Anchoring What Drifts: Layer Representation Preservation for LLM Fine-Tuning

Anonymous ACL submission

Abstract

Fine-tuning a large language model on a downstream task improves in-domain performance but erodes general capability. Existing anti-forgetting methods either regularize broadly or apply parameter-level heuristics; neither asks where in the network the representational drift actually concentrates. We address this by measuring *per-layer hidden-state drift*: the distance between a fine-tuned model and its frozen pretrained reference at every decoder layer on a held-out anchor batch, available from a single forward pass. Drift is sharply non-uniform across decoder layers, happens in early and last layers, and the profile is stable across probe distributions. Building on this, we propose **RepAnchor**, a fine-tuning framework that anchors representations by penalizing per-layer hidden-state MSE on an out-of-distribution corpus, using a cosine-decaying regularization weight λ. RepAnchor leaves the backbone untouched, works under both full-parameter and LoRA fine-tuning, and is insensitive to the choice of anchor corpus. Across eight baselines spanning five anti-forgetting families, four task datasets, and three model scales, RepAnchor leads on held-out retention while staying competitive on task accuracy.

1 Introduction

Deploying large language model typically entails using pretrained backbone and adapting it to target domain. Fine-tuning reliably improves in-domain performance, but at a cost: the model loses broad competence as it gains narrow expertise. This degradation has been documented across instruction following, code generation, mathematical reasoning, and domain question answering (Luo et al., 2025), and it grows with gap between the task distribution and the pretraining mixture. The standard remedy is an anti-forgetting regularizer: parameter-space penalties (Kirkpatrick et al., 2017; Xuhong et al., 2018), output-distribution distillation (Li and Hoiem, 2017; Hinton et al., 2015), architectural separation (Wei et al., 2025; Wu et al., 2024), or replay against a generic corpus (Zhang et al., 2025). These methods differ in *what* they regularize, but they share a limitation in granularity: they apply constraints to final output distribution or rely on parameter-level heuristics. Neither targets a more direct intervention: anchoring representational drift on per-layer basis.

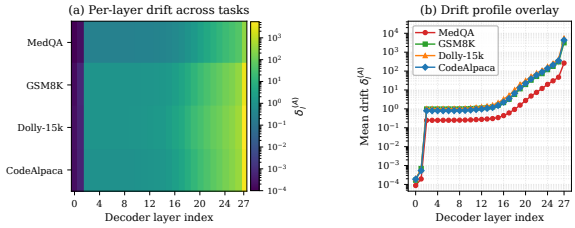

Figure 1: Per-layer Type-A drift, Full Qwen3-1.7B on four tasks (log y). Drift is the layer-wise representation distance between a fine-tuned model and its frozen pretrained reference on the same input; Type-A is its dimension-normalized squared ℓ_2 form (Eq. 1). **(a)** Heatmap over 28 layers; **(b)** per-task profiles. Probe: 64-sample held-out Alpaca.

We start with a diagnostic question: *where in the model does general-purpose knowledge degrade under task fine-tuning?* We answer it by measuring *per-layer hidden-state drift*, distance between hidden states of a fine-tuned model and its frozen pretrained reference on same input. The measurement runs from any checkpoint in single forward pass on held-out anchor batch, with no training-time hooks or learned probes (Belrose et al., 2023). While prior works have analyzed layer-wise behaviors (Shi et al., 2024; Zheng et al., 2025), we observe that representational drift is highly non-uniform across decoder layers in Figure 1, spanning several orders of magnitude: first three and last layers drift substantially more than the middle blocks, and the same profile recurs across all four tasks we evaluate.

This profile motivates **RepAnchor**, a fine-tuning framework that regularizes per-layer hidden-state mean squared error (MSE) against a frozen pre-

trained reference on an out-of-distribution (OOD) anchor stream interleaved with task batches, weighted by a cosine-decayed coefficient λ. The reference-acquisition mechanism is decoupled from the fine-tuning regime: an independent frozen copy for full-parameter fine-tuning, and a toggled adapter for LoRA (Hu et al., 2022), with no change to the operator or the primary loss. RepAnchor pairs per-layer projection as a placement signal (Geva et al., 2022; Belrose et al., 2023) with hidden-state distillation as the operator (Romero et al., 2015; Sanh et al., 2019; Douillard et al., 2020).

We evaluate RepAnchor on three Qwen scales (Yang et al., 2025) under both full-parameter and LoRA fine-tuning, across four task axes and eight baselines from the five anti-forgetting families surveyed. RepAnchor leads in general-knowledge and instruction-following retention at all three scales while remaining competitive on task accuracy, and at 8B RepAnchor-LoRA rises to a column-mean lead on task accuracy as well, with one documented regression in long-form summarization. Our contributions are: **1. Per-layer drift diagnostic.** A training-free diagnostic that, from a single forward pass, extracts a structural drift profile and pinpoints where representations degrade. **2. Architecture-agnostic framework (RepAnchor).** A dual-stream method that anchors hidden states to a frozen reference via a cosine-decayed MSE penalty (λ). It leaves the backbone architecture unchanged and supports both full-parameter and LoRA fine-tuning. **3. Strong retention and robustness.** RepAnchor leads the retention column-mean averages at all three model scales. Performance is also insensitive to the size and source of the anchor corpus, so the method does not depend on a carefully curated replay distribution.

2 Our Method

Our method couples a per-layer drift diagnostic (Section 2.1) with the RepAnchor (Section 2.2).

2.1 Layers Drift Diagnostic

2.1.1 Drift formalism

We define **per-layer drift** as layer-wise representation distance between a fine-tuned model and frozen pretrained reference on the same input. Let $\hat{h}_i(x) \in \mathbb{R}^d$ be the layer-i hidden state of the *fine-tuned* model on a probe input x, and let $h_i(x) \in \mathbb{R}^d$ be the layer-i hidden state of the *frozen pretrained reference* on the same x, for $i \in \{1, \ldots, k\}$. The *Type-A drift* used throughout this work is the dimension-normalized squared ℓ_2 distance:

$$\delta_i^{(A)}(x) \;=\; \frac{1}{d}\left\|\hat{h}_i(x) - h_i(x)\right\|_2^2. \qquad (1)$$

We report the mean of $\delta_i^{(A)}$ over a held-out probe batch; cosine similarity and raw ℓ_2-norm are available as robustness checks. Squared ℓ_2 is the primary choice for two reasons. Cosine similarity discards the magnitude information that RMSNorm-equipped decoders carry in the residual stream and that the late-stack drift profile (Section 2.1.2) is dominated by. The squared form (rather than raw ℓ_2) also extends cleanly to a training objective of the same form (Section 2.2.2): the gradient $\nabla\|\hat{h}_i - h_i\|_2^2 = 2(\hat{h}_i - h_i)$ is the drift vector itself, so the optimization signal scales with the very quantity the diagnostic measures. Computing $\{\delta_i^{(A)}\}_{i=1}^k$ from checkpoint requires one forward pass of model and one of the reference on the same batch, no training-time hooks and no learned probe.

2.1.2 Drift profiles

We measure drift profiles on Qwen3-1.7B checkpoints fine-tuned without any anti-forgetting regularizer, i.e. vanilla full fine-tuning (Full), across four tasks: MedQA (Jin et al., 2021), GSM8K (Cobbe et al., 2021), Dolly-15k (Conover et al., 2023), and CodeAlpaca-20k (Chaudhary, 2023). The probe is a held-out 64-sample batch drawn from the Alpaca (Taori et al., 2023) instruction-tuning corpus. Alpaca is out-of-distribution relative to each task, which surfaces drift in regions of the input space the task data does not exercise. Figure 1 reports the resulting profiles: panel (a) is a heatmap with four rows (one per task) and 28 columns (one per Qwen3-1.7B decoder layer), color-coded by mean $\delta_i^{(A)}$ on a log scale; panel (b) overlays the four per-task drift curves on a log y-axis.

