MEMSFT: MITIGATING ALIGNMENT TAX WITH AN EXTERNAL PARAMETRIC MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting Large Language Models (LLMs) to specialized domains often incurs an alignment tax, as fine-tuning on domain-specific tasks can cause catastrophic forgetting on general tasks. We propose **MemSFT**, which mitigates the alignment tax by decoupling domain specialization from backbone LLM parameter updates through a plug-and-play parametric memory. Building on Memory Decoder (Cao et al., 2026a), which primarily studies external memory for pretrained LLMs, we develop supervised fine-tuning on external memory to support domain specialization of post-trained LLMs. To selectively invoke domain expertise, we introduce a dynamic memory injector that fuses memory and backbone output distributions at each decoding step. We evaluate MemSFT across three domains, namely biology, geoscience, and law, covering small dense and large mixture-of-experts (MoE) LLMs ranging from 8B to 397B parameters. MemSFT consistently improves domain performance without catastrophic forgetting of general capabilities, whereas full SFT suffers severe forgetting on general tasks. Our results demonstrate a practical path to domain specialization of modern LLMs, enabling memory reuse across multiple backbones with only $0.27\times$ and $0.41\times$ the adaptation FLOPs of full SFT and LoRA for a 397B model.

1 INTRODUCTION

Adapting large language models (LLMs) to perform optimally in specific domains remains a central challenge for their practical use (Ling et al., 2025; Beltagy et al., 2019; Lee et al., 2020). Applications in specialized fields such as biology, geoscience, and law (He et al., 2024; Liu et al., 2025; Yue et al., 2024) require expertise and reliability beyond general-purpose capabilities. Supervised fine-tuning (SFT) (Wei et al., 2021; Chung et al., 2024) and continual pretraining (CPT) (Gururangan et al., 2020) are widely used for domain specialization, but they often lead to catastrophic forgetting, eroding previously acquired capabilities (McCloskey & Cohen, 1989; French, 1999; Parisi et al., 2019). This challenge is especially acute as more frontier models are released only in instruction-tuned or reinforcement-trained form (Zeng et al., 2026; Team et al., 2026; Xu et al., 2026). Further domain fine-tuning can disrupt the alignment established during post-training and substantially degrade general performance (Qi et al., 2024; Ji et al., 2025; Li et al., 2024). This creates a pressing need to incorporate domain expertise into post-trained LLMs without compromising their general capabilities.

Recent studies show that further fine-tuning can substantially impair the knowledge, reasoning, and safety behaviors of post-trained LLMs (Luo et al., 2025; Li et al., 2024; Shi et al., 2025; Zheng et al., 2025; Dong et al., 2024; Kalajdzievski, 2024). Notably, Ji et al. (2025) demonstrate that post-alignment language models exhibit an "elastic" nature: subsequent fine-tuning can disrupt their alignment equilibrium and rapidly shift model behavior back toward the broader distribution shaped by pre-training. These findings highlight the vulnerability of post-training gains to further parameter updates, exposing an alignment tax in which domain specialization comes at the expense of general capabilities. Parameter-efficient methods such as LoRA constrain updates to a low-rank subspace (Hu et al., 2022; Dettmers et al., 2023), but still suffer from systematic catastrophic forgetting (Shuttleworth et al., 2026; Koubbi et al., 2024; Biderman et al., 2024). Although LoRA can retain general performance better than full fine-tuning, this advantage may come with weaker domain learning (Biderman et al., 2024); studies also identify a strong linear relationship between fine-tuning performance and forgetting under LoRA (Kalajdzievski, 2024).

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

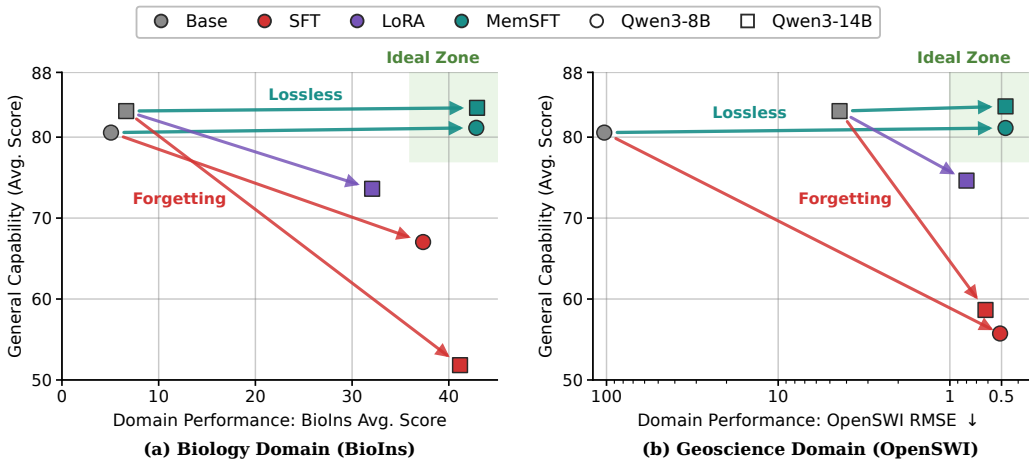


Figure 1: Domain and general performance on Biology-Instructions (BioIns) for biology and OpenSWI for geoscience, evaluated with Qwen3 backbones. Within each domain, MemSFT reuses the same domain-specific memory across different backbone sizes. SFT and LoRA improve domain performance at the cost of general capabilities, while MemSFT achieves strong performance in both domain and general tasks. Full results are reported in Tables 1 and 3 and Appendix D.

We propose **MemSFT**, which achieves efficient domain specialization of frontier post-trained LLMs through an external parametric memory while preserving their general capabilities. Building on Memory Decoder (Cao et al., 2026a), an external memory for pretrained LLMs, MemSFT introduces supervised fine-tuning of external memory for post-trained LLMs. To preserve the backbone LLM's capabilities, we keep it frozen and fine-tune only the memory through two complementary objectives: language modeling and retrieval imitation, using a retrieval datastore constructed from domain SFT data. We further introduce a dynamic memory injector that fuses the output distributions of the memory and the frozen backbone LLM with a context-dependent weight at each decoding step. These designs make MemSFT fundamentally different from existing SFT methods.

We evaluate MemSFT across three domains, namely biology, geoscience, and law, on Qwen3 models up to 235B parameters, and further validate its effectiveness on Intern-S2-Preview-397B. On Qwen3-235B-A22B, MemSFT substantially improves the Biology-Instructions average score from 1.14 to 42.05, a gain of 40.91 points, with no decline in average performance on general tasks. Even on Intern-S2-Preview-397B, which has already undergone extensive scientific training, MemSFT yields a further 3.40-point improvement on Biology-Instructions, approaching the gain from Intern-S1-Pro to Intern-S2-Preview-397B. Comprehensive evaluations covering reasoning, scientific, and multimodal capabilities show that MemSFT preserves existing capabilities even on challenging tasks such as complex multi-step reasoning, molecular design, and multimodal reasoning, with a maximum decrease of only 0.12 points in category-average performance. MemSFT thus offers a practical path to effective domain specialization without sacrificing existing capabilities.

The key advantages of MemSFT are summarized as follows:

- **Domain specialization without forgetting.** MemSFT acquires domain expertise without catastrophic forgetting of general capabilities, mitigating the alignment tax associated with further fine-tuning of post-trained LLMs.
- **Plug-and-play memory reuse.** Domain memory is trained independently of the backbone LLM and can be reused across a series of LLMs without retraining the memory or modifying their parameters.
- **Lower adaptation cost.** For a 397B model, adaptation with a 4B memory requires an estimated 56.13 EFLOPs, only 0.27× the FLOPs of full SFT and 0.41× those of LoRA.
- **Support for sensitive-data applications.** MemSFT allows domain expertise to be delivered through a standalone memory without sharing the underlying data or modifying the backbone LLM, supporting applications involving sensitive domain data.

2 METHOD

2.1 OVERVIEW

We present **MemSFT**, which achieves domain specialization of post-trained LLMs through supervised fine-tuning of an external parametric memory. As illustrated in Figure 2, MemSFT first trains the memory LM to approximate retrieval-based teacher distributions from a token-level datastore (upper-left part, Sections 2.2 and 2.3), then trains the dynamic memory injector with both the backbone and memory frozen (lower-left part, Section 2.3), and finally augments different backbone LLMs with the memory through a dynamic inference mechanism (right part, Section 2.4).

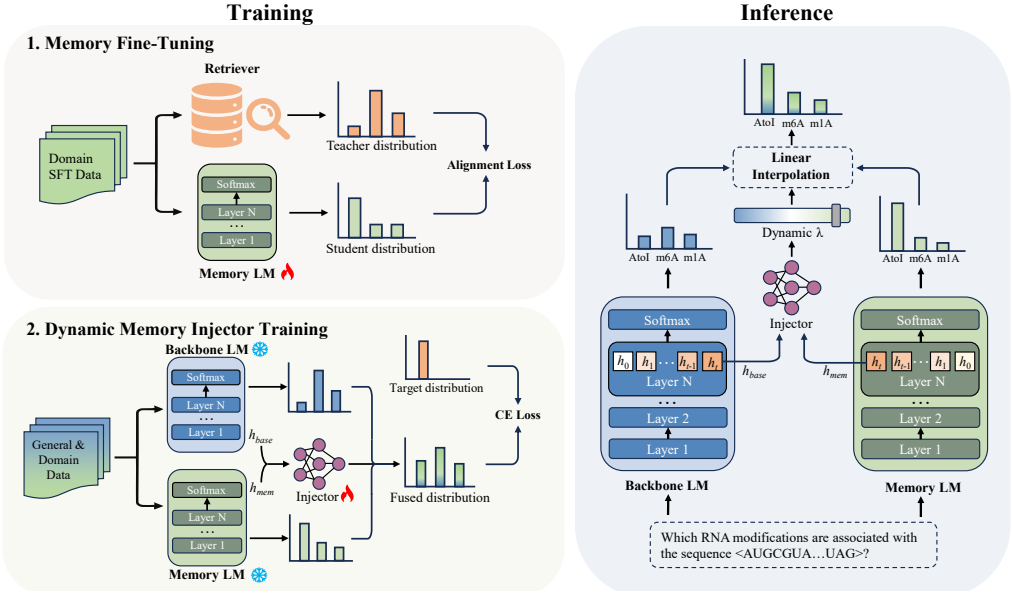


Figure 2: Overview of MemSFT architecture. **Upper-left**: § 2.2 & § 2.3: During memory fine-tuning, a QA-oriented datastore provides retrieval-based teacher distributions, and the memory LM learns to mimic them as a standalone parametric memory. **Lower-left**: § 2.3: During dynamic memory injector training, the backbone LM and memory LM are frozen, and a lightweight injector learns token-wise interpolation weights. **Right**: § 2.4: During inference, the backbone and memory process the same input sequence, and their output distributions are fused for final prediction.

2.2 CONVERTING SFT DATA INTO A RETRIEVAL DATASTORE

MemSFT first constructs a QA-oriented datastore to provide retrieval-based supervision for memory fine-tuning. The datastore acts as a token-level RAG teacher: given a decoding context, it retrieves similar contexts from the domain SFT corpus and converts their corresponding next-token values into a retrieval-based teacher distribution (Khandelwal et al., 2019). Let the domain SFT corpus be

$$\mathcal{D}_{sft} = \{(q^{(i)}, a^{(i)})\}_{i=1}^{N}, \tag{1}$$

where $q^{(i)}$ is a question and $a^{(i)} = (y_1^{(i)}, \ldots, y_{T_i}^{(i)})$ is its answer token sequence. For each answer position t, we define the prefix context $c_t^{(i)} = [q^{(i)}; y_{<t}^{(i)}]$ and the corresponding key $k_t^{(i)} = \phi(c_t^{(i)})$, where $\phi(\cdot)$ extracts the hidden representation from a fixed teacher model. The datastore is constructed as

$$(\mathcal{K}, \mathcal{V}) = \bigcup_{i=1}^{N} \left\{ (k_t^{(i)}, y_t^{(i)}) \right\}_{t=1}^{T_i}. \tag{2}$$

During memory fine-tuning, for each SFT pair $(q, a) \in \mathcal{D}_{sft}$ and each answer position t, we form the teacher-forced decoding context $c_t = [q; y_{<t}]$. We then query the datastore with key $k_t = \phi(c_t)$, retrieve its neighbor set $\mathcal{N}(k_t)$, and construct the following non-parametric teacher distribution:

$$p_{teacher}(y \mid c_t) \propto \sum_{(k_j, v_j) \in \mathcal{N}(k_t)} \mathbf{1}[y = v_j] \exp\big(-d(k_t, k_j)/\tau\big), \qquad y \in \mathcal{V}, \tag{3}$$

where $d(\cdot, \cdot)$ denotes a distance function and τ denotes temperature. This distribution serves as the retrieval-based teacher for memory fine-tuning. Additional datastore and teacher-construction details are provided in Appendix B.

2.3 Two-Stage Supervised Fine-Tuning

Training proceeds in two stages: we first fine-tune the external memory with retrieval-based supervision, then train the dynamic memory injector with both the backbone and memory frozen.

Memory fine-tuning Given the retrieval-based teacher distribution constructed above, we train the memory LM to approximate this distribution at answer-side positions. The retrieval distribution provides a richer target by assigning probability mass to multiple domain-relevant tokens (Xu et al., 2023), encouraging the memory to internalize domain knowledge from the datastore. For an answer token y_t with prefix context $c_t = [q; y_{<t}]$, the memory model predicts $p_{mem}(\cdot \mid c_t)$, while the datastore provides $p_{teacher}(\cdot \mid c_t)$. We combine the Kullback-Leibler divergence between the teacher and memory distributions with the cross-entropy loss:

$$\mathcal{L}_{KL}(c_t) = \mathrm{KL}(p_{teacher}(\cdot \mid c_t) \parallel p_{mem}(\cdot \mid c_t)), \qquad \mathcal{L}_{CE}(c_t) = -\log p_{mem}(y_t \mid c_t). \tag{4}$$

The final memory training loss is

$$\mathcal{L}_{mem}(c_t) = \beta \cdot \mathcal{L}_{KL}(c_t) + (1 - \beta) \cdot \mathcal{L}_{CE}(c_t), \tag{5}$$

where $\beta \in [0, 1]$ controls the balance between the retrieval-based teacher distribution and the gold SFT answer. The KL term aligns the memory LM with the teacher distribution derived from the datastore, while the cross-entropy term anchors it to the gold answer token.

