TASK VECTORS IN THE LOOP: ONLINE GEOMETRIC COORDINATION FOR MULTI-TASK RL FOR LLM

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement-learning (RL) post-training has advanced the reasoning and problem-solving capabilities of large language models, but extending these gains across heterogeneous tasks requires balancing cross-task transfer with task specialization. Mixed RL and post-hoc model merging approach this challenge from opposite directions: joint training allows transfer but risks interference, whereas independent training preserves specialization but leaves task-specific updates to be reconciled afterward. Rather than treating specialization and joint learning as competing choices, we use brief specialization to reveal task structure that guides joint optimization. We introduce **Task-Vector-Guided Reinforcement Learning (TVG-RL)**, which turns short task-specific warm-ups into a geometric prior for multi-task RL. Starting from a common initialization, TVG-RL extracts task vectors and decomposes their structure into shared and task-private subspaces. It then trains a single policy from the original base, using this decomposition to project and rescale subsequent RL gradients. Shared and private components receive distinct treatment, while a complementary component permits updates outside these task-conditioned shared–private subspaces. The resulting method requires neither fully trained experts nor a final model-merging stage. Experimentally, on `Qwen2.5-7B`, TVG-RL improves over mixRL on five of six main benchmarks, increasing KK accuracy by 34.25 percentage points, and leads all six merging baselines on four benchmarks. In the four-task `Qwen3-1.7B` setting, it achieves the highest mean scores on all five reported benchmarks among the evaluated generalist methods. TVG-RL also exceeds selected single-task experts and attains the highest mean accuracy on held-out GPQA-D in both settings. More broadly, these results suggest that task vectors need not remain a post-hoc tool for composing specialized models; they can instead serve as an optimization interface for directly learning stronger generalist policies.

1 INTRODUCTION

Reinforcement-learning (RL) post-training has become an effective way to strengthen the reasoning and problem-solving capabilities of large language models (LLMs) (Hugging Face, 2025; Guo et al., 2025; Team et al., 2025; Zhang et al., 2025). As these advances extend across mathematics, coding, and tool use, an important challenge is to develop these capabilities together within a single policy (Wang et al., 2026). Success in optimizing individual tasks does not, however, ensure that they can be learned equally well together. A generalist must retain the distinct abilities each task demands while allowing related tasks to reinforce one another. Effective multi-task post-training must therefore reconcile task specialization with beneficial transfer. Two natural strategies approach this challenge from opposite directions. *Mixed RL* learns from multiple task distributions within a shared training run, allowing cross-task transfer throughout optimization. Yet tasks can exert unequal influence on the shared policy (Ramesh et al., 2026), and their updates may conflict (Cai et al., 2026). *Post-hoc model merging* instead trains task specialists independently and combines their parameter changes afterward (Wang et al., 2026). This avoids direct competition during specialist training, but leaves the resulting updates to be reconciled at the integration stage. When all specialists must be trained from a common base, it also requires a separate RL run for each task. These strategies offer complementary benefits: joint training permits interaction, while separate training

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

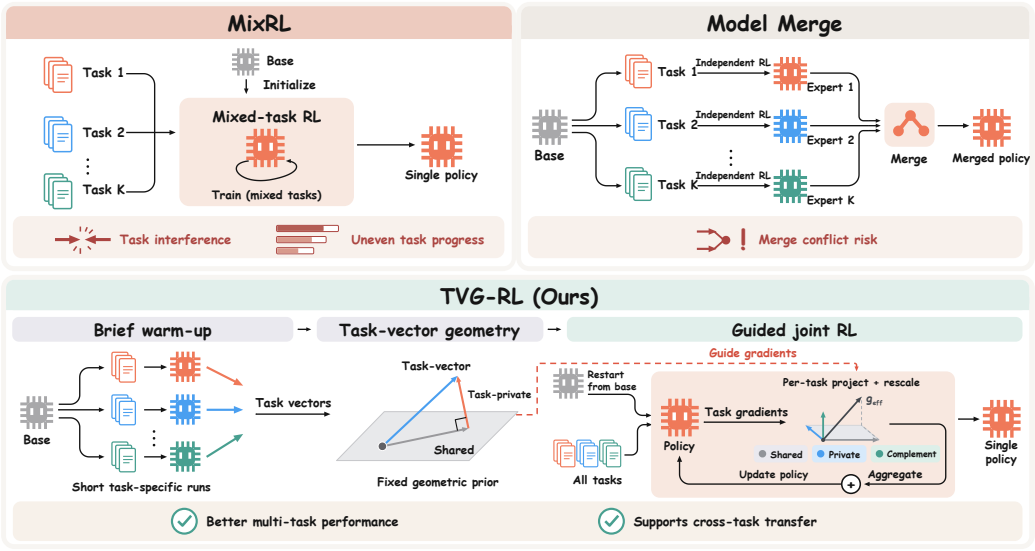


Figure 1: **Comparison of multi-task RL strategies.** *Mixed RL* jointly trains a shared policy but may suffer from task interference and uneven progress. *Model merging* combines independently trained experts and resolves their conflicts only after training. *TVG-RL* instead uses brief task-specific warm-ups to build a fixed shared–private geometric prior, then restarts from the base model and uses it to project and rescale per-task gradients before aggregation, while retaining a complementary component for updates outside the task-conditioned subspaces.

exposes task-specific learning structure. This suggests an alternative to choosing between them: use the structure revealed by separate training to improve joint optimization.

The task-vector perspective on model merging provides a starting point. A task vector records the parameter change induced by adapting a common model to a particular task (Ilharco et al., 2022). Subspace-based merging further shows the value of retaining both common and task-specific update structure when combining models (Marczak et al., 2025). This distinction also matters during joint RL: a direction need not be shared across tasks to be useful, and suppressing task differences indiscriminately can discard the specialization a generalist needs. Rather than treating model merging solely as a way to compose checkpoints, we draw on its representation of task structure to organize learning. Our central question is: **Can task-vector geometry guide how a generalist is learned, rather than only how specialists are combined?** Crucially, such guidance need not come from fully trained specialists. Short task-conditioned runs from the same initialization reveal how each task begins to modify the policy. Analysis under standard gradient descent connects early task vectors to local optimization directions (Zhou et al., 2025a). Motivated by this connection, we hypothesize that brief specialization can provide a useful geometric prior for subsequent multi-task RL. The purpose of these warm-ups is not to achieve expert-level performance, but to estimate which directions recur across tasks and which capture task-specific variation. Because this early structure is only an approximation, it should guide subsequent learning without fixing the policy's eventual search space.

We introduce **Task-Vector-Guided Reinforcement Learning (TVG-RL)** to instantiate this idea (Figure 1). TVG-RL first performs short task-specific warm-ups from a common base and estimates shared and task-private directions within each selected weight tensor. It then trains a single policy from the original base, projecting and rescaling each task's RL gradient before aggregation. Shared components capture dominant cross-task structure, while private components retain variation associated with individual tasks. A separately weighted complementary component permits updates outside these task-conditioned shared–private subspaces. The resulting method directly optimizes the shared policy and requires no terminal model-merging stage. Under fully sharded data parallelism

(Zhao et al., 2023), we preserve these tensor-wise projections by aggregating low-dimensional coefficients, without gathering full gradients for the projection operation.

We evaluate TVG-RL in a five-task setting with `Qwen2.5-7B` and a four-task setting with `Qwen3-1.7B`, spanning reasoning and preference optimization, with instruction following additionally included in the 7B setting. In the primary 7B experiment, TVG-RL improves over mixRL on HH, KK, Science, and IF, with comparable GSM8K accuracy. Its KK accuracy reaches 79.00%, exceeding mixRL and the strongest merging baseline by 34.25 and 16.00 percentage points, respectively. TVG-RL also achieves the highest mean scores among generalist methods on HH and Science, and surpasses the HH and KK experts while remaining comparable to the Science expert. In the 1.7B setting, TVG-RL achieves the highest mean score among the evaluated generalist methods on all five reported benchmarks. Both settings yield the highest mean accuracy on held-out GPQA-D. The measured online overhead on 7B is 3.1%–3.8%, excluding task-specific warm-ups and basis construction. Together, these results support using specialization as a source of structure for joint optimization, rather than only as a route to independently trained experts.

Our contributions are threefold:

- **Task vectors as training priors.** We use the task structure revealed by brief specialization to guide subsequent multi-task RL, extending the role of task-vector geometry from composing learned capabilities to informing how they are learned together.
- **Shared–private guidance for joint RL.** We develop tensor-wise gradient coordination that separately weights shared, task-private, and complementary components, training a single policy without completed experts or a final model merge. A sharding-compatible implementation preserves the projection through low-dimensional coefficient aggregation.
- **Multi-task performance and transfer.** We show that TVG-RL improves over mixRL in the primary 7B setting and over post-hoc merging across two backbones, with specialist-level performance on selected tasks and transfer to held-out GPQA-D.

