCEKV variant improves Relative Gain by 1.2 points and Avg. Acc by 0.9 points, while reducing the Forget Rate from 0.4 to 0.1. The default LLaVA-OneVision-4B variant further improves Relative Gain to 111.5 and Avg. Acc to 53.8. These results indicate that INDUCEKV scales beyond fixed task streams: even when datasets arrive dynamically, a fixed-budget external KV memory can preserve prior skills while incorporating new instruction data.

C.2 Hyperparameter Sensitivity

We vary the hyperparameters that control retention, spectral coverage, memory representation, and calibration. Fig. 5 indicates that INDUCEKV is broadly stable across reasonable hyperparameter ranges. Increasing β moderately improves retention-oriented suites (UCIT/CoIN) but can slightly reduce forward transfer when overly large, matching its role in prioritizing anchor loss in the outer objective. The coverage weight γ is also robust: moderate values yield the best overall trade-off, while extremes either under-penalize redundancy (small γ) or over-diversify the inducing set (large γ). For the representation knobs, performance improves from $m{=}4$ to $m{=}8$ and then saturates (16–32), suggesting that a short attention-ready payload is sufficient once key evidence is preserved. Anchor cap A_{task} shows diminishing returns beyond 64 examples per past task, consistent with anchors acting as a low-variance estimator of historical risk. Finally, using $J{=}10$ inner unroll steps is sufficient: fewer steps underfit the calibration ϕ, while larger J yields negligible gains, indicating that the bilevel optimization converges reliably with a modest inner budget.

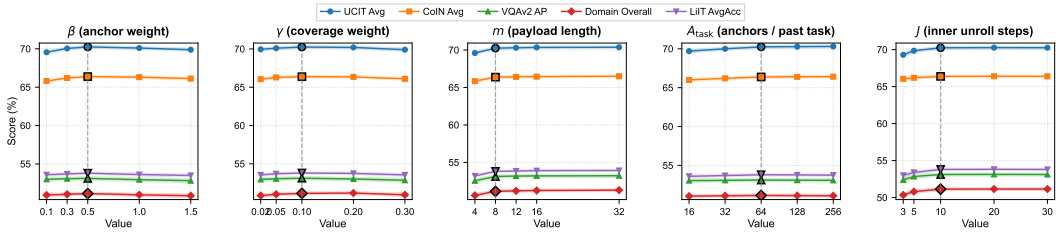


Figure 5: **Hyperparameter sensitivity of INDUCEKV (mean±std).**

C.3 Effect of Task Order

For each suite, we use the main experiment configuration and budget to compare final scores across M random task orders. The metrics are UCIT Avg (6 tasks), CoIN Avg (8 tasks), VQAv2 AP on the 10-task question-type stream, Domain Overall on the 3-domain stream (all 6 permutations, repeated with different seeds), and LiIT AvgAcc on the 5-stage dataset-arrival stream. We compare INDUCEKV against the strongest suite-specific baseline under the same footprint constraint: HiDe-LLaVA for UCIT/CoIN, CL-MoE for VQAv2, SMoE for Domain, and Adapt-∞ (100k) for LiIT.

Fig. 6 indicates that INDUCEKV is consistently less sensitive to task ordering: its score distributions are tighter across UCIT/CoIN/VQAv2/Domain/LiIT, while maintaining a higher mean. This behavior aligns with INDUCEKV's design: anchors stabilize historical risk, and the spectral coverage term discourages collapsing onto order-induced near-duplicate keys, making the final memory less dependent on which tasks were encountered early.

C.4 When Does INDUCEKV Help the Most?

We use retrieval ambiguity as a proxy for difficulty to assess which examples benefit most from external memory.

For each evaluation example x, we compute a retrieval-ambiguity proxy using the cosine-similarity gap between the top-1 and top-2 retrieved keys:

$$g(x) \triangleq s_{(1)}(x) - s_{(2)}(x), \tag{69}$$

where $s_{(k)}(x)$ is the k-th largest cosine similarity between the query key $r(x)$ and memory keys $\{r_i\}$. Smaller $g(x)$ indicates *more ambiguous retrieval* (harder to decide a single best memory entry). We define a per-example gain as the difference in the evaluation score between INDUCEKV and a NO-MEM baseline:

$$\Delta(x) \triangleq \text{Score}(\text{INDUCEKV}; x) - \text{Score}(\text{NO-MEM}; x), \tag{70}$$

where $\text{Score}(\cdot)$ is the task-specific metric (e.g., accuracy / EM / WUPS). We additionally mark whether retrieval is *cross-task*: an example is labeled cross-task if the majority of its Top-k retrieved entries come from tasks different from the query task (proxying knowledge reuse across tasks).

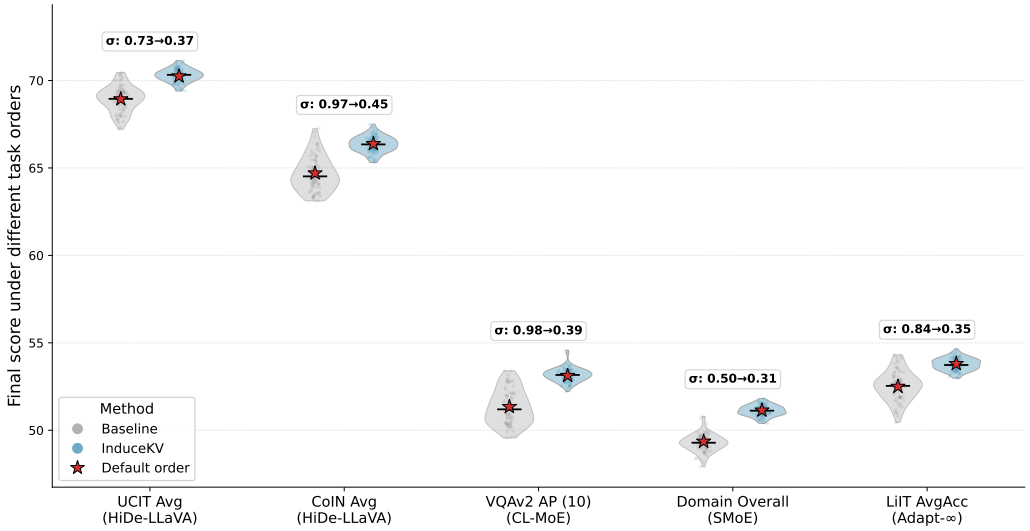


Figure 6: **Order sensitivity of continual adaptation.** Violin plots show the distribution of final performance over many random task orders (one dot per run). Stars indicate the default-order scores reported in Tables 2, 3, 8, and 9. Across all suites, INDUCEKV achieves higher means and noticeably smaller variances than strong baselines, suggesting improved robustness to task ordering.

