

# The Dark Side of VLM Rewards: Understanding and Mitigating False Positive Noise

Anonymous Author(s)

Affiliation

Address

email

1           **Abstract:** Vision-Language Models (VLMs) have shown promise as reward mod-  
2       els for reinforcement learning (RL) agents, potentially providing rich reward sig-  
3       nals based on how well an agent’s trajectory aligns with expert instructions. How-  
4       ever, our observations reveal that RL agents trained with VLM rewards often un-  
5       derperform compared to those employing only intrinsic (exploration-driven) re-  
6       wards, contradicting expectations set by recent work. To understand this unex-  
7       pected outcome, we analyze the noisy reward signals from VLMs, particularly  
8       focusing on the prevalence and consequences of false positives and false nega-  
9       tives. Our analysis revealed that false positive rewards of VLMs were a major  
10      factor contributing to the poor learning efficacy. To address this, we developed  
11      a novel reward function, BiMI, that prioritizes minimizing false positive rewards  
12      while accepting a slight increase in false negatives as a necessary trade-off. Experi-  
13      ments demonstrate that BiMI substantially improves agent performance across  
14      challenging environments, highlighting the critical importance of addressing re-  
15      ward noise when applying VLM-based rewards to complex real-world scenarios.

16           **Keywords:** RL, VLM-based Reward Model, Sparse Reward Environments

## 17      1 Introduction

18      Natural language instructions are increasingly recognized as a valuable source of reward signals for  
19      reinforcement learning (RL) agents to learn complex tasks. In particular, a growing trend in agent  
20      learning involves using vision-language models (VLMs) for reward modeling. This approach mea-  
21      sures the semantic similarity – often quantified by cosine similarity – between the embedding rep-  
22      resentations of an agent’s behaviors (i.e., past trajectories) and the provided instructions, all within  
23      the same embedding space [1, 2, 3, 4].

24      However, we observed that RL agents trained with VLM reward models, while effective in simpli-  
25      fied settings, often struggled with tasks involving complex dynamics and longer action horizons.  
26      This is evident in several recent works – for instance, Goyal et al. [2] reported the effective use  
27      of VLM rewards in Montezuma’s Revenge, a notoriously challenging Atari game. However, we  
28      observed that this success was confined to individual sub-tasks and the agent struggled when at-  
29      tempting to scale up to the full game. Similarly, Du et al. [4] demonstrated impressive performance  
30      of VLM rewards in guiding agents within a 2D survival game. However, their study was conducted  
31      in a modified environment with a reduced observation and action space using internal game state  
32      information and manually defined macro actions. Consequently, when tested in the original, un-  
33      modified environment, their agent’s performance did not exceed that of agents using only *intrinsic*  
34      (exploration-driven) rewards.

35      The consistent underperformance of VLM-based models in complex environments, **particularly**  
36      **their unexpected failure to outperform even simple exploration-driven rewards**, raised sig-  
37      nificant concerns about their reliability in real-world applications. This discrepancy, where VLM

38 reward models underperformed contrary to their perceived potential, prompted us to investigate the  
 39 underlying causes of this performance gap. Our findings indicate that **noisy reward estimates** in  
 40 VLMs are a key factor contributing to poor learning efficacy. We attribute this noise primarily to  
 41 the approximation errors inherent in the *cosine similarity metric* commonly used in these models.  
 42 Specifically, our analysis focuses on two types of noise: false positives, which involve rewarding  
 43 unintended behaviors, and false negatives, which occur when correct behaviors are not rewarded.

44 We posit that false positive reward noise (see Figure 1)  
 45 are not only more prevalent but potentially more detrimental to the learning process than false negatives, a  
 46 hypothesis supported by our empirical and theoretical  
 47 findings. To this end, we propose a novel reward function,  
 48 **BIMI** (**B**inary **M**utual **I**nformation **R
 49 leverages binary reward signals to directly reduce the  
 50 occurrence of false positives and incorporates a mutual  
 51 information term to prevent overfitting to potentially  
 52 noisy signal sources. Our experiments demonstrate  
 53 that BIMI significantly improves the learning efficacy  
 54 of agents trained by VLM rewards across various chal-  
 55 lenging environments.**

## 57 2 Related Work

58 **Use of VLMs as Reward Models.** VLMs have been  
 59 pivotal in robotics studies, serving as reward models  
 60 that guide agents to follow instructions [5, 6, 7]. While  
 61 most research primarily focuses on leveraging VLMs  
 62 to overcome the challenge of manual reward design  
 63 for complex tasks [8], the impact of reward noise and  
 64 its implications for policy convergence rates are often  
 65 overlooked. As mentioned in Section 1, some work  
 66 sidesteps the noisy reward problem by accessing inter-  
 67 internal state information from the game engine as well as  
 68 providing predefined action macros [4, 9], thereby pre-  
 69 venting the accumulation of noise over longer horizon.  
 70 However, understanding the impact of reward noise  
 71 from VLMs is crucial for developing reliable language  
 72 interface for RL agents, as it directly affects their ability to learn effectively in real-world tasks.

73 **Mitigating Reward Model Failures.** Research by Ghosal et al. [10] and Fu et al. [11] has intro-  
 74 duced methods to counteract unreliable reward signals from learned VLMs. These strategies involve  
 75 employing a parallel exploration-based policy alongside the reward-maximizing policy, thereby re-  
 76 ducing reliance on potentially misspecified VLM rewards. Our work contributes to this growing  
 77 body of research by proposing a novel reward function that directly addresses the issue of noisy  
 78 rewards in VLM-based models, complementing approaches that use exploration policies to escape  
 79 local optima. Furthermore, we have tested the synergy of combining our reward function with explo-  
 80 ration strategies, demonstrating how these approaches can be integrated for improved performance.

## 81 3 Formal Problem Statement

82 In the context of using instructions as auxiliary guide for RL agents, we frame our task as a MDP  
 83 defined by a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0, r^e, \gamma \rangle$ , where  $\mathcal{S}$  represents a set of states  $s \in \mathcal{S}$ ,  $\mathcal{A}$  represents a set  
 84 of actions  $a \in \mathcal{A}$ , and  $\mathcal{T}(s' | s, a)$  describes the dynamics of the environment.  $s_0 \in \mathcal{S}$  is the initial  
 85 state, and  $\gamma \in [0, 1]$  is the reward discount factor.  $r^e(s, a, s')$  is the environmental reward function.

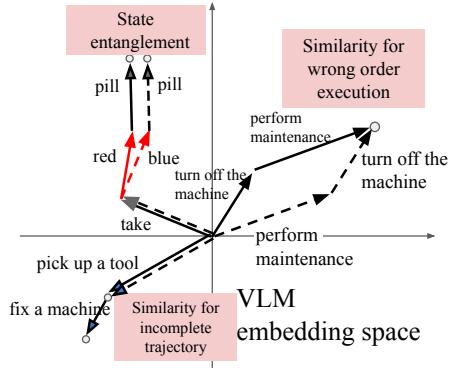


Figure 1: Schematic diagram of false positives in a VLM embedding space. The unintended agent’s trajectory (dashed line) may exhibit high cosine similarity to the instruction (solid line) in the embedding space, as indicated by the proximity of their endpoints in the embedding space. Despite this apparent similarity, the unintended trajectories fundamentally fail to fulfill the intended instruction. Therefore, the rewards guides the agent towards incorrect behaviors. The figure shows three distinct cases of false positives. Refer to Section 4 for more details.

86 In this work, we focus on a sparse reward setting, where the agent receives a non-zero reward only  
 87 when reaching goal states  $S_G \subset \mathcal{S}$ , and 0 otherwise, with  $|S_G| \ll |\mathcal{S}|$ . An agent's trajectory is a  
 88 sequence of states and actions  $\tau = \langle s_t, a_t, \dots, s_{t+T} \rangle$ , where  $t$  is the initial time step and  $T$  is the  
 89 length of the trajectory. The objective is to learn a policy  $\pi$  that maximizes the expected cumulative  
 90 reward of the trajectories  $\tau^\pi = \{\tau_1, \dots, \tau_n \mid \tau_i \text{ induced by } \pi\}$ .

91 A sparse reward function only *realizes* a set of acceptable policies, rather than distinguishing be-  
 92 tween them more finely (as derived from Abel et al. [12]; see Appendix A.1 for details). Building  
 93 on this, we define the following:

94 (1) *Policy Universe*,  $\Pi$ : the set of all possible policies for the MDP.

95 (2) *Acceptable Policies*,  $\Pi_G$ : the policies in  $\Pi_G$  are those policies whose start-state values  $V^\pi(s_0)$   
 96 are within a small  $\epsilon$  of the optimal value, while all other policies have strictly lower values.

97 (3) *Unacceptable Policies*,  $\Pi_B$ : formally,  $\Pi_B = \Pi \setminus \Pi_G$ , which refers to the set of policies that do  
 98 not meet the near-optimal criteria and are therefore considered not acceptable.

99 A task instruction  $L$  is a sequence of natural language sentences  $L = \{l_1, l_2, \dots, l_n\}$ , where each  $l_k$   
 100 induces a set of desired sub-trajectories  $\tau^{l_k}$ . A complete task execution is represented by a trajectory  
 101 that concatenates instructions' corresponding sub-trajectories in order, traversing from the initial  
 102 state  $s_0$  to a goal state in  $S_G$ .