2 Related Work

Model merging and task-vector geometry. Model merging combines specialized checkpoints into a single model without ensemble inference costs. Early approaches include weight averaging (Wortsman et al., 2022), task arithmetic (Ilharco et al., 2022), and interference-aware update editing such as TIES (Yadav et al., 2023) and DARE (Yu et al., 2024). More recent methods preserve higher-level task structure through subspace modeling, including Task Singular Vectors (Gargiulo et al., 2025), Iso-CTS (Marczak et al., 2025), adaptive projective optimization (Wei et al., 2025), and effective-rank preservation (Skorobogat et al., 2025). Their evaluation has also expanded to domain-specialized and in-the-wild LLM settings (He et al., 2026; Hitit et al., 2025). These works primarily use task structure to compose already learned updates, whereas analyses linking early task vectors to local optimization directions suggest another possibility: estimating useful task geometry before specialization is complete (Zhou et al., 2025a). These merging methods use task-vector geometry primarily to preserve capabilities when composing trained models. TVG-RL instead uses this geometry to guide how capabilities are learned: brief task-specific runs provide a shared–private prior for subsequent joint RL. Task specialization thus serves a different purpose: **revealing structure for optimization rather than supplying parameters for composition.**

Capability integration in RL-post-trained LLMs. Recent work investigates how to consolidate specialized LLM capabilities acquired through RL. M2RL compares joint multi-domain RL with independent specialization followed by model merging (Wang et al., 2026). RAM adapts merging to RL-trained agents by preserving task-specific updates that global averaging can dilute (Yuan et al., 2026). Beyond parameter merging, MOPD transfers multiple RL experts into a shared student through supervision on student-generated trajectories (Ma et al., 2026a). A broader comparison relates domain-dependent outcomes across mixing, merging, and distillation to task-vector geometry (Wu et al., 2026). However, a shared training mixture does not ensure balanced task progress (Ramesh et al., 2026), while choosing between mixing and expert integration does not itself specify how task updates should interact. TVG-RL bridges joint training and post-hoc integration by using brief task-specific warm-ups to reveal shared and task-private structure before the main RL

run. This structure then guides a single policy, promoting cross-task transfer while preserving task-specific progress.

Connecting merging with training. Model merging can also inform optimization rather than serve only as a terminal integration step. ATM alternates task-specific tuning and parameter merging (Zhou et al., 2025b), while WARP interleaves reward optimization with policy averaging and interpolation (Ramé et al., 2024). MergOPT incorporates merge-induced perturbations during expert fine-tuning to improve robustness to later composition (Yang et al., 2026), and Merge to Mix uses merged models as surrogates for selecting training-data mixtures (Tao et al., 2025). Closely related to geometry-guided training, CARE-RL extracts capability directions from previous LLM RL stages to modulate subsequent updates (Zhang et al., 2026). Rather than inserting a merging operation into training or preparing experts for later composition, TVG-RL treats task-vector geometry as an optimization prior for a single shared policy. Its goal is not to produce more mergeable endpoints, but to train a stronger generalist directly.

3 PRELIMINARIES

3.1 MULTI-TASK REINFORCEMENT LEARNING FOR LLMS

Multi-task reinforcement learning learns a shared policy across a collection of tasks sampled from a task distribution, with each task corresponding to a decision process (Yang et al., 2020). For autoregressive LLM post-training, we represent task k by a prompt distribution $\mathcal{D}_k$ and a scalar reward function $R_k(x, y)$ that evaluates response y to prompt x (Ramesh et al., 2026). A state consists of the prompt and generated prefix, and an action appends the next token. Tasks share these token-generation dynamics but can differ in their prompt distributions and reward rules. A single policy $\pi_\theta(y \mid x)$ serves all tasks, with task context supplied through the prompt.

Let $w = (w_1, \ldots, w_K)$ be a task distribution, where $w_k \geq 0$ and $\sum_k w_k = 1$. A standard multi-task return objective is

$$\max_\theta J_{\mathrm{MT}}(\theta) = \mathbb{E}_{k\sim w}\left[\mathbb{E}_{x\sim\mathcal{D}_k,\ y\sim\pi_\theta(\cdot|x)} R_k(x, y)\right]. \tag{1}$$

Here, w_k specifies the mixture over tasks, rather than a preference over multiple reward components for the same response. Reference-policy regularization is incorporated in the training surrogate below. In this work, we use a uniform task distribution throughout training, assigning equal weight to all K tasks, i.e., $w_k = 1/K$. In this work, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as the underlying RL objective; a brief formulation and implementation details are provided in Appendix A.

3.2 TASK VECTORS AND LOCAL OPTIMIZATION DIRECTIONS

A task vector represents task-specific adaptation as a parameter displacement from a common initialization (Ilharco et al., 2022). Let $\theta_{k,T}$ be the parameters after T adaptation steps on task k, starting from $\theta_{k,0} = \theta_{\mathrm{b}}$. The task vector and its use in task arithmetic are

$$\Delta_k^{(T)} = \theta_{k,T} - \theta_{\mathrm{b}}, \qquad \theta_{\mathrm{TA}} = \theta_{\mathrm{b}} + \lambda \sum_{k=1}^{K} \Delta_k^{(T)}, \tag{2}$$

where λ controls the scale of the merged displacement. This representation applies to intermediate checkpoints as well as fully adapted models.

Task vectors can also encode local optimization information (Zhou et al., 2025a). Under ordinary gradient descent on a fixed differentiable task loss F_k, with step size η, the displacement satisfies

$$\Delta_k^{(T)} = -\eta \sum_{j=0}^{T-1} \nabla F_k(\theta_{k,j}) \approx -\eta T \nabla F_k(\theta_{\mathrm{b}}), \tag{3}$$

where the approximation requires gradients to vary little along the adaptation trajectory. This fixed-loss gradient-descent relation does not establish an equivalence for RL with changing rollout distributions or adaptive optimizers. We therefore use RL task vectors as summaries of early task-conditioned adaptation, rather than exact estimates of policy gradients.

4 TASK VECTOR GUIDED REINFORCEMENT LEARNING

TVG-RL uses brief task-specific adaptation to construct a geometric prior for joint RL. It first estimates shared and private directions from warm-ups initialized at a common base, then restarts training from that base and uses the resulting subspaces to coordinate individual GRPO gradients. The prior is constructed independently for each selected weight tensor and remains fixed during joint training.

4.1 TASK GEOMETRY FROM BRIEF SPECIALIZATION

We estimate early task-specific adaptation directions using short, independent warm-ups rather than fully trained specialists. For each task k, we initialize from θ_{b} and run T_{w} GRPO updates on $(\mathcal{D}_k, R_k)$, obtaining $\theta_{k,T_{\mathrm{w}}}$ and the task vector $\Delta_k^{(T_{\mathrm{w}})}$ in Eq. (2). These warm-up checkpoints are used only to construct the geometric prior; they are neither merged nor used to initialize joint training. Because task vectors contain both cross-task structure and task-specific variation, we model shared and private subspaces following Marczak et al. (2025). As in prior tensor-wise task decomposition (Gargiulo et al., 2025), we construct these subspaces independently for selected parameter tensors. Let $\mathcal{I}$ index the selected tensors, with θ_ℓ denoting tensor ℓ and d_ℓ its number of scalar parameters. Its warm-up displacement is

$$\Delta_{k,\ell} = \mathrm{vec}(\theta_{k,T_{\mathrm{w}},\ell} - \theta_{\mathrm{b},\ell}) \in \mathbb{R}^{d_\ell}, \tag{4}$$

where vec flattens the weight tensor into a vector. We describe the construction and gradient transformation for a single selected tensor and omit ℓ below. Both operations are applied independently to every tensor in $\mathcal{I}$. For each task, we decompose the parameter space into three orthogonal components: a shared direction capturing dominant cross-task structure, a task-private direction preserving task-specific variation, and a complementary subspace containing the remaining directions.

Shared Direction To compare directions rather than displacement magnitudes, we normalize the nonzero task vectors and collect them into a matrix:

$$\widetilde{\Delta}_k = \frac{\Delta_k}{\|\Delta_k\|_2}, \qquad D = [\widetilde{\Delta}_1, \ldots, \widetilde{\Delta}_K] \in \mathbb{R}^{d \times K}. \tag{5}$$

This prevents a task from dominating the geometric estimate solely through a larger warm-up displacement. We use the leading left singular vector of D to define the shared direction:

$$D = USV^\top, \qquad u_{\mathrm{s}} = U_{:,1}, \qquad P_{\mathrm{s}} = u_{\mathrm{s}} u_{\mathrm{s}}^\top. \tag{6}$$

This direction maximizes total squared alignment with the normalized task vectors. In implementation, we obtain it from a unit leading eigenvector v_1 of the Gram matrix $G = D^\top D$ through $u_{\mathrm{s}} = Dv_1/\|Dv_1\|_2$. This is equivalent to the SVD construction but requires eigendecomposition only in the K-dimensional task space.

Private Direction To preserve each task's variation outside the shared direction, we remove the shared component and normalize the residual:

$$\begin{aligned} r_k &= (I - P_{\mathrm{s}})\widetilde{\Delta}_k, \\ u_k &= \frac{r_k}{\|r_k\|_2}, \qquad P_k = u_k u_k^\top. \end{aligned} \tag{7}$$

The normalization applies to nonzero residuals. By construction, $u_{\mathrm{s}}^\top u_k = 0$, although private directions from different tasks need not be mutually orthogonal. We retain one shared direction and one private direction per task for each tensor.

