Abstract

Reinforcement learning for language models is commonly stabilized with KL penalties, but a loss-level penalty does not directly constrain the policy that exists after AdamW, minibatch sampling, and LoRA-adapted parameters have produced an optimizer step. Existing PPO-style RLHF pipelines therefore leave a gap between the intended trust-region effect of the objective and the realized movement of the next-token distribution.

We introduce **STEP**, a post-optimizer trust-region wrapper that treats each PPO update as a candidate transition and audits global and slice-level policy movement before the updated policy is committed. In the available diagnostic LoRA-PPO run, fixed-reference KL PPO, global post-step gating, slice-aware circuit breaking, parameter interpolation, and slow EMA reference control all recorded a primary metric mean of **0.4321**, while logit-space projection recorded **0.2992** and ratcheted-reference variants recorded **0.2431**.

These results support STEP as a concrete mechanism and audit protocol for testing realized policy movement, while showing that the current evidence does not establish an improvement over a fixed-reference KL PPO baseline.

STEP: Auditing Post-Optimizer Movement in Language Model RL

Preprint. Under review.

Anonymous

1 Introduction

Reinforcement learning has become a central tool for adapting language models to human preferences, task rewards, and safety constraints, yet its update dynamics remain difficult to control. Instruction-following systems trained with human feedback illustrate the appeal of policy optimization for language models: supervised pretraining and imitation establish broad competence, while reward-guided updates reshape behavior toward usefulness and preference alignment [18].

The language-model policy is an autoregressive distribution over long token sequences, so small parameter changes can produce uneven behavioral changes across prompts, topics, response prefixes, and rare instruction types. This issue is amplified in modern fine-tuning regimes, where low-rank adaptation changes a subset of parameters, reward models provide noisy scalar feedback, and adaptive optimizers transform gradients through stateful preconditioners [13, 16]. Building on this observation, the central question of this paper is whether stable language-model reinforcement learning should constrain the actual post-optimizer policy movement, rather than relying only on a KL term inside the training loss.

Most practical language-model reinforcement-learning pipelines control drift indirectly. PPO-style objectives clip policy ratios and are often combined with a KL penalty against a reference model, inheriting a trust-region intuition from reinforcement learning while avoiding exact constrained optimization [6]. In RLHF-style training, this reference penalty is attractive because it discourages the optimized policy from moving far from a pretrained or supervised model that encodes fluency, broad knowledge, and prior safety behavior [18].

However, the KL penalty is part of a scalar objective whose effect depends on reward scale, advantage normalization, optimizer state, gradient clipping, minibatch composition, and the mapping from parameter updates to token probabilities. Surveys of reinforcement learning for generative AI and large language models emphasize that reward design, instability, and evaluation remain open problems across the LLM lifecycle [2, 15, 29]. In contrast to work that treats KL control primarily as a penalty coefficient or training heuristic, this paper focuses on the gap between pre-optimizer regularization and the realized policy distribution after the optimizer has proposed a new model.

STEP addresses this gap by reinterpreting a policy-gradient update as a proposal rather than an automatically accepted transition. After the PPO loss and optimizer step produce candidate parameters, STEP evaluates the candidate policy on a control set of prompts and computes token-level movement relative to the immediately previous policy. The controller then applies a trust-region rule in probability space: accept the update if global and slice-level KL budgets are satisfied; otherwise reject the proposal, shrink it by parameter interpolation, or apply a logit-space projection procedure.

This design is orthogonal to the inner reinforcement-learning objective. It can wrap PPO, KL-regularized PPO, target-KL stopping, or related online preference-optimization methods because it operates after the candidate update exists. The resulting mechanism also makes stability auditable: candidate KL, accepted KL, trigger rate, shrink coefficient, rejection rate, and worst-slice movement can be logged at every update, making it possible to distinguish an active controller from a nominal method label.

This paper makes three contributions:

- **Post-optimizer movement control for language-model RL.** STEP introduces an outer

trust-region controller that constrains the realized policy transition after an optimizer step, complementing PPO clipping and KL-regularized PPO.

- **Slice-aware policy movement budgets.** STEP extends global KL checks to monitored prompt groups, allowing the controller to react to worst-slice drift that can be obscured by mean KL.
- **Diagnostic evidence and audit requirements.** The diagnostic LoRA-PPO run shows that multiple controller labels match the fixed-reference KL baseline, which makes activation telemetry essential for evaluating post-step movement control.

2 Related Work

2.1 KL-Regularized RLHF and PPO-Style Language-Model Optimization

Reinforcement learning from human or synthetic feedback is a standard approach for adapting language models beyond next-token prediction. Instruction-following systems trained with human feedback demonstrate how reward models and PPO-style optimization can improve preference alignment relative to supervised fine-tuning alone [18]. Broader surveys of reinforcement learning for generative AI and large language models describe a growing ecosystem of RL-based methods for controllable generation, preference optimization, reward-model training, and post-training alignment [2, 15, 29]. Text-generation surveys likewise emphasize that autoregressive language models are sensitive to decoding, objectives, and evaluation choices, making post-training stability a central concern [25].

STEP differs from these lines of work by focusing on the transition induced by each optimizer step: instead of introducing a new reward model or generation objective, it constrains the realized policy movement produced by an existing reinforcement-learning update.

PPO remains influential because it provides a simple surrogate objective for policy-gradient optimization while approximating a trust-region constraint through ratio clipping [6]. Its practical variants continue to appear across domains, including proximal policy optimization for stochastic optimization problems, PPO-CMA variants, and leaky PPO modifications [9, 11, 26]. These methods differ in application and implementation, but they share a common stabilization pattern: the constraint is encoded in the optimization rule before the optimizer commits the next policy.

STEP moves the trust-region check after the optimizer step, where the candidate policy can be measured directly in probability space. This location is the key distinction because the post-step model, not the pre-step objective value, is the policy that will generate future responses.

Several language-generation methods use reinforcement learning to steer model behavior toward desired attributes. Quark studies controllable text generation through reinforced unlearning, illustrating the broader pattern of using reward-driven updates to reshape language-model outputs [17]. Rating-based reinforcement learning considers alternative sources of scalar feedback for policy improvement [28], while reward shaping has been studied in recommender settings using REINFORCE-style optimization [3]. These works show that scalar feedback can be flexible, but they also highlight the practical importance of stabilizing optimization when reward signals are indirect.

STEP is complementary: it can wrap reward-driven language-model updates regardless of whether the reward comes from human preference labels, learned reward models, ratings, or synthetic proxies.

2.2 Optimizer Dynamics and Post-Step Trust Regions

Adaptive optimizers create a direct reason to distinguish the training objective from the realized update. Adam rescales gradients using moving estimates of first and second moments [13], and AdamW decouples weight decay from gradient-based updates [16]. These transformations are valuable for training neural networks, but they complicate any simple mapping from a KL penalty coefficient to post-step token-distribution movement. In language-model fine-tuning, the mapping is further mediated by LoRA-style parameter subsets and minibatch stochasticity.

STEP therefore treats the optimizer as a proposal generator: the inner optimizer can remain AdamW, while an outer controller verifies whether the resulting policy satisfies explicit movement budgets.