103 An auxiliary VLM-based reward model provides more  
 104 frequent rewards by evaluating the semantic similarity  
 105 between the agent's trajectory and the specified in-  
 106 struction sentence. This can be formally represented as  
 107  $r^v(\tau, l_k) = p(l_k|\tau)$  (see Figure 2), where  $r^v$  denotes the  
 108 VLM-based reward function. The use of *non-Markovian*  
 109 reward functions in MDP environments has been well-  
 110 established in the literature, particularly through the  
 111 work on reward machines [13, 14]. Such a VLM-based  
 112 reward function consists of two components: an embed-  
 113 ding encoder  $\phi(\cdot)$  and a similarity function. The reward  
 114 is calculated as the cosine similarity between the embed-  
 115 dings, which can be expressed as  $\frac{\phi(\tau) \cdot \phi(l_k)}{\|\phi(\tau)\| \|\phi(l_k)\|}$ .

116 **Reduction of Convergence Rate.** To understand the  
 117 effects of reward noise on policy learning, it's crucial to  
 118 analyze its impact on the convergence rate. In sparse re-  
 119 ward settings, the convergence rate – defined as the number of training iterations needed to learn an  
 120  $\epsilon$ -optimal policy – can be significantly affected by false positive rewards that are incorrectly given  
 121 for policies that do not lead to the goal states.

122 To illustrate this, we analyze the convergence rate by measuring the probability of learning accept-  
 123 able policies ( $P(\pi \in \Pi_G)$ ) over each update iteration. The following theorem demonstrates how  
 124 false positive rewards impact the convergence rate:

125 **Theorem 3.1** (Convergence rate reduction). *In Actor-Critic algorithm, gradient ascent on*  
 126  $Q(s, a)\pi_{\theta_i}(a|s)$  *pushes the next updated policy*  $\pi_{\theta_{i+1}}$  *in the direction provided by the Q value func-*  
 127 *tion. In the direction of maximizing*  $P(\pi \in \Pi_G)$ , *the gradient of*  $Q(s, a)\pi(a|s)$  *can be expressed as*  
 128 *follows:*

$$\begin{aligned} \nabla_\theta Q_\phi(s, a)\pi_\theta(a|s) &= \text{const} \cdot \mathbb{E}[G^{\pi \in \Pi_B}] \nabla_\theta \pi_\theta(a|s) \\ &\quad + (\mathbb{E}[G^{\pi \in \Pi_G}] - \mathbb{E}[G^{\pi \in \Pi_B}]) P(\pi \in \Pi_G) \nabla_\theta \pi_\theta(a|s) \end{aligned}$$

129 where  $\Pi_G$  is the set of acceptable policies;  $\Pi_B$  is the set of unacceptable policies that follow in-  
 130 structions but fail to reach the goal;  $\mathbb{E}[G^\pi]$  is expectation of the cumulative rewards by executing  
 131 policy  $\pi$  in one episode.

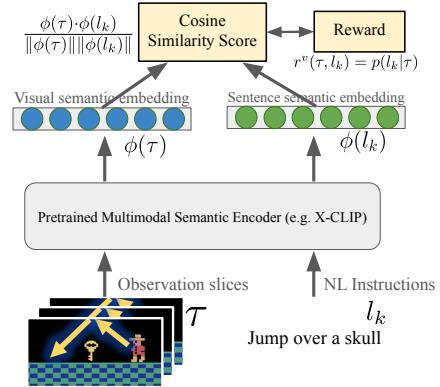


Figure 2: Illustration of reward signals from a vision-language reward model

132 The proof is in Appendix A.2. In this context, false positive rewards correspond to rewards assigned  
133 to policies within the set  $\Pi_B$ , which represents undesirable policies. Therefore,  $\mathbb{E}[G^{\pi \in \Pi_B}]$  can  
134 be interpreted as the expected cumulative false positive rewards. This theorem highlights how the  
135 presence of  $\mathbb{E}[G^{\pi \in \Pi_B}]$  reduce the convergence rate by: (1) introducing deviation in the policy update  
136 directions (the first term), and (2) reducing the gradient of the desired policy update (the second  
137 term). Both effects can significantly hinder the convergence process.

138 The issue of noisy rewards is not merely a theoretical concern; it has practical implications for  
139 the effectiveness of instruction-guided RL in real-world applications. In the next section, we will  
140 investigate how cosine similarity metric used in VLM-based reward models inevitably introduce  
141 false positive rewards. Additionally, we will empirically evaluate the impact of both false positives  
142 and false negatives on the learning efficacy of agents in sparse reward environments.

## 143 4 Approximation Error of Cosine Similarity

144 In this section, we will discuss two fundamental issues associated with cosine similarity scores in RL  
145 contexts: *state entanglement* and *composition insensitivity*. The former issue, state entanglement,  
146 refers to the metric’s inability to recognize trajectories  $\{\tau_1, \dots, \tau_n \mid \tau_i \text{ induced by } \pi \in \Pi_B\}$  that,  
147 while being cosine similar to the target instruction in the embedding space, fail to reach the goal  
148 states in  $S_G$ . The latter issue refers to the metric’s tendency to reward trajectories that execute  
149 the sub-tasks (corresponding to individual instruction sentences  $l_k$ ) in an incorrect order. This is  
150 problematic in RL tasks where the specific sequence of executing sub-tasks, as defined by the order  
151 of instructions in  $L = \{l_1, l_2, \dots, l_n\}$ , is crucial for reaching the goal states.

152 **The Issue of State Entanglement** State entanglement refers to the issue where the cosine similarity  
153 metric erroneously pays more attention to lexical-level similarity while lacking comprehension  
154 of the underlying state transitions. Consequently, rewards are given to trajectory-instruction pairs  
155 that are cosine similar in embedding space but in fact result in distinct state transitions. For in-  
156 stance, consider the significant contrast between “take the red pill” and “take the blue pill”. Despite  
157 their lexical similarity, they lead to vastly different states. However, the cosine similarity metric  
158 may represent them as similar due to the shared action “take” and shared object “pill”, disre-  
159 garding the critical difference in state outcomes. Understanding state transitions is crucial in sequential  
160 decision-making scenarios. Otherwise, rewards may be given to trajectories that lead to unintended  
161 states, resulting in the learning of unacceptable policies.

162 **The Issue of Composition Insensitivity** Composition insensitivity in cosine similarity metrics  
163 gives rise to two issues: (1) *rewarding incomplete task execution* – cosine similarity may incorrectly  
164 reward partial task completion, as even incomplete trajectories can receive high similarity score in  
165 the embedding space. For instance, in a task to “pick up a tool, then fix a machine,” the model  
166 might prematurely reward the agent for merely picking up the tool, neglecting the crucial repair  
167 action. We also observed this phenomenon particularly in the Montezuma’s Revenge environment,  
168 where RL agents tend to exploit the VLM reward on simpler aspects of the instruction (e.g., moving  
169 towards a direction) while avoiding the more challenging task of finally grabbing the key and  
170 escaping the room. **Eventually, it leads to suboptimal learning, as agents get stuck in local op-**  
171 **tima, repeatedly performing easier subtasks without progressing towards the ultimate goal.**  
172 (2) *insensitivity to the ordering of execution* – cosine similarity often fails to adequately penalize  
173 incorrect execution sequences. In a safety protocol requiring an agent to “turn off the machinery,  
174 then perform maintenance,” the metric might assign high rewards based merely on the presence of  
175 relevant actions, disregarding their order. While some large language models have become sensitive  
176 to the order of tokens [15], the compact visual and sentence embeddings from multimodal VLMs  
177 remains largely insensitive to sequential information [16]. This limitation can lead to potentially  
178 dangerous outcomes in scenarios where execution sequence is critical, such as in safety protocols or  
179 complex manufacturing processes.

180 Figure 1 illustrates various scenarios where high similarity scores are erroneously assigned to false  
 181 positive cases. To empirically demonstrate the issue, Section 4.1 presents experiments on false  
 182 positive rewards and their impact on agent learning in sparse reward environments.

### 183 4.1 Experiments on Reward Noise Impact

184 Our experiments test the following hypothesis: **(H1)** The two issues of *state entanglement* and  
 185 *composition insensitivity* exist; **(H2)** *false positive* rewards are prevalent during training; **(H3)** VLM  
 186 reward models lacking noise handling mechanisms underperform against intrinsic reward models in  
 187 sparse reward environments; **(H4)** *false negatives* may not be as harmful as *false positives*.

188 **Setup.** We evaluate these hypotheses through various challenging sparse-reward environments:  
 189 (1) *Crafter*, an open-ended 2D Minecraft environment [17]; (2) *Montezuma’s Revenge*, a classic  
 190 hard adventure game in Atari [18]; and (3) *Minigrid ‘Go To Seq’*, a hard task involving long-horizon  
 191 navigation and object interactions [19]. Two cosine similarity-based reward models were tested:  
 192 (1) *Pixel2R* by Goyal et al. [3], which uses only the current video frame to determine if the goal  
 193 state specified in the instruction has been reached; and (2) *ELLM-*, a variant of *ELLM* by Du et al.  
 194 [4]. Unlike *ELLM*, which queries instructions from LLMs in real-time, *ELLM-* directly uses pre-  
 195 set expert instructions and compares them with the transition differences of the agent’s trajectory.  
 196 The VLM backbones used are: (1) *CLIP* [20], pretrained by image-text pairs; and (2) *X-CLIP* [21],  
 197 pretrained by video-text pairs. To ensure high-quality finetuning data, we used internal information  
 198 from the game engine to annotate expert trajectories from expert agents. To demonstrate how noisy  
 199 reward signals hinder learning, we selected a strong intrinsic reward model *DEIR* [22] for com-  
 200 parison. It provides auxiliary rewards based on observation novelty to encourage exploration. See  
 201 Appendix A.3 for detailed implementation of the experiments.

