PITA: PREFERENCE-GUIDED INFERENCE-TIME ALIGNMENT FOR LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference-time alignment enables large language models (LLMs) to generate outputs aligned with end-user preferences while keeping the base model fixed. Recent methods use small guidance models to modify token generation under a KL-regularized reward objective. When supervision consists of pairwise preferences, a common approach first estimates a reward model and then learns a decoding guide from its outputs. We introduce PITA, a framework that learns the guidance model directly from preference feedback. PITA uses predicted preference probabilities to reweight the base model's next-token distribution, with iterative data collection and refinement of the guide. We connect preference probabilities to the optimal tilted output distribution and establish performance bounds under linear function approximation. Our experiments cover mathematical reasoning, graph reasoning, and alignment from noisy human preferences. On reasoning tasks, PITA attains accuracy comparable to inference-time alignment methods with access to the true reward function, demonstrating the value of learning guidance directly from preferences. Code: https://anonymous.4open.science/r/pref-sharp-D223.

1 INTRODUCTION

Large language model (LLM) alignment traditionally relies on additional training, such as fine-tuning with human feedback, to ensure that outputs conform to human preferences. Such training requires access to the base model's weights to optimize its policy. Fine-tuning LLMs is computationally expensive, and their weights may be unavailable when models are served through an API. This has motivated the study of inference-time alignment, where *the pre-trained LLM's weights remain frozen and alignment is achieved by steering its decoding process* to meet end-user needs. By intervening during generation, this approach avoids retraining the base model and enables users to adapt its outputs to their alignment criteria.

Recent inference-time alignment methods employ small guidance models or value functions to influence token generation during decoding (Mudgal et al., 2023; Han et al., 2024; Zhou et al., 2025). However, traditional reinforcement learning requires scalar reward feedback, and reward-supervised methods such as Q# (Zhou et al., 2025) rely on *access to the true task reward function* to train their guides. The guidance model scores candidate tokens according to the anticipated reward of their continuations, and these scores adjust the base LLM's next-token probabilities. Such methods optimize a KL-regularized objective, balancing reward against deviation from the reference model. The desired output distribution is an exponential tilt of the reference distribution, placing greater weight on high-reward completions. This requires next-token probabilities and control over decoding. Supported OpenAI models, for example, expose top-token log probabilities (OpenAI, 2026).

Reward-guided alignment thus relies on a signal that scores the quality of generated outputs. In many applications, however, the available feedback takes the form of *pairwise preferences*, indicating which of two responses a user prefers. One route first learns a reward model from these comparisons and then trains a decoding guide using its predicted rewards. Obtaining a reliable reward model requires labeled comparisons and an additional, computationally costly training stage. Noisy or inconsistent preferences make this estimation difficult, and errors in the fitted rewards propagate to the decoding guide. This two-stage procedure can therefore be brittle. *How do we learn inference-time guidance directly from preference feedback?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

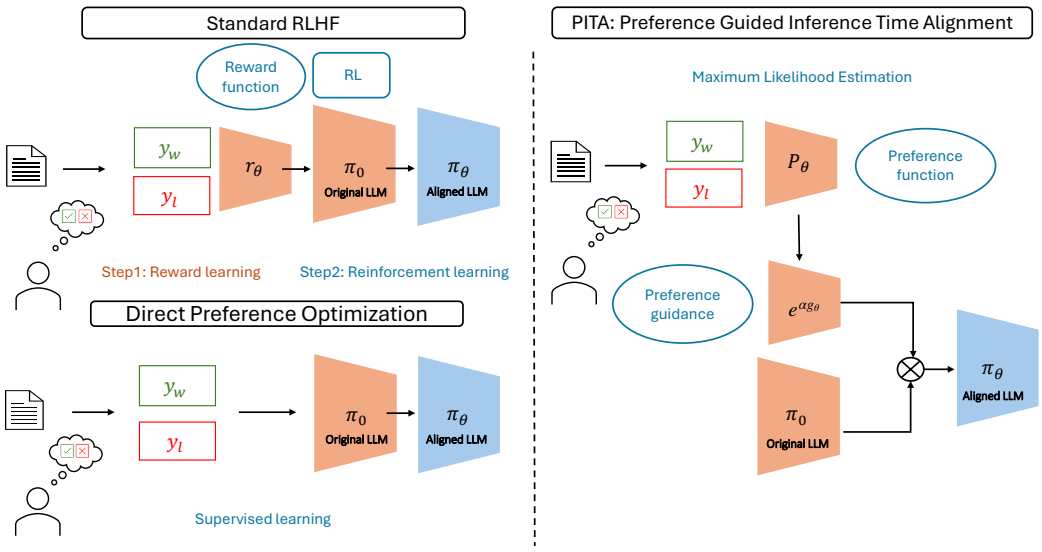

Figure 1: PITA alongside RLHF and DPO. PITA learns a preference guide and reweights the fixed reference policy at inference. The correction uses $g_\theta = \Psi(\hat{\mathcal{P}}^\theta)$ and strength α, as in Equation 13.

In this work, we introduce **PITA: Preference-Guided Inference-Time Alignment**. Its key innovation is to express the optimal decoding policy in terms of preference probabilities, thereby connecting pairwise feedback directly to token selection. PITA uses this connection to learn a small preference-based guidance model that operates alongside a frozen LLM and adjusts token probabilities during generation. Learning the guide directly from preference labels sidesteps explicit reward-model training and the propagation of errors from that intermediate fit. Figure 1 illustrates how this approach relates to other alignment methods.

To learn this guide, PITA estimates preference probabilities over the model's continuations through iterative data collection and refinement. During training, the current guide selects prefixes, and continuations sampled from the reference model receive preference labels against a greedy reference completion. The accumulated comparisons then refine the guide, allowing successive rounds to explore continuations favored by the updated policy. Once trained, the guide reweights token probabilities during inference while the base LLM remains fixed.

We provide a performance analysis using linear function approximation for the latent reward, which gives a preference model with linear log-odds and parameter θ. We bound the error in estimating θ from adaptively collected comparisons and relate this error to the performance of the exact preference-guided policy. Under bounded features and a coverage condition, we establish sublinear cumulative regret and show that the induced completion distribution approaches the optimal exponential tilt as the dataset grows. The practical algorithm uses a neural preference predictor to guide decoding.

We then evaluate PITA on mathematical reasoning, graph reasoning, and AlpacaEval with noisy human preference labels. Figure 2 illustrates the effect of preference guidance on the solution generated for a GSM8K problem. PITA achieves reasoning accuracy comparable to Q#, which has access to the true task reward function. On GSM8K, it substantially outperforms Q#-HF, which first learns a reward model from the same preference data and then uses it to tilt the reference distribution. These results demonstrate the value of learning inference-time guidance directly from preferences.

2 Related Work

Preference learning has a long history in RL and bandit literature (Akrour et al., 2012; Auer et al., 2014; Sadigh et al., 2017). Alongside the empirical success of training LLMs to generate outputs aligned with human preferences referred to as Reinforcement Learning with Human Feedback (RLHF), the theoretical aspects of preference learning have been extensively studied (Chambers et al., 2021; Shah et al., 2016; Heckel et al., 2018). Direct preference optimization (DPO) updates

Problem from GSM8K: Christina is planning a birthday party and needs .75 gift bags per invited guest, because 1/4 of attendees don't show up. She invited 16 friends. Gift bags are $2 each. How much will she spend?

Incorrect Solution from π^{ref}: Let's break this down step by step: 1. Christina invited 16 friends. 2. Since 1/4 of attendees don't show up, she'll only need to give gift bags to 3/4 of the invited guests. To find this amount, multiply 16 by 3/4: 16 × (3/4) = 12. 3. Since she needs 0.75 gift bags per invited guest, she'll need 0.75 × 12 = 9 gift bags. 4. Each gift bag costs $2, so the total cost will be 9 × $2 = $18. So, Christina will spend $\boxed{\$18}$ on gift bags.

Correct Solution from PITA: Since Christina invited 16 friends, and 1/4 of attendees don't show up, the number of friends who will actually attend the party is 16 - (1/4)*16 = 16 - 4 = 12. Since Christina needs 0.75 gift bags per invited guest, and she invited 16 friends, she needs 0.75*16 = 12 gift bags. The cost of each gift bag is $2, so the total cost of the gift bags is 12*2 = $24. Final Answer: The final answer is $\boxed{24}$. I hope it is correct.

Figure 2: Comparison of Incorrect Solution from the Reference model and the Correct Solution from PITA to a math reasoning problem.

the language model directly from pairwise preferences through a classification objective (Rafailov et al., 2023). Several works characterize the alignment problem as a general preference optimization objective, which is the focus of our work. For instance, Azar et al. (2024) define a general RLHF objective, with RLHF and DPO as specific examples. Contrastive Preference Learning (CPL) (Hejna et al., 2023) models human preferences based on regret rather than reward, enabling scalable, off-policy learning without requiring reward functions or reinforcement learning. Additionally, Ye et al. (2024) propose a general RLHF framework without assuming a reward function or Bradley-Terry preferences, using a reverse-KL minimax game between LLMs, and introduce sample-efficient algorithms for both offline and online learning with empirical validation. An extended version of the related work is provided in Appendix A.

3 PRELIMINARIES

We begin by introducing the finite-horizon Markov Decision Process (MDP) framework that formalizes the generation process of language modeling.

A finite-horizon MDP is formulated as $\mathcal{M} = (\mathcal{P}, \mathcal{S}, \mathcal{A}, T, \rho)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $T \in \mathbb{N}$ is the maximal length of the episode, and ρ denotes the starting state distribution. A policy in MDPs, denoted by $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, maps each state to a distribution over actions. The initial state is sampled from ρ, and the policy generates a trajectory of at most T steps through this transition kernel.

LLMs as Deterministic MDPs: In text generation applications using an LLM, π is the LLM's policy to choose the next token (action) given the current sequence of tokens (state). The transition kernel is deterministic. Each new state is obtained by concatenating the previous tokens with the current token chosen by the policy. Each sentence generated by the policy is termed a trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$, where $s_0 := x$ is the initial instruction sampled from the starting state distribution ρ, and $a_t \sim \pi(\cdot|s_t)$. Therefore, at each step t, the state $s_t = (x|a_{0:t-1})$ consists of the instruction x and tokens generated up to $t-1$. The generation process ends when the LLM policy samples a special end-of-sequence token or the trajectory reaches a predefined length limit, characterized by T. We index actions by $t = 0, \ldots, T-1$ and pad shorter generations with an absorbing terminal state. The score of a trajectory generated by the LLM is defined using a reward function $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The reward function is typically sparse with $r(s_t, a_t) = 0$ for $0 \leq t < T-1$, quantifying the desirability of the final output of the LLM for the given instruction x. An LLM policy can be learned using T-horizon reinforcement learning by maximizing the expected reward for a given policy defined as $J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)] = \mathbb{E}_{x\sim\rho}[V^\pi(x)]$, where $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} r(s_t, a_t)|s_0 = s\right]$ is the value of a state s under policy π. Similarly, the

state-action value function and the advantage function for a state-action pair (s, a) are given by $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} r(s_t, a_t) | s_0 = s, a_0 = a\right]$, and $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ respectively.

RL Fine-Tuning: Reinforcement learning uses a reward function to score responses and optimizes a policy under a KL penalty relative to a fixed reference model π_{ref}. For a completion y of prompt x, let $r(x, y)$ denote its reward. The fine-tuning objective is

$$\max_\pi \mathbb{E}_{x\sim\rho}\left[\mathbb{E}_{y\sim\pi(\cdot|x)} r(x,y) - \eta \mathrm{KL}\left(\pi(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\right)\right], \tag{1}$$

where $\eta > 0$ balances higher reward against deviation from the reference distribution and $\mathrm{KL}(p \,\|\, q) = \mathbb{E}_{z\sim p}[\log(p(z)/q(z))]$.

