TOWARDS CONTINUAL LEARNING: MITIGATING FORGETTING WITH SUBSPACE TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) acquire new skills through fine-tuning, but it can destroy capabilities they already have: catastrophic forgetting remains a central obstacle to continual learning in LLMs. Recent works connect forgetting to how far, and along which directions, fine-tuning moves the weights. On-policy methods such as reinforcement learning and on-policy distillation avoid updating the principal singular directions of weight matrices and thus forget less. In contrast, SFT concentrates updates along these directions and is susceptible to catastrophic forgetting. In this work, we introduce Subspace Tuning (SST), which restricts each weight update to the bottom singular subspace, eliminating changes along the principal directions. SST is optimizer-agnostic and composes well with fine-tuning recipes including SFT and self-distillation. SST combines tight subspace protection, dense gradient estimation, and a simple refresh mechanism, enabling stronger learning than prior orthogonal fine-tuning methods. With these mechanisms, protecting as much as 99% weight subspace does not hinder learning on new tasks while retaining prior capabilities. This behavior strengthens with model and data size and nearly eliminates forgetting in the large-model regime. SST also keeps the model close to its base distribution, inducing 2–4$\times$ less KL drift than the baseline recipes. Overall, SST advances the learning–forgetting Pareto frontier, making it a practical default for fine-tuning.

1 INTRODUCTION

Large language models (LLMs) are useful across a rapidly growing range of applications, including assistants, coding, search, science, and finance (OpenAI, 2023; Handa et al., 2025; Chatterji et al., 2025). Many of these applications follow the same paradigm. A small number of strong base models are pre-trained once (Brown et al., 2020; Touvron et al., 2023; Yang et al., 2025), and each use case then fine-tunes one of them on specialized data (Singhal et al., 2025; Cursor Research et al., 2026; Devlin et al., 2019; Bommasani et al., 2021; Ouyang et al., 2022). Fine-tuning has thus become a standard way to specialize an LLM for a target task or domain.

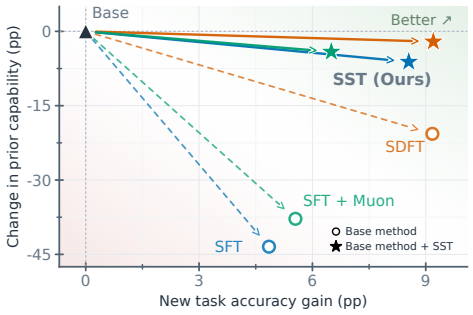


Figure 1: **SST improves learning and retention across fine-tuning recipes.** Results on Qwen3-8B are averaged over four domains (Table 4). Both axes show changes from the base model in percentage points (pp); higher is better. Circles denote training without SST, and stars denote training with SST. The black triangle marks the base model.

This paradigm comes at the cost of catastrophic forgetting. As a model learns a new task, capabilities acquired over trillions of training tokens, including instruction following, coding, reasoning, and general knowledge, can degrade severely (Luo et al., 2025; Lu & Lab, 2025; Kotha et al., 2024). This phenomenon is particularly difficult to address because the original pre-training data is typically unavailable (Kirkpatrick et al., 2017), while post-hoc repair may recover only part of what was lost (Lu & Lab, 2025; Shenfeld et al., 2026a) (Sec. 3). More broadly, forgetting stands in the way of continual learning. The next

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.

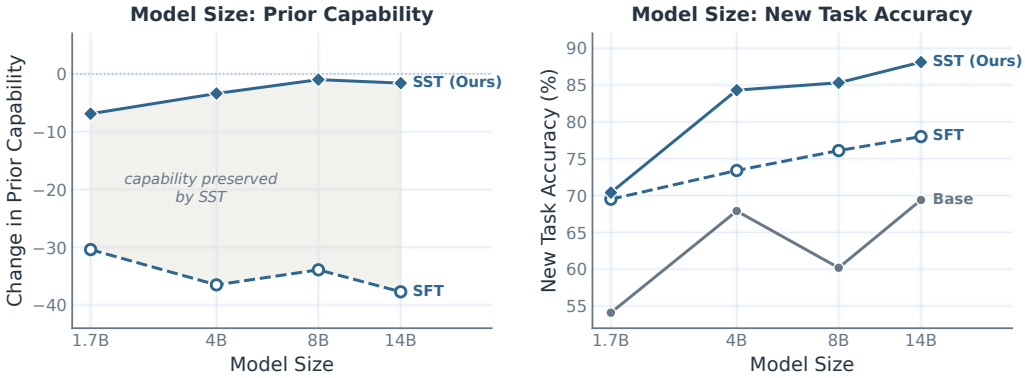


Figure 2: **Scaling model size.** Qwen3 models from 1.7B to 14B fine-tuned on Science for five epochs. SFT forgets heavily at every scale, whereas forgetting under SST shrinks as models grow and nearly vanishes from 8B onward. SST also reaches higher new-task accuracy at every size.

generation of foundation models should be able to improve over their deployed lifetime, adding new skills, domains, and corrections without sacrificing what they already do well.

Recent works provide insights into the origin of this forgetting. The leading singular directions of pretrained weight matrices have been associated with important model functionality (Martin & Mahoney, 2021; Sharma et al., 2024), while reinforcement learning has been found to fine-tune relatively small parameter subnetworks (Mukherjee et al., 2025). More directly, RLVR updates concentrate outside the principal directions and induce relatively little spectral drift compared with SFT (Zhu et al., 2025). On-policy RL also stays closer to the base policy and forgets less than SFT (Shenfeld et al., 2026b). These observations suggest a simple hypothesis. Preserving the principal singular subspaces of pretrained weights may preserve prior capabilities while leaving sufficient capacity elsewhere in weight space to learn new ones. Existing methods induce this behavior indirectly. For example, SDFT (Shenfeld et al., 2026a) trains the model on its own rewrites of the training data, so that supervision stays close to the model's output distribution—approximately on-policy, and hence expected to forget less. This helps on non-reasoning instruct models, but we find substantial forgetting remains on longer reasoning traces (Sec. 3).

We introduce *Subspace Tuning* (SST), which restricts weight updates to a non-principal subspace of the weight spectrum. Concretely, let ΔW be the optimizer's final update to a weight matrix W, and let U_k and V_k contain the top-k left and right singular vectors of W. SST applies the projected update

$$\Pi(\Delta W) \;=\; (I - U_k U_k^\top)\, \Delta W\, (I - V_k V_k^\top) \tag{1}$$

in place of ΔW. The two projectors remove from the update any component along the principal directions, on the output and input sides respectively, so every applied step stays in the non-principal subspace (Sec. 2.1). Importantly, SST constrains the final optimizer update rather than the raw gradient. Stateful optimizers such as Adam rescale the gradient coordinate-wise before it reaches the weights, so a projected gradient does not produce a projected step and part of the update returns to the directions we mean to freeze (Sec. 2). Because the projection acts on the applied update, SST is optimizer-agnostic and composes with any fine-tuning recipe, with no other change to training.

We conduct extensive empirical analysis and establish the following properties of SST (Sec. 3):

- **SST learns new tasks and retains prior capabilities.** Despite protecting the top 99% of singular directions, SST matches and often exceeds the target-task accuracy of unconstrained fine-tuning, while retaining the prior capabilities that standard fine-tuning severely degrade (Figure 1). This behavior holds across Science, Medicine, Pathology, Finance, and Tool Use, across SFT and SDFT, and with both Adam and Muon; Tables 4 and 1 provide the full per-domain results. We hypothesize that such constrained learning is possible because LLMs are heavily over-parameterized (Aghajanyan et al., 2021; Hu et al., 2022), leaving enough capacity in the non-principal subspace to fit a new task. When a fixed complement becomes restrictive, a refresh step recomputes the SVD on the current weights, realigning the protected subspace so adaptation can continue (Section 2.3).

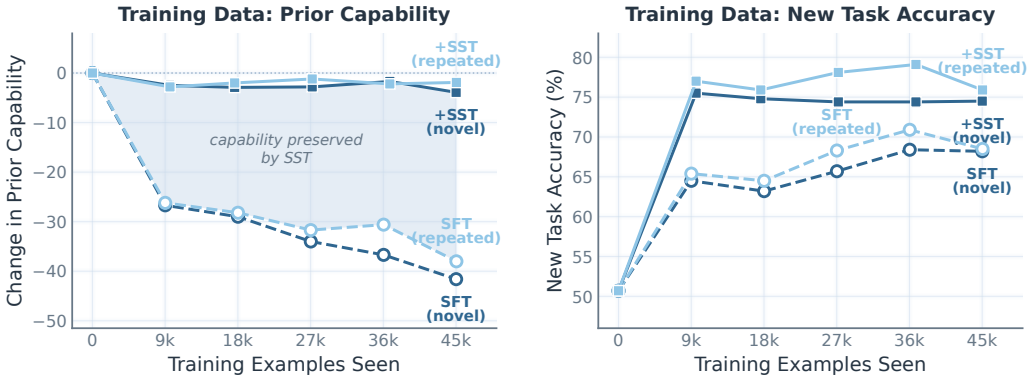


Figure 3: **Scaling training data.** Qwen3-8B fine-tuned on Science under a matched 45k-example budget: the *novel* arms see a fresh stream once, the *repeated* arms cycle 9k examples for five epochs. Under both exposures SFT drifts progressively farther from the base model, while SST stays flat and learns the task better.

- **SST scales well.** With more training data, SFT forgets progressively more while SST stays flat and still reaches higher target-task accuracy (Figure 3). With larger models, forgetting under SST shrinks even further, staying within $\sim 1\%$ of the base model from 8B onward (Figure 2).
- **SST stays close to the base model.** It induces 2-4$\times$ less KL divergence from the pretrained model than unconstrained recipes and preserves behavior that accuracy alone does not capture. A thinking model fine-tuned with SST keeps its natural reasoning length where SFT collapses it, and adding SST to SDFT reduces fabricated reference-solution attributions (Figure 8).
- **SST makes sequential fine-tuning cumulative.** Over two- and three-stage fine-tuning chains of the above domains, each skill rises when it is trained and holds its level as later stages are learned, whereas under SFT each new stage degrades the dormant ones (Figure 9,10).

Overall, this work shows that catastrophic forgetting is not an inevitable consequence of learning, but depends on where the update lands in weight space, and provides a simple, practical method for acquiring new capabilities while preserving prior ones.

2 Subspace Tuning

Across many complex systems, a small number of leading modes of a linear operator capture much of its dominant structure, while the remaining spectrum describes lower-energy variation (Berkooz et al., 1993; Schmid, 2010). Deep neural network weight matrices exhibit similarly concentrated and heavy-tailed spectra (Martin & Mahoney, 2021), and interventions on their leading singular components can disproportionately affect model performance (Sharma et al., 2024). These observations motivate preserving the principal singular directions of pretrained weights during adaptation.

We first formalize this constraint (Sec. 2.1), show that it must be enforced on the applied optimizer update rather than the gradient (Sec. 2.2), and introduce a basis refresh that trades stricter protection for continued adaptability (Sec. 2.3).

