NEW LORA SKILLS SHOULD READ BUT NEVER WRITE

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adapters (LoRA) make it cheap to fine-tune a large language model once per task, but combining several independently trained adapters into one model remains difficult: merging the updates in weight space causes interference, retraining on all task data is expensive, and routing between separate adapters gives up the goal of a single combined model. We trace the difficulty to two choices that every composition method makes implicitly. A LoRA update admits infinitely many equivalent factorizations; the choice among them is invisible while an adapter serves alone, but it determines what a learned interaction between adapters can see. A coupling between an old skill and a new one can likewise point in either direction, and the direction decides whether the old skills keep computing what they computed before. We introduce READ (Read-only Expansion of Adapter Deltas), which fixes both choices: each adapter is rewritten into a balanced canonical form that preserves its update exactly, and the coupling grows in one direction only, so a new skill can read the input subspaces of old skills but cannot write into their output subspaces. The only trainable object at each append is the new skill's row of the coupling matrix, and the composed update folds into the base weights with no inference cost, routing, or task-specific rules. We evaluate READ across four benchmark suites and two model families, adding skills one at a time. Across several families, READ improves every suite average over the strongest published baselines built from the same adapters—by more than twenty points on SuperGLUE and more than seven points on the domain suite—and nearly all complete addition sequences end above every direct baseline. Factor coordinates and coupling direction, which a lone adapter never exposes, are what decide whether composed skills survive.

1 INTRODUCTION

Fine-tuning large language models with LoRA adapters is standard practice, and each task yields a small, reusable module (Hu et al., 2022; Houlsby et al., 2019). A practical system therefore accumulates many independently trained adapters, but combining them into one model remains difficult. Adding updates in weight space causes interference (Ilharco et al., 2023; Wortsman et al., 2022; Matena & Raffel, 2022). Joint retraining on all task data is expensive and can damage the skills already in the model. Routing keeps every adapter separate (Pfeiffer et al., 2021), but then the system never becomes a single combined model at all.

Existing methods either improve merge arithmetic or avoid merging. TIES, DARE, LoRA–LEGO, and KnOTS repair, decompose, or align updates before averaging (Yadav et al., 2023; Yu et al., 2024; Zhao et al., 2024; Stoica et al., 2025; Entezari et al., 2022; Ainsworth et al., 2023), but a new adapter still requires rebuilding the merge and moving old skills again. LoRAHub, PHATGOOSE, LoRA-Mixer, and GraftLLM keep modules separate through coefficients, experts, routers, or grafts (Huang et al., 2024a; Muqeeth et al., 2024; Li et al., 2026; Du et al., 2026). The old skills survive, but the serving system keeps an extra object and never learns a folded interaction among skills.

These approaches treat an adapter as a finished object whose internal form does not matter. Composition is where this form starts to matter, because a LoRA update $\Delta W = BA$ can also be written as $(BR)(R^{-1}A)$ for any invertible R. This gauge freedom is invisible while an adapter serves alone, but a learned coupling between adapters can inherit arbitrary factor choices rather than skill relations. Interaction also adds a directional choice, since a link between an old skill and a new one can point either way, and existing practice does not decide the direction. Composition therefore forces a question that no existing method makes explicit. When a new skill must share a model with old ones,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

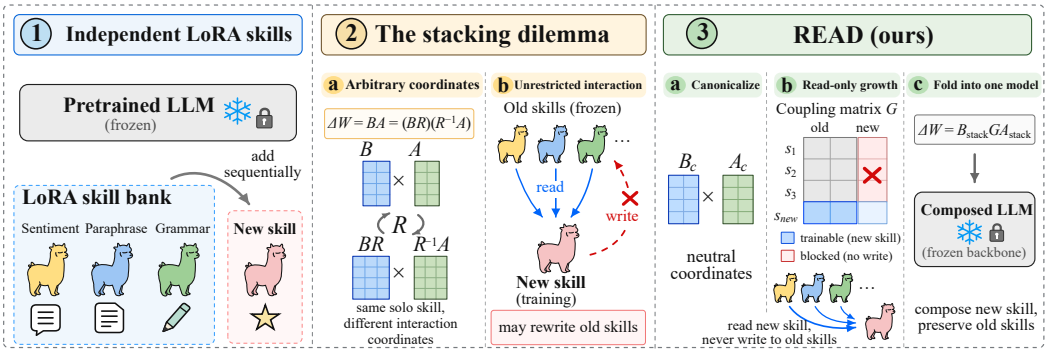


Figure 1: Overview. Naive composition of independently trained LoRA skills leaves two freedoms unmanaged, arbitrary factor coordinates and unrestricted write directions. READ canonicalizes the factors, trains only the new skill's read row in G, blocks writes into old skills, and folds $B_{\text{stack}}GA_{\text{stack}}$ into one update.

one side has to adapt, and the choice of which side determines whether the old skills survive. Figure 1 (middle) illustrates both.

READ (Read-only Expansion of Adapter Deltas) fixes both choices (Figure 1, right; §3). The trained adapters form a frozen bank, joined by a small coupling matrix G between their stacked factors, $\Delta W = B_{\text{stack}}GA_{\text{stack}}$. READ rewrites every adapter into a balanced canonical form that preserves its update exactly, and at each append trains only the new skill's read row of G into the frozen bank. Everything previously learned is preserved by construction because all factors and established couplings stay frozen. The composed update folds into a single rank-kr weight change, so serving costs nothing beyond the base model and needs neither a router nor a per-task head.