The profile is sharply **non-uniform** across decoder layers, and non-uniformity spans more than seven orders of magnitude. Across all four tasks, drift is essentially zero in two embedding-adjacent layers ($\delta_i^{(A)} \sim 10^{-4}$), grows gently through the middle of the decoder ($\delta_i^{(A)} \sim 10^{0}$), then ramps steeply over the final third of the stack before exploding at the last layer, which carries roughly an order of magnitude more drift than its predecessor (e.g. 3.1×10^3 at layer 27 versus 3.1×10^2 at layer 26 for GSM8K). The shape recurs across all four tasks: math, code, and instruction-following share nearly identical curves, and domain QA (MedQA) tracks

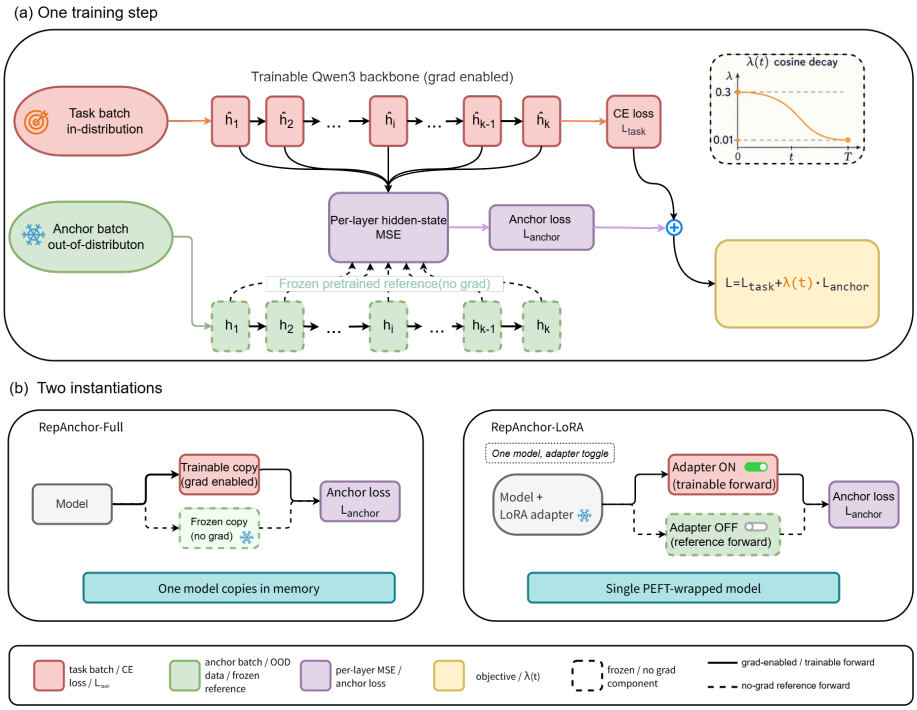

Figure 2: RepAnchor framework. Overview of the dual data stream (task batch + OOD anchor batch), the cosine-decayed λ schedule, the per-layer hidden-state MSE operator against the frozen pretrained reference, and the two reference-acquisition mechanics (-Full: two model copies; -LoRA: on a single PEFT-wrapped model).

the same shape at roughly $10\times$ lower magnitude. A regularizer that respects this profile must be *per-layer*: any signal that aggregates across stack treats a near-stationary layer the same as one that has shifted by three orders of magnitude.

2.1.3 Diagnostic implications for the operator

We therefore couple the diagnostic to the operator. The regularizer has the same form as the diagnostic: a weighted per-layer sum of the drift quantity, batch-averaged over the anchor stream,

$$\tilde{\mathcal{L}}_{\text{anchor}} = \sum_{i=1}^{k} w_i \cdot \bar{\delta}_i^{(A)}, \qquad w_i = 1/k, \qquad (2)$$

where $\bar{\delta}_i^{(A)}$ is the attention-masked mean of $\delta_i^{(A)}$ over the anchor batch. The actual operator $\mathcal{L}_{\text{anchor}}$ realizes $\tilde{\mathcal{L}}_{\text{anchor}}$ with explicit token-level aggregation in Section 2.2.2 (Eq. 4).

Uniform weighting at the operator level matches the diagnostic profile rather than ignoring it. Because $\tilde{\mathcal{L}}_{\text{anchor}}$ sums per-layer squared distances, a layer at $\delta_i^{(A)} \sim 10^3$ contributes orders of magnitude more to the loss than one at $\delta_i^{(A)} \sim 10^0$, even when w_i is identical. The non-uniform allocation of regularization pressure across layers is therefore a consequence of the per-layer operator acting on a non-uniform drift profile; no hand-prescribed weights are needed, and our experiments use none. The same reasoning explains why we anchor every decoder layer rather than a selected band: low-drift layers contribute little to the loss *by construction*, so including them is nearly free and keeps the operator architecture-agnostic.

2.2 RepAnchor

2.2.1 Framework

RepAnchor (Figure 2) has three components: a dual data stream that interleaves task batches with out-of-distribution anchor batches, a cosine-decayed coefficient λ that schedules the anchor-loss weight, and a frozen pretrained reference whose acquisition mechanic is decoupled from the fine-tuning regime. At each optimizer step the model consumes one task batch (optimized with cross-entropy loss and label smoothing 0.1) and one anchor batch drawn from a held-out OOD corpus (Alpaca by default[1]) used only for the anchor loss. The anchor dataloader cycles infinitely so the OOD signal stays available throughout training. The combined objective is

$$\mathcal{L}(t) = \mathcal{L}_{\text{task}} + \lambda(t) \cdot \mathcal{L}_{\text{anchor}}, \qquad (3)$$

where $\lambda(t)$ follows a cosine decay from $\lambda_{\max} = 0.3$ to $\lambda_{\min} = 0.01$ over the total number of optimizer steps. The schedule applies strong anchoring

[1] "Alpaca" denotes a 1k held-out subsample, used uniformly as anchor or replay corpus throughout (including GeRe-Alpaca); pool-size and source robustness in Section 4.5.

during the disruptive early phase of fine-tuning, when the task gradient first pulls representations away from the pretrained manifold, then relaxes near the end so the model can converge to the task. The framework is regime-agnostic: it applies to any fine-tuning method that preserves the layer-wise forward graph of the decoder, and the only component that changes across instantiations is *how* the reference forward is obtained (Section 2.2.3).

2.2.2 Operator

Choice of constraint space. Anti-forgetting penalties can be imposed on three candidate spaces, and the choice determines what the regularizer can see. *Parameter-space* penalties (L2-to-init, EWC (Kirkpatrick et al., 2017)) suffer from the non-uniqueness of the parameter-to-function map: the same behavior is realized by infinitely many parameter configurations under layer-wise permutations and channel-wise rescalings, so parameter shifts need not track functional shifts. *Output-space* penalties (LwF (Li and Hoiem, 2017) and the broader KD line) collapse the k-layer pipeline into a single head-level distribution and are structurally blind to the mid-stack drift that Section 2 flags as load-bearing. *Representation-space* penalties act on the per-layer hidden states themselves. RepAnchor anchors in this third space, consistent with the per-layer drift $\delta_i^{(A)}$ that motivated the framework.

