GRPO-JEPA: MITIGATING SEMANTIC DRIFT IN GRPO VIA JOINT EMBEDDING PREDICTIVE ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) is a standard method for RL post-training of large language models, yet its objective stays in the output space: neither the policy gradient nor the token-level KL term observes representation-space dynamics. This blind spot lets semantic drift occur while the loss stays low. We present GRPO-JEPA, which brings the Joint-Embedding Predictive Architecture (JEPA) into LLM RL post-training as explicit representation-space supervision: an advantage-weighted InfoNCE signal turns GRPO's group-sampled completions into multi-view data, propagating scalar reward signals into representation-space alignment. This JEPA supervision signal enters training in two ways: (i) Method A (direct regularization) adds it to the GRPO objective, and (ii) Method B (soft modulation) converts it into a dynamic controller of the KL coefficient. Across four benchmarks (Spider, GSM8K, HellaSwag, and Yelp) and five models from four families (Qwen, OLMo, SmolLM, and Gemma), GRPO-JEPA improves on standard GRPO on all 20 model–dataset pairs: average gains are 5.9 points for Method A and 4.6 for Method B. It also surpasses recent GRPO variants (SAPO, BNPO, and DAPO) on every benchmark for Method A and on average for Method B. Representation-space predictive consistency offers a usable JEPA supervision signal for RL post-training.

1 INTRODUCTION

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates the Critic network (Schulman et al., 2017) through intra-group relative advantage estimation and is now a standard paradigm for LLM RL post-training. Its reward and policy loss act on output tokens, and the KL divergence constrains per-token deviation from the reference model; neither acts on the policy's internal representations. This insensitivity has documented consequences: self-reinforcing likelihood decline, mode collapse, and alignment collapse all implicate the representation space (Deng et al., 2025; Liang et al., 2026; Gauthier et al., 2026), since token-level objectives do not determine hidden-state movement between prompt and answer (Pan et al., 2026; Wu & Tang, 2026). The constraint therefore fails in both directions: it under-penalizes semantic drift while token distributions stay close to the reference model, yet over-constrains the updates that would correct it in verifiable-reward tasks (Liu et al., 2025).

Under-penalization of semantic drift is most evident in Text-to-SQL tasks with set-overlap rewards (section B.2). Under Jaccard similarity between executed result sets, a query that retrieves the entire table (e.g., `SELECT * FROM table` as Shortcut SQL) earns partial credit despite low precision. Each additional filtering clause moves the result closer to gold, but the reward assigns -1 to a non-executable query. The shortcut is a rational local optimum: GRPO's advantage can only rank the sampled candidates, not assess whether they are semantically adequate. Models trained with GRPO often collapse into syntactically correct but semantically drifting shortcuts (Wu & Tang, 2026) while their token distributions stay close to the reference model, so KL divergence cannot detect the collapse. In fig. 1(a) the baseline's shortcut rate peaks above 50% and stays high, while Method A suppresses it and Method B mitigates it partially.

We call the phenomenon shared by these failures **semantic drift**: the policy's latent representation of its answer deviates from the semantics of the gold answer. Two measures capture it: behaviorally, the *shortcut rate*, the fraction of syntactically valid but semantically drifting completions (fig. 1(a)); and

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.

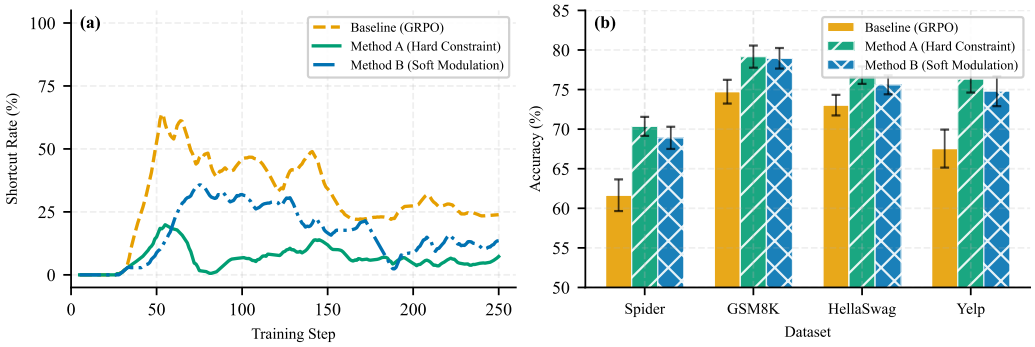

Figure 1: (a) Shortcut SQL rate on Spider over training steps (Qwen2.5-3B, $G = 64$). (b) Final accuracy on all four benchmarks (Qwen2.5-3B, mean $\pm$ std over five seeds).

in representation space, the cosine similarity S_{ans} between the query-conditioned prediction and the gold-answer embedding (section 3.2).

JEPA as a semantic monitor. Monitoring such drift requires a signal defined in representation space itself. The Joint-Embedding Predictive Architecture (JEPA) (LeCun, 2022) provides such a signal: instead of reconstructing tokens, it predicts a target view's representation from a context view, yielding semantically abstract representations (Assran et al., 2023; Bardes et al., 2024). Its similarity to the gold answer is a directly monitorable quantity. LLM-JEPA (Huang et al., 2026) validated this for language models by aligning natural-language and code views, but only under supervised fine-tuning, where the views are static and pre-aligned with the target. GRPO violates both assumptions: (a) it draws the views from the current policy as training proceeds, so the views within one group vary in quality and none is pre-aligned with the gold answer, and (b) the GRPO objective keeps improving while nothing anchors the representations, so drift draws no penalty.

We address *how to turn GRPO's sparse scalar rewards into a continuous representation-space signal using the policy's own sampled completions as views*. GRPO supplies both missing ingredients. Group sampling provides multiple completions per problem as advantage-weighted multi-view data, resolving (a). For (b), the gold answer, encoded by the frozen reference model, is the only positive anchor the predictive embedding is attracted to, so supervision keeps a non-drifting anchor.

We make three contributions:

1. **JEPA supervision signal for RL post-training.** An advantage-weighted InfoNCE signal that turns GRPO's group sampling and advantages into representation-space supervision, anchored to the gold answer by the frozen reference model.
2. **Two injection strategies.** Method A (direct regularization) adds the signal to the GRPO objective; Method B (soft modulation) uses it to modulate the KL coefficient. Both add one forward pass and require no extra model.
3. **Validation.** Both variants improve on all 20 model–dataset pairs, and the representation-space dynamics show that the JEPA signal, not the shared task loss, drives the gain.

2 Related Work

2.1 GRPO Stability Improvements and Dynamic Regularization

GRPO (Shao et al., 2024) eliminates PPO's (Schulman et al., 2017) Critic network through intra-group relative advantage, yet training instability remains prominent. Several works redesign the surrogate objective or its clipping: DisCO (Li et al., 2025), BAPO (Xi et al., 2026), SAPO (Gao et al., 2025), and GSPO (Zheng et al., 2025) replace hard clipping with constrained optimization, adaptive clipping, smooth gating, or sequence-level importance weights. Others adjust the regularization or reward structure: DAPO removes the KL penalty with dynamic sampling (Yu et al., 2025), BNPO normalizes

binary rewards via a Beta distribution (Xiao et al., 2025), and GDPO decouples normalization across multiple rewards (Liu et al., 2026). None of these methods constrains the representation space.

We borrow the idea for Method B from dynamic KL coefficients: InstructGPT (Ouyang et al., 2022) uses a PI controller to hold a target KL budget, and FRPO (Yao et al., 2026) constrains future f-divergence for process-level credit assignment. Both treat KL divergence itself as the control target rather than a proxy for semantic state, and so cannot detect collapse in representation space.