Desired Tilted Distribution: Solving Equation 1 gives the exponential tilt (Rafailov et al., 2023)

$$\pi_r^*(y \mid x) = \frac{1}{Z_r(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{r(x,y)}{\eta}\right), \tag{2}$$

where $Z_r(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp(r(x,y)/\eta)$ normalizes the distribution. This product form retains the reference model's distribution and reweights completions according to reward. Reward feedback supplies the signal used by RL fine-tuning to move the learned policy toward this desired tilt.

Preference Feedback and Reward Modeling: In the preference regime, the observed feedback consists of pairwise comparisons, while the underlying rewards are latent. Given prompt x, annotators compare responses y_w and y_l and indicate $y_w \succ y_l \mid x$. The Bradley–Terry model (Bradley & Terry, 1952) relates such comparisons to a latent reward r^* through

$$\mathcal{P}^*(y_w \succ y_l \mid x) = \sigma(r^*(x, y_w) - r^*(x, y_l)), \tag{3}$$

where σ is the sigmoid function. One route estimates a reward model r_θ by maximum likelihood on preference pairs $\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^n$. The training objective is

$$\mathcal{L}(r_\theta, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\theta(x, y_w) - r_\theta(x, y_l))\right]. \tag{4}$$

The fitted reward then supplies the RL fine-tuning objective above.

DPO: Direct preference optimization (Rafailov et al., 2023) seeks the same tilted distribution through supervised learning on preference pairs. Substituting the log-policy ratios from Equation 2 into the Bradley–Terry model cancels the partition function and yields a loss for training the policy directly. However, this approach trains the base model itself and requires access to its full set of weights.

The product form in Equation 2 also suggests applying the tilt directly during generation. *Can we obtain an inference-time alignment method that explicitly tilts the frozen reference model's distribution using only preference feedback?* This is the question we address next.

4 PITA: Preference-Guided Inference-Time Alignment

We answer this question by expressing the optimal decoding policy through preferences over completions, using the structure of the soft state-action value function in deterministic MDPs (Zhou et al., 2025). Here, r denotes the latent reward r^* underlying the Bradley–Terry model.

For a given parameter $\eta > 0$, the soft-value function $V^{\pi,\eta}$ of a policy π is defined as:

$$V^{\pi,\eta}(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \left[r(s_t, a_t) - \eta \,\mathrm{KL}(\pi(\cdot \mid s_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t))\right] \middle| s_0 = s\right] \tag{5}$$

It can be shown that the KL-regularized RL objective is solvable using the soft Bellman equations. In particular, the optimal policy is given by Rafailov et al. (2024):

$$\begin{aligned} V_T^{\star,\eta}(s) &= 0, \\ Q_t^{\star,\eta}(s,a) &= r(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V_{t+1}^{\star,\eta}(s')\right], \\ \pi^{\star,\eta}(a_t \mid s_t) &\propto \pi_{\text{ref}}(a_t \mid s_t) \exp\left(\eta^{-1} Q_t^{\star,\eta}(s_t, a_t)\right), \\ V_t^{\star,\eta}(s) &= \eta \ln \mathbb{E}_{a\sim\pi_{\text{ref}}(\cdot|s)}\left[\exp\left(\eta^{-1} Q_t^{\star,\eta}(s,a)\right)\right]. \end{aligned} \tag{6}$$

Here, $Q^{\star,\eta}$ represents the soft-state-action value function under the optimal policy π^*. We specialize our discussion to the deterministic transition structure of the LLM generative process. In the sparse reward setting, the optimal values $V^{\star,\eta}, Q^{\star,\eta}$ are functions of the conditional reward distributions of the generations under π_{ref} (Zhou et al., 2025). Specifically:

$$V_t^{\star,\eta}(s) = \eta \ln \mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} r(s_{T-1}, a_{T-1})\right) \middle| s_t = s\right], \tag{7}$$

$$Q_t^{\star,\eta}(s,a) = \eta \ln \mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} r(s_{T-1}, a_{T-1})\right) \middle| s_t = s, a_t = a\right]. \tag{8}$$

For the sake of completeness, we present the proof of the above results in Appendix B. The Q# approach utilizes a policy of the form given in (6) with $Q_t^{\star,\eta}(s,a)$ as given in (8).

Preference-based Q Value function: PITA uses preference data to learn a decoding guide directly.

We first derive the soft-state-action value function and the value function in terms of the preference distribution. Fix the reference policy π_{ref}. For a given state s, let y_s^{ref} be its greedy completion, with ties resolved by a fixed ordering of tokens. We use this same parent-state completion for every candidate action at s. The win probability of a completion y_s against this anchor is $\mathcal{P}^*(y_s \succ y_s^{\text{ref}})$. We write $r(y)$ for its latent terminal reward.

Theorem 1. *Let $\Psi(x) \triangleq \log \frac{x}{(1-x)}$. Then, assuming the BT preference model, we have*

$$\begin{aligned} V_t^{\star,\eta}(s) &= r(y_s^{\text{ref}}) + \eta \ln \mathbb{E}_{y_s \sim \pi_{ref}}\left[\exp\left(\eta^{-1}\Psi(\mathcal{P}^*(y_s \succ y_s^{\text{ref}}))\right) \middle| s_t = s\right], \\ Q_t^{\star,\eta}(s,a) &= r(y_s^{\text{ref}}) + \eta \ln \mathbb{E}_{y_{s'} \sim \pi_{ref}}\left[\exp\left(\eta^{-1}\Psi(\mathcal{P}^*(y_{s'} \succ y_s^{\text{ref}}))\right) \middle| s_t = s, a_t = a\right], \end{aligned} \tag{9}$$

where $s' = (s|a)$ is the state obtained by concatenating a to state s.

The proof of Theorem 1 is provided in Appendix B. The theorem expresses $Q_t^{\star,\eta}$ and $V_t^{\star,\eta}$ through reference-rollout preferences and a shared reward offset. This offset cancels when forming the optimal policy, so the preference distribution determines the decoding correction.

Preference distribution over greedy decoding: The preference probability $\mathcal{P}^*(y_x \succ y_x^{\text{ref}})$ is the probability that a completion wins against greedy decoding for a given context. Since the reference greedy completion y_x^{ref} is unique, using the reward-equivalence property of the Bradley-Terry model, we derive the following lemma.

Lemma 1. *Let r^* be the implicit reward function. For each context x, define $\tilde{r}_x(y) = r^*(y) - r^*(y_x^{\text{ref}})$. Then $\mathcal{P}^*(y \succ y_x^{\text{ref}}) = \sigma(\tilde{r}_x(y))$ for every completion y of x.*

The proof of Lemma 1 is given in Appendix B. The above reward equivalence is crucial for developing our algorithm. It states that the preference probability is solely a function of the completion and context. We can parameterize $\mathcal{P}^\theta(y_x)$ using a neural network, which estimates the win probability $\mathcal{P}^*(y_x \succ y_x^{\text{ref}})$ over greedy decoding. The soft-Bellman equations give:

$$\pi^{\star,\eta}(a \mid s) \propto \pi_{\text{ref}}(a \mid s) \mathbb{E}_{y_{s'} \sim \pi_{\text{ref}}}\left[\exp\left(\eta^{-1}\Psi(\mathcal{P}^*(y_{s'} \succ y_s^{\text{ref}}))\right) \middle| s_t = s, a_t = a\right] \tag{10}$$

This naturally motivates us to consider parametrized policies of the form π^θ, such as:

$$\pi^{\theta,\eta}(a \mid s) \propto \pi_{\text{ref}}(a \mid s) \mathbb{E}_{y_{s'} \sim \pi_{\text{ref}}}\left[\exp\left(\eta^{-1}\Psi(\mathcal{P}^\theta(y_{s'} \succ y_s^{\text{ref}}))\right) \middle| s_t = s, a_t = a\right] \tag{11}$$

where $s' = (s|a)$ is the state obtained by concatenating s with action a.

4.1 Algorithm

In this section, we present PITA, a preference-guided inference-time alignment algorithm for the KL-regularized RLHF objective. PITA iteratively learns a parametrized preference guide while keeping the reference model fixed. We describe the neural guide below, and Section 5 analyzes the exact policy in Equation 11 under linear reward approximation.

The exact policy in Equation 11 depends on a nonlinear moment of the preference probabilities under reference rollouts. For the neural guide, we use a scalar mean-preference predictor and a certainty-equivalent decoding rule. Let $\hat{\mathcal{P}}^\theta : \mathcal{S} \times \mathcal{A} \to (0,1)$ estimate

$$\hat{\mathcal{P}}^\theta(s,a) \approx \mathbb{E}_{y_{s'} \sim \pi_{\text{ref}}}\left[\mathcal{P}^*(y_{s'} \succ y_s^{\text{ref}}) \mid s_t = s, a_t = a\right].$$

The predictor is trained using binary cross-entropy (BCE):

$$\mathcal{L}(o, p) = -o \log p - (1 - o) \log(1 - p), \qquad p = \hat{\mathcal{P}}^{\theta}(s, a), \tag{12}$$

where $o = \mathbf{1}_{y \succ y_s^{\mathrm{ref}}}$ is the observed preference label. Applying the log-odds transform to this mean gives the certainty-equivalent rule

$$\widetilde{\pi}^{\theta,\alpha}(a \mid s) \propto \pi_{\mathrm{ref}}(a \mid s) \exp\left(\alpha \Psi\left(\hat{\mathcal{P}}^{\theta}(s, a)\right)\right), \qquad \alpha = \eta^{-1}. \tag{13}$$

Here α is the decoding guidance strength. This approximation replaces the nonlinear preference moment by the transformed scalar mean, and a trained guide can be reused at several values of α.

During guide training, the current policy rolls in to a state s_t, and π_{ref} supplies sampled continuations and a greedy anchor. Each sampled continuation supplies its first action and a preference label against that anchor. We aggregate these examples and update the guide using BCE, as detailed in Algorithm 1. At inference, decoding uses the trained guide.

Algorithm 1 Preference-Guided Inference-Time Alignment (PITA)

Input: Reference policy π_{ref}, guidance strength α. Initialize θ_0 and dataset $\mathcal{D} = \emptyset$.
for $k = 1, 2, \ldots$ until convergence **do**
  Let $\pi^{k-1} \leftarrow \widetilde{\pi}^{\theta_{k-1},\alpha}$ be the policy induced by $\hat{\mathcal{P}}^{\theta_{k-1}}$.
  for $i = 1, 2, \ldots, N$ **do**
    Sample a rollout step uniformly: $t \sim \mathrm{Unif}(\{0, \ldots, T-1\})$.
    Sample $s_0 \sim \rho$ and roll in with π^{k-1} for t steps to obtain s_t.
    Obtain the anchor $y_{s_t}^{\mathrm{ref}}$ by greedy decoding with π_{ref}.
    Sample M responses $y_{s_t}^{j}$ from π_{ref}, with first actions a_t^j, $j \in [M]$.
    Observe $o_j = \mathbf{1}_{y_{s_t}^{j} \succ y_{s_t}^{\mathrm{ref}}}$ and add $(s_t, a_t^j, y_{s_t}^{j}, y_{s_t}^{\mathrm{ref}}, o_j)$ to $\mathcal{D}$ for $j \in [M]$.
  end for
  Update θ_k using maximum likelihood estimation on the aggregated data:

$$\theta_k \leftarrow \arg\min_{\theta} \mathbb{E}_{(s,a,y,y^{\mathrm{ref}},o)\sim\mathcal{D}} \left[\mathcal{L}(o, \hat{\mathcal{P}}^{\theta}(s, a))\right]$$

end for
Output: Final θ_k.