Dynamic memory injector training The external memory serves as a domain expert during generation, but uniformly applying it at every decoding step can perturb the frozen backbone's general reasoning and instruction following behavior. We therefore train a dynamic memory injector, a lightweight two-layer MLP, to predict how much the memory should contribute at each token.

For the same prefix context c_t, the backbone and memory produce distributions $p_{base}(\cdot \mid c_t)$ and $p_{mem}(\cdot \mid c_t)$. At each step, the injector takes the current hidden representations from the two frozen models, together with confidence and entropy features derived from their output distributions, and predicts $\lambda_t \in [0, 1]$. The fused distribution is

$$p_{fused}(\cdot \mid c_t) = (1 - \lambda_t) \cdot p_{base}(\cdot \mid c_t) + \lambda_t \cdot p_{mem}(\cdot \mid c_t). \tag{6}$$

We train the injector on mixed general and domain instruction data. The prediction term is the cross-entropy loss on the fused distribution, and we additionally apply a signed linear regularizer to the memory interpolation weight:

$$\mathcal{L}_{CE}(c_t) = -\log p_{fused}(y_t \mid c_t), \qquad \mathcal{R}(c_t) = s_t \lambda_t, \tag{7}$$

where $s_t < 0$ for domain examples and $s_t > 0$ for general examples. The resulting per-token injector objective is

$$\mathcal{L}_{injector}(c_t) = \mathcal{L}_{CE}(c_t) + \alpha_s \mathcal{R}(c_t). \tag{8}$$

Here, $\alpha_s > 0$ controls the strength of the signed regularizer. Minimizing the signed term pushes λ_t upward on domain examples and downward on general examples, while the cross-entropy term directly optimizes the final next-token distribution. Detailed injector architecture and training configurations are provided in Appendix C.

2.4 Inference

At inference time, both the frozen backbone and the trained memory LM process the same input sequence. For the current decoding prefix $c_t = [x; y_{<t}]$, the injector predicts λ_t, and the final next-token distribution is obtained by Eq. (6).

MemSFT avoids overwriting the backbone LLM: domain-specific updates are confined to the external memory and injector, while the backbone parameters remain frozen. The memory provides domain-specialized token distributions, and the injector controls their contribution at each decoding step. The design is also modular: because memory training is decoupled from a particular target backbone, the trained memory can be reused across frozen LLMs that share a tokenizer. In practice, the backbone provides general reasoning and instruction following capabilities, while the memory strengthens domain-specific predictions.

3 EXPERIMENTAL SETUP

Domain Datasets We evaluate MemSFT on three specialized domains. For biology, we use Biology-Instructions (BioIns), a multi-omics sequence-understanding dataset with 21 tasks (He et al., 2024). For geoscience, we use OpenSWI, a dataset for surface-wave dispersion-curve inversion (Liu et al., 2025). For law, we use DISC-Law-SFT data from DISC-LawLLM (Yue et al., 2024) for specialization and evaluate legal capability on LawBench (Fei et al., 2024). Dataset and evaluation details are in Appendix A.

Backbones and Main Baselines We evaluate Qwen3-8B, Qwen3-14B, Qwen3-32B, and Qwen3-235B-A22B (Yang et al., 2025) on BioIns and OpenSWI, Qwen3-14B on LawBench, and Intern-S2-Preview-397B (Bai et al., 2026) on BioIns. Our three baselines are the original post-trained backbone, full-parameter SFT (Wei et al., 2021; Chung et al., 2024), and LoRA (Hu et al., 2022). For LoRA, we adopt an all-linear target-module configuration following QLoRA (Dettmers et al., 2023), attaching adapters to all attention and MLP projections in each Transformer block.

Anti-Forgetting Baselines We evaluate two anti-forgetting baselines with Qwen3-14B on BioIns. First, we evaluate LoRA + MixTraining(1:1), which mixes BioIns examples with general instruction data during LoRA adaptation, following Liu et al. (2024). Second, we evaluate Wise-FT-style weight-space interpolation (Wortsman et al., 2022) by scaling the domain LoRA module before merging it into the original backbone.

Training Details For Qwen3, we train one 8B memory per domain and reuse it across backbone sizes. The dynamic memory injector is trained with both the backbone and memory frozen, using a small mixture of domain examples and general instruction data from NVIDIA's Nemotron-Post-Training-Dataset-v1 (Bercovich et al., 2025). For Intern, we train a 4B domain memory. Training details are provided in Appendix C.

Evaluation Metrics For BioIns, we report the average task score across all 21 tasks (He et al., 2024). For OpenSWI, we follow the OpenCompass generation-style evaluation built on OpenSWI and report RMSE in the shallow setting (Liu et al., 2025; Cao et al., 2026b), where lower values are better. For law, we report the LawBench average over the evaluated legal subtasks (Fei et al., 2024).

For Qwen3, we evaluate general capability retention on MATH-500 (Lightman et al., 2024), C-Eval (Huang et al., 2023), IFEval (Zhou et al., 2023), MMLU-Redux (Gema et al., 2025), and INCLUDE (Romanou et al., 2025), using lm-evaluation-harness (Gao et al., 2024) and the Qwen3 evaluation settings (Yang et al., 2025). For Intern-S2-Preview-397B, we evaluate on ten benchmarks across reasoning, scientific, and multimodal capabilities, following the Intern-S2 evaluation setting. Evaluation details are provided in Appendices A and D.3.

4 RESULTS

4.1 RESULTS ACROSS DOMAINS

Domain specialization Tables 1, 3, and 4 report results across three specialized domains. Original Qwen3 backbones perform poorly on BioIns and OpenSWI despite their strong general performance. MemSFT improves the Biology-Instructions average by over 540% relative to each original backbone and substantially reduces OpenSWI prediction error. On LawBench, it improves the domain average by 6.64 points, matching the strongest trainable baseline while preserving general performance.

General capability retention Full SFT substantially reduces general performance across all three domains, with severe losses in mathematical reasoning and instruction following. Domain-specific updates can shift the backbone away from the linguistic and instruction following distributions acquired during post-training, particularly with supervision dominated by biological sequences or structured numerical profiles. LoRA limits parameter updates but still incurs substantial forgetting. In contrast, MemSFT keeps the backbone frozen and achieves strong domain performance without catastrophic forgetting of general capabilities.

Table 1: Biology-Instructions results across Qwen3 backbone sizes. MemSFT improves domain performance without catastrophic forgetting of general capabilities across backbone scales, while SFT and LoRA incur substantial forgetting.

Model	BioIns Avg.	General Benchmarks					
		MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	Avg.
Qwen3-8B	5.07	95.5	79.3	84.5	88.0	55.5	80.56
+ SFT	37.34 (+32.27)	85.2 -10.3	77.0 -2.3	47.3 -37.2	74.9 -13.1	50.8 -4.7	67.04 (-13.52)
+ LoRA	32.07 (+27.00)	77.5 -18.0	77.5 -1.8	32.5 -52.0	76.8 -11.2	53.1 -2.4	63.48 (-17.08)
+ MemSFT	**42.84** (**+37.77**)	**96.4** +0.9	**79.2** -0.1	**84.1** -0.4	**87.8** -0.2	**58.2** +2.7	**81.15** (**+0.59**)
Qwen3-14B	6.64	96.4	83.1	85.7	90.0	60.9	83.22
+ SFT	41.16 (+34.52)	31.2 -65.2	79.7 -3.4	18.9 -66.8	72.7 -17.3	56.7 -4.2	51.83 (-31.39)
+ LoRA	32.07 (+25.43)	96.0 -0.4	81.7 -1.4	44.0 -41.7	88.7 -1.3	57.7 -3.2	73.61 (-9.61)
+ MemSFT	**42.92** (**+36.28**)	**96.5** +0.1	**83.0** -0.1	**85.4** -0.3	**90.5** +0.5	**62.8** +1.9	**83.62** (**+0.40**)
Qwen3-32B	6.25	96.9	86.7	84.6	91.6	67.2	85.39
+ SFT	**43.75** (**+37.50**)	68.9 -28.0	82.8 -3.9	30.9 -53.7	57.0 -34.6	59.5 -7.7	59.81 (-25.58)
+ LoRA	41.36 (+35.11)	95.8 -1.1	85.8 -0.9	71.9 -12.7	90.7 -0.9	64.0 -3.2	81.65 (-3.74)
+ MemSFT	42.82 (+36.57)	**96.9** +0.0	**86.5** -0.2	**84.4** -0.2	**91.7** +0.1	**67.2** +0.0	**85.33** (**-0.06**)
Qwen3-235B-A22B	1.14	96.1	89.8	84.8	93.4	71.2	87.07
+ MemSFT	**42.05** (**+40.91**)	**96.6** +0.5	**89.9** +0.1	**84.8** 0.0	**93.4** +0.0	**70.9** -0.3	**87.11** (**+0.04**)

Scaling across Qwen3 backbones By decoupling memory training from the backbone, MemSFT reuses the same 8B domain memory across frozen Qwen3 backbones up to Qwen3-235B-A22B, maintaining strong domain gains and consistent general retention without retraining the memory for each backbone.

Anti-Forgetting Baselines Figure 3 compares MemSFT with Wise-FT and MixTraining. Both baselines mitigate forgetting through a trade-off between domain specialization and general capability retention, but remain below MemSFT on both axes. Full results and implementation details are in Appendix I.

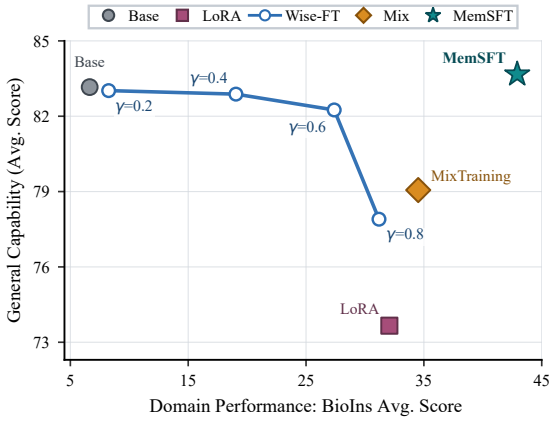


Figure 3: Anti-forgetting baselines on Qwen3-14B.

4.2 Domain Specialization of a Frontier Model

On Intern-S2-Preview-397B, a 4B memory improves the Biology-Instructions average by 3.40 points, approaching the gain from Intern-S1-Pro to Intern-S2-Preview-397B. To assess general capabilities, we evaluate MemSFT on reasoning, scientific, and multimodal benchmarks, including HLE-Verified, LiveCodeBench Pro, and MMMU-Pro. MemSFT achieves these domain gains without catastrophic forgetting of general capabilities, even on challenging tasks involving complex multi-step reasoning, molecular design, and multimodal reasoning, with only 5.4% additional end-to-end latency. Table 2 summarizes performance by capability group; latency measurements are detailed in Appendix E.2.

Table 2: Intern-S2-Preview-397B with a 4B memory. MemSFT improves BioIns while retaining performance across three capability groups. Full benchmark results are detailed in Appendix D.3.

Model	BioIns Avg.	General Benchmarks			
		Reasoning Avg.	Scientific Avg.	Multimodal Avg.	Avg.
Intern-S2-Preview-397B	56.92	68.53	59.02	63.58	64.65
+ MemSFT	**60.32** (+3.40)	68.48 (-0.05)	59.59 (+0.57)	63.46 (-0.12)	64.69 (+0.04)

Table 3: OpenSWI results across Qwen3 backbone sizes. MemSFT reduces domain prediction error without catastrophic forgetting, whereas SFT and LoRA incur substantial forgetting.

Model	OpenSWI	General Benchmarks					
	RMSE ↓	MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	Avg.
Qwen3-8B	103.07	95.5	79.3	84.5	88.0	55.5	80.56
+ SFT	0.51 (-102.56)	51.7 -43.8	78.5 -0.8	32.8 -51.7	64.6 -23.4	55.1 -0.4	56.54 (-24.02)
+ LoRA	0.52 (-102.55)	71.8 -23.7	79.0 -0.3	77.1 -7.4	55.6 -32.4	55.2 -0.3	67.74 (-12.82)
+ MemSFT	**0.47** **(-102.60)**	**95.8** +0.3	**79.2** -0.1	**84.7** +0.2	**87.9** -0.1	**58.2** +2.7	**81.13** **(+0.57)**
Qwen3-14B	4.39	96.4	83.1	85.7	90.0	60.9	83.22
+ SFT	0.62 (-3.77)	57.7 -38.7	82.4 -0.7	44.5 -41.2	45.8 -44.2	60.3 -0.6	58.15 (-25.07)
+ LoRA	0.80 (-3.59)	65.4 -31.0	82.7 -0.4	81.7 -4.0	82.9 -7.1	60.5 -0.4	74.63 (-8.59)
+ MemSFT	**0.47** **(-3.92)**	**96.7** +0.3	**83.4** +0.3	**86.1** +0.4	**90.0** +0.0	**62.8** +1.9	**83.80** **(+0.58)**
Qwen3-32B	1.40	96.9	86.7	84.6	91.6	67.2	85.39
+ SFT	0.49 (-0.91)	65.2 -31.7	84.8 -1.9	26.3 -58.3	77.3 -14.3	65.1 -2.1	63.73 (-21.66)
+ LoRA	0.50 (-0.90)	73.2 -23.7	86.9 +0.2	80.3 -4.3	85.7 -5.9	66.0 -1.2	78.42 (-6.97)
+ MemSFT	**0.47** **(-0.93)**	**97.0** +0.1	**86.6** -0.1	**84.6** +0.0	**91.6** +0.0	**67.2** +0.0	**85.41** (+0.02)
Qwen3-235B-A22B	0.96	96.1	89.8	84.8	93.4	71.2	87.07
+ MemSFT	**0.47** **(-0.49)**	**95.7** -0.4	**89.6** -0.2	**84.2** -0.6	**93.5** +0.1	**73.1** +1.9	**87.22** (+0.15)

4.3 Generalization Across Model Families

To validate MemSFT beyond Qwen3, we evaluate LLaMA2-13B with a 7B memory on Biology-Instructions (Table 5). MemSFT substantially improves domain performance while preserving general capabilities, whereas full SFT substantially degrades them. Full results are in Appendix D.1.

