Merging RLVR-Trained Experts via Policy-Shift-Guided Spectral Alignment

Geeho Kim[1*] **Minsik Choi**[2,3*] **Kyle Min**[4] **Young Geun Kim**[3†] **Bohyung Han**[1†]

[1]Seoul National University [2]FuriosaAI [3]Korea University [4]Oracle
{snow1234, bhhan}@snu.ac.kr {mszzang2002, younggeun_kim}@korea.ac.kr

Abstract

We propose Policy-Shift-Guided Spectral Alignment (PSA), a novel model-merging framework for experts independently post-trained using reinforcement learning with verifiable rewards (RLVR). It consolidates these experts into a single model without joint retraining. We first show that existing merging methods fail to preserve RLVR-acquired behaviors by attenuating expert–base log-probability shifts at token positions where an expert diverges from the base model. To address this limitation, PSA formulates a low-rank projection objective that preserves layer outputs along rollout trajectories, identifying expert subspaces active at reasoning-pivotal tokens. Specifically, it constructs a policy-shift-weighted Gram matrix using token-level log-probability differences and uses this matrix to guide the low-rank truncation of each expert update. PSA also polar-orthogonalizes the truncated bases across experts to reduce cross-expert conflict and rescales each retained update to the Frobenius norm of its original task vector before aggregation. Across five Qwen2.5-7B and three Qwen3-1.7B RLVR experts, PSA outperforms existing merging techniques in macro-averaged performance across benchmarks, while retaining expert-induced token-level policy shifts.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) is increasingly producing task-specialized language models for reasoning, coding, tool use, and long-horizon interaction [Shao et al., 2024, Wang et al., 2025b, Qian et al., 2025, Qin et al., 2025, Yu et al., 2025a]. However, serving separate checkpoints incurs high memory and routing overheads while preventing cross-expert capability fusion. Joint multi-task RLVR requires full access to task environments and demands costly retraining as expert pools evolve. Model merging provides a practical, retraining-free alternative by consolidating same-base RLVR experts into a single model without access to their original reward environments or additional gradient updates.

Most existing merging methods rely on task-vector statistics to determine how expert updates should be combined. These approaches generally fall into two categories. Coordinate-wise methods operate on individual matrix entries using linear addition [Ilharco et al., 2023], sign-conflict resolution [Yadav et al., 2023], stochastic sparsification [Yu et al., 2024], or magnitude-based partitioning and rescaling [Yuan et al., 2026]. Because these methods process individual scalar parameters independently, they ignore layer-wise update geometry. To address this limitation, spectral methods operate directly on the singular subspaces of layer-wise task vectors. TSV [Gargiulo et al., 2025] truncates task vectors and polar-aligns retained bases, STAR [Lee et al., 2025] rescales top singular components, and Iso-CTS [Marczak et al., 2025] separates common from task-specific subspaces. Both

*Equal Contribution
†Corresponding Authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

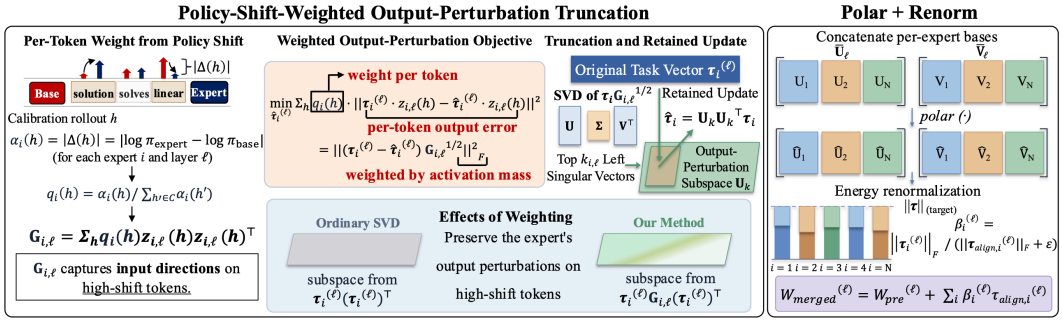


Figure 1: **PSA pipeline.** PSA scores tokens by the magnitude of the expert–base log-probability difference $|\Delta_i(h)|$ under the same prefix. It uses these weights to form a policy-shift-weighted activation Gram matrix $\mathbf{G}_{i,\ell}$ that captures input directions at high-shift tokens. The low-rank truncation selects a rank-$k_{i,\ell}$ subspace $\mathbf{U}_{i,\ell,k}$ that preserves the expert's output perturbations on those tokens. The retained bases are polar-aligned across experts, and $\beta_i^{(\ell)}$ restores each expert's Frobenius scale.

paradigms assume that static parameter statistics, such as weight magnitudes or singular values, directly dictate an update's behavioral impact.

However, this assumption does not hold for RLVR updates. While supervised fine-tuning aligns updates with dominant singular directions, RLVR updates lie outside these directions, within low-curvature, spectrum-preserving subspaces [Zhu et al., 2025]. Recent work indicates that RLVR updates induce log-probability shifts localized to a sparse set of reasoning-pivotal tokens on identical prefixes [Huang et al., 2026]. Evaluating these shifts across rollouts generated by the base model, we find that existing merging methods fail to preserve the expert–base log-probability differences at pivotal reasoning tokens. The merged model's policy collapses back to that of the base model.

To preserve these token-level policy shifts, we propose Policy-Shift-Guided Spectral Alignment (PSA), a calibration-based merging framework. PSA shifts the low-rank approximation target from unweighted parameter energy to policy-shift-weighted layer-output reconstruction. As shown in Figure 1, PSA first generates calibration rollouts using the shared base model and computes expert–base log-probability differences under the same prefix. The magnitude of each difference serves as a token-level weight, quantifying how strongly the expert policy deviates from the base policy at each trajectory step. For each layer, PSA aggregates the token activations using these weights to form a policy-shift-weighted Gram matrix. This matrix guides the truncation of each expert update to retain the directions that best reproduce layer-output perturbations at pivotal tokens. Finally, PSA polar-aligns the low-rank bases across experts to mitigate inter-expert interference and rescales each truncated update to the Frobenius norm of its original task vector to compensate for truncation-induced norm shrinkage. Extensive evaluations across five Qwen2.5-7B and three Qwen3-1.7B RLVR experts demonstrate that PSA achieves higher overall task-average performance than representative parameter-level, spectral, and RLVR-specific merging baselines.

Our main contributions are:

- We show that existing model-merging methods fail to preserve expert–base log-probability shifts at token positions where an expert diverges from the shared base, shifting the merged model's behavior back toward that of the base model.
- We propose a novel merging framework for RLVR post-trained experts that formulates low-rank truncation as behavior-weighted layer-output reconstruction. We derive an exact full-Gram solution alongside a scalable diagonal variant and integrate both with cross-expert polar alignment and norm rescaling.
- We evaluate PSA across five Qwen2.5-7B and three Qwen3-1.7B RLVR post-trained experts, demonstrating that it achieves the highest overall task-average performance among the compared merging methods while retaining token-level expert policy shifts.

2 Preliminaries

RLVR post-training We address the problem of merging experts obtained by RLVR post-training from the same pretrained model. RLVR incentivizes reasoning capabilities by optimizing for verifiable outcome rewards—such as answer correctness or execution success—over generated rollouts [Shao et al., 2024, Yu et al., 2025b]. Recent work [Huang et al., 2026, Zhu et al., 2025] shows that this outcome-driven objective makes RLVR updates sparse and directional at the token level. These studies measure this directional behavior by comparing the log-probabilities assigned by the expert and pretrained models to the same sampled token under the same prefix. Let $\mathcal{C}$ denote the sampled token instances in rollouts. For each token instance indexed by $h \in \mathcal{C}$, let v_h be the sampled token and $v_{<h}$ its causal prefix. For expert i, the signed log-probability difference between the expert and pretrained models is

$$\Delta_i(v_h \mid v_{<h}) \;\coloneqq\; \log \pi_{\boldsymbol{\theta}_i}(v_h \mid v_{<h}) - \log \pi_{\boldsymbol{\theta}_{\text{pre}}}(v_h \mid v_{<h}). \tag{1}$$

For brevity, we write $\Delta_i(h) \coloneqq \Delta_i(v_h \mid v_{<h})$ for $h \in \mathcal{C}$. The magnitude of this difference quantifies the intensity of the policy shift, indicating how strongly the expert deviates from the pretrained model at that specific token. Huang et al. [2026] show that log-probability differences reveal the direction of RLVR's token-level policy shifts, whereas entropy and KL divergence mainly measure their magnitude. Their selective token-replacement experiment further shows that sampled token instances with large $|\Delta_i(h)|$ mark cases where RLVR strongly changes the base policy. In this work, we use the linear-map input states at these sampled token instances to guide the merge toward preserving task-specific RLVR behavior.

Model merging When merging multiple experts into a single generalist model, the standard paradigm is to linearly combine their parameter-space updates, known as task vectors. Let $\boldsymbol{\theta}_{\text{pre}}$ be a pretrained model and let $\{\boldsymbol{\theta}_i\}_{i=1}^N$ be experts obtained by RLVR post-training from it. For a matrix-valued layer ℓ, let $\mathbf{W}_{\text{pre}}^{(\ell)}$ and $\mathbf{W}_i^{(\ell)}$ denote the pretrained and expert weights. We define the layer-wise task vector of expert i as

$$\boldsymbol{\tau}_i^{(\ell)} \;\coloneqq\; \mathbf{W}_i^{(\ell)} - \mathbf{W}_{\text{pre}}^{(\ell)} \in \mathbb{R}^{R \times C}, \tag{2}$$

where the goal is to synthesize a merged update $\boldsymbol{\tau}_{\text{merged}}^{(\ell)}$ such that $\mathbf{W}_{\text{merged}}^{(\ell)} = \mathbf{W}_{\text{pre}}^{(\ell)} + \boldsymbol{\tau}_{\text{merged}}^{(\ell)}$. Beyond direct parameter combination, spectral methods apply singular value decomposition (SVD) to each $\boldsymbol{\tau}_i^{(\ell)}$ and retain only its top-k singular directions before aggregating across experts [Gargiulo et al., 2025, Lee et al., 2025, Marczak et al., 2025]. We refer to this top-k retention step as low-rank truncation, which suppresses small components and reduces inter-expert directional interference at merge time. To connect the task vectors $\boldsymbol{\tau}_i^{(\ell)}$ to token-level behavior, we view each as a linear map from layer inputs to layer-output perturbations. For a layer input state $\boldsymbol{z} \in \mathbb{R}^C$, the expert update changes the layer output by $\boldsymbol{\tau}_i^{(\ell)}\boldsymbol{z} \in \mathbb{R}^R$. This input-output view allows merging to preserve the perturbations that the expert induces on token prefixes where the log-probability difference in Eq. (1) is large.

3 Policy-Shift-Guided Spectral Alignment

This section begins by showing that existing merging methods often fail to preserve the sparse, token-level policy shifts that characterize RLVR experts (Section 3.1). To preserve these shifts, we introduce Policy-Shift-Guided Spectral Alignment (PSA), a three-stage merging method that converts these token-level policy-shift signals into a layer-wise spectral objective. First (Section 3.2), PSA scores calibration rollout tokens by the magnitude of expert-vs-base policy shifts and uses the resulting weighted linear-map input states to define a layer-wise activation Gram matrix. Second (also Section 3.2), it solves the resulting weighted low-rank approximation problem so that truncation preserves the subspace spanned by output perturbations at high-shift tokens. Third (Section 3.3), it applies cross-expert polar alignment to orthogonalize the retained bases and resolve directional conflicts among the independently truncated experts. It then applies energy renormalization to restore each expert's scale before final aggregation. This weighted truncation criterion contrasts with ordinary SVD, which retains directions based only on their unweighted parameter energy. Algorithm 1 gives the full procedure.

