READ TOGETHER, WRITE APART: DECOUPLING ROLLOUT ACCESS FROM UPDATE OWNERSHIP IN MULTI-ROLE LLM TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

When one language model plays solver, verifier, and reviser, "parameter sharing" conflates two choices: which adapters a role generates from, and which adapter receives its gradient. A shared adapter couples reads and writes; role isolation severs both. We introduce DUAL ROUTING, which makes the two routes explicit: the verifier and reviser generate from a common adapter pool, while each role's update trains only the adapter it owns. With seven independently trained Llama-3.1-8B checkpoints per method, the downstream workflows of shared training and of role isolation erase accuracy from their own first answers in every replicate, whereas DUAL ROUTING's does not, and DUAL ROUTING beats capacity-matched role isolation in every replicate, by 26.7 points on average. A preregistered route factorial shows the gain is learned: read exactly as isolated adapters are read, Dual-trained adapters still win by 30.0 points. Further controls identify the mechanism: training downstream roles on reads at the merged pool's reduced scale is what keeps the workflow from collapsing. Of all trained variants, DUAL ROUTING's reviser is the only one that improves on the untrained model both at repairing wrong answers and at preserving correct ones. Collaborative reads with role-owned writes are a simple design point for training multi-role LLM workflows that adds no forward cost.

1 INTRODUCTION

A single language model is increasingly asked to play several roles in one workflow: a solver drafts an answer, a verifier decides whether to keep it, and a reviser repairs rejected cases. Prompting makes these roles cheap to instantiate, but leaves every role at the same pretrained prior. Training the roles should improve specialization, yet it creates a parameter-sharing problem that prompting alone never has to solve (Wu et al., 2024; Hong et al., 2024; Chen et al., 2024; Madaan et al., 2023).

The difficulty is not simply that three losses disagree. "Parameter sharing" conflates two design axes: the adapters used to generate a role's output and the adapter that receives that role's update. These axes can be varied independently even when the textual workflow remains solver–verifier–reviser. Making them explicit is necessary both for implementation and for a controlled comparison.

The two obvious training choices trade off in opposite directions. A shared adapter lets every role use the same learned capabilities, but pushes incompatible solver, verifier, and reviser objectives through the same parameters. Fully isolated adapters protect each update, but also remove any cross-role adapter access. Existing gradient-conflict methods reduce destructive updates inside shared parameters (Yu et al., 2020; Liu et al., 2021); adapter composition methods combine independently learned modules (Pfeiffer et al., 2021; Huang et al., 2024). Neither makes information access and update ownership separate decisions during joint online training.

We introduce DUAL ROUTING, a multi-LoRA interface organized around *shared reading, separate writing*. Each role has a read route specifying which adapters supply its generation weights and a write route specifying which adapter its role-labeled update trains. The evaluated topology is fixed rather than inferred from the workflow graph: the solver uses only its own adapter, whereas the verifier and reviser share the same read pool $\{\mathsf{v}, \mathsf{r}\}$ and write only v and r, respectively. LoRA is the enabling substrate rather than the scientific starting point: a frozen backbone supplies common

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.

capability, while compact low-rank modules can be composed and independently owned (Hu et al., 2022).

We evaluate the read/write principle on 1,472 fixed program-reasoning items under seven trained checkpoints, for 10,304 item–replicate records per method, with backbone, prompts, per-module LoRA shape, horizon, parser, and per-forward active rank matched. Dual Routing beats role isolation, which has the same three role-owned modules, by 26.74 points in every replicate; with the replicate as the unit, the contrasts survive Holm correction. Unlike shared training and role isolation, its workflow does not erase accuracy from its own first answers. Preregistered controls then take the interface apart. A route factorial shows the gain is learned: Dual-trained adapters read exactly as isolated ones still win by 29.96 points. A scale-matched control shows why: training at the merged pool's reduced read scale lifts role isolation from 7.64% to 34.66% in every seed. Conditioning on the first answer, Dual Routing's reviser repairs 20.5% of wrong answers, against at most 8.2% for every baseline and 5.0% for the untrained model, while preserving more correct answers than the untrained reviser.

Key results. Dual Routing beats capacity-matched role isolation by 26.7 points in all seven training replicates, and unlike shared training and role isolation its workflow does not erase its own first answers. A preregistered factorial shows the gain is learned in training (30.0 points with isolated reads), and its reviser is the only one that improves on the untrained model at both repairing and preserving answers.

Contributions.

- We identify a two-sided sharing problem in trainable multi-role LLMs: shared parameters mix objectives, while role isolation removes useful cross-role reads and collapses the downstream workflow.
- We introduce explicit read and write routes over role-specific LoRA modules, enabling collaborative forward computation with role-owned updates at no extra forward cost.
- We show with seven replicates and preregistered controls that the gain is learned in training, and identify reduced read scale during training as the mechanism that prevents downstream collapse.

2 From Parameter Sharing to Dual Routing

2.1 The Two Axes Hidden by "Sharing"

Figure 1 separates access from ownership. Shared training permits every read but assigns every loss to the same update. Role isolation protects updates but removes cross-role access. Gradient projection cannot help that topology, because role-owned updates never share parameters. Dual Routing instead lets roles share a generation-time read pool while each role-labeled update trains only its owned adapter.

2.2 Multi-Role Workflow

Given problem x, solver s produces answer a_s, verifier v returns an accept/revise decision, and reviser r produces a_r; a revision is generated for every episode but becomes the final answer only when the verifier requests it. The final output is

$$a_{\text{final}} = \begin{cases} a_s, & \mathsf{v} \text{ accepts,} \\ a_r, & \mathsf{v} \text{ requests revision.} \end{cases} \tag{1}$$

The roles share an instruction-tuned backbone with frozen weights W_0. For role τ, LoRA represents its update as

$$\Delta W_\tau = B_\tau A_\tau, \quad A_\tau \in \mathbb{R}^{r \times d}, \quad B_\tau \in \mathbb{R}^{d' \times r}. \tag{2}$$

2.3 Read Routes

A read route $\mathcal{R}_\tau$ specifies the adapters visible to role τ. The executed routes are

$$\mathcal{R}_\mathsf{s} = \{\mathsf{s}\}, \qquad \mathcal{R}_\mathsf{v} = \mathcal{R}_\mathsf{r} = \{\mathsf{v}, \mathsf{r}\}. \tag{3}$$

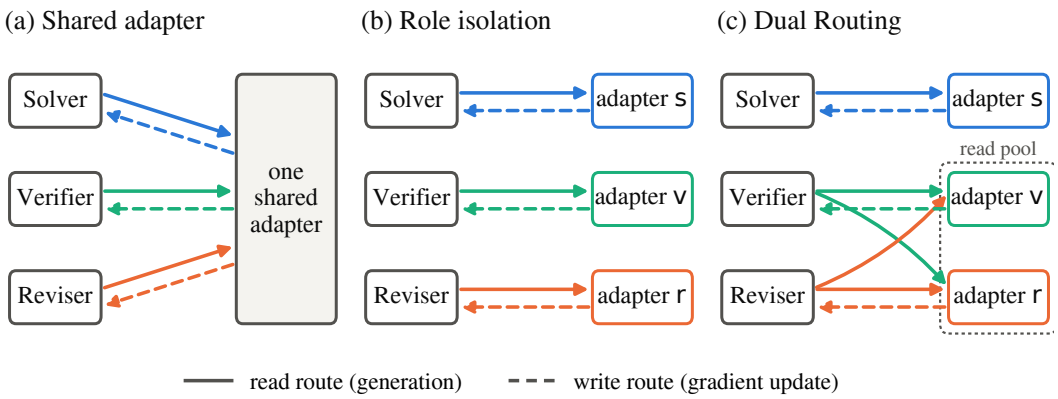


Figure 1: "Parameter sharing" fixes two choices at once: which adapters a role generates from (solid, read) and which adapter its gradient updates (dashed, write). (a) A shared adapter couples both. (b) Role isolation severs both. (c) Dual Routing decouples them: the verifier and reviser generate from a fixed 0.5/0.5 factor-wise merge of $\{\mathsf{v}, \mathsf{r}\}$, while each role's update trains only the adapter it owns.

