

Negative Results in Cross-Fine-Tune LLM Merge: A Failure-Mode Catalog of SLERP, TIES, DARE, Observation-Driven Rescue, and Passthrough on Qwen2.5-Instruct $\times$ Qwen2.5-Coder

Chen, Ho Yiing¹

¹Independent researcher, charenix.com, ORCID: 0009-0006-6816-9891
abbychen981008@gmail.com

2026-05-24

Abstract

I run eight cross-fine-tune blend recipes on the Qwen2.5-7B-Instruct $\times$ Qwen2.5-Coder-7B-Instruct pair, evaluate on a five-prompt panel covering multilingual reasoning, structured argument, and code refactor, and report four reproducible failure modes that the public model-merging literature has not catalogued. (1) Conservative SLERP at $t \approx 0.20$ produces complete collapse on all five inputs (multiple-choice loops, list collapse), worse than aggressive SLERP at $t = 0.9$ which only drifts on some inputs. (2) DARE-TIES at standard density 0.55 produces uninterpretable token noise on every input. (3) Observation-driven TIES, which sets per-block density from Mercury observation data [1], regresses on a code-bug input that the uniform-density TIES baseline handles correctly. (4) Scalar-only post-merge anchor rescue, which patches layer-norm gain at observation-identified anchor coordinates but leaves downstream attention and MLP read paths unchanged, breaks generation on two of five inputs; full-subspace rescue that also patches the read paths preserves coherence. Among the eight recipes, simple passthrough Frankenmerge with conservative coder injection (24 Instruct blocks plus 4 Coder blocks at the tail, "v7b") is the empirical winner, outperforming all averaging-based approaches on coherence preservation. The reading I propose is that a coding fine-tune adjusts layer-norm and adjacent paths as an integrated subspace adaptation, and averaging methods that touch only one component of that subspace destroy the integration. Open data, configurations, and inference logs are at the Mercury paper handoff repository [2].

1 Introduction

The open-weight model-merging community has converged on three standard recipes: spherical linear interpolation [6], popularized in the LLM context by the MergeKit toolkit [3]; TIES-Merging [4], which resolves sign conflicts via magnitude voting; and DARE [5], which adds random parameter dropping before voting. A fourth family, passthrough Frankenmerges that splice blocks from one source into another without averaging weights, has substantial empirical traction in the practitioner community but very limited formal evaluation.

A systematic comparison of all four families on a single cross-fine-tune pair, with each recipe's failure mode catalogued rather than averaged into a single benchmark score, is missing from the public record. This paper fills that gap on the Qwen2.5-7B-Instruct $\times$ Qwen2.5-Coder-7B-Instruct pair. Both

endpoints share the same architecture and the same tokenizer, differing only in fine-tune objective, which isolates the cross-fine-tune question from cross-architecture confounds.

The headline result is negative across the averaging family. SLERP collapses, TIES regresses on observation-driven configurations, DARE produces token noise. Only passthrough preserves coherent generation across the test panel. The reading I propose, and partially support with a separate post-merge rescue experiment, is that fine-tunes adapt layer-norm and adjacent attention/MLP slices as an integrated subspace. Averaging-based blends break the integration in different ways depending on how aggressively they interpolate.

2 Method

2.1 Setup

Both endpoints are loaded at fp16 from HuggingFace. Merging runs use mergekit 0.1.4 with a pydantic 2.10 compatibility patch. Inference runs on a Mac mini M4 Pro at 24 GB unified memory in bf16 on CPU because Metal hit OOM on full 14 GB weight loads. Five inputs comprise the evaluation panel: a Chinese-language race-condition explainer, a structured-argument prompt, a debugging task, a code-refactor task, and a multilingual translation task. Each input is evaluated by manual classification into one of strong, partial, drift, collapse, gibberish.