Complementary Subspace. The shared and private directions span only a portion of the parameter space. To retain the remaining degrees of freedom, we define the task-conditioned complementary projector as

$$P_{\perp,k} = I - P_{\mathrm{s}} - P_k. \tag{8}$$

The resulting subspace contains directions that are orthogonal to both the shared direction and the current task's private direction. It is defined separately for each task and therefore does not correspond to the orthogonal complement of the union of all task-private directions.

4.2 GEOMETRY-GUIDED MULTI-TASK OPTIMIZATION

The warm-up subspaces capture early task structure but need not span all directions useful for subsequent learning. Hard projection would remove every gradient component outside them. TVG-RL instead weights the private and complementary components while leaving the shared component unchanged, allowing the prior to guide learning without confining it to the retained directions.

Joint training starts from $\theta_0 = \theta_{\mathrm{b}}$. At iteration t, let $g_{k,t}$ denote the vectorized gradient with respect to the selected tensor, computed from a minibatch estimate of task k's GRPO loss in Eq. (15) at θ_t. We transform this gradient as

$$g_{k,t}^{\mathrm{eff}} = (P_{\mathrm{s}} + \alpha_{\mathrm{p}} P_k + \alpha_{\perp} P_{\perp,k})\, g_{k,t}, \tag{9}$$

where $\alpha_{\mathrm{p}} > 0$ is the private coefficient shared by all tasks and $\alpha_{\perp} \geq 0$ controls the complementary component. Taking $\alpha_{\perp} > 0$ retains gradient information outside the estimated shared and private subspaces.

Following the equal task weighting in Section 3.1, we aggregate the transformed gradients:

$$\overline{g}_t = \frac{1}{K} \sum_{k=1}^{K} g_{k,t}^{\mathrm{eff}}. \tag{10}$$

Each task gradient is transformed before aggregation, and the aggregate is supplied to the underlying optimizer. Setting $\alpha_{\mathrm{p}} = \alpha_{\perp} = 1$ recovers the unmodified mixed-task gradient. The jointly trained policy is the final output, with no subsequent model-merging stage.

The role of soft complementary weighting. The complementary component can contain both useful learning signal and sampling noise. Its weight therefore controls how much signal is retained alongside that noise. To characterize this tradeoff, condition on the current policy, rollout policy, and fixed subspaces, and omit t. Assume the task gradients g_k are conditionally independent, with means μ_k, covariances Σ_k, and finite second moments. Define $\mu = K^{-1} \sum_k \mu_k$ and $H = P_{\mathrm{s}} + \alpha_{\mathrm{p}}(I - P_{\mathrm{s}})$. The target $H\mu$ is the mean update obtained when private and complementary directions are both weighted by α_{p}. For a candidate complementary weight a, let $\overline{g}(a)$ denote Eq. (10) with $\alpha_{\perp} = a$, and define its gradient estimation error as $\mathcal{E}(a) = \mathbb{E}\|\overline{g}(a) - H\mu\|_2^2$.

Proposition 1 (Soft complementary weighting)**.** *Let the net complementary signal and its sampling variance be*

$$B = \left\| \frac{1}{K} \sum_{k=1}^{K} P_{\perp,k} \mu_k \right\|_2^2, \qquad V_{\perp} = \frac{1}{K^2} \sum_{k=1}^{K} \mathrm{tr}(P_{\perp,k} \Sigma_k). \tag{11}$$

For $a \in [0, \alpha_{\mathrm{p}}]$, the error decomposes as

$$\mathcal{E}(a) = C + (\alpha_{\mathrm{p}} - a)^2 B + a^2 V_{\perp}, \tag{12}$$

where C is the variance retained in the shared and private components, independent of a. If $B + V_{\perp} > 0$, the error is minimized at

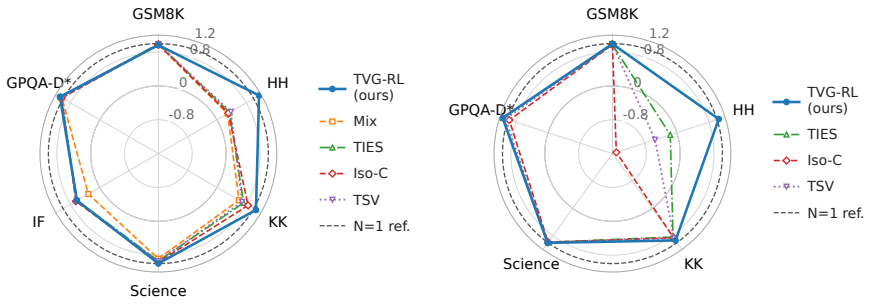
$$\alpha_{\perp}^{*} = \alpha_{\mathrm{p}} \frac{B}{B + V_{\perp}}. \tag{13}$$

When $B > 0$ and $V_{\perp} > 0$, this weight lies strictly between 0 and α_{p} and yields lower error than both hard projection ($a = 0$) and retaining the complementary component at the same strength as the private component ($a = \alpha_{\mathrm{p}}$).

The proposition gives a statistical basis for soft complementary updates: stronger noise favors greater attenuation, whereas stronger net signal favors retaining more of the complementary component. The optimal coefficient characterizes this tradeoff; it is not an additional online estimation step in TVG-RL. The proof is given in Appendix B.

4.3 IMPLEMENTATION

Explicitly constructing a projector for a tensor with d parameters would require $O(d^2)$ storage. Even with a low-rank representation, gathering full gradients for projection would introduce additional



(a) `Qwen2.5-7B` (b) `Qwen3-1.7B`

Figure 2: Expert-normalized scores of TVG-RL, mixRL (7B only), and the three highest-NAvg merging baselines. The dashed ring marks the benchmark-wise expert reference ($N = 1$). Radial coordinates use $r = N + 1.6$; ticks show signed N, retaining negative HH scores. GPQA-D* is held out from RL training. Full results: Appendix F.

memory and communication costs under fully sharded data parallelism (FSDP) (Zhao et al., 2023). We avoid dense projectors by evaluating each rank-one projection as $Pg = u(u^\top g)$. To avoid full-gradient gathers, the basis vector is sharded along the same parameter coordinates as the gradient. Each worker computes its local contribution to $u^\top g$, and an all-reduce sum recovers the complete coefficient before local reconstruction of the projected gradient shard. This requires reducing two scalar coefficients per tensor and task, one shared and one private. The complementary component is obtained as the residual. The resulting transformation matches the unsharded operator up to floating-point error. Appendix C provides the derivation and computational and storage costs. The algorithm is presented in Algorithm 1 in Appendix.

5 Experiments

We evaluate TVG-RL in terms of multi-task performance, generalization across benchmarks and model scales, and online training overhead.

5.1 Experimental Setup

Backbones and task configurations. Our primary five-task experiment uses `Qwen2.5-7B` with GSM8K(Cobbe et al., 2021), Full-HH-RLHF (HH)(Bai et al., 2022), KK Logic (KK)(Xie et al., 2024), Nemotron Science (Science)(NVIDIA Corporation, 2025b), and Nemotron Instruction Following (IF)(NVIDIA Corporation, 2025a). A complementary `Qwen3-1.7B` experiment uses GSM8K, HH, KK, Science, combining numerical, logical, and scientific reasoning. We also evaluate GPQA-Diamond (GPQA-D)(Rein et al., 2024), which is held out from RL training in both settings.

Comparisons and training. We evaluate six post-hoc merging methods: parameter averaging (Average) (Wortsman et al., 2022), TIES (Yadav et al., 2023), Breadcrumbs (Davari & Belilovsky, 2025), Iso-C, Iso-CTS (Marczak et al., 2025), and TSV (Gargiulo et al., 2025). These methods combine independently trained step-150 specialists. For the primary 7B experiment, we also compare with direct mixed-task GRPO (mixRL), which uses the same training mixture and online rollout budget as TVG-RL but aggregates task gradients without geometric transformations. Implementation details are given in Appendix D.

Evaluation. We report `avg@4`, averaging scores over four independently sampled responses per problem and then over problems, with temperature 1.0 and top-p 1.0. GSM8K, KK, and GPQA-D use answer accuracy; Science and IF use rule-based verifier scores; HH uses the raw Skywork reward-model score. Expert-normalized scores are used for the radar plots and reported in Appendix F; raw rewards are not averaged with accuracy or verifier scores. Further evaluation details are provided in Appendix E.

Table 1: Raw `avg@4` results on `Qwen2.5-7B` in the five-task setting (mean ± SE; higher is better). Non-HH means and SEs are multiplied by 100; HH uses raw reward units. Bold and underlined means indicate the best and second-best generalist results before rounding. GPQA-D is held out from RL training.

