# POETS: Uncertainty-Aware LLM Optimization via Compute-Efficient Policy Ensembles

**Nicolas Menet**[1,2] nicolas.menet@ibm.com
**Andreas Krause**[2] krausea@ethz.ch
**Abbas Rahimi**[1] abr@zurich.ibm.com

[1]IBM Research – Zurich, [2]Department of Computer Science, ETH Zürich

## Abstract

Balancing exploration with exploitation is a core challenge in black-box optimization and reinforcement learning. We introduce POETS (**Po**licy **E**nsembles for **T**hompson **S**ampling), a novel framework that bridges uncertainty quantification and policy optimization. Our approach is grounded in the insight that policies trained with Kullback-Leibler (KL) regularization implicitly encode an underlying reward function. Building on this, POETS bypasses the complex, nested process of training an uncertainty-aware reward model and separately fitting a policy to this model. Instead, we directly train a policy ensemble to capture epistemic uncertainty by matching implicitly encoded reward functions to online, bootstrapped data. To overcome the prohibitive compute and memory constraints of ensembling Large Language Models (LLMs), POETS utilizes an efficient architecture: the ensemble shares a pre-trained backbone while maintaining diversity through independent Low-Rank Adaptation (LoRA) branches. Theoretically, we show that POETS implicitly approximates KL-regularized Thompson sampling, a principled optimization strategy with optimal cumulative regret bounds of $\mathcal{O}(\sqrt{T\gamma_T})$. Empirically, we demonstrate that POETS achieves state-of-the-art sample efficiency across diverse scientific discovery domains, including protein search, therapeutic molecular design, and quantum circuit design. POETS also improves the optimization trajectory and out-of-distribution transfer of reinforcement learning across mathematical reasoning and code generation tasks, proving particularly robust in off-policy settings with experience replay or in small dataset regimes.

## 1 Introduction

Efficient exploration in reinforcement learning and black-box optimization requires principled uncertainty quantification [Shahriari et al., 2015, Garnett, 2023]. Ensembling multiple independently trained models is a proven, model-agnostic method to estimate predictive uncertainty by treating the variation in their predictions as a proxy for epistemic uncertainty [Lakshminarayanan et al., 2017]. However, directly applying naive ensembling to Large Language Models (LLMs) is computationally prohibitive. Furthermore, existing exploration methods for LLMs often rely on explicit reward models [Lightman et al., 2024, Coste et al., 2024, Menet et al., 2026b] or depend heavily on the instruction-following capabilities of the model itself [Sutter et al., 2025, Romera-Paredes et al., 2024].

To circumvent the complexities of explicit reward modeling (see Figure 1), this work builds upon a foundational duality: policies optimized with Kullback-Leibler (KL) regularization inherently encode their underlying reward functions [Ziebart, 2010, Neu et al., 2017, Haarnoja et al., 2018, Geist et al., 2019]. Following Rafailov et al. [2023], Gao et al. [2024], Matrenok et al. [2025], we directly match the reward function encoded by the policy to the observed data rather than learning a reward function and fitting a policy to it in two separate steps. In contrast to prior work, we use policy ensembles

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

diversified via Poisson bootstrapping to reliably capture epistemic uncertainty and drive exploration. The resulting algorithm is called **Po**licy **E**nsembles for **T**hompson **S**ampling (POETS).

To overcome the prohibitive compute and memory cost of a naive LLM ensemble, we implement a parameter-efficient "Trunk & Branch" architecture [Lee et al., 2015]. The vast majority of weights are shared in a common backbone, which is only computed once per forward/backward pass. In contrast, the ensemble diversity is effectively captured via independent, low-rank adaptation (LoRA) modules [Hu et al., 2022] applied to the final layers. Our contributions are as follows:

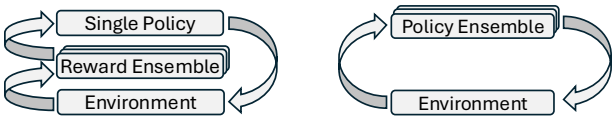



Figure 1: Previous methods for sample efficient RL fit an uncertainty-aware reward model and separately train a single policy to trade off exploration and exploitation (left). POETS directly estimates reward uncertainty with policy ensembles (right).

1. We introduce POETS, a novel framework that captures epistemic uncertainty via policy ensembles. We show that POETS implicitly conducts approximate Thompson sampling [Thompson, 1933, Russo et al., 2018] without an explicit probabilistic reward model.
2. We propose the first policy ensemble that scales to billion-parameter LLMs. By leveraging a specialized Trunk & Branch architecture with intermediate gradient accumulation, running a 16-member ensemble introduces merely a $7.7\%$ runtime and $16.8\%$ memory overhead compared to a single policy instead of the usual $1600\%$ compute & memory overhead.
3. We empirically demonstrate state-of-the-art sample efficiency with POETS across diverse combinatorial black-box optimization domains, including protein and molecular design.
4. We empirically demonstrate that POETS fundamentally improves GRPO's optimization trajectories on both mathematical and coding benchmarks, unlocking the effective use of experience replay without premature overfitting, and improving model generalization.

## 2 Preliminaries

**Setting** We consider a contextual bandit setting [Langford and Zhang, 2007], which is the standard reinforcement learning paradigm for LLM post-training and LLM-driven scientific discovery. Let $\mathcal{X}$ denote the discrete space of contexts (e.g., user prompts) and $\mathcal{A}$ denote the finite space of actions (i.e., generated responses of maximum length). A policy $\pi_\theta(a \mid x)$ parameterized by $\theta \in \mathbb{R}^d$ maps a context $x \in \mathcal{X}$ to a distribution over actions $a \in \mathcal{A}$. We assume access to a reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. For a fixed context $x$, we drop the explicit conditioning and write $r(a)$ and $\pi(a)$ for brevity.

**Ensembling for uncertainty quantification** Ensembling is a highly effective, model-agnostic approach for uncertainty quantification [Lakshminarayanan et al., 2017]. By training multiple models independently with different random initializations and data bootstraps, the variance across their predictions serves as a reliable proxy for epistemic uncertainty [Chua et al., 2018, Pathak et al., 2019, Sukhija et al., 2023]. Traditionally, bootstrapping generates diverse training sets by sampling with replacement from a fixed, offline dataset of size $s$. This makes it ill-suited for the sequential, online data generation process inherent to black-box optimization and reinforcement learning without sacrificing bootstrap consistency across time. To overcome this, we follow Osband et al. [2016] and implement *Poisson bootstrapping* [Oza and Russell, 2001, Qin et al., 2013]. In standard bootstrapping, the number of times a specific data point is included in a resampled dataset of size $s$ follows a binomial distribution, which asymptotically converges to a Poisson distribution: $\mathrm{Bin}(s, 1/s) \to \mathrm{Poisson}(1)$ as $s \to \infty$. Poisson bootstrapping leverages this equivalence for online learning by directly assigning each incoming data point an integer weight drawn independently from $\mathrm{Poisson}(1)$ for every ensemble member.

**Regularized policies are implicit reward functions** A foundational concept for our method is the duality between reward functions and optimal policies under regularized objectives [Ziebart, 2010, Haarnoja et al., 2018]. Consider the "soft reward" objective function regularized by Kullback-Leibler divergence:

$$J(\pi, r) := \mathbb{E}_{a\sim\pi}[r(a)] - \beta D_{KL}(\pi \parallel \pi_{ref}). \tag{1}$$

The optimal policy that maximizes this objective takes a known closed form [Geist et al., 2019]:

$$\pi^*(a) = \pi_{ref}(a) \exp\big(r(a)/\beta - \log Z\big).$$

Here, the log-partition function $\log Z := \log \sum_a \pi_{ref}(a) \exp(r(a)/\beta)$ ensures the policy integrates to one. By rearranging this expression, we see that any policy $\pi$ implicitly encodes a corresponding reward function $r_\pi$ for which it is optimal [Rafailov et al., 2023]:

$$r_\pi(a) := \beta \cdot (\log \pi(a) - \log \pi_{ref}(a) + \log Z). \tag{2}$$

While $\log Z$ is notoriously intractable, it acts strictly as an action-independent constant. This reflects the fundamental property that shifting a reward function by a global additive constant does not alter the optimal policy, a feature we exploit in the next section to bypass estimating $Z$ entirely.

**Notation** For a reward function $r : \mathcal{A} \to \mathbb{R}$ and a distribution $\rho$ over actions in $\mathcal{A}$, we define the expected reward as $r(\rho) := \mathbb{E}_{a' \sim \rho}[r(a')]$ and the centered reward as $r(a - \rho) := r(a) - r(\rho)$. We denote an ensemble of $n$ policies as $\Pi := \{\pi_1, \ldots, \pi_n\}$ with global policy $\pi_{avg}(a) := \frac{1}{n} \sum_{i=1}^{n} \pi_i(a)$.

# 3 Method

The foundational principle of POETS is to *skip explicit reward modeling* (see Figure 1). Instead of learning a separate ensemble of reward functions and fitting a policy to it that balances exploration with exploitation, *we directly match the implicit reward functions encoded by a policy ensemble to the data.* POETS achieves this through three core components: (1) formulating an implicit reward-matching objective that reduces to a simple policy gradient, (2) capturing epistemic uncertainty via online Poisson bootstrapping, and (3) scaling to LLMs via a highly compute-efficient Trunk & Branch architecture.

## 3.1 Ensembling implicit reward functions via bootstrapped policy gradients

To formalize this, we first define the objective that aligns a single policy's implicit reward with the true reward and derive its gradients. Then, we introduce bootstrapping. All proofs are in Appendix D.

**Definition 1** (POETS loss)**.** *Define* $L^\rho_{poets}(a, \pi) := \big(r(a - \rho) - r_\pi(a - \rho)\big)^2$ *where* $\rho$ *is an arbitrary baseline distribution over actions, canceling out the intractable partition function* $\log Z$*.*

**Proposition 1** (POETS gradients)**.** *Let* $\pi$ *be parameterized by* $\theta$ *and define the KL-regularized reward* $\tilde{r}_\pi(a) := r(a) + \beta \cdot (\log \pi_{ref}(a) - \log \pi(a))$*. Then the gradients of the* POETS *loss* $L^\rho_{poets}$ *are*

$$\nabla_\theta L^\rho_{poets}(a, \pi) = -2\beta \cdot \tilde{r}_\pi(a - \rho) \cdot (\nabla_\theta \log \pi(a) - \mathbb{E}_{a' \sim \rho}[\nabla_\theta \log \pi(a')]) \tag{3}$$

$$\mathbb{E}_{a \sim \rho}[\nabla_\theta L^\rho_{poets}(a, \pi)] = -2\beta \cdot \mathbb{E}_{a \sim \rho}\big[\tilde{r}_\pi(a - \rho) \nabla_\theta \log \pi(a)\big]. \tag{4}$$

Note that if we choose $\rho$ to be the empirical distribution over entries in a batch sampled from the current policy, i.e., $\rho(a) = \frac{1}{G} \sum_{i=1}^{G} \delta_{a=a_i}$ for $a_i \sim \pi$, we obtain a surprisingly familiar result. Up to rescaling (via advantage standardization and the factor $2\beta$), Equation (4) is exactly the gradient of GRPO. Moreover, the standard policy gradient baseline [Kool et al., 2019, Ahmadian et al., 2024] used for variance reduction emerges naturally as the anchor across which implicit reward differences are computed. Still, $\rho$ can also be off-policy, mirroring recent analysis of GRPO [Yao et al., 2026].

**Takeaway 1**

Standard GRPO is mathematically equivalent to optimizing a single-policy instance of the POETS loss. By extending this implicit reward-matching objective to an ensemble, POETS seamlessly integrates principled epistemic uncertainty quantification into the GRPO framework.

**Bootstrapping GRPO provides an approximate reward posterior** Crucially, Proposition 1 proves that a policy ensemble $\Pi$ trained via Poisson-bootstrapped GRPO is mathematically equivalent to a bootstrapped ensemble of mean-squared-error (MSE) fitted implicit reward functions. Since bootstrapped MSE-fitted neural networks are well-calibrated and established frequentist approximations of Bayesian posteriors [Osband et al., 2016, Lakshminarayanan et al., 2017, Chua et al., 2018, Pathak et al., 2019, Curi et al., 2020, Sekar et al., 2020, Sukhija et al., 2023], the implicit reward ensemble of POETS inherits this posterior-approximation interpretation. In Section 4, we build on this to show that our practical algorithm approximates Thompson sampling, a principled strategy for balancing exploration and exploitation. In Appendix A.1, we empirically

examine whether uncertainty quantification remains informative in our computationally efficient implementation, and find near-perfect Pearson correlation between the ensemble's implicit reward variance (epistemic uncertainty) and the squared error on the TDC molecular design benchmarks.

**Algorithm** Transforming standard GRPO into POETS requires only a few modifications to the training loop, summarized in Algorithm 1. Instead of updating a single policy, we maintain a lightweight policy ensemble $\Pi = \{\pi_1, \ldots, \pi_n\}$. For each group of rollouts, we assign independent integer weights $w \sim \text{Poisson}(1)$ to each response for every ensemble member. We then compute and apply the bootstrapped policy gradient (Equation (4)) to each policy based on its Poisson weights.

**Algorithm 1** POETS (for single context $x$)

**Input:** policies $\Pi = (\pi_i)_{i=1}^n$, group size $G$, buffer size $T$, regularization $\beta$, optimizer $\text{Optim}$
**while** not converged **do**
  Sample actions $(a_1, \ldots, a_G) \sim \pi_{avg}$ and evaluate rewards $(r_1, \ldots, r_G)$
  Sample Poisson weights $(w_1^i, \ldots, w_G^i) \sim \text{Poisson}(1)$ for each policy $i \in \{1, \ldots, n\}$
  Add actions to $A_{buffer}$, rewards to $R_{buffer}$, and Poisson weights to $W_{buffer}$
  **for** $t = 1, \ldots, T$ **do** ▷ Experience replay over the last $T$ batches (rollout groups)
    Fetch historical batch $a, r, w = A_{buffer}[t], R_{buffer}[t], W_{buffer}[t]$
    **for** $i = 1, \ldots, n$ **do**
      Compute regularized rewards: $\tilde{r}_{\pi_i}(a_j) = r_j + \beta \log \pi_{ref}(a_j) - \beta \log \pi_i(a_j)\ \forall j \in [G]$
      Construct data bootstrap $\rho_i(a) \propto \sum_{j=1}^G w_j^i \cdot \delta_{a=a_j}$ as empirical distribution of batch
      Accumulate gradients $\mathbb{E}_{\rho_i}[\nabla_\theta L_{poets}^{\rho_i}]$ via bootstrapped policy gradient (e.g., GRPO)
    **end for**
    Update shared (trunk) and individual (branch) parameters: $\theta \leftarrow \text{Optim}(\theta, \nabla\theta)$
  **end for**
**end while**

In Algorithm 1, $\pi_{avg}(a) := \frac{1}{n}\sum_{i=1}^n \pi_i(a)$ is the approximate Thompson sampling policy (see Section 4). To sample from it, each group rollout $a_j$ is assigned an index $i \sim \mathcal{U}([n])$ that identifies the policy $\pi_i$ from which it is generated (see Appendix B). $A_{buffer}$, $R_{buffer}$, $W_{buffer}$ are first-in-first-out buffers of size $T$. The entirety of parameters is denoted by $\theta = (\theta_{shared}, \theta_1, \ldots, \theta_n)$. Note that the policy ensemble can share parameters, a crucial property for computational and memory efficiency.

### 3.2 Sublinear compute & memory scaling via Trunk & Branch architecture

A standard implementation of policy ensembles would require maintaining $n$ independent copies of an LLM, which is prohibitively expensive in terms of both memory and compute. Instead, we adopt a parameter-efficient "Trunk & Branch" [Lee et al., 2015] architecture using Low-Rank Adaptation (LoRA) [Hu et al., 2022, Wang et al., 2023]. We decompose the overall parameters $\theta$ into a shared backbone $\theta_{shared}$ (the Trunk) and $n$ independent adapter modules $\theta_i = \{A_i, B_i\}_{i=1}^n$ (the Branches).

**Architecture** While $\theta_{shared}$ includes the vast majority of the weights, we encapsulate the ensemble diversity entirely through the final readout layer, relying on shared contextualized features [Park et al., 2024]. Note that both $\theta_{shared}$ and $\theta_i$ are fine-tuned on the data, so LoRA does not limit the capacity of single-policy fine-tuning (and thus $\pi_{avg}$). We use separate learning rates for trunk & branches to ensure effective diversification. For a language head $h(x) = Wx$, the $i$-th ensemble member computes:

$$h_i(x) = Wx + B_i A_i x,$$

where $W \in \mathbb{R}^{d_v \times d_h}$ is a shared readout matrix, and $A_i \in \mathbb{R}^{r \times d_h}, B_i \in \mathbb{R}^{d_v \times r}$ are the low-rank decomposition matrices for branch $i$ with rank $r \ll d_h, d_v$. Maintaining $n$ branches adds only $n \times r \times (d_h + d_v)$ parameters to the $\Omega(l \times d_h^2 + d_h \times d_v)$ parameters spread across the $l$ layers of the trunk. Sampling from $\pi_{avg}$ comes at zero overhead, since for each rollout of batched generation only one policy (selected uniformly at random) is played. Moreover, because the shared trunk is executed exactly once per pass during training (via gradient accumulation), the primary computational cost is completely decoupled from the ensemble size $n$. As we confirm empirically in Section 5.3, generating responses and accumulating gradients for $n$ policies takes virtually the same wall-clock time as a single policy. In contrast, prior policy ensemble methods require $n$ times more memory and computation [Yang et al., 2022, Liu et al., 2017, Eysenbach et al., 2019, Parker-Holder et al., 2020].

