ALIGNING VISION-LANGUAGE MODELS FOR GOAL PLANNING IN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration remains a major challenge for reinforcement learning (RL) agents in Craftax-style environments due to sparse rewards and long horizons. LLM-guided RL is a promising direction, but existing methods still suffer from substantial computational and financial costs as well as training instability due to their reliance on frequent queries to large-scale LLMs or carefully designed task-specific prompts. In this paper, we focus on developing an alternative approach that offers strong performance, stability, and cost-effectiveness. Specifically, we construct CCGP-2M, a multimodal goal-preference dataset for the Craftax-Classic environment, and propose a VLM-guided RL framework that fine-tunes a lightweight VLM to guide the RL agent's exploration. This framework consists of an offline stage, in which a Kullback-Leibler Token-level Direct Preference Optimization (KLTDPO) method is introduced to align the VLM with the goal preferences in CCGP-2M, and an online stage, in which the fine-tuned VLM guides the RL agent's exploration. Experiments on Craftax-Classic show that the proposed approach outperforms state-of-the-art baselines in terms of both score and reward, while achieving the highest success rates on 15 out of 22 achievements and the second-highest success rates on another five.

1 INTRODUCTION

Reinforcement learning (RL) has achieved remarkable success in a wide range of sequential decision-making tasks (Mnih et al., 2015; Shakya et al., 2023; Oh et al., 2025). However, these successes have largely been achieved in closed environments characterized by predefined tasks, explicit goals, and immediate feedback, whereas efficient exploration remains challenging in open-world environments such as Minecraft (Fan et al., 2022; Li et al., 2025) and Craftax-Classic (Matthews et al., 2024; Simmons-Edler et al., 2025). In these environments, sparse rewards and long horizons make trial-and-error exploration inefficient. Recently, LLM-guided RL has emerged as a promising method for addressing this challenge, driven by the remarkable capabilities of LLMs in complex decision-making and reasoning tasks (Zhou et al., 2024; Schoepp et al., 2025).

Existing LLM-guided RL methods mainly follow two categories. The first category, illustrated in Figure 1(a), directly leverages LLMs to generate goals that guide RL exploration (Du et al., 2023; Zhang & Lu, 2024; Fu et al., 2026). However, these methods typically require frequent interactions with large-scale LLMs, such as models with more than 70B parameters or proprietary commercial models, resulting in substantial latency, financial costs, and potential instability, thereby limiting their practical applicability. Another line of work uses LLMs to extract reusable structured task knowledge or generate executable planning code, thereby providing task priors for lightweight planners that guide RL agents (Chen et al., 2025; Fan et al., 2025; Qi et al., 2025), as shown in Figure 1(b). Nevertheless, these methods often depend heavily on task-specific prompt engineering and structured knowledge construction. Moreover, a common limitation of these methods is the lack of explicit task-specific fine-tuning for LLMs, which may result in seemingly plausible yet impractical guidance goals and lead to unstable exploration behavior.

These limitations motivate us to develop an alternative that enables efficient exploration while reducing computational cost and maintaining training stability. Specifically, as illustrated in Figure 1(c), we construct CCGP-2M, a multimodal goal preference dataset derived from environment interaction

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.

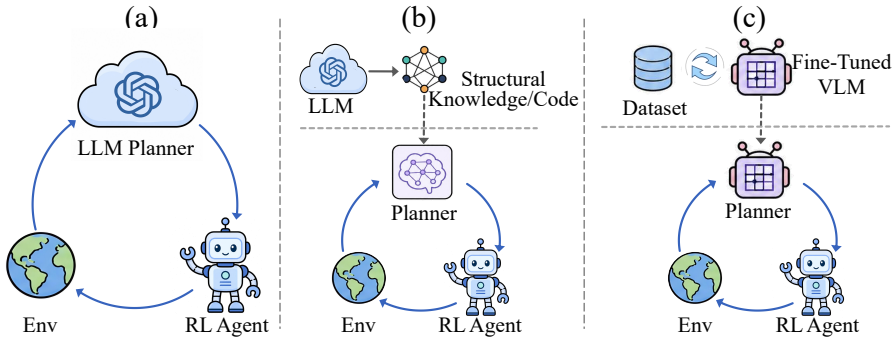

Figure 1: Comparison of existing methods and our framework. Prior work either (a) directly uses cloud-based LLM planners for RL guidance or (b) enhances local planners with LLM-derived structured knowledge or code. (c) Our framework fine-tunes a local VLM planner with goal-preference data to provide RL guidance.

trajectories, and use it to fine-tune a lightweight VLM that generates goals to guide RL exploration. Our main contributions are summarized as follows:

- We construct a large-scale multimodal dataset, called the Craftax-Classic Goal Preference dataset (CCGP-2M), derived from goal-conditioned interaction trajectories without manual annotation. CCGP-2M contains 2 million multimodal preference samples and covers all 22 achievements, a broad range of environment states, and diverse task goals.
- We propose a two-stage VLM-guided RL framework consisting of offline VLM fine-tuning and online goal-guided RL. In the offline stage, we introduce KLTDPO, which integrates token-level sequential KL regularization with distributionally robust reweighting to fine-tune a lightweight VLM on CCGP-2M dataset. In the online stage, the fine-tuned VLM serves as a planner and generates high-level goals to guide RL exploration.
- Extensive experiments on Craftax-Classic show that our approach achieves the highest success rates on 15 out of 22 achievements and the second-highest success rates on another five compared with state-of-the-art baselines. In addition, our method demonstrates substantial advantages in terms of both score and reward.

2 Preliminaries

2.1 Goal-Conditioned POMDP

A goal-conditioned partially observable Markov decision process (GC-POMDP) is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{G}, \mathcal{A}, \Omega, P, \mathcal{O}, R, \gamma)$ (Zhao & Grover, 2023), where $\mathcal{S}$, $\mathcal{G}$, $\mathcal{A}$, and Ω denote the state, goal, action, and observation spaces, respectively. The transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ governs the environment dynamics, while the observation kernel $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \Delta(\Omega)$ generates the agent's partial observations.

At each time step t, the agent's observation is $o_t \in \Omega$, and the goal is $g \in \mathcal{G}$. The agent takes an action $a_t \in \mathcal{A}$ according to the policy $\pi_\theta(\cdot \mid o_t, g)$, causing the environment to transition to a new state s_{t+1} according to $P(s_{t+1} \mid s_t, a_t)$. The agent aims to maximize the expected sum of discounted rewards under goal g: $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)\right]$.

2.2 Direct Preference Optimization

Direct Preference Optimization (DPO) (Rafailov et al., 2023) aligns a policy by comparing pairwise preferences without explicitly learning a reward model. Specifically, starting from the following RL objective:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(x,y) - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x))]], \tag{1}$$

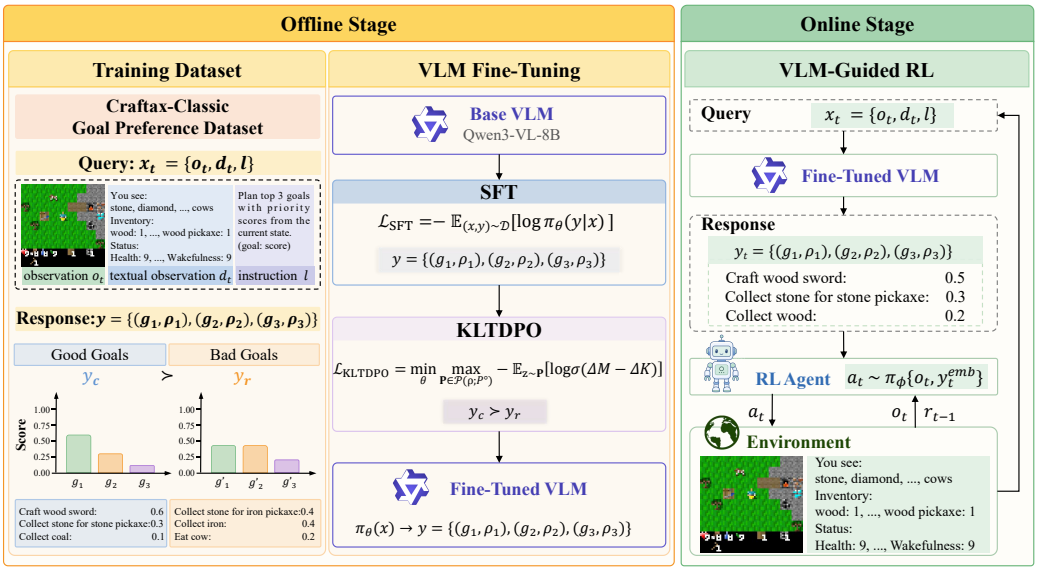

Figure 2: Overview of our framework.

a Bradley-Terry (BT) (Bradley & Terry, 1952) preference predictor is employed, with the likelihood of a preference pair expressed as:

$$P_{\text{BT}}\left(y_c \succ y_r \mid x\right) = \frac{\exp\left(r^*\left(x, y_c\right)\right)}{\exp\left(r^*\left(x, y_c\right)\right) + \exp\left(r^*\left(x, y_r\right)\right)},$$

where $y_c \succ y_r \mid x$ denotes that y_c (chosen) is preferred over y_r (rejected) given x, β is the coefficient of the reverse KL divergence penalty, and $r^*(x, y_c)$ and $r^*(x, y_r)$ stand for the reward function at the response level from the preferred and dispreferred answers, respectively.