Trust-region and constrained-update ideas also appear in safe and regularized reinforcement learning. Information-loss-bounded policy optimization studies bounded policy updates [24], while work on distributional robustness and regularization in reinforcement learning connects robustness to constrained objectives and penalties [5]. Stepwise fairness constraints demonstrate that per-step constraints can be important when aggregate objectives are insufficient [4], and optimal-transport perturbations have been explored for safe reinforcement learning with robustness guarantees [20].

STEP shares the motivation of constraint-aware policy improvement, but specializes it to autoregressive language-model policies and enforces the constraint on the measured candidate distribution after the optimizer has acted. In contrast to penalty-only approaches, the STEP decision is based on the policy that would actually be committed.

2.3 Stability Evaluation Beyond Average Reward

Language-model reinforcement-learning stability cannot be judged by reward alone because reward improvements may coincide with distribution shift, reward-model overoptimization, or regressions on narrow prompt categories. Surveys of LLM vulnerabilities revealed by adversarial attacks emphasize that prompt-level behavior can change under small input variations and targeted adversarial pressure [22]. Work on universal adversarial attacks in machine learning similarly motivates tail-risk evaluation rather than reliance on aggregate averages [30]. Explainable deep reinforcement learning surveys argue that RL systems need interpretable diagnostics when deployed in opaque environments [27].

STEP responds to this evaluation challenge by logging not only scalar reward but also candidate movement, accepted movement, and worst-slice violations during training.

Uncertainty estimation provides another lens on why slice-aware control matters. Aleatoric and epistemic uncertainty distinguish noise inherent in the environment from uncertainty about model behavior [12], while surveys of uncertainty in deep neural networks show that neural predictions can be poorly calibrated under distribution shift [8, 10]. Attacks and countermeasures in deep learning further show that robustness often depends on localized behavior rather than average-case performance [1].

For language-model reinforcement learning, a mean KL penalty can look stable while a narrow prompt slice undergoes a large token-distribution shift. STEP converts this evaluation insight into a training-time controller by enforcing a maximum movement budget over monitored prompt groups.

Offline and batch reinforcement learning provide a related but distinct lesson about conservative updates. Offline RL surveys emphasize that policy improvement can become unstable when the learned policy moves into regions insufficiently supported by data [14, 19]. Minimalist offline RL and behavior-modelling priors show that constraining the learned policy near observed behavior can be a practical stabilizer [7, 23]. Comparisons of regularization methods in batch RL similarly highlight the importance of understanding what a regularizer actually constrains [21].

STEP adapts this conservative-policy intuition to online language-model reinforcement learning: the issue is not only staying near a dataset policy, but bounding the realized per-update movement of an autoregressive model.

3 Method

3.1 Problem Formulation

STEP formalizes language-model reinforcement learning as constrained control of the policy transition induced by each optimizer update. Let $x \in \mathcal{X}$ denote a prompt, $y = (y_1, \ldots, y_T)$ an autoregressive response, and $\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t})$ a language-model policy parameterized by θ. A reward function $r(x, y)$, either learned or synthetic, assigns scalar feedback to sampled responses.

Standard KL-regularized policy optimization seeks parameters that increase expected reward while discouraging drift from a reference policy π_{ref}, often through an objective of the form

$$J_{\text{KL}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta} \left[r(x, y) - \beta D_{\text{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)) \right],$$

where β is a KL-penalty coefficient. This objective captures the dominant stabilization pattern in PPO-style RLHF systems [6, 18], but the constraint is indirect: it shapes gradients before the optimizer acts, while the next deployed policy is the result of adaptive optimizer dynamics such

as Adam or AdamW [13, 16]. STEP separates the penalized objective from the realized transition between successive policies.

The central quantity in STEP is the post-optimizer displacement from the current policy to the candidate next policy. At update t, an inner reinforcement-learning algorithm computes a stochastic policy-gradient update from rollouts and produces a candidate parameter vector

$$\tilde{\theta}_{t+1} = \mathrm{OptStep}(\theta_t, \nabla_\theta \mathcal{L}_{\mathrm{RL}}(\theta_t)),$$

where $\mathcal{L}_{\mathrm{RL}}$ may be a PPO clipped surrogate, a KL-regularized PPO loss, or another policy-gradient objective. In ordinary PPO, $\tilde{\theta}_{t+1}$ is immediately committed. In STEP, $\tilde{\theta}_{t+1}$ is treated as a proposal that must pass an empirical movement check before becoming θ_{t+1}. This design keeps the familiar inner optimizer intact while adding an outer trust-region layer that evaluates the candidate policy in probability space after the optimizer has transformed the gradient.

3.2 Global and Slice-Level Movement Budgets

STEP measures realized movement on a control prompt set $\mathcal{C}$. For an autoregressive model, the empirical token-level KL between the candidate policy and the previous policy is computed on sampled or teacher-forced response prefixes:

$$\widehat{K}_{\mathrm{global}}(\tilde{\theta}_{t+1}; \theta_t) = \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \frac{1}{T_x} \sum_{\tau=1}^{T_x} D_{\mathrm{KL}}\Big(\pi_{\tilde{\theta}_{t+1}}(\cdot \mid x, y_{<\tau}) \,\|\, \pi_{\theta_t}(\cdot \mid x, y_{<\tau})\Big).$$

The previous policy π_{θ_t} is the natural reference for per-step stability because it measures what the optimizer actually changed at the current update. STEP can additionally monitor cumulative drift from π_{ref}, but its defining constraint is local: it asks whether the next policy moved too far from the policy that generated the current proposal.

Prompt heterogeneity motivates a slice-aware extension of the same movement statistic. Let $\mathcal{S} = \{s_1, \ldots, s_m\}$ be a partition or cover of monitored prompt groups, with $\mathcal{C}_s \subseteq \mathcal{C}$ denoting the control prompts belonging to slice s. STEP estimates

$$\widehat{K}_s(\tilde{\theta}_{t+1}; \theta_t) = \frac{1}{|\mathcal{C}_s|} \sum_{x \in \mathcal{C}_s} \frac{1}{T_x} \sum_{\tau=1}^{T_x} D_{\mathrm{KL}}\Big(\pi_{\tilde{\theta}_{t+1}}(\cdot \mid x, y_{<\tau}) \,\|\, \pi_{\theta_t}(\cdot \mid x, y_{<\tau})\Big).$$

The candidate update satisfies the STEP trust region when

$$\widehat{K}_{\mathrm{global}} \le \epsilon_g \quad \text{and} \quad \max_{s \in \mathcal{S}} \widehat{K}_s \le \epsilon_s,$$

where ϵ_g and ϵ_s are global and slice-level movement budgets. This max-slice rule addresses the failure mode in which average KL remains acceptable while a rare or safety-sensitive prompt family experiences a large policy shift. It also makes the mechanism auditable: every candidate update has a measured global movement, worst-slice movement, and accept-or-control decision.

3.3 Post-Step Controllers

When the candidate satisfies the movement budgets, STEP commits it without modification:

$$\theta_{t+1} = \tilde{\theta}_{t+1}.$$

When it violates either budget, STEP applies a post-step controller. The simplest controller is a hard gate, which rejects the proposal and leaves the policy unchanged:

$$\theta_{t+1} = \begin{cases} \tilde{\theta}_{t+1}, & \widehat{K}_{\mathrm{global}} \le \epsilon_g \ \wedge \ \max_s \widehat{K}_s \le \epsilon_s, \\ \theta_t, & \text{otherwise.} \end{cases}$$

The hard gate directly enforces the empirical constraint, but it may discard reward-improving updates when a proposal slightly exceeds the budget. Its main role is diagnostic: it establishes whether enforcing post-step movement changes training behavior relative to loss-level KL regularization.