**Takeaway 2**

By capturing epistemic uncertainty within a lightweight Trunk & Branch architecture, POETS essentially matches the memory and computational footprint of standard GRPO.

# 4 Theory

As established in Section 3 and Appendix A.1, POETS maintains an implicit reward ensemble that acts as a frequentist approximation of the Bayesian posterior over reward functions. We now formalize how POETS leverages this property to efficiently approximate KL-regularized Thompson sampling.

**Definition 2** (KL-regularized Thompson sampling)**.** *Given a belief over rewards, KL-regularized Thompson sampling draws a reward function $r$ and then acts according to the policy $\pi$ satisfying*

$$\pi(a) \propto \pi_{ref}(a) \exp(r(a)/\beta) \iff \pi = \arg\max_{\pi'} J(\pi', r). \tag{5}$$

Note that, as the regularization coefficient $\beta \to 0$, the policy concentrates entirely on the maximum reward, i.e., $\pi(a) \to \delta_{a=\arg\max_{a'} r(a')}$, recovering standard non-regularized Thompson sampling.

Crucially, each ensemble member $\pi_i$ in Algorithm 1 is, by definition, the analytical solution to Equation (5) for its corresponding implicit reward $r_{\pi_i}$. Because the ensemble $(r_{\pi_i})_{i=1}^n$ represents our approximate reward posterior, executing KL-regularized Thompson sampling reduces to drawing and playing a policy uniformly at random from the ensemble $\Pi = \{\pi_1, \ldots, \pi_n\}$ [Lu and Van Roy, 2017].

**Takeaway 3**

POETS implicitly approximates KL-regularized Thompson sampling by directly sampling from an uncertainty-aware policy ensemble, bypassing the need for an explicit belief over rewards.

This connection is theoretically significant, since Thompson sampling is a provably efficient algorithm that naturally balances exploration with exploitation. In particular, under standard Bayesian assumptions—where the true reward vector is a realization of a Gaussian prior—KL-regularized Thompson sampling inherits the strong theoretical guarantees of standard Thompson sampling.

**Theorem 1.** *Sequential KL-regularized Thompson sampling has Bayesian cumulative (soft) regret*

$$\textstyle\sum_{t=1}^{T} \mathbb{E}_{r,\pi_t}[\max_\pi J(\pi, r) - J(\pi_t, r)] \in \mathcal{O}(\sqrt{T\gamma_T})$$

*where $\gamma_T$ is the maximal information gain [Srinivas et al., 2010] of a Gaussian prior over reward $r$.*

Because our policies optimize a regularized objective, we bound performance using soft regret. This theorem clarifies why GRPO is prone to overfitting: it is the $n = 1$ instance of POETS, i.e., Thompson sampling with a single MSE-fitted implicit reward function, ceding posterior exploration.

# 5 Experiments

We empirically evaluate POETS on black-box optimization across four diverse domains for scientific discovery as well as on reinforcement learning via verifiable rewards in the settings of math and coding. We report the mean and standard error of each metric using 25 random seeds. Our code is available at `https://github.com/IBM/thompson-sampling-via-fine-tuning-of-llms` and the experimental setup is detailed in Appendix C. Additional results and ablations are in Appendix A.

## 5.1 POETS for black-box optimization

We consider the *best-seen reward* of a bandit without observation noise, defined as the maximum reward observed up to step $t$, i.e., $\max_{\tau \le t} r(a_\tau)$. This metric captures the best solution found so far and is often used in black-box optimization benchmarks. Note the connection to simple regret, given by $\max_a r(a) - \max_{\tau \le t} r(a_\tau)$. Within a collection of methods covering standard policy gradient, in-context and gradient-based Bayesian optimization, and evolutionary search, POETS exhibits state-of-the-art sample efficiency and computational efficiency. Moreover, sample efficiency can be improved even more by increasing the computational effort per round of Bayesian optimization via replay buffers, a technique that we demonstrate fails without ensembling. Details are in Appendix C.1.

### 5.1.1 Baselines

We compare against four baselines. As a strong reinforcement learning baseline, we consider an improved version of GRPO [Shao et al., 2024] introduced by Zheng et al. [2025] and used to post-train models in the Qwen3 family [Qwen Team, 2025]. As alternative methods for implementing Thompson sampling with LLMs, we consider TOSFIT [Menet et al., 2026b] and FIBO [Sutter et al., 2025, Menet et al., 2026a]. Whereas TOSFIT fine-tunes LLMs toward a variational Thompson sampling policy based on a Gaussian process reward surrogate, FIBO approximates Thompson sampling in-context by providing the strongest candidate solutions thus far along with their scores to the LLM. Among reward-model-based uncertainty approaches, TOSFIT represents a principled upper bound due to its calibrated probabilistic reward model. Finally, EVOSEARCH prompts a language model to conduct search via evolutionary operators such as crossover and mutation of candidates. It is a simplified implementation by Menet et al. [2026b] of FunSearch/AlphaEvolve/OpenEvolve [Romera-Paredes et al., 2024, Novikov et al., 2025, Sharma, 2025]. To verify its performance, we reproduce the FunSearch bin-packing benchmark in Appendix A.10. For classical non-LLM baselines, consider Appendix A.11. All methods operate in batched mode with batch size 16. Table 1 provides a qualitative comparison.

Table 1: POETS is the first method for black-box optimization that is aware of epistemic uncertainty while neither relying on explicit reward models nor on instruction-following capabilities of LLMs.

| | EVOSEARCH | FIBO | TOSFIT | GRPO | POETS (ours) |
|---|---|---|---|---|---|
| Uncertainty aware | no | yes | yes | no | yes |
| Instruction following agnostic | no | no | yes | yes | yes |
| Reward model free | yes | yes | no | yes | yes |
| Risk of policy collapse | low | high | low | high | low |
| Architectural overhead | Multiple calls | Long context | Reward model | None | Policy ensemble |
| Exploration strategy | Mutation | TS | TS | Entropy | TS+Entropy |

### 5.1.2 Settings

**FAQ refinement** is a task proposed by Menet et al. [2026b] that tests the ability to optimize text based on semantic alignment. We ask a Qwen3-8B model [Qwen Team, 2025] to write a frequently-asked-questions (FAQ) response. The reward is modeled as the alignment to an unknown ground-truth, judged by the Qwen3-Embedding-0.6B model [Zhang et al., 2025]. As a deep kernel for the TOSFIT baseline, we use a linear kernel over the first 256 entries of the embeddings of Qwen3-Embedding-0.6B. The search space contains all token sequences, growing exponentially in the response length.

**Protein search** is another setup from Menet et al. [2026b] that explores the challenge of designing thermally stable proteins, a task with significant implications for industrial biotechnology. We sample amino acid sequences from ProtGPT2 [Ferruz et al., 2022] (0.738B parameters) and score them according to their negative thermal instability index [Guruprasad et al., 1990]. Two baselines, FIBO and EVOSEARCH (LLM), require instruction-tuned models. There, we use the much larger and more expensive Qwen3-8B model instead. For TOSFIT, we follow Menet et al. [2026b] and adopt a GP with linear kernel over the mean token embeddings from ProtGPT2 projected to the unit sphere.

**Quantum circuit design** is the final task considered from Menet et al. [2026b]. It entails designing quantum circuits that prepare low-energy quantum states in unknown environments. To generate 7-qubit Qiskit circuits [Javadi-Abhari et al., 2024], we use a Qwen3-8B model. As reward, we consider the negative energy of the prepared state under an unknown Hamiltonian, requiring superposition and entanglement of qubits for optimal performance. The Gaussian process feature map for TOSFIT consists of a code validity bit as well as all two-qubit Pauli observables, which can be efficiently simulated using quantum computers. Note that for the Hamiltonian considered, the ground-truth reward is a linear map of these features, a setup tailored to benefit the TOSFIT baseline.

**TDC molecular design** explores multi-objective de novo molecular generation using the GP-MoLFormer model [Ross et al., 2025]. We optimize SMILES strings to balance target-specific binding affinity, drug-likeness, and synthesizability across three therapeutic targets (GSK3$\beta$, JNK3, DRD2) [Huang et al., 2021], with full details of the setup and results provided in Appendix A.9.

### 5.1.3 POETS obtains state-of-the-art sample efficiency even without replay buffer

Consider Figure 3, where the replay buffer is disabled by setting $T = 1$ in Algorithm 1. POETS is the only method to consistently obtain state-of-the-art sample efficiency across the highly diverse combinatorial tasks. Note that the baselines EvoSearch and GRPO conduct undirected exploration without any notion of *optimism in the face of uncertainty*, i.e., an exploration bias guided by the reward potential of unexplored regions. As confirmed in Figure 2, the optimism of POETS results in a more stable exploration-exploitation tradeoff than that of GRPO, thus managing to better retain generation diversity. FIBO and ToSFiT are closest to POETS in spirit, but the former relies on in-context learning instead of parameter updates and the latter requires an external reward model. As can be seen from the result, these requirements lead to less reliable performance than POETS.

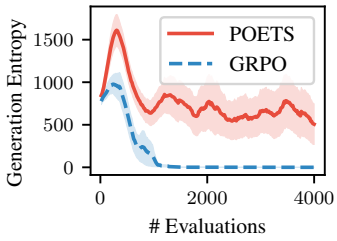



Figure 2: POETS balances exploration with exploitation, leading to stable policy diversity. Shown on Protein Search, for other settings, see Section A.2.

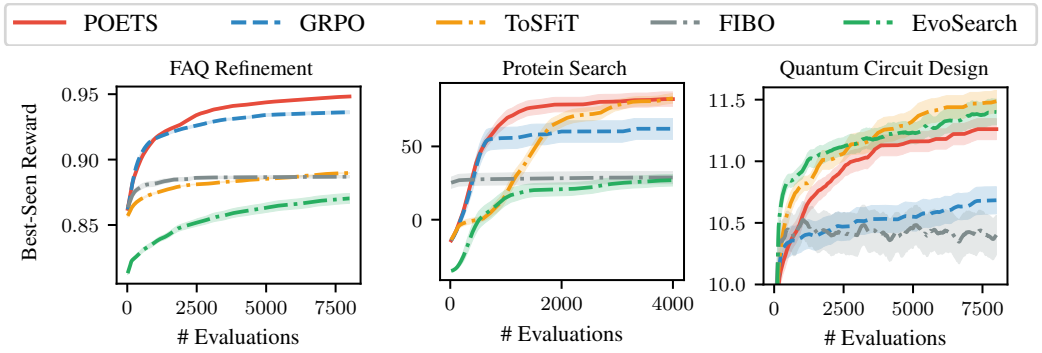



Figure 3: POETS reaches SoTA sample efficiency across diverse tasks, even without a replay buffer.

### 5.1.4 POETS trades off compute efficiency with sample efficiency using a replay buffer

Increasing the size of the replay buffer provides a substantial boost to sample efficiency. As demonstrated on the Protein Search task, setting the buffer size to $T > 1$ in Algorithm 1 yields significant improvements in both best-seen and expected reward (Figure 4). For results on quantum circuit design, see Appendix A.8. In contrast, standard GRPO is blind to epistemic uncertainty; rather than benefiting from replay, it overfits to early candidates. To understand the mechanism preventing POETS from premature overfitting, we can track the normalized Jensen-Shannon divergence across the policy heads: $JSD(\pi_1, \ldots, \pi_n) := \frac{1}{n \log n} \sum_{i=1}^{n} D_{KL}(\pi_i || \pi_{avg}) \in [0, 1]$. As shown in the rightmost panel of Figure 4, the policies naturally diversify from their shared initialization. This diversity is maintained until convergence to the global optimum is achieved, at which point the policies align.

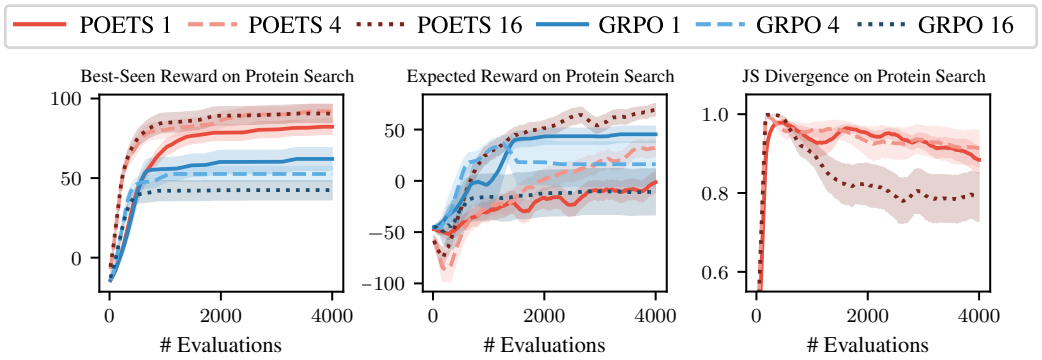



Figure 4: POETS benefits substantially from a replay buffer ($T = 4, 16$). In contrast, GRPO overfits to suboptimal early solutions. After zero-initialization of LoRA, the policy heads diversify quickly.

In addition, the JSD trajectory provides a reliable early-stopping metric for hyperparameter tuning. For example, in all our experiments we tune the LoRA learning rate by sweeping across 3 to 4 values between $10\times$ and $100\times$ of the shared trunk learning rate; the ideal configuration rapidly maximizes JSD before slowly decaying, a sweet spot identifiable at just 5% of the training budget (Appendix A.3). Branch-only updates cannot destabilize the shared backbone, so large branch learning rates are safe.

### 5.2 POETS for reinforcement learning

Next, we consider reinforcement learning with verifiable rewards (RLVR) [Lambert et al., 2025] on the AIME dataset. In this setting, the environment provides a sparse, binary reward strictly based on whether the final answer generated within the `\boxed{}` tags matches the ground truth. Since the policy is trained to act for a distribution of questions, RLVR can be modeled as a contextual bandit problem. Adapting POETS to this setting is straightforward: we simply add another loop over contexts in Algorithm 1. Figure 5 compares the optimization trajectories of POETS and GRPO as a function of the number of rollouts/generations. We evaluate both on-policy optimization (buffer size $T = 1$) and off-policy optimization with experience replay ($T = 4$). As shown, standard GRPO is highly prone to overfitting to suboptimal solutions when exposed to repeated contexts, both in multi-epoch training (Figure 5, left panel) and when utilizing experience replay (dashed lines).

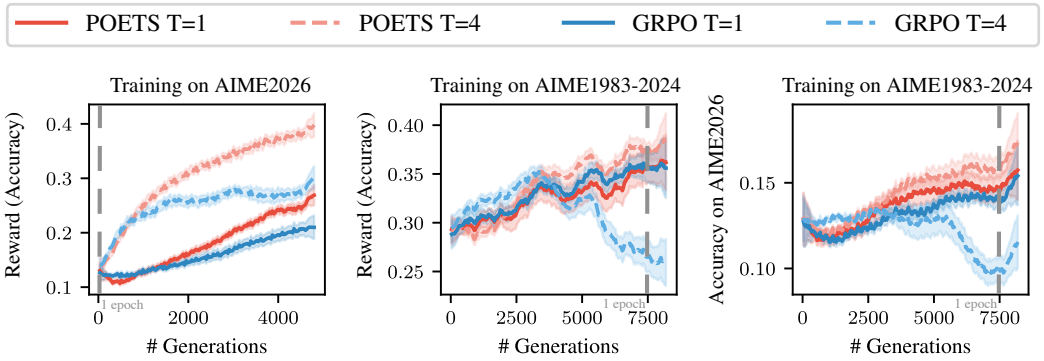



Figure 5: POETS improves the sample efficiency and robustness of RL both on-policy and off-policy. Using a Qwen3-8B model, rollouts are generated in groups of 8 for 4 questions simultaneously.

In contrast, POETS successfully mitigates these failures by implicitly realizing Thompson sampling. Because the policy ensemble maintains awareness of epistemic uncertainty, it naturally balances exploration and exploitation. This demonstrates that principled uncertainty quantification becomes essential as the risk of overfitting increases due to context repetition.