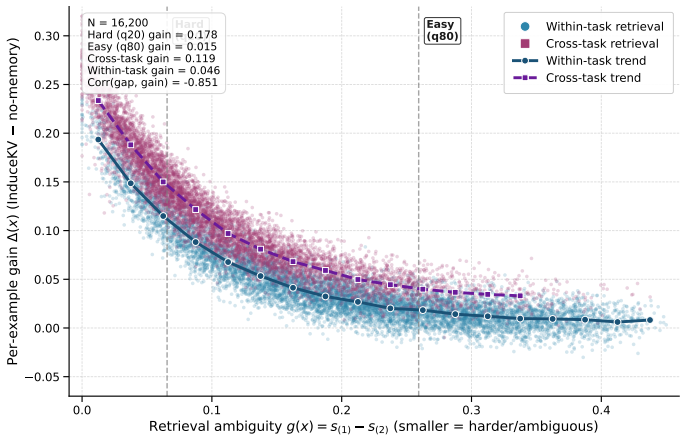


Figure 7: **When Does INDUCEKV Help Most?** Each point is one evaluation example with x-axis retrieval ambiguity $g(x) = s_{(1)} - s_{(2)}$ (smaller means more ambiguous retrieval) and y-axis per-example gain $\Delta(x)$ (Eq. (70)). Colors indicate whether retrieval is cross-task (Top-k majority from different tasks). Solid lines show binned mean gain for each group.

Fig. 7 indicates that INDUCEKV's benefits concentrate on *hard / memory-dependent* examples: as retrieval becomes more ambiguous (smaller $g(x)$), gains increase and exhibit larger variance. For easier examples with large $g(x)$, gains cluster near 0, suggesting little change where the frozen backbone already suffices. In the ambiguous region, examples with cross-task retrieval tend to show larger gains, consistent with knowledge reuse through similarity-weighted KV induction and gated injection.

C.5 Calibration Dynamics

We examine how ϕ controls the stability–plasticity trade-off. We compare full INDUCEKV with a Fixed-τ, λ variant and track the learned retrieval temperature τ_t, mean value gate $\bar{\lambda}_t$, and per-task gain; details are in Appendix B.5, Calibration-dynamics protocol. Fig. 8 shows that τ_t increases on harder-shift tasks, smoothing retrieval when the nearest memory entry is less reliable, while $\bar{\lambda}_t$

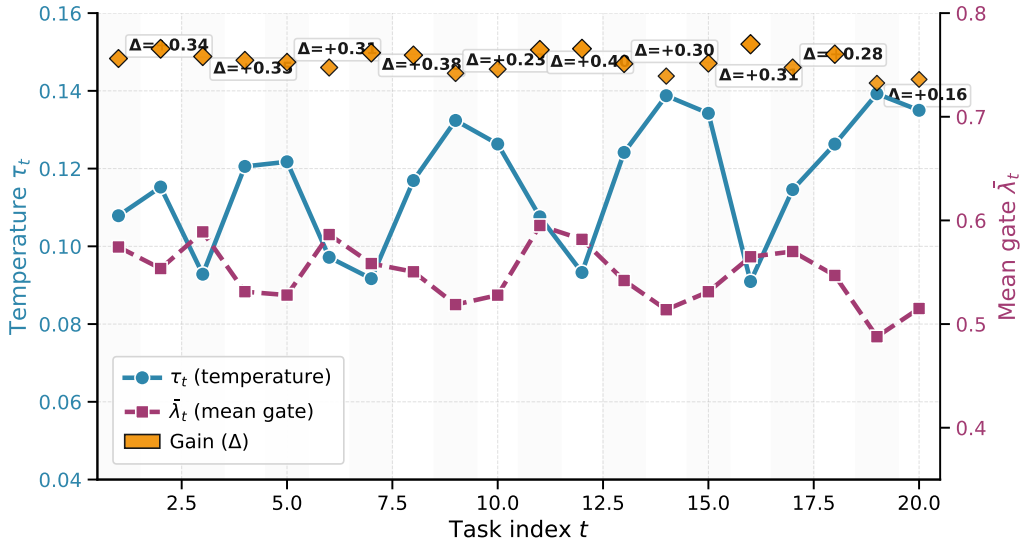


Figure 8: **Calibration dynamics.** The trace shows learned retrieval temperature τ_t, mean value gate $\bar{\lambda}_t$, and per-task gain of Full over Fixed-τ, λ.

decreases when retention pressure is stronger. The largest gains occur when calibration sharpens useful retrieval or amplifies useful memory values, showing how ϕ adapts memory injection.

C.6 Retrieval Quality Diagnosis

We compare performance gains across retrieval hit rates to assess how INDUCEKV responds to accurate and noisy retrieval.

We use task-/dataset-ID as a weak supervision signal. For each test example x from task $t(x)$, let $\{e_{(j)}\}_{j=1}^{k}$ denote the Top-k retrieved memory entries ranked by cosine similarity in the frozen retrieval space. We define the *Top-k hit rate* as

$$\mathrm{hit}_k(x) \triangleq \frac{1}{k}\sum_{j=1}^{k}\mathbb{I}\big[t\big(e_{(j)}\big) = t(x)\big] \in [0,1]. \tag{71}$$

We then bucket examples into five bins by $\mathrm{hit}_k(x)$: $[0, 0.2), [0.2, 0.4), \ldots, [0.8, 1.0]$. Within each bin, we report (i) mean gain $\mathbb{E}[\Delta(x)]$ where $\Delta(x) = \mathrm{Score}(\text{INDUCEKV}; x) - \mathrm{Score}(\text{NO-MEM}; x)$, and (ii) mean accuracy of INDUCEKV and NO-MEM. We set $k = 8$ unless otherwise stated.