2.2 JEPA AND PREDICTIVE REPRESENTATION LEARNING

JEPA (LeCun, 2022) replaces pixel- or token-level reconstruction with conditional embedding prediction, learning structural dependencies in representation space. I-JEPA and V-JEPA validate this paradigm on images and video (Assran et al., 2023; Bardes et al., 2024).

Later work applies the paradigm across domains, including self-supervised depth estimation (Grigore & Popa, 2026), vision-language-action policies (Sun et al., 2026a), imitation learning (Vujinović & Kovačević, 2026), and visual RL from offline transitions (Bagatella et al., 2026). Closer to our setting, MIND-V penalizes physically implausible dynamics in latent feature space (Zhang et al., 2025), and AtomVLA evaluates candidate action chunks with a predictive latent world model (Sun et al., 2026b).

LLM-JEPA (Huang et al., 2026) applies the same predictive objective to language models, but only under supervised fine-tuning, where the views are fixed and pre-aligned. We are not aware of prior work that applies JEPA to the RL training of large language models; world-model robotics, the nearest RL-side prior, supervises physical plausibility rather than question–answer semantics.

3 METHOD

3.1 DESIGN MOTIVATION

We posit that JEPA stabilizes GRPO training by constraining predictive consistency in representation space. The next subsections define the three embeddings (section 3.2), the advantage-weighted InfoNCE signal (section 3.3), and the two injection strategies.

3.2 EMBEDDING DEFINITIONS

We define three embeddings over the prompt $\mathbf{x}_{\text{user}}$, the gold answer $\mathbf{y}_{\text{ans}}$, and each sampled completion $\mathbf{y}_{\text{comp}_j}$:

- **Predictive Embedding** E_{pred} (gradient enabled): Append k special predictor tokens $\langle|\text{predictor}_k|\rangle \dots \langle|\text{predictor}_1|\rangle$ to the user query $\mathbf{x}_{\text{user}}$ in descending index order and perform a forward pass with the current policy, extracting the hidden state at the last predictor-token position (no pooling is applied):
$$E_{\text{pred}} = \text{LLM}(\mathbf{x}_{\text{user}} \parallel \langle|\text{predictor}_k|\rangle \dots \langle|\text{predictor}_1|\rangle)[\text{idx}_{\text{last}}] \tag{1}$$
where k is a hyperparameter inherited from LLM-JEPA (Huang et al., 2026) (a sweep over k is reported in section D) and $\mathbf{x}_{\text{user}}$ denotes the raw user query before embedding. The predictor tokens are added to the tokenizer as new special tokens before training and appear in no other forward pass.
- **Answer Embedding** E_{ans} (gradient blocked): Encode the gold answer $\mathbf{y}_{\text{ans}}$ with the frozen reference model and pool over the query–answer sequence:
$$E_{\text{ans}} = \text{LLM}_{\text{ref}}(\mathbf{x}_{\text{user}} \parallel \mathbf{y}_{\text{ans}})\,[\text{idx}_{\text{last}},\ \text{pool}_{\text{mean}}] \tag{2}$$
where we take the embedding either as the hidden state at the last token position (idx_{last}) or as global mean pooling ($\text{pool}_{\text{mean}}$) over the concatenated query–answer sequence; the two choices are compared in table 7.
- **Completion Embedding** E_{comp_j} (gradient blocked): Encode the j-th sampled completion $\mathbf{y}_{\text{comp}_j}$ within the group with the reference model and pool over the query–completion sequence similarly:
$$E_{\text{comp}_j} = \text{LLM}_{\text{ref}}(\mathbf{x}_{\text{user}} \parallel \mathbf{y}_{\text{comp}_j})\,[\text{idx}_{\text{last}},\ \text{pool}_{\text{mean}}] \tag{3}$$

Both E_{ans} and E_{comp} reuse the frozen reference model that we already keep for the KL term, so the method adds no second model.

3.3 The JEPA Supervision Signal

The JEPA signal. Given the three embeddings, we compute the signal via advantage-weighted InfoNCE over each group, taking E_{ans} as the only positive anchor and the completion embeddings as soft negatives:

$$\mathcal{L}_{\text{JEPA}} = -\log\left(\frac{\exp(S_{\text{ans}})}{\exp(S_{\text{ans}}) + \sum_j w_j \cdot \exp(S_{\text{comp}_j})}\right) \tag{4}$$

where $S_{\text{ans}} = \cos(E_{\text{pred}}, E_{\text{ans}})$ and $S_{\text{comp}_j} = \cos(E_{\text{pred}}, E_{\text{comp}_j})$ are cosine similarities, and $w_j = \exp(-\hat{A}_j)$ weights each completion by its intra-group relative advantage $\hat{A}_j$. The InfoNCE temperature is fixed at 1 throughout; section D.2 gives the rationale and the supporting cross-dataset sweep.

The exponential weighting $w_j = \exp(-\hat{A}_j)$ sets the push–pull balance of the signal in representation space: w_j scales the completion's term in the InfoNCE denominator, so a larger weight makes the signal push E_{comp_j} further from E_{pred}. A positive advantage ($\hat{A}_j > 0$) gives $w_j < 1$ and drives the weight toward 0 as the advantage grows, so completions the reward already ranks as good receive almost no repulsion. A negative advantage ($\hat{A}_j < 0$) gives $w_j > 1$ and raises the weight as the advantage becomes more negative, so the worst completions are pushed hardest. The signal is therefore graded rather than selective, and bounded throughout training: the weight varies continuously with $\hat{A}_j$, with no threshold at which a completion stops contributing, and the clamp on $\hat{A}_j$ of table 2 bounds the largest weight.

Two consequences follow for training. First, completions serve only as soft negatives, never as additional positives: the gold answer is the only positive anchor, whereas even high-advantage completions reflect the model's own current state, and attracting E_{pred} toward them risks reinforcing its own errors. Second, when all rewards in a group are equal (the advantage-starvation regime of Liang et al. (2026)), every advantage $\hat{A}_j$ is zero, so $w_j = 1$, reducing the signal to uniform contrastive repulsion toward E_{ans}. GRPO-JEPA therefore still receives a JEPA signal in the advantage-starvation regime.

3.4 Joint Optimization Strategies

The standard GRPO objective is:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}\left[\min\left(\rho\hat{A},\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)\hat{A}\right)\right] + \beta \cdot D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \tag{5}$$

written in loss form (to be minimized), where $\rho = \pi_\theta(\mathbf{y} \mid \mathbf{x}) / \pi_{\theta_{\text{old}}}(\mathbf{y} \mid \mathbf{x})$ is the importance-sampling ratio and ϵ is the clipping range. Neither term of this objective observes representation-space signals. We add the JEPA signal in one of two ways.

Method A: direct loss regularization. We add the JEPA supervision signal as an explicit penalty term to the total objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}} + \lambda \cdot \mathcal{L}_{\text{JEPA}} \tag{6}$$

where $\mathcal{L}_{\text{GRPO}}$ is the standard GRPO policy surrogate loss and $\lambda \geq 0$ is a hyperparameter that sets how strongly the signal enters the total objective; $\lambda \cdot \mathcal{L}_{\text{JEPA}}$ is the *JEPA loss*.

A sharp drift in the policy's representations raises $\mathcal{L}_{\text{JEPA}}$, whose gradient pulls the update back toward semantically stable regions, strengthening both the E_{pred}–E_{ans} alignment and the E_{pred}–E_{comp} separation. This restores a gradient path from representation space to the objective, which the KL term alone does not provide. Method B carries the same signal through β_{dyn} instead of a loss term, and both methods compute the predictor embedding with gradients enabled.