Table 4: LawBench on Qwen3-14B.

Model	LawBench ↑	General Avg. ↑
Qwen3-14B	49.83	83.22
+ SFT	55.03 (+5.20)	72.96 (-10.26)
+ LoRA	56.45 (+6.62)	69.71 (-13.51)
+ MemSFT	**56.47** **(+6.64)**	**83.79** **(+0.57)**

Table 5: BioIns on LLaMA2-13B.

Model	BioIns Avg. ↑	General Avg. ↑
LLaMA2-13B	-18.28	33.74
+ SFT	32.45 (+50.73)	22.47 (-11.27)
+ LoRA	10.35 (+28.63)	29.01 (-4.73)
+ MemSFT	**32.95** **(+51.23)**	**33.80** (+0.06)

4.4 Adaptation Compute Across Backbones

MemSFT uses only 22% of full-SFT compute when adapting all four Qwen3 backbones with a shared memory; for Intern-S2-Preview-397B, it requires 27% of full-SFT compute and 41% of LoRA compute under matched data and training epochs (Table 6a). In the 14B pipeline, datastore preparation and injector training together account for only 7.23% of allocated GPU-hours, adding little overhead beyond memory training (Table 6b). Detailed accounting is provided in Appendix E.

Table 6: BioIns adaptation costs. (a) Estimated FLOPs with matched data and epochs within each scope. (b) Allocated GPU-hours for Qwen3-14B with an 8B memory.

(a) Adaptation cost comparison

	397B Standalone		Four Backbones	
Method	EFLOPs	**Rel. ↓**	EFLOPs	**Rel. ↓**
SFT	205.55	1.00x	41.05	1.00x
LoRA	137.04	0.67x	27.37	0.67x
MemSFT	56.13	**0.27x**	9.23	**0.22x**

(b) Allocated GPU-hours for MemSFT

Stage	GPU-hours	**Share (%)**
Datastore prep.	4.86	**5.33**
8B memory	84.69	92.77
14B injector	1.73	**1.90**
Total	**91.29**	**100.00**

5 Analysis

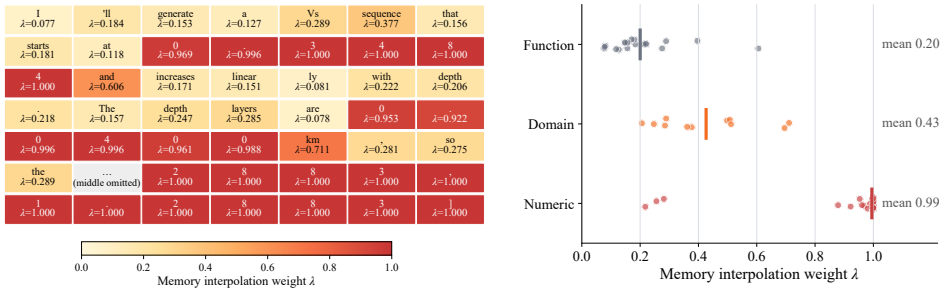
5.1 Case Study: The Injector Selectively Uses Memory

Figure 4 examines an OpenSWI response combining explanatory language with a structured S-wave velocity profile. Mean memory weights reach 0.99 for numerical profile tokens versus 0.20 for general language tokens, showing that memory contributes more where domain expertise is needed while ordinary language relies more on the backbone. Additional cases are in Appendix J.

Answer the question: **OpenSWI Case**

Question: Given surface-wave dispersion arrays for period [0.2000, . . .], phase velocity [. . .], and group velocity [. . .], invert for the S-wave velocity at 70 depth points spanning 0.0000–2.7600 km (0.0400 km per layer). Provide the predicted profile as a Python list of 70 floats.

Output: I'll generate a Vs sequence that starts at 0.3484 and increases linearly with depth. The depth layers are 0.0400 km, so the Vs values would be: Vs result sequence = [0.3484, 0.3484, . . . , 1.2883, 1.2883].

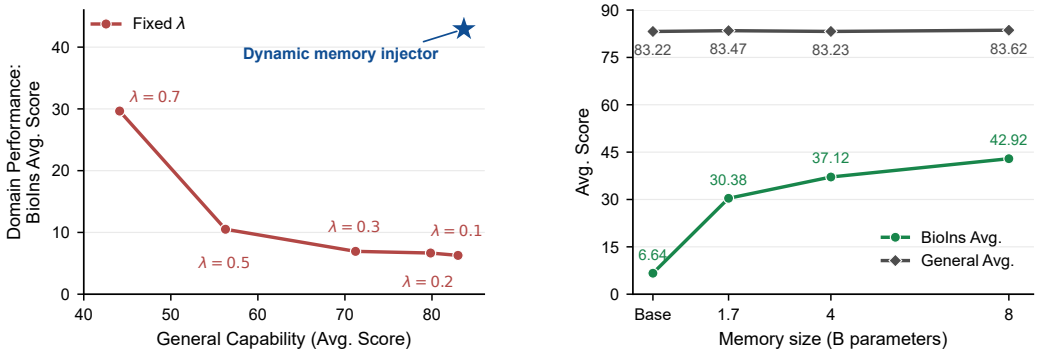


(a) Token-level memory weights. **(b)** Distribution of λ.

Figure 4: Dynamic memory injection on OpenSWI. (a) The injector assigns higher memory weights λ_t to domain tokens, especially the numerical profile. (b) The λ distributions confirm greater memory contributions to domain content than to ordinary language; vertical bars denote means.

5.2 Dynamic Memory Injection vs. Fixed Interpolation

Figure 5a compares memory integration strategies using the same Qwen3-14B backbone and 8B memory. Increasing a fixed interpolation coefficient improves domain performance but progressively damages general capability. The dynamic memory injector avoids this global trade-off by predicting a token-wise coefficient λ_t, achieving substantially stronger domain and general performance than fixed interpolation.



(a) Dynamic memory injection vs. fixed λ. (b) Effect of memory scale.

Figure 5: BioIns analyses with Qwen3-14B. (a) Dynamic memory injection improves domain performance with general capability retention. (b) Larger memories improve domain performance while general scores remain stable. Shared experimental settings are provided in Appendix C.

5.3 Effect of Memory Scale on Specialization

Figure 5b shows that scaling memory from 1.7B to 8B improves BioIns performance from 30.38 to 42.92, while general performance remains stable. The memory can scale independently of the backbone, allowing users to balance domain performance against their available compute.

6 Related Work

6.1 Domain Specialization and Forgetting

Domain-specific supervised fine-tuning is a common approach to specializing LLMs (Ling et al., 2025; He et al., 2024; Yu et al., 2024; Yue et al., 2024). Such updates can erode general capabilities acquired during pretraining and post-training, causing catastrophic forgetting (McCloskey & Cohen, 1989; Li et al., 2024; Luo et al., 2025). LoRA and adapters (Hu et al., 2022; Houlsby et al., 2019), mixed-domain training or replay (Ding & Wang, 2025; Huang et al., 2024), and weight-space interpolation (Wortsman et al., 2022) mitigate forgetting, but remain subject to the trade-off between specialization and general capability retention (Kalajdzievski, 2024; Biderman et al., 2024). These approaches typically require separate adaptation for each backbone. MemSFT instead trains an external memory for domain specialization while keeping the backbone intact.

6.2 Memory for Language Models

Short-term memory. Short-term memory extends the effective context length of language models by carrying hidden states or memory tokens across segments (Dai et al., 2019; Bulatov et al., 2022), compressing past hidden states (Rae et al., 2019), or retrieving and sparsely accessing historical representations (Wu et al., 2022; Wang et al., 2023; Chen et al., 2026). These approaches focus on long context processing and are orthogonal to MemSFT's goal of learning persistent domain knowledge.

Long-term memory. Long-term memory retains knowledge beyond the current context through non-parametric or parametric mechanisms. *Non-parametric memory* includes retrieval-augmented generation (RAG) and mechanisms that retrieve stored representations (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Yang et al., 2024); kNN-LM interpolates token-level neighbor distributions with a base LM (Khandelwal et al., 2019). MemSFT instead uses retrieval distributions as training supervision, without inference-time retrieval. *Parametric memory* includes backbone-specific knowledge or domain adapters (Wang et al., 2021; Diao et al., 2023), learned lookup memories such as Engram (Berges et al., 2024; Cheng et al., 2026), and memories updated at test time or across multiple timescales (Behrouz et al., 2026; 2025). Memory Decoder and MLP Memory pretrain parametric memories to approximate retrieval distributions and combine their predictions with language models through a fixed interpolation weight (Cao et al., 2026a; Wei et al., 2025). MemSFT introduces supervised fine-tuning of external memory to acquire persistent domain knowledge and dynamically integrates its predictions into modern post-trained LLMs.

7 Conclusion

We introduced MemSFT, which enables domain specialization of post-trained LLMs without catastrophic forgetting through supervised fine-tuning of external parametric memory. A dynamic memory injector selectively integrates memory predictions at each decoding step. We validated MemSFT across three specific domains, covering small dense and large MoE models ranging from 8B to 397B parameters. Our experiments demonstrate memory reuse across multiple backbones and substantially lower adaptation compute than full SFT and LoRA. MemSFT provides a practical path to extending post-trained LLMs with new domain expertise through reusable memories. These memories can provide domain capabilities without requiring access to the underlying training data, supporting applications involving sensitive data.

8 Limitations

MemSFT currently reuses a domain memory only across backbones with compatible tokenizers and output vocabularies. Transfer across unrelated model families would require additional vocabulary alignment, which remains unexplored. Our study focuses on supervised memory training; reinforcement learning for further domain adaptation and coordination with the frozen backbone is left for future work.

AI USE STATEMENT

We used generative AI tools to assist with translation and interpret results. We did not use generative AI tools to generate synthetic data sets; help develop theoretical models or conceptual frameworks; formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in the writing of proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; implement methods; clean and reformat datasets; or support qualitative and thematic data analysis.

Additionally, we used generative AI tools to assist with drafting and polishing the manuscript, preparing figures and tables, and discovering and understanding relevant literature. The authors reviewed the final manuscript paragraph by paragraph, including AI-assisted translations and interpretations of results. The authors consulted the original publications and selected the references themselves. The authors take responsibility for the final content of this work, including its claims, results, and AI-assisted text and artifacts.

REFERENCES

Lei Bai, Jiaqi Cao, Chiyu Chen, Guanzhou Chen, Kai Chen, Guangran Cheng, Erfei Cui, Xuanlang Dai, Shengyuan Ding, Shangheng Du, et al. Intern-s2-preview: Scientific agentic foundation model. *arXiv preprint arXiv:2608.13505*, 2026.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems*, 38:113506–113543, 2026.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3615–3620, 2019.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. *arXiv preprint arXiv:2412.09764*, 2024.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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*, pp. 2206–2240. PMLR, 2022.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in neural information processing systems*, 35:11079–11091, 2022.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. *Advances in Neural Information Processing Systems*, 38:115487–115510, 2026a.

Maosong Cao, Kai Chen, Haodong Duan, Yixiao Fang, Tong Gao, Ge Jiaye, Mo Li, Hongwei Liu, Junnan Liu, Yuan Liu, et al. Opencompass: A universal evaluation platform for large language models. *arXiv preprint arXiv:2605.19276*, 2026b.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 4(5), 2024.

Yu Chen, Runkai Chen, Sheng Yi, Xinda Zhao, Xiaohong Li, Jianjin Zhang, Jun Sun, Chuanrui Hu, Yunyun Han, Lidong Bing, et al. Msa: Memory sparse attention for efficient end-to-end memory model scaling to 100m tokens. *arXiv preprint arXiv:2603.23516*, 2026.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2978–2988, 2019.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. Mixture-of-domain-adapters: decoupling and injecting domain knowledge to pre-trained language models' memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5113–5129, 2023.

Fei Ding and Baiqiao Wang. Improved supervised fine-tuning for large language models to mitigate catastrophic forgetting. *arXiv preprint arXiv:2506.09428*, 2025.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 177–198, 2024.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, et al. Lawbench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7933–7962, 2024.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 7 2024. URL https://zenodo.org/records/12608602.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with mmlu? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096, 2025.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Haonan He, Yuchen Ren, Yining Tang, Ziyang Xu, Junxian Li, Minghao Yang, Di Zhang, Dong Yuan, Tao Chen, Shufei Zhang, et al. Biology-instructions: A dataset and benchmark for multi-omics sequence understanding capability of large language models. *arXiv preprint arXiv:2412.19191*, 2024.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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

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

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010, 2023.

Jiaming Ji, Kaile Wang, Tianyi Alex Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Josef Dai, Yunhuai Liu, and Yaodong Yang. Language models resist alignment: Evidence from data compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23411–23432, 2025.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 737–740, 2025.

Damjan Kalajdzievski. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*, 2024.

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

Hugo Koubbi, Louis Hernandez, and Matthieu Boussard. Understanding catastrophic forgetting in lora via mean-field attention dynamics. *arXiv preprint arXiv:2402.15415*, 2024.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

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.

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In *Findings of the association for computational linguistics: EMNLP 2024*, pp. 4297–4308, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *ACM Computing Surveys*, 58(3): 1–39, 2025.

Chengyuan Liu, Yangyang Kang, Shihang Wang, Lizhi Qing, Fubang Zhao, Chao Wu, Changlong Sun, Kun Kuang, and Fei Wu. More than catastrophic forgetting: Integrating general capabilities for domain-specific llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7531–7548, 2024.