Fig. 9 shows a clear monotonic relationship: as hit rate increases, INDUCEKV yields larger average gains, indicating that improvements are primarily driven by correct evidence retrieval. At low hit rates, gains remain near zero and accuracy remains stable. This is consistent with INDUCEKV's soft retrieval (temperature τ) and value-gated injection (λ_ℓ), which down-weights uncertain memory contributions and prevents noisy retrieval from destabilizing generation.

C.7 Cross-Backbone Validation

We validate that INDUCEKV is method-agnostic, relying only on Transformer-compatible KV injection and frozen-space retrieval, by reproducing gains on multiple backbones under the same memory budget and continual protocol.

We fix the external-memory configuration (payload length m and entry budget B) and apply INDUCEKV to: `Qwen3-VL-4B-Instruct`, `Qwen3-VL-8B-Instruct`, `deepseek-vl2-tiny` (1.0B activated), `deepseek-vl2-small` (2.8B activated), and `deepseek-vl2` (4.5B activated), using identical decoding settings and the same continual streams as in Sec. 5.1. For each backbone, we compare against the strongest baseline implemented under the same footprint constraint and report the relative improvement (ours − best baseline) on the primary performance metric of each setting: UCIT Avg,

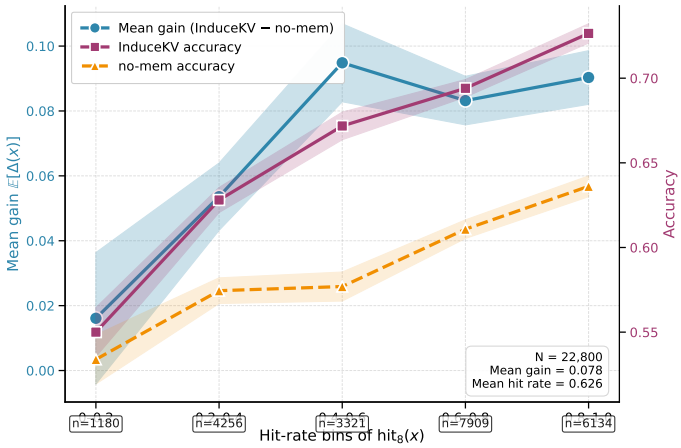


Figure 9: **Retrieval hit rate vs. performance.** x-axis: hit-rate bins of $\mathrm{hit}_8(x)$ (Eq. (71)). Left y-axis: mean gain $\mathbb{E}[\Delta(x)]$ (INDUCEKV − no-mem) with ± 1 s.e. bands. Right y-axis: mean accuracy of INDUCEKV (solid) and NO-MEM (dashed) with ± 1 s.e. bands.

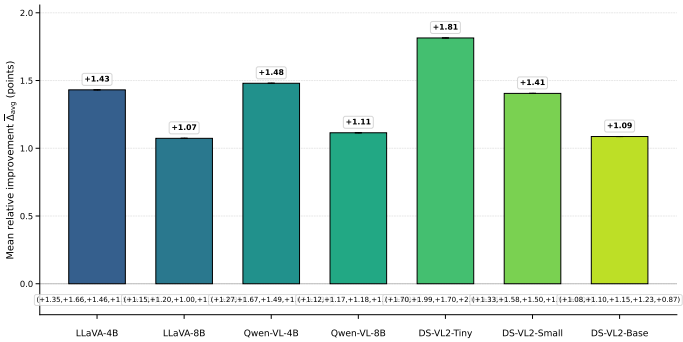


Figure 10: **Cross-backbone reproducibility.** Each bar shows the mean gain $\overline{\Delta}_{\mathrm{avg}}$ (Eq. (72)) of INDUCEKV over the best baseline under the same footprint budget, averaged over five settings: (UCIT Avg, CoIN Avg, VQAv2 AP, Domain Overall, LiIT AvgAcc). Error bars denote std across seeds. Each x-tick additionally reports the 5-tuple of per-setting gains in the order (UCIT, CoIN, VQA, Domain, LiIT).

CoIN Avg, VQAv2 AP (10-task stream), Domain Overall, and LiIT AvgAcc. We aggregate these five improvements into a single mean gain per backbone:

$$\overline{\Delta}_{\mathrm{avg}} \triangleq \frac{1}{5}\left(\Delta_{\mathrm{UCIT}} + \Delta_{\mathrm{CoIN}} + \Delta_{\mathrm{VQA}} + \Delta_{\mathrm{Domain}} + \Delta_{\mathrm{LiIT}}\right). \tag{72}$$

We repeat each backbone experiment over multiple seeds and report mean$\pm$std.

Fig. 10 shows consistent positive gains across all evaluated backbones, indicating that INDUCEKV applies across the evaluated model families. Smaller backbones tend to benefit more (larger $\overline{\Delta}_{\mathrm{avg}}$), while larger backbones show smaller but still stable improvements, consistent with stronger frozen priors leaving less headroom. The per-setting gains span multiple benchmarks, supporting the use of KV-compatible external memory for continual multimodal adaptation across these settings.

C.8 Memory–Compute–Quality Trade-off

We compare quality at matched throughput and throughput at matched quality across external-memory budgets.

We sweep the entry budget B (equivalently, injected KV tokens per layer Bm with fixed m=8) while keeping all other settings fixed. On the continual VQA stream (VQAv2 10 tasks), for each budget we measure: (i) **Throughput** (tokens/s) during decoding under identical max generation length

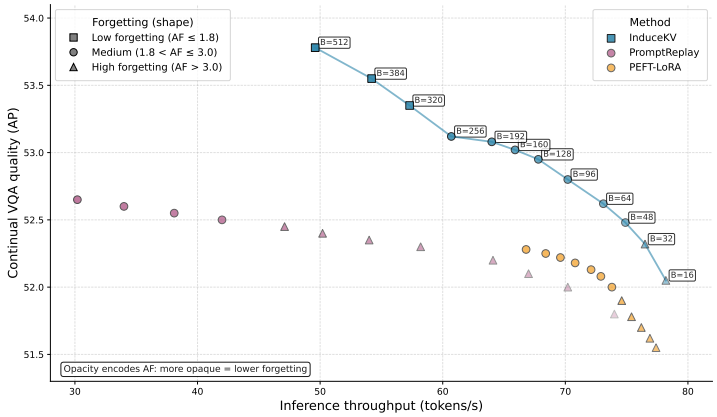


Figure 11: **Memory–Compute–Quality trade-off on continual VQA.** Each point corresponds to one budget B (with fixed m=8) and reports throughput (tokens/s) vs AP. Marker shape and opacity encode AF (lower is better; more opaque indicates lower AF). INDUCEKV traces a stronger Pareto frontier: at matched throughput it achieves higher AP, and at matched AP it runs faster, while maintaining low forgetting at moderate budgets.