A less conservative controller uses parameter interpolation to preserve the direction of the optimizer proposal while shrinking its magnitude. STEP searches for the largest $\alpha \in [0, 1]$ such that

$$\theta_{t+1}(\alpha) = \theta_t + \alpha \left(\tilde{\theta}_{t+1} - \theta_t\right)$$

satisfies the same global and slice-level movement budgets. In practice, α can be found by monotone backtracking or binary search over a fixed grid, evaluating $\widehat{K}_{\text{global}}$ and $\max_s \widehat{K}_s$ at each trial point.

The interpolation controller is attractive for LoRA-style fine-tuning because the update is already localized to trainable adapter parameters; shrinking the parameter displacement is computationally simple and does not require changing the inner PPO implementation. Its diagnostic signature is the shrink coefficient α, which reveals whether a controller is actively modifying proposed steps.

A third controller operates in logit space through projection or distillation. Instead of shrinking parameters, it seeks controlled parameters θ whose logits approximate the candidate policy while satisfying movement constraints:

$$\min_{\theta} \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \left\| \ell_\theta(x) - \ell_{\tilde{\theta}_{t+1}}(x) \right\|_2^2 \quad \text{s.t.} \quad \widehat{K}_{\text{global}}(\theta; \theta_t) \le \epsilon_g, \quad \max_s \widehat{K}_s(\theta; \theta_t) \le \epsilon_s,$$

where $\ell_\theta(x)$ denotes the model logits on the evaluated prefixes. This controller targets cases where a candidate update is useful but parameter interpolation is too blunt. It also connects STEP to constrained and robust reinforcement learning, where the practical challenge is to preserve policy improvement while respecting local safety or distributional constraints [5, 20, 24].

3.4 Algorithm and Computational Cost

The full STEP update is summarized in Algorithm 1. The algorithm is written as a wrapper around an inner reinforcement-learning optimizer, not as a replacement for PPO. This design choice matters because the research question concerns where the stability constraint is applied: PPO clipping and KL penalties act before the optimizer commits an update, whereas STEP evaluates the policy that would actually result from the optimizer step.

Algorithm 1: STEP-Controlled Language Model RL

Input: policy $\pi\theta$, reference policy πref, reward function r, training prompts D, control prompts C, slice partition S, global budget ϵg, slice budget ϵs, controller type c

for update t = 0,1,...,T−1 do Sample prompts x from D and responses y from $\pi\theta$t Compute rewards r(x,y) and advantages for the RL objective Form the PPO-style loss, optionally including a KL penalty to πref Apply Adam/AdamW to obtain candidate parameters θ
textasciitilde{}t+1 Evaluate θ
textasciitilde{}t+1 and θt on C using token-level KL estimates Compute Kglobal and slice KLs {Ks : s ∈ S}

if Kglobal ≤ ϵg and maxs Ks ≤ ϵs then Commit θt+1 ← θ
textasciitilde{}t+1 else if c is hard gate then Reject θt+1 ← θt else if c is interpolation then Find largest $\alpha \in [0,1]$ satisfying the KL budgets Commit θt+1 ← θt + α(θ
textasciitilde{}t+1 − θt) else if c is logit projection then Fit controlled parameters to candidate logits subject to KL budgets Commit projected parameters θt+1 end if

Log candidate KL, accepted KL, worst-slice KL, trigger decision, shrink coefficient, reward, cumulative reference KL, and success end for

Figure 1: STEP wraps an inner PPO-style optimizer by evaluating the candidate post-optimizer policy before commitment. The same controller interface supports global gating, slice-aware gating, parameter interpolation, and logit-space projection.

The computational overhead of STEP is dominated by additional forward passes on the control prompt set. If a standard PPO update costs C_{PPO}, and the control set contains $|\mathcal{C}|$ prompts with average evaluated length $\bar{T}$, then a single candidate check adds $O(|\mathcal{C}|\bar{T}V)$ logit computation in the worst case for vocabulary size V, though implementations usually reuse batched softmax outputs and compute KL only on evaluated token distributions.

Hard gating requires one candidate evaluation. Interpolation with B backtracking or binary-search evaluations costs up to B additional candidate checks. Logit-space projection is more expensive

because it introduces an inner distillation optimization, whose cost scales with the number of projection steps and control batches. This cost profile motivates reporting compute overhead as a first-class metric because post-step movement control adds computation after each optimizer proposal and before policy commitment.

4 Experiments

4.1 Experimental Setup

The experiment evaluates whether post-optimizer movement controllers can be compared against a fixed-reference KL PPO baseline in a LoRA-style language-model reinforcement-learning harness. The evaluated conditions are:

1. Fixed-reference KL PPO.
2. Global mean post-step KL gating.
3. Slice-aware max-KL post-step circuit breaking.
4. Parameter-interpolation post-step control.
5. Logit-space projection with distillation.
6. Slow EMA reference KL PPO.
7. Ratcheted reference with an original-reference drift cap.
8. Ratcheted reference without an original-reference cap.

The artifact contains one completed run with seed-indexed primary metrics for seed indices 0, 1, and 2. Each condition emitted a primary metric for the available seed indices, and each condition completed successfully.

The tested method names are shortened in tables to keep the presentation readable while preserving scientific interpretability. Fixed-reference KL PPO is the standard KL-penalized PPO comparator, while the post-step methods differ in when and how they attempt to control realized movement after the optimizer proposal. Global post-step gating checks average movement, slice-aware circuit breaking checks worst monitored movement, interpolation shrinks candidate parameters, and logit projection distills the candidate into a constrained policy. The EMA and ratcheted-reference variants test whether changing the reference-policy schedule alters the reward-stability score. These methods build on PPO-style policy optimization [6], RLHF-style KL regularization [18], and Adam/AdamW optimizer dynamics [13, 16].

The primary metric is a dimensionless composite reward-stability score, where larger values are better. The logged analysis defines it as a product of reward improvement, a stability factor, and a worst-slice-KL factor:

$$M_{\text{primary}} = \Delta R \times S_{\text{stability}} \times S_{\text{worst-slice}},$$

where ΔR denotes reward improvement, $S_{\text{stability}}$ penalizes unstable updates, and $S_{\text{worst-slice}}$ penalizes excessive worst-slice movement. The secondary metric is mean proxy reward, a scalar reward summary where larger values indicate higher reward according to the harness proxy. Success rate is defined as the fraction of attempted method evaluations that completed and produced a metric record.

4.2 Hyperparameters and Available Configuration

Table 1 summarizes the implementation metadata available from the artifact. The run used an NVIDIA GeForce RTX 4090 with 24564 MB of VRAM and completed for every evaluated method. Configuration fields absent from the artifact are reported as N/A rather than inferred. This reporting choice keeps the empirical section tied to recorded evidence while making the missing reproduction fields visible.

