UNDERSTANDING UPDATE SELECTION IN LOW-RANK FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) is commonly evaluated through final benchmark scores, which do not isolate the effects of individual training updates. We study whether additional evidence improves one-step update selection and what explains the resulting gains. Holding the model and optimizer history fixed, we compare candidate updates and measure their one-step cross-entropy reduction on held-out examples. Locally improving updates can harm held-out data, and proposals with matched effective-weight displacement norms can have different held-out gains. A frozen selection rule using additional task-balanced examples achieves mean gain 0.002071 on previously unused examples with Llama-3.2-1B, compared with -0.001090 for full updates and -0.000011 for current-loss selection. A Qwen3-4B-Base replication also shows higher mean gain with additional evidence. Reassigning choices within trajectories reduces gain while preserving candidate frequencies and acceptance rates, showing that overall preferences and fewer updates do not fully explain the benefit. Preserving acceptance locations retains much of the gain. On Qwen, simpler evidence-based policies and controls that preserve training step also retain much of the benefit. These results provide a controlled account of the gains from one-step update selection in low-rank fine-tuning.

1 INTRODUCTION

Low-rank adaptation (LoRA) enables parameter-efficient fine-tuning by training low-rank factors while keeping the pretrained backbone fixed (Hu et al., 2022). Its performance depends on how these factors are optimized and how training configurations are selected (Hayou et al., 2024; Wang et al., 2025; Lee et al., 2026; He et al., 2026). At a given training state, however, there is a more immediate choice: apply the optimizer's proposed update, reduce it, change the relative updates of the two factors, or skip it. We study what evidence helps choose among these alternatives to reduce loss on held-out data.

Final benchmark scores summarize a complete training trajectory, but do not isolate the effect of the current update. Comparing updates from different trajectories also mixes the current choice with differences in earlier training. Local signals offer a more direct view: line-search and cautious optimization methods use current optimization information to rescale or filter updates (Vaswani et al., 2019; Liang et al., 2026). Whether an update that improves the current batch also benefits other data is a separate question. Additional data have been used to guide training decisions (Mindermann et al., 2022; Xia et al., 2024; Yang et al., 2026a), but evaluating update selection requires distinguishing better evidence from a general preference for smaller or fewer updates.

We address this problem through fixed-state comparisons. We restore the same LoRA parameters, current gradient, and optimizer history before evaluating each candidate update. We then separate the data used to construct candidates, the evidence used to select one, and the held-out examples used to measure its effect. This design lets us ask whether changing the selection evidence improves one-step utility without changing the available candidates. Retrospective controls then test whether any gain depends on which choices are assigned to which source states.

Our experiments use Llama-3.2-1B (Meta, 2024) on six classification tasks and repeat the analysis on Qwen3-4B-Base (Qwen Team, 2025). Current-batch improvement can fail to transfer to held-out data, and proposals with matched effective-weight displacement norms can have different held-out gains. We therefore evaluate selection using additional task-balanced examples. After exploration

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.

on six tasks, we freeze the selection procedure and confirm its effect on previously unused examples from four tasks at 120 Llama source states. The selected updates achieve mean one-step cross-entropy gain 0.002071, compared with -0.001090 for full updates and -0.000011 for current-loss selection. The Qwen replication also shows higher mean gain with additional evidence.

The controls clarify how to interpret these gains. Reassigning choices within trajectories lowers mean gain even when candidate frequencies and acceptance rates remain unchanged. Preserving the states at which updates are accepted retains much of the original gain, making acceptance locations an important part of the comparison. Qwen's simpler evidence-based policies and controls that preserve training step also retain much of the benefit. Together, these results establish the value of additional selection evidence while distinguishing it from the value of a richer candidate set or finer state matching.

Our contributions are threefold:

- **Controlled analysis of local update evidence.** We compare alternative LoRA updates from the same training state and show that current-batch improvement and effective-weight displacement do not determine held-out utility.
- **Confirmation of one-step selection gains.** A frozen selection procedure using additional examples improves mean held-out loss reduction on unused Llama evaluation data, with the same pattern observed in a Qwen replication.
- **Selection gains beyond overall preferences.** We show that overall candidate preferences and acceptance rates do not fully explain the gains, and use acceptance-location, candidate-assignment, and training-step controls to examine the value of matching choices to states.

2 Related Work

Low-rank fine-tuning and configuration-level evaluation. LoRA introduces parameter-efficient adaptation by representing weight changes through trainable low-rank factors while keeping pre-trained parameters frozen (Hu et al., 2022). Subsequent work improves different aspects of this parameterization and optimization process (Liu et al., 2024; Meng et al., 2024; Wang et al., 2024; Wu et al., 2026). LoRA+ assigns different learning rates to the two LoRA factors (Hayou et al., 2024), while LoRA-Pro modifies factor gradients to better approximate the update behavior of full fine-tuning (Wang et al., 2025). Other studies investigate the effective capacity and scaling behavior of low-rank adaptation, including rank-dependent scaling and higher-rank parameterizations (Kalajdzievski, 2023; Huang et al., 2025; Jiang et al., 2024). Recent analyses further show that optimization behavior depends jointly on factors such as rank, initialization, scaling, and learning-rate choices (Li et al., 2025; Chen et al., 2026; Wu et al., 2026).

Beyond proposing new parameterizations, recent work has examined how fine-tuning configurations should be compared. Learning-rate retuning can substantially change observed gaps between LoRA variants (Lee et al., 2026; He et al., 2026), while downstream classification metrics can depend on prediction rules that convert model scores into discrete decisions (Lipton et al., 2014). These studies show that reported configuration-level differences depend on both training and evaluation choices. Our analysis compares candidate updates directly from the same training state.

Local update signals and update control. Optimization methods often modify updates using information available during training. Hypergradient methods adapt learning rates using hypergradients through the optimization process (Baydin et al., 2017), while line-search methods adjust step sizes according to objective changes (Vaswani et al., 2019). Gradient surgery methods modify updates by reducing conflicts among multiple objectives (Yu et al., 2020), and cautious optimization methods mask update components whose signs disagree with the current gradient (Liang et al., 2026). These approaches demonstrate different ways of constructing, rescaling, or filtering updates based on local optimization information.

Related ideas also appear in continual and online parameter-efficient adaptation. Online LoRA and related methods modify how models incorporate new information through mechanisms such as dynamic adaptation, subspace constraints, or adapter management (Wei et al., 2025; Wang et al., 2023; He et al., 2025; Yang et al., 2026b). These approaches primarily address how adaptation can incorporate new information while retaining previously learned knowledge. Our focus is the utility

of individual updates: we hold the source model and optimizer history fixed, compare candidate updates generated from the same state, and measure their effects on held-out data. This separates candidate construction from the evidence used to assess candidate utility.