DPO establishes a direct functional mapping between the reward model and the optimal policy: $r(x, y) = \beta \log \frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$, where $\mathcal{D}$ is the preference dataset, $\pi_{\text{ref}}(\cdot \mid x)$ is the policy of a fixed reference model (typically selected to be the model that has undergone post-SFT), $\pi_\theta(\cdot \mid x)$ represents the policy of the trained model, initialized with $\pi_\theta = \pi_{\text{ref}}$, and $Z(x)$ is the partition function. Subsequently, Equation (1) can be reformulated as the DPO loss:

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(x,y_c,y_r)\sim\mathcal{D}}\left[\log\sigma\left(\beta \log \frac{\pi_\theta\left(y_c \mid x\right)}{\pi_{\text{ref}}\left(y_c \mid x\right)} - \beta \log \frac{\pi_\theta\left(y_r \mid x\right)}{\pi_{\text{ref}}\left(y_r \mid x\right)}\right)\right]. \quad (2)$$

Minimizing this objective increases the relative likelihood of chosen responses while maintaining proximity to the reference policy. Building upon DPO, TDPO (Zeng et al., 2024) introduces token-level KL regularization, while KLDPO (Xu et al., 2025b) improves preference optimization through robust loss weighting. Details of these variants are provided in Appendix A.

3 Method

In this section, we develop a novel VLM-guided RL approach. Specifically, we first construct CCGP-2M, a multimodal goal preference dataset containing 2 million multimodal preference samples and covering all 22 achievements, a broad range of environment states, and diverse task goals. Second, as illustrated in Figure 2, we propose a two-stage VLM-guided RL framework that combines offline VLM fine-tuning and online goal-guided RL.

3.1 Dataset Construction

In this subsection, we construct CCGP-2M, a multimodal goal preference dataset derived from goal-conditioned interaction trajectories without manual goal annotations. Its construction faces three

challenges: (1) difficult achievements require long prerequisite behaviors and are hard to collect; (2) raw trajectories contain redundant observations and repeated goals; and (3) preference learning requires structurally valid yet contextually inappropriate rejected responses. To address these challenges, we collect diverse trajectories under random and partial initialization, apply quality-aware sample selection to retain informative samples, and construct preference pairs by contrasting trajectory-consistent plans with carefully designed rejected alternatives.

3.1.1 TRAJECTORY COLLECTION.

Following the implementation of SGRL (Qi et al., 2025), we collect goal-conditioned interaction trajectories in the Craftax-Classic environment, where the exploration goals for the RL agent are specified by planning codes generated by DeepSeek-R1 (Guo et al., 2025) and Claude-Opus-4.7 (Anthropic, 2026). Each collected trajectory is represented as $\tau_i = \{(o_t, d_t, y_t, r_t, score_t)\}_{t=1}^{T_i}$, where o_t is the observation, d_t is its textual description, y_t is the goal set, r_t is the reward, and $score_t$ is the score at time step t. Here, T_i denotes the length of the i-th trajectory. Notably, we employ two environment initialization strategies to increase trajectory diversity and ensure sufficient coverage of states that enable the completion of deep achievements. Specifically, we use random initialization together with a manually specified setting in which the agent starts with an iron pickaxe to facilitate diamond-related exploration. Under these two settings, we collect 60M and 30M environment steps, respectively, and retain approximately 27,000 and 360 valid trajectories, yielding a dataset containing over 8.9 million observations. Statistics of the collected goal-conditioned interaction trajectories are summarized in Table 1. More details are provided in Appendix B.1.

Table 1: Statistics of the goal-conditioned interaction trajectories collected under different initialization settings, retaining only episodes that achieve the iron-pickaxe achievement as valid trajectories for dataset construction.

Statistic	**Random**	**Manual**	**Total**
Environment steps	60M	30M	90M
Trajectories	27,355	360	27,715
Observations	8,858,665	77,816	8,936,481
Observations with diamond	12,675	41,608	54,283

3.1.2 QUALITY-AWARE SAMPLE SELECTION

Raw trajectories contain redundant and low-quality records, such as repeated observations, malformed planner outputs, and persistent local goals, which may bias planner fine-tuning toward frequent but less informative behaviors.

Invalid and duplicated records are first removed according to observation validity, goal format consistency, priority constraints, and normalized observation-goal uniqueness. For each remaining candidate at step t, we compute a quality score

$$S_t^{\text{quality}} = B_t \min(R_t, R_{\max}) P_t^{\text{goal}} + \epsilon_t, \tag{3}$$

where B_t captures informative trajectory events, including goal transitions, achievement proximity, state changes, and survival needs; R_t is a frequency-based rarity multiplier; $R_{\max}$ controls the maximum contribution of goal rarity; P_t^{goal} penalizes recently repeated goals; and ϵ_t breaks ties between candidates with identical deterministic scores.

To avoid overrepresenting long trajectories while preserving the empirical state distribution, each episode is assigned a sampling quota

$$M_i = \left\lfloor \mathcal{T} \left(\lambda \frac{1}{N_{\text{ep}}} + (1-\lambda) \frac{N_i}{\sum_j N_j} \right) \right\rfloor, \tag{4}$$

where M_i denotes the sampling quota assigned to episode i, $\mathcal{T}$ represents the total sampling budget, and N_{ep} is the total number of episodes. N_i denotes the number of valid candidates extracted from episode i. The coefficient $\lambda \in [0, 1]$ controls the trade-off between uniform episode-level coverage and candidate-proportional allocation. Then, all valid candidates within episode i are ranked according to S_t^{quality} and the top $\min(M_i, N_i)$ candidates are retained. We set $R_{\max} = 4.0$ and sample ϵ_t from $\mathcal{U}(0, 0.15)$ only for tie breaking. The exact signal weights and decision rules are reported in Appendix B.2.

3.1.3 PREFERENCE PAIR CONSTRUCTION

For each retained query x at trajectory step t, we use the recorded planner response y_t as the chosen response y_c. A rejected response y_r is then constructed to preserve the required output format while being less suitable for the current state or task stage. Specifically, y_r is sampled from a mixture of context-dependent negative construction strategies: $y_r \sim q_{\text{neg}}(\cdot \mid \tau, t, y_c)$, where $q_{\text{neg}} = \lambda_f q_{\text{future}} + \lambda_p q_{\text{past}} + \lambda_m q_{\text{mixed}} + \lambda_s q_{\text{priority}} + \lambda_g q_{\text{global}}$, the nonnegative weights satisfy $\lambda_f + \lambda_p + \lambda_m + \lambda_s + \lambda_g = 1$. Here, q_{future} and q_{past} sample responses from nearby future and past steps to create premature or obsolete plans. q_{mixed} replaces part of y_c with subgoals from neighboring steps, q_{priority} changes goal ordering or priority scores while preserving the goal set, and q_{global} samples a different response from the same trajectory. These strategies produce hard negatives that remain fluent and structurally valid but differ from y_c in temporal relevance, goal composition, or priority assignment. Then, the resulting preference dataset is named CCGP-2M, which is denoted by $\mathcal{D}_{\text{pref}} = \left\{(x^i, y_c^i, y_r^i)\right\}_{i=1}^{N}$, where $y_c^i \succ y_r^i$. The CCGP-2M dataset contains 2 million preference samples covering 43 distinct goals. The distribution of goal types in CCGP-2M is illustrated in Figure 3.

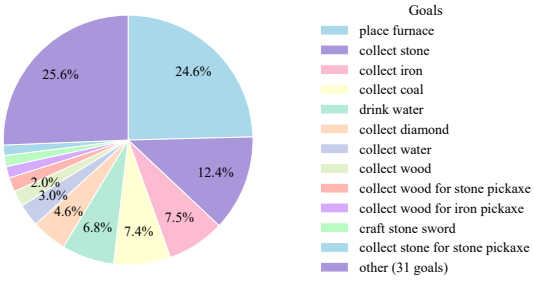

Figure 3: Distribution of the goals in CCGP-2M preference dataset.

In practice, for SFT, we ignore the rejected responses and use only the preferred pairs: $(x, y_c) \sim \mathcal{D}_{\text{pref}}^{+}$, where $\mathcal{D}_{\text{pref}}^{+}$ denotes the positive subset extracted from $\mathcal{D}_{\text{pref}}$. For preference optimization, the complete preference samples in $\mathcal{D}_{\text{pref}}$ are used to optimize the preference objective. More details are provided in Appendix B.3.