2.1 Projection Formulation

Consider a single weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, which maps an input activation x to an output Wx. Its SVD, $W = \sum_i \sigma_i u_i v_i^\top$, reads off the component of x along the input direction v_i (via $v_i^\top x$) and writes it, scaled by σ_i, onto the output direction u_i. So the right singular vectors v_i are the *input* directions the layer responds to, and the left singular vectors u_i are the *output* directions it produces. The top-k of these, $V_k = [v_1, \ldots, v_k]$ and $U_k = [u_1, \ldots, u_k]$, are what we intend to protect. Let $P_U = U_k U_k^\top$ and $P_V = V_k V_k^\top$ be the orthogonal projectors onto the protected output and input subspaces, and $Q_U = I - P_U$, $Q_V = I - P_V$ be their complements (the free subspaces). Any output vector splits as protected-plus-free, $I = P_U + Q_U$, and likewise any input, $I = P_V + Q_V$. Inserting these identities on the left and right of ΔW decomposes *any* update into four blocks, one

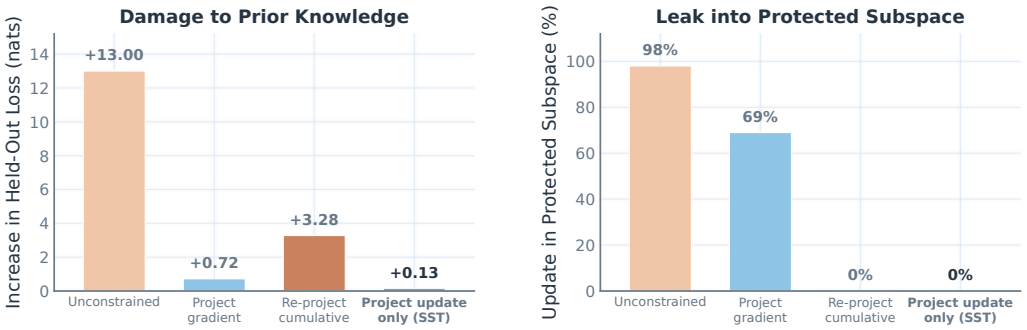


Figure 4: **Controlled study on Qwen2.5-0.5B.** Left: Increase in held-out NLL after fitting the toy task (lower is better). Right: Share of the weight change entering protected directions (lower is better). Only update projection achieves exact confinement while keeping the held-out NLL nearly unchanged.

per (input-side, output-side) combination:

$$\Delta W = \underbrace{P_U \Delta W P_V}_{\text{(1) protected}\to\text{protected}} + \underbrace{P_U \Delta W Q_V}_{\text{(2) free}\to\text{protected}} + \underbrace{Q_U \Delta W P_V}_{\text{(3) protected}\to\text{free}} + \underbrace{Q_U \Delta W Q_V}_{\text{(4) free}\to\text{free}}. \tag{2}$$

Freezing the top directions therefore means keeping only block (4) and discarding (1)-(3) from the update ΔW. Keeping only block (4) is exactly the two-sided projection of Eq. 1, which we restate:

$$\Pi(\Delta W) \;=\; Q_U\, \Delta W\, Q_V \;=\; (I - U_k U_k^\top)\, \Delta W\, (I - V_k V_k^\top). \tag{3}$$

In particular, the updated matrix $W_{\text{ref}} + \Pi(\Delta W)$ acts identically to W_{ref} on every protected input direction and produces no new component along any protected output direction: the protected singular triplets (σ_i, u_i, v_i), $i \le k$, are preserved exactly. Any of blocks (1)-(3) would break this, which is why the projection must be two-sided. We fix $k = \lfloor r \cdot \min(d_{\text{out}}, d_{\text{in}}) \rfloor$ with a rank fraction r close to 1 so only the low-energy tail is free, and compute U_k, V_k from W_{ref}, the weights at the start of training (the anchor that refresh later updates, Sec. 2.3).

2.2 Projecting the Optimizer Update

SST applies Π to the optimizer's final update ΔW. The projection could instead act on the gradient before the optimizer, either alone or together with the applied update. These choices are not equivalent. For plain SGD, whose step is a scalar multiple of the gradient, projecting the gradient and projecting the update coincide. Under Adam and other stateful optimizers they differ. Adam rescales each coordinate by its running moments, and this diagonal preconditioning does not commute with Π, so a gradient confined to the free subspace can still produce an applied update with a component in the protected subspace.

We compare three possible enforcement points (full equations in Appendix B.7):

(1) **Project the gradient**: apply Π to the raw gradient before Adam, $g \leftarrow \Pi(g)$.

(2) **Project the applied update**: let Adam update its moments from the full gradient, then apply Π to the step that Adam produces (this is what SST does).

(3) **Re-project the cumulative change**: project the gradient g and also project the applied update ΔW. This is equivalent to projecting the gradient and, after each step, re-projecting the total change from the initial anchor, $W - W_{\text{ref}}$.

To isolate this choice, we fine-tune Qwen2.5-0.5B for 60 Adam steps on a fixed eight-example next-token prediction task. We use r=0.99 and apply the constraint to the attention and MLP linear maps. Because the task is deliberately easy, every method reaches a final training loss of approximately 10^{-5}. We can therefore compare their collateral effects using two measurements: the increase in NLL on held-out generic sentences, and the fraction of the total weight change that falls outside the allowed free-to-free block.

Our results are in Figure 4. Unconstrained fine-tuning increases the held-out NLL by 13.0, with 98% of the weight change entering protected directions. Projecting the gradient reduces the NLL

increase to 0.72, but Adam reintroduces substantial leakage: 69% of the final change still leaks into the protected directions. Re-projecting the cumulative change eliminates leakage, but raises the held-out NLL by 3.28. Projecting the applied update is the only scheme that both eliminates leakage and keeps the probe nearly unchanged, with an NLL increase of just 0.13. We therefore design SST to project the final applied update. The full procedure is given in Algorithm 1 in Appendix B.7.

2.3 Refreshing the Protected Basis

The default variant computes U_k, V_k once from the pretrained weights and keeps them fixed. During longer runs, the leading singular subspaces of the current weights can rotate away from this initial basis. SST can therefore re-anchor W_{ref} to the current weights and recompute U_k, V_k from it every K optimizer steps, using a Gram-based eigendecomposition for efficiency (Appendix B.11 discusses the minimal time overhead of SST compared to SFT).

A refresh defines the free complement relative to the current weights and opens new directions for adaptation. As the singular vectors rotate, a direction protected by the initial basis can acquire a component in a later free complement. Refresh can improve continued task learning, but it mildly weakens protection relative to the original model because SST now preserves a sequence of updated principal subspaces rather than one fixed pretrained subspace. We therefore treat K as an adaptation–retention control and report no refresh, R100 (K=100), and R10 (K=10) through many of our experiments.

3 Experiments

We ask the central question: *When a model is fine-tuned on a new task, does confining the update to the non-principal subspace let it learn the task while retaining what it already knew?*

Models and tasks. We fine-tune Qwen3-8B (Yang et al., 2025) on data containing explicit `<think>...</think>` traces in four domains: Science, Finance, Pathology, and Medicine. Science uses DeepSeek-R1-distilled traces from the `open-r1/Mixture-of-Thoughts` release (Open-R1, 2025). Finance uses the reasoning-annotated FinQA/ConvFinQA release (Chen et al., 2021b; 2022). Pathology and Medicine use MedMCQA questions (Pal et al., 2022) with traces generated by Claude Sonnet 4.6 (Anthropic, 2026).

Evaluation. *In-distribution accuracy* (ID) measures how well the new task was learned. *Prior capability* measures forgetting: the average over held-out general-purpose benchmarks not used for fine-tuning. We evaluate on a broad set of benchmarks before and after training, namely GPQA-Diamond (Rein et al., 2024), MMLU-Pro (Wang et al., 2024), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021a), and IFEval (Zhou et al., 2023).

Baselines. We compare against standard supervised fine-tuning (SFT) and self-distillation fine-tuning (SDFT, Shenfeld et al., 2026a), an on-policy method designed to reduce forgetting. We report SFT and SDFT alone, and each combined with SST at rank fraction r=0.99 unless specified otherwise, optionally with basis refresh every 100 (R100) or 10 (R10) steps. We also include a Muon-optimizer variant (Bernstein & Newhouse, 2024) and the post-hoc KL-repair baseline ReInvoke (Shenfeld et al., 2026a) where relevant.

3.1 SST learns without forgetting

We begin by fine-tuning a thinking model on external chain-of-thought data from a new domain.

Unconstrained recipes collapse. Figure 6 shows broad losses in prior capability after both SFT and SDFT across the four domains. For example, as the model learns the new task, it often stops producing compilable code or respecting requested output formats. Post-hoc KL repair recovers only part of this capability and reduces in-domain gains, leaving a learning–retention trade-off rather than improving both simultaneously.

SST retains prior capability. Across domains, SST remains close to the base model while matching or exceeding the in-domain performance of the corresponding unconstrained recipe. The benefit composes with SDFT and with the Muon optimizer, indicating it generalizes across training recipes. Figure 1 summarizes the improved learning–retention trade-off with R10 refresh. Figure 6 gives the

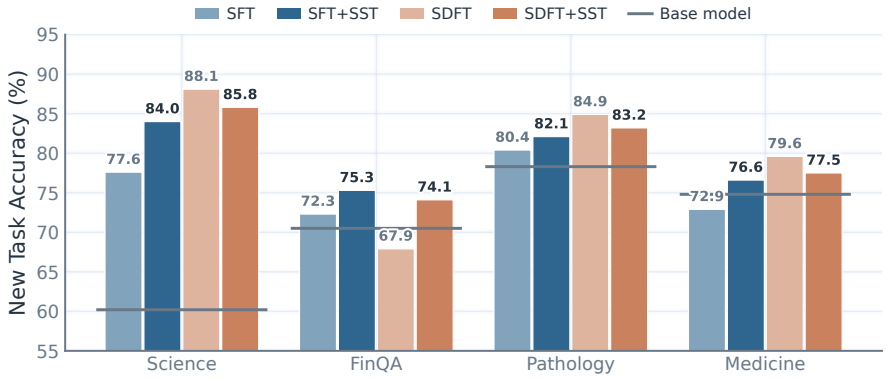


Figure 5: **Increase in in-domain accuracy** for each recipe and its +SST+R10 version (Table 4). Light bars are the recipe alone, full-color bars the +SST version, and the gray rule in each group is the base model. SST improves in-domain accuracy over SFT using the refresh mechanism.

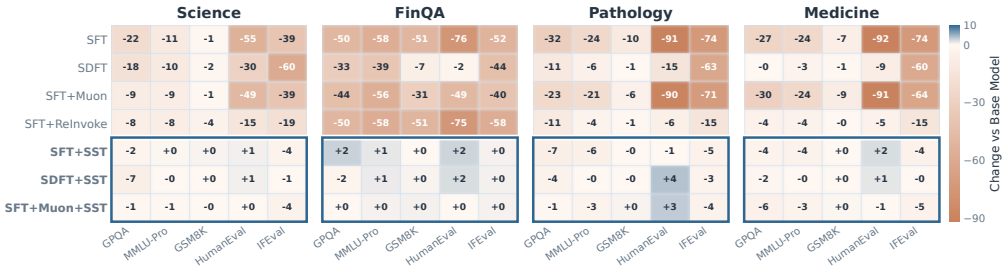


Figure 6: **Prior capability change compared to the base model** Unconstrained recipes incur broad capability losses. SST variants stay much closer to the base model across benchmarks and domains.

per-benchmark breakdown for the no-refresh variant, whose protection is strictest; switching to R10 costs only $\sim 3-5$ additional points of prior capability on average (Table 4). Figure 5 shows that with R10, retention does not come at the expense of target-task learning. More frequent refresh favors continued adaptation at a small retention cost, Appendix B.5 reports every variant.

