RELPO: RESOLUTION-LIFTED POLICY OPTIMIZATION FOR AUTOREGRESSIVE–DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive (AR)–diffusion hybrid language models combine revisable latent refinement with irreversible token realization, creating a distinct challenge for reinforcement learning (RL) post-training in assigning outcome credit across both forms of decision making. We observe that hybrid generation naturally induces a *native relaxation hierarchy*. Each partially resolved state defines a completion fiber containing all terminal sequences consistent with its committed tokens, while its unresolved latent state determines a continuation distribution over this fiber. Latent refinement reshapes this distribution while preserving possible completions, whereas autoregressive realization progressively contracts the fiber toward a single discrete output. Building on this structure, we introduce RELPO (**Resolution-Lifted Policy Optimization**), which propagates terminal outcome information through the hierarchy using a *Resolution-Lifted Outcome Value* (RLOV). RLOV defines a soft potential over partially resolved states and is learned from completed rollouts and terminal verifier rewards through a consistency relation spanning both refinement and resolution transitions. The resulting shared potential guides continuous latent refinement and autoregressive realization through a coupled policy improvement operator, allowing terminal outcomes to coordinate refinement, deferral, and token commitment throughout generation. Across eight mathematical reasoning and code-generation benchmarks, RELPO improves the shared Evo backbone from 52.5 to 59.7 average accuracy, compared with 56.4 for the strongest of eight RL baselines. Using separate potentials for the two branches reaches 57.7 under matched capacity, demonstrating the benefit of coupling refinement and realization through a shared potential. Under a unified protocol across six diffusion and AR–diffusion backbones, RELPO improves every model.

1 INTRODUCTION

Autoregressive (AR) language models define explicit sequence likelihoods and integrate naturally with reinforcement learning (RL) (Brown et al., 2020; Touvron et al., 2023; Grattafiori et al., 2024; Yang et al., 2024; Schulman et al., 2017; Ouyang et al., 2022). Each rollout consists of irreversible token choices, allowing outcome-based objectives to increase the likelihood of completions that receive high verifier rewards, with notable success in mathematical reasoning and code generation (Shao et al., 2024; Cobbe et al., 2021; Chen et al., 2021). Yet each local decision immediately constrains all subsequent continuations, making it difficult to revise earlier semantic choices when solving problems that require global coordination. Diffusion language models (Sahoo et al., 2024; Zhu et al., 2026a) instead maintain revisable states and progressively refine a global semantic configuration. Recently, AR–diffusion hybrid LLMs (Wu et al., 2023; Han et al., 2023; Arriola et al., 2025; Cheng et al., 2026; Wu et al., 2026) combine these two regimes, preserving latent refinement while uncertainty remains consequential and transitioning to discrete token realization once local structure becomes sufficiently stable.

A natural question that arises is how to effectively post-train such hybrid models using reinforcement learning. The challenge is credit assignment across two qualitatively different forms of decision making, latent refinement and discrete realization. Policy-gradient methods can in principle operate on this mixed-action Markov process, but sparse terminal rewards provide weak and high-variance

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.

learning signals for early revisable states. Moreover, marginal completion likelihoods require integrating over latent trajectories and resolution schedules (Zhao et al., 2026; Tang et al., 2026; Wang et al., 2026a). Trajectory-based methods (Wang et al., 2026b; Zhong et al., 2026) address the latter likelihood issue, but do not by themselves provide a shared credit signal that coordinates refinement, deferral, and token commitment throughout a hybrid rollout.

One possible approach is to introduce process rewards or explicitly supervise intermediate commitment decisions (Uesato et al., 2022; Lightman et al., 2024). We show that neither is necessary. Irreversible token commitments already induce a *native relaxation hierarchy* over the hybrid state space. A partially resolved hybrid state defines a *completion fiber*, the set of terminal sequences consistent with the tokens committed thus far, while its unresolved latent state induces a continuation distribution over that fiber. Diffusion-style updates redistribute probability mass within the current fiber while preserving the possibility of revision, whereas autoregressive realization contracts the fiber by fixing additional tokens. A hybrid rollout therefore traverses a sequence of increasingly constrained relaxations, from the full completion space to a single discrete completion.

This hierarchy provides a natural substrate for credit assignment. Each partially resolved state aggregates terminal outcomes over its possible continuations and therefore admits a conditional value defined directly from terminal rewards. A completed rollout can consequently propagate terminal outcome information through every partially resolved state it visits, including early states whose continuation fibers still contain many alternatives. This provides a state-dependent learning signal even in settings where group-relative comparisons become uninformative because all sampled completions for a prompt receive the same terminal reward.

Motivated by this observation, we propose RELPO (**Resolution-Lifted Policy Optimization**), an RL post-training framework for AR–diffusion hybrid LLMs. RELPO lifts terminal verifier rewards onto partially resolved hybrid states through a *Resolution-Lifted Outcome Value* (RLOV), a conditional soft value defined over the completion fiber of each state. Early in generation, RLOV aggregates outcomes across a broad distribution of possible completions. As resolution proceeds and the completion fiber contracts, it progressively sharpens toward the reward of an individual discrete completion.

Our construction builds on KL-regularized control. Exponential desirability linearizes the backward recursion (Kappen, 2005; Todorov, 2006), exponential tilting of the reference transition kernel yields the corresponding policy improvement operator, and control as inference connects these quantities to probabilistic policy optimization (Levine, 2018). RELPO instantiates this machinery on the mixed state space of AR–diffusion generation. Crucially, a single potential governs both branches of the hybrid process. During latent refinement, its gradient shifts probability mass toward regions whose completion fibers carry higher outcome value. During realization, differences in the same potential across candidate successor states reweight discrete commitments, determining both whether to resolve a position and which token to realize. Because both operations are derived from a shared potential and normalizer, refinement and commitment are optimized as coupled alternatives rather than as independently trained mechanisms.

RELPO then learns RLOV from groups of completed rollouts and their terminal rewards. A soft consistency relation propagates outcome information backward through both refinement and resolution transitions. The learned potential then guides the generator through coupled refinement and realization objectives, with reference regularization providing additional stability in regions with limited reward support.

Under binary verifier rewards, the population RLOV corresponds to a monotone transformation of the conditional success probability. Lifting this quantity onto the native resolution hierarchy gives each partially resolved state a potential that summarizes the outcomes available through its continuation fiber. This potential supports the soft backup for value learning, provides a latent gradient for continuous refinement, and defines jointly normalized successor differences for resolution decisions. These components allow a single outcome signal to coordinate refinement, deferral, and token commitment throughout the hybrid rollout. On a shared Evo backbone Wu et al. (2026), RELPO achieves an average score of 59.7, compared with 57.7 when using separate potentials for the two branches, 58.7 with a conventional success-probability critic using the same inputs, and 56.4 for the strongest of eight RL baselines.

Our contributions are as follows.

- We formalize the *native relaxation hierarchy* of AR–diffusion generation, in which partially resolved states define nested completion fibers induced by committed tokens, while latent refinement shapes the continuation distribution within each fiber.
- We lift sparse terminal outcome optimization onto this hierarchy through a *Resolution-Lifted Outcome Value*, a terminally grounded potential that aggregates outcomes over partially resolved states and can be learned without process supervision.
- We propose RELPO, an RL post-training framework that learns this lifted potential from hybrid rollouts and distills its coupled improvement operator into both continuous latent refinement and autoregressive realization.
- On shared AR-diffusion backbones, RELPO outperforms eight RL post-training baselines across all eight benchmarks, including compute-matched GRPO and TraDo. Under a unified protocol across six diffusion and AR–diffusion backbones, RELPO improves every model.

2 METHOD

2.1 A NATIVE RELAXATION HIERARCHY IN HYBRID GENERATION

Let x denote an input prompt and $y = (y_1, \ldots, y_N) \in \mathcal{V}^N$ a completed response. An AR–diffusion hybrid generator produces y through a sequence of partially resolved states $s_0 \to s_1 \to \cdots \to s_K \to y$. At generation stage k, we represent the hybrid state as $s_k = (c_k, y_{c_k}, z_{\bar{c}_k}, \rho_{\bar{c}_k})$, where $c_k \in \{0,1\}^N$ is a resolution mask, y_{c_k} denotes the discrete token values at resolved positions, $z_{\bar{c}_k} \in \mathbb{R}^{|\bar{c}_k| \times d}$ denotes the latent semantic variables at unresolved positions, and $\rho_{\bar{c}_k} \in [0,1]^{|\bar{c}_k|}$ denotes their progression coordinates. Here, $\bar{c}_k = \{i : c_{k,i} = 0\}$ is the set of unresolved positions. Variable-length outputs use an EOS token followed by an absorbing padding suffix up to N. The suffix is fixed at termination, and the verifier reads the prefix before EOS.

Some content of s_k has become irreversible, while the remaining content still admits revision. Let P_{c_k} project a complete sequence onto the resolved positions. We define the *completion fiber* as the inverse image of the committed values,

$$\mathcal{F}(s_k) = P_{c_k}^{-1}(y_{c_k}) = \{y' \in \mathcal{V}^N \mid y'_{c_k} = y_{c_k}\}. \tag{1}$$

The unresolved variables $(z_{\bar{c}_k}, \rho_{\bar{c}_k})$ parameterize the reference continuation law $p_{\pi_0}(\cdot \mid s_k)$, whose support lies within this fiber, so two states with the same committed tokens share a fiber and can differ in continuation probabilities. The set and its state-dependent measure together describe the partially resolved state, and the algorithm represents the measure implicitly through sampled transitions and a value network.

Let $J_{i,a}(s_k)$ denote the state obtained by resolving position $i \in \bar{c}_k$ as token $a \in \mathcal{V}$. Then $\mathcal{F}(J_{i,a}(s_k)) \subseteq \mathcal{F}(s_k)$, and refinement leaves $\mathcal{F}(s_k)$ unchanged. Thus, along a fully resolving hybrid trajectory,

$$\mathcal{F}(s_0) \supseteq \mathcal{F}(s_1) \supseteq \cdots \supseteq \mathcal{F}(s_K) = \{y\}. \tag{2}$$

Equation (2) is the native hierarchy of relaxations used throughout the paper. Early states correspond to broad sets of token-consistent completions, resolution contracts these sets toward the realized answer, and refinement reshapes the continuation distribution on a fixed set between resolution events. The nesting follows directly from the irreversibility of commitments (Appendix C.1). Pure autoregressive generation is the case in which every transition contracts the fiber, continuous latent diffusion with terminal decoding the case in which no intermediate transition does, and masked diffusion contracts fibers through unmasking under its native denoising kernel. Hybrid models expose both transition types as separately controllable mechanisms.

2.2 RESOLUTION-LIFTED OUTCOME VALUES

Let $R(x, y) \in \mathbb{R}$ be a terminal verifier reward and π_0 a reference hybrid policy. Direct policy optimization updates the probability of complete rollouts according to $R(x, y)$, and when successful completions occupy a thin subset of the rollout space this gives high-variance credit to the early semantic stages of generation. A partially resolved state represents an entire fiber of candidate completions, so terminal reward can be aggregated over the fiber before the generator commits to one realization. For a state s and temperature $\beta > 0$, the reward-weighted mass of its completion fiber

is $\mathcal{Z}_\beta(s) = \int_{\mathcal{F}(s)} \exp(R(x,y)/\beta)\, p_{\pi_0}(dy \mid s)$. For bounded R and a continuation law that varies continuously in the unresolved latent variables, $\mathcal{Z}_\beta(s)$ is a continuous functional of the latent state at a fixed resolution mask, and at a fully resolved state it degenerates to $\exp(R(x,y)/\beta)$ (Appendix C.2). Rewarding completions can still be rare within a large fiber, and the usefulness of the aggregate depends on how often the reference reaches them. The *Resolution-Lifted Outcome Value* (RLOV) is the terminally grounded soft value built from this mass.

Definition 1 (Resolution-Lifted Outcome Value)**.** Given a reference hybrid policy π_0, terminal reward $R(x,y)$, and temperature $\beta > 0$, the Resolution-Lifted Outcome Value of state s at generation time k is

$$\Phi_k^\star(s) = \beta \log \mathbb{E}_{Y \sim \pi_0(\cdot|s)} \left[\exp\left(\frac{R(x,Y)}{\beta} \right) \right] = \beta \log \mathcal{Z}_\beta(s). \tag{3}$$

The RLOV is a soft terminal desirability over the completion fiber induced by s, summarizing the terminal reward distribution over the completions reachable from that state. At complete resolution, the fiber degenerates to a singleton and the RLOV recovers the original terminal objective, $\Phi_K^\star(y) = R(x,y)$. At early stages, $\Phi_k^\star(s)$ averages over a broad semantic solution set. Thus, it interpolates between a dense relaxed objective over unresolved hybrid states and the sparse verifier objective at the terminal sequence.

Theorem 1 (Resolution-Lifted Outcome Consistency)**.** *Let s_k be a partially resolved hybrid state and assume that $R(x,y)$ is bounded. Then the RLOV satisfies*

$$\Phi_k^\star(s_k) = \beta \log \mathbb{E}_{s_{k+1} \sim \mathcal{K}_0(\cdot|s_k)} \left[\exp\left(\frac{\Phi_{k+1}^\star(s_{k+1})}{\beta} \right) \right], \tag{4}$$

with terminal condition $\Phi_K^\star(y) = R(x,y)$, where $\mathcal{K}_0$ denotes the one-step transition kernel of the reference hybrid generator.

Equation (4) is a soft Bellman recursion over the native resolution hierarchy. The successor distribution contains two qualitatively different transitions, in-fiber latent refinement and resolution jumps that contract the completion fiber, so the RLOV propagates terminal outcome information through the same mixed transition structure used by the hybrid generator at inference. The recursion rests on the Markov property of the reference kernel and holds for any Markov generator. The fiber hierarchy specifies what the expectation ranges over at each stage, a mixture of in-fiber moves and sub-fiber jumps from which Section 2.3 derives one update for both branches. The proof of Theorem 1 is given in Appendix C.3.