Table 1: Read and write routes used in the solver–verifier–reviser workflow.

Role	Read route $\mathcal{R}_\tau$	Write route $\mathcal{W}_\tau$
Solver	$\{\mathsf{s}\}$	$\{\mathsf{s}\}$
Verifier	$\{\mathsf{v}, \mathsf{r}\}$	$\{\mathsf{v}\}$
Reviser	$\{\mathsf{v}, \mathsf{r}\}$	$\{\mathsf{r}\}$

The solver layer uses $W_0 h + B_\mathsf{s} A_\mathsf{s} h$. For the verifier and reviser, the runtime separately averages the two low-rank factors,

$$\bar{A}_{vr} = \tfrac{1}{2} A_\mathsf{v} + \tfrac{1}{2} A_\mathsf{r}, \qquad \bar{B}_{vr} = \tfrac{1}{2} B_\mathsf{v} + \tfrac{1}{2} B_\mathsf{r}, \tag{4}$$

and uses

$$h'_\mathsf{v} = h'_\mathsf{r} = W_0 h + \bar{B}_{vr} \bar{A}_{vr} h. \tag{5}$$

This is a fixed factor-wise linear merge, not delta addition: expanding $\bar{B}_{vr}\bar{A}_{vr}$ produces within-adapter and cross-adapter terms.

2.4 Write Routes

A write route $\mathcal{W}_\tau$ specifies which adapters receive gradients. We use $\mathcal{W}_\tau = \{\tau\}$ for every role. Operationally, the rollout adapters determine the weights that generate role-labeled trajectories, while the training adapter determines the module selected for that role's update. Thus verifier trajectories are generated from the shared $\{\mathsf{v}, \mathsf{r}\}$ pool but verifier updates train only v; reviser trajectories use the same pool but reviser updates train only r. No learned router or extra expert network is needed.

For a role loss $\mathcal{L}_\tau$, the write contract is

$$\nabla_{A_j, B_j} \mathcal{L}_\tau = 0 \quad \text{for all } j \neq \tau, \tag{6}$$

where the equation records update ownership. It does not assert that the factor-wise merged rollout is differentiated as one end-to-end autograd graph. This operational split is the core asymmetry: a module can affect trajectory generation without receiving that trajectory's role-labeled update.

2.5 Route Topology Is Not the Composition Operator

Read/write topology and the numerical composition operator are separate design choices. The main panel fixes the factor-wise linear merge in Eq. 4 with weights 0.5/0.5 whenever two adapters are

Table 2: Executable contract for one role trajectory.

Step	Operation
1	Construct generation weights from the role's read route.
2	Generate the role output and record the role-labeled trajectory.
3	Select the role's write-route adapter for optimization.
4	Keep the backbone and all non-owned adapters outside that update.
5	Step the owned factors and restore the next role's routes.

read. At the delta-weight level, block-factor concatenation instead satisfies

$$[B_1 \ \cdots \ B_m] \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix} h = \sum_{j=1}^{m} B_j A_j h, \tag{7}$$

whereas combining A and B separately, as the main runtime does, adds cross terms. The main comparison fixes this factor-wise operator; Section 5 separates the contributions of reads, writes, and read scale, and Appendix A reports a historical operator screen.

2.6 Implementation Contract and Cost

At each role step, the runtime constructs generation weights from the rollout route and assigns optimization to the role's training adapter. The backbone remains frozen throughout. Relative to one LoRA adapter, the method stores one low-rank module per role; it adds no router network or auxiliary routing loss. The solver uses one factor set directly. The verifier and reviser each use a temporary rank-r merge constructed from two factor sets; vLLM then executes that single merged adapter. Single-forward active rank therefore matches the role-isolated and shared controls and does not grow linearly with the number of factor inputs. The extra cost is three-module storage/trainable capacity and merge construction. We do not report an end-to-end latency measurement.

The contract also makes failures auditable. If a non-owned adapter changes during a role-labeled update, the write route is violated. If a verifier or reviser is generated without the configured two-adapter merge, its read route is incomplete. These two checks are more informative than treating every degradation as an undifferentiated multi-task optimization problem.

3 Experimental Protocol

Model and optimization. The controlled panel uses Meta-Llama-3.1-8B-Instruct (Meta AI, 2024) with frozen backbone weights. Each LoRA slot has rank 64, scale 64, zero dropout, and targets the query, key, value, output, gate, up, and down projections. All jobs use GDPO with train batch 8, two rollouts per prompt, and components for pipeline outcome, reasoning format, checker calibration, and gate advantage (weight 0.5). AdamW uses learning rate 10^{-5}, weight decay 0.01, $\beta = (0.9, 0.999)$, a constant schedule, and unit clipping. One PPO epoch uses minibatch 128 and per-GPU microbatch 8; prompt/response caps are 8192/2048. Training samples at temperature 1; evaluation is deterministic with one candidate per role. We evaluate global step 100 for every method and replicate, a common fixed checkpoint rather than a method-specific best checkpoint.

Data and observation count. The evaluation panel contains 1,472 fixed, aligned items: 512 interval-union, 64 maximum-subarray, 192 range-sum, and 704 unweighted-shortest-path problems. All are generated in-project from four deterministic templates; the panel has no third-party benchmark rows. Training uses a 496-item mixed curriculum, including 80 APPS examples (Hendrycks et al., 2021). Its separate 184-item training-time validation mix includes 24 APPS and 32 ZebraLogic examples (Lin et al., 2025); ZebraLogic is used only for this non-gradient validation. All other rows in these mixes are generated in-project. We evaluate that same panel under independently trained checkpoints with seeds 42–48. Thus each method contributes 1,472 unique evaluation items and 10,304 item–replicate observations. We use a fixed parser and stop-aware final-answer policy, and hold the data, initialization family, per-module LoRA shape, training horizon, prompts, and generation settings constant.

Methods. Every job instantiates three same-shape adapter slots so that checkpoint and runtime interfaces remain constant. *Shared* routes all roles through one active adapter. *Role-isolated* and *Dual Routing* both use three role-owned modules, but only Dual Routing gives the verifier and reviser the shared $\{\mathsf{v}, \mathsf{r}\}$ read pool in Table 1. Role isolation is therefore the closest capacity-matched contrast. *PCGrad-A* and *PCGrad-B* are configured as role isolation plus a gradient-projection step (PCGrad; the second configuration was labeled CAGrad but its branch called the same projector). With role-owned adapters, however, each role's loss produces gradients only on its own adapter, so any two role gradients have disjoint support and zero inner product, and PCGrad's projection, which fires only on a negative inner product, never changes the update. We verified this on the training code path: across all role pairs the inner product is exactly zero, no projection fires, and the resulting update equals plain role-isolated accumulation exactly (supplementary `verify_pcgrad_noop.py`, `ff5434a9d474`). The two configurations are therefore additional role-isolated run families, and we report them as such (seed-blocked Friedman test across the three families, $p = 0.62$). Dual Routing uses the fixed 0.5/0.5 factor-wise merge throughout.

Metrics. Final accuracy is correctness after the accept/revise decision and is the primary endpoint. Solver accuracy evaluates first answers. System gain is final minus solver accuracy within the same run. Every episode generates a revision, so redo accuracy is the accuracy of that revision over all episodes, including those where the verifier kept the solver answer. Stop is the fraction of all episodes that retain the solver answer. False acceptance and false rejection are, respectively, the fractions of all episodes with an incorrect retained solver answer and a correct solver answer sent to revision. Verdict parsed is the fraction of all episodes with a valid verifier decision. We report paired bootstrap intervals and results for each training replicate.