We evaluate READ on the GLUE, SuperGLUE, Domain, and BBH benchmarks with Llama-3.2-3B and Qwen3-4B, adding one skill at a time. On Llama-3.2-3B, READ wins the GLUE, SuperGLUE, and Domain terminal suite averages against the strongest foldable alternative, including 0.783 versus 0.605 on SuperGLUE and 0.887 versus 0.846 on Domain. On Qwen3-4B it likewise wins those three suites (GLUE 0.838 versus 0.775 for the strongest alternative). Across the shared 32-lineage terminal-bank comparison, READ wins 24 lineages and loses eight, all on BBH. This paper makes three contributions.

- We identify two choices that every LoRA composition method makes implicitly, factor coordinates and coupling direction, and show that both matter once skills interact.
- We propose READ, a composition operator that canonicalizes adapter factors, trains only a new skill's read direction into a frozen bank, and folds the result into a single weight change that serves at base latency with no router or per-task head.
- We evaluate READ across two model families and four benchmark suites. Across all 32 lineages the composed model beats the strongest of fourteen foldable alternatives per lineage by $+0.073$ on average (95% CI $+0.047$ to $+0.101$), and all eight losses are confined to BBH.

2 Related Work

2.1 Merging in Weight Space

Weight-space merging rests on an empirical fact. Independently fine-tuned solutions can often be averaged without destroying their task behavior (Wortsman et al., 2022; Matena & Raffel, 2022), and fine-tuned networks sit in connected low-loss regions (Garipov et al., 2018; Frankle et al., 2020). Task arithmetic makes this view explicit by treating each fine-tuning run as an update vector that can be added or subtracted (Ilharco et al., 2023). Later work makes this vector arithmetic less fragile, and each method repairs a different failure. TIES trims small entries and resolves sign conflicts between updates (Yadav et al., 2023). DARE drops most entries and rescales the survivors (Yu et al., 2024). RegMean replaces averaging with a per-neuron least-squares problem (Jin et al.,

2023), and EMR-Merging keeps a compact shared memory so that further merges stay cheap (Huang et al., 2024b). Model Breadcrumbs removes the smallest and largest entries of each update as noise (Davari & Belilovsky, 2024), and Model Stock pulls the merge toward the center of the fine-tuned weights (Jang et al., 2024). Yadav et al. (2024) test whether these rules still hold as models grow. A separate geometric line aligns hidden units before interpolation. Permutation symmetries explain why separately trained networks can be connected at all (Entezari et al., 2022), Git Re-Basin searches for a suitable alignment explicitly (Ainsworth et al., 2023), and ZipIt extends the idea to models trained on different tasks (Stoica et al., 2024).

These methods produce a merged model from a completed collection of updates, and none of them learns an interaction while skills are being added. A newly arriving adapter triggers another global merge, which revises the representation of every earlier skill. READ instead asks for an append operator whose new degrees of freedom adapt to the existing bank while the bank itself stays fixed.

2.2 Combining modules at run time

Parameter-efficient fine-tuning makes keeping separate skills practical. The trainable object can be an inserted adapter layer (Houlsby et al., 2019), a learned continuous prompt (Li & Liang, 2021; Lester et al., 2021), only the bias terms (Ben Zaken et al., 2022), or a pair of low-rank factors (Hu et al., 2022). Later variants rescale activations (Liu et al., 2022) or compress the adapter further (Mahabadi et al., 2021), and He et al. (2022) place most of these designs in one framework. Composition in this setting keeps the modules intact and chooses among them at inference. AdapterFusion learns attention over frozen adapters (Pfeiffer et al., 2021). AdapterDrop shows that many adapters can be skipped to save computation (Rücklé et al., 2021). LoRAHub estimates a fixed coefficient vector over adapters from a few examples (Huang et al., 2024a). LoRA-Mixer trains a token-level router over LoRA experts, and the router keeps running at every layer during serving (Li et al., 2026). GraftLLM attaches skill packs grafted from heterogeneous source models to a lightweight host (Du et al., 2026). Continual learning and sparse experts make the same trade-off from the other side. EWC and VCL protect old tasks by constraining parameter drift (Kirkpatrick et al., 2017; Nguyen et al., 2018), progressive prompts and O-LoRA allocate a new prompt or module to each task (Razdaibiedina et al., 2023; Wang et al., 2023), and mixture-of-experts models route each input to a parameter subset (Shazeer et al., 2017; Fedus et al., 2022).

All of these leave an extra mechanism on the serving path, whether a coefficient, a switch, or an attached module. Weight-space merging removes that mechanism but treats the adapters as finished updates. READ works inside the LoRA factorization instead. It rewrites each adapter into canonical coordinates without changing its update, trains only the new skill's read direction into a frozen bank, and folds the result into the base weights, so serving uses one ordinary model with no router and no per-task head. Routing remains the appropriate choice where each solo adapter must stay intact per task, and §5 returns to this boundary.