Method B: dynamic β modulation. Method B leaves the GRPO objective unchanged and uses the JEPA signal as a scalar controller of the KL coefficient:

$$\beta_{\text{dyn}} = \max\Big(2\beta_0 \cdot \sigma\Big(\frac{z}{2}\Big),\ \beta_{\min}\Big), \qquad z = \text{clip}\Big(\frac{\mathcal{L}_{\text{JEPA}} - \tau_{\text{ema}}}{\sigma_{\text{ema}}},\ -5,\ 5\Big) \tag{7}$$

where σ is the sigmoid, and τ_{ema} and σ_{ema} are the exponential moving average and standard deviation of the raw $\mathcal{L}_{\text{JEPA}}$. The z-score removes any dependence on the signal's absolute scale, so the controller transfers across datasets and model families without retuning; both statistics are computed online, so Method B adds no threshold or sensitivity hyperparameter beyond the nominal KL coefficient β_0. The floor $\beta_{\min} = 0.1\,\beta_0$ is a safeguard that our runs never reach, since the clip on z keeps $2\sigma(z/2) \geq 0.15$. The EMA statistics use a decay of 0.95 and are updated only at optimizer steps; during a 20-step warmup the controller is inactive and $\beta = \beta_0$.

As $\mathcal{L}_{\text{JEPA}}$ rises above τ_{ema}, β_{dyn} increases toward $2\beta_0$, tightening the KL constraint; as it falls below, β_{dyn} decays toward $\beta_{\min}$, relaxing the constraint. β_{dyn} therefore tracks the representation-space state instead of a fixed strength.

4 Experiments

Unless otherwise noted, training curves are smoothed with an EMA span of 5 logging points. We average the main results over five random seeds and report standard deviations; the appendix runs (the hyperparameter and design sweeps, the pooling and cross-dataset grids, and the method-combination analysis of section E) use a single training run at seed 42. Table 2 lists the full hyperparameter configuration.

4.1 Final Performance

Does explicit semantic supervision help across models and datasets, and does it beat tuned GRPO variants? Table 1 answers both.

Model	Method	Spider	GSM8K	HellaSwag	Yelp	Avg. Δ
Qwen2.5-3B	Baseline (GRPO)	61.65$\pm$2.0	74.73$\pm$1.5	73.03$\pm$1.3	67.54$\pm$2.4	—
	Method A	**70.35**$\pm$1.2	**79.16**$\pm$1.4	**76.82**$\pm$1.1	**76.32**$\pm$1.7	+6.4
	Method B	68.90$\pm$1.4	78.95$\pm$1.3	75.60$\pm$1.2	74.80$\pm$1.9	+5.3
	SAPO (Gao et al., 2025)	68.58$\pm$1.1	76.85$\pm$1.7	74.25$\pm$1.1	73.25$\pm$1.8	+4.0
	BNPO (Xiao et al., 2025)	69.15$\pm$0.5	77.25$\pm$0.5	72.23$\pm$0.9	72.14$\pm$1.1	+3.5
	DAPO (Yu et al., 2025)	66.28$\pm$1.8	74.91$\pm$1.2	73.89$\pm$1.4	75.25$\pm$0.5	+3.3
Qwen2.5-7B	Baseline (GRPO)	72.45$\pm$1.2	73.58$\pm$1.1	81.33$\pm$0.8	69.68$\pm$0.8	—
	Method A	80.54$\pm$0.7	75.42$\pm$1.4	82.54$\pm$1.1	**72.58**$\pm$1.6	+3.5
	Method B	**80.87**$\pm$0.8	**76.95**$\pm$1.2	**83.69**$\pm$1.4	71.25$\pm$1.2	+3.9
OLMo2-1B	Baseline (GRPO)	40.90$\pm$2.1	30.96$\pm$1.8	37.25$\pm$1.5	41.34$\pm$2.3	—
	Method A	**45.78**$\pm$1.6	**33.24**$\pm$1.4	**39.15**$\pm$1.2	**47.61**$\pm$1.9	+3.8
	Method B	44.12$\pm$1.9	32.85$\pm$1.5	38.70$\pm$1.4	46.20$\pm$2.1	+2.9
SmolLM2-3B	Baseline (GRPO)	66.81$\pm$1.8	67.17$\pm$2.2	66.63$\pm$1.9	62.69$\pm$2.5	—
	Method A	**72.63**$\pm$1.3	70.22$\pm$1.6	**70.75**$\pm$1.4	**66.86**$\pm$1.8	+4.3
	Method B	71.20$\pm$1.5	**72.10**$\pm$1.8	70.67$\pm$1.7	64.50$\pm$2.0	+3.8
Gemma2-2B	Baseline (GRPO)	53.59$\pm$2.6	31.67$\pm$2.8	39.12$\pm$2.1	54.10$\pm$2.7	—
	Method A	**67.51**$\pm$1.5	**43.64**$\pm$2.0	**51.42**$\pm$1.8	**62.74**$\pm$2.2	+11.7
	Method B	64.30$\pm$1.8	33.68$\pm$2.2	48.70$\pm$2.0	60.50$\pm$2.5	+7.2

Table 1: Final accuracy (%) across four benchmarks and five models from four model families ($n = 5$ seeds; Avg. Δ is the mean gain over the GRPO baseline). SAPO, BNPO, and DAPO use the baseline's hyperparameters.

Protocol. We compare the GRPO baseline, Method A, and Method B on Spider (Yu et al., 2018), GSM8K (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019), and Yelp (Zhang et al., 2015) across Qwen (Qwen et al., 2025), OLMo (Walsh et al., 2025), SmolLM (Allal et al., 2025), and Gemma (Team et al., 2024) models, and evaluate SAPO (Gao et al., 2025), BNPO (Xiao et al., 2025),

and DAPO (Yu et al., 2025) on Qwen2.5-3B at the baseline's hyperparameters and the same seeds. Metrics are execution accuracy (Spider), exact match (GSM8K), option accuracy (HellaSwag), and label accuracy (Yelp). Each dataset supplies E_{ans} through a *code view*: the gold SQL for Spider, which is why our dynamics analyses center on it, and a semantic target for the other three. Section B gives the construction and prompt template, and section A the tuning protocol.

Main results. Table 1 reports the numbers: Method A takes the larger gain on 16 of the 20 pairs, and Method B's mean of 74.6 exceeds the best variant mean of 73.2, although Method B does not lead on every individual benchmark. The combined variant gives the best average accuracy on Qwen2.5-3B (section E); that run is single-seeded, so we read the margins as suggestive.

Summary. Both variants help beyond the datasets and models they were tuned on: they improve on the GRPO baseline on all 20 model–dataset pairs, Method A beats the tuned GRPO variants on every benchmark, and the two injections stack, with the combined variant ahead of both single-method arms on average (section E).

4.2 Necessity and Stability under Explicit Semantic Supervision

Does optimizing the GRPO objective already minimize the JEPA supervision signal, and what does adding that signal cost? The GRPO baseline logs $\mathcal{L}_{\text{JEPA}}$ without optimizing it, so if the GRPO objective minimized the signal implicitly, the baseline's curve would fall as reward improves. Figures 2 and 3 compare the two methods against the baseline along four quantities, and section C reports the compute cost.