We note that our algorithm belongs to the class of value-based algorithms for post-training LLMs (Mudgal et al., 2023; Han et al., 2024; Zhou et al., 2025). CD (Mudgal et al., 2023) and VAS (Han et al., 2024) learn guidance from the expected reward-to-go under reference-policy continuations. In contrast, Q# (Zhou et al., 2025) aims to learn the optimal state-action value function using distributional reinforcement learning (Bellemare et al., 2023). PITA shares the iterative guide-learning structure of Q# and trains its guide directly from preference labels, avoiding a separate terminal reward-model estimation stage.

5 Theoretical Results on PITA Performance

To understand how preference learning improves alignment, we analyze the exact preference-guided policy (11) under linear reward approximation. We first bound the error in estimating the preference parameter, then use this bound to show that the induced completion distribution approaches the optimal exponential tilt of the reference model. Proofs appear in Appendix C.

Assumption 1. *The reward has the form $r_\theta(x, y) = \theta^\top \Phi(x, y)$ for a known feature map with $\sup_{x,y} \|\Phi(x, y)\|_2 \le L$. The true parameter $\theta^\star$ belongs to the bounded class*

$$\Theta_B = \left\{\theta \in \mathbb{R}^d \mid \langle \mathbf{1}, \theta \rangle = 0,\ \|\theta\|_2 \le B\right\}.$$

5.1 Preference Modeling in PITA

For each context x, let y_x^{ref} be the greedy completion under the fixed reference policy, with deterministic tie breaking. Define the centered feature $\psi_x(y) = \Phi(x, y) - \Phi(x, y_x^{\mathrm{ref}})$, so that $\|\psi_x(y)\|_2 \le 2L$.

The Bradley–Terry preference model becomes

$$\mathcal{P}^{\theta}(y \succ y_x^{\mathrm{ref}} \mid x) = \sigma(\langle \theta, \psi_x(y) \rangle). \tag{14}$$

Centering subtracts a context-dependent reward offset and preserves the optimal policy.

Maximum Likelihood Estimation (MLE). Given $\mathcal{D} = \{(x_i, y_i, y_{x_i}^{\mathrm{ref}}, o_i)\}_{i=1}^n$, let $z_i = \psi_{x_i}(y_i)$ and $\hat{\theta} \in \arg\min_{\theta \in \Theta_B} \mathcal{L}_{\mathcal{D}}(\theta)$, where

$$\mathcal{L}_{\mathcal{D}}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \left[o_i \log \sigma(\langle \theta, z_i \rangle) + (1 - o_i) \log\big(1 - \sigma(\langle \theta, z_i \rangle)\big) \right]. \tag{15}$$

Each comparison is selected before observing its label, which is conditionally Bernoulli with mean $\sigma(\langle \theta^\star, z_i \rangle)$. This permits adaptive data collection. Following the concentration approach used for preference models (Faury et al., 2020; Shah et al., 2016; Pacchiano et al., 2021), we obtain:

Lemma 2. *For fixed $n \geq 1$ and $\lambda > 0$, Assumption 1 and the sampling model give, with probability $\geq 1 - \delta$,*

$$\|\hat{\theta} - \theta^\star\|_{\Sigma_{\mathcal{D}} + \lambda I} \leq C \sqrt{\frac{d \log(1 + 4L^2/(d\lambda)) + \log(1/\delta)}{\gamma^2 n} + \lambda B^2}, \tag{16}$$

where C is a universal constant, $\gamma = (2 + e^{-2LB} + e^{2LB})^{-1}$, and $\Sigma_{\mathcal{D}} = n^{-1} \sum_{i=1}^n z_i z_i^\top$.

5.2 Regret Bound

The exact policy induces the completion distribution

$$\pi^{\theta,\eta}(y \mid x) = \frac{\pi_{\mathrm{ref}}(y \mid x) \exp(\langle \theta, \psi_x(y) \rangle / \eta)}{\mathbb{E}_{y' \sim \pi_{\mathrm{ref}}(\cdot|x)}[\exp(\langle \theta, \psi_x(y') \rangle / \eta)]}. \tag{17}$$

Write $V^{\theta,\eta} = \mathbb{E}_{x \sim \rho}[V^{\pi^{\theta,\eta},\eta}(x)]$ for its value under the true reward, and $V^{\star,\eta} = V^{\theta^\star,\eta}$. Let $\mathcal{D}_k$ contain the first n_k comparisons, θ_k be its MLE, and $\Sigma_k = \Sigma_{\mathcal{D}_k}$.

Assumption 2. *For the chosen deterministic schedules n_k and $\lambda_k > 0$, the data provide uniform coverage of the policy feature means:*

$$G_k \triangleq \sup_{x \in \mathcal{X},\, \theta \in \Theta_B} \left\| (\Sigma_k + \lambda_k I)^{-1/2} \mathbb{E}_{y \sim \pi^{\theta,\eta}(\cdot|x)}[\psi_x(y)] \right\|_2 \leq G.$$

Theorem 2. *Under Assumptions 1–2 and the stated sampling model, with probability $\geq 1 - \delta$,*

$$\sum_{k=1}^{K} (V^{\star,\eta} - V^{\theta_k,\eta}) \leq C \sum_{k=1}^{K} G_k \sqrt{\frac{d \log(1 + 4L^2/(d\lambda_k)) + \log(K/\delta)}{\gamma^2 n_k} + \lambda_k B^2}. \tag{18}$$

With $n_k = mk$ comparisons for a fixed $m \geq 1$ and $\lambda_k = 1/n_k$, uniform coverage gives cumulative regret of order $\sqrt{K \log(K/\delta)}$, with other problem parameters fixed. The proof uses the closed-form preference representation to relate estimation error to value suboptimality through the expected feature under the induced policy.

The value gap also measures distance to the optimal tilted distribution:

$$V^{\star,\eta} - V^{\theta_k,\eta} = \eta\, \mathbb{E}_{x \sim \rho} \Big[\mathrm{KL}\Big(\pi^{\theta_k,\eta}(\cdot \mid x) \,\|\, \pi^{\theta^\star,\eta}(\cdot \mid x) \Big) \Big]. \tag{19}$$

For fixed $\eta > 0$, the proof's per-round bound therefore gives context-averaged KL divergence $O(\sqrt{\log(K/\delta)/K})$ at round K under these schedules.

6 Experimental Evaluation

6.1 Experimental Setup

Models and Datasets. We compare direct preference guidance with reward-based guidance on GSM8K (Cobbe et al., 2021), star-graph reasoning (Bachmann & Nagarajan, 2024), and AlpacaEval (Dubois et al., 2023). For mathematical reasoning, we use Llama-3-8B-Instruct as the reference

Table 1: Accuracy (%) and measured forward KL on GSM8K and MATH-500. Q#-HF fits a separate reward model from preferences, while PITA trains its guide directly.

Dataset	**GSM8K (ID)**			**MATH-500 (OOD)**		
Algorithm	**pass@1** ↑	**maj@8** ↑	**KL**	**pass@1** ↑	**maj@8** ↑	**KL**
π_{ref}	69.1	85.8	–	25.4	35.3	–
Alignment from exact rewards						
Q#	78.4	88.1	2.65	26.8	35.4	6.9
Alignment from preference pairs						
Q#-HF	67.23	78.09	2.48	25.43	35.2	5.39
PITA (ours)	77.11	86.20	6.15	26.4	34.8	4.68

model and Llama-3.2-1B-Instruct as the guide, following the setup of Zhou et al. (2025). We use GPT-2 for star graphs, and Llama and Qwen models for AlpacaEval. We use AlpacaFarm's noisy human preferences and construct preference pairs for reasoning. Math preferences favor correct answers, with an OpenMath judge breaking correctness ties (Toshniwal et al., 2024).

Metrics and Algorithms. We report *pass@1*, the mean correctness of a sampled response, *maj@8*, the majority-vote accuracy over eight responses, and forward KL $\mathrm{KL}(\pi \,\|\, \pi_{\text{ref}})$, summed over generated tokens and averaged over responses. For mathematical reasoning, **Q#** Zhou et al. (2025) learns its guide from binary correctness rewards. **Q#-HF** and **PITA** use the same pairwise preferences: Q#-HF trains its guide using a fitted terminal reward model, while PITA trains directly on the comparisons.

6.2 Performance on Arithmetic Reasoning

GSM8K tests in-distribution reasoning, while MATH-500 evaluates transfer to harder mathematical problems (Lightman et al., 2023).

Main Results: On GSM8K (Table 1), PITA improves pass@1 from 69.10% to 77.11% and maj@8 from 85.80% to 86.20%. Among the preference-supervised methods, PITA exceeds Q#-HF by 9.88 and 8.11 percentage points on these metrics at the reported operating points. On MATH-500, PITA obtains 26.4% pass@1 and 34.8% maj@8, versus the reference model's 25.4% and 35.3%.

Preference Guidance and Reward Estimation: The single-round ablation compares the two preference-supervised methods at similar KL values. PITA achieves 74.8% pass@1 and 86.1% maj@8 at KL 18.4, while Q#-HF achieves 49.9% and 72.0% at KL 18.85 (Appendix E). The reward-score histograms show weaker separation between correct and incorrect responses when the intermediate reward model is trained with less data. Together, these results support learning the guide directly from preferences. Across eight evaluation seeds, PITA obtains mean GSM8K pass@1 of 76.62% and maj@8 of 85.72%. Mean pass@1 exceeds the reference by 7.52 percentage points.

Guidance Strength and Policy Deviation: Greater deviation from the reference model is useful when it improves task accuracy. We therefore vary the decoding strength $\alpha = 1/\eta$ for fixed trained guides and report accuracy alongside measured KL in Table 2. For PITA, increasing α from 0.5 to 6 raises KL from 2.37 to 6.15 and pass@1 from 66.04% to 77.11%. These accuracy gains are measured against benchmark answers, providing evidence of useful alignment rather than reward hacking. At nearby KL values of 2.37 and 2.31, PITA and Q#-HF attain 66.04% and 62.24% pass@1, respectively. Around KL 2.6, both attain approximately 69%. Stronger guidance then brings PITA to 77.11%, while Q#-HF reaches 70.57% as its KL saturates near 2.68. Increasing α from 6 to 10 raises maj@8 from 86.20% to 88.80%, while pass@1 changes from 77.11% to 75.91%.

6.3 Additional Tasks and Computational Cost

Star-Graph Reasoning: We next examine whether preference guidance can correct a reasoning shortcut learned by the reference model. A star graph $\mathcal{G}(d, l)$ has d paths of length l from a common root, and the task is to recover the correct root-to-leaf path. The reference GPT-2 model selects the

Table 2: GSM8K guidance sweep: measured KL and accuracy (%) for fixed trained guides. Rows vary decoding strength α, with each method's resulting KL shown explicitly.

	Q#-HF			PITA (ours)		
α	KL	pass@1	maj@8	KL	pass@1	maj@8
0.5	2.26	60.65	75.89	2.37	66.04	78.92
1.0	2.31	62.24	76.95	2.58	68.90	79.98
2.0	2.48	67.23	78.09	3.22	76.64	85.51
4.0	2.61	69.37	81.96	4.44	76.70	86.12
6.0	2.67	70.14	81.65	6.15	77.11	86.20
8.0	2.68	70.51	82.87	8.59	76.46	87.26
10.0	2.68	70.57	82.71	21.67	75.91	88.80

Table 3: Star-graph accuracy (%). Q# trains its guide with task rewards, while PITA trains directly from preference pairs.

Algorithm	$\mathcal{G}(2,5)$	$\mathcal{G}(5,5)$	$\mathcal{G}(3,8)$
π_{ref}	49.94	19.80	33.24
Q#	99.90	97.00	98.62
PITA (ours)	99.97	99.70	99.93

Table 4: AlpacaEval length-controlled win rate (%). Q# converts the shared preference labels into binary rewards.