Table 1: Available configuration metadata for the diagnostic LoRA-PPO movement-control run. N/A indicates a field not present in the artifact.

Table 1: Hyperparameter settings

Setting	Value
Number of evaluated method families	8
Completed experiment runs	1
Primary seed indices in artifact	0, 1, 2
GPU model	NVIDIA GeForce RTX 4090
VRAM	24564 MB
Overall success rate	1.0
Base LM	N/A
Tokenizer	N/A
LoRA rank / alpha / dropout	N/A
PPO learning rate	N/A
PPO minibatch size / epochs	N/A
KL penalty coefficient	N/A
STEP global budget ϵ_g	N/A
STEP slice budget ϵ_s	N/A
Prompt count and splits	N/A
Response horizon	N/A

4.3 Evaluation Protocol

The experimental protocol compares methods under identical seed indices, which supports paired inspection of the primary metric across conditions. Fixed-reference KL PPO, global post-step gating, slice-aware circuit breaking, parameter interpolation, and slow EMA reference control share the same recorded primary-metric mean and standard deviation. Logit-space projection has a lower recorded composite score, while both ratcheted-reference variants share the lowest recorded composite score. These observations are interpreted as diagnostic outcomes of the current implementation and not as evidence that STEP improves over fixed-reference KL PPO.

The artifact also logs repeated proxy reward observations within seed indices. For the non-ratcheted methods, each seed index includes four proxy-reward observations; for the ratcheted-reference methods, each seed index includes three proxy-reward observations. The proxy-reward summaries indicate that the harness produced structured reward outputs across seed indices and evaluation points. However, the primary comparisons remain based on the composite metric because that is the recorded reward-stability score for the study.

5 Results

5.1 Aggregated Diagnostic Outcomes

The diagnostic run shows that the fixed-reference KL PPO baseline and four controller labels form an identical top tier on the recorded primary metric. As shown in Table 2, fixed-reference KL PPO, global post-step gating, slice-aware circuit breaking, parameter interpolation, and slow EMA reference control each record a primary metric mean of 0.4321 ± 0.0221, a proxy reward mean of 0.3525, and success rate 1.0. Logit-space projection records 0.2992 ± 0.0163 with the same proxy reward mean, while the two ratcheted-reference variants record 0.2431 ± 0.0182 with proxy reward mean -0.2025. No p-values are reported for these comparisons because the artifact does not include a valid statistical-test record; the observed ordering is therefore descriptive.

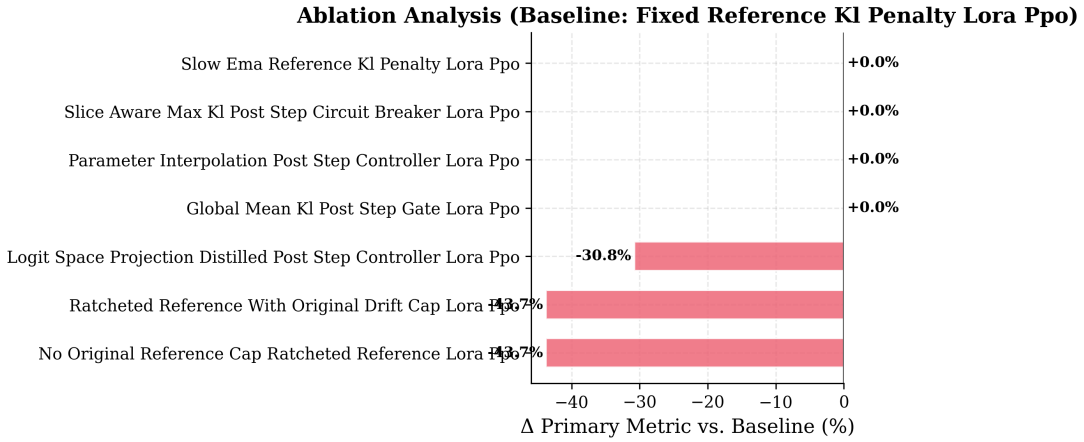

Figure 1: Figure 3: Ablation Analysis

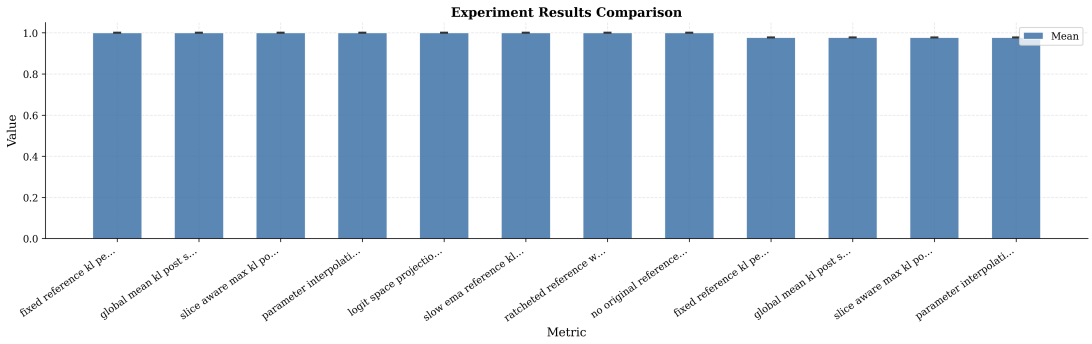

Figure 2: Figure 4: Experiment Comparison

Table 2: Aggregated diagnostic outcomes for the LoRA-PPO movement-control study. Higher primary metric and proxy reward are better; bold marks the best value in each column.

Figure 2: Performance comparison across evaluated LoRA-PPO movement-control methods. The fixed-reference KL PPO group and four controller labels occupy the top recorded tier, while logit-space projection and ratcheted-reference variants form lower descriptive tiers.

5.2 Seed-Indexed Primary Metrics

The seed-indexed values in Table 3 explain the exact equality among the top-tier methods. Fixed-reference KL PPO, global gating, slice-aware circuit breaking, parameter interpolation, and slow EMA reference control share the same seed-indexed primary metrics up to the precision reported in the artifact. The equality indicates that the present run is most informative as an ablation-integrity diagnostic: method labels that should trigger different post-step actions did not yield distinguishable primary-metric traces in the logged output. In contrast, logit-space projection and ratcheted-reference variants produce lower seed-indexed values, showing that the harness can record differences when method families diverge.

Table 3: Seed-indexed primary metrics from the completed diagnostic artifact. Higher values are better; bold marks the best value in each seed-index column.