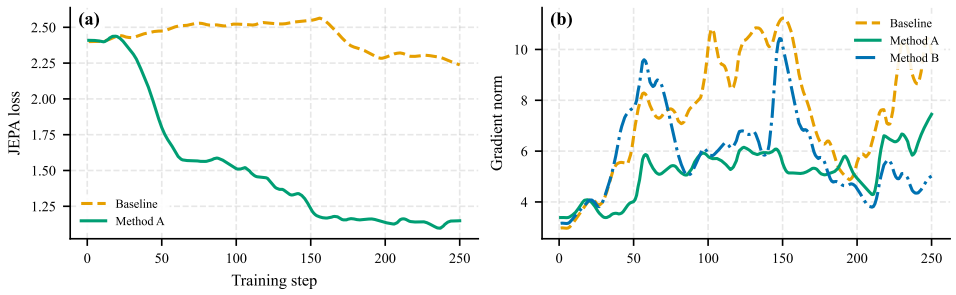

Figure 2: Necessity of explicit semantic supervision on Qwen2.5-3B / Spider. (a) Monitored $\mathcal{L}_{\text{JEPA}}$; Method B does not log this quantity. (b) Gradient norm.

JEPA supervision signal (fig. 2(a)). Under the baseline, the monitored $\mathcal{L}_{\text{JEPA}}$ is a flat line with noise: it stays inside a 2.2–2.6 band, about $\pm 7\%$ of its mean, and its excursions follow neither the task reward ($r = 0.12$) nor the realized KL ($r = -0.28$). Method A instead falls steeply and then stops, from 2.4 to 1.6 by step 100 and little further after step 150; the two runs differ only in whether $\mathcal{L}_{\text{JEPA}}$ enters the GRPO objective.

Gradient norm (fig. 2(b)). The extra forward pass adds JEPA gradients that could inflate the update magnitude and destabilize training under Method A. For most of the run the opposite holds: after the warmup Method A is the lowest and flattest of the three curves, never exceeding 7, while the baseline crests near 10 and Method B near 8. The baseline dips after its crest and climbs back at the end; Method B decays from its crest on; and Method A climbs late to its peak of 6.6, which stays below the baseline's closing 8.5.

Realized KL (fig. 3(c)). All three arms stay flat through the warmup, surge once, and settle back. The baseline and Method B then surge a second time, the baseline overshooting 2.0, while Method A never leaves the band it enters at step 60 and stays around half the baseline's level. Method B's KL controller, driven by the JEPA signal, moves the coefficient only within 0.89–1.16 β_0, yet that narrow range brings its late plateau about 19% below the baseline's. Method A uses the same KL term and

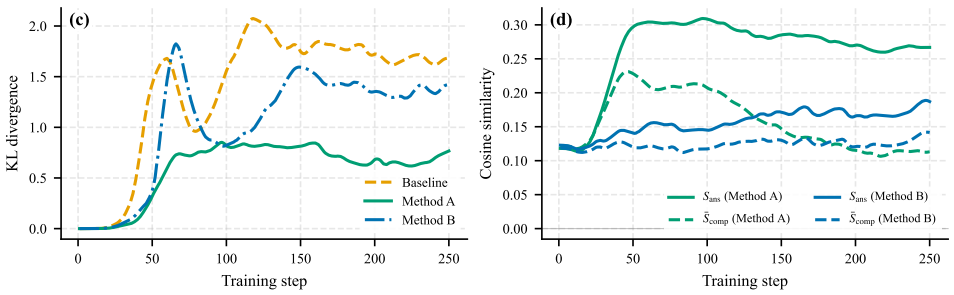

Figure 3: Training dynamics under each method on Qwen2.5-3B / Spider. (c) Realized KL; the baseline and Method A use a fixed KL coefficient. (d) S_{ans} (solid) and $\bar{S}_{\text{comp}}$ (dashed).

coefficient as the baseline, so its flatter curve comes from the JEPA loss rather than from a tighter KL budget.

Similarity geometry (fig. 3(d)). The loss should pull the prediction toward the gold answer and away from poor completions, and panel (d) shows it does: under Method A, S_{ans} rises and then holds while the group-mean completion similarity $\bar{S}_{\text{comp}}$ declines, so the two separate over training. Method B separates them in the same direction but weakly: its S_{ans} ends about 0.04 above $\bar{S}_{\text{comp}}$, against about 0.15 under Method A. The direct loss term opens the wider gap, and the KL modulation reproduces it at about a quarter of the size.

Compute cost. Each step adds one forward pass, about $0.3\times$ the cost of a standard GRPO step (section C).

Summary. The JEPA signal is not implicit in the GRPO objective, and adding it costs no stability: the updates stay the smallest and least jumpy, the realized KL stays flat, the alignment gap widens, and the extra forward pass does not lower the final accuracy. All four monitored quantities settle in the middle of the run while the task reward keeps rising, and the design ablations of table 5 show that the JEPA computation, not the shared task loss, drives the gain.

4.3 Embedding Geometry at Inference

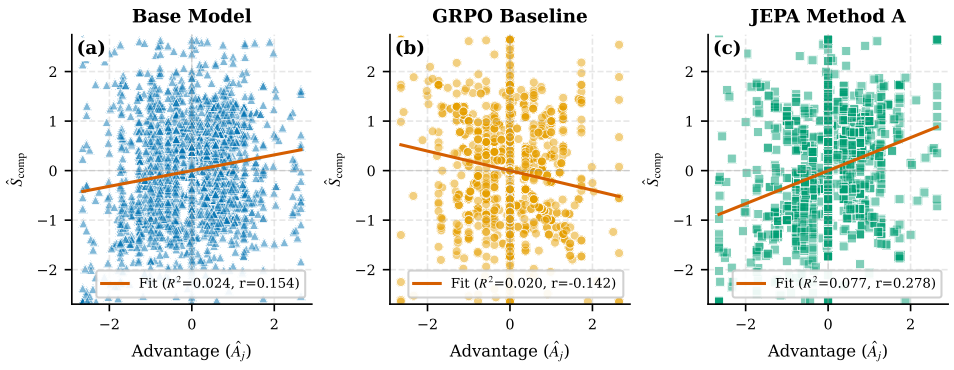

Figure 4: Intra-group relative advantage $\hat{A}_j$ against completion similarity $\hat{S}_{\text{comp}}$ on Spider (Qwen2.5-3B). Each point is one sampled completion; $\hat{S}_{\text{comp}}$ is normalized within each prompt's candidate group. (a) Base model. (b) GRPO baseline. (c) Method A.

The training dynamics show what the signal does during optimization. We then ask whether it also reshapes the geometry at inference, probing the final checkpoint of the Qwen2.5-3B / Spider run with three analyses: advantage–similarity coupling, the singular-value structure of the residuals, and

the t-SNE embedding geometry. All use 300 evaluation cases, 8 completions per case. The three analyses examine Method A, the larger of the two single-method gains, against the base model and the GRPO baseline.

Advantage–similarity coupling (fig. 4). Figure 4 tests whether output-space quality reaches representation space, plotting per-candidate advantage against completion similarity normalized within each prompt's candidate group. The base model and the GRPO baseline show little coupling, so the reward ranks completions without moving the embedding geometry, and the GRPO baseline trends slightly negative ($r = -0.142$). Method A shows the strongest coupling ($r = 0.278$, about $1.8\times$ the base model's 0.154; $R^2 = 0.077$), so the signal tilts the geometry in the intended direction without dominating it. Since the loss repels low-advantage completions by construction, the correlation checks that the mechanism is self-consistent.