Algorithm	Llama	Qwen
π_{ref}	46.91	37.28
Q#	47.15	36.86
PITA (ours)	47.22	38.86

correct path with probability close to $1/d$. PITA guides this frozen model to at least 99.7% accuracy across the three configurations in Table 3.

Noisy Human Preferences: To examine alignment for open-ended responses, we train on noisy human preferences from AlpacaFarm and evaluate using AlpacaEval. Both PITA and Q# use the same preference data, with Q# converting the labels into binary rewards. Table 4 shows that PITA increases the length-controlled win rate over the reference by 0.31 percentage points for Llama and 1.58 points for Qwen, attaining the highest win rate for both models.

Preference Data and Guide Capacity: With 4, 8, and 16 sampled completions per training problem, PITA achieves 42.10%, 68.69%, and 77.11% pass@1, respectively. The corresponding maj@8 values are 75.05%, 82.03%, and 86.20%, showing the benefit of additional preference comparisons. The three- and five-head guides reach peak pass@1 of 74.76% and 76.01% (Table 10 in the appendix).

Computational Cost: To assess the accuracy gained from additional compute, we compare PITA with reward-supervised Q# and reference sampling in the fixed-step H100 study (Table 13 in Appendix E.6). PITA and Q# require 0.472 and 0.505 GPU-hours for guide fitting. At these checkpoints, PITA attains 69.29% GSM8K pass@1 versus 62.63% for Q#, with normalized evaluation times of 1,286 and 1,247 seconds. Thus, PITA gains 6.66 percentage points in accuracy for 3.1% more evaluation time. Reference decoding obtains 58.45% with one sample in 726 seconds. Sampling two reference responses yields 74.75% oracle pass@2 in 1,420 seconds, representing best-of-two selection with perfect correctness feedback. PITA uses less time than this two-sample budget. Guided times are normalized by dividing eight-completion totals by eight, while reference times are measured directly.

7 Conclusion

We presented **PITA**, which learns inference-time guidance directly from preferences while keeping the base LLM fixed. Our analysis connects preference estimation to the optimal tilted distribution under linear function approximation. Experiments on reasoning and noisy human preferences demonstrate the value of this approach, with PITA achieving accuracy comparable to reward-supervised guidance on reasoning tasks and higher attainable GSM8K accuracy than the two-stage Q#-HF pipeline. The same trained guide can be reused at different decoding strengths, giving users control over how strongly preferences steer the model's generation.

AI USE STATEMENT

OpenAI Codex assisted with literature organization, results formatting, and LaTeX layout. AI-assisted development tools, including Cursor, were also used for code completion, debugging, refactoring, and related software development tasks. The core algorithmic implementations were developed by the authors, and all AI assisted code was reviewed and validated by the authors. The authors take responsibility for the final manuscript and accompanying artifacts.

REPRODUCIBILITY STATEMENT

The preference-learning objective and decoding rule are specified in Section 4 and Algorithm 1. The assumptions and performance analysis appear in Section 5, with proofs in Appendices B and C. Model configurations, preference-data construction, evaluation protocols, ablations, and results across random seeds are described in Appendices D and E.

Project address: https://anonymous.4open.science/r/pref-sharp-D223.

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. *arXiv preprint arXiv:2402.14740*, 2024.

Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*, pp. 116–131. Springer, 2012.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50:5–43, 2003.

Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles. *Algorithmic Learning Theory: 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014, Proceedings*, volume 8776. Springer, 2014.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. MIT Press, 2023.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Christopher P Chambers, Federico Echenique, and Nicolas S Lambert. Recovering preferences from finite data. *Econometrica*, 89(4):1633–1664, 2021.

Sapana Chaudhary, Ujwal Dinesha, Dileep Kalathil, and Srinivas Shakkottai. Risk-averse finetuning of large language models, 2025. URL https://arxiv.org/abs/2501.06911.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. Kcts: knowledge-constrained tree search decoding with token-level hallucination detection. *arXiv preprint arXiv:2310.09044*, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Louis Faury, Marc Abeille, Clément Calauzènes, and Olivier Fercoq. Improved optimistic algorithms for logistic bandits. In *International Conference on Machine Learning*, pp. 3052–3060. PMLR, 2020.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

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

Seungwook Han, Idan Shenfeld, Akash Srivastava, Yoon Kim, and Pulkit Agrawal. Value augmented sampling for language model alignment and personalization. *arXiv preprint arXiv:2405.06639*, 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Reinhard Heckel, Max Simchowitz, Kannan Ramchandran, and Martin Wainwright. Approximate ranking from pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*, pp. 1057–1066. PMLR, 2018.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W Bradley Knox, and Dorsa Sadigh. Contrastive preference learning: learning from human feedback without rl. *arXiv preprint arXiv:2310.13639*, 2023.

Edward S Hu, Kwangjun Ahn, Qinghua Liu, Haoran Xu, Manan Tomar, Ada Langford, Dinesh Jayaraman, Alex Lamb, and John Langford. Learning to achieve goals with belief state transformers. *arXiv preprint arXiv:2410.23506*, 2024.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

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 *The Twelfth International Conference on Learning Representations*, 2023.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*, 2023.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Create chat completion: OpenAI API reference, 2026. URL https://developers.openai.com/api/reference/resources/chat/subresources/completions/methods/create. Accessed September 22, 2026.

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

Aldo Pacchiano, Aadirupa Saha, and Jonathan Lee. Dueling rl: reinforcement learning with trajectory preferences. *arXiv preprint arXiv:2111.04850*, 2021.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. *arXiv preprint arXiv:2410.16033*, 2024.

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. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a Q-function. *arXiv preprint arXiv:2404.12358*, 2024.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization, 2023. URL https://arxiv.org/abs/2210.01241.

Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. *Active preference-based learning of reward functions*. CoRR, 2017.

Nihar B Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramch, Martin J Wainwright, et al. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *Journal of Machine Learning Research*, 17(58):1–47, 2016.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Chenlu Ye, Wei Xiong, Yuheng Zhang, Hanze Dong, Nan Jiang, and Tong Zhang. Online iterative reinforcement learning from human feedback with general preference model. *Advances in Neural Information Processing Systems*, 37:81773–81807, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488, 2022.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Jin Peng Zhou, Kaiwen Wang, Jonathan Chang, Zhaolin Gao, Nathan Kallus, Kilian Q Weinberger, Kianté Brantley, and Wen Sun. Q#: Provably optimal distributional RL for LLM post-training. *Advances in Neural Information Processing Systems*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Additional Related Work

We provide an expanded discussion of related work omitted from the main paper.

Alignment: Large language models (LLMs) have demonstrated emerging capabilities in advanced natural language tasks through the use of **attention mechanisms** and the **transformer architecture** (Vaswani et al., 2017; Radford et al., 2019; Brown et al., 2020; Devlin et al., 2018; Bubeck et al., 2023). In the context of LLMs, the goal is to fine-tune the responses generated for downstream tasks. **Reinforcement Learning from Human Feedback (RLHF)** (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) has become a significant technique in aligning LLMs with human values and preferences. A key aspect of the RLHF approach is the use of a **reward model** trained from preference feedback or task reward labels.

An alternative approach for utilizing preference information is **Direct Preference Optimization (DPO)** (Rafailov et al., 2023), which updates the model's policy directly using preference data. Unlike RLHF, DPO avoids reliance on a separate reward model by learning from relative preferences between pairs of outputs. The approach has been extended to relate DPO to learning an optimal soft Q-function (Rafailov et al., 2024), as well as more general preference functions (Azar et al., 2024).

LLM Reasoning: Recent methods further extend the capabilities of LLMs to generate intermediate reasoning steps. The standard approach in LLM-based reasoning begins with an initial policy π instantiated by a reference LLM. The output is generated as a sequence of steps by auto-regressively predicting the next step using prompts. This idea is widely known through the **Chain-of-Thought (CoT)** (Wei et al., 2022) approach to reasoning tasks. **Self-Consistency** (Wang et al., 2022) selects the most frequent answer from multiple generations. **Tree-of-Thought (ToT)** (Yao et al., 2023) extends the CoT method by generating tree-based reasoning traces.

Another approach uses a **value function** to guide the selection of reasoning traces from a combinatorially large set. Two common approaches include **Outcome Reward Models (ORMs)** (Cobbe et al., 2021) and **Process Reward Models (PRMs)** (Li & Li, 2024). While ORMs are trained only on the accuracy of the final answer, PRMs are trained on the accuracy of intermediate reasoning steps. A popular approach, **Best-of-N** (Lightman et al., 2023), combines the utility of a value function (either PRM or ORM) by selecting the reasoning trace with the highest value.

Inference-Time Scaling: Inference-time compute methods aim to enhance the reasoning capabilities of LLMs by allowing the models to 'think' longer or perform additional computation during inference. (Snell et al., 2024) provides a unified perspective on test-time computation as modifying an LLM's distribution. Related approaches modify outputs using **post-hoc scorers**, typically employed in **Markov Chain Monte Carlo (MCMC) algorithms** (Andrieu et al., 2003).

ReST^{EM} (Singh et al., 2023) is a self-training method where the model generates samples filtered by binary feedback, fine-tunes on these filtered samples, and repeats the process. It outperforms human-only fine-tuning and reduces reliance on human data. **STaR (Self-Taught Reasoner)** (Zelikman et al., 2022) generates rationales using a few examples, retries with the correct answer if the output is wrong, fine-tunes on successful rationales, and repeats the process. This enhances performance on reasoning tasks without large rationale datasets, rivaling much larger models.

MCTS-based approaches and similar planning strategies (Liu et al., 2023; Zhang et al., 2023; Zhou et al., 2023; Choi et al., 2023) that leverage a learned value network have demonstrated powerful test-time scaling by guiding inference with value estimates, resulting in more coherent and preferable text generation. Tree-based methods, including **Tree-of-Thought** (Yao et al., 2023) and **RAP** (Hao et al., 2023), use shallow tree search to explore reasoning paths.

TS-LLM (Feng et al., 2023) uses an AlphaZero-inspired framework with a learned value function, enabling deeper, more adaptable reasoning and improving LLM performance during both inference and training. **ReST-MCTS*** (Zhang et al., 2024) integrates enhancements from ReST and TS-LLM by using tree-search reinforcement learning to infer per-step rewards, enabling higher-quality reasoning trace collection and iterative policy training, and outperforming Best-of-N.

TreeBoN (Qiu et al., 2024) improves upon Best-of-N sampling by using speculative tree-search and token-level DPO rewards to prune low-quality paths, achieving better output quality with lower computational cost and outperforming standard Best-of-N across multiple benchmarks.

s1 (Muennighoff et al., 2025) is a simple test-time scaling method that improves model reasoning by using a curated reasoning dataset (s1K) and a technique called **budget forcing** to control its thinking length. This method forcefully terminates the model's output or extends it by appending a "Wait" token multiple times when the model attempts to end.

B Proofs of Base Results

In this section, we provide detailed proofs of the main and auxiliary results needed to derive the structure of the optimal functions $Q^{*,\eta}$ and $V^{*,\eta}$.

B.1 Notational Preliminaries

We now introduce several notational conventions used frequently throughout the article. For clarity and conciseness, we typically adopt the most compact notation possible, often omitting information when the meaning is clear from context.

Given a set $\mathcal{A}$, for any two elements $x, y \in \mathcal{A}$, we denote that x is preferred over y by $x \succ y$. Therefore, the indicator function that denotes preference between two elements is $\mathbf{1}_{x \succ y}$. Recall we denote the generation starting from a given context by y_x^π according to a policy π. We interchangeably use π to denote the policy, and the LLM model.

Greedy Decoding (GD): Let π be a language model (LLM) policy, a fixed context x, and a target generation length T. The greedy decoding strategy produces an output sequence $y_{\pi;\mathrm{GD}} = (x, a_{0:T-1})$, where each token a_t is deterministically selected according to

$$a_t \in \arg\max_{a \in \mathcal{A}} \pi(a \mid x, a_{0:t-1}), \qquad t = 0, \ldots, T-1. \tag{20}$$

Ties are resolved by a fixed ordering of tokens. The anchor y_x^{ref} uses the fixed policy π_{ref}.