Definition. The anchor loss is the per-layer hidden-state MSE against the frozen reference, attention-masked to ignore padding:

$$\mathcal{L}_{\text{anchor}} = \sum_{i=1}^{k} w_i \cdot \frac{\sum_{b,t} m_{b,t} \left\| \hat{h}_i^{(b,t)} - h_i^{(b,t)} \right\|_2^2}{\left(\sum_{b,t} m_{b,t}\right) \cdot d}, \tag{4}$$

where $\hat{h}_i^{(b,t)}$ and $h_i^{(b,t)}$ are the layer-i hidden states of the *fine-tuned* and *frozen-reference* models for token t in batch element b of the anchor stream, $m_{b,t} \in \{0, 1\}$ is the attention mask, and d is the hidden dimension. Each per-layer term is the masked mean of dimension-normalized squared ℓ_2 distance, where Type-A drift quantity from Eq. 1 averaged over the anchor batch; so Eq. 4 is token-explicit realization of $\tilde{\mathcal{L}}_{\text{anchor}} = \sum_i w_i \, \bar{\delta}_i^{(A)}$ from Eq. 2.

Aggregation and numerics. Per-token *mean* (rather than sum) decouples the loss from sequence length; division by d normalizes each per-layer term to a per-coordinate squared error so magnitudes remain comparable across hidden sizes; the attention mask excludes padding residuals that would dilute the mean. Following standard practice for hidden-distillation losses, both hidden states are cast to fp32 before the difference, avoiding catastrophic cancellation in bf16 near late-stack layers where the two models agree most closely.

Layer weighting and density. We use uniform per-layer weights $w_i = 1/k$ across every decoder layer; the rationale is given in Section 2.1.3. Acting on a non-uniform drift profile, the per-layer operator already concentrates pressure where it is needed, so explicit non-uniform w_i is redundant and empirically counterproductive. Anchoring every layer keeps the operator architecture-agnostic.

Positioning. The operator itself belongs to the per-layer hidden-distillation family (FitNets (Romero et al., 2015), DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), PODNet (Douillard et al., 2020)); its mathematical form is not new. What turns RepAnchor into an anti-forgetting framework rather than a compression method is what surrounds the operator: OOD-anchor data stream and cosine-λ schedule that decide where and when it applies, the regime-decoupled reference acquisition that makes the same operator deployable under full and LoRA fine-tuning, and per-layer drift diagnostic that motivates the choice of representation space, uniform layer weights, and per-layer density.

2.2.3 Instantiations across FT regimes

The reference forward, the source of $h_i^{(b,t)}$, is the only component of RepAnchor that depends on the fine-tuning regime. We report two instantiations.

RepAnchor-Full holds two model copies in memory: a trainable copy updated with the gradient of $\mathcal{L}_{\text{task}} + \lambda(t)\,\mathcal{L}_{\text{anchor}}$, and a frozen reference copy. Forward hooks on the decoder-layer list of both copies capture per-layer hidden states; the hooks handle both tuple and bare-tensor return shapes of the decoder layer (a tuple under `output_attentions`, a tensor otherwise).

RepAnchor-LoRA wraps a single PEFT-adapted model and obtains the reference forward by calling `model.disable_adapter()` inside a no-grad block, which removes the second model copy. Hooks are registered on the decoder-layer list *before* PEFT wrapping: PEFT replaces the inner `Linear` modules but leaves the layer objects intact, so the hook references stay valid. Captured reference hidden states are immediately `detach().clone()`'d; the clone is load-bearing under gradient checkpointing, where recomputation of the trainable forward would otherwise overwrite the reference buffers.

3 Experimental Setup

3.1 Datasets and metrics

We fine-tune on four task datasets spanning four representative downstream axes: **MedQA** (classification-style domain QA), **GSM8K** (multi-step math), **Dolly-15k** (open-ended instruction-following), and **CodeAlpaca-20k** (code generation). Task performance is measured by MedQA / GSM8K accuracy, Dolly ROUGE-Lsum (Lin, 2004) on a 500-sample held-out subset, and MBPP (Austin et al., 2021) pass@1 as a true held-out probe of CodeAlpaca-trained code generation (training and test sets are independent). Held-out retention is measured on MMLU-Pro (Wang et al., 2024) (macro accuracy) and IFEval (Zhou et al., 2023b) (prompt-strict); held-out generation on CNN/DailyMail (Hermann et al., 2015) ROUGE-L. Per-cell results are in Appendix A.

3.2 Baselines

We compare RepAnchor against eight baselines spanning five regularization families, plus the two RepAnchor instantiations introduced in Section 2. **Group A** (parameter-efficiency): Full, LoRA at rank 16, and DoRA (Liu et al., 2024) at rank 16. **Group B** (anti-forgetting, full-parameter side): L2-SP (Xuhong et al., 2018), LwF (Li and Hoiem, 2017), GeRe (Zhang et al., 2025), and Control LLM (Wei et al., 2025); LoRA side: O-LoRA (Wang et al., 2023). **Group C**: RepAnchor-Full and RepAnchor-LoRA. Per-method configuration details are in Appendix B.

3.3 Models and training

All experiments use Qwen3-1.7B, 4B, and 8B instruction-tuned backbones in bfloat16 mixed precision, trained for one epoch with AdamW and a cosine learning-rate schedule. Full-parameter methods use a learning rate of 1×10^{-5}; parameter-efficient methods use 1×10^{-4}. Complete hyperparameter details are in Appendix B.

4 Results and Analysis

4.1 Main comparison

Table 1 reports per-dataset task accuracy (left block) and held-out retention (right block, each metric already averaged over the four fine-tuning runs) for the ten methods at Qwen3-1.7B, 4B, and 8B, with a per-block **Avg** column summarizing each side. Two patterns stand out.

Retention. RepAnchor tops the Retention **Avg** at all three scales: 42.84 / 42.66 (Full / LoRA) at 1.7B versus the strongest baseline Control LLM 42.31; 50.72 / 50.87 at 4B versus LwF 50.48; and 52.69 / 53.26 at 8B versus Control LLM 53.01 (RepAnchor-LoRA leads). The Avg lead is supported per-metric: at least one RepAnchor instantiation tops every retention-metric column mean at 1.7B and 4B; at 8B RepAnchor-LoRA leads MMLU-Pro (49.39) and trails the IFEval and CNN/DM leaders by at most 1.4 pt. Per-cell breakdown is in Appendix A.

Task accuracy. On the Task **Avg**, RepAnchor-Full leads at 1.7B (44.39, ahead of LoRA 43.32), sits within half a point of task-side leaders at 4B (53.65 / 53.49 for Full / LoRA vs. DoRA 54.11 and LoRA 53.68), and is taken over by RepAnchor-LoRA at 8B (56.89, ahead of RepAnchor-Full 56.77 and GeRe-SlimPajama 56.56). Per-column wins favor RepAnchor: CodeAlpaca-MBPP and Medical-QA at 1.7B (Full), the largest Medical-QA jump over Full at 4B (52.32 vs. 41.79), and CodeAlpaca-MBPP at 8B (RepAnchor-Full 61.00, ahead of Control LLM 60.80) together with Dolly-RLsum at 8B (RepAnchor-LoRA 33.56). Two caveated rows behave atypically. Control LLM's MedQA accuracies (1.7B 22.23, 4B 8.09, 8B 25.14) are structurally low because the 7-expanded-block configuration is weak on multiple-choice tasks (Appendix B). GeRe-Alpaca collapses on IFEval for CodeAlpaca and Dolly fine-tuning (8–13 at 1.7B; 6–13 at 4B; uniformly 14.60 across all four cells at 8B with task accuracy at zero) despite using the same Alpaca anchor corpus as RepAnchor, isolating per-layer MSE mechanism, not anchor corpus, as operative driver of RepAnchor's retention advantage.