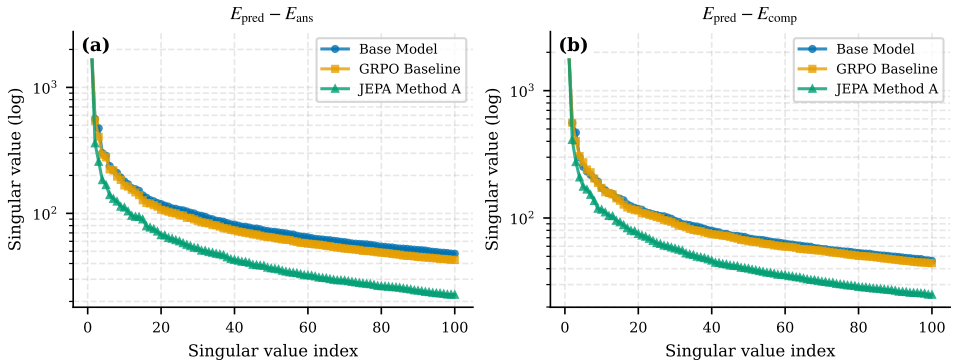

Figure 5: Singular-value spectra of the residual embeddings $E_{\text{pred}} - E_{\text{ans}}$ (left) and $E_{\text{pred}} - E_{\text{comp}}$ (right) on Spider (Qwen2.5-3B), sorted from largest to smallest.

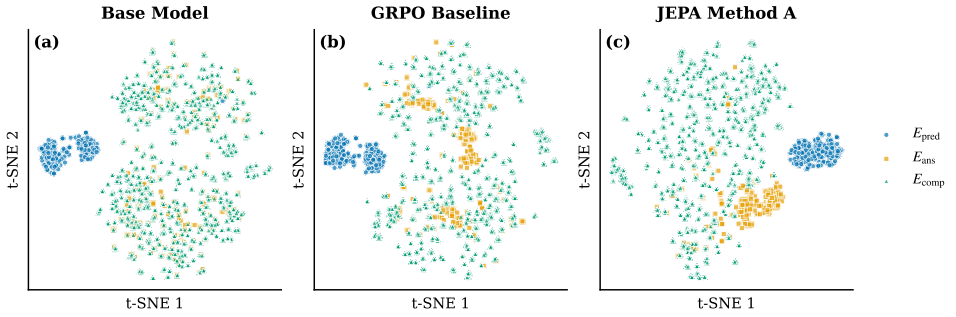

Figure 6: t-SNE geometry on Spider (Qwen2.5-3B) for the base model, the GRPO baseline, and Method A, with all embeddings extracted at inference.

Singular-value structure (fig. 5). Each panel sorts one residual's singular values, so the leftmost point carries the most energy. Method A's curve lies below both baselines at every index, and its separation from the base model's widens toward the right, so the two spectra differ in shape rather than by a constant factor.

The summary statistics agree for $E_{\text{pred}} - E_{\text{ans}}$ over the leading 100 directions: the effective rank falls from 4.0 (base model) and 3.6 (GRPO baseline) to 1.9, and the top 10 directions hold 96.5% of the residual energy against 90.4% and 91.5%. The $E_{\text{pred}} - E_{\text{comp}}$ panel gives the same ordering at slightly different values (3.8 and 3.6 against 2.0; 90.9%, 91.6%, and 96.4%). Both statistics are invariant to a uniform rescaling, so they measure the shape of the spectrum rather than its size, and the base model sits above the GRPO baseline rather than below it: GRPO training alone leaves the spectrum close to the base model's, and only Method A concentrates the energy.

Figure 6 examines the inference-time representation space of the three models. Clear structure emerges under Method A: E_{pred}, E_{ans}, and the completion cloud occupy distinct regions, whereas under the base model and the GRPO baseline E_{ans} stays mixed into the completion cloud.

Summary. The gain is also visible at inference: the prediction sits closer to the gold answer than under the baseline, the residual between prediction and answer concentrates in a few directions, and E_{ans} leaves the completion cloud. The JEPA signal reshapes the geometry rather than rescaling it. Method A induces better-structured representations than the base model and the GRPO baseline.

4.4 Ablations and Sensitivity

Which design choices carry the gain? The ablations in section D isolate them: removing the predictor tokens costs 3.1 points, letting gradients flow into E_{comp} collapses training to 58.5%, and using uniform or linear advantage weighting instead costs 4.9–8.2 points relative to the exponential weighting.

How much do hyperparameter changes affect the performance gain? The sweeps show a wide tolerance: Method A tolerates λ across a $50\times$ range, and Method B has a floor at low β_0 that is consistent with the controller's invariance to loss scale (section 3); the predictor-token count, the pooling strategy, and the per-dataset λ / β_0 grids reproduce these orderings (tables 6 to 8).

Summary. The gain comes from the architecture rather than from tuning: the predictor tokens, the stop-gradient on the targets, and the exponential weighting are load-bearing, while every added coefficient keeps a wide stable region.

5 Conclusion and Limitations

Conclusion. GRPO optimizes a reward over sampled completions and never observes whether the model's internal representation of an answer stays aligned with the gold answer; GRPO-JEPA supplies that signal from state the algorithm already produces: group-sampled completions as multiple views, and the frozen reference model as a non-drifting anchor. Both injection strategies, a direct loss term on the representation and a modulation of the KL coefficient, improve on standard GRPO across four benchmarks and five models, and the gain comes from the architecture rather than from tuning. Since the GRPO objective does not minimize the signal on its own, and adding it costs no stability or final accuracy, representation-space consistency is supervision the RL objective does not supply. The same construction should carry over to other post-training settings where the output is verifiable but the representation is not watched.

Limitations. Limited compute and time kept the largest model at 7B and left the hyperparameter search partial rather than exhaustive. The remaining recent GRPO variants discussed in the related work are also not compared, since the training library *trl* does not support them. All benchmarks supply a gold answer as the anchor, leaving rule-based-judgment-only and open-ended generation out of scope. GRPO-JEPA is more informative when the gold answer itself carries explicit semantics: adding a rationale such as a `<thought>` target helps in our setting (table 4), so the method may favor datasets whose answers are semantically rich, and how to synthesize such rationales when a dataset supplies only a bare label is outside our scope.

AI Use Statement

In this work, generative AI tools were used for drafting and editing the paper text, for creating and editing the code that generates the paper's figures, and for generating the rationales used to construct the Yelp code view (section B.5). These tools were not used for hypotheses, theoretical or conceptual frameworks, methodology or experiment design, implementing the method, result interpretation, or dataset cleaning and reformatting; those rationales are the only AI-generated data in this work. They serve as E_{ans}, and on Yelp also as the reference text for the reward's rationale-similarity term, so the reward there partially measures agreement with the generated rationale. Mathematical claims and ingredients for proofs, writing proofs, translation, and qualitative or thematic analysis are not

applicable to this work. All AI-assisted work has been reviewed; the authors conceived and verified all ideas, experimental designs, analyses, and claims, and they take responsibility for the final content of this work.

Ethics Statement

The authors have read and adhere to the ICLR Code of Ethics. This work does not involve human subjects, personal data, or new dataset releases; all experiments use publicly available benchmarks (Spider, GSM8K, HellaSwag, Yelp) and publicly available open-weight models. The method is a training-dynamics regularizer and does not introduce new deployment capabilities beyond standard RL post-training; we note that improved alignment stability could amplify both beneficial and harmful downstream uses of the underlying models, and there is no conflict of interest or sponsorship affecting this work.

Reproducibility Statement

We specify the method in section 3 and name and tabulate every hyperparameter in table 2. The protocol, seeds, and smoothing are in section 4 and the tuning protocol in section A; per-dataset metric definitions in section 4.1; dataset construction, code views, and prompt templates in section B; and the design sweeps in tables 5 and 6. The training and evaluation code are released in an anonymized GitHub repository: https://github.com/GRPO-JEPA/GRPO-JEPA.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. *arXiv preprint arXiv:2502.02737*, 2025.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SzXDuBN8M1.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

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.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On group relative policy optimization collapse in agent search: The lazy likelihood-displacement. *arXiv preprint arXiv:2512.04220*, 2025.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Etienne Gauthier, Francis Bach, and Michael I Jordan. Explaining and preventing alignment collapse in iterative rlhf. *arXiv preprint arXiv:2605.04266*, 2026.

