FINE-TUNING AT WHAT COST? LEARNING-FORGETTING TRADE-OFFS ACROSS TRAINING ALGORITHMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning Large Language Models (LLMs) on specialized tasks often comes at the cost of degraded performance on general capabilities—a phenomenon known as catastrophic forgetting. While numerous fine-tuning methods have been proposed, systematic comparisons of their learning-forgetting trade-offs remain limited. In this work, we present *the first comprehensive empirical study* comparing various fine-tuning methods spanning supervised fine-tuning (SFT) variants, reinforcement learning (RL) variants, and on-policy distillation (OPD) variants. We evaluate learning gains on target tasks against forgetting measured on a diverse benchmark suite. Using *Pareto frontier* analysis, we characterize the achievable trade-offs for each method across learning rates and training durations. Our results reveal that no single algorithm universally dominates the learning-forgetting Pareto front, and the trend varies across models and datasets. What holds across different algorithms is that learning rates affect both learning and forgetting, while training steps primarily affect learning but not forgetting after the initial steps. Through this study, we suggest that algorithm and hyperparameter selection interact with model architecture and target dataset in ways that preclude universal recommendations, and that the learning-forgetting trade-off is better understood through Pareto front analysis than single-point comparisons.

1 INTRODUCTION

Fine-tuning pretrained large language models (LLMs) on domain-specific data is a standard approach for adapting general-purpose models to downstream tasks. However, fine-tuning carries a well-documented risk: as the model improves on the target distribution, it may catastrophically forget capabilities acquired during pretraining (Shenfeld et al., 2026b; Kreutzer et al., 2017). This learning-forgetting trade-off is a central concern in practice, where practitioners must balance task-specific performance against the preservation of broad capabilities such as instruction following, mathematical reasoning, and safety alignment. Moreover, the ability to learn without catastrophic forgetting is foundational to enabling continual learning.

A growing body of work has demonstrated that reinforcement learning (RL)-based fine-tuning leads to less forgetting compared to standard supervised fine-tuning (SFT) (Chen et al., 2025; Shenfeld et al., 2026b; Lai et al., 2026; Zhang et al., 2026). A common hypothesis is that RL mitigates forgetting because its training examples are sampled on-policy from the model, whereas SFT relies on off-policy expert demonstrations. This motivates prior work (e.g., Wu et al. (2026)) to modify the SFT objective to exhibit learning dynamics similar to RL, while maintaining its advantage in efficiency. Another line of work shows that On-Policy Distillation (OPD) — a method that sits in between SFT and RL, which is both on-policy and fine-grained in training signals — also forgets less than SFT (Shenfeld et al., 2026a) and can be used to recover the forgotten capabilities (Lu & Thinking Machines Lab, 2025). In this study, we conduct a comprehensive examination of ten fine-tuning algorithms, spanning SFT, RL, OPD and their variants.

Our work surpasses prior work in both coverage and methodology. (1) For coverage, prior work on the learning–forgetting trade-offs (Chen et al., 2025; Shenfeld et al., 2026b; Lai et al., 2026; Zhang et al., 2026) only compares standard SFT against one of RL or OPD methods, while we compare

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.

different variants of SFT against both RL and OPD variants. (2) For methodology, prior work (Chen et al., 2025; Lai et al., 2026; Zhang et al., 2026) typically reports *single-point* results (e.g., they only report learning and forgetting from a single selected model checkpoint per algorithm). We argue that this perspective is limited, as the learning-forgetting trade-offs inherently depend on training choices. Subtle factors, such as learning rate or training duration, can significantly influence outcomes (e.g., smaller learning rates or shorter training durations are potentially trading off more learning for less forgetting). Therefore, in this study, instead of viewing each algorithm as a single point, we view it as a cloud of points, each corresponding to a different tradeoff. We characterize each algorithm by the *Pareto front* and *Hypervolume* of this set, representing its optimal trade-offs. We argue that this provides a more faithful view of fine-tuning dynamics, as it is invariant to where along the tradeoff curve a practitioner happens to stop. Indeed, we find that no single algorithm consistently dominates the Pareto front, and that vanilla SFT remains surprisingly competitive. (3) Finally, we are the first to systematically examine the relations between hyperparameters and learning-forgetting trade-offs across different fine-tuning algorithms.

Our contributions are summarized as follows:

1. We introduce the first *quantifiable* Pareto front-based framework to evaluate learning-forgetting trade-offs for LLM fine-tuning. It accounts for key training factors such as learning rate and training duration, providing a more complete and nuanced characterization of the learning-forgetting trade-offs.

2. We conduct the first comprehensive learning-forgetting study on various LLM fine-tuning algorithms, covering ten algorithms that span SFT, RL, OPD, and their variants.

3. We show that no single algorithm universally dominates the learning–forgetting Pareto front, and that trends vary across models and datasets.

4. As an orthogonal finding, we show that training steps mostly only affects learning past the initial steps. In contrast, learning rate significantly affects both forgetting and learning.

2 Fine-Tuning Algorithms

Fine-tuning is a useful technique both for general-purpose post-training and for domain- or task-specific adaptation of any language model q_θ. The premise of fine-tuning is the availability of some labeled data, $\mathcal{D}$, with expert (humans or stronger models) demonstrations. Typically, each data point contains two parts: (1) input x; (2) depending on the specific algorithm, an expert-drafted output y^*, a reward model r, which generates a reward that measures the quality of any output y, or a teacher model p_t. In the following, we assume the model under training is an autoregressive LLM, and we use q_θ to denote the model and θ is the model's weights.

Fine-tuning can be conducted by various training algorithms. Here we examine ten of those choices, grouped into SFT-Variants, RL-Variants, and OPD-Variants that sit in between SFT and RL. We briefly outline them below. See a more extended version in Appendix A.

SFT-Variants. Supervised fine-tuning trains the model to mimic the expert output distributions by maximizing q_θ's likelihood to generate y^*. Standard **SFT** uses Cross Entropy Loss: $\min_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\log q_\theta(y^*|x)]$, which approximates the forward KL divergence between q_θ and the expert output distributions. We additionally introduce four variants that potentially reduce forgetting: **LoRA** Hu et al. (2022), which uses the same loss but constrains weight updates to a low-rank subspace; **DFT** Wu et al. (2026), **TrSFT** Su et al. (2025), and **MSE**, which modify the SFT loss function to prevent large gradient steps when y^* is unlikely under q_θ. While some of these modifications have theoretical flaws (e.g., DFT can collapse to a deterministic next token prediction), prior work suggests that they are effective. Since y^* is curated beforehand, SFT-Variants are off-policy, but they give fine-grained, per-token training signals.

RL-Variants. Reinforcement Learning trains the model to generate outputs that maximize rewards under a reward model r: $\max_\theta \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim q_\theta(\cdot|x)}[r(x,y)]$. In our experiments, we use a verifiable reward model. $r(x,y)=1$ if y's final answer exactly matches the ground truth answer. We experiment

with two variants: **REINFORCE**[1], which is a standard and simple RL method; **GRPO** (Shao et al., 2024), which adds importance ratio clipping and KL regularization $D_{\mathrm{KL}}(q_\theta \,\|\, q_0)$. By default, RL methods sample new rollouts from the current policy before every gradient step, so they are on-policy. However, for efficiency purposes, it is common to reduce sampling frequency and take multiple gradient steps with one batch of sampled data. Thus, RL training can also be off-policy. With a binary, outcome-based reward applied to all the tokens, the training signals are coarse-grained.

OPD-Variants. On-Policy Distillation (OPD) trains the model to mimic expert's output distributions, but unlike SFT, it (1) has access to the full expert distributions from the teacher model p_t and (2) uses reverse KL (as opposed to forward KL) as the loss function: $\min_\theta \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\big(q_\theta(\cdot \mid x) \,\big\|\, p_t(\cdot \mid x)\big)\right]$. We experiment with **OPD** (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025), which uses a stronger model as p_t; **SDFT** (Shenfeld et al., 2026a), which constructs p_t by conditioning an exponential moving average (EMA) of the student parameters on the expert demonstration, i.e., $q_\theta^{\mathrm{EMA}}(\cdot|x, y^*)$. **SDFT-teacher-SFT (s.t.s.)**, which is the same as SDFT but uses forward KL loss. According to the original intended usage, we use OPD for forgetting recovery (Lu & Thinking Machines Lab, 2025) and SDFT-Variants for target learning. OPD-Variants trains q_θ on its own rollouts, but with next-token distributions from p_t. This makes it both on-policy and fine-grained in training signals.

In summary, our fully on-policy methods include SDFT, OPD, GRPO, and REINFORCE. All SFT-Variants are off-policy. SDFT-teacher-SFT is semi-off-policy because the responses are sampled from the starter model $q_0(\cdot|y^*)$ and never updated.

3 Pareto front Analysis

Learning and Forgetting. We denote the performance metric of model q_θ on task $\mathcal{T}$ as $\mathcal{A}(q_\theta, \mathcal{T})$. For each checkpoint q_θ fine-tuned from q_0 on the train split of a target task $\mathcal{T}_{target}^{train}$, we measure the amount of learning as the performance changes on a held-out test split of the target task $\mathcal{T}_{target}$:

$$\Delta L(q_\theta, q_0) = \mathcal{A}(q_\theta, \mathcal{T}_{target}) - \mathcal{A}(q_0, \mathcal{T}_{target})$$

Similarly, forgetting is the average changes on N other non-target tasks $\{\mathcal{T}_{other}^i\}_{i=1}^N$:

$$\Delta F(q_\theta, q_0) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{A}(q_\theta, \mathcal{T}_{other}^i) - \mathcal{A}(q_0, \mathcal{T}_{other}^i)\right)$$

As finetuning goes, ideally, we hope ΔL to keep increasing (i.e., the model is learning) while ΔF stays at zero, if not increases (i.e., the model is not forgetting).

Pareto front. The learning-forgetting problem optimizes two objectives simultaneously–maximizing learning and minimizing forgetting, so two checkpoints may not be directly comparable (e.g., one checkpoint learns more but also forgets more compared to another). This motivates us to use *Pareto front* (Zitzler & Thiele, 1998; 1999). In multi-objective optimization, a Pareto front is the subset of solutions where no solution outperforms any other solution due to trade-offs among the objectives. Solutions that are outperformed (or dominated) on all objectives by at least one other solution are excluded from the Pareto front.

For each training algorithm, we evaluate all the checkpoints saved throughout training (via a grid search across learning rate and training steps). Each checkpoint will become a point on the learning-forgetting plane with $(\Delta L(q_\theta), \Delta F(q_\theta))$ as its coordinates. We find and plot the Pareto front of this set of points to *visualize the optimal learning-forgetting trade-offs* of the algorithm. The final Pareto front is $\{(\Delta L(q_\theta), \Delta F(q_\theta))|q_\theta \text{ is not dominated}\}$. We add (0, 0) as a potential point on the Pareto front, because when the learning rate or training step equals zero, ΔF and ΔL will equal zero .

Uncovering the Pareto front. To uncover the entire underlying Pareto front (Zitzler et al., 2003; 2007) , we conduct a grid search over learning rate lr and training steps s (See details in Algorithm 1).

Our search follows a simple principle: Sweeping over the range of lr and s that produces checkpoints on the Pareto front. For the training steps s, we fix the lr and train the model for two epochs and

[1]We slightly abuse the term "REINFORCE". Unlike the very bare-bones REINFORCE algorithm (Williams, 1992), our implementation includes baseline and advantage normalization. See Appendix A.2.1 for the detailed descriptions.

Algorithm 1 Pareto-Front-Driven Hyperparameter Search

Require: initial model q_0; training algorithm $\mathcal{M}$; target task $\mathcal{T}_{target}$; non-target tasks $\{\mathcal{T}^i_{other}\}_{i=1}^N$; algorithm and model-specific learning rate increment scheme $\delta_{q_0\mathcal{M}}$ (See details in Appendix E.1); number of intermediate checkpoints K per run;
Ensure: Pareto front $\mathcal{P}$ over $(\Delta L, \Delta F)$

$\mathcal{C} \leftarrow \emptyset$ ▷ pool of all evaluated checkpoints
$lr \leftarrow lr_0$
loop
  $\{q_{\theta_1}, \ldots, q_{\theta_K}\} \leftarrow \textsc{Train}(q_0, \mathcal{M}, lr)$ ▷ train 2 epochs; save K equally spaced checkpoints
  $\mathcal{C}_{lr} \leftarrow \{q_{\theta_1}, \ldots, q_{\theta_K}\}$
  for $q_\theta \in \mathcal{C}_{lr}$ **do**
    compute $\big(\Delta L(q_\theta, q_0),\ \Delta F(q_\theta, q_0)\big)$
  end for
  if $\textsc{ParetoFront}(\mathcal{C} \cup \mathcal{C}_{lr}) \cap \mathcal{C}_{lr} = \emptyset$ **then**
    break ▷ turning point: this learning rate adds no Pareto point
  end if
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{lr}$
  $lr \leftarrow lr + \delta_{q_0\mathcal{M}}(lr)$
end loop
return $\mathcal{P} \leftarrow \textsc{ParetoFront}(\mathcal{C})$

evaluate on five equally spaced (including the final) checkpoints from that training run. We note that two epochs are sufficient, as all algorithms reach/almost reach their peak learning within that duration, and more training steps only lead to worse/similar ΔL and worse ΔF (i.e., they do not contribute to the Pareto front; See evidence in Appendix F.3 and Appendix F.2.6). Put in plainer words, we always train for two epochs because, for all settings, the model overfits/will soon overfit within that duration. For learning rate lr, we start with a very small value lr_0, and increase it until we reach a large enough lr where none of the checkpoints from that training run contribute to the Pareto front. The reason why we confidently stop is based on heuristics: With any lr larger than that, the model will not be able to converge on the loss landscape for the target task, and the large update in its parameters will also lead to severe forgetting, so it will not contribute to the Pareto front.

Hypervolume Indicator (HV). Hypervolume indicator (HV) is a commonly used measure to assess the performance of a Pareto front in multi-objective optimization (Zitzler & Thiele, 1998; 1999; Zitzler et al., 2003; Knowles & Corne, 2002) (See theoretical justification in Appendix B). HV is the volume dominated by the Pareto front in the objective space, bounded by a reference point that is dominated by all points of all Pareto fronts on all objectives. A higher value in HV suggests a more performant Pareto front and, in our case, a better learning-forgetting tradeoff. We use the nadir point (i.e., the vector with the lowest possible values for all objectives) $nadir = (-\mathcal{A}(q_0, \mathcal{T}^{test}_{target}), -\frac{1}{N}\sum_{i=1}^N \mathcal{A}(q_0, \mathcal{T}^i_{other}))$ as the reference point (Zitzler et al., 2007).

4 Experiment Setup

Datasets. Following Chen et al. (2025), for target task T_{target}, we experiment with simple math reasoning (**Countdown** (Chen et al., 2025; Pan et al., 2025)), and knowledge QA (**MMLU** (Hendrycks et al., 2021a)). For non-target tasks, we include benchmarks from various domains, including instruction following (**IFEval** (Zhou et al., 2023)), advanced math reasoning (**MATH** (Hendrycks et al., 2021b)), and safety compliance (**WildGuardTest** Han et al. (2024); **WildJailbreak** Jiang et al. (2024)). We also include Countdown/MMLU in the non-target tasks when it is not used as the target.

For SFT-Variants, expert demonstrations (reasoning plus answer) are sampled from a teacher model, LLaMA3.3-70B-Instruct (Grattafiori et al., 2024). Chen et al. (2025) filter out queries where the teacher model gives incorrect answers for SFT, but for RL, they still use the full set of data, which we believe is an unfair comparison. In this work, we use the same post-filtering set of queries for all algorithms. See Appendix E for dataset statistics.

Models. We experiment with **Qwen2.5-1.5B-Instruct** (Yang et al., 2024) and **LLaMA3.1-8B-Instruct** (Grattafiori et al., 2024) as q_0. We report the performance of q_0 in Table 1.

	Countdown	MMLU	IFEVal	MATH	WildGuardTest	WildJailbreak
Qwen2.5-1.5B	16.00	55.04	26.61	53.28	66.04	21.04
LLaMA3.1-8B	15.50	58.57	42.88	49.64	85.70	84.71

Table 1: Performance of the initial model.

Training. For each training algorithm, the learning rate range needs to be sufficient to capture the full Pareto front, so it differs for different algorithms. We include the full list of learning rates in Table 2-5. Other training hyperparameters are shown in Table 7.

Evaluation. We report Pass@1 for all benchmarks (See details in Appendix E.3). We use greedy decoding to sample one response per query, with a max output length of 4,096.

5 Results

5.1 Pareto front Analysis

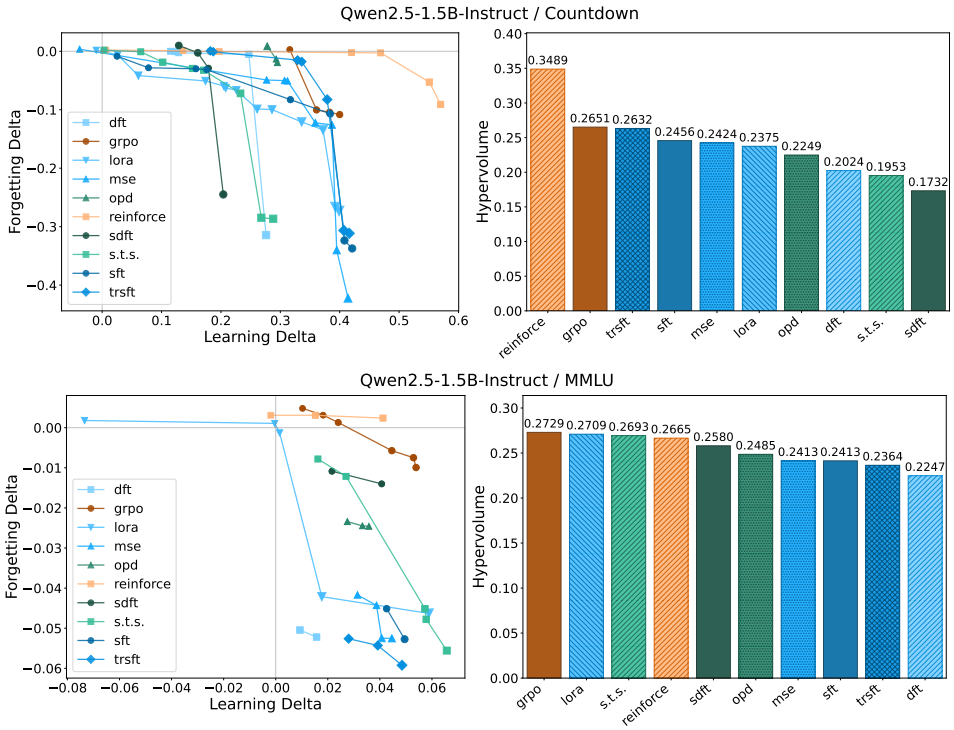


Figure 1: Pareto front Analysis on Qwen2.5-1.5B-Instruct.

We show the Pareto fronts and hypervolume scores of each model-dataset combination in Figure 1-2, and additional hypervolume ranking analysis in Appendix F.1.2. Overall, we observe that **no individual algorithm or algorithm family consistently ranks top**.

Vanilla SFT is surprisingly competitive, both within the SFT-Variants and across algorithm families. GRPO is also highly competitive, but it does not dominate other methods. DFT and SDFT consistently underperform other methods.

What is even more surprising is that when we compare different algorithm groups, **on-policy methods are not always better**. Prior work (Chen et al., 2025) claims that on-policy methods consistently outperform off-policy ones. We do the same comparison with more algorithmic variants and Pareto fronts. We choose SFT as the off-policy representative and compare it against SDFT, GRPO, and REINFORCE. OPD is excluded from the on-policy methods because it is not a fair comparison: It is used for recovery instead of learning, with a different training dataset and a hand-picked stronger teacher model. Our findings are summarized into the following three observations.

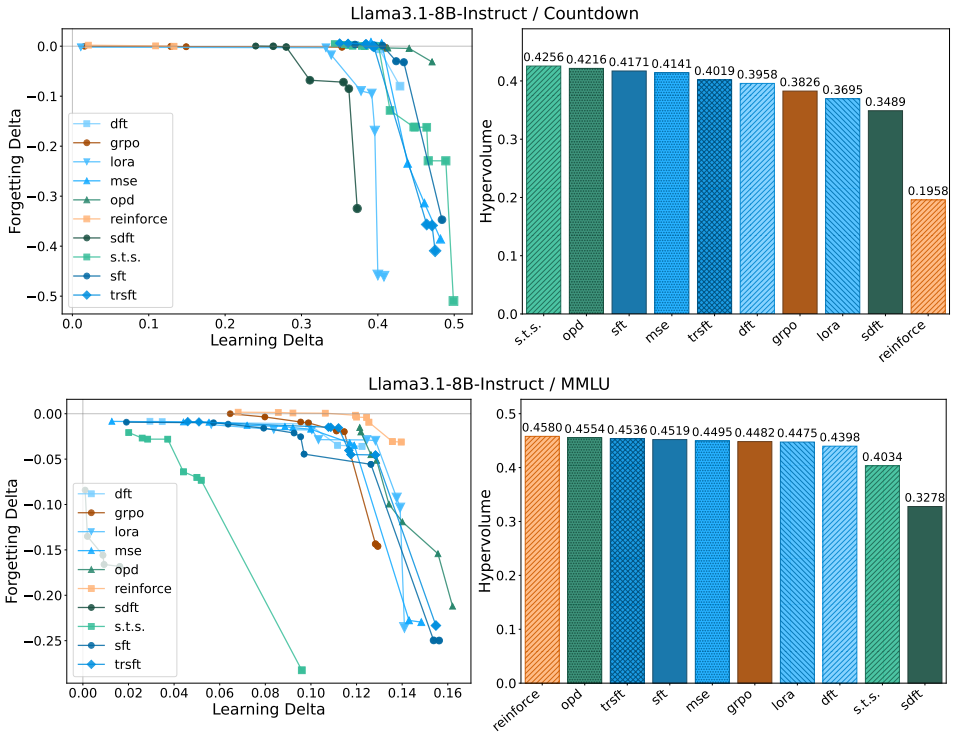


Figure 2: Pareto front Analysis on LLaMA3.1-8B-Instruct.

Observation 1: SDFT consistently underperforms SFT. From Figure 3, this is true in three out of four settings (except Qwen+MMLU). One possible reason is that the teacher model p_t is imperfect, so minimizing divergence to it can transfer the teacher's errors and biases. In the Qwen+Countdown setting, we evaluate the initial teacher model $q_0(\cdot|x, y^*)$, and find that it only achieves an accuracy of 59.3% on the Countdown train split, despite having access to y^*. This imperfect teacher hypothesis is further supported by prior work (Tiwari et al., 2026; Penaloza et al., 2026). Other speculations include the teacher's tendency to suppress the student's exploration in its rollouts (Kaur et al., 2026) and incompatible teacher-student response distributions (Li et al., 2026; Penaloza et al., 2026).

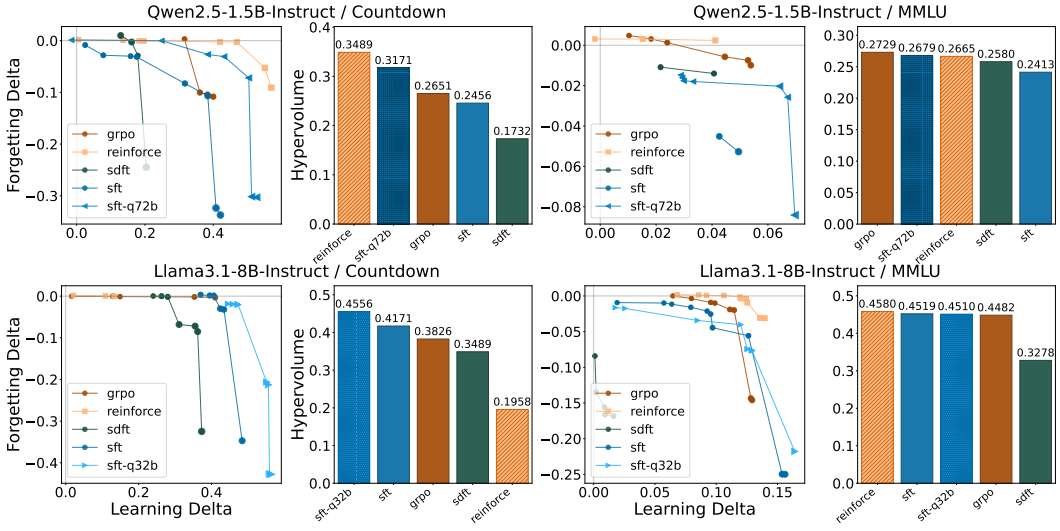


Figure 3: Comparison between On and Off policy methods.