3 The READ Operator

READ adds one LoRA skill to an existing bank without changing what the bank already does. A LoRA adapter stores a skill as two small matrices whose product is the weight update. When several skills share one model, two choices that a lone adapter never exposes start to matter. The first is the factor coordinates of each update, and the second is the direction of each cross-skill link. READ fixes the coordinates once, lets the new skill read the frozen bank but never write into it, and folds the result into one model.

3.1 The frozen skill bank

Consider one projection module and omit its index. Each source skill is an independently trained generative-classification LoRA (r=8) on the q- and v-projections of every attention block, following a fixed prompt/completion recipe (§4.1). All tasks share the causal-LM head; labels are emitted as text. For k skills, stack the frozen factors.

$$\Delta W \;=\; B_{\text{stack}}\, G\, A_{\text{stack}}, \qquad A_{\text{stack}} \in \mathbb{R}^{kr\times d_{\text{in}}},\; B_{\text{stack}} \in \mathbb{R}^{d_{\text{out}}\times kr},\; G \in \mathbb{R}^{kr\times kr}, \tag{1}$$

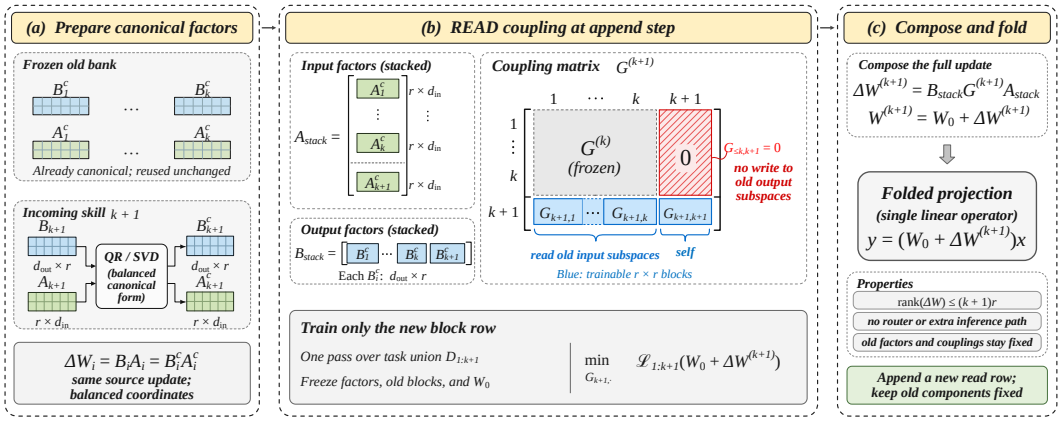


Figure 2: The READ operator at one append. (a) Every source adapter is rewritten into balanced canonical factors with $B_i^c A_i^c = B_i A_i$, so skills meet in coordinates determined by their updates. (b) The stacked factors are joined by a coupling matrix. The old block $G^{(k)}$ stays frozen, the write column $G_{\le k,k+1}$ is fixed to zero, and only the new skill's row is trained. (c) The product folds into the base weights W_0, giving one merged projection, no router, and no extra inference path.

G is block-partitioned into $r \times r$ blocks G_{ij}, $i, j \le k$. G_{ii} scales and mixes skill i's own directions; G_{ij}, $i \neq j$, maps skill j's input directions into skill i's output directions. With G equal to the identity the update is the naive sum of the adapters; every departure from that sum is learned inside G. Inference applies one effective rank-kr update per module, the same functional form as a single LoRA of rank kr, and since factors and G are frozen between appends, ΔW can be precomputed and folded into the frozen base projection W_0, leaving zero residual serving cost relative to the base model. The update requires no task identity, adapter routing, or per-task weight/head path at inference; the task instruction in the ordinary input prompt specifies the output label space.

3.2 Canonicalizing every adapter

For G to relate directions across skills, those directions must first be comparable. The factorization of a source update is gauge-ambiguous, since for any invertible R, $(B_i R)(R^{-1} A_i) = B_i A_i$. While an adapter serves alone, this freedom is invisible because the product is all the model sees. Composition exposes it. G maps one skill's input basis into another's output basis, but neither skill's function determines those bases. They are accidents of optimization, set by initialization, data order, and seed, so a coupling learned between raw factorizations can encode those accidents instead of the skills. The same two adapters, factored differently by two training runs, would force different coupling values for the same functional relationship. We therefore fix the gauge before any coupling is trained, so the coordinates in which skills meet are determined by the update itself. We compute thin QR decompositions $B_i = Q_B R_B$ and $A_i^\top = Q_A R_A$, followed by $R_B R_A^\top = U \Sigma V^\top$, and set

$$B_i^c = Q_B U \Sigma^{1/2}, \qquad A_i^c = \Sigma^{1/2} V^\top Q_A^\top. \tag{2}$$

Then $B_i^c A_i^c = B_i A_i$ exactly, while $(B_i^c)^\top B_i^c = A_i^c (A_i^c)^\top = \Sigma$. Both factors carry the same metric, so no skill's basis is privileged in the interaction, and the remaining freedom is a rotation inside degenerate singular subspaces, which cannot change any product. The source skill is unchanged; only the coordinates in which it meets the others are.

