CARE-RL: Capability-Aware Reinforcement Learning for Mitigating Cross-Domain Conflicts

Anonymous ACL submission

Abstract

Reinforcement learning (RL) with verifiable rewards has achieved strong progress in reasoning-oriented LLMs, but extending it to multi-domain RL remains challenging due to reward unreliability in non-verifiable tasks and capability interference across domains. We propose CARE-RL to combine protocol-aware reward generation with capability-aware optimization for mitigating cross-domain conflicts. For non-verifiable tasks, the Protocol-Aware Generative Reward Model (PA-GRM) constructs prompt-level evaluation protocols and schemas before producing trace-conditioned rewards, enabling task-adaptive yet comparable evaluation of open-ended responses. For multi-domain optimization, Direction-Aware Capability Subspace Projection (DACSP) extracts historical capability directions from previous RL stages and modulates later updates by amplifying aligned components, suppressing conflicting components, and preserving orthogonal updates. Experiments across math, chat, and instruction-following benchmarks show that CARE-RL consistently outperforms standard multi-domain RL baselines, achieving Total Avg scores of 47.9 and 50.7 on Qwen2.5-7B and Qwen3-4B, respectively.

1 Introduction

Reinforcement learning (RL) has become a central paradigm for post-training large language models (LLMs), especially in large reasoning models (DeepSeek-AI, 2025; Kimi Team, 2025). A key driver of this progress is Reinforcement Learning with Verifiable Rewards (RLVR), where rewards are derived directly from task outcomes rather than from reward models (Wen et al., 2025). For example, mathematical responses can be checked against reference answers, and generated code can be evaluated by unit tests. Because such rewards are grounded in task correctness, they are less exposed to subjective preference noise and reward-model bias. This has made RLVR effective in verifier-rich domains such as mathematics and coding (Shao et al., 2024), while recent studies have begun to push RLVR beyond narrowly verifiable tasks toward broader, multi-domain settings with free-form or less structured answers (Su et al., 2025; Yu et al., 2025; Gunjal et al., 2025).

However, extending RLVR to multi-domain learning remains fundamentally challenging, because many real-world tasks are open-ended then non-verifiable. To provide rewards for such tasks, recent work has explored generative reward models (Zhang et al., 2025) and rubric-based evaluation protocols (Shao et al., 2025; Gunjal et al., 2025), replacing direct scalar scoring with fine-grained evaluation. However, these methods do not fully solve the reward construction problem as even non-verifiable tasks may require different evaluation protocols. For example, a story-writing prompt is often better judged holistically by its coherence, style and overall engagement, while a travel-planning prompt requires checking whether the response satisfies concrete constraints such as budget, schedule and user preferences. Therefore, applying a single fixed judging protocol to heterogeneous open-ended prompts can still produce unstable or misaligned rewards, even when the judge model itself is strong. Beyond reward construction, multi-domain RL also requires a single model to learn across diverse capabilities and data distributions. Existing multi-domain RL methods typically adopt either joint or sequential training, but joint training may suffer from conflicts among multiple objectives, while sequential training may cause later updates to degrade capabilities acquired in earlier stages (Cheng et al., 2025; Yang et al., 2026a; Zheng et al., 2025). Therefore, multi-domain RL requires not only a more flexible reward construction mechanism for non-verifiable tasks, but also an optimization mechanism that preserves accumulated capabilities while adapting to new domains.

To address these challenges, we propose **CARE-RL**, a reward-adaptive and capability-aware cascade learning framework for mitigating conflicts in cross-domain RL. On the reward side, we introduce the Protocol-Aware Generative Reward Model (**PA-GRM**) for non-verifiable tasks. Instead of assigning rewards under a fixed evaluation scheme, PA-GRM first constructs a prompt-level evaluation protocol and scoring schema, and then generates response-level scoring traces under the protocol. By routing prompts between holistic and rubric-based judging, PA-GRM produces rewards that better match the structure of heterogeneous open-ended tasks. On the optimization side, we introduce Direction-Aware Capability Subspace Projection (**DACSP**). The parameter change of each completed RL stage encodes the capability acquired in that stage, and its leading singular directions approximate capability-increasing directions. DACSP uses them as a low-rank reference for later updates: amplifying aligned components, suppressing opposing ones and leaving orthogonal ones unchanged. In this way, CARE-RL integrates prompt-adaptive reward construction with update modulation across cascade stages, aiming to provide more suitable supervision for non-verifiable tasks while reducing update components that conflict with previously learned domains.

Our contributions are summarized as follows:

- We propose **CARE-RL**, a unified framework that simultaneously addresses two key challenges in multi-domain RL: reward unreliability on non-verifiable tasks and cross-domain capability degradation.
- PA-GRM adaptively constructs per-prompt evaluation protocols, alleviating the gap of fixed evaluation schemes in heterogeneous tasks and providing more reliable rewards; DACSP extracts capability subspaces from prior RL stages and applies direction-aware modulation to subsequent updates, preventing the erosion of acquired capabilities.
- CARE-RL achieves the highest average scores on `Qwen2.5-7B`/`Qwen3-4B`, outperforming the strongest baseline by +1.0/+0.9 points.

2 Related Works

Reinforcement Learning towards General Domains. Recent LLM post-training research has extended RL from verifier-rich reasoning tasks to broader general-domain settings. In verifiable domains, pure RL and RLVR-style training have shown strong effectiveness for reasoning-oriented models (DeepSeek-AI, 2025; Shao et al., 2024), while scalable RL algorithms such as DAPO and VAPO further improve advanced reasoning training (Yu et al., 2026; Yue et al., 2025). Beyond narrow verifier-rich domains, recent systems broaden RL through cross-domain data construction and verifiability-aware data mixing (Ma et al., 2025; Akter et al., 2025). Other studies explore multi-domain RL settings that combine verifiable and non-verifiable training signals (Zeng et al., 2025; Bhaskar et al., 2025). As the training scope expands, mixed-domain and sequential-domain optimization have become increasingly important, motivating methods based on gradient surgery, task balancing, and cascaded RL (Cai et al., 2026; Ramesh et al., 2026; Wang et al., 2025a; Yang et al., 2026b). This reflects the shift from verifier-rich RL in single domain toward multi-domain RL over heterogeneous capabilities and reward sources.