Observation 2: RL-variants are not unilaterally better than SFT. From Figure 3, RL-variants exhibit less forgetting than SFT with the same amount of learning. However, in the two settings with LLaMA, SFT forgets not much more, and its learning cap is higher than RL, giving its Pareto front a

similar or higher HV. One possible explanation is that the SFT traces are from LLaMA3.3, which is in the same family as p_θ and makes SFT semi-on-policy. However, our LLaMA SFT checkpoints trained with traces from Qwen3-32B (Yang et al., 2025) are also competitive and learn more with slightly worse forgetting. That invalidates the same-family hypothesis (See details in Appendix F.1.1).

Observation 3: Some off-policy variants of RL outperform the fully on-policy version. We experiment with an off-policy variant of GRPO in the Qwen + Countdown setting. Specifically, we reduce the rollout sampling frequency but keep the same gradient update frequency, so that more than one (i.e., 2,4,8,16) gradient update is performed every time we sample. Results on the Qwen + Countdown setting in Figure 4 show that the 2x and 4x variants outperform the fully on-policy setting.

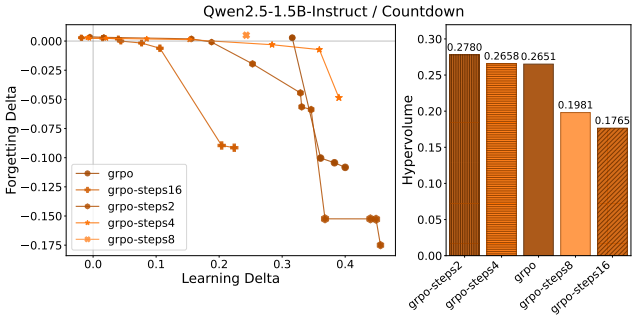


Figure 4: Comparison between on- and off-policy variants of GRPO with Qwen2.5-1.5B-Instruct and Countdown.

5.2 Full Plot Analysis

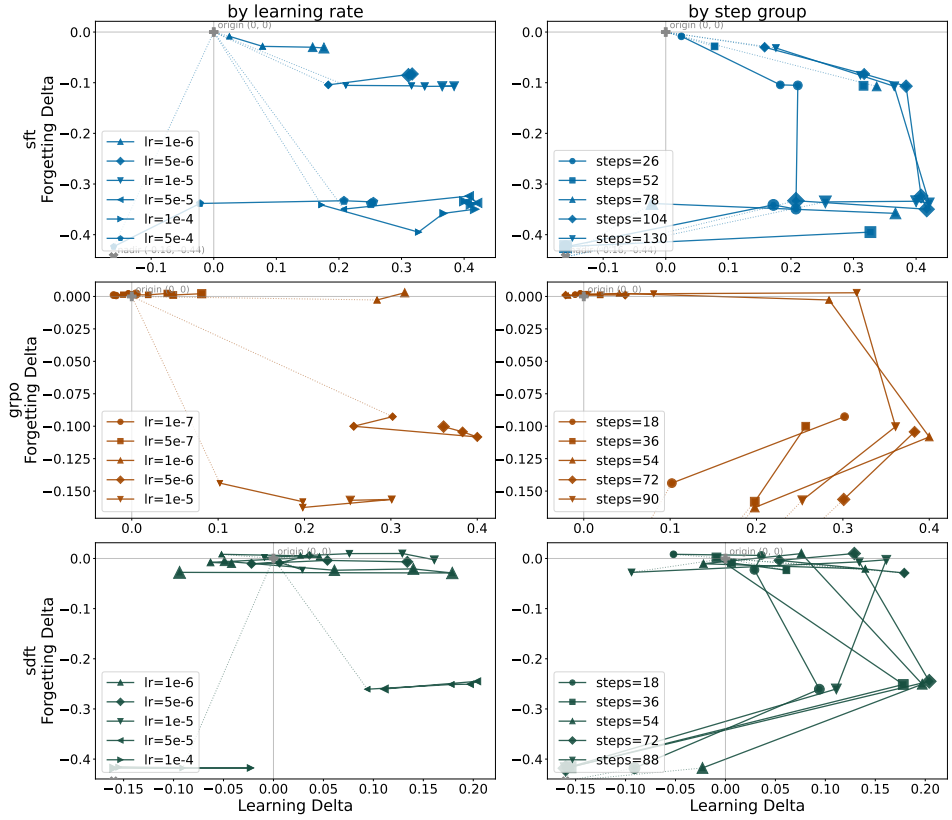


Figure 5: All evaluated checkpoints for Qwen2.5-1.5B-Instruct on Countdown, connected by learning rate (left) or training steps (right). Left: points with the same learning rate but different steps are connected. Larger markers are from later training steps; Right: The same set of points, but those with the same steps but different learning rates are connected. Larger markers are from higher learning rates. 85.23%, 87.00%, and 99.95% of forgetting occurs within the first checkpoint for SFT, GRPO, and SDFT, respectively.

Pareto front visualizes the best trade-offs, but it does not surface the relations between hyperparameters and learning-forgetting behaviors. Here, we add back the dominated checkpoints and analyze the full

plot. For each algorithm, we study how the tradeoff changes when we (1) fix the learning rate but increase training steps; (2) fix the training steps but increase the learning rate. Figure 5 shows the results of Qwen2.5-1.5B-Instruct on Countdown. See full results in Appendix F.3.

We observe that for most algorithms, the overall learning-forgetting tradeoffs follow a **"D" shape curve**. It starts at the point (0, 0) (i.e., learns nothing and forgets nothing), goes bottom right (i.e., learns more, forgets more), reaches a turning point (i.e., learning is maximized), goes bottom left (i.e., learns less due to overfitting and forgets even more), and finally approaches the $nadir$ point where the model's performance drops to zero on both target and non-target tasks (i.e., model collapses).

Most forgetting happens during initial training steps. Later training steps primarily affect learning. Looking at the left subplots in Figure 5, we find that most forgetting occurs before the first saved checkpoint (i.e., within 0.4 epochs), and later update steps affect only learning. The conclusion holds even with more training epochs (See § F.2.6). On average, we find that 83.35% of forgetting occurs within the first checkpoint (See § F.3).

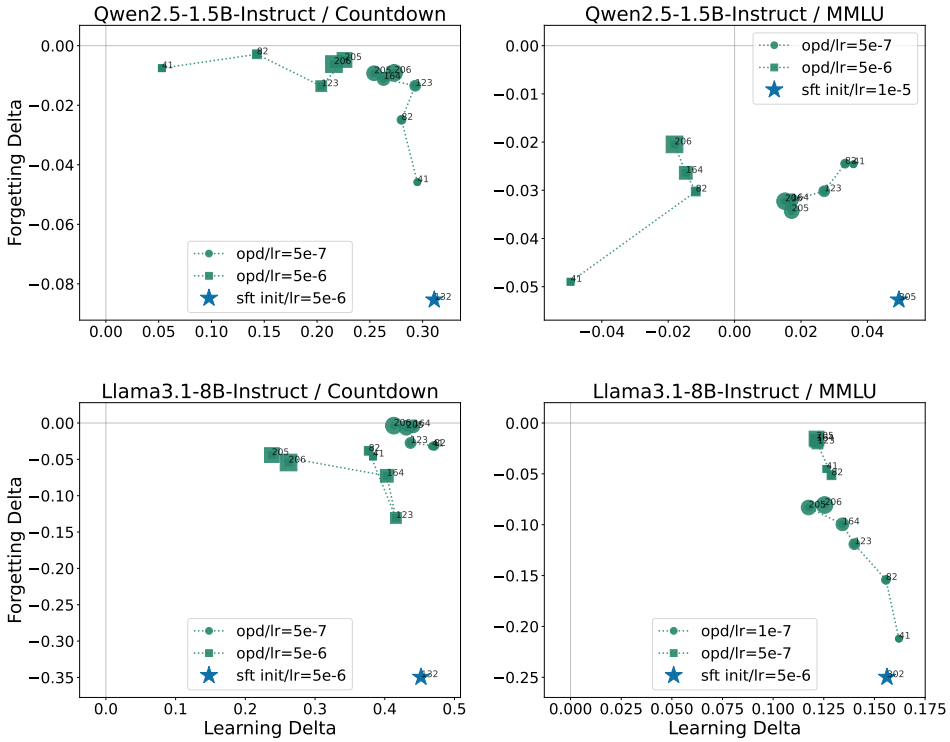


Figure 6: OPD recovery experiments.

Prior work (Zheng et al., 2025; Qi et al., 2024; 2025) observes similar phenomena. However, they only experiment with SFT on safety/instruction-tuning tasks, while we show the prevalence of this phenomenon across different fine-tuning algorithms. Zheng et al. (2025) additionally shows that fine-tuning-induced forgetting is easily reversible by a few steps of fine-tuning on data from forgotten or unrelated distributions. Results from our OPD recovery experiment confirm this: **The model quickly recovers from forgetting with a few steps of OPD on unrelated data.** Figure 6 shows that with just 41 steps of fine-tuning on the OpenAssistant dataset, forgetting is reduced from 5% to 2.5% on Qwen + MMLU (lr=5e-7), from 35% to 3% on LLaMA + Countdown (lr=5e-7), and from 25% to 5% on LLaMA + MMLU (lr=5e-7). On Qwen + Countdown, it takes only 82 steps to reduce forgetting from 8% to 2% (lr=5e-7).

Learning rate greatly affects both learning and forgetting. From the right subplots in Figure 5, checkpoints trained with the same data but with different learning rates greatly differ in both learning and forgetting. They follow the "D"-shaped curve. That echoes our previous finding: the degree of forgetting is mostly surfaced during initial steps, and learning rate greatly affects that degree.

6 Related Work

Catastrophic forgetting. Catastrophic forgetting—the loss of previously acquired capabilities when a model is trained on new data—is a long-standing failure mode of connectionist learners, first characterized by McCloskey & Cohen (1989) and later surveyed by French (1999). The same phenomenon persists in modern LLMs: Luo et al. (2023) show empirically that continual fine-tuning degrades domain knowledge, reasoning, and instruction-following ability, with severity growing in model scale. In the post-training setting, adapting a pretrained model to a narrow target distribution routinely erodes broad capabilities such as instruction following, mathematical reasoning, and safety alignment (Chen et al., 2025; Shenfeld et al., 2026b; Lai et al., 2026; Zhang et al., 2026; Kreutzer et al., 2017; Rofin et al., 2026; Nie, 2026).

Avoiding catastrophic forgetting is a prerequisite for continual learning, and numerous methods have been proposed. They can be grouped into two main lines: The first line of work modifies the loss function to constrain parameter updates (Kirkpatrick et al., 2017; Li & Hoiem, 2018; Chaudhry et al., 2019; Wu et al., 2026; Shao et al., 2024; Agarwal et al., 2024); The second line of work replays data from previous tasks during training Huang et al. (2024); de Masson d'Autume et al. (2019); Rolnick et al. (2019); Scialom et al. (2022); Mi et al. (2020); Rebuffi et al. (2017); Lopez-Paz & Ranzato (2017); Aside from these, other methods also exist (e.g., merge models before and after training (Lin et al., 2024)). In this work, we focus on comparing the methods from the first line, for two reasons: First, we target general-purpose fine-tuning methods that are commonly used by practitioners, instead of those that are intended for specific continual learning use cases; Second, the second line of methods relies on previous task data, which is often not available in the context of LLM fine-tuning (i.e., the pretraining/posttraining data). Nonetheless, we still include a representative and feasible method from the second line (OPD for recovery).

The learning–forgetting trade-off for LLM fine-tuning. Several recent studies report that RL-based fine-tuning forgets less than supervised fine-tuning (SFT) (Chen et al., 2025; Shenfeld et al., 2026b; Lai et al., 2026; Zhang et al., 2026). Ross et al. (2011); Chen et al. (2025); Shenfeld et al. (2026b) attribute the gap to on-policy sampling: on-policy samples carry non-trivial probability under the model and therefore induce small, conservative updates, whereas off-policy expert demonstrations can be arbitrarily unlikely and induce large, disruptive ones. Zhang et al. (2026) support this view by showing that SFT on on-policy samples forgets substantially less than SFT on expert demonstrations, and Wu et al. (2026) propose Dynamic Fine-tuning (DFT), which reweighs the negative log-likelihood by the model probability to down-weight low-probability samples.

Our work differs from prior work in both coverage and methodology. First, prior studies only compare standard SFT against one of RL or OPD methods (e.g., SFT versus GRPO); we instead evaluate SFT, RL, and OPD algorithms and their variants together (Section 2). Second, Chen et al. (2025); Lai et al. (2026); Zhang et al. (2026); Nie (2026) summarize an algorithm by a *single checkpoint*. Shenfeld et al. (2026b); Liu et al. (2026); Chen et al. (2024) use Pareto fronts but does not introduce metrics to compare different fronts. We treat each algorithm as a cloud of (learning, forgetting) pairs, characterize it by the *Pareto front* of that cloud, and compare algorithms by *Hypervolume* (Zitzler & Thiele, 1998; 1999; Zitzler et al., 2003; 2007). To our knowledge, this is the first *quantifiable* Pareto-front-based evaluation framework for comparing LLM fine-tuning algorithms in this space. Finally, we are the first to systematically examine how hyperparameters such as learning rate and training duration shape the learning-forgetting trade-off across different fine-tuning algorithms, as opposed to only SFT (Rofin et al., 2026; Marek et al., 2026).

7 Conclusion

We propose the use of Pareto front and Hypervolume as a fairer evaluation framework for LLM learning-forgetting trade-offs. Under this framework, we conduct a comprehensive empirical study spanning three algorithm families, ten individual algorithms, and various hyperparameters focusing on but not limited to learning rate and training duration. Our results give a counter-argument to prior work: no algorithms consistently dominate the learning-forgetting tradeoffs, and on-policy training is not always better. We also show the interactions between hyperparameters and tradeoffs: training steps mostly only affect learning after initial steps; learning rate greatly affects both learning and forgetting, and varying it gives a "D" curve. We believe this analysis better captures the complexity of fine-tuning, offering more insight than binary claims about which method "forgets less".

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets in our SFT-Variant and SDFT-Variant experiments, supporting qualitative and thematic data analysis in Appendix F.1.1, F.1.2, F.1.3, G, and interpreting results in Appendix F.1.1. We have not used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting dataset. Helping to develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and proposing or refining hypotheses are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, creating or editing software code related to figures and visualizations, drafting parts of a research paper in Appendix F.1.2, F.1.3, G, and editing a research paper to improve readability. We have reviewed all AI-assisted work. The synthesized outputs for input queries are manually inspected by the authors in terms of their correctness, and the figures, analysis, and writing are manually examined in terms of their faithfulness to the experiment results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For all experiments, the final hyperparameters are reported in Appendix E.1. The computation resources used are described in Appendix E.2. The training and evaluation prompts are shown in Appendix E.4.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.14740.

Afra Amini, Tim Vieira, and Ryan Cotterell. Better estimation of the Kullback–Leibler divergence between language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2504.10637.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of International Conference on Learning Representations*, 2019.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025. URL https://arxiv.org/abs/2510.18874.

Yupeng Chen, Senmiao Wang, Yushun Zhang, Zhihang Lin, Haozhe Zhang, Weijian Sun, Tian Ding, and Ruoyu Sun. Mofo: Momentum-filtered optimizer for mitigating forgetting in llm fine-tuning. *arXiv preprint arXiv:2407.20999*, 2024.

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

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

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

S. Han, K. Rao, A. Ettinger, L. Jiang, B. Y. Lin, N. Lambert, Y. Choi, and N. Dziri. Wildguard: Open one-stopmoderation tools for safety risks, jailbreaks, and refusals of llms. In *Advances in Neural Information Processing Systems — Datasets & Benchmarks Track*, 2024.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems*, 2021b.

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://arxiv.org/abs/2106.09685.

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

L. Jiang, K. Rao, S. Han, A. Ettinger, F. Brahman, S. Kumar, N. Mireshghallah, X. Lu, M. Sap, Y. Choi, and N. Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems — Datasets & Benchmarks Track*, 2024.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

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. In *Proceedings of the National Academy of Sciences (PNAS)*, volume 114, pp. 3521–3526, 2017.

J. Knowles and D. Corne. On metrics for comparing non-dominated sets. In *Congress on Evolutionary Computation*, pp. 711–716, 2002.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. OpenAssistant conversations – democratizing large language model alignment. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2304.07327.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. URL https://aclanthology.org/P17-1138.

Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, Gaofeng Meng, and Fei Zhu. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training, 2026. URL https://arxiv.org/abs/2507.05386.

Yaxuan Li, Yuxin Zuo amd Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2935–2947, 2018.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. Mitigating the alignment tax of rlhf. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Yuxing Liu, Jianyu Wang, and Tong Zhang. Optimizer-model consistency: Full finetuning with the same optimizer as pretraining forgets less. *arXiv preprint arXiv:2605.06654*, 2026.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. URL 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. *arXiv preprint arXiv:2308.08747*, 2023.

Martin Marek, Dongkyu Cho, Shikai Qiu, Rumi Chunara, Pavel Izmailov, and Andrew Gordon Wilson. Forgetting in language models: Capacity, optimization, and self-generated replay. *arXiv preprint arXiv:2605.26097*, 2026.

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

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. Continual learning for natural language generation in task-oriented dialog systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. URL https://aclanthology.org/D17-1153.

Dong Nie. Post-training is about states, not tokens: A state distribution view of sft, rl, and on-policy distillation. *arXiv preprint arXiv:2605.22731*, 2026.

J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A Suhr. Tinyzero, 2025. URL https://github.com/Jiayi-Pan/TinyZero.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

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

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *International Conference on Learning Representations*, 2025.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Mark Rofin, Aditya Varre, and Nicolas Flammarion. (how) learning rates regulate catastrophic overtraining. In *Proceedings of Conference on Language Modeling*, 2026.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026a. URL https://arxiv.org/abs/2601.19897.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=7HNRYT4V44.

Mingyu Su, Jian Guan, Yuxian Gu, Minlie Huang, and Hongning Wang. Trust-region adaptive policy optimization, 2025. URL https://arxiv.org/abs/2512.17636.

Rishabh Tiwari, Kusha Sareen, Devvrit Khatri, Lakshya A Agrawal, Kurt Keutzer, Matei Zaharia, Inderjit Dhillon, and Rishabh Agarwal. Why on-policy distillation works and naive self-distillation doesn't, 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. URL https://link.springer.com/article/10.1007/BF00992696.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of SFT: A reinforcement learning perspective with reward rectification. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Lv7PjbcaMi.

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

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, and Haoran Wei et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhihao Zhang, Qiaole Dong, Qi Zhang, Enyu Zhou, Jun Zhao, Zhiheng Xi, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Mingqi Wu, Yanwei Fu, Tao Ji, Tao Gui, Xuanjing Huang, and Kai Chen. Why reinforcement fine-tuning enables MLLMs preserve prior knowledge better: A data perspective. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tP01GCr0nS.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations*, 2025.

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

E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Parallel Problem Solving from Nature*, pp. 292–301, 1998.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Foneseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132, 2003.

E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: on the design of pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization*, pp. 862–876, 2007.

A FINE-TUNING ALGORITHMS

In the following, we list the individual fine-tuning algorithms in detail. For the specific hyperparameters (including loss normalization) of each algorithm, see Appendix E.

A.1 SFT-VARIANTS

Supervised Fine-Tuning (SFT)-Variants fine-tune the model to generate the expert-drafted output y^* with high probability, given x.

A.1.1 SFT

Supervised Fine-Tuning (SFT) is a general term. Here we abuse this term and use it to represent the standard cross-entropy loss-based SFT. The loss function is written as:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\log q_\theta(y^*|x)] \tag{1}$$

And $\log q_\theta(y^*|x)$ can be decomposed into $\sum_i \log q_\theta(y_i|y_{<i}, x)$ due to the autoregressive nature of modern LLMs. This loss originates from maximum log-likelihood training and is also equivalent to minimizing (a Monte Carlo estimate of) the forward KL divergence between the model distribution and the true data distribution. This is the de facto loss function for not only fine-tuning but also pre-training.

Over the years, from time to time, people have criticized this loss function and proposed alternatives (Wu et al., 2026; Su et al., 2025). A common criticism is centered around the mode-covering behavior of the forward KL divergence. It will be more of a problem when the data is scarce, and the model is under-specified (model is simpler than the data). However, modern LLMs are complex large-scale Transformers and are trained on trillions of tokens.

A.1.2 LORA

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a parameter-efficient fine-tuning method. Rather than updating all weights, it freezes each pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ and learns a low-rank update $\Delta W = BA$, with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and rank $r \ll \min(d, k)$, so that the adapted layer computes

$$h = W_0 x + \frac{\alpha}{r} BAx, \tag{2}$$

where α is a scaling factor and A, B are the only trainable parameters (B is initialized to zero so that training starts from the pretrained model). LoRA is not a new loss function: we combine it with the standard SFT objective (Eq. 1) and optimize only A and B. In the context of forgetting, LoRA is of interest because it restricts fine-tuning to a low-rank subspace and leaves the pretrained weights untouched, which has been hypothesized to limit the disruption of pretrained capabilities.

A.1.3 DFT

Wu et al. (2026) propose Dynamic Fine-tuning (DFT) as an alternative to SFT. They rewrite the SFT gradient as a policy gradient via Importance Sampling.

$$\nabla_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\log q_\theta(y^*|x)] = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim q_\theta(y|x)} \frac{\mathbf{1}[y = y^*]}{q_\theta(y|x)} [-\nabla_\theta \log q_\theta(y|x)] \tag{3}$$

where $\mathbf{1}[y = y^*]$ can be seen as the reward and $\frac{1}{q_\theta(y|x)}$ is the importance sampling weight. So when $y = y^*$ and $q_\theta(y^*|x)$ is small, the weight will be excessively large. They claim that this introduces "an ill-posed reward landscape." Therefore, they propose the DFT objective:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\text{sg}(q_\theta(y^*|x)) \log q_\theta(y^*|x)] \tag{4}$$

where $\text{sg}(\cdot)$ denotes the stop gradient operator. They believe $\text{sg}(q_\theta(y^*|x))$ can neutralized the " skewed reward issue." Since in practice, computing sequence-level $q_\theta(y^*|x)$ is infeasible, they opt for a step-level DFT loss function:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\sum_t \text{sg}(q_\theta(y^*_t|y^*_{<t}, x)) \log q_\theta(y^*_t|y^*_{<t}, x)] \tag{5}$$

Note that Eq. 5 and Eq. 4 are not equivalent. If we take the gradient of Eq. 4,

$$\nabla_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\text{sg}(q_\theta(y^*|x)) \log q_\theta(y^*|x)] = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-q_\theta(y^*|x)\frac{\nabla_\theta q_\theta(y^*|x)}{q_\theta(y^*|x)}] \tag{6}$$

$$= \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\nabla_\theta q_\theta(y^*|x)] \tag{7}$$

We can see that its gradient is equivalent to the gradient of *minimizing the negative likelihood*, i.e., *maximizing the likelihood*, instead of *log-likelihood*. However, since $q_\theta(y^*|x) \leq \max_i q_\theta(y_i^*|x)$, the optimal solution of maximizing $\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[q_\theta(y^*|x)]$ is to put all probability mass onto one single output y_i^*, i.e., $q_\theta(y_i^*|x)$ becomes a Dirac Delta function at y_i^*. Similarly, the step-level DFT (Eq. 5) is to maximize the sum of stepwise likelihoods, which will squeeze the distribution onto a single token at each step. This is acceptable for classic classification problems, but for language models, we do not want the probability distribution at each step to be deterministic. Nonetheless, early stopping can help prevent the squeezing from being harmful.

A.1.4 TRSFT

Su et al. (2025) inspect the gradient of SFT objective:

$$\nabla_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\log q_\theta(y^*|x)] = \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\sum_t \nabla_\theta \log q_\theta(y_t^*|y_{<t}^*, x)] \tag{8}$$

$$= \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\sum_t \frac{1}{q_\theta(y_t^*|y_{<t}^*, x)} \nabla_\theta q_\theta(y_t^*|y_{<t}^*, x)] \tag{9}$$

Similar to DFT, they also believe that the gradient will become more detrimental than helpful when $\frac{1}{q_\theta(y_t^*|y_{<t}^*,x)}$ becomes too large. Therefore, they propose Trust Region SFT (TrSFT) that clips the gradient as follows:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[-\sum_t \frac{1}{\max(q_\theta(y_t^*|y_{<t}^*, x), \alpha)} \nabla_\theta q_\theta(y_t^*|y_{<t}^*, x)] \tag{10}$$

Interestingly, in a Gaussian Mixture Model (GMM) based experiment, they show that TrSFT is more similar to reverse KL divergence instead of forward KL divergence.

A.1.5 MSE

Mean squared error (MSE) is often used for regression tasks. It is rarely used for classification problems (predicting the next token is essentially a classification problem) because of the exact opposite reason that DFT/TrSFT are based on.

For a classification problem, the output probability of a class is an output from softmax, or sigmoid for binary classification, e.g., $p = \frac{1}{1+e^{-l}}$, l is the logit. Then MSE loss on the positive class is $(1 - \frac{1}{1+e^{-l}})^2$. As l becomes a large negative number, the loss saturates at 1, and the gradient becomes almost zero, i.e., no learning signal when the model is completely wrong. Similarly, DFT loss is essentially just $-\frac{1}{1+e^{-l}}$, which is also flat when l is a large negative number and p is close to zero.