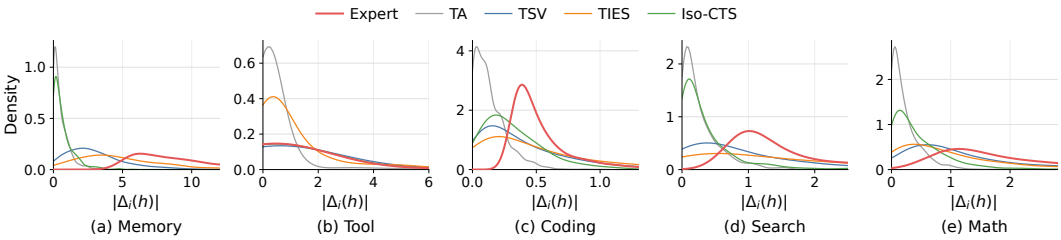


Figure 2: **Existing merging methods fail to preserve experts' token-level policy shifts.** Kernel density estimates of $|\Delta_i(h)|$ over the top 5% of token positions ranked by the expert-vs-base absolute log-probability difference for each task. For merged models, $|\Delta_i(h)|$ is recomputed against the same base model at the same tokens and prefixes.

3.1 Preservation of token-level policy shifts under merging

Section 2 characterizes token-level log-probability differences as a policy-shift signal for RLVR experts. We evaluate whether models produced by existing SFT-based merging methods retain this signal after merging. For each task, we first sample rollout sequences from the base model on the task queries. We then teacher-force the same generated tokens through the base model and the corresponding expert to obtain their per-token log-probabilities. For each token position indexed by h, we compute the magnitude of the policy shift $|\Delta_i(h)|$. We rank token positions by the absolute difference $|\Delta_i(h)|$ and retain the top 5% of positions. Holding these token positions and prefixes fixed, we replace the expert with each merged model and recompute the magnitude of the policy shift relative to the base model. Figure 2 compares the resulting $|\Delta_i(h)|$ distributions for the expert and merged models. The results show that each task expert induces large policy shifts on these base-sampled tokens, with the strongest shift observed for the memory expert. These token-level shifts arise from the expert's output perturbations on the corresponding prefixes. An effective merging method must preserve these output perturbations. An analysis of calibration trajectories split by correctness (Figure A in Appendix C.2) provides further evidence that the magnitude of the policy shift is task-relevant: on incorrect memory trajectories, the expert suppresses base-sampled tokens more strongly than on correct ones. In contrast, models produced by existing SFT-based merging techniques fail to retain these token-level shifts: their distributions remain concentrated near zero rather than matching the expert's distribution. Since these methods mainly target parameter-level criteria—such as element-wise sign consistency [Yadav et al., 2023, Yu et al., 2024] or global task-vector geometry [Gargiulo et al., 2025, Marczak et al., 2025]—they can fail to fully preserve RLVR's token-level policy shifts.

3.2 Policy-shift-weighted output-perturbation truncation (PS-SVD)

The truncation step should preserve the expert's output perturbations on the prefixes of these high-shift tokens. For expert i and layer ℓ, let $\mathcal{C}$ be the set of sampled calibration token instances. Each token instance $h \in \mathcal{C}$ has weight $\alpha_i(h) = |\Delta_i(h)|$, the magnitude of the expert-vs-base policy shift at that token. Let $\boldsymbol{z}_{i,\ell}(h) \in \mathbb{R}^C$ be the input to the layer-ℓ linear map (with output dimension R) at the same token instance. The task vector $\boldsymbol{\tau}_i^{(\ell)}$ changes that layer output by $\boldsymbol{\tau}_i^{(\ell)}\boldsymbol{z}_{i,\ell}(h)$. We therefore choose the rank-$k_{i,\ell}$ approximation that preserves these output changes, giving greater weight to errors at token instances where the expert deviates more from the base model. The rank $k_{i,\ell}$ is chosen from an energy budget η_E with the shared energy-threshold adjustment described in Appendix A. Assume $\sum_{h\in\mathcal{C}} \alpha_i(h) > 0$ and define $q_i(h) = \alpha_i(h)/\sum_{h'\in\mathcal{C}} \alpha_i(h')$. The resulting objective is

$$\begin{aligned}
&\min_{\operatorname{rank}(\hat{\boldsymbol{\tau}})\le k_{i,\ell}} \sum_{h\in\mathcal{C}} q_i(h) \left\| \boldsymbol{\tau}_i^{(\ell)}\boldsymbol{z}_{i,\ell}(h) - \hat{\boldsymbol{\tau}}\boldsymbol{z}_{i,\ell}(h) \right\|_2^2 \\
&\quad = \min_{\operatorname{rank}(\hat{\boldsymbol{\tau}})\le k_{i,\ell}} \sum_{h\in\mathcal{C}} q_i(h) \left\| \mathbf{E}\boldsymbol{z}_{i,\ell}(h) \right\|_2^2 \\
&\quad = \min_{\operatorname{rank}(\hat{\boldsymbol{\tau}})\le k_{i,\ell}} \operatorname{tr}\left(\mathbf{E}\mathbf{G}_{i,\ell}\mathbf{E}^\top\right) = \min_{\operatorname{rank}(\hat{\boldsymbol{\tau}})\le k_{i,\ell}} \left\| \mathbf{E}\mathbf{G}_{i,\ell}^{1/2} \right\|_{\mathrm{F}}^2,
\end{aligned} \tag{3}$$

where $\hat{\tau}$ is the candidate approximation for the fixed expert–layer pair (i,ℓ), $\mathbf{E} = \boldsymbol{\tau}_i^{(\ell)} - \hat{\tau}$, and $\mathbf{G}_{i,\ell} = \sum_{h\in\mathcal{C}} q_i(h)\boldsymbol{z}_{i,\ell}(h)\boldsymbol{z}_{i,\ell}(h)^\top$ is the policy-shift-weighted Gram matrix of the linear-map inputs. Thus, this formulation replaces ordinary SVD's unweighted parameter error with a low-rank objective that evaluates the task-vector error on input directions that occur on prefixes of high-shift tokens.

Exact truncation via SVD on output perturbations. We can compute the exact solution to Eq. (3) directly via a single SVD of the output-perturbation matrix. This removes the need to form $\mathbf{G}_{i,\ell}^{1/2}$ and $\mathbf{G}_{i,\ell}^{-1/2}$, which would otherwise incur $O(C^3)$ computational overhead and amplify numerical noise. Let $\mathbf{Z}_{q,i,\ell} \in \mathbb{R}^{|\mathcal{C}|\times C}$ stack the weighted linear-map inputs, with rows $\mathbf{Z}_{q,i,\ell}[h,:] = \sqrt{q_i(h)}\,\boldsymbol{z}_{i,\ell}(h)^\top$. Then $\mathbf{G}_{i,\ell} = \mathbf{Z}_{q,i,\ell}^\top \mathbf{Z}_{q,i,\ell}$. We define the weighted output-perturbation matrix as

$$\mathbf{Y}_{i,\ell} \coloneqq \boldsymbol{\tau}_i^{(\ell)} \mathbf{Z}_{q,i,\ell}^\top \in \mathbb{R}^{R\times|\mathcal{C}|}, \tag{4}$$

where each column is the expert's scaled output perturbation on one calibration state. The matrix $\mathbf{Y}_{i,\ell}$ has the same left Gram matrix as $\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}$:

$$\mathbf{Y}_{i,\ell}\mathbf{Y}_{i,\ell}^\top = \boldsymbol{\tau}_i^{(\ell)} \mathbf{Z}_{q,i,\ell}^\top \mathbf{Z}_{q,i,\ell} \boldsymbol{\tau}_i^{(\ell)\top} = \boldsymbol{\tau}_i^{(\ell)} \mathbf{G}_{i,\ell} \boldsymbol{\tau}_i^{(\ell)\top} = \left(\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}\right)\left(\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}\right)^\top.$$

Therefore, the required top-$k_{i,\ell}$ left singular subspace can be extracted directly from $\mathbf{Y}_{i,\ell}$:

$$\mathbf{U}_{i,\ell,k} = \mathrm{TopLeftSVD}_{k_{i,\ell}}(\mathbf{Y}_{i,\ell}). \tag{5}$$

Because $\mathbf{Y}_{i,\ell}$ and $\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}$ share this left Gram matrix, $\mathbf{U}_{i,\ell,k}$ also spans a valid top-$k_{i,\ell}$ left singular subspace of $\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}$. Using this shared subspace, the retained update $\hat{\boldsymbol{\tau}}_i^{(\ell)}$ is

$$\begin{aligned}
\hat{\boldsymbol{\tau}}_i^{(\ell)} &= \underbrace{\left[\mathbf{U}_{i,\ell,k}\mathbf{U}_{i,\ell,k}^\top \left(\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}\right)\right]}_{\text{rank-}k_{i,\ell}\text{ truncation of }\boldsymbol{\tau}_i^{(\ell)}\mathbf{G}_{i,\ell}^{1/2}} \mathbf{G}_{i,\ell}^{-1/2} \\
&= \mathbf{U}_{i,\ell,k}\mathbf{U}_{i,\ell,k}^\top \boldsymbol{\tau}_i^{(\ell)} \left(\mathbf{G}_{i,\ell}^{1/2}\mathbf{G}_{i,\ell}^{-1/2}\right) \\
&= \mathbf{U}_{i,\ell,k}\mathbf{U}_{i,\ell,k}^\top \boldsymbol{\tau}_i^{(\ell)}.
\end{aligned} \tag{6}$$

The derivation above shows that $\hat{\boldsymbol{\tau}}_i^{(\ell)} = \mathbf{U}_{i,\ell,k}\mathbf{U}_{i,\ell,k}^\top\boldsymbol{\tau}_i^{(\ell)}$ in Eq. (6) is the exact minimizer of Eq. (3), without requiring $\mathbf{G}_{i,\ell}^{1/2}$ or $\mathbf{G}_{i,\ell}^{-1/2}$.

Diagonalized variant. Although the implicit SVD on $\mathbf{Y}_{i,\ell}$ avoids constructing $\mathbf{G}_{i,\ell}$, computing the weighted output-perturbation matrix $\mathbf{Y}_{i,\ell} \in \mathbb{R}^{R\times|\mathcal{C}|}$ remains computationally expensive, costing $O(R\cdot C\cdot|\mathcal{C}|)$. PSA-D eliminates this bottleneck, reducing the complexity to a scalable $O(|\mathcal{C}|\cdot C)$ by avoiding the computation of full output perturbations. Instead of evaluating the full perturbation $\boldsymbol{\tau}_i^{(\ell)}\boldsymbol{z}_{i,\ell}(h) \in \mathbb{R}^R$ for every token, it directly accumulates the per-coordinate diagonal entries of the Gram matrix:

$$\omega_{i,\ell}[c] \;\coloneqq\; \sum_{h\in\mathcal{C}} q_i(h)\,\boldsymbol{z}_{i,\ell}(h)[c]^2 = \frac{\sum_{h\in\mathcal{C}}\alpha_i(h)\,\boldsymbol{z}_{i,\ell}(h)[c]^2}{\sum_{h\in\mathcal{C}}\alpha_i(h)}. \tag{7}$$

Stacking these per-coordinate values gives $\boldsymbol{\omega}_{i,\ell} = (\omega_{i,\ell}[1],\ldots,\omega_{i,\ell}[C]) \in \mathbb{R}^C$, so $\mathbf{D}_{i,\ell} = \mathrm{diag}(\boldsymbol{\omega}_{i,\ell}) = \mathrm{diag}(\mathbf{G}_{i,\ell})$. Substituting $\mathbf{D}_{i,\ell}$ for $\mathbf{G}_{i,\ell}$ in Eq. (3) yields the diagonal low-rank objective:

$$\hat{\boldsymbol{\tau}}_i^{(\ell)} \;\coloneqq\; \underset{\mathrm{rank}(\hat{\boldsymbol{\tau}})\le k_{i,\ell}}{\arg\min} \left\|(\boldsymbol{\tau}_i^{(\ell)} - \hat{\boldsymbol{\tau}})\mathbf{D}_{i,\ell}^{1/2}\right\|_{\mathrm{F}}^2. \tag{8}$$

Because $\mathbf{D}_{i,\ell}$ is diagonal, Eq. (8) reduces to an ordinary truncated SVD on the column-scaled task vector $\boldsymbol{\tau}_i^{(\ell)}\mathbf{D}_{i,\ell}^{1/2}$, followed by right multiplication by $\mathbf{D}_{i,\ell}^{-1/2}$ to undo the column scaling. A coordinate that is inactive on all calibration tokens has $\omega_{i,\ell}[c] = 0$. We therefore apply the element-wise floor $\omega_{i,\ell}[c] \leftarrow \max(\omega_{i,\ell}[c], \epsilon_0)$ with $\epsilon_0 = 10^{-2}$, so that $\mathbf{D}_{i,\ell} \succeq \epsilon_0\mathbf{I} \succ 0$ and $\mathbf{D}_{i,\ell}^{-1/2}$ is well defined. Thus, PSA-D is a computationally efficient approximation to the exact full-Gram truncation in PSA-G.