Evidence-guided update selection. Using additional information beyond the current training batch to guide optimization has been explored in several areas (Mindermann et al., 2022; Xia et al., 2024; Yang et al., 2026a). Example reweighting methods use validation feedback to assign importance to training examples (Ren et al., 2018), while bilevel optimization provides a framework for hyperparameter optimization and meta-learning through inner training and outer validation objectives (Franceschi et al., 2018). More recently, holdout-loss-based data selection estimates training-example utility through an in-context approximation of its effect on holdout loss (Zhang et al., 2026). These studies show that additional evidence can improve decisions related to training examples or optimization parameters.

Our work differs by studying the selection of candidate updates rather than the selection or weighting of training examples. We compare selection evidence within a fixed candidate set and evaluate choices on separate examples. We then use simple selection policies and retrospective reassignment controls to test whether the gains reflect richer choices, overall candidate preferences, or the assignment of choices to states.

3 A Controlled Framework for Update Selection

We study a single decision: given a training state and a set of candidate updates, which evidence helps select an update that improves held-out data? Our framework separates candidate construction, selection evidence, and outcome evaluation. Every candidate is evaluated from the same source state, so earlier training history is held fixed.

3.1 Candidate Updates from a Fixed State

A source state consists of the LoRA parameters, the current-batch gradient, and optimizer information, including AdamW (Loshchilov & Hutter, 2019) moments and step count. We restore this state before applying each candidate update. Candidates are alternatives to one another, rather than successive steps in a new training trajectory.

The shared optimizer history matters because an AdamW proposal depends on accumulated moments as well as the current gradient. Restoring only the parameters would therefore leave an uncontrolled difference between proposals. For each source state, we construct its candidates, score each resulting model on the same examples, and return to the source before the next comparison. This produces a paired comparison of alternative decisions at that point in training.

For an adapted matrix, LoRA represents the weight as $W = W_0 + \gamma BA$, where W_0 is frozen and A and B are trainable factors. A factor update $(\Delta A, \Delta B)$ induces the effective-weight change

$$\Delta W = \gamma\big[(B + \Delta B)(A + \Delta A) - BA\big]. \tag{1}$$

We measure its aggregate size across adapted matrices by $\|\Delta W\|_F = (\sum_\ell \|\Delta W^{(\ell)}\|_F^2)^{1/2}$.

Our candidates include no update, scaled versions of the source optimizer proposal, proposals with different relative learning rates for the two factors, and norm-matched versions of these proposals. Scaling tests whether smaller steps improve target utility. Norm matching tests whether utility differences remain when aggregate effective displacement is held constant. Equation 1 also shows why scaling both factor updates is not a linear scaling of the effective-weight change.

The factor-ratio comparison uses a choice specific to LoRA's parameterization. A smaller source step reduces both factor updates together, whereas changing their learning-rate ratio changes their relative contributions to ΔW. Both proposals use the same gradient and optimizer history. Their norm-matched versions then ask whether a difference in target gain remains after removing the difference in total effective displacement.

Table 1: Three comparisons used to study update selection. All start from saved source states and use the same one-step evaluation objective.

Comparison	Held fixed	Changed	Core question
Selection evidence	Source states, candidates, evaluation examples	Current batch vs. additional examples	Which evidence improves selection?
Candidate choice	Source states, selection and evaluation examples	Accept/reject, step sizes, or full candidate set	Do more choices improve gain?
Choice assignment	State-by-candidate evaluation gains	Correspondence between choices and states	Is the original assignment useful?

3.2 Selection Evidence and One-Step Utility

Let θ denote the source LoRA parameters and δ a candidate update. For evaluation examples Q, the one-step gain is

$$G_Q(\delta) = \mathcal{L}(Q;\theta) - \mathcal{L}(Q;\theta+\delta), \tag{2}$$

where $\mathcal{L}$ is mean cross-entropy (CE). Positive gain means lower loss, and no update has zero gain. CE captures small changes that may not yet alter discrete predictions. Throughout the paper, update utility refers to this one-step effect.

The current batch supplies the gradient used to construct candidates. Selection evidence chooses among them, and separate evaluation examples measure the resulting gain. Given selection data S and candidate set $\mathcal{C}$, we choose

$$\hat{\delta} = \arg\min_{\delta\in\mathcal{C}} \mathcal{L}(S;\theta+\delta). \tag{3}$$

Current-loss selection uses the current batch as S; additional-evidence selection uses a separate set of examples. Changing S changes the evidence for the decision, while the evaluation objective in Equation 2 remains the same. Both selectors include no update; ties favor smaller effective displacement and then a fixed candidate order.

Including no update makes acceptance part of selection: a nonzero candidate is chosen only if it wins against leaving the source unchanged. Consequently, the selected candidate cannot have higher selection loss than no update, apart from numerical ties. Its gain on the separate evaluation set can still be negative. That difference between selection loss and evaluation gain is the transfer question studied here.

3.3 Evaluating Selection Gains

We first compare selection evidence within the same candidate set. Full updates, fixed half steps, and no update provide reference choices. Exploratory analysis determines the additional-evidence procedure; confirmation freezes the candidate set, evidence budget, and selection rule before evaluation on unused examples.

We then examine what the selection gain depends on. Simpler policies use the same additional evidence to accept or reject the full update, or to choose its step size. These comparisons test the value of expanding the available choices. Retrospective controls instead reassign the selector's choices across sampled states while preserving candidate frequencies and acceptance rates. Each state keeps its own candidate updates; only the selected candidate identity changes. Comparing the original gain with expected gain after reassignment tests the value of the original state–choice correspondence. Holding acceptance locations fixed further distinguishes where updates are applied from which nonzero candidate is assigned there.

Table 1 summarizes the three comparisons. They separate the evidence supplied to a decision, the choices it can make, and the assignment of those choices to source states.

Table 2: Prediction rules change configuration comparisons without changing checkpoints. Three-seed means. **(a)** Rank-8 QQP F1; Δ is LoRA+ minus LoRA. **(b)** Shuffled LoRA rank comparison. "Thr." uses development-selected thresholds; CE is macro development loss.

(a) Method comparison

Order	Rule	LoRA	LoRA+	Δ
Alt.	Raw	.5476	.5692	$+.0217$
Alt.	Thr.	.5822	.6410	$+.0587$
Shuf.	Raw	.6862	.5639	$-.1223$
Shuf.	Thr.	.7473	.7402	$-.0071$

(b) Rank comparison

Rank	CE	Raw F1	Thr. F1
4	.5071	.5104	.7365
8	.4901	.6862	.7473
16	.4706	.6358	.7536
32	.4712	.5618	.7532

4 Experimental Setup

We obtain source states from completed fine-tuning runs, then compare candidate updates from each state using the protocol in Section 3. The experiments combine exploratory analysis, confirmation on unused examples, and replication on a second backbone.