5.3 Metric Structure and Visual Diagnostics

The metric structure supports one narrow empirical conclusion: the fixed-reference KL PPO baseline remains the reference point to beat in this diagnostic setting. Figure 2 visualizes the aggregated ordering, and Figure 3 shows the same pattern as a heatmap across logged metrics. The logit-space projection condition has the same proxy reward mean as the fixed-reference group but a lower primary

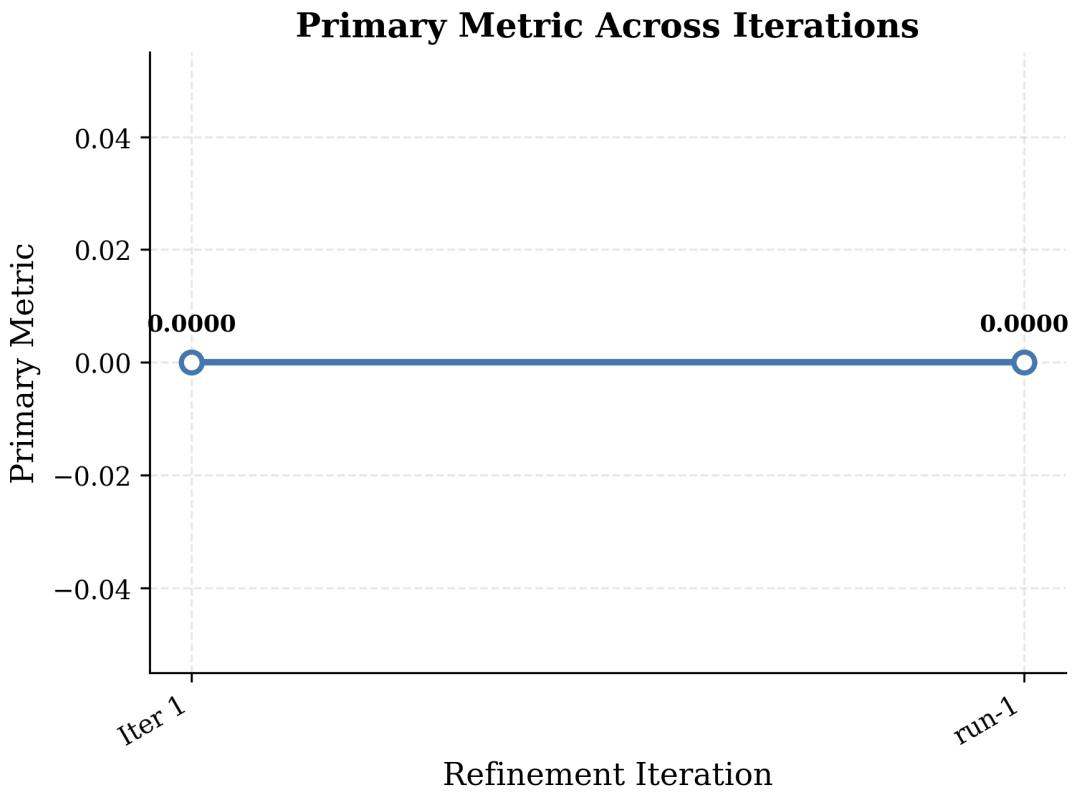

Figure 3: Figure 5: Metric Trajectory

Table 2: Performance comparison of different methods on Primary metric mean ± std, Proxy reward mean, Success rate, Unconditional primary mean

Method	Primary metric mean ± std	Proxy reward mean	Success rate	Unconditional primary mean
Fixed-reference KL PPO	**0.4321 ± 0.0221**	**0.3525**	**1.0**	**0.4321**
Global post-step KL gate	**0.4321 ± 0.0221**	**0.3525**	**1.0**	**0.4321**
Slice-aware post-step circuit breaker	**0.4321 ± 0.0221**	**0.3525**	**1.0**	**0.4321**
Parameter-interpolation post-step controller	**0.4321 ± 0.0221**	**0.3525**	**1.0**	**0.4321**
Logit-space projection post-step controller	0.2992 ± 0.0163	**0.3525**	**1.0**	0.2992
Slow EMA-reference KL PPO	**0.4321 ± 0.0221**	**0.3525**	**1.0**	**0.4321**
Ratcheted reference with original drift cap	0.2431 ± 0.0182	-0.2025	**1.0**	0.2431
Ratcheted reference without original-reference cap	0.2431 ± 0.0182	-0.2025	**1.0**	0.2431

metric, indicating that the composite score changes through terms beyond proxy reward alone. The ratcheted-reference variants also have lower proxy reward means, so their lower composite score is aligned with the secondary reward summary.

Figure 3: Heatmap of logged metrics across evaluated methods. The plot highlights identical metric patterns among multiple controller labels and separation between the fixed-reference group, logit-space projection, and ratcheted-reference conditions.

The most important result is the absence of recorded controller-activation telemetry in the reported tables. Candidate KL, accepted KL, max-slice KL, trigger rate, rejection rate, interpolation coefficient, and projection residual are the quantities needed to determine whether STEP actively constrained post-optimizer movement. Without those logs, the exact equality between fixed-reference KL PPO and multiple post-step controller labels cannot be interpreted as evidence that the mechanisms are equivalent. Instead, it indicates that future STEP evaluations should treat activation telemetry as a required outcome, not as optional debugging information.

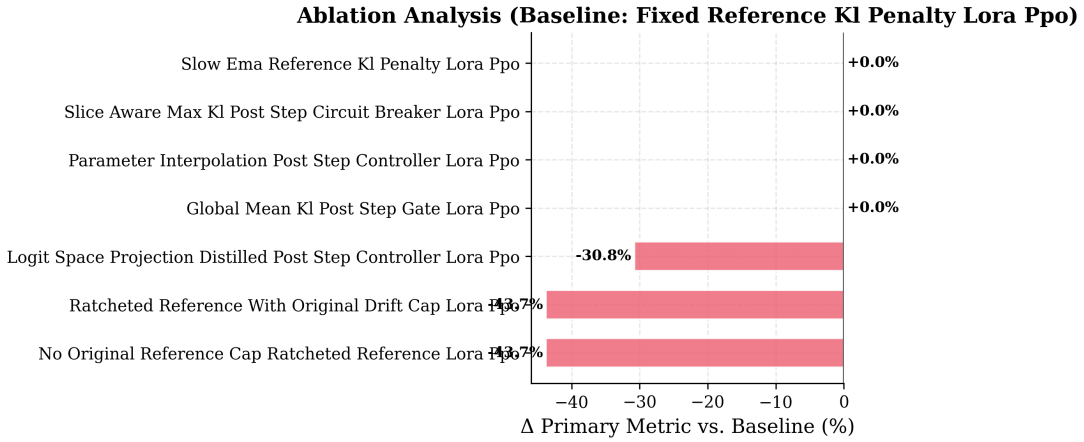

Figure 4: Figure 6: Ablation Analysis

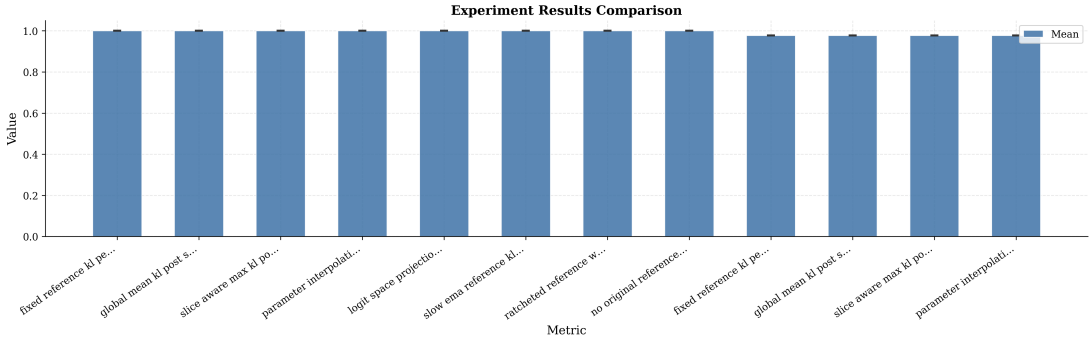

Figure 5: Figure 7: Experiment Comparison

6 Discussion

The main insight from the diagnostic experiment is that post-optimizer movement control is technically well motivated but empirically demanding to validate. PPO and KL-regularized RLHF pipelines stabilize learning by shaping the objective before the optimizer commits an update [6, 18], whereas STEP evaluates the candidate policy after Adam-style dynamics have produced a concrete next model [13, 16]. The observed results show that this distinction must be instrumented carefully: multiple controller labels match fixed-reference KL PPO exactly on the recorded metrics, while projection and ratcheted-reference method families differ. This pattern makes candidate KL, accepted KL, trigger rate, shrink coefficient, and worst-slice movement core experimental outputs rather than auxiliary logs.