202 **Evaluation Metric.** We used the *score* metric adapted from the Crafter benchmark [17] for per-  
 203 formance evaluation, as it better captures an agent’s consistent performance across multiple subtasks.  
 204 This metric follows the intuitive ‘higher is better’ principle. Unlike *maximum total rewards* metric,  
 205 which fail to capture consistent performance in sparse reward settings, the score metric provides a  
 206 more reliable measure of learning progress.

207 **Reward Noise Issue.** To investigate H1, we evaluated the models’ sensitivity by examining how  
 208 cosine similarity scores change for manipulated trajectory-instruction pairs. The state entanglement  
 209 test involved reversing trajectories and negating instructions (i.e., “do not do  $l_k$ ”). The composi-  
 210 tion insensitivity test examined concatenated pairs of trajectory-instruction data. For example, given  
 211  $(\tau_1, l_1)$  and  $(\tau_2, l_2)$ , we create a concatenated pair  $(\tau_1 + \tau_2, l_1 + l_2)$ . We then test two types of ma-  
 212 nipulations – (1) swapping the order within one modality: e.g.,  $(\tau_2 + \tau_1, l_1 + l_2)$ ; and (2) truncating  
 213 one modality: e.g.,  $(\tau_1, l_1 + l_2)$ . Overall, we categorized the trajectory-instruction pairs for evalua-  
 214 tion into three types: (1) matched pairs – these are positive examples where the trajectory correctly

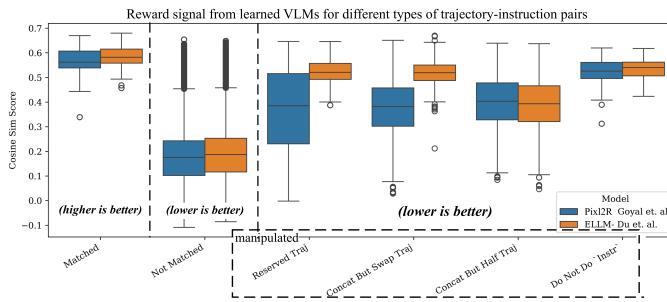


Figure 3: Learned VLM models differentiate between matched and not-matched pairs, but struggle with O.O.D. cases. They incorrectly assign high scores to manipulated pairs, which should be low as the trajectories in the manipulated pairs fail the instruction

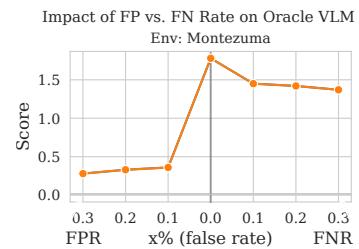


Figure 4: The false positive vs. false negative oracle model. The false positive model get a more severe drop in the final training score.

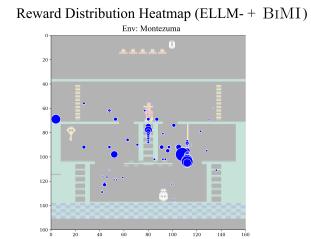
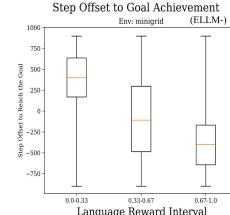
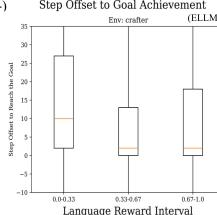
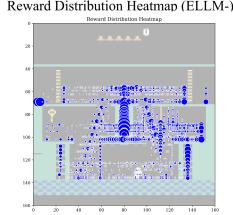


Figure 5: The heatmap shows the cumulative rewards received at various locations, with larger circle sizes indicating higher rewards. The later figures shows the offsets between the state where rewards are given and the actual goal-reaching state. Agents are getting both issues of false positives and false negatives during training

Figure 6: The ratio of false positive rewards is significantly reduced after applying BiMI

215 corresponds to the given instruction. (2) not-matched pairs – these are negative examples where the  
 216 trajectory does not correspond to the given instruction. (3) manipulated pairs: derived from matched  
 217 pairs by altering either the trajectory or the instruction. Ideally, manipulated pairs should receive low  
 218 similarity scores as essentially the trajectory are not fulfilling the instruction. However, our results  
 219 reveal that the reward model assigns high scores to these manipulated pairs, particularly in the “do  
 220 not do  $l_k$ ” case. This finding highlights the noise issue in cosine similarity-based reward models  
 221 (see Figure 3). It’s worth noting that the poor performance in the negation case aligns with broader  
 222 challenges in natural language processing. Recent studies [23, 24] have highlighted that negation is  
 223 central to language understanding but is not properly captured by modern language models. This  
 224 limitation extends to VLMs and significantly impacts their ability to provide accurate rewards in  
 225 complex scenarios and instructions.

226 **Prevalence of False Positives.** To  
 227 address **H2**, we analyzed reward dis-  
 228 tribution heatmap from VLM-based  
 229 reward models during training. The  
 230 heatmap revealed a concerning trend:  
 231 RL agents engage in reward hacking,  
 232 receiving rewards across vast areas  
 233 of the environment rather than just  
 234 at goal states. For instance, in *Mon-*  
 235 *tezuma* environment where the goal is  
 236 to grab the key and escape the room,

237 we observed that agents received rewards even for falling off cliffs, which undoubtedly contribute to  
 238 the detrimental  $\mathbb{E}[G^{\pi \in \Pi_B}]$  specified in Theorem 3.1. For environments without fixed camera views,  
 239 we calculated the step offset between the current rewarded state and the actual goal state. A positive  
 240 offset indicates that a reward was given before reaching the goal state, signifying a false positive  
 241 reward, while a negative offset suggests that the agent reached the goal state but the reward model  
 242 failed to recognize it, indicating a false negative reward (see Figure 5). Interestingly, we observed  
 243 a large amount of negative step offsets in Minigrid environments. We attribute this to Minigrid’s  
 244 abstract shape-based visual representations, which fall outside the VLM’s pretraining distribution.

245 **Impact on Learning.** We trained agents using learned VLM reward models and compared their  
 246 learning efficacy against intrinsic reward models. As shown in Table 7, our results confirmed **H3**:  
 247 *instruction-guided RL agents using learned VLM reward models without noise handling consistently*  
 248 *underperform compared to DEIR, the intrinsic reward-based RL agent.* To investigate the impact of  
 249 false negatives versus false positives (**H4**), we designed an oracle Pixl2R model with two variants: a  
 250 false negative model and a false positive model. The false negative model only rewards the agent for  
 251 reaching subgoal states described in the instruction, with a probability of  $x\%$  that some rewarding  
 252 states in the map are removed. In contrast, the false positive model rewards the agent for reaching  
 253 every subgoal, but also introduces a 0.1 one-off reward for certain locations, covering  $x\%$  of the

Figure 7: Score metric across environments (equivalent to total rewards, higher is better). \* denotes baseline intrinsic reward model. VLM reward models without noise handling underperformed. All models are based on PPO.

Models	Type	Monte.	Minigrid	Crafter	% vs. DEIR
PPO	Pure	0.151	24.993	16.863	-28%
DEIR *	Intrinsic	0.174	55.556	19.758	-
Pixl2R	VLM	0.142	12.422	9.409	-49%
ELLM-	VLM	0.150	19.406	10.826	-41%
Pixl2R + DEIR	VLM + intr.	0.176	17.372	10.440	-38%
ELLM- + DEIR	VLM + intr.	0.178	30.985	11.857	-27%

254 map. The results indicate that false negatives may be less detrimental to agent performance than  
 255 false positives (see Figure 4). This performance difference can be explained through our theoretical  
 256 framework: (1) False negative models maintain the coefficient  $\mathbb{E}[G^{\pi \in \Pi_G}] - \mathbb{E}[G^{\pi \in \Pi_B}]$ , ensuring  
 257 steady gradient ascent towards maximizing  $P(\pi \in \Pi_G)$ ; (2) By decreasing  $\mathbb{E}[G^{\pi \in \Pi_B}]$ , false  
 258 negative models minimize deviations from the target direction, leading to more stable learning.

259 In contrast, scenarios with high  $\mathbb{E}[G^{\pi \in \Pi_B}]$  (i.e., the false positive case) significantly reduce the  
 260 gradient ascent rate and also introduce deviations from the target direction. **These findings challenge**  
 261 **the belief that false negatives are RL degeneracy’s main problem [4]. This insight motivates**  
 262 **our proposed solution, which focuses on reducing false positive rewards, accepting a slight**  
 263 **increase in false negatives as a trade-off.**

## 264 5 Solution to the Reward Noise Issue

### 265 5.1 Binary Signal and Conformal Prediction Thresholding

266 Our experiments have shown that the issue of false positives may be more detrimental to learning  
 267 than false negatives. Based on these findings, we propose a reward function that provides  
 268 one-time binary rewards only when the similarity between the agent’s current trajectory and the  
 269 instruction exceeds a high confidence threshold. This method effectively reduces the likelihood of  
 270 rewarding unacceptable trajectories. To achieve this, we introduce a thresholding mechanism using  
 271 a calibration set of true positive trajectory-instruction pairs. This threshold, denoted as  $\hat{q}$ , is set to  
 272 the empirical quantile of cosine similarity scores at the significance level  $1 - \alpha$ . Pairs whose simi-  
 273 larity scores fall below this threshold  $\hat{q}$  receive no reward. Conversely, pairs exceeding  $\hat{q}$  receive  
 274 a one-time +1 reward, i.e.,  $r_{\text{BI}}^v(\tau, l_k) = \mathbf{1}_{\{p(l_k | \tau) \geq \hat{q}\}}$ . This approach statistically guarantees a high  
 275 probability (at least  $1 - \alpha$ ) that true positive pairs are recognized while minimizing frequency of  
 276 false positives errors [25]. See Appendix A.5 for detailed threshold calculation.