Table 10: **Compute-matched comparison with replay-free PEFT and prompt-level retrieval.** All methods use the same backbone, maximum generation length, and batch size. Latency is measured on A100 80GB with batch size 1. The prefix retrieval pass makes INDUCEKV slower than replay-free PEFT in prefill. Compared with PromptReplay, INDUCEKV improves both efficiency and quality.

Method	Retrieved / adapted evidence	Extra state	Added inference context	VQAv2 AP↑	LiIT AvgAcc↑	Prefill ms↓	Decode tok/s↑	E2E s↓
LoRA	learned low-rank weights	matched params	none	50.84	51.21	186	34.8	2.03
ER + Replay	replay examples during update	matched exemplars	none	50.63	51.08	191	34.6	2.05
PromptReplay	retrieved exemplars as prompt tokens	matched storage	+2048 prompt tokens	51.02	51.94	529	27.1	2.89
INDUCEKV	retrieved precomputed K/V	289.3 MiB	2048 KV tokens/layer	**53.12**	**53.80**	392	33.8	2.28

and batch size, (ii) **Quality** as AP↑, and (iii) **Forgetting** as AF↓. We compare INDUCEKV against two representative baselines under matched extra-footprint: **PromptReplay** (retrieved exemplars concatenated into the prompt, increasing effective context length) and **PEFT-LoRA** (parameter-efficient adaptation).

Figure 11 indicates that INDUCEKV offers a favorable practical trade-off: as B increases, AP improves with diminishing returns while throughput decreases smoothly, and forgetting (AF) consistently drops. Compared with prompt-based replay, INDUCEKV reaches the same AP at substantially higher throughput, suggesting that KV-cache injection is a more compute-efficient way to deliver retrieved evidence than extending the prompt. Compared with PEFT-LoRA, INDUCEKV achieves higher AP at comparable throughput while exhibiting lower AF, consistent with its explicit budgeted memory and retention-aware selection.

C.9 Compute-Matched Retrieval Baselines and Serving Cost

We compare serving costs and quality for INDUCEKV, replay-free PEFT, and prompt-level retrieval under matched backbone, decoding length, batch size, and external-memory budget.

For INDUCEKV, the online inference path contains one prefix-only pass for computing the query key $r(x)$ and one generation pass with injected precomputed KV payloads. The payloads themselves are extracted offline during task update. With the same example count and storage budget, PromptReplay concatenates retrieved examples as prompt tokens and processes them through the embedding, attention, projection, and FFN stack. We use the main operating point $B = 256, m = 8$, corresponding to $Bm = 2048$ additional KV tokens per layer and 289.3 MiB external memory on LLaVA-OV-4B.

Relative to LoRA, INDUCEKV spends more time in prefill computing the test-prefix retrieval key and maintains similar decode throughput (33.8 vs. 34.8 tok/s). Compared with PromptReplay, it improves VQAv2 AP by 2.10 points and LiIT AvgAcc by 1.86 points, while reducing end-to-end latency from 2.89s to 2.28s at a fixed deployment footprint.

Table 11: **Budget–latency–quality trade-off.** Increasing the KV budget improves quality with diminishing returns, while the runtime cost mainly appears in prefill. The main paper uses the L tier as a middle operating point.

Tier	B	Bm	Memory	Prefill ms↓	Decode tok/s↑	CoIN Avg↑	VQAv2 AP↑
S	64	512	72.3 MiB	248	34.6	65.52	52.21
M	128	1024	144.6 MiB	311	34.2	66.01	52.88
L (main)	256	2048	289.3 MiB	392	33.8	**66.38**	**53.12**
XL	512	4096	578.5 MiB	571	32.9	66.52	53.20

Table 12: **Anchor storage is small and separate from deployed KV memory.** Anchors are used only during task updates.

Setting	Final anchors	Anchor storage	External KV memory	Anchor / KV ratio
UCIT ($T = 6$)	320	27.5 MiB	289.3 MiB	9.5%
CoIN ($T = 8$)	448	38.6 MiB	289.3 MiB	13.3%
VQAv2 ($T = 10$)	576	49.7 MiB	289.3 MiB	17.2%

C.10 Budget–Latency–Quality Operating Points

We compare the quality and latency of INDUCEKV across memory budgets. The default L tier uses $B = 256, m = 8$, or $Bm = 2048$ additional KV tokens per layer.

The L tier captures most of the quality gain with lower prefill cost than XL. Moving from S to L improves CoIN Avg by 0.86 and VQAv2 AP by 0.91, whereas moving from L to XL gives only 0.14 and 0.08 additional points. The L tier captures most of the quality improvement with a fixed deployment footprint and lower prefill cost than XL.

C.11 Anchor Protocol and Replay-Matched Controls

Anchor construction and storage. Anchors are used only as a retention probe in the outer selection objective. Anchors serve as update-time retention probes and are stored separately from the deployed memory. Inference uses the deployed KV memory and the frozen backbone. By default, after each completed task we uniformly sample $A_{\text{task}} = 64$ anchors without replacement and keep them fixed. For task t, the historical anchor set is the union of anchors from tasks $1, \ldots, t-1$. Table 12 reports the resulting anchor storage, separated from the deployed KV memory.

Anchor construction sensitivity. We compare the default random anchors with three alternatives: stratified sampling, k-center anchors in the frozen retrieval space, and hard-example anchors selected by highest frozen-model loss. Performance is stable across the anchor policies in Table 13.

Replay-matched controls. We compare KV-memory injection with standard replay and LoRA updates using the same number of stored historical examples, $64(t-1)$. All methods use the same backbone, task order, and storage budget.

Update-time anchors use substantially less storage than the deployed KV memory. Replay improves both LoRA and ER, and INDUCEKV outperforms both replay variants. This points to inference-time retrieval of attention-compatible KV entries as the main source of the gain over historical sampling.