4.1 Models, Tasks, and Source Trajectories

We fine-tune Llama-3.2-1B (Meta, 2024) on six binary GLUE tasks (Wang et al., 2018): CoLA (Warstadt et al., 2019), MRPC (Dolan & Brockett, 2005), SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2016), QQP (Quora, 2017), and RTE (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). Each run processes the same 12,288 examples once, in alternating task phases or globally shuffled order. Standard LoRA at ranks $4, 8, 16, 32$ and rank-8 LoRA+ (Hayou et al., 2024) each receive four learning-rate trials per order and three seeds, giving 120 runs. For each method/rank, order, and seed, we select the learning rate by final-checkpoint development CE averaged equally over the six tasks. The resulting 30 trajectories provide 120 source states at steps $10, 60, 110, 160$.

Each task contributes 2,048 training examples. Alternating order visits each task in two phases of 1,024 examples, while shuffled order mixes the same examples globally. The two orders therefore retain the same training budget and overall task counts but expose the optimizer to different local task mixtures. The sampled steps let us compare selection decisions at several positions along each resulting trajectory.

For Qwen3-4B-Base (Qwen Team, 2025), we use rank-8 LoRA and LoRA+, two data orders, three seeds, and four learning-rate trials per configuration. Development CE selects 12 trajectories, which provide 48 source states at the same four steps. The candidate definitions and one-step evaluation objective are unchanged.

Both backbones use query/value adapters with fixed LoRA scaling $\gamma = 2$, a frozen BF16 backbone, and FP32 adapter parameters. Training takes 192 AdamW steps at effective batch size 64, with maximum sequence length 512. Each classification prompt states the task and its two label meanings; the model predicts the corresponding token, `0` or `1`. CE is computed from probabilities normalized over these two label logits, using the frozen language-model output weights. This gives the six tasks a common loss interface without generating or parsing free-form answers.

Final-checkpoint comparisons provide context for these source trajectories. For example, the shuffled-order QQP F1 gap between rank-8 LoRA+ and LoRA changes from -0.1223 to -0.0071 when development-selected thresholds replace raw predictions on the same checkpoints (Table 2). This illustrates how endpoint comparisons depend on the evaluation rule; our update analysis instead measures candidate effects directly from fixed states. Training details and the full configuration comparisons appear in Appendices A.1–A.3.

4.2 Candidates and Selection Data

The candidate set contains nine choices: no update; source-proposal fractions $c \in \{1/16, 1/8, 1/4, 1/2\}$; proposals with B-to-A factor learning-rate ratios 1 and 16; and norm-matched versions of both proposals. The full source update is one of the two raw-ratio proposals.

We vary the A-factor learning rate while holding the B-factor rate fixed. Norm matching scales both factor paths to the smaller raw effective-displacement norm. All candidates share the source parameters, clipped current-batch gradient, AdamW moments, and step count.

For Llama exploration, selection and evaluation panels each contain 768 examples from the earlier development data, with a fixed 192-example evaluation subset. Exploration determines the evidence budget and task-balancing procedure. Confirmation then uses unused examples from SST-2, QNLI, QQP, and RTE: 48 selection and 64 evaluation examples per task, giving totals of 192 and 256. These are the four tasks with sufficient eligible unused examples. The panels are disjoint, excluded from gradient training, and fixed across the 120 source states.

Qwen uses the same four target tasks with disjoint selection and evaluation panels. Five fixed draws use nested selection budgets of 64, 128, and 192 examples; the main comparison uses 192. These examples are separate from Qwen training and development data but were observed in earlier experiments, so we treat Qwen as a cross-backbone replication. Details of data roles and exploratory choices appear in Appendices A.4 and A.5.

4.3 Measurements

We compute all gains using FP32 loss evaluation on stored model values. A harmful update has gain below $-\tau$, and a positive event has gain above τ, where $\tau = 10^{-4}$. Mean gains include all values without thresholding. We average states within trajectories and then across trajectories; Qwen additionally averages the five selection draws. Harm rates count all evaluated decisions, including skips, unless explicitly reported among accepted updates. Paired bootstrap intervals and their resampling units are described in Appendix A.5.

5 Update Selection: Gains and Their Sources

We first test whether local signals identify useful updates, then ask whether additional evidence improves selection among the same candidates. Finally, we examine whether the gains reflect overall candidate preferences, fewer accepted updates, or the assignment of choices to particular states.

5.1 Local Improvement and Update Magnitude

Current-batch improvement does not consistently transfer to held-out data. Among 87 locally improving full updates in the Llama analysis, 33 increase loss on the 768-example evaluation panel. The same pattern appears in Qwen: 25 of 47 locally improving full updates increase loss on the four-task panel. Thus, an update can fit the data that generated its gradient while harming the evaluation objective.

Reducing the step does not reliably turn harmful updates into beneficial ones. For Llama, testing the four smaller source-proposal fractions repairs 7.0%–17.0% of harmful updates across the evaluated targets, where repair requires gain to move from below $-\tau$ to above τ. Detailed transfer and repair results appear in Appendix A.6.

We also compare proposals with different factor learning-rate ratios after matching their aggregate effective-weight displacement norms. Their held-out gains can still differ: 47 of 48 Qwen pairs differ by more than τ. The Llama comparisons likewise show that a proposal can improve held-out gain while reducing current-batch gain (Appendix A.7). Matching total displacement leaves direction and layer-wise allocation free to vary. For example, at alternating rank 8 on Llama, the matched source and alternative gains are -0.001906 and -0.000768. The alternative reduces the harm at the same total movement, but neither proposal improves the target loss. The example separates two questions: which proposal is better, and whether either should be accepted. These results motivate selecting candidates using evidence about the target data, rather than relying on current-batch improvement or total movement alone.

5.2 Additional Evidence Improves Selection

We compare current-loss selection with selection based on additional task-balanced examples. Both use the same nine candidates and minimize post-update CE. The 192-example budget comes from

Table 3: Additional evidence improves mean one-step selection gain. Gain is in 10^{-3} CE; harm and acceptance are percentages. Harm counts all decisions. Additional-evidence policies use 192 selection examples.

Llama-3.2-1B

Policy	Gain	Harm	Accept
No update	0.000	0.0	0.0
Full	−1.090	59.2	100.0
Half	−0.050	53.3	100.0
Current loss	−0.011	39.2	80.0
Balanced 192	**+2.071**	**8.3**	40.0

Qwen3-4B-Base

Policy	Gain	Harm	Accept
No update	0.000	0.0	0.0
Full	+1.453	54.2	100.0
Current loss	+1.996	45.8	97.9
Accept only	+2.099	**17.1**	60.8
Line search	+2.097	20.0	64.6
Full selection	**+2.558**	24.6	75.4

Table 4: Llama confirmation gains across source groups, in CE units. Each row averages four states per trajectory and three seeds using the frozen four-task panels. Balanced 192 exceeds all three references in every row.