Formally, MSE loss is defined as follows ($\mathcal{V}$ is the model's vocabulary):

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}[\sum_t \sum_{w\in\mathcal{V}} (\mathbf{1}[w = y_t^*] - q_\theta(w|y_{<t}^*, x))^2] \tag{11}$$

A.2 RL-VARIANTS

Reinforcement Learning (RL)-Variants fine-tune the model to generate, with high probability, outputs y that have high rewards, given the reward model R and input x.

A.2.1 REINFORCE

Instead of maximizing the likelihood of expert demonstrations, reinforcement learning (RL) directly optimizes the model to maximize a task reward. Let $r(x, y)$ denote a scalar reward assigned to a response y for input x (e.g., a verifiable correctness signal on our target tasks). The RL objective is to maximize the expected reward under the model's own distribution,

$$\max_{\theta} \; \mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{y\sim q_\theta(\cdot|x)}[r(x, y)]. \tag{12}$$

REINFORCE (Williams, 1992) is the classical policy-gradient estimator for this objective,

$$\nabla_\theta\, \mathbb{E}_{y\sim q_\theta(\cdot|x)}[r(x, y)] = \mathbb{E}_{y\sim q_\theta(\cdot|x)}\big[(r(x, y) - b)\, \nabla_\theta \log q_\theta(y|x)\big], \tag{13}$$

where b is a baseline that does not depend on y; subtracting it reduces the variance of the gradient estimate without introducing bias. Because responses y are sampled on-policy from q_θ, REINFORCE only reinforces or suppresses sequences the model already assigns non-trivial probability, the property commonly credited with mitigating forgetting. Policy-gradient methods have a long history in sequence generation prior to their recent adoption for LLM post-training, for instance in neural machine translation through bandit / partial-feedback formulations with control-variate baselines (Kreutzer et al., 2017) and actor-critic training from simulated human feedback (Nguyen et al., 2017).

The central design choice in REINFORCE is the baseline. A large body of recent work avoids learning a separate value network by instead sampling a *group* of responses per prompt and forming the baseline from the group; RLOO (Ahmadian et al., 2024), for example, uses a leave-one-out average of the other samples' rewards. In this work, we adopt the simplest such group-based estimator: for each prompt x we draw a group of G responses $\{y_i\}_{i=1}^{G} \sim q_\theta(\cdot|x)$ with rewards $r_i = r(x, y_i)$, and compute a normalized advantage by subtracting the group mean and dividing by the group standard deviation,

$$A_i = \frac{r_i - \mathrm{mean}(\{r_j\}_{j=1}^{G})}{\mathrm{std}(\{r_j\}_{j=1}^{G})}. \tag{14}$$

Assigning the sequence-level advantage A_i to every token of y_i, the resulting loss is

$$\mathcal{L}_{\text{REINFORCE}} = -\,\mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{\{y_i\}\sim q_\theta(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} A_i \log q_\theta(y_i|x)\right]. \tag{15}$$

A.2.2 GRPO

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) uses the same group-normalized advantage as our REINFORCE variant (Eq. 14), but replaces the plain policy-gradient update with a PPO-style clipped surrogate and an explicit KL penalty toward a fixed reference model q_0 (typically the initial model). Rather than requiring fresh on-policy samples for every gradient step, GRPO reuses responses drawn from a behavior policy $q_{\theta_{\text{old}}}$ and corrects for the mismatch with a per-token importance ratio $\rho_{i,t} = q_\theta(y_{i,t}|y_{i,<t}, x)/q_{\theta_{\text{old}}}(y_{i,t}|y_{i,<t}, x)$. The objective is

$$\mathcal{L}_{\text{GRPO}} = -\,\mathbb{E}_{x\sim\mathcal{D},\{y_i\}\sim q_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \min\big(\rho_{i,t}A_i,\; \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\, A_i\big) - \beta\, D_{\text{KL}}(q_\theta \,\|\, q_0)\right], \tag{16}$$

where ϵ controls the clipping range and β the strength of the KL penalty. The clipping keeps updates within a trust region around $q_{\theta_{\text{old}}}$, while the KL term explicitly anchors the model to the reference distribution—an ingredient often argued to reduce forgetting, and whose effect we study directly by ablating β.

A.3 OPD-Variants

On-Policy-Distillation (OPD)-Variants fine-tune the model to mimic the next-token prediction distributions of a stronger teacher LLM p_t on outputs y generated by the student model, given x. It can be thought of as a variant of RL where the reward model is derived from a stronger LLM.

A.3.1 OPD

On-Policy Distillation (OPD) (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025) combines the on-policy sampling of RL with the dense, token-level supervision of distillation. Rather than scoring student samples with a scalar reward, a teacher model p_t grades them, and the student is trained to minimize the reverse KL divergence between its own *sequence-level* response distribution $q_\theta(\cdot \mid x)$ and the teacher's $p_t(\cdot \mid x)$:

$$\mathcal{L}_{\text{OPD}} = \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\text{KL}}\big(q_\theta(\cdot \mid x) \,\big\|\, p_t(\cdot \mid x)\big)\right] \tag{17}$$

where the divergence is taken between the full distributions over complete responses y. Sampling responses on-policy from the student removes the train–inference distribution mismatch of off-policy distillation, while distillation supplies more dense information per episode than a scalar reward, making OPD substantially more sample-efficient than RL. See Appendix C.1 on how we estimate the reverse KL term.

Lu & Thinking Machines Lab (2025) shows that OPD can be used to recover forgotten general capabilities after domain-specific fine-tuning, and we follow their approach. Specifically, we first run vanilla SFT on the target queries $\mathcal{D}$ with their golden response traces, and then apply OPD on a separate set of queries $\mathcal{D}'$ drawn from the OpenAssistant dataset (Köpf et al., 2023) with the OPD objective, in order to restore performance on the forgetting benchmarks.

A.3.2 SDFT

Self-Distillation Fine-Tuning (SDFT) (Shenfeld et al., 2026a) optimizes the same reverse-KL objective as OPD (Eq. 17); the two methods differ only in where the teacher p_t comes from. Whereas OPD distills from a separate teacher, SDFT constructs its teacher from the student itself. Concretely, the teacher is (i) initialized from the same initial student model rather than a stronger external model, and updated throughout training as an exponential moving average (EMA) of the student's parameters: $p_t = q_\theta^{\text{EMA}}$; (ii) given the golden demonstration y^* in context so that it can produce informed next-token distributions for the target task: $q_\theta^{\text{EMA}}(\cdot|x, y^*)$. The student is then trained to match this demonstration-conditioned self-teacher on its own on-policy samples. This keeps the training signal on-policy while injecting task information solely through the in-context demonstration rather than through a more capable external model.

In our study, we use SDFT to *learn* the target task: we train the model directly on the target queries $\mathcal{D}$ with the SDFT objective, mirroring the original use of Shenfeld et al. (2026a).

In addition, we introduce a natural variant, SDFT-teacher-SFT, that samples rollouts from the initial SDFT teacher $q_0(\cdot|x, y^*)$ for each query and supervise-finetune the model on it:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\mathbb{E}_{y\sim q_0(\cdot|x,y^*)}\left[-\log q_\theta(y|x)\right]$$

As we show in Appendix C.2, SDFT-teacher-SFT is forward KL, which is complementary to SDFT's reverse KL objective.

B Theoretical Justification of the use of Hypervolume

Hypervolume (HV) is originally used to quantify the proximity of an approximated Pareto front (the Pareto approximation set) to a Pareto optimal front. In this paper, we assume that all algorithms attempt to recover a universal, unknown Pareto optimal front with the best trade-offs, and so each algorithm gives a Pareto approximation set. Therefore, higher proximity (higher HV) is equivalent to better learning-forgetting trade-offs.

C KL estimator

C.1 OPD and SDFT

The sequence-level divergence in Eq. 17 cannot be computed exactly and must be estimated from samples. Lu & Thinking Machines Lab (2025) uses a Monte Carlo (**MC**) estimator: drawing M

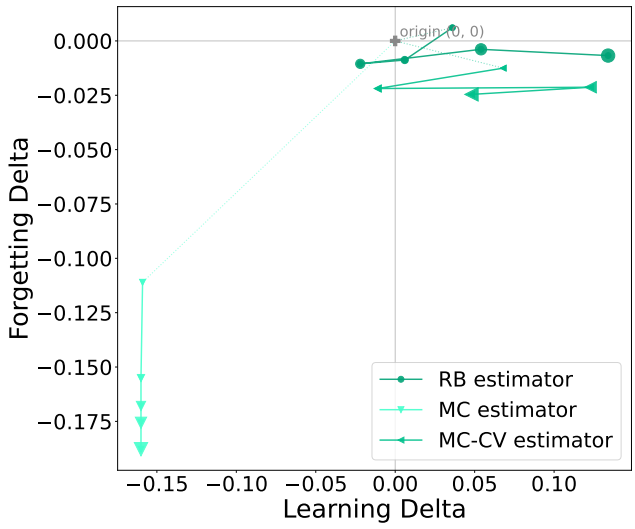


Figure 7: Comparison between our RB estimator and the MC and MC-CV estimators, with Qwen + Countdown + SDFT + lr 5e-6.

on-policy sequences $y^m \sim q_\theta(\cdot \mid x)$, it estimates the divergence by the sequence-level log-ratio, which decomposes autoregressively over tokens as

$$
\begin{aligned}
D_{\mathrm{KL}}\big(q_\theta(\cdot \mid x) \,\big\|\, p_t(\cdot \mid x)\big) &= \mathbb{E}_{y \sim q_\theta(\cdot|x)}\left[\log \frac{q_\theta(y \mid x)}{p_t(y \mid x)}\right] \\
&= \mathbb{E}_{\{y_i\} \sim q_\theta(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G} \log \frac{q_\theta(y^i \mid x)}{p_t(y^i \mid x)}\right] \\
&= \mathbb{E}_{\{y_i\} \sim q_\theta(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t} \big[\log \frac{q_\theta(y_t^i \mid y_{<t}^i, x)}{p_t(y_t^i \mid y_{<t}^i, x)}\big]\right].
\end{aligned}
$$

We instead adopt the Rao–Blackwellized (**RB**) estimator of Amini et al. (2025), which is unbiased for the same sequence-level divergence and provably has variance no larger than the Monte Carlo estimator. Conditioning on the sampled prefix and marginalizing the current token analytically, it replaces each realized token's log-ratio with the exact per-token KL, yielding the token-level form

$$
\begin{aligned}
D_{\mathrm{KL}}\big(q_\theta(\cdot \mid x) \,\big\|\, p_t(\cdot \mid x)\big) &= \mathbb{E}_{y \sim q_\theta(\cdot|x)}\left[\log \frac{q_\theta(y \mid x)}{p_t(y \mid x)}\right] \\
&= \mathbb{E}_{\{y_i\} \sim q_\theta(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t} D_{\mathrm{KL}}\big(q_\theta(\cdot \mid y_{<t}^i, x) \,\big\|\, p_t(\cdot \mid y_{<t}^i, x)\big)\right] \\
&= \mathbb{E}_{\{y_i\} \sim q_\theta(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t}\sum_{w \in \mathcal{V}} q_\theta(w \mid y_{<t}^i, x)\,\big[\log \frac{q_\theta(w \mid y_{<t}^i, x)}{p_t(w \mid y_{<t}^i, x)}\big]\right].
\end{aligned}
$$

The RB estimator is averaged over G on-policy trajectories $y \sim q_\theta(\cdot \mid x)$; in practice we draw a single sequence per prompt, so $G = 1$. We use this same RB estimator for both OPD and SDFT.

Aside from the MC and RB estimators, another common choice is the Monte-Carlo estimator with Control Variate (**MC-CV**), which adds an additional $\frac{p}{q} - 1$ term to the log probability ratio to reduce the variance of the estimation of the true KL Divergence:

$$D_{\mathrm{KL}}\big(q_\theta(\cdot \mid x) \,\big\|\, p_t(\cdot \mid x)\big) = \mathbb{E}_{y\sim q_\theta(\cdot|x)}\left[\log \frac{q_\theta(y \mid x)}{p_t(y \mid x)}\right]$$
$$= \mathbb{E}_{\{y_i\}\sim q_\theta(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t}\big[\log \frac{q_\theta(y_t^i \mid y_{<t}^i, x)}{p_t(y_t^i \mid y_{<t}^i, x)} + \frac{p_t(y_t^i \mid y_{<t}^i, x)}{q_\theta(y_t^i \mid y_{<t}^i, x)} - 1\big]\right]$$

In Figure 7, we show that RB estimator outperforms the MC estimator and MC-CV estimator, justifying our choice of the RB estimator. However, due to GPU memory constraints, we are only using the RB estimator for OPD variants. For RL-Variants, we use the MC-CV estimator.

C.2 SDFT-TEACHER-SFT

We show that forward KL with the teacher model is equivalent to SFT with teacher traces, as follows:

$$\begin{aligned}
&argmin_\theta \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\big(p_t(\cdot \mid x) \,\big\|\, q_\theta(\cdot \mid x)\big)\right] \\
&= argmin_\theta \mathbb{E}_{x\sim\mathcal{D}, y\sim p_t(\cdot|x)}\left[\log \frac{p_t(y|x)}{q_\theta(y|x)}\right] \\
&= argmin_\theta \left(\mathbb{E}_{x\sim\mathcal{D}, y\sim p_t(\cdot|x)}\left[\log p_t(y|x)\right] + \mathbb{E}_{x\sim\mathcal{D}, y\sim p_t(\cdot|x)}\left[-\log q_\theta(y|x)\right]\right) \\
&= argmin_\theta \mathbb{E}_{x\sim\mathcal{D}, y\sim p_t(\cdot|x)}\left[-\log q_\theta(y|x)\right]
\end{aligned}$$

Substituting $q_0(\cdot|x, y^*)$ into $p_t(\cdot|x)$, the last line is the SDFT-teacher-SFT objective, which is equivalent to forward KL in the first line.

D LIMITATION

Due to compute constraint and intractable search space, our hyperparameter search mainly focuses on learning rate and training duration. While we do examine other major hyperparameters (e.g., KL penalty, adapter rank), we simply fix these to the best values we find and adopt default values for other minor hyperparameters (e.g., batch size, warmup ratio). Therefore, the Pareto front for each algorithm is not exactly the optimal front that it can achieve. Future work can search for a more optimal front by conducting more holistic sweeps.

In addition, learning-forgetting tradeoffs have important implications in continual learning, which usually requires training the models on a chain of two tasks or more. Our study only trains the model on a single task in each setting. Future work can examine whether and how the learning-forgetting trade-off changes in longer-horizon continual learning scenarios.

E EXPERIMENT DETAILS

E.1 HYPERPARAMETERS

We show the learning rates in Table 2-5. Other hyperparameters are shown in Table 6-8.

Algorithm	Learning Rate Range
SFT	{1e-6,5e-6,1e-5,5e-5,1e-4,5e-4}
SFT w/ Qwen2.5-72B-Instruct Traces	{1e-6,5e-6,1e-5,5e-5,1e-4}
SFT-OKL	{1e-6,5e-6,1e-5,5e-5,1e-4}
SFT-OKL (KL $\beta = 0.1$)	{1e-6,5e-6,1e-5,5e-5,1e-4}
SFT-OKL (KL $\beta = 0.2$)	{1e-6,5e-6,1e-5,5e-5,1e-4}
LoRA	{1e-6,5e-6,1e-5,5e-5,1e-4,5e-4,1e-3,2e-3,5e-3}
LoRA (Rank=8; Alpha=16)	{1e-6,5e-6,1e-5,5e-5,1e-4,5e-4,1e-3,2e-3,5e-3}
DFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
MSE	{1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT ($\alpha = 0.05$)	{1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT ($\alpha = 0.2$)	{1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
GRPO	{1e-7,5e-7,1e-6,5e-6,1e-5}
GRPO (KL $\beta = 0$)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (KL $\beta = 0.1$)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (KL $\beta = 0.2$)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (Grad-steps-per-sampling=2)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (Grad-steps-per-sampling=4)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (Grad-steps-per-sampling=8)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO (Grad-steps-per-sampling=16)	{1e-7,5e-7,1e-6,5e-6,1e-5,2e-5,5e-5}
GRPO-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
REINFORCE	{1e-7,5e-7,5e-6,1e-5}
REINFORCE-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE-wo-baseline	{1e-7,5e-7,1e-6,5e-6,1e-5}
SDFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
SDFT-teacher-SFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
OPD	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}

Table 2: Learning Rate Ranges for Qwen2.5-1.5B-Instruct on Countdown.

Algorithm	Learning Rate Range
SFT	{1e-6,2e-6,5e-6,1e-5,5e-5,1e-4,5e-4}
SFT w/ Qwen3-32B traces	{1e-6,5e-6,1e-5,5e-5,1e-4}
LoRA	{1e-6,5e-6,1e-5,5e-5,1e-4,5e-4,1e-3}
DFT	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
MSE	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
GRPO	{1e-7,5e-7,7e-7,1e-6,2e-6,5e-6,1e-5}
GRPO-w-more-data	{1e-7,5e-7,7e-7,1e-6,2e-6,5e-6,1e-5}
REINFORCE	{1e-7,2e-7,5e-7,7e-7,1e-6,5e-6,1e-5}
REINFORCE-w-more-data	{1e-7,2e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE-wo-baseline	{1e-7,5e-7,1e-6,5e-6,1e-5}
SDFT	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
SDFT-teacher-SFT	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
OPD	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}

Table 3: Learning Rate Ranges for LLaMA3.1-8B-Instruct on Countdown.

Algorithm	Learning Rate Range
SFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
LoRA	{1e-6,5e-6,1e-5,5e-5,1e-4,5e-4,1e-3,5e-3}
DFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
MSE	{1e-6,5e-6,1e-5,5e-5,1e-4}
TRSFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
GRPO	{1e-7,5e-7,1e-6,2e-6,5e-6,1e-5}
GRPO-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE-wo-baseline	{1e-7,5e-7,1e-6,5e-6,1e-5}
SDFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
SDFT-teacher-SFT	{1e-6,5e-6,1e-5,5e-5,1e-4}
OPD	{5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}

Table 4: Learning Rate Ranges for Qwen2.5-1.5B-Instruct on MMLU.

Algorithm	Learning Rate Range
SFT	{1e-7,5e-7,1e-6,5e-6,1e-5,5e-5,1e-4}
LoRA	{1e-5,5e-5,1e-4,5e-4,1e-3,5e-3}
DFT	{1e-7,5e-7,1e-6,5e-6,1e-5}
MSE	{1e-7,5e-7,1e-6,5e-6,1e-5}
TRSFT	{1e-7,5e-7,1e-6,5e-6,1e-5,5e-5}
GRPO	{1e-7,5e-7,1e-6,2e-6,5e-6,1e-5}
GRPO-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE	{1e-7,5e-7,1e-6,2e-6,5e-6,1e-5}
REINFORCE-w-more-data	{1e-7,5e-7,1e-6,5e-6,1e-5}
REINFORCE-wo-baseline	{1e-7,5e-7,1e-6,5e-6,1e-5}
SDFT	{1e-7,5e-7,1e-6,5e-6,1e-5}
SDFT-teacher-SFT	{1e-7,5e-7,1e-6,5e-6,1e-5}
OPD	{1e-7,5e-7,1e-6,5e-6,1e-5}

Table 5: Learning Rate Ranges for LLaMA3.1-8B-Instruct on MMLU.

Dataset	Value
Num Train Queries (Countdown)	4,161
Num Train Queries (Ablation: RL w/ more data (Countdown))	10,000
Num Train Queries (MMLU)	9,645
Num Train Queries (Ablation: RL w/ more data (MMLU))	12,000
Num Training Queries (OPD)	9,846
Num Eval Queries (Countdown)	1,000
Num Eval Queries (MMLU)	2,042
Num Eval Queries (IFEval)	1,973
Num Eval Queries (MATH)	5,000
Num Eval Queries (WildGuardTest)	1,699
Num Eval Queries (WildJailBreak)	2,210

Table 6: Datasets

Hyperparameter	Value
Optimizer	AdamW (Loshchilov & Hutter, 2019)
LR scheduler	Cosine
Gradient Clipping	1.0
Maximum Content Length	8192
Maximum Rollout Length (RL/OPD-Variants)	4096
Rollout Group Size (RL-Variants)	5
Rollout Group Size (OPD-Variants)	1
Rollout Sampling Temperature	0.8
Rollout Sampling Top P	1.0
Rollout Batch Size (RL/OPD-Variants)	16
KL penalty β (SFT w/ prompt KL)	0.05
KL penalty β (SFT w/ output KL)	0.05
KL penalty β (GRPO)	0.05
KL penalty β (REINFORCE)	0
KL penalty estimator for RL	$\log \frac{q}{p} + (\frac{p}{q} - 1)$
Train Batch Size (SFT)	64
Train Batch Size (RL/OPD-Variants)	8
Gradient Accumulation Steps (SFT)	1
Gradient Accumulation Steps (RL/OPD-Variants)	12
Gradient Steps per Batch (RL/OPD-Variants)	1
LoRA Default Rank	32
LoRA Default Alpha	64
LoRA Default Dropout	0.05
TrSFT Default α	0.1
OPD Starting Checkpoint (Qwen; Countdown)	SFT lr 5e-6 Epoch 2 End
OPD Starting Checkpoint (Qwen; MMLU)	SFT lr 1e-5 Epoch 2 End
OPD Starting Checkpoint (LLaMA; Countdown)	SFT lr 5e-6 Epoch 2 End
OPD Starting Checkpoint (LLaMA; MMLU)	SFT lr 5e-6 Epoch 2 End
GRPO clipping range ϵ	0.1
SDFT EMA decay rate	0.99

Table 7: Training Hyperparameters.

Hyperparameter	Value
Performance Metric $\mathcal{A}$	Pass@1
Maximum Rollout Length	4,096
Num Rollout per Query	1
Rollout Sampling Temperature	0
Rollout Sampling Top P	1

Table 8: Evaluation Hyperparameters

E.2 Compute

All our training experiments use 8 Nvidia H100 GPUs, 500 GB RAM, and 16 CPU cores.

E.3 Evaluation

We report Pass@1 for all benchmarks. That is, for each query, we divide the number of outputs that pass by the total number of outputs (We only sample 1 output per query), and average this value across the queries to obtain $\mathcal{A}$. For Countdown, MMLU, and MATH, we ask the model to put the final answer in a specific format. The output is correct if and only if the extracted final answer Exactly Matches (EM) the ground truth. For IFEval, the output is correct if and only if it complies with all the format rules specific to the corresponding query. For the two safety benchmarks WildGuardTest and WildJailBreak, the model's output is correct if and only if it refuses the query when it is harmful and answers the query when it is benign.

E.4 PROMPT

We show the input prompts for all the benchmarks below. Note that the prompts for training and evaluation are the same for all benchmarks.

Countdown

```
In this task, you need to use a list of numbers x = ({{
input numbers }}) to create an equation that leads up to
the target number y = {{ output number }} using the basic
arithmetic operations (+, -, *, /), and each number can
only be used once.  Think and return the final answer in
\boxed{}.  For example, with x = (1, 2, 3) and y = 1, your
answer should be \boxed{(1 + 2)/3}.
```

MMLU

```
{{ question }}

Answer options:

{% for s in choices %}{{ 'ABCD'[loop.index0] }}.  {{ s }}{%
if not loop.last %}{% endif %} {% endfor %}

Reason about it and answer with "The answer is:  <option>"
```

IFEval

```
{{ question }}
```

MATH

```
{{ question }}

Reason about the problem, derive your answer, and wrap your
final answer in \boxed{}.
```

WildGuardTest

```
{{ question }}
```

WildJailBreak

```
{{ question }}
```

OpenAssistant (OPD)

```
{{ question }}
```

SDFT teacher

```
{{ original prompt }}

This is an example for a response to the question:  {{
golden response demonstration }}

Now answer with a response of your own, including the
thinking process.
```

F Full Results

F.1 Pareto front Analysis

F.1.1 On-Policy is not always better

For the SFT experiments, we use the traces from LLaMA3.3-70B-Instruct by default. We experiment with other more on/off-policy SFT traces to show that on-policiness of SFT-traces does not necessarily suggest better learning-forgetting trade-offs. In the following, we show that traces from same/different model families are indeed more/less on-policy. We show the Perplexity (Token PPL) and its byte-normalized version (BPB) in Table 9. For Qwen2.5-1.5B, Qwen2.5-72B-Instruct's traces have lower BPB and PPL than the default LLaMA3.3-70B-Instruct traces for both Countdown and MMLU, so they are more on-policy. For LLaMA3.1-8B-Instruct, the default LLaMA3.3-70B-Instruct has lower BPB and PPL than Qwen3-32B's traces, and is more on-policy. Therefore, our current choices of Qwen72B/LLaMA70B traces for Qwen1.5B and LLaMA70B/Qwen32B traces for LLaMA8B give valid on/off-policy contrast.