Finally, we consider reinforcement learning on LiveCodeBench v6, a subset of LiveCodeBench [Jain et al., 2025] containing 175 contest-style coding problems released between January and April 2025. In this setting, the environment provides dense rewards based on the fraction of passed unit tests, averaging 2.65 public tests for training and 37.35 private tests for evaluation. We report on the (public test & private test) accuracy as the fraction of codes that pass all tests. In contrast to AIME, where we follow standard GRPO best-practice with a unified learning rate, we now modify GRPO and consider separate learning rates for the main model parameters $\theta_{shared}$ and a LoRA subspace of the readout layer, mirroring the architecture of our compute-efficient POETS. As demonstrated by Figure 6, GRPO overfits to the public tests given the additional faster-learning LoRA readout head, while the inherent uncertainty quantification of POETS results in a reliable transfer from public test performance to private test performance. The improved robustness to changes in learning rates is also mirrored in our results on Therapeutics Data Commons in Appendix A.9. Moreover, we also observe improved OOD transfer on the task of bin-packing heuristic search, see Appendix A.10.

**Takeaway 4**

POETS consistently improves optimization trajectories of GRPO, achieving state-of-the-art black-box optimization, more robust off-policy reinforcement learning, and less reward hacking.

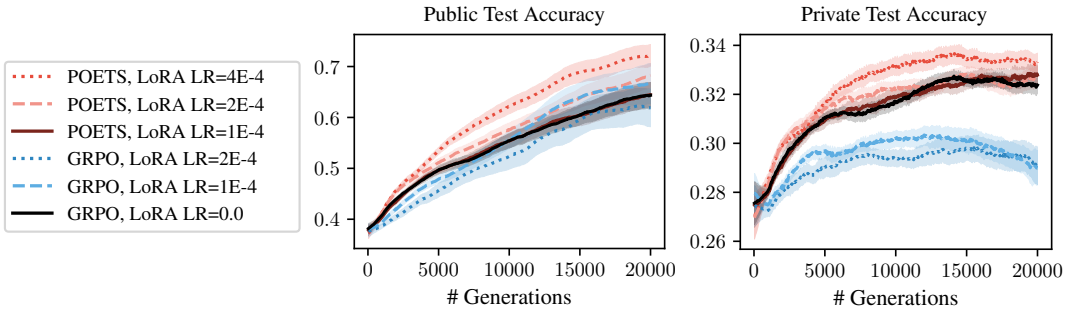



Figure 6: We train a Qwen3-8B model on the public tests of LiveCodeBench v6. Whereas GRPO overfits to the public tests, the uncertainty-aware optimization of POETS ensures reliable transfer from public to private tests. The trunk $\theta_{shared}$ is always trained under learning rate $10^{-5}$, rollouts are generated in groups of 8 for 4 questions simultaneously, and a replay buffer of size T=4 is used.

### 5.3 Computational overhead

Due to the low overhead of our last-layer "Trunk & Branch" LoRA implementation, the benefits of POETS come at minimal computational cost. As detailed in Table 2, training Qwen3-8B on AIME1983-2024 with a replay buffer of $T = 4$ on a single NVIDIA H200 **introduces a mere 7.7% total runtime overhead** compared to standard GRPO. To cut costs further, note that the ensemble is already fully diversified and would achieve equivalent accuracy at $n = 8$ (Figure 10 in Appendix A).

Table 2: Our Trunk & Branch implementation adds only a small overhead compared to a single policy.

| | $n = 1$ | $n = 2$ | $n = 4$ | $n = 8$ | $n = 16$ (default) |
|---|---|---|---|---|---|
| Peak VRAM (GB) | 116.0 | $116.8^{+0.7\%}$ | $118.0^{+1.7\%}$ | $124.9^{+7.7\%}$ | $135.5^{+16.8\%}$ |
| AR Generation (s / round) | 81.4 | $80.7^{-0.9\%}$ | $81.4^{+0.0\%}$ | $81.1^{-0.4\%}$ | $81.0^{-0.5\%}$ |
| Parallel Training (s / round) | 36.6 | $37.2^{+1.6\%}$ | $38.6^{+5.5\%}$ | $41.1^{+12.3\%}$ | $46.1^{+26.0\%}$ |
| AR Generation + Parallel Training (s / round) | 118.0 | $117.9^{+0.0\%}$ | $120.0^{+1.7\%}$ | $122.2^{+3.6\%}$ | $127.1^{+7.7\%}$ |

## 6 Related work

### 6.1 Large language models for optimization and scientific discovery

Applying LLMs to complex search spaces like protein design [Ferruz et al., 2022, Lin et al., 2023] and quantum circuit synthesis [Javadi-Abhari et al., 2024, Vishwakarma et al., 2024] is a rapidly growing area [Kristiadi et al., 2024, Ranković and Schwaller, 2025]. Existing approaches leverage LLMs as evolutionary operators [Romera-Paredes et al., 2024], or implement Thompson sampling [Thompson, 1933, Russo et al., 2018] either via in-context conditioning (FIBO) [Sutter et al., 2025, Menet et al., 2026a] or via fine-tuning based on an explicit reward model (ToSFiT) [Menet et al., 2026b]. Unlike these methods, POETS does not rely on fragile in-context instruction following or additional external reward models, allowing it to fully leverage domain-specific pre-trained models.

### 6.2 Policy gradient methods for large language model post-training

GRPO [Shao et al., 2024] and its variants [Zheng et al., 2025, Liu et al., 2025, Yu et al., 2025] have emerged as strong, critic-free RL baselines for post-training LLMs [DeepSeek-AI, 2024, 2025, Qwen Team, 2025]. However, standard policy gradient methods [Williams, 1992, Schulman et al., 2017] such as GRPO conduct undirected exploration and are blind to epistemic uncertainty. POETS introduces a simple structural modification to policy gradient methods that captures this uncertainty, stabilizing the exploration-exploitation tradeoff without any explicit external reward model.

### 6.3 Ensembles for uncertainty quantification and targeted exploration

Deep ensembles are a popular method for uncertainty quantification [Lakshminarayanan et al., 2017, Yang et al., 2026], previously applied to Thompson sampling by fitting independent reward models to bootstrapped data [Eckles and Kaptein, 2014, Osband et al., 2016, Lu and Van Roy, 2017]. Instead of reward ensembles, policy ensembles have also been considered in prior works, where diversity is induced via additional loss terms [Yang et al., 2022], e.g., based on Stein discrepancies [Liu et al., 2017], determinantal diversity [Parker-Holder et al., 2020], or mutual information [Eysenbach et al., 2019]. Alternative uncertainty mechanisms such as dropout-based uncertainty [Gal and Ghahramani, 2016] are computationally convenient but provide a weak proxy for epistemic uncertainty, failing to exhibit the desired progression from high initial uncertainty to confident convergence [Osband et al., 2016].

These prior methods for policy ensembling force exploration indiscriminately, wasting compute on known low-reward regions. In contrast, POETS grounds policy diversity strictly in epistemic uncertainty via online Poisson bootstrapping [Osband et al., 2016]. Beyond these algorithmic differences, prior policy ensemble methods require fully independent networks, resulting in $n$ times more memory and compute for ensembles of size $n$, which is computationally prohibitive for LLMs. By combining a shared backbone with independent Low-Rank Adaptation [Hu et al., 2022] heads via intermediate gradient accumulation, POETS introduces the first policy ensemble that scales to billion-parameter LLMs, capturing the diversity required for effective exploration [Russo and Van Roy, 2014, 2016] while barely exceeding the memory and compute footprint of a single policy.

## 7 Conclusion

In this work, we introduce POETS (**Po**licy **E**nsembles for **T**hompson **S**ampling), a framework that enables principled, uncertainty-aware exploration for scientific discovery and reinforcement learning. Based on a well-established duality between regularized policies and implicit reward functions, POETS approximates Thompson sampling without any explicit reward belief. It thus sidesteps the nested optimization procedure associated with separately training a reward model and an acting policy.

To overcome the prohibitive computational cost of ensembling LLMs, we deploy a "Trunk & Branch" architecture. By sharing the pre-trained backbone, confining ensemble diversity to independent last-layer LoRA adapters, and accumulating gradients, POETS achieves the exploration benefits of deep ensembles while maintaining a memory and computational footprint comparable to standard GRPO.

Theoretically, we cast POETS as approximate Thompson sampling, which is known to achieve an optimal cumulative regret bound of $\mathcal{O}(\sqrt{T\gamma_T})$ [Scarlett et al., 2017]. Empirically, POETS demonstrates state-of-the-art sample efficiency across diverse black-box optimization tasks, including FAQ refinement, protein search, therapeutic molecular design, and quantum circuit design. Furthermore, in reinforcement learning settings such as mathematical reasoning and code generation, POETS balances the exploration-exploitation tradeoff to maintain generation diversity and navigate unknown but promising regions. This approach improves the sample efficiency of GRPO in both on-policy and off-policy settings, and enables the effective use of experience replay without premature overfitting.

**Limitations & Future Work:** Theorem 1 characterizes the regret of exact KL-regularized Thompson sampling; the $\mathcal{O}(\sqrt{T\gamma_T})$ bound therefore does not constitute a formal regret guarantee for POETS itself. Nevertheless, the theorem serves as theoretical motivation: it provides a principled explanation for why maintaining a reward posterior can improve exploration over single-policy methods such as GRPO, which lack posterior exploration. Quantifying the approximation error introduced by the Poisson bootstrapped ensemble in order to derive formal guarantees for POETS is an interesting open problem, which we leave to future work. Although our efficient Trunk & Branch LoRA implementation significantly improves computational efficiency, it may also reduce the approximation quality of the implicit reward posterior compared to a standard ensemble, and requires some tuning of the branch learning rate (Section 5.1.4) to ensure sufficient diversification. These are deliberate tradeoffs and the key ingredients that make POETS the first policy ensemble that scales to billion-parameter LLMs. Future research directions include extending POETS from the contextual bandit setting of LLM post-training and scientific discovery to multi-step settings with step-wise rewards. This is non-trivial, since the log partition function $\log Z$ becomes state-dependent and can no longer be easily canceled. By offering a scalable and model-agnostic approach to exploration, POETS provides a foundation for applying LLMs to complex sequential decision-making and scientific discovery.

## Acknowledgment

This work is supported by the Swiss National Science Foundation (SNSF), grant 10002666. We thank Jonas Hübotter and Simon Matrenok for providing feedback on an initial version of the paper.

## References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, volume 1, 2024.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160. Proceedings of Machine Learning Research, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, 2024.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *Advances in Neural Information Processing Systems*, 33:14156–14170, 2020.

DeepSeek-AI. DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437, 2024.

DeepSeek-AI. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z.

Dean Eckles and Maurits Kaptein. Thompson sampling with the online bootstrap. arXiv preprint arXiv:1410.4009, 2014.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, 2009.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348, 2022.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. Proceedings of Machine Learning Research, 2016.

Zhaolin Gao, Jonathan Chang, Wenhao Zhan, Owen Oertell, Gokul Swamy, Kianté Brantley, Thorsten Joachims, Drew Bagnell, Jason D Lee, and Wen Sun. Rebel: Reinforcement learning via regressing relative rewards. In *Advances in Neural Information Processing Systems*, volume 37, pages 52354–52400, 2024.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, 2019.

Kunchur Guruprasad, BV Bhasker Reddy, and Madhusudan W Pandit. Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering, Design and Selection*, 4, 1990.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics Data Commons: Machine learning datasets and tasks for drug discovery and development. arXiv preprint arXiv:2102.09548, 2021.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit. arXiv preprint arXiv:2405.08810, 2024.

Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *International Conference on Learning Representations 2019 Deep Reinforcement Learning meets Structured Prediction Workshop*, 2019.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at LLMs for material discovery: Are they actually good for Bayesian optimization over molecules? In *International Conference on Machine Learning*, 2024.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why M heads are better than one: Training a diverse ensemble of deep networks. arXiv preprint arXiv:1511.06314, 2015.

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*, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. arXiv preprint arXiv:1704.02399, 2017.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. In *Conference on Language Modeling*, 2025.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. *Advances in Neural Information Processing Systems*, 30, 2017.

Simon Matrenok, Skander Moalla, and Caglar Gulcehre. Quantile reward policy optimization: Alignment with pointwise regression and exact partition functions. In *Advances in Neural Information Processing Systems*, 2025.

Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. Proceedings of Machine Learning Research, 2020.

Nicolas Menet, Jonas Hübotter, Parnian Kassraie, and Andreas Krause. LITE: Efficiently estimating Gaussian probability of maximality. In *International Conference on Artificial Intelligence and Statistics*, volume 258, pages 4996–5004. Proceedings of Machine Learning Research, 2025.

Nicolas Menet, Michael Hersche, Andreas Krause, and Abbas Rahimi. A theoretical analysis of test-driven code generation. *Advances in Neural Information Processing Systems*, 39, 2026a.

Nicolas Menet, Aleksandar Terzic, Michael Hersche, Andreas Krause, and Abbas Rahimi. Thompson sampling via fine-tuning of LLMs. In *International Conference on Learning Representations*, 2026b.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. Proceedings of Machine Learning Research, 2016.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized Markov decision processes. arXiv preprint arXiv:1705.07798, 2017.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131, 2025.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. Proceedings of Machine Learning Research, 2001.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *International Conference on Machine Learning*, 2024.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18050–18062, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071. Proceedings of Machine Learning Research, 2019.

Zhen Qin, Vaclav Petricek, Nikos Karampatziakis, Lihong Li, and John Langford. Efficient online bootstrapping for large scale learning. arXiv preprint arXiv:1312.5021, 2013.

Qwen Team. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741, 2023.

Bojana Ranković and Philippe Schwaller. GOLLuM: Gaussian process optimized LLMs – reframing LLM finetuning through Bayesian optimization. arXiv preprint arXiv:2504.06265, 2025.

RDKit. RDKit: Open-source cheminformatics, 2026. URL `https://www.rdkit.org`.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Jerret Ross, Brian Belgodere, Samuel C Hoffman, Vijil Chenthamarakshan, Jiri Navratil, Youssef Mroueh, and Payel Das. GP-MoLFormer: A foundation model for molecular generation. *Digital Discovery*, 4(10):2684–2696, 2025.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39, 2014.

Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research*, 17, 2016.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11, 2018.

Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. Lower bounds on regret for noisy Gaussian process bandit optimization. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1723–1742. Proceedings of Machine Learning Research, 2017.

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

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. Proceedings of Machine Learning Research, 2020.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

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.

Asankhaya Sharma. OpenEvolve: an open-source evolutionary coding agent, 2025. URL `https://github.com/algorithmicsuperintelligence/openevolve`.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

Bhavya Sukhija, Lenart Treven, Cansu Sancaktar, Sebastian Blaes, Stelian Coros, and Andreas Krause. Optimistic active exploration of dynamical systems. *Advances in Neural Information Processing Systems*, 36:38122–38153, 2023.

Gustavo Sutter, Mohammed Abdulrahman, Hao Wang, Sriram Ganapathi Subramanian, Marc St-Aubin, Sharon O'Sullivan, Lawrence Wan, Luis Ricardez-Sandoval, Pascal Poupart, and Agustinus Kristiadi. Simplifying Bayesian optimization via in-context direct optimum sampling. arXiv preprint arXiv:2505.23913, 2025.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Sanjay Vishwakarma, Francis Harkins, Siddharth Golecha, Vishal Sharathchandra Bajpe, Nicolas Dupuis, Luca Buratti, David Kremer, Ismael Faro, Ruchir Puri, and Juan Cruz-Benito. Qiskit HumanEval: An evaluation benchmark for quantum code generative models. In *IEEE International Conference on Quantum Computing and Engineering*, volume 1, pages 1169–1176. IEEE, 2024.

Xi Wang, Laurence Aitchison, and Maja Rudolph. LoRA ensembles for large language model fine-tuning. arXiv preprint arXiv:2310.00035, 2023.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 1992.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2020.

Daniel Yang, Samuel Stante, Florian Redhardt, Lena Libon, Parnian Kassraie, Ido Hakimi, Barna Pásztor, and Andreas Krause. RewardUQ: A unified framework for uncertainty-aware reward models. arXiv preprint arXiv:2602.24040, 2026.

Zhengyu Yang, Kan Ren, Xufang Luo, Minghuan Liu, Weiqing Liu, Jiang Bian, Weinan Zhang, and Dongsheng Li. Towards applicable reinforcement learning: Improving the generalization and sample efficiency with policy ensemble. arXiv preprint arXiv:2205.09284, 2022.

Chaorui Yao, Yanxi Chen, Yuchang Sun, Yushuo Chen, Wenhao Zhang, Xuchen Pan, Yaliang Li, and Bolin Ding. Group-relative REINFORCE is secretly an off-policy algorithm: Demystifying some myths about GRPO and its friends. In *International Conference on Learning Representations*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025.

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

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Carnegie Mellon University, 2010.

# A Additional experiments and ablations

## A.1 POETS learns a calibrated reward ensemble

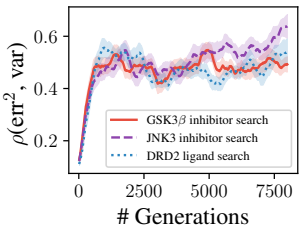



Figure 7: The implicit reward ensemble of POETS achieves calibrated uncertainty estimation.