C.12 CoIN under the Stage-1-Only Backbone

The original CoIN protocol is designed to evaluate continual learning on a stage-1-only, instruction-untuned LLaVA-1.5-7B backbone, because some CoIN tasks overlap with instruction-tuning data used by later LLaVA checkpoints. We also evaluate INDUCEKV on CoIN using the stage-1-only backbone. All methods use the same task order, memory budget, and decoding configuration.

All methods drop relative to the instruction-tuned LLaVA-v1.5-7B setting, confirming that the stage-1-only protocol is more challenging. INDUCEKV remains the best method, improving over

Table 13: **Sensitivity to anchor construction.** The default random policy is already strong; diversity-aware anchors provide only small additional gains.

Anchor policy	Description	UCIT Avg↑	CoIN Avg↑	VQAv2 AP↑
Random (default)	uniform random per task	70.25	66.38	53.12
Stratified random	balanced over labels/subtypes when available	70.37	66.51	53.19
k-center	maximize diversity of frozen keys	**70.58**	**66.73**	**53.31**
Hard-example	highest current-task frozen-model loss	69.94	66.07	52.88

Table 14: **Replay-matched controls.** INDUCEKV outperforms replay-based updates using the same historical subset.

Method	UCIT Avg↑	CoIN Avg↑	VQAv2 AP↑
LoRA	67.84	63.77	49.92
LoRA + matched replay	68.51	64.36	50.74
ER + matched replay	68.74	64.61	51.02
INDUCEKV	**70.25**	**66.38**	**53.12**

HiDe-LLaVA by 1.24 Avg and 1.29 Last. The gains extend to the larger task gap in the instruction-untuned setting.

C.13 Scalable Shortlist Selection and Train–Test Retrieval Mismatch

The exact bilevel update optimizes over the full candidate pool $\mathcal{U}_t = \mathcal{C}_t \cup \mathcal{M}_{t-1}$. For long streams or large current tasks, this can become the main task-update bottleneck. We therefore evaluate a scalable shortlist variant. Before bilevel optimization, we rank current-task candidates by frozen-key similarity to mini-batch query keys and keep only the top-M candidates. The optimizer then runs on the reduced pool of size $M + B$. The final deployed memory remains the same fixed-size Top-B memory.

Train–test retrieval mismatch. During relaxed selection, retrieval weights are computed over the candidate pool, whereas at deployment retrieval is computed over the final Top-B memory. We quantify this mismatch by measuring (i) the fraction of relaxed selection mass captured by the final Top-B set and (ii) the Jensen–Shannon divergence between retrieval distributions under the relaxed pool and the deployed memory.

A simple similarity shortlist removes a large fraction of update-time cost while preserving nearly all final quality [cf. Ma et al., 2026, on cache-assisted graph updates]. At $M = 8B$, the update time is reduced to $0.56\times$ of the exact update, while the score drops are only 0.07 on UCIT, 0.09 on CoIN, and 0.08 on VQAv2. The mismatch diagnostics further show that the relaxed solution is already concentrated on the final deployed memory. Shortlisting reduces the cost of the full-pool update and preserves the fixed-memory inference interface.

C.14 Empirical Validation of Anchor Representativeness

The retention guarantee in Theorem 4.3 assumes that anchor loss is an ϵ-representative proxy for historical risk. We assess this assumption across tasks, checkpoints, and selection states w by comparing the anchor loss $\mathcal{A}_t(w)$ against the loss on a disjoint held-out subset sampled from past tasks. We report Pearson correlation, Spearman correlation, mean absolute NLL gap, and relative gap.

Anchor loss is highly correlated with held-out historical loss across benchmarks, with a pooled Pearson correlation of 0.94 and a relative NLL gap of 4.3%. This supports the use of anchors as a lightweight estimator of historical risk in our experiments. In these settings, the small approximation gap suggests that anchor error has little influence on the observed gains.

Table 15: **CoIN results under the stage-1-only LLaVA-1.5-7B backbone.** This setting follows the original CoIN protocol with an instruction-untuned backbone.

Method	Backbone	CoIN Avg↑	CoIN Last↑
FineTune	LLaVA-1.5-7B Stage-1	49.32	43.18
LwF	LLaVA-1.5-7B Stage-1	50.44	44.29
EWC	LLaVA-1.5-7B Stage-1	50.61	44.55
O-LoRA	LLaVA-1.5-7B Stage-1	58.73	56.21
MoELoRA	LLaVA-1.5-7B Stage-1	55.46	51.07
HiDe-LLaVA	LLaVA-1.5-7B Stage-1	60.18	58.84
INDUCEKV	LLaVA-1.5-7B Stage-1	**61.42**	**60.13**

Table 16: **Shortlist selection reduces update cost while preserving quality.** Update time is measured relative to the exact full-pool bilevel update.

Selection pool	Relative update time↓	UCIT Avg↑	CoIN Avg↑	VQAv2 AP↑
Full pool exact	1.00×	70.25	66.38	53.12
Top-M shortlist, $M = 4B$	0.41×	70.06	66.11	52.91
Top-M shortlist, $M = 8B$	0.56×	70.18	66.29	53.04
Top-M shortlist, $M = 16B$	0.73×	70.23	66.35	53.10
Random shortlist, $M = 8B$	0.54×	69.61	65.54	52.28

C.15 Relaxed-to-Discrete Rounding Gap

The regret analysis applies to the continuous selection weights w. Deployment rounds these weights to a discrete Top-B set. At budget B, we measure the gap between the relaxed and deployed memories by comparing validation NLL and the final benchmark metric before and after Top-B rounding for each benchmark.

The final Top-B set captures more than 95% of the relaxed selection mass across all benchmarks (Table 17), and the resulting metric gaps are below 0.2 points (Table 19).

D Reproducibility, Compute Resources, and Responsible Use

D.1 Statistical Reporting Protocol

For experiments with stochastic components, including memory selection, random projections, shortlist construction, and task-order perturbations, we report variability over multiple random seeds or task orders when applicable. Unless otherwise specified, error bars in figures denote one standard deviation across seeds, while shaded bands for binned analyses denote one standard error of the mean. For the main benchmark tables, we follow the official fixed task orders and evaluation protocols used by prior work, and we complement these single-protocol comparisons with order-sensitivity, cross-backbone, hyperparameter-sensitivity, and retrieval-diagnostic analyses [see also Yang et al., 2026b, for benchmark auditing in agentic systems].