We use $x \in \mathbb{R}^d$ to denote a d-dimensional vector, $M \in \mathbb{R}^{m \times n}$ denotes a matrix of dimension $m \times n$. The semi-norm is defined as $\|u\|_{\Sigma_{\mathcal{D}}+\lambda I} \triangleq \sqrt{u^\top (\Sigma_{\mathcal{D}} + \lambda I) u}$. The d-dimensional all-ones vector is

$$\mathbf{1} = [1, 1, \cdots, 1]^\top \in \mathbb{R}^d$$

The logistic function is denoted by $\sigma(x) \triangleq \frac{1}{1+e^{-x}}$, and its inverse by $\Psi(t) \triangleq \log \frac{t}{1-t}$.

B.2 Optimal Soft Q and V Functions

We provide the proof of optimality for the policy structure considered in equations 6 for the KL-regularized optimization objective 5. This is a well-established result and we only provide it for completeness. With a slight abuse of notation, we denote the optimal policy by π_t^*, and the corresponding optimal soft-value functions by V_t^*, Q_t^*

The KL-regularized objective (1) has the optimizer (Rafailov et al., 2023):

$$\pi_r^*(y \mid x) \propto \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\eta} r(x, y)\right) \tag{21}$$

We can make the following inductive argument for the optimality of the soft-value functions $V_t^{\star,\eta}, Q_t^{\star,\eta}$ starting at a given state $s_t = s$ and action $a_t = a$.

Base case: let $s_{T-1} = s$. Since $V_T^{\star,\eta}(s) = 0$, we have $Q_{T-1}^{\star,\eta}(s,a) = r(s,a)$. Equation 21 gives the optimal policy and soft-value function:

$$\pi_{T-1}^*(a \mid s) \propto \pi_{\text{ref}}(a \mid s) \exp\left(\frac{1}{\eta} Q_{T-1}^{\star,\eta}(s,a)\right) \propto \pi_{T-1}^{*,\eta}(a \mid s)$$

$$\pi_{T-1}^*(a \mid s) = \pi_{T-1}^{*,\eta}(a \mid s) = \frac{\pi_{\text{ref}}(a \mid s) \exp\left(\eta^{-1} Q_{T-1}^{\star,\eta}(s,a)\right)}{\mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} Q_{T-1}^{\star,\eta}(s,a)\right)\right]}$$

$$\begin{aligned}
V_{T-1}^{\star,\eta}(s) &= \mathbb{E}_{\pi^{*,\eta}}\left[r_{T-1}(s,a) - \eta\,\mathrm{KL}(\pi^{*,\eta}(s_{T-1}) \parallel \pi_{\text{ref}}(s_{T-1}))\middle| s_{T-1} = s\right] \\
&= \mathbb{E}_{\pi^*}\left[Q_{T-1}^{\star,\eta}(s,a) - \eta\,\mathrm{KL}(\pi^*(s_{T-1}) \parallel \pi_{\text{ref}}(s_{T-1}))\middle| s_{T-1} = s\right] \\
&= \mathbb{E}_{\pi^*}\left[Q_{T-1}^{\star,\eta}(s,a)\right] - \eta\left[\mathrm{KL}(\pi^*(s) \parallel \pi_{\text{ref}}(s))\right] \\
&= \mathbb{E}_{\pi^*}\left[Q_{T-1}^{\star,\eta}(s,a)\right] - \eta\mathbb{E}_{\pi^*}\left[\eta^{-1} Q_{T-1}^{\star,\eta}(s,a)\right] + \eta \ln \mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} Q_{T-1}^{\star,\eta}(s,a)\right)\right] \\
&= \eta \ln \mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} Q_{T-1}^{\star,\eta}(s,a)\right)\right]
\end{aligned}$$

Inductive step: Suppose $\pi_t^{\star,\eta}$ is optimal from horizon $h+1$ until $T-1$.

$$\begin{aligned}
Q_h^{\star,\eta}(s,a) &= r_h(s,a) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V_{h+1}^{*,\eta}(s')\right] \\
&= r_h(s,a) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V_{h+1}^{*}(s')\right] = Q_h^{\star}(s,a)
\end{aligned}$$

At $s_h = s$, the policy $\pi = (\pi_h, \pi_{h+1}^\star, \ldots, \pi_{T-1}^\star)$ satisfies

$$\begin{aligned}
V_h^{\pi,\eta}(s) &= \mathbb{E}_\pi\left[\sum_{t=h}^{T-1}\left[r_t(s_t,a_t) - \eta\,\mathrm{KL}(\pi(\cdot \mid s_t) \parallel \pi_{\text{ref}}(\cdot \mid s_t))\right]\middle| s_h = s\right] \\
&= \mathbb{E}_{\pi_h}\Bigg[r_h(s,a) - \eta\,\mathrm{KL}(\pi_h(s_h) \parallel \pi_{\text{ref}}(s_h)) \\
&\qquad + \mathbb{E}_{\pi^*}\left[\sum_{t=h+1}^{T-1}\left[r_t(s_t,a_t) - \eta\,\mathrm{KL}(\pi^*(\cdot \mid s_t) \parallel \pi_{\text{ref}}(\cdot \mid s_t))\right]\middle| s_h = s, a_h = a\right]\Bigg] \\
&= \mathbb{E}_{\pi_h}\left[r_h(s,a) - \eta\,\mathrm{KL}(\pi_h(s_h) \parallel \pi_{\text{ref}}(s_h)) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V_{h+1}^*(s')\right]\right] \\
&= \mathbb{E}_{\pi_h}\left[Q_h^{\star,\eta}(s,a) - \eta\,\mathrm{KL}(\pi_h(s_h) \parallel \pi_{\text{ref}}(s_h))\right]
\end{aligned}$$

Using the results of Rafailov et al. (2023) by setting $r(s,a) = Q_h^\star(s,a)$,

$$V_h^\star(s) = \max_{\pi_h} \mathbb{E}_{\pi_h}\left[Q_h^\star(s,a) - \eta\,\mathrm{KL}(\pi(s_h) \parallel \pi_{\text{ref}}(s_h))\right] = V_h^{\star,\eta}(s)$$

For deterministic MDPs, the optimal soft-value functions have a simple form in terms of the cumulative rewards from rollouts according to π_{ref} Zhou et al. (2025) as shown below.

$$\begin{aligned}
\exp(\eta^{-1} V_t^{\star,\eta}(s)) &= \mathbb{E}_{\pi^{\text{ref}}}\left[\exp\left(\eta^{-1} r_t(s_t,a_t) + \eta^{-1} V_{t+1}^{\star,\eta}(s_{t+1})\right)\middle| s_t = s\right] \\
&= \mathbb{E}_{\pi^{\text{ref}}}\left[\exp\left(\eta^{-1}\sum_{u=t}^{T-1} r_u(s_u,a_u)\right)\middle| s_t = s\right]
\end{aligned}$$

Taking logarithm on both sides, we get

$$V_t^{\star,\eta}(s) = \eta \ln \mathbb{E}_{\pi^{\text{ref}}}\left[\exp\left(\eta^{-1}\sum_{u=t}^{T-1} r_u(s_u,a_u)\right)\middle| s_t = s\right]$$

The result for $Q_t^{\star,\eta}$ follows a similar line of analysis.

B.3 Proof of Lemma 1

Proof. Fix the greedy anchor y_x^{ref} under π_{ref} and let $\tilde{r}_x(y) = r^*(y) - r^*(y_x^{\text{ref}})$. This context-dependent shift preserves every reward difference, as in the reward-equivalence property of Rafailov et al. (2023, Lemma 1). By the Bradley–Terry model,

$$\mathcal{P}^*(y \succ y_x^{\text{ref}}) = \sigma\big(r^*(y) - r^*(y_x^{\text{ref}})\big) = \sigma(\tilde{r}_x(y)).$$

□

B.4 Proof of Theorem 1

Proof. The inverse function of the logistic function $\sigma(x)$ is given by $\Psi(x) \triangleq \log \frac{x}{(1-x)}$. Suppose that the Bradley-Terry model holds. For any y, y' given a context x, we have:

$$\begin{aligned}\Psi\left(\mathcal{P}^*(y \succ y'|x)\right) &= \Psi(\sigma(r(y) - r(y'))) \\ &= r(y) - r(y') \\ r(y) &= r(y') + \Psi\left(\mathcal{P}^*(y \succ y'|x)\right).\end{aligned}$$

Setting $y := s_T$, and $y' := y_s^{\text{ref}}$, the above equivalence of the reward function to the preference distribution yields the following expressions for the optimal value function $V_t^{\star,\eta}$

$$\begin{aligned}V_t^{\star,\eta}(s) &= \eta \ln \mathbb{E}_{\pi_{\text{ref}}}\left[\exp\left(\eta^{-1} r(s_{T-1}, a_{T-1})\right) \middle| s_t = s\right] && (22)\\ &= \eta \ln \mathbb{E}_{y_s \sim \pi_{\text{ref}}}\left[\exp\left(\eta^{-1}\left[r(y_s^{\text{ref}}) + \Psi\left(\mathcal{P}^*(y_s \succ y_s^{\text{ref}})\right)\right]\right) \middle| s_t = s\right], && (23)\\ &= r(y_s^{\text{ref}}) + \eta \ln \mathbb{E}_{y_s \sim \pi_{\text{ref}}}\left[\exp\left(\eta^{-1}\left[\Psi\left(\mathcal{P}^*(y_s \succ y_s^{\text{ref}})\right)\right]\right) \middle| s_t = s\right], && (24)\\ & && (25)\end{aligned}$$

Conditioning additionally on $a_t = a$ yields the expression for $Q_t^{\star,\eta}$ with the same parent-state anchor y_s^{ref}. Its reward offset is shared across candidate actions and cancels when normalizing the policy. □

C Convergence of PITA

C.1 Bounding the MLE estimation error

We first establish curvature of the logistic loss, then bound its gradient at the true parameter. The gradient bound uses the conditional Bradley–Terry model with adaptively chosen comparisons.

Lemma 3 (Curvature of the MLE loss)**.** *For $\theta^\star, \theta^\star + \Delta \in \Theta_B$,*

$$\mathcal{L}_{\mathcal{D}}(\theta^\star + \Delta) - \mathcal{L}_{\mathcal{D}}(\theta^\star) - \langle \nabla \mathcal{L}_{\mathcal{D}}(\theta^\star), \Delta \rangle \geq \frac{\gamma}{2} \|\Delta\|_{\Sigma_{\mathcal{D}}}^2, \tag{26}$$

where $\gamma = (2 + e^{-2LB} + e^{2LB})^{-1}$ and $\|u\|_A^2 = u^\top A u$.