Method	GSM8K	HH	KK	Science	IF	GPQA-D
mixRL	89.44± 0.19	4.07± 0.05	44.75± 4.31	53.25± 0.24	13.09± 0.43	33.21± 0.66
AVER.	80.50± 0.59	-6.24± 0.43	27.25± 3.64	51.34± 1.30	11.99± 0.34	30.30± 1.44
TIES	**91.47**± 0.05	4.90± 0.05	54.00± 1.08	56.47± 0.20	25.90± 0.27	<u>34.34</u>± 0.71
BREAD.	72.97± 1.29	-13.44± 1.32	8.75± 1.65	45.46± 1.12	8.89± 0.33	27.40± 3.09
ISO-C	90.28± 0.28	4.38± 0.06	<u>63.00</u>± 2.27	<u>58.13</u>± 0.41	**27.42**± 0.30	33.33± 0.71
ISO-CTS	90.03± 0.22	2.74± 0.05	60.00± 1.00	57.96± 0.49	24.37± 0.88	33.08± 0.73
TSV	<u>90.39</u>± 0.34	<u>5.35</u>± 0.05	50.75± 3.17	56.98± 0.55	<u>26.42</u>± 0.47	33.21± 1.29
TVG-RL	88.82± 0.11	**16.21**± 0.05	**79.00**± 1.00	**60.03**± 0.46	26.12± 0.34	**34.85**± 0.65
Expert	92.00± 0.20	14.12± 0.03	74.25± 1.44	60.21± 0.47	41.99± 0.71	32.07± 0.76

Table 2: Raw `avg@4` results on `Qwen3-1.7B` in the four-task setting (mean ± SE; higher is better). Non-HH means and SEs are multiplied by 100; HH uses raw reward units. Bold and underlined means indicate the best and second-best generalist results before rounding. GPQA-D is held out from RL training.

Method	GSM8K	HH	KK	Science	GPQA-D
AVER.	76.57± 0.34	-18.75± 1.08	12.00± 1.58	43.90± 1.51	25.76± 0.74
TIES	83.26± 0.43	<u>-0.52</u>± 0.04	33.25± 0.75	54.79± 0.31	<u>31.44</u>± 0.43
BREAD.	71.15± 0.62	-23.15± 1.07	9.75± 1.89	39.87± 0.75	23.36± 1.21
ISO-C	83.19± 0.53	-4.79± 0.10	34.25± 0.63	54.64± 0.30	27.02± 0.60
ISO-CTS	81.97± 0.56	-7.02± 0.17	29.50± 0.87	53.93± 0.82	28.91± 0.73
TSV	<u>83.78</u>± 0.33	-1.72± 0.05	<u>34.75</u>± 1.38	<u>55.40</u>± 0.33	29.67± 0.43
TVG-RL	**84.78**± 0.10	**3.32**± 0.03	**37.25**± 0.25	**55.76**± 0.31	**31.69**± 0.98
Expert	85.24± 0.10	3.19± 0.02	39.75± 0.48	55.30± 0.13	28.03± 1.18

5.2 MAIN RESULTS

We first evaluate TVG-RL on the five-task 7B configuration, comparing its joint performance with mixRL and model merging, and its individual capabilities with the single-task experts.

TVG-RL improves multi-task performance while remaining competitive with specialists. As shown in Table 1, TVG-RL outperforms mixRL on HH, KK, Science, and IF, with comparable GSM8K accuracy. In particular, KK accuracy increases from 44.75% to 79.00%, and the IF score rises from 13.09 to 26.12. Against post-hoc merging, TVG-RL achieves the highest mean scores on HH, KK, and Science. Its KK accuracy exceeds the strongest merging baseline by 16.00 percentage points, while HH reward reaches 16.212 compared with the best merging result of 5.351. The Science score of 60.03 is also comparable to the single-task expert's 60.21, alongside above-expert HH and KK performance (Appendix F.2). These results show that TVG-RL can develop several specialist-level capabilities within one jointly trained policy, using only brief specialization to construct its training prior.

Specialization need not set the generalist's performance ceiling. TVG-RL exceeds the HH expert (16.212 versus 14.124) and the KK expert (79.00% versus 74.25%). Single-task experts are outcomes of finite-budget optimization, so their scores need not bound what a jointly trained policy can achieve. One plausible explanation is that heterogeneous tasks supply complementary learning signals, consistent with evidence that domain-diverse RL data can improve reasoning (Ma et al., 2026b). TVG-RL uses shared and task-private structure to coordinate these signals, while retaining

a complementary component that permits learning beyond the early task-conditioned subspaces. Its improvement over mixRL under the same training mixture is consistent with more effective use of cross-task information, in addition to the opportunities provided by data diversity itself.

5.3 Generalization Across Benchmarks and Backbones

We next examine transfer beyond the training-domain test sets and whether the results extend to the four-task 1.7B configuration.

TVG-RL also performs strongly on the 1.7B backbone. Table 2 shows that TVG-RL achieves the highest mean score on all five reported benchmarks, outperforming every evaluated merging method. It reaches 84.78% on GSM8K and 37.25% on KK, compared with 83.78% and 34.75% for the strongest merging results, respectively; HH reward increases to 3.321 from the best merging score of -0.518. TVG-RL also remains close to the GSM8K expert (84.78% versus 85.24%) and slightly exceeds the HH and Science experts. On held-out GPQA-D, its 31.69% accuracy again exceeds the best expert associated with this four-task configuration (28.03%). Together, these results extend the evidence for TVG-RL beyond the primary 7B configuration, across both training-domain capabilities and held-out reasoning.

Transfer extends to a held-out benchmark. In Table 1 and 2, on GPQA-D, TVG-RL achieves the highest mean accuracy in the 7B comparison: 34.85%, versus 33.21% for mixRL, 34.34% for the strongest merging baseline, and 32.07% for the best single-task expert. Because GPQA-D is not used in RL training, this result supports transfer of the jointly learned capabilities to an additional reasoning benchmark. Figure 2 provides a complementary view of these results: TVG-RL's strongest gains on 7B are visible on HH and KK, while the 1.7B profile leads the displayed merging baselines on every axis.

5.4 Online Cost of Geometric Coordination

We evaluate the online efficiency of TVG-RL against mixRL under the same five-task 7B training configuration. As shown in Figure 3 in Appendix, TVG-RL introduces only 3.1%–3.8% additional step time, with the task-gradient projections accounting for approximately 16.1 seconds per step (2.7%–2.9% of total TVG-RL step time). Excluding projection, the actor-update times of TVG-RL and mixRL are nearly identical. Over 150 joint-training steps, the projections correspond to approximately 0.67 wall-clock hours (5.37 GPU-hours on eight GPUs). These measurements characterize the online overhead only and exclude task-specific warm-ups and basis construction.

6 Conclusion

We presented TVG-RL, which turns brief task-specific specialization into a geometric prior for multi-task RL. By coordinating shared, task-private, and complementary gradient components, TVG-RL trains a single policy without completed experts or a final model merge. On `Qwen2.5-7B`, it improves over mixRL on most main benchmarks, outperforms the evaluated merging methods on HH, KK, and Science, and matches or exceeds specialists on selected tasks. The four-task `Qwen3-1.7B` results extend these gains to a second backbone, with the highest mean scores on all five reported benchmarks among the generalist methods. TVG-RL also achieves the highest mean accuracy on held-out GPQA-D in both settings. Its measured online overhead on 7B is 3.1%–3.8%, excluding warm-up training and basis construction. Task vectors can therefore serve not only to compose trained capabilities, but also to improve how a generalist acquires them.

Our experiments focus on two model backbones and a compact set of heterogeneous RL tasks, and broader evaluations across model scales, domains, and agentic settings would further test the generality of TVG-RL. The current method constructs its geometric prior once from brief warm-ups and keeps it fixed throughout joint training. This design is simple and incurs little online overhead, but future work could investigate whether periodically updating the task geometry better captures how task relationships evolve during training. Moreover, TVG-RL currently uses a compact shared–private decomposition for each selected tensor; richer low-rank subspaces and adaptive component weights may capture more complex task interactions.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the paper and pre-coding. We have not used generative AI tools for generating the research idea and designing the algorithm. We have reviewed all AI-assisted work. For example, the polished text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We confirm that all authors of this submission have read and agree to abide by the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The code is provided in the supplementary material to enable the reproduction of our results.

REFERENCES

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

Min Cai, Yu Liang, Longzheng Wang, Yan Wang, Yueyang Zhang, Long Xia, Zhiyuan Sun, Xi Ye, and Daiting Shi. Advancing general-purpose reasoning models with modular gradient surgery. *arXiv preprint arXiv:2602.02301*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 270–287, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73226-3.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705. IEEE, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized llms. *Advances in Neural Information Processing Systems*, 38, 2026.

Oğuz Kağan Hitit, Leander Girrbach, and Zeynep Akata. A systematic study of in-the-wild model merging for large language models. *arXiv preprint arXiv:2511.21437*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, et al. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026a.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *Advances in Neural Information Processing Systems*, 38:56596–56618, 2026b.

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. *arXiv preprint arXiv:2502.04959*, 2025.

NVIDIA Corporation. Nemotron-RL-instruction_following, 2025a. URL `https://huggingface.co/datasets/nvidia/Nemotron-RL-instruction_following`. Hugging Face dataset, accessed 2026-09-26.

NVIDIA Corporation. Nemotron-RL-knowledge-mcqa, 2025b. URL `https://huggingface.co/datasets/nvidia/Nemotron-RL-knowledge-mcqa`. Hugging Face dataset, accessed 2026-09-26.

Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. Warp: On the benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2406.16768*, 2024.