2.2 Eight blend recipes

Label	Family	Configuration
v1	SLERP	layer-wise t profile [0.1, 0.2, 0.4, 0.6, 0.8] for attention, [0.1, 0.2, 0.5, 0.7, 0.9] for MLP
v2	SLERP	uniform $t = 0.20$ attention, $t = 0.25$ MLP
v3	TIES	uniform density 0.55, weight 65/35 Instruct/Coder
v4	TIES	observation-driven per-block density profile from Mercury
v5	DARE-TIES	density 0.55, weight 65/35
v6	Passthrough	20 Instruct blocks + 8 Coder blocks
v7a/b/c	Passthrough	18+10, 24+4, sandwich Instruct-Coder-Instruct
v8/v9	Anchor rescue	LN-only (v8) and full subspace (v9) post-merge patch

Table 1: Eight blend recipes evaluated on the same five-input panel.

2.3 Evaluation

I classify each model’s per-input output qualitatively. The categories are: strong (matches or exceeds either endpoint on coherence and task completion); partial (some degradation but still readable and on-topic); drift (language switching or repetition that degrades the answer); collapse (the output disintegrates into a non-answer such as a multiple-choice template); gibberish (tokens outside the expected language). I avoid HumanEval or MMLU here because the failures span coherence and language drift rather than answer correctness, and benchmark scores conceal that distinction.

3 Results

3.1 The SLERP cliff (v1, v2)

Two recipes. Two outcomes. The high- t output drifts; the low- t output collapses.

Aggressive SLERP (v1, layer-wise t up to 0.9 at the tail) drifts on all five inputs. Most visibly, on the Chinese race-condition input, v1 switches from traditional to simplified script mid-output and falls into a repetition loop on the polite closing phrase “謝謝你的耐心傾聽,” repeated five times. Capability loss is partial: the output is recognizable as an attempted answer.

Conservative SLERP (v2, uniform $t \approx 0.20$) does worse, not better. On every input, v2 collapses into a multiple-choice template (“A/B/C/D/E/F/G/...”) or a numbered list with no content per item. The base Instruct model on the same inputs produces 2 strong and 2 partial answers.

This is counter-intuitive given the community heuristic that low SLERP t preserves the dominant model’s behavior. A plausible reading: at low t , the merge sits in a narrow band where neither fine-tune’s integrated subspace is intact, producing weights that locally satisfy both gradients but globally satisfy neither. At high t , one fine-tune wins and the other becomes a perturbation. The cliff between collapse and partial-coherence is sharper than blending intuition suggests.

3.2 DARE-TIES destroys the tokenizer surface (v5)

Unreadable.

DARE-TIES at the literature-default density 0.55 produces token-noise output on every input. Example fragment: “0 武 □ bénéfici-masteruantkea Pendant_equalTo...”. The random parameter drop at high rate, combined with magnitude voting that retains weights from whichever endpoint was retained at each position, evidently breaks something in the embedding-to-logit map that affects basic tokenization fidelity rather than just task quality. I did not attempt lower DARE drop rates, which may recover usable output; that ablation is on the immediate follow-up list.

3.3 Observation-driven TIES regresses (v4)

The v4 configuration uses Mercury observation [1] to set per-block TIES density. Specifically: lower blocks (0–9) receive high Instruct weight (0.78); middle blocks (10–19) receive 50/50 weights with high density (0.65); upper blocks (20–27) receive high Coder weight (0.62). The intended logic is to give each fine-tune more influence at the depths where it was observed to fire most.

The result is that v4 regresses on a code-bug input where v3 (uniform density) succeeds. On a debugging task asking about Python dictionary semantics, v4 produces the assertion that “`dict.update()` is a set operation,” which is false. v3 on the same input correctly identifies it as a dictionary mutation method. The depth-banded weight profile, derived from observation data that was not designed for merge-parameter prescription, introduced a new error mode.

The lesson, I think, is that an observation dataset that reports where a model fires hardest does not directly translate to “which blocks should receive the larger merge weight.” The mapping from observation to merge parameter is itself a hypothesis that requires its own validation.

3.4 Passthrough wins (v6, v7a/b/c)

No averaging.