3.2 SST SCALES WELL

SST reduces forgetting as parameters increase. Figure 2 shows the effect of scaling model size on the Science dataset, with each model trained for five epochs. Across model sizes, unconstrained SFT forgets substantially, losing 30-40 points of prior capability. By contrast, SST preserves the base model better as scale grows. SST also remains competitive with or improves upon SFT on the new task, so the retention benefit is not obtained by suppressing learning. Training details are in Appendix B.4, and a separate two-epoch sweep through 32B in Table 3 shows the same trend.

SST prevents monotonic decline in capability as the training data scales. Figure 3 shows the impact of scaling data on performance on prior and new tasks. We compare two setting under a 45,000-example budget. In the first, we have a novel stream of data and in the second we look at the multi-epoch setting: a 9k example subset repeated for five epochs. In both settings, SFT moves progressively farther from the base model as training proceeds, while SST maintains prior capability and continues improving on the target task. SST therefore remains effective in both the single-epoch novel-data setting and the multi-epoch repeated-data setting. The corresponding table can be found in Appendix B.4.

3.3 SST STAYS CLOSE TO THE BASE MODEL

SST keeps the model close to its base distribution. Figure 7 measures divergence from the base model on WikiText-2 (Merity et al., 2017), a general-domain corpus. Across the training-data (Figure 7) and model-scale studies (Figure 13, Appendix B.6), SST remains substantially closer to the base distribution than the corresponding unconstrained model. The right panel of Figure 7 traces new-

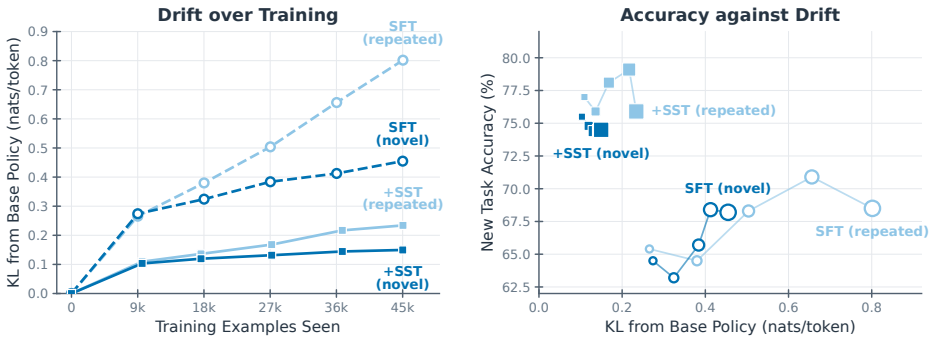


Figure 7: **Policy drift from the base model**, $\mathrm{KL}(\pi_{\text{base}} \,\|\, \pi_\theta)$ on WikiText-2, a general-domain dataset containing no task data. Left: drift over training. Right: the same drift against the in-domain accuracy, marker size growing with data seen.

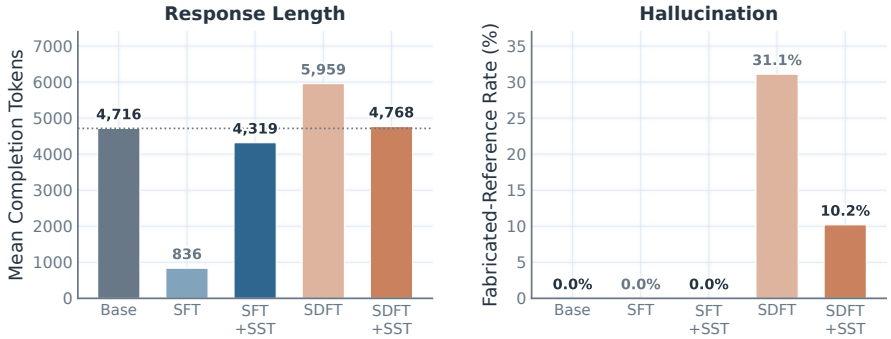


Figure 8: **Each recipe damages the base model in its own way, and SST pulls it back.** Olmo-3-7B-Think fine-tuned on answer-only medical data (+R10 arms shown); the gray bar in each panel is the base model. Left: Mean response length, with the dotted rule marking the base model's response length. Right: Rate of fabricating a "reference solution" the prompt never contained.

task accuracy against drift as training proceeds, and shows that SST improves accuracy with roughly $3\times$ less movement away from the base policy. Across model sizes and training recipes, the reduction is 2-4$\times$ (Appendix B.6).

A natural alternative is to enforce closeness directly, by adding a reverse-KL penalty toward the base model during SFT. Appendix B.9 (Table 6, Figure 15) shows that this does not recover SST's trade-off. Weak regularization fails to mitigate forgetting, and stronger regularization suppresses task learning without preserving prior capability.

SST reduces hallucinations induced by self-distillation. Finally, we stress-test a difficult case for a reasoning model: finetuning with non-reasoning data. This setting explores a challenge where supervised finetuning can substantially degrade their reasoning capabilities. We fine-tune Olmo-3-7B-Think (Team Olmo, 2025) on answer-only medical supervision from HuatuoGPT-o1 (Chen et al., 2025) and summarize the results in Figure 8. SFT improves answer accuracy but sharply shortens the model's reasoning, while SST keeps its response length close to the base model. SDFT maintains longer reasoning but can overshoot and fabricate references to a solution that was never provided. Adding SST's subspace constraint substantially reduces this behavior. Together, these results show that staying close to the base model preserves qualitative properties not captured by task accuracy alone.

3.4 SST PRESERVES GAINS ACROSS SUCCESSIVE FINE-TUNING STAGES

A stronger test is whether SST lets a model acquire several skills in sequence without each stage erasing the last. We fine-tune Qwen3-8B on two consecutive tasks ($A \to B$), tracking retention on task A and learning on task B. We also evaluate three-stage chains ($A \to B \to C$) and track each domain throughout.

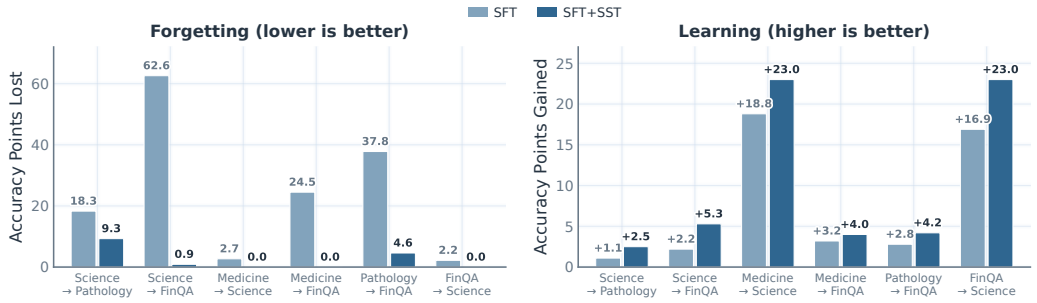


Figure 9: **Two-stage continual fine-tuning** ($A \rightarrow B$) **on Qwen3-8B.** Left: accuracy lost on domain A relative to its stage-1 level while stage 2 trains on B. Right: accuracy gained on B over the base model. SST arms use R10 refresh. SST improves *both* quantities in every tested chain.

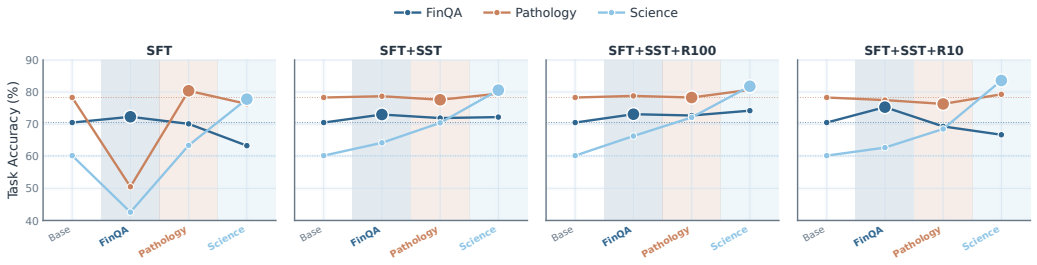


Figure 10: **Three-stage continual fine-tuning on Qwen3-8B.** The SFT family is trained on FinQA $\rightarrow$ Pathology $\rightarrow$ Science. Lines track task accuracy on all three domains, dotted lines mark each domain's base-model accuracy, each stage's background is tinted with the domain trained at that stage, and the enlarged marker is the domain just trained. A line should rise in its own stage and hold its level afterward. Under SFT the dormant domains drop sharply and recover only when trained, while the SST variants hold all three domains much closer to their trained levels.

SST learns more and forgets less. Figure 9 shows multiple $A \rightarrow B$ sequences across domain pairs. We use R10 refresh in these runs. Across all the tested combinations, SST outperforms SFT in both retention and new-task learning.

SST accumulates task knowledge. In the FinQA $\rightarrow$ Pathology $\rightarrow$ Science chain of Figure 10, each domain improves when it is trained and remains near that level as later stages are learned, so the final model retains all of the acquired skills. Under plain SFT the same chain is not cumulative. Accuracy on the untrained Pathology and Science drops by 28 and 18 points while FinQA trains, and FinQA itself gives back 9 points over the later stages. The complementary Medicine $\rightarrow$ Pathology $\rightarrow$ Science chain in Appendix B.2 shows the same pattern for the SDFT family. The refresh variants expose a useful trade-off in this sequential setting. More frequent refresh improves accuracy on the current task at a minor cost to retention on previously learned tasks.

As a complementary composition test, Appendix B.1 shows that independently trained SST experts can be merged with far less interference than SFT experts. Figure 11 reports the exact task-level changes relative to each method's own experts.

4 Related Work

Classical continual learning restricts where updates go using stored data or gradients from earlier tasks (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Farajtabar et al., 2020; Saha et al., 2021; Wang et al., 2021). These constraints require data that is unavailable at LLM scale and task boundaries that deployed fine-tuning does not have, whereas SST needs neither. A complementary line reduces forgetting through the data instead of the update. Self-distillation rewrites supervision into the model's own distribution (Shenfeld et al., 2026a), and on-policy RL forgets less because its updates stay close to the base policy (Shenfeld et al., 2026b; Zhu et al., 2025). Those methods induce a favorable update geometry through on-policy data. SST, by contrast,

constrains weight update, and as a result can compose with on-policy methods (Table 4). We defer discussion of many other related works to Appendix A.

4.1 Comparison with subspace-protecting methods

Two recent methods share our starting point of protecting the top singular directions of pretrained weights while adapting elsewhere. Both differ from SST in the same structural way. They rewrite the update as a product of trained factors and constrain the factors, whereas SST trains the weights directly and constrains the one quantity that reaches them, the applied update.

Sculpting (Nayak et al., 2026) freezes the top singular vectors of each weight matrix and projects the gradient before every AdamW step. Its projection removes only block (1) of Eq. 2, the component that reads from and writes onto protected directions at once. The cross blocks (2) and (3), which touch a protected direction on one side, are never removed. The projection moreover acts on the gradient, rather than the update as suggested by our controlled study. Even removing all three blocks from the gradient leaves 69% of the applied update in protected directions (Figure 4).