For a binary verifier, writing $p(s) = \Pr_{\pi_0}(R = 1 \mid s)$ gives $\Phi_k^\star(s) = \beta \log[1 + (e^{1/\beta} - 1)\, p(s)]$, a strictly increasing function of the conditional success probability, so the population RLOV ranks states exactly as a calibrated success critic does. The construction acts through the objects derived from the potential, the soft backup of Eq. (4) used as regression target, the latent gradient that gives the refinement drift, and the successor differences, normalized jointly over refinement and resolution, that give the resolve-or-defer policy of Section 2.3. Section 3.2 compares this route against a success-probability critic trained on the arithmetic backup.

We parameterize the RLOV by a potential network $\Phi_\phi(s_k, k)$. For each rollout group sampled from the behavior policy π_{old}, we compute normalized group-relative advantages $A_g = (R(x,y_g) - \mu_R)/(\sigma_R + \epsilon)$ and define positive regression weights $w_g = \exp(A_g/\beta)/\sum_{j=1}^G \exp(A_j/\beta)$. Within a group, $w_g \propto \exp(R_g/[\beta(\sigma_R + \epsilon)])$, so standardization gives a group-dependent regression temperature. The RLOV target in Eq. (3) retains the fixed temperature β. Positive weighting preserves the realizable zero-residual solution under the coverage assumption and shifts the finite-capacity regression emphasis toward states on successful trajectories. Appendix D.1 compares uniform and raw-reward weights. We train the potential through a weighted soft Bellman residual,

$$\mathcal{L}_{\text{RLOV}} = \mathbb{E}_{g,k}\left[w_g \left(\Phi_\phi(s_{g,k}, k) - \beta \log \mathbb{E}_{s' \sim \mathcal{K}_0(\cdot|s_{g,k})} \left[\exp\left(\frac{\Phi_{\bar{\phi}}(s', k+1)}{\beta} \right) \right] \right)^2 \right], \tag{5}$$

where $\Phi_{\bar{\phi}}$ is an exponential-moving-average target network, and the terminal boundary condition is enforced through $\mathcal{L}_{\text{term}} = \mathbb{E}_g[w_g(\Phi_\phi(y_g, K) - R(x,y_g))^2]$. In practice, the expectation in Eq. (5) is approximated by successors sampled from the reference kernel, which the model's rollout dynamics

already instantiate, so no annotation beyond the final verifier reward is required. With M sampled successors, concavity of the logarithm gives a downward bias in the estimated soft target. The estimator is consistent as $M \to \infty$, while Eq. (5) and Proposition 1 concern the exact population expectation.

Proposition 1 (Outcome-Only Identification)**.** *Assume that the reference transition kernel $\mathcal{K}_0$ is Markovian, the function class for Φ_ϕ is realizable, and terminal rewards are observed without bias. Let d_k^0 and d_k^{old} denote reference and behavior state occupancies, and assume $d_k^0 \ll d_k^{\mathrm{old}}$ at every stage. At a zero-residual fixed point of the population objectives, Φ_ϕ recovers Eq. (3) reference-almost everywhere on the covered reachable set.*

Proposition 1 identifies the target of outcome-only value learning under population assumptions. The proof is given in Appendix C.4.

2.3 Resolution-Lifted Policy Optimization

The RLOV defines the policy improvement direction for both components of hybrid generation, which provides a natural post-training objective for AR–diffusion models. We call the resulting method ReLPO. Specifically, let the reference hybrid generator contain a continuous latent refinement field $F_0(s_k, k)$, which evolves unresolved semantic variables $z_{\bar{c}_k}$, and a resolution policy $\lambda_0(i, a \mid s_k, k)$, which gives the one-step probability of resolving unresolved position $i \in \bar{c}_k$ as token $a \in \mathcal{V}$. Let $A_k(s_k)$ denote the local covariance or metric associated with the latent refinement process. The complementary probability $\lambda_0(\mathrm{defer} \mid s_k, k) = 1 - \sum_{i,a} \lambda_0(i, a \mid s_k, k)$ selects latent refinement. In the continuous-time or small-step limit, the RLOV-guided refinement field is

$$F^\star(s_k, k) = F_0(s_k, k) + A_k(s_k) \nabla_{z_{\bar{c}_k}} \frac{\Phi_k^\star(s_k)}{\beta}. \tag{6}$$

Equation (6) transports unresolved semantic variables toward regions whose completion fibers admit higher terminal outcome value.

For autoregressive realization, let $J_{i,a}(s_k)$ denote the state obtained by resolving position i as token a. For notational simplicity, Eq. (7) writes the realization rule for a single position-token pair. More generally, a resolution transition may jointly commit multiple currently unresolved positions within the same stage. The corresponding RLOV-guided realization policy is

$$\lambda^\star(i, a \mid s_k, k) = \lambda_0(i, a \mid s_k, k) \exp\left(\frac{\Phi_{k+1}^\star(J_{i,a}(s_k)) - \Phi_k^\star(s_k)}{\beta} \right). \tag{7}$$

Equation (7) determines both whether a semantic variable should be resolved and how it should be realized. The total resolution probability is $\Lambda^\star(i \mid s_k, k) = \sum_{a \in \mathcal{V}} \lambda^\star(i, a \mid s_k, k)$, while the token distribution conditional on resolving position i is $\pi^\star(a \mid s_k, i, \mathrm{resolve}) = \lambda^\star(i, a \mid s_k, k) / \Lambda^\star(i \mid s_k, k)$. A large increase in RLOV after a resolution jump raises the corresponding probability, encouraging early AR realization. When committing a position sharply decreases the value of its remaining completion fiber, the model retains the position in the revisable latent regime and continues diffusion-style refinement. Equation (7) is the single-position restriction of the tilted kernel to resolution atoms, while Eq. (6) gives a local realization of its refinement branch. At the stage level, a resolution atom may jointly commit a subset of currently unresolved positions. Both follow the KL-regularized policy improvement operator associated with the terminal reward $R(x, y)$, with both branch updates depending on shared normalization over successor states. The proof is given in Appendix C.5.

We distill the RLOV-induced targets into the parameterized hybrid model. For the continuous refinement component, we optimize

$$\mathcal{L}_{\mathrm{flow}} = \mathbb{E}_{s_k} \left[\left\| F_\theta(s_k, k) - F_0(s_k, k) - A_k(s_k) \nabla_{z_{\bar{c}_k}} \frac{\Phi_\phi(s_k, k)}{\beta} \right\|^2_{A_k(s_k)^{-1}} \right]. \tag{8}$$

For the resolution component, the population target is the normalized potential-tilted kernel,

$$q^\phi(ds' \mid s_k) = \frac{\mathcal{K}_0(ds' \mid s_k) \exp(\Phi_\phi(s', k+1)/\beta)}{\mathbb{E}_{\mathcal{K}_0}[\exp(\Phi_\phi(s', k+1)/\beta)]}.$$

Its resolution masses define the stage-level realization distribution over $\mathcal{A}_{\mathrm{res}}(s_k)$, where each $b \in \mathcal{A}_{\mathrm{res}}(s_k)$ may jointly resolve one or more currently unresolved position-token pairs, and its integrated refinement mass defines $\lambda^{\phi}(\mathrm{defer} \mid s_k, k)$. These probabilities sum to one. At exact consistency, the denominator equals $\exp(\Phi_{\phi}(s_k, k)/\beta)$, recovering Eq. (7) for singleton resolution actions. We minimize

$$\mathcal{L}_{\mathrm{resolve}} = -\mathbb{E}_{s_k}\left[\sum_{b \in \mathcal{B}(s_k)} \lambda^{\phi}(b \mid s_k, k) \log \lambda_{\theta}(b \mid s_k, k)\right], \qquad \mathcal{B}(s_k) = \{\mathrm{defer}\} \cup \mathcal{A}_{\mathrm{res}}(s_k). \quad (9)$$

Here, k indexes refinement-resolution stages rather than individual token commitments. A single stage may therefore resolve multiple positions, and K denotes the number of such stages rather than the maximum response length.

The population objective admits a sampled implementation that self-normalizes the successor weights over the four reference successors drawn for Eq. (5), with the weights detached during policy updates (Appendix B). To preserve the pretrained geometry outside reward-supported regions, we include a reference regularizer

$$\mathcal{L}_{\mathrm{ref}} = \mathbb{E}_{s_k}\left[\|F_{\theta}(s_k, k) - F_0(s_k, k)\|^2\right] + \gamma D_{\mathrm{KL}}\left(\lambda_{\theta}(\cdot \mid s_k, k) \parallel \lambda_0(\cdot \mid s_k, k)\right). \quad (10)$$

The KL in Eq. (10) is taken over the full decision set $\mathcal{B}(s_k)$, including deferral, and averaged over the same state occupancy. The final objective is

$$\mathcal{L}_{\mathrm{ReLPO}} = \mathcal{L}_{\mathrm{RLOV}} + \lambda_{\mathrm{term}}\mathcal{L}_{\mathrm{term}} + \lambda_{\mathrm{flow}}\mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{resolve}}\mathcal{L}_{\mathrm{resolve}} + \lambda_{\mathrm{ref}}\mathcal{L}_{\mathrm{ref}}.$$

3 Experiments

3.1 Experimental Setting

ReLPO is an RL post-training method, and every run starts from a fixed supervised diffusion or hybrid language model checkpoint. This checkpoint initializes the trainable generator π_{θ}, while a frozen copy serves as the reference policy π_0, which defines the RLOV in Eq. (3), provides the transition kernel $\mathcal{K}_0$ in Eq. (4), and anchors the regularization in Eq. (10). The RL prompt pool contains roughly 58K verifiable problems drawn from mathematics and code, filtered to 41K by discarding prompts the reference policy solves in all eight or none of eight rollouts, which keeps group-relative advantages defined for every method. Rewards are binary with no partial credit and no step-level annotation. Rollouts are collected in groups of $G = 8$ at temperature 1.0, run the hybrid generator for $K = 20$ stages, and generate at most 2048 new tokens. One iteration draws 256 prompts and we train for 500 iterations, about 2.0B generated tokens. The potential Φ_{ϕ} is a value head over a low-rank adapted copy of the backbone trunk, paired with an exponential-moving-average target, and we set the temperature $\beta = 0.5$. Training runs on 64 NVIDIA A100 GPUs for roughly 5.6 days per run, and all reported numbers are means over three RL seeds (Appendix B).

We mainly run two comparisons. The first holds the backbone fixed and varies the post-training objective, evaluating ReLPO against ESPO (Ou et al., 2026), d1 (Zhao et al., 2026), WD1 (Tang et al., 2026), StableDRL (Zhong et al., 2026), SPG (Wang et al., 2026a), TraDo (Wang et al., 2026b), J-GRPO (Ni et al., 2026), and GRPO (Shao et al., 2024). Every method starts from the identical Evo-XXL (8.5B parameters) (Wu et al., 2026) checkpoint. Both refinement and resolve/defer control are available to every method, with each objective applied to the transitions covered by its formulation. The second holds the objective fixed and varies the backbone, applying ReLPO to five further models initialized from their checkpoints following (Wu et al., 2026), including the hybrids BD3-LM (Arriola et al., 2025), ARD (Wu et al., 2023), and SDAR (Cheng et al., 2026), and the diffusion-only LLaDA (Zhu et al., 2026a) and MDLM (Sahoo et al., 2024). The benchmark suite covers mathematical and scientific reasoning (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AIME 2024, GPQA (Rein et al., 2023)) and code generation (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), SWE-Bench Verified (Jimenez et al., 2024), SWE-Context Bench (Zhu et al., 2026b)). Every benchmark uses greedy decoding at temperature 0 with a single sample of at most 2048 tokens and $K = 20$ stages. Code tasks report pass@1 under the benchmark tests, and mathematics tasks report final-answer accuracy under the same symbolic checker used during training. The two SWE benchmarks are run as single-pass patch generation from the issue text and the benchmark's code

context within the 4K window, without repository navigation, tool calls, or test feedback, so their scores sit well below those of agentic harnesses. Appendix B gives the full protocol.

Table 1: Benchmark results on the shared Evo backbone at 2.0B rollout tokens. ReLPO uses $1.24\times$ GRPO training FLOPs. Means over three seeds. Figure 2 summarizes seed variability.

Category	Benchmark	SFT	GRPO	ESPO	d1	WD1	StableDRL	SPG	TraDo	J-GRPO	ReLPO
Math	GSM8K	89.3	92.0	90.5	90.2	91.4	91.0	91.6	92.0	91.8	**92.9**
	MATH	78.8	81.4	80.4	79.8	81.2	81.4	80.8	82.4	82.6	**84.4**
	AIME 2024	10.0	20.0	16.7	13.3	16.7	16.7	20.0	20.0	20.0	**30.0**
	GPQA	39.1	40.2	40.4	39.9	40.9	40.9	41.4	41.4	41.9	**43.4**
Coding	HumanEval	86.6	89.6	88.4	87.8	89.0	89.6	89.0	90.2	89.6	**92.1**
	MBPP	81.1	83.2	82.2	81.6	83.0	82.4	83.4	83.6	83.2	**85.4**
	SWE-Bench Verified	9.6	12.2	10.6	10.0	11.4	11.0	11.8	12.6	12.2	**16.2**
	SWE-Context	25.3	28.6	26.3	25.5	27.4	26.9	28.2	29.0	28.5	**33.0**
Overall Avg.		52.5	55.9	54.4	53.5	55.1	55.0	55.8	56.4	56.2	**59.7**

3.2 Main Results

Table 1 compares ReLPO against eight RL post-training methods on the shared Evo backbone. ReLPO is the strongest method on all eight benchmarks and lifts the overall average from 52.5 for the supervised checkpoint to 59.7, against 56.4 for TraDo, the best baseline. The margin is widest on the benchmarks with the sparsest reward-bearing completions, 10.0 points on AIME 2024 (30.0 vs. 20.0), 4.0 points on SWE-Context (33.0 vs. 29.0), and 3.6 points on SWE-Bench Verified (16.2 vs. 12.6). AIME 2024 contains 30 problems, so its gap corresponds to three additional correct answers, and with AIME 2024 excluded the seven-benchmark mean is 63.9 for ReLPO and 61.6 for TraDo. Seed variation and finite-test uncertainty are quantified in Appendix D.1.