Judge for Open-ended Problems. Open-ended reward modeling has also moved from direct scalar judging toward more structured evaluation. Reasoning-based reward models formulate judgment as a reasoning process, and branch-and-rethink mechanisms further improve difficult evaluations (Chen et al., 2025; Jiao et al., 2025). Rubric-based reward methods provide another structured approach by decomposing open-ended evaluation into explicit criteria; recent work shows that rubric feedback can support RL beyond verifiable domains and can be refined or evolved for long-horizon open-ended training (Gunjal et al., 2025; Shao et al., 2025; Shen et al., 2026). Overall, these works indicate a trend from black-box scalar reward prediction toward more interpretable and structured evaluation processes.

3 Methodology

As shown in Figure 1, CARE-RL organizes multi-domain RL as a cascade over multiple training domains. For verifiable domains, rewards are obtained from task-specific verifiers, while CARE-RL invokes PA-GRM to construct rewards for non-verifiable domains. After each RL stage, CARE-RL records the parameter change induced by that stage and uses DACSP to extract historical capability directions from the update. In later stages, before applying each optimizer step, DACSP decomposes the proposed update with respect to these historical

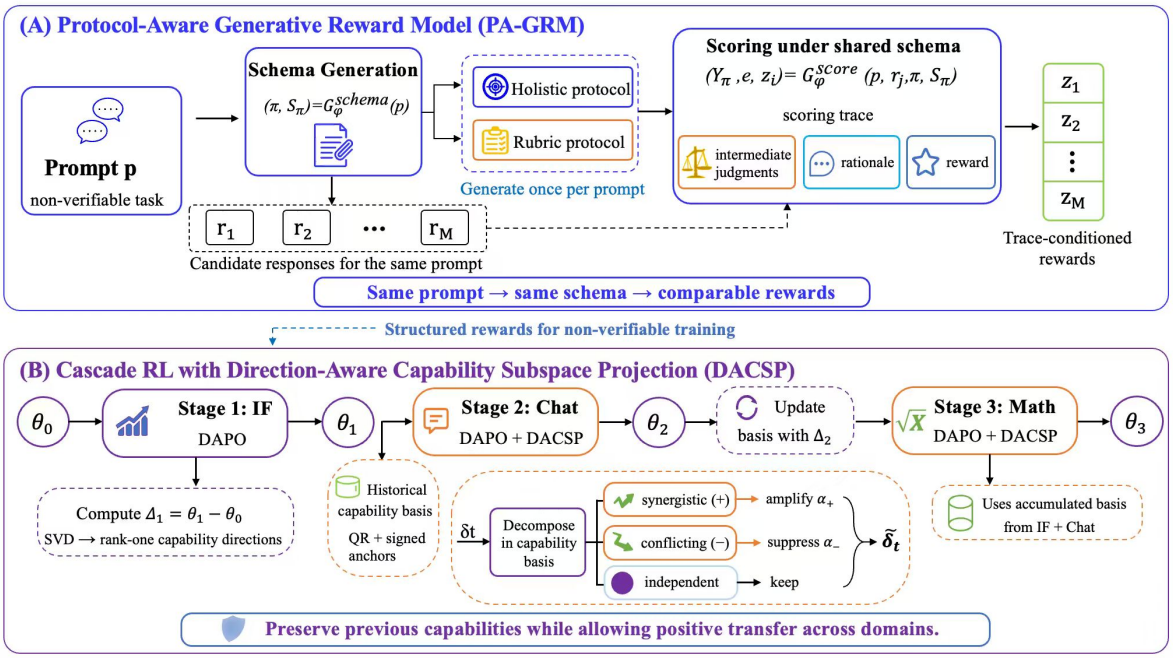

Figure 1: Overview of CARE-RL. (A) PA-GRM builds a prompt-level protocol and schema, then scores all candidates under the shared schema for trace-conditioned rewards. (B) Cascade RL over IF, Chat, and Math with DAPO; DACSP modulates later updates along historical capability directions, amplifying synergistic, suppressing conflicting, and preserving independent components.

directions and modulates its components according to their alignment. Therefore, PA-GRM determines how rewards are produced for open-ended tasks, while DACSP determines how updates are applied across stages. The remainder of this section first describes PA-GRM, then presents DACSP.

3.1 Protocol-Aware Generative Reward Model

In order to route heterogeneous non-verifiable prompts to evaluation protocols that match their task structure, we introduce the PA-GRM, denoted as G_ψ. PA-GRM decomposes reward generation into prompt-level protocol construction and response-level scoring. Given a prompt, it first selects an evaluation protocol and constructs the corresponding scoring schema; given a candidate response, it then generates a scoring trace under this prompt-specific schema, from which the final scalar reward is extracted. Thus, the evaluation standard is adapted to each prompt, while all candidate responses to the same prompt are judged under a shared schema. The complete prompt templates are provided in Appendix B.

We consider two evaluation protocols:

$$\pi \in \{\text{HOLISTIC}, \text{RUBRIC}\}.$$

The protocol specifies how quality factors are operationalized during scoring. The HOLISTIC protocol is used for open-ended prompts that are difficult to decompose into independently checkable criteria, such as creative writing, where quality is better judged holistically. The RUBRIC protocol is used for prompts with explicit structural or content requirements, where each requirement can be checked more independently.

Given a prompt p, PA-GRM first determines the protocol and schema:

$$(\pi, \mathcal{S}_\pi) = G_\psi^{\text{schema}}(p),$$

where $\mathcal{S}_\pi$ denotes the evaluation schema under protocol π. Conditioned on this fixed prompt-level standard, PA-GRM scores a candidate response r:

$$(\mathcal{Y}_\pi, e, z) = G_\psi^{\text{score}}(p, r, \pi, \mathcal{S}_\pi),$$

where $\mathcal{Y}_\pi$ denotes protocol-specific intermediate judgments, e denotes the natural-language reasoning produced during scoring, and $z \in [0, 1]$ is the final scalar reward. For a group of candidate responses $\{r_j\}_{j=1}^M$ to the same prompt, all responses share the same $(\pi, \mathcal{S}_\pi)$ to ensure comparability.

3.1.1 Holistic Protocol

When a prompt cannot be reliably decomposed into independent checklist items, PA-GRM uses

the HOLISTIC protocol for structured holistic evaluation. Under this protocol, the schema consists of a set of scoring considerations:

$$\mathcal{S}_{\text{HOLISTIC}} = \{(a_i, d_i)\}_{i=1}^{m},$$

where a_i denotes a scoring aspect and d_i describes how this aspect should inform the judgment. These considerations make the basis of evaluation explicit, but are not treated as independent reward items.