3.2 VLM-GUIDED RL FRAMEWORK

The VLM-guided RL framework consists of two stages: offline VLM fine-tuning and online goal-guided RL. In the offline stage, we introduce KLTDPO, which integrates token-level sequential KL regularization with distributionally robust reweighting to fine-tune a lightweight VLM on CCGP-2M dataset. In the online stage, the fine-tuned VLM serves as a planner and generates high-level goals to guide RL exploration.

3.2.1 VLM FINE-TUNING

In this subsection, we focus on how to further fine-tune the VLM model that has undergone SFT with the loss $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y_c)\sim\mathcal{D}_{\text{pref}}^{+}}\left[\sum_{t=1}^{|y_c|} \log \pi_\theta\left(y_c^t \mid x, y_c^{<t}\right)\right]$, using the CCGP-2M preference dataset.

Specifically, we first use DPO and KLDPO to fine-tune the VLM and find that both methods rapidly distinguish chosen responses from rejected ones, as evidenced by their increasing reward margins and reward accuracies, which quickly approach 1 in Figure 4. However, when the fine-tuned VLMs are used as planners to guide RL exploration, the performance gains of the RL agent remain limited or even negative, as shown in Table 2. We hypothesize that this may be attributed to the distinctive data distribution of CCGP-2M. Specifically, the chosen response is a trajectory-consistent plan, whereas the rejected responses are diverse but often constitute easy negatives generated by modifying the goals, their ordering, or their priority scores.

Motivated by TDPO (Zeng et al., 2024), we propose KLTDPO, which combines token-level sequential KL regularization with the distributionally robust reweighting mechanism of KLDPO (Xu et al., 2025b) to emphasize difficult preference samples and enable finer-grained preference optimization for planning decisions. The detailed training procedure is as follows.

Training procedure. We first formulate autoregressive response generation as an augmented preference-based MDP, where the state at time step t includes the prompt x and the first $t-1$

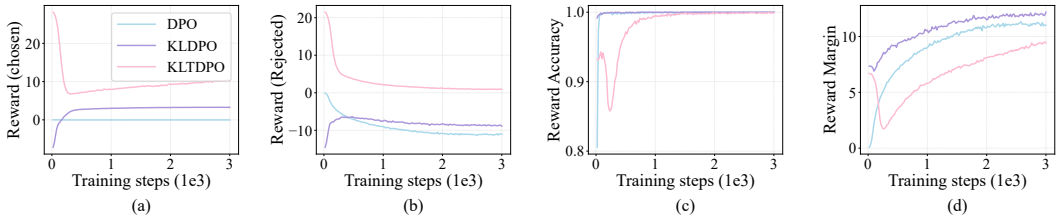

Figure 4: The reward curves for VLMs fine-tuned with DPO, KLDPO, and KLTDPO.

tokens $y^{<t}$ of the response, denoted as $s_t = [x, y^{<t}]$, the action corresponds to the next generated token, denoted as $a_t = y^t$, and the reward function $r(x, y)$ of the entire prompt-response can be decomposed into token-wise rewards, i.e., $r(x, y) = \sum_{t=1}^{T} \gamma^{t-1} R\left([x, y^{<t}], y^t\right)$. Then, the state-action value function and state value function can be defined as:

$$Q_\pi\left([x, y^{<t}], y^t\right) = \mathbb{E}_\pi\left[\sum_{m=0}^{T-t} \gamma^m R_{t+m} \middle| [x, y^{<t}], y^t\right],$$
$$V_\pi\left([x, y^{<t}]\right) = \mathbb{E}_{z\sim\pi(\cdot|[x,y^{<t}])}\left[Q_\pi([x, y^{<t}], z)\right].$$

The advantage function can be expressed as follows:

$$A_\pi\left([x, y^{<t}], y^t\right) = Q_\pi\left([x, y^{<t}], y^t\right) - V_\pi\left([x, y^{<t}]\right). \tag{5}$$

Since token generation is formulated as an undiscounted finite-horizon process, we set $\gamma = 1$. Further, let $\mathcal{P}(\rho; \mathbf{P}^o)$ be a distributional uncertainty set centered around $\mathbf{P}^o$ with radius $\rho > 0$, defined as:

$$\mathcal{P}\big(\rho; \mathbf{P}^\circ\big) := \left\{\mathbb{P} \in \mathcal{P}(\mathcal{Z}) : D\big(\mathbf{P}, \mathbf{P}^\circ\big) \le \rho\right\}, \tag{6}$$

where $D(\cdot, \cdot)$ denotes a distance metric between two probability measures, e.g., D_{KL}. Following the principles of distributionally robust optimization (Xu et al., 2025b), we formulate a distributionally robust token-level objective function:

$$\min_\theta \max_{\mathbf{P}\in\mathcal{P}(\rho;\mathbf{P}^o)} \mathbb{E}_{z\sim\mathbf{P}}\left[A_{\pi_{\mathrm{ref}}}\left([x, y^{<t}], z\right) - \beta D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid [x, y^{<t}]) \| \pi_{\mathrm{ref}}(\cdot \mid [x, y^{<t}])\right)\right]. \tag{7}$$

Then, based on Lemma 4.4 in Zeng et al. (2024), we can establish the equivalence between the BT model and the Regret Preference Model as shown in the following lemma.

Lemma 3.1. *Given a prompt x and pairwise responses (y_c, y_r), the Bradley-Terry model expresses the preference probability in terms of the goal-conditioned optimal policy π_θ^* and reference policy π_{ref}:*

$$P_{\mathrm{BT}}^*\left(y_c \succ y_r \mid x\right) = \sigma\left(\Delta M - \Delta K\right) \tag{8}$$

where ΔM represents the difference in implicit rewards defined by the optimal policy π_θ^* and the reference policy π_{ref}, weighted by β, represented as:

$$\Delta M = \beta \log \frac{\pi_\theta\left(y_c \mid x\right)}{\pi_{\mathrm{ref}}\left(y_c \mid x\right)} - \beta \log \frac{\pi_\theta\left(y_r \mid x\right)}{\pi_{\mathrm{ref}}\left(y_r \mid x\right)},$$

and ΔK represents the difference in sequential KL divergence between two pairs (x, y_c) and (x, y_r), expressed as:

$$\Delta K = \beta D_{\mathrm{SeqKL}}\left(x, y_r; \pi_{\mathrm{ref}} \| \pi_\theta\right) - \beta D_{\mathrm{SeqKL}}\left(x, y_c; \pi_{\mathrm{ref}} \| \pi_\theta\right).$$

Drawing on Lemma 3.1, we reformulate the BT model into a structure solely relevant to the policy, which enables us to formulate a likelihood maximization objective for a parametrized policy π_θ. The loss function is given by:

$$\min_\theta \max_{\mathbf{P}\in\mathcal{P}(\rho;\mathbf{P}^o)} -\mathbb{E}_{z\sim\mathbf{P}}\left[\log \sigma\left(\Delta M - \Delta K\right)\right]. \tag{9}$$

3.2.2 VLM-GUIDED RL

In the online stage, we use the fine-tuned VLM as a goal planner to guide the RL agent. At each time step t, the VLM planner receives the query $x_t = (o_t, d_t, l)$, where o_t is the current visual observation, d_t is the textual description of the observation, and l is the task instruction. Then, the VLM planner generates a prioritized goal set $y_t = \{(g_t^k, \rho_t^k)\}_{k=1}^K$, where g_t^k is the k-th candidate goal, and ρ_t^k is its priority score. The RL agent selects the action according to $a_t \sim \pi_\phi(\cdot \mid o_t, y_t^{\text{emb}})$ to maximize the expected sum of discounted rewards, where y_t^{emb} is the text embedding of the prioritized goal set.

The policy is optimized using PPO (Schulman et al., 2017) with goal-conditioned state augmentation. Here, the probability ratio is defined as $\eta_t(\phi) = \frac{\pi_\phi(a_t|o_t, y_t^{\text{emb}})}{\pi_{\phi_{\text{old}}}(a_t|o_t, y_t^{\text{emb}})}$, where $\pi_{\phi_{\text{old}}}$ is the policy before the update. The clipped PPO surrogate objective is

$$\mathcal{L}_{\text{PPO}}(\phi) = \mathbb{E}_t\left[\min\left(\eta_t(\phi)\hat{A}_t, \text{clip}(\eta_t(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{10}$$

where $\hat{A}_t$ is the estimated advantage function and ϵ is the clipping coefficient.

4 EXPERIMENTS

We conduct experiments on the open-world environment Craftax-Classic (Matthews et al., 2024). Our experiments aim to assess whether the fine-tuned VLM can generate effective goals that guide the RL agent toward efficient exploration and improved task performance. Our data and code can be found at `https://anonymous.4open.science/r/vlm4rl-iclr2027-C011`.