### 277 5.2 Mutual Information Maximization

278 Intuitively, when we observe rewards coming from a particular signal source too frequently, we  
 279 tend to downplay the significance of that signal to avoid over-reliance. This intuition is effectively  
 280 captured by incorporating a *mutual information maximization* term into the reward function. Specif-  
 281 cally, the updated reward function  $r_{\text{MI}}^v(\tau, l_k)$  measures the mutual information between the agent’s  
 282 trajectory and the instruction. Mathematically, it can be expressed as:

$$r_{\text{MI}}^v(\tau, l_k) = I(l_k; \tau) = D_{KL}(p(l_k, \tau) || p(l_k)p(\tau)) \\ = \mathbb{E}_{\tau \sim \pi_\theta, l_k \sim L} [\log p(l_k | \tau) - \log p(l_k)] \quad (1)$$

283 where  $p(l_k | \tau)$  comes from the similarity score provided by VLMs, and  $p(l_k)$  is the frequency  
 284 of the instruction  $l_k$  being fulfilled by the agent’s trajectory. Therefore, the second term in the  
 285 equation serves as a regularization term that downplays the significance of the reward signal when  
 286 it is too frequent. For instance, if a VLM frequently detects that the agent’s actions are fulfilling  
 287 the “climbing the ladder” instruction, even when the agent is performing unrelated tasks, the reward  
 288 signal for this instruction will be downplayed.  $p(l_k)$  is calculated as follows:

$$p(l_k) = \mathbb{E}_{\tau \sim \pi_{\theta-1}} \left[ \frac{\sum_{t=1}^{T_\tau} \mathbf{1}_{\{p(l_k | \tau_t) \geq \hat{q}\}}}{T_\tau} \right] \quad (2)$$

289 Here,  $\tau_t$  is the agent’s trajectory up to time  $t$ , and if the VLM deems the trajectory as fulfilling  
 290 the instruction (i.e.,  $p(l_k | \tau_t) \geq \hat{q}$ ), we increment the count. Dividing the count by the total  
 291 trajectory length  $T_\tau$  gives the empirical frequency of the instruction being fulfilled. The subscript  
 292  $\theta-1$  in  $\pi_{\theta-1}$  indicates that the trajectories are sourced from rollouts in the previous policy iteration,  
 293 acknowledging the impracticality of real-time computation of  $p(l_k)$  during an ongoing episode.

294 To enhance the stability of the training process, we adopt a linearized version of the mutual infor-  
 295 mation maximization approach, as proposed by Li et al. [26]. Overall, BiMI, the proposed reward

296 function that enhances the noise resilience of VLM-based reward models, can be expressed as fol-  
 297 lows:

$$r_{\text{BiMI}}^v(\tau, l_k) = \max(\mathbf{1}_{\{p(l_k|\tau) \geq \hat{q}\}} - p(l_k), 0) \quad (3)$$

298 It's important to note that the BiMI approach primarily mitigates false positives (FP) rather than  
 299 false negatives (FN). By implementing a high confidence threshold and a binary reward signal, we  
 300 effectively reduce the likelihood of rewarding incorrect trajectories, thus addressing the FP issue.  
 301 While this conservative approach may potentially increase FNs, as some correct trajectories might  
 302 fall below the threshold, our experiments indicate that this trade-off is beneficial. We found that false  
 303 positives are more detrimental to learning in instruction-guided RL tasks, justifying our preference  
 304 for a method that prioritizes FP reduction.

## 305 6 Experiments

306 We evaluate the BiMI reward function using baseline agents (*Pixl2R* and *ELLM-*) and their BiMI-  
 307 enhanced counterparts, while also exploring potential synergies with the intrinsic reward model  
 308 DEIR. We follow the same experimental setup as in Section 4.1, with additional details in Ap-  
 309 pendix A.6.

Table 1: Model score across various environments.  $\star$  is the baseline agents with a learned VLM-based reward model to compare with. BiMI significantly improves performance in *Montezuma* and *Minigrid*, while showing mixed results in *Crafter* due to task-specific characteristics

Methods	Montezuma	% vs. $\star$	Minigrid	% vs. $\star$	Crafter	% vs. $\star$
<i>Pixl2R</i> *	$0.142 \pm 0.003$	—	$12.422 \pm 2.439$	—	$9.409 \pm 0.022$	—
<i>Pixl2R</i> + Bi	$0.137 \pm 0.009$	-3.5%	$31.236 \pm 2.040$	151%	$10.735 \pm 0.784$	14%
<i>Pixl2R</i> + BiMI	$0.162 \pm 0.022$	14%	$37.507 \pm 7.832$	199%	$7.951 \pm 0.351$	-15%
<i>Pixl2R</i> + DEIR	$0.176 \pm 0.009$	23%	$17.372 \pm 0.514$	39%	$10.440 \pm 1.015$	10%
<i>Pixl2R</i> + BiMI + DEIR	$0.267 \pm 0.016$	88%	$57.759 \pm 2.157$	364%	$11.014 \pm 0.190$	17%
<i>ELLM</i> *	$0.150 \pm 0.004$	—	$19.406 \pm 10.067$	—	$10.826 \pm 1.017$	—
<i>ELLM</i> + Bi	$0.151 \pm 0.016$	0.6%	$29.788 \pm 1.290$	53%	$11.175 \pm 0.601$	3.2%
<i>ELLM</i> + BiMI	$0.156 \pm 0.014$	4.0%	$33.683 \pm 3.990$	74%	$9.425 \pm 0.267$	-12%
<i>ELLM</i> + DEIR	$0.178 \pm 0.029$	20%	$30.985 \pm 3.507$	59%	$11.857 \pm 1.152$	9.5%
<i>ELLM</i> + BiMI + DEIR	$0.279 \pm 0.078$	86%	$56.281 \pm 6.193$	190%	$13.170 \pm 0.393$	22%

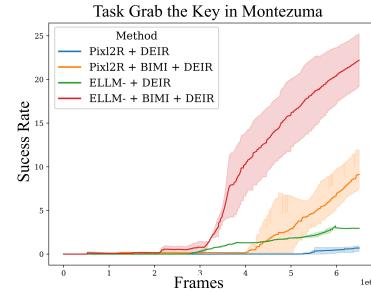


Figure 8: BiMI reward showed faster and higher success rates on difficult tasks in Montezuma

### 310 6.1 Overall Performance and Ablation Study

311 The overall improvements were substantial. As shown in Table 1, BiMI led to a 67% improve-  
 312 ment for *Pixl2R* and a 22% improvement for *ELLM-*. These results are also illustrated in Figure 8  
 313 Figure 13 in Appendix A.7. Our ablation study highlights the distinct contributions of the binary  
 314 reward (Bi) and Mutual Information (MI) components within the BiMI framework. The binary  
 315 reward mechanism alone accounted for a substantial 36.5% improvement in performance. When  
 316 excluding the results from *Crafter*, MI component further contributes a 23% improvement over the  
 317 binary reward alone. Please see Appendix A.7 for result details for each test environment.

## 318 7 Conclusion

319 Our research reveals two key findings in agent learning using VLM-based reward models: (1) false  
 320 positive rewards, rather than false negatives, are more detrimental to policy learning; and (2) our  
 321 proposed BiMI reward function significantly mitigates the slowdown in learning caused by false  
 322 positives, thereby paving the way for more reliable and effective “language as an interface” RL  
 323 systems in practical settings.

324 **Limitation.** Our study primarily focused on linear sequences of language instructions, excluding  
 325 more complex cases. Future research should investigate conditional and ambiguous instructions,  
 326 which likely introduce additional challenges for VLM-based reward models. We also did not explore  
 327 finetuning the VLM during agent training, a useful strategy as discussed by Fu et al. [11].

328 **References**

- 329 [1] R. Kaplan et al. Beating atari with natural language guided reinforcement learning. *ArXiv*,  
330 abs/1704.05539, 2017. URL <https://api.semanticscholar.org/CorpusID:6022828>.
- 331 [2] P. Goyal et al. Using natural language for reward shaping in reinforcement learning.  
332 In *International Joint Conference on Artificial Intelligence*, 2019. URL <https://api.semanticscholar.org/CorpusID:70350059>.
- 334 [3] P. Goyal et al. Pixl2r: Guiding reinforcement learning using natural language by mapping  
335 pixels to rewards. In *Language in Reinforcement Learning Workshop at ICML 2020*, 2020.
- 336 [4] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guid-  
337 ing pretraining in reinforcement learning with large language models. In *International Confer-  
338 ence on Machine Learning*, 2023. URL <https://api.semanticscholar.org/CorpusID:256846700>.
- 340 [5] X. E. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y. fang Wang, W. Y. Wang,  
341 and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for  
342 vision-language navigation. *2019 IEEE/CVF Conference on Computer Vision and Pattern  
343 Recognition (CVPR)*, pages 6622–6631, 2018. URL <https://api.semanticscholar.org/CorpusID:53735892>.
- 345 [6] M. Shridhar et al. Cliprot: What and where pathways for robotic manipulation. In *Conference  
346 on robot learning*, pages 894–906. PMLR, 2022.
- 347 [7] P. Mahmoudieh et al. Zero-shot reward specification via grounded natural language. In *Inter-  
348 national Conference on Machine Learning*, 2022. URL <https://api.semanticscholar.org/CorpusID:250340669>.
- 350 [8] J. Clark. Faulty reward functions in the wild. [https://openai.com/index/  
351 faulty-reward-functions/](https://openai.com/index/faulty-reward-functions/), 2016. Accessed: 2024-08-06.
- 352 [9] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang. Describe, explain, plan and se-  
353 lect: Interactive planning with llms enables open-world multi-task agents. In *Neural Infor-  
354 mation Processing Systems*, 2023. URL <https://api.semanticscholar.org/CorpusID:268042457>.
- 356 [10] G. R. Ghosal, M. Zurek, D. S. Brown, and A. D. Dragan. The effect of modeling human ratio-  
357 nality level on learning rewards from multiple feedback types. In *AAAI Conference on Artificial  
358 Intelligence*, 2022. URL <https://api.semanticscholar.org/CorpusID:251740992>.
- 359 [11] Y. Fu, H. Zhang, D. Wu, W. Xu, and B. Boulet. Furl: Visual-language models as fuzzy re-  
360 wards for reinforcement learning. In *Forty-first International Conference on Machine Learn-  
361 ing*, 2024.
- 362 [12] D. Abel, W. Dabney, A. Harutyunyan, M. K. Ho, M. Littman, D. Precup, and S. Singh. On  
363 the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:  
364 7799–7812, 2021.
- 365 [13] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith. Using reward machines  
366 for high-level task specification and decomposition in reinforcement learning. In *Inter-  
367 national Conference on Machine Learning*, 2018. URL <https://api.semanticscholar.org/CorpusID:51868784>.
- 369 [14] J. Corazza et al. Reinforcement learning with stochastic reward machines. In *AAAI Conference  
370 on Artificial Intelligence*, 2022. URL <https://api.semanticscholar.org/CorpusID:250297195>.