Given a response r, PA-GRM generates a scoring trace in a single pass: it first emits a local judgment $\{y_i\}_{i=1}^{m}$ for each aspect, and then emits the final scalar score z conditioned on the full set of local judgments.

3.1.2 Rubric Protocol

When a prompt contains explicit structural or content requirements, PA-GRM uses the RUBRIC protocol to instantiate the evaluation standard as relatively independent criteria:

$$\mathcal{S}_{\text{RUBRIC}} = \mathcal{R}(p) = \{(\rho_i, \omega_i, \tau_i)\}_{i=1}^{K},$$

where ρ_i is the i-th rubric criterion, ω_i is its importance weight, and τ_i denotes its criterion type. Each ρ_i is prompt-specific, sufficiently informative for judgment, and relatively independently checkable against a candidate response.

Rubric items are not generic dimensions such as completeness or coherence; instead, they instantiate prompt-specific requirements, such as key points to cover or reasoning relations to satisfy. Given the rubric schema, PA-GRM generates item-level judgments $\{y_i\}_{i=1}^{K}$, which may include satisfaction degrees, violation reasons, local evidence, and local scores, and then emits the final scalar score z conditioned on all item-level judgments.

3.1.3 Training and Inference of PA-GRM

To enable both protocol selection and response scoring, PA-GRM is trained with complete scoring traces. We represent the target output as

$$t = (\pi, \mathcal{S}_\pi, \mathcal{Y}_\pi, e, z),$$

where $(\pi, \mathcal{S}_\pi)$ defines the prompt-level evaluation standard and $(\mathcal{Y}_\pi, e, z)$ defines the response-level scoring result. G_ψ is trained to model the decomposed generation process:

$$P_\psi(t \mid p, r) = P_\psi(\pi, S_\pi \mid p)\, P_\psi(Y_\pi, e, z \mid p, r, \pi, S_\pi),$$

rather than a direct mapping $(p, r) \mapsto z$.

The trace-level supervision is collected by distilling from DeepSeek-V4-Pro on 10k prompt-response pairs sampled from `UltraFeedback` (Cui et al., 2023) and `HelpSteer3` (Wang et al., 2025b). For each pair, the teacher produces a complete scoring trace. These traces are used as supervised targets:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(p,r,t)}\Big[\log P_\psi(\pi, \mathcal{S}_\pi \mid p) + \log P_\psi(\mathcal{Y}_\pi, e, z \mid p, r, \pi, \mathcal{S}_\pi)\Big].$$

After training, PA-GRM serves as the reward model for non-verifiable tasks during RL. At inference time, it first generates and caches $(\pi, \mathcal{S}_\pi)$ for a prompt p, and then scores each candidate response r_j under this shared schema. The resulting scalar $z^{(j)} \in [0, 1]$ is used as the non-verifiable reward:

$$R_{\text{NV}}(p, r_j) = z^{(j)}.$$

3.2 Direction-Aware Capability Subspace Projection

In order to make cascade RL accumulate capabilities across domains, we introduce DACSP. In cascade RL, domains are optimized sequentially, with each stage using its own reward signal and training configuration. DACSP controls how the optimizer updates in later stages are applied with respect to capabilities acquired earlier. After each completed stage, DACSP extracts layer-wise capability directions from the induced parameter change. During subsequent stages, it projects each optimizer-proposed update onto the historical directions and modulates the aligned, opposing, and orthogonal components differently. In this way, later stages can acquire new capabilities while reducing destructive updates against previously learned ones.

3.2.1 Stage-wise Capability Directions

After completing the RL stage for domain d, we have obtained the parameter change induced by this domain. To use this change as a reference for later stages, DACSP first converts it into layer-wise capability directions. Let θ_{d-1} and θ_d denote the model parameters before and after the RL stage for domain d. We define the stage-wise task vector as: $\Delta_d = \theta_d - \theta_{d-1}$. Since different layers may encode different types of capabilities, we extract capability directions independently for each layer. For layer l, we reshape $\Delta_d^{(l)}$ into a matrix and compute its thin SVD:

$$\Delta_d^{(l)} = U_d^{(l)} \Sigma_d^{(l)} (V_d^{(l)})^\top.$$

Let $u_{d,i}^{(l)}$, $v_{d,i}^{(l)}$, and $\sigma_{d,i}^{(l)}$ denote the i-th left singular vector, right singular vector, and singular value, respectively. We retain the smallest number $k_d^{(l)}$ of leading singular components such that $\sum_{i=1}^{k_d^{(l)}}(\sigma_{d,i}^{(l)})^2 / \sum_i (\sigma_{d,i}^{(l)})^2 \geq \tau$. For each retained component, we define a rank-one capability direction in the original layer-parameter space:

$$B_{d,i}^{(l)} = u_{d,i}^{(l)} (v_{d,i}^{(l)})^\top, \quad i = 1, \ldots, k_d^{(l)}.$$

These directions $\{B_{d,i}^{(l)}\}_{i=1}^{k_d^{(l)}}$ summarize the dominant parameter changes induced by domain d at layer l. We validate this capability-direction interpretation via checkpoint surgery in Appendix C. Since each direction has the same shape as the layer parameters, later updates can be compared with them using the Frobenius inner product.

3.2.2 Historical Direction Basis

After several stages, we have collected capability directions from multiple previous domains. To use them jointly in later optimization, DACSP aggregates these directions into a compact and orthonormal historical basis. Before training stage d, for layer l, let:

$$\widetilde{\mathcal{B}}_{<d}^{(l)} = \{B_{k,i}^{(l)} : k < d,\ i = 1, \ldots, k_k^{(l)}\}$$

be the set of historical capability directions. We vectorize and concatenate them:

$$Z_{<d}^{(l)} = [\mathrm{vec}(B_{1,1}^{(l)}), \ldots, \mathrm{vec}(B_{d-1,k_{d-1}^{(l)}}^{(l)})].$$

We then perform rank-revealing QR decomposition: $Z_{<d}^{(l)} \Pi^{(l)} = \widehat{Q}^{(l)} R^{(l)}$, and retain the columns whose corresponding diagonal values in $R^{(l)}$ exceed a small tolerance. The retained columns are reshaped back into matrices with the same shape as the layer parameters, yielding an orthonormal historical direction basis: $\mathcal{B}_{<d}^{(l)} = \{Q_i^{(l)}\}_{i=1}^{K_{<d}^{(l)}}$.