Table 2 applies ReLPO to six backbones under one protocol and measures the gain within each backbone. Every model improves on every benchmark, the four hybrid backbones by 4.0 to 7.2 points on average and the two masked-diffusion backbones by 2.9. The table establishes transfer across architectures, and comparisons with other RL methods are confined to the shared Evo backbone. On masked-diffusion backbones the potential acts through the tilted denoising kernel alone, while the hybrid backbones support both coupled updates of Eqs. (8)–(9), and backbone quality and usable reward frequency under the shared prompt pool also shape each gain. Figure 2 shows that ReLPO has the lowest seed standard deviation on the shared Evo backbone, 0.32 points against 0.37–0.71 for the RL baselines, and 0.32 to 0.77 across the six backbones.

3.3 Ablation Study

Ablations use the Evo-XXL backbone and report the average over the four mathematics benchmarks and the average over the four coding benchmarks.

Table 3 separates the learned potential from the two policy updates, with each component contributing to performance. Replacing RLOV with a flat terminal advantage reduces accuracy by 4.1 points from 59.7 to 55.6, while direct Monte-Carlo soft targets cost 2.9 points. The latter have finite-sample bias and high variance at low resolution ratios, while Proposition 1 shows how the recursion determines the potential at partially resolved states. Removing the flow or resolution update reduces accuracy by 3.4 and 2.6 points and increases refinement usage to 3.65 and 3.92 steps per token, respectively. Removing the reference anchor lowers the mean to 55.6 and raises the seed standard deviation from 0.3 to 1.1. The full ReLPO achieves both the highest accuracy and the lowest refinement usage at 2.67 steps per token, compared with 3.08 for the supervised checkpoint.

Table 4 isolates the role of the shared potential. A matched-capacity variant with separate potentials for the two branches reaches 57.7 compared with 59.7 for ReLPO, reduces correlation with the Monte-Carlo target from 0.81 to 0.63, and increases premature resolution from 8.0% to 12.3%. Separate potentials therefore recover part of the improvement over flat outcome RL, while coupling the two branches through a shared potential provides an additional 2.0 points. Single-branch tilts further decrease performance to 56.7 and 56.0, with complementary failure modes detailed in Appendix D.2.

We then test whether resolution lifting improves sample efficiency in the sparse outcome regime in Figure 1. We vary the number of rollouts per prompt while fixing the total number of generated rollout

Table 2: Architecture generality of ReLPO. Gains are measured within backbone. Means over three seeds. Figure 2 summarizes seed variability.

	Evo-XXL		BD3-LM 7B		ARD 7B		LLaDA 8B		MDLM 9B		SDAR 8B	
	Base	+ReLPO	Base	+ReLPO	Base	+ReLPO	Base	+ReLPO	Base	+ReLPO	Base	+ReLPO
Math												
GSM8K	89.3	**92.9**	85.8	**89.2**	31.2	**36.0**	78.8	**81.1**	76.9	**79.0**	81.3	**85.3**
MATH	78.8	**84.4**	74.4	**79.0**	15.8	**19.8**	26.8	**29.2**	44.4	**47.0**	52.6	**57.4**
AIME 2024	10.0	**30.0**	10.0	**23.3**	3.3	**10.0**	3.3	**10.0**	6.7	**13.3**	6.7	**16.7**
GPQA	39.1	**43.4**	34.8	**38.4**	24.7	**27.8**	32.3	**34.3**	33.3	**35.4**	34.3	**37.4**
Coding												
HumanEval	86.6	**92.1**	83.5	**87.8**	17.1	**22.0**	48.2	**51.2**	69.5	**72.6**	72.0	**76.8**
MBPP	81.1	**85.4**	67.4	**71.8**	20.8	**25.4**	34.4	**37.2**	75.0	**77.4**	67.0	**71.4**
SWE-Bench Verified	9.6	**16.2**	6.6	**11.6**	0.8	**1.8**	1.4	**2.4**	4.0	**5.6**	4.4	**8.2**
SWE-Context	25.3	**33.0**	20.2	**26.9**	3.5	**6.1**	7.2	**9.8**	12.8	**15.4**	14.9	**20.7**
Avg. gain	**+7.2**		**+5.7**		**+4.0**		**+2.9**		**+2.9**		**+5.1**	

Table 3: Component ablation of ReLPO on Evo. "Flat outcome RL" applies the terminal group-relative advantage uniformly to the hybrid trace without a RLOV, and "Direct MC target" regresses the potential to Monte-Carlo terminal returns without the soft consistency objective. "Steps/tok." is the mean number of latent refinement updates per generated token. RL variants report mean ± standard deviation across three seeds.

Variant	RLOV	Soft consistency	Flow update	Resolution update	Reference anchor	Math Avg.	Code Avg.	Overall Avg.	Steps/tok.
SFT	–	–	–	–	–	54.3	50.7	52.5	3.08
Flat outcome RL	–	–	✓	✓	✓	58.0 ± 0.5	53.2 ± 0.6	55.6 ± 0.5	3.46 ± 0.12
RLOV w/ direct MC target	✓	–	✓	✓	✓	59.2 ± 0.6	54.4 ± 0.5	56.8 ± 0.5	3.04 ± 0.10
w/o flow update	✓	✓	–	✓	✓	58.7 ± 0.4	53.8 ± 0.4	56.3 ± 0.4	3.65 ± 0.09
w/o resolution update	✓	✓	✓	–	✓	59.5 ± 0.5	54.7 ± 0.5	57.1 ± 0.4	3.92 ± 0.13
w/o reference anchor	✓	✓	✓	✓	–	58.2 ± 1.0	53.0 ± 1.2	55.6 ± 1.1	2.78 ± 0.22
ReLPO	✓	✓	✓	✓	✓	$\mathbf{62.7 \pm 0.4}$	$\mathbf{56.7 \pm 0.4}$	$\mathbf{59.7 \pm 0.3}$	$\mathbf{2.67 \pm 0.07}$

Table 4: Shared and branch-conditioned potentials on the same Evo checkpoint at matched parameter count and rollout budget. Both branches are active in the first two rows. $\rho(\Phi, \widehat{\Phi}_{\mathrm{MC}})$ is pooled over the four resolution strata, and for the factorized arm it is the mean over the two branch potentials.

Variant	Both branches tilted	Shared potential	Math Avg.	Code Avg.	Overall Avg.	$\rho(\Phi, \widehat{\Phi}_{\mathrm{MC}})$	Premature res. (%) ↓	Steps/ tok. ↓
ReLPO (shared $\Phi^\star$)	✓	✓	62.7	56.7	59.7	0.81	8.0	2.67
Factorized (Φ^d, Φ^a)	✓	–	60.2	55.1	57.7	0.63	12.3	3.06
Refinement-only tilt	–	✓	59.3	54.1	56.7	0.64	15.5	3.48
Resolution-only tilt	–	✓	58.5	53.5	56.0	0.61	11.1	2.45

tokens across methods, so smaller groups yield weaker within-prompt outcome comparisons but identical total sampling compute. All three methods degrade as the group shrinks, at different rates. Between $G = 16$ and $G = 2$, GRPO and TraDo each lose 2.3 points overall, while ReLPO loses 1.7. ReLPO at $G = 2$ reaches 58.2, above both GRPO (56.3) and TraDo (57.0) at $G = 16$, so it exceeds with two rollouts per prompt what the group-relative baselines obtain with sixteen. A group-relative estimator has nothing to compare when all G completions receive the same terminal reward, and the probability of such a degenerate group grows quickly as G falls. The RLOV scores states along each individual trajectory through Eq. (5), so a single reward-bearing rollout still propagates information to every partially resolved state it passed through.

3.4 Calibration of the Resolution-Lifted Outcome Value

We test whether the learned RLOV captures terminal outcome desirability at partially resolved states. We cache hybrid states from held-out rollouts and stratify them by their resolution ratio $r(s) = \|c\|_1/N$, where c is the resolution mask. For every cached state, we estimate a Monte-Carlo soft outcome target using M independent continuations, $\widehat{\Phi}_{\mathrm{MC}}(s) = \beta \log \left[\frac{1}{M} \sum_{m=1}^{M} \exp\left(\frac{R_m}{\beta}\right) \right]$. We also test whether the potential difference of a candidate resolution transition predicts the empirical benefit of resolving over deferring it for a fixed number of refinement steps.

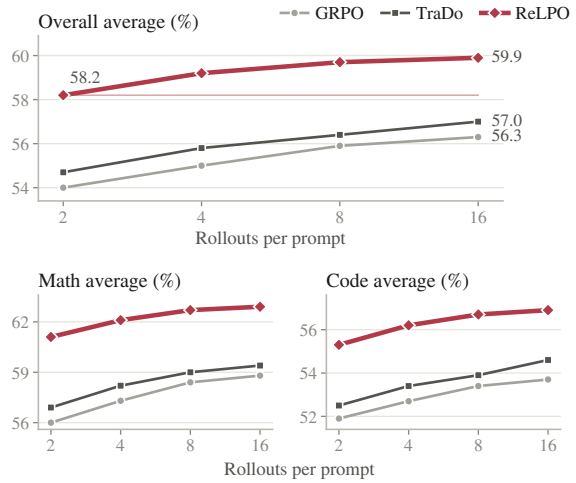


Figure 1: Rollout-budget ablation on Evo under a fixed total rollout-token budget. Lines show math, code, and overall averages.

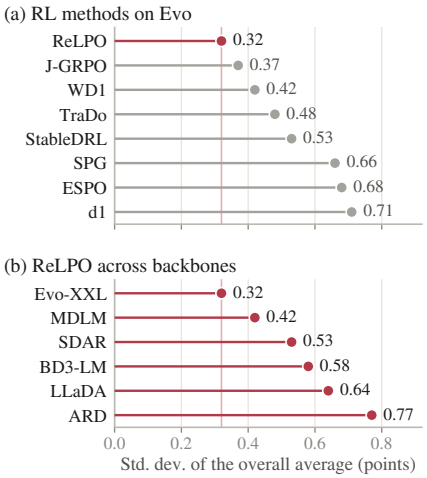


Figure 2: Standard deviation of the average across 3 seeds for the shared-backbone and cross-backbone comparisons, in percentage points.

Table 5: Validation of the Resolution-Lifted Outcome Value across the native resolution hierarchy. "Decision sign accuracy" measures agreement between the sign of the predicted potential difference and the sign of the empirical resolve-versus-defer outcome difference.

Metric	$r \in [0, .25)$	$r \in [.25, .50)$	$r \in [.50, .75)$	$r \in [.75, 1)$
Spearman $\rho(\Phi_\phi, \widehat{\Phi}_{\mathrm{MC}}) \uparrow$	0.64	0.73	0.81	**0.89**
$\mathrm{MAE}(\Phi_\phi, \widehat{\Phi}_{\mathrm{MC}}) \downarrow$	0.109	0.091	0.068	**0.041**
Decision sign accuracy (%) $\uparrow$	72.1	77.5	83.0	**89.1**
Resolution-value calibration error $\downarrow$	0.083	0.065	0.046	**0.026**

Table 5 indicates that the potential tracks terminal outcome across the whole hierarchy, with fidelity that improves as the completion fiber contracts. Spearman correlation against the Monte-Carlo target rises from 0.64 on the broadest stratum to 0.89 on the most resolved one, and mean absolute error falls from 0.109 to 0.041 on a potential whose range is $[0, 1]$ under a binary verifier and $\beta = 0.5$. A correlation of 0.64 and a decision sign accuracy of 72.1% at $r < 0.25$ mean that the potential is already informative while three quarters of the sequence remains revisable, where terminal policy-gradient estimates can have high variance. Decision sign accuracy reaches 89.1% on the most resolved stratum, so at the most resolved states, the predicted potential difference agrees with the empirical resolve-versus-defer outcome in roughly nine cases out of ten. These metrics measure agreement with the reference-continuation target that the method fits, and Appendix D.6 examines how the resulting policy allocates refinement and resolution relative to flat outcome-RL objectives.

4 Conclusion

We introduced ReLPO, a reinforcement learning post-training method for autoregressive–diffusion language models. Hybrid generation carries a hierarchy of progressively contracted completion fibers, and a single terminally grounded potential on this hierarchy, learned from completed rollouts and verifier rewards, induces one KL-regularized improvement step for continuous latent refinement and for discrete autoregressive realization. On a shared Evo backbone, ReLPO outperforms eight RL baselines on all eight benchmarks, including compute-matched GRPO and TraDo, and a factorized potential recovers only part of its gain. Applied to six diffusion and hybrid backbones under one protocol, it improves every model.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing the manuscript. The authors reviewed all AI-assisted material, including all text and claims with the aid of generative AI.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

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

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026.

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

Yijia Fan, Jing Yang, Mingyu Liu, Kaitong Cai, Jian Wang, Keze Wang, and Jusheng Zhang. Reinforcement learning for diffusion LLMs via energy-based gibbs alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://aclanthology.org/2026.acl-long.2131/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024.

Hilbert J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics Theory and Experiment*, 2005(11):P11011, 2005. URL https://arxiv.org/abs/physics/0505066.

Sergey Levine. Reinforcement learning and control as probabilistic inference tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. URL https://arxiv.org/abs/1805.00909.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

LiveCodeBench Team. LiveCodeBench. https://github.com/LiveCodeBench/LiveCodeBench, 2025.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled rl for diffusion llms emerges from a sequence-level perspective. In *International Conference on Learning Representations*, volume 2026, pp. 4941–4966, 2026.

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.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749*, 2021.