OPLoRA (Xiong & Xie, 2026) freezes the weights and trains a LoRA adapter, the product BA of two low-rank trainable matrices, sandwiched between fixed projectors, $\Delta W = Q_U\, BA\, Q_V$ in our notation. This is the projection Π of Eq. 3 applied to a low-rank product rather than to the optimizer's step, and it confines the update fully to the free subspace. The amount protected is set by the projector rank k, and their experiments protect only the top 16-128 directions while adapting at low rank in everything that remains, nearly the inverted regime from SST, which protects 99% of the spectrum and adapts densely in the tail. By the chain rule through the fixed projectors, the gradient components along protected directions never reach the trainable factors, and the applied update is projected again by construction. This corresponds to scheme (3) in Sec. 2.2, which our controlled study found mis-scales the step (Figure 4), with Adam's step sizes moreover set in factor space rather than for the weight update.

Another difference is that neither method considers refreshing the protected subspace. SST introduces refresh as part of the method, re-anchoring the basis every K steps, and studies it as an explicit retention-adaptation control - swept from no refresh to R10 throughout our experiments (Sec. 2.3).

We compare both methods on the four-domain setup under their official settings (Table 5). Both retain more capability than SFT but lose 18-26 points of prior capability on average where no-refresh SST loses about one, at comparable in-domain accuracy. Giving OPLoRA our protection level (k=$0.99d$) recovers much of the gap, yet it still trails SST on three of the four domains (Table 4). Under matched-exposure data scaling on FinQA with official settings, both OPLoRA and Sculpting forgets more as training continues, while SST stays flat (Appendix B.8, Figure 14). We therefore keep SFT and SDFT, the dominant fine-tuning recipes in practice, as the primary baselines throughout the paper, while comparing against Sculpting and OPLoRA in this main setting and on data scaling.

5 Conclusion

Catastrophic forgetting is usually treated as the price of learning new tasks. Our work studies it instead as a property of *where* the update lands, directly controls it, and mitigates forgetting. Confining the applied optimizer step to the complement of each weight matrix's top singular subspace (Eq. 3) turns recipes that destroy prior capability into ones that retain it. The same operator works across recipes (SFT, SDFT, and Muon), from 1.7B to 32B parameters, and across data scale, where SFT degrades monotonically and SST stays flat. It also turns sequential fine-tuning from destructive into cumulative and keeps the model 2–4$\times$ closer to its base distribution in KL, thus preserving behavior that accuracy alone does not capture, such as a thinking model's deliberation length and self-distillation's fabricated reference solutions.

Limitations and Future Work. Scaling SST to models with hundreds of billions of parameters and to longer task sequences is an important next test. Additionally, using information about the update for each batch to dynamically update the protection rather than refreshing at fixed intervals is a next step to make the algorithm more efficient.

Use of AI Assistants. Claude Sonnet 4.6 was used to generate the synthetic medical reasoning traces and to grade open-ended medical accuracy. AI assistants were also used to assist with language editing, plotting, and code. The authors verified the data, analyses, citations, and claims.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328. Association for Computational Linguistics, August 2021. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568/.

Anthropic. Claude Sonnet 4.6 system card. Technical report, Anthropic, February 2026. URL https://www.anthropic.com/claude-sonnet-4-6-system-card. Revised March 6, 2026.

Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993. doi: 10.1146/annurev.fl.25.010193.002543. URL https://doi.org/10.1146/annurev.fl.25.010193.002543.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024. URL https://arxiv.org/abs/2409.20325.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models, 2021. URL https://arxiv.org/abs/2108.07258.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Aaron Chatterji, Thomas Cunningham, David J. Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman. How people use ChatGPT. Working Paper 34255, National Bureau of Economic Research, September 2025. URL https://www.nber.org/papers/w34255.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, and Benyou Wang. Towards medical complex reasoning with LLMs through medical verifiable problems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14552–14573. Association for Computational Linguistics, July 2025. doi: 10.18653/v1/2025.findings-acl.751. URL https://aclanthology.org/2025.findings-acl.751/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec

Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021a. URL https://arxiv.org/abs/2107.03374.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711. Association for Computational Linguistics, November 2021b. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6279–6292. Association for Computational Linguistics, December 2022. doi: 10.18653/v1/2022.emnlp-main.421. URL https://aclanthology.org/2022.emnlp-main.421/.

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. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, Chen Shen, Emily Jia, Federico Cassano, Hanpeng Liu, Haoyu Chen, Henry Wildermuth, Jacob Jackson, Janet Li, Jediah Katz, Jiajun Yao, Joey Hejna, Josh Warner, Julius Vering, Kevin Frans, Lee Danilek, Less Wright, Lujing Cen, Luke Melas-Kyriazi, Michael Truell, Michiel de Jong, Naman Jain, Nate Schmidt, Nathan Wang, Niklas Muennighoff, Oleg Rybkin, Paul Loh, Phillip Kravtsov, Rishabh Yadav, Sahil Shah, Sam Kottler, Alexander M. Rush, Shengtong Zhang, Shomil Jain, Sriram Sankar, Stefan Heule, Stuart H. Sul, Sualeh Asif, Victor Rong, Wanqi Zhu, William Lin, Yuchen Wu, Yuri Volkov, Yury Zemlyanskiy, Zack Holbrook, and Zhiyuan Zhang. Composer 2 technical report, March 2026. URL https://arxiv.org/abs/2603.24477.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, June 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. SciKnowEval: Evaluating multi-level scientific knowledge of large language models, 2024. URL https://arxiv.org/abs/2406.09098.

Kunal Handa, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, Kevin K. Troy, Dario Amodei, Jared Kaplan, Jack Clark, and Deep Ganguli. Which economic tasks are performed with AI? evidence from millions of Claude conversations, 2025. URL https://arxiv.org/abs/2503.04761.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. doi: 10.1073/pnas.1611835114. URL https://doi.org/10.1073/pnas.1611835114.

Suhas Kotha, Jacob Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *International Conference on Learning Representations*, volume 2024, pp. 24110–24139, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/692ae28fda9bfbde7c01b13bf5a03395-Paper-Conference.pdf.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, October 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252. Association for Computational Linguistics, May 2022. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6467–6476. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231. URL https://doi.org/10.1109/TASLPRO.2025.3606231.

Charles H. Martin and Michael W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73, 2021. URL https://jmlr.org/papers/v22/20-410.html.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121038–121072. Curran Associates, Inc., 2024. doi: 10.52202/079017-3846. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/db36f4d603cc9e3a2a5e10b93e6428f2-Paper-Conference.pdf.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 132119–132138. Curran Associates, Inc., 2025. doi: 10.52202/085713-4399. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/bf235a1d6780afd979f2f81676f43413-Paper-Conference.pdf.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *International Conference on Learning Representations*, volume 2026, pp. 20212–20243, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/22620cccae29bbb3f18d226ea0320ff5-Paper-Conference.pdf.

Open-R1. Mixture-of-thoughts. Hugging Face dataset, May 2025. URL https://huggingface.co/datasets/open-r1/Mixture-of-Thoughts.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506. ACM, August 2020. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ZK4Cvv5Eci.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, April 2020. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217. URL https://doi.org/10.1017/S0022112010001217.

Pratyusha Sharma, Jordan Ash, and Dipendra Kumar Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *International Conference on Learning Representations*, volume 2024, pp. 17632–17651, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4c2092ec0b1370cce3fb5965ab255fae-Paper-Conference.pdf.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Lifelong Agents: Learning, Aligning, Evolving Workshop at ICLR 2026*, 2026a. URL https://iclr.cc/virtual/2026/10012530.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, volume 2026, pp. 59839–59864, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/618c95f4557c15b253fb0e6f548ea0c0-Paper-Conference.pdf.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R. Pfohl, Heather Cole-Lewis, et al. Toward expert-level medical question answering with large language models. *Nature Medicine*, 31(3):943–950, January 2025. doi: 10.1038/s41591-024-03423-7. URL https://doi.org/10.1038/s41591-024-03423-7.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases, 2023. URL https://arxiv.org/abs/2306.05301.

Team Olmo. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Rishabh Tiwari, Kusha Sareen, Lakshya A Agrawal, Joseph E. Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, fast and slow: Towards llms that adapt continually, 2026. URL https://arxiv.org/abs/2605.12484.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient llm finetuning. 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. 4823–4836. Association for Computational Linguistics, April 2025. doi: 10.18653/v1/2025.naacl-long.248. URL https://aclanthology.org/2025.naacl-long.248/.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 184–193. IEEE, 2021. doi: 10.1109/CVPR46437.2021.00025. URL https://doi.org/10.1109/CVPR46437.2021.00025.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671. Association for Computational Linguistics, December 2023. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Yifeng Xiong and Xiaohui Xie. OPLoRA: Orthogonal projection LoRA prevents catastrophic forgetting during parameter-efficient fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(40):34088–34096, March 2026. doi: 10.1609/aaai.v40i40.40703. URL https://doi.org/10.1609/aaai.v40i40.40703.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

An Yang et al. Qwen2.5 technical report, 2024a. URL https://arxiv.org/abs/2412.15115.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 71768–71791. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2292. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/83f95bb0ac5046338ea2afe3390e9f4b-Paper-Conference.pdf.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, July 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, David Z. Pan, Zhangyang Wang, Yuandong Tian, and Kai Sheng Tai. The path not taken: RLVR provably learns off the principals, 2025. URL https://arxiv.org/abs/2511.08567.

APPENDIX CONTENTS

A Extended Related Work **17**

B Additional Experiments **17**

B.1 Model Merging 17
B.2 Three-stage continual learning with SDFT 18
B.3 Instruct tuning without forgetting 18
B.4 Scaling Experiments 20
B.5 Qwen3-8B thinking: full result tables 20
B.6 Policy drift across model sizes 20
B.7 Toy study: full specification 20
B.8 Comparison with subspace-constrained baselines 25
B.9 KL regularisation does not prevent forgetting 26
B.10 Rank ablation on FinQA 26
B.11 Computational Cost 27
B.12 Training and evaluation details 28

A Extended Related Work

Classical continual-learning approaches often restrict the update direction: regularization penalizes movement along Fisher-important directions (Kirkpatrick et al., 2017), while projected-gradient methods enforce constraints by replacing gradient g with a projection $\Pi(g)$ built from stored data, past-task gradients (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Farajtabar et al., 2020), activation subspaces (Saha et al., 2021), the feature-covariance null space (Wang et al., 2021), or earlier tasks' adapters (Wang et al., 2023). Continual learning for LLMs differs from these settings in several ways. These methods estimate subspaces from data, but reconstructing a distribution representative of an LLM's pretraining corpus is infeasible at scale. Truly continual deployment may also lack explicit task boundaries. SST instead preserves a subspace derived from the weight spectrum (Eq. 3), requiring neither stored examples nor task boundaries.

Several parameter-efficient fine-tuning methods also construct adapters from singular directions. PiSSA (Meng et al., 2024) initializes and trains LoRA factors from the principal singular components, prioritizing high-energy directions rather than protecting them. MiLoRA (Wang et al., 2025) is closer in motivation: it retains the principal components and adapts minor singular components, but represents the update through a fixed low-rank factorization. CorDA (Yang et al., 2024b) first forms a context-oriented decomposition using input-activation covariance and initializes low-rank adapters from task- or knowledge-oriented components. In all three methods, the spectral choice is encoded in a parameter-efficient factorization derived at initialization. SST instead projects the dense update actually proposed by the optimizer into the two-sided non-principal complement. It therefore constrains the realized weight change after stateful optimizer preconditioning, does not rely on a trainable adapter factorization or task-specific covariance estimate, and can refresh its protected basis during training.