As established in Section 3, POETS optimizes a policy ensemble that inherently encodes an ensemble of implicit reward functions. To justify the connection to Thompson sampling and thus Theorem 1, we now provide evidence that the implicit reward ensemble is well-calibrated and thus a strong frequentist approximation to the Bayesian reward posterior. Because the intractable log-partition function $\log Z$ acts strictly as an action-independent constant, the implicit rewards are only defined up to a global offset. To bypass this constant, our GRPO-based objective (Equation 4) anchors the optimization to an empirical baseline, specifically the average reward of the rollout group. Consequently, POETS effectively learns an approximate posterior over *implicit advantages* (reward differences relative to a baseline) rather than absolute rewards. To verify that this implicit advantage ensemble is well-calibrated, we evaluate it on the Therapeutics Data Commons (TDC) molecular design benchmarks. For each generated candidate, we compute the Pearson correlation $\rho(\text{err}^2, \text{var})$ between the ensemble's variance (the epistemic uncertainty over the implicit advantages) and the actual squared error (the squared difference between the ensemble's mean implicit advantage and the true oracle advantage). For perfectly well-calibrated ensembles, a Pearson correlation of at most $57.7\%$ can be reached (see Proposition 2). As shown in Figure 7, POETS yields a near-optimally calibrated posterior: the Pearson correlation averages $47\%$ for GSK3$\beta$, $50\%$ for JNK3, and $47\%$ for DRD2.

**Proposition 2** (Pearson correlation for perfect Gaussian calibration)**.** *Let* $\text{var} > 0$ *be a random variable representing the predicted variance, and let* $\text{err}$ *be the signed error, such that* $\text{err}^2$ *is the squared error. Assume the model is perfectly calibrated under Gaussian assumptions, i.e.,* $\text{err}^2 = \text{var} \cdot Z^2$ *for* $Z \sim \mathcal{N}(0, 1)$ *independent of* var*. Then, the Pearson correlation* $\rho(\text{err}^2, \text{var})$ *is bounded from above:*

$$\rho(\text{err}^2, \text{var}) \leq \frac{1}{\sqrt{3}} \approx 0.577$$

## A.2 Policy ensembling ensures generation diversity in black-box optimization

A longstanding challenge of policy gradient methods is entropy collapse [Mei et al., 2020]. By considering epistemic uncertainty, POETS is naturally robust to such degeneracies, see Figure 8. Notably, POETS does not rely on strong entropy regularization [Mnih et al., 2016], which affects the reward landscape and thus changes the optimal policy [Haarnoja et al., 2018, Ahmed et al., 2019].

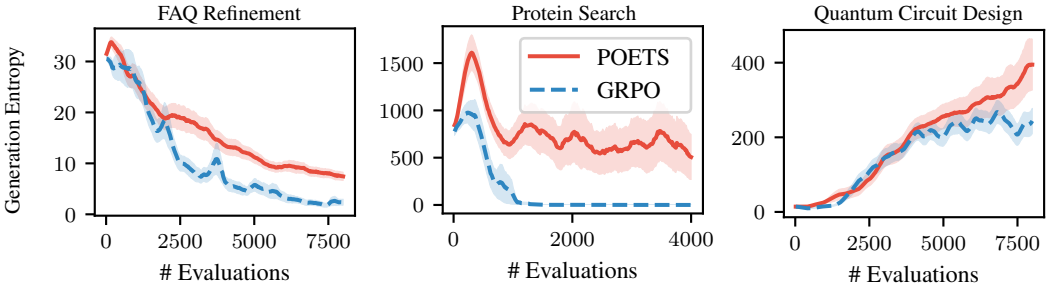



Figure 8: POETS does not suffer from the diversity collapse of GRPO. Note that we have added entropy regularization to the quantum circuit design task, since the starting entropy is extremely low.

We note that since rollout generation and model training share the same HuggingFace Transformers stack, the importance sampling ratio is exactly 1.0 in the on-policy setting (T=1), meaning importance clipping is inactive. DAPO's [Yu et al., 2025] clip-higher design to avoid diversity collapse is thus inapplicable (and unnecessary) for the majority of our experimental configurations, which is why we universally default to the well-established GSPO [Zheng et al., 2025] variant of GRPO [Shao et al., 2024].

### A.3 The policy ensemble diversifies effectively despite parameter sharing

Independent ensembles are known to diversify naturally [Lakshminarayanan et al., 2017]. However, given the trend in up-scaling LLMs, running a forward/backward pass for $n$ independent models would be prohibitive both in terms of compute and memory. Our proposed solution in Section 3.2 is to use a single LLM as a shared backbone, and then construct a policy ensemble through LoRA head adapters. While computationally efficient, the tying of weights between separate policies in the ensemble risks diversity collapse across ensemble members. As a remedy, we propose to use separate learning rates for the shared backbone and the LoRA heads. Consider our ablation on the LoRA head learning rate in Figure 9 using Qwen3-8B with $T = 4$, where the backbone learning rate was selected via grid-search to $2.5 \times 10^{-6}$ while keeping the ensemble learning rate fixed to 0 (see Section C.4).

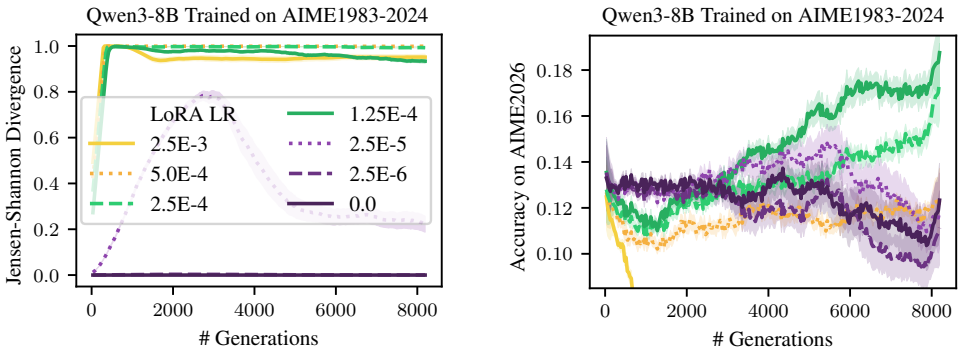



Figure 9: By selecting the correct learning rate for the LoRA heads of POETS, the policy ensemble is effectively diversified despite adopting a shared backbone (the trunk) for training efficiency.

We remark that the best performing configuration has the property that the (normalized) Jensen-Shannon divergence across the policy ensemble approximately reaches its maximum value of 1 after just 5% of the training budget, but then slowly decays, thereby allowing the ensemble to gradually converge to a single optimal policy. This pattern of rapid diversification followed by slow convergence consistently leads to the best trajectories across all our experiments. As such, LoRA learning rates that strike a good balance between exploration and exploitation can be identified early in training.

### A.4 Small ensembles suffice to effectively capture the reward posterior

In Figure 10, we ablate the number of policy heads $n$ that make up the ensemble. Recall that each policy implicitly encodes a reward function, as such, the size of the ensemble directly affects the fidelity of the probabilistic reward model. While larger policy ensembles improve performance, the marginal improvement vanishes for ensembles larger than 8. Unless otherwise stated, we adopt an ensemble size 16 in all experiments.

Setting the maximum generation length to 2048 tokens, autoregressively generating from $\pi_{avg}$ 8 rollouts for each of the 4 task descriptions requires 81.1 seconds. Note that, according to Algorithm 1, rollout cost does not depend on the ensemble size. In contrast, training does. However, using our efficient "Trunk & Branch" implementation, the computational overhead is minor at 7.7% in total. Table 2 provides detailed measurements.

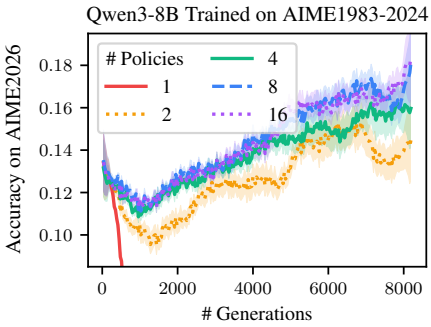



Figure 10: Increasing the ensemble size results in improved performance. At 8 members, the ensemble is sufficiently diverse.

### A.5 Large LoRA ranks improve ensemble diversification

Next, we consider the impact of the LoRA rank on the policy ensemble diversity and thus the optimization trajectory. As can be seen in Figure 11, increasing the LoRA rank from 16 to 256 provides consistent gains on the ensemble diversity, as measured by the Jensen-Shannon divergence.

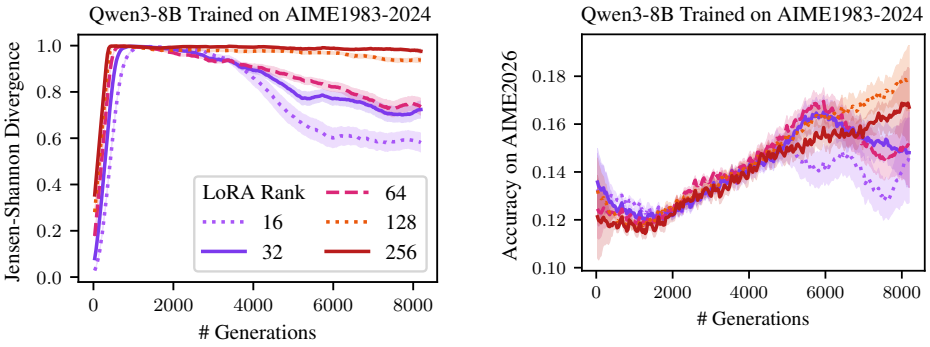



Figure 11: Increasing the rank of the last-layer LoRA heads increases the diversity of the policy ensemble and thus improves the optimization trajectory. Here, the replay buffer is set to $T = 4$.

Moreover, the best optimization trajectories are achieved by ranks of 128 and 256, demonstrating the utility of large ranks. Note that in the rest of the paper, we consistently adopt the LoRA rank 128, which only results in a small computational overhead compared to single-policy GRPO, see Table 2.

### A.6 Bootstrapping contributes to policy diversity

In Figure 12, we investigate whether the Poisson bootstrapping of Algorithm 1 is really required to ensure a diversified policy ensemble. To that end, we ablate the Poisson weights $w_1^i, \dots, w_G^i \sim \text{Poisson}(1)$ drawn independently for each policy $i \in \{1, \dots, n\}$. Note that if $w_j^i = 1 \; \forall i, j$, no bootstrapping would occur and all policy heads would see the same data. In that case, diversification would rely entirely on the random initialization of the $A_i$ matrices ($B_i$ is initialized to zero). In our ablation, we transform the Poisson weights via $(w_j^i)' \leftarrow \lambda w_j^i + 1 - \lambda$. In other words, we take a convex combination between bootstrapping ($\lambda = 1$) and no bootstrapping ($\lambda = 0$). As Figure 12 demonstrates, bootstrapping is a key component that ensures a diversified policy ensemble.

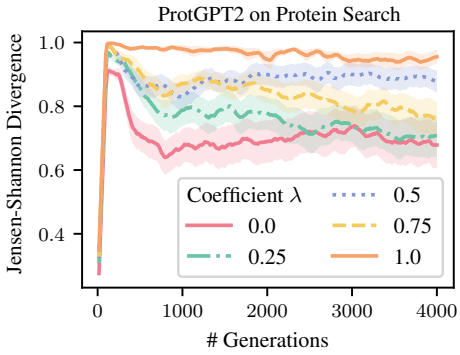



Figure 12: Bootstrapping ensures a diversified ensemble that captures epistemic uncertainty.

### A.7 KL and entropy regularization cannot achieve the targeted exploration of POETS

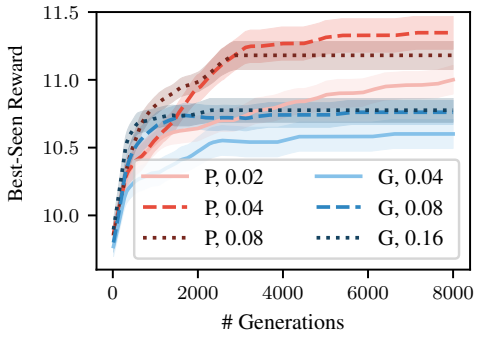
In the main text, we present POETS strictly using Kullback-Leibler (KL) regularization to simplify the exposition. However, in environments with extremely low initial entropy—such as quantum circuit design—it is common practice to introduce an explicit entropy regularization term $\alpha H[\pi]$ to prevent premature policy collapse, where $H[\pi] := \mathbb{E}_{a\sim\pi}[-\log \pi(a)]$ denotes the Shannon entropy.

Mathematically, optimizing an objective with both KL and entropy regularization is equivalent to optimizing a purely KL-regularized objective using a tempered reference distribution and a modified penalty coefficient. Specifically, we can rewrite the combined objective as:

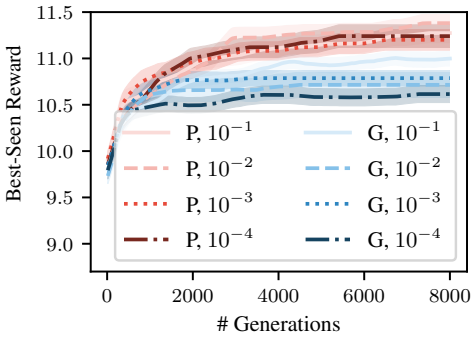
$$\mathbb{E}_{a\sim\pi}[r(a)] - \beta D_{KL}(\pi \parallel \pi_{ref}) + \alpha H[\pi] = \mathbb{E}_{a\sim\pi}[r(a)] - \beta' D_{KL}(\pi \parallel \pi'_{ref}) + C$$

where $\beta' = \beta + \alpha$, $\pi'_{ref}(a) \propto \pi_{ref}(a)^{\frac{\beta}{\beta+\alpha}}$, and $C$ is a constant independent of $\pi$. While this shows that explicit entropy regularization can always be absorbed into the standard POETS formulation, in this section we consider entropy and KL-regularization separately to facilitate interpretable ablations.

We ablate the hyper-parameters $\alpha$ and $\beta$ on the quantum circuit design task using a Qwen3-8B. KL and entropy regularization are a staple of reinforcement learning used to prevent policy collapse. However, neither take into account epistemic uncertainty, i.e., a progression of large initial uncertainty for exploration followed by convergence to the optimal policy for exploitation. As we demonstrate in Figure 13, POETS introduces a calibrated and adaptive exploration-exploitation tradeoff that cannot




(a) P and G denote POETS and GRPO. We ablate the entropy regularization coefficient $\alpha$ for both.




(b) P and G denote POETS and GRPO. We ablate the Kullback-Leibler regularization coefficient $\beta$.

Figure 13: Entropy/KL-regularization does not achieve the well-calibrated exploration-exploitation tradeoff inherent to POETS. This is expected due to its Thompson sampling nature, see Section 4.

be achieved via Kullback-Leibler or entropy regularization. However, such targeted exploration is precisely needed to efficiently identify the global optimum in black-box optimization tasks.

### A.8 Replay buffer for quantum circuit design

Recall from Section 5.1.4 that increasing the replay buffer provides consistent improvements in sample efficiency on the protein search task. We now repeat this experiment for the task of quantum circuit design. As can be seen in Figure 14, given a replay buffer of size 4 or 16, POETS recovers the SoTA sample-efficiency of ToSFiT. Note that ToSFiT leverages a kernel specifically handcrafted for this task, i.e., the ground-truth reward function is linear in the features. In contrast, POETS is completely task agnostic, i.e., does not need any task-dependent engineering.

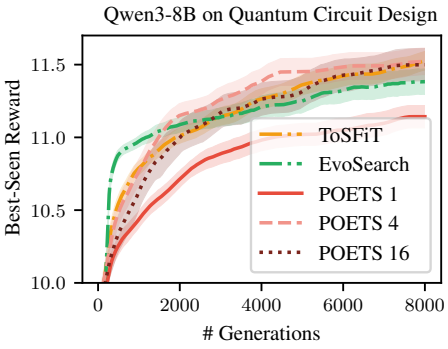



Figure 14: Upsizing the replay buffer results in SoTA sample efficiency for quantum circuit design.

### A.9 Black-box optimization on Therapeutics Data Commons

**Therapeutics Data Commons (TDC) molecular design** explores de novo molecular design under scalarized multi-objective rewards, balancing target-specific binding affinity against pharmacokinetic and synthesizability criteria. Once again, the search space is combinatorially large. We optimize molecular SMILES strings generated by GP-MoLFormer (47M parameters) [Ross et al., 2025] across three distinct therapeutic targets from the TDC benchmark collection [Huang et al., 2021]: (i) *GSK3$\beta$ inhibitor search* (regulates key signaling pathways in neurodegeneration and diabetes), (ii) *JNK3 inhibitor search* (mediates stress-induced neuronal apoptosis), and (iii) *DRD2 ligand search* (GPCR target linked to neuropsychiatric disorders). To enforce drug-likeness constraints, we optimize a composite multiplicative reward: $\text{Reward} = \text{Score}_{\text{Target}} \times \text{QED} \times \text{SA}_{\text{Norm}}$, where the target score is the predicted binding probability from TDC oracles (two pre-trained random forests and an SVM acting as wetlab surrogates), QED is the Quantitative Estimate of Drug-likeness measuring bioavailability, and $\text{SA}_{\text{Norm}}$ is the normalized Synthetic Accessibility score [Ertl and Schuffenhauer, 2009]. The latter two are computed via RDKit [RDKit, 2026]. All reward components are normalized to $[0, 1]$.