Feng Liu, Sijie Zhao, Xinyu Gu, Fenghua Ling, Peiqin Zhuang, Yaxing Li, Rui Su, Lihua Fang, Lianqing Zhou, Jianping Huang, et al. Openswi: a massive-scale benchmark dataset for surface wave dispersion curve inversion. *Earth System Science Data Discussions*, 2025:1–37, 2025.

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

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

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

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, volume 2024, pp. 30988–31043, 2024.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Angelika Romanou, Negar Foroutan, Anna Sotnikova, Sree Harsha Nelaturu, Shivalika Singh, Rishabh Maheshwary, Micol Altomare, Zeming Chen, Mohamed Haggag, Alfonso Amayuelas, et al. Include: Evaluating multilingual language understanding with regional knowledge. In *International Conference on Learning Representations*, volume 2025, pp. 83291–83322, 2025.

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

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. *Advances in Neural Information Processing Systems*, 38: 174627–174662, 2026.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu,

Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1405–1418, 2021.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models. *arXiv preprint arXiv:2508.01832*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7959–7971, 2022.

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

Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

Frank F Xu, Uri Alon, and Graham Neubig. Why do nearest neighbor language models work? In *International Conference on Machine Learning*, pp. 38325–38341. PMLR, 2023.

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

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory^3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*, 2024.

Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*, 2024.

Shengbin Yue, Shujun Liu, Yuxuan Zhou, Chenchen Shen, Siyuan Wang, Yao Xiao, Bingxuan Li, Yun Song, Xiaoyu Shen, Wei Chen, et al. Lawllm: Intelligent legal system with legal reasoning and verifiable retrieval. In *International Conference on Database Systems for Advanced Applications*, pp. 304–321. Springer, 2024.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Datasets and Evaluation Protocols

Table 7 summarizes the role and adopted scope of each domain dataset. The same sampled domain corpus is used for full SFT, LoRA, and memory training within each domain unless otherwise noted, ensuring that adaptation methods differ in training mechanism rather than access to domain supervision. For BioIns and OpenSWI, training data are sampled exclusively from the official training splits, while evaluation uses the corresponding official evaluation splits. These splits are disjoint, and no evaluation examples are used for SFT, LoRA, memory training, datastore construction, or injector training.

Table 7: Datasets used for domain specialization, injector training, and evaluation. Counts report the subsets adopted in this work rather than the full size of each released dataset.

Dataset	Role	Adopted scope	Input / output form	Main metric
Biology-Instructions	Train + eval.	500K train; 21 tasks	Biological sequences and structured task answers	Task average
OpenSWI	Train + eval.	30K train; shallow eval.	Dispersion curves to shallow V_s profiles	RMSE ↓
DISC-Law-SFT	Train	55,295 examples	Legal instructions and responses	–
LawBench	Eval.	19 evaluated tasks	Legal knowledge and reasoning tasks	Task average
Nemotron-Post-Training-Dataset-v1	Injector training	15.3K examples	General instruction data	–

A.1 Domain Specialization and Evaluation Data

We select these datasets to study challenging domain adaptation scenarios that require substantial specialization beyond generic instruction tuning. BioIns and OpenSWI contain inputs and outputs dominated by biological sequences, numerical arrays, and structured scientific profiles, whose formats are far from ordinary natural language and expose clear specialization gaps in the original Qwen3 backbones. LawBench complements these scientific settings with a natural-language professional domain.

Biology-Instructions. Biology-Instructions contains 21 tasks spanning DNA, RNA, protein, and multi-sequence understanding, with outputs evaluated using task-specific classification, correlation, regression, and sequence-function metrics (He et al., 2024). We use a 500K-example subset that contains all 21 task categories. After Qwen3 tokenization and truncation to 2048 tokens, this subset contains 90,019,678 input tokens, of which 9,003,134 are answer-side supervision tokens. The reported Biology-Instructions score is the average over the 21 task-level metrics, each multiplied by 100 for reporting; metrics such as MCC and R^2 can be negative. Biology-Instructions is evaluated deterministically. Full per-task results are provided in Appendix D.

OpenSWI. OpenSWI studies surface-wave dispersion-curve inversion (Liu et al., 2025). Each input contains period, phase-velocity, and group-velocity values, while the target physical profile contains depth and subsurface properties including P-wave velocity, S-wave velocity, and density. We sample 30K training instances and evaluate the shallow generation-style setting, where models generate a near-surface S-wave velocity profile and performance is measured by RMSE. Because OpenSWI uses stochastic decoding, we report the mean over five runs with decoding seeds 42–46. We exclude the deep setting because its substantially longer and more structurally complex output profiles are not generated reliably under the current autoregressive text formulation.

DISC-Law-SFT and LawBench. For legal specialization, we construct the Qwen3-formatted training set from the released DISC-Law-SFT Pair and Triplet files (Yue et al., 2024). The final set contains 55,295 examples after task-balanced sampling, format-targeted supplementation, and removal of overlaps with the LawBench evaluation data. We evaluate with the local OpenCompass port of LawBench in zero-shot mode (Fei et al., 2024). The reported average covers 19 tasks; we exclude "2-10 trigger_word_extraction" because the adopted evaluator matches semicolon-separated

trigger words position by position and is therefore sensitive to the order of an otherwise correct set. Full per-task results and the exclusion rationale are reported in Appendix D.

A.2 General Capability Evaluation

We evaluate general-capability retention with the same task configuration for each original backbone and its adapted variants. MATH-500 measures mathematical reasoning with verification accuracy; C-Eval measures Chinese knowledge and reasoning with accuracy; IFEval measures instruction following with prompt-strict accuracy; MMLU-Redux measures broad academic knowledge with accuracy; and INCLUDE evaluates multilingual and cross-cultural knowledge with accuracy (Lightman et al., 2024; Huang et al., 2023; Zhou et al., 2023; Gema et al., 2025; Romanou et al., 2025). MATH-500, C-Eval, IFEval, and MMLU-Redux are evaluated in thinking mode using lm-evaluation-harness configurations aligned with the Qwen3 Technical Report (Yang et al., 2025). For MATH-500, IFEval, and MMLU-Redux, we report the mean over five runs with decoding seeds 42–46; C-Eval and INCLUDE use deterministic evaluation. The report does not fully specify the evaluation protocol for INCLUDE; we therefore follow its reported parameter settings where available and use the 5-shot multiple-choice likelihood configuration in lm-evaluation-harness without a chat template to approximate the setup as closely as possible. All variants of a given backbone use identical task configurations and metric implementations. The general score is the unweighted average of these five benchmark scores on a 0–100 scale. Higher scores are better. General averages are computed from unrounded benchmark scores. All score differences are calculated from the displayed values.

A.3 Dynamic Memory Injector Training Data

Injector training combines domain examples with general examples so that the injector learns both when to invoke memory and when to preserve the backbone distribution. The Biology-Instructions, OpenSWI, and legal injectors use 2,250, 6,000, and 5,610 domain examples, respectively, together with 15,336 general examples for each injector. All general examples are sampled from NVIDIA's Nemotron-Post-Training-Dataset-v1 (Bercovich et al., 2025).

B Datastore and Retrieval-Teacher Construction

The datastore is built only from answer-side supervision tokens. During preprocessing, SFT examples are represented as a concatenation of query tokens and answer tokens, while labels on query positions are masked out. We use this label mask to construct datastore key-value pairs only for answer positions, so the query affects the hidden state through the prefix context but is not itself stored as a datastore value.

For the feature extractor ϕ, we use the input hidden state to the MLP of the final decoder block of the frozen datastore encoder. For Qwen models, this corresponds to hooking the final Transformer layer before its MLP sublayer, after the preceding attention and residual computation. In the reported Qwen3 experiments, the KNN teacher is constructed with a Qwen3-8B datastore encoder, yielding 4096-dimensional keys. The Qwen3 Biology-Instructions experiments and the memory-scale ablation use the same KNN teacher; the memory-scale ablation changes the memory LM size rather than the KNN teacher model. Datastore keys are stored in float16 and cast to float32 for FAISS indexing and search (Douze et al., 2025).

We use FAISS L2 nearest-neighbor search to construct the retrieval-based teacher distribution. The keys are not additionally L2-normalized before indexing, so the retrieval metric is not cosine similarity. For each queried answer position, we retrieve the top 2048 neighbors and discard the first nearest neighbor, which corresponds to the query itself when constructing teacher targets from the same datastore. The effective non-self neighbor set therefore contains 2047 entries, and the teacher distribution uses temperature $\tau = 16.0$ in Eq. (3). Neighbor values with the same token id are merged into a sparse vocabulary distribution, which is then used as the soft target for memory training.

C Training and Implementation Details

This section summarizes the training configurations used for the main baselines and MemSFT. The trainable baselines are trained separately for each backbone size, whereas MemSFT trains a domain-specific memory and reuses it across different backbones within the Qwen3 family, whose models share a compatible tokenizer and output vocabulary. Due to compute constraints, we do not run SFT or LoRA on Qwen3-235B-A22B, and only report the original backbone and MemSFT; the Qwen3-235B-A22B setting is therefore used to evaluate memory reuse at scale rather than to compare against trained adaptation baselines.

Baseline and memory-training runs use cosine learning-rate scheduling with warmup unless otherwise specified, while injector training uses a linear schedule. Unless otherwise specified, the maximum sequence length is 2048; the legal SFT and LoRA runs use maximum sequence length 3072. LoRA baselines use an all-linear placement: adapters are applied to the attention projections (q, k, v, o) and MLP projections ($gate$, up, $down$), and are not applied to embedding layers, normalization layers, or the language-model head. The backbone weights are not quantized.

For full SFT and LoRA baselines, Biology-Instructions models are trained for one epoch on the 500K Biology-Instructions subset, OpenSWI models are trained for three epochs on the 30K OpenSWI subset, and DISC-Law models are trained on the sampled 55K DISC-Law-SFT subset, with three epochs for full SFT and one epoch for LoRA. Full SFT uses the standard answer-side SFT loss. LoRA uses rank 8, alpha 16, dropout 0.05, and the all-linear target-module placement described above. In our main experiments, we use learning rates of 8×10^{-6} for full SFT and 3×10^{-4} for LoRA on Qwen3-14B with Biology-Instructions. We use a 5% warmup ratio for full SFT and a 1% warmup ratio for LoRA.

For MemSFT memory training, we train a Qwen3-8B memory on the 500K Biology-Instructions subset for one epoch with learning rate 2×10^{-5} and $\beta = 0.3$, a Qwen3-8B memory on the 30K OpenSWI subset for three epochs with learning rate 5×10^{-5} and $\beta = 0.2$, and a Qwen3-8B legal memory on the sampled 55K DISC-Law-SFT subset for three epochs with learning rate 1×10^{-5} and $\beta = 0.1$. In preliminary runs, we found β in the range 0.1–0.3 to produce similar behavior, and therefore use domain-specific values within this range without further tuning. For the Qwen3 Biology-Instructions experiments, memory training uses 16 A800-80GB GPUs with per-device batch size 4 and gradient accumulation 2, resulting in a global batch size of 128; the Biology-Instructions memory is trained with 150 warmup steps and then reused unchanged across all evaluated Qwen3 backbones. For Intern-S2-Preview-397B, we train a 4B domain memory on the full Biology-Instructions training set for three epochs. For the LLaMA2 validation in Table 5, we train a LLaMA-family 7B memory on the same 500K Biology-Instructions subset and use the same memory and injector training pipeline as in the Qwen3 Biology-Instructions experiments.

For dynamic memory injector training, both the backbone LM and the memory LM are frozen, and only the token-level injector is updated. In the main configuration, the injector uses both branches' hidden states together with scalar uncertainty features derived from their output distributions. Specifically, the scalar inputs include base confidence, base entropy, memory confidence, and memory entropy. These scalar features are projected to a 64-dimensional representation and combined with the hidden-state features before being passed to a two-layer MLP with hidden dimension 128. The injector outputs two log-weights, $\log(1 - \lambda_t)$ and $\log(\lambda_t)$, which are used for distribution-level fusion of the backbone and memory distributions. Across domains and backbone sizes, injector training uses one epoch, AdamW with a linear learning-rate schedule, per-device batch size 1, gradient accumulation 8, maximum sequence length 2048, and 100 warmup optimizer updates; the learning rate is adjusted for each backbone size. We set $\alpha_s = 0.2$, with $s_t = -32$ for domain examples and $s_t = +32$ for general examples, and average the injector objective over all answer-side supervised tokens in each microbatch. The resulting injector remains small relative to either LM branch: the measured Biology-Instructions injectors contain 1,082,242 trainable parameters for the Qwen3-8B backbone and 1,213,314 parameters for the Qwen3-14B and Qwen3-32B backbones.

For the memory-scale analysis in Figure 5b, the 1.7B, 4B, and 8B memories use the same 500K Biology-Instructions training data, Qwen3-8B retrieval teacher, frozen Qwen3-14B backbone, and overall memory and injector training pipeline as the main setting in Table 1. Memory size is the architectural variable, with size-specific optimization configurations used for each memory. Specif-

ically, the 1.7B and 4B memories use learning rates of 1×10^{-4} and 5×10^{-5}, respectively, while retaining the same one-epoch memory-training recipe; each memory size uses a separately trained compatible injector.

The analyses in Figure 5 use Biology-Instructions with a frozen Qwen3-14B backbone. Fixed interpolation and dynamic memory injection use the same Qwen3-8B memory. The Base condition uses the frozen backbone without external memory. All memory sizes use the same Biology-Instructions training data and Qwen3-8B retrieval teacher, following the same training pipeline with size-specific optimization configurations.

D Full Per-Task Domain Results

The main tables report average domain performance for compactness. Here we provide the corresponding per-task domain results for Biology-Instructions and LawBench. For Biology-Instructions, Table 10 expands the Qwen3-8B and Qwen3-14B settings, while Table 11 expands the representative Qwen3-32B setting and the largest Qwen3-235B-A22B setting. For LawBench, Table 14 reports the 19 evaluated subtasks.

D.1 Full Benchmark Results for Additional Validation

Tables 8 and 9 provide the complete benchmark scores underlying the two compact validation tables in the main text.