4.2 Scaling behavior

In Table 1, RepAnchor closes the accuracy gap with the strongest baseline at 4B. At 8B, it sweeps the top two Task-Avg spots (RepAnchor-LoRA 56.89, RepAnchor-Full 56.77), outperforming all baselines. RepAnchor-Full beats Full on Medical-QA at both 1.7B and 4B, with the lead widening from $+4.1$ to $+10.5$ pts; on CodeAlpaca-MBPP the 1.7B leader RepAnchor-Full cedes the top spot at 4B to Control LLM but reclaims it at 8B (61.00 vs. Control LLM 60.80). On GSM8K it sits within half a point of the LoRA leader at 1.7B and 4B; at 8B O-LoRA edges out column (81.50) with RepAnchor-LoRA 1.1 pt behind. The MMLU-Pro and IFEval retention advantages **persist** across scales (RepAnchor-LoRA tops MMLU-Pro at all three; on IFEval RepAnchor-Full leads at 1.7B and

Method	Task accuracy					Held-out retention			
	CodeAlpaca MBPP	Dolly RLsum	GSM8K acc	Medical acc	Avg	MMLU macro	IFEval pStrict	CNN RLsum	Avg
Qwen3-1.7B									
Full	41.40	**31.13**	55.04	39.20	41.69	31.54	60.40	28.73	40.22
LoRA	43.00	30.21	**58.98**	41.08	43.32	31.75	49.72	28.72	36.73
DoRA	43.40	29.94	56.63	42.70	43.17	31.58	49.21	28.70	36.50
L2-SP	41.40	30.16	58.61	41.16	42.83	31.72	61.69	**28.83**	40.75
LwF	42.20	29.30	52.46	42.26	41.56	31.41	62.71	28.70	40.94
GeRe-SlimPajama	41.60	30.15	55.72	40.85	42.08	30.65	57.67	28.61	38.98
GeRe-Alpaca	40.20	30.27	56.79	38.41	41.42	23.37	19.32	19.95	20.88
Control LLM	44.00	28.98	58.68	22.23	38.47	32.14	66.73	28.05	42.31
O-LoRA	40.20	28.79	51.18	32.52	38.17	31.47	59.38	28.56	39.80
RepAnchor-Full	**44.60**	30.71	58.95	**43.30**	**44.39**	32.33	**67.38**	28.82	**42.84**
RepAnchor-LoRA	43.80	30.14	57.09	42.81	43.46	**32.77**	66.40	28.81	42.66
Qwen3-4B									
Full	51.00	32.10	76.57	41.79	50.37	43.81	75.46	29.42	49.56
LoRA	53.20	31.78	**76.95**	52.79	53.68	43.76	66.36	29.37	46.50
DoRA	54.00	31.63	75.82	**54.99**	**54.11**	43.92	67.10	29.21	46.74
L2-SP	54.20	30.60	75.74	37.16	49.43	43.83	77.59	29.35	50.26
LwF	54.40	32.00	76.57	36.14	49.78	43.84	78.56	29.05	50.48
GeRe-SlimPajama	52.40	31.95	75.51	32.60	48.12	43.55	75.83	**29.53**	49.64
GeRe-Alpaca	49.60	32.00	76.80	38.15	49.14	42.81	25.00	20.65	29.49
Control LLM	**56.00**	28.98	65.96	8.09	39.76	43.88	78.33	28.20	50.14
O-LoRA	51.00	29.66	74.60	15.16	42.61	42.95	76.62	29.06	49.54
RepAnchor-Full	53.20	**32.50**	76.59	52.32	53.65	43.97	79.62	28.57	50.72
RepAnchor-LoRA	54.20	31.31	75.28	53.15	53.49	**44.19**	**79.90**	28.53	**50.87**
Qwen3-8B									
Full	59.40	31.74	80.52	54.28	56.48	49.18	75.88	29.90	51.65
LoRA	57.60	31.79	79.91	53.64	55.74	48.72	67.84	29.78	48.78
DoRA	57.60	32.80	79.53	53.64	55.89	48.72	67.84	29.78	48.78
L2-SP	59.20	33.51	80.89	43.68	54.32	48.95	78.98	29.82	52.58
LwF	59.00	32.84	80.21	50.39	55.61	48.89	78.70	29.66	52.42
GeRe-SlimPajama	57.60	32.92	81.12	**54.60**	56.56	49.01	75.60	**30.00**	51.54
GeRe-Alpaca[2]	0.00	0.00	0.00	0.00	0.00	11.74	14.60	0.00	8.78
Control LLM	60.80	30.46	81.12	25.14	49.38	48.72	**81.43**	28.88	53.01
O-LoRA	53.00	30.35	**81.50**	42.50	51.84	48.57	76.94	29.43	51.65
RepAnchor-Full	**61.00**	33.46	79.53	53.08	56.77	48.79	80.64	28.64	52.69
RepAnchor-LoRA	59.60	**33.56**	80.44	53.97	**56.89**	**49.39**	81.38	29.01	**53.26**

Table 1: Main comparison. Left block: per-dataset task accuracy (train=test). Right block: held-out retention, averaged over the four fine-tuning runs (per-cell breakdown in Appendix A). Values are percentages; RLsum = ROUGE-Lsum $\times$ 100. **Bold**/underline = per-column best/second-best.

4B and RepAnchor-LoRA trails Control LLM by 0.05 pt at 8B), with smaller absolute margins as base capability grows. Overall the 1.7B task gap is therefore a small-scale artifact rather than a structural cost, and at 8B the retention advantage broadens into a Task-Avg lead, so neither the retention nor the task-accuracy story rests on a single scale.

4.3 Training-efficiency comparison

Table 2 reports five compute axes per method. Four points stand out. **Inference parity** holds everywhere except Control LLM ($\approx \times 1.25$): RepAnchor's frozen reference is dropped after training, LoRA-family adapters merge into the base, and GeRe's replay is training-only. **RepAnchor-LoRA** reaches retention parity with RepAnchor-Full at 0.37% trainable parameters and 9.7 GB peak GPU ($\times 2.4$ below RepAnchor-Full) at $\times 1$ inference, the strongest single data point for C2. **Control LLM pays twice:** heavier inference and lower retention than RepAnchor. **GeRe** adds $\sim 33\%$ training compute from its replay batch.

4.4 Drift signature under RepAnchor

The retention numbers of Section 4.1 raise a mechanism question: does RepAnchor improve retention because the anchor loss successfully damps per-layer drift, or for some other reason that happens to coincide with drift damping? We test the mechanism directly. For each of the four task datasets, we recompute the Type-A drift profile of Section 2 on a RepAnchor-Full-1.7B checkpoint and on the matching Full-1.7B checkpoint, using the same held-out Alpaca probe batch in both cases. Figure 3 overlays the two profiles, one panel per task.

[2] GeRe-Alpaca on Qwen3-8B collapsed across all four fine-tuning runs; we re-ran each with multiple seeds and hyperparameter settings but could not recover non-zero performance.

Method	Year	Param %	GPU (GB)	Reference setup	Extra Data	Inf. cost
Full		100%	19.3	–	–	×1
LoRA	2022	0.37%	9.1	frozen base	–	×1
DoRA	2024	0.38%	9.1	frozen base	–	×1
O-LoRA	2023	0.09%	6.6	frozen base	–	×1
L2-SP	2018	100%	21.5	frozen copy	–	×1
LwF	2017	100%	28.7	frozen copy + fwd	–	×1
GeRe-SlimPajama	2025	100%	21.7	–	SlimPajama 1k	×1
GeRe-Alpaca	2025	100%	21.7	–	Alpaca 1k	×1
Control LLM	2025	20.48%	11.8	–	–	$\sim \times 1.25$
RepAnchor-Full	(our)	100%	23.1	frozen copy + fwd	Alpaca 1k	×1
RepAnchor-LoRA	(our)	0.37%	9.7	frozen base + fwd	Alpaca 1k	×1

Table 2: Training-efficiency comparison on Qwen3-1.7B; GPU averaged across the four task datasets. **Reference setup**: extra weights kept in GPU memory at training ("+fwd" = an additional forward on the frozen reference). **Extra Data**: OOD stream beyond task data. **Inf. cost**: relative to bare backbone. Details in Appendix C.