4.1 EXPERIMENTAL SETUP

Environment. Craftax-Classic is a JAX-based reimplementation of Crafter (Hafner, 2022) with open-world survival and crafting mechanics (Matthews et al., 2024). It includes 22 achievements, $63 \times 63 \times 3$ visual observations, and 17 discrete actions, providing a challenging benchmark for goal-guided exploration and long-horizon planning. Further details about the environment can be found in Appendix C.1.1.

Baselines. We compare against three groups of baselines. PPO (Schulman et al., 2017) represents conventional RL without language-model guidance. ELLM (Du et al., 2023) and AdaRefiner (Zhang & Lu, 2024) rely on cloud-based LLMs to provide guidance for RL, whereas Causal-aware LLMs (Chen et al., 2025) and SGRL (Qi et al., 2025) employ locally deployed planners for goal generation and policy guidance. For comparison, we also report results using the Base VLM and VLMs fine-tuned with SFT, DPO (Rafailov et al., 2023), TDPO (Zeng et al., 2024), and KLDPO (Xu et al., 2025b) as planners.

Training Configuration. We use Qwen3-VL-8B-Instruct[1] as the base model. All model fine-tuning methods are conducted on a cluster of 4 NVIDIA H100 GPUs, each with 80 GB of memory, using nearly identical configurations. Detailed PPO hyperparameter settings, VLM configurations, and model training details for all methods are provided in Appendix C.1.2.

Evaluation Metrics. We evaluate RL agent performance using three metrics: cumulative reward, overall score and achievement success rate. The cumulative reward measures achievement completion and survival feedback, with the agent receives $+1$ for each newly unlocked achievement and $+0.1$ or -0.1 for recovering or losing health points. The overall score aggregates achievement difficulties using the log-space average: $S \doteq \exp\left(\frac{1}{N}\sum_{i=1}^N \ln(1+s_i)\right) - 1$, where $s_i \in [0, 100]$ is the success rate of the i-th achievement and $N = 22$. The success rate of each achievement is computed as the proportion of episodes in which the achievement is unlocked.

[1] Available at `https://huggingface.co/Qwen/Qwen3-VL-8B-Instruct`

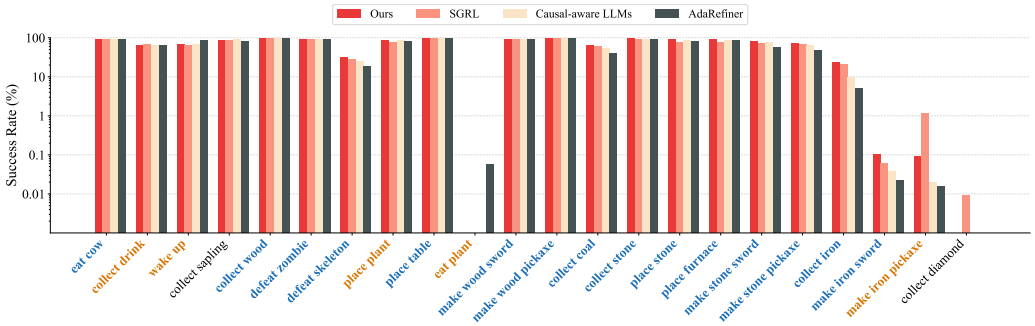

Figure 5: The success rates across the 22 Craftax-Classic achievements. Blue and orange achievement names denote the tasks where our method ranks first and second, respectively.

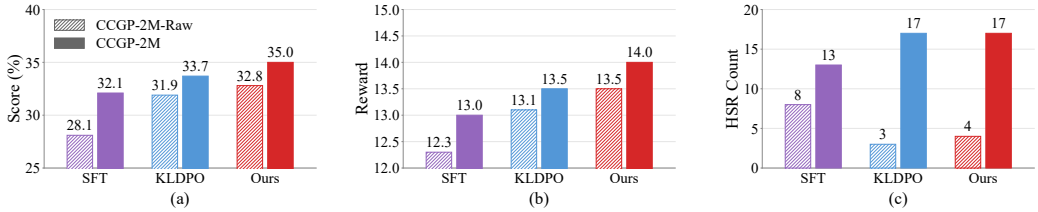

Figure 6: The scores, rewards, and HSR counts of RL agents guided by different VLMs on the CCGP-2M-Raw and CCGP-2M datasets.

4.2 Main Results

Table 2 and Figure 5 summarize the performance of our method, strong baselines, and other VLM fine-tuning methods on the Craftax-Classic environment. As shown in Table 2, RL agents guided by fine-tuned VLMs generally achieve rewards that are higher than or comparable to those of the strongest baseline. Furthermore, our KLTDPO-based method achieves the highest score and reward among all evaluated methods. The superior performance of KLTDPO supports the effectiveness of its token-level, distributionally robust preference optimization design. Figure 5 further presents the success rates across the 22 achievements, with our method achieves the highest success rate on 15 achievements and the second-highest on 5, demonstrating broad advantages across achievements of varying difficulty. More detailed experimental results are provided in Appendix C.2.

Table 2: Performance on Craftax-Classic (mean $\pm$ std).

Method	**Score (%)**	**Reward**
PPO (Schulman et al., 2017)	24.8 ± 5.7	11.9 ± 1.1
ELLM (Du et al., 2023)	28.4 ± 2.5	12.2 ± 1.0
AdaRefiner (Zhang & Lu, 2024)	28.5 ± 2.3	12.3 ± 0.9
Causal-aware LLMs (Chen et al., 2025)	31.8 ± 0.5	12.9 ± 0.4
SGRL (Qi et al., 2025)	33.8 ± 1.5	13.0 ± 0.3
Base VLM	26.7 ± 3.3	11.8 ± 0.5
SFT	32.1 ± 0.7	13.0 ± 0.2
DPO	31.9 ± 1.1	13.1 ± 0.3
TDPO	32.6 ± 1.0	13.4 ± 0.2
KLDPO	33.7 ± 1.0	13.5 ± 0.3
KLTDPO (**Ours**)	$\mathbf{35.0 \pm 0.6}$	$\mathbf{14.0 \pm 0.2}$

4.3 Data Quality Analysis

Figure 6 presents the scores, rewards, and highest success rate (HSR) counts of RL agents guided by VLMs fine-tuned with SFT, KLDPO, and KLTDPO on the CCGP-2M-Raw[2] and CCGP-2M

[2]CCGP-2M-Raw is a preference dataset constructed by randomly selecting two million samples from the complete preference dataset.

datasets, respectively. Across all three training methods, using CCGP-2M leads to higher scores and rewards as well as lower standard deviations than using CCGP-2M-Raw. These results indicate that removing redundant and low-quality records provides more reliable supervision for VLM fine-tuning and improves downstream RL performance. More details are provided in Appendix C.3.

5 Related Work

5.1 LM-Guided RL

In recent years, large language models (LLMs) have been leveraged to guide RL agents in addressing the challenges of sparse rewards and long-horizon challenges in open-world environments (Zhou et al., 2024; Schoepp et al., 2025). One line of research leverages cloud-based LLMs as external planners to guide RL agents by generating auxiliary rewards or goals (Du et al., 2023), refining decisions from agent feedback (Zhang & Lu, 2024), or producing subgoal sequences and planning programs (Shukla et al., 2024; Gu et al., 2026; Qi et al., 2025). However, their reliance on large-scale external models limits local deployment and frequent interaction. Another line of research explores locally deployable planners using task-specific prompts (Ruiz-Gonzalez et al., 2024), structured planning knowledge such as causal relations and subgoal graphs (Yuan et al., 2023; Chen et al., 2025; Fan et al., 2025), or self-correction mechanisms (Chi et al., 2025; Dou et al., 2026). Nevertheless, these approaches mainly enhance general planning capabilities without explicitly learning environment-specific goal preferences. Recent VLM-based RL methods further incorporate visual-semantic understanding through reward modeling (Jiang et al., 2024), trajectory feedback (Luu et al., 2025), or representation learning (Xu et al., 2025a). Unlike these approaches, we learn environment-specific goal preferences from interaction trajectories and align a local VLM planner through preference optimization for adaptive RL guidance.

5.2 Preference Optimization

Beyond SFT, preference-based methods align models by learning from comparisons between preferred and rejected responses. RLHF first learns a reward model from preference data and then optimizes the policy using RL (Christiano et al., 2017; Ouyang et al., 2022). DPO simplifies this process by directly optimizing the policy from preference samples without training an explicit reward model (Rafailov et al., 2023). Subsequent methods improve DPO from different perspectives. KLDPO formulates preference optimization under distributional uncertainty and assigns larger weights to preference samples with higher losses, improving robustness to preference shifts (Xu et al., 2025b). TDPO introduces token-level sequential KL regularization to control policy deviations at each generation step (Zeng et al., 2024). However, KLDPO still evaluates each response as a whole, while TDPO does not account for differences in preference-pair difficulty. We therefore propose KLTDPO, which combines token-level sequential KL regularization with distributionally robust loss reweighting. This design is tailored to CCGP-2M, where chosen and rejected plans may share most of their valid content and differ only in a goal, ordering decision, or priority score.