Shyam Sundhar Ramesh, Xiaotong Ji, Matthieu Zimmer, Sangwoong Yoon, Zhiyong Wang, Haitham Bou Ammar, Aurelien Lucchi, and Ilija Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks. *arXiv preprint arXiv:2602.05547*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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

Ronald Skorobogat, Karsten Roth, and Mariana-Iuliana Georgescu. Subspace-boosted model merging. *arXiv preprint arXiv:2506.16506*, 2025.

Zhixu Silvia Tao, Kasper Vinken, Hao-Wei Yeh, Avi Cooper, and Xavier Boix. Merge to mix: Mixing datasets via model merging. *arXiv preprint arXiv:2505.16066*, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Haoqing Wang, Xiang Long, Ziheng Li, Yilong Xu, Tingguang Li, and Yehui Tang. To mix or to merge: Toward multi-domain reinforcement learning for large language models. *arXiv preprint arXiv:2602.12566*, 2026.

Yongxian Wei, Anke Tang, Li Shen, Zixuan Hu, Chun Yuan, and Xiaochun Cao. Modeling multi-task model merging as adaptive projective gradient descent. *arXiv preprint arXiv:2501.01230*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Siye Wu, Kai Yang, Yuchen Cai, Xin Xu, Peng-Yuan Wang, Jiaxuan Wang, Jiashun Liu, Jiafei Lyu, Yangkun Chen, Saiyong Yang, et al. Consolidating rlvr capabilities across domains: A deep dive into fusion paradigms. *arXiv preprint arXiv:2608.27409*, 2026.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. 2024. URL https://arxiv.org/abs/2410.23123.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Enneng Yang, Qun Yang, Peng Wang, Anke Tang, Guibing Guo, Xiaochun Cao, and Li Shen. Mergopt: A merge-aware optimizer for robust model merging. In *International Conference on Learning Representations*, volume 2026, pp. 48400–48425, 2026.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Linyan 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*, volume 2, pp. 21, 2024.

Xiangchi Yuan, Dachuan Shi, Chunhui Zhang, Zheyuan Liu, Shenglong Yao, Soroush Vosoughi, and Wenke Lee. Behavior knowledge merge in reinforced agentic models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33007–33028, 2026.

Chong Zhang, Yue Deng, Xiang Lin, Bin Wang, Dianwen Ng, Hai Ye, Xingxuan Li, Yao Xiao, Zhanfeng Mo, Qi Zhang, et al. 100 days after deepseek-r1: A survey on replication studies and more directions for reasoning language models. *arXiv preprint arXiv:2505.00551*, 2025.

Rui Zhang, Xinle Wu, and Yao Lu. Care-rl: Capability-aware reinforcement learning for mitigating cross-domain conflicts. *arXiv preprint arXiv:2606.00609*, 2026.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Giuseppe Alessio D'Inverno, Fabrizio Silvestri, and Emanuele Rodolà. On task vectors and gradients. *arXiv preprint arXiv:2508.16082*, 2025a.

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Fabrizio Silvestri, and Emanuele Rodolà. Atm: Improving model merging by alternating tuning and merging. In *International Conference on Image Analysis and Processing*, pp. 192–201. Springer, 2025b.

A GRPO FORMULATION

GRPO constructs advantages from within-prompt reward comparisons without learning a separate value function Shao et al. (2024). For $x \sim \mathcal{D}_k$, sample N responses $\{y_i\}_{i=1}^N$ from the rollout policy $\pi_{\theta_{\text{old}}}$ and assign rewards $r_i = R_k(x, y_i)$. For outcome supervision, the advantage and token-level probability ratio are

$$\widehat{A}_i = \frac{r_i - \bar{r}}{\sigma_r + \delta}, \qquad \rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}, \tag{14}$$

where $\bar{r}$ and σ_r are the mean and standard deviation of rewards within this response group, and $\delta > 0$ ensures numerical stability. The same $\widehat{A}_i$ is used for all tokens of response y_i. The task-specific loss is

$$-\mathbb{E}_{\substack{x\sim\mathcal{D}_k \\ \{y_i\}_{i=1}^N \sim \pi_{\theta_{\text{old}}}(\cdot|x)}} \left[\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \Big(\min\Big\{ \rho_{i,t}\widehat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i \Big\} - \beta \widehat{d}_{i,t} \Big) \right], \tag{15}$$

where ϵ is the clipping threshold and $\beta \geq 0$ weights the KL penalty relative to a reference policy π_{ref}. The standard sampled penalty is $\widehat{d}_{i,t} = z_{i,t} - \log z_{i,t} - 1$, with $z_{i,t} = \frac{\pi_{\text{ref}}(y_{i,t}|x, y_{i,<t})}{\pi_\theta(y_{i,t}|x, y_{i,<t})}$. Writing $g_k = \nabla_\theta \widehat{\mathcal{L}}_k^{\text{GRPO}}(\theta)$ for the corresponding minibatch gradient, conventional mixed-task training combines task gradients as $g_{\text{mix}} = \sum_{k=1}^K w_k g_k$.

B ANALYSIS OF SOFT COMPLEMENTARY WEIGHTING

B.1 PROOF OF PROPOSITION 1

All expectations and covariances are conditional on the current policy, rollout policy, and fixed geometric prior. We analyze one selected tensor and suppress its index. Let

$$A_k(a) = P_{\text{s}} + \alpha_{\text{p}} P_k + a P_{\perp,k}, \qquad H = P_{\text{s}} + \alpha_{\text{p}}(I - P_{\text{s}}). \tag{16}$$

The three task-conditioned projectors are mutually orthogonal and sum to the identity. Consequently,

$$A_k(a) = H - (\alpha_{\text{p}} - a) P_{\perp,k}. \tag{17}$$

The aggregate gradient is $\overline{g}(a) = K^{-1} \sum_k A_k(a) g_k$, with $\mu = K^{-1} \sum_k \mu_k$. Its mean therefore satisfies

$$\mathbb{E}[\overline{g}(a)] - H\mu = -(\alpha_{\text{p}} - a)e, \qquad e = \frac{1}{K} \sum_{k=1}^{K} P_{\perp,k} \mu_k. \tag{18}$$

Thus, the squared bias is $(\alpha_{\text{p}} - a)^2 B$, where $B = \|e\|_2^2$.

By conditional independence across task gradient estimates,

$$\text{Cov}(\overline{g}(a)) = \frac{1}{K^2} \sum_{k=1}^{K} A_k(a) \Sigma_k A_k(a). \tag{19}$$

Orthogonality and idempotence give $A_k(a)^2 = P_{\text{s}} + \alpha_{\text{p}}^2 P_k + a^2 P_{\perp,k}$. Using cyclic invariance of the trace,

$$\text{tr}\,\text{Cov}(\overline{g}(a)) = C + a^2 V_\perp, \tag{20}$$

where

$$C = \frac{1}{K^2} \sum_{k=1}^{K} \text{tr}\big[(P_{\text{s}} + \alpha_{\text{p}}^2 P_k)\Sigma_k\big], \qquad V_\perp = \frac{1}{K^2} \sum_{k=1}^{K} \text{tr}(P_{\perp,k}\Sigma_k). \tag{21}$$

This calculation allows arbitrary within-task noise covariance; it does not require isotropy or independence between projected components of the same task gradient.

Combining squared bias and variance yields

$$\begin{aligned} \mathcal{E}(a) &= \|\mathbb{E}[\overline{g}(a)] - H\mu\|_2^2 + \text{tr}\,\text{Cov}(\overline{g}(a)) \\ &= C + (\alpha_{\text{p}} - a)^2 B + a^2 V_\perp. \end{aligned} \tag{22}$$

When $B + V_\perp > 0$, completing the square gives

$$\mathcal{E}(a) = C + \alpha_{\mathrm{p}}^2 \frac{BV_\perp}{B + V_\perp} + (B + V_\perp)\left(a - \alpha_{\mathrm{p}} \frac{B}{B + V_\perp}\right)^2. \tag{23}$$

The unique minimizer is $\alpha_\perp^* = \alpha_{\mathrm{p}} B/(B + V_\perp)$, which belongs to $[0, \alpha_{\mathrm{p}}]$. If $B > 0$ and $V_\perp > 0$, it is strictly inside the interval. In particular, the errors of hard projection and full complementary retention satisfy

$$\begin{aligned} \mathcal{E}(0) - \mathcal{E}(\alpha_\perp^*) &= \alpha_{\mathrm{p}}^2 \frac{B^2}{B + V_\perp} > 0, \\ \mathcal{E}(\alpha_{\mathrm{p}}) - \mathcal{E}(\alpha_\perp^*) &= \alpha_{\mathrm{p}}^2 \frac{V_\perp^2}{B + V_\perp} > 0. \end{aligned} \tag{24}$$

Here, $a = 0$ removes the complementary component, whereas $a = \alpha_{\mathrm{p}}$ weights it identically to the private component. If $B = V_\perp = 0$, the error is constant in a. This completes the proof.