The pattern matches what the mechanism predicts. RepAnchor drift is below Full drift at every layer, and the absolute shrinkage is largest in the late decoder layers and at the final layer, where Section 2.1.2 identified the bulk of unanchored drift. The anchor loss does its job: it is not merely correlated with better downstream retention; it suppresses per-layer hidden-state drift across the stack, with the largest absolute reductions in the layers that drift most under unanchored fine-tuning, which is what the per-layer operator with uniform weights $w_i = 1/k$ should produce (Section 2.1.3).

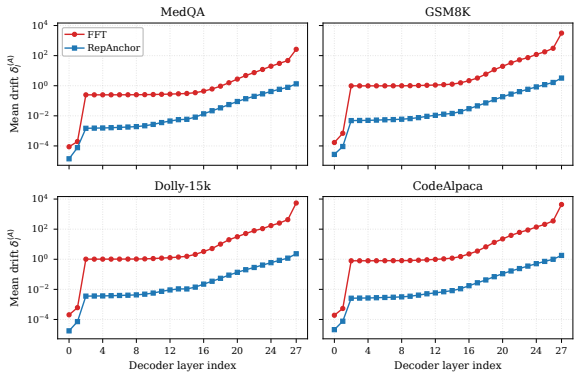

Figure 3: Per-layer Type-A drift $\delta_i^{(A)}$ (log scale): RepAnchor (blue) vs. Full (red), Qwen3-1.7B, one panel per task. Probe: 64-sample held-out Alpaca batch.

4.5 λ schedule and magnitude

Table 3 varies $\lambda_{\max}$ (0.1, 0.3, 0.5, 1.0) and schedule shape (cosine, constant, linear) for RepAnchor-Full on Qwen3-1.7B fine-tuned on GSM8K. Our configuration ($\lambda_{\max} = 0.3$, cosine) is the best GSM8K cell in the table. Schedule shape matters more than $\lambda_{\max}$: at the same peak value, the linear schedule drops 3.2 pts on GSM8K relative to cosine, while the constant schedule is within 0.2 pt of cosine; varying $\lambda_{\max}$ between 0.1 and 1.0 under cosine produces at most a 1.9-pt GSM8K spread. The composite retention average (mean of MMLU-Pro, IFEval, and CNN ROUGE-Lsum) is essentially flat

Config	GSM8K	Retention Avg
λ=0.3, cos **(our)**	**57.23**	44.40
λ=0.1, cos	55.95	43.85
λ=0.5, cos	56.48	44.88
λ=1.0, cos	55.34	**44.96**
λ=0.3, const	56.09	44.52
λ=0.3, linear	53.98	44.47

Table 3: λ ablation (RepAnchor-Full, Qwen3-1.7B). λ is $\lambda_{\max}$ for cosine/linear; "const" is fixed. Retention are MMLU-Pro, IFEval, and CNN/DM. **Bold** = best.

(range $\approx$ 1.1 pp across all six configurations), so the anchoring mechanism holds representations in place across schedule and $\lambda_{\max}$ choices.

4.6 Anchor-data sensitivity

Table 4 varies the anchor pool along two axes: quantity Alpaca subsampled to 1k, 5k, and 20k, against the full 52k corpus and qualitative source LIMA (Zhou et al., 2023a), a 1k-sample curated instruction set; OASST1 (Köpf et al., 2023), multi-turn multilingual chat filtered to English prompter–assistant pairs. Subsampling Alpaca actually helps on GSM8K: the 1k subsample we adopt as default reaches 58.24, about 5 pts above the full 52k corpus (53.30), and all three subsizes beat 52k while composite retention stays within $\approx$ 1 pp across all four Alpaca sizes. 1k samples already suffice to hold the retention line. The qualitative-source rows reinforce this: LIMA tracks Alpaca-1k within 0.5 pt on both axes, and OASST1 reaches the table's best GSM8K (58.38) while keeping retention within 1.1 pp of the Alpaca-1k peak. The mechanism generalizes across anchor distributions; the Alpaca-1k default in the headline results is not load-bearing for either task accuracy or retention.

5 Related Work

5.1 LLMs forgetting

Per-layer projection of intermediate hidden states has long served as an interpretability tool: Geva et al. (2022) read predictions through the unem-

Config	GSM8K	Retention Avg
Alpaca-1k **(our)**	58.24	**45.36**
Alpaca-5k	55.88	44.83
Alpaca-20k	54.81	44.48
Alpaca-52k	53.30	44.40
LIMA anchor	57.77	45.04
OASST1 anchor	**58.38**	44.22

Table 4: Anchor-data sensitivity (RepAnchor-Full, Qwen3-1.7B). Retention are MMLU-Pro, IFEval, and CNN/DM. **Bold** = per-column best.

bedding head at each layer, and Tuned Lens (Belrose et al., 2023) adds a learned per-layer probe. Layer-wise probing (Tenney et al., 2019) and layer-importance analyses for fine-tuning (Shi et al., 2024) likewise find depth-dependent signal. Concurrent with our work, Zheng et al. (2025) observe that forgetting is layer-localized and recoverable by freezing a small band of bottom layers. We repurpose per-layer projection as a graded *placement signal* rather than a binary freeze decision.

5.2 Anti-forgetting in fine-tuning

We group prior methods by what they regularize. *Parameter-space*: EWC's Fisher-weighted L2 (Kirkpatrick et al., 2017), L2-SP's uniform L2 (Xuhong et al., 2018), and Song et al. (2025)'s per-element importance with hand-tuned layer coefficients. *Output-distillation*: LwF (Li and Hoiem, 2017) adapts Hinton-style KD (Hinton et al., 2015); function-vector-guided training (Jiang et al., 2025) adds a probe-basis output term. *Architectural isolation*: LLaMA-Pro's interleaved expansion blocks (Wu et al., 2024). *Data replay*: GeRe (Zhang et al., 2025) replays a generic corpus under a discretized-margin loss; SSR (Huang et al., 2024) uses LLM-generated samples. *Gradient-subspace*: O-LoRA (Wang et al., 2023) restricts each task's LoRA update to a subspace orthogonal to prior tasks; MIGU (Du et al., 2024) masks gradients by activation magnitude. In every family the regularization target comes from an architectural prior, a data heuristic, or an algebraic invariant, not from a measurement of where forgetting actually concentrates.

5.3 Distillation: the operator family

Hidden-state MSE between teacher and student representations is a long-established knowledge-distillation operator. FitNets (Romero et al., 2015) introduced hint-based single-layer matching, and Patient KD (Sun et al., 2019) extended it to a strided selection of intermediate layers when compressing BERT. DistilBERT (Sanh et al., 2019) pairs output-distribution KL with a cosine term on hidden states, and TinyBERT (Jiao et al., 2020) applies MSE to hidden states, attentions, and embeddings. In class-incremental vision, PODNet (Douillard et al., 2020) matches pooled features across task boundaries, and iCaRL (Rebuffi et al., 2017) combines logit distillation with an exemplar memory.