- 372 [15] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with  
373 rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- 374 [16] T. Pham, T. Bui, L. Mai, and A. Nguyen. Out of order: How important is the sequential order  
375 of words in a sentence in natural language understanding tasks? In *Findings of the Association*  
376 for *Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, 2021.
- 377 [17] D. Hafner. Benchmarking the spectrum of agent capabilities. In *Deep RL Workshop NeurIPS*  
378 2021, 2021.
- 379 [18] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment:  
380 An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:  
381 253–279, 2013.
- 382 [19] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and  
383 Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning.  
384 In *International Conference on Learning Representations*, 2018.
- 385 [20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell,  
386 P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from  
387 natural language supervision. In *International Conference on Machine Learning*, 2021. URL  
388 <https://api.semanticscholar.org/CorpusID:231591445>.
- 389 [21] Y. Ma, G. Xu, X. Sun, M. Yan, J. Zhang, and R. Ji. X-clip: End-to-end multi-grained  
390 contrastive learning for video-text retrieval. *Proceedings of the 30th ACM International*  
391 *Conference on Multimedia*, 2022. URL <https://api.semanticscholar.org/CorpusID:250607505>.
- 392 [22] T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian. Noveld: A simple  
393 yet effective exploration criterion. In *Neural Information Processing Systems*, 2021. URL  
394 <https://api.semanticscholar.org/CorpusID:245877021>.
- 395 [23] M. M. Hossain et al. An analysis of negation in natural language understanding corpora. *arXiv*  
396 preprint [arXiv:2203.08929](https://arxiv.org/abs/2203.08929), 2022.
- 397 [24] T. H. Truong, T. Baldwin, K. Verspoor, and T. Cohn. Language models are not naysayers: an  
398 analysis of language models on negation benchmarks. *arXiv preprint arXiv:2306.08189*, 2023.
- 399 [25] M. Sadinle et al. Least ambiguous set-valued classifiers with bounded error levels. *Journal of*  
400 *the American Statistical Association*, 114(525):223–234, 2019.
- 401 [26] M. Li, X. Zhao, J. H. Lee, C. Weber, and S. Wermter. Internally rewarded reinforcement  
402 learning. In *International Conference on Machine Learning*, pages 20556–20574. PMLR,  
403 2023.
- 404 [27] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. On the theory of policy gradient  
405 methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.*, 22(98):  
406 1–76, 2021.
- 407 [28] K. Hornik et al. Multilayer feedforward networks are universal approximators. *Neural net-*  
408 *works*, 2(5):359–366, 1989.
- 409 [29] S. Moon, J. Yeom, B. Park, and H. O. Song. Discovering hierarchical achievements in rein-  
410 forcement learning via contrastive learning. In *Neural Information Processing Systems*, 2023.

412 **A Technical Appendix**

413 Continued from the main text of *Overcoming Reward Model Noise in Instruction-Guided Reinforce-*  
414 *ment Learning*, the technical appendix consist of the following:

- 415 • §A.1 **Property of Sparse Reward Function**, which discuss why sparse reward function  
416 results in the categorization of policies into a set of acceptable policies and a set of unac-  
417 ceptable policies but not others.
- 418 • §A.2 **Proof of the Reduction of Convergence Rate**, which is referred in Section 3.
- 419 • §A.3 **Implementation Details of the Experiments**, which specify the implementation de-  
420 tails for both the first and second stage experiments in Section 4.1 and Section 6 respec-  
421 tively.
- 422 • §A.4 **Additional Details of the Experiments of False Positive Rewards**, which is referred  
423 in Section 4.1.
- 424 • §A.5 **Pseudo-code for Empirical Quantile Calculation for Binary Signal Threshold**,  
425 which is referred in Section 5.1.
- 426 • §A.6 **Additional Implementation Details of the Experiments of BiMI Reward Func-  
427 tion**, which is referred in Section 6.
- 428 • §A.7 **Detailed Experiment Results of BiMI Reward Function**, which provides detailed  
429 experiment results of BiMI reward function, which is referred in Section 6.

430 **A.1 Sparse Reward Function and Range-SOAP**

431 Formally, Abel et al. [12] defined that:

432 **Theorem A.1** (Abel et al. [12]). *A reward function realizes a Range Set of Acceptable Policies*  
433 *(Range-SOAP)  $\Pi_G$  when there exists an  $\epsilon \geq 0$  such that every  $\pi_g \in \Pi_G$  is  $\epsilon$ -optimal in start-state*  
434 *value,  $V^*(s_0) - V^{\pi_g}(s_0) \leq \epsilon$ , while all other policies are worse.*

435 When the reward signal is sparse, the agent only receives a reward upon reaching some goal states,  
436 and the reward function does not provide any feedback during the intermediate steps. We argue  
437 that the sparse reward function realizes a Range-SOAP, leading to the categorization of policies into  
438 acceptable and unacceptable policies.

439 **Proposition A.2.** *Sparse reward function “realizes” Range-SOAP (i.e., Range Set of Acceptable  
440 Policies).*

441 **Justification:**

442 A sparse reward function, where the agent only receives a reward upon completing the task, cannot  
443 prefer policies that lead to shorter task completion times. This is because either the agent completes  
444 the goal very quickly or slowly, they will receive nearly the same amount of cumulative rewards,  
445 and the reward function will not show a strong preference.

446 Since the sparse reward function does not induce a strict partial ordering on the policies, we say  
447 this reward function cannot realize a Partial Ordering (PO) task. Specifically, a PO on policies is  
448 a generalization of a Set of Acceptable Policies (SOAP) task. In a PO, the agent specifies a partial  
449 ordering on the policy space, where some policies are identified as “great”, some as “good”, and  
450 some as “bad” to strictly avoid, while remaining indifferent to the rest.

451 Therefore, the sparse reward function can realize a Set of Acceptable Policies (SOAP), where there  
452 is a set of policies that are all considered “good” or near-optimal, while all other policies are worse.

453 Furthermore, the sparse reward function will lead to a Range-SOAP, rather than an Equal-SOAP.  
454 Specifically, Equal-SOAP is a SOAP where all the acceptable policies are equally optimal in start-  
455 state value. This is because the good policies in the SOAP may differ slightly in their start-state  
456 values, as some may reach multiple goal states in the environment and thereby receiving different

457 cumulative rewards. Therefore, the sparse reward function will realize a Range-SOAP, where there  
458 is a range of acceptable policies that are all near-optimal in start-state value.

459 **A.2 Proof of the Reduction of Convergence Rate**

460 Specifically, the update rule of Actor-Critic algorithm is:

461 • **Critic:**

$$\phi \leftarrow \phi - \alpha_\phi \nabla_\phi (\delta)^2 \quad (4)$$

462 where

463  $\delta = \mathbb{E}_{\pi_\theta}[G^{\pi_\theta} - Q_\phi(s, a)]$  is the Monte Carlo (MC) estimation error,  $Q_\phi(s, a)$  is the  $Q$ -  
464 value function which measures the expected discounted cumulative reward given the  
465 state  $s$  and action  $a$ , and  $\pi_\theta$  is the policy.

466  $G^{\pi_\theta}$  is the rollout cumulative rewards from the trajectory  $\tau^{\pi_\theta}$  generated from  $\pi_\theta$ .

467 • **Actor:**

$$\theta \leftarrow \theta + \alpha_\theta \frac{Q_\phi(s, a) \nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \quad (5)$$

468 We need to make the following assumptions to simplify the theoretical analysis:

- 469 1. We use  $\Pi_G$  to represent the set of acceptable policies and use  $\Pi_B$  as an arbitrary set of “not  
470 acceptable” policies that are learned to follow instruction but fail to reach subgoals, i.e.,  
471  $\Pi_B \cap \Pi_G = \emptyset$ .
- 472 2. Assume the policy class parameterized by  $\theta$  should be expressive enough to capture optimal  
473 or near-optimal policies, and the policy is initialized randomly from uniform distribution,  
474 i.e.,  $\pi_{\theta_0} \sim \mathcal{U}(\Pi)$ . Meanwhile, the Q-value is initialized as  $Q_{\phi_0}(s, a) = 0, \forall s \in S \forall a \in A$ .
- 475 3. We assume that  $|\Pi_G|$  and  $|\Pi_B|$  is a fixed number predefined by the task environment while  
476  $\mathbb{E}[G^\pi | \pi \in \Pi_B]$  is non-zero as false positive rewards are unavoidable in real-world VLMs.