Because QR decomposition may flip basis signs, we orient each basis vector using its pivoted source direction as a signed anchor. Let $z_{\pi(i)}^{(l)}$ denote the original historical direction selected as the pivot for the i-th retained basis vector. We normalize the sign by

$$Q_i^{(l)} \leftarrow \mathrm{sign}\left(\left\langle \mathrm{vec}(Q_i^{(l)}), z_{\pi(i)}^{(l)} \right\rangle\right) Q_i^{(l)}.$$

After this step, the positive direction of each basis vector is aligned with a historical capability-increasing direction, making alignment and opposition well-defined for later update modulation.

3.2.3 Direction-Aware Projection

With the historical direction basis constructed, DACSP can regulate the optimizer update in the current stage. During stage d, the base optimizer first computes a parameter update for each layer. Let $\delta_t^{(l)}$ denote the update that would be added to the parameters of layer l at step t before applying DACSP. This is the optimizer-processed update, such as an Adam-preconditioned step, rather than the raw gradient.

For each historical basis vector $Q_i^{(l)}$, we compute the projection coefficient $s_i = \left\langle \delta_t^{(l)}, Q_i^{(l)} \right\rangle_F$. A positive s_i means that the proposed update moves along a historical capability direction, while a negative s_i means that it moves against that direction. Since $\{Q_i^{(l)}\}$ forms an orthonormal basis of the historical capability subspace, we decompose the update into its historical-subspace component and orthogonal residual:

$$\delta_t^{(l)} = \sum_i s_i Q_i^{(l)} + \delta_\perp^{(l)}, \quad \delta_\perp^{(l)} = \delta_t^{(l)} - \sum_i s_i Q_i^{(l)}.$$

Then applies direction-aware modulation:

$$\widetilde{\delta}_t^{(l)} = \delta_\perp^{(l)} + \alpha_+ \sum_{i:s_i>0} s_i Q_i^{(l)} + \alpha_- \sum_{i:s_i<0} s_i Q_i^{(l)}.$$

Here $\alpha_+ \geq 1$ preserves or amplifies updates aligned with historical capability directions, while $\alpha_- \in [0, 1)$ suppresses updates moving against them. The orthogonal component is left unchanged, allowing the current stage to acquire capabilities outside the historical subspace. The layer parameters are updated by $\theta_{t+1}^{(l)} = \theta_t^{(l)} + \widetilde{\delta}_t^{(l)}$.

Operationally, the first RL stage is trained with the base optimizer. After each completed stage, DACSP computes the stage-wise parameter change, extracts layer-wise capability directions by SVD and updates the historical direction basis through QR orthogonalization and signed-anchor normalization. In later stages, the base optimizer first proposes $\delta_t^{(l)}$ and DACSP modulates it before applying the final update $\widetilde{\delta}_t^{(l)}$. SVD and QR are performed only at stage transitions, while the per-step overhead is limited to projections onto the retained historical basis.

4 Experiments

4.1 Experimental Setup

Training Datasets and Models. We use three representative domains for RL training: math reasoning, general chat, and instruction following (IF).

To prevent the training signal from being dominated by any single domain due to data scale, we use 8,000 prompts for each domain. For math, we retain the full `AIME83-24` as hard mathematical data and randomly sample the remaining examples from `OpenR1-Math-220k`. For general chat and instruction following, we randomly sample 8,000 prompts from `No-Robots` and `RLVR-IFeval` (Lambert et al., 2024; Zhou et al., 2023), respectively. Table 1 summarizes the training datasets used in our experiments. For math reasoning, instruction-following, and general chat, we compute rewards using binary answer matching, the rule-based verifier from `RLVR-IFeval`, and PA-GRM, respectively. We study two backbone models, `Qwen2.5-7B` and `Qwen3-4B` (Yang et al., 2025).

Table 1: Overview of Training Datasets and Domains

Dataset	Size	Domain
`OpenR1-Math-220k`	7,067	*Math*
`AIME83-24`	933	*Math*
`No-Robots`	8,000	*General Chat*
`RLVR-IFeval`	8,000	*IF*

Implementation Details. CARE-RL uses the same IF→Chat→Math (Yang et al., 2026b) DAPO configuration with 4-bit LoRA and bf16, trained on 2×H200 via VeRL with seed 42. LoRA uses rank 16, alpha 32, dropout 0.05, per-device batch size 1, gradient accumulation 8, and one epoch per stage. We use asymmetric clipping with $\epsilon_{\text{low}} = 0.2$ and different ϵ_{high}. The stage settings $(\eta, G, L, T, \epsilon_{\text{high}})$ with G = generations and L = max_tokens are IF $(2 \times 10^{-6}, 8, 1024, 1.0, 0.24)$, Chat $(1 \times 10^{-6}, 4, 1024, 0.9, 0.26)$, and Math $(5 \times 10^{-6}, 8, 2048, 1.0, 0.28)$. PA-GRM based on `Qwen3-8B` is LoRA-SFT trained on `DeepSeek-V4-Pro` trajectories with 4-bit bf16 loading, rank 32, alpha 64, dropout 0.05, learning rate 5×10^{-6}, sequence length 4096, batch size 1, gradient accumulation 16, and one epoch. DACSP performs layer-wise SVD on each stage's LoRA update with energy threshold τ=0.95 for retaining leading singular components, and in later stages scales aligned/opposing projections by α_+=1.2 and α_-=0.2, with <1% overhead. Evaluation samples at temperature 1.0, top-p 0.95, with avg@4. The code and configurations are available at https://anonymous.4open.science/status/Anonymous_code_may-E003.

Baselines. We compare five popular training configurations in multi-task RL. **V→NV** and **NV→V** are two cascade RL baselines without cross-domain protection. In our experiments, V→NV follows the order IF → Math → Chat, while NV→V follows IF → Chat → Math. **Naive Mixing** places prompts from all domains into a single training batch with per-domain advantage normalization. **MGS** (Cai et al., 2026) mitigates cross-domain interference in mixed-domain RL by applying gradient surgery locally at the Transformer-module level rather than globally over the whole model. **MOPD** (Yang et al., 2026b) uses the best intermediate checkpoint per domain after each cascade stage as a teacher, applying on-policy distillation to recover benchmark regressions before proceeding to the next stage.