3.3 Read-only coupling

With coordinates fixed, the remaining choice is the direction of a link between an old skill and a new one. When skill $k{+}1$ arrives, its cross blocks could point either way. $G_{i,k+1}$ with $i \le k$ writes the new skill's input directions into old output bases, while $G_{k+1,i}$ lets the new skill read old input directions into its own output basis. The two directions differ in what they can damage. A nonzero $G_{i,k+1}$ adds a term to what every old skill computes on every input; old behavior is then protected only insofar as optimization happens to move it little. A nonzero $G_{k+1,i}$ adds nothing to any old output, so the old skills' function is preserved by construction. The asymmetry extends

Algorithm 1 One READ append. Skill $k{+}1$ joins a bank of k skills.

Require: frozen bank $\{B_i^c, A_i^c\}_{i\le k}$ and coupling $G^{(k)}$; incoming solo factors B_{k+1}, A_{k+1}; base weights W_0; task-union data $\mathcal{D}_{1:k+1}$
Ensure: folded weights $W_0' = W_0 + B_{\text{stack}} G' A_{\text{stack}}$, a bank of $k{+}1$ skills
1: **Canonicalize** (§3.2). Thin QR $B_{k+1}{=}Q_B R_B$, $A_{k+1}^\top{=}Q_A R_A$, then $R_B R_A^\top{=}U\Sigma V^\top$;
set $B_{k+1}^c{=}Q_B U\Sigma^{1/2}$ and $A_{k+1}^c{=}\Sigma^{1/2}V^\top Q_A^\top$ ▷ $B^cA^c{=}BA$ exactly
2: **Embed**. Install $G^{(k)}$ frozen in the upper-left block; set the write column $G'_{\le k,k+1}{=}0$, the new row $G'_{k+1,\le k}{=}0$, and the new diagonal $G'_{k+1,k+1}{=}0.3\,I_r$
3: **Train** (§3.3). One pass over $\mathcal{D}_{1:k+1}$ optimizing only $G'_{k+1,k+1}$ and $G'_{k+1,\le k}$ ▷ factors, old blocks, and W_0 stay frozen
4: **Fold**. $W_0' \leftarrow W_0 + B_{\text{stack}} G' A_{\text{stack}}$ ▷ one merged projection, zero serving cost

to gradients. During the append pass over the union of old and new task data (§4.1), a write block receives gradients from old-task examples, while a read block is optimized only through the new skill's output. READ therefore fixes $G_{\le k,k+1} = 0$ and trains only the new skill's row and diagonal. At the append, the trainable addition is

$$B_{k+1}^c \left(G_{k+1,k+1} A_{k+1}^c + \sum_{j\le k} G_{k+1,j} A_j^c \right). \tag{3}$$

The new skill may reuse old input subspaces, but it cannot write through its input directions into old output bases. All earlier blocks, every canonical factor, the backbone, and the task head are frozen.

3.4 Appending and folding a skill

The incoming adapter now becomes the new bank row in five steps (Figure 2). **Extract.** Take its trained solo factors A_{k+1}, B_{k+1}. **Canonicalize.** Rewrite all factors into the balanced form of §3.2; no source update changes. **Embed.** Grow G from $(kr)^2$ to $((k{+}1)r)^2$ per module, install the old $G^{(k)}$ verbatim in its upper-left block, and freeze it. **Train.** Make one pass over the stage's task union, optimizing only the new row and diagonal (§4.1). **Fold.** Precompute $B_{\text{stack}} G A_{\text{stack}}$, add it to W_0, and serve with one merged projection per module.

3.5 One READ append and its cost

Algorithm 1 makes the five steps explicit. Canonicalize and fold run per projection module; train runs jointly over all modules and is the only step that touches data.

The costs follow the same split. Canonicalization costs two thin QR decompositions of $d{\times}r$ matrices and one $r{\times}r$ SVD per module, $O(dr^2 + r^3)$ in the projection dimension d, under a million flops at Llama sizes, negligible next to any training step. Embedding places the frozen block and does no arithmetic. Training optimizes $(k{+}1)r^2$ floats per module, the new diagonal and its k row blocks, and nothing else. That is 384 floats per module, 21,504 in total, at a six-skill append. Per token, the effective update costs the same as one LoRA layer of rank kr (§3.1). The fold multiplies the three factors once per module, $O(d^2kr)$ multiply-adds in closed form; after it, serving runs the plain base model. Storage beyond the $k{+}1$ adapters is the coupling itself, $(kr)^2$ floats per module, which is 129k across the 56 Llama q/v projections at $k{=}6$, about 4.7% of one adapter. The per-append trainable set grows linearly in k and the coupling storage quadratically; both remain small fractions of one adapter over the measured range.

4 Experiments

4.1 Setup

We evaluate Llama-3.2-3B-Instruct (Grattafiori et al., 2024) and Qwen3-4B (Yang et al., 2025) on GLUE-6 (Wang et al., 2019), SuperGLUE-4, BBH-6, and Domain-3. Each task first trains an

Table 1: Terminal-bank macro comparison across READ and fourteen foldable alternatives. Cells average the four complete terminal lineages per suite–model; the first baseline row is the per-lineage strongest alternative used for the primary comparison.