Proof. Let $z_i = \psi_{x_i}(y_i)$. The normalized negative log-likelihood is

$$\mathcal{L}_{\mathcal{D}}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \left[o_i \log \sigma(\langle \theta, z_i \rangle) + (1 - o_i) \log\big(1 - \sigma(\langle \theta, z_i \rangle)\big)\right].$$

Its Hessian is

$$\nabla^2 \mathcal{L}_{\mathcal{D}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \sigma(\langle \theta, z_i \rangle)\big(1 - \sigma(\langle \theta, z_i \rangle)\big) z_i z_i^\top.$$

The centered features satisfy $\|z_i\|_2 \leq 2L$, and hence $|\langle \theta, z_i \rangle| \leq 2LB$ on Θ_B. Thus

$$\sigma(\langle \theta, z_i \rangle)\big(1 - \sigma(\langle \theta, z_i \rangle)\big) \geq \gamma, \qquad \nabla^2 \mathcal{L}_{\mathcal{D}}(\theta) \succeq \gamma \Sigma_{\mathcal{D}}.$$

Taylor's formula along the segment from $\theta^\star$ to $\theta^\star + \Delta$ gives the bound with factor $1/2$. □

Proof of Lemma 2. Write $\varepsilon_i = o_i - \sigma(\langle \theta^\star, z_i \rangle)$, $S_n = \sum_{i=1}^n \varepsilon_i z_i$, and $A_n = \sum_{i=1}^n z_i z_i^\top$. Given the history and comparison z_i, ε_i has mean zero and lies in $[-1, 1]$. Thus,

$$\mathbb{E}[\exp(t\varepsilon_i) \mid \text{history}, z_i] \leq \exp(t^2/2).$$

For any fixed $u \in \mathbb{R}^d$, successive conditioning gives

$$\mathbb{E}\left[\exp\left(u^\top S_n - \tfrac{1}{2} u^\top A_n u\right)\right] \leq 1.$$

Fix $a > 0$ and integrate this inequality against $u \sim \mathcal{N}(0, a^{-1} I)$. Completing the square yields

$$\mathbb{E}\left[\frac{\exp\left(\frac{1}{2}\|S_n\|^2_{(A_n + aI)^{-1}}\right)}{\det(I + A_n/a)^{1/2}}\right] \leq 1.$$

Markov's inequality, with $a = n\lambda$, now gives, with probability at least $1 - \delta$,

$$\begin{aligned}\|\nabla\mathcal{L}_{\mathcal{D}}(\theta^\star)\|^2_{(\Sigma_{\mathcal{D}}+\lambda I)^{-1}} &= \frac{1}{n}\|S_n\|^2_{(A_n+n\lambda I)^{-1}} \\ &\le \frac{\log\det(I + \Sigma_{\mathcal{D}}/\lambda) + 2\log(1/\delta)}{n} \\ &\le \frac{d\log(1 + 4L^2/(d\lambda)) + 2\log(1/\delta)}{n}. \end{aligned} \tag{27}$$

The last inequality follows from $\mathrm{tr}(\Sigma_{\mathcal{D}}) \le 4L^2$ and concavity of the logarithm.

Set $\Delta = \hat{\theta} - \theta^\star$ and $H = \Sigma_{\mathcal{D}} + \lambda I$. MLE optimality, Lemma 3, and norm duality give

$$\frac{\gamma}{2}\|\Delta\|^2_{\Sigma_{\mathcal{D}}} \le -\langle\nabla\mathcal{L}_{\mathcal{D}}(\theta^\star), \Delta\rangle \le \|\nabla\mathcal{L}_{\mathcal{D}}(\theta^\star)\|_{H^{-1}}\|\Delta\|_H.$$

Since $\|\Delta\|_2 \le 2B$, writing $g = \|\nabla\mathcal{L}_{\mathcal{D}}(\theta^\star)\|_{H^{-1}}$ gives

$$\|\Delta\|^2_H \le \frac{2g}{\gamma}\|\Delta\|_H + 4\lambda B^2, \qquad \|\Delta\|_H \le \frac{2g}{\gamma} + 2B\sqrt{\lambda}.$$

Combining this with Equation (27) proves the lemma for a universal constant C. □

C.2 Proof of Theorem 2

We express the learned policy's value under the true reward using a log-partition function and a linear correction. Both terms are controlled by the same estimation radius and coverage factor.

Proof. For a fixed context x, define

$$F_x(\theta) = \eta\log\mathbb{E}_{y\sim\pi_{\mathrm{ref}}(\cdot|x)}[\exp(\langle\theta, \psi_x(y)\rangle/\eta)], \qquad \mu_\theta(x) = \mathbb{E}_{y\sim\pi^{\theta,\eta}(\cdot|x)}[\psi_x(y)].$$

Differentiating the log-partition function gives

$$\nabla_\theta F_x(\theta) = \mu_\theta(x). \tag{28}$$

The Gibbs policy satisfies

$$\eta\log\frac{\pi^{\theta,\eta}(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} = \langle\theta, \psi_x(y)\rangle - F_x(\theta).$$

Since $r^*(x, y) = r^*(x, y^{\mathrm{ref}}_x) + \langle\theta^\star, \psi_x(y)\rangle$, its KL-regularized value under the true reward is

$$V^{\pi^{\theta,\eta},\eta}(x) = r^*(x, y^{\mathrm{ref}}_x) + F_x(\theta) + \langle\theta^\star - \theta, \mu_\theta(x)\rangle. \tag{29}$$

In particular, $V^{\star,\eta}(x) = r^*(x, y^{\mathrm{ref}}_x) + F_x(\theta^\star)$. Thus, for the round-$k$ estimate,

$$\begin{aligned} V^{\star,\eta}(x) - V^{\pi^{\theta_k,\eta},\eta}(x) &= F_x(\theta^\star) - F_x(\theta_k) - \langle\theta^\star - \theta_k, \mu_{\theta_k}(x)\rangle \\ &= \langle\theta^\star - \theta_k, \mu_{\bar{\theta}}(x) - \mu_{\theta_k}(x)\rangle, \end{aligned} \tag{30}$$

Here $\bar{\theta} \in \Theta_B$ lies on the segment from θ_k to $\theta^\star$. Norm duality and Assumption 2 give

$$V^{\star,\eta}(x) - V^{\pi^{\theta_k,\eta},\eta}(x) \le 2G_k\|\theta_k - \theta^\star\|_{\Sigma_k+\lambda_k I}.$$

Apply Lemma 2 with confidence level δ/K at each deterministic sample count n_k. A union bound gives, simultaneously for all $k \le K$, with probability at least $1 - \delta$,

$$\|\theta_k - \theta^\star\|_{\Sigma_k+\lambda_k I} \le C\sqrt{\frac{d\log(1 + 4L^2/(d\lambda_k)) + \log(K/\delta)}{\gamma^2 n_k} + \lambda_k B^2}.$$

Taking the expectation over $x \sim \rho$ and summing over k proves the stated bound, with the factor 2 absorbed into C. For $n_k = mk$ and $\lambda_k = 1/(mk)$, use $G_k \le G$ and $\sum_{k=1}^K k^{-1/2} \le 2\sqrt{K}$. □

D MATH REASONING, STAR GRAPH REASONING, AND SENTIMENT GENERATION

In this section we present our results for sentiment generation tasks Chaudhary et al. (2025), star graph reasoning, and provide the training details for PITA on GSM8K.

D.1 MATH REASONING

Dataset. **GSM8K** contains 7,500 training and 1,319 test problems. We create a $90\%/10\%$ split of the original training set for learning and validation, reserving the full test set for final reporting.

Models. We use the **Llama-3 8B-Instruct** model Grattafiori et al. (2024) as our reference policy and the **Llama-3.2 1B-Instruct** model as the backbone for our preference guide. For Q#-HF, the reward model trained from preference data is also a Llama-3.2 1B-Instruct. We use `nvidia/OpenMath2-Llama3.1-8B`, trained on OpenMathInstruct-2 Toshniwal et al. (2024), as the judge for response pairs with equal answer correctness.

Training details. We collect pairwise samples for each question in the training set of GSM8K and label every sample either as *'preferred'* (1) or *'not preferred'* (0) based on the quality of final answer. Note that for arithmetic reasoning, there is an exact numeric answer for each question. A correct answer is automatically preferred over a wrong answer. When both responses have the same correctness label, the OpenMath2 judge assigns the preference. PITA uses two rounds of data collection and guide fitting, with five training epochs per round, a learning rate of 2×10^{-5}, and a context length of 4096. The model is retrained from scratch at the start of each round, using data collected from both the current and all previous rounds. The baseline model is trained with identical hyperparameters. Each optimization run is performed on a single NVIDIA A100 GPU.

Evaluation protocol. We estimate single-sample accuracy (*pass@1*) by averaging correctness across sampled responses, and report majority-vote accuracy over k=8 responses (*maj@8*). Generations use temperature $T = 0.8$ and nucleus sampling $p = 0.9$. We use the inference optimizations for value function V given in Zhou et al. (2025). All evaluations use zero-shot prompting. At each generated prefix, the math evaluator computes forward KL from the processed guided distribution to the temperature-scaled reference distribution. It sums token KL over nonpadding generated positions and averages the resulting values across responses.

The prompt template used for evaluation is: `'Problem:\n\n{0} Write your answer inside \\boxed{{}}.\n\nSolution:'` where `{0}` is replaced by the actual question from the dataset.

D.2 STAR-GRAPH REASONING

To better understand the effectiveness of our method in guiding a reference model at test time, we analyze the behavior of PITA alongside other value-guided algorithms on star-graph reasoning tasks (Bachmann & Nagarajan, 2024).

A star-graph $\mathcal{G}(d, l)$ is defined as a tree with d paths of length l emanating from a common root node. The task is for the language model (LM) to generate the correct path from the root to a leaf, given the edge set defining the graph. Figure 3 illustrates the star-graph task.

We compare frozen-base decoding methods (PITA, Q#, and CD) with methods that update the base model (REINFORCE, DPO, and RPO):

- **CD** Mudgal et al. (2023): A controlled decoding algorithm trained with access to true binary reward labels.
- **REINFORCE** (Ahmadian et al., 2024): A post-training algorithm that uses REINFORCE instead of the canonical Proximal Policy Optimization (PPO) in reinforcement learning from human feedback.
- **DPO** (Rafailov et al., 2023): A post-training algorithm that aligns language models with human preferences by optimizing for preferred responses over less preferred ones, without using reinforcement learning.

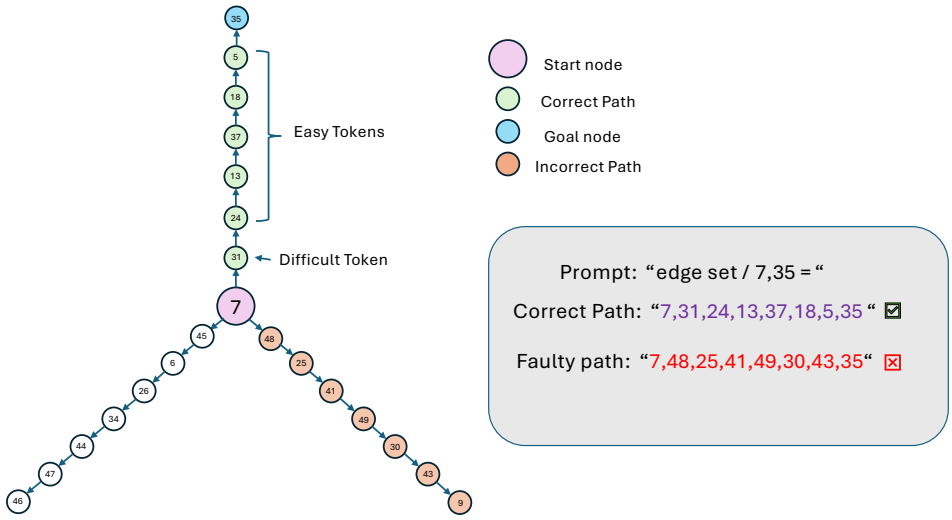

Figure 3: A $\mathcal{G} = (3, 8)$ star-graph configuration with degree $d = 3$ and path length $l = 8$. The pre-trained model, using next-token prediction, learns a faulty shortcut: it selects a random first node and follows the path from there.

- **RPO** (Pang et al., 2024): A post-training algorithm that modifies the DPO loss to improve performance on reasoning tasks.

Despite the task's structural simplicity, Bachmann & Nagarajan (2024) highlight a key failure mode in models trained via next-token prediction. Specifically, the model often learns a *Clever-Hans* shortcut by selecting the first node in the root's adjacency list and following it to a leaf. While this heuristic yields perfect accuracy on the training set, it fails to generalize: on unseen test graphs, the model selects the correct path only with probability $1/d$, as the first node is correct with probability $1/d$.