Evaluation Benchmarks and Metrics. We construct a comprehensive evaluation suite covering the three training domains for testing CARE-RL. Specifically, we use:

- **Math:** MATH500 (Lightman et al., 2023), AIME25, and GSM8K (Cobbe et al., 2021).
- **Chat:** WildBench (Lin et al., 2024), Creative Writing v3 and ResearchQA (Yifei et al., 2025).
- **Instruction Following:** IFBench (Pyatkin et al., 2025) and IFEval (Zhou et al., 2023).

We cap each evaluation benchmark at 500 examples and perform train–test decontamination against all RL and PA-GRM training data using normalized exact matching and token-level 5-gram Jaccard filtering with a threshold of 0.8. To reduce variance from single-sample evaluation, we adopt an **avg@4** protocol. For each prompt, we sample four responses and report the average sample-level score. To get more reliable evaluation and reduce bias, all chat benchmarks are scored by average of `Gemini-2.5-Pro`, `Kimi-K2.5` and `Qwen3-Max`. Math benchmarks are graded by answer matching and IF benchmarks use the official verifiers.

4.2 Main Results

Table 2 presents the main results across our method and baselines. We report domain-group averages (M. Avg for math, C. Avg for chat) and the overall Total Avg across all benchmarks.

CARE-RL achieves the best Total Avg on both backbones, improving over the strongest baseline MOPD by +1.0 on `Qwen2.5-7B` and +0.9 on `Qwen3-4B`. We also observe that unprotected cascade baselines are sensitive to the placement of

Table 2: Evaluation Results. Bold indicates the best result and underline indicates the second-best result.

		Math				Chat				IF		
Backbone	Method	MATH	GSM	AIME	M. Avg.	WB	CW3	RQA	C. Avg.	IFB	IFE	Total Avg.
Qwen2.5-7B	Base	60.9	84.7	4.2	49.9	22.1	43.5	41.1	35.6	15.0	31.2	37.8
	V→NV	75.9	90.8	5.8	57.5	36.5	46.8	40.1	41.1	24.8	38.2	44.9
	NV→V	78.2	92.2	8.3	59.6	35.7	35.4	43.9	38.3	24.4	39.2	44.7
	Naive Mixing	76.2	91.4	5.8	57.8	35.9	39.2	42.3	39.1	24.2	38.9	44.2
	MGS	77.4	92.0	7.5	59.0	37.4	44.5	43.0	41.6	25.3	40.4	45.9
	MOPD	78.6	92.7	7.5	59.6	38.5	47.1	43.8	43.1	26.1	41.2	46.9
	CARE-RL	**79.7**	**93.2**	**9.2**	**60.7**	**39.2**	**48.2**	**44.7**	**44.0**	**27.2**	**41.8**	**47.9**
Qwen3-4B	Base	68.4	87.8	4.2	53.5	25.3	45.0	43.4	37.9	18.0	39.4	41.4
	V→NV	77.6	92.6	10.8	60.3	42.0	48.1	44.5	44.9	25.6	45.0	48.3
	NV→V	79.0	93.6	**14.2**	62.3	40.5	39.8	46.0	42.1	26.8	46.6	48.3
	Naive Mixing	78.1	93.0	9.2	60.1	41.1	43.6	45.3	43.3	26.0	45.7	47.8
	MGS	78.8	93.7	13.3	61.9	42.8	46.5	45.8	45.0	26.3	46.9	49.3
	MOPD	79.4	93.9	11.7	61.7	43.6	48.3	46.7	46.2	26.9	47.6	49.8
	CARE-RL	**80.1**	**94.3**	**14.2**	**62.9**	**44.1**	**49.2**	**47.5**	**46.9**	**27.3**	**48.9**	**50.7**

verifiable and non-verifiable stages: V→NV favors Chat performance while NV→V favors Math performance. Compared with the mixed-domain training of MGS, the cascade structure of CARE-RL allows per-domain hyperparameter tuning, and PA-GRM provides more reliable rewards. Compared with the post-hoc distillation of MOPD, DACSP modulates update directions during optimization, preventing regression at its source while avoiding extra distillation overhead and its potential interference with other domains. Appendix A further reports the expanded benchmark-level results with standard deviations. The gains are consistent across both backbones and domains, indicating that the improvements come from the framework.

4.3 Cross-Domain Interference in RL

To understand the gains of CARE-RL, we analyze the cross-domain interference in standard RL.

Single-domain RL. We train `Qwen3-4B` on each domain independently and report the average change on the two non-target domains relative to the base model as OOD Change.

Table 3: Single-domain RL on `Qwen3-4B`.

Training	M. Avg	C. Avg	IF Avg.	OOD Change
Base	53.5	37.9	28.7	–
Math-only	61.7	37.0	27.6	-1.0
Chat-only	51.6	45.1	30.7	+0.1
IF-only	52.7	36.8	38.6	-1.0
CARE-RL	62.9	46.9	38.1	–

Table 3 shows that interference arises even without multi-stage interaction: Math-only and IF-only yield negative OOD Change, while Chat-only still regresses on Math; CARE-RL achieves stronger Math and Chat averages than the corresponding single-domain runs and remains competitive on IF while covering all three domains.

Stage-wise cascade RL. We evaluate the checkpoint after each stage of the cascade IF → Chat → Math, and report Reg. as the average drop on previously trained domains.

Table 4: Stage-wise cascade RL on `Qwen3-4B`.

Checkpoint	M. Avg	C. Avg	IF Avg.	Total	Reg.↓
Base	53.5	37.9	28.7	41.4	0.0
After IF	52.7	36.8	38.6	43.2	0.0
After Chat	48.1	43.4	36.9	43.5	1.6
After Math	62.3	42.1	36.7	48.3	1.5
CARE-RL	62.9	46.9	38.1	50.7	–

Table 4 shows that interference also accumulates across cascade stages: tracking IF, its 38.6 after stage one drops to 36.9 after Chat and 36.7 after Math, eroding about one-fifth of the IF-stage gain (1.9 out of 9.9 points). CARE-RL preserves the IF gain while still improving Chat and Math, yielding a 2.4-point Total improvement over standard cascade (50.7 vs. 48.3) and showing that capability-aware modulation, not ordering alone, is what makes cumulative gains possible.