Ionuţ Grigore and Călin-Adrian Popa. Jepadepth: Masked predictive representation learning for self-supervised monocular depth estimation. *arXiv preprint arXiv:2607.26600*, 2026.

Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: Large language models meet joint embedding predictive architectures. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GbXKPo9QfH.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf. Version 0.9.2.

Gang Li, Ming Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. DisCO: Reinforcing large reasoning models with discriminative constrained optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zzUXS4f91r.

Zhenwen Liang, Yujun Zhou, Sidi Lu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Too correct to learn: Reinforcement learning on saturated reasoning data. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 205–215, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-391-3. doi: 10.18653/v1/2026.acl-short.19. URL https://aclanthology.org/2026.acl-short.19/.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Yurui Pan, Ke Xu, and Bo Peng. Topology-enhanced alignment for large language models: Trajectory topology loss and topological preference optimization. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24807–24821, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1242. URL https://aclanthology.org/2026.findings-acl.1242/.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, et al. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2026.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026a.

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, et al. Atomvla: Scalable post-training for robotic manipulation via predictive latent world models. *arXiv preprint arXiv:2603.08519*, 2026b.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, et al. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv preprint arXiv:2408.00118*, 2024.

Aleksandar Vujinović and Aleksandar Kovačević. Act-jepa: Novel joint-embedding predictive architecture for efficient policy representation learning. *IEEE Access*, 14:78895–78906, 2026. ISSN 2169-3536. doi: 10.1109/access.2026.3696039. URL http://dx.doi.org/10.1109/ACCESS.2026.3696039.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Rui Wu and Ruixiang Tang. From rebound to remedy: Understanding and mitigating reward hacking via representation engineering. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=LZEBw6n0pB.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Xun Deng, Zhihao Zhang, Honglin Guo, Zhikai Lei, Miao Zheng, Guoteng Wang, Peng Sun, Rui Zheng, Hang Yan, Tao Gui, Qi Zhang, and Xuanjing Huang. BAPO: Stabilizing off-policy reinforcement learning for LLMs via balanced policy optimization with adaptive clipping. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jIeJJqG7dz.

Changyi Xiao, Mengdi Zhang, and Yixin Cao. Bnpo: Beta normalization policy optimization. *arXiv preprint arXiv:2506.02864*, 2025.

Jiarui Yao, Ruida Wang, Hao Bai, and Tong Zhang. Future-kl regularized grpo: Process-level credit assignment from f-divergence regularization. *arXiv preprint arXiv:2601.10201*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Ruicheng Zhang, Mingyang Zhang, Jun Zhou, Xiaofan Liu, Zunnan Xu, Zhizhou Zhong, Puxin Yan, Haocheng Luo, and Xiu Li. Mind-v: Hierarchical world model for long-horizon robotic manipulation with rl-based physical alignment. *arXiv preprint arXiv:2512.06628*, 2025.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

A Implementation Details

Tuning protocol. We tuned the standard GRPO hyperparameters on Spider with OLMo2-1B and Qwen2.5-3B, then held them fixed across all datasets and model families; Qwen2.5-7B reused the Qwen2.5-3B configuration. The hyperparameters GRPO-JEPA adds (λ, pooling, and k) were swept per dataset with Qwen2.5-3B, and the main results use each dataset's best setting. SAPO, BNPO, and DAPO modify the GRPO surrogate itself, so they are evaluated at the baseline's hyperparameters: that comparison is a consistency check, while the baseline and ablation comparisons supply the primary evidence.

Table 2 lists the full hyperparameter configuration.

Hyperparameter	Value
Optimizer / learning rate	adamw_torch_fused / 3e-6
LR schedule / warmup	cosine / 0.1
Group size G	64
Prompts per batch	8
Sampling temperature / top-p	1.0/0.9
Max completion length	2048
InfoNCE temperature τ	1
Advantage clamp	[-20, 20]
KL coefficient β_0	[0.01, 0.05, 0.1, 0.2, 0.4]
JEPA loss weight λ (Method A)	[0.01, 0.05, 0.1, 0.2, 0.5]
Predictor tokens k	[0, 1, 2, 3, 4]
Selected k per dataset	Spider 2, GSM8K 2, HellaSwag 1, Yelp 0
EMA decay (Method B)	0.95
Warmup steps / z-score clip (Method B)	20 / [-5, 5]
β floor (Method B)	0.1 β_0
Training steps	250
Random seeds	[1, 42, 58, 69, 108]

Table 2: Full hyperparameter configuration. Bracketed values are swept grids; the k and τ sweeps are reported in section D, and the λ / β_0 grids in table 8.

B Dataset Construction Details

We convert each benchmark into GRPO-JEPA training data: the text view, the code view, the GRPO reward, and the evaluation metric. table 3 gives an overview.

B.1 COMMON PROTOCOL

For every dataset, GRPO training uses $2{,}000$ cases sampled from the public benchmark, and all hyperparameter sweeps select on a held-out validation split of over $1{,}000$ cases per dataset that training never sees. Each case is a standard chat triple (system, user, assistant). The *text view* used for the JEPA prediction is the raw user input (cf. section 3.2), while the *code view* is a semantic target text whose construction is dataset-specific and described below. Rewards are graded (not binary) so that the advantages carry a fine-grained quality signal.

Dataset	Task	Code view	Reward range	Metric
Spider	Text-to-SQL	gold SQL	$\{-1\} \cup [0, 1] \cup [2, 3]$	execution accuracy
GSM8K	Math	reasoning + answer	$[-1, 2]$	exact match
HellaSwag	NLI (4-way)	letter + option text	$[0, 2]$	option accuracy
Yelp	Sentiment (5-way)	reasoning + label	$[-1, 2]$	label accuracy

Table 3: Overview of the four benchmarks. All datasets contribute $2{,}000$ GRPO training cases; reward ranges refer to the graded training rewards defined below.

B.2 SPIDER

Construction. Spider (Yu et al., 2018) is a cross-domain Text-to-SQL benchmark. The model input consists of the user query together with the database schema, serialized as SQLite DDL statements. The schema is part of the policy's input, but it is stripped from all three JEPA embeddings: E_{pred}, E_{ans}, and E_{comp_j} are computed from the raw user query, the gold answer, and the sampled completions alone (section 3.2). The code view is the gold SQL query used directly, with no added rationale, as SQL already encodes the query semantics. One constructed case: system *"Convert User Query to SQL, only output SQL."*; user *"For db_id:[college_1] Find the first names of all professors in the Accounting department who is teaching some course and the class room."*; assistant *"SELECT T2.emp_fname, T1.class_room FROM CLASS AS T1 JOIN employee AS T2 ON ... WHERE T4.dept_name = 'Accounting'"*.

Reward. We extract and validate the predicted SQL from the completion. Completions with missing, malformed, or non-executable SQL receive a reward of -1. Executable completions receive the Jaccard similarity between their result set and the gold query's result set (in $[0, 1]$). Exact matches (Jaccard $= 1$) enter $[2, 3]$, ranked by the EXPLAIN cost of their SQL and spaced evenly by that rank, with ties broken by query length. This rewards queries that are both correct and efficient.

Metric. Execution accuracy: fraction of completions whose result set exactly matches gold.