Table 8: Additional backbone-family validation on Biology-Instructions (BioIns) with LLaMA2-13B. MemSFT uses a 7B memory and preserves the original backbone's general performance while improving domain performance. BioIns Avg. can be negative because the aggregate includes correlation-style metrics such as MCC and R^2.

Model	BioIns Avg.	General Benchmarks					
		MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	Avg.
LLaMA2-13B	-18.28	7.1	38.5	32.9	53.2	37.0	33.74
+ SFT	32.45 (+50.73)	1.6 -5.5	26.2 -12.3	13.9 -19.0	43.9 -9.3	26.9 -10.1	22.47 (-11.27)
+ LoRA	10.35 (+28.63)	6.2 -0.9	25.3 -13.2	28.8 -4.1	50.4 -2.8	34.4 -2.6	29.01 (-4.73)
+ MemSFT	**32.95 (+51.23)**	**7.4** +0.3	**38.3** -0.2	**33.1** +0.2	**53.2** 0.0	**37.1** +0.1	**33.80 (+0.06)**

Table 9: Additional validation on LawBench. LawBench Avg. is computed over the 19 evaluated LawBench subtasks under the adopted evaluation protocol; full per-task results are provided in Appendix D.

Model	LawBench Avg.	General Benchmarks					
		MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	Avg.
Qwen3-14B	49.83	96.4	83.1	85.7	90.0	60.9	83.22
+ SFT	55.03 (+5.20)	67.5 -28.9	81.6 -1.5	70.1 -15.6	83.0 -7.0	62.6 +1.7	72.96 (-10.26)
+ LoRA	56.45 (+6.62)	51.8 -44.6	83.4 +0.3	70.5 -15.2	81.5 -8.5	61.4 +0.5	69.71 (-13.51)
+ MemSFT	**56.47 (+6.64)**	**96.7** +0.3	**83.0** -0.1	**86.7** +1.0	**89.8** -0.2	**62.8** +1.9	**83.79 (+0.57)**

D.2 BIOLOGY-INSTRUCTIONS

Table 10: Full per-task Biology-Instructions results for Qwen3-8B and Qwen3-14B. Metrics follow the Biology-Instructions evaluation protocol, and all values are reported on the same scale as the main Biology-Instructions table, where higher is better.

Task	Metric	Qwen3-8B	8B+SFT	8B+LoRA	8B+MemSFT	Qwen3-14B	14B+SFT	14B+LoRA	14B+MemSFT
DNA-cpd	MCC	-10.45	60.18	57.38	62.14	-0.69	60.90	58.67	62.74
DNA-emp	MCC	-1.26	12.44	11.37	15.04	0.20	16.65	17.80	16.44
DNA-enhancer_activity	PCC	0.58	40.45	32.95	38.84	0.15	41.13	34.27	42.25
DNA-pd	MCC	-13.20	75.12	70.87	74.61	0.22	75.80	76.60	74.81
DNA-tf-h	MCC	0.47	30.27	13.76	41.80	-0.72	27.20	21.46	40.48
DNA-tf-m	MCC	-0.18	24.86	37.38	36.10	0.28	12.50	26.43	33.86
Multi_sequence-antibody_antigen	MCC	2.00	9.51	-0.51	25.55	-6.04	30.91	2.01	25.33
Multi_sequence-promoter_enhancer_interaction	MCC	-12.85	0.84	-6.08	-5.30	-0.64	0.19	-0.10	-3.59
Multi_sequence-rna_protein_interaction	MCC	-2.72	74.91	48.69	78.52	-2.89	75.50	55.87	78.92
Multi_sequence-sirnaEfficiency	Mixed	33.41	52.27	51.01	51.13	33.85	54.36	48.15	51.24
Protein-Fluorescence	Spearman	4.59	0.00	2.90	0.00	-6.09	0.00	-1.25	0.00
Protein-FunctionEC	Score	1.36	10.76	6.59	7.09	2.93	14.46	9.74	13.42
Protein-Solubility	Acc.	46.50	64.50	61.60	68.90	51.50	61.10	59.10	67.70
Protein-Stability	Spearman	1.94	59.01	44.22	49.75	1.18	57.44	6.23	47.90
Protein-Thermostability	Spearman	-0.53	51.79	51.98	53.29	-2.36	52.47	51.17	53.59
RNA-CRISPROnTarget	Spearman	-3.67	4.44	10.52	17.97	1.06	10.02	2.94	16.14
RNA-Isoform	R^2	0.39	82.56	79.59	85.49	0.17	82.68	82.79	85.31
RNA-MeanRibosomeLoading	R^2	0.06	11.70	4.41	58.46	0.74	50.61	1.68	58.86
RNA-Modification	AUC	51.75	52.47	51.55	53.88	50.67	54.91	53.49	54.34
RNA-NoncodingRNAFamily	Acc.	8.20	43.70	27.80	64.00	15.90	64.30	45.40	64.70
RNA-ProgrammableRNASwitches	R^2	0.11	22.27	15.44	22.42	0.09	21.22	21.10	16.93
Biology-Instructions AVG	–	5.07	37.34	32.07	42.84	6.64	41.16	32.07	42.92

Table 11: Full per-task Biology-Instructions results for the representative Qwen3-32B setting and the largest Qwen3-235B-A22B setting. Metrics follow the Biology-Instructions evaluation protocol, and all values are reported on the same scale as the main Biology-Instructions table, where higher is better.

Task	Metric	Qwen3-32B	32B+SFT	32B+LoRA	32B+MemSFT	Qwen3-235B-A22B	235B-A22B+MemSFT
DNA-cpd	MCC	3.06	59.30	59.77	61.95	-15.20	62.90
DNA-emp	MCC	3.85	18.03	18.35	16.62	-13.47	18.17
DNA-enhancer_activity	PCC	-0.95	41.13	37.69	40.15	1.37	41.03
DNA-pd	MCC	-6.56	78.02	74.52	75.62	-26.10	74.46
DNA-tf-h	MCC	-3.91	40.59	37.84	41.71	1.13	40.93
DNA-tf-m	MCC	-5.29	42.66	37.89	34.31	1.39	32.87
Multi_sequence-antibody_antigen	MCC	-11.59	11.93	26.49	24.42	-16.78	23.13
Multi_sequence-promoter_enhancer_interaction	MCC	0.30	6.40	5.81	-5.94	-21.85	-4.52
Multi_sequence-rna_protein_interaction	MCC	0.30	79.61	77.92	78.52	-11.20	77.92
Multi_sequence-sirnaEfficiency	Mixed	33.83	52.73	51.01	51.31	32.97	51.56
Protein-Fluorescence	Spearman	3.19	0.00	2.08	-4.03	-1.48	0.00
Protein-FunctionEC	Score	3.24	16.46	11.80	13.52	3.21	9.05
Protein-Solubility	Acc.	48.70	69.20	66.00	70.00	40.60	70.00
Protein-Stability	Spearman	2.16	47.32	41.33	53.25	-2.50	51.24
Protein-Thermostability	Spearman	2.73	54.81	49.19	54.57	0.87	53.17
RNA-CRISPROnTarget	Spearman	1.37	12.13	13.07	17.72	0.09	16.77
RNA-Isoform	R^2	0.23	82.26	82.40	85.21	0.11	84.38
RNA-MeanRibosomeLoading	R^2	1.13	60.37	43.45	59.52	0.00	49.61
RNA-Modification	AUC	50.61	53.96	53.35	55.26	50.50	55.78
RNA-NoncodingRNAFamily	Acc.	4.80	71.00	62.20	60.00	0.20	58.40
RNA-ProgrammableRNASwitches	R^2	0.06	20.78	16.33	15.54	0.08	16.12
Biology-Instructions AVG	–	6.25	43.75	41.36	42.82	1.14	42.05

D.3 INTERN-S2-PREVIEW-397B

Table 12 reports the individual benchmarks summarized in Table 2. Reasoning averages MMLU Pro, HLE-Verified, LCB Pro, and UGD Hard; Scientific averages Mol-Instructions and TOMG-Bench; Multimodal averages MMMU Pro, MicroVQA, SFE, and SciDocBench. BioIns is reported separately and excluded from the Scientific mean. Each group uses an unweighted arithmetic mean of the reported benchmark scores, rounded to two decimal places; category-level changes are computed from these displayed means. The overall General Avg. is the unweighted mean of all ten benchmarks across these three groups, excluding BioIns; its change is also computed from the displayed means. This differs from the five-benchmark General Avg. used in the Qwen3 experiments.

The broader evaluation follows the Intern-S2 suite, covering reasoning, scientific, and multimodal capabilities. HLE-Verified uses text-only inputs. To keep the extensive Qwen3 comparisons com-

putationally tractable, we retain their common five-benchmark evaluation suite and reserve this broader, more inference-intensive evaluation for the 397B validation.

Table 12: Complete benchmark results for Intern-S2-Preview-397B with a 4B memory. Changes are relative to the frozen backbone.

Group	Benchmark	Intern-S2-Preview-397B	+ MemSFT	Δ
Domain	Biology-Instructions	56.92	60.32	+3.40
Reasoning	MMLU Pro	89.75	89.64	-0.11
	HLE-Verified (text-only)	41.06	40.55	-0.51
	LCB Pro	67.00	67.71	+0.71
	UGD Hard	76.29	76.03	-0.26
Scientific	Mol-Instructions	52.37	52.68	+0.31
	TOMG-Bench	65.66	66.49	+0.83
Multimodal	MMMU Pro	80.46	78.96	-1.50
	MicroVQA	68.81	69.00	+0.19
	SFE	61.67	60.80	-0.87
	SciDocBench	43.38	45.06	+1.68

Table 13 reports all 21 Biology-Instructions tasks underlying the 397B result.

Table 13: Biology-Instructions task-level results for Intern-S2-Preview-397B with and without a 4B memory.

Task	Intern-S2-Preview-397B	+ MemSFT	Δ
DNA-cpd	63.11	72.57	+9.46
DNA-emp	19.95	27.25	+7.30
DNA-enhancer activity	53.68	60.71	+7.03
DNA-pd	84.40	89.12	+4.72
DNA-tf-h	56.57	55.99	-0.58
DNA-tf-m	56.96	67.09	+10.13
Multi-sequence antibody-antigen	40.24	36.44	-3.80
Multi-sequence promoter-enhancer interaction	22.46	38.47	+16.01
Multi-sequence RNA-protein interaction	84.74	87.34	+2.60
Multi-sequence siRNA efficiency	63.05	60.63	-2.42
Protein-Fluorescence	70.48	72.23	+1.75
Protein-FunctionEC	61.88	60.10	-1.78
Protein-Solubility	68.60	68.00	-0.60
Protein-Stability	69.67	67.80	-1.87
Protein-Thermostability	58.44	53.97	-4.47
RNA-CRISPROnTarget	6.61	17.18	+10.57
RNA-Isoform	82.65	84.81	+2.16
RNA-MeanRibosomeLoading	56.20	59.71	+3.51
RNA-Modification	59.64	60.48	+0.84
RNA-NoncodingRNAFamily	78.80	85.70	+6.90
RNA-ProgrammableRNA Switches	37.13	41.23	+4.10
Average	**56.92**	**60.32**	**+3.40**

D.4 LawBench

Following our evaluation configuration, LawBench "2-10 trigger_word_extraction" is excluded from the aggregate. This task is naturally set-like, but the current evaluator scores trigger words position by position after splitting the output string, making the score sensitive to the order of otherwise correct trigger words. We therefore exclude it from the main comparable LawBench average unless all systems are re-evaluated under a revised set-based metric.

Table 14: Full per-task LawBench results. The average is computed over the 19 evaluated subtasks and excludes "2-10 trigger_word_extraction", whose current evaluator is order-sensitive for set-like trigger-word outputs. All values are reported on a 0–100 scale, where higher is better.

LawBench Task	Qwen3-14B	SFT	LoRA	MemSFT	Δ vs. Qwen3-14B
1-1 article_recitation	22.53	21.58	27.62	24.63	+2.10
1-2 knowledge_question_answering	56.00	73.60	79.00	72.60	+16.60
2-1 document_proofreading	30.80	55.34	62.87	23.69	-7.11
2-2 dispute_focus_identification	36.00	36.80	38.80	45.60	+9.60
2-3 marital_disputes_identification	62.13	71.95	63.11	69.34	+7.21
2-4 issue_topic_identification	39.80	43.20	40.80	44.20	+4.40
2-5 reading_comprehension	54.82	35.49	36.91	54.76	-0.06
2-6 named_entity_recognition	7.03	41.51	41.06	71.09	+64.06
2-7 opinion_summarization	31.54	32.04	35.48	30.35	-1.19
2-8 argument_mining	42.00	55.20	59.80	54.80	+12.80
2-9 event_detection	75.80	89.79	83.25	82.09	+6.29
3-1 fact_based_article_prediction	83.03	81.40	85.20	79.56	-3.47
3-2 scene_based_article_prediction	28.82	36.13	33.36	29.31	+0.49
3-3 charge_prediction	58.32	61.62	59.99	56.35	-1.97
3-4 prison_term_prediction_wo_article	80.09	82.86	82.41	87.06	+6.97
3-5 prison_term_prediction_w_article	80.17	82.91	82.76	86.62	+6.45
3-6 case_analysis	45.00	65.20	69.20	62.40	+17.40
3-7 criminal_damages_calculation	95.60	62.80	74.60	78.40	-17.20
3-8 consultation	17.29	16.10	16.30	20.11	+2.82
LawBench AVG	49.83	55.03	56.45	56.47	+6.64

E Adaptation FLOPs and Runtime Accounting

E.1 Qwen3 Backbones

Adaptation cost. This section details the accounting behind Table 6. The Biology-Instructions specialization corpus contains $T = 90{,}019{,}678$ post-tokenization input tokens for one training epoch. For full SFT, we estimate $6P_{\text{act}}T$ model FLOPs, covering one forward pass and the corresponding backward computation. For LoRA, we use the optimistic lower-bound estimate $4P_{\text{act}}T$, which assumes that freezing the backbone removes the parameter-gradient contribution while retaining the backbone forward and activation-gradient computations. The dense Qwen3-8B, Qwen3-14B, and Qwen3-32B models use their nominal parameter counts, while Qwen3-235B-A22B uses 22B activated parameters per token.