A complementary line leaves the optimizer alone and reduces the distribution shift driving forgetting: self-distillation rewrites the supervision into the model's own output distribution (Shenfeld et al., 2026a), and online RL forgets less because its updates stay KL-close to the base policy (Shenfeld et al., 2026b; Tiwari et al., 2026), both consistent with forgetting tracking distributional movement (Kotha et al., 2024; Luo et al., 2025). These methods act on the data, whereas SST acts on the update, and the two compose (Table 4).

B Additional Experiments

This appendix contains additional results supporting the main text. In particular, we report extra results on model merging independently fine-tuned experts, the Qwen2.5 instruct-model, complete model- and data-scaling results, full Qwen3 tables, policy drift across model sizes, the toy study and algorithm specification, comparisons with Sculpting and OPLoRA, reverse-KL and protected-rank controls, and full training and evaluation details.

B.1 Model Merging

If a constrained update stays in the low-energy complement of the weight spectrum, then two updates trained independently should interfere less when added back together. We test this with model merging experiments. In particular, we fine-tune a separate single-domain expert for Science, Medicine, and FinQA, then form $W_{\text{merged}} = W_{\text{base}} + \sum_d (W_d - W_{\text{base}})$ over all the 252 weight matrices. We copy every other parameter from the base model. No merge coefficients are tuned and no data is used to fit the combination. Each merged model is then scored in-domain against *its own* single-domain expert under identical decode settings. We report the interference caused by merging rather than the raw accuracy gap, which already favors SST.

SST experts compose, SFT experts collide. Figure 11 shows that independently trained SFT experts interfere strongly when their task vectors are combined, and the interference compounds as more experts are merged. SST experts compose much more cleanly: pairwise merges remain close to their corresponding single-domain experts, and the three-expert merge degrades substantially more slowly. Each SST expert modifies only the low-energy subspace, so adding their updates leaves the protected principal directions unchanged. Unconstrained experts can each alter these directions, causing their changes to accumulate during merging.

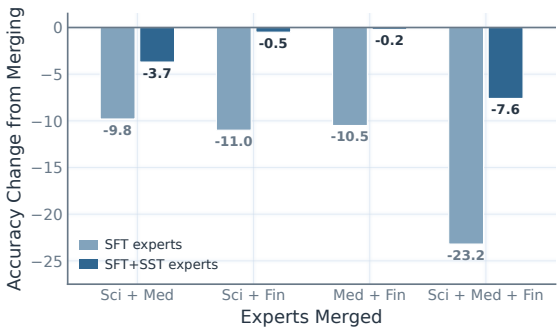


Figure 11: Interference from task-arithmetic merging. Each bar is the mean change in task accuracy relative to the same method's own single-domain experts.

B.2 Three-stage continual learning with SDFT

Figure 12 shows the Medicine → Pathology → Science continual learning chain discussed in the main text on SDFT. Across the three stages, plain SDFT consistently degrades performance on Medicine and Science. SST stabilizes these trajectories, especially the R10 variant accumulates learning across all three domains.

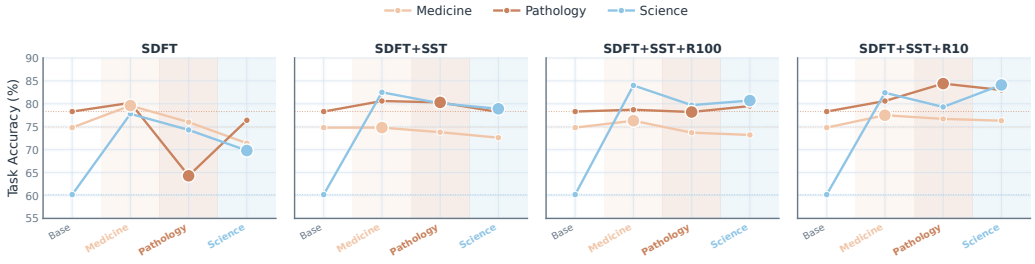


Figure 12: **Three-stage SDFT chain on Qwen3-8B.** The SDFT family is trained on Medicine → Pathology → Science. Lines track all three domains, dotted lines mark base-model accuracy, and the background tint identifies the domain trained at each stage.

B.3 Instruct tuning without forgetting

We also study a non-thinking instruction model with shorter sequences, where forgetting is less severe. Table 1 reports Qwen2.5-7B-Instruct (Yang et al., 2024a) on ToolAlpaca (Tang et al., 2023), HuatuoGPT-o1 (Chen et al., 2025), and SciKnowEval (Feng et al., 2024), matching the SDFT setup (Shenfeld et al., 2026a). All cells use rank fraction r=0.99 except Science, which uses r=0.90. On this model and domain, the tail left free at r=0.99 was too small to fit the task. This is the only setting in the paper that required a looser protected rank.

SFT learns the task but forgets. Table 1 shows that standard fine-tuning improves the target task but reduces prior capability, with the clearest losses in instruction following and knowledge. SDFT moderates this effect in some settings but is not consistently protective across domains.

SST holds capability at matched fit. Adding SST recovers much of the lost prior capability on Tool Use and Medical while remaining competitive on in-domain accuracy. Science is a narrower trade-off: SST variants can improve task learning, but no single variant dominates both learning and retention. Table 1 gives the exact per-domain and per-benchmark values. The capability average covers HellaSwag (Zellers et al., 2019), TruthfulQA-MC1 (Lin et al., 2022), MMLU (Hendrycks et al., 2021), IFEval (Zhou et al., 2023), and WinoGrande (Sakaguchi et al., 2020).

Table 1: Qwen2.5-7B-Instruct. ID = in-distribution accuracy; LM average over HellaSwag, TruthfulQA-MC1, MMLU, IFEval, and WinoGrande; Δ is relative to the pretrained baseline (68.5). Bold entries mark column maxima among non-base recipes within each domain.

Domain	Method	ID	HellaSwag	TQA-mc1	MMLU	IFEval	Winogr.	LM avg (Δ)
ToolUse	Baseline	—	80.5	47.6	71.8	71.5	71.0	68.5 (+0.0)
	SFT	70.1	**74.3**	39.5	57.7	45.7	**68.7**	57.2 (−11.3)
	SFT+SST	70.1	65.5	**47.6**	68.4	70.4	62.6	62.9 (−5.6)
	SFT+SST+R100	69.1	65.3	47.2	68.3	69.5	63.0	62.7 (−5.8)
	SFT+SST+R10	71.1	67.6	47.4	68.5	69.7	63.3	63.3 (−5.2)
	SDFT	70.1	70.8	44.2	**68.9**	65.8	67.7	63.5 (−5.0)
	SDFT+SST	66.0	66.7	43.5	68.5	71.3	64.3	62.9 (−5.6)
	SDFT+SST+R100	64.9	67.0	44.6	68.8	71.7	63.8	63.2 (−5.3)
	SDFT+SST+R10	**75.3**	68.2	44.7	68.8	**74.5**	63.5	**63.9** (−4.6)
Medical	Baseline	—	80.5	47.6	71.8	71.5	71.0	68.5 (+0.0)
	SFT	20.5	**72.3**	36.5	68.5	61.9	67.6	61.4 (−7.1)
	SFT+SST	19.5	70.8	41.4	69.6	68.8	68.3	**63.8** (−4.7)
	SFT+SST+R100	19.0	71.1	41.2	69.3	68.2	67.6	63.5 (−5.0)
	SFT+SST+R10	19.2	70.5	41.7	**70.1**	68.9	67.0	63.7 (−4.8)
	SDFT	**29.7**	67.1	39.7	55.7	58.8	**68.4**	57.9 (−10.6)
	SDFT+SST	26.9	68.0	40.5	69.0	**73.0**	68.0	63.7 (−4.8)
	SDFT+SST+R100	29.6	68.1	41.7	69.2	72.1	68.1	**63.8** (−4.7)
	SDFT+SST+R10	28.0	67.3	**41.9**	68.3	**73.0**	67.2	63.5 (−5.0)
Science	Baseline	—	80.5	47.6	71.8	71.5	71.0	68.5 (+0.0)
	SFT	68.6	**73.2**	43.0	69.2	61.4	68.2	**63.0** (−5.5)
	SFT+SST	67.7	71.3	42.5	**70.6**	63.8	66.1	62.8 (−5.7)
	SFT+SST+R100	67.5	71.4	42.4	69.7	63.6	66.8	62.8 (−5.7)
	SFT+SST+R10	69.8	**73.2**	42.2	68.2	55.3	67.2	61.2 (−7.3)
	SDFT	69.0	70.1	**43.7**	67.2	61.4	66.5	61.8 (−6.7)
	SDFT+SST	67.1	68.9	43.5	67.4	62.5	66.4	61.7 (−6.8)
	SDFT+SST+R100	**70.8**	69.3	42.8	66.1	**64.1**	68.0	62.1 (−6.4)
	SDFT+SST+R10	70.0	69.7	41.1	65.6	50.3	**69.0**	59.1 (−9.4)

B.4 Scaling Experiments

The complete scaling controls reproduce the main trend. Across both model-scale sweeps, SST preserves substantially more prior capability than unconstrained tuning while remaining competitive on the target task, and the retention advantage strengthens with model size. The matched-budget data comparison shows the same separation under both repeated and novel exposure (Figure 3 in the main text): SFT drifts steadily, whereas SST remains stable and continues learning. Tables 2 and 3 report the exact values.

Table 2: **Five-epoch model-scale results underlying Figure 2.** Science fine-tuning with the same configuration at each size. LM avg is the five-benchmark average and Δ is relative to each model's own base. Bold entries mark column maxima among non-base recipes within each model size.

Model	Method	ID	GPQA	MMLU-Pro	GSM8K	HumanEval	IFEval	LM avg	Δ
Qwen3-14B	Base	69.4	62.1	75.5	95.6	93.9	85.8	82.6	+0.0
	SFT	78.0	39.9	48.9	72.4	36.6	26.8	44.9	−37.7
	SFT+SST	86.5	**63.6**	75.4	94.5	**95.1**	79.7	**81.7**	**−0.9**
	SFT+SST+R10	**88.1**	62.6	**76.2**	**94.7**	89.6	**82.1**	81.0	−1.6
Qwen3-8B	Base	60.2	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	76.1	35.4	48.7	81.0	36.0	34.2	47.1	−33.9
	SFT+SST	**85.4**	**61.1**	72.3	94.4	91.5	**81.1**	**80.1**	**−0.9**
	SFT+SST+R10	85.3	58.6	**74.1**	**94.5**	**92.7**	80.2	80.0	−1.0
Qwen3-4B	Base	67.9	55.1	70.0	94.0	92.1	82.8	78.8	+0.0
	SFT	73.4	32.3	44.2	78.3	29.9	27.0	42.3	−36.5
	SFT+SST	80.4	51.5	67.7	**94.1**	**90.2**	72.5	75.2	−3.6
	SFT+SST+R10	**84.3**	**53.0**	**68.7**	93.9	86.0	**75.6**	**75.4**	**−3.4**
Qwen3-1.7B	Base	54.1	38.4	58.6	89.3	83.5	72.3	68.4	+0.0
	SFT	69.5	27.3	38.6	72.4	23.8	27.9	38.0	−30.4
	SFT+SST	67.1	32.3	54.3	**87.1**	67.7	**60.4**	60.4	−8.0
	SFT+SST+R10	**70.4**	**34.8**	**56.0**	86.6	**71.3**	58.6	**61.5**	**−6.9**

B.5 Qwen3-8B thinking: full result tables

The full per-domain sweep confirms the aggregate result. Across Science, Finance, Pathology, and Medicine, unconstrained recipes learn the target task but broadly degrade prior capability, while adding SST keeps the model much closer to its baseline at comparable or better in-domain accuracy. Table 4 reports the exact values for every recipe and benchmark.