	Llama-3.2-3B				Qwen3-4B					
Method	GLUE-6	SuperGLUE-4	Domain-3	BBH-6	GLUE-6	SuperGLUE-4	Domain-3	BBH-6	Avg	ΔR
READ (ours)	**0.823**	**0.783**	**0.887**	0.445	**0.838**	**0.797**	**0.852**	0.770	**0.774**	–
Strongest alt. (per lineage)	0.631	0.605	0.846	0.459	0.775	0.709	0.794	0.793	0.702	+0.073
Soup	0.016	0.547	0.695	0.392	0.339	0.550	0.666	0.772	0.497	+0.277
Fisher	0.098	0.425	0.640	0.406	0.363	0.000	0.365	0.764	0.383	+0.392
Task Arithmetic	−0.083	0.550	0.682	0.421	0.087	0.574	0.653	0.776	0.458	+0.317
RegMean	0.631	0.605	0.844	0.439	0.775	0.619	0.794	0.768	0.684	+0.090
TIES	0.047	0.455	0.617	0.425	0.154	0.432	0.403	0.779	0.414	+0.361
DARE	−0.085	0.551	0.678	0.429	0.083	0.573	0.657	0.781	0.458	+0.316
Breadcrumbs	−0.063	0.546	0.663	0.411	0.226	0.529	0.588	0.789	0.461	+0.313
LoRA–LEGO	−0.097	0.543	0.785	0.392	0.074	0.488	0.723	**0.790**	0.462	+0.312
KnOTS–TIES	0.238	0.537	0.704	0.394	0.342	0.477	0.625	0.748	0.508	+0.266
LoRAHub–5shot	0.209	0.567	0.811	0.438	0.210	0.709	0.647	0.439	0.504	+0.271
OSRM	−0.109	0.233	0.178	0.362	−0.167	0.000	0.000	0.717	0.152	+0.622
LoRA–Soups/CAT	0.024	0.547	0.699	0.396	0.354	0.548	0.667	0.759	0.499	+0.275
Compress-then-Merge	−0.167	0.366	0.718	**0.459**	−0.167	0.492	0.624	0.771	0.387	+0.387
NSC	−0.010	0.546	0.725	0.394	0.331	0.565	0.667	0.755	0.497	+0.278

independent generative-classification LoRA source adapter (r=8, α=16, q/v projections, shared causal-LM head) under one fixed recipe. Every READ append then trains one pass over the full stage task union with the same optimizer, schedule, batch, and length for all compared growth arms; the manipulated variable is the factor coordinate system and/or the trainable mask on G, not data exposure or tuning. The comparison therefore isolates the foldable interaction rule, since appending itself still trains on old-task data. All final evaluations use full configured splits, batch 64, input length 512, and each task's official primary metric; suite "terminal macro" is the mean over task scores.

The headline campaign contains 32 READ lineages (2 models $\times$ 4 suites $\times$ 2 seeds $\times$ 2 task orders) and 120 adjacent appends. The terminal comparison matches each final bank against the strongest of fourteen *exact-source foldable* alternatives: seven tensor merges (Wortsman et al., 2022; Matena & Raffel, 2022; Ilharco et al., 2023; Jin et al., 2023; Yadav et al., 2023; Yu et al., 2024; Davari & Belilovsky, 2024), three direct operators (Zhao et al., 2024; Stoica et al., 2025; Huang et al., 2024a), and four recent published systems (Zhang & Zhou, 2025; Prabhakar et al., 2025; He et al., 2026; Lee et al., 2026). The append comparison applies the preregistered rule, requiring at least half the Soup-to-refit incoming gap closed, mean old-task drop no worse than 2 points, and no old task worse than 5 points. A compact gauge check tests the coordinate choice, and the append ledger traces bank-level stability across additions. Intervals are bootstrap 95% confidence intervals over lineages.

4.2 Q1: Does READ build a better final skill bank?

The strongest foldable alternative changes from bank to bank. RegMean is best on 18 lineages, LoRAHub–5shot on six, and Compress-then-Merge on four. Table 1 therefore compares READ against the best composition available from the same source adapters in each case. READ improves on that per-lineage maximum on 24 of 32 lineages, by $+0.073$ on average (95% CI $[+0.047, +0.101]$). The eight losses are all on BBH and are narrow, -0.014 on Llama and -0.023 on Qwen at suite level.

It is worth noting who the strongest competitor usually is. RegMean, a dataless merge from 2022, occupies that row far more often than any recent system, while the newest published methods sit at the bottom of the table, with overall averages of 0.152 for OSRM and 0.387 for Compress-then-Merge against RegMean's 0.684. On GLUE in particular, seven of the fourteen alternatives score below zero on Llama, which means a merged bank can end up worse than no merge on the suite's harder components. A method that has to beat the per-lineage maximum therefore has to beat a strong classical baseline, not a weak recent one.

The per-suite margins are consistent across seeds and task orders. All 24 non-BBH terminal lineages are READ wins. Llama gains most on GLUE and SuperGLUE ($+0.192$ and $+0.178$), and Qwen's margins on the three non-BBH suites are $+0.063$, $+0.088$, and $+0.058$. Domain shows the smallest Llama margin ($+0.041$), where the strongest alternative already reaches 0.846. Read directly off

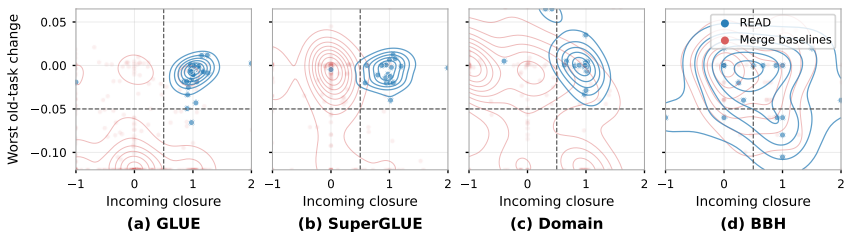


Figure 3: Append phase space by suite. Blue marks READ and red marks merge baselines; dashed lines mark pass thresholds.