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, Xiao 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.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *International Conference on Learning Representations*, 2026. arXiv:2507.08838.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, volume 19, 2006. URL https://papers.nips.cc/paper_files/paper/2006/file/d806ca13ca3449af72a1ea5aedbed26a-Paper.pdf.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. Spg: Sandwiched policy gradient for masked diffusion language models. In *International Conference on Learning Representations*, 2026a. arXiv:2510.09541.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *International Conference on Learning Representations*, volume 2026, pp. 101476–101501, 2026b.

Junde Wu, Minhao Hu, Jiayuan Zhu, Yuyuan Liu, Tianyi Zhang, Kang Li, Jingkun Chen, Jiazhen Pan, Min Xu, and Yueming Jin. Evo: Autoregressive–diffusion large language models with evolving balance. *arXiv preprint arXiv:2603.06617*, 2026.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing*

Systems, 38:56729–56762, 2026.

Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. Stabilizing reinforcement learning for diffusion language models. *arXiv preprint arXiv:2603.06743*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026a.

Jiayuan Zhu, Junde Wu, Minhao Hu, Shengda Zhu, Jiazhen Pan, Weixiang Shen, Yijun Yang, Fenglin Liu, Jianye Hao, Yueming Jin, Qirong Ho, and Min Xu. SWE context bench: A benchmark for context learning in coding. *arXiv preprint arXiv:2602.08316*, 2026b.

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyanag He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. DiRL an efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*, 2025. URL https://arxiv.org/abs/2512.22234.

A Related Work

A.1 Autoregressive and Diffusion Hybrid Models

Recent work has explored hybrid language generators that combine the causal factorization of autoregressive decoding with the revisability of diffusion-based generation. Early approaches include step-unrolled denoising autoencoders (Savinov et al., 2021) and flow-based text models (Hoogeboom et al., 2021), which demonstrate that iterative denoising and sequential generation can be combined through continuous latent dynamics or unrolled refinement. However, these methods typically use fixed generation schedules or separately specified objectives, and do not explicitly expose a controllable transition between unresolved semantic states and discrete token realization.

More recent work develops tighter AR–diffusion couplings with explicit causal structure. AR-Diffusion (Wu et al., 2023) imposes a position-dependent denoising schedule such that earlier tokens are resolved first and condition later positions. SSD-LM (Han et al., 2023) similarly adopts a semi-autoregressive diffusion formulation over structured blocks. Block Diffusion (Arriola et al., 2025) interpolates between autoregressive and diffusion generation by generating blocks autoregressively while denoising within each block, improving variable-length generation and sampling efficiency. Evo (Wu et al., 2026) further models AR realization and diffusion-style refinement as two regimes of a shared latent trajectory, allowing token-wise progression variables to determine which semantic regions remain revisable and which transition toward autoregressive realization.

A.2 RL Post-Training for Autoregressive and Diffusion Language Models

Outcome-based RL for autoregressive language models typically optimizes a KL-regularized policy objective over left-to-right token trajectories. PPO (Schulman et al., 2017) and RLHF-style post-training (Ouyang et al., 2022) established the standard policy-optimization framework, while GRPO (Shao et al., 2024) adapts this setting to verifiable reasoning rewards through group-relative advantages. These methods rely on the autoregressive factorization, under which the likelihood of a sampled rollout is directly available as a product of token-level conditional probabilities.

RL for diffusion language models faces a different technical obstacle. The likelihood of a final completion generally marginalizes over latent corruption and denoising trajectories. d1 (Zhao et al., 2026) introduced DIFFU-GRPO for masked diffusion LLMs, while wd1 (Tang et al., 2026) reformulates diffusion RL as ratio-free weighted policy optimization. SPG (Wang et al., 2026a) further shows that one-sided ELBO surrogates can produce biased policy updates and derives a sign-aware objective using upper and lower likelihood bounds. These methods address how to construct valid or efficient outcome-RL updates for diffusion trajectories. DiRL introduces DiPO, a generation-step GRPO objective for discrete diffusion (Zhu et al., 2025), and J-GRPO (Ni et al., 2026) studies outcome-based optimization under a fixed generation order. Diffusion-Gibbs Alignment learns an energy model over complete samples and matches the reward-tilted Gibbs distribution of final outputs through weighted diffusion alignment (Fan et al., 2026). ReLPO shares this Gibbs-tilt view of the target distribution and differs in the object it learns. Its potential is defined on partially resolved states, satisfies a backup across refinement and resolution transitions, and its successor differences set the timing of token commitment, a decision that an energy over complete outputs leaves unspecified.

Recent work has begun extending diffusion-LM RL to semi-autoregressive and block-diffusion settings. TraceRL (**?**) develops a trajectory-level RL framework for diffusion language models and demonstrates its applicability to block diffusion models, yielding the TraDo family of reasoning models. Subsequent work studies optimization instability in diffusion-LM RL and explicitly generalizes its analysis and training recipe to block diffusion (Zhong et al., 2026). These methods establish that diffusion-style RL can be transferred to hybrid architectures. TraceRL in particular pairs its trajectory-level policy gradient with a diffusion-based value model that lowers the variance of terminal-reward credit at intermediate steps. ReLPO assigns its potential a different role. The potential is the value of a KL-regularized control problem on the hybrid state, its exponential tilt of the reference kernel is the policy target itself, and one normalizer couples the refinement drift with the resolve-or-defer decision. This coupling has no counterpart in trajectory-level baselines, and the factorized control of Table 4 measures its contribution.

B TRAINING DETAILS

RELPO acts only on the RL post-training stage. If not specified, the backbone is used as a decoder-only Transformer that represents a partially resolved state as the tuple $s_k = (c_k, y_{c_k}, z_{\bar{c}_k}, \rho_{\bar{c}_k})$ of Section 2.1, namely a resolution mask, the committed token values, the latent semantic variables at unresolved positions, and their progression coordinates. Latent refinement is carried out by the shared field F_0 under full-sequence self-attention, so unresolved positions attend to committed ones and to each other at every stage. Tokenization uses the 64K SentencePiece BPE vocabulary and a 4K context window. The same checkpoint initializes the trainable generator π_θ, and a frozen copy provides the reference policy π_0 that defines the RLOV in Eq. (3), supplies the transition kernel $\mathcal{K}_0$ in Eq. (4), and anchors the regularizer in Eq. (10).

The RL prompt pool contains roughly 58K verifiable problems, drawn from mathematics and code. The mathematics portion draws on the GSM8K and MATH training splits together with approximately 20K competition-style problems whose final answers are numeric or symbolic and can be checked automatically. The code portion draws on the MBPP training split, approximately 18K function-level tasks from permissively licensed repositories that ship executable unit tests, and approximately 4K repository-level tasks with test harnesses in the style of the SWE benchmarks.

The source composition comprises 35K mathematics prompts from GSM8K, MATH, and historical competitions, and 23K code prompts from MBPP and repository-derived tasks. Screening normalizes whitespace and mathematical formatting, removes exact duplicates, and reviews candidate overlaps identified by token 5-gram Jaccard similarity of at least 0.8 using MinHash. Evaluation prompts, AIME 2024 and 2025 problems, and evaluation repository, issue, and commit identifiers are excluded before difficulty filtering.

Rewards are binary and terminal. A mathematics rollout receives $R = 1$ when the extracted final answer is symbolically equivalent to the reference and 0 otherwise, a code rollout receives $R = 1$ when the generated program passes every hidden test inside a sandbox under a fixed time limit and 0 otherwise. We use no partial credit, no format or length shaping, and no step-level annotation for RELPO. The MC success-critic comparison is described in Appendix D.4. Before training we score every prompt with eight rollouts from the reference policy and discard prompts solved eight times out of eight or zero times out of eight, since neither case produces a usable spread of group-relative advantages A_g. This filter retains approximately 41K prompts. The Evo reference defines the fixed pool used across architectures, so Table 2 measures within-backbone gains under a shared data distribution. Weaker backbones can have fewer groups with both successful and unsuccessful completions.

Rollouts are collected in groups of $G = 8$ per prompt at temperature 1.0 with no nucleus truncation, so that the group statistics μ_R and σ_R are computed under the sampling distribution they are meant to describe. Each rollout runs the hybrid generator for $K = 20$ stages, matching the maximum refinement depth used during pretraining, and generates at most 2048 new tokens. One RL iteration draws 256 prompts, hence 2048 rollouts and approximately 4.0M generated tokens, and we run 500 iterations for a total of roughly 2.0B generated tokens per run. This total rollout-token budget is held fixed across every method in Table 1, the rollout-budget ablation in Figure 1 redistributes that budget across group sizes and can change the number of distinct prompts encountered.

Fitting the potential requires states drawn from the interior of the hierarchy. From each rollout we cache six of the $K + 1$ visited states, sampled uniformly over the stage index so that early, mixed, and nearly resolved states are represented in proportion. For every cached state we additionally draw four successors from the frozen reference kernel $\mathcal{K}_0$, which supply the inner expectation of Eq. (5). Drawing successors from the frozen reference keeps the regression target from drifting as π_θ moves, and costs one transition per successor. Each iteration therefore contributes roughly 12K Bellman targets and 2048 terminal targets for $\mathcal{L}_{\text{term}}$.

The resolution objective of Eq. (9) uses the same successors. For M successors $s'_m \sim \mathcal{K}_0(\cdot \mid s_k)$, let b_m denote the sampled resolve action or defer event, and set

$$\widehat{w}_m = \frac{\exp(\Phi_\phi(s'_m, k+1)/\beta)}{\sum_{j=1}^{M} \exp(\Phi_\phi(s'_j, k+1)/\beta)}, \qquad \widehat{\ell}_{\text{resolve}} = -\sum_{m=1}^{M} \widehat{w}_m \log \lambda_\theta(b_m \mid s_k, k).$$

This self-normalized importance estimate is consistent as M grows and has finite-sample bias. Refinement samples contribute to the defer mass. Target construction requires M successor-potential evaluations per state, which are batched, while the policy retains its native normalized output head over positions and tokens. The weights are detached during policy updates.

The potential Φ_ϕ is a value head over a LoRA-adapted copy of the backbone trunk with rank 32. The head is a two-layer MLP of width 2048 with GELU activations, applied to the pooled hidden representation of the hybrid state concatenated with a sinusoidal embedding of the stage index k and the scalar resolution ratio $r(s) = \|c\|_1/N$. Such a Φ_ϕ consumes the latent variables $z_{\bar{c}_k}$ directly and differentiably, so the gradient $\nabla_{z_{\bar{c}_k}} \Phi_\phi$ required by Eq. (8) is defined. In addition, the adapter is separate from the generator's own weights, so fitting the potential does not implicitly update the policy outside the distillation losses. The target network $\Phi_{\bar{\phi}}$ in Eq. (5) is an exponential moving average of Φ_ϕ with decay 0.995, refreshed every optimizer step. We set the temperature $\beta = 0.5$, selected on the development split from $\{0.1, 0.25, 0.5, 1.0\}$, and $\epsilon = 10^{-6}$ in the group-relative advantage. The metric $A_k(s_k)$ is taken to be the diagonal covariance of the pretrained refinement schedule at stage k, so the Mahalanobis norm in Eq. (8) reduces to a per-stage rescaled mean-squared error and introduces no additional parameters.

The composite objective $\mathcal{L}_{\mathrm{ReLPO}}$ uses $\lambda_{\mathrm{term}} = 1.0$, $\lambda_{\mathrm{flow}} = 1.0$, $\lambda_{\mathrm{resolve}} = 0.5$, $\lambda_{\mathrm{ref}} = 0.1$, and $\gamma = 0.05$ inside Eq. (10). Weights were chosen so that the four terms contribute gradients of comparable magnitude after the first few iterations. The two distillation terms are disabled for the first 50 iterations, during which only $\mathcal{L}_{\mathrm{RLOV}}$ and $\mathcal{L}_{\mathrm{term}}$ are active. After warmup all five terms remain active with constant weights for the rest of training.

Optimization uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$. The generator is trained at a peak learning rate of 1×10^{-6} with no weight decay, and the potential at 1×10^{-5} with weight decay 0.01, both follow a cosine schedule with 20 iterations of linear warmup. Gradients are clipped to a global norm of 1.0. Training runs in bfloat16 mixed precision under DeepSpeed ZeRO-3 with sequence parallelism, on 64 NVIDIA A100 80GB GPUs partitioned into 32 devices for rollout generation and 32 for optimization, with the reference policy resident on the generation devices. A complete run takes approximately 5.6 days of wall-clock time. Every number reported for ReLPO and for the baselines is the mean over three RL seeds, which differ in rollout sampling, prompt ordering, and potential-head initialization.

Baselines are matched as tightly as the methods allow. Each starts from the same supervised checkpoint, consumes the same filtered prompt pool, is scored by the same verifier, and is held to the same total rollout-token budget and the same decoding configuration at evaluation. For each baseline we sweep the learning rate over $\{5 \times 10^{-7}, 1 \times 10^{-6}, 2 \times 10^{-6}\}$ and its principal method-specific coefficient over three values on the development split, then train the selected configuration with three seeds. The development search for ReLPO covers the same three generator learning rates and the temperature β over four values, with the loss weights fixed once by gradient balancing before the search, so the two searches are of similar size. All methods have access to the same refinement and resolve/defer components, with both stages eligible for optimization. Each loss is applied wherever its formulation defines an update. The joint flat outcome-RL control applies terminal group-relative advantages to both components without a learned potential and uses the same tuning protocol.

The mixed policy uses a categorical resolve/defer decision with targets normalized under the full transition kernel. On Evo, a refinement transition carries the Gaussian density of the latent update under the model's refinement field with covariance A_k, and a resolution transition carries the categorical probability of the resolve/defer decision and of the committed token. Table 6 specifies, for each baseline, the unit its objective acts on, the transitions it updates directly, its likelihood surrogate, and the coefficient swept here. Remaining hyperparameters follow each method's released configuration. Both refinement and resolve/defer control are available to every method. Transition coverage follows the objective, with no branch-freezing restriction imposed by the comparison. Native masked-diffusion runs distill the tilted denoising kernel through weighted denoising likelihood.