477 Since the update rule for  $Q$ -value is a gradient descent on  $\|\mathbb{E}[G^{\pi_\theta} - Q_\phi(s, a)]\|^2$  and also we have  
478 that  $\{\Pi_B, \Pi_G, \Pi \setminus (\Pi_B \cup \Pi_G)\}$  is a countable partition of the policy universe  $\Pi$ , the updated  $Q$ -value  
479 will approach as follows:

$$\begin{aligned} Q_\phi(s, a) &\rightarrow \mathbb{E}[G^{\pi_\theta}] \\ &= P(\pi_\theta \in \Pi_B) \cdot \mathbb{E}[G^{\pi_\theta} | \pi_\theta \in \Pi_B] \\ &\quad + P(\pi_\theta \in \Pi_G) \cdot \mathbb{E}[G^{\pi_\theta} | \pi_\theta \in \Pi_G] \\ &\quad + P(\pi_\theta \in \Pi \setminus (\Pi_B \cup \Pi_G)) \\ &\quad \cdot \mathbb{E}[G^{\pi_\theta} | \pi_\theta \in \Pi \setminus (\Pi_B \cup \Pi_G)] \\ &= P(\pi_\theta \in \Pi_B) \cdot \mathbb{E}[G^{\pi_\theta} | \pi_\theta \in \Pi_B] \\ &\quad + P(\pi_\theta \in \Pi_G) \cdot \mathbb{E}[G^{\pi_\theta} | \pi_\theta \in \Pi_G] \end{aligned} \quad (6)$$

480 Given that the update rule for policy  $\pi$  is the gradient ascent on  $Q_\phi(s, a)\pi_\theta(a|s)$ , we have the  
 481 following:

$$\begin{aligned}
 & \nabla_\theta Q_\phi(s, a)\pi_\theta(a|s) \\
 &= (\nabla_\theta P(\pi_{\theta_{old}} \in \Pi_B) \cdot \mathbb{E}[G^{\pi_{\theta_{old}}} | \pi_{\theta_{old}} \in \Pi_B] \pi_\theta(a|s)) \\
 &\quad + (\nabla_\theta P(\pi_{\theta_{old}} \in \Pi_G) \\
 &\quad \cdot \mathbb{E}[G^{\pi_{\theta_{old}}} | \pi_{\theta_{old}} \in \Pi_G] \pi_\theta(a|s)) \\
 &= P(\Pi_B) \cdot \mathbb{E}[G^{\Pi_B}] \nabla_\theta \pi_\theta(a|s) \\
 &\quad + P(\Pi_G) \cdot \mathbb{E}[G^{\Pi_G}] \nabla_\theta \pi_\theta(a|s) \\
 &= (1 - P(\Pi_G) - P(\Pi_B)) \\
 &\quad \cdot \mathbb{E}[G^{\Pi_B}] \nabla_\theta \pi_\theta(a|s) + P(\Pi_G) \cdot \mathbb{E}[G^{\Pi_G}] \nabla_\theta \pi_\theta(a|s) \\
 &= const \cdot \mathbb{E}[G^{\Pi_B}] \nabla_\theta \pi_\theta(a|s) \\
 &\quad + (\mathbb{E}[G^{\Pi_G}] - \mathbb{E}[G^{\Pi_B}]) P(\Pi_G) \nabla_\theta \pi_\theta(a|s)
 \end{aligned} \tag{7}$$

482 Justification of the assumptions.

- 483 • Regarding the non-zero probability of recovering the optimal policy at initialization, it is  
 484 standard in theoretical analyses to assume a uniform distribution of a random variable at  
 485 initialization (see Agarwal et al. [27]). This assumption does not contradict the conclusion  
 486 about the convergence rate deterioration in Theorem 3.1.
- 487 • In reference to the realizability condition implied by Assumption 2, the expressiveness of  
 488 the policy class parameterized by  $\theta$  is an underlying assumption for deep learning models,  
 489 supported by the The Universal Approximation Theorem [28].

490 **Observation.** Since the goal of the learning agent is to maximize  $P(\pi \in \Pi_G)$  (i.e., to converge  
 491 to an acceptable policy), we can see that the second term provides the target direction with rate  
 492 ( $\mathbb{E}[G^{\pi \in \Pi_G}] - \mathbb{E}[G^{\pi \in \Pi_B}]$ ). Therefore, the ascent rate will decrease when the cumulative rewards  
 493 from trajectories that cannot reach the goal state (i.e., the false positive rewards) gets higher. More-  
 494 over, the first term  $const \cdot \mathbb{E}[G^{\Pi_B}] \nabla_\theta \pi_\theta(a|s)$  can be regarded as the deviation of target direction. It  
 495 shows that the level of deviation is also positively proportional to the magnitude of rewards coming  
 496 from failed trajectories. This theorem follows the intuition that the presence of false positive rewards  
 497 can slow down the convergence rate of the learning agent.

### 498 A.3 Implementation Details of the Experiments

#### 499 A.3.1 Environment Details

500 We describe each testing environment used in our experiments. More details introduction can be  
 501 found in on the official project homepage of each benchmark [19, 18, 17].

- 502 • **Crafter** features randomly generated 2D worlds where the player needs to forage for food  
 503 and water, find shelter to sleep, defend against monsters, collect materials, and build tools.  
 504 The original Crafter environment does not have a clear goal trajectory or instructions;  
 505 agents are aimed at surviving as long as possible and exploring the environment to un-  
 506 lock new crafting recipes. We modified the environment to include a preset linear sequence  
 507 of instructions to guide the agent to mine diamond. However, this instruction was found  
 508 to hinder the agent’s performance. The nature of the task requires dynamic strategies and  
 509 real-time decision-making, but the fixed instructions limited the agent. For example, the  
 510 instruction did not account for what to do when the agent is attacked by zombies.
- 511 • **Montezuma’s Revenge** is a classic adventure platform game where the player must nav-  
 512 igitate through a series of rooms to collect treasures and keys. The game is known for its  
 513 sparse rewards and challenging exploration requirements. We manually annotate 97 in-  
 514 structions for the agent to follow, guiding it to conquer the game. The instructions were

515 designed to guide the agent through the game’s key challenges, such as avoiding enemies,  
 516 collecting keys, and unlocking doors.

- 517 • **Minigrid ‘Go to seq’ Task:** We use the ‘Go to seq’ task in the Minigrid environment,  
 518 where the agent must navigate through a sequence of rooms and touch target objects in the  
 519 correct order. This is a sparse reward task where the agent receives a reward of 1 only upon  
 520 completing the entire sequence correctly. During the training phase, we randomly generate  
 521 50 different tasks, each with a room size of 5, 3 rows, and 3 columns. Each task features  
 522 a unique room layout and target object sequence. The instruction complexity is set to 3,  
 523 meaning there are at least 3 target objects to interact with in a specific order.

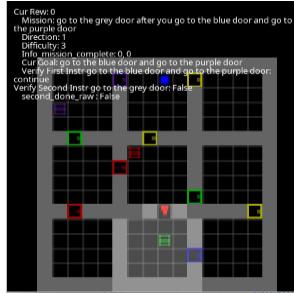
### Montezuma and Instructions



Goal	Room	x	y
climb down the middle ladder	1	74	192
jump right to the yellow rope	1	109	199
jump right to the right platform	1	133	192
climb down the right ladder	1	133	148
jump over the skull to its left	1	76	151
climb up the left ladder	1	21	192
jump up to grab the key	1	13	209
jump left to the left roof	1	44	235

Figure 9: Illustration of the Montezuma’s Revenge task. The agent must navigate through a series of rooms to collect treasures and keys.

### Minigrid and Instructions



Goal
go to the grey door after you go to the blue door and go to the purple door
go to the box, then go to the yellow ball and go to a red door
go to the purple door and go to the purple door; then go to the red box
go to the green box, then go to a box
go to the ball and go to the yellow door after you go to a grey door
go to a ball, then go to the blue ball

Figure 10: Illustration of the Minigrid ‘Go to seq’ task. The agent must navigate through a sequence of rooms and touch target objects in the correct order.

### Crafter and Instructions



Goal (Collect diamond)	
1. Collect wood	2. Collect wood again
3. Place table	4. Make wood pickaxe
5. Collect stone	6. Make stone pickaxe
7. Collect stone	8. Place furnace
9. Collect coal	10. Collect iron
11. Make iron pickaxe	12. Collect diamond

Figure 11: Illustration of the Crafter task. The agent must survive as long as possible and explore for new crafting recipes.

524 **A.3.2 Instruction-Guide Procedure Details**

525 The VLM-based reward model will have a pointer to the sequence of the instruction sentence, start-  
 526 ing at the first sentence. For original models *Pixl2R* and *ELLM*-, we follow the setting in there  
 527 original work where for each instruction sentence (yes, the full instruction essay will be split into  
 528 multiple sentences and treat each sentence as atomic instruction  $l_k$ ), the reward model will have a  
 529 maximum cap of rewards (2.0) it can assign to the agent in one episode. When the cap is reached,  
 530 the reward model will move its pointer to the next instruction sentence. For the BiMI reward model,  
 531 the reward model will move its pointer to the next instruction sentence when the binary signal is  
 532 triggered.