6 Conclusion

In this paper, we develop a novel VLM-guided RL approach that enables efficient exploration while reducing computational cost and maintaining training stability. Specifically, we first construct CCGP-2M, a multimodal goal preference dataset containing 2 million multimodal preference samples and covering all 22 achievements, a broad range of environment states, and diverse task goals. Then, a two-stage VLM-guided RL framework is introduced that fine-tunes a lightweight VLM as a planner to provide goal guidance for exploration. Experiments on Craftax-Classic show that the proposed approach achieves superior performance in terms of score and reward compared with the state-of-the-art baselines and improves success rates on most tasks. However, due to the substantial time and computational costs associated with data collection, model fine-tuning, and RL training, our current experiments are limited to Craftax-Classic. Future work will explore more cost-efficient solutions and evaluate the proposed framework across a broader range of environments.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript translation and to modify code for dataset filtering. We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate research hypotheses or mathematical claims, derive proofs, design research methods or experiments, or interpret experimental results. Additionally, we used generative AI tools for grammar checking and the generation and revision of plotting code. We reviewed all AI-assisted work: the authors manually reviewed the translated and edited text, inspected the AI-assisted code changes, and ran and tested the modified code. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops a VLM-guided RL method and evaluates it in the publicly available Craftax-Classic environment. The study does not involve human participants, personal data, or sensitive information. We have not identified direct ethical concerns arising from these experiments.

REPRODUCIBILITY STATEMENT

The appendix describes the trajectory collection procedure, dataset construction, training configurations, and evaluation protocol. Source code and scripts are provided in the supplementary material to support reproduction of the reported results.

REFERENCES

Anthropic. Claude opus 4.7. Anthropic Blog Post, 2026. URL `https://www.anthropic.com/news/claude-opus-4-7`.

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.

Wei Chen, Jiahao Zhang, Haipeng Zhu, Boyan Xu, Zhifeng Hao, Keli Zhang, Junjian Ye, and Ruichu Cai. Causal-aware large language models: enhancing decision-making through learning, adapting and acting. In *IJCAI*, 2025.

Haotian Chi, Songwei Zhao, Ivor Tsang, Yew-Soon Ong, Hechang Chen, Yi Chang, and Haiyan Yin. LLMV-age: verifying LLM-guided planning for agentic exploration in open-world RL. In *ICLR 2025 Workshop: VerifAI: AI Verification in the Wild*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

Zhihao Dou, Dongfei Cui, Jun Yan, Weida Wang, Benteng Chen, Haoming Wang, Zeke Xie, and Shufei Zhang. Dsadf: Thinking fast and slow for decision making. *International Journal of Computer Vision*, 134(6):270, 2026.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *ICML*, 2023.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *NeurIPS*, 2022.

Shanwei Fan, Bin Zhang, Zhiwei Xu, Yingxuan Teng, Siqi Dai, Lin Cheng, and Guoliang Fan. Subgoal graph-augmented planning for LLM-guided open-world reinforcement learning. *arXiv preprint arXiv:2511.20993*, 2025.

Jiarun Fu, Lizhong Ding, Qiuning Wei, Yuhan Guo, Yurong Cheng, and Junyu Zhang. Counterfactual planning for generalizable agents' actions. In *AAAI*, 2026.

Chengyang Gu, Yuxin Pan, Hui Xiong, and Yize Chen. STO-RL: Offline RL under sparse rewards via LLM-guided subgoal temporal order. *arXiv preprint arXiv:2601.08107*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *ICLR*, 2022.

Haobin Jiang, Junpeng Yue, Hao Luo, Ziluo Ding, and Zongqing Lu. Reinforcement learning friendly vision-language model for minecraft. In *European Conference on Computer Vision*, 2024.

Jiajian Li, Qi Wang, Yunbo Wang, Xin Jin, Yang Li, Wenjun Zeng, and Xiaokang Yang. Open-world reinforcement learning over long short-term imagination. In *ICLR*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Tung Minh Luu, Younghwan Lee, Donghoon Lee, Sunho Kim, Min Jun Kim, and Chang D Yoo. Enhancing rating-based reinforcement learning to effectively leverage feedback from large vision-language models. In *ICML*, 2025.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *ICML*, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Junhyuk Oh, Gregory Farquhar, Iurii Kemaev, Dan A Calian, Matteo Hessel, Luisa Zintgraf, Satinder Singh, Hado Van Hasselt, and David Silver. Discovering state-of-the-art reinforcement learning algorithms. *Nature*, 648(8093):312–319, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Yajie Qi, Wei Wei, Lin Li, Lijun Zhang, Zhidong Gao, Da Wang, and Huizhong Song. Goal-guided efficient exploration via large language model in reinforcement learning. *arXiv preprint arXiv:2509.22008*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP-IJCNLP*, 2019.

Unai Ruiz-Gonzalez, Alain Andres, Pedro G Bascoy, and Javier Del Ser. Words as beacons: guiding RL agents with high-level language prompts. *arXiv preprint arXiv:2410.08632*, 2024.

Sheila Schoepp, Masoud Jafaripour, Yingyue Cao, Tianpei Yang, Fatemeh Abdollahi, Shadan Golestan, Zahin Sufiyan, Osmar R Zaiane, and Matthew E Taylor. The evolving landscape of llm- and vlm-integrated reinforcement learning. *arXiv preprint arXiv:2502.15214*, 2025.

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

Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, 2023.

Yash Shukla, Wenchang Gao, Vasanth Sarathy, Alvaro Velasquez, Robert Wright, and Jivko Sinapov. LgTS: Dynamic task sampling using LLM-generated sub-goals for reinforcement learning agents. In *AAMAS*, 2024.

Riley Simmons-Edler, Ryan Badman, Felix Berg, Raymond Chua, John Vastola, Joshua Lunger, William Qian, and Kanaka Rajan. Deep RL needs deep behavior analysis: Exploring implicit planning by model-free agents in open-ended environments. In *NeurIPS*, 2025.

Haoran Xu, Peixi Peng, Guang Tan, Yiqian Chang, Luntong Li, and Yonghong Tian. Vlms-guided representation distillation for efficient vision-based reinforcement learning. In *CVPR*, 2025a.

Zaiyan Xu, Sushil Vemuri, Kishan Panaganti, Dileep Kalathil, Rahul Jain, and Deepak Ramachandran. Robust LLM alignment via distributionally robust direct preference optimization. In *NeurIPS*, 2025b.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. *NeurIPS*, 2023.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In *ICML*, 2024.

Wanpeng Zhang and Zongqing Lu. Adarefiner: Refining decisions of language models with adaptive feedback. In *NAACL*, 2024.

Siyan Zhao and Aditya Grover. Decision stacks: Flexible reinforcement learning via modular generative models. In *NeurIPS*, 2023.

Jiehan Zhou, Yang Zhao, Jiahong Liu, Peijun Dong, Xiaoyu Luo, Hang Tao, Shi Chang, and Hanjiang Luo. LLM4RL: Enhancing reinforcement learning with large language models. In *CCECE*, 2024.

A Supplementary Material for Preliminaries

A.1 Kullback–Leibler Direct Preference Optimization

Kullback–Leibler Direct Preference Optimization (KLDPO) (Xu et al., 2025b) formulates preference optimization under distribution shift as a distributionally robust optimization problem. Given a preference dataset consisting of samples $z_i = (x_i, y_i^c, y_i^r)$, where x_i denotes the input query, y_i^c and y_i^r denote the chosen and rejected responses, respectively, let P_n^0 denote the empirical distribution over the preference dataset. KLDPO considers a set of possible preference distributions within a KL divergence neighborhood of P_n^0:

$$\mathcal{U}(P_n^0) = \left\{ P : D_{\mathrm{KL}}(P \| P_n^0) \le \rho \right\}, \tag{11}$$

where P denotes a reweighted preference distribution and ρ controls the size of the distributional uncertainty set.

The distributionally robust objective is formulated as

$$\min_{\theta} \max_{P \in \mathcal{U}(P_n^0)} \mathbb{E}_{z_i \sim P} \left[\ell_{\mathrm{DPO}}(z_i; \theta) \right], \tag{12}$$

where θ denotes the parameters of the policy model and ℓ_{DPO} represents the standard DPO loss computed on a preference sample.

A tractable approximation assigns each preference sample an exponentially tilted weight:

$$w_i = \frac{\exp(\ell_i / \tau)}{\sum_{j=1}^{n} \exp(\ell_j / \tau)}, \tag{13}$$

where $\ell_i = \ell_{\mathrm{DPO}}(z_i; \theta)$ is the DPO loss of the i-th preference sample, and $\tau > 0$ controls the intensity of reweighting.