To the best of our knowledge, no existiper-layer drift measurements to drive an anti-forgetting intervention duringgainst catastrophic forgetting in fine-tuning. To address this, we introduce a regime-decoupled framework for OOD anti-forgetting. We formulate a unified mechanism where per-layer projections (Geva et al., 2022; Belrose et al., 2023) quantify representational shift, and hidden-state distillation (Romero et al., 2015; Sanh et al., 2019; Douillard et al., 2020) enacts the corrective operation. Crucially, our reference-acquisition strategy generalizes seamlessly across both full-parameter and PEFT paradigms without altering the underlying model architecture, operator or loss.

6 Conclusion

We started from the question, *"where in the model does general-purpose knowledge degrade under task fine-tuning?"*, and showed the answer is sharply non-uniform across decoder layers and stable across probe distributions. The diagnostic motivated **RepAnchor**, a fine-tuning framework that anchors per-layer hidden-state MSE on an OOD corpus under a cosine-decayed λ. It leaves the backbone untouched, applies under both full-parameter and LoRA fine-tuning via a reference-acquisition mechanic decoupled from the regime, and is robust to the λ schedule, anchor-pool size, and anchor-corpus source. Across eight baselines spanning five anti-forgetting families, four task datasets, and three model scales, RepAnchor leads the held-out retention column-mean averages at every scale; task accuracy stays competitive at 1.7B and 4B and flips to a column-mean lead at 8B. The drift signature of RepAnchor checkpoints confirms that the mechanism behaves as designed.

Three directions remain open. First, **PEFT extensions**: RepAnchor-DoRA is a configuration-level flip that leaves the hook surface and loss unchanged. Second, **multi-modal RepAnchor**: anchoring on a general corpus to preserve LLM-side representations under vision-language fine-tuning. Third, **the diagnostic as a standalone tool**: a single forward pass per probe batch suffices to produce a per-layer drift profile for any released fine-tuned model, useful for post-hoc model selection or debugging without training-time instrumentation.

Limitations

We acknowledge four scope limitations.

Scale. Our experiments cover Qwen3-1.7B, Qwen3-4B, and Qwen3-8B. Behavior beyond 8B and on heterogeneous architectures (Llama, Mistral) is not empirically established.

Placement. Section 2 applies the per-layer anchor with uniform weights $w_i = 1/k$ across all decoder layers; Section 2.1.3 argues this is the right choice because the per-layer MSE sum already routes more loss to the layers that drift most. We do not run a placement ablation against an explicitly non-uniform allocation in this paper, e.g. a top-K band restricted to the high-drift late layers, or hand-prescribed drift-proportional layer weights $w_i \propto \delta_i^{(A)}$. Whether such an allocation can extract additional retention beyond that provided by uniform weighting, given that the operator is drift-weighted by construction, remains an open empirical question.

Anchor corpus. The headline results use Alpaca as the out-of-distribution anchor stream. Section 4.5 probes anchor-pool size and corpus choice on GSM8K with a handful of points (Alpaca subsamples around the 1k operating point; LIMA and OASST1 as alternative general-instruction corpora), and the mechanism is robust across that slice. This only covers a small portion of the design space: substantially larger or smaller anchor pools, and corpora drawn from genuinely different distributions (e.g. domain-specialized, multilingual, or code-heavy text), are not characterized, and we make no claim about how the mechanism would behave outside the explored range.

PEFT coverage. We validate full-parameter and LoRA regimes only. RepAnchor-DoRA and multimodal RepAnchor would be future tasks.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used Grammarly to check grammar, spelling, and language clarity, and used AI coding assistants (Claude) to support implementation of training and evaluation scripts. After using these tools, the authors reviewed and edited the content and the code as needed and take full responsibility for the content of the publication.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned LLM. https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. 2020. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision*, pages 86–102. Springer.

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

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 30–45.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. volume 28.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. volume 1, page 3.

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

Gangwei Jiang, Zhaoyi Li, Siqiao Xue, JUN ZHOU, Linqi Song, Defu Lian, Ying Wei, and 1 others. 2025. Unlocking the power of function vectors for characterizing and mitigating catastrophic forgetting in continual instruction tuning. In *International Conference on Learning Representations*, volume 2025, pages 47218–47243.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the association for computational linguistics: EMNLP 2020*, pages 4163–4174.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, and 1 others. 2023. Openassistant conversations: Democratizing large language model alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 47669–47681.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

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

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

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

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. *Preprint*, arXiv:1412.6550.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Shezheng Song, Hao Xu, Jun Ma, Shasha Li, Long Peng, Qian Wan, Xiaodong Liu, and Jie Yu. 2025. How to alleviate catastrophic forgetting in LLMs finetuning? Hierarchical Layer-Wise and Element-Wise regularization. *arXiv preprint arXiv:2501.13669*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. 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)*, pages 4323–4332.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4593–4601.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. volume 37, pages 95266–95290.

Haichao Wei, Yunxiang Ren, Zhoutong Fu, Aman Lunia, Yi-Lin Chen, Alice Leung, and Ya Xu. 2025. Control llm: Controlled evolution for intelligence retention in llm. *arXiv preprint arXiv:2501.10979*.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. 2024. Llama pro: Progressive llama with block expansion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6518–6537.

LI Xuhong, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International conference on machine learning*, pages 2825–2834. PMLR.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yunan Zhang, Shuoran Jiang, Mengchen Zhao, Yuefeng Li, Yang Fan, Xiangping Wu, and Qingcai Chen. 2025. Gere: Towards efficient anti-forgetting in continual learning of llm via general samples replay. *arXiv preprint arXiv:2508.04676*.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. 2025. Spurious forgetting in continual learning of language models.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021.

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

A Per-dataset retention breakdown

This appendix expands the three right-block columns of Table 1 (MMLU-Pro macro, IFEval prompt-strict, CNN/DailyMail ROUGE-Lsum) into the underlying per-(method, fine-tuning dataset, scale) cells whose row averages produced the main-text numbers. Tables 5, 6, and 7 report the full per-cell decomposition for MMLU-Pro macro, IFEval prompt-strict, and CNN/DailyMail ROUGE-Lsum respectively. All metrics are percentages (RLsum = ROUGE-Lsum $\times$ 100). **Bold** = per-column best, underline = second-best, both computed within each scale band. "–" marks task-specific eval gaps.

The per-cell numbers tell a sharper story than the column means, so we expand them here. Three patterns are visible only at this resolution. First, the RepAnchor MMLU-Pro and IFEval advantages do not rest on a single favorable cell: both variants lead or tie the leader in the majority of cells within each scale band, so the column-mean leadership in the main table is not an averaging artifact. Second, the GeRe-Alpaca collapse pattern is per-task, not uniform: it leaves GSM8K and Medical near baseline but drops Code and Dolly by 30–60 IFEval points and 5–20 ROUGE points. Because GeRe-Alpaca uses the same anchor corpus as RepAnchor, this isolates the per-layer MSE mechanism, not the corpus, as the operative difference, and only the per-cell view makes the contrast visible. Third, the CNN/DailyMail picture is bounded: at 1.7B RepAnchor leads two cells outright (GSM8K and Medical) and trails the column leader by at most 1.8 ROUGE points in the other two; at 4B the gap is at most 2.1 points, with no catastrophic failure. The retention win is broad, the documented loss is bounded, and the GeRe-Alpaca comparison cleanly isolates the mechanism.