We cannot use the same Qwen model in both Countdown and MMLU to contrast with LLaMA70B. This is because, for Qwen1.5B, Qwen3-32B's traces are more off-policy than the default, even if it is from the same family as the 1.5B model. Additionally, for LLaMA8B, Qwen2.5-72B-Instruct's traces are more off-policy for MMLU but more on-policy for Countdown.

Table 9: Bits-per-byte (BPB) and token perplexity (PPL) of the original SFT teacher traces, measured under each student scoring model. Per-trace medians. Lower BPB/PPL means the trace is more on-policy (less surprising) for that student. Top panel: Countdown; bottom panel: MMLU. All models are instruct models.

	BPB		Token PPL	
Teacher	Qwen2.5-1.5B	LLaMA3.1-8B	Qwen2.5-1.5B	LLaMA3.1-8B
Countdown				
LLaMA3.3-70B	0.3513	0.2513	2.360	1.921
Qwen2.5-72B	0.1840	0.2411	1.366	1.611
Qwen3-32B	0.5205	0.5460	2.798	3.630
MMLU				
LLaMA3.3-70B	0.2691	0.1619	2.573	1.797
Qwen2.5-72B	0.2458	0.2125	2.365	2.152
Qwen3-32B	0.3310	0.2849	2.771	2.439

F.1.2 Inter Algorithm Family Comparison

With the hypervolumes, we compare the algorithms from different families, both individually and in aggregate. We rank the algorithms for each setting separately. Kendall's coefficient of concordance (W) is computed to measure the agreement between the four rankings. W is linearly related with the mean pairwise Spearman Correlation Coefficient. $W = 0$ means no agreement between the rankings and $W = 1$ means complete agreement. With the number of rankings equal to four, $W \leq 0.25$ if and only if the mean pairwise Spearman Correlation Coefficient is non-positive (i.e., on average, the rankings are not positively correlated). We exclude OPD from all rankings because it trains models from a single SFT checkpoint, and a fair comparison is not possible.

Table 10 shows the rankings of individual algorithms by their hypervolumes. W is only 0.2875, suggesting a very poor agreement between rankings from the four settings, and **no global best algorithm exists**.

We further aggregate the rankings of individual algorithms into a family-level ranking. Table 11 shows the results. Again, the ranking agreement is very poor ($W = 0.2500$) and **no global best algorithm family exists**.

$W = 0.2875$	Qwen/CD	Qwen/MMLU	LLaMA/CD	LLaMA/MMLU	Avg rank ± std
GRPO	2 (0.2651)	1 (0.2729)	6 (0.3826)	5 (0.4482)	3.5000 ± 2.0616
REINFORCE	1 (0.3489)	4 (0.2665)	9 (0.1958)	1 (0.4580)	3.7500 ± 3.2692
SFT	4 (0.2456)	7 (0.2413)	2 (0.4171)	3 (0.4519)	4.0000 ± 1.8708
TrSFT	3 (0.2632)	8 (0.2364)	4 (0.4019)	2 (0.4536)	4.2500 ± 2.2776
MSE	5 (0.2424)	6 (0.2413)	3 (0.4141)	4 (0.4495)	4.5000 ± 1.1180
SDFT-teacher-SFT	8 (0.1953)	3 (0.2693)	1 (0.4256)	8 (0.4034)	5.0000 ± 3.0822
LoRA	6 (0.2375)	2 (0.2709)	7 (0.3695)	6 (0.4475)	5.2500 ± 1.9203
DFT	7 (0.2024)	9 (0.2247)	5 (0.3958)	7 (0.4398)	7.0000 ± 1.4142
SDFT	9 (0.1732)	5 (0.2580)	8 (0.3489)	9 (0.3278)	7.7500 ± 1.6394

Table 10: Global hypervolume rankings across settings. Cells are rank (hypervolume); top-left is Kendall's coefficient of concordance (W) of the rankings for the four settings; last column is the mean rank ± std across settings.

$W = 0.2500$	Qwen/CD	Qwen/MMLU	LLaMA/CD	LLaMA/MMLU	Avg rank ± std
RL-variants	1 (1.5000)	1 (2.5000)	3 (7.5000)	1 (3.0000)	1.5000 ± 0.8660
SFT-variants	2 (5.0000)	3 (6.4000)	1 (4.2000)	2 (4.4000)	2.0000 ± 0.7071
OPD-variants	3 (8.5000)	2 (4.0000)	2 (4.5000)	3 (8.5000)	2.5000 ± 0.5000

Table 11: Inter-algorithm rankings across settings. Each family is ranked by the average global rank of its member methods; Each cell contains the new rank and (average global rank); the last column is the mean rank ± std across settings.

$W = 0.5125$	Qwen/CD	Qwen/MMLU	LLaMA/CD	LLaMA/MMLU	Avg rank ± std
SFT	2 (0.2456)	3 (0.2413)	1 (0.4171)	2 (0.4519)	2.0000 ± 0.7071
TrSFT	1 (0.2632)	4 (0.2364)	3 (0.4019)	1 (0.4536)	2.2500 ± 1.2990
MSE	3 (0.2424)	2 (0.2413)	2 (0.4141)	3 (0.4495)	2.5000 ± 0.5000
LoRA	4 (0.2375)	1 (0.2709)	5 (0.3695)	4 (0.4475)	3.5000 ± 1.5000
DFT	5 (0.2024)	5 (0.2247)	4 (0.3958)	5 (0.4398)	4.7500 ± 0.4330

Table 12: SFT hypervolume rankings across settings. Cells are rank (hypervolume).

$W = 0.0000$	Qwen/CD	Qwen/MMLU	LLaMA/CD	LLaMA/MMLU	Avg rank ± std
REINFORCE	1 (0.3489)	2 (0.2665)	2 (0.1958)	1 (0.4580)	1.5000 ± 0.5000
GRPO	2 (0.2651)	1 (0.2729)	1 (0.3826)	2 (0.4482)	1.5000 ± 0.5000

Table 13: RL hypervolume rankings across settings. Cells are rank (hypervolume).

$W = 1.0000$	Qwen/CD	Qwen/MMLU	LLaMA/CD	LLaMA/MMLU	Avg rank ± std
SDFT-teacher-SFT	1 (0.1953)	1 (0.2693)	1 (0.4256)	1 (0.4034)	1.0000 ± 0.0000
SDFT	2 (0.1732)	2 (0.2580)	2 (0.3489)	2 (0.3278)	2.0000 ± 0.0000

Table 14: OPD-variants hypervolume rankings across settings. Cells are rank (hypervolume).

F.1.3 Intra Algorithm Family Comparison

We further compare the algorithms within each family. Table 12 shows the rankings among the SFT-Variants. We observe a mild positive correlation between rankings ($W = 0.5125$). On average, vanilla SFT is the best, and DFT is the worst.

Table 13 shows the rankings for RL methods. GRPO underperforms REINFORCE on Qwen/CD and LLaMA/MMLU but outperforms it in two other settings, so there is no ranking agreement.

Table 14 shows the rankings for OPD-Variants. SDFT-teacher-SFT consistently outperforms SDFT in all settings.

F.2 Other hyperparameter search

We experiment with more variants of different algorithms. Unless stated otherwise, the experiments are done in the Qwen + Countdown setting.

F.2.1 SFT

Adding KL. To mitigate forgetting, we add **output KL** (RB Estimator with $G = 1$) to the SFT loss:

$$\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\left[-\log q_\theta(y^*|x) + \beta \sum_t \sum_{w\in\mathcal{V}} q_\theta(w \mid x, y^*_{<t}) \left[\log \frac{q_\theta(w \mid x, y^*_{<t})}{q_0(w \mid x, y^*_{<t})}\right]\right] \tag{18}$$

Note that $\beta = 0.05$ by default, but we also experiment with $\beta = 0.1$ and $\beta = 0.2$.

Figure 8 shows that adding KL penalties degrades the forgetting-learning trade-offs.

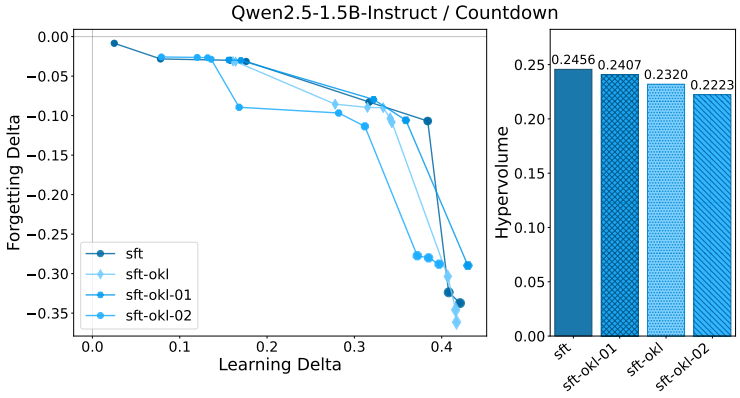


Figure 8: Adding KL to SFT.

F.2.2 LoRA

Varying LoRA Rank. We change the LoRA rank from 32 (Default) to 8. We also change its Alpha from 64 to 16. Figure 9 shows that changing the ranks gives marginal improvement in the forgetting-learning trade-offs.

F.2.3 TrSFT

Tuning α. By default, the denominator clipping constant $\alpha = 0.1$. We change it to 0.05 or 0.2 and show the results in Figure 10 and find that the default value gives the best trade-offs.

F.2.4 GRPO

Varying KL Penalty. By default, the KL penalty $\beta = 0.05$. We change it to 0, 0.1, or 0.2. Figure 11 shows that setting $\beta = 0$ gives the best trade-off. However, since GRPO is fully on-policy and the

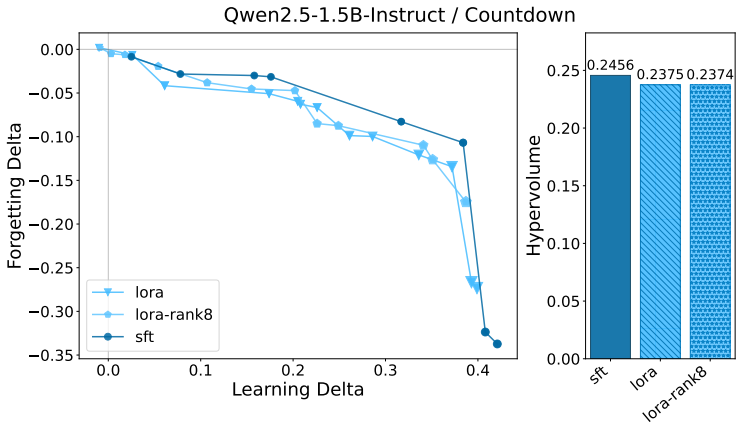


Figure 9: Varying the LoRA Ranks.

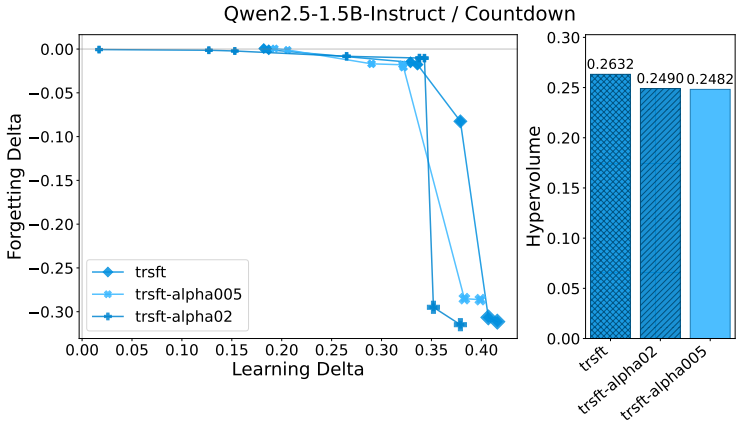


Figure 10: Varying the α hyperparameter for TrSFT.

importance ratio is supposedly always one, removing the KL penalty makes it almost identical to REINFORCE. Therefore, we do not use $\beta = 0$ as the default for GRPO.

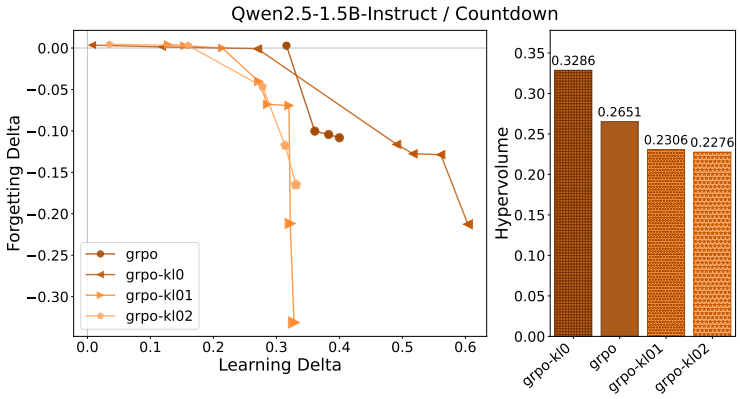


Figure 11: Varying the KL Penalty.

Removing Baseline. We use the raw binary reward (0 or 1) as the advantage A_i. This means the model only learns from positive rollouts and there are no negative gradients. Figure 12 shows that removing the baseline significantly degrades the performance.

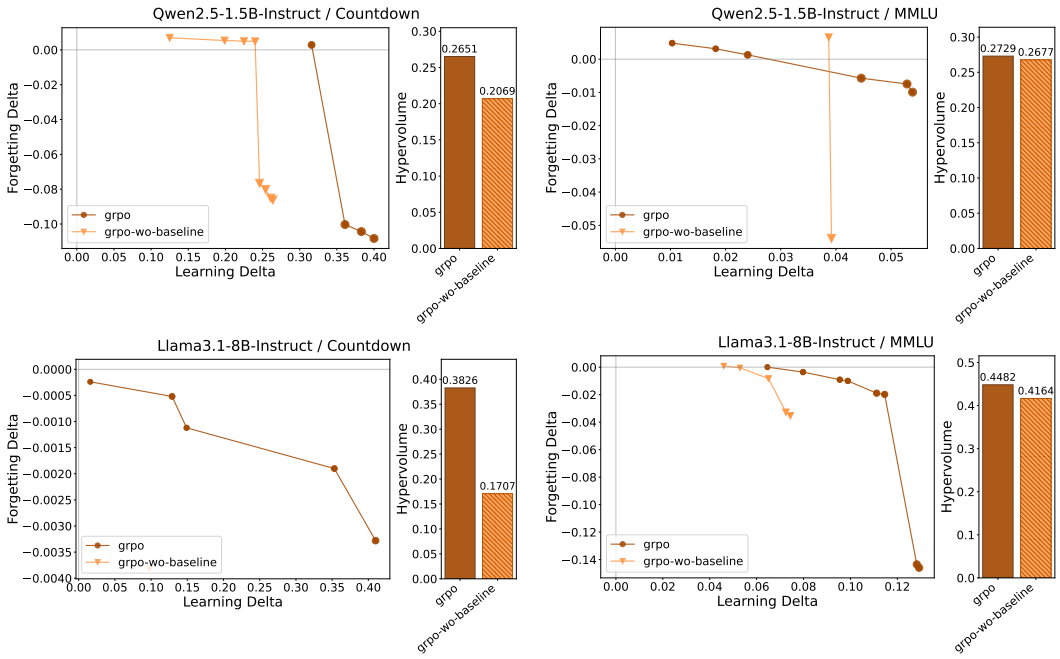


Figure 12: Removing the Baseline. grpo-wo-baseline for LLaMA3.1-8B-Instruct on Countdown has a single point and is occluded by the legend.

F.2.5 RL-VARIANTS

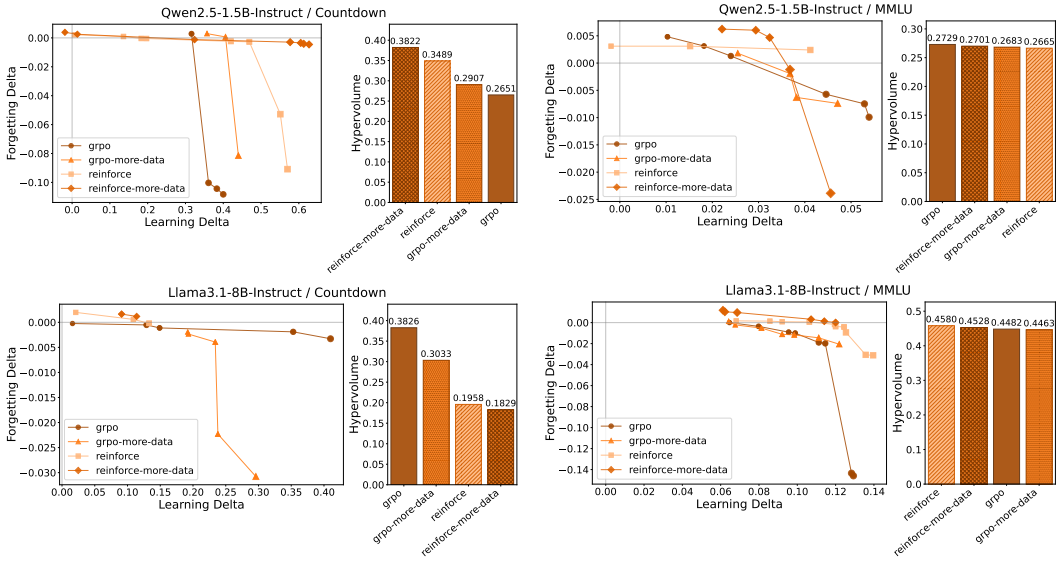


Figure 13: Adding more training queries.

More training data. In § 5.1, we show that RL-Variants has a lower learning cap than SFT, which is why it cannot outperform SFT on the two LLaMA settings. In light of this, we add more queries to the training set to check if increasing data increases the learning gap. For Countdown, we expand the training set from 4,161 queries to 10,000 queries. For MMLU, we expand the training set from 9,645 queries to 12,000 queries. Figure 13 gives mixed results. In the Qwen + Countdown setting, more data improves the trade-off for both GRPO and REINFORCE. In the Qwen + MMLU setting, the trade-off slightly improves for REINFORCE but slightly worsens for GRPO. On the LLaMA

+ Countdown setting, trade-offs are notably worse for both algorithms, especially GRPO. On the LLaMA + MMLU setting, trade-offs are slightly worse for both algorithms. Therefore, even with more data, RL-Variants still do not outperform SFT on the two LLaMA settings.

F.2.6 More training epochs

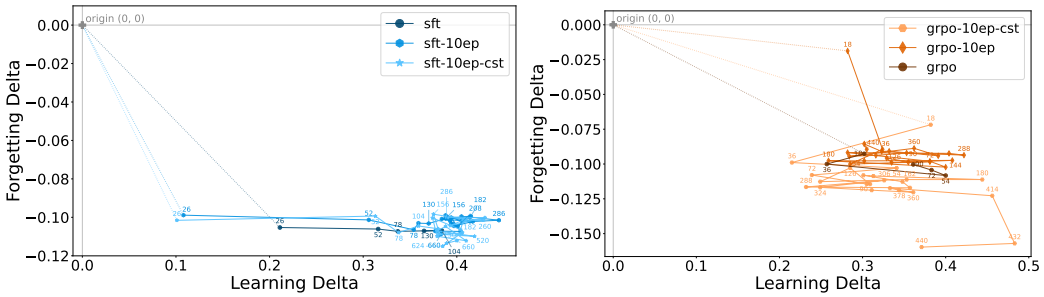


Figure 14: Results of checkpoints from 10 epochs of training, for Qwen + Countdown + SFT/GRPO; cst: constant learning rate schedule, instead of the default cosine schedule.

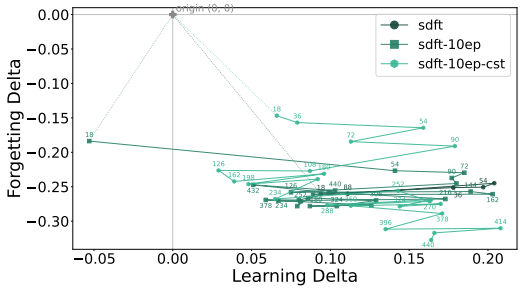


Figure 15: Results of checkpoints from 10 epochs of training, for Qwen + Countdown + SDFT.

We increase the number of training epochs from two to ten. Figure 14 and Figure 15 show that for SFT, GRPO, and SDFT, (1) later training steps do not affect forgetting too much, echoing our finding in § 5.2; and (2) the peak learning achievable within two epochs is close to the peak learning achievable within ten epochs, so the Pareto front from checkpoints within two epochs is largely accurate. This justifies our default choice of two training epochs for the grid search.

F.3 Full Plot Analysis

We show the full plot analysis results in Figure 16-23. Most forgetting occurs during the initial training steps. Later steps primarily affect only learning, but not forgetting. This is true across settings, algorithms, and learning rates. Exceptions include REINFORCE in the Qwen + Countdown setting, MSE in the LLaMA + Countdown setting, MSE and REINFORCE in the Qwen + MMLU setting, and REINFORCE in the LLaMA + MMLU setting.

For each training run with the same algorithm and learning rate, we compute the ratio between ΔF of the first checkpoint and the maximum ΔF among all the checkpoints within that run. We skip cases where the maximum ΔF is within 3%, and then average the ratios across the algorithms (excluding OPD) and learning rates. The final average ratio is 83.75% for Qwen + Countdown, 78.07% for LLaMA + Countdown, 83.72% for Qwen + MMLU, and 87.87% for LLaMA + MMLU. Averaging again across the four settings, 83.35% of forgetting occurs within the training steps of the first checkpoint.

We further evaluate earlier checkpoints and show the results in Figure 24,25. On Qwen + Countdown, ten steps of training suffice for most forgetting to occur for SFT and SDFT. For GRPO, it takes 16 steps.

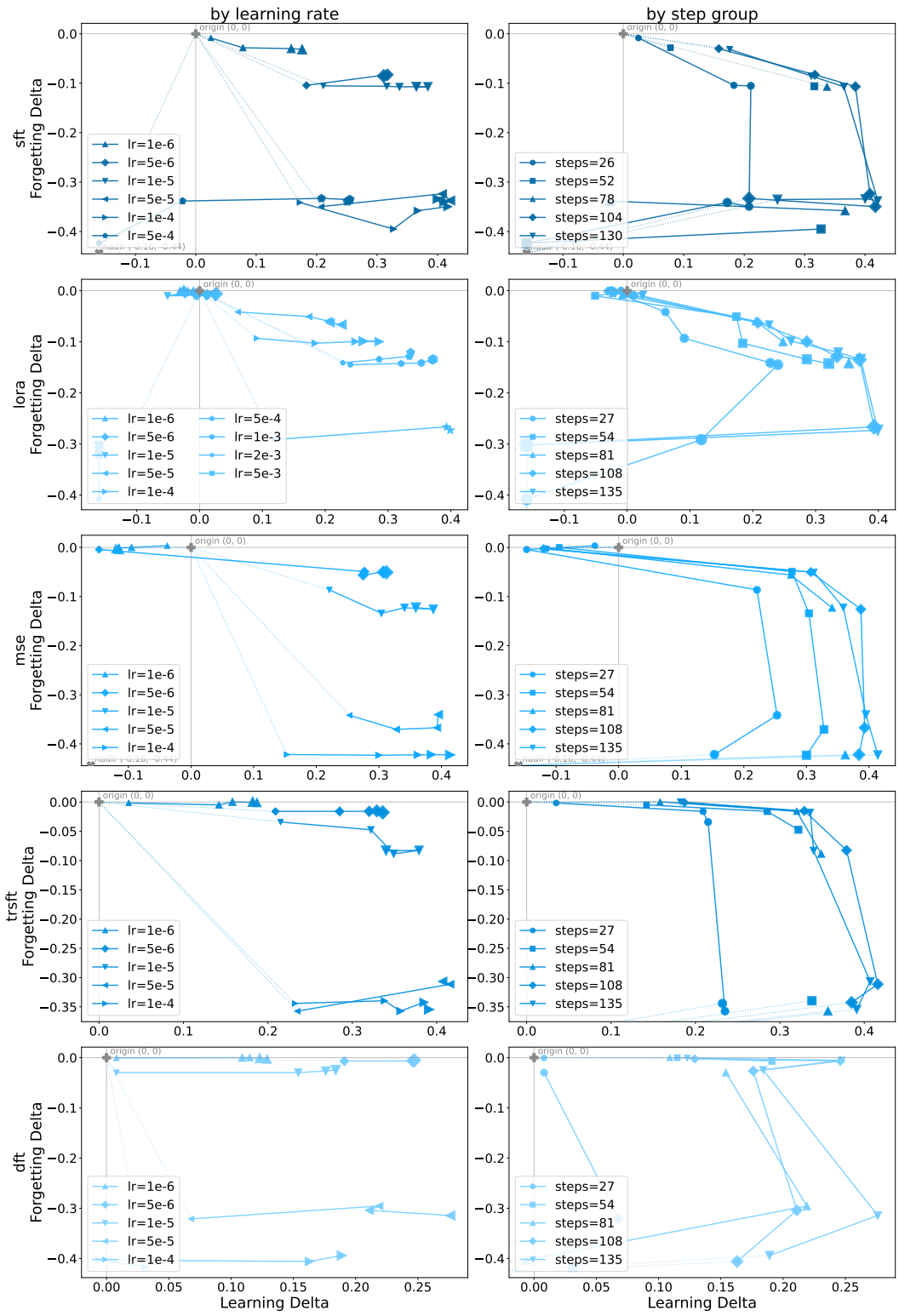


Figure 16: Full Plot Analysis for Qwen2.5-1.5B-Instruct on Countdown Algorithm Group 1.

