LANGUAGE-CRITIQUE IMITATION LEARNING FROM SUBOPTIMAL DEMONSTRATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Prior work on imitation learning from suboptimal demonstrations typically extracts supervision as a scalar per state or trajectory (*e.g.*, a confidence estimate, discriminator score, or importance weight), which ranks behavior but cannot express task stage, what went wrong, or how to correct it. We propose language-critique imitation learning, an offline algorithm that learns policies with language labels based on expert and suboptimal demonstrations. Our method constructs language labels describing task progress, action optimality, and movement guidance, produced either by a lightweight feature-based generator or an off-the-shelf VLM. We then introduce a language-critique loss to train the policy using language signals without compressing them to scalars. We instantiate this loss for behavior cloning and diffusion policies, yielding LC-BC and LC-DP. We theoretically show that the proposed objective upper-bounds the expert performance gap under standard assumptions. Across seven continuous-control tasks in navigation, driving, and manipulation, LC-BC and LC-DP achieve the highest average success rates in their policy families, improving average success by 8.2% (BC) and 4.3% (DP). Further experiments demonstrate robustness to noisy expert selection and, on one task, comparable performance to full-data BC and diffusion policies using only 10% and 5% of the expert data, respectively. These results show that language critiques provide a practical and reliable learning signal for suboptimal demonstrations.

1 INTRODUCTION

Imitation learning (IL; Pomerleau, 1988; Schaal, 1996; Hussein et al., 2017) has achieved strong results in navigation, driving, and robotic manipulation, but policies trained on limited demonstrations suffer from compounding error (Ross & Bagnell, 2010) and distribution shift (Ross et al., 2011). Collecting more expert demonstrations is a solution, but costly and difficult to scale (Osa et al., 2018). Online methods can reduce this gap by querying the environment or an expert during training (Ho & Ermon, 2016; Fu et al., 2018), but are often unsafe or impractical in real-world settings. These limitations motivate imitation learning from suboptimal demonstrations (Sasaki & Yamashina, 2021; Yu et al., 2023a), in which a small expert set is supplemented with abundant imperfect trajectories that are inexpensive to collect and already available. These suboptimal demonstrations can contain successful behavior, partial progress, and failure modes across a broad range of states and actions. When properly exploited, they can improve policy robustness without further environment interaction.

The key challenge is extracting useful supervision from mixed-quality offline data. Behavior cloning on both expert and suboptimal data risks imitating undesirable behaviors, but filtering out suboptimal data can discard useful experience. Existing offline IL methods exploit both the expert and suboptimal data through scalar signals such as confidence estimates (Zhang et al., 2021), discriminator scores (Xu et al., 2022; Wang et al., 2024b), and importance weights (Sasaki & Yamashina, 2021). These signals can rank which trajectories to imitate, but they do not directly explain why one action is preferable to another or how to adjust it given the current task stage and next subgoal. This type of supervision can be especially helpful for long-horizon and multimodal tasks, where the right action depends on task progress, and for low-quality datasets, where learning requires identifying useful behavior.

Natural language can express the reasons for action preferences and provide guidance for improvement. A language label can describe task progress, the next subgoal, action quality, and corrective guidance (Huan & Fidler, 2017; Zhang et al., 2023; Chen et al., 2024a; Feng et al., 2025; Lee et al.,

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.

2025). In a multi-stage task (BLOCKPUSH in Figure 3d), where a robot must push two blocks into target regions, language labels can specify which block to approach, which target to prioritize, and how to adjust motion, rather than just giving higher or lower scores. This flexibility makes language an attractive learning signal for learning from suboptimal demonstrations. However, realizing this potential requires addressing three challenges: (1) offline datasets usually lack language labels, which must be produced at scale; (2) prior methods typically convert language into scalar signals or preferences (Cao et al., 2024; Wang et al., 2024a), discarding much of its detailed descriptive content; and (3) it remains unclear which aspects of language can contribute to policy improvement.

To address these challenges, we introduce language-critique imitation learning, an offline IL framework that uses language critiques to supervise policies with explicit descriptions of task progress, action quality, and how actions should be improved, turning both expert and suboptimal demonstrations into informative supervision for policy learning. First, we generate structured language labels for each offline data point, using either a feature-based generator or an off-the-shelf VLM. We then formulate a language-critique loss that directly uses the content of these labels to guide policy learning. This loss applies to both feedforward and diffusion policies (Chi et al., 2023), yielding language-critique behavior cloning (LC-BC) and language-critique diffusion policy (LC-DP). We also derive a theoretical bound on the expert-policy performance gap in terms of the language-critique loss (Kakade & Langford, 2002; Jin et al., 2023), under stated assumptions and constraints.

Across seven control tasks spanning navigation, driving, and manipulation, including multimodal settings, LC-BC and LC-DP outperform prior methods (Kim et al., 2022; Xu et al., 2022; Yue et al., 2024; Sun & Song, 2025) in average success rate, improving average success by 8.2% (BC) and 4.3% (DP). We also evaluate robustness to mixed-quality demonstrations and limited expert data. With mixed-quality data, LC-BC and LC-DP achieve the highest success rates within their policy classes at different data-quality levels. With limited expert data, they match full-data BC and DP performance using only 10% and 5% of the expert data. We also examined whether human annotators can generate informative labels that improve policy performance, and whether our method can effectively leverage those human-provided labels. Finally, to examine why language helps, we compare against non-linguistic supervision with matched content and find that language offers benefits beyond the information the labels carry. Together, these results establish language critiques as a practical and reliable learning signal for offline imitation learning from suboptimal demonstrations.

2 RELATED WORKS

Imitation learning (IL) learns policies from expert demonstrations (Schaal, 1996; Mandlekar et al., 2020; Shafiullah et al., 2022; Chen et al., 2024b). Online IL methods (Ross et al., 2011; Ho & Ermon, 2016; Fu et al., 2018; Kostrikov et al., 2020; Lee et al., 2021; Lai et al., 2024; Huang et al., 2024) improve robustness through environment interaction and expert correction, which can be costly and unsafe. Offline IL methods (Pomerleau, 1988; Sasaki & Yamashina, 2021) instead use static expert datasets but suffer from compounding errors (Ross & Bagnell, 2010) and distribution shift (Ross et al., 2011). To address this, recent work augments limited expert data with abundant suboptimal demonstrations (Yu et al., 2023a; Yang et al., 2022; Wu et al., 2019; Brown et al., 2019b), using discriminator or confidence-based reweighting and behavior purification (Kim et al., 2022; Wang et al., 2021; Zhang et al., 2021; Xu et al., 2022; Wang et al., 2024b; Yue et al., 2024; Florence et al., 2021; Chen et al., 2026). Also, offline inverse reinforcement learning approaches recover rewards or world models from mixed-quality data (Watson et al., 2023; Zeng et al., 2023; Brown et al., 2019a; Chen et al., 2020). Furthermore, diffusion policies improve robustness and multimodal behavior modeling (Chi et al., 2023; Pearce et al., 2023; Reuss et al., 2023; 2024). Our method incorporates language label supervision into offline IL with expert and suboptimal demonstrations.

Learning from language. Natural language provides task instructions or conditioning signals to improve generalization in control (Shi et al., 2024; Lynch & Sermanet, 2021; Stepputtis et al., 2020; Sun et al., 2018; 2020; Liu et al., 2023; Lin et al., 2024; Liu et al., 2025). However, most methods focus on discrete or text-based domains, with limited study in continuous control (Hermann et al., 2017; Dai et al., 2025; Xu et al., 2025; Xi et al., 2024). Other approaches convert language into scalar signals or preferences (Cao et al., 2024; Wang et al., 2024a; Jian et al., 2025; Urcelay et al., 2025; Yu et al., 2023b), losing much of its structural information. In contrast, we use language labels as direct supervision, preserving the expressiveness of language to enable robust policy learning.

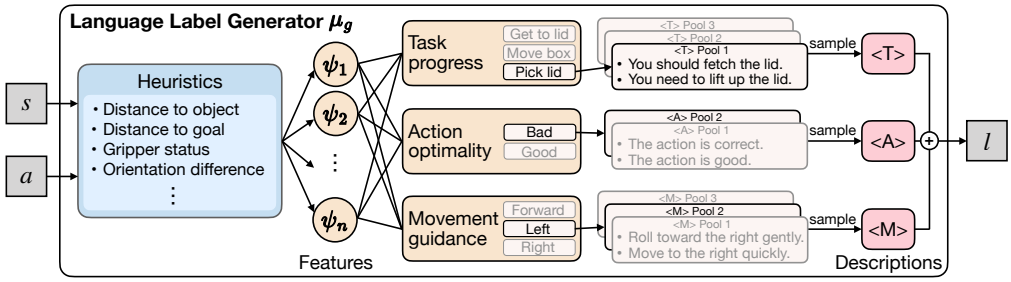


Figure 1: **Language label generator μ_g.** Given a state-action pair (s, a), μ_g extracts task-specific features (*e.g.*, object distance, gripper status) and routes them to three selectors: task progress <T>, action optimality <A>, and movement guidance <M>. Each selector samples a description snippet, concatenated into the final language label l for scalable and structured supervision.

3 PRELIMINARIES

Markov decision process. A finite-horizon Markov decision process (MDP) is defined by the tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma, T)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $P(s' \mid s, a)$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ_0 is the initial state distribution, $\gamma \in (0, 1)$ is the discount factor, and T is the horizon. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ induces a trajectory distribution with $s_0 \sim \rho_0$, and we denote the expected discounted return by $J_r(\pi) := \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]$.

Offline imitation learning from suboptimal demonstrations. We consider offline imitation learning with an expert dataset $\mathcal{D}_E = \{\tau_i^E\}_{i=1}^M$ of M trajectories from expert policy π_E, where $\tau_i^E = \{(s_{i,t}, a_{i,t}^E)\}_{t=0}^{|\tau_i^E|-1}$. The learner also has access to a general dataset $\mathcal{D}_G = \{\tau_i^\beta\}_{i=1}^N$ of N state-action trajectories $\tau_i^\beta = \{(s_{i,t}, a_{i,t})\}_{t=0}^{|\tau_i^\beta|-1}$ collected by behavior policies π_β (Levine et al., 2020; Fu et al., 2020), spanning expert-like and suboptimal behaviors, with $\mathcal{D}_E \subset \mathcal{D}_G$. In our main experiments, both our method and all offline IL baselines are given $\mathcal{D}_E$ and $\mathcal{D}_G$, with the demonstrations belonging to $\mathcal{D}_E$ identified as expert data. The goal is to learn a policy π_θ that imitates expert behavior while exploiting useful demonstrations in $\mathcal{D}_G$, without further environment interaction.

4 APPROACH

We present language-critique imitation learning from suboptimal demonstrations, an offline framework that supervises policy learning with structured language labels instead of scalar signals. Section 4.1 formulates the language-critique objective and performance guarantee, Section 4.2 instantiates a structured language label generator μ_g, and Section 4.3 distills μ_g into a differentiable LLM-Captioner μ_ϕ with the language-critique loss to train BC and diffusion policies.

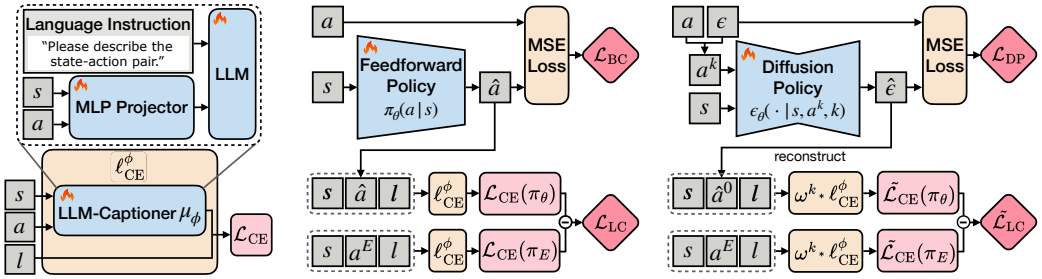
4.1 FROM ACTION IMITATION TO LANGUAGE-CRITIQUE IMITATION

We use language labels to describe state-action features, distinguish expert from suboptimal behaviors, and exploit them through our language-critique imitation objective. Let μ denote a language label function conditioned on a state-action pair. For expert state $s_t \sim d_E^t$, action $a_t^E \sim \pi_E(\cdot \mid s_t)$, and label $l_t^E \sim \mu(\cdot \mid s_t, a_t^E)$, we score a policy action $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$ by the likelihood it assigns to l_t^E:

$$J_{\mathrm{LC}}(\pi_\theta) := \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t\sim d_E^t, a_t^E\sim\pi_E(\cdot|s_t), l_t^E\sim\mu(\cdot|s_t,a_t^E), \hat{a}_t\sim\pi_\theta(\cdot|s_t)} \left[\log \mu(l_t^E \mid s_t, \hat{a}_t)\right]. \tag{1}$$

Mirroring the performance difference formulation in Lemma A.1, we define the language-critique gap as $\Delta_{\mathrm{LC}}(\pi_\theta) := J_{\mathrm{LC}}(\pi_E) - J_{\mathrm{LC}}(\pi_\theta)$. Since $J_{\mathrm{LC}}(\pi_E)$ is independent of π_θ, minimizing $\Delta_{\mathrm{LC}}(\pi_\theta)$ is equivalent to maximizing $J_{\mathrm{LC}}(\pi_\theta)$. Expanding the gap gives

$$\Delta_{\mathrm{LC}}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t\sim d_E^t, a_t^E\sim\pi_E(\cdot|s_t), l_t^E\sim\mu, \hat{a}_t\sim\pi_\theta(\cdot|s_t)} \left[\log \mu(l_t^E|s_t, a_t^E) - \log \mu(l_t^E|s_t, \hat{a}_t)\right]. \tag{2}$$



(a) LLM-Captioner (b) LC-BC (c) LC-DP

Figure 2: **(a) LLM-Captioner architecture and fine-tuning.** LLM-Captioner uses an LLM backbone and MLP projector to map state-action pairs into the transformer's hidden space. We fine-tune μ_ϕ on $\mathcal{D}_G^{\text{lang}}$ to distill μ_g. **(b) Language-critique behavior cloning (LC-BC).** A feedforward policy π_θ is trained with $\mathcal{L}_{\text{BC}}$ and LC-loss $\mathcal{L}_{\text{LC}}$. The LC-loss applies cross-entropy ℓ^ϕ_{CE} to policy and expert actions via a frozen LLM-Captioner. **(c) Language-critique diffusion policy (LC-DP).** A diffusion policy ϵ_θ is trained with $\mathcal{L}_{\text{DP}}$ and reweighted $\tilde{\mathcal{L}}_{\text{LC}}$. A step-dependent reweighting factor ω^k handles varying diffusion steps, and LC-loss applies to clean actions $\hat{a}^0$ reconstructed from predicted noise $\hat{\epsilon}$.

Intuitively, the gap penalizes policy actions that make the expert critique less likely than the expert action. In Section A, we show that under standard linear-feature realizability and a **language-critique sufficiency condition** on μ, minimizing $\Delta_{\text{LC}}(\pi_\theta)$ controls the expert-state performance gap and recovers expert behavior. The condition rules out uninformative labels and requires μ to preserve distinctions in state-action features $\psi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$: if two actions a, a' at the same state differ, $\mu(\cdot \mid s, a)$ and $\mu(\cdot \mid s, a')$ should be distinguishable. We realize this principle with a structured generator μ_g (Section 4.2) and differentiable distillation μ_ϕ for policy optimization (Section 4.3).

4.2 Language Label Generation

To satisfy the language-critique sufficiency in Section 4.1, we define a structured generator μ_g over state-action pairs. Compared with human or free-form LLM/VLM annotations, this design is cheaper and more consistent, but requires state-based information and task-specific generation code. Motivated by reward design principles (Zhang et al., 2025; Ng et al., 1999), each label has three components: task progress `<T>`, action optimality `<A>`, and movement guidance `<M>`. They capture the current stage, whether an action is desirable, and how it should be corrected, while suppressing irrelevant linguistic variation. As shown in Figure 1 and detailed in Section C, μ_g extracts task-relevant heuristics (*e.g.*, object distance and gripper status) from a state-action pair (s_t, a_t), and routes them to three component-specific selectors, each sampling a snippet from a pool. Concatenating the snippets yields the final language label l_t. Applying μ_g to each state-action pair in $\mathcal{D}_E$ and $\mathcal{D}_G$ gives language-labeled datasets:

$$\mathcal{D}_E^{\text{lang}} = \{(s_t, a_t^E, l_t^E) \mid (s_t, a_t^E) \in \mathcal{D}_E,\ l_t^E \sim \mu_g(\cdot \mid s_t, a_t^E)\}, \tag{3}$$

$$\mathcal{D}_G^{\text{lang}} = \{(s_t, a_t, l_t) \mid (s_t, a_t) \in \mathcal{D}_G,\ l_t \sim \mu_g(\cdot \mid s_t, a_t)\}. \tag{4}$$

To reduce reliance on these requirements, we also evaluate VLM-generated labels as an alternative.

4.3 Language-Critique Imitation Learning

To enable end-to-end policy optimization, we instantiate the theoretical gap $\Delta_{\text{LC}}(\pi_\theta)$ with $\mathcal{L}_{\text{LC}}$, a token-level cross-entropy loss computed through the differentiable language label function μ_ϕ, namely the LLM-Captioner. The full method is illustrated in Figure 2.

LLM-Captioner. We distill the generator μ_g into a differentiable LLM-Captioner μ_ϕ that maps state-action pairs to structured language labels. Gradient updates from the language objective in Equation (2) pass through μ_ϕ to π_θ without collapsing language into scalar targets. We provide further details of μ_ϕ in Section D. We fine-tune μ_ϕ on $\mathcal{D}_G^{\text{lang}}$ using token-level cross entropy:

$$\ell^\phi_{\text{CE}}(l_t, s_t, a_t) = -\frac{1}{N_t} \sum_{n=1}^{N_t} \log \mu_\phi(l_{t,n} \mid l_{t,<n}, s_t, a_t) \tag{5}$$

where N_t denotes the tokenized sequence length of l_t. Both the LLM-backbone and the projector are updated according to this objective:

$$\mathcal{L}_{\text{CE}} = \mathbb{E}_{(s_t,a_t,l_t)\sim\mathcal{D}_G^{\text{lang}}}\left[\ell_{\text{CE}}^{\phi}(l_t, s_t, a_t)\right] \tag{6}$$

After fine-tuning, we freeze μ_ϕ and use it for downstream policy optimization. Since the general dataset contains both expert and suboptimal demonstrations, μ_ϕ learns a discriminative mapping from state-action pairs to language. Each component of the predicted language label exposes behavior-quality discrepancies, providing structured supervision that can be exploited during policy learning.

Language-critique loss. We then instantiate the practical language-critique loss through μ_ϕ. For a policy action $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$ and expert action a_t^E, we define

$$\mathcal{L}_{\text{CE}}(\pi_\theta) = \mathbb{E}_{(s_t,l_t)\sim\mathcal{D}_E^{\text{lang}},\hat{a}_t\sim\pi_\theta}\left[\ell_{\text{CE}}^{\phi}(l_t, s_t, \hat{a}_t)\right] \tag{7}$$

$$\mathcal{L}_{\text{CE}}(\pi_E) = \mathbb{E}_{(s_t,a_t^E,l_t)\sim\mathcal{D}_E^{\text{lang}}}\left[\ell_{\text{CE}}^{\phi}(l_t, s_t, a_t^E)\right]. \tag{8}$$

The LC-loss clips the policy loss at the expert loss:

$$\mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E) = \left[\mathcal{L}_{\text{CE}}(\pi_\theta) - \text{sg}(\mathcal{L}_{\text{CE}}(\pi_E))\right]_+, \tag{9}$$

where $[x]_+ = \max(x, 0)$ and $\text{sg}(\cdot)$ denotes stop-gradient. This loss is a differentiable surrogate for $\Delta_{\text{LC}}(\pi_\theta)$: it penalizes policy actions that make the expert language label less likely than the corresponding expert actions, and vanishes once the policy matches the expert under μ_ϕ. Each update transfers the behavior-quality distinctions learned by μ_ϕ from both expert and suboptimal data into the policy. In Section A.5, we show that $\mathcal{L}_{\text{LC}}$ upper-bounds the performance gap between π_θ and π_E up to the captioner distillation residual, extending Theorem A.4 to the practical objective.

Language-critique imitation learning. We apply the LC-loss to behavior cloning (BC) and diffusion policy (DP) (Chi et al., 2023). Training uses the expert-labeled dataset $\mathcal{D}_E^{\text{lang}}$, while the LC-loss transfers the behavior distinctions learned by μ_ϕ from the broader labeled dataset $\mathcal{D}_G^{\text{lang}}$, utilizing suboptimal demonstrations as auxiliary supervision. The LLM-Captioner is used only during training and incurs no test-time cost. First, for BC, a feedforward MLP policy π_θ is updated with

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{(s_t,a_t^E)\sim\mathcal{D}_E,\hat{a}_t\sim\pi_\theta}\left[\left\|a_t^E - \hat{a}_t\right\|_2^2\right]. \tag{10}$$

Language-Critique Behavior Cloning (LC-BC) optimizes Equation (10) with the LC-loss: $\mathcal{L}_{\text{BC}} + \lambda\mathcal{L}_{\text{LC}}$, where λ controls LC-loss strength. For DP, the policy ϵ_θ is a UNet (Ronneberger et al., 2015) noise-prediction model, yielding Language-Critique Diffusion Policy (LC-DP). Its objective is

$$\mathcal{L}_{\text{DP}} = \mathbb{E}_{(s_t,a_t^E)\sim\mathcal{D}_E,\epsilon\sim\mathcal{N}(0,\mathbf{I}),k}\left[\|\epsilon - \hat{\epsilon}\|_2^2\right] \tag{11}$$

with $\hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k)$ and $a_t^k = \sqrt{\bar{\alpha}^k}a_t^E + \sqrt{1-\bar{\alpha}^k}\epsilon$. Since DP predicts noise rather than actions, we apply the LC-loss to one-step reconstructed action $\hat{a}_t^0$ from the sample a_t^k at diffusion step k:

$$\hat{a}_t^0 = \frac{1}{\sqrt{\bar{\alpha}^k}}a_t^k - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}}\hat{\epsilon}, \qquad \hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k). \tag{12}$$

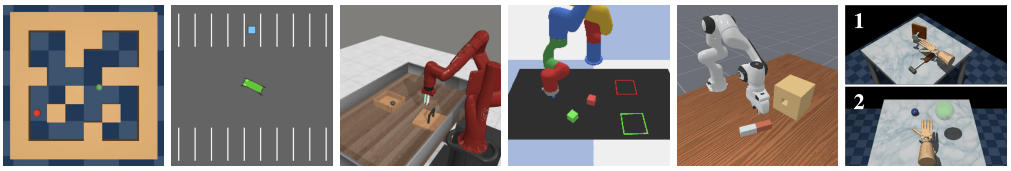
Since reconstructions from larger k are noisier, we weight each diffusion-step samples by $\omega^k = \frac{\bar{\alpha}^k}{1-\bar{\alpha}^k}$ and define the reweighted LC-loss as

$$\tilde{\mathcal{L}}_{\text{CE}}(\pi_\theta) = \mathbb{E}_{(s_t,l_t)\sim\mathcal{D}_E^{\text{lang}},\hat{a}_t^0\sim\epsilon_\theta(s_t),k}\left[\omega^k\ell_{\text{CE}}^{\phi}(l_t, s_t, \hat{a}_t^0)\right] \tag{13}$$

$$\tilde{\mathcal{L}}_{\text{CE}}(\pi_E) = \mathbb{E}_{(s_t,a_t^E,l_t)\sim\mathcal{D}_E^{\text{lang}},k}\left[\omega^k\ell_{\text{CE}}^{\phi}(l_t, s_t, a_t^E)\right] \tag{14}$$

$$\tilde{\mathcal{L}}_{\text{LC}}(\pi_\theta, \pi_E) = \left[\tilde{\mathcal{L}}_{\text{CE}}(\pi_\theta) - \text{sg}(\tilde{\mathcal{L}}_{\text{CE}}(\pi_E))\right]_+ \tag{15}$$

The LC-DP objective, $\mathcal{L}_{\text{DP}} + \lambda\tilde{\mathcal{L}}_{\text{LC}}$, encourages reconstructed actions to stay close to expert actions while inducing expert-consistent language labels under μ_ϕ's learned expert-suboptimal distinctions. We provide LC-DP details in Section E and full algorithms for both methods in Section B.



(a) MAZE (b) PARKING (c) BOX-CLOSE (d) BLOCKPUSH (e) PEGINSERT (f) ADROIT

Figure 3: **Environments & tasks. (a) MAZE**: Navigate a dot to a red goal. **(b) PARKING**: Park a car into the spot. **(c) BOX-CLOSE**: Place a lid onto a box. **(d) BLOCKPUSH**: Multi-stage multimodal task pushing two blocks to their targets. **(e) PEGINSERT**: Precisely insert a peg into a side hole. **(f) HAMMER & RELOCATE**: High-dimensional control to hammer a nail and relocate a ball.

5 EXPERIMENTS

5.1 ENVIRONMENTS

We evaluate on seven tasks spanning navigation, driving, and manipulation, with varying horizon length, action precision, multimodality, and state-action dimensionality. Figure 3 shows the experimental tasks. For each task, we collect datasets $\mathcal{D}_E$ and $\mathcal{D}_G$. See Section F and G for details.

- **MAZE**: A navigation task (Fu et al., 2020) where an agent applies x/y forces to traverse a maze toward a sampled goal, observing its position, velocity, and the goal location.
- **PARKING**: A driving task (Leurent, 2018) where an agent parks in a target spot via throttle and steering. We use image observations rather than the original environment's state-based vectors.
- **BOX-CLOSE**: A manipulation task (Yu et al., 2019) where a Sawyer robot places a lid on a box.
- **BLOCKPUSH**: A multistage task (Florence et al., 2021) requiring the robot to push two blocks into two target zones. Multiple valid strategies make this task inherently multimodal.
- **PEGINSERT**: A high-precision task (Tao et al., 2025) requiring joint-level control to insert a peg into a tightly matched side hole, demanding fine-grained corrective feedback.
- **HAMMER** and **RELOCATE**: High-dimensional Adroit dexterous hand tasks (Rajeswaran et al., 2018) involving hammering a nail and relocating a ball.

For BOX-CLOSE, evaluation samples the lid's initial position from an expanded range, except in the degraded-data experiments (Table 2), which retain the original range; Section F details both settings.

5.2 TASK PERFORMANCE

We evaluate whether additional language label supervision can improve imitation learning from a small expert dataset $\mathcal{D}_E$ and a broader general dataset $\mathcal{D}_G$. We compare LC-BC and LC-DP against representative IL from suboptimal demonstrations. To examine whether expressing behavioral feedback in language offers benefits beyond scalar or categorical supervision, we also compare with variants that convert the same language label into scalar scores or structured categorical targets.

- **BC** (Pomerleau, 1988): Behavior cloning that imitates expert action likelihood.
- **DWBC** (Xu et al., 2022): A discriminator-weighted variant of BC that learns a discriminator to distinguish expert from non-expert data and reweights the imitation loss accordingly.
- **DemoDICE** (Kim et al., 2022): An offline imitation method that matches stationary distributions with a KL regularizer toward general data, favoring expert-like state-action visitations.
- **ILID** (Yue et al., 2024): An offline imitation learning method that selects demonstrations whose resulting states lie within the expert manifold, then performs BC on the expert and selected data.
- **DP** (Chi et al., 2023): A diffusion-policy baseline that models the action distribution with a conditional denoising process, trained on the expert dataset only.
- **LPB-Offline**: An offline adaptation of LPB (Sun & Song, 2025), which leverages a dynamic model trained on general data to guide the DP near demonstrated behaviors.
- **LC-BC-scalar** and **LC-DP-scalar**: Scalar-critique variants reduce language label to a single scalar under an LC-loss-style objective. See Section H.1 for details.
- **LC-BC-classifier and LC-DP-classifier**: Class-critique variants that map language label to categorical `<T>`, `<A>`, `<M>` targets, preserving their structure. See Section H.2 for details.

Table 1: **Main results.** We report the average success rate across seven tasks, using five training seeds and 50 environment rollouts per seed. The last column reports the mean success rate averaged over all tasks. We pair feedforward policies with BC, DWBC, DemoDICE, ILID, LC-BC-scalar, LC-BC-classifier, and LC-BC and diffusion policies with DP, LPB-Offline, LC-DP-scalar, LC-DP-classifier, and LC-DP. Best results are in **bold**; second-best are underlined. LC-BC and LC-DP achieve competitive performance relative to all baselines across environments. (% omitted).

METHOD	MAZE	PARKING	BOX-CLOSE	BLOCKPUSH	PEGINSERT	HAMMER	RELOCATE	AVERAGE
BC	79.6 ± 4.8	41.6 ± 7.1	66.4 ± 14.6	34.4 ± 3.3	66.4 ± 5.4	96.0 ± 3.7	70.8 ± 8.3	65.0
DWBC	**85.2** ± 5.8	25.6 ± 0.9	70.4 ± 8.3	26.0 ± 4.2	60.8 ± 1.8	88.0 ± 9.9	48.0 ± 6.6	57.7
DEMODICE	78.0 ± 3.2	37.6 ± 5.4	**88.4** ± 5.0	3.6 ± 2.6	**67.2** ± 6.1	96.8 ± 7.2	65.2 ± 8.1	62.4
ILID	82.4 ± 3.6	40.8 ± 5.8	75.2 ± 15.3	28.4 ± 4.1	59.6 ± 2.6	90.4 ± 9.1	**80.8** ± 6.9	65.4
LC-BC-SCALAR	79.6 ± 4.8	42.0 ± 6.5	74.8 ± 13.8	38.0 ± 4.7	63.6 ± 3.3	90.0 ± 6.9	65.6 ± 11.2	64.8
LC-BC-CLASSIFIER	78.0 ± 3.2	28.8 ± 2.3	87.2 ± 11.2	40.4 ± 6.1	65.2 ± 3.6	75.6 ± 9.8	66.8 ± 12.6	63.1
LC-BC (OURS)	81.6 ± 6.1	**44.8** ± 3.9	87.6 ± 6.5	**47.2** ± 4.1	**67.2** ± 5.4	**99.2** ± 1.8	71.2 ± 5.9	**71.3**
DP	**94.4** ± 2.2	37.2 ± 1.8	72.4 ± 4.3	83.6 ± 4.8	53.6 ± 1.7	79.2 ± 3.0	67.2 ± 24.2	69.7
LPB-OFFLINE	93.6 ± 2.2	38.8 ± 1.1	67.2 ± 7.8	78.0 ± 4.9	51.6 ± 5.0	74.4 ± 3.6	59.6 ± 27.2	66.2
LC-DP-SCALAR	94.0 ± 2.0	38.8 ± 1.1	67.2 ± 3.6	85.2 ± 1.1	53.2 ± 2.7	76.0 ± 1.4	66.8 ± 21.8	68.7
LC-DP-CLASSIFIER	92.4 ± 2.2	**39.2** ± 3.9	67.2 ± 4.6	82.8 ± 2.3	52.8 ± 1.8	77.6 ± 0.9	68.0 ± 24.8	68.6
LC-DP (OURS)	**94.4** ± 1.7	37.6 ± 3.8	**74.0** ± 5.7	**88.0** ± 1.4	**56.8** ± 2.7	**80.0** ± 3.2	**77.6** ± 4.1	**72.6**

As shown in Table 1, LC-BC achieves higher mean success rates than BC on all seven tasks, while LC-DP improves over DP on six tasks and ties on MAZE. Against the second-best baseline, average performance rises from 65.4% to 71.3% for feedforward policies and from 69.7% to 72.6% for diffusion policies, leading their respective classes on average. For feedforward policies, the largest improvements over BC occur on BOX-CLOSE (66.4% to 87.6%) and BLOCKPUSH (34.4% to 47.2%). LC-BC is competitive with suboptimal IL methods, although some baselines remain strongest on some tasks: DWBC on MAZE, DemoDICE on BOX-CLOSE, and ILID on RELOCATE. For diffusion policies, the largest improvements over DP are on RELOCATE (67.2% to 77.6%) and BLOCKPUSH (83.6% to 88.0%). LC-DP also exceeds LPB-Offline on six of the seven tasks, except for PARKING.

The method variants further support the utility of language-critique supervision. LC-BC outperforms LC-BC-scalar on all tasks, while LC-DP exceeds LC-DP-scalar on six tasks, excluding PARKING. Compared with categorical supervision over the same <T>, <A>, and <M> components, LC-BC achieves higher mean success rates on all tasks, and LC-DP outperforms LC-DP-classifier on six tasks. These comparisons suggest that representing structured behavioral supervision through language can improve policy learning, although the benefit varies across tasks and policy architecture. Overall, the results support language-critique supervision as an effective component of imitation learning from suboptimal demonstrations. Qualitative rollouts comparing BC and LC-BC, together with the corresponding language label and LC-loss, are provided in Section J.

5.3 LEARNING FROM DEGRADED DEMONSTRATION DATASETS

Collecting high-quality expert demonstrations is labor-intensive (*e.g.*, real-world teleoperation), and even curated data can include suboptimal actions. We therefore evaluate whether language critiques remain effective when expert data are imperfect or scarce. As shown in Table 2, we study two degraded regimes: (1) a single mixed-quality dataset with no separate expert set, and (2) a shrinking expert dataset. These experiments test whether the additional labeling and training cost of language critiques (Section G.4) is worthwhile when obtaining more or better expert data is expensive.

Learning from mixed-quality demonstrations. Expert data are often noisy (*e.g.*, teleoperated demonstrations include hesitations and corrections) so a reliable expert partition may not exist. Thus, we take a single offline dataset $\mathcal{D}_G$ with language label, and we form a pseudo-expert set by selecting sequences of N consecutive transitions whose <A> component is positive, $N \in \{2, 3, 5, 10\}$; larger N filters more strictly, and the fraction of selected samples filtered from the true expert set (*i.e.*, expert accuracy) rises from 32.9% to 86.8%. All methods train on identical partitions. LC-BC and LC-DP obtain the highest mean success rate within their policy families at every N, and the largest margins occur with the noisier pseudo-expert sets: at $N = 3$, LC-BC reaches 78.0% vs. 64.7% for the next best method, and at $N = 2$, LC-DP reaches 70.0% vs. 58.7%. Both methods also outperform their variants at every threshold, attributing the gain to language critiques rather than data filtering.

Learning with fewer expert demonstrations. Expert demonstrations are the most expensive part of the dataset to collect, whereas suboptimal data are cheaper. We therefore subsample the expert dataset to a fraction $\rho \in \{2\%, 5\%, 10\%, 100\%\}$ of its size while leaving the remaining general data

Table 2: **Learning from degraded demonstration datasets.** We report success rates on BOX-CLOSE over 50 rollouts per seed (% omitted). **(Left) Mixed-quality demonstrations.** We relax the clean expert dataset assumption and use language-based heuristic filtering to extract varying levels of expert-quality data from the general dataset, at four quality levels *N*={*2,3,5,10*}, over 3 seeds. Higher N indicates stricter filtering; expert accuracy (in parentheses) measures the fraction of selected samples drawn from the ground-truth expert dataset, as a proxy for data quality. **(Right) Data efficiency.** We subsample the expert dataset to fractions ρ of its original size, keeping the quality distribution fixed, over 5 seeds. Best results are in **bold**; second-best are underlined.

METHOD	MIXED-QUALITY (N)				DATA EFFICIENCY (ρ)			
	2 (32.9%)	*3 (48.9%)*	*5 (68.4%)*	*10 (86.8%)*	*2%*	*5%*	*10%*	*100%*
BC	50.0 ± 4.0	55.3 ± 11.0	87.3 ± 4.6	84.7 ± 2.3	48.8 ± 11.9	77.6 ± 11.9	73.6 ± 14.3	94.0 ± 3.7
DWBC	34.7 ± 2.3	52.7 ± 7.0	71.3 ± 3.1	80.0 ± 2.0	52.0 ± 14.4	61.6 ± 7.1	66.8 ± 5.2	92.4 ± 3.6
DEMODICE	65.3 ± 15.5	61.3 ± 5.0	87.3 ± 5.0	84.0 ± 5.3	72.8 ± 5.0	71.6 ± 3.8	80.4 ± 4.8	94.8 ± 2.3
ILID	68.0 ± 7.2	63.3 ± 6.4	50.7 ± 43.9	88.0 ± 2.0	64.4 ± 4.3	83.6 ± 3.6	90.8 ± 2.3	95.6 ± 3.3
LC-BC-SCALAR	58.7 ± 5.0	56.7 ± 13.3	82.0 ± 8.0	85.3 ± 5.0	63.2 ± 8.9	82.4 ± 5.9	82.4 ± 3.3	96.4 ± 2.6
LC-BC-CLASSIFIER	64.7 ± 4.2	64.7 ± 5.8	80.0 ± 7.2	87.3 ± 6.4	**74.4** ± 6.4	85.2 ± 3.9	81.6 ± 4.6	96.4 ± 2.2
LC-BC (OURS)	**71.3** ± 8.3	**78.0** ± 3.5	**88.7** ± 7.0	**94.0** ± 2.0	71.6 ± 5.7	**86.4** ± 5.0	**93.2** ± 4.4	**99.2** ± 1.8
DP	56.0 ± 4.0	91.3 ± 1.2	92.0 ± 2.0	95.3 ± 2.3	71.2 ± 1.8	85.6 ± 3.3	92.8 ± 1.1	94.4 ± 0.9
LPB-OFFLINE	54.0 ± 5.3	82.0 ± 4.0	86.7 ± 6.1	91.3 ± 4.6	72.0 ± 6.2	79.6 ± 9.7	92.4 ± 2.2	94.0 ± 0.0
LC-DP-SCALAR	58.0 ± 3.5	86.0 ± 2.0	91.3 ± 1.2	95.3 ± 1.2	70.4 ± 2.2	85.6 ± 2.6	**95.2** ± 1.1	94.0 ± 0.0
LC-DP-CLASSIFIER	58.7 ± 3.1	88.7 ± 1.2	92.7 ± 2.3	96.0 ± 2.0	73.2 ± 3.6	85.6 ± 1.7	94.0 ± 2.0	94.0 ± 0.0
LC-DP (OURS)	**70.0** ± 2.0	**92.0** ± 2.0	**94.0** ± 0.0	**96.7** ± 1.2	**77.6** ± 7.1	**92.4** ± 1.7	94.4 ± 1.7	**98.8** ± 1.1

and its language label supervision untouched. Language supervision compensates for the expert data shortage: LC-BC at ρ=10% (93.2%) matches BC trained on the full expert set (94.0%), and LC-DP at ρ=5% (92.4%) is within two points of DP at full data (94.4%). At every ρ, both methods improve over their base policies, and over the scalar and categorical counterparts in all but two cells (LC-BC-classifier at ρ=2% and LC-DP-scalar at ρ=10%, both within one standard deviation). Together, these results show that language critiques improve robustness to imperfect expert data and achieve performance comparable to full-data BC and DP with far fewer expert demonstrations, suggesting a useful trade-off between additional computation and expert data collection.

5.4 LANGUAGE LABEL ABLATION STUDIES

We conduct a series of ablation studies to identify the contribution of each proposed component (<T><A><M>), compare our language label generator μ_g against a general-purpose VLM labeling pipeline, and isolate which properties of natural language carry the performance gain.

Compositional ablation of language labels. In Table 3a, we ablate the three components. Full language label is the only configuration that improves over DP and performs within the across-seed variability of the best LC-BC subset (<T>+<M>: 92.4% vs. 87.6%). All the components are complementary rather than redundant.

VLM-generated language labels. μ_g is task-specific: it is lightweight and easy to validate, but must be redesigned for each new task. In Table 3b, we replace it with labels from a general-purpose VLM (`o4-mini`). With a 360M-parameter μ_ϕ, VLM labels match μ_g for LC-DP (74.8% vs. 74.0%) and reach 74.0% for LC-BC, well above BC but below μ_g labels. We observe that VLM-generated labels are more diverse and verbose (see Table 9), potentially making them harder for μ_ϕ to model. This motivates either using a larger backbone or post-processing the labels into a concise and structured form to improve policy learning. Scaling from 135M to 360M improves performance with VLM labels (+4.0%/+4.4%) but reduces it with μ_g labels (to 70.4%/70.8%), suggesting that additional capacity benefits richer labels more than low-entropy templates. In contrast, making VLM labels concise leaves performance unchanged, while rewriting them into the μ_g template lowers LC-BC performance to 64.4% but raises LC-DP performance to 72.8%. Human critiques also improve over BC and DP in the 2% expert-data setting, although they underperform μ_g labels, with a smaller gap for LC-DP. See Section I.3 for the annotation protocol and result details. These mixed effects motivate a closer examination of which properties of language critiques matter for policy learning.

What properties of language matter? To investigate why and how language benefits policy learning, we examine four properties on BOX-CLOSE: μ_ϕ pretraining, lexical semantics, compactness, and accuracy (Table 3c). First, labels must align with the state-action pair: shuffling them preserves format and vocabulary but reduces success from 87.6%/74.0% to 69.2%/67.6%. This suggests

Table 3: **Language label ablation on BOX-CLOSE.** Success rates averaged over 50 rollouts and 5 seeds (% omitted); best **bold**, second underlined. **(a) Structure.** Ablating the three language label components (<T><A><M>). **(b) Generator.** Replacing μ_g with OpenAI `o4-mini` language labels, raw or post-processed (*concise*: shortened; μ_g*-style*: rewritten into the μ_g template), across μ_ϕ sizes. **(c) Label form.** Varying four properties: *Pre.*: pretrained μ_ϕ; *Sem.*: tokens carry lexical semantics (○: partially, via subword names); *Cmp.*: compact, variable-length labels; *Acc.*: state-action-aligned labels. Token formats replace μ_g with categorical tokens (opaque, named, or variable-length named). μ_g-shuffled permutes labels within $\mathcal{D}_G^{\text{lang}}$; μ_g-verbose rewrites them using human verbal language.

(a) Structure

<T>	<A>	<M>	LC-BC	LC-DP
×	×	✓	74.8 ± 5.8	66.0 ± 4.9
×	✓	×	79.6 ± 11.4	68.0 ± 5.8
✓	×	×	58.8 ± 7.9	70.8 ± 6.4
✓	✓	×	67.6 ± 9.5	72.0 ± 10.2
✓	×	✓	**92.4** ± 2.6	65.6 ± 3.6
×	✓	✓	85.6 ± 7.7	68.0 ± 5.8
✓	✓	✓	87.6 ± 6.5	**74.0** ± 5.7

(b) Generator

Language label	μ_ϕ	LC-BC	LC-DP
`o4-mini`	135M	70.0 ± 12.4	70.4 ± 4.8
`o4-mini` concise	135M	70.4 ± 5.9	70.0 ± 6.8
`o4-mini` μ_g-style	135M	64.4 ± 12.8	72.8 ± 6.4
`o4-mini`	360M	74.0 ± 9.5	**74.8** ± 9.7
μ_g	135M	**87.6** ± 6.5	74.0 ± 5.7
μ_g	360M	70.4 ± 9.2	70.8 ± 3.6

(c) Label form

Supervision	Pre.	Sem.	Cmp.	Acc.	LC-BC	LC-DP
None (BC / DP)	–	–	–	–	66.4 ± 14.6	72.4 ± 4.3
Classifier heads	–	×	×	✓	87.2 ± 11.2	67.2 ± 4.6
Opaque tokens	✓	×	×	✓	84.4 ± 9.0	66.0 ± 5.8
Named tokens	✓	○	×	✓	86.0 ± 3.7	67.6 ± 3.3
Named tokens, variable length	✓	○	✓	✓	88.0 ± 5.8	70.0 ± 5.5
μ_g, μ_ϕ from scratch	×	✓	✓	✓	**88.8** ± 5.0	63.2 ± 1.8
μ_g-verbose	✓	✓	×	✓	74.0 ± 7.6	68.0 ± 5.8
μ_g-shuffled	✓	✓	✓	×	69.2 ± 12.7	67.6 ± 4.3
μ_g (ours)	✓	✓	✓	✓	87.6 ± 6.5	**74.0** ± 5.7

that improving label form alone may be insufficient when the underlying critiques are inaccurate. Second, accurate categorical targets recover most of the gain for LC-BC on this task (87.2% versus 87.6%), and training μ_ϕ from scratch achieves 88.8%, indicating that this gain does not require pretrained language representations. However, this finding does not extend uniformly across tasks: in Table 1, LC-BC exceeds LC-BC-classifier on all seven tasks, with larger gaps on PARKING and HAMMER, and average success rates of 71.3% versus 63.1%. For LC-DP, pretrained natural-language supervision achieves 74.0% on BOX-CLOSE, compared with 67.2% for categorical classifier heads and 63.2% without pretraining, suggesting a greater benefit from pretrained language representations in this setting. Replacing opaque tokens with descriptively named categorical tokens (samples in Table 9) improves both policies by 1.6%, suggesting that lexical semantics contributes to performance. Compact, variable-length named tokens improve success by 2.0%/2.4% over fixed-length variants, while verbose labels hurt both policies, suggesting that concise labels better focus learning on relevant content. More broadly, LC-DP is less sensitive to changes in label source but relies more on μ_ϕ pretraining, highlighting distinct supervision needs across policy architectures. Together, these results suggest that accurate, compact supervision is important, while the additional benefits of language representation and pretraining vary across tasks and policy architectures. See Section I.2 for details.

6 DISCUSSION

We propose language-critique imitation learning, an offline framework that learns from expert and suboptimal demonstrations through natural-language critiques of task progress, action optimality, and movement guidance. Across seven tasks, LC-BC and LC-DP improve average success rates over prior baselines (Section 5.2), with additional benefits under scarce or noisy expert data (Section 5.3). In Section 5.4, our ablations suggest that accurate, compact, structured language accounts for much of the feedforward-policy gain, and diffusion policies benefit more from pretrained language representations. VLM labels also perform comparably to μ_g labels for LC-DP, reducing reliance on task-specific label generators. These findings motivate extending language-critique supervision to vision-language-action models (VLAs) with diffusion-based action heads (Li et al., 2024; Black et al., 2024; 2025) and real-world settings. Future work could also develop reliable critique models that transfer across tasks, enabling broader reuse of behavioral supervision. Our human-label experiment further demonstrates the feasibility of combining sparse human critiques with generated labels, improving over BC and DP in the evaluated low-data setting (Section I.3). This motivates extending the framework toward human-in-the-loop policy learning.

BIBLIOGRAPHY

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Neural Information Processing Systems*, 2017.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, 2025.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019a.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, 2019b.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In *Empirical Methods in Natural Language Processing*, 2024.

Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. Learning from natural language feedback. *Transactions on machine learning research*, 2024a.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on Robot Learning*, 2020.

Shang-Fu Chen, Hsiang-Chun Wang, Ming-Hao Hsu, Chun-Mao Lai, and Shao-Hua Sun. Diffusion model-augmented behavioral cloning. In *International Conference on Machine Learning*, 2024b.

Shang-Fu Chen, Co Yong, and Shao-Hua Sun. Restoring noisy demonstration for imitation learning with diffusion models. *IEEE Transactions on Neural Networks and Learning Systems*, 37:401–413, 2026.

Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. LLF-bench: Benchmark for interactive learning from language feedback. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning. In *International Conference on Robotics and Automation*, 2025.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, 2023.

Xidong Feng, Bo Liu, Ziyu Wan, Haotian Fu, Girish A. Koushik, Zhiyuan Hu, Mengyue Yang, Ying Wen, and Jun Wang. Natural language reinforcement learning. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, 2021.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Neural Information Processing Systems*, 2020.

Ling Huan and Sanja Fidler. Teaching machines to describe images with natural language feedback. In *Neural Information Processing Systems*, 2017.

Bo-Ruei Huang, Chun-Kai Yang, Chun-Mao Lai, Dai-Jie Wu, and Shao-Hua Sun. Diffusion imitation from observation. In *Neural Information Processing Systems*, 2024.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50, 2017.

Pingcheng Jian, Xiao Wei, Yanbaihui Liu, Samuel A. Moore, Michael M. Zavlanos, and Boyuan Chen. LAPP: Large language model feedback for preference-driven reinforcement learning. *Transactions on Machine Learning Research*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. *Mathematics of Operations Research*, 48, 2023.

Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. In *International Conference on Robotics and Automation*, 2025.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Neural Information Processing Systems*, 2023.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*, 2020.

Chun-Mao Lai, Hsiang-Chun Wang, Ping-Chun Hsieh, Yu-Chiang Frank Wang, Min-Hung Chen, and Shao-Hua Sun. Diffusion-reward adversarial imitation learning. In *Neural Information Processing Systems*, 2024.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback descent: Open-ended text optimization via pairwise comparison. *arXiv preprint arXiv:2511.07919*, 2025.

Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In *Neural Information Processing Systems*, 2021.

Edouard Leurent. An environment for autonomous driving decision-making, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Yu-An Lin, Chen-Tao Lee, Chih-Han Yang, Guan-Ting Liu, and Shao-Hua Sun. Hierarchical programmatic option framework. In *Neural Information Processing Systems*, 2024.

Guan-Ting Liu, En-Pei Hu, Pu-Jen Cheng, Hung-Yi Lee, and Shao-Hua Sun. Hierarchical programmatic reinforcement learning via learning to compose programs. In *International Conference on Machine Learning*, 2023.

Max Liu, Chan-Hung Yu, Wei-Hsu Lee, Cheng-Wei Hung, Yen-Chun Chen, and Shao-Hua Sun. Synthesizing programmatic reinforcement learning policies with large language model guided search. In *International Conference on Learning Representations*, 2025.

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *Robotics: Science and Systems*, 2021.

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Gti: Learning to generalize across long-horizon tasks from human demonstrations. In *Robotics: Science and Systems*, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7, 2018.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, 1988.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

Stéphane Ross and J Andrew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

Stéphane Ross, Geoffrey Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021.

Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems*, 1996.

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

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Neural Information Processing Systems*, 2022.

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. In *Robotics: Science and Systems*, 2024.

Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. In *Neural Information Processing Systems*, 2020.

Theodore Sumers, Kenneth Marino, Arun Ahuja, Rob Fergus, and Ishita Dasgupta. Distilling internet-scale vision-language models into embodied agents. In *International Conference on Machine Learning*, 2023.

Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, and Joseph J. Lim. Neural program synthesis from diverse demonstration videos. In *International Conference on Machine Learning*, 2018.

Shao-Hua Sun, Te-Lin Wu, and Joseph J. Lim. Program guided agent. In *International Conference on Learning Representations*, 2020.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. In *Neural Information Processing Systems*, 2025.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh N, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: GPU parallelized robot simulation and rendering for generalizable embodied AI. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

Belen Martin Urcelay, Andreas Krause, and Giorgia Ramponi. From words to rewards: Leveraging natural language for reinforcement learning. In *The Exploration in AI Today Workshop at ICML 2025*, 2025.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. In *International Conference on Machine Learning*, 2024a.

Yunke Wang, Chang Xu, Bo Du, and Honglak Lee. Learning to weight imperfect demonstrations. In *International Conference on Machine Learning*, 2021.

Yunke Wang, Minjing Dong, Yukun Zhao, Bo Du, and Chang Xu. Imitation learning from purified demonstrations. In *International Conference on Machine Learning*, 2024b.

Joe Watson, Sandy Huang, and Nicolas Heess. Coherent soft imitation learning. In *Neural Information Processing Systems*, 2023.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, 2019.

Jiajun Xi, Yinong He, Jianing Yang, Yinpei Dai, and Joyce Chai. Teaching embodied reinforcement learning agents: Informativeness and diversity of language use. In *Empirical Methods in Natural Language Processing*, 2024.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, 2022.

Wanqiao Xu, Allen Nie, Ruijie Zheng, Aditya Modi, Adith Swaminathan, and Ching-An Cheng. Provably learning from language feedback. In *The Exploration in AI Today Workshop at ICML 2025*, 2025.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *eprint arXiv: 2407.10671*, 2024.

Mengjiao Yang, Sergey Levine, and Ofir Nachum. TRAIL: Near-optimal imitation learning with suboptimal data. In *International Conference on Learning Representations*, 2022.

Lantao Yu, Tianhe Yu, Jiaming Song, Willie Neiswanger, and Stefano Ermon. Offline imitation learning with suboptimal demonstrations via relaxed distribution matching. In *Association for the Advancement of Artificial Intelligence*, 2023a.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning*, 2023b.

Sheng Yue, Jiani Liu, Xingyuan Hua, Ju Ren, Sen Lin, Junshan Zhang, and Yaoxue Zhang. How to leverage diverse demonstrations in offline imitation learning. In *International Conference on Machine Learning*, 2024.

Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. When demonstrations meet generative world models: A maximum likelihood framework for offline inverse reinforcement learning. In *Neural Information Processing Systems*, 2023.

Jesse Zhang, Karl Pertsch Jiahui Zhang, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *Conference on Robot Learning*, 2023.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. In *Conference on Robot Learning*, 2025.

Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. Confidence-aware imitation learning from demonstrations with varying optimality. In *Neural Information Processing Systems*, 2021.

APPENDIX

Table of Contents

A Theoretical Analysis **16**
- A.1 Expert-state language-critique objective . . . 16
- A.2 Language-critique gap as distribution matching . . . 16
- A.3 Language-critique gap bounds feature gap . . . 17
- A.4 Proof of the performance difference bound . . . 17
- A.5 Connection to the practical LC-loss . . . 18
- A.6 Empirical diagnostics of the language-critique assumptions . . . 20

B Algorithm **27**

C Language label details **29**

D LLM-Captioner details **33**

E LC-DP details **34**
- E.1 Diffusion model: forward and reverse process . . . 34
- E.2 Diffusion model: training algorithm . . . 35
- E.3 Reweighting factor derivation . . . 36

F Environment and tasks details **39**

G Training details **40**
- G.1 Model architectures . . . 40
- G.2 Hyperparameters . . . 43
- G.3 Computational resources . . . 44
- G.4 Baseline runtimes . . . 44

H Baseline details **44**
- H.1 LC-BC-scalar and LC-DP-scalar . . . 45
- H.2 LC-BC-classifier and LC-DP-classifier . . . 47

I Additional experimental results **49**
- I.1 VLM-generated language labels . . . 49
- I.2 What properties of language matter? . . . 49
- I.3 Human-generated language labels . . . 51
- I.4 LLM-Captioner ablation . . . 51
- I.5 LLM-Captioner training data analysis . . . 52
- I.6 Scalar critiques derived directly from task features . . . 53
- I.7 λ sensitivity . . . 54

J Qualitative results of LC-loss and LLM-Captioner **54**

K Ethics Statement **56**

A THEORETICAL ANALYSIS

In this section, we provide theoretical justification for the language-critique objective introduced in Section 4.1. We show that the expert-state language-critique gap naturally reflects the sub-optimality gap. Specifically, under mild assumptions, minimizing the language-critique gap $\Delta_{\mathrm{LC}}(\pi_\theta)$ decreases an upper bound on the gap $J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta)$. This provides a theoretical basis for using language label as an auxiliary imitation signal beyond action matching.

A.1 EXPERT-STATE LANGUAGE-CRITIQUE OBJECTIVE

Our method uses both expert and suboptimal demonstrations, where suboptimal data may contain useful near-optimal behaviors as well as noisy rollouts that should be discounted. To identify useful behaviors, we need a unified measure of policy quality for both π_θ and the behavior policies underlying the demonstrations.

Formally, let $d_E^t(s) \coloneqq \Pr(s_t = s \mid \pi_E)$ be the expert state distribution at timestep t, and let $A^\pi_{r,t}(s,a) \coloneqq Q^\pi_{r,t}(s,a) - V^\pi_{r,t}(s)$ be the time-dependent advantage, where $Q^\pi_{r,t}$ and $V^\pi_{r,t}$ are the action- and state-value functions of π under reward r over the remaining horizon.

We build on the following performance difference lemma:

Lemma A.1 (Performance difference lemma.)**.** *Following Kakade & Langford (2002), the return difference between the expert policy and an arbitrary policy π, evaluated under the task reward r^*, admits the following decomposition:*

$$J_{r^*}(\pi_E) - J_{r^*}(\pi) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t,\, a_t^E \sim \pi_E(\cdot|s_t)} \left[A^\pi_{r^*,t}(s_t, a_t^E) \right]. \tag{16}$$

This provides such a measure and motivates the objective introduced in Section 4.1, which we derive step by step in the remainder of this section.

For an expert state $s_t \sim d_E^t$, let $a_t^E \sim \pi_E(\cdot \mid s_t)$ denote the expert action. Given a language label function μ, the corresponding expert language label is sampled as

$$l_t^E \sim \mu(\cdot \mid s_t, a_t^E).$$

For a learned policy π_θ, we evaluate the policy action $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$ by the likelihood with which it explains the same expert language label. We define the expert-state language-critique objective as

$$J_{\mathrm{LC}}(\pi_\theta) := \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E(\cdot \mid s_t), l_t^E \sim \mu(\cdot \mid s_t, a_t^E), \hat{a}_t \sim \pi_\theta(\cdot \mid s_t)} \left[\log \mu(l_t^E \mid s_t, \hat{a}_t) \right]. \tag{17}$$

where T is the horizon of each episode. Similarly, the expert policy obtains

$$J_{\mathrm{LC}}(\pi_E) := \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E(\cdot \mid s_t), l_t^E \sim \mu(\cdot \mid s_t, a_t^E)} \left[\log \mu(l_t^E \mid s_t, a_t^E) \right]. \tag{18}$$

The language-critique gap is defined as

$$\Delta_{\mathrm{LC}}(\pi_\theta) \coloneqq J_{\mathrm{LC}}(\pi_E) - J_{\mathrm{LC}}(\pi_\theta) \tag{19}$$

Since $J_{\mathrm{LC}}(\pi_E)$ is independent of the learned policy, minimizing $\Delta_{\mathrm{LC}}(\pi_\theta)$ is equivalent to maximizing $J_{\mathrm{LC}}(\pi_\theta)$. Expanding $\Delta_{\mathrm{LC}}(\pi_\theta)$, we obtain

$$\Delta_{\mathrm{LC}}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E, l_t^E \sim \mu(\cdot|s_t, a_t^E), \hat{a}_t \sim \pi_\theta} \left[\log \mu(l_t^E | s_t, a_t^E) - \log \mu(l_t^E | s_t, \hat{a}_t) \right]. \tag{20}$$

A.2 LANGUAGE-CRITIQUE GAP AS DISTRIBUTION MATCHING

First, we show that the language-critique gap $\Delta_{\mathrm{LC}}(\pi_\theta)$ is exactly a KL divergence between the expert language label distribution and the policy-yielded language label distribution. Following the definition of KL divergence, we have:

$$D_{\mathrm{KL}}(\mu(\cdot \mid s_t, a_t^E) || \mu(\cdot \mid s_t, \hat{a}_t)) = \mathbb{E}_{l_t^E \sim \mu(\cdot \mid s_t, a_t^E)} \left[\log \frac{\mu(l_t^E \mid s_t, a_t^E)}{\mu(l_t^E \mid s_t, \hat{a}_t)} \right].$$

Therefore,

$$\Delta_{\mathrm{LC}}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E(\cdot \mid s_t), \hat{a}_t \sim \pi_\theta(\cdot \mid s_t)} \left[D_{\mathrm{KL}}(\mu(\cdot \mid s_t, a_t^E) || \mu(\cdot \mid s_t, \hat{a}_t)) \right]. \tag{21}$$

Therefore, minimizing $\Delta_{\mathrm{LC}}(\pi_\theta)$ encourages the policy-induced language distribution $\mu(\cdot \mid s_t, \hat{a}_t)$ to match the expert-induced language distribution $\mu(\cdot \mid s_t, a_t^E)$ on expert states $s_t \sim d_E^t$.

A.3 LANGUAGE-CRITIQUE GAP BOUNDS FEATURE GAP

We now connect language distribution matching to feature matching. We assume the following:

Assumption A.2 (Linear realizability)**.** For any policy π, the action-value function under the optimal task reward r^* is linear in a reward-relevant feature representation: there exist weights $w_t^\pi \in \mathbb{R}^d$ for each t such that $Q^\pi_{r^*,t}(s,a) = {w_t^\pi}^\top \psi(s,a)$, where $\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. We assume $w_{\max} \coloneqq \sup_{\pi,t} \|w_t^\pi\|_2$ is finite.

Notably, this assumption is standard in the literature of linear MDPs (Jin et al., 2023) and successor features (Barreto et al., 2017), and underlies feature-expectation matching in apprenticeship learning (Abbeel & Ng, 2004).

Assumption A.3 (Language-critique sufficiency)**.** The language label preserves reward-relevant distinctions between actions: there exists $c > 0$ such that, for any state s and actions a, a',

$$D_{\mathrm{KL}}\left(\mu(\cdot \mid s, a') \parallel \mu(\cdot \mid s, a)\right) \geq c \, \|\psi(s,a') - \psi(s,a)\|_2^2 . \tag{22}$$

This assumption states that reward-relevant differences in the state-action feature space are reflected by the induced language distributions. The condition rules out uninformative labels and requires μ to preserve distinctions in the features of state-action $\psi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$: if their reward-relevant feature representations differ, $\mu(\cdot \mid s, a)$ and $\mu(\cdot \mid s, a')$ shall be distinguishable. This assumption motivates structured language labels that encode task progress, action optimality, and movement guidance, rather than arbitrary free-form descriptions. See Section 4.2 for details. However, this design alone does not guarantee the inequality for a given feature representation. In Section A.6, we examine this distinction empirically.

Under Assumption A.3, the expected feature gap is bounded by the language-critique gap:

$$\sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E(\cdot \mid s_t), \hat{a}_t \sim \pi_\theta(\cdot \mid s_t)} \left[\left\| \psi(s_t, a_t^E) - \psi(s_t, \hat{a}_t) \right\|_2^2 \right] \leq \frac{1}{c} \Delta_{\mathrm{LC}}(\pi_\theta). \tag{23}$$

A.4 PROOF OF THE PERFORMANCE DIFFERENCE BOUND

Below we formally state the performance difference bound in Theorem A.4 with the detailed proof.

Theorem A.4 (Language-critique gap bounds the performance difference gap)**.** *Under Assumptions A.2 and A.3, the performance difference between the expert policy π_E and the learned policy π_θ can be upper bounded by the expert-referenced language-critique gap:*

$$|J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta)| \leq w_{\max} \sqrt{\frac{1}{c} \frac{1-\gamma^T}{1-\gamma} \Delta_{LC}(\pi_\theta)}. \tag{24}$$

Proof. By the performance difference lemma (Kakade & Langford, 2002), we have

$$J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t,\, a_t^E \sim \pi_E} \left[A^{\pi_\theta}_{r^*,t}(s_t, a_t^E) \right]. \tag{25}$$

Since $A^{\pi_\theta}_{r^*,t}(s,a) = Q^{\pi_\theta}_{r^*,t}(s,a) - V^{\pi_\theta}_{r^*,t}(s)$ and $V^{\pi_\theta}_{r^*,t}(s) = \mathbb{E}_{\hat{a} \sim \pi_\theta(\cdot \mid s)}[Q^{\pi_\theta}_{r^*,t}(s,\hat{a})]$, we have

$$\mathbb{E}_{a^E \sim \pi_E} \left[A^{\pi_\theta}_{r^*,t}(s, a^E) \right] = \mathbb{E}_{a^E \sim \pi_E,\, \hat{a} \sim \pi_\theta} \left[Q^{\pi_\theta}_{r^*,t}(s, a^E) - Q^{\pi_\theta}_{r^*,t}(s, \hat{a}) \right]. \tag{26}$$

Under Assumption A.2, $Q^{\pi_\theta}_{r^*,t}(s,a) = {w_t^{\pi_\theta}}^\top \psi(s,a)$, so

$$Q^{\pi_\theta}_{r^*,t}(s, a^E) - Q^{\pi_\theta}_{r^*,t}(s, \hat{a}) = {w_t^{\pi_\theta}}^\top \left(\psi(s, a^E) - \psi(s, \hat{a}) \right) = {w_t^{\pi_\theta}}^\top X_t, \tag{27}$$

where $X_t \coloneqq \psi(s_t, a_t^E) - \psi(s_t, \hat{a}_t)$. Plugging this back into Equation (25), we have

$$J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \, \mathbb{E} \left[{w_t^{\pi_\theta}}^\top X_t \right]. \tag{28}$$

By Jensen's inequality, $\|\mathbb{E}[X_t]\|_2^2 \le \mathbb{E}[\|X_t\|_2^2]$. Applying weighted Cauchy–Schwarz to the discounted sum,

$$\left(\sum_{t=0}^{T-1}\gamma^t\mathbb{E}[X_t]\right)^2 \le \left(\sum_{t=0}^{T-1}\gamma^t\right)\left(\sum_{t=0}^{T-1}\gamma^t\|\mathbb{E}[X_t]\|_2^2\right) \le \frac{1-\gamma^T}{1-\gamma}\sum_{t=0}^{T-1}\gamma^t\,\mathbb{E}\big[\|X_t\|_2^2\big]. \tag{29}$$

By Equation (23) (following Assumption A.3),

$$\sum_{t=0}^{T-1}\gamma^t\,\mathbb{E}\big[\|X_t\|_2^2\big] \le \frac{1}{c}\,\Delta_{LC}(\pi_\theta). \tag{30}$$

Combining Equation (29) and Equation (30),

$$\sum_{t=0}^{T-1}\gamma^t\,\|\mathbb{E}[X_t]\|_2 \le \sqrt{\frac{1}{c}\,\frac{1-\gamma^T}{1-\gamma}\,\Delta_{LC}(\pi_\theta)}. \tag{31}$$

Putting everything together,

$$|J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta)| = \left|\sum_{t=0}^{T-1}\gamma^t\mathbb{E}\left[{w_t^{\pi_\theta}}^\top X_t\right]\right| \tag{32}$$

$$\le w_{\max}\sum_{t=0}^{T-1}\gamma^t\,\|\mathbb{E}[X_t]\|_2 \tag{33}$$

$$\le w_{\max}\sqrt{\frac{1}{c}\,\frac{1-\gamma^T}{1-\gamma}\,\Delta_{LC}(\pi_\theta)}, \tag{34}$$

which proves Theorem A.4.

□

A.5 Connection to the practical LC-loss

The theoretical objective above is defined using the language-critique function μ. In practice, μ is instantiated in two stages. First, the language label generator μ_g produces structured language labels for the expert and general offline datasets. Second, the differentiable LLM-Captioner μ_ϕ is trained on $\mathcal{D}_G^{\text{lang}}$ to approximate these labels and is then frozen during policy training. In this subsection, we identify the relationship between the theoretical objective and the practical objective. First, we go through the details of the practical LC-loss.

For a language label $l_t = \{l_{t,1}, \ldots, l_{t,N_t}\}$ of token length N_t, the token-level cross entropy loss under μ_ϕ is

$$\ell_{\text{CE}}^\phi(l_t, s_t, a_t) = -\frac{1}{N_t}\sum_{n=1}^{N_t}\log\mu_\phi(l_{t,n}\,|\,l_{t,<n}, s_t, a_t).$$

By the chain rule of conditional probability, this satisfies the exact identity:

$$\ell_{\text{CE}}^\phi(l_t, s_t, a_t) = -\frac{1}{N_t}\log\mu_\phi(l_t\,|\,s_t, a_t), \tag{35}$$

i.e., the token-averaged loss is exactly the sequence log-likelihood scaled by $1/N_t$. The dataset-level losses are:

$$\mathcal{L}_{\text{CE}}(\pi_\theta) = \mathbb{E}_{(s_t,l_t)\sim\mathcal{D}_E^{\text{lang}},\hat{a}_t\sim\pi_\theta}\left[\ell_{\text{CE}}^\phi(l_t, s_t, \hat{a}_t)\right] \tag{36}$$

$$\mathcal{L}_{\text{CE}}(\pi_E) = \mathbb{E}_{(s_t,a_t^E,l_t)\sim\mathcal{D}_E^{\text{lang}}}\left[\ell_{\text{CE}}^\phi(l_t, s_t, a_t^E)\right] \tag{37}$$

$$\mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E) = [\mathcal{L}_{\text{CE}}(\pi_\theta) - \text{sg}(\mathcal{L}_{\text{CE}}(\pi_E))]_+ \tag{38}$$

where $[x]_+ = \max(x, 0)$, and $\text{sg}(\cdot)$ denotes stop-gradient, and $\mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E)$ is the token-level practical LC-loss. To formally connect $\mathcal{L}_{\text{LC}}$ to Δ_{LC}, we introduce the following assumptions corresponding to three key aspects in terms of approximation: bounded language label lengths, μ_ϕ fidelity, and time-step t weighting.

Assumption A.5 (Bounded language label lengths)**.** All the language labels yielded by μ_g have a limited length under some tokenization strategy. There exist constants $0 < N_{\min} \leq N_{\max} < \infty$ such that $N_{\min} \leq N_t \leq N_{\max}$ for all l_t in the $\mathcal{D}_E^{\text{lang}}$ and every policy sample.

Assumption A.6 (LLM-Captioner approximation on expert labels)**.** There exists $\epsilon_\phi \geq 0$ such that, for every $s_t \sim d_E^t$, $a_t^E \sim \pi_E(\cdot \mid s_t)$, $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$, and $l_t^E \sim \mu_g(\cdot \mid s_t, a_t^E)$,

$$\left|\log \mu_g(l_t^E \mid s_t, a) - \log \mu_\phi(l_t^E \mid s_t, a)\right| \leq \epsilon_\phi, \qquad a \in \{a_t^E, \hat{a}_t\}. \tag{39}$$

This assumes positive probabilities and uniformly bounded log-probability error for expert labels under both expert and policy actions.

Assumption A.7 (Time-step t coverage)**.** Let p_t denote the empirical frequency of timestep t in $\mathcal{D}_E^{\text{lang}}$, and define $q_t \coloneqq \gamma^t / p_t$. There exist constants $0 < q_{\min} \leq q_{\max} < \infty$ such that $q_{\min} \leq q_t \leq q_{\max}$ for all $t \in \{0, \ldots, T-1\}$. In the undiscounted finite-horizon setting, where $\gamma = 1$ with uniform timestep coverage $p_t = 1/T$, $q_t = T$ exactly.

The above three assumptions are fairly mild and can be satisfied by our implementation in practice. We now combine these assumptions to bound the theoretical gap $\Delta_{\text{LC}}(\pi_\theta)$ by the practical loss $\mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E)$. First, we define the captioner-based language critique gap, identical in form to $\Delta_{\text{LC}}(\pi_\theta)$, but using μ_ϕ instead of a general μ.

$$\Delta_{r,\phi}(\pi_\theta) \coloneqq \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E(\cdot|s_t), l_t^E \sim \mu_g, \hat{a}_t \sim \pi_\theta(\cdot|s_t)} \left[\log \mu_\phi(l_t^E | s_t, a_t^E) - \log \mu_\phi(l_t^E | s_t, \hat{a}_t)\right]. \tag{40}$$

We now bound $\Delta_{\text{LC}}(\pi_\theta)$ — instantiated with the generator $\mu = \mu_g$ — by $\Delta_{r,\phi}(\pi_\theta)$ using Assumption A.6. For every $(s_t, a_t, l_t) \in \mathcal{D}_E^{\text{lang}}$ and every policy sample $\hat{a}_t$, the assumption gives

$$\left|\log \mu_g(l_t^E \mid s_t, a_t^E) - \log \mu_\phi(l_t^E \mid s_t, a_t^E)\right| \leq \epsilon_\phi, \tag{41}$$

$$\left|\log \mu_g(l_t^E \mid s_t, \hat{a}_t) - \log \mu_\phi(l_t^E \mid s_t, \hat{a}_t)\right| \leq \epsilon_\phi. \tag{42}$$

Combining the two bounds, we have:

$$\left[\log \mu_g(l_t^E \mid s_t, a_t^E) - \log \mu_g(l_t^E \mid s_t, \hat{a}_t)\right] \leq \left[\log \mu_\phi(l_t^E \mid s_t, a_t^E) - \log \mu_\phi(l_t^E \mid s_t, \hat{a}_t)\right] + 2\epsilon_\phi.$$

Taking the expectation under $s_t \sim d_E^t$, $a_t^E \sim \pi_E$, $l_t^E \sim \mu_g$, $\hat{a}_t \sim \pi_\theta$, and the discounted sum $\sum_t \gamma^t$ on both sides, we obtain

$$\Delta_{\text{LC}}(\pi_\theta) \leq \Delta_{r,\phi}(\pi_\theta) + 2\epsilon_\phi \sum_{t=0}^{T-1} \gamma^t = \Delta_{r,\phi}(\pi_\theta) + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma}. \tag{43}$$

According to Equation (35) and Assumption A.5, we define a length-weighted gap for the LLM-style log-likelihood we have in practice:

$$\bar{\Delta}_{r,\phi}(\pi_\theta) \coloneqq \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E, l_t^E \sim \mu_g, \hat{a}_t \sim \pi_\theta} \left[\frac{1}{N_t}\left(\log \mu_\phi(l_t^E | s_t, a_t^E) - \log \mu_\phi(l_t^E | s_t, \hat{a}_t)\right)\right], \tag{44}$$

Then, we apply the clipping at 0 that we designed in our LC-loss, and we can have:

$$\bar{\Delta}_{r,\phi}^+(\pi_\theta) \coloneqq \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim d_E^t, a_t^E \sim \pi_E, l_t^E \sim \mu_g, \hat{a}_t \sim \pi_\theta} \left[\frac{1}{N_t}\left[\log \mu_\phi(l_t^E | s_t, a_t^E) - \log \mu_\phi(l_t^E | s_t, \hat{a}_t)\right]_+\right], \tag{45}$$

$$\Delta_{r,\phi}(\pi_\theta) \leq N_{\max} \bar{\Delta}_{r,\phi}^+(\pi_\theta) \tag{46}$$

Then, under Assumption A.7, we can rewrite the

$$\bar{\Delta}^{+}_{r,\phi}(\pi_\theta) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t\sim d_t^E, a_t^E\sim\pi_E, l_t^E\sim\mu_g, \hat{a}_t\sim\pi_\theta} \left[\frac{1}{N_t} \left[\log \mu_\phi(l_t^E|s_t, a_t^E) - \log \mu_\phi(l_t^E|s_t, \hat{a}_t) \right]_+ \right] \tag{47}$$

$$= \sum_{t=0}^{T-1} q_t p_t \mathbb{E}_{(s_t,a_t,l_t^E)\sim\mathcal{D}_E^{\text{lang}}, \hat{a}_t\sim\pi_\theta(\cdot\,|\,s_t)} \left[\frac{1}{N_t} \left[\log \mu_\phi(l_t^E|s_t, a_t^E) - \log \mu_\phi(l_t^E|s_t, \hat{a}_t) \right]_+ \right] \tag{48}$$

$$\leq q_{\max} \sum_{t=0}^{T-1} p_t \mathbb{E}_{(s_t,a_t,l_t^E)\sim\mathcal{D}_E^{\text{lang}}, \hat{a}_t\sim\pi_\theta(\cdot\,|\,s_t)} \left[\frac{1}{N_t} \left[\log \mu_\phi(l_t^E|s_t, a_t^E) - \log \mu_\phi(l_t^E|s_t, \hat{a}_t) \right]_+ \right] \tag{49}$$

$$= q_{\max}[\mathcal{L}_{\text{CE}}(\pi_\theta) - \mathcal{L}_{\text{CE}}(\pi_E)]_+ = q_{\max}\mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E). \tag{50}$$

Finally, by putting everything together, we have:

$$\Delta_{\text{LC}}(\pi_\theta) \leq \Delta_{r,\phi}(\pi_\theta) + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma} \tag{51}$$

$$\leq N_{\max}\bar{\Delta}^{+}_{r,\phi}(\pi_\theta) + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma} \tag{52}$$

$$\leq N_{\max} q_{\max} \left[\mathcal{L}_{\text{CE}}(\pi_\theta) - \mathcal{L}_{\text{CE}}(\pi_E)\right]_+ + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma} \tag{53}$$

$$= N_{\max} q_{\max}\, \mathcal{L}_{\text{LC}}(\pi_\theta, \pi_E) + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma}. \tag{54}$$

Combining Theorem A.4 and Equation (54), we establish the following bound:

Theorem A.8 (LC-loss bounds the performance difference gap)**.** *Combining the result in Equation* (54) *with Theorem A.4, under Assumptions A.2, A.3 and Assumptions A.5, A.6, A.7, the performance difference gap between the expert policy and the learned policy is bounded by the realized LC-loss up to an LLM-Captioner distillation residual:*

$$|J_{r^*}(\pi_E) - J_{r^*}(\pi_\theta)| \leq w_{\max} \sqrt{\frac{1}{c}\frac{1-\gamma^T}{1-\gamma} \left(N_{\max} q_{\max}\, \mathcal{L}_{LC}(\pi_\theta, \pi_E) + 2\epsilon_\phi \frac{1-\gamma^T}{1-\gamma} \right)}. \tag{55}$$

A.6 Empirical diagnostics of the language-critique assumptions

We examine empirical quantities related to language-critique sufficiency, captioner approximation, and bounded label lengths in Assumptions A.3, A.6, and A.5. We distinguish the original generator, a smoothed generator used for finite KL comparisons, and the learned captioner. Their measurements describe language separation, score discrepancies, and expert-label preferences on the evaluated examples; they do not establish the assumptions or numerically evaluate the performance bound.

A.6.1 Evaluation protocol and probability scoring

On BOX-CLOSE, we randomly sample $M = 10000$ examples without replacement from an augmented general dataset $\mathcal{D}_G^{\text{theo}}$. Here M counts evaluation examples, whereas N denotes label token length as in the theoretical analysis. Each example of timestep t stores a state-action pair (s_t, a_t), an expert action $a_t^E \sim \pi_E(\cdot \mid s_t)$, the features $\psi(s_t, a_t)$ and $\psi(s_t, a_t^E)$, and language labels $l_t \sim \mu_g(\cdot \mid s_t, a_t)$ and $l_t^E \sim \mu_g(\cdot \mid s_t, a_t^E)$. The 8-dimensional features for BOX-CLOSE are listed in Table 4.

The dataset action a_t is not assumed to be a draw $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$, and the sampled states are not assumed to follow d_E^t. We also retain the selected semantic-pool identifiers $\mathbf{z}_t, \mathbf{z}_t^E$ (where $\mathbf{z}_t = \{z_t^1, \ldots, z_t^H\}$) and their paraphrase pool size $\mathbf{m}_t, \mathbf{m}_t^E$ (where $\mathbf{m}_t = \{m_t^1, \ldots, m_t^H\}$) corresponding to the generator μ_g and l_t, l_t^E. Then, we evaluate the original $\mathcal{D}_G^{\text{lang}}$-distilled 135M LLM-Captioner

Table 4: Raw features underlying the language feedback on BOX-CLOSE.

Feature	Description
`hand_lid_dist`	Euclidean distance between the hand and lid.
`lid_box_dist`	Euclidean distance from the lid to the box placement target, whose height is fixed at 0.15 m.
`lid_height`	Absolute vertical coordinate of the lid.
`target_delta_x`	Signed displacement from the hand to the reference target along the x-axis.
`target_delta_y`	Signed displacement from the hand to the reference target along the y-axis.
`target_delta_z`	Signed displacement from the hand to the reference target along the z-axis.
`gripper_diff`	Reference minus executed gripper command; positive values indicate a command toward greater closure.
`dist_delta`	Hand-to-target distance before execution minus that after execution, using the same reference target; positive values indicate movement to the target.

μ_ϕ for our main results using the original eight-dimensional features for BOX-CLOSE. All logarithms are natural, with log-probability differences and KL divergences reported in nats.

Under the μ_g generator's independent, uniform paraphrase-selection model, the probability of a resulting label is:

$$\log \mu_g(l_t \mid s_t, a_t) = -\sum_{h=1}^{H} \log m_t^h, \tag{56}$$

where $H = 17$ is the number of categorical slots and m_h is the number of paraphrases in the selected pool for μ_g of BOX-CLOSE. For a stored label $l_t = (l_{t,1}, \ldots, l_{t,N_t})$, we obtain the captioner's label-token log score:

$$S_\phi(l_t; s_t, a_t) = \sum_{n=1}^{N_t} \log \mu_\phi(l_{t,n} \mid l_{t,<n}, s_t, a_t). \tag{57}$$

Prompt and padding tokens are excluded. In the label-token convention of the cross-entropy identity in Equation (35), $S_\phi(l_t; s_t, a_t) = -N_t \ell^\phi_{\mathrm{CE}}(l_t, s_t, a_t)$. We retain the score notation to make explicit that the implementation does not add an end-of-sequence token when scoring stored labels.

A.6.2 LANGUAGE-CRITIQUE SUFFICIENCY: EMPIRICAL CONSTANT RATIOS

The language-critique sufficiency in Assumption A.3 motivates examining

$$r_{\mu,t} = \frac{D_{\mathrm{KL}}\big(\mu(\cdot \mid s_t, a_t^E) \,\big\|\, \mu(\cdot \mid s_t, a_t)\big)}{\|\psi(s_t, a_t^E) - \psi(s_t, a_t)\|_2^2}. \tag{58}$$

We evaluate ratios only for pairs with $\|\psi(s_t, a_t^E) - \psi(s_t, a_t)\|_2^2 \geq 10^{-8}$, excluding pairs with zero feature differences. For any nonempty subset $\mathcal{I}$ of these eligible pairs, we define the empirical minimum $\widehat{c}_\mu(\mathcal{I}) = \min_{t \in \mathcal{I}} r_{\mu,t}$. Unlike the uniform constant c in Assumption A.3, this minimum concerns only the evaluated subset. With exact KL values, any constants satisfying the assumption over the full state-action domain must obey $c \leq \widehat{c}_\mu(\mathcal{I})$; a positive empirical minimum does not say that the above assumption holds globally.

The feature-distance threshold excludes 529 of the 10000 examples, leaving 9471 eligible pairs. Among them, 9112 pairs have different semantic-pool selections of their language labels ($\mathbf{z}_t \neq \mathbf{z}_t^E$), while 359 pairs have identical selections ($\mathbf{z}_t = \mathbf{z}_t^E$).

LLM-Captioner μ_ϕ constant ratios. We draw $k = 32$ autoregressive samples from $\mu_\phi(\cdot \mid s_t, a_t^E)$ using temperature 1, without top-k or top-p filtering, and a fixed budget of 100 generated tokens. At each scored prefix, we compute the KL between the full next-token distributions under the two action conditionings. The estimate is

$$\widehat{K}_{\phi,t} = \frac{1}{k} \sum_{j=1}^{k} \sum_{n=1}^{N_{t,j}} D_{\mathrm{KL}}\Big(\mu_\phi(\cdot \mid s_t, a_t^E, l_{t,<n}^{(j)}) \,\Big\|\, \mu_\phi(\cdot \mid s_t, a_t, l_{t,<n}^{(j)})\Big), \tag{59}$$

where $N_{t,j}$ denotes the number of scored positions. At each token position, we compute the KL divergence by summing over all possible next tokens, then average the accumulated KL across the sampled sequences. Because the generated sequences are converted to text and tokenized again before scoring, their tokenization may change. We therefore treat the resulting values as approximate KL estimates over the generated prefixes.

Generator μ_g constant ratios. The language generator μ_g constructs a label by selecting a semantic pool for each component and sampling a paraphrase from that pool. Different actions may select different pools. Consequently, a label that can be generated under the expert action may have zero probability under the dataset action. In this case, the KL divergence between the two language distributions is infinite, preventing a finite numerical comparison with the captioner.

To obtain a finite comparison, we evaluate a smoothed version of the language generator, denoted μ_g^δ. For each component, this generator selects the original semantic pool with probability $1-\delta$, while assigning a small probability to each alternative. Specifically, the remaining probability δ is distributed uniformly across the B^h alternative states of component h, including an absent state representing omission of that component where applicable. We use $\delta = 0.05$, so the original selection receives probability 0.95, and each alternative receives probability $0.05/B^h$.

Let z_t^h and $z_t^{E,h}$ denote the original pool selections for component h under the dataset and expert actions, respectively. If these selections are identical, the two smoothed distributions for that component are identical and contribute zero KL. If they differ, each distribution assigns probability $1-\delta$ to its own original selection and δ/B^h to the other selection. Because the generator samples the components independently, their KL contributions add, giving

$$K_{g,t}^\delta = \sum_{h:, z_t^{E,h} \neq z_t^h} \left(1 - \delta - \frac{\delta}{B^h}\right) \log\left(\frac{(1-\delta)B^h}{\delta}\right). \tag{60}$$

We then divide this generator KL by the squared feature difference to obtain the per-pair constant ratio:

$$r_{g,t}^\delta = \frac{K_{g,t}^\delta}{\|\psi(s_t, a_t^E) - \psi(s_t, a_t)\|_2^2}. \tag{61}$$

We report the empirical minimum $\widehat{c}_g^{\delta,(F)}$, lower percentiles, and median of the adjusted ratios defined below, separately for all eligible pairs, different-pool pairs, and same-pool pairs.

This smoothing is used only for the empirical comparison and does not change the labels used to train the captioner. It also does not distinguish actions that select the same pools: their generator KL remains zero, even when their feature vectors differ. Such pairs therefore prevent a positive generator sufficiency constant over all evaluated pairs.

Reporting convention. For the numerical ratio summaries, we add $F = 10^{-6}$ nats to both the estimated captioner KL and the smoothed-generator KL:

$$\widehat{r}_{\phi,t}^{(F)} = \frac{\widehat{K}_{\phi,t} + F}{\|\psi(s_t, a_t^E) - \psi(s_t, a_t)\|_2^2}, \tag{62}$$

$$r_{g,t}^{\delta,(F)} = \frac{K_{g,t}^\delta + F}{\|\psi(s_t, a_t^E) - \psi(s_t, a_t)\|_2^2}. \tag{63}$$

For any nonempty subset $\mathcal{I}$ of eligible pairs, we define the empirical minimum as:

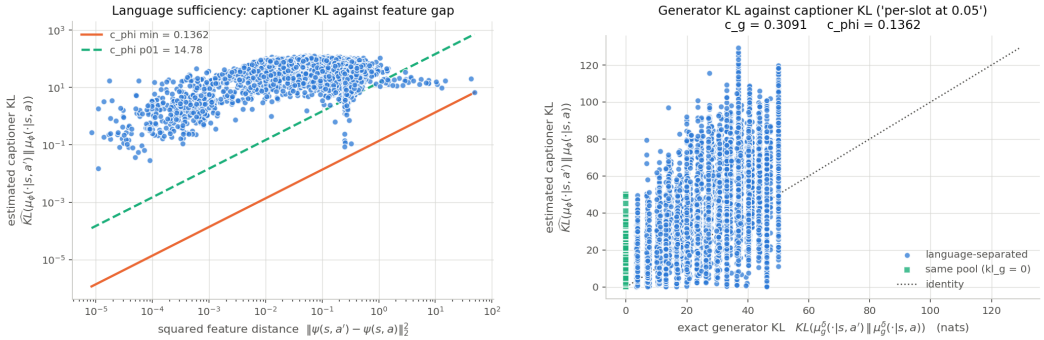
$$\widehat{c}_\phi^{(F)}(\mathcal{I}) = \min_{t \in \mathcal{I}} \widehat{r}_{\phi,t}^{(F)}, \tag{64}$$

$$\widehat{c}_g^{\delta,(F)}(\mathcal{I}) = \min_{t \in \mathcal{I}} r_{g,t}^{\delta,(F)}. \tag{65}$$

This numerical adjustment affects only the ratio summaries; KL summaries and correlations use the unmodified KL values. In particular, same-pool generator pairs have zero raw KL but positive adjusted ratios. These adjusted ratios do not establish a positive sufficiency constant for the original KL inequality.

Table 5: **Language-critique sufficiency: empirical constant ratios.** Ratios divide language KL by squared feature distance; minima are $\widehat{c}_{\phi}^{(F)}$ for the captioner and $\widehat{c}_{g}^{\delta}$ for the smoothed generator, each restricted to the listed subset. "Different" and "Same" denote different and identical semantic-pool selections. "Reduced" removes `gripper_diff`. Captioner summaries retain the same-pool floor $F = 10^{-6}$ nats; the reported minima for the full and reduced all-pairs sets are attained on different-pool pairs and are not directly altered by that floor. Generator rows use per-slot smoothing with $\delta = 0.05$ and no KL floor.

Distribution	Features	Pools	Pairs	Minimum	1st pct.	Median
Captioner	Full	All	9,471	0.136	14.78	156.56
Captioner	Full	Different	9,112	0.136	18.82	161.10
Captioner	Full	Same	359	0.341	3.53	64.80
Generator μ_g^{δ}	Full	All	9,471	0	0	104.49
Generator μ_g^{δ}	Full	Different	9,112	0.309	13.39	112.92
Generator μ_g^{δ}	Full	Same	359	0	0	0
Captioner	Reduced	All	9,446	1.243	63.40	1,731.13
Generator μ_g^{δ}	Reduced	Different	9,112	1.621	83.42	832.67



(a) Captioner KL and squared feature distance.

(b) Generator and LLM-Captioner KL divergences.

Figure 4: **Language-critique sufficiency and constant ratios.** (a) All 9,471 eligible pairs with the full feature representation. The solid line uses the reported minimum ratio; the dashed line uses the first-percentile ratio and is not a lower bound for every pair. (b) Generator KL uses per-slot smoothing with $\delta = 0.05$. Across the 9,112 different-pool pairs, Pearson $r = 0.699$ and Spearman $\rho = 0.687$. Same-pool generator KL is exactly zero (green points). The $c_g = 0.3091$ annotation in panel (b) refers only to different-pool pairs; the unfloored all-pairs generator minimum is zero. In the plot axes, a' denotes a^E; the plotted c annotations denote empirical minima, not certified global constants. The dotted identity line is a reference, not a theoretical equality.

Results. As shown in Table 5, the reported captioner constant ratios have a minimum of 0.136, a first percentile of 14.78, and a median of 156.56 across all eligible pairs. Each ratio measures how much the captioner's language distribution changes between two actions relative to their squared feature difference. A larger ratio therefore indicates a greater difference between the language distributions for a given feature difference. The large gap between the minimum and the first percentile shows that the smallest ratios are substantially below those of most evaluated pairs. This distinction matters because the sufficiency assumption requires a lower bound that holds for every pair, making the empirical minimum sensitive to even a single pair with a small ratio.

The pair attaining the reported minimum has different semantic-pool selections. Its estimated captioner KL is 6.71 nats, with a Monte Carlo standard error of 0.326 nats, while its squared feature difference is 49.27. Adding $F = 10^{-6}$ nats to the KL and dividing by the squared feature difference gives a ratio of approximately 0.136; the adjustment is negligible at the reported precision. Thus, the captioner distinguishes the two actions.

Table 6: **LLM-Captioner μ_ϕ approximation errors (nats).** Self-support errors compare the original generator and captioner on stored labels under their generating actions. Cross-action rows use pair-specific backoff with $\eta = 10^{-6}$ and are convention-dependent diagnostics. Each directional row contains 10,000 scores; the combined self-support row contains 20,000.

Label / conditioning	Mean	Median	95th pct.	Maximum
Dataset label / dataset action	1.48	0.77	5.45	23.30
Expert label / expert action	4.07	3.24	10.76	37.24
Combined self-support	2.77	1.49	9.02	37.24
Expert label / dataset action (backoff)	11.47	8.86	31.42	55.37
Dataset label / expert action (backoff)	9.09	7.67	23.85	61.04

We also examine whether the captioner and the smoothed generator agree about which action pairs produce larger differences in language. Across pairs with different semantic-pool selections, their KL divergences are positively correlated: Pearson $r = 0.699$ and Spearman $\rho = 0.687$ (Figure 4). In other words, pairs that produce a larger change in the generator's language distribution also tend to produce a larger change in the captioner's distribution. This association suggests that the captioner reflects some of the action distinctions encoded by the generator, although it does not establish that the two models assign similar probabilities to individual labels.

The pairs with identical semantic-pool selections reveal a limitation of the generator. Among the eligible pairs, 359 pairs, or 3.79%, have different feature vectors but select the same pools for every language component. The generator produces identical language distributions for these actions, giving zero KL despite a nonzero feature difference. Smoothing preserves this equality. The numerical adjustment makes the reported ratios positive, but does not change the underlying distributions or their KL. Thus, these pairs violate Assumption A.3 for both μ_g and μ_g^δ under the full feature representation. The raw generator ratio minimum is zero over all eligible pairs, whereas its positive raw minimum on different-pool pairs applies only to that subset.

A.6.3 LLM-Captioner μ_ϕ approximation and language-critique margins

Motivated by Assumption A.6, we first measure approximation errors on stored generator labels under their own generating actions:

$$\epsilon_t^a = \left|\log \mu_g(l_t \mid s_t, a_t) - S_\phi(l_t; s_t, a_t)\right|, \tag{66}$$

$$\epsilon_t^E = \left|\log \mu_g(l_t^E \mid s_t, a_t^E) - S_\phi(l_t^E; s_t, a_t^E)\right|, \tag{67}$$

and define $\widehat{\epsilon}_{\phi,\text{self}} = \max_t\{\epsilon_t^a, \epsilon_t^E\}$. The "self" subscript emphasizes that this observed maximum covers only labels scored under their generating actions. It does not control the error on expert labels scored under other actions, which is also required by Assumption A.6. Each stored generator label is scored token by token, conditioned on its preceding label tokens.

For cross-action diagnostics, μ_g assigns zero probability to the expert label under the dataset action in 91.1% of examples. Because the captioner assigns these labels positive probability, the corresponding absolute log-probability discrepancies are infinite. Thus, no finite approximation constant covers these evaluated cross-action comparisons. To obtain a finite diagnostic, we use a separate, pair-specific backoff reference:

$$\widetilde{\mu}_{g,t}^{\eta,a} = (1-\eta)\mu_g(\cdot \mid s_t, a_t) + \eta\mu_g(\cdot \mid s_t, a_t^E), \qquad \eta = 10^{-6}, \tag{68}$$

using $|\log \widetilde{\mu}_{g,t}^{\eta,a}(l_t^E) - S_\phi(l_t^E; s_t, a_t)|$, with the reverse direction defined analogously. This convention differs from the per-slot smoothing used for generator KL, and its errors are reported separately; they are not approximation errors against the original generator. These comparisons use stored dataset actions rather than online samples from π_θ, so they only test the assumption under the theorem's expert-state and offline behavior policy distribution.

Across 20,000 self-support scores, the mean and median errors are 2.77 and 1.49 nats, while the 95th percentile and maximum are 9.02 and 37.24 nats (Table 6). Expert-action tuples have larger mean error than dataset-action tuples (4.07 versus 1.48 nats). The difference between typical and maximum

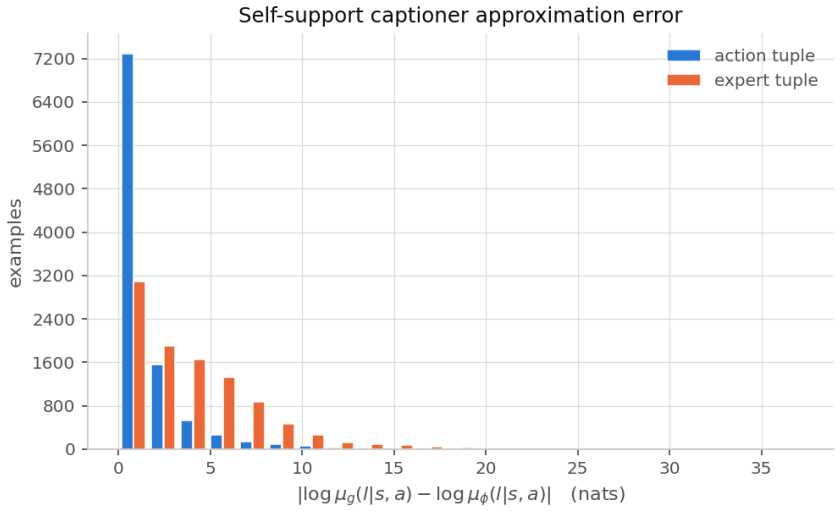


Figure 5: **LLM-Captioner μ_ϕ approximation on self-support labels.** Absolute differences between original-generator log probabilities and captioner label-token log scores. Each stored label is evaluated under its generating state–action conditioning. Dataset-action tuples have a mean error of 1.48 nats, compared with 4.07 nats for expert-action tuples. The combined median is 1.49 nats, substantially below the empirical maximum of 37.24 nats.

errors shows that a uniform approximation residual can be substantially larger than the error on a typical evaluated label.

To examine the preference relevant to policy supervision, we score the same expert label under both actions:

$$\delta_{\phi,t} = S_\phi(l_t^E; s_t, a_t^E) - S_\phi(l_t^E; s_t, a_t). \tag{69}$$

This is a sample counterpart of the integrand defining $\Delta_{r,\phi}(\pi_\theta)$, with the stored dataset action replacing the learned-policy action. It is not a direct estimate of the discounted expert-state quantity $\Delta_{r,\phi}(\pi_\theta)$ or of $\Delta_{\mathrm{LC}}(\pi_\theta)$. Nor is it itself a captioner KL: its label is drawn from μ_g, whereas the captioner KL averages over labels drawn from μ_ϕ. The margin is positive on 87.66% of examples, with mean 16.12 and median 14.04 nats. Thus, the captioner usually assigns higher likelihood to expert feedback under the expert action than under the paired dataset action. This supports the intended ranking behavior of the auxiliary objective, although it does not by itself establish the quality of its action gradients or an improvement in return.

A.6.4 Bounded language label lengths

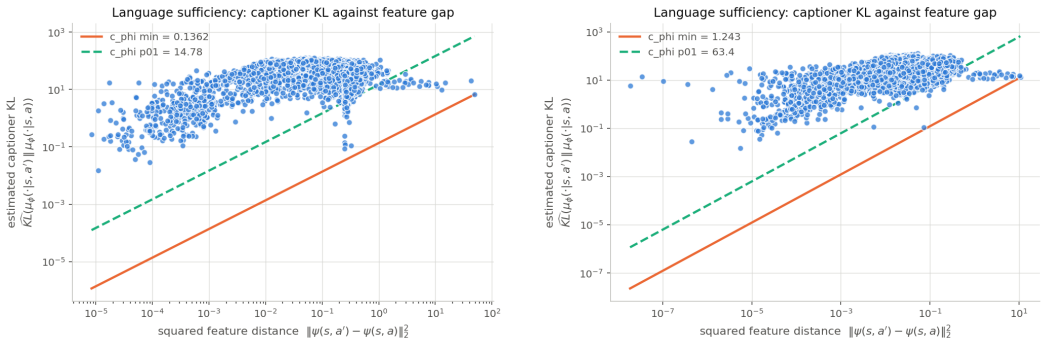
For the bounded language label lengths in Assumption A.5, we report observed extrema across the stored dataset-action and expert-action labels:

$$\widehat{N}_{\min} = \min_t\{N_t, N_t^E\} = 16, \qquad \widehat{N}_{\max} = \max_t\{N_t, N_t^E\} = 58. \tag{70}$$

These values concern label token lengths under the scoring tokenizer. They are distinct from the 100-token generation budget used to estimate captioner KL, and they do not establish the assumption's global $N_{\min}$ and $N_{\max}$ over all possible labels.

A.6.5 Sensitivity to features and evaluation sample size

Feature representation. On the different-pool subset, we discovered that `gripper_diff` accounts for 76.84% of the summed squared feature gap. We recompute the ratios after removing this component, retaining the same KL estimates. A further 25 pairs fall below the feature-distance threshold, leaving 9446 eligible pairs. The reported all-pairs LLM-Captioner ratio minimum $\widehat{c}_\phi^{(F)}(\mathcal{I})$ increases from 0.136 to 1.243, and the first percentile from 14.78 to 63.40 (Table 5 and Figure 6).



(a) Full eight-dimensional feature map.

(b) Feature map without `gripper_diff`.

Figure 6: **Sensitivity of constant ratios to the feature representation.** The captioner KL estimates are unchanged; removing `gripper_diff` changes only the squared feature distances and excludes 25 additional near-zero-gap pairs. The reported all-pairs minimum ratio increases from 0.136 to 1.243. On the unchanged different-pool subset, the Spearman correlation between squared feature distance and captioner KL changes from -0.124 to 0.363. The ratio increase is partly a consequence of reducing the denominator and does not demonstrate improved captioner behavior or a tighter return bound. The plots use different axis ranges.

On the unchanged different-pool subset, the Spearman correlation between squared feature distance and captioner KL changes from -0.124 to 0.363. This suggests that the original feature metric is dominated by a component whose variation is poorly aligned with language separation in this sample. Increasing the ratios is partly automatic when the denominator decreases; the ablation does not improve the LLM-Captioner and does not establish value-function realizability for the reduced feature map. It is a post hoc diagnostic, reported alongside the full-feature results.

Evaluation sample size. To examine how the reported statistics depend on the number and selection of evaluation examples, we repeatedly select smaller subsets from the 10000 examples already measured. We randomly select 1000 examples and recompute the statistics using only those examples, repeating this procedure 50 times. We then repeat the same procedure with subsets of 5000 examples. Each subset contains no duplicate examples, although different subsets may overlap. We reuse the previously computed KL estimates and approximation errors, so this analysis requires no additional captioner generation or scoring. For each subset size, we report the median and range of the statistics across the 50 selections. This measures sensitivity to the choice of evaluation examples; it does not measure variability from retraining the captioner or drawing new language samples.

For each subset, we compute captioner constant ratios over eligible pairs with different semantic-pool selections. Approximation errors include both dataset-action and expert-action labels, each scored under its generating action. In Table 7, we report the median and range of each statistic across the 50 random selections. Mean approximation error remains near 2.8 nats. The fifth-percentile ratio is also stable: its median across selections is 42.58 at both subset sizes, close to 42.45 on the full set. In contrast, minimum ratios and maximum errors vary more because smaller subsets may miss extreme examples. We therefore report means and percentiles alongside extrema to show both typical behavior and the most extreme observed cases. This analysis measures sensitivity to example selection within the existing evaluation set. We do not assess variability from new language samples, training runs, or datasets.

A.6.6 Scope and Limitations

These diagnostics concern one task, one captioner checkpoint, and general-dataset states paired with stored expert actions; they do not directly reproduce the expert-policy or learned-policy sampling distribution. Decoding, re-tokenization, and stopping-token handling introduce further differences from exact sequence-distribution KL; neither unbiasedness nor a guaranteed lower bound is claimed for the numerical estimator. Finite-sample extrema do not certify global assumptions, and the additive KL adjustment just makes the ratio measurable but does not resolve the observed violations

Table 7: **Sample-size stability of constant ratios and LLM-Captioner approximation errors.** For $M' = 1000$ and 5000, each cell gives the median across 50 random subsample sets, with the observed range underneath in brackets. The final row uses all 10000 measured examples. Ratio statistics use different-pool pairs; approximation errors use both self-support conditions and are in nats. Subsampling reuses existing KL estimates; ranges are not confidence intervals or independent training-run variability.

Sample size	$\widehat{c}_\phi$	5th-pct. ratio	Mean ϵ error	Maximum ϵ error
1,000	1.537 [0.136, 13.275]	42.58 [38.10, 46.85]	2.789 [2.611, 2.954]	27.88 [23.08, 37.24]
5,000	0.136 [0.136, 1.847]	42.58 [40.44, 44.30]	2.774 [2.724, 2.813]	36.83 [30.23, 37.24]
10,000	0.136	42.45	2.773	37.24

of generator sufficiency. The observed self-conditioned errors characterize μ_ϕ LLM-Captioner fidelity on generator-supported labels. Cross-action support mismatches, however, produce infinite discrepancies against the original generator on the affected pairs, and the finite backoff errors concern a modified reference. Neither statistic therefore fully supplies the uniform ϵ_ϕ required by the generator-to-captioner argument for Assumption A.6. The available tuples also do not identify the time-step coverage factor $q_{\max}$ in Assumption A.7 or the linear-realizability coefficient norm bound $w_{\max} = \sup_{\pi,t} \|w_t^\pi\|_2$ in Assumption A.2. Together, these diagnostics show that the captioner assigns higher likelihood to expert critiques under expert actions in 87.66% of evaluated pairs, and its language separation correlates positively with that of the smoothed generator. Typical approximation errors are substantially below the observed maximum, while mean errors and lower-percentile separation ratios remain stable across evaluation subsamples. These findings support the intended expert-action preference and indicate that the captioner retains meaningful distinctions from the generator's supervision. However, the observed sufficiency violations arise from the coarse semantic partition used by μ_g, which maps some feature-distinct actions to identical language distributions. A richer language-critique function μ that preserves these distinctions could better satisfy the sufficiency condition and enable a more informative theoretical bound.

B Algorithm

The detailed procedures of LC-BC and LC-DP are presented in Algorithm 1 and Algorithm 2, respectively. First, we annotate both the expert dataset $\mathcal{D}_E$ and the general dataset $\mathcal{D}_G$ with language labels by the language label generator μ_g to form $\mathcal{D}_E^{\text{lang}}$ and $\mathcal{D}_G^{\text{lang}}$. Then, the LLM-Captioner μ_ϕ, initialized from a pretrained LLM Ψ and projector weight W, is finetuned from a pretrained LLM on the general dataset $\mathcal{D}_G^{\text{lang}}$ to distill μ_g. In our main results, we use the SmolLM2-135M-Instruct model (Allal et al., 2025) as the LLM-Captioner's backbone. Finally, the policy is trained using the proposed LC-loss $\mathcal{L}_{\text{LC}}$ computed via the LLM-Captioner, simultaneously optimizing the original behavior cloning objective $\mathcal{L}_{\text{BC}}$ on the expert dataset $\mathcal{D}_E^{\text{lang}}$. This forms the algorithm of language-critique behavior cloning (LC-BC).

For diffusion policies (Chi et al., 2023), we additionally incorporate a reweighted LC-loss to balance contributions of LC-loss across different diffusion timesteps k, where the reweighting factor is derived in Section E.3. Following Kang et al. (2023), we apply one-step reconstruction to recover the clean action from the predicted noise at each timestep, and compute the reweighted LC-loss $\tilde{\mathcal{L}}_{\text{LC}}$ on the reconstructed action $\hat{a}_t^0$. The diffusion policy is then optimized jointly with this reweighted LC-loss and the standard diffusion objective $\mathcal{L}_{\text{DP}}$ on the expert dataset $\mathcal{D}_E^{\text{lang}}$.

C Language label details

This section provides additional details on the design and sources of language labels. We first describe the semantic structure of the labels, and then discuss how the language-generation function μ can be instantiated in practice.

Algorithm 1 Language-critique Behavior Cloning

Input: Expert dataset $\mathcal{D}_E$, General dataset $\mathcal{D}_G$, LC-loss weight λ, Feedforward policy π_θ, Language label generator μ_g, and LLM-Captioner μ_ϕ
Generate $\mathcal{D}_E^{\text{lang}}$ by μ_g with $\mathcal{D}_E$
Generate $\mathcal{D}_G^{\text{lang}}$ by μ_g with $\mathcal{D}_G$
// Finetune LLM-Captioner μ_ϕ
Initialize μ_ϕ with a pretrained LLM Ψ and a projector W
for each W finetuning iteration **do**
  Sample $(s_t, a_t, l_t) \sim \mathcal{D}_G^{\text{lang}}$
  Update W with ℓ_{CE}^{ϕ} from Equation (5)
end for
for each μ_ϕ finetuning iteration **do**
  Sample $(s_t, a_t, l_t) \sim \mathcal{D}_G^{\text{lang}}$
  Update μ_ϕ with ℓ_{CE}^{ϕ} from Equation (5)
end for
// Learn policy π_θ
Randomly initialize π_θ
for each policy iteration **do**
  Sample $(s_t, a_t^E, l_t) \sim \mathcal{D}_E^{\text{lang}}$
  $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$
  Compute $\mathcal{L}_{\text{BC}}$ from Equation (10)
  Compute $\mathcal{L}_{\text{LC}}$ from Equation (9)
  Update π_θ with $\mathcal{L}_{\text{BC}} + \lambda\mathcal{L}_{\text{LC}}$
end for
return π_θ

Algorithm 2 Language-critique Diffusion Policy

Input: Expert dataset $\mathcal{D}_E$, General dataset $\mathcal{D}_G$, LC-loss weight λ, Diffusion policy ϵ_θ, timestep K, Language label generator μ_g, and LLM-Captioner μ_ϕ
Generate $\mathcal{D}_E^{\text{lang}}$ by μ_g with $\mathcal{D}_E$
Generate $\mathcal{D}_G^{\text{lang}}$ by μ_g with $\mathcal{D}_G$
// Finetune LLM-Captioner μ_ϕ
Initialize μ_ϕ with a pretrained LLM Ψ and a projector W
for each W finetuning iteration **do**
  Sample $(s_t, a_t, l_t) \sim \mathcal{D}_G^{\text{lang}}$
  Update W with ℓ_{CE}^{ϕ} from Equation (5)
end for
for each μ_ϕ finetuning iteration **do**
  Sample $(s_t, a_t, l_t) \sim \mathcal{D}_G^{\text{lang}}$
  Update μ_ϕ with ℓ_{CE}^{ϕ} from Equation (5)
end for
// Learn diffusion policy ϵ_θ
Randomly initialize ϵ_θ
for each policy iteration **do**
  Sample $(s_t, a_t^E, l_t) \sim \mathcal{D}_E^{\text{lang}}$
  Sample $k \sim \text{Uniform}(\{1, \ldots, K\})$
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $a_t^k = \sqrt{\bar{\alpha}^k} a_t^E + \sqrt{1 - \bar{\alpha}^k}\epsilon$
  $\hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k)$
  Compute $\mathcal{L}_{\text{DP}}$ from Equation (11)
  Reconstruct $\hat{a}_t^0$ from Equation (12)
  Compute $\tilde{\mathcal{L}}_{\text{LC}}$ with ω^k
  Update ϵ_θ with $\mathcal{L} = \mathcal{L}_{\text{DP}} + \lambda\tilde{\mathcal{L}}_{\text{LC}}$
end for
return ϵ_θ

Language label design. Assumption A.3 requires that reward-relevant differences in feature space are reflected in the induced language distributions. In other words, when two state-action pairs differ in their reward-relevant semantics, their corresponding language labels should also be distinguishable. This requirement motivates labels that expose task-relevant information rather than unconstrained descriptions of every transition.

Motivated by reward design principles (Zhang et al., 2025; Ng et al., 1999), we decompose each language label into three components: task progress <T>, action optimality <A>, and movement guidance <M>. The goal of this decomposition is not to exhaustively describe all aspects of a transition, but to provide structured and compact supervision for learning from mixed-quality demonstrations.

The task-progress component <T> describes the semantic stage of the current state. This component identifies what part of the task has been achieved and which subgoal is currently relevant. For example, in manipulation tasks, <T> may distinguish whether the end-effector is approaching an object, whether the object has been contacted, whether the object is being moved toward the target, or whether the task is completed. By explicitly representing task progress, <T> helps distinguish states that may be close in raw observation space but correspond to different stages of the long-horizon objective.

The action-optimality component <A> characterizes the quality of the demonstrated action under the current task context. Instead of assigning a scalar signal or confidence weight, <A> expresses whether the action is beneficial, ineffective, or harmful for task completion. This component is particularly important in mixed-quality datasets, where suboptimal demonstrations may contain both useful intermediate states and undesirable actions. <A> provides supervision for deciding which behaviors should be imitated and which should be down-weighted or corrected.

The movement-guidance component <M> provides fine-grained corrective information about how the agent should move. While <T> captures the high-level task stage and <A> captures action quality, <M> describes the direction or adjustment needed to make progress. For instance, it may indicate that the agent should move toward a target, align with an object, push in a particular direction, reduce excessive motion, or correct an overshooting behavior. This component is especially useful for continuous-control tasks, where small differences in actions can lead to different outcomes but may not be fully captured by coarse task-stage labels alone.

Together, <T>, <A>, and <M> provide complementary supervision. <T> specifies where the agent is in the task, <A> evaluates whether the demonstrated action is appropriate, and <M> describes how the behavior should be adjusted. This structured design encourages the induced language distribution to distinguish behaviors that matter for task success while reducing irrelevant variation caused by unconstrained natural language. We evaluate the contribution of these components in Section 5.4.

For example, in the PEGINSERT task (Figure 3e), the robot arm must grasp a peg and insert it into the side of a box. A generated language label may include task progress, such as *Align the peg to the hole*; action optimality, such as *The action is bad for the state, since the peg misaligns with the hole.*; and movement guidance, such as *Plus, you can pitch to the top softly.* Combined, the final label is: *Align the peg to the hole. But the action is bad for the state, since the peg misaligns with the hole. Plus, you can pitch to the top softly.* Additional examples across tasks are provided in Table 8.

Sources of language labels. Given the desired label structure, the remaining question is how the language-generation function μ is instantiated. In general, μ can take several forms. Human annotators can provide semantically rich feedback, but annotation is labor-intensive, expensive to scale, and may introduce inconsistency. LLMs and VLMs can generate language labels from symbolic states, actions, or visual observations, and have been widely used for language-based reasoning and decision-making in embodied agents (Sumers et al., 2023; Driess et al., 2023; Jones et al., 2025). However, the induced language distribution can vary substantially with the model, prompt template, decoding strategy, and input representation. In addition, LLM-based or VLM-based generation may introduce computational overhead when applied to large offline datasets. Our proposed language label generator provides a controlled alternative: it is less flexible than human or model-generated feedback, but offers reproducible and consistent labels when task states and relevant geometric relations are available.

In this work, we mainly instantiate μ with a language label generator μ_g built on LLF-Bench (Cheng et al., 2024). We implement a customized version of the LLF-Bench generation pipeline to produce

Figure 7: **VLM Prompting.** The prompt we use to generate language labels with `o4-mini`.

VLM Prompt with `o4-mini`.

You are shown two images:

- Image 1 is the state BEFORE the action (current state).
- Image 2 is the state AFTER the action (next state).

Below is additional information about the system:
(STATE SPACE) {state space description}
(ACTION SPACE) {action space description}
You are also given the numeric representations of both states and the action taken:
(NUMERIC STATE) current state: {state} ; next state: {next state} ; action taken: {action}
The agent's task is described as follows (in second-person view): {task instruction}
Use BOTH the images and the numeric states to evaluate the transition. Before anything else, identify the main objects in the scene (robot arm, gripper, target object, goal region, obstacles) and reason about their spatial relationships (left/right, nearer/farther, higher/lower, touching/not touching). Use the numeric state values to clarify small or ambiguous visual changes, such as exact distances, object displacement, or contact. Focus on how positions, distances, and contacts changed from the previous state to the next state. Answer the following questions:

1. Using BOTH images and numeric states, describe the relative positions of the agent, the main object, and the goal.
2. In the previous state, determine the agent's task stage based on the gripper–object–goal configuration.
3. Describe the key spatial differences between the two states, focusing ONLY on task-relevant objects such as changes in distance, movement direction, contact, or orientation.
4. Decide whether the action was GOOD or BAD for accomplishing the task goal, tolerating small errors or minor regressions unless the action clearly moves the agent meaningfully farther from success.
5. Provide a concise recommendation for what the agent should do NEXT, specifying clear spatial directions such as move left/right/forward/backward/up/down, open or close the gripper, or push or pull.

First, write a few sentences of reasoning that:

- Describe the relative positions of the agent, the main object, and the goal using both images and numeric states.
- Identify the task stage in the previous state.
- Explain what visually and numerically changed between the two states.
- Justify why the action is good or bad in terms of progress toward the goal.
- Explain why your recommended next movement is appropriate based on spatial reasoning and numeric-state dynamics.

Then, on a NEW LINE at the end, output a single human-style message in the following EXACT format: RESULT: followed by 1–5 short sentences addressed directly to the agent in second person that describe the current task stage, clearly state whether the last action was good or bad, and give a specific spatial recommendation for what to do next.
In the RESULT message:

- Speak directly to the agent using you and your.
- Do NOT mention images, numeric states, coordinates, or technical terms.
- Make it sound like natural human advice rather than a formal report.

You MUST follow the format.

Table 8: **Language label samples.** We show expert and suboptimal samples for all tasks generated with μ_g.

Task	Expert Samples	Suboptimal Samples
MAZE	You should go in the west direction. Furthermore, the action is good for the state. Plus, you should make a left here.	You should continue in the north direction. Yet, the action is not working for the state. Moreover, you should go slower. Also, you should take a left here.
PARKING	You should now gently park your car into the spot. Moreover, the action is helpful for the state. Furthermore, brake a bit. Also, steer a bit left.	You should first align your car to the parking spot. Even so, the action is not helpful for the state. Plus, slow down a bit. Moreover, steer a bit left.
BOX-CLOSE	You should fetch the lid. Furthermore, the action is useful for the state. Even so, you should move to the front calmly.	You need to raise the lid. Yet, the action is not good for the state. Plus, you should move to the front quickly. Moreover, you need to move to the top quickly.
BLOCKPUSH	Push the second block to the second target. Furthermore, the action is good for the state.	Move to the first block. Plus, the action is bad for the state. Plus, you should move to the front strongly.
PEGINSERT	Insert the peg into the hole. Furthermore, the action is good for the state because the peg is being pushed into the hole. Also, take a move to the bottom gently.	Go to the peg. Yet, the action is not right for the state, since the gripper moves away from the peg. Plus, make a move toward the right gently. Furthermore, make a move to the bottom softly.
HAMMER	You should swing the hammer to the nail. Furthermore, the action is helpful for the state.	You need to move to the hammer. But, the action is not useful for the state. Also, you need to move to the back.
RELOCATE	You need to go to the ball. Furthermore, the action is correct for the state.	You need to move to the ball. Even so, the action is not helpful for the state. Also, you need to move right. Furthermore, you should move toward the front. Plus, you need to reach down.

Table 9: **Language label variant samples.** We listed expert and suboptimal samples on task BOX-CLOSE generated with `o4-mini` and other variants in Section 5.4.

Task	Expert Samples	Suboptimal Samples
BOX-CLOSE (`o4-mini`)	You're in the lift stage and that last move was good. Now move the lid forward and to the right until it's centered above the box opening.	You're still trying to grab the lid, and lifting away was a bad move. Lower down and move forward and right to center on the handle. Then close your gripper to secure the lid.
BOX-CLOSE (`o4-mini` concise)	Lower gripper straight onto handle, open fully and center, clamp down firmly, then lift up.	Maintain the handle alignment, lower gripper, then move forward and close. Last move was unhelpful.
BOX-CLOSE (`o4-mini` μ_g-style)	You should center the lid. Moreover, the action is good for the state. Yet, lower your gripper straight onto the handle. Also, lift up firmly.	You need to lower the gripper onto the handle and shift forward. However, the action is suboptimal for the state. Moreover, line up before closing the grip.
BOX-CLOSE (Opaque tokens)	<H00_C00> <H01_C01> <H02_C00> <H03_C00> <H04_C01> <H05_C00>	<H00_C01> <H01_C00> <H02_C00> <H03_C00> <H04_C00> <H05_C01>
BOX-CLOSE (Named tokens)	<TASK_PROGRESS_C00> <ACTION_OPTIMALITY_C01> <GUIDANCE_GRIPPER_C00> <GUIDANCE_X_C00> <GUIDANCE_Z_C01> <GUIDANCE_Y_C00>	<TASK_PROGRESS_C01> <ACTION_OPTIMALITY_C00> <GUIDANCE_GRIPPER_C00> <GUIDANCE_X_C00> <GUIDANCE_Z_C00> <GUIDANCE_Y_C01>
BOX-CLOSE (Named tokens, variable length)	<TASK_PROGRESS_C00> <ACTION_OPTIMALITY_C01> <GUIDANCE_Z_C01>	<TASK_PROGRESS_C01> <ACTION_OPTIMALITY_C00> <GUIDANCE_X_C02> <GUIDANCE_Z_C02> <GUIDANCE_Y_C01>
BOX-CLOSE (μ_g-verbose)	It's great that you're trying to retrieve the lid. This action will indeed help with the current situation. Just remember to approach it calmly and smoothly from the front. Keep up the good work!	It seems like you're trying to get the lid, but at the moment, that action doesn't seem to be advancing our goal. Let's adjust our approach here. Instead, move towards the left more decisively, and then slowly inch forward.

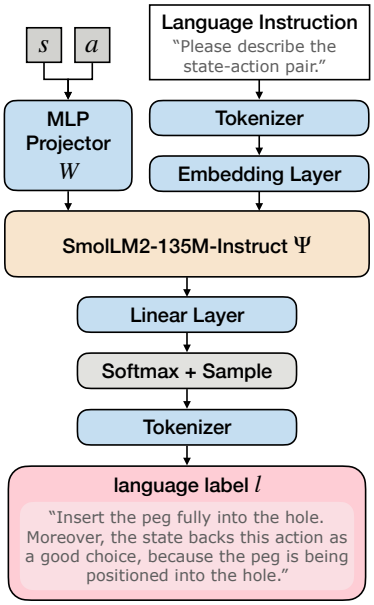


Figure 8: **Model Architecture of LLM-Captioner.** The state-action pair (s, a) is projected into the LLM's hidden space as a single token via the MLP projector W. This token, concatenated with the language instruction embeddings, serves as input to the decoder transformer Ψ. The final output layer applies a softmax function to sample tokens, generating the resulting language labels l.

task-specific labels following the <T>, <A>, and <M> structure described above. This choice allows us to study whether structured language supervision improves offline imitation learning without variation from model sampling, prompt design, or human annotator inconsistency. Although our generator is built on LLF-Bench (Cheng et al., 2024), it differs from the original benchmark in two important ways. First, we propose and implement the structured <T>, <A>, and <M> decomposition, whereas the original LLF-Bench feedback mainly indicates whether the behavior is good or bad and often contains larger language variations. Our structure is designed to make task progress, action quality, and corrective movement guidance explicit and consistent across the dataset. Second, we extend the generation pipeline beyond the original LLF-Bench domains to cover a broader set of embodied decision-making tasks, including MAZE, BLOCKPUSH, PEGINSERT, and Adroit hand tasks such as HAMMER and RELOCATE. The overall generation details of μ_g are provided in Figure 1.

We additionally evaluate alternative language label sources based on LLM/VLM generation in Section 5.4. Specifically, we collect language labels by prompting `o4-mini`, a multimodal vision-language reasoning model, to generate language labels from trajectory observations. The prompting template is provided in Figure 7, and generated examples are included in Table 8 and Table 9. These experiments allow us to examine whether the proposed framework can benefit from more flexible model-generated language, while the main results use μ_g labels for consistency.

D LLM-CAPTIONER DETAILS

In this section, we introduce the model architecture and the detailed training procedure of the LLM-Captioner μ_ϕ. Our objective is to enable end-to-end policy learning to optimize the language-critique imitation objective. To achieve this, the differentiable LLM-Captioner μ_ϕ must distill the language label generator μ_g and provide gradient signals that encourage the policy to match the objective.

The LLM-Captioner μ_ϕ consists of a pretrained transformer backbone Ψ—specifically SmolLM2-135M-Instruct (Allal et al., 2025)—and an MLP projector W that embeds state–action pairs into the hidden space of the transformer, as illustrated in Figure 8. Further details for the model architecture are provided in Section G.1. The resulting model defines a conditional distribution $\mu_\phi(\cdot \mid s_t, a_t)$ over language labels, where $\phi = \{\Psi, W\}$ denotes all trainable parameters.

We train the LLM-Captioner on the labeled general dataset $\mathcal{D}_G^{\text{lang}}$ using an instruction-tuning pipeline. Training proceeds in two stages: (1) we first update the MLP projector W while keeping the LLM backbone frozen, and (2) we finetune the entire LLM-Captioner μ_ϕ end-to-end, jointly updating both W and Ψ. In both stages, the objective is the cross-entropy loss in Equation (5):

$$\mathbb{E}_{(s_t,a_t,l_t)\sim\mathcal{D}_G^{\text{lang}}}\left[\ell_{\text{CE}}^{\phi}(l_t,s_t,a_t)\right]$$

The purpose of the finetuned LLM-Captioner μ_ϕ is to distill the generator μ_g for end-to-end policy optimization via gradient descent. Then, the LC-loss constructed by LLM-Captioner encourages the policy to produce actions that align with the expert policy's behavior with respect to the expert language-critique objective. We ablate backbone choices and finetuning modes in Section I.4.

E LC-DP DETAILS

In this section, we provide detailed derivations and implementation details for applying the proposed reweighted LC-loss $\tilde{\mathcal{L}}_{\text{LC}}$ to diffusion policies (Chi et al., 2023) of our method LC-DP. We first review the diffusion model backbone underlying diffusion policies, including the forward diffusion process and the reverse generative process, in Section E.1. Next, we describe the diffusion model training algorithm used for policy learning in Section E.2. Finally, in Section E.3, we derive the timestep-dependent reweighting factor ω^k for LC-DP from the diffusion training algorithm and provide experimental evidence justifying its design.

E.1 DIFFUSION MODEL: FORWARD AND REVERSE PROCESS

We first review the probabilistic formulation of diffusion models, which define paired forward and reverse Markov processes over a sequence of latent variables. As illustrated in Figure 9, diffusion models can be interpreted as latent variable models, where the data distribution over clean samples x^0 is augmented with a sequence of latent variables $\{x^k\}_{k=1}^K$. The joint distribution is defined through a reverse generative process, while the forward process serves as an approximate posterior.

Concretely, in the case of denoising diffusion probabilistic model (DDPM) (Ho et al., 2020), the forward diffusion process $q(x_{1:K} \mid x_0)$ is defined as a Markov chain that gradually adds Gaussian noise to the data according to a predefined variance schedule. The forward transitions take the form:

$$q(x^k \mid x^{k-1}) = \mathcal{N}\left(x^k;\, \sqrt{\alpha^k}x^{k-1}, (1-\alpha^k)\mathbf{I}\right), \tag{71}$$

where $\{\alpha^k\}_{k=1}^K$ schedules the noise magnitude at each timestep and $\bar{\alpha}^k = \prod_{i=1}^k \alpha^i$. Due to the linear-Gaussian structure of this process, the latent variable x^k admits a closed-form expression conditioned directly on the clean sample x^0:

$$x^k = \sqrt{\bar{\alpha}^k}x^0 + \sqrt{1-\bar{\alpha}^k}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0,\mathbf{I}). \tag{72}$$

As the timestep k increases, the signal-to-noise ratio decreases and x^k approaches an isotropic Gaussian distribution.

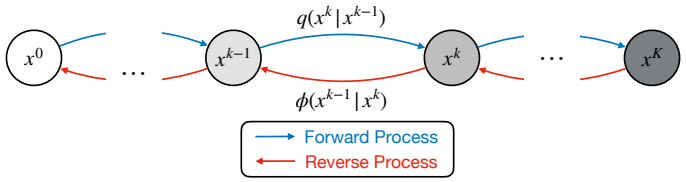


Figure 9: **Illustration of the probabilistic diffusion process.** Latent variables $x^1, \ldots, x^K$ are generated from the clean data sample x^0 through a forward diffusion process defined by a probabilistic Markov chain $q(x^k \mid x^{k-1})$. The reverse process ϕ samples the latent variables from the previous ones to reconstruct the original data sample x^0 via $\phi(x^{k-1} \mid x^k)$.

The reverse diffusion process aims to invert the forward diffusion process in order to generatively sample clean data. This is achieved by modeling the conditional distributions $\phi(x^{k-1} \mid x^k)$, such that samples drawn from the resulting reverse Markov chain approximate the original data distribution. However, directly parameterizing these reverse transitions is generally intractable.

To address this challenge, DDPMs adopt a denoising-based parameterization, in which a neural network is trained to predict the noise injected at each diffusion timestep. Specifically, a model $\epsilon_\theta(x_k, k)$ is optimized to estimate the noise ϵ, allowing recovery of the clean sample from its noisy latent representation. This denoising formulation leads to a simple and stable training objective based on noise prediction, while implicitly defining the reverse generative process.

Diffusion policies (Chi et al., 2023) adapt the denoising-based diffusion framework to continuous control by modeling a reverse generative process over actions. As illustrated in Figure 10, conditioned on the current state s_t, the diffusion policy samples a clean action $\hat{a}_t^0$ starting from Gaussian noise $a_t^K \sim \mathcal{N}(0, \mathbf{I})$ and iteratively applying the learned reverse diffusion process. At each diffusion timestep k, the policy predicts the noise component $\hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k)$ based on the current state s_t, the intermediate noisy action a_t^k, and the timestep index k, and uses this prediction to produce the next action sample according to the reverse transition:

$$\hat{a}_t^{k-1} = \frac{1}{\sqrt{\alpha^k}} \left(a_t^k - \frac{1-\alpha^k}{\sqrt{1-\bar{\alpha}^k}} \epsilon_\theta(\cdot \mid s_t, a_t^k, k) \right) + \sigma^k z, \tag{73}$$

where $z \sim \mathcal{N}(0, \mathbf{I})$ for $k > 1$ and $z = 0$ for $k = 1$. By iteratively applying this reverse update from $k = K$ to $k = 1$, the diffusion policy progressively denoises the action sample, yielding a clean action $\hat{a}_t^0$. This reverse-process formulation implicitly defines a conditional action distribution given the state.

E.2 Diffusion model: training algorithm

Training a diffusion policy reduces to learning a conditional denoising model from expert demonstrations. Given a state-action pair (s_t, a_t^0) sampled from the expert dataset $\mathcal{D}_E$, where a_t^0 denotes the clean expert action, the training procedure follows the forward–reverse diffusion formulation described in Section E.1.

Specifically, a diffusion timestep k is first sampled uniformly from $\{1, \dots, K\}$. Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is then drawn and applied to the clean action a_t^0 according to the forward diffusion process,

$$a_t^k = \sqrt{\bar{\alpha}^k} a_t^0 + \sqrt{1-\bar{\alpha}^k}\, \epsilon, \tag{74}$$

where $\bar{\alpha}^k = \prod_{i=1}^k \alpha^i$ denotes the cumulative noise schedule. This operation yields a noisy action a_t^k whose signal-to-noise ratio decreases as the timestep k increases.

The diffusion policy ϵ_θ is trained to predict the injected noise conditioned on the current state, the noisy action, and the diffusion timestep. In practice, the model receives (s_t, a_t^k, k) as input and outputs a noise estimate $\hat{\epsilon} \sim \epsilon_\theta(a_t^k, s_t, k)$. The training objective minimizes the mean squared error between the predicted and true noise,

$$\mathcal{L}_{\text{DP}} = \mathbb{E}_{(s_t, a_t^E) \sim \mathcal{D}_E,\, k,\, \epsilon \sim \mathcal{N}(0, \mathbf{I})} \left[\left\| \epsilon - \epsilon_\theta(\cdot \mid s_t, a_t^k, k) \right\|^2 \right]. \tag{75}$$

By training the policy to denoise actions at different noise levels, the model implicitly learns a conditional distribution over clean actions given the state. At inference time, action generation begins from a Gaussian noise sample and iteratively applies the learned reverse process to produce a clean action conditioned on the current state, as illustrated in Figure 10.

E.3 Reweighting factor derivation

To apply language label supervision to diffusion policies, we leverage one-step reconstruction to obtain a clean action estimate from the predicted noise and compute the LC-loss on the reconstructed action. Concretely, although the diffusion policy is trained to predict noise, the predicted noise $\hat{\epsilon} \sim \epsilon_\theta$ can be converted into a reconstructed clean action $\hat{a}_t^0$ via one-step reconstruction. This allows us to reuse the same LC-loss formulation as in feedforward policies.

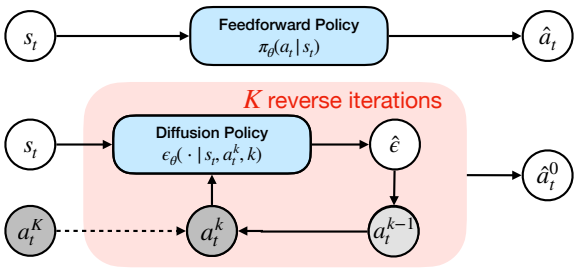


Figure 10: **Action sampling process of a diffusion policy.** Starting from Gaussian noise a_t^K, the diffusion policy iteratively denoises actions conditioned on the current state s_t, producing a clean action sample $\hat{a}_t^0$ through the learned reverse diffusion process. On the other hand, the feedforward policy predicts an action $\hat{a}_t$ with a single-step inference conditioned on the state.

A straightforward approach is to directly apply the LC-loss on the reconstructed action without any additional design. However, as shown in Figure 12b, this naive strategy yields limited improvement compared to feedforward policies, suggesting a mismatch between diffusion training dynamics and the action-level supervision induced by the LC-loss.

To better understand this behavior, we analyze the properties of one-step reconstruction across diffusion timesteps. Empirically, we observe that the quality of the reconstructed action $\hat{a}_t^0$ strongly depends on the diffusion timestep k. Reconstructions from small diffusion steps (small k) are generally accurate, while reconstructions from large diffusion steps (large k) are substantially noisier. This introduces an inherent imbalance: errors from large-k steps dominate the action-level objective, while informative signals from small-k steps are underrepresented.

The standard diffusion policy training objective optimizes noise prediction via the mean squared error

$$\mathcal{L}_{\text{DP}} = \mathbb{E}_{(s_t, a_t^E)\sim\mathcal{D}_E, \epsilon\sim\mathcal{N}(0,\mathbf{I}), k}\left[\left\|\epsilon - \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right\|^2\right] \tag{76}$$

as defined in Equation (11). In contrast, when applying supervision on reconstructed actions, the objective becomes an action-level distance:

$$\mathcal{L}^a_{\text{DP}} = \mathbb{E}_{(s_t, a_t^E)\sim\mathcal{D}_E, \epsilon\sim\mathcal{N}(0,\mathbf{I}), k}\left[\left\|a_t^0 - \hat{a}_t^0\right\|^2\right] \tag{77}$$

where $\hat{a}_t^0$ is obtained via one-step reconstruction in Equation (12). Expanding this action-level objective yields:

$$\left\|a_t^0 - \hat{a}_t^0\right\|^2 \tag{78}$$

$$= \left\|a_t^0 - \left(\frac{1}{\sqrt{\bar{\alpha}^k}} a_t^k - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}} \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right)\right\|^2 \tag{79}$$

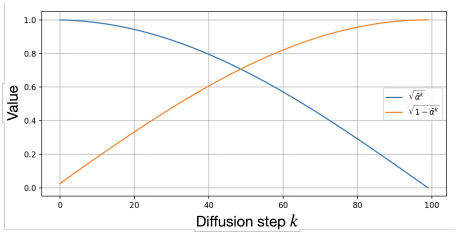
$$= \left\|a_t^0 - \left(\frac{1}{\sqrt{\bar{\alpha}^k}} \left(\sqrt{\bar{\alpha}^k} a_t^0 + \sqrt{1-\bar{\alpha}^k}\epsilon\right) - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}} \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right)\right\|^2 \tag{80}$$

$$= \left\|a_t^0 - \left(a_t^0 + \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}}\epsilon - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}} \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right)\right\|^2 \tag{81}$$

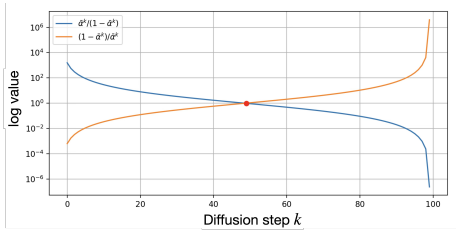
$$= \left\|a_t^0 - a_t^0 + \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}} \left(\epsilon - \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right)\right\|^2 \tag{82}$$

$$= \frac{1-\bar{\alpha}^k}{\bar{\alpha}^k} \left\|\epsilon - \epsilon_\theta(\cdot \mid s_t, a_t^k, k)\right\|^2 \tag{83}$$

showing that the action reconstruction error is equivalent to the original noise-prediction error weighted by a timestep-dependent factor $\frac{1-\bar{\alpha}^k}{\bar{\alpha}^k}$.



(a) Diffusion schedule components over timesteps k: $\sqrt{\bar{\alpha}^k}$ (signal term) and $\sqrt{1-\bar{\alpha}^k}$ (noise term).



(b) k-dependent weighting induced by one-step reconstruction: $\frac{1-\bar{\alpha}^k}{\bar{\alpha}^k}$, and its reciprocal reweighting factor ω^k.

Figure 11: **Diffusion schedules and k-dependent reweighting.**

As illustrated in Figure 11, this weighting scheme disproportionately amplifies contributions from large diffusion k-steps at the expense of small-k steps. Directly optimizing this action-level objective introduces unstable training loss convergence; as seen in Figure 12a, the loss exhibits stronger fluctuation, leading to performance degradation. While the resulting policy remains functional, it ultimately underperforms compared to a standard diffusion policy trained with the original noise-prediction loss (Figure 12b).

The above analysis reveals that the action-level objective $\mathcal{L}^a_{\text{DP}}$ implicitly reweights the original diffusion objective $\mathcal{L}_{\text{DP}}$. To recover the balanced training behavior of the standard diffusion loss with one-step reconstructed actions $\hat{a}^0_t$, we apply the reciprocal of this k-dependent weight $\frac{1-\bar{\alpha}^k}{\bar{\alpha}^k}$. Formally, we define the reweighting factor

$$\omega^k := \frac{\bar{\alpha}^k}{1-\bar{\alpha}^k}. \tag{84}$$

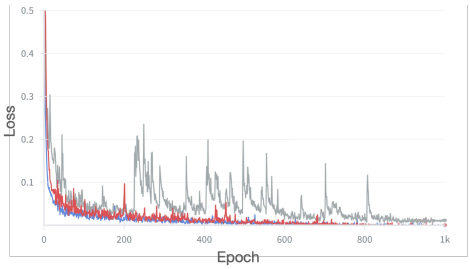
Applying this factor to the action-level objective yields the reweighted action loss

$$\tilde{\mathcal{L}}^a_{\text{DP}} = \mathbb{E}_{(s_t,a^E_t)\sim\mathcal{D}_E,\epsilon\sim\mathcal{N}(0,\mathbf{I}),k}\left[\frac{\bar{\alpha}^k}{1-\bar{\alpha}^k}\left\|a^0_t-\hat{a}^0_t\right\|^2\right] \tag{85}$$

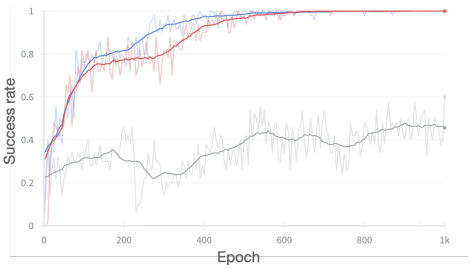
$$= \mathbb{E}_{(s_t,a^E_t)\sim\mathcal{D}_E,\epsilon\sim\mathcal{N}(0,\mathbf{I}),k}\left[\left\|\epsilon-\epsilon_\theta(\cdot\mid s_t,a^k_t,k)\right\|^2\right] \tag{86}$$

$$= \mathcal{L}_{\text{DP}} \tag{87}$$

showing that the reweighted action objective $\tilde{\mathcal{L}}^a_{\text{DP}}$ is exactly equivalent to the original noise-prediction objective $\mathcal{L}_{\text{DP}}$.



(a) Training MSE loss curves



(b) Success rate curves

Figure 12: **Diffusion policy training with different objectives.** Red: standard noise-prediction loss $\mathcal{L}_{\text{DP}}$; gray: unweighted action-level loss $\mathcal{L}^a_{\text{DP}}$; blue: reweighted action-level loss $\tilde{\mathcal{L}}^a_{\text{DP}}$. **Left**: training MSE, where $\mathcal{L}^a_{\text{DP}}$ exhibits large fluctuations due to timestep k imbalance, while the reweighted and original objectives converge stably. **Right**: success rate (evaluated every 5 epochs), showing that reweighting restores near-optimal performance.

We empirically validate this equivalence by training diffusion policies under three objectives: the standard noise-prediction loss $\mathcal{L}_{\text{DP}}$ (red), the unweighted action-level loss $\mathcal{L}^a_{\text{DP}}$ (gray), and the

reweighted action-level loss $\tilde{\mathcal{L}}^a_{\text{DP}}$ (blue). As shown in Figure 12a and Figure 12b, applying the reweighting factor ω^k aligns both the training loss trajectories and the final performance with those of $\mathcal{L}_{\text{DP}}$, whereas the unweighted action-level objective exhibits noticeable discrepancies. These results confirm the effect and necessity of the proposed reweighting factor ω^k.

With this derivation, we apply the same reweighting factor ω^k to the LC-loss when training diffusion policies. Intuitively, this reweighting compensates for the k-dependent reconstruction bias: without reweighting, inaccuracies at large diffusion steps are overemphasized, while informative gradients from small diffusion steps, where the signal-to-noise ratio is high, are diminished.

This behavior is visualized in Figure 11a and Figure 11b. The original k-dependent weight $\frac{1-\bar{\alpha}^k}{\bar{\alpha}^k}$ (orange curve in **??**) suppresses contributions from small diffusion steps while amplifying those from large steps. In contrast, its reciprocal ω^k (blue curve in Figure 11a) restores balance by emphasizing small-k steps—where the signal-to-noise ratio is high—and down-weighting large-k steps.

F ENVIRONMENT AND TASK DETAILS

We list the details of the task environments of Section 5 here. In Table 10, we list the details of expert and general datasets we used for experiments. It comprises the number of trajectories, expert steps, general steps, and the collection methods. We also provide the environment names, version numbers, the dimensions of state spaces, and action spaces of each task in Table 11.

MAZE. A 2D navigation task from Fu et al. (2020) in which a point agent learns to navigate from an initial position to a goal position within the medium maze. The 2-DoF agent controls its motion by iteratively applying linear forces along the x and y directions. The 6D states include the agent's two-dimensional current position, velocity, and the goal position. The initial and goal positions of the agent are randomly sampled when an episode is initialized. The maximum episode length for this task is set at 250; termination occurs if the goal is reached earlier.

PARKING. An image-based continuous control driving task from Leurent (2018), where the vehicle is asked to park in the designated target spot within a parking lot. The 2-DoF agent navigates the car by controlling the acceleration and steering angle. The original 18D observation space includes the current position, the longitudinal velocity, the heading angle of the car, along with the achieved goal state and the desired target configuration. In our setting, we use the RGB rendered image as observation for the policies. Due to the partial observability, we stack 2 consecutive RGB images as input and output a 4-step action chunk. The maximum episode length for this task is set at 80; termination occurs if the vehicle reaches the target position earlier.

BOX-CLOSE. A manipulation task from Metaworld (Yu et al., 2019) in which the agent should grasp the cover and close the box with it. The robot arm should precisely align the position of the cover, grasp the handle, hold it towards the opened box, and put it in a closed position. The agent controls the position of the end-effector and the gripper actuation via a 4D continuous action space. The 9D observation states provide the x, y, z positions of the robot arm, the manipulation lid, and the target. The maximum episode length for this task is set at 30, and termination occurs if the box is closed with the cover earlier. We evaluate BOX-CLOSE under two settings that differ only in the sampling range of the lid's initial position. Because full-data performance approached the success-rate ceiling under the original range, we reevaluate saved policy checkpoints using an expanded range without retraining. Specifically, the expanded setting changes the lid's initial y-coordinate range from $[0.50, 0.55]$ to $[0.48, 0.55]$, while keeping $x \in [-0.05, 0.05]$ and $z = 0.02$ unchanged. This expanded range is used for Table 1 and all BOX-CLOSE ablations. The mixed-quality and data-efficiency experiments in Table 2 retain the original range for every condition, including the full-data reference at $\rho = 100\%$, to study demonstration degradation at a fixed evaluation difficulty. Consequently, the full-data results differ between Table 1 and Table 2 because they are measured under different evaluation distributions. All methods within each experiment use the same evaluation setting.

BLOCKPUSH. A multistage table-top manipulation task from Florence et al. (2021) where a 6-DoF robot arm should push two blocks into two target zones in any order. It is a multimodal task that allows a robot arm to first move one randomly selected block to one target zone, then move the second block to the other zone. The end-effector is controlled via absolute x and y pose on the 2D plane. The 16D observation space encapsulates the full system state, including the current end-effector pose, the position and orientation of both blocks and target zones. The maximum episode length for this task is set at 350, and termination occurs if two blocks are pushed into their corresponding target zones.

Table 10: **Dataset.** Detailed information on the expert dataset and the general dataset for each task.

Task	# of expert trajectories	# of expert steps	# of general steps	Algorithm/Retrieved Sources
MAZE	200	26 037	153 177	D4RL controller (Fu et al., 2020)
PARKING	100	2046	14 871	SAC-trained policy expert (Haarnoja et al., 2018)
BOX-CLOSE	500	4214	76 031	Metaworld controller (Yu et al., 2019)
BLOCKPUSH	500	37 994	301 876	default controller (Florence et al., 2021)
PEGINSERT	500	18 589	246 576	PPO-trained policy (Schulman et al., 2017)
HAMMER	10	439	24 752	DAPG-trained policy (Rajeswaran et al., 2018)
RELOCATE	20	1118	30 136	DAPG-trained policy (Rajeswaran et al., 2018)

Table 11: **Environments.** Detailed environment setting of each task.

Task	Environment	ID	Observation space	Action space
MAZE	D4RL (Fu et al., 2020)	pointmaze-maze2d-medium-v0	6	2
PARKING	Highway-Env (Leurent, 2018)	highway-parking-v0	$2 \times 3 \times 256 \times 256$	4×2
BOX-CLOSE	Metaworld (Yu et al., 2019)	metaworld-box-close-v2	9	4
BLOCKPUSH	BlockPush (Florence et al., 2021)	blockpushing-BlockPushMultimodal-v0	16	2
PEGINSERT	Maniskill (Tao et al., 2025)	maniskill-PegInsertionSide-v1	43	8
HAMMER	Gymnasium-Robotics (de Lazcano et al., 2024)	adroit-hand-hammer-v1	46	26
RELOCATE	Gymnasium-Robotics (de Lazcano et al., 2024)	adroit-hand-relocate-v1	39	30

PEGINSERT. A high-precision manipulation task from ManiSkill-3 (Tao et al., 2025) where a 7-DoF robot arm equipped with a gripper must grasp a peg from a table and insert it into a horizontal box hole. This task poses accurate control challenges due to the tight clearance between the peg and the hole. The agent operates within a 43D state space, which comprises joint positions and velocities of the robot arm, as well as the poses of both the peg and the target hole. The maximum episode length for this task is set at 100, and termination occurs if the end of the peg is successfully inserted within 0.015m of the center of the box.

HAMMER. A high-dimensional dexterous manipulation task from Gymnasium-Robotics (de Lazcano et al., 2024) that requires a 28-DoF system, consisting of a 24-DoF ShadowHand and a 4-DoF arm, to pick up a hammer and drive a nail into a board. The environment features a randomized nail position with dry friction capable of absorbing up to 15N of force, requiring the agent to exert significant physical impact. The agent operates within a 26D continuous action space representing the scaled absolute angular positions of the joints, including the arm, wrist, and all finger segments. The 46D observation space tracks joint angles, palm and nail positions, hammer pose and velocities, nail displacement, and impact force. Success is achieved when the entire length of the nail is driven into the board, and the episode is truncated after a maximum of 200 steps.

RELOCATE. A complex manipulation task from Gymnasium-Robotics (de Lazcano et al., 2024) that utilizes a 30-DoF system, comprising a 24-DoF ShadowHand mounted on a 6-DoF robotic arm. The objective is to pick up a blue ball from a tabletop and move it to a randomly initialized green target in the 3D workspace. The agent utilizes a 30D action space to control arm translation, orientation, and hand joint positions. The 39D observation space tracks joint states and relative 3D vectors between the palm, ball, and target. This task is challenging due to the randomization of both the object and the target, requiring the agent to master a multi-stage behavior: approaching the object, achieving a stable grasp, and precisely transporting the object through free space to an aerial target. The maximum episode length for this task is set at 200, and termination occurs if the blue ball is successfully moved within an epsilon-ball of the target.

G TRAINING DETAILS

Our implementation is built on top of the Diffusion Policy codebase (Chi et al., 2023). In this section, we describe the model architectures used for all baselines and our proposed methods, along with the corresponding training hyperparameters and computational resources.

G.1 MODEL ARCHITECTURES

Table 12 and 13 summarize the detailed architectural configurations for all baselines.

Table 12: **Model architectures (Part I):** MAZE, PARKING, and BOX-CLOSE.

Method	Models	MAZE	PARKING	BOX-CLOSE
BC	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
DWBC	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
	Discriminator Hidden Dimension	256	512	256
DemoDICE	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
	Discriminator Network Type	MLP	MLP	MLP
	Discriminator Hidden Dimension	[256, 256]	[256, 256]	[256, 256]
	Discriminator Activation Function	ReLU	ReLU	ReLU
	Nu Network Type	MLP	MLP	MLP
	Nu Hidden Dimension	[256, 256]	[256, 256]	[256, 256]
	Nu Activation Function	ReLU	ReLU	ReLU
ILID	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
	Discriminator Network Type	MLP	MLP	MLP
	Discriminator Hidden Dimension	[256]	[256]	[256]
	Discriminator Activation Function	ReLU	ReLU	ReLU
LC-BC-scalar	Y_ϕ Model Type	MLP	MLP	MLP
	Y_ϕ Hidden Dimension	[256, 256, 256]	[256]	[256, 256, 256]
	Y_ϕ Activation Function	ReLU	ReLU	ReLU
	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
LC-BC-classifier	C_ϕ Model Type	MLP	MLP	MLP
	C_ϕ Hidden Dimension	[128, 128]	[256]	[256, 256]
	C_ϕ Activation Function	ReLU	ReLU	ReLU
	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
LC-BC (Ours)	Network Type	MLP	MLP	MLP
	Hidden Dimension	[256, 256, 256, 256]	[512, 512, 512]	[256, 256]
	Activation Function	tanh	ReLU	ReLU
DP	Network Type	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LPB-Offline	Dynamics Network Type	MLP	MLP	MLP
	Dynamics Hidden Dimension	[256, 256]	[256]	[256, 256]
	Dynamics Activation Function	ReLU	ReLU	ReLU
LC-DP-scalar	Y_ϕ Model Type	MLP	MLP	MLP
	Y_ϕ Hidden Dimension	[64]	[256]	[128, 128]
	Y_ϕ Activation Function	ReLU	ReLU	ReLU
	Network Type	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LC-DP-classifier	C_ϕ Model Type	MLP	MLP	MLP
	C_ϕ Hidden Dimension	[256, 256]	[256, 256]	[256, 256]
	C_ϕ Activation Function	ReLU	ReLU	ReLU
	Network Type	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LC-DP (Ours)	Network Type	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]

Table 13: **Model architectures (Part II):** BLOCKPUSH, PEGINSERT, HAMMER, RELOCATE

Method	Models	BLOCKPUSH	PEGINSERT	HAMMER	RELOCATE
BC	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
DWBC	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
	Discriminator Hidden Dimension	256	256	256	256
DemoDICE	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
	Discriminator Network Type	MLP	MLP	MLP	MLP
	Discriminator Hidden Dimension	[128, 128]	[256, 256]	[256, 256]	[256, 256]
	Discriminator Activation Function	ReLU	ReLU	ReLU	ReLU
	Nu Network Type	MLP	MLP	MLP	MLP
	Nu Hidden Dimension	[128, 128]	[256, 256]	[256, 256]	[256, 256]
	Nu Activation Function	ReLU	ReLU	ReLU	ReLU
ILID	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
	Discriminator Network Type	MLP	MLP	MLP	MLP
	Discriminator Hidden Dimension	[256]	[256]	[256]	[256]
	Discriminator Activation Function	ReLU	ReLU	ReLU	ReLU
LC-BC-scalar	Y_ϕ Model Type	MLP	MLP	MLP	MLP
	Y_ϕ Hidden Dimension	[256, 256]	[256, 256]	[128]	[128]
	Y_ϕ Activation Function	ReLU	ReLU	ReLU	ReLU
	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
LC-BC-classifier	C_ϕ Model Type	MLP	MLP	MLP	MLP
	C_ϕ Hidden Dimension	[128]	[256, 256]	[256]	[256]
	C_ϕ Activation Function	ReLU	ReLU	ReLU	ReLU
	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
LC-BC (Ours)	Network Type	Unet	MLP	MLP	MLP
	Hidden Dimension	[256, 512, 1024]	[256, 256]	[256, 256]	[256, 256]
	Activation Function	-	tanh	ReLU	ReLU
DP	Network Type	Unet	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LPB-Offline	Dynamics Network Type	MLP	MLP	MLP	MLP
	Dynamics Hidden Dimension	[256, 256]	[256, 256]	[256, 256]	[256, 256]
	Dynamics Activation Function	ReLU	ReLU	ReLU	ReLU
LC-DP-scalar	Y_ϕ Model Type	MLP	MLP	MLP	MLP
	Y_ϕ Hidden Dimension	[64]	[128]	[128]	[64]
	Y_ϕ Activation Function	ReLU	ReLU	ReLU	ReLU
	Network Type	Unet	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LC-DP-classifier	C_ϕ Model Type	MLP	MLP	MLP	MLP
	C_ϕ Hidden Dimension	[128]	[256, 256]	[128]	[128]
	C_ϕ Activation Function	ReLU	ReLU	ReLU	ReLU
	Network Type	Unet	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]
LC-DP (Ours)	Network Type	Unet	Unet	Unet	Unet
	Hidden Dimension	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]	[256, 512, 1024]

G.2 HYPERPARAMETERS

Table 14 lists the training hyperparameters used for all baselines and our proposed methods. Unless otherwise specified, we set the gradient accumulation step to 1. For LC-BC and LC-DP, we explicitly increase the gradient accumulation steps to accommodate the additional memory cost introduced by LLM-based computations, allowing us to use smaller per-step batch sizes within our computational constraints.

Table 14: **Hyperparameters.** The overview of the hyperparameters used for all the methods in every task. We abbreviate "Gradient Accumulation" as "Grad Acc." and "Finetune" as "FT" in this table.

Method	Hyperparameter	MAZE	PARKING	BOX-CLOSE	BLOCKPUSH	PEGINSERT	HAMMER	RELOCATE
BC	Policy Batch Size	128	128	128	256	128	128	128
	Policy Learning Rate	1e-4	1e-3	1e-2	2e-4	1e-3	1e-3	1e-3
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
DWBC	Policy Batch Size	128	128	128	256	128	128	128
	Policy Learning Rate	1e-4	1e-4	3e-4	2e-4	1e-4	3e-4	3e-5
	α	7.5	10	7.5	15	12	7.5	10
	Discriminator Hidden Dim. logpi	128	128	128	128	128	128	128
	Discriminator Hidden Dim. sa	128	128	128	128	128	128	128
	Policy Epochs	1001	1001	1001	401	401	1001	1001
DemoDICE	Policy Batch Size	128	128	128	256	128	128	128
	Policy Learning Rate	1e-4	1e-4	5e-4	1e-4	1e-3	1e-4	5e-5
	α	0.1	0.1	0.05	0.01	0.1	0.1	0.1
	Policy Epochs	1001	1001	1001	1001	401	1001	1001
ILID	Policy Batch Size	128	128	128	256	128	128	128
	Policy Learning Rate	1e-4	1e-4	1e-2	2e-4	1e-3	1e-3	1e-3
	Discriminator Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Rollback steps	5	5	5	5	3	5	3
	Rollback Decay Rate	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	Select Threshold	0.89	0.89	0.89	0.89	0.89	0.89	0.89
LC-BC-scalar	Scalar Predictor Model Batch Size	256	128	256	512	512	256	256
	Scalar Predictor Model Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Scalar Predictor Model Epochs	2001	1000	1001	501	501	200	500
	Policy Batch Size	128	128	128	256	128	128	128
	Policy Scalar Loss Weight	1e-2	1e-2	1e-3	1e-4	1e-2	1e-2	1e-2
	Policy Learning Rate	1e-4	1e-3	1e-2	2e-4	1e-3	1e-3	1e-3
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
LC-BC-classifier	Classifier Model Batch Size	256	128	256	512	256	256	256
	Classifier Model Learning Rate	1e-4	1e-4	1e-4	5e-5	1e-4	1e-4	1e-4
	Classifier Model Epochs	1000	1000	1000	1000	1000	1000	1000
	Classifier Heads	6	7	6	4	10	7	6
	Classifier Classes	8	3	3	4	11	4	3
	Policy Batch Size	128	128	128	256	128	128	128
	Policy Class Loss Weight	1e-4	1e-4	1e-4	2e-4	1e-4	1e-1	1e-4
	Policy Learning Rate	1e-4	1e-4	1e-2	1e-3	1e-3	1e-2	1e-3
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
LC-BC (Ours)	MLP Projector Batch Size	128	-	128	128	64	128	128
	MLP Projector Grad Acc.	4	-	2	4	8	2	2
	MLP Projector Learning Rate	1e-4	-	1e-4	1e-4	1e-4	1e-4	1e-4
	MLP Projector Epochs	100	-	100	100	100	100	100
	μ_ϕ FT Batch Size	128	32	128	128	64	128	128
	μ_ϕ FT Grad Acc.	1	8	1	1	2	1	1
	LLM-Captioner FT Learning Rate	1e-5	1e-4	1e-4	1e-5	1e-5	1e-4	1e-4
	MLP Projector FT Learning Rate	1e-5	1e-4	1e-4	1e-5	1e-5	5e-5	5e-5
	μ_ϕ FT Epochs	10	10	10	10	10	10	10
	Policy Batch Size	16	16	16	16	16	32	16
	Policy Grad Acc.	8	8	8	16	8	4	8
	LC-loss Weight	1e-2	1e-3	1e-2	1e-4	1e-2	1e-2	1e-2
	Policy Learning Rate	1e-4	1e-3	1e-2	2e-4	1e-3	1e-3	1e-3
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
DP	Policy Batch Size	256	128	256	256	256	256	256
	Policy Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
LPB-Offline	Dynamic model Batch Size	256	128	256	256	256	256	256
	Dynamic model Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-5	1e-4	1e-4
	Dynamic model Epochs	1001	1001	1001	1001	1001	1001	1001
LC-DP-scalar	Scalar Predictor Model Batch Size	256	128	256	256	512	256	256
	Scalar Predictor Model Learning Rate	1e-6	1e-4	1e-5	1e-6	1e-6	1e-4	1e-4
	Scalar Predictor Model Epochs	1000	1000	500	100	500	200	500
	Policy Batch Size	256	128	256	256	256	256	256
	Policy Scalar Loss Weight	1e-3	1e-4	1e-3	1e-4	1e-3	1e-2	1e-2
	Policy Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001

Table 14: Hyperparameters (Continued)

Method	Hyperparameter	MAZE	PARKING	BOX-CLOSE	BLOCKPUSH	PEGINSERT	HAMMER	RELOCATE
LC-DP-classifier	Classifier Model Batch Size	256	128	256	512	256	256	256
	Classifier Model Learning Rate	1e-4	1e-4	1e-4	1e-5	1e-4	1e-4	1e-4
	Classifier Model Epochs	1000	1000	1001	1000	1000	1000	1000
	Classifier Heads	6	7	6	4	10	7	6
	Classifier Classes	8	3	3	4	11	4	3
	Policy Batch Size	256	128	256	256	256	256	256
	Policy Class Loss Weight	1e-4	1e-3	1e-4	1e-3	1e-4	1e-4	1e-3
	Policy Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001
LC-DP (Ours)	MLP Projector Batch Size	128	-	128	128	64	128	128
	MLP Projector Grad Acc.	4	-	2	4	8	2	2
	MLP Projector Learning Rate	1e-4	-	1e-4	1e-4	1e-4	1e-4	1e-4
	MLP Projector Epochs	100	-	100	100	100	100	100
	μ_ϕ FT Batch Size	128	32	128	128	64	128	128
	μ_ϕ FT Grad Acc.	1	8	1	1	2	1	1
	LLM-Captioner FT Learning Rate	1e-5	1e-4	5e-5	1e-5	1e-5	1e-4	1e-4
	MLP Projector FT Learning Rate	1e-5	1e-4	5e-5	1e-5	1e-5	5e-5	5e-5
	μ_ϕ FT Epochs	10	10	10	10	10	10	10
	Policy Batch Size	16	16	16	16	16	16	16
	Policy Grad Acc.	16	8	16	16	16	16	16
	LC-loss Weight	1e-2	1e-4	1e-4	1e-5	1e-2	5e-2	1e-3
	Policy Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
	Policy Epochs	1001	1001	1001	1001	1001	1001	1001

G.3 COMPUTATIONAL RESOURCES

Table 15 reports the computational resources we have used for our experiments, including the workstation configurations.

Table 15: **Computation resources.**

Workstation	CPU	GPU	RAM
1	Intel Xeon W-2255	NVIDIA GeForce RTX 3090, NVIDIA GeForce RTX 3080	125G
2	Intel Xeon W-2255	NVIDIA GeForce RTX 3090, NVIDIA GeForce RTX 3080	125G
3	Intel Xeon W-2255	NVIDIA GeForce RTX 3080 Ti $\times$ 2	125G
4	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti $\times$ 2	125G
5	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti $\times$ 2	125G
6	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti $\times$ 2	125G
7	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti $\times$ 2	125G
8	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti $\times$ 2	125G
9	Intel Xeon w7-2475X	NVIDIA GeForce RTX 4090 $\times$ 2	125G
10	Intel Xeon w7-2475X	NVIDIA GeForce RTX 4090 $\times$ 2	125G
11	Intel Xeon w5-2455X	NVIDIA RTX A6000 $\times$ 2	125G
12	Intel Xeon Platinum 8480+	NVIDIA H100 $\times$ 2	500G

G.4 BASELINE RUNTIMES

The runtime comparison in BOX-CLOSE shows that computational overhead is a limitation of our method. As shown in Table 16, LC-BC and LC-DP require approximately 7.9 and 12.9 hours of policy training, respectively, compared with 21.5 minutes for BC and 3.3 hours for DP. Both methods additionally require approximately 24.8 hours to train the LLM-Captioner guidance module. In contrast, the categorical variants incur substantially lower costs for both policy and auxiliary module training. These results highlight the computational cost of incorporating language-based guidance: its potential performance benefits come with substantially longer training times, which may limit its practicality under constrained computational budgets. Note that our approach adds no inference overhead; the language-critique supervision is used only during training, and all baselines incur the same inference-time overhead within each policy class.

H BASELINE DETAILS

We provide details of the LC-BC and LC-DP variants. Given language-labeled datasets, we ask what the best way is to exploit the data and the language label learning signal. To this end, we develop

Table 16: Training runtime comparison on BOX-CLOSE. We report the runtime of all baselines on the BOX-CLOSE task. Each baseline is trained for 1001 epochs on a single RTX 4090 GPU under identical system load. Runtimes are reported in seconds. As the results show, a limitation of our method is its substantially larger training overhead, which stems from the LLM-Captioner module.

METHOD	POLICY TRAINING TIME	EXTRA MODULE TRAINING TIME
BC	1289S	-
DWBC	3481S	-
DEMODICE	5332S	-
ILID	2763S	-
LC-BC-SCALAR	1248S	1081S
LC-BC-CLASSIFIER	1369S	1865S
LC-BC (OURS)	28425S	89557S
DP	11760S	-
LPB-OFFLINE	11760S	8269S
LC-DP-SCALAR	9512S	1336S
LC-DP-CLASSIFIER	10462S	1922S
LC-DP (OURS)	46485S	89017S

LC-BC-scalar and LC-DP-scalar, which are trained on scalar signals converted from language labels, and LC-BC-classifier and LC-DP-classifier, which are trained on multi-head categorical signals converted from language labels.

H.1 LC-BC-SCALAR AND LC-DP-SCALAR

We design two scalar-based ablation baselines, LC-BC-scalar and LC-DP-scalar, as scalar signal counterparts to LC-BC and LC-DP, respectively. These baselines test whether the gains of language-critique imitation arise merely from learning an auxiliary scalar signal model, or from preserving the structured information contained in language label supervision. Both baselines follow the same overall pipeline as our language-based methods: first, a predictor is trained on the general dataset; second, the scalar predictor is frozen and used to provide auxiliary supervision during policy learning on the expert dataset. Thus, LC-BC-scalar and LC-DP-scalar replace the language model μ_ϕ and the LC-loss with a scalar predictor model y_ϕ and a scalar-critique loss.

Scalar dataset conversion. Given the language-labelled datasets $\mathcal{D}_E^{\text{lang}}$ and $\mathcal{D}_G^{\text{lang}}$, we first convert the language labels into scalar signals using a counting-based procedure. Following Cao et al. (2024), we prompt `Mistral-7B-Instruct-v0.3` to identify explicit positive and negative evaluative statements in each label, and define the raw signal as the number of positive statements minus the number of negative statements. The raw signals are then normalized to $[0, 1]$. The prompt template is given in Figure 13, and we denote the resulting datasets by $\mathcal{D}_E^y$ and $\mathcal{D}_G^y$.

Scalar predictor model. Given a scalar-labeled general dataset $\mathcal{D}_G^y = \{(s_t, a_t, y_t)\}$, we train a scalar predictor model Y_ϕ to predict the scalar signal associated with each state-action pair. The predictor model is optimized with a mean squared error objective:

$$\mathcal{L}_Y^\phi = \mathbb{E}_{(s_t,a_t,y_t)\sim\mathcal{D}_G^y}\left[\|Y_\phi(s_t, a_t) - y_t\|_2^2\right]. \tag{88}$$

After training, Y_ϕ is frozen and used for downstream policy optimization.

Scalar-critique loss. Given the frozen predictor model Y_ϕ, we define a scalar-based analogue of the LC-loss on the expert dataset $\mathcal{D}_E^y = \{(s_t, a_t^E, y_t)\}$. For a policy action $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$, we define the policy scalar-prediction loss and the expert scalar-prediction loss as

$$\mathcal{L}_Y(\pi_\theta) = \mathbb{E}_{(s_t,y_t)\sim\mathcal{D}_E^y,\hat{a}_t\sim\pi_\theta(\cdot|s_t)}\left[\|Y_\phi(s_t, \hat{a}_t) - y_t\|_2^2\right], \tag{89}$$

$$\mathcal{L}_Y(\pi_E) = \mathbb{E}_{(s_t,a_t^E,y_t)\sim\mathcal{D}_E^y}\left[\left\|Y_\phi(s_t, a_t^E) - y_t\right\|_2^2\right]. \tag{90}$$

Figure 13: **Prompting for language-to-scalar conversion.** The prompt we use to generate scalar signals from language labels with `Mistral-7B-Instruct-v0.3`.

Prompt for language-to-scalar conversion.

You are given a natural-language description that may contain multiple short statements evaluating an action. Classify each *atomic* evaluation as Positive or Negative, then compute: reward = (# of Positive statements) - (# of Negative statements)
Description: desc
Instructions:

- Count only explicit evaluative statements. Ignore neutral narration.
- Treat synonyms like "good/correct/helpful/effective/safe" as Positive.
- Treat synonyms like "bad/wrong/unhelpful/ineffective/unsafe" as Negative.
- If one sentence has both a positive and a negative clause, split and count each.
- Do not guess beyond the text; if unclear, do not count it.
- The reward must be an integer (can be negative).

Output format:

- First list the counts in one short line: pos=<int>, neg=<int>
- On the very last line, output exactly: Final reward: <integer>

The scalar-critique loss clips the policy loss at the expert loss:

$$\mathcal{L}_{\text{SC}}(\pi_\theta, \pi_E) = \left[\mathcal{L}_Y(\pi_\theta) - \text{sg}(\mathcal{L}_Y(\pi_E))\right]_+, \tag{91}$$

where $[x]_+ = \max(x, 0)$ and $\text{sg}(\cdot)$ denotes stop-gradient. This objective penalizes policy actions whose predicted scalar signals are less consistent with the expert scalar signals than the corresponding expert actions, and the penalty vanishes once the policy matches the expert under the trained, frozen predictor model.

Scalar-critique behavior cloning. LC-BC-scalar applies the scalar-critique loss to a feedforward behavior cloning policy. The standard behavior cloning objective is

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{(s_t, a_t^E)\sim\mathcal{D}_E, \hat{a}_t\sim\pi_\theta(\cdot|s_t)}\left[\left\|a_t^E - \hat{a}_t\right\|_2^2\right]. \tag{92}$$

LC-BC-scalar jointly optimizes the imitation loss and the scalar-critique loss:

$$\mathcal{L}_{\text{BC}} + \lambda\mathcal{L}_{\text{SC}}(\pi_\theta, \pi_E), \tag{93}$$

where λ controls the strength of scalar-critique guidance.

Scalar-critique diffusion policy. LC-DP-scalar applies the same scalar-critique principle to diffusion policy. The diffusion policy ϵ_θ is trained with the standard noise-prediction objective

$$\mathcal{L}_{\text{DP}} = \mathbb{E}_{(s_t, a_t^E)\sim\mathcal{D}_E, \epsilon\sim\mathcal{N}(0,\mathbf{I}), k}\left[\|\epsilon - \hat{\epsilon}\|_2^2\right], \qquad \hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k). \tag{94}$$

Since the diffusion policy predicts noise rather than actions, we apply the scalar-critique loss to the one-step reconstructed action

$$\hat{a}_t^0 = \frac{1}{\sqrt{\bar{\alpha}^k}} a_t^k - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}}\hat{\epsilon}, \qquad \hat{\epsilon} \sim \epsilon_\theta(\cdot \mid s_t, a_t^k, k). \tag{95}$$

Following Section E.3, we use the timestep-dependent weight $\omega^k = \frac{\bar{\alpha}^k}{1-\bar{\alpha}^k}$ to reduce the imbalance caused by noisier reconstructions at larger diffusion steps. The reweighted scalar-critique losses are

$$\tilde{\mathcal{L}}_Y(\pi_\theta) = \mathbb{E}_{(s_t, y_t)\sim\mathcal{D}_E^y, \hat{a}_t^0\sim\epsilon_\theta(s_t), k}\left[\omega^k \left\|Y_\phi(s_t, \hat{a}_t^0) - y_t\right\|_2^2\right], \tag{96}$$

$$\tilde{\mathcal{L}}_Y(\pi_E) = \mathbb{E}_{(s_t, a_t^E, y_t)\sim\mathcal{D}_E^y, k}\left[\omega^k \left\|Y_\phi(s_t, a_t^E) - y_t\right\|_2^2\right]. \tag{97}$$

The reweighted scalar-critique loss is then

$$\tilde{\mathcal{L}}_{\text{SC}}(\pi_\theta, \pi_E) = \left[\tilde{\mathcal{L}}_Y(\pi_\theta) - \text{sg}\Big(\tilde{\mathcal{L}}_Y(\pi_E)\Big)\right]_+. \tag{98}$$

The final LC-DP-scalar objective is

$$\mathcal{L}_{\text{DP}} + \lambda\tilde{\mathcal{L}}_{\text{SC}}(\pi_\theta, \pi_E). \tag{99}$$

LC-DP-scalar therefore trains the diffusion policy to predict noise such that the reconstructed actions both remain close to expert actions and match the expert scalar signals under the trained, frozen predictor model. Unlike LC-DP, however, LC-DP-scalar compresses behavioral supervision into scalar signals and therefore cannot preserve the structured behavioral distinctions provided by language label.

H.2 LC-BC-CLASSIFIER AND LC-DP-CLASSIFIER

We design two multi-head categorical baselines, LC-BC-classifier and LC-DP-classifier, as discrete-classification variants of LC-BC and LC-DP, respectively. These baselines convert each language label l_t into a tuple of categorical indices $c_t = (c_t^T, c_t^A, c_t^M)$ corresponding to the task progress <T>, action optimality <A>, and movement guidance <M>, discarding the natural-language form.

We test whether the gains of language-critique imitation arise from the expressiveness of natural language and the capacity of a pretrained language model to capture inter-label dependencies, or the independent categorical predictions over the same label dimensions. Both baselines follow the same two-stage pipeline: first, a multi-head classifier is trained on the general dataset with categorical labels c_t; second, the classifier is frozen and used for auxiliary supervision during policy learning on the expert dataset. Thus, LC-BC-classifier and LC-DP-classifier replace the language model μ_ϕ and the LC-loss with a multi-head classifier C_ϕ and a classifier-critique loss.

Classifier model. Each language label is decomposed into H independent categorical dimensions corresponding to task progress <T>, action optimality <A>, and movement guidance <M>, with C_h classes for each dimension $h \in \{1, \dots, H\}$. Given a categorically labeled general dataset $\mathcal{D}_G^c = \{(s_t, a_t, c_t)\}$, where $c_t = (c_t^1, \dots, c_t^H) \in \{1, \dots, C_1\} \times \cdots \times \{1, \dots, C_H\}$ encodes the class index for each dimension, we train a multi-head classifier C_ϕ that maps each state-action pair to per-head logits. The classifier consists of a shared MLP encoder followed by H parallel linear heads, where head h outputs logits $C_\phi^h(s_t, a_t) \in \mathbb{R}^{C_h}$. The classifier is optimized with a cross-entropy objective averaged over heads:

$$\mathcal{L}_C^\phi = \mathbb{E}_{(s_t, a_t, c_t)\sim\mathcal{D}_G^c}\left[\frac{1}{H}\sum_{h=1}^{H}\text{CE}\big(C_\phi^h(s_t, a_t),\ c_t^h\big)\right], \tag{100}$$

where $\text{CE}(\cdot, \cdot)$ denotes the standard cross-entropy loss. After training, C_ϕ is frozen and used for downstream policy optimization.

Classifier-critique loss. Given the frozen classifier C_ϕ, we define a classification-based analogue of the LC-loss on the expert dataset $\mathcal{D}_E^c = \{(s_t, a_t^E, c_t)\}$. For a policy action $\hat{a}_t \sim \pi_\theta(\cdot \mid s_t)$, the policy classification loss and the expert classification loss are

$$\mathcal{L}_C(\pi_\theta) = \mathbb{E}_{(s_t, c_t)\sim\mathcal{D}_E^c,\ \hat{a}_t\sim\pi_\theta(\cdot|s_t)}\left[\frac{1}{H}\sum_{h=1}^{H}\text{CE}\big(C_\phi^h(s_t, \hat{a}_t),\ c_t^h\big)\right], \tag{101}$$

$$\mathcal{L}_C(\pi_E) = \mathbb{E}_{(s_t, a_t^E, c_t)\sim\mathcal{D}_E^c}\left[\frac{1}{H}\sum_{h=1}^{H}\text{CE}\big(C_\phi^h(s_t, a_t^E),\ c_t^h\big)\right]. \tag{102}$$

The classifier-critique loss clips the policy loss at the expert loss:

$$\mathcal{L}_{\text{CC}}(\pi_\theta, \pi_E) = \left[\mathcal{L}_C(\pi_\theta) - \text{sg}(\mathcal{L}_C(\pi_E))\right]_+. \tag{103}$$

This objective penalizes policy actions whose predicted class distributions are less consistent with the expert categorical labels than the corresponding expert actions.

Classifier-critique behavior cloning. LC-BC-classifier jointly optimizes the behavior cloning loss (Equation (92)) and the classifier-critique loss:

$$\mathcal{L}_{\text{BC}} + \lambda \mathcal{L}_{\text{CC}}(\pi_\theta, \pi_E), \tag{104}$$

where λ controls the strength of classifier guidance.

Classifier-critique diffusion policy. LC-DP-classifier applies the classifier-critique principle to diffusion policy. As in LC-DP-scalar, the classifier-critique loss is applied to the one-step reconstructed action $\hat{a}_t^0$ (Equation (95)). Following Section E.3, we use the timestep-dependent weight $\omega^k = \frac{\bar{\alpha}^k}{1-\bar{\alpha}^k}$. The reweighted classification losses are

$$\tilde{\mathcal{L}}_C(\pi_\theta) = \mathbb{E}_{(s_t, c_t)\sim\mathcal{D}_E^c,\ \hat{a}_t^0\sim\epsilon_\theta(s_t),\ k}\left[\frac{\omega^k}{H}\sum_{h=1}^{H}\text{CE}\big(C_\phi^h(s_t, \hat{a}_t^0),\ c_t^h\big)\right], \tag{105}$$

$$\tilde{\mathcal{L}}_C(\pi_E) = \mathbb{E}_{(s_t, a_t^E, c_t)\sim\mathcal{D}_E^c,\ k}\left[\frac{\omega^k}{H}\sum_{h=1}^{H}\text{CE}\big(C_\phi^h(s_t, a_t^E),\ c_t^h\big)\right]. \tag{106}$$

The reweighted classifier-critique loss is then

$$\tilde{\mathcal{L}}_{\text{CC}}(\pi_\theta, \pi_E) = \Big[\tilde{\mathcal{L}}_C(\pi_\theta) - \text{sg}\Big(\tilde{\mathcal{L}}_C(\pi_E)\Big)\Big]_+. \tag{107}$$

The final LC-DP-classifier objective is

$$\mathcal{L}_{\text{DP}} + \lambda \tilde{\mathcal{L}}_{\text{CC}}(\pi_\theta, \pi_E). \tag{108}$$

I Additional experimental results

I.1 VLM-generated language labels

In Table 3b, we test whether a general-purpose VLM can replace the task-specific generator μ_g on BOX-CLOSE. We replace μ_g in Section 4.2 with `o4-mini`, prompted with the template in Figure 7, and keep every other part of the pipeline fixed. Sample labels from each variant are given in Table 9.

Raw VLM labels. With the default `SmolLM2-135M-Instruct` backbone, `o4-mini` labels reach 70.0% for LC-BC and 70.4% for LC-DP. LC-BC improve over the base policies (66.4%/72.4%), but 17.6% below μ_g. The most visible difference from μ_g is linguistic: VLM labels are longer, more varied in phrasing, and often padded with encouragement (Table 9), which plausibly makes their distribution harder for a small μ_ϕ to model.

Post-processing. If verbosity were the bottleneck, reducing it should help. We test two rewrites with 135M LLM-Captioner μ_ϕ: *concise* label shortens each label with an LLM while preserving its content, and μ_g*-style* rewrites each label into the μ_g template so that it is more structured like μ_g labels. However, neither post-processing helps. Concise labels leave performance unchanged (70.4%/70.0%), and μ_g-style labels reduce LC-BC to 64.4%. To understand why, we measure how often the VLM labels are correct, treating μ_g as a feature-aligned ground-truth target. `o4-mini` agrees with μ_g on 51.8% of `<T>` descriptions, 46.1% of `<A>`, and only 18.8% of `<M>`. Accuracy degrades with the spatial precision the component demands: task progress is a coarse judgment, whereas movement guidance requires reasoning about the effect of a continuous action on the state, which `o4-mini` does poorly. Rewriting cannot add information the VLM did not produce, so a label that was wrong about the movement direction is still wrong once it looks like μ_g. Therefore, accuracy is the bottleneck here, though label form and structure are not irrelevant: Section I.2 shows with the shuffled and verbose controls that accuracy is necessary, and that, given accurate labels, a compact form adds further gain.

Backbone capacity. Given that the labels are only partially accurate, the remaining lever is how well μ_ϕ models them. Scaling to `SmolLM2-360M-Instruct` raises VLM-label performance to 74.0%/74.8%, matching μ_g for LC-DP and narrowing the LC-BC gap to 13.6%. However, the increasing μ_g size lowers μ_g-label performance to 70.4%/70.8%. We attribute the asymmetry to label entropy: μ_g produces a small set of templated sentences that the 135M model already fits, so the larger model adds capacity without adding signal, and the LC-loss contributes less useful gradient to the policy; the diverse VLM text is underfit at 135M and uses the extra capacity of 360M models. Backbone size should therefore be chosen jointly with the label source rather than scaled unconditionally.

I.2 What properties of language matter?

The results above show that copying the form of μ_g does not recover its benefit, and our main result shows that accurate categorical targets recover most of it for LC-BC. This raises the question of what natural language contributes beyond an accurate structured signal. In Table 3c, we decompose the language label into four properties and construct supervision signals that remove one at a time:

- *Accuracy* (Acc.): the label is correct for the state-action pair. μ_g-shuffled randomly permutes labels within $\mathcal{D}_G^{\text{lang}}$, preserving the marginal distribution, format, and vocabulary while breaking the semantic accuracy.
- *Compactness* (Cmp.): the label is short and variable-length, omitting what does not apply. μ_g-verbose rewrites each label into human-like verbal feedback with the same content; fixed-length token formats always emit every component.
- *Lexical semantics* (Sem.): tokens carry meaning in the pretrained vocabulary. Opaque tokens (`<H00_C00>`) carry none; named tokens (`<TASK_PROGRESS_C00>`) carry partial meaning through their subwords; μ_g uses ordinary words.
- *Pretraining* (Pre.): μ_ϕ is initialized from a pretrained LLM. The from-scratch variant trains the same architecture with random initialization on the same μ_g labels.

Classifier variants (LC-BC-classifier, LC-DP-classifier) remove the language model entirely and predict each component with a categorical head, serving as the accurate, non-linguistic reference. Examples of each variant are in Table 9.

Accuracy is necessary. Shuffling reduces LC-BC from 87.6% to 69.2%, erasing nearly the entire gain over BC (66.4%), and reduces LC-DP from 74.0% to 67.6%, below DP (72.4%). Since format, vocabulary, and label statistics are unchanged, the LC-loss improves the policy only through the correctness of the mapping from the state-action pair to language label. This explains the VLM results above, where post-processing changes form but does not make labels accurate.

Semantics and language pretraining matter. Opaque, named, and natural-language labels carry identical information. For LC-DP they yield 66.0%, 67.6%, and 74.0%; the first two fall below DP, and only natural language improves over it. Training μ_ϕ from scratch (without language pretraining) on the same μ_g labels drops LC-DP to 63.2%, the lowest value in the table. For LC-BC, the same sequence is 84.4%, 86.0%, 87.6%, with from-scratch at 88.8%. We hypothesize that pretrained tokens share semantic structure that categorical tokens lack. For example, in Table 8, *front* and *back* encode related directions, *lid* recurs across different components, and *slightly* modifies several verbs. Predicting these tokens may encourage μ_ϕ to learn shared features across state-action pairs that require similar corrections. Named tokens may retain part of this structure through their pretrained subwords, which could explain their intermediate performance. But this interpretation remains a hypothesis. Since the three label formats are informationally equivalent, the ablation shows that label information alone does not explain the benefit, but it does not establish the underlying mechanism. The results suggest that the effectiveness of LC-loss depends on how μ_ϕ learns from language label on the general dataset. This leads to another question: does it also matter how the loss is distributed across tokens—specifically, which tokens are supervised and how many receive supervision? We examine this in the next property.

Compactness matters. Removing zero-valued guidance tokens from the named-token format yields the compact variable-length named-token baseline, which improves both policies (86.0% $\rightarrow$ 88.0% and 67.6% $\rightarrow$ 70.0%), and verbose labels reduce both (74.0%/68.0%) despite identical content. We hypothesize that label length reweights the loss: in the fixed-length format, most <M> tokens encode *no correction*, so most of the language loss is spent on states the policy is already handling; the variable-length format spends it on states that actually need correction. Verbose labels have the opposite problem, spreading the loss over filler tokens (*e.g.*, *Keep up the good work!*) that carry no information about the state, and a 135M model spends capacity on style rather than semantically accurate content.

Summary. The benefit of natural language over categorical supervision cannot be explained by information content alone. Our results instead point to three properties that matter jointly: accurate feedback, a compositional target space inherited from language pretraining, and a compact format. The two policies, however, depend on these properties to different degrees. LC-BC gains roughly 20% from any accurate, compact signal, including classifier heads (87.2%), with all such variants achieving 84–89% success. This suggests that the feedforward policy, which receives the observation directly, can benefit from structured auxiliary supervision even without pretrained language structure. The auxiliary target may help organize its shared features, while the direct observation input provides the information needed for action prediction. In contrast, LC-DP improves over DP only when the supervision is accurate, compact, and expressed in pretrained natural language; removing any one of these properties brings performance back to or below DP. One possible explanation is that the diffusion policy accesses the observation through its conditioning features and learns under gradients that vary with the sampled noise and noise level. It may therefore depend more strongly on the structure of the auxiliary target to learn useful conditioning features. This interpretation is consistent with Table 3a, where every partial language label is neutral or harmful for LC-DP, and only the full set improves performance. Together with Section 5.3, these results suggest that language supervision is most valuable when demonstrations are scarce or imperfect, and when the policy depends more strongly on the representation learned through auxiliary supervision. The results on BLOCKPUSH in Table 1 support this interpretation: in this multistage, multimodal task, language supervision may help organize features that distinguish task stages and capture multiple modes of valid behaviors.

Table 17: **Human-labeled critiques in the low-data setting.** We report success rates on BOX-CLOSE over 50 rollouts per seed and 5 seeds (% omitted). To limit annotation effort, we evaluate human labeling with only 2% of the expert data, retaining the full suboptimal dataset with its original μ_g labels. Human-label variants are manually annotated for the selected expert data. Best results within each policy family are in **bold**; second-best are underlined.

Method	BOX-CLOSE 2% data
BC	48.8 ± 11.9
DWBC	52.0 ± 14.4
DemoDICE	<u>72.8</u> ± 5.0
ILID	64.4 ± 4.3
LC-BC-scalar	63.2 ± 8.9
LC-BC-classifier	**74.4** ± 6.4
LC-BC (Ours)	71.6 ± 5.7
LC-BC (Human labels)	64.8 ± 6.9
DP	71.2 ± 1.8
LPB-Offline	72.0 ± 6.2
LC-DP-scalar	70.4 ± 2.2
LC-DP-classifier	73.2 ± 3.6
LC-DP (Ours)	**77.6** ± 7.1
LC-DP (Human labels)	<u>74.4</u> ± 2.6

I.3 HUMAN-GENERATED LANGUAGE LABELS

We investigate whether language-critique supervision remains effective when critiques reflect human judgment rather than task-specific labeling rules. To keep annotation effort manageable, we conduct this experiment on BOX-CLOSE using 2% of the expert data, replacing their μ_g labels with human annotations while retaining the full suboptimal dataset and its original μ_g labels. We compare against baselines trained under the same data configuration.

We recruit 10 people, none of whom are coauthors, to annotate the selected expert data. All annotators agree to participate and to have their annotations used in this research and reported in the paper. Annotators are shown images of the current and next states to support intuitive visual assessment of task progress and action quality, rather than requiring them to interpret numerical state representations. Then, annotators receive templates for the `<T>`, `<A>`, and `<M>` components and can select and freely revise their segments, sentences, and wording. They are instructed to assess each example according to their own judgment, without access to other annotators' labels. In particular, they are not required to evaluate expert actions positively: they may identify shortcomings and suggest corrections even for expert demonstrations. This protocol allows critiques to reflect individual standards and preferences, while introducing potential variation in annotation strictness.

As shown in Table 17, human-labeled LC-BC achieves a success rate of 64.8%, improving over BC by 16.0% and outperforming all prior offline IL baselines except for DemoDICE. Human-labeled LC-DP achieves 74.4%, exceeding DP by 3.2% and obtaining the second-highest mean success rate within the diffusion-policy class. Both variants score below their μ_g-labeled counterparts (71.6% and 77.6%, respectively), with a smaller gap for LC-DP. Under the evaluated configuration, LC-DP exhibits a smaller performance decrease when replacing expert-data critiques with human labels. These results support the feasibility of incorporating human judgments into language-critique supervision, with particularly encouraging performance for diffusion policies. Variation across annotators and differences between human critiques and the retained μ_g labels may contribute to the remaining gap, motivating future investigation of annotation consistency.

I.4 LLM-CAPTIONER ABLATION

Table 18 examines the choice of LLM-Captioner backbone and finetuning strategy on BOX-CLOSE. We compare `SmolLM2-135M-Instruct` (Allal et al., 2025), `SmolLM2-360M-Instruct` (Allal et al., 2025), `Qwen2-0.5B-Instruct` (Yang et al., 2024), and `Mistral-7B-Instruct-v0.3` (Jiang et al., 2023). For the 135M backbone, we evaluate full-model finetuning, projector-only finetuning with the backbone frozen, and a pretrained baseline without task-specific adaptation. We additionally evaluate full-model finetuning for the

Table 18: **LLM-Captioner ablation.** We ablate the choice of LLM-Captioner backbone and finetuning strategy, reporting mean and standard deviation across 50 environment rollouts and 5 seeds for both LC-BC and LC-DP on the task BOX-CLOSE. (% omitted.)

Backbone	Finetune mode	LC-BC	LC-DP
`SmolLM2-135M-Instruct`	Full-model	**87.6** $\pm$ 6.5	**74.0** $\pm$ 5.7
`SmolLM2-135M-Instruct`	Projector-only	80.8 $\pm$ 9.4	<u>72.0</u> $\pm$ 3.7
`SmolLM2-135M-Instruct`	Pretrained	61.2 $\pm$ 3.0	71.2 $\pm$ 5.4
`SmolLM2-360M-Instruct`	Full-model	70.4 $\pm$ 9.2	70.8 $\pm$ 3.6
`Qwen2-0.5B-Instruct`	Full-model	<u>79.2</u> $\pm$ 11.1	70.8 $\pm$ 7.7
`Mistral-7B-Instruct-v0.3`	Projector-only	64.8 $\pm$ 7.6	65.2 $\pm$ 4.8

360M and 0.5B backbones and projector-only finetuning for the 7B backbone. Architectural details are provided in Section D.

Task-specific adaptation benefits LC-BC more strongly than LC-DP. With the 135M backbone, LC-BC improves from 61.2% without adaptation to 80.8% with projector-only training and 87.6% with full-model finetuning. Thus, learning the projector mapping into the pretrained language space already provides a substantial benefit, while adapting the backbone provides a further gain. For LC-DP, the corresponding scores are 71.2%, 72.0%, and 74.0%. Full-model finetuning again achieves the highest mean.

Unconditionally increasing backbone size does not improve performance in the configurations tested. Under full-model finetuning, the 360M and 0.5B backbones achieve 70.4% and 79.2% for LC-BC, respectively, compared with 87.6% for the 135M backbone, and both exhibit greater variation across seeds. For LC-DP, both larger backbones achieve 70.8%, compared with 74.0% for the 135M model. Projector-only training with the 7B backbone also underperforms the same strategy with the 135M backbone for both policies, suggesting that a larger frozen language model does not necessarily provide a more useful target space for a smaller policy model in this setting. We therefore adopt `SmolLM2-135M-Instruct` with full-model fine-tuning as the default LLM-Captioner. It is the smallest backbone evaluated and yields the highest success rate for both policies on BOX-CLOSE, indicating that a compact, task-adapted language model suffices for templated language labels and that additional capacity can be counterproductive (see Section I.1). The choice also minimizes the overhead of the LC-loss, which is the main cost of our framework relative to BC and DP (Section G.4).

I.5 LLM-CAPTIONER TRAINING DATA ANALYSIS

Our framework uses the general dataset $\mathcal{D}_G$ and the expert dataset $\mathcal{D}_E$, following the typical problem setup of imitation learning from suboptimal demonstrations. The policy is trained on $\mathcal{D}_E$ (with the standard BC or diffusion objective on expert actions and the LC-loss evaluated on expert language labels), while the LLM-Captioner is fine-tuned on the full $\mathcal{D}_G^{\text{lang}}$. Since the policy training objective does not directly consume suboptimal trajectories, it is important to verify that the general dataset provides a meaningful contribution through the LLM-Captioner. To this end, we train an expert-only captioner μ_ϕ^E on $\mathcal{D}_E^{\text{lang}}$ alone and use it for downstream LC-BC and LC-DP training. Table 19 reports results on BOX-CLOSE over 50 environment rollouts and 5 training seeds.

Removing $\mathcal{D}_G$ from captioner training degrades LC-BC from 87.6% to 68.4%–recovering only 2.0% over plain BC (66.4%). For LC-DP, the effect is the same: the expert-only captioner drops performance to 66.8%, below the plain DP baseline (72.4%), indicating that a captioner without contrastive supervision can actively harm policy learning.

When μ_ϕ is trained on the full $\mathcal{D}_G^{\text{lang}}$, it observes both expert and suboptimal behaviors paired with their structured labels and learns a discriminative mapping that distinguishes good from bad behaviors. The LC-loss then transmits this discrimination to the policy. When μ_ϕ is trained only on $\mathcal{D}_E^{\text{lang}}$, it never observes negative or corrective labels, so its likelihood landscape over language labels becomes unable to distinguish action quality. The resulting LC-loss provides weak, non-discriminative gradients for LC-BC and noisy gradients that destabilize diffusion training for LC-DP. These results confirm that the general dataset is essential despite never entering the policy objective directly. $\mathcal{D}_G$ provides the

Table 19: **LLM-Captioner training data analysis.** We compare LLM-Captioner training on the full general dataset $\mathcal{D}_G$ against training on the expert subset $\mathcal{D}_E$ only, evaluated on BOX-CLOSE over 50 rollouts and 5 seeds (% omitted). Removing $\mathcal{D}_G$ from captioner training degrades LC-BC by 19.2% and LC-DP by 7.2%, driving LC-DP below the plain DP baseline.

Method	μ_ϕ training data	BOX-CLOSE
BC	–	66.4 ± 14.6
LC-BC	$\mathcal{D}_E$ only	68.4 ± 6.4
LC-BC	$\mathcal{D}_G$ (ours)	**87.6** ± 6.5
DP	–	72.4 ± 4.3
LC-DP	$\mathcal{D}_E$ only	66.8 ± 6.1
LC-DP	$\mathcal{D}_G$ (ours)	**74.0** ± 5.7

Table 20: **Scalar critiques derived from language versus task features.** We compare language-derived scalar critiques (LC-BC-scalar, LC-DP-scalar) with scalar scores computed directly from the task features used by μ_g (LC-BC-scalar-f, LC-DP-scalar-f), alongside LC-BC and LC-DP. We report task success rates across 5 training seeds and 50 environmental rollouts on all seven tasks, along with the average across tasks. Bold and underlined entries denote the best and second-best distinct mean success rates within each policy class. (% omitted)

Method	MAZE	PARKING	BOX-CLOSE	BLOCKPUSH	PEGINSERT	HAMMER	RELOCATE	**Average**
LC-BC-scalar	79.6 $\pm$ 4.8	42.0 $\pm$ 6.5	74.8 $\pm$ 13.8	38.0 $\pm$ 4.7	63.6 $\pm$ 3.3	90.0 $\pm$ 6.9	65.6 $\pm$ 11.2	64.8
LC-BC-scalar-f	78.4 $\pm$ 5.2	43.2 $\pm$ 3.0	**87.6** $\pm$ 5.4	41.2 $\pm$ 5.8	**71.2** $\pm$ 3.6	88.0 $\pm$ 15.0	70.4 $\pm$ 8.6	68.6
LC-BC (Ours)	**81.6** $\pm$ 6.1	**44.8** $\pm$ 3.9	**87.6** $\pm$ 6.5	**47.2** $\pm$ 4.1	67.2 $\pm$ 5.4	**99.2** $\pm$ 1.8	**71.2** $\pm$ 5.9	**71.3**
LC-DP-scalar	94.0 $\pm$ 2.0	**38.8** $\pm$ 1.1	67.2 $\pm$ 3.6	85.2 $\pm$ 1.1	53.2 $\pm$ 2.7	76.0 $\pm$ 1.4	66.8 $\pm$ 21.8	68.7
LC-DP-scalar-f	93.6 $\pm$ 0.9	**38.8** $\pm$ 2.3	64.8 $\pm$ 6.4	85.2 $\pm$ 3.0	52.8 $\pm$ 2.3	**80.8** $\pm$ 6.4	65.2 $\pm$ 23.2	68.7
LC-DP (Ours)	**94.4** $\pm$ 1.7	37.6 $\pm$ 3.8	**74.0** $\pm$ 5.7	**88.0** $\pm$ 1.4	**56.8** $\pm$ 2.7	80.0 $\pm$ 3.2	**77.6** $\pm$ 4.1	**72.6**

contrastive supervision that shapes μ_ϕ, which in turn transmits behavioral-quality distinctions to the policy via the LC-loss. Without it, the framework collapses toward standard expert-only imitation.

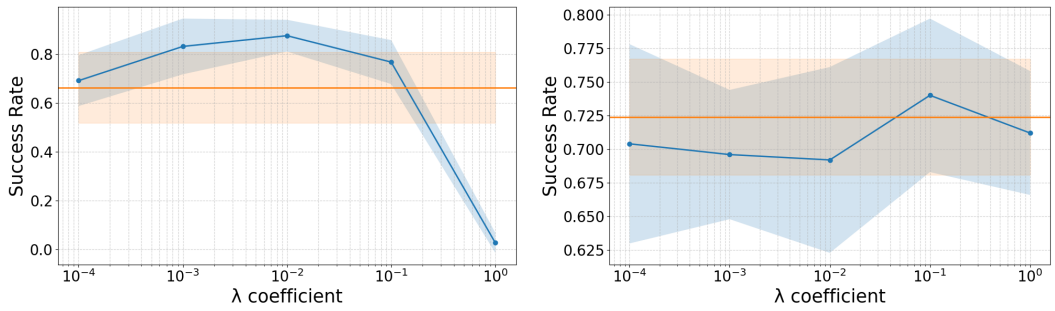
I.6 SCALAR CRITIQUES DERIVED DIRECTLY FROM TASK FEATURES

Our main experiments compare scalar-critique (LC-BC-scalar, LC-DP-scalar) and classifier-critique (LC-BC-classifier, LC-DP-classifier) variants to investigate how best to leverage language label in offline imitation learning. The scalar-critique variants convert language label into scalar scores, whereas the classifier-critique variants use categorical labels corresponding to the categories defined by μ_g. Since μ_g generates language label from task-relevant state-action features, we further investigate whether these features alone provide sufficient guidance for policy learning. Converting language into scalar scores may discard information useful for learning. To examine this possibility, we evaluate variants of LC-BC-scalar and LC-DP-scalar that derive scalar scores directly from the features used by μ_g, bypassing language generation. We negate features for which smaller values are desirable, such as gripper-to-object and gripper-to-goal distances, so that larger values consistently indicate more desirable behavior. We denote these variants by LC-BC-scalar-f and LC-DP-scalar-f and compare them with their language-derived counterparts, LC-BC-scalar and LC-DP-scalar, as well as LC-BC and LC-DP.

As shown in Table 20, deriving scalar scores directly from task features improves the average success rate of LC-BC-scalar from 64.8% to 68.6 %, whereas LC-DP-scalar remains unchanged at 68.7 %. This suggests that bypassing language conversion can benefit scalar critiques by a slight amount, although the gains depend on the policy type. The feature-derived variants are also competitive on individual tasks: LC-BC-scalar-f matches LC-BC on BOX-CLOSE and outperforms it on PEGINSERT, suggesting that scores derived directly from task features can provide fine-grained supervision for precision-sensitive tasks. In contrast, the broader gains of LC-BC suggest that language label may offer richer guidance for multimodal behavior, multi-stage tasks, and general control. Nevertheless, LC-BC and LC-DP achieve the highest average success rates of 71.3 % and 72.6 %, respectively, yielding absolute success rate gains of 2.7% and 3.9 % over their feature-derived scalar counterparts. These results suggest that the advantage of our approach is not explained solely by information lost when converting features into language-derived scalar scores.

I.7 λ SENSITIVITY

We also study the effect of the coefficient λ, which balances the behavior cloning (BC) objective and the LC-loss for LC-BC, and the diffusion objective and the LC-loss for LC-DP. In Figure 14, we evaluate $\lambda \in \{1.0, 0.1, 0.01, 0.001, 0.0001\}$. The results show that feedforward policies improve consistently by a large margin, but are sensitive to larger values of λ, with $\lambda = 1.0$ leading to degraded performance, whereas diffusion policies improve only at $\lambda = 0.1$, which is more sensitive than LC-BC.



(a) LC-BC λ sensitivity

(b) LC-DP λ sensitivity

Figure 14: λ **sensitivity.** Blue curves represent LC-BC and LC-DP, while orange horizontal lines represent BC and DP performance. We vary the balancing coefficient $\lambda \in \{1.0, 0.1, 0.01, 0.001, 0.0001\}$.

J QUALITATIVE RESULTS OF LC-LOSS AND LLM-CAPTIONER

To evaluate the qualitative performance, we visualize the execution of PEGINSERT and BOX-CLOSE tasks. We display six key frames for each, accompanied by the instantaneous LC-loss and the generated language labels. The results demonstrate that LC-BC achieves significantly lower LC-loss compared to BC due to the direct optimization of this objective. This stability allows LC-BC to generate language labels that offer clear, corrective semantic guidance even when the agent faces potential misalignment. This capability to detect and correct errors via language ensures robust performance and task success.

In the PEGINSERT task (Figure 15), the LC-BC agent successfully aligns and inserts the peg and maintains a LC-loss of 0.0. The generated language labels provide accurate, actionable corrections and generate precise instructions like, *"Yet the action is not good for the state, as the peg misaligns with the hole. Also, you should yaw toward the left softly."* allowing it to identify the misalignment of the peg and the hole and include accurate correction, which leads to the success of the task. In contrast, the BC policy achieves a relatively low LC-loss, but the peg remains outside the hole; when the observation deviates from the desired state, the policy fails to recover.

For the BOX-CLOSE task (Figure 16), the LC-BC agent maintains a consistently low LC-loss throughout the episode; this stability is reflected in the generated language labels. From the fourth key frame of LC-BC, it shows that instructions like *"the action is good for the state."* and *"you need to move toward the left gently"* provide assessment of action and movement guidance. As a result, the LC-BC agent successfully aligns the lid and closes the box. On the other hand, the BC agent struggles as the task progresses; its LC-loss spikes (reaching 0.17 and 0.61) in the later stages, and the BC agent cannot recover from errors once the observation drifts from the training distribution, thus resulting in failure of the task.

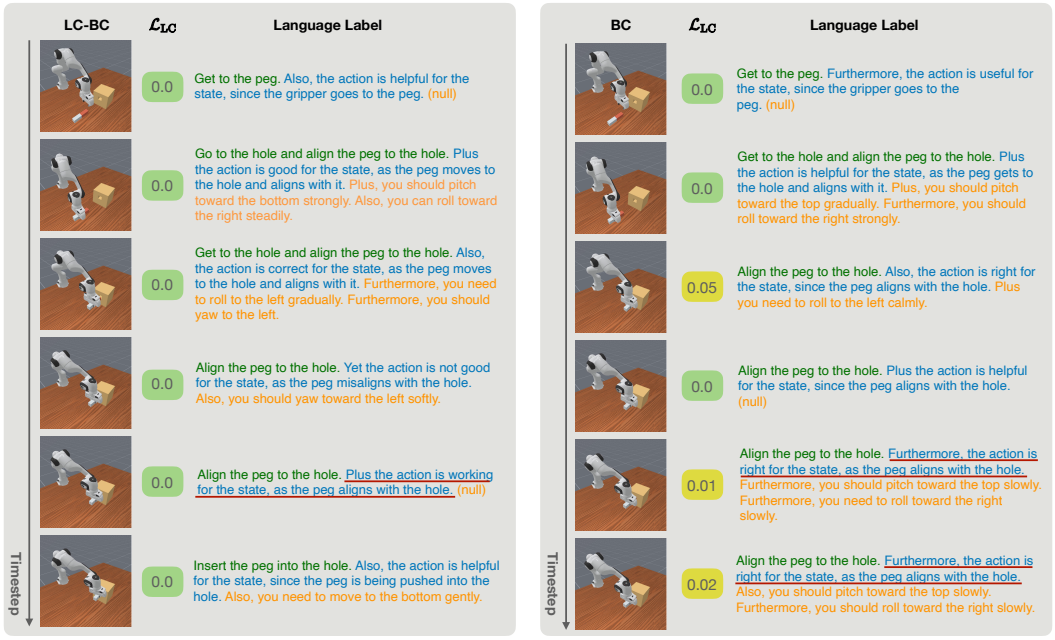


Figure 15: **Qualitative result of PEGINSERT.**

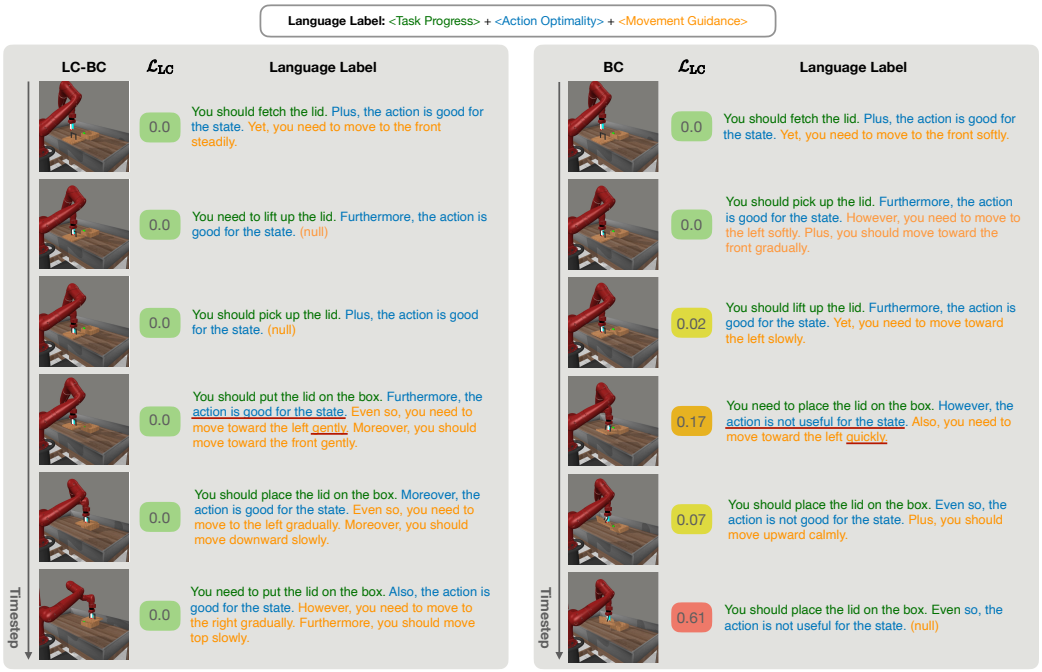


Figure 16: **Qualitative result of BOX-CLOSE.**

K AI USE STATEMENT

We used ChatGPT to assist with manuscript drafting and editing, literature and citation checking, and refining and checking mathematical details. As described in the paper, we used OpenAI `o4-mini` to generate and rewrite language critiques and `Mistral-7B` to convert language labels into scalar scores for baseline comparisons. Pretrained language models also serve as the LLM-Captioner backbones in our method and ablations. The authors are responsible for verifying AI-assisted content and take full responsibility for the final manuscript, theoretical claims, and experimental results.

L ETHICS STATEMENT

Our human-annotation experiment in Section I.3 involves 10 labmates who are not coauthors. All annotators agreed to participate and to the use of their annotations for this research and its publication. The annotation task consists of inspecting images of simulated robot transitions and providing language critiques of robot behavior.