B.3 GSM8K

Construction. GSM8K (Cobbe et al., 2021) is a grade-school math word-problem benchmark. The gold solution is restructured into the `<thought>` format: the reasoning steps are wrapped in `<thought>` tags and followed by the final numerical answer, and this target serves as the code view. A constructed case: system *"Answer the math question, show steps."*; user *"Every day, Wendi feeds each of her chickens three cups of mixed chicken feed..."*; assistant *"<thought>If each chicken eats 3 cups of feed per day, then for 20 chickens they would need* $3 \times 20 = 60$ *cups... the final meal would require* $60 - 15 - 25 = 20$ *cups of chicken feed.</thought> 20"*.

Reward. The reward logic is described below: completions must follow the format `<thought>...</thought>` followed by a numeric final answer; malformed completions receive -1. Valid completions receive the word-level Jaccard similarity between the generated and reference reasoning wrapped in the `<thought>` tags (in $[0, 1]$), plus 1 if the final numeric answer matches exactly, giving a range of $[-1, 2]$.

Metric. Exact match of the final numeric answer.

B.4 HELLASWAG

Construction. HellaSwag (Zellers et al., 2019) is a four-way multiple-choice task, and a bare option letter (A/B/C/D) carries almost no semantics for the JEPA supervision signal. We move the option list before the context in the user prompt and replace the assistant's answer letter with the letter concatenated with its full option text. For example, the answer *"B"* becomes *"B) [substeps] If you do use coconut milk, combine it with lemon juice from half a lemon…"*, which serves as the code view.

Reward. The reward follows the graded design of GSM8K, but scores the option text instead of a `<thought>` rationale. A matching gold letter earns 1, and the word-level Jaccard similarity between generated and reference option text adds up to 1 more, for a range of $[0, 2]$.

Metric. Option accuracy: fraction of completions whose letter matches gold.

B.5 YELP

Construction. Each Yelp review (Zhang et al., 2015) is expanded with a GPT-5 (Singh et al., 2026)-generated rationale (at most 100 tokens) using the prompt template below. The rationale, wrapped in `<thought>` tags, is prepended to the sentiment label to form the assistant target. For the JEPA computation, the text view is the original review, and the code view is the rationale wrapped in `<thought>` tags followed by the label. The generation prompt is:

```
You are a sentiment analysis expert. Analyze the following
Yelp review and determine its sentiment.

System Prompt: {system_content}
Review: {user_content}

Think step by step about the sentiment of this review. Consider
the key words, tone, and overall impression. Then conclude with
your final classification. Your entire response must be within
100 tokens.

Important: Your analysis must naturally lead to the conclusion:
{answer}. Write the reasoning as if you are independently
arriving at this conclusion, do NOT mention that you were
given the answer. Output ONLY the reasoning text, without any
labels, prefixes, or XML tags.
```

For example, we build the target for the review *"I don't quite get this place or why Asians love it, but it is very good :)"* as *"<thought>The user expresses confusion about the venue's appeal, but explicitly concludes that the experience is 'very good'… </thought> Good"*.

Reward. Completions must follow the format `<thought>...</thought>` followed by one of the five labels (*Very Bad, Bad, Mediocre, Good, Very Good*); malformed completions receive -1. Valid completions receive the word-level Jaccard similarity between the generated and reference rationales (in $[0, 1]$), plus 1 if the label matches exactly, giving a range of $[-1, 2]$.

Metric. Label accuracy on the five-way sentiment classification.

Anchor ablation. To check whether the Yelp gain needs the rationale, we reran with a label-only variant. The code view and the assistant target become the sentiment label alone, with no GPT-5 rationale, and the reward drops its rationale-similarity term to a binary label match (1 if correct, 0 otherwise). Table 4 reports label accuracy. The default configuration beats the label-only one on all three arms, and the two methods behave differently: Method A keeps a smaller but positive gain ($+4.6$), whereas Method B falls below its own baseline (62.16 vs. 65.23). This shows that the Yelp gains in table 1 require the full rationale-bearing setup rather than a label-only target with a binary reward.

Yelp anchor	Baseline (GRPO)	Method A	Method B
rationale + label (default)	67.54	76.32	74.80
label only	65.23	69.84	62.16

Table 4: Label accuracy (%) on Yelp under the default rationale-bearing anchor and a label-only anchor (Qwen2.5-3B). The default rows are five-seed means; the label-only rows are a single run without a reported standard deviation.

C Compute Cost

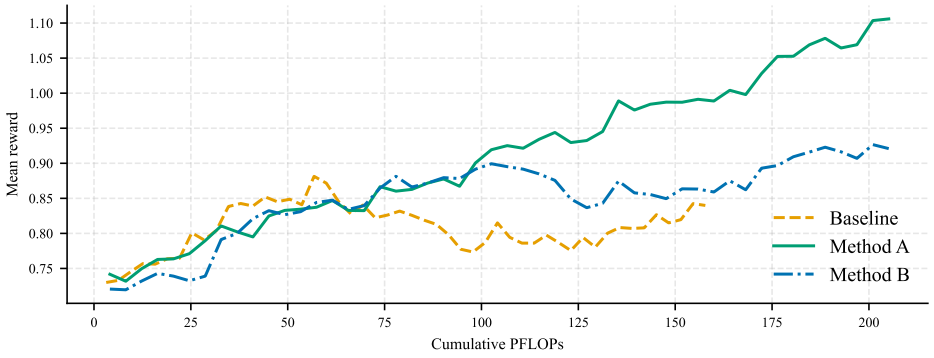

Figure 7: Mean task reward against estimated cumulative compute on Qwen2.5-3B / Spider. Reward curves use an EMA span of 25 logging points.

Each training step adds one forward pass through the predictor, about $0.3\times$ the cost of a standard GRPO step. Figure 7 plots mean task reward against estimated cumulative compute for the baseline, Method A, and Method B. The first 25 steps are a learning-rate warmup, so the curves stay close until the rate peaks at step 25. Method A then separates upward and ends above both other curves, while Method B ends slightly above the baseline. The extra forward pass therefore costs no final accuracy. The runs are single runs, and the reward curves use a wider EMA span than elsewhere; we read them for trend rather than as a compute-optimality claim.

D Hyperparameter Sensitivity and Ablations

We sweep the weights λ and β_0 (section D.4), the design ablations (section D.1), the predictor-token count k and temperature τ (section D.2), and target pooling (section D.3). All sweeps use Qwen2.5-3B, Method A (or Method B where β_0 is concerned), on all four benchmarks to check cross-dataset transfer; each configuration is trained once. The per-dataset λ and β_0 grids are reported in table 8, and the design ablations in table 5.

D.1 Design Choices

We validate the two load-bearing design choices, the advantage weighting and the three-embedding architecture, on Spider (Qwen2.5-3B); table 5 isolates each. Uniform weighting assigns every completion the same negative weight, injecting noise into the push direction, and linear weighting repels mid-quality completions; exponential weighting avoids both and outperforms them by 4.9–8.2 points. Removing the predictor tokens costs 3.1 points (67.2; table 5, table 6). With $k = 0$ the predictor reads E_{pred} from the last hidden state of the raw query and holds no predictor-token parameters of its own, yet it still performs the query→answer prediction and recovers about two thirds of Method A's gain. Gradients on E_{comp} are worse still (58.5%, near-collapse). Once the completion embeddings can move, the contrastive loss admits a trivial solution: bring E_{comp} toward E_{pred} and the loss falls without any gain in prediction quality. The two embeddings then collapse onto each other. The stop-gradient on the targets removes this solution. For the GRPO baseline in

Configuration	Accuracy (%) ↑	S_{ans} ↑
Baseline (GRPO)	61.6	0.12
Method A	70.3	0.64
Weighting strategy		
Uniform ($w_j = 1$)	62.1	0.31
Linear ($w_j \propto -\hat{A}_j$)	65.4	0.46
Architecture		
$k = 0$	67.2	0.49
Gradients on E_{comp}	58.5	0.14

Table 5: Design ablations on Spider (Qwen2.5-3B); each configuration is a single training run. Execution accuracy and S_{ans} under alternative weighting and architecture choices.

table 5, the predictor tokens are untrained, so its S_{ans} is a lower bound rather than a measurement of alignment.