Pairing and uncertainty. Every method is evaluated on aligned item identifiers, so contrasts are paired within an item and replicate. Pooled intervals independently resample the 10,304 aligned item–replicate records; per-replicate intervals resample 1,472 items. The pooled procedure preserves method pairing but does not cluster a repeated item across checkpoints. Its intervals are therefore record-level uncertainty conditional on seven realized checkpoints, not seed-population uncertainty. We report seed-population uncertainty separately, taking the training replicate as the unit of analysis: each replicate contributes one mean delta, giving $n = 7$ observations per contrast. Because n is small we report a distribution-free sign test, a Wilcoxon signed-rank test, a paired t-test on replicate means, and a bootstrap over replicates rather than records, and treat agreement among them as the evidence (Appendix D). It matters most for the shared comparison, where the two units disagree.

Why system gain is central. Final accuracy mixes the quality of the initial solver with the value added by the two downstream roles. System gain subtracts solver accuracy from final accuracy within the same item–replicate panel, asking whether the downstream workflow earns back its intervention. A method can therefore have a capable solver and a negative gain if verification destroys good answers. Because system gain also depends on the solver's starting point and available headroom, we treat it as a secondary workflow diagnostic and final accuracy as the primary endpoint. Reporting both prevents improvements in one role from concealing a destructive workflow.

4 Results

4.1 Collaborative Reads with Role-Owned Writes

Table 3 pools the 10,304 item–replicate observations. Dual Routing is the only method with positive system gain: final accuracy is 1.99 points above solver accuracy, whereas every control is negative. Per replicate, every control's gain is negative in all seven runs, whereas Dual Routing's ranges from -0.14 to $+8.22$ and is positive in five. The most direct comparison is role isolation, which keeps three role-owned modules but removes verifier–reviser cross-reading. PCGrad-A and PCGrad-B are two more role-isolated run families (Section 3), so together the isolated arm has 21 checkpoints. Dual Routing's reviser is also the only trained reviser that beats the untrained backbone's reviser both at repairing wrong first answers (20.5% versus 5.0%) and at keeping correct ones correct (76.9% versus 74.6%).

Table 3: Controlled program-reasoning panel: 1,472 fixed items under seven training replicates (10,304 item–replicate observations per method). All values are percentages; gain is final minus solver within each method.

Method	Final	Solver	System gain	False reject	Verdict parsed
Shared adapter	30.21	46.47	−16.26	40.33	98.78
Role-isolated	7.73	25.00	−17.27	22.21	97.78
PCGrad-A	4.49	32.70	−28.20	32.00	84.26
PCGrad-B	1.81	35.32	−33.51	33.83	98.92
Dual Routing	**34.46**	32.47	**+1.99**	**11.39**	94.88

Table 4: Downstream behavior on the same pooled item–replicate observations. Redo acc. is the accuracy of the generated revision over all episodes, including those whose solver answer was kept; stop is the fraction of episodes that retain the solver answer. FA and FR are the fractions of all episodes with, respectively, an incorrect retained solver answer and a correct solver answer sent to revision.

Method	Redo acc.	Stop	FA	FR	Verdict parsed
Shared adapter	28.55	12.64	6.50	40.33	98.78
Role-isolated	5.03	6.85	4.07	22.21	97.78
PCGrad-A	4.11	1.81	1.11	32.00	84.26
PCGrad-B	0.36	3.72	2.23	33.83	98.92
Dual Routing	**39.52**	**59.37**	38.29	**11.39**	94.88

Dual Routing rejects far fewer correct answers than any control (Figure 2) and turns the complete pipeline positive; we evaluate the workflow, not verdict accuracy or either error rate alone (Appendix I).

The reviser difference does not come from a weaker solver. Conditioning on the first answer removes solver strength (Figure 3a,b): when the solver is wrong, Dual Routing's revision is correct 20.5% of the time versus 0.2–8.2% for every control, and when the solver is right its revision stays right 76.9% of the time versus 0.7–51.9%. Both gaps are positive in all seven replicates against every control, shared included (seed-level $p \leq 0.007$; Appendix I). Isolated revisers destroy nearly every correct answer they touch, which is why their workflows collapse.

4.2 Paired Comparisons

Dual Routing improves final accuracy over role isolation, PCGrad-A, and PCGrad-B by 26.74, 29.97, and 32.66 points, respectively (Table 5); the conditional record-level intervals are positive. Because seven replicates make the seed a usable unit of analysis, we also test these three contrasts with the replicate as the observation ($n = 7$). All three are positive in every replicate, so the one-sided sign test attains its smallest attainable value at this sample size ($p = 1/128$); the paired t-test on replicate means gives $p \leq 7.2 \times 10^{-4}$, still below 0.05 after Holm correction across the three-contrast family, Wilcoxon agrees at its exact floor ($p = 0.016$), and effect sizes are large (d_z of 2.40, 2.69, 4.80). Seed-level intervals are correspondingly wider: role isolation is $+26.74$ with a 95% t interval of $[+16.42, +37.06]$ over seeds versus $[+25.74, +27.77]$ over records (Appendix D).

Against the shared adapter, Dual Routing is ahead by $+4.25$ points ($[+3.04, +5.43]$ over records; 5 of 7 replicates; seed-level $[-5.82, +14.32]$). The strongest controlled result is the improvement over role isolation in every replicate, and Dual Routing also exceeds each matched-seed run of the two further isolated families, by at least 7.47 points.

Role isolation has the same number and shape of role-owned modules as Dual Routing, yet its downstream stages erase 17.27 points once verifier–reviser cross-reading is removed. PCGrad-A and PCGrad-B repeat that layout with different random streams; their lower means are run-to-run variation.

The workflow difference against the shared adapter is decisive: Dual Routing's system gain exceeds Shared's in all seven replicates ($+5.50$ to $+36.28$ points; seed-level t interval $[+7.92, +28.57]$,

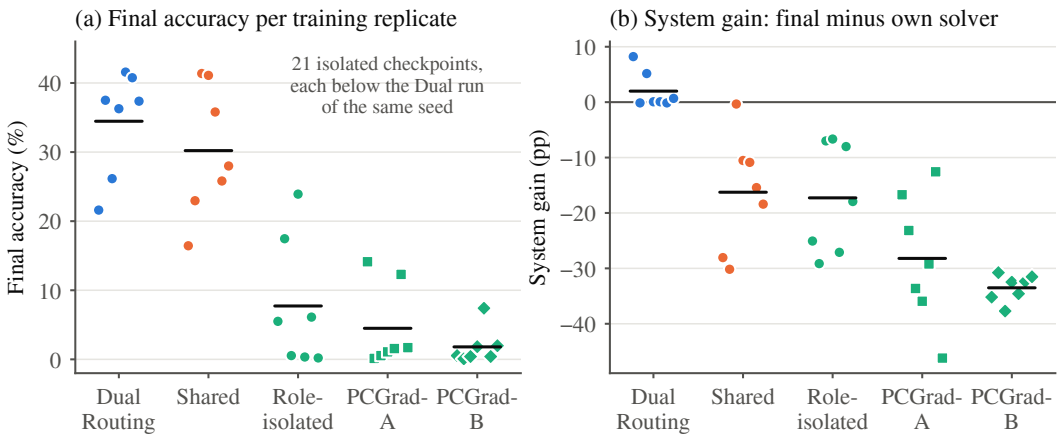


Figure 2: Every training replicate, not just the pooled mean. Each dot is one checkpoint on the same 1,472 items; bars are method means. (a) Dual Routing's final accuracy exceeds the isolated run of the same seed in all 21 comparisons (role isolation and the two PCGrad-labeled run families, which reduce to role isolation; Section 3). (b) Every control's workflow loses accuracy relative to its own first answers in all seven replicates, whereas Dual Routing's stays at or above zero on average.

Table 5: Paired final-accuracy differences on 10,304 item–replicate observations from seven training replicates. Intervals resample item–replicate records conditional on the realized checkpoints.

Dual Routing minus	Δ final	95% interval
Shared	+4.25	[+3.04, +5.43]
Role-isolated	**+26.74**	[+25.74, +27.77]
PCGrad-A	**+29.97**	[+28.98, +30.96]
PCGrad-B	**+32.66**	[+31.71, +33.61]

Appendix D). The shared adapter begins from a stronger solver and erases 16.26 points of it downstream.