4.4 Ablation Study

We conduct ablation studies on `Qwen3-4B` to isolate the contribution of each component in CARE-RL.

Component ablation of CARE-RL. We next evaluate how PA-GRM and DACSP contribute to CARE-RL. Standard Cascade removes both PA-GRM and DACSP. CARE-RL w/o PA-GRM keeps DACSP but replaces PA-GRM with direct judge using `Qwen3-8B` for non-verifiable rewards. CARE-RL w/o DACSP keeps PA-GRM but removes capability-aware update modulation.

Table 5: Component ablation on `Qwen3-4B`.

Variant	M. Avg	C. Avg	IF Avg.	Total
Standard Cascade	62.3	42.1	36.7	48.3
CARE-RL w/o PA-GRM	63.5	43.5	37.5	49.5
CARE-RL w/o DACSP	62.0	45.0	37.0	49.4
CARE-RL	62.9	46.9	38.1	50.7

Table 5 shows that PA-GRM and DACSP individually bring +1.1 and +1.2 over Standard Cascade, while combining them yields +2.4, slightly exceeding the sum of single contributions. The components are therefore roughly additive with a small positive interaction: PA-GRM produces cleaner reward signals on non-verifiable tasks, which in turn lets DACSP more accurately isolate stable capability directions across stages.

Effectiveness of PA-GRM. PA-GRM improves reward generation for non-verifiable tasks by constructing prompt-level evaluation protocols before scoring candidate responses. We compare it with direct judging, fixed-protocol variants, PA-GRM and stronger PA-GRM variants using `DeepSeek-V4-Pro` as upper bound. We randomly sample 500 items from three common judging benchmarks and report *judging accuracy*, defined as the fraction of items for which the judge recovers the benchmark gold judgment label.

Table 6: PA-GRM judging accuracy. `DS` denotes replacing the trained PA-GRM with `DeepSeek-V4-Pro` as the judge backbone under the same protocol routing.

Variant	RewardBench2	JudgeBench	RM-Bench
Direct	64.7	58.4	64.2
Holistic	69.8	62.7	69.5
Rubric	72.8	67.9	74.1
PA-GRM	83.2	85.1	81.8
PA-GRM-DS	87.0	86.7	84.2

Table 6 shows that adaptive protocol construction substantially improves judging accuracy: compared with the fixed Rubric protocol, PA-GRM improves by +7.7 to +17.2 points in the three benchmarks. Replacing the trained PA-GRM model with the stronger `DeepSeek-V4-Pro` in same method adds +1.6 to +3.8 points. These results suggest that prompt-level protocol adaptation is an important source of reward reliability, which shows that different problems require different evaluation methods.

Analysis of DACSP. We treat the modulation coefficients (α_+, α_-) as the central design knob and place four diagnostic points around the default $(1.2, 0.2)$: Standard Cascade $(1.0, 1.0)$ disables modulation, Hard Projection $(0, 0)$ is a degenerate orthogonal-only endpoint, while No Amplification (α_+=1) and No Suppression (α_-=1) isolate one side.

Table 7: DACSP ablation on `Qwen3-4B`.

Variant	M. Avg	C. Avg	IF Avg.	Total	Reg.↓
Standard Cascade	62.3	42.1	36.7	48.3	1.5
Hard Projection	62.7	42.6	36.9	48.7	0.4
No Amplification	63.2	44.0	37.4	49.5	0.9
No Suppression	63.0	44.7	37.8	49.8	2.1
DACSP	62.9	46.9	38.1	50.7	0.7

Table 7 shows that $(0, 0)$ over-protects history, achieving the lowest Reg. (0.4) but only 48.7 Total; the diagnostic α_-=1 setting raises Reg. to 2.1, worse than no modulation, confirming that suppression is what protects earlier capabilities; setting α_+=1 drops Total by 1.2, confirming that amplification yields a real positive-transfer gain. The default active setting $(1.2, 0.2)$ is the only point with both high Total and low Reg. We set τ=0.95 following standard SVD truncation; smaller/larger τ would degenerate toward the Standard Cascade / Hard Projection endpoints already covered, and we keep (α_+, α_-) stage-invariant to avoid grid explosion and overfitting to domain order.

5 Conclusion

In this paper, we proposed CARE-RL, a RL framework for multi-domain RL. CARE-RL combines PA-GRM, which generates protocol-aware reasoning rewards for non-verifiable tasks, with DACSP, which modulates cascade RL updates according to historical capability directions to reduce cross-domain interference. Experiments on math, chat, and IF benchmarks show that CARE-RL consistently improves overall performance across two backbone models and better preserves previously acquired capabilities. These results suggest that multi-domain RL requires both reliable reward construction and capability-aware optimization.

Limitations

Due to compute constraints, our experiments cover three domains (math, chat, and instruction following) and two backbones up to 7B parameters; extending CARE-RL to more domains and larger backbone scales is left to future work. In addition, PA-GRM is supervised by distilling from `DeepSeek-V4-Pro`, so its initial reward quality is bounded by teacher availability. Rigorous human screening and data curation are still needed in subsequent iterations. The current training order selection mainly aligns with the methods corresponding to MOPD for fair comparison which is the best order by validation of MOPD. In the future, we will expand the testing to assess the robustness of the methods under different domain orders.

References

Syeda Nahida Akter, Shrimai Prabhumoye, Matvei Novikov, Seungju Han, Ying Lin, Evelina Bakhturina, Eric Nyberg, Yejin Choi, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-crossthink: Scaling self-learning beyond math reasoning. *arXiv preprint arXiv:2504.13941*.

Adithya Bhaskar, Xi Ye, and Danqi Chen. 2025. Language models that think, chat better. *arXiv preprint arXiv:2509.20357*.

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

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, and 5 others. 2025. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *arXiv preprint arXiv:2506.14965*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, and 1 others. 2023. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Yizhu Jiao, Jiaqi Zeng, Julien Veron Vialard, Oleksii Kuchaiev, Jiawei Han, and Olivier Delalleau. 2025. Think twice: Branch-and-rethink reasoning reward model. *arXiv preprint arXiv:2510.23596*.