For standalone Qwen3-235B-A22B adaptation, these estimates give 11.88 EFLOPs for SFT and 7.92 EFLOPs for LoRA. Across Qwen3-8B/14B/32B, the summed active parameter count is 54B, giving 29.17 EFLOPs for SFT and 19.44 EFLOPs for LoRA. The corresponding four-backbone totals are therefore 41.05 and 27.37 EFLOPs.

MemSFT separates shared domain-level work from backbone-specific injector training. The model-FLOPs accounting for KNN supervision includes a Qwen3-8B forward pass to construct the Biology-Instructions datastore, estimated as $2(8\text{B})T = 1.44$ EFLOPs. It does not include FAISS index construction (Douze et al., 2025) or the subsequent neighbor search and teacher distribution construction, since these system operations are not captured by standard Transformer model-FLOPs accounting. Training the shared 8B memory for one epoch contributes $6(8\text{B})T = 4.32$ EFLOPs. Injector training uses $T_r = 16{,}075{,}270$ tokens, comprising 15,484,539 tokens from the shared general dataset and 590,731 tokens from the BioIns domain subsets. Because both LM branches are frozen, we estimate the training FLOPs of each backbone-specific injector as $2(P_{\text{act}}^{\text{backbone}} + 8\text{B})T_r$, corresponding to one forward pass through the backbone and memory; the injector's own compute is negligible. Training the injectors for Qwen3-8B/14B/32B jointly contributes 2.51 EFLOPs, while the Qwen3-235B-A22B injector training contributes 0.96 EFLOPs. Thus, standalone Qwen3-235B-A22B specialization includes 1.44 EFLOPs for datastore construction, 4.32 EFLOPs for memory

training, and 0.96 EFLOPs for injector training, totaling 6.73 EFLOPs. Reusing the same datastore and memory across all four backbones increases only the injector training component to 3.47 EFLOPs, yielding a total of 9.23 EFLOPs. These components are detailed in Table 15(a).

Measured runtime. To quantify the system operations omitted from the analytical model-FLOPs total, we additionally profile a representative BioIns specialization pipeline with a Qwen3-14B backbone on A800-80GB GPUs. The 8B memory stage is a profiling run that uses the same data and optimization settings as the main experiment. Table 15(b) reports the measured wall-clock and allocation footprint. The datastore preparation GPU-hours in Table 6(b) sum datastore construction, FAISS indexing, and teacher distribution construction. Aggregates and shares are computed before rounding.

Table 15: Complete Biology-Instructions adaptation accounting. (a) Model FLOPs and stage breakdown; relative costs use SFT within each scope, and totals use unrounded estimates. (b) Measured wall-clock time and allocation footprint on A800-80GB GPUs. The 8B memory profiling run uses the main experiment's data and optimization settings. GPU-hours equal allocated GPUs multiplied by wall-clock time.

(a) Analytical model FLOPs

	235B standalone		Four backbones	
Method / Stage	EFLOPs	Rel.	EFLOPs	Rel.
Full SFT	11.88	1.00×	41.05	1.00×
LoRA	7.92	0.67×	27.37	0.67×
MemSFT total	**6.73**	**0.57×**	**9.23**	**0.22×**
Datastore construction	1.44	–	1.44	–
8B memory training	4.32	–	4.32	–
Injector training	0.96	–	3.47	–

(b) Measured runtime: Qwen3-14B backbone + 8B memory

Stage	GPUs	Wall-clock	Allocated GPU-hours	Share (%)
Datastore construction	8	00:24:25	3.26	3.57
FAISS index construction	8	00:06:32	0.87	**0.95**
teacher distribution construction	8	00:05:31	0.74	**0.81**
8B memory training	8	10:35:11	84.69	92.77
Qwen3-14B injector training	1	01:44:05	1.73	1.90
Total	–	12:55:44	91.29	100.00

FAISS index construction and teacher distribution construction together take 12 minutes and 3 seconds and account for 1.61 allocated GPU-hours. Relative to the measured pipeline from datastore construction through Qwen3-14B injector training, these operations represent 1.55% of the cumulative wall-clock time and 1.76% of the allocated GPU-hours. Although they are not included in the analytical model-FLOPs total, their measured overhead is small relative to memory and injector training and does not materially alter the overall resource profile in this representative run.

E.2 INTERN-S2-PREVIEW-397B

Adaptation cost. We estimate model FLOPs from the training configuration used for the evaluated 397B model with a 4B memory. The expanded memory-training corpus contains 671,741,464 input tokens per epoch, and memory fine-tuning runs for three epochs, totaling 2,015,224,392 tokens. Injector training processes 56,905,050 full-sequence tokens. We count full sequences rather than only supervised positions because both frozen LM branches process the complete input. The MoE backbone uses 17B activated parameters per token, and the memory uses its nominal 4B parameter count.

For a matched comparison, full SFT and LoRA are estimated on the same data and training epochs. We use the same $6P_{\text{act}}T$ and $4P_{\text{act}}T$ formulas as above; the LoRA estimate omits adapter overhead. Datastore construction requires one memory-sized teacher forward pass over the corpus, memory

fine-tuning includes three forward/backward epochs, and injector training includes forward passes through both frozen branches. Table 16 reports these analytical model-FLOPs estimates, not measured full-SFT or LoRA training runs. FAISS indexing, neighbor search, and teacher distribution construction are excluded, consistently with the Qwen3 accounting.

Table 16: Adaptation FLOPs for the evaluated 397B configuration.

Method / stage	EFLOPs	Relative to SFT	Relative to LoRA
Full SFT	205.55	1.00×	1.50×
LoRA	137.04	0.67×	1.00×
Datastore construction	5.37	–	–
4B memory fine-tuning	48.37	–	–
Injector training	2.39	–	–
MemSFT total	**56.13**	**0.27**×	**0.41**×

Inference latency. We compare the frozen backbone and its 4B-memory extension using the same LMDeploy PyTorch backend, FP8 backbone, DP=4, EP=8, batch size 1, concurrency 1, and cache fraction 0.5. Speculative decoding and thinking are disabled. Both configurations receive the same requests with exactly 512 input tokens and generate exactly 128 output tokens using greedy decoding with EOS ignored. Each configuration has 10 warm-up requests followed by 50 measured requests. Model loading is excluded, and unique prompt nonces prevent shared-prefix cache reuse. Both configurations pass the fixed-length checks for all measured requests.

TTFT measures request submission to the first generated token; E2E measures request submission to the final token; TPOT measures time after the first token divided by the remaining output-token count. Adding memory increases E2E latency by 5.4% and TPOT by 4.1% in this configuration. The backbone and memory maintain separate weights and caches, and their forward passes execute sequentially before injection and sampling. These measurements characterize this single-request serving setup rather than high-concurrency throughput.

Table 17: Latency for 512 input and 128 output tokens. Overhead is relative to the same frozen 397B backbone without memory.

Metric	Backbone	+ MemSFT	Overhead
TTFT (s)	0.2231	0.2670	+19.7%
E2E latency (s)	2.9069	3.0641	+5.4%
TPOT (ms)	21.18	22.04	+4.1%

F Ablating the Injector and External Memory

We conduct a unified ablation to examine how the dynamic memory injector and the training strategy for the external module contribute to the overall MemSFT framework. We first keep the frozen Qwen3-14B backbone and Qwen3-8B memory unchanged, but replace dynamic memory injection with fixed interpolation coefficients. We then compare two systems using the injector with different external modules: a Qwen3-8B model trained by standard domain SFT and a Qwen3-8B memory trained with retrieval-based supervision, under the same training budget and settings. A separate injector is trained for each external module using the same architecture and training recipe. We evaluate all configurations on the official Biology-Instructions benchmark and the five general benchmarks used in the main experiments.

Table 18: Ablations of the injector and external memory with a frozen Qwen3-14B backbone. The fixed λ rows use the same Qwen3-8B memory without an injector. The final two rows compare a standard domain SFT model with a memory trained using retrieval-based supervision under dynamic memory injection. General Avg. is averaged over MATH-500, C-Eval, IFEval, MMLU-Redux, and INCLUDE. All scores are reported on a 0–100 scale.

Fusion Strategy	External Module	BioIns Avg.	MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	General Avg.
Backbone only	–	6.64	96.44	83.10	85.66	90.01	60.90	83.22
Fixed $\lambda = 0.1$	Qwen3-8B memory	6.28	96.32	82.84	84.40	89.77	62.74	83.21
Fixed $\lambda = 0.2$	Qwen3-8B memory	6.66	94.40	82.91	74.09	82.22	62.85	79.29
Fixed $\lambda = 0.3$	Qwen3-8B memory	6.94	83.56	82.99	58.41	67.15	62.62	70.95
Fixed $\lambda = 0.5$	Qwen3-8B memory	10.51	49.68	81.87	23.40	61.17	59.78	55.18
Fixed $\lambda = 0.7$	Qwen3-8B memory	29.65	22.04	73.85	14.68	60.14	51.01	44.34
Dynamic injector	Qwen3-8B SFT	37.34	96.72	83.06	86.14	90.32	62.80	83.81
Dynamic injector	**Qwen3-8B memory**	**42.92**	96.48	82.99	85.40	90.49	62.77	83.62

Table 18 shows that fixed interpolation cannot effectively balance domain specialization and general capability: increasing the fixed memory weight strengthens domain performance but increasingly interferes with the general benchmarks. Dynamic memory injection instead allows the memory to contribute selectively while preserving the behavior of the frozen backbone. With the dynamic memory injector, the memory trained with retrieval-based supervision also provides stronger domain specialization than the standard SFT external module, with comparable general performance. Together, these results show how the two components work within the complete MemSFT framework: retrieval-based supervision strengthens the external domain expert, while the injector controls how that expertise is integrated with the backbone.

G kNN-LM Baseline

Experimental setting We evaluate direct token-level retrieval using kNN-LM (Khandelwal et al., 2019), without training a parametric memory. At each decoding step, the next-token distribution is $(1 - \lambda)p_{\text{LLM}} + \lambda p_{\text{kNN}}$, with a fixed interpolation weight throughout generation. We use the frozen Qwen3-14B and a datastore constructed using Qwen3-14B from the same 500K Biology-Instructions training subset. Retrieval uses $k = 2047$, temperature $T = 16$, and FAISS (Douze et al., 2025) with `nprobe`=1. We evaluate all 21 tasks and 20,416 examples with the official parsers and metrics, using greedy decoding, thinking disabled, and a 512-token generation limit. We use the same prompt configuration as the backbone evaluation in the main experiments.

Table 19: Direct kNN-LM on Biology-Instructions with Qwen3-14B. Performance is nearly unchanged at small interpolation weights and deteriorates sharply at larger weights.

Fixed λ	0 (Backbone)	0.1	0.2	0.25	0.3	0.5	0.7
BioIns Avg.	6.64	6.68	**6.69**	6.67	6.59	-4.91	-14.49

Results and interpretation Table 19 shows little sensitivity to λ between 0.1 and 0.3, with the highest score of 6.69 at $\lambda = 0.2$, only 0.05 points above the backbone score of 6.64. Raising λ to 0.5 and 0.7 reduces the average to -4.91 and -14.49, respectively. Several classification tasks exhibit strongly negative MCC values, accompanied by deterioration in output formatting. For MCC, zero corresponds to chance-level prediction and negative values indicate performance below chance.

A fixed interpolation weight creates a tension between retrieval influence and generation stability. At small weights, the LLM can dominate next-token selection under greedy decoding, limiting the effect of retrieved domain information. At larger weights, the empirical distribution over retrieved next-token targets can override the LLM even where the retrieved domain patterns are inappropriate for the current context or output format. Unlike a trained memory LM, this retrieval branch does not learn to generalize beyond its retrieved neighbors and cannot reliably sustain fluent expressions and sentences when assigned a dominant weight in this setting. This interpretation is consistent with the fixed-interpolation ablation in Appendix F, where small memory weights yield limited domain gains

and larger weights degrade general performance. MemSFT combines a trained parametric memory with token-dependent injection to address these two limitations.

H Retrieval-Augmented Generation Baselines

Experimental setting We evaluate BM25, dense, and hybrid retrieval-augmented generation with the original post-trained Qwen3-14B on Biology-Instructions, without updating the generator parameters (Robertson & Zaragoza, 2009; Lewis et al., 2020). All methods retrieve from the same 500K training examples used by MemSFT, indexed globally as input–output pairs rather than in separate task-specific indexes. The original evaluation prompt serves as the query, and retrieved examples are prepended using the same prompt template. Evaluation covers all 21 tasks and 20,416 examples, using the same system prompt, task parsers, and metrics. Generation uses LMDeploy with greedy decoding, thinking disabled, a 32,768-token context limit, and at most 2,048 output tokens.

Retrieval methods The BM25 baseline uses lowercase word tokenization with $k_1 = 1.5$ and $b = 0.75$. We implement dense and hybrid retrieval using FlashRAG (Jin et al., 2025). Dense retrieval uses normalized BGE-M3 embeddings (Chen et al., 2024) and exact FAISS inner-product search (Douze et al., 2025) to retrieve 50 candidates, followed by cross-encoder reranking with `BAAI/bge-reranker-v2-m3`. Hybrid retrieval merges BGE-M3 top-50 and BM25s top-50 candidates, removes duplicate documents, and reranks the merged pool with the same cross-encoder. For both methods, we retain the top 50 reranked candidates and prepend the first k examples to the generation prompt.

We evaluate $k \in \{1, 3, 5, 10, 15, 20, 50\}$, where k denotes the number of examples supplied to the generator. Each retrieved example is truncated to at most 2,400 characters for $k \leq 20$ and 1,200 characters for $k = 50$ to fit the context budget. Thus, the top-50 setting changes both retrieval depth and per-example truncation. Figure 6 summarizes the results; Tables 20–22 report all per-task scores.