B.6 Policy drift across model sizes

SST reduces policy drift at every model scale. Figure 13 reports WikiText-2 KL divergence at the final checkpoint, with each model scored against its own pretrained base. Across sizes, SST remains substantially closer to the base distribution than SFT, and the separation persists at the largest scale. The figure gives the exact drift values for each model.

B.7 Toy study: full specification

Applied-update projection is the only constraint that is both exact and stable. The controlled setup below isolates where the projection is applied. All schemes learn the toy task, but only projecting the optimizer's applied update eliminates protected-subspace leakage without substantially changing the held-out capability.

Model, optimizer, targets. Qwen2.5-0.5B, fine-tuned with plain AdamW at learning rate 1×10^{-4} and weight decay 0 (unless stated). Each scheme's projection is applied to the standard target linears—attention q, k, v, o and MLP $gate, up, down$.

Table 3: **Two-epoch model-scale sweep on Science.** Δ is relative to each model's own baseline. Cells marked — were not produced; bold entries mark column maxima among non-base recipes within each model size.

Model	Method	ID	GPQA	MMLU-Pro	GSM8K	HumanEval	IFEval	LM avg	Δ
Qwen3-32B	Baseline	68.2	64.6	73.3	95.2	94.5	85.2	82.6	+0.0
	SFT	83.6	50.5	62.4	93.6	80.5	48.2	67.0	−15.6
	SFT+SST	84.1	62.6	73.5	94.8	**97.0**	79.9	81.6	−1.0
	SFT+SST+R100	85.6	58.6	73.8	**95.2**	95.1	**81.0**	80.7	−1.9
	SFT+SST+R10	**87.3**	**65.7**	**74.3**	95.1	94.5	79.1	**81.7**	**−0.9**
Qwen3-14B	Baseline	69.4	62.1	75.5	95.6	93.9	85.8	82.6	+0.0
	SFT	82.3	50.0	64.7	93.9	64.0	54.5	65.4	−17.2
	SFT+SST	84.0	**66.2**	75.5	**94.8**	93.9	82.6	**82.6**	**+0.0**
	SFT+SST+R100	85.0	59.6	**75.9**	94.5	94.5	**84.8**	81.9	−0.7
	SFT+SST+R10	86.3	60.6	75.6	**94.8**	92.1	81.0	80.8	−1.8
	SDFT	85.2	38.9	60.2	91.0	77.4	50.6	63.6	−19.0
	SDFT+SST	83.7	53.0	71.1	94.4	**95.7**	83.7	79.6	−3.0
	SDFT+SST+R100	87.9	53.5	73.5	94.7	94.5	84.1	80.1	−2.5
	SDFT+SST+R10	**88.1**	55.1	73.9	94.4	94.5	82.8	80.1	−2.5
Qwen3-8B	Baseline	60.2	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	77.6	38.4	62.3	93.9	36.6	46.2	55.5	−25.5
	SFT+SST	80.1	57.6	73.1	**95.1**	92.1	82.1	80.0	−1.0
	SFT+SST+R100	81.7	57.6	73.1	**95.1**	92.1	82.1	80.0	−1.0
	SFT+SST+R10	84.0	**58.6**	**74.4**	94.9	92.7	80.4	80.2	−0.8
	SDFT	**88.1**	42.4	62.5	92.9	61.6	25.3	56.9	−24.1
	SDFT+SST	82.9	53.0	72.5	94.8	92.1	**85.0**	79.5	−1.5
	SDFT+SST+R100	83.4	57.1	73.1	**95.1**	93.3	84.7	**80.7**	**−0.3**
	SDFT+SST+R10	85.8	56.6	70.2	93.6	**95.7**	84.1	80.0	−1.0
Qwen3-4B	Baseline	67.9	55.1	70.0	94.0	92.1	82.8	78.8	+0.0
	SFT	76.5	38.9	61.1	92.1	24.4	42.1	51.7	−27.1
	SFT+SST	78.1	53.0	68.2	**93.8**	**93.9**	79.9	**77.8**	**−1.0**
	SFT+SST+R100	78.7	**55.1**	67.8	93.6	92.7	79.5	77.7	−1.1
	SFT+SST+R10	80.6	53.5	**68.6**	93.5	88.4	79.5	76.7	−2.1
	SDFT	81.9	34.8	51.5	84.1	58.5	42.3	54.2	−24.6
	SDFT+SST	81.1	42.4	61.7	92.4	89.6	73.8	72.0	−6.8
	SDFT+SST+R100	83.9	44.9	65.0	92.6	92.7	77.8	74.6	−4.2
	SDFT+SST+R10	**86.1**	48.5	67.9	92.7	90.9	**80.2**	76.0	−2.8
Qwen3-1.7B	Baseline	54.1	38.4	58.6	89.3	83.5	72.3	68.4	+0.0
	SFT	66.2	29.3	51.5	85.2	48.2	34.8	49.8	−18.6
	SFT+SST	65.3	31.8	**55.8**	87.6	**75.0**	63.6	62.8	−5.6
	SFT+SST+R100	65.1	33.3	55.1	**88.2**	72.6	**65.4**	**62.9**	**−5.5**
	SFT+SST+R10	63.8	**37.4**	54.3	86.8	69.5	61.0	61.8	−6.6
	SDFT	75.4	27.3	36.0	57.2	41.5	25.7	37.5	−30.9
	SDFT+SST	65.7	23.2	40.4	80.3	69.5	61.7	55.0	−13.4
	SDFT+SST+R100	72.7	27.3	45.5	82.6	72.6	64.1	58.4	−10.0
	SDFT+SST+R10	**76.1**	30.3	48.6	84.2	74.4	61.7	59.8	−8.6

Table 4: Qwen3-8B (thinking mode). Methods are grouped by recipe. Bold entries mark column maxima among non-base recipes within each domain. Δ = LM avg $-$ baseline (81.0).

Domain	Method	ID	GPQA	MMLU-Pro	GSM8K	HumanEval	IFEval	LM avg	Δ
Science	Baseline	60.2	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	77.6	38.4	62.3	93.9	36.6	46.2	55.5	−25.5
	SFT+SST	80.1	57.6	73.1	**95.1**	92.1	82.1	80.0	−1.0
	SFT+SST+R100	81.7	57.6	73.1	**95.1**	92.1	82.1	80.0	−1.0
	SFT+SST+R10	84.0	58.6	**74.4**	94.9	92.7	80.4	80.2	−0.8
	SDFT	**88.1**	42.4	62.5	92.9	61.6	25.3	57.0	−24.0
	SDFT+SST	82.9	53.0	72.5	94.8	92.1	**85.0**	79.5	−1.5
	SDFT+SST+R100	83.4	57.1	73.1	**95.1**	93.3	84.7	**80.7**	**−0.3**
	SDFT+SST+R10	85.8	56.6	70.2	93.6	**95.7**	84.1	80.0	−1.0
	SFT+Muon	80.2	51.0	64.4	93.8	42.1	46.3	59.5	−21.5
	SFT+Muon+SST	80.9	**59.1**	72.0	94.5	91.5	81.3	79.7	−1.3
	SFT+Muon+SST+R10	82.1	55.6	73.3	95.0	90.2	82.8	79.4	−1.6
	SFT+ReInvoke	68.4	52.5	64.6	91.1	76.8	66.4	70.3	−10.7
	OPLoRA (k=0.99d)	80.8	53.5	72.2	94.9	90.9	84.7	79.2	−1.8
FinQA	Baseline	70.5	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	72.3	10.1	15.5	43.9	15.9	33.1	23.7	−57.3
	SFT+SST	73.0	**62.6**	**74.1**	94.8	**93.9**	86.0	**82.3**	**+1.3**
	SFT+SST+R100	73.1	53.5	73.9	94.7	93.3	84.8	80.1	−0.9
	SFT+SST+R10	**75.3**	40.4	68.3	94.6	89.0	**86.3**	75.7	−5.3
	SDFT	67.9	27.3	34.2	88.1	89.0	41.4	56.0	−25.0
	SDFT+SST	64.1	57.6	73.9	95.1	93.3	86.1	81.2	+0.2
	SDFT+SST+R100	67.6	55.1	72.3	94.5	**93.9**	84.3	80.0	−1.0
	SDFT+SST+R10	74.1	51.0	65.5	95.2	**93.9**	80.8	77.3	−3.7
	SFT+Muon	74.9	16.2	16.9	63.7	42.1	45.8	36.9	−44.1
	SFT+Muon+SST	71.4	60.1	73.4	**95.3**	91.5	86.0	81.2	+0.2
	SFT+Muon+SST+R10	74.3	36.9	67.9	94.1	92.1	85.0	75.2	−5.8
	SFT+ReInvoke	72.2	9.6	15.3	43.7	16.5	27.4	22.5	−58.5
	OPLoRA (k=0.99d)	71.5	32.8	63.4	93.6	89.0	82.8	72.3	−8.7
Pathology	Baseline	78.3	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	80.4	28.3	48.6	84.5	0.6	11.6	34.7	−46.3
	SFT+SST	77.3	53.0	67.0	94.4	90.9	81.0	77.2	−3.8
	SFT+SST+R100	77.6	55.1	68.6	94.8	93.3	81.0	78.5	−2.5
	SFT+SST+R10	82.1	39.9	65.6	91.4	92.7	72.1	72.3	−8.7
	SDFT	**84.9**	49.0	67.5	94.2	76.2	22.9	62.0	−19.0
	SDFT+SST	82.0	56.1	72.5	94.6	**95.1**	82.6	80.2	−0.8
	SDFT+SST+R100	80.3	**59.1**	**72.8**	95.1	92.1	**83.4**	**80.5**	**−0.5**
	SDFT+SST+R10	83.2	53.0	72.0	**95.6**	93.9	82.3	79.4	−1.6
	SFT+Muon	79.3	36.9	52.2	89.1	1.8	14.2	38.8	−42.2
	SFT+Muon+SST	77.0	**59.1**	70.0	95.1	94.5	81.5	80.0	−1.0
	SFT+Muon+SST+R10	78.4	51.0	64.0	95.4	93.3	78.7	76.5	−4.5
	SFT+ReInvoke	77.6	49.5	69.5	93.9	85.4	70.2	73.7	−7.3
	OPLoRA (k=0.99d)	76.2	38.9	58.4	93.6	91.5	71.3	70.7	−10.3
Medicine	Baseline	74.8	60.1	73.0	94.8	91.5	85.6	81.0	+0.0
	SFT	72.9	33.3	49.0	87.6	0.0	11.6	36.3	−44.7
	SFT+SST	72.4	56.6	68.8	94.8	93.3	81.1	78.9	−2.1
	SFT+SST+R100	74.3	55.1	68.1	94.7	93.3	80.2	78.3	−2.7
	SFT+SST+R10	76.6	46.0	65.9	86.8	87.2	70.2	71.2	−9.8
	SDFT	**79.6**	59.6	70.1	93.9	82.9	25.1	66.3	−14.7
	SDFT+SST	74.8	58.1	**72.5**	94.8	92.7	85.2	80.7	−0.3
	SDFT+SST+R100	76.3	**60.6**	**72.5**	**95.1**	**94.5**	**85.4**	**81.6**	**+0.6**
	SDFT+SST+R10	77.5	52.5	72.1	94.1	93.9	82.3	79.0	−2.0
	SFT+Muon	71.6	29.8	49.1	86.1	0.6	21.7	37.5	−43.5
	SFT+Muon+SST	72.8	54.5	70.0	94.9	90.9	80.8	78.2	−2.8
	SFT+Muon+SST+R10	75.0	52.5	63.8	94.8	92.1	78.9	76.4	−4.6
	SFT+ReInvoke	71.6	55.6	69.4	94.4	86.6	70.6	75.3	−5.7
	OPLoRA (k=0.99d)	70.8	34.3	59.5	93.0	92.1	71.0	70.0	−11.0