MMLU-Pro discussion. The per-cell view shows that RepAnchor's MMLU-Pro column-mean advantage comes from broad cell coverage, not a single favorable cell. At 1.7B, RepAnchor-LoRA leads the Code, GSM8K, and Medical cells (32.98, 32.92, 32.21) and RepAnchor-Full leads Dolly (32.86); together the two variants occupy the top-two positions in six of the eight 1.7B cells. At 4B the cross-variant complementarity continues: RepAnchor-LoRA leads Code (44.85) and GSM8K (44.91), and RepAnchor-Full leads Dolly (44.75). 4B Medical is the documented exception: DoRA r=16 (43.66) and LwF (43.21) lead, with both RepAnchor variants below the cell mean. The exception is interpretable: fine-tuning on a classification-heavy task (Medical-QA) at the larger scale induces an MMLU-Pro cost that the anchor only partly recovers. At 8B RepAnchor-LoRA sweeps every per-task cell (49.48 / 49.28 / 49.25 / 49.54) and the column mean (49.39), with RepAnchor-Full trailing by 0.34–1.25 pt across cells; the 4B Medical regression does not recur. The per-scale column-mean leadership (RepAnchor-LoRA 32.77 at 1.7B, 44.19 at 4B, 49.39 at 8B; RepAnchor-Full runner-up at 32.33 / 43.97 / second-band) holds across all three scales. RepAnchor-LoRA also leads or ties RepAnchor-Full in ten of the twelve cells (the two

Method	Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical
Full	32.70	31.96	30.44	31.06	44.59	44.20	43.87	42.58	49.32	49.27	48.71	49.41
LoRA	32.87	31.13	31.71	31.27	44.18	43.94	43.93	42.98	48.17	48.50	48.84	49.37
DoRA	32.42	31.04	31.57	31.28	44.26	43.92	43.85	**43.66**	48.07	48.60	48.91	49.30
L2-SP	32.80	31.97	30.82	31.30	44.48	43.97	43.97	42.90	49.28	49.19	48.62	48.72
LwF	32.30	31.77	30.93	30.65	44.21	44.12	43.81	43.21	49.19	48.65	48.59	49.14
GeRe-SlimPajama	32.01	31.18	30.66	28.73	44.01	43.89	43.89	42.42	49.02	48.89	48.77	49.37
GeRe-Alpaca	20.03	16.29	31.37	25.80	41.73	40.95	43.93	44.63	11.74	11.74	11.74	11.74
Control LLM	32.68	32.79	32.39	30.68	44.01	44.45	44.28	42.76	48.69	48.87	48.86	48.44
O-LoRA	32.71	31.81	30.60	30.76	43.85	43.50	43.57	40.86	48.58	48.68	48.91	48.10
RepAnchor-Full	32.93	**32.86**	32.45	30.90	44.64	**44.75**	44.52	42.14	49.14	48.84	48.90	48.29
RepAnchor-LoRA	**32.98**	32.79	**32.92**	**32.21**	**44.85**	44.62	**44.91**	42.17	**49.48**	**49.28**	**49.25**	**49.54**

Table 5: MMLU-Pro testing macro accuracy (%). Column averages within each scale band are the MMLU-Pro column of Table 1. Bold and underline as defined above.

exceptions are 1.7B and 4B Dolly), which supports the regime-decoupling claim of C2.

IFEval discussion. The per-cell view sharpens two claims from Section 4.1. (i) RepAnchor's IFEval lead at the column level comes from broad cell coverage, not one outlier: RepAnchor-Full takes Dolly and Medical at 1.7B (67.84 and 67.47), GSM8K at 4B (80.41), and Medical at 8B (82.07); RepAnchor-LoRA takes Code at 8B (82.26); with underline placements in GSM8K (1.7B, 68.02), Code (4B, 79.11), Dolly (4B, 78.74), Medical (4B, 80.22), Dolly (8B, 80.04), and GSM8K (8B, 82.07). Across the twelve cells RepAnchor variants are bold or underline in nine. (ii) The GeRe-Alpaca row is the cleanest mechanism-isolation evidence in the paper: it shares RepAnchor's anchor corpus but collapses to single-digit / low-teens IFEval on Code and Dolly at 1.7B and 4B (12.57/8.13 and 12.57/5.55) while keeping partial IFEval on GSM8K and Medical at those scales, and at 8B the collapse becomes uniform at 14.60 across all four cells. The collapse pattern is task-shaped (1.7B/4B) sharpening to a corpus-wide failure (8B), but in either case the corpus identity is not the operative cause; the per-layer hidden-state anchor in RepAnchor and the discretized-margin replay loss in GeRe diverge in retention behavior despite consuming the same OOD stream. A second-order observation: full-parameter anti-forgetting methods that use a frozen-copy reference (L2-SP, LwF) recover Code and Dolly IFEval at 1.7B relative to bare Full, but they trail RepAnchor on three of the four 1.7B cells, on Dolly and GSM8K at 4B, and on Code, Dolly, and Medical at 8B.

CNN/DailyMail discussion. The per-cell view characterizes the small CNN/DailyMail regression the main text discloses. The regression is *bounded and uniform at the column level*: RepAnchor-Full sits in a 28.58–29.05 band across the four 1.7B cells, a 27.70–29.42 band at 4B, and a 28.19–29.51 band at 8B, against per-scale leaders L2-SP/DoRA (27.82–30.77 at 1.7B), DoRA/L2-SP (27.94–31.03 at 4B), and Full/L2-SP/GeRe-SlimPajama (28.46–31.35 at 8B). At 1.7B the per-cell picture is mixed: RepAnchor-Full takes the GSM8K cell (28.58) and RepAnchor-LoRA takes the Medical cell (28.95), and the gap to the leader in the other cells is at most ≈ 1.8 ROUGE-Lsum points (Dolly: 30.77 vs. 29.05). At 4B the gap to the L2-SP leader is at most ≈ 2.1 ROUGE-Lsum points (Dolly: 31.03 vs. 28.92), and more typically 0.5–1.5 points. At 8B the gap is similar in shape (Dolly: 31.35 vs. 29.76, about 1.6 pt; Code: 30.46 vs. 28.74, about 1.7 pt; smaller on GSM8K and Medical), and the column-mean difference between RepAnchor-LoRA and L2-SP is only 0.81 ROUGE-Lsum points. No cell catastrophically fails at any scale, which rules out a task-specific failure mode (e.g. RepAnchor collapsing only on long-form generation when trained on Code, or only when trained on Medical). The bounded shape is consistent with generic over-regularization: an OOD anchor stream that holds general-purpose representations rigid also slightly suppresses the stylistic adaptations that maximize ROUGE-Lsum on a long-form abstractive-summarization probe. By contrast, GeRe-Alpaca shows the opposite pattern: ROUGE-Lsum swings between 5.14 and 32.00 across cells at 1.7B and 4B, and collapses to 0.00 across every 8B cell, which confirms that RepAnchor's regression is a smooth retention-versus-generation trade-off, not training instability.

B Hyperparameters and training details

Common. Backbone Qwen3-1.7B, Qwen3-4B, or Qwen3-8B (instruction-tuned), bfloat16 mixed precision, SDPA attention, gradient checkpointing enabled, AdamW, cosine learning-rate schedule, one

Method	Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical
Full	65.06	45.10	66.54	64.88	77.63	65.62	77.63	**80.96**	77.63	64.14	80.78	80.96
LoRA	47.87	38.08	53.97	58.96	65.80	50.83	71.16	77.63	65.62	51.39	72.46	81.89
DoRA	44.73	38.63	53.05	60.44	67.28	52.87	71.16	77.08	68.39	49.72	73.01	80.22
L2-SP	**66.91**	51.02	64.88	63.96	**80.04**	71.72	78.93	79.67	79.67	73.20	**82.44**	80.59
LwF	65.62	54.16	67.47	63.59	**80.04**	74.68	79.30	80.22	80.04	71.90	81.70	81.15
GeRe-SlimPajama	63.96	49.35	56.56	60.81	78.37	67.84	79.30	77.82	76.71	66.36	79.48	79.85
GeRe-Alpaca	12.57	8.13	22.92	33.64	12.57	5.55	53.97	27.91	14.60	14.60	14.60	14.60
Control LLM	65.06	66.36	**69.50**	65.99	78.74	77.63	78.19	78.74	80.96	**81.89**	81.52	81.33
O-LoRA	62.29	46.77	64.14	64.33	77.82	73.94	76.16	78.56	77.08	70.43	78.74	81.52
RepAnchor-Full	66.17	**67.84**	68.02	**67.47**	79.11	78.74	**80.41**	80.22	80.22	80.04	80.22	**82.07**
RepAnchor-LoRA	66.36	66.91	66.54	65.80	79.11	**80.96**	79.30	80.22	**82.26**	79.85	82.07	81.33

Table 6: IFEval prompt-strict testing accuracy (%). Column averages within each scale band are the IFEval column of Table 1. Bold and underline as defined above.