Kimi Team. 2025. Kimi K1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. WildBench: Benchmarking LLMs with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. 2025. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*.

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

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, and 2 others. 2025. Dr

tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

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

William F. Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. 2026. Rethinking rubric generation for improving LLM judge and reward modeling for open-ended tasks. *arXiv preprint arXiv:2602.05125*.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. 2025. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*.

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. 2025a. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Daniel Egert, Ellie Evans, Hoo-Chang Shin, Felipe Soares, Yi Dong, and Oleksii Kuchaiev. 2025b. Helpsteer3: Human-annotated feedback and edit data to empower inference-time scaling in open-ended general-domain tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25640–25662.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. *arXiv preprint arXiv:2506.14245*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wang Yang, Shouren Wang, Chaoda Song, Chuang Ma, Xinpeng Li, Nengbo Wang, Kaixiong Zhou, Vipin Chaudhary, and Xiaotian Han. 2026a. When domains interact: Asymmetric and order-sensitive cross-domain effects in reinforcement learning for reasoning. *arXiv preprint arXiv:2602.01365*.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2026b. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation. *arXiv preprint arXiv:2603.19220*.

Li S Yifei, Allen Chang, Chaitanya Malaviya, and Mark Yatskar. 2025. Researchqa: Evaluating scholarly question answering at scale across 75 fields with survey-mined questions and rubrics. *arXiv preprint arXiv:2509.00496*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. 2025. Rlpr: Extrapolating rlvr to general domains without verifiers. *arXiv preprint arXiv:2506.18254*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Yuyuan Zeng, Yufei Huang, Can Xu, Qingfeng Sun, Jianfeng Yan, Guanghui Xu, Tao Yang, and Fengzong Lian. 2025. Zero reinforcement learning towards general domains. *arXiv preprint arXiv:2510.25528*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Seyed Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, volume 2025, pages 12476–12505.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. 2025. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations*.

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

A Expanded Main Results

Table 8–13 provide an expanded view of the main results, with one table per domain for each backbone. Each cell reports the mean score followed by standard deviation in parentheses under the same avg@4 evaluation protocol used in Table 2.

Table 8: Expanded math results for `Qwen2.5-7B`.

Method	MATH	GSM	AIME	M. Avg.
Base	60.9 (0.54)	84.7 (0.38)	4.2 (3.19)	49.9 (0.46)
V→NV	75.9 (0.66)	90.8 (0.33)	5.8 (1.67)	57.5 (0.58)
NV→V	78.2 (0.63)	92.2 (0.31)	8.3 (1.92)	59.6 (0.69)
Naive Mixing	76.2 (0.65)	91.4 (0.32)	5.8 (1.67)	57.8 (0.59)
MGS	77.4 (0.62)	92.0 (0.30)	7.5 (1.67)	59.0 (0.63)
MOPD	78.6 (0.59)	92.7 (0.28)	7.5 (1.67)	59.6 (0.61)
CARE-RL	**79.7** (0.55)	**93.2** (0.27)	**9.2** (1.67)	**60.7** (0.58)

Table 9: Expanded chat results for `Qwen2.5-7B`.

Method	WB	CW3	RQA	C. Avg.
Base	22.1 (1.72)	43.5 (2.14)	41.1 (1.94)	35.6 (1.19)
V→NV	36.5 (2.05)	46.8 (2.26)	40.1 (2.02)	41.1 (1.31)
NV→V	35.7 (2.10)	35.4 (2.39)	43.9 (2.05)	38.3 (1.36)
Naive Mixing	35.9 (2.16)	39.2 (2.34)	42.3 (2.08)	39.1 (1.35)
MGS	37.4 (1.98)	44.5 (2.14)	43.0 (1.95)	41.6 (1.25)
MOPD	38.5 (1.86)	47.1 (2.06)	43.8 (1.88)	43.1 (1.19)
CARE-RL	**39.2** (1.72)	**48.2** (1.96)	**44.7** (1.79)	**44.0** (1.11)

Table 10: Expanded instruction-following results for `Qwen2.5-7B`.

Method	IFB	IFE
Base	15.0 (0.86)	31.2 (1.18)
V→NV	24.8 (1.05)	38.2 (1.27)
NV→V	24.4 (1.08)	39.2 (1.31)
Naive Mixing	24.2 (1.09)	38.9 (1.29)
MGS	25.3 (1.03)	40.4 (1.22)
MOPD	26.1 (0.99)	41.2 (1.17)
CARE-RL	**27.2** (0.94)	**41.8** (1.10)

Table 11: Expanded math results for `Qwen3-4B`.

Method	MATH	GSM	AIME	M. Avg.
Base	68.4 (0.58)	87.8 (0.36)	4.2 (3.19)	53.5 (0.48)
V→NV	77.6 (0.61)	92.6 (0.29)	10.8 (3.19)	60.3 (0.62)
NV→V	79.0 (0.58)	93.6 (0.27)	**14.2** (3.19)	62.3 (0.66)
Naive Mixing	78.1 (0.60)	93.0 (0.28)	9.2 (1.67)	60.1 (0.61)
MGS	78.8 (0.57)	93.7 (0.26)	13.3 (2.72)	61.9 (0.64)
MOPD	79.4 (0.54)	93.9 (0.25)	11.7 (1.92)	61.7 (0.61)
CARE-RL	**80.1** (0.51)	**94.3** (0.24)	**14.2** (3.19)	**62.9** (0.60)

Table 12: Expanded chat results for `Qwen3-4B`.

Method	WB	CW3	RQA	C. Avg.
Base	25.3 (1.66)	45.0 (2.08)	43.4 (1.91)	37.9 (1.16)
V→NV	42.0 (1.91)	48.1 (2.08)	44.5 (1.88)	44.9 (1.21)
NV→V	40.5 (2.00)	39.8 (2.28)	46.0 (1.95)	42.1 (1.31)
Naive Mixing	41.1 (1.97)	43.6 (2.18)	45.3 (1.93)	43.3 (1.27)
MGS	42.8 (1.82)	46.5 (2.00)	45.8 (1.82)	45.0 (1.15)
MOPD	43.6 (1.74)	48.3 (1.91)	46.7 (1.76)	46.2 (1.09)
CARE-RL	**44.1** (1.62)	**49.2** (1.82)	**47.5** (1.68)	**46.9** (1.02)

Table 13: Expanded instruction-following results for `Qwen3-4B`.