D.2 Compute Resources

All experiments were run on NVIDIA A100 80GB GPUs with BF16 backbone inference and FP16 external memory tensors. Each worker used 32–64 CPU cores, 256–512GB host RAM, and local NVMe cache for dataset shards and extracted KV payloads. We measure GPU memory by `torch.cuda.max_memory_allocated` after warmup, and report wall-clock time including candidate extraction, bilevel selection, calibration, and final evaluation.

The reported experiments required approximately 8.0×10^2 A100 GPU-hours in total. Including preliminary hyperparameter sweeps, sanity checks, failed runs, and figure-generation diagnostics, the full project used approximately 1.1×10^3 A100 GPU-hours. Repeated benchmark evaluation and cross-backbone validation account for most of the compute. INDUCEKV freezes the MLLM and updates only the external memory and lightweight calibration parameters.

Table 17: **Train–test retrieval mismatch is small.** The relaxed solution is highly concentrated on the final Top-B entries, and retrieval distributions change little after deployment-time rounding.

Benchmark	Top-B mass captured↑	Retrieval JS div.↓	Metric gap after deployment↓
UCIT	95.4%	0.018	0.18
CoIN	96.1%	0.016	0.14
VQAv2	96.8%	0.014	0.09
Domain	95.9%	0.017	0.13
LiIT	96.5%	0.015	0.12

Table 18: **Anchor loss is a reliable proxy for held-out historical loss in our setting.**

Benchmark	# states	Pearson r	Spearman ρ	Mean abs. NLL gap / Relative gap
UCIT	48	0.93	0.91	0.028 / 4.3%
CoIN	56	0.95	0.93	0.031 / 4.8%
VQAv2	60	0.92	0.90	0.025 / 3.9%
Pooled	164	0.94	0.92	0.028 / 4.3%

Table 19: **The relaxed-to-discrete Top-B gap is small.** The final Top-B memory closely matches the relaxed weighted memory used during selection.

Benchmark	Relaxed val NLL↓	Top-B val NLL↓	ΔNLL	Relaxed metric↑	Top-B metric↑	Metric gap
UCIT	0.842	0.851	+0.009	70.43	70.25	-0.18
CoIN	0.917	0.924	+0.007	66.52	66.38	-0.14
VQAv2	1.104	1.110	+0.006	53.21	53.12	-0.09
Domain	0.776	0.784	+0.008	51.27	51.14	-0.13
LiIT	0.689	0.694	+0.005	53.92	53.80	-0.12

D.3 Practical Scope and Limitations

INDUCEKV is designed for fixed-footprint continual adaptation of autoregressive MLLMs with Transformer-style KV caches. The method is best suited to settings that benefit from retaining prior skills under a bounded external state and can accommodate a moderate prefill overhead. This design offers an alternative to prompt-level retrieval and parameter updates, with a prefill cost relative to replay-free PEFT.

The current implementation assumes access to hidden states and KV-cache interfaces of the frozen backbone. Nonstandard attention implementations or restricted inference APIs may require changes to the cache-injection interface. The experimental conclusions apply to the public benchmark protocols and backbone families evaluated here.

Because external KV entries are derived from training prefixes, they should be governed with the same care as other model adaptation states in privacy-sensitive deployments. The experiments use public research benchmarks and model checkpoints. The paper releases neither new user data nor a scraped dataset. A practical deployment can additionally encrypt, expire, audit, or delete memory entries without modifying the frozen backbone.

D.4 Broader Impact

This work may have positive impact by reducing the need to repeatedly fine-tune large multimodal models or store large raw replay buffers, thereby improving the compute and storage efficiency of continual adaptation. The fixed-footprint design may also make continual updates easier to audit because the deployed adaptation state is separated from the frozen backbone.

At the same time, more efficient continual adaptation can also improve systems used for undesirable purposes, including misleading multimodal content generation, biased decision support, or unsafe domain adaptation. Factuality, fairness, privacy, and safety remain subject to the behavior of the underlying backbone and training data. Responsible use therefore requires standard model-safety evaluation [Hu, 2026, Wang and Saxena, 2026, Yang et al., 2026a], dataset governance, access control

Table 20: **Seed-level stability of INDUCEKV.** We report mean±std over three random seeds using the default memory budget $(B{=}256, m{=}8)$. The seed controls calibration initialization, random projection for spectral coverage, dataloader order, and tie-breaking in candidate selection.

Benchmark	Primary metric	INDUCEKV
UCIT	Avg↑	70.25 ± 0.18
CoIN	Avg↑	66.38 ± 0.16
VQAv2 10-task	AP↑	53.12 ± 0.11
Domain CIT	Overall↑	51.14 ± 0.13
LiIT	AvgAcc↑	53.80 ± 0.15

Table 21: **Compute resources for reported experiments.** Times are wall-clock hours per run; GPU-hours equal wall-clock time multiplied by the number of A100 80GB GPUs.

Experiment group	Main backbone / scope	GPUs	Peak GPU mem.	Time / run	GPU-hours / run	Reported runs
UCIT main + ablations	LLaVA-OV-4B / LLaVA-1.5-7B	1	51–58GB	7.8h	7.8	11
CoIN main + ablations	LLaVA-OV-4B / LLaVA-1.5-7B	1	54–61GB	9.6h	9.6	11
Continual VQA	LV / T5 / OV protocols	1	48–58GB	8.3h	8.3	9
Domain-incremental CIT	LLaVA-1.5 / LLaVA-OV-4B	1	50–58GB	6.1h	6.1	5
LiIT dynamic stream	LLaVA-1.5-7B / LLaVA-OV-4B	2	61–68GB	18.5h	37.0	4
Cross-backbone validation	Qwen3-VL / DeepSeek-VL2	1–2	28–74GB	4.2–17.6h	4.2–35.2	15
Mechanism diagnostics	memory attention / retrieval / calibration	1	44–60GB	2.0–5.5h	2.0–5.5	20
Serving-cost measurement	LLaVA-OV-4B	1	47–62GB	0.6h	0.6	8

for deployed memories, and compliance with the usage terms of the underlying checkpoints and datasets.