D.2 Predictor Token Count and InfoNCE Temperature

Table 6 shows that small predictor counts are best: $k = 2$ achieves the top score on Spider and GSM8K, $k = 1$ on HellaSwag, and the $k = 0$ variant remains competitive everywhere, even best on Yelp. That $k = 0$ stays competitive shows the JEPA signal draws its predictive capacity from the shared Transformer weights rather than from the predictor-token parameters. Larger counts degrade the signal, most sharply on Yelp (76.32 at $k = 0$ to 68.10 at $k = 4$). The temperature τ barely matters: on every dataset the three values sit within 0.9 points, so the loss weight λ absorbs τ and τ needs no separate tuning. We fix $\tau = 1$: under Method A, λ sets the loss scale (table 8); under Method B, z-score normalization removes any scale dependence. An explicit temperature would be redundant.

Setting	Spider	GSM8K	HellaSwag	Yelp
$k = 0$ (no predictor tokens)	67.20	76.18	75.71	**76.32**
$k = 1$	69.96	76.41	**76.82**	74.21
$k = 2$	**70.35**	**79.16**	75.11	73.12
$k = 3$	68.25	68.21	75.04	71.25
$k = 4$	65.31	67.52	72.23	68.10
$\tau = 0.5$	70.13	79.32	75.94	76.25
$\tau = 1$ (default)	70.35	79.16	76.82	76.32
$\tau = 2$	70.05	79.02	76.01	76.53

Table 6: Sensitivity to the predictor-token count k and the InfoNCE temperature τ (Method A) across all four benchmarks (Qwen2.5-3B; execution accuracy / exact match / option accuracy / label accuracy, %). The k grid uses $\tau = 1$; the τ grid uses the best k per dataset.

D.3 Pooling Strategy

Pooling	Spider	GSM8K	HellaSwag	Yelp
last-token (idx_{last})	68.21	79.16	73.94	76.32
mean pooling ($\text{pool}_{\text{mean}}$)	70.35	75.58	76.82	73.10

Table 7: Pooling strategy for E_{ans} and E_{comp} across all four benchmarks (Qwen2.5-3B, Method A).

Table 7 compares the two pooling choices for E_{ans} and E_{comp}. The two choices differ per dataset: mean pooling is better on Spider (+2.1) and HellaSwag (+2.9), while last-token is better on GSM8K (+3.6) and Yelp (+3.2). We treat pooling as a mild per-dataset design choice rather than a sensitive hyperparameter, and the main experiments use the better variant for each dataset.

D.4 Loss Weight λ and KL Coefficient β_0

Setting	Spider	GSM8K	HellaSwag	Yelp
$\lambda = 0.01$	63.2	78.03	73.52	71.89
$\lambda = 0.05$	66.7	**79.16**	75.81	72.11
$\lambda = 0.1$	**70.3**	78.51	**76.82**	74.28
$\lambda = 0.2$	66.9	77.23	74.91	**76.32**
$\lambda = 0.5$	65.6	77.16	74.27	75.91
$\beta_0 = 0.01$	59.7	75.20	73.59	67.23
$\beta_0 = 0.05$	65.1	**78.95**	**75.60**	70.98
$\beta_0 = 0.1$	**68.9**	76.50	73.70	72.58
$\beta_0 = 0.2$	67.8	75.89	72.29	**74.80**
$\beta_0 = 0.4$	65.9	75.20	72.08	73.89

Table 8: Sweep of λ (Method A) and β_0 (Method B) on all four benchmarks (Qwen2.5-3B).

Table 8 sweeps λ and β_0 on all four benchmarks. The best setting is dataset-specific: λ peaks at $0.1/0.05/0.1/0.2$ and β_0 at $0.1/0.05/0.05/0.2$ on Spider/GSM8K/HellaSwag/Yelp, so neither coefficient is tuning-free. Performance nevertheless varies smoothly across the swept ranges, with no sharp optimum. Method A tolerates λ across a $50\times$ range with a mild optimum near $\lambda = 0.1$, because λ rescales the JEPA signal while the advantage weights w_j set the direction inside it. Method B has a floor at the low end: within $\beta_0 \in [0.05, 0.2]$ accuracy varies by at most 3.8 points, consistent with the controller's invariance to loss scale (section 3). At $\beta_0 = 0.01$ the KL term is under-constrained: Spider drops to 59.7%, below the GRPO baseline, and Yelp to 67.23, where the policy drifts. We characterize both coefficients as having a wide stable region rather than a single transferable optimum.

E Combining Method A and Method B

Motivation. Methods A and B inject the same JEPA signal two ways: Method A adds it as a loss term on the representation, while Method B uses it to modulate the KL coefficient without adding a term of its own. We test whether the two methods complement or conflict.

Objective. The total objective activates both methods:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}} + \lambda \cdot \mathcal{L}_{\text{JEPA}} + \beta_{\text{dyn}} \cdot D_{\text{KL}}, \tag{8}$$

where $\mathcal{L}_{\text{JEPA}}$ enters as in Method A and β_{dyn} is computed by the Method B controller from the same $\mathcal{L}_{\text{JEPA}}$ values; the two methods therefore share one JEPA forward pass, so the combined variant adds no extra compute beyond Method A.

Protocol. The experiment runs on all four benchmarks with Qwen2.5-3B. We train only the total objective of eq. (8), with the full configuration of table 2 in a single run at seed 42, and fix λ and β_0 at the best per-dataset settings of table 8. It is compared against the single-method arms of table 1, which are five-seed means under the same configuration and data pipeline. The combined variant is a single run rather than a five-seed mean.

Variant	Spider	GSM8K	HellaSwag	Yelp
GRPO baseline	61.65	74.73	73.03	67.54
Method A only	70.35	**79.16**	76.82	76.32
Method B only	68.90	78.95	75.60	74.80
Combined (A + B)	**72.17**	78.25	**78.14**	**77.66**

Table 9: Comparison of the two methods, separately and jointly, across four benchmarks (Qwen2.5-3B; execution accuracy / exact match / option accuracy / label accuracy, %).

Results. Before running the experiment, three outcomes were pre-registered. (i) If the combined variant matches or beats the better single-method arm, the methods are complementary: Method A shapes the representation space directly, Method B regulates exploration intensity, and the two stack. (ii) If it matches Method A only, the direct constraint already subsumes Method B's benefit. (iii) If it falls below the better single-method arm, the methods interfere, for instance when the dynamic KL rescales the gradient signal Method A relies on. Table 9 supports outcome (i) on three of the four benchmarks. On Spider, HellaSwag, and Yelp the combined variant beats both single-method arms (Spider: 72.17 vs. 70.35 for Method A), and it gives the best average accuracy (76.6, against 75.7 and 74.6). Because the combined variant is a single run, we compare its margins with the seed noise of the five-seed means: Spider is $+1.8$ against ± 1.2, HellaSwag $+1.3$ against ± 1.1, so complementarity is suggestive. GSM8K is the exception: the combined variant (78.25) falls 0.7–0.9 points below Method A (79.16) and Method B (78.95), within the seed noise, so no interaction is detectable; it still beats the baseline (74.73). Running both methods together gives the largest average gain.