Method	IFB	IFE
Base	18.0 (0.90)	39.4 (1.20)
V→NV	25.6 (1.02)	45.0 (1.18)
NV→V	26.8 (1.00)	46.6 (1.16)
Naive Mixing	26.0 (1.04)	45.7 (1.19)
MGS	26.3 (0.98)	46.9 (1.12)
MOPD	26.9 (0.95)	47.6 (1.08)
CARE-RL	**27.3** (0.91)	**48.9** (1.02)

B Prompt Templates for PA-GRM

This appendix provides the prompt templates used by PA-GRM. The first prompt performs prompt-level protocol routing and schema construction, and the second prompt performs response-level generative reward scoring under the fixed schema.

PA-GRM Protocol Routing Prompt

```
You are a protocol-aware reward model for
    non-verifiable tasks.

TASK:
Given a prompt, route it to the evaluation
    protocol that best matches its task
    structure and construct a
    prompt-specific evaluation schema.

PROTOCOLS:
1. HOLISTIC
Use HOLISTIC when the prompt is open-ended
    and cannot be naturally decomposed into
    independently checkable requirements.
    HOLISTIC gives structured
    considerations for holistic judgment,
    but these considerations are not
    independent checklist items.

2. RUBRIC
Use RUBRIC when the prompt contains explicit
    structural, content, reasoning,
    formatting, safety, or
    instruction-following requirements that
    can be checked relatively independently.

ROUTING RULES:
```

```
- Route to RUBRIC if the prompt specifies
    named deliverables, multiple parts,
    explicit constraints, required
    coverage, comparison targets, output
    format, stepwise reasoning, safety
    constraints, or factual requirements.
- Route to HOLISTIC if forcing the prompt
    into checklist criteria would be
    artificial and the response should be
    judged by integrated quality.
- Build the schema from the prompt only. Do
    not inspect or assume any candidate
    response.
- Make the schema prompt-specific. Avoid
    generic dimensions unless concretely
    tied to the prompt.
- Include safety and epistemic calibration
    when relevant.

HOLISTIC SCHEMA:
Output 4 to 7 aspects:
- aspect: short name
- description: how this aspect informs
    holistic judgment

RUBRIC SCHEMA:
Output 3 to 8 criteria:
- criterion: prompt-specific requirement
- type: content, structure, format,
    reasoning, style, factuality, safety,
    or helpfulness
- weight: criterion weight, with all weights
    summing to 1.0
- check: evidence that satisfies or violates
    this criterion

OUTPUT:
Return exactly one valid JSON object:
{
  "protocol": "HOLISTIC" or "RUBRIC",
  "schema": {
    "aspects": [...]
  }
  OR
  {
    "criteria": [...]
  },
  "routing_reason": "brief explanation of
    why this protocol matches the prompt
    structure"
}

INPUT:
Prompt:
<prompt>
{prompt}
</prompt>
```

PA-GRM Generative Reward Scoring Prompt

```
You are a protocol-aware generative reward
    model for non-verifiable tasks.

TASK:
Given a prompt, a candidate response, and a
    fixed prompt-level schema, generate a
    scoring trace and extract a scalar
    reward.

CONSTRAINTS:
- Do not change the protocol or schema.
- Judge only the candidate response for the
    given prompt.
- Apply the same schema to all candidate
    responses for this prompt.
- Do not reward verbosity by itself.
- Penalize irrelevant, evasive, fabricated,
    unsafe, or instruction-violating
    content.
- If external verification is unavailable,
    judge plausibility, consistency, and
    uncertainty calibration.
- final_reward must be in [0, 1], where 1 is
    excellent and 0 is unusable or harmful.

HOLISTIC SCORING:
Generate one local judgment for each schema
    aspect. These judgments support
    holistic assessment, but final_reward
    is not a simple average.
Each judgment contains:
- name: aspect name
- assessment: specific judgment
- evidence: short response evidence or
    missing evidence

RUBRIC SCORING:
Generate one item-level judgment for each
    criterion. The final reward should
    follow the weighted criteria, while
    severe safety or instruction-following
    failures may cap the score.
Each judgment contains:
- name: criterion name
- assessment: whether the response satisfies
    the criterion
- evidence: short response evidence or
    missing evidence
- local_score: score in [0, 1]
- weighted_contribution: local_score
    multiplied by criterion weight

OUTPUT:
Return exactly one valid JSON object:
{
  "protocol": "HOLISTIC" or "RUBRIC",
  "intermediate_judgments": [
    {
      "name": "...",
      "assessment": "...",
      "evidence": "...",
      "local_score": 0.0,
      "weighted_contribution": 0.0
    }
  ],
  "reasoning": "concise overall scoring
    rationale",
  "final_reward": 0.0
}

For HOLISTIC, omit local_score and
    weighted_contribution.
For RUBRIC, include local_score and
    weighted_contribution.

INPUTS:
```

```
Fixed protocol:
{protocol}

Fixed schema:
{schema}

Prompt:
<prompt>
{prompt}
</prompt>

Candidate response:
<response>
{response}
</response>
```

C Checkpoint surgery validation.

To test whether the leading singular directions actually carry acquired capabilities, we apply checkpoint surgery to the three `Qwen3-4B` single-domain updates in Table 3. For each update, we keep only the DACSP top subspace, remove that subspace, or apply the same edits to a random per-layer subspace with equal rank.

Table 14: Checkpoint surgery on `Qwen3-4B` single-domain updates. Scores are target-domain averages; Ret. Gain is the retained fraction of the full single-domain gain.

Edited checkpoint	Math	Chat	IF	Ret. Gain (%)
Base	53.5	37.9	28.7	–
Full update	60.4	43.3	38.6	100.0
Top only	59.6	42.9	37.8	91.0
Random only	52.1	38.8	30.5	4.9
Remove top	51.8	38.7	30.1	1.4
Remove random	59.2	42.7	37.4	86.5

Table 14 supports the capability direction interpretation. The top subspace alone preserves most of the target-domain gain, while an equal-rank random subspace retains little. Conversely, removing the top subspace eliminates most of the gain, whereas removing a random subspace leaves the edited checkpoint close to the full update. These results indicate that DACSP's historical bases capture update structure tied to acquired capability, not just optimizer noise or LoRA artifacts. The small residual gains outside the top subspace also show that the basis remains an approximation.