533 **A.3.3 Finetuning VLM-based Reward Models**

534 In contrast to previous work on Instruction-guided RL where they rely on hand-crafted oracle multi-  
 535 modal reward models, we use actual pretrained VLMs to generate reward signals. 2 VLM backbone  
 536 models are used in our experiments: 1) *CLIP* [20], pretrained by image-text pairs; and (2) *X-CLIP*  
 537 [21], pretrained by video-text pairs. In particular, *Pixl2R* uses *CLIP* because it only uses the single  
 538 latest frame as input. In contrast, *ELLM*- takes a slice of trajectory (i.e., multiple frames) as input,  
 539 and thus uses either *X-CLIP* or *CLIP* with additional RNN encoder as the reward model.  
 540 Due to the cartoonish and abstract visuals of the testing environments, we need to further fine-  
 541 tune the VLMs to adapt to this new visual domain. We use well-trained expert agents based on  
 542 Moon et al. [29] to generate expert trajectories for the Crafter environments and annotate them  
 543 with instructions using internal information from the game engine. For Minigrid environments, we  
 544 use classical search-based planning robots to generate expert trajectories and annotate them with  
 545 the corresponding task instructions. For Montezuma’s Revenge, we manually annotate the expert  
 546 trajectories.  
 547 For Minigrid and Crafter, we have 80,000 training pairs, while for Montezuma’s Revenge, we have  
 548 around 300 training pairs. These training data are of high quality, as we have made every effort to  
 549 avoid false positive rewards due to poor training data quality. **To enhance our models’ robustness,**  
 550 **we also employed contrastive learning techniques during VLM training, utilizing similar ma-**  
 551 **nipulated data as hard negatives.** However, despite the fine-tuning process, false positive rewards  
 552 remain unavoidable.

```

1  data_id,instruction,trajectory_chunk_file,trajectory_local_idx
2  0,climb down the middle ladder,montezuma/expert_traj_chunk_0.pkl,0
3  1,walk to the right side of the conveyor belt,montezuma/expert_traj_chunk_0
4  2,jump right to the yellow rope,montezuma/expert_traj_chunk_0.pkl,2
5  3,jump right to the right platform,montezuma/expert_traj_chunk_0.pkl,3
6  4,climb down the right ladder,montezuma/expert_traj_chunk_0.pkl,4
7  5,jump over the skull,montezuma/expert_traj_chunk_0.pkl,5
8  6,climb up the left ladder,montezuma/expert_traj_chunk_0.pkl,6
9  7,jump to grab the key,montezuma/expert_traj_chunk_0.pkl,7
10 8,jump left to the left roof ,montezuma/expert_traj_chunk_0.pkl,8
11 9,use key to open the left door,montezuma/expert_traj_chunk_0.pkl,9
12 10,walk left when the laser gate disappears,montezuma/expert_traj_chunk_0.p
13 11,walk to the middle when the laser gate disappears,montezuma/expert traj_
14 12,wait until the laser gate disappears,montezuma/expert_traj_chunk_0.pkl,1
15 13,approach to the gem,montezuma/expert_traj_chunk_0.pkl,13
16 14,jump to grab the gem,montezuma/expert_traj_chunk_0.pkl,14
17 15,walk to the middle when the laser gate disappears,montezuma/expert traj_
18 16,climb down the middle ladder,montezuma/expert_traj_chunk_0.pkl,16
19 17,wait until the spider goes away,montezuma/expert_traj_chunk_0.pkl,17

```

Figure 12: Example of training data for the Montezuma environment.

553 We used the threshold  $\hat{q}$  introduced in Section 5.1 to make binary classification on the testing pairs  
 554 to evaluate the performance of the fine-tuned VLM-based reward models. We found that VLM  
 555 models had difficulty achieving high accuracy on Minigrid environment, which is likely due to the  
 556 too abstract and cartoonish nature of the environment, causing the VLMs to struggle to learn the  
 557 visual-textual correspondence. We also found that X-CLIP did not perform better than CLIP in  
 558 our experiments. We hypothesize that the cartoonish nature of the testing environments may have  
 559 caused the X-CLIP model to struggle to learn the visual-textual correspondence. Thus, we used  
 560 CLIP as the backbone model throughout our following experiments. The performance of the fine-  
 561 tuned VLM-based reward models is shown in Table 2. **Even when the precision score reaches 0.98,**  
 562 **indicating that only 2% of the rewards are false positives in the validation set, the agent can**  
 563 **still significantly underperform in the testing environments. The core issue is that in out-of-**  
 564 **distribution (O.O.D.) testing environments, false positive rewards are prevalent and inevitable.**  
 565 **Therefore, it is crucial to design a reward function that is robust to reward noise.**

Table 2: Performance of fine-tuned VLM reward model on the testing dataset using the 90th percentile empirical quantile as threshold

Environment	Precision	Accuracy	F1 Score	Recall	Model
Crafter	0.9847	0.9466	0.8538	0.9702	CLIP ELLM-
Crafter	0.9799	0.9028	0.7618	0.9842	CLIP PixI2R
Crafter	0.2095	0.2514	0.2868	0.9657	XCLIP ELLM-
Minigrid	0.7260	0.9200	0.7849	0.9763	CLIP ELLM-
Minigrid	0.6992	0.9086	0.7592	0.9616	CLIP PixI2R
Minigrid	0.1716	0.2310	0.2642	0.9704	XCLIP ELLM-
Montezuma	0.8838	0.9638	0.8825	0.9478	CLIP ELLM-
Montezuma	0.8343	0.9108	0.7652	0.9842	CLIP PixI2R
Montezuma	0.8044	0.9259	0.8045	0.9657	XCLIP ELLM-

### 566 A.3.4 Hyperparameters for Instruction-Guided RL Agents

567 In the experiments, all methods are implemented based on PPO with same model architecture. The  
 568 Minigrid and Crafter environments use the same training hyperparameters as the Achievement Dis-  
 569 tillation paper [29]. For Montezuma’s Revenge, we found that the performance of the agent was  
 570 sensitive to the gamma and GAE lambda parameters. To improve the performance of agents in  
 571 Montezuma’s Revenge, we took two additional steps: (1) normalizing the observation inputs when  
 572 computing the rewards, and (2) not normalizing the advantage during the GAE calculation. The  
 573 hyperparameters are shown in the following tables.

Table 3: Model Parameters

Parameter	Value
model_cls	“PPORNNModel”
hidsize	1024
gru_layers	1
impala_kwarg	
- chans	[64, 128, 128]
- outsize	256
- nblock	2
- post_pool_groups	1
- init_norm_kwarg	
- batch_norm	false
- group_norm_groups	1
dense_init_norm_kwarg	
- layer_norm	true

Table 4: Crafter and Minigrid RL Parameters

Parameter	Value
gamma	0.95
gae_lambda	0.65
algorithm_cls	“PPOAlgorithm”
algorithm_kwargs	
- ppo_nepoch	3
- ppo_nbatches	8
- clip_param	0.2
- vf_loss_coeff	0.5
- ent_coeff	0.01
- lr	3.0e-4
- max_grad_norm	0.5
- aux_freq	8
- aux_nepoch	6
- pi_dist_coeff	1.0
- vf_dist_coeff	1.0

Table 5: Montezuma RL Training Parameters

Parameter	Value
gamma	0.99
gae_lambda	0.95
int_rew_type	“rnd”
pre_obs_norm_steps	50
algorithm_cls	“PPOAlgorithm”
algorithm_kwargs	
- update_proportion	0.25
- ppo_nepoch	3
- ppo_batch_size	256
- clip_param	0.1
- vf_loss_coeff	0.5
- ent_coeff	0.001
- lr	1.0e-4

#### 574 A.4 Additional Details of the Experiments on the Impact of Noisy Rewards

575 **Evaluation Metric Details** In our experiments, we used a score metric adapted from the Crafter  
 576 paper to evaluate agent performance across different environments. This score metric aggregates  
 577 success rates for individual subtasks using a geometric mean. This metric was chosen over the  
 578 *maximum total rewards* metric for several reasons:

- 579 **1. Consistency in Sparse Reward Settings:** Sparse reward environments often pose significant  
 580 challenges for reinforcement learning agents. An agent might occasionally achieve  
 581 high rewards by chance in one rollout but fail to replicate this success consistently in sub-  
 582 sequent rollouts. This variability can lead to misleading evaluations if only the maximum  
 583 total rewards are considered. The Score metric, by measuring the success rate of achieving  
 584 each subgoal, provides a more stable and consistent measure of an agent’s performance.
- 585 **2. Capturing Learning Stability:** The Score metric evaluates the agent’s ability to consist-  
 586 ently reproduce successful behaviors across multiple episodes. This is crucial in sparse  
 587 reward settings, where the agent’s performance can fluctuate significantly. By focusing on  
 588 the success rates of individual subtasks, the Score metric offers a more granular and reliable  
 589 assessment of the agent’s learning progress and stability.
- 590 **3. Crafter Benchmark Standard:** The Crafter benchmark, which introduces the Score met-  
 591 ric, is a well-regarded standard.

592 Crafter codebase provides *score* metric calculation by default. For Minigrid and Montezuma environments,  
 593 we use the internal information from the game engine to detect whether the subtasks are  
 594 completed, thus facilitating the calculation of the *score* metric.