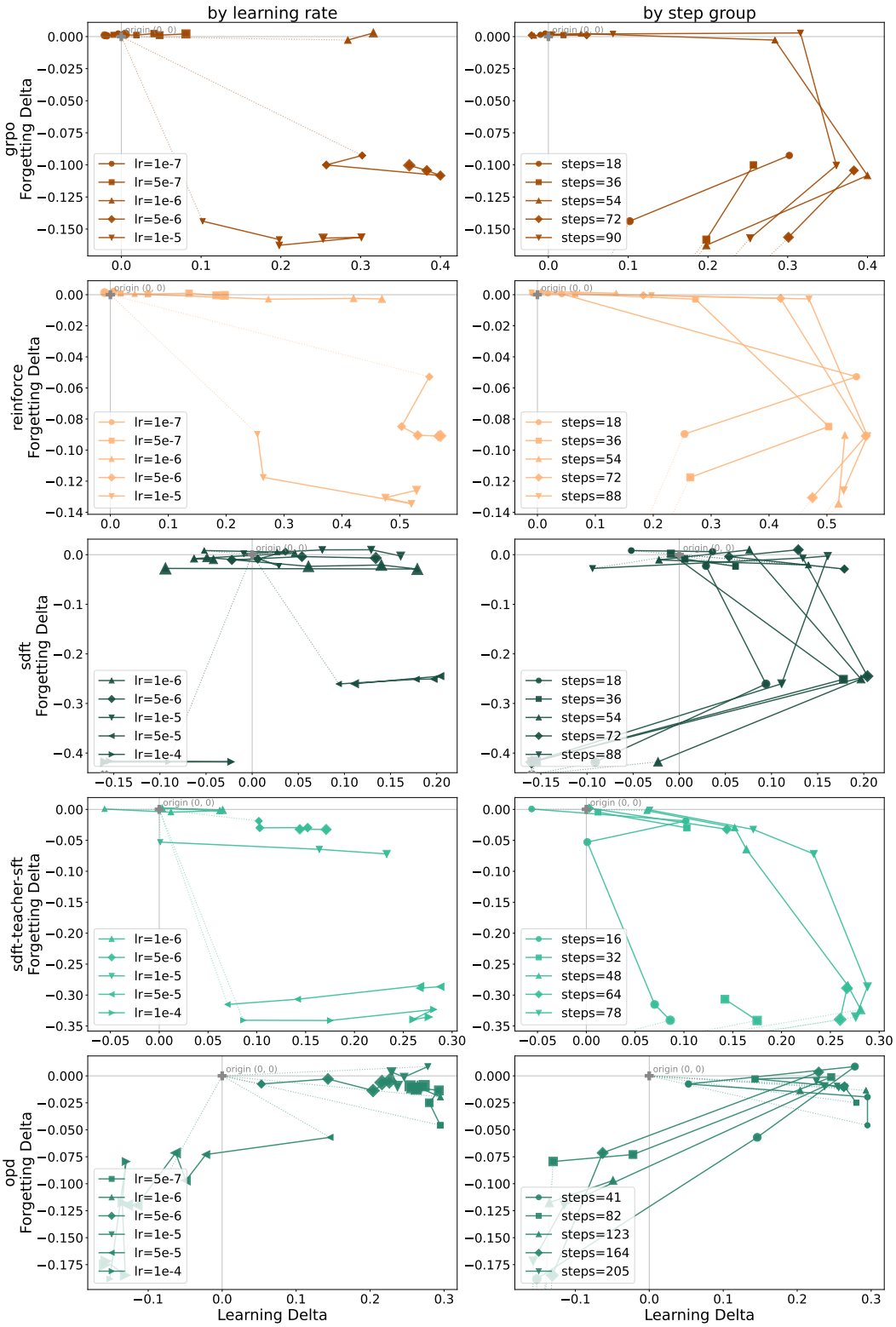


Figure 17: Full Plot Analysis for Qwen2.5-1.5B-Instruct on Countdown Algorithm Group 2.

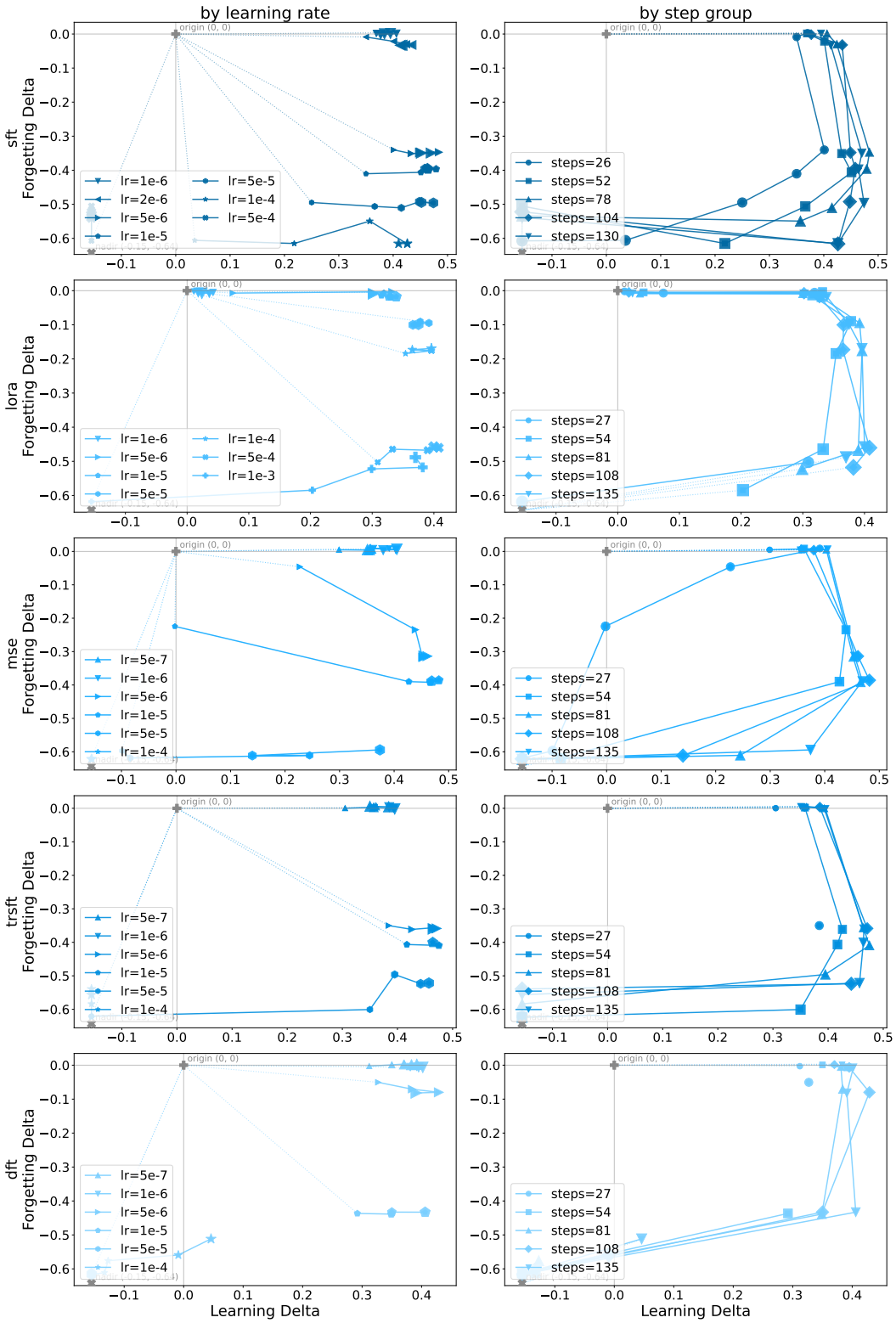


Figure 18: Full Plot Analysis for LLaMA3.1-8B-Instruct on Countdown Algorithm Group 1.

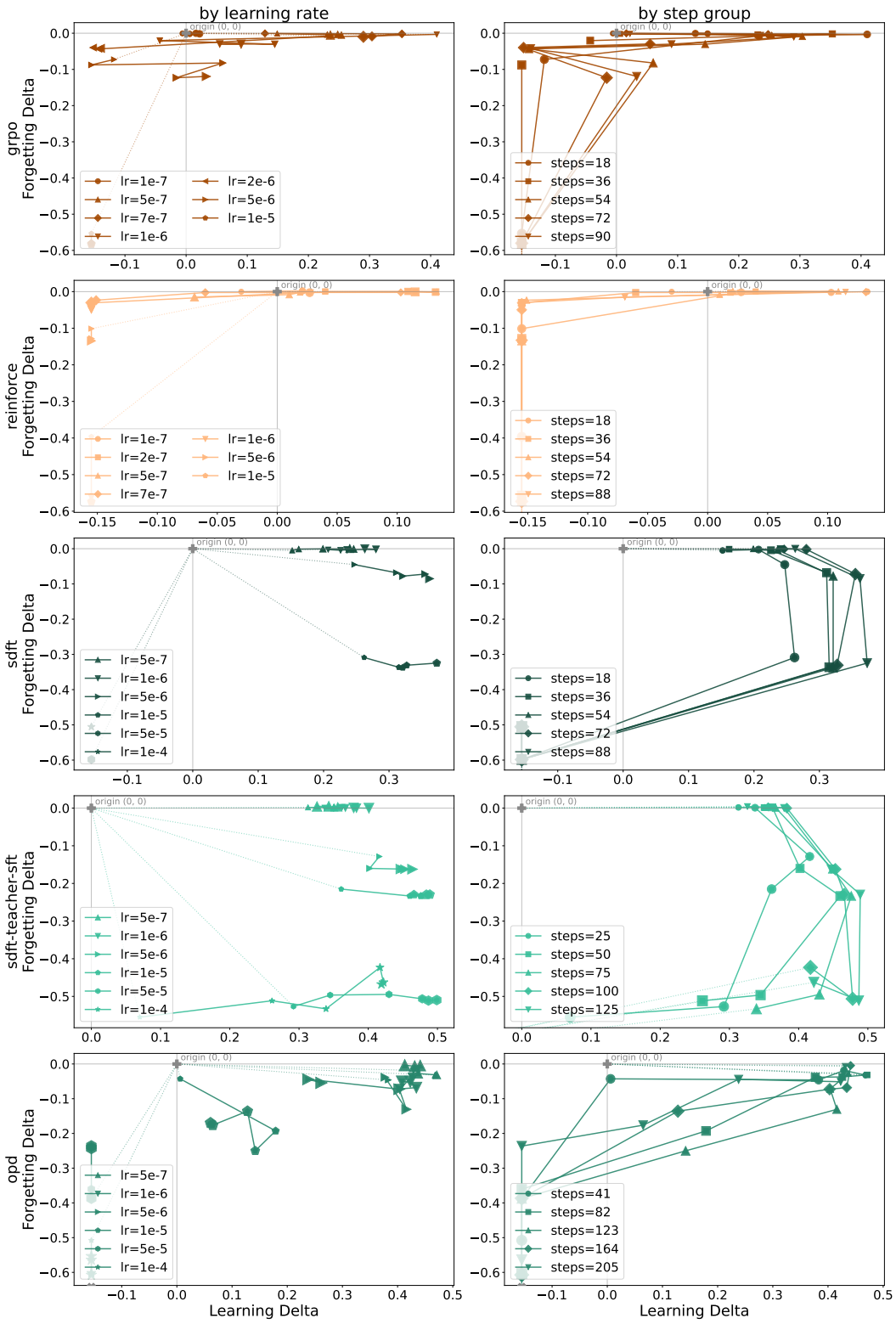


Figure 19: Full Plot Analysis for LLaMA3.1-8B-Instruct on Countdown Algorithm Group 2.

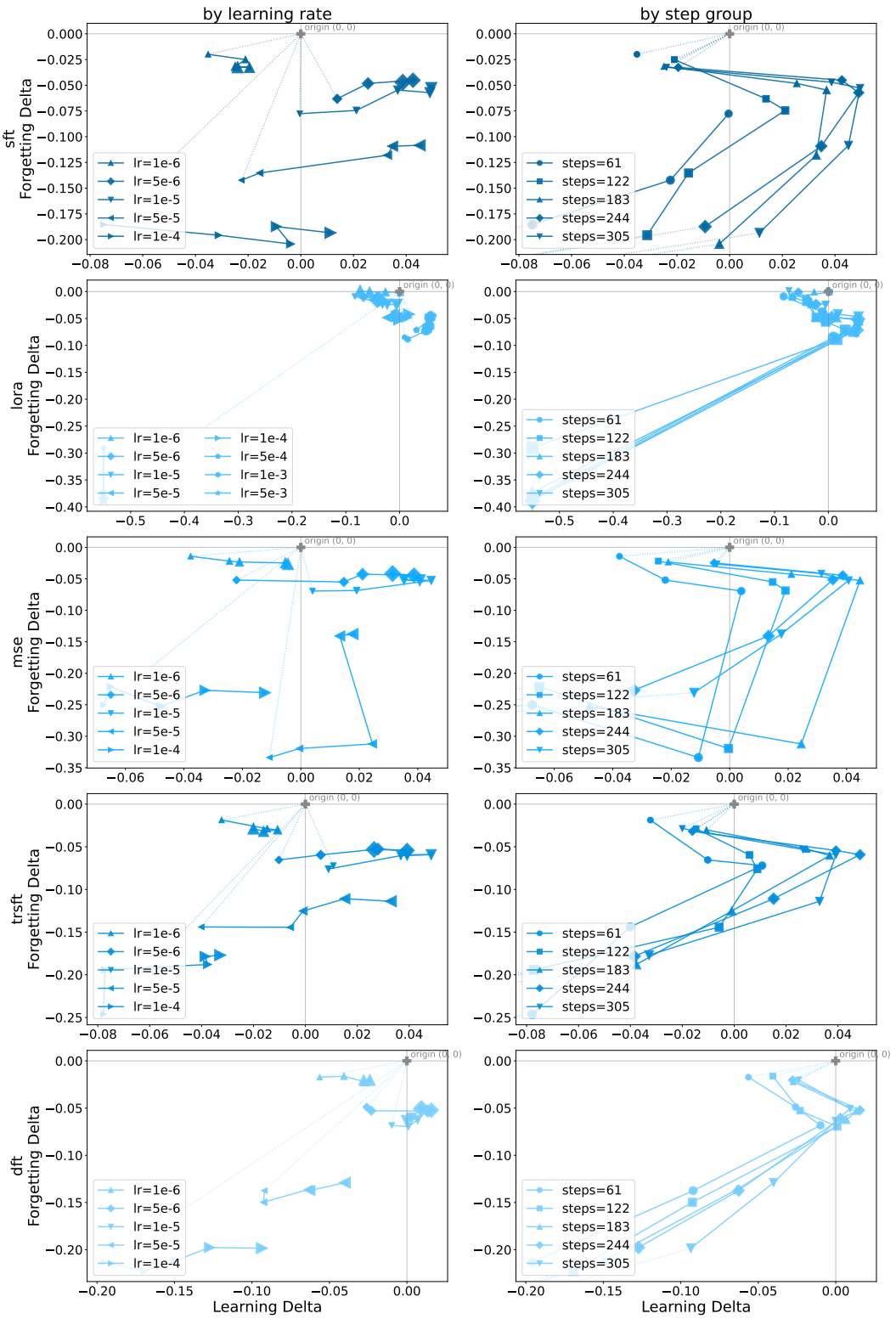


Figure 20: Full Plot Analysis for Qwen2.5-1.5B-Instruct on MMLU Algorithm Group 1.

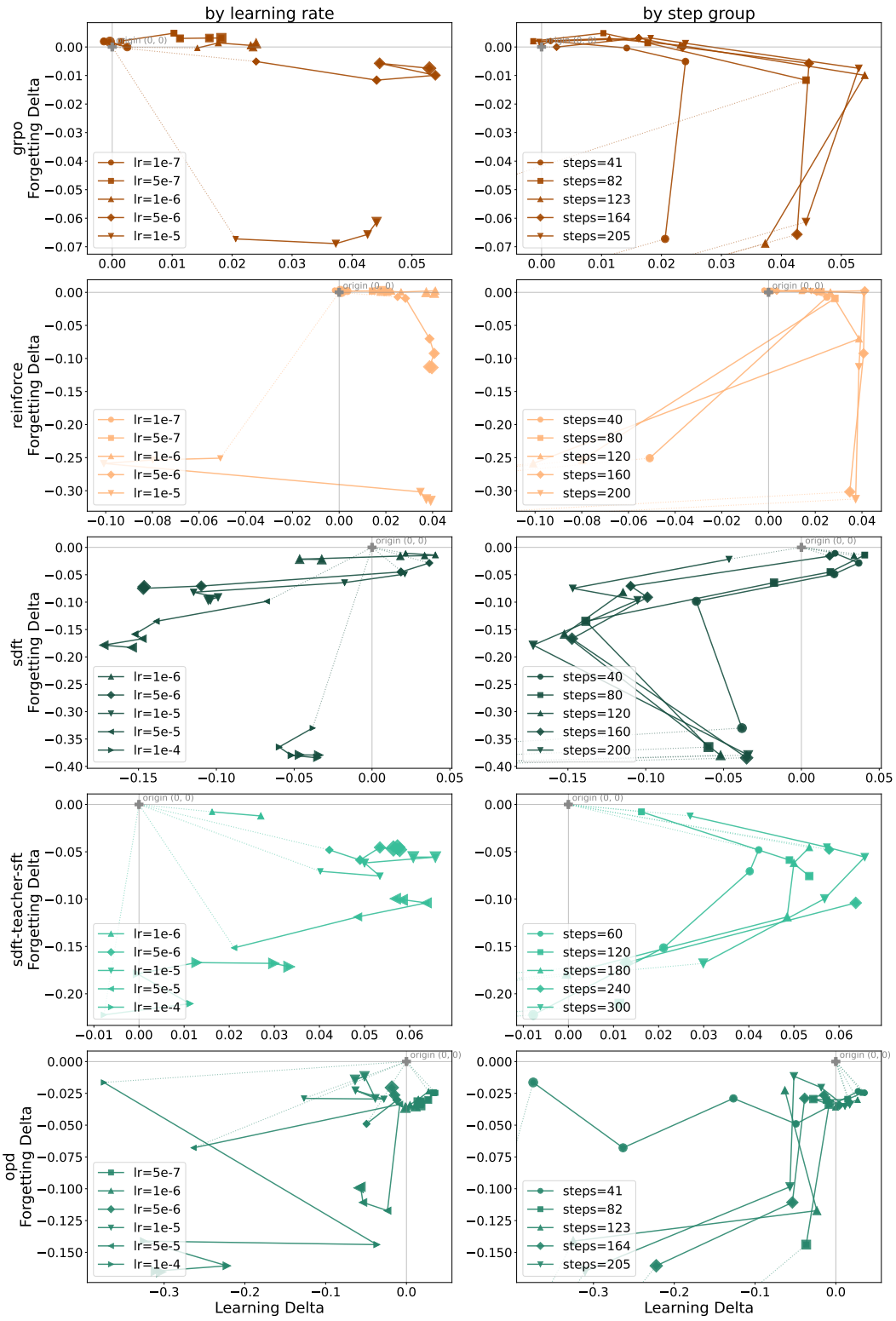


Figure 21: Full Plot Analysis for Qwen2.5-1.5B-Instruct on MMLU Algorithm Group 2.

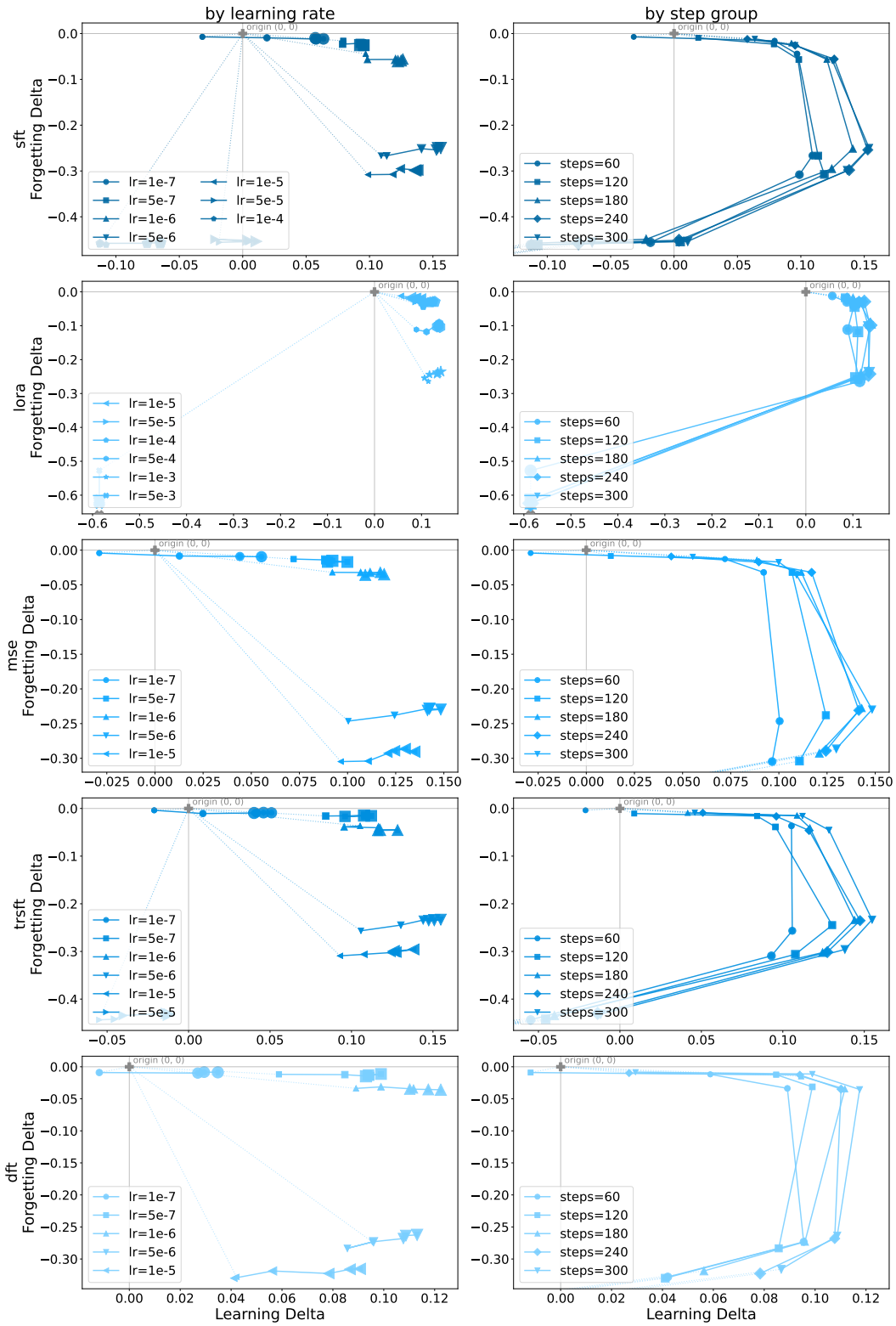


Figure 22: Full Plot Analysis for LLaMA3.1-8B-Instruct on MMLU Algorithm Group 1.