The cross-architecture experiment in Table 2 keeps the RL data, verifier, budget, and evaluation protocol fixed while changing the backbone. On a purely autoregressive backbone there is no revisable latent content, so $A_k \equiv 0$, Eq. (8) is inactive, and ReLPO reduces to the resolution update of Eq. (9). For LLaDA and MDLM, the state comprises committed tokens and unresolved masks, and the native denoising kernel includes discrete unmasking transitions. The RLOV and its two

Table 6: Adaptation of the baselines to the Evo hybrid rollout. "Refine." marks a direct objective on latent refinement transitions, "Token" on the committed token, and "Timing" on the resolve/defer decision. The swept coefficient is searched over three values together with the learning rate, and other settings follow each method's released configuration.

Method	Update unit on Evo	Refine.	Token	Timing	Likelihood or objective	Swept coefficient
GRPO	Full hybrid trace	✓	✓	✓	Clipped ratio of the trace likelihood with group-relative advantage and KL to reference	KL coefficient
J-GRPO	Hybrid trace with left-to-right commitment order	✓	✓	✓	Clipped ratio of the exact trace likelihood at fixed order	KL coefficient
ESPO	Full hybrid trace	✓	✓	✓	Sequence-level likelihood ratio with sequence-level clipping	Clipping range
d1	Refinement steps and committed tokens	✓	✓	–	Mean-field token log-probability from one-step prediction with group-relative advantage	Prompt-masking rate
WD1	Refinement steps and committed tokens	✓	✓	–	Advantage-weighted transition log-likelihood without ratios	Weight temperature
StableDRL	Refinement steps and committed tokens	✓	✓	–	Trajectory objective with the original stabilization terms	Clipping range
SPG	Refinement steps and committed tokens	✓	✓	–	Sign-aware objective from upper and lower likelihood bounds	Bound mixing coefficient
TraDo	Full hybrid trace	✓	✓	✓	Trajectory-level ratio with a learned intermediate value baseline	Value-loss coefficient
Flat outcome RL	Full hybrid trace	✓	✓	✓	Terminal group-relative advantage on every transition	KL coefficient
RELPO	Cached states and reference successors	✓	✓	✓	Eqs. (5) and (8)–(10)	Temperature β

losses apply to this discrete kernel. The policy target is its normalized potential tilt, distilled through the model-compatible denoising objective. The continuous-gradient update of Eq. (8) applies to backbones with continuous unresolved variables. The policy implementation and baseline objective coverage follow the common setup above.

At evaluation, every benchmark uses greedy decoding at temperature 0, one sample per problem, at most 2048 generated tokens, and $K = 20$ stages, matching training. Code benchmarks report pass@1 under the benchmark tests, and mathematics benchmarks report final-answer accuracy under the same symbolic checker used during training, with the answer extracted from the final boxed expression. GSM8K, MATH, AIME 2024, and GPQA use zero-shot prompts with a fixed instruction to box the final answer, and GPQA is scored by the selected option letter. HumanEval and MBPP use the standard function-completion prompts and the benchmark unit tests. SWE-Bench Verified and SWE-Context Bench are run as single-pass patch generation. The prompt contains the issue text followed by the code context that the benchmark supplies for the task, truncated to fit the 4K window, and for SWE-Context Bench this context includes the benchmark's record of relevant experience from related tasks in the same repository. The model emits one unified diff, which is applied to the repository at the specified commit and scored by the benchmark tests, and no repository navigation, tool call, or test feedback is available during generation. This protocol measures patch generation from provided context and yields lower absolute scores than agentic harnesses. End-to-end latency is measured single-request on one NVIDIA A100 including prompt encoding, and the "Steps/tok." column reports the mean number of latent refinement updates per generated token, averaged over the full evaluation suite.

The analyses in Table 5 and Table 12 use a held-out cache of 4000 hybrid states collected from rollouts on development prompts, stratified into four bins by the resolution ratio $r(s)$. For each cached state we draw $M = 32$ independent continuations under the reference kernel to form the Monte-Carlo target $\widehat{\Phi}_{\mathrm{MC}}$, and report Spearman correlation and mean absolute error against Φ_ϕ within each stratum. Decision sign accuracy is computed over 2000 resolve-versus-defer pairs, at a cached state we either commit the position with the highest reference resolution probability or defer it for eight further refinement steps, estimate both branches with 16 continuations each, and check whether the sign of the predicted potential difference agrees with the sign of the empirical outcome difference. For the behavioral columns we call a resolution premature when committing lowers the Monte-Carlo estimate by more than $\tau = 0.05$ relative to deferring, and a refinement step *redundant* when eight additional steps change the estimate by less than τ. The same threshold is applied to every method.

C Proofs of Theoretical Results

This appendix proves the results stated in Section 2. Throughout, $\mathcal{V}$ is finite, responses use the fixed-length EOS and padding representation of Section 2.1, and $R_{\max} = \sup_{x,y}|R(x,y)| < \infty$. States, completion fibers, and the resolution operator $J_{i,a}$ are as defined in Section 2.1.

C.1 Native Relaxation Hierarchy

Assume that realization preserves committed tokens and latent refinement leaves the resolution mask and committed values unchanged. Then the fibers in Eq. (1) satisfy Eq. (2). Refinement preserves the fiber, and realization contracts it.

Proof. Fix a rollout $s_0 \to \cdots \to s_K \to y$. During refinement, $c_{k+1} = c_k$ and $y_{c_{k+1}} = y_{c_k}$. The projection and its specified values in Eq. (1) are unchanged, so

$$\mathcal{F}(s_{k+1}) = \mathcal{F}(s_k). \tag{11}$$

The continuation probabilities may change as the unresolved variables evolve.

Resolving position $i \in \bar{c}_k$ as token a adds the constraint $y'_i = a$, giving

$$\mathcal{F}(J_{i,a}(s_k)) = \mathcal{F}(s_k) \cap \{y' \in \mathcal{V}^N \mid y'_i = a\} \subseteq \mathcal{F}(s_k).$$

For $|\mathcal{V}| > 1$, this inclusion is strict whenever a new position is resolved. Irreversibility preserves each added constraint at subsequent stages. Induction gives Eq. (2), and $c_K = \mathbf{1}$ yields $\mathcal{F}(s_K) = \{y\}$.

Set inclusion partially orders the completion fibers. Pure autoregressive decoding contracts the fiber at every token commitment. Continuous latent diffusion with terminal decoding keeps $\mathcal{F}(s_k) = \mathcal{V}^N$ before the terminal readout, while its continuation distribution evolves. Masked diffusion contracts the fiber whenever tokens are irreversibly unmasked. These conclusions follow directly from the resolution mask and committed values.

C.2 Fiber Aggregation Smooths Sparse Outcome Optimization

Suppose that $R(x,y)$ is bounded and that $p_{\pi_0}(y \mid s)$ varies continuously with respect to the unresolved latent variables $z_{\bar{c}}$. Then, for every non-terminal state s, the reward-weighted mass $\mathcal{Z}_\beta(s)$ is a continuous functional of the latent state, whereas at a fully resolved state $s = y$ it degenerates to $\exp(R(x,y)/\beta)$.

Proof. Fix the resolved part (c, y_c) and regard the reward-weighted mass $\mathcal{Z}_\beta$ as a function of the unresolved coordinates alone. Because $\mathcal{V}$ and N are finite, $\mathcal{F}(s)$ is a fixed finite set at fixed (c, y_c) and $\mathcal{Z}_\beta$ is the finite sum

$$\mathcal{Z}_\beta(s) = \sum_{y \in \mathcal{F}(s)} \exp\left(\frac{R(x,y)}{\beta}\right) p_{\pi_0}(y \mid s). \tag{12}$$

The coefficients $\exp(R(x,y)/\beta)$ do not depend on $z_{\bar{c}}$ and lie in $[e^{-R_{\max}/\beta}, e^{R_{\max}/\beta}]$. Since $p_{\pi_0}(\cdot \mid s)$ is a probability measure on the fiber,

$$0 < e^{-R_{\max}/\beta} \;\le\; \mathcal{Z}_\beta(s) \;\le\; e^{R_{\max}/\beta} < \infty. \tag{13}$$

Each map $z_{\bar{c}} \mapsto p_{\pi_0}(y \mid s)$ is continuous by hypothesis, and a finite sum of continuous functions with constant coefficients is continuous, so $\mathcal{Z}_\beta$ is continuous in $z_{\bar{c}}$. When the fiber is described by a continuous measure the same conclusion follows from dominated convergence, with the constant $e^{R_{\max}/\beta}$ as dominating function.

At a fully resolved state $c = \mathbf{1}$ the fiber is the singleton $\{y\}$ by the hierarchy result of Appendix C.1, so $p_{\pi_0}(\cdot \mid s) = \delta_y$ and Eq. (12) collapses to $\mathcal{Z}_\beta(y) = \exp(R(x, y)/\beta)$.

First, Eq. (13) bounds the mass away from zero, so $\Phi = \beta \log \mathcal{Z}_\beta$ is well defined and inherits continuity. This is the RLOV of Definition 1. Second, wherever p_{π_0} is differentiable in $z_{\bar{c}}$, differentiating Eq. (12) gives

$$\nabla_{z_{\bar{c}}} \Phi(s) = \beta \, \mathbb{E}_{y \sim \mu_\beta(\cdot|s)} \left[\nabla_{z_{\bar{c}}} \log p_{\pi_0}(y \mid s) \right], \qquad \mu_\beta(y \mid s) \propto \exp\left(\frac{R(x, y)}{\beta} \right) p_{\pi_0}(y \mid s). \quad (14)$$

The gradient therefore exists and is a reward-tilted expectation of the latent score, bounded whenever that score is bounded. It is the object that Eq. (6) uses to steer refinement.

The two displays exhibit the announced interpolation. At a broadly unresolved state the sum in Eq. (12) runs over the token-consistent fiber and varies smoothly with $z_{\bar{c}}$ at a rate controlled by Eq. (14), as positions are committed, terms leave the sum until a single term survives and the aggregate degenerates to the sparse verifier reward $\exp(R(x, y)/\beta)$.

C.3 Resolution-Lifted Outcome Consistency (Theorem 1)

Let s_k be a partially resolved hybrid state and assume that $R(x, y)$ is bounded. Then the RLOV satisfies the soft recursion of Eq. (4) with terminal condition $\Phi^\star_K(y) = R(x, y)$, where $\mathcal{K}_0$ is the one-step transition kernel of the reference hybrid generator.

Proof. Write $Z_k(s) = \mathbb{E}_{Y \sim \pi_0(\cdot|s)}[\exp(R(x, Y)/\beta)]$, so that $\Phi^\star_k(s) = \beta \log Z_k(s)$ by Eq. (3). Equation (13) gives $0 < Z_k < \infty$, so the logarithm is well defined and every manipulation below is between finite quantities.

The reference generator is Markov in the hybrid state. Conditioned on s_{k+1}, the law of the completion Y is $p_{\pi_0}(\cdot \mid s_{k+1})$ and does not depend on s_k. The completion law at stage k therefore factorizes as the mixture

$$p_{\pi_0}(dy \mid s_k) = \int p_{\pi_0}(dy \mid s_{k+1}) \, \mathcal{K}_0(ds_{k+1} \mid s_k). \quad (15)$$

Equation (15) requires no distinction between the two transition types. Refinement moves and resolution jumps are both realizations of $\mathcal{K}_0$, and the identity holds for any mixture of them. This is the step at which the mixed structure of hybrid generation is absorbed.

Applying the tower property to Eq. (15),

$$Z_k(s_k) = \mathbb{E}_{s_{k+1} \sim \mathcal{K}_0(\cdot|s_k)} \left[\mathbb{E}_{Y \sim \pi_0(\cdot|s_{k+1})} \left[\exp\left(\frac{R(x, Y)}{\beta} \right) \right] \right] = \mathbb{E}_{s_{k+1} \sim \mathcal{K}_0(\cdot|s_k)} \left[Z_{k+1}(s_{k+1}) \right]. \quad (16)$$

By definition $Z_{k+1}(s_{k+1}) = \exp(\Phi^\star_{k+1}(s_{k+1})/\beta)$. Substituting this into Eq. (16) and applying $\beta \log(\cdot)$ to both sides yields Eq. (4).

For the boundary condition, a stage-K state is fully resolved, so by Appendix C.1 its fiber is $\{y\}$. Hence $Z_K(y) = \exp(R(x, y)/\beta)$ and $\Phi^\star_K(y) = R(x, y)$, which is the stated boundary condition.

The recursion is exact and no independence between refinement and resolution is assumed. The mixture in Eq. (15) passes through the logarithm as a log-sum-exp, whereas replacing $\beta \log \mathbb{E} \exp(\cdot/\beta)$ by $\mathbb{E}[\cdot]$ would break Eq. (16) unless $\Phi^\star_{k+1}$ were constant on the support of $\mathcal{K}_0(\cdot \mid s_k)$.

C.4 Outcome-Only Identification (Proposition 1)

Assume a Markovian reference kernel, a realizable potential class, unbiased terminal rewards, and reference occupancies absolutely continuous with respect to behavior occupancies at every stage. Then a zero-residual fixed point of the population objectives recovers Eq. (3) reference-almost everywhere on the covered reachable set.