3.3 Polar-aligned merging with energy renormalization

After the policy-shift-weighted truncation of Section 3.2, PSA merges the retained expert updates in two steps. It first aligns the truncated bases to reduce residual directional conflict. It then restores the per-expert update scale before aggregating the aligned updates.

Polar alignment. The policy-shift-weighted truncation selects a policy-shift-aware output subspace for each expert independently. These subspaces can still overlap across experts, so direct summation can reintroduce directional interference. We reduce this overlap with the polar Procrustes projection [Schönemann, 1966, Higham, 1986]. Here, $\mathrm{polar}(\mathbf{M})$ denotes the nearest column-orthonormal matrix to $\mathbf{M}$ in Frobenius norm. Let $\hat{\boldsymbol{\tau}}_i^{(\ell)} = \mathbf{U}_i^{(\ell)} \boldsymbol{\Sigma}_i^{(\ell)} \mathbf{V}_i^{(\ell)\top}$ be the compact SVD of the retained update, with $\mathbf{U}_i^{(\ell)} \in \mathbb{R}^{R \times k_{i,\ell}}$ and $\mathbf{V}_i^{(\ell)} \in \mathbb{R}^{C \times k_{i,\ell}}$. The columns of $\mathbf{U}_i^{(\ell)}$ span the same subspace as those of $\mathbf{U}_{i,\ell,k}$ from Eq. (5). We concatenate the left and right bases across experts,

$$\bar{\mathbf{U}}_\ell = [\mathbf{U}_1^{(\ell)} \mid \cdots \mid \mathbf{U}_N^{(\ell)}] \in \mathbb{R}^{R \times K_\ell}, \qquad \bar{\mathbf{V}}_\ell = [\mathbf{V}_1^{(\ell)} \mid \cdots \mid \mathbf{V}_N^{(\ell)}] \in \mathbb{R}^{C \times K_\ell},$$

where $K_\ell = \sum_{i=1}^{N} k_{i,\ell}$ and $K_\ell \le \min(R, C)$. We satisfy this constraint with a shared energy-threshold adjustment (Appendix A), which ensures that the polar projection is well defined.

Following Gargiulo et al. [2025], we compute the corresponding polar frames $\hat{\mathbf{U}}_\ell := \mathrm{polar}(\bar{\mathbf{U}}_\ell)$ and $\hat{\mathbf{V}}_\ell := \mathrm{polar}(\bar{\mathbf{V}}_\ell)$. We partition $\hat{\mathbf{U}}_\ell$ and $\hat{\mathbf{V}}_\ell$ by the original expert block sizes and denote the resulting blocks by $\hat{\mathbf{U}}_\ell^{(i)} \in \mathbb{R}^{R \times k_{i,\ell}}$ and $\hat{\mathbf{V}}_\ell^{(i)} \in \mathbb{R}^{C \times k_{i,\ell}}$. The aligned expert update is

$$\boldsymbol{\tau}_{\mathrm{align},i}^{(\ell)} := \hat{\mathbf{U}}_\ell^{(i)} \boldsymbol{\Sigma}_i^{(\ell)} \big(\hat{\mathbf{V}}_\ell^{(i)}\big)^\top. \tag{9}$$

Because the columns of each polar frame are orthonormal, different expert blocks are mutually orthogonal after partitioning, i.e., $\langle \boldsymbol{\tau}_{\mathrm{align},i}^{(\ell)}, \boldsymbol{\tau}_{\mathrm{align},j}^{(\ell)} \rangle_{\mathrm{F}} = 0$ for $i \neq j$.

Energy renormalization. After directional alignment, PSA restores the per-expert update scale before summing the aligned updates. This step does not assume that discarded tail directions are recoverable from the retained subspace. Rather, it compensates for the norm shrinkage introduced by low-rank truncation, so experts or layers with more aggressive truncation are not systematically underweighted in the final sum. For each expert and layer, we use the ϵ-stabilized coefficient $\beta_i^{(\ell)} := \|\boldsymbol{\tau}_i^{(\ell)}\|_{\mathrm{F}} / (\|\boldsymbol{\tau}_{\mathrm{align},i}^{(\ell)}\|_{\mathrm{F}} + \epsilon)$, where $\epsilon > 0$. This coefficient matches the aligned update to the original task-vector Frobenius scale. The merged layer is

$$\mathbf{W}_{\mathrm{merged}}^{(\ell)} = \mathbf{W}_{\mathrm{pre}}^{(\ell)} + \sum_{i=1}^{N} \beta_i^{(\ell)} \boldsymbol{\tau}_{\mathrm{align},i}^{(\ell)}. \tag{10}$$

One-dimensional parameters, such as biases and normalization scales, are averaged across experts.

4 Experimental Results

4.1 Experimental setup

We evaluate PSA in two settings: a five-expert Qwen2.5-7B setting and a three-expert Qwen3-1.7B setting. For Qwen2.5-7B, we use Qwen2.5-7B-Instruct [Yang et al., 2025b] as the shared base and merge five public RLVR expert checkpoints: `ReasonFlux-Coder-7B` [Wang et al., 2025b] for coding, `Qwen2.5-7B-Instruct-ToolRL-grpo-cold` [Qian et al., 2025] for tool use, `RL-MemoryAgent-7B` [Yu et al., 2025a] for long-context memory, `SPEAR-SearchQA-Qwen2.5-7B` [Qin et al., 2025] for agentic search, and `LUFFY-Qwen-Instruct-7B` [Yan et al., 2025] for math. For Qwen3-1.7B [Yang et al., 2025a], we merge three RLVR experts: two custom experts trained for instruction following and math using the VERL framework, and the publicly released Lucy [Dao et al., 2025] checkpoint for agentic search (URLs in Appendix D.3).

We compare PSA with a broad set of model merging baselines. These include three representative parameter-level methods, namely Task Arithmetic [Ilharco et al., 2023], TIES [Yadav et al., 2023],

Table 1: **Merging results on Qwen2.5-7B with five RLVR experts.** We merge experts for coding, tool use, long-context memory, search, and math. For Coding, we report the average of ACC and UT on LiveBench (LB) and LiveCodeBench (LCB). For Tool Use, we report BFCL v4 Live and Non-Live AST scores. For Memory, we report the average SubEM on RULER HotpotQA (7K, 14K) and SQuAD (32K, 64K) in the Recurrent setting. For Search, we report SimpleQA ACC; for Math, we report AIME24/25/26 mean@8. Avg. is the macro-average over the five task groups. Within each block, **bold** marks the best score in each column; highlighted rows indicate our methods.

Method	Coding		Tool Use		Memory	Search	Math			Avg.
	LB	LCB	Live	Non-Live	avg	SimpleQA	AIME24	AIME25	AIME26	
	ACC/UT	ACC/UT	AST	AST	SubEM	ACC	mean@8	mean@8	mean@8	
Reference Models										
Base	43.19	36.53	74.02	86.31	58.99	77.68	10.42	8.33	7.08	53.06
Coding Expert	**46.63**	**38.46**	73.80	86.44	58.20	73.82	12.92	8.33	6.25	52.77
Tool Expert	42.28	36.04	**77.05**	**88.06**	59.77	77.80	10.00	9.17	9.58	53.77
Memory Expert	42.96	37.58	75.50	86.27	**78.00**	74.32	9.58	5.83	7.50	56.22
Search Expert	38.03	34.44	61.50	74.30	55.28	**86.40**	10.83	7.08	7.50	50.86
Math Expert	34.87	34.03	52.40	44.60	61.52	78.28	**17.08**	**16.25**	**17.08**	47.91
Merging Methods										
Task Arithmetic	43.48	37.66	75.90	83.70	65.23	72.19	12.92	9.17	8.33	53.59
TIES	44.45	39.20	76.61	84.50	76.95	85.51	14.58	10.00	6.67	59.05
DARE	44.92	40.19	63.80	78.79	79.69	82.95	12.92	12.50	9.17	57.61
TSV	42.91	38.62	77.13	84.67	74.22	82.19	12.92	11.25	10.83	57.95
STAR	44.43	39.00	77.40	85.21	72.66	80.08	12.92	10.83	9.58	57.38
Iso-CTS	40.06	36.98	75.21	83.25	66.99	78.93	11.25	10.00	10.42	54.85
RAM	44.37	38.04	77.57	85.33	71.29	81.26	13.33	9.58	12.08	57.37
RAM+	42.03	37.88	**77.65**	85.42	73.44	79.88	13.75	10.00	7.08	57.02
PSA-D (Ours)	45.41	40.50	75.72	84.25	82.23	86.18	15.83	12.92	**13.33**	61.08
PSA-G (Ours)	**47.80**	**41.81**	75.65	**86.44**	**82.99**	**86.20**	**17.08**	**14.58**	**13.33**	**62.01**

and DARE [Yu et al., 2024], and three spectral merging methods, namely TSV [Gargiulo et al., 2025], STAR [Lee et al., 2025], and Iso-CTS [Marczak et al., 2025]. We further compare against RAM [Yuan et al., 2026] and its extension RAM+, two recent merging methods designed for RLVR experts. The algorithmic details of these baselines are provided in Appendix E. For our method, PSA-G denotes the full-Gram output-perturbation variant described in Section 3.2, and PSA-D denotes its diagonalized surrogate. We describe the Qwen3-1.7B training recipe in Appendix D.1. Appendix D details the evaluation protocols for LiveBench [White et al., 2025], LiveCodeBench [Jain et al., 2024], BFCL [Patil et al., 2025], RULER [Hsieh et al., 2024], IFEval [Zhou et al., 2023], AIME24/25/26, and SimpleQA [Wei et al., 2024].

Calibration and rank selection. We use 128 calibration queries per expert in both model settings, disjoint from the evaluation benchmarks. For Qwen2.5-7B, we use validation splits of CodeContests, ToolRL, and HotpotQA for Coding, Tool Use, and Memory, respectively, NQ-open validation for Search, and Hendrycks-MATH training samples for Math. We generate rollouts with the shared base model and teacher-force the same tokens through the base and corresponding expert to obtain log-probability differences and expert linear-map input states. Starting from the energy budget $\eta_E = 0.9$, we lower a threshold shared across all experts and layers when needed to satisfy the polar-alignment constraint $\sum_i k_{i,\ell} \le \min(R, C)$ in every layer (Appendix A).