4.3 Where the Gain Appears

The source breakdown is positive against role isolation, PCGrad-A, and PCGrad-B on all four program families (Figure 3c). Against role isolation, final gains range from +9.46 points on interval-union length to +52.68 on maximum-subarray sum. The error profile also changes qualitatively: Dual Routing reduces false rejection by 10.82 points relative to role isolation, 20.60 relative to PCGrad-A, and 22.44 relative to PCGrad-B.

No single family creates the pooled advantage over the isolated runs: the two largest families gain +9.46 and +35.61 points. The same step-100 checkpoints were also scored on a separate 184-item validation mix that shares no question with the panel and adds six task families, including APPS and ZebraLogic: Dual Routing beats every isolated run family in all seven replicates there (+7.69 to +9.86 points), and beats role isolation by +3.67 on its 152 non-template items (Appendix H).

Most item–replicate comparisons tie on final correctness (Table 9, Appendix C). Among changed outcomes, wins outnumber losses by factors of 7.3, 12.1, and 38.0 against role isolation, PCGrad-A, and PCGrad-B, and wins also outnumber losses against shared training (2,212 versus 1,774).

5 Analysis

What the update is. Verifier and reviser roll out from the merged pool; the policy loss is then recomputed with only the owned adapter active, and both the old and the new log-probabilities in the ratio come from that owned adapter, so reading shapes the trajectories a role learns from without granting it write access to another role's adapter. Each update is one PPO epoch over a single

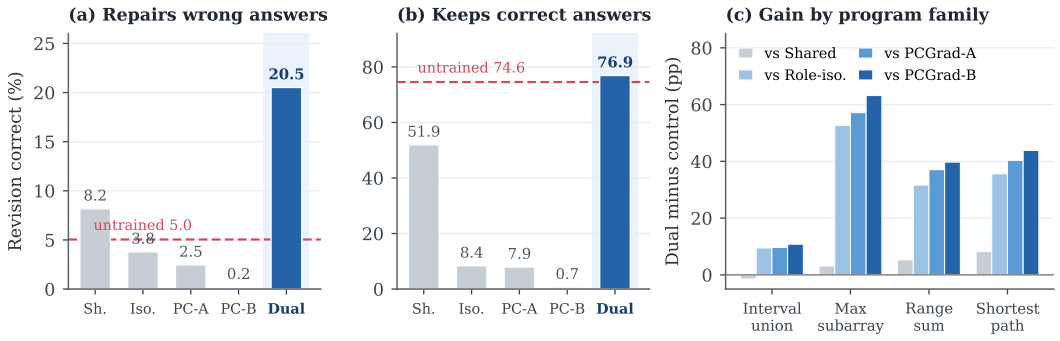


Figure 3: Where Dual Routing's advantage comes from. (a,b) Revision accuracy at fixed first-answer correctness, mean over seven replicates; the dashed line is the untrained backbone's reviser. Only Dual Routing's reviser improves on it in both. (c) Dual Routing's final-accuracy gain over each control on each program family (Table 11).

Table 6: Preregistered controls, seeds 45–48 (final accuracy, percent; mean over seeds, with the number of seeds in which Dual Routing is ahead).

Control	Changes from Dual Routing	Final	Dual ahead
Dual Routing	—	**38.99**	—
Isolated-trained, merged reads	training route	30.04	4/4
Dual-trained, isolated reads	evaluation route	37.60	2/4
Shared downstream writes	write route	34.04	4/4
Own-only reads, merge scale	cross-role access	34.66	3/4
Role isolation	reads and scale	7.64	4/4

minibatch, so the ratio equals one and clipping never binds (logged PPO-KL and clip fraction are zero in all 4,007 updates of the 42 Llama runs). Each role therefore takes an advantage-weighted likelihood step for its own adapter on trajectories that a shared behavior policy generated, the same surrogate that every baseline uses for its own rollouts. Because role-owned updates never share parameters, gradient-conflict methods have nothing to act on; what each role reads at rollout is the remaining lever.

What each route contributes. Preregistered controls on seeds 45–48 take the interface apart (Table 6, Figure 4). First, the gain is learned in training: read exactly as role isolation reads them, Dual-trained adapters reach 37.60% against 7.64% for isolated-trained ones, positive in all four seeds ($+29.96$, $p = 0.014$; Appendix E), and their reviser, read alone, keeps 71.5% of correct first answers correct versus 12.6%. Second, read scale is the mechanism: role isolation trained with its own adapter at the merge's reduced scale rises from 7.64% to 34.66% (4/4; Appendix G), so training on the smaller merged delta is what keeps downstream roles from destroying first answers. Third, with the reads or the scale held fixed, owned writes and cross-role reads each add further accuracy for the paper's Dual checkpoints ($+4.96$ and $+4.33$ points; Appendices F and G).

6 Related Work

Parameter-efficient adaptation and composition. LoRA and adapters provide compact task-specific updates (Hu et al., 2022; Houlsby et al., 2019); AdapterFusion, LoRAHub, LoRARetriever, Polytropon, and LoRI combine, select, or prepare modules for later merging (Pfeiffer et al., 2021; Huang et al., 2024; Zhao et al., 2024; Ponti et al., 2022; Zhang et al., 2025). Dual Routing instead separates a fixed role-indexed read pool from single-adapter update ownership.

Gradient conflict and modular experts. PCGrad, CAGrad, GradNorm, and related multi-task methods reshape gradients in a shared parameter space (Yu et al., 2020; Liu et al., 2021; Chen et al., 2018; Shi et al., 2023). They need overlapping parameters to act on: once each role owns its adapter,



Figure 4: Preregistered route controls on seeds 45–48 (bars: mean final accuracy; dots: individual seeds). Each control changes one element of Dual Routing; role isolation collapses, and every partial variant stays below the full interface on average.

role gradients are disjoint and projection is a no-op, so the remaining design variable is what each role reads at rollout. Mixture-of- experts models route tokens to parameter regions (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Our route unit is a workflow role rather than a token, and read and write routes are explicit instead of learned.

Inference-time verification. Concurrent anonymous work studies consequence-aware accept/revise control (Anonymous, 2027). That work changes the inference-time decision over generated answers; DUAL ROUTING changes how role parameters are trained.

7 CONCLUSION

Dual Routing separates rollout reads from update ownership. In a three-role program-reasoning setting it beats capacity-matched role isolation in every replicate, unlike shared training and role isolation its workflow does not erase its own first answers, and it trains the only reviser that improves on the untrained model at both repairing and preserving answers. Preregistered controls show that the gain is learned in training and identify its mechanism: training downstream roles on reads at the merged pool's reduced scale. Collaborative reads with role-owned writes are a simple design point for training multi-role LLM workflows, with the same per-forward active rank as its baselines.

8 LIMITATIONS

The controlled comparison uses one verifiable program-reasoning panel with four task templates and one fixed step-100 checkpoint per run; the development validation mix adds six task families and shows the same ordering against the isolated runs. On a second backbone, Qwen2.5-7B-Instruct, whose solver is already strong, Dual Routing reproduces the lower false rejection with a smaller final-accuracy margin (Appendix J). Routes are pre-specified and symmetric, the merge operator is fixed, and learned or directional topologies remain open. Seven replicates make the seed a usable unit of analysis, but seed-level intervals are wider than record-level ones, and three added Dual seeds use second training runs; Appendix I repeats every contrast with the first runs. The configurations labeled PCGrad reduce to role isolation under role-owned writes, so gradient surgery on overlapping parameters is not evaluated here.