Order	Source	Full	Half	Current loss	Balanced 192
Alternating	LoRA $r = 4$	−0.005197	−0.002589	−0.004076	**+0.000024**
Alternating	LoRA $r = 8$	−0.003358	−0.001547	−0.001418	**−0.000021**
Alternating	LoRA $r = 16$	−0.001613	−0.000705	−0.000881	**+0.000100**
Alternating	LoRA $r = 32$	−0.000981	−0.000376	−0.000235	**+0.000351**
Alternating	LoRA+ $r = 8$	+0.000183	+0.000301	−0.000820	**+0.004248**
Shuffled	LoRA $r = 4$	+0.001830	+0.001276	+0.002721	**+0.003022**
Shuffled	LoRA $r = 8$	−0.005889	−0.001559	−0.002001	**+0.000419**
Shuffled	LoRA $r = 16$	+0.001682	+0.001828	+0.003283	**+0.005484**
Shuffled	LoRA $r = 32$	+0.003608	+0.002861	+0.004804	**+0.006144**
Shuffled	LoRA+ $r = 8$	−0.001161	+0.000012	−0.001482	**+0.000937**

the six-task exploratory analysis. Task-balanced selection with either 192 or 768 examples produces positive mean panel gain in all ten method/rank–order groups. Increasing the budget from 192 to 768 lowers gain in three groups, so the larger panel does not improve every comparison. At a fixed budget of 48 examples, task-balanced sampling also exceeds sampling matched to the current batch's task composition in eight groups. These results motivate the 192-example balanced procedure used for confirmation; the complete exploratory results remain in Appendix A.4.

For Llama, the candidate set, selection rule, 192-example budget, and balancing procedure are frozen before confirmation. The selected updates achieve mean gain 0.002071, compared with -0.001090 for full updates, -0.000050 for half steps, and -0.000011 for current-loss selection (Table 3). The 95% bootstrap interval for the selected mean gain is $[0.000449, 0.003731]$.

The selector exceeds full updates, half steps, and current-loss selection in all ten Llama source groups. Its absolute mean gain is positive in nine groups and slightly negative in alternating rank-8 LoRA. Of the 48 accepted updates, ten remain harmful: 20.8% of accepted updates and 8.3% of all states. The improvement is therefore in average utility, with fewer harmful choices.

Table 4 shows how the gain varies across source configurations. The benefit is not confined to one rank or data order. For alternating LoRA, selection often turns a negative mean into a value near zero; for shuffled ranks 16 and 32, it improves an already positive mean. Alternating rank-8 LoRA remains slightly negative, but improves from -0.001418 under current-loss selection to -0.000021. These comparisons show both avoidance of harmful choices and improvement over useful reference updates.

Cross-backbone replication. On Qwen, the 192-example selector achieves mean gain 0.002558, compared with 0.001453 for full updates and 0.001996 for current-loss selection. The paired improvements are 0.001105 and 0.000562, respectively, both with positive bootstrap intervals. Across the two backbones, changing the selection evidence improves average one-step utility without expanding the candidate set.

Table 5: Selection gains depend on the assignment of choices to states. Mean gains and paired gaps are in 10^{-3} CE. Reassignment preserves candidate frequencies; "fix accepted states" also preserves acceptance locations. The same-step control groups Qwen states by data order, method, and step, then reassigns choices across seeds. The oracle selects one fixed identity per trajectory using evaluation gains.

(a) Mean gain

Control	Llama	Qwen
Original	**2.071**	**2.558**
Reassign all	−0.002	1.928
Fix accepted states	1.642	2.485
Same step, cross-seed	—	2.546
Best fixed (oracle)	1.172	2.200

(b) Qwen original-minus-control

Reassignment	Gap	95% interval
All choices	0.630	$[0.357, 0.910]$
Accepted only	0.073	$[0.011, 0.158]$
Same step	0.013	$[-0.016, 0.045]$

5.3 Evidence Value and Candidate Choice

The previous comparison changes the evidence available for selection. We next ask how much decision flexibility is needed to benefit from that evidence. On Qwen, acceptance-only chooses between the full update and no update; validation line search also considers the four smaller source fractions. Both use the same 192-example selection panels as full candidate selection.

Acceptance-only and line search achieve mean gains 0.002099 and 0.002097, respectively, compared with 0.002558 for full selection. The paired advantages of full selection have intervals containing zero. Additional evidence thus benefits simple policies as well, while the incremental gain from the richer candidate set remains uncertain.

Expanding the choices also changes where updates are accepted. Full selection accepts 75.4% of Qwen decisions, compared with 60.8% for acceptance-only and 64.6% for line search. An alternative proposal can win on the selection panel even when the full source update would be rejected. The resulting policies trade off mean gain and harmful choices differently: full selection has the highest mean gain, while acceptance-only has the lowest harm rate of these three policies. This motivates separating acceptance locations from candidate assignment in the following controls.

5.4 Acceptance Locations and Candidate Assignment

Does the selector gain mainly by choosing some candidates more often or accepting fewer updates? We test this by reassigning its observed candidate identities across the four sampled states within each trajectory. The control preserves candidate frequencies and acceptance rates, while each state retains its own candidate updates.

The control does not move an update tensor from one state to another. It moves only the choice label, such as full update, a source fraction, or no update, and evaluates that choice using the recipient state's saved candidate scores. We average over the possible assignments, including repeated labels, rather than report a single random shuffle. Thus, any difference from the original selection concerns the assignment of choices, not a change in the available updates.

For Llama, within-trajectory reassignment reduces mean gain from 0.002071 to approximately zero. The original-minus-reassignment gap is positive in all three seed aggregates. The aggregate gap has a paired 95% bootstrap interval of $[0.001233, 0.002917]$ CE. This interval reflects evaluation-example variation conditional on the fixed tasks, source states, and selection panel, rather than variation from new trajectories or selection panels. For Qwen, reassignment reduces mean gain from 0.002558 to 0.001928, with a paired gap interval entirely above zero (Table 5). Because reassignment preserves candidate frequencies and acceptance rates within each trajectory, these results indicate that overall candidate preferences and update frequency do not fully explain the observed gains.

A second reference chooses the best fixed candidate identity separately for each trajectory using its evaluation gains, then applies that identity at all four sampled states. Even this retrospective oracle has lower mean gain than the original choices: 0.001172 on Llama and 0.002200 on Qwen. Allowing each trajectory its own preferred candidate therefore does not reproduce the observed mean benefit of varying choices across states.