B.2 A Common Coefficient across Selected Tensors

The same characterization applies when one complementary coefficient is used across all selected tensors. Let B_ℓ, $V_{\perp,\ell}$, and C_ℓ be the quantities above for tensor ℓ. The squared Euclidean error of the concatenated guided gradient is the sum of its tensor-wise errors. Hence, with $B_{\mathrm{all}} = \sum_{\ell \in \mathcal{I}} B_\ell$, $V_{\mathrm{all}} = \sum_{\ell \in \mathcal{I}} V_{\perp,\ell}$, and $C_{\mathrm{all}} = \sum_{\ell \in \mathcal{I}} C_\ell$,

$$\mathcal{E}_{\mathrm{all}}(a) = C_{\mathrm{all}} + (\alpha_{\mathrm{p}} - a)^2 B_{\mathrm{all}} + a^2 V_{\mathrm{all}}. \tag{25}$$

Its minimizer is $\alpha_{\mathrm{p}} B_{\mathrm{all}}/(B_{\mathrm{all}} + V_{\mathrm{all}})$ when the denominator is positive. No independence assumption across tensors is needed, since their squared errors add over disjoint parameter coordinates.

C Exact Tensor-Wise Projection with Sharded Gradients

This appendix derives a sharded implementation of the gradient transformation in Eq. (9). Exactness refers to equivalence with the corresponding unsharded tensor-wise operator in exact arithmetic, not to the accuracy of the warm-up geometry as a model of future task updates.

C.1 Low-Rank Evaluation of the Update

Consider one selected tensor and one task k; we omit the tensor and iteration indices. Let $g \in \mathbb{R}^d$ be the task gradient. Write $U_{\mathrm{s}} \in \mathbb{R}^{d \times q_{\mathrm{s}}}$ and $U_k \in \mathbb{R}^{d \times q_k}$ for orthonormal bases of the shared and task-private subspaces, respectively, with $U_{\mathrm{s}}^\top U_k = 0$. The construction in Section 4.1 uses $q_{\mathrm{s}} = q_k = 1$. Substituting $P_{\perp,k} = I - P_{\mathrm{s}} - P_k$ gives

$$g^{\mathrm{eff}} = \alpha_\perp g + (1 - \alpha_\perp) U_{\mathrm{s}}(U_{\mathrm{s}}^\top g) + (\alpha_{\mathrm{p}} - \alpha_\perp) U_k(U_k^\top g). \tag{26}$$

Thus, the transformation requires only the coefficient vectors $c_{\mathrm{s}} = U_{\mathrm{s}}^\top g$ and $c_k = U_k^\top g$ and their reconstruction through the bases. Neither a dense $d \times d$ projector nor an explicit basis for the complementary subspace is needed.

C.2 Coefficient Aggregation across Shards

FSDP distributes gradients and other model states across workers Zhao et al. (2023). Suppose the synchronized gradient of the selected tensor is partitioned across R ranks, and partition the rows of each basis using the same parameter-coordinate ranges:

$$g = \begin{bmatrix} g^{(0)} \\ \vdots \\ g^{(R-1)} \end{bmatrix}, \qquad U_a = \begin{bmatrix} U_a^{(0)} \\ \vdots \\ U_a^{(R-1)} \end{bmatrix}, \qquad a \in \{\mathrm{s}, k\}. \tag{27}$$

Here, $g^{(r)} \in \mathbb{R}^{d_r}$, $U_a^{(r)} \in \mathbb{R}^{d_r \times q_a}$, and $\sum_r d_r = d$. For each basis, rank r first computes

$$c_a^{(r)} = (U_a^{(r)})^\top g^{(r)}. \tag{28}$$

An all-reduce with summation over the sharding group then produces

$$c_a = \sum_{r=0}^{R-1} c_a^{(r)} = \sum_{r=0}^{R-1} (U_a^{(r)})^\top g^{(r)} = U_a^\top g. \tag{29}$$

Each rank reconstructs its effective gradient shard as

$$\begin{aligned} g^{\mathrm{eff},(r)} = \alpha_\perp g^{(r)} + (1-\alpha_\perp) U_{\mathrm{s}}^{(r)} c_{\mathrm{s}} \\ + (\alpha_{\mathrm{p}} - \alpha_\perp) U_k^{(r)} c_k. \end{aligned} \tag{30}$$

Equivalence to the unsharded operator. For either basis U_a, Eqs. (29) and (27) imply

$$\mathrm{concat}_{r=0}^{R-1}(U_a^{(r)} c_a) = U_a c_a = U_a U_a^\top g. \tag{31}$$

Combining the shared, private, and complementary terms therefore yields

$$\mathrm{concat}_{r=0}^{R-1}(g^{\mathrm{eff},(r)}) = (P_{\mathrm{s}} + \alpha_{\mathrm{p}} P_k + \alpha_\perp P_{\perp,k}) g. \tag{32}$$

Hence, sharding does not alter the tensor-wise transformation in Eq. (9). In floating-point arithmetic, reduction order and numerical precision can introduce discrepancies.

Why shard-local projection is insufficient. Computing $U_a^{(r)}(U_a^{(r)})^\top g^{(r)}$ independently on each rank retains only the diagonal blocks of the full operator. The correct output on rank r is instead

$$(U_a U_a^\top g)^{(r)} = \sum_{j=0}^{R-1} U_a^{(r)} (U_a^{(j)})^\top g^{(j)}, \tag{33}$$

which also depends on gradient coordinates held by other ranks. Coefficient aggregation retains these cross-shard contributions without constructing the corresponding matrix blocks.

C.3 Cost and Integration Requirements

For one task and one tensor, let $q = q_{\mathrm{s}} + q_k$. Computing the coefficients and reconstructing the projected shard requires $O(d_r q)$ local arithmetic and access to $O(d_r q)$ basis entries. The coefficient data to be reduced has q entries, rather than d gradient entries; this is the collective payload size, not a claim about total network traffic or collective-call count. If all task bases are kept resident, their storage on rank r is $O(d_r(q_{\mathrm{s}} + \sum_{k=1}^{K} q_k))$.

With one shared and one private direction, two coefficients are reduced per tensor and task. Processing all 196 selected tensors therefore involves 392 coefficient entries per task. This count does not imply that the implementation fuses them into one collective operation. It excludes ordinary FSDP communication, rollout generation, warm-up training, and optimizer costs.

The equivalence requires coordinate-aligned basis and gradient shards for the same task and synchronized gradient. The coefficient reduction is a sum over disjoint parameter coordinates, not an additional average over data replicas. Each rank must retain its slice of the globally normalized basis rather than renormalizing the slice locally. Padding or flattened-parameter layouts must preserve this coordinate correspondence. Once reconstructed, each task's effective gradient can be included in the weighted update in Eq. (10). These requirements preserve the projection operator; they do not establish that a subsequent adaptive-optimizer update lies in the same subspace.

D Experimental and Implementation Details

This section provides the complete training configuration for the experiments described in Section 5.1. TVG-RL and Mix are both implemented using the `verl` framework and trained on eight NVIDIA A800 GPUs. They use the same base model, training data, task sampling scheme, number of rollouts, optimizer, and number of training steps. Their only algorithmic difference is that TVG-RL applies a task-vector-guided transformation to each task gradient before aggregation, whereas Mix directly aggregates the original task gradients.

Table 3: Datasets used for five-task joint training.

Dataset	Training Size	Test Size
GSM8K	7,473	1,319
Full-HH-RLHF	112,052	12,451
KK Logic	900	100
Nemotron Science	19,200	1,024
Nemotron Instruction Following	19,200	1,024

D.1 Training Data

The datasets and data sizes used for five-task joint training are listed in Table 3. Each global step samples 64 prompts from each task. Training runs for 150 global steps, resulting in 9,600 sampled prompts per task. These numbers count sampling events during training and do not require all sampled examples to be unique.

D.2 TVG Configuration

The TVG geometric prior is constructed from the step-50 checkpoints of five short-horizon task-specific training runs. For task k, its task vector is defined as

$$\Delta_k^{50} = \theta_k^{50} - \theta_0, \tag{34}$$

where θ_0 denotes the common `Qwen2.5-7B` initialization and θ_k^{50} is the step-50 task-specific checkpoint for task k. These checkpoints are used only to construct the fixed task subspaces; the formal TVG-RL run is initialized again from θ_0.

We construct the TVG bases for the query, key, value, output, gate, up, and down projection matrices in each of the 28 Transformer layers, covering 196 weight tensors in total. The coefficients assigned to the shared, task-private, and complementary components are

$$\alpha_s = 1, \qquad \alpha_p = 5, \qquad \alpha_\perp = 0.05. \tag{35}$$

For task k and selected tensor ℓ, we first apply the TVG transformation defined in the main text to obtain $\widehat{\mathbf{g}}_{k,t,\ell}$. We then restore the norm of the pre-transformation gradient using

$$c_{k,t,\ell} = \frac{\|\mathbf{g}_{k,t,\ell}\|_2}{\|\widehat{\mathbf{g}}_{k,t,\ell}\|_2 + \epsilon}, \tag{36}$$

and define

$$\widetilde{\mathbf{g}}_{k,t,\ell} = c_{k,t,\ell}\widehat{\mathbf{g}}_{k,t,\ell}. \tag{37}$$

The transformed task gradients are aggregated using equal task weights:

$$\mathbf{g}_t^{\text{joint}} = \sum_{k=1}^{5} w_k \widetilde{\mathbf{g}}_{k,t}, \qquad w_k = 0.2. \tag{38}$$

The aggregated gradient is clipped once and used for a single AdamW update; no additional TVG projection is applied after aggregation. Gradients of parameters not covered by the TVG bases remain unchanged.