4.2 Model merging results

Tables 1 and 2 compare the task-level performance of individual experts and merged models in the five-expert Qwen2.5-7B setting and the three-expert Qwen3-1.7B setting, respectively. PSA-G achieves the highest macro-average among all merging methods on both base models. It attains the best merged score on every Qwen3-1.7B metric and on every Qwen2.5-7B metric except BFCL Live AST. Among the spectral baselines, TSV and STAR achieve strong Tool Use performance, but their gains do not transfer uniformly across tasks. In particular, both methods suffer substantial degradation on Memory (74.22 and 72.66 vs. 78.00 for the Memory expert), indicating that strong performance in one expert task can come at the cost of another. This imbalance is most pronounced on Memory tasks. While most parameter-level and spectral baselines lose substantial long-context

Table 2: **Merging results on Qwen3-1.7B with three RLVR experts.** We report ACC for IFEval and SimpleQA, and mean@8 for AIME24/25/26. Avg. is the macro-average over the three task groups. Our methods are averaged over three decoding seeds, and other rows are single runs. Within each block, **bold** marks the best score in each column; highlighted rows indicate our methods.

Method	IF	Math			Search	Avg.
	IFEval	AIME24	AIME25	AIME26	SimpleQA	
	ACC	mean@8	mean@8	mean@8	ACC	
Reference Models						
Base	60.79	47.50	36.25	32.90	71.28	56.98
IF Expert	**64.99**	48.75	35.83	35.83	75.62	60.25
Math Expert	62.47	**54.17**	**43.75**	**41.67**	71.40	60.13
Search Expert (Lucy)	64.15	51.25	39.58	36.67	**76.77**	61.14
Merging Methods						
Task Arithmetic	64.39	51.67	38.33	37.50	76.15	61.01
TIES	64.15	49.58	37.06	40.42	75.38	60.63
DARE	66.31	52.08	38.33	37.50	75.54	61.50
TSV	65.23	53.33	41.25	42.08	75.38	62.05
STAR	65.71	51.25	38.75	38.75	76.43	61.69
Iso-CTS	64.51	50.42	33.33	39.58	74.56	60.06
RAM	65.83	51.67	37.50	39.17	72.37	60.33
RAM+	64.99	54.17	40.83	41.67	76.03	62.19
PSA-D (Ours)	67.31	57.08	41.67	43.33	77.46	64.04
PSA-G (Ours)	**67.55**	**57.36**	**42.36**	**45.50**	**78.11**	**64.69**

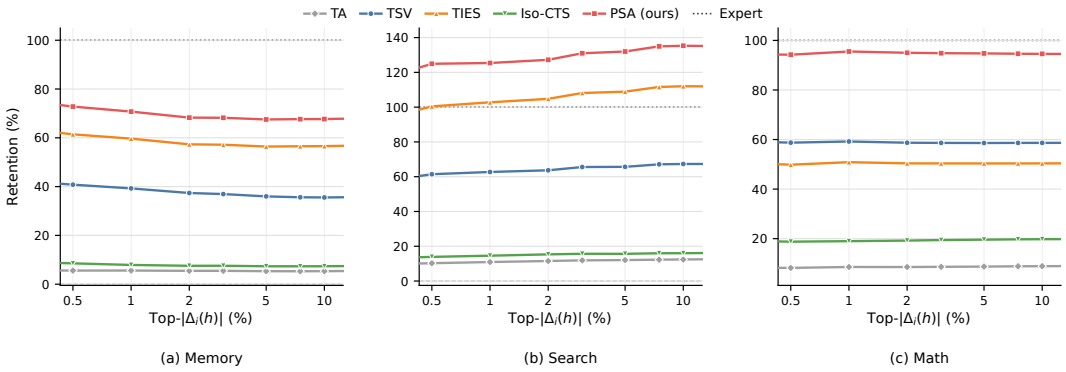


Figure 3: **Preservation of expert policy shifts on tokens generated by the base model.** For Memory, Search, and Math, we generate rollouts with the base model and score each generated token under both the expert and the base model. We rank base-generated tokens by $|\Delta_i(h)|$ and retain those with the largest values at each plotted percentage. The y-axis reports how much of the expert's log-probability change is retained by the merged model on the same token set.

performance, PSA preserves the memory expert's performance across all Memory metrics. This suggests that PSA better retains memory-relevant behavior than conventional merging methods.

4.3 Analysis

Preservation of token-level policy shifts. We evaluate the retention of expert-induced log-probability changes on Memory, Search, and Math. For each task, we rank base-generated tokens by the magnitude of the expert–base log-probability difference and compare retention across the selected token fractions. Figure 3 shows that PSA retains approximately 67–74% of the expert shift on Memory, compared with 36–41% for TSV and less than 10% for Iso-CTS. On Math, PSA remains close to the expert reference at approximately 94–96%, while TSV and TIES retain about 59% and 50%, respectively. On Search, PSA also achieves higher policy-shift retention than the compared baselines across all plotted token fractions.

Table 3: **Component analysis of PSA on Qwen2.5-7B with five RLVR experts.** We compare standard SVD with the proposed PS-SVD and progressively add cross-expert polar alignment (Polar) and per-expert energy renormalization (Renorm.) within each group. Avg. is the macro-average over the available task groups. **Bold** marks the best score in each column.

Method	Coding	Tool Use		Memory	Search	Math	Avg.
	LB	Live	Non-Live	avg	SimpleQA	AIME26	
	ACC/UT	AST	AST	SubEM	ACC	mean@8	
(a) SVD	45.17	65.06	81.25	77.24	81.00	10.00	57.31
(b) SVD + Polar	43.01	76.68	87.73	73.80	82.80	9.17	58.20
(c) SVD + Polar + Renorm	43.26	76.46	87.21	76.56	83.00	10.83	59.10
(d) PS-SVD	47.44	65.00	80.75	81.64	83.20	11.67	59.36
(e) PS-SVD + Polar	47.04	**77.57**	**88.58**	79.58	83.60	11.25	60.91
(f) PS-SVD + Polar + Renorm (PSA-G)	**47.80**	75.65	86.44	**82.99**	**86.20**	**13.33**	**62.27**

Table 4: **Sensitivity of PSA to the calibration data distribution on Qwen2.5-7B** (N=5 RLVR experts). **In-domain**: the default expert-specific calibration sources in Section 4.1. **Same-task proxy**: public datasets disjoint from training and evaluation (APPS, xLAM-60k, 2WikiMultihopQA, MuSiQue, GSM8K). Benchmarks and column ordering match Table 1. The highlighted row indicates our default configuration.

Calibration	Coding		Tool Use		Memory	Search	Math			Avg.
	LB	LCB	Live	Non-Live	avg	SimpleQA	AIME24	AIME25	AIME26	
	ACC/UT	ACC/UT	AST	AST	SubEM	ACC	mean@8	mean@8	mean@8	
Same-task proxy	45.22	39.91	74.46	85.04	82.03	84.96	15.42	12.92	11.67	60.53
In-domain (PSA-G)	**47.80**	**41.81**	**75.65**	**86.44**	**82.99**	**86.20**	**17.08**	**14.58**	**13.33**	**62.01**

Ablation study. Table 3 presents a component-wise ablation of PSA-G on Qwen2.5-7B by merging five RLVR-trained experts across coding, tool use, long-context memory, search, and math tasks. We evaluate the individual and cumulative contributions of our key components: the proposed policy-shift-weighted output-perturbation truncation method (PS-SVD), cross-expert polar alignment (*Polar*), and per-expert energy renormalization (*Renorm.*). Adding each component increases the macro-average in both groups, and the full configuration (f) achieves the best macro-average. Replacing standard SVD with PS-SVD delivers a clear performance gain across all corresponding configurations (e.g., $+2.05$ points in the macro-average from (a) to (d) and $+3.17$ points from (c) to (f)). This indicates that evaluating task-vector errors along input directions that occur at high-shift token prefixes effectively preserves critical RLVR expert behaviors during low-rank approximation.

Sensitivity to calibration data distribution. To test whether PSA requires in-domain calibration data, we replace the default calibration sources described in Section 4.1 with disjoint, same-task public datasets. We randomly sample 128 queries per expert from APPS for coding, xLAM-60k for tool use, 2WikiMultihopQA for memory, MuSiQue for search, and GSM8K for math (URLs in Appendix D.3). As shown in Table 4, same-task proxy calibration lowers the macro-average from 62.01 to 60.53. PSA therefore benefits from calibration queries drawn from each expert's own RL training corpus. Even with proxy calibration, it still exceeds the strongest merging baseline (TIES) in Table 1.

PSA-D Ablation on Policy-Shift Diagonal Importance. PSA-D benefits from the coordinate-level weighting pattern induced by policy shifts. To test whether diagonal reweighting alone can reproduce this gain, we replace the policy-shift diagonal $\mathbf{D}_{i,\ell} = \mathrm{diag}(\mathbf{G}_{i,\ell})$ with two controls: a range-matched random diagonal with entries sampled uniformly between each expert's minimum and maximum policy-shift weights, and a task-balanced Dirichlet diagonal that preserves task-level mass balance while randomizing the allocation across coordinates. We keep the ϵ_0 floor, weighted-energy truncation, polar alignment, and energy renormalization unchanged. As shown in Table 5, PSA-D achieves an average score of 69.62, whereas the random and task-balanced diagonals score 66.25 and 67.43, respectively, both below standard SVD.

Table 5: **Ablation of policy-shift diagonal weights in PSA-D on Qwen2.5-7B ($N = 3$).** Coding averages LiveBench ACC and UT; Tool averages BFCL Live and Non-Live; Memory averages HotpotQA SubEM at 7K and 14K. Avg. is the mean across the three domains. PSA-D-Random results are averaged over three independent seeds, with the diagonal resampled for each seed.

Method	Coding	Tool	Memory	Avg.
Standard SVD ($\mathbf{D}$=$\mathbf{I}$)	45.88	82.45	77.35	68.56
PSA-D-**Random** (random diag.)	44.57	80.88	73.31	66.25
PSA-D-**Dir-bal** (task-balanced Dirichlet diag.)	43.60	82.50	76.17	67.43
PSA-D	**46.53**	**82.64**	**79.69**	**69.62**

5 Related Work

Coordinate-wise merging Parameter-level merging methods operate directly on task vectors in weight space. Task Arithmetic [Ilharco et al., 2023] linearly adds task vectors, while TIES [Yadav et al., 2023] and DARE [Yu et al., 2024] improve this recipe through sign resolution and stochastic sparsification, respectively. Fisher Merging [Matena and Raffel, 2022], RegMean [Jin et al., 2023], and AdaMerging [Yang et al., 2024] add data-dependent parameter weights or learned merge coefficients. For RLVR checkpoints, RAM [Yuan et al., 2026] separates shared and task-specific parameters by magnitude-based masking and rescales the task-specific component. These approaches remain element-wise or layer-wise, making it difficult to interpret which functional components are removed or preserved during merging. This motivates subspace-level merging criteria that operate on structured components of the update.

Subspace-based spectral merging To obtain more interpretable and structured merge components, spectral merging methods decompose task vectors into low-rank subspaces before recombination. TSV [Gargiulo et al., 2025] truncates each task vector with SVD and polar-aligns the retained bases. CART [Choi et al., 2024] separates mean and residual task directions, STAR [Lee et al., 2025] rescales truncated singular components, and Iso-CTS [Marczak et al., 2025] separates common and task-specific subspaces with isotropic spectra. These methods align global task-vector geometry, but their subspaces are still selected by unweighted spectral energy. PSA reinterprets spectral merging for RLVR experts by constructing a policy-shift-weighted activation Gram matrix from high-shift tokens. The resulting full-Gram objective selects subspaces that preserve the expert's layer-output perturbations on those tokens, rather than subspaces that only have large unweighted parameter energy.