Policy gradient methods such as GRPO optimize $\pi_\theta$ according to unbiased gradients of the concave objective $J(\pi, r)$ in Equation (1). In the tabular setting (where $\pi_\theta(a) \propto \exp(\theta_a)$), a small enough decaying learning rate would guarantee convergence to the global optimum. In combinatorial domains, such guarantees require a minuscule learning rate. Indeed, as can be observed empirically in Figure 15, GRPO converges prematurely for acceptable learning rates. In contrast, POETS captures epistemic uncertainty and thus enables accelerated (sample-efficient) learning without premature convergence.

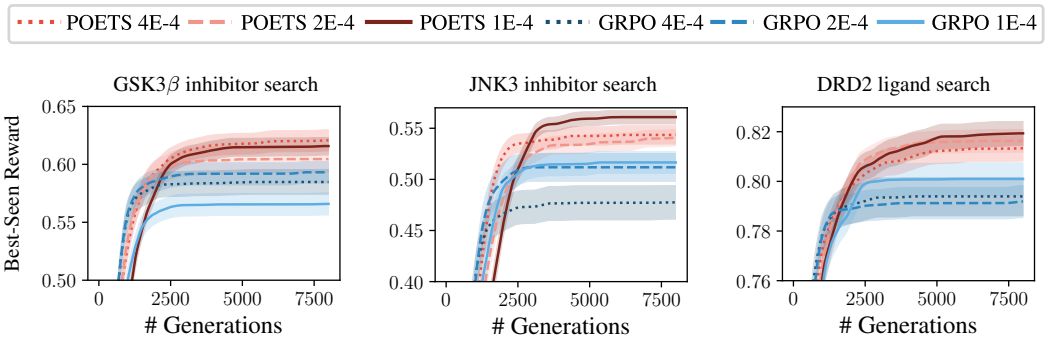



Figure 15: POETS outperforms GRPO on therapeutic molecular design and is more robust to learning rates. Here, only the readout layer is trained with the learning rate indicated to the right.

### A.10 Black-box optimization on bin packing

To validate EvoSearch [Menet et al., 2026b] as a faithful FunSearch-style baseline [Romera-Paredes et al., 2024] for program discovery, we reproduce the bin packing task on Weibull distributions from Romera-Paredes et al. [2024]. By treating our 25 random seeds as independent evolutions ("islands"), EvoSearch manages to replicate the performance of FunSearch with $10\times$ fewer evaluations ($10^5$ versus $10^6$). Note that our experiments utilize the Qwen3-8B model, whereas FunSearch relied on the proprietary Codey (PaLM 2) model, which is estimated to be orders of magnitude larger.

Table 3: Average excess bins of best bin packing heuristic generated. Heuristics are optimized with feedback on the Weibull 5K distribution. The Weibull 10K distribution is an out-of-distribution test.

| **Method** | **# Evaluations** | **Weibull 5K** | **Weibull 10K** |
|---|---|---|---|
| First Fit | N/A | 4.23% | 4.20% |
| Best Fit | N/A | 3.98% | 3.90% |
| FunSearch | $1 \times 10^6$ | 0.68% | 0.32% |
| EvoSearch (1 island) | $4 \times 10^3$ | $0.85\% \pm 0.03\%$ | $0.61\% \pm 0.03\%$ |
| EvoSearch (25 islands) | $1 \times 10^5$ | 0.64% | 0.41% |
| GRPO (1 island) | $4 \times 10^3$ | $1.05\% \pm 0.05\%$ | $1.01\% \pm 0.19\%$ |
| GRPO (25 islands) | $1 \times 10^5$ | 0.76% | 0.56% |
| POETS (1 island) | $4 \times 10^3$ | $1.02\% \pm 0.03\%$ | $0.71\% \pm 0.03\%$ |
| POETS (25 islands) | $1 \times 10^5$ | 0.83% | 0.44% |

While POETS does not establish a new state-of-the-art for finding bin-packing heuristics—trailing slightly behind the specialized EvoSearch—these experiments validate a core capability demonstrated also in our RLVR experiments. Specifically, through its uncertainty-aware optimization procedure, POETS achieves superior out-of-distribution (OOD) transfer compared to standard GRPO. Following the original FunSearch protocol, all methods optimize heuristics exclusively on the Weibull 5K setting, with the Weibull 10K setting serving as the unseen OOD evaluation.

Ultimately, while evolutionary methods excel at specialized local refinement in structured programmatic spaces, POETS provides scalable, uncertainty-guided global search. Combining these complementary strengths presents an exciting direction for future hybrid black-box optimizers.

### A.11 Extended baselines for black-box optimization

To isolate the performance gains of LLM-based optimization and provide a fully transparent picture of the black-box optimization landscape, we evaluate domain-specific classical evolutionary search methods on the Quantum Circuit Design and Protein Search tasks. Specifically, we implement a standard Genetic Algorithm (GA) and a $(1+16)$ Evolution Strategy (ES). In all cases, the substitution and insertion/deletion rates were exhaustively tuned via grid-search for optimal performance.

**Quantum Circuit Design** Qiskit code is parameterized as a sequence of quantum gates. Mutation and crossover are directly implemented at the gate granularity, which ensures that no syntax errors can be generated. Both classical methods initialize from an empty circuit.

- **Genetic Algorithm:** Population = 128, Tournament Size = 3, Substitution Rate / Position = 0.05, Insertion & Deletion Rate / Position = 0.1.
- $(1 + 16)$ **Evolution Strategy:** Population = 128, Substitution Rate / Position = 0.1, Insertion & Deletion Rate / Position = 0.2.

As shown in Table 4, the classical evolutionary baselines perform extremely well on this task, eventually surpassing the LLM-based methods at the 8k evaluation mark. However, it is important to note that classical evolutionary search requires hand-engineered mutation and cross-over operations tailored to the environment. In contrast, LLM-based EvoSearch, GRPO, and POETS remain completely task-agnostic and benefit from the strong priors for improved few-step sample efficiency.

Table 4: Comparison against classical non-LLM baselines on Quantum Circuit Design. All methods except GA and ES use a Qwen3-8B model. We report the mean best-seen reward (BSR) on 25 seeds.

| Method | Time / Batch | BSR @ 250 | BSR @ 500 | BSR @ 1k | BSR @ 2k | BSR @ 4k | BSR @ 8k |
|---|---|---|---|---|---|---|---|
| FIBO | 31.3 s | 10.26 | 10.29 | 10.31 | 10.20 | 10.36 | 10.39 |
| ToSFiT | 13.7 s | 10.34 | 10.50 | 10.81 | 11.04 | 11.27 | 11.58 |
| EvoSearch | 19.4 s | **10.86** | **10.95** | **11.00** | 11.07 | 11.17 | 11.37 |
| GRPO | 7.1 s | 10.11 | 10.11 | 10.11 | 10.16 | 10.16 | 10.31 |
| POETS (T=4) | 9.9 s | 10.31 | 10.45 | 10.77 | 11.15 | 11.39 | 11.52 |
| Genetic Algorithm | 0.0 s | 9.56 | 9.89 | 10.32 | 11.12 | 12.42 | 12.92 |
| $(1 + 16)$ Evolution Strategy | 0.0 s | 9.57 | 10.10 | 10.65 | **11.54** | **12.58** | **12.93** |

**Protein Search** Proteins are parameterized as a sequence of amino acids. Mutation and crossover are implemented at the amino acid granularity, meaning invalid sequences cannot be generated. The initial candidate for the classical methods is a fixed, unconditional sample from ProtGPT2 to match the starting conditions of the LLM-based methods.

- **Genetic Algorithm:** Population = 128, Tournament Size = 3, Substitution Rate / Position = 0.00125, Insertion & Deletion Rate / Position = 0.008.
- $(1 + 16)$ **Evolution Strategy:** Population = 128, Substitution Rate / Position = 0.0025, Insertion & Deletion Rate / Position = 0.004.

As shown in Table 5, the classical methods perform significantly worse on the Protein Search task compared to POETS and (LLM-based) EvoSearch. While EvoSearch paired with a massive 284B frontier model (DeepSeek-V4-Flash) eventually reaches a higher asymptote, POETS applied to a tiny but domain-specific 0.7B model achieves vastly superior sample efficiency early in training while being significantly more computationally efficient. We attribute this to the strong biology priors learnt from pre-training. Note also that evolutionary methods are pure search algorithms; they cannot post-train an LLM. Our main point of comparison therefore remains POETS versus GRPO, as these methods actually update the model's parameters and serve as full reinforcement learning algorithms.

Table 5: Comparison against classical non-LLM baselines and scaled LLM baselines on the Protein Search task. Values denote the mean best-seen reward (BSR) $\pm$ standard error across 25 seeds.

| Method | Model | Params | Hardware | Time | BSR @ 500 | BSR @ 1k | BSR @ 2k | BSR @ 4k |
|---|---|---|---|---|---|---|---|---|
| POETS | ProtGPT2 | 0.7B | 1x H200 | 7 s | $38.5 \pm 2.7$ | $67.5 \pm 4.5$ | $78.5 \pm 4.9$ | $82.3 \pm 4.9$ |
| POETS (T=4) | ProtGPT2 | 0.7B | 1x H200 | 8 s | $\mathbf{70.4 \pm 4.6}$ | $\mathbf{80.4 \pm 5.1}$ | $\mathbf{86.0 \pm 4.9}$ | $92.1 \pm 4.8$ |
| EvoSearch | Qwen3-8B | 8B | 1x H200 | 15 s | $-3.9 \pm 5.2$ | $14.1 \pm 5.0$ | $20.7 \pm 4.4$ | $26.9 \pm 4.0$ |
| EvoSearch | Deepseek-V4-Flash | 284B | 2x H200 | 125 s | $7.5 \pm 2.8$ | $35.8 \pm 4.6$ | $68.8 \pm 3.6$ | $\mathbf{95.8 \pm 3.2}$ |
| Genetic Algorithm | None | 0 | CPU | 0 s | $-10.6 \pm 0.3$ | $1.4 \pm 0.3$ | $16.4 \pm 0.4$ | $35.3 \pm 0.4$ |
| $(1 + 16)$ Evo Strategy | None | 0 | CPU | 0 s | $7.6 \pm 0.5$ | $25.4 \pm 0.4$ | $47.7 \pm 0.4$ | $72.3 \pm 0.7$ |

## B Implementation

To scale POETS to billion-parameter LLMs, we implement the Trunk & Branch architecture using PyTorch's native automatic differentiation and dynamic execution graph. The codebase is designed to avoid $n$ redundant passes through the shared trunk during both training and generation.

**Single forward/backward pass training for the full ensemble** A naive ensemble requires $n$ independent forward & backward passes. In POETS, we execute a single forward pass through the shared trunk. The resulting intermediate hidden states are then branched into the $n$ LoRA heads, which are computed in parallel via batched matrix multiplication. Because all $n$ parallel heads branch off from the same intermediate node, calling `loss.backward()` on the summed (Poisson bootstrapped) ensemble loss natively forces PyTorch's autograd engine to combine and accumulate the gradients from all branches. This accumulated gradient is then propagated through the shared trunk in a single backward pass, decoupling the trunk's computational cost from the ensemble size $n$.

**Batched Thompson sampling generation** During inference, Thompson sampling requires, for each sample, drawing a policy from the approximate posterior, which we implement by selecting one of the $n$ ensemble members. To maximize generation throughput, we sample multiple candidate solutions in a batch, where each sequence is generated by a randomly drawn LoRA branch. To achieve this without instantiating $n$ separate model copies, we dynamically patch the forward pass of the base language modeling head. As shown in Listing 1, we route the hidden states of each sequence in the batch to its assigned LoRA weights $(A_i, B_i)$, allowing the entire batch to be processed concurrently.

```
# 1. Sample an active head index for each sequence in the batch
active_heads = torch.randint(0, n_ensembles, (batch_size,))
# 2. Gather specific LoRA weights for the sampled heads
sel_A = torch.stack([ensembles_A[i] for i in active_heads])
sel_B = torch.stack([ensembles_B[i] for i in active_heads])
# 3. Patch the forward pass of the final linear layer
original_forward = base_projection.forward
def batched_lora_forward(hidden_states):
    # hidden_states shape: (batch_size, seq_len, hidden_dim)
    base_logits = torch.nn.functional.linear(hidden_states, base_projection.weight)
    lora_logits = torch.bmm(
        torch.bmm(hidden_states, sel_A.transpose(1, 2)), sel_B.transpose(1, 2)
    )
    return base_logits + lora_logits
base_projection.forward = batched_lora_forward
# 4. Generate the batch concurrently using the patched head
generated_ids = generate_tokens(prompt_ids, model, group_size)
base_projection.forward = original_forward
```

Listing 1: Dynamic routing for batched Thompson sampling generation.

# C Experimental details

To facilitate reproducibility, we describe the exact setup for each of the experiments presented in Section 5. Additionally, our complete code base is available at `https://github.com/IBM/thompson-sampling-via-fine-tuning-of-llms`, including extensive experimental configuration files and an environment file specifying all Python packages and their versions. Our implementation is based on Pytorch [Paszke et al., 2019] and Huggingface Transformers [Wolf et al., 2020].

To ensure statistical significance, we run all experiments for 25 seeds and report on the mean and standard error. Each experiment was conducted on a single NVIDIA H200 GPU (141GB). The total cost of running all experiments, including preliminary hyper-parameter tuning that did not make it into the paper, amounts to $145\,000$ GPU hours. More detailed information is provided in Table 6.

Table 6: Runtime of a single seed of POETS on each of the main reported benchmarks.

| FAQ | Protein | Quantum Circuit | TDC (all) | AIME | LiveCodeBench v6 |
|---|---|---|---|---|---|
| 2h 0m | 50m | 8h 40m | 5h 0m | 16h 30m | 3d 23h 40m |

Because the large memory footprint of the logits would make memory bandwidth a bottleneck, we chunk the readout layer along the sequence dimension to maximize cache efficiency. By processing smaller blocks, specifically, sizes of 256 for black-box optimization and 512 for reinforcement learning, we keep the data within the cache and drastically reduce slow memory transfers. The strictness of this bandwidth limitation is evident in our quantum circuit design experiments, where using larger chunks of 512 resulted in a slight slowdown despite fitting into VRAM.

### C.1 POETS for black-box optimization (excluding therapeutics data commons benchmarks)

Across all tasks and methods, batched black-box optimization is conducted by generating solutions in groups of size $G = 16$. We consistently adopt an ensemble size of $n = 16$, with a LoRA rank of $r = 128$ for the branches. The LoRA $B$ matrices are initialized to zero, meaning the ensemble members initially behave identically to the shared backbone before diversifying during training via Poisson bootstrapping. GRPO is given one LoRA branch to fine-tune separately from the backbone.

To implement POETS, we bootstrap the data given to the GRPO [Shao et al., 2024] objective (more precisely the variant proposed by Zheng et al. [2025]) with a clipping parameter of $\epsilon = 0.2$. For POETS, we add the KL-regularization terms to the computation of the baseline to ensure consistency with Proposition 1. The shared backbone and the ensemble branches are optimized using the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and no weight decay. Given that generation in these tasks requires broad exploration, the generation temperature is uniformly set to $1.0$ across all methods. We retain the same learning rates independently of the size of the training buffer.

The learning rate of both the backbone and the ensembles is always tuned for GRPO and carried over zero-shot to POETS except for the FAQ response refinement task, where we double the ensemble learning rate to ensure sufficient ensemble diversity. The learning rate of the last-layer LoRA branches can typically be set much larger than that of the backbone while retaining stability. An additional benefit of this decomposition into fast-weights and slow-weights [Kahneman, 2011] is that the shared backbone cannot easily bypass the diversity introduced from bootstrapping the ensemble.

For each specific task, the problem setting dictates the prompt template, system instructions, and precise hyperparameter tuning:

**FAQ refinement** We optimize a Qwen3-8B model with the system prompt set to `"You are a helpful assistant."` to generate responses up to 512 tokens. Thinking mode is disabled and the prompt given to the model for generation is:

```
Write an FAQ response to the question "How do I reset my password?".
```

The reward function consists of the text similarity (cosine similarity score) between the generated response and a hidden ground-truth target text, computed using the `Qwen3-Embedding-0.6B` model.

As a reward function, we consider the alignment to the unknown ground-truth response *To reset your password, go to the login page and click the "Forgot Password" link. Enter your registered email*

*address, and we'll send you instructions to create a new password. Make sure to check your spam folder if you don't see the email within a few minutes.*, computed as the cosine similarity between the Qwen3-Embedding-0.6B [Zhang et al., 2025] embeddings of the candidate and the ground-truth.