Method	Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical	Code	Dolly	GSM8K	Medical
Full	29.01	30.55	27.50	27.84	29.66	30.65	28.67	28.71	**30.46**	30.94	29.19	29.02
LoRA	28.66	30.53	27.80	27.87	29.40	30.92	28.42	28.74	30.08	30.97	28.98	29.10
DoRA	**29.14**	30.13	27.34	28.19	**29.73**	30.51	27.94	28.65	30.29	31.02	28.64	**29.18**
L2-SP	28.72	**30.77**	27.82	28.01	29.67	**31.03**	28.31	28.40	30.31	**31.35**	28.93	28.67
LwF	28.65	30.55	27.75	27.83	29.35	30.59	28.21	28.04	29.46	31.10	29.12	28.96
GeRe-SlimPajama	28.28	30.76	27.65	27.75	29.63	30.74	28.72	**29.02**	30.42	31.19	**29.47**	28.93
GeRe-Alpaca	18.01	5.14	24.66	32.00	5.69	16.56	29.31	31.04	0.00	0.00	0.00	0.00
Control LLM	28.06	28.58	27.71	27.85	28.28	28.88	27.71	27.91	28.88	29.61	28.46	28.55
O-LoRA	28.79	29.76	27.72	27.98	29.59	30.01	**28.81**	27.83	29.57	30.64	28.91	28.61
RepAnchor-Full	28.79	29.05	**28.58**	28.87	27.70	29.42	28.47	28.67	28.19	29.51	28.45	28.43
RepAnchor-LoRA	28.82	28.97	28.50	**28.95**	28.22	28.92	28.66	28.31	28.74	29.76	28.96	28.57

Table 7: CNN/DailyMail testing ROUGE-Lsum (%) on the held-out summarization eval. Column averages within each scale band are the CNN column of Table 1. Bold and underline as defined above.

epoch, batch size 4, gradient accumulation 8, label smoothing 0.1.

Method-specific learning rates. 1×10^{-5} for full-parameter methods (Full, L2-SP, LwF, GeRe, Control LLM, RepAnchor-Full); 1×10^{-4} for parameter-efficient methods (LoRA, DoRA, O-LoRA, RepAnchor-LoRA).

LoRA-family details. Rank $r = 16$, $\alpha = 32$, dropout 0.05, target modules $\{q, k, v, o\}_{\text{proj}}$.

RepAnchor schedule and anchor corpus. Cosine $\lambda(t)$: $\lambda_{\max} = 0.3$, $\lambda_{\min} = 0.01$ over the full training horizon. Anchor corpus: Alpaca (1k subsample), held out from training, dataloader cycles infinitely.

Per-layer placement knob. The implementation supports stride-based placement via an `every_n_layers` parameter (anchor only every n-th decoder layer); all reported experiments use `every_n_layers=1` (every decoder layer), as motivated in Section 2.1.3. The Limitations section notes non-uniform allocations (top-K, drift-proportional weights) as open empirical questions.

Control LLM configuration. Seven expanded decoder layers, frozen base, lerp $\alpha = 0.5$ between original and expanded outputs.

GeRe configuration. Two replay-corpus settings. *GeRe-SlimPajama:* 1k SlimPajama (Soboleva et al., 2023) samples, with auto-balanced replay-loss weighting $w = \mathcal{L}_{\text{CE}}/\mathcal{L}_{\text{replay}}$ (the dy strategy), which replaces the originally published constant-weight configuration that was unstable on held-out retention. *GeRe-Alpaca:* same loss formulation and replay-weight strategy, but the replay buffer is replaced with Alpaca to match the anchor corpus used by RepAnchor, which isolates the replay-corpus effect from the loss formulation. Despite using the same Alpaca anchor corpus as RepAnchor, GeRe-Alpaca still collapses on IFEval for CodeAlpaca and Dolly fine-tuning (IFEval pStrict: 12.57/8.13 at 1.7B; 12.57/5.55 at 4B) and stays near baseline on GSM8K and Medical at those scales; at 8B the collapse becomes uniform at 14.60 across all four cells with task accuracy at zero. This corpus-matched comparison supports the claim that the per-layer MSE mechanism, not the anchor-corpus choice, is the operative difference.

Control LLM caveat. The Control LLM row in Table 1 shows structurally low MedQA accuracies (1.7B: 22.23; 4B: 8.09; 8B: 25.14). The same direction holds across all three scales and comes from a known weakness of the 7-expanded-block con-

figuration on multiple-choice classification tasks: the lerp-blended outputs suppress the sharp logit contrast that multiple-choice tasks need. This is not a configuration bug. The numbers are reported as-is.

O-LoRA comparability scope. O-LoRA (Wang et al., 2023) is a LoRA-only continual-learning method; its orthogonal-subspace constraint is defined only for the LoRA adapter update. The O-LoRA row in Table 1 is therefore only comparable to RepAnchor-LoRA, LoRA, and DoRA, not to the full-parameter anti-forgetting methods (L2-SP, LwF, GeRe, Control LLM, RepAnchor-Full).

C "Reference setup" column of Table 2

The "Reference setup" column of Table 2 captures two distinctions in one cell: what is held in GPU memory at training time, and whether the reference is queried by an extra forward pass beyond the task forward. The five cases that appear in the table are expanded below.

No reference ("–"). Full, GeRe-SlimPajama, GeRe-Alpaca, and Control LLM never maintain a frozen reference of the original weights. Control LLM's expanded architecture does freeze the original decoder layers under its lerp blend, but those frozen layers are part of the single task forward, not a separate reference object, so they fall under this category for the purpose of this column.

trainable + frozen base. LoRA, DoRA, and O-LoRA freeze the backbone weights but consume them in the same single task forward as the trainable adapter. No separate model copy is held in memory and no extra forward pass is needed; the GPU column (6.6–9.1 GB) reflects this single-model footprint.

trainable + frozen copy. L2-SP keeps a separate snapshot of the original weights in memory as a parameter-space L2 anchor, but never forwards through it. The reference contributes to a parameter-space L2 distance only, so the extra cost is GPU memory (21.5 vs. 19.3 GB for Full) but not compute.

trainable + frozen copy + extra fwd. LwF and RepAnchor-Full each hold a second frozen model copy in memory and run an additional forward pass through it at every training step. This is the most expensive setup: 28.7 GB for LwF and 23.1 GB for RepAnchor-Full, against 19.3 GB for Full.

trainable + frozen base + extra fwd. RepAnchor-LoRA obtains the reference forward by toggling the LoRA adapter off on the same PEFT-wrapped model rather than holding a second copy in memory (Section 2.2.3). The base weights are shared with the trainable adapter as in LoRA, with an additional forward through them with the adapter toggled off at each step. The GPU footprint is 9.7 GB, only 0.6 GB above bare LoRA, because the extra cost is one extra forward through activations rather than a doubled parameter footprint.