ETHICS STATEMENT

Separating role updates can improve a reasoning workflow, but it does not make its answers reliable or safe. A verifier may still accept an incorrect answer or trigger a harmful revision, so the method should not be treated as a high-stakes assurance mechanism without task-specific validation. Our study uses author-generated synthetic tasks, APPS, ZebraLogic, and model-generated outputs rather

than newly recruited participants. The APPS paper specifies CC BY-SA 3.0 for its data, whereas the CodeParrot mirror metadata says MIT; the ZebraLogic card provides citation metadata but no explicit license or usage terms; we do not claim redistribution or relicensing rights. We limit both datasets to non-redistributed research use, report negative and heterogeneous results, and redistribute neither benchmark rows, model weights, nor derived checkpoints. Llama use remains subject to the Llama 3.1 Community License and Acceptable Use Policy; Qwen2.5, used only in the historical screen, is Apache-2.0. The aggregate evidence and original consistency code are distributed under Apache-2.0. All authors have read and adhere to the ICLR Code of Ethics.

Reproducibility Statement

The read and write routes, the merge operator, and the executable per-role contract are specified in Section 2, and the backbone, optimization settings, fixed evaluation panel, replicate protocol, and pairing and bootstrap procedures are given in Section 3. Appendix L states the reproduction requirements, including the fixed item panel, the common checkpoint step, and the aggregation rules behind every reported table; Appendix A separately bounds the historical composition screen so it is not read as protocol-matched evidence. Anonymized per-item evaluation outcomes and a script that rebuilds every reported table from them are provided as supplementary material.

AI Use Statement

We used generative AI tools to assist with writing, coding, and data analysis. The authors reviewed all AI-assisted content and take full responsibility for this paper.

References

Anonymous. When revision backfires: Consequence-aware verification for multi-role LLM reasoning. Manuscript under review, 2027.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024. arXiv:2308.10848.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2105.09938.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. arXiv:2308.00352.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, et al. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling*, 2024. arXiv:2307.13269.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, et al. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. arXiv preprint arXiv:2502.01100, 2025. URL https://arxiv.org/abs/2502.01100.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, 2021.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Meta AI. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rueckle, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2021.

Edoardo M. Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining modular skills in multitask learning. arXiv preprint arXiv:2202.13914, 2022.

Qwen Team. Qwen2.5: A party of foundation models! https://qwenlm.github.io/blog/qwen2.5/, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. Recon: Reducing conflicting gradients from the root for multi-task learning. In *International Conference on Learning Representations*, 2023. arXiv:2302.11289.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling*, 2024. arXiv:2308.08155.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. LoRI: Reducing cross-task interference in multi-task low-rank adaptation. In *Conference on Language Modeling*, 2025. arXiv:2504.07448.

Ziyu Zhao, Leilei Gan, Guoyin Wang, et al. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Table 7: Historical composition audit, separate from the controlled routing result. "Add" and "concat" name the audited implementation paths, not a universal algebraic operator ranking.

Screen	Concat wins	Sign p
All matched settings	8/9	0.0195
Both runs ≥ 150 steps	6/7	0.0625
Long-run mean	Add -0.583	Concat $+0.129$

A Historical Composition Screen

Protocol. The composition screen is separate from the controlled routing panel. It contains 36 historical rows from a Qwen2.5-7B-Instruct study (Qwen Team, 2024), matched back to their source records. Nine configuration-matched pairs compare implementation paths labeled additive and concatenative under source-defined late-training windows. The study was not randomized, and the displayed legacy means did not share one recoverable aggregation rule. We therefore retain the source-matched signs and a minimum-run-length sensitivity analysis rather than combine the legacy means as if they came from a prospective factorial.

Historical matched result. Among the nine matched pairs, the path labeled concatenative has higher late-training system gain in 8/9 cases (one-sided exact sign $p = 0.0195$). Requiring both runs to have at least 150 logged steps leaves 6/7 pairs ($p = 0.0625$). After removing two short severe additive runs, additive mean late gain is -0.583 points, versus $+0.129$ for the long-run concatenation rows. Table 7 preserves this historical audit in full, but none of these quantities enters the headline routing result.

Operator and checkpoint boundary. The labels in this archive do not denote two guaranteed algebraic reparameterizations of the same delta. Inspection of one historical path labeled additive found that it combines low-rank A and B factors separately, which can introduce cross terms; the concatenative path also changes structural rank, execution, and effective output scale. Later strict cross-evaluation exhibits a composition-by-checkpoint interaction rather than one operator winning at every checkpoint. On Llama checkpoints, a post hoc 0.25 output-scale probe materially repairs concatenative parsing and final accuracy, further showing that the unscaled comparison mixes operator choice with calibration.

Backbone and scale boundary. A later complete common-interval diagnostic covers both Qwen and Llama checkpoints. The ordering among routing and shared-gradient controls reverses across backbones, even when the implemented additive path exceeds the concatenative path on that panel. Here "additive" is the historical run label; the inspected path is a factor-wise merge. Together, these observations make the historical 8/9 sign pattern operator-, checkpoint-, backbone-, and scale-dependent. It is useful as provenance for why composition must be specified exactly, but it does not establish concatenation as universally more stable, identify an optimizer-state mechanism, or strengthen the evidential scope of the main routing comparison.

B Additional Design and Cost Details

Evaluated topology and alternatives. The evaluated routes do not follow the directed textual workflow. The solver reads only s, while verifier and reviser share the symmetric $\{\mathsf{v}, \mathsf{r}\}$ pool. A graph-derived topology in which later roles read solver or other predecessor adapters is a plausible alternative, as are directional peer edges, round-indexed debate, and learned routes. These are design hypotheses, not evaluated scaling claims.

Intended regime. We expect read/write separation to be most relevant when roles share a backbone and can benefit from a cross-role behavior-policy pool. It is less likely to help when roles are independent, use separate full models, or expose all relevant information directly in text. The present paper tests only one three-role topology and does not claim that it is optimal elsewhere.

Table 8: Factor inputs and update ownership in the evaluated topology.

Role	Factor inputs	Writes	Invoked
Solver	1	1	Every item
Verifier	2	1	Every item
Reviser	2	1	Every item; used if rejected

Training objective. GDPO normalizes each reward component within its group and sums the normalized advantages before a batch-level normalization. The four components are the pipeline's final correctness (grouped by prompt), a format score (by prompt), a calibration term that moves the verifier's accept rate toward the batch's correct-answer rate (by prompt), and, with weight 0.5, the verifier's counterfactual action value, $S - R$ when it accepts and $R - S$ when it rejects (grouped by role and turn). Every role's update therefore sees the pipeline outcome; only the verifier sees the action-value term. The components and weights are identical for all five methods.

Parameter accounting. At one projection, a rank-r LoRA module with input width d and output width d' adds $r(d + d')$ trainable parameters. Dual Routing, role isolation, PCGrad-A, and PCGrad-B each use three same-shape stored/trainable role modules and are capacity matched; Shared trains one module in the same three-slot checkpoint interface. Each forward activates one rank-r adapter: a direct role module, the common Shared module, or a temporary rank-r factor merge. The main comparison matches per-module rank, per-forward active rank, and projection set across methods, but not total trainable capacity for Shared.

The solver uses one stored factor pair directly. Verifier and reviser each form one temporary rank-r factor pair from two stored factor pairs, so the adapted layer forward remains rank r after merge construction. Policy-loss recomputation and optimizer state involve only the owned factors in $\mathcal{W}_\tau$. We report these structural costs rather than an unmeasured latency or throughput estimate.

C Paired Workflow Contrasts

Table 9: Item–replicate wins/ties/losses for Dual Routing.

Comparator	Wins	Ties	Losses
Shared	2,212	6,318	1,774
Role-isolated	3,192	6,675	437
PCGrad-A	3,367	6,658	279
PCGrad-B	3,456	6,757	91

Table 10 reports the same 10,304 item–replicate contrasts as the main paired table, with system-gain intervals and false rejection. System-gain intervals are positive against all controls.