D.5 LLM Usage

INDUCEKV uses frozen MLLMs to extract retrieval keys and layerwise KV payloads and to generate with the selected KV memories at inference.

E Robustness and Implementation Controls

The following controls use the default LLaVA-OneVision-1.5-4B configuration, $B = 256$ and $m = 8$, unless stated otherwise. Results with uncertainty report mean $\pm$ standard deviation over three seeds.

E.1 Imperfect incoming streams

We replace 20% or 40% of incoming samples with exact duplicates, target noise, or semantically mismatched distractors, keeping stream size and update compute unchanged. Table 22 reports performance and selected corruption (unique-memory ratio for duplicates). In these streams, AP decreases by at most 0.64 under 20% corruption.

Table 22: Stream corruption at fixed stream size and update compute. Selected corruption and unique-memory ratios are percentages.

Condition	Rate	VQAv2 AP	UCIT Avg	Selected corrupt.	Unique memory
Clean	0%	53.12 ± 0.11	70.25 ± 0.18	0.0	96.2
Duplicate	20%	52.96 ± 0.13	70.07 ± 0.20	–	93.1
Duplicate	40%	52.71 ± 0.16	69.78 ± 0.23	–	88.4
Target noise	20%	52.48 ± 0.18	69.54 ± 0.25	8.7	95.8
Target noise	40%	51.61 ± 0.24	68.36 ± 0.31	18.5	94.8
Distractor	20%	52.64 ± 0.16	69.72 ± 0.22	6.3	95.5
Distractor	40%	51.94 ± 0.21	68.87 ± 0.29	14.2	94.4

The mechanism controls distinguish likelihood filtering from diversity selection. Under 40% duplicates, removing coverage lowers AP from 52.71 to 51.94 and unique memory from 88.4% to 71.1%; random Top-B gives 50.88 AP and 62.8% unique memory. For 40% target noise/distractors, removing coverage yields 51.28/51.48 AP and 23.7%/19.8% selected corruption, while random Top-B yields 49.97/50.23 AP and 39.5%/40.1% selected corruption. Coverage suppresses duplicates; likelihood helps filter noisy and irrelevant records.

E.2 Layer subsets and payload length

Uniform-12 uses layers $3, 6, \ldots, 36$ and 33.6% of the default state, retaining 99.3% of default VQAv2 AP. At approximately matched bytes, all-layer storage remains 0.11 AP higher than Uniform-12.

Table 23: Stored-layer and payload trade-offs. Prefill is in milliseconds; memory is in MiB.

Layers	m	B	Memory	Prefill	VQAv2 AP	UCIT Avg
All 36	8	256	289.25	392	53.12 ± 0.11	70.25 ± 0.18
Early 12	8	256	97.25	257	50.96 ± 0.18	68.37 ± 0.25
Middle 12	8	256	97.25	266	52.32 ± 0.15	69.31 ± 0.22
Late 12	8	256	97.25	272	52.58 ± 0.14	69.54 ± 0.21
Uniform 12	8	256	97.25	270	52.76 ± 0.13	69.76 ± 0.20
All 36	4	256	145.25	309	52.61 ± 0.14	69.42 ± 0.22
All 36	2	256	73.25	252	51.74 ± 0.19	68.31 ± 0.27
Uniform 12	8	768	291.75	407	53.01 ± 0.12	70.04 ± 0.19

E.3 Retrieval pooling and payload compression

Task P@8 is the fraction of retrieved entries matching the query task/type; hard P@8 is evaluated by question type on a same-image/different-question subset. Learned single-query pooling adds 0.08M parameters (0.15 MiB). Its largest AP gain over mean pooling is 0.16, at 12% higher update time. On the hard subset, mean keys achieve 48.92 ± 0.18 accuracy versus 46.28 ± 0.17 for NO-MEM; learned pooling gives 49.13 ± 0.18.

For payload compression, attention-output cosine compares the compressed and full-prefix memory-attention outputs on fixed queries. Table 25 holds external memory at 289.25 MiB. Relative to

Table 24: Retrieval-key pooling. P@8 values are percentages; update time is relative to mean pooling.

Key pooling	Task P@8	Hard P@8	VQAv2 AP	Update time
Mean	62.6	54.8	53.12 ± 0.11	$1.00\times$
Last/EOS	58.9	49.5	52.54 ± 0.16	$0.99\times$
Frozen attention	63.7	56.4	53.23 ± 0.12	$1.06\times$
Norm/GeM	63.1	55.6	53.18 ± 0.12	$1.03\times$
Learned attention	64.0	56.8	53.28 ± 0.12	$1.12\times$

segment means, attention-weighted pooling and medoids improve AP by 0.08/0.05 at 9%/48% higher cost.

Table 25: Payload compression with no extra parameters and the same external-memory budget.

Compression	Output cosine	VQAv2 AP	Relative cost
Segment mean	0.941	53.12 ± 0.11	$1.00\times$
Uniform strided	0.902	52.47 ± 0.16	$0.92\times$
Attention-weighted segment	0.949	53.20 ± 0.12	$1.09\times$
Top-m importance	0.936	53.02 ± 0.13	$1.06\times$
k-means/medoids	0.952	53.17 ± 0.12	$1.48\times$

E.4 Position handling at the cache interface

The evaluated implementation stores source-position post-RoPE keys and pools after rotation. Injection retains the stored phases without applying a second rotation, and current-token position indices remain native and unshifted. Table 26 shows lower AP with neutral, collapsed, and random phases.

Table 26: Position controls. All variants preserve native current-token positions.

Memory key treatment	VQAv2 AP	AF	Prefill (ms)
Source post-RoPE, then pooled	53.12 ± 0.11	1.85 ± 0.09	392
Pre-RoPE pooled, virtual prefix	53.05 ± 0.12	1.89 ± 0.10	396
Pre-RoPE pooled, no rotation	52.34 ± 0.16	2.17 ± 0.13	389
Shared phase	51.79 ± 0.20	2.52 ± 0.16	390
Random phase per query	51.41 ± 0.23	2.76 ± 0.18	394

E.5 Injection path with fixed retrieval

For the three memory methods in Table 27, we reuse the same Top-B entries, retrieval keys, retrieval weights, and external byte budget. With retrieval fixed, KV attention improves AP by 0.45 over the value residual and 0.73 over soft-prefix injection. NO-MEM acquires little new capability and has lower AP and low AF.