Table 2: Sequential append reliability. Pass rate and mean incoming closure are shown in separate rows; coverage gives eligible/observed transitions.

		GLUE-6		SuperGLUE-4		BBH-6		Domain-3	
Method	Metric	Llama	Qwen	Llama	Qwen	Llama	Qwen	Llama	Qwen
Coverage	Eligible / observed	19/20	20/20	11/12	9/12	7/20	10/20	8/8	8/8
Soup	Pass rate (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Incoming closure (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Fisher	Pass rate (%)	0.0	5.0	0.0	0.0	42.9	20.0	0.0	0.0
	Incoming closure (%)	-4522.8	-810.0	-45.8	-1481.8	65.5	17.5	-856.5	-594.6
Task Arithmetic	Pass rate (%)	0.0	0.0	0.0	11.1	14.3	40.0	0.0	12.5
	Incoming closure (%)	-144.0	-57.9	-16.6	24.3	18.2	49.2	-241.8	-85.5
RegMean	Pass rate (%)	26.3	50.0	27.3	22.2	85.7	10.0	25.0	50.0
	Incoming closure (%)	-25.9	-276.0	29.1	25.4	72.3	72.3	13.8	89.0
TIES	Pass rate (%)	10.5	0.0	0.0	22.2	57.1	30.0	0.0	0.0
	Incoming closure (%)	19.2	-296.6	0.3	-36.1	83.4	124.3	-9.5	-89.5
DARE	Pass rate (%)	0.0	0.0	0.0	11.1	14.3	50.0	0.0	12.5
	Incoming closure (%)	-157.7	-64.3	-14.8	23.5	19.0	83.3	-239.9	-91.7
Breadcrumbs	Pass rate (%)	0.0	0.0	0.0	0.0	14.3	40.0	0.0	0.0
	Incoming closure (%)	-180.5	-252.6	-25.1	-1065.6	41.4	24.5	-369.4	-740.7
LoRAHub	Pass rate (%)	15.8	5.0	9.1	0.0	42.9	20.0	0.0	0.0
	Incoming closure (%)	67.8	-433.9	-310.5	-164.0	120.4	86.8	-1014.1	56.0
READ (ours)	Pass rate (%)	89.5	85.0	90.9	100.0	28.6	40.0	87.5	75.0
	Incoming closure (%)	89.3	123.2	104.0	95.3	57.8	60.0	70.7	82.7

Table 1, READ is the column best on all seven non-BBH cells and trails the winner on the two BBH cells by only 0.014 (Llama) and 0.020 (Qwen).

4.3 Q2: Can READ append skills without erasing the bank?

A strong terminal bank can hide damaging intermediate appends, so we examine every transition. An append passes when it closes at least half of the Soup-to-refit incoming gap, with mean and worst old-task changes no lower than -2 and -5 points. READ passes 72 of 92 eligible transitions across the expanded grid (Table 2).

The merge baselines fail on both axes of Figure 3 at once. Fisher's mean incoming closure is negative in six of the eight model–suite cells, down to -4522.8% on Llama GLUE, which means the merged bank ends up further from the refit target on the new task than the soup anchor it started from. Task Arithmetic, DARE, and Breadcrumbs show the same pattern at smaller magnitudes. READ's mean closure is positive in every cell, never below 57.8%, and its worst old-task changes stay inside the -5 bound wherever the append passes.

READ's pass rates split cleanly by suite. On GLUE, SuperGLUE, and Domain, READ passes at least 75% of eligible transitions for both models. On BBH it passes only 28.6% on Llama and 40.0% on Qwen, and these are acquisition failures. Mean incoming closure on BBH stays positive (57.8% and 60.0%), so the bank still gains part of the new skill on average, but one pass over the stage union does not reliably close half the gap on these tasks. Old-bank preservation is not what fails; the mean and worst old-task bounds hold even on the failed BBH transitions. The group-level view agrees. Six of the eight model–suite groups pass the group rule. The two misses, Llama SuperGLUE and Llama Domain, fail only the retained-share clause while their mean old-task damage stays within the -2

bound. Read across methods, READ is also the only row of Table 2 whose incoming-closure entries are all positive.

4.4 Q3: Does read-only growth stabilize the skill bank?

Terminal scores do not show whether the bank followed a stable path to reach them. The gauge check quantifies the coordinate problem directly. Re-factoring the same adapters into equivalent raw factorizations shifts downstream scores by 2.8 points on average, and canonicalization shrinks that spread to 0.7. Without canonical coordinates, what a coupling learns depends partly on which factorization the source training happened to produce.

The append ledger then traces the bank itself (Figure 4). Merge trajectories turn and spread after individual additions, so a later skill joins a representation that has already moved away from the bank its earlier skills were composed into. READ's trajectory makes small, similarly oriented moves from the initial bank onward. The distribution panels show the same contrast in score terms. READ's closures sit mostly on the positive side of the pass threshold, and its worst old-task changes concentrate near zero, while the merge baselines spread toward retention loss.

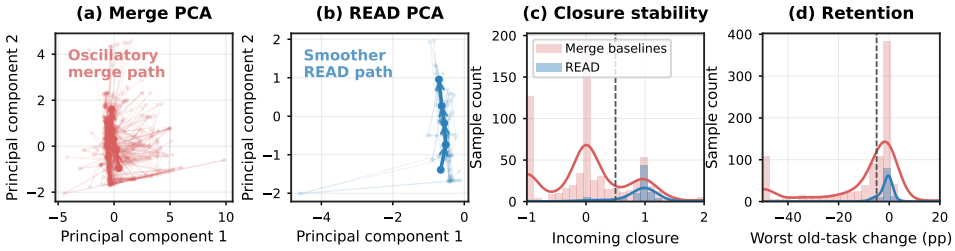


Figure 4: Trajectory and stability summaries from the append ledger. The first two panels show PCA trajectories; the last two show closure and retention-damage distributions.

Two controls isolate where the stability comes from. Random norm-matched factors underperform the learned factors in 26 of 27 measured cells, so the coupling is exploiting structure in the trained updates rather than generic extra capacity. Global diagonal refits reopen established couplings and become seed sensitive, which is the failure READ's freeze rule removes by construction.

4.5 Q4: Does READ remain a single-model operator?

Six of the eight model–suite groups pass the whole-lineage rule, and canonicalization contracts the mean spread across equivalent source factorizations from 2.8 to 0.7 points. The gauge contraction is a robustness result rather than an average-utility gain: the complete component factorial places all six coordinate–direction arms within 0.017 terminal macro, with overlapping confidence intervals. The remaining question is whether the resulting operator survives as a serving object.

Figure 5 first confirms that reliability is a property of the growing bank rather than of the terminal prefix alone. Old-bank change stays near zero for READ at every append depth on every suite, and incoming score follows the boundary already visible in Q2, with READ tracking the acquisition ceiling on GLUE, SuperGLUE, and Domain while BBH flattens for both READ and the merge baselines.

Folding then removes any residual READ path from that serving object. It reproduces the unfolded operator in all 128 model–configuration checks, with maximum logit deviation 4.25×10^{-4}. Before folding, the bank adds about 10–28% mean latency across the four measured batch–length settings; after folding, the corresponding means remain within roughly one percent of the base model. The auxiliary bank occupies 79.5–166.9 MB and 6.9–17.9 million floats before folding, while the deployed checkpoint is an ordinary 6.4–12.9 GB model with no router, task identity, or extra inference branch. Against an oracle-router upper bound trained on the same adapters, the folded model averages $+0.064$ terminal macro at order-averaged bank size, so one merged projection serves at or above the routing headroom on the suites where composition adds signal, and READ pays its interaction cost while a skill is appended rather than each time the resulting bank is served.

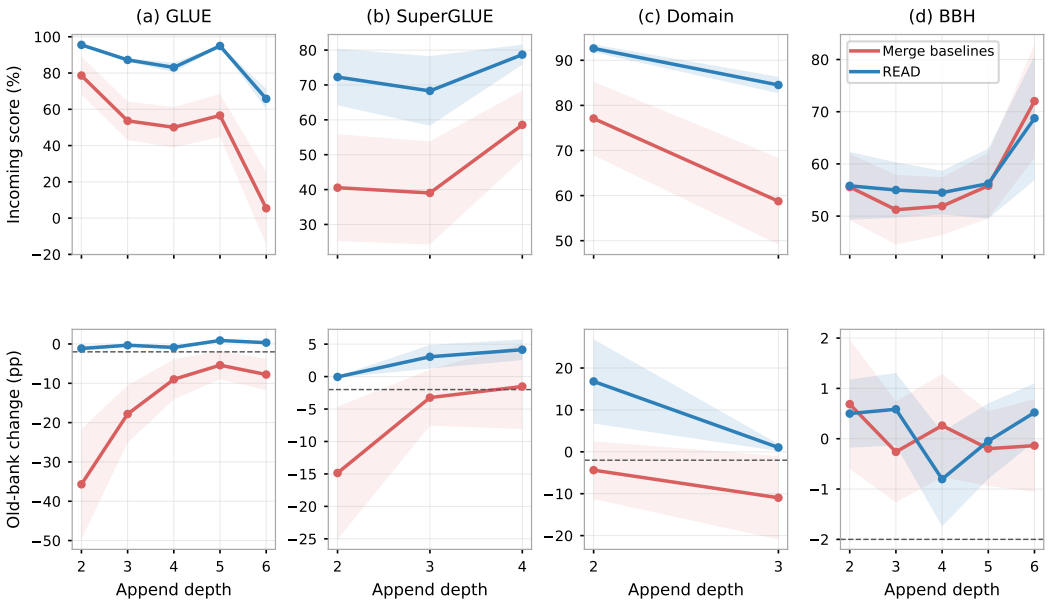


Figure 5: Append dynamics by suite. Top row: incoming score at each append depth. Bottom row: mean old-bank change. Bands are one standard deviation across lineages; the dashed line marks the -2-point retention threshold.

5 Conclusion

When a new skill must share a model with old ones, one side has to adapt, and the choice of which side determines whether the old skills survive. This paper traced that choice to two freedoms a lone adapter never exposes, the coordinates of its factors and the direction of each cross-skill link. Composition only works reliably when a method fixes both, instead of inheriting whatever the source training runs happened to produce.

READ fixes them. Every adapter is rewritten into balanced canonical coordinates that preserve its update exactly, and the coupling grows in one direction only, so each new skill reads the frozen bank without writing into old output subspaces. Across 32 terminal lineages this beats the strongest of fourteen foldable alternatives built from the same adapters by $+0.073$ on average, and 72 of 92 sequential appends pass a preregistered reliability rule. The composed update folds into a single rank-kr weight change that serves at base-model latency and, at the same order-averaged bank size, sits above an oracle-router upper bound on the suites where composition adds signal.

The remaining boundary is BBH, where all eight terminal losses concentrate. The failures there are acquisition failures rather than retention failures. One pass over the stage union does not always give the new skill enough signal, and the read-only constraint protects the bank but cannot itself supply that missing signal. How to strengthen acquisition while keeping the bank frozen, for example by giving the new skill's row more than one pass or a richer state than the frozen factors alone, is the direction we consider most worth pursuing. Two boundaries lie outside the present evaluation and belong on that same agenda: open-ended generative composition, where the primary metric is not classification accuracy, and banks beyond six skills, where the coupling stays quadratic in k even though the trained slice does not.

AI Use Statement

AI tools were used to aid and polish writing, and to support retrieval and discovery of related work. All scientific claims, experiments, analyses, and final manuscript decisions were reviewed and controlled by the authors.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2209.04836.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1/. arXiv:2106.10199.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision (ECCV)*, pp. 270–287, 2024. arXiv:2312.06795.

Guodong Du, Zhuo Li, Xuanning Zhou, Junlin Li, Zesheng Shi, Wanyu Lin, Ho-Kin Tang, Xiucheng Li, Fangming Liu, Wenya Wang, Min Zhang, and Jing Li. Knowledge fusion of large language models via modular SkillPacks. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=wJtD28wGV1. arXiv:2505.18502.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2110.06296.

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. arXiv:2101.03961.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, volume 119, pp. 3259–3269, 2020. arXiv:1912.05671.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8803–8812, 2018. arXiv:1802.10026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2110.04366.

Zhengbao He, Ruiqi Ding, Zhehao Huang, Ruikai Yang, Tao Li, and Xiaolin Huang. Compress then merge: From multiple loras into one low-rank adapter. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2606.03723.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html. arXiv:1902.00751.

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 (ICLR)*, 2022. arXiv:2106.09685.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling (COLM)*, 2024a. arXiv:2307.13269.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-Merging: Tuning-free high-performance model merging. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 122741–122769, 2024b. arXiv:2405.17461.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2212.04089.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision (ECCV)*, pp. 207–223, 2024. arXiv:2403.19522.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2212.09849.

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

Wonyoung Lee, Wooseong Jeong, and Kuk-Jin Yoon. Label-free cross-task lora merging with null-space compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2603.26317.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3045–3059, 2021. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/. arXiv:2104.08691.

Wenbing Li, Zikai Song, Hang Zhou, Yunyao Zhang, Junqing Yu, and Wei Yang. LoRA-Mixer: Coordinate modular LoRA experts through serial attention routing. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.00029.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/. arXiv:2101.00190.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1950–1965, 2022. arXiv:2205.05638.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2106.04647.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 17703–17716, 2022. arXiv:2111.09832.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 36829–36846, 2024. arXiv:2402.05859.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1710.10628.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, 2021. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39/. arXiv:2005.00247.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. Lora soups: Merging loras for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 644–655, 2025. doi: 10.18653/v1/2025.coling-industry.55. URL https://aclanthology.org/2025.coling-industry.55/. arXiv:2410.13025.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_. arXiv:2301.12314.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7930–7946, 2021. doi: 10.18653/v1/2021.emnlp-main.626. URL https://aclanthology.org/2021.emnlp-main.626/. arXiv:2010.11918.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg. arXiv:1701.06538.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq. arXiv:2305.03053.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the KnOTS. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=67X93aZHII. arXiv:2410.19735.

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 *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1804.07461.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/. arXiv:2310.14152.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, pp. 23965–23998, 2022. arXiv:2203.05482.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7093–7115, 2023. arXiv:2306.01708.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *arXiv preprint arXiv:2410.03617*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning (ICML)*, pp. 57755–57775, 2024. arXiv:2311.03099.

Haobo Zhang and Jiayu Zhou. Unraveling lora interference: Orthogonal subspaces for robust model merging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26459–26472, 2025. doi: 10.18653/v1/2025.acl-long.1284. URL https://aclanthology.org/2025.acl-long.1284/. arXiv:2505.22934.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Kun Kuang, and Fei Wu. Merging LoRAs like playing LEGO: Pushing the modularity of LoRA to extremes through rank-wise clustering. *arXiv preprint arXiv:2409.16167*, 2024.