The resulting approximate objective is

$$\mathcal{L}_{\mathrm{KLDPO}} = \sum_{i=1}^{n} w_i \ell_{\mathrm{DPO}}(z_i; \theta). \tag{14}$$

Thus, KLDPO assigns larger weights to preference samples with relatively high DPO losses, emphasizing challenging preference comparisons and improving robustness against preference distribution shifts.

A.2 Token-level Direct Preference Optimization

Token-level Direct Preference Optimization (TDPO) (Zeng et al., 2024) extends DPO by explicitly incorporating token-level divergence between the policy and reference distributions. For a response $y = (y_1, \ldots, y_T)$, where y_t denotes the t-th token and T is the response length, the sequential forward KL divergence is defined as

$$\mathrm{KL}_{\mathrm{seq}}(\pi_\theta \| \pi_{\mathrm{ref}}) = \sum_{t=1}^{T} \mathrm{KL}\left(\pi_\theta(\cdot | x, y_{<t}) \| \pi_{\mathrm{ref}}(\cdot | x, y_{<t}) \right), \tag{15}$$

where x denotes the input prompt, $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the generated prefix before token y_t, π_θ denotes the policy model, and π_{ref} denotes the reference model.

Given a preference example $z = (x, y_c, y_r)$, where y_c and y_r denote the chosen and rejected responses, respectively, TDPO defines the token-level divergence difference as

$$\Delta\mathrm{KL}_{\mathrm{seq}} = \mathrm{KL}_{\mathrm{seq}}(\pi_\theta(\cdot|x, y_c) \| \pi_{\mathrm{ref}}(\cdot|x, y_c)) - \mathrm{KL}_{\mathrm{seq}}(\pi_\theta(\cdot|x, y_r) \| \pi_{\mathrm{ref}}(\cdot|x, y_r)). \tag{16}$$

TDPO incorporates the sequential KL difference into the DPO preference objective as

$$\mathcal{L}_{\mathrm{TDPO}} = -\log \sigma \left(\beta \left(h_\theta(x, y_c, y_r) - \alpha \Delta\mathrm{KL}_{\mathrm{seq}} \right) \right), \tag{17}$$

where $h_\theta(x, y_c, y_r)$ denotes the policy-to-reference log-ratio difference, $\alpha > 0$ controls the contribution of the sequential KL difference, $\beta > 0$ is the inverse temperature parameter controlling the strength of preference optimization, and $\sigma(\cdot)$ denotes the sigmoid function.

The stop-gradient operator is denoted as $sg(\cdot)$ when it is applied to prevent gradient propagation through the reference distribution.

B Supplementary Material for Details of Method

B.1 Trajectory Collection

Planner-guided interaction. We employ SGRL (Qi et al., 2025) to collect goal-conditioned interaction trajectories in Craftax-Classic. SGRL (Qi et al., 2025) is a method that leverages code generated by LLMs as planners to guide RL agents. Following the original SGRL setting, we use planning codes generated by two different LLMs, DeepSeek-R1 (Guo et al., 2025) and Claude-Opus-4.7 (Anthropic, 2026), to guide RL exploration and environment interaction, thereby increasing the diversity of goals and states. At the time step t, the planner receives the current textual observation d_t and produces a prioritized goal set $y_t = \left\{(g_t^k, \rho_t^k)\right\}_{k=1}^{K}$, where g_t^k denotes the k-th goal and $\rho_t^k \in [0, 1]$ denotes its priority.

The goal set is encoded as an additional condition and provided to the RL policy. The policy subsequently interacts with the environment, while the visual observation, textual observation, planner output, and environmental feedback are recorded at every step. Formally, the collected trajectory buffer is

$$\mathcal{D}_\tau = \{\tau_i\}_{i=1}^{N_\tau}, \qquad \tau_i = \{(o_t, d_t, y_t, r_t, \mathrm{score}_t)\}_{t=1}^{T_i}, \tag{18}$$

where r_t is the environmental reward, score_t is the current benchmark score, and T_i is the length of trajectory i.

Initialization settings. Craftax-Classic is a sparse-reward and long-horizon open-world environment, where the difficulty of achieving different goals increases with their achievement depth. Diamond collection corresponds to one of the deepest achievements (depth 8) and remains challenging due to the long sequence of prerequisite behaviors required for obtaining a diamond. Therefore, relying only on the default benchmark initialization would result in insufficient trajectories for late-stage achievements and limit the diversity of collected data. To improve trajectory coverage, we employ two complementary initialization settings. The random initialization follows the original benchmark setting and collects natural gameplay trajectories from diverse initial states. The partial initialization modifies the agent inventory by providing an iron pickaxe at the beginning of an episode, enabling faster access to diamond-related trajectories while preserving the corresponding late-stage environmental context.

Under *random initialization*, the agent starts from the default environment state with an empty inventory. This setting preserves the original task difficulty and generates trajectories covering the complete progression from basic resource collection to advanced crafting and exploration. DeepSeek-R1 and Claude-Opus-4.7 are each evaluated using three random seeds, with 10 million environment steps per seed. The resulting six runs contain 60 million environment steps. We retain trajectories that successfully complete either the iron-pickaxe achievement or the diamond achievement, resulting in 27,355 trajectories and 8,858,665 states.

Under *partial initialization*, the agent is initialized with an iron pickaxe in its inventory. This initialization removes a substantial portion of the early-game prerequisite sequence and allows the policy to directly explore states relevant to diamond collection. We execute an additional 30 million environment steps using the same two planning-code sources. This procedure yields 360 diamond trajectories containing 77,816 states.

The two settings are complementary: random initialization preserves the natural state distribution and long-horizon dependency structure, whereas partial initialization improves the sampling efficiency of rare diamond-related decisions.

Prompts for Trajectory Collection The exact prompts used to construct the planning codes and query the planner are reported below.

Prompt for Planning-Code Generation

You are an experienced Python developer tasked with implementing a class named PriGoal that contains a method called determine_goal. This method should determine the goals an agent should pursue based on its current state (including its view range, inventory, and status)

Prompt for Planning-Code Generation (continued)

and the rules of the game. The game rules are as follows:
tree: {require: {}, receive: {wood: 1}, leaves: grass}
stone: {require: {wood_pickaxe: 1}, receive: {stone: 1}, leaves: path}
coal: {require: {wood_pickaxe: 1}, receive: {coal: 1}, leaves: path}
iron: {require: {stone_pickaxe: 1}, receive: {iron: 1}, leaves: path}
...
wood_sword: {uses: {wood: 1}, nearby: [table], gives: 1}
stone_sword: {uses: {wood: 1, stone: 1}, nearby: [table], gives: 1}
achievements:
- collect_coal - collect_diamond - collect_drink - collect_iron - collect_sapling...
The environment information is generated by the following code:

```
def render_text_describ(self, view_arr, index):
        (map_view, mob_map, inventory_values, status_values) =
            view_arr
        mob_id_to_name = ["zombie", "cows", "skeletons", "arrows"]
        block_id_to_name = ["invalid", "out of bounds", "grass",
            "water", "stone", "tree", "wood", "path", "coal",
            "iron", "diamond", "crafting table", "furnace", "sand",
            "lava", "plant", "ripe plant"]
        ...

        text_obs = "You see: " + text_view + "\n Inventory: " +
            text_obs_inv + "\n Status: " + status
        return text_obs
```

The environment state is represented by text_obs:
Example 1:
You see: plant, zombie, tree, grass, sand, path, stone
Inventory: wood: 1
Status: health: 11%, Fullness: 0%, Hydration: 0%, Wakefulness: 88%
Example 2:
You see: plant, tree, grass, path, stone
Inventory:
Status: health: 99%, Fullness: 77%, Hydration: 66%, Wakefulness: 77%
Please complete the following code to implement the goal determination function. The function should determine the goals the agent should pursue based on what it can see, what it has in its inventory, its status, and the weather conditions, according to the game rules. Ensure the agent can complete all 22 achievements. Code style requirements: Use clear logic and comments to ensure readability. Follow Python's PEP 8 guidelines.

```
class PriGoal:
        def __init__(self):
        ...
        def determine_goal(self, text_obs):
```

B.2 Quality-Aware Sample Selection

Not all collected trajectory steps are equally informative for planner fine-tuning. The collected trajectories cannot be directly used for VLM fine-tuning because they contain a large number of redundant and less informative steps. For example, consecutive states may repeatedly generate identical goals without providing additional planning information, whereas states near achievement completion or important goal transitions usually contain more valuable supervision signals. Therefore, we design a quality-aware sample selection strategy that assigns each trajectory step a quality score based on multiple signals derived from its trajectory context.

Specifically, the score considers five types of informative signals, including goal transitions, achievement proximity, state evolution, survival urgency, and goal rarity, while applying a repetition penalty to reduce redundant samples. The final quality score combines the base importance score, rarity multiplier, and repetition penalty to prioritize trajectory steps that provide more valuable supervision for planner fine-tuning.