To avoid premature entropy collapse of the GRPO baseline, we set the KL-divergence coefficient for all gradient-based methods to $\beta = 1.0 \times 10^{-4}$ against the base reference model set to the LLM at round 0, with no additional entropy regularization ($\alpha = 0.0$). The learning rates are $1.0 \times 10^{-6}$ for the shared backbone and $2.0 \times 10^{-3}/1.0 \times 10^{-3}$ for the ensemble branches (POETS/GRPO).

**Protein search** We optimize the ProtGPT2 model (0.738B parameters), starting with an empty generation prompt (<|*endoftext*|>) to freely sample amino acid sequences up to 512 tokens. The objective is to maximize thermal stability, estimated via the negative thermal instability index [Guruprasad et al., 1990]. Once again, we apply a KL-penalty of $\beta = 1.0 \times 10^{-4}$ without entropy regularization ($\alpha = 0.0$). Learning rates are elevated for this smaller architecture: $1.0 \times 10^{-4}$ for the shared backbone and $1.0 \times 10^{-2}$ for the ensemble branches.

**Quantum circuit design** We use a Qwen3-8B model with system prompt set to `"You are a helpful assistant."` to generate quantum circuits with up to 512 tokens of Qiskit [Vishwakarma et al., 2024]. Thinking is disabled and the model is presented with the following task prompt:

```
Define a 7-qubit Qiskit quantum circuit that can be called with qc =
quantum_circuit(). Include all imports and enclose your code block with
triple backticks in markdown format (```).
```

The reward corresponds to the simulated (negative) energy of the state prepared by the generated circuit under a complex, highly entangled synthetic Pauli Hamiltonian. If the candidate code contains invalid syntax or fails to run, it receives a harsh minimal reward. Because the model has very low initial entropy, small weight updates can easily introduce entropy collapse. Thus, we apply an explicit

Table 7: Hyperparameters for black-box optimization. The learning rates are tuned for GRPO and then directly transferred to POETS except for FAQ Refinement, where the LoRA LR was doubled to increase diversification. We keep a fixed learning rate for all replay buffer sizes. For FAQ refinement, we swept the backbone LR across $1\times10^{-6}$, $2\times10^{-6}$, $5\times10^{-6}$, $1\times10^{-5}$, $2\times10^{-5}$ and the LoRA LR across $5\times10^{-4}$, $1\times10^{-3}$, $2\times10^{-3}$. For protein search, we swept the backbone LR across $5\times10^{-5}, 1\times10^{-4}, 2\times10^{-4}$ and the LoRA LR across $5\times10^{-3}$, $1\times10^{-2}$, $2\times10^{-2}$. For quantum circuit design, we swept the backbone learning rate across $5\times10^{-7}, 1\times10^{-6}, 2\times10^{-6}$ and the LoRA LR across $2\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}$. Appendix A ablates other hyper-parameters.

| Hyperparameter | FAQ Refinement | | Protein Search | | Quantum Circuit Design | |
|---|---|---|---|---|---|---|
| | POETS | GRPO | POETS | GRPO | POETS | GRPO |
| Group Size | 16 | 16 | 16 | 16 | 16 | 16 |
| GRPO Variant | GSPO | GSPO | GSPO | GSPO | GSPO | GSPO |
| Importance Clipping | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| Backbone LR | $10^{-6}$ | $10^{-6}$ | $10^{-4}$ | $10^{-4}$ | $10^{-6}$ | $10^{-6}$ |
| LoRA LR | $2\times10^{-3}$ | $10^{-3}$ | $10^{-2}$ | $10^{-2}$ | $5.0\times10^{-4}$ | $5.0\times10^{-4}$ |
| AdamW $\beta_1$ | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |
| AdamW $\beta_2$ | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 |
| Weight Decay | 0 | 0 | 0 | 0 | 0 | 0 |
| KL-Regularization $\beta$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-3}$ | $10^{-3}$ |
| Entropy Regularization $\alpha$ | 0 | 0 | 0 | 0 | 0.02 | 0.02 |
| Poisson Bootstrapping | True | False | True | False | True | False |
| Ensemble Size | 16 | 1 | 16 | 1 | 16 | 1 |
| LoRA Rank | 128 | 128 | 128 | 128 | 128 | 128 |
| Temperature | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| Max Output Tokens | 512 | 512 | 512 | 512 | 512 | 512 |

entropy regularization penalty of $\alpha = 0.02$, alongside a KL-penalty of $\beta = 1.0 \times 10^{-3}$ across all gradient-based methods. The learning rates are $1.0 \times 10^{-6}$ for the shared backbone and $5.0 \times 10^{-4}$ for the ensemble branches.

### C.1.1 Baseline implementations for black-box optimization (excluding TDC)

**FIBO** [Sutter et al., 2025, Menet et al., 2026a] implements Thompson sampling fully in context by attending to a history of best-scoring previously evaluated candidate solutions and their associated rewards in a chat format. The memory size of the history is set to 128 for FAQ Refinement and Protein Search, but reduced to 64 for Quantum Circuit Design to prevent out-of-memory errors. Thinking is disabled and in-context Bayesian optimization is induced via the following system prompt:

```
You are conducting Bayesian optimization fully in context. You are
provided a history of candidates with associated rewards in a chat
format. Propose a new candidate that might maximize the reward. Your
novel solution must be enclosed using markdown (...). Never repeat a
previous solution. Your search space is over [TASK SPECIFIC SPACE].
```

The history is initially seeded with an example response (for FAQ and Protein Search) or an empty quantum circuit (for Quantum Circuit Design).

**EvoSearch** [Romera-Paredes et al., 2024] performs evolutionary search in context through instruction following. It maintains a population of size 128 and selects two parent candidates in each round via tournament selection with a tournament size of 3. The language model acts as the crossover and mutation operator prompted by:

```
You are conducting evolutionary search in context. You are provided two
candidates. Propose a new distinct candidate by combining the two
provided candidates and mutating the result. Your novel candidate must
be enclosed using markdown (...). Never repeat previous candidates.
Your search space is over [TASK SPECIFIC SPACE].
```

Chain-of-thought reasoning is disabled. Similar to FIBO, the initial population is seeded with an example response or an empty circuit to provide the desired format to the model.

**ToSFiT** [Menet et al., 2026b] fine-tunes the language model toward a variational Thompson sampling policy based on a Gaussian process reward surrogate. Across all experiments, an exploration bonus of 4.0 is applied to the prior amplitude inferred via marginal likelihood maximization, and the noise-to-amplitude ratio is set to 0.01. The marginal likelihood maximization uses 4 warmup steps. For the Gaussian process feature map, FAQ Refinement uses the primary 256 dimensions of the `Qwen3-Embedding-0.6B` [Zhang et al., 2025] model, Protein Search uses the mean token embeddings from ProtGPT2 projected to the unit sphere (1280 dimensions), and Quantum Circuit Design uses 211 two-qubit Pauli observables. Following GRPO and POETS, we apply a KL-penalty of $\beta = 1.0 \times 10^{-4}$ for FAQ and Protein Search, and $\beta = 1.0 \times 10^{-3}$ for Quantum Circuit Design. The learning rates are tuned individually, reaching $1.0 \times 10^{-6}$, $1.0 \times 10^{-4}$, and $2.0 \times 10^{-6}$, respectively.

## C.2 POETS on therapeutics data commons (TDC) molecular design benchmarks

We utilize the `GP-MoLFormer-Uniq` [Ross et al., 2025] molecule generator model (47M parameters pre-trained autoregressively on 1.1 billion SMILES strings) to unconditionally sample SMILES strings up to 200 tokens. The objective in our TDC benchmarks is to navigate highly non-linear, multi-objective search landscapes across three distinct biological profiles modeled by ExCAPE-DB trained wetlab surrogates: (i) **GSK3**$\beta$, a kinase regulating glycogen metabolism and a major target for Alzheimer's and diabetes (Random Forest oracle); (ii) **JNK3**, a MAPK mediating stress-induced neuronal apoptosis targeted for neuroprotective therapies (Random Forest oracle); and (iii) **DRD2**, a GPCR targeted by antipsychotic medications for schizophrenia and bipolar disorder (SVM oracle).

For each target, the agent optimizes a composite multiplicative reward that balances target-specific binding probability against drug-likeness and synthesizability:

$$\text{Reward} = \text{Score}_{\text{Target}} \times \text{QED} \times \text{SA}_{\text{Norm}} \tag{6}$$

Here, $\text{Score}_{\text{Target}} \in [0, 1]$ is the oracle binding probability, QED $\in [0, 1]$ assesses drug-likeness (e.g., molecular weight, $\log P$) via RDKit [RDKit, 2026] and is essential to gauge bioavailability, and $\text{SA}_{\text{Norm}} \in [0, 1]$ normalizes the Synthetic Accessibility score [Ertl and Schuffenhauer, 2009] via $\frac{10.0-\max(1.0,\min(10.0,\text{SA}))}{9.0}$, measuring the difficulty of synthesizing the drug in a lab or in production.

Across all therapeutic targets, batched black-box optimization is conducted by generating SMILES strings in groups of size $G = 16$. We consistently adopt an ensemble size of $n = 64$, with a LoRA rank of $r = 768$ for the branches (equal to the hidden layer size of the generator model to provide full-rank adaptation capability). Noticing that both POETS and vanilla GRPO achieve better performance by only fine-tuning the final readout layer, we freeze the (shared) model trunk. As in all other tasks, the LoRA $B$ matrices are initialized to zero, meaning the ensemble members initially behave identically to the shared backbone before diversifying during training via Poisson bootstrapping. GRPO is given one "LoRA" branch to fine-tune separately from the frozen backbone.

GRPO (including POETS) is stabilized with a clipping parameter of $\epsilon = 0.2$ and a KL-divergence penalty of $\beta = 0.1$ against the base reference model without additional entropy regularization ($\alpha = 0.0$). We adopt a replay buffer containing the last 16 batches and enable importance sampling in the style of GSPO [Zheng et al., 2025]. The active LoRA branches are optimized using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and no weight decay. The generation temperature is uniformly set to 1.0. Optimization runs for 8000 interactions/generations with a batch size of 16.

Table 8: Hyperparameters for TDC Molecular Design. The shared trunk is completely frozen, and only the full-rank last-layer adapter is optimized. On these tasks, with `GP-MoLFormer-Uniq` as generator, updating the backbone model parameters deteriorates performance of GRPO and POETS.

| Hyperparameter | TDC Targets (All) | |
|---|---|---|
| | POETS | GRPO |
| Group Size | 16 | 16 |
| GRPO Variant | GSPO | GSPO |
| Importance Clipping | 0.2 | 0.2 |
| Backbone LR | 0.0 | 0.0 |
| LoRA LR | ablated | ablated |
| AdamW $\beta_1$ | 0.9 | 0.9 |
| AdamW $\beta_2$ | 0.999 | 0.999 |
| Weight Decay | 0 | 0 |
| KL-Regularization $\beta$ | 0.1 | 0.1 |
| Entropy Regularization $\alpha$ | 0 | 0 |
| Poisson Bootstrapping | True | False |
| Ensemble Size | 64 | 1 |
| LoRA Rank | 768 | 768 |
| Temperature | 1.0 | 1.0 |
| Max Output Tokens | 200 | 200 |

### C.3 POETS for bin packing

We evaluate POETS on the online 1D bin packing problem following the programmatic heuristic discovery setup introduced by FunSearch [Romera-Paredes et al., 2024]. In online bin packing, a stream of items of varying sizes arrives sequentially and must be immediately assigned to one of an unlimited set of bins with capacity $C = 100$ without knowledge of future items. The objective is to discover a Python function named `heuristic(item_size: float, bin_capacities: np.ndarray) -> np.ndarray` that computes priority scores for all currently feasible bins (bins with remaining capacity at least equal to `item_size`), with the item placed in the bin receiving the maximum score.

**Dataset and evaluation** Item sizes are drawn from a Weibull distribution: item sizes are generated as $45 \times \text{Weibull}(3)$, rounded to the nearest integer, and clipped to the range $[1, 100]$ [Romera-Paredes et al., 2024]. Following FunSearch, heuristics are optimized against feedback on instances with $5{,}000$ items (**Weibull 5K**), while generalization is evaluated out-of-distribution (OOD) on instances with $10{,}000$ items (**Weibull 10K**). Each evaluation averages over $5$ simulated packing experiments. The performance metric is the *average excess bins* relative to the theoretical $L_2$ lower bound [Romera-Paredes et al., 2024]:

$$\text{Excess Bins} = \frac{\text{Bins Used}}{\text{Bins}_{L_2}} - 1.0. \tag{7}$$

The environment reward is the negative excess bins, $r = -\text{Excess Bins}$ (higher is better, with an ideal optimum of $0.0$). Unexecutable scripts, syntax errors, or functions exceeding execution timeouts in the isolated sandbox return `NaN` and receive an invalid generation penalty.

**Prompting and generation** We optimize the `Qwen3-8B` model [Qwen Team, 2025]. Thinking mode is disabled, and the generation limit is set to $512$ new tokens with sampling temperature $1.0$. The system prompt is:

```
You are an expert algorithm designer tasked with discovering novel,
innovative, and non-obvious online bin packing heuristics. Avoid
plain textbook heuristics like standard Best-Fit, First-Fit, or
Worst-Fit. Design a creative mathematical priority score function.
/no_think
```

The user prompt provided to the model is:

```
Define a Python function named `heuristic(item_size: float,
bin_capacities: np.ndarray) -> np.ndarray` that assigns novel
and creative priority scores to bin capacities for online bin
packing to minimize excess bins. Avoid outputting standard
textbook heuristics like plain Best-Fit or First-Fit. Output
only the function definition and body inside triple backticks
(```python ... ```)/no_think.
```

**Optimization and hyperparameters** Optimization runs for $4{,}000$ interactions in groups of size $G = 32$, corresponding to $125$ policy gradient updates. We operate in the online setting without a replay buffer ($T = 1$). Both GRPO and POETS optimize the GSPO loss [Zheng et al., 2025] with clipping parameter $\epsilon = 0.2$, group advantage standardization, importance sampling, and length rescaling. We apply a KL-divergence penalty of $\beta = 1.0 \times 10^{-3}$ against the base reference model without entropy regularization ($\alpha = 0.0$). Validation on Weibull 10K is performed every $4$ rounds.

For POETS, we maintain an ensemble of $n = 16$ policy heads with LoRA rank $r = 128$, initialized with $B_i = 0$. Epistemic uncertainty is captured via online Poisson bootstrapping with weights $w \sim \text{Poisson}(1)$. For this task, the shared backbone language head is frozen (`freeze_backbone_head: True`), confining policy differentiation entirely to the LoRA branches. For GRPO, we fine-tune the backbone head directly ($n = 1$) without LoRA heads and without bootstrapping. The backbone learning rate was swept across $1.0 \times 10^{-6}$, $2.0 \times 10^{-6}$, and $4.0 \times 10^{-6}$ for GRPO; the optimal rate ($2.0 \times 10^{-6}$) was transferred to POETS. For POETS, the LoRA ensemble learning rate was swept across $2.0 \times 10^{-5}$, $4.0 \times 10^{-5}$, and $8.0 \times 10^{-5}$, with $4.0 \times 10^{-5}$ achieving the best performance. Both components are trained using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and zero weight decay. Table 9 summarizes all hyperparameters.

**EvoSearch baseline setup** For the EvoSearch baseline [Romera-Paredes et al., 2024, Menet et al., 2026b], we maintain an evolutionary population of $128$ candidate heuristics with a tournament selection size of 2. The population is seeded with an initial candidate implementing First-Fit/Worst-Fit: `def heuristic(item_size, bin_capacities): return -bin_capacities`. The LLM acts as the crossover and mutation operator prompted to combine two parent heuristics into an improved child heuristic. Following the multi-island design of FunSearch, our 25 seeds act as 25 independent evolutionary islands.