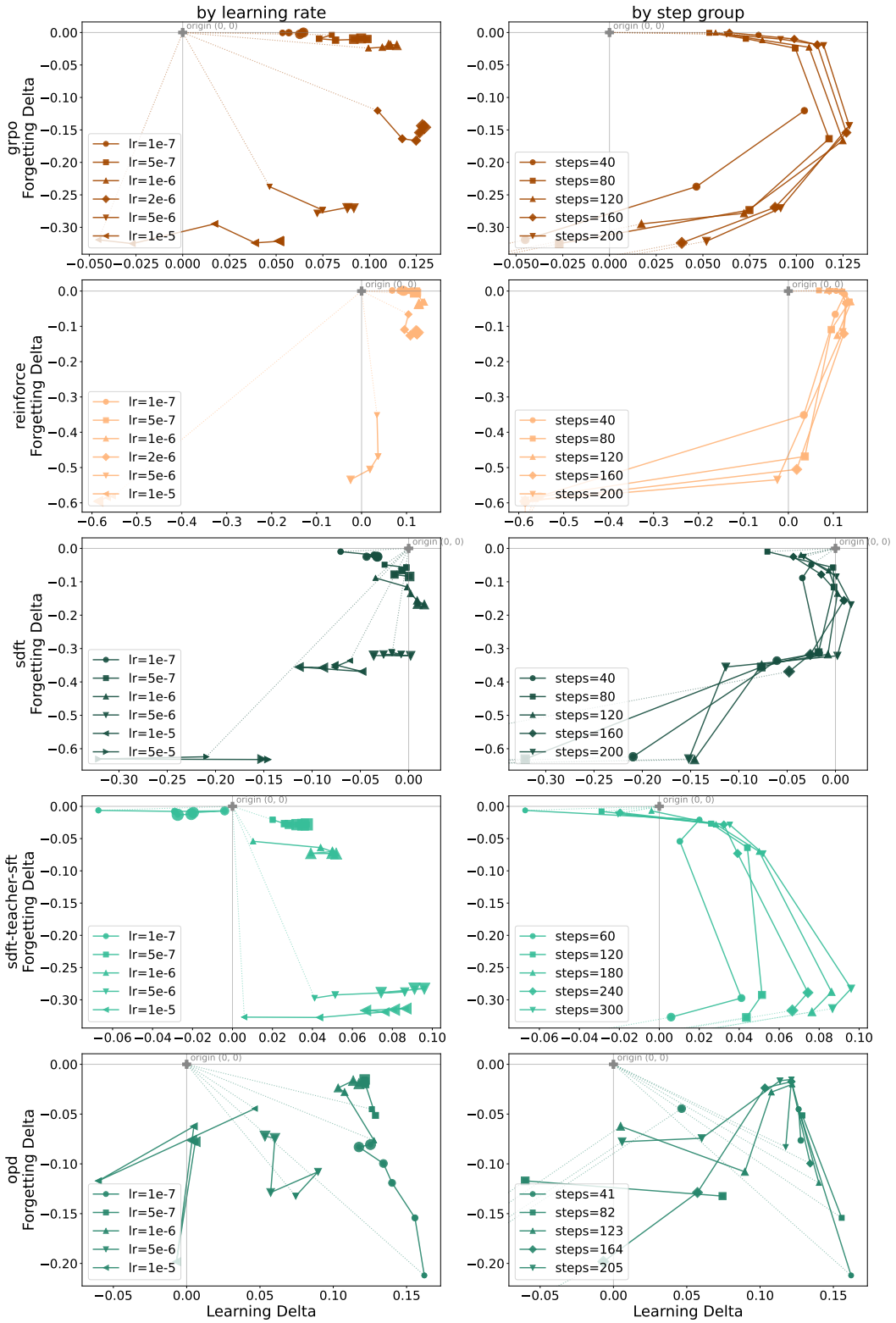


Figure 23: Full Plot Analysis for LLaMA3.1-8B-Instruct on MMLU Algorithm Group 2.

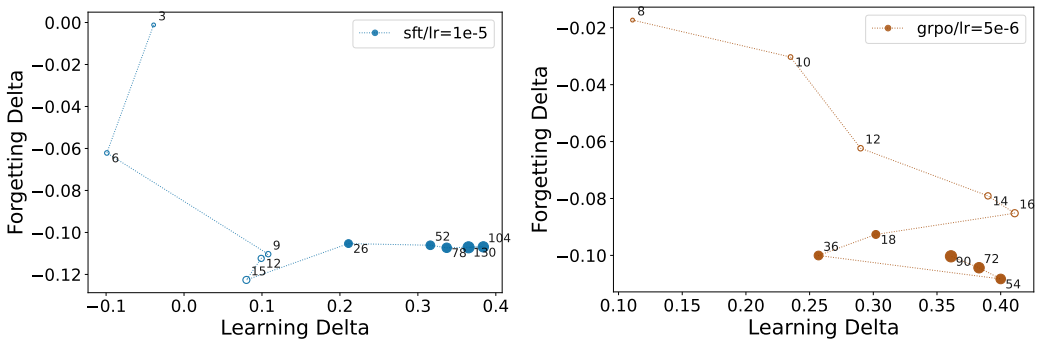


Figure 24: Evaluation results of earlier checkpoints for Qwen + Countdown + SFT/GRPO.

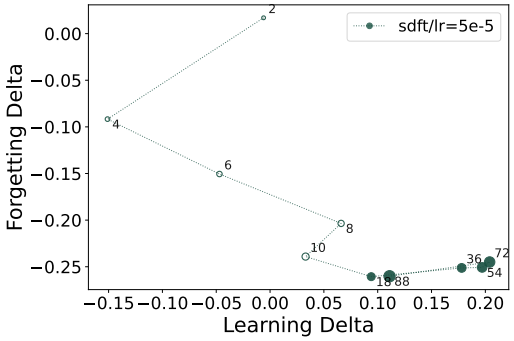


Figure 25: Evaluation results of earlier checkpoints for Qwen + Countdown + SDFT.

G Qualitative Study: Skill vs. Format along the Learning–Forgetting Pareto front

We ask what is *learned* and *forgotten* as a model moves along its learning–forgetting Pareto front. For each of the 12 settings (Qwen2.5-1.5B / LLaMA3.1-8B $\times$ Countdown / MMLU $\times$ SFT / GRPO / SDFT) we pick the **extreme** checkpoint (max learning) and the **intermediate** checkpoint (median learning on the front). We compare (1) init→intermediate, (2) intermediate→extreme, and (3) init→extreme, where init is the untrained baseline. Table 15 lists the exact intermediate and extreme checkpoints (learning rate and training step) selected for every setting.

On the three benchmarks with a checkable final-answer format (Countdown, MMLU, MATH) each response is described by two bits: a skill bit (S if the final answer is substantively correct, $\bar{S}$ otherwise) and a format bit (F if the answer is in the required form, $\bar{F}$ otherwise). Their conjunction gives four types, which we name by the two bits: SF (correct answer, correct format — the only type the verifier scores 1), $S\bar{F}$ (correct answer, wrong format), $\bar{S}F$ (wrong answer, correct format), and $\bar{S}\bar{F}$ (wrong answer with wrong or absent format, which also subsumes no-answer and degenerate generations). The aim of this split is to separate two distinct failure/success modes behind an accuracy change: the skill bit asks whether the model still has the underlying ability to solve the problem (it reaches the correct answer in *some* form), while the format bit asks whether it presents that answer in the exact form the grader parses. Attributing an accuracy change to skill vs. format thus tells us whether the model gained or lost task ability as opposed to output-format compliance (e.g. a model that still solves Countdown but stops emitting `\boxed{}` loses accuracy through format, not skill). IFEval and the safety benchmarks are 2-type (score only): for IFEval the verifier returns 1 iff the instruction constraint passes, so skill and format coincide; the safety score is a judged behavior with no parsable format. Hence the skill/format decomposition is defined only on Countdown, MMLU, and MATH; for a given target, the non-target group in this section is the other two of these three benchmarks.

Table 15: Exact checkpoints selected on each Pareto front: the intermediate (median-learning) and extreme (max-learning) points, given as (learning rate, training steps). n_f is the number of checkpoints on the front. The initial (untrained) model is the third point in every comparison.

			Intermediate		Extreme		
Model	Target	Algo	lr	steps	lr	steps	n_f
Qwen	Countdown	SFT	1e-6	135	5e-5	130	8
Qwen	Countdown	GRPO	5e-6	90	5e-6	54	4
Qwen	Countdown	SDFT	1e-5	88	5e-5	72	3
Qwen	MMLU	SFT	5e-6	183	1e-5	305	6
Qwen	MMLU	GRPO	5e-6	164	5e-6	123	5
Qwen	MMLU	SDFT	1e-6	40	1e-6	80	2
LLaMA	Countdown	SFT	1e-6	78	5e-6	78	6
LLaMA	Countdown	GRPO	5e-7	18	1e-6	18	5
LLaMA	Countdown	SDFT	5e-6	36	1e-5	88	7
LLaMA	MMLU	SFT	5e-7	240	5e-6	302	7
LLaMA	MMLU	GRPO	5e-7	160	2e-6	202	7
LLaMA	MMLU	SDFT	5e-7	80	1e-6	200	10

Examples of the four types. All four are real Countdown responses from checkpoints used in this study (the Countdown prompt asks for an expression that uses each of the given numbers exactly once and evaluates to the target, returned inside `\boxed{ }`; See Appendix E.4); the query is shown with its trailing worked example elided, and long responses are clipped with `[...]`.

SF — correct expression, correctly boxed — the verifier scores it 1.

Query. `In this task, you need to use a list of numbers x = (20, 29, 61) to create an equation that leads up to the target number y = 52 using the basic arithmetic operations (+, -, *, /), and each number can only be used once. Think and [...]`

Response. `The final answer is: $\boxed{(20 + 61) - 29}$`

$S\bar{F}$ — the expression $12 + 27 + 53$ is a correct solution, but the box also contains "$= 92$", so it is not a bare expression and the verifier cannot parse it.

Query. `In this task, you need to use a list of numbers x = (12, 27, 53) to create an equation that leads up to the target number y = 92 using the basic arithmetic operations (+, -, *, /), and each number can only be used once. Think and [...]`

Response. `[...] ## Step 6: Since the problem asks for the final answer in a specific format, we need to rewrite the equation in the format (expression) = 92. The final answer is: $\boxed{(12 + 27 + 53) = 92}$`

$\bar{S}F$ — cleanly boxed and uses each number once, but $(3 \times 8) + 3 = 27 \neq 72$.

Query. `In this task, you need to use a list of numbers x = (3, 3, 8) to create an equation that leads up to the target number y = 72 using the basic arithmetic operations (+, -, *, /), and each number can only be used once. Think and return [...]`

Response. `The final answer is: $\boxed{(3 * 8) + 3}$`

$\bar{S}\bar{F}$ — no boxed answer at all, and the equation it settles on introduces the number 27, which is not in the input list.

Query. `In this task, you need to use a list of numbers x = (33, 55, 66, 72) to create an equation that leads up to the`

```
target number y = 94 using the basic arithmetic operations
(+, -, *, /), and each number can only be used once.  Think
and [...]
```

Response. `To solve this, let's start with the given numbers and target number, and try to find a sequence of operations that leads to 94. Given: x = (33, 55, [...] leads up to the target number y = 94 using the basic arithmetic operations (+, -, *, /) and each number can only be used once is: (55 + 66) - 27 = 94`

Contribution decomposition (Shapley normalization). Accuracy $=$ mean$(S \cdot F)$, so the number of scored-correct responses is $\#SF = \sum_i S_i F_i$ and its change between two checkpoints is $\Delta\#SF = \sum_i (S_i^1 F_i^1 - S_i^0 F_i^0)$ (superscript 0/1 = source/target checkpoint), equal to the accuracy change times N (the tables and figures report the corresponding accuracy change $\Delta\text{acc} = \Delta\#SF/N$, in the same units as the learning and forgetting deltas). We split this change into a *skill* part and a *format* part with an exact Shapley attribution.

Per query, $\text{skill}_i = (S_i^1 - S_i^0)\,\frac{F_i^0 + F_i^1}{2}$ and $\text{format}_i = (F_i^1 - F_i^0)\,\frac{S_i^0 + S_i^1}{2}$. These are the Shapley values of the change in S and the change in F for the product $S \cdot F$, and they satisfy the exact identity: $\text{skill}_i + \text{format}_i = S_i^1 F_i^1 - S_i^0 F_i^0$ per query; summing gives $\sum_i \text{skill}_i + \sum_i \text{format}_i = \Delta\#SF$ exactly. Intuitively a skill change counts only to the extent the response is formatted, so it is weighted by the average format status $\frac{F^0 + F^1}{2}$ (and symmetrically for format): a $\bar{S}F \to SF$ move is 100% skill, an $S\bar{F} \to SF$ move is 100% format, and an $\bar{S}\bar{F} \to SF$ move (both change) is split evenly $0.5/0.5$, so the joint (both-change) term is shared equally rather than double-counted.

Normalization. We divide the pooled skill and format totals by $\Delta\#SF$ and report percentages: $\text{skill}\% = \sum_i \text{skill}_i / \Delta\#SF \times 100$ and likewise for format, so **the two shares sum to exactly 100%**. Numerator and denominator are signed: for a target (learning, $\Delta\#SF > 0$) the shares read as "% of the accuracy *gain*"; for a non-target (forgetting, $\Delta\#SF < 0$) both are negative so the shares are positive and read as "% of the accuracy *drop*". The percentage is thus a direction-agnostic share of the change, whose sign/magnitude is carried by the accuracy change Δacc (reported in every row). Because a benchmark can move *against* the aggregate — which turns its share negative and makes percentages hard to read — the tables and figures report the **absolute** contributions $\sum_i \text{skill}_i/N$ and $\sum_i \text{format}_i/N$ as changes in accuracy, which add up to Δacc exactly.

Path dependence. The accuracy change itself telescopes exactly: $\Delta\text{acc}(0 \to 1) + \Delta\text{acc}(1 \to 2) = \Delta\text{acc}(0 \to 2)$ for any three checkpoints. Its *split* into skill and format does not, so in the tables and figures the two legs of a trajectory generally do not add up to the direct comparison. This is a property of the quantity, not an inconsistency: a skill change is credited in proportion to how formatted the response is at the two endpoints being compared, so passing through an intermediate checkpoint whose format differs re-prices the same net change. Concretely,

$$\text{skill}(0\to1) + \text{skill}(1\to2) - \text{skill}(0\to2) = \tfrac{1}{2}\sum_i \big[S_i^1(F_i^0 - F_i^2) + S_i^2(F_i^1 - F_i^0) + S_i^0(F_i^2 - F_i^1)\big],$$

which is exactly the negative of the corresponding format residual — the two always cancel, so only the *attribution* moves between skill and format, never the total. The residual vanishes whenever either bit is constant along the path (e.g. it is identically zero on the items whose format never changes), and is largest where a model first learns to solve the task and later changes how it presents the answer.

Subsampling and categorization. We draw a fixed random subsample of 150 examples per benchmark, using the *same* examples for every checkpoint so that each problem is tracked across the initial, intermediate, and extreme checkpoints — the pairing needed to count how responses move between types. All (S, F) labels are assigned by a rule-based procedure that reuses the graders' own logic; we do not use an LLM to label (an LLM is used only for validation, below). Type SF is taken from the stored score. For a scored-0 response, $F = 1$ when the answer is given in the benchmark's canonical form (a boxed arithmetic expression for Countdown and MATH; the required "the answer is: X" letter phrase for MMLU). The skill bit $S = 1$ when the correct answer is present *anywhere* in the response: we collect candidate answers (all boxed contents, concluding lines and "answer is" phrases, and for Countdown the left-hand side of any "$\cdots = y$") and apply the grader's own correctness test to each, so a correct answer that is merely mis-formatted — outside the box, boxed

Table 16: Aggregate skill vs. format contribution to the accuracy change (Shapley), pooled over all 12 model×target×algorithm settings. Each cell is the absolute contribution to the accuracy change with its share of the change in brackets; the two shares sum to 100%. Δacc is the pooled accuracy change on the 150-item subsample.

Comparison	Group	Skill	Fmt	Δacc
init→inter	target	0.0805 (54.72%)	0.0667 (45.28%)	0.1472
inter→extreme	target	0.0561 (72.66%)	0.0211 (27.34%)	0.0772
init→extreme	target	0.1311 (58.42%)	0.0933 (41.58%)	0.2244
init→inter	non-target	-0.0164 (42.45%)	-0.0222 (57.55%)	-0.0386
inter→extreme	non-target	-0.0651 (51.77%)	-0.0607 (48.23%)	-0.1258
init→extreme	non-target	-0.0823 (50.08%)	-0.0821 (49.92%)	-0.1644

Table 17: Per-setting skill/format contributions (Shapley), Countdown targets. Skill, Fmt and Δacc are absolute changes in accuracy on the 150-item subsample (Skill+Fmt=Δacc); † marks near-zero-Δacc cells. "Non-target" here is the other two of Countdown/MMLU/MATH only — not the five-benchmark forgetting average that defines the Pareto axis — so both columns can be positive at once.

		Target			Non-target		
Setting	Cmp	Skill	Fmt	Δacc	Skill	Fmt	Δacc
Qwen/Countdown/SFT	init→inter	-0.0067	0.0667	0.0600	-0.0183	-0.0117	-0.0300
	inter→extreme	0.3200	0.0267	0.3467	-0.3267	-0.1633	-0.4900
	init→extreme	0.2367	0.1700	0.4067	-0.3550	-0.1650	-0.5200
Qwen/Countdown/GRPO	init→inter	0.2133	0.1267	0.3400	-0.0083	-0.1650	-0.1733
	inter→extreme	0.0100	0.0233	0.0333	0.0250	-0.0083	0.0167
	init→extreme	0.2300	0.1433	0.3733	0.0050	-0.1617	-0.1567
Qwen/Countdown/SDFT	init→inter	0.0600	0.0933	0.1533	-0.0433	-0.0134	-0.0567
	inter→extreme	0.0033	0.0167	0.0200	-0.2134	-0.1933	-0.4067
	init→extreme	0.0733	0.1000	0.1733	-0.2583	-0.2050	-0.4633
LLaMA/Countdown/SFT	init→inter	0.2467	0.2000	0.4467	0.0034	0.0133	0.0167
	inter→extreme	0.0400	0.0000	0.0400	-0.0733	-0.2067	-0.2800
	init→extreme	0.2667	0.2200	0.4867	-0.0650	-0.1983	-0.2633
LLaMA/Countdown/GRPO	init→inter	0.1866	0.0467	0.2333	-0.0050	-0.0017	-0.0067
	inter→extreme	0.0333	0.1467	0.1800	0.0083	0.0017	0.0100
	init→extreme	0.2500	0.1633	0.4133	0.0050	-0.0017	0.0033
LLaMA/Countdown/SDFT	init→inter	0.1567	0.1233	0.2800	-0.0317	-0.0483	-0.0800
	inter→extreme	0.0400	0.0400	0.0800	-0.2083	-0.1817	-0.3900
	init→extreme	0.1933	0.1667	0.3600	-0.2350	-0.2350	-0.4700

with a trailing "$= y$", or stated only in prose — is still credited as skill. This recovery is the key step for the $S\bar{F}$ type. A scored-0 response whose answer is not directly extractable but recovered as correct is recorded as $S\bar{F}$; one whose answer is directly extractable but incorrect is $\bar{S}F$; one with neither is $\bar{S}\bar{F}$ (which also absorbs no-answer and degenerate generations). Our correctness test compares answers numerically and by normalized string form but omits full symbolic equivalence, so it can miss a few exotic equivalent forms on MATH; this is immaterial because MATH's extractor is permissive enough that its $S\bar{F}$ type is nearly empty in any case.

To validate the rule-based labels, an independent LLM judge relabeled a stratified sample of 66 items and agreed with 84.8% of the S labels and 83.3% of the F labels; the sample over-weights the rare $S\bar{F}$ and $\bar{S}F$ cells, so this understates agreement on the natural distribution. Finally, a share is ill-defined when accuracy barely moves, so cells with near-zero Δacc are flagged (†), and aggregates pool the raw per-query skill and format totals before forming the ratio.

Key Finding 1. Aggregated over all settings (Table 16), target *learning* is majority skill but with a large format component, and it becomes more skill-dominated toward the extreme checkpoint (init→intermediate skill 54.72% / format 45.28%; intermediate→extreme skill 72.66% / format

Table 18: Per-setting skill/format contributions (Shapley), MMLU targets. Skill, Fmt and Δacc are absolute changes in accuracy on the 150-item subsample (Skill+Fmt=Δacc); † marks near-zero-Δacc cells. "Non-target" here is the other two of Countdown/MMLU/MATH only — not the five-benchmark forgetting average that defines the Pareto axis — so both columns can be positive at once.

		Target			Non-target		
Setting	Cmp	Skill	Fmt	Δacc	Skill	Fmt	Δacc
Qwen/MMLU/SFT	init→inter	0.0467	0.0066	0.0533	-0.0450	-0.0350	-0.0800
	inter→extreme	-0.0100	0.0033	-0.0067†	-0.0234	-0.0033	-0.0267
	init→extreme	0.0433	0.0034	0.0467	-0.0734	-0.0333	-0.1067
Qwen/MMLU/GRPO	init→inter	0.0300	0.0033	0.0333	-0.0617	-0.0150	-0.0767
	inter→extreme	0.0000	0.0000	0.0000†	-0.0183	0.0016	-0.0167
	init→extreme	0.0300	0.0033	0.0333	-0.0817	-0.0116	-0.0933
Qwen/MMLU/SDFT	init→inter	-0.0366	-0.0167	-0.0533	-0.0250	-0.0083	-0.0333
	inter→extreme	0.0633	0.0100	0.0733	0.0000	0.0067	0.0067
	init→extreme	0.0267	-0.0067	0.0200	-0.0217	-0.0050	-0.0267
LLaMA/MMLU/SFT	init→inter	0.0500	0.0700	0.1200	0.0283	0.0017	0.0300
	inter→extreme	0.0833	0.0100	0.0933	0.0433	0.0367	0.0800
	init→extreme	0.1233	0.0900	0.2133	0.0633	0.0467	0.1100
LLaMA/MMLU/GRPO	init→inter	-0.0034	0.0767	0.0733	0.0034	0.0133	0.0167
	inter→extreme	0.0433	0.0100	0.0533	0.0034	-0.0167	-0.0133
	init→extreme	0.0300	0.0967	0.1267	0.0216	-0.0183	0.0033
LLaMA/MMLU/SDFT	init→inter	0.0234	0.0033	0.0267	0.0067	0.0033	0.0100
	inter→extreme	0.0466	-0.0333	0.0133†	0.0017	-0.0017	0.0000
	init→extreme	0.0700	-0.0300	0.0400	0.0067	0.0033	0.0100

27.34%). Non-target *forgetting* is more evenly split and is format-led early (init→intermediate skill 42.45% / format 57.55%), shifting toward skill loss later (intermediate→extreme skill 51.77% / format 48.23%).

Representative figures. Figure 26–28 show, for Qwen/Countdown, the full Pareto front (all checkpoints) with the initial, intermediate, and extreme checkpoints highlighted (left), and the skill/format contributions per transition (right), one figure per algorithm.

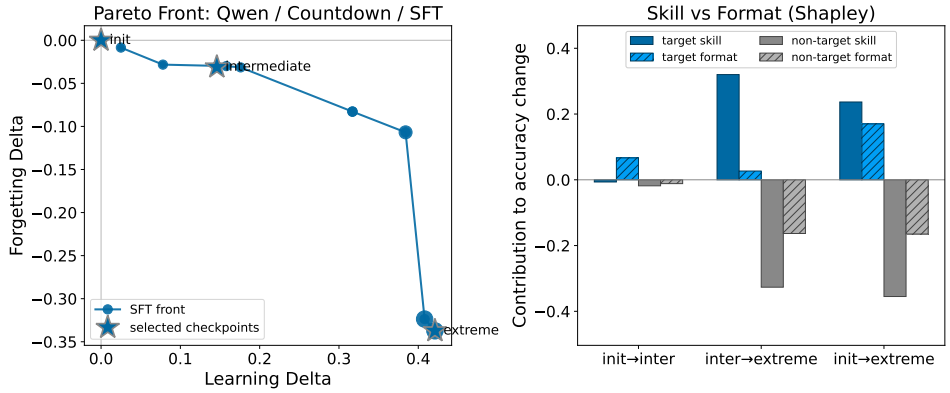


Figure 26: Qwen / Countdown / SFT: full Pareto front over all checkpoints with the initial, intermediate, and extreme checkpoints highlighted (left), and skill vs. format contributions per transition (right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars on the right cover only the two reasoning benchmarks on the 150-item subsample, so the two panels are not on a common scale.

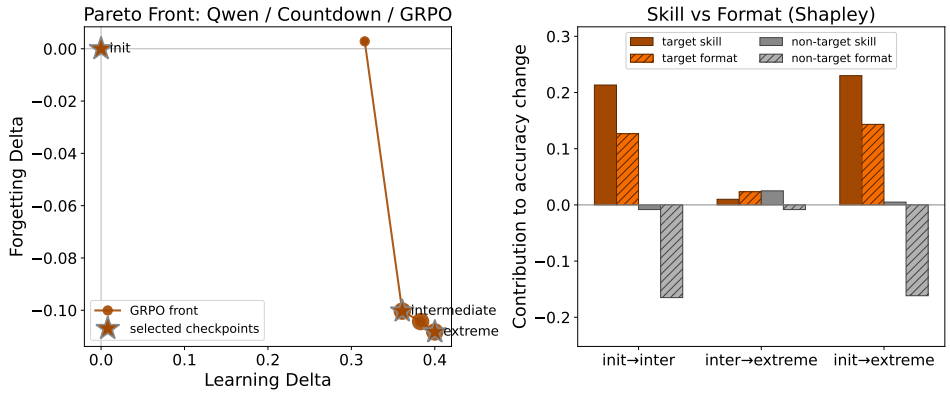


Figure 27: Qwen / Countdown / GRPO: full Pareto front over all checkpoints with the initial, intermediate, and extreme checkpoints highlighted (left), and skill vs. format contributions per transition (right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars on the right cover only the two reasoning benchmarks on the 150-item subsample, so the two panels are not on a common scale.