#### 595 A.4.1 Details of Manipulated Trajectory-Instruction Pairs to Evaluate Robustness

596 We evaluated the models' sensitivity by examining how cosine similarity scores change for manipulated  
 597 trajectory-instruction pairs. These manipulations were designed to test the robustness of the  
 598 models against various types of noise. Here's a detailed breakdown of the manipulations:

- 599 1. **Trajectory Reversal:** We inverted the sequence of frames within each trajectory (i.e.,  
 600 `frames = frame[::-1]`) to assess the model's ability to detect reversed state transitions.  
 601 This manipulation tests whether the model can distinguish between forward and backward  
 602 progression in the state transition.
- 603 2. **Instruction Negation:** We modified the original instructions by adding negation (e.g.,  
 604 changing "do  $l_k$ " to "do not do  $l_k$ " or "avoid  $l_k$ "). This tests the model's sensitivity to  
 605 semantic changes in the instruction that fundamentally alter the goal.
- 606 3. **Instruction Rephrasing:** We rephrase the original instructions while maintaining their  
 607 core meaning. This evaluates the model's robustness to linguistic variations and its ability  
 608 to capture the essential semantic content of instructions.
- 609 4. **Concatenation and Order Swapping:** Given two trajectory-instruction pairs  $(\tau_1, l_1)$  and  
 610  $(\tau_2, l_2)$ , we created concatenated pairs and then swapped the order in one modality. For  
 611 example:

- 612 • Original concatenation:  $(\tau_1 + \tau_2, l_1 + l_2)$
- 613 • Swapped trajectory:  $(\tau_2 + \tau_1, l_1 + l_2)$
- 614 • Swapped instruction:  $(\tau_1 + \tau_2, l_2 + l_1)$

615 This tests the model's sensitivity to the order of components in multi-step tasks.

- 616 5. **Concatenation with Partial Content:** We concatenated pairs but truncated one modality.  
 617 For instance:

- 618 • Truncated trajectory:  $(\tau_1, l_1 + l_2)$
- 619 • Truncated instruction:  $(\tau_1 + \tau_2, l_1)$

620 This assesses the model's ability to detect partial mismatches in longer sequences.

#### 621 A.5 Pseudo-code for Empirical Quantile Calculation for Binary Signal Threshold

622 Using empirical quantile as threshold guarantees a high probability (at least  $1 - \alpha$ ) that the true  
 623 positive pairs are recognized while minimizing the average number of mistakes predicting false  
 624 positives [25]:

---

##### Algorithm 1 Calculate Empirical Quantile ( $\hat{q}$ )

---

**Require:** Calibration set  $\{\tau, l\}_n$ , where  $l$  is the instruction sentence,  $\tau$  is the corresponding trajectory, and  $n$  is the number of samples;

Significance level  $\alpha$ ;

VLM model reward model  $v$

- 1:  $\triangleright$  Obtain the similarity-based score
  - 2:  $\{r\}_n \leftarrow \{v(\tau, l)\}_n$
  - 3:  $\triangleright$  Compute the quantile level
  - 4:  $q_{level} \leftarrow \frac{\lceil (n-1) \times (1-\alpha) \rceil}{n}$
  - 5:  $\triangleright$  Compute the empirical quantile
  - 6:  $\hat{q} \leftarrow np.quantile(\{r\}_n, q_{level}, method='lower')$
  - 7: **return**  $\hat{q}$
-

625 **A.6 Implementation Details of the Experiments of BiMI Reward Function**

626 We set confidence level for empirical quantile calculation to be  $1 - \alpha = 0.9$ . We adhered to the  
627 standard requirement of limiting the training budget to 1 million frames [17]. This constraint poses  
628 a significant challenge, particularly in sparse reward settings, as it demands that agents both explore  
629 efficiently and exploit their knowledge effectively within this limited budget.

630 To achieve the 1 million frame budget, we used the following configuration:

- 631 • nproc: 8 (Number of processes used for parallel environments)  
632 • nstep: 512 (Length of the rollout stored in the buffer)  
633 • nepoch: 250 (Number of epochs to train the RL policy)

634 This configuration results in approximately 1 million steps:  $250 \text{ epochs} \times 512 \text{ steps} \times 8 \text{ processes}$   
635  $= 1,024,000 \text{ frames}$ .

636 In the case of Montezuma’s Revenge, we found that the 1 million frame limit used in Crafter was  
637 insufficient due to the game’s complexity and sparse reward structure. To address this, we extended  
638 the training budget to 8 million frames. It’s important to note that even with this increased frame  
639 count, agents were still unable to fully solve the task. As Zhang et al. [22] pointed out, about 1  
640 billion frames are required to truly master Montezuma’s Revenge. This vast difference in required  
641 training time (8 million vs 1 billion frames) underscores the exceptional difficulty of Montezuma’s  
642 Revenge as a sparse reward task.

643 The implementation details for the BiMI reward function are consistent with those outlined in the  
644 first stage of experiments.

645 **A.7 Detailed Experiment Results of BiMI Reward Function**

646 **A.7.1 Montezuma’s Revenge**

647 In *Montezuma*, Pixl2R+BiMI demonstrated an 14% performance increase compared to the original  
648 models (see Table 1), which is slightly below our expectations. We attribute this result to BiMI’s in-  
649 tentional strategy of providing less frequent discrete rewards. While this strategy effectively reduces  
650 false positives, it does not substantially mitigate the inherent reward sparsity issue in *Montezuma*.  
651 However, **we discovered a remarkable synergy between BiMI and intrinsic reward models**.  
652 While previous models showed no significant improvements with *DEIR* (the intrinsic reward model)  
653 alone, combining BiMI and *DEIR* led to a 65% performance gain. The gap in collaboration ef-  
654 fectiveness can be attributed to two factors. In the previous setup, the consistent presence of false  
655 positive rewards misled agents towards unacceptable behaviors and hindered further exploration.  
656 Now, BiMI’s less frequent but more meaningful rewards provide anchor points for the agent’s learn-  
657 ing. Meanwhile, *DEIR*’s intrinsic rewards fill the gaps between these anchor points, encouraging the  
658 agent to explore efficiently in the interim.

659 As illustrated in Figure 6, BiMI rewards are now concentrated on key locations. A significant  
660 improvement is the minimal rewards given for falling off cliffs, which was a common source of  
661 false positives in the original model. Figure 8 demonstrates higher success rate grabbing the key  
662 in the first room, one of the most difficult task in *Montezuma*, highlighting the effectiveness of the  
663 proposed reward function and its collaboration with intrinsic reward models in guiding agents to  
664 solve difficult sparse-reward tasks.

665 **A.7.2 Minigrid**

666 ELLM-+BiMI achieved a remarkable 74% improvement in performance compared to the original  
667 models. This substantial gain is particularly noteworthy given the unique challenges presented by  
668 *Minigrid*. The abstract, shape-based visuals in *Minigrid* diverge drastically from the natural images  
669 used in VLMs’ pretraining, preventing the models from effectively utilizing their prior pretraining

knowledge. Consequently, VLMs struggled to accurately assess similarities between Minigrid's abstract visuals and textual instructions, resulting in highly noisy reward signals. The significant improvement demonstrated by BiMI underscores its effectiveness in handling noisy signals, directly addressing our primary research challenge. This capability is crucial for deploying instruction-guided agents in real-world, unfamiliar scenarios, where visual inputs often deviate from the VLMs' training distribution, leading to noisy reward signals.

### 676 A.7.3 Crafter

677 We observed an intriguing pattern of results. The BI component alone led to 14% and 3.2% improvement in performance over the original models. However, contrary to our observations in other environments, the addition of the MI component actually decreased this improvement. This unexpected outcome can be attributed to the unique nature of *Crafter* task, where agents must repeatedly achieve the same subtasks (e.g., drinking water) for survival. The MI component, designed to discourage over-reliance on frequently occurring signals, inadvertently penalized the necessary repetition of survival-critical actions. Furthermore, note that instruction-guided RL agents, regardless of the reward model employed, were unable to outperform pure RL agents in *Crafter*. This discrepancy is due to the open-world nature of *Crafter*, which requires dynamic strategies and real-time decision-making that our testing setups did not fully capture. Despite these challenges, it is noteworthy that BI alone still managed to improve performance over vanilla VLM-based reward models, suggesting that reducing false positives is still beneficial across all testing environments. The combination of BiMI with *DEIR* (the intrinsic reward model) also showed promising results, indicating a productive balance between exploration (driven by *DEIR*) and exploitation (guided by BiMI instruction reward).

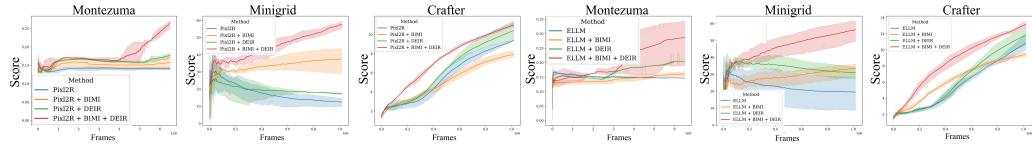


Figure 13: Besides the improvements of the score performance of agents across different environments with the BiMI reward function, it also collaborates well with intrinsic rewards. Combining both can lead to significant performance improvements

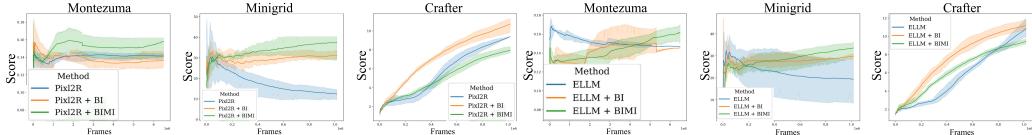


Figure 14: Ablation on the components of BiMI reward function. The binary reward (Bi) alone led to a 36.5% improvement compared to original models. Excluding Crafter, Mutual Information (MI) provided a 23% further improvement over Bi alone