Proof. Define the soft backup operator $\mathcal{T}$ acting on bounded functions of a state and stage index,

$$(\mathcal{T}\Phi)(s,k) = \beta \log \mathbb{E}_{s' \sim \mathcal{K}_0(\cdot|s)} \left[\exp\left(\frac{\Phi(s', k+1)}{\beta} \right) \right]. \tag{17}$$

Step 1. $\mathcal{T}$ is a sup-norm non-expansion. Let Φ_1, Φ_2 be bounded and set $c = \|\Phi_1 - \Phi_2\|_\infty$, so $\Phi_1 \le \Phi_2 + c$ pointwise. Monotonicity of the exponential and of expectation gives $\mathbb{E}[\exp(\Phi_1/\beta)] \le e^{c/\beta}\, \mathbb{E}[\exp(\Phi_2/\beta)]$, hence $\mathcal{T}\Phi_1 \le \mathcal{T}\Phi_2 + c$. Exchanging the roles of Φ_1 and Φ_2 yields

$$\|\mathcal{T}\Phi_1 - \mathcal{T}\Phi_2\|_\infty \le \|\Phi_1 - \Phi_2\|_\infty. \tag{18}$$

Step 2. uniqueness on the reachable set. The horizon is finite, with stages $k = 0, \dots, K$. The terminal loss $\mathcal{L}_{\mathrm{term}}$ pins the stage-K layer to the terminal reward, and by Eq. (17) the stage-k layer is determined by the stage-$(k+1)$ layer. Backward induction on k therefore determines a unique function reference-almost everywhere on the covered reachable set. Strict contraction is not required for this argument, and Eq. (18) serves only to show that the induction is stable under perturbations of the terminal layer.

Step 3. zeros of the population objectives. Both losses are expectations of squares and hence non-negative. Because R is bounded, the advantages A_g are bounded and the tilted weights w_g satisfy $w_g > 0$ for every rollout in a group, the weighting thus reweights visited states without removing any from the support. The coverage assumption $d_k^0 \ll d_k^{\mathrm{old}}$ transfers behavior-almost-everywhere identities to the reference occupancies at every stage, including successor and terminal states. Consequently $\mathcal{L}_{\mathrm{RLOV}}$ vanishes if and only if

$$\Phi_\phi(s,k) = (\mathcal{T}\Phi_{\bar{\phi}})(s,k) \tag{19}$$

behavior-almost everywhere in (s,k), and $\mathcal{L}_{\mathrm{term}}$ vanishes if and only if $\Phi_\phi(y,K) = R(x,y)$. At a fixed point of the exponential moving average the target parameters coincide with the online parameters, $\Phi_{\bar{\phi}} = \Phi_\phi$, so Eq. (19) becomes $\Phi_\phi = \mathcal{T}\Phi_\phi$.

Step 4. identification. By Theorem 1, the RLOV satisfies $\Phi^\star = \mathcal{T}\Phi^\star$ together with the terminal condition $\Phi^\star_K(y) = R(x,y)$. By Step 2 that pair of conditions admits a unique solution on the reachable set, and by realizability this solution lies in the function class of Φ_ϕ. Any minimizer driving both population losses to zero therefore satisfies $\Phi_\phi = \Phi^\star$ reference-almost everywhere on the covered reachable set, which is Eq. (3).

C.5 Coupled Hybrid Policy Improvement

Suppose that both latent refinement and autoregressive realization are available at state s_k, and that $\Phi^\star_k$ is exact. Then the exact tilted kernel is the KL-regularized policy improvement operator associated with $R(x,y)$. Equation (7) describes its resolution masses and Eq. (6) describes its refinement drift in the small-step limit. Both branch updates depend on their relative partition values under the shared successor potential.

Proof. *Part (i). the joint update is the KL-regularized improvement step.* Consider the one-step control problem at s_k over successor laws q,

$$\max_q \ \left\{ \mathbb{E}_{s' \sim q}\left[\Phi^\star_{k+1}(s')\right] - \beta\, D_{\mathrm{KL}}(q \parallel \mathcal{K}_0(\cdot \mid s_k)) \right\}. \tag{20}$$

The Gibbs variational principle gives the unique maximizer

$$q^\star(ds' \mid s_k) = \frac{\mathcal{K}_0(ds' \mid s_k) \exp\big(\Phi^\star_{k+1}(s')/\beta\big)}{\mathbb{E}_{\mathcal{K}_0}\left[\exp\big(\Phi^\star_{k+1}/\beta\big)\right]}, \tag{21}$$

with optimal value $\beta \log \mathbb{E}_{\mathcal{K}_0}[\exp(\Phi^\star_{k+1}/\beta)]$, which equals $\Phi^\star_k(s_k)$ by Theorem 1. Equation (21) is therefore a policy improvement operator for the terminal reward, regularized toward the reference generator, and the RLOV is its value.

To obtain the refinement update, consider a locally Gaussian reference step with mean $\mu = z_{\bar{c}_k} + F_0(s_k,k)\Delta t$ and covariance $A_k \Delta t$. A first-order expansion of the successor potential at μ gives

$$\Phi^\star_{k+1}(z') \simeq \Phi^\star_{k+1}(\mu) + \nabla \Phi^\star_{k+1}(\mu)^\top (z' - \mu). \tag{22}$$

Completing the square in Eq. (21) shifts the conditional mean by $A_k \Delta t \nabla \Phi^\star_{k+1}(\mu)/\beta$. Under smoothness and the small-step limit, this yields Eq. (6). At finite step sizes, higher-order potential terms can also change the covariance and shape of the tilted distribution. Equation (21) remains exact for the discrete kernel, while Eq. (6) is its local approximation. The correction acts on unresolved coordinates. For a local quadratic potential with gradient g, Hessian H, and reference covariance $\Sigma = A_k \Delta t$, completing the square gives

$$\Sigma_q = (\Sigma^{-1} - H/\beta)^{-1}, \qquad \mu_q = \mu + \Sigma_q g/\beta.$$

These expressions apply on the nondegenerate unresolved subspace when the precision is positive definite. If $\eta = \|\Sigma^{1/2} H \Sigma^{1/2}/\beta\|_2 < 1$, then $\|\Sigma^{-1/2}\Sigma_q \Sigma^{-1/2} - I\|_2 \le \eta/(1-\eta)$. This bound quantifies the covariance correction in the quadratic model and explains the small-step requirement for the first-order drift.

On the resolution branch the successors are the atoms $J_{i,a}(s_k)$ carrying reference probabilities $\lambda_0(i, a \mid s_k, k)$. Equation (21) multiplies each atom by $\exp(\Phi^\star_{k+1}(J_{i,a}(s_k))/\beta)$ and divides by the common normalizer, which Theorem 1 identifies as $\exp(\Phi^\star_k(s_k)/\beta)$. This gives

$$\lambda^\star(i, a \mid s_k, k) = \lambda_0(i, a \mid s_k, k) \exp\left(\frac{\Phi^\star_{k+1}(J_{i,a}(s_k)) - \Phi^\star_k(s_k)}{\beta} \right), \tag{23}$$

which is Eq. (7). Summing over a and renormalizing recovers the total resolution probability $\Lambda^\star$ and the conditional token distribution $\pi^\star$. The remaining mass is the probability of deferral and refinement. This establishes the exact joint kernel and its local refinement realization.

Part (ii). shared branch normalization. Write $\mathcal{K}_0 = \sum_b \alpha_b \mathcal{K}_{0,b}$, where b indexes refinement and resolution and $\sum_b \alpha_b = 1$. Define $Z_b = \mathbb{E}_{\mathcal{K}_{0,b}}[e^{\Phi^\star_{k+1}/\beta}]$. The tilted branch probability is

$$q^\star(b \mid s_k) = \frac{\alpha_b Z_b}{\sum_c \alpha_c Z_c}.$$

Independent within-branch normalization determines the conditional successor laws, while selection between branches also requires their relative partition values. Shared normalization supplies both. Separate estimators can reproduce the joint kernel when their partition values are calibrated and combined through this expression. Learning separate branch-conditioned backups generally changes those values, which motivates the control in Table 4.

Part (iii). limiting regimes. If $\lambda_0 \equiv 0$ the resolution branch is empty, Eq. (7) is vacuous, and Eq. (21) reduces to the h-transform of the reference refinement process, which is outcome-guided diffusion. If $A_k \equiv 0$ there is no revisable latent content, Eq. (6) is vacuous, and Eq. (21) reduces to the familiar KL-regularized softmax improvement over discrete continuations, with $\Phi^\star$ in the role of the soft state value. Both degenerate cases are trivially decomposable because only one branch survives, the shared branch normalization in part (ii) applies when both branches carry positive mass.

D Additional Experiments

The experiments in Section 3.1 establish the effectiveness of ReLPO over flat outcome-RL objectives. This appendix examines several theoretical and practical properties of the framework, including the shared normalization of the coupled improvement operator in Appendix C.5, the role of the temperature β in controlling the interpolation characterized in Appendix C.2, and the effectiveness of terminal reward relative to additional continuation supervision. We report controls addressing these aspects, together with a compute accounting that further controls for rollout cost in the budget comparison of Figure 1.

D.1 Controlled Objectives and Evaluation Uncertainty

A conventional success value provides a direct control for the RLOV objective. For binary rewards, let $V^\star(s) = \Pr_{\pi_0}(R = 1 \mid s)$. Then

$$\Phi^\star(s) = \beta \log\Big[1 + (e^{1/\beta} - 1) V^\star(s)\Big],$$

Table 7: Objective and sampling controls on Evo. Overall averages use the eight benchmarks of Table 1. The first block matches total training FLOPs. The second holds the rollout-token budget fixed and varies one fitting choice.

Method or control	Math Avg.	Code Avg.	Overall Avg.
Joint actor–critic	60.6	55.4	58.0
TD success-value tilt	61.4	56.0	58.7
ReLPO	62.7	56.7	59.7
Uniform state-fitting weights	62.0	56.2	59.1
Raw-reward fitting weights	62.4	56.4	59.4
One reference successor	61.5	55.9	58.7
Sixteen reference successors	62.9	56.9	59.9
Retain zero-of-eight prompts	61.3	55.9	58.6

Table 8: Dated evaluation on the shared Evo backbone. Scores are percentages averaged over three training seeds.

Method	AIME 2025	LiveCodeBench Jan.–Jun. 2025
SFT	13.3	28.7
GRPO	20.0	31.6
ReLPO	26.7	34.8

so a calibrated success critic recovers the same population tilt through this transformation, and the comparison isolates the finite-sample fitting objective and the policy update. The TD success-value control uses the same rank-32 adapter, value-head width, cached states, four reference successors, and terminal targets as ReLPO, regresses the arithmetic backup of V, and drives Eqs. (8)–(9) through the transformed value. The joint actor–critic uses the same critic capacity and full hybrid state, fits on-policy returns, and applies clipped PPO to both Gaussian refinement actions and categorical resolve/defer actions. Both controls use the nine-configuration development search, three seeds, and the total training-FLOP budget of ReLPO.

Table 7 separates value parameterization, policy optimization, state weighting, and successor sampling. The success-value critic and the joint actor–critic trail ReLPO by 1.0 and 1.7 points at matched training FLOPs, so the soft backup and the tilted-kernel distillation contribute beyond the information carried by a success probability. Uniform weights use $w_g = 1/G$, while raw-reward weights use $w_g \propto e^{R_g/\beta}$. Both retain the fixed-temperature soft backup and lose 0.6 and 0.3 points. Successor sampling matters at the low end, with one successor costing 1.0 point and sixteen adding 0.2 over the default four. The final row retains prompts with zero successes in the eight-rollout reference screen and continues to exclude prompts solved eight times. It reaches 58.6, which is 1.1 points below the filtered pool and 2.2 points above the strongest baseline in Table 1. A zero-of-eight observation is compatible with a small positive success probability, and the reference-supported formulation raises that probability once a successful continuation is sampled.

The seed statistics in Figure 2 measure variation across training runs on a fixed test set. Using the three-seed standard deviations, a Student-t interval for the overall mean is $\bar{x} \pm 4.303s/\sqrt{3}$, giving $[58.9, 60.5]$ for ReLPO and $[55.2, 57.6]$ for TraDo. These intervals assume approximately normal independent seed means and exclude test-item uncertainty. For scale, 9 correct answers among 30 AIME problems has a 95% Wilson interval of $[16.7, 47.9]$ percentage points, while 6 correct answers has $[9.5, 37.3]$. A paired comparison additionally requires item-level predictions. The broad benchmark gains and the three-seed variation provide complementary evidence, with the small AIME test set carrying greater uncertainty.

For a dated evaluation, we exclude post-training sources dated after December 2024 and evaluate AIME 2025 and LiveCodeBench (LiveCodeBench Team, 2025) problems released from January through June 2025. We remove exact and normalized-text matches to these evaluation prompts from the RL pool and use the same decoding configuration as Table 1. A second mathematics checker verifies extracted answers independently, and code is evaluated against the benchmark's held-out tests. Table 8 shows the same ordering of methods as Table 1 on both dated sets.

Table 9: Temperature sweep on Evo. "ρ at $r < .25$" is the Spearman correlation between Φ_ϕ and the Monte-Carlo target on the least resolved stratum, a regime with noisy terminal credit assignment. Seed std is over three RL seeds.

β	Math Avg.	Code Avg.	Overall Avg.	ρ at $r < .25$	Steps/ tok.	Seed std
0.10	58.2	53.4	55.8	0.43	2.42	1.0
0.25	61.0	55.6	58.3	0.57	2.55	0.6
0.50	**62.7**	**56.7**	**59.7**	**0.64**	**2.67**	**0.3**
1.00	61.4	56.0	58.7	0.60	2.98	0.5
2.00	58.8	54.0	56.4	0.46	3.41	0.7

D.2 Shared and Branch-Conditioned Potentials

Table 2 compares performance across architectures, where differences in generator type may also reflect differences in the underlying generator quality. To examine the shared-value construction of Appendix C.5, we therefore hold the backbone, parameter count, and training budget fixed, and vary only the structure of the potential. The key comparison uses two separate potentials, Φ^d, estimated exclusively from refinement transitions, and Φ^a, estimated exclusively from resolution transitions. Each potential independently drives its corresponding component of the update through Eqs. (8) and (9). This decoupled variant retains the same model capacity and applies tilting to both branches, but replaces the shared normalizer provided by Theorem 1 with branch-specific potentials. We also evaluate the two limiting regimes on the same hybrid generator, which are a refinement-only tilt, which keeps the realization probability fixed at λ_0, and a resolution-only tilt, which keeps the latent field fixed at F_0.