6 Conclusion

We studied model merging for RLVR post-trained experts, where the goal is to consolidate complementary same-base checkpoints without multi-task RLVR training. Our central observation is that such experts should not be merged solely based on global task-vector geometry. Instead, spectral selection should prioritize layer-output perturbations at tokens with large absolute expert–base log-probability differences. We proposed PSA, a calibration-based spectral merging method that uses policy-shift-weighted input states of linear maps to select subspaces that preserve output perturbations. The method then applies polar alignment and restores the Frobenius scales of the expert updates. Across Qwen2.5-7B and Qwen3-1.7B expert sets, PSA achieves the highest macro-average over representative parameter-level, spectral, and RLVR-specific baselines, showing that policy-shift-aware subspace selection is an effective principle for merging RLVR experts.

Limitations. PSA requires calibration rollouts and performs best when the queries come from each expert's own RL training corpus. Its gains are largest on Memory, whereas on Tool Use it is comparable to the strongest baselines. Our experiments cover models up to 7B and up to five experts. Polar alignment may also distort useful directions shared by experts whose updates overlap heavily, which we leave to future work.

Acknowledgments and Disclosure of Funding

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants [RS-2022-II220959 (No.2022-0-00959), (Part 2) Few-Shot Learning of Causal Inference in Vision and Language for Decision Making; RS-2026-25526850, High-Efficiency Neural Networks for Artificial General Intelligence (HERMES-Net); No.RS2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University); No. RS-2024-00398353, Development of Countermeasure Technologies for Generative AI Security Threats; IITP-2026-RS-2023-00260091, 10%, ITRC (Information Technology Research Center) support program; IITP-2026-RS-2020-II201819, 10%, ICT Creative Consilience Program; IITP-2026-RS-2026-25616664, AI Star Fellowship Support Program] funded by the Korea government (MSIT), and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT and MOE) (No. RS-2026-25522655; No. RS-2026-25500579).

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Jiho Choi, Donggyun Kim, Chanhyuk Lee, and Seunghoon Hong. Revisiting weight averaging for model merging. *arXiv preprint arXiv:2412.12153*, 2024.

Alan Dao, Dinh Bach Vu, Alex Nguyen, and Norapat Buppodom. Lucy: edgerunning agentic web search on mobile with machine generated task vectors. *arXiv preprint arXiv:2508.00360*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *CVPR*, pages 18695–18705, 2025.

Nicholas J Higham. Computing the polar decomposition—with applications. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1160–1174, 1986.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *COLM*, 2024.

Kexin Huang, Haoming Meng, Junkang Wu, Jinda Lu, Chiyu Ma, Ziqian Chen, Xue Wang, Bolin Ding, Jiancan Wu, Xiang Wang, Xiangnan He, Guoyin Wang, and Jingren Zhou. On the direction of RLVR updates for LLM reasoning: Identification and exploitation. *arXiv preprint arXiv:2603.22117*, 2026.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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

Yu-Ang Lee, Ching-Yun Ko, Tejaswini Pedapati, I-Hsin Chung, Mi-Yen Yeh, and Pin-Yu Chen. STAR: Spectral truncation and rescale for model merging. In *NAACL*, pages 496–505, 2025.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceMath: Advancing frontier math reasoning with post-training and reward modeling. In *ACL Findings*, pages 3993–4015, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

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*, pages 43177–43199, 2025.

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

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *ICML*, 2025.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, et al. Learn the ropes, then trust the wins: Self-imitation with progressive exploration for agentic reinforcement learning. *arXiv preprint arXiv:2509.22601*, 2025.

Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025a.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving LLM coder and unit tester via reinforcement learning. In *NeurIPS*, 2025b.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. LiveBench: A challenging, contamination-limited LLM benchmark. In *ICLR*, 2025.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *NeurIPS*, pages 7093–7115, 2023.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *NeurIPS*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025b.

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*, 2024.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

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*, pages 57755–57775, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *NeurIPS*, 2025b.

Xiangchi Yuan, Dachuan Shi, Chunhui Zhang, Zheyuan Liu, Shenglong Yao, Soroush Vosoughi, and Wenke Lee. Behavior knowledge merge in reinforced agentic models. *arXiv preprint arXiv:2601.13572*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, et al. The path not taken: RLVR provably learns off the principals. *arXiv preprint arXiv:2511.08567*, 2025.

Appendix

Table of Contents

A. Algorithm: PSA ..... p. 15

B. Incremental Merging ..... p. 15

C. Additional Analyses ..... p. 15
- C.1. Expert-Prefix Cascade Attribution for Full Gram ..... p. 15
- C.2. Token-Level Policy-Shift Analysis on RLVR-Training Calibration Data ..... p. 17
- C.3. Policy-Shift Weighting versus Non-Policy-Shift Column Metrics ..... p. 18
- C.4. Sensitivity to the Energy Budget and the Calibration Set ..... p. 19

D. Experimental Details ..... p. 19
- D.1. Training Recipes for Qwen3-1.7B RLVR Experts ..... p. 19
- D.2. Detailed Evaluation Protocols ..... p. 20
  - D.2.1. Coding (LiveBench, LiveCodeBench) ..... p. 20
  - D.2.2. Tool Use (BFCL v4) ..... p. 21
  - D.2.3. Long-Context Memory (RULER) ..... p. 21
  - D.2.4. Instruction Following (IFEval) ..... p. 21
  - D.2.5. Mathematical Reasoning (AIME) ..... p. 21
  - D.2.6. Search (SimpleQA) ..... p. 22
- D.3. Model Checkpoints and Datasets ..... p. 22

E. Detailed Descriptions of Merging Baselines ..... p. 22

A Algorithm: PSA

Algorithm 1 gives the full PSA procedure. The method runs in two offline phases. *Phase A* scores calibration tokens for each expert i using $\alpha_i(h) = |\Delta_i(h)|$ and stores the corresponding expert layer inputs $\boldsymbol{z}_{i,\ell}(h)$. *Phase B* performs implicit full-Gram spectral truncation, polar alignment, and energy renormalization for each 2D weight matrix and averages 1D parameters across experts. For an energy threshold η, let $k_{i,\ell}(\eta)$ be the smallest rank that preserves an η fraction of the spectral energy of $\mathbf{Y}_{i,\ell} = \boldsymbol{\tau}_i^{(\ell)} \mathbf{Z}_{q,i,\ell}^\top$. Polar alignment requires $\sum_i k_{i,\ell}(\eta) \leq \min(R, C)$ in every layer. We enforce this constraint with a shared energy-threshold adjustment. Starting from the energy budget $\eta_E \in (0, 1)$, we lower the threshold η shared by all experts until the constraint holds in every layer. Every expert therefore keeps the same fraction of its weighted spectral energy, and experts with slower spectral decay retain more directions. We set $\eta_E = 0.9$ as the largest common budget that satisfies the constraint, rather than tuning it on downstream performance; Appendix C.4 reports other budgets.

B Incremental Merging

In practice, new RLVR experts may become available after a merged model has already been deployed, so we test whether an existing three-expert merge can be extended without rebuilding the merge from scratch. Starting from each method's merged checkpoint for Coding, Tool, and Memory, we add the previously described Search and Math experts to obtain one five-expert model. Since all five experts are post-trained from the same Qwen2.5-7B-Instruct base, the second-stage merge still operates in a shared parameter space. For Stage 2 calibration, we use one 128-query set for the existing three-task merged model and one 128-query set for each newly added task, for 384 (128×3) calibration queries in total.

Table A reports task-group benchmarks for both stages of the incremental merge. In Stage 1, PSA outperforms every merging baseline in terms of the overall average score. In Stage 2, after the Search and Math experts are added, PSA again leads every merging baseline in terms of the overall average score, so its margin over the baselines is largely maintained. This confirms that policy-shift-guided subspace selection remains effective when experts arrive incrementally rather than all at once, supporting its use in the more realistic setting where the expert pool grows over time.

C Additional Analyses

C.1 Expert-Prefix Cascade Attribution for Full Gram

This ablation tests whether the full-Gram variant benefits from a layer-specific estimate of how each block contributes to the expert's next-token log-probability shift. The shared-q full Gram uses $q_i(h) = |\Delta_i(h)|$ for every layer, so a token receives the same importance weight in all blocks whenever the full expert changes its target-token log probability. That shared weight is coarse because the final expert shift is produced by a sequence of blocks. A block should receive high weight for a token only when adding that block moves the target-token log probability in the same direction as the full expert. We therefore build an expert-prefix cascade that estimates a block-level marginal contribution before constructing the full Gram matrix.

For expert i and prefix length k, the mixed model $M_i^{\leq k}$ uses expert blocks $[0, k)$ and base blocks $[k, L)$, while keeping the base embedding, final norm, and language-model head. For token h, we define the prefix shift as $\Delta_i^{\leq k}(h) = \log p_{M_i^{\leq k}}(v_h \mid v_{<h}) - \log p_{\text{base}}(v_h \mid v_{<h})$. The marginal contribution of block ℓ is the difference between two adjacent prefixes, $a_{i,\ell}(h) = \Delta_i^{\leq \ell+1}(h) - \Delta_i^{\leq \ell}(h)$. This difference measures how much the target-token log probability changes when block ℓ is added to the expert prefix. We keep the part of this marginal contribution that agrees with the full expert shift, $c_{i,\ell}(h) = \max(0, \text{sign}(\Delta_i(h))\, a_{i,\ell}(h))$, where $\Delta_i(h) = \log p_{\text{expert}}(v_h \mid v_{<h}) - \log p_{\text{base}}(v_h \mid v_{<h})$. The raw layer weight is approximately $|\Delta_i(h)|\, c_{i,\ell}(h)$ up to a numerical floor $\epsilon = 10^{-8}$. This procedure is not an inference-time cascade. It is used only to assign layer-specific token weights for $\mathbf{G}_{i,\ell} = \sum_h q_{i,\ell}(h) \boldsymbol{z}_{i,\ell}(h) \boldsymbol{z}_{i,\ell}(h)^\top / \sum_h q_{i,\ell}(h)$, where $\boldsymbol{z}_{i,\ell}(h)$ is still collected

Algorithm 1: PSA

Input: Pretrained $\boldsymbol{\theta}_{\text{pre}}$; experts $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$; calibration queries $\mathcal{Q}_{\text{cal}}$; energy budget η_E; stability constant $\epsilon = 10^{-12}$.