We next keep acceptance locations fixed and reassign only nonzero candidate identities among accepted states. Expected gain is then 0.001642 for Llama and 0.002485 for Qwen, retaining much of the original benefit. The remaining gap measures the value of the original candidate assignment at these accepted states. This distinguishes the role of where updates are accepted from how often they are accepted.

This control differs from the acceptance-only policy in Section 5.3. It inherits the acceptance locations of full candidate selection and changes only the nonzero assignments. Acceptance-only instead makes its own decisions between the full update and no update using the selection examples. The two comparisons address different aspects of selection and need not have the same acceptance rate or gain.

Matching beyond training step. Qwen also permits a control that preserves data order, method, and training step while reassigning choices across seeds. Its mean gain is 0.002546, close to the original 0.002558; the paired gap interval contains zero. Thus, the gain lost through within-trajectory reassignment does not establish an additional benefit from finer matching beyond these groups. The controls support the value of the observed state–choice correspondence, while showing that its useful level of detail remains to be determined.

6 Discussion

Evidence for the target objective. The local-signal results distinguish fitting the current batch from improving the target objective. A smaller step can reduce harm without producing a positive gain, and equal effective displacement can conceal different target effects. Constructing an update and judging its utility are therefore separate problems. The confirmation and replication results show that additional target-relevant examples improve selection among the same candidates. This makes evidence design a concrete way to improve decisions without changing how candidate updates are constructed.

Acceptance locations carry useful information. Reassignment preserves the overall distribution of choices but lowers gain on both backbones. Thus, a selector's average acceptance rate or preferred candidate does not fully describe its benefit. Retaining acceptance locations preserves much of the original gain, making where updates are applied an important part of the explanation. Qwen's same-step control also retains much of the benefit across seeds within the same method, data order, and training step. The useful structure in these choices can therefore be shared across states; its value need not require fine-grained matching to each individual state.

Evidence value and candidate complexity. The Qwen comparisons give stronger support to additional selection evidence than to an expanded candidate set. Acceptance-only and line search already retain much of the mean gain, and the incremental advantage of full selection is uncertain. Full selection also accepts more updates and has a higher harm rate than these simpler policies, despite its higher mean gain. Maximizing average loss reduction and minimizing harmful decisions are thus different goals. Comparisons should hold evidence fixed when expanding candidate sets, and report gain alongside harm and acceptance. Reassignment controls then test what the original assignment of choices contributes beyond these aggregate statistics.

7 Conclusion

Our fixed-state study shows that additional target-relevant evidence improves mean one-step LoRA update selection on Llama and Qwen. Current-batch improvement and total displacement do not determine held-out utility. Selection gains extend beyond overall candidate preferences and acceptance rates, with acceptance locations retaining much of the benefit. Simple evidence-based policies also capture much of the Qwen gain. These findings motivate evaluating update selection through the evidence supporting a decision, the available choices, and their assignment to training states.

ETHICS STATEMENT

This work studies update selection in low-rank fine-tuning using existing GLUE datasets and publicly available pretrained models. It involves no recruitment of human participants or collection of new personal data. Our experiments focus on controlled comparisons of training updates. Existing models and datasets may contain societal biases; the reported improvements in held-out loss do not constitute an assessment of fairness or deployment safety.

REPRODUCIBILITY STATEMENT

Section 3 defines the fixed-state comparison protocol, candidate updates, and selection objective. Section 4 describes the models, tasks, source trajectories, and data roles. Appendix A provides training hyperparameters, random seeds, selection and evaluation protocols, retrospective controls, and uncertainty estimation procedures. These details specify how the update comparisons and reported selection gains are computed.

LLM USAGE

We utilized LLM assistants to assist in implementing and debugging the experimental codebase, as well as in drafting and revising the manuscript (including structural organization and LaTeX formatting). All reported numerical results are obtained directly from executed experiments and logged outputs. The authors retain full responsibility for verifying all analyses, and for the final content, claims, and associated research artifacts.

REFERENCES

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop*, Venice, Italy, 2006. URL https://cris.biu.ac.il/en/publications/the-second-pascal-recognising-textual-entailment-challenge/.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Text Analysis Conference (TAC)*, 2009. URL https://tac.nist.gov/publications/2009/additional.papers/RTE5_overview.proceedings.pdf.

Nan Chen, Soledad Villar, and Soufiane Hayou. Learning rate scaling across lora ranks and transfer to full finetuning, 2026.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges: Evaluating Predictive Uncertainty, Visual Object Classification, and Recognizing Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pp. 177–190. Springer, 2006. doi: 10.1007/11736790_9. URL https://link.springer.com/chapter/10.1007/11736790_9.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://aclanthology.org/I05-5002/.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pp. 1568–1577. PMLR, 2018.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9. Association for Computational Linguistics, 2007. URL https://aclanthology.org/W07-1401/.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=NEv8YqBROO.

Haonan He, Jingqi Ye, Minglei Li, Zhengbo Wang, Tao Chen, Lei Bai, and Peng Ye. A unified study of lora variants: Taxonomy, review, codebase, and empirical evaluation, 2026.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30534–30544. IEEE, 2025.

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

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 29614–29633, 2025.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. Mora: High-rank updating for parameter-efficient finetuning, 2024.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

Yu-Ang Lee, Ching-Yun Ko, Pin-Yu Chen, and Mi-Yen Yeh. Learning rate matters: Vanilla lora may suffice for llm fine-tuning. *arXiv preprint arXiv:2602.04998*, 2026.

Shiwei Li, Xiandi Luo, Xing Tang, Haozhao Wang, Hao Chen, Weihong Luo, Yuhua Li, Xiuqiang He, and Ruixuan Li. Beyond zero initialization: Investigating the impact of non-zero initialization on lora fine-tuning dynamics. *arXiv preprint arXiv:2505.23194*, 2025.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code, 2026.

Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. Thresholding classifiers to maximize f1 score. *arXiv preprint arXiv:1402.1892*, 2014.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Meta. Llama-3.2-1B model card. Hugging Face model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-1B.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

Quora. Question pairs dataset. Kaggle dataset, 2017. URL https://www.kaggle.com/datasets/quora/question-pairs-dataset.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pp. 4334–4343. PMLR, 2018.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1170/.

Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *Advances in neural information processing systems*, 32, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning, 2023.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, volume 2025, pp. 93787–93808, 2025.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Xiwen Wei, Guihong Li, and Radu Marculescu. Online-lora: Task-free online continual learning via low rank adaptation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6634–6645. IEEE, 2025.

Yize Wu, KE GAO, Ling Li, and Yanjun Wu. Stable-loRA: Stabilizing feature learning of low-rank adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xSa19DAieH.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PG5fV50maR.

Huitao Yang, Hengzhi He, and Guang Cheng. Let the target select for itself: Data selection via target-aligned paths, 2026a.