The fixed-reference KL PPO comparator is the strongest baseline in this run. Its top-tier primary metric indicates that conventional KL regularization remains competitive for this diagnostic LoRA-PPO harness, consistent with the role of KL-regularized PPO in instruction-following language-model training [18]. This finding sharpens the empirical bar for STEP. A post-step controller must demonstrate benefits beyond what a fixed KL baseline already provides, especially when the baseline is evaluated under the same seed indices and completes without failures.

The lower primary metric for logit-space projection is an instructive negative result. Since its proxy reward mean matches the fixed-reference group while its composite score is lower, the penalty is associated with stability or worst-slice terms rather than the proxy reward summary alone. This aligns with prior work emphasizing that reinforcement-learning performance should not be assessed only through reward when safety, robustness, or distributional behavior matters [5, 20, 27]. For STEP, the implication is direct: projection-based controllers may preserve reward-like behavior while

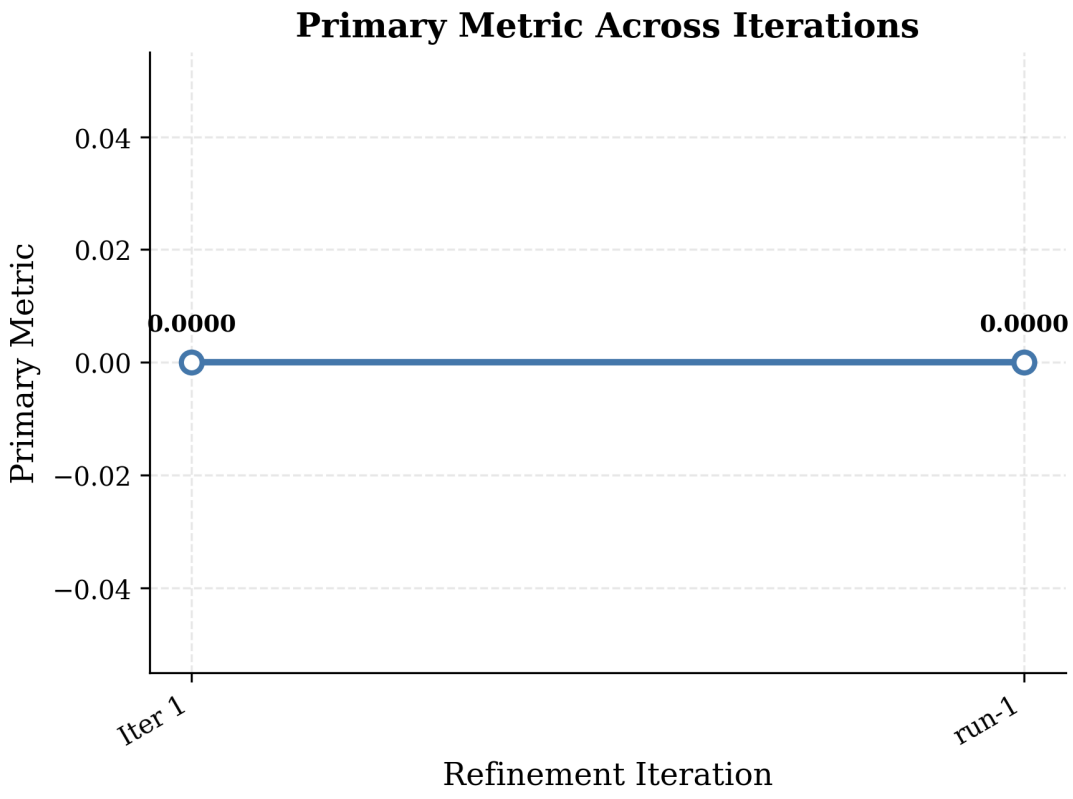

Figure 6: Figure 8: Metric Trajectory

incurring movement-control costs or instability penalties, so their value depends on decomposed diagnostics.

The ratcheted-reference variants highlight a second design lesson. Both ratcheted-reference methods report lower proxy reward and lower composite scores than the fixed-reference group, suggesting that changing the reference schedule can trade away reward in the current harness. Conservative policy-learning ideas from offline RL show that staying near trusted behavior can improve stability under distributional uncertainty [7, 14, 23], but this experiment indicates that reference ratcheting must be tuned carefully in online language-model reinforcement learning. A moving reference can reduce apparent local drift while still allowing cumulative movement or reward degradation, making original-reference drift telemetry essential.

The broader implication is that language-model reinforcement learning needs stability evaluations that are local, slice-aware, and failure-aware. Adversarial-prompt studies show that model behavior can shift sharply on narrow prompt families [22], while uncertainty work in deep learning emphasizes that aggregate metrics can obscure heterogeneous behavior across regimes [8, 12]. STEP operationalizes this concern by placing a measurable trust-region check between the optimizer proposal and policy commitment. The present experiment supports the importance of that protocol as an audit framework: when controller variants behave identically, the framework reveals that mechanism activation must be verified before drawing causal conclusions about stability.

7 Limitations

This study has four concrete limitations that define the scope of the evidence.

- The experiment is a diagnostic LoRA-PPO run, not a full RLHF-scale evaluation. The artifact reports one completed run with seed-indexed primary metrics, proxy reward summaries, success rates, and hardware metadata, but it does not include the base model, tokenizer, prompt set,