For branch $b \in \{d, a\}$, the factorized potential uses the conditional reference kernel $\mathcal{K}_{0,b}$ and the target $\mathcal{T}_b\Phi^b(s,k) = \beta \log \mathbb{E}_{s' \sim \mathcal{K}_{0,b}(\cdot|s)}[e^{\Phi^b(s',k+1)/\beta}]$. Both potentials use the terminal reward boundary. If a branch has zero probability at a state, its backup uses the full reference kernel.

In Table 4, the factorized variant achieves 57.7, 2.0 points below RELPO (59.7), under the matched rollout budget and parameter count. The branch-local potentials also correlate less with the Monte-Carlo RLOV target ($\rho = 0.63$ versus 0.81), consistent with neither satisfying the recursion in Eq. (4) on its own. A potential estimated solely from refinement transitions never observes the outcome mass reached through resolution transitions and so evaluates the resolution decisions governed by Eq. (7) with a miscalibrated partition value. Shared supervision and target variance contribute to the difference alongside the normalizer itself.

The two single-branch variants degrade further, to 56.7 and 56.0, but exhibit complementary failure modes. The refinement-only tilt shows less effective commitment behavior, with premature resolution increasing to 15.5% and refinement usage rising to 3.48 steps per token. Conversely, the resolution-only tilt commits with fewer refinement steps (2.45, the lowest in the table) but does so over lower-quality latent states, resulting in the lowest accuracy. Together, these results illustrate the practical role of shared branch normalization, showing that improving the resolution policy alone is insufficient when the latent content being resolved has not been correspondingly improved, while improving refinement alone does not ensure that the resulting content is resolved at the appropriate time.

D.3 The Temperature β

The fiber-aggregation argument of Appendix C.2 describes a family of objectives indexed by β, interpolating between a smooth aggregate over broad fibers and the sparse terminal reward. To further examine this interpolation beyond the default setting of $\beta = 0.5$ used in the main experiments, we sweep $\beta \in \{0.1, 0.25, 0.5, 1, 2\}$ while holding all other settings fixed.

As shown in Table 9 and Figure 3, accuracy follows an inverted-U pattern, peaking at $\beta = 0.5$ and approaching the flat outcome-RL baseline toward both ends of the sweep, with the two regimes exhibiting distinct behaviors. As $\beta \to 0$, the soft aggregate in Eq. (3) approaches a maximum over the fiber, shifting the potential toward the best reachable completion with increasing emphasis on

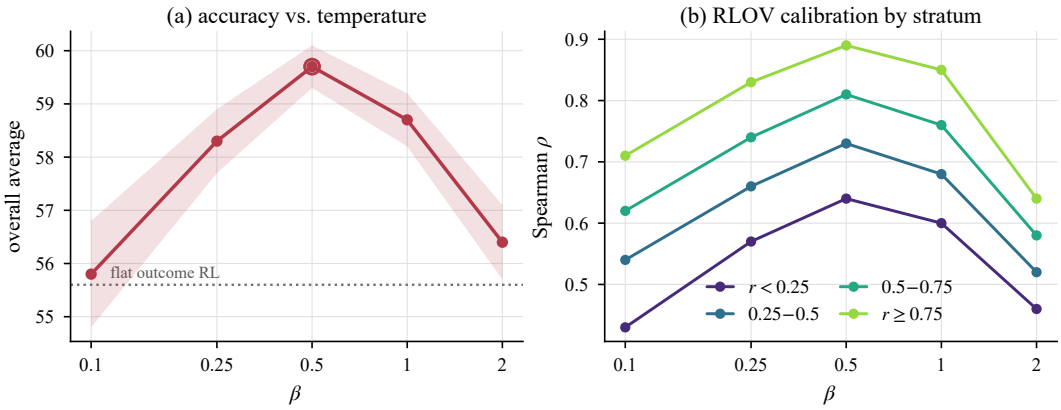


Figure 3: Effect of the temperature β. (a) Overall average against β, with the flat outcome-RL level marked and a band showing the seed standard deviation. (b) RLOV calibration within each resolution stratum. Both panels peak at $\beta = 0.5$, and the two tails degrade for different reasons.

Table 10: Outcome-only learning and additional Monte-Carlo state supervision. The MC-supervised row uses a differentiable potential fitted to high-sample targets.

Method	MC state targets	Additional supervision	Math Avg.	Code Avg.	Overall Avg.	$\rho(\Phi, \widehat{\Phi}_{\mathrm{MC}})$
SFT	–	–	54.3	50.7	52.5	–
Flat outcome RL	–	–	58.0	53.2	55.6	–
MC success critic	✓	40K states, 32 continuations	61.1	55.7	58.4	0.73
ReLPO	–	none	**62.7**	**56.7**	**59.7**	**0.81**
MC-supervised potential (M=256)	–	256 continuations/state	63.6	57.8	60.7	0.99

high-reward continuations. Correspondingly, its Monte-Carlo estimate becomes higher variance, with the seed standard deviation increasing to 1.0 and calibration on the least-resolved stratum decreasing to $\rho = 0.43$. This behavior is consistent with the objective approaching the sparse-reward limit that the lifting formulation is designed to smooth.

As β increases, the tilt $\exp(\Delta\Phi/\beta)$ approaches unity for bounded rewards. Across the finite sweep, refinement usage increases from 2.42 to 3.41 steps per token. The value at $\beta = 2$ exceeds the reference usage of 3.08, reflecting the jointly trained policy and approximation error at finite temperature. The asymptotic tilt alone does not determine this finite-temperature behavior. Calibration also decreases toward both ends of the sweep (Figure 3b), and $\beta = 0.5$ provides the strongest performance and calibration among the evaluated settings.

D.4 Sufficiency of Terminal Reward

To further examine whether a dense learning signal over partially resolved states can be obtained from terminal reward alone, we compare ReLPO with an MC success critic trained on 40K labelled cached states and a differentiable MC-supervised potential using $M = 256$ continuations per state. The latter uses the same potential architecture, with regression to Monte-Carlo soft targets and gradients obtained by differentiating the fitted network. The MC success critic uses the success fraction of 32 reference continuations to label each cached state and fits the value head by binary cross-entropy. The MC-supervised potential regresses to $\beta \log[M^{-1} \sum_m e^{R_m/\beta}]$ on the same cache. Fitting uses a 90/10 prompt-disjoint training and validation split, three epochs, and a learning rate of 10^{-5}.

In Table 10, ReLPO exceeds the MC success critic by 1.3 points and recovers 80% of the observed gap between flat outcome RL and the MC-supervised potential. For binary rewards, writing $p_s = \Pr_{\pi_0}(R = 1 \mid s)$ gives $\Phi^\star(s) = \beta \log[1 + p_s(e^{1/\beta} - 1)]$, a strictly increasing function of p_s. Ideal success-probability and RLOV estimates therefore induce the same ranking. The empirical correlations of 0.73 and 0.81 depend on the supervision targets, learning procedure, and supervision

Table 11: Training cost. "Potential FLOP share" is the fraction of total training FLOPs spent on Φ_ϕ and its successor samples. The last two rows give each baseline the same total budget ReLPO consumes.

Method	Rollout tok. (B)	Successor tok. (B)	Potential FLOP share	Total train FLOPs (rel.)	Wall clock (h)	Overall Avg.
SFT (no RL)	–	–	–	–	–	52.5
GRPO	2.0	0.00	0.0%	1.00×	120	55.9
TraDo	2.0	0.00	0.0%	1.03×	123	56.4
ReLPO	2.0	0.39	6.6%	1.24×	135	**59.7**
GRPO @ ReLPO budget	2.5	0.00	0.0%	1.24×	149	56.3
TraDo @ ReLPO budget	2.5	0.00	0.0%	1.24×	152	57.0

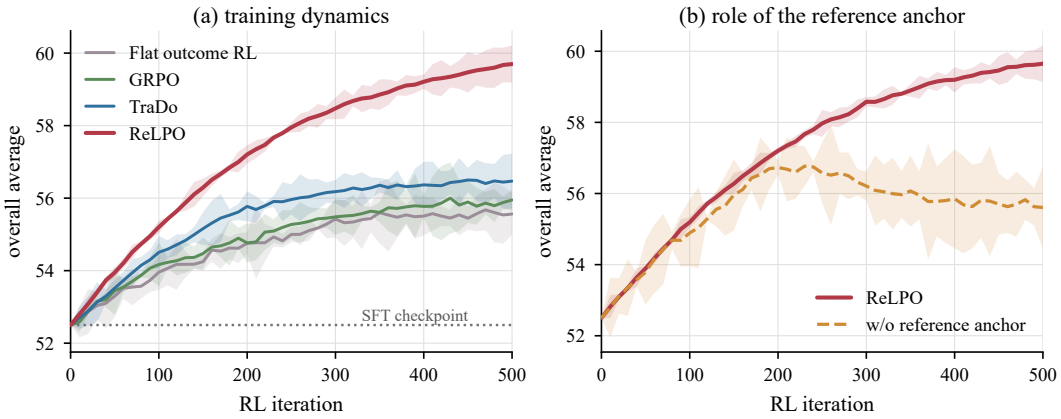


Figure 4: Training dynamics over 500 RL iterations, three seeds, bands are 95% confidence intervals of the seed mean. (a) ReLPO against the outcome-RL baselines. (b) The same ReLPO run against an otherwise identical run with the reference regularizer of Eq. (10) removed.

budget. The high-sample learned potential provides a reference comparison with finite sampling and function-approximation error.

D.5 Compute-Matched Comparison and Training Dynamics

Figure 1 compares methods under a matched generated-rollout-token budget. A fuller accounting of training cost also counts the computation used by ReLPO to sample four successors for each of six cached states per rollout. We therefore report the total training cost for each method and re-run both baselines at ReLPO's full compute budget, which gives a comparison under matched overall training cost.

Table 11 reports separate FLOP and wall-clock costs. ReLPO uses $1.24\times$ the GRPO training FLOPs and $135/120 = 1.125\times$ its wall-clock time, an increase of 15 hours. With 64 allocated GPUs, these totals correspond to 8,640 GPU-hours for ReLPO and 7,680 for GRPO per run, or 25,920 and 23,040 over three seeds. These totals cover the selected training runs and exclude development sweeps. Potential fitting and successor sampling account for $0.066 \times 1.24 = 0.08184$ of the GRPO FLOP budget. The remaining overhead is 0.15816 of that budget, attributable to latent-gradient evaluation and coupled policy distillation. Differences in utilization and scheduling can produce different FLOP and wall-clock ratios. The compute-matched baselines reach 56.3 and 57.0, remaining 3.4 and 2.7 points below ReLPO. For the compute-matched baselines, cosine decay extends from 500 to approximately 625 iterations for 2.5B rollout tokens, retains 20 warmup iterations, and stops at the matched accumulated FLOP budget.

Figure 4 (a) provides a complementary view of the training dynamics. ReLPO begins to separate from the baselines within the first hundred iterations and continues to improve after their performance

Table 12: Behavioral and efficiency analysis of different RL methods. 'Premature resolution" is the percentage of realization events for which committing lowers the Monte-Carlo return estimate by more than $\tau = 0.05$ relative to counterfactual deferral. 'Redundant refinement" is the percentage of refinement events for which eight additional refinement steps change the Monte-Carlo estimate by less than τ. Both are computed using matched continuation rollouts from cached hybrid states, and the same threshold is applied to every method. Lower is better for both quantities.

Method	Math Avg.	Code Avg.	Premature resolution (%) ↓	Redundant refinement (%) ↓	Refinement steps/tok. ↓	E2E latency (s) ↓
SFT	54.3	50.7	18.4	24.0	3.08	4.79
GRPO	58.4	53.4	16.1	21.8	3.39	5.18
TraDo	59.0	53.9	14.6	20.3	3.32	5.07
ReLPO	**62.7**	**56.7**	**8.0**	**11.4**	**2.67**	**4.15**

has largely stabilized. GRPO and flat outcome RL reach 95% of their total improvement by iteration 370, whereas ReLPO reaches the same fraction at iteration 430 and continues to improve toward the end of training. ReLPO also exhibits the lowest per-iteration variability among the compared methods ($\sigma_\Delta = 0.17$, compared with 0.29 for GRPO and 0.33 for flat outcome RL), consistent with the dense lifted signal providing informative updates even when a sampled rollout group contains uniformly sparse terminal rewards.

Figure 4 (b) further examines the role of the reference regularizer. Without the regularizer, performance remains within 0.2 points of ReLPO during the first hundred iterations, reaches a peak of 56.7 around iteration 210, and subsequently decreases by 1.1 points, while the cross-seed variation increases to the ± 1.1 in Table 3, compared with ± 0.3 for ReLPO. These dynamics provide additional context for the variance results in Table 3. As the distilled fields move away from the region in which the potential was estimated, the resolution target λ^ϕ is increasingly evaluated on states with less constrained potential estimates. The reference regularizer limits this distributional drift, yielding more stable updates across seeds and more consistent improvement over training.

D.6 Refinement–Resolution Behavior and Efficiency

We finally evaluate whether ReLPO produces the intended refinement–resolution behavior. For each realized position, we estimate a counterfactual *premature-resolution* indicator by comparing the terminal reward of the observed realization against a continuation that forces the position to remain unresolved for $\delta = 8$ additional refinement steps. Similarly, a *redundant-refinement* event occurs when the model continues latent refinement although immediate realization does not reduce the estimated terminal return. These quantities are used only for analysis. Table 12 shows improved commitment decisions alongside lower inference cost. ReLPO more than halves premature resolution relative to the supervised checkpoint (8.0% against 18.4%) and reduces redundant refinement by a similar factor (11.4% against 24.0%), whereas the flat outcome-RL baselines improve both only marginally, reaching 16.1% and 21.8% for GRPO and 14.6% and 20.3% for TraDo. The efficiency columns show both baselines increase refinement usage relative to the supervised checkpoint, from 3.08 to 3.39 and 3.32 steps per token, and end-to-end latency rises accordingly to 5.18 s and 5.07 s, alongside their greater refinement usage. ReLPO moves in the opposite direction, reducing usage to 2.67 steps per token and latency to 4.15 s, a 13% reduction against the supervised checkpoint and 20% against GRPO, while remaining the most accurate method in the table. End-to-end latency includes output length, scheduling, and kernel utilization, while steps per token isolates refinement usage per emitted token.

