DUAL-PROCESS ATOMIC SKILL LEARNING: DECOUPLING SEMANTIC REASONING AND REAL-TIME CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-conditioned imitation learning must discover reusable skills from demonstrations to generalize to multi-step instructions. Joint training of discrete skill hierarchies can produce concentrated codebook usage and unstable skill representations. We propose Dual-Process Atomic Skill Learning (**DASL**), which independently parameterizes the semantic update interval and low-level trajectory horizon, supervises each shared skill with actions across its interval, and regularizes skill-conditioned latent trajectories with a diffusion module used only during training. A slow Option Transformer predicts vector-quantized skills, while a Decision Transformer generates actions from full-rate observations without diffusion sampling at inference. Evaluations on LOReL, Franka Kitchen, a six-task CALVIN-D state-based protocol, BabyAI, and a real robot demonstrate improved skill acquisition and compositional performance in these settings. Direct codebook diagnostics and an isolated interval-supervision ablation support the proposed training mechanism, while the deterministic inference path preserves efficient control.

1 INTRODUCTION

Language-conditioned imitation learning (IL) aims to train robots to execute tasks specified by natural language instructions using paired language-trajectory demonstrations. However, this objective becomes increasingly arduous when tasks involve sequential execution of multiple sub-goals (Gupta et al., 2019; Yu et al., 2020). A key strategy to address this complexity exploits the inherent hierarchical structure of natural language to transform complex task solving into learning atomic skills. Recombining these atomic skills enables robots to generalize to unseen task configurations, a capability particularly critical in data-constrained regimes where covering the entire task space is infeasible.

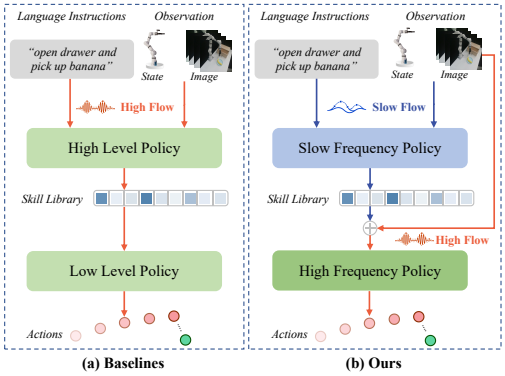


Figure 1: Hierarchical information flow. (a) Conventional skill hierarchy. (b) DASL independently sets I and H and fuses slowly updated skills with full-rate observations. Baselines may also hold skills across multiple steps.

Hierarchical skill learning (Lynch et al., 2020; Rosete-Beas et al., 2023; Garg et al., 2022) provides an intermediate representation between language and control. A central challenge is learning a useful discrete vocabulary through joint optimization: concentrated codebook usage can leave much of the intended skill library inactive, while broad activation alone does not ensure coherent language–behavior associations. LCSD and LADS also identify difficulties in learning stable language-conditioned latent skills (Ju et al., 2024; Jiang et al., 2025). We investigate how the temporal organization of action supervision and latent trajectory regularization affect this problem.

Dual-process cognition motivates slow semantic reasoning and fast action execution (Shentu et al., 2024; Wen et al., 2025; Chen et al., 2025), and this execution pattern is already established in hierar-

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.

chical policies and dual-system VLAs. HiRT and Fast-in-Slow coordinate slowly updated semantic features with fast action policies (Zhang et al., 2025; Chen et al., 2025). DASL studies discrete skill discovery from language-conditioned demonstrations without skill or subtask annotations. Its contribution concerns the joint organization of independent temporal parameters, interval-level skill supervision, and training-only latent diffusion for this learning problem.

DASL combines a slow Option Transformer, a vector-quantized skill library, and a full-rate Decision Transformer (Figure 1). The semantic interval I is selected independently of the low-level trajectory horizon H. Each skill is repeated across its interval and fused with current observations, so multiple action losses contribute supervision to the shared representation. An auxiliary skill-conditioned diffusion model regularizes the fused latent trajectories during training and is removed during rollout. This unified design retains closed-loop action prediction while organizing the learning signals received by the skill planner.

We evaluate DASL on LOReL (Nair et al., 2022a), Franka Kitchen (Gupta et al., 2019), a six-task CALVIN-D protocol (Mees et al., 2022; Ju et al., 2024), BabyAI (Chevalier-Boisvert et al., 2018), and a Realman robot. The results show gains in language-conditioned skill acquisition and compositional task performance. Component ablations, quantitative codebook diagnostics, and a controlled removal of interval-level supervision examine the method's design. Additional perturbation and deployment analyses report performance under state changes and inference cost.

In summary, our main contributions are as follows:

1. We formulate DASL for discovering reusable discrete skills from language-conditioned demonstrations, with independently configured semantic intervals and low-level trajectory horizons.
2. We jointly combine interval-level multi-step skill supervision, full-rate skill–observation fusion, and training-only latent diffusion; direct diagnostics and a targeted ablation examine their effects on codebook use and task performance.
3. We evaluate the method across simulated and physical tasks, reporting compositional generalization, benchmark-specific limitations, robustness to unseen perturbations, and efficient deterministic inference.

2 Related Work

2.1 Language-Conditioned Imitation Learning

IL allows embodied agents to learn behaviors directly from expert demonstrations Stepputtis et al. (2020); Finn et al. (2017). Scaling IL to complex multi-task settings demands effective task specification. Unlike goal images or trajectories Nair et al. (2018); Finn et al. (2017), natural language serves as a uniquely powerful and general interface Garg et al. (2022). Existing language-conditioned IL methods fall into two paradigms: aligning pretrained language embeddings with visual observations for direct action prediction Shridhar et al. (2022); Kim et al. (2024); Wen et al. (2025); Wang et al. (2026), or leveraging Large Language Models (LLMs) to decompose instructions into sub-task sequences Ahn et al. (2022); Shentu et al. (2024); Liu et al. (2025); Gu et al. (2026). Instead of using external LLM planners, we adopt pretrained language embeddings to condition a hierarchical policy, executing complex multi-step tasks via structured skill learning.

2.2 Language-Conditioned Skill Learning

Traditional skill learning uses latent variable models (Eysenbach et al., 2018), options (Konidaris & Barto, 2009), and mutual information objectives (Hausman et al., 2018). Language-conditioned approaches add semantic grounding: LISA jointly learns vector-quantized skills and a low-level policy and already predicts skills periodically (Garg et al., 2022); LCSD regularizes language–skill association (Ju et al., 2024); SkillDiffuser uses skill-conditioned diffusion for task execution (Liang et al., 2024); and LADS learns latent-plan abstractions separately from its language-conditioned high-level policy (Jiang et al., 2025). RoLD models robot behavior through latent diffusion (Tan et al., 2025), while DynaMind reasons over abstract video dynamics (Wang et al., 2025). These works motivate examining stable skill representations together with their training and execution interfaces.

DASL differs from these skill hierarchies through the joint organization of independently configured semantic intervals and low-level horizons, interval-level action supervision of shared skills, and training-only latent trajectory diffusion. LISA uses a shared skill horizon, while LADS separates latent-plan learning from language-conditioned high-level learning. DASL retains joint training and full-rate skill–observation fusion; its Decision Transformer predicts actions without the iterative diffusion sampling used for execution in SkillDiffuser. This combination targets codebook use and coherent language-conditioned behavior.

2.3 Hierarchical Policy Learning

Hierarchical policy learning decomposes complex tasks into sub-goals to support sample-efficient learning and generalization. Early approaches used latent variables to represent skills Eysenbach et al. (2018). More recent methods Zhang et al. (2024); Wan et al. (2024); Kim et al. (2025) incorporate vision foundation models Oquab et al. (2023); Nair et al. (2022b) for high-level skill abstraction. DASL focuses on learning discrete, language-conditioned skills and their action mappings from demonstrations, with explicit temporal organization of skill supervision.

Dual-system VLA architectures coordinate slow semantic processing with fast execution (Zhang et al., 2025; Chen et al., 2025; Bjorck et al., 2025; Song et al., 2025). HiRT uses slowly refreshed VLM features to guide a faster vision policy, and Fast-in-Slow coordinates reasoning and execution within a unified VLA. DASL focuses on discovering reusable discrete skills and learning their action mappings from language-conditioned offline demonstrations, without subtask annotations or a pretrained VLA policy. This setting requires learning a skill vocabulary with effective codebook utilization and coherent language–skill–action associations. DASL jointly trains a slow discrete-skill planner and a fast action policy, using interval-level action supervision and training-only latent diffusion to regularize skill-conditioned representations.

3 Method

3.1 Overview

Figure 2 summarizes DASL. A slow Option Transformer maps language and observations to vector-quantized skills, which guide a fast Decision Transformer through skill–observation fusion. Joint training combines action prediction, vector quantization, and latent diffusion losses without subtask annotations. At inference, the policies update skills and actions at their respective rates, while the diffusion module is removed.

3.2 Problem Setup

We model multi-task imitation learning as a Task-Augmented Markov Decision Process (MDP) Garg et al. (2022) with tasks specified by natural language instructions $l \in \mathcal{L}$. Given an offline dataset of language–trajectory pairs $\mathcal{D} = \{(l^i, o_1^i, a_1^i, \ldots, o_{T_i}^i, a_{T_i}^i)\}_{i=1}^N$, we aim to learn a language-conditioned sequential policy $\pi(a_t \mid o_{\leq t}, a_{<t}, l)$ that generalizes to unseen task and environment. Furthermore, the lack of sub-task annotations necessitates the unsupervised discovery of reusable skill primitives.

3.3 High-Level Semantic Skill Planning

Within our dual-process architecture, the high-level policy governs language-guided skill planning. Operating on a slower timescale than the low-level controller, its core objective is to achieve temporal abstraction by mapping multimodal observations into a sequence of discrete behavioral primitives.

Slow-Frequency Timescale. Let L denote the sliding observation–action context length, H the low-level trajectory horizon, and I the number of control steps between skill updates. A shared-horizon setting uses $I = H$; DASL selects I independently of H, allowing skill duration and trajectory horizon to be configured separately. The high-level policy processes the context on the strided timeline $\mathcal{T}_{\text{slow}} = \{1 + kI \mid k = 0, 1, \ldots, \lfloor (L-1)/I \rfloor\}$. At each $t_s \in \mathcal{T}_{\text{slow}}$, it predicts a continuous

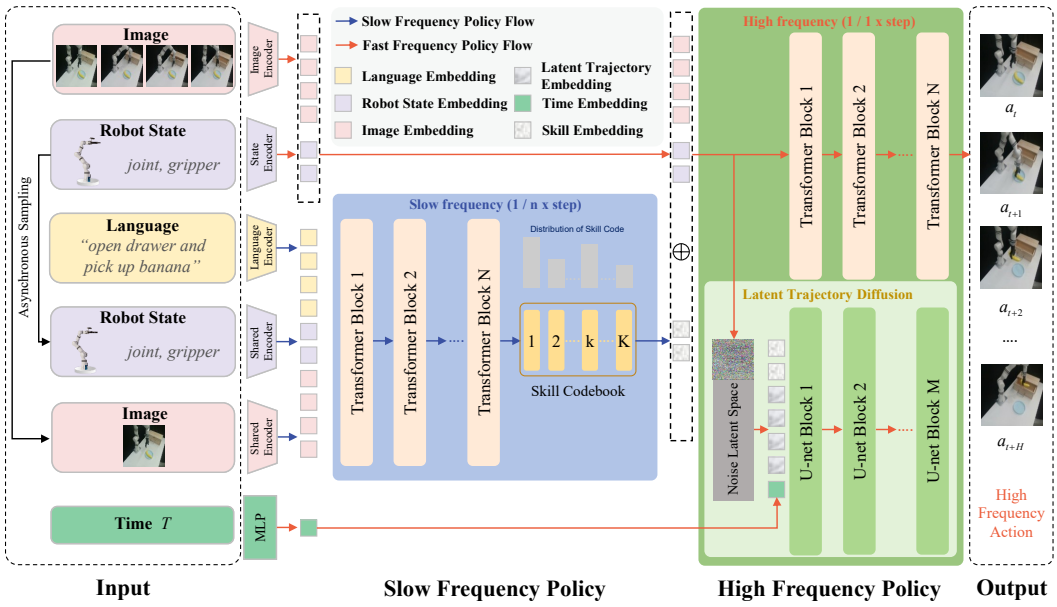


Figure 2: Overview of DASL, an asynchronous hierarchical imitation learning framework. The high-level policy operates at a slow timescale to generate discrete semantic skills from language instructions and sparse observations, while the low-level policy executes skill-conditioned actions at a fast timescale. A latent diffusion module regularizes the latent trajectory space during training and is removed at inference for efficient real-time control.

skill $\bar{z}_{t_s}$ from the instruction and observations available up to that point. The selected skill guides the fast policy until the next semantic update.

Skill Predictor. We employ an Option Transformer for cross-modal reasoning. First, observations at each slow timestep are encoded via a visual encoder ϕ to form the sequence $\mathbf{O} = [\phi(o_{t_s})]_{t_s \in \mathcal{T}_{\text{slow}}} \in \mathbb{R}^{|\mathcal{T}_{\text{slow}}| \times d}$. This is temporally concatenated with embedded instructions $\mathbf{E}^{\text{lang}} \in \mathbb{R}^{L_{\text{lang}} \times d}$ to form the joint sequence $\mathbf{S} = [\mathbf{E}^{\text{lang}}; \mathbf{O}]$. Processing $\mathbf{S}$ through causal self-attention allows observation tokens to attend to the global semantic context. We then extract the hidden states $\mathbf{h}_{t_s}$ exclusively from the observation positions and project them via a linear head to yield continuous latent skills $\bar{z}_{t_s} = \mathbf{W}_{\text{skill}} \mathbf{h}_{t_s} + b_{\text{skill}}$. Stacking these yields the continuous skill matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{|\mathcal{T}_{\text{slow}}| \times d_{\text{option}}}$.

Discrete Skill Vector Quantization. To obtain a discrete skill library, we map each continuous pre-activation $\bar{z}_{t_s}$ to a learnable codebook $\mathcal{C} = \{c_m\}_{m=1}^{M} \subset \mathbb{R}^{d_{\text{option}}}$ via a VQ bottleneck. The discrete skill z_{t_s} is retrieved via nearest-neighbor lookup:

$$z_{t_s} = \text{VQ}(\bar{z}_{t_s}) = \underset{c \in \mathcal{C}}{\text{argmin}} \, \|\bar{z}_{t_s} - c\|_2, \tag{1}$$

Applying this quantization row-wise to $\bar{\mathbf{Z}}$ yields the discrete skill matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{T}_{\text{slow}}| \times d_{\text{option}}}$, serving as the high-level temporal guidance for the low-level controller. During training, we optimize the codebook using an auxiliary commitment loss and approximate gradients through the quantization step via the Straight-Through Estimator (STE).

3.4 Low-Level Skill-Conditioned Decision Transformer

Operating at a high frequency, the low-level policy synthesizes actions conditioned on observations and skills. We introduce a hybrid architecture coupling a Decision Transformer for real-time inference with a latent diffusion module employed solely during training. This design uses diffusion to structure the latent manifold, ensuring semantic alignment between skills and trajectories without compromising the efficiency of the transformer backbone.

Skill Fusion. We condition the low-level policy by fusing discrete skills with high-frequency observations. Specifically, the retrieved skill sequence is projected into embeddings $e_{t_s}^{\text{skill}}$ and upsampled by repeating each vector I times, with the final interval truncated to the context length

L. At each fast timestep, the aligned skill embedding is added element-wise to the observation embedding e_t^{obs}, incorporating semantic guidance without increasing the feature dimensionality. Stacking the fused embeddings yields $\mathbf{E}^{\text{fused}}$:

$$e_t^{\text{fused}} = e_t^{\text{obs}} + e_{1+I\lfloor (t-1)/I \rfloor}^{\text{skill}}, \qquad t = 1, \ldots, L. \tag{2}$$

Latent Trajectory Diffusion Module. The auxiliary diffusion module operates on the fused latent trajectory $\tau_0 = \mathbf{E}_{1:H_{\text{diff}}}^{\text{fused}}$ before timestep embeddings are added, where $H_{\text{diff}} = \min(H, L)$. Its forward process adds Gaussian noise at diffusion step n using a cosine schedule $\bar{\alpha}_n$:

$$\tau_n = \sqrt{\bar{\alpha}_n}\tau_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3}$$

For the reverse process, a 1D U-Net ϵ_θ is trained to predict this injected noise, conditioned on the skill sequence and the diffusion step n. The diffusion objective regularizes the fused trajectory representation during training; rollout uses the action policy without iterative denoising.

Closed-loop execution. At each control step, the Decision Transformer predicts a new action from current observations and action history; the slow planner also receives refreshed observations when selecting the next skill. Holding a skill for I steps therefore preserves observation feedback in the fast policy. This combination supports stepwise action correction and subsequent skill reselection in response to observed state deviations.

3.5 Training Objectives

DASL jointly optimizes action prediction, vector quantization, and latent trajectory diffusion to learn discrete skills for continuous control. The independently configured semantic update interval and low-level trajectory horizon allow a selected skill to guide multiple fast control steps. Action losses from these steps jointly supervise the shared skill, while the action policy remains conditioned on current observations. During training, the auxiliary diffusion objective regularizes the fused skill-conditioned latent trajectory, linking the learned skills to demonstrated motions.

Action Prediction Loss. The low-level Decision Transformer performs behavioral cloning to reproduce expert actions given observations and target skills. The loss over a context window of length L is the mean squared error between predicted actions $\hat{a}_t$ and ground-truth a_t^{expert}:

$$\mathcal{L}_{\text{action}} = \frac{1}{L}\sum_{t=1}^{L} \|\hat{a}_t - a_t^{\text{expert}}\|_2^2. \tag{4}$$

Interval-level supervision and VQ commitment. To stabilize discrete skill representations, we optimize the Option Transformer's continuous outputs $\bar{z}_{t_s}$ and codebook $\mathcal{C} = \{c_m\}_{m=1}^{M}$ via the VQ objective:

$$\mathcal{L}_{\text{vq}} = \|\bar{z}_{t_s} - \text{sg}[z_{t_s}]\|_2^2 + \beta\|\,\text{sg}[\bar{z}_{t_s}] - z_{t_s}\|_2^2, \tag{5}$$

where z_{t_s} is the retrieved code, β controls the relative weight of the codebook update term, and $\text{sg}[\cdot]$ denotes the stop-gradient operator. This objective organizes skills around reusable codebook entries. Each selected skill also receives supervision from the action losses at the fast steps that share it. For skill z_k, let S_k be those steps and $\ell_t = \|\hat{a}_t - a_t^{\text{expert}}\|_2^2$; the interval-averaged action gradient is

$$g_k = \frac{1}{|S_k|}\sum_{t \in S_k} \frac{\partial \ell_t}{\partial z_k}. \tag{6}$$

The action gradients pass through skill–observation fusion and the straight-through VQ estimator, jointly training the discrete representation and action policy.

Latent Diffusion Loss. We introduce an auxiliary latent diffusion loss $\mathcal{L}_{\text{diff}}$ to model the skill-conditioned latent trajectory distribution and regularize the fused representations:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\tau_0, n, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\left\|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_n}\tau_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon}_{\tau_n}, n, c)\right\|_2^2\right]. \tag{7}$$

Conditioning the denoising objective on the skill sequence c links the discrete skills to the fused latent trajectory distribution. Its role is auxiliary representation learning: no generated trajectory or

Table 1: **Rephrasal-wise success rates (%) on LOReL.** Average is the macro-average across five language categories. Blue zero standard deviations denote cited results with unreported uncertainty; they are placeholders, not measured deviations. Our runs report mean ± standard deviation over three seeds. Best/second-best values are bold/underlined.

Rephrasal Type	Obs	Lang DT	LISA	LCSD	SkillDiffuser	LADS	DASL(ours)
seen	State	49.67 ± 5.77	12.78 ± 9.02	60.20 ± 0.00	38.89 ± 14.30	51.56 ± 1.89	**62.22 ± 3.85**
	Image	15.00 ± 0.00	40.00 ± 0.00	50.80 ± 0.00	43.65 ± 4.70	40.20 ± 4.40	**53.33 ± 5.36**
unseen noun	State	35.44 ± 0.53	10.00 ± 6.70	50.20 ± 0.00	36.67 ± 3.34	50.56 ± 11.33	**61.11 ± 7.03**
	Image	13.33 ± 0.00	33.33 ± 0.00	30.40 ± 0.00	36.01 ± 6.30	35.10 ± 6.50	**50.11 ± 5.47**
unseen verb	State	31.56 ± 5.85	11.22 ± 9.88	29.30 ± 0.00	31.11 ± 10.75	44.56 ± 1.37	**50.00 ± 10.00**
	Image	28.33 ± 0.00	30.00 ± 0.00	27.50 ± 0.00	36.70 ± 9.50	31.10 ± 0.50	**58.67 ± 9.53**
unseen noun&verb	State	17.33 ± 9.19	11.00 ± 7.20	28.70 ± 0.00	26.67 ± 3.34	34.44 ± 5.19	**45.56 ± 5.08**
	Image	6.70 ± 0.00	20.00 ± 0.00	36.50 ± 0.00	**42.02 ± 3.80**	21.70 ± 1.20	39.55 ± 0.39
human provided	State	34.35 ± 7.47	10.42 ± 7.23	35.50 ± 0.00	35.47 ± 3.93	50.06 ± 4.75	**52.20 ± 0.77**
	Image	26.98 ± 0.00	27.35 ± 0.00	33.40 ± 0.00	40.16 ± 2.10	33.90 ± 1.30	**49.01 ± 2.10**
Average	State	33.67 ± 5.41	11.08 ± 7.98*	40.78 ± 0.00	33.76 ± 4.83	46.23 ± 3.95	**54.22 ± 4.65**
	Image	18.07 ± 0.00	30.14 ± 0.00	35.70 ± 0.00	39.71 ± 0.00	32.40 ± 1.40	**50.13 ± 0.43**

Table 2: Kitchen N-rates for seen and unseen combinations under state and image observations.

Task Split	Obs	Lang DT	LISA	LADS	DASL(ours)
seen	State	1.39±0.12	0.71±0.44	**2.25±0.14**	1.66±0.05
	Image	1.32±0.41	0.93±0.12	1.61±0.28	**1.74±0.09**
unseen	State	1.27±0.17	0.71±0.53	**2.23±0.16**	1.76±0.14
	Image	1.07±0.32	1.29±0.20	1.44±0.32	**1.71±0.17**

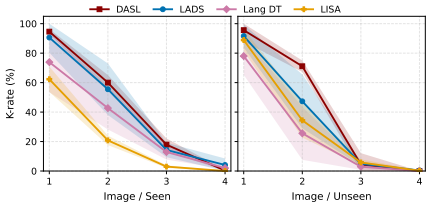


Figure 3: K-rates on seen and unseen tasks in Kitchen (image).

iterative diffusion sample is required during rollout. The component ablations and diagnostics in Section 4.3 examine when this regularization improves skill use and task performance.

Total Training Objective. This decoupled formulation allows DASL to jointly optimize the asynchronous modules while learning stable behavioral semantics:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{action}} + \lambda_{\text{vq}}\mathcal{L}_{\text{vq}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}}, \tag{8}$$

where λ_{vq} and λ_{diff} are hyperparameters. This multi-objective loss function enables the model to learn a complete mapping from language instructions to action sequences. The complete procedure is detailed in Appendix A. Details regarding the network architecture are provided in Appendix D.1.

4 Experiments

In this section, we evaluate DASL on robotic manipulation and grid-world navigation tasks, comparing it against strong hierarchical and non-hierarchical baselines. We further analyze the generalization of the framework to compositional instructions, the contributions of the architectural components, the efficiency of inference, and the practical viability of the approach through a real-world experiment.

Environments. We evaluate the proposed method across simulated benchmarks, including robotic manipulation tasks (LOReL Sawyer Nair et al. (2022a), Franka Kitchen (Gupta et al., 2019), and CALVIN (Mees et al., 2022)) and grid-world navigation tasks (BabyAI (Chevalier-Boisvert et al., 2018)). Beyond simulations, we validate DASL in a real-world experiment to demonstrate its effectiveness. Please refer to Appendix B for dataset specifications and Appendix G for the real-world experiment setup.

Baselines. We compare DASL with LCBC (Stepputtis et al., 2020), LangDT and LISA (Garg et al., 2022), HULC (Mees et al., 2022), LCSD (Ju et al., 2024), SkillDiffuser (Liang et al., 2024), and LADS (Jiang et al., 2025). Results are taken from published studies under matching evaluation protocols or obtained using official implementations. Detailed descriptions and configurations of all baselines, together with the sources of reported results, are provided in Appendix C.

Table 3: Six-task CALVIN-D state success rates.

Method	Lang DT	LISA	HULC	LCSD	DASL(ours)
SR	11.70 ± 0.80	10.10 ± 3.30	32.50 ± 2.50	35.60 ± 1.80	**48.05 ± 1.27**

Table 4: Performance on LOReL Compositional Tasks

Obs	Lang DT	LISA	SkillDiffuser	LADS	DASL(ours)
State	11.11 ± 1.78	0.17 ± 0.29	10.55 ± 2.52	23.11 ± 2.76	**23.61 ± 2.10**
Image	13.33 ± 1.30	20.89 ± 0.60	25.21 ± 2.70	20.40 ± 1.80	**32.50 ± 3.99**

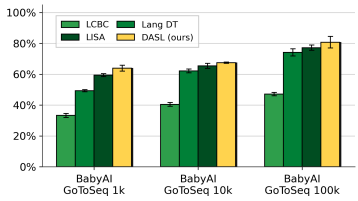


Figure 4: Success rates (%) on BabyAI GoToSeq task with varying numbers of demonstrations.

4.1 Performance on Language-Conditioned Benchmarks

We evaluate skill learning under the benchmark protocols described in Appendix B, using the same vision and language encoder architectures as LISA where applicable. Our main reproduced results use three random seeds and report mean and standard deviation; published results retain the statistics supplied by their sources. Component ablations, codebook diagnostics, and efficiency measurements use a common fixed seed. Appendix B.1 defines the LOReL averaging schemes.

Performance on the LOReL Sawyer dataset. Table 1 compares success rates across five language-rephrasal categories with state and image observations. DASL achieves the highest macro-average SR in both modalities: 54.22% for state and 50.13% for image inputs, outperforming both the flat LangDT policy and the hierarchical LISA baseline. These results demonstrate effective language-conditioned skill execution across diverse instruction formulations. Section 4.3 analyzes the contribution of temporal supervision and latent trajectory regularization through component ablations and codebook diagnostics. Task-wise results and qualitative skill analyses are provided in Appendices E.1 and E.5.

Performance on Franka Kitchen. We report N-rate, the average number of completed subtasks out of four, and K-rate, the probability of completing at least K subtasks (Table 2 and Figure 3). DASL achieves the highest image-based N-rates of 1.74 on seen and 1.71 on unseen task combinations, compared with LADS at 1.61/1.44 and LISA at 0.93/1.29. On unseen image instructions, DASL reaches 71.1% completion for $K = 2$ and 95.6% for $K = 1$, compared with LADS at 47.3% and 91.6%, respectively. These gains show more effective subtask sequencing and generalization to new combinations of familiar skills. Appendix E.3 provides state-based results and the full K-rate distributions.

Performance on CALVIN. We follow the LCSD six-task CALVIN-D state-based protocol, using 21-dimensional observations and an MLP encoder (Appendix B.3). Table 3 reports 48.05% SR for DASL, compared with HULC at 32.50%, LCSD at 35.60%, LangDT at 11.70%, and LISA at 10.10%. The 15.55 percentage-point improvement over HULC demonstrates the effectiveness of DASL's language-conditioned skill learning for the evaluated manipulation tasks. Additional execution examples are provided in Appendix E.4.

Performance on BabyAI. Figure 4 reports GoToSeq results across demonstration budgets, with dataset details in Appendix B.4. With 1k trajectories, DASL achieves $63.93\% \pm 1.91\%$ SR versus LISA's $59.3\% \pm 0.9\%$, a 4.63 percentage-point gain. The results show effective skill learning in the low-data grid-world setting.

4.2 Generalization to Compositional Instructions

We evaluate compositional instructions on LOReL using the tasks in Table 8. As shown in Table 4, DASL achieves the highest mean SR with both state and image observations, reaching 23.61% and 32.50%, respectively. Compared with the strongest baseline in each modality, DASL improves state SR over LADS (23.11%) by 0.50 percentage points and image SR over SkillDiffuser (25.21%) by 7.29 percentage points. These gains demonstrate the effectiveness of DASL's learned skill hierarchy for compositional execution, with particularly strong improvements under image observations.

4.3 Component Ablations and Codebook Diagnostics

Table 5 evaluates the skill embedding, independent temporal parameters, and latent diffusion on LOReL. The merged results include task success and LOReL-state codebook diagnostics; an isolated interval-supervision ablation tests the contribution of multi-step action feedback. Further diagnostics and protocols appear in Appendices E.6 and F.7.

Table 5: LOReL component ablations and codebook diagnostics. Emb.: skill embedding; Indep.: independent I, H; Diff.: latent diffusion; IS: interval supervision; SR: success rate (%); Active/20: used codes out of 20; H_{norm}: normalized code-use entropy; Soft MI: mutual information between instructions and soft skill assignments. Diagnostics use state inputs; rows share one fixed seed.

Method	Emb.	$I = H$	Indep. I, H	Diff.	State SR	Image SR	Active/20	H_{norm}	Soft MI
LISA	×	✓	×	×	19.08	30.14	4	0.451	0.015
Variant A	✓	✓	×	×	28.34	32.26	20	0.942	0.244
Variant B	✓	×	✓	×	38.26	38.00	19	0.866	0.254
Variant C	✓	✓	×	✓	32.08	31.19	20	0.937	0.249
DASL w/o IS	✓	×	✓	✓	43.12	40.37	15	0.759	0.254
DASL (ours)	✓	×	✓	✓	**58.49**	**50.66**	19	0.930	**0.269**

Effectiveness of Latent Skill Embedding. Moving from LISA to Variant A increases state SR from 19.08% to 28.34%, activates all 20 codes, and raises normalized usage entropy from 0.451 to 0.942. This shows improved codebook utilization with the skill embedding design. Combining it with independent temporal parameters and latent diffusion yields DASL at 58.49% SR, while the average number of skills associated with an instruction decreases from 8.92 to 6.13 (Table 16). The complete design thus couples broad codebook use with more compact language-conditioned assignments and more effective task execution.

Effectiveness of Independent Temporal Parameters. Variant B improves state SR from 28.34% to 38.26% while retaining 19 active codes and reducing the reported skills-per-instruction summary from 8.92 to 5.21. To isolate interval supervision from the forward update schedule, we retain DASL's skill sharing, model capacity, and full-rate fast-policy loss, but allow the slow branch to receive action supervision from only one step per interval. SR falls from 58.49% to 43.12%, active codes from 19 to 15, and normalized entropy from 0.930 to 0.759. This supports the training role of multi-step skill supervision beyond periodic skill retention.

Effectiveness of Latent Trajectory Diffusion. Under shared-horizon coordination, adding diffusion changes state SR from 28.34% (A) to 32.08% (C), while image SR changes from 32.26% to 31.19%. Combining diffusion with independent timescales yields DASL at 58.49% state SR and 50.66% image SR. Relative to Variant C, DASL raises Soft MI from 0.249 to 0.269 while maintaining broad codebook use (19/20 active codes versus 20/20 for C). Its state SR also exceeds the additive reference of 42.00% formed from the separate A-to-B and A-to-C gains. These results show the benefit of combining interval-level supervision with latent trajectory regularization.

4.4 Inference Speed and Computational Cost

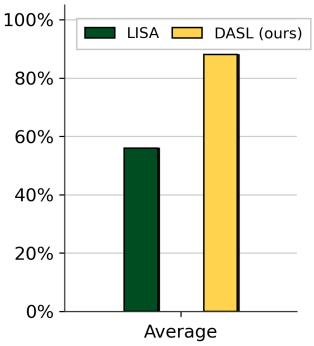
Table 6 compares full-policy inference cost under Eager-FP32 execution. With state inputs, DASL reaches 13.61 Hz, close to LISA's 15.68 Hz and substantially higher than SkillDiffuser's 2.32 Hz. With image inputs, DASL reaches 13.97 Hz and a 50.66% success rate, compared with SkillDiffuser's 2.02 Hz and 39.71%. DASL also uses fewer parameters and less memory for image-based inference: 74.80M parameters and 804 MB, compared with SkillDiffuser's 190.08M and 2422 MB. The diffusion module is active only during training, so the deployed policy predicts actions without iterative denoising. Appendix H.1 further profiles the complete policy stack: history buffering and CUDA Graphs reduce latency from 73.48 ms to 8.029 ms, raising the measured inference frequency from 13.61 Hz to 124.55 Hz. The results show that DASL combines competitive inference speed with stronger task performance, while deployment optimizations further increase its throughput.

Table 6: LOReL inference cost and task success under unoptimized Eager-FP32 execution.

Method	Parameters (M)		Memory (MB)		Inference (Hz)		Success Rate (%)	
	State	Image	State	Image	State	Image	State	Image
LISA	67.00	73.90	764	798	15.68	15.01	19.08	30.14
SkillDiffuser	186.62	190.08	2390	2422	2.32	2.02	33.77	39.71
LADS	67.00	141.30	738	1028	12.44	13.03	49.91	33.30
DASL (ours)	**67.90**	**74.80**	**768**	**804**	**13.61**	**13.97**	**58.49**	**50.66**

4.5 Evaluation on a Realman Manipulator

We evaluate DASL on a Realman manipulator with a fixed RealSense camera, using combined state and image observations. The demonstration set contains 50 trajectories for each of the three atomic tasks and 20 for each of the two compositional tasks; we run 20 trials per task. DASL achieves an average success rate of 88% across all five tasks, compared with 56% for LISA. On closing the drawer, opening the drawer, and picking up the banana, DASL reaches 95%, 90%, and 100%, respectively, compared with 70%, 55%, and 60% for LISA. The largest atomic-task gains occur in picking up the banana and opening the drawer. On the two compositional tasks, DASL achieves 75% for opening and closing the drawer and 80% for opening the drawer and picking up the banana, improving over LISA by 25 and 35 percentage points. These results show that the learned skills support both individual manipulations and their sequential composition on the physical robot.



(a) Average success rate



(b) Compositional Task: open drawer and close drawer



(c) Compositional Task: open drawer and pick up banana from the plate

Figure 5: Real-world quantitative and qualitative results for DASL. (a) compares average success rates. (b)-(c) display successful key-frame sequences for complex compositional tasks.

5 Conclusion

DASL learns reusable discrete skills from language-conditioned demonstrations by coordinating independent semantic and action timescales. Interval-level action supervision and training-only latent diffusion strengthen the learned skill representations, while full-rate observation feedback supports closed-loop execution. Results across LOReL, Kitchen, CALVIN, BabyAI, and physical manipulation show gains in skill use and compositional task performance. Codebook diagnostics, component ablations, and inference measurements further support the effectiveness and practicality of the design. On the real robot, DASL achieves an average success rate of 88% across three atomic manipulation tasks and two compositional tasks.

Limitations and Future Work. DASL is trained from offline demonstrations, and its current evaluation covers a limited set of tasks and skill-library sizes. Future work will study online adaptation, larger structured codebooks, and broader language-conditioned manipulation.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript editing and language polishing, including improving grammar, wording, and readability, as well as manuscript formatting. We reviewed the AI-assisted edits to ensure that they preserve the intended meaning and accurately reflect our methods and results. We take responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies language-conditioned skill learning for robotic manipulation and grid-world navigation using publicly available benchmarks and a constrained real-world robotic setup. Our study does not introduce human-subject experiments or crowdsourcing, and does not collect sensitive personal data. Existing datasets and software are cited and used in accordance with their respective licenses. The empirical evaluation is limited to the benchmark tasks and physical setup described in this paper; these results do not establish safety for unrestricted deployment. Applications beyond these settings should assess physical safety, privacy, and potential misuse, especially when robots operate near people.

REPRODUCIBILITY STATEMENT

Section 3 specifies the two-timescale architecture, vector quantization, action prediction, latent diffusion regularization, and joint training objective. Algorithm 1 in Appendix A traces a training minibatch through slow skill prediction, codebook lookup, skill–observation fusion, fast action prediction, diffusion loss computation, and AdamW updates, including the trajectory masks and temporal parameters. Appendix D.1 describes the encoders and policy modules; Appendix D.2 and Table 12 report dataset-specific batch sizes, iteration counts, horizons, semantic update intervals, codebook settings, diffusion configurations, and optimization parameters. Appendix B documents tasks, observation modalities, language splits, and evaluation metrics, while Appendix C records baseline configurations and result sources. The main repeated evaluations report means and standard deviations over three seeds; component ablations and codebook diagnostics use a common fixed seed, with definitions and matched comparisons in Appendices E.6 and F.7. Appendices E.7, G, and H.1 provide the perturbation protocol, real-robot trial setup, and inference measurements. We plan to release the implementation code after the conference decision.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: A dual-system vla model unifying fast manipulation within slow reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pp. 357–368. PMLR, 2017.

Divyansh Garg, Skanda Vaidyanath, Kuno Kim, Jiaming Song, and Stefano Ermon. Lisa: Learning interpretable skill abstractions from language. *Advances in Neural Information Processing Systems*, 35:21711–21724, 2022.

Ningquan Gu, Yuquan Li, Kazuhiro Kosuge, and Mitsuhiro Hayashibe. Language-driven multi-task manipulation with action-mask-enhanced multimodal learning. *IEEE Robotics and Automation Letters*, 2026.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Haobin Jiang, Jiangxing Wang, and Zongqing Lu. Discrete latent plans via semantic skill abstractions. In *ICLR*, 2025.

Zhaoxun Ju, Chao Yang, Fuchun Sun, Hongbo Wang, and Yu Qiao. Rethinking mutual information for language conditioned skill discovery on imitation learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pp. 301–309, 2024.

Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. Uniskill: Imitating human videos via cross-embodiment skill representations. *arXiv preprint arXiv:2505.08787*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22, 2009.

Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16467–16476, 2024.

Yuchen Liu, Luigi Palmieri, Sebastian Koch, Ilche Georgievski, and Marco Aiello. Delta: Decomposed efficient long-term robot task planning using large language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10995–11001. IEEE, 2025.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. Pmlr, 2020.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pp. 1303–1315. PMLR, 2022a.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022b.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pp. 1838–1849. PMLR, 2023.

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8539–8546. IEEE, 2024.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. PMLR, 2022.

Wenxuan Song, Jiayi Chen, Wenxue Li, Xu He, Han Zhao, Can Cui, Pengxiang Ding Shiyan Su, Feilong Tang, Xuelian Cheng, Donglin Wang, et al. Rationalvla: A rational vision-language-action model with dual system. *arXiv preprint arXiv:2506.10826*, 2025.

Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

Wenhui Tan, Bei Liu, Junbo Zhang, Ruihua Song, and Jianlong Fu. Rold: Robot latent diffusion for multi-task policy modeling. In *International Conference on Multimedia Modeling*, pp. 340–353. Springer, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 537–544. IEEE, 2024.

Ye Wang, Sipeng Zheng, Hao Luo, Wanpeng Zhang, Haoqi Yuan, Chaoyi Xu, Haiweng Xu, Yicheng Feng, Mingyang Yu, Zhiyu Kang, et al. Rethinking visual-language-action model scaling: Alignment, mixture, and regularization. *arXiv preprint arXiv:2602.09722*, 2026.

Ziru Wang, Mengmeng Wang, Jade Dai, Teli Ma, Guo-Jun Qi, Yong Liu, Guang Dai, and Jingdong Wang. Dynamind: Reasoning over abstract video dynamics for embodied decision-making. In *Forty-second International Conference on Machine Learning*, 2025.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

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*, pp. 1094–1100. PMLR, 2020.

Jesse Zhang, Minho Heo, Zuxin Liu, Erdem Biyik, Joseph J Lim, Yao Liu, and Rasool Fakoor. Extract: Efficient policy learning by extracting transferable robot skills from offline data. *arXiv preprint arXiv:2406.17768*, 2024.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 933–946. PMLR, 2025. URL `https://proceedings.mlr.press/v270/zhang25b.html`.

APPENDIX

A PSEUDO-CODE OF TRAINING DASL

Algorithm 1 illustrates the training pseudocode of the DASL algorithm, which describes the main training procedure and the essential steps of the proposed method.

Algorithm 1 Training Procedure of DASL (Dual-Process Atomic Skill Learning)

Input: Expert dataset $\mathcal{D} = \{(l, \mathbf{o}, \mathbf{a}, \mathbf{t}, \mathbf{m})\}$, where l is language, $\mathbf{o}$ observations, $\mathbf{a}$ actions, $\mathbf{t}$ timesteps, and $\mathbf{m}$ binary padding masks; batch size B; slow-frequency interval I; trajectory horizon H; diffusion steps T_d; codebook size M; weights $\lambda_{\text{vq}}, \lambda_{\text{diff}}$.
Initialize: Option Transformer f_{slow}, Codebook $\mathcal{C}$, Decision Transformer π_{fast}, Latent Diffusion ϵ_θ.

Set AdamW optimizer, learning rate scheduler, and gradient clipping threshold.
while not converged **do**
  Sample minibatch $\{(l^i, \mathbf{o}^i, \mathbf{a}^i, \mathbf{t}^i, \mathbf{m}^i)\}_{i=1}^{B} \sim \mathcal{D}$
  $\mathbf{E}^{\text{lang}} \leftarrow \phi_{\text{lang}}(l)$ ▷ Extract language embeddings for goal conditioning

  # 1. High-Level Semantic Skill Planning
  $\mathbf{o}^{\text{slow}}, \mathbf{t}^{\text{slow}}, \mathbf{m}^{\text{slow}} \leftarrow \mathbf{o}_{::I},\ \mathbf{t}_{::I},\ \mathbf{m}_{::I}$ ▷ Subsample at interval I
  $\bar{\mathbf{Z}} \leftarrow f_{\text{slow}}(\mathbf{E}^{\text{lang}}, \mathbf{o}^{\text{slow}}, \mathbf{t}^{\text{slow}}, \mathbf{m}^{\text{slow}})$ ▷ Predict skills conditioned on language
  $\mathbf{Z}, \mathcal{L}_{\text{vq}} \leftarrow \text{VQ}(\bar{\mathbf{Z}}; \mathcal{C})$ ▷ Quantize skills & compute commitment loss

  # 2. Low-Level Skill-Conditioned Decision Transformer
  $\mathbf{E}^{\text{obs}} \leftarrow e_o(\mathbf{o})$ **and** $\mathbf{E}^{\text{skill}} \leftarrow e_z(\mathbf{Z})$ ▷ Encode observations and discrete skills
  $\tilde{\mathbf{E}}^{\text{skill}} \leftarrow \text{TRIMORPAD}(\text{REPEATINTERLEAVE}(\mathbf{E}^{\text{skill}}, I), |\mathbf{E}^{\text{obs}}|)$ ▷ Upsample skills to match obs
  $\mathbf{E}^{\text{fused}} \leftarrow \mathbf{E}^{\text{obs}} + \tilde{\mathbf{E}}^{\text{skill}}$ ▷ Element-wise embedding fusion
  $\hat{\mathbf{a}} \leftarrow \pi_{\text{fast}}(\mathbf{E}^{\text{fused}}, \mathbf{a}, \mathbf{t}, \mathbf{m})$ ▷ Autoregressive action prediction
  $\mathcal{L}_{\text{action}} \leftarrow \text{MSE}(\hat{\mathbf{a}}, \mathbf{a}; \mathbf{m})$ ▷ Action prediction loss masked by $\mathbf{m}$

  # 3. Latent Trajectory Diffusion Regularization
  if Latent Diffusion ϵ_θ is enabled **then**
    $H_{\text{diff}} \leftarrow \min(H, |\mathbf{E}^{\text{fused}}|)$
    $\tau_0 \leftarrow \mathbf{E}^{\text{fused}}_{1:H_{\text{diff}}}$ **and** $\mathbf{Z}_{H_{\text{diff}}} \leftarrow \text{TRIMORPAD}(\text{REPEATINTERLEAVE}(\mathbf{E}^{\text{skill}}, I), H_{\text{diff}})$
    Sample step $n \sim \mathcal{U}(1, T_d)$ and noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Bold $\mathbf{I}$ for Identity matrix
    $\tau_n \leftarrow \sqrt{\bar{\alpha}_n}\tau_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon$ ▷ Forward noise injection
    $\mathcal{L}_{\text{diff}} \leftarrow \|\epsilon - \epsilon_\theta(\tau_n, n, \mathbf{Z}_{H_{\text{diff}}})\|_2^2$ ▷ Reverse denoising loss
  else
    $\mathcal{L}_{\text{diff}} \leftarrow 0$
  end if

  # 4. Joint Optimization
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{action}} + \lambda_{\text{vq}}\mathcal{L}_{\text{vq}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}}$
  BACKPROP($\mathcal{L}_{\text{total}}$); CLIPGRADIENTS(); OPTIMIZERSTEP()
end while

B DATASETS DESCRIPTIONS.

B.1 LOREL SAWYER.

LOReL Sawyer Dataset. The LOReL (Language-conditioned Offline Reward Learning) Sawyer dataset comprises pseudo-expert trajectories or play data collected from arbitrary Reinforcement Learning (RL) policies, annotated with hindsight crowdsourced language instructions. Developed on the Meta-World benchmark, this dataset features a Sawyer robot interacting with a drawer, a faucet, and two mugs (see Figure 6a). It contains 50,000 trajectories—each with a horizon of 20 time steps—collected by random RL policies within the environment and labeled using procedurally generated hindsight language instructions. We utilize the same six tasks as the original paper to evaluate our method. Performance is assessed across five distinct scenarios: seen instructions, unseen verbs, unseen nouns, unseen verbs+nouns, and human-provided instructions. Examples of these linguistic variations are provided in Table 7. Collectively, the six tasks encompass 77 instructions.

Table 7: LOReL Example rephrasals for the instruction.

Seen	Unseen Verb	Unseen Noun	Unseen Verb + Noun	Human Provided
close drawer	shut drawer	close container	shut container	push the drawer shut
open drawer	pull drawer	open container	pull container	pull the drawer open
turn faucet left	rotate faucet left	turn tap left	rotate tap left	rotate nozzle left
turn faucet right	rotate faucet right	turn tap right	rotate tap right	rotate nozzle right
move black mug right	push black mug right	move dark cup right	push dark cup right	push black cup right
move white mug down	push white mug down	move light cup down	push light cup down	bring white cup down

Evaluation metrics and aggregation. For random seed s, let $r_{s,c}$ be the success rate for language category c and $q_{s,j}$ the success rate for physical task j. We compute the rephrasal and task macro-averages as

$$R_s^{\text{reph}} = \frac{1}{5}\sum_{c=1}^{5} r_{s,c}, \qquad R_s^{\text{task}} = \frac{1}{6}\sum_{j=1}^{6} q_{s,j}. \tag{9}$$

The five language categories are seen, unseen noun, unseen verb, unseen noun+verb, and human-provided language; the six manipulation tasks are listed below. For experiments repeated with three random seeds, we report the mean and standard deviation of the corresponding seed-level macro-averages. Table 1 gives three-seed macro-averages over the five language categories (54.22%/50.13%); Table 13 gives the six-task macro-average (53%/47%). Tables 5 and 6 use the common fixed-seed component/efficiency evaluation (58.49%/50.66% for DASL). The frequency-ratio sweep reports the six-task mean for each tested ratio; the selected 1:2 setting yields 54.51%/50.96%. The codebook-size sweep reports results by codebook size, with the selected $K = 20$ setting at 54.51%/50.96%. The action-generation comparison reports 54.51%/50.96% for its Decision Transformer configuration, whereas latent-regularization and skill-discretization tables report state-only DASL results of 54.51%. Table 24 reports success by horizon and extrapolation strategy; 61.9% is Clamp at the in-distribution horizon. These tables therefore summarize different groupings, configurations, and aggregation protocols: language-category averages, task averages, fixed-seed component/efficiency results, sweep results, and horizon-specific results.

Composition Instructions. For the composition instructions, we took these evaluation instructions from the original paper and combined them to form 12 new composition instructions, as detailed in Table 8.

Table 8: LOReL Composition tasks

Composite Instructions
open drawer and move black mug right
pull the handle and move black mug down
move white mug right
move black mug down
close drawer and turn faucet right
close drawer and turn faucet left
turn faucet left and move white mug down
turn faucet right and close drawer
move white mug down and turn faucet left
close the drawer, turn the faucet left and move black mug right
open drawer and turn faucet counterclockwise
slide the drawer closed and then shift white mug down

B.2 FRANKA-KITCHEN.

Franka-Kitchen Dataset. Kitchen benchmark Gupta et al. (2019) is a physics-based simulation environment built upon the MuJoCo engine Todorov et al. (2012), where a Franka robotic manipulator operates in a kitchen scenario containing multiple interactive objects, including a microwave, a kettle, a sliding cabinet, a hinged cabinet, a switch, and two stove burners (see Figure 6b). The robot is actuated through joint position control, and the environment provides both low-dimensional state

observations and RGB visual inputs. This dataset comprises 566 human demonstrations collected via VR teleoperation. In each trajectory, the demonstrator sequentially performs four manipulation actions on different kitchen components.

Language Labeling. Given the absence of native linguistic labels in the base environment, we systematically synthesize natural language instructions for the demonstrations. Each trajectory is characterized by a sequence of four executed sub-tasks, which we directly map to their corresponding atomic phrases (detailed in Table 9). To construct the final composite instruction for an entire episode, these four atomic descriptions are chronologically chained together using the conjunction "and".

Table 9: Atomic instructions labeled to sub-tasks.

Sub-tasks	**Atomic Instructions**
bottom burner	activate bottom burner
top burner	activate top burner
light switch	turn on light switch
slide cabinet	open sliding cabinet
hinge cabinet	open left hinge cabinet
microwave	open microwave door
kettle	move kettle to top left burner

Seen and Unseen Instructions. To evaluate compositional generalization, we partition the dataset by holding out three specific composite instructions, leaving 509 demonstrations across 22 distinct instructions for training. Crucially, while the specific combinations are unseen during testing, every individual sub-task is thoroughly represented in the training distribution. We utilize the following three held-out instructions for evaluation:

- *Open microwave door and move kettle to top left burner and activate bottom burner and turn on light switch.*
- *Move kettle to top left burner and activate bottom burner and open sliding cabinet and open left hinge cabinet.*
- *Move kettle to top left burner and activate top burner and turn on light switch and open sliding cabinet.*

These evaluation tasks are strategically designed to present an escalating level of compositional difficulty. Specifically, the training corpus contains trajectories that perfectly match the *first three* sub-tasks of the first instruction. For the second instruction, this overlap is reduced to only the *first two* sub-tasks. The third instruction poses the most rigorous out-of-distribution challenge, as the training data lacks any sequences sharing even its two-task prefix. Finally, to systematically assess compositional capabilities, we quantify model performance by counting the number of sub-tasks successfully executed within a strict 280-step episode limit.

B.3 CALVIN

CALVIN Dataset. CALVIN (Mees et al., 2022) is a language-conditioned robotic manipulation benchmark featuring a 7-DOF Franka Emika Panda robot (Figure 6c). Following the six-task CALVIN-D state-based setting of LCSD (Ju et al., 2024), we evaluate drawer opening and closing, lightbulb switching on and off, and slider movement to the left and right. We select the corresponding CALVIN-D trajectories and encode the 21-dimensional state observations with an MLP.

Language Instructions. CALVIN provides varied phrasings of manipulation tasks for training and evaluation. Table 10 gives examples of these instructions.

B.4 BABYAI.

BabyAI Dataset. BabyAI presents an environment where an agent navigates and interacts within a grid world to achieve goals described via natural language. The dataset includes various environmental configurations where level difficulty and instruction complexity increase progressively. Each level

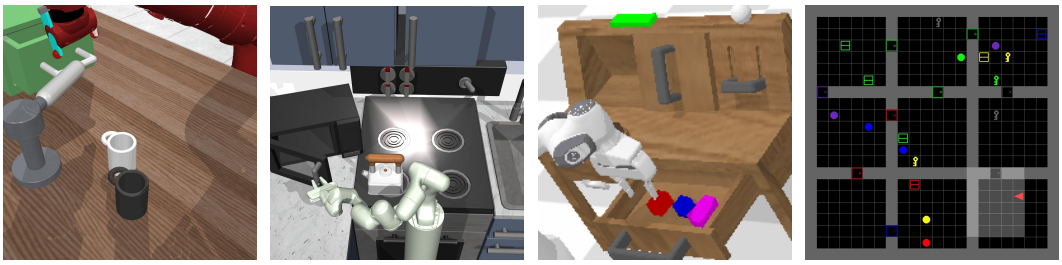
Table 10: Examples of CALVIN training and evaluation instructions.

Category	Detail Tasks	Training Language	Evaluation Language	Num
lightbulb	turn on/off lightbulb	toggle the light switch to turn on/off the light bulb turn on/off the light bulb move the light switch to turn on/off the light bulb	use the switch to turn on/off the light bulb	95
drawer	open/close drawer	open/close the cabinet drawer grasp the drawer handle and open/close it pull/push the handle of the drawer	pull/push the handle to open/close the drawer	303
slider	move slider left/right	grasp the door handle, then slide the door to the left/right move the door all the way to the left/right slide/push the door to the left/right	push the sliding door to the left/right side	382
light	push down the button to turn on/off the led	press the button to turn on/off the led light turn on/off the led light toggle the button to turn on/off the led light	push the sliding door to the left/right side	382
Unknown tasks	Turn on/off green/yellow lamp move/toggle the light switch to turn off the yellow/green light			227

is situated in a grid world where the agent observes a partially visible 7×7 square area from an egocentric perspective. While the dataset provides one million expert trajectories for each of the 19 levels, we evaluate DASL across varying data regimes, specifically utilizing subsets of 1k, 10k, and 100k trajectories. We evaluate our policy on 100 distinct instructions within the Gym environment for each level, covering unseen layouts and language instructions to assess generalization capabilities.

GoToSeq Task. This task requires the agent to execute a sequence of go-to-object commands (as illustrated in Figure 6c).

- **Example command:** *"go to a box and go to the purple door, then go to the grey door"*
- **Average demo length:** 72.7 ± 52.2 steps



(a) LOReL-Sawyer (b) Franka Kitchen (c) CALVIN (d) BabyAI GoToSeq

Figure 6: Simulation environments for manipulation and navigation tasks.

C Baselines Descriptions.

LCBC. Language-Conditioned Behavioral Cloning (LCBC) serves as a supervised imitation learning baseline that directly maps language instructions and observations to actions using demonstration trajectories. As a straightforward language-conditioned policy, it relies neither on hierarchical planning nor reinforcement learning. We directly report the success rate of LCBC on the BabyAI GoToSeq task from Garg et al. (2022).

Lang DT. Language Decision Transformer (langDT) extends the standard Decision Transformer by conditioning the transformer policy on language instructions alongside state and action histories. It models the sequential decision-making process in a flat manner, operating without explicit skill abstractions or hierarchical structures. For performance comparison, we directly report the success rates of Lang DT on the LOReL(image) tasks from Liang et al. (2024), on the Franka-Kitchen benchmark from Jiang et al. (2025), on the CALVIN benchmark from Ju et al. (2024) and on the BabyAI GoToSeq task from Garg et al. (2022). Since Liang et al. (2024) do not provide the results for the LOReL(state) tasks, we evaluate their official implementation (`https://github.com/Div99/LISA`) to obtain the corresponding metrics.

LISA. Learning Interpretable Skill Abstractions (LISA) employs a vector-quantization bottleneck to learn discrete skills from language-conditioned demonstrations and uses them to condition a low-level policy. We report its published LOReL image results from Liang et al. (2024), Franka Kitchen results from Jiang et al. (2025), CALVIN results from Ju et al. (2024), and BabyAI results from Garg et al. (2022). Because Liang et al. (2024) do not report LOReL state results, we evaluate the same official

implementation for that modality. Codebook usage for evaluated methods is summarized separately in Appendix E.6.

HULC. Hierarchical Universal Language Conditioned (HULC) learns language-conditioned manipulation policies through multi-context imitation learning (Mees et al., 2022). For the six-task CALVIN state-input evaluation, we report the MLP-based adaptation and success rate from Ju et al. (2024).

LCSD. Language-Conditioned Skill Discovery (LCSD) focuses on learning discrete skill representations by maximizing the mutual information between linguistic instructions and latent skills. This objective enables the agent to acquire a structured skill space, facilitating compositional generalization by associating each unique instruction with a corresponding learned skill embedding. The success rates of LCSD on both LOReL(state) and LOReL(image) are directly reported from the original study by Ju et al. (2024).

SkillDiffuser. SkillDiffuser is a diffusion-based hierarchical policy that generates latent skill trajectories conditioned on both language instructions and environmental observations. The framework utilizes a low-level controller to execute actions derived from predicted skill sequences, facilitating robust compositional planning through flexible skill composition. We directly report the success rates of SkillDiffuser on LOReL(image) from the original study by Liang et al. (2024). As LOReL(state) results were not provided in Liang et al. (2024), we conducted an independent evaluation using the official source code (`https://github.com/Liang-ZX/SkillDiffuser`) to obtain the corresponding metrics.

LADS. LADS learns a discrete latent action space to connect language-conditioned planning with low-level skill execution. We report its Franka Kitchen results from Jiang et al. (2025). For LOReL, we evaluate the official implementation (`https://github.com/PKU-RL/LADS`) to obtain success rates for state and image observations across the instruction-rephrasal categories.

Table 11 lists the evaluation source for each baseline across the benchmarks used in this work.

Table 11: Sources of baseline results across the evaluation benchmarks.

Benchmark / modality	Baselines	Source	Reference or implementation
BabyAI	LCBC, LangDT, LISA	Published	Garg et al. (2022)
Kitchen state/image	LangDT, LISA, LADS	Published	Jiang et al. (2025)
CALVIN six-task state	LangDT, LISA, HULC, LCSD	Published	Ju et al. (2024); includes HULC's adapted state/MLP setting
LOReL image	LangDT, LISA, SkillDiffuser	Published	Liang et al. (2024)
LOReL state/image	LCSD	Published	Ju et al. (2024)
LOReL state	LangDT, LISA	Our runs	Official LISA implementation, cited above
LOReL state	SkillDiffuser	Our runs	Official SkillDiffuser implementation, cited above
LOReL state/image	LADS	Our runs	Official LADS implementation

D Training Details

D.1 Network Architecture

Language Encoder. We employ a pre-trained DistilBERT as the language encoder, maintaining a configuration consistent with LOReL and LISA, while freezing its parameters to ensure the stability of language understanding. Given a natural language instruction l, it is encoded into a word embedding sequence $\mathbf{E}^{\text{lang}} = \text{DistilBERT}(l) \in \mathbb{R}^{L_{\text{lang}} \times d}$, where L_{lang} is the token length and d is the shared hidden dimension. This helps preserve the high-level semantic information of the language instructions.

Observation Encoder. We consider both vector observations and pixel observations. For vector inputs (e.g., proprioceptive states or precomputed visual embeddings), we apply a linear projection to match the transformer hidden dimension d. For image inputs, we use a lightweight convolutional encoder ϕ (following prior LOReL-style implementations) to map each frame into a fixed-dimensional embedding.

Slow Frequency Policy. The slow policy acts as a semantic predictor that produces a high-level latent skill at a lower temporal frequency, updating once every I environment steps (the fixed interval). Specifically, we employ an Option Transformer that predicts continuous skill embeddings by reasoning over both the language instruction and the subsampled observation sequence. These continuous skills are subsequently discretized via a vector-quantization (VQ) bottleneck, which is optimized using a standard commitment regularization.

High Frequency Policy. The high frequency policy is a Decision Transformer that outputs low-level actions at every high-frequency time step. Using a GPT-2 backbone, it interleaves timestep-encoded action and fused observation embeddings $\mathbf{E}^{\text{fused}}$ to predict the action $\hat{a}_t$ at each timestep. During deployment, the policy executes the action predicted at the final position of the sliding context window L. In our DASL framework, this transformer is explicitly conditioned on the selected discrete skill fused with the observation-action sequence. To align the high frequency policy's temporal receptive field with the slow semantic schedule, the retrieved skill embedding is repeated I times to remain constant within each interval segment. Furthermore, to structure the latent manifold and ensure robust semantic alignment, we augment this low-level policy with an auxiliary latent diffusion module exclusively during training. This hybrid design ensures consistent low-level execution guided by stable high-level semantics, without compromising real-time inference efficiency.

D.2 Training Hyperparameters.

Table 12 details the comprehensive hyperparameter configurations for DASL across all environments, including LOReL, Franka Kitchen, BabyAI, and CALVIN. To ensure stable convergence, we consistently employ the AdamW optimizer with a $1e^{-5}$ learning rate and $1e^{-4}$ weight decay across all tasks. Key structural parameters, such as the sequence horizon are scaled by environment complexity, ranging from 5 in LOReL (state) to 20 in Franka Kitchen. Notably, the diffusion module (configured with 100 timesteps and 4 blocks) is enabled across all settings to regularize the latent space. Batch sizes and iteration counts are tuned per dataset to balance computational efficiency with high-level semantic alignment.

Table 12: **Hyperparameters on Different Datasets**

Hyperparameter	LOReL (state)	LOReL (image)	Kitchen (state)	Kitchen (image)	BabyAI	CALVIN
Batch Size	512	512	16	16	128/64/180	24
Learning Rate	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$
Max Iterations	2500	2500	3000	3000	2500	5000
Horizon	5	10	20	20	10	10
Slow Frequency Interval	2	2	20	20	2	5
Option Dimension	128	128	128	128	128	128
Number of Options	20	20	20	30	20	20
Codebook Dimension	16	16	16	16	16	16
Diffusion Enabled	True	True	True	True	True	True
Diffusion Loss Weight	0.5	0.01	0.1	0.1	0.1	0.1
Diffusion Timesteps	100	100	100	100	100	100
Diffusion Blocks	4	4	4	4	4	4
Warmup Steps	500	500	500	500	500	500
Evaluation Frequency	20	20	20	20	100	100
Evaluation Episodes	50	50	50	50	50	50
Weight Decay	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW

E Supplementary Results

E.1 Additional Experimental Results on LOReL

Table 13 reports the six-task LOReL macro-average, yielding 53% for state and 47% for image observations. This task-wise summary complements the five-category rephrasal macro-average in

Table 1. DASL achieves the highest task-average success rate in both modalities. Appendix B.1 defines both averaging schemes.

Table 13: **Task success rates (%) on LOReL Sawyer.** Average is the macro-average across six tasks. Blue zero standard deviations denote cited results with unreported uncertainty; they are placeholders, not measured deviations. Best/second-best values are bold/underlined.

Task Instruction	Obs	LCBC	Lang DT	LISA	SkillDiffuser	LADS	DASL(ours)
close drawer	State	45 ± 11	52 ± 14	52 ± 41	80 ± 6	83 ± 19	**96 ± 1**
	Image	50 ± 0	10 ± 0	**100 ± 0**	95 ± 3	62 ± 18	50 ± 13
open drawer	State	0 ± 0	33 ± 20	4 ± 6	31 ± 14	**84 ± 12**	44 ± 12
	Image	0 ± 0	**60 ± 0**	20 ± 0	55 ± 13	29 ± 19	46 ± 12
turn faucet left	State	25 ± 18	48 ± 22	7 ± 8	37 ± 10	27 ± 25	**68 ± 2**
	Image	12 ± 0	0 ± 0	0 ± 0	**55 ± 9**	27 ± 8	40 ± 2
turn faucet right	State	30 ± 7	12 ± 6	1 ± 1	34 ± 6	25 ± 21	**43 ± 17**
	Image	31 ± 0	0 ± 0	30 ± 0	25 ± 4	41 ± 17	**50 ± 7**
move black mug right	State	**65 ± 4**	50 ± 13	0 ± 1	17 ± 2	55 ± 23	55 ± 11
	Image	**73 ± 0**	20 ± 0	60 ± 0	18 ± 4	53 ± 8	61 ± 26
move white mug down	State	5 ± 8	10 ± 5	0 ± 0	10 ± 6	**35 ± 3**	11 ± 10
	Image	6 ± 0	0 ± 0	30 ± 0	10 ± 2	29 ± 7	**33 ± 3**
Average	State	28 ± 5	34 ± 7	11 ± 7	35 ± 4	52 ± 2	**53 ± 2**
	Image	29 ± 0	15 ± 0	40 ± 0	43 ± 1	40 ± 4	**47 ± 2**

A fine-grained analysis of individual tasks reveals that DASL excels in maneuvers requiring sustained directional control and precise spatial execution. For instance, in the state modality of the turn faucet left and turn faucet right tasks, DASL achieves 68% and 43% success rates respectively, whereas baselines like LISA completely fail, often scoring near zero. Furthermore, in tasks demanding decisive endpoint manipulation, such as close drawer, DASL reaches a near-perfect 96% success rate under state observations. This consistency across diverse instruction types underscores the framework's adaptability to both continuous control semantics and discrete state changes.

E.2 Qualitative Results in LOReL

Figure 7 illustrates a qualitative execution sequence of the DASL framework performing a multi-step, compositional instruction—"open drawer and move black mug right"—within the LOReL State environment. The sequence demonstrates the capability of the model to parse and execute complex hierarchical commands. Panels (a) through (c) depict the robot arm locating and fully opening the drawer; subsequently, panels (d) and (e) show the agent transitioning to the second sub-task to manipulate the black mug. This visual trajectory validates the effectiveness of the proposed hierarchical architecture and the latent skill representations in maintaining semantic consistency and execution stability across multi-stage manipulation tasks.



(a) (b) (c) (d) (e)

Figure 7: LOReL State: open drawer and move black mug right

Figure 8 presents another qualitative execution sequence of the DASL framework on the LOReL Image dataset for the multi-step instruction—"turn faucet right and close drawer". The panels depict the transition between distinct sub-tasks: panels (a) through (c) show the robot arm interacting with the faucet handle, followed by panels (d) and (e), which display the closing of the drawer. This visual

trajectory confirms the ability of DASL to maintain semantic consistency and execution stability across compositional manipulation tasks using purely image-based observations.



(a) (b) (c) (d) (e)

Figure 8: LOReL Image: turn faucet right and close drawer

E.3 Additional Experimental Results on Kitchen

Tables 14 and 15 report K-rates for completing up to four subtasks in Franka Kitchen with state and image observations. DASL achieves the highest image-based $K = 1, 2$ completion rates on both seen and unseen instructions. On state inputs, LADS has higher seen/unseen N-rates of 2.25/2.23, compared with DASL at 1.66/1.76. All compared image policies have $K = 4$ completion below 5%, indicating remaining difficulty with full four-subtask execution. The 22-training/3-held-out combination split and familiar-prefix overlaps are documented in Appendix B.2.

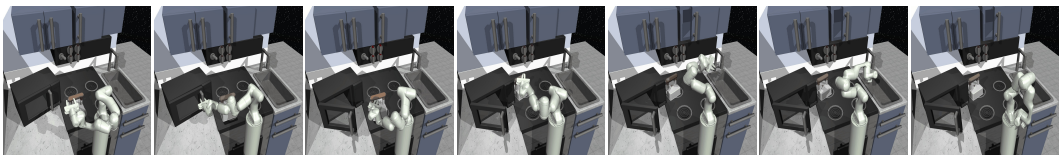
Table 14: K-rates of different methods on seen and unseen instructions in Kitchen (state)

Methods	**seen**				**unseen**			
	1	2	3	4	1	2	3	4
Lang DT	83.1 ± 3.2	40.0 ± 7.4	12.6 ± 4.7	3.0 ± 1.6	83.3 ± 6.4	40.9 ± 20.8	1.1 ± 0.8	0.0 ± 0.0
LISA	52.5 ± 28.1	16.9 ± 14.4	1.6 ± 2.1	0.0 ± 0.0	51.1 ± 33.0	19.8 ± 19.4	0.0 ± 0.0	0.0 ± 0.0
LADS	**98.4 ± 1.1**	**79.9 ± 5.8**	**39.3 ± 5.9**	**6.9 ± 1.1**	**99.8 ± 0.4**	**80.9 ± 7.4**	**41.8 ± 7.7**	**0.9 ± 1.0**
DASL	97.3 ± 1.8	58.7 ± 1.1	9.7 ± 1.9	0.0 ± 0.0	97.8 ± 3.9	77.8 ± 15.4	0.0 ± 0.0	0.0 ± 0.0

Table 15: K-rates of different methods on seen and unseen instructions in Kitchen (image)

Methods	**seen**				**unseen**			
	1	2	3	4	1	2	3	4
Lang DT	73.9 ± 20.4	42.7 ± 14.5	12.9 ± 5.1	2.0 ± 0.9	78.0 ± 12.7	25.6 ± 18.0	2.9 ± 2.1	0.0 ± 0.0
LISA	62.3 ± 8.6	20.8 ± 3.2	3.0 ± 0.9	0.0 ± 0.0	88.9 ± 7.4	34.4 ± 11.7	5.8 ± 2.1	0.2 ± 0.4
LADS	90.6 ± 10.3	55.5 ± 17.5	14.3 ± 4.9	**4.1 ± 4.4**	91.6 ± 7.8	47.3 ± 18.3	**4.9 ± 1.5**	**0.4 ± 0.4**
DASL	**94.6 ± 1.8**	**60.0 ± 5.1**	**17.9 ± 4.2**	0.30 ± 0.53	**95.6 ± 7.7**	**71.1 ± 3.9**	4.44 ± 7.7	0.0 ± 0.0

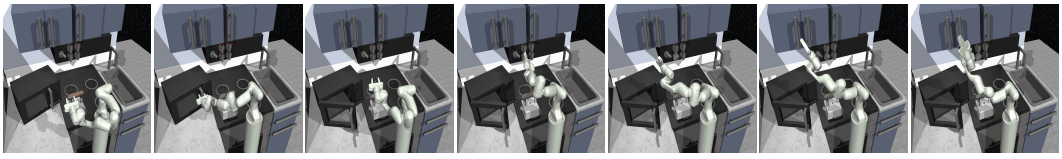
Figure 9 presents a qualitative execution sequence in the Kitchen (state), where the DASL agent successfully performs a complex, compositional task comprising four sequential sub-tasks: activating the bottom burner, activating the top burner, turning on the light switch, and opening the sliding cabinet. Panels (a) through (g) illustrate the robot arm's precise manipulation and smooth transitions between diverse interaction points within the high-dimensional kitchen workspace. This successful execution of a multi-stage trajectory highlights the framework's robustness in maintaining long-term semantic alignment and stable control in intricate, -inspired environments.



(a) (b) (c) (d) (e) (f) (g)

Figure 9: Kitchen State: activate bottom burner and activate top burner and turn on light switch and open sliding cabinet

Figure 10 illustrates a qualitative execution sequence of the DASL framework in the Kitchen (image). The sequence showcases the agent's successful completion of a complex compositional task consisting of four sub-tasks: activating the bottom burner, activating the top burner, turning on the light switch, and opening the sliding cabinet. Panels (a) through (g) highlight the agent's ability to maintain semantic alignment and precise control across multiple manipulation points while operating directly from visual inputs in a high-dimensional workspace.

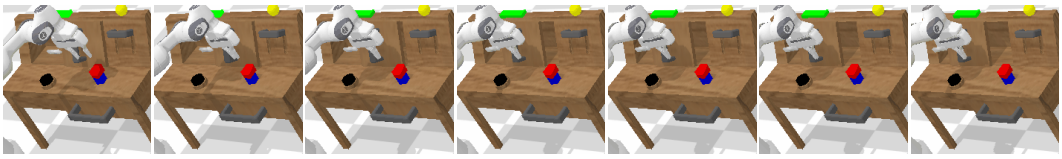


(a) (b) (c) (d) (e) (f) (g)

Figure 10: Kitchen Image: activate bottom burner and activate top burner and turn on light switch and open sliding cabinet

E.4 Additional Experimental Results on CALVIN

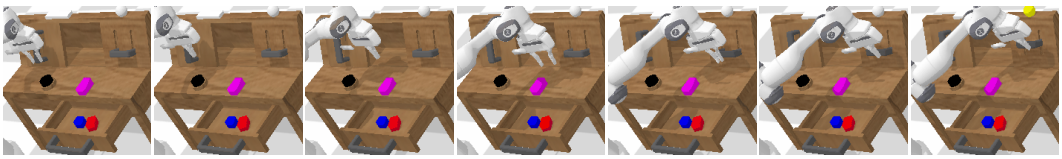
Figure 11 shows a qualitative execution sequence in the CALVIN environment, where the DASL completes a language-conditioned task: moving the green slider to the left. Panels (a) through (g) capture the continuous motion of the robotic arm as it interacts with the articulated object on the tabletop. The successful completion of this fine-grained trajectory demonstrates the capability of the framework to maintain accurate spatial alignment and stable low-level control in contact-rich simulations.



(a) (b) (c) (d) (e) (f) (g)

Figure 11: CALVIN: push the sliding door to the left side

Figure 12 shows a qualitative execution sequence in the CALVIN environment, where the DASL agent completes a language-conditioned task: turning on the lightbulb. Panels (a) through (g) capture the continuous motion of the robotic arm as it interacts with the light switch on the tabletop. The successful completion of this trajectory demonstrates the capability of the framework to maintain accurate spatial alignment and stable low-level control during contact-rich physical interactions.



(a) (b) (c) (d) (e) (f) (g)

Figure 12: CALVIN: use the switch to turn on the light bulb

E.5 Visualization.

Skill Heat Map and Codebook Collapse. Figure 14 contrasts LISA's concentrated activations with DASL's distributed word–skill associations. Visual inspection identifies 14 semantically interpretable skills in DASL's learned vocabulary, with distinct associations between language and skill.

Word Clouds. Figure 13 presents word clouds for the atomic skills learned on the LOReL Sawyer state-based compositional tasks. The visualization identifies 14 semantically interpretable skills

associated with task-specific language. For example, **Skill 5** is associated with the token ‘mug’ and the instruction ‘move white mug right’. **Skill 1** corresponds to ‘turn faucet right’ and is associated with the tokens ‘turn’ and ‘right’, as well as fragmented sub-word tokens (e.g., ‘fa’, ‘##uce’, ‘##t’).

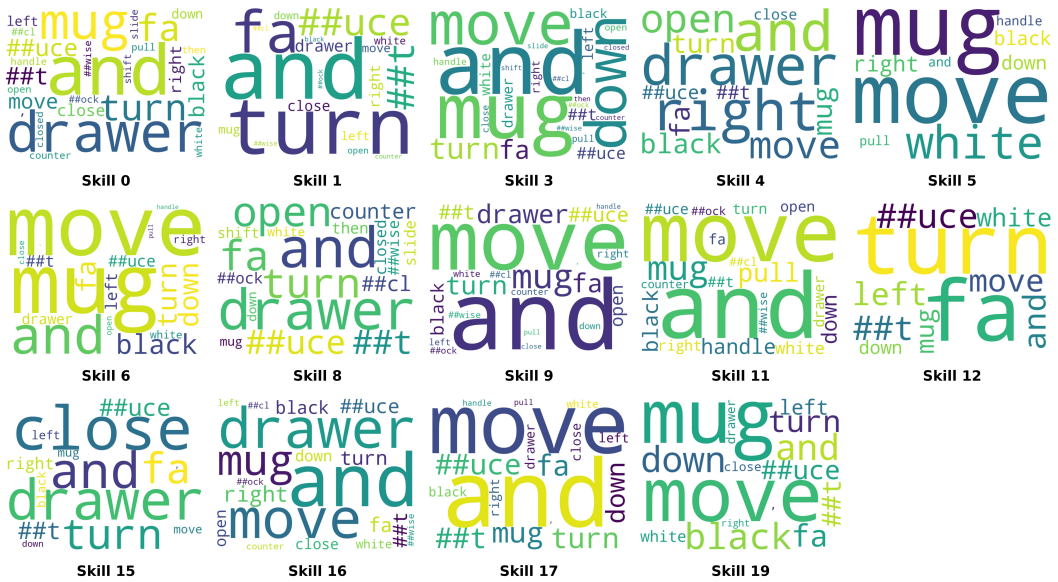
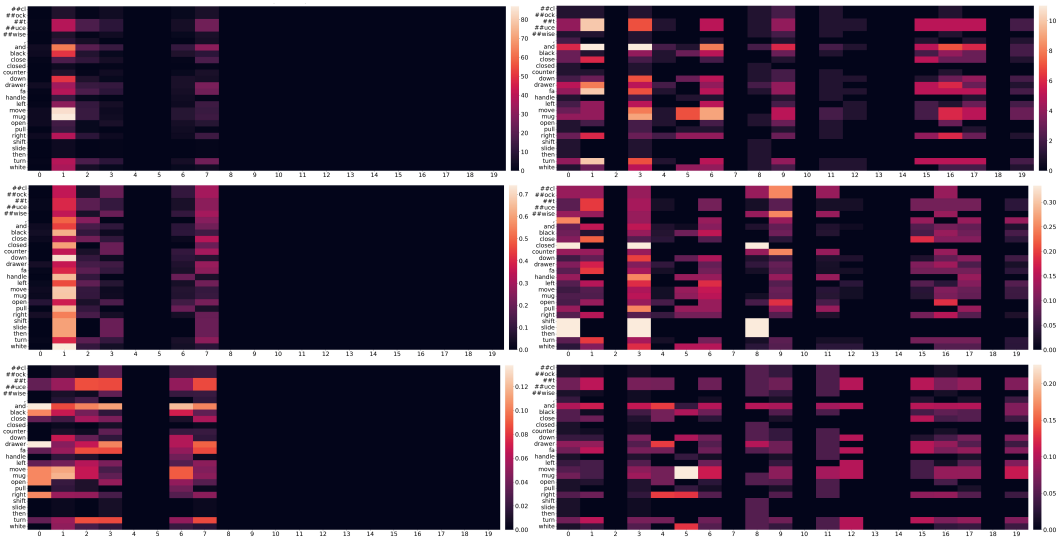


Figure 13: Word cloud of skills learned in LOReL Sawyer (state) compositional tasks.



(a) LISA **(b) DASL (Ours)**

Figure 14: Qualitative comparison of learned skill representations for LISA and DASL on LOReL Sawyer tasks. The rows display code correlation (Word $\times$ Skill), option frequency (normalized row-wise), and word frequency (normalized column-wise) for a size-20 skill set. Column (a) illustrates significant index collapse in LISA, where activations are limited to a small subset of indices like 1 and 7. In contrast, the DASL heatmaps in column (b) show marked sparsity and distinct activation regions across the codebook. These disentangled associations between linguistic tokens and atomic skills highlight the superior interpretability and utilization of our framework. (Zoom in for best view).

Skill Selection. Visual analysis of Figure 15 confirms the capability of the hierarchical framework to perform autonomous task decomposition across varying time horizons. For the instruction turn

faucet right and close drawer, the model invokes a sequence of discrete skill indices [16, 16, 4, 9, 1, 1, 6], where transitions in the latent indices directly correspond to the physical shift from faucet manipulation to drawer closure. A similar adaptive behavior is observed in the second episode with the sequence [0, 18, 12] for the task of opening the drawer and moving the mug. This precise mapping between temporal sub-goals and discrete skill vectors suggests that the learned codebook effectively encapsulates the prerequisite atomic actions required for multi-stage manipulation, ensuring semantic consistency throughout the trajectory.

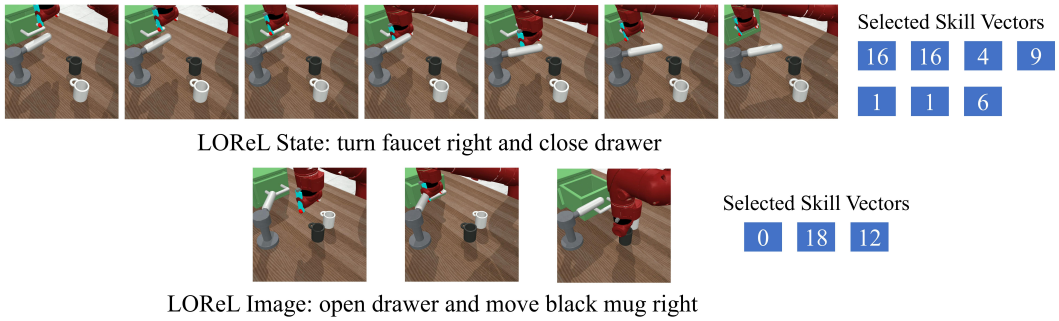
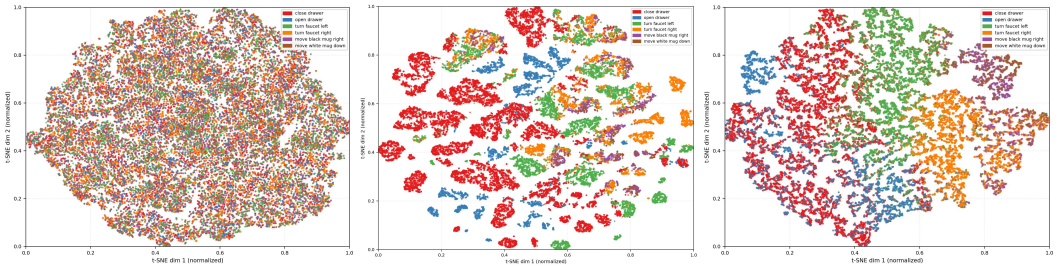


Figure 15: Visualize two LOReL episodes with aligned keyframes and the discrete skill index selected at each timestep.

Latent Skill Analysis. Figure 16 illustrates the evolution of the latent space structure during hierarchical learning. The raw expert states in panel (a) are heavily entangled, making various task distributions nearly indistinguishable. Continuous skill modeling leads to the emergent clustering seen in panel (b), where broad task categories begin to diverge. Despite this, these pre-quantization embeddings still have diffuse boundaries and significant overlap, which limits the precision of semantic grounding. In contrast, the fused trajectories in panel (c) show a sharp increase in cluster separation and density. By using a vector quantization bottleneck and the DASL fusion mechanism, the framework compresses diffuse embeddings into compact, semantically explicit clusters. This structural refinement ensures that each atomic skill occupies a unique region in the latent manifold.



(a) Raw Expert States (b) Continuous Skills (Pre-VQ) (c) Fused Trajectories (Ours)

Figure 16: Evolution of latent space representations across different stages. t-SNE visualizations illustrate the progression from (a) raw expert states with high entanglement to (b) continuous skill primitives prior to vector quantization (Pre-VQ), and finally to (c) our fused trajectories. The transition demonstrates how the DASL framework effectively transforms diffuse continuous embeddings into semantically distinct and well-separated clusters, significantly enhancing representational discriminability.

To validate DASL's capability to compose atomic skills for solving compositional manipulation tasks, we visualized the distributions of composite and atomic instructions on the LOReL Sawyer (state-based) dataset. Figure 17 illustrates the frequency distribution of skill codebook usage for the composite task "close drawer and turn faucet right" alongside its constituent atomic tasks. We observe a significant distributional alignment between the composite task distribution (green solid line) and the arithmetic mean of the atomic task distributions (black dashed line). Notably, the high-probability

modes corresponding to Atomic Task A (indices 9–11) and Atomic Task B (indices 5–8) are explicitly preserved in the composite distribution, while task-irrelevant skill codes in low-probability regions (indices 0–4 and 12) are successfully suppressed. This indicates that DASL captures reusable latent atomic skills and effectively solves compositional tasks by composing these atomic primitives.

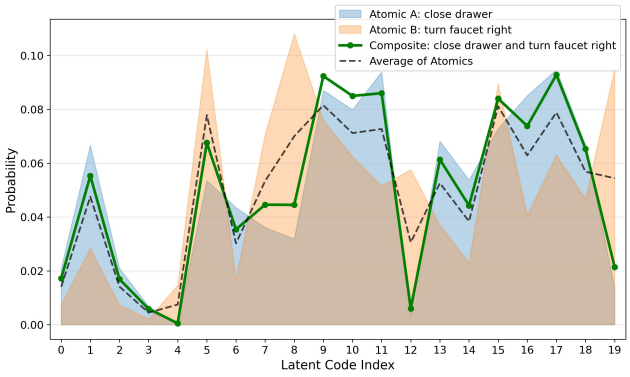


Figure 17: Visualization of Latent Skill Distributions. Comparing skill codebook usage between composite and atomic skills.

E.6 Quantitative Codebook Diagnostics

We report direct diagnostics for the component variants on LOReL-state with a 20-entry codebook (Table 16). Active Codes counts entries used at least once. For usage counts n_m and $p_m = n_m / \sum_j n_j$, normalized usage entropy and the most-used-code ratio are

$$A = \sum_{m=1}^{20} \mathbf{1}[n_m > 0], \quad H_{\text{norm}} = -\frac{\sum_{m=1}^{20} p_m \log p_m}{\log 20}, \quad R_{\text{top1}} = \max_m p_m, \tag{10}$$

with $0 \log 0 = 0$. Soft MI denotes the mutual-information diagnostic between language instructions and soft skill assignments. Avg. Skills / Instruction summarizes the spread of language-conditioned assignments: low values may indicate collapse, while high values may indicate diffuse organization. These measures are interpreted jointly with SR. All component variants use the same fixed seed and training configuration except for the tested components.

Table 16: LOReL-state codebook diagnostics for component and interval-supervision ablations. IS denotes interval supervision; the codebook has 20 entries. The 14 visually interpretable skills in Appendix E.5 differ from active codes, which count every code selected at least once.

Method	Active / 20	H_{norm}	Top-1 Ratio	Soft MI	Skills / Instruction	SR (%)
LISA	4	0.451	0.344	0.015	3.60	19.08
A: Embedding + Shared horizon	20	0.942	0.095	0.244	8.92	28.34
B: Embedding + Independent I, H	19	0.866	0.119	0.254	5.21	38.26
C: Embedding + Shared horizon + Diffusion	20	0.937	0.096	0.249	8.67	32.08
DASL	19	0.930	0.104	0.269	6.13	58.49
DASL w/o IS	15	0.759	0.296	0.254	6.15	43.12

Relative to LISA, DASL increases active-code count from 4 to 19, normalized entropy from 0.451 to 0.930, and Soft MI from 0.015 to 0.269, while reducing the Top-1 usage ratio from 0.344 to 0.104. Variant A activates all 20 codes and reaches 28.34% SR. Independent temporal parameters in Variant B increase SR to 38.26% and reduce the skills-per-instruction summary from 8.92 to 5.21. Adding latent diffusion yields DASL at 58.49% SR, with 19 active codes, 0.930 entropy, and 0.269 Soft MI. Under shared-horizon coordination, Variant C reaches 32.08% SR and 0.249 Soft MI. Together, the comparisons show stronger task performance and language–skill association when independent temporal coordination is combined with latent diffusion.

E.7 Performance under Unseen Mid-Episode Perturbations

We evaluate unseen perturbations on LOReL-state using episodes of approximately 20 steps. At $t = 5$, the target object is displaced opposite to the instructed direction; the policy then continues without retraining, with the fast policy observing the current state at every action step and the skill planner selecting a new skill every $I = 2$ steps. Such mid-episode displacements are absent from the training demonstrations. Each perturbed rollout is paired with an unperturbed rollout using the same initial state and random seed. The tasks involve constrained translation (drawers), constrained rotation (faucets), and planar translation (mugs). Perturbation magnitude is normalized by the task success threshold τ, with $\delta = 0.5\tau$ for every task. Table 17 gives the physical magnitudes; Table 18 reports task-level performance. Relative retention is the perturbed SR divided by the unperturbed SR, rounded to whole percentages.

Table 17: Perturbation sizes applied at $t = 5$, opposite to the instruction.

Task	Success threshold τ	Perturbation δ	δ/τ
close drawer	0.02 m	0.01 m	0.50
open drawer	0.02 m	0.01 m	0.50
turn faucet left	$\pi/10 \approx 0.314$ rad	0.157 rad	0.50
turn faucet right	$\pi/10 \approx 0.314$ rad	0.157 rad	0.50
move black mug right	0.02 m	0.01 m	0.50
move white mug down	0.02 m	0.01 m	0.50

Table 18: LOReL-state task success under unseen perturbations. Δ is perturbed minus unperturbed SR, in percentage points; retention is their ratio.

Task	Unperturbed SR (%)	Perturbed SR (%)	Δ (pp)	Retention
close drawer	96.0	95.0	-1.0	99%
open drawer	44.0	15.0	-29.0	34%
turn faucet left	68.0	67.7	-0.3	100%
turn faucet right	43.0	32.8	-10.2	76%
move black mug right	55.0	46.9	-8.1	85%
move white mug down	11.0	10.4	-0.6	95%
Task average	53.0	44.6	-8.4	84%

Across the six LOReL-state tasks, the mean success rate is 44.6% under unseen mid-episode perturbations, compared with 53.0% for matched unperturbed rollouts. Task-level perturbed/unperturbed results are 95.0%/96.0% for closing a drawer, 15.0%/44.0% for opening a drawer, 67.7%/68.0% for turning a faucet left, 32.8%/43.0% for turning a faucet right, 46.9%/55.0% for moving a black mug right, and 10.4%/11.0% for moving a white mug down. During these trials, the fast policy receives the updated observation at every action step, and the skill planner selects a new skill every $I = 2$ steps.

F Additional Ablation Studies

F.1 Sensitivity Analysis of Frequency Ratios

Table 19 compares four semantic update intervals on LOReL. The 1:2 ratio gives the highest average SR in both modalities: 54.51% with state inputs and 50.96% with image inputs. The corresponding results are 37.05%/25.75% for 1:1, 33.18%/22.59% for 1:4, and 30.51%/19.55% for 1:8. The two-step semantic update schedule therefore provides the strongest task performance among the evaluated frequency ratios.

Table 19: Performance comparison under different frequency ratios for state and image observations.

Task Instruction	Obs	1:1	1:2(ours)	1:4	1:8
close drawer	State	70.77	96.92	78.46	86.15
	Image	13.85	64.00	30.77	26.15
open drawer	State	30.00	31.67	28.33	0
	Image	8.33	58.67	41.67	5.00
turn faucet left	State	53.85	66.15	41.53	80
	Image	35.38	41.69	13.85	58.46
turn faucet right	State	1.54	52.31	16.92	0
	Image	18.46	41.69	4.62	0.00
move black mug right	State	43.08	66.15	32.31	0
	Image	50.77	70.46	23.08	16.92
move white mug down	State	23.08	13.85	1.54	16.92
	Image	27.69	29.23	21.54	10.77
Average	State	37.05	54.51	33.18	30.51
	Image	25.75	50.96	22.59	19.55

F.2 Additional Ablation on Action Generation Architectures

This comparison replaces the deterministic Decision Transformer action head with a diffusion-based action generator while retaining the latent diffusion module for trajectory regularization. Table 20 reports success rates of 54.51%/50.96% for the Decision Transformer variant and 27.53%/42.33% for the dual-diffusion variant on LOReL state/image inputs. In the separate inference comparison in Table 6, DASL runs at 13.61 Hz for state inputs, while SkillDiffuser runs at 2.32 Hz. These measurements report task success and inference frequency for the evaluated configurations.

Table 20: Success rate comparison across different architectures.

Architecture	LOReL (State)	LOReL (Obs)
DT w/ Diffusion (Ours)	**54.51**	**50.96**
Dual-Diffusion (Pure Generative)	27.53	42.33

F.3 Additional Ablation on Latent Space Regularization

This section investigates the impact of latent space regularization by comparing the proposed diffusion module against parameter-matched baselines, as detailed in Table 21. Empirical results indicate that introducing basic regularization via an MLP Autoencoder or a VAE yields limited improvements over the unregularized baseline (38.26%). In contrast, the Latent Diffusion module achieves a substantially higher success rate of 54.51% under an identical parameter budget. These findings indicate that regularizing the latent space is inherently beneficial for policy learning. Furthermore, the performance discrepancy across parameter-matched models suggests that the performance leap stems not from increased model capacity, but from the diffusion formulation's specific capability to effectively model complex, multimodal trajectory distributions.

Table 21: Success rate comparison of parameter-matched regularization methods on LOReL (State).

Regularization Method	LOReL (State)
None (Baseline)	38.26
VAE (KL Regularization)	41.30
MLP Autoencoder	44.15
Latent Diffusion (Ours)	**54.51**

F.4 Additional Ablation on Skill Discretization

This section compares discrete and continuous skill representations. Vector quantization maps continuous skill embeddings to a finite vocabulary that can be reused across task compositions. In Table 22, discrete skills achieve 54.51% success on LOReL state-based tasks, an improvement of 22.56 percentage points over continuous skills (31.95%). On BabyAI GoToSeq, continuous skills achieve 65.26%, compared with 63.93% for discrete skills. The results show that the relative performance of the two representations varies across these task settings.

Table 22: Performance comparison between continuous and discrete skill spaces.

Skill Space	**LOReL (State)**	**BabyAI GoToSeq 1k**
Continuous	31.95	**65.26**
Discrete VQ (Ours)	**54.51**	63.93

F.5 Additional Ablation on the Skill Codebook Size

Table 23 compares codebook sizes $K \in \{5, 10, 20, 40\}$ on LOReL. The 20-code setting achieves the highest success rates in both modalities, reaching 54.51% with state inputs and 50.96% with image inputs. The corresponding state/image results are 47.53%/36.36% for $K = 5$, 45.97%/32.58% for $K = 10$, and 34.80%/37.40% for $K = 40$. LOReL contains six physical tasks with many language rephrasals; sequences of skill codes allow a compact library to compose behaviors across instructions. For skill dimension d, exhaustive VQ lookup and codebook storage scale as $O(Kd)$. Validation success, active-code count, and usage entropy provide complementary signals for selecting codebook capacity.

Table 23: Performance comparison across different skill codebook sizes (K) on LOReL tasks.

Codebook Size (K)	**LOReL (State)**	**LOReL (Obs)**
5	47.53	36.36
10	45.97	32.58
20 (Ours)	**54.51**	**50.96**
40	34.80	37.40

F.6 Additional Ablation on Temporal Embeddings and OOD Horizons

This section investigates the role of temporal embeddings and the model's robustness to out-of-distribution (OOD) episode lengths. A potential concern when utilizing temporal embeddings is the risk of the policy overfitting to specific time intervals observed during training, leading to brittle performance when evaluated over longer horizons. To systematically assess this, empirical evaluations were conducted on the LOReL state-based benchmark using extended episode horizons up to 3.0× the maximum training length ($T_{\max}$). The analysis compares four distinct temporal extrapolation strategies to process the time step t:

- **Clamp (Ours):** $t = \min(t, T_{\max})$. This strategy caps the temporal signal at the maximum training length, signaling to the model that the task has entered a sustained terminal phase.
- **Modular:** $t = t \pmod{T_{\max}}$. This approach cyclically resets the temporal signal based on the original training horizon.
- **Relative:** $t = t \times \frac{T_{\max}}{\text{horizon}}$. This method proportionally scales the temporal signal relative to the new, extended evaluation horizon, stretching the embeddings to match the original distribution.
- **Zero:** $t = 0$. This serves as a pure ablation baseline by completely removing the temporal signal.

Table 24 compares temporal-conditioning strategies across extended horizons. Clamp achieves the highest success rate at each evaluated horizon, increasing from 61.9% at the in-distribution horizon

to 69.3% at 3.0×. Zero, which removes the temporal signal, yields lower success rates across all three horizons. These results show that temporal conditioning contributes to performance under the evaluated horizon changes, with the Clamp strategy performing consistently across the tested lengths.

Table 24: Performance evaluation on LOReL (State) across out-of-distribution (OOD) episode horizons using different temporal embedding extrapolation strategies. Success rates (SR (%)) are reported.

Episode Length	Clamp (Ours)	Modular	Relative	Zero
1.0× (In-dist)	**61.9**	60.2	57.6	40.7
2.0× (OOD)	**66.7**	62.8	54.5	38.1
3.0× (OOD)	**69.3**	64.9	63.2	45.9

F.7 Isolating Interval-Level Skill Supervision

The ablation preserves DASL's forward skill sharing, update interval, model capacity, and full-rate action loss for the fast policy. Only the action supervision reaching the slow skill branch is restricted to a single fast step per interval. Table 16 places this comparison alongside the component variants using the same fixed-seed LOReL-state protocol. Equation 6 defines the action supervision accumulated over the interval for the shared skill. Restricting slow-branch supervision reduces state SR by 15.37 percentage points and concentrates code use: active entries decline from 19 to 15, entropy from 0.930 to 0.759, and the Top-1 ratio rises from 0.104 to 0.296. Image SR decreases from 50.66% to 40.37%. The instruction-level skill-count summary changes little (6.13 to 6.15), emphasizing the complementary information in the codebook diagnostics. **Matched configurations.** On LOReL, DASL uses $I = 2$ with $H = 5$ for state and $H = 10$ for image observations. The reproduced LISA-state configuration uses $I = H = 10$. Diffusion-enabled internal component variants use the same modality-specific weight as DASL: $\lambda_{\text{diff}} = 0.5$ for state and 0.01 for image. Thus the cross-modality weight difference is retained explicitly, while the within-modality comparison changes only the components under study. The codebook diagnostics are reported for state observations.

G Real-world Experiment

To evaluate the practical viability of DASL in real-world scenarios, we deploy the framework in a physical setup equipped with a single-arm Realman manipulator and a statically mounted third-person RealSense RGB camera. As illustrated in Figure 18, all models are trained and evaluated using combined state and image inputs. We design five distinct manipulation tasks, categorized into fundamental atomic skills and complex compositional tasks. The atomic tasks are *open drawer*, *pick up the banana from the plate*, and *close drawer*; we collect 50 demonstration trajectories for each task. The compositional tasks are *open drawer and close drawer* and *open drawer and pick up banana from the plate*; we collect 20 demonstration trajectories for each task.

We evaluate each task in 20 trials. Table 25 reports success rates for three atomic and two compositional tasks; DASL achieves an average of 88%, compared with 56% for LISA.

Table 25: **Success rates of real-world robot experiments.** We evaluate LISA and DASL on a Realman robot across five manipulation tasks, categorized into atomic and compositional instructions. The results report the success rate (%) over 20 trials for each task. The best results are highlighted in **bold**.

Task Instruction	LISA	DASL (ours)
close drawer	70	**95**
open drawer	55	**90**
pick up the banana from the plate	60	**100**
open drawer and close drawer	50	**75**
open drawer and pick up banana from the plate	45	**80**
Average	56	**88**

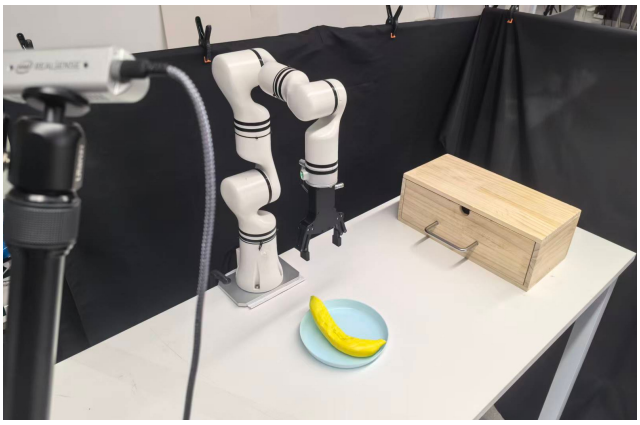


Figure 18: **Real-world experimental setup.** We evaluate the proposed DASL framework on a Realman robot. The setup involves various interactive objects, such as a drawer and a banana on a plate, to validate the performance of the model across atomic and composite manipulation tasks.

Figure 20 shows DASL execution sequences for the atomic and compositional tasks. Figure 19 presents illustrative rollouts for two compositional instructions. In the displayed LISA examples, the drawer remains partially closed in (a), and the banana grasp is unstable in (c). The corresponding DASL examples show completed drawer closing in (b) and a maintained grasp in (d). The task-level success rates are reported in Table 25.

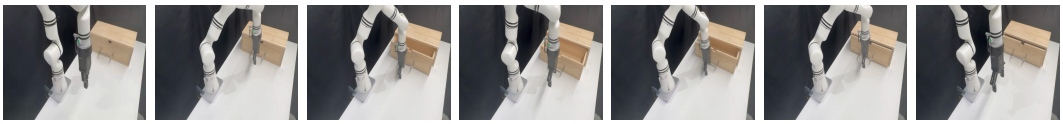
H Additional Deployment Analyses

H.1 Inference Latency Breakdown

The 13.61 Hz state result in Table 6 corresponds to an unoptimized Eager-FP32 pipeline with 73.48 ms end-to-end latency. The full policy stack comprises history preprocessing, option selection, skill–observation fusion, Decision Transformer prediction, and action postprocessing. DistilBERT is evaluated when the instruction changes, with its cost amortized across the episode. The inference path uses no diffusion denoising. Table 26 reports the measured component costs.

Table 26: Unoptimized DASL state inference latency decomposition. Values are rounded; the diffusion module incurs no rollout cost.

Component	Latency (ms)	Proportion (%)
Option selector	31.817	43.3
Decision Transformer	17.635	24.0
Policy-framework overhead	9.405	12.8
Skill–observation fusion	8.597	11.7
Preprocessing	3.894	5.3
DistilBERT (amortized)	1.910	2.6
Action head	0.220	0.3
Total	73.480	100.0



(a) LISA Baseline: open drawer and close drawer



(b) DASL (Ours): open drawer and close drawer



(c) LISA Baseline: open drawer and pick up banana from the plate



(d) DASL (Ours): open drawer and pick up banana from the plate

Figure 19: Qualitative comparison on compositional tasks. The illustrated LISA rollouts show incomplete drawer closing (a) and an unstable grasp (c); the DASL rollouts show successful execution and skill transitions in the corresponding examples (b, d).



(a) Atomic Task: close drawer



(b) Atomic Task: open drawer



(c) Atomic Task: pick up the banana from the plate



(d) Compositional Task: open drawer and close drawer



(e) Compositional Task: open drawer and pick up banana from the plate

Figure 20: Qualitative visualizations of the proposed DASL framework executing atomic and compositional tasks on a Realman robot. Each row shows a key-frame sequence of a successful trial.

H.2 Progressive Deployment Optimizations

Starting from the 73.48 ms Eager-FP32 baseline, we profile progressive deployment optimizations (Table 27). Stage 1 uses a fixed-length preallocated history buffer with the same context as the sliding window, avoiding repeated concatenation and allocation. Stage 2 captures the Decision Transformer with a CUDA Graph to reduce kernel-launch and framework overhead. Stage 3 also captures the option selector, covering the primary GPU inference paths.

Table 27: Progressive full-policy-stack inference optimizations for DASL. Speedup is relative to the Eager-FP32 baseline.

Stage	Configuration	Latency (ms)	Frequency (Hz)	Speedup
0	Eager FP32	73.48	13.61	1.00×
1	+ Bounded history buffer	26.34	37.97	2.79×
2	+ Decision Transformer CUDA Graph	15.35	65.13	4.79×
3	+ Option selector CUDA Graph	8.029	124.55	9.15×

Progressive deployment optimization reduces full-policy latency from 73.48 to 8.029 ms and increases inference frequency from 13.61 to 124.55 Hz, a 9.15× speedup. The bounded history buffer and CUDA Graphs deliver this acceleration on top of DASL's deterministic inference path, which requires no iterative diffusion sampling.