All weighting coefficients and allocation ratios are manually determined based on empirical observations of trajectory characteristics and are not optimized using downstream evaluation results. These heuristic choices are intended to balance different sources of supervision, including goal transitions, achievement-related states, state changes, and rare behaviors, rather than to fit a specific task distribution.

Goal-transition signals. A change in the generated goal set indicates that the planner has revised its high-level strategy in response to new observations or completed subtasks. We add 2.0 when the current goal differs from the preceding step and 1.2 when it differs from the subsequent step. The asymmetric weights emphasize the point at which a new plan first becomes active.

Achievement proximity. A candidate receives an additional 2.0 if it occurs within a 20-step window before or after an achievement is unlocked. Such samples capture states in which the selected goals are closely related to measurable task progress.

State evolution. Inventory changes usually indicate resource acquisition, item consumption, crafting, or tool use. A candidate therefore receives an additional 1.4 when its inventory differs from the preceding state. A change in the set of visible objects contributes an additional 0.8, as it reflects navigation or an updated local interaction context.

Survival urgency. Craftax-Classic requires the agent to maintain health, fullness, hydration, and wakefulness. When any normalized survival statistic falls below 0.5, the current decision may require the planner to trade off survival and long-horizon progression. We assign an additional 1.5 to these states.

Goal rarity. For each normalized goal representation, we count its occurrence frequency among the valid candidates. A frequency-dependent rarity multiplier R_t increases the sampling probability of infrequent goals and is clipped at 4.0. The mapping from occurrence frequency to R_t is empirically defined according to the trajectory statistics and remains fixed during dataset construction.

Repetition penalty. If the same normalized goal appears within the preceding five steps, the candidate is multiplied by 0.35. This penalty is not applied when the candidate already has a base quality score of at least 4.0, since a repeated goal may remain informative near an achievement or substantial state transition.

Let

$$\begin{aligned} B_t = 1.0 &+ 2.0\,\mathbb{I}[y_t \neq y_{t-1}] + 1.2\,\mathbb{I}[y_t \neq y_{t+1}] \\ &+ 2.0\,\mathbb{I}[\mathrm{dist}_{\mathrm{ach}}(t) \leq 20] + 1.4\,\mathbb{I}[\Delta \mathrm{Inv}_t] \\ &+ 0.8\,\mathbb{I}[\Delta \mathrm{Vis}_t] + 1.5\,\mathbb{I}[\min(l_t) < 0.5], \end{aligned} \tag{19}$$

where $\mathrm{dist}_{\mathrm{ach}}(t)$ denotes the temporal distance to the nearest unlocked achievement, $\Delta \mathrm{Inv}_t$ denotes an inventory change, and $\Delta \mathrm{Vis}_t$ denotes a change in visible objects.

The repetition penalty is

$$P_t = \begin{cases} 0.35, & \exists\, j \in \{t-5, \ldots, t-1\} : y_j = y_t \text{ and } B_t < 4.0, \\ 1.0, & \text{otherwise.} \end{cases} \tag{20}$$

The final quality score is

$$Q_t = B_t \cdot \min(R_t, R_{\max}) \cdot P_t + \epsilon_t, \qquad \epsilon_t \sim \mathrm{Uniform}(0, 0.15), \tag{21}$$

A frequency-dependent rarity multiplier R_t increases the sampling weight of infrequent goals and is clipped by the hyperparameter $R_{\max}$. We set $R_{\max} = 4.0$ in our experiments.

Episode-Level Balanced Allocation. A purely global ranking would favor long trajectories because they contain more candidate states, whereas uniform sampling over trajectories would overrepresent short episodes and distort the natural interaction distribution. We combine uniform episode coverage with size-proportional allocation.

Let N_i denote the number of valid candidates in episode i, N_{ep} the number of episodes containing at least one valid candidate, and T the target number of samples. The quota assigned to episode i is

$$M_i = \left\lfloor \mathcal{T}\left(0.30\frac{1}{N_{\text{ep}}} + 0.70\frac{N_i}{\sum_j N_j}\right)\right\rfloor. \tag{22}$$

The first term assigns 30% of the target capacity uniformly across episodes, ensuring broad trajectory coverage. The second term allocates the remaining 70% in proportion to the number of valid candidates, approximately preserving the empirical trajectory distribution. Within each episode, valid candidates are sorted by Q_t, and the top $\min(M_i, N_i)$ candidates are retained.

B.3 Preference Pair Construction

For each selected state x, the trajectory-consistent goal response is used as the chosen response y_c. We construct a rejected response y_r that remains syntactically valid but is less consistent with the current trajectory context. Rather than relying on arbitrary global negative sampling, we use goals observed in the same trajectory whenever possible. This produces hard negatives that share the environment, achievement progression, and planner vocabulary of the chosen response.

We employ the following five strategies.

Future sampling. With probability 0.35, we sample a goal response from the subsequent 40 trajectory steps. The sampled response must have a goal set different from y_c. Future goals can be individually plausible but premature under the current state, making them useful negatives for learning temporal feasibility.

Past sampling. With probability 0.25, we sample a different response from the preceding 40 steps. Past goals commonly correspond to already completed prerequisites or states that are no longer relevant.

Mixed construction. With probability 0.25, we retain one or two subgoals from y_c and replace the remaining subgoals using alternatives observed in neighboring steps. This strategy creates fine-grained preference samples in which the chosen and rejected responses overlap partially but differ in their overall goal composition.

Priority shuffling. With probability 0.05, we preserve the three goals but alter their order and corresponding priority assignment. These pairs supervise the planner to distinguish not only which goals are relevant but also which goal should be prioritized.

Global trajectory sampling. With probability 0.10, we sample a different goal response from any position in the same trajectory. This strategy increases negative diversity when suitable local alternatives are unavailable.

Each rejected response is accepted only if it satisfies the following conditions:

1. it contains exactly three parseable subgoals;
2. all priority values lie in $[0, 1]$;
3. its priority values satisfy the required uniqueness constraint;
4. its normalized textual form differs from that of y_c;
5. its goal composition or priority ordering differs from y_c.

The resulting preference dataset is

$$\mathcal{D}_{\text{pref}} = \{(x_i, y_c^i, y_r^i)\}_{i=1}^{N_{\text{pref}}}, \qquad y_c^i \succ y_r^i, \tag{23}$$

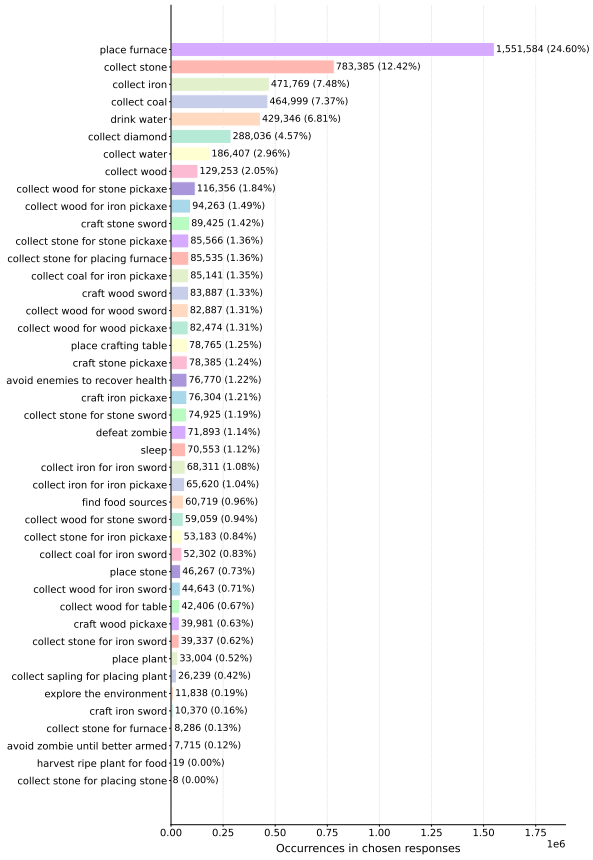

Figure 7: Complete distribution of the 42 goals in the constructed preference dataset.

where y_c^i and y_r^i denote the chosen and rejected goal responses, respectively. The chosen response is grounded in the planner-guided trajectory, whereas the rejected response is constructed to violate its temporal relevance, goal composition, or priority ordering.

Figure 7 presents the complete distribution of the chosen goals in the constructed preference dataset. The dataset covers 42 distinct goals and exhibits a naturally long-tailed distribution. Early-game goals related to resource collection and tool construction occur most frequently because they serve as prerequisites for many later achievements. Goals corresponding to advanced crafting, exploration, combat, and diamond acquisition appear less frequently but are still retained through the quality-aware sample selection procedure. As a result, the dataset preserves the natural gameplay distribution while maintaining sufficient coverage of rare planning decisions across different task stages.