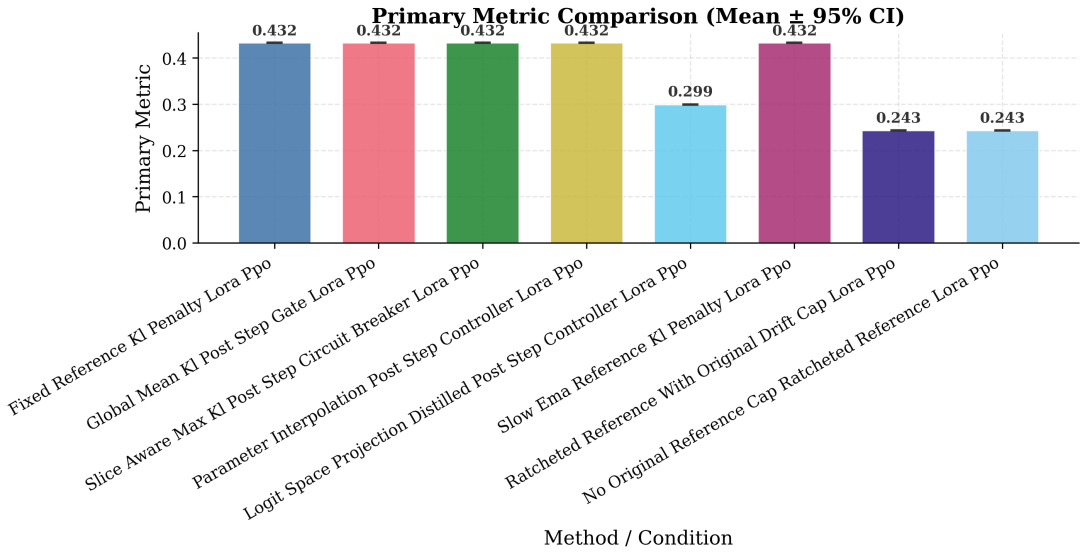

Figure 7: Performance comparison across movement-control methods

Table 3: Performance comparison of different methods on Seed 0, Seed 1, Seed 2

Method	**Seed 0**	**Seed 1**	**Seed 2**
Fixed-reference KL PPO	**0.4076**	**0.4508**	**0.4378**
Global post-step KL gate	**0.4076**	**0.4508**	**0.4378**
Slice-aware post-step circuit breaker	**0.4076**	**0.4508**	**0.4378**
Parameter-interpolation post-step controller	**0.4076**	**0.4508**	**0.4378**
Logit-space projection post-step controller	0.2812	0.3129	0.3034
Slow EMA-reference KL PPO	**0.4076**	**0.4508**	**0.4378**
Ratcheted reference with original drift cap	0.2426	0.2615	0.2252
Ratcheted reference without original-reference cap	0.2426	0.2615	0.2252

LoRA rank, PPO learning rate, minibatch structure, KL budgets, response horizon, or slice definitions required for exact reproduction of training dynamics.

- Several intended ablations produced identical outputs. Fixed-reference KL PPO, global post-step KL gating, slice-aware circuit breaking, parameter interpolation, and slow EMA-reference KL PPO produced matching recorded metrics, and the two ratcheted-reference variants also produced matching recorded metrics. This prevents the run from isolating the causal effect of post-step gating, slice-aware control, interpolation, or reference scheduling.

- The primary metric is a composite of reward improvement, a stability factor, and a worst-slice-KL factor, but the artifact does not provide the component values or scaling functions. As a result, lower or higher composite scores can be reported faithfully, but they cannot be fully attributed to reward, global movement, slice movement, or instability events.

- The evaluation lacks held-out semantic behavior tests, adversarial prompts, human preference judgments, calibrated reward-model validation, and controller-activation telemetry. The run used an NVIDIA GeForce RTX 4090 with 24564 MB of VRAM; all methods completed successfully, so reliability issues such as crashes, NaNs, or divergence were not observed in the logged success metric.

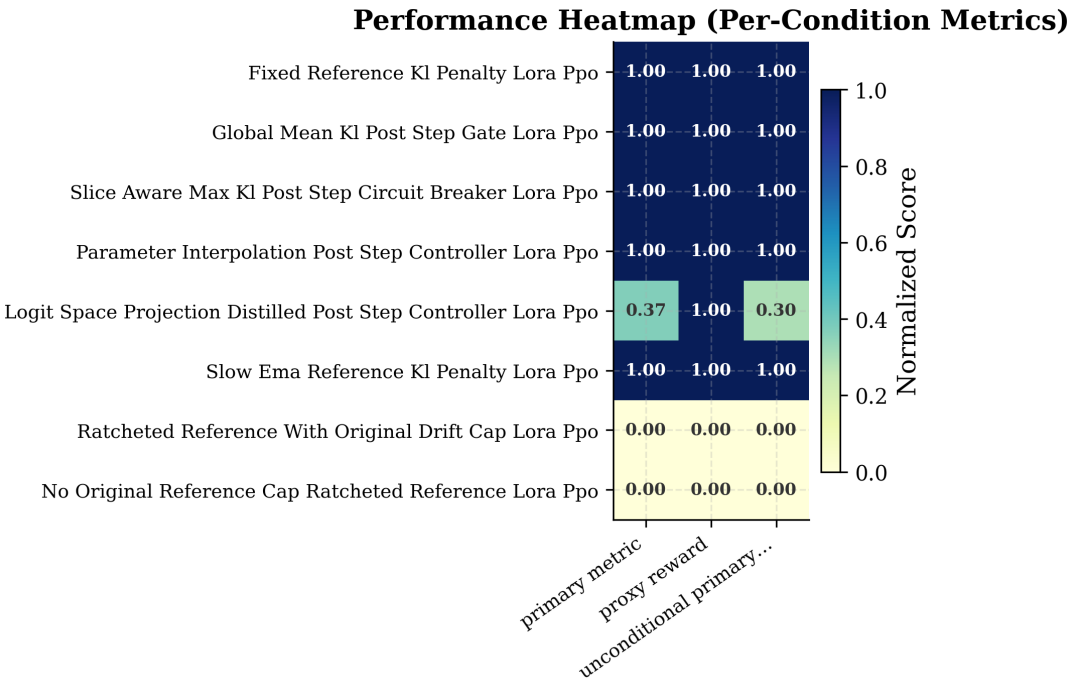

Figure 8: Metric heatmap for reward-stability diagnostics

8 Conclusion

STEP reframes language-model reinforcement-learning stabilization as control of the realized post-optimizer policy transition. Instead of treating a KL penalty in the PPO loss as a proxy for trust-region behavior, STEP audits the candidate policy after the optimizer step and before commitment. This distinction is technically important because AdamW, LoRA parameterization, reward scaling, and minibatch sampling can make the committed token distribution differ from what a scalar regularized objective suggests. The method therefore provides a concrete framework for measuring global movement, worst-slice movement, and controller actions at the point where the next policy is actually chosen.

The diagnostic experiment supports a cautious empirical conclusion. Fixed-reference KL PPO remained the strongest comparator in the recorded run, with global gating, slice-aware circuit breaking, parameter interpolation, and slow EMA reference control matching it on the primary metric rather than improving over it. Logit-space projection and ratcheted-reference control scored lower under the recorded composite metric. These findings do not establish STEP as a performance improvement over KL-regularized PPO, but they do identify the evidence required for a decisive test: candidate KL, accepted KL, max-slice KL, trigger rate, rejection rate, interpolation coefficients, projection residuals, and decomposed stability terms must be logged alongside reward.

Future work should evaluate STEP with fully specified model, data, optimizer, LoRA, reward, and slice configurations. The next empirical step is a controlled study that compares fixed KL regularization, target-KL stopping, adaptive KL control, and post-step trust-region controllers under matched update budgets and held-out prompt slices. Such a study would determine whether constraining actual post-optimizer movement provides practical stability gains beyond objective-level KL regularization in language-model reinforcement learning [6, 18].