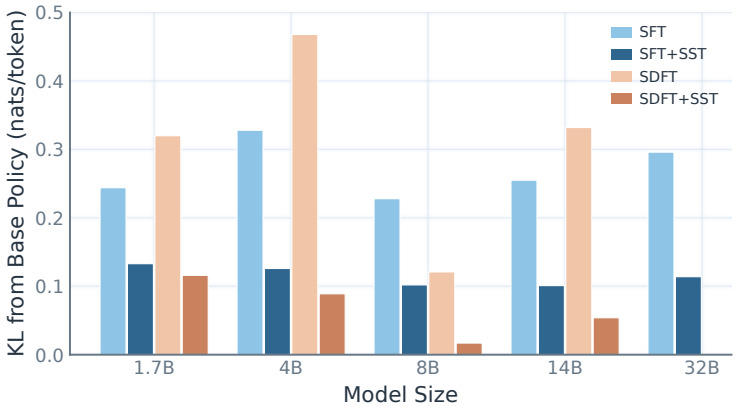


Figure 13: Policy drift at the final fine-tuned checkpoint across Qwen3 sizes. Light bars are the recipe alone, full-color bars their + SST version (no-refresh).

Task (8 fixed examples). Next-token cross-entropy on the completion tokens only (prompt tokens masked), on eight short prompt→completion pairs: `("The capital of France is", " Paris."), ("2 plus 3 equals", " 5."), ("The opposite of hot is", " cold."), ("Water is made of hydrogen and", " oxygen."), ("The first month of the year is", " January."), ("A group of wolves is called a", " pack."), ("The chemical symbol for gold is", " Au."), ("Seven times eight is", " 56.")`. The task is trivially learnable, so every scheme drives task loss to ≈0; the discriminator is the forgetting on out of domain tasks.

Capability probe (held-out). Mean per-token NLL on four generic sentences the model never trains on; **Probe**$\Delta = \mathrm{NLL}_{\text{after}} - \mathrm{NLL}_{\text{before}}$ (base-model probe NLL ≈2.81 nats/token). The probe sentences: *"The history of the Roman Empire spans over a thousand years of political change."*; *"Photosynthesis converts carbon dioxide and water into glucose using sunlight."*; *"In distributed systems, consensus algorithms ensure agreement despite failures."*; *"The novel explores themes of memory, loss, and the passage of time."*

The three enforcement schemes. With projector Π (Eq. 3), Adam moments $\hat{m}, \hat{v}$, and preconditioned raw step $\hat{m}/(\sqrt{\hat{v}}+\epsilon)$:

- **Project gradient**: $g \leftarrow \Pi(g)$; then $\hat{m}, \hat{v} \leftarrow \mathrm{Adam}(g)$; $W \leftarrow W - \mathrm{lr} \cdot \hat{m}/(\sqrt{\hat{v}}+\epsilon)$. Here $\hat{v}$ sees only free-subspace energy, and the applied step $D\,\Pi(\hat{m})$ with $D = \mathrm{diag}(1/(\sqrt{\hat{v}}+\epsilon))$ re-enters the protected space because D and Π do not commute.
- **Project the applied update**: $\hat{m}, \hat{v} \leftarrow \mathrm{Adam}(g)$ on the *full* gradient; $W_{\text{pre}} \leftarrow W$; $W \leftarrow W - \mathrm{lr} \cdot \hat{m}/(\sqrt{\hat{v}}+\epsilon)$; $W \leftarrow W_{\text{pre}} + \Pi(W - W_{\text{pre}})$. The update is confined to the low singular value subspace because projection is last, and the free step keeps the scale induced by the full gradient.
- **Re-project cumulative change**: $g \leftarrow \Pi(g)$; take the Adam step; then set $W \leftarrow W_{\text{ref}} + \Pi(W - W_{\text{ref}})$. Note that $W \leftarrow W_{\text{ref}} + \Pi(W - W_{\text{ref}})$ is equivalent to $W_t + \Pi(\delta_t)$, with δ_t the step just applied, since every previously applied change already lies in the free subspace. The update lies in the correct subspace, but the step is based on projected-gradient moments and is therefore mis-scaled.

The full procedure. Algorithm 1 details SST's implementation, using the second of these three schemes. A superscript ℓ indexes one constrained matrix, so W_t^ℓ is that matrix at step t and plays the role of W_{pre} above. The optimizer itself is never modified. It receives the full gradient, updates its moments, and proposes an ordinary step; SST then replaces the part of that step which lands on a constrained matrix with its free-to-free block. Parameters outside $\mathcal{I}$, such as embeddings and layer norms, are left to the optimizer.

Algorithm 1 Subspace Tuning (SST). The optimizer runs untouched; SST edits only the update it produces, keeping the free-to-free block of Eq. 3 and discarding the rest.

Require: Constrained matrices $\mathcal{I}$ (attention q, k, v, o and MLP gate, up, down) with pretrained weights W^ℓ of shape $d^\ell_{\text{out}} \times d^\ell_{\text{in}}$; rank fraction r, the share of singular directions held fixed; refresh interval K in optimizer steps; optimizer $\mathcal{O}$ with state s; loss $\mathcal{L}$ with minibatches $\mathcal{B}_t$; step budget T.

Setup: protect the leading singular subspace of all matrices in $\mathcal{I}$.

for all $\ell \in \mathcal{I}$ **do**
  $W^\ell_{\text{ref}} \leftarrow W^\ell$
  $k_\ell \leftarrow \lfloor r \min(d^\ell_{\text{out}}, d^\ell_{\text{in}}) \rfloor$ ▷ number of directions to protect
  $U^\ell_k, V^\ell_k \leftarrow$ top-k_ℓ left and right singular vectors of W^ℓ_{ref}
  $Q^\ell_U \leftarrow I - U^\ell_k (U^\ell_k)^\top, \quad Q^\ell_V \leftarrow I - V^\ell_k (V^\ell_k)^\top$ ▷ the free complements
end for

Training: take the ordinary step, then project the part that reaches the weights.

for $t = 0, \ldots, T-1$ **do**
  $g_t \leftarrow \nabla_W \mathcal{L}(W_t; \mathcal{B}_t)$ ▷ full gradient
  $(\widetilde{W}_{t+1}, s_{t+1}) \leftarrow \mathcal{O}(W_t, g_t, s_t)$ ▷ ordinary step
  $W_{t+1} \leftarrow \widetilde{W}_{t+1}$
  for all $\ell \in \mathcal{I}$ **do**
    $\Delta W^\ell \leftarrow \widetilde{W}^\ell_{t+1} - W^\ell_t$ ▷ the optimizer update
    $W^\ell_{t+1} \leftarrow W^\ell_t + Q^\ell_U \, \Delta W^\ell \, Q^\ell_V$ ▷ apply $\Pi(\Delta W^\ell)$ instead, Eq. 3
  end for
  if $K < \infty$ and $(t+1) \bmod K = 0$ **then**
    for all $\ell \in \mathcal{I}$ **do**
      $W^\ell_{\text{ref}} \leftarrow W^\ell_{t+1}$ ▷ re-compute projection with current weights
      $U^\ell_k, V^\ell_k \leftarrow$ top-k_ℓ singular vectors of W^ℓ_{ref}
      $Q^\ell_U \leftarrow I - U^\ell_k (U^\ell_k)^\top, \quad Q^\ell_V \leftarrow I - V^\ell_k (V^\ell_k)^\top$
    end for
  end if
end for
return W_T

B.8 Comparison with subspace-constrained baselines

The closest baselines are Sculpting (Nayak et al., 2026) and OPLoRA (Xiong & Xie, 2026). Both restrict updates to low-energy directions through a trainable factorisation; SST instead projects the applied full-parameter update. We use the official hyperparameters reported in each baseline's paper and otherwise keep the data, training budget, and evaluation harness fixed.

SST preserves more capability than factorised subspace baselines. Table 5 shows that Sculpting and OPLoRA reduce forgetting relative to SFT, but remain below no-refresh SST on prior capability across the four-domain comparison. The gap is modest on Science and becomes clearer on Finance and the medical domains. Refresh shifts SST toward stronger task learning at some cost to retention. The table reports the exact domain-level endpoints.

FinQA data scaling. We use FinQA for this comparison because the filtered task contains only about 6,000 unique reasoning traces. This makes it practical to evaluate every checkpoint for all three constrained methods, without the compute of the Science data-scaling study, which requires up to 45,000 long reasoning traces per arm.

Figure 14 compares two matched exposures: 2,000 unique examples repeated for three epochs and 6,000 unique examples seen once. Both arms see 6,000 examples, the 2k set is a prefix of the 6k set, and all methods use the same held-out 1,000 examples.

SST's advantage grows with exposure. Figure 14 shows that SST keeps prior capability nearly stable under both repeated and novel data, whereas Sculpting and OPLoRA drift progressively as training continues. The separation therefore widens with exposure rather than appearing only at the endpoint.

Retention does not suppress task learning. All three constrained methods continue learning FinQA, and their final in-domain performance remains comparable. Figure 14 gives the exact checkpoint trajectories; SFT is omitted so the constrained curves remain legible.

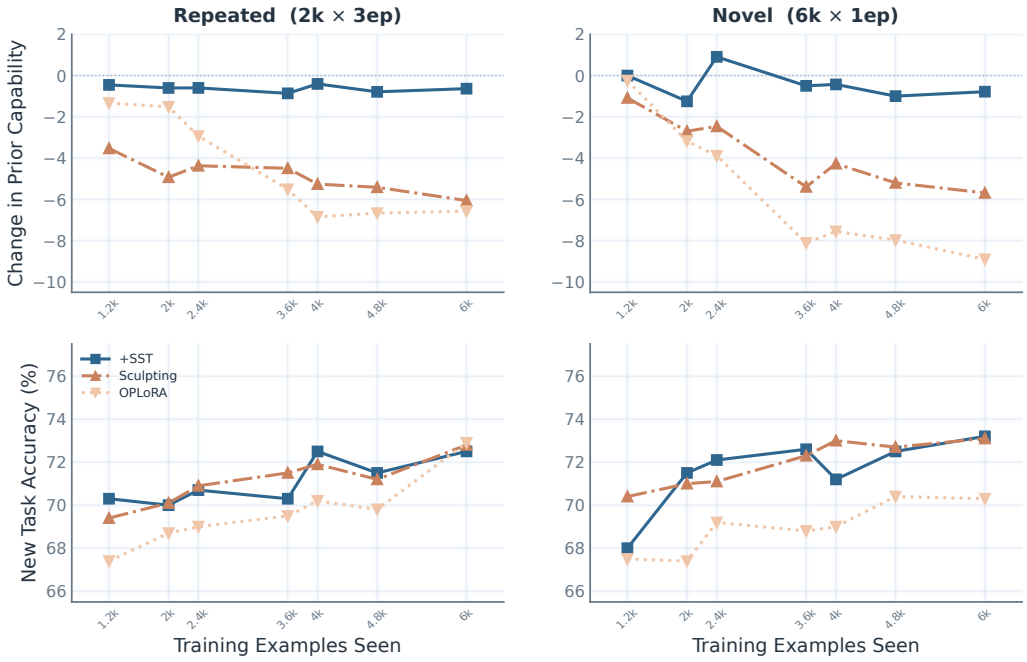


Figure 14: **FinQA data scaling against subspace-constrained baselines on Qwen3-8B.** Both columns contain 6,000 examples of training exposure: 2k unique traces repeated for three epochs (left) or 6k unique traces seen once (right). The upper panels show change in prior capability; the lower panels show FinQA accuracy. Curves are shown for no-refresh SST, Sculpting and OPLoRA. SFT is omitted to keep the constrained curves legible.