Across the passthrough variants, v7b (24 Instruct blocks plus 4 Coder blocks appended at the tail) returns the best per-input outcome: 3 strong, 1 best (exceeds base on the race-condition input by producing a clean three-section answer with explicit rebuttal-and-counter-rebuttal structure), and 1 partial. v6 (20+8) and v7a (18+10) both produce mixed results with one or two collapses each. The sandwich variant v7c (Instruct–Coder–Instruct) produces 3 strong and 2 collapse, suggesting that re-injecting Instruct blocks above Coder blocks is incompatible with the integration the Coder fine-tune assumes about its position in the stack.

Passthrough preserves the original layer-norm, attention, and MLP weights inside each donated block without averaging. The recipient stack absorbs the donor blocks as-is. The empirical observation is that this preservation matters more than any averaging-based blend recipe at the cross-fine-tune scale.

3.5 Subspace constraint (v8, v9)

A separate experiment patches the v1 merge result post-hoc by overwriting layer-norm gain at a fixed set of observation-identified anchor coordinates with the corresponding values from the Coder endpoint. v8 patches only layer-norm gain (99 scalar values). v9 patches layer-norm gain plus the corresponding rows of the input projection, output projection, and MLP gate for those anchor coordinates (550 slices).

v8 produces generation collapse on two of five inputs. v9 preserves coherence across all five. Patching only the layer-norm scalar without patching the attention and MLP read paths that read from those coordinates leaves the network in a state where the layer-norm gain advertises a re-scaled coordinate but downstream paths are still scaled for the original gain.

Variant	LN patched	Attention + MLP patched	Result
v8 LN-only	yes (99 scalars)	no	collapse on 2/5 inputs
v9 full subspace	yes	yes (550 slices)	coherence preserved

Table 2: Scalar-only rescue breaks downstream integration. The full-subspace patch preserves it.

The interpretation: fine-tuning a coding model adjusts layer-norm gain at certain coordinates as part of an integrated change that also adjusts which downstream attention and MLP slices read from those coordinates. Patching one component of that integrated change in isolation leaves the rest in a stale state.

4 Discussion

4.1 Why averaging blends fail more than passthrough

The four averaging-based families I tried (two SLERP variants, two TIES variants, one DARE-TIES) share the property that they take coordinate-wise convex combinations of two weight tensors. If both endpoints adapted that coordinate as part of an integrated subspace change, the convex combination of the two adaptations is not itself an integrated adaptation. The combination satisfies each endpoint’s gradient locally but no longer participates in a globally consistent representation. Passthrough avoids the problem by never averaging: each donated block carries its integrated subspace intact.

This reading is consistent with the v8 vs v9 result. v8 averages (or in this case replaces) the layer-norm component of an integrated subspace, leaving downstream paths unchanged; the result collapses. v9 patches all components of the subspace together; the result is coherent.

4.2 Why low- t SLERP collapses harder than high- t SLERP

At low SLERP t (say 0.2), every coordinate carries a mixture that is closer to the donor’s adaptation than to the recipient’s. The recipient’s integration is degraded everywhere, and the donor’s integration is incomplete. At high t (say 0.9), most coordinates are close to the donor and the recipient becomes a small perturbation that the donor’s integration can tolerate. The intuition that low t “preserves the recipient” assumes that small per-coordinate changes preserve coherence; the subspace-integration framing predicts the opposite at the cross-fine-tune scale, and the v1-vs-v2 result is consistent with that prediction.

4.3 Implications for the merge community

Three practical readings follow.

1. For cross-fine-tune merges of strong-fine-tune endpoints (Instruct and Coder both have substantial post-training), passthrough should be the first attempt, not the last.
2. Observation-driven merge parameter setting needs explicit validation. Mercury-style firing observations do not automatically map to TIES density or SLERP t choice.

3. Post-merge rescue methods (anchor patching, weight scaling) must patch full subspaces, not isolated components. A layer-norm gain change without the corresponding read-path change is not a partial rescue, it is a fresh failure mode.