Lina Yang, Yusheng Liao, Yanfeng Wang, and Yu Wang. Slora: Balancing plasticity and forgetting in large language models for continual learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5437–5454, 2026b.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Ling Zhang, Xianliang Yang, Juwon Yu, Park Cheonyoung, Lei Song, and Jiang Bian. Holdout-loss-based data selection for llm finetuning via in-context learning, 2026.

A Supporting Experimental Details and Results

A.1 Configuration Protocol

Configuration selection and evaluation. For a configuration grid $\mathcal{H}$, we select $\hat{h} = \arg\min_{h \in \mathcal{H}} J_{\text{dev}}(h)$, where J_{dev} is CE averaged equally over the six tasks. We report MCC for CoLA, F1 for MRPC and QQP, and accuracy otherwise, using prepared evaluation splits and FP32 evaluation. Raw label-token argmax is compared with thresholds maximizing each checkpoint's development task metric, then applied unchanged to evaluation data. Evaluation labels select neither learning rates nor thresholds.

The backbone is frozen in BF16, with FP32 query/value adapters and fixed LoRA scaling $\gamma = 2$. Each run uses 192 AdamW steps at batch size 64. Task identities and boundaries are not supplied to the optimizer. Evaluation uses FP32 arithmetic on stored weights and prepared task splits rather than the official GLUE test server. All runs use initialization seeds 46, 47, and 48, maximum sequence length 512, and AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and gradient clipping at norm 1. The shuffled order uses order seed 20260925. Standard LoRA uses $\eta_A = \eta_B \in \{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}\}$. LoRA+ uses $\eta_A \in \{6.25 \times 10^{-6}, 1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}\}$ and $\eta_B = 16\eta_A$. All comparisons use final checkpoints after 192 steps. Exact ties in macro development CE favor the smaller learning rate. Equal trial counts match the number of configurations evaluated, not search coverage or compute cost.

Classification uses probabilities normalized over the two label-token logits. For thresholded predictions, class 1 is predicted when $p_1 > t$. For each checkpoint and task, candidate thresholds comprise development probabilities, adjacent probability midpoints, 0.5, and all-positive/all-negative sentinels -1 and 1. The development task metric selects the threshold; ties favor proximity to 0.5, then the larger threshold. These thresholds remain fixed on the evaluation split. Threshold fitting is part of the development procedure and is not an unbiased assessment of development performance.

There are 17 boundary selections among the 30 method/rank–seed–order combinations. The rank-8 method comparison contains 8 boundary selections out of 12, and the LoRA rank comparison contains 12 out of 24. These subsets overlap at rank-8 LoRA and their counts must not be added.

A.2 Additional Configuration Results

A.2.1 Evaluation Rules Change Configuration Interpretation

We first examine whether reported configuration differences depend on how model outputs are converted into predictions. We keep the selected checkpoints fixed and change only the prediction rule.

Under shuffled order, raw QQP F1 is 0.6862 for LoRA and 0.5639 for LoRA+, producing a difference of -0.1223. Using development-selected thresholds on the same checkpoints changes these scores to 0.7473 and 0.7402, reducing the observed difference to -0.0071 (Table 2a). No parameters, optimizer states, or training trajectories are modified between these evaluations.

The effect is not limited to reducing the magnitude of a gap. Under alternating order, the LoRA+ advantage increases from +0.0217 to +0.0587 after threshold selection. Across tasks, the relative ordering of configurations can also change with the prediction rule (Appendix A.2). These results show that reported configuration differences depend partly on evaluation choices applied after training.

The same phenomenon appears when comparing adapter ranks. Under shuffled order, macro development CE decreases from 0.5071 at rank 4 to 0.4706 at rank 16, with little additional change at rank 32. However, the observed mean QQP F1 ordering differs across prediction rules. Raw predictions give $8 > 16 > 32 > 4$, while development-selected thresholds give $16 > 32 > 8 > 4$ (Table 2b). The checkpoints are identical between these evaluations; only the prediction rule changes.

These examples do not establish a universally preferable method or rank. Instead, they show that final configuration comparisons combine effects from training choices and evaluation rules. Consequently, endpoint comparisons alone do not identify the utility of the updates that produced the evaluated models.

Table 6: Prediction-rule comparison on all six tasks, using the same selected rank-8 checkpoints within each order. Three-seed means in each task's native metric; Δ is LoRA+ minus LoRA.

		Raw argmax			Development threshold		
Order	Task	LoRA	LoRA+	Δ	LoRA	LoRA+	Δ
Alt.	CoLA	0.0071	0.0000	-0.0071	0.0776	0.1741	$+0.0966$
Alt.	MRPC	0.5407	0.5208	-0.0198	0.8086	0.8103	$+0.0017$
Alt.	SST-2	0.7840	0.8823	$+0.0982$	0.8609	0.8888	$+0.0279$
Alt.	QNLI	0.6126	0.6551	$+0.0425$	0.6434	0.7000	$+0.0566$
Alt.	QQP	0.5476	0.5692	$+0.0217$	0.5822	0.6410	$+0.0587$
Alt.	RTE	0.5331	0.5969	$+0.0638$	0.5511	0.5933	$+0.0421$
Shuf.	CoLA	0.0000	0.0086	$+0.0086$	0.2835	0.2806	-0.0028
Shuf.	MRPC	0.8503	0.8446	-0.0056	0.8526	0.8463	-0.0063
Shuf.	SST-2	0.9308	0.9266	-0.0042	0.9258	0.9232	-0.0027
Shuf.	QNLI	0.7855	0.7747	-0.0107	0.8151	0.7930	-0.0221
Shuf.	QQP	0.6862	0.5639	-0.1223	0.7473	0.7402	-0.0071
Shuf.	RTE	0.7184	0.6980	-0.0205	0.6955	0.6992	$+0.0036$

Table 7: Method comparison under the primary shared-configuration protocol. Values are three-seed mean evaluation scores with raw argmax predictions; the difference is LoRA+ minus LoRA. Metrics remain in their native scales.

Order	Task (metric)	LoRA	LoRA+	Difference
Alternating	CoLA (MCC)	0.0071	0.0000	-0.0071
Alternating	MRPC (F1)	0.5407	0.5208	-0.0198
Alternating	SST-2 (Acc.)	0.7840	0.8823	$+0.0982$
Alternating	QNLI (Acc.)	0.6126	0.6551	$+0.0425$
Alternating	QQP (F1)	0.5476	0.5692	$+0.0217$
Alternating	RTE (Acc.)	0.5331	0.5969	$+0.0638$
Shuffled	CoLA (MCC)	0.0000	0.0086	$+0.0086$
Shuffled	MRPC (F1)	0.8503	0.8446	-0.0056
Shuffled	SST-2 (Acc.)	0.9308	0.9266	-0.0042
Shuffled	QNLI (Acc.)	0.7855	0.7747	-0.0107
Shuffled	QQP (F1)	0.6862	0.5639	-0.1223
Shuffled	RTE (Acc.)	0.7184	0.6980	-0.0205

Table 7 retains all six tasks under the primary shared-configuration protocol. Table 6 additionally compares both prediction rules on identical checkpoints.