Zhou et al. (2025) show that this faulty behavior can be corrected during post-training using value-based methods. However, widely used policy-based approaches such as REINFORCE, DPO, and RPO fail to overcome this shortcut, achieving test accuracy no better than $1/d$. This limitation is likely due to the difficulty transformer-based architectures face in unlearning spurious heuristics via policy-gradient optimization (Hu et al., 2024).

Main Results: In Table 5, we report the test accuracies of PITA on three classes of graphs: smaller graphs $\mathcal{G}(2, 5)$ and $\mathcal{G}(5, 5)$, where we use the GPT-2 small model, and a larger graph $\mathcal{G}(3, 8)$, where we use the GPT-2 medium model Radford et al. (2019) to train the set of pre-trained models. We observe that **PITA** effectively guides the reference LM to achieve near-perfect generalization accuracy on the star-graph task, outperforming strong value-based baselines such as Q# and CD, which themselves achieve high generalization performance.

Table 5: Star-graph test accuracy (%). Q#, CD, and PITA guide a frozen base model. REINFORCE, DPO, and RPO update its weights.

Algorithm	$\mathcal{G}(2, 5)$	$\mathcal{G}(5, 5)$	$\mathcal{G}(3, 8)$
pre-trained	49.94	19.8	33.24
REINFORCE	49.19	13.13	25.96
DPO	37.61	0.00	0.00
RPO	49.69	19.93	33.47
Q#	99.90	97.00	98.62
CD	99.94	98.62	99.52
PITA (ours)	99.97	99.70	99.93

PITA achieves test accuracies of 99.97%, 99.70%, and 99.93% on the three graph configurations. These results show that preference guidance can recover the correct path from the frozen model's generation distribution. The weight-updating baselines exhibit the shortcut behavior reported in Zhou et al. (2025), while all three decoding guides attain high accuracy on this synthetic task.

We now describe the training methodology used to apply **PITA** to the star-graph task.

Training Setup: We follow the experimental setup of Q# Zhou et al. (2025) to generate and evaluate the star-graph datasets. All models are initially pretrained using next-token prediction on a dataset of 200k randomly sampled graphs paired with their correct paths.

We consider several baselines, including **REINFORCE**, **DPO**, **RPO**, as well as value-based methods **Q#** and **CD**. Post-training is conducted on a separate dataset of 200k new random graphs. To ensure a fair comparison, we reproduce the results with an identical reward structure and reuse the datasets used in Zhou et al. (2025). For REINFORCE, a reward of 1 is assigned for correct paths and -0.1 for incorrect ones. For DPO and RPO, pairwise comparisons between a correct path y_{chosen} and an incorrect shortcut path y_{reject} are sampled from the pre-trained model. Similarly, for **Q#**, a classifier is trained on the same pairwise dataset, labeling correct paths with reward 1 and incorrect ones with 0. Note that PITA uses a unique reference generation at the core of its algorithm. In PITA, we treat correct paths as the *'preferred'* solution and the incorrect path as the *'not-preferred'* answer to estimate the preference distribution. The unique correct answer naturally serves as the reference completion for the star-graph task.

All models are trained for 10 epochs using the AdamW optimizer with a weight decay of 0.1 and a batch size of 256. The learning rates are set to 2.5×10^{-4} for pretraining, 1×10^{-5} for REINFORCE, and 1×10^{-4} for DPO, RPO, CD, and Q#. For Q# and CD, we use the star-graph implementation's decoding parameter $\eta_{\text{graph}} = 0.1$. Training is performed on a single A100 GPU. Evaluation is conducted on a held-out test set of 20k graphs using top-$k = 10$ sampling and temperature 1.0.

D.3 IMDB Sentiment Generation

In this section we evaluate **PITA** on controlled sentiment generation. The task follows the IMDB-Gen setting of Ramamurthy et al. (2023) and Chaudhary et al. (2025). Given the first 60 tokens of a movie review, the language model continues the review to maximize positive sentiment.

Dataset. We use the **IMDB Reviews** dataset, which contains $25{,}000$ training and $25{,}000$ test reviews, each labeled *positive* or *negative*. For each example, we truncate the review to its first 60 tokens, which form the prompt x; the model then generates up to 60 continuation tokens y. The reward model for this task is the **lvwerra/distilbert-imdb** classifier, with the logit corresponding to the positive sentiment output node used as the scalar reward. For the creation of the preference dataset, we select the generation with the higher reward as the preferred generation.

Models. For this task, we use **TinyLlama-1.1B-Chat-v1.0**[1] as both our reference policy and the backbone for our value function. For the reward model learned from the preference data for Q#-HF, we use a **LLama 3.2 1B-Instruct** model.

Evaluation We evaluate the same set of algorithms as above. For each of the $5{,}000$ **IMDB** test set prompts, we sample a *single* continuation using nucleus sampling with top-$p = 0.9$ and temperature $T = 0.8$. The full prompt + completion is scored by the `lvwerra/distilbert-imdb` sentiment classifier. We report the mean sentiment logit score (SENT.↑) and the token-level KL divergence (KL↓) between the guided policy and the reference model π_{ref}. An example of the completions generated by π_{ref} and by PITA is shown in Figure 4.

Discussion. Using the fixed sentiment classifier's scalar scores, Q# achieves the highest mean score, 0.22. Among the preference-supervised methods, PITA raises the mean score to 0.01, compared with -0.44 for Q#-HF and -0.52 for the reference model. Thus, direct preference guidance produces a larger shift toward positive sentiment than the two-stage reward-fitting pipeline in this experiment.

[1] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

Table 6: IMDB-Gen sentiment score and KL on 5,000 test prompts. Q# uses scalar classifier scores. Q#-HF and PITA use preferences constructed from those scores, with a separate reward fit in Q#-HF.

Algorithm	SENT.↑	KL↓
π_{ref}	-0.52	-
Alignment from scalar rewards		
Q#	0.22	7.51
Alignment from preference pairs		
Q#-HF	-0.44	5.22
PITA (ours)	0.01	7.27

Prompt: Paul Mazursky misfires on this film. The writing, direction, casting, and acting (with the exception of Victorio Gassman) are all off the mark. I remember the reviews from 20+ years ago being mediocre, but I thought it still might

---

Completion from π^{ref}: be good. But by today's standards, the film just isn't good enough. It's a muddled and overly long (about 2 1/2 hours) film that ultimately does not live up to its potential as a classic film noir.

Completion from PITA: be good. I'm glad to see how far it's come. This is not a great film, but it's definitely a lot better than the original. I think that makes it more enjoyable to watch. The acting is excellent, with a lot of great performances by both Mar

Figure 4: IMDB continuations (60-token limit). PITA produces a more positive review.

D.4 AlpacaEval with Noisy Preference Training

In this section, we evaluate **PITA** on instruction-following generation using noisy human preference data. Here the objective is to generate open-ended responses preferred by human annotators. We use the AlpacaEval framework Dubois et al. (2024) as an automatic proxy for human evaluation.

Dataset. For training Q# and PITA, we use the `noisy_multi_preference` subset of the offline human-labeled dataset from AlpacaFarm Dubois et al. (2023). Each example consists of an instruction, a pair of candidate responses, and a human preference label. The preference labels are inherently noisy, since they are collected from multiple human annotators. For Q#, we convert the preference label into a binary reward for each response, assigning 1 to the preferred response and 0 otherwise.

Models. We evaluate the reference model π_{ref}, Q#, and PITA under the same instruction-following setting. PITA is trained directly from pairwise preference labels, while Q# uses the same labels after converting them into binary rewards. This comparison isolates whether preference feedback is better used directly through PITA or indirectly through reward-style supervision.

Evaluation. We evaluate using AlpacaEval over 805 queries. Candidate responses are compared against baseline responses generated by `text_davinci_003`. For our judge, we use the `weighted_alpaca_eval_vllm_llama3_70b` annotator. Specifically, a locally hosted `Meta-Llama-3-70B-Instruct` model is shown the query and two responses, and it is instructed to output one of two tokens indicating which response is preferred. The log probabilities assigned to these tokens are used to determine the final binary preference. We report length-controlled win rates (LCWR) for the Llama and Qwen models.

Discussion. PITA improves length-controlled win rate over the reference model by 0.31 percentage points for Llama and 1.58 points for Qwen. Both PITA and Q# use the same noisy preference labels.

Table 7: AlpacaEval results using noisy human preference training data. We report length-controlled win rates against `text_davinci_003` baseline responses over 805 evaluation queries.

Algorithm	LLAMA (LCWR) (%) ↑	QWEN (LCWR) (%) ↑
π_{ref}	46.91	37.28
Q# (binary preference labels)	47.15	36.86
PITA (ours)	47.22	38.86

PITA achieves the higher win rate for both models, supporting the use of direct preference guidance for this instruction-following task.

E Additional Experimental Details

E.1 Reward-Guided Decoding on GSM8K

To compare the guidance learned from preferences with alternative reward-based guides, we also evaluate controlled decoding (CD) (Mudgal et al., 2023) on GSM8K. Both CD and Q# use task rewards to train their guides, while PITA learns its guide from preference pairs. Table 8 separates these sources of supervision. At the reported operating points, Q# achieves higher pass@1 and maj@8 than CD with lower measured KL. PITA attains 77.11% pass@1 from preference pairs.

Table 8: GSM8K accuracy (%) and measured forward KL. The grouping distinguishes direct task-reward supervision from preference supervision.

Algorithm	pass@1	maj@8	KL
Alignment from exact rewards			
Q#	78.4	88.1	2.65
CD	77.8	87.2	6.39
Alignment from preference pairs			
PITA (ours)	77.11	86.20	6.15

E.2 Ablation Study on PITA training for GSM8K

We benchmark **PITA** on the GSM8K grade-school arithmetic corpus Cobbe et al. (2021), which was also considered in the main paper. In contrast to the main paper, we evaluate PITA with only a single round of training i.e. ($k = 1$ in Algorithm 1). We present our results in Table 9.

Table 9: GSM8K test accuracy (%) in the single-round ablation. Q# uses task rewards. Q#-HF and PITA use preference pairs, with an intermediate reward fit in Q#-HF.

Algorithm	pass@1 ↑	maj@8 ↑	KL
π_{ref}	69	85	-
Q#	78.4	88.1	2.65
Q#-HF	49.9	72.0	18.85
PITA (ours)	74.8	86.1	18.4

With one training round, PITA achieves 74.8% pass@1 and 86.1% maj@8, exceeding Q#-HF by 24.9 and 14.1 percentage points at comparable measured KL (18.4 versus 18.85). Using two rounds increases PITA's pass@1 to 77.11% and maj@8 to 86.20%, as reported in Table 1. The second round expands the preference data using the updated guide.

Evaluation across random seeds. Across eight random seeds, PITA obtains mean GSM8K pass@1 of 76.62% and mean maj@8 of 85.72%. The reported spreads are 0.99 and 0.31 percentage points, respectively. The mean pass@1 is 7.52 percentage points above the reference model.

E.3 Reward-Model Calibration in Q#-HF:

In Q#-HF, we first train a reward model as an intermediate step before applying the Q# algorithm to derive a policy. The results reported in Table 9 are obtained using a single round of training, corresponding to 50% of the data used in the main paper. To ensure a fair comparison with other methods, the reward model for Q#-HF is also trained using data from a single round, so that the total amount of data used across reward estimation and policy optimization remains comparable.

Figures 5 and 6 show the learned Bradley–Terry reward scores for correct and incorrect baseline generations under full and limited preference data.