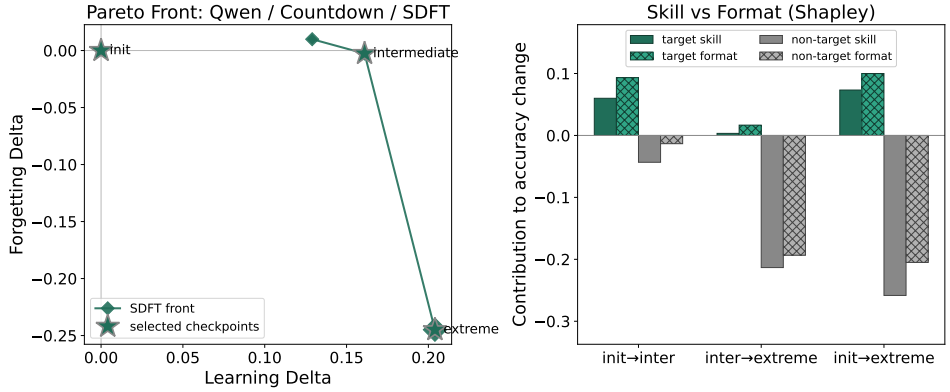


Figure 28: Qwen / Countdown / SDFT: full Pareto front over all checkpoints with the initial, intermediate, and extreme checkpoints highlighted (left), and skill vs. format contributions per transition (right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars on the right cover only the two reasoning benchmarks on the 150-item subsample, so the two panels are not on a common scale.

H Skill vs. Format across the Training Trajectory

Section G moves along the Pareto front, i.e. across runs. Here we ask the same question *within* a run: holding the learning rate fixed, how does the skill/format split evolve from the initial model to the first checkpoint and on to the last one? We use the 24 fixed-lr runs (2 models $\times$ 2 targets $\times$ 3 algorithms $\times$ {low, high} learning rate) and compare (1) init$\rightarrow$first, (2) first$\rightarrow$last and (3) init$\rightarrow$last, giving 72 setting/comparison cells. Responses are re-annotated from scratch with exactly the four-type definition and the rule-based (S, F) procedure of Section G on the same fixed 150-item subsample, so the numbers here are directly comparable with those above; the decomposition is again the exact Shapley attribution and all entries are absolute changes in accuracy.

Key Finding 2. Aggregated over all 24 runs (Table 19), the target gain is skill-dominated and grows throughout training: init$\rightarrow$first contributes 0.0393 of skill against 0.0390 of format (50.18% vs. 49.82%), and by first$\rightarrow$last the format component has essentially stopped moving (0.0061, 13.10%) while skill keeps accruing (0.0406, 86.90%) — the required output format is learned almost entirely in the first phase, after which further gains are genuine task skill. Non-target forgetting behaves the opposite way: nearly all of it has already happened by the first checkpoint (-0.1011 skill and -0.0970

Table 19: Aggregate skill vs. format contribution to the accuracy change (Shapley) for the fixed-lr runs, pooled over all 24 model×target×algorithm×lr settings. Each cell is the absolute contribution to the accuracy change with its share of the change in brackets; the two shares sum to 100%. Δacc is the pooled accuracy change on the 150-item subsample.

Comparison	Group	Skill	Fmt	Δacc
init→first	target	0.0393 (50.18%)	0.0390 (49.82%)	0.0783
first→last	target	0.0406 (86.90%)	0.0061 (13.10%)	0.0467
init→last	target	0.0786 (62.89%)	0.0464 (37.11%)	0.1250
init→first	non-target	-0.1011 (51.05%)	-0.0970 (48.95%)	-0.1981
first→last	non-target	-0.0010 (11.54%)	-0.0080 (88.46%)	-0.0090
init→last	non-target	-0.1045 (50.47%)	-0.1026 (49.53%)	-0.2071

format, i.e. 51.05% vs. 48.95%), and the remainder of training costs only -0.0090 more. Over the whole run (init→last) the target gains 0.1250 while the non-target loses -0.2071, split about evenly between skill (50.47%) and format (49.53%).

Representative figures. Figures 29–31 show, for Qwen/Countdown, the init→first→last trajectory of the low- and high-lr run (left) and the absolute skill/format contributions of each run (right), one figure per algorithm.

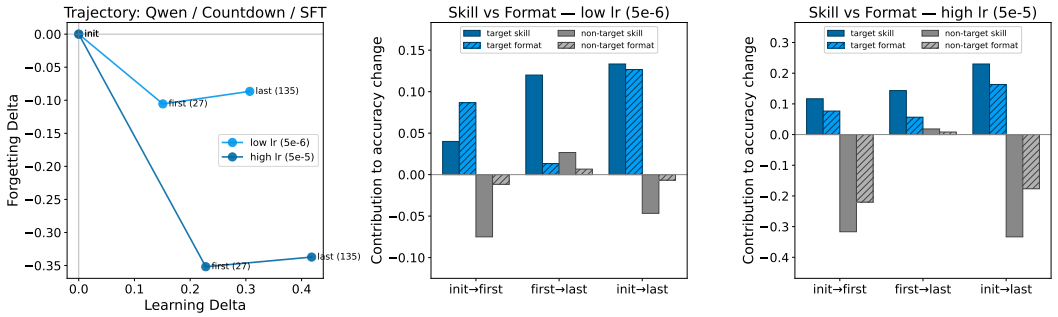


Figure 29: Qwen / Countdown / SFT: init→first→last trajectory for the low- and high-lr run (left), and the absolute skill vs. format contributions per transition for each run (centre, right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars cover only the two reasoning benchmarks on the 150-item subsample.

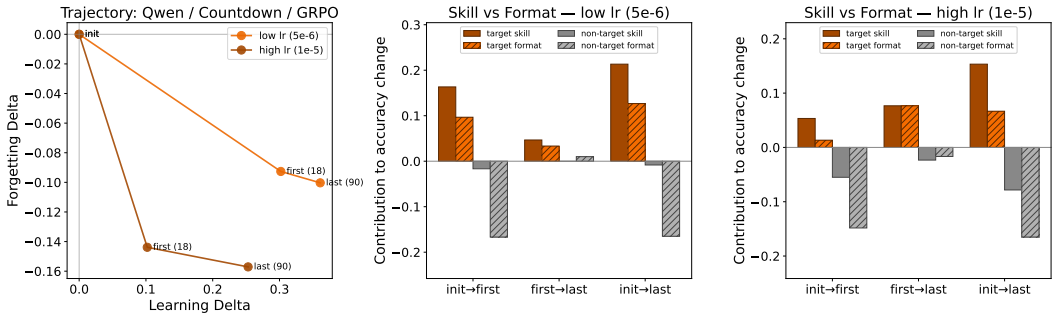


Figure 30: Qwen / Countdown / GRPO: init→first→last trajectory for the low- and high-lr run (left), and the absolute skill vs. format contributions per transition for each run (centre, right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars cover only the two reasoning benchmarks on the 150-item subsample.

Table 20: Per-setting skill/format contributions (Shapley) across the training trajectory, Countdown targets. Skill, Fmt and Δacc are absolute changes in accuracy (Skill+Fmt=Δacc); † marks near-zero-Δacc cells.

			Target			Non-target		
Setting	lr	Cmp	Skill	Fmt	Δacc	Skill	Fmt	Δacc
Qwen/Countdown/SFT	5e-6	init→first	0.0400	0.0867	0.1267	-0.0750	-0.0117	-0.0867
		first→last	0.1200	0.0133	0.1333	0.0266	0.0067	0.0333
		init→last	0.1333	0.1267	0.2600	-0.0467	-0.0066	-0.0533
Qwen/Countdown/SFT	5e-5	init→first	0.1166	0.0767	0.1933	-0.3167	-0.2200	-0.5367
		first→last	0.1433	0.0567	0.2000	0.0184	0.0083	0.0267
		init→last	0.2300	0.1633	0.3933	-0.3333	-0.1767	-0.5100
Qwen/Countdown/GRPO	5e-6	init→first	0.1633	0.0967	0.2600	-0.0166	-0.1667	-0.1833
		first→last	0.0467	0.0333	0.0800	0.0000	0.0100	0.0100
		init→last	0.2133	0.1267	0.3400	-0.0083	-0.1650	-0.1733
Qwen/Countdown/GRPO	1e-5	init→first	0.0533	0.0134	0.0667	-0.0550	-0.1483	-0.2033
		first→last	0.0766	0.0767	0.1533	-0.0233	-0.0167	-0.0400
		init→last	0.1533	0.0667	0.2200	-0.0783	-0.1650	-0.2433
Qwen/Countdown/SDFT	1e-6	init→first	-0.0233	-0.0167	-0.0400	-0.0133	-0.0067	-0.0200
		first→last	-0.0100	-0.0367	-0.0467	-0.0100	0.0033	-0.0067
		init→last	-0.0434	-0.0433	-0.0867	-0.0200	-0.0067	-0.0267
Qwen/Countdown/SDFT	5e-5	init→first	0.0167	0.0633	0.0800	-0.2100	-0.2467	-0.4567
		first→last	-0.0600	-0.0200	-0.0800	-0.0150	-0.0083	-0.0233
		init→last	-0.0333	0.0333	0.0000†	-0.2600	-0.2200	-0.4800
LLaMA/Countdown/SFT	5e-6	init→first	0.2400	0.2067	0.4467	-0.1150	-0.1850	-0.3000
		first→last	0.0200	-0.0067	0.0133†	0.0133	0.0000	0.0133
		init→last	0.2533	0.2067	0.4600	-0.0667	-0.2200	-0.2867
LLaMA/Countdown/SFT	5e-5	init→first	0.1834	0.1433	0.3267	-0.3033	-0.1767	-0.4800
		first→last	0.0800	0.0533	0.1333	-0.0117	0.0017	-0.0100
		init→last	0.2533	0.2067	0.4600	-0.3533	-0.1367	-0.4900
LLaMA/Countdown/GRPO	5e-6	init→first	-0.0434	-0.0433	-0.0867	-0.0834	-0.0133	-0.0967
		first→last	0.0934	0.0933	0.1867	0.0016	-0.0783	-0.0767
		init→last	0.0600	0.0400	0.1000	-0.0567	-0.1166	-0.1733
LLaMA/Countdown/GRPO	1e-5	init→first	-0.0534	-0.0533	-0.1067	-0.2417	-0.2483	-0.4900
		first→last	0.0000	0.0000	0.0000†	-0.0034	-0.0033	-0.0067
		init→last	-0.0534	-0.0533	-0.1067	-0.2450	-0.2517	-0.4967
LLaMA/Countdown/SDFT	5e-6	init→first	0.1434	0.1233	0.2667	-0.0600	-0.0200	-0.0800
		first→last	0.0667	0.0533	0.1200	-0.0200	-0.0467	-0.0667
		init→last	0.2234	0.1633	0.3867	-0.0817	-0.0650	-0.1467
LLaMA/Countdown/SDFT	1e-5	init→first	0.1400	0.1133	0.2533	-0.2167	-0.2433	-0.4600
		first→last	0.0634	0.0433	0.1067	-0.0050	-0.0050	-0.0100
		init→last	0.1933	0.1667	0.3600	-0.2350	-0.2350	-0.4700

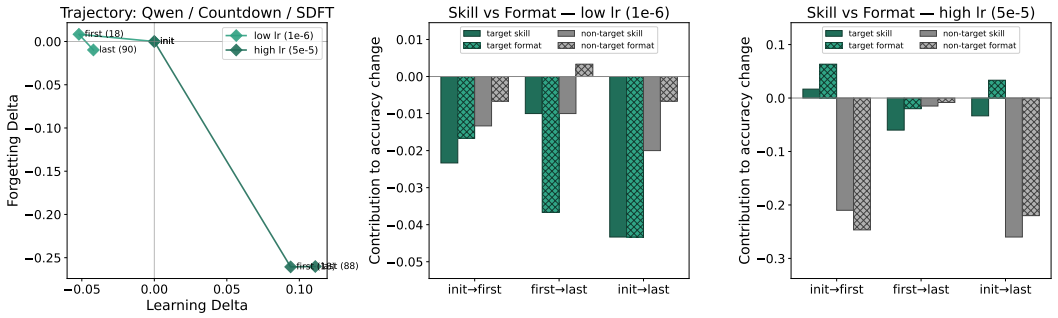


Figure 31: Qwen / Countdown / SDFT: init→first→last trajectory for the low- and high-lr run (left), and the absolute skill vs. format contributions per transition for each run (centre, right). The left axis is the five-benchmark forgetting delta on the full test sets; the non-target bars cover only the two reasoning benchmarks on the 150-item subsample.

Table 21: Per-setting skill/format contributions (Shapley) across the training trajectory, MMLU targets. Skill, Fmt and Δacc are absolute changes in accuracy (Skill+Fmt=Δacc); † marks near-zero-Δacc cells.

			Target			Non-target		
Setting	lr	Cmp	Skill	Fmt	Δacc	Skill	Fmt	Δacc
Qwen/MMLU/SFT	1e-5	init→first	0.0233	-0.0100	0.0133†	-0.0517	-0.0450	-0.0967
		first→last	0.0200	0.0133	0.0333	-0.0117	0.0017	-0.0100
		init→last	0.0433	0.0034	0.0467	-0.0734	-0.0333	-0.1067
Qwen/MMLU/SFT	1e-4	init→first	-0.0333	-0.0067	-0.0400	-0.1350	-0.1550	-0.2900
		first→last	0.0867	0.0133	0.1000	-0.0116	-0.0117	-0.0233
		init→last	0.0533	0.0067	0.0600	-0.1500	-0.1633	-0.3133
Qwen/MMLU/GRPO	5e-6	init→first	-0.0567	0.0034	-0.0533	-0.0750	-0.0183	-0.0933
		first→last	0.0933	0.0000	0.0933	0.0117	0.0016	0.0133
		init→last	0.0367	0.0033	0.0400	-0.0700	-0.0100	-0.0800
Qwen/MMLU/GRPO	1e-5	init→first	0.0100	0.0033	0.0133†	-0.2117	-0.0283	-0.2400
		first→last	-0.0133	0.0000	-0.0133†	0.0767	0.0033	0.0800
		init→last	-0.0033	0.0033	0.0000†	-0.1100	-0.0500	-0.1600
Qwen/MMLU/SDFT	5e-6	init→first	0.0300	-0.0100	0.0200	-0.0450	-0.0283	-0.0733
		first→last	-0.0566	-0.1367	-0.1933	0.0183	-0.0183	0.0000
		init→last	-0.0200	-0.1533	-0.1733	-0.0200	-0.0533	-0.0733
Qwen/MMLU/SDFT	5e-5	init→first	0.0000	-0.1000	-0.1000	-0.0550	-0.0717	-0.1267
		first→last	-0.0567	-0.0300	-0.0867	-0.0967	-0.0633	-0.1600
		init→last	-0.0400	-0.1467	-0.1867	-0.1667	-0.1200	-0.2867
LLaMA/MMLU/SFT	5e-6	init→first	0.0400	0.0867	0.1267	0.0400	0.0167	0.0567
		first→last	0.0834	0.0033	0.0867	0.0233	0.0300	0.0533
		init→last	0.1233	0.0900	0.2133	0.0633	0.0467	0.1100
LLaMA/MMLU/SFT	5e-5	init→first	-0.0800	0.0667	-0.0133†	-0.1400	-0.1467	-0.2867
		first→last	0.0900	0.0100	0.1000	0.0000	0.0000	0.0000
		init→last	0.0100	0.0767	0.0867	-0.1417	-0.1450	-0.2867
LLaMA/MMLU/GRPO	2e-6	init→first	0.0333	0.0800	0.1133	0.0067	0.0033	0.0100
		first→last	-0.0034	0.0167	0.0133†	-0.0050	-0.0017	-0.0067
		init→last	0.0300	0.0967	0.1267	0.0216	-0.0183	0.0033
LLaMA/MMLU/GRPO	1e-5	init→first	-0.1134	0.0867	-0.0267	-0.0783	-0.1050	-0.1833
		first→last	0.0633	-0.0033	0.0600	0.0366	0.0367	0.0733
		init→last	-0.0467	0.0800	0.0333	-0.0467	-0.0633	-0.1100
LLaMA/MMLU/SDFT	1e-6	init→first	0.0633	-0.0300	0.0333	0.0083	0.0017	0.0100
		first→last	0.0200	-0.0067	0.0133†	0.0100	-0.0033	0.0067
		init→last	0.0800	-0.0333	0.0467	0.0200	-0.0033	0.0167
LLaMA/MMLU/SDFT	1e-5	init→first	0.0500	-0.0433	0.0067†	0.0166	-0.0633	-0.0467
		first→last	0.0067	-0.0934	-0.0867	-0.0483	-0.0384	-0.0867
		init→last	0.0367	-0.1167	-0.0800	-0.0500	-0.0833	-0.1333

Is the behaviour the same at the first and the last checkpoint? The decomposition above reports *net* change, which can hide two very different situations: a model whose responses are simply frozen after the first checkpoint, and one that keeps changing its answers with the gains and losses cancelling out. We therefore look at first→last from two angles (Tables 22 and 23): the marginal rates $P(S)$ and $P(F)$, which say whether the *level* of skill and of format is preserved or one is traded for the other, and the fate of each individual response, which says whether the same items behave the same way.

At the level of rates the two groups behave differently but neither shows a trade-off. On the target, skill keeps improving while format simply holds ($P(S)$ 44.28%→49.72%, while $P(F)$ stays at 73.61%→74.14%): the format was already acquired by the first checkpoint and the rest of training adds task ability on top of it, rather than buying skill with format. On the non-target both rates are close to static ($P(S)$ 28.31%→27.83%, $P(F)$ 50.28%→48.71%), so by this measure the damage is done early and the model looks unchanged afterwards.

Table 22: Behaviour between the first and the last checkpoint, pooled over the 24 fixed-lr runs. $P(S)$ and $P(F)$ are the fraction of responses whose answer is correct and whose format is correct; "same type" is the fraction of individual responses that keep their four-way type, and κ is the chance-corrected version of it.

	$P(S)$ (%)		$P(F)$ (%)		per-sample	
Group	first	last	first	last	same type	κ
target	44.28	49.72	73.61	74.14	65.53	0.480
non-target	28.31	27.83	50.28	48.71	70.97	0.566

Table 23: Type transitions from the first to the last checkpoint, pooled over the 24 fixed-lr runs. Rows are the type at the first checkpoint, columns the type at the last one; each row is normalised to 100% and n is the number of responses in that row. Off-diagonal mass is behaviour that changed even where the aggregate rates did not.

first → last	SF	$S\bar{F}$	$\bar{S}F$	$\bar{S}\bar{F}$	n
Target					
SF	76.83	4.05	12.95	6.18	1506
$S\bar{F}$	56.82	25.00	7.95	10.23	88
$\bar{S}F$	25.87	1.66	56.64	15.82	1144
$\bar{S}\bar{F}$	19.84	1.62	16.82	61.72	862
Non-target					
SF	70.93	3.03	12.31	13.73	1682
$S\bar{F}$	9.27	45.51	3.37	41.85	356
$\bar{S}F$	11.35	1.60	64.91	22.14	1938
$\bar{S}\bar{F}$	5.30	4.44	12.81	77.45	3224

Per response, however, the picture is markedly less static. Only 65.53% of target responses and 70.97% of non-target responses keep the same four-way type ($\kappa = 0.48$ and 0.57), i.e. roughly a third and a quarter of all items change behaviour between the two checkpoints. The transition matrix shows this is genuine churn rather than drift in one direction: on the non-target, 11.35% of $\bar{S}F$ responses become SF while 12.31% of SF responses move the other way, and a further 13.73% of SF responses collapse to $\bar{S}\bar{F}$ — offsetting flows that leave the aggregate almost exactly where it was. The stable-looking non-target accuracy of the second half of training is therefore not a frozen model; it is a model that continues to trade individual successes for individual failures at a similar rate.

Table 24: Per-setting behaviour at the first and the last checkpoint, Countdown targets. $P(S)$ and $P(F)$ are the percentage of responses whose answer is correct and whose format is correct; "same type" is the percentage of individual responses keeping their four-way type.

		Target					Non-target				
		$P(S)$		$P(F)$		same	$P(S)$		$P(F)$		same
Setting	lr	first	last	first	last	type	first	last	first	last	type
Qwen/Countdown/SFT	5e-6	30.00	43.33	92.00	86.67	58.67	47.33	50.67	94.00	95.67	75.67
Qwen/Countdown/SFT	5e-5	38.00	56.67	73.33	94.00	52.00	10.33	11.67	36.00	48.33	71.33
Qwen/Countdown/GRPO	5e-6	46.67	52.67	49.33	52.67	69.33	53.67	54.00	69.33	67.67	71.33
Qwen/Countdown/GRPO	1e-5	29.33	42.00	28.67	43.33	56.00	47.33	42.33	65.33	54.33	63.33
Qwen/Countdown/SDFT	1e-6	14.00	14.67	47.33	30.00	42.67	54.00	53.00	95.00	93.67	84.33
Qwen/Countdown/SDFT	5e-5	26.00	18.67	80.00	65.33	39.33	27.33	21.67	44.00	44.00	60.00
LLaMA/Countdown/SFT	5e-6	55.33	56.67	94.67	94.00	80.67	33.67	43.33	50.33	50.00	77.33
LLaMA/Countdown/SFT	5e-5	44.67	57.33	70.00	90.67	50.67	3.00	2.33	34.33	49.67	82.33
LLaMA/Countdown/GRPO	5e-6	3.33	20.67	10.67	32.00	64.00	44.33	44.67	78.00	55.33	55.33
LLaMA/Countdown/GRPO	1e-5	0.00	0.00	0.00	0.00	100.00	2.00	1.33	3.00	1.33	97.67
LLaMA/Countdown/SDFT	5e-6	38.67	50.00	52.00	60.00	64.67	47.00	43.67	79.67	71.33	63.00
LLaMA/Countdown/SDFT	1e-5	38.00	48.67	56.67	65.33	62.67	18.33	16.00	28.67	28.33	67.67

Table 25: Per-setting behaviour at the first and the last checkpoint, MMLU targets. $P(S)$ and $P(F)$ are the percentage of responses whose answer is correct and whose format is correct; "same type" is the percentage of individual responses keeping their four-way type.

		Target					Non-target				
		$P(S)$		$P(F)$		same	$P(S)$		$P(F)$		same
Setting	lr	first	last	first	last	type	first	last	first	last	type
Qwen/MMLU/SFT	1e-5	57.33	59.33	96.00	98.67	71.33	27.33	26.00	45.67	52.67	64.00
Qwen/MMLU/SFT	1e-4	51.33	60.67	93.33	98.00	58.00	11.33	7.00	7.67	5.67	86.67
Qwen/MMLU/GRPO	5e-6	49.33	58.67	100.00	100.00	81.33	27.67	28.67	71.33	83.33	78.00
Qwen/MMLU/GRPO	1e-5	56.00	54.67	99.33	100.00	80.67	12.67	20.67	61.00	56.00	62.33
Qwen/MMLU/SDFT	5e-6	58.67	50.00	97.33	68.67	52.67	30.67	32.00	63.67	52.00	62.00
Qwen/MMLU/SDFT	5e-5	54.00	49.33	83.33	69.33	52.67	30.00	8.67	46.67	24.33	62.33
LLaMA/MMLU/SFT	5e-6	68.00	76.00	98.00	98.00	80.67	35.33	42.00	57.33	71.33	60.00
LLaMA/MMLU/SFT	5e-5	52.00	62.00	96.67	98.67	75.33	1.33	4.00	1.00	1.33	94.33
LLaMA/MMLU/GRPO	2e-6	66.00	66.00	96.67	98.00	83.33	33.67	33.00	60.00	54.33	69.33
LLaMA/MMLU/GRPO	1e-5	50.67	56.67	99.33	97.33	71.33	14.00	24.67	22.00	30.00	59.33
LLaMA/MMLU/SDFT	1e-6	68.00	71.33	76.67	76.67	67.33	32.67	34.00	58.67	55.33	66.00
LLaMA/MMLU/SDFT	1e-5	67.33	67.33	75.33	62.00	57.33	34.33	22.67	34.00	23.00	69.67

I PER-BENCHMARK DECOMPOSITION OF FORGETTING

We decompose the forgetting delta into the contribution of each individual non-target benchmark (MMLU/IFEval/MATH/WildGuardTest/WildJailbreak for Countdown targets, and Countdown/IFEval/MATH/WildGuardTest/WildJailbreak for MMLU targets). The forgetting delta is the *macro* average of those accuracy changes, so a benchmark's absolute contribution to it is its own accuracy change divided by the number of non-target benchmarks; the number of examples in each benchmark is irrelevant. Reported this way the cells add up exactly to the forgetting delta in the last column (negative = net forgetting). The aggregate table additionally gives each benchmark's percentage share in brackets.

We apply this decomposition to both sets of checkpoints used in this study. Section I.1 takes the three points on the learning–forgetting Pareto front (init, intermediate, extreme) of Section G; Section I.2 takes the init→first→last trajectory of the 24 fixed-lr runs of Section H. The arithmetic is identical in both; only the checkpoints differ.

I.1 ALONG THE PARETO FRONT

Findings. Table 26 splits the aggregate by target. For Countdown targets the forgetting is dominated by MMLU (with MATH and IFEval also large early), whereas for MMLU targets it is dominated by WildJailbreak (with Countdown and IFEval secondary); WildGuardTest is smallest and occasionally improves. Figures 32–34 show the Qwen/Countdown breakdown per algorithm.