Table 8 extends the main rank comparison to alternating order. These are descriptive averages over three seeds, not claims of statistical superiority.

A.3 Sensitivity to the Development Selection Criterion

Changing configuration selection also changes the comparison. On alternating QQP with raw predictions, the LoRA+ minus LoRA gap is $+0.0217$, $+0.3848$, or $+0.0325$ when selecting from the same trained grids using shared macro development CE, QQP-specific CE, or QQP-specific F1, respectively. Eight of twelve rank-8 method selections and twelve of twenty-four rank selections occur at learning-rate grid boundaries. These results characterize the tested search spaces; complete selection comparisons appear in Appendix A.3.

Table 9 varies the development learning-rate criterion while retaining the same trained candidate grids and raw prediction rule. Table 10 instead summarizes task-specific selection agreement across all six tasks. For each method, task, seed, order, and prediction rule, a match means that the development-selected learning rate attains the best retrospective evaluation score within the corresponding grid; all evaluation-best ties within 10^{-12} count as matches. Task-metric selection also treats differences within 10^{-12} as ties and favors smaller learning rates. Each prediction rule has its own retrospective comparator. The evaluation-best configuration is never used to select the reported checkpoint.

Table 8: Rank results under both data orders. Each row uses separate learning-rate selection for its rank, seed, and order. Entries are three-seed means.

Order	Rank	Macro dev CE	QQP: argmax	QQP: dev threshold
Alternating	4	0.6244	0.5855	0.5918
Alternating	8	0.6330	0.5476	0.5822
Alternating	16	0.6201	0.4739	0.5946
Alternating	32	0.5938	0.4838	0.6612
Shuffled	4	0.5071	0.5104	0.7365
Shuffled	8	0.4901	0.6862	0.7473
Shuffled	16	0.4706	0.6358	0.7536
Shuffled	32	0.4712	0.5618	0.7532

Table 9: The development selection criterion changes the method comparison. Alternating QQP, raw predictions, three-seed means. The shared criterion averages CE across all six tasks; the other two are QQP-specific.

LR criterion	LoRA F1	LoRA+ F1	Difference
Shared macro CE	0.5476	0.5692	$+0.0217$
QQP CE	0.0000	0.3848	$+0.3848$
QQP F1	0.5845	0.6170	$+0.0325$

A.4 Exploratory Selection Evidence

We next examine whether evidence beyond the current batch improves selection among a fixed candidate set. All loss-based selectors minimize post-update CE on their respective selection data and include no update as a candidate.

Task-balanced selectors using 192 or 768 examples achieve positive mean evaluation-panel gain in all ten groups defined by data order, source method, and rank. Increasing the evidence budget from 192 to 768 examples decreases gain in three of the ten groups, indicating that additional examples do not provide uniform improvement. At a fixed budget of 48 examples, task-balanced evidence achieves higher mean gain than matching the current batch's task composition in eight of ten groups. Both selectors maintain the same class balance, with 24 positive and 24 negative examples (Appendix A.4).

These exploratory results motivate a task-balanced selector with a 192-example evidence budget. Because the exploratory panels were drawn from development data involved in configuration analysis, we freeze the selection procedure before confirmation on unused evaluation examples.

The selection bank and evaluation panel each contain 128 development examples per task. For the sample-budget comparison, each state uses three nested sampling draws: within a draw, the 48- and 192-example task-balanced sets are subsets of the 768-example bank. For the composition comparison, each selector contains 48 examples with exactly 24 examples of each label. One balances task counts; the other matches the current batch's task proportions while preserving label balance. The two composition selectors differ in sampled examples as well as task proportions, so their comparison does not isolate task composition as a unique mechanism. We first average draws within states, then states within trajectories, then seeds within each source group.

Candidates are scored by mean post-update CE. Losses within 10^{-12} of the minimum are treated as tied; ties prefer smaller effective-weight movement and then the fixed order: no update, source fractions $1/16, 1/8, 1/4, 1/2$, raw ratios $1, 16$, and matched ratios $1, 16$. Candidate identities can correspond to nearly identical updates after matching, and the analysis retains the declared identities. No update has zero gain and zero effective displacement.

A.5 Confirmation and Retrospective Controls

Confirmation retains a total budget of 192 selection examples but changes from six tasks to the four tasks with eligible unused examples: SST-2, QNLI, QQP, and RTE, allocating 48 per task. The

Table 10: Task-specific development selection does not uniformly agree with the retrospective evaluation best. Each cell counts matches among 36 method–task–seed comparisons in the declared rank-8 grids. All evaluation-best ties count as matches.

Dev criterion	Prediction	Alternating	Shuffled
CE	Argmax	25/36	24/36
Task metric	Argmax	30/36	22/36
CE	Dev threshold	24/36	20/36
Task metric	Dev threshold	25/36	27/36

Table 11: Exploratory selection evidence. Mean gain on the 768-example evaluation panel. The first three columns use nested task-balanced selection samples. The last two use separate 48-example selectors with exactly 24 labels per class; their task proportions are balanced or matched to the current batch.

Order	Source	$N = 48$	$N = 192$	$N = 768$	Bal. 48	Current 48
Alternating	LoRA $r = 4$	-0.001499	$+0.000278$	$+0.000448$	-0.000987	-0.004011
Alternating	LoRA $r = 8$	-0.000849	$+0.000509$	$+0.000618$	-0.000848	-0.002205
Alternating	LoRA $r = 16$	-0.000780	$+0.000164$	$+0.000403$	-0.000909	-0.001303
Alternating	LoRA $r = 32$	$+0.000226$	$+0.000903$	$+0.001150$	-0.000168	-0.001251
Alternating	LoRA+ $r = 8$	-0.002254	$+0.003455$	$+0.003750$	$+0.000441$	-0.005932
Shuffled	LoRA $r = 4$	$+0.000137$	$+0.000674$	$+0.000294$	$+0.000515$	$+0.000324$
Shuffled	LoRA $r = 8$	$+0.000045$	$+0.000456$	$+0.000421$	-0.000211	-0.000241
Shuffled	LoRA $r = 16$	$+0.005238$	$+0.005766$	$+0.005478$	$+0.004629$	$+0.004528$
Shuffled	LoRA $r = 32$	$+0.005925$	$+0.006290$	$+0.006555$	$+0.006239$	$+0.006500$
Shuffled	LoRA+ $r = 8$	$+0.000315$	$+0.001204$	$+0.001401$	-0.000470	$+0.000496$

disjoint evaluation panel contains 64 examples per task. Both panels are fixed across the 120 source states; the selector is never trained by backpropagation on these panels. Table 4 reports every source group. Ten of the 48 accepted updates have gain below -10^{-4}, corresponding to 20.8% of accepted updates and 8.3% of all sampled states. Mean gain among accepted updates is $+0.005177$.