Results and interpretation Hybrid retrieval achieves the highest RAG average at every evaluated retrieval depth, reaching 13.45 at top-50 compared with 6.64 without retrieval. Dense retrieval outperforms BM25 through top-20, although BM25 performs slightly better at top-50 (12.23 versus 11.91). Despite these improvements, the strongest evaluated RAG configuration remains 29.47 points below MemSFT's 42.92. Gains vary across tasks and are not consistently monotonic in retrieval depth. These results show that adding dense retrieval and cross-encoder reranking does not close the specialization gap on the biological sequence tasks evaluated here. MemSFT instead learns domain knowledge from token-level retrieval supervision and selectively integrates the resulting memory predictions during generation.

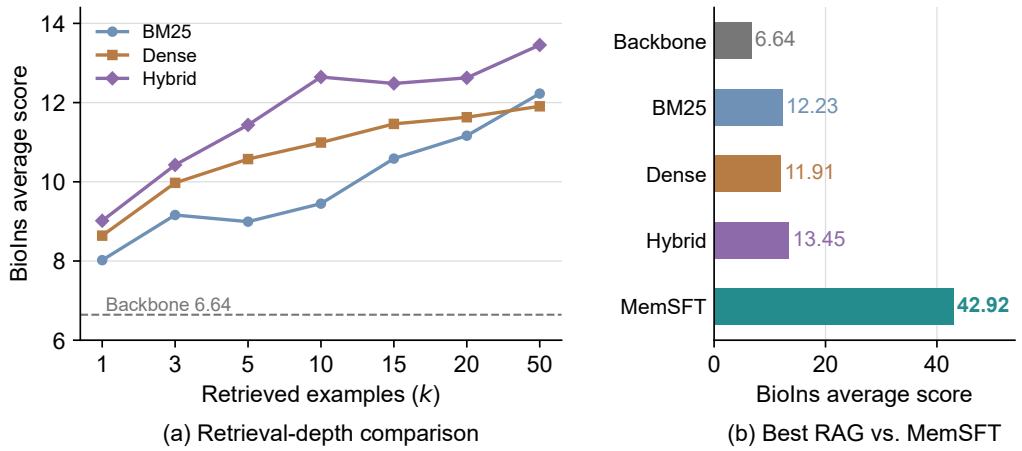


Figure 6: RAG baselines on BioIns with Qwen3-14B. (a) Hybrid retrieval achieves the highest RAG average at every tested k. (b) All three RAG methods attain their best average at top-50 but remain below MemSFT. Top-50 uses shorter per-example truncation than the other settings.

Table 20: Full BM25 RAG results on Biology-Instructions with Qwen3-14B. All values are OpenCompass summary metrics. Top-50 uses shorter retrieved-example truncation to keep the prompt within the context budget.

Task	Metric	Qwen3-14B	Top-1	Top-3	Top-5	Top-10	Top-15	Top-20	Top-50
DNA-cpd	MCC	-0.69	2.09	-3.47	-2.70	7.51	11.92	12.01	16.31
DNA-emp	MCC	0.20	1.19	3.86	2.59	1.63	3.89	-2.19	4.01
DNA-enhancer_activity	PCC	0.15	-3.04	-7.34	-3.91	-5.05	-5.22	-3.78	-6.63
DNA-pd	MCC	0.22	-1.65	-2.48	-5.86	-4.46	1.00	-2.40	-3.28
DNA-tf-h	MCC	-0.72	-1.39	4.11	3.90	4.61	1.14	1.85	0.44
DNA-tf-m	MCC	0.28	5.51	3.82	1.82	2.54	3.93	0.05	2.88
Multi_sequence-antibody_antigen	MCC	-6.04	2.17	-0.56	-1.39	-1.28	1.50	1.51	3.18
Multi_sequence-promoter_enhancer_interaction	MCC	-0.64	3.44	-0.81	-0.72	-3.51	-5.68	-0.17	0.58
Multi_sequence-rna_protein_interaction	MCC	-2.89	9.22	21.99	22.35	22.38	23.99	22.01	12.92
Multi_sequence-sirnaEfficiency	Mixed	33.85	38.10	42.18	45.52	46.35	48.98	48.22	51.57
Protein-Fluorescence	Spearman	-6.09	-2.77	2.72	1.51	8.40	5.62	12.23	12.76
Protein-FunctionEC	Score	2.93	3.12	2.88	3.20	3.23	3.62	4.53	5.09
Protein-Solubility	Acc.	51.50	50.30	51.60	53.00	55.30	56.70	58.70	58.40
Protein-Stability	Spearman	1.18	0.22	6.62	5.57	5.47	3.34	7.99	19.43
Protein-Thermostability	Spearman	-2.36	-3.81	-5.33	-2.13	-3.42	-3.81	-0.05	7.68
RNA-CRISPROnTarget	Spearman	1.06	2.75	10.74	5.22	-3.05	6.86	8.75	5.82
RNA-Isoform	R^2	0.17	0.02	0.00	0.10	0.21	1.33	1.67	2.18
RNA-MeanRibosomeLoading	R^2	0.74	0.08	0.11	0.01	1.01	0.38	0.24	0.34
RNA-Modification	AUC	50.67	49.99	50.18	49.58	49.90	49.72	49.80	49.62
RNA-NoncodingRNAFamily	Acc.	15.90	12.90	11.40	11.00	10.60	12.90	13.20	12.50
RNA-ProgrammableRNASwitches	R^2	0.09	0.00	0.16	0.20	0.04	0.20	0.26	0.98
Biology-Instructions AVG	–	6.64	8.02	9.16	8.99	9.45	10.59	11.16	12.23

Table 21: Full dense RAG results on Biology-Instructions with Qwen3-14B. Candidates are reranked with BGE-reranker-v2-m3 before selecting the top-k examples. Top-50 uses the same shorter per-example truncation as the BM25 baseline.

Task	Metric	Top-1	Top-3	Top-5	Top-10	Top-15	Top-20	Top-50
DNA-cpd	MCC	-1.21	7.26	4.99	4.57	2.38	2.81	2.77
DNA-emp	MCC	3.23	3.14	3.17	4.26	2.44	2.45	0.00
DNA-enhancer_activity	PCC	1.86	2.39	2.11	0.21	-0.50	1.68	-0.09
DNA-pd	MCC	3.77	3.26	2.31	-0.49	0.22	-2.59	-4.60
DNA-tf-h	MCC	-0.85	4.31	2.71	1.92	0.07	1.92	5.30
DNA-tf-m	MCC	-2.82	-0.51	2.72	1.82	2.14	1.43	6.11
Multi_sequence-antibody_antigen	MCC	3.47	-0.28	1.59	0.61	0.55	-0.83	0.00
Multi_sequence-promoter_enhancer_interaction	MCC	-0.20	-3.02	-0.39	-1.48	0.05	1.39	4.21
Multi_sequence-rna_protein_interaction	MCC	0.05	3.30	5.17	11.35	10.13	9.13	9.11
Multi_sequence-sirnaEfficiency	Mixed	41.34	46.56	48.20	48.27	49.25	51.23	50.97
Protein-Fluorescence	Spearman	6.01	9.85	10.34	8.23	12.87	9.90	11.85
Protein-FunctionEC	Score	2.77	5.57	4.77	5.10	5.65	5.80	5.94
Protein-Solubility	Acc.	55.90	58.10	59.70	59.40	60.50	61.80	62.50
Protein-Stability	Spearman	-4.99	-5.69	-7.27	-3.16	-4.25	-4.72	-3.22
Protein-Thermostability	Spearman	3.27	12.77	12.45	18.92	22.99	24.60	20.56
RNA-CRISPROnTarget	Spearman	5.38	-3.18	1.60	2.72	5.52	8.42	6.07
RNA-Isoform	R^2	0.18	0.00	0.34	0.13	0.04	0.10	0.39
RNA-MeanRibosomeLoading	R^2	0.04	0.04	0.00	0.00	0.00	0.13	0.00
RNA-Modification	AUC	49.52	50.16	50.76	50.88	51.15	51.05	51.25
RNA-NoncodingRNAFamily	Acc.	14.20	14.70	16.00	17.10	18.60	17.00	19.10
RNA-ProgrammableRNASwitches	R^2	0.53	0.71	0.77	0.46	0.90	1.54	1.82
Biology-Instructions AVG	–	8.64	9.97	10.57	10.99	11.46	11.63	11.91

Table 22: Full hybrid RAG results on Biology-Instructions with Qwen3-14B. Candidates are reranked with BGE-reranker-v2-m3 before selecting the top-k examples. Top-50 uses the same shorter per-example truncation as the BM25 baseline.

Task	Metric	Top-1	Top-3	Top-5	Top-10	Top-15	Top-20	Top-50
DNA-cpd	MCC	-1.63	4.55	8.39	7.86	7.90	11.76	7.59
DNA-emp	MCC	3.26	3.74	5.84	12.74	8.36	9.82	13.12
DNA-enhancer_activity	PCC	1.65	-1.51	1.38	-2.53	-2.30	0.26	0.43
DNA-pd	MCC	3.00	0.68	0.62	-0.95	-1.80	-4.19	-6.74
DNA-tf-h	MCC	2.02	2.37	3.80	3.35	1.56	2.18	3.60
DNA-tf-m	MCC	-0.39	3.15	5.86	2.93	1.24	0.37	-2.34
Multi_sequence-antibody_antigen	MCC	7.55	6.62	5.36	2.15	-1.02	4.00	1.38
Multi_sequence-promoter_enhancer_interaction	MCC	-1.80	-3.17	-0.55	-3.53	-1.50	0.72	-1.95
Multi_sequence-rna_protein_interaction	MCC	2.30	8.44	11.21	14.68	18.99	21.89	17.29
Multi_sequence-sirnaEfficiency	Mixed	41.76	51.15	50.91	54.03	52.03	53.84	52.05
Protein-Fluorescence	Spearman	6.96	9.12	10.40	11.53	10.06	9.96	7.86
Protein-FunctionEC	Score	3.03	5.60	4.81	4.82	6.07	5.70	6.65
Protein-Solubility	Acc.	56.90	57.70	58.10	60.80	61.40	61.20	61.40
Protein-Stability	Spearman	-4.84	-2.61	-1.55	-5.83	-1.92	-3.50	-1.73
Protein-Thermostability	Spearman	0.72	8.89	6.41	15.34	16.72	19.89	25.96
RNA-CRISPROnTarget	Spearman	3.50	-1.70	1.35	16.94	8.44	0.23	12.68
RNA-Isoform	R^2	0.12	0.03	0.26	0.01	6.48	0.09	9.33
RNA-MeanRibosomeLoading	R^2	0.00	0.01	0.03	0.03	0.01	0.02	0.00
RNA-Modification	AUC	49.74	50.64	49.80	51.12	50.26	50.75	50.18
RNA-NoncodingRNAFamily	Acc.	14.70	14.60	17.00	19.40	20.40	18.50	23.30
RNA-ProgrammableRNASwitches	R^2	0.78	0.64	0.73	0.67	0.76	1.64	2.48
Biology-Instructions AVG	–	9.02	10.43	11.44	12.65	12.48	12.63	13.45

I FORGETTING-MITIGATION BASELINES

Table 23 reports the full values underlying Figure 3. Both baselines for mitigating forgetting use the Qwen3-14B Biology-Instructions setting. MixTraining(1:1), following Liu et al. (2024), uses the same LoRA configuration as the main Qwen3-14B Biology-Instructions LoRA baseline, but trains on a 1:1 mixture of 500K Biology-Instructions examples and 500K general instruction examples sampled from NVIDIA's Nemotron-Post-Training-Dataset-v1 (Bercovich et al., 2025). For Wise-FT-style interpolation (Wortsman et al., 2022), we do not train a separate model; instead, we scale the Biology-Instructions LoRA update before merging it into the original backbone, i.e., $\theta = \theta_{\text{base}} + \gamma\Delta_{\text{LoRA}}$, with $\gamma \in \{0.2, 0.4, 0.6, 0.8\}$.

Table 23: Baseline results for mitigating forgetting on Qwen3-14B Biology-Instructions. MixTraining uses additional general instruction replay data, while Wise-FT interpolates between the original backbone and the Biology-Instructions LoRA model by scaling the LoRA update. All scores are reported on a 0–100 scale.

Method	Biology-Instructions Avg.	MATH-500	C-Eval	IFEval	MMLU-Redux	INCLUDE	General Avg.
Qwen3-14B	6.64	96.44	83.10	85.66	90.01	60.90	83.22
LoRA r8/lr3e-4	32.07	96.04	81.72	43.99	88.65	57.66	73.61
Wise-FT $\gamma = 0.2$	8.25	96.52	82.91	85.77	89.88	61.00	83.22
Wise-FT $\gamma = 0.4$	19.05	96.52	82.99	85.40	89.78	60.74	83.09
Wise-FT $\gamma = 0.6$	27.39	96.28	82.39	82.62	89.47	59.97	82.15
Wise-FT $\gamma = 0.8$	31.19	95.96	81.87	63.77	89.25	58.90	77.95
LoRA + MixTraining(1:1)	34.52	91.32	81.87	70.61	88.07	60.74	78.52
MemSFT	42.92	96.48	82.99	85.40	90.49	62.77	83.62

J ADDITIONAL MEMORY INJECTION CASE STUDIES

For the OpenSWI case in Figure 4, the mean memory weight is 0.20 for function tokens, 0.43 for domain terms, and 0.99 for numerical profile tokens. These statistics summarize memory weights over the complete answer passage; the token heatmap displays an abbreviated sequence.

Figures 4 and 7–11 analyze token-level memory weights from OpenSWI to BioIns, MATH, and MMLU. Each box gives the question and output, panel (a) shows token-level weights, and panel (b) gives their role-wise distribution; the cases progress from domain-dependent to general tasks.