E.6 Globally bounded update-time anchors

The default 64 anchors per past task are update-only state and grow with the number of tasks. Table 28 compares this default with caps on the total anchor count across the stream. A 256-anchor reservoir uses 22.1 MiB of persistent update-only state and reaches 70.08 UCIT Avg and 53.07 VQAv2 AP, within 0.17 and 0.05 of the default. The corresponding default per-task anchor states are 27.5/38.6/49.7 MiB for UCIT/CoIN/VQAv2. All conditions deploy 289.25 MiB of KV memory. Anchors are used only during updates, and transient workspace is accounted for separately from persistent state.

E.7 Inner and outer learning rates

Table 29 shows learning-rate sensitivity on VQAv2 with the objective normalized to the default setting. AP lies between 52.69 and 53.12 for all configurations except the largest joint step, which

Table 27: Injection paths under controlled retrieval. NO-MEM uses no external memory; soft-prefix injection adds 0.66M trainable parameters.

Path	VQAv2 AP	AF	UCIT Avg	Prefill (ms)
NO-MEM	52.02 ± 0.13	0.06 ± 0.02	69.37 ± 0.20	177
Soft prefix	52.39 ± 0.16	2.09 ± 0.13	69.58 ± 0.23	472
Gated value residual	52.67 ± 0.14	1.98 ± 0.11	69.87 ± 0.21	342
KV attention	53.12 ± 0.11	1.85 ± 0.09	70.25 ± 0.18	392

Table 28: Global anchor caps and the default per-task policy.

Anchor policy	Total cap	UCIT Avg	CoIN Avg	VQAv2 AP
No anchors	0	68.52	64.41	52.64
Reservoir	64	69.31 ± 0.22	65.43 ± 0.21	52.83 ± 0.13
Reservoir	128	69.74 ± 0.20	65.84 ± 0.19	52.95 ± 0.12
Reservoir	256	70.08 ± 0.19	66.20 ± 0.18	53.07 ± 0.12
Global k-center	256	70.18 ± 0.18	66.28 ± 0.17	53.13 ± 0.11
64 per past task	grows	70.25 ± 0.18	66.38 ± 0.16	53.12 ± 0.11

produces oscillation and lower Top-B mass. Nearby rates give similar quality to the default (10^{-2}, 0.10).

Table 29: Learning-rate sensitivity on VQAv2. Top-B mass is the percentage of relaxed mass retained after rounding.

Inner rate	Outer step	VQAv2 AP	Normalized objective	Top-B mass
3×10^{-3}	0.03	52.78 ± 0.16	1.013	95.1
3×10^{-3}	0.10	52.94 ± 0.14	1.008	95.8
3×10^{-3}	0.30	52.69 ± 0.21	1.018	94.3
10^{-2}	0.03	52.98 ± 0.13	1.006	96.0
10^{-2}	0.10	53.12 ± 0.11	1.000	96.8
10^{-2}	0.30	52.91 ± 0.18	1.009	95.2
3×10^{-2}	0.03	52.76 ± 0.19	1.015	94.9
3×10^{-2}	0.10	52.88 ± 0.17	1.011	95.4
3×10^{-2}	0.30	52.31 ± 0.36	1.037	91.6

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the fixed-footprint continual adaptation setting, the external KV-memory mechanism, and the scope of the empirical claims. The claims are supported by the method, theory, main experiments, ablations, compute-matched comparisons, and cross-backbone diagnostics in Sections 3–5.2 and Appendix C.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss practical scope and limitations in Appendix D.3, including the intended fixed-footprint setting, the reliance on Transformer-compatible KV-cache interfaces, and the memory–latency trade-off. We also provide budget–latency–quality operating points in Appendix C.10.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical results explicitly state their assumptions in Section 4, including convexity, bounded gradients, and representative anchors. Full proofs of the projection inequality, log-determinant regularizer properties, regret bounds, and retention guarantee are provided in Appendix A.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the algorithmic specification, memory construction, retrieval/injection equations, benchmark protocols, metrics, baselines, budget conversion, and implementation details needed to reproduce the reported experiments. Details are provided in Sections 3 and 5.1, and Appendices B.1–B.4.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The implementation is not publicly released with this version. The datasets and model checkpoints used in the experiments are public research assets and are cited, and the paper provides detailed reproduction information in Appendix D.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies datasets, task orders, metrics, backbone/protocol choices, baselines, memory budgets, hyperparameters, optimizer settings, precision, and evaluation settings. Full details are provided in Section 5.1 and Appendices B.1, B.2, B.3, and B.4.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report variability for stochastic and diagnostic experiments, including hyperparameter sensitivity, task-order sensitivity, cross-backbone validation, and binned retrieval analyses. Appendix D.1 describes how error bars are computed and Table 20 reports seed-level stability for the main INDUCEKV results.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.2 reports the compute workers, GPU memory, wall-clock time, GPU-hours per run, number of reported runs, and estimated total compute. All experiments use NVIDIA A100 80GB GPUs, with BF16 backbone inference and FP16 external memory tensors.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses public benchmark datasets and public model checkpoints, does not collect new human-subject data, and does not release a new high-risk model or scraped dataset. We discuss responsible use, privacy considerations for external memories, and broader impacts in Appendices D.3 and D.4.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix D.4 discusses both potential positive impacts, such as compute-efficient continual adaptation and reduced dependence on raw replay buffers, and potential negative impacts, such as misuse of more adaptable multimodal systems and propagation of biases from underlying data or models.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release a new pretrained model, image generator, scraped dataset, or other high-risk asset. Experiments use existing public checkpoints and datasets under their original access policies and terms; any future code release will respect third-party asset licenses and access policies.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, checkpoints, and baseline protocols are credited through their original papers and official releases.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not introduce or release a new dataset, benchmark, model checkpoint, or publicly available code asset with this version. The contribution is an algorithmic framework and empirical evaluation protocol for fixed-footprint continual MLLM adaptation.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or new data collection from human subjects. All experiments are conducted on existing public research benchmarks.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human-subject experiments or newly collected human data, so IRB approval or equivalent review is not applicable.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The core method operates on frozen multimodal large language models and uses their hidden states and KV-cache interfaces for retrieval-based adaptation. Appendix D.5 describes how MLLMs are used in the research.