Scoring all nine candidates uses 1,728 example–candidate evaluations per state for the 192-example selector, compared with 576 for current-loss selection on a 64-example batch. These counts reuse the same examples across candidates. The reported 9.68 and 3.37 seconds exclude candidate construction and target evaluation and are not total deployment costs.

Reassignment controls. Within each trajectory, the primary control averages uniformly over permutations of its four observed candidate choices, including repeated identities. The expected gain is computed exactly from each trajectory's choice frequencies and its saved state-by-candidate gain matrix, and verified by enumerating all $4!$ indexed permutations. The acceptance-location control fixes rejected states at no update and permutes only the identities at accepted states. Trajectories with fewer than two accepted states cannot have their nonzero assignments changed by this control. The best-fixed oracle uses target gains to choose one identity per trajectory and applies that identity at all four states. It need not preserve the original acceptance rate and is distinct from the frequency-preserving controls.

Llama conditional uncertainty. A task-stratified bootstrap resamples the 64 evaluation examples per task, applying the same sampled indices to every source state, candidate, and choice. Across 5,000 draws, the percentile 95% interval for the original mean gain is $[0.000449, 0.003731]$; the interval for its difference from expected within-trajectory reassignment is $[0.001233, 0.002917]$. These intervals describe evaluation-example variation conditional on the four tasks, fixed source states, and fixed selection panel. They do not quantify variation from new training trajectories or newly sampled selection evidence.

Qwen uncertainty and same-step control. Qwen intervals jointly resample the three seed blocks, the five selection-panel draws, and evaluation examples within each task. Each seed block retains all method/order trajectories with that seed, and the four source states within each trajectory remain

Table 12: The original choices outperform within-trajectory reassignment in all three seed aggregates. Each seed includes ten source trajectories and forty sampled states.

Seed	Original gain	Expected reassigned gain
46	$+0.001500$	-0.000112
47	$+0.003067$	$+0.000040$
48	$+0.001645$	$+0.000064$

Table 14: Matched effective-displacement norms do not imply equal utility. Entries are mean gain differences between matched ratio-16 and ratio-1 proposals on standard LoRA source states.

	Alternating			Shuffled
Rank	Current	Next	Panel	Panel
4	-0.004007	-0.001835	+0.001792	+0.000582
8	-0.002607	-0.001731	+0.001138	+0.003184
16	-0.001993	-0.001230	+0.000618	-0.000788
32	-0.002365	-0.001989	+0.000265	-0.002282

together. The same resampled indices are used for all paired policy comparisons. These intervals include variation across the observed seeds and selection draws, unlike the Llama intervals above. For the same-step control, states are grouped by data order, method, and step; candidate identities are reassigned across seeds separately for each selection draw. The control preserves the candidate composition within each such group, rather than within each trajectory.

A.6 Local Transfer and Scaling Diagnostics

Among the 120 full updates in the Llama analysis, 87 improve current-batch loss by more than τ. Of these, 21 increase loss on the next batch, 47 on the next-16-batch window, and 33 on the 768-example evaluation panel (Table 13a). Each candidate is applied once, without further training, before evaluation on these targets. Their composition and temporal positions differ, so the frequencies do not establish a horizon trend.

For harmful full updates, Table 13b reports whether any tested source fraction $c \in \{1/16, 1/8, 1/4, 1/2\}$ changes gain from below $-\tau$ to above τ. This is a retrospective repair test over the stated coefficient set.

Table 13: Local transfer and strict repair on Llama. Negative gain is below $-\tau$; strict repair changes gain from below $-\tau$ to above τ. Denominators count locally improving updates in (a) and harmful updates in (b).

(a) Transfer of local improvement

Target	Negative gain	Fraction
Next batch	21/87	24.1%
Next 16	47/87	54.0%
Panel 768	33/87	37.9%

(b) Strict repair by smaller steps

Target	Repaired/harmful	Fraction
Next batch	3/43	7.0%
Next 16	9/80	11.3%
Panel 192	8/47	17.0%
Panel 768	8/56	14.3%

A.7 Matched Proposal Gains by Rank

Table 14 retains the complete norm-matched comparison moved from the main text. Both proposals are matched to the smaller raw effective-displacement norm; a positive difference need not imply positive absolute gain. The panel contains 768 evaluation examples. At alternating rank 8, the matched source and alternative gains are -0.001906 and -0.000768, respectively: the alternative is better, but both remain harmful. For Qwen, 47 of 48 matched pairs differ in target gain by more than τ, with maximum relative norm residual 3.41×10^{-8}.

A.8 FIXED-STATE PROTOCOL DETAILS

We use 30 source trajectories selected under the configuration protocol described in Section 4: five method/rank settings, two data orders, and three seeds. Four saved states per trajectory, at steps $10, 60, 110, 160$, provide 120 fixed-state comparison points. At each point, candidate updates share the same source parameters, clipped current-batch gradient, AdamW moments, and step count. Recorded parameter and optimizer replay errors are zero.

The candidate set contains nine update choices: no update; source-proposal fractions $c \in \{1/16, 1/8, 1/4, 1/2\}$; proposals with B-to-A learning-rate ratios 1 and 16; and norm-matched versions of both proposals. The original full update uses the trajectory's original factor learning-rate ratio. Alternative proposals are generated by changing the relative learning rates of the two LoRA factors. The B-factor learning rate is held fixed while the A-factor learning rate changes to obtain the alternative ratio. Norm matching scales both factor paths to the smaller of the two raw effective-displacement norms and can therefore reduce either proposal.

Data roles and measurements. For exploratory analyses, selection and evaluation panels are disjoint subsets of the six-task development data. Each panel contains 768 examples, with 128 examples per task. A 192-example evaluation panel is a fixed subset of the larger panel. These examples are separate from gradient-training data but were previously involved in configuration selection; corresponding analyses are therefore exploratory. We also evaluate candidate updates on the next batch and the next 16 batches, applying each update once without further training.

All gains are computed using FP32 loss evaluation on stored model values. A positive or negative event requires gain above τ or below $-\tau$, respectively, with $\tau = 10^{-4}$. Mean gains are computed using all values without thresholding. We average the four states within each trajectory and then average across trajectories. The 120 source states share training histories and evaluation examples and are not independent training replications.