Output: Merged parameters $\boldsymbol{\theta}_{\text{merged}}$.

foreach *expert* i **do**
  $\boldsymbol{\tau}_i \leftarrow \boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{pre}}$;
end
Roll out $\mathcal{Q}_{\text{cal}}$ with $\boldsymbol{\theta}_{\text{pre}}$ to form sampled token instances $\mathcal{C}$;
Teacher-force $\mathcal{C}$ through $\boldsymbol{\theta}_{\text{pre}}$ and $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$ to obtain sampled-token probabilities and expert layer inputs $\boldsymbol{z}_{i,\ell}(h)$;
for $i = 1, \ldots, N$ **do**
  Compute $\Delta_i(h)$ by Eq. (1) and set $\alpha_i(h) \leftarrow |\Delta_i(h)|$ for all $h \in \mathcal{C}$;
end
foreach *2D weight matrix* $\mathbf{W}^{(\ell)} \in \mathbb{R}^{R \times C}$ **do**
  for $i = 1, \ldots, N$ **do**
    $q_i(h) \leftarrow \alpha_i(h) / \sum_{h' \in \mathcal{C}} \alpha_i(h')$ for all $h \in \mathcal{C}$;
    Form $\mathbf{Y}_{i,\ell} = \boldsymbol{\tau}_i^{(\ell)} \mathbf{Z}_{q,i,\ell}^{\top}$ by Eq. (4);
  end
end
$\eta \leftarrow \max\{\eta' \le \eta_E : \sum_i k_{i,\ell}(\eta') \le \min(R, C)\ \forall \ell\}$; `// shared energy threshold`
foreach *2D weight matrix* $\mathbf{W}^{(\ell)} \in \mathbb{R}^{R \times C}$ **do**
  for $i = 1, \ldots, N$ **do**
    $k_{i,\ell} \leftarrow k_{i,\ell}(\eta)$; $\mathbf{U}_{i,\ell,k} \leftarrow \text{TopLeftSVD}_{k_{i,\ell}}(\mathbf{Y}_{i,\ell})$;
    $\hat{\boldsymbol{\tau}}_i^{(\ell)} \leftarrow \mathbf{U}_{i,\ell,k} \mathbf{U}_{i,\ell,k}^{\top} \boldsymbol{\tau}_i^{(\ell)}$; `// Eq. (6)`
    $\mathbf{A}_{i,\ell} \leftarrow \mathbf{U}_{i,\ell,k}^{\top} \boldsymbol{\tau}_i^{(\ell)}$; $(\mathbf{P}_{i,\ell}, \boldsymbol{\Sigma}_i^{(\ell)}, \mathbf{V}_i^{(\ell)}) \leftarrow \text{SVD}(\mathbf{A}_{i,\ell})$;
    $\mathbf{U}_i^{(\ell)} \leftarrow \mathbf{U}_{i,\ell,k} \mathbf{P}_{i,\ell}$;
  end
  $\hat{\mathbf{U}}_\ell \leftarrow \text{polar}([\mathbf{U}_1^{(\ell)} \mid \cdots \mid \mathbf{U}_N^{(\ell)}])$; $\hat{\mathbf{V}}_\ell \leftarrow \text{polar}([\mathbf{V}_1^{(\ell)} \mid \cdots \mid \mathbf{V}_N^{(\ell)}])$;
  for $i = 1, \ldots, N$ **do**
    $\boldsymbol{\tau}_{\text{align},i}^{(\ell)} \leftarrow \hat{\mathbf{U}}_\ell^{(i)} \boldsymbol{\Sigma}_i^{(\ell)} (\hat{\mathbf{V}}_\ell^{(i)})^{\top}$; `// Eq. (9)`
    $\beta_i^{(\ell)} \leftarrow \left\| \boldsymbol{\tau}_i^{(\ell)} \right\|_{\text{F}} / (\left\| \boldsymbol{\tau}_{\text{align},i}^{(\ell)} \right\|_{\text{F}} + \epsilon)$; `// energy renorm`
  end
  $\mathbf{W}_{\text{merged}}^{(\ell)} \leftarrow \mathbf{W}_{\text{pre}}^{(\ell)} + \sum_i \beta_i^{(\ell)} \boldsymbol{\tau}_{\text{align},i}^{(\ell)}$; `// Eq. (10)`
end
foreach *1D parameter* p **do**
  $p_{\text{merged}} \leftarrow p_{\text{pre}} + \frac{1}{N} \sum_i \boldsymbol{\tau}_i[p]$;
end
return $\boldsymbol{\theta}_{\text{merged}}$;

from the expert forward pass. The merge then uses the same cross-expert polar alignment and energy renormalization as the shared-q full-Gram baseline.

Table B compares the shared-q full Gram with the expert-prefix cascade full Gram in the three-expert setup (Coding, Tool Use, Memory). The cascade variant outperforms the shared-q variant on three of the four Coding metrics and three of the four Memory metrics. It performs slightly worse on both Tool Use subtasks (BFCL Live and Non-Live). The cascade attains a macro-average of 69.48, compared with 68.93 for the shared-q variant, a gain of 0.55. Despite this gain, the cascade is substantially more expensive during Gram construction—for Qwen2.5-7B with L=28 blocks, it requires about 21$\times$ more forward computation per expert. Shared-q therefore remains a reasonable cost–performance trade-off and is used as the default PSA-G configuration in the main paper.

C.2 Token-Level Policy-Shift Analysis on RLVR-Training Calibration Data

We test whether the expert's suppression of base-sampled tokens differs between correct and incorrect trajectories on each expert's own training corpus.

We sample n=128 queries from the validation split of each expert's training corpus (CodeContests-train[3], ToolRL-train[4], HotpotQA-train-32K[5]), disjoint from the main evaluation benchmarks in Section 4 (no leakage). For each query, we generate a rollout using Qwen2.5-7B-Instruct and

[3] https://huggingface.co/datasets/Gen-Verse/CodeContests_train
[4] https://huggingface.co/datasets/emrecanacikgoz/ToolRL
[5] https://huggingface.co/datasets/BytedTsinghua-SIA/hotpotqa

Table A: **Two-stage incremental merging on Qwen2.5-7B.** (a) reports Stage 1 over Coding, Tool Use, and Memory. (b) reports Stage 2 after adding Search and Math. Both subtables use the same metric format. Coding reports LB and LCB ACC/UT. Tool Use reports BFCL Live and Non-Live AST. Memory reports average RULER SubEM. Stage 2 also reports SimpleQA ACC for Search and AIME24/25/26 mean@8 for Math. Avg. is the macro-average over the task groups available in each subtable. **Bold** marks the best score in each column, separately among the reference models and among the merged models; the highlighted row indicates our method.

(a) Stage 1 over three tasks: Coding, Tool Use, and Memory.

Method	Coding				Tool Use		Memory (Recurrent)				Avg.
	LB		LCB		Live	Non-Live	HotpotQA		SQuAD		
	ACC	UT	ACC	UT	AST	AST	7K	14K	32K	64K	
Reference Models											
Base	37.74	48.63	28.86	44.20	74.02	86.31	57.29	53.65	65.89	59.12	59.67
Coding Expert	**40.92**	**52.33**	**31.08**	**45.84**	73.80	86.44	58.33	52.08	64.58	57.81	60.29
Tool Expert	36.28	48.28	28.50	43.58	**77.05**	**88.06**	60.42	55.73	60.68	62.24	60.49
Memory Expert	37.26	48.66	29.88	45.28	75.50	86.27	**78.65**	**78.65**	**76.04**	**78.64**	66.38
Merging Baselines											
Task Arithmetic	36.72	48.33	29.97	44.62	77.13	88.37	67.45	60.68	72.13	69.53	63.37
TIES	38.09	49.66	31.57	46.35	76.46	88.62	74.74	75.52	76.07	76.27	66.54
DARE	40.23	51.34	**32.72**	47.62	73.58	85.73	75.78	76.56	74.22	77.34	66.20
TSV	38.74	50.00	31.50	46.55	**77.28**	**88.75**	75.52	73.96	75.17	73.97	66.46
STAR	39.01	50.10	31.65	46.91	75.72	88.23	75.78	73.55	73.44	74.22	66.05
Iso-CTS	37.89	48.24	31.19	46.12	77.05	88.65	63.28	62.50	64.06	65.62	62.53
RAM	39.26	50.36	31.24	46.79	76.31	88.08	73.70	76.82	72.63	75.52	66.26
RAM+	37.70	49.89	31.46	47.14	76.09	88.17	76.56	76.56	74.30	74.22	66.36
PSA-G (Ours)	**41.46**	**52.12**	32.64	**47.95**	76.83	**88.75**	**80.47**	**81.25**	**79.69**	**80.47**	**68.93**

(b) Stage 2 over five tasks, adding Search and Math.

Method	Coding		Tool Use		Memory	Search	Math			Avg.
	LB	LCB	Live	Non-Live	avg	SimpleQA	AIME24	AIME25	AIME26	
	ACC/UT	ACC/UT	AST	AST	SubEM	ACC	mean@8	mean@8	mean@8	
Reference Models										
Base	43.19	36.53	74.02	86.31	58.99	77.68	10.42	8.33	7.08	53.06
Coding Expert	**46.63**	**38.46**	73.80	86.44	58.20	73.82	12.92	8.33	6.25	52.77
Tool Expert	42.28	36.04	**77.05**	**88.06**	59.77	77.80	10.00	9.17	9.58	53.77
Memory Expert	42.96	37.58	75.50	86.27	**78.00**	74.32	9.58	5.83	7.50	56.22
Search Expert	38.03	34.44	61.50	74.30	55.28	**86.40**	10.83	7.08	7.50	50.86
Math Expert	34.87	34.03	52.40	44.60	61.52	78.28	**17.08**	**16.25**	**17.08**	47.91
Merging Baselines										
Task Arithmetic	41.77	38.28	75.87	83.12	71.68	79.00	14.17	13.33	10.00	56.54
TIES	43.88	38.14	75.72	86.77	78.71	81.80	15.00	7.50	8.33	58.61
DARE	43.39	39.23	59.73	79.08	79.49	79.00	12.50	11.25	7.92	55.95
TSV	39.47	35.93	77.13	87.71	74.61	82.00	14.17	12.08	11.25	57.85
STAR	41.27	36.98	75.06	87.02	78.12	82.20	13.33	12.92	11.25	58.60
Iso-CTS	41.79	36.79	73.13	85.54	64.65	76.40	11.67	9.58	7.08	53.82
RAM	41.78	37.71	**77.94**	88.06	73.44	82.00	13.75	10.83	7.08	57.75
RAM+	43.25	37.41	77.87	**88.48**	73.83	82.20	12.08	8.75	7.08	57.77
PSA-G (Ours)	**45.73**	**40.07**	76.24	86.50	**82.03**	**85.00**	**15.42**	**15.42**	**12.08**	**61.12**

Table B: **Shared-q full Gram vs. expert-prefix cascade full Gram on Qwen2.5-7B with three RLVR experts.** The column structure matches Stage 1 of Table A. The shared-q row reproduces the PSA-G (Ours) numbers from Stage 1. The highlighted row uses layer-specific token weights from expert-prefix cascade attribution.

Method	Coding				Tool Use		Memory (Recurrent)				Avg.
	LB		LCB		Live	Non-Live	HotpotQA		SQuAD		
	ACC	UT	ACC	UT	AST	AST	7K	14K	32K	64K	
PSA-G-shared	41.46	52.12	32.64	**47.95**	**76.83**	**88.75**	80.47	**81.25**	79.69	80.47	68.93
PSA-G-prefix	**42.82**	**53.89**	**32.78**	47.89	75.72	88.42	**81.25**	80.47	**82.81**	**83.59**	**69.48**

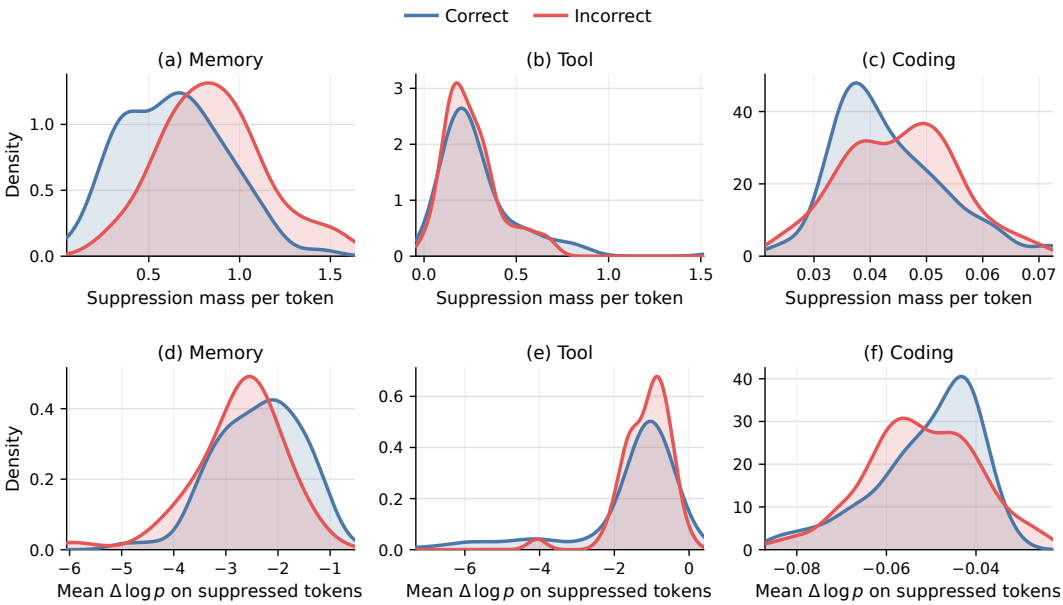


Figure A: **Per-trajectory policy-shift statistics, split by correctness** (n=128 trajectories per task). Top: suppression mass per token m. Bottom: mean log-probability difference $\bar{\Delta}$ on suppressed tokens. On memory tasks, incorrect trajectories show larger suppression mass and more negative $\bar{\Delta}$ than correct trajectories. On tool-use and coding tasks, the two distributions largely overlap.

teacher-force the resulting tokens under both the base model and the corresponding expert. We compute the per-token gap $\Delta_i(h)$ using Eq. (1). We call a token suppressed if the expert assigns it a lower log probability than the base model, i.e., $\Delta_i(h) < 0$. For a trajectory y with $|y|$ generated tokens, let $\mathcal{S}_i(y)$ be its set of suppressed tokens. We report the suppression mass per token $m = |y|^{-1} \sum_{h \in \mathcal{S}_i(y)} |\Delta_i(h)|$ and the mean log-probability difference on suppressed tokens $\bar{\Delta} = |\mathcal{S}_i(y)|^{-1} \sum_{h \in \mathcal{S}_i(y)} \Delta_i(h)$. Trajectories are graded by SubEM (memory), AST match (tool use), and execution match (coding).