D.3 Training Hyperparameters

The complete training configuration is summarized in Table 4. Each task contributes 64 prompts per global step, and 16 responses are sampled for each prompt. Consequently, each global step contains 320 prompts and 5,120 responses.

The maximum prompt and completion lengths used for each task are listed in Table 5.

Table 4: Joint-training hyperparameters for TVG-RL and MIX.

Hyperparameter	Value
Base model	Qwen2.5-7B
Training framework	`verl`
RL algorithm	GRPO
Global steps	150
Prompts per task per step	64
Total prompts per step	320
Responses per prompt	16
Total responses per step	5,120
PPO epochs	1
Optimizer	AdamW
Learning rate	3×10^{-7}
Weight decay	0.01
Adam coefficients	$(0.9, 0.999)$
Learning-rate schedule	Constant
Task aggregation weights	0.2 for each task
TVG warm-up checkpoint	Step 50
Shared coefficient α_s	1.0
Private coefficient α_p	5.0
Complementary coefficient $\alpha_\perp$	0.05
Number of TVG-covered tensors	196
GPUs	8 × NVIDIA A800
Training random seed	One fixed training seed

Table 5: Maximum prompt and completion lengths for each task.

Task	Maximum Prompt Length	Maximum Completion Length
GSM8K	512	2,048
HH	1,024	512
KK Logic	400	2,048
Science	2,048	2,048
Instruction Following	2,048	2,048

D.4 DISTRIBUTED IMPLEMENTATION

TVG-RL uses FSDP to shard model parameters and gradients. For each TVG-constrained weight tensor, every rank locally computes the inner products between its gradient shard and the corresponding basis shards. These local inner products are then reduced across ranks to obtain the global projection coefficients. Each rank uses the resulting low-dimensional coefficients to reconstruct its local projected-gradient shard. This implementation does not require gathering the complete gradient on any single GPU and is algebraically equivalent to applying the same projection to the full unsharded tensor.

D.5 ALGORITHM

Algorithm 1 summarizes geometry estimation and joint training. Tensor indices are shown explicitly in the algorithm. The SVD step uses the equivalent Gram computation described above, and inner products use coefficient aggregation when gradients are sharded. Projector matrices are never materialized.

E EVALUATION PROTOCOL

All checkpoints are evaluated using consistent task definitions and evaluation semantics. We standardize the test data, prompt templates, chat templates, tokenization, maximum generation lengths,

Algorithm 1 Task Vector Guided Reinforcement Learning

Require: Base parameters θ_{b}; tasks $\{(\mathcal{D}_k, R_k)\}_{k=1}^{K}$; selected tensors $\mathcal{I}$.
Require: Warm-up budget T_{w}; joint-training budget T; coefficients α_{p} and $\alpha_{\perp}$.
Ensure: A single jointly trained policy π_{θ_T}.

Estimate task geometry.

for $k = 1, \ldots, K$ **do**
  Initialize at θ_{b} and run T_{w} GRPO updates on task k, obtaining $\theta_{k,T_{\mathrm{w}}}$.
  Store $\Delta_{k,\ell} \leftarrow \mathrm{vec}(\theta_{k,T_{\mathrm{w}},\ell} - \theta_{\mathrm{b},\ell})$ for every $\ell \in \mathcal{I}$.
end for
for each tensor $\ell \in \mathcal{I}$ **do**
  $\widetilde{\Delta}_{k,\ell} \leftarrow \Delta_{k,\ell}/\|\Delta_{k,\ell}\|_2$ for all k; $D_\ell \leftarrow [\widetilde{\Delta}_{1,\ell}, \ldots, \widetilde{\Delta}_{K,\ell}]$.
  Obtain the leading SVD direction from $D_\ell = U_\ell S_\ell V_\ell^{\top}$: $u_{\mathrm{s},\ell} \leftarrow (U_\ell)_{:,1}$.
  for $k = 1, \ldots, K$ **do**
    $r_{k,\ell} \leftarrow \widetilde{\Delta}_{k,\ell} - u_{\mathrm{s},\ell}(u_{\mathrm{s},\ell}^{\top}\widetilde{\Delta}_{k,\ell})$; $u_{k,\ell} \leftarrow r_{k,\ell}/\|r_{k,\ell}\|_2$.
  end for
end for

Train the shared policy.

Set $\theta_0 \leftarrow \theta_{\mathrm{b}}$ and fix all bases.
for $t = 0, \ldots, T-1$ **do**
  Initialize $\overline{g}_{t,\ell} \leftarrow 0$ for all $\ell \in \mathcal{I}$.
  for $k = 1, \ldots, K$ **do**
    Compute task k's GRPO minibatch gradients at θ_t.
    for each tensor $\ell \in \mathcal{I}$ **do**
      $c_{\mathrm{s}} \leftarrow u_{\mathrm{s},\ell}^{\top} g_{k,t,\ell}$; $c_{\mathrm{p}} \leftarrow u_{k,\ell}^{\top} g_{k,t,\ell}$.
      $g_{k,t,\ell}^{\mathrm{eff}} \leftarrow \alpha_{\perp} g_{k,t,\ell} + (1-\alpha_{\perp}) u_{\mathrm{s},\ell} c_{\mathrm{s}} + (\alpha_{\mathrm{p}} - \alpha_{\perp}) u_{k,\ell} c_{\mathrm{p}}$.
      $\overline{g}_{t,\ell} \leftarrow \overline{g}_{t,\ell} + g_{k,t,\ell}^{\mathrm{eff}}/K$.
    end for
  end for
  Apply the guided gradients through the underlying optimizer to obtain θ_{t+1}.
end for
return π_{θ_T}.

Table 6: Evaluation datasets, numbers of examples, and metrics.

Dataset	Number of Examples	Evaluation Metric
GSM8K	1,319	Answer accuracy
Full-HH-RLHF	12,451	Skywork reward
KK Logic	100	Answer accuracy
Nemotron Science	1,024	Rule-based verifier score
Instruction Following	1,024	Rule-based verifier score
GPQA-Diamond	198	Answer accuracy

decoding parameters, answer extraction rules, task verifiers, and metric computation. The model merging baselines are re-evaluated using the same pipeline rather than using numbers reported in their original papers.

For each problem, we independently sample four responses with temperature 1.0 and top-p 1.0 and report the mean score across the four samples, denoted by `avg@4`.

To improve evaluation throughput, different models and tasks use different distributed execution configurations depending on the available resources. These include question-level data sharding and different tensor-parallel and pipeline-parallel configurations. We therefore standardize the test examples, generation parameters, and scoring protocol, rather than requiring identical GPU partitioning for every evaluation run. After merging the sharded outputs, we verify the completeness and uniqueness of the question IDs and compute the final metric over the complete test set. The evaluation set sizes are listed in Table 6.

For GSM8K, KK Logic, GPQA-Diamond, MATH-500, and AMC23, a response receives a score of 1 if its extracted final answer matches the reference answer and 0 otherwise. Science and Instruction Following are scored using their corresponding rule-based verifiers. HH is scored using the Skywork reward model, and we report the unnormalized mean reward.

Because the HH reward is not on the same scale as the accuracy and rule-verifier metrics, we neither compute an unnormalized arithmetic mean over the eight metrics nor use such an average to rank the methods.

F Additional Experimental Results

This section supplements the raw-score comparisons in Sections 5.2 and 5.3 with expert-normalized results, specialist transfer matrices, and additional mathematical evaluation. Training and evaluation procedures remain in Appendices D and E.

F.1 Expert Normalization

Let $S_{b,m,j}$ denote the raw score of method m on benchmark j for backbone configuration b. For a training-domain benchmark, the reference $e_{b,j}$ is the corresponding single-task specialist's score. For GPQA-D, it is the highest score among the specialists associated with the configuration's training tasks:

$$e_{b,\mathrm{GPQA-D}} = \max_{k \in \mathcal{T}_b} S_{b,\mathrm{Expert}(k),\mathrm{GPQA-D}}, \qquad N_{b,m,j} = \frac{S_{b,m,j}}{e_{b,j}}, \tag{39}$$

where $\mathcal{T}_b$ contains five tasks for 7B and four for 1.7B. The GPQA-D references are 0.3207070707 and 0.2803030303, respectively, rounded here for display. All calculations use the unrounded raw scores.

We report the equal-weight normalized average over the benchmarks $\mathcal{B}_b$ evaluated in each configuration:

$$\mathrm{NAvg}_{b,m} = \frac{1}{|\mathcal{B}_b|} \sum_{j \in \mathcal{B}_b} N_{b,m,j}. \tag{40}$$

The two configurations contain six and five benchmarks, respectively; their averages are compared within, not across, configurations. All reference scores are positive. A normalized value above 1

Table 7: Expert-normalized 7B results. NAvg includes all six benchmarks. Bold and underlined entries denote the best and second-best generalist scores, determined before rounding.