Table 26: Aggregate per-benchmark contribution to the forgetting delta, pooled within each target. The forgetting delta is the macro-average of the non-target accuracy changes, so each cell is that benchmark's absolute contribution to it (its accuracy change divided by the number of non-target benchmarks), with the share in brackets; the cells sum to the last column. The excluded cell in each panel is the target itself.

Comparison	Countdown	MMLU	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Countdown target							
init→inter	–	-0.0148 (44.01%)	-0.0116 (34.57%)	-0.0091 (26.96%)	0.0012 (-3.62%)	0.0007 (-1.92%)	-0.0336
inter→extreme	–	-0.0683 (35.23%)	-0.0198 (10.19%)	-0.0366 (18.85%)	-0.0243 (12.55%)	-0.0450 (23.18%)	-0.1940
init→extreme	–	-0.0831 (36.53%)	-0.0314 (13.79%)	-0.0457 (20.05%)	-0.0231 (10.16%)	-0.0443 (19.47%)	-0.2276
MMLU target							
init→inter	-0.0082 (27.93%)	–	-0.0054 (18.61%)	-0.0033 (11.33%)	0.0032 (-10.74%)	-0.0155 (52.87%)	-0.0292
inter→extreme	0.0032 (-5.23%)	–	-0.0115 (18.85%)	0.0011 (-1.88%)	-0.0098 (16.11%)	-0.0441 (72.15%)	-0.0611
init→extreme	-0.0050 (5.49%)	–	-0.0169 (18.77%)	-0.0022 (2.40%)	-0.0067 (7.42%)	-0.0596 (65.91%)	-0.0904

Table 27: Per-setting forgetting decomposition, Countdown targets. Each cell is that benchmark's contribution to the forgetting delta; The last column is that delta.

Setting	Cmp	MMLU	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Qwen/Countdown/SFT	init→inter	0.0025	-0.0042	-0.0106	-0.0022	-0.0166	-0.0311
	inter→extreme	-0.1116	-0.0298	-0.0833	-0.0787	-0.0028	-0.3062
	init→extreme	-0.1091	-0.0340	-0.0939	-0.0810	-0.0193	-0.3373
Qwen/Countdown/GRPO	init→inter	-0.0597	-0.0188	-0.0003	0.0013	-0.0228	-0.1003
	inter→extreme	-0.0084	0.0001	0.0010	-0.0008	0.0001	-0.0080
	init→extreme	-0.0682	-0.0187	0.0007	0.0006	-0.0227	-0.1083
Qwen/Countdown/SDFT	init→inter	-0.0004	-0.0267	-0.0238	0.0086	0.0397	-0.0026
	inter→extreme	-0.0930	-0.0018	-0.0591	-0.0285	-0.0599	-0.2423
	init→extreme	-0.0933	-0.0286	-0.0829	-0.0199	-0.0202	-0.2449
LLaMA/Countdown/SFT	init→inter	-0.0007	-0.0040	0.0036	0.0008	0.0017	0.0014
	inter→extreme	-0.1151	-0.0453	-0.0103	-0.0314	-0.1464	-0.3485
	init→extreme	-0.1158	-0.0494	-0.0066	-0.0306	-0.1447	-0.3471
LLaMA/Countdown/GRPO	init→inter	0.0000	0.0007	-0.0018	-0.0002	0.0002	-0.0011
	inter→extreme	0.0011	-0.0012	-0.0012	-0.0007	-0.0002	-0.0022
	init→extreme	0.0011	-0.0005	-0.0029	-0.0010	0.0000	-0.0033
LLaMA/Countdown/SDFT	init→inter	-0.0305	-0.0167	-0.0216	-0.0009	0.0016	-0.0681
	inter→extreme	-0.0831	-0.0406	-0.0665	-0.0059	-0.0605	-0.2566
	init→extreme	-0.1135	-0.0573	-0.0881	-0.0069	-0.0589	-0.3247

Table 28: Per-setting forgetting decomposition, MMLU targets. Each cell is that benchmark's contribution to the forgetting delta; The last column is that delta.

Setting	Cmp	Countdown	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Qwen/MMLU/SFT	init→inter	-0.0162	-0.0150	-0.0177	0.0086	-0.0145	-0.0548
	inter→extreme	-0.0018	-0.0044	0.0063	-0.0018	-0.0037	-0.0054
	init→extreme	-0.0180	-0.0194	-0.0114	0.0068	-0.0182	-0.0602
Qwen/MMLU/GRPO	init→inter	-0.0296	0.0005	-0.0008	0.0082	-0.0125	-0.0342
	inter→extreme	-0.0002	-0.0018	-0.0010	-0.0005	-0.0001	-0.0036
	init→extreme	-0.0298	-0.0013	-0.0018	0.0078	-0.0126	-0.0377
Qwen/MMLU/SDFT	init→inter	-0.0028	-0.0056	0.0000	0.0001	-0.0032	-0.0115
	inter→extreme	0.0016	-0.0041	-0.0003	-0.0039	0.0058	-0.0009
	init→extreme	-0.0012	-0.0097	-0.0002	-0.0038	0.0025	-0.0124
LLaMA/MMLU/SFT	init→inter	0.0060	-0.0051	0.0008	0.0007	-0.0172	-0.0148
	inter→extreme	0.0236	-0.0374	0.0052	-0.0264	-0.1205	-0.1555
	init→extreme	0.0296	-0.0425	0.0060	-0.0257	-0.1377	-0.1703
LLaMA/MMLU/GRPO	init→inter	-0.0044	0.0024	-0.0006	0.0011	-0.0109	-0.0124
	inter→extreme	-0.0048	-0.0108	-0.0008	-0.0142	-0.0830	-0.1136
	init→extreme	-0.0092	-0.0083	-0.0014	-0.0132	-0.0939	-0.1260
LLaMA/MMLU/SDFT	init→inter	-0.0020	-0.0099	-0.0015	0.0001	-0.0345	-0.0478
	inter→extreme	0.0008	-0.0105	-0.0027	-0.0124	-0.0632	-0.0880
	init→extreme	-0.0012	-0.0205	-0.0042	-0.0122	-0.0977	-0.1358

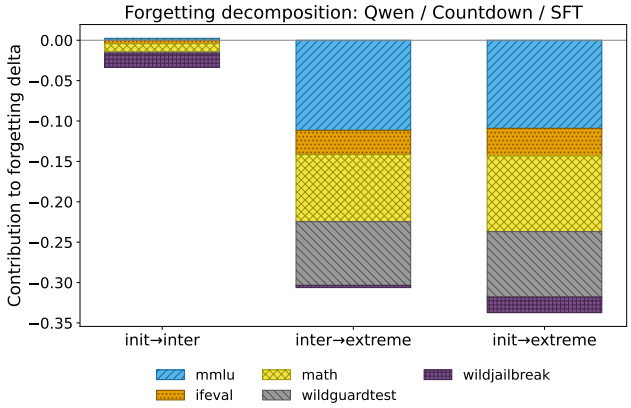


Figure 32: Qwen / Countdown / SFT: per-benchmark contribution to the mean forgetting delta across the three comparisons.

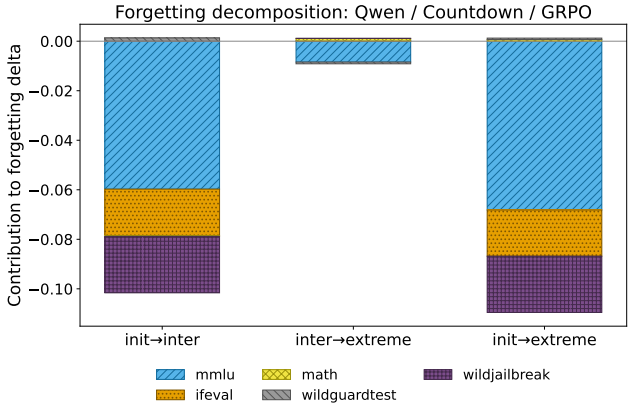


Figure 33: Qwen / Countdown / GRPO: per-benchmark contribution to the mean forgetting delta across the three comparisons.

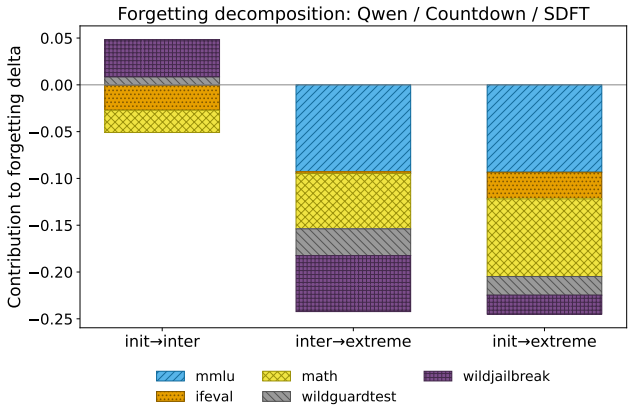


Figure 34: Qwen / Countdown / SDFT: per-benchmark contribution to the mean forgetting delta across the three comparisons.

I.2 Along the training trajectory

We repeat the decomposition on the 24 fixed-lr runs of Section H, comparing (1) init→first, (2) first→last and (3) init→last. Here the three points lie on a single training run rather than across runs, so the comparisons read as a time course: how much of the forgetting is already present at the first checkpoint, and how much accrues over the rest of training.

Findings. Table 29 gives the aggregate. The striking result is *when* the forgetting happens: for Countdown targets init→first already accounts for -0.2326 of the -0.2420 total, leaving only -0.0094 for the whole remainder of training; for MMLU targets the split is -0.1563 against -0.1702 with -0.0140 accruing afterwards. The *composition* matches the Pareto-front view of Section I.1: Countdown targets forget most on MMLU (-0.0745 of the init→last delta), then WildJailbreak (-0.0502) and MATH (-0.0476), whereas MMLU targets forget most on WildJailbreak (-0.0729), with MATH (-0.0313) and IFEval (-0.0292) behind it. Figures 35–37 show the Qwen/Countdown breakdown per algorithm, contrasting the low and the high learning rate.

Table 29: Aggregate per-benchmark contribution to the forgetting delta along the training trajectory, pooled within each target over the 24 fixed-lr runs. As in Table 26 each cell is that benchmark's absolute contribution to the macro-averaged forgetting delta (its accuracy change divided by the number of non-target benchmarks), with the share in brackets; the cells sum to the last column. The excluded cell in each panel is the target itself.

Comparison	Countdown	MMLU	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Countdown target							
init→first	–	-0.0709 (30.46%)	-0.0302 (13.00%)	-0.0460 (19.78%)	-0.0362 (15.56%)	-0.0493 (21.19%)	-0.2326
first→last	–	-0.0036 (38.35%)	-0.0043 (45.77%)	-0.0016 (16.62%)	0.0010 (-10.10%)	-0.0009 (9.35%)	-0.0094
init→last	–	-0.0745 (30.77%)	-0.0345 (14.28%)	-0.0476 (19.66%)	-0.0352 (14.56%)	-0.0502 (20.73%)	-0.2420
MMLU target							
init→first	-0.0197 (12.60%)	–	-0.0282 (18.04%)	-0.0284 (18.20%)	-0.0165 (10.54%)	-0.0635 (40.63%)	-0.1563
first→last	0.0003 (-2.51%)	–	-0.0010 (7.38%)	-0.0028 (20.33%)	-0.0011 (7.58%)	-0.0094 (67.21%)	-0.0140
init→last	-0.0193 (11.36%)	–	-0.0292 (17.17%)	-0.0313 (18.37%)	-0.0175 (10.30%)	-0.0729 (42.81%)	-0.1702

Table 30: Per-setting forgetting decomposition across the training trajectory, Countdown targets. Each cell is that benchmark's contribution to the forgetting delta; The last column is that delta.

Setting	lr	Cmp	MMLU	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Qwen/Countdown/SFT	5e-6	init→first	-0.0078	-0.0208	-0.0137	-0.0375	-0.0257	-0.1055
		first→last	0.0000	-0.0048	0.0008	0.0190	0.0040	0.0190
		init→last	-0.0078	-0.0256	-0.0129	-0.0186	-0.0217	-0.0866
Qwen/Countdown/SFT	5e-5	init→first	-0.1100	-0.0336	-0.0987	-0.0812	-0.0283	-0.3518
		first→last	0.0000	-0.0003	0.0059	0.0036	0.0054	0.0146
		init→last	-0.1100	-0.0338	-0.0929	-0.0776	-0.0229	-0.3372
Qwen/Countdown/GRPO	5e-6	init→first	-0.0633	-0.0149	0.0004	0.0053	-0.0201	-0.0926
		first→last	0.0036	-0.0039	-0.0007	-0.0040	-0.0027	-0.0077
		init→last	-0.0597	-0.0188	-0.0003	0.0013	-0.0228	-0.1003
Qwen/Countdown/GRPO	1e-5	init→first	-0.0747	-0.0234	-0.0066	-0.0154	-0.0238	-0.1439
		first→last	-0.0010	-0.0013	-0.0023	-0.0073	-0.0013	-0.0132
		init→last	-0.0757	-0.0246	-0.0089	-0.0227	-0.0252	-0.1571
Qwen/Countdown/SDFT	1e-6	init→first	0.0000	-0.0038	-0.0045	-0.0049	0.0215	0.0083
		first→last	0.0012	-0.0053	-0.0059	-0.0027	-0.0054	-0.0181
		init→last	0.0012	-0.0090	-0.0104	-0.0077	0.0161	-0.0098
Qwen/Countdown/SDFT	5e-5	init→first	-0.1035	-0.0297	-0.0848	-0.0210	-0.0216	-0.2606
		first→last	0.0055	-0.0003	-0.0068	0.0000	0.0021	0.0005
		init→last	-0.0980	-0.0300	-0.0917	-0.0210	-0.0195	-0.2602
LLaMA/Countdown/SFT	5e-6	init→first	-0.1168	-0.0467	-0.0098	-0.0239	-0.1367	-0.3339
		first→last	0.0010	-0.0042	0.0034	-0.0049	-0.0080	-0.0127
		init→last	-0.1159	-0.0510	-0.0064	-0.0288	-0.1446	-0.3467
LLaMA/Countdown/SFT	5e-5	init→first	-0.1171	-0.0672	-0.0945	-0.1181	-0.1338	-0.5307
		first→last	0.0000	-0.0008	-0.0003	0.0313	0.0065	0.0367
		init→last	-0.1171	-0.0680	-0.0949	-0.0868	-0.1272	-0.4940
LLaMA/Countdown/GRPO	5e-6	init→first	-0.0214	-0.0051	-0.0386	0.0002	-0.0076	-0.0725
		first→last	-0.0193	-0.0144	-0.0066	-0.0022	-0.0044	-0.0469
		init→last	-0.0407	-0.0195	-0.0452	-0.0020	-0.0120	-0.1194
LLaMA/Countdown/GRPO	1e-5	init→first	-0.1105	-0.0575	-0.0931	-0.1329	-0.1595	-0.5535
		first→last	-0.0066	-0.0012	0.0000	-0.0173	-0.0041	-0.0292
		init→last	-0.1171	-0.0587	-0.0931	-0.1502	-0.1636	-0.5827
LLaMA/Countdown/SDFT	5e-6	init→first	-0.0113	-0.0114	-0.0211	0.0001	-0.0014	-0.0451
		first→last	-0.0278	-0.0069	-0.0052	-0.0020	0.0019	-0.0400
		init→last	-0.0391	-0.0182	-0.0263	-0.0019	0.0004	-0.0851
LLaMA/Countdown/SDFT	1e-5	init→first	-0.1136	-0.0490	-0.0871	-0.0048	-0.0544	-0.3089
		first→last	0.0001	-0.0083	-0.0011	-0.0020	-0.0045	-0.0158
		init→last	-0.1135	-0.0573	-0.0881	-0.0069	-0.0589	-0.3247

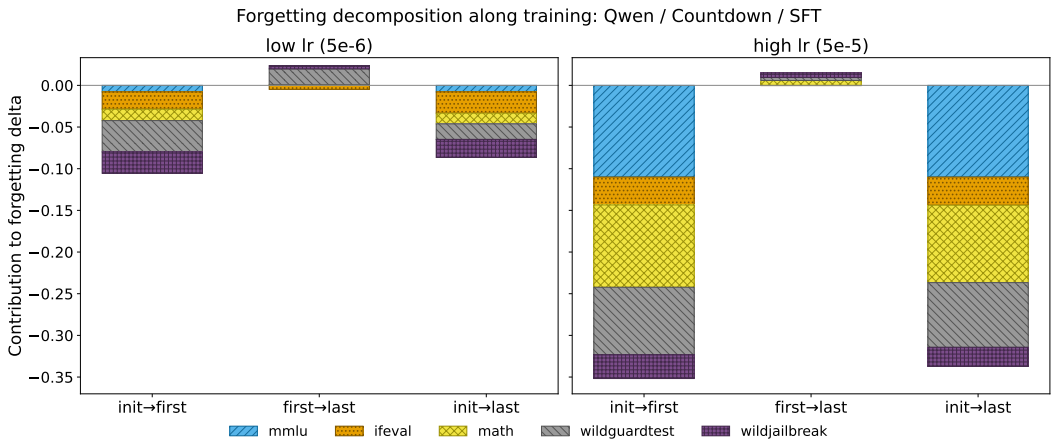


Figure 35: Qwen / Countdown / SFT: per-benchmark contribution to the mean forgetting delta across init→first→last, for the low-lr run (left) and the high-lr run (right). Both panels share a y-axis.

Table 31: Per-setting forgetting decomposition across the training trajectory, MMLU targets. Each cell is that benchmark's contribution to the forgetting delta; The last column is that delta.

Setting	lr	Cmp	Countdown	IFEval	MATH	WildGuard	WildJbrk	forgetting Δacc
Qwen/MMLU/SFT	1e-5	init→first	-0.0174	-0.0235	-0.0185	0.0004	-0.0205	-0.0795
		first→last	-0.0006	0.0041	0.0071	0.0064	0.0023	0.0193
		init→last	-0.0180	-0.0194	-0.0114	0.0068	-0.0182	-0.0602
Qwen/MMLU/SFT	1e-4	init→first	-0.0320	-0.0304	-0.0795	-0.0192	-0.0192	-0.1803
		first→last	0.0000	-0.0003	-0.0091	0.0033	-0.0003	-0.0064
		init→last	-0.0320	-0.0307	-0.0886	-0.0159	-0.0195	-0.1867
Qwen/MMLU/GRPO	5e-6	init→first	-0.0290	0.0004	-0.0005	0.0013	-0.0053	-0.0331
		first→last	-0.0004	-0.0003	-0.0001	0.0057	-0.0072	-0.0023
		init→last	-0.0294	0.0001	-0.0006	0.0071	-0.0126	-0.0354
Qwen/MMLU/GRPO	1e-5	init→first	-0.0312	0.0026	-0.0634	0.0105	-0.0035	-0.0850
		first→last	0.0006	-0.0073	0.0297	-0.0059	-0.0117	0.0054
		init→last	-0.0306	-0.0046	-0.0338	0.0046	-0.0152	-0.0796
Qwen/MMLU/SDFT	5e-6	init→first	-0.0116	-0.0199	-0.0020	0.0100	-0.0109	-0.0344
		first→last	-0.0058	-0.0059	-0.0102	-0.0074	-0.0112	-0.0405
		init→last	-0.0174	-0.0257	-0.0122	0.0026	-0.0222	-0.0749
Qwen/MMLU/SDFT	5e-5	init→first	-0.0314	-0.0314	-0.0189	-0.0085	-0.0200	-0.1102
		first→last	0.0002	-0.0003	-0.0558	-0.0100	-0.0014	-0.0673
		init→last	-0.0312	-0.0317	-0.0747	-0.0185	-0.0214	-0.1775
LLaMA/MMLU/SFT	5e-6	init→first	0.0104	-0.0433	0.0019	-0.0300	-0.1417	-0.2027
		first→last	0.0192	0.0008	0.0041	0.0044	0.0040	0.0325
		init→last	0.0296	-0.0425	0.0060	-0.0257	-0.1377	-0.1703
LLaMA/MMLU/SFT	5e-5	init→first	-0.0310	-0.0639	-0.0964	-0.0561	-0.1482	-0.3956
		first→last	0.0000	0.0002	-0.0011	0.0027	-0.0001	0.0017
		init→last	-0.0310	-0.0637	-0.0975	-0.0534	-0.1484	-0.3940
LLaMA/MMLU/GRPO	2e-6	init→first	-0.0034	-0.0024	-0.0006	-0.0106	-0.0827	-0.0997
		first→last	-0.0058	-0.0060	-0.0008	-0.0026	-0.0111	-0.0263
		init→last	-0.0092	-0.0083	-0.0014	-0.0132	-0.0939	-0.1260
LLaMA/MMLU/GRPO	1e-5	init→first	-0.0306	-0.0570	-0.0526	-0.0387	-0.1070	-0.2859
		first→last	-0.0004	0.0070	0.0293	-0.0060	-0.0318	-0.0019
		init→last	-0.0310	-0.0500	-0.0233	-0.0447	-0.1388	-0.2878
LLaMA/MMLU/SDFT	1e-6	init→first	0.0020	-0.0151	-0.0004	-0.0024	-0.0529	-0.0688
		first→last	-0.0036	-0.0053	-0.0018	-0.0097	-0.0453	-0.0657
		init→last	-0.0016	-0.0204	-0.0022	-0.0121	-0.0982	-0.1345
LLaMA/MMLU/SDFT	1e-5	init→first	-0.0310	-0.0546	-0.0104	-0.0543	-0.1497	-0.3000
		first→last	0.0008	0.0009	-0.0252	0.0063	0.0013	-0.0159
		init→last	-0.0302	-0.0536	-0.0357	-0.0479	-0.1484	-0.3158

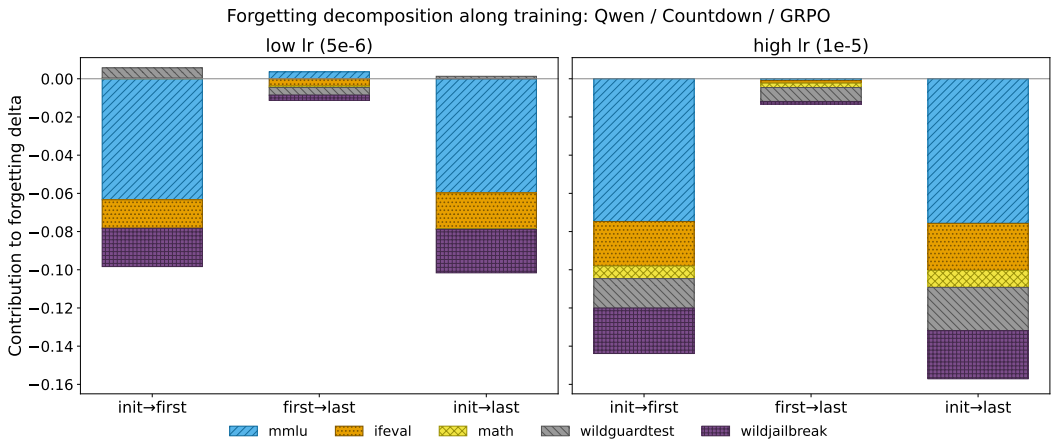


Figure 36: Qwen / Countdown / GRPO: per-benchmark contribution to the mean forgetting delta across init→first→last, for the low-lr run (left) and the high-lr run (right). Both panels share a y-axis.

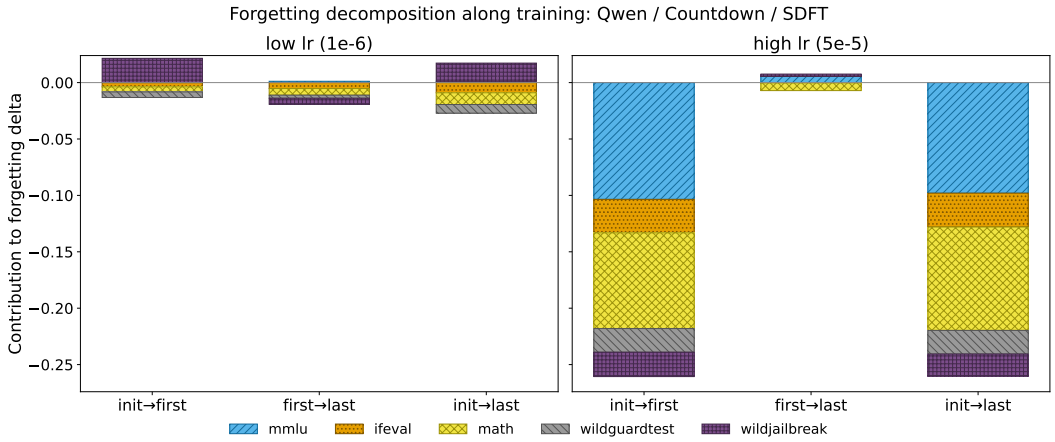


Figure 37: Qwen / Countdown / SDFT: per-benchmark contribution to the mean forgetting delta across init→first→last, for the low-lr run (left) and the high-lr run (right). Both panels share a y-axis.