D Seed-Level Inference

Table 13 treats the training replicate as the unit of analysis. Each replicate contributes one mean paired delta, giving $n = 7$ observations per contrast, so these tests generalise over training seeds rather than conditioning on the realized checkpoints. Because $n = 7$ is small we report four procedures with different assumptions and treat their agreement as the evidence: an exact one-sided sign test (distribution-free, but floored at $1/128 = 0.0078$ at this sample size), an exact two-sided Wilcoxon signed-rank test (floored at 0.0156), a paired t-test on replicate means, and a percentile bootstrap that resamples replicates rather than records. Holm adjustment is applied across the three capacity-matched contrasts; the shared reference, which is not capacity-matched, is reported alongside rather than inside that family.

All four procedures agree on all three capacity-matched contrasts and disagree on none. The shared contrast is the informative exception: it is the only one whose record-level and seed-level intervals reach different conclusions, and we report it as unresolved.

Table 10: Paired workflow contrasts over 10,304 item–replicate observations. Intervals are conditional record-level bootstraps.

Dual Routing minus	Δ final [95% interval]	Δ system gain [95% interval]	Δ false rejection
Shared	$+4.25$ [$+3.06, +5.45$]	$+18.25$ [$+17.20, +19.33$]	-28.94
Role-isolated	$+26.74$ [$+25.73, +27.75$]	$+19.26$ [$+18.23, +20.31$]	-10.82
PCGrad-A	$+29.97$ [$+28.97, +30.97$]	$+30.19$ [$+29.10, +31.26$]	-20.60
PCGrad-B	$+32.66$ [$+31.72, +33.59$]	$+35.50$ [$+34.39, +36.61$]	-22.44

Table 11: Dual Routing final-accuracy differences by source family, pooled over training replicates. Differences from role isolation, PCGrad-A, and PCGrad-B are positive for every family; the shared comparison is positive on three of four.

Program family	minus shared	minus isolated	minus PCGrad-A	minus PCGrad-B
Interval-union length	-1.45	$+9.46$	$+9.65$	$+10.80$
Maximum-subarray sum	$+3.12$	$+52.68$	$+57.14$	$+63.17$
Range-sum queries	$+5.28$	$+31.62$	$+37.05$	$+39.73$
Unweighted shortest path	$+8.22$	$+35.61$	$+40.34$	$+43.85$

The unadjusted t-test p-values are 0.34, 7.2×10^{-4}, 3.8×10^{-4}, and 1.1×10^{-5} for shared, role isolation, PCGrad-A, and PCGrad-B. The seed bootstrap gives $[-3.60, +11.13]$, $[+19.17, +34.31]$, $[+21.53, +36.36]$, and $[+27.70, +36.95]$; it is narrower than the t interval because the percentile method does not inflate for the small-sample tail, and we quote the wider t interval in the table for that reason. The sign test reaches 0.0078 for all three capacity-matched contrasts and 0.2266 for shared.

The same replicate-level procedure applied to the workflow endpoints of Table 10 separates the two questions the shared contrast mixes (Table 14). System gain, final minus the method's own solver accuracy, favors Dual Routing in all seven replicates against every control, shared included: $+18.25$ points with a seed-level t interval of $[+7.92, +28.57]$ ($p = 0.005$). What stays unresolved against shared is therefore final accuracy, not what the downstream workflow does: the shared adapter starts from a stronger solver (46.47% versus 32.47% pooled) and then loses 16.26 points to its own verifier–reviser stages. False rejection is lower under Dual Routing in all seven replicates against shared and the PCGrad-A and PCGrad-B run families, and in five of seven against role isolation, whose seed-level interval is correspondingly wider.

E Training-Route by Evaluation-Route Factorial

Dual Routing changes the read route both while the adapters are trained and when they are evaluated. To separate the two, we preregistered a factorial on the four added seeds, 45–48, before any cross-routed panel existed. Each step-100 checkpoint is evaluated with its own read routes and with the other method's: role-isolated adapters read through the merged $\{\mathsf{v}, \mathsf{r}\}$ pool, and Dual-trained adapters read in isolation, each role using only its own adapter. Nothing is retrained. The panel, parser, and decoding are the paper's, and all eight new evaluations completed.

The training route. Read in isolation, Dual-trained adapters beat isolated-trained ones by $+29.96$ points (4/4 seeds; seed-level t interval $[+11.77, +48.15]$, $p = 0.014$). Part of this is a stronger solver: every role's reward includes the pipeline outcome, so the solver trained alongside a non-destructive downstream is itself better ($+14.35$). The downstream roles improve even after conditioning on the first answer: the Dual-trained reviser, read alone, repairs 18.7% of wrong answers against 5.8% ($+12.88$, $[+1.95, +23.81]$) and keeps 71.5% of correct ones correct against 12.6% ($+58.95$, $[+30.67, +87.24]$), 4/4 seeds each. Under merged reads the training-route difference shrinks to $+8.95$ and its interval includes zero ($[-2.76, +20.66]$, still 4/4 positive).

The evaluation route. Reading Dual-trained adapters through the merge changes final accuracy by only $+1.39$ ($[-3.41, +6.20]$, 2/4); it mainly shifts the verifier's operating point toward

Table 12: Per-replicate final-accuracy differences for Dual Routing under seven training replicates. Each training seed is evaluated on the same 1,472 fixed items. Brackets are paired 95% item-level bootstrap intervals conditional on that checkpoint.

Seed	minus shared	minus role-isolated	minus PCGrad-A	minus PCGrad-B
42	+5.16 [+2.38, +7.95]	+16.10 [+13.72, +18.48]	+7.47 [+4.82, +10.12]	+21.06 [+18.95, +23.17]
43	+14.54 [+11.62, +17.53]	+20.04 [+17.05, +22.96]	+37.36 [+34.92, +39.88]	+37.43 [+34.99, +39.95]
44	−15.22 [−18.41, −12.02]	+25.61 [+23.37, +27.92]	+25.61 [+23.23, +27.92]	+25.75 [+23.57, +28.12]
45	−4.82 [−8.29, −1.43]	+12.36 [+9.24, +15.56]	+35.19 [+32.68, +37.70]	+34.44 [+31.86, +36.96]
46	+5.77 [+2.58, +8.97]	+41.24 [+38.72, +43.75]	+40.01 [+37.43, +42.53]	+34.17 [+31.32, +36.96]
47	+14.95 [+11.75, +18.07]	+34.65 [+32.00, +37.36]	+28.46 [+25.54, +31.39]	+40.35 [+37.84, +42.87]
48	+9.38 [+6.32, +12.50]	+37.16 [+34.71, +39.67]	+35.67 [+33.15, +38.32]	+35.39 [+32.88, +37.98]

Table 13: Seed-level inference with the training replicate as the unit of analysis ($n = 7$). Sign-test p is one-sided; Wilcoxon and t are two-sided. Holm adjustment covers the three capacity-matched contrasts only. Record-level intervals from Table 5 are repeated for comparison.

Dual Routing minus	Pos.	Mean	95% CI (seeds)	95% CI (records)	d_z	Holm p (t)
Shared	5/7	+4.25	[−5.82, +14.32]	[+3.04, +5.43]	+0.39	–
Role-isolated	7/7	**+26.74**	[+16.42, +37.06]	[+25.74, +27.77]	+2.40	0.00077
PCGrad-A	7/7	**+29.97**	[+19.68, +40.26]	[+28.98, +30.96]	+2.69	0.00077
PCGrad-B	7/7	**+32.66**	[+26.37, +38.94]	[+31.71, +33.61]	+4.80	0.00004

keeping answers. Reading isolated-trained adapters through the merge, by contrast, adds +22.40 ([+7.97, +36.83], 4/4): averaging two individually destructive role adapters dampens both.

F Write-Route Control

To test the write half of the interface directly, we preregistered a control that keeps Dual Routing's reads exactly (verifier and reviser generate from the 0.5/0.5 factor-wise merge of $\{\mathsf{v}, \mathsf{r}\}$) but shares the downstream writes: both roles' updates train both v and r. Everything else follows the paper's protocol (100 steps, rank 64, the same prompts and rewards, the frozen step-100 checkpoint, and the same 1,472-item panel), at seeds 45–48 (Table 16).