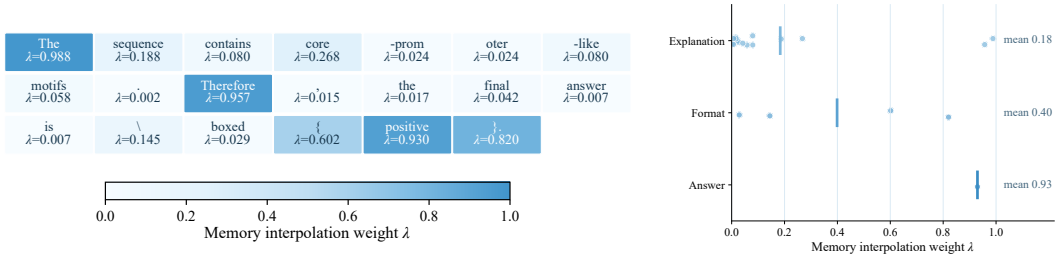
J.1 BioIns Case

Figure 7 shows mean λ rising from 0.18 for explanation tokens to 0.40 for boxed-format tokens and 0.93 for `positive`, concentrating memory on the specialized BioIns decision.

Answer the question: **BioIns Case**

Question: For core-promoter detection, is the following DNA sequence a core promoter?
<DNA>GAATGAACAGGAGTCGGTTCTCACCCAACTTCCATTAAGGACTCGGGGCAGGAGGGGCAG
AAGTTGCGCG</DNA>

Output: The sequence contains core-promoter-like motifs. Therefore, the final answer is positive.



(a) Token-level memory weights.

(b) Distribution of λ.

Figure 7: Dynamic memory injection on a BioIns core-promoter classification case. Panel (a) displays all 20 answer tokens. Panel (b) groups them into explanation, boxed-format, and answer tokens; vertical bars mark category means.

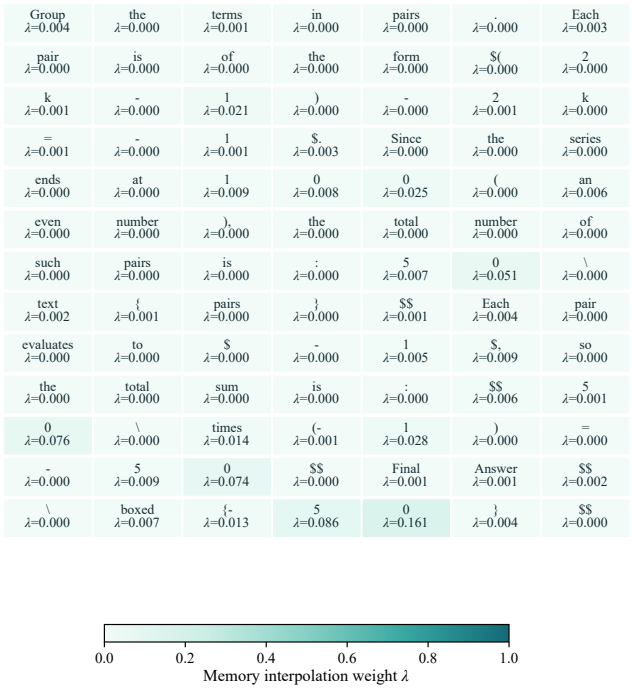
J.2 MATH CASE I: ALTERNATING SUM

Figure 8 contrasts the specialized BioIns answer with a general arithmetic derivation that stays backbone-dominant. Among the displayed tokens, text and mathematical-formatting tokens have mean weights of 0.001 and 0.002, respectively, while numeric tokens average 0.033. Memory weights are higher for numeric tokens but remain far below the memory weights on OpenSWI profile values.

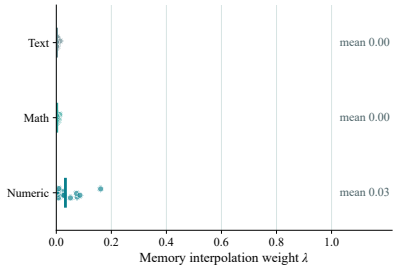
Answer the question: **MATH Case I**

Question: Compute $1 - 2 + 3 - 4 + 5 - \cdots + 99 - 100$.

Output: Group the terms in pairs. Each pair is of the form $(2k - 1) - 2k = -1$. Since the series ends at 100 (an even number), the total number of such pairs is: 50 pairs Each pair evaluates to -1, so the total sum is: $50 \times (-1) = -50$ Final Answer $\boxed{-50}$



(a) Token-level memory weights.



(b) Distribution of λ.

Figure 8: Dynamic memory injection on the alternating-sum MATH case. Panel (a) follows the displayed derivation from pairing consecutive terms to the boxed result. Panel (b) groups the same displayed tokens into text, mathematical-formatting, and numeric categories.

J.3 MATH CASE II: COMPLEX ROTATION

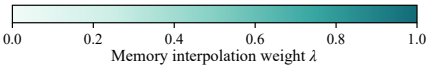
Figure 9 is a stricter control with low memory weights. Across the displayed excerpts of the three-step calculation, text and mathematical-formatting tokens have mean weights of 0.0006 and 0.0009, respectively, while numeric tokens average 0.0169. Every displayed token remains below 0.1 (maximum 0.0942), showing that even a multi-step symbolic calculation can rely almost entirely on the backbone.

Answer the question: **MATH Case II**

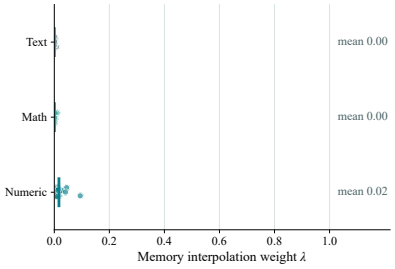
Question: Let $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$ and $c = 2 - 3i$. Let w be the result when z is rotated around c counter-clockwise by $\pi/4$. Find w.

Output: Step 1: Translate the point z by subtracting c. Move the center of rotation to the origin: $z - c = \sqrt{2} - 3\sqrt{2}i$. **Step 2: Apply the rotation by $\pi/4$.** $(1 - 3i)(1 + i) = 1 + i - 3i - 3i^2 = 1 - 2i + 3 = 4 - 2i$. **Step 3: Translate back by adding c.** $w = (4 - 2i) + (2 - 3i) = 6 - 5i$. **Final Answer:** $\boxed{6 - 5i}$.

Step λ=0.000	1 λ=0.021	: λ=0.000	Translate λ=0.002	the λ=0.000	point λ=0.000	$ λ=0.000
z λ=0.000	$ λ=0.000	by λ=0.000	subtract λ=0.000	ing λ=0.000	$ λ=0.000	c λ=0.000
… (middle omitted)	z λ=0.000	- λ=0.000	c λ=0.000	= λ=0.000	\ λ=0.000	sqrt λ=0.001
{ λ=0.002	2 λ=0.011	} λ=0.000	- λ=0.000	3 λ=0.001	\ λ=0.000	sqrt λ=0.000
{ λ=0.004	2 λ=0.017	} λ=0.002	i λ=0.000	$$ λ=0.002	Step λ=0.000	2 λ=0.013
: λ=0.000	Apply λ=0.008	the λ=0.000	rotation λ=0.000	by λ=0.000	$ λ=0.000	\ λ=0.000
frac λ=0.002	{\ λ=0.001	pi λ=0.000	}{ λ=0.002	4 λ=0.045	} λ=0.001	$ λ=0.001
(λ=0.000	1 λ=0.004	- λ=0.000	3 λ=0.006	i λ=0.000	)(λ=0.003	1 λ=0.006
+ λ=0.000	i λ=0.000	) λ=0.000	$$ λ=0.001	… (middle omitted)	= λ=0.000	1 λ=0.011
- λ=0.000	2 λ=0.015	i λ=0.000	+ λ=0.001	3 λ=0.011	= λ=0.000	4 λ=0.006
- λ=0.000	2 λ=0.009	i λ=0.000	$$ λ=0.000	Step λ=0.000	3 λ=0.003	: λ=0.000
Translate λ=0.003	back λ=0.000	by λ=0.000	adding λ=0.000	$ λ=0.000	c λ=0.000	$ λ=0.000
w λ=0.001	= λ=0.000	(λ=0.000	4 λ=0.006	- λ=0.000	2 λ=0.003	i λ=0.001
) λ=0.000	+ λ=0.000	(λ=0.002	2 λ=0.027	- λ=0.000	3 λ=0.015	i λ=0.001
) λ=0.000	= λ=0.000	6 λ=0.094	- λ=0.000	5 λ=0.013	i λ=0.001	$$ λ=0.000
Final λ=0.001	Answer λ=0.001	$$ λ=0.001	\ λ=0.000	boxed λ=0.010	{ λ=0.004	6 λ=0.041
- λ=0.000	5 λ=0.010	i λ=0.001	} λ=0.003	$$ λ=0.001		



(a) Token-level memory weights.



(b) Distribution of λ.

Figure 9: Dynamic memory injection on a complex-rotation MATH case. Panel (a) traces the three-step solution, with intermediate explanatory and algebraic tokens omitted for compactness. Panel (b) groups the displayed tokens into text, mathematical-formatting, and numeric categories.

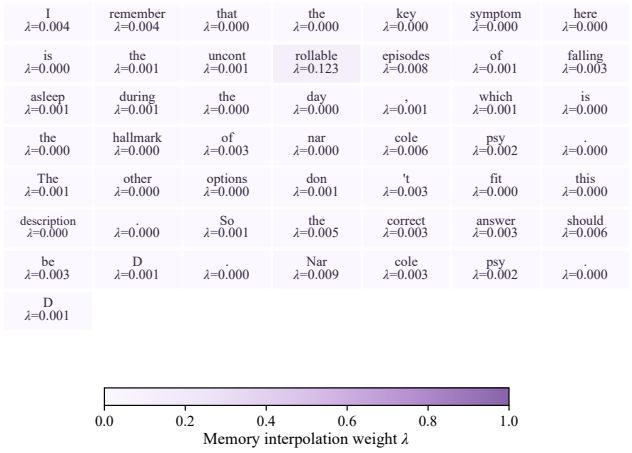
J.4 MMLU CASE I: NARCOLEPSY

Figure 10 considers a general factual-recognition question rather than a BioIns-memory task. Within the displayed sequence, function and content tokens have mean weights of 0.0012 and 0.0070, respectively, and the final D token has weight 0.0006. The injector therefore leaves both the explanation and the decision to the frozen backbone.

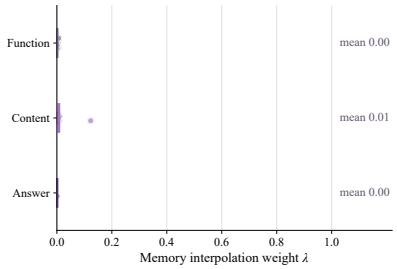
Answer the question: **MMLU Case I**

Question: Which disorder is characterized by uncontrollable episodes of falling asleep during the day? **A.** Dyslexia; **B.** Epilepsy; **C.** Hydrocephalus; **D.** Narcolepsy.

Output: I remember that the key symptom here is the uncontrollable episodes of falling asleep during the day, which is the hallmark of narcolepsy. The other options don't fit this description. So the correct answer should be D. Narcolepsy. D



(a) Token-level memory weights.



(b) Distribution of λ.

Figure 10: Dynamic memory injection on an MMLU anatomy question about narcolepsy. Panel (a) shows the concluding analysis and final option token. Panel (b) groups exactly the displayed sequence into function, content, and answer tokens.

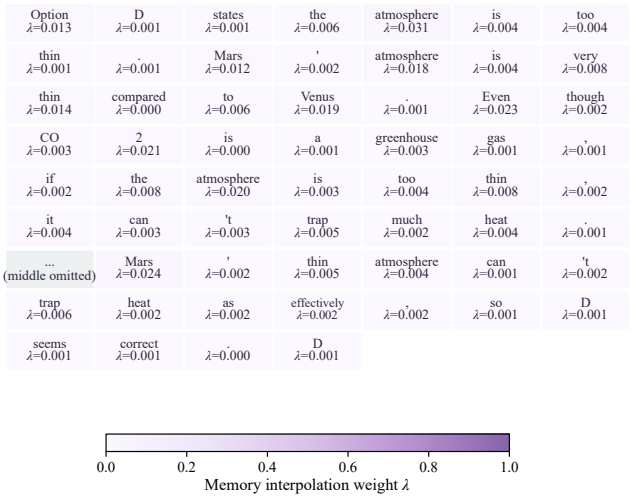
J.5 MMLU Case II: Mars Greenhouse Effect

Figure 11 shows a second uniformly backbone-dominant MMLU example. Across the complete displayed answer, function and content tokens have mean weights of 0.0027 and 0.0066, and the final D token has weight 0.0005; the complete reasoning-and-answer trace also remains below 0.1 (maximum 0.0591). Together with Figure 4 and the other appendix cases, this comparison indicates that memory use follows task relevance: it is strongest for specialized biological labels and geophysical profile values, remains low for ordinary mathematical reasoning, and is negligible for general multiple-choice answers.

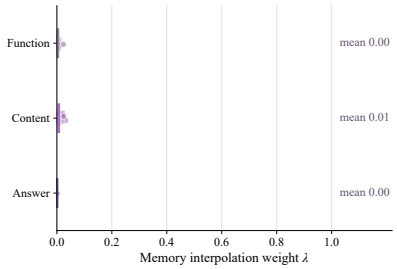
Answer the question: **MMLU Case II**

Question: Mars has an atmosphere that is almost entirely carbon dioxide. Why is there not a strong greenhouse effect keeping the planet warm? **A.** Mars does not have enough internal heat to drive the greenhouse effect; **B.** Mars is too far from the Sun for the greenhouse effect to work; **C.** the greenhouse effect requires an ozone layer, which Mars does not have; **D.** the atmosphere on Mars is too thin to trap a significant amount of heat.

Output: Option D states the atmosphere is too thin. Mars' atmosphere is very thin compared to Venus. Even though CO_2 is a greenhouse gas, if the atmosphere is too thin, it can't trap much heat. That makes sense. Venus has a dense atmosphere, so even though it's CO_2, it's effective. Mars' thin atmosphere can't trap heat as effectively, so D seems correct. D



(a) Token-level memory weights.



(b) Distribution of λ.

Figure 11: Dynamic memory injection on an MMLU astronomy question about Mars' greenhouse effect. Panel (a) displays the retained tokens from the output, with a middle span omitted for compactness. Panel (b) groups the output tokens into function, content, and answer categories.