Table 9: Hyperparameters for Online Bin Packing Heuristic Search. The backbone learning rate was optimized on GRPO and transferred to POETS. In POETS, the backbone language head is frozen and only the LoRA adapter branches are updated.

| Hyperparameter | Bin Packing (Qwen3-8B) | |
|---|---|---|
| | POETS | GRPO |
| Group Size $G$ | 32 | 32 |
| Number of Interactions | 4000 | 4000 |
| Replay Buffer Size $T$ | 1 | 1 |
| GRPO Variant | GSPO | GSPO |
| Importance Clipping $\epsilon$ | 0.2 | 0.2 |
| Backbone LR | $2.0\times10^{-6}$ | $2.0\times10^{-6}$ |
| LoRA LR | $4.0\times10^{-5}$ | 0.0 |
| Freeze Backbone Head | True | False |
| AdamW $\beta_1$ | 0.9 | 0.9 |
| AdamW $\beta_2$ | 0.999 | 0.999 |
| Gradient Clip Norm | 1.0 | 1.0 |
| Weight Decay | 0 | 0 |
| KL-Regularization $\beta$ | $10^{-3}$ | $10^{-3}$ |
| Entropy Regularization $\alpha$ | 0 | 0 |
| Poisson Bootstrapping | True | False |
| Ensemble Size $n$ | 16 | 1 |
| LoRA Rank $r$ | 128 | 128 |
| Temperature | 1.0 | 1.0 |
| Max Output Tokens | 512 | 512 |

### C.4 POETS for reinforcement learning

Across all tasks, batched optimization is conducted by sampling $4$ prompts and generating candidate solutions in groups of size $G = 8$, i.e., a batch of 32 responses is formed and evaluated. To avoid out-of-memory during training, we vectorize computation across groups but treat each prompt as a separate sequential mini-batch. For all experiments, we adopt an ensemble size of $n = 16$, with a LoRA rank of $r = 128$ for the branches in POETS. As before, the LoRA $B$ matrices are initialized to zero. GRPO is fine-tuned directly without any additional LoRA branches. To implement POETS, we bootstrap the GRPO objective (with both importance sampling and length rescaling enabled) and retain its clipping parameter of $\epsilon = 0.2$. We apply a consistent KL-divergence penalty coefficient of $\beta = 1.0 \times 10^{-3}$ against the base reference model across all methods and include it in the baseline calculation. We do not use additional entropy regularization ($\alpha = 0.0$). The shared backbone and the ensemble branches are optimized using the AdamW optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay, and a gradient norm clipping with maximal value $1.0$.

For both GRPO and POETS, the backbone learning rate is $1.0 \times 10^{-5}$, except for the off-policy variants ($T = 4$) trained on AIME1983-2024, where a backbone learning rate of $2.5 \times 10^{-6}$ is adopted. In all cases the learning rate was optimized for GRPO via grid search and retained for POETS without further tuning. For POETS, the learning rate of the ensemble branches is tuned via grid search and set to $5.0 \times 10^{-4}$ except for the off-policy variants ($T = 4$) trained on AIME1983-2024, where it is set to $1.25 \times 10^{-4}$, and for LiveCodeBench v6, where we directly ablate on it in our main results. As for black-box optimization, we find that the large learning rate of the LoRA heads is crucial to achieve calibrated policy diversification.

We intentionally use two complementary RLVR experimental designs. For AIME, GRPO is run with its standard configuration (no separate LoRA head), representing the strongest published single-policy baseline; POETS extends this with a Trunk & Branch architecture. For LiveCodeBench, we give both GRPO and POETS an identical Trunk & Branch architecture (LoRA rank 128 on the readout layer,

Table 10: Hyperparameters for RLVR on AIME. The backbone learning rates are tuned for GRPO and then directly transferred to POETS. Since LoRA is not used for GRPO, the LoRA learning rates are tuned for POETS. More precisely, for AIME 2026 and AIME 1983-2024 (T=1), we swept the backbone LR across $5\times10^{-6}, 1\times10^{-5}, 2\times10^{-5}$ and the LoRA LR across $2\times10^{-4}$, $5\times10^{-4}$, $1\times10^{-3}$. For AIME 1983-2024 (T=4), we swept the backbone LR across $1\times10^{-6}$, $2\times10^{-6}$, $2.5\times10^{-6}$, $3\times10^{-6}$, $4\times10^{-6}$ and upon noticing that the optimal learning rate is $4$ times smaller than for $T = 1$, we also decreased the LoRA LR by a factor $4$. Appendix A ablates other hyper-parameters.

| Hyperparameter | AIME 2026 (T=1,4) & AIME 1983-2024 (T=1) | | AIME 1983-2024 (T=4) | |
|---|---|---|---|---|
| | POETS | GRPO | POETS | GRPO |
| Question Batch Size | 4 | 4 | 4 | 4 |
| Group Size | 8 | 8 | 8 | 8 |
| GRPO Variant | GSPO | GSPO | GSPO | GSPO |
| Importance Clipping | 0.2 | 0.2 | 0.2 | 0.2 |
| Backbone LR | $10^{-5}$ | $10^{-5}$ | $2.5\times10^{-6}$ | $2.5\times10^{-6}$ |
| LoRA LR | $5\times10^{-4}$ | N/A | $1.25\times10^{-4}$ | N/A |
| AdamW $\beta_1$ | 0.9 | 0.9 | 0.9 | 0.9 |
| AdamW $\beta_2$ | 0.999 | 0.999 | 0.999 | 0.999 |
| Gradient Clip Norm | 1.0 | 1.0 | 1.0 | 1.0 |
| Weight Decay | 0 | 0 | 0 | 0 |
| KL-regularization $\beta$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ |
| Entropy regularization $\alpha$ | 0 | 0 | 0 | 0 |
| Poisson Bootstrapping | True | False | True | False |
| Ensemble Size | 16 | 1 | 16 | 1 |
| LoRA Rank | 128 | 0 | 128 | 0 |
| Temperature | 1.0 | 1.0 | 1.0 | 1.0 |
| Max Output Tokens | 2048 | 2048 | 2048 | 2048 |

same dual LR scheme), explicitly isolating the contribution of Poisson bootstrapping from architectural changes and learning dynamics. The LiveCodeBench experiment thus serves as the controlled ablation, and in fact demonstrates that GRPO does not benefit from two separate learning rates.

**AIME** We optimize a Qwen3-8B model with temperature $1.0$ and generation limit set to 2048 new tokens. Thinking mode is disabled, and the model is prompted with the following system instructions:

```
You are an expert mathematician. You must reason step-by-step and place
your final answer within \boxed{}. Be concise.
```

**AIME 2026** Models are trained over a total of $4800$ interactions, equating to $150$ gradient updates and $20$ passes through the dataset. In the spirit of classical contextual bandits where there is no train-test split, the model is also validated on AIME 2026.

**AIME 1983-2024** Models are trained over a total of $8192$ interactions, which corresponds to $256$ gradient updates and $1.1$ passes through the dataset. We report accuracy both on the training set (i.e., the observed rewards) as well as on a separate held-out validation set (AIME2026).

**LiveCodeBench v6** We optimize a Qwen3-8B model with generation limit set to $1536$ new tokens. Thinking mode is disabled, and the model is prompted with the following system instructions:

```
You are an expert programmer. You must place your final answer within
a markdown python block (```python ... ```). Be concise.
```

We follow [DeepSeek-AI, 2025] and decrease the validation temperature to $0.6$ while retaining the training temperature at $1.0$ for rollout generation. To improve training efficiency and GPU utilization, we train with a replay buffer of size T=4.

Table 11: Hyperparameters for RLVR on LiveCodeBench. The backbone learning rates are tuned for GRPO and then directly transferred to POETS. More precisely, we swept the backbone LR across $5\times10^{-6}, 1\times10^{-5}, 2\times10^{-5}$, with $1\times10^{-5}$ achieving the highest accuracy on the private tests.

| Hyperparameter | LiveCodeBench v6 (T=4) | |
|---|---|---|
| | POETS | GRPO |
| Question Batch Size | 4 | 4 |
| Group Size | 8 | 8 |
| GRPO Variant | GSPO | GSPO |
| Importance Clipping | 0.2 | 0.2 |
| Backbone LR | $10^{-5}$ | $10^{-5}$ |
| LoRA LR | ablated | ablated |
| AdamW $\beta_1$ | 0.9 | 0.9 |
| AdamW $\beta_2$ | 0.999 | 0.999 |
| Gradient Clip Norm | 1.0 | 1.0 |
| Weight Decay | 0 | 0 |
| KL-regularization $\beta$ | $10^{-3}$ | $10^{-3}$ |
| Entropy regularization $\alpha$ | 0 | 0 |
| Poisson Bootstrapping | True | False |
| Ensemble Size | 16 | 1 |
| LoRA Rank | 128 | 128 |
| Training Temperature | 1.0 | 1.0 |
| Validation Temperature | 0.6 | 0.6 |
| Max Output Tokens | 1536 | 1536 |

### C.5 Why extended reasoning mode is disabled

We disable reasoning mode in the considered LLMs for three critical reasons:

1. **Credit assignment:** The credit assignment of GRPO on long sequences is highly sparse. As a result, disabling extended reasoning mode leads to more stable optimization trajectories for both GRPO and POETS in black-box optimization.
2. **RL learns CoT:** In model post-training (RLVR), chain-of-thought is typically learnt via reinforcement learning. By disabling reasoning mode, the RLVR experiments become more meaningful and follow standard practice in the literature. Note that disabling thinking mode does not prevent the model from producing step-by-step reasoning in its response, but rather suppresses the extended pre-response thinking tokens. The RLVR setup therefore still allows CoT-style reasoning to emerge from RL pressure.
3. **Compute cost:** Our total compute investment amounts to 145,000 H200 GPU hours. Enabling reasoning mode adds another order of magnitude to this cost. POETS aims for computational efficiency; forcing CoT expands the compute gap against already expensive baselines like EvoSearch. Indeed, enabling reasoning mode for EvoSearch in preliminary experiments did not result in improved best-seen reward, resulted in frequent out-of-memory crashes, and reduced generation speed by at least one order of magnitude.

# D Proofs

## D.1 Proof sketches

*Proof sketch of Proposition 1.* The proof begins by expanding the reward difference $r(a-\rho)-r_\pi(a-\rho)$ and showing that the intractable normalization constant $\log Z$ naturally cancels out, allowing the difference to be rewritten strictly in terms of the soft rewards $\tilde{r}_\pi(a) - \tilde{r}_\pi(\rho)$. By differentiating the squared loss with respect to the model parameters $\theta$, the chain rule produces the scaled gradient of the log-policy offset by the expected baseline. Finally, taking the outer expectation over the (empirical) baseline distribution $\rho$ eliminates the subtracted baseline term entirely because the average advantage evaluates to zero, perfectly recovering the final average gradient form. □

*Proof sketch of Theorem 1.* The proof heavily leverages the core Thompson Sampling identity, which establishes that, conditioned on the history, the expected objective value of the optimal policy under the true reward is equal to the expected objective value of the sampled policy under the sampled reward. Substituting this identity into the expected single-step soft regret, the KL-divergence and entropy regularization terms perfectly cancel each other out because they depend only on the policy and are independent of the reward functions. This cancellation reduces the expected regularized regret purely to the expected reward estimation error at the evaluation points, which is then bounded by $\mathcal{O}(\sqrt{T\gamma_T})$ using standard Gaussian process confidence bounds from prior works. □

*Proof sketch of Proposition 2.* Under the perfect calibration assumption $E^2 = V \cdot Z^2$ for actual error $E$, ensemble variance $V$, and standard normal noise $Z$. The expected squared noise is 1, which conveniently reduces the covariance $\text{Cov}(V, E^2)$ strictly to $\text{Var}(V)$. However, the variance of the actual squared error, $\text{Var}(E^2)$, depends on the fourth moment of the Gaussian noise (which is 3). This causes $\text{Var}(E^2)$ to expand to $3\text{Var}(V) + 2\mathbb{E}[V]^2$. Substituting these terms into the Pearson correlation definition results in the desired bound on the denominator: $\sqrt{3 + 2\mathbb{E}[V]^2/\text{Var}(V)} \geq \sqrt{3}$. □

## D.2 Formal proofs

**Proposition** (Restatement of Proposition 1)**.** *Let $\pi$ be parameterized by $\theta$ and define the KL-regularized reward $\tilde{r}_\pi(a) := r(a) + \beta \cdot (\log \pi_{ref}(a) - \log \pi(a))$. Then the gradients of the* POETS *loss $L^\rho_{poets}$ are*

$$\nabla_\theta L^\rho_{poets}(a, \pi) = -2\beta \cdot \tilde{r}_\pi(a-\rho)(\nabla_\theta \log \pi(a) - \mathbb{E}_{a'\sim\rho}[\nabla_\theta \log \pi(a')]) \tag{8}$$

$$\mathbb{E}_{a\sim\rho}[\nabla_\theta L^\rho_{poets}(a, \pi)] = -2\beta \cdot \mathbb{E}_{a\sim\rho}\big[\tilde{r}_\pi(a-\rho)\nabla_\theta \log \pi(a)\big]. \tag{9}$$

*Proof of Proposition 1.* We start off by expanding the definition of $r(a-\rho)$ and $r_\pi(a-\rho)$ and canceling $\log Z$:

$$\begin{aligned} r(a-\rho) - r_\pi(a-\rho) &= r(a) - r_\pi(a) - (r(\rho) - r_\pi(\rho)) \\ &= r(a) - (\beta \log \pi(a) - \beta \log \pi_{ref}(a) + \beta \log Z) \\ &\quad - \mathbb{E}_{a'\sim\rho}[r(a') - (\beta \log \pi(a') - \beta \log \pi_{ref}(a') + \beta \log Z)] \\ &= \tilde{r}_\pi(a) - \tilde{r}_\pi(\rho) = \tilde{r}_\pi(a-\rho). \end{aligned}$$

Using this identity, we next calculate the derivative of $L^\rho_{poets}(a, \pi)$ step by step:

$$\begin{aligned} \nabla_\theta L^\rho_{poets}(a, \pi) &= \nabla_\theta \big(r(a-\rho) - r_\pi(a-\rho)\big)^2 = -2(r(a-\rho) - r_\pi(a-\rho))\nabla_\theta r_\pi(a-\rho) \\ &= -2\tilde{r}_\pi(a-\rho)\nabla_\theta r_\pi(a-\rho) \\ &= -2\beta\tilde{r}_\pi(a-\rho)(\nabla_\theta \log \pi(a) - \mathbb{E}_{a'\sim\rho}[\nabla_\theta \log \pi(a')]) \end{aligned}$$

For the second result, we take the outer expectation over $\rho$, i.e., the reward difference is taken with respect to the average reward on the entire dataset (or batch) $\{a_i\}$, formally described by the empirical distribution $\rho(a) = \frac{1}{G}\sum_{i=1}^G \delta_{a=a_i}$. The result then follows from $\mathbb{E}_{a\sim\rho}[\tilde{r}_\pi(a-\rho)\mathbb{E}_{a'\sim\rho}[\nabla_\theta \log \pi(a')]] = \mathbb{E}_{a\sim\rho}[\tilde{r}_\pi(a-\rho)] \cdot \mathbb{E}_{a'\sim\rho}[\nabla_\theta \log \pi(a')] = 0 \cdot \mathbb{E}_{a'\sim\rho}[\nabla_\theta \log \pi(a')] = 0$ □

**Theorem** (Detailed version of Theorem 1)**.** *Suppose a Gaussian prior over the true reward vector $r$ with standard deviation $\leq 1$. Then, KL-regularized Thompson sampling (observing $r$ at $a_t \sim \pi_t$ for $\pi_t := \arg\max_\pi J(\pi, \tilde{r}_t)$ where $\tilde{r}_t \sim \mathbb{P}[\, r \mid \mathcal{H}_{t-1}]$ given observation history $\mathcal{H}_{t-1}$) has regret*

$$\textstyle\sum_{t=1}^{T} \mathbb{E}_{r,\pi_t}[\max_\pi J(\pi, r) - J(\pi_t, r)] \leq \zeta \sqrt{C_\eta} \cdot \sqrt{T\gamma_T} \in \mathcal{O}(\sqrt{T\gamma_T}).$$

*$\zeta := 1 + \sqrt{2\log(2 \cdot |\mathcal{A}|) + 2}$, $C_\eta := 2/\ln(1 + \eta^{-2})$ where $\eta$ upper bounds the standard deviation of independent additive Gaussian observation noise, and $\gamma_T$ is the maximal information gain [Srinivas et al., 2010] of the Gaussian prior over $r$. If $r$ is linear in $d$ features, then $\gamma_T \in \mathcal{O}(d \log T)$.*

*Proof of Theorem 1.* Let $r$ be the true reward, let $\mathcal{H}_{t-1}$ denote the observation history up to time $t$, and let $\tilde{r}_t \sim \mathbb{P}[\, r \mid \mathcal{H}_{t-1}]$. Define the optimal policy $\pi^* := \arg\max_\pi J(\pi, r)$. By definition, KL-regularized Thompson sampling draws from $\pi_t := \arg\max_\pi J(\pi, \tilde{r}_t)$. Now, note that $\pi^*, r \overset{d}{=} \pi_t, \tilde{r}_t$ given $\mathcal{H}_{t-1}$. As a result, the (soft) reward conditioned on $\mathcal{H}_{t-1}$ is also identically distributed:

$$\mathbb{E}[J(\pi^*, r) \mid \mathcal{H}_{t-1}] = \mathbb{E}[J(\pi_t, \tilde{r}_t) \mid \mathcal{H}_{t-1}].$$

Defining $\Delta_t := J(\pi^*, r) - J(\pi_t, r)$, we substitute this identity into the expected single-step regret:

$$\mathbb{E}[\Delta_t \mid \mathcal{H}_{t-1}] = \mathbb{E}[J(\pi^*, r) - J(\pi_t, r) \mid \mathcal{H}_{t-1}] = \mathbb{E}[J(\pi_t, \tilde{r}_t) - J(\pi_t, r) \mid \mathcal{H}_{t-1}].$$