Owned writes win in all four seeds. The shared-write runs lose most of that accuracy in the first answer (solver 30.67% against 38.82%): every role's reward includes the pipeline outcome, and when downstream updates are shared the solver trains alongside a less stable downstream stage.

G Read-Scale and Cross-Term Controls

The factor-wise merge does three things at once: it lets each downstream role read the other role's adapter, it shrinks the effective delta (measured on one seed, the merged delta has 0.43–0.61 of the norm of either owned delta), and it adds cross terms, which are as large as the within-adapter terms because the two adapters' A factors are nearly orthogonal. We preregistered two read routes that separate these, before any of their panels existed. *Own, half* reads the role's own adapter at half scale: no other role and no cross terms. *Delta average* reads $\frac{1}{2}(\mathsf{B_v A_v} + \mathsf{B_r A_r})$: the other role's adapter without cross terms. We apply them at evaluation to both trained arms, and we also *trained* role isolation with the own-half read route, a control with Dual Routing's read scale and rollout/update mismatch but no access to the other role (Table 17).

Three results. First, reading at the merge's scale is what keeps isolated adapters from collapsing: halving their own delta at evaluation alone adds +12.23 points ([+4.65, +19.81], 4/4), and training role isolation at that scale lifts it from 7.64% to 34.66% (4/4). Second, Dual-trained adapters are robust to how they are read: own, delta-average, and factor-merge reads all land within 1.4 points. Third, at matched scale, reading the other role's adapter adds +4.33 points over reading only one's own (+5.57, +1.63, +10.26, −0.12), and the factor merge adds another +1.46 over the delta average for Dual-trained adapters.

Table 14: Seed-level inference on the workflow endpoints ($n = 7$ replicates). Positive values favor Dual Routing: system gain is higher, and false rejection is lower. Holm adjustment covers the three capacity-matched contrasts within each endpoint. The seed means equal the record-level values of Table 10.

	Δ system gain			False-rejection reduction		
Dual Routing minus	Pos.	95% CI (seeds)	Holm p (t)	Pos.	95% CI (seeds)	Holm p (t)
Shared	7/7	$[+7.92, +28.57]$	–	7/7	$[+20.28, +37.61]$	–
Role-isolated	7/7	$[+9.33, +29.20]$	0.0032	5/7	$[+1.59, +20.06]$	0.029
PCGrad-A	7/7	$[+19.64, +40.75]$	0.00085	7/7	$[+11.36, +29.84]$	0.0032
PCGrad-B	7/7	$[+30.55, +40.45]$	0.00001	7/7	$[+16.88, +27.99]$	0.00019

Table 15: Route factorial, seeds 45–48, 1,472 items each (percent, mean over seeds). Repair and preserve are the reviser's accuracy when the solver was wrong and right; keep-right is the verifier's acceptance rate on correct first answers.

Trained	Read	Final	Solver	Gain	FR	Repair	Preserve	Keep-right
Isolated	isolated	7.64	22.57	−14.93	22.45	5.8	12.6	0.4
Isolated	merged	30.04	23.08	+6.96	13.21	25.9	68.4	43.3
Dual	isolated	37.60	36.92	+0.68	30.94	18.7	71.5	16.1
Dual	merged	**38.99**	38.82	+0.17	13.65	17.5	**77.5**	64.9

H A Disjoint Validation Mix with Six More Task Families

Training logged a separate 184-item validation mix every ten steps. It shares no question with the 1,472-item panel and, besides the four program templates, contains 24-game, maze, 4×4 sudoku, and ordering puzzles, APPS, and ZebraLogic, which never appears in training. It served as the development validation split, so it is not an untouched test set. We score it at the same step-100 checkpoints, for all 35 Llama runs, with the same parser (Table 18). Absolute accuracy is low: maze, sudoku, 24-game, and ZebraLogic sit near zero for every method, so the non-template difference comes mostly from ordering puzzles and APPS.

I Revision Quality, Simple Answer Policies, and Rerun Sensitivity

Revision quality at fixed first-answer correctness. Because methods start from different solvers, Table 19 conditions on the first answer. Repair is the probability that the generated revision is correct when the solver was wrong; preservation is the probability that it stays correct when the solver was right. On Llama, Dual Routing's reviser is better on both in all seven replicates against every control, including Shared (repair $+12.34$ points, seed-level t interval $[+4.78, +19.91]$; preservation $+24.96$, $[+13.10, +36.82]$). On Qwen, where every solver is already strong, the pattern does not hold: Dual Routing preserves at least as well but repairs less, which is consistent with its more conservative verifier there (Appendix J).

Simple answer policies. Table 20 compares each method's gated output with two policies that need no verifier: keeping the solver answer and always taking the revision. Two facts matter. First, the Shared adapter's own first answers (46.47%) beat every gated workflow, including Dual Routing's (34.46%); a strong shared solver with verification switched off is a competitive system, which is why we make no final-accuracy claim against Shared. Second, Dual Routing's revision alone reaches 39.52%, above its own gate, so the interface's value lies in the reviser it trains.

Rerun sensitivity. Three added Dual seeds (45, 47, 48) were re-trained after their first runs had been scored, and the second runs are used throughout (Appendix K). Table 21 recomputes every contrast with the first runs instead. The capacity-matched conclusions survive: the isolation contrast falls from $+26.74$ to $+23.50$ points, remains significant at the seed level, and stays positive in

Table 16: Owned versus shared downstream writes with identical merged reads, seeds 45–48 (final accuracy in percent).

	Seed 45	Seed 46	Seed 47	Seed 48	Mean
Owned writes (Dual Routing)	36.28	41.58	40.76	37.36	**38.99**
Shared downstream writes	30.43	40.83	37.09	27.79	34.04
Difference	+5.84	+0.75	+3.67	+9.58	+4.96

Table 17: Read-route controls, seeds 45–48 (percent, mean over the seeds shown in column n). Repair and preserve are the reviser's accuracy when the solver was wrong and right.

Trained with	Read at evaluation	n	Final	Solver	Gain	Repair	Preserve
Isolated	own, full	4	7.64	22.57	−14.93	5.8	12.6
Isolated	own, half	4	19.87	22.50	−2.63	15.2	37.5
Isolated	{v, r} delta avg.	4	17.70	22.32	−4.62	12.9	34.3
Isolated	{v, r} factor merge	4	30.04	23.08	+6.96	25.9	68.4
Own, half	own, half	4	34.66	33.49	+1.17	18.8	73.5
Dual Routing	own, full	4	37.60	36.92	+0.68	18.7	71.5
Dual Routing	{v, r} delta avg.	4	37.53	37.36	+0.17	17.8	73.6
Dual Routing	{v, r} factor merge	4	38.99	38.82	+0.17	17.5	77.5

six of seven replicates; the two PCGrad-labeled families remain 7/7. The Shared contrast, already unresolved, moves toward zero.

J Second Backbone: Qwen2.5-7B-Instruct

The same five-arm protocol was also run on Qwen2.5-7B-Instruct at seeds 42–46: identical routes, rank-64 slots, GDPO training for 100 steps, and evaluation of the common step-100 checkpoint on a 1,280-item panel from three of the four program families. We report every arm and seed that exists, scored with the same stop-aware parser (Table 22).

The final-accuracy result does not transfer. Every Qwen arm lands between 72.75% and 74.31%; Dual Routing is highest, but its margin over each control is under two points and every seed-level interval contains zero. The workflow result does transfer. The Qwen verifier stops on 85% of episodes under Dual Routing versus 43–79% for the controls, and false rejection falls from 7.88–34.56% to 5.02%; the reduction is positive in at least four of five seeds against every control and its seed-level interval excludes zero against Shared and both PCGrad-labeled families. So on a backbone whose solver is already strong (about 71–73% first-answer accuracy), Dual Routing still produces the more conservative, less destructive verifier, but there is little headroom for that to change final accuracy. We therefore do not claim a cross-backbone final-accuracy result.