E Case Studies

We present qualitative reasoning examples from **Evo-XXL + ReLPO** showing more complete constraint checking before the final answer. Each emitted token, including an algebraic placeholder, is a discrete commitment. These decoded traces illustrate reasoning behavior. Attribution to latent refinement rests on the controlled ablations and on the GRPO and TraDo comparisons in Table 12. The selected successful examples provide limited coverage of failure behavior. Residual premature resolution and redundant refinement occur at rates of 8.0% and 11.4%, respectively. Inaccurate

potential differences can favor an early commitment or prolong refinement after the useful alternatives have been resolved.

Multi-Constraint Mathematical Reasoning. Table 13 provides a direct example of premature commitment after only part of the problem has been resolved. The supervised Evo-XXL correctly combines the first two congruences and obtains $n \equiv 17 \pmod{35}$. However, it immediately reports 17 as the final answer without incorporating the third congruence. The intermediate result is locally correct because 17 satisfies the first two constraints, but the corresponding reasoning state still contains an unresolved degree of freedom and does not yet determine a globally valid solution.

Evo-XXL+RELPO retains this degree of freedom by writing $n = 35k + 17$ and continues reasoning with the third congruence. Only after determining k does it resolve the trajectory to $n = 157$, followed by an explicit check of all three conditions. This example shows continued textual reasoning after a locally valid intermediate result. Preserving the unresolved variable prevents the model from collapsing a partially solved system into an incorrect final answer.

Combinatorial Reasoning with Coupled Constraints. Table 14 illustrates premature aggregation across interacting constraints. The supervised model identifies the correct decomposition according to whether the last digit is 0 or 5, but counts candidates within those branches without completely enforcing digit uniqueness. In the branch ending in 5, the pairs $(2, 5)$ and $(5, 2)$ satisfy the digit-sum requirement while violating the distinct-digit requirement because the digit 5 has already been fixed.

The RELPO-trained model keeps the two branches separate until their internal constraints have been checked. It verifies all six candidates in the branch ending in 0, then explicitly removes the two invalid candidates in the branch ending in 5 before combining the branch counts. The example therefore shows how RELPO can preserve distinctions between partially resolved alternatives while checking each branch. The improvement comes from delaying the final count until each branch is compatible with the complete set of constraints.

Reasoning Under Spurious Intermediate Solutions. Table 15 considers a case in which an intermediate transformation enlarges the candidate solution set. After squaring the equation, the supervised model correctly obtains two roots of the resulting quadratic but immediately treats both as solutions of the original equation. The algebra following the transformation is valid, yet the transformation itself has introduced candidates whose validity remains unresolved.

Evo-XXL+RELPO preserves information from the original equation by first deriving the requirement $x \geq 2$. The two quadratic roots are then treated as provisional candidates whose validity is checked against the original equation. The model subsequently eliminates $(5 - \sqrt{33})/2$ because it violates the preserved domain constraint. This case illustrates the value of delaying resolution when an intermediate operation produces a broader candidate set than the original problem. The successful trajectory retains the earlier constraint until the newly generated alternatives can be distinguished.

Scientific Reasoning with a Globally Conserved Quantity. Table 16 shows a different form of early commitment in which the model must choose between competing physical principles. The supervised model immediately assumes that electrostatic energy is conserved during charge redistribution. Once this assumption is fixed, the remaining reasoning is internally consistent but necessarily produces the incorrect fraction 1. The failure therefore originates from resolving the governing physical principle before accounting for the complete transition process.

The RELPO-trained model identifies total charge as the quantity preserved during redistribution. It uses charge conservation to determine the equilibrium voltage and only then computes the final stored energy. This produces the correct fraction $1/2$ and also explains the missing energy through dissipation. The example shows that RELPO can improve reasoning when several locally plausible continuations are available but only one remains consistent with the global physical process. The model avoids organizing the entire trajectory around an appealing but incompatible early assumption.

Exact-Answer Number-Theoretic Reasoning. Table 17 illustrates the difference between finding a value that satisfies a condition and establishing the smallest value that satisfies it. The supervised model correctly applies Euler's theorem to obtain $2^{24} \equiv 1 \pmod{35}$. However, it immediately interprets this sufficient exponent as the minimum exponent requested by the problem. The reasoning

therefore resolves the answer before the remaining possibility of a smaller valid exponent has been eliminated.

Evo-XXL+RELPO keeps the minimality question unresolved. It decomposes 35 into 5 and 7, determines that the corresponding multiplicative orders are 4 and 3, and only then combines these conditions through their least common multiple to obtain 12. This example shows how resolution-lifted post-training can discourage commitment to a merely sufficient candidate when the question requires a stronger property. Continued reasoning allows the model to distinguish a valid answer from the minimal valid answer.

Across the five examples, the supervised model repeatedly reaches an intermediate statement that is locally correct or superficially plausible and then resolves it before all relevant information has been incorporated. The value 17 satisfies two of the three congruences, the unfiltered counting branches satisfy part of the combinatorial conditions, both quadratic roots satisfy the transformed equation, energy conservation appears locally plausible as a physical principle, and 24 is indeed an exponent for which the modular equality holds. None of these intermediate results is arbitrary. The failures arise because they are promoted to final decisions while important constraints or alternatives remain unresolved.

The RELPO-trained model behaves differently at these same decision points. It retains the free parameter in Table 13, preserves the counting branches until their candidates have been validated in Table 14, keeps algebraic roots provisional in Table 15, delays selection of the governing physical principle in Table 16, and continues beyond a sufficient exponent to establish minimality in Table 17. These examples illustrate improved checking of constraints and alternatives before a final answer. Event-level refinement traces and matched outcome-RL examples would further clarify the contribution of the latent mechanism.

Table 13: Multi-constraint reasoning before and after RELPO post-training.

User. Find the smallest positive integer n satisfying

$$n \equiv 2 \pmod 5, \qquad n \equiv 3 \pmod 7, \qquad n \equiv 4 \pmod 9.$$

Evo-XXL SFT. Combining the first two congruences gives $n \equiv 17 \pmod{35}$. Therefore, the smallest positive integer satisfying the conditions is $\boxed{17}$.

Evo-XXL + RELPO. The first two constraints give

$$n = 35k + 17.$$

We must still enforce the third constraint.

$$35k + 17 \equiv 4 \pmod 9.$$

Since $35 \equiv 8$ and $17 \equiv 8 \pmod 9$,

$$8k + 8 \equiv 4 \pmod 9,$$

so $8k \equiv 5 \pmod 9$. Because $8^{-1} \equiv 8 \pmod 9$, we obtain

$$k \equiv 4 \pmod 9.$$

The smallest choice is $k = 4$, giving

$$n = 35(4) + 17 = \boxed{157}.$$

Indeed, 157 leaves remainders 2, 3, and 4 modulo 5, 7, and 9, respectively.

Table 14: Combinatorial reasoning with interacting constraints.

User. How many three-digit positive integers have distinct digits, are divisible by 5, and have digits that sum to 12?
Evo-XXL SFT. A number divisible by 5 must end in 0 or 5. Counting the possible pairs for the first two digits gives 6 possibilities when the last digit is 0 and 7 when it is 5. Therefore there are $\boxed{13}$ such numbers.
Evo-XXL + RELPO. We split according to the final digit.
If the units digit is 0, the first two digits must sum to 12. The valid ordered pairs with distinct nonzero hundreds digit are

$$(3,9),(4,8),(5,7),(7,5),(8,4),(9,3),$$

giving 6 numbers.
If the units digit is 5, the first two digits must sum to 7. The candidate ordered pairs are

$$(1,6),(2,5),(3,4),(4,3),(5,2),(6,1),(7,0).$$

The pairs $(2,5)$ and $(5,2)$ repeat the already fixed digit 5, so they are invalid. This leaves 5 numbers.
Hence the total is

$$6+5=\boxed{11}.$$

Table 15: Avoiding premature commitment to an extraneous algebraic solution.

User. Solve

$$\sqrt{x+6} = x - 2.$$

Evo-XXL SFT. Squaring both sides gives

$$x + 6 = (x-2)^2 = x^2 - 4x + 4,$$

so

$$x^2 - 5x - 2 = 0.$$

Therefore,

$$x = \frac{5 \pm \sqrt{33}}{2}.$$

Thus the solutions are

$$\boxed{\frac{5+\sqrt{33}}{2}, \frac{5-\sqrt{33}}{2}}.$$

Evo-XXL + RELPO. Because the left-hand side is nonnegative, the original equation requires

$$x - 2 \geq 0,$$

so $x \geq 2$.
Squaring gives

$$x + 6 = (x-2)^2,$$

and therefore

$$x^2 - 5x - 2 = 0.$$

The two algebraic candidates are

$$x = \frac{5 \pm \sqrt{33}}{2}.$$

However,

$$\frac{5-\sqrt{33}}{2} < 0,$$

which violates the required condition $x \geq 2$ and is therefore extraneous. Hence the unique solution is

$$\boxed{\frac{5+\sqrt{33}}{2}}.$$

Table 16: Scientific reasoning requiring selection of the correct conserved quantity.

User. Two identical capacitors each have capacitance C. One is initially charged to voltage V and the other is uncharged. The charged capacitor is disconnected from its source and then connected in parallel to the uncharged capacitor. What fraction of the initial stored energy remains after equilibrium?

Evo-XXL SFT. The total energy is conserved when the capacitors are connected. Therefore the final stored energy is equal to the initial stored energy, so the fraction remaining is $\boxed{1}$.

Evo-XXL + RELPO. The conserved quantity during charge redistribution is total charge, not electrostatic energy.

Initially,

$$Q_{\text{tot}} = CV, \qquad U_i = \frac{1}{2}CV^2.$$

After the two identical capacitors are connected in parallel, their equivalent capacitance is $2C$. Charge conservation therefore gives

$$V_f = \frac{CV}{2C} = \frac{V}{2}.$$

The final stored energy is

$$U_f = 2\left(\frac{1}{2}C\left(\frac{V}{2}\right)^2\right) = \frac{1}{4}CV^2.$$

Thus

$$\frac{U_f}{U_i} = \frac{\frac{1}{4}CV^2}{\frac{1}{2}CV^2} = \boxed{\frac{1}{2}}.$$

The remaining half of the initial energy is dissipated during charge redistribution.

Table 17: Exact-answer number-theoretic reasoning after RELPO post-training.

User. Find the smallest positive integer n such that

$$2^n \equiv 1 \pmod{35}.$$

Evo-XXL SFT. Since $\phi(35) = 24$, Euler's theorem gives $2^{24} \equiv 1 \pmod{35}$. Therefore the smallest such positive integer is $\boxed{24}$.

Evo-XXL + RELPO. Euler's theorem shows that 24 is a possible exponent, but it does not establish that it is the smallest.

Because $35 = 5 \cdot 7$, we need

$$2^n \equiv 1 \pmod{5} \quad \text{and} \quad 2^n \equiv 1 \pmod{7}.$$

Modulo 5, the multiplicative order of 2 is 4.

$$2^4 \equiv 1 \pmod{5}.$$

Modulo 7, the order is 3.

$$2^3 \equiv 1 \pmod{7}.$$

Therefore n must be a multiple of both 4 and 3. The smallest such value is

$$n = \operatorname{lcm}(4, 3) = \boxed{12}.$$

Indeed, $2^{12} = 4096 \equiv 1 \pmod{35}$.

F LIMITATIONS

Several limitations remain. First, our experiments focus on mathematical reasoning and code generation, where correctness can be evaluated by reliable terminal verifiers. Although RELPO itself only requires a terminal reward, all main experiments use binary outcome signals from symbolic answer checking or executable tests. It therefore remains unclear how well resolution lifting behaves with noisy learned reward models, subjective preferences, or tasks whose quality cannot be determined from a single terminal outcome. Extending the framework to such settings would require studying how verifier error propagates through the lifted potential. Second, the practical RLOV is an approximation to the ideal potential in Eq. (3). Proposition 1 establishes outcome-only identification under a Markovian reference transition kernel, a realizable potential function class, unbiased terminal rewards, and reference-state coverage by the behavior occupancy, assumptions that need not hold exactly for finite neural models. Finite-successor log–Monte-Carlo targets also introduce downward bias. The frozen reference defines reachable outcome mass under π_0, and its continuation distribution can increasingly differ from the improving behavior policy. This mismatch can affect calibration in either direction and limits transfer to states with weak reference support. The nesting result describes hard token commitments, while concentration of the continuation distribution during refinement depends on the learned dynamics. Multiplicative tilting preserves reference support. Rewarding regions with zero reference probability receive zero tilted probability, and very low reference mass still requires exploration and successful samples. The evaluation also uses established benchmarks with possible pretraining overlap. The dated evaluation protocol in Appendix D.1 separates post-training source dates from evaluation dates. Backbone pretraining corpora can still contain related material. Capability retention outside mathematics and code and comparisons against native post-training methods on each diffusion backbone remain outside the present evaluation. Finally, the strongest benefits of RELPO rely on a generator exposing a meaningful hierarchy of revisable states and irreversible resolution transitions. Masked diffusion uses its native discrete denoising update, while purely autoregressive models use discrete realization. The full continuous-refinement and autoregressive-realization coupling applies to hybrid generators. A purely autoregressive control, on which the method reduces to the discrete tilt of Eq. (9), would measure how much of the gain is specific to revisable states and is left for future work. We evaluate several diffusion and AR–diffusion architectures, but all experiments remain within language models and two task domains. Transfer of the resolution hierarchy to other hybrid generative architectures, and of the temperature β and potential parameterization across model families and reward regimes, remains to be established.