Method	GSM8K	HH	KK	Science	IF	GPQA-D	NAvg
TVG-RL (ours)	0.965	**1.148**	**1.064**	**0.997**	0.622	**1.087**	**0.981**
mixRL	0.972	0.288	0.603	0.884	0.312	1.035	0.682
AVERAGE	0.875	-0.442	0.367	0.853	0.285	0.945	0.481
TIES	**0.994**	0.347	0.727	0.938	0.617	<u>1.071</u>	0.782
BREADCRUMBS	0.793	-0.951	0.118	0.755	0.212	0.854	0.297
ISO-C	0.981	0.310	<u>0.848</u>	<u>0.966</u>	**0.653**	1.039	<u>0.800</u>
ISO-CTS	0.979	0.194	0.808	0.963	0.580	1.031	0.759
TSV	<u>0.982</u>	<u>0.379</u>	0.684	0.946	<u>0.629</u>	1.035	0.776

Table 8: Expert-normalized 1.7B results for the four-task configuration. The GPQA-D reference is the strongest of the four associated specialists; NAvg includes all five benchmarks. Bold and underlined entries denote the best and second-best generalist scores, determined before rounding.

Method	GSM8K	HH	KK	Science	GPQA-D	NAvg
TVG-RL (ours)	**0.995**	**1.043**	**0.937**	**1.008**	**1.131**	**1.023**
AVERAGE	0.898	-5.887	0.302	0.794	0.919	-0.595
TIES	0.977	<u>-0.163</u>	0.836	0.991	<u>1.122</u>	<u>0.753</u>
BREADCRUMBS	0.835	-7.266	0.245	0.721	0.833	-0.926
ISO-C	0.976	-1.504	0.862	0.988	0.964	0.457
ISO-CTS	0.962	-2.205	0.742	0.975	1.032	0.301
TSV	<u>0.983</u>	-0.541	<u>0.874</u>	<u>1.002</u>	1.059	0.675

therefore denotes a higher numerical score than its expert reference. For HH, however, this ratio is a rescaling of an uncalibrated reward, not a multiplicative measure of response quality. Negative HH values are retained without clipping in both the tables and the radar plots.

Aggregate comparison and sensitivity to HH. TVG-RL obtains the highest NAvg in both configurations: 0.981 versus 0.800 for the strongest 7B merging baseline, ISO-C, and 1.023 versus 0.753 for the strongest 1.7B merging baseline, TIES. Because reward normalization can strongly affect a cross-task mean, we also recompute the summary after excluding HH. TVG-RL remains highest: 0.947 versus 0.898 for ISO-C and 0.761 for mixRL on 7B, and 1.018 versus 0.981 for TIES on 1.7B. The aggregate advantage in these comparisons therefore does not depend on including HH.

Visualization convention. Figure 2 displays the same three strongest merging methods by NAvg in each configuration: TIES, ISO-C, and TSV. All six merging methods remain in the result tables. To preserve negative scores without reversing polar angles, each normalized value N is drawn at nonnegative radius $r = N - r_{\min}$ with the common radial origin $r_{\min} = -1.6$. Tick labels display N, not r. The reference ring at $N = 1$ is a benchmark-wise composite reference, not a single specialist model. Neither radial polygon areas nor cross-panel area differences are used for quantitative comparisons.

F.2 SPECIALIST TRANSFER MATRICES

Tables 9 and 10 report the complete raw evaluation profiles of the specialists associated with each configuration. The diagonal entries quantify in-domain specialization, whereas the remaining entries show transfer to other benchmarks. These specialists are finite-budget reference models and are not assumed to upper-bound the performance of a jointly trained policy.

Scope of the specialist comparison. For 7B, the reported HH specialist and merging suite use the updated HH checkpoint; the evaluated TVG-RL run retains its original warm-up geometry. The

Table 9: Raw `avg@4` scores of TVG-RL and the `Qwen2.5-7B` specialists (mean ± SE). Non-HH means and SEs are multiplied by 100. Bold specialist means identify the corresponding training-domain scores, not column-wise ranking.

Method	GSM8K	HH	KK	Science	IF	GPQA-D
TVG-RL (ours)	88.82± 0.11	16.212± 0.053	79.00± 1.00	60.03± 0.46	26.12± 0.34	34.85± 0.65
Expert (GSM8K)	**92.00**± 0.20	-6.775± 0.362	5.75± 1.44	49.17± 0.57	10.72± 0.31	31.06± 0.98
Expert (HH)	87.32± 0.30	**14.124**± 0.026	3.25± 1.11	49.80± 0.50	10.42± 0.17	32.07± 2.19
Expert (KK)	50.42± 0.40	-16.038± 1.552	**74.25**± 1.44	44.73± 0.97	7.18± 0.19	24.49± 1.47
Expert (Science)	76.63± 0.43	-8.551± 0.343	7.00± 1.29	**60.21**± 0.47	9.74± 0.13	30.68± 1.93
Expert (IF)	20.17± 0.30	-10.285± 0.156	7.75± 1.89	49.44± 0.87	**41.99**± 0.71	32.07± 0.76

Table 10: Raw `avg@4` scores of TVG-RL and the `Qwen3-1.7B` specialists (mean ± SE). Non-HH means and SEs are multiplied by 100. Bold specialist means identify the corresponding training-domain scores, not column-wise ranking.

Method	GSM8K	HH	KK	Science	GPQA-D
TVG-RL (ours)	84.78± 0.10	3.321± 0.034	37.25± 0.25	55.76± 0.31	31.69± 0.98
Expert (GSM8K)	**85.24**± 0.10	-17.977± 0.607	2.75± 1.11	34.16± 1.09	25.88± 2.21
Expert (HH)	74.85± 0.79	**3.185**± 0.019	3.25± 1.49	38.65± 0.97	28.03± 1.18
Expert (KK)	27.27± 8.41	-27.913± 0.855	**39.75**± 0.48	32.91± 1.41	14.90± 2.36
Expert (Science)	61.66± 3.43	-23.849± 0.860	7.50± 0.87	**55.30**± 0.13	26.77± 1.90
Base	28.45± 7.46	-27.522± 0.835	4.75± 1.31	26.76± 1.70	16.79± 2.24

comparisons therefore concern the evaluated training and integration pipelines, rather than an experiment using identical HH checkpoints for every role. Equal numbers of joint and specialist steps also do not establish equal total training data, optimizer updates across all models, or compute. Consequently, scores above a specialist reference are evidence that the evaluated generalist can exceed that reference, but do not by themselves isolate cross-task transfer from all budget and optimization effects.

F.3 Online Cost

We measure the online cost of TVG-RL on the five-task 7B configuration, separately from task-specific warm-ups and basis construction. TVG-RL and mixRL are timed on eight A800 GPUs under matched hardware, software, and input prompts, using 64 prompts per task and 16 responses per prompt. At each evaluated checkpoint, the learning rate is set to zero and checkpoint saving is disabled; we discard one timing warm-up step and average the next four.

Mean end-to-end step times are 585.8–597.9 seconds for TVG-RL and 568.4–577.4 seconds for mixRL (Figure 3), corresponding to an online overhead of 3.1%–3.8%. The five task-gradient projections across 196 target tensors take 15.7–17.0 seconds per step in total (mean: 16.1 seconds), or 2.7%–2.9% of TVG-RL step time. At the step-150 checkpoint, actor-update times excluding projection are nearly identical: 333.40 seconds for TVG-RL and 333.65 seconds for mixRL. TVG-RL therefore adds a small measured overhead to the joint-training loop.

Extrapolating the mean projection time over 150 joint steps gives approximately 0.67 wall-clock hours, or 5.37 GPU-hours on eight GPUs. These projection-only estimates and the online overhead exclude task-specific warm-up training and basis construction.

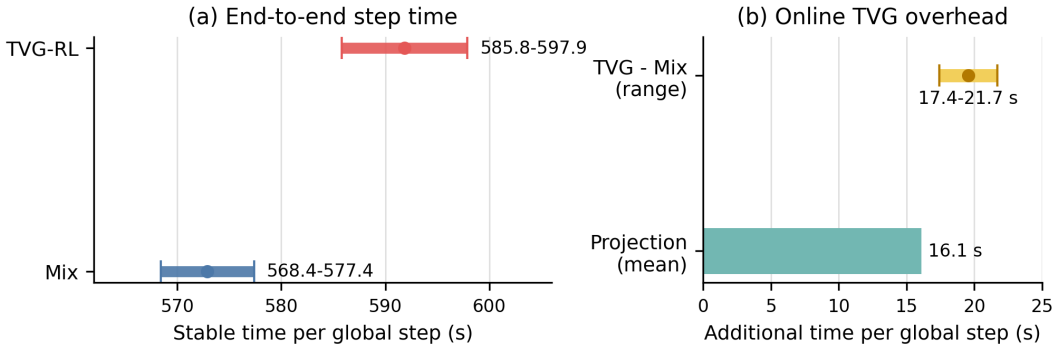


Figure 3: Online cost in the matched five-task 7B timing benchmark. Left: ranges of mean end-to-end step times across checkpoints. Right: online overhead and time spent on the five task-gradient projections. Each mean uses four steps after one timing warm-up. Task-specific warm-up training and basis construction are excluded.