5 Related work

MergeKit [3] is the toolkit through which the community uses these recipes; this paper depends on it for v1–v7. TIES-Merging [4] introduced magnitude-voting interference resolution. DARE [5] added stochastic parameter dropping. Spherical linear interpolation traces to Shoemake [6]; its use for weight averaging in deep nets builds on the Model Soups line [7]. The subspace-integration reading I propose is not original to this paper; the broader claim that fine-tuning adapts neighborhoods of weights as integrated structures has been part of the sparse-feature interpretability programme [8] since at least 2023. What this paper adds is a small, reproducible, cross-fine-tune negative-result set on a single Qwen pair with all configurations open at the Mercury handoff repository [2].

6 Limitations

Single endpoint pair. Only Qwen2.5 Instruct $\times$ Coder. Llama-Instruct $\times$ Llama-Code and Mistral-Instruct $\times$ Mistral-Code reproductions are the immediate follow-up, and the four failure modes may or may not survive the cross-pair check.

Five inputs. The panel is small and qualitatively scored. Scaling to twenty or one hundred inputs with per-input automated scoring (e.g. language-detection + perplexity + task-specific metrics) would let the failure-mode counts carry confidence intervals.

No standard benchmarks. I report no HumanEval or MMLU numbers. The reason is that the dominant failure mode in the averaging family is coherence collapse, which benchmark scores hide; but a benchmark-equipped follow-up would strengthen the v7b winner claim.

Architecture-specific. Only qwen2 architecture. Different architectures may show different passthrough winners or different averaging-family failure profiles.

CPU bf16 inference noise. The Mac mini bf16 CPU inference produced occasional blank-output runs on v6 and v7a that are not deterministic. I retained the deterministic outputs only. A GPU re-run on a larger machine would eliminate this noise channel.

7 Conclusion

Four failures. One winner.

Across eight cross-fine-tune blend recipes on the Qwen2.5-Instruct $\times$ Qwen2.5-Coder pair, the four averaging-based families (SLERP, TIES, observation-driven TIES, DARE-TIES) each fail in a distinct, reproducible way on a five-input panel. Conservative SLERP collapses harder than aggressive SLERP; DARE-TIES at standard density produces token noise; observation-driven TIES introduces a new regression on a code-bug input; scalar-only post-merge rescue breaks integration. The passthrough variant v7b is the empirical winner. The proposed reading is that fine-tunes adapt layer-norm and adjacent paths as an integrated subspace, and averaging methods that touch only one component destroy the integration. All configurations and inference logs are open [2].

Acknowledgements

Drafted in a paired-engineering session with an Anthropic Claude Code agent. All numeric claims trace back to JSON files in the Mercury paper handoff repository.

References

- [1] Chen, Ho Yiing. Mercury: An addressable million-cell sensor grid for browser-accessible LLM observability on consumer hardware. Zenodo, 2026. DOI: 10.5281/zenodo.20313154.
- [2] Chen, Ho Yiing. Mercury paper handoff: drafts, analysis scripts, and Tier-A / Tier-B / Tier-C observation data. <https://github.com/norika1207-lab/mercury-paper-handoff>, 2026.
- [3] Goddard, C., Siriwardhana, S., Ehghaghi, M., Meyers, L., Karpukhin, V., Benedict, B., McQuade, M., and Solawetz, J. Arcee’s MergeKit: A Toolkit for Merging Large Language Models. arXiv preprint, March 2024. DOI: 10.48550/arXiv.2403.13257.
- [4] Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems*, 2023. DOI: 10.48550/arXiv.2306.01708.
- [5] Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. arXiv preprint, November 2023. DOI: 10.48550/arXiv.2311.03099.
- [6] Shoemake, K. Animating Rotation with Quaternion Curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’85)*, pages 245–254, 1985. DOI: 10.1145/325334.325242.
- [7] Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy Without Increasing Inference Time. arXiv preprint, March 2022. DOI: 10.48550/arXiv.2203.05482.
- [8] Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askeel, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Tamkin, A., Nguyen, K., McLean, B., Burke, J., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. Transformer Circuits Thread, Anthropic, October 2023. <https://transformer-circuits.pub/2023/monosemantic-features>.