K Per-Seed Detail for the Added Replicates

Table 23 breaks out the four replicates added after the original three, on the same 1,472-item panel. Dual-45, Dual-47, and Dual-48 were each trained twice and the paper uses the second run. The first Dual-45 run had been preempted twice mid-training; all three second runs were started after the first runs had been scored on this panel, so the substitution is not outcome-blind. The first runs scored 23.71, 37.70, and 30.30 final accuracy (gain +9.99, −0.88, +3.80), and Table 21 repeats every contrast with them. No control run was replaced after evaluation; four role-isolated attempts preempted before step 100 were restarted from scratch and never scored. Across these four, Dual (45-rerun, 46, 47-rerun, 48-rerun) has mean final 39.00 / gain +0.17; Shared 45–48 mean 32.68 / −13.81. Dual leaves 10.6 pp of $\max(\text{solver}, \text{redo})$ oracle headroom without a negative mean gain; Shared leaves 18.4 pp after already losing 13.8. Paired Dual−Shared final deltas are −4.82, +5.77, +14.95, and +9.38; seed 45 is the one Dual−Shared reversal among them.

Table 18: The 184-item validation mix at step 100, seven replicates per method. Top: pooled percentages. Bottom: Dual Routing minus control, positive replicates of seven, mean, and seed-level t interval, on all 184 items and on the 152 items outside the panel's four templates.

Method	Final	Solver	Gain	False reject
Shared	10.64	14.83	-4.19	13.12
Role-isolated	4.19	8.54	-4.35	7.30
PCGrad-A	4.50	11.34	-6.83	11.02
PCGrad-B	2.33	10.64	-8.31	9.94
Dual Routing	**12.19**	10.48	$+\mathbf{1.71}$	**4.43**

Dual Routing minus		All 184 items		152 non-template items
Shared	4/7	$+1.55$ $[-1.86, +4.97]$	5/7	$+0.94$ $[-0.27, +2.15]$
Role-isolated	7/7	$+8.00$ $[+3.92, +12.08]$	6/7	$+3.67$ $[+1.42, +5.91]$
PCGrad-A	7/7	$+7.69$ $[+4.88, +10.49]$	5/7	$+1.60$ $[-0.62, +3.82]$
PCGrad-B	7/7	$+9.86$ $[+6.87, +12.85]$	7/7	$+3.67$ $[+1.88, +5.45]$

Table 19: Revision quality conditional on the first answer (percent, pooled over replicates). Repair $= P(\text{revision correct} \mid \text{solver wrong})$; preserve $= P(\text{revision correct} \mid \text{solver correct})$.

	Llama-3.1-8B (7 seeds)		Qwen2.5-7B (5 seeds)	
Method	Repair	Preserve	Repair	Preserve
Shared	8.2	51.9	15.3	96.8
Role-isolated	3.8	8.4	19.4	94.7
PCGrad-A	2.5	7.9	20.6	92.8
PCGrad-B	0.2	0.7	17.7	95.2
Dual Routing	**20.5**	**76.9**	11.8	**97.2**

L Reproduction Requirements

A faithful reproduction needs more than three adapter names. It must record the rollout/behavior read set and single training adapter for each role, the fact that update recomputation activates only that training adapter, the frozen-backbone guarantee, factor-wise linear merge with weights 0.5/0.5, per-module rank, number of active modules, parser, and stop-aware final-answer policy. Logs must retain item identifiers, solver answers, verdicts, revisions, and selected final answers so final accuracy, system gain, redo accuracy, false acceptance, false rejection, and paired contrasts can be recomputed without comparing unrelated aggregates.

The supplementary archive contains the per-item evaluation outcomes of the controlled panel (solver correct, revision correct, stop, and verdict parsed for 5 methods $\times$ 7 replicates $\times$ 1,472 items; SHA-256 prefix `338a8e230217`), the aggregate summary, the routing contract, and a standard-library script (`8edcc0ab9bb7`) that rebuilds every pooled endpoint, paired contrast, win/tie/loss count, family breakdown, per-replicate delta, and seed-level statistic from those outcomes; with `–bootstrap` it also regenerates the paired intervals exactly. A second script (`e3fd89a19c69`) rebuilds the diagnostic appendices, including the route factorial, the validation mix, and the first-run sensitivity analysis, from their shipped per-item outcomes. These are evaluation records, not model training or raw-trajectory reconstruction. End-to-end reproduction additionally requires the training configurations, checkpoint manifest, and raw generation logs. Reporting must preserve the seven training replicates and four source families instead of replacing them with one pooled mean, and must keep the record-level and seed-level uncertainty separate rather than quoting whichever is narrower.

Table 20: Final accuracy (percent, Llama, seven replicates) under the method's gate and under two verifier-free policies. Oracle takes the better of the two answers.

Method	Gated	Keep solver	Always revise	Oracle
Shared	30.21	**46.47**	28.55	50.96
Role-isolated	7.73	25.00	5.03	27.76
PCGrad-A	4.49	32.70	4.11	34.47
PCGrad-B	1.81	35.32	0.36	35.43
Dual Routing	34.46	32.47	39.52	46.98

Table 21: Dual Routing minus control, final accuracy (points), with the reruns used in the paper and with the original first runs. Positive replicates out of seven, mean, and seed-level t interval.

	Reruns (paper)		Original first runs	
Dual Routing minus	Pos.	Mean [95% CI]	Pos.	Mean [95% CI]
Shared	5/7	$+4.25$ $[-5.82, +14.32]$	5/7	$+1.01$ $[-10.60, +12.62]$
Role-isolated	7/7	$+26.74$ $[+16.42, +37.06]$	6/7	$+23.50$ $[+11.22, +35.77]$
PCGrad-A	7/7	$+29.97$ $[+19.68, +40.26]$	7/7	$+26.73$ $[+16.85, +36.60]$
PCGrad-B	7/7	$+32.66$ $[+26.37, +38.94]$	7/7	$+29.42$ $[+22.98, +35.85]$

Table 22: Qwen2.5-7B-Instruct, five training seeds, 1,280 items per checkpoint. Top: pooled percentages. Bottom: Dual Routing minus control per seed, positive in how many of five seeds, with the mean and a seed-level t interval; false-rejection reduction is positive when Dual Routing rejects fewer correct answers.

Method	Final	Solver	Gain	False reject	Stop
Shared	73.50	71.17	$+2.33$	7.88	79.23
Role-isolated	73.41	70.83	$+2.58$	19.56	61.53
PCGrad-A	72.75	71.66	$+1.09$	34.56	42.86
PCGrad-B	73.53	71.47	$+2.06$	15.36	67.81
Dual Routing	**74.31**	73.17	$+1.14$	**5.02**	85.19

Dual Routing minus		Δ final		False-rejection reduction
Shared	3/5	$+0.81$ $[-2.55, +4.18]$	4/5	$+2.86$ $[+0.09, +5.62]$
Role-isolated	3/5	$+0.91$ $[-1.32, +3.14]$	4/5	$+14.55$ $[-8.71, +37.80]$
PCGrad-A	4/5	$+1.56$ $[-0.88, +4.00]$	5/5	$+29.55$ $[+6.42, +52.67]$
PCGrad-B	3/5	$+0.78$ $[-2.58, +4.14]$	5/5	$+10.34$ $[+0.29, +20.40]$

Table 23: The four added Dual and Shared replicates on the 1,472-item panel. Oracle is $\max(\text{solver}, \text{redo})$; headroom is oracle minus final. Percentages.

Method (seed)	Solver	Final	Gain	Oracle	Headroom
Dual 45 rerun	36.21	36.28	$+0.07$	48.91	12.64
Dual 46	41.51	41.58	$+0.07$	50.88	9.31
Dual 47 rerun	40.90	40.76	-0.14	49.52	8.76
Dual 48 rerun	36.68	37.36	$+0.68$	48.98	11.62
Shared 45	51.63	41.10	-10.53	55.30	14.20
Shared 46	46.67	35.80	-10.87	52.79	16.98
Shared 47	41.24	25.82	-15.42	47.69	21.88
Shared 48	46.40	27.99	-18.41	48.57	20.58