Figure A shows that, on memory tasks, the expert suppresses base-sampled tokens more strongly on incorrect trajectories than on correct ones. On tool-use and coding tasks, the two distributions largely overlap. This memory-task result supports conditioning spectral truncation on activations at tokens with large policy shifts in memory-like tasks.

C.3 Policy-Shift Weighting versus Non-Policy-Shift Column Metrics

This analysis tests whether the gain of PSA-D comes from the policy-shift weights or from column weighting in general. We use the three-expert Qwen2.5-7B setting (Coding, Tool Use, Memory) of Table A(a). All variants keep the PSA-D pipeline and change only the token weight and the diagonal column metric used for truncation. Uniform uses uniform token weights and the identity metric, so

truncation reduces to standard SVD. RegMean-style uses uniform token weights and the diagonal of the unweighted activation second moment, following RegMean [Jin et al., 2023]. Fisher-diagonal uses uniform token weights and the empirical Fisher diagonal [Matena and Raffel, 2022]. PSA-D uses policy-shift token weights and the diagonal $\mathbf{D}_{i,\ell}$ from Eq. (7).

Table C shows that the RegMean-style diagonal is the strongest non-policy-shift metric, with an average of 67.39. Activation-weighted truncation errors are therefore a better criterion than raw parameter energy. Replacing the uniform token weights with policy-shift weights raises the average further to 68.58. Policy-shift weighting thus adds a benefit beyond activation-based column weighting.

Table C: **Policy-shift weighting versus non-policy-shift column metrics on Qwen2.5-7B with three RLVR experts.** All variants share truncation, polar alignment, and energy renormalization. Coding averages LB and LCB ACC and UT, Tool averages BFCL Live and Non-Live AST, and Memory averages the four RULER SubEM scores of Table A(a). Scores are mean ± standard deviation over three decoding seeds. **Bold** marks the best score in each column.

Variant	Token weight	Column metric	Coding	Tool	Memory	Avg.
Uniform	uniform	identity	41.00 ± 0.95	**82.77** ± 0.04	77.15 ± 0.90	66.97 ± 0.44
RegMean-style	uniform	activation diagonal	40.93 ± 0.43	82.32 ± 0.13	78.91 ± 0.51	67.39 ± 0.23
Fisher-diagonal	uniform	Fisher diagonal	37.48 ± 1.48	82.38 ± 0.04	75.00 ± 1.41	64.95 ± 0.68
PSA-D (Ours)	policy shift	$\mathbf{D}_{i,\ell}$ (Eq. (7))	**43.42** ± 0.08	82.64 ± 0.02	**79.69** ± 0.19	**68.58** ± 0.05

C.4 Sensitivity to the Energy Budget and the Calibration Set

Table D(a) varies the energy budget η_E of PSA-D in the same three-expert setting. The average changes by at most 0.82 points over $\eta_E \in \{0.6, 0.7, 0.8, 0.9\}$, and the default $\eta_E = 0.9$ gives the best average. Table D(b) calibrates all three experts with one shared query set instead of expert-specific queries. The shared set is either the union of the three task query sets or the task-agnostic Alpaca corpus [Taori et al., 2023]. Expert-specific calibration performs best, and both shared sets stay within 1.07 points of its average. The largest drop occurs on Memory with Alpaca calibration.

Table D: **Sensitivity of PSA-D to the energy budget and the calibration set on Qwen2.5-7B with three RLVR experts.** (a) uses the metrics of Table C. (b) uses the metrics of Table 5, where Coding averages LiveBench ACC and UT and Memory averages HotpotQA SubEM at 7K and 14K. Highlighted rows indicate the default configuration.

(a) Energy budget η_E.

η_E	Coding	Tool	Memory	Avg.
0.6	41.80	81.39	80.08	67.76
0.7	42.63	81.70	**80.47**	68.27
0.8	42.38	82.19	**80.47**	68.35
0.9	**43.42**	**82.64**	79.69	**68.58**

(b) Calibration set.

Calibration	Coding	Tool	Memory	Avg.
Pooled task queries	45.15	82.59	79.10	68.95
Alpaca (task-agnostic)	45.18	82.54	77.93	68.55
Expert-specific	**46.53**	**82.64**	**79.69**	**69.62**

D Experimental Details

D.1 Training Recipes for Qwen3-1.7B RLVR Experts

D.1.1 Instruction-Following Reinforcement Learning (IF-RL)

Training data and optimization pipeline. Our IF-RL training follows a simplified variant of the Nemotron-Cascade [Wang et al., 2025a] training pipeline. We use the publicly released RL instruction-following dataset[6]. We train in a single stage on instruction constraints from the IFEval taxonomy using the GRPO [Shao et al., 2024] objective function. For a given prompt q and a group of generated responses $\{o_i\}_{i=1}^{G}$, the reward for each response o_i is defined as

$$r_i = R_{\mathrm{IF}}(o_i), \tag{11}$$

[6] https://huggingface.co/datasets/nvidia/Nemotron-Cascade-RL-Instruction-Following

where $R_{\text{IF}}(o_i) \in \{0, 1\}$ is a binary reward from a rule-based instruction-following verifier that evaluates whether the response satisfies all specified constraints. Unlike Nemotron-Cascade, which combines R_{IF} with a sigmoid-scaled, group-normalized human preference reward $\hat{R}_{\text{RM}}$ to preserve alignment capabilities from RLHF, we use R_{IF} as the sole reward signal without a reward model. We apply dynamic filtering to ensure all prompts in each batch contribute effective gradients, stabilizing training throughout. To further improve training stability, we perform both policy learning and rollout generation in FP16 precision, following Qi et al. [2025].

Hyperparameters. We set the maximum response length to 8K tokens and use a batch size of 32 with 8 rollouts. The policy is optimized using AdamW [Loshchilov and Hutter, 2019] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a learning rate of 2×10^{-6}. During rollout generation, we sample with a temperature of 0.6.

D.1.2 Math RL

Training data and optimization pipeline. For mathematical reasoning, we use the publicly released RL math dataset from Nemotron-Cascade[7]. Rewards are assigned strictly based on answer correctness. The final answer is extracted from the `\boxed{}` expression following the `</think>` token and verified using the AceMath [Liu et al., 2025] rule-based verifier. Correct answers receive a reward of 1, and incorrect answers receive a reward of 0. To prevent language mixing during the reasoning process, we apply a code-switching penalty by assigning a reward of -1 whenever tokens from a language different from the original prompt's language are detected in the reasoning chain. Following Wang et al. [2025a], we adopt a two-stage training curriculum: the first stage sets the maximum response length to 24K tokens to stabilize the reasoning chain, and the second stage increases it to 32K tokens to accommodate longer reasoning traces. We apply dynamic filtering [Wang et al., 2025a] in the first stage to stabilize model accuracy during training. Both policy learning and rollout generation are performed in FP16 precision, following Qi et al. [2025].

Hyperparameters. We use a batch size of 64, sampling 8 responses per prompt with a temperature of 1.0 and a top-p value of 0.95. We adopt a learning rate of 1×10^{-6} for stage 1 and 2×10^{-6} for stage 2, using AdamW [Loshchilov and Hutter, 2019] with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The entropy loss coefficient is set to 0, and the KL loss coefficient is set to 0.002 for training stability.

D.2 Detailed Evaluation Protocols

D.2.1 Coding: LiveBench and LiveCodeBench

LiveBench. We evaluate on the Coding category of LiveBench [White et al., 2025], a contamination-free benchmark updated monthly with problems released strictly after the evaluation date cutoff. Tasks include LCB Generation (derived from LiveCodeBench), Code Completion, and Math Programming; all are graded by unit-test execution with no LLM judge.

LiveCodeBench. LiveCodeBench [Jain et al., 2024] collects competitive programming problems from LeetCode, Codeforces, and AtCoder published after a fixed date cutoff to prevent contamination. We use the Code Generation task, graded by executing model-generated code in a sandboxed Python environment subject to a per-problem time limit; a solution is correct only if all hidden test cases pass.

Shared configuration. We use the same evaluation setup for both benchmarks. We use sampling-based decoding with temperature$= 1.0$, top-p=0.95, and top-k disabled, generating k=16 samples per problem with `max_tokens`$= 10{,}000$. We report two metrics, each averaged over the 16 samples per problem. ACC is the fraction of problems whose generated program passes all unit tests, which equals the unbiased pass@1 estimate [Chen et al., 2021]. UT is the fraction of individual unit tests that the generated program passes. Table columns labeled ACC/UT report the mean of ACC and UT. All evaluations are zero-shot. The system prompt is fixed to "`You are a helpful assistant help user solve problems.`" and the user prompt instructs the model to reason prior to writing code, with `input()` and `print()` required for I/O.

[7] https://huggingface.co/datasets/nvidia/Nemotron-Cascade-RL-Math

D.2.2 Tool Use: BFCL v4

We evaluate function-calling ability using the Berkeley Function-Calling Leaderboard v4 (BFCL v4) [Patil et al., 2025]. Scoring follows the BFCL AST-based protocol, which verifies function names, argument names, values, and types against reference parses. The full category breakdown and per-category weights used to compute the Live and Non-Live scores are shown in Table E. The Live score is the average over all four Live subcategories, weighted by their instance counts (total: 1,351). The Non-Live score is the unweighted average over its four top-level categories (total: 1,150).

Table E: BFCL v4 category hierarchy and weights used to compute Live and Non-Live accuracy scores.

Split	Category	Subcategory (metric)	# Inst.	Weight
Live (Weighted, total 1,351)	Simple	Live Simple AST	258	$258/1351 \approx 19.1\%$
	Multiple	Live Multiple AST	1053	$1053/1351 \approx 78.0\%$
	Parallel	Live Parallel AST	16	$16/1351 \approx 1.2\%$
	Para. Multi.	Live Para. Multi. AST	24	$24/1351 \approx 1.8\%$
Non-Live (Unweighted, total 1,150)	Simple (avg. of 3, $= 550$)	Python Simple AST	400	$1/4 = 25\%$
		Java Simple AST	100	
		JavaScript Simple AST	50	
	Multiple	Multiple AST	200	$1/4 = 25\%$
	Parallel	Parallel AST	200	$1/4 = 25\%$
	Para. Multi.	Para. Multi. AST	200	$1/4 = 25\%$

D.2.3 Long-Context Memory: RULER

We assess long-context memory retention using the HotpotQA and SQuAD QA subtasks of RULER [Hsieh et al., 2024]. HotpotQA requires multi-hop reasoning, while SQuAD targets single-hop retrieval. Both are evaluated over long contexts padded with distractors. Distractor counts scale with context length: 50 at 7K, 100 at 14K, 200 at 32K, and 400 at 64K tokens. Correctness is determined by substring exact match (`judge_sub_em`). Decoding uses sampling with temperature$= 0.7$, top-p=0.95, and `max_tokens`$= 10{,}000$, with `max_input_tokens` capped at 120K tokens. All evaluations are zero-shot; the system prompt is applied via the tokenizer chat template.