Illustrative Example. Figure 8 presents a representative preference pair from CCGP-2M. For readability, we simplify the shared instruction template and show only selected instructions together with the sample-specific image, textual state, and goal responses.

C Supplementary Material for Environments

C.1 Experimental Setup

C.1.1 Environments

Craftax-Classic is a JAX-based reimplementation of Crafter (Hafner, 2022) that preserves its open-world survival and crafting mechanics while enabling faster simulation (Matthews et al., 2024). It features a procedurally generated 2D world in which the agent explores, gathers resources, crafts

Representative Preference-Pair Sample

Multimodal Input x



You are a high-level planner for the Craftax-Classic game. Your task is to generate training goals based on the environment state. Output format (strictly follow):
subgoal1: score1, subgoal2: score2, subgoal3: score3

You see: tree, arrows, lava, grass, coal, skeletons, path, and stone.

Inventory: stone: 2, coal: 1, iron: 1, wood pickaxe: 1, stone pickaxe: 1, wood sword: 1, and stone sword: 1.

Status: health: 7, fullness: 8, hydration: 4, wakefulness: 6, and sky brightness: 80%.

⇓

Chosen Response y_c

Goal	Priority
Place furnace	0.354
Collect coal	**0.338**
Collect wood for iron pickaxe	**0.308**

Rejected Response y_r

Goal	Priority
Place furnace	0.377
Collect stone	**0.328**
Collect iron for iron sword	**0.295**

Figure 8: A representative preference-pair sample from CCGP-2M. The multimodal input contains a visual observation and its textual description.

tools, combats hostile entities, and unlocks achievements. The environment includes 22 achievements, $63 \times 63 \times 3$ visual observations, and 17 discrete actions. Its sparse rewards, partial observability, and resource dependencies make it well suited for evaluating goal-guided exploration and long-horizon planning.

For a fair comparison, we implement PPO using the official Craftax codebase[3] (Matthews et al., 2024). The PPO hyperparameters used for Craftax-Classic are summarized in Table 3.

C.1.2 Training Configuration

For ELLM and AdaRefiner, we use DeepSeek-V3 (Liu et al., 2024) as the planner. For SGRL, the planning code is generated by DeepSeek-R1. For both the baselines and our framework, goal-text embeddings are computed using `paraphrase-MiniLM-L6-v2` (Reimers & Gurevych, 2019).

Both Causal-aware LLMs and our framework use the open-source Qwen3-VL-8B-Instruct model as the planner. Causal-aware LLMs use the model without fine-tuning, whereas our framework employs a preference-aligned planner. The training configurations for SFT, direct preference optimization (DPO), KL-DPO, and KL-TDPO are reported in Table 4.

C.2 Main Results

Figures 9 and 10 report the final success rates across the 22 achievements in Craftax-Classic. Overall, our method achieves higher success rates on most achievements, indicating that its performance gains are not limited to a small subset of tasks. Figure 9 compares VLM-guided RL methods that use planners trained with different fine-tuning strategies. The RL agent guided by the KLTDPO-trained planner ranks first on 14 achievements and second on 2 achievements. These results show that KLTDPO provides more effective goal guidance than the Base VLM and the planners trained

[3] `https://github.com/MichaelTMatthews/Craftax`.

Parameter	Value
Learning Rate	7×10^{-4}
Optimizer	AdamW
Batch Size	512
Number of Environments	256
Update Epochs	4
Clip Ratio	0.2
Discount Factor γ	0.97
Entropy Coefficient	0.01
Value Function Coefficient	0.5
Activation Function	ReLU

Table 3: PPO hyperparameters for Craftax-Classic.

Configuration	SFT	DPO	KLDPO	TDPO	KLTDPO
Fine-tuning Method	LoRA	LoRA	LoRA	LoRA	LoRA
LoRA Rank	32	32	32	32	32
LoRA Alpha	64	64	64	64	64
LoRA Dropout	0.05	0.05	0.05	0.05	0.05
Preference Coefficient β	–	0.1	0.1	0.1	0.1
Learning Rate	4×10^{-5}	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Per-device Batch Size	32	8	8	8	8
Gradient Accumulation Steps	2	4	4	4	4
Effective Batch Size per Device	64	32	32	32	32
Training Epochs	1	1	1	1	1
Learning-rate Scheduler	Cosine	Cosine	Cosine	Cosine	Cosine
Warm-up Ratio	0.03	0.03	0.03	0.03	0.03
Maximum Sequence Length	Default	2,048	2,048	2,048	2,048
Numerical Precision	BF16	BF16	BF16	BF16	BF16
Gradient Checkpointing	Enabled	Enabled	Enabled	Enabled	Enabled
KL Weight	–	–	0.05	–	0.05

Table 4: Configurations for supervised fine-tuning and preference optimization.

with SFT, DPO, or KLDPO. Figure 10 compares our method with ELLM, which uses large-model guidance, and PPO, which does not use language-model guidance. Our method ranks first on 17 achievements and second on 2 achievements. This comparison shows that the proposed local VLM planner provides stronger exploration guidance than both conventional RL and the evaluated LLM-guided RL method. Together, these comparisons demonstrate that our method improves overall task performance rather than benefiting only a few individual achievements.

C.3 Data Quality Analysis

As shown in Figure 11, we further compare planners trained using different fine-tuning methods on datasets with and without quality-aware sample selection. The planner trained on CCGP-2M with KLTDPO achieves the highest success rate on 13 achievements and the second-highest success rate on 3 achievements. This result indicates that the combination of quality-aware preference data and KLTDPO improves downstream RL performance across a broad range of achievements.

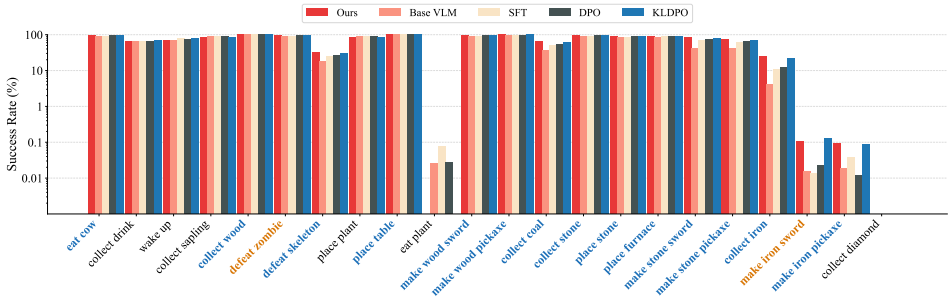

Figure 9: The success rates across 22 Craftax-Classic achievements under different methods. Blue and orange achievement names denote the tasks where our method ranks first and second, respectively.

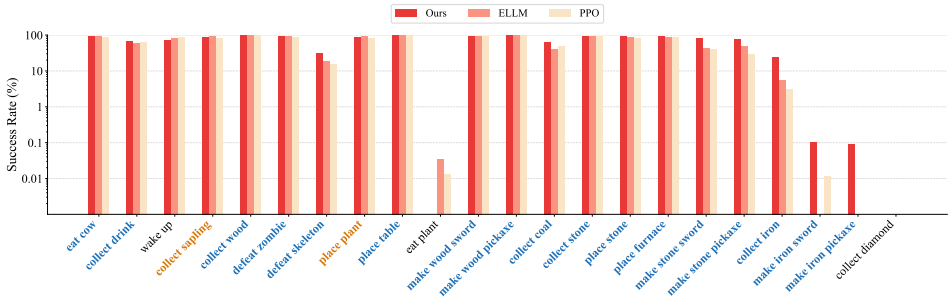

Figure 10: The success rates across 22 Craftax-Classic achievements under different methods. Blue and orange achievement names denote the tasks where our method ranks first and second, respectively.

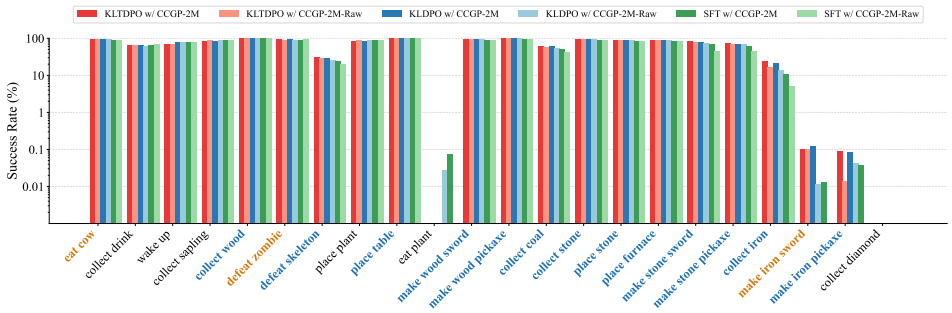

Figure 11: The success rates across 22 Craftax-Classic achievements under different methods. Blue and orange achievement names denote the tasks where our method (KLTDPO w/ CCGP-2M) ranks first and second, respectively.