We expand the objective function $J$ for the same policy $\pi_t$ under both reward functions $\tilde{r}_t$ and $r$:

$$\begin{aligned}\mathbb{E}[\Delta_t \mid \mathcal{H}_{t-1}] = \mathbb{E}\Big[&\big(\textstyle\sum_a \tilde{r}_t(a)\pi_t(a) - \beta D_{KL}(\pi_t||\pi_{ref})\big) \\ &- \big(\textstyle\sum_a r(a)\pi_t(a) - \beta D_{KL}(\pi_t||\pi_{ref})\big) \mid \mathcal{H}_{t-1}\Big].\end{aligned}$$

Note that here $\pi_t$ is itself a random quantity that depends on $\tilde{r}_t$. Because the KL-divergence and entropy terms depend solely on the policy $\pi_t$ and are independent of the reward, they cancel out:

$$\mathbb{E}[\Delta_t \mid \mathcal{H}_{t-1}] = \mathbb{E}\big[\textstyle\sum_a (\tilde{r}_t(a) - r(a)) \cdot \pi_t(a) \mid \mathcal{H}_{t-1}\big].$$

The expected regularized regret simplifies entirely to the expected reward estimation error at evaluation points. We next bound this term using standard confidence bounds over $T$ steps, in particular via the maximal information gain [Srinivas et al., 2010]. More precisely, according to Lemma 1, it holds that

$$\mathbb{E}[\Delta_t \mid \mathcal{H}_{t-1}] \leq \mathbb{E}\big[\textstyle\sum_a |\tilde{r}_t(a) - r(a)| \cdot \pi_t(a) \mid \mathcal{H}_{t-1}\big] \leq \zeta \sqrt{\mathbb{E}[\textstyle\sum_a \sigma_t^2(a) \cdot \pi_t(a) \mid \mathcal{H}_{t-1}]}.$$

Here, the last step used that under Gaussian priors $r, \tilde{r}_t$ are i.i.d. element-wise $\sigma_t$-subgaussian given observation history $\mathcal{H}_{t-1}$, where $\sigma_t(a)$ is the posterior standard deviation for the reward of action $a$. Next, we apply Jensen's inequality and Cauchy-Schwarz across the sequence dimension to obtain

$$\begin{aligned}&\textstyle\sum_{t=1}^{T} \mathbb{E}_{r,\pi_t}[\max_\pi J(\pi, r) - J(\pi_t, r)] = \sum_{t=1}^{T} \mathbb{E}[\Delta_t] = \sum_{t=1}^{T} \mathbb{E}[\mathbb{E}[\Delta_t \mid \mathcal{H}_{t-1}]] \\ \leq &\textstyle\sum_{t=1}^{T} \mathbb{E}\big[\zeta \sqrt{\mathbb{E}[\sum_a \sigma_t^2(a) \cdot \pi_t(a) \mid \mathcal{H}_{t-1}]}\big] \leq \zeta \sum_{t=1}^{T} 1 \cdot \sqrt{\mathbb{E}[\textstyle\sum_a \sigma_t^2(a) \cdot \pi_t(a)]} \\ \leq &\zeta \sqrt{T \cdot \textstyle\sum_{t=1}^{T} \mathbb{E}[\sum_a \sigma_t^2(a) \cdot \pi_t(a)]} = \zeta \sqrt{T \cdot \mathbb{E}[\textstyle\sum_{t=1}^{T} \sigma_t^2(a_t)]} \text{ where } a_t \sim \pi_t.\end{aligned}$$

To finish the proof, we apply Lemma 2, which results in the familiar $\mathcal{O}(\sqrt{T\gamma_T})$ scaling. Note that by rescaling the reward, the technical condition $\sigma_1(a) \leq 1\ \forall a$ holds without loss of generality and can be absorbed as a constant in $\mathcal{O}$ notation. For bounds on $\gamma_T$, we refer to Srinivas et al. [2010]. □

**Proposition** (Restatement of Proposition 2)**.** *Let $V > 0$ be a random variable representing the predicted variance, and let $E$ be the signed error, such that $E^2$ is the squared error. Assume the model is perfectly calibrated under Gaussian assumptions, i.e., $E^2 = V \cdot Z^2$ for $Z \sim \mathcal{N}(0, 1)$ independent of $V$. Then, the Pearson correlation $\rho(V, E^2)$ is strictly bounded from above:*

$$\rho(V, E^2) \leq \frac{1}{\sqrt{3}} \approx 0.577$$

*Proof.* The Pearson correlation is defined as:

$$\rho(V, E^2) = \frac{\mathrm{Cov}(V, E^2)}{\sqrt{\mathrm{Var}(V)\mathrm{Var}(E^2)}}$$

Because $Z \sim \mathcal{N}(0,1)$, its relevant moments are $\mathbb{E}[Z^2] = 1$ and $\mathbb{E}[Z^4] = 3$. Since $V$ and $Z$ are independent, we first compute the expectations required for the covariance:

$$\mathbb{E}[E^2] = \mathbb{E}[VZ^2] = \mathbb{E}[V]\mathbb{E}[Z^2] = \mathbb{E}[V]$$

$$\mathbb{E}[VE^2] = \mathbb{E}[V^2Z^2] = \mathbb{E}[V^2]\mathbb{E}[Z^2] = \mathbb{E}[V^2]$$

$$\text{Cov}(V, E^2) = \mathbb{E}[VE^2] - \mathbb{E}[V]\mathbb{E}[E^2] = \mathbb{E}[V^2] - \mathbb{E}[V]^2 = \text{Var}(V)$$

Next, we determine the variance of $E^2$ by looking at the expectation of $E^4$:

$$\mathbb{E}[E^4] = \mathbb{E}[V^2Z^4] = \mathbb{E}[V^2]\mathbb{E}[Z^4] = 3\mathbb{E}[V^2]$$

Using the standard identity $\mathbb{E}[V^2] = \text{Var}(V) + \mathbb{E}[V]^2$, we expand $\text{Var}(E^2)$:

$$\text{Var}(E^2) = \mathbb{E}[E^4] - \mathbb{E}[E^2]^2 = 3(\text{Var}(V) + \mathbb{E}[V]^2) - \mathbb{E}[V]^2 = 3\text{Var}(V) + 2\mathbb{E}[V]^2$$

Substituting these values back into the Pearson correlation formula yields the desired statement:

$$\rho(V, E^2) = \frac{\text{Var}(V)}{\sqrt{\text{Var}(V)(3\text{Var}(V) + 2\mathbb{E}[V]^2)}} = \frac{1}{\sqrt{3 + 2\frac{\mathbb{E}[V]^2}{\text{Var}(V)}}} \leq \frac{1}{\sqrt{3}}.$$

□

### D.3 Lemmas

**Lemma 1.** *Let $r, r'$ be i.i.d., element-wise $\sigma_x$-subgaussian random vectors, and let $x \sim p_r \in \{1, \ldots, |\mathcal{A}|\}$ for any distribution $p_r$ parameterized by $r$. Then*

$$\mathbb{E}[|r_x - \mu_x|] \leq \sqrt{\mathbb{E}[\sigma_x^2] \cdot (2\log(2|\mathcal{A}|) + 2)} \text{ and}$$
$$\mathbb{E}[|r_x - r'_x|] \leq \zeta \cdot \sqrt{\mathbb{E}[\sigma_x^2]}.$$

*for $\zeta := 1 + \sqrt{2\log(2 \cdot |\mathcal{A}|) + 2}$.*

*Proof.* We generalize the proof given by Menet et al. [2026a] from deterministic $x = f(r)$ to distributions $x \sim p_r$. Define $Z_z = (r_z - \mu_z)/\sigma_z$. Since $r_z - \mu_z$ is $\sigma_z$-subgaussian, $Z_z$ is 1-subgaussian. Cauchy-Schwarz then gives, $\mathbb{E}[|r_x - \mu_x|] = \mathbb{E}[\sigma_x|Z_x|] \leq \sqrt{\mathbb{E}[\sigma_x^2]\mathbb{E}[Z_x^2]}$. Now, $Z_x^2 \leq \max_{1 \leq j \leq |\mathcal{A}|} Z_j^2$, so the tail integral formula for expectations with union bound gives $\mathbb{E}[Z_x^2] \leq \mathbb{E}[\max_{1 \leq j \leq |\mathcal{A}|} Z_j^2] = \int_0^\infty \mathbb{P}[\max_{1 \leq j \leq |\mathcal{A}|} Z_j^2 \geq t]dt \leq \int_0^\infty \min(1, 2|\mathcal{A}|e^{-t/2})dt = 2\log(2|\mathcal{A}|) + 2$. The first result then follows immediately. For the second result, apply the triangle inequality $\mathbb{E}[|r_x - r'_x|] \leq \mathbb{E}[|r_x - \mu_x|] + \mathbb{E}[|r'_x - \mu_x|]$, bound the first term as above, and the second by $\sqrt{\mathbb{E}[\sigma_x^2]}$, using that $r' \perp x$ and thus $\mathbb{E}[(Z'_x)^2] \leq 1$. □

**Lemma 2.** *Let $r \sim \mathcal{N}(\mu, \Sigma)$ be a multivariate Gaussian with uniformly bounded standard deviations $\sigma_1(x) \leq 1 \; \forall x$ that is consecutively evaluated at locations $(x_t)_{t=1}^T$ with observations $y_t = r(x_t) + \varepsilon_t$. The noise is i.i.d. $\varepsilon_t \sim \mathcal{N}(0, \eta_t^2)$ with $\eta_t \leq \eta \; \forall t$. Then, the maximum information gain $\gamma_T$ upper bounds the aggregated predictive variances $\sigma_t^2(x) := \text{Var}[r(x) \mid y_1, \ldots, y_{t-1}]$ at the evaluation locations, i.e.,*

$$\gamma_T := \max_{x_1, \ldots, x_T} I(y_{1:T}; r) \geq I(y_{1:T}; r) \geq \underbrace{\frac{\ln(1+\eta^{-2})}{2}}_{=:1/C_\eta} \sum_{t=1}^T \sigma_t^2(x_t).$$

*Proof.* We repeat the proof given by Menet et al. [2025], which generalizes the proof of Srinivas et al. [2010] from homoscedastic to heteroscedastic noise. First, note that the expression on the right hand side is well-defined, because $\sigma_t(x_t)$ only depends on the observation locations $x_1, \ldots, x_{t-1}$, but not on the observed value. Moreover, since Gaussian conditioning cannot increase the variance, $\sigma_t(x_t) \leq \sigma_1(x_t) \leq 1 \; \forall x$. Next, note that $y_{1:T} \mid r$ is a multivariate normal with independent components of variance $\eta_1^2, \ldots, \eta_T^2$, thus

$$I(y_{1:T}; r) = H[y_{1:T}] - H[y_{1:T}|r] = H[y_{1:T}] - \tfrac{1}{2}\sum_{t=1}^T \ln(2\pi e \eta_t^2).$$

Furthermore, one may decompose

$$H[y_{1:T}] = H[y_{1:T\text{-}1}] + H[y_T|y_{1:T\text{-}1}] = H[y_{1:T\text{-}1}] + \tfrac{1}{2}\ln(2\pi e(\eta_T^2 + \sigma_T^2(x_T))),$$

using that $y_T|y_{1:T\text{-}1}$ is Gaussian with variance $\eta_T^2 + \sigma_T^2(x_T)$. Recursively expanding then results in

$$I(y_{1:T}; r) = \tfrac{1}{2}\sum_{t=1}^{T}\ln(1 + \eta_t^{\text{-}2}\sigma_t^2(x_t)).$$

Finally, by assumption $\sigma_t(x_t) \in [0, 1]$, allowing to lower bound each summand

$$\sigma_t^2(x_t) \leq \tfrac{1}{2}\ln(1 + \eta_t^{\text{-}2}\sigma_t^2(x_t))\underbrace{\tfrac{2}{\ln(1+\eta_t^{\text{-}2})}}_{=:C_{\eta_t}}.$$

The lower bound follows from $g(\sigma_t^2(x_t)) := \ln(1+\eta_t^{\text{-}2}\sigma_t(x_t)^2) - \sigma_t^2(x_t)\ln(1+\eta_t^{\text{-}2}) \geq 0\ \forall\sigma_t(x_t) \in [0, 1]$, which in turn follows from $g(0) = 0$, $g(1) = 0$, and concavity of $g$. □

# E Broader impact, assets, and licenses

## E.1 Broader impact

This work proposes POETS, a general-purpose framework for uncertainty-aware exploration in reinforcement learning and black-box optimization. By improving sample efficiency and mitigating premature convergence, POETS may positively impact scientific discovery settings where evaluations are costly, such as protein design, therapeutic molecular design, quantum circuit synthesis, and other combinatorial optimization problems. In LLM post-training, the method can improve robustness by maintaining exploration in high-uncertainty regimes and enabling effective use of experience replay.

As a foundational algorithmic contribution, POETS does not introduce new datasets, release trained models, or directly target downstream applications. Nevertheless, like other methods that improve the efficiency of reward-driven optimization, it could indirectly accelerate the development of more capable models, amplifying both beneficial and harmful uses depending on deployment. POETS does not address reward misspecification, fairness, or alignment; these concerns remain the responsibility of system designers through careful reward construction, evaluation, and deployment safeguards.

## E.2 Assets and licenses

In this work, we utilize several open-source models, datasets, and software libraries. We acknowledge the creators of these assets and respect their licensing terms:

- **Models:** We utilize the **Qwen3-8B** and **Qwen3-Embedding-0.6B** models [Qwen Team, 2025, Zhang et al., 2025], which are released by Alibaba Cloud under the Apache 2.0 License. We also use **ProtGPT2** [Ferruz et al., 2022], built on the GPT-2 architecture [Radford et al., 2019], which is also made available under the Apache 2.0 License. Finally, we report on **GP-MoLFormer-Uniq** [Ross et al., 2025], released by IBM Research under Apache 2.0.
- **Datasets:** We evaluate our reinforcement learning approach using problems from the American Invitational Mathematics Examination (**AIME**) and LiveCodeBench [Jain et al., 2025]. The AIME problems are the copyrighted property of the Mathematical Association of America (MAA) and are used here strictly for non-commercial academic research purposes. The compiled LiveCodeBench dataset is released under the Creative Commons license (CC BY-SA 4.0), while the underlying problem text remains the property of the respective competitive programming platforms.
- **Software & Frameworks:** Our framework is implemented in **PyTorch** [Paszke et al., 2019], licensed under the Modified BSD License. We use the **Hugging Face Transformers** library [Wolf et al., 2020] and **Qiskit** [Javadi-Abhari et al., 2024], both of which are licensed under the Apache 2.0 License. For the molecular tasks, we calculate descriptors using **RDKit** [RDKit, 2026] (licensed under the BSD 3-Clause License) and query target binding oracles using the **PyTDC** library [Huang et al., 2021] (licensed under the MIT License).

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the claims about introducing the POETS framework, its theoretical cumulative regret bounds, and its empirical performance. These claims are directly supported by the proofs in Appendix D and the results in the experimental section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the limitations of POETS in the context of future work. The computational efficiency of our method is benchmarked in detail in Section 5.3.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper presents its theoretical results clearly in Section 3 and Section 4 of the main text. In Appendix D, both proof sketches and formal mathematical proofs are provided.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Appendix C entails a comprehensive "Experimental Details" section, which outlines exact hyperparameter settings, prompt templates, and evaluation environments. Additionally, the entire codebase is available in the supplementary materials for reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The entire code base, complete with experimental configuration files and environment files specifying required dependencies, was provided in the supplementary materials during the reviewing process and is now publicly available at `https://github.com/IBM/thompson-sampling-via-fine-tuning-of-llms`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix C meticulously specifies all necessary experimental settings, including the use of the AdamW optimizer, exact learning rates for shared backbones and LoRA branches, gradient clipping parameters, and prompt instructions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: To ensure statistical significance, the experiments report the mean and (its) standard error for all experimental metrics across 25 random seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper explicitly states that all experiments were conducted on a single NVIDIA H200 GPU with 141GB of memory. Moreover, Appendix C states the compute hours for each of the reported experiments as well as a total that includes preliminary experiments and hyper-parameter tuning.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research revolves around the development of a theoretical and algorithmic exploration framework for LLMs and does not contain methods, applications, or subject matter that inherently violate the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: A dedicated broader impacts section is provided in the Appendix (Section E.1).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The contribution of this work is the POETS algorithmic framework; it does not involve the release of new language models, image generators, or scraped datasets

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: All owners of code, data, and models are properly credited and the respective licenses are respected and explicitly mentioned within a dedicated Section E.2 in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: POETS is a novel framework for sample-efficient reinforcement learning. The full codebase is provided in the supplementary materials, including proper documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The experimental setup relies entirely on automated black-box optimization benchmarks (such as FAQ refinement, Protein Search, and Quantum Circuit Design, Therapeutics Data Commons) and existing mathematical or coding datasets. Therefore, the research does not involve crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Because the paper's methodology does not involve research with human subjects, IRB approvals or equivalent reviews are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the central focus of the optimization method proposed in this research. Thus, their usage is comprehensively documented, including system instructions, generation prompts, and fine-tuning details throughout the main text and Appendix C.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.