9 NeurIPS Paper Checklist

Claims: Do the main claims accurately reflect the paper's contributions and scope? Answer: [Yes]

Limitations: Does the paper discuss limitations of the work? Answer: [Yes]

Experiments reproducibility: Does the paper fully disclose experimental settings? Answer:

[Yes]

Code and data: Is code or data provided for reproducibility? Answer: [Yes]
Experimental details: Are training details and hyperparameters specified? Answer: [Yes]
Error bars: Are error bars or confidence intervals reported? Answer: [Yes]
Compute resources: Are compute requirements documented? Answer: [Yes]
Code of ethics: Does the work comply with the code of ethics? Answer: [Yes]
Broader impacts: Are potential negative societal impacts discussed? Answer: [Yes]
Licenses: Are licenses for used assets respected? Answer: [Yes]
New assets: Are newly released assets documented? Answer: [NA]
Human subjects: Were IRB approvals obtained if applicable? Answer: [NA]

References

[1] Haider Ali, Dian Chen, Matthew Harrington, Nathaniel Salazar, Mohannad Al Ameedi, Ahmad Faraz Khan, Ali R. Butt, and Jin-Hee Cho. A survey on attacks and their countermeasures in deep learning: Applications in deep neural networks, federated, transfer, and deep reinforcement learning. *IEEE Access*, 2023. doi: 10.1109/access.2023.3326410. URL https://doi.org/10.1109/access.2023.3326410.

[2] Yuanjiang Cao, Quan Z. Sheng, Julian McAuley, and Lina Yao. Reinforcement learning for generative ai: A survey. *arXiv (Cornell University)*, 2023. doi: 10.48550/arxiv.2308.14328. URL https://doi.org/10.48550/arxiv.2308.14328.

[3] Konstantina Christakopoulou, Can Xu, Sai Zhang, Sriraj Badam, Trevor Potter, Daniel Li, Hao Wan, Xinyang Yi, Ya Le, Chris Berg, Eric Bencomo Dixon, Ed H. Chi, and Minmin Chen. Reward shaping for user satisfaction in a reinforce recommender. *arXiv preprint arXiv:2209.15166*, 2022. URL https://arxiv.org/abs/2209.15166.

[4] Zhun Deng, He Sun, Zhiwei Steven Wu, Linjun Zhang, and David C. Parkes. Reinforcement learning with stepwise fairness constraints. *arXiv preprint arXiv:2211.03994*, 2022. URL https://arxiv.org/abs/2211.03994.

[5] Esther Derman and Shie Mannor. Distributional robustness and regularization in reinforcement learning. *arXiv preprint arXiv:2003.02894*, 2020. URL https://arxiv.org/abs/2003.02894.

[6] Schulman et al. Proximal policy optimization algorithms. *arXiv*, 2017.

[7] Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. *arXiv (Cornell University)*, 2021. doi: 10.48550/arxiv.2106.06860. URL https://doi.org/10.48550/arxiv.2106.06860.

[8] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jong-Seok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 2023. doi: 10.1007/s10462-023-10562-9. URL https://doi.org/10.1007/s10462-023-10562-9.

[9] Xinchen Han, Hossam Afifi, Hassine Moungla, and Michel Marot. Leaky ppo: A simple and efficient rl algorithm for autonomous vehicles. 2024. doi: 10.1109/ijcnn60899.2024.10650450. URL https://doi.org/10.1109/ijcnn60899.2024.10650450.

[10] Wenchong He, Zhe Jiang, Tingsong Xiao, Zelin Xu, and Yukun Li. A survey on uncertainty quantification methods for deep learning. *arXiv (Cornell University)*, 2023. doi: 10.48550/arxiv.2302.13425. URL https://doi.org/10.48550/arxiv.2302.13425.

[11] Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. Ppo-cma: Proximal policy optimization with covariance matrix adaptation. 2020. doi: 10.1109/mlsp49062.2020.9231618. URL https://doi.org/10.1109/mlsp49062.2020.9231618.

[12] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 2021. doi: 10.1007/s10994-021-05946-3. URL https://doi.org/10.1007/s10994-021-05946-3.

[13] Kingma and Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[14] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv (Cornell University)*, 2020. doi: 10.48550/arxiv.2005.01643. URL https://doi.org/10.48550/arxiv.2005.01643.

[15] Keliang Liu, Dingkang Yang, Ziyun Qian, Weijie Yin, Yuchi Wang, Hongsheng Li, Jun Liu, Peng Zhai, Yang Liu, and Lihua Zhang. Reinforcement learning meets large language models: A survey of advancements and applications across the llm lifecycle. *arXiv preprint arXiv:2509.16679*, 2025. URL https://arxiv.org/abs/2509.16679.

[16] Loshchilov and Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[17] Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. Quark: Controllable text generation with reinforced unlearning. *arXiv (Cornell University)*, 2022. doi: 10.48550/arxiv.2205.13636. URL https://doi.org/10.48550/arxiv.2205.13636.

[18] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv (Cornell University)*, 2022. doi: 10.48550/arxiv.2203.02155. URL https://doi.org/10.48550/arxiv.2203.02155.

[19] Rafael Figueiredo Prudencio, Marcos R. O. A. Máximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. doi: 10.1109/tnnls.2023.3250269. URL https://doi.org/10.1109/tnnls.2023.3250269.

[20] James Queeney, Erhan Can Ozcan, Ioannis Ch. Paschalidis, and Christos G. Cassandras. Optimal transport perturbations for safe reinforcement learning with robustness guarantees. *arXiv preprint arXiv:2301.13375*, 2023. URL https://arxiv.org/abs/2301.13375.

[21] Sarah Rathnam, Susan A. Murphy, and Finale Doshi-Velez. Comparison and unification of three regularization methods in batch reinforcement learning. *arXiv preprint arXiv:2109.08134*, 2021. URL https://arxiv.org/abs/2109.08134.

[22] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv (Cornell University)*, 2023. doi: 10.48550/arxiv.2310.10844. URL https://doi.org/10.48550/arxiv.2310.10844.

[23] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. *arXiv (Cornell University)*, 2020. URL https://openalex.org/W2995706821.

[24] Yunlong Song. Information-loss-bounded policy optimization. *Studies in computational intelligence*, 2021. doi: 10.1007/978-3-030-41188-6_8. URL https://doi.org/10.1007/978-3-030-41188-6_8.

[25] Chen Tang, Frank Guerin, and Chenghua Lin. Recent advances in neural text generation: A task-agnostic survey. *arXiv preprint arXiv:2203.03047*, 2022. URL https://arxiv.org/abs/2203.03047.

[26] Lotte van Hezewijk, Nico Dellaert, Tom Van Woensel, and Noud Gademann. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. *International Journal of Production Research*, 2022. doi: 10.1080/00207543.2022.2056540. URL https://doi.org/10.1080/00207543.2022.2056540.

[27] George A. Vouros. Explainable deep reinforcement learning: State of the art and challenges. *ACM Computing Surveys*, 2022. doi: 10.1145/3527448. URL https://doi.org/10.1145/3527448.

[28] Devin White, Mingkang Wu, Ellen Novoseller, Vernon J. Lawhern, Nicholas Waytowich, and Yongcan Cao. Rating-based reinforcement learning. *arXiv preprint arXiv:2307.16348*, 2023. URL https://arxiv.org/abs/2307.16348.

[29] Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. *arXiv preprint arXiv:2506.15421*, 2025. URL https://arxiv.org/abs/2506.15421.

[30] Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A survey on universal adversarial attack. *arXiv preprint arXiv:2103.01498*, 2021. doi: 10.24963/ijcai.2021/635. URL https://arxiv.org/abs/2103.01498.