# WATCH WHERE YOU LINEARIZE: PREDICTING AND CORRECTING MODEL-MERGING INTERFERENCE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Model merging composes separately fine-tuned models by adding their task vectors to a shared backbone. In standard merging pipelines, the resulting functional interference is known only after candidate merges are built and evaluated. We show that a single directional derivative, the receiving model's output response $J_A(\theta_A)\tau_B$ to another task's vector, forecasts the interference before any candidate is evaluated, predicting logit drift, label flips, per-example damage, and per-task accuracy of the merge. The forecast works only in the receiver's local geometry: read at the shared pretrained initialization, the same derivative is nearly useless, because fine-tuning reorients the cross-task responses at the very beginning of training. Freezing the receiver-local responses yields a surrogate whose outputs are affine in the merge coefficients, and with it **LR-Merge**, a label-free correction that selects coefficients through a sequence of convex teacher-matching problems on task-specific unlabeled probes, never backpropagating through a candidate merged network. On vision backbones and language models, LR-Merge improves every strong base merger we test. Its advantage concentrates where merging is data-scarce or repeated. At low probe budgets it surpasses strong coefficient-optimization baselines given several times more data and compute, and the frozen responses are built once and reused, so each further coefficient setting or task subset costs seconds rather than a rebuild.

## 1 INTRODUCTION

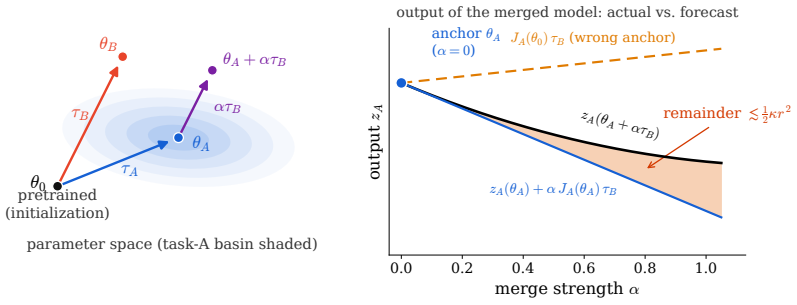


Figure 1: Interference is receiver-local: the response at the receiving model (blue) tracks the merged network's output (black). At the pretrained $\theta_0$ (orange) it points in a nearly unrelated direction.

Model merging composes separately fine-tuned models into one network, without joint training or the original data, by adding each model's parameter difference from the shared pretrained starting point, its *task vector* (Wortsman et al., 2022; Ilharco et al., 2023). The updates interfere: a vector learned for one task changes predictions on another and costs accuracy. Existing methods reduce the damage by altering the composition: sparsification (Yadav et al., 2023; Yu et al., 2024), per-task coefficients (Yang et al., 2024b), parameter statistics (Matena & Raffel, 2022; Jin et al., 2023), representation repair (Yang et al., 2024a). But the functional effect of a candidate merge is typically known only after the merged model is built and evaluated. Yet everything the merge will be made



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.

of is known in advance: the task vectors and the models that will receive them. This paper treats merging as a *prediction* problem: can the effect of adding a task vector be known before any merge is built?

Answering this question changes how a merging pipeline can be organized. Pipelines that search over merge coefficients, sparsity settings, or task subsets repeatedly construct and evaluate candidate networks, with costs that grow with the number of candidates and tasks. Closed-form mergers provide an initial composition without such a search; an interference forecast can then guide coefficient refinement around that model. Candidates can be scored before their networks are built, reserving full-network evaluation for selected candidates and identifying examples and tasks likely to be affected by the update. The computational savings come from reuse: responses are built once per anchor, and further coefficient settings or task subsets are scored using the cached responses. This paper develops such a predictor and a merging method built on it.

The object that answers the question is the *cross-task directional response* $J_A(\theta)\,\tau_B$: with $\theta_0$ the shared pretrained model, fine-tuned models $\theta_A = \theta_0 + \tau_A$, $\theta_B = \theta_0 + \tau_B$, and $J_A$ the output Jacobian on target task $A$'s inputs, it is the first-order change task $A$'s outputs undergo when task $B$'s vector is added at $\theta$, one Jacobian–vector product (JVP). First-order expansion is standard. The scientific question is *at which $\theta$ the response must be read.*

Our central finding is that interference is **receiver-local**. Evaluated at the fine-tuned model receiving the update, $J_A(\theta_A)\tau_B$ accurately describes the interference that actually occurs: which examples flip label, how much each task's accuracy drops, in what direction the logits drift. Evaluated at the pretrained initialization, the common anchor of task arithmetic and of tangent-space reasoning, $J_A(\theta_0)\tau_B$ carries almost no usable signal. The failure coincides with a dynamical event: at the very beginning of fine-tuning the network reorients its cross-task responses, and the initialization response becomes obsolete. Generic perturbations of the receiver at matched distance, by contrast, leave the forecast intact: the failure depends on direction rather than distance alone.

The discovery has a direct algorithmic consequence. If the receiver-local responses are accurate, then *freezing* them gives a surrogate for nearby merges whose outputs are affine in the merge coefficients. Choosing coefficients then becomes, at each anchor, a convex, label-free problem, matching the fine-tuned teachers' predictions on unlabeled probes, that never backpropagates through a candidate merged network. We call the resulting method **Linear-Response Merge (LR-Merge)**. Because the response is a derivative, its accuracy is local: it is sharpest near an already-strong merge and degrades far from one. LR-Merge therefore completes existing mergers rather than competing with them: the base merger supplies a strong candidate, and the frozen responses replace the expensive search around that candidate with convex solves, on vision and language backbones alike. The method directly depends on receiver-locality: rebuilding the surrogate from pretrained responses, everything else held fixed, destroys the correction and pushes the corrected merge *below* its own base (Section 6.1).

**Contributions.**

1. **Merging as prediction.** We recast merging pipelines from build-and-evaluate search to forecast-then-build, and provide an interference predictor for this role, finer-grained than prior mergeability and error predictors: a receiver-local directional derivative that reveals, before any candidate is evaluated, which examples will flip, how the logits will drift, and what each task's accuracy will be. Interference can then be estimated before constructing the candidate merge. Richer predictors can substitute into the same role.
2. **Receiver-locality of the response.** A sharp empirical regularity of fine-tuning: within the first steps of training the network re-aims its cross-task responses, leaving the initialization anchor inherited from task arithmetic and tangent-space methods with almost no usable signal, while the receiver's anchor forecasts nearly perfectly and generic matched-distance perturbations cost nothing. A nine-anchor ladder strongly associates forecast degradation with displacement along the receiver's task vector, the displacement along the receiver's own task vector.
3. **LR-Merge: fast, probe-efficient correction.** Freezing receiver-local responses turns coefficient selection into a sequence of cheap convex teacher-matching fits, with no backpropagation through any candidate network. Across vision backbones and language models,

it improves every strong base merger we test, wins by the widest margins exactly where merging is hardest, under scarce unlabeled probes and growing task counts, and its one-off build becomes a cache that answers each further coefficient setting or task subset in seconds.

## 2 Related Work

**Model merging and interference.** Merging began as weight averaging of models fine-tuned from a shared initialization (Wortsman et al., 2022), and task arithmetic (Ilharco et al., 2023) made the composition explicit by treating each model's parameter difference as a task vector to be added at chosen coefficients. Once vectors from different tasks are added together they interfere, and the field's main response has been to edit the vectors before adding them: sparsifying entries and resolving sign conflicts (Yadav et al., 2023; Yu et al., 2024; Davari & Belilovsky, 2024; Du et al., 2024), masking or selecting parameters per task (Wang et al., 2024; Qi et al., 2025; Huang et al., 2024), or restructuring the vectors in low-rank and spectral subspaces (Xiong et al., 2024; Cheng et al., 2025; Liu et al., 2025; Marczak et al., 2025; Zheng & Wang, 2025), or adaptively projecting the merged update toward the task models without data (Wei et al., 2025). A second line spends calibration data instead of editing vectors, weighting the composition by parameter importance (Matena & Raffel, 2022), solving layer-wise regressions from activation statistics (Jin et al., 2023), descending prediction entropy on unlabeled inputs (Yang et al., 2024b), or repairing the merged representations after the fact (Yang et al., 2024a). These methods also provide starting points for output-space correction. Output-space projection merging (Evans et al., 2026) formulates calibration against fine-tuned outputs as a convex quadratic program over layer-wise residual weights, with a basis-dependent error analysis and a sequential multi-layer extension. LR-Merge shares the use of an affine output model and teacher targets. It evaluates directional responses to task vectors or their parameter groups at the current merge, then jointly optimizes their coefficients using cross-entropy. Our analysis characterizes how the choice of anchor affects these responses after nonlinear fine-tuning, when their reorientation occurs, and how accurately they forecast per-example interference. Freezing the responses then supports their reuse across coefficient and task-subset queries.

**Tangent-space and linearized composition.** A different route to controlling interference is to make linearity true by construction: tangent task arithmetic trains and composes models within a linearization fixed at initialization, fully (Ortiz-Jimenez et al., 2023; Liu & Soatto, 2023) or over a parameter subset (Tang et al., 2024), and at large scale fine-tuning has been argued to stay close to its initial kernel (Malladi et al., 2023). We study the complementary setting in which task vectors come from ordinary nonlinear fine-tuning. We ask where their directional responses should be evaluated to forecast interference at a fine-tuned receiver. Across the studied checkpoints, receiver-anchored responses give substantially more accurate forecasts than pretrained responses. Tracking these responses during fine-tuning reveals when this anchor gap emerges.