D.2.4 Instruction Following: IFEval

IFEval [Zhou et al., 2023] consists of 541 prompts with 834 verifiable instructions across 25 categories including keyword constraints, length limits, formatting requirements, and stylistic conditions. Evaluation is fully rule-based without any LLM judge. We use greedy decoding (temperature$= 0.0$) with `max_tokens`$= 2{,}048$ and `max_model_length`$= 8{,}192$. The system prompt is applied via the tokenizer chat template. We report instruction-level strict accuracy (ACC), the fraction of the 834 instructions that the response satisfies under strict rule-based checking.

D.2.5 Mathematical Reasoning: AIME24, AIME25, and AIME26

AIME 2024. We evaluate on AIME I and II from 2024 (30 problems in total).[8] All problems require an exact integer answer in the range 0–999. We generate n=8 samples per problem with temperature$= 0.6$, top-p=0.95, top-k disabled, and `max_new_tokens`$= 34{,}816$. We report mean@8. Note that AIME 2024 problems may overlap with some models' pretraining data; we therefore also report AIME 2025 and AIME 2026 results below.

AIME 2025. AIME 2025[9] follows the same problem format as AIME 2024. We generate n=8 samples with `max_new_tokens`$= 34{,}816$ and report mean@8.

[8] https://artofproblemsolving.com/wiki/index.php/2024_AIME_I, https://artofproblemsolving.com/wiki/index.php/2024_AIME_II

[9] https://artofproblemsolving.com/wiki/index.php/2025_AIME_I, https://artofproblemsolving.com/wiki/index.php/2025_AIME_II

AIME 2026. AIME 2026[10] follows the same problem format as AIME 2024 and AIME 2025, providing the most recent and least contamination-prone evaluation of mathematical reasoning. We generate n=8 samples with `max_new_tokens`= 34,816 and report mean@8.

D.2.6 Search: SimpleQA

We evaluate on SimpleQA [Wei et al., 2024] using a fixed 500-question subsample (deterministic, identical across all models). The model acts as a tool-using agent with `search` (Google Serper) and `fetch_url` (Crawl4AI) tools for up to 6 tool-loop steps. Responses are graded by GPT-4o-mini using the official grading prompt. Questions whose tool loop fails are excluded from the ACC denominator, so the number of graded questions can differ slightly across models. We sample with temperature= 0.7 and `max_tokens`= 4,096, and report accuracy (ACC).

D.3 Model Checkpoints and Datasets

Table F lists the checkpoints, training framework, and calibration datasets used in Section 4.

Table F: Sources of the checkpoints, training framework, and same-task proxy calibration datasets.

Role	URL
Qwen2.5-7B checkpoints	
Base	https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
Coding	https://huggingface.co/Gen-Verse/ReasonFlux-Coder-7B
Tool use	https://huggingface.co/emrgnt-cmplxty/Qwen2.5-7B-Instruct-ToolRL-grpo-cold
Memory	https://huggingface.co/BytedTsinghua-SIA/RL-MemoryAgent-7B
Search	https://huggingface.co/yolay/SPEAR-SearchQA-Qwen2.5-7B
Math	https://huggingface.co/Elliott/LUFFY-Qwen-Instruct-7B
Qwen3-1.7B checkpoints and training framework	
Base	https://huggingface.co/Qwen/Qwen3-1.7B
Search	https://huggingface.co/Menlo/Lucy
RL training	https://github.com/verl-project/verl
Same-task proxy calibration datasets (Table 4)	
Coding	https://huggingface.co/datasets/codeparrot/apps
Tool use	https://huggingface.co/datasets/Salesforce/xlam-function-calling-60k
Memory	https://huggingface.co/datasets/voidful/2WikiMultihopQA
Search	https://huggingface.co/datasets/StonyBrookNLP/musique
Math	https://huggingface.co/datasets/openai/gsm8k

E Detailed Descriptions of Merging Baselines

This section provides detailed descriptions of all merging baselines used in our evaluation.

- **Task Arithmetic** [Ilharco et al., 2023] (ICLR 2023) constructs each task vector as the difference between fine-tuned and pretrained weights and combines them via uniform scalar-weighted summation, requiring no additional information beyond the model checkpoints.
- **TIES** [Yadav et al., 2023] (NeurIPS 2023) reduces parameter interference in three steps: trimming low-magnitude entries of each task vector, electing a consensus sign for each parameter, and averaging only those entries that agree with the elected sign.
- **DARE** [Yu et al., 2024] (ICML 2024) operates in the parameter space by applying a high-sparsity Bernoulli mask (e.g., 90% drop rate) to task vectors and rescaling surviving weights to preserve expected values before merging.
- **TSV** [Gargiulo et al., 2025] (CVPR 2025) extracts the top-k singular components of each task vector and applies polar decomposition to the concatenated singular bases, orthogonalizing task subspaces to reduce directional interference within the low-rank approximation.

[10] https://artofproblemsolving.com/wiki/index.php/2026_AIME_I, https://artofproblemsolving.com/wiki/index.php/2026_AIME_II

- **STAR** [Lee et al., 2025] (NAACL 2025) denoises individual task vectors by truncating small singular components and rescaling the survivors to restore the per-expert nuclear norm.
- **Iso-CTS** [Marczak et al., 2025] (ICML 2025) decomposes task vectors into common and task-specific subspaces, flattening the singular value spectrum within each subspace to its mean and discarding the original spectral energy distribution.
- **RAM** [Yuan et al., 2026] (arXiv 2026) disentangles shared and task-specific parameter updates in RL-trained agents, averaging the shared component while preserving and rescaling the task-specific component to counteract the signal dilution that arises when sparse RL updates are uniformly averaged.
- **RAM+** [Yuan et al., 2026] (arXiv 2026) extends RAM by introducing an additional `rescale_factor` hyperparameter that amplifies task-specific (non-overlapping) parameters in proportion to each expert's overlap ratio, preventing unique RL behaviors from being diluted during merging.

Relation to multi-teacher on-policy distillation. Multi-teacher on-policy distillation (OPD), used in DeepSeek-V4 [DeepSeek-AI, 2026], trains a single student on its own rollouts to match several domain-expert teachers. OPD therefore treats expert behavior as an optimization target and requires gradient-based retraining of the student. PSA addresses a different regime: it merges existing same-base checkpoints without retraining and uses expert behavior only to select which task-vector subspaces to preserve.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the three contributions (under-preservation of token-level policy shifts by SFT-oriented mergers; the proposed PSA method; empirical validation on Qwen2.5-7B and Qwen3-1.7B RLVR experts), all of which are supported by Sections 3–4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 includes a Limitations paragraph on calibration dependence, task-dependent gains, model scale, and overlapping expert subspaces. PSA also assumes that all experts are post-trained from the same base model (Section 2). Section 4.3 and Appendix C.4 study the sensitivity to the calibration source and the energy budget. The full-Gram variant PSA-G costs $O(R \cdot C \cdot |\mathcal{C}|)$ per layer, and Section 3.2 introduces the diagonal variant PSA-D to reduce this cost.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 3.2 states the policy-shift-weighted low-rank objective (Eq. (3)) and derives that its minimizer is obtained from a single SVD of the weighted output-perturbation matrix (Eqs. (4)–(6)). The derivation relies on the Eckart–Young–Mirsky theorem, and the polar alignment step in Section 3.3 uses the standard polar Procrustes solution [Schönemann, 1966, Higham, 1986].

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Algorithm 1 in Appendix A specifies the full PSA procedure, including the energy budget η_E and the stability constant ϵ. Training recipes for the two custom Qwen3-1.7B experts are detailed in Appendix D.1, and per-benchmark evaluation protocols (decoding, sampling, scoring) are given in Appendix D.2. The five Qwen2.5-7B experts and the Qwen3-1.7B search expert (Lucy) are publicly released checkpoints, linked in Appendix D.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The PSA merging code is not included with the paper, and we plan to release it separately. All evaluation benchmarks (LiveBench, LiveCodeBench, BFCL v4, RULER, IFEval, AIME 2024/25/26, SimpleQA) are public. Links to the public expert checkpoints and calibration datasets are provided in Appendices D.3 and C.2.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: PSA hyperparameters ($\eta_E = 0.9$, $\epsilon_0 = 10^{-2}$ for PSA-D, $\epsilon = 10^{-12}$) are stated in Section 3.2, Appendix A, and Algorithm 1. Calibration query sources are described in Section 4.3 and Appendices B and C.2. Training recipes for the IF and Math experts (optimizer, learning rate, batch size, sequence length, GRPO objective, dynamic filtering) are in Appendix D.1. Per-benchmark decoding settings (temperature, top-p, max tokens, sample count) and grading protocols are in Appendix D.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix C.3 reports means and standard deviations over three decoding seeds for PSA-D and the non-policy-shift column metrics, and the random-diagonal control in Table 5 is averaged over three seeds. In the main tables, the rows for our methods in Table 2 are means over three decoding seeds, and the other entries are single runs because each evaluation is costly.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper does not report hardware or wall-clock time. Section 3.2 gives the asymptotic cost of the truncation step for PSA-G and PSA-D, and Appendix C.1 reports the relative forward cost of the cascade variant. The merge is a one-time offline cost, and the merged model has the same per-token inference cost as the base model.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: This work studies retraining-free model merging on publicly released LLM checkpoints and standard public benchmarks. It does not involve human subjects, sensitive data, or deployment. We have reviewed and conform to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The paper proposes a general-purpose model merging algorithm for RLVR-trained experts. It does not introduce new model capabilities or training data, and does not have a direct path to negative societal impact beyond those already inherited from the underlying open-source LLMs and benchmarks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release new pretrained models, generative datasets, or scraped corpora. The merged checkpoints are derivable by applying the released algorithm to publicly available expert checkpoints and pose no incremental misuse risk beyond the base components.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original references for the base models (Qwen2.5-7B-Instruct [Yang et al., 2025b], Qwen3-1.7B [Yang et al., 2025a]), the expert checkpoints (ReasonFlux-Coder [Wang et al., 2025b], ToolRL [Qian et al., 2025], MemAgent [Yu et al., 2025a], SPEAR [Qin et al., 2025], LUFFY [Yan et al., 2025], Lucy [Dao et al., 2025]), and the evaluation benchmarks (LiveBench, LiveCodeBench, BFCL v4, RULER, IFEval, AIME 2024/25/26, SimpleQA). Appendices C.2, D.1, and D.3 give URLs for all checkpoints and for the training and calibration datasets (Nemotron-Cascade, CodeContests, ToolRL, HotpotQA, APPS, xLAM-60k, 2WikiMultihopQA, MuSiQue, GSM8K), and Appendix C.4 cites Alpaca [Taori et al., 2023]. All assets are publicly released under their respective licenses, and we use them for academic research within their terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or model checkpoints. The contribution is the PSA merging algorithm, and the two Qwen3-1.7B experts trained for our experiments are documented in Appendix D.1.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the subjects of the merging task itself (publicly released RLVR-trained Qwen2.5-7B and Qwen3-1.7B experts) rather than an auxiliary component of the core methodology. Any LLM assistance was limited to writing and editing and did not affect the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.