Table 5: Comparison with subspace-constrained baselines using the official method-specific settings from their papers. All methods use the same datasets and evaluation suite. ID is in-domain accuracy and LM is the five-benchmark capability average (base 81.0). Bold entries mark column maxima among non-base recipes.

	Science		FinQA		Pathology		Medicine		Mean	
Method	ID	LM	ID	LM	ID	LM	ID	LM	ID	LM
Baseline	60.2	81.0	70.5	81.0	78.3	81.0	74.8	81.0	71.0	81.0
SFT	77.6	55.5	72.3	23.7	80.4	34.7	72.9	36.3	75.8	37.6
Sculpting	83.2	77.2	74.5	57.7	79.4	43.5	74.6	41.3	77.9	54.9
OPLoRA	78.5	79.1	72.8	61.7	77.4	51.6	70.0	60.0	74.7	63.1
SFT+SST	80.1	80.0	73.0	**82.3**	77.3	77.2	72.4	**78.9**	75.7	**79.6**
SFT+SST+R100	81.7	80.0	73.1	80.1	77.6	**78.5**	74.3	78.3	76.7	79.2
SFT+SST+R10	**84.0**	**80.2**	**75.3**	75.7	**82.1**	72.3	**76.6**	71.2	**79.5**	74.9

B.9 KL REGULARISATION DOES NOT PREVENT FORGETTING

Penalising the policy does not mitigate forgetting. The natural alternative to constraining *where* an update may move is to leave the update unconstrained and instead penalise how far the resulting policy drifts. We fine-tune Qwen3-8B on FinQA and on Science with $\mathcal{L} = \mathrm{CE} + \lambda\,\mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{base}})$, a reverse KL over the response tokens against a frozen copy of the base model, at $\lambda \in \{0.001, 0.01, 0.1\}$, plus a $\lambda{=}0$ control on FinQA. Table 6 and Figure 15 show that reverse-KL regularisation does not provide a reliable learning–retention trade-off. Weak regularisation leaves forgetting essentially unchanged, while stronger regularisation can reduce target-task learning without restoring prior capability. The effect also varies across tasks, unlike the consistent retention obtained by constraining the update subspace. The table reports the exact per-task and per-benchmark values.

Table 6: **SFT with reverse-KL regularisation, Qwen3-8B.** Prior capability is the mean of the five held-out benchmarks shown. Bold entries mark column maxima among non-base recipes across the reported rows.

Task	Method	ID	GPQA	MMLU-Pro	GSM8K	HumanEval	IFEval	Prior cap.
—	Base		58.1	59.7	94.4	92.1	85.8	78.0
FinQA	$\lambda{=}0$	73.4	10.1	13.4	41.8	24.4	28.1	23.6
	$\lambda{=}0.001$	73.4	7.6	14.2	43.5	24.4	26.4	23.2
	$\lambda{=}0.01$	71.3	7.1	13.2	42.0	17.7	26.6	21.3
	$\lambda{=}0.1$	66.0	13.6	15.8	14.2	0.0	9.4	10.6
Science	$\lambda{=}0.001$	**81.7**	**49.5**	54.4	93.3	60.4	**55.1**	62.5
	$\lambda{=}0.01$	79.7	46.5	**55.0**	**93.5**	50.6	**55.1**	60.1
	$\lambda{=}0.1$	75.9	47.5	51.2	93.4	**77.4**	51.9	**64.3**

B.10 RANK ABLATION ON FINQA

The constraint maintains prior capability. FinQA is the hardest case in Section B.9: every KL coefficient leaves prior capability at or below plain SFT. We repeat that setting with SST instead, sweeping the protected rank fraction $r \in \{0.1, 0.5, 0.9\}$ with no refresh. Figure 16 reports the sweep against the same base control and the same $\lambda{=}0$ SFT run.

Rank controls retention. Figure 16 shows prior capability improving monotonically as a larger fraction of the weight subspace is protected. The held-out benchmarks move together, indicating a broad recovery rather than an average driven by one task.

Protecting more only impacts in-domain accuracy for high ranks. In-domain accuracy also improves across the sweep until $r = 0.99$, where the new task accuracy declines. Figure 16 shows the

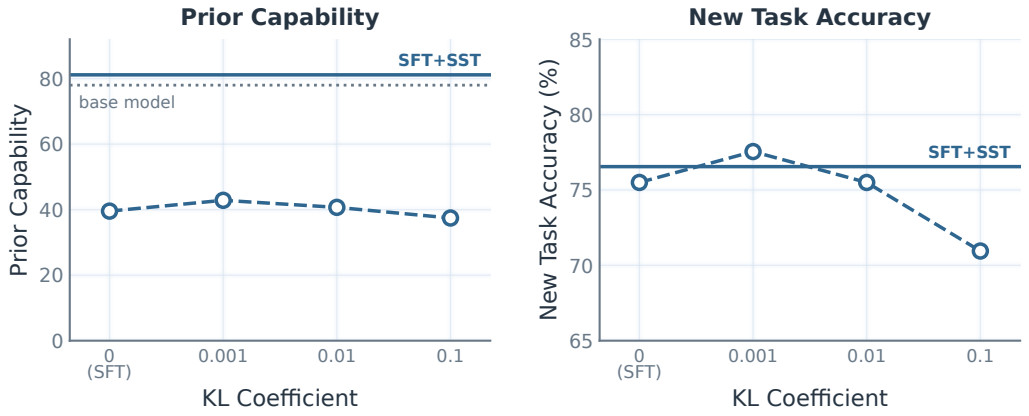


Figure 15: **Reverse-KL regularisation across three coefficients.** Prior capability (left) and in-domain accuracy (right) against the KL coefficient, averaged over FinQA and Science. The grey rule is the un-finetuned base model and the solid blue rule is SST at r=0.99 on the same two tasks.

trajectory. Although a lower r permits adaptation along more directions, supervision with a dated projection matrix can disrupt general model behavior (e.g., collapse response length), ultimately hindering learning of the target distribution.

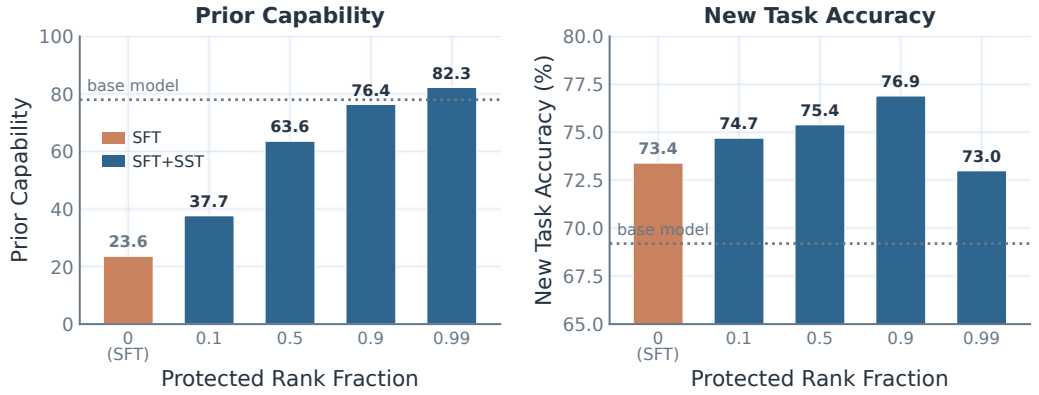


Figure 16: **Protected rank fraction on FinQA.** Prior capability (left) and in-domain accuracy (right) against the fraction of each weight's subspace held fixed, with plain SFT as the r=0 end of the same sweep. The grey rule is the un-finetuned base model. Where no KL coefficient recovers prior capability (Figure 15), rank recovers it almost completely and in-domain accuracy rises rather than falls.

B.11 Computational Cost

In this section, we consider the computation cost of subspace-constrained finetuning. SST adds no extra trainable parameters or inference-time cost. The method projects each optimizer update and optionally refreshes the protected basis. Refresh uses Gram eigendecompositions, using the smaller Gram matrix for rectangular weights. The multi-GPU training setup uses DeepSpeed ZeRO-3 (Rasley et al., 2020), and the estimates distribute the independent matrix operations across eight GPUs.

We report the measured SFT step from full-parameter bfloat16 Qwen3-8B training on eight NVIDIA L40S GPUs, with effective batch size 16, and maximum sequence length 4096. The measured SFT

step took 1.73 s. Across the model's 36 layers and 252 projected matrices, projection adds 26.2 ms (1.5%) per step in all settings. Refresh adds an amortized 62.4 ms per step for R100 and 623.5 ms for R10, giving total overheads of 5.1% and 37.6%, respectively.

B.12 Training and evaluation details

Training. Unless set by an official baseline recipe in Appendix B.8, experiments use learning rate 5×10^{-5} and effective batch size 16. Training is full-parameter except for the factored Sculpting and OPLoRA baselines, and uses DeepSpeed ZeRO-3 (Rasley et al., 2020).

Datasets and synthetic traces. The Qwen2.5 instruct study uses ToolAlpaca (Tang et al., 2023), HuatuoGPT-o1 (Chen et al., 2025), and SciKnowEval (Feng et al., 2024). Qwen3 Pathology and Medicine use MedMCQA questions and answer labels (Pal et al., 2022). For each question, the generator receives the answer options followed by `Choose the correct answer (A, B, C, or D) and explain your reasoning.` We use the Bedrock model identifier `us.anthropic.claude-sonnet-4-6`; its response is serialized inside `<think>...</think>` with a boxed answer. The archived sets contain 8,996 Medicine traces and 8,999 Pathology traces, all with a nonempty thinking block and boxed answer.

Evaluation. Evaluation uses vLLM (Kwon et al., 2023). Prior capability is the average of GPQA-Diamond, MMLU-Pro, GSM8K, HumanEval, and IFEval; in-domain accuracy uses the task's native answer extractor.

For the Olmo-3 medical study, open-ended accuracy is graded with the Bedrock model identifier `us.anthropic.claude-sonnet-4-6`. The judge receives the question, reference answer, and full model response and returns `CORRECT` or `INCORRECT`. The fabrication metric is a deterministic lower bound that flags explicit claims of an absent source, including "reference solution," "provided answer," and "based on the feedback." None of the 1,000 evaluation prompts contains a trigger phrase.