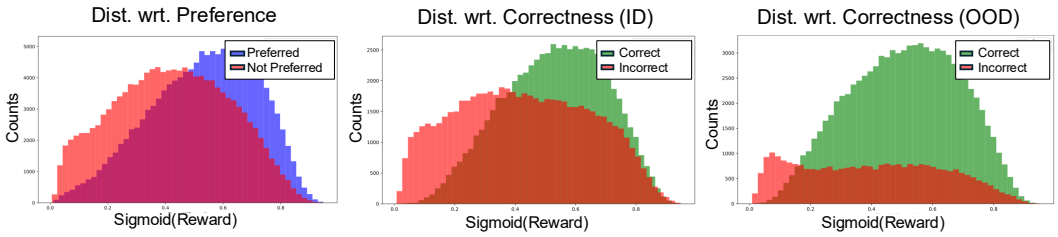

Figure 5: Full preference data: Distribution of learned reward scores for preferred/correct (green) and not preferred/incorrect (red) generations on in-distribution (ID) and out-of-distribution (OOD) data.

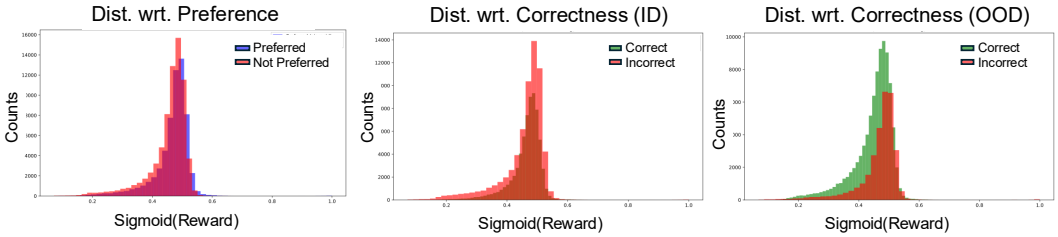

Figure 6: Limited preference data (single-round): Distribution of learned reward scores for preferred/-correct (green) and not preferred/incorrect (red) generations on ID and OOD data.

Q#-HF's lower accuracy in the single-round experiment accompanies weaker reward separation between correct and incorrect generations (Figure 6). The full-data reward model produces clearer separation (Figure 5). This comparison illustrates how the intermediate reward fit can affect the quality of the resulting guide.

E.4 Boundedness of Hidden Representation

We report the norms of final-layer hidden representations during training and evaluation to characterize their observed scale. Gradient norms are reported for the guidance model trained by PITA.

Training Phase

- **Hidden representations:** Reference Model: mean 155.15, std 5.03, min/max 93.34/162.77; Guidance Model: mean 48.10, std 8.74, min/max 34.70/107.52.
- **Gradient norms:** Guidance Model: mean 11.19, std 8.42, min/max 0.69/87.5.

Evaluation Phase

- **Hidden representations:** Reference Model: mean 155.13, std 5.06, min/max 90.39/162.56; Guidance Model: mean 46.11, std 5.49, min/max 38.36/88.77.

The reference-model hidden norms have similar ranges during training and evaluation. The guidance model has a wider spread during training, with its maximum norm decreasing from 107.52 in training to 88.77 in evaluation. These measurements characterize the representation scales observed in the 7B reference model and the 1B guidance model.

Normalization of Hidden Representations LLMs commonly use RMSNorm or LayerNorm to control the scale of inputs within a transformer block. For example,

$$\mathrm{RMSNorm}(h) = \frac{h}{\sqrt{\frac{1}{d}\sum_i h_i^2 + \varepsilon}} \cdot \gamma,$$

where γ is a learned scale parameter. The norm statistics above describe the resulting representations in our training and evaluation runs.

E.5 Sensitivity to Number of Linear Heads and Data Requirements

We further study the sensitivity of PITA to the capacity of the preference-guidance model and the amount of preference data used during refinement. In addition to the refinement-round ablation in Table 9, we evaluate different numbers of linear heads while keeping the data scale fixed as in the original experiments. Table 10 shows that five heads attain 76.01% pass@1 at $\alpha = 4$, while three heads attain 74.76% at $\alpha = 2$. Both configurations attain their best pass@1 at intermediate α.

Table 10: GSM8K accuracy (%) for three- and five-head preference guides.

	Three heads		Five heads	
α	pass@1	maj@8	pass@1	maj@8
0.5	66.75	79.45	66.29	80.59
1	70.43	82.64	70.23	81.65
2	74.76	85.29	75.36	86.35
4	74.72	85.82	76.01	86.81
6	65.97	84.76	74.72	86.73

We also study the effect of the number of sampled completions per problem, which determines the number of preference pairs available for training. Specifically, we compare training with 16, 8, and 4 sampled completions per example. The results are summarized below:

- **Sample count = 16:** pass@1 $= 77.11$, maj@8 $= 86.20$.
- **Sample count = 8:** pass@1 $= 68.69$, maj@8 $= 82.03$.
- **Sample count = 4:** pass@1 $= 42.10$, maj@8 $= 75.05$.

Increasing from 8 to 16 sampled completions improves pass@1 by 8.42 percentage points and maj@8 by 4.17 points. The reference model achieves 69.10% and 85.80%, respectively. Sampling more completions provides more preference pairs for guide training.

E.6 Training and Inference Costs

We measure the cost of training the auxiliary preference guide and using it to steer a frozen base model. We report a 1B-scale training comparison with DPO, followed by fixed-step training and evaluation measurements for PITA, Q#, and CD.

Training requirements at the 1B scale. For a direct cost comparison with standard preference optimization, we train both PITA and DPO with LoRA using a LLaMA 1B model, on the same dataset, and using a single NVIDIA A100 GPU.

- **PITA (LLaMA 1B):** Training required 5,494 seconds, corresponding to 1.53 GPU-hours, over 19,875 optimization steps. Each step processes 4,096 tokens (81.4M in total).
- **DPO with LoRA (LLaMA 1B):** Training required 13,159 seconds, corresponding to 3.66 GPU-hours, over 44,173 optimization steps. Each step processes 2,048 tokens (90.5M in total).

For these training schedules, PITA uses 1.53 GPU-hours and DPO uses 3.66 GPU-hours. The step and token counts above specify the workload for each measurement.

A100 inference compute estimate. Inference was benchmarked on the GSM8K test set with 1,319 examples using a single NVIDIA A100 GPU. Reference decoding uses the LLaMA 8B policy. For PITA, inference uses the same LLaMA 8B base model together with a LLaMA 1B guidance model that modifies logits at decoding time. The estimated inference compute is:

- **PITA inference:** estimated compute cost of 2.91×10^{17} FLOPs.
- **Reference model inference:** estimated compute cost of 2.60×10^{17} FLOPs.

Fixed-step training and evaluation with auxiliary guides. In Tables 11 and 12, we further compare PITA against guidance-based methods that also use small auxiliary models, namely Q# Zhou et al. (2025) and CD Mudgal et al. (2023). These measurements cover guide fitting and the associated evaluation runs. All methods are run on a single NVIDIA H100 GPU for one epoch of training and evaluated on GSM8K. The reference model, guidance model, dataset, and number of optimizer steps are kept fixed across methods.

A step-level record corresponds to one logged optimizer update after gradient accumulation. We measure wall-clock time and throughput for each update interval. Evaluation uses eight completions for each of the 1,319 GSM8K test problems. Per-completion latency includes the prompt and generation, with the total evaluation latency also including KL computation. Reported spreads are standard deviations across completions.

Table 11: Guide-fitting efficiency for the fixed-step H100 runs. Percentages are relative to PITA.

Metric	PITA (ours)	Q#	CD
Average step time (s)	0.1519	0.1581 (+4.1%)	0.1525 (+0.4%)
Step throughput (examples/s)	56.12	53.80 (-4.1%)	55.82 (-0.5%)
Step throughput (tokens/s)	12,401	11,895 (-4.1%)	12,339 (-0.5%)
Per-example training latency (s)	0.01898	0.01976 (+4.1%)	0.01906 (+0.4%)
Run-level throughput (examples/s)	47.48	44.41 (-6.5%)	45.74 (-3.7%)
Run-level throughput (tokens/s)	10,609	9,921 (-6.4%)	10,218.65 (-3.7%)
Overall training cost (GPU-hours)	0.472	0.505 (+6.9%)	0.490 (+3.8%)
Overall training wall-clock time (s)	1,699.94	1,817.80 (+6.9%)	1,764.89 (+3.8%)

Table 12: H100 evaluation efficiency for eight completions per problem (10,552 completions). Latency and token statistics are per completion, with mean $\pm$ standard deviation.

Metric	PITA (ours)	Q#	CD
Overall evaluation wall-clock time (s)	10,291	9,979.10	10,452
Overall evaluation cost (GPU-hours)	2.86	2.77	2.90
Total generated tokens	1,559,842	1,511,414	1,517,496
Evaluation latency per completion (s)	0.9753 ± 0.4050	0.9457 ± 0.3889	0.9905 ± 0.4644
Generation-only latency per completion (s)	0.9022 ± 0.3649	0.8758 ± 0.3535	0.9201 ± 0.4218
Generated tokens per completion	147.82 ± 67.37	143.23 ± 65.96	143.81 ± 70.06
Run-level throughput (tokens/s)	151.58	151.46	145.18
Guidance calls per completion	31.60 ± 11.85	30.61 ± 11.45	31.37 ± 13.34
Reference-only compute estimate (FLOPs)	2.505×10^{16}	2.427×10^{16}	2.437×10^{16}
Guided compute estimate (FLOPs)	1.525×10^{17}	1.438×10^{17}	1.484×10^{17}

PITA's guide-fitting run takes 0.472 GPU-hours, compared with 0.505 for Q# and 0.490 for CD. Evaluation takes 2.86, 2.77, and 2.90 GPU-hours, respectively. Thus, the three auxiliary-guidance methods have similar measured costs for these workloads.

Reference decoding budgets. Table 13 pairs accuracy and time from the same efficiency-study checkpoints. For each guided method, dividing the eight-completion evaluation time by eight gives a single-completion-equivalent estimate. Ref@1 and Ref@2 are measured runs with one and two completions per problem. Ref@2 accuracy is oracle pass@2, the fraction of problems with at least one correct completion. PITA achieves 69.29% single-sample accuracy at a normalized evaluation time of 1,286.36 seconds, between the two reference decoding budgets.

Table 13: Accuracy and evaluation time for the efficiency-study checkpoints. Guided times are normalized single-completion equivalents. Reference times are measured totals.

Method	Accuracy (%)	Evaluation time (s)
Ref@1 (pass@1)	58.45	726.26
Ref@2 (oracle pass@2)	74.75	1,420.11
PITA (ours) (pass@1)	69.29	1,286.36
Q# (pass@1)	62.63	1,247.39
CD (pass@1)	61.81	1,306.53

E.7 Existing assets and Licenses

Our experiments use third-party datasets, models, and evaluation code only through their published releases, with attribution to the original creators in the corresponding citations and asset descriptions.

The datasets include GSM8K (MIT), the MATH benchmark/MATH-500 subset (MIT for the underlying MATH release), AlpacaFarm/AlpacaEval preference and evaluation assets (AlpacaFarm data released for research/non-commercial use under CC BY-NC 4.0; AlpacaEval code under Apache-2.0), OpenMathInstruct-2/OpenMath2 assets (OpenMathInstruct-2 data under CC BY 4.0 and OpenMath2 models under the Llama 3.1 Community License), and the IMDB Large Movie Review Dataset (distributed as a standard research benchmark; Hugging Face lists the license field as "other", and we do not redistribute the raw reviews).

Model assets follow their published terms: Meta Llama 3/3.1/3.2 models under the applicable Llama Community License and Acceptable Use Policy, TinyLlama, Qwen, and lvwerra/distilbert-imdb under Apache-2.0, and GPT-2 under OpenAI's modified MIT license.

For the synthetic star-graph experiments, we credit the generation protocol and baselines to the cited prior work and do not redistribute third-party code or data unless an explicit license or permission is available. We respect the applicable attribution, license-notice, non-commercial/research-use, and acceptable-use requirements for all third-party assets.