**Predicting the effects of parameter updates.** Zhou et al. (2026) predict overall merge quality using function-space quantities, while Daheim et al. (2024) bound merge error through gradient mismatch. Tangent overlap informs forgetting analyses (Doan et al., 2021; Evron et al., 2022; Hidekel & Raviv, 2026), and influence functions (Koh & Liang, 2017; Grosse et al., 2023) and learned update predictors (Jin & Ren, 2024) estimate parameter-update effects more broadly. Zhou et al. (2024) observe cross-task feature linearity under weight interpolation. We forecast per-example output changes caused by task-vector updates after nonlinear fine-tuning, characterize the anchor dependence and early reorientation of directional responses, and reuse receiver-anchored responses for merge correction.

## 3 Predicting Merge Interference

**Setup.** Unless stated otherwise, all measurements use CLIP ViT-B/32 (Radford et al., 2021) on six tasks (MNIST, SVHN, GTSRB, EuroSAT, DTD, CIFAR-10), each fine-tuned independently from the same pretrained model under the standard task-vector protocol (Ilharco et al., 2023), with 256 unlabeled probe examples per task and index-disjoint 1000-example evaluation splits. Headline comparisons are replicated over six independently fine-tuned checkpoint sets (Appendix E).

**The forecasting problem.** Fine-tuning task $t \in \{1, \dots, T\}$ independently from $\theta_0 \in \mathbb{R}^P$ produces parameters $\theta_t$ and task vector $\tau_t := \theta_t - \theta_0$. Fix a target task $A$ with probe inputs $X_A = \{x_{A,i}\}_{i=1}^{n_A}$, let $z_A(\theta) \in \mathbb{R}^{n_A C_A}$ be the stacked logits on those inputs (fixed task heads absorbed into $z_A$), and write $J_A(\theta) := \frac{\partial z_A(\theta)}{\partial \theta}$. The interference from adding task $B$'s vector to the model fine-tuned on $A$ is the displacement $\Delta z_{A \leftarrow B}(\alpha) := z_A(\theta_A + \alpha\tau_B) - z_A(\theta_A)$; first-order expansion at the model receiving the update gives

$$z_A(\theta_A + \alpha\tau_B) = z_A(\theta_A) + \alpha\, J_A(\theta_A)\tau_B + R_{A,B}(\alpha), \tag{1}$$

which motivates the *receiver-local forecast* $\widehat{\Delta z}_{A \leftarrow B}(\alpha) := \alpha\, J_A(\theta_A)\tau_B$ (one JVP at the receiving model, without ever building the merge) (remainder bounds in Appendix A).

Table 1 compares this forecast with the realized displacement across all 30 ordered target–source pairs. At the merge strengths coefficient search visits ($\alpha \lesssim 0.5$), the forecast gets per-task accuracy right to a few tenths of a point, identifies about four in five of the examples that flip label, and ranks which examples get hurt. This is the information a practitioner currently obtains only by building and evaluating the candidate, available here before the merged model exists. The forecast replicates across the six checkpoint sets and across CLIP backbones (Section 6.3 and Appendix B).

Table 1: Pairwise forecast quality, 30 ordered task pairs, checkpoint set 0, three probe seeds; metrics defined in the text, per-checkpoint-set values in Table 7.

| $\alpha$ | acc. MAE (pp) | flip recall | damage $\rho$ | mean drift cos |
|---|---|---|---|---|
| 0.2 | 0.13 | 0.82 | 0.917 | 0.946 |
| 0.4 | 0.30 | 0.75 | 0.824 | 0.905 |

Each column of Table 1 answers a different operational question. The accuracy error says how far the forecasted merged accuracy can be trusted: the per-task numbers a practitioner reads off the forecast are within a few tenths of a point of what the built merge would score. Flip recall is triage: among the examples whose labels the merge will actually change, the forecast names most of them individually, so the inputs worth inspecting are known before any damage occurs. The damage rank correlation orders the rest: when inspection budgets are limited, it says which examples to look at first. Together, these metrics measure aggregate accuracy, identify affected examples, and rank the most damaging cases.

Interference is not readable off the task vectors alone. Scores computed from the vectors by themselves (task-vector cosine and norm, sign conflict, feature cosine) correlate weakly or negatively with the realized per-pair accuracy drop, while the receiver-local forecast correlates strongly. The effect depends on the receiver-specific $J_A(\theta)$. The forecast also applies to the update of any merger taken as a black box: whatever method produced a merged model $\theta_M$, the receiver sees the displacement $\theta_M - \theta_A$, and the same directional derivative $J_A(\theta_A)(\theta_M - \theta_A)$ predicts its response.

A single receiver-local JVP is a local forecast of a nearby candidate merge. For larger moves, the response can be re-read along the merge path (Appendix A.2): on six-way merges, moving from one midpoint re-evaluation of the response to four makes the forecast nearly exact.

Everything above reads the response at the receiving model. The place one would *want* to read it is the shared pretrained initialization $\theta_0$, the anchor of task arithmetic and of tangent-space composition, the one point common to all tasks, and the only one available before fine-tuning. Read there, the same derivative predicts almost nothing: the mean drift cosine falls from 0.949 to 0.122 (Table 2). Why the same expansion, on the same vectors and probes, works at $\theta_A$ and fails at $\theta_0$ is the subject of the next section.

## 4 Anchor Dependence of the Forecast

Task arithmetic defines vectors relative to $\theta_0$ (Ilharco et al., 2023), and tangent-space composition fixes its linearization there (Ortiz-Jimenez et al., 2023; Liu & Soatto, 2023). Kernel-regime analyses also suggest that initial linearized dynamics can persist during fine-tuning (Malladi et al., 2023). We therefore test whether pretrained directional responses accurately forecast updates at fine-tuned receivers. Replacing $J_A(\theta_A)\tau_B$ with $J_A(\theta_0)\tau_B$ introduces the response mismatch

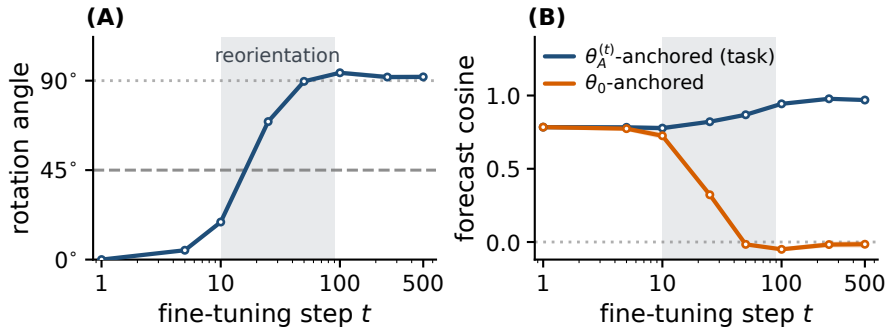


Figure 2: Cross-task responses rotate at the very beginning of fine-tuning. (A) Angle between the response at the current checkpoint, $J_A(\theta_A^{(t)})\tau_B$, and the same response at initialization, $J_A(\theta_0)\tau_B$, as fine-tuning of the receiver proceeds. (B) The $\theta_0$-anchored forecast collapses over the same window; the $\theta_A^{(t)}$-anchored forecast keeps rising.

$[J_A(\theta_A) - J_A(\theta_0)]\tau_B$. Section 3 measures a substantial forecast-quality gap between these anchors. We now trace its emergence during training and examine which anchor displacements degrade the forecast.

The failure is an early-training event. We track $r_{A\leftarrow B}(t) := J_A(\theta_A^{(t)})\tau_B$ along the actual optimization checkpoints $\theta_A^{(t)}$ for ten ordered target–source pairs over two receivers, replicated over three fine-tuning seeds (Figure 2). The angle to the initialization direction $J_A(\theta_0)\tau_B$ crosses $45°$ within steps 10–25 and saturates by step 50, at the very beginning of fine-tuning. The $\theta_0$-anchored forecast collapses over exactly this window, while a forecast re-read at the current checkpoint stays accurate throughout (Figure 2B). Two further receivers replicate the early crossing at different plateau heights, so the receiver-robust statement is the early crossing and the plateau itself, not a universal plateau height (Appendix B).

The anchor controls (Table 2) empirically localize what breaks. An isotropic anchor at the same distance from $\theta_A$ forecasts as well as the receiver itself: distance is not the cause, and the receiver's exact weights are not required. Here, "receiver-local" refers to direction rather than Euclidean proximity, an anchor being effectively local when its displacement from the receiver has little component along the task-vector directions. What fails is structured direction: an anchor displaced within the span of the *other* task vectors at matched norm degrades substantially, and $\theta_0$, reached purely along the fine-tuning direction $-\tau_A$, collapses. The full ladder (a direction dose–response, a third fine-tuned model, an isotropically perturbed $\theta_0$) orders by a single coordinate, the displacement of the anchor along the receiver's own $\tau_A$ (Spearman $\rho = -0.98$ against $-0.16$ for distance, Appendix B). Fine-tuning primarily re-aims the response rather than attenuating it (Appendix B).

## 5 LR-Merge: Freezing the Local Response

Table 2: Forecast drift cosine by anchor: 30 ordered pairs at $\alpha$=0.2, mean±sd over six checkpoint sets, min = worst pair. Full ladder in Table 6.

| anchor | what it tests | mean cos | min cos | dist. from $\theta_A$ |
|---|---|---|---|---|
| $\theta_A$ (receiving model) | the forecast as used | 0.949±0.008 | 0.296 | 0 |
| isotropic, matched distance | is distance the cause? | 0.945±0.009 | 0.315 | $\|\tau_A\|$ |
| span of other task vectors | is the direction structured? | 0.665±0.036 | −0.320 | $\|\tau_A\|$ |
| $\theta_0$ (pretrained) | the natural alternative | 0.122±0.011 | −0.719 | $\|\tau_A\|$ |

Forecasting one candidate is useful, but coefficient selection for the task vectors requires comparing many nearby candidates. LR-Merge therefore freezes the receiver-local responses at one anchor and reuses them throughout the search: build a local linear response model around a strong merge, and optimize the merge coefficients against fine-tuned teacher predictions inside that surrogate. Let $\bar{\theta}$ denote the current anchor, in the intended use, a merge produced by a base method such as Reg-

Mean (Jin et al., 2023), TIES (Yadav et al., 2023), or Iso-C (Marczak et al., 2025). We search over local task-vector corrections $\tilde{\theta}(\boldsymbol{\delta}) := \bar{\theta} + \sum_{t=1}^{T} \delta_t \tau_t$. Linearizing the target-task outputs at the anchor gives the affine output-space surrogate

$$\widehat{z}_A(\boldsymbol{\delta}) := z_A(\bar{\theta}) + \sum_{t=1}^{T} \delta_t D_{A,t}, \qquad D_{A,t} := J_A(\bar{\theta})\tau_t. \tag{2}$$

Here $D_{A,t}$ is shorthand, introduced now that the method iterates over many task pairs, for the receiver-local response $J_A(\cdot)\tau_t$ of Section 3, read at the current anchor. Once frozen, every candidate's outputs are linear combinations of stored vectors: no merged network is built or run during the search, and nothing is backpropagated through a candidate. The cost is $T$ JVPs per target, $T^2$ in all. The grouped setting uses per-group responses $D_{A,t,g} := J_A(\bar{\theta})[\Pi_g \tau_t]$ ($\Pi_g$ zeroes parameters outside group $g$), with $G{=}13$ groups (one per transformer block plus embedding/projection) at $T^2G$ JVPs per anchor, and the surrogate remains affine. This is the operative form of receiver-locality: read the response at the *current iterate*, $\theta_A$ for pairwise interference and the current merge $\bar{\theta}$ for correction (measured directly at the deployed anchors, the forecast reaches drift cosine $0.91$ at the typical correction magnitude, Appendix B). The anchor comparison motivates evaluating responses at the current merge.

Why is a frozen response useful, when Section 4 has just shown that responses go stale? The surrogate is a local first-order model (the same object Section 3 validated pairwise, reused across candidates instead of recomputed for each), so LR-Merge centers each fixed-anchor objective at the current coefficients through the ridge on $\boldsymbol{\alpha} - \mathbf{a}^{(r)}$. The optimizer itself is unconstrained. After the fixed optimization schedule, only the clipped coefficient vector is applied to the model, and in the deployed configuration the clamp does not bind (Appendix C). The method then re-anchors at that model and recomputes the responses before the next round. This is what the second round is for. LR-Merge handles locality by favoring small corrections and re-reading the geometry after each applied move, rather than by requiring every optimizer iterate to stay inside a local neighborhood.

**Convex coefficient selection.** For each task $A$, the fine-tuned teacher assigns pseudo-labels $\widehat{y}_{A,i} := \arg\max_c z_A(\theta_A)_{i,c}$ to its unlabeled probes. Hard-label cross-entropy targets agreement with the teacher's class decisions without matching its logit magnitudes or full probability distribution. Selection uses task-specific probes but requires no ground-truth labels. The search is over the task vectors themselves: a candidate merge is $\theta(\boldsymbol{\alpha}) := \theta_{\text{base}} + \sum_{t=1}^{T} \alpha_t \tau_t$ with *total* coefficients $\boldsymbol{\alpha} \in \mathbb{R}^T$, where $\theta_{\text{base}}$ is the model the correction starts from. (The grouped variant gives each per-block piece $\Pi_g \tau_t$ its own coefficient, replacing the task index $t$ by the pair $(t, g)$ with $M{=}TG$ coefficients in all. Everything below reads unchanged.) Round $r$ holds current total coefficients $\mathbf{a}^{(r)}$ and the corresponding anchor model $\bar{\theta}^{(r)} := \theta(\mathbf{a}^{(r)})$; a candidate $\boldsymbol{\alpha}$ has local correction $\boldsymbol{\delta} := \boldsymbol{\alpha} - \mathbf{a}^{(r)}$, and the surrogate equation 2, rebuilt at $\bar{\theta}^{(r)}$, reads $\widehat{z}_A^{(r)}(\boldsymbol{\alpha}) = z_A(\bar{\theta}^{(r)}) + \sum_{t=1}^{T} \big(\alpha_t - a_t^{(r)}\big) D_{A,t}^{(r)}$ with $D_{A,t}^{(r)} := J_A(\bar{\theta}^{(r)})\, \tau_t$. LR-Merge selects coefficients by minimizing, over $\boldsymbol{\alpha} \in \mathbb{R}^M$,

$$F_r(\boldsymbol{\alpha}) := \frac{1}{T} \sum_{A=1}^{T} \frac{1}{n_A} \sum_{i=1}^{n_A} \ell_{\text{CE}}\Big(\widehat{z}_A^{(r)}(\boldsymbol{\alpha})_{i,\cdot}, \widehat{y}_{A,i}\Big) + \lambda \|\boldsymbol{\alpha} - \mathbf{a}^{(r)}\|_2^2. \tag{3}$$

The surrogate logits are affine in $\boldsymbol{\alpha}$, so each fixed-anchor objective $F_r$ is convex and, with the ridge on the correction, $2\lambda$-strongly convex on $\mathbb{R}^M$ with a unique unconstrained minimizer (Appendix A.3). Each round approximately minimizes $F_r$ using Adam initialized at $\mathbf{a}^{(r)}$ for a fixed number of steps (Appendix E). The final iterate $\boldsymbol{\alpha}_{\text{Adam}}^{(r)}$ is then clipped:

$$\mathbf{a}^{(r+1)} := \text{clip}_{[l,u]}\big(\boldsymbol{\alpha}_{\text{Adam}}^{(r)}\big).$$

Here $[l, u]$ bounds the total coefficients. Clipping enforces these bounds without solving the box-constrained optimization problem; it is inactive in the default correction runs (Appendix C). The anchor moves to $\theta(\mathbf{a}^{(r+1)})$, where responses are recomputed for the next round. We use two rounds throughout.

**A low-rank response reduces the $T^2G$ build cost.** A $k$-dimensional output projection replaces target–source JVPs with inner products against target-specific vector–Jacobian products (Eq. 11,

**Algorithm 1** LR-Merge

**Require:** $\{\tau_t\}_{t=1}^{T}$, $\{\theta_A\}_{A=1}^{T}$, $\{X_A\}_{A=1}^{T}$, $\theta_{\text{base}}$, $\mathbf{a}^{(0)}$, $[l, u]$, $\lambda$, $R$

$\widehat{y}_{A,i} \leftarrow \arg\max_c \; z_A(\theta_A)_{i,c} \quad \forall A, i$ ▷ teacher pseudo-labels, once

**for** $r = 0, \ldots, R-1$ **do**

  $\bar{\theta}^{(r)} \leftarrow \theta_{\text{base}} + \sum_t a_t^{(r)} \tau_t$ ▷ anchor

  $D_{A,t}^{(r)} \leftarrow J_A(\bar{\theta}^{(r)})\, \tau_t \quad \forall A, t$ ▷ responses, one JVP each

  $\widehat{z}_A^{(r)}(\boldsymbol{\alpha}) \leftarrow z_A(\bar{\theta}^{(r)}) + \sum_t \big(\alpha_t - a_t^{(r)}\big) D_{A,t}^{(r)}$ ▷ affine surrogate

  $\boldsymbol{\alpha}_{\text{Adam}}^{(r)} \leftarrow \textsc{Adam}\big[F_r(\boldsymbol{\alpha});\, \boldsymbol{\alpha} = \mathbf{a}^{(r)}\big]$ ▷ equation 3, unconstrained

  $\mathbf{a}^{(r+1)} \leftarrow \text{clip}_{[l,u]}\big(\boldsymbol{\alpha}_{\text{Adam}}^{(r)}\big)$

**end for**

**return** $\theta_{\text{base}} + \sum_t a_t^{(R)} \tau_t$

Appendix C). With $q$ power iterations and $p$ oversampling columns, construction uses $T(2q+2)(k+p)$ derivative passes. Source count and grouping still determine the number of inner products. The surrogate remains affine and the fixed-anchor objective convex. Rank 16 gives similar measured accuracy to the exact build at lower wall-clock (Appendix C).

## 6 Experiments

The setup is that of Section 3. An eight-task suite adds RESISC45 and SUN397, and all wall-clocks are measured on one GPU with identical probe budgets (Appendix E). Three questions organize the results: does the correction improve strong mergers and why, is it fast and probe-efficient, and does it scale, reuse, and transfer.

### 6.1 Does LR-Merge Improve Strong Mergers, and Why?

The local response is most reliable near an already-good merge, which is the intended use. Under one matched protocol, LR-Merge improves closed-form, isotropic, and sparsified bases alike, on every matched seed (Table 3). All four bases improve on the eight-task suite as well, over the same six checkpoint sets (Table 12), and the pattern repeats on the larger ViT-B/16 and ViT-L/14 backbones (Table 14, Appendix C). From scratch, LR-Merge provides a cheap alternative to optimization-based coefficient selection (Appendix C), but correction of a strong existing merge is its intended operating mode. The correction is also worth more as tasks accumulate: its gain over the base grows with the number of merged tasks (Appendix C).

Table 3: LR-Merge correction of six-task merges, six checkpoint sets, one protocol: every base improves on every matched seed. Cost excludes the base merger.

| base merger | base | +ours | Δ | cost (s) |
|---|---|---|---|---|
| RegMean | 87.49±0.22 | **90.58±0.20** | +3.10 ± 0.37 | 953 |
| Iso-C | 88.78±0.35 | **91.28±0.27** | +2.50 ± 0.31 | 952 |
| Iso-CTS | 88.15±0.38 | **91.49±0.23** | +3.34 ± 0.49 | 955 |
| TIES | 77.06±0.66 | **86.08±0.45** | +9.02 ± 0.34 | 973 |

Why it works is isolated by the wrong-anchor ablation, which moves exactly one thing: where the responses are read. Rebuilding the surrogate from pretrained responses, $D_{A,t} = J_A(\theta_0)\tau_t$, with objective, probes, pseudo-labels, solver, rounds, and wall-clock identical, the flagship correction (RegMean base) drops by roughly seven points and lands significantly *below* the uncorrected base it was meant to improve (per-seed values and paired tests in Appendix C). The wrong-anchor correction does not merely fail: it damages a working merge. Receiver-locality is part of the algorithm itself.

### 6.2 Correction Cost and Probe Budget

Figure 3A compares LR-Merge and layer-wise AdaMerging (Yang et al., 2024b) as corrections of the same RegMean base on identical unlabeled probe pools (Table 10, Appendix C).LR-Merge is

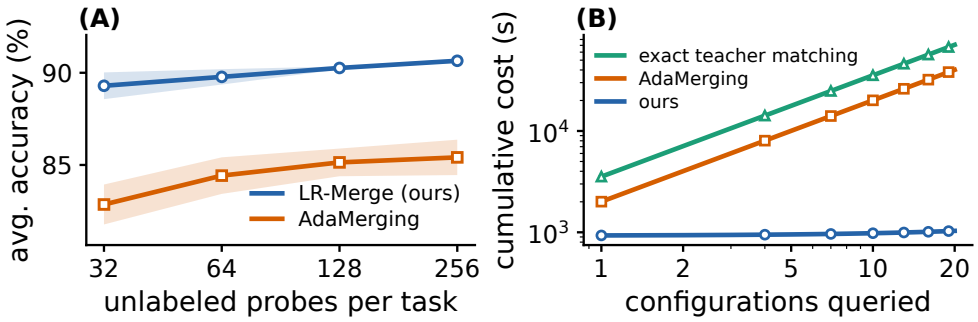


Figure 3: (A) Accuracy vs probe budget for LR-Merge and layer-wise AdaMerging, both correcting the same RegMean base using identical probe pools. (B) Cumulative cost of repeated coefficient queries (39–69× cheaper at twenty queries). Mean±sd over three checkpoint sets.

Table 4: A sixth task arrives at a deployed five-task corrected merge (three checkpoint sets): existing tasks are preserved incrementally, the arriving task pays for the stale anchor.

| | accuracy (6 tasks) | new task (CIFAR-10) | cost (s) |
|---|---|---|---|
| incremental (new JVPs + re-solve) | 89.54±0.34 | 83.7±2.4 | **146±1** |
| + surrogate re-anchor | 90.09±0.33 | 87.5±2.0 | 613 |
| full rebuild (base + correction) | 90.65±0.10 | 94.5 | 964 |
| AdaMerging from scratch | – | – | 2007 |

ahead at every budget, and the margin widens as the budget shrinks: at 32 probes per task it exceeds AdaMerging given 256 probes by 3.9 points at ∼26× less compute. A correction that returns in 130 s also supports iterative retuning (correct, inspect, adjust, re-correct) whereas the matched entropy descent takes roughly forty minutes per configuration. Reported times cover the correction stage on one GPU, excluding construction of the shared base merge.

The low-probe controls compare entropy minimization, exact teacher matching, and surrogate teacher matching around RegMean. Teacher matching reaches higher accuracy at 32 probes per task, while the surrogate reduces its computational cost (Appendix C). At 256 probes, exact matching and LR-Merge reach similar measured accuracies, including at matched computational budgets. The surrogate additionally supports reuse across coefficient queries (Figure 4, Appendix C).

### 6.3 Does It Scale, Reuse, and Transfer?

**Scale and reuse.** The rank-16 sketch reduces response-construction cost with similar measured accuracy, while caching amortizes construction across subsequent coefficient queries. Constructed once (926 s), the frozen responses answer each further coefficient query (a new trade-off, task subset, or clamp) in a 5.4 s convex solve. Twenty queries cost 1033 s cumulative against 40,133 s for per-configuration AdaMerging and 70,867 s for per-configuration exact matching (Figure 3B, Appendix C). For a growing model zoo, the frozen responses become a reusable per-anchor cache, and the marginal cost of considering a merge falls toward that of a convex solve.

**Task arrival.** Table 4 studies CIFAR-10 arriving at a deployed five-task corrected merge. The incremental update extends the response cache to include the arriving task and re-solves the coefficients. The five deployed tasks are preserved at a fraction of the rebuild cost, but the arriving task pays for the stale geometry, and re-anchoring recovers much of the gap. In practice, the anchor must follow the model it describes.

**Transfer.** Receiver-local forecasting is not specific to the primary backbone or to vision (Table 5): the receiver-vs-pretrained anchor gap holds on ViT-B/16 and ViT-L/14 and on LoRA language models (RoBERTa on GLUE, and Pythia at 410M and 1.4B, where forecasting targets teacher-forced logit displacement). The correction transfers as proof-of-concept: on Pythia-410M, LR-Merge

Table 5: Forecast drift cosine at the receiver vs at $\theta_0$ across backbones and model families (vision at $\alpha$=0.2, language in teacher-forced logit space).

| arm | cos @ receiver | cos @ $\theta_0$ |
|---|---|---|
| CLIP ViT-B/32 | 0.946 | 0.124 |
| CLIP ViT-B/16 | 0.966 | 0.152 |
| CLIP ViT-L/14 | 0.924 | 0.090 |
| RoBERTa-base/LoRA (3 GLUE tasks) | 0.77 | 0.11 |
| Pythia-410M/LoRA (5 tasks) | 0.92 | 0.23–0.68 |
| Pythia-1.4B/LoRA (2 pairs) | 0.97–0.98 | 0.53–0.54 |

matches exact teacher matching through the real merged network at a fraction of the wall-clock, on a fixed budget where the exact optimizer's best step varies by seed (Table 15, Appendix D).

## 7 Limitations and Future Work

LR-Merge assumes compatible models derived from a shared pretrained initialization and requires task-specific unlabeled probes and fine-tuned teachers. The anchor analysis is empirical, and the bounds in Appendix A depend on curvature constants that we do not estimate. Additional JVPs could support empirical error estimates and adaptive re-anchoring; certification would require curvature control between sampled points. Larger-backbone correction uses one checkpoint set per backbone. Language experiments use LoRA, with correction demonstrated on Pythia-410M under teacher-forced evaluation. Full-parameter merging of large decoders and generation-based evaluation remain open. Incremental reuse is demonstrated for one arriving task; broader arrival orders and longer sequences would clarify when cached responses require refreshing.

## 8 Conclusion

This work reframes merging as a problem that can be predicted before the merged model is built. Our main findings:

- **Merge interference is locally predictable.** A single directional derivative, read at the model receiving the update, forecasts how a task vector changes its outputs, per task and per example, to within $0.13$ accuracy points at $\alpha$=0.2.
- **The linearization point matters.** Fine-tuning rapidly reorients the cross-task response, which makes the pretrained anchor, the point task arithmetic uses implicitly, a poor predictor. Displacement along the receiver's task vector strongly orders forecast quality across the tested anchors.
- **These forecasts can be reused to improve merging.** LR-Merge freezes the local responses and turns coefficient selection into a cheap, convex, label-free fit against fine-tuned teachers. On three vision backbones and a language proof of concept it improves strong base mergers, tracking optimization through the full network at a fraction of its cost and under small probe budgets. Frozen responses are reusable, so retuning coefficients or task subsets costs seconds, and deployed tasks survive the arrival of new ones.

A task vector, in short, has no receiver-independent effect: watch where you linearize.

### Reproducibility statement

All experiments run from per-experiment scripts with fixed seeds, and tables are transcribed from stored result files. Code, per-experiment scripts, stored result files, and the fine-tuned checkpoint sets will be released upon acceptance. Section 3 and Appendix E specify models, data splits, probe/evaluation disjointness, fine-tuning and selection hyperparameters, and hardware, and Appendix C reports per-seed results for the headline comparisons.

### AI USE STATEMENT

In this work, we used generative AI tools as coding and writing assistants for generating and refactoring experimental and analysis code and for refining text. All AI-assisted code was reviewed and its outputs verified by the authors (re-running key configurations and validating tables and figures against the underlying result files), and all AI-assisted text was reviewed and edited by the authors. We did not use generative AI for research ideation. We used AI for verification of the proofs in Appendix A. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

## REFERENCES

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *ICML*, 2025.

Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *ICLR*, 2024.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *ECCV*, 2024.

Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the NTK overlap matrix. In *AISTATS*, 2021.

Guodong Du, Junlin Lee, Jing Li, et al. Parameter competition balancing for model merging. In *NeurIPS*, 2024.

Bethan Evans, Benjamin Etheridge, Stephen Roberts, and Jared Tanner. Model merging by output-space projection. *arXiv preprint arXiv:2605.29101*, 2026.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *COLT*, 2022.

Roger Grosse, Juhan Bae, Cem Anil, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Ido Nitzan Hidekel and Dan Raviv. Catastrophic forgetting is low-rank: A function-space theory for continual adaptation. In *ICML 2026 Workshop on Continual Adaptation at Scale: Towards Sustainable AI (CATS)*, 2026.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-merging: Tuning-free high-performance model merging. In *NeurIPS*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

Xisen Jin and Xiang Ren. What will my model forget? Forecasting forgotten examples in language model refinement. In *ICML*, 2024.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *ICLR*, 2023.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

Tian Yu Liu and Stefano Soatto. Tangent model composition for ensembling and continual fine-tuning. In *ICCV*, 2023.

Zehua Liu, Han Wu, Yuxuan Yao, Xiaojin Fu, Ruifeng She, Xiongwei Han, Tao Zhong, and Mingxuan Yuan. LoRE-merging: Exploring low-rank estimation for large language model merging. In *Findings of EMNLP*, 2025.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *ICML*, 2023.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In *ICML*, 2025.

Michael Matena and Colin Raffel. Merging models with Fisher-weighted averaging. In *NeurIPS*, 2022.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *NeurIPS*, 2023.

Biqing Qi, Fangyuan Li, Zhen Wang, Junqi Gao, Dong Li, Peng Ye, and Bowen Zhou. Less is more: Efficient model merging with binary task switch. In *CVPR*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter-efficient multi-task model fusion with partial linearization. In *ICLR*, 2024.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *ICML*, 2024.

Yongxian Wei, Anke Tang, Li Shen, Zixuan Hu, Chun Yuan, and Xiaochun Cao. Modeling multi-task model merging as adaptive projective gradient descent. In *ICML*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, et al. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022.

Feng Xiong, Runxi Cheng, Wang Chen, Zhanqiu Zhang, Yiwen Guo, Chun Yuan, and Ruifeng Xu. Multi-task model merging via adaptive weight disentanglement. *arXiv preprint arXiv:2411.18729*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *ICML*, 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024b.

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 *ICML*, 2024.

Shenghe Zheng and Hongzhi Wang. FREE-merging: Fourier transform for efficient model merging. In *ICCV*, 2025.

Luca Zhou, Bo Zhao, Rose Yu, and Emanuele Rodolà. Demystifying mergeability: Interpretable properties to predict model merging success. *arXiv preprint arXiv:2601.22285*, 2026.

Zhanpeng Zhou, Zijun Chen, Yilan Chen, Bo Zhang, and Junchi Yan. On the emergence of cross-task linearity in pretraining-finetuning paradigm. In *ICML*, 2024.

## A Bounds and Convexity

These results express the receiver-local approximation error through curvature along the update segment and how it propagates to coefficient selection, the curvature constants are problem-dependent and are not estimated here.

**Notation: everything happens along a line.** Every derivative in this appendix is an ordinary derivative of a function of one real variable. To forecast the update $v$, we only ever look at the outputs along the straight segment from the anchor to the candidate, so we give that restriction a name: for the update $v := \alpha\tau_B$ with length $r := \|v\|_2$ and unit direction $u := v/r$, let

$$h(s) := z_A(\theta_A + s\,u), \qquad s \in [0, r], \tag{4}$$

the outputs as the parameters travel from $\theta_A$ toward $\theta_A{+}v$ at unit speed. $h$ is a curve in output space. $h'(0) = J_A(\theta_A)u$ is (up to the factor $r$, since $r\,h'(0) = \alpha J_A(\theta_A)\tau_B$) the forecast of the main text, $h''(s)$ is the acceleration of the outputs at position $s$ along the segment, i.e. how fast the response itself is changing there, and $h'''$ is the next derivative. All norms $\|\cdot\|_2$ are Euclidean on the stacked outputs.

### A.1 Pairwise Taylor Remainder

The forecast replaces the curve by its tangent line at $s{=}0$. One-variable Taylor expansion of $h$ with integral remainder gives exactly the forecast error:

$$z_A(\theta_A + v) - z_A(\theta_A) - J_A(\theta_A)v = h(r) - h(0) - r\,h'(0) = \int_0^r (r - s)\,h''(s)\,\mathrm{d}s. \tag{5}$$

The error is the accumulated output acceleration along the segment. Bounding the integrand by the worst acceleration on the segment,

$$\kappa_A(\theta_A, v) := \sup_{s\in[0,r]} \|h''(s)\|_2\,, \tag{6}$$

the forecast error is second order in the merge radius (at most half the worst curvature times the squared distance moved):

$$\|z_A(\theta_A + v) - z_A(\theta_A) - J_A(\theta_A)v\|_2 \le \frac{1}{2}\kappa_A(\theta_A, v)\,r^2. \tag{7}$$

### A.2 Longer Merge Paths as Path Integration

A multi-task merge path is covered by re-reading the local response along it: with reference model $\vartheta$, candidate $\theta^\star$, $\Delta := \theta^\star - \vartheta$ and $\gamma(s) := \vartheta + s\Delta$,

$$z_A(\theta^\star) - z_A(\vartheta) = \int_0^1 J_A(\gamma(s))\Delta\,\mathrm{d}s, \tag{8}$$

so forecasting a merge is numerical integration of directional responses, each node one JVP. The interior nodes evaluate the Jacobian along the path, so they require the intermediate weights. The $K$-point composite midpoint forecast is

$$\widehat{\Delta z}_A^{(K)} := \frac{1}{K}\sum_{k=1}^{K} J_A\big(\gamma\big(\tfrac{2k-1}{2K}\big)\big)\,\Delta. \tag{9}$$

Write $\varphi(s) := z_A(\gamma(s))$ for the outputs along the path, a curve in the same sense as equation 4; the integrand of equation 8 is its first derivative $\varphi'(s) = J_A(\gamma(s))\Delta$. If $\varphi$ is three times continuously differentiable, the standard composite-midpoint error bound applies, with the integrand's second derivative being $\varphi'''$:

$$\left\| z_A(\theta^\star) - z_A(\vartheta) - \widehat{\Delta z}_A^{(K)} \right\|_2 \le \frac{1}{24K^2} \sup_{s\in[0,1]} \|\varphi'''(s)\|_2 = O\!\left(\frac{\|\Delta\|_2^3}{K^2}\right). \tag{10}$$

Empirically, on six-way merges at $\alpha{=}0.3$, going from one midpoint node to four raises the drift cosine from $0.878$ to $0.995$ and cuts per-task accuracy MAE from $3.97$ to $0.35$ pp over the six checkpoint sets: when the receiver changes substantially along a path, the local geometry must be updated, which is receiver-locality restated as a forecasting rule.

### A.3 Convexity of the LR-Merge Objective

For fixed target $A$ and probe $i$ the surrogate logits are affine in the total coefficients $\boldsymbol{\alpha}$: $\widehat{z}_A(\boldsymbol{\alpha})_{i,\cdot}^\top = b_{A,i} + M_{A,i}(\boldsymbol{\alpha} - \mathbf{a})$ with $b_{A,i} := z_A(\bar{\theta})_{i,\cdot}^\top$ and $(M_{A,i})_{\cdot,j} := (J_A(\bar{\theta})\, v_j)_{i,\cdot}^\top$, where $v_j$ is the $j$-th correction direction of the main text ($\tau_t$ task-wise, $\Pi_g \tau_t$ grouped), and the pseudo-label is fixed during optimization. Cross-entropy has logit Hessian $\mathrm{Diag}(p) - pp^\top \succeq 0$ ($p$ the softmax), so each per-example loss has Hessian $M_{A,i}^\top[\mathrm{Diag}(p_{A,i}) - p_{A,i}p_{A,i}^\top]M_{A,i} \succeq 0$; nonnegative sums preserve convexity, and the ridge adds $2\lambda I$, so the objective Hessian satisfies $\sum_{A,i} \frac{1}{Tn_A} M_{A,i}^\top[\mathrm{Diag}(p_{A,i}) - p_{A,i}p_{A,i}^\top]M_{A,i} + 2\lambda I \succeq 2\lambda I$. Hence the fixed-anchor objective is $2\lambda$-strongly convex on $\mathbb{R}^M$ for $\lambda > 0$ and has a unique unconstrained minimizer. The experiments optimize this objective with finite-step Adam and clamp the resulting total coefficients only after the solve. The clamp is not part of the convex optimization itself.

## B Receiver-Locality: Robustness

### B.1 Full Anchor Ladder

Table 6: Forecast quality under all tested anchors: 30 ordered pairs at $\alpha$=0.2, mean±sd over six checkpoint sets, min = worst pair. Last column: displacement along $\tau_A$ in units of $\|\tau_A\|$, i.e. $|\langle d, \tau_A\rangle|/\|\tau_A\|^2$, $d$ the anchor displacement from $\theta_A$.

| anchor | what it tests | mean cos | min cos | dist. from $\theta_A$ | disp. along $\tau_A$ |
|---|---|---|---|---|---|
| $\theta_A$ (receiving model) | the forecast as used | 0.949±0.008 | 0.296 | 0 | 0 |
| isotropic, matched distance | is distance the cause? | 0.945±0.009 | 0.315 | $\|\tau_A\|$ | 0.00 |
| isotropic, 1.41× distance | farther still random | 0.942±0.009 | – | $1.41\,\|\tau_A\|$ | 0.00 |
| span of other task vectors | is the direction structured? | 0.665±0.036 | −0.320 | $\|\tau_A\|$ | 0.07 |
| toward $\theta_C$, matched distance | third model, confound removed | 0.461±0.014 | −0.007 | $\|\tau_A\|$ | 0.67 |
| isotropic tilted 45° to $-\tau_A$ | direction dose–response | 0.352±0.018 | – | $\|\tau_A\|$ | 0.71 |
| $\theta_C$, a third fine-tuned model | does any fine-tuned point work? | 0.349±0.014 | −0.650 | $1.41\,\|\tau_A\|$ | 0.94 |
| $\theta_0$ (pretrained) | the natural alternative | 0.122±0.011 | −0.719 | $\|\tau_A\|$ | 1.00 |
| $\theta_0$ + isotropic kick | is the point $\theta_0$ special? | 0.116±0.022 | – | $1.41\,\|\tau_A\|$ | 1.00 |

Table 6 extends the ladder of Table 2. An isotropic anchor at distance $1.41\|\tau_A\|$ from $\theta_A$ retains high forecast quality ($0.942 \pm 0.009$). The tilted-anchor control (45° toward $-\tau_A$) scores $0.352 \pm 0.018$. Rescaling the third-model displacement $\theta_C - \theta_A$ to $\|\tau_A\|$ improves its forecast from $0.349 \pm 0.014$ to $0.461 \pm 0.014$, while reducing its along-$\tau_A$ displacement from 0.94 to 0.67. A full-size isotropic perturbation of $\theta_0$ also leaves the forecast poor ($0.116 \pm 0.022$), showing that the failure is not restricted to the exact pretrained anchor.

The last column quantifies the normalized magnitude of displacement along the receiver's task vector: $|\langle d, \tau_A\rangle|/\|\tau_A\|^2$, where $d = \theta_{\text{anchor}} - \theta_A$. Across the tested anchors, this coordinate is negatively correlated with forecast quality (Spearman $\rho = -0.98$, $p = 3 \times 10^{-5}$), whereas parameter distance has a much weaker association ($\rho = -0.16$, $p = 0.70$). Projection onto the full six-dimensional task-vector span is also insufficient to explain the ordering: it equals 1.0 for the span, $\theta_C$, and $\theta_0$ anchors despite their different forecast qualities. Nor does the receiver-task coordinate fully determine forecast quality: the span-of-others anchor scores 0.665 despite its small along-$\tau_A$ component (0.07). Together, these controls identify displacement direction as an important aspect of receiver-locality beyond parameter distance.

A class-split control using disjoint halves of CIFAR-100 and a from-scratch ResNet pilot further examine dependence on the task and training regime. Across the tested settings, the anchor effect is strongest under full fine-tuning of pretrained encoders.

**Forecast quality at the deployed anchors.** The ladder's anchors sit at $\|\tau_A\|$-scale displacements, while LR-Merge reads responses at a base merge $\bar{\theta}$ that is far larger in norm ($73 \pm 17$ times $\|\tau_A\|$ from each receiver, 89–108 in parameter norm) yet nearly orthogonal to the task-vector span (projection fraction 0.015, per-unit-distance projection on $\tau_A$ −0.009). The distance is a property of the RegMean base, not of the correction: 99.9% of $\bar{\theta}$'s displacement energy from $\theta_0$ sits in the first two transformer blocks, where the per-layer least-squares solve is ill-conditioned (attention input projections grow from $\max|w|$=0.22 pretrained to 4.79), and the base still merges well. Forecast quality

there is measured directly, not extrapolated from the ladder. Under the method's own convention (forecast and realized drift both around $\bar{\theta}$, mirroring Eq. 2), the drift cosine at the RegMean anchor is $0.858 \pm 0.006$ at $\delta$=0.2 and $0.909 \pm 0.017$ at the typical solved correction magnitude, against $0.949$ at $\theta_A$. The same responses transported back to the receiver's frame (the ladder's convention) score only $0.552 \pm 0.022$, but the method never asks for that transport. Re-anchoring leaves forecast fidelity unchanged ($-0.015 \pm 0.085$): its role is the Taylor remainder of the long first-round step, not a stale anchor.

### B.2 Reorientation Dynamics: Details

This section documents the measurement behind Figure 2 and the controls that pin down what rotates and when. The tracked pairs are ten ordered target–source pairs over receivers SVHN and GTSRB, at checkpoints $t \in \{1, 5, 10, 25, 50, 100, 250, 500, 1000, 2000\}$ of 2000 iterations, with receivers and sources independently re-fine-tuned per seed. The crossing is early and consistent: per-seed $45^\circ$ crossings on the primary receivers are $16.19/15.82/16.15$ (log-interpolated crossing $\approx 16$), per-curve crossings over all 30 pair×seed curves range 12.9–22.3 (median 16.0), and the plateau is reached at only $\approx 20\%$ of the final weight displacement. Until it saturates, the angle tracks weight displacement (Spearman $+0.73$ full-curve, $+0.94$ for $t \leq 50$, $\approx 0$ on the saturated tail). Norm-matched random Gaussian directions (five per receiver, two norm conventions, three seeds) rotate on nearly the same schedule as the task vectors (interpolated $45^\circ$ crossing $t \approx 17.6$ vs $\approx 17.9$). Early reorientation thus occurs for the sampled random directions as well as task vectors. What is pair-specific is the plateau dispersion: cross-task angles spread to sd $27^\circ$ at $t$=500 where random directions sit in a tight $\pm 9^\circ$ band (Levene $p<10^{-3}$, pooled over seeds and conventions). Two additional receivers replicate the onset: MNIST crosses $45^\circ$ at $t \approx 8.9$ and plateaus near $97^\circ$, while EuroSAT crosses at $t \approx 11.7$ and plateaus near $75^\circ$. The $\theta_0$-anchored forecast collapses on the same schedule for the new receivers ($0.83 \rightarrow 0.07$) while the current-anchor forecast reaches $0.975$. A segment sweep separates training time from position: the cosine of $J_A(\theta_0 + s\tau_A)\tau_B$ with the endpoint response rises smoothly with $s$, so the checkpoint result is not mere progress along the straight segment.

The reorientation is directional rather than a change in magnitude or output support: over the 30 pairs the norm ratio between the initialization and receiver responses has median $0.97$, both keep most of their energy in the same top-32 output subspace ($72\%$ and $64\%$), and their mutual signed cosine is only $0.11$. Fine-tuning re-aims the response rather than attenuating it.

### B.3 Replication over Fine-Tuning Seeds

Table 7: The forecast and the anchor gap over six independently fine-tuned checkpoint sets, 30 ordered pairs each at $\alpha$=0.2: the fine-tuning run itself is resampled.

| checkpoint set | cos @ $\theta_A$ | cos @ $\theta_0$ | damage $\rho$ | acc. MAE (pp) |
|---|---|---|---|---|
| 0 | 0.946 | 0.124 | 0.916 | 0.13 |
| 1 | 0.954 | 0.126 | 0.925 | 0.17 |
| 2 | 0.958 | 0.107 | 0.927 | 0.16 |
| 3 | 0.946 | 0.109 | 0.916 | 0.09 |
| 4 | 0.953 | 0.132 | 0.930 | 0.13 |
| 5 | 0.936 | 0.133 | 0.916 | 0.10 |
| mean ± sd | **0.949±0.008** | **0.122±0.011** | 0.922±0.006 | 0.13±0.03 |

Table 7 resamples the fine-tuning runs: the receiving-anchor forecast and the pretrained-anchor failure replicate on every set.

## C LR-Merge Controls

### C.1 Wrong Anchor, Per Seed

This section gives the per-seed evidence behind Section 6.1. The ablation swaps exactly one thing in the correction pipeline: the anchor at which the responses are read, from the current merge to

Table 8: Where the surrogate's responses are evaluated, everything else held fixed: six-task standalone selection, six independently fine-tuned checkpoint sets.

| surrogate built at | s0 | s1 | s2 | s3 | s4 | s5 | mean ± sd |
|---|---|---|---|---|---|---|---|
| receiving model $\theta_A$ | 80.65 | 80.18 | 80.72 | 79.98 | 80.83 | 80.00 | **80.39±0.38** |
| pretrained $\theta_0$ | 73.77 | 73.83 | 73.80 | 74.00 | 74.65 | 73.65 | 73.95±0.36 |
| paired drop | 6.88 | 6.35 | 6.92 | 5.98 | 6.18 | 6.35 | **6.44±0.38** |

$\theta_0$, with the RegMean base, objective, probes, pseudo-labels, solver, and rounds unchanged. The receiver-anchored arm reproduces the stored flagship run bit-for-bit, so the two arms differ in nothing but the anchor. Per-seed corrected accuracy falls from $90.53/90.68/90.73/90.23/90.80/90.52$ to $84.12/83.67/84.07/84.45/82.90/83.03$, a paired drop of $6.88 \pm 0.76$ ($t(5)$=22.2), and the damage concentrates on the tasks whose vectors move farthest from pretrained (SVHN $-24.5$, GTSRB $-7.9$). Table 8 repeats the swap in the standalone setting, where the surrogate carries the whole solution rather than a correction. The same one-line change costs $6.44 \pm 0.38$ points there, positive on every checkpoint set and statistically indistinguishable in size from the correction-pipeline drop (Welch $p$=0.25). Solved total coefficients lie strictly inside the clamp box in both arms and wall-clock is identical, so the correct anchor is purely a modeling choice, not an extra cost.

The clamp itself is inert in the deployed configuration. Across every stored correction run (six checkpoint sets, both rounds), no solved coordinate reaches the $[-1, 1.5]$ box, with slack at least $0.48$, and re-solving the flagship correction under boxes from $[-0.5, 1.0]$ to $[-1, 3]$ on one response build per set changes no coefficient and no accuracy: any box wide enough to contain the unconstrained solution gives the identical result. A box that does clip changes the method: a sign-constrained $[0, 1.5]$ box, applied by clipping the unconstrained solution, zeroes $46\%$ of coordinates and costs $0.85 \pm 0.37$ points (paired $p$=0.0025): the correction is genuinely subtractive in nearly half its coordinates, and removing that subtractive part is harmful.

### C.2 Exact Teacher Matching at Matched Budgets

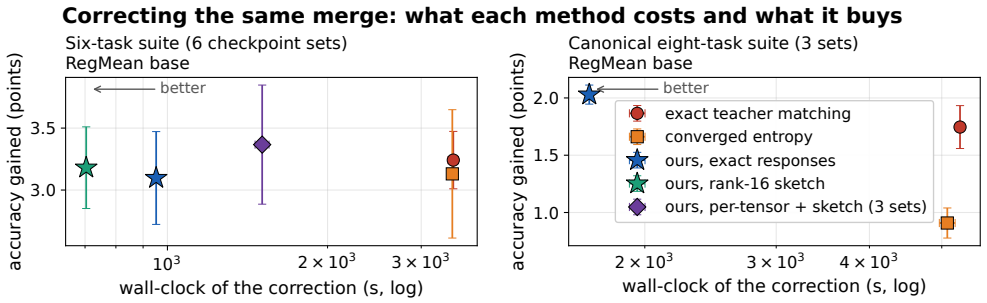


Figure 4: What correcting the same merge costs and buys: six-task suite (left) and canonical eight-task benchmark (right), log-scale wall-clock, bars are sds. The surrogate arms reach the exact optimizers' accuracy at a fraction of their cost.

Table 9: Four ways to correct the same RegMean base, six checkpoint sets, each arm timed alone on one GPU. Build is the response construction inside the total.

| correction of the same RegMean base | Δ accuracy | paired vs. ours | wall-clock (s) | of which build |
|---|---|---|---|---|
| exact teacher matching | +3.24±0.23 | +0.14 ± 0.24 | 3447±78 | – |
| converged entropy polish | +3.13±0.52 | +0.03 ± 0.22 | 3432±32 | – |
| LR-Merge, exact responses | +3.10±0.37 | – | 952±15 | 936 |
| **LR-Merge, rank-16 sketch** | +3.18±0.33 | +0.08 ± 0.12 | **704±6** | 686 |

Table 9 and Figure 4 compare four ways of correcting the same RegMean base, and answer the natural question of whether the surrogate buys accuracy or only cost. The exact teacher-matching runs record their convergence curves, which allow a budget-matched reading. On the six-task suite, exact optimization stopped at step 500 of 2000 ($\approx 0.9 \times 10^3$ s, at or below the correction's 952 s)

gains $+3.18 \pm 0.30$ over the base against the correction's $+3.10 \pm 0.37$ and the converged run's $+3.24 \pm 0.23$. The paired difference, exact minus the correction, is $+0.08 \pm 0.14$ ($p$=0.19). On the eight-task suite: base $83.10 \pm 0.25$; exact teacher matching at 2000 steps $86.32 \pm 0.12$ at 5404 s vs the correction's $86.19 \pm 0.20$ at 1685 s (paired $+0.13 \pm 0.10$, $p$=0.023, at $3.2\times$ the cost), and at the correction's budget $86.08 \pm 0.15$ (paired $-0.12 \pm 0.09$, $p$=0.027). Matched-budget exact optimization is parity. The surrogate's advantage is cost and amortization.

A gradient-free control bounds what the budget alone buys. Random search over the coefficient box at LR-Merge's full matched wall-clock (about 584 directly materialized and forward-evaluated candidates, three checkpoint sets) stays 2.6 points below the correction in the plain six-coefficient space and 5.1 points below in the grouped 78-coefficient space, and sampling the full clamp box falls below the uncorrected base; across all arms, only 1 of $\sim$1750 sampled candidates improves the teacher-matching objective over leaving the base unchanged. Our solve restricted to the same six-dimensional space reaches $89.49 \pm 0.12$ in 88 s against random search's $88.05 \pm 0.33$ in 969 s. What the surrogate contributes is not cheap scoring of arbitrary candidates but cheap *gradients*: blind sampling cannot locate good coefficients even at matched cost.

### C.3 Low-Probe Controls

Table 10: LR-Merge and layer-wise AdaMerging correcting the same RegMean base on matched unlabeled-probe pools, checkpoint sets 0–2: per-budget results behind Figure 3A.

| probes/task | ours | AdaMerging | gap | ours / Ada wall-clock (s) |
|---|---|---|---|---|
| 32 | **89.29±0.73** | 82.86±1.08 | +6.44 | 130 / 2378 |
| 64 | **89.78±0.42** | 84.42±0.99 | +5.36 | 246 / 3416 |
| 128 | **90.26±0.08** | 85.14±0.75 | +5.12 | 480 / 3435 |
| 256 | **90.65±0.10** | 85.41±0.96 | +5.24 | 943 / 3437 |

Table 10 lists the per-budget values behind Figure 3A. Two controls check that the low-probe margin is not an artifact of AdaMerging's fixed step budget. First, truncating AdaMerging does not produce a cheap competitive point at the 32-probe budget: over three sets its accuracy at roughly our wall-clock ($\sim$125 s) is $80.66 \pm 0.34$, and even an oracle stop chosen on labeled evaluation data reaches only $83.43 \pm 0.56$ at 580–2340 s (5.9 points below us at 4–18$\times$ our cost). Second, exact teacher matching through the real network on the same 32 probes does converge ($89.08 \pm 0.11$ at 3428 s; peak $89.71 \pm 0.23$ at $\sim$858 s): the low-probe margin over AdaMerging is the teacher-matching objective, and the surrogate's contribution at this budget is cost (130–252 s, i.e. 3–7$\times$ cheaper than the exact objective's useful points), plus a fixed optimization schedule with no label-based early stopping, where the exact curve decays 0.6 points past its peak and its best step is not identifiable label-free. On DTD, with 32 unlabeled examples across 47 classes, entropy-based selection reaches 47.8% accuracy versus 66.0% for teacher matching.

### C.4 Standalone Selection

Table 11: Standalone selection on the six-task suite: accuracy and selection cost. LR-Merge matches the optimization-based methods at a fraction of their cost. The closed-form Iso-C remains stronger standalone, motivating correction as the intended use.

| method (six-task, from scratch) | avg | sel. cost (s) |
|---|---|---|
| Iso-C (closed-form) | **88.9±0.3** | 54 |
| AdaMerging (task-wise) | 80.1±0.5 | 1185 |
| AdaMerging (layer-wise) | 85.2±0.8 | 2007 |
| Exact teacher matching (layer-wise) | 85.6 | 2160 |
| LR-Merge (task-wise) | 80.4±0.4 | **84** |
| LR-Merge (layer-wise) | 85.4±0.5 | 933 |

Run from scratch rather than as a correction, LR-Merge is a drop-in alternative to optimization-based coefficient selection at a fraction of its cost (Table 11). Against AdaMerging, the same search over the same task vectors and probes, optimized through the exact merged network, LR-Merge is statistically tied on the six-task suite (85.4 vs 85.2 layer-wise at 933 vs 2007 s; 80.4 vs 80.1 task-wise at 84

vs 1185 s), and ahead on the eight-task suite at both granularities on all six checkpoint sets ($+1.60$, $p$=0.0010 task-wise; $+1.97$, $p$=0.0003 layer-wise; $+0.82$ against AdaMerging's best checkpoint). Exact teacher matching through the real network reaches 85.6 at 2160 s, and the surrogate matches it at less than half the cost. Closed-form Iso-C remains the strongest standalone six-task merger, which is why correction of a strong base, not from-scratch selection, is the intended use, but where coefficients must be optimized, the frozen surrogate reaches similar measured accuracy at lower cost.

### C.5 Eight-Task Correction and Reuse

Table 12: Eight-task correction over six checkpoint sets per row.

| base merger (8-task) | base | +ours | $\Delta$ | sets | cost (s) |
|---|---|---|---|---|---|
| RegMean | 83.10±0.25 | **86.19±0.20** | $+3.09 \pm 0.34$ | 6 | 1685 |
| Iso-C | 85.92±0.22 | **88.28±0.19** | $+2.36 \pm 0.15$ | 6 | 1700 |
| Iso-CTS | 86.18±0.46 | **88.66±0.22** | $+2.47 \pm 0.52$ | 6 | 1688 |
| TIES | 73.62±0.44 | **80.62±0.65** | $+7.00 \pm 0.31$ | 6 | 1693 |

Table 12 gives the per-base eight-task corrections summarized in Section 6.1. The reuse measurements of Section 6.3 come from three checkpoint sets: a two-round build of $925.9 \pm 1.6$ s, then 20 distinct coefficient queries per set (a ridge/clamp grid and task-subset restrictions) at $5.37 \pm 0.19$ s each, with validation queries reproducing the stored flagship accuracies exactly.

Table 13: Average accuracy vs number of merged tasks $T$ (nested task ordering over the eight-task suite, mean±sd over three checkpoint sets; task arithmetic at $\alpha$=0.3; $p$ from a paired test of the correction against its base).

| $T$ | task arithmetic | RegMean base | RegMean + ours | gain | $p$ |
|---|---|---|---|---|---|
| 2 | 90.50±0.39 | 98.23±0.21 | 98.28±0.15 | $+0.05 \pm 0.17$ | 0.67 |
| 3 | 86.06±0.48 | 97.09±0.16 | 97.41±0.19 | $+0.32 \pm 0.24$ | 0.15 |
| 4 | 81.73±0.25 | 96.18±0.21 | 97.03±0.21 | $+0.85 \pm 0.35$ | 0.052 |
| 5 | 75.23±0.12 | 88.82±0.20 | 91.05±0.15 | $+2.23 \pm 0.34$ | 0.0077 |
| 6 | 75.80±0.15 | 87.37±0.25 | 90.65±0.10 | $+3.28 \pm 0.35$ | 0.0037 |
| 7 | 75.77±0.29 | 86.53±0.10 | 89.98±0.15 | $+3.45 \pm 0.25$ | 0.0018 |
| 8 | 73.28±0.45 | 82.98±0.19 | 86.29±0.08 | $+3.31 \pm 0.23$ | 0.0015 |

The correction's value grows with the number of merged tasks (Table 13). Over a nested task ordering, base and corrected curves diverge as $T$ grows: the gain rises monotonically (Spearman $\rho$=0.92 across all $(T, \text{set})$ points, $p$=$3.5 \times 10^{-9}$), is significant at every $T \geq 5$, and plateaus at $T \geq 6$. At $T$=2 the base sits 0.35 points below the fine-tuned ceiling, so there is little damage to repair. Interference, and with it the correction's headroom, accumulates with task count. The response build follows its predicted $T^2G$ pass count to within 8% over a 16× range (113 s at $T$=2 to 1666 s at $T$=8), while solve cost is independent of $T$. Rebuilt bases at $T \in \{6, 8\}$ are bitwise identical to the stored campaign artifacts, and on the shared checkpoint sets the corrected accuracies reproduce them to the reported digit.

### C.6 Larger Backbones

The correction was also run once on ViT-B/16 and once on ViT-L/14 (Table 14). Every base improves on both backbones, by $+0.82$ to $+2.60$ points on ViT-B/16 and $+0.48$ to $+1.35$ on ViT-L/14, in the same direction as the six-set ViT-B/32 result of Table 3. The cost column shows the response build scaling with model size, from ∼3.6 ks on ViT-B/16 to ∼29 ks on ViT-L/14.

### C.7 Low-Rank Identity

The response sketch rests on the identity

$$U_k^\top D_{A,t,g} = U_k^\top J_A(\bar{\theta})\,[\Pi_g \tau_t] = \big(J_A(\bar{\theta})^\top U_k\big)^\top [\Pi_g \tau_t], \tag{11}$$

Table 14: Six-task correction on the larger backbones. **One checkpoint set per backbone**: each accuracy is a mean over the six tasks of that single set, so unlike every other correction table in this paper these rows carry no across-set spread and no paired test.

| backbone | base merger | base | +ours | Δ | cost (s) |
|---|---|---|---|---|---|
| ViT-B/16 | RegMean | 89.27 | **91.87** | +2.60 | 3633 |
| | Iso-C | 92.18 | **93.00** | +0.82 | 3634 |
| | Iso-CTS | 92.37 | **93.35** | +0.98 | 3632 |
| ViT-L/14 | RegMean | 92.83 | **94.18** | +1.35 | 29099 |
| | Iso-C | 93.53 | **94.65** | +1.12 | 29200 |
| | Iso-CTS | 94.57 | **95.05** | +0.48 | 29128 |

where $J_A(\bar{\theta})^\top U_k$ is $k$ vector–Jacobian products depending on the target alone. Oversampling and power iteration make the implemented cost $T(2q+2)(k+p)$ passes. At rank 16, the sketch gives similar measured accuracy at lower construction cost: standalone selection reaches $85.60 \pm 0.40$ vs $85.65 \pm 0.46$ for the exact build, and the rank-16 correction row of Table 9 gains $+3.18 \pm 0.33$ at 704 s against the exact construction's $+3.10 \pm 0.37$ at 952 s.

## D Language-Model Arm

RoBERTa-base with rank-8 LoRA on three GLUE tasks, and Pythia-410M/1.4B with rank-16 LoRA. Forecasting uses held-out teacher-forced token blocks, and the target is the full token-by-vocabulary logit displacement, so every claim in this arm is a logit-space claim. The anchor gap transfers and narrows with scale (Table 5), consistent with kernel-regime descriptions of fine-tuning (Malladi et al., 2023).

Table 15: Pythia-410M/LoRA cost head-to-head, mean±sd over three probe/optimizer draws (one LoRA fine-tune, teacher-forced held-out metrics).

| arm (Pythia-410M/LoRA, five tasks) | held-out NLL | top-1 token acc. (%) | wall (s) |
|---|---|---|---|
| base merge (TA $\alpha$=0.3) | 2.1154±0.0208 | 57.10±0.49 | – |
| **LR-Merge (ours)** | 2.0895±0.0225 | 57.83±0.51 | **78** |
| exact TM at ours' budget | 2.0935±0.0213 | 57.89±0.52 | ≤78 |
| exact TM, oracle best step | 2.0924±0.0222 | 57.88±0.53 | – |
| exact TM, final (1000 steps) | 2.1059±0.0228 | 57.85±0.48 | 1227 |

The correction head-to-head of Table 15 starts both methods from the same task-arithmetic base merge, with the same teacher-argmax objective and GPU. The surrogate solves in 78 s. Exact teacher matching through the real merged network needs 1227 s for 1000 steps; its budget-matched and oracle-stopped points sit within 0.004 nats of the surrogate, with no significant token-accuracy difference. We claim parity, not superiority. The oracle stopping step differs across seeds $(500/50/100)$, so exact optimization's best case is not reachable label-free, while the surrogate uses a fixed optimization budget and does not require selecting an oracle stopping point. Both methods beat the base clearly ($p$=0.006 NLL, $p$=0.0015 token accuracy).

## E Reproducibility

**Tasks and splits.** The primary six-task vision suite is MNIST, SVHN, GTSRB (43 classes), EuroSAT, DTD (47 classes), and CIFAR-10. The eight-task suite adds RESISC45 and SUN397. Each task uses 256 unlabeled probe examples for JVP computation, forecasting, and coefficient selection, and an index-disjoint 1000-example split for evaluation. Probe seeds vary independently of fine-tuning seeds. Reported standard deviations over six checkpoint sets are sample deviations; the three-set low-probe and gradient-free controls report population deviations.

**Fine-tuning.** Vision fine-tuning follows the standard task-vector protocol: AdamW with learning rate $10^{-5}$, weight decay 0.1, batch size 128, 2000 iterations, cosine learning-rate decay, and frozen zero-shot text heads. The primary experiments use the non-QuickGELU OpenAI-weight variant

exposed by `open_clip`. All methods, forecasts, and checkpoints in a comparison use the same backbone.

**JVP implementation.** JVPs are computed in evaluation mode by forward-mode automatic differentiation. No full Jacobians or per-example gradients are formed. For CLIP, the math scaled-dot-product-attention backend is used because the fused backend does not support the required forward-mode derivatives. Task-wise LR-Merge needs $T^2$ target–source JVPs per anchor, and grouped LR-Merge needs $T^2G$ with $G{=}13$ groups for ViT-B/32 (one per transformer block plus a combined embedding/projection group).

**LR-Merge hyperparameters.** The optimization variable is the total coefficient vector $\boldsymbol{\alpha}$ of equation 3, initialized at the current anchor coefficients $\mathbf{a}^{(r)}$: $\mathbf{a}^{(0)} = 0.3$ uniform for standalone selection, and $\mathbf{a}^{(0)} = 0$ for the correction runs, whose initial anchor is the base merge itself. The ridge on the correction $\boldsymbol{\alpha} - \mathbf{a}^{(r)}$ uses $\lambda = 0.01$. Each fixed-anchor subproblem is optimized without box constraints using Adam (learning rate $0.01$), for $800$ steps task-wise, $1500$ steps for grouped standalone selection, and $800$ steps for the grouped correction runs. After the final optimizer step, the total coefficients are clamped coordinate-wise to $[0, 1.5]$ for selection or $[-1, 1.5]$ for the reported correction runs. Two rounds are used throughout: in the second correction round, the clamped coefficients from round one define the new coefficient anchor and the responses are recomputed there. Reported selection cost covers response construction in both rounds, the solver, and re-anchoring. Teacher pseudo-labels are computed once beforehand.

**Baselines and hardware.** AdaMerging runs entropy descent (Adam, learning rate $10^{-3}$, batch $64$) on the same unlabeled probe pool as LR-Merge. In the probe-budget comparison (Figure 3A and Table 10), both LR-Merge and layer-wise AdaMerging optimize corrections to the same RegMean base using identical unlabeled probe pools. All wall-clock comparisons are run on one RTX 8000 GPU with identical probe budgets. Reported costs exclude